SETAP Database Collection, Validation, Processing and Documentation

N. Miglani, M. Sosnick, D. Petkovic
San Francisco State University
May 2016

Abstract

This report covers full details and documentation about SETAP (Software Engineering Teamwork Assessment and Prediction) project led by SFSU in collaboration with Fulda University, Germany, and FAU Florida. It is intended for machine learning researchers who want to understand how SETAP data is collected, what they are, what is the infrastructure used, and finally how the data is organized and disseminated. Actual SETSAP ML data are downloadable from SETAP project WWW site at http://setapproject.org/.

Citation: N. Miglani, M. Sosnick, D. Petkovic: ”SETAP Database Collection, Validation, Processing and Documentation”, SFSU CS Department Technical Report TR-16.02, May 2016

1. Introduction to SETAP ML Database

SETAP stands for Software Engineering Teamwork Assessment and Prediction Using Machine Learning. It is an analytical approach to assess and most importantly predict learning outcomes in SE teamwork based on data from joint software engineering class concurrently taught at San Francisco State University (SFSU), Florida Atlantic University (FAU) and Fulda University, Germany (Fulda) from Fall 2012 to Fall 2015. These classes constitute 74 student teams of over 350 students. The data collected is quantitative and objective information of students' teamwork performance while they are working on team class project in a software engineering class.

Initially, data of individual student activity is collected from various raw data sources like Weekly Timecard Surveys (WTS), Team Placement Surveys (TPS) etc. These data sources are known as SAMs – Student Activity Measures which are aggregated to form Team Activity Measures, TAMs. TAMs for each team are paired with two Machine Learning class labels, one
for software engineering Process (A or F) and one for software engineering Product (A or F) to form Machine Learning training database.

1.1. Problem Statement
Data collection process for SETAP is complex. Multiple conversions and aggregations have been applied to data to convert it from individual student activity measures to team activity measures which are then coupled with SE product and process outcomes to form ML training database. Due to complexity of the process, it becomes very important to document it and also to obtain accurate and reliable results from ML analysis, it becomes important to clean and test the data.

1.2. Project Goals
The main goals of this project are: a) Organize SETAP machine learning training database. b) Clean and verify the contents of ML training database. c) Perform testing of various components like data collection and aggregation. d) Document the data, data collection and aggregation process and finally e) upload the ML training data on SETAP WWW site for dissemination to other researchers.

1.3. Our Contributions
To achieve the previously identified goals, this project documents complete process of collecting data from raw data sources (SAMs) and aggregating it to Team Activity Measures (TAMs). It also covers various types of data testing and curation strategies that were adopted to ensure accuracy and validity of data. This was accomplished by three ways: a) testing of all data gathering, aggregation and extraction software with real and synthetic data; b) manual spot checking of data by two independent researchers; c) dealing with NULL or missing data in appropriate ways (some records are dropped, some are handled by appropriate statistics and some are imputed based on specific ways variables were extracted). Finally, it explains in detail the data processing pipeline workflow, CSV files and their headers that were generated for disseminating training database to other researchers.
2. SETAP Overview

SETAP data are obtained from a joint software engineering class taught concurrently at SFSU, Fulda and FAU, where student teams at all three schools are “embedded and observed” in as realistic project and teamwork development environment as possible, thus providing a rich learning environment for students and more realistic data for researchers. The class now involves about 140 students each year, working in 25-30 teams of 5-6 students each. Local student teams are composed of students from the same university, and global student teams are composed of volunteer students from multiple usually two universities.

2.1. Term Project and Grading

During the class, all student teams develop the same web application, with mandatory use of a suite of modern SE development and communications tools. Starting with only a single page, high-level description of the product, student teams develop their application in five well-defined milestones:

1. M1: high level requirements
2. M2: detailed requirements and specification
3. M3: prototype development and review
4. M4: beta release
5. M5: final delivery and demo

Student team composition may be local (comprising students from the same school) or global (comprising three students from each of SFSU and FAU or SFSU and Fulda schools). Teams meet weekly in class for a mandatory meeting with instructors, and independent team meetings outside of class. Teams (especially global teams) may use Skype or Google Chat for the independent meetings. Student teams are observed by instructors during the class meeting, and are graded for the components of Software Engineering product and Software Engineering process as described in more details in Table 1.
Table 1: Process and Product Grading Rubrics used in Determining Outcomes

To ensure teamwork culture and student commitment, all members of a student team share the same grade for the process and product components, which each contribute 25% to the student's overall class grade. To ensure that students quickly learn the class SE tools, an individual milestone, M0, is instituted early in the class, requiring students to install and learn these tools through the development of a small example application. M0 is worth 5% of a student's grade. A comprehensive final exam testing students’ knowledge of class material contributes the remaining 45% to a student's grade.

Great effort has been made to smoothly integrate the research and its related data collection with the SE class teaching and grading. Class teaching is “just in time” e.g. teaching topics are offered at the time when students need them for project milestones. At the end of each class for about one hour the instructors meet with student teams where they observe, advise and record their observations in instructors’ observation log (IO). A weekly time card survey (WTS) is used to collect “time spent” information from students (e.g. time spent on meetings, coding, documentation). It is made very clear to all students that no information collected or derived from this research influences student grade. Students are given the choice to participate in the research study or not, and those who choose to participate in the study sign informed consent documents. Student privacy is strictly observed and analysis is done and published only at the aggregated team level from team activity measurements (TAM). The SE process
component of the outcomes of student teamwork learning is graded by instructors reviewing observation logs and student project documentation, using the rubric in Table 1. to evaluate proper adherence to SE processes by the student team and its members. The SE product component of the learning outcomes is graded both by instructors and by independent observers, who use the rubric in Table 1. to evaluate the quality of the team's final product. Each team receives an absolute score in points, and is also ranked relative to the other teams in the same class for that semester. Following grading and ranking, for the purpose of this research, the instructors classify each team's SE process and SE product achievements into two ML classes: at or above expectations receives class label A, and below expectations or needing attention receives a class label F.

2.2. Team Organization

In order to focus our analysis only on factors influencing team success exhibited during the class and minimize the influence of an individual student’s experience and skills developed prior to the class, it is critical to form student teams with approximately the same overall distribution (mix) of skills and experience. We have developed, and recently improved and formalized, the following process for student team selection:

- A Team Placement Survey (TPS) is administered to all students at the start of the class. In this survey, student is asked about their prior product development and teamwork experience, GPA, gender, etc. The student is asked to self-rate on a scale their proficiency in various programming languages used in the class. Finally, it includes 3 simple programming proficiency tests.

- Each Team Placement Survey is rated by the weighted sum of responses to questions and instructor grading of the programming tests to determine student skill scores for each student.

- Teams are formed such that team skill scores, obtained by averaging student skill scores are approximately equal.

- Global teams are formed primarily from students who have volunteered to be on a global team.

- Each team is asked to recommend a team lead, who is evaluated and must be approved by instructors. Global teams have a team lead in each participating school.
2.3. Data Collection and Processing Workflow

Figure 1: SETAP data flow diagram

SETAP data is collected and processed in two phases:

1. **Data Collection**
   
   Data that records activity of each student and are obtained by various raw data sources like Weekly Timecard Surveys (WTS), Tool Logs (TL). This data is called as SAM – Student Activity Measures. This data is stored in tables like `student`, `studentConfidential`, in master MySQL database with the help of custom conversion and extraction scripts.

2. **Data Processing**
   
   Team activity measurements (TAM) are formed by aggregating individual student activity measurements (SAM). This is done by applying custom data extraction scripts and SQL procedures. TAM data for each student team is paired with separate evaluations of software engineering teamwork learning outcomes, one for software engineering process and one for software engineering product to form ML training database.

2.4. **Raw Data Sources – Student Activity Measures (SAMs)**

The data derived from raw data based on individual student behavior are termed Student Activity Measures (SAM). They record the activity of each student and are obtained by various sources such as weekly online surveys and analysis of usage of software engineering tools. These are generally quantitative measures, such as time used for certain activity, counts of SVN commits, teamwork incidents, etc., which are either measured by automated tools or observed and
recorded by instructors. This raw data is stored to various tables like studentConfidential, selectionSurvey in the swe database on the server. The swe database is a MySQL database that holds all the confidential information related to activities to individual student. Since machine learning analysis is done on team data, this raw data is converted to Team data or Team Activity Measures (TAM) with the help of custom scripts or SQL functions. Following is a detailed description of the SAM sources currently in use:

1. **Team Placement Surveys (TPS)**
   This is initial survey taken by the class to determine the team composition. The survey is given on paper to students at the beginning of the semester, and their responses are manually entered into a spreadsheet. The goal is to create teams with roughly the same skill set on average.

2. **Weekly Timecard Surveys (WTS)**
   These are online surveys given to students at the end of every week. They are mandatory and require students to fill in hours they spent that week in teamwork related activities like Meeting hours in person Time spent on coding or non-coding deliverables etc.

3. **Tool Logs (TL)**
   Tool Logs comprise the collected statistics of individual student usage of software engineering communication and development tools such as code repository

4. **Instructor Observation (IO)**
   Every week at the end of the class, Instructors do weekly SCRUM with team where they log team activity such as team member participation, the number of issues requiring instructor intervention, number and percent of issues closed late, etc.

5. **Project Evaluation Worksheets**
   Evaluation worksheets used to obtain student team outcomes for Software Engineering Product and Process where Product measures how well the team applied best software engineering practices and Process measures quality of the finished product produced by team. There are three types of worksheets Evaluator product grading worksheets, Instructor product grading worksheets and Instructor process grading worksheets.
2.5. SETAP Data Collection on Team Activity – Team Activity Measures (TAMs)

Team activity measurements (TAM) are formed by aggregating individual student activity measurements (SAM). The design of TAM and SAM was motivated by the real-world experience and intuition gained from teaching joint SE class for several years, trying to formalize and understand how to better assess and predict student teamwork learning. The challenge was to choose only objective and quantifiable measurements (e.g. time spent, counts of events/issues, tool usage like e-mails, postings, etc.) suitable to more advanced data analysis techniques and those motivated by our teaching and teamwork evaluation experience. For example, it was observed that teams who struggle to establish communication early on tend to fail more often, so we measure time spent in meetings and collect statistics about e-mail usage. We observed that teams writing poor software repository commit messages, such as messages that are empty or repeated, tend to produce a lower quality software product, so we measure the percent of unique commit messages to the code repository. To measure dynamics in time and within the team, we compute standard deviation of certain measures such repository commits over time in weeks, and over team members. These measures are intended to help reveal cases such as a student doing most of the work in a team instead of the work being evenly distributed among the team members. We have also conjectured that TAM predictive importance and student team dynamics may change during the course of project development (i.e. in each milestone), hence we record TAM data separately in each time interval corresponding to five project milestones.

2.6. Summary of SETAP Data collected from Fall 2012 to Fall 2015

In the period of seven semesters we collected data from fall 2012 through fall 2015.

We have collected the following data:

<table>
<thead>
<tr>
<th>General Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of semesters</td>
</tr>
<tr>
<td>First semester</td>
</tr>
<tr>
<td>Last semester</td>
</tr>
<tr>
<td>Number of students</td>
</tr>
<tr>
<td>Class sections</td>
</tr>
<tr>
<td>Number of TAM features</td>
</tr>
<tr>
<td>Number of class labels (outcomes)</td>
</tr>
</tbody>
</table>

Table 2: Displays general stats of data collected in all the semesters
<table>
<thead>
<tr>
<th>Team Composition Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Teams</td>
</tr>
<tr>
<td>Global Teams</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>

Table 3: Total number global and local teams

<table>
<thead>
<tr>
<th>Outcome Classification Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Outcomes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outcome</strong></td>
<td>A</td>
</tr>
<tr>
<td>T1 – T5</td>
<td>290</td>
</tr>
<tr>
<td>T6 – T11</td>
<td>348</td>
</tr>
<tr>
<td>Total</td>
<td>638</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outcome</strong></td>
<td>A(1)</td>
</tr>
<tr>
<td>T1 – T5</td>
<td>255</td>
</tr>
<tr>
<td>T6 – T11</td>
<td>306</td>
</tr>
<tr>
<td>Total</td>
<td>561</td>
</tr>
</tbody>
</table>

Table 4: Product and Process stats for outcomes A and F in all Time Intervals

So Summarizing, the current training data is collected from 74 student teams from Fall 2012 through Fall 2015 from joint software engineering classes. This data involves 383 Students and 18 class sections. For each team 115 TAM measures have been aggregated from related SAM measures. Total number of grades for software engineering Process were 49 As and 25 Fs, and for software engineering Product 42 As and 32 Fs. For each team about 400 data items have been collected (student team selection survey, time cards, deliverable tracking, grading of outcomes etc.), hence training DB involves about 30000 data points.
3. Time Intervals

3.1. Milestones

Milestones are major deliverable points in the class. Each Milestone represents an important stage of software engineering class. The class is divided into 6 milestones.

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milestone 0</td>
<td>Individual work to learn Team and collaborative tools. (Not used in ML training)</td>
</tr>
<tr>
<td>Milestone 1</td>
<td>High Level Specs, UI specs and Use cases</td>
</tr>
<tr>
<td>Milestone 2</td>
<td>More detailed specs</td>
</tr>
<tr>
<td></td>
<td>Fall sections only: Milestone 2a (SFSU-Fulda): Hire Fulda team, bring them up to speed, develop plans and interfaces</td>
</tr>
<tr>
<td>Milestone 3</td>
<td>First prototype and feedback; final commitment (meeting with instructors)</td>
</tr>
<tr>
<td>Milestone 4</td>
<td>Beta delivery, QA plans and usability feedback, final functionality check</td>
</tr>
<tr>
<td>Milestone 5</td>
<td>Final Project demo and delivery – week after individual exam (during final exam week)</td>
</tr>
</tbody>
</table>
Time Intervals correspond to five predefined Milestones (Milestone 1 to 5, Milestone 0 is individual work and not included in time intervals) but can be aggregated across multiple milestones.

<table>
<thead>
<tr>
<th>Time Interval</th>
<th>Corresponding Weeks in Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Milestone 0</td>
</tr>
<tr>
<td>1</td>
<td>Milestone 1</td>
</tr>
<tr>
<td>2</td>
<td>Milestone 2</td>
</tr>
<tr>
<td>3</td>
<td>Milestone 3</td>
</tr>
<tr>
<td>4</td>
<td>Milestone 4</td>
</tr>
<tr>
<td>5</td>
<td>Milestone 5</td>
</tr>
<tr>
<td>6</td>
<td>Milestone 1 - Milestone 2 (inclusive)</td>
</tr>
<tr>
<td>7</td>
<td>Milestone 1 - Milestone 3 (inclusive)</td>
</tr>
<tr>
<td>8</td>
<td>Milestone 1 - Milestone 4 (inclusive)</td>
</tr>
<tr>
<td>9</td>
<td>Milestone 1 - Milestone 5 (inclusive)</td>
</tr>
<tr>
<td>10</td>
<td>Milestone 4 - Milestone 5 (inclusive)</td>
</tr>
<tr>
<td>11</td>
<td>Milestone 3 - Milestone 5 (inclusive)</td>
</tr>
</tbody>
</table>

Table 5: Time Interval and Milestones Mapping

All time intervals are used individually for training against the semester outcome. T1-T5 attempt to identify A/F rules in a specific milestone. T6-T11 attempts to identify A/F rules for behavior that occurs across multiple milestones (trends). Figure 2 shows the logical view of SETAP ML training database for each time interval.

Figure 2: logical view of SETAP ML training database for each time interval
4. SETAP Data Collection Components

The data derived from raw data based on individual student behavior are termed Student Activity Measures (SAM). They record the activity of each student and are obtained by various sources such as weekly online surveys and analysis of usage of software engineering tools. These are generally quantitative measures, such as time used for certain activity, counts of SVN commits, teamwork incidents, etc., which are either measured by automated tools or observed and recorded by instructors. This raw data is stored to various tables like `studentConfidential`, `selectionSurvey` in the `swe` database on the server. The `swe` database is a MySQL database that holds all the confidential information related to activities to individual student. Since machine learning analysis is done on team data, this raw data is converted to Team data or Team Activity Measures (TAM) with the help of custom scripts or SQL functions. Following is a detailed description of the SAM sources currently in use, and documentation of sources that were previously but are no longer in use.

4.1. SAM Currently in Use

1. Team Placement Survey

The team placement survey is the initial survey taken by students in class, which is used by the instructors to assign the student to a team. The goal is to create teams with roughly the same skill set on average. An algorithm (Table 6), based on round-robin selection, has been developed for this process. The survey has 17 questions of various types. There are short answer, fill in the blank, selection, Likert scale, and yes/no questions, and a small programming proficiency test (Figure 3). The survey is given on paper to students at the beginning of the semester, and their responses are manually entered into a spreadsheet. The spreadsheet is then used to organize students into teams (see Figure 3). The spreadsheet is converted to a CSV (comma separated value) format file, and uploaded database using a custom PHP script located on the host system. New shell accounts are created for the students using their school email address before the @. By inserting the student information into a SQL record, the student is then assigned unique student identification (`studentId`) which is used throughout the system to identify a particular student in a particular semester. The Student name specified by the students on the paper is matched by `studentId` provided by the school roster.

A number of tables store the survey information on the system. The table `student` that identifies student by its record identification number (`studentId`) and contains information such as gender, team number; `studentConfidential` contains uniquely identifying student information from the survey and other sources, such as name, email address, etc. and the `selectionSurvey` table that holds the responses of the team selection survey.
1.1. Team Selection Process

1. Students take the surveys.
2. Programming questions evaluated.
3. Surveys entered into excel.
4. Averages calculated of the programming question scores
5. List sorted by average score on quizzes.
6. Top student assigned to first team, continue assigning students round-robin style to all teams.
7. All teams are manually evaluated, and students manually switched from teams with the goal that each team has at least one experienced programmer, or graduate student, and teams have a similar average of the self-reported skills scores.

Table 6: Team Selection Algorithm

<table>
<thead>
<tr>
<th>Programming Questions</th>
<th>Required</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code for the blog with HTML and PHP</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Code for the blog with MySQL and PHP</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Experience with frameworks such as Bootstrap</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Experience with frameworks such as jQuery</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6 illustrates the team selection algorithm. After the students have taken the team selection survey (Appendix A001: Team Placement Survey) and it has been entered into a spreadsheet, the students are ranked highest to lowest in order of the average of their scores to the three quiz questions on PHP, CSS and HTML. A round-robin selection (1) is then done from top to bottom with one student being selected for each team. The next round-robin selection is done from bottom to top, and it continues alternating until all the students are sorted into teams. The team averages are then analyzed for experience with frameworks such as Bootstrap and jQuery, and some minor team placement adjustment will take place. Once the students have been placed in their team, they are not allowed to move to other teams.
2. Weekly Timecard Surveys:

Weekly timecard surveys are online surveys given to students at the end of every week. These are mandatory surveys which require students to fill in information about the hours they spent that week in teamwork related activities like meeting hours in person or online, time spent on coding and non-coding deliverables etc. Figure 4 shows first page of Timecard Survey.

Initially (ca. fall 2012) these surveys were conducted using the open source LimeSurvey tool. But after 2013, LimeSurvey was discarded and instead Google Forms (2) were adopted. This was done because it was extremely difficult and complex to integrate data from LimeSurvey to the SETAP database. Google Forms on the other hand, stores data directly in Google spreadsheets which are downloaded and converted to CSV files. Using Excel, student usernames are matched to the usernames provided by the student in the survey, thus connecting the studentId to the survey record. This data is then uploaded to the timecardSurvey table in the SETAP database.

![Figure 4: Weekly Timecard Survey](image-url)
3. Subversion Data

SVN (3) usage data for students is retrieved from SVN logs (Figure 5) that capture activities done by them in the repository over the semester, especially during software development phase. The data is then exported to XML format, that contains information about commit messages and files modified by each student by using SVN's verbose output. This is done by typing following command on the shell:

```
svn log --xml --verbose http://sfsuswe.com/svn/syygnn
```

where s stands for semester, yy for year and nn for group number.

The problem with this data is that it contains uniquely identifiable username for each student. Therefore it is then processed by a custom Python script which is run as `./svntosetap.py`. The `svntosetap` script looks in the `svnlogs` subdirectory from the working directory for .xml files from which the team and semester information is gleaned from the user's login name to the SETAP unix shell system. The primary function of doing this is to connect username to `studentId` field of the database.

The following two tables of MySQL store the SVN information.

- **SvnLog**: Stores information such as revision number, author, date of the commit, commit message etc. Each `svnlog` entry corresponds to one revision, and is grouped together in xml files by team.

![SvnLog Table](image1.png)

- **SvnPath** stores information about the files involved like files modified while committing etc. Each entry is associated with one `svnlog` entry, so if one revision changed eight files, there would be eight `svnpath` records for that one `svnlog` entry.

![SvnPath Table](image2.png)
4. Instructor class observations

4.1. Weekly SCRUM Notes:

For about one hour at the end of each class, instructors meet individually with every student team, and do a weekly scrum or weekly meeting in which they discuss their project, what they've achieved, what they hope to achieve, and any problems preventing them from achieving it. After the scrum with a team, the instructor subjectively evaluates the effectiveness of the team and team lead by completing two Likert-scale type questions about team and team lead effectiveness. Objective data is created through notation of checkpoints and issues. These checkpoints define deliverables that the team must produce by a given time to help the team overcome any particular issue they may be having. Issues are observations by the instructor about poor team performance, or about late deliverables, etc. These observations are manually recorded by instructors in the observation log sheets (Figure 6). This data is then entered into spreadsheets and uploaded on the server.

4.2. Checkpoint Emails

For every checkpoint issue recorded in the Observation Log, an Incident email is generated to the team lead with subject line CHECKPOINT. This subject line is used to track the incident and follow-up with the team. When incident is cleared, email is sent with a new subject stating either CLOSED ON TIME or CLOSED LATE depending upon whether the issue was resolved on time. None of these emails is stored to the database. At the time of evaluation these emails are consulted to arrive at grades of process. Any unclosed checkpoints are detrimental to team grade.
5. Project Evaluation worksheets

At the end of the semester, worksheets are used to obtain student team outcomes. Two of these worksheets are used only by instructors outside of class while evaluating teams' product and process quality. There are three kinds of worksheets that record evaluation results of the team. These results are manually entered into excel and saved as CSV which are then uploaded to the server.

At the end of the semester, student teams present their product. Outcomes are determined at the end of the semester through evaluation of student team work in two categories: software engineering process (how well the team applied best software engineering practices), and software engineering product (the quality of the finished product the team produced). Thus for each team, two outcomes are determined, process and product, respectively. Outcomes are classified into two class grades, A or F. A represents teams that are at or above expectations, F represents teams that are below expectations or need attention. For more information, please visit setapproject.org. The process and product evaluations produce a percentage outcome result. To obtain an individual student semester grade, the process and product percentage are combined with the student's individual performance on quizzes and tests in class.
5.1. Evaluator Worksheets:

This worksheet is completed by instructors and evaluators during class presentations on the last day of the semester (Figure 7). Evaluators grade the presentation on the product’s correctness and reliability and ease of use, and also evaluate the presentation’s performance and style. The average of these scores are then used in instructor Product Grading Worksheet. Since evaluators may not be cognizant of the previous or existing issues with the team product, these scores can be changed or overridden by the instructors during formal grading.

![Grading Template: Software Engineering Class Demos (V 12/14/15)](image)

Figure 7: Evaluator Product Grading Worksheet (see Appendix A0004 below)
5.2. Instructor Product Grading Worksheets:

Instructors use this sheet to obtain scores for the product component of team outcomes Figure 8. It averages the class presentation results and additionally grades the database design, architecture and code quality. For this the instructors consult the documents submitted by the students, SVN Logs and also the project source code.

![Grading Template: Software Engineering Product (V 05/17/15)](image)

**Figure 8: Product Grading Template for Instructors**
5.3. Instructor Process Grading Worksheets:

Like product grading worksheets grade product component, process grading worksheets grade process component of the outcomes Figure 9. These parameters concern all the non-software items like team participation in meetings, documentation, incidents or issues etc. Instructors consult their observation logs, project reports and also any relevant emails to produce a score.

![Figure 9: Instructor Process Grading Template](image)

6. Outcomes

The score from the process and product grading templates is computed and classified into two classes – A and F where A stands for acceptable and F for below expectations. These grades constitute “decision classes” for the ML (machine learning) algorithm and are based on a strict arbitrarily selected percentage which is same for all semesters. Though it should be noted that team class grades (final grades) derived from the results are not same as the outcomes. This decoupling helps instructors to add/subtract from team grade without impact to research data results. Currently the outcomes are stored manually into the outcomes table using the PhpMyAdmin (4) database interface.

4.2. SAM Data Sources No Longer in Use

1. Emails from Students

Data from emails was recorded until Fall 2014. Its use was abandoned as students were not reliably using sfuswe.com email server. Also many preferred to use collaboration tools like Google Docs (5), Gmail. Due to the lack of consistency of use and due to privacy concerns are not using lexical analysis of the emails, it was decided to abandon the collection of email records as they are an unreliable data source.
2. Bugzilla (6)

Plans originally included integrating Bugzilla into the curriculum, but it was never completely integrated. It was simply impossible to introduce another tool with a learning curve as steep. Its use was abandoned after Fall 2014.

3. LimeSurvey.

LimeSurvey (7) a survey tool was used for weekly Time Card surveys before Google Survey. However, though this tool was good for generalized survey, its data structure was hopelessly disjointed and confusing. It dispersed data across too many tables and they had to be renamed, merged and modified, simply to store one week worth data. This created the potential (and actuality) of errors when converting. Therefore, this tool was replaced by Google Forms

4.3. SAMs Collected for Each Team

The current training data is collected from 74 student teams from Fall 2012 through Fall 2015 from joint software engineering classes. This data involves 383 Students and 18 class sections. For each team 115 TAM measures have been aggregated from related SAM measures. Total number of grades for software engineering Process were 49 As and 25 Fs, and for software engineering Product 42 As and 32 Fs. For each team about 400 data items have been collected (student team selection survey, time cards, deliverable tracking, grading of outcomes etc.), hence training DB involves about 30000 data points.

4.4. Data Privacy

To assure strict adherence to student privacy, machine learning analysis is done and published only at the aggregated team level from team activity measurements (TAM). This is done to ensure our training database contains no individually identifiable student information.
5. TAM – Database Naming Conventions and Data Aggregation Methods

Team Activity Measures (TAM) are paired with Machine Learning class labels A and F for each software engineering Process and Product to constitute feature vectors in Machine Learning training database. Each TAM (training parameter) aggregation (means of combining SAM data into TAM data) follows a precise process described with the grammar below, and also follows a precise naming convention for easy maintenance by users. By using this systematic approach, TAM feature names are produced that are human understandable and intuitive. Further, when in question the same systematic approach can be used to determine exactly how the data were collected, combined and derived.

5.1 TAM Field Naming Conventions
For purposes of this grammar, aggregation is the process of grouping responses in a time interval. The available methods are by time interval (all the values in a time interval for that team), by student (all the values for a particular student in a time interval), or by week (all the values in a week for a particular team).

The rationale for TAM aggregated measures by week and by student was to reflect statistics of change of overall team behavior in time and also across students (e.g. was the work uniformly distributed within the team). TAMs are aggregated by time (corresponding to team project milestones) since dynamics and tool usage varies in time intervals.
This convention for aggregating SAM into TAM is:

\[ S \rightarrow n \mid ma \mid amad \mid k \mid ke \mid ake \]

where:

- **k** is a measure name without student aggregation with a lower case initial letter. Starts with capital initial letter if not at the start of the name.

- **n** is a non-aggregatable measure name

- **m** is a measure name with a lower case initial letter. Starts with capital initial letter if not at the start of the name.

- **a** is an aggregation method with a capital initial letter. Starts with a capital letter if not at the start of the name.

- **d** is an aggregation domain

- **e** is an aggregation domain excluding ByStudent domain

and:

- **k** is in \{issueCount, onTimeIssueCount, lateIssueCount, unresolvedIssueCount\}

- **n** is in \{year, semester, timeInterval, teamNumber, semesterId, teamMemberCount, femaleTeamMembersPercent, teamLeadGender, teamDistribution, processLetterGrade, productLetterGrade\}

- **m** is in \{responseCount, meetingHours, inPersonMeetingHours, nonCodingDeliverablesHours, codingDeliverablesHours, helpHours, leadAdminHours, globalAdminHours, commitMessageCount, commitMessageLength, uniqueCommitMessagePercent, resolvedIssueCount, unresolvedIssueCount\}

- **a** is in \{total, average, standardDeviation\}

- **d** is in \{<epsilon>, ByWeek, ByStudent\}
  where \(<epsilon>\) represents the empty string, and indicates the aggregation is ByTimeInterval. Since all data samples are aggregated over time interval, this is left implied.

- **e** is in \{<epsilon>, ByWeek\}
  where \(<epsilon>\) represents the empty string, and
indicates the aggregation is ByTimeInterval. Since all data samples are aggregated over time interval, this is left implied.

Some Examples

- **timeInterval** – is the time interval for which a row's data is calculated. This is a non-aggregable measure, so the $n$ derivation is chosen with $n = \text{timeInterval}$.

- **meetingHoursAverage** – is the average number of hours spent meeting by all team members in the time interval, averaged by tall the team responses in the time interval. In this case, the $ma$ derivation is chosen, where $m = \text{meetingHours}$ and $a = \text{average}$.

- **averageMeetingHoursTotalByWeek** – average of total meeting hours spent by team members per week for a time interval. So here the derivation is $am\text{ad}$ where $a = \text{average}$, $m = \text{MeetingHours}$, $a = \text{Total}$ and $d = \text{ByWeek}$.

- **averageCommitMessageLengthTotalByWeek** – average of total commit message length by weeks in time interval. So here the derivation is $akae$ where $a = \text{average}$, $k = \text{CommitMessageLength}$, $a = \text{Total}$ and $e = \text{ByWeek}$

### 5.2. TAM Table Naming Convention

Unless otherwise indicated, SETAP table names have standard names such as milestone, outcome, etc. There are two special types of tables which use the following naming convention:

**I. Temporary Tables**

$t_<$source$><tablename$>  
where $<source>$ is one of {Svn, IssueLog, Timecard} and $<tablename>$ is a descriptive name for the table.

**II. Derived Tables**

$d_<$tablename$>$  
These contain information that is gathered through counting, grouping, calculating the SD, etc. of other (usually SAM) data. These are not temporary files and should not be deleted. $d_<$tablename$>$ files are joined to create the TAM master output data file.
There are two types of derived tables, complete and derived.

A. d_timecardComplete

The original timecard survey records are copied for each time interval in which they exist, along with the percentage that they contribute for that week in the semester's time interval. SQL for d_timecardComplete could be referred in Appendix B0001 below. The following temporary tables are used to calculate d_timecardComplete:

- t_timecardTimeIntervalDate
  contains the start and end date of each time interval for every semester. There are count (time interval) * number-of-semesters records in this table.

- t_timecardWeeksInTimeInterval
  calculates to which time intervals each week belongs, and the percentage to which it belongs to that week. This is what provides the scaling for weeks that straddle time intervals.

- t_timecardSurveyPlusWeekId
  identifies each week to which the particular timecard survey entry applies

B. d_timecardDerived

This table contains all the information for the timecard surveys derived from the d_timecardComplete tables. This aggregation process conceals any possible identification of an individual student inside a team. Error! Reference source not found. SQL query for calculating TAMs from d_timecardDerived. could be referred in Appendix B0002 below. The d_timecardDerived table uses the following temporary tables in its creation:

- t_timecardResultsByTimeInterval
  This table aggregates information over the entire time interval. This is accomplished through the use of the GROUP BY MySQL function over the team and time interval. The sum, average and standard deviation over the time interval are calculated. For example, this is where the total number of meeting hours all team members have participated in over a time interval are calculated.

- t_timecardDerivedByWeek
  This table aggregates information by week per team. The GROUP BY function aggregates information by team, time interval, and week. The sum, average and standard deviation over the time interval are calculated.
• t_timecardDerivedByWeekAvSd
  This table further calculates the average and standard deviation of the weekly statistics generated by t_timecardDerivedByWeek. This table produces such statistics as the average number of commits per week in a time interval per team.

• t_timecardDerivedByStudent
  This table aggregates information by week per student in team. The GROUP BY function aggregates information by team, time interval, and student. The sum, average and standard deviation over the time interval are calculated.

• t_timecardDerivedByStudentAvSd
  This table further calculates the average and standard deviation of the statistics generated by t_timecardDerivedByStudent to generate comparative information about all students in a team. This table produces such statistics as the average number of commits per student in a time interval per team.

**Example** of calculating TAM’s for meeting hours. The whole process consists of 3 stages:

**Stage 1:** Create final table d_timecardComplete table from 3 temporary tables:
1. t_timecardTimeIntervalDate
2. t_timecardWeeksInTimeInterval
3. t_timecardSurveyPlusWeekId

**Temporary table t_timecardTimeIntervalDate:**
\[
\begin{align*}
1 & \text{CREATE TABLE `t_timecardTimeIntervalDate`} \\
2 & \text{SELECT t.*, m.`semesterId`,} \\
3 & \text{MIN(m.`start`) as start,} \\
4 & \text{MAX(m.`end`) as end} \\
5 & \text{FROM `timeInterval` t} \\
6 & \text{JOIN `Milestone` m} \\
7 & \text{WHERE m.`number` >= t.`milestoneStart`} \\
8 & \text{AND m.`number` <= t.`milestoneEnd`} \\
9 & \text{GROUP BY m.`semesterId`,t.`timeIntervalId`} \\
10 & \text{ORDER BY m.`semesterId`,t.`timeIntervalId`;}
\end{align*}
\]

**Temporary table t_timecardWeeksInTimeInterval:**
\[
\begin{align*}
1 & \text{CREATE TABLE `t_timecardWeeksInTimeInterval`} \\
2 & \text{SELECT w.`semesterId`,} \\
3 & \text{t.`timeIntervalId`,} \\
4 & \text{w.`number` as weekNumber,} \\
5 & \text{w.`weekId` as weekId,} \\
6 & \text{weekPercent(w.`start`,w.`end`,t.`start`,t.`end`) as percent,} \\
7 & \text{w.`start` weekStart, w.`end` weekEnd, t.`start`} \\
8 & \text{intervalStart,t.`end` intervalEnd FROM `week` w} \\
9 & \text{JOIN `t_timecardTimeIntervalDate` t} \\
10 & \text{WHERE weekPercent(w.`start`,w.`end`,t.`start`,t.`end`) != 0}
\end{align*}
\]
Temporary table t_timecardSurveyPlusWeekId:
1. CREATE TABLE `t_timecardSurveyPlusWeekId`
2. SELECT weekIdFromDate(s.datestamp) as weekId, s.*
3. FROM timecardSurvey s
4. WHERE s.studentId <> 0;

Final table d_timecardComplete:
1. CREATE TABLE `d_timecardComplete`
2. SELECT
3. w.weekId,
4. w.percent,
5. w.timeIntervalId,
6. s.`teamId`,
7. s.`leadStudentId`,
8. s.`timecardSurveyId`,
9. s.`studentId`,
10. s.`datestamp`,
11. s.meetingHours,
12. FROM `t_studentTeamTimecard` s
13. JOIN `t_timecardWeeksInTimeInterval` w
14. WHERE w.weekId = s.weekId
15. AND NOT s.teamNumber IS NULL;

Stage 2: Generate d_timecardDerived from d_timecardComplete table and 5 temporary tables:
1. t_timecardResultsByTimeInterval
2. t_timecardDerivedByWeek
3. t_timecardDerivedByWeekAvSd
4. t_timecardDerivedByStudent
5. t_timecardDerivedByStudentAvSd

Temporary table t_timecardResultsByTimeInterval
1. CREATE TABLE `t_timecardResultsByTimeInterval`
2. SELECT teamId, timeIntervalId,
3. SUM(meetingHours*percent) AS meetingHoursTotal,
4. SUM(meetingHours*percent)/COUNT(meetingHours) AS meetingHoursAverage,
5. STD(meetingHours) AS meetingHoursStandardDeviation,
6. FROM `d_timecardComplete`
7. GROUP BY `teamId`, `timeIntervalId`;

Temporary table t_timecardDerivedByWeek
1. CREATE TEMPORARY TABLE `t_timecardDerivedByWeek`
2. SELECT teamId, timeIntervalId, weekId,
3. SUM(meetingHours*percent) AS meetingHoursTotalByWeek,
4. SUM(meetingHours*percent)/COUNT(meetingHours) AS meetingHoursAverageByWeek,
5. FROM `d_timecardComplete`
6. GROUP BY `teamId`, `timeIntervalId`, `weekId`;
### Temporary table `t_timecardDerivedByWeekAvSd`

```sql
CREATE TEMPORARY TABLE `t_timecardDerivedByWeekAvSd`
SELECT teamId, timeIntervalId,
    AVG(meetingHoursTotalByWeek) AS averageMeetingHoursTotalByWeek,
    STD(meetingHoursTotalByWeek) AS standardDeviationMeetingHoursTotalByWeek,
    AVG(meetingHoursAverageByWeek) AS averageMeetingHoursAverageByWeek,
    STD(meetingHoursAverageByWeek) AS standardDeviationMeetingHoursAverageByWeek,
FROM `t_timecardDerivedByWeek` 
GROUP BY `teamId`, `timeIntervalId`;
```

### Temporary table `t_timecardDerivedByStudent`

```sql
CREATE TEMPORARY TABLE `t_timecardDerivedByStudent`
SELECT teamId, timeIntervalId, studentId,
    SUM(meetingHours*percent) AS meetingHoursTotalByStudent,
    SUM(meetingHours*percent)/COUNT(meetingHours) AS meetingHoursAverageByStudent,
FROM `d_timecardComplete` 
group by `teamId`, `timeIntervalId`, `studentId`;
```

### Temporary table `t_timecardDerivedByStudentAvSd`

```sql
CREATE TEMPORARY TABLE `t_timecardDerivedByStudentAvSd`
SELECT teamId, timeIntervalId,
    AVG(meetingHoursTotalByStudent) AS averageMeetingHoursTotalByStudent,
    STD(meetingHoursTotalByStudent) AS standardDeviationMeetingHoursTotalByStudent,
    AVG(meetingHoursAverageByStudent) AS averageMeetingHoursAverageByStudent,
    STD(meetingHoursAverageByStudent) AS standardDeviationMeetingHoursAverageByStudent,
FROM `t_timecardDerivedByStudent` 
group by `teamId`, `timeIntervalId`;
```

### Final Table `d_timecardDerived`

```sql
CREATE TABLE `d_timecardDerived`
SELECT v.*,
    x.`meetingHoursAverage`,
    x.`meetingHoursStandardDeviation`,
    y.`averageMeetingHoursTotalByWeek`,
    y.`standardDeviationMeetingHoursTotalByWeek`,
    y.`averageMeetingHoursAverageByWeek`,
    y.`standardDeviationMeetingHoursAverageByWeek`,
FROM `t_timecardDerivedByStudentAvSd` v
    JOIN `t_timecardResultsByTimeInterval` x
WHERE v.`teamId` = x.`teamId`
    AND v.`timeIntervalId` = x.`timeIntervalId`
    AND v.`teamId` = y.`teamId`
    AND v.`timeIntervalId` = y.`timeIntervalId`;
```
Stage 3: Generating final TAMS by joining data from d_timecardDerived and outcomes.

```sql
CREATE TABLE IF NOT EXISTS `TAM`
SELECT t.*,
    IF(ISNULL(d.`teamMemberResponseCount`), 0, d.`teamMemberResponseCount`) AS teamMemberResponseCount,
    d.`meetingHoursTotal`,
    meetingHoursAverage,
    meetingHoursStandardDeviation,
FROM `t_tamHeader` t
    LEFT JOIN `d_timecardDerived` d
    ON t.`teamNumber` = d.`teamId`
    AND t.`timeInterval` = d.`timeIntervalId`
    JOIN `outcomes` o
WHERE t.`teamNumber` = o.`teamId`;
```

5.3. TAM Aggregation Process

Creating and Aggregating TAMS involve applying number of SQL functions and procedures on various database tables. Figure 10 depicts an overview of this process.

The first step performs a join of the data source (such as SVN log data or SAM), with student, team, time period (including percentage) allowing for easy aggregation later. This data is stored in temporary tables which may be made non-temporary for purposes of debugging. These tables are referred to as "complete", for example.
\textit{d\_timecardComplete}, where d indicates derived from source data, timeCard the data source. Since time intervals 1-5 are repeated in time intervals 6 and above, the records for a particular data row can be repeated, but with different time intervals and percentages. Each of these tables are created using an SQL function. Currently there are three "complete" tables (Table 7):

<table>
<thead>
<tr>
<th>Data</th>
<th>Source Tables</th>
<th>SQL Procedure</th>
<th>Output Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suvbersion</td>
<td>svnlog, svnpath</td>
<td>createSvnComplete()</td>
<td>\textit{d_svnComplete}</td>
</tr>
<tr>
<td>Time Card Surveys</td>
<td>timecard</td>
<td>createTimecardComplete</td>
<td>\textit{d_timecardComplete}</td>
</tr>
<tr>
<td>Team Issues</td>
<td>issuelog</td>
<td>createIssuelogComplete</td>
<td>\textit{d_issuelog}</td>
</tr>
</tbody>
</table>

*Table 7: Complete tables while creating TAMs*

The TAM table is created by executing the createTAM() SQL function. This function creates the final TAM table by aggregating all the team member responses for a team in a time interval. The values in a time interval are summed, averaged, or the standard deviation is taken.

The complete data is then sub-aggregated by all the team member responses by week. This allows each week's data to be summed, averaged, or have the standard deviation taken. These "week data points" then have their average, sim and standard deviation taken. This is indicated in the TAM name as "ByWeek".

Finally, the complete data is sub-aggregated by each student in the team for a time interval. This allows each student's data to be summed, averaged, or have the standard deviation taken. These "student data points" then have their average, sim and standard deviation taken. This is indicated in the TAM name as "ByStudent".
### 5.4. List of TAMs

<table>
<thead>
<tr>
<th>TAMs</th>
<th>Aggregation Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td></td>
</tr>
<tr>
<td>semester</td>
<td></td>
</tr>
<tr>
<td>timeInterval</td>
<td></td>
</tr>
<tr>
<td>teamNumber</td>
<td></td>
</tr>
<tr>
<td>semesterId</td>
<td></td>
</tr>
<tr>
<td>teamMemberCount</td>
<td></td>
</tr>
<tr>
<td>femaleTeamMembersPercent</td>
<td></td>
</tr>
<tr>
<td>teamLeadGender</td>
<td></td>
</tr>
<tr>
<td>teamDistribution</td>
<td></td>
</tr>
<tr>
<td>teamMemberResponseCount</td>
<td></td>
</tr>
<tr>
<td>meetingHours</td>
<td>Total, Average, StandardDeviation</td>
</tr>
<tr>
<td>inPersonMeetingHours</td>
<td>Total, Average, StandardDeviation</td>
</tr>
<tr>
<td>nonCodingDeliverablesHours</td>
<td>Total, Average, StandardDeviation</td>
</tr>
<tr>
<td>codingDeliverablesHours</td>
<td>Total, Average, StandardDeviation</td>
</tr>
<tr>
<td>helpHours</td>
<td>Total, Average, StandardDeviation</td>
</tr>
<tr>
<td>leadAdminHours</td>
<td>ResponseCount, Total, Average, StandardDeviation</td>
</tr>
<tr>
<td>globalLeadAdminHours</td>
<td>ResponseCount, Total, Average, StandardDeviation</td>
</tr>
<tr>
<td>ResponsesByWeek/ByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>MeetingHoursTotalByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>MeetingHoursAverageByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>InPersonMeetingHoursTotalByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>InPersonMeetingHoursAverageByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>NonCodingDeliverablesHoursTotalByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>NonCodingDeliverablesHoursAverageByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>CodingDeliverablesHoursTotalByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>Measure</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>CodingDeliverablesHoursAverageByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>HelpHoursTotalByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>HelpHoursAverageByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>LeadAdminHoursResponseCountByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>LeadAdminHoursTotalByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>GlobalLeadAdminHoursResponseCountByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>GlobalLeadAdminHoursTotalByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>GlobalLeadAdminHoursAverageByWeek/TotalByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>commitCount</td>
<td></td>
</tr>
<tr>
<td>uniqueCommitMessageCount</td>
<td></td>
</tr>
<tr>
<td>uniqueCommitMessagePercent</td>
<td></td>
</tr>
<tr>
<td>commitMessageLength</td>
<td>Total, Average, StandardDeviation</td>
</tr>
<tr>
<td>CommitCountByWeek/ByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>UniqueCommitMessageCountByWeek/ByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>UniqueCommitMessagePercentByWeek/ByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>CommitMessageLengthTotalByWeek/ByStudent</td>
<td>Average, StandardDeviation</td>
</tr>
<tr>
<td>issueCount</td>
<td></td>
</tr>
<tr>
<td>onTimeIssueCount</td>
<td></td>
</tr>
<tr>
<td>lateIssueCount</td>
<td></td>
</tr>
<tr>
<td>processLetterGrade</td>
<td></td>
</tr>
<tr>
<td>productLetterGrade</td>
<td></td>
</tr>
</tbody>
</table>

Table 8: List of all TAMs
6. Data Curation and Cleanup

In order to ensure integrity of machine learning results, data was carefully vetted and tested for validity. Data is derived from number of data sources and also analysis is done on TAMs which are derived from SAMs. The process involves many calculations and conversions which are achieved by custom scripts and SQL functions. Presently we have data from 7 semesters from Fall 2012 to Fall 2015. We have 383 students with 18 class sections and 142 TAM features which makes total number of data points to be over 30000. Due to large amount of data and number of operations on it, the possibility of errors increases.

We curated the data in several ways:
- We performed random spot checks to test for data consistency
- We identified NULL values and their source problems such as missing entries for non-global team leads, etc.
- We imputed (interpolated) NULL values which resulted in mismatch between end of time intervals for milestones vs, times of entry for weekly time cards. See Figure 17.

6.1. NULL VALUES

NULL values are used in the training data to indicate that no SAM were recorded in that particular time period, week, or for that student. Frequently TAM features involving teamLeadHours or globalTeamLead hours will result in a NULL for a particular training sample. For local team leads, that usually means that the local team lead did not complete any timecard surveys for the aggregation in question. While for global team lead TAM features this may also be the case, the more usual cause of NULLS in global team lead TAM features comes from the fact that most teams are not global, and therefore this statistic was not gathered for these teams. It is left to the individual researcher to decide how to accommodate NULL values, and the data is included in this file. Though these may not be useful for machine learning directly, valuable information can be obtained with some processing.

6.1. Testing and Evaluation

In order to ensure data accuracy, the complete aggregation process, SAMs to TAMs conversions was tested. This was done by testing the real data and also creating synthetic/test data in two ways:

1. Random Spot Checking
2. Synthetic Data Creation
1. **Random Spot Checking**

   The TAM table was printed out on large format paper. Randomly selected rows were then examined, and the values were calculated manually by looking up the data in the database that would contribute to that TAM entry. The manual calculation's result was then compared to the TAM table's entry.

   For example:

   For semester Fall 2013, say Time Interval 2 is chosen which corresponds to Milestone 2. Then for the weeks falling in that Milestone (in this case M2), subset of teams is selected. For each team following TAM measures are generated with SQL scripts:

   - averageCodingDeliverablesHoursTotalByStudent
   - standardDeviationCodingDeliverablesHoursTotalByStudent
   - averageCodingDeliverablesHoursAverageByStudent
   - standardDeviationCodingDeliverablesHoursAverageByStudent

2. **Synthetic Data Creation**

   Due to complexity of the data management and aggregation SW we performed functional quality assurance (QA) testing. This was done by creating synthetic values for semesters, time intervals, teams and their TAM features and creating dummy entries into the swe database. The test data was generated by a JAVA script which produces random values for each of the above mentioned variables.

   This test is a black-box test, meaning that no code is examined to determine the functionality. The data generated is checked against specification rather than by examining code.
Test Plan
1. **Test Objective**: To test correct TAM values are produced by inputting synthetic values for SAMs. This is a functional QA test, so SQL code shall not be examined to determine how the TAMs are generated or aggregated. The tests would only concentrate on correctness of results.

2. **Hardware and Software Setup**: A computer or laptop with MySQL and MS Excel installed.

3. **Resources**: Tester is required to have basic data manipulation (DML) rights to the swe database.

4. **Test Sequence**: Synthetic data is generated for SAMs and manually aggregated to produce TAMs in an Excel sheet. Same SAM data would then be entered into swe database and TAMs would be generated using SQL procedures. Both data from Excel sheet and database would be compared to each other.

5. **Expected Result**: The data generated both manually and through SQL procedures should match. In case it fails to match, defect is created and SQL queries are checked.

```java
for (int team = 1; team <= 2; team++){
    int week = 0;
    for (int timeInterval = 1; timeInterval <= 3; timeInterval++){
        for (int i = 1; i <= 2; i++){
            week = week + 1;
            for (int member = 1; member <= 5; member++){
                int meetingTime = randomGenerator.nextInt(10);
                RawData data = new RawData(semester,team,timeInterval,week,member,meetingTime);
            }
        }
    }
}
```

**Code Snippet Used For Random Values Generation While Generating Synthetic Data**
7. Implementation of SETAP Data Collection

In this chapter we present an overview of data processing pipeline and then implementation details of individual components.

7.1. Overview of Data Processing Pipeline

SETAP data is first collected from each student (Student Activity Measures or SAM) via weekly online time card surveys (WTS), statistics from SE Tool logs (TL) and Instructors Observations (IO) covering adherence to checkpoint delivery, document and Software deliverables, teamwork issues, problems etc. This raw data is converted into CSV files either manually or by custom scripts and then stored on master MySQL database called swe in the form of tables. SAM data from student team members is then aggregated into Team Activity Measures (TAM) with the help of SQL procedures Figure 11. TAM data is paired with instructor determined grades/outcomes for student learning of SE Process and SE Product to form ML training database. This data, together with extensive header documentation is disseminated via number of CSV files, each for determined time period of the class.

Following is the detail explanation of conversion procedures that describe how data from various data sources is entered into master database.

1. **Team Placement Survey (TPS):** Manually entered into CSV file and then a PHP script used to enter CSV file into swe database
2. **Weekly Timecard Surveys:** Data from Google Doc spreadsheet, converted to CSV files then manually exported into swe database
3. **Tool Logs - Subversion Logs:** Data first exported to xml format. Python script filters team and semester information from user's login name and then stores it into swe database
4. **Instructor observations (IO) and Evaluations:** Data entered into spreadsheets and manually exported on swe database
5. **Data Vetting and Cleanup:** done both manually and automated.
6. **SAM to TAM conversion:** Accomplished by Automated SQL
7. **Training Data:** Python script used to extract all TAM’s for 11 time intervals for SE Product and SE Process to form training data.
7.2. Data Processing Pipeline – Output File Creation

A script running on a remote user’s machine named setapexport calls createTam() in the database, then examines the TAM table. This script can output the data in various forms, such as CSV (for public distribution) or .arff files for internal ML experimentation. After TAM’s have been generated, two types of output files, one CSV and other arff are created where CSV is intended for general distribution to the researchers and arff are used as input to SETAP machine learning program. This is achieved by running a single python script setapexport.py that handles file creation.
7.3. Infrastructure Architecture for SETAP data collection

The SETAP Project’s data infrastructure is composed of two servers (Figure 12). The first server, hosted at sfsuswe.com is the server on which the main SETAP database is hosted. On this server is also hosted the Unix shell accounts that the students use during the semester to develop their projects. A second server located at setapproject.org hosts the project’s website, and non-confidential material such as previously published papers and the TAM feature training database.

Both of these servers are served using AWS (Amazon Web Services) instances. Users interact with these servers from their local computers over the internet.

7.4. Data for Dissemination

One of the goals of SETAP is to disseminate training database for others to use. TAM data is paired with instructor determined grades/outcomes for student learning of SE Process and SE Product to form ML training database. A custom Python script then exports training database data for chosen time interval into CSV files ready to be used by ML analysis software. Each of these CSV files has extensive human-readable header information automatically generated for data provenance and management.
CSV file

This file includes all TAMs as well as process and product outcomes. In order to generate this file the above mentioned script is invoked with the csv flag:

./setapexport -csv . The header of the file contains extensive information. It has three main types of headers:

1. Introduction
   This header introduces basic project and contact information. This information is extracted from file csvCopyrightHeader.txt which must be kept in the same directory as script. Figure 13

   **INTRODUCTION**
   The data contained in this file was collected over a period of many semesters from students engaged in software engineering classes at San Francisco State University (CSC 640, 648 and 848). All students consented to this data being shared for research purposes as long as no uniquely identifiable information was contained in the distributed files. The information was collected through various means, with emphasis being placed on the collection of objective, quantifiable information. For more information on the data collection procedures please see the papers that are available at setapproject.org. These give detailed information on data collection methodology and the classes themselves.

   Figure 13: Introduction header of CSV file

2. Data Statistics
   The header provides information about various aspects of data, e.g. number of teams, data points, records, etc. It is automatically generated from the createDataStatistics svn procedure. createDataStatistics() creates the d_statistics table which is then read into and formatted for the CSV file comment header. Figure 14

   **GENERAL STATISTICS**
   Number of semesters: 7
   First semester: Fall 2012
   Last semester: Fall 2015
   Number of students: 383
   Class sections: 18

   Number of TAM features: 115
   Number of class labels (outcomes): 2
   Issues closed on time: 202
   Issues closed late: 353
   Total issues: 255

   **TEAM COMPOSITION STATISTICS**
   Local Teams: 59
   Global Teams: 15
   Total: 74 Teams

   **OUTCOME (CLASSIFICATION) STATISTICS**
   Total Outcomes: 74
   Values: A F A F
   Counts: 56 18 42 32

   Figure 14: Snapshot of data statistics header of CSV file
3. TAM headers

They are derived from the comments listed in csvTamHeader.txt file which must be located in the same directory as setapexport.py. All TAM headers are described automatically from the database. The information for each TAM is located in the comment field of the TAM table. These comments can be adjusted in the createTamDerived() procedure in the database. Figure 15

![Figure 15: Snapshot of TAM header of CSV file](image)

We disseminate total 22 CSV files for 11 time intervals (11 files for SE Process Component and 11 for SE Product Component) Figure 16. These files containing ML training data could be downloaded from following SETAP website: [http://setapproject.org/](http://setapproject.org/). In order to download the data, users have to register (so SETAP team can track usage for grant purposes):

![Figure 16: Thumbnail of CSV file comprising of ML training data](image)
8. Challenges Encountered while Collecting and Processing Data.

There were some challenges while collecting data and making it ready for machine learning analysis. Some of these challenges are listed below:

8.1. Team Lead Hours
   We discovered that due to the very nature of the fact that not all team leads had entered timecard surveys every semester, that there were time intervals with NULLs. Further, since not every team was a global team, there were nulls in all the non-global team lead prep time fields. Therefore, these two fields were left out.

8.2. Global Team Data
   Due to the variability of reliability in the data collection means, we did not include global teams in the data. All the data was collected from SFSU. We did include the data from the SFSU teams that did participate globally. This created a problem which was that since the SVN repo is shared by all member of the team (local and global), in processing the SVN logs, there were entries for their global counterparts that we could not assign user numbers to. Therefore, we threw out all the SVN log entries for these global team members, keeping only the entries for the local teams.

8.3. SVN Logs
   For some students this was the first use SVN in a team setting. The use of the tool was variable. In a normal situation, each member of a team would commit work to the team repository under their individual username, thus identifying the person committing the work. Each repository also has their group account as a user, allowing the team to use this user to deploy into their account from the repository. For whatever reason, some teams committed to their repository under their group account username. These records were therefore thrown out.

   Due to the above factors which cause SVN records to be excluded, when calculating the number of commits the SVN revision number is not used. Instead a SQL COUNT is used to determine the actual number of legitimate commits.
8.4 Time Intervals

One major complication while recording hours in milestone is that team surveys are given weekly, and may cross milestone boundaries. So if a survey week crosses milestone boundaries, percentages of results are divided between the two milestones by percentage of week. Figure 17 explains this in detail.

For example, when a team member enters coding hours (hoursCoding) in the weekly time card survey for 4th week, then if 2 days of that week fall in M1 and 5 days in M2, M1 gets $2/7 \times \text{hoursCoding}$ and M2 gets $5/7 \times \text{hoursCoding}$.
Appendix A. SETAP Project Documents

This appendix contains full-size reproductions of the various documents used in class and during data processing.

Appendix A0001: Team Placement Survey

Team Placement Survey for Software Engineering Class

Fall 2015 (Rev. 08/30/15)

The results of the following questions are used only for forming teams with approximately the same experience mix of skills, background, programming, etc. Your response to these questions does not affect your class grade in any manner. Please print legibly and provide your honest self-assessment!

<table>
<thead>
<tr>
<th>Name:</th>
<th>Student ID:</th>
</tr>
</thead>
<tbody>
<tr>
<td>email:</td>
<td>Your gender (circle one): male female</td>
</tr>
<tr>
<td>Class in which you are enrolled (circle one):</td>
<td>CSC 640 CSC 848</td>
</tr>
<tr>
<td>Did you take the equivalent of CSC 640 at another school (circle one)?</td>
<td>Y N</td>
</tr>
<tr>
<td>Have you taken or are you taking CSC 412 (circle one)?</td>
<td>No Yes, Currently Yes, Past Semester</td>
</tr>
<tr>
<td>Are you currently working?</td>
<td>Y N</td>
</tr>
<tr>
<td>Your current GPA (if new student, your GPA from last school):</td>
<td></td>
</tr>
<tr>
<td>Are you here on an international student visa?</td>
<td>Y N</td>
</tr>
<tr>
<td>Do you have work experience in the tech industry?</td>
<td>Y N</td>
</tr>
<tr>
<td>Would you like to be a team lead?</td>
<td>Y N</td>
</tr>
<tr>
<td>Would you participate in global teams?</td>
<td>Y N</td>
</tr>
<tr>
<td>Approximately how many lines of code have you written in last six months:</td>
<td></td>
</tr>
</tbody>
</table>

For the following questions, please circle one number on the scale to the right, where 1 represents beginner level, and 5 represents expert level:

<table>
<thead>
<tr>
<th>Your programming skills in general are (circle one):</th>
<th>Beginner</th>
<th>Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your skill in developing web applications is: (circle one):</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>Your skill in developing web apps with PHP is (circle one):</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>Your skill in JavaScript for GUI development is (circle one):</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>Your experience in teamwork in SW development is (circle one):</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>Your experience in distributed SW teams is (circle one):</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>Your experience with SW development tools (IDEs, SVN, bug tracking etc.):</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>Your project management experience is (circle one):</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
</tbody>
</table>

Do you have experience using JavaScript libraries, CSS templates or Frameworks (e.g. bootstrap, jQuery, CodeIgniter?) (circle one): Y N

If yes, please list which technologies:

Continued on Back
For the following questions, please write the code indicated. If you have no experience with the specified topic, or cannot answer the question, just leave the response area blank.

**HTML proficiency**
Write a simple html file that will display *Foo Bar* in the browser when the page is loaded.

**PHP proficiency**
Write a PHP program that displays *Hello Fall 2015 648/848* when the program is run.

**CSS proficiency**
Write a CSS class selector named good that sets the text color to green.
Appendix A0002: Weekly Timecard Survey

Timecard Survey CSC 648/848 Spring 2016

This survey covers the time you spent from Saturday, March 5, 2016 (12:00 AM) through Friday, March 11, 2016 (11:59 PM) (estimates).

This survey is mandatory; however, the answers you provide will not be used in grading.

Please enter times accurately and as close as you remember them for the week indicated.

All students are required to answer questions denoted by an asterisk (*).

* Required

Week *
○ Saturday 3/5/16 00:00:00 through Friday 3/11/16 23:59:59

Team Number *
Choose

Your sfauws.com username *
The part of your email address before the @

Your answer

Please enter the total time (in hours) you spent this week on all meetings (local or global groups, personal or teleconferencing):

Your answer

Please enter the total time (in hours) you spent this week on all face-to-face meetings which involved at least two team members:

Your answer

Please enter the total time (in hours) you spent this week on non-coding related deliverables (documentation, talking to customers etc.):

Your answer

Please enter the total time (in hours) you spent this week on coding related deliverables (SW design, code, testing):

Your answer

Please enter the total time (in hours) you spent this week providing any kind of help to your teammates:

Your answer

For Team Leads ONLY: Please enter the total time (in hours) spent this week on administrative/organizational efforts such as setting the meetings, organizing activities etc. (excluding the time spent in the meetings or activities):

Your answer

Comments

Your answer

SUBMIT

Never submit passwords through Google Forms.
Appendix A0003: Instructor Class Log

**Instructor Class Log** (Rev. 09/21/15)

Use this log to record pertinent observations of student teamwork during the development of the final class project. This form can be used to log observations during instructor meetings with teams (either regular scrum meetings during class or extra meetings outside the classroom), or at any time when instructor gets involved in resolving student team problems and issues. Please keep hard copy of all logs and group them by student teams. They will be used later for team evaluation, grading and research on teaching teamwork in SE education.

Instructor: Date of meeting or issue:

Team #: This is recording a (Check one):

- Regular scrum team meeting
- Extra team Meeting
- Teamwork Issue
- Other:____________

For Meeting Only: # of team members in team / # present at meeting: /
Names of team members absent, and reasons for absence (e.g. excused):

<table>
<thead>
<tr>
<th>Team effectiveness as observed by the instructors during the meeting (circle one):</th>
<th>Team lead effectiveness as observed by the instructors during the meeting (circle one):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Below Expectation</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Below Expectation</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Issues identified by instructor or raised by students (record the issue, follow up, deadline):

Status of old issues (identify which issue, date opened, progress, date closed):

Comments:

☐ Check if continued on reverse
Appendix A0004: Evaluator Product Grading Worksheet

**Grading Template: Software Engineering Class Demos (V 12/14/15)**

<table>
<thead>
<tr>
<th>Rubric Item</th>
<th>Comment/Guidance</th>
<th>Grade Scale&lt;sup&gt;1&lt;/sup&gt;</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Correctness and reliability of operation.</td>
<td>Operates correctly, does not crash; browser renders items correctly.</td>
<td>Lowest: 1</td>
<td>Highest: 5</td>
</tr>
<tr>
<td>2. Ease of use, user interface</td>
<td>Flow, layout, alignment, minimal scrolling, intuitiveness, follows basic UI principles. Exclude quality of graphics from grade.</td>
<td>Lowest: 1</td>
<td>Highest: 5</td>
</tr>
<tr>
<td>3. Performance</td>
<td>System performance as observed during the demo: speed of rendering items after user actions.</td>
<td>Lowest: 1</td>
<td>Highest: 5</td>
</tr>
<tr>
<td>4. Presentation style and effectiveness of final product demo</td>
<td>Demo delivery: use cases and story shown, presentation style and delivery, adherence to time, ability to deal with Q&amp;A</td>
<td>Lowest: 1</td>
<td>Highest: 5</td>
</tr>
</tbody>
</table>

<sup>1</sup> Rank: Relative ranking of the quality of teams' product. Each item should be ranked between 1 (the best team) and the number of teams, with no repetition of numbers.

<sup>2</sup> Grading scale: 1 - represents lowest level of performance, 5 - highest level of performance.

Rev. 24/13-2

---

Final Grade (out of 25 points): reviewer: leave blank

Final Grade (A-F): reviewer: leave blank
# Appendix A0005: Instructor Product Grading Worksheet

## Grading Template: Software Engineering Product (V 05/17/15)

<table>
<thead>
<tr>
<th>Team #:</th>
<th>School:</th>
<th>Date:</th>
<th>Grader:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rubric Item</th>
<th>Comment/Guidance</th>
<th>Grade Scale①</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Correctness and reliability of operation.</td>
<td>Operates correctly, does not crash; browser renders items correctly.</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td></td>
</tr>
<tr>
<td>2. Functionality of product vs. requirements and use cases</td>
<td>Functionality as committed in M4 priority 1; all these functions work.</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td></td>
</tr>
<tr>
<td>3. Ease of use, user interface</td>
<td>Flow, layout, alignment, minimal scrolling, intuitiveness, follows basic UI principles. Exclude quality of graphics from grade.</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td></td>
</tr>
<tr>
<td>4. Performance</td>
<td>System performance as observed during the demo: speed of rendering items after user actions.</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td></td>
</tr>
<tr>
<td>5. Presentation style and effectiveness of final product demo</td>
<td>Demo delivery: use cases and story shown, presentation style and delivery, adherence to time, ability to deal with Q&amp;A</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td></td>
</tr>
<tr>
<td>6. Architecture</td>
<td>Check M3. Check adherence to implementation following basic architectural patterns and principles?</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td>Reviewers: leave blank</td>
</tr>
<tr>
<td>7. Database Design</td>
<td>Check M3. Well organized, good naming convention and convention use, etc.</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td>Reviewers: leave blank</td>
</tr>
<tr>
<td>8. Code Quality</td>
<td>Check code (in M5 and on system). Naming conventions followed? Quality number of comments?</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td>Reviewers: leave blank</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank①</th>
<th>Final Grade (out of 25 points): reviewer: leave blank</th>
<th>Final Grade (A, F): reviewer: leave blank</th>
</tr>
</thead>
</table>

① Rank: Relative ranking of the quality of teams' product. Each item should be ranked between 1 (the best team) and the number of teams, with no repetition of numbers.

② Grade scale: 1 = represents lowest level of performance, 5 = average level of performance, 10 = highest level of performance.
Appendix A0006: Instructor Process Grading Worksheet

<table>
<thead>
<tr>
<th>Rubric Item</th>
<th>Comment/Guidance</th>
<th>Grade Scale¹ / Count</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Fraction of team participating at meetings with the instructor.</td>
<td>Check logs. 95% or more attendance of team is A.</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td></td>
</tr>
<tr>
<td>2. Quality and timing of followup on outstanding issues.</td>
<td>Check logs. Did the team respond on time? Quality of response?</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td></td>
</tr>
<tr>
<td>3. Ability to deal with feedback constructively</td>
<td>Check logs. Does the team take and process feedback and suggestions? Timeliness of actions? Any non-responses?</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td></td>
</tr>
<tr>
<td>4. On time production of SW and non-SW deliverables.</td>
<td>Check logs. If late, how many incidents? How many incidents when permission was obtained?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Quality and completeness of non-SW deliverables.</td>
<td>Examine MS docs, make sure team followed specific format/content guidelines.</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td></td>
</tr>
<tr>
<td>6. Number of teamwork issues in which the instructor had to intervene.</td>
<td>Check logs. A.C.F should reflect overall level of instructor involvement. high number reduces the grade.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Ability to apply best SE processes and teamwork practices</td>
<td>Check logs.</td>
<td>Lowest 1 2 3 4 Highest 5</td>
<td></td>
</tr>
</tbody>
</table>

¹ Grading scale: 1 - represents lowest level of performance, 3 - average level of performance, 5 - highest level of performance.
² Rank: Relative ranking of the teams’ adherence to SPE process. Each team should be ranked between 1 (the best) and the number of teams, with no repetition of numbers.
Rev. 2/11/13-2
Appendix B. SQL Code

Appendix B0001: createTimecardComplete()

CREATE DEFINER=`msosnick`@`localhost` PROCEDURE `createTimecardComplete`()
BEGIN
DROP TABLE IF EXISTS `t_timecardTimeIntervalDate`;
CREATE TABLE `t_timecardTimeIntervalDate`
SELECT t.*, m.`semesterId`,
MIN(m.`start`) as start,
MAX(m.`end`) as end
FROM `timeInterval` t
JOIN `Milestone` m
WHERE m.`number` >= t.`milestoneStart`
AND m.`number` <= t.`milestoneEnd`
GROUP BY m.`semesterId`,t.`timeIntervalId`
ORDER BY m.`semesterId`,t.`timeIntervalId`;

DROP TABLE IF EXISTS `t_timecardWeeksInTimeInterval`;
CREATE TABLE `t_timecardWeeksInTimeInterval`
SELECT
w.`semesterId`,
t.`timeIntervalId`,
w.`number` as weekNumber,
w.`weekId` as weekId,
weekPercent(w.`start`,w.`end`,t.`start`,t.`end`) as percent,
w.`start` weekStart, w.`end` weekEnd, t.`start` intervalStart,t.`end` intervalEnd FROM `week` w
JOIN `t_timecardTimeIntervalDate` t
WHERE weekPercent(w.`start`,w.`end`,t.`start`,t.`end`) != 0
ORDER BY w.`semesterId`, t.`timeIntervalId`, w.`number`;

DROP TABLE IF EXISTS `t_timecardSurveyPlusWeekId`;
CREATE TABLE `t_timecardSurveyPlusWeekId`
SELECT weekIdFromDate(s.datestamp) as weekId, s.* FROM timecardSurvey s
WHERE s.studentId <> 0;

DROP TEMPORARY TABLE IF EXISTS `t_timecardStudentTeam`;
CREATE TEMPORARY TABLE `t_timecardStudentTeam`
SELECT
t.`teamId`,
t.`semesterId`,
t.`teamTypeId`,
t.`leadStudentId`,
t.`teamName`,
t.`teamDistribution`,
t.`description`,
t.`D_globalType`,
t.`D_grouptype`,
s.`studentId`,
s.`gender`,
s.`class`
FROM `team` t
JOIN `student` s
WHERE t.`teamId` = s.`teamId`;

DROP TABLE IF EXISTS `t_studentTeamTimecard`;
CREATE TABLE `t_studentTeamTimecard`
SELECT
st.`teamId`,
st.`semesterId`,
st.`teamTypeId`,
st.`leadStudentId`,
st.`teamName`,
st.`teamDistribution`,
st.`description`,
st.`D_globalType`,
st.`D_grouptype`,
st.`gender`,
st.`class`,
tw.*
FROM `t_timecardStudentTeam` st
JOIN `t_timecardSurveyPlusWeekId` tw
WHERE tw.`studentId` = st.`studentId`;

DROP TABLE IF EXISTS `d_timecardComplete`;
CREATE TABLE `d_timecardComplete`
SELECT
w.weekId,
w.percent,
w.timeIntervalId,
s.`teamId`,
s.`leadStudentId`,
s.`timecardSurveyId`,
s.`studentId`,
s.`datestamp`,
s.meetingHours,
s.inPersonMeetingHours,
s.nonCodingDeliverablesHours,
s.codingDeliverablesHours,
s.helpHours,
s.leadAdminHours,
s.globalLeadAdminHours,
s.comments
FROM `t_studentTeamTimecard` s
JOIN `t_timecardWeeksInTimeInterval` w
WHERE w.weekId = s.weekId
AND NOT s.teamNumber IS NULL;
END
Appendix B0002: createTimecardDerived()

CREATE DEFINER=`msosnick`@`localhost` PROCEDURE `createTimecardDerived`()
BEGIN
CALL createTimecardComplete();
DROP TABLE IF EXISTS `t_timecardResultsByTimeInterval`;
CREATE TABLE `t_timecardResultsByTimeInterval`
SELECT teamId, timeIntervalId,
    COUNT(timecardSurveyId) AS teamMemberResponseCount,
    SUM(meetingHours*percent) AS meetingHoursTotal,
    SUM(meetingHours)/COUNT(meetingHours) AS meetingHoursAverage,
    STD(meetingHours) AS meetingHoursStandardDeviation,
    SUM(inPersonMeetingHours*percent) AS inPersonMeetingHoursTotal,
    SUM(inPersonMeetingHours)/COUNT(inPersonMeetingHours) AS inPersonMeetingHoursAverage,
    STD(inPersonMeetingHours) AS inPersonMeetingHoursStandardDeviation,
    SUM(nonCodingDeliverablesHours*percent) AS nonCodingDeliverablesHoursTotal,
    SUM(nonCodingDeliverablesHours)/COUNT(nonCodingDeliverablesHours) AS nonCodingDeliverablesHoursAverage,
    STD(nonCodingDeliverablesHours) AS nonCodingDeliverablesHoursStandardDeviation,
    SUM(codingDeliverablesHours*percent) AS codingDeliverablesHoursTotal,
    SUM(codingDeliverablesHours)/COUNT(codingDeliverablesHours) AS codingDeliverablesHoursAverage,
    STD(codingDeliverablesHours) AS codingDeliverablesHoursStandardDeviation,
    SUM(helpHours*percent) AS helpHoursTotal,
    SUM(helpHours)/COUNT(helpHours) AS helpHoursAverage,
    STD(helpHours) AS helpHoursStandardDeviation,
    COUNT(IF(studentId=leadStudentId,leadAdminHours,NULL)) AS leadAdminHoursResponseCount,
    SUM(IF(studentId=leadStudentId,leadAdminHours*percent,NULL)) AS leadAdminHoursTotal,
    SUM(IF(studentId=leadStudentId,leadAdminHours*percent)/COUNT(IF(studentId=leadStudentId,leadAdminHours,NULL))) AS leadAdminHoursAverage,
    STD(IF(studentId=leadStudentId,leadAdminHours)) AS leadAdminHoursStandardDeviation,
    COUNT(IF(studentId=leadStudentId,globalLeadAdminHours,NULL)) AS globalLeadAdminHoursResponseCount,
    SUM(IF(studentId=leadStudentId,globalLeadAdminHours*percent,NULL)) AS globalLeadAdminHoursTotal,
    SUM(IF(studentId=leadStudentId,globalLeadAdminHours*percent)/COUNT(IF(studentId=leadStudentId,globalLeadAdminHours,NULL))) AS globalLeadAdminHoursAverage,
    STD(IF(studentId=leadStudentId,globalLeadAdminHours)) AS globalLeadAdminHoursStandardDeviation
FROM `d_timecardComplete`
GROUP BY `teamId`, `timeIntervalId`;

DROP TEMPORARY TABLE IF EXISTS `t_timecardDerivedByWeek`;
CREATE TEMPORARY TABLE `t_timecardDerivedByWeek`
SELECT
  teamId, timeIntervalId, weekId,
  COUNT(weekId) AS responsesByWeek,
  SUM(meetingHours*percent) AS meetingHoursTotalByWeek,
  SUM(meetingHours*percent)/COUNT(meetingHours) AS meetingHoursAverageByWeek,
  SUM(inPersonMeetingHours*percent) AS inPersonMeetingHoursTotalByWeek,
  SUM(inPersonMeetingHours*percent)/COUNT(meetingHours) AS inPersonMeetingHoursAverageByWeek,
  SUM(nonCodingDeliverablesHours*percent) AS nonCodingDeliverablesHoursTotalByWeek,
  SUM(nonCodingDeliverablesHours*percent)/COUNT(meetingHours) AS nonCodingDeliverablesHoursAverageByWeek,
  SUM(codingDeliverablesHours*percent) AS codingDeliverablesHoursTotalByWeek,
  SUM(codingDeliverablesHours*percent)/COUNT(meetingHours) AS codingDeliverablesHoursAverageByWeek,
  SUM(helpHours*percent) AS helpHoursTotalByWeek,
  SUM(helpHours*percent)/COUNT(meetingHours) AS helpHoursAverageByWeek,
  COUNT(IF(studentId=leadStudentId,leadAdminHours,NULL)) AS leadAdminHoursResponseCountByWeek,
  SUM(IF(studentId=leadStudentId,leadAdminHours*percent,NULL)) AS leadAdminHoursTotalByWeek,
  SUM(IF(studentId=leadStudentId,leadAdminHours*percent,NULL))/COUNT(IF(studentId=leadStudentId,leadAdminHours,NULL)) AS leadAdminHoursAverageByWeek,
  COUNT(IF(studentId=leadStudentId,globalLeadAdminHours,NULL)) AS globalLeadAdminHoursResponseCountByWeek,
  SUM(IF(studentId=leadStudentId,globalLeadAdminHours*percent,NULL)) AS globalLeadAdminHoursTotalByWeek,
  SUM(IF(studentId=leadStudentId,globalLeadAdminHours*percent,NULL))/COUNT(IF(studentId=leadStudentId,globalLeadAdminHours,NULL)) AS globalLeadAdminHoursAverageByWeek
FROM `d_timecardComplete`
GROUP BY `teamId`,`timeIntervalId`,`weekId`;

#-----------------------------------------------------------------------------------

DROP TEMPORARY TABLE IF EXISTS `t_timecardDerivedByWeekAvSd`;
CREATE TEMPORARY TABLE `t_timecardDerivedByWeekAvSd`
SELECT teamId, timeIntervalId,
  AVG(responsesByWeek) AS averageResponsesByWeek,
  STD(responsesByWeek) AS standardDeviationResponsesByWeek,
  AVG(meetingHoursTotalByWeek) AS averageMeetingHoursTotalByWeek,
  STD(meetingHoursTotalByWeek) AS standardDeviationMeetingHoursTotalByWeek,
  AVG(meetingHoursAverageByWeek) AS averageMeetingHoursAverageByWeek,
  STD(meetingHoursAverageByWeek) AS standardDeviationMeetingHoursAverageByWeek,
  AVG(inPersonMeetingHoursTotalByWeek) AS averageInPersonMeetingHoursTotalByWeek,
  STD(inPersonMeetingHoursTotalByWeek) AS standardDeviationInPersonMeetingHoursTotalByWeek,
AVG(inPersonMeetingHoursAverageByWeek) AS averageInPersonMeetingHoursAverageByWeek,
STD(inPersonMeetingHoursAverageByWeek) AS standardDeviationInPersonMeetingHoursAverageByWeek,
AVG(nonCodingDeliverablesHoursTotalByWeek) AS averageNonCodingDeliverablesHoursTotalByWeek,
STD(nonCodingDeliverablesHoursTotalByWeek) AS standardDeviationNonCodingDeliverablesHoursTotalByWeek,
AVG(nonCodingDeliverablesHoursAverageByWeek) AS averageNonCodingDeliverablesHoursAverageByWeek,
STD(nonCodingDeliverablesHoursAverageByWeek) AS standardDeviationNonCodingDeliverablesHoursAverageByWeek,
AVG(codingDeliverablesHoursTotalByWeek) AS averageCodingDeliverablesHoursTotalByWeek,
STD(codingDeliverablesHoursTotalByWeek) AS standardDeviationCodingDeliverablesHoursTotalByWeek,
AVG(codingDeliverablesHoursAverageByWeek) AS averageCodingDeliverablesHoursAverageByWeek,
STD(codingDeliverablesHoursAverageByWeek) AS standardDeviationCodingDeliverablesHoursAverageByWeek,
AVG(helpHoursTotalByWeek) AS averageHelpHoursTotalByWeek,
STD(helpHoursTotalByWeek) AS standardDeviationHelpHoursTotalByWeek,
AVG(helpHoursAverageByWeek) AS averageHelpHoursAverageByWeek,
STD(helpHoursAverageByWeek) AS standardDeviationHelpHoursAverageByWeek,
AVG(leadAdminHoursResponseCountByWeek) AS averageLeadAdminHoursResponseCountByWeek,
STD(leadAdminHoursResponseCountByWeek) AS standardDeviationLeadAdminHoursResponseCountByWeek,
AVG(leadAdminHoursTotalByWeek) AS averageLeadAdminHoursTotalByWeek,
STD(leadAdminHoursTotalByWeek) AS standardDeviationLeadAdminHoursTotalByWeek,
AVG(globalLeadAdminHoursResponseCountByWeek) AS averageGlobalLeadAdminHoursResponseCountByWeek,
STD(globalLeadAdminHoursResponseCountByWeek) AS standardDeviationGlobalLeadAdminHoursResponseCountByWeek,
AVG(globalLeadAdminHoursTotalByWeek) AS averageGlobalLeadAdminHoursTotalByWeek,
STD(globalLeadAdminHoursTotalByWeek) AS standardDeviationGlobalLeadAdminHoursTotalByWeek,
AVG(globalLeadAdminHoursAverageByWeek) AS averageGlobalLeadAdminHoursAverageByWeek,
STD(globalLeadAdminHoursAverageByWeek) AS standardDeviationGlobalLeadAdminHoursAverageByWeek
FROM `t_timecardDerivedByWeek`
GROUP BY `teamId`, `timeIntervalId`;

#-------------------------------------------------------------

DROP TEMPORARY TABLE IF EXISTS `t_timecardDerivedByStudent`;
CREATE TEMPORARY TABLE `t_timecardDerivedByStudent`
SELECT
teamId, timeIntervalId, studentId,
COUNT(weekId) AS responsesByStudent,
SUM(meetingHours*percent) AS meetingHoursTotalByStudent,
SUM(meetingHours*percent)/COUNT(meetingHours) AS meetingHoursAverageByStudent,
SUM(inPersonMeetingHours*percent) AS inPersonMeetingHoursTotalByStudent,
SUM(inPersonMeetingHours*percent)/COUNT(meetingHours) AS inPersonMeetingHoursAverageByStudent,
SUM(nonCodingDeliverablesHours*percent) AS nonCodingDeliverablesHoursTotalByStudent,
SUM(nonCodingDeliverablesHours*percent)/COUNT(meetingHours) AS nonCodingDeliverablesHoursAverageByStudent,
SUM(codingDeliverablesHours*percent) AS codingDeliverablesHoursTotalByStudent,
SUM(codingDeliverablesHours*percent)/COUNT(meetingHours) AS codingDeliverablesHoursAverageByStudent,
SUM(helpHours*percent) AS helpHoursTotalByStudent,
SUM(helpHours*percent)/COUNT(meetingHours) AS helpHoursAverageByStudent,
COUNT(IF(studentId=leadStudentId,leadAdminHours,NULL)) AS leadAdminHoursResponseCountByStudent,
SUM(IF(studentId=leadStudentId,leadAdminHours*percent,NULL)) AS leadAdminHoursTotalByStudent,
SUM(IF(studentId=leadStudentId,leadAdminHours*percent,NULL))/COUNT(IF(studentId=leadStudentId,leadAdminHours,NULL)) AS leadAdminHoursAverageByStudent,
COUNT(IF(studentId=leadStudentId,globalLeadAdminHours,NULL)) AS globalLeadAdminHoursResponseCountByStudent,
SUM(IF(studentId=leadStudentId,globalLeadAdminHours*percent,NULL)) AS globalLeadAdminHoursTotalByStudent,
SUM(IF(studentId=leadStudentId,globalLeadAdminHours*percent,NULL))/COUNT(IF(studentId=leadStudentId,globalLeadAdminHours,NULL)) AS globalLeadAdminHoursAverageByStudent
FROM `d_timecardComplete`
group by `teamId`,`timeIntervalId`,`studentId`;

DROP TEMPORARY TABLE IF EXISTS `t_timecardDerivedByStudentAvSd`;
CREATE TEMPORARY TABLE `t_timecardDerivedByStudentAvSd`
SELECT teamId, timeIntervalId,
AVG(responsesByStudent) AS averageResponsesByStudent,
STD(responsesByStudent) AS standardDeviationResponsesByStudent,
AVG(meetingHoursTotalByStudent) AS averageMeetingHoursTotalByStudent,
STD(meetingHoursTotalByStudent) AS standardDeviationMeetingHoursTotalByStudent,
AVG(meetingHoursAverageByStudent) AS averageMeetingHoursAverageByStudent,
STD(meetingHoursAverageByStudent) AS standardDeviationMeetingHoursAverageByStudent,
AVG(inPersonMeetingHoursTotalByStudent) AS averageInPersonMeetingHoursTotalByStudent,
STD(inPersonMeetingHoursTotalByStudent) AS standardDeviationInPersonMeetingHoursTotalByStudent,
AVG(inPersonMeetingHoursAverageByStudent) AS averageInPersonMeetingHoursAverageByStudent,
STD(inPersonMeetingHoursAverageByStudent) AS standardDeviationInPersonMeetingHoursAverageByStudent,
AVG(nonCodingDeliverablesHoursTotalByStudent) AS averageNonCodingDeliverablesHoursTotalByStudent,
STD(nonCodingDeliverablesHoursTotalByStudent) AS standardDeviationNonCodingDeliverablesHoursTotalByStudent,
AVG(nonCodingDeliverablesHoursAverageByStudent) AS averageNonCodingDeliverablesHoursAverageByStudent,
STD(nonCodingDeliverablesHoursAverageByStudent) AS standardDeviationNonCodingDeliverablesHoursAverageByStudent,
AVG(codingDeliverablesHoursTotalByStudent) AS averageCodingDeliverablesHoursTotalByStudent,
STD(codingDeliverablesHoursTotalByStudent) AS standardDeviationCodingDeliverablesHoursTotalByStudent,
AVG(codingDeliverablesHoursAverageByStudent) AS averageCodingDeliverablesHoursAverageByStudent,
STD(codingDeliverablesHoursAverageByStudent) AS standardDeviationCodingDeliverablesHoursAverageByStudent,
AVG(helpHoursTotalByStudent) AS averageHelpHoursTotalByStudent,
STD(helpHoursTotalByStudent) AS standardDeviationHelpHoursTotalByStudent,
AVG(helpHoursAverageByStudent) AS averageHelpHoursAverageByStudent,
STD(helpHoursAverageByStudent) AS standardDeviationHelpHoursAverageByStudent,
AVG(leadAdminHoursResponseCountByStudent) AS averageLeadAdminHoursResponseCountByStudent,
STD(leadAdminHoursResponseCountByStudent) AS standardDeviationLeadAdminHoursResponseCountByStudent,
AVG(leadAdminHoursTotalByStudent) AS averageLeadAdminHoursTotalByStudent,
STD(leadAdminHoursTotalByStudent) AS standardDeviationLeadAdminHoursTotalByStudent,
AVG(globalLeadAdminHoursResponseCountByStudent) AS averageGlobalLeadAdminHoursResponseCountByStudent,
STD(globalLeadAdminHoursResponseCountByStudent) AS standardDeviationGlobalLeadAdminHoursResponseCountByStudent,
AVG(globalLeadAdminHoursTotalByStudent) AS averageGlobalLeadAdminHoursTotalByStudent,
STD(globalLeadAdminHoursTotalByStudent) AS standardDeviationGlobalLeadAdminHoursTotalByStudent,
AVG(globalLeadAdminHoursAverageByStudent) AS averageGlobalLeadAdminHoursAverageByStudent,
STD(globalLeadAdminHoursAverageByStudent) AS standardDeviationGlobalLeadAdminHoursAverageByStudent
FROM `t_timecardDerivedByStudent`
group by `teamId`, `timeIntervalId`;

# Join the two temporary timecard by week tables into a single derived table

DROP TABLE IF EXISTS `d_timecardDerived`;
CREATE TABLE `d_timecardDerived` AS
SELECT v.*,
x.`teamMemberResponseCount`,
x.`meetingHoursTotal`,
x.`meetingHoursAverage`,
x.`meetingHoursStandardDeviation`,
x.`inPersonmeetingHoursTotal`,
x.`inPersonmeetingHoursAverage`,
x.`inPersonMeetingHoursStandardDeviation`,
x.`nonCodingDeliverablesHoursTotal`,
x.`nonCodingDeliverablesHoursAverage`,
x.`nonCodingDeliverablesHoursStandardDeviation`,
x.`codingDeliverablesHoursTotal`,
\begin{verbatim}
FROM `t_timecardDerivedByStudentAvSd` v
    JOIN `t_timecardResultsByTimeInterval` x
    JOIN `t_timecardDerivedByWeekAvSd` y
WHERE v.`teamId` = x.`teamId`
AND v.`timeIntervalId` = x.`timeIntervalId`
AND v.`teamId` = y.`teamId`
AND v.`timeIntervalId` = y.`timeIntervalId`;
\end{verbatim}
Appendix B0003: createTam()

CREATE DEFINER=`msosnick`@`localhost` PROCEDURE `createTam`()
    NO SQL
    DETERMINISTIC
BEGIN

    CALL timeIntervalDate();
    CALL createTimecardDerived();
    CALL createSvnDerived();
    CALL createIssueLogDerived();

    DROP TEMPORARY TABLE IF EXISTS `t_tamHeader`;

    CREATE TEMPORARY TABLE `t_tamHeader`
    SELECT s.`year`,
           s.`name` as semester,
           tid.`timeIntervalId` as timeInterval,
           t.`teamId` as teamNumber,
           t.`semesterId`,
           teamMemberCount(t.`teamId`) as teamMemberCount,
           femaleMemberCount(t.`teamId`) / teamMemberCount(t.`teamId`) as femaleTeamMembersPercent,
           teamLeadGender(t.`teamId`) as teamLeadGender,
           teamDistribution(t.`teamId`) as teamDistribution
    FROM `semester` s
    JOIN `team` t
    JOIN `t_timeIntervaldate` tid
    WHERE t.`semesterId` = s.`semesterId`
    AND teamMemberCount(t.`teamId`) IS NOT NULL
    AND t.`teamTypeId` BETWEEN 1 AND 3
    AND tid.`semesterId` = t.`semesterId`
    ORDER BY s.`year`, s.`name`, t.`teamId`,
            tid.`timeIntervalId`;

    DROP TABLE IF EXISTS `TAM`;
    CREATE TABLE IF NOT EXISTS `TAM`  
    SELECT t.*,
           IF(ISNULL(d.`teamMemberResponseCount`), 0, d.`teamMemberResponseCount`) AS teamMemberResponseCount,
           d.`meetingHoursTotal`,
           meetingHoursAverage,
           meetingHoursStandardDeviation,
           inPersonMeetingHoursTotal,
           inPersonMeetingHoursAverage,
           inPersonMeetingHoursStandardDeviation,
           nonCodingDeliverablesHoursTotal,
           nonCodingDeliverablesHoursAverage,
           nonCodingDeliverablesHoursStandardDeviation,
           codingDeliverablesHoursTotal,
           codingDeliverablesHoursAverage,
           codingDeliverablesHoursStandardDeviation,
           helpHoursTotal,
           helpHoursAverage,
helpHoursStandardDeviation, leadAdminHoursResponseCount, leadAdminHoursTotal, leadAdminHoursAverage, leadAdminHoursStandardDeviation, globalLeadAdminHoursResponseCount, globalLeadAdminHoursTotal, globalLeadAdminHoursAverage, globalLeadAdminHoursStandardDeviation, averageResponsesByWeek, standardDeviationResponsesByWeek, averageMeetingHoursTotalByWeek, standardDeviationMeetingHoursTotalByWeek, averageMeetingHoursAverageByWeek, standardDeviationMeetingHoursAverageByWeek, averageInPersonMeetingHoursTotalByWeek, averageInPersonMeetingHoursAverageByWeek, standardDeviationInPersonMeetingHoursAverageByWeek, averageNonCodingDeliverablesHoursTotalByWeek, standardDeviationNonCodingDeliverablesHoursTotalByWeek, averageNonCodingDeliverablesHoursAverageByWeek, standardDeviationNonCodingDeliverablesHoursAverageByWeek, averageCodingDeliverablesHoursTotalByWeek, standardDeviationCodingDeliverablesHoursTotalByWeek, averageCodingDeliverablesHoursAverageByWeek, standardDeviationCodingDeliverablesHoursAverageByWeek, averageHelpHoursTotalByWeek, standardDeviationHelpHoursTotalByWeek, averageHelpHoursAverageByWeek, standardDeviationHelpHoursAverageByWeek, averageLeadAdminHoursResponseCountByWeek, standardDeviationLeadAdminHoursResponseCountByWeek, averageLeadAdminHoursTotalByWeek, standardDeviationLeadAdminHoursTotalByWeek, averageGlobalLeadAdminHoursResponseCountByWeek, standardDeviationGlobalLeadAdminHoursResponseCountByWeek, averageGlobalLeadAdminHoursTotalByWeek, standardDeviationGlobalLeadAdminHoursTotalByWeek, averageGlobalLeadAdminHoursAverageByWeek, standardDeviationGlobalLeadAdminHoursAverageByWeek, averageResponsesByStudent, standardDeviationResponsesByStudent, averageMeetingHoursTotalByStudent, standardDeviationMeetingHoursTotalByStudent, averageMeetingHoursAverageByStudent, standardDeviationMeetingHoursAverageByStudent, averageInPersonMeetingHoursTotalByStudent, standardDeviationInPersonMeetingHoursTotalByStudent, averageInPersonMeetingHoursAverageByStudent, standardDeviationInPersonMeetingHoursAverageByStudent, averageNonCodingDeliverablesHoursTotalByStudent, standardDeviationNonCodingDeliverablesHoursTotalByStudent, averageNonCodingDeliverablesHoursAverageByStudent, standardDeviationNonCodingDeliverablesHoursAverageByStudent, averageCodingDeliverablesHoursTotalByStudent, standardDeviationCodingDeliverablesHoursTotalByStudent, averageCodingDeliverablesHoursAverageByStudent, standardDeviationCodingDeliverablesHoursAverageByStudent,
averageCodingDeliverablesHoursAverageByStudent,
standardDeviationCodingDeliverablesHoursAverageByStudent,
averageHelpHoursTotalByStudent,
standardDeviationHelpHoursTotalByStudent,
averageHelpHoursAverageByStudent,
standardDeviationHelpHoursAverageByStudent,

IF(ISNULL(s.'commitCount'), 0, 'commitCount') AS commitCount,
IF(ISNULL(s.'commitCount'), 0, 'uniqueCommitMessageCount') AS uniqueCommitMessageCount,
IF(ISNULL(s.'commitCount'), 0, 'uniqueCommitMessagePercent') AS uniqueCommitMessagePercent,
IF(ISNULL(s.'commitCount'), 0, 'commitMessageLengthTotal') AS commitMessageLengthTotal,
IF(ISNULL(s.'commitCount'), 0, 'commitMessageLengthAverage') AS commitMessageLengthAverage,
IF(ISNULL(s.'commitCount'), 0, 'commitMessageLengthStandardDeviation') AS commitMessageLengthStandardDeviation,
IF(ISNULL(s.'commitCount'), 0, 'averageCommitCountByWeek') AS averageCommitCountByWeek,
IF(ISNULL(s.'commitCount'), 0, 'standardDeviationCommitCountByWeek') AS standardDeviationCommitCountByWeek,
IF(ISNULL(s.'commitCount'), 0, 'averageUniqueCommitMessageCountByWeek') AS averageUniqueCommitMessageCountByWeek,
IF(ISNULL(s.'commitCount'), 0, 'standardDeviationUniqueCommitMessageCountByWeek') AS standardDeviationUniqueCommitMessageCountByWeek,
IF(ISNULL(s.'commitCount'), 0, 'averageUniqueCommitMessagePercentByWeek') AS averageUniqueCommitMessagePercentByWeek,
IF(ISNULL(s.'commitCount'), 0, 'standardDeviationUniqueCommitMessagePercentByWeek') AS standardDeviationUniqueCommitMessagePercentByWeek,
IF(ISNULL(s.'commitCount'), 0, 'averageCommitMessageLengthTotalByWeek') AS averageCommitMessageLengthTotalByWeek,
# IF(ISNULL(s.'commitCount'), 0, 'averageCommitMessageLengthAverageByWeek') AS averageCommitMessageLengthAverageByWeek,
# IF(ISNULL(s.'commitCount'), 0, 'standardDeviationCommitMessageLengthAverageByWeek') AS standardDeviationCommitMessageLengthAverageByWeek,
# IF(ISNULL(s.'commitCount'), 0, 'averageCommitMessageLengthStandardDeviationByWeek') AS averageCommitMessageLengthStandardDeviationByWeek,
# IF(ISNULL(s.'commitCount'), 0, 'standardDeviationCommitMessageLengthStandardDeviationByWeek') AS standardDeviationCommitMessageLengthStandardDeviationByWeek,
IF(ISNULL(s.'commitCount'), 0, 'averageCommitCountByStudent') AS averageCommitCountByStudent,
IF(ISNULL(s.'commitCount'), 0, 'standardDeviationCommitCountByStudent') AS standardDeviationCommitCountByStudent,
IF(ISNULL(s.'commitCount'), 0, 'averageUniqueCommitMessageCountByStudent') AS averageUniqueCommitMessageCountByStudent,
IF(ISNULL(s.`commitCount`), 0, 
'tandardDeviationUniqueCommitMessageCountByStudent') AS standardDeviationUniqueCommitMessageCountByStudent,
IF(ISNULL(s.`commitCount`), 0, 
'averageUniqueCommitMessagePercentByStudent') AS averageUniqueCommitMessagePercentByStudent,
IF(ISNULL(s.`commitCount`), 0, 
'standardDeviationUniqueCommitMessagePercentByStudent') AS standardDeviationUniqueCommitMessagePercentByStudent,
IF(ISNULL(s.`commitCount`), 0, 
'averageCommitMessageLengthTotalByStudent') AS averageCommitMessageLengthTotalByStudent,
IF(ISNULL(s.`commitCount`), 0, 
'standardDeviationCommitMessageLengthTotalByStudent') AS standardDeviationCommitMessageLengthTotalByStudent,
IF(ISNULL(s.`commitCount`), 0, 
'averageCommitMessageLengthAverageByStudent') AS averageCommitMessageLengthAverageByStudent,
IF(ISNULL(s.`commitCount`), 0, 
'standardDeviationCommitMessageLengthAverageByStudent') AS standardDeviationCommitMessageLengthAverageByStudent,
IF(ISNULL(s.`commitCount`), 0, 
'averageCommitMessageLengthStandardDeviationByStudent') AS averageCommitMessageLengthStandardDeviationByStudent,
if(ISNULL(i.`issueCount`), 0, i.`issueCount`) AS issueCount,
if(ISNULL(i.`issueCount`), 0, i.`onTimeIssueCount`) AS onTimeIssueCount,
if(ISNULL(i.`issueCount`), 0, i.`lateIssueCount`) AS lateIssueCount,
o.`processLettergrade`,
o.`productLetterGrade`
FROM `t_tamHeader` t
LEFT JOIN `d_timecardDerived` d
  ON t.`teamNumber` = d.`teamId`
  AND t.`timeInterval` = d.`timeIntervalId`

LEFT JOIN `d_svnDerived` s
  ON t.`teamNumber` = s.`teamId`
  AND t.`timeInterval` = s.`timeIntervalId`

LEFT JOIN `d_issuelogDerived` i
  ON t.`teamNumber` = i.`teamId`
  AND t.`timeInterval` = i.`timeIntervalId`
JOIN `outcomes` o

WHERE t.`teamNumber` = o.`teamId`;

# Comments for each TAM column. The comments are divided into two parts. The first part is the standard English description used and the part after the || is used for generating the SETAP .arff training files used for machine learning.

ALTER TABLE `TAM` MODIFY COLUMN `year` int(11) NOT NULL COMMENT 'The year in which the data was collected as a four-digit number.||NUMERIC';

ALTER TABLE `TAM` MODIFY COLUMN `semester` varchar(8) NOT NULL COMMENT 'The semester in which the data was collected as a string, one of {Spring,Fall,Summer,Winter}||{Spring,Fall,Summer,Winter}';
ALTER TABLE `TAM` MODIFY COLUMN `semesterId` int(11) NOT NULL COMMENT 'Unique number indicating from which year and semester data was collected. 1=Fall 2012, 2=Spring 2013, etc. ||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `timeInterval` int(11) NOT NULL COMMENT 'The time interval for which the row\'s data is calculated||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `teamNumber` int(11) NOT NULL COMMENT 'Number assigned to each team unique across all semesters and years.||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `teamMemberCount` int(11) NOT NULL COMMENT 'Number of members on the team.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `femaleTeamMembersPercent` double DEFAULT NULL COMMENT 'Ratio of female to male team members||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `teamLeadGender` varchar(10) NOT NULL COMMENT 'Gender of the team lead.||{M,F}';
ALTER TABLE `TAM` MODIFY COLUMN `teamDistribution` varchar(10) NOT NULL COMMENT 'Geographic distribution of teams. Teams composed of students from one university are local. Teams composed of students from more than one university are global. Stored as a string, one of {Global,Local}.||{Global,Local}';
ALTER TABLE `TAM` MODIFY COLUMN `teamMemberResponseCount` int(11) COMMENT 'Number of times team members submitted weekly timecard survey in time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `meetingHoursTotal` double DEFAULT NULL COMMENT 'Average number of hours spent meeting by all team members in the time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `meetingHoursAverage` double DEFAULT NULL COMMENT 'Average number of hours spent meeting by all team members, computed as meetingsHoursTotaled over all team responses in time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `meetingHoursStandardDeviation` double DEFAULT NULL COMMENT 'Standard deviation of the number of hours spent in meetings by all team members in the time interval. Standard deviation is over all the team responses in the time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `inPersonMeetingHoursTotal` double DEFAULT NULL COMMENT 'Total number of hours spent meeting in person by all team members in the time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `inPersonMeetingHoursAverage` double DEFAULT NULL COMMENT 'Average number of hours spent meeting in person by all team members in the time interval, over all the team responses in the time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `inPersonMeetingHoursStandardDeviation` double DEFAULT NULL COMMENT 'Standard deviation of the number of hours spent meeting in person by all team members in the time interval. Standard deviation is over all the team responses in the time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `nonCodingDeliverablesHoursTotal` double DEFAULT NULL COMMENT 'Total number of hours spent preparing non-coding deliverables by all team members in the time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `nonCodingDeliverablesHoursAverage` double DEFAULT NULL COMMENT 'Average number of hours spent preparing non-coding deliverables by all team members in the time interval, over all the team responses in the time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `nonCodingDeliverablesHoursStandardDeviation` double DEFAULT NULL COMMENT 'Standard deviation of the number of hours spent preparing non-coding deliverables by all team members in the time interval. Standard deviation is over all the team responses in the time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `codingDeliverablesHoursTotal` double DEFAULT NULL COMMENT 'Total number of hours spent preparing coding deliverables by all team members in the time interval.'||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `codingDeliverablesHoursAverage` double DEFAULT NULL COMMENT 'Average number of hours spent preparing coding deliverables by all team members in the time interval, over all the team responses in the time interval.'||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `codingDeliverablesHoursStandardDeviation` double DEFAULT NULL COMMENT 'Standard deviation of the number of hours spent preparing coding deliverables by all team members in the time interval. Standard deviation is over all the team responses in the time interval.'||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `helpHoursTotal` double DEFAULT NULL COMMENT 'Total number of hours spent helping other team members by all team members in the time interval.'||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `helpHoursAverage` double DEFAULT NULL COMMENT 'Average number of hours spent helping other team members by all team members in the time interval, over all the team responses in the time interval.'||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `helpHoursStandardDeviation` double DEFAULT NULL COMMENT 'Standard deviation of the number of hours spent helping other team members by all team members in the time interval. Standard deviation is over all the team responses in the time interval.'||NUMERIC';

# EXCLUDE FROM .arff
ALTER TABLE `TAM` MODIFY COLUMN `leadAdminHoursResponseCount` bigint(21) NULL DEFAULT '0' COMMENT 'Total number of timecard surveys the team lead has submitted during the time interval.'||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `leadAdminHoursTotal` double DEFAULT NULL COMMENT 'Total number of hours team lead has spent administering the team during the time interval.'||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `leadAdminHoursAverage` double DEFAULT NULL COMMENT 'Average number of hours team lead has spent administering the team for the time interval.'||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `leadAdminHoursStandardDeviation` double DEFAULT NULL COMMENT 'Standard deviation of reported hours spent by team lead administering the team over the time interval.'||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `globalLeadAdminHoursResponseCount` bigint(21) NULL DEFAULT '0' COMMENT 'Total number of timecard surveys the global team lead has submitted during the time interval.'||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `globalLeadAdminHoursTotal` double DEFAULT NULL COMMENT 'Total number of hours global team lead spent administering the global team portion of the team during the time interval.'||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `globalLeadAdminHoursAverage` double DEFAULT NULL COMMENT 'Average number of hours global team lead spent administering the global team portion of the team for the time interval.'||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `globalLeadAdminHoursStandardDeviation` double DEFAULT NULL COMMENT 'Standard deviation of the number of hours global team lead spent administering the global team portion of the team for the time interval. Standard deviation is over all the global team lead responses for the team in the time interval.'||EXCLUDE';

# INCLUDE IN .arff
ALTER TABLE `TAM` MODIFY COLUMN `averageResponsesByWeek` double DEFAULT NULL COMMENT 'The average number of weekly timecard responses over the weeks in the time interval.'||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationResponsesByWeek` double DEFAULT NULL COMMENT 'The standard deviation of the response count to the timecard survey by team members by week, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageMeetingHoursTotalByWeek` double DEFAULT NULL COMMENT 'Average of the meeting hours totalled by week, over the number of weeks in the time interval, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationMeetingHoursTotalByWeek` double DEFAULT NULL COMMENT 'Standard deviation of the weekly total meeting hours over the weeks in the time interval, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageMeetingHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationMeetingHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageInPersonMeetingHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationInPersonMeetingHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageInPersonMeetingHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationInPersonMeetingHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageNonCodingDeliverablesHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationNonCodingDeliverablesHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageNonCodingDeliverablesHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationNonCodingDeliverablesHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageCodingDeliverablesHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationCodingDeliverablesHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageCodingDeliverablesHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationCodingDeliverablesHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageHelpHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationHelpHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageHelpHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationHelpHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***, NUMERIC';

# EXCLUDE FROM .arff
ALTER TABLE `TAM` MODIFY COLUMN `averageLeadAdminHoursResponseCountByWeek` double DEFAULT NULL COMMENT '***TODO***, EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationLeadAdminHoursResponseCountByWeek` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `averageLeadAdminHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationLeadAdminHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `averageGlobalLeadAdminHoursResponseCountByWeek` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationGlobalLeadAdminHoursResponseCountByWeek` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `averageGlobalLeadAdminHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationGlobalLeadAdminHoursTotalByWeek` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `averageGlobalLeadAdminHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationGlobalLeadAdminHoursAverageByWeek` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';

# INCLUDE IN .arff
ALTER TABLE `TAM` MODIFY COLUMN `averageResponsesByStudent` double DEFAULT NULL COMMENT 'Average of the number of responsesxxxxx||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationResponsesByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageMeetingHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationMeetingHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageMeetingHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationMeetingHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageInPersonMeetingHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationInPersonMeetingHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageInPersonMeetingHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationInPersonMeetingHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageNonCodingDeliverablesHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationNonCodingDeliverablesHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageNonCodingDeliverablesHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***|NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN
`standardDeviationNonCodingDeliverablesHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN
`averageCodingDeliverablesHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN
`standardDeviationCodingDeliverablesHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN
`averageCodingDeliverablesHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN
`standardDeviationCodingDeliverablesHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN
`averageHelpHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN
`standardDeviationHelpHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN
`averageHelpHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN
`standardDeviationHelpHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';

# EXCLUDE FROM .arff ALL INVALID, ELIMINATE
# ALTER TABLE `TAM` MODIFY COLUMN
`averageLeadAdminHoursResponseCountByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN
`standardDeviationLeadAdminHoursResponseCountByStudent` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN
`averageLeadAdminHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN
`standardDeviationLeadAdminHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN
`averageGlobalLeadAdminHoursResponseCountByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN
`standardDeviationGlobalLeadAdminHoursResponseCountByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN
`averageGlobalLeadAdminHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN
`standardDeviationGlobalLeadAdminHoursTotalByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN
`averageGlobalLeadAdminHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN
`standardDeviationGlobalLeadAdminHoursAverageByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';

# INCLUDE IN .arff
ALTER TABLE `TAM` MODIFY COLUMN `commitCount` bigint(21)  NULL DEFAULT '0' COMMENT 'Total number of repository commits by the team in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `commitMessageCount` bigint(21)  NULL DEFAULT '0' COMMENT 'Total number of repository commits with unique commit messages by the team in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `commitMessagePercent` decimal(24,4) DEFAULT NULL COMMENT 'Percent of unique vs. total number of commit messages in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `commitMessageLengthTotal` decimal(31,0) DEFAULT NULL COMMENT 'Total length of all commit messages by the team in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `commitMessageLengthAverage` decimal(13,4) DEFAULT NULL COMMENT 'Average length of commit messages by the team over the total count in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `commitMessageLengthStandardDeviation` double DEFAULT NULL COMMENT 'Standard deviation of the commit messages by the team in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageCommitCountByWeek` decimal(24,4) DEFAULT NULL COMMENT 'Average of the weekly total number of commits over the number of weeks in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationCommitCountByWeek` double DEFAULT NULL COMMENT 'Standard deviation of the weekly total number of commits over the number of weeks in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageUniqueCommitMessageCountByWeek` decimal(28,8) DEFAULT NULL COMMENT 'Average of the weekly total number of commits with unique commit messages over the number of weeks in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationUniqueCommitMessageCountByWeek` double DEFAULT NULL COMMENT 'Standard deviation of the weekly total number of commits with unique commit messages over the number of weeks in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageCommitMessageLengthTotalByWeek` decimal(35,4) DEFAULT NULL COMMENT 'Average of the weekly total length of all commit messages over the number of weeks in the time interval||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationCommitMessageLengthTotalByWeek` double DEFAULT NULL COMMENT 'Standard deviation of the total length of all commit messages over the number of weeks in the time interval||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `averageCommitMessageLengthAverageByWeek` decimal(17,8) DEFAULT NULL COMMENT '***TODO***||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationCommitMessageLengthAverageByWeek` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `averageCommitMessageLengthStandardDeviationByWeek` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationCommitMessageLengthStandardDeviationByWeek` double DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageCommitCountByStudent` decimal(24,4) DEFAULT NULL COMMENT 'Total number of commit counts divided by the number of students in the team by time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationCommitCountByStudent` double DEFAULT NULL COMMENT 'Standard deviation of the commit counts by student.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `averageUniqueCommitMessageCountByStudent` decimal(24,4) DEFAULT NULL COMMENT '***TODO***||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationUniqueCommitMessageCountByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `averageUniqueCommitMessagePercentByStudent` decimal(28,8) DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationUniqueCommitMessagePercentByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `averageCommitMessageLengthTotalByStudent` decimal(35,4) DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationCommitMessageLengthTotalByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `averageCommitMessageLengthAverageByStudent` decimal(17,8) DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationCommitMessageLengthAverageByStudent` double DEFAULT NULL COMMENT '***TODO***||EXCLUDE';
ALTER TABLE `TAM` MODIFY COLUMN `issueCount` bigint(21) DEFAULT '0' COMMENT 'Total number of issues per team opened in the time interval.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `onTimeIssueCount` bigint(21) DEFAULT '0' COMMENT 'Total number of issues per team opened in the time interval that were closed on time or satisfactorily.||NUMERIC';
ALTER TABLE `TAM` MODIFY COLUMN `lateIssueCount` bigint(21) DEFAULT '0' COMMENT 'Total number of issues opened per team in the time interval that were not closed on time or not satisfactorily.||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `unresolvedIssueCount` bigint(21) DEFAULT '0' COMMENT 'Total number of issues opened in the time interval that were not closed.||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `totalIssueCountByWeek` bigint(21) NULL DEFAULT '0' COMMENT '***************** THIS INTERVAL DOES NOT MAKE SENSE*****************||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN `averageIssueCountByWeek` decimal(24,4) DEFAULT NULL COMMENT 'For the time interval, the weekly count of issues opened averaged over the time interval||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationIssueCountByWeek` double DEFAULT NULL COMMENT 'For the time interval, the standard deviation the weekly count of issues opened||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `totalOnTimeIssueCountByWeek` bigint(21) NULL DEFAULT '0' COMMENT '***************** THIS INTERVAL DOES NOT MAKE SENSE*****************||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN `averageOnTimeIssueCountByWeek` decimal(24,4) DEFAULT NULL COMMENT 'For the time interval, the weekly count of issues opened which were closed on time, averaged over the time interval.||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationOnTimeIssueCountByWeek` double DEFAULT NULL COMMENT 'For the time interval, the standard deviation the weekly count of issues opened which were closed on time.||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `totalLateIssueCountByWeek` bigint(21) NULL DEFAULT '0' COMMENT '***************** THIS INTERVAL DOES NOT MAKE SENSE *****************||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN `averageLateIssueCountByWeek` decimal(24,4) DEFAULT NULL COMMENT 'For the time interval, the weekly count of issues opened which were not closed on time, averaged over the time interval.||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `standardDeviationLateIssueCountByWeek` double DEFAULT NULL COMMENT 'For the time interval, the standard deviation the weekly count of issues opened which were not closed on time.||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `totalUnresolvedIssueCountByWeek` bigint(21) NULL DEFAULT '0' COMMENT '***************** THIS INTERVAL DOES NOT MAKE SENSE *****************||EXCLUDE';
# ALTER TABLE `TAM` MODIFY COLUMN `averageUnresolvedIssueCountByWeek` decimal(24,4) DEFAULT NULL COMMENT 'For the time interval, the weekly count of issues opened which were not closed, averaged over the time interval||NUMERIC';
# ALTER TABLE `TAM` MODIFY COLUMN `processLetterGrade` varchar(1) DEFAULT NULL COMMENT 'Semester class label (outcome) for the team\'s adherence to software engineering best practices. A indicates meets expectations, F indicates below expectations.||{A,F}';
# ALTER TABLE `TAM` MODIFY COLUMN `productLetterGrade` varchar(1) DEFAULT NULL COMMENT 'Semester class label (outcome) for the team\'s product quality. A indicates meets expectations, F indicates below expectations.||{A,F}';

SELECT * FROM `TAM`
ORDER BY year, semester, teamNumber, timeInterval;

END