CIS 450/550: Database and Information Systems

Course Project Guidelines

Table of Contents

A. Overview

1. Description

The goal of this project is to identify at least two large, overlapping datasets of interest, import and integrate them into a SQL relational database, then create a web-enabled application of your own choosing over the database. You will work in teams of four, and you can use any SQL database technology and any web-development stack.

The objective of this project is to assess whether you're able to apply the concepts you're learning about in this course to solve unstructured problems with limited hands-on guidance. Specifically, the project will require you to practice:

- wrangling and cleaning data,
- performing entity resolution (ER),
- designing schema,
- writing SQL queries,
- choosing database indexes,
- designing a web application,
- optimizing SQL queries.

If you use this as an opportunity to build something you're proud of, you are welcome to use the final product to demonstrate your skills to potential employers.

We also want you to enjoy completing the project. This is one of the most open-ended projects in the Penn computer science curriculum. You can build just about any game, application, or website you want, as long as it's built on top of a SQL database and has multiple distinct pages. For example, past students have built trading card games, recipe recommendation platforms, and many other interesting apps. So we encourage you to have fun with it and be creative. Check out the resources section for more inspiration.

In this document, you will find deliverables you'll produce while completing the project, each project milestone described in detail, the grading criteria and several sources of extra credit, and several resources that might benefit you, including dataset sources, descriptions of past students' projects, and links to useful tools.

You will also be assigned a TA as a project mentor, who will be your primary point-of-contact on staff for questions about your project. You may also post your questions relating to this project on Piazza. Good luck and have fun!

2. Deliverables

The table below provides a brief description of each deliverable you'll need to submit to complete the project. Please see the milestone section for more detailed instructions. **Check Canvas module Project Information and Tutorials for past project examples and video overviews of the project goals and expectations.**

Milestone deliverables due dates are tentative and subject to change

Deliverable	Description	Milestone
Project Group	Form Groups of 4 on Canvas: People > Project Groups	Milestone 0 (Feb. 20)
Project Proposal	A 1-2 page document that identifies the datasets you will use, gives a rough idea of what your application may do, and demonstrates you have performed some basic descriptive analysis on the data	Milestone 1 (Mar. 3)
Project Outline	A 2-3 page document that describes what your application will do, explains its significance, and gives the schema your database will implement	Milestone 2 (Mar. 20)
Database Population	A 3NF- or BCNF-normalized MySQL database hosted on AWS RDS that contains all your data	Milestone 3 (Apr. 3)
SQL Queries	A list of SQL queries that can run on your database with short explanations of what each query is supposed to do. At least two queries must be complex.	Milestone 3 (Apr. 3)
Mentor Meeting and API Specs	An informal Zoom meeting with your project mentor where you demonstrate that you have implemented some basic functionality of your application, backend API should be done and testable.	Milestone 4 (Apr. 17)
Final Report	A 6-10 page document that thoroughly describes the problem you tried to solve, your application functionality, your database design, your query optimization efforts, etc.	Milestone 5 (May 2)
Application Code	A zip file that contains your application code, a list of dependencies, instructions for building the app and any code you wrote to populate the database or wrangle data	Milestone 5 (May 2)
Application Demo	A 2-4 minute screen-captured video that demonstrates your application's main functionalities and includes narration from at least one group member. Upload the video to your UPenn account-associated Google drive and attach the link to Canvas.	Milestone 5 (May 2)

Final Demo	A video conference call with your project mentor, another TA, and the professor where you demonstrate functionality of your application. We use this to test edge cases and ensure your video demo wasn't heavily edited.	(Apr. 28 - May 2)

B. Milestones and final presentation

1. Milestone 1 - Project Proposal (Mar. 3)

Develop an initial idea.

The initial step is to select your teammates, then brainstorm ideas applications and search for datasets. Remember your application must have several distinct pages and a SQL database with at least several tables. Your datasets must be:

1. **Large** -- We recommend that you have at least 2 large datasets containing at least 10,000 rows (post-cleaning) to achieve meaningful query optimizations. In the past, most projects involved datasets of 100,000 rows so the bigger, the better.
2. **Overlapping** -- Your application will need to include queries that require information from both datasets. This means the datasets need to be related and probably need to contain references to the same entities.

Please Note that: While 10,000 is the minimum, we recommend that you have at least 2 databases on the order of minimum 100,000 rows in order to do meaningful optimizations to queries.

Before finalizing your dataset choices, you should conduct some basic exploratory analysis (EDA). Poke around, compute summary statistics, and try to get a sense of how clean, large, and complete the data is. These factors will affect how difficult it will be to clean/pre-process the data, create and query a database, and perform entity resolution later. We provide an introduction to Pandas--a Python library for data analysis-- in this tutorial and demonstrate how to perform EDA with Pandas in this tutorial.

Next, write your project proposal. The project proposal should contain the following:

1. A List of group members, email addresses, and GitHub usernames
2. A Description of application/website idea
3. For each dataset you've chosen:
    a. A 1-2 sentence description of the dataset
    b. A link to where you found the dataset
    c. If you're scraping the data, a description of how you will scrape it
    d. If you're not scraping the data:
        i. relevant size statistics (e.g. For a table, mb/gb, number of rows, and number of attributes. For a graph, mb, number of nodes, and number of edges)

   ii. summary statistics of several attributes (e.g. report mean, standard deviation)
4. A list of at least 5 queries (in natural language) you could write for your datasets. Some of these should require complex SQL (aggregations, subqueries, joins, etc.)

Milestone 1 Submission Instructions: One member should upload the proposal as PDF to Gradescope and add the other three members of the group to the submission as teammates.

Based on your project proposal, we will assign each group a TA who will serve as your project mentor for the remainder of the semester. Your project mentor will email your group with feedback on your proposal. One person should add every group member and your project mentor (see Piazza for GitHub info) to a **private GitHub repository** before the next milestone.

2. Milestone 2 - Project Outline (Mar. 20)

Detailed functionality description and schema design, and initial set up.

In this phase, you will set up your version control environment, explain your project idea in more detail, and assign responsibilities to each group member.

First, make sure to create a GitHub repository and share it with your Project Mentor. During development, you will use the repository to share code and perform version control. Your project mentor will also use the repository during grading to review the quality of your code and ensure you haven't plagiarized any of it.

Next, write your project outline. The outline should contain the following:

1. Motivation for the idea/description of the problem the application solves
2. List of features you will definitely implement in the application
3. List of features you might implement in the application, given enough time
4. List of pages the application will have and a 1-2 sentence description of each page. We expect that the functionality of each page will be meaningfully different than the functionality of the other pages.
5. Relational schema as an ER diagram
6. SQL DDL for creating the database
7. Explanation of how you will clean and pre-process the data. This tutorial demonstrates how to do simple pre-processing in Python.
8. List of technologies you will use. You must use some kind of SQL database. We recommend using MySQL or Oracle specifically because you will use MySQL in HW2, and we will provide guidance for setting up a MySQL database. Some groups in the past have had issues with MySQL, but Oracle is another option.
9. Description of what each group member will be responsible for

Milestone 2 Submission Instructions: One group member should upload the project outline to Gradescope as a PDF and add all other group members to the submission as teammates.

IMPORTANT NOTE ABOUT AWS: For the next milestone, you will populate and host your database on AWS. We recommend that you use AWS Academy over AWS Free Tier to avoid students getting charged for overages. Separate instructions on getting started with AWS Academy will be made available on Canvas and through Piazza. However, if you choose to host your database using your AWS Free Tier account, you wil be responsible for overages.

3. Milestone 3 - Database Population/SQL Queries (Apr. 3)

Pre-process your data, perform entity resolution, populate the database, and write queries.

Now that you have a schema, clean your data, perform entity resolution as needed (make sure your ER diagram is connected), create a database and populate it with your data. You may find the following resources useful for performing these steps (check Canvas):

1. **Processing Tutorial** - Interactive Google Colab notebook that demonstrates pre-processing and entity resolution techniques in Python
2. **RDS setup instructions and data upload** - from past exercises and homeworks

Next, create a .txt file containing the following:

1. **Queries –** A list of 10 SQL queries. At least 4 of these must be complex ones. Complex queries draw on multiples tables to find interesting insights about your data. Our expectation on complex queries is explained right below.
2. **Descriptions** -- A 1-2 sentence description of what each query is supposed to do
3. **Credentials** -- Instructions and guest credentials for accessing the database

Clarification about complex queries:

A complex query uses a combination of elements such as but not limited to: **multiple joins, subqueries or views, aggregations, universal/existential checks**. One or two aggregations or a simple JOIN between two tables wouldn't suffice. A tip on designing a complex query is that you should start with a complex idea to implement; if you try to overcomplicate a simple idea just to meet this requirement, you'll struggle and most importantly, we'll notice. We also expect complex queries to have non-trivial runtime (>20s before optimization, >1s after optimization). Check out this doc to get a sense of what a complex query should look like. Query 1 would be on the border of complex and trivial, but Queries 2, 3, and 4 meet our expectations.

Milestone 3 Submission Instructions: One group member should upload the **.txt** file of SQL queries to Gradescope and add all other group members to the submission as teammates.

Make sure that your **Github is up to date** (TAs will be looking at commits and individual contributions), and that you have a ReadMe file in your Github repo including a description of your project and what each directory in your repo corresponds to.

Now, begin building your application! Here are a few recommendations for getting started:

1. **Use the code from HW2 as a starting point.** Feel free to copy and paste it into your project directory just as long as you mention that you used it, in your final report.
2. **Before building anything, make sure you can retrieve data from your database and display it on a webpage.** This requires setting up a server, connecting it to your database, and creating one route that queries your database then sends the data to the web client. You shouldn't need to write much code for any of these steps, but if you're not familiar with web-development, putting all the pieces together can be challenging. Once you've done it once, it's easy to replicate and modify for your different features. Reach out to your project mentor early if you struggle with this step.

4. Milestone 4 - API Specification and Check-In (Apr. 17)

4.1 API Specification

Make a simple document containing the following for each route your app executes:

- Description of the functionality of the route
- Request path
- Request parameters, types, and descriptions
- Query parameters
- Response parameters, types, and descriptions.

Using (possibly updated) queries from Milestone 3, consolidate the API routes for your server application. While you are not required to implement much on the client side, you should think about the client-side functionality supported by each route you design.

Roughly, every route should use (possibly different versions of) one complex query with a few parameters. Specifically, you must have routes corresponding to each of the (at least 10) queries, plus any other 'auxiliary' routes that may or may not use SQL queries. Please note that a route may exist without a query associated with it (e.g., adding a user). Also be mindful of the 4 complex queries that you must include in the final project.

You get to decide what request parameters are important based on what your query needs to execute properly. You can access the route parameters in req.params (if the param is a path param) or in req.query (if the param is a query param).

Create an API specification for these routes. We encourage you to use the specification from the HW2 MS1 instructions as a starting point. Feel free to include other useful information (like HTTP status codes), et cetera. Alternatively, you may also choose to create an OpenAPI-style specification. It is acceptable to make a few changes to the API as you progress. However, you will be required to include the final version of the specification in the final project report.

An effective way to define an API spec will include, for each route, the following:

1. The request path (ex. /getUser) along with the method (i.e., GET, POST, DELETE, PUT) and a description of what the route is doing (e.g., retrieves a user by username).
2. The request params, including the name (e.g., username), param type (path or query), data type (e.g., integer, string, etc.), required / optional indicator, and description (e.g., username corresponding to a user).
3. The response params, including the name, data type, and description.

Submit a **typed** version of the specification as a .pdf file on Gradescope.

4.2 Mentor Check-in (Demo Basic Functionality)

Schedule a 15-minute Zoom conference call with your project mentor and teammates to demonstrate that you have implemented a portion of the necessary features you listed in the project outline. All members of your team must attend the call. You don't need to prepare a formal presentation for this meeting. Just plan to walk your project mentor through the features you've implemented (no document necessary for the demo).

Your mentor will give you feedback on the features you've built and indicate whether they think your team is on track to finish a polished application. If there are technical challenges your team has been unable to resolve, this is a good opportunity to ask your mentor for advice.

5. Milestone 5 - Final Report, Demo Video, and Code (May 2)

Finish the application, record a demo of it, and write your report

a. Final Report

Write a polished final report containing the following information:

1. **Title --** Name of your website/application (Be creative!)
2. **Introduction --** Explanation of project goals/target problem, description of application functionality, list of group members
3. **Architecture --** List of technologies, description of system architecture/application
4. **Data --** For each dataset, a link to the source, a description of the data, relevant summary statistics, and an explanation of how you use the data
5. **Database --** Explanation of data ingestion procedure and entity resolution efforts, ER diagram, number of instances in each table, normal form and justification
6. **Queries** -- Examples of at least 5 queries in your application and explanations of how they're used. Please report the queries you think are most complex.
7. **Performance evaluation** -- recorded timings before and after optimization(a table might be useful here), descriptions of events being timed, and explanations of why caching, indexing, and other optimizations improved timings. For example, record how long it takes to execute queries, respond to client requests, and execute any other common tasks.
8. **Technical challenges** -- List of technical challenges and how you overcame them

In most cases, the final report turns out to be about 6-10 pages (excluding appendices). DO NOT write beyond 10 pages. Feel free to copy text from your earlier submissions where appropriate, if you think it's sufficiently polished and clear.

b. Code

Create a zip file containing all the code for the application. You should NOT include the node/Python packages your application depends on. You SHOULD include:

1. Application code
2. All code used for cleaning, wrangling, and ingesting the data
3. A list of dependencies
4. Instructions for building it locally

c. Demo

Use screen capture software to record a 2-4 minute video demo of your application. The instructors will only watch the first 4 minutes of your video, so don't make it longer. Your video should include demonstrations of all your

application's pages and features. One or more of your group members should narrate it and describe the queries and optimizations associated with each feature. Describe how much each optimization affected performance.

Record timings associated with your application's features before and after performing optimizations, such as caching and/or indexing. For example, record how long it takes to execute queries, respond to client requests, and execute any other common tasks.

Milestone 5 Submission Instructions: Please submit each deliverable to the appropriate assignment in Canvas. For the Demo video, please upload it to the Penn google mail drive and ensure to make the link accessible to University of Pennsylvania - upload the link to Gradescope.

6. Live Demo (Apr. 28 - May 2)

Demo the final version of your application in a meeting with your project mentor, another TA, and the professor.

You will schedule a 20 minute demo on Zoom with your group. We highly recommend that you make presentation slides for this demo. These slides should be used to cover elements under "Project Overview" below.

We expect the following during the demo:

- Project Overview:
    - Present basic problem, goal, and concept of project
    - Present quick overview of datasets
    - Explain how data was cleaned and normalized
    - Show schema design and ER diagram
    - Show and explain your most complex queries (general idea of what they do/how they work, runtimes/optimizations)
    - Technical Challenges
    - Briefly mention any extra credit features if applicable
    - Anything you would like to add
- Live Demo of Application:
    - You will share the link to your deployed application with the TAs
    - Describe and demonstrate the application's main features and functionality
    - Demonstrate extra credit features if applicable

- Questions:
    - Your project mentor, secondary TA, and/or professor may ask questions about database or application design, query optimizations, etc.
    - Your mentor may ask to see how your application handles certain inputs

Expect the Project Overview + Live Demo of Application to take 10-15 minutes and Questions to take 2-5 minutes.

C. Grading

1. Criteria

The project will be graded out of 100 points. The seven following criteria will be used to evaluate your work. The exact rubric will be released.

1. Milestones - 5%

Was each milestone completed on time with clear effort?

2. Technical Quality - 40%

Does the application solve a non-trivial problem or answer non-trivial questions? Is the application robust and functional? Are the queries complex? Is the database designed well? Is the schema normalized? Was entity resolution and data cleaning performed correctly? Does the application perform interesting joins between data from different sources? Is the code clearly organized? Does the code use multiple files for different components/routes/config/apis? Is the code well-commented?

3. Robustness - 5%

Is there any app-breaking bugs? Does the features and queries work for various user inputs? Is there any renderin for UI elements? Does the app work for incorrect + null inputs?

4. Scope - 15%

Does the application implement a sufficient number of features? Does the application have multiple pages? Does each page interact with a database? Are the datasets large?

5. Presentation, Report, and Demo - 15%

Is the final report written using clear and concise language and correct grammar? Is the final report well-organized and easy to read? Does the final report make use of charts and tables when appropriate? During the video demo, is the narrator easier to hear and understand? Is the information presented accurate?

6. Optimization - 15%

Did the developers attempt to optimize their more complex queries using relevant optimization techniques (e.g. improving the query, caching, indexing)? Does the final report provide reasonable, correct explanations for why each optimization failed or succeeded? Does the final report include relevant and accurate pre and post optimization timings?

7. Look and feel - 5%

Is the application visually appealing? Is the user interface easy to use? Does the application make use of data visualizations and images when appropriate and feasible?

8. Participation - Manual Deductions From Final Project Grade

Did all members of the team contribute to the project? Were all members of the team given the opportunity to contribute to the project? Did team members work cooperatively and support each other? As an individual, did you complete your fair share of the work? As an individual, did you fully complete the tasks assigned to you by the group in a timely manner? As an individual, did you communicate with your group members regarding your ability to meet internal deadlines, as necessary?

2. Awards

The instructors will nominate and vote on several projects to receive awards at the end of the semester. Each project that wins at least one award will receive an extra credit feature. Additional points will not be awarded for receiving multiple nominations or wins.

The following awards will be distributed:

1. Best all-around application
2. Most beautiful application
3. Most technically complex application

3. Extra Credits

Here are some of the aspects that may be awarded with extra credit. However, individual TAs may grant additional extra credit under their discretion. 2 points per EC feature for a maximum of 4 points may be awarded, and the final project score cannot exceed 100%.

- Used NoSQL in addition to SQL
- Integrated with apis to fetch streaming data
- Code coverage (unit testing >80% for backend and/or >80% frontend)
- Application Security (implemented password hashing, privacy modes, SSL certificates etc) (hashing alone will not count)

- Integration with other applications like colab, bing search etc.
- User login experience (Integrating with at least 2 of the following: Google, Facebook, Twitter, etc. Sign In and in addition to having standard sign in)
- Awards (rate 1-5 on technical complexity, best looking, and all-around best)
- Anything cool we might've missed (Subjective and depends on complexity)
- Deployment (need to have the website deployed so TAs can access during presentation)

D. Resources

1. Dataset Resources

We've compiled a list of datasets and dataset aggregators to help jumpstart your search for data. But don't feel compelled to choose something off this list. This is just the tip of the iceberg. There's a lot more data out there!

Dataset Aggregators

1. Wikidata
2. DBpedia
3. World Bank Open Data
4. Tableau datasets
5. sqlbelle's list of datasets and dataset aggregators
6. AwesomeData's list of public datasets
7. fivethirtyeight datasets
8. OpenDataPhilly
9. NYC OpenData
10. Zillow Datasets
11. AWS Datasets
12. Google Cloud Datasets
13. Reddit Datasets
14. Buzzfeed Datasets
15. Data.world
16. Kaggle
17. Federal Government Datasets
18. UCI ML Repository
19. Academic Torrents
20. WorldData.AI

Knowledge Bases

1. World Factbook

2. Wikipedia
3. Greatest Sporting Nation
4. Summer Olympics Medals

Datasets

1. OpenFlights
2. Yelp
3. NYC Taxis

2. Example projects

The list below gives a few examples of projects people have done in the past in this course. We hope these help inspire your group and give you a better idea of what we're looking for. You can use the datasets, but must develop your own original application over the datasets using your own code.

World Bank database Factbook

The data made available by WorldBank presents the most current and accurate global development data available, and it includes national, regional and global estimates. A group of students used this data to make an application that displayed development data for selected countries/regions in interactive visualizations.

World Travel Guide

A group used travel datasets like Google's to build an interface that lets users see information about places they could travel to, based on their travel preferences and needs.

Soccer Fantasy League

A group used this soccer dataset along with data scraped from ESPN to build a Soccer fantasy league. They built interfaces that let users draft their own teams and implemented an account system to save teams and expose different features to different kinds of users.

Olympics App

A group used the Summer Olympics dataset augmented with data from the WorldBank to create an application that helped people learn more about each country, while viewing how well that country has performed at the Olympics historically.

Bike Rental Routes

A group developed a web-app that allowed people to find routes between two points in NYC using a combination of bike, subway and walking. Platforms such as Google Maps assume that the user either owns their own bike or has none at all. This project used Citi Bike

database JSON and combined that with subway stations database of New York City and calculated the shortest distance using a combination of all three, allowing the user to pick up a Citi Bike and drop it off during the journey.

3. Useful Tools

We've compiled a list of software tools that you may find helpful for scraping, cleaning, and ingesting data or making your demo video. If you know of some convenient tool we didn't include, please let us know on Piazza!

Content Extractors

1. Apache Tika -- Extracts metadata and text from word documents, PDF, powerpoints, and many other types of files
2. Jackson Project -- Efficient, easy-to-use JSON parser for Java
3. Trail -- XML Parser for Java
4. jsoup -- HTML parser for Java
5. Selenium for Python -- A Python library for automating interactions with a web browser to perform web scraping or application testing
6. Beautiful Soup -- A Python library for extracting data from HTML and XML files

Data cleaning and exploration tools

1. Pandas -- A Python library for data analysis that provides DataFrame, a convenient data structure for storing and processing large amounts of data
2. Dora -- A Python library for cleaning data and performing exploratory analysis
3. JupyterLab -- Software that enables you to work with Jupyter notebooks -- documents that contain live runnable Python code and display code output inline. Useful for data exploration, cleaning, and pre-processing
4. Google Colab – An online version of JupyterLab that allows Google-Drive-style collaboration

Screen Recording Software

1. ScreenRec -- Free screen recording software for Mac, Windows, and Linux
2. Open Broadcaster Software -- Free screen recording software for Mac, Windows, and Linux

Database Software

1. MongoDB Atlas -- Cloud database hosting service for MongoDB with 512 MB of free storage
2. MongoDB Compass -- GUI for MongoDB
3. MySQL Workbench -- GUI for MySQL

E. Plagiarism Policy

You can refer to the web or any other resource for ideas, but you are STRICTLY NOT ALLOWED to use other people's code directly. If you would like to use some code or snippets, please consult your project mentor and obtain permission before you do so. Please make sure that you cite the original author/source if you are approved to use it. If you are caught violating this policy, you will receive a 0 for the final project and/or the class, and we will refer your case to the Office of Student Conduct (OSC), which may take further disciplinary action.