Project Reference Guide

M4, M5, and M6

1 Overview

The goal of this document is to help you get an overview of what is needed for Milestones 4, 5, and 6. Please refer to Canvas for the specific rubric items that you must satisfy. Please note that this document is new, in an effort to help with issues that students had with past versions of the project, so it's possible that there are bugs. We apologize in advance if that is the case.

At the highest level, Milestones 4-6 are designed to assess various aspects of your final project. Roughly speaking, the milestones are:

- Milestone 4: things about your project that are best assessed by looking at your group's files.
- Milestone 5: things about your project that are best assessed during the demo.
- Milestone 6: helping us to understand any group dynamics that may require differentiating grades.

2 Requirements

This section contains the general implementation requirements. These requirements correspond to rubric items, so make sure to read them carefully.

2.1 Queries

You are expected to implement ten queries, specified below.

Each of the 10 queries must be different, and no query can be a subset of the other. For example, even if your Aggregation with HAVING query uses join(s), you must write and implement a separate query for join to get credit for it, and you can't just remove the Aggregation and HAVING clauses and use them for your join query.

For each item, the chosen query and table(s) should make sense given the context of the application. We want to see you have tried to think of meaningful queries.

Queries 2.1.1-2.1.6 inclusive *may not* be "hardcoded." That is, the completed SQL query cannot be written or saved before application runtime. The user must be able to input or select specific value(s) (specified in each query description) to be used in the query. The variables that contain these values should be part of the SQL expression (e.g., :empNum in the expression WHERE empID = :empNum).

2.1.1 INSERT

Relations needed:

- You must choose one relation to implement INSERT on.

Input format:

- The user should be able to specify what values to insert.

Additional notes:

- The INSERT operation should affect more than one relation. This means the insert must occur on a relation with a foreign key.
    - The INSERT operation should be able to handle the case where the foreign key value in the tuple being inserted does not exist in the relation that is being referred to.
    - This "handling" can either be that the inserted tuple is rejected by the GUI with an appropriate error message or that the values that are being referred to are inserted.
- Note the INSERT statements used in the SQL DDL file do not count towards this query requirement.

2.1.2 UPDATE

Relations needed:

- You must choose one relation to implement UPDATE on.

Input format:

- In this relation, the user should be able to update any number of non-primary key attributes.

Additional notes:

- The relation chosen for the update operation must have at least two non-primary key attributes.
- At least one non-primary key attribute must have either a UNIQUE constraint or be a foreign key that references another relation.
- The application should display the tuples that are available for the relation so the user can select which tuple they want to update (based on the key).
- Oracle does not support ON UPDATE (e.g., for ON UPDATE CASCADE). If you are using Oracle, note in your documentation you know foreign keys *should* be updated. You do not need to do anything else to manually update them.

2.1.3 DELETE

Relations needed:

- You must choose one relation to implement DELETE on.

Input format:

- The user should be able to choose what values to delete. They can do this by specifying the primary key, selecting from a list of tuples, or using another reasonable method of your choice.

Additional notes:

- Implement a cascade-on-delete situation for this relation (or an alternative that was agreed to by the TA if the DB system doesn't provide this).

2.1.4 Selection

Relations needed:

- You must choose one relation to implement Selection on.

Input format:

- The user should be allowed to search for tuples using any number of AND/OR clauses and combinations of attributes. Conditions can be based on equality or (if you want) more complex operations (e.g., less than).

Additional notes:

- There are two main ways to implement the input format:
    1. Using a dynamically generated dropdown of AND/OR options.
    2. By allowing the user to enter a string specifying the conditions. For example, if the relation contained attributes firstName and lastName, the user could enter an arbitrarily long string like "firstName = 'Ada' OR firstName = 'Alan' OR lastName = 'Hamilton' ..." Note this string must not be directly copied into the WHERE clause in the query. Your string must not require that the user knows SQL. **You must do your own parsing and validation of it**. For example, you should accept as input "cats = true AND colour = brindled OR colour = snowshoe"

2.1.5 Projection

Relations needed:

- You must choose one relation to implement Projection on.

Input format:

- The user can choose any number of attributes to view from this relation. Non-selected attributes must not appear in the result.

2.1.6 Join

Relations needed:

- You must implement one query of this category.
- This query must join at least two relations of your choice.

Input format:

- The user must provide at least one value to qualify in the WHERE clause (e.g., join the Customer and the Transaction relation to find the names and phone numbers of all customers *who have purchased a specific item*).

Queries 2.1.7-2.1.10 inclusive *may* be "hardcoded." That is, the completed SQL query can be written and saved before application runtime. However, the results of the query must not be predetermined. The query must be run in the database every time the user wishes to execute the query and new results displayed in the GUI.

2.1.7 Aggregation with GROUP BY

Relations needed:

- You must implement one query of this category using relation(s) of your choice.

Input format:

- You must provide an interface (e.g., button, dropdown, etc.) for the user to execute the query.

Additional notes:

- The query must use aggregation (e.g., min, max, average, or count).

2.1.8 Aggregation with HAVING

Relations needed:

- You must implement one query of this category using relation(s) of your choice.

Input format:

- You must provide an interface (e.g., button, dropdown, etc.) for the user to execute the query.

Additional notes:

- The query must include a HAVING clause.

2.1.9 Nested aggregation with GROUP BY

Relations needed:

- You must implement one query of this category using relation(s) of your choice.

Input format:

- You must provide an interface (e.g., button, dropdown, etc.) for the user to execute the query.

Additional notes:

- The query must find some aggregated value for each group.
- It is fine to use a view to get the desired behaviour.
- Examples of nested aggregation:

```
SELECT    S.rating
FROM      Sailors S
WHERE     AVG(S.age) <= ALL ( SELECT    AVG(s2.age)
                              FROM      Sailors s2
                              GROUP BY  rating );

SELECT    S.rating, AVG (S.age) as avgage
FROM      Sailors S
GROUP BY  S.rating
HAVING    1 < ( SELECT COUNT(*)
                FROM SAILORS S2
                WHERE S.rating = S2.rating );
```

2.1.10 Division

Relations needed:

- You must implement one query of this category using relation(s) of your choice.

Input format:

- You must provide an interface (e.g., button, dropdown, etc.) for the user to execute the query.

Additional notes:

- The query must do division.

2.2 Other features

2.2.1 Graphical User Interface (GUI)

The GUI doesn't need to be fancy, but at least a basic GUI is necessary. The GUI should be usable from the perspective of a non-computer scientist and should not include SQL syntax. If the project does not have a GUI (e.g., it is run through command line) or the GUI is severely broken, a penalty of 40% on the total value of the project grade will be applied.

2.2.2 Sanitization

Values from the user are not directly used in the database. Basic security practices to prevent injection and rainbow attacks have been followed. Encryption is not required. But cover cases as discussed in class, e.g., https://xkcd.com/327/Links to an external site.

2.2.3 Error handling

The user receives notifications about user errors such as trying to insert a duplicate value, invalid input (e.g., invalid characters or an int when only strings are allowed), etc.

2.2.4 User-friendliness

The application is designed for someone who has no knowledge of Computer Science. The interaction and required inputs are reasonable for a non-Computer Scientists (e.g., the user is not required to input a condition such as attributeName <op> value to do a search). The application is designed in a way that is reasonable for users (e.g., having everything on a really long page is not reasonable).

2.2.5 User-friendly query results

Queries are designed to only return the data that is needed and in the right order if required. The client does not do any processing of the data such as filtering/sorting etc.

2.2.6 User notification

The user will receive a success or failure notification upon the completion of an insert, update, delete action and will have a way to verify the action's effect on the database.

2.2.7 Drop, Recreate, and Reload Tables

Groups should be able to use the .sql file submitted for Milestone 4 to drop, recreate, and reload tables.

2.2.8 Sufficient user data

Note that each of your queries must have some non-trivial answers. For example, division with only two products is trivial. Likewise, aggregation (GROUP BY) must produce a reasonable number of groups, and some groups must have more than one row. This means that you must insert enough data in your database using this script to ensure non-trivial query results.

3 Milestone 4 – Implementation

3.1 Purpose

The purpose of Milestone 4 is to implement your project. The following restrictions, in addition to the requirements above, will guide your implementation process. As you design your GUI and plan for Milestone 4, make sure to review these guidelines carefully.

3.2 Repository Deliverables

Commit the deliverables below to the CPSC 304 provided repository and submit a link to your repository on Canvas.

3.2.1 Cover page

Your completed cover page, as usual.

3.2.2 SQL script

- A single SQL script that can be used to create all the tables and data in the database. If you are using multiple scripts while developing, ensure you concatenate them and hand in only a SINGLE SQL script.
- DROP TABLE statements should be included at the beginning of the file to allow the file to be run multiple times if needed.
- The SQL initialization script is how you can set up your database such that you have some data to work with when testing your code. Your database also needs data so that you can demonstrate the functionality and correctness of your project during the Milestone 5 demo.
- This script provides a standardized environment for every member to test against. Every group member can run this SQL script to set up their database in the same way. This is preferable to having everyone use the same account on the database as you may get unexpected test results if multiple people are developing and testing at the same time.
- Note that if something happens such that you cause an unwanted and irreversible change in your database, you can easily restore it to its original state using this script. You will also not affect your other group members.
- This SQL script should be runnable as is (i.e., if we transferred this file to the undergrad servers, we should be able to run it as is without further tweaking).
- This script can be over 500+ lines long with all the CREATE and INSERT statements so start on this early. You are not expected to have a GUI button that calls this setup script (but you can if you want to!).
- If constraints are needed to model some aspects of the ER diagram, they have been added. Note that due to a lack of available examples, if assertions or triggers are required, you may simply say what they would need to do, but not implement them. If you DID implement them, please make a big fuss of them so that your TA can consider giving you extra credit.

3.2.3 PDF file

A single PDF file containing:

- A short description of the final project, and what it accomplished.
- A description of how your final schema differed from the schema you turned in. If the final schema differed, explain why.
- A list of all SQL queries used to satisfy the rubric items and where each query can be found in the code (file name and line number(s)).
- For SQL queries 2.1.7 through 2.1.10 inclusive, include a copy of your SQL query and a maximum of 1-2 sentences describing what that query does. You can embed this in your above list of queries. You don't need to include the output of the query.

The purpose of this requirement is to allow your TAs time outside of your presentation to verify these more complex queries are well formed.

3.3 Additional Information

3.3.1 Use of generative AI tools

You are not allowed to use a programming assistance tool (e.g., GitHub Copilot, ChatGPT, etc.) nor can you use GUI generating tools like drag and drop tools. As a reminder, the use of generative artificial intelligence tools to complete coursework in this course is prohibited in all cases. Use of these tools is considered an unauthorized means to complete an examination or other assignment or assessment and would be considered academic misconduct.

3.3.2 Citing your work

In general, please cite whatever code you use that is not yours. This is a good habit to get into and will save you grief later, even if it's not explicitly required in an assignment. In the case of code we have provided, it's a great idea that you started with it and/or took lines from it. Don't go overboard (a simple code comment is fine), just make sure that if someone comes back to you, you can have the moral high ground.

3.3.3 Maintaining your repository

There should be multiple commits in your repository that show incremental progress. We do not want to see one giant commit with all your code. We may also spot check various commits each group member has made when grading your project. Even if you pair program, you should ensure that there is evidence of equal work contribution from every group member. This can take the form in alternating who commits code. At the end of the day, if there are disputes, one of the first things we will be doing is looking through your repository. If you have no commits, that is likely to put you in a disadvantageous position as compared to your teammates.

3.3.4 Repository deadlines

You cannot change anything in your project between the commit you wish to submit for grading consideration and your demo. The code you hand in for Milestone 4 should be the code you present at your demo. See the syllabus for the rules around submissions/late penalties.

3.3.5 Using an unsupported language

If you wish to use anything other than JavaScript/CS department provided Oracle or PHP/CS department provided Oracle, every group member must send the course coordinator an email. Please sure to also cc. your project mentor. The email should contain the following pieces of information:

- The exact combination of programming language and relational database
- Something stating that you understand this is completely at your own risk and that we will not be able to provide any technical support whatsoever
- Something stating that you understand that the teaching team reserves the right to ask you to come in during the exam period to go through your project again. The email addresses for the course coordinator and your project mentor can found on Canvas.

3.3.6 GitHub account

You must use your UBC GitHub account for all commits to your repository. This makes it easy for your TAs to see what is going on and who has done the commits. If you accidentally commit to your repository using another account, make sure that you make it clear in the documentation that you've used your personal account and what email that is. Moving forward, please use your UBC account. This is essential if a group dispute were to result as not having to track multiple usernames would speed up the decision and resolution process.

3.3.7 Changing your Milestone 2 relations

You may use the relations you created in Milestone 2 before or after normalization or after making other changes. The requirement is the relations you use must fulfill the Milestone 1 requirements (i.e., seven relationships, seven entities, one ISA, and one weak entity or additional ISA) and the Milestone 2 requirement that each relation must be populated with at least five tuples. In short, denormalization and simplification is ok, but major changes or restarting from scratch is not.

3.3.8 Where to start your implementation

You can reuse the starter code provided in Tutorial 6 and 7; in fact, this is recommended as opposed to starting from scratch. As a disclaimer, the starter code is meant to help you get started and is not necessarily representative of the requirements of the project. It is your responsibility to read through the various documents and rubrics to ensure that your project meets the given criteria. The expectations do vary semester by semester so even if you were previously enrolled in the class or know someone who has taken the class, it does not necessarily mean that their information is accurate. You must use the provisioned repository given to you by the CPSC 304 course staff.

3.3.9 CS server database setup

If you are using a department provided database (e.g., Oracle), you do not have to do any server setup. If you are using a local installation of a database, then you will have to first configure the database so that your code can connect to it when running queries. The CPSC 304 teaching team does not provide any help or troubleshooting services for local database installations. To save yourself some time, we recommend using the department-provided databases; the CS department provides an Oracle server for you to connect to. We recommend that everyone connect to the department database using their own account (as opposed to everyone using the same account).

3.3.10 TA support

The office hours close to a deadline tend to be very busy and you may not get a chance to speak to the TA. We encourage you to start the project early so you can seek help when needed. The TAs are not responsible for staying behind/offering additional hours/debugging your project remotely to help you finish your project. It is very hard to debug things remotely/asynchronously. We will likely redirect you to tutorials/technical support office hours if your question pertains to code/environment issues. You can e-mail your project TA if you have questions that are not answered already on Piazza. However, vague or pre-grading questions like "Is what we are doing correct?" will NOT be answered. Please look on the resources page and Piazza for answers to common errors.

3.3.11 Common technical issues

I am having trouble cloning the repository to my computer. It keeps saying I have the wrong username/password.

To clone the repository to your computer, you will need to use a personal access token for the password (this is identical to what you had to do for CPSC 210). If you no longer have your personal access token, regenerate a new one.

I am having some other technical issues with my repository

Issues not being able to force push or use rebase on the provided repository are beyond what the course staff can help with. We realize that this has to do with permissions, but our job is to help you with the database concepts, not help manage your git repository. Your best bet is to reach out to help@cs.ubc.ca but we are uncertain of whether they will be able to accommodate this request.

I am having technical trouble doing Tutorial 6 and/or 7

We have dedicated office hours for PHP and JavaScript help. Please see the office hours schedule on Canvas and/or Piazza.

I am having trouble executing queries through my code

First, try your query in SQL Plus (see the related tutorial for instructions on how to log in/use SQL Plus). If your query is causing an error in SQL Plus, then you need to resolve that first. A common mistake is when people try to use EXCEPT in Oracle; you will need to use MINUS instead (as a side note, Oracle 20c accepts EXCEPT but the CS department is not using that version of Oracle). If your query works in SQL Plus but not when executed through your code, then see the FAQ question below.

My design requires assertions, but I don't know how to do this in Oracle

There is no `CREATE ASSERTION` in Oracle. This discussion gives more details. There are multiple approaches for how to get around this, including this one using triggers.

I am having some other code-related issues

The FAQ page on the course website may contain an answer to your question. As some TAs are more/less experienced with certain tech stacks, we encourage you to post technical questions or issues to Piazza. This way, you can hopefully get a response from a TA or student who has seen the issue before and knows exactly how to resolve it. For in-person support, you can get help during office hours or tutorials.

4 Milestone 5 – Demo

4.1 Purpose

The purpose of Milestone 5 is to present your Milestone 4 work to your TA. This provides an opportunity for them to evaluate the functionality of your queries and other features, as well as asses your understanding of the project.

4.2 Additional Information

4.2.1 Sign-up

Each group must book a half-hour slot. Sign-up information specific to your TA will be announced closer to the date.

4.2.2 Participation

All group members must be present and on time for the demo. All group members should participate meaningfully in the demo.

4.2.3 Preparedness

The group should be ready to go when the demo starts. That is, you should not use the first part of the demo to boot up your laptop and search for the files you need. For those of you using Oracle, be sure to also have SQL Plus ready so it can be used to double check the result of a given action in your application.

If the demo starts late, no extra time will be given. In cases where the demo starts late because of an issue on the part of the teaching team, the group will be given the full 30 minutes to complete the demo.

Penalties may be given to group members who are late or do not show up. Due to the short duration of the demo, anyone who does not arrive within the first five minutes of the start of the demo will be considered as absent – even if they show up to the demo. Arriving within the first five minutes may be subject to a late penalty of 15% on the full value of the deliverable.

If you are not using Oracle, have some way for the TA to check the status of your database. You will need to let your TA know about this ahead of time so they can prepare. The team should be aware of how to quickly change the user login credentials if needed. Your TA may choose to use their own SQL Plus account for the demo.

4.2.4 Grading

Groups will not receive their demo grade right away. Your TAs will need time to look over the deliverables submitted on Canvas.

4.2.5 What the demo may look like

1. The group will be asked to demonstrate that the code used in the demo has not been changed since the milestone 4 deadline. A 25% deduction will be applied to the milestone if you are unable to show the files are the same.
    - This can be done through Git (e.g., re-pulling from the repo or running something like git log/git diff) or recompiling the code. Your TA will let you know which method they prefer.
2. The TA will ask the group if anything listed in their formal specs (list of deliverables) does *not* work. This will save a bunch of demo time, and it will help the TA with marking the deliverables.

3. The group will be asked to re-create and repopulate the tables with their .sql file. Teams should have the commands to do this ready to go so time is not wasted on dealing with constraints due to command order.
4. The group will start demonstrating their queries in whatever order they choose. During the demonstration, the TA may ask to change input values (e.g., something like “Instead of inserting this value, please insert this other value instead”).
5. After the group has finished this demo, the TA will ask them technical questions. For example, “If I wanted to change ‘this’ to ‘that,’ how would I do so and where in the code would that change have to be implemented?”

5 Milestone 6 – Peer and self-evaluation

Milestone 6 is a Canvas quiz that must be submitted on or before the deadline specified on Canvas. The quiz includes peer and self-assessment sections and must be completed individually (not with your group members).

The self-assessment is an opportunity to share your specific contributions to the project. In the peer assessment you should provide a rough estimate or your own and your team member’s contribution to the project as a percentage of the overall work. To share equally, we don’t expect a perfect split (e.g., 33.3% each), but we expect all teammates to be meaningful contributors.

If your group experienced significant collaboration issues, you could provide additional details about how tasks were distributed, alongside the percentage split. Ideally, this information should have been communicated to your instructor or TA earlier, and this section serves to supplement that as needed.

Side note: what you say here (and what is said about you) can be used in the future if you ask for letters of recommendation, so saying nice things about your teammates is also valuable!