CS2102: Database Systems
AY 2022-2023 – Semester 2

Assignment 1: SQL

INSTRUCTIONS TO CANDIDATES

1. This is an **individual** assignment. You must submit your own work.
2. This assignment consists of **TEN (10)** half-mark SQL questions. The total is 5 marks.
   - Answer **ALL** questions.
3. The deadline for the submission is 25 February 2023 at 12:00 (Noon)
4. **Late submission penalty:**
   - Two marks will be deducted for each late day *up to one late days.*
   - Submissions *after* the first late day will receive **zero** marks and will not be graded.
   - The solution will be released by **Sunday, 26 February 2023** at 12:00 (Noon) to help revision before Midterm.
5. The assignment is to be submitted using Canvas.
6. The assignment will be **auto-graded**.
   - Additional *hidden* test cases may be added to ensure no hard-coding.
   - The SQL query will be run on PostgreSQL v14.
   - You **must** follow the requirement given in **Instructions** (Section 1).

Good Luck!

1 Instructions

In this assignment, you will formulate **TEN (10)** SQL queries and copy your solutions into the provided template file `Assignment1.sql`. As the assignment will be auto-graded, it is very important that your submitted answers conform to the following requirements:

- For each question, you are to write a **single** `CREATE OR REPLACE VIEW` **SQL statement** to answer the question. You **MUST** use the view schema provided for each question without changing any part of the view schema (*i.e., view name, column names, or the order of the columns*). For example, the provided view schema for Question 1 is "`v1 (area, num_stations)`", and if your answer to this question is the query "`SELECT 'dummy' AS code, 10 AS count`", then you must enter your answer in the answer box as follows:

```
CREATE OR REPLACE VIEW q1 (code,count) AS
SELECT 'dummy' AS code, 10 AS count
;
```

- Each `CREATE OR REPLACE VIEW` statement must terminate with a semicolon (*i.e.,* `;`).

- Each answer must be a syntactically valid SQL query without any typographical errors: an SQL answer that is syntactically invalid or contains very minor typographical error will receive **ZERO (0)** marks even if the answer is semantically correct.

- For each question, your answer view must not contain any duplicate records. You may use `DISTINCT` as you see fit.

- Each question must be answered independently of other questions, i.e., the answer for a question must not refer to any other view that is created for another question.

- You are allowed to create the answer view using a single CTE statement (defining possibly multiple temporary tables). If your answer uses a CTE statement to define temporary table(s), you must not use any of the other 10 answer views `v1`, `v2`, ..., `v10`.

- If your answer uses the CTE statement, you must define at most **TWO (2)** temporary tables.

- Your answers Questions 1-9 must not use SQL constructs that are not covered in class. Question 10 aims to encourage you to explore a bit the functionality of PostgreSQL, so anything is allowed here; we also give some hints and pointers.

- Your answers must be executable on PostgreSQL.

1.1 Before Submission

Before you submit your completed template file to Canvas, check

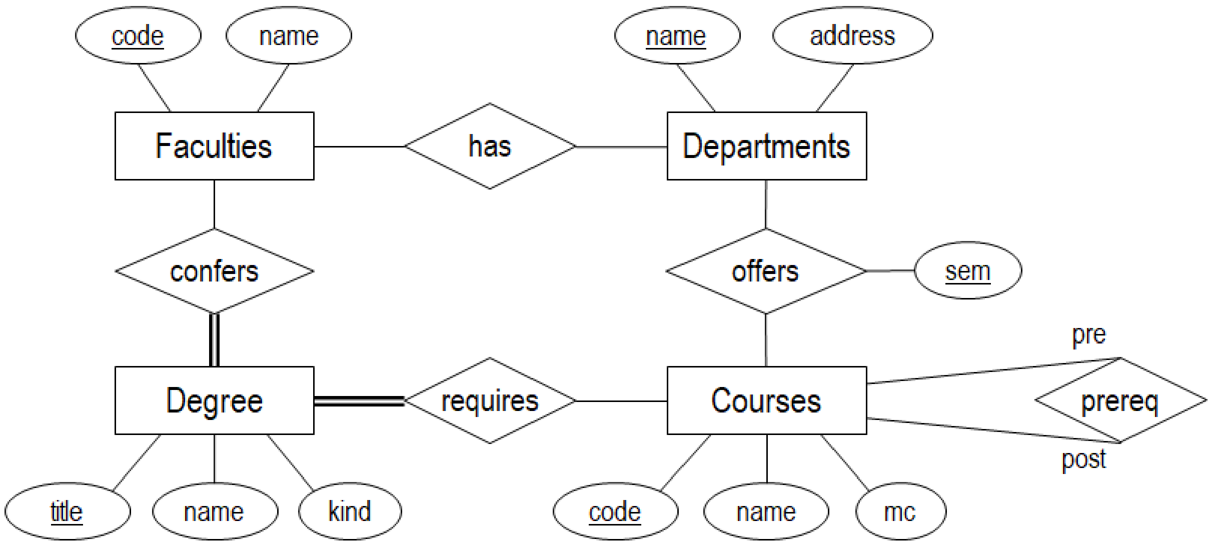
1. that you have updated the `student` view to contain your Student and NUSNET ID,
2. that your template file is valid with the easiest way to test this is to import it into the database by copy-pasting `Assignment1-DDL.sql` and `Assignment1-Data.sql`,
3. that you have renamed the template file to contain your Student and NUSNET ID (*e.g.*, `Assignment1-A0000000X-e0000000.sql`)

The last step is important so that we can be ***doubly-sure*** that we can associate all submissions correctly with all students.

2 Database

We already made the database we use for this assignment available on Canvas. However, for completeness, we will provide you here with the ER diagram and the `CREATE TABLE` statements.

2.1 ER Diagram



NOTE: The ER diagram does not follow universal schema assumption. Be careful with joins.

2.2 Database Schema

```sql
CREATE TABLE Faculties (
  code  VARCHAR(4) PRIMARY KEY,
  name  VARCHAR(50) NOT NULL
);
```

```sql
CREATE TABLE Departments (
  name    VARCHAR(50) PRIMARY KEY,
  address VARCHAR(50)
);
```

```sql
CREATE TABLE Has (
  code  VARCHAR(4) REFERENCES Faculties,
  name  VARCHAR(50) REFERENCES Departments,
  PRIMARY KEY (code, name)
);
```

```sql
CREATE TABLE Degree (
  title VARCHAR(15) PRIMARY KEY,
  name  VARCHAR(50) NOT NULL,
  kind  VARCHAR(10) NOT NULL
);
```

```sql
CREATE TABLE Courses (
  code  VARCHAR(5) PRIMARY KEY,
  name  VARCHAR(50) NOT NULL,
  mc    INT NOT NULL CHECK (mc > 0)
);
```

```sql
CREATE TABLE Offers (
  code  VARCHAR(5) REFERENCES Courses,
  name  VARCHAR(50) REFERENCES Departments,
  sem   INT NOT NULL,
  PRIMARY KEY (code, name, sem)
);
```

```sql
CREATE TABLE Prereq (
  pre   VARCHAR(5) REFERENCES Courses,
  post  VARCHAR(5) REFERENCES Courses,
  PRIMARY KEY (pre, post)
);
```

```sql
CREATE TABLE Requires (
  title VARCHAR(15) REFERENCES Degree,
  code  VARCHAR(5) REFERENCES Courses,
  PRIMARY KEY (title, code)
);
```

```sql
CREATE TABLE Confers (
  title VARCHAR(15) REFERENCES Degree,
  code  VARCHAR(4) REFERENCES Faculties,
  PRIMARY KEY (title, code)
);
```

2.3 Constraints

The following constraints are *guaranteed* but not captured by the schema. In fact, some are not captured by both ER diagram and schema.

- Faculty codes are in uppercase (*e.g., FASS, and SOC*).
- Course codes uses the following format: `XXNNN` where `X` are uppercase alphabets and `N` are numeric characters. In other words, it is always two uppercase alphabets followed by three digits.
- In the relationship `prereq`, the courses captured as `pre` is the prerequisite of the courses captured as `post`.
- The relationship `prereq` is not symmetric (*i.e., if course `C1` is a prerequisite for `C2`, then we will not have course `C2` as a prerequisite for `C1`*) and non-cyclic.
- If there are two tuples `(C1,C)` and `(C2,C)` such that `C1` and `C2` are the `pre` for `C`, then both courses `C1` and `C2` must be taken before taking `C`. In other words, the prerequisite condition is always *conjunction*.
- Every degree must be conferred by at least one faculties but there may be a faculty that do not confer any degree.
- Every degree must have –as part of its graduation requirement– a course for the student to be taken.
- All text (*e.g.,* `VARCHAR`) will never be an empty string. If the value is allowed to not exist, we will be using `NULL` value.

3 Questions

Q1. You are thinking of taking a degree, but you do not know which one. So you try to limit yourself to degree with few required courses only.

Your task for this question is, for each degree, find the number of required courses to graduate with the degree. Limit it to only courses that requires at most 3 courses.

Note that we only look for the explicitly required courses. Of course, to take the course, you need to first take the prerequisite, but we can ignore those for now.

For instance, to graduate with a BPhil (*i.e., Bachelor of Philosophy*), students only need to have taken PH400 and MA420 so it should be included. This is true even when taking into account the need to take PH300, PH200, and PH100 to actually take PH400.

NOTE: The `VIEW` is created with attribute name `code` but the first column should correspond to the degree `title`. If you have create the `VIEW` that returned the course `code` before, you may need to drop the `VIEW` before running with the updated `SELECT` statement that returns degree `title`.

```
CREATE OR REPLACE VIEW q1 (code,count) AS







;
```

Q2. Now you are thinking maybe Mathematics and Computer Science department are good departments. They are –after all– still in a rather high demand. You want to figure out the courses that they offer.

Your task for this question is to find all course codes that is offered in semester 2 (*i.e., the* `sem` *is equal to 2*) by the Mathematics department or offered in semester 1 by the Computer Science department.

There are many example of this. CS202 should be one of the output. On the other hand, MA102 should not be part of the answer because it is offered by Computer Science in semester 2.

```
CREATE OR REPLACE VIEW q2 (code) AS







;
```

Q3. Seems like Computer Science is more interesting than Mathematics. And since you are just starting your undergraduate journey, you want to restrict yourself to just the undergraduate degree.

Find all undergraduate degree title and name that is conferred by the School of Computing. A degree is an undergraduate degree when the value of `kind` is `'Undergrad'`. Include degrees that is conferred by multiple faculties.

In our sample data, there should only be three output: BComp, BComp (Sec), and BSc (Comp). Note that BSc (Comp) is confered by both SOC and SCI.

```
CREATE OR REPLACE VIEW q3 (title,name) AS









;
```

Q4. Since you are already selecting Computer Science because of its low number of required courses, you want to expand the breadth of your knowledge instead. You want to look for departments that are not bound by any faculties, maybe they offer an interesting courses!

Find all department names that is not part of any faculties. Note that you only need the department names, you do not need to find the courses they offer.

You should find only Maritime Studies from the sample data. However, you should not simply hardcode the name `'Maritime Studies'` as in the future, the University may add new departments!

```
CREATE OR REPLACE VIEW q4 (name) AS









;
```

Q5. You heard from your seniors that database is a very marketable skill. Luckily, database is part of the courses offered by Computer Science. You decided to look further into this.

Find all courses (*i.e., the code, name, and mc*) for courses with `'Database'` in its name. It can be in the beginning, the end, or the middle. You are only interested in cases where the name contains exactly the phrase `'Database'` with the initial character `D` in uppercase.

There should be three courses: *Database Systems*; *Implementing Database and Optimization*; and lastly *Distributed Database*.

```
CREATE OR REPLACE VIEW q5 (code, name, mc) AS











;
```

Q6. Now you start to have doubts. You do not want to restrict yourself to just one faculty. Maybe having a degree that is conferred by more than one faculties can help. After all, the future is uncertain and the more options you have, the better your chances are in the future.

Find all degree names (note, name and not title) as well as the faculty names such that the degree is conferred by more than one faculties.

Here you will see *Bachelor of Philosophy* and *Bachelor of Science in Computing* both appearing twice. That is intended because we need the pair to be distinct.

```
CREATE OR REPLACE VIEW q6 (degree,faculty) AS











;
```

NOTE:

From Q7 onwards, the queries are typically solved using subqueries, recursive queries, or even CTEs. For midterm, you can be rest assured that there will typically be solutions that do not involve subqueries, recursive queries, and/or CTEs.

Q7. Ok, maybe it is not so much about the choice of faculty but the departments and the courses they offered. The more courses they offered, the broader your knowledge would be. Maybe even deeper knowledge is possible. Plus, the requirement for graduation is the number of MC taken.

For each department, find the sum of all MC but only for *distinct* courses offered by the department in at least one semester. So, if a department offers the course in two different semesters then you should not count it twice. If a department is not even offering the course but it is listed as a valid course, you should not even count that course.

This may require a little elaboration. Let's start from the simpler one. First, look at Geography. It only offers two courses: FY400 and PW200. Although FY400 is offered in two semesters, it is only counted once and so the total for Geography should be 16 MCs. Therefore, `('Geography', 16)` should be part of the result.

Now consider the course Inter-Universal Teichmuller Theory[1]. It has 999 MC, but it is not offered in any semester. So it should not be counted.

Lastly, look at the Language department. It currently has no courses to offer so its total MC is 0.

```
CREATE OR REPLACE VIEW q7 (name, total) AS















;
```

[1] https://en.wikipedia.org/wiki/Inter-universal_Teichm%C3%BCller_theory

Q8. You are now more sure about your choice. You still wish to broaden your horizon, but you also want to deepen your knowledge. To do this, you want to find courses that are offered by only one faculty (*note, faculty and not department*).

For each faculty code (*e.g, SOC*), find the number of distinct course codes that is offered only by that faculty. This quickly excludes courses offered by departments that is in more than one faculties such as Computer Science.

However, this includes courses that may be offered by two different departments as long as the two departments are in the same faculty. For instance, the course PC333 Statistical Physics is offered by both Physics department and Statistics department. But because both are in SCI, it should be part of the 14 other courses offered exclusively by SCI.

Exclude faculties that do not offer any such courses. In other words, if the count is to be 0, it should not even be in the result at all.

You should see `('BIZ',5)` and this can be easily double-checked from the data. There are 5 courses offered exclusively by the Marketing department: MT101, MT201, MT303, MT304, and MT481.

For SOC, because Computer Science is in both SOC and SCI, none of the courses offered by Computer Science can be considered. This leaves us with only 4 other courses by Information System, namely: IS108, IS221, IS312, and IS406. Unfortunately we have to exclude IS101 because it is also offered by Computer Science in this sample data.

Hint: Can you solve the simpler problem of finding all courses that is only offered by one faculty first (*i.e., which tables are involved?*). Then check your understanding. Did you manage to get 14 courses for SCI?

```sql
CREATE OR REPLACE VIEW q8 (faculty, count) AS















;
```

Q9. Some courses are also offered by all departments. This means that you are allowed to work outside of your field and such work still counts towards your graduation requirements! That is a very tempting courses indeed.

As an example, typically all departments allows you to do research work such as your final year project. Ignoring the potential administrative matters that need to be satisfied, it may be possible (*probably not in NUS*) for you to do your final year project in another department.

There may be other courses too. A contender is internship, but unfortunately for the data that we have, it does not currently count because there is no internship courses offered by the Physics department.

Find all course codes as well as the course names such that the course is offered by all known departments in the database *except* departments that do not offer any courses. Note that the course can be offered in any semester, only one of the semester, or even both semesters as long as it is offered.

Note the requirement that we have to ignore departments that do not offer any courses. Since the Language department in FASS does not offer any courses, without this requirement, the result should be empty because there is one department without courses yet.

But that is not interesting. Maybe the department is newly created and they have no courses to offer yet. So, we choose this more interesting choice that the department that counts are departments that offer at least one course.

Implicitly, we should consider departments that are not in any faculty such as Maritime Studies. Luckily, they indeed offer FY400. In fact, FY400 is the only course that satisfy this requirement for our current database.

```
CREATE OR REPLACE VIEW q9 (code, name) AS
















;
```

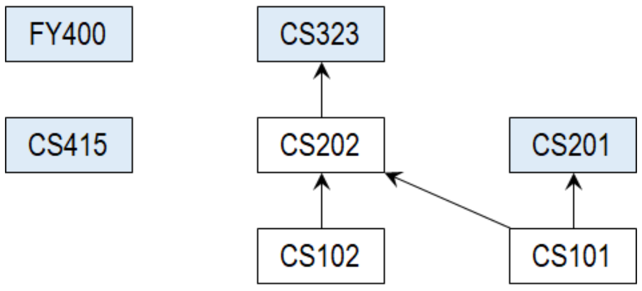
Q10. Finally, you will need to consider your graduation requirement too. Remember that there are required courses for each degree. You will need to take the courses if you wish to graduate.

Of course, to take the course, you need to have taken the prerequisite course for that course first. And to take the prerequisite course, you need to have taken the prerequisite course for the prerequisite course, ..., *ad infinitum*.

Okay, maybe not *ad infinitum* because there surely is some courses that does not require any other courses.

> We say that we can take a course if we have taken the prerequisite to that course. We say that we have completed a course if we have taken the course.
>
> To graduate, you have to complete the required courses. Find all courses that you need to complete to graduate with a `BComp`. Note that it is a `BComp` and not `BComp (Sec)`.

The only courses for this sample database are: CS101, CS102, CS201, CS202, CS323, CS415, and FY400. The requirement can be shown as diagram below with the required courses in blue.



The arrow is pointing from the prerequisite course. For instance, CS202 has two prerequisite course namely CS101 and CS102.

```
CREATE OR REPLACE VIEW q10 (code) AS



;
```

4 Check Your Solution

To help you test and debug your answers in PostgreSQL, we will provide a Web interface where you can check the correctness of your queries. But please keep the following things in minds:

- Please use this interface only to test the correctness of a query **after** you have written and run the query using your local PostgreSQL installation of the database!
- If you think that your solution is correct, but the check (partially) fails, please send us an email with your solution so we can clarify if maybe the question can be misinterpreted or our reference solution is flawed.

4.1 Web Interface

You can find the Web interface here: http://172.26.191.108/modules/cs2102/sql/. Note the machine is only accessible within the NUS network (or via VPN). To test the correctness of a query, you need to do the following steps:

- Select the query you want to check using the dropdown box (e.g., Query 1)
- Copy your solution for the query **without the** `CREATE OR REPLACE VIEW` **part** into the textbox.
- Click "Check Query" and wait for the report (depending on the complexity of your query this might take a bit)

The screenshot in Figure 1 shows an example for a report. The current answer for the test query `'Query 0'` is:

```
SELECT code
FROM   Courses
WHERE  mc = 8;
```

Feel free to use and modify the query to see how the changes affect the results of the report. For example, in the screenshot, the correct attribute `code` has been renamed to `silly_rename` to illustrate the warning the returned attribute names do not match the expected attribute names (but again, this just causes a warning as the `CREATE OR REPLACE VIEW` statements handle the final renaming of the attributes).

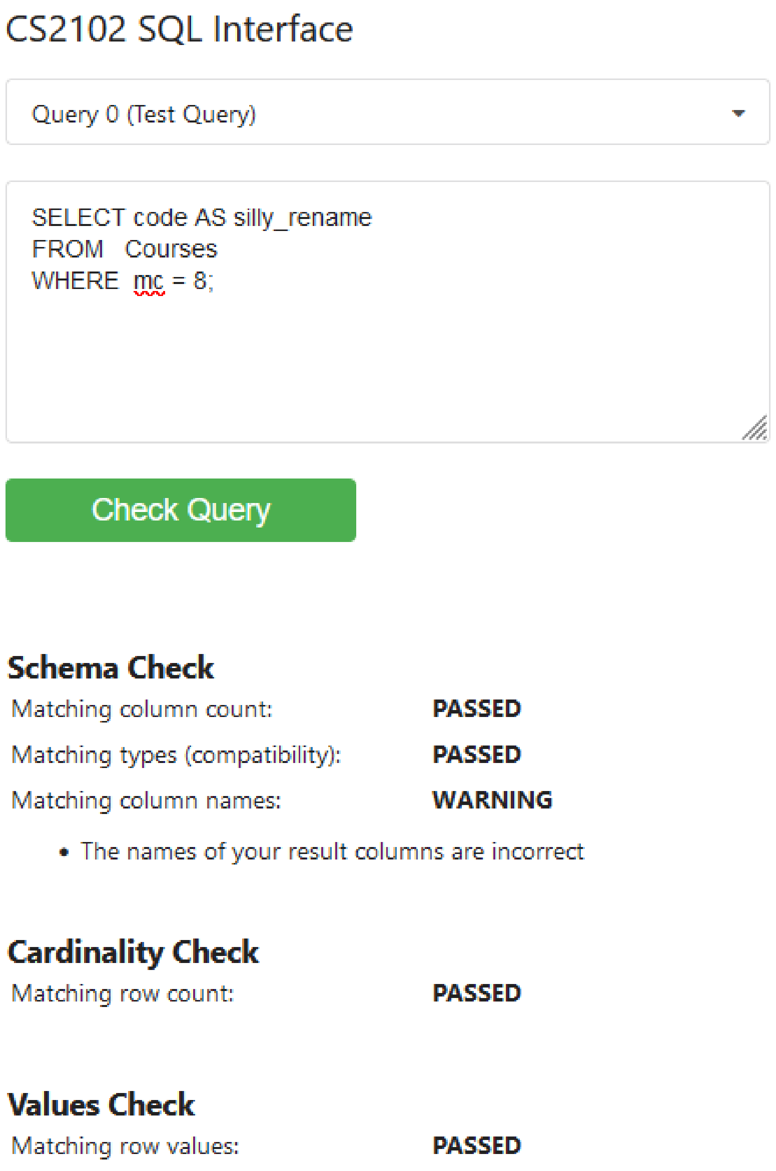
4.2 Description of Report

The report performs 3 types of checks:

- **Schema Check:** The Schema Check will check if your solution and the reference solutions have "comparable" schemas. This means:

  - Both schemas have the same number of columns.
  - Both schemas are union-compatible (i.e., the respective columns have comparable data types).
  - Both schemas should have matching column names (note this will throw only a warning as the final column names will be enforced when creating the view)

- **Cardinality Check:** The Cardinality Check will check if your solution and the reference solutions have the same number of output rows/tuples. If the cardinalities do not match, the report will indicate how many rows/tuples are missing.
- **Values Check:** The Values Check will check if your solution and the reference solutions return the same result.



```
SELECT code AS silly_rename
FROM   Courses
WHERE  mc = 8;
```

Schema Check

Matching column count:	**PASSED**
Matching types (compatibility):	**PASSED**
Matching column names:	**WARNING**

- The names of your result columns are incorrect

Cardinality Check

Matching row count:	**PASSED**

Values Check

Matching row values:	**PASSED**

Figure 1: Screenshot of SQL Web Interface to check you queries.

Important:

- Note that the checks are not independent. For example, if the Schema Check fails because the number of columns do not match, the Values Check will naturally also fail. So try to address any failed checks in a meaningful order.
- A query that passes all checks is not 100% guaranteed to be correct; although the likelihood should be high. For the evaluation of your submission, we will use a modified version of the database. This is to avoid that some students might might submiy "hard-coded" queries. For example, if a valid solution for a query would be "`SELECT COUNT(*) AS ctx FROM Requires;`" the query "`SELECT 25 AS ctx;`" would yield the same result. To catch such sneaky solution, we use a modified version of the database which will not affect correction solutions.