CS2102 Assignment 1 (SQL)

- The assignment consists of 10 one-mark questions and is due on **October 16 (Saturday) at 11.59 pm**.
- **Late submission penalty:** One mark will be deducted for each late day up to two late days; submissions after the second late day will receive zero marks and will not be graded.
- The assignment is to be submitted using LumiNUS.
- As the assignment will be auto-graded, it is very important that your submitted answers conform to the requirements in Instructions.

1. Instructions

In this assignment, you will formulate 10 SQL queries and copy your solutions into the provided template file `cs2102-assignment1-views.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 (city_name)`", and if your answer to this question is the query "`SELECT name FROM cities`", then you must enter your answer in the answer box as follows:

```
CREATE OR REPLACE VIEW v1 (name) AS
SELECT name
FROM cities
;
```

- Each CREATE OR REPLACE VIEW statement must terminate with a semicolon.
- Each answer must be a syntactically valid SQL query without any typographical errors: a SQL answer that is syntactically invalid or contains very minor typographical error will receive 0 marks even if the answer is semantically correct.
- For each question, your answer view must not contain any duplicate records.
- 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 2 temporary tables**.
- Your answers must not use SQL constructs that are not covered in class (e.g., window functions).
- Your answers must be executable on PostgreSQL.
- To help you test and debug your solutions in PostgreSQL, we will provide Web interface where you can check the correctness of your query. More details in Section 4 Check your Solutions.

Before you Submit. Before you submit your completed template file to LumiNUS, check if (a) you updated the `student` view to contain your Student ID and NUSNET ID, and (b) if your template file is valid The easiest way to test this is to import it into the database, e.g., using

```
psql -d my_cs2102_db -f cs2102-assignment1-views.sql
```

The exact command will depend on the database name and the path of your template file. If your template file is valid and does not throw any errors, you should see the following output:

```
CREATE VIEW
CREATE VIEW
...
CREATE VIEW
```

containing 11 `CREATE VIEW` messages, 1 for the `student` view and 10 for the query views.

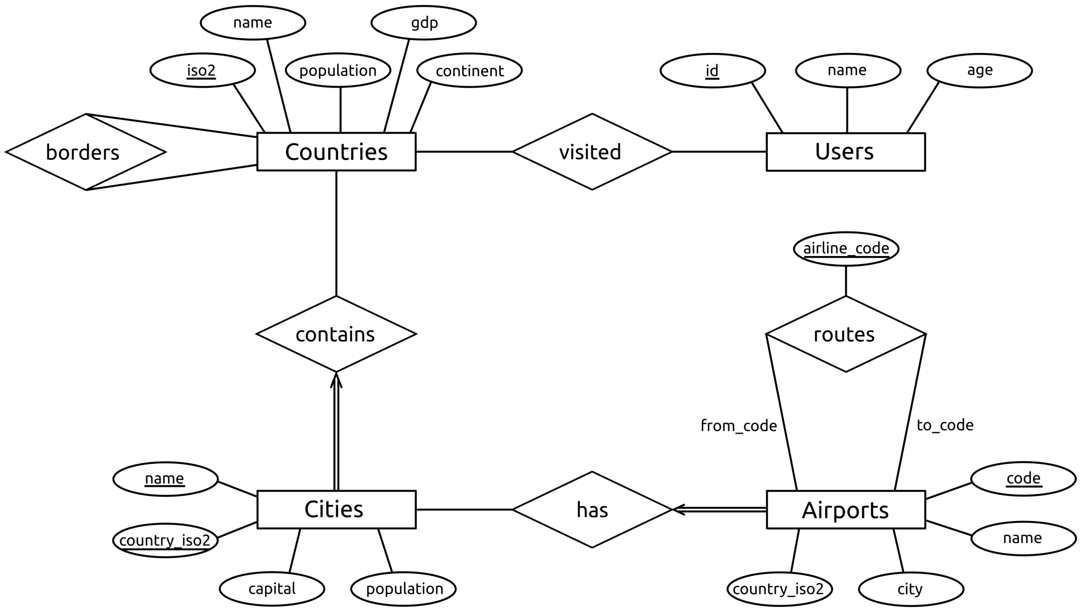
Lastly, please rename the template file `cs2102-assignment1-views.sql` to contain your student number and NUSNET id (e.g., `cs2102-assignment1-views-A00000000-e0000000.sql`)! This is to be doubly sure that we can associate all submissions correctly with all students.

2. Database

We use the from database from the Lectures 5 & 6. However, for completeness, we provide you here with the ER diagram and the `CREATE TABLE` statements.

Important: Note that the database contains a table `connections` which is created based on table `routes` to only store which destination airport can be reached from which origin airport, independent from any airline that serves this connection (cf. last `CREATE TABLE` statement below). As such, table `connections` is not explicitly represented in the ER diagram but you can, of course, use this table in your queries. This should make some queries easier compared to using `routes`.

ER Digram



Database Schema

```
CREATE TABLE countries (
 iso2 CHAR(2) PRIMARY KEY,
 name VARCHAR(255) UNIQUE,
 population INTEGER CHECK (population >= 0),
 gdp BIGINT CHECK (gdp >= 0),
 continent VARCHAR(255) NOT NULL
);
```

```
CREATE TABLE cities (
 name VARCHAR(255),
 country_iso2 CHAR(2),
 capital VARCHAR(255),
 PRIMARY KEY (name, country_iso2),
 population INTEGER CHECK (population >= 0),
 FOREIGN KEY (country_iso2) REFERENCES countries (iso2)
);
```

```
CREATE TABLE borders (
 country1_iso2 CHAR(2),
 country2_iso2 CHAR(2),
 PRIMARY KEY (country1_iso2, country2_iso2),
 FOREIGN KEY (country1_iso2) REFERENCES countries (iso2),
 FOREIGN KEY (country2_iso2) REFERENCES countries (iso2)
);
```

```
CREATE TABLE airports (
 code CHAR(3) PRIMARY KEY,
 name VARCHAR(255) NOT NULL,
 city VARCHAR(255) NOT NULL,
 country_iso2 CHAR(2),
 FOREIGN KEY (city, country_iso2) REFERENCES cities (name, country_iso2)
);
```

```
CREATE TABLE routes (
 from_code CHAR(3),
 to_code CHAR(3),
 airline_code CHAR(3),
 PRIMARY KEY (from_code, to_code, airline_code),
 FOREIGN KEY (from_code) REFERENCES airports (code),
 FOREIGN KEY (to_code) REFERENCES airports (code)
);
```

```
CREATE TABLE users (
 user_id INTEGER PRIMARY KEY,
 name VARCHAR(255) NOT NULL,
 age INTEGER
);
```

```
CREATE TABLE visited (
 user_id INTEGER,
 iso2 CHAR(2),
 PRIMARY KEY (user_id, iso2)
);
```

```
CREATE TABLE connections AS
SELECT DISTINCT from_code, to_code
FROM routes;
```

---

3 Questions

Q1: Find the names of all capital cities with a population of more than 10,000,000 people! Recall that a capital city is identifie by a value of `'primary'` for attribute `capital` in Table `Cities`.

```
CREATE OR REPLACE VIEW v1 (city_name) AS

;
```

Q2: Find the names of all countries with more than 1 official capital city! For example, South Africa has 3 official capital cities. Return the name and the number of capital cities for each country2. Recall that a capital city is identifie by a value of `'primary'` for attribute `capital` in Table `Cities`.

```
CREATE OR REPLACE VIEW v2 (country_name, capital_count) AS

;
```

Q3: Find the names of all European countries that have a larger GDP per capita than Singapore (country code 'SG'). Exlude all countries for which the information about their GDP is not available (recall that this is represented by `gdp=0`).

```
CREATE OR REPLACE VIEW v3 (country_name) AS

;
```

Q4: Find the names of all countries for which we only have the capital city in the database. For example, for the country Barbados, we only have the city Bridgetown in the database, which is the capital of Barbados. Recall that a capital city is identifie by a value of `'primary'` for attribute `capital` in Table `Cities`.

```
CREATE OR REPLACE VIEW v4 (country_name) AS

;
```

Q5: How many countries have more than 100 domestic flight connections, i.e., flight connections between airports of the same country? Return the name of the country and the number of domenstic connections, sorted by the number of domestic connections in a descending order. If there is a connection from A to B and from B to A, this counts as 2 connections.

```
CREATE OR REPLACE VIEW v5 (country_name, domestic_connections_count) AS

;
```

Q6: Find all pairs of Asian countries that share a land border! For each pair of names, always list the country with the smaller population size first! For example, Nepal and China share a land border and Nepal has a smaller population than China. So the result set should contain (Nepal, China) but not (China, Nepal).

```
CREATE OR REPLACE VIEW v6 (country_name1, country_name2) AS

;
```

Q7: Find the names of all the Asian countries that neither Marie, Bill, Sam, nor Sarah have visited! They want to travel together to Asia and, of course, go a country that is new to all of them.

```
CREATE OR REPLACE VIEW v7 (country_name) AS

;
```

Q8: Find the names of all cities in the US (country code: ‘US’) that can be reached from Changi Airport (airport code: SIN) exclusively with Singapore Airlines (airline code: SQ), either

- directly with a non-stop flight *OR*
- with 1 stop in *any* European country!

```
CREATE OR REPLACE VIEW v8 (city_name) AS

;
```

Q9: Find the names of all African countries that can be reached starting from Malaysia (country code ‘MY’) with no more than 10 land border crossings. Note that the same country can be reached with different number of crossings. In this case, report the smallest number of crossings!

```
CREATE OR REPLACE VIEW v9 (country_name, crossing_count) AS

;
```

Q10: Find the names of all airports from which you can reach all continents with only non-stop connections (independent from the airline). For example, there is no non-stop connection from Changi Airport to both North and South America in the database. So Changi Airport is not in the result set.

```
CREATE OR REPLACE VIEW v10 (airport_name) AS

;
```

4 Check your Solutions

To help you test and debug your answers in PostgreSQL, we will provide Web interface where you can check the correctness of your queries.

Important:

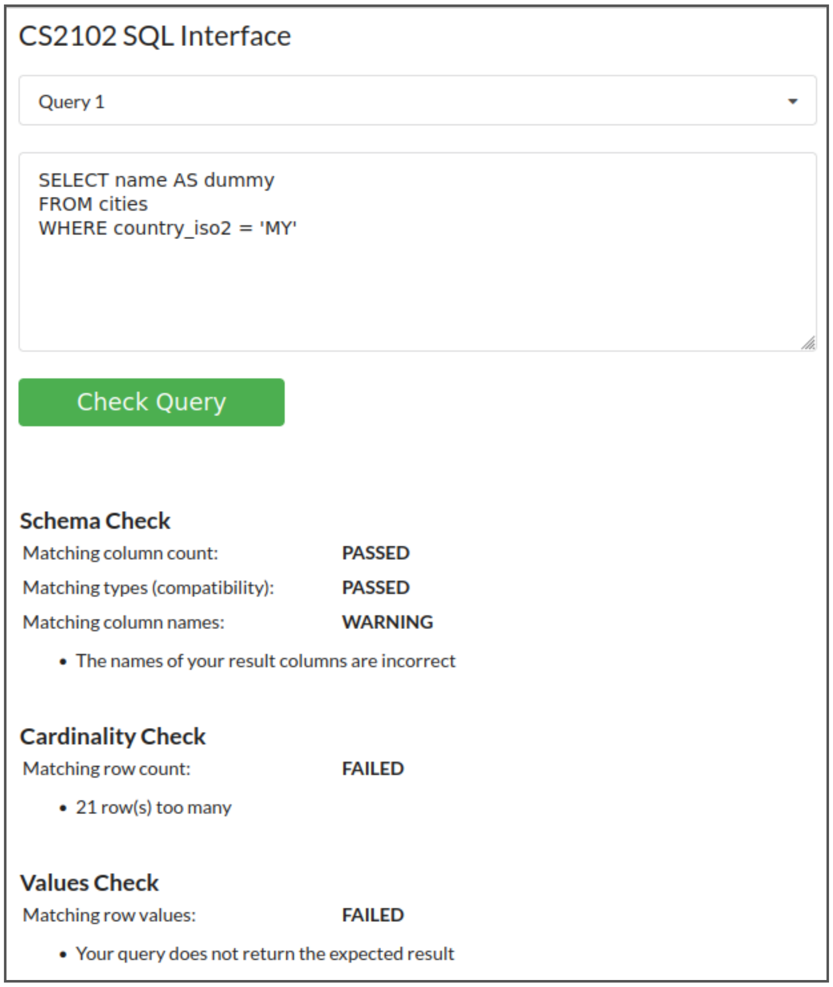
- Please use this interface only to test the correctness of a query **after** you have written and run the query using your local PostgreSQL installation of the database!
- If you think that your solution is correct, but the check (partially) fails, please send us an email with your solution so we can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

Setup

You can find the Web interface here: http://172.27.80.85/cs2102/sql/index.html. 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 folling steps:

- Select the query you want to check using the dropdown box (e.g., Query 1)
- Copy the 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 below shows an example for a report:



```
SELECT name AS dummy
FROM cities
WHERE country_iso2 = 'MY'
```

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:	FAILED

- 21 row(s) too many

Values Check

Matching row values:	FAILED

- Your query does not return the expected result

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 havs 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 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.

Important: Not 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.
