BT5110

Test 1

INSTRUCTIONS

1. This assessment **starts at 18:40**.

2. This assessment **ends at 20:30**.

3. The **maximum mark is 25.**

4. This is an **open book, open computer, and open Internet assessment**.

5. You are not allowed to communicate with anyone but members of the teaching team. Shall you use any external source of information make sure that you include a reference to the source in your answer (e.g. the URL).

6. Any student who is alleged to have committed or attempted to commit, or caused or attempted to cause any other person to commit any of the following offences: plagiarism, giving or receiving unauthorised assistance in academic work, or other forms of academic dishonesty, may be subject to disciplinary proceedings.

7. **Download this question paper and the template answer file**:

    - "test1.pdf",
    - "answers.sql",

    from the Luminus directory:

    "Files > Tests > Test 1".

8. **Download the following files**:

    - "app.sql",
    - "appfunctionality.sql",
    - "available.sql",
    - "country.sql",
    - "functionality.sql", and
    - "store.sql".

    from the Luminus directory:

    "Files > Cases > Covid19".

9. **Write your student number** in the corresponding section of the file "answers.sql".

10. **Write your answers** to the questions in the corresponding sections of the file "answers.sql".

11. **Within the 10 minutes following the end of the assessment, upload the file** "answers.sql" to the Luminus directory:

    "Files > Tests > Test 1 > Submissions".

1. The European Commission Joint Research Centre maintains a data set of mobile applications (apps) published across the world to fight the COVID-19 crisis. We consider a simplified subset of the data set stored in a relational schema comprising six self-describing tables.

   Read the SQL data definition language code in the SQL files provided and explore the provided instances of the tables to understand the application. Notice that the table country is denormalised.

   Prefer simple queries to nested queries, algebraic queries, and aggregate queries, in this order unless otherwise specified or strictly necessary. Do not use subqueries in the FROM clause unless otherwise specified or absolutely necessary. Do not create temporary tables, views, or stored functions and triggers unless otherwise specified or absolutely necessary. Your answers should be correct for the example database instance and for other instances of the same schema (for instance, new or updated apps, stores, functionalities, etc.). Your SQL code must run on PostgreSQL version 13 and above.

   (a) (3 points) (SQL) Print the names and ISO two letter codes of the different continents. The result should be similar to that in the table below in any order.

continent_name	continent_code
"Africa"	"AF"
"Europe"	"EU"
"South America"	"SA"
"Asia"	"AS"
"North America"	"NA"
"Oceania"	"OC"
"Antarctica"	"AN"

   Solution:

   ```
   SELECT DISTINCT continent_name, continent_code
   FROM country;
   ```

   (b) (3 points) (SQL) Find the contact tracing apps that are available in Europe and work for both iOS and Android operating systems. For each app, print the name of the app (rename the column as "app") and the name of the country (rename the column as "country") in which the app is available. The result should be similar to that in the table below in any order.

app	country
"COVID-19.eus"	"Spain, Kingdom of"
"Rakning C-19"	"Iceland, Republic of"
"Stopp Corona"	"Austria, Republic of"
"Corona-Warn-App"	"Germany, Federal Republic of"
"Immuni"	"Italy, Italian Republic"
"StopKorona!"	"Macedonia, The Former Yugoslav Republic of"
"SwissCovid"	"Switzerland, Swiss Confederation"
"ProteGO Safe"	"Poland, Republic of"
"Apturi Covid Latvia – SPKC"	"Latvia, Republic of"
"StopCovid France"	"France, French Republic"

   Solution:

   [Note by Mark] Read the question carefully and don't miss ANY conditions given in the question body. Around half of our students' lose marks due to assuming all apps are for "contact tracing", which is actually not true.

   ```
   SELECT a.name as app, c.name as country
   FROM appfunctionality a, available av, country c, store s1, store s2
   WHERE a.name=av.name
   AND av.country=c.code3
   ```

```
AND c.continent_name='Europe'
AND a.functionality = 'contact tracing'
AND a.name=s1.name
AND s1.os='iOS'
AND a.name=s2.name
AND s2.os='Android';
```

DISTINCT is redundant since the continent is specified.

(c) (3 points) (SQL) Find the names of the countries that are spanning over several continents. Use aggregate functions. The result should be similar to that in the table below in any order.

name
"Armenia, Republic of"
"Azerbaijan, Republic of"
"Cyprus, Republic of"
"Georgia"
"Kazakhstan, Republic of"
"Russian Federation"
"Turkey, Republic of"
"United States Minor Outlying Islands"

Solution:

```
SELECT c1.name
FROM country c1
GROUP BY c1.name
HAVING count(DISTINCT c1.continent_name) > 1;
```

(d) (3 points) (SQL) Find the names of the countries that are spanning over several continents. This is the same query as (1.c) only do not use an aggregate query. The result should be similar to that in the table above in any order.

Solution:

```
SELECT DISTINCT c1.name
FROM country c1, country c2
WHERE c1.name = c2.name
AND c1.continent_name < c2.continent_name;
```

[Note by Mark] You can obtain the same outputs by replacing "<" with either ">" or "<>", or query with the code below:

```
SELECT DISTINCT *
FROM (SELECT name
  FROM country c1
  EXCEPT ALL
  SELECT DISTINCT c1.name
  FROM country c1)
AS cross_continent_countries;
```

[Note by Mark] Need to be careful here because you cannot assume all cross continent countries only span 2 continents, e.g., France has territory in all continents except Asia: Oceania (Tahiti), Africa (Reunion), N.America (St. Pierre & Miqueion) and S.America (French Guiana).

(e) (3 points) (SQL) Find the names of the apps available in countries in Oceania that work for

all recorded operating systems. The result should be similar to that in the table below in any order. Make sure that your query produces the correct result in all possible database instances including when new apps with new operating systems are added (you may neither hardcode the operating systems as in Question 1.b nor the number of operating systems.) Do not use an aggregate query.

name
"TCC COVID Monitor"
"healthdirect"
"Coronavirus Australia"
"MyAus COVID-19"
"#BeatCovid19Now"
"Bupa Aged Care Connect"
"NZ COVID Tracer"
"COVIDSafe"

Solution:

[Note by Mark] Same as the note in Q1(b), you are suggested to be extremely sensitive to every word in the question body. "... work for ALL recorded operating system" means you are required to iterate all app stores recorded in the database rather than hardcoding "Android" and "iOS". Quiet a lot students lose marks for that reason.

```
SELECT a.name
FROM app a, available av, country c
WHERE a.name = av.name
AND av.country = c.code3
AND c.continent_name='Oceania'
AND NOT EXISTS (SELECT *
 FROM store s1
 WHERE
 NOT EXISTS (SELECT *
  FROM store s2
  WHERE s2.name = a.name
  AND s1.os = s2.os))
```

The following is not entirely satisfactory as a country may span over Oceania and another continent (USA and France have territories in Oceania etc.)

```
SELECT a.name
FROM app a, available av, country c, store s1
WHERE a.name=av.name
AND av.country=c.code3
AND a.name = s1.name
AND c.continent_name='Oceania'
GROUP BY a.name
HAVING COUNT(*) = (SELECT COUNT(DISTINCT s2.os) FROM store s2)
```

(f) (3 points) (SQL) Find the countries with the top 6 largest number of apps available (this is a dense ranking of the countries according to the number of apps). Print the names of the countries and the numbers of apps in descending order of the numbers of apps. The result should be similar to that in the table below. For full mark, make sure that your query produces the correct result for the top 7, 8 to 16 as well. Do not use "RANK()", "DENSE_RANK()", "ROW_NUMBER()", window functions and partitions (although they could be better choices in this case.) Use "GROUP BY", "ORDER BY", and "LIMIT".

name	count
"United States of America"	65
"India, Republic of"	52
"Mexico, United Mexican States"	21
"Brazil, Federative Republic of"	17
"Spain, Kingdom of"	16
"United Kingdom of Great Britain & Northern Ireland"	14

Solution:

[Note by Mark] This question asks for "countries with the top 6 largest number of apps" rather than "top 6 countries regarding the number of apps". It is not the exactly same question with the one extra question (winner and loser case to solve top-K students) that has been demonstrated in class. This question asks you to "dense rank" countries, you may search on Internet for this term.

```
SELECT c1.name, count(a1.name) as count
 FROM (SELECT DISTINCT c3.name, c3.code3 FROM country c3) AS c1
    LEFT OUTER JOIN available a1
    ON c1.code3 = a1.country
 GROUP BY c1.name
        HAVING COUNT(a1.name) IN
( SELECT DISTINCT COUNT(a2.name)
 FROM (SELECT DISTINCT c4.name, c4.code3 FROM country c4) AS c2
    LEFT OUTER JOIN available a2
    ON c2.code3 = a2.country
 GROUP BY c2.name
 ORDER BY count DESC
 LIMIT 6)
 ORDER BY count DESC;
```

One has to group by name and not by code3, because some code3 are null.

This following solution is almost correct but it counts countries spanning over two continents twice. RUS is in top 7, it should appear in top 10.

```
SELECT c1.code3, count(a1.name) as count
 FROM country c1 LEFT OUTER JOIN available a1 ON c1.code3 = a1.country
 GROUP BY c1.code3
        HAVING COUNT(a1.name) IN
( SELECT DISTINCT COUNT(a2.name)
 FROM country c2 LEFT OUTER JOIN available a2 ON c2.code3 = a2.country
 GROUP BY c2.code3
 ORDER BY count DESC
 LIMIT 6)
 ORDER BY count DESC;
```

This following solution is almost correct but misses the 0 counts. Test with top 16 (87 instead of 254 results).

```
SELECT c1.name, count(a1.name) as count
 FROM country c1, available a1
 WHERE c1.code3 = a1.country
 GROUP BY c1.name
        HAVING COUNT(a1.name) IN
( SELECT DISTINCT COUNT(a2.name)
 FROM country c2, available a2
 WHERE c2.code3 = a2.country
 GROUP BY c2.name
 ORDER BY count DESC
```

```
 LIMIT 6)
 ORDER BY count DESC;
```

The following answer is not correct. Try with top 8 and 10.

```
SELECT c.name, COUNT(*) as count
FROM available a, country c
WHERE c.code3=a.country
GROUP BY c.name
ORDER BY count
DESC LIMIT 6
```
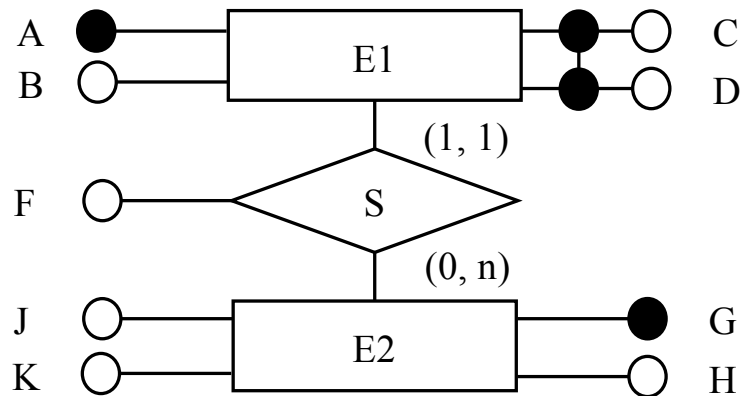
Figure 1: Entity-relationship diagram.

2. Consider the entity-relationship diagram in Figure 1.
   (a) (5 points) Translate the entity-relationship diagram into SQL following the translation rules given in the lecture. Use "TEXT" as the domain of the columns. Make sure that the design does not allow null values. The code should run for PostgreSQL version 13 and above (try it).

   Solution:

   ```
   CREATE TABLE E1S (
           A TEXT UNIQUE NOT NULL,
           B TEXT NOT NULL,
           C TEXT,
           D TEXT,
           F TEXT NOT NULL,
           G TEXT REFERENCES E2(G),
           PRIMARY KEY (C, D));
   ```

   or

   ```
   CREATE TABLE E1S (
           A TEXT PRIMARY KEY,
           B TEXT NOT NULL,
           C TEXT UNIQUE NOT NULL,
           D TEXT UNIQUE NOT NULL,
           F TEXT NOT NULL,
           G TEXT REFERENCES E2(G));

   CREATE TABLE E2 (
           J TEXT NOT NULL,
           K TEXT NOT NULL,
           G TEXT PRIMARY KEY,
           H TEXT NOT NULL);
   ```

   If the answer has three tables, for partial marks:

   ```
   CREATE TABLE E1 (
           A TEXT UNIQUE NOT NULL,
           B TEXT NOT NULL,
           C TEXT,
           D TEXT,
           PRIMARY KEY (C, D));
   ```
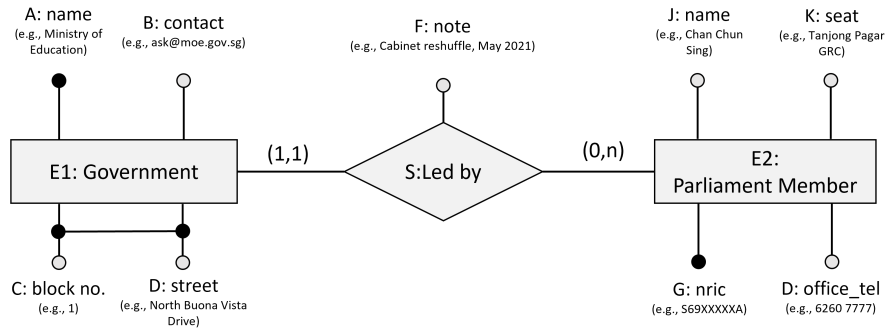
Figure 2: Entity-relationship diagram with examples.

```
or

CREATE TABLE E1 (
        A TEXT PRIMARY KEY,
        B TEXT NOT NULL,
        C TEXT UNIQUE NOT NULL,
        D TEXT UNIQUE NOT NULL);

CREATE TABLE E2 (
        J TEXT NOT NULL,
        K TEXT NOT NULL,
        G TEXT PRIMARY KEY,
        H TEXT NOT NULL);

CREATE TABLE S (
        C TEXT,
        D TEXT,
        F TEXT NOT NULL,
        G TEXT REFERENCES E2(G),
        FOREIGN KEY (C, D) REFERENCES E1(C, D),
        PRIMARY KEY (C, D, G));

or

CREATE TABLE S (
        A TEXT REFERENCES E1(A),
        F TEXT NOT NULL,
        G TEXT REFERENCES E2(G),
        PRIMARY KEY (C, A));
```

(b) (2 points) Describe in English an original real world example to which the entity-relationship diagram in Figure 1 could correspond. Give the real world meaning of entity sets E_1 and E_2, of relationship set S, and of attributes A, B, C, D, F, G, H, J, and K. Justify the candidate keys and participation constraints in the entity-relationship diagram.

Solution:

[Note by Mark] Although you have just define the primary keys in the last sub-question, you still need to present all candidate keys here. Many students submission only gave one option of defining primary key (which is correct for Q2(a)), but forgot to mention there are two candidate keys for E1 in this question.

============ Solution provided by TA Mark Meng ============

Please refer to the case that has been used in Tutorial 4, we reuse the model and propose a real-world example as shown in Fig. 2:

Candidate keys: For E1, two candidate keys can be identified: name of the government department (A), or the composite of block number (C) and street name (D). For E2, there is only one candidate key, which is the NRIC number of the parliament member (G).

Participation: As a typical commonwealth nation's cabinet, each government department (ministry level) has one and only one head, literally a (1,1) participation. The head of such government department needs be chosen from the parliament members belonging to the ruling political party. Therefore, for each parliament member, he/she may not been appointed as the head of any government department. In some cases, one parliament member can hold multiple appointments at different ministry at the same time (e.g., Minister of Law & Minister of Home Affairs). For those reasons, the participation of E2 with regard of the relationship is (0,n).

Poor examples (that should be avoided):
(1) Take non-text information as example attributes (e.g., year of founded, which is DATE);
(2) Take null-able information as example attributes (e.g., spouse);
(3) Take information has an explicit not one-to-one dependency as example attributes (e.g., number of employees, which is better to be maintained with another table);
(4) Take last name and first name as the composite primary key (C and D), because even the full name usually cannot identify a person.

– END OF PAPER –