Data Science 2021/2022: 24-Hour Take-Home Exam

This is the individual 24-hour take-home portion of the exam of Data Science (DS). The exam is made available via Digital Exam on June 16, 2022, 9:00 and your solution should be handed in via Digital Exam by June 17, 2022, 9:00. A well-formed solution to this exam consists of a PDF file including the answers to all the questions posed below.

Hand-ins for this exam must be individual. Cooperation on or discussion of the contents of the exam with other students is strictly forbidden. The solution you provide should reflect your knowledge of the material alone. The exam is open-book and you are allowed to make use of the book and other reading material of the course. If you use on-line sources for any of your solutions, they must be cited appropriately. By submitting a solution, you commit to have abided by the academic integrity expectations at the University of Copenhagen.

Question 1: Queries (40%)

A global food company enlists you to help with writing queries over their database of restaurants and meals in order to explore their patterns of dish sales around the world. The database schema contains the following relations:

```
restaurants(rid, name, num_tables, neighborhood, city, country)
meals(mid, rid, day, month, year)
meal_dishes(mid, did)
dishes(did, name, price)
```

Restaurants are recorded in the `restaurants` relation. A restaurant is identified by a restaurant ID and its attributes comprise the restaurant's name (e.g., 'Nordic Magic'), its number of tables (e.g., 12), as well as the neighborhood (e.g., 'Oesterbro'), city (e.g., 'Copenhagen'), and the country (e.g., 'Denmark') where the restaurant is located. A restaurant serves many meals, recorded in the `meals` relation, and each meal is served in a given day, month, and year. A meal is composed of orders of potentially many dishes from the company's menu, and a dish in the menu can be ordered for different meals (relation `meal_dishes`). The relation `dishes` records the dishes in the menu of the company, and includes the name (e.g., 'Olives') and price (e.g., 12.00) of each dish. Note that in the relations above, underlines denote primary keys while italics denote foreign keys.

Answer each of the queries below in the language requested. *NOTE: If the query is formulated in a language different than the one requested, the answer will not be considered.*

I. List the names of dishes ordered in meals in June 2022 that took place in restaurants in Denmark with more than 10 tables. [Language: Relational algebra]

II. For each restaurant in Denmark, list the restaurant name and the total price of all dishes ordered in meals in 2022. [Language: Extended relational algebra]
NOTE: You should only list restaurants that have at least one dish ordered in 2022. You may assume that aggregate functions follow the same semantics of SQL and that prices in the `dishes` relation are not null.

III. List the months and years in which no dishes named 'Olives' were ordered in restaurants in the country 'Greece'. [Language: SQL]
NOTE: Each month-year pair in the output should be listed only once, even if multiple instances of the same month-year pair occur across meals.

IV. List the pairs of dish names such that the corresponding dishes were ordered together in at least one meal. [Language: SQL]
NOTE: Each pair returned should correspond to names of different dishes (not necessarily different dish names!) and each pair of dishes should be considered in the output only once, i.e., pairs of dishes (i, i) should not be included and if a pair of dishes (i,j) is output, then pair (j,i) should not.

Question 2: Materialized Views (20%)

Consider again the schema of Question 1:

```
restaurants(rid, name, num_tables, neighborhood, city, country)
meals(mid, rid, day, month, year)
meal_dishes(mid, did)
dishes(did, name, price)
```

After interviewing the analysts of the global food company, you find out that the following two queries are often formulated as part of their analysis workflows:

Query 1:

```
SELECT r.name, SUM(d.price)
FROM restaurants r INNER JOIN meals m USING (rid)
                   INNER JOIN meal_dishes md USING (mid)
                   INNER JOIN dishes d USING (did)
WHERE r.country = 'Denmark'
  AND r.num_tables >= 12
GROUP BY r.name
HAVING COUNT(*) > 100
```

Query 2:

```
SELECT m.day, m.month, m.year, SUM(d.price)
FROM meals m INNER JOIN meal_dishes md USING (mid)
             INNER JOIN dishes d USING (did)
WHERE m.year = 2022
GROUP BY m.day, m.month, m.year
```

You are enlisted to improve the execution efficiency of these queries by employing a materialized view. Given this input, answer the following questions:

I. Provide the SQL definition of a **single** materialized view that can best help speed up both Query 1 and Query 2. Briefly explain (2-3 sentences) why the materialized view would be helpful.

NOTE: You do not need to specify a maintenance policy for the materialized view, i.e., you may assume that the DBA will periodically issue manual refresh commands to keep the view sufficiently up-to-date.

II. Show how to rewrite Query 1 to make use of the materialized view you propose by providing an equivalent SQL query that accesses as much as possible the materialized view and as little as possible the original tables.

III. Show how to rewrite Query 2 to make use of the materialized view you propose by providing an equivalent SQL query that accesses as much as possible the materialized view and as little as possible the original tables.

NOTE: We expect you to write your solutions to this question using PostgreSQL syntax.

Question 3: Database Design Theory (40%)

An experimental facility serving scientific research groups around the world wishes to analyze the experiment metadata originating from their instruments and engages you to derive a good database design. The experiment metadata can be directly exported from the instruments as a relation with the following schema:

```
experiment_log(scientist, lab, institution, country, experiment_date,
machine, result_path)
```

The relation lists the name of the scientist who ran the experiment, the lab, institution, and country where the scientist works, the date when the experiment was performed, the name of the machine (i.e., the instrument) used, and the path to the data file with the resulting experimental data. After discussing with the people in the experimental facility, you derived the following functional dependencies over `experiment_log`:

```
scientist → lab, institution, country
lab, country → institution
institution → country
experiment_date, machine → result_path
experiment_date, machine → scientist
```

Given this input, answer the following questions:

I. What are all the keys of `experiment_log`? Why?

II. The relation `experiment_log` is *not* in BCNF. Why?

III. Is the relation `experiment_log` in 3NF? Why or why not?

IV. Decompose `experiment_log` into a set of relations in BCNF, justifying the steps of your decomposition process.

V. Is the decomposition you have derived dependency-preserving? Why or why not?