Lab 6: Counting with SQL

Due date: Thursday, June 1, 7pm.

Note: Lab 7, the last SQL lab, will be assigned on Thursday, June 1 in class.

Lab Assignment

Assignment Preparation

This is an individual lab. Each student has to complete all work required in the lab, and submit all required materials **exactly as specified** in this assignment.

The assignment will involve writing SQL queries for different information needs (quesions asked in English) for each of the five course datasets.

You will continue using instructor's databases you used in Labs 4 and 5. The read-only versions of these databases are available on the MySQL server as databases with the following names:

```
AIRLINES
BAKERY
CARS
CSU
INN
KATZENJAMMER
MARATHON
STUDENTS
WINE
```

(note, the Labthreesixfive.com labs are set up so that you are automatically working with the right database for each question).

Each of you has been granted the `SELECT` priveleges on each of these databases (contact the instructor if this is not the case). Your queries must

be written for the tables in these databases and must run properly on them and produce correct output.

You will prepare one SQL script for each database.

The Task

You are to write and debug (to ensure correct output) the SQL queries that return information as requested for each of the information needs outlined below. Each information need in this lab can (and must) be represented by either a single SELECT statement possibly including aggregate operations, GROUP BY and HAVING clauses , or by a number of SELECT statements combined using the UNION operator. For this assignment, you will prepare one SQL script for each database.

Notes: If you are comfortable using it, use of JOIN syntax in the FROM clause is permitted (we will cover it some time during the duration of the lab), but **all** queries can be written without the use of JOIN syntax. Use of **nested queries** is still **not permitted**.

Additional Note: In queries that start with the phrase "For each YYY report ...", you are expected to include the column representing YYY in your output. For example, the query "For each grade report the sum of all classrooms" should result in a query that outputs two columns: `GRADE` and `SUM(CLASSROOM)`. This applies to all datasets and all upcoming labs as well.

Filenames. For this lab, name the SQL scripts containing your queries `<DATASET>-count.sql`. E.g., for the `CARS` dataset, the script file name is `CARS-count.sql`.

STUDENTS dataset

For the STUDENTS dataset, write an SQL script containing SQL statements answering the following information requests.

1. Report the names of teachers who have two or three students in their classes. Sort output in alphabetical order by teacher's last name.
2. For each grade, report, in a single row, all classrooms this grade is taught in.
3. For each kindergarden classroom, report the total number of students. Sort output in the descending order by the number of students.
4. For each first grade classroom, report the student (last name) who is the first (alphabetically) on the class roster. Sort output by classroom.

BAKERY dataset

Write an SQL script containing SQL statements answering the following information requests.

1. Report all customers who made more than 10 different purchases (unique receipts) from the bakery. Report first and last name of the customer, sort output in alphabetical order by last name.

2. For each flavor of cookie report total number of times it was purchased (count the actual cookies, not receipts), number of unique customers who purchased the cookie, and the total sales amount. Sort output in alphabetical order by cookie flavor.

3. For each day of the week of October 15 (Monday to Sunday) report the total number of purchases (receipts), the total number of pastries purchased and the overall daily revenue. Report results in chronological order and include both the day of the week and the date. (Note: the total amounts paid may look strange, if you are using floating points for prices.)

4. Find all purchases that totaled $25 or more. Report the customer who made the purchase, the receipt number, and the total amount of the purchase. Sort output in descending order by the total amount.

5. For each customer, count the number of times they purchased exactly five items from the bakery on a single receipt. Report last name, first name, and the number of five-item purchases sorted in chronological order by the last purchase made, breaking the ties in alphabetical order by the last name. (note: if you study the database carefully, you will discover that the maximum number of items purchased on the same receipt in is five).

CARS dataset

1. For each European car maker (reported by their short name) report the average mileage per gallon of a car produced by and the corresponding standard deviation [1]. Sort output in ascending order by the **best** mileage of a car produced by the car maker. Exclude any NULL values.

2. For each US car maker (reported by their short name), report the number of 4-cylinder cars that are lighter than 4000 lbs with 0 to 60 mph acceleration better than 14 seconds. Sort the output in descending order by the number of cars reported.

3. For each year in which honda produced more than 2 models, report the best, the worst and the average gas mileage of toyotaThis is NOT A TYPO! vehicles produced that year. Report results in chronological order.

   NOTE: Solve this query WITHOUT using NESTED QUERIES/Subqueries!

[1] Feel free to use `STD()` for all standard deviations in this lab

4. For each year from 1975 to 1979 (inclusive) report the number of Japanese cars with less than 150 horsepowers. Sort output in chronological order.

CSU dataset

Here are the queries for the CSU dataset.

1. For each campus that averaged more than $2300 in fees between 2000 and 2004 (inclusive), report the average FTE undergraduate enrollment over those years. Sort output in descending order of average enrollment.

2. For each campus for which data exists for more than 60 years, report the average, the maximum and the minimum enrollment (for all years), and the standard deviation. Sort your output in descending order by average enrollment.

3. For each campus in LA and Orange counties compute the overall revenue received from students starting the year 2001 (i.e., in the 21st century). Use "warm body" enrollment numbers (not FTEs) for your computations[2]. Sort output in descending order by the revenue.

4. For each campus that had more than 20000 enrolled students in 2004 report the number of disciplines for which the campus had non-zero graduate enrollment. Sort the output in alphabetical order by the name of the campus. (This query should exclude campuses that had no graduate enrollment at all).

MARATHON dataset

For this dataset, all times must be output in the same format as in the original dataset (in the file `marathon.csv`).

Note: please remember that the **best**, i.e., the **fastest** time is the smallest one!

1. For each sex/age group, report total number of runners in the group, the overall place of the best runner in the group and the overall place of the worst runner in the group. Output result sorted by age group and sorted by sex (F followed by M) within each age group.

2. Report the total number of sex/age groups for which both the first and the second place runners (within the group) hail from the same state.

[2]The number you compute technically is not quite the full revenue, but it's a good approximation.

3. For each full minute, report the total number of runners whose pace was between that number of minutes and the next. (That is, how many runners ran the marathon at a pace between 5 and 6 mins, how many - at a pace between 6 and 7 mins, and so on).

4. For each state, whose representatives participated in the marathon report the number of runners from it who finished in top 10 in their sex-age group (if a state did not have runners in top 10s, do not output information about the state). Output in descending order by the computed number.

AIRLINES dataset

1. Find all airports with exactly 19 outgoing flights. Report airport code and the full name of the airport sorted in alphabetical order by the code.

2. Find the number of airports from which airport `LTS` can be reached with exactly one transfer. (make sure to exclude `LTS` itself from the count). Report just the number.

3. Find the number of airports from which airport `LTS` can be reached with *at most* one transfer. (make sure to exclude `LTS` itself from the count). Report just the number.

4. For each airline report the total number of airports from which it has at least two different outgoing flights. Report the full name of the airline and the number of airports computed. Report the results sorted by the number of airports in descending order, resolve the ties in alphabetical order by the name of the airline.

INN dataset

1. For each room report the total revenue for all stays and the average revenue per stay generated by stays in the room that originated in the months of June, July, and August (combined). Sort output in descending order by total revenue. (Output full room names).

2. Report the total number of reservations in 2010 that commenced on Mondays and ended on Saturdays and the total revenue they brought in.

3. For each day of the week, report the total number of reservations longer than five (5) days that commenced on it and the total revenue these reservations brought. Report days of week as Monday, Tuesday, etc. Sort output by day of week (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday)

   (NOTE: this query is a bit nasty, but is very doable. You may need the built-in MOD() function)

4. For each room report the total number of people who stayed in it in 2010 (all reservations that started in 2010). Report full names of the rooms, sort in descending order by the total number of people.

5. For each room report how many nights in 2010 the room was occupied. Report the room code, the full name of the room and the number of occupied nights. Sort in descending order by occupied nights. (Note: it has to be *number of nights in 2010* - the last reservation in each room *may* and *will* can go beyond December 31, 2010, so the "extra" nights in 2011 need to be deducted).

   Note/Hint: This is almost an extra credit problem. While multiple solutions are possible, my solution uses SQL's `SIGN()` built-in function which returns -1 for negative numbers, +1 for positive numbers and 0 for 0.

WINE dataset

1. For each wine score value above 88, report average price, the cheapest price and the most expensive price for a bottle of wine with that score (for all vintage years combined), the total number of wines with that score and the total number of cases produced. Sort by the wine score.

2. For each red grape varietal for which there are more than 10 wines in the database, report the highest price of a case of wine. Report in descending order of the case price.

3. Report the list of wineries that produced Zinfandel wines from Sonoma Valley grapes and the names of the Sonoma Valley Zinfandel wines they produce; use one row for each winery, report each unique name of wine exactly once, sort output in alphabetical order by winery name.

4. For each county in the database, report the score of the highest ranked 2009 red wine. Exclude wines that do not have a county of origin (`'N/A'`). Sort output in descending order by the best score.

KATZENJAMMER dataset

1. For each performer (use first name) report how many times she sang lead vocals on a song. Sort output in descending order by the number of leads.

2. Report how many different unique instruments each performer plays on songs from 'Rockland'. Sort the output by the first name of the performers.

3. Report the number of times Solveig stood at each stage position when performing live. Sort output in ascending order of the number of times she performed in each position.

4. Report how many times each of the remaining peformers played `bass balalaika` on the songs where Anne-Marit was positioned on the left side of the stage. Sort output alphabetically by the name of the performer.

5. Report all instruments (in alphabetical order) that were played by all four members of Katzenjammer.

6. For each performer, report the number of times they played more than one instrument on the same song. Sort output in alphabetical order by first name of the performer[3].

Submission Instructions

You must submit nine `<DATASET>-count.sql` files. In addition, submit a simple `README` file with your name and email address.

There is no need to submit any other files.

You must submit all your files in a single archive. Accepted formats are `gzipped tar` (`.tar.gz`) or `zip` (`.zip`). The file you are submitting must be named `lab6.ext` where `ext` is one of the extensions above.

When unpacked, your archive must place the nine `<DATASET>-count.sql` files into the current directory (i.e., in the root of your `handin` directory for Lab 6 submission). No other subdirectories are needed.

Submit using handin:

```
$ handin dekhtyar 365-lab06 <file>
```

[3] Yes, you can do it without using nested queries!