Lab 4: Simple Queries

Due date: Thursday May 18, 7:00pm.

Note: Lab 5 will be released on May 18 in class.

Lab Assignment

Assignment Preparation

This is an individual lab.You have 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 nine course datasets.

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.

Information needs for each database are shown below. They are also available on `LabThreeSixFive.com`, where for each database I created a separate assignment. This will allow you to save all your work for each assignment in a file that, with some minor alterations (see submission instructions) can be submitted.

You can develop your solutions in whatever environment you like (command line mysql client, desktop IDE, or Lab365).

To make your life somewhat easier, (a) Lab365 allows you to check for correctness of results each query you submit, and (b) for the **vast majority** of Lab 4 queries, it also will display the expected results (I hid outputs of some queries, but only when I wanted you to debug a specific query by hand).

In preparing your SQL SELECT statements, **please follow the instructions below.** The goal of the lab is to get you familiar with a certain style/syntax of writing SQL statements. In future labs, we will expand the syntax, and you will be able to use a more extensive set of features. Use of SQL features outside of the proscribed list will lead to deductions, *even if you submit the correct query.*

1. **ALL information needs must be addressed with a single simple SELECT statement (i.e., a SELECT statement without grouping, aggregation and nested subqueries) and/or with UNION statements**[1].

2. For this lab, you are also not allowed to use the **explicit JOIN syntax** in the `FROM` clauses of your `SELECT` statements.

For this assignment and further SQL labs you will be working with the instructor's versions of the databases from Labs 2 and 3. The read-only versions of these databases are available on the Lab365 MySQL server as databases with the following names:

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

Starting on the date of the assignment, you have 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. Please note: every row of every resulting table must be printed in a single line.

NOTE: Please provide a comment in front of each SQL statement in each of your files. The simplest comment can just state the query number (e.g., `"-- Q3."`) for this particular database. This is very useful for the situations when for one reason or another you elected not to implement a query. (If you are saving files from Lab365, it does it for you automatically).

[1] MySQL does not support ANSI SQL `INTERSECT` (intersection) and `EXCEPT` (set difference) operations. We will discuss how to perform these operations using nested SQL queries later in the course.

Note: Please, make your queries human-readable. This means ensuring that all your queries fit 80-character lines (so that your files could be printed), and breaking the queries into multiple lines to improve readability. Use indentation where possible.

STUDENTS dataset

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

1. Find all students who study in classroom 112. For each student list first and last name. Sort the output by the last name of the student in descending order.
2. For each classroom report the grade that is taught in it. Report just the classroom number and the grade number. Sort output by grade in ascending order.
3. Find find the teacher of JEFFRY FLACHS. Report the teacher's first and last name.
4. Assuming all classrooms in the school are organized in a long row with 101 on one end, 112 on the other end, and neighboring classroom numbers being off by one (for example, neighboring classes for 109 being 108 and 110), report the teachers who teach in the classrooms next door to LORIA ONDERSMA. Report the first and the last name of each teacher, sort output in ascending order by classroom number.
5. Find all students who are in the same grade as LYNETTE HOESCHEN but have a different teacher (i.e., study in a different classroom). Report results in ascending order by classroom, and students in each classroom in ascending order by their last name.

BAKERY dataset

Write an SQL script `BAKERY-info.sql` containing SQL statements answering the following information requests.

Note: Here, and everywhere else your queries must match exactly the wording of the information need. For example, if you are asked to find the price of an `Apricot Tart`, the following query

```
 SELECT price FROM goods WHERE
CODE = '90-APR-PF';
```

is considered to be incorrect because nowhere in the query was the code `'90-APR-PF'` mentioned. (This is especially important when you are expected to produce a join of two or more tables, but instead look up the

foreign key value and use it verbatim in the query. Such queries will be marked as incorrect on the spot).

1. Find all cookies that cost between $1 and $2 (inclusive). Report the name of the cookie (flavor, food) and the price, sort output in alphabetical order by the cookie flavor.

2. Report the prices of the following items:

   - any Pie;
   - any item priced between $3.40 and $3.65 (inclusive);
   - Any Croissant except the Apple one.

   Output the flavor, the name (food type) and the price of each pastry. Output results in ascending order by price.

3. Find all customers who made a purchase on October 17 , 2007. Report the name of the customer (first, last). Sort the output in alphabetical order by the customer's last name. The output shall contain NO duplicate names.

4. Find all different Croissants items purchased on October 9, 2007. Each tart (flavor, food) is to be listed once. Sort output in alphabetical order by the Croissant flavor.

5. Find all purchases which involved two (or more) of the same Apricot-flavored item. Report the receipt number, the name of the item purchases, and the date of the purchase. Sort in chronological order.

6. Find the day on which MIGDALIA STADICK made multiple purchases (i.e., had at least two different receipts from the bakery). Report just the date (if multiple dates match the query, report them in chronological order).

CARS dataset

1. Find all pontiacs in the database (pontiac here is a model of the car) released before 1977. For each, report the name of the car and the year. Sort output by year.

2. Find all cars produced by Chrysler (the company) in 1976 and 1977. Report the name of the car and the year it was produced, sort output in ascending order by the year, and in alphabetical order within a single year.

3. Report all French and Swedish automakers. Output the full name of the automaker and the country of origin sorted alphabetically by the country, and then the full name of the automaker.

4. Find all non-four cylinder cars produced in 1979 that have a better fuel economy better than 20 MPG and that accelerate to 60 mph faster than in 18 seconds. Report the name of the car and the name of the automaker.

5. Find all American car makers which produced at least one light (weight less than 2000lbs) car between 1977 and 1979 (inclusively). Output the full name of the company and its home country sorted alphabetically by the company name. Each company should be reported just once.

6. For each Saab ('saab') released after 1971, compute the ratio between the weight of the car and its number of horsepowers. Report the full name of the car, the year it was produced and the ratio sorted in descending order by the ratio.

CSU dataset

Here are the queries for the CSU dataset. Name the SQL scrips `CSU-info.sql`

Note: See note for the **BAKERY** dataset. For this dataset, you must use the name of the campus in the query, whenever the name is provided. It is **incorrect** to replace the name of the campus with the campus id number.

1. Report all campuses (full name of campus, year of foundation) that started their work before 1920, sort output in ascending order by the year of foundation.

2. For each year between 2000 and 2004 (inclusive) report the number of students who graduated from San Diego State University Output the year and the number of degrees granted. Sort output by year.

3. Report undergraduate and graduate enrollments (as two numbers) in `'Mathematics'`, `'Engineering'` and `'Computer and Info. Sciences'` disciplines for both Polytechnic universities of the CSU system in 2004, as well as for the newest Polytechnic university, which in our database is still known as 'Humboldt State University'. Output the name of the campus, the discipline and the number of graduate and the number of undergraduate students enrolled. Sort output by campus name, and by discipline for each campus.

4. Find all situations when more than 25% of students graduated from a CSU campus in a given year (for enrollments, use the FTE number) in the 1990s. Report the name of the campus, the year when this happened, and the percentage of the campus full-time-equivalent student body that graduated that year. Sort output in chronological order, and in alphabetical order by campus name within a single year.

5. Find all disciplines and campuses where graduate enrollment in 2004 was at least four times higher than undergraduate enrollment (and where both graduate and graduate enrollments were non-zero). Report

campus names and discipline names. Sort output by campus name, then by discipline name in alphabetical order.

6. Report the total amount of money collected from student fees (use the full-time equivalent enrollment for computations) at `'Fresno State University'` for each year between 2002 and 2004 inclusively, and the amount of money collected from student fees per one full-time equivalent faculty. Output the year, the two computed numbers sorted chronologically by year.

7. Find all campuses where enrollment in 2003 (use the FTE numbers), was higher than the 2003 enrollment in `'San Jose State University'`. Report the name of campus, the 2003 enrollment number, the number of faculty teaching that year, and the student-to-faculty ratio. Sort output in ascending order by student-to-faculty ratio.

INN dataset

For the INN dataset, create a SQL script file `INN-info.sql` with SQL queries for the following information needs. (When no year is supplied in the query descriptions below, assume 2010).

Note: If the full name of a room is provided in the question, you cannot replace with with a three-letter code in the text of the query.

1. Find all `traditional` rooms with a base price above $170 and two beds. Report room names and codes in alphabetical order by the code.

2. Find all July reservations (i.e., all reservations that both start AND end in July) for the 'Frugal not apropos' room. For each reservation report the last name of the person who reserved it, check-in and check-out dates, the total number of people staying and the total cost of the reservation. Output reservations in chronological order.

3. Find all rooms occupied on September 23, 2010. Report full name of the room, the check-in and checkout dates of the reservation. Sort output in alphabetical order by room name.

4. For each stay of `GRANT KNERIEN` in the hotel, calculate the total amount of money, he paid. Report reservation code, checkin and checkout dates, room name (full) and the total amount of money the stay cost. Sort output in chronological order by the day of arrival. Output check-in and check-out dates in the "Month Day" format (e.g., 'July 19', no need to report year).

5. Find out all modern room reservations that overlapped in time with the stays of FRITZ SPECTOR. Report the full name of the room, the last name of the person placing the reservation, the checkin date and the number of nights. Sort output in chronological order, and in

alphabetical order by room name for reservations that started on the same day.

6. Report all reservations in rooms with double beds that contained four adults. For each reservation report its code, the full name and the code of the room, check-in and check out dates. Report reservations in chronological order, and sorted by the three-letter room code (in alphabetical order) for any reservations that commenced on the same day. Report check-in and check-out dates in the format 'Day Mon' (day followed by the abbreviated month), no year is required.

MARATHON dataset

For this dataset, all times must be shown in the output in the same format as in the original dataset (in the file `marathon.csv`). Also, please give time and pace columns in your output appropriate column headers (in your SQL commands). The information needs are below. Name the file `MARATHON-info.sql`.

1. Report the time, pace, name, and the overall place of all runners from LITTLE FERRY, NJ, sort output in order of finish.

2. Report names (first, last), times, overall places as well as places in their gender-age group for all female runners from `QUNICY, MA`. Sort output by overall place in the race.

3. Find the results for all 27-year old female runners from Rhode Island (RI). For each runner, output name (first, last), town, their place in the race and their place in their sex-age category, and the running time. Sort by time.

4. Find all duplicate bibs in the race. Report just the bib numbers. Sort in ascending order of the bib number. Each duplicate bib number must be reported exactly once. (Note: without restrictions on what SQL functionality to use, this query can be implemented in a number of valid ways. Your task is to implement it using the SQL SELECT features allowed in this lab).

5. List all runners who took first place and second place in their respective age/gender groups. For each age group, output name (first, last) and age for both the winner and the runner up (in a single row). Order the output by gender, then by age group.

AIRLINES dataset

For the AIRLINES dataset, create a SQL script file `AIRLINES-info.sql` with SQL queries for the following information needs. You may not substitute numeric codes for airlines in the place of airline names in the queries

below. You may use three-letter airport abbreviations whenever they are used in the questions.

1. Find all airlines that have at least one flight out of the Akutan (KQA) airport. Report the full name and the abbreviation of each airline. Report each name only once. Sort the airlines in alphabetical order.

2. Find all destinations served from the KQA airport by Delta. Report flight number, airport code and the full name of the airport. Sort in ascending order by flight number.

3. Find all other destinations that are accessible from KQA on only Delta flights with exactly one change-over. Report pairs of flight numbers, airport codes for the final destinations, and full names of the airports sorted in alphabetical order by the airport code.

4. Report all pairs of airports served by both Delta and JetBlue. Each pair must be reported exactly once (if a pair X,Y is reported, than a pair Y,X is redundant and should not be reported). For each airport, report its code and its full name.

   Note: while my query reports pairs in a specific order (first airport-second airport), the query that results in reporting the same pairs in any order will be considered correct. Use "Show Expected Result" to ensure that you are reporting the same pairs.

5. Find all airports served by ALL five of the airlines listed below: `Delta`, `Frontier`, `USAir`, `UAL` and `Southwest`. Report just the airport codes, sorted in alphebetical order.

6. Find all airports that are served by at least three Delta flights. Report just the three-letter codes of the airports — each code excactly once, in alphabetical order.

WINE dataset

Create a SQL script `WINE-info.sql` containing SQL statements representing the following information needs.

1. Find all 2008 Zinfandels produced in Napa Valley (Appellation). Report the name of the wine, the winery it is produced by, and the wine score, sort in descending order by the score.

2. List all white grape varieties for which at least one wine of the 2009 vintage is rated at 90 points or above in the database. Each grape variety needs to be reported once. Sort the output in alphabetical order.

3. List all Sonoma county appellations for which the database contains at least one rating for a `'Grenache'`. For each appellation list its

name and county. Sort output in alphabetical order by county, then by appellation name. Report each appellation once.

4. List all vintage years in which at least one `Zinfandel` from Sonoma County (any appellation) scored above 92. Each year needs to be reported once. Sort in chronological order.

5. A case of wine is 12 bottles. For each `Altamura` (name of the winery) wine compute the total revenue assuming that all the wine sold at the specified price. Report the name of the wine, its vintage wine score and overall revenue. Sort in descending order by revenue. Exclude NULL values.

6. Compute the total price of a bottle of `Kosta Browne`'s `Koplen Vineyard` 2008 `Pinot Noir`, two bottles of `Darioush`'s 2007 `Darius II Cabernet Sauvingnon` and a bottle of `Kistler`'s `McCrea Vineyard` 2006 `Chardonnay`. Report just the one number.

KATZENJAMMER dataset

Create a SQL script `KATZENJAMMER-info.sql` containing SQL statements representing the following information needs.

1. Report, in order, the tracklist for 'Le Pop'. Output just the names of the songs in the order in which they occur on the album.

2. List the instruments each performer plays on 'To the Sea'. Output the first name of each performer and the instrument, sort alphabetically by the first name (and by the name of the instrument for the same person if needed).

3. List all instruments played by Solveig at least once during the performances. Report the instruments in alphabetical order (each instrument needs to be reported exactly once).

4. Find all songs that featured toy piano playing (by any of the performers). Report song titles in alphabetical order and the name of the person who played the harmonica.

5. Find all instruments Turid ever played on the songs where she sang lead vocals. Report the names of instruments in alphabetical order (each instrument needs to be reported exactly once).

6. Find all songs where the lead vocalist is not positioned center stage. For each song, report the name, the name of the lead vocalist (first name) and her position on the stage. Output results in alphabetical order by the song. (Note: if a song had more than one lead vocalist, you may see multiple rows returned for that song. This is the expected behavior).

7. Find a song on which Anne-Marit played three different instruments. Report the name of the song. (The name of the song shall be reported exactly once)

8. In the order of columns right - center - back - left, report the positioning of the band during `'A Bar In Amsterdam'`. (just one record needs to be returned with four columns containing the first names of the performers who were staged at the specific positions during the song).

Submission Instructions

You must submit nine `<DATASET>-info.sql` files. In addition, submit a simple `README` file with your name and email address. Your `<DATASET>-info.sql` files must contain a header comment with your name as well.

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 `lab4.ext` where `ext` is one of the extensions above.

When unpacked, your archive must place the nine `<DATASET>-info.sql` files into the current directory (i.e., in the root of your `handin` directory for Lab 4 submission). No other subdirectories are needed.

Submit using handin:

```
$ handin dekhtyar 365-lab04 <file>
```