Lab 4: Simple Queries

Due date: Tuesday, May 4, **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.

The Task

You are to write and debug (to ensure correct output) the SQL queries that return information as requested in each of the information needs outlined below. **All** information needs can be addressed with a single simple SELECT statement (i.e., a SELECT statement without grouping, aggregation and nested subqueries) and/or with UNION, MINUS and INTERSECT statements.

For this assignment, you will prepare one SQL script for each database. In addition to SQL statements you may need to include some SQL*plus formatting instructions to ensure that your output looks good. In particular, every row of every resulting table must be printed in a single line. If that means changing the size of the line - do it. Similarly, there should not be awkward pagination of the answers — change page size as needed. I also recommend running

```
set feedback 1
```

command to obtain tuple counts for ANY number of returned tuples.

Please Note: Each SQL query you will write for this assignment will result in **relatively few tuples** returned. If your SELECT statement returns *hundreds of tuples* — **it is wrong!** Debug it! For some information needs you can predict how many tuples there should be in the answer (sometimes - just one). When in doubt - *examine the data files* to find which tuples should be returned.

STUDENT dataset

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

1. Find all students who attend second grade. For each student list first and last name.
2. Find all grades represented in the database. Report each grade just once, sort grades in ascending order.
3. Find all students whose first name is `ELVIRA`. Return the last name, the grade and the classroom for each of them.
4. Find all classrooms in which kindergaden is taught. Report each classroom exactly once. Report classrooms in ascending order.
5. Find all students taught by `ALFREDA SUGAI`. Output first and last names of students.
6. Find all teachers who teach 4th grade. Report first and last name of each teacher (each name to be reported once).
7. For each teacher, report the grade (s)he teaches. Each name has to be reported exactly once. Sort the output in ascending order by grade.
8. Find all students from the 4th grade whose teacher is NOT BILLIE KRIENER. For each student report their name and classroom number.

BAKERY dataset

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

1. Find all cakes on the menu. For each item output the flavor, the name (food type) of the item, and the price. Sort your output in ascending order by price.
2. Find all pastries that are not cookies whose price is between \$1.00 and \$1.20. Output the flavor, the name of the pastry and the price. Sort your output in ascending order by price.

3. Find all dates on which a `Lemon Tart` was purchased. Report just the dates (each date - once) in ascending order.

4. Find all customers who purchased `'Almond Tart'`. Output only the first and the last names of the customers and the date of purchase (thus if the same customer purchased an almond tart on different occasions, his/her name will be reported twice). Sort by the last name of the customer.

5. Find all types of pastry purchased by `STEPHEN ZEME`. Each pastry type (flavor, food) is to be listed once. Order output by the type of pastry (`food`).

6. Find all customers who made a purchase on October 18, 2007. List just their first and last names.

7. Find all dates on which `SIXTA ENGLEY` made multiple purchases. Report each date exactly once, output dates sorted in ascending order.

8. Find all customers who purchased (at some point in time, possibly during different trips to the bakery) both a `Chocolate Croissnat` and a `Vanilla Eclair`. Output first and last names of each customer. (each name must be reported once).

9. Find all customers who DID NOT make a purchase on October 1, 2007. List their first and last names.

10. Find all days on which DAVID CALLENDAR bought at least one cookie. Report the dates sorted in ascending order.

CARS dataset

1. Find all 6-cylinder cars produced in 1973. For each car report its full name (e.g., `'plymouth fury iii'`) and acceleration. The list should be displayed in ascending order by acceleration.

2. Find all 4-cylinder cars that accelerate from 0 to 60mph in 12.5 seconds or less produced in the 1980s. For each car report its full name and year or production, acceleration time and horspower.

3. Find all cars with with better gas milage **and** smaller weight than the 1973 'toyota carina'. Output the full name and year or production, horsepower and gas milage for each car.

4. List all 1974 cars produced by `'Ford Motor Company'`. Output just the full names of the cars.

5. Find all Eurpopean car makers which produced at least one light (weight less than 2000lbs) car between 1973 and 1978 (inclusively). Output the full name of the company and its home country. Each company should be reported just once.

6. Find all 6-cylinder cars produced by Italian automakers. Report the full name of the car, the year it was produced and the name of the maker.

7. For each 4-cylinder car produced in 1970 report its full name, the full name of the manufacturer, the name of the country of origin of the manufacturer and the manufacturer's continent.

CSU dataset

Here are the queries for the CSU dataset.

1. List all the counties in which CSU campuses are located. Each county shall be listed exactly once.

2. All CSU campuses in ascending order by the year of establishment. For each campus list its name.

3. For each year between 1990 and 1994 (inclusive) report the number of students who graduated from Cal Poly (San Luis Obispo). Output the year and the number of graduates. Sort output by year.

4. Find all years in which the number of graduates from California State University-San Marcos exceeded the number of graduates from California State University-Stanislaus. Output the years, and the respective numbers of degrees granted for each of the two campuses. (here we assume "graduates" = "degrees granted").

5. Report total (both undergraduate and graduate) enrollments in `'Engineering'` disciplines for each San Diego, Ventura and Orange county campus in 2004. Output the full name of the campus, the country it resides in and the enrollment. Sort the output in ascending order by the enrollment.

6. Find campuses where the faculty FTE has increased in two consecutive years. Report the full campus name, and the two consecutive years and the FTEs for each of them. (Note, if such increase in FTEs happened on a single campus more than once, report all such occurrences). Sort the output in ascending order by the first of the pair of years.

   *Note: increase on **two consecutive years** means that in year X, FTE increased comparing to year X-1, and the in year X+1 FTE increased comparing to year X.*

7. List all campuses where campus fees in 2005 were greater than those at San Jose State University. For each campus, list its full name and the fee amount. Sort the output by the fee amount.

8. For each campus with enrollment (use student enrollment FTE) over 15,000, report the student to faculty ratio in 2004. (Use the student enrollment FTE and the faculty FTE for the year.) Output the full name of the campus, student enrollment, faculty size and the student-to-faculty ratio. Output in ascending order by the computed ratio.

9. Find all campuses and disciplines where in 2004 graduate enrollment exceeded undergraduate enrollemnt. Report the name of the campus, the name of the discipline and graduate and undergraduate enrollment numbers.

10. Find all campuses where the 2004 enrollment in Engineering exceeds the 2004 enrollment in Business and Management. Report campus names and enrollments in the two disciplines.

MARATHON dataset

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

1. Find the results of all male runners from the `'40-49'` age group from `DARTHMOUTH, MA`. Output first and last name, their place in the group and their time.

2. Find the results for all 51-year old male runners. For each runner, output name (first, last), town, state, time running and gender. Sort by gender, then by time.

3. Find all women who ran the race faster than the race participant with Bib number 381. Report their names (first, last), age categories and times.

4. List all runners who took second place in their respective age/gender groups. For each runner, output name (first, last), age, gender, age/gender group, and overall place in the race.

5. List all female runners whose time was between 1:33:11 and 1:39:40. For each runner, report the name (first, last), age group, bib number and the time.

6. List all runners who arrived to the finish line at the same pace as `NORBERT FELDSTEIN`. For each runner report their name, gender and age group.

7. List all male runners who ran faster than the runner who placed at number 6 in the 40-49 age category for men, but slower than JAMA PEAD. Report the names (first, last) and home towns and states for each runner.

AIRLINES dataset

1. Find all airlines that have at least one flight out of `ALX` airport. Report the full name and the abbreviation of each airline. Report each name only once.

2. Find all airlines that have at least one flight to `TDW` airport. Report the full name and the abbreviation of each airline. Report each name only once.

3. Find all airlines that have at least one flight from `ALX` airport and at least one flight to `TDW` airport. Report the full name and the abbreviation of each airline. Report each name only once.

4. Find all airlines that have at least one flight between `ANP` and `ANY` airports. Report the full name and the abbreviation of each airline. Report each name only once.

5. Find all airports with a direct flight from `ANY` airport. For each airport report its full name and the airport code.

6. Find all airports no direct flight from `ANY` airport. For each airport report its full name and the airport code.

7. We want to fly from `ANY` airport to `ASY` airport with just a single connection. Find the list of connecting airports (i.e. airports that have a flight from `ANY` and a flight to `ASY`). For each airport report its full name and the airport code. Report each airport only once.

8. We want to fly from `ANY` airport to `ASY` airport with just a single connection. For each such connection, specify the names of the airlines for each of the flights.

Submission Instructions

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 `lab04.ext`.

The archive shall contain six directories: `AIRLINES`, `CARS`, `CSU`, `BAKERY`, `STUDENTS` and `MARATHON`.

Each directory shall contain the following SQL scripts:

- Database creation script. (e.g., `CARS-setup.sql`). Use the scripts from Lab 2 and Lab 3 submissions.
- Table creation script from Lab 3. (e.g., `CARS-insert.sql`).
- The cleanup script (e.g., `CARS-cleanup.sql`). Use the scripts from Lab 2 and Lab 3.
- **NEW script.** One script per database, containing all SQL statements and any `SQL*plus` statements needed for formatting. Name the script `<DATASET>-lab04.sql` (e.g., `CARS-lab04.sql`.