Lab 4: Simple Queries

Due date: Tuesday, May 3, **midnight**!.

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 for each of the information needs outlined below. The information needs can be addressed with a simple SELECT statement (i.e., a SELECT statement without grouping, aggregation and nested subqueries) and/or with UNION, MINUS and INTERSECT statements. However, each information need **must be met** with a **single** SQL statement.

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.

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.

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 attend fifth 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 descending order.
3. Find all students whose first name is `CYRUS`. Return the last name, and the grade for each of them.
4. Find all classrooms in which first grade is taught. Report each classroom exactly once. Report classrooms in ascending order.
5. Find all students taught by `JEROME COVIN`. Output first and last names of students sorted in alphabetical order by their last name followed by their first name.
6. Find all teachers who teach fourth grade. Report first and last name of each teacher (each name to be reported once) and the classroom they teach in. Sort output by classroom in ascending order.
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 and then by teacher's last name within each grade.
8. List first and last names of all students whose first names start with letter `'R'`. Sort the output in ascending order by student's grade and within grade – by in alphabetical order by their last name.

BAKERY dataset

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

Note: 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 eclairs on the menu. For each item output the flavor, the name (food type) of the item, and the price. Sort your output in descending order by price.
2. Find all pastries that are not chocolate-flavored and not tarts whose price is between \$3.00 and \$7.00. Output the pastry code, its flavor, the name of the pastry and the price. Sort your output in ascending order by price.
3. Find all dates on which a(t least one) `Blueberry Tart` was purchased. Report just the dates (each date - once) in ascending order.
4. Find all customers who purchased `Lemon Cake`. Output only the first and the last names of the customers. Sort by the last name of the customer.
5. Find all types of pastry purchased by `ALMETA DOMKOWSKI`. Each pastry type (flavor, food) is to be listed once. Order output by the type of pastry (`food`).
6. Find all different pastries purchased on October 2, 2007. List each pastry type once. List flavor and pastry type and the price. Sort the output in ascending order by price.
7. Find all dates on which `SHARRON TOUSSAND` made multiple purchases. Report each date exactly once, output dates sorted in ascending order.
8. Find all customers who purchased (during the same trip to the bakery) both an `Apple Tart` and an `Apricot Tart`. Output first and last names of each customer. (each name must be reported once).

CARS dataset

1. Find all German car makers in the database. Report the full names of the car makers sorted in alphabetical order.
2. Find all models[1] produced by General Motors (`'gm'`) stored in the database. Report just the names of the models in alphabetical order.
3. Find all Oldsmobiles in the database. For each, report the make and the year. Sort output by year.

[1] Just models, not makes.

4. Find all cars made by Japanese automakers that were released in 1973. For each car report its maker, it make and it engine displacement. Sort the output by the engine displacement.

5. Find all 5-cylinder cars in the database. For each car report its full name (e.g., `'plymouth fury iii'`) and acceleration. The list should be displayed in ascending order by acceleration.

6. Find all 4-cylinder cars that accelerate from 0 to 60mph in 14 seconds or less produced in the 1980s. For each car report its full name and year or production, acceleration time and horsepower.

7. Find all cars that have better gas milage than the 1980 `renault lecar deluxe`. For each vehicle report its make and year as well as the acceleration and the horsepower rating. Report the results sorted in ascending order by the acceleration.

8. Find all non-Volkswagen[2] cars produced in 1980 and 1981 that have better gas milage than the 1980 `renault lecar deluxe`. For each vehicle report its make and year as well as the acceleration and the horsepower rating. Report the results sorted in ascending order by the acceleration.

9. Find all Eurpopean car makers which produced at least one light (weight less than 2000lbs) car between 1972 and 1975 (inclusively). Output the full name of the company and its home country. Each company should be reported just once.

10. Find all 3-cylinder cars. Report the full name of the car, the year it was produced, the acceleration and the gas milage of the car and the name of the maker. Sort the output by gas milage.

CSU dataset

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

1. Report all CSU campuses located in the San Diego county. Output the full name of the CSU and its location.

2. For each year between 2000 and 2004 (inclusive) report the number of students who graduated from California Polytechnic State University-San Luis Obispo. Output the year and the number of graduates. Sort output by year.

3. Report all years when the enrollment in California State University-Dominguez Hills was between 10,000 and 12,000 students. For each year, report the enrollment, sort output in ascending order by the number of enrolled students.

[2] I.e., cars produced by automakes other than Volkswagen.

4. Report total (both undergraduate and graduate) enrollments in `'Computer and Info. Sciences'` disciplines for each Ventura, San Luis Obispo, Santa Clara and Alameida county campus in 2004. Output the full name of the campus, the county it resides in and the enrollment. Sort the output in ascending order by the enrollment.

5. Find all campuses where in 2005, reported student fees were higher than those in California State University-Bakersfield. Report the full name of the campus, the county it is and the fee. Sort output in descending order of the fee value.

6. Report all disciplines in which undergraduate enrollment in 2004 in California State Polytechnic University-Pomona exceeded the enrollment in California Polytechnic State University-San Luis Obispo. Report the discipline names and the enrollments in both universitied. Output the discipline names in alphabetical order.

7. Find all years in which the number of graduates from Sonoma State University was smaller than the number of graduates from Humboldt State University. Output the years, and the respective numbers of degrees granted for each of the two campuses. (here we assume "graduates" = "degrees granted"). Sort output in ascending order by year.

8. Find all years in which (a) the number of graduates from Sonoma State University has exceeded the number of graduates from Humboldt State University, while (b) the number of students enrolled in Sonoma State University was less than the number of students enrolled in Humboldt State University. Report just the years in chronological order.

9. For each campus with enrollment (use student enrollment FTE) over 22,000, report the student to faculty ratio in 2003. (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.

10. Find all campuses that had in 2004 better student to faculty ratio[3] (use both enrollment FTE and faculty FTE fields) than California Polytechnic State University-San Luis Obispo. For each campus report its full name, the student to faculty ratio, the enrolled and faculty FTEs. Sort output in ascending order by the ratio.

INN dataset

For the INN dataset, create a SQL script file `INN-info.sql` with SQL queries for the following information needs.

1. Find all rooms whose decor is something other than `'traditional'`. For each room report its code, the full name of the room, max occu-

[3] Smaller is better.

pancy and the base rate. Sort the output in ascending order by the base rate.

2. Find all September reservations (a.k.a., all reservations that both start AND end in September) for the 'Interim but salutary' room. For each reservation report reservation code, checkin and checkout dates, the total number of people staying and the daily rate. Output reservations in chronological order.

3. Find all rooms occupied on August 17, 2010. Report full name of the room, number of beds, max occupancy rate, decor and the base rate. Sort output in alphabetical order by room name.

4. Find all rooms that are not occupied on September 1 and 2. Report room code, full name of the room and the base rate. Sort output in alpabetical order by room name.

5. Find all customers who stayed at the inn more than once during the year, with one of the stays falling completely inside the month of October. Report customer names (first, last) in alphabetical order. Each name shall be reported once.

6. For each reservation that starts on May 1 report the room name, nightly rate, number of nights spent and the total amount of money paid. Sort output in descending order by the number of nights stayed.

7. Find the names of all people[4] staying at the inn at the same time as `ANNIS YESSIOS`. Sort the output in alphabetical order by last name.

   (hint: you may use the fact that the stay of `ANNIS YESSIOS` was only for one night).

8. Find all occupants of `'modern'` rooms on the night of Halloween. Report names of occupants (last, first), full name of the room they occupy, total number of occupants and the checkin date. Sort the output in chronological order by checkin date.

9. Find all reservations for February (both checkin and checkout dates) where one adult is staying with two or more kids. Report reservation code, full room name, checkin and checkout dates and the number of kids. Sort in chronological order by checkin date, then by checkout date.

10. Order all reservations that commenced on a Friday of the month of June[5] by the total amount paid for the reservation. For each reservation report the room name, the last name of the person staying, checkin date, number of nights and the total paid.

[4] We only know the names of the people who made the reservations, so only those names are subject to the query.

[5] look up the dates in the calendar

MARATHON dataset

For this dataset, all times must be outputed in the same format as in the original dataset (in the file `marathon.csv`). The information needs are below. Name the file `MARATHON-info.sql`.

1. Find all towns in the state of Connecticut, runners from which participated in the competition. Sort the town names in alphabetical order. Each town needs to be reported only once.
2. Find the results of all male runners from the from `NEW MILFORD, CT`. Output first and last name of the runner, their overall place in the race and their time. Sort the output in ascending order by the overall place.
3. Find the results for all 25-year old female runners. For each runner, output name (first, last), town, state, and the running time. Sort by time.
4. Find all women who ran the race faster than the race participant with Bib number 284. For each runner report first and last name, age group, overall place in the race, place within the age group and running time. Sort by the overall place.
5. List all runners who took first 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. Sort by age group, then by gender.
6. List all towns (with states) a runner in the 50-59 category from which showed a better time than at least one runner from the same town in the 20-39 age category for the same gender. Sort towns by state, then by town name. Report each town once.

AIRLINES dataset

1. Find all airlines that have at least one flight out of `ATO` 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 airlines that have at least one flight to `AXB` airport. Report the full name and the abbreviation of each airline. Report each name only once. Sort the airlines in alphabetical order.
3. Find all airlines that have at least one flight from `ATO` airport and at least one flight to `AXB` airport. Report the full name and the abbreviation of each airline. Report each name only once. Sort the airlines in alphabetical order.
4. Find all airlines that have at least one flight between `LTS` and `RLI` airports. Report the full name and the abbreviation of each airline. Report each name only once. Sort the airlines in alphabetical order.

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

6. Find all airports wih no direct flight from `ALX` airport. For each airport report its full name and the airport code. Sort output by 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.

WINE dataset

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

1. List all AVAs located in Sonoma County. Output just the names of the AVA appellations and sort them in alphabetical order.

2. List all `Central Coast` appellations that are not located in San Luis Obispo County. List the appellation name, county and whether it is an AVA. Sort output by county name, and by appellation name within the county.

3. List all AVAs located in the same county as the the `'Carneros'` AVA. List just AVA names and sort them in alphabetical order. (do not include `'Carneros'` itself).

4. List all red grape varieties for which at least one wine is rated in the database. Each grape variety needs to be reported once. Sort the output in alphabetical order.

5. List all appellations for which the database contains at least one rating for a 2006 `'Zinfandel'`. For each appellation list its name and county. Sort output in alphabetical order by county, then by appellation name. Report each appellation once.

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

7. List all `'Cabernet Sauvingnon'` wines with scores above 95 sorted in descending order by price. For each wine, list the winery, the name, vintage, score and price.

8. A case of wine is 12 bottles. For each Paso Robles Zinfandel from the 2006 vintage compute the total revenue assuming that all the wine sold at the specified price. Report the winery and name of the wine, wine score and overall revenue. Sort in descending order by revenue. Exclude NULL values.

9. List all 2006 vintage wines from Napa (any appellation within the county) whose total revenue exceeds that of the 2006 `'Appelation Series'`[6] Paso Robles Zinfandel from `'Rosenblum'` winery. For each wine report grape, winery and name, score and revenue. Order by revenue.

10. List all wines that have scores on two consecutive years where the scores for both vintages are the same. (for this to count, the winery, the grape, the appellation and the name of the wine must coincide). Report the grape, winery, the wine name and the two vintage years (earlier year first) Sort in alphabetical order grape, winery, then by the earlier vintage year.

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 `lab4.ext` where `ext` is one of the extensions above. The archive shall contain eight directories: `AIRLINES`, `CARS`, `CSU`, `INN BAKERY`, `STUDENTS`, `MARATHON` and `WINE`.

Each directory shall contain the following SQL scripts:

- Database creation script. (e.g., `CARS-setup.sql`). Use the scripts from Lab 2 and (for MARATHON) Lab 3 submissions.
- Table creation script. `cat` all `<DATASET>-build-<table>.sql` scripts together into one big script. Name it `<DATASET>-insert.sql` (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 (as specified above) `<DATASET>-info.sql` (e.g., `CARS-info.sql`.

Submit using handin:

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

[6] There is a typo there. Let it be for now.