HW4: Summary/Subquery

Start Assignment

Due Nov 5 by 11:59pm **Points** 100 **Submitting** a file upload
File Types sql, txt, docx, and pdf **Available** until Nov 6 at 11:59pm

Instructions:

- Download **this file** (https://utexas.instructure.com/courses/1318318/files/62832957/download?download_frd=1) first and run the script. You can click the hyperlink or find it under Canvas/Files/HW_Files. The file is named: **HW4-Starter-DDL-Script.sql** (https://utexas.instructure.com/courses/1318318/files/62832957/download?download_frd=1) but it is identical to the previous HW starter file.

Warning:

- **Do your own work:** This is a team assignment, but your team must do original work create own SQL statements. If you are caught cheating on this or using someone else's work, you will fail in this class and be reported to the Dean of Students. Also, this homework prepares you for the exam coming up so doing the work now will help you learn and do well when it counts more.

Summary Questions:

1. Write a SELECT statement that returns a single row with these columns:

- The count of the records in the *customer* table with a column alias of count_of_customers
- The minimum of the stay_credits_earned column in the *customer* table with column alias of min_credits
- The maximum of the stay_credits_earned column in the *customer* table with column alias of max_credits

2. We want to know the number of reservations and earliest first check_in_date for each customer to understand which customers are our biggest fans and oldest patrons.

- **Task:** Write a SELECT statement that returns one row for each customer that has a reservation

- **Task**: Write a SELECT statement that returns one row for each customer that has a reservation with the following columns and please use table aliases as well:
- The customer_id column from the *customer* table.
- The number of reservations on the *reservation* table with column alias of Number_of_Reservations
- The earliest check_in_date for that customer. This helps us know how long they've been doing business with us. Give this column the alias of earliest_check_in.

3. We're curious how popular we are based on where customers are from.

- **Task**: Write a SELECT statement that pulls info about customers and summarizes it by city and state. The following columns should be returned:
- The city column from the *customer* table
- The state column from the *customer* table
- The average stay_credits_earned for customers in that city, state. Please round the results to the nearest whole number and give the column an alias of *avg_credits_earned*.
- Sort the results by state ascending and then avg_credits_earned descending (i.e. largest first)

4. We'd like to know how many times each customer has stayed in a particular room at the South Austin location (i.e. Location_ID 1). Write a SELECT statement that properly joins *customer, reservation, reservation_details,* and *room* tables and then returns the following columns for only reservations at location 1:

  - The customer_id column
  - The last_name column
  - The room_number column
  - The count of reservations (i.e. reseravation_id) for each room with a column alias of *stay_count*
- Sort the result set by customer_id ascending and then by the stay_count alias in descending order.
- This query should highlight for each customer if they've had more than 1 stay in a room or not.

5. Make a copy of the previous query and let's make two updates to it. We decided we only want to look at completed reservations and not in-progress or future reservations so filter to only show status of C. Then filter out any results that have a stay_count of 2 or less. This should result in showing us who has already stayed in a particular room 3 or more times.

6. We want to know the anticipated number of guests by location and check_in_date.

Part A - Write a query that creates a report to show the sum of number_of_guests on *Reservation* broken out by the location_name and check_in_date for all reservations that have a future check_in_date (i.e. greater than today's date). The report should include the following

- The location_name column from the *location* table
- The check_in_date column from the *reservation* table
- The sum of the number_of_guests column in the *reservation* table
- Filter data to always show only records where the check_in_date is in the future.
- Use the ROLLUP operator to include a row that gives the subtotal location_name, check_in_date.

Part B – Explain in a commented sentence how the CUBE operator is different than ROLLUP and why it is useful.

7. See if you can figure out which features exist at all 3 locations by joining tables, aggregating data, and using having clause. Write a query that query lists all the feature names and the count of locations that have that specific feature. Show the feature_name and the count of location_ids but use an alias to rename the count to count_of_locations. Lastly, filter out any rows in the query to show only rows that have a count greater than 2 because we only want to see the features that are at 3 locations (i.e. features that are at all locations)

Subquery Questions:

8. Write a query that returns the same result set as this select statement, but don't use a left join this time. Instead, use a subquery in a WHERE clause that uses the NOT IN keyword. *Hint: Start by getting a list of all the customer_ids in the reservation table. This will be used as the subquery in a query that pulls all customers where the customer_id is not in that list.*

```
select distinct c.customer_id, c.first_name, c.last_name, c.email
from customer c left join reservation r on c.customer_id = r.customer_id
where reservation_id is null;
```

9. We want to know all the customers that have earned more than the average number of stay credits.

- Write a query statement that answers this question: Which customers have a stay_credits_earned balance that's greater than the average stay_credits_earned all customers?
- Return the first_name, last_name, email, phone, and stay_credits_earned for each customer.
- Sort the results by the stay_credits_earned from highest to lowest

10. Write a query that returns four columns: city, state, the sum of stay_credits_earned (with an alias of *total_earned*), and the sum of stay_credits_used (with an alias of *total_used* statement). To do this, you need to use GROUP BY. Sort the results by state, city. This should return a list of the cities by state and the total credits earned and used among all customers from that city.

- Then, write a second query that uses the first select statement in its FROM clause just to prove you know how to select from a query (i.e. an in-line join). The main query should return the following columns city, state, credits_remaining. To return credits_remaining, use an expression that subtracts *total_used* from *total_earned.*
- Order final results by order by credits_remaining desc.

11. Goal: We want to know who is going to be staying in a room that hasn't been used much. We define a room that hasn't been used much, as a room that has been reserved less than 5 times

- Write a query that returns the confirmation_nbr, date_created, check_in_date, status, and room_id of each reservation that exists for a room that has less than 5 reservation_detail records. We want you to utilize a subquery to do this. *Hint: Start by selecting all room_id and count of room_ids from reservation_details and grouping by room_id and filter down to only the room_ids that have a reservation_detail count less than 5. Once you have this query running you can remove the count(room_id) from the select so you only have a list of room_ids. Once you have your list of room_ids that are not used much, use this as a subquery in the WHERE portion of another query that pulls the columns from reservation and reservation_details.*
- Include a final filter on the outer query to ignore reservations that are completed (i.e. status of C)

12. Goal: We want to know which customers that are using Mastercard (i.e. MSTR) have only made a single reservation because Mastercard users will get a special $10 credit on their cards on their first stay at our hotel.

- Using an inline view, write a statement that will return one row per customer that has only 1 completed reservation and is using Mastercard. Each row should include the following columns we need to process the credit: cardholder_first_name, cardholder_last_name, card_number, expiration_date, and cc_id
- Using an inline view means you will have a subquery in the main query's FROM clause that acts as its own table that you will join *customer_payment* to.
- *HINT: Start by coding the subquery that pulls customer_id and the count of reservation_ids but only for reservations that have a status of C and only where the count of reservations is 1. Then join this subquery as an inline view to the customer_payment table to complete the query. After you have the subquery complete you can then incorporate it with the final outer query as an inline table join and then add a filter to the outer query that only shows cardholders using Mastercard (i.e. card_type of MSTR)*

Deliverables (What to turn in: two files)

- **Script in a (.sql) file format.** If you have issues doing this, at least save it as a .txt file. Do not submit in any other format other than .sql and .txt. The SQL questions above will be based on the DDL script that is posted on the Canvas instructions. Download that script and run it before you start.
  - Clearly separate your code for each question. Save your code into one SQL file with the naming format: Group_Number.SQL. Please make sure the team number you use matches what is in Canvas/People/Groups. For example: Team_1.SQL
  - Save your file either as a .sql file or as a .txt file. If you need help doing this, refer to the page linked in the Canvas assignment. Files saved in a different format will be 50% and files in a different format that cannot be read into SQL (example: PDF) will result in a 0%.
  - Submit your .sql file on Canvas before the deadline. Late submissions receive 50% off. No submissions will be accepted 24 hours after the deadline.
  - Do not include the DDL in your submission. If you do, you will lose 5 points. Only provide SQL with comments and nothing else. Do this going forward on all other assignments unless noted.
- **Write an executive summary (one page .pdf or .docx)** to explain the code, any assumptions your team made to deliver the SQL code to the customer. Save it as a .pdf or .docx file and submit along with the code.