HW3: Database Querying

Due Oct 26 by 11:59pm **Points** 100 **Submitting** a file upload
File Types sql, txt, docx, and pdf **Available** until Oct 27 at 11:59pm

This assignment was locked Oct 27 at 11:59pm.

Instructions:

Download **this file** **(https://utexas.instructure.com/courses/1318318/files/62543973/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: **HW3-Starter-DDL-Script.sql** **(https://utexas.instructure.com/courses/1318318/files/62543973/download?download_frd=1)**

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.

Questions:

1. Write a SELECT statement that returns the following columns from the *Customer_Payment* table: Cardholder first name, cardholder last name, the card type, and the card expiration date. Then, run this statement to make sure it works correctly. Add an ORDER BY clause to this statement that sorts the result set by expiration date in ascending order (i.e. oldest to newest). Then, run this statement again to make sure it works correctly. Note, this is a good way to iteratively build and test a statement, one clause at a time.
2. Write a SELECT statement that returns one column from the *Customer* table named customer_full_name that combines the first_name and last_name columns. Format this column with the first name, a space, and last name like this: **Michael Jordan.** Sort the result set by last name in descending sequence. Use the "IN" operator to return only the customers whose first name begins with letters of A, B, or C.
3. Write a SELECT statement that returns these columns from *Reservation*: customer_id, confirmation_nbr, date_created, check_in_date, and number_of_guests. Return only the rows for reservations that have a status of "upcoming" that have check_in_dates that are today or in the future but only for this year. That means to filter where only the check_in_date is greater or equal the current date (note: use SYSDATE here) and on or before Dec 31st. See if you can do this with

the current date (note: use SYSDATE here) and on or before Dec 31st. See if you can do this with only the following operators (<, >, <=, or >=).

4. **This is a two-part question:**

   Part A: Create a duplicate of the previous query but this time update the WHERE clause to use the BETWEEN operator. Keep the rest of the query the same.

   Part B: Using the MINUS operator, compare the query from #3 to the query from Part A in #4. If you get no rows returned, that means the queries produce the same results. Pretty cool huh!?! 😉

5. Write a SELECT statement that returns these column names and data from the *Reservation* table:

```
customer_id     The customer_id column
location_id     The location_id column
length_of_stay  This is calculated by subtracting check_in_date from the check_out_date. Assign a
n alias of length_of_stay
```

   Filter the query to only show completed reservations (i.e. status = 'C'). After you have that running correctly, update filter to use the ROWNUM pseudo column so the result set contains only the first 10 rows from the table.

   Sort the result set by the column alias **length_of_stay** in descending order and then also by customer_id ascending

6. Write a SELECT statement that returns the first_name, last_name, email from *Customer* and also a fourth column called *credits_available*. The credits available is calculated by subtracting credits used from credits earned. Once you have this, filter to only show customers with at least 10 or more credits available. Sort results by the column alias *credits_available*.
7. Write a SELECT statement that returns the first, middle, and last name of a customer's payment profile on *Customer_Payment*

   Using the NULL operator, return only rows for those customers with a middle name. Sort by column positions 2 and then 3 in ascending order

8. Using the DUAL table write a SELECT statement that uses SYSDATE function to create a row with these columns:

```
today_unformatted          The SYSDATE function unformatted
today_formatted              The SYSDATE function in this format: MM/DD/YYYY
```

This displays a number for the month, a number for the day, and a four-digit year. Use a FROM clause that specifies the Dual table. *Hint: You will need to implement the TO_CHAR function to format the sysdate in the format designated above.*

After you write this add the following columns to the row:

```
Credits_Earned        25
Stays_Earned          25 / 10
Redeemable_stays      (25/10) returned with FLOOR() function
Next_Stay_to_earn     (25/10) rounded to nearest whole number
```

Your result table contains only one row.

9. Write a SELECT statement that pulls *Reservation* records for all reservations that are completed (i.e. status of C) for location 2. Return only the following columns: Customer_id, Location_id, and a calculated column called length_of_stay which is just checkout date minus check-in date. Sort the results by length_of_stay descending and customer_id ascending. Lastly only pull in the top 20 rows. That means we want you to sort the table before filtering the 20 rows. *Hint:* Do this using FETCH command and not with a subquery since subqueries come later.
10. Pull all customers and their reservations. Filter data to only show completed reservations (i.e. status is C). Return just the following columns: first_name, last_name, confirmation_nbr, date_created, check_in_date, check_out_date. Sort results by customer_id ascending and check_out_date descending so that we see all customers and their most recent checkouts first.
11. Write a query that joins matching records between the following tables (Customer, Reservation, Reservation_Details, and Room) so we can understand what rooms customers are staying in. Only return rows for upcoming reservations (i.e. status of U) and for customers that have earned more than 40 credits. The query should display the following columns:

- Name – a concatenation of a customer's first and last name like so: e.g. Tayfun Keskin
- Location_ID
- Confirmation_Nbr
- Check_in_date
- Room_number

12. Write a query that returns any customers in our system's *customer* table that have never had a reservation. Show that you can use the proper type of join that will return all customers even when there's no matching reservation for them. Results should display the following columns for the customer: first_name, last_name, confirmation_nbr, date_created, check_in_date, check_out_date
13. Use the UNION operator to generate a result set consisting of five columns (four directly from the *Customer* table, and one calculated) as below:

```
Status_level              A calculated column that contains a value of '1-Gold Member', '2-Platin
um Member', or '3-Diamond Club'
First_name
Last_name
Email
Stay_Credits_earned
```

If the customer has less than 10 credits (i.e. they haven't earned a free stay yet), they are considered to be Gold Level and so their status_level column should contain a *literal string* value of ‘1-Gold Member’. If the customer has earned more than or equal to 10 credits but less than 40, their status_level column should contain a *literal* string value of ‘2-Platinum Member’. Otherwise, it the status_level is ‘3-Diamond Club’

Sort the final result set by the first and third columns in the results.

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.