Project Assignment for the course MySQL for Data Analytics 2023

Name: Ali Amaan

1. In the table orders at the car retailer (**classicmodels**) database, what is the customerNumber of the customer who has the highest frequency of placing orders to the company in 2004 [orderDate in 2004]? You don't need to consider whether the products have been finally shipped or not.

   The customerNumber of the customer: **141** (6 points)

   The highest frequency is: **9** (6 points)

   The MySQL code that generates the result:

```sql
SELECT customerNumber,
    COUNT(*) as orderFrequency
-- Counting the number of orders for each customer
FROM orders
WHERE YEAR(orderDate) = 2004
GROUP BY customerNumber
-- Grouping the results by customer number
ORDER BY orderFrequency DESC
LIMIT 1;  -- Limiting the result to the customer with the highest order frequency
```

2. In the **classicmodels** database, customer names and customer numbers can be found [see table "customers"]. Customers make different payments on different dates [See table "payments"]. Please specify the names of two customers who are most **often** (count frequency) to make payments during the weekend [12 points].
   Name of Customer 1: **Mini Gifts Distributors Ltd.**
   Name of Customer 2: **Marseille Mini Autos**
   Note: i) Please provide "customerName", not the contact names from table "customers"

   The MySQL code that generates the result:

```
SELECT c.customerName, COUNT(p.paymentDate) AS weekendPayments
FROM customers c
    JOIN payments p ON c.customerNumber = p.customerNumber
WHERE WEEKDAY(p.paymentDate) >= 5
GROUP BY c.customerName
ORDER BY weekendPayments DESC
LIMIT 2;
```

3. In the classicmodels database, one sales representative is responsible for one or several customers [see “salesRepEmployeeNumber” in table “customers”]. Customers make different payments on different dates [See table “payments”]. In other words, we can say sales representatives help the company get customers to make payments. Now the question is, who is the sales representative that brings the most **revenue** [or **the total amount of the payments from customers in the table payments**] to the company?
   Note: i) Ignoring those customers who are not assigned to any sales representative – they have not made any purchases yet.

   The salesRepEmployeeNumber of the sales representative who brings the most **revenue** is: **Gerard Hernandez** [12 points].

   The MySQL code that generates the result:

```
SELECT c.salesRepEmployeeNumber, e.lastName, e.firstName,
    SUM(p.amount) as totalRevenue
-- Calculate the total revenue for each sales rep-resentative
FROM customers c
    -- based on the relationship between salesRepEmployeeNumber and employ-
eeNumber
    JOIN employees e ON c.salesRepEmployeeNumber = e.employeeNumber
    JOIN payments p ON c.customerNumber = p.customerNumber
-- Join based on custom-er number
GROUP BY c.salesRepEmployeeNumber
ORDER BY totalRevenue DESC
-- Sort in descending order of total revenue
LIMIT 1;
```

4. In the **cfpb_complaints_2500** database, you can find "closed with relief" in **Company_response** column. Please find the **name of the company** that has the highest **ratio** of cases that is '**closed with relief**'? Only those companies with more than 30 cases [all different kinds of cases, no matter whether they are featured with 'closed with relief' or not] in the database are considered (10 points).
   - **ratio** [or percentage] for an individual company is calculated as:
     The amount of its cases featured with 'closed with relief' / the amount of its total cases

The name of the company: **Barclays** (5 points)

The ratio of the cases for the company: **42.8571** (5 points)

The MySQL code that generates the result:

```
SELECT
    Company, -- Counting total number of cases
    COUNT(*) AS Total_Cases, -- Summing up cases closed with relief
    SUM(CASE WHEN Company_response = 'Closed with relief' THEN 1 ELSE 0 END) AS
Cases_with_Relief,
    -- Calculating the ratio of cases closed with relief to total cases
    (SUM(CASE WHEN Company_response = 'Closed with relief' THEN 1 ELSE 0 END) /
COUNT(*)) * 100 AS Relief_Ratio
FROM cfpb_complaints_2500
GROUP BY Company
-- Filtering out companies with less than 30 cases
HAVING Total_Cases > 30
-- Ordering the results by Relief Ratio in descending order
ORDER BY Relief_Ratio DESC
LIMIT 1;
```

5. In the **cfpb_complaints_2500** database, many complaints are related to 'loan' (those cases where the word 'loan' is included in the column 'Issue'). Please specify **the name of the company** that has the most issues related to '**loan'** on **Wednesday (DATA_received)**. Only complaints with the column '**State'** starting with character "**A**" are considered (8 points).

The name of the company: **Bank of America** (8 points)

The MySQL code that generates the result:

```
SELECT
    Company,
    COUNT(*) AS Total_Loan_Issues
FROM cfpb_complaints_2500
WHERE
-- Filter for complaints received on a Wednesday (4 corresponds to Wednesday)
  DAYOFWEEK(Data_received) = 4
    -- Filter for states starting with 'A'
    AND State LIKE 'A%'
    -- Filter for issues containing the word 'loan'
    AND Issue LIKE '%loan%'
GROUP BY Company
ORDER BY Total_Loan_Issues DESC
-- Limiting the result to the company with the highest number of loan-related
issues
LIMIT 1;
```

6. In the Chile database, let's assume that an income less than 10,000 is a low income; an income between 10,000 and 100,000 is a middle income; an income higher than 100,000 is a high income. We would like to know whether the income level and the statusquo have a certain relationship for females who voted yes to Pinochet. To answer this question, you need to provide the average statusquo value for the females who voted yes to Pinochet in correspondence to their different income levels. (9 points)

   [Please carefully read the question so that you will not miss any important condition when answering the question; Please provide **three** digits after the decimal point in the results]

Suggestion: you may need to update the table by adding a new column of income_level to answer the question.

Income level	Mean statusquo
High_income	**1.077**
Middle_income	**0.927**
Low_income	**0.937**

The MySQL code that generates the result:

```
-- Add a new column for income level
ALTER TABLE chile
ADD COLUMN income_level VARCHAR(15);

-- Update income level based on income value
UPDATE chile
SET income_level = CASE
    WHEN income > 100000 THEN 'High_income'
    WHEN income BETWEEN 10000 AND 100000 THEN 'Middle_income'
    WHEN income < 10000 THEN 'Low_income'
    ELSE NULL
END;

-- Calculate the average statusquo for each income level with 3 digits after the
decimal point
SELECT income_level, ROUND(AVG(statusquo), 3) as Mean_statusquo
FROM chile
WHERE vote = 'Y' AND sex = 'F' AND income_level IS NOT NULL
GROUP BY income_level;
```

7. Based on the use of cfpb consumer complaint database (2500 rows), please count the frequency of the complaints that satisfy the following three conditions at the same time: i) consumer finally disputed with the company (*Consumer_disputed*); and ii) were received on Friday (*Data_received*) and iii) the difference between *Data_received* and *Data_sent_to_company* is more than 5 days. (10 points)

   Please specify the name of the company that has the biggest amount of the above-mentioned complaints.

   The name of the company: **Euro+ Shopping Channel** (5 points)

   Frequency of the mentioned complaints of the company: **26** (5 points)

   The MySQL code that generates the result:

```sql
SELECT
    c.customerName AS companyName,
    COUNT(*) AS complaintFrequency
FROM
    orders o
    -- Joining the orders table  with the customers table
    -- based on the relationship between customerNumber in orders and customers
    JOIN
    customers c ON o.customerNumber = c.customerNumber
    JOIN
    payments p ON o.customerNumber = p.customerNumber

-- Filtering the results to include only shipped orders on a Friday,
-- where the shipped date is more than 5 days after the order date
WHERE
    o.status = 'Shipped'
    AND DAYOFWEEK(o.orderDate) = 6 -- Friday
    AND DATEDIFF(o.shippedDate, o.orderDate) > 5
    AND p.paymentDate IS NOT NULL
GROUP BY
    c.customerName
ORDER BY
    complaintFrequency DESC
LIMIT 1;
```

8. In the data “tripadvisor_review_sample_without_reviewtext”, based on the review titles that have **at least two words** [a total of 5 points]:
   1) Regarding the **first** word used in the review title, what is the most popular word [case-insensitive]?

   The word is: **Great** [1 points] The frequency of the word is: **6435** [1 points]

   2) Regarding the **second** word used in the review title, what is the most popular word [case-insensitive]?

   The word is: **Hotel** [1 points] The frequency of the word is: **4257** [1 points]

   3) For those titles that **start with** the words “bad” OR “terrible”, what are the most popular **second** word [case-insensitive]:
   The word is: **Experience** [1 points]

   Note: Before answering question 6, please clean the title based on the following two requirements.
   i) Please remember to first **remove** six types of punctuation marks from the title, including:

“	”	"	-	,	!

In order to obtain consistent results, please do not remove more punctuation marks than we specified above and also do not replace the punctuation with an empty space.
   ii) Please remove empty spaces from both sides of the title.

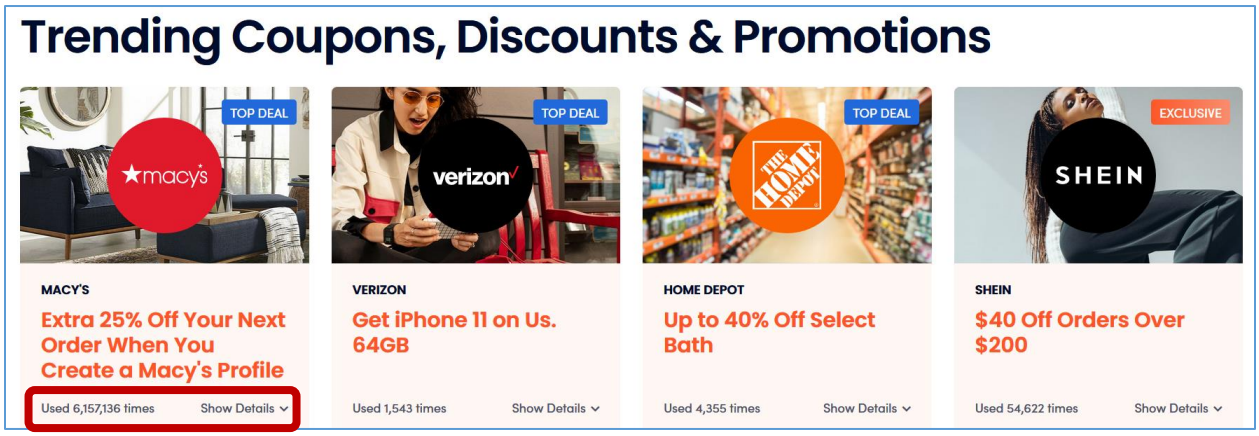
The MySQL code that generates the result:

```
-- cleaning the titles by removing the 6 type of punctuation marks
-- clean up and standardize title format
UPDATE tripadvisor_review_sample_without_reviewtext
SET title = TRIM(BOTH ' “”"-,! ' FROM REGEXP_REPLACE(title, '[“”"\\-\\,\\!]',
''));

-- Find the first most popular word used in the reviews
SELECT SUBSTRING_INDEX(title, ' ', 1) AS first_word, COUNT(*) AS frequency
FROM tripadvisor_review_sample_without_reviewtext
WHERE LENGTH(title) - LENGTH(REPLACE(title, ' ', '')) >= 1
GROUP BY first_word
ORDER BY frequency DESC
LIMIT 1;

-- Find the second most popular word used in the reviews
SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 2), ' ', -1) AS second_word,
COUNT(*) AS frequency
FROM tripadvisor_review_sample_without_reviewtext
```

```sql
WHERE LENGTH(title) - LENGTH(REPLACE(title, ' ', '')) >= 1
GROUP BY second_word
ORDER BY frequency DESC
LIMIT 1;

-- Find the most popular word after 'terrible' or 'bad' in the title
SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(title, ' ', 2), ' ', -1) AS second_word,
COUNT(*) AS frequency
FROM tripadvisor_review_sample_without_reviewtext
WHERE (LOWER(SUBSTRING_INDEX(title, ' ', 1)) = 'bad'
    OR LOWER(SUBSTRING_INDEX(title, ' ', 1)) = 'terrible')
    AND LENGTH(title) - LENGTH(REPLACE(title, ' ', '')) >= 1
GROUP BY second_word
ORDER BY frequency DESC
LIMIT 1;
```

9. The following screenshot shows the sale volume of a coupon site (https://www.savings.com/). Assuming that you have collected the sale volume information of each coupon on this website **on a daily basis**. For instance, the **total sale volume** for the Coupon "Macy's" is 6,157,136 at the data collection date.

You want to calculate the daily sale volume for each coupon of this website (maybe because this website is the competitor of your company) for further analysis. How can you do that task using MySQL? **(5 points)**



Based on the data of the **total sale volume** for each coupon (like **Table A)**. Please calculate **the daily sale volume**, as shown in **Table B**.

Explanation: For instance, the **daily** sale volume for CouponA on '2021-09-05' is 25, because the **total** sale volume for CouponA is 37 on '2021-09-**05'** and 12 on '2021-09-**04'.** The difference in total sale volumes between the two dates is the daily sale volume, which is 37 – 12 = 25.

Coupon_ID	Sale_date	Total_sale_volume
CouponA	2021-09-03	0
CouponA	2021-09-04	12
CouponA	2021-09-05	37
CouponA	2021-09-06	40
CouponB	2021-08-14	43
CouponB	2021-08-15	75
CouponB	2021-08-13	20
CouponC	2021-09-04	16
CouponC	2021-09-05	38
CouponC	2021-09-06	77
CouponC	2021-09-07	97
CouponC	2021-09-03	2

Row Data: Table A

Coupon_ID	Sale_date	Total_sale_volume	daily_sale_volume
CouponA	2021-09-03	0	0
CouponA	2021-09-04	12	12
CouponA	2021-09-05	37	25
CouponA	2021-09-06	40	3
CouponB	2021-08-13	20	20
CouponB	2021-08-14	43	23
CouponB	2021-08-15	75	32
CouponC	2021-09-03	2	2
CouponC	2021-09-04	16	14
CouponC	2021-09-05	38	22
CouponC	2021-09-06	77	39
CouponC	2021-09-07	97	20

Result: Table B

Requirement:

1. Please write code to generate the results as shown in Table B – the code will be the answer.
2. You can download Table A from Mycourse (coupon_sale_volume.sql)

The MySQL code that generates the result **(5 points)***:*

```
-- Calculate and insert the daily sale volume
INSERT INTO `daily_sale_volume` (`Coupon_ID`, `Sale_date`, `Total_sale_volume`,
`Daily_sale_volume`)
SELECT
    a.`Coupon_ID`,
    a.`Sale_date`,
    a.`Total_sale_volume`,
    COALESCE(a.`Total_sale_volume`, 0) - COALESCE(b.`Total_sale_volume`, 0) AS
`Daily_sale_volume`
FROM
    `coupon_sale_volume` a
LEFT JOIN
    `coupon_sale_volume` b
ON
    a.`Coupon_ID` = b.`Coupon_ID`
    AND a.`Sale_date` = DATE_ADD(b.`Sale_date`, INTERVAL 1 DAY);

-- Display the results
SELECT * FROM `daily_sale_volume`;
```

10. Assuming you are now a business analyst offering consultant service to the tourism minister of Finland **(a total of 5 points)**. The minister wants to know how tourists travel within Finland between different cities.

- Specifically, **<u>for those tourists whose first visit</u>** to Finland is Helsinki city [i.e., the first review in the database (in terms of review_date) is about a hotel in **Helsinki_Uusimaa**], which city would most likely be visited by those tourists in the future?

  i) **<u>For those tourists whose first visit</u>** to Finland is Helsinki city [i.e., the first review in the database (in terms of review_date) is about a hotel in **Helsinki_Uusimaa]**, they also visited "**Rovaniemi_Lapland"** for **__2.0769__ times <u>in the future</u>** (2 points).

  ii) **<u>For those tourists whose first visit</u>** to Finland is to Helsinki city [i.e. first review in the database (in terms of review_date) is about a hotel in **Helsinki_Uusimaa]**, **<u>235</u>** of **them** also visited **both** "**Saariselka_Lapland"** and "**Rovaniemi_Lapland**" cities **<u>in the future</u>** (3 points).

Note:

- Please read the assignment questions carefully!
- Please download the "assignment_tourist_Finland.sql" dataset from MyCourse.
  - It would be good to remove previous review-related tables before you import the database file so that you won't mix the current assignment tables with previous review-related tables.
  - The sql file contains two tables of "hotel" [431 records] and "review2" [56,709 records]. Column 'Id' of the table 'hotel' is connected to the column 'hotel_id' of the table 'review2'
- It may happen that a traveler wrote multiple reviews about hotels in different cities as his/her **first** reviews (on the same but earliest review date). If one of these 'first' reviews includes Helsinki city, the traveler should be counted as a tourist whose first visit to Finland is Helsinki city. If one of these 'first' reviews includes 'Rovaniemi_Lapland' city, that trip is **NOT** considered visiting 'Rovaniemi_Lapland' city. "**<u>In the future</u>**" means future trips after these first reviews that were written on the same but earliest review day - *I know this does not sound so logical, but it is good to increase the difficulty of the assignment question for training your mind).*
- Assume that different values in the column "city" of the "hotel" table represent different cities. For instance, "**Saariselka**" and "**Rovaniemi**" are two different cities.

The MySQL code that generates the results **(2+3 points)**:

```sql
-- task1: Calculate the average number of visits to Rovaniemi for users who first visited Helsinki_Uusimaa
SELECT AVG(visits_to_rovaniemi) AS average_visits_to_rovaniemi
FROM (
    -- Subquery: Count visits to Rovaniemi for users who first visited Helsin-ki_Uusimaa
    SELECT user_id, COUNT(*) AS visits_to_rovaniemi
    FROM review2 r1
    WHERE r1.review_date = (
            -- Find the first visit date to Helsinki_Uusimaa for each user
            SELECT MIN(review_date)
            FROM review2 r2
            WHERE r2.user_id = r1.user_id AND r2.hotel_id IN (
                -- Select hotels in Helsinki_Uusimaa
                SELECT id
                FROM hotel
                WHERE city = 'Helsinki_Uusimaa'
            )
        )
        AND r1.hotel_id IN (
            -- Select hotels in Rovaniemi
            SELECT id
            FROM hotel
            WHERE city = 'Rovaniemi_Lapland'
        )
    GROUP BY user_id
)
AS subquery;

-- task1: Calculate the average number of visits to Rovaniemi for users who first visited Helsin-ki_Uusimaa
SELECT AVG(visits_to_rovaniemi) AS average_visits_to_rovaniemi
FROM (
    -- Subquery: Count visits to Rovaniemi for users who first visited Helsin-ki_Uusimaa
    SELECT user_id, COUNT(*) AS visits_to_rovaniemi
    FROM review2 r1
    WHERE r1.review_date = (
            -- Find the first visit date to Helsinki_Uusimaa for each user
            SELECT MIN(review_date)
            FROM review2 r2
            WHERE r2.user_id = r1.user_id AND r2.hotel_id IN (
                -- Select hotels in Helsinki_Uusimaa
                SELECT id
                FROM hotel
```

```
                WHERE city = 'Helsinki_Uusimaa'
            )
        )
        AND r1.hotel_id IN (
            -- Select hotels in Rovaniemi
            SELECT id
            FROM hotel
            WHERE city = 'Rovaniemi_Lapland'
        )
    GROUP BY user_id
)
AS subquery;

-- task2: Identify users who first visited Helsinki_Uusimaa and count future visits to Saariselka_Lapland and Rovaniemi_Lapland
-- step 1: Identify users who first visited Helsinki_Uusimaa
SELECT DISTINCT r1.user_id
FROM review2 r1 INNER JOIN hotel h1 ON r1.hotel_id = h1.id
WHERE r1.review_date = (
    -- Find the first visit date to Helsinki_Uusimaa for each user
    SELECT MIN(review_date)
    FROM review2 r2 INNER JOIN hotel h2 ON r2.hotel_id = h2.id
    WHERE r2.user_id = r1.user_id AND h2.city = 'Helsinki_Uusimaa'
);

-- Step 2: Count tourists who will visit both "Saariselka_Lapland" and "Rovaniemi_Lapland" in the future
WITH FutureVisits AS (
    SELECT h1.user_id, h2.city AS future_visit_city
    FROM (
        SELECT DISTINCT r1.user_id
        FROM review2 r1 INNER JOIN hotel h1 ON r1.hotel_id = h1.id
        WHERE r1.review_date = (
            -- Find the first visit date to Helsinki_Uusimaa for each user
            SELECT MIN(review_date)
            FROM review2 r2 INNER JOIN hotel h2 ON r2.hotel_id = h2.id
            WHERE r2.user_id = r1.user_id AND h2.city = 'Helsinki_Uusimaa'
        )
    )
    h1 JOIN review2 r ON h1.user_id = r.user_id
        INNER JOIN hotel h2 ON r.hotel_id = h2.id
)

-- Main query for Step 2: Count and calculate the percentage of tourists visiting both cities
```

```sql
SELECT
    COUNT(DISTINCT h1.user_id) AS helsinki_tourists,
    COUNT(DISTINCT CASE WHEN fv.future_visit_city = 'Saariselka_Lapland' THEN
h1.user_id END) AS saariselka_visits,
    COUNT(DISTINCT CASE WHEN fv.future_visit_city = 'Rovaniemi_Lapland' THEN
h1.user_id END) AS rovaniemi_visits,
    COUNT(DISTINCT CASE WHEN fv.future_visit_city = 'Saariselka_Lapland' AND
fv.user_id IS NOT NULL THEN h1.user_id END) AS both_cities_visits,
    (COUNT(DISTINCT CASE WHEN fv.future_visit_city = 'Saariselka_Lapland' AND
fv.user_id IS NOT NULL THEN h1.user_id END) * 100.0) / COUNT(DISTINCT h1.user_id)
AS percentage
FROM (
    SELECT DISTINCT r1.user_id
    FROM review2 r1 INNER JOIN hotel h1 ON r1.hotel_id = h1.id
    WHERE r1.review_date = (
        -- Find the first visit date to Helsinki_Uusimaa for each user
        SELECT MIN(review_date)
        FROM review2 r2 INNER JOIN hotel h2 ON r2.hotel_id = h2.id
        WHERE r2.user_id = r1.user_id AND h2.city = 'Helsinki_Uusimaa'
        )
    ) h1 LEFT JOIN FutureVisits fv ON h1.user_id = fv.user_id;
```