Data Analyst and Health Data Fellow Screening Exercise

By Jessey Sitima

Exercise 1: An SQL query that will display the duplicate records.

Query:

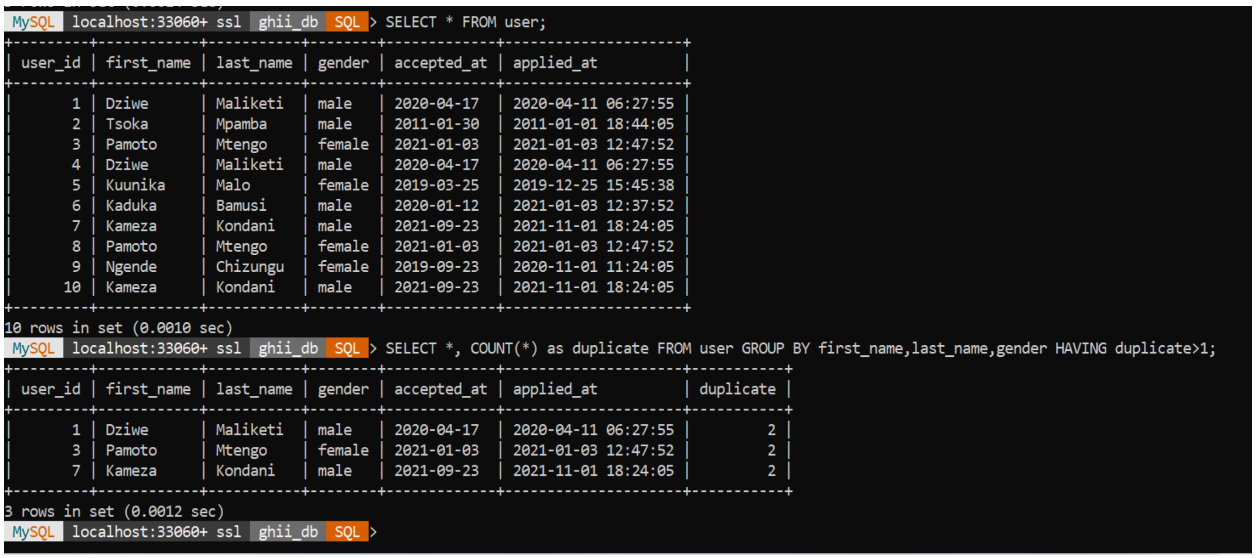
SELECT *, COUNT(*) as duplicate FROM user GROUP BY first_name, last_name, gender HAVING duplicate>1;

Explanation:

This SQL query selects all rows from the "user" table where there are duplicates in the combination of "first_name," "last_name," and "gender." It counts these duplicates and only returns rows where the count is greater than 1.

Solution:

The screenshot below shows the table before and after running the query;



```
 MySQL  localhost:33060+ ssl  ghii_db  SQL > SELECT * FROM user;
+---------+------------+-----------+--------+-------------+---------------------+
| user_id | first_name | last_name | gender | accepted_at | applied_at          |
+---------+------------+-----------+--------+-------------+---------------------+
|       1 | Dziwe      | Maliketi  | male   | 2020-04-17  | 2020-04-11 06:27:55 |
|       2 | Tsoka      | Mpamba    | male   | 2011-01-30  | 2011-01-01 18:44:05 |
|       3 | Pamoto     | Mtengo    | female | 2021-01-03  | 2021-01-03 12:47:52 |
|       4 | Dziwe      | Maliketi  | male   | 2020-04-17  | 2020-04-11 06:27:55 |
|       5 | Kuunika    | Malo      | female | 2019-03-25  | 2019-12-25 15:45:38 |
|       6 | Kaduka     | Bamusi    | male   | 2020-01-12  | 2021-01-03 12:37:52 |
|       7 | Kameza     | Kondani   | male   | 2021-09-23  | 2021-11-01 18:24:05 |
|       8 | Pamoto     | Mtengo    | female | 2021-01-03  | 2021-01-03 12:47:52 |
|       9 | Ngende     | Chizungu  | female | 2019-09-23  | 2020-11-01 11:24:05 |
|      10 | Kameza     | Kondani   | male   | 2021-09-23  | 2021-11-01 18:24:05 |
+---------+------------+-----------+--------+-------------+---------------------+
10 rows in set (0.0010 sec)
 MySQL  localhost:33060+ ssl  ghii_db  SQL > SELECT *, COUNT(*) as duplicate FROM user GROUP BY first_name,last_name,gender HAVING duplicate>1;
+---------+------------+-----------+--------+-------------+---------------------+-----------+
| user_id | first_name | last_name | gender | accepted_at | applied_at          | duplicate |
+---------+------------+-----------+--------+-------------+---------------------+-----------+
|       1 | Dziwe      | Maliketi  | male   | 2020-04-17  | 2020-04-11 06:27:55 |         2 |
|       3 | Pamoto     | Mtengo    | female | 2021-01-03  | 2021-01-03 12:47:52 |         2 |
|       7 | Kameza     | Kondani   | male   | 2021-09-23  | 2021-11-01 18:24:05 |         2 |
+---------+------------+-----------+--------+-------------+---------------------+-----------+
3 rows in set (0.0012 sec)
 MySQL  localhost:33060+ ssl  ghii_db  SQL >
```

Exercise 2:

(a) Two ways in which the query can be restructured (optimized) to run efficiently

1. **Use of Common Table Expression (CTE) for the Subquery:**

   Instead of using a subquery within the main query's FROM clause, we can rewrite the subquery as a Common Table Expression (CTE). CTEs are often easier to read and maintain, and they sometimes result in better query optimization. Using a CTE can make the query more readable and potentially improve its performance.

2. **Indexing:**

   Ensure that the relevant columns used for joining and filtering the data are properly indexed. In this case, we can add indexes to the following columns:

   - Index the "person_id" column in the "Patient_Visit" table since it's used for grouping and joining.
   - Index the "person_id" column in the "Diagnosis_Stage" table.
   - Proper indexing can significantly improve the query's performance by reducing the time it takes to access and retrieve the required data.

(b) Query for the solution above:

Making the Subquery as Common Table Expression CTE:

```
WITH MaxVisitDates AS (
            SELECT person_id, MAX(visit_date) AS visit_date
            FROM Patient_Visit
            GROUP BY person_id )
    SELECT pd.person_id, pd.first_name, pd.last_name, mvd.visit_date
    FROM Patient_Demographic pd
    INNER JOIN MaxVisitDates mvd ON pd.person_id = mvd.person_id
    WHERE pd.person_id IN (SELECT DISTINCT person_id FROM
    Diagnosis_Stage);
```

Exercise 3: Data Duplication

The below python script uses the panda library for reading data. The python script and the csv supporting docs can be found at https://github.com/JesseySitima/GHII-Data-analyst-Exercise.git

```python
import pandas as pd

# task a: Read the CSV file
input_file = "client_purchases.csv"
df = pd.read_csv(input_file)

# task b: Find and identify duplicate records
duplicates = df[df.duplicated(subset=['First Name', 'Last Name', 'Email',
'Phone', 'Address', 'Zip Code', 'Gender', 'Date of Birth', 'Product',
'Quantity', 'Cost'])]

# task c: Remove the duplicate records
df = df.drop_duplicates(subset=['First Name', 'Last Name', 'Email',
'Phone', 'Address', 'Zip Code', 'Gender', 'Date of Birth', 'Product',
'Quantity', 'Cost'], keep='first')

# task d: Export the cleaned CSV file
cleaned_output_file = "client_purchases_deduplicated.csv"
df.to_csv(cleaned_output_file, index=False)

# task e: Identify unique clients and assign unique IDs
unique_clients = df[['First Name', 'Last Name', 'Email', 'Phone',
'Address', 'Zip Code', 'Gender', 'Date of
Birth']].drop_duplicates().reset_index(drop=True).reset_index()
unique_clients.rename(columns={'index': 'Client ID', 'First Name': 'First
Name', 'Last Name': 'Last Name', 'Email': 'Email', 'Phone': 'Phone',
'Address': 'Address', 'Zip Code': 'Zip Code', 'Gender': 'Gender', 'Date of
Birth': 'Date of Birth'}, inplace=True)

# Export the unique clients to a CSV file
unique_clients_output_file = "clients_unique.csv"
unique_clients.to_csv(unique_clients_output_file, index=False)
```

Exercise 4: Data Privacy and Security

a) From the given deduplicated CVS, variables that qualify as Personal Identifiable Information are:

- First Name
- Last Name
- Email
- Phone
- Address
- Zip Code
- Gender
- Date of Birth

b) Below is a python script using the Faker library which generates realistic but fake data.

```python
import pandas as pd
from faker import Faker

# Initialize the Faker generator
fake = Faker()

# Read the CSV file with PII
input_file = "client_purchases_deduplicated.csv"
df = pd.read_csv(input_file)

# Anonymize sensitive data
df['First Name'] = [fake.first_name() for _ in range(len(df))]
df['Last Name'] = [fake.last_name() for _ in range(len(df))]
df['Email'] = [fake.email() for _ in range(len(df))]
df['Phone'] = [fake.phone_number() for _ in range(len(df))]
df['Address'] = [fake.street_address() for _ in range(len(df))]
df['Zip Code'] = [fake.zipcode() for _ in range(len(df))]
df['Gender'] = [fake.random_element(elements=('Male', 'Female')) for
_ in range(len(df))]
df['Date of Birth'] = [fake.date_of_birth(minimum_age=18,
maximum_age=90).strftime('%m/%d/%Y') for _ in range(len(df))]

# Saving the anonymized data to a new CSV file
anonymized_output_file = "clients_deidentified.csv"
df.to_csv(anonymized_output_file, index=False)
```