PlantinumRx Assignment Document

1.Objective

The goal of this project is to assess and demonstrate core data analysis skills. You will be building a small portfolio of solutions that covers:

- **Database Management (SQL):** Creating schemas and querying data for Hotel and Clinic management systems.
- **Data Manipulation (Spreadsheets):** Using functions to look up data and perform time-based analysis.
- **Programming Logic (Python):** Writing scripts for data conversion and string manipulation.

Phase 1: SQL Proficiency

This section involves two scenarios: a Hotel Management System and a Clinic Management System.

Database creation:

-> create database hotelManagementSystem;

-> use hotelManagementSystem;

Database changed

Users table creation:

create table users(user_id varchar(50) primary key,name varchar(100),phone_number int(10),mail_id varchar(100),billing_address varchar(50));

Values insertion:

-> insert into users values('21wrcxuy-67erfn','John Doe',975412365,'john.doe@example.com','UK'),

(‘21erthdy-69erfn’,’Sadie’,’987654321,’sadie123@example.com’,’USA’);

Bookings table creation:

```
-> CREATE TABLE bookings (
    ->     booking_id VARCHAR(50) PRIMARY KEY,
    ->     booking_date DATETIME NOT NULL,
    ->     room_no VARCHAR(50) NOT NULL,
    ->     user_id VARCHAR(50) NOT NULL
    -> );
```

Values insertion:

```
INSERT INTO bookings (booking_id, booking_date, room_no, user_id)
VALUES
('bk-09f3e-95hj', '2021-09-23 07:36:48', 'rm-bhf9-aerjn', '21wrcxuy-67erfn'),
('bk-06f3e-96hj', '2021-11-23 07:12:48', 'rm-ghj9-aerjn', '21erthdy-69erfn');
```

Items table creation:

```
-> CREATE TABLE items (
    ->     item_id VARCHAR(50) PRIMARY KEY,
    ->     item_name VARCHAR(100) NOT NULL,
    ->     item_rate DECIMAL(10,2) NOT NULL
    -> );
```

Values insertion

```
-> INSERT INTO items (item_id, item_name, item_rate)
    -> VALUES
```

-> ('itm-a9e8-q8fu', 'Tawa Paratha', 18),

-> ('itm-a07vh-aer8', 'Mix Veg', 89);

Booking_commercials table creation:

->CREATE TABLE booking_commercials (

id VARCHAR(50) PRIMARY KEY,

booking_id VARCHAR(50) NOT NULL,

bill_id VARCHAR(50) NOT NULL,

bill_date DATETIME NOT NULL,

item_id VARCHAR(50) NOT NULL,

item_quantity DECIMAL(10,2) NOT NULL);

Values insertion:

INSERT INTO booking_commercials

(id, booking_id, bill_id, bill_date, item_id, item_quantity)

VALUES

('q34r-3q4o8-q34u', 'bk-09f3e-95hj', 'bl-0a87y-q340', '2021-09-23 12:03:22', 'itm-a9e8-q8fu', 3),

('q3o4-ahf32-o2u4', 'bk-09f3e-95hj', 'bl-0a87y-q340', '2021-09-23 12:03:22', 'itm-a07vh-aer8', 1),

('134lr-oyfo8-3qk4', 'bk-q034-q4o', 'bl-34qhd-r7h8', '2021-09-23 12:05:37', 'itm-w978-23u4', 0.5),

('34qj-k3q4h-q34k','bk-06f3e-96hj', 'bl-54dft-r8u8', '2021-11-23 12:05:37', 'itm-a9e8-q8fu', 3);

A. 1. For every user in the system, get the user_id and last booked room_no

```
 SELECT b.user_id, b.room_no

FROM bookings b

JOIN (

    SELECT user_id, MAX(booking_date) AS last_booking

    FROM bookings

    GROUP BY user_id

) AS last_bookings

ON b.user_id = last_bookings.user_id

   AND b.booking_date = last_bookings.last_booking;
```

output:

```
+----------------+--------------+

| user_id        | room_no      |

+----------------+--------------+

| 21erthdy-69erfn | rm-ghj9-aerjn |

| 21wrcxuy-67erfn | rm-bhf9-aerjn |

+----------------+--------------+

2 rows in set (0.01 sec)
```

2. Get booking_id and total billing amount of every booking created in November, 2021

```
SELECT
    b.booking_id,
    SUM(bc.item_quantity * i.item_rate) AS total_amount
```

```
FROM bookings b
JOIN booking_commercials bc ON b.booking_id = bc.booking_id
JOIN items i ON bc.item_id = i.item_id
WHERE b.booking_date >= '2021-11-01 00:00:00'
  AND b.booking_date < '2021-12-01 00:00:00'
GROUP BY b.booking_id;
```

output:

```
+---------------+--------------+
| booking_id    | total_amount |
+---------------+--------------+
| bk-06f3e-96hj |      54.0000 |
+---------------+--------------+
```

3. Get bill_id and bill amount of all the bills raised in October, 2021 having bill amount

```
SELECT
    bc.bill_id,
    SUM(bc.item_quantity * i.item_rate) AS bill_amount
FROM booking_commercials bc
JOIN items i ON bc.item_id = i.item_id
WHERE bc.bill_date >= '2021-10-01 00:00:00'
  AND bc.bill_date < '2021-11-01 00:00:00'
GROUP BY bc.bill_id
HAVING SUM(bc.item_quantity * i.item_rate) > 1000;
```

Output:

Empty set (0.04 sec)

4. Determine the most ordered and least ordered item of each month of year 2021

```
SELECT
    YEAR(bc.bill_date) AS yr,
    MONTH(bc.bill_date) AS mnth,
    i.item_name,
    SUM(bc.item_quantity) AS total_quantity,
    CASE
        WHEN SUM(bc.item_quantity) = MAX(SUM(bc.item_quantity)) OVER (PARTITION
BY YEAR(bc.bill_date), MONTH(bc.bill_date))
        THEN 'Most Ordered'
        ELSE 'Least Ordered'
```

```
    END AS order_type
FROM booking_commercials bc
JOIN items i ON bc.item_id = i.item_id
WHERE YEAR(bc.bill_date) = 2021
GROUP BY yr, mnth, i.item_name;
```

Output:

```
+------+------+-------------+---------------+--------------+
| yr   | mnth | item_name    | total_quantity | order_type    |
+------+------+-------------+---------------+--------------+
| 2021 |    9 | Tawa Paratha |          3.00 | Most Ordered  |
| 2021 |    9 | Mix Veg      |          1.00 | Least Ordered |
| 2021 |   11 | Tawa Paratha |          3.00 | Most Ordered  |
+------+------+-------------+---------------+--------------+
3 rows in set (0.04 sec)
```

5. Find the customers with the second highest bill value of each month of year 2021

```
WITH monthly_bills AS (
    SELECT
        b.user_id,
        bc.bill_id,
        SUM(bc.item_quantity * i.item_rate) AS bill_amount,
        YEAR(bc.bill_date) AS yr,
        MONTH(bc.bill_date) AS mnth
    FROM booking_commercials bc
    JOIN bookings b ON bc.booking_id = b.booking_id
    JOIN items i ON bc.item_id = i.item_id
    WHERE YEAR(bc.bill_date) = 2021
    GROUP BY b.user_id, bc.bill_id, yr, mnth
),
ranked_bills AS (
    SELECT *,
        DENSE_RANK() OVER (PARTITION BY yr, mnth ORDER BY bill_amount
DESC) AS rnk
    FROM monthly_bills
)
SELECT user_id, bill_id, bill_amount, yr, mnth
FROM ranked_bills
WHERE rnk = 2;
```

Output:set (0.02 sec)

B. For the below schema for a clinic management system, provide queries that solve for below questions :-

Clinics table creation:

```
CREATE TABLE clinics (
    cid VARCHAR(50) PRIMARY KEY,
    clinic_name VARCHAR(100),
    city VARCHAR(50),
    state VARCHAR(50),
    country VARCHAR(50)
);
```

Values insertion:

```
INSERT INTO clinics (cid, clinic_name, city, state, country) VALUES
('cnc-0100001', 'XYZ Clinic', 'Lorem', 'Ipsum', 'Dolor'),
('cnc-0100002', 'ABC Clinic', 'Amet', 'Consectetur', 'Adipiscing');
```

Customer table creation:

```
CREATE TABLE customer (
    uid VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100),
    mobile VARCHAR(15)
);
```

Values insertion:

```
INSERT INTO customer (uid, name, mobile) VALUES

('bk-09f3e-95hj', 'Jon Doe', '975412365'),

('bk-06f3e-96hj', 'Sadie', '987654321');
```

Clinic_sales table creation:

```
CREATE TABLE clinic_sales (

    oid VARCHAR(50) PRIMARY KEY,
```

```
    uid VARCHAR(50) NOT NULL,
    cid VARCHAR(50) NOT NULL,
    amount DECIMAL(12,2) NOT NULL,
    datetime DATETIME NOT NULL,
    sales_channel VARCHAR(50));
```

Values insertion:

```
INSERT INTO clinic_sales (oid, uid, cid, amount, datetime, sales_channel)
VALUES
('ord-00100-00100', 'bk-09f3e-95hj', 'cnc-0100001', 24999, '2021-09-23
12:03:22', 'sodat'),
('ord-00100-00101', 'bk-06f3e-96hj', 'cnc-0100002', 15999, '2021-11-23
12:05:37', 'online');
```

1. Find the revenue we got from each sales channel in a given year

```
SELECT
    sales_channel,
    SUM(amount) AS revenue
FROM clinic_sales
WHERE YEAR(datetime) = 2021
GROUP BY sales_channel
ORDER BY revenue DESC;
```

Output:

```
+---------------+----------+
| sales_channel | revenue  |
+---------------+----------+
```

| sodat | 24999.00 |

| online | 15999.00 |

+---------------+----------+

2 rows in set (0.00 sec)

2. Find top 10 the most valuable customers for a given year

```
SELECT
    c.uid,
    c.name,
    SUM(cs.amount) AS total_spent
FROM clinic_sales cs
JOIN customer c ON cs.uid = c.uid
WHERE YEAR(cs.datetime) = 2021
GROUP BY c.uid, c.name
ORDER BY total_spent DESC
LIMIT 10;
```

Output:

+---------------+---------+-------------+

| uid | name | total_spent |

+---------------+---------+-------------+

| bk-09f3e-95hj | Jon Doe | 24999.00 |

| bk-06f3e-96hj | Sadie | 15999.00 |

```
+---------------+---------+-------------+
```

2 rows in set (0.00 sec)

3. Find month wise revenue, expense, profit , status (profitable / not-profitable) for a given year

```
SELECT
   MONTH(datetime) AS month,
   SUM(amount) AS revenue,
   0 AS expense,  -- replace 0 with actual expense column if available
   SUM(amount) - 0 AS profit,
   CASE
      WHEN SUM(amount) - 0 > 0 THEN 'Profitable'
      ELSE 'Not-Profitable'
   END AS status
FROM clinic_sales
WHERE YEAR(datetime) = 2021
GROUP BY MONTH(datetime)
ORDER BY month;
```

Output:

```
+-------+----------+---------+----------+------------+
| month | revenue  | expense | profit   | status     |
+-------+----------+---------+----------+------------+
|     9 | 24999.00 |       0 | 24999.00 | Profitable |
|    11 | 15999.00 |       0 | 15999.00 | Profitable |
+-------+----------+---------+----------+------------+
2 rows in set (0.03 sec)
```

4. For each city find the most profitable clinic for a given month

```
WITH clinic_profit AS (
   SELECT
      cl.cid,
      cl.clinic_name,
      cl.city,
      SUM(cs.amount) AS revenue
   FROM clinic_sales cs
```

```
    JOIN clinics cl ON cs.cid = cl.cid
    WHERE YEAR(cs.datetime) = 2021 AND MONTH(cs.datetime) = 11
    GROUP BY cl.cid, cl.clinic_name, cl.city
)
SELECT cp.city, cp.clinic_name, cp.revenue AS profit
FROM clinic_profit cp
JOIN (
    SELECT city, MAX(revenue) AS max_profit
    FROM clinic_profit
    GROUP BY city
) AS max_cp
ON cp.city = max_cp.city AND cp.revenue = max_cp.max_profit;
```

Output:

```
+------+-------------+----------+
| city | clinic_name | profit   |
+------+-------------+----------+
| Amet | ABC Clinic  | 15999.00 |
+------+-------------+----------+
1 row in set (0.00 sec)
```

5. For each state find the second least profitable clinic for a given month

```
WITH clinic_profit AS (
    SELECT
        cl.cid,
        cl.clinic_name,
        cl.state,
        SUM(cs.amount) AS revenue
    FROM clinic_sales cs
    JOIN clinics cl ON cs.cid = cl.cid
    WHERE YEAR(cs.datetime) = 2021 AND MONTH(cs.datetime) = 11
    GROUP BY cl.cid, cl.clinic_name, cl.state
),
ranked_clinics AS (
    SELECT *,
          DENSE_RANK() OVER (PARTITION BY state ORDER BY revenue ASC)
AS rnk
    FROM clinic_profit
)
```

```
SELECT state, clinic_name, revenue AS profit
FROM ranked_clinics
WHERE rnk = 2;
```

Output:
set (0.00 sec)

Spreadsheet Proficiency

1. Populating ticket_created_at in the Feedbacks Table
Two worksheets are provided:
- ticket containing ticket_id, created_at, closed_at, outlet_id, cms_id
- feedbacks containing cms_id, feedback_at, feedback_rating, and an empty ticket_created_at.

To populate ticket_created_at, match cms_id in feedbacks with cms_id in ticket.

Formula:
=INDEX(ticket!$B:$B, MATCH(A2, ticket!$E:$E, 0))
2. Outlet-Wise Count of Tickets Created and Closed on the Same Day
Add helper column same_day in ticket sheet.

Formula:
=INT(B2)=INT(C2)
3. Count of Tickets Created and Closed in the Same Hour of the Same Day
Add helper column same_hour in ticket sheet.

Formula:
=AND(INT(B2)=INT(C2), HOUR(B2)=HOUR(C2))
4. Bringing Helper Columns into the Feedbacks Sheet
Use INDEX–MATCH to import outlet_id, same_day, same_hour.

Examples:
Outlet ID:
=INDEX(ticket!$D:$D, MATCH(A2, ticket!$E:$E, 0))

Same day:
=INDEX(ticket!$F:$F, MATCH(A2, ticket!$E:$E, 0))

Same hour:
=INDEX(ticket!$G:$G, MATCH(A2, ticket!$E:$E, 0))
5. Outlet-Wise Count Using COUNTIFS
Same-day count:
=COUNTIFS(ticket_outlet_range, outlet, ticket_same_day_range, TRUE)

Same-hour count:
=COUNTIFS(ticket_outlet_range, outlet, ticket_same_hour_range, TRUE)
6. Pivot Table Alternative
Pivot setup:
Rows → ticket_outlet
Values → Count(cms_id)
Filters → same_day or same_hour

Set filter to TRUE to isolate required tickets.

Python Proficiency

1) Given number of minutes, convert it into human readable form.
Code:
```
minutes = int(input("Enter minutes: "))
hours = minutes // 60
remaining = minutes % 60
print(f"{hours} hrs {remaining} minutes")
```

Output:
-> Enter minutes: 130
2 hrs 10 minutes
-> Enter minutes: 110
1 hrs 50 minutes

2) You are given a string, remove all the duplicates and print the unique string.Use loop in the python.
Code:

```
s = input("Enter a string: ")
result = ""
for ch in s:
    if ch not in result:
        result += ch
print(result)
```

Output:
-> Enter a string: mounikajavvaji
Mounikajv
->Enter a string: dataanalyst
datnlys