Name : Nilesh Patil

Pizaas Project with MySQL

mysql> SHOW TABLES;

```
+---------------+
| Tables_in_pz  |
+---------------+
| order_details |
| orders        |
| pizza_types   |
| pizzas        |
+---------------+
```

1) order_details table.

```
+------------------+--------------+------+-----+---------+----------------+
| Field            | Type         | Null | Key | Default | Extra          |
+------------------+--------------+------+-----+---------+----------------+
| order_details_id | int unsigned | NO   | PRI | NULL    | auto_increment |
| order_id         | int          | NO   | MUL | NULL    |                |
| pizza_id         | varchar(100) | NO   | MUL | NULL    |                |
| quantity         | int          | NO   |     | NULL    |                |
+------------------+--------------+------+-----+---------+----------------+
```

2) orders table.

```
+------------+------+------+-----+---------+----------------+
| Field      | Type | Null | Key | Default | Extra          |
+------------+------+------+-----+---------+----------------+
| order_id   | int  | NO   | PRI | NULL    | auto_increment |
| order_date | date | NO   |     | NULL    |                |
| order_time | time | NO   |     | NULL    |                |
+------------+------+------+-----+---------+----------------+
```

3) pizzas_types table.

```
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| pizza_type_id | varchar(100) | NO   | PRI | NULL    |       |
| name          | varchar(100) | NO   |     | NULL    |       |
| category      | varchar(50)  | NO   |     | NULL    |       |
| ingredients   | text         | NO   |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
```

4) Pizzas table.

```
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| pizza_id      | varchar(100) | NO   | PRI | NULL    |       |
| pizza_type_id | varchar(50)  | NO   | MUL | NULL    |       |
| size          | varchar(10)  | NO   |     | NULL    |       |
| price         | decimal(6,2) | NO   |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
```

Basic:

1. Retrieve the total number of orders placed.

```
mysql> SELECT COUNT(DISTINCT order_id) AS total_orders FROM orders;
```

OUTPUT :-

```
+--------------+
| total_orders |
+--------------+
|        21350 |
+--------------+
1 row in set (0.164 sec)
```

2. Calculate the total revenue generated from pizza sales.

```
mysql> SELECT SUM(order_details.quantity * pizzas.price) AS total_revenue
    -> FROM order_details
    -> INNER JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id;
```

OUTPUT :-

```
+---------------+
| total_revenue |
+---------------+
|     817860.05 |
+---------------+
1 row in set (0.804 sec)
```

3. Identify the highest-priced pizza.

```
mysql> SELECT pizzas.pizza_id,pizzas.price,pizzas.size,pizza_types.name
    -> FROM pizzas
    -> INNER JOIN pizza_types ON pizzas.pizza_type_id=pizza_types.pizza_type_id
    -> ORDER BY pizzas.price DESC
    -> LIMIT 1;
```

OUTPUT :-

```
+---------------+-------+------+-----------------+
| pizza_id      | price | size | name            |
+---------------+-------+------+-----------------+
| the_greek_xxl | 35.95 | XXL  | The Greek Pizza |
+---------------+-------+------+-----------------+
1 row in set (0.014 sec)
```

4. Identify the most common pizza size ordered.

```
mysql> SELECT size, COUNT(*) AS mt_co_sz
    -> FROM pizzas
    -> GROUP BY size,
    -> ORDER BY DESC size
    -> LIMIT 1;
```

OUTPUT :-

```
+------+----------+
| size | mt_co_sz |
+------+----------+
| L    |       31 |
+------+----------+
1 rows in set (0.034 sec)
```

5. List the top 5 most ordered pizza types along with their quantities.

```
mysql> SELECT pizza_id, SUM(quantity) AS total_qty
    -> FROM order_details
    -> GROUP BY pizza_id
    -> ORDER BY SUM(quantity) DESC
    -> LIMIT 5;
```

OUTPUT :-

```
+---------------+---------------+
| pizza_id      | total_qty     |
+---------------+---------------+
| big_meat_s    |          1914 |
| thai_ckn_l    |          1410 |
| five_cheese_l |          1409 |
| four_cheese_l |          1316 |
| classic_dlx_m |          1181 |
+---------------+---------------+
5 rows in set (1.172 sec)
```

Intermediate :

6. Join the necessary tables to find the total quantity of each pizza category ordered.

```
mysql> SELECT category, SUM(quantity) AS total_qty
    -> FROM order_details
    -> JOIN pizzas USING (pizza_id)
    -> JOIN pizza_types USING (pizza_type_id)
    -> GROUP BY category;
```

OUTPUT :-

```
+----------+---------------+
| category | total_qty     |
+----------+---------------+
| Chicken  |         11050 |
| Classic  |         14888 |
| Supreme  |         11987 |
| Veggie   |         11649 |
+----------+---------------+
4 rows in set (1.760 sec)
```

7. Determine the distribution of orders by hour of the day.

```
mysql> SELECT HOUR(time), AS hours COUNT(*) AS counts
    -> FROM orders
    -> GROUP BY HOUR(time);
```

OUTPUT :-

```
+------------+----------+
| hours      | counts   |
+------------+----------+
|         11 |     1231 |
|         12 |     2520 |
|         13 |     2455 |
|         14 |     1472 |
|         15 |     1468 |
|         16 |     1920 |
|         17 |     2336 |
|         18 |     2399 |
|         19 |     2009 |
|         20 |     1642 |
|         21 |     1198 |
|         22 |      663 |
|         23 |       28 |
|         10 |        8 |
|          9 |        1 |
+------------+----------+
15 rows in set (0.262 sec)
```

8. Join relevant tables to find the category-wise distribution of pizzas.

```
mysql> SELECT pizza_types.category, COUNT(pizzas.pizza_id) AS pizza_count
    -> FROM pizzas
    -> JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
    -> GROUP BY pizza_types.category;
```

OUTPUT :-

```
+----------+-------------+
| category | pizza_count |
+----------+-------------+
| Chicken  |          18 |
| Classic  |          26 |
| Supreme  |          25 |
| Veggie   |          27 |
+----------+-------------+
4 rows in set (0.018 sec)
```

9. Group the orders by date and calculate the average number of pizzas ordered per day.

```
mysql> SELECT orders.date,
    -> SUM(order_details.quantity) AS total_pizzas_ordered,
    -> ROUND(SUM(order_details.quantity) / COUNT(DISTINCT orders.order_id), 2) AS avg_pizzas_per_order
    -> FROM orders
    -> JOIN order_details ON orders.order_id = order_details.order_id
    -> GROUP BY orders.date
    -> ORDER BY orders.date;
```

OUTPUT :-

```
+------------+----------------------+----------------------+
| date       | total_pizzas_ordered | avg_pizzas_per_order |
+------------+----------------------+----------------------+
| 2015-01-01 |                  162 |                 2.35 |
| 2015-01-02 |                  165 |                 2.46 |
| 2015-01-03 |                  158 |                 2.39 |
| 2015-01-04 |                  106 |                 2.04 |
|          There are still 350 records here                |
| 2015-01-06 |                  147 |                 2.30 |
| 2015-12-29 |                   80 |                 2.96 |
| 2015-12-30 |                   82 |                 2.56 |
| 2015-12-31 |                  178 |                 2.44 |
+------------+----------------------+----------------------+
358 rows in set (1.294 sec)
```

10. Determine the top 3 most ordered pizza types based on revenue.

```
mysql> SELECT pizza_types.name,
-> SUM(order_details.quantity * pizzas.price) AS total_revenue
-> FROM order_details
-> JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
-> JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
-> GROUP BY pizza_types.name
-> ORDER BY total_revenue DESC
-> LIMIT 3;
```

OUTPUT :-

```
+------------------------------+---------------+
| name                         | total_revenue |
+------------------------------+---------------+
| The Thai Chicken Pizza       |      43434.25 |
| The Barbecue Chicken Pizza   |      42768.00 |
| The California Chicken Pizza |      41409.50 |
+------------------------------+---------------+
3 rows in set (1.892 sec)
```

Advanced:

11. Calculate the percentage contribution of each pizza type to total revenue.

```
mysql> SELECT
-> pizza_types.name AS pizza_type,
-> SUM(pizzas.price * order_details.quantity) AS revenue,
-> ROUND(
-> (SUM(pizzas.price * order_details.quantity) * 100.0) /
-> (SELECT SUM(pizzas.price * order_details.quantity)
-> FROM order_details
-> JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id), 2) AS percentage_contribution
-> FROM order_details
-> JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
-> JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
-> GROUP BY pizza_types.name
-> ORDER BY percentage_contribution DESC;
```

OUTPUT :-

```
+----------------------------------------+---------+-------------------------+
| pizza_type                             | revenue | percentage_contribution |
+----------------------------------------+---------+-------------------------+
| The Thai Chicken Pizza                 | 43434.25 |                    5.31 |
| The Barbecue Chicken Pizza             | 42768.00 |                    5.23 |
| The California Chicken Pizza           | 41409.50 |                    5.06 |
| The Classic Deluxe Pizza               | 38180.50 |                    4.67 |
|                    There are still 24 records here                              |
| The Mediterranean Pizza                | 15360.50 |                    1.88 |
| The Spinach Supreme Pizza              | 15277.75 |                    1.87 |
| The Green Garden Pizza                 | 13955.75 |                    1.71 |
| The Brie Carre Pizza                   | 11588.50 |                    1.42 |
+----------------------------------------+---------+-------------------------+
32 rows in set (3.440 sec)
```

12. Analyze the cumulative revenue generated over time.

```
mysql> SELECT orders.date,
-> SUM(pizzas.price * order_details.quantity) AS daily_revenue,
-> SUM(SUM(pizzas.price * order_details.quantity)) OVER (ORDER BY orders.date) AS cumulative_revenue
-> FROM order_details
-> JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
-> JOIN orders ON order_details.order_id = orders.order_id
-> GROUP BY orders.date
-> ORDER BY orders.date;
```

OUTPUT :-

```
+------------+---------------+--------------------+
| date       | daily_revenue | cumulative_revenue |
+------------+---------------+--------------------+
| 2015-01-01 |       2713.85 |            2713.85 |
| 2015-01-02 |       2731.90 |            5445.75 |
| 2015-01-03 |       2662.40 |            8108.15 |
| 2015-01-04 |       1755.45 |            9863.60 |
|       There are still 350 records here         |
| 2015-12-28 |       1637.20 |          812253.00 |
| 2015-12-29 |       1353.25 |          813606.25 |
| 2015-12-30 |       1337.80 |          814944.05 |
| 2015-12-31 |       2916.00 |          817860.05 |
+------------+---------------+--------------------+
358 rows in set (2.991 sec)
```

13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.

```
mysql> SELECT
-> pizza_types.category,
-> pizza_types.name AS pizza_type,
-> SUM(pizzas.price * order_details.quantity) AS revenue
-> FROM order_details
-> JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
-> JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
-> GROUP BY pizza_types.category, pizza_types.name
-> ORDER BY pizza_types.category, revenue DESC
-> LIMIT 3;
```

OUTPUT :-

```
+----------+------------------------------+----------+
| category | pizzas_type                  | revenue  |
+----------+------------------------------+----------+
| Chicken  | The Thai Chicken Pizza       | 43434.25 |
| Chicken  | The Barbecue Chicken Pizza   | 42768.00 |
| Chicken  | The California Chicken Pizza | 41409.50 |
+----------+------------------------------+----------+
3 rows in set (1.973 sec)
```