Home About Contact

PIZZA SALES DATA ANALYSIS PROJECT

Github link:

Home About Contact

PROJECT DESCRIPTION

This project involves analyzing pizza sales data using MySQL to extract meaningful business insights. I performed data analysis to evaluate revenue trends, top-selling products, and category-wise performance.

During the analysis, I applied SQL concepts such as joins, aggregate functions, subqueries, and window functions to efficiently process and interpret the data.

SQL CONCEPTS APPLIED

During this project, I applied:

- ✓ INNER JOIN
- ✓ GROUP BY
- ✓ ORDER BY
- ✓ Aggregate Functions (SUM, AVG, COUNT)
- ✓ Subqueries
- ✓ Window Functions (OVER())

These concepts helped in performing structured and meaningful data analysis.

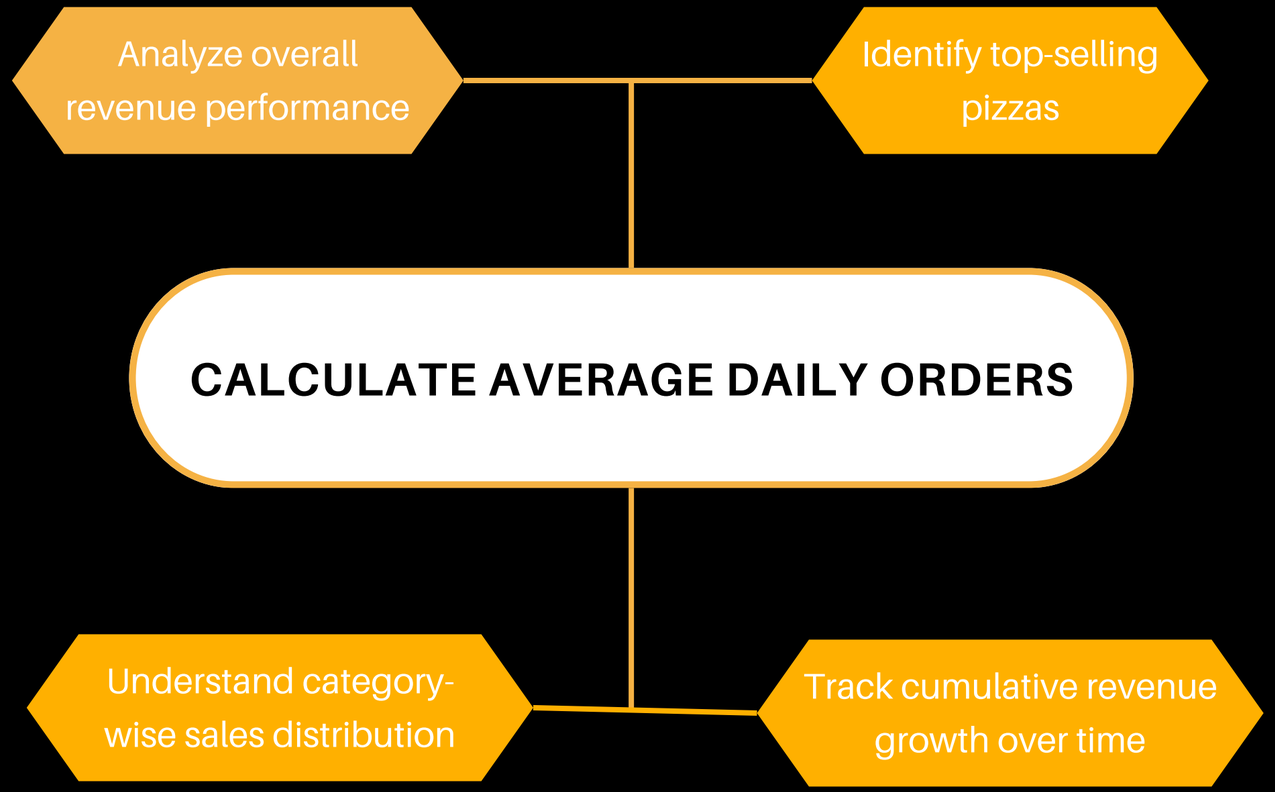
KEY ANALYSIS PERFORMED

Business Analysis Conducted

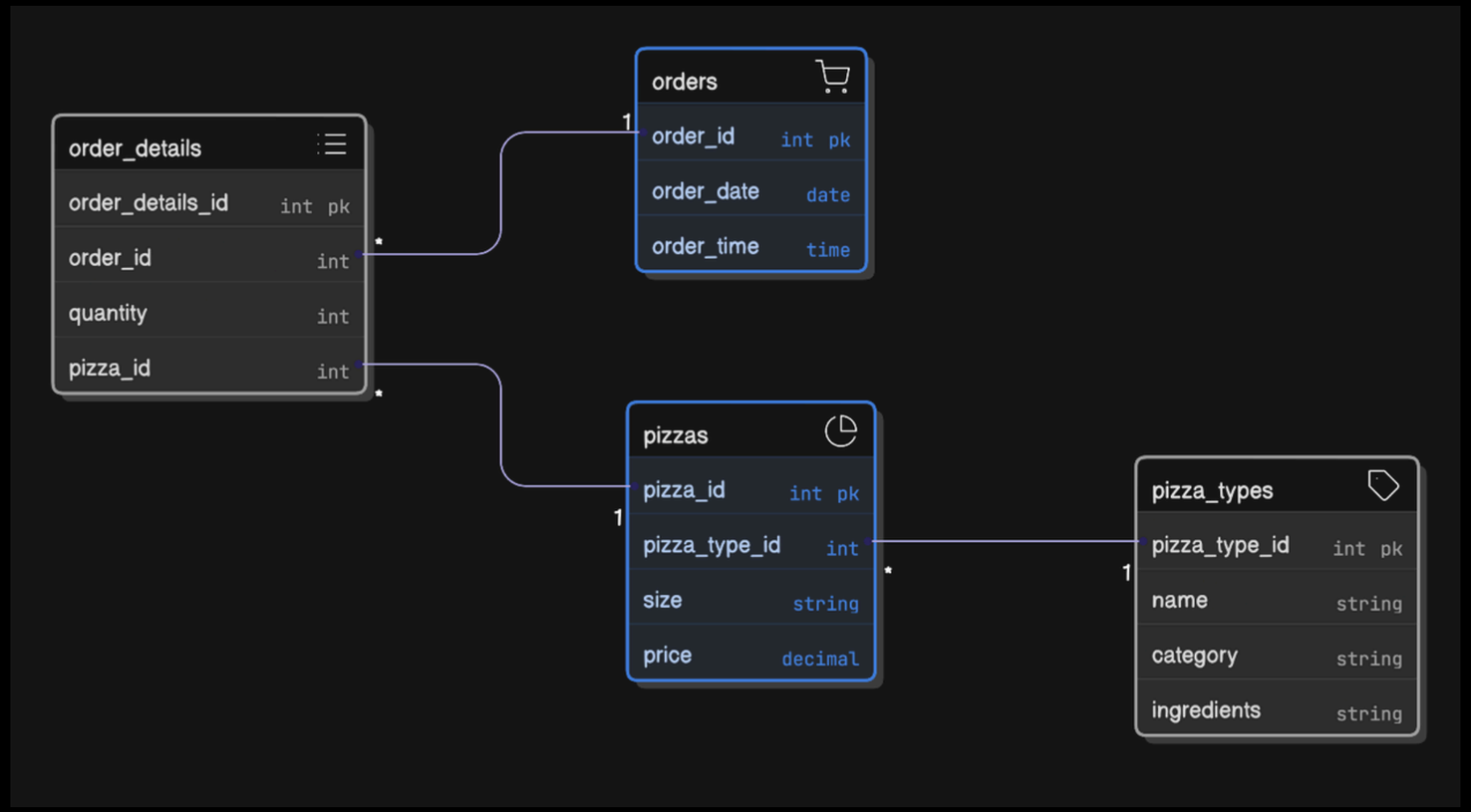
- Total revenue generated
- Top 5 most ordered pizzas
- Revenue by pizza category
- Average number of pizzas ordered per day
- Cumulative revenue over time
- Size-wise sales distribution

Home About Contact

PROBLEM STATEMENT



DATASET & TABLES USED



TOTAL ORDERS

```sql
-- Retrieve the total number if orders placed
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders;
```

100% 12:6

Result Grid Filter Rows: Search Export:

tota...
21350

ction Output

	Time	Action	Response	Duration / Fetch Time
1	06:52:06	use pizzahut	0 row(s) affected	0.0027 sec
2	06:52:06	SELECT COUNT(order_id) AS total_orders FROM orders LIMIT 0, 1000	1 row(s) returned	0.0085 sec / 0.0000...

TOTAL REVENUE

```sql
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_sale
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

```
15          pizzas ON pizzas.pizza_id = order_deta
```

100% 1:14

Result Grid | Filter Rows: Search | Export:

total_sale
817860.05

Result 2

3	06:55:39	use pizzahut		0 row(s) affected	0.031 sec
4	06:55:39	SELECT ROUND(SUM(order_details.quantity * pizzas.price),	2) AS total_sa...	1 row(s) returned	0.086 sec / 0.00000...

Home About Contact

HIGHEST PRICE PIZZA

```sql
-- -- Identify the highest- priced pizza
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

26 -- SELECT

100% 9:24

Result Grid Filter Rows: Search Export: Fetch rows:

name	price
The Greek Pizza	35.95

Result 3

4	06:55:39	SELECT ROUND(SUM(order_details.quantity * pizzas.price),	2) AS total_sa...	1 row(s) returned	0.086 sec / 0.00000...
5	07:11:15	SELECT pizza_types.name, pizzas.price FROM pizza_types	JOIN pizzas...	1 row(s) returned	0.091 sec / 0.00025 s...

TOP 3 REVENUE-GENERATING PIZZAS BY CATEGORY

```sql
-- Determine the top 3 most ordered pizza types based on the revenue for each pizza category
select name, revenue
from
(select category, name, revenue,
rank() over(partition by category order by revenue desc) as rn
from
  (select pizza_types.category, pizza_types.name,
  sum(pizzas.price * order_details.quantity) as revenue
  from pizzas join pizza_types on
  pizzas.pizza_type_id= pizza_types.pizza_type_id join
  order_details on pizzas.pizza_id = order_details.pizza_id
  group by pizza_types.category, pizza_types.name ) as a) as  b
  where rn<=3;
```

Result Grid Filter Rows: Search Export:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5
The Four Cheese Pizza	32265.70000000065
The Mexicana Pizza	26780.75

Result 4

```
6  07:14:33  select name, revenue  from  (select category, name, revenue,  rank() over(partition b...  12 row(s) returned  0.208 sec / 0.00043...
```

AVERAGE NUMBER OF PIZZA ORDERED

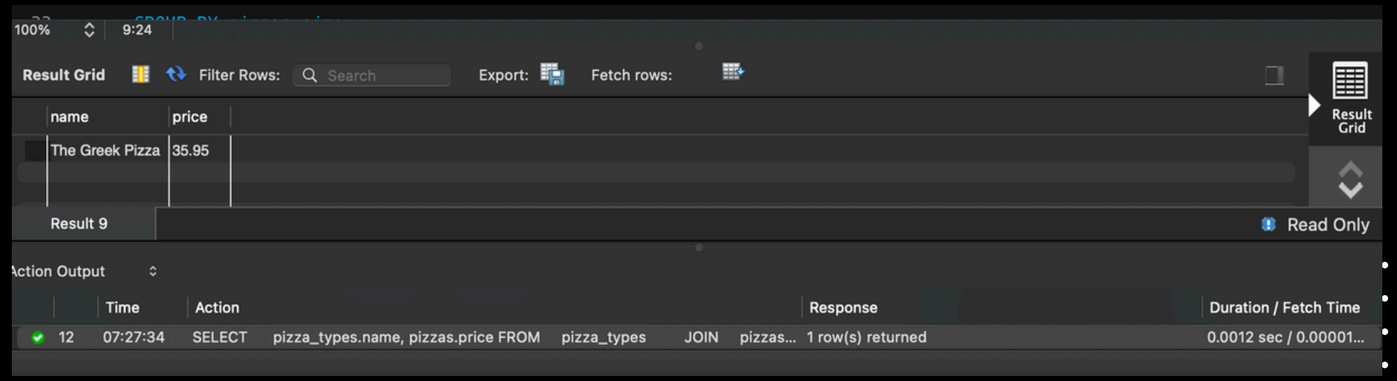
```sql
-- -- Group the orderes by the date and calculate the average number of pizzas ordered per day
SELECT
    ROUND(AVG(quantity), 0) as avg_pizza_Ordered_per_day
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

avg_pizza_Ordered_per_...
138

	Time	Action		Response	Duration / Fetch Time
10	07:24:59	SELECT	ROUND(AVG(quantity), 0) as avg_pizza_Ordered_per_day FROM (SEL...	1 row(s) returned	0.082 sec / 0.000011...

AVERAGE NUMBER OF PIZZA ORDERED

```sql
-- -- Identify the highest- priced pizza
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```



CUMULATIVE REVENUE (WINDOW FUNCTION

```sql
-- -- Analyse the commulative revenue generaed over time
select order_date,
sum(revenue) over(order by order_date) as cum_revenue
from
(select orders.order_date,
sum(order_details.quantity * pizzas.price) as 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.order_date) as sales ;
```

order_date	cum_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65

```
6  07:14:33  select name, revenue  from  (select category, name, revenue,  rank() over(partition b...  12 row(s) returned  0.208 sec / 0.00043...
7  07:19:00  select order_date, sum(revenue) over(order by order_date) as cum_revenue  from  (...  358 row(s) returned  0.116 sec / 0.000067...
```

PERCENTAGE CONTRIBUATION

```sql
-- -- caluculate the percentage contribuation of each pizza type to total revenue
SELECT
    pizza_types.category,
    round((SUM(order_details.quantity * pizzas.price)/(SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_sale
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id)* 100) ,2)as revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC
LIMIT 3;
```

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96

```
7   07:19:00   select order_date, sum(revenue) over(order by order_date) as cum_revenue  from  (...   358 row(s) returned   0.116 sec / 0.000067...
8   07:22:15   SELECT    pizza_types.category,    round((SUM(order_details.quantity * pizzas.pric...   3 row(s) returned   0.141 sec / 0.000010...
```

CONCLUSION

Example content:

- The business generated consistent revenue growth.
- Certain categories dominate total sales.
- A small number of pizza types drive most revenue.
- Window functions helped analyze revenue trends over time.

SKILL DEMONSTRATED

- Joins
- Aggregations
- Group By
- Subqueries
- Window Functions
- Data Cleaning
- Relational Database Understanding