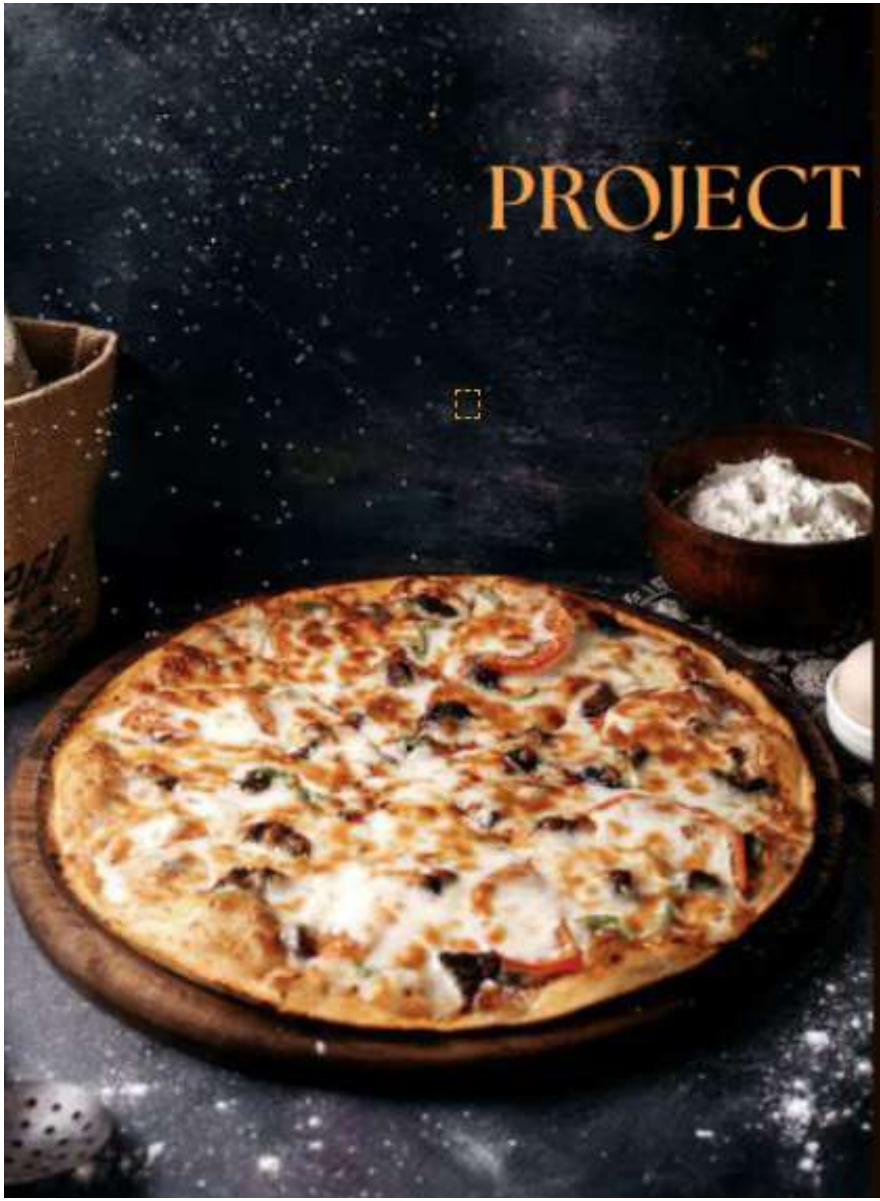


SQL PIZZA SALES
PROJECT

PROJECT OVERVIEW

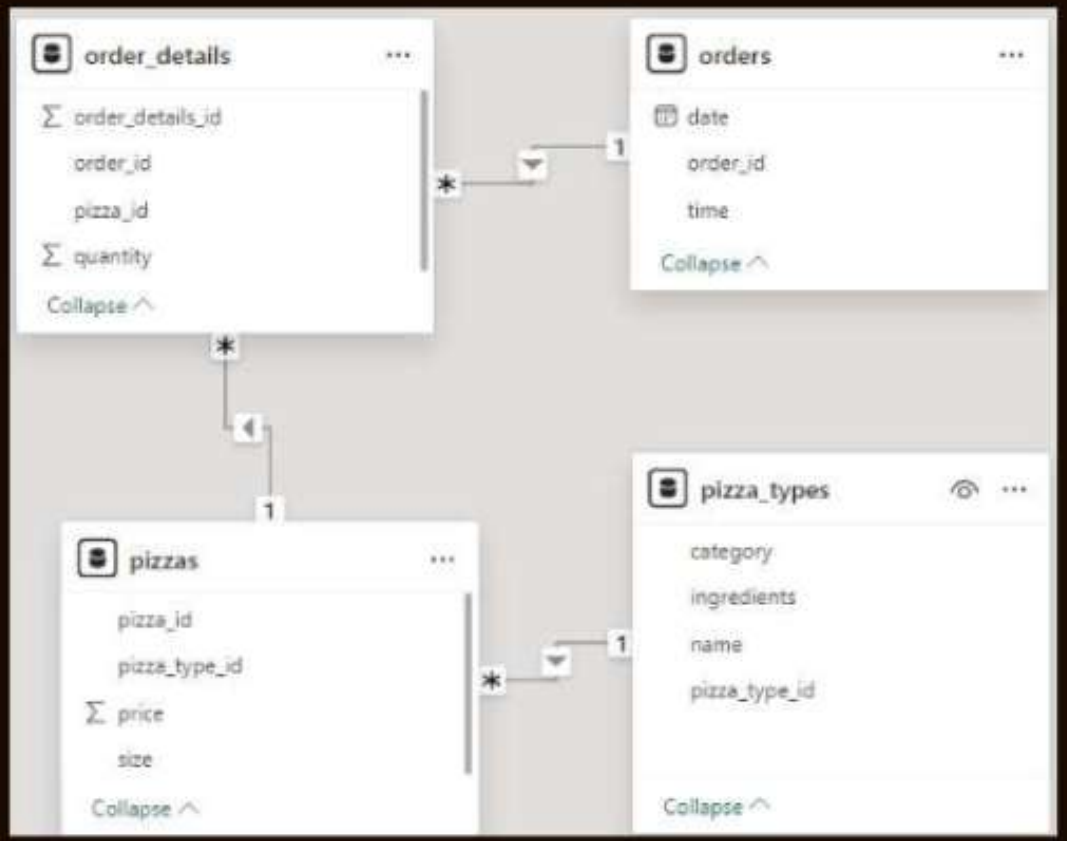
This project uses SQL to analyze the dataset of pizza sales. The goal is to gain valuable insights into the sales operations, including customer segmentation, trends, artist effectiveness and growth, customer interests, and expenditure patterns and revenue.

.

Brian Saldanha

SQL PIZZA SALES
PROJECT

DATABASE SCHEMA



THE MAIN QUERIES WE ARE LOOKING AT ARE

- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza.
- Identify the most common pizza size ordered.
- List the top 5 most ordered pizza types along with their quantities.
- Join the necessary tables to find the total quantity of each pizza category ordered.
- Determine the distribution of orders by hour of the day.
- Join relevant tables to find the category-wise distribution of pizzas.
- Group the orders by date and calculate the average number of pizzas ordered per day.
- Determine the top 3 most ordered pizza types based on revenue.
- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Brian Saldanha

SQL PIZZA SALES
PROJECT

1 RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```
SELECT
    COUNT(order_id) AS Total_Orders
FROM
    orders;
```

	Total_Orders
▶	21350

Brian Saldanha

2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```
SELECT
    ROUND(SUM(d.quantity * p.price), 3) AS totalsales
FROM
    order_details d
        JOIN
    pizzas p USING (pizza_id)
```

	totalsales
▶	817860.05

SQL PIZZA SALES
PROJECT

3. IDENTIFY THE HIGHEST-PRICED PIZZA.

```
SELECT
    t.name, t.category, p.price
FROM
    pizza_types t
        JOIN
    pizzas p USING (pizza_type_id)
ORDER BY p.price DESC
LIMIT 1 ;
```

	name	category	price
▶	The Greek Pizza	Classic	35.95

Brian Saldanha

SQL PIZZA SALES
PROJECT

4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```
SELECT
    p.size, COUNT(d.quantity) 'no of orders'
FROM
    pizzas p
        JOIN
    order_details d USING (pizza_id)
GROUP BY p.size
ORDER BY COUNT(d.quantity) DESC
```

size	no of orders
L	18526
M	15385
S	14137
XL	544
XXL	28

Brian Saldanha

5. LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

SQL PIZZA SALES
PROJECT

```
SELECT
    t.pizza_type_id, t.name, SUM(d.quantity) AS 'no of orders'
FROM
    pizza_types t
        JOIN
    pizzas p USING (pizza_type_id)
        JOIN
    order_details d USING (pizza_id)
GROUP BY t.name , t.pizza_type_id
ORDER BY 'no of orders' DESC
LIMIT 5
```

	pizza_type_id	name	no of orders
▶	hawaiian	The Hawaiian Pizza	2422
	classic_dlx	The Classic Deluxe Pizza	2453
	five_cheese	The Five Cheese Pizza	1409
	ital_supr	The Italian Supreme Pizza	1884
	mexicana	The Mexicana Pizza	1484

Brian Saldanha

6. JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```sql
SELECT
    t.category, SUM(d.quantity) AS quantity
FROM
    pizza_types t
        JOIN
    pizzas p USING (pizza_type_id)
        JOIN
    order_details d USING (pizza_id)
GROUP BY t.category
ORDER BY quantity DESC
```

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

SQL PIZZA SALES
PROJECT

Brian Saldanha

7. DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

SQL PIZZA SALES
PROJECT

```sql
SELECT
    HOUR(o.order_time) AS Hour, COUNT(o.order_id) 'orders'
FROM
    orders o
GROUP BY HOUR(o.order_time)
```

Hour	orders
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

Brian Saldanha

8. JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

SQL PIZZA SALES
PROJECT

```sql
SELECT
    category, COUNT(name) Distribution
FROM
    pizza_types
GROUP BY category
ORDER BY distribution DESC
```

category	Distribution
Supreme	9
Veggie	9
Classic	8
Chicken	6

Brian Saldanha

9. GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

SQL PIZZA SALES
PROJECT

WAY 1
BY CTE

```
--- way 1 by CTE
with data as(SELECT
    o.order_date, SUM(d.quantity) AS quantity
FROM
    orders o
        JOIN
    order_details d USING (order_id)
GROUP BY o.order_date
ORDER BY o.order_date)
SELECT
    ROUND(AVG(quantity), 0) 'avg. order per day'
FROM
    data
```

WAY 2
BY SUBQUERY

```
--- way 2 BY SUBQUERY
SELECT
    ROUND(AVG(quantity), 0) 'avg. order per day'
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details USING (order_id)
    GROUP BY orders.order_date) AS order_quantity
```

avg. order per day
138

-Brian Saldanha

10. DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.


```sql
SELECT
    t.name, SUM(p.price * d.quantity) AS Revenue
FROM
    pizzas p
        JOIN
    pizza_types t USING (pizza_type_id)
        JOIN
    order_details d USING (pizza_id)
GROUP BY t.name
ORDER BY Revenue DESC
LIMIT 3
```

name	Revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

-Brian Saldanha

11. CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.


```sql
SELECT
    t.category AS category,
   ( SUM(p.price * d.quantity)  / (SELECT
    ROUND(SUM(d.quantity * p.price), 3) AS totalsales
FROM
    order_details d
        JOIN
    pizzas p USING (pizza_id)))*100 as revenue
    from pizza_types t
        JOIN
     pizzas p USING (pizza_type_id)
        JOIN
    order_details d USING (pizza_id)
GROUP BY t.category
order by revenue desc
```

category	revenue
Classic	26.90596025566967
Supreme	25.45631126009862
Chicken	23.955137556847287
Veggie	23.682590927384577

-Brian Saldanha

12. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

SQL PIZZA SALES
PROJECT

```sql
select order_date , sum(revenue) over (order by order_date) as cum_revenue
from
(SELECT
    o.order_date, SUM(d.quantity * p.price) AS revenue
FROM
    orders o
        JOIN
    order_details d USING (order_id)
        JOIN
    pizzas p USING (pizza_id)
GROUP BY o.order_date) as sales_data
```

order_date	cum_revenue
2015-01-01	2713.850000000004
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
2015-01-12	27781.7
2015-01-13	29831.300000000003
2015-01-14	32358.700000000
2015-01-15	34343.500000000

Brian Saldanha

13. ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
SELECT * FROM
(SELECT CATEGORY , NAME , REVENUE , RANK() OVER( PARTITION BY CATEGORY ORDER BY REVENUE DESC) AS RN
FROM
(SELECT
    T.CATEGORY, T.NAME, SUM(D.QUANTITY * P.PRICE) AS REVENUE
FROM
    PIZZA_TYPES T
        JOIN
    PIZZAS P USING (PIZZA_TYPE_ID)
        JOIN
    ORDER_DETAILS D USING (PIZZA_ID)
GROUP BY T.CATEGORY , T.NAME) AS A) AS B
WHERE RN <= 3
```

CATEGORY	NAME	REVENUE	RN
Chicken	The Thai Chicken Pizza	43434.25	1
Chicken	The Barbecue Chicken Pizza	42768	2
Chicken	The California Chicken Pizza	41409.5	3
Classic	The Classic Deluxe Pizza	38180.5	1
Classic	The Hawaiian Pizza	32273.25	2
Classic	The Pepperoni Pizza	30161.75	3
Supreme	The Spicy Italian Pizza	34831.25	1
Supreme	The Italian Supreme Pizza	33476.75	2
Supreme	The Sicilian Pizza	30940.5	3
Veggie	The Four Cheese Pizza	32265.70000000065	1
Veggie	The Mexicana Pizza	26780.75	2
Veggie	The Five Cheese Pizza	26066.5	3

SQL PIZZA SALES
PROJECT

THANK YOU

Brian Saldanha

SQL PIZZA SALES

PROJECT

Brian Saldanha