Pizza Sales Analysis Project

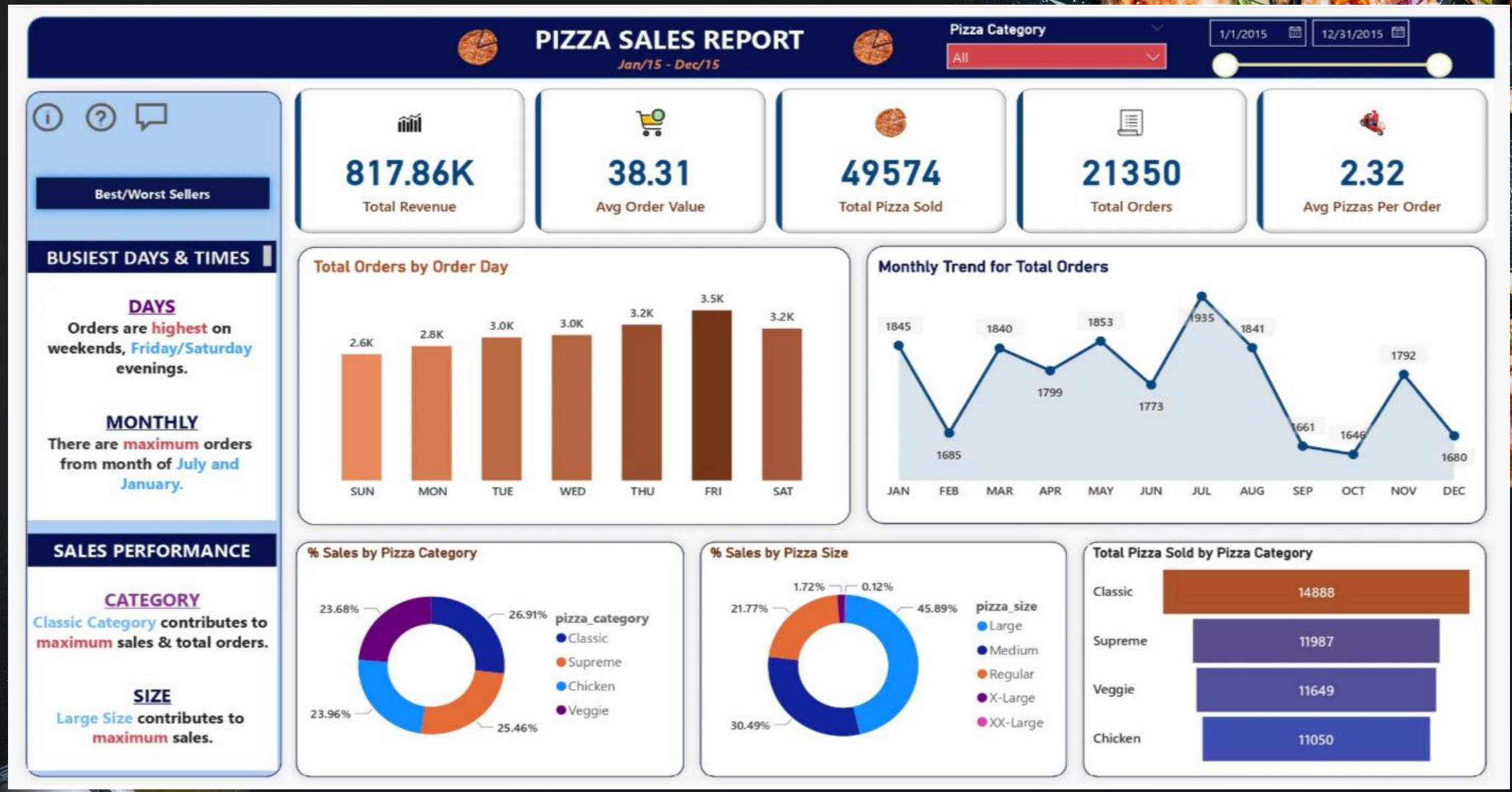
Utilizing SQL Queries for Business Insights

By Swati Singh

About the Project

This project explores pizza sales data using SQL queries to extract valuable business insights. It a nalyzes order trends, revenue generation, and customer preferences, helping businesses make data-driven decisions. Through structured queries, we identify top-selling pizzas, peak sales hours, and category-wise performance to optimize inventory and marketing strategies.

Power BI Dashboard: Pizza Sales


PIZZA SALES REPORT
Jan/15 - Dec/15
Pizza Category
All
1/1/2015
12/31/2015
Best/Worst Sellers
817.86K
Total Revenue
38.31
Avg Order Value
49574
Total Pizza Sold
21350
Total Orders
2.32
Avg Pizzas Per Order
BUSIEST DAYS & TIMES
DAYS
Orders are highest on weekends, Friday/Saturday evenings.
MONTHLY
There are maximum orders from month of July and January.
SALES PERFORMANCE
CATEGORY
Classic Category contributes to maximum sales & total orders.
SIZE
Large Size contributes to maximum sales.
Total Orders by Order Day
2.6K
2.8K
3.0K
3.0K
3.2K
3.5K
3.2K
SUN
MON
TUE
WED
THU
FRI
SAT
Monthly Trend for Total Orders
1845
1685
1840
1799
1853
1773
1935
1841
1661
1646
1792
1680
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
% Sales by Pizza Category
23.68%
26.91%
23.96%
25.46%
pizza_category
Classic
Supreme
Chicken
Veggie
% Sales by Pizza Size
1.72%
0.12%
21.77%
45.89%
30.49%
pizza_size
Large
Medium
Regular
X-Large
XX-Large
Total Pizza Sold by Pizza Category
Classic
14888
Supreme
11987
Veggie
11649
Chicken
11050

-1 Retrieve the total number of orders placed.

SQLQuery pizza sql....PIZZAHUT (sa (54))*

```
USE PIZZAHUT
 select * from dbo.Pizzas
  select * from dbo.orders
  select * from dbo.order_details
  select * from dbo.pizza_types


 --1 Retrieve the total number of orders placed.

 select count(order_id) as total_orders from orders;

 --2 Calculate the total revenue generated from pizza sales.

 select
 round(sum (order_details.quantity *pizzas.price),2) as total_sales
 from order_details join pizzas on pizzas.pizza_id =order_details.pizza_id

 --3  Identify the highest-priced pizza.
```

70 %

Results | Messages

	total_orders
1	21350

-2 Calculate the total revenue generated from pizza sales.

```
--2 Calculate the total revenue generated from pizza sales.

select
round(sum (order_details.quantity *pizzas.price),2) as total_sales
from order_details join pizzas on pizzas.pizza_id =order_details.pizza_id

--3  Identify the highest-priced pizza.
```

Results | Messages

total_sales
817860.05

-3 Identify the highest-priced pizza.

```sql
--3  Identify the highest-priced pizza.

SELECT TOP 1 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;

--4 Identify the most common pizza size ordered.

select pizzas.size, count (order_details.order_details_id) as order_count
from pizzas
```

Results | Messages

name	price
The Greek Pizza	35.9500007629395

-4 Identify the most common pizza size ordered.

```sql
--4 Identify the most common pizza size ordered.

select pizzas.size, count (order_details.order_details_id) as order_count
from pizzas
join  order_details
on pizzas.pizza_id=order_details.pizza_id
group by pizzas.size
order by order_count desc;

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

70 %

Results | Messages

	size	order_count
1	L	18526
2	M	15385
3	S	14137
4	XL	544
5	XXL	28

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

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

SELECT TOP 5 pizza_types.name, SUM(TRY_CAST(order_details.quantity AS INT)) AS quantity
FROM pizza_types
JOIN pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC;

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

SELECT pizza_types.category,
       SUM(CAST(order_details.quantity AS INT)) AS quantity
```

70 %

data type int

Results | Messages

	name	quantity
1	The Classic Deluxe Pizza	2453
2	The Barbecue Chicken Pizza	2432
3	The Hawaiian Pizza	2422
4	The Pepperoni Pizza	2418
5	The Thai Chicken Pizza	2371

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

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

SELECT pizza_types.category,
       SUM(CAST(order_details.quantity AS INT)) AS quantity
FROM pizza_types
JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details
ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;

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

70 %

Results | Messages

	category	quantity
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	Chicken	11050

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

```sql
--7 Determine the distribution of orders by hour of the day.

SELECT DATEPART(HOUR, time) AS hour, COUNT(order_id) AS order_count
FROM orders
GROUP BY DATEPART(HOUR, time)
ORDER BY hour;

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

select category ,count (name) from pizza_types
group by category;

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


SELECT AVG(total_quantity) AS average_quantity
```

0 %

Results | Messages

	hour	order_count
1	9	1
2	10	8
3	11	1231
4	12	2520
5	13	2455
6	14	1472
7	15	1468
8	16	1920
9	17	2336
10	18	2399
11	19	2009

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

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

select category ,count (name) from pizza_types
group by category;

--9 Group the orders by date and calculate the average number of pizzas or


SELECT AVG(total_quantity) AS average_quantity
```

70 %

Results | Messages

	category	(No column name)
1	Chicken	6
2	Classic	8
3	Supreme	9
4	Veggie	9

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

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


SELECT AVG(total_quantity) AS average_quantity
FROM (
    SELECT orders.date, SUM(CAST(order_details.quantity AS INT)) AS total_quantity
    FROM orders
    JOIN order_details
    ON orders.order_id = order_details.order_id
    GROUP BY orders.date
) AS order_quantity;

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

SELECT TOP 3 pt.name AS pizza_name,
             SUM(CAST(od.quantity AS INT) * p.price) AS total_revenue
```

70 %

Results | Messages

	average_quantity
1	138

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

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

SELECT TOP 3 pt.name AS pizza_name,
             SUM(CAST(od.quantity AS INT) * p.price) AS total_revenue
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY total_revenue DESC;

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

SELECT pt.name AS pizza_name,
       SUM(CAST(od.quantity AS INT) * p.price) AS total_revenue,
       (SUM(CAST(od.quantity AS INT) * p.price) * 100.0 /
        (SELECT SUM(CAST(quantity AS INT) * price)
```

70 %

Results | Messages

	pizza_name	total_revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5

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

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

SELECT pt.name AS pizza_name,
       SUM(CAST(od.quantity AS INT) * p.price) AS total_revenue,
       (SUM(CAST(od.quantity AS INT) * p.price) * 100.0 /
        (SELECT SUM(CAST(quantity AS INT) * price)
         FROM order_details
         JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id)) AS revenue_percentage
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY revenue_percentage DESC;

--12 Analyze the cumulative revenue generated over time.
```

70 %

Results | Messages

	pizza_name	total_revenue	revenue_percentage
1	The Thai Chicken Pizza	43434.25	5.31071910841851
2	The Barbecue Chicken Pizza	42768	5.22925651597168
3	The California Chicken Pizza	41409.5	5.06315230308009
4	The Classic Deluxe Pizza	38180.5	4.66834147979931
5	The Spicy Italian Pizza	34831.25	4.25882765202812
6	The Southwest Chicken Pizza	34705.75	4.24348272842275
7	The Italian Supreme Pizza	33476.75	4.09321252036698
8	The Hawaiian Pizza	32273.25	3.94606020515533
9	The Four Cheese Pizza	32265.7010040283	3.9451371870957
10	The Sicilian Pizza	30940.5	3.78310445268476
11	The Pepperoni Pizza	30161.75	3.68788645063152

-12 Analyze the cumulative revenue generated over time.

```sql
--12 Analyze the cumulative revenue generated over time.

SELECT orders.date AS order_date,
       SUM(CAST(order_details.quantity AS INT) * pizzas.price) AS daily_revenue,
       SUM(SUM(CAST(order_details.quantity AS INT) * pizzas.price))
           OVER (ORDER BY orders.date) AS cumulative_revenue
FROM orders
JOIN order_details ON orders.order_id = order_details.order_id
JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
GROUP BY orders.date
ORDER BY orders.date;

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

WITH PizzaRevenue AS (
    SELECT pt.category,
```

70 %

Results | Messages

	order_date	daily_revenue	cumulative_revenue
1	2015-01-01	2713.85000228882	2713.85000228882
2	2015-01-02	2731.90000152588	5445.7500038147
3	2015-01-03	2662.40000343323	8108.15000724792
4	2015-01-04	1755.45000076294	9863.60000801086
5	2015-01-05	2065.95000076294	11929.5500087738
6	2015-01-06	2428.95000267029	14358.5000114441
7	2015-01-07	2202.20000076294	16560.700012207
8	2015-01-08	2838.35000610352	19399.0500183105
9	2015-01-09	2127.35000419617	21526.4000225067
10	2015-01-10	2463.95000267029	23990.350025177
11	2015-01-11	1872.30000114441	25862.6500263214

Query executed successfully.

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

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

WITH PizzaRevenue AS (
    SELECT pt.category,
           pt.name AS pizza_name,
           SUM(CAST(od.quantity AS INT) * p.price) AS total_revenue,
           ROW_NUMBER() OVER (PARTITION BY pt.category ORDER BY SUM(CAST(od.quantity AS INT) * p.price) DESC) AS rank
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    GROUP BY pt.category, pt.name
)
SELECT category, pizza_name, total_revenue
FROM PizzaRevenue
WHERE rank <= 3
ORDER BY category, rank;
```

70 %

Results | Messages

	category	pizza_name	total_revenue
1	Chicken	The Thai Chicken Pizza	43434.25
2	Chicken	The Barbecue Chicken Pizza	42768
3	Chicken	The California Chicken Pizza	41409.5
4	Classic	The Classic Deluxe Pizza	38180.5
5	Classic	The Hawaiian Pizza	32273.25
6	Classic	The Pepperoni Pizza	30161.75
7	Supreme	The Spicy Italian Pizza	34831.25
8	Supreme	The Italian Supreme Pizza	33476.75
9	Supreme	The Sicilian Pizza	30940.5
10	Veggie	The Four Cheese Pizza	32265.7010040283
11	Veggie	The Mexicana Pizza	26780.75

Query executed successfully. DESKTOP-P5PUPF7\SWATI (16.0...

Conclusion

- This project provided **valuable insights into pizza sales trends** using SQL queries.
- Identified **top-selling pizzas, revenue contributors, and peak order times**.
- Helped understand **customer preferences** and optimize **inventory and marketing strategies**.
- Demonstrated how **data-driven decisions** can enhance business performance.

THANK YOU