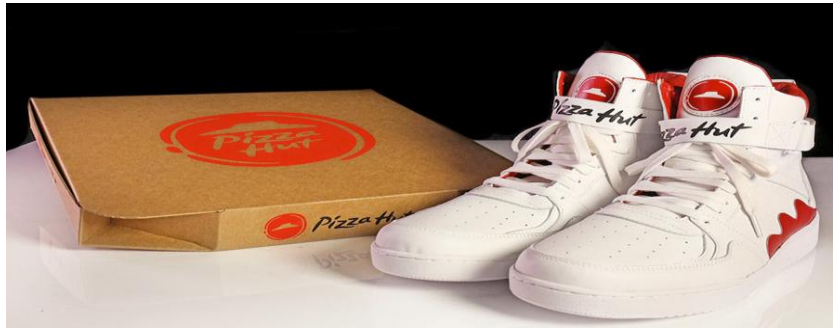
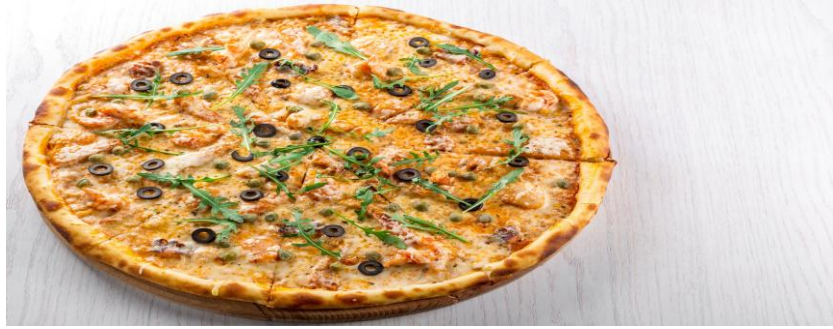
PIZZAHUT SALES ANALYSIS USING SQL





PROJECT OVERVIEW

OBJECTIVE: TO ANALYZE PIZZAHUT'S SALES DATA USING SQL TO GAIN INSIGHTS ON PERFORMANCE, CUSTOMER PREFERENCES, AND REVENUE .

TOOLS: SQL(MYSQL) , EXCEL , CSV FILES

FOUCS AREAS:-

ORDER VOLUME AND TIMING BEST-SELLING PIZZAS

REVENUE PATTERNS PIZZA CATEGORY BREAKDOWNS.

DATASET SUMMARY

- ORDERS.CSV – ORDER_ID , ORDER_DATE , ORDER_TIME
- ORDER_DETAILS.CSV – ORDER_DETAILS , ORDER_ID , PIZZA_ID, QUANTITY
- PIZZAS.CSV- PIZZA_ID , PIZZA_TYPE_ID , SIZE , PRICE
- PIZZA_TYPES.CSV – PIZZA_TYPE_ID , NAME , CATEGORY, INGREDIENTS
-

BASIC SQL ANALYSIS

. TOTAL ORDES PLACED : QUERY USING COUNT(*) ON ORDERS

. TOTAL REVENUES : SUM(QUANTITY*PRICE) AFTER JOINS

. HIGHEST PRICED PIZZA: ORDER BY PRICE DESC LIMIT 1

. MOST COMMON SIZE : GROUP BY SIZE WITH MAX COUNT

. TOP 5 MOST ORDERED PIZZAS:

. JOINED ORDER DETAILS WITH PIZZA TYPES

. RANKED BY QUANTITY ORDERD.

INTERMEDIATE SQL ANALYSIS

. TOTAL QUANTITY BY CATEGORY : JOIN & GROUP BY CATEGORY

. ORDER DISTRIBUTION BY HOUR : EXTRACT (HOUR FROM TIME)

. CATEGORY- WISE PIZAA DISTRBUTION: JOIN PIZZA TABLES AND GROUP

. AVG PIZZAS PER DAY : GROUP BY DATE AVG(QUANTITY)

. TOP 3 REVENUE- GENERATING PIZZAS:

.SUM (QUNATITY*PRICE) ORDER BY REVENUE DESC LIMIT 3

ADVANCED SQL ANALYSIS

. % CONTRIBUTION TO REVENUE (PER PIZZA)

. PIZZA REVENUE/ TOTAL REVENUE)*100

. CUMULATIVE REVENUE OVER TIME:

. ORDER BY DATE+ SUM (....) OVER(ORDER BY DATE)

. TOP 3 PIZZAS BY REVENUE IN EACH CATEGORY:

. GROUP BY CATEGORY AND PIZZA WINDOW FUNCTION OR SUBQUERY FOR TOP 3

MY SQL SCREENSHOST

Limit to 10000 rows

```sql
CREATE DATABASE PizzHut;
USE PIZZHUT;
CREATE TABLE ORDERS (
Order_id int not null,
Order_date date not null,
Order_time time not null,
primary key (order_id));

create table orders_details (
order_details_id int not null,
order_id int not null,
pizza_id text not null,
quantity int not null,
primary key (order_details_id));
```

INSERT VALUES IN TABLES

```sql
INSERT INTO ORDERS (order_id, order_date, order_time) VALUES
(1, '2025-05-25', '12:30:00'),
(2, '2025-05-26', '14:45:00'),
(3, '2025-05-27', '18:15:00');
```

```sql
INSERT INTO ORDERS_DETAILS (order_details_id, order_id, pizza_id, quantity) VALUES
(101, 1, 'PEP001', 2),
(102, 1, 'VEG002', 1),
(103, 2, 'MARG003', 3),
(104, 3, 'CHEESE004', 1),
(105, 3, 'BBQ005', 2);
```

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```
---- RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.
```

- `SELECT * FROM ORDERS;`
- `SELECT COUNT(Order_id) from  orders;`
- `SELECT COUNT(Order_id) as total_orders from orders;`

Result Grid | Filter Rows:

total_orders
3

WE ADD NEW TABLES FOR JOINS

```sql
USE PIZZHUT;
CREATE TABLE PIZZAS (
    pizza_id VARCHAR(10) NOT NULL,  -- changed from TEXT to VARCHAR(10)
    name VARCHAR(100),
    price DECIMAL(5,2),
    PRIMARY KEY (pizza_id)
);

INSERT INTO PIZZAS (pizza_id, name, price) VALUES
('PEP001', 'Pepperoni', 8.99),
('VEG002', 'Veggie Delight', 7.49),
('MARG003', 'Margherita', 6.99),
('CHEESE004', 'Cheese Burst', 9.49),
('BBQ005', 'BBQ Chicken', 10.99);
```

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES

```
----- CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES
SELECT ROUND(SUM(ORDERS_DETAILS.QUANTITY*PRICE),
2) AS TOTAL_SALES
FROM
ORDERS_DETAILS
JOIN
PIZZAS ON PIZZAS.PIZZA_ID = ORDERS_DETAILS.PIZZA_ID;
```

Result Grid | Filter Rows:

TOTAL_SALES
77.91

Result 7

IDENTIFY THE HIGHEST - PRICED PIZZA

```sql
CREATE TABLE PIZZA_TYPES (
    pizza_type_id INT PRIMARY KEY,
    name VARCHAR(100)
);
INSERT INTO PIZZA_TYPES (pizza_type_id, name) VALUES
(1, 'Veg'),
(2, 'Non-Veg'),
(3, 'Cheese'),
(4, 'BBQ');
```

```sql
CREATE TABLE PIZZA (
    pizza_id VARCHAR(10) PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(5,2),
    pizza_type_id INT,
    FOREIGN KEY (pizza_type_id) REFERENCES PIZZA_TYPES(pizza_type_id)
);
INSERT INTO PIZZA (pizza_id, name, price, pizza_type_id) VALUES
('PEP001', 'Pepperoni', 8.99, 2),
('VEG002', 'Veggie Delight', 7.49, 1),
('MARG003', 'Margherita', 6.99, 1),
('CHEESE004', 'Cheese Burst', 9.49, 3),
('BBQ005', 'BBQ Chicken', 10.99, 4);
```

```sql
---- IDENTIFY THE HIGHEST - PRICED PIZZA
SELECT PIZZA_TYPES.name AS type_name, PIZZA.name AS pizza_name, PIZZA.price
FROM PIZZA
JOIN PIZZA_TYPES ON PIZZA_TYPES.pizza_type_id = PIZZA.pizza_type_id
ORDER BY PIZZA.price DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export:

type_name	pizza_name	price
BBQ	BBQ Chicken	10.99

identify the most common pizza size ordered

```sql
ALTER TABLE PIZZAS ADD size VARCHAR(10);
UPDATE PIZZAS SET size = 'Large' WHERE pizza_id = 'PEP001';
UPDATE PIZZAS SET size = 'Medium' WHERE pizza_id = 'VEG002';
UPDATE PIZZAS SET size = 'small' where pizza_id = 'MARG003';
UPDATE PIZZAS SET size = 'large' where pizza_id = 'CHEESE004';
UPDATE PIZZAS SET size = 'Medium' WHERE pizza_id = 'VEG002';
UPDATE PIZZAS SET size = 'Medium' WHERE pizza_id = 'BBQ005';
```

```sql
---- identify the most common pizza size ordered
SELECT pizzas.size,COUNT(orders_details.order_details_id) AS order_count
FROM pizzas
JOIN orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```

Limit to 10000 rows

```sql
---- identify the most common pizza size ordered
SELECT pizzas.size,COUNT(orders_details.order_details_id) AS order_count
FROM pizzas
JOIN orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC
limit 1;
```

Result Grid | Filter Rows:

size	order_count
Medium	2

Result Grid | Filter Rows:

size	order_count
Medium	2
NULL	1
small	1
large	1

LIST THE TOP 5 MOST ORDERD PIZZA TYPES ALONG WITH THEIR QUANTITIES

```sql
--- list the top 5 most orderd pizza types along with their quantities.--

SELECT pizza_types.name, SUM(orders_details.quantity) AS quantity
FROM pizza_types
JOIN pizzas ON pizza_types.name = pizzas.name
JOIN orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;
```

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

```
--- Join the necessary tables to find the total quantity ofeach pizza category ordered

SELECT
    pizza_types.NAME,
    SUM(orders_details.quantity) AS total_quantity
FROM
    orders_details
JOIN pizzas ON orders_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_quantity DESC;
```

Result Grid | Filter

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY

```sql
--- Determine the distribution of orders by hour of the day.
select HOUR(ORDER_TIME) AS HOUR , COUNT(ORDER_ID) AS ORDER_COUNT
FROM ORDERS
GROUP BY HOUR(ORDER_TIME);
```

Result Grid | Filter Rows:

HOUR	ORDER_COUNT
12	1
14	1
18	1

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

```
---- Join relevant tables to find the category-wis distribution of pizzas.
SELECT CATEGORY, COUNT(NAME)
FROM PIZZA_TYPES
GROUP BY CATEGORY ;
```

Result Grid | Filter Rows:

CATEGORY	COUNT(NAME)
Veg	4

Result Grid | Filter Rows:

CATEGORY	COUNT(NAME)
NON-VEG	4

Result Grid | Filter Rows:

CATEGORY	COUNT(NAME)
CLASSICS	4

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

```
--- Group the orders by date and calculate the average umber of pizzas ordered per day.
SELECT ROUND (AVG(QUANTITY),0) AS AVG_PIZZA_ORDERED_PER_DAY
FROM (SELECT ORDERS.ORDER_DATE, SUM(ORDERS_DETAILS.QUANTITY) AS QUANTITY
FROM ORDERS
JOIN ORDERS_DETAILS ON ORDERS.ORDER_ID = ORDERS_DETAILS.ORDER_ID
GROUP BY ORDERS.ORDER_DATE) AS ORDER_QUANTITY;
```

Result Grid | Filter Rows:

AVG_PIZZA_ORDERED_PER_DAY
3

Calculate the percentage contribution of each pizza type to total revenue

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

SELECT pt.category,(SUM(od.quantity * p.price) /
        (SELECT ROUND(SUM(od2.quantity * p2.price), 2)
         FROM orders_details od2
JOIN pizzas p2 ON od2.pizza_id = p2.pizza_id)) * 100 AS revenue_percentage
FROM pizza_types pt
    JOIN pizzas p ON pt.name = p.name
    JOIN orders_details od ON od.pizza_id = p.pizza_id
GROUP BY pt.category
ORDER BY revenue_percentage DESC;
```

Result Grid | Filter Rows:

category	revenue
Classic	26.90596025566967
Supreme	25.45631126009862
Chicken	23.955137556847287
Veggie	23.682590927384577

Analyze the cumulative revenue generated over time

```sql
--- Analyze the cumulative revenue generated over time
SELECT order_date,SUM(revenue) OVER (ORDER BY order_date) AS cum_revenue
FROM (SELECT orders.order_date,SUM(orders_details.quantity * pizzas.price) AS revenue
    FROM orders_details
        JOIN pizzas ON orders_details.pizza_id = pizzas.pizza_id
        JOIN orders ON orders.order_id = orders_details.order_id
    GROUP BY orders.order_date) AS sales;
```

Result Grid | Filter Rows:

order_date	cum_revenue
2025-05-25	25.47
2025-05-26	46.44
2025-05-27	77.91

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

```
    --- Determine the top 3 most ordered pizza types based onrevenue 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(orders_details.quantity * pizzas.price) AS revenue
        FROM pizza_types
            JOIN pizzas ON pizza_types.name = pizzas.name
            JOIN orders_details ON orders_details.pizza_id = pizzas.pizza_id
        GROUP BY pizza_types.category, pizza_types.nam) AS a
) AS b
WHERE rn <= 3;
```

Result Grid | Filter Rows:

name	revenue

Presented by: SONU BHARTIA Aspiring Data Analyst

EMAIL-sonubhartia6@gmail.com

contact – 9737425336