PIZZA

HUB

SQL Queries
easy - advance

ABOUT

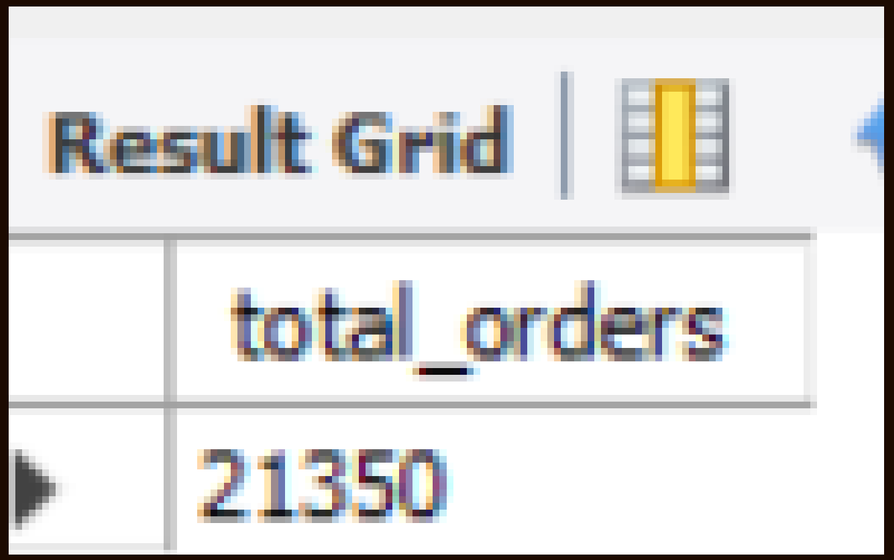
This project delves into a Pizza Hut dataset, leveraging SQL to extract and analyze data. We explore trends, uncover insights, and unlock the potential of data for informed decision-making within the pizza industry.

BASIC QUERIES

1. Retrieve the total number of orders placed.
2. Calculate the total revenue generated from pizza sales.
3. Identify the highest-priced pizza.
4. Identify the most common pizza size ordered.
5. List the top 5 most ordered pizza types along with their quantities.

BASIC Query 1

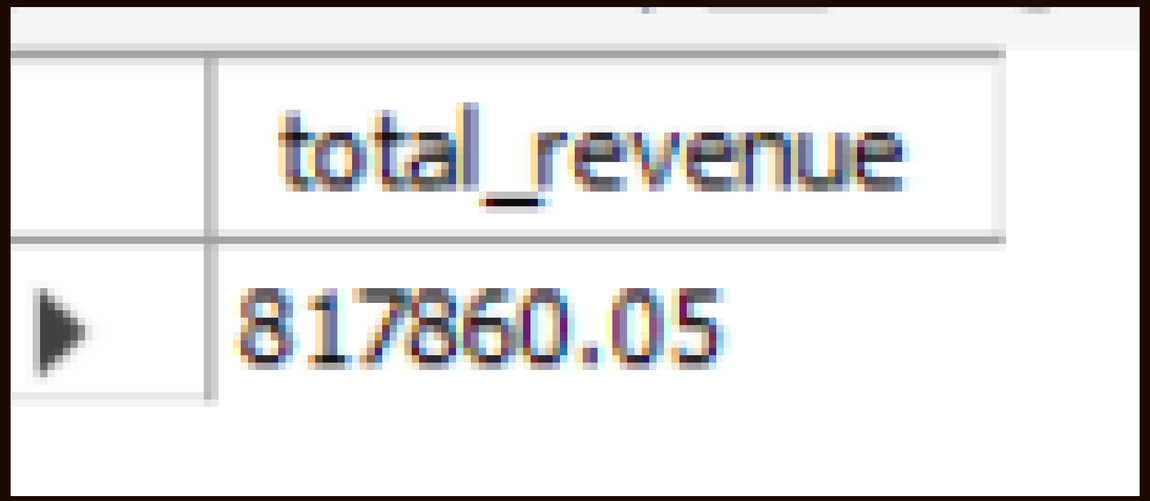
```sql
/* 1.Retrieve the total number of orders placed. */
SELECT count(order_id) AS total_orders FROM orders;
```



BASIC QUERIES 2

```sql
/* 2.Calculate the total revenue generated from pizza sales. */

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



total_revenue
817860.05

BASIC Query 3

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

name	price
The Greek Pizza	35.95

BASIC Query 4

```sql
/* 4. Identify the most common pizza size ordered. */
 SELECT pizzas.size, count(order_details.quantity) 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 Limit 1;
```

size	order_count
L	18526

BASIC Query 5

```sql
-- 5.  List the top 5 most ordered pizza types along with their quantities
SELECT pizza_types.name , sum(order_details.quantity) AS order_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 order_quantity DESC LIMIT 5;
```

name	order_quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

INTERMEDIATE QUERIES

1. Join the necessary tables to find the total quantity of each pizza category ordered.
2. Determine the distribution of orders by hour of the day.
3. Join relevant tables to find the category-wise orders of pizzas.
4. Join relevant tables to find the category-wise distribution of pizzas.
5. Group the orders by date and calculate the average number of pizzas ordered per day.
6. Determine REVENUE generated per order of pizzas.
7. Determine the top 3 most ordered pizza types based on revenue.

INTERMEDIATE QUERY 1

```
10
11    /* 1. Join the necessary tables to find the total quantity of each pizza category ordered. */
12
13    SELECT pizza_types.category, SUM(order_details.quantity)
14    FROM pizza_types
15    JOIN pizzas
16    ON pizza_types.pizza_type_id = pizzas.pizza_type_id
17    JOIN order_details
18    ON pizzas.pizza_id = order_details.pizza_id
19    GROUP BY pizza_types.category;
```

category	SUM(order_details.quantity)
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

INTERMEDIATE QUERY 2

```sql
/* 2. Determine the distribution of orders by hour of the day. */
SELECT HOUR(orders.time) AS hour, COUNT(orders.order_id) AS order_placed
FROM orders
GROUP BY hour
ORDER BY hour ASC;
```

Result Grid | Filter Rows:

category	SUM(order_details.quantity)
Classic	14388
Veggie	11649
Supreme	11987
Chicken	11050

INTERMEDIATE QUERY 3

```sql
/* 3. Join relevant tables to find the category-wise orders of pizzas.
SELECT pizza_types.category, COUNT(orders.order_id)
FROM pizza_types
JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details
ON pizzas.pizza_id = order_details.pizza_id
JOIN orders
ON order_details.order_id = orders.order_id
GROUP BY pizza_types.category;
```

category	COUNT(orders.order_id)
Classic	14579
Veggie	11449
Supreme	11777
Chicken	10815

INTERMEDIATE QUERY 4

```sql
-- 4. Join relevant tables to find the category-wise distribution of pizzas.
SELECT category, COUNT(name)
FROM pizza_types
GROUP BY category;
```

Result Grid | Filter Rows:

category	COUNT(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

INTERMEDIATE QUERY 5

```sql
/* 5. Group the orders by date and calculate the average 'number of pizzas ordered per day'*/

SELECT ROUND(AVG(quantity_ordered), 0) AS avg_orders_per_day FROM
(SELECT date, sum(order_details.quantity) AS quantity_ordered
FROM orders
JOIN order_details
ON orders.order_id = order_details.order_id
GROUP BY date) AS per_date_quantity_ordered;
```

Result Grid | Filter Rows

avg_orders_per_day
138

INTERMEDIATE QUERY 6

```sql
/* 6. Determine REVENUE generated per order of pizzas. */

SELECT pizza_types.name AS pizza_name , (order_details.quantity * pizzas.price ) AS revenue_per_order
FROM pizza_types
JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details
ON pizzas.pizza_id = order_details.pizza_id;
```

pizza_name	revenue_per_order
The Hawaiian Pizza	13.25
The Classic Deluxe Pizza	16
The Five Cheese Pizza	18.5
The Italian Supreme Pizza	20.75
The Mexicana Pizza	16
The Thai Chicken Pizza	20.75
The Italian Supreme Pizza	16.5
The Prosciutto and Arugula Pizza	20.75
The Italian Supreme Pizza	16.5
The Italian Supreme Pizza	16.5
The Barbecue Chicken Pizza	12.75
The Greek Pizza	12
The Spinach Supreme Pizza	12.5
The Spinach Supreme Pizza	12.5
The Classic Deluxe Pizza	12
The Green Garden Pizza	12
The Italian Capocollo Pizza	20.5
The Italian Supreme Pizza	20.75
The Italian Supreme Pizza	12.5
The Mexicana Pizza	12

Result 6

INTERMEDIATE QUERY 7

```sql
/* 7. Determine the top 3 most ordered 'pizza types' based on revenue. */

SELECT pizza_types.name AS pizza_name , SUM(order_details.quantity * pizzas.price ) AS REVENUE
FROM pizza_types
JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details
ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.name            /* get revenue based on type of pizza */
ORDER BY REVENUE DESC
LIMIT 3;
```

Result Grid | Filter Rows:

pizza_name	REVENUE
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

ADVANCE QUERIES

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

2. Analyze the cumulative revenue generated over time.

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

Advance Query 1

```sql
/* 1. Calculate the percentage contribution of each pizza category to total revenue. */

/* USE FORMULLA:  % contribution of each category = (each_category_revenue/ Total_revenue ) * 100  */

SELECT pizza_types.category , ( SUM(order_details.quantity * pizzas.price) /
                                ( SELECT SUM(order_details.quantity * pizzas.price) AS total_revenue
                                FROM order_details JOIN pizzas
                                ON pizzas.pizza_id = order_details.pizza_id )
    ) * 100  AS percentage_revenue
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 percentage_revenue DESC;
```

Result Grid | Filter Rows:

category	percentage_revenue
Classic	26.905960255669903
Supreme	25.45631126009884
Chicken	23.955137556847493
Veggie	23.682590927384783

INTERMEDIATE QUERY 2

```sql
/* 2. Analyze the cumulative revenue generated over time. ( i.e bases on date ) */

SELECT date,
ROUND(SUM(revenue) over(ORDER BY date),2) AS Commulative_Revenue
FROM
(SELECT orders.date  , ROUND(SUM(order_details.quantity * pizzas.price),2) AS revenue  /* this subquery provided simple
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) AS date_wise_revenue;
```

Result Grid | Filter Rows:

date	Commulative_Revenue
2015-01-01	2713.85
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	23990.35
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.3
2015-01-14	32358.7
2015-01-15	34343.5
2015-01-16	36937.65
2015-01-17	39001.75
2015-01-18	40978.6
2015-01-19	43365.75
2015-01-20	45763.65
2015-01-21	47804.2
2015-01-22	50300.9
2015-01-23	52724.6
2015-01-24	55013.85

INTERMEDIATE QUERY 3

```sql
/* 3. Determine the top 3 most ordered pizza types based on revenue for each pizza category. */
SELECT name, category , revenue    /* STEP 3 : NEED ONLY top 3/topRanked pizzaz from below subQuery table2 */
FROM
(SELECT name, category , revenue,                    /* STEP 2:  from subquery below */
RANK() OVER(PARTITION BY category ORDER BY revenue DESC) AS ranking          /* ranking over categories (repeat ranking when new category comes) */
FROM
(SELECT pizza_types.name, pizza_types.category, SUM(order_details.quantity * pizzas.price) AS revenue        /* STEP 1: SUBQUERY: provedes name,category based revenue */
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, pizza_types.category) AS rev_table1)  AS rank_table2
WHERE ranking <=3 ;
```

Result Grid | Filter Rows: | Export:

name	category	revenue
The Thai Chicken Pizza	Chicken	43434.25
The Barbecue Chicken Pizza	Chicken	42768
The California Chicken Pizza	Chicken	41409.5
The Classic Deluxe Pizza	Classic	38180.5
The Hawaiian Pizza	Classic	32273.25
The Pepperoni Pizza	Classic	30161.75
The Spicy Italian Pizza	Supreme	34831.25
The Italian Supreme Pizza	Supreme	33476.75
The Sicilian Pizza	Supreme	30940.5
The Four Cheese Pizza	Veggie	32265.70000000065
The Mexicana Pizza	Veggie	26780.75
The Five Cheese Pizza	Veggie	26066.5

THANK YOU

github: https://github.com/MushtaqUmar/SQL-pIzza_hub-project