PIZZA SALES ANALYSIS

USING MYSQL QUERIES

In this project ----

Through comprehensive SQL query analysis on a pizza database, I have extracted key insights regarding order trends, customer preferences, and operational efficiency, which will help in optimizing sales and enhancing customer satisfaction.

AGENDA

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.

AGENDA

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

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```sql
-- retrieve the total no of order placed
SELECT COUNT(ORDER_ID) AS TOTAL_ORDERS FROM ORDERS;
```

	TOTAL_ORDERS
▶	21350

- COUNT(): Returns the number of rows.
- SUM(): Returns the total sum of a numeric column.
- AVG(): Returns the average value of a numeric column.
- MAX(): Returns the maximum value.
- MIN(): Returns the minimum value.

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```sql
-- 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
```

total_sales
817860.05

- CONCAT(): Concatenates two or more strings.
- SUBSTRING(): Extracts a portion of a string.
- UPPER() / LOWER(): Converts a string to uppercase or lowercase.
- LENGTH(): Returns the length of a string.
- REPLACE(): Replaces occurrences of a specified substring.

IDENTIFY THE HIGHEST-PRICED 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 4
```

name	price
The Greek Pizza	35.95
The Greek Pizza	25.5
The Brie Carre Pizza	23.65
The Italian Vegetables Pizza	21

- NOW(): Returns the current date and time.
- DATE(): Extracts the date part of a date or datetime expression.
- YEAR(), MONTH(), DAY(): Extracts the year, month, or day part of a date.
- DATEDIFF(): Returns the number of days between two dates

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```sql
-- 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;
```

size	order_count
L	18526
M	15385
S	14137
XL	544
XXL	28

- ROUND(): Rounds a number to the specified number of decimal places.
- ABS(): Returns the absolute value of a number.
- **CEIL() or CEILING(): Rounds a number up to the nearest integer.
- FLOOR(): Rounds a number down to the nearest integer.
- MOD(): Returns the remainder of a division operation.

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

```sql
-- List the top 5 most ordered pizza types along with their quantities.
select pizza_types.name,
sum(order_details.quantity) 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 limit 5;
```

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

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

```sql
-- Join the necessary tables to find the
-- total quantity of each pizza category ordered.
select pizza_types.category,
sum(order_details.quantity) 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;
```

	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) order by hour;
```

hour	order_count
9	1
10	8
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

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

```sql
-- Group the orders by date and calculate the average number of
--  pizzas ordered per day.
select category, count(name) from pizza_types
group by category;
```

	category	count(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

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

```sql
-- Group the orders by date and calculate the average
-- number of pizzas ordered per day.
select avg(quantity) 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(quantity)
▶	138.4749

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

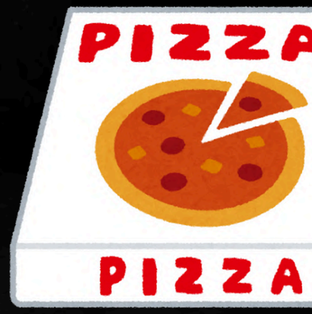
```sql
-- Determine the top 3 most ordered pizza types based on revenue.

select pizza_types.name,
sum(order_details.quantity*pizzas.price) 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.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

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

```sql
-- Calculate the percentage contribution 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_sales
from order_details join pizzas on pizzas.pizza_id=order_details.pizza_id)*100,
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



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 sum_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 orders.order_id=order_details.order_id
group by orders.order_date)as sales;
```

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
2015-01-12	27781.7
2015-01-13	29831.300000000003
2015-01-14	32358.700000000004
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001

2015-12-15	787777
2015-12-16	790011.8
2015-12-17	791892.55
2015-12-18	794778.8500000001
2015-12-19	797083.05
2015-12-20	799187.9500000001
2015-12-21	801288.65
2015-12-22	803171.6
2015-12-23	805415.9
2015-12-24	807553.75
2015-12-26	809196.8
2015-12-27	810615.8
2015-12-28	812253
2015-12-29	813606.25
2015-12-30	814944.05
2015-12-31	817860.05

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

```sql
-- Determine the top 3 most ordered pizza
--  types based on 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((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 order_details.pizza_id=pizzas.pizza_id
group by pizza_types.category,pizza_types.name) as a) as b
where rn <=3;
```

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
The Five Cheese Pizza	26066.5

THANKYOU

In conclusion, the SQL query results provide valuable insights into the pizza database, revealing key trends and patterns in sales, customer preferences, and order details, which can be leveraged to enhance business strategies and improve operational efficiency.

TANYA MALL