PIZZA SALES PROJECT USING SQL

NEXT >

Problem Statement

This project focuses on analyzing pizza sales data using SQL operations on multiple tables, including pizzas, orders, order_details, and pizza_types. It offers hands-on experience with key data analytics concepts such as filtering, aggregation, joins, and subqueries. The goal is to extract insights, including popular pizzas, order patterns, and sales trends, to support data-driven business decisions. Through this project, I have developed a deeper understanding of SQL, enhancing my ability to manipulate, analyze, and interpret data efficiently, thereby strengthening my overall data analytics skills.

NEXT >

TABLES USED

- pizzas
- orders
- order_details
- pizza_types

NEXT >

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```
USE pizzahut;
SELECT
COUNT(order_id)  AS
total_orders     FROM
orders;
```

	total_orders
▶	21350

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

IDENTIFY THE HIGHEST-PRICED PIZZA..

```
SELECT MAX(price)
AS max_price
FROM pizzas;
```

	max_price
▶	35.95

< BACK

NEXT >

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 LIMIT 1;
```

	size	order_count
▶	L	18526

< BACK

NEXT >

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

```
SELECT pizza_types.name,
COUNT(order_details.quantity) AS
order_quantity
FROM pizzas JOIN pizza_types
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 order_quantity DESC LIMIT 5;
```

name	order_quantity
The Classic Deluxe Pizza	2416
The Barbecue Chicken Pizza	2372
The Hawaiian Pizza	2370
The Pepperoni Pizza	2369
The Thai Chicken Pizza	2315

< BACK

NEXT >

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

```
SELECT pizza_types.category
,COUNT(order_details.quantity) AS cat_qty
FROM pizzas JOIN pizza_types
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 cat_qty;
```

	category	cat_qty
▶	Chicken	10815
	Veggie	11449
	Supreme	11777
	Classic	14579

< BACK

NEXT >

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.(NO OF ORDERS IN AN HOUR)

```
SELECT
HOUR(order_time) AS
hour ,
COUNT(order_id) AS
order_count
FROM orders
GROUP BY hour;
```

hour	order_count
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

< BACK

NEXT >

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

```
SELECT    category    ,
COUNT(pizza_type_id)
AS cat_count
FROM pizza_types
GROUP BY category;
```

category	cat_count
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

```
SELECT ROUND(AVG(order_quantity),0)
FROM
(SELECT orders.order_date,
SUM(order_details.quantity) AS
order_quantity
FROM orders JOIN order_details
ON orders.order_id =
order_details.order_id
GROUP BY order_date) AS or_qty;
```

ROUND(AVG(order_quantity),0)
138

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

```
SELECT pizza_types.name ,
SUM(order_details.quantity * pizzas.price) AS
revenue
FROM order_details JOIN pizzas
ON order_details.pizza_id = pizzas.pizza_id
JOIN pizza_types ON pizza_types.pizza_type_id
= pizzas.pizza_type_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

< BACK

NEXT >

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_revenue
FROM order_details JOIN pizzas ON pizzas.pizza_id =
order_details.pizza_id)* 100,2) AS revenue
FROM order_details JOIN pizzas
ON order_details.pizza_id = pizzas.pizza_id
JOIN pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.category
ORDER BY revenue DESC;
```

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
SELECT order_date,
SUM(revenue) OVER (ORDER BY order_date) AS cum_rev
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;
```

order_date	cum_rev
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

< BACK

EXT >

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;
```

< BACK

NEXT >

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

< BACK

NEXT >

THANK YOU!

< BACK