Home About Contact

PIZZA SALES REPORT

- USING SQL

SHODWE
Pizza Resto

Home About Contact

ABOUT PROJECT

ANLYSIS OF PIZZA SALES USING SQL QUERIES ON 4 DATA SETS PROVIDED AND SOLVED 10 QUESTIONS TO THE INSIGHTS FROM THE RAW DATA SETS. THIS PROJECT ALSO COVERS JOINS , SUB-QUERIES AND OTHER ADVANCED CONCEPTS

SHODWE
Pizza Resto

About Contact

ABOUT DATASETS

THERE ARE 4 TABLES (DATASETS) IN THE PROJECT INCLUDING BOTH PRIMARY KEY AND FORIEGN KEY TO PERFORM JOINS

1. ORDER DETAILS TABLE - ORDER_ID , ORDER_DETAILS , PIZZA_ID (FORIEGN KEY) , QUANTITY
2. ORDERS TABLE - ORDER_ID , DATE , TIME
3. PIZZA TYPES TABLE - PIZZA_TYPE_ID (FORIEGN KEY) , NAME , CATEGORY , INGREDIENTS
4. PIZZAS TABLE- PIZZA_ID , PIZZA_TYPE_ID , SIZE , PRICE

Q1 . RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```sql
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders;
```

total_orders
21350

Q2 . CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

SQL File 3* | SQL File 3* | SQL File 4* | SQL File 5* | SQL File 6* | SQL File 7* | SQL File 10*

Limit to 1000 rows

```sql
-- Calculate the total revenue generated from pizza sales.
select * from pizzas;
select Round(SUM(orders_details.quantity * pizzas.price),2) as total_sales
from orders_details join pizzas
on orders_details.pizza_id = pizzas.pizza_id;
```

Result Grid

	total_sales
▶	817860.05

Q3 . IDENTIFY THE HIGHEST-PRICED PIZZA.

SQL File 3* | SQL File 3* | SQL File 4* | SQL File 5* | SQL File 6* | SQL File 7* | SQL File 10* | SQL File 11*

Limit to 1000 rows

```
1   -- Identify the highest-priced pizza.
2   select pizza_types.name , pizzas.price from
3   pizza_types join pizzas
4   on pizza_types.pizza_type_id = pizzas.pizza_type_id order by pizzas.price desc limit 1;
```

Result Grid | Filter Row

name	price
The Greek Pizza	35.95

Q4 . IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED

SQL File 3* | SQL File 3* | SQL File 4* | SQL File 5* | SQL File 6* | SQL File 7* | SQL File 10*

Limit to 1000 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
L	18526

Q5 . LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

```
1   -- List the top 5 most ordered pizza types along with their quantities.
2   select pizza_types.name , sum(orders_details.quantity) as total_quantities
3   from pizza_types join pizzas
4   on pizza_types.pizza_type_id = pizzas.pizza_type_id
5   join orders_details
6   on orders_details.pizza_id = pizzas.pizza_id
7   group by pizza_types.name order by total_quantities desc limit 5;
8
```

Result Grid | Filter Rows:

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

Q6. JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

SQL File 3* | SQL File 3* | SQL File 4* | SQL File 5* | SQL File 6* | SQL File 7* | SQL File 10* | SQL File 11*

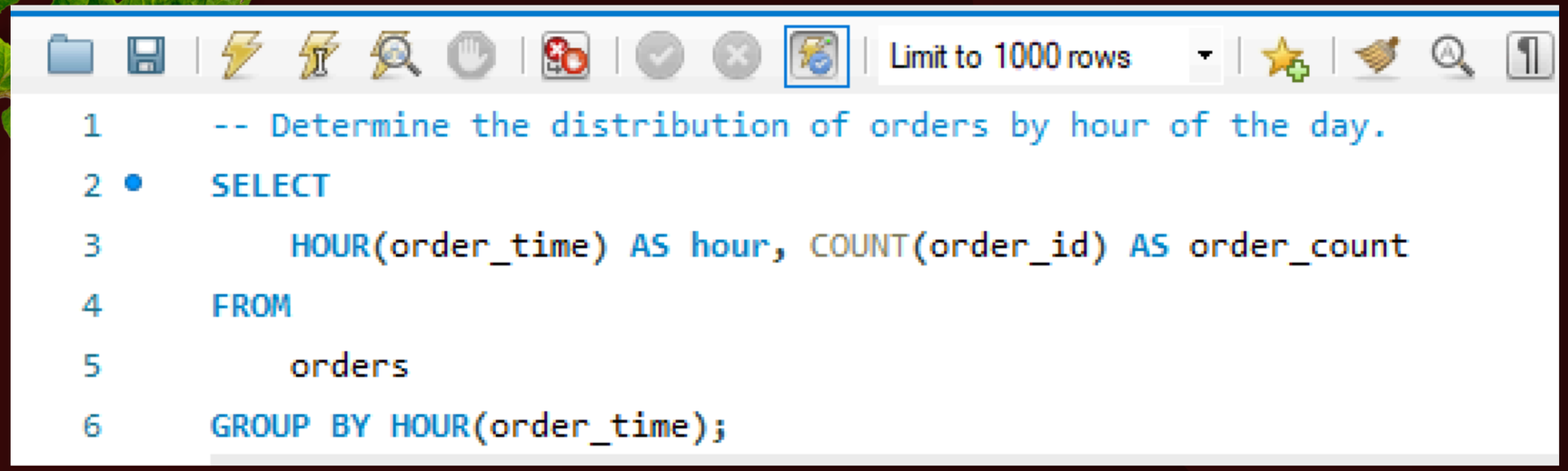
Limit to 1000 rows

```sql
-- Join the necessary tables to find the total quantity of each pizza category ordered.
select pizza_types.category , sum(orders_details.quantity) as Total
from pizza_types join pizzas on
pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on orders_details.pizza_id = pizzas.pizza_id
group by pizza_types.category  ;
```

Result Grid

category	Total
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

Q7. DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.



hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468

Q8. JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

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

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

Q9. 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
    ROUND(AVG(quantity), 0) AS avg_pizzas_order_perday
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;
```

	avg_pizzas_order_perday
▶	138

Q10 . DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```
1   -- Determine the top 3 most ordered pizza types based on revenue.
2   select pizza_types.name , sum(orders_details.quantity * pizzas.price) as revenue
3   from pizza_types join pizzas
4   on pizza_types.pizza_type_id = pizzas.pizza_type_id
5   join orders_details on
6   orders_details.pizza_id = pizzas.pizza_id
```

Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

Q11 . CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

Limit to 1000 rows

```sql
-- Calculate the percentage contribution of each pizza type to total revenue.
use pizzadb;
select pizza_types.category , (sum(orders_details.quantity * pizzas.price) / (select Round(SUM(orders_details.quantity * pizzas.price),2) as total_sales
from orders_details join pizzas
on orders_details.pizza_id = pizzas.pizza_id)) * 100 as revenue
from pizza_types join pizzas
on pizza_types.pizza_type_id = pizzas.pizza_type_id
join orders_details
on orders_details.pizza_id = pizzas.pizza_id
group by pizza_types.category order by revenue desc;
```

	category	revenue
▶	Classic	26.90596025566967
	Supreme	25.456311260098 62
	Chicken	23.955137556847287
	Veggie	23.682590927384577

Q12 . 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;
```

order_date	cum_revenue
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

Q13 . DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

```sql
use pizzadb;
select name , revenue
from
(select name , category , 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.pizza_type_id = pizzas.pizza_type_id
join orders_details
on orders_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

SHODWE
Pizza Resto

Home

THANK YOU