SQL PROJECT ON

PIZZA SALES

ABOUT ME

- Name: Adithya Kannoth
- Skills Used: SQL, Data Analysis
- What I Did in This Project:
  -Performed end-to-end analysis of Pizza Sales dataset
  -Wrote SQL queries ranging from basic to advanced
  -Extracted revenue insights, order patterns, top sellers
  -Applied joins, window functions, aggregates & grouping

DATASETS USED

orders.csv

-Contains each order placed
-Columns include:

- order_id
- order_date
- order_time

pizzas.csv

-Contains details of each pizza variant (size + price)
-Columns include:

- pizza_id
- pizza_type_id
- size (S, M, L, XL, etc.)
- price

order_details.csv

-Contains item-level details of every order
-Columns include:

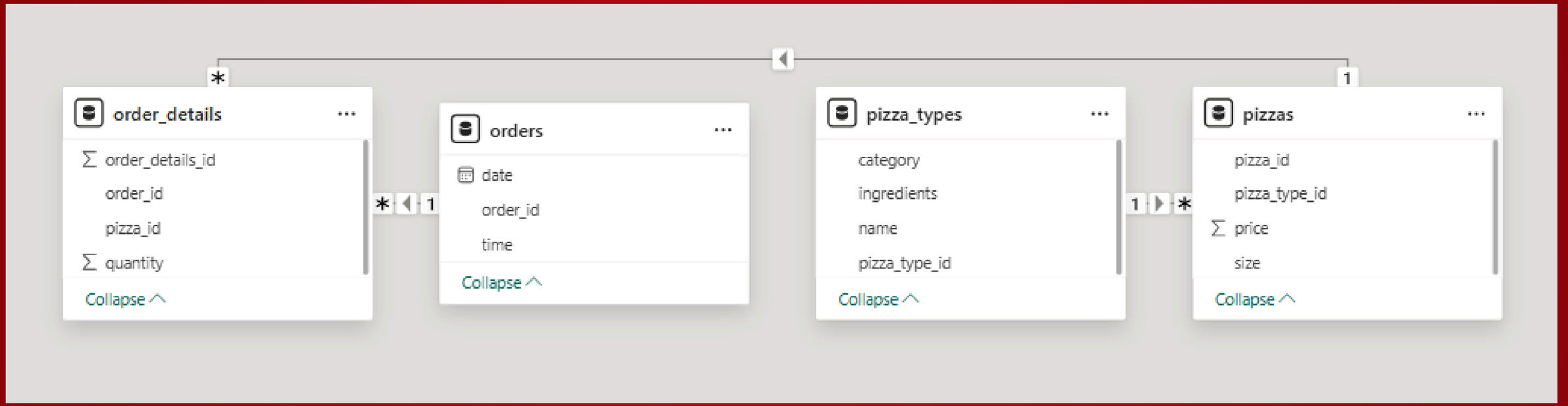
- order_details_id
- order_id (links to orders table)
- pizza_id
- quantity

pizza_types.csv

-Contains each pizza recipe/type information
-Columns include:

- pizza_type_id
- name
- category (Classic, Veggie, Supreme, etc.)
- ingredients

DATABASE SCHEMA



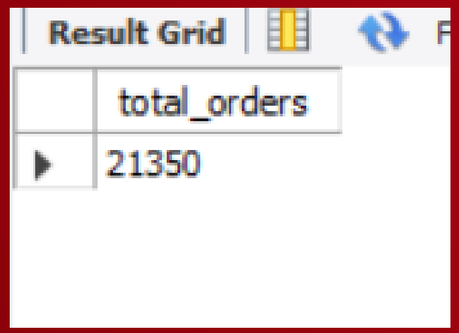
-- RETREIVE THE TOTAL NO OF ORDERS PLACED.

SQL QUERY

```sql
-- Retreive the Total no of orders placed.

select count(order_id) as total_orders from orders
```

OUTPUT



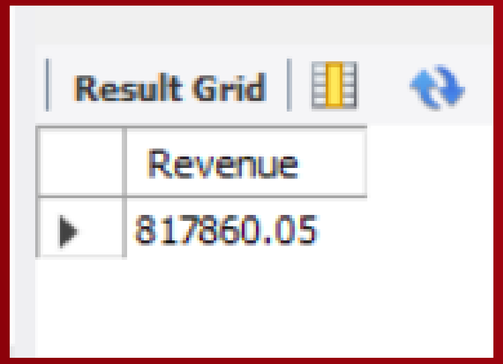
-- CALCULATE TOTAL REVENUE GENERATED FROM PIZZA SALES

SQL QUERY

```sql
-- Calculate total Revenue generated from pizza sales

select round(sum(p.price * o.quantity),2) as Revenue
from order_details as o
join pizzas as p
on o.pizza_id=p.pizza_id
```

OUTPUT



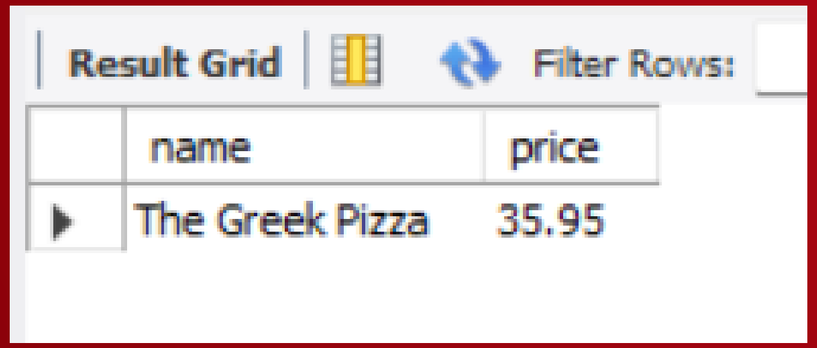
-- IDENTIFY THE HIGHEST PRICED PIZZA

SQL QUERY

```sql
-- Identify the highest priced pizza

select pt.name, p.price
from pizza_types as pt
join pizzas as p
on pt.pizza_type_id= p.pizza_type_id
order by p.price desc limit 1;
```

OUTPUT



	name	price
▶	The Greek Pizza	35.95

-- IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED

SQL QUERY

```sql
-- Identify the most common pizza size ordered

select p.size as pizza_size ,count(o.order_details_id) as order_count
from order_details as o
join pizzas as p
group by p.size order by order_count desc limit 1;
```

OUTPUT

Result Grid | Filter Rows:

	pizza_size	order_count
▶	S	1555840

-- LIST THE TOP 5 MOST ORDERED PIZZA TYPE ALONG WITH THEIR QUANTITIES

SQL QUERY

```sql
-- List the top 5 most ordered pizza type along with their quantities

select pt.name,sum(od.quantity) as order_quantity
from pizza_types as pt
join pizzas as p
on pt.pizza_type_id= p.pizza_type_id
join order_details as od
on od.pizza_id=p.pizza_id
group by pt.name order by order_quantity desc limit 5;
```

OUTPUT

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

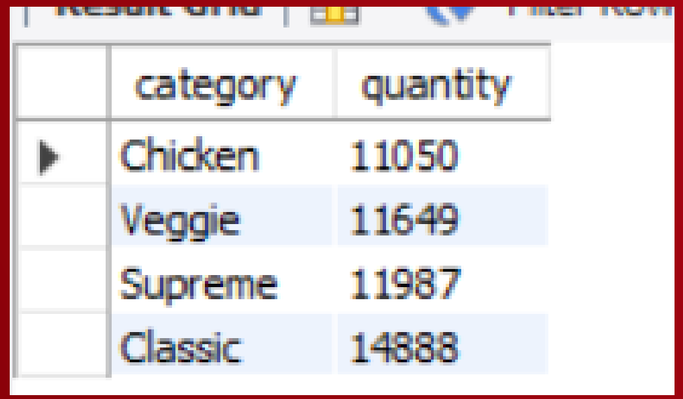
-- FIND TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED

SQL QUERY

```sql
-- Find Total Quantity of each pizza Category ordered

select pizza_types.category as 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 pizzas.pizza_id=order_details.pizza_id
group by category order by quantity
```

OUTPUT



category	quantity
Chicken	11050
Veggie	11649
Supreme	11987
Classic	14888

-- DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY

SQL QUERY

```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
```

OUTPUT

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
9	1

-- CATEGORY WISE DISTRIBUTION OF PIZZAS

SQL QUERY

```sql
-- Category wise distribution of pizzas

select category, count(name ) as pizza_name
from pizza_types
group by category order by pizza_name desc
```

OUTPUT

	category	pizza_name
▶	Supreme	9
	Veggie	9
	Classic	8
	Chicken	6

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

SQL QUERY

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

select round(avg(quantity),2) as avg_quantity
from
(select sum(order_details.quantity) as quantity, orders.order_date as order_date
from order_details join orders
on order_details.order_id=orders.order_id
group by order_date order by quantity desc) sub
```

OUTPUT

	avg_quantity
▶	138.47

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

SQL QUERY

```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 pizza_types.pizza_type_id=pizzas.pizza_type_id
join order_details
on order_details.pizza_id= pizzas.pizza_id
group by name order by revenue desc limit 3
```

OUTPUT

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 QUERY

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

select round(sum(pizzas.price*order_details.quantity)/(select round(sum(p.price * o.quantity),2)
from order_details as o join pizzas as p
on o.pizza_id=p.pizza_id)*100,2) as revenue_percent, pizza_types.category
from pizzas  join order_details
on pizzas.pizza_id=order_details.pizza_id
join pizza_types on pizza_types.pizza_type_id=pizzas.pizza_type_id
group by pizza_types.category order by revenue_percent desc
```

OUTPUT

revenue_percent	category
26.91	Classic
25.46	Supreme
23.96	Chicken
23.68	Veggie

-- ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME

SQL QUERY

```sql
-- Analyze the cumulative revenue generated over time

select order_date, sum(revenue) over (order by order_date)
from
(select orders.order_date, sum(order_details.quantity*pizzas.price) as revenue
from orders
join order_details on orders.order_id=order_details.order_id
join pizzas on pizzas.pizza_id=order_details.pizza_id
group by order_date) sales
```

OUTPUT

order_date	sum(revenue) over (order by order_date)
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
2015-01-17	39001.75000000001
2015-01-18	40978.600000000006
2015-01-19	43365.75000000001

-- DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPE BASED ON REVENUE FOR EACH PIZZA CATEGORY

SQL QUERY

```sql
-- Determine the top 3 most ordered pizza type based on revenue for each pizza category

select * from
(select *,
dense_rank() over (partition by category order by revenue desc) as rnk
from
(select pizza_types.category, sum(order_details.quantity*pizzas.price) as revenue,pizza_types.name
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 rnk<4;
```

OUTPUT

category	revenue	name	rnk
Chicken	43434.25	The Thai Chicken Pizza	1
Chicken	42768	The Barbecue Chicken Pizza	2
Chicken	41409.5	The California Chicken Pizza	3
Classic	38180.5	The Classic Deluxe Pizza	1
Classic	32273.25	The Hawaiian Pizza	2
Classic	30161.75	The Pepperoni Pizza	3
Supreme	34831.25	The Spicy Italian Pizza	1
Supreme	33476.75	The Italian Supreme Pizza	2
Supreme	30940.5	The Sicilian Pizza	3
Veggie	32265.70000000065	The Four Cheese Pizza	1
Veggie	26780.75	The Mexicana Pizza	2
Veggie	26066.5	The Five Cheese Pizza	3

KEY INSIGHTS & LEARNINGS

Key Insights:

- Certain pizza sizes dominate sales (e.g., Small)
- Categories contribute differently to revenue
- Peak business hours identified
- Revenue distribution reveals profitability patterns

What I Learned:

- Writing multi-table joins
- Using window functions
- Business interpretation of SQL outputs

THANK YOU!