PIZZA SALES

MISSION :

Project Introduction – Pizza Sales Data Exploration

In this project, I analyzed a Pizza Sales dataset to gain insights into sales trends, customer preferences, and business performance.
The dataset consisted of multiple related tables, which I explored and analyzed using MySQL queries.
The dataset structure is as follows:

I performed various SQL queries to:

- Explore the dataset and understand relationships between tables.
- Analyze sales distribution by category, size, and price.
- Identify top-selling pizzas and revenue-generating products.
- Study time-based trends such as daily and monthly sales patterns.

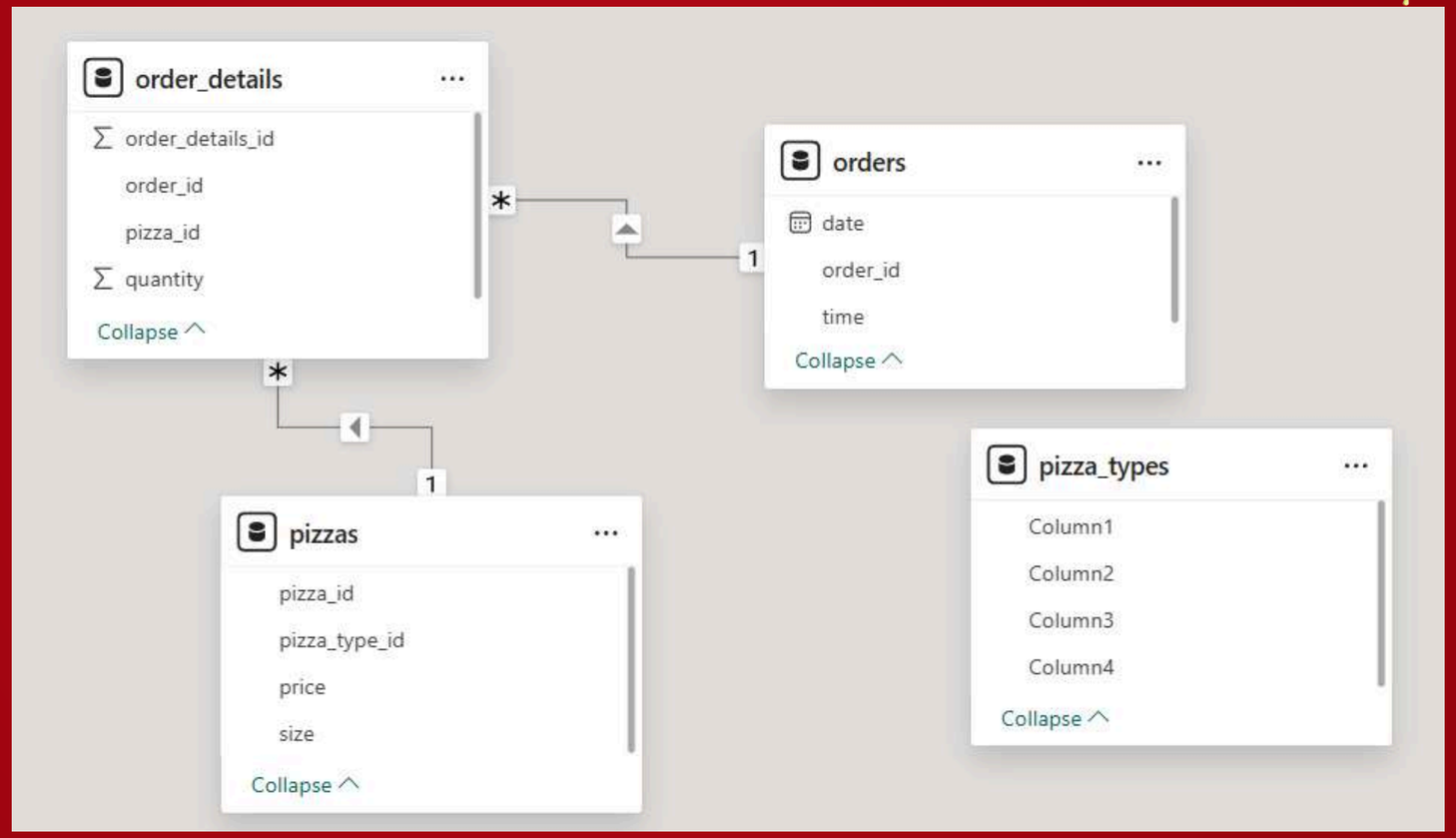
The purpose of this project was to practice database querying skills and derive meaningful insights from a real-world-like dataset for decision-making in a pizza business scenario.

DATASET INTRO

TABLES :

- order_details
  - Columns: order_details_id, order_id, pizza_id, quantity
  - Rows: 48,621
- pizzas
  - Columns: pizza_id, pizza_type_id, size, price
  - Rows: 97
- pizza_types
  - Columns: pizza_type_id, name, category, ingredients
  - Rows: 33
- orders
  - Columns: order_id, date, time
  - Rows: 21,350

THIS IS HOW OUR DATA MODEL LOOKS LIKE :



1. CREATION

```sql
create database dominos;
use dominos;
```

```sql
create table order_details(
order_details_id int not null,
order_id int not null,
pizza_id text not null,
quantity int not null,
primary key(order_details_id));
```

```sql
create table orders(
order_id int not null,
order_date date not null,
order_time time not null,
primary key(order_id));
```

```sql
select * from order_details;
select * from orders;
select * from pizza_types;
select * from pizzas;
```

2. BASIC QUERIES :

1. Retrieve the total number of orders placed.

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

Result :

Result Grid

	total_orders
▶	21350

2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES. MAKE USE OF JOIN.

```
round(sum(order_details.quantity * pizzas.price),2) as total_revenue
order_details join pizzas
zas.pizza_id = order_details.pizza_id;
```

RESULT :

Result Grid

	total_revenue
▶	35227.5

3. IDENTIFY THE HIGHEST PRIZE 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 1;
```

RESULTS :

Result Grid | Filter Rows

	name	price
▶	The Greek Pizza	35.95

4.IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

NOTE: WHEN U USE AGGREGATE FUNCTION MAKE COMPULSORILY USE OF GROUP BY FUNCTION .

```
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 2 ;
```

RESULTS :

Result Grid | Filter

size	order_count
L	813
M	676

5. 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;
```

RESULTS :

Result Grid | Filter Rows:

name	quantity
The Pepperoni Pizza	126
The Barbecue Chicken Pizza	100
The Classic Deluxe Pizza	99
The Thai Chicken Pizza	98
The Sicilian Pizza	96

3. INTERMEDIATE QUERIES :

1. 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;
```

Result :

Result Grid

category	quantity
Classic	624
Supreme	536
Veggie	510
Chicken	459

2. 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 order_count desc;
```

RESULTS :

Result Grid

hour	order_count
17	2336
19	2009
16	1920
20	1642
14	1472
15	1468
11	1231
21	1198
22	663
23	28
10	8
9	1

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

```
select category , count(name) as name_count from pizza_types
group by category;
```

RESULTS :

Result Grid | Filter Ro

category	name_count
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

```
select round( avg(quantity),2) as avg_pizza_ordered_per_day
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 ordered_quantity;
```

RESULT :

Result Grid | Filter Rows:

	avg_pizza_ordered_per_day
▶	133.06

5. 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 pizza_types.name
order by revenue
limit 3;
```

RESULTS :

Result Grid | Filter Rows:

name	revenue
The Brie Carre Pizza	307.45
The Mediterranean Pizza	543.75
The Calabrese Pizza	564.25

4. HARD QUERIES :

1. 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)*100,2) 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
order by revenue desc;
```

Result :

Result Grid

category	revenue
Supreme	26.2
Classic	26.16
Veggie	24.39
Chicken	23.24

2. 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(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 order_date) as sales ;
```

RESULT :

Result Grid | Filter Rows:

order_date	cum_revenue
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
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	35227.50000000001

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

```
(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 ;
```

RESULTS :

Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	1829.5
The Barbecue Chicken Pizza	1783
The California Chicken Pizza	1551.5
The Pepperoni Pizza	1591.5
The Classic Deluxe Pizza	1559
The Greek Pizza	1340
The Italian Supreme Pizza	1684
The Spicy Italian Pizza	1567.25
The Sicilian Pizza	1492
The Five Cheese Pizza	1350.5
The Four Cheese Pizza	1292.5...
The Mexicana Pizza	1198

SKILLS USED :

- DATA MODELING: DESIGNED RELATIONSHIPS BETWEEN MULTIPLE TABLES (ORDERS, ORDER_DETAILS, PIZZAS, PIZZA_TYPES) FOR EFFICIENT QUERYING.
- FULL DATASET ANALYSIS: EXPLORED 21,350 ORDERS AND 48,621 ORDER DETAILS TO EXTRACT VALUABLE TRENDS.
- COMPLEX JOINS: USED INNER JOIN, LEFT JOIN, AND MULTI-TABLE JOINS TO COMBINE AND ANALYZE DATA.
- ADVANCED SQL FUNCTIONS: APPLIED WINDOW FUNCTIONS (RANK, DENSE_RANK), CASE WHEN, CTES, SUBQUERIES, AND AGGREGATION FOR DEEP INSIGHTS.
- TIME-BASED ANALYSIS: ANALYZED DAILY, MONTHLY, AND HOURLY SALES TO IDENTIFY PEAK BUSINESS PERIODS.
- PERFORMANCE OPTIMIZATION: STRUCTURED QUERIES FOR FASTER EXECUTION AND ACCURATE RESULTS.

DATASET ACCESS LINK :

- https://github.com/NNDINI7/SQL-Data-Analysis-Projects

THANK YOU!