PIZZA

HUT

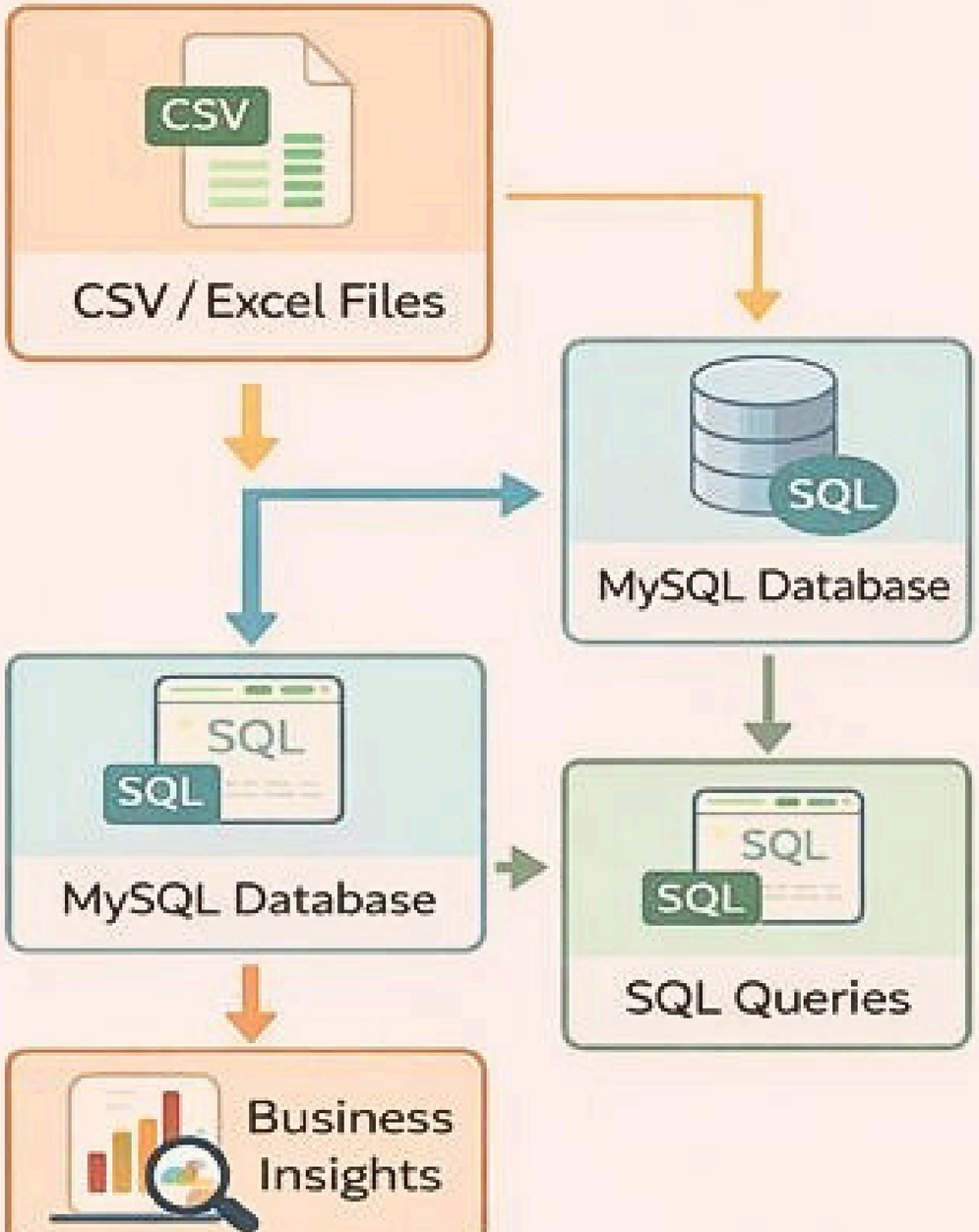
PROJECT

WELCOME OUR PIZZA HUT

description here

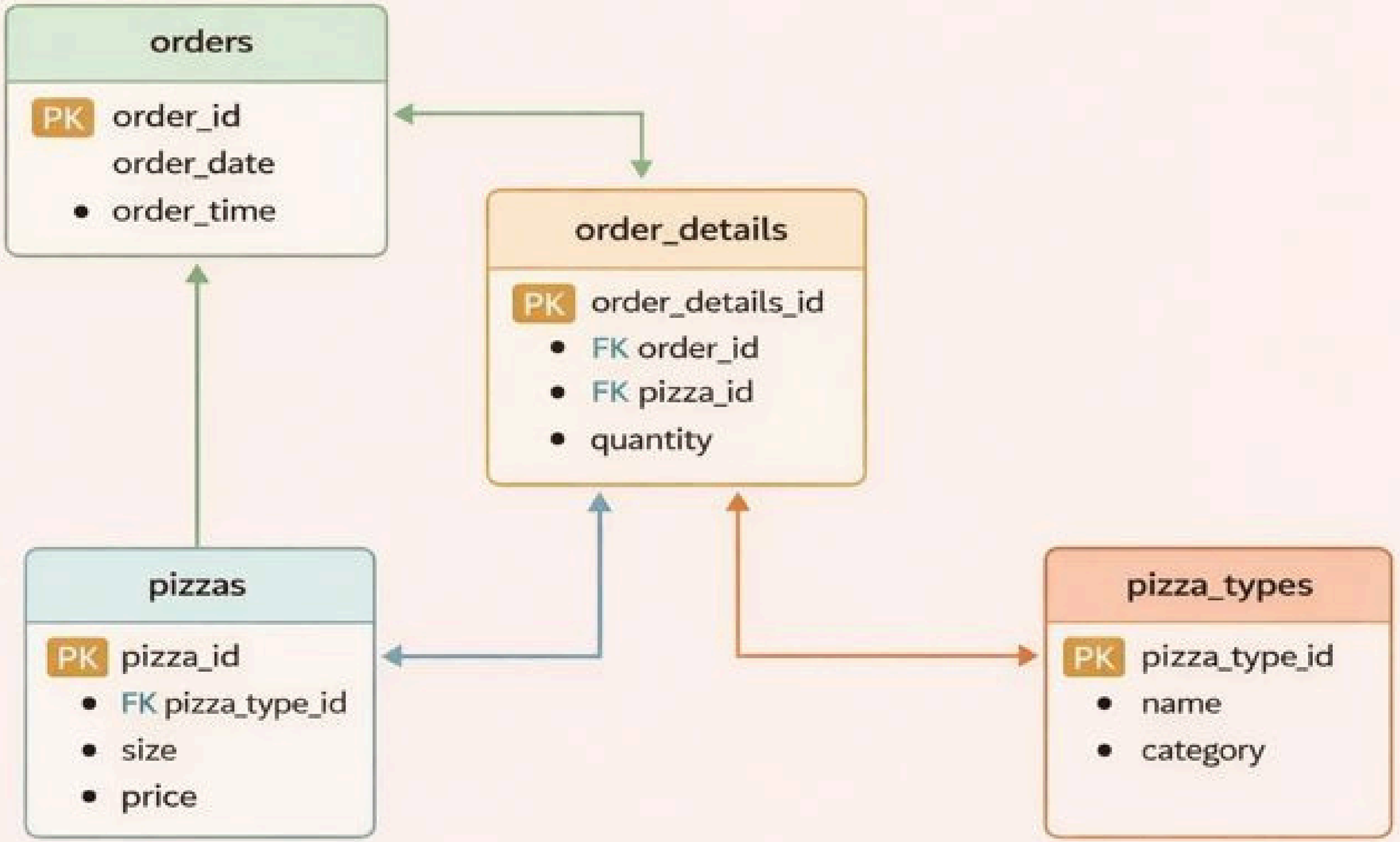
Hello everyone i am Robin Pandit and in this project i built a SQL-based Pizza Hut sales analysis project to evaluate orders, revenue, top-selling pizzas, and customer trends using joins, aggregations, and window functions.

Data Pipeline



- Raw sales data imported from CSV files
- Data cleaned and structured into relational tables
- SQL queries used for analysis and reporting

Data Source & Database Schema



Key Business Insights

- Total revenue calculated from pizza sales
- Identified top 5 best-selling pizzas
- Determined peak order hours during the day
- Analyzed category-wise revenue contribution
- Calculated cumulative revenue over time

Key Insights & SQL Queries

Total Revenue Generated ₹8.2M

Totat: ₹8.2M

SQL

```
SELECT SUM(od.quantity * p.price)
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id;
```

Top-Selling Pizza Pepperoni Pizza

Total Orders: (8,851 Orders)

SQL

```
SELECT pizza_id, COUNT(*) as order_count
    FROM order_details
    GROUP BY pizza_id
    ORDER BY order_count DESC
    LIMIT 1;
```

Peak Order Times 7 PM - 9 PM

Category-Wise Revenue Classic: 62%

Classic: 62%

SQL

```
SELECT pt.category, SUM(od.quantity * p.price)
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id =
    pt.pizza_type_id
    GROUP BY  pt.category
    ORDER BY revenue DESc;
```

Cumulative Revenue Growth Daily

Daily running total

running total

💡Business Logic & SQL Queries

- how much total revenue did we generate ?

```
SELECT ROUND(SUM(od.quantity * p.price), 2) AS
Total_revenue
FROM pizzas p
JOIN order_details od
ON p.pizza_id = od.pizza_id;
```

- During which hours do we receive the most orders ?

```
SELECT HOUR (order_time) as hour,
FROM order_details od as order_count
GROUP BY hour
ORDER BY order_count DESC
LIMIT 1;
```

- What is our top-selling most common pizza size ?

```
SELECT quantity, count(order_details_id)
AS total_order
FROM order_details
GROUP BY quantity;
```

💡Business Logic & SQL Queries

- **Join the necessary tables to find the total quantity of each pizza category ordered.**

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

```
SELECT
pt.category, SUM(od.quantity) AS quantity
FROM
pizza_types pt
JOIN
pizzas p ON pt.pizza_type_id = p.pizza_type_id
JOIN
order_details od ON p.pizza_id = od.pizza_id
GROUP BY category
ORDER BY 2 DESC;
```

- **Join relevant tables to find the category-wise distribution of pizzas.**

	category	total_pizza
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

```
SELECT
category, COUNT(name) AS total_pizza
FROM
pizza_types
GROUP BY category;
```

💡Business Logic & SQL Queries

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

	avg_pizza_ordered_per_day
▶	106

```
SELECT
ROUND(AVG(quantity),0) 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 order_quantity;
```

💡Business Logic & SQL Queries

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

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

```
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 pizzas.pizza_id = order_details.pizza_id
group by pizza_types.name
order by 2 desc
limit 3;
```

💡Business Logic & SQL Queries

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

category	revenue
Classic	26.91
Veggie	23.68
Supreme	25.46
Chicken	23.96

```
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 pizzas.pizza_id = order_details.pizza_id
group by pizza_types.category;
```

💡Business Logic & SQL Queries

- **Analyze the cumulative revenue generated over time.**

order_date	cum_revenue
2015-01-01 00:00:00	2713.8500000000004
2015-01-02 00:00:00	5445.75
2015-01-03 00:00:00	8108.15
2015-01-04 00:00:00	9863.6
2015-01-05 00:00:00	11904.800000000001
2015-01-06 00:00:00	14333.75
2015-01-07 00:00:00	16535.95
2015-01-08 00:00:00	19374.3
2015-01-09 00:00:00	21476.9
2015-01-10 00:00:00	23924.100000000002
2015-01-11 00:00:00	25796.4
2015-01-12 00:00:00	27715.45
2015-01-13 00:00:00	29752.55
2015-01-14 00:00:00	32279.949999999997
2015-01-15 00:00:00	34236

```
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 orders.order_date) as sales;
```

💡Business Logic & SQL Queries

- Determine the top 3 most ordered pizza types
- based on revenue for each pizza category

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

```
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 pizzas.pizza_id = order_details.pizza_id
group by pizza_types.category, pizza_types.name) as a) as b
where rn <= 3;
```


THANK YOU