Pizza Sales Analysis (SQL Project)

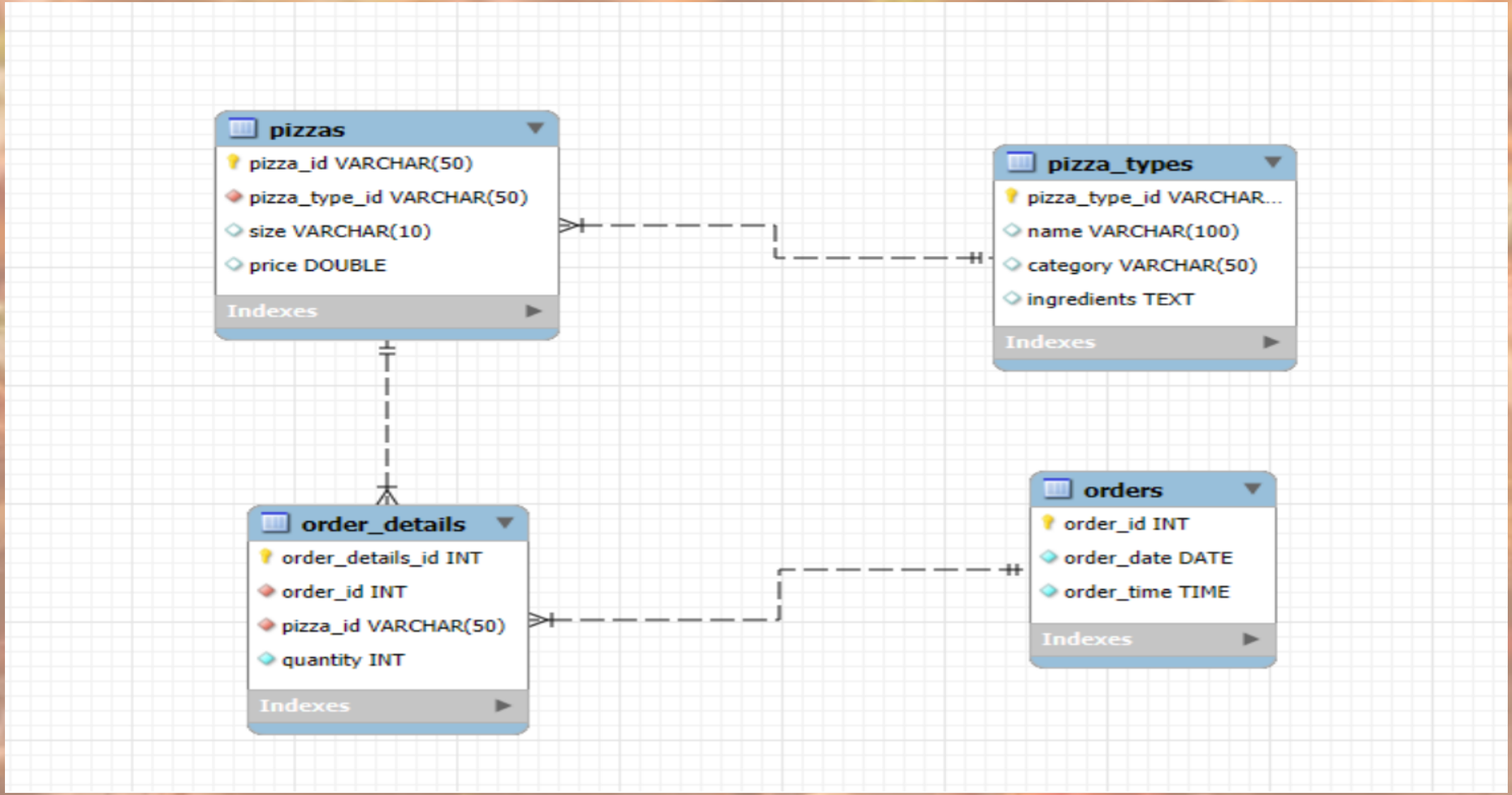
Exploring Sales Trends and Insights using SQL

EER Diagram • Schema • Queries • Insights

- **Language - SQL**
- **Database MySQL**

Kumari Neha | B. Tech, Lovely professional university

EER - Diagram


pizzas
pizza_id VARCHAR(50)
pizza_type_id VARCHAR(50)
size VARCHAR(10)
price DOUBLE
Indexes
pizza_types
pizza_type_id VARCHAR...
name VARCHAR(100)
category VARCHAR(50)
ingredients TEXT
Indexes
order_details
order_details_id INT
order_id INT
pizza_id VARCHAR(50)
quantity INT
Indexes
orders
order_id INT
order_date DATE
order_time TIME
Indexes

Database Schema Design

```sql
-- Create database
CREATE DATABASE pizzahut;
USE pizzahut;
-- Orders table
CREATE TABLE orders (
    order_id INT NOT NULL,
    order_date DATE NOT NULL,
    order_time TIME NOT NULL,
    PRIMARY KEY (order_id)
);
-- Order Details table
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
-- Pizza Types table
CREATE TABLE pizza_types (
    pizza_type_id VARCHAR(50) NOT NULL,
    name TEXT,
    category TEXT,
    ingredients TEXT,
    PRIMARY KEY (pizza_type_id)
);
-- Pizzas table
CREATE TABLE pizzas (
    pizza_id VARCHAR(50) NOT NULL,
    pizza_type_id VARCHAR(50) NOT NULL,
    size TEXT,
    price DOUBLE,
    PRIMARY KEY (pizza_id)
);
-- Add FK order_details → orders
ALTER TABLE order_details
ADD CONSTRAINT fk_order FOREIGN KEY (order_id)
REFERENCES orders(order_id);
```

```sql
-- Fix pizza_id consistency
ALTER TABLE pizzas
MODIFY pizza_id VARCHAR(50) NOT NULL;

ALTER TABLE order_details
MODIFY pizza_id VARCHAR(50) NOT NULL;

-- Improve readability of other columns
ALTER TABLE pizzas
MODIFY size VARCHAR(10);

ALTER TABLE pizza_types
MODIFY name VARCHAR(100),
MODIFY category VARCHAR(50);
```

```sql
-- pizza_id datatype in order_details to allow FK
ALTER TABLE order_details
MODIFY pizza_id VARCHAR(50) NOT NULL;

-- Add FK order_details → pizzas
ALTER TABLE order_details
ADD CONSTRAINT fk_pizza FOREIGN KEY (pizza_id)
REFERENCES pizzas(pizza_id);

-- Fix pizza_type_id datatype (ensure consistency)
ALTER TABLE pizza_types
MODIFY pizza_type_id VARCHAR(50) NOT NULL;

ALTER TABLE pizzas
MODIFY pizza_type_id VARCHAR(50) NOT NULL;

-- Add FK pizzas → pizza_types
ALTER TABLE pizzas
ADD CONSTRAINT fk_pizza_type FOREIGN KEY (pizza_type_id)
REFERENCES pizza_types(pizza_type_id);
```

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

SELECT
    SUM(order_details.quantity) AS quantity,
    COUNT(order_details.order_id) AS total_orders,
    pizza_types.category
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 quantity DESC;
```

Result Grid | Filter Rows:

quantity	total_orders	category
14888	14579	Classic
11987	11777	Supreme
11649	11449	Veggie
11050	10815	Chicken

```sql
-- Calculate the total revenue generated from pizza sales.
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;
```

Result Grid

total_sales
817860.05

```sql
-- Identify the highest-priced pizza.

SELECT
    pizzas.pizza_id, pizzas.price, pizza_types.name
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

Result Grid | Filter Rows:

pizza_id	price	name
the_greek_xxl	35.95	The Greek Pizza

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

SELECT
    pizzas.size, COUNT(order_details.order_id) AS total_sale
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY total_sale DESC
LIMIT 1;
```

Result Grid | Filt

size	total_sale
L	18526

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

SELECT
    COUNT(order_details.order_id) AS total_order,
    pizzas.pizza_type_id,
    SUM(order_details.quantity) AS total_quantity
FROM
    pizzas
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizzas.pizza_type_id
ORDER BY total_order DESC
LIMIT 5;
```

Result Grid | Filter Rows:

total_order	pizza_type_id	total_quantity
2416	classic_dlx	2453
2372	bbq_ckn	2432
2370	hawaiian	2422
2369	pepperoni	2418
2315	thai_ckn	2371

```sql
-- find the top 3 pizza sizes that generated the highest revenue
-- Solution: - L, M, S

SELECT
    pizzas.size,
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_revenue
FROM
    order_details
        JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizzas.size
ORDER BY total_revenue DESC
LIMIT 3;
```

Result Grid | Filter

size	total_revenue
L	375318.7
M	249382.25
S	178076.5

```sql
-- Determine the distribution of orders by hour of the day.

SELECT
    COUNT(orders.order_id) AS total_orders,
    HOUR(orders.order_time) AS hours
FROM
    orders
GROUP BY hours
ORDER BY total_orders DESC;
```

Result Grid

total_orders	hours
2520	12
2455	13
2399	18
2336	17
2009	19
1920	16
1642	20
1472	14
1468	15

```sql
-- Retrieve the total number of orders placed.

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

Result Grid

total_orders
21350

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

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

Result Grid | Filter R

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

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

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

Result Grid | Filter Ro

ROUND(AVG(quantity), 0)
138

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

select category,name, revenue from (
select category, name, revenue,
rank() over(partition by category order by revenue desc) as rn from
(select sum(pizzas.price * order_details.quantity) as revenue,
pizza_types.category, pizza_types.name
from pizza_types join pizzas on
pizzas.pizza_type_id = pizza_types.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;
```

Result Grid | Filter Rows:

category	name	revenue
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5
Classic	The Classic Deluxe Pizza	38180.5
Classic	The Hawaiian Pizza	32273.25
Classic	The Pepperoni Pizza	30161.75
Supreme	The Spicy Italian Pizza	34831.25
Supreme	The Italian Supreme Pizza	33476.75

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

select 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 total_revenue, pizza_types.category
from pizza_types
join pizzas
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 total_revenue desc;
```

Result Grid | Filter Rov

total_revenue	category
26.91	Classic
25.46	Supreme
23.96	Chicken
23.68	Veggie

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

select order_date, round(sum(revenue) over (order by order_date),2)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 Grid | Filter Rows

order_date	cum_revenue
2015-01-01	2713.85
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

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

Result Grid | Filter Rows:

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

Final Report

- Total revenue – 817860.05
- Most Common pizza size ordered – Large (18526 Orders)
- Highest revenue generating pizza size – Large (375318.70 rupees)
- Highest revenue generating order hours – 12 A.M
- Average number of orders per day – 138
- Highest revenue generating pizza type – Chicken pizza