

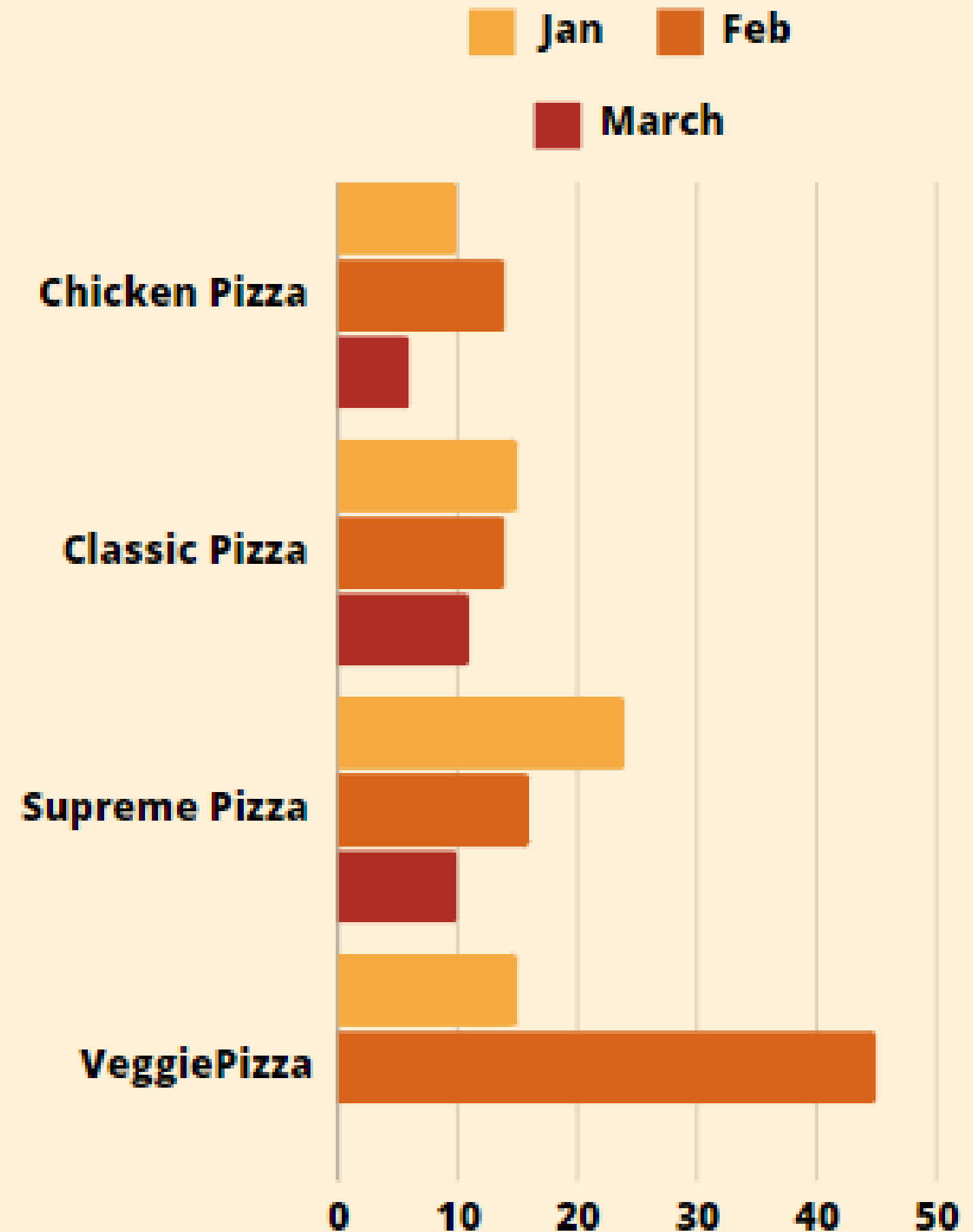
LICERIA  
& CO.

# PIZZA SALES

AN INDEPHTH ANALYSIS  
USING SQL

PRESENTED BY

PURAV ANAND





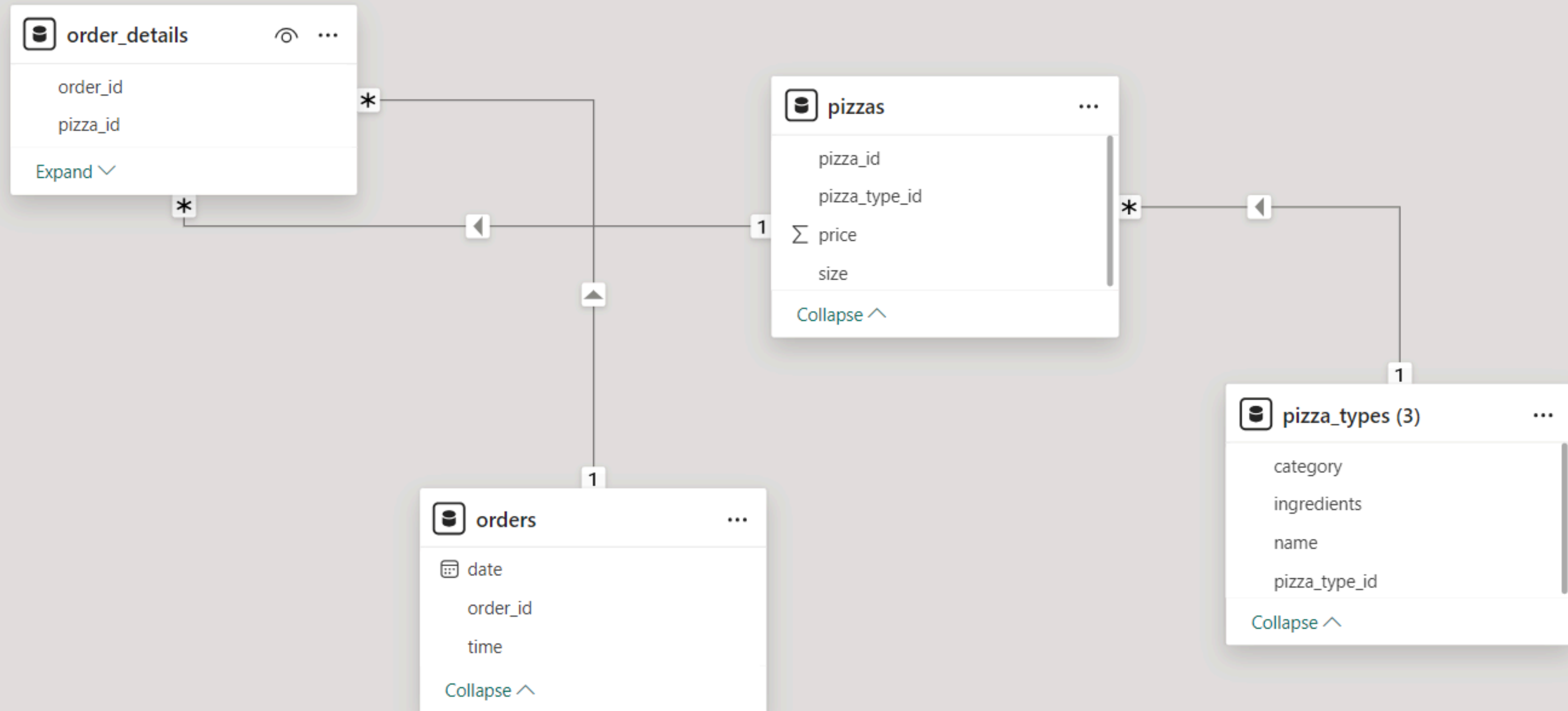
**PIZZA**

designed by  freepik.com

# INTRODUCTION

Welcome to the sales analysis project for PizzaHut. This project utilizes SQL to explore and analyze our sales data comprehensively. By examining sales trends, customer preferences, and product performance, this analysis aims to provide actionable insights that can drive strategic decision-making and enhance business outcomes..

# Schema Representation





The PizzaHut database contains crucial information structured into four primary tables. These tables collectively provide a comprehensive view of orders, pizzas, and their details, allowing for an in-depth analysis of sales and customer preferences.

# DATABASE OVERVIEW

## TABLE KEY POINTS

### 1. Orders

Description: This table records each order placed by customers, capturing essential details of the transaction.

#### Key Columns:

- order\_id: Unique identifier for each order (Primary Key)
- order\_date: The date and time when the order was placed
- customer\_id: Identifier for the customer who placed the order.

### 2. Order Details

Description: This table contains detailed information about the items included in each order, linking orders to specific pizzas.

#### Key Columns:

- order\_detail\_id: Unique identifier for each order detail entry (Primary Key)
- order\_id: Identifier linking to the Orders table (Foreign Key)
- pizza\_id: Identifier linking to the Pizzas table (Foreign Key)
- quantity: Number of each pizza type ordered



The PizzaHut database contains crucial information structured into four primary tables. These tables collectively provide a comprehensive view of orders, pizzas, and their details, allowing for an in-depth analysis of sales and customer preferences.

# DATABASE OVERVIEW

## TABLE KEY POINTS

- 

### 3.Pizzas

- Description: This table lists all the pizzas available at PizzaHut, including their prices and types.
- Key Columns:
  - pizza\_id: Unique identifier for each pizza (Primary Key)
  - pizza\_name: Name of the pizza
  - price: Price of the pizza
  - pizza\_type\_id: Identifier linking to the Pizza Types table (Foreign Key)
  -

### 4. Pizza Types

- Description: This table categorizes pizzas into different types, such as vegetarian, non-vegetarian, etc.
- Key Columns:
  - pizza\_type\_id: Unique identifier for each pizza type (Primary Key)
  - type\_name: Name of the pizza type category

# KEY ANALYTIC QUESTIONS

Que7. Determine the distribution of orders by hour of the day..

Que1.Retreive the total number of pizza orders..

Que3.Identified the highest price pizzas..

Que5. list the most ordered top 5 pizzas type along with their quantities..

Que2.Calculate the total no. of revenue generated..

Que4.Identify most common pizza size ordered..

Que6.Join neccessary table to find the total quantity of each pizza category ordered..

# KEY ANALYTIC QUESTIONS

Que8. find the category wise distribution of pizzas

Que11.calculate percentage contribution of each pizza type to total revenue.

Que9.Groups the order by date and calculate the average pizza ordered by per day.

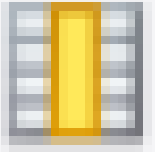


-- Que12..Analyze the commulative revenue generated by over time

Que10. Determine most top 3 ordered pizza types based on revenue..

-- Que13. Determine most ordered pizza type based on revenue for each pizza category

# 1. RETREIVE THE TOTAL NUMBER OF PIZZA ORDERS..

```
-- Que1.Retreive the total number of pizza orders..  
SELECT  
    COUNT(order_id)  
FROM  
    orders AS total_pizzas_ordered
```

Result Grid			
	COUNT(order_id)		
	21350		



## 2.CALCULATE THE TOTAL NO. OF REVENUE GENERATED..

```
SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),
          2) AS total_revenue
FROM
    pizzas
    JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
```

Result Grid | 

	total_revenue
▶	757009.55

### 3.IDENTIFIED THE HIGHEST PRICE PIZZAS..

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

Result Grid			Filter
	name	price	
▶	The Greek Pizza	35.95	

## 4.IDENTIFY MOST COMMON PIZZA SIZE ORDERED..

```
SELECT
  pizzas.size,
  COUNT(orders_details.order_details_id) AS order_details
FROM
  pizzas
  JOIN
  orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_details DESC
```

Result Grid		Filter Row
	size	order_details
▶	L	17137
	M	14262
	S	13061
	XL	508
	XXL	27

## 5. LIST THE MOST ORDERED TOP 5 PIZZAS TYPE ALONG WITH THEIR QUANTITIES..

```
SELECT
    pizza_types.name, SUM(orders_details.quantity) AS quantity
FROM
    pizza_types
    JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5
```

name	quantity
The Barbecue Chicken Pizza	2268
The Classic Deluxe Pizza	2264
The Pepperoni Pizza	2245
The Hawaiian Pizza	2229
The Thai Chicken Pizza	2180

## 6. JOIN NECESSARY TABLE TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED..

```
SELECT
    pizza_types.category, SUM(orders_details.quantity) AS quantity
FROM
    pizza_types
    JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5
```

	category	quantity
▶	Classic	13768
	Supreme	11111
	Veggie	10774
	Chicken	10223

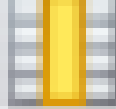

## 7. DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY...

```
SELECT
    HOUR(order_time) AS hours, COUNT(order_id) AS count
FROM
    orders
GROUP BY hours
ORDER BY hours
```

	hours	count
	9	1
	10	8
	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

## 8. FIND THE CATEGORY WISE DISTRIBUTION OF PIZZAS..

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

Result Grid     Filter Rows		
	category	COUNT(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

## 9. GROUPS THE ORDER BY DATE AND CALCULATE THE AVERAGE PIZZA ORDERED BY PER DAY.

```
SELECT
    ROUND(AVG(quantity), 0) AS AVG_PIZZA_ORDERED
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 1) AS order_quantity
```

AVG_PIZZA_ORDERED
139



## 10. DETERMINE MOST TOP 3 ORDERED PIZZA TYPES BASED ON REVENUE..

```
SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS each_pizza_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 1
ORDER BY 2 DESC
LIMIT 3
```

	name	each_pizza_revenue
▶	The Thai Chicken Pizza	39871
	The Barbecue Chicken Pizza	39869
	The California Chicken Pizza	38052

# 11. CALCULATE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE

```
SELECT
    pizza_types.category,
    ROUND(SUM(orders_details.quantity * pizzas.price) / (SELECT
        ROUND(SUM(orders_details.quantity * pizzas.price),
            2)
        FROM
            pizzas
            JOIN
            orders_details ON pizzas.pizza_id = orders_details.pizza_id) * 100,
        2)
FROM
    pizzas
    JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
    JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY 1
ORDER BY 2
```

▶	Veggie	23.68
	Chicken	23.94
	Supreme	25.5
	Classic	26.89

## 12. ANALYZE THE COMMULATIVE REVENUE GENERATED BY OVER TIME

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

order_date	cummulative_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
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

# 13. DETERMINE MOST ORDERED PIZZA TYPE BASED ON REVENUE FOR EACH PIZZA CATEGORY..

```
select category , revenue ,rn 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(orders_details.quantity * pizzas.price) as revenue
from pizzas join pizza_types on pizzas.pizza_type_id=pizza_types.pizza_type_id
join orders_details on orders_details.pizza_id=pizzas.pizza_id
group by 1 ,2) as a) as b
where rn<=3
```

	category	revenue	rn
	Chicken	39871	1
	Chicken	39869	2
	Chicken	38052	3
	Classic	35217.5	1
	Classic	29671	2
	Classic	28002	3
	Supreme	32267.5	1
	Supreme	31120.5	2
	Supreme	28486.75	3
	Veggie	29743.500000000057	1
	Veggie	24898.75	2
	Veggie	24586.5	3

# KEY FINDINGS

1. **Sales Trends**: Identified peak sales periods and overall growth patterns.
  - Insight: Sales peak during weekends, holidays, and from afternoon to evening.
2. **Top-Selling Products**: Determined the most popular pizzas.
  - Insight: Margherita and Pepperoni pizzas are the top sellers.
  -
3. **Sales by Pizza Type**: Evaluated performance of different pizza types.
  - Insight: Vegetarian pizzas are gaining popularity.
4. **Promotional Impact**: Assessed the effectiveness of various promotions.
  - Insight: Discount promotions significantly boost short-term sales.

## RECOMENDATIONS FOR INCREASING PIZZAHUT SALES

1. **Enhance Marketing Strategies**: Focus on promoting top-selling pizzas and leveraging peak sales periods.
2. **Diversify Menu**: Expand the vegetarian pizza options to cater to growing demand.
3. **Optimize Promotions**: Continue and refine discount promotions to maintain sales momentum during off-peak periods.
4. **Enhance Delivery Services**: Ensure fast and reliable delivery with real-time tracking.
5. **Customer Loyalty Programs**: Offer rewards and discounts for repeat purchases to encourage customer loyalty.

**NAME : PURAV ANAND**

**PHONE NO. : +91 9306242131**

**LINKED IN : [www.linkedin.com/in/purav-anand-0700431b6](https://www.linkedin.com/in/purav-anand-0700431b6)**

**EMAIL : puravanand.rewari@gmail.com**

**THANKYOU**