La’Pinoz Pizza Shop Analysis on SQL

--Creating database

```
CREATE DATABASE lapinoz_pizza;


lapinoz_pizza;
```

--Changing the datestyle

```
SHOW DATESTYLE;

SET DATESTYLE TO 'DMY';
```

-- 1. creating order table

```
CREATE TABLE Orders(
                                    order_id SERIAL PRIMARY KEY,
                                    order_date DATE,
                                    order_time TIME
);

COPY Orders(order_id, order_date, order_time)
FROM 'D:\PROGRAMMING\DATA SCIENCE\DATA ANALYTICS\SQL\Project\Pizza Sales\orders.csv'
CSV HEADER;

SELECT * FROM Orders;
```

-- 2. creating pizza type table

```
DROP TABLE IF EXISTS pizza_type;

CREATE TABLE pizza_type(
                                             pizza_types_id VARCHAR(25) PRIMARY KEY,
                                             pizza_name VARCHAR(50),
                                             category VARCHAR(15),
                                             ingredients TEXT
);
```

```
SELECT * FROM pizza_type;


COPY pizza_type(pizza_types_id, pizza_name, category, ingredients)

FROM 'D:\PROGRAMMING\DATA SCIENCE\DATA ANALYTICS\SQL\Project\Pizza
Sales\pizza_types2.csv'

CSV HEADER;
```

-- 3. creating pizza table

```
DROP TABLE IF EXISTS Pizzas;


CREATE TABLE Pizzas(

                    pizza_id VARCHAR(25) PRIMARY KEY,

                    pizza_types_id VARCHAR(25) REFERENCES pizza_type(pizza_types_id),

                    pizza_size VARCHAR(10),

                    price NUMERIC(10,2)

);


SELECT * FROM Pizzas;


COPY Pizzas(pizza_id, pizza_types_id, pizza_size, price)

FROM 'D:\PROGRAMMING\DATA SCIENCE\DATA ANALYTICS\SQL\Project\Pizza Sales\pizzas.csv'

CSV HEADER;
```

-- 4. creating order detail table

```
DROP TABLE IF EXISTS Orders_details;


CREATE TABLE Orders_details(

                order_details_id SERIAL PRIMARY KEY,

                order_id INT REFERENCES Orders(order_id),

                pizza_id VARCHAR(25) REFERENCES Pizzas(pizza_id),

                quantity INT

);


SELECT * FROM Orders_details;
```

COPY Orders_details(order_details_id, order_id, pizza_id, quantity)

FROM 'D:\PROGRAMMING\DATA SCIENCE\DATA ANALYTICS\SQL\Project\Pizza Sales\order_details.csv'

CSV HEADER;

-- Get data

SELECT * FROM Orders_details;

SELECT * FROM Orders;

SELECT * FROM Pizzas;

SELECT * FROM pizza_type;

-- Basic:

-- 1. Retrieve the total number of orders placed.

SELECT COUNT(order_id) AS total_orders

FROM Orders;

-- 2. Calculate the total revenue generated from pizza sales.

SELECT SUM(p.price*od.quantity) AS total_revenue

FROM Pizzas p

JOIN Orders_details od

ON p.pizza_id = od.pizza_id;

-- 3. Identify the highest-priced pizza.

SELECT *

FROM Pizzas

ORDER BY price DESC

LIMIT 1;

-- 4. Identify the most common pizza size ordered.

SELECT pizza_size, COUNT(pizza_size) AS frequnecy

FROM Pizzas p

JOIN Orders_details od

ON p.pizza_id = od.pizza_id

GROUP BY p.pizza_size

ORDER BY frequnecy DESC

LIMIT 1;

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

SELECT pizza_id, SUM(quantity) AS total_quanity

FROM Orders_details

GROUP BY pizza_id

ORDER BY total_quantity DESC;

-- Intermediate:

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

SELECT pt.category, SUM(od.quantity) AS total_quantity_orders

FROM pizzas p

JOIN orders_details od

ON p.pizza_id = od.pizza_id

JOIN pizza_type pt

ON pt.pizza_types_id = p.pizza_types_id

GROUP BY pt.category;

-- 2. Determine the distribution of orders by hour of the day.

SELECT EXTRACT('Hour' FROM o.order_time) AS hour, COUNT(o.order_id) AS distribution

FROM Orders o

JOIN orders_details od

ON o.order_id = od.order_id

GROUP BY hour

ORDER BY hour;

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

SELECT category, COUNT(pizza_types_id) AS distribution

FROM pizza_type

GROUP BY category

```
ORDER BY distribution DESC;
```

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

```
SELECT o.order_date, SUM(od.quantity) AS sum_ordered_quantity, AVG(od.quantity) AS avg_ordered_quantity

FROM Orders o

JOIN orders_details od

ON o.order_id = od.order_id

GROUP BY o.order_date

ORDER BY o.order_date;
```

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

```
SELECT p.pizza_types_id, SUM(p.price * od.quantity) AS revenue

FROM pizzas p

JOIN orders_details od

ON p.pizza_id = od.pizza_id

GROUP BY p.pizza_types_id

ORDER BY revenue DESC

LIMIT 3;
```

-- Advanced:

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

```
SELECT

    pt.pizza_name,

    sub.revenue AS pizza_type_revenue,

    (sub.revenue / total_revenue.total_rev) * 100 AS percentage_contribution

FROM

    pizza_type pt

JOIN

    (SELECT

        p.pizza_types_id,

        SUM(p.price * od.quantity) AS revenue

    FROM
```

```
        pizzas p
    JOIN
        Orders_details od ON p.pizza_id = od.pizza_id
    GROUP BY
        p.pizza_types_id) AS sub
        ON pt.pizza_types_id = sub.pizza_types_id
CROSS JOIN
    (SELECT SUM(p.price * od.quantity) AS total_rev
     FROM pizzas p
     JOIN Orders_details od ON p.pizza_id = od.pizza_id) AS total_revenue;
```

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

```
-- Daily commulative
SELECT
          o.order_date,
          SUM(p.price*od.quantity) AS revenue,
          EXTRACT('Month' FROM o.order_date) AS month_no,
          SUM(SUM(p.price*od.quantity)) OVER (ORDER BY o.order_date) AS commulative_revenue
FROM pizzas p
JOIN Orders_details od
ON p.pizza_id = od.pizza_id
JOIN Orders o
ON o.order_id = od.order_id
GROUP BY o.order_date
ORDER BY month_no;


-- Monthly commulative
SELECT
    DATE_TRUNC('month', o.order_date) AS month_start,
    SUM(p.price * od.quantity) AS monthly_revenue,
    SUM(SUM(p.price * od.quantity)) OVER (ORDER BY DATE_TRUNC('month', o.order_date)) AS
cumulative_monthly_revenue
FROM
    pizzas p
```

```
JOIN
    Orders_details od ON p.pizza_id = od.pizza_id
JOIN
    Orders o ON o.order_id = od.order_id
GROUP BY
    DATE_TRUNC('month', o.order_date)
ORDER BY
    DATE_TRUNC('month', o.order_date);
```

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

```
SELECT * FROM Orders_details;
SELECT * FROM Orders;
SELECT * FROM Pizzas;
SELECT * FROM pizza_type;


WITH PizzaRevenueByCategory AS (
    SELECT
        pt.category,
        pt.pizza_name,
        SUM(p.price * od.quantity) AS revenue,
        ROW_NUMBER() OVER (PARTITION BY pt.category ORDER BY SUM(p.price * od.quantity) DESC) AS
rank_within_category
    FROM
        pizzas p
    JOIN
        Orders_details od
                    ON p.pizza_id = od.pizza_id
    JOIN
        pizza_type pt
                    ON pt.pizza_types_id = p.pizza_types_id
    GROUP BY
        pt.category,
        pt.pizza_name
)
SELECT
    category,
    pizza_name,
    revenue
FROM
    PizzaRevenueByCategory
WHERE
    rank_within_category <= 3
ORDER BY
    category,
    revenue DESC;


--combine
SELECT DISTINCT pt.category, SUM(p.price*od.quantity) AS revenue, (SELECT DISTINCT p.pizza_id
FROM pizzas p
JOIN Orders_details od
ON p.pizza_id = od.pizza_id
JOIN pizza_type pt
ON pt.pizza_types_id = p.pizza_types_id
ORDER BY p.price*od.quantity DESC
LIMIT 3)
FROM pizzas p
JOIN Orders_details od
ON p.pizza_id = od.pizza_id
JOIN pizza_type pt
ON pt.pizza_types_id = p.pizza_types_id
GROUP BY pt.category;
```

-- Created by VIVEK SAHU

-- File Link:- "https://github.com/VIVEKSAHU-06/Data-Analysis/tree/main/SQL/Pizza%20Sales"