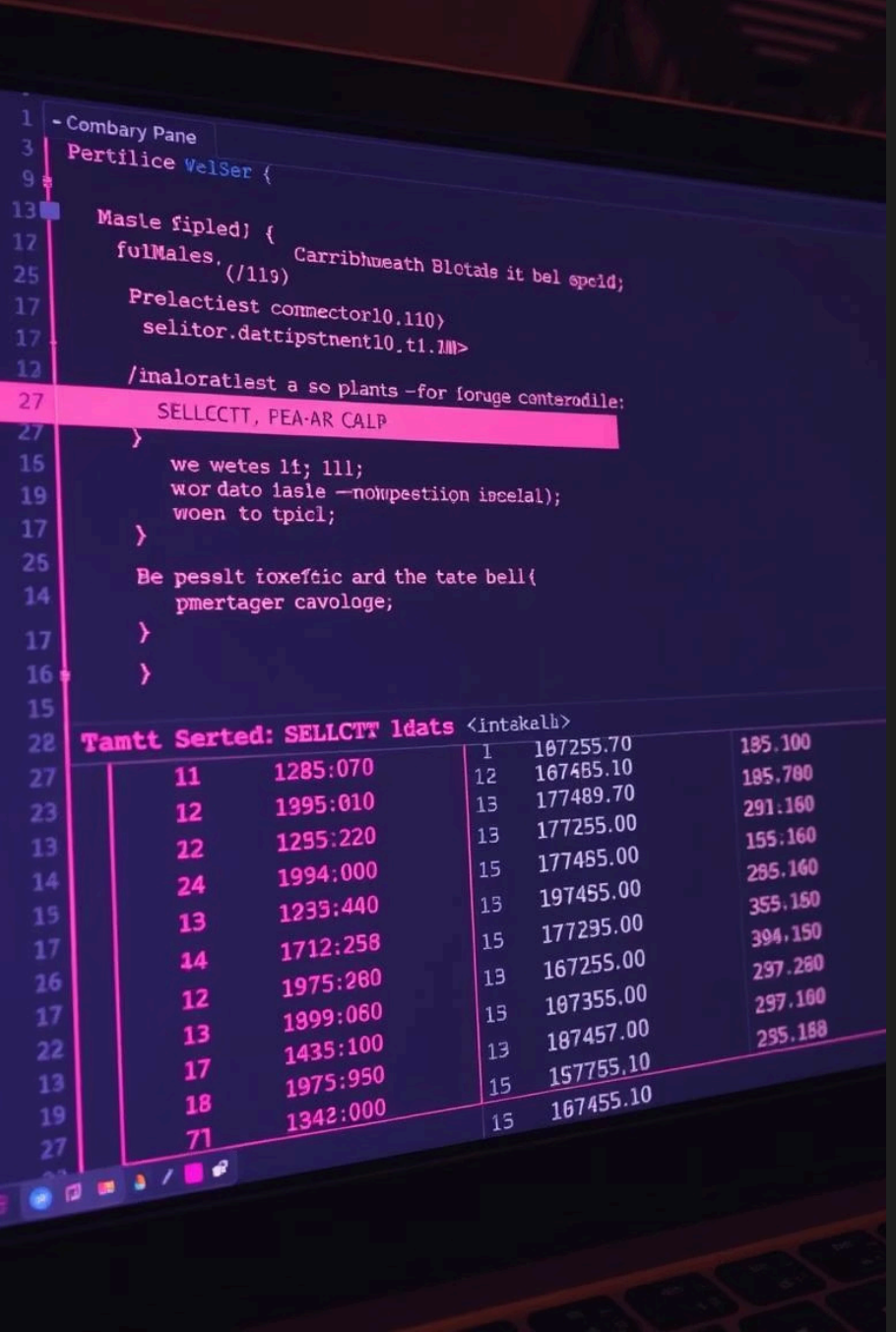
Optimizing Pizza Hut Sales with SQL

Discover how SQL analysis unlocks valuable insights from Pizza Hut sales data. Drive smarter business decisions using efficient data extraction and trend visualization.

by Harshika Nirwan



Essential SQL Queries: Data Extraction

1 Retrieve Data

Basic SELECT to get records from tables.

2 Filter Results

Use WHERE for targeted data (e.g., sales in Q1 2015).

3 Sort Data

ORDER BY for top-selling pizzas or dates.

4 Example Query

SELECT * FROM orders WHERE order_date LIKE '2015-01%';

Retrieve the total number of orders placed.

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

Result Grid

total_orders
21350

Calculate the total revenue generated from pizza sales.

SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 11

Limit to 50000 rows

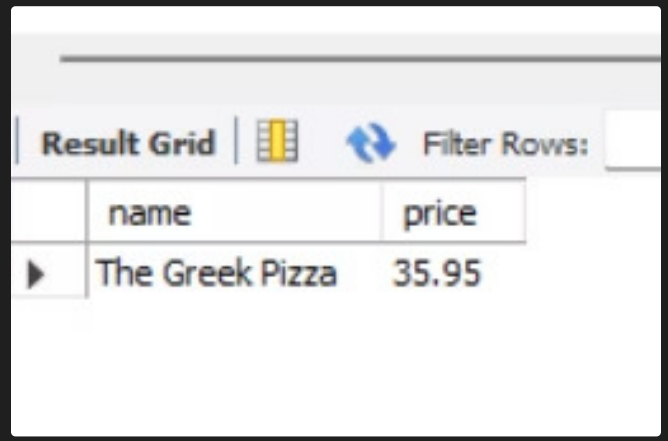
```sql
SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS total_sales
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id

```

Result Grid Filter Rov

total_sales
817860.05

Identify the highest-priced pizza.

```sql
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;
```



Identify the most common pizza size ordered.

```
2
3 •  SELECT
4        pizzas.size,
5        COUNT(orders_details.order_delails_id) AS order_count
6    FROM
7        pizzas
8            JOIN
9        orders_details ON pizzas.pizza_id = orders_details.pizza_id
0    GROUP BY pizzas.size
1    ORDER BY order_count;
2
```

Result Grid | Filter Rows:

size	order_count
XXL	28
XL	544
S	14137
M	15385
L	18526

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

```sql
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 orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;
```

Result Grid | Filter Rows:

name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

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

```sql
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 orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```

Result Grid | Filter Rov

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Determine the distribution of orders by hour of the day.

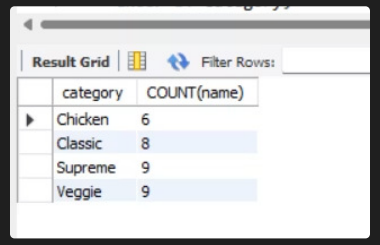
```sql
SELECT
    HOUR(oder_time) AS hour, COUNT(order_id) AS orders
FROM
    orders
GROUP BY HOUR(oder_time);
```

Result Grid | Filter Ro

hour	orders
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
10	8
9	1

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

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

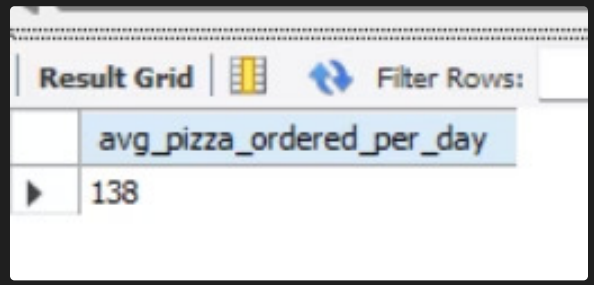


Result Grid | Filter Rows:

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

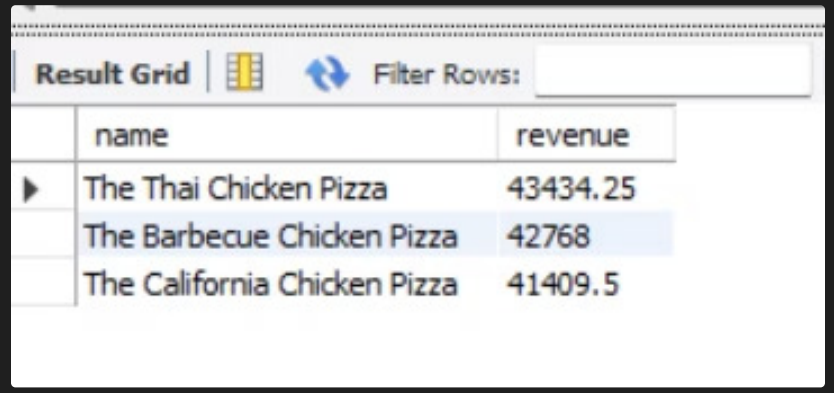
Group the orders by date and calculate the average number of pizzas ordered per day.

```sql
SELECT
    ROUND(AVG(quantity), 0) as avg_pizza_ordered_per_day
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 orders.order_date) AS orders_quantity;
```



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

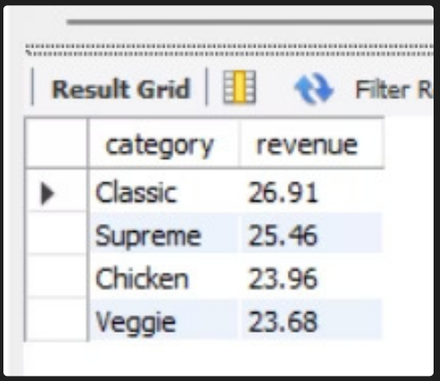
```sql
SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```



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

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

```sql
SELECT
    pizza_types.category,
    round(SUM(orders_details.quantity * pizzas.price) / ( select   ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS total_sales
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id)* 100,2) as 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 pizza_types.category
ORDER BY revenue DESC;
```


Result Grid | Filter R

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Analyze the cumulative revenue generated over time.

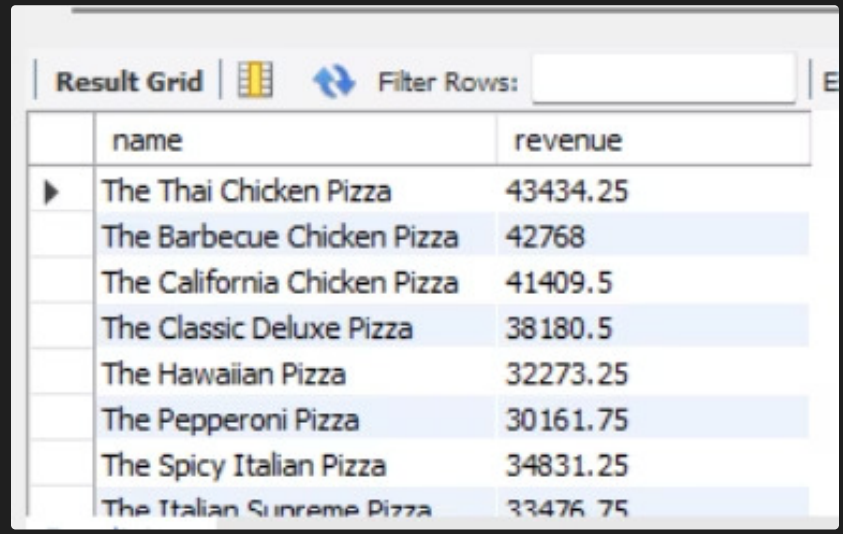
```sql
select order_date,
sum(revenue) over(order by order_date)as cum_revenue
from
(select
orders.order_date,
sum(orders_details.quantity*pizzas.price) as revenue
from orders_details join pizzas
on orders_details.pizza_id= pizzas.pizza_id
join orders
on orders.order_id = orders_details.order_delails_id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

Result Grid | Filter Rows:

order_date	cum_revenue
2015-01-01	1171.45
2015-01-02	2316.1000000000004
2015-01-03	3433.8
2015-01-04	4341.8
2015-01-05	5247.25
2015-01-06	6299.9
2015-01-07	7284.7
2015-01-08	8542.35

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

```sql
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(orders_details.quantity*pizzas.price) as 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  pizza_types.category, pizza_types.name) as A) as B
where RN <=3;
```



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

Advanced SQL Queries: Aggregation

1

GROUP BY

Aggregate sales by pizza type or category.

2

Aggregate Functions

- SUM: total sales
- AVG: average order value
- COUNT: number of orders

3

Example

SELECT pizza_type_id, SUM(quantity) AS total_quantity FROM order_details GROUP BY pizza_type_id ORDER BY total_quantity DESC;

SQL for Sales Trend Analysis

Trend Analysis

Use date functions to analyze sales over time.

Peak Identification

Find top sales days and months.

Moving Averages

Smooth sales fluctuations using averages.

Example Query

```
SELECT strftime('%Y-%m', order_date) AS month, SUM(price) AS monthly_revenue FROM orders JOIN order_details ON orders.order_id = order_details.order_id JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id GROUP BY month ORDER BY month;
```

Business Implications and Recommendations

Inventory Optimization

Stock more ingredients for Classic and Chicken pizzas.

Targeted Promotions

Run offers during peak months and evening hours.

Boost Online Experience

Enhance digital ordering for growing demand.

Adjust Staffing

Increase team during busy periods for efficiency.

Marketing Focus

Promote heavily during slow sales months.

Conclusion: Data-Driven Decision Making

SQL analysis unlocks key insights for sales optimization. Keep monitoring data regularly to adapt fast. Data-driven choices drive revenue and business growth.