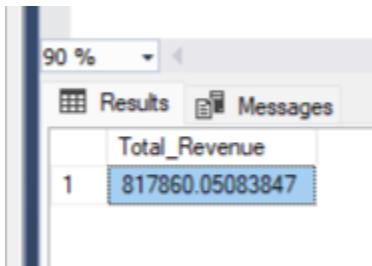


Pizza Sales SQL Queries

A. KPI's

1. Total Revenue:

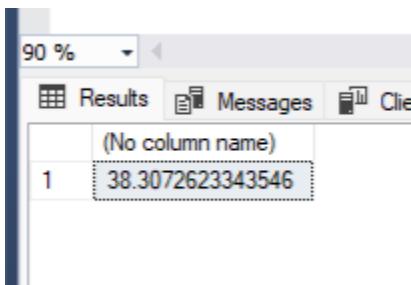
```
SELECT SUM(total_price) AS Total_Revenue FROM pizza_sales;
```



| Total_Revenue |
|-----------------|
| 817860.05083847 |

2. Average Order Value

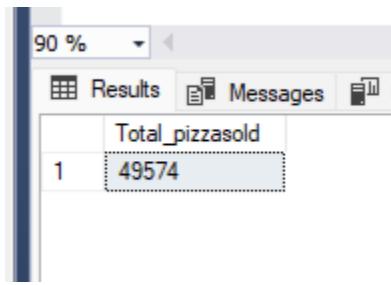
```
SELECT (SUM(total_price) / COUNT(DISTINCT order_id)) AS Avg_order_Value  
FROM pizza_sales
```



| (No column name) |
|------------------|
| 38.3072623343546 |

3. Total Pizzas Sold:

```
SELECT SUM(quantity) AS Total_pizza_sold FROM pizza_sales;
```



| Total_pizzasold |
|-----------------|
| 49574 |

4. Total Orders:

```
SELECT COUNT(DISTINCT order_id) AS Total_Orders FROM pizza_sales;
```

A screenshot of the SQL Server Management Studio (SSMS) interface. The window title is '90 %'. At the top, there are tabs for 'Results' (which is selected), 'Messages', and 'Client Status'. The results grid contains one row with the header 'Total_orders' and a value of '21350'.

| Total_orders |
|--------------|
| 21350 |

5. Average Pizzas Per Order:

```
SELECT CAST(CAST(SUM(quantity) AS DECIMAL(10,2)) /
CAST(COUNT(DISTINCT order_id) AS DECIMAL(10,2)) AS DECIMAL(10,2))
AS Avg_Pizzas_per_order
FROM pizza_sales;
```

A screenshot of the SQL Server Management Studio (SSMS) interface. The window title is '90 %'. At the top, there are tabs for 'Results' (selected), 'Messages', and 'Client Status'. The results grid contains one row with the header 'Avg_pizzas_perorder' and a value of '2.32'.

| Avg_pizzas_perorder |
|---------------------|
| 2.32 |

B.1. Daily Trend for Total Orders:

```
SELECT DATENAME(dw, order_date) AS order_day, COUNT(DISTINCT order_id) AS
total_orders
FROM pizza_sales
GROUP BY DATENAME(dw, order_date);
```

A screenshot of the SQL Server Management Studio (SSMS) interface. The window title is '90 %'. At the top, there are tabs for 'Results' (selected), 'Messages', and 'Client Status'. The results grid shows the total number of orders for each day of the week, ordered by day.

| | order_day | Total_orders |
|---|-----------|--------------|
| 1 | Saturday | 3158 |
| 2 | Wednesday | 3024 |
| 3 | Monday | 2794 |
| 4 | Sunday | 2624 |
| 5 | Friday | 3538 |
| 6 | Thursday | 3239 |
| 7 | Tuesday | 2973 |

2. Hourly Trend for Orders:

```
SELECT DATEPART(HOUR, order_time) as order_hours, COUNT(DISTINCT order_id)
```

```

as total_orders
FROM pizza_sales
GROUP BY DATEPART(HOUR, order_time)
ORDER BY DATEPART(HOUR, order_time);

```

The screenshot shows a SQL query results window with three tabs: Results, Messages, and Client. The Results tab displays a table with two columns: 'order_hours' and 'Total_orders'. The data is as follows:

| | order_hours | Total_orders |
|----|-------------|--------------|
| 1 | 9 | 1 |
| 2 | 10 | 8 |
| 3 | 11 | 1231 |
| 4 | 12 | 2520 |
| 5 | 13 | 2455 |
| 6 | 14 | 1472 |
| 7 | 15 | 1468 |
| 8 | 16 | 1920 |
| 9 | 17 | 2336 |
| 10 | 18 | 2399 |
| 11 | 19 | 2009 |

3.% of Sales by Pizza Category:

```

SELECT pizza_category, CAST(SUM(total_price) AS DECIMAL(10,2)) as
total_revenue,
CAST(SUM(total_price) * 100 / (SELECT SUM(total_price) from pizza_sales) AS
DECIMAL(10,2)) AS PCT
FROM pizza_sales
GROUP BY pizza_category;

```

The screenshot shows a SQL query results window with three tabs: Results, Messages, and Client. The Results tab displays a table with three columns: 'pizza_category', 'total_revenue', and 'PCT'. The data is as follows:

| | pizza_category | total_revenue | PCT |
|---|----------------|---------------|-------|
| 1 | Classic | 220053.10 | 26.91 |
| 2 | Chicken | 195919.50 | 23.96 |
| 3 | Veggie | 193690.45 | 23.68 |
| 4 | Supreme | 208197.00 | 25.46 |

4.% of Sales by Pizza Size:

```

SELECT pizza_size, CAST(SUM(total_price) AS DECIMAL(10,2)) as total_revenue,
CAST(SUM(total_price) * 100 / (SELECT SUM(total_price) from pizza_sales) AS
DECIMAL(10,2)) AS PCT

```

```
FROM pizza_sales  
GROUP BY pizza_size  
ORDER BY pizza_size;
```

| | pizza_size | PCT |
|---|------------|-------|
| 1 | L | 46.37 |
| 2 | M | 29.78 |
| 3 | S | 22.10 |
| 4 | XL | 1.60 |
| 5 | XXL | 0.14 |

5. Total Pizzas Sold by Pizza Category:

```
SELECT pizza_category, SUM(quantity) as Total_Quantity_Sold  
FROM pizza_sales  
WHERE MONTH(order_date) = 2  
GROUP BY pizza_category  
ORDER BY Total_Quantity_Sold DESC;
```

| | pizza_category | Total_pizzas_sold |
|---|----------------|-------------------|
| 1 | Classic | 14888 |
| 2 | Chicken | 11050 |
| 3 | Veggie | 11649 |
| 4 | Supreme | 11987 |

6. Top 5 Best Sellers by Total Pizzas Sold:

```
SELECT TOP 5 pizza_name, SUM(quantity) AS Total_Pizza_Sold  
FROM pizza_sales  
GROUP BY pizza_name  
ORDER BY Total_Pizza_Sold DESC;
```

The screenshot shows a SQL query results window with a title bar containing '100 %' and tabs for 'Results' and 'Messages'. The 'Results' tab is selected, displaying a table with two columns: 'pizza_name' and 'Total_Pizza_Sold'. The data is as follows:

| | pizza_name | Total_Pizza_Sold |
|---|----------------------------|------------------|
| 1 | The Classic Deluxe Pizza | 2453 |
| 2 | The Barbecue Chicken Pizza | 2432 |
| 3 | The Hawaiian Pizza | 2422 |
| 4 | The Pepperoni Pizza | 2418 |
| 5 | The Thai Chicken Pizza | 2371 |

7. Bottom 5 Best Sellers by Total Pizzas Sold:

```
SELECT TOP 5 pizza_name, SUM(quantity) AS Total_Pizza_Sold
FROM pizza_sales
GROUP BY pizza_name
ORDER BY Total_Pizza_Sold ASC;
```

The screenshot shows a SQL query results window with tabs for 'Results', 'Messages', and 'Client Statistics'. The 'Results' tab is selected, displaying a table with two columns: 'pizza_name' and 'Total_pizzas_sold'. The data is as follows:

| | pizza_name | Total_pizzas_sold |
|---|---------------------------|-------------------|
| 1 | The Brie Carré Pizza | 490 |
| 2 | The Mediterranean Pizza | 934 |
| 3 | The Calabrese Pizza | 937 |
| 4 | The Spinach Supreme Pizza | 950 |
| 5 | The Soppressata Pizza | 961 |

NOTE:

If you want to apply the Month, Quarter, Week filters to the above queries you can use WHERE clause.

Example 1:

```
SELECT DATENAME(DW, order_date) AS order_day, COUNT(DISTINCT order_id) AS
total_orders
FROM pizza_sales
WHERE MONTH(order_date) = 1
GROUP BY DATENAME(DW, order_date);
```

*Here MONTH(order_date) = 1 indicates that the output is for the month of January.

MONTH(order_date) = 4 indicates output for Month of April.

Example 2:

```
SELECT DATENAME(DW, order_date) AS order_day, COUNT(DISTINCT order_id) AS  
total_orders  
FROM pizza_sales  
WHERE DATEPART(QUARTER, order_date) = 1  
GROUP BY DATENAME(DW, order_date);
```

*Here DATEPART(QUARTER, order_date) = 1 indicates that the output is for the Quarter 1. DATEPART(QUARTER, order_date) = 3 indicates output for Quarter 3.