Blinkit Grocery Data SQL Project

Introduction

This project was undertaken to analyze grocery sales data from Blinkit using SQL. The dataset consisted of a single table named `Grocery` with the following columns:

- **Item_Fat_Content**
- **Item_Identifier**
- **Item_Type**
- **Outlet_Establishment_Year**
- **Outlet_Identifier**
- **Outlet_Location_Type**
- **Outlet_Size**
- **Outlet_Type**
- **Item_Visibility**
- **Item_Weight**
- **Total_Sales**
- **Rating**

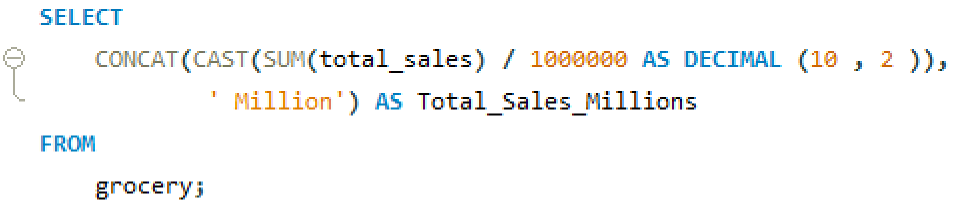
The purpose of this project was to extract valuable insights from the grocery data using SQL queries. The findings from these queries provide a better understanding of sales distribution, customer ratings, and performance across different categories.

Key Questions and Insights

1. Total Sales: The overall revenue generated from all items sold.

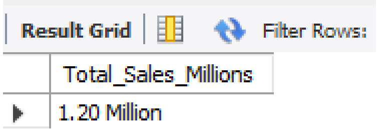
Agenda: This query helps in understanding the total revenue generated across all grocery items, giving a broad view of business performance.

Query:



```
SELECT
    CONCAT(CAST(SUM(total_sales) / 1000000 AS DECIMAL (10 , 2 )),
            ' Million') AS Total_Sales_Millions
FROM
    grocery;
```

Result:

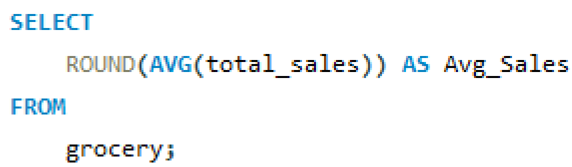


Total_Sales_Millions
1.20 Million

2. Average Sale: The average revenue per sale.

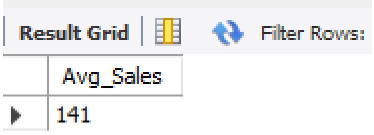
Agenda: This query helps in understanding the average amount earned per sale, which can be useful for pricing strategies.

Query:



```
SELECT
    ROUND(AVG(total_sales)) AS Avg_Sales
FROM
    grocery;
```

Result:

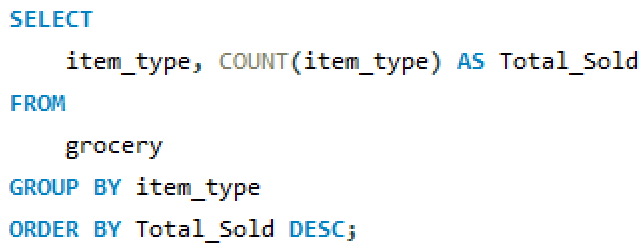


	Avg_Sales
▶	141

3. Number of Items: The total count of different items sold.

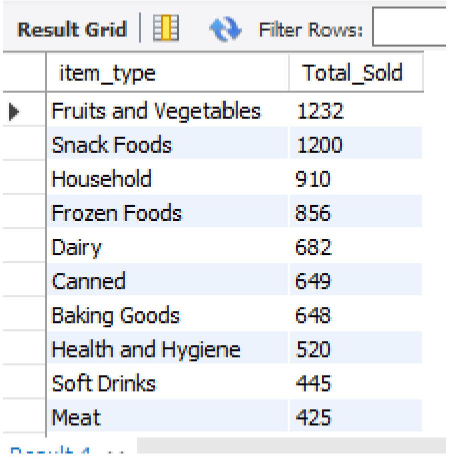
Agenda: This query provides insight into the variety of products sold and their overall distribution.

Query:



```
SELECT
    item_type, COUNT(item_type) AS Total_Sold
FROM
    grocery
GROUP BY item_type
ORDER BY Total_Sold DESC;
```

Result:

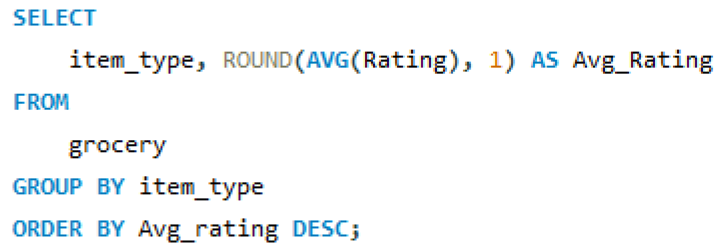


	item_type	Total_Sold
▶	Fruits and Vegetables	1232
	Snack Foods	1200
	Household	910
	Frozen Foods	856
	Dairy	682
	Canned	649
	Baking Goods	648
	Health and Hygiene	520
	Soft Drinks	445
	Meat	425

4. Average Rating: The average customer rating for items sold.

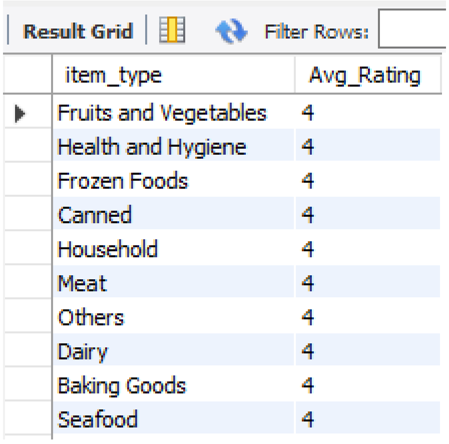
Agenda: This helps in assessing customer satisfaction and identifying well-received products.

Query:



```
SELECT
    item_type, ROUND(AVG(Rating), 1) AS Avg_Rating
FROM
    grocery
GROUP BY item_type
ORDER BY Avg_rating DESC;
```

Result:

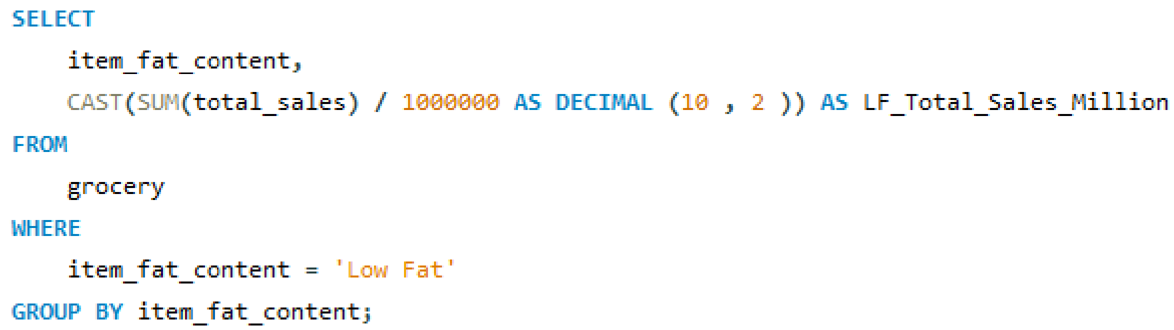


item_type	Avg_Rating
Fruits and Vegetables	4
Health and Hygiene	4
Frozen Foods	4
Canned	4
Household	4
Meat	4
Others	4
Dairy	4
Baking Goods	4
Seafood	4

5. Total Sales for "Low Fat" Category.

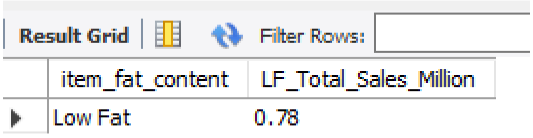
Agenda: This query provides insights into how well "Low Fat" products are performing in terms of sales.

Query:



```
SELECT
    item_fat_content,
    CAST(SUM(total_sales) / 1000000 AS DECIMAL (10 , 2 )) AS LF_Total_Sales_Million
FROM
    grocery
WHERE
    item_fat_content = 'Low Fat'
GROUP BY item_fat_content;
```

Result:

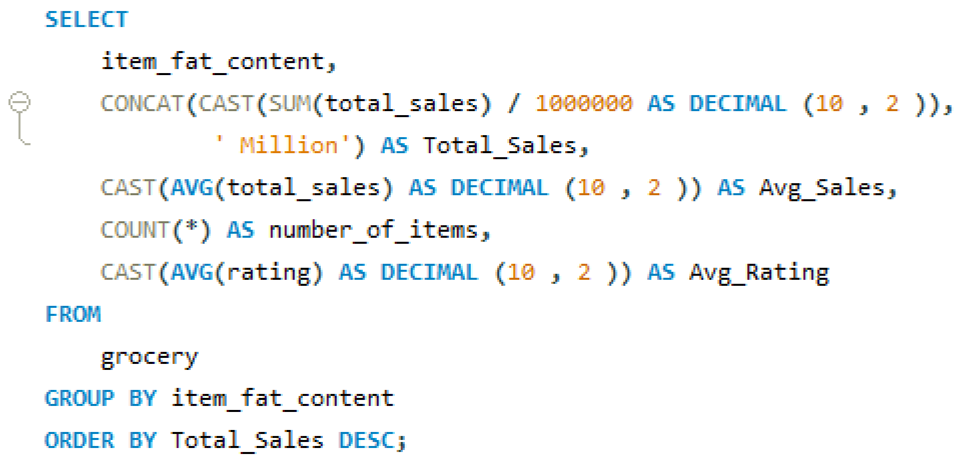


item_fat_content	LF_Total_Sales_Million
Low Fat	0.78

6. Total Sales, Average Sales, Total Number of Items, and Average Ratings for Each Category.

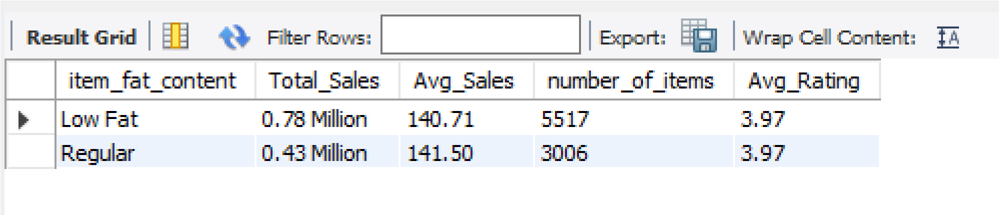
Agenda: This query helps analyze the performance of each product category, allowing businesses to identify top-selling categories.

Query:



```
SELECT
    item_fat_content,
    CONCAT(CAST(SUM(total_sales) / 1000000 AS DECIMAL (10 , 2 )),
            ' Million') AS Total_Sales,
    CAST(AVG(total_sales) AS DECIMAL (10 , 2 )) AS Avg_Sales,
    COUNT(*) AS number_of_items,
    CAST(AVG(rating) AS DECIMAL (10 , 2 )) AS Avg_Rating
FROM
    grocery
GROUP BY item_fat_content
ORDER BY Total_Sales DESC;
```

Result:



item_fat_content	Total_Sales	Avg_Sales	number_of_items	Avg_Rating
Low Fat	0.78 Million	140.71	5517	3.97
Regular	0.43 Million	141.50	3006	3.97

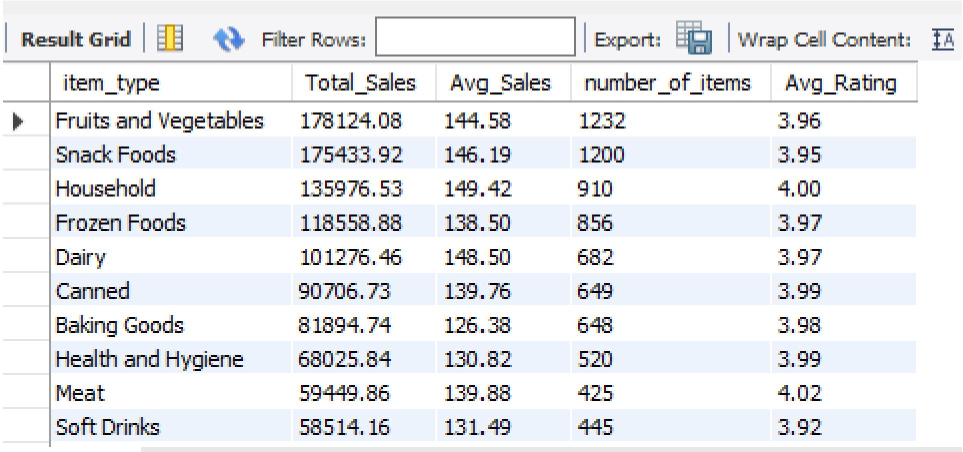
7. Total Sales, Average Sales, Total Number of Items, and Average Ratings for Each Item Type.

Agenda: This breakdown enables businesses to evaluate which specific item types contribute most to revenue and customer satisfaction.

Query:

```
SELECT
    item_type,
    CAST(SUM(total_sales) AS DECIMAL (10 , 2 )) AS Total_Sales,
    CAST(AVG(total_sales) AS DECIMAL (10 , 2 )) AS Avg_Sales,
    COUNT(*) AS number_of_items,
    CAST(AVG(rating) AS DECIMAL (10 , 2 )) AS Avg_Rating
FROM
    grocery
GROUP BY item_type
ORDER BY Total_Sales DESC;
```

Result:

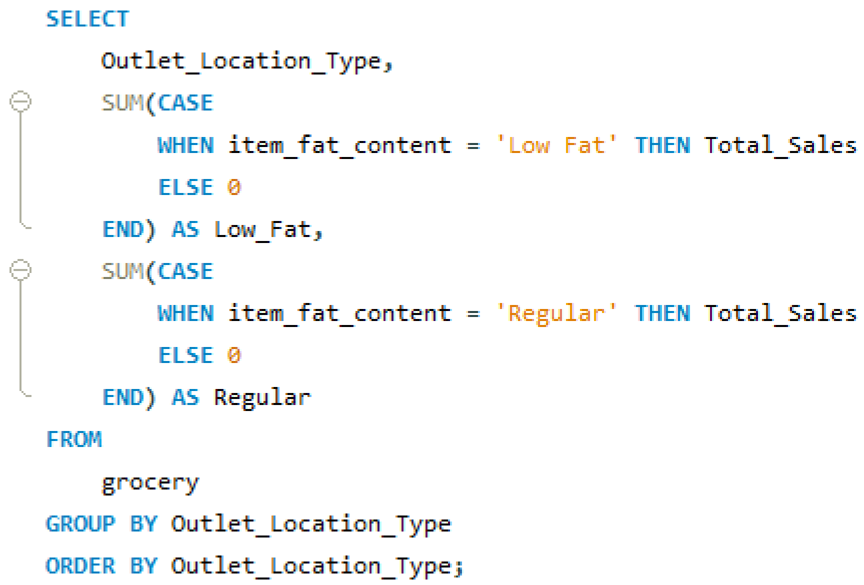


item_type	Total_Sales	Avg_Sales	number_of_items	Avg_Rating
Fruits and Vegetables	178124.08	144.58	1232	3.96
Snack Foods	175433.92	146.19	1200	3.95
Household	135976.53	149.42	910	4.00
Frozen Foods	118558.88	138.50	856	3.97
Dairy	101276.46	148.50	682	3.97
Canned	90706.73	139.76	649	3.99
Baking Goods	81894.74	126.38	648	3.98
Health and Hygiene	68025.84	130.82	520	3.99
Meat	59449.86	139.88	425	4.02
Soft Drinks	58514.16	131.49	445	3.92

8. Fat Content by Outlet for Total Sales, Average Sales, Number of Items, and Average Ratings.

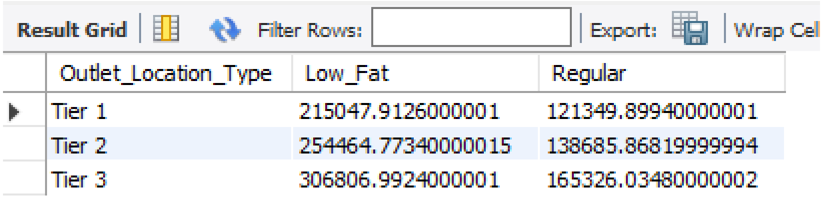
Agenda: This query helps analyze if an outlet's sales performance is influenced by the type of fat content in products.

Query:



```
SELECT
    Outlet_Location_Type,
    SUM(CASE
        WHEN item_fat_content = 'Low Fat' THEN Total_Sales
        ELSE 0
    END) AS Low_Fat,
    SUM(CASE
        WHEN item_fat_content = 'Regular' THEN Total_Sales
        ELSE 0
    END) AS Regular
FROM
    grocery
GROUP BY Outlet_Location_Type
ORDER BY Outlet_Location_Type;
```

Result:

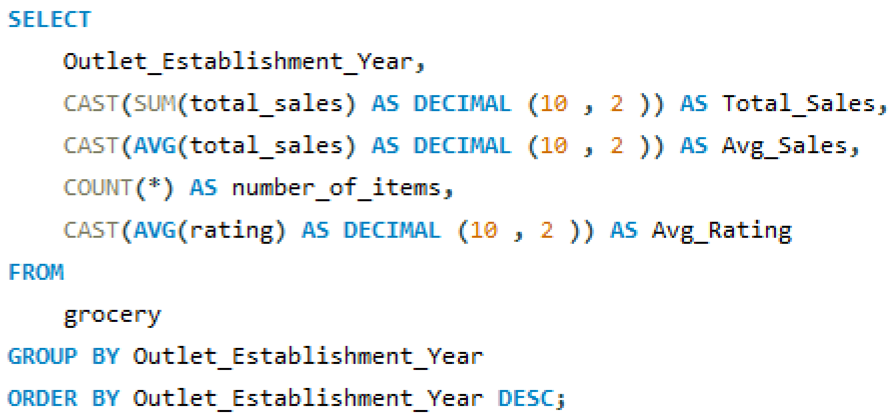


Outlet_Location_Type	Low_Fat	Regular
Tier 1	215047.9126000001	121349.89940000001
Tier 2	254464.77340000015	138685.86819999994
Tier 3	306806.9924000001	165326.03480000002

9. Total Sales, Average Sales, Number of Items, and Average Ratings by Outlet Establishment Year.

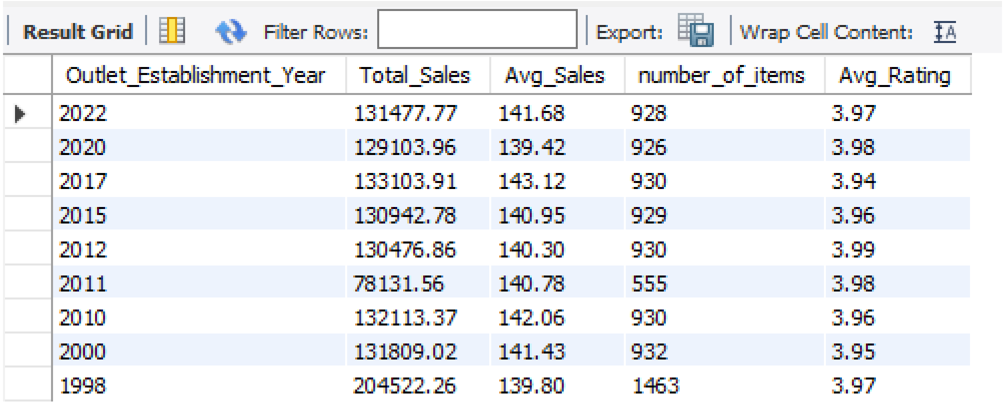
Agenda: This query allows us to analyze how the age of an outlet impacts its sales and customer ratings.

Query:



```sql
SELECT
    Outlet_Establishment_Year,
    CAST(SUM(total_sales) AS DECIMAL (10 , 2 )) AS Total_Sales,
    CAST(AVG(total_sales) AS DECIMAL (10 , 2 )) AS Avg_Sales,
    COUNT(*) AS number_of_items,
    CAST(AVG(rating) AS DECIMAL (10 , 2 )) AS Avg_Rating
FROM
    grocery
GROUP BY Outlet_Establishment_Year
ORDER BY Outlet_Establishment_Year DESC;
```

Result:

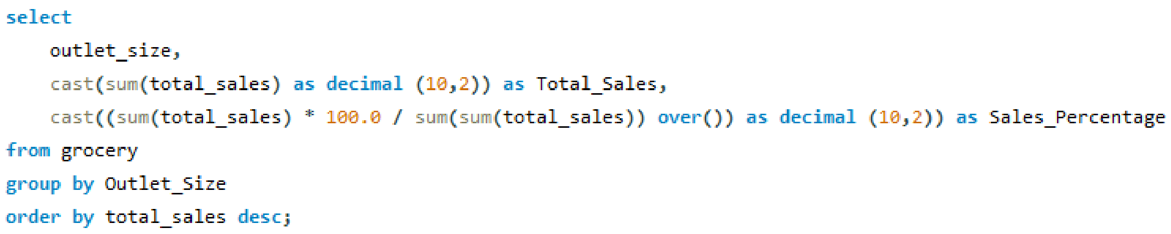


Outlet_Establishment_Year	Total_Sales	Avg_Sales	number_of_items	Avg_Rating
2022	131477.77	141.68	928	3.97
2020	129103.96	139.42	926	3.98
2017	133103.91	143.12	930	3.94
2015	130942.78	140.95	929	3.96
2012	130476.86	140.30	930	3.99
2011	78131.56	140.78	555	3.98
2010	132113.37	142.06	930	3.96
2000	131809.02	141.43	932	3.95
1998	204522.26	139.80	1463	3.97

10. Percentage of Sales by Outlet Size (Correlation Between Outlet Size and Total Sales).

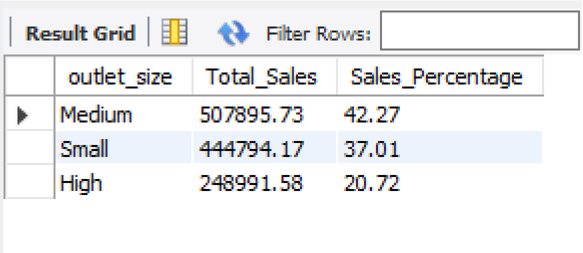
Agenda: This helps assess whether larger outlets generate higher sales and whether size impacts revenue generation.

Query:



```sql
select
    outlet_size,
    cast(sum(total_sales) as decimal (10,2)) as Total_Sales,
    cast((sum(total_sales) * 100.0 / sum(sum(total_sales)) over()) as decimal (10,2)) as Sales_Percentage
from grocery
group by Outlet_Size
order by total_sales desc;
```

Result:

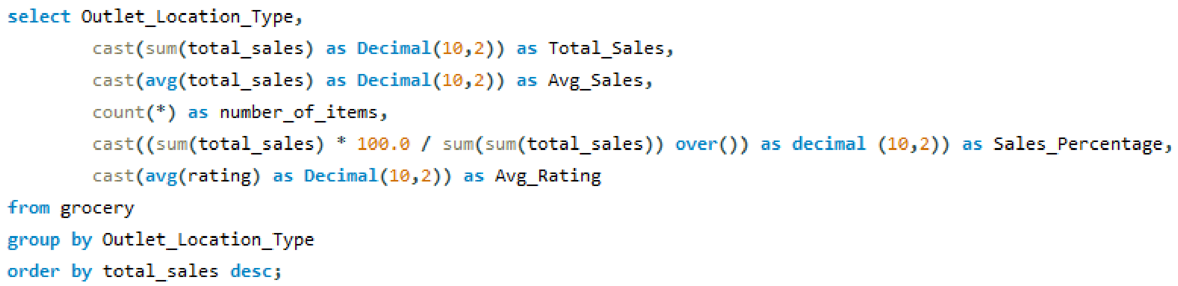


	outlet_size	Total_Sales	Sales_Percentage
▶	Medium	507895.73	42.27
	Small	444794.17	37.01
	High	248991.58	20.72

11. Sales by Outlet Location (Geographic Distribution of Sales Across Different Locations).

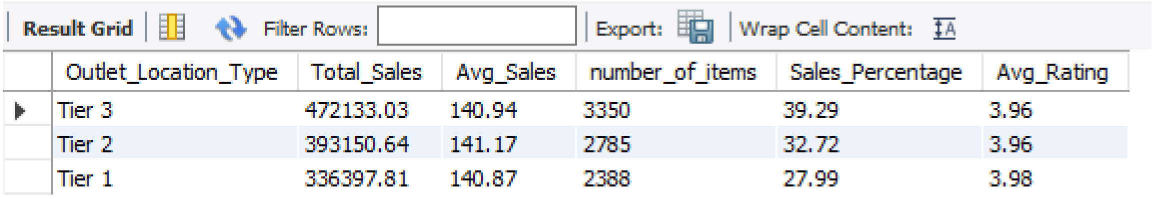
Agenda: This analysis provides insights into which locations perform best and can help optimize future outlet expansion.

Query:



```
select Outlet_Location_Type,
        cast(sum(total_sales) as Decimal(10,2)) as Total_Sales,
        cast(avg(total_sales) as Decimal(10,2)) as Avg_Sales,
        count(*) as number_of_items,
        cast((sum(total_sales) * 100.0 / sum(sum(total_sales)) over()) as decimal (10,2)) as Sales_Percentage,
        cast(avg(rating) as Decimal(10,2)) as Avg_Rating
from grocery
group by Outlet_Location_Type
order by total_sales desc;
```

Result:

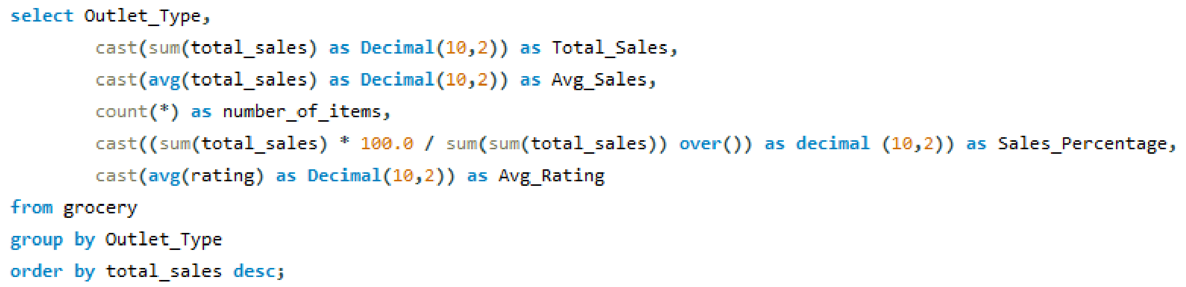


	Outlet_Location_Type	Total_Sales	Avg_Sales	number_of_items	Sales_Percentage	Avg_Rating
▶	Tier 3	472133.03	140.94	3350	39.29	3.96
	Tier 2	393150.64	141.17	2785	32.72	3.96
	Tier 1	336397.81	140.87	2388	27.99	3.98

12. All Metrics by Outlet Type (Comprehensive Breakdown of Total Sales, Average Sales, Number of Items, and Average Rating).

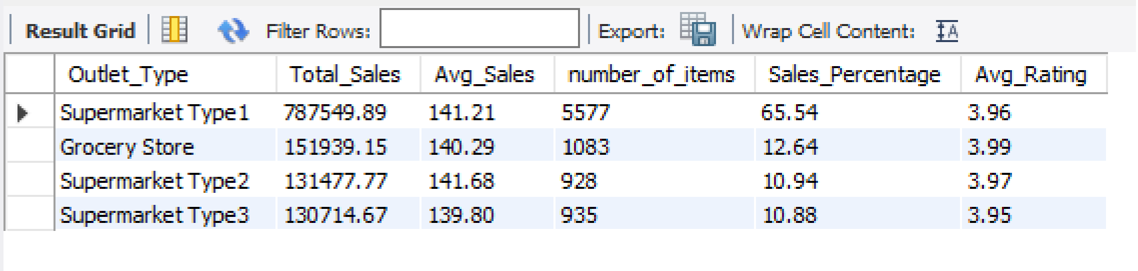
Agenda: This query helps compare different outlet types (e.g., supermarkets, grocery stores) to determine which are most profitable and well-rated.

Query:



```sql
select Outlet_Type,
        cast(sum(total_sales) as Decimal(10,2)) as Total_Sales,
        cast(avg(total_sales) as Decimal(10,2)) as Avg_Sales,
        count(*) as number_of_items,
        cast((sum(total_sales) * 100.0 / sum(sum(total_sales)) over()) as decimal (10,2)) as Sales_Percentage,
        cast(avg(rating) as Decimal(10,2)) as Avg_Rating
from grocery
group by Outlet_Type
order by total_sales desc;
```

Result:



Outlet_Type	Total_Sales	Avg_Sales	number_of_items	Sales_Percentage	Avg_Rating
Supermarket Type1	787549.89	141.21	5577	65.54	3.96
Grocery Store	151939.15	140.29	1083	12.64	3.99
Supermarket Type2	131477.77	141.68	928	10.94	3.97
Supermarket Type3	130714.67	139.80	935	10.88	3.95

Conclusion

This project provided valuable insights into the grocery sales data by leveraging SQL queries. We were able to analyze total sales, average ratings, sales distribution across different categories, and the impact of outlet size and location on sales. These insights can help businesses optimize their inventory, pricing strategies, and outlet performance.