# Blinkit Grocery Sales Analysis

SQL Portfolio Project
August 2025

Swetha K swethakogilaswaran@gmail.com

## **Blinkit Grocery Data Analysis**

This project analyzes Blinkit grocery sales data using **SQL** in SSMS. It includes data cleaning, aggregation, and insights into sales, ratings, outlet performance, and item visibility. The queries demonstrate practical SQL skills for real-world business analysis.

Below is a clear explanation for each SQL query from your project.

#### 1). Select all the data

```
SELECT * FROM blinkit data;
```

Retrieves every column and row from the table blinkit data for exploration.

## **Data Cleaning**

#### 2).Clean inconsistent labels in Item Fat Content

Cleaning the Item\_Fat\_Content field ensures consistency and accuracy in the data. Variations like "LF", "low fat", and "Low Fat" can lead to misleading results during analysis. Standardizing these values improves data quality, simplifies filtering and grouping, and ensures reliable insights.

```
UPDATE blinkit_data

SET item_Fat_Content =

CASE

WHEN item_Fat_Content IN ('LF','Low Fat') THEN 'Low Fat'

WHEN item_Fat_Content = 'reg' THEN 'Regular'

ELSE item_Fat_Content

END;
```

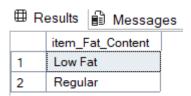
Standardizes inconsistent labels in the item\_Fat\_Content column for accurate grouping and analysis.

#### 3). Check whether the data has been cleaned

SELECT DISTINCT (item\_Fat\_Content)

FROM blinkit data;

Lists unique fat content types to confirm successful standardization.



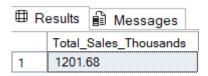
## **KPIs**

#### 4). Total Sales

SELECT CAST (SUM(Total\_Sales)/1000 AS DECIMAL (10,2) ) AS Total\_Sales\_Thousands

FROM blinkit\_data;

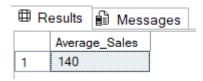
Shows total sales converted into thousands for readability.



## 5). Average Sales

SELECT CAST (AVG(Total\_Sales) AS INT) Average\_Sales FROM blinkit\_data;

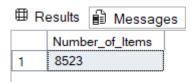
Calculates the average sales value across all records.



## 6). Number of items

```
SELECT COUNT(*) AS Number_of_Items
FROM blinkit_data;
```

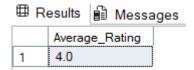
Returns the total number of records/items in the dataset.



### 7). Average Rating

```
SELECT CAST(AVG(Rating) AS DECIMAL(10,1))AS Average_Rating FROM blinkit_data;
```

Provides the average item rating with one decimal precision.



## **Sales Insights**

## 8). Total Sales by Fat Content

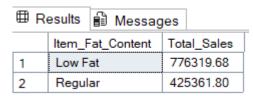
```
SELECT Item_Fat_Content,

CAST (SUM(Total_Sales)AS DECIMAL (10,2) )AS Total_Sales
FROM blinkit_data

GROUP BY Item_Fat_Content

ORDER BY Total Sales DESC;
```

Breaks down total sales by different fat content categories.



### 9). Total Sales by Item Type

```
SELECT Item_Type,

CAST(SUM(Total_Sales) AS DECIMAL (10,2) ) AS Total_Sales

FROM blinkit_data

GROUP BY Item_Type

ORDER BY Total_Sales DESC;
```

Displays how each item type contributes to total sales.

⊞ R	esults 🖺 Messages	
	Item_Type	Total_Sales
1	Fruits and Vegetables	178124.08
2	Snack Foods	175433.92
3	Household	135976.53
4	Frozen Foods	118558.88
5	Dairy	101276.46
6	Canned	90706.73
7	Baking Goods	81894.74
8	Health and Hygiene	68025.84
9	Meat	59449.86
10	Soft Drinks	58514.16
11	Breads	35379.12
12	Hard Drinks	29334.68
13	Others	22451.89
14	Starchy Foods	21880.03
15	Breakfast	15596.70
16	Seafood	9077.87

## 10). Fat Content by Outlet Location Type for Total Sales

```
SELECT Outlet_Location_Type,

CAST(SUM(CASE WHEN Item_Fat_Content = 'Low Fat' THEN Total_Sales

ELSE 0 END)AS DECIMAL (10,2)) AS Low_Fat_Sales,

CAST(SUM(CASE WHEN Item_Fat_Content = 'Regular' THEN Total_Sales

ELSE 0 END)AS DECIMAL (10,2))AS Regular_Sales

FROM blinkit_data
```

GROUP BY Outlet\_Location\_Type
ORDER BY Outlet\_Location\_Type;

Compares sales for low-fat and regular-fat products across different outlet locations.



## **Query Explanation:**

Query Line	Description
Outlet_Location_T ype	Groups data by each location type (e.g., Tier 1, Tier 2, Tier 3).
CASE WHEN THEN END	Separately sums sales for "Low Fat" and "Regular" items using conditional logic.
CAST( AS DECIMAL(10,2))	Formats the total sales to two decimal places.
GROUP BY	Ensures totals are calculated for each outlet location type.
ORDER BY	Sorts the results alphabetically by location type.

## 11).Percentage of Sales by Outlet size

```
SELECT
```

```
Outlet_Size,

CAST (SUM(Total_Sales) AS DECIMAL (10,2)) AS Total_Sales,

ROUND(

(SUM(Total_Sales) * 100.0) /
```

```
(SELECT SUM(Total_Sales) FROM blinkit_data), 2
) AS Sales_Percentage
FROM blinkit_data
GROUP BY Outlet_Size
ORDER BY Sales_Percentage DESC;
```

Shows sales contribution by outlet size in percentage terms.

⊞ Results				
	Outlet_Size	Total_Sales	Sales_Percentage	
1	Medium	507895.73	42.27	
2	Small	444794.17	37.01	
3	High	248991.58	20.72	

## **Query Explanation:**

Query Line	Description
Outlet_Size	Groups the results by outlet size (e.g., Small, Medium, High).
SUM(Total_Sales)	Calculates total sales for each outlet size.
CAST( AS DECIMAL(10,2))	Formats the total sales to two decimal places.
ROUND(, 2)	Computes the percentage of overall sales for each outlet size, rounded to two decimals.
ORDER BY Sales_Percentage DESC	Sorts outlet sizes from highest to lowest percentage contribution.

## 12). Sales by Outlet Location

```
SELECT Outlet_Location_Type,

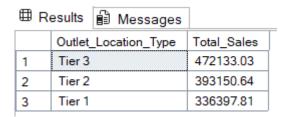
CAST(SUM(Total_Sales) AS DECIMAL(10,2)) AS Total_Sales

FROM blinkit_data

GROUP BY Outlet_Location_Type
```

## ORDER BY Total\_Sales DESC;

Compares total sales by outlet location tiers.



#### 13). Highest Rated Item

**SELECT\*** 

FROM blinkit\_data

WHERE Rating = (SELECT MAX(Rating) FROM blinkit\_data);

Shows all the items with the highest rating in the dataset.

## 14). Top 3 Best-Selling Item Types by Total Sales

SELECT TOP 3 Item\_Type,

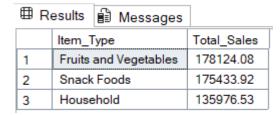
CAST (SUM(Total\_Sales)AS DECIMAL(10,2)) AS Total\_Sales

FROM blinkit\_data

**GROUP BY Item\_Type** 

ORDER BY Total\_Sales DESC;

Identifies the top 3 item categories by total sales



## 15).Top Item in Each Outlet Type

```
SELECT Outlet_Type, Item_Type, Total_Sales

FROM (

SELECT Outlet_Type,

Item_Type,

CAST(SUM(Total_Sales)AS DECIMAL(10,2))AS Total_Sales,

ROW_NUMBER() OVER (PARTITION BY Outlet_Type ORDER BY SUM(Total_Sales) DESC) AS rn

FROM blinkit_data

GROUP BY Outlet_Type, Item_Type
) ranked

WHERE rn = 1;
```

Uses window function to get the highest-selling item type for each outlet type.

⊞ Results 🖺 Messages				
	Outlet_	Туре	Item_Type	Total_Sales
1	Grocer	y Store	Fruits and Vegetables	21423.41
2	Superi	market Type1	Fruits and Vegetables	117431.99
3	Superi	market Type2	Snack Foods	20003.24
4	Superi	market Type3	Fruits and Vegetables	20301.39

## **Query Explanation:**

Query Line	Description
SELECT Outlet_Type, Item_Type, Total_Sales	Selects the final result: top item by outlet type.
CAST(SUM(Total_Sales) AS DECIMAL(10,2))	Calculates total sales per item in each outlet, formatted to two decimals.
ROW_NUMBER( )	A window function that assigns a sequential number (1, 2, 3) to rows within each group.

OVER ()	Defines the window (i.e., how the data is grouped and sorted before assigning the row number).
PARTITION BY Outlet_Type	Groups the data by each outlet_type, so numbering starts over for each outlet type.
ORDER BY SUM(Total_Sales) DESC	Within each outlet group, items are sorted by total sales, from highest to lowest.
WHERE rn = 1	Filters to show only the top-selling item (rank 1) for each outlet type.

## 16). Analyze sales growth over years by outlet establishment year

SELECT Outlet\_Establishment\_Year,

CAST(SUM(Total\_Sales) AS DECIMAL (10,2)) AS Yearly\_Sales

FROM blinkit\_data

GROUP BY Outlet\_Establishment\_Year

ORDER BY Outlet\_Establishment\_Year;

Tracks sales performance across the years of outlet establishment.

⊞ R	esults 🛍 Messages	
	Outlet_Establishment_Ye	ear Yearly_Sales
1	1998	204522.26
2	2000	131809.02
3	2010	132113.37
4	2011	78131.56
5	2012	130476.86
6	2015	130942.78
7	2017	133103.91
8	2020	129103.96
9	2022	131477.77

## 17). Average Sales and Rating by Outlet Size

WITH Outlet\_Summary AS (

```
SELECT Outlet_Size,

CAST(AVG(Total_Sales)AS DECIMAL (10,2))AS Average_Sales,

CAST(AVG(Rating) AS DECIMAL(10,2)) AS Avg_Rating

FROM blinkit_data

GROUP BY Outlet_Size
)

SELECT * FROM Outlet_Summary

ORDER BY Average_Sales DESC;
```

Summarizes average sales and ratings by outlet size using a common table expression (CTE).

⊞ R	esults	i Me	essages		
	Outlet_S	Size	Average_	Sales	Avg_Rating
1	High		142.04		3.95
2	Small		141.70		3.96
3	Medium	ı	139.88		3.98

## **Query Explanation:**

Query Line	Description
WITH Outlet_Summary AS ()	Defines a CTE, like a temporary named result set, called Outlet_Summary.
CAST(AVG(Total_Sales) AS DECIMAL(10,2)) AS Average_Sales	Calculates and formats average sales per outlet size.
CAST(AVG(Rating) AS DECIMAL(10,2)) AS Avg_Rating	Calculates and formats average customer rating per outlet size.
GROUP BY Outlet_Size	Groups the results by outlet size (e.g., small, medium, high).
SELECT * FROM Outlet_Summary	Retrieves the summarized results from the CTE.
ORDER BY Average_Sales DESC	Sorts the outlet sizes from highest to lowest average sales.

#### 18).RANK Top 3 Item\_Types by Sales per Outlet\_Type

Uses ranking function to list the top 3 best-selling items in each outlet type.

⊞ Results					
	Outlet	_Туре	Item_Type	Total_Sales	rnk
1	Groce	ry Store	Fruits and Vegetables	21423.4145393372	1
2	Groce	ry Store	Snack Foods	21327.8401908875	2
3	Groce	ry Store	Household	18082.1093406677	3
4	Super	market Type1	Fruits and Vegetables	117431.988025665	1
5	Super	market Type1	Snack Foods	114296.130218506	2
6	Super	market Type1	Household	89143.4092063904	3
7	Super	market Type2	Snack Foods	20003.2424430847	1
8	Super	market Type2	Fruits and Vegetables	18967.2917633057	2
9	Super	market Type2	Household	14244.6485176086	3
10	Super	market Type3	Fruits and Vegetables	20301.386592865	1
11	Super	market Type3	Snack Foods	19806.7075805664	2
12	Supermarket Type3		Household	14506.3581428528	3

## 19). Total Sales by Item Visibility Range

**SELECT** 

**CASE** 

```
WHEN Item_Visibility < 0.05 THEN 'Low Visibility'

WHEN Item_Visibility < 0.15 THEN 'Medium Visibility'

ELSE 'High Visibility'

END AS Visibility_Level,

COUNT(*) AS Item_Count,

CAST(SUM(Total_Sales) AS DECIMAL(10,2)) AS Total_Sales,

CAST(AVG(Total_Sales) AS DECIMAL(10,2)) AS Avg_Sale

FROM blinkit_data

GROUP BY

CASE

WHEN Item_Visibility < 0.05 THEN 'Low Visibility'

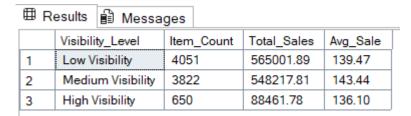
WHEN Item_Visibility < 0.15 THEN 'Medium Visibility'

ELSE 'High Visibility'

END

ORDER BY Total_Sales DESC;
```

Categorizes items based on visibility and analyzes total and average sales for each group.



#### **Query Explanation:**

Query Line	Description
CASE WHEN Item_Visibility < 0.05 THEN 'Low Visibility'	Creates custom visibility categories based on numeric visibility values.
COUNT(*) AS Item_COUNT	Counts how many items fall into each visibility category.

SUM(Total_Sales) and AVG(Total_Sales)	Calculates total and average sales for each visibility level.
GROUP BY CASE END	Groups the dataset based on the custom visibility_level.
ORDER BY Total_Sales DESC	Displays the groups sorted by total sales, from highest to lowest.

## 20).All Metrics by Outlet Type

```
SELECT Outlet_Type,

CAST(SUM(Total_Sales) AS DECIMAL (10,2)) AS Total_Sales,

CAST(AVG(Total_Sales) AS DECIMAL (10,2)) AS Average_Sales,

COUNT(*) AS Number_of_Items,

CAST(AVG(Rating) AS DECIMAL(10,2)) AS Avg_Rating,

CAST(AVG(Item_Visibility) AS DECIMAL(10,2)) AS Item_Visibility

FROM blinkit_data

GROUP BY Outlet_Type

ORDER BY Total_Sales DESC;
```

Gives a comprehensive summary of performance by outlet type across multiple metrics.



## **Conclusion:**

This project demonstrates practical SQL proficiency, including data cleaning, transformation, aggregation, and advanced analytics using SSMS. These

ypes.			

insights can help businesses optimize performance across products and store