# Blinkit Grocery Sales Analysis – SQL Project Documentation

# **Project Overview**

The purpose of this project is to analyze grocery sales data from Blinkit using SQL.

#### **Database Table Creation**

A table named BlinkitGrocery was created to store grocery sales data, with appropriate data types to reflect the nature of each field:

```
CREATE TABLE BlinkitGrocery (Item_FatContent VARCHAR(30),

Item_Identifier VARCHAR(10),

Item_Type VARCHAR(100),

Outlet_Establishment_Year INTEGER,

Outlet_Identifier VARCHAR(30),

Outlet_Location_Type VARCHAR(20),

Outlet_Size VARCHAR(20),

Outlet_Type VARCHAR(80),

Item_Visibility DOUBLE PRECISION,

Item_Weight NUMERIC(8,5),

Sales NUMERIC(12,2),

Rating NUMERIC(2,1)

);
```

## **Data Import**

Data was imported using the PostgreSQL \COPY command from a CSV file:

```
\COPY BlinkitGrocery(Item_FatContent, Item_Identifier, Item_Type,
Outlet_Establishment_Year, Outlet_Identifier, Outlet_Location_Type,
Outlet_Size, Outlet_Type, Item_Visibility, Item_Weight, Sales, Rating)
FROM 'C:/Users/hp/Downloads/BlinkIT Grocery Data.csv'
WITH (FORMAT csv, HEADER true);
```

## **Data Cleaning**

Standardization of the Item\_FatContent column was performed to unify data representation. The presence of multiple variations of the same category(e.g. LF, low fat vs Low Fat) can cause issue in reporting ,aggregations and filtering. By standardizing these values, we improve data quality, making it easier to generate insights and maintain uniformity in our datasets.

```
UPDATE BlinkitGrocery
SET Item_FatContent =
    CASE
         WHEN Item_FatContent IN ('LF', 'low fat') THEN 'Low Fat'
         WHEN Item_FatContent = 'reg' THEN 'Regular'
         ELSE Item_FatContent
END;
```

Verification after cleaning:

SELECT DISTINCT (Item FatContent) FROM BlinkitGrocery;



# A. KPI Requirements

**KPI Queries:** Focus on summarizing overall sales, averages, and counts.

#### 1. Total Sales (in Millions)

Calculates the total revenue from all items sold, displayed in millions for readability:

SELECT CAST(SUM(Sales)/1000000 AS DECIMAL(10,2)) AS Total\_Sales\_Millions FROM BlinkitGrocery;



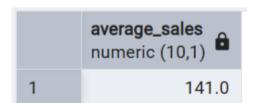
#### **Explanation:**

- SUM(sales): Adds up the total sales from all rows.
- Divided by 1,000,000 to express the total sales in **millions** for readability.
- CAST (... AS DECIMAL (10, 2)): Rounds the result to 2 decimal places for clarity.

#### 2. Average Sales

Computes the average revenue per transaction:

SELECT CAST(AVG(Sales) AS DECIMAL(10,1)) AS Average\_Sales
FROM BlinkitGrocery;



#### **Explanation:**

• AVG(sales): Calculates the mean sales value from all rows.

#### 3. Number of Items Sold

Counts the distinct types of items sold:

SELECT COUNT(\*) AS Number\_of\_Items
FROM BlinkitGrocery;

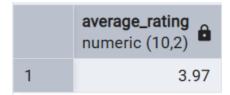


• COUNT (\*): Counts how many items are present in the dataset.

#### 4. Average Rating

Calculates the average customer rating for all items sold, rounded to two decimal places:

SELECT CAST(AVG(Rating) AS DECIMAL(10,2)) AS Average\_Rating FROM BlinkitGrocery;



#### **Explanation:**

- AVG (rating): Computes the average of the rating column.
- CAST(... AS DECIMAL(10,2)): Formats the average to 2 decimal places.

# B. Granular Requirements

**Granular Queries:** Focus on breaking down sales by specific categories like fat content, item types, and outlet attributes.

#### 1. Total Sales by Fat Content

#### **Objective:**

Analyze how fat content categories contribute to total sales.

	item_fatcontent character varying (30)	total_sales_thousand numeric (10,2)	
1	Low Fat	776.32	
2	Regular	425.36	

#### **Explanation:**

- Groups the dataset based on fat content (Low Fat, Regular).
- Sums sales for each fat content group.
- Divides by 1,000 to express sales in thousands.
- Orders the results from highest to lowest sales.

#### 2. Total Sales by Item Type

#### **Objective:**

Identify the contribution of different item types in terms of total sales.

	item_type character varying (100)	total_sales numeric (10,2)
1	Fruits and Vegetables	178124.26
2	Snack Foods	175434.21

- Groups data by each unique item type.
- Calculates total sales per item type.
- Orders the results from highest to lowest sales.

#### 3. Fat Content by Outlet for Total Sales

#### **Objective:**

Compare total sales across different outlet locations, segmented by fat content.

	item_type character varying (100)	total_sales numeric (10,2)	
1	Fruits and Vegetables	178124.26	
2	Snack Foods	175434.21	
3	Household	135976.83	
4	Frozen Foods	118559.35	
5	Dairy	101276.66	
6	Canned	90706.56	
7	Baking Goods	81894.55	
8	Health and Hygiene	68025.59	
9	Meat	59449.61	
10	Soft Drinks	58514.21	
11	Breads	35379.31	
12	Hard Drinks	29334.68	
13	Others	22451.84	
14	Starchy Foods	21879.94	
15	Breakfast 15596.		
16	Seafood	9078.00	

- Aggregates total sales grouped by both outlet location and item fat content.
- Converts rows into columns using a **Pivot Table**, showing separate columns for **Low Fat** and **Regular** sales per outlet.
- Missing values are replaced with zero using ISNULL().

#### 4. Total Sales by Outlet Establishment Year

#### **Objective:**

Understand how the establishment year of outlets affects total sales.

```
SELECT outlet_establishment_year,

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

FROM blinkitgrocery

GROUP BY outlet_establishment_year

ORDER BY outlet_establishment_year;
```

	outlet_establishment_year integer	total_sales numeric (10,2)	
1	2011	78131.64	
2	2012	130476.89	
3	2014	131809.04	
4	2015	130942.91	
5	2016	132113.52	
6	2017	133103.99	
7	2018	204522.35	
8	2020	129104.07	
9	2022	131477.89	

- Groups sales data by the year each outlet was established.
- Sums total sales for each year group.
- Orders the results in chronological order of establishment.

# C. Chart's Requirements

**Chart Queries**: Breaking down sales by specific categories like fat content, item types, and outlet attributes for clear visualization.

#### 1. Percentage of Sales by Outlet Size

#### **Objective:**

Analyze how much each outlet size contributes to total sales in percentage terms.

```
SELECT
    outlet_size,
    CAST(SUM(sales) AS DECIMAL(10, 2)) AS Total_Sales,
    CAST(SUM(sales) * 100.0 / SUM(SUM(sales)) OVER () AS DECIMAL(10, 2)) AS
Sales_Percentage
FROM
    blinkitgrocery
GROUP BY
    outlet_size
```

	outlet_size character varying (20)	total_sales numeric (10,2)	sales_percentage numeric (10,2)
1	Medium	507896.10	42.27
2	Small	444794.56	37.01
3	High	248991.64	20.72

- Groups the sales data by outlet size.
- Sums sales per outlet size.
- Calculates the total contribution of each outlet size to the company's total sales.
- Useful for understanding which outlet sizes are driving the most revenue.

#### 2. Sales by Outlet Location

#### **Objective:**

Assess how total sales are distributed across various outlet locations.

```
SELECT
    outlet_location_type,
    CAST(SUM(sales) AS DECIMAL(10, 2)) AS Total_Sales

FROM
    blinkitgrocery

GROUP BY
    outlet_location_type

ORDER BY
    outlet_location_type;
```

	outlet_location_type character varying (20)	total_sales numeric (10,2)
1	Tier 1	336398.02
2	Tier 2	393150.97
3	Tier 3	472133.31

- Groups sales data by outlet\_location\_type.
- Calculates total sales per location type (e.g., Tier 1, Tier 2 cities).
- Helps identify geographic performance differences.

#### 3. All Metrics by Outlet Type

#### **Objective:**

Provide a complete overview of key sales and performance metrics for each outlet type.

```
SELECT
    outlet_type,
    CAST(SUM(sales) AS DECIMAL(10, 2)) AS Total_Sales,
    CAST(AVG(sales) AS DECIMAL(10, 2)) AS Average_Sales,
    COUNT(*) AS Number_of_Items,
    CAST(AVG(rating) AS DECIMAL(10, 2)) AS Average_Rating
FROM
    blinkitgrocery
GROUP BY
    outlet_type
ORDER BY
    Total_Sales DESC;
```

	outlet_type character varying (80)	total_sales numeric (10,2)	average_sales numeric (10,2)	number_of_items bigint	average_rating numeric (10,2)
1	Supermarket Type3	130714.74	139.80	935	3.95
2	Supermarket Type2	131477.89	141.68	928	3.97
3	Grocery Store	151939.25	140.29	1083	3.99
4	Supermarket Type1	787550.42	141.21	5577	3.96

- Groups data by outlet\_type.
- Calculates:
  - o Total Sales (SUM(sales))
  - o Average Sales (AVG(sales))
  - Number of Items Sold (COUNT (\*))
  - o Average Customer Rating (AVG (rating))
- Offers a multi-metric view to assess outlet performance more holistically.