**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'

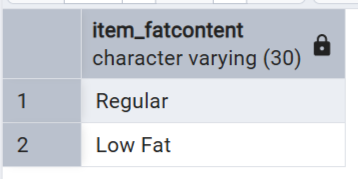
WHEN Item\_FatContent = 'reg' THEN 'Regular'

ELSE Item\_FatContent

END;

Verification after cleaning:

SELECT DISTINCT(Item\_FatContent) FROM BlinkitGrocery;



## ****A. KPI Requirements****

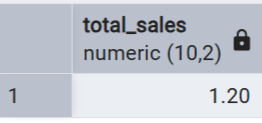
**KPI Queries:** Focus on summarizing overall sales, averages, and counts.

### 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:**

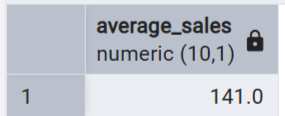
* SUM(sales): Adds up the total sales from all rows.
* Divided by 1,000,000 to express the total sales in **millions** for readability.
* CAST(... AS DECIMAL(10,2)): Rounds the result to 2 decimal places for clarity.

### ****2. Average Sales****

Computes the average revenue per transaction:

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

FROM BlinkitGrocery;

****

**Explanation:**

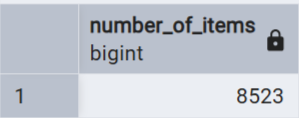
* AVG(sales): Calculates the mean sales value from all rows.

### ****3. Number of Items Sold****

Counts the distinct types of items sold:

SELECT COUNT(\*) AS Number\_of\_Items

FROM BlinkitGrocery;

****

**Explanation:**

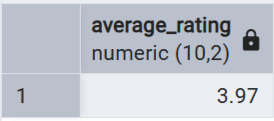
* COUNT(\*): Counts how many items are present in the dataset.

### ****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.

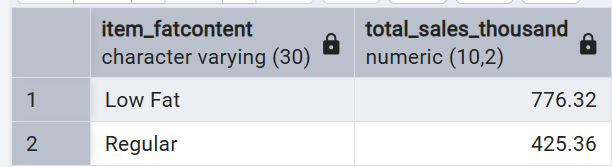
SELECT item\_fatcontent,

CAST(SUM(sales) / 1000 AS DECIMAL(10,2)) AS Total\_sales\_thousand

FROM blinkitgrocery

GROUP BY item\_fatcontent

ORDER BY Total\_sales\_thousand DESC;

****

**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.

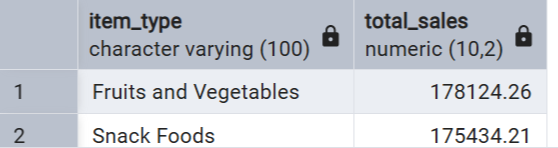
SELECT item\_type,

CAST(SUM(sales) AS DECIMAL(10,2)) AS Total\_sales

FROM blinkitgrocery

GROUP BY item\_type

ORDER BY Total\_sales DESC;

****

**Explanation:**

* 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.

SELECT outlet\_location\_type,

ISNULL([Low Fat], 0) AS [Low Fat],

ISNULL([Regular], 0) AS [Regular]

FROM

(

SELECT outlet\_location\_type,

item\_fatcontent,

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

FROM blinkitgrocery

GROUP BY outlet\_location\_type, item\_fatcontent

) AS sourceTable

PIVOT

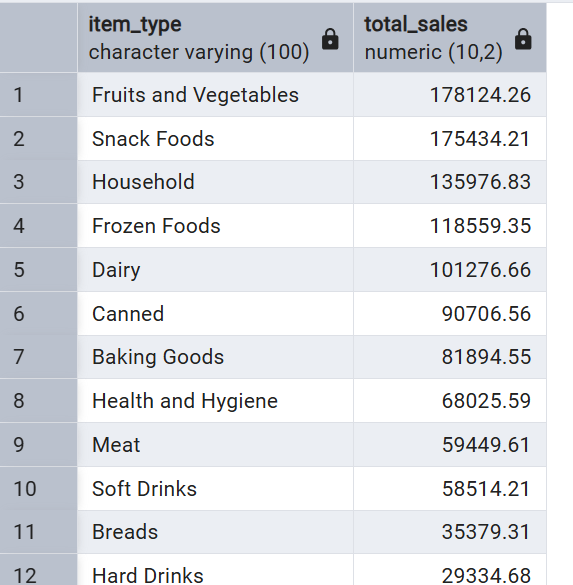
(

SUM(Total\_Sales)

FOR item\_fatcontent IN ([Low Fat], [Regular])

) AS PivotTable

ORDER BY outlet\_location\_type;

****

****

**Explanation:**

* 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.

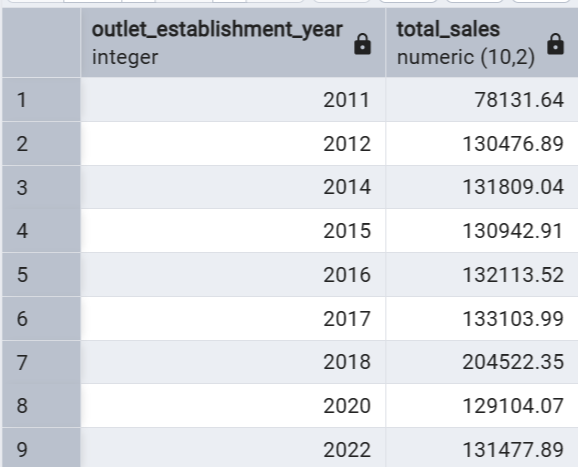
SELECT outlet\_establishment\_year,

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

FROM blinkitgrocery

GROUP BY outlet\_establishment\_year

ORDER BY outlet\_establishment\_year;

****

**Explanation:**

* 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

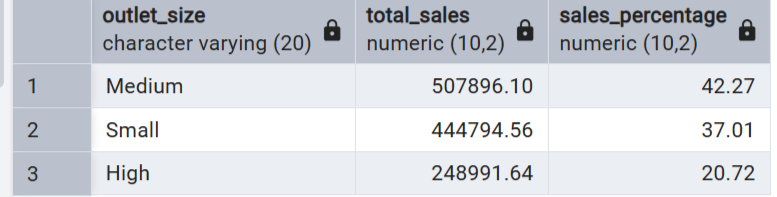
blinkitgrocery

GROUP BY

outlet\_size

ORDER BY

Total\_Sales DESC;

****

**Explanation:**

* 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

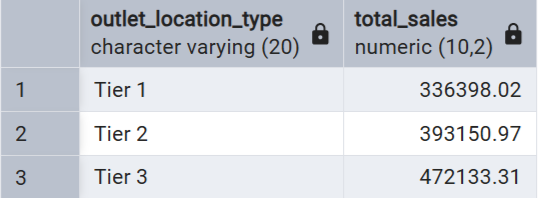
blinkitgrocery

GROUP BY

outlet\_location\_type

ORDER BY

outlet\_location\_type;

****

**Explanation:**

* 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

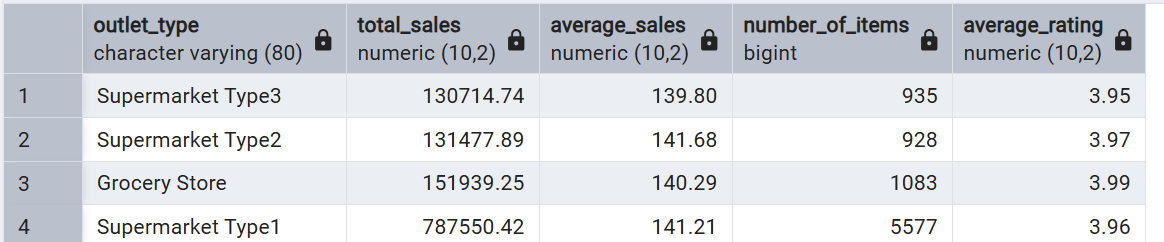
blinkitgrocery

GROUP BY

outlet\_type

ORDER BY

Total\_Sales DESC;

****

**Explanation:**

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