BlinkIt Analysis

- See all the data imported:

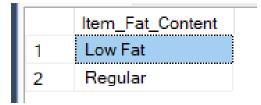SELECT * FROM Blinkit_Grocery_Data

- **DATA CLEANING:**

Cleaning the Item_Fat_Content field ensures data consistency and accuracy in analysis. The presence of multiple variations of the same category (e.g., LF, low fat vs. Low Fat) can cause issues 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.

```sql
UPDATE Blinkit_Grocery_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;
```

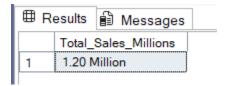After executing this query check the data has been cleaned or not using below query

SELECT DISTINCT Item_Fat_Content FROM Blinkit_Grocery_Data;

	Item_Fat_Content
1	Low Fat
2	Regular

A. KPI's

1. TOTAL SALES:

```
SELECT CONCAT(CAST(SUM(Total_Sales) / 1000000.0 AS DECIMAL(10,2)), "Million") AS
Total_Sales_Million
FROM Blinkit_Grocery_Data;
```

	Total_Sales_Millions
1	1.20 Million

2. AVERAGE SALES

```
SELECT CAST(AVG(Total_Sales) AS INT) AS Avg_Sales
FROM Blinkit_Grocery_Data;
```

	Avg_Sales
1	140

3. NO OF ITEMS

```
SELECT COUNT(*) AS No_of_Orders
FROM Blinkit_Grocery_Data;
```

	No_of_Orders
1	8523

4. AVG RATING

```
SELECT CAST(AVG(Rating) AS DECIMAL(10, 2)) AS Avg_Rating
FROM Blinkit_Grocery_Data;
```

	Avg_Rating
1	3.97

5. Totals Sales of Low Fat in Millions

```
select    Concat(CAST(sum(Total_Sales)/1000000    as    decimal(10,2)),    '    Million')    as
Total_Sales_Millions from Blinkit_Grocery_Data where Item_Fat_Content = 'Low Fat'
```

	Total_Sales_Millions
1	0.78 Million

6. Totals Sales of Outlet_Establishment_Year 2022

select   Concat(CAST(sum(Total_Sales)/1000000   as   decimal(10,2)),   '   Million')   as
Total_Sales_Millions from Blinkit_Grocery_Data where Outlet_Establishment_Year = 2022

	Total_Sales_Outlet_Establishment_Year_2022
1	0.13 Million

B. Total Sales by Fat Content:

SELECT Item_Fat_Content, CAST(SUM(Total_Sales) AS DECIMAL(10,2)) AS Total_Sales
FROM Blinkit_Grocery_Data
GROUP BY Item_Fat_Content

	Item_Fat_Content	Total_Sales
1	Low Fat	776319.68
2	Regular	425361.80

AvgSales, Total Sales, Count by Fat Content:

select Item_Fat_Content,
        Concat(Cast(SUM(Total_Sales) as decimal(10,2)),'K') as Total_Sales_Thousands,
        Cast(AVG(Total_Sales) as decimal(10,2)) as Avg_Sales,
        Count(*) as No_Of_Items,
        Cast(AVG(Rating) as decimal(10,2)) as Avg_Rating
from Blinkit_Grocery_Data group by Item_Fat_Content

	Item_Fat_Content	Total_Sales_Thousands	Avg_Sales	No_Of_Items	Avg_Rating
1	Low Fat	776.32 K	140.71	5517	3.97
2	Regular	425.36 K	141.50	3006	3.97

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 No_Of_Items,
Cast(AVG(Rating) as decimal(10,2)) as Avg_Rating
from Blinkit_Grocery_Data group by Item_Type order by Total_Sales desc

⊞ Results   📄 Messages

	Item_Type	Total_Sales	Avg_Sales	No_Of_Items	Avg_Rating
1	Fruits and Vegetables	178124.08	144.58	1232	3.96
2	Snack Foods	175433.92	146.19	1200	3.95
3	Household	135976.53	149.42	910	4.00
4	Frozen Foods	118558.88	138.50	856	3.97
5	Dairy	101276.46	148.50	682	3.97
6	Canned	90706.73	139.76	649	3.99
7	Baking Goods	81894.74	126.38	648	3.98
8	Health and Hygiene	68025.84	130.82	520	3.99
9	Meat	59449.86	139.88	425	4.02
10	Soft Drinks	58514.16	131.49	445	3.92
11	Breads	35379.12	140.95	251	3.88
12	Hard Drinks	29334.68	137.08	214	3.91
13	Others	22451.89	132.85	169	3.95
14	Starchy Foods	21880.03	147.84	148	3.92
15	Breakfast	15596.70	141.79	110	3.93
16	Seafood	9077.87	141.84	64	3.96

D. Fat Content by Outlet for Total Sales
SELECT Outlet_Location_Type,
    ISNULL([Low Fat], 0) AS Low_Fat,
    ISNULL([Regular], 0) AS Regular
FROM
(
  SELECT Outlet_Location_Type, Item_Fat_Content,
      CAST(SUM(Total_Sales) AS DECIMAL(10,2)) AS Total_Sales
  FROM Blinkit_Grocery_Data

```sql
    GROUP BY Outlet_Location_Type, Item_Fat_Content
) AS SourceTable
PIVOT
(
    SUM(Total_Sales)
    FOR Item_Fat_Content IN ([Low Fat], [Regular])
) AS PivotTable
ORDER BY Outlet_Location_Type;
```

Outlet_Location_Type	Low_Fat	Regular
Tier 1	215047.91	121349.90
Tier 2	254464.77	138685.87
Tier 3	306806.99	165326.03

Query Explanations

This query aims to transform the Blinkit_Grocery_Data table to display total sales (Total_Sales) for each combination of Outlet_Location_Type and Item_Fat_Content. The result will show Outlet_Location_Type as rows and Item_Fat_Content categories ("Low Fat" and "Regular") as columns. If there are no sales for a particular combination, the query will display 0 instead of NULL.

Detailed Explanation:

1. **Subquery**
   a. **Aggregation:**

      sql
      CopyEdit
      ```sql
      SELECT
          Outlet_Location_Type,
          Item_Fat_Content,
          CAST(SUM(Total_Sales) AS DECIMAL(10,2)) AS Total_Sales
      FROM
          Blinkit_Grocery_Data
      GROUP BY
          Outlet_Location_Type,
          Item_Fat_Content
      ```

- **Purpose:** This subquery groups the data by Outlet_Location_Type and Item_Fat_Content, calculating the total sales for each combination.
- **CAST(SUM(Total_Sales) AS DECIMAL(10,2)):** Sums the Total_Sales for each group and casts the result to a decimal with two decimal places for precision.
2. **PIVOT Operation:**
   a. **Pivoting:**

   sql
   CopyEdit
   PIVOT
   (
   * SUM(Total_Sales)*
   * FOR Item_Fat_Content IN ([Low Fat], [Regular])*
) AS PivotTable

- **Purpose:** Transforms the rows of Item_Fat_Content into columns ([Low Fat] and [Regular]).
- **SUM(Total_Sales):** Aggregates the Total_Sales for each Item_Fat_Content category within each Outlet_Location_Type.
3. **Main Query:**
   a. **Selecting and Handling NULLs:**

   sql
   CopyEdit
   SELECT
   * Outlet_Location_Type,*
   * ISNULL([Low Fat], 0) AS Low_Fat,*
   * ISNULL([Regular], 0) AS Regular*
   FROM
   * PivotTable*
   ORDER BY
   * Outlet_Location_Type;*

- **ISNULL([Low Fat], 0) AS Low_Fat:** Replaces any NULL values in the [Low Fat] column with 0 and renames the column to Low_Fat.
- **ISNULL([Regular], 0) AS Regular:** Similarly, replaces NULL values in the [Regular] column with 0.
- **ORDER BY Outlet_Location_Type:** Sorts the final result set by Outlet_Location_Type.

Why Use ISNULL?

When performing a PIVOT operation, if a particular combination of Outlet_Location_Type and Item_Fat_Content doesn't exist in the data, the resulting cell will contain a NULL value. Using ISNULL(column)

E. Total Sales by Outlet Establishment

SELECT Outlet_Establishment_Year, CAST(SUM(Total_Sales) AS DECIMAL(10,2)) AS Total_Sales
FROM Blinkit_Grocery_Data
GROUP BY Outlet_Establishment_Year
ORDER BY Outlet_Establishment_Year

	Outlet_Establishment_Year2	Total_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

F. Percentage of Sales by Outlet Size

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

```sql
    CAST((SUM(Total_Sales) * 100.0 / SUM(SUM(Total_Sales)) OVER()) AS DECIMAL(10,2)) AS
Sales_Percentage
FROM Blinkit_Grocery_Data
GROUP BY Outlet_Size
ORDER BY Total_Sales DESC;
```

Query Explanation:

Outlet_Size: This column represents the size category of the outlet (e.g., Small, Medium, Large).

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

- **SUM(Total_Sales)**: Calculates the total sales for each Outlet_Size.
- **CAST(... AS DECIMAL(10,2))**: Formats the resulting sum to a decimal number with two decimal places for precision.

CAST((SUM(Total_Sales) * 100.0 / SUM(SUM(Total_Sales)) OVER()) AS DECIMAL(10,2)) AS Sales_Percentage:

- **SUM(Total_Sales) * 100.0**: Multiplies the total sales of the current Outlet_Size by 100 to prepare for percentage calculation.
- **SUM(SUM(Total_Sales)) OVER()**:
    - **SUM(Total_Sales)**: Within the GROUP BY context, this computes the total sales for each Outlet_Size.
    - **SUM(...) OVER()**: The outer SUM combined with the OVER() clause calculates the grand total of all Total_Sales across all outlet sizes without collapsing the result set.
- **SUM(Total_Sales) * 100.0 / SUM(SUM(Total_Sales)) OVER()**: Divides the total sales of the current Outlet_Size by the grand total sales and multiplies by 100 to get the percentage contribution of each outlet size to the overall sales.
- **CAST(... AS DECIMAL(10,2))**: Formats the resulting percentage to two decimal places.

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

G. Sales by Outlet Location

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

FROM Blinkit_Grocery_Data

GROUP BY Outlet_Location_Type

ORDER BY Total_Sales DESC

	Outlet_Location_Type	Total_Sales
1	Tier 3	472133.03
2	Tier 2	393150.64
3	Tier 1	336397.81

H. 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(10,0)) AS Avg_Sales,

COUNT(*) AS No_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_Grocery_Data

GROUP BY Outlet_Type

ORDER BY Total_Sales DESC

	Outlet_Type	Total_Sales	Avg_Sales	No_Of_Items	Avg_Rating	Item_Visibility
1	Supermarket Type1	787549.89	141	5577	3.96	0.06
2	Grocery Store	151939.15	140	1083	3.99	0.10
3	Supermarket Type2	131477.77	142	928	3.97	0.06
4	Supermarket Type3	130714.67	140	935	3.95	0.06