QUERIES USED:

• See all the data imported:

```
SELECT * FROM blinkit_data
```

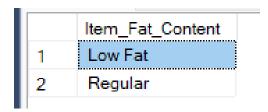
• DATA CLEANING:

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.

```
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;
```

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

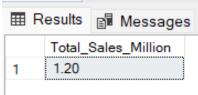
```
SELECT DISTINCT Item_Fat_Content FROM blinkit_data;
```



A. KPI's

1. TOTAL SALES:

```
SELECT CAST(SUM(Total_Sales) / 1000000.0 AS DECIMAL(10,2)) AS Total_Sales_Million
FROM blinkit_data;
```



2. AVERAGE SALES

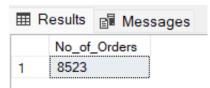
```
SELECT CAST(AVG(Total_Sales) AS INT) AS Avg_Sales
FROM blinkit_data;

Results Messages

Avg_Sales
1 140
```

3. NO OF ITEMS

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



4. AVG RATING

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



B. 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

Results Messages			
	Item_Fat_Content	Total_Sales	
1	Low Fat	776319.68	
2	Regular	425361.80	
	_		

C. 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



D. Fat Content by Outlet for Total Sales

⊞ F	Results 📳 Messages		
	Outlet_Location_Type	Low_Fat	Regular
1	Tier 1	215047.91	121349.90
2	Tier 2	254464.77	138685.87
3	Tier 3	306806.99	165326.03

Query Explanations

This query aims to transform the blinkit_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

```
Aggregation:
sql
CopyEdit
SELECT
Outlet_Location_Type,
Item_Fat_Content,
CAST(SUM(Total_Sales) AS DECIMAL(10,2)) AS Total_Sales
FROM
blinkit_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:

```
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:

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_data
GROUP BY Outlet_Establishment_Year
ORDER BY Outlet_Establishment_Year
```

Results 🖺 Messages				
	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,

CAST((SUM(Total_Sales) * 100.0 / SUM(SUM(Total_Sales)) OVER()) AS

DECIMAL(10,2)) AS Sales_Percentage

FROM blinkit_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.

${\it CAST((SUM(Total_Sales)*100.0 / SUM(SUM(Total_Sales))\ OVER())\ AS\ DECIMAL(10,2))\ AS\ Sales_Percentage:}$

- 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():
 - o SUM(Total_Sales): Within the GROUP BY context, this computes the total sales for each Outlet_Size.
 - o **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.

Results Messages				
	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_data
GROUP BY Outlet_Location_Type
ORDER BY Total_Sales DESC
```

Ⅲ F	Results	B Messages	
	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:

 	Results 🗐 Messag	jes				
	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