Blinkit Analysis — MySQL Executed queries

🔹 **See all the data**

```
SELECT *
FROM blinkit_data;
```

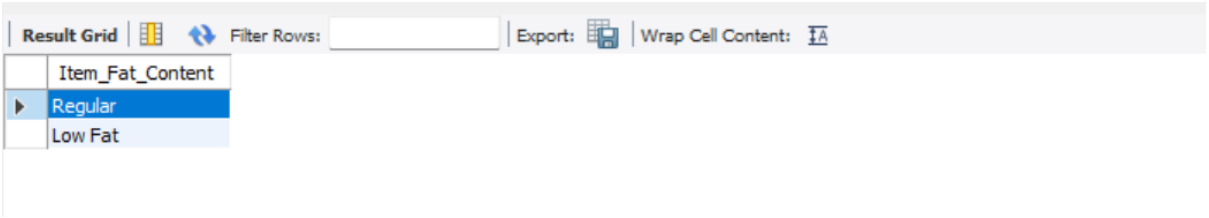
🔹 **DATA CLEANING (MySQL)**

👉 CASE works the same in MySQL, so this is almost unchanged.

```
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;
```

Check cleaned values

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




```
SET SQL_SAFE_UPDATES = 0;

  SET SQL_SAFE_UPDATES = 1;
```

🅐 **KPI's**

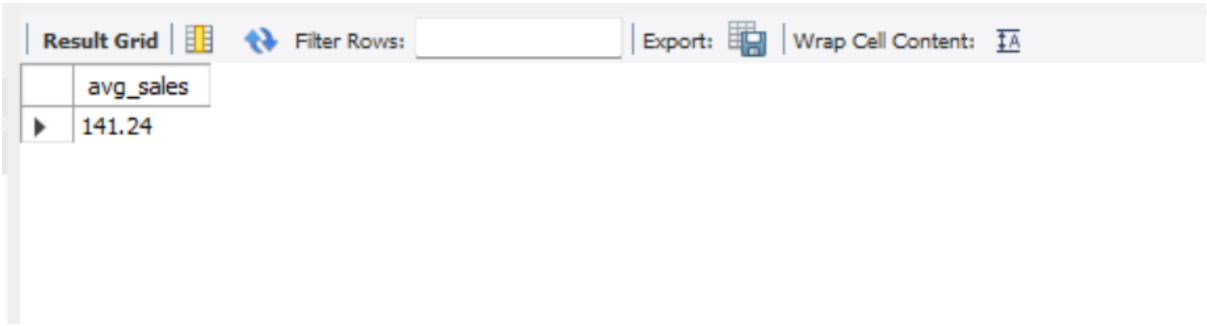
1️⃣ **TOTAL SALES (in Millions)**

```
SELECT
ROUND(SUM(Total_Sales) / 1000000, 2) AS Total_Sales_Million
FROM blinkit_data;
```

---

2 AVERAGE SALES

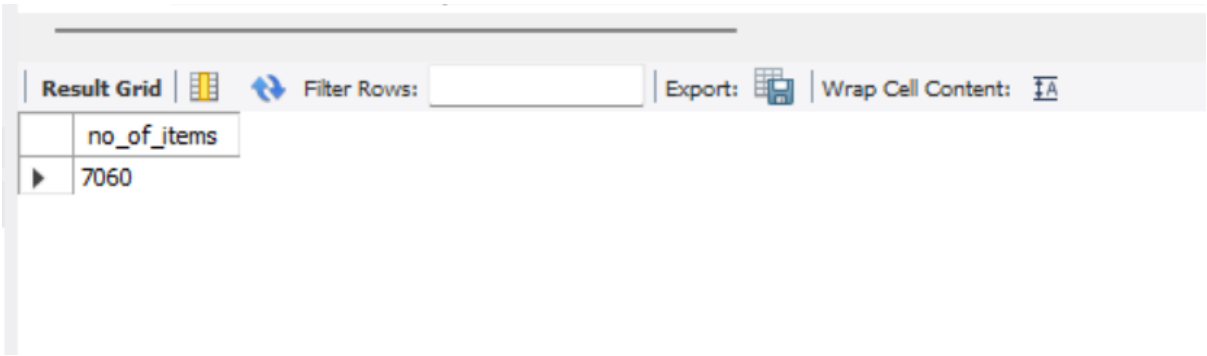
```
SELECT
ROUND(AVG(Total_Sales), 0) AS Avg_Sales
FROM blinkit_data;
```



---

3 NUMBER OF ITEMS

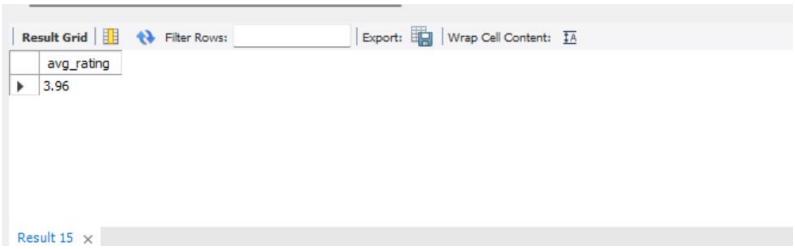
```
SELECT
COUNT(*) AS No_of_Orders
FROM blinkit_data;
```



---

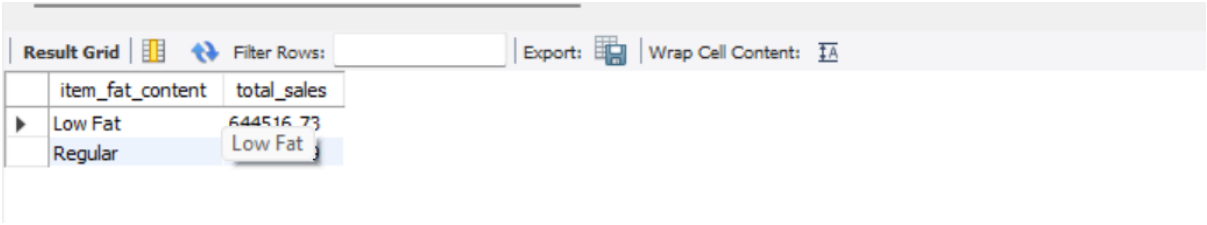
4 AVERAGE RATING

```
SELECT
ROUND(AVG(Rating), 1) AS Avg_Rating
FROM blinkit_data;
```



🅑 Total Sales by Fat Content

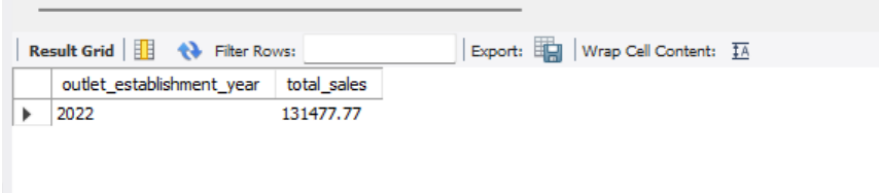
```
SELECT
Item_Fat_Content,
ROUND(SUM(Total_Sales), 2) AS Total_Sales
FROM blinkit_data
GROUP BY Item_Fat_Content;
```



If we want to find

What is the total sales for outlets established in **2022**?

```
77 •  SELECT
78        outlet_establishment_year,
79        ROUND(SUM(total_sales), 2) AS total_sales
80    FROM blinkit_data
81    WHERE outlet_establishment_year = 2022
82    GROUP BY outlet_establishment_year;
```



🅒 Total Sales by Item Type

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

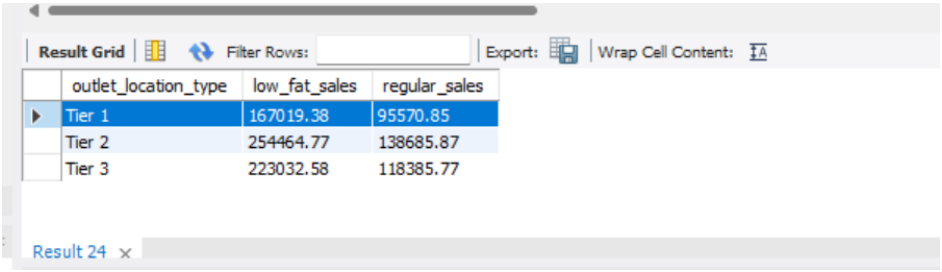
---

❶ Fat Content by Outlet for Total Sales

⚠️ MySQL does NOT support PIVOT

So we use **CASE + SUM** (industry-standard approach).

```
SELECT
Outlet_Location_Type,
ROUND(SUM(CASE WHEN Item_Fat_Content = 'Low Fat' THEN Total_Sales ELSE 0
END), 2) AS Low_Fat,
ROUND(SUM(CASE WHEN Item_Fat_Content = 'Regular' THEN Total_Sales ELSE 0
END), 2) AS Regular
FROM blinkit_data
GROUP BY Outlet_Location_Type
ORDER BY Outlet_Location_Type;
```



outlet_location_type	low_fat_sales	regular_sales
Tier 1	167019.38	95570.85
Tier 2	254464.77	138685.87
Tier 3	223032.58	118385.77

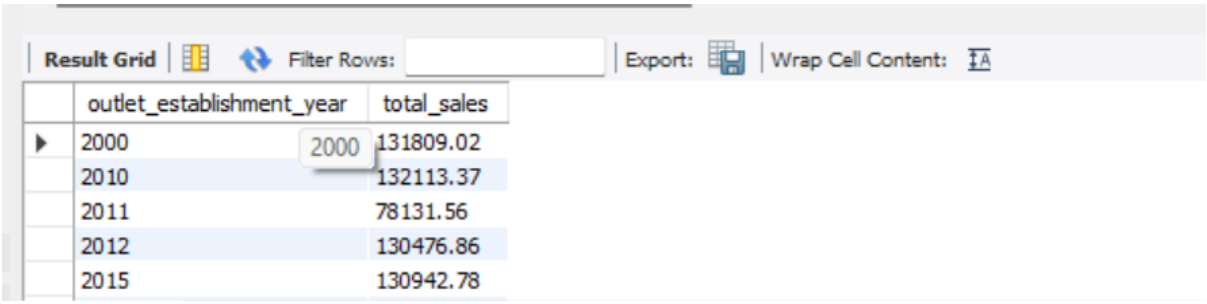
📌 Interview line

"Since MySQL doesn't support PIVOT, I used conditional aggregation with CASE."

---

❷ Total Sales by Outlet Establishment Year

```
SELECT
Outlet_Establishment_Year,
ROUND(SUM(Total_Sales), 2) AS Total_Sales
FROM blinkit_data
GROUP BY Outlet_Establishment_Year
ORDER BY Outlet_Establishment_Year;
```



outlet_establishment_year	total_sales
2000	131809.02
2010	132113.37
2011	78131.56
2012	130476.86
2015	130942.78

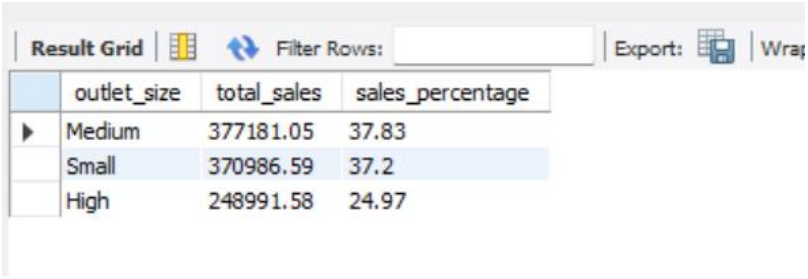
---

❶ Percentage of Sales by Outlet Size

⚠️ Window functions supported in MySQL 8+

```
SELECT
Outlet_Size,
ROUND(SUM(Total_Sales), 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 Total_Sales DESC;
```

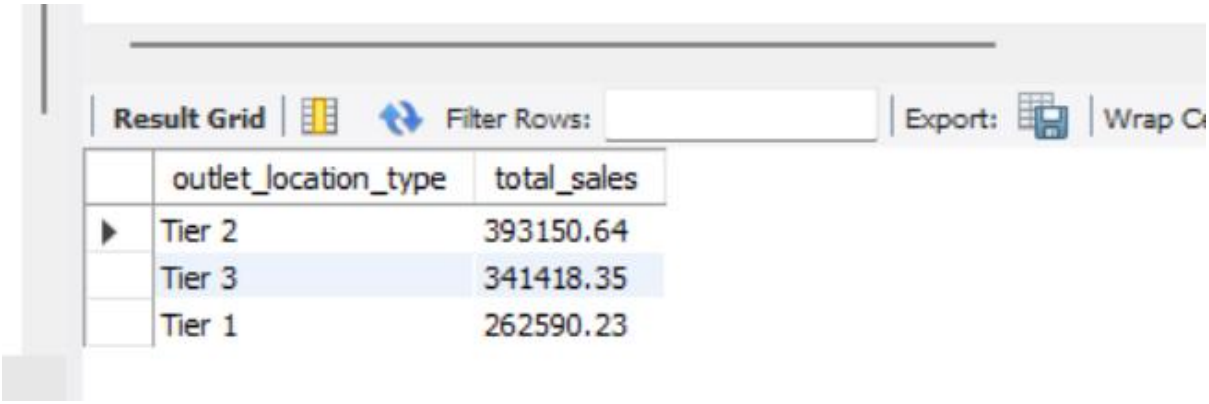
📌 Cleaner and more compatible than OVER().



---

❷ Sales by Outlet Location

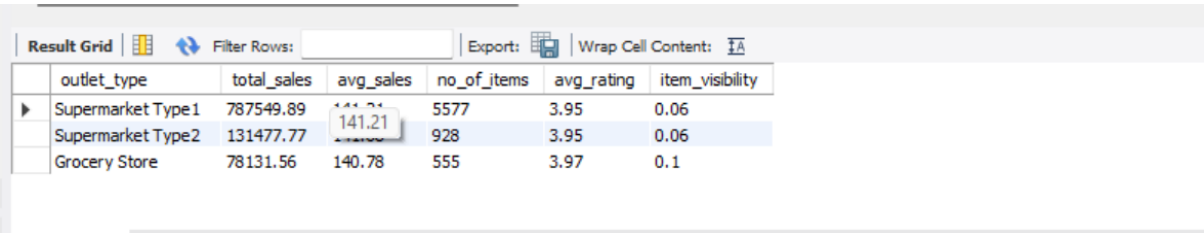
```
SELECT
Outlet_Location_Type,
ROUND(SUM(Total_Sales), 2) AS Total_Sales
FROM blinkit_data
GROUP BY Outlet_Location_Type
ORDER BY Total_Sales DESC;
```



---

❽ All Metrics by Outlet Type

```
SELECT
Outlet_Type,
ROUND(SUM(Total_Sales), 2) AS Total_Sales,
ROUND(AVG(Total_Sales), 0) AS Avg_Sales,
COUNT(*) AS No_Of_Items,
ROUND(AVG(Rating), 2) AS Avg_Rating,
ROUND(AVG(Item_Visibility), 2) AS Item_Visibility
FROM blinkit_data
GROUP BY Outlet_Type
ORDER BY Total_Sales DESC;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

outlet_type	total_sales	avg_sales	no_of_items	avg_rating	item_visibility
Supermarket Type1	787549.89	141.21	5577	3.95	0.06
Supermarket Type2	131477.77		928	3.95	0.06
Grocery Store	78131.56	140.78	555	3.97	0.1