# Blinkit Sales Analysis Report

Shaik Abdul Khadar

June 8, 2025

**Abstract**

This report presents a comprehensive SQL-based analysis of sales data from Blinkit's database, specifically the `Blinkit_data` table. The analysis leverages a variety of SQL queries to uncover insights into sales performance, item characteristics, outlet efficiency, and temporal trends. The report is structured to address basic data retrieval, aggregation, filtering, advanced analytics, and custom business logic, providing actionable insights for Blinkit's operational and strategic decision-making.

## 1 Introduction

Blinkit, a leading e-commerce platform, maintains a robust data set that captures sales, item attributes, and outlet characteristics. The `Blinkit_data` table contains key fields such as

- `Item_Fat_Content`,
- `Item_Identifier`,
- `Item_Type`,
- `Outlet_Establishment_Year`,
- `Outlet_Identifier`,
- `Outlet_Location_Type`,
- `Outlet_Size`,
- `Outlet_Type`,
- `Item_Visibility`,
- `Item_Weight`,
- `Sales`,
- `Rating`.

This report employs SQL queries to analyze these data, addressing 40 distinct analytical tasks grouped into categories such as basic queries, aggregation, filtering, window functions, data

cleaning, and advanced analytics.

# 2 Methodology

The analysis was conducted using SQL queries executed on the `blinkit_db` database, specifically the `Blinkit_data` table. The queries range from simple data retrieval to complex analytical operations, including aggregations, window functions, and correlation analysis. Each query is designed to address a specific business question, and the results are presented in a structured format to facilitate interpretation. The SQL queries are categorized as follows:

- **Basic SQL Queries**: Retrieve and summarize data.
- **Intermediate Aggregation**: Aggregate data by groups.
- **Filtering and Conditional Analysis**: Apply conditions to filter data.
- **Advanced Analysis**: Use complex SQL features like window functions.
- **Data Cleaning**: Handle missing or inconsistent data.
- **Custom Business Logic**: Derive business-specific metrics.
- **Analytical Queries**: Perform statistical and efficiency analyses.
- **Data Segmentation**: Segment data for targeted insights.
- **Comparative and Temporal Analysis**: Analyze trends over time.

# 3 Analysis and Results

## 3.1 Basic SQL Queries

1. **Retrieve all records from the Blinkit_data table.**

   ```
   SELECT * FROM Blinkit_data;
   ```

   *Result*: Retrieves all columns and rows, providing a complete view of the dataset for initial exploration.

2. **Number of unique item types.**

   ```
   SELECT COUNT(DISTINCT Item_Type) AS unique_item_types
   FROM Blinkit_data;
   ```

   *Result*: Returns the count of distinct `Item_Type` values, indicating product diversity.

3. **Average item weight across all records.**

```
SELECT AVG(Item_Weight) AS avg_weight FROM Blinkit_data;
```

*Result*: Computes the mean `Item_Weight`, useful for understanding typical product weights.

## 3.2 Intermediate Aggregation Questions

4. **Total sales amount per item type.**

```
SELECT Item_Type, SUM(Sales) AS total_sales FROM Blinkit_data
GROUP BY Item_Type;
```

*Result*: Aggregates sales by `Item_Type`, highlighting top-performing categories.

5. **Outlet with the highest total sales.**

```
SELECT Outlet_Identifier, SUM(Sales) AS total_sales
FROM Blinkit_data
GROUP BY Outlet_Identifier
ORDER BY total_sales DESC
LIMIT 1;
```

*Result*: Identifies the top-performing outlet by total sales.

6. **Average rating for each fat content type.**

```
SELECT Item_Fat_Content, AVG(Rating) AS avg_rating
FROM Blinkit_data
GROUP BY Item_Fat_Content;
```

*Result*: Shows average customer ratings for `Low Fat` and `Regular` items.

## 3.3 Filtering and Conditional Analysis

7. **Items with a rating greater than 4.5.**

```
SELECT * FROM Blinkit_data WHERE Rating > 4.5 ORDER BY Rating DESC;
```

*Result*: Lists high-rated items, useful for identifying customer favorites.

8. **Sales and ratings for Tier 1 outlets.**

```
SELECT Sales, Rating FROM Blinkit_data WHERE
Outlet_Location_Type = 'Tier 1';
```

*Result*: Filters data for Tier 1 locations, focusing on urban market performance.

9. **Count of Low Fat items weighing more than 1.5 kg.**

```
SELECT COUNT(*) AS count_items
FROM Blinkit_data
WHERE Item_Fat_Content = 'Low Fat' AND Item_Weight > 1.5;
```

*Result*: Quantifies items meeting specific health and weight criteria.

## 3.4 Advanced Analysis

10. **Year with the highest average item sales.**

```
SELECT Outlet_Establishment_Year, AVG(Sales) AS avg_sales
FROM Blinkit_data
GROUP BY Outlet_Establishment_Year
ORDER BY avg_sales DESC
LIMIT 1;
```

*Result*: Identifies the year with the highest average sales per outlet.

11. **Compare total sales of Low Fat vs Regular items by outlet type.**

```
SELECT
  Outlet_Type,
  SUM(CASE WHEN Item_Fat_Content = 'Low Fat' THEN Sales ELSE 0 END)
  AS Low_Fat_Sales,
  SUM(CASE WHEN Item_Fat_Content = 'Regular' THEN Sales ELSE 0 END)
  AS Regular_Sales
FROM Blinkit_data
GROUP BY Outlet_Type;
```

*Result*: Provides a comparison of sales contributions by fat content across outlet types.

12. **Percentage of total sales from Supermarket Type1 outlets.**

```
SELECT
  Outlet_Type,
  SUM(Sales) AS total_sales,
  SUM(Sales) * 100.0 / SUM(SUM(Sales)) OVER () AS sales_percentage
FROM Blinkit_data
WHERE Outlet_Type = 'Supermarket Type1'
GROUP BY Outlet_Type;
```

*Result*: Calculates the contribution of `Supermarket Type1` to total sales.

## 3.5 Window Functions & Ranking

13. **Rank outlets by total sales.**

```
SELECT Outlet_Identifier, SUM(Sales) AS total_sales,
       RANK() OVER (ORDER BY SUM(Sales) DESC) AS rank
FROM Blinkit_data
GROUP BY Outlet_Identifier;
```

*Result*: Ranks outlets based on total sales, highlighting top performers.

14. **Running total of sales per establishment year.**

```
SELECT Outlet_Establishment_Year, SUM(Sales) AS yearlyarch
       SUM(SUM(Sales)) OVER (ORDER BY Outlet_Establishment_Year)
       AS running_total
FROM Blinkit_data
GROUP BY Outlet_Establishment_Year;
```

*Result*: Shows cumulative sales over time by establishment year.

## 3.6 Data Cleaning / Updates

15. **Normalize fat content values.**

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

*Result*: Standardizes `Item_Fat_Content` values for consistency.

16. **Outlet establishment year with highest total sales.**

```
SELECT Outlet_Establishment_Year, SUM(Sales) AS total_sales
FROM Blinkit_data
GROUP BY Outlet_Establishment_Year
ORDER BY total_sales DESC
LIMIT 1;
```

*Result*: Identifies the year with the highest total sales contribution.

17. **Sales growth trend before and after 2010.**

```
SELECT
  CASE WHEN Outlet_Establishment_Year < 2010 THEN 'Before 2010'
  ELSE '2010 & After' END AS group_year,
  AVG(Sales) AS avg_sales
FROM Blinkit_data
GROUP BY group_year;
```

*Result*: Compares average sales between older and newer outlets.

18. **Average rating for outlets established after 2015.**

```
SELECT AVG(Rating) AS avg_rating
FROM Blinkit_data
WHERE Outlet_Establishment_Year > 2015;
```

*Result*: Evaluates customer satisfaction for newer outlets.

## 3.7 Correlation & Outlier Analysis

19. **Items with above-average visibility but below-average sales.**

```
WITH avgs AS (
  SELECT AVG(Item_Visibility) AS avg_vis, AVG(Sales)
  AS avg_sales FROM Blinkit_data
)
SELECT * FROM Blinkit_data, avgs
WHERE Item_Visibility > avg_vis AND Sales < avg_sales;
```

*Result*: Identifies potential underperforming items despite high visibility.

20. **Fat content types with low ratings but high sales.**

```
SELECT Item_Fat_Content, AVG(Sales) AS avg_sales,
AVG(Rating) AS avg_rating
FROM Blinkit_data
GROUP BY Item_Fat_Content
ORDER BY avg_sales DESC;
```

*Result*: Highlights fat content types with high sales but poor customer ratings.

21. **Item types with negative correlation between weight and rating.**

```
SELECT Item_Type, CORR(Item_Weight, Rating)
AS corr_weight_rating
FROM Blinkit_data
GROUP BY Item_Type;
```

*Result*: Identifies item types where heavier items receive lower ratings.

## 3.8 Top-N Analysis

22. **Top 5 item types by average rating.**

```
SELECT Item_Type, AVG(Rating) AS avg_rating
FROM Blinkit_data
GROUP BY Item_Type
ORDER BY avg_rating DESC
LIMIT 5;
```

*Result*: Lists the highest-rated item categories.

23. **Top 3 outlets per location tier by total sales.**

```
SELECT *
FROM (
  SELECT Outlet_Location_Type, Outlet_Identifier, SUM(Sales)
  AS total_sales,
         RANK() OVER (PARTITION BY Outlet_Location_Type
         ORDER BY SUM(Sales) DESC) AS rnk
  FROM Blinkit_data
  GROUP BY Outlet_Location_Type, Outlet_Identifier
) AS ranked
WHERE rnk <= 3;
```

*Result*: Identifies top-performing outlets within each location tier.

24. **Top 10 items by revenue.**

```
SELECT Item_Identifier, SUM(Sales) AS total_sales
FROM Blinkit_data
GROUP BY Item_Identifier
ORDER BY total_sales DESC
LIMIT 10;
```

*Result*: Highlights the highest revenue-generating items.

## 3.9 Multi-Level Grouping

25. **Average sales and rating by item type and outlet type.**

```
SELECT Outlet_Type, Item_Type, AVG(Sales) AS
avg_sales, AVG(Rating) AS avg_rating
```

```
FROM Blinkit_data
GROUP BY Outlet_Type, Item_Type;
```

*Result*: Provides detailed performance metrics across outlet and item types.

26. **Average sales by outlet size and fat content.**

```
SELECT Outlet_Size, Item_Fat_Content,
AVG(Sales) AS avg_sales
FROM Blinkit_data
GROUP BY Outlet_Size, Item_Fat_Content;
```

*Result*: Compares sales performance across outlet sizes and fat content.

## 3.10 Null and Missing Data Handling

27. **Count records with missing Item_Weight.**

```
SELECT COUNT(*) AS null_weight_count
FROM Blinkit_data
WHERE Item_Weight IS NULL;
```

*Result*: Quantifies missing data in the `Item_Weight` column.

28. **Average sales for non-null Item_Weight.**

```
SELECT AVG(Sales) AS avg_sales
FROM Blinkit_data
WHERE Item_Weight IS NOT NULL;
```

*Result*: Evaluates sales performance for records with valid weight data.

29. **Replace null Item_Weight with average by Item_Type.**

```
UPDATE Blinkit_data
SET Item_Weight = (
  SELECT AVG(b2.Item_Weight)
  FROM Blinkit_data b2
  WHERE b2.Item_Type = Blinkit_data.Item_Type
  AND b2.Item_Weight IS NOT NULL
)
WHERE Item_Weight IS NULL;
```

*Result*: Imputes missing weights with item-type-specific averages.

## 3.11 Custom Business Logic Scenarios

30. **Flag items as Best Seller.**

```
SELECT Item_Identifier, Sales, Rating,
  CASE WHEN Sales > 5000 AND Rating > 4.5 THEN 'Best Seller'
  ELSE 'Standard' END AS status
FROM Blinkit_data;
```

*Result*: Identifies top-performing items based on sales and rating thresholds.

31. **Performance score for each item.**

```
SELECT Item_Identifier, Sales, Rating, Item_Visibility,
       ROUND((Sales * Rating) / NULLIF(Item_Visibility, 0), 2)
       AS Performance_Score
FROM Blinkit_data;
```

*Result*: Calculates a custom performance metric for each item.

32. **Outlet with the widest variety of item types.**

```
SELECT Outlet_Identifier, COUNT(DISTINCT Item_Type)
AS item_type_count
FROM Blinkit_data
GROUP BY Outlet_Identifier
ORDER BY item_type_count DESC
LIMIT 1;
```

*Result*: Identifies the outlet with the most diverse product offerings.

## 3.12 Analytical Queries

33. **Highest average sales by Item Type and Fat Content.**

```
SELECT
  Item_Type,
  Item_Fat_Content,
  AVG(Sales) AS avg_sales
FROM Blinkit_data
GROUP BY Item_Type, Item_Fat_Content
ORDER BY avg_sales DESC;
```

*Result*: Identifies top-performing item type and fat content combinations.

34. **Outlet type with widest variation in ratings.**

```
SELECT
  Outlet_Type,
  STDDEV(Rating) AS rating_variation
FROM Blinkit_data
GROUP BY Outlet_Type
ORDER BY rating_variation DESC;
```

*Result*: Highlights outlet types with inconsistent customer ratings.

35. **Year with lowest average visibility.**

```
SELECT
  Outlet_Establishment_Year,
  AVG(Item_Visibility) AS avg_visibility
FROM Blinkit_data
GROUP BY Outlet_Establishment_Year
ORDER BY avg_visibility ASC
LIMIT 1;
```

*Result*: Identifies the year with the least visible products.

36. **Efficiency of smaller outlets (sales per kg).**

```
SELECT
  Outlet_Size,
  SUM(Sales) / SUM(Item_Weight) AS sales_per_kg
FROM Blinkit_data
WHERE Item_Weight IS NOT NULL
GROUP BY Outlet_Size
ORDER BY sales_per_kg DESC;
```

*Result*: Measures sales efficiency relative to item weight.

37. **Top 5 items by total sales.**

```
SELECT
  Item_Identifier,
  SUM(Sales) AS total_sales
FROM Blinkit_data
GROUP BY Item_Identifier
ORDER BY total_sales DESC
LIMIT 5;
```

*Result*: Lists the top revenue-generating items.

## 3.13 Data Segmentation Queries

38. **Average rating per outlet type in Tier 2.**

```
SELECT
  Outlet_Type,
  AVG(Rating) AS avg_rating
FROM Blinkit_data
WHERE Outlet_Location_Type = 'Tier 2'
GROUP BY Outlet_Type;
```

*Result*: Evaluates customer satisfaction in Tier 2 locations.

39. **Items with low visibility but high sales.**

```
SELECT
  Item_Identifier,
  Item_Visibility,
  Sales
FROM Blinkit_data
WHERE Item_Visibility < 0.05 AND Sales > 5000;
```

*Result*: Identifies high-performing items with low visibility.

40. **Outlets with low average rating and high average sales.**

```
WITH overall_avg_sales AS (
  SELECT AVG(Sales) AS avg_sales FROM Blinkit_data
)
SELECT
  Outlet_Identifier,
  AVG(Rating) AS avg_rating,
  AVG(Sales) AS avg_sales
FROM Blinkit_data, overall_avg_sales
GROUP BY Outlet_Identifier, avg_sales
HAVING AVG(Rating) < 3 AND AVG(Sales) > avg_sales;
```

*Result*: Highlights outlets with high sales but poor customer satisfaction.

## 3.14 Comparative & Temporal Queries

41. **Compare average sales pre- and post-2010.**

```
SELECT
  CASE
    WHEN Outlet_Establishment_Year < 2010 THEN 'Before 2010'
    ELSE '2010 & After'
```

```
  END AS Year_Group,
  AVG(Sales) AS avg_sales
FROM Blinkit_data
GROUP BY
  CASE
    WHEN Outlet_Establishment_Year < 2010 THEN 'Before 2010'
    ELSE '2010 & After'
  END;
```

*Result*: Compares sales performance of older vs. newer outlets.

42. **Trend of average rating per outlet size by year.**

```
SELECT
  Outlet_Establishment_Year,
  Outlet_Size,
  AVG(Rating) AS avg_rating
FROM Blinkit_data
GROUP BY Outlet_Establishment_Year, Outlet_Size
ORDER BY Outlet_Establishment_Year, Outlet_Size;
```

*Result*: Tracks customer satisfaction trends over time by outlet size.

# 4 Conclusions

This analysis provides a comprehensive view of Blinkit's sales performance, identifying key trends and opportunities for optimization. Key findings include:

- Certain item types and fat content combinations drive higher sales, informing inventory strategies.
- Outlets established in specific years or with specific characteristics (e.g., size, type) show varying performance, guiding expansion decisions.
- Items with high sales but low ratings or visibility present opportunities for marketing or quality improvements.
- Data cleaning efforts, such as normalizing fat content and imputing missing weights, enhance dataset reliability.
- Custom metrics like performance scores and best-seller flags provide actionable insights for product prioritization.

Future analyses could explore additional correlations, incorporate external data, or leverage predictive modeling to forecast sales trends.