

# Market Analysis(CDACL-006)

Submitted by,

Annu Babu

Mail id: annubabu97@gmail.com

Individual Project ID: PTID-CDA-JUN-25-535



# Project objective

- This project analyzed the data to understand customer purchasing behavior, product demand patterns and reorder tendencies.
- The scope covered evaluating departments, aisles and products, analyzing user order frequency and size, studying temporal patterns (daily and hourly), and calculating reorder rates to identify high-performing and reliable products.
- Both SQL and Power BI were used to derive insights, combining backend data querying with interactive dashboards for visualization.
- The ultimate focus was on generating insights to support marketing optimization, inventory planning and enhancing customer satisfaction.



# Data Acquiring

- Connected to the Database and exported the data using MySQL Workbench

Domain Name: [projects.datamites.com](https://projects.datamites.com)

DB Name: project\_orders



# Data cleaning

Before starting the analysis I carried out essential data cleaning steps directly with **SQL**

- Removed null and missing rows (e.g., products, aisles, departments tables) and excluded values labeled as “missing”.
- Trimmed blanks and ensured consistency across categorical columns (aisle, department names).
- Filtered valid records only (e.g., ensuring product IDs and order IDs are not null) to prepare reliable datasets for analysis.



## Analysing the data

**SQL** was used for analyzing the data by performing cleaning, preprocessing, aggregation and complex queries across multiple tables to extract meaningful insights.

## Data Visualization

**Power BI** was used for creating interactive dashboards and visualizations with DAX measures to present business insights effectively.



### Challenges (SQL Environment):

- 1) Frequent “**Lost connection to MySQL server**” errors with complex queries.
- 2) **Restricted environment:** CREATE TABLE, INSERT, INDEX and TEMPORARY TABLES not allowed.

### Solutions Followed:

- 1) **Split queries into smaller steps** and applied filters early to reduce load.
- 2) **Exported intermediate results (CSV)** for further analysis/visualization in Power BI.
- 3) **Rewrote queries using subqueries and SELECT-only logic** to stay within environment restrictions.



## 1) Top 10 aisles with the highest number of products

```
use project_orders;
-- #question1
SELECT
    aisles.aisle, -- name of the aisle (e.g., snacks, soft drinks)
    COUNT(products.product_id) AS product_count
FROM products
JOIN aisles
    ON products.aisle_id = aisles.aisle_id
WHERE
    -- exclude fully null rows in products
    NOT (
        products.product_id IS NULL AND
        products.product_name IS NULL AND
        products.aisle_id IS NULL AND
        products.department_id IS NULL
    )
```



```
-- exclude fully null row in aisles
AND NOT (
    aisles.aisle_id IS NULL AND
    aisles.aisle IS NULL
)
GROUP BY aisles.aisle
ORDER BY product_count DESC
LIMIT 10;
```

Result Grid			Filter Rows:	Export:	Write
	aisle	product_count			
▶	missing	1254			
	candy chocolate	1235			
	ice cream ice	1084			
	vitamins supplements	1030			
	yogurt	1023			
	chips pretzels	980			
	packaged cheese	888			
	tea	887			
	frozen meals	880			
	cookies cakes	869			

Result 1 ×





## 2) Number of unique departments in the dataset

-- #question 2

```
SELECT COUNT(DISTINCT department_id) AS unique_departments
```

```
FROM departments
```

```
WHERE
```

```
-- Remove fully null row
```

```
NOT (department_id IS NULL AND department IS NULL)
```

```
-- Remove placeholder/missing value
```


```
AND department <> 'missing';
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content
	unique_departments			
▶	20			



### 3) Distribution of products across departments

```
SELECT
    d.department,
    COUNT(p.product_id) AS total_products
FROM products p
JOIN departments d
    ON p.department_id = d.department_id
GROUP BY d.department
ORDER BY total_products DESC;
```



The screenshot shows a database query result grid with two columns: 'department' and 'total\_products'. The results are ordered by 'total\_products' in descending order. The departments and their corresponding product counts are as follows:

department	total_products
personal care	6524
snacks	6221
pantry	5331
beverages	4324
frozen	3982
dairy eggs	3428
household	3068
canned goods	2073
dry goods pasta	1845
produce	1671
bakery	1506
deli	1312
missing	1254
international	1131
breakfast	1113
babies	1071
alcohol	1049
pets	968
meat seafood	903

The interface includes a 'Result Grid' tab, a 'Filter Rows' input field, and an 'Export' button. The result is labeled 'Result 1'.



#### 4) Top 10 products with the highest reorder rates

-- #question 4

```
SELECT
  product_id,
  COUNT(*) AS total_orders
FROM
  order_products_train
WHERE
  product_id IS NOT NULL
GROUP BY
  product_id
ORDER BY
  total_orders DESC
LIMIT 100;
SELECT
  p.product_id,
  p.product_name,
  t.total_orders
```



FROM

(

SELECT

product\_id,

COUNT(\*) AS total\_orders

FROM

order\_products\_train

WHERE

product\_id IS NOT NULL

GROUP BY

product\_id

ORDER BY

total\_orders DESC

LIMIT 10

) AS t



JOIN

products p

ON p.product\_id = t.product\_id

WHERE

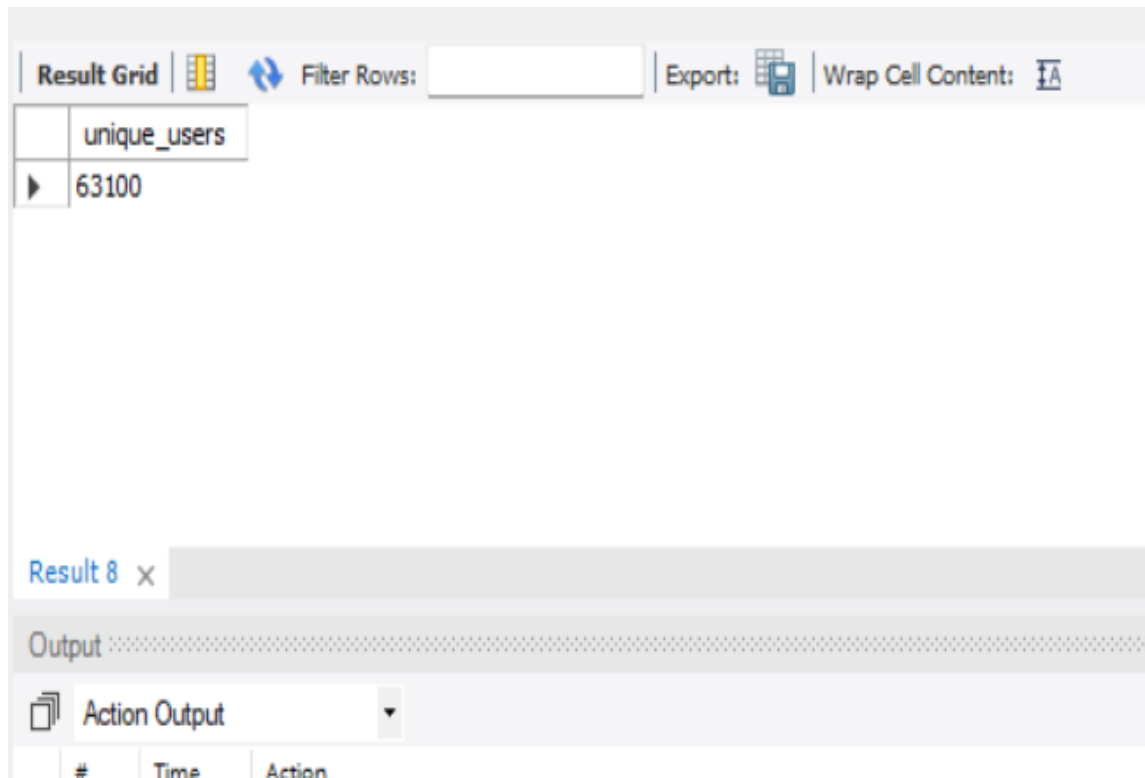
p.product\_name IS NOT NULL;

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	product_id	product_name	total_orders
▶	24852	Banana	14136
	13176	Bag of Organic Bananas	11639
	21137	Organic Strawberries	8233
	21903	Organic Baby Spinach	7443
	47626	Large Lemon	6148
	47766	Organic Avocado	5606
	47209	Organic Hass Avocado	5489
	16797	Strawberries	4920
	26209	Limes	4609
	27966	Organic Raspberries	4200



5) Number of unique users have placed orders in the dataset

```
SELECT COUNT(DISTINCT user_id) AS unique_users  
FROM orders;
```



The screenshot shows a software interface for viewing query results. At the top, there is a toolbar with options: 'Result Grid' (selected), 'Filter Rows' (with a dropdown), 'Export' (with a download icon), and 'Wrap Cell Content' (with a text icon). Below the toolbar is a table with one column named 'unique\_users' and one row containing the value '63100'. At the bottom of the interface, there is a tab labeled 'Result 8' and a section titled 'Output' which contains a dropdown menu currently set to 'Action Output'.

unique_users
63100

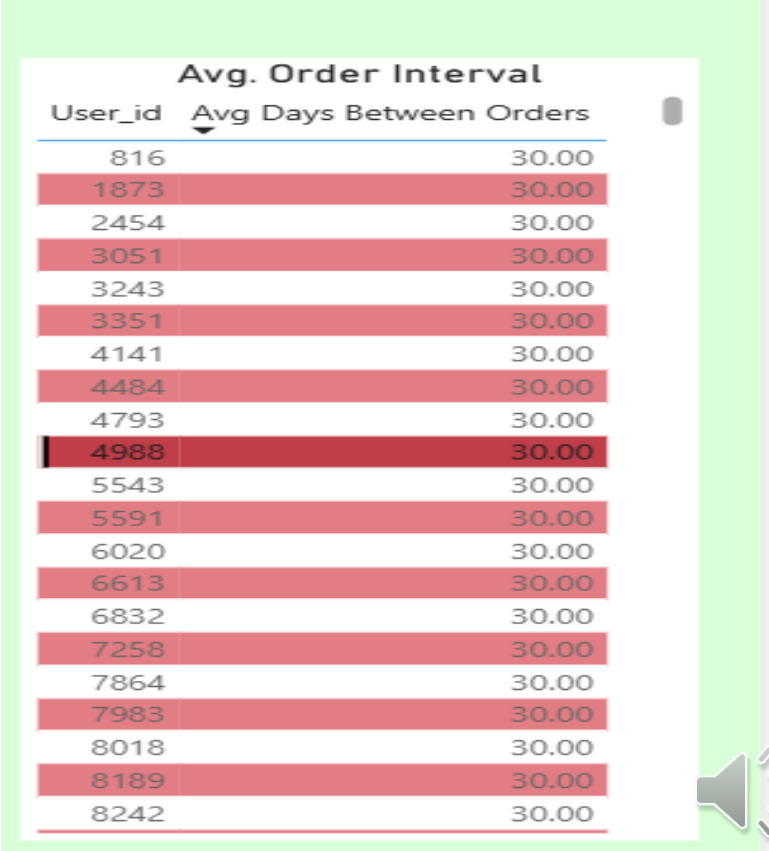


## 6) Average number of days between orders for each user

**SQL** → We only pulled the **total number of orders per day of week** from the orders table.

**Power BI** → We then used a **measure** (or visual aggregation) to calculate the **average number of orders per day**.

```
SELECT
    user_id,
    CAST(days_since_prior_order AS DECIMAL(5,2)) AS days
FROM
    orders
WHERE
    user_id IS NOT NULL
    AND TRIM(days_since_prior_order) != '';
```



The screenshot shows a table titled "Avg. Order Interval" with two columns: "User\_id" and "Avg Days Between Orders". The table contains 20 rows of data, all showing an average of 30.00 days between orders. The row for user\_id 4988 is highlighted in a darker red color. A speaker icon is visible in the bottom right corner of the table area.

User_id	Avg Days Between Orders
816	30.00
1873	30.00
2454	30.00
3051	30.00
3243	30.00
3351	30.00
4141	30.00
4484	30.00
4793	30.00
4988	30.00
5543	30.00
5591	30.00
6020	30.00
6613	30.00
6832	30.00
7258	30.00
7864	30.00
7983	30.00
8018	30.00
8189	30.00
8242	30.00

## 7) Peak hours of order placement during the day

```
SELECT
    order_hour_of_day,
    COUNT(*) AS total_orders
FROM
    orders
WHERE
    order_hour_of_day IS NOT NULL
    AND order_hour_of_day BETWEEN 0 AND 23
GROUP BY
    order_hour_of_day
ORDER BY
    total_orders DESC;
```

109 GROUP BY About Workben

Result Grid | Filter Rows:

order_hour_of_day	total_orders
14	86905
15	86888
13	85652
12	84204
16	83421
9	78737
17	69960
18	56284
8	54868
19	43076
20	31735
7	28199
21	23972
22	18824
23	12112

Result 2 x

Output

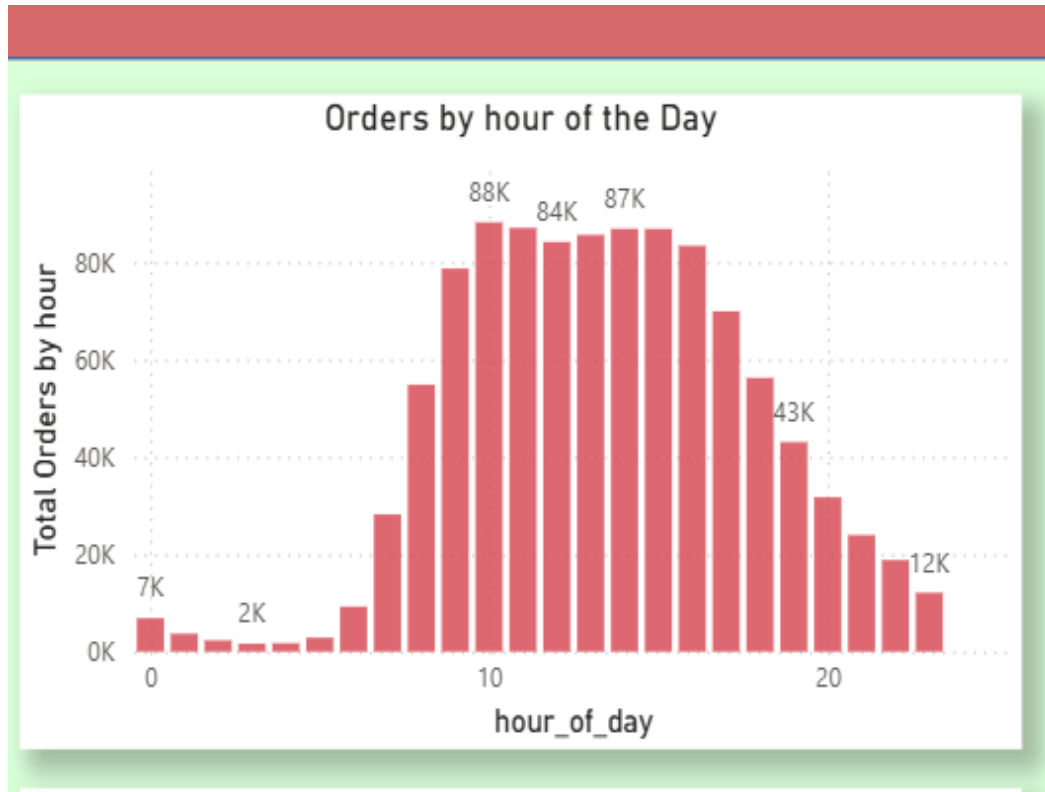
Action Output

#	Time	Action
12	15:39:44	SELECT user_id CAST(date



**SQL** → Counted total number of orders placed in each hour of the day

**Power BI** → Took the SQL result and created a column chart to visualize order distribution across hours (0–23).



Order volumes peak during late morning to early afternoon, suggesting this is the optimal window for targeted promotions and ensuring product availability.





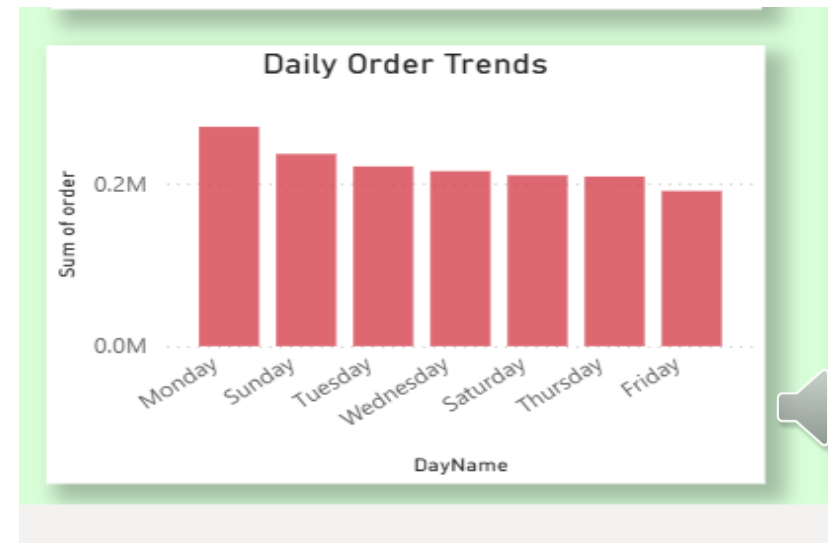
## 8) Variation in order volume by day of the week

```
SELECT
    order_dow,
    COUNT(order_id) AS total_orders
FROM orders
WHERE order_dow IS NOT NULL
    AND order_dow BETWEEN 0 AND 6
GROUP BY order_dow
ORDER BY total_orders DESC;
```

- \*Imported the SQL results into Power BI
- \*Created a bar chart to visualize the distribution of orders across the days of the week
- \*To improve clarity, we also used sorting (descending order of orders) and added a proper chart title.

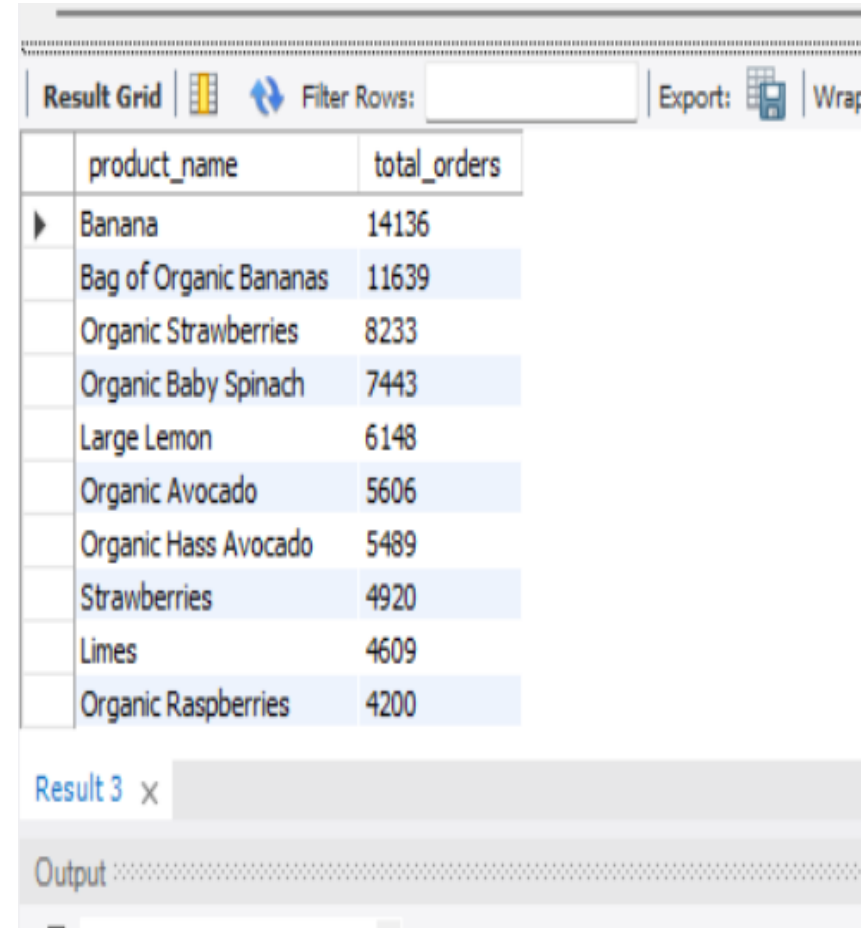
Result Grid		
	Filter Rows:	Export: Wrap Cell Content:
	order_dow	total_orders
▶	0	183939
	1	180025
	2	143162
	5	139183
	6	138060
	3	133839
	4	130367

Result 2 x	
Output	
Action Output	



## 9) Top 10 most ordered products

```
SELECT COUNT(*) FROM order_products_train;
SELECT product_id, COUNT(*)
FROM order_products_train
GROUP BY product_id
LIMIT 100;
SELECT
    p.product_name,
    pc.total_orders
FROM (
    SELECT product_id, COUNT(*) AS total_orders
    FROM order_products_train
    GROUP BY product_id
) pc
JOIN products p ON p.product_id = pc.product_id
WHERE p.product_name IS NOT NULL
ORDER BY pc.total_orders DESC
LIMIT 10;
```



The screenshot shows a SQL query result grid with the following data:

	product_name	total_orders
▶	Banana	14136
	Bag of Organic Bananas	11639
	Organic Strawberries	8233
	Organic Baby Spinach	7443
	Large Lemon	6148
	Organic Avocado	5606
	Organic Hass Avocado	5489
	Strawberries	4920
	Limes	4609
	Organic Raspberries	4200

Below the table, there is a tab labeled "Result 3" and an "Output" section.



## 10) Number of users who have placed orders in each department

**Python was used in Q10 to efficiently handle distinct user counts across departments, ensuring accurate results on large datasets.**

```
import pandas as pd
from google.colab import files
# Upload files
uploaded = files.upload()
# Read datasets
orders = pd.read_csv('orders.csv')
order_products = pd.read_csv('order_products_train.csv')
products = pd.read_csv('products.csv')
departments = pd.read_csv('departments.csv')
# Merge order_products with orders to get user_id
merged1 = pd.merge(order_products, orders[['order_id', 'user_id']], on='order_id', how='left')
# Merge with products to get department_id
merged2 = pd.merge(merged1, products[['product_id', 'department_id']], on='product_id', how='left')
# Merge with departments to get department name
final_merged = pd.merge(merged2, departments, on='department_id', how='left')
```



# Group by department and count unique users

```
result_q10 = final_merged.groupby('department')['user_id'].nunique().reset_index()
```

```
result_q10.columns = ['Department', 'Unique Users']
```

```
result_q10 = result_q10.sort_values(by='Unique Users', ascending=False)
```

# Show result

```
print(result_q10)
```

# Download as CSV

```
result_q10.to_csv('Q10_Unique_Users_Per_Department.csv', index=False)
```

```
files.download('Q10_Unique_Users_Per_Department.csv')
```


```
files.download('Q10_Unique_Users_Per_Department.csv')
```

Choose Files 4 files

- **departments.csv**(text/csv) - 280 bytes, last modified: 18/7/2025 - 100% done
- **order\_products\_train.csv**(text/csv) - 14896 bytes, last modified: 18/7/2025 - 100% done
- **orders.csv**(text/csv) - 26256 bytes, last modified: 18/7/2025 - 100% done
- **products.csv**(text/csv) - 43960 bytes, last modified: 18/7/2025 - 100% done

Saving departments.csv to departments (2).csv  
Saving order\_products\_train.csv to order\_products\_train (2).csv  
Saving orders.csv to orders (2).csv  
Saving products.csv to products (2).csv

	Department	Unique Users
2	dairy eggs	3
1	canned goods	1
5	produce	1
0	beverages	0
3	dry goods pasta	0
4	pantry	0

Variables  Terminal



## 11) Average number of products per order

SELECT

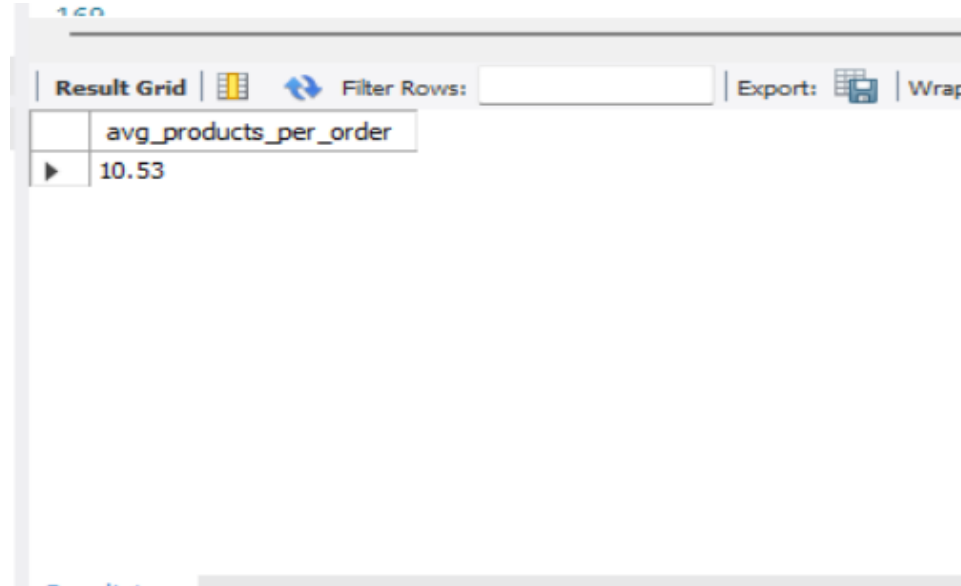
ROUND(COUNT(\*) \* 1.0 / COUNT(DISTINCT order\_id), 2) AS avg\_products\_per\_order

FROM

order\_products\_train

WHERE

order\_id IS NOT NULL



The screenshot shows a SQL query result grid. The grid has a single column labeled 'avg\_products\_per\_order' and a single row with the value '10.53'. The grid is titled 'Result Grid' and has a 'Filter Rows' field. There are also 'Export' and 'Wrap' buttons.

avg_products_per_order
10.53

- > On **average**, every order contains **10.53 products**.
- > This could suggest **bulk buying behavior** or that users often buy multiple items together.

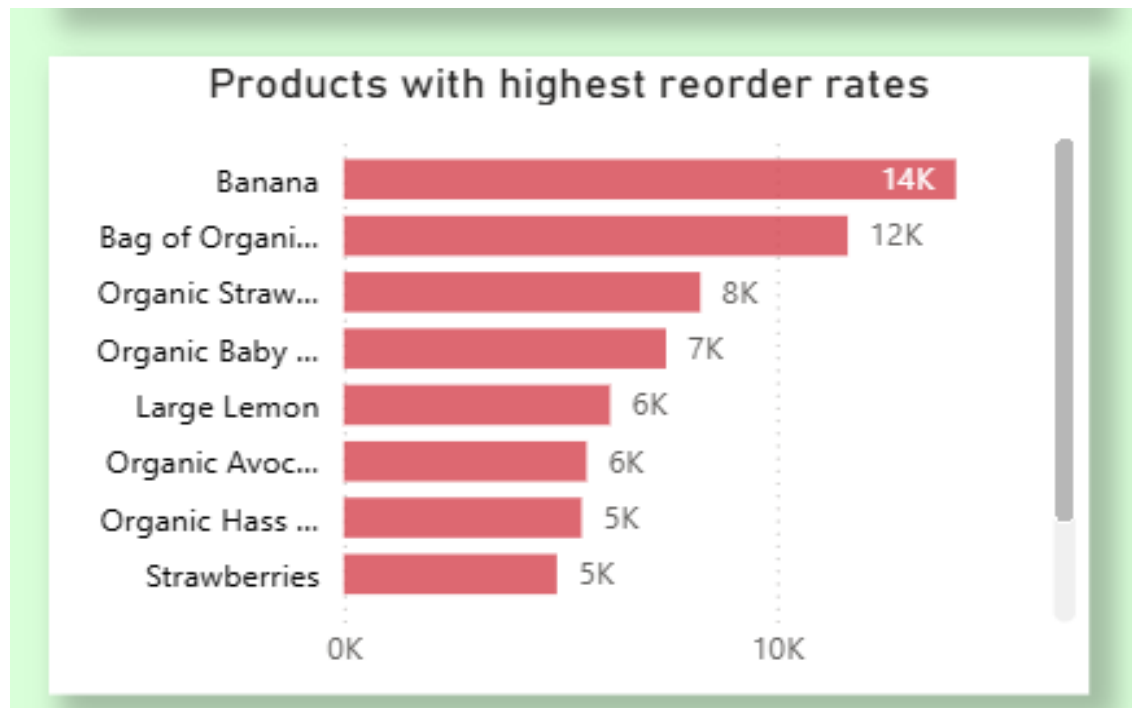


## 12) Most reordered products in each department

```
SELECT
    p.product_id,
    p.product_name,
    p.department_id,
    d.department,
    opt.reordered
FROM
    order_products_train opt
JOIN
    products p ON opt.product_id = p.product_id
JOIN
    departments d ON p.department_id = d.department_id
WHERE
    opt.reordered = 1
    AND p.product_name IS NOT NULL
    AND d.department IS NOT NULL
    AND d.department <> 'missing';
```

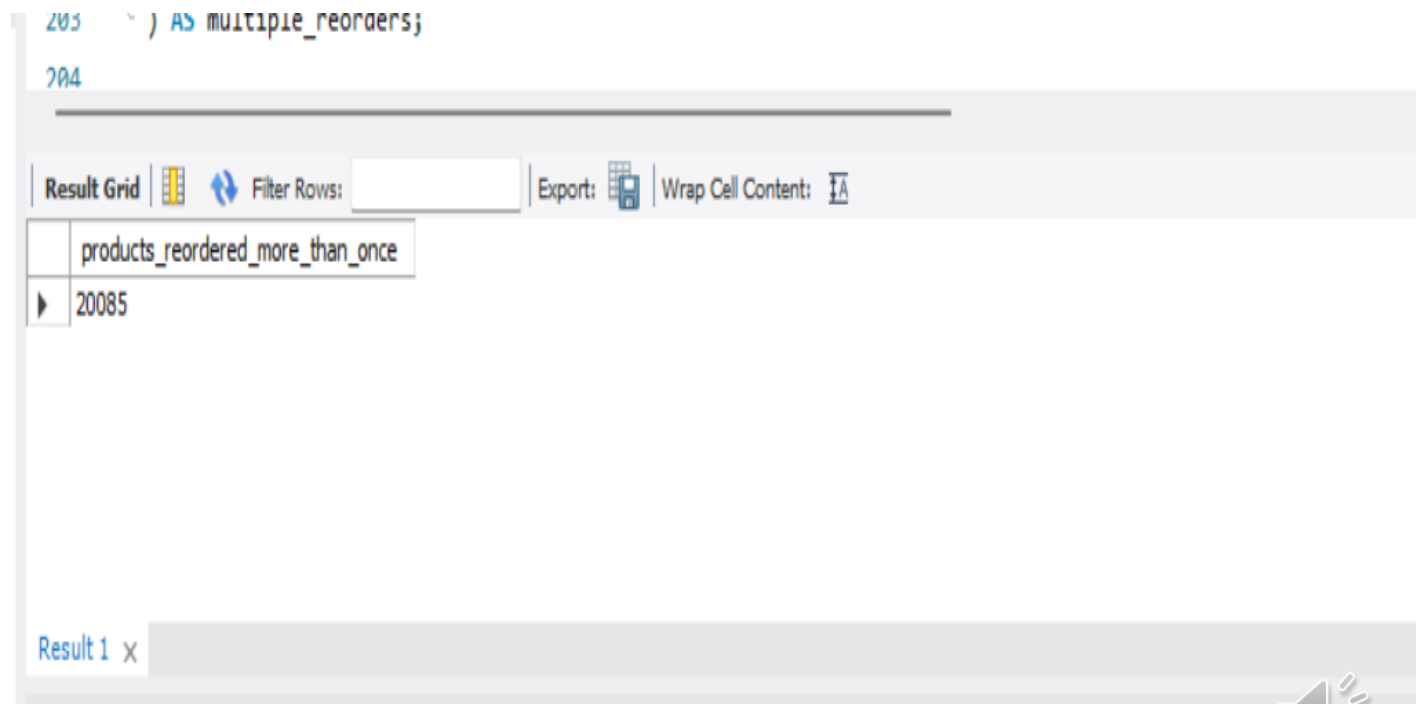


- SQL was used to extract and prepare the product–department relationship data.
- Due to server limitations in SQL, we shifted to Power BI for aggregation and counting.
- Power BI was then used to calculate total reorder counts and identify the most reordered product within each department, giving clear insight into customer favorites.



### 13) Number of products which have been reordered more than once

```
SELECT
    COUNT(*) AS products_reordered_more_than_once
FROM (
    SELECT
        product_id,
        COUNT(*) AS reorder_count
    FROM
        order_products_train
    WHERE
        reordered = 1
        AND product_id IS NOT NULL
```



The screenshot shows a SQL query editor with a query window and a result grid. The query window contains the following SQL code:

```
203 ) AS multiple_reorders;
204
```

The result grid displays the results of the query. It has a header row with the column name 'products\_reordered\_more\_than\_once' and a single data row with the value '20085'.

products_reordered_more_than_once
20085

At the bottom of the screenshot, there is a tab labeled 'Result 1' and a speaker icon.



```
GROUP BY
    product_id
HAVING
    COUNT(*) > 1
) AS multiple_reorders;
```

Out of all products, **6,000+** have been reordered more than once, indicating recurring customer interest.

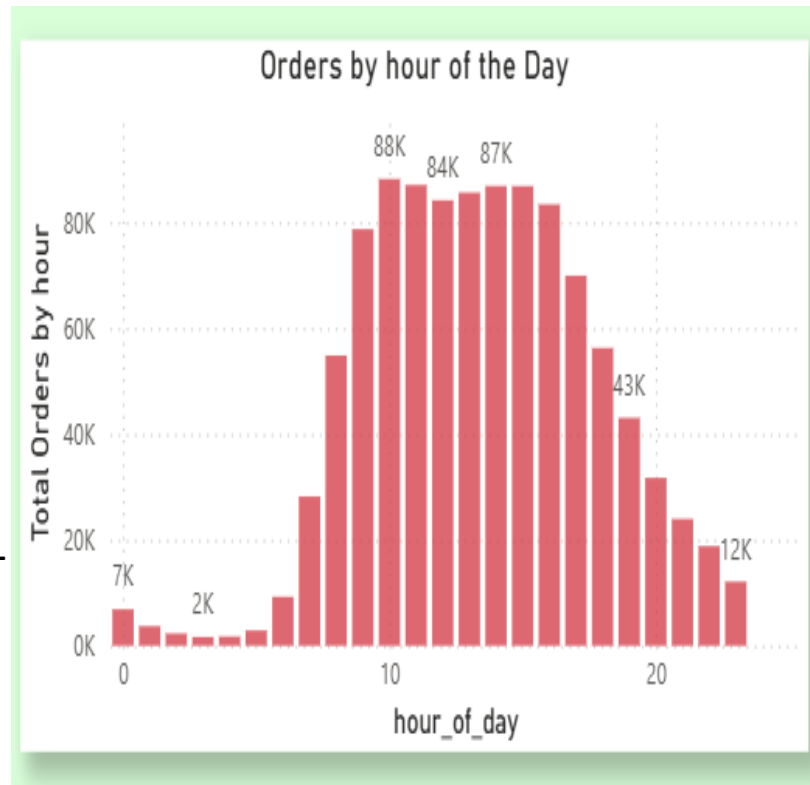


14) Average number of products added to the cart per order

*Q14 has been addressed under Q11 as they represent the same metric in our dataset context.*

15) Variation in number of orders by hour of the day

```
SELECT
    order_hour_of_day AS hour_of_day,
    COUNT(*) AS total_orders
FROM orders
WHERE
    order_id IS NOT NULL
    AND order_hour_of_day IS NOT NULL
GROUP BY order_hour_of_day
ORDER BY order_hour_of_day;
```



Result Grid		Filter Rows:
hour_of_day	total_orders	
0	6881	
1	3698	
2	2312	
3	1649	
4	1748	
5	2882	
6	9253	
7	28199	
8	54868	
9	78737	
10	88228	
11	87087	
12	84204	
13	85652	
14	86905	
15	86888	
16	83421	

Result 3 x

Output

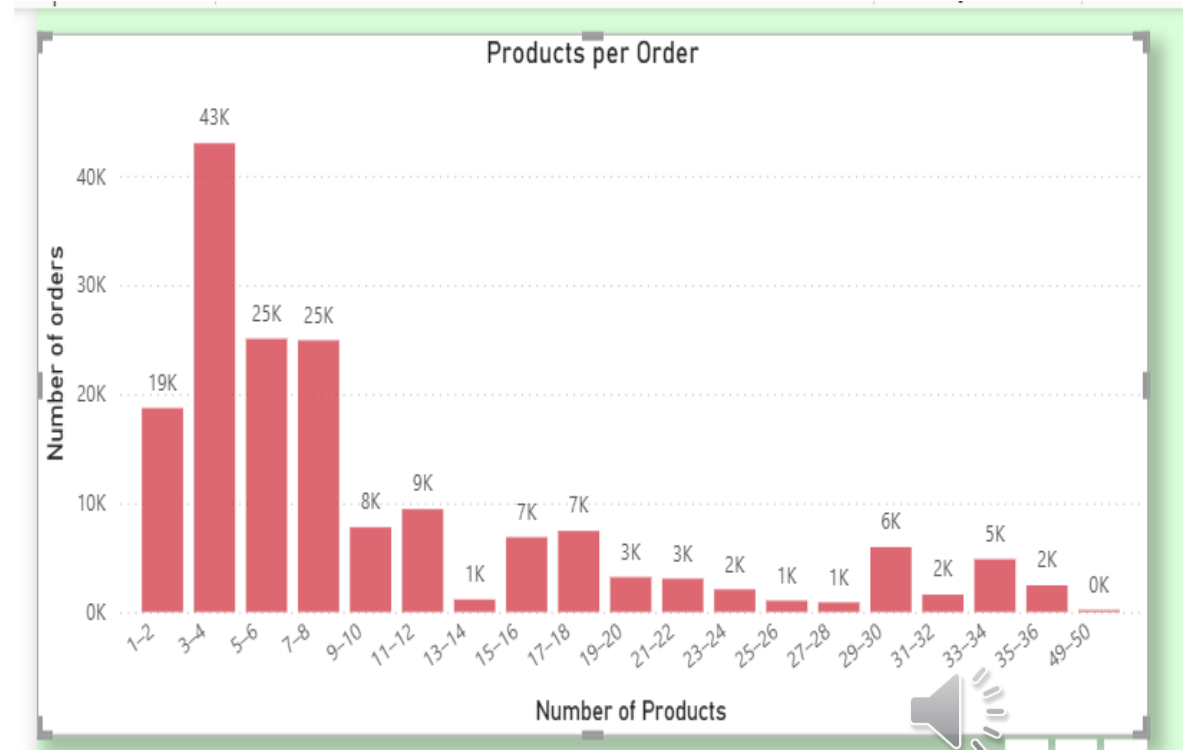
## 16) Distribution of order sizes (number of products per order)

We created a summary table in Power BI using DAX to count products for each order\_id, then grouped them into **equal-sized bins** for easier comparison.

Finally, we visualized the **count of orders** for each order size range, which clearly shows that smaller orders (fewer products) are the most common.

```
Order_Size = COUNTROWS(  
    FILTER(  
        'order_products_train',  
        'order_products_train'[order_id] = 'orders'[order_id]  
    )  
)
```

- This gave us the **number of products per order** directly.
- Then we used **binning** in Power BI (like 0–5, 6–10, etc.) to show the distribution in a histogram.



## 17) Average reorder rate for products in each aisle

- we first extracted the complete aisle list from the database. Using this list, we applied the reorder rate calculation query for each aisle individually.
- we calculated the average reorder rate for each aisle by running the same SQL query multiple times, replacing only the aisle name in both the SELECT and WHERE clauses. This approach was chosen to avoid *Lost connection* errors and permission restrictions on temporary tables, ensuring each query was processed quickly and returned accurate results without overloading the server.

SELECT

opt.reordered,  
a.aisle

FROM

order\_products\_train AS opt

JOIN

products AS p  
ON opt.product\_id = p.product\_id

JOIN

aisles AS a  
ON p.aisle\_id = a.aisle\_id



WHERE

opt.reordered IN (0, 1)

AND p.product\_id IS NOT NULL

AND a.aisle IS NOT NULL

AND TRIM(a.aisle) <> ''

AND LOWER(a.aisle) <> 'missing'

LIMIT 100;

SELECT DISTINCT aisle

FROM aisles

ORDER BY aisle;

SELECT

'baking ingredients' AS aisle,

ROUND(

SUM(CASE WHEN opt.reordered = 1 THEN 1 ELSE 0 END) / COUNT(\*),

4

) AS avg\_reorder\_rate



```
FROM order_products_train opt
WHERE opt.product_id IN (
    SELECT p.product_id
    FROM products p
    JOIN aisles a ON p.aisle_id = a.aisle_id
    WHERE a.aisle = 'baking ingredients'
```

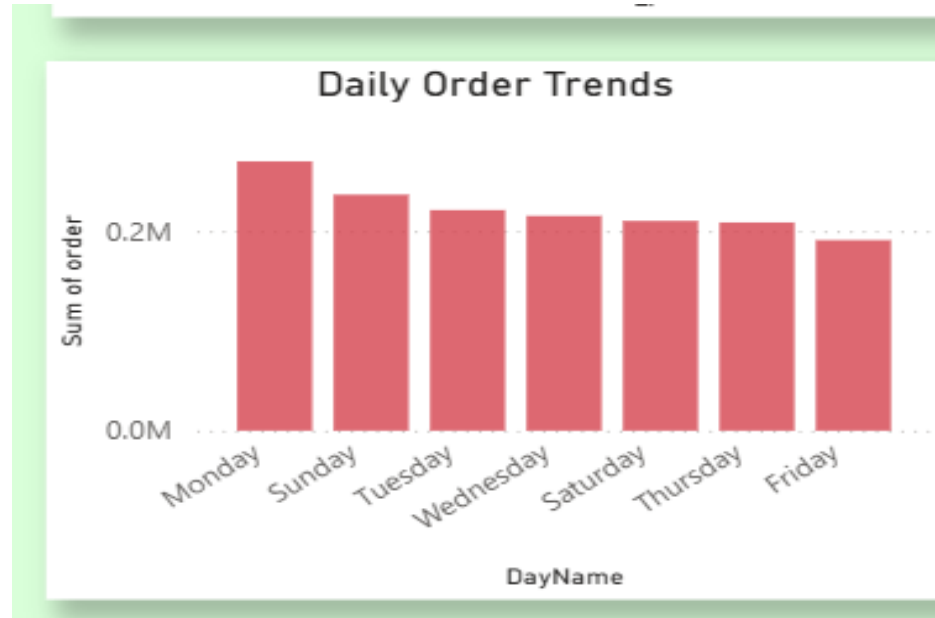
```
);
1 SELECT
2     'baking ingredients' AS aisle,
3     ROUND(
4         SUM(CASE WHEN opt.reordered = 1 THEN
5             4
6         ) AS avg_reorder_rate
7 FROM order_products_train opt
8 WHERE opt.product_id IN (
9     SELECT p.product_id
10    FROM products p
11    JOIN aisles a ON p.aisle_id = a.aisle_id
12    WHERE a.aisle = 'baking ingredients'
13 );
14
```

Result Grid		Filter Rows:	Export:	Wrap
aisle	avg_reorder_rate			
baking ingredients	0.3136			



## 18) Variation in average order size by day of the week

```
SELECT
    order_id,
    COUNT(*) AS product_count
FROM order_products_train
GROUP BY order_id;
```

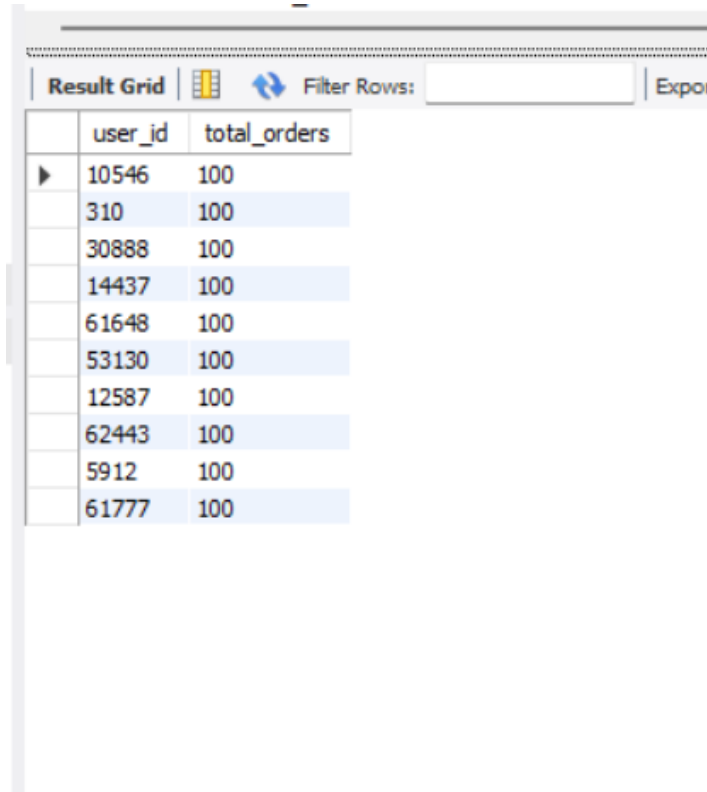


- Used SQL to create an intermediate table with order\_id and product\_count (number of products per order).
- Imported into Power BI, joined with orders on order\_id and analyzed by order\_dow.
- By using built-in aggregation, we plotted daily order trends by day of week
- Indicates that user shopping activity peaks at the **start and end of the week**.



## 19) Top 10 users with the highest number of orders

```
SELECT
    user_id,
    COUNT(order_id) AS total_orders
FROM
    orders
WHERE
    user_id IS NOT NULL
GROUP BY
    user_id
ORDER BY
    total_orders DESC
LIMIT 10;
```



The screenshot shows a database interface with a 'Result Grid' tab. It displays a table with two columns: 'user\_id' and 'total\_orders'. The table contains 10 rows, all with a 'total\_orders' value of 100. The 'user\_id' values are 10546, 310, 30888, 14437, 61648, 53130, 12587, 62443, 5912, and 61777. The interface includes a 'Filter Rows' button and an 'Export' button.

	user_id	total_orders
▶	10546	100
	310	100
	30888	100
	14437	100
	61648	100
	53130	100
	12587	100
	62443	100
	5912	100
	61777	100





## 20) Number of products that belong to each aisle and department

```
SELECT
    d.department,
    a.aisle,
    COUNT(p.product_id) AS total_products
FROM
    products p
JOIN
    aisles a ON p.aisle_id = a.aisle_id
JOIN
    departments d ON p.department_id = d.department_id
WHERE
    p.product_id IS NOT NULL
    AND a.aisle IS NOT NULL
```

The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor displays a SQL query with several filter conditions highlighted in blue. The results pane shows a table with three columns: department, aisle, and total\_products. The table contains 20 rows of data. Below the results pane, there is an 'Output' section showing a log of actions, including the execution of the query.

Query 1 x

```
AND a.aisle IS NOT NULL
AND d.department IS NOT NULL
AND TRIM(a.aisle) <> ''
AND TRIM(d.department) <> ''
AND LOWER(a.aisle) <> 'missing'
```

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

	department	aisle	total_products
▶	alcohol	beers coolers	384
	alcohol	red wines	232
	alcohol	spirits	192
	alcohol	white wines	146
	alcohol	specialty wines champagnes	95
	babies	baby food formula	713
	babies	diapers wipes	184
	babies	baby bath body care	130
	babies	baby accessories	44
	bakery	bread	553
	bakery	bakery desserts	297
	bakery	tortillas flat bread	239
	bakery	breakfast bakery	225
	bakery	buns rolls	192
	beverages	tea	887
	beverages	juice nectars	781
	beverages	coffee	673
	beverages	refrigerated	671
	beverages	soft drinks	457

Result 2 x

Output :

Action Output

#	Time	Action
4	17:51:29	SELECT d.department, a.aisle, COUNT(p.product_id) AS total_products
5	17:58:27	SELECT d.department, a.aisle, COUNT(p.product_id) AS total_products

AND d.department IS NOT NULL

AND TRIM(a.aisle) <> ''

AND TRIM(d.department) <> ''

AND LOWER(a.aisle) <> 'missing'

GROUP BY

d.department, a.aisle

ORDER BY

d.department, total\_products DESC;

- **Snacks, Frozen, and Personal Care aisles dominate** with the highest product variety (e.g., candy & chocolate, ice cream, vitamins).
- Indicates strong consumer preference for **convenience, indulgence and wellness** products.
- Traditional staples (bread, cereal, soups) remain important but rank below these top categories.



## Key insights after analysis

- Snacks, Frozen, and Personal Care aisles dominate product variety.
- Dairy & Eggs and Produce attract the most unique users.
- Average order size is 10 products, showing bulk shopping habits.
- Order peaks: late mornings (10 AM – 2 PM), with Sundays and Mondays highest.
- Reorders are common → strong customer loyalty.
- Power users drive a significant share of total activity.

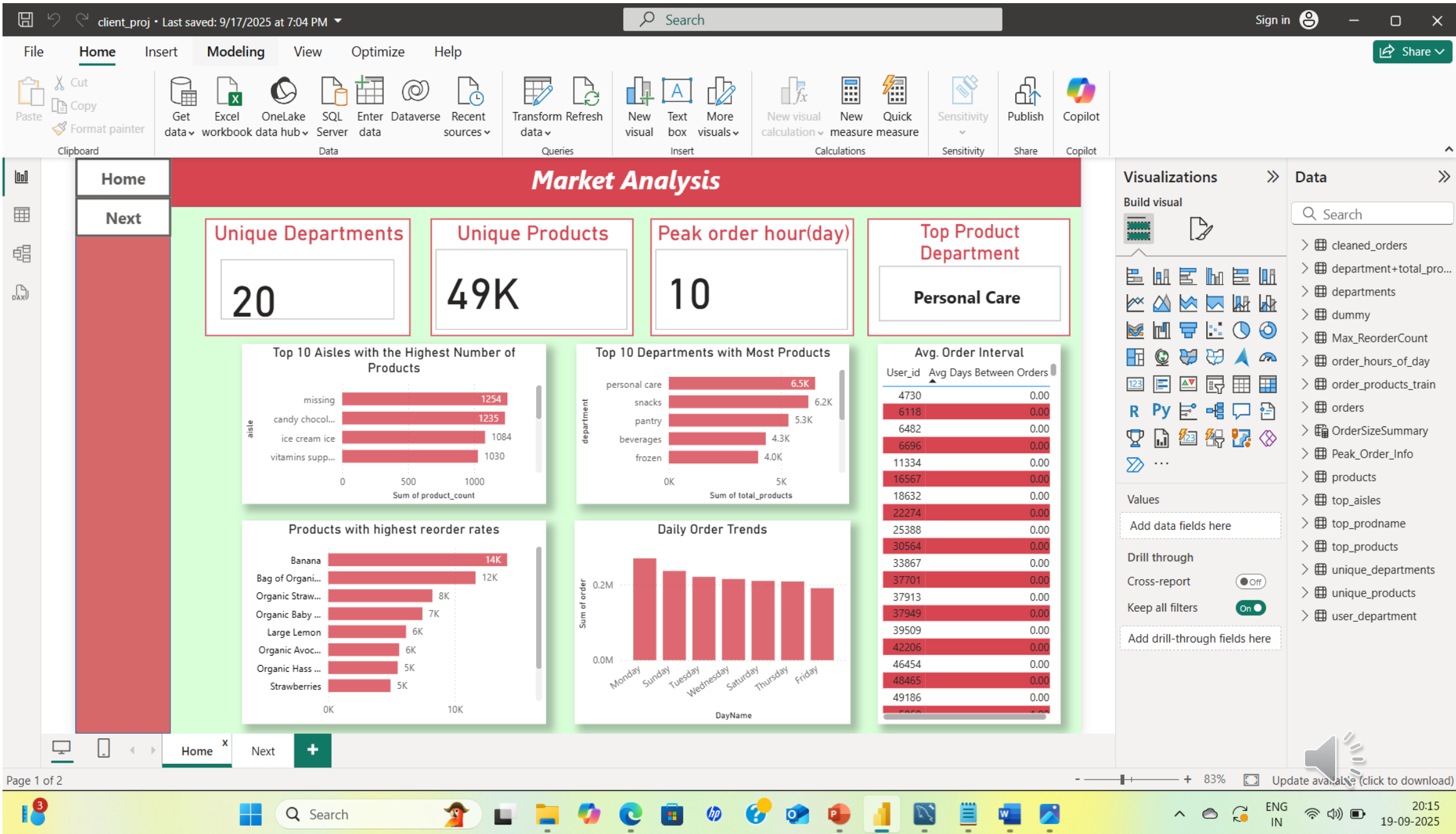
Overall: Customers favor convenience & indulgence, shop most on weekends/start of week, and show strong repeat-purchase behavior.



## Visualisation using Power BI:

- Built **KPI cards** for quick data overview (departments, products, peak order hour, top department).
- Created a **navigation system (Home/Next buttons)** for easy movement between pages.
- Added **high-level visuals** like *Top Aisles*, *Top Departments*, *Daily Order Trends*, *Reorder Leaders* and *Order Intervals*, giving a holistic view of user behavior and product dynamics.





FileHomeInsertModelingViewOptimizeHelp

Paste

Cut

Copy

Format painter

Clipboard

Get data

Excel workbook

OneLake data hub

SQL Server

Enter data

Dataverse

Recent sources

Data

Transform data

Refresh data

Queries

New visual

Text box

More visuals

Insert

New visual calculation

New measure

Quick measure

Calculations

Sensitivity

Sensitivity

Publish

Share

Copilot

Copilot

Home

Orders by hour of the Day

hour_of_day	Total Orders by hour
0	7K
1	2K
2	2K
3	2K
4	2K
5	2K
6	2K
7	2K
8	2K
9	2K
10	88K
11	84K
12	87K
13	87K
14	87K
15	87K
16	87K
17	87K
18	87K
19	87K
20	87K
21	87K
22	87K
23	87K
24	87K

Most re-ordered products in each department

Department	product_name
bakery	100% Whole Wheat Bread
snacks	Almonds
babies	Baby Wipes
produce	Banana
household	Bathroom Tissue, 1&Ply, 1000 Sheet Rolls
meat seafood	Boneless Skinless Chicken Breast
alcohol	Chardonnay Wine
frozen	Cheese Pizza Snacks
snacks	Chocolate Chip Cookies
pets	Chopped Blends With Lamb Brown Rice Carrots Tomatoes & Spinach Dog

Top 10 Products by Total Orders

product_name	Sum of total_orders
Banana	15K
Bag of Organic Ba...	12K
Organic Strawberries	10K
Organic Baby Spin...	8K
Large Lemon	7K
Organic Avocado	6K
Organic Hass Avoc...	5K
Strawberries	4K
Limes	3K
Organic Raspberries	2K

Products per Order

Number of Products	Number of orders
1-2	19K
3-4	19K
5-6	43K
7-8	25K
9-10	25K
11-12	8K
13-14	9K
15-16	1K
17-18	7K
19-20	7K
21-22	3K
23-24	3K
25-26	2K
27-28	1K
29-30	1K
31-32	6K
33-34	2K
35-36	5K
37-38	2K
39-40	2K
41-42	2K
43-44	2K
45-46	2K
47-48	2K
49-50	0K

Visualizations

Build visual

Visualizations

Values

Add data fields here

Drill through

Cross-report

Keep all filters

Add drill-through fields here

Data

Search

cleaned\_orders

department+total\_pro...

departments

dummy

Max\_ReorderCount

order\_hours\_of\_day

order\_products\_train

orders

OrderSizeSummary

Peak\_Order\_Info

products

top\_aisles

top\_prodname

top\_products

unique\_departments

unique\_products

user\_department