DATAMITES INTERNSHIP

PURCHASE PATTERN ANALYTICS

Market Basket Analysis (MBA) – Company Project Report

Project ID: **CDACL005 - Purchase Pattern Analytics**

Project Team ID: **PTID-CDA-SEP-25-1036**

Project By**: RAKSHITA SHANKRAPPA APPAJI**

Database Management System (DBMS)**: SQL SERVER**

Language**: SQL, Python**

Tool**: Power BI**

This professionally enhanced report now contains detailed explanations below every SQL query screenshot for better understanding and presentation quality.

TABLE OF CONTENTS

Introduction

Market Basket Analysis is a data analysis technique used to understand customer purchasing behaviour by identifying relationships between products bought together. In today's data-driven retail environment, analysing transactional data helps businesses improve sales strategies, product placement, and customer experience.

This project analyses purchase transaction data using SQL for data cleaning and analysis, Power BI for interactive visualization, and Python for implementing the Apriori algorithm. The objective is to uncover frequent item sets and association rules that reveal hidden buying patterns and support effective business decision-making.

Problem Statement

Retail businesses generate large volumes of transactional data every day, but they often struggle to identify which products are frequently purchased together. Manual analysis of such large datasets is time-consuming, inefficient, and prone to errors. Without data-driven insights, businesses miss opportunities for effective product bundling, cross-selling, and revenue optimization. This project addresses the need for an automated analytical approach to uncover purchasing patterns and support informed business decision-making.

Project Overview

This project aims to analyse customer purchase behaviour and identify product combinations that are frequently bought together using Market Basket Analysis. By leveraging SQL for data cleaning and analysis, Power BI for interactive dashboards, and the Apriori algorithm in Python for association rule mining, the project transforms raw transactional data into actionable business insights.

The analysis helps retailers understand buying patterns, optimize product placement, design effective bundling and cross-selling strategies, and improve overall sales performance through data-driven decision-making.

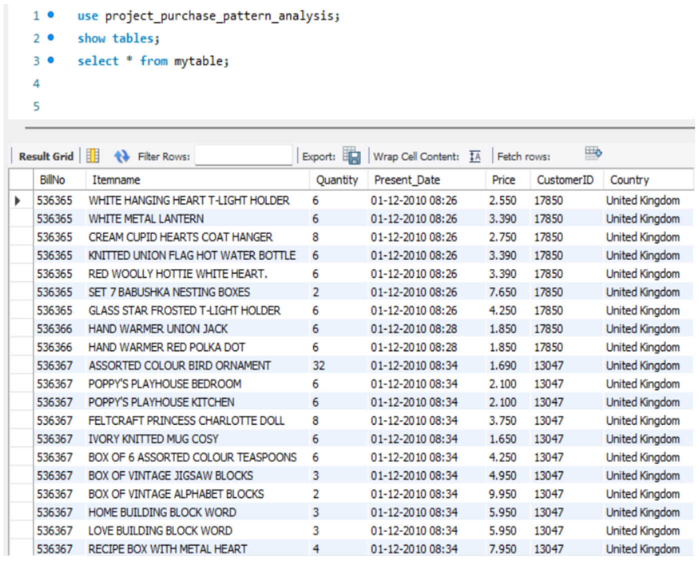
Dataset Description

Table Structure

Column Name	**Description (Business Meaning)**
BillNo	Unique transaction or invoice number
Itemname	Name of the product purchased
Quantity	Number of units bought in the transaction
Price	Price per unit of the product
CustomerID	Unique identifier for each customer
Country	Country where the purchase occurred
Present_Date *(if present)*	Date of transaction

|. ACCESSED SQL SERVER DATABASE:

```
use project_purchase_pattern_analysis;

show tables;

select * from mytable;
```



```
use project_purchase_pattern_analysis;
show tables;
select * from mytable;
```

BillNo	Itemname	Quantity	Present_Date	Price	CustomerID	Country
536365	WHITE HANGING HEART T-LIGHT HOLDER	6	01-12-2010 08:26	2.550	17850	United Kingdom
536365	WHITE METAL LANTERN	6	01-12-2010 08:26	3.390	17850	United Kingdom
536365	CREAM CUPID HEARTS COAT HANGER	8	01-12-2010 08:26	2.750	17850	United Kingdom
536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-12-2010 08:26	3.390	17850	United Kingdom
536365	RED WOOLLY HOTTIE WHITE HEART.	6	01-12-2010 08:26	3.390	17850	United Kingdom
536365	SET 7 BABUSHKA NESTING BOXES	2	01-12-2010 08:26	7.650	17850	United Kingdom
536365	GLASS STAR FROSTED T-LIGHT HOLDER	6	01-12-2010 08:26	4.250	17850	United Kingdom
536366	HAND WARMER UNION JACK	6	01-12-2010 08:28	1.850	17850	United Kingdom
536366	HAND WARMER RED POLKA DOT	6	01-12-2010 08:28	1.850	17850	United Kingdom
536367	ASSORTED COLOUR BIRD ORNAMENT	32	01-12-2010 08:34	1.690	13047	United Kingdom
536367	POPPY'S PLAYHOUSE BEDROOM	6	01-12-2010 08:34	2.100	13047	United Kingdom
536367	POPPY'S PLAYHOUSE KITCHEN	6	01-12-2010 08:34	2.100	13047	United Kingdom
536367	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	01-12-2010 08:34	3.750	13047	United Kingdom
536367	IVORY KNITTED MUG COSY	6	01-12-2010 08:34	1.650	13047	United Kingdom
536367	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	01-12-2010 08:34	4.250	13047	United Kingdom
536367	BOX OF VINTAGE JIGSAW BLOCKS	3	01-12-2010 08:34	4.950	13047	United Kingdom
536367	BOX OF VINTAGE ALPHABET BLOCKS	2	01-12-2010 08:34	9.950	13047	United Kingdom
536367	HOME BUILDING BLOCK WORD	3	01-12-2010 08:34	5.950	13047	United Kingdom
536367	LOVE BUILDING BLOCK WORD	3	01-12-2010 08:34	5.950	13047	United Kingdom
536367	RECIPE BOX WITH METAL HEART	4	01-12-2010 08:34	7.950	13047	United Kingdom

It shows the available database as **‘project_ purchase_pattern_analysis’** and the table present within the database as **‘mytable’.**

||. SQL-Based Exploratory Data Analysis & Business Queries

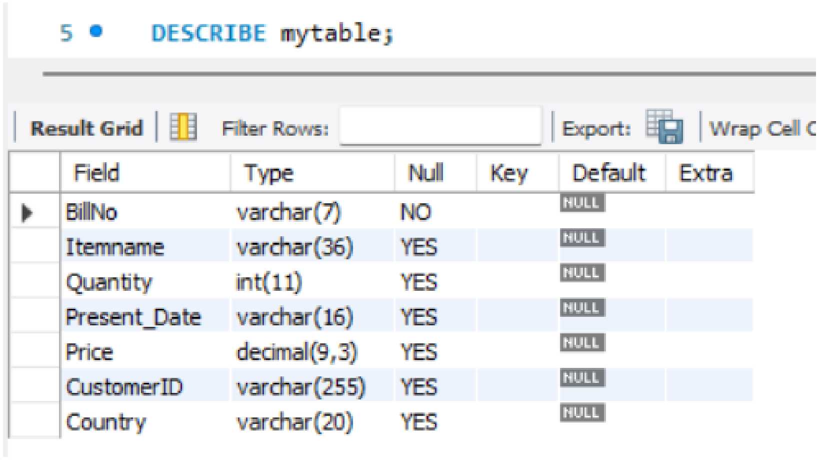
Database Initialization and Data Review

The analysis begins by selecting the project database and reviewing the available tables. The transaction table is explored to understand its structure, column names, and data types. This step ensures familiarity with the dataset before performing analytical operations.

A full table preview and schema description are performed to validate data completeness and field definitions.

DESCRIBE mytable;

It shows the available fields, datatypes of the fields and also it shows whether there is null values present in the data This query is used to examine the structure of the transaction This query is used to examine the structure of the transaction table. It provides details about each column, including column names, data types, and whether null values are allowed.



```
DESCRIBE mytable;
```

Field	Type	Null	Key	Default	Extra
BillNo	varchar(7)	NO		NULL	
Itemname	varchar(36)	YES		NULL	
Quantity	int(11)	YES		NULL	
Present_Date	varchar(16)	YES		NULL	
Price	decimal(9,3)	YES		NULL	
CustomerID	varchar(255)	YES		NULL	
Country	varchar(20)	YES		NULL	

Business Purpose:
Understanding the table schema ensures correct interpretation of fields such as transaction ID, product name, quantity, price, customer ID, and date before performing analysis.

Why it is Important:

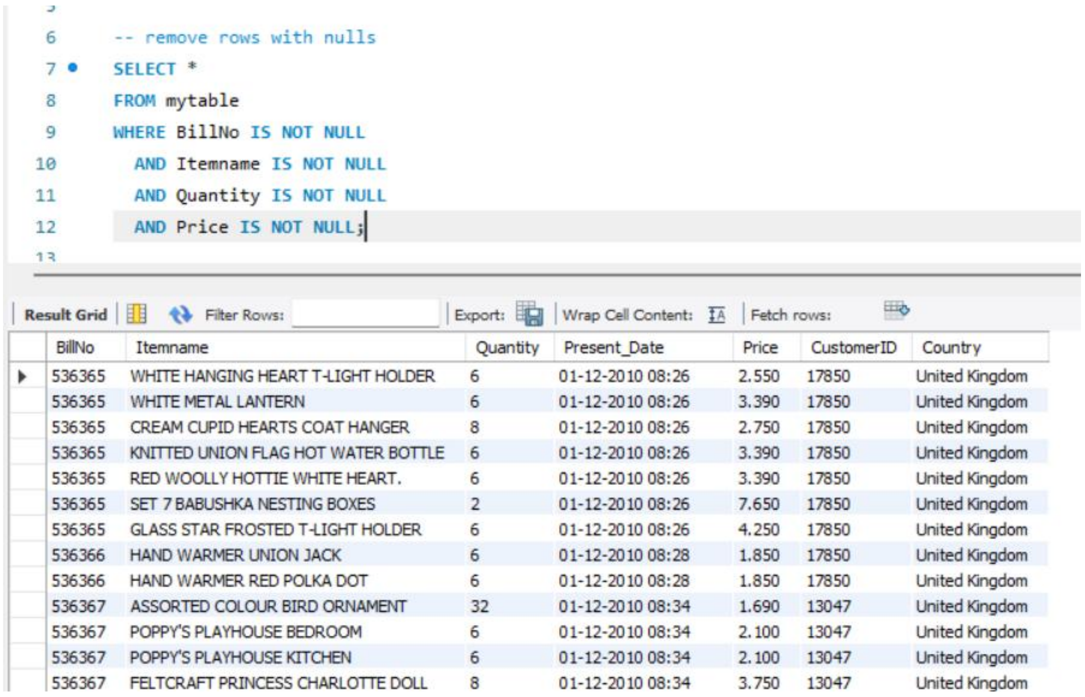
- **Validates data types for accurate calculations**
- **Helps identify columns that may contain missing values**

Supports effective data cleaning and preprocessing

1: EXPLORATORY DATA ANALYSIS (EDA) with Data Cleaning

remove rows with nulls

```
SELECT *
FROM mytable
WHERE BillNo IS NOT NULL
AND Itemname IS NOT NULL
AND Quantity IS NOT NULL
AND Price IS NOT NULL;
```



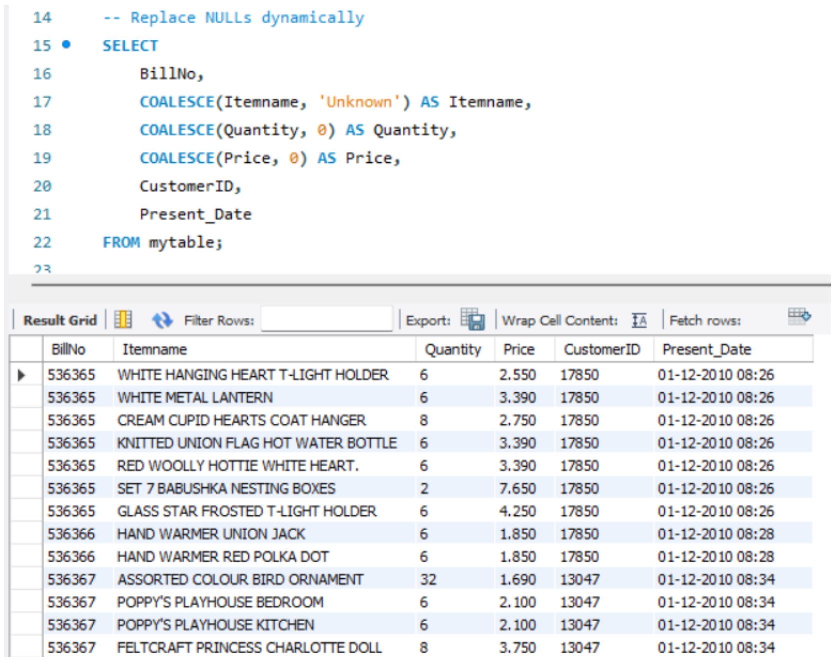
```
-- remove rows with nulls
SELECT *
FROM mytable
WHERE BillNo IS NOT NULL
  AND Itemname IS NOT NULL
  AND Quantity IS NOT NULL
  AND Price IS NOT NULL;
```

BillNo	Itemname	Quantity	Present_Date	Price	CustomerID	Country
536365	WHITE HANGING HEART T-LIGHT HOLDER	6	01-12-2010 08:26	2.550	17850	United Kingdom
536365	WHITE METAL LANTERN	6	01-12-2010 08:26	3.390	17850	United Kingdom
536365	CREAM CUPID HEARTS COAT HANGER	8	01-12-2010 08:26	2.750	17850	United Kingdom
536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-12-2010 08:26	3.390	17850	United Kingdom
536365	RED WOOLLY HOTTIE WHITE HEART.	6	01-12-2010 08:26	3.390	17850	United Kingdom
536365	SET 7 BABUSHKA NESTING BOXES	2	01-12-2010 08:26	7.650	17850	United Kingdom
536365	GLASS STAR FROSTED T-LIGHT HOLDER	6	01-12-2010 08:26	4.250	17850	United Kingdom
536366	HAND WARMER UNION JACK	6	01-12-2010 08:28	1.850	17850	United Kingdom
536366	HAND WARMER RED POLKA DOT	6	01-12-2010 08:28	1.850	17850	United Kingdom
536367	ASSORTED COLOUR BIRD ORNAMENT	32	01-12-2010 08:34	1.690	13047	United Kingdom
536367	POPPY'S PLAYHOUSE BEDROOM	6	01-12-2010 08:34	2.100	13047	United Kingdom
536367	POPPY'S PLAYHOUSE KITCHEN	6	01-12-2010 08:34	2.100	13047	United Kingdom
536367	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	01-12-2010 08:34	3.750	13047	United Kingdom

This SQL query filters the dataset to keep only **valid transaction records**. It removes any rows where critical fields such as **Bill Number, Item Name, Quantity, or Price** are missing (NULL). By doing this, the dataset becomes cleaner and more reliable for analysis, ensuring that sales calculations, dashboards, and pattern analysis are based on complete and accurate data.

Replace NULLs dynamically

```
SELECT BillNo,
    COALESCE (Itemname, 'Unknown') AS Itemname,
    COALESCE (Quantity, 0) AS Quantity,
    COALESCE (Price, 0) AS Price,
    CustomerID, Present_Date
FROM mytable;
```



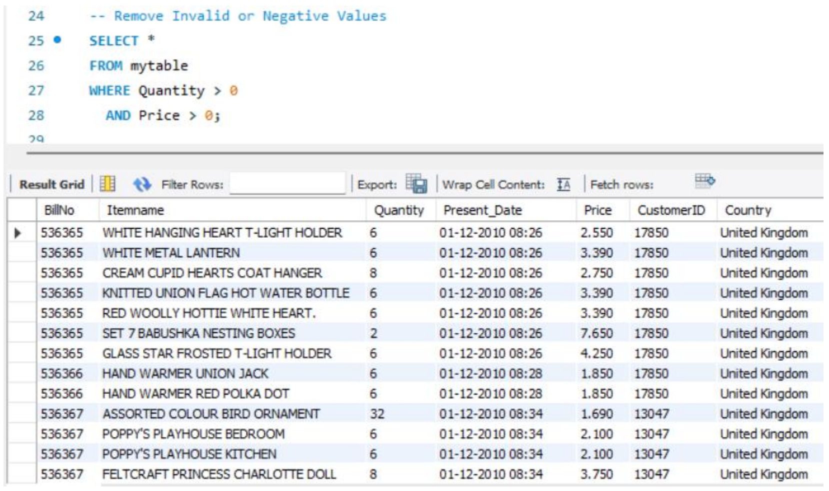
```
-- Replace NULLs dynamically
SELECT
    BillNo,
    COALESCE(Itemname, 'Unknown') AS Itemname,
    COALESCE(Quantity, 0) AS Quantity,
    COALESCE(Price, 0) AS Price,
    CustomerID,
    Present_Date
FROM mytable;
```

BillNo	Itemname	Quantity	Price	CustomerID	Present_Date
536365	WHITE HANGING HEART T-LIGHT HOLDER	6	2.550	17850	01-12-2010 08:26
536365	WHITE METAL LANTERN	6	3.390	17850	01-12-2010 08:26
536365	CREAM CUPID HEARTS COAT HANGER	8	2.750	17850	01-12-2010 08:26
536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	3.390	17850	01-12-2010 08:26
536365	RED WOOLLY HOTTIE WHITE HEART.	6	3.390	17850	01-12-2010 08:26
536365	SET 7 BABUSHKA NESTING BOXES	2	7.650	17850	01-12-2010 08:26
536365	GLASS STAR FROSTED T-LIGHT HOLDER	6	4.250	17850	01-12-2010 08:26
536366	HAND WARMER UNION JACK	6	1.850	17850	01-12-2010 08:28
536366	HAND WARMER RED POLKA DOT	6	1.850	17850	01-12-2010 08:28
536367	ASSORTED COLOUR BIRD ORNAMENT	32	1.690	13047	01-12-2010 08:34
536367	POPPY'S PLAYHOUSE BEDROOM	6	2.100	13047	01-12-2010 08:34
536367	POPPY'S PLAYHOUSE KITCHEN	6	2.100	13047	01-12-2010 08:34
536367	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	3.750	13047	01-12-2010 08:34

- Retrieves transaction details such as **BillNo, Itemname, Quantity, Price, CustomerID, and Date.**
- Replaces **NULL values dynamically** using the COALESCE () function:
- Missing product names are replaced with **'Unknown'**
- Missing quantity values are replaced with **0**
- Missing price values are replaced with **0**
- Ensures that the dataset.

Remove Invalid or Negative Values

```
SELECT *

FROM mytable

WHERE Quantity > 0

  AND Price > 0;
```



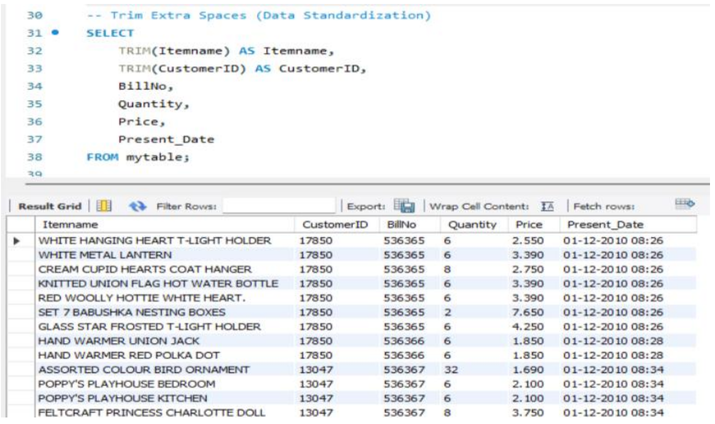
```
-- Remove Invalid or Negative Values
SELECT *
FROM mytable
WHERE Quantity > 0
  AND Price > 0;
```

BillNo	Itemname	Quantity	Present_Date	Price	CustomerID	Country
536365	WHITE HANGING HEART T-LIGHT HOLDER	6	01-12-2010 08:26	2.550	17850	United Kingdom
536365	WHITE METAL LANTERN	6	01-12-2010 08:26	3.390	17850	United Kingdom
536365	CREAM CUPID HEARTS COAT HANGER	8	01-12-2010 08:26	2.750	17850	United Kingdom
536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-12-2010 08:26	3.390	17850	United Kingdom
536365	RED WOOLLY HOTTIE WHITE HEART.	6	01-12-2010 08:26	3.390	17850	United Kingdom
536365	SET 7 BABUSHKA NESTING BOXES	2	01-12-2010 08:26	7.650	17850	United Kingdom
536365	GLASS STAR FROSTED T-LIGHT HOLDER	6	01-12-2010 08:26	4.250	17850	United Kingdom
536366	HAND WARMER UNION JACK	6	01-12-2010 08:28	1.850	17850	United Kingdom
536366	HAND WARMER RED POLKA DOT	6	01-12-2010 08:28	1.850	17850	United Kingdom
536367	ASSORTED COLOUR BIRD ORNAMENT	32	01-12-2010 08:34	1.690	13047	United Kingdom
536367	POPPY'S PLAYHOUSE BEDROOM	6	01-12-2010 08:34	2.100	13047	United Kingdom
536367	POPPY'S PLAYHOUSE KITCHEN	6	01-12-2010 08:34	2.100	13047	United Kingdom
536367	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	01-12-2010 08:34	3.750	13047	United Kingdom

- **Quantity > 0** → Removes returns, cancellations, or incorrect entries where quantity is zero or negative
- **Price > 0** → Removes free items, pricing errors, or invalid price values

Trim Extra Spaces (Data Standardization)

```
SELECT TRIM(Itemname) AS Itemname,

    TRIM(CustomerID) AS CustomerID,

     BillNo, Quantity, Price, Present_Date

FROM mytable;
```



```
-- Trim Extra Spaces (Data Standardization)
SELECT
    TRIM(Itemname) AS Itemname,
    TRIM(CustomerID) AS CustomerID,
    BillNo,
    Quantity,
    Price,
    Present_Date
FROM mytable;
```

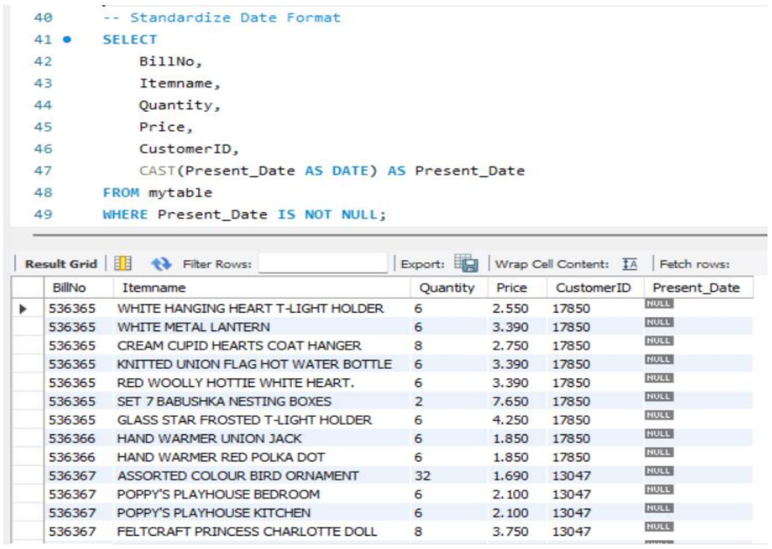
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Itemname	CustomerID	BillNo	Quantity	Price	Present_Date
WHITE HANGING HEART T-LIGHT HOLDER	17850	536365	6	2.550	01-12-2010 08:26
WHITE METAL LANTERN	17850	536365	6	3.390	01-12-2010 08:26
CREAM CUPID HEARTS COAT HANGER	17850	536365	8	2.750	01-12-2010 08:26
KNITTED UNION FLAG HOT WATER BOTTLE	17850	536365	6	3.390	01-12-2010 08:26
RED WOOLLY HOTTIE WHITE HEART.	17850	536365	6	3.390	01-12-2010 08:26
SET 7 BABUSHKA NESTING BOXES	17850	536365	2	7.650	01-12-2010 08:26
GLASS STAR FROSTED T-LIGHT HOLDER	17850	536365	6	4.250	01-12-2010 08:26
HAND WARMER UNION JACK	17850	536366	6	1.850	01-12-2010 08:28
HAND WARMER RED POLKA DOT	17850	536366	6	1.850	01-12-2010 08:28
ASSORTED COLOUR BIRD ORNAMENT	13047	536367	32	1.690	01-12-2010 08:34
POPPY'S PLAYHOUSE BEDROOM	13047	536367	6	2.100	01-12-2010 08:34
POPPY'S PLAYHOUSE KITCHEN	13047	536367	6	2.100	01-12-2010 08:34
FELTCRAFT PRINCESS CHARLOTTE DOLL	13047	536367	8	3.750	01-12-2010 08:34

This query is used for **data standardization and cleaning**. It removes unwanted leading and trailing spaces from the **Itemname** and **CustomerID** columns using the TRIM () function. This ensures consistency in product and customer values, prevents duplicate records caused by spacing issues, and improves the accuracy of analysis and reporting. All other fields are selected as they are to preserve the transaction details.

Standardize Date Format

```
SELECT

    BillNo, Itemname, Quantity,

    Price, CustomerID,

    CAST (Present_Date AS DATE) AS Present_Date

FROM mytable

WHERE Present_Date IS NOT NULL;
```



```
-- Standardize Date Format
SELECT
    BillNo,
    Itemname,
    Quantity,
    Price,
    CustomerID,
    CAST(Present_Date AS DATE) AS Present_Date
FROM mytable
WHERE Present_Date IS NOT NULL;
```

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

BillNo	Itemname	Quantity	Price	CustomerID	Present_Date
536365	WHITE HANGING HEART T-LIGHT HOLDER	6	2.550	17850	NULL
536365	WHITE METAL LANTERN	6	3.390	17850	NULL
536365	CREAM CUPID HEARTS COAT HANGER	8	2.750	17850	NULL
536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	3.390	17850	NULL
536365	RED WOOLLY HOTTIE WHITE HEART.	6	3.390	17850	NULL
536365	SET 7 BABUSHKA NESTING BOXES	2	7.650	17850	NULL
536365	GLASS STAR FROSTED T-LIGHT HOLDER	6	4.250	17850	NULL
536366	HAND WARMER UNION JACK	6	1.850	17850	NULL
536366	HAND WARMER RED POLKA DOT	6	1.850	17850	NULL
536367	ASSORTED COLOUR BIRD ORNAMENT	32	1.690	13047	NULL
536367	POPPY'S PLAYHOUSE BEDROOM	6	2.100	13047	NULL
536367	POPPY'S PLAYHOUSE KITCHEN	6	2.100	13047	NULL
536367	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	3.750	13047	NULL

This query is used to **standardize the date format** of the transaction date column. It converts the Present_Date field into a proper DATE data type so that all records follow a consistent date format. Records with missing dates are excluded to ensure data accuracy. This makes the dataset suitable for reliable time-based analysis such as daily, monthly, or yearly sales trends.

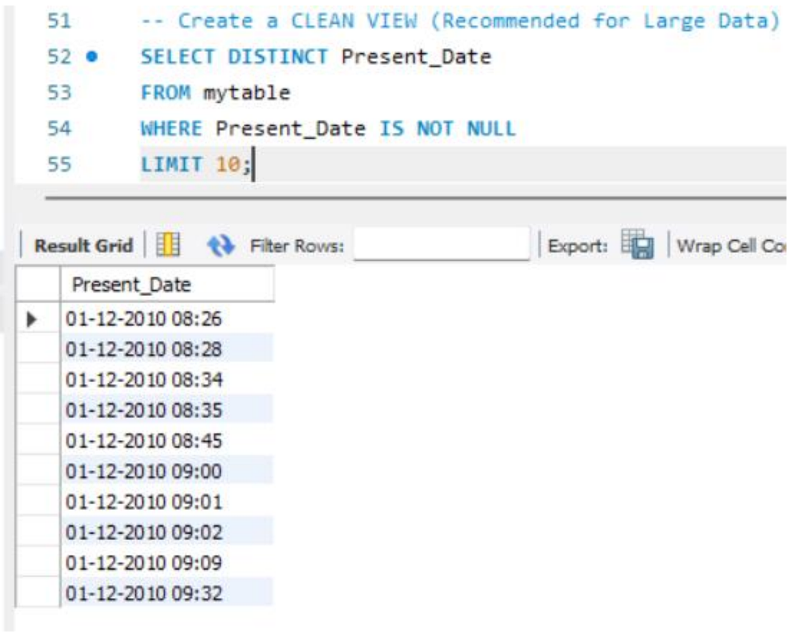
Create a CLEAN VIEW

SELECT DISTINCT Present_Date

FROM mytable

WHERE Present_Date IS NOT NULL

LIMIT 10;



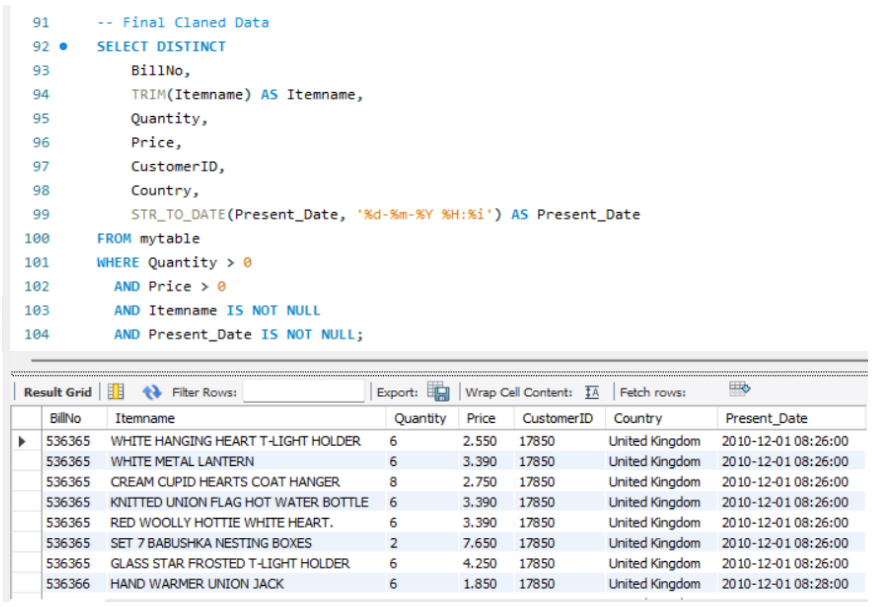
```
-- Create a CLEAN VIEW (Recommended for Large Data)
SELECT DISTINCT Present_Date
FROM mytable
WHERE Present_Date IS NOT NULL
LIMIT 10;
```

```
Present_Date
01-12-2010 08:26
01-12-2010 08:28
01-12-2010 08:34
01-12-2010 08:35
01-12-2010 08:45
01-12-2010 09:00
01-12-2010 09:01
01-12-2010 09:02
01-12-2010 09:09
01-12-2010 09:32
```

This SQL query retrieves **unique (distinct) transaction dates** from the table while excluding any **NULL date values**. Using DISTINCT ensures that duplicate dates are removed, and LIMIT 10 restricts the output to the first 10 records for quick inspection. This is useful for validating date data and understanding the time range of transactions in large datasets.

Cleaned Data

```
SELECT DISTINCT

    BillNo, TRIM(Itemname) AS Itemname,

    Quantity, Price, CustomerID, Country,

    STR_TO_DATE (Present_Date, '%d-%m-%Y %H: %i') AS Present_Date

FROM mytable WHERE Quantity > 0 AND Price > 0

AND Itemname IS NOT NULL

AND Present_Date IS NOT NULL;
```



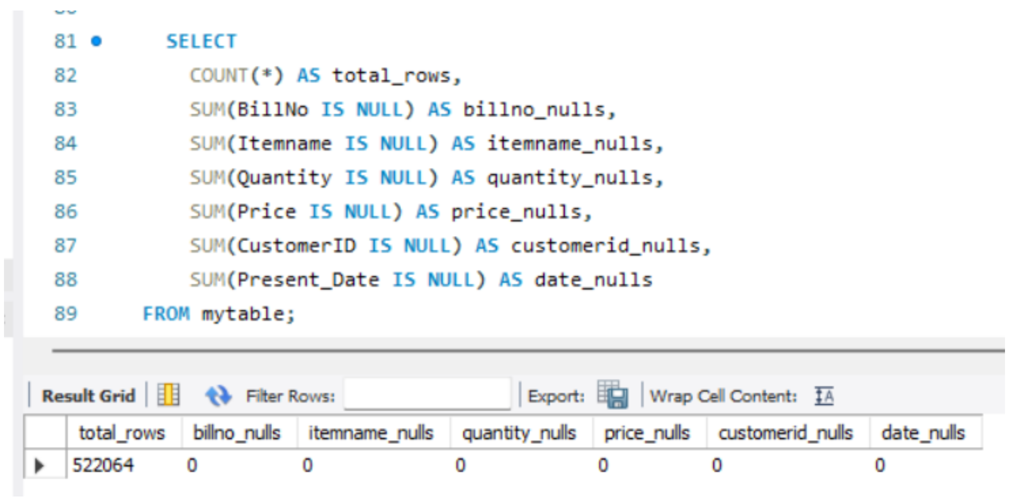
```
-- Final Claned Data
SELECT DISTINCT
    BillNo,
    TRIM(Itemname) AS Itemname,
    Quantity,
    Price,
    CustomerID,
    Country,
    STR_TO_DATE(Present_Date, '%d-%m-%Y %H:%i') AS Present_Date
FROM mytable
WHERE Quantity > 0
  AND Price > 0
  AND Itemname IS NOT NULL
  AND Present_Date IS NOT NULL;
```

BillNo	Itemname	Quantity	Price	CustomerID	Country	Present_Date
536365	WHITE HANGING HEART T-LIGHT HOLDER	6	2.550	17850	United Kingdom	2010-12-01 08:26:00
536365	WHITE METAL LANTERN	6	3.390	17850	United Kingdom	2010-12-01 08:26:00
536365	CREAM CUPID HEARTS COAT HANGER	8	2.750	17850	United Kingdom	2010-12-01 08:26:00
536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	3.390	17850	United Kingdom	2010-12-01 08:26:00
536365	RED WOOLLY HOTTIE WHITE HEART.	6	3.390	17850	United Kingdom	2010-12-01 08:26:00
536365	SET 7 BABUSHKA NESTING BOXES	2	7.650	17850	United Kingdom	2010-12-01 08:26:00
536365	GLASS STAR FROSTED T-LIGHT HOLDER	6	4.250	17850	United Kingdom	2010-12-01 08:26:00
536366	HAND WARMER UNION JACK	6	1.850	17850	United Kingdom	2010-12-01 08:28:00

This SQL query is used to **clean and standardize transactional sales data** before analysis. It retrieves **distinct purchase records** by selecting unique combinations of bill number, product name, quantity, price, customer ID, country, and transaction date. The TRIM(Itemname) function removes unnecessary spaces from product names to ensure consistency, while STR_TO_DATE converts the transaction date into a proper date–time format for accurate time-based analysis. The WHERE clause filters out invalid records by keeping only rows with **positive quantity and price values**, excluding missing product names and null transaction dates. Overall, this query ensures that only **valid, clean, and analysis-ready data** is used for further reporting, visualization, and market basket analysis.

Final Data Check

```
SELECT
    COUNT (*) AS total rows,
    SUM (BillNo IS NULL) AS billno_nulls,
    SUM (Itemname IS NULL) AS itemname_nulls,
    SUM (Quantity IS NULL) AS quantity_nulls,
    SUM (Price IS NULL) AS price_nulls,
    SUM (CustomerID IS NULL) AS customerid_nulls,
    SUM (Present_Date IS NULL) AS date_nulls
FROM mytable;
```



```
    SELECT
      COUNT(*) AS total_rows,
      SUM(BillNo IS NULL) AS billno_nulls,
      SUM(Itemname IS NULL) AS itemname_nulls,
      SUM(Quantity IS NULL) AS quantity_nulls,
      SUM(Price IS NULL) AS price_nulls,
      SUM(CustomerID IS NULL) AS customerid_nulls,
      SUM(Present_Date IS NULL) AS date_nulls
  FROM mytable;
```

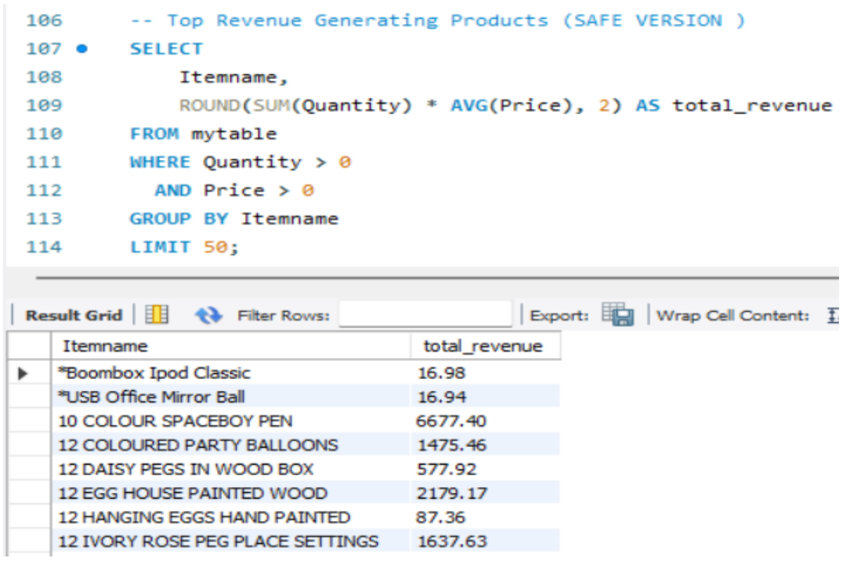
total_rows	billno_nulls	itemname_nulls	quantity_nulls	price_nulls	customerid_nulls	date_nulls
522064	0	0	0	0	0	0

This SQL query is used to **assess data quality** by identifying missing values in the dataset. It first calculates the total number of records in the table and then checks each important column—BillNo, Itemname, Quantity, Price, CustomerID, and Present_Date—to count how many entries contain NULL values. By summarizing the number of missing values in each column, the query helps evaluate the completeness and reliability of the data before analysis. This step is essential to decide what cleaning actions (such as removing or correcting records) are required to ensure accurate reporting, visualization, and modeling.

2. SQL Analysis & Transformation

a) Top Revenue Generating Products

```
SELECT Itemname,

   ROUND(SUM(Quantity) * AVG(Price), 2) AS total_revenue

FROM mytable WHERE Quantity > 0

  AND Price > 0 GROUP BY Itemname

LIMIT 50;
```



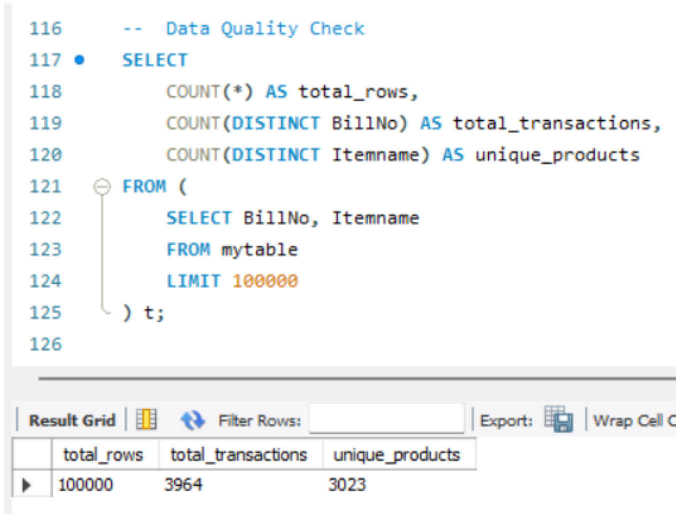
```
-- Top Revenue Generating Products (SAFE VERSION )
SELECT
    Itemname,
    ROUND(SUM(Quantity) * AVG(Price), 2) AS total_revenue
FROM mytable
WHERE Quantity > 0
  AND Price > 0
GROUP BY Itemname
LIMIT 50;
```

Itemname	total_revenue
*Boombox Ipod Classic	16.98
*USB Office Mirror Ball	16.94
10 COLOUR SPACEBOY PEN	6677.40
12 COLOURED PARTY BALLOONS	1475.46
12 DAISY PEGS IN WOOD BOX	577.92
12 EGG HOUSE PAINTED WOOD	2179.17
12 HANGING EGGS HAND PAINTED	87.36
12 IVORY ROSE PEG PLACE SETTINGS	1637.63

This SQL query identifies the **top revenue-generating products** by calculating the total revenue contributed by each item. It filters out invalid records by considering only transactions with positive quantity and price values, ensuring accurate revenue calculations. The query groups the data by product name and computes revenue as the product of the total quantity sold and the average price for each item. The result highlights the products that contribute the most to overall sales, helping businesses understand which items drive revenue and should be prioritized for inventory planning, promotions, and strategic decision-making.

b) Data Quality Check

```
SELECT
   COUNT (*) AS total_rows,
   COUNT (DISTINCT BillNo) AS total_transactions,
   COUNT (DISTINCT Itemname) AS unique_products
FROM (
   SELECT BillNo, Itemname
   FROM mytable
   LIMIT 100000
) t;
```



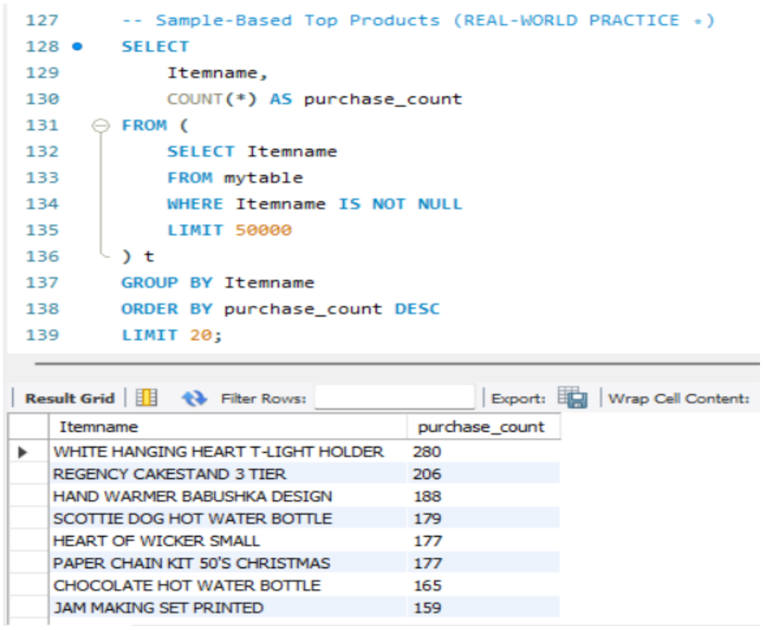
```
-- Data Quality Check
SELECT
    COUNT(*) AS total_rows,
    COUNT(DISTINCT BillNo) AS total_transactions,
    COUNT(DISTINCT Itemname) AS unique_products
FROM (
    SELECT BillNo, Itemname
    FROM mytable
    LIMIT 100000
) t;
```

total_rows	total_transactions	unique_products
100000	3964	3023

This query is used to perform a **data quality and data understanding check** on the transactional dataset. It calculates the total number of records, the number of unique transactions, and the number of unique products present in the data. By counting total rows, it helps assess the overall size of the dataset, while the count of distinct BillNo values indicates how many individual transactions or shopping baskets exist. The count of distinct Itemname values shows the variety of products available. Together, these metrics provide a quick overview of data volume, transaction coverage, and product diversity, which is essential for validating data completeness and preparing the dataset for further analysis such as dashboarding and Market Basket Analysis.

c). Sample-Based Top Products

```
SELECT Itemname,
   COUNT (*) AS purchase_count
FROM (SELECT Itemname
   FROM mytable WHERE Itemname IS NOT NULL
   LIMIT 50000) t GROUP BY Itemname
ORDER BY purchase_count DESC
LIMIT 20;
```



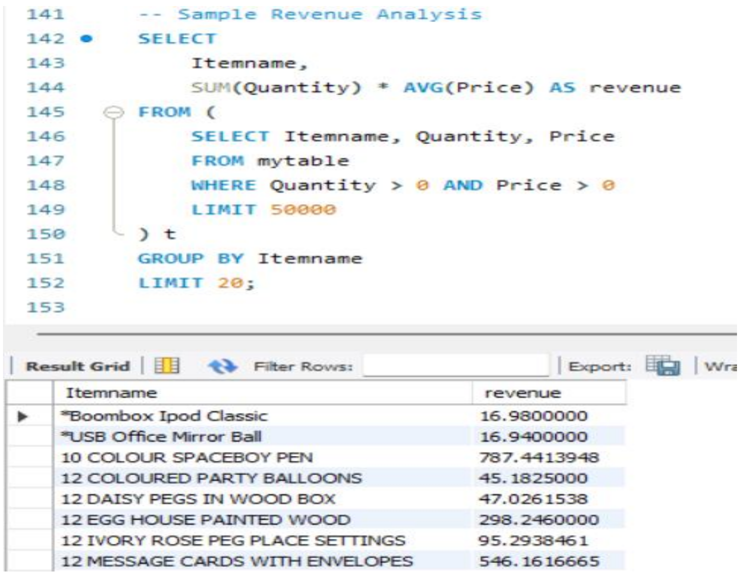
```
-- Sample-Based Top Products (REAL-WORLD PRACTICE *)
SELECT
    Itemname,
    COUNT(*) AS purchase_count
FROM (
    SELECT Itemname
    FROM mytable
    WHERE Itemname IS NOT NULL
    LIMIT 50000
) t
GROUP BY Itemname
ORDER BY purchase_count DESC
LIMIT 20;
```

Itemname	purchase_count
WHITE HANGING HEART T-LIGHT HOLDER	280
REGENCY CAKESTAND 3 TIER	206
HAND WARMER BABUSHKA DESIGN	188
SCOTTIE DOG HOT WATER BOTTLE	179
HEART OF WICKER SMALL	177
PAPER CHAIN KIT 50'S CHRISTMAS	177
CHOCOLATE HOT WATER BOTTLE	165
JAM MAKING SET PRINTED	159

This SQL query is used to identify the **top 20 most frequently purchased products** from the dataset. First, it selects up to **50,000 valid product records** where the item name is not null, helping to control data volume and improve query performance. The outer query then groups the data by **Itemname** and counts how many times each product appears, which represents how often each item was purchased. Finally, the results are sorted in descending order of purchase count, and only the top 20 products are displayed. This analysis helps businesses understand high-demand products, enabling better inventory planning, product placement, and sales strategy decisions.

d). Sample Revenue Analysis

```
SELECT Itemname,
   SUM(Quantity) * AVG(Price) AS revenue
FROM (SELECT Itemname, Quantity, Price
   FROM mytable
   WHERE Quantity > 0 AND Price > 0
   LIMIT 50000) t
GROUP BY Itemname
LIMIT 20;
```



```
-- Sample Revenue Analysis
SELECT
    Itemname,
    SUM(Quantity) * AVG(Price) AS revenue
FROM (
    SELECT Itemname, Quantity, Price
    FROM mytable
    WHERE Quantity > 0 AND Price > 0
    LIMIT 50000
) t
GROUP BY Itemname
LIMIT 20;
```

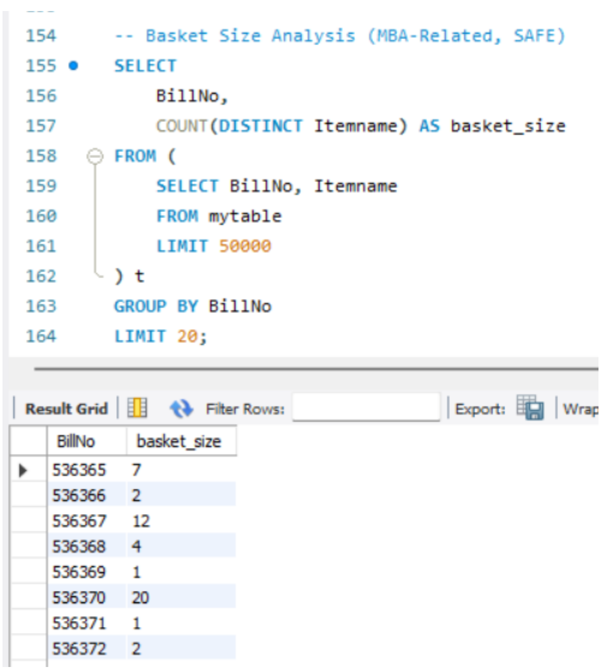
Result Grid | Filter Rows: | Export: | Wra

Itemname	revenue
*Boombox Ipod Classic	16.9800000
*USB Office Mirror Ball	16.9400000
10 COLOUR SPACEBOY PEN	787.4413948
12 COLOURED PARTY BALLOONS	45.1825000
12 DAISY PEGS IN WOOD BOX	47.0261538
12 EGG HOUSE PAINTED WOOD	298.2460000
12 IVORY ROSE PEG PLACE SETTINGS	95.2938461
12 MESSAGE CARDS WITH ENVELOPES	546.1616665

This SQL query is used to analyze **product-wise revenue contribution** from the sales data. It first filters the dataset to include only valid transactions where both quantity and price are greater than zero, ensuring that returns, errors, or invalid records are excluded. To improve performance on large datasets, a limited sample of records is selected. The query then groups the data by product name and calculates the estimated revenue for each product by multiplying the total quantity sold with the average selling price. Finally, it returns the top 20 products, helping the business identify high-revenue–generating items that have the greatest impact on overall sales performance.

e). Basket Size Analysis

```
SELECT BillNo,
   COUNT (DISTINCT Itemname) AS basket_size
FROM (SELECT BillNo, Itemname
   FROM mytable LIMIT 50000) t
GROUP BY BillNo
LIMIT 20;
```



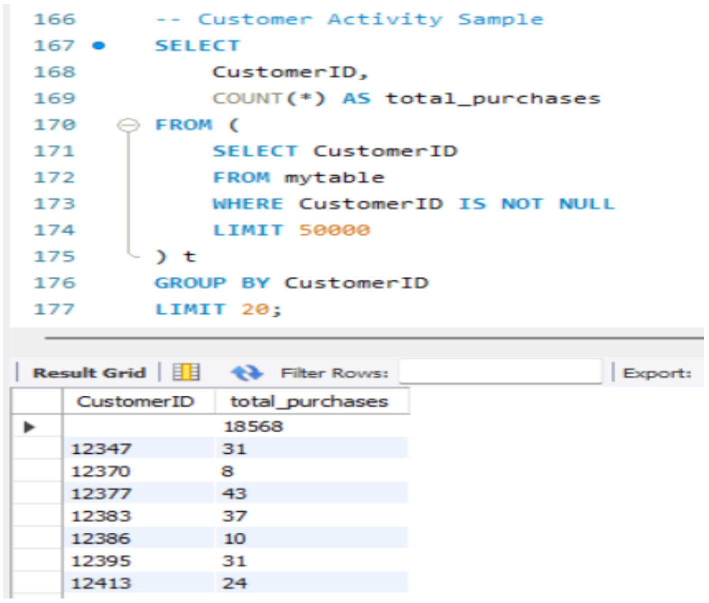
```
-- Basket Size Analysis (MBA-Related, SAFE)
SELECT
    BillNo,
    COUNT(DISTINCT Itemname) AS basket_size
FROM (
    SELECT BillNo, Itemname
    FROM mytable
    LIMIT 50000
) t
GROUP BY BillNo
LIMIT 20;
```

BillNo	basket_size
536365	7
536366	2
536367	12
536368	4
536369	1
536370	20
536371	1
536372	2

This query is used to perform **Basket Size Analysis**, which is an important concept in Market Basket Analysis. It calculates how many **distinct products** are included in each customer transaction. The inner query limits the dataset to a manageable number of records to ensure safe and efficient execution. The outer query then groups the data by BillNo (transaction ID) and counts the number of unique items purchased in each bill, resulting in the basket size. This analysis helps businesses understand typical shopping behavior, such as whether customers usually buy single items or multiple products in one transaction, which can be used to design bundling strategies, promotions, and store layout decisions.

f). Customer Activity Sample

```
SELECT CustomerID,
   COUNT (*) AS total_purchases
FROM (SELECT CustomerID
   FROM mytable
   WHERE CustomerID IS NOT NULL
   LIMIT 50000) t
GROUP BY CustomerID
LIMIT 20;
```



```
-- Customer Activity Sample
SELECT
    CustomerID,
    COUNT(*) AS total_purchases
FROM (
    SELECT CustomerID
    FROM mytable
    WHERE CustomerID IS NOT NULL
    LIMIT 50000
) t
GROUP BY CustomerID
LIMIT 20;
```

CustomerID	total_purchases
	18568
12347	31
12370	8
12377	43
12383	37
12386	10
12395	31
12413	24

This query analyzes **customer purchase activity** by calculating how many transactions each customer has made. First, it selects up to **50,000 non-null CustomerID records** from the table to limit the data size and improve performance. Then, it groups the data by **CustomerID** and counts the number of records for each customer, representing the **total number of purchases** made by that customer. Finally, it displays a sample of **20 customers**, providing a quick view of customer engagement levels and helping identify repeat or high-activity customers for further analysis.

g). Peak Transaction Analysis

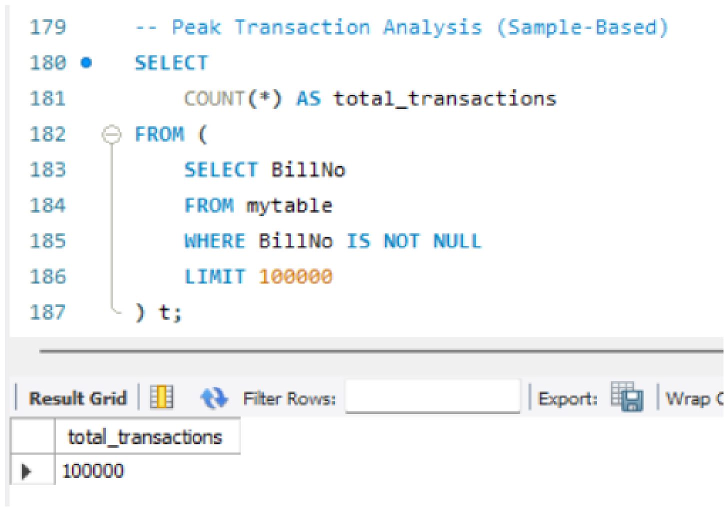
SELECT

COUNT (*) AS total_transactions

FROM (SELECT BillNo

FROM mytable WHERE BillNo IS NOT NULL

LIMIT 100000) t;



```
-- Peak Transaction Analysis (Sample-Based)
SELECT
    COUNT(*) AS total_transactions
FROM (
    SELECT BillNo
    FROM mytable
    WHERE BillNo IS NOT NULL
    LIMIT 100000
) t;
```

```
total_transactions
100000
```

This query is used to perform a **sample-based peak transaction analysis** by counting the total number of valid transactions from the dataset. It first selects a limited sample of transaction records (up to 100,000 rows) where the transaction identifier (BillNo) is not null, ensuring only valid transactions are considered. By applying the COUNT (*) function on this sampled data, the query provides an estimate of transaction volume without scanning the entire table, which helps improve query performance on large datasets. This approach is useful for quickly understanding transaction intensity and workload patterns while working with high-volume retail data.

h). Product Diversity in Sample Data

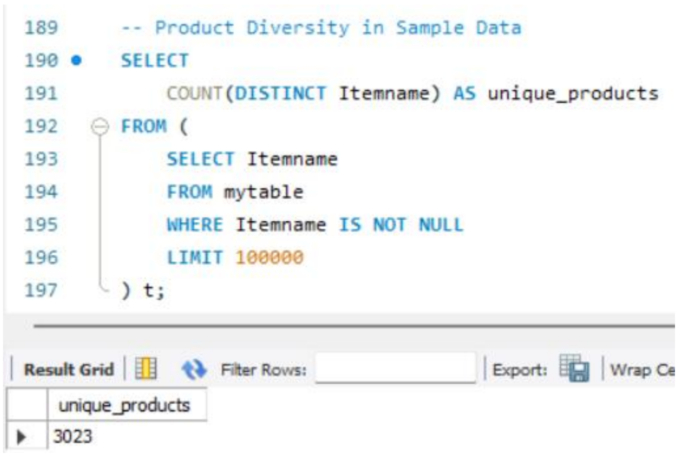
SELECT

COUNT (DISTINCT Itemname) AS unique_products

FROM (SELECT Itemname

FROM mytable WHERE Itemname IS NOT NULL

LIMIT 100000) t;



```
-- Product Diversity in Sample Data
SELECT
    COUNT(DISTINCT Itemname) AS unique_products
FROM (
    SELECT Itemname
    FROM mytable
    WHERE Itemname IS NOT NULL
    LIMIT 100000
) t;
```

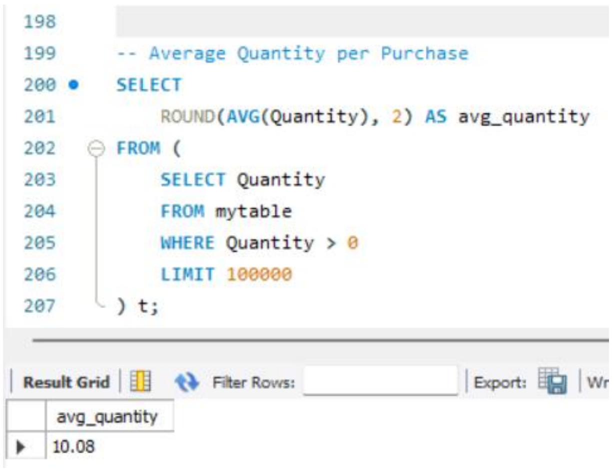
Result Grid | Filter Rows: | Export: | Wrap Ce

```
unique_products
3023
```

This query is used to understand **product diversity** within the dataset. It counts the number of **unique products** present in a sample of the data by calculating the distinct values of Itemname. The inner query first filters out any NULL product names and limits the dataset to 100,000 records to ensure efficient processing on large tables. The outer query then computes the count of distinct product names from this sample. This helps assess the variety of products available for analysis and provides an estimate of dataset complexity, which is important for understanding the scale of market basket and association analysis.

i). Average Quantity per Purchase

```
SELECT
   ROUND(AVG(Quantity), 2) AS avg_quantity
FROM (SELECT Quantity
   FROM mytable WHERE Quantity > 0
   LIMIT 100000) t;
```



```

-- Average Quantity per Purchase
SELECT
    ROUND(AVG(Quantity), 2) AS avg_quantity
FROM (
    SELECT Quantity
    FROM mytable
    WHERE Quantity > 0
    LIMIT 100000
) t;
```

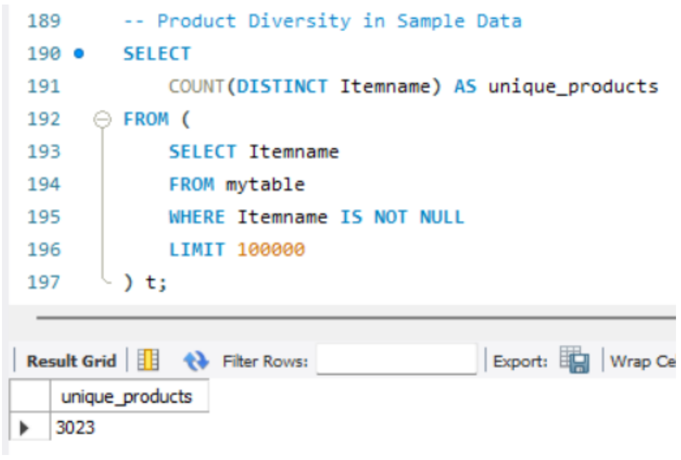
Result Grid | Filter Rows: | Export: | Wr

```
avg_quantity
10.08
```

This query calculates the **average quantity of items purchased per transaction** while ensuring safe and efficient execution on large datasets. First, it filters the data to include only records where the quantity is greater than zero, eliminating invalid or return entries. The inner query limits the number of rows processed to 100,000 to avoid performance issues on very large tables. The outer query then computes the average of these valid quantities and rounds the result to two decimal places. This metric helps businesses understand typical purchase volumes per transaction, which is useful for inventory planning, demand forecasting, and evaluating customer buying behavior.

j). Product Diversity in Sample Data

```
SELECT
   COUNT (DISTINCT Itemname) AS unique_products
FROM (SELECT Itemname
   FROM mytable
   WHERE Itemname IS NOT NULL
   LIMIT 100000) t;
```



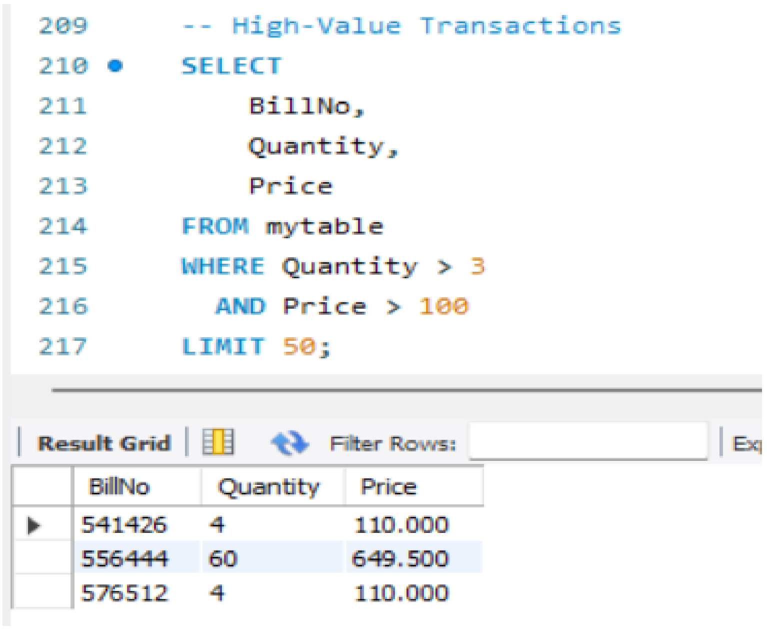
```
-- Product Diversity in Sample Data
SELECT
    COUNT(DISTINCT Itemname) AS unique_products
FROM (
    SELECT Itemname
    FROM mytable
    WHERE Itemname IS NOT NULL
    LIMIT 100000
) t;
```

```
unique_products
3023
```

This query is used to measure **product diversity** in the dataset by identifying how many **unique products** are present in a sample of the data. It first filters out records with null product names to ensure accuracy, then limits the analysis to 100,000 rows to improve performance when working with large datasets. By counting the distinct values of Itemname, the query provides an estimate of how many different products appear in the sample transactions. This helps the business understand the breadth of its product assortment and supports decisions related to inventory management, assortment planning, and further analytical modeling such as Market Basket Analysis.

k). High-Value Transactions

```
SELECT BillNo,
   Quantity, Price
FROM mytable WHERE Quantity > 3
 AND Price > 100
LIMIT 50;
```



```
-- High-Value Transactions
SELECT
    BillNo,
    Quantity,
    Price
FROM mytable
WHERE Quantity > 3
  AND Price > 100
LIMIT 50;
```

	BillNo	Quantity	Price
▶	541426	4	110.000
	556444	60	649.500
	576512	4	110.000

This query is used to identify **high-value transactions** from the dataset. It retrieves records where customers have purchased **more than three units of a product** and where the **price of the product is greater than 100**, indicating transactions that contribute higher revenue. By filtering on both quantity and price, the query focuses on purchases that are more valuable from a business perspective. Limiting the result to 50 records allows quick inspection of sample high-value sales, which can help businesses analyze premium product demand, identify profitable transactions, and design targeted marketing or loyalty strategies.

3) Python & Apriori Algorithm Implementation: Theoretical Expiation

1. Data Selection and Cleaning

The analysis begins by selecting only the required attributes—**Bill Number** and **Item Name**-from the original dataset. Missing values are removed to ensure data integrity. This step ensures that only valid transactions and products are considered for further analysis.

Purpose:
To eliminate incomplete records and focus only on transactional information relevant to market basket analysis.

---

2. Transformation into Transaction (Basket) Format

The cleaned data is converted into a **transaction matrix**, where:

- Each row represents a unique transaction (BillNo)
- Each column represents a unique product (Itemname)
- Cell values indicate whether a product appears in a transaction

The data is reshaped using grouping and pivoting operations to create a structured basket format.

Purpose:
Apriori requires data in transactional format to identify product co-occurrence patterns.

---

3. Binary Encoding of Transactions

The transaction matrix is transformed into a **binary format**, where:

- True / 1 indicates the presence of an item in a transaction
- False / 0 indicates absence

This binary encoding is essential for efficient computation of itemset frequencies.

Purpose:
To standardize input data for the Apriori algorithm and reduce computational complexity.

---

4. Frequent Itemset Generation using Apriori Algorithm

The Apriori algorithm is applied to the binary transaction matrix to identify **frequent itemsets**-groups of products that frequently appear together in transactions.

Key constraints are applied:

- **Minimum support threshold** to filter insignificant itemsets
- **Maximum itemset length** to prevent combinatorial explosion and improve performance

Purpose:
To extract meaningful and statistically significant product combinations.

---

5. Association Rule Mining

From the frequent itemsets, **association rules** are generated in the form:

If item A is purchased, item B is likely to be purchased.

Rules are evaluated using:

- **Support** – how often the rule occurs
- **Confidence** – reliability of the rule
- **Lift** – strength of the association compared to random chance

Only strong and reliable rules are retained by applying confidence thresholds.

Purpose:
To identify actionable insights that support cross-selling, bundling, and recommendation strategies.

---

6. Visualization of Apriori Results

The results are visualized using charts such as:

- **Bar charts** to highlight top frequent products or itemsets
- **Scatter plots** to analyse rule strength using confidence and lift

These visualizations simplify complex association patterns and make insights more interpretable.

Purpose:
To communicate analytical findings clearly to business stakeholders.

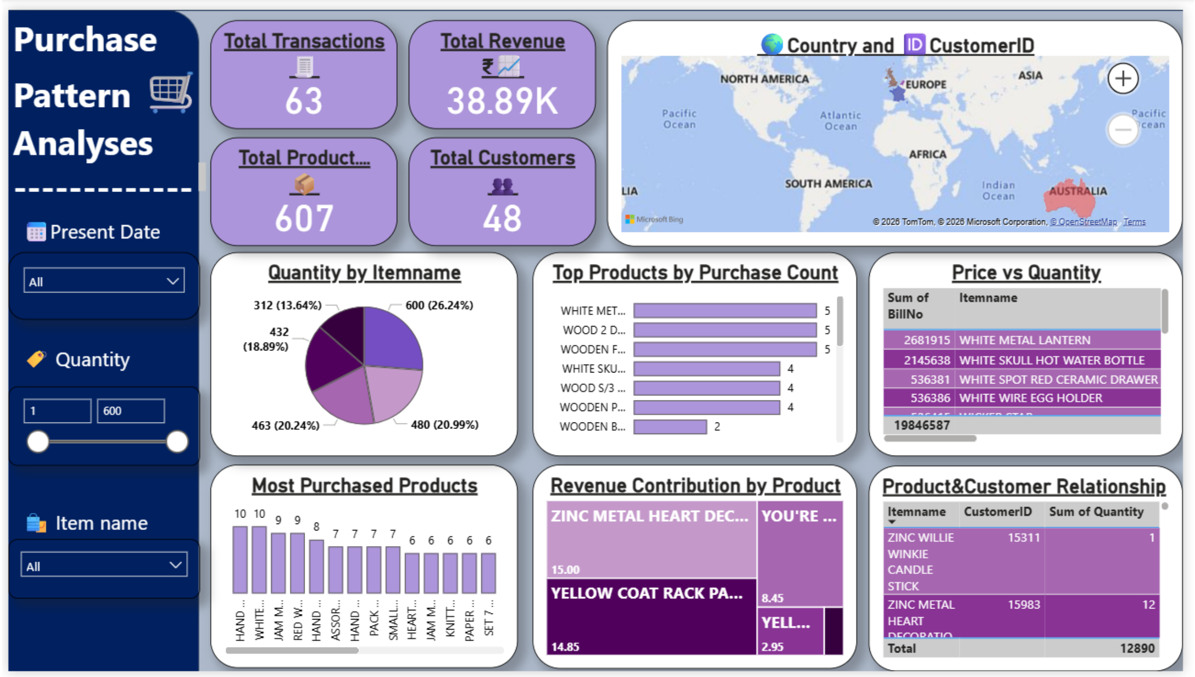
---

7. Exporting Results for Business Intelligence Tools

The frequent itemsets and association rules are exported as CSV files for further analysis and dashboard creation in tools like **Power BI**.

Purpose:
To enable interactive exploration and presentation of insights in enterprise BI platforms.

4). Power BI Dashboard Development



Price vs Quantity

Sum of BillNo	Itemname
2681915	WHITE METAL LANTERN
2145638	WHITE SKULL HOT WATER BOTTLE
536381	WHITE SPOT RED CERAMIC DRAWER
536386	WHITE WIRE EGG HOLDER
19846587	

Product&Customer Relationship

Itemname	CustomerID	Sum of Quantity
ZINC WILLIE WINKIE CANDLE STICK	15311	1
ZINC METAL HEART DECORATIO	15983	12
Total		12890

1.Executive KPIs

- **Total Transactions (63):** Overall sales activity level in the selected period/filters.
- **Total Revenue (₹38.89K):** Business performance indicator; used to track growth and profitability.
- **Total Products (607):** Product diversity sold; highlights catalog breadth and demand spread.
- **Total Customers (48):** Active customer base; helps assess repeat vs. new buyers.

Analyst view: These KPIs give an at-a-glance health check. Any change in slicers instantly shows impact on revenue, demand, and customer reach.

2.Geographic & Customer Distribution (Map)

- **Country vs CustomerID:** Visualizes where customers are located and how sales are geographically distributed.

Analyst view: Identifies high-performing regions and markets with expansion potential. Useful for regional targeting and logistics planning.

3.Product Demand Analysis

Quantity by Item Name (Pie)

- Shows how total quantity is split across top products.

Insight: Quickly reveals **high-volume products** and dependency on a few items.

Top Products by Purchase Count (Bar)

- Ranks products by how often they are purchased.

Insight: Distinguishes **frequently bought** items from occasional purchases—useful for promotions and shelf prioritization.

4.Price vs Quantity

- Compares items based on **transaction frequency (BillNo)** against **item price**.

Analyst view: Helps classify products into:

- High-price / low-quantity (premium items)
- Low-price / high-quantity (volume drivers)
- This supports pricing strategy and margin optimization.

5.Most Purchased Products

- Highlights items with the highest total purchases.

Analyst view: These are **anchor products**-ideal candidates for:

- Bundling
- Cross-selling (used later with Apriori results)
- Inventory prioritization

6.Revenue Contribution by Product (Treemap)

- Shows which products contribute most to total revenue.

Analyst view: Identifies **revenue concentration**. A small number of products often drive a large share of revenue (Pareto effect).

7.Product & Customer Relationship (Table)

- Links **Itemname → CustomerID → Quantity**.

Analyst view: Useful for:

- Identifying loyal customers
- Detecting bulk buyers
- Customer-product affinity analysis

8.Interactive Slicers

Date, Quantity, Item Name filters

Analyst view: Enables **what-if analysis**-e.g., how revenue changes when focusing on high-quantity purchases or specific products.

5.Insights & Business Recommendations:

1. **Revenue is driven by a small set of products**: - The revenue contribution analysis shows that a limited number of product account for a significant portion of total revenue, indicating a concentration of sales around high-performing items.
2. **Certain products are purchased frequently across transactions: -** The top products by purchase count reveal consistent customer demand, identifying these items as anchor products in the overall sales strategy.

3. **High-value transactions involve higher quantities and prices: -** Transactions with higher quantities and premium prices contribute disproportionately to total revenue, highlighting opportunities to target bulk and premium buyers.
4. **Customer purchasing behavior varies by product category: -** The product-customer relationship analysis indicates that some customers repeatedly purchase specific products, suggesting strong customer-product affinity.
5. **Geographic concentration of customers exists: -** The country-wise distribution shows that sales are concentrated in specific regions, while other regions remain under-penetrated.
6. **Price and quantity show distinct buying patterns: -** The price vs quantity analysis reveals both volume-driven products and high-margin products, requiring different sales and marketing approaches.
7. **Product associations enable cross-selling opportunities: -** Results from the Apriori algorithm confirm that certain products are frequently purchased together, providing a foundation for bundle and recommendation strategies.

Business Recommendations

1. **Introduce product bundling strategies: -** Bundle frequently associated products identified through Apriori analysis to increase average basket size and overall revenue.
2. **Focus inventory on high-demand products: -** Ensure consistent stock availability for top-selling and high-revenue products to avoid missed sales opportunities.
3. **Design targeted promotions for high-value customers: -** Offer exclusive deals or loyalty programs to customers involved in high-quantity or high-price transactions.
4. **Apply region-specific marketing strategies: -** Strengthen marketing efforts in high-performing regions while developing targeted campaigns to grow sales in under-performing locations.
5. **Use differentiated pricing strategies: -** Apply volume discounts for frequently purchased items and maintain premium pricing for high-margin products.
6. **Leverage insights for personalized recommendations: -** Use association rules to recommend complementary products during checkout, improving cross-sell conversion rates.

Conclusion

This project successfully demonstrates how transactional purchase data can be transformed into meaningful business insights using a combination of SQL, Power BI, and Python-based analytics. SQL enabled efficient data cleaning and structured analysis, ensuring data accuracy and reliability. Power BI dashboards provided an interactive and intuitive view of sales performance, customer behavior, and product demand, allowing stakeholders to quickly identify trends and key performance drivers.

The implementation of the Apriori algorithm further enhanced the analysis by uncovering hidden relationships between products and identifying frequently purchased item combinations. These insights support data-driven decisions such as product bundling, cross-selling, inventory optimization, and targeted marketing strategies. Overall, the project highlights the value of integrating data analysis, visualization, and machine learning techniques to support strategic business decision-making and improve retail performance.