Project Title : # Market Basket Insights

Problem Statement : Unveiling Customer Behaviour through Association Analysis: Utilize market basket analysis on the provided dataset to uncover hidden patterns and associations between products, aiming to understand customer purchasing behaviour and identify potential cross-selling opportunities for the retail business.

Phase 3 : ## Development Part - 1

Start the market basket insights project by loading and preprocessing the transaction data .

Introduction :

We will explore the fascinating world of Market Basket Analysis using a real-world dataset. Market Basket Analysis is a powerful technique that allows us to uncover patterns and associations between items that customers tend to purchase together. By analyzing these patterns, we can gain valuable insights that can drive business decisions and strategies.

We will work with a Market Basket dataset that captures customer transactions in a retail or e-commerce setting. The dataset provides a wealth of information about customer purchases, allowing us to dive deep into their buying behaviour. By leveraging data mining techniques and association rule mining algorithms, we will unravel the relationships between items and discover interesting patterns.Through this analysis, we can derive actionable insights to improve various aspects of business operations. We can identify frequently co-purchased items, enabling us to make targeted product recommendations and enhance cross-selling and upselling opportunities. By optimizing product placement and store layout based on association patterns, we can create more enticing shopping experiences. Furthermore, we can design effective promotional campaigns by leveraging the discovered item associations, resulting in higher customer engagement and increased sales.

We will take you through the entire process of Market Basket Analysis, from data preprocessing to association rule mining and visualization. By following along with the provided code and explanations, you will gain a solid understanding of how to extract valuable insights from Market Basket datasets and apply them to real-world scenarios.

So let's dive in and unlock the secrets hidden within the Market Basket dataset to gain a deeper understanding of customer behaviour and optimize business strategies !!!

Overview of the Market Basket Analysis dataset :

This dataset contains 522,065 rows and 7 attributes that provide valuable information about customer transactions and product details. Here is a breakdown of the attributes:

BillNo: This attribute represents a 6-digit number assigned to each transaction. It serves as a unique identifier for identifying individual purchases.

Itemname: This attribute stores the name of the product purchased in each transaction. It provides nominal data representing different products.

Quantity: This attribute captures the quantity of each product purchased in a transaction. It is a numeric value that indicates the number of units of a specific item.

Date: The Date attribute records the day and time when each transaction occurred. It provides valuable information about the timing of purchases.

Price: This attribute represents the price of each product. It is a numeric value that indicates the cost of a single unit of the item.

CustomerID: Each customer is assigned a 5-digit number as their unique identifier. This attribute helps track customer-specific information and analyze individual buying patterns.

Country: The Country attribute denotes the name of the country where each customer resides. It provides nominal data representing different geographic regions.

By analyzing this dataset, we can gain insights into customer purchasing behaviour, identify popular products, examine sales trends over time, and explore the impact of factors such as price and geography on customer preferences. These insights can be used to optimize marketing strategies, improve inventory management, and enhance customer satisfaction.

Importance of loading and preprocessing the dataset in Market Basket Insights :

Loading and preprocessing the dataset is a crucial initial step in any market basket insights project. Here are some key reasons why it's important:

1. **Data Quality Assurance:**

Loading and preprocessing your dataset allows you to assess and ensure the quality of the data. You can identify and handle issues such as missing values, duplicates, and outliers. Poor data quality can lead to inaccurate results and misleading insights in market basket analysis.

2. **Data Consistency:**

Preprocessing ensures that the data is consistent and follows a standardized format. It's important that all transaction records are uniform in structure, making it easier to analyze and draw meaningful insights.

3. **Data Reduction:**

Preprocessing can involve reducing the data's dimensionality by removing irrelevant or redundant information. This can lead to more efficient analysis and reduced **computational requirements, especially for large datasets.**

4. **Data Transformation** :

In market basket analysis, transforming the raw data into a transaction format is essential. This transformation involves grouping transactions by a unique identifier (e.g., transaction ID) and aggregating items into lists. This format is necessary for association rule mining algorithms to discover item co-occurrence patterns.

5. **Item Encoding**:

Categorical data, such as item names, usually needs to be encoded into numerical values to perform mathematical operations and build association rules. Preprocessing involves this encoding step to prepare the data for analysis.

6. **Efficient Analysis**:

Clean, consistent, and properly structured data leads to more efficient and accurate analysis. Preprocessing ensures that you are working with a dataset that is ready for various data mining and machine learning techniques.

7. **Improved Model Performance**:

Proper data preprocessing can lead to improved performance of the market basket analysis models. Removing noise, handling missing values, and ensuring data consistency can result in more reliable insights and stronger association rules.

8. **Time and Resource Savings**:

Preprocessing also involves removing unnecessary data, which can lead to significant time and resource savings during analysis. It makes the entire process more streamlined and manageable.

9. **Better Insights**:

High-quality, well-preprocessed data is more likely to yield meaningful and actionable insights. It allows you to discover hidden patterns and trends in customer purchasing behavior, which can inform marketing strategies, inventory management, and other business decisions.

Loading and preprocessing the dataset is a fundamental step in market basket insights projects. It ensures data quality, consistency, and suitability for association rule mining, leading to more accurate and valuable insights that can drive business decisions and strategies.

Data Preprocessing

Importing Required Libraries

```
import numpy as np  # Import numpy library for efficient array operations
import pandas as pd  # Import pandas library for data processing
import matplotlib.pyplot as plt  # Import matplotlib.pyplot for data visualization
```

Data Loading

Retrieving and Loading the Dataset

```
df = pd.read_csv('../input/market-basket-analysis/Assignment-1_Data.csv', sep=';',parse_dates=['Date'])
```

```
df.head()
```

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-01-12 08:26:00	2,55	17850.0	United Kingdom
1	536365	WHITE METAL LANTERN	6	2010-01-12 08:26:00	3,39	17850.0	United Kingdom
2	536365	CREAM CUPID HEARTS COAT HANGER	8	2010-01-12 08:26:00	2,75	17850.0	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-01-12 08:26:00	3,39	17850.0	United Kingdom
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	2010-01-12 08:26:00	3,39	17850.0	United Kingdom

```
# Convert the 'Price' column to float64 data type after replacing commas with dots
df['Price'] = df['Price'].str.replace(',', '.').astype('float64')
```

 # Display the information about the DataFrame which is to provide an overview of the DataFrame's structure and column data types.

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 522064 entries, 0 to 522063
Data columns (total 7 columns):

 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   BillNo      522064 non-null  object
 1   Itemname    520609 non-null  object
 2   Quantity    522064 non-null  int64
 3   Date        522064 non-null  datetime64[ns]
 4   Price       522064 non-null  float64
 5   CustomerID  388023 non-null  float64
 6   Country     522064 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 27.9+ MB
```

 # Calculate the number of missing values for each column and sort them in descending order

```
df.isna().sum().sort_values(ascending=False)

CustomerID    134041
Itemname        1455
BillNo             0
Quantity           0
Date               0
Price              0
Country            0
dtype: int64
```

 # Calculate the total price by multiplying the quantity and price columns

```
df['Total_Price'] = df.Quantity * df.Price
df.describe(include='all')
```

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country	Total_Price
count	522064.0	520609	522064.000000	522064	522064.000000	388023.000000	522064	522064.000000
unique	21665.0	4185	NaN	19641	NaN	NaN	30	NaN
top	573585.0	WHITE HANGING HEART T-LIGHT HOLDER	NaN	2011-10-31 14:41:00	NaN	NaN	United Kingdom	NaN

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country	Total_Price
freq	1114.0	2269	NaN	1114	NaN	NaN	487622	NaN
first	NaN	NaN	NaN	2010-01-12 08:26:00	NaN	NaN	NaN	NaN
last	NaN	NaN	NaN	2011-12-10 17:19:00	NaN	NaN	NaN	NaN
mean	NaN	NaN	10.090435	NaN	3.826801	15316.931710	NaN	19.690633
std	NaN	NaN	161.110525	NaN	41.900599	1721.846964	NaN	273.068938
min	NaN	NaN	-9600.000000	NaN	-11062.060000	12346.000000	NaN	-11062.060000
25%	NaN	NaN	1.000000	NaN	1.250000	13950.000000	NaN	3.750000
50%	NaN	NaN	3.000000	NaN	2.080000	15265.000000	NaN	9.780000
75%	NaN	NaN	10.000000	NaN	4.130000	16837.000000	NaN	17.400000
max	NaN	NaN	80995.000000	NaN	13541.330000	18287.000000	NaN	168469.600000

```
# Print the number of unique countries in the 'Country' column
print("Number of unique countries:", df['Country'].nunique())

# Calculate and print the normalized value counts of the top 5 countries in the 'Country' column
print(df['Country'].value_counts(normalize=True)[:5])
```

```
Number of unique countries: 30
United Kingdom    0.934027
Germany           0.017320
France            0.016105
Spain             0.004760
Netherlands       0.004526
Name: Country, dtype: float64
```

Considering that the majority of transactions (approximately 93%) in the dataset originate from the UK, the 'Country' column may not contribute significant diversity or variability to the analysis. Therefore, we can choose to remove the 'Country' column from the DataFrame df. we indicate that we want to drop a column, This step allows us to focus on other attributes that may provide more valuable insights for our analysis.

```
# Delete the 'Country' column from the DataFrame
df.drop('Country', axis=1, inplace=True)
```

```
# Filter the DataFrame to display rows where 'BillNo' column contains non-digit values
df[df['BillNo'].str.isdigit() == False]
```

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
288772	A563185	Adjust bad debt	1	2011-12-08 14:50:00	11062.06	NaN	11062.06
288773	A563186	Adjust bad debt	1	2011-12-08 14:51:00	-11062.06	NaN	-11062.06
288774	A563187	Adjust bad debt	1	2011-12-08 14:52:00	-11062.06	NaN	-11062.06

Since the item name "Adjust bad debt" was filled accidentally and does not provide any useful information for our analysis, we can choose to remove the corresponding rows from the DataFrame. The code snippet above filters the DataFrame df to retain only the rows where the 'Itemname' column does not contain the value "Adjust bad debt". This operation effectively eliminates the rows associated with the accidental data entry, ensuring the dataset is free from this irrelevant item name.

```
# Remove rows where the 'Itemname' column contains "Adjust bad debt"
df = df[df['Itemname'] != "Adjust bad debt"]
```

```
# Here to check if all BillNo doesn't inculde letters
df['BillNo'].astype("int64")
```

```
0       536365
1       536365
2       536365
3       536365
4       536365
         ...
522059   581587
522060   581587
522061   581587
```

```
522062    581587
522063    581587
Name: BillNo, Length: 522061, dtype: int64
```

```
# Calculate the sum of 'Price' for rows where 'Itemname' is missing
df[df['Itemname'].isna()] ['Price'].sum()
```

```
0.0
```

Exploring Rows with Missing Item Names:

To investigate the data where the 'Itemname' column has missing values, we can filter the dataset to display only those rows. This subset of the data will provide insights into the records where the item names are not available.

```
# Filter the DataFrame to display rows where 'Itemname' is missing
df[df['Itemname'].isna()]
```

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
613	536414	NaN	56	2010-01-12 11:52:00	0.0	NaN	0.0
1937	536545	NaN	1	2010-01-12 14:32:00	0.0	NaN	0.0
1938	536546	NaN	1	2010-01-12 14:33:00	0.0	NaN	0.0
1939	536547	NaN	1	2010-01-12 14:33:00	0.0	NaN	0.0
1940	536549	NaN	1	2010-01-12 14:34:00	0.0	NaN	0.0
...	...	...	...	...	...	...	...
515623	581199	NaN	-2	2011-07-12 18:26:00	0.0	NaN	-0.0
515627	581203	NaN	15	2011-07-12 18:31:00	0.0	NaN	0.0
515633	581209	NaN	6	2011-07-12 18:35:00	0.0	NaN	0.0
517266	581234	NaN	27	2011-08-12 10:33:00	0.0	NaN	0.0

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
518820	581408	NaN	20	2011-08-12 14:06:00	0.0	NaN	0.0

1455 rows × 7 columns

Upon examining the data where the 'Itemname' column has missing values, it becomes evident that these missing entries do not contribute any meaningful information. Given that the item names are not available for these records, it suggests that these instances may not be crucial for our analysis. As a result, we can consider these missing values as non-significant and proceed with our analysis without incorporating them.

```
# Filter the DataFrame to exclude rows where 'Itemname' is missing (not NaN)
df = df[df['Itemname'].notna()]

# Print the number of unique items in the 'Itemname' column
print("Number of unique items:", df['Itemname'].nunique())

# Calculate and print the normalized value counts of the top 5 items in the 'Itemname' column
print(df['Itemname'].value_counts(normalize=True)[:5])
Number of unique items: 4184
WHITE HANGING HEART T-LIGHT HOLDER    0.004358
JUMBO BAG RED RETROSPOT               0.004009
REGENCY CAKESTAND 3 TIER              0.003707
PARTY BUNTING                         0.003221
LUNCH BAG RED RETROSPOT               0.003016
Name: Itemname, dtype: float64
```

A curious observation has caught our attention—the presence of a negative quantity in the 515,623rd row.

we are intrigued by the existence of negative quantities within the dataset. To gain a deeper understanding of this phenomenon, we focus our attention on these specific instances and aim to uncover the underlying reasons behind their occurrence. Through this exploration, we expect to gain valuable insights into the nature of these negative quantities and their potential impact on our analysis. Our investigation aims to reveal the intriguing stories that lie within this aspect of the data.

```
# Filter the DataFrame to display rows where 'Quantity' is less than 1
df[df['Quantity'] < 1]
```

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
7122	537032	?	-30	2010-03-12 16:50:00	0.0	NaN	-0.0
12926	537425	check	-20	2010-06-12 15:35:00	0.0	NaN	-0.0
12927	537426	check	-35	2010-06-12 15:36:00	0.0	NaN	-0.0
12973	537432	damages	-43	2010-06-12 16:10:00	0.0	NaN	-0.0
20844	538072	faulty	-13	2010-09-12 14:10:00	0.0	NaN	-0.0
...	...	...	...	...	...	...	...
515634	581210	check	-26	2011-07-12 18:36:00	0.0	NaN	-0.0
515636	581212	lost	-1050	2011-07-12 18:38:00	0.0	NaN	-0.0
515637	581213	check	-30	2011-07-12 18:38:00	0.0	NaN	-0.0
517209	581226	missing	-338	2011-08-12 09:56:00	0.0	NaN	-0.0
519172	581422	smashed	-235	2011-08-12 15:24:00	0.0	NaN	-0.0

473 rows × 7 columns

Given the observation that negative quantities might be filled with system issues or irrelevant information for our analysis, it is reasonable to proceed with removing these rows from the dataset. By doing so, we can ensure the accuracy and reliability of our data, as well as eliminate potential biases or misleading information stemming from negative quantities.

```
# Remove rows where 'Quantity' is less than 1
df = df[df['Quantity'] >= 1]
```

Next, we turn our attention to the presence of missing values in the 'CustomerID' column. By investigating these missing values, we aim to identify any potential issues or data quality concerns associated with them. Analyzing the impact of missing 'CustomerID' values will help us assess the completeness and reliability of the dataset, enabling us to make informed decisions on handling or imputing these missing values. Let's dive deeper into this aspect and gain a comprehensive understanding of any issues related to missing 'CustomerID' values.

 # Select a random sample of 30 rows where 'CustomerID' is missing
df[df['CustomerID'].isna()].sample(30)

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
195946	554511	SPOTTY BUNTING	3	2011-05-24 15:52:00	4.95	NaN	14.85
83590	543533	CARROT CHARLIE+LOLA COASTER SET	1	2011-09-02 13:00:00	5.79	NaN	5.79
84884	543660	AGED GLASS SILVER T-LIGHT HOLDER	1	2011-11-02 10:40:00	1.63	NaN	1.63
10795	537240	CALENDAR PAPER CUT DESIGN	2	2010-06-12 10:08:00	5.91	NaN	11.82
64890	541810	JUMBO STORAGE BAG SUKI	4	2011-01-21 15:00:00	4.13	NaN	16.52
244338	559163	GREEN DIAMANTE PEN IN GIFT BOX	1	2011-06-07 16:33:00	2.46	NaN	2.46
188612	553718	CHOCOLATE THIS WAY METAL SIGN	1	2011-05-18 16:14:00	4.13	NaN	4.13
433816	575177	PACK OF 12 DOLLY GIRL TISSUES	1	2011-08-11 18:41:00	0.83	NaN	0.83
124804	547385	GOLD FISHING GNOME	1	2011-03-22 15:48:00	12.46	NaN	12.46
307100	564840	STRAWBERRY RAFFIA FOOD COVER	5	2011-08-30 12:49:00	3.29	NaN	16.45
20341	538071	GIN AND TONIC MUG	3	2010-09-12 14:09:00	3.36	NaN	10.08
213287	556237	REGENCY TEAPOT ROSES	1	2011-09-06 15:34:00	19.96	NaN	19.96

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
352333	568721	SPACEBOY LUNCH BOX	1	2011-09-28 16:24:00	4.13	NaN	4.13
41964	540026	CARROT CHARLIE+LOLA COASTER SET	1	2011-04-01 13:25:00	3.36	NaN	3.36
151856	550207	EDWARDIAN PARASOL NATURAL	1	2011-04-15 10:38:00	12.46	NaN	12.46
232220	558118	GINGHAM HEART DOORSTOP RED	1	2011-06-27 09:11:00	8.29	NaN	8.29
203161	555278	EASTER DECORATION HANGING BUNNY	1	2011-01-06 17:33:00	1.25	NaN	1.25
472679	578067	OVEN MITT APPLES DESIGN	1	2011-11-22 15:43:00	3.29	NaN	3.29
307124	564840	HOT WATER BOTTLE TEA AND SYMPATHY	1	2011-08-30 12:49:00	8.29	NaN	8.29
424124	574561	RED TOADSTOOL LED NIGHT LIGHT	5	2011-04-11 15:52:00	3.29	NaN	16.45
170326	552000	PINK VINTAGE PAISLEY PICNIC BAG	1	2011-05-05 15:56:00	2.46	NaN	2.46
79639	543182	PACK OF 20 NAPKINS PANTRY DESIGN	1	2011-04-02 10:40:00	1.63	NaN	1.63
191588	554054	TOY TIDY SPACEBOY	2	2011-05-20 15:29:00	4.13	NaN	8.26
253520	559923	SET 12 KIDS COLOUR CHALK STICKS	2	2011-07-13 16:07:00	0.83	NaN	1.66
474316	578149	DOORMAT KEEP CALM AND COME IN	6	2011-11-23 11:11:00	15.79	NaN	94.74

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
443582	575930	BUNDLE OF 3 RETRO NOTE BOOKS	1	2011-11-11 17:58:00	3.29	NaN	3.29
242810	559055	FOLDING UMBRELLA CHOCOLATE POLKADOT	2	2011-05-07 17:09:00	3.29	NaN	6.58
433704	575177	FELTCRAFT PRINCESS LOLA DOLL	1	2011-08-11 18:41:00	7.46	NaN	7.46
272207	561651	3 HOOK HANGER MAGIC GARDEN	1	2011-07-28 15:36:00	4.13	NaN	4.13
45241	540355	DINOSAURS WRITING SET	4	2011-06-01 15:11:00	3.36	NaN	13.44

This sample can provide us with a glimpse into the specific instances where 'CustomerID' is missing, aiding us in further analysis or decision-making related to handling these missing values.

Upon analyzing a sample of rows where the 'CustomerID' is missing, it appears that there is no discernible pattern or specific reason behind the absence of these values. This observation suggests that the missing 'CustomerID' entries were not filled accidentally or due to a systematic issue. Instead, it is possible that these missing values occur naturally in the dataset, without any particular significance or underlying cause.

Identifying Issues in the Price Column: Ensuring Data Quality

In our analysis, we shift our focus to the 'Price' column and investigate it for any potential issues or anomalies. By thoroughly examining the data within this column, we aim to identify any irregularities, inconsistencies, or outliers that may affect the overall quality and integrity of the dataset. Analyzing the 'Price' column is crucial in ensuring accurate and reliable pricing information for our analysis. Let's dive deeper into the 'Price' column and uncover any issues that may require attention.

```
# Counting the number of rows where the price is zero
zero_price_count = len(df[df['Price'] == 0])
print("Number of rows where price is zero:", zero_price_count)

# Counting the number of rows where the price is negative
negative_price_count = len(df[df['Price'] < 0])
print("Number of rows where price is negative:", negative_price_count)
```

```
Number of rows where price is zero: 583
```

Number of rows where price is negative: 0

our attention now turns to the presence of zero charges in the 'Price' column. It is important to explore instances where products were offered free of cost, as this information can provide valuable insights into promotional activities, giveaways, or other unique aspects of the dataset. By examining the data related to zero charges in the 'Price' column, we can gain a deeper understanding of these transactions and their potential impact on our analysis. Let's delve into the details of these zero-priced transactions and uncover any significant findings.

```
# Selecting a random sample of 20 rows where the price is zero
df[df['Price'] == 0].sample(20)
```

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
40262	539856	FRENCH BLUE METAL DOOR SIGN 4	2	2010-12-22 14:41:00	0.0	NaN	0.0
6275	536941	amazon	20	2010-03-12 12:08:00	0.0	NaN	0.0
14051	537534	RED KITCHEN SCALES	1	2010-07-12 11:48:00	0.0	NaN	0.0
14054	537534	CHILDS GARDEN TROWEL BLUE	1	2010-07-12 11:48:00	0.0	NaN	0.0
402035	572725	Found	57	2011-10-25 15:14:00	0.0	NaN	0.0
331148	566983	Amazon	1	2011-09-16 10:16:00	0.0	NaN	0.0
478778	578374	found	50	2011-11-24 11:21:00	0.0	NaN	0.0
301866	564530	TOADSTOOL MONEY BOX	1	2011-08-25 14:57:00	0.0	NaN	0.0
287200	563015	POLYESTER FILLER PAD 40x40cm	220	2011-11-08 12:24:00	0.0	NaN	0.0

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Total_Price
119494	546933	RECIPE BOX PANTRY YELLOW DESIGN	9	2011-03-18 11:02:00	0.0	NaN	0.0
100983	545160	MINT KITCHEN SCALES	1	2011-02-28 13:31:00	0.0	NaN	0.0
14087	537534	HOLIDAY FUN LUDO	1	2010-07-12 11:48:00	0.0	NaN	0.0
446896	576109	adjustment	180	2011-11-14 10:33:00	0.0	NaN	0.0
502098	580366	FRIDGE MAGNETS LES ENFANTS ASSORTED	6	2011-02-12 16:38:00	0.0	NaN	0.0
40257	539856	FRENCH BLUE METAL DOOR SIGN 9	1	2010-12-22 14:41:00	0.0	NaN	0.0
14093	537534	BLUE POLKADOT LUGGAGE TAG	1	2010-07-12 11:48:00	0.0	NaN	0.0
186570	553521	FRENCH BLUE METAL DOOR SIGN 8	5	2011-05-17 14:35:00	0.0	NaN	0.0
429045	574879	RED KITCHEN SCALES	2	2011-07-11 13:22:00	0.0	13014.0	0.0
45228	540355	RED RETROSPOT CHARLOTTE BAG	1	2011-06-01 15:11:00	0.0	NaN	0.0
14045	537534	FRENCH BLUE METAL DOOR SIGN 5	1	2010-07-12 11:48:00	0.0	NaN	0.0

Removing Rows with Zero Price: Eliminating Misleading Data Entries

Upon reviewing the sample of rows where the price is zero, we have identified that these entries might provide misleading or inaccurate information for our analysis. Therefore, it is prudent to proceed with removing these rows from the dataset to ensure the integrity and reliability of our analysis.

```
# Remove rows where the price is zero
df = df[df['Price'] != 0]
```

Data Understanding: Exploring and Interpreting the Dataset

In the data analysis process, data understanding plays a crucial role in gaining insights and formulating meaningful conclusions. By thoroughly examining the dataset, we aim to understand its structure, contents, and underlying patterns. This understanding empowers us to make informed decisions regarding data cleaning, feature engineering, and subsequent analysis steps.

Key aspects of data understanding include:

- Exploring the Dataset: We investigate the dataset's dimensions, such as the number of rows and columns, to gauge its size and complexity. Additionally, we examine the data types of each column to understand the nature of the variables.

- Assessing Data Quality: We scrutinize the data for inconsistencies, outliers, or other data quality issues that may require attention. Addressing these issues ensures the reliability and accuracy of the data.

- Identifying Relationships: We analyze the relationships between variables by examining correlations, associations, or dependencies. This analysis allows us to uncover meaningful connections that can drive insights and guide our analysis.

- Detecting Patterns and Trends: We look for recurring patterns, trends, or distributions within the data. This step can reveal valuable information about customer behavior, market dynamics, or other relevant factors.

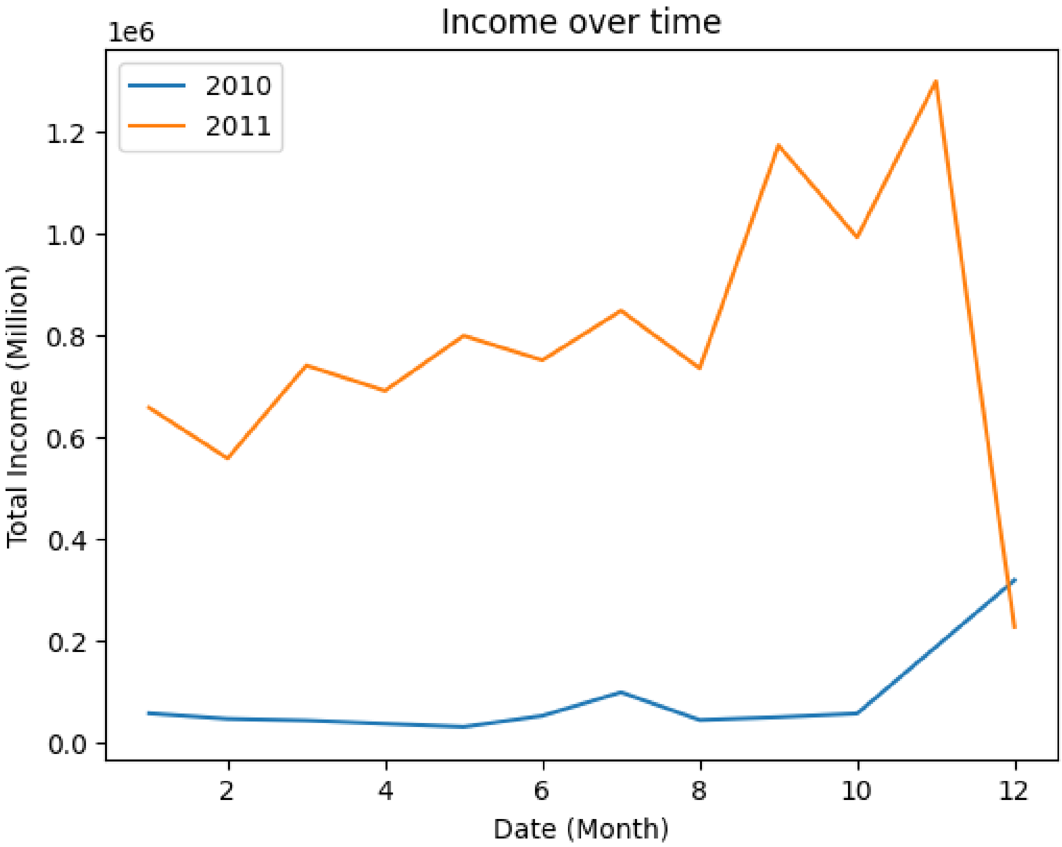
By thoroughly understanding the dataset, we lay the foundation for meaningful data analysis and generate insights that contribute to informed decision-making and problem-solving.

```
# Grouping the data by month and summing the total price for the year 2010
df[df["Date"].dt.year == 2010].groupby(df["Date"].dt.month)["Total_Price"].sum().plot()

# Grouping the data by month and summing the total price for the year 2011
df[df["Date"].dt.year == 2011].groupby(df["Date"].dt.month)["Total_Price"].sum().plot()

# Adding legend and plot labels
plt.legend(["2010", "2011"])
plt.title("Income over time")
plt.ylabel('Total Income (Million)')
plt.xlabel("Date (Month)")
```

```
Text(0.5, 0, 'Date (Month)')
```



The code snippet above creates a line plot to visualize the income over time for the years 2010 and 2011. First, the data is filtered based on the year using the dt.year attribute of the 'Date' column. The data is then grouped by month, and the 'Total_Price' column is summed. Two line plots are created, one for each year, showing the monthly total income. The legend is added to indicate the respective years, and the plot is labeled with a title, y-axis label, and x-axis label. This visualization allows us to observe the trend and compare the income between the two years.

Upon observing the line plot of income over time for the years 2010 and 2011, it becomes apparent that the sales remained relatively stable and consistent until October 2010. This suggests that the business was growing steadily during this period, as the sales continued to increase.

However, a significant drop in sales is observed in the last month of the dataset. This sudden decline indicates a notable deviation from the previously observed growth trend. Exploring the potential factors contributing to this drop becomes crucial in understanding the underlying reasons for the decline in sales during that specific period.

To verify if the data is complete for the entire last month in the dataset, we can compare the maximum date in the 'Date' column with the last day of that month. If they match, it indicates that the data is filled for the entire last month.

```
df["Date"].max()
```

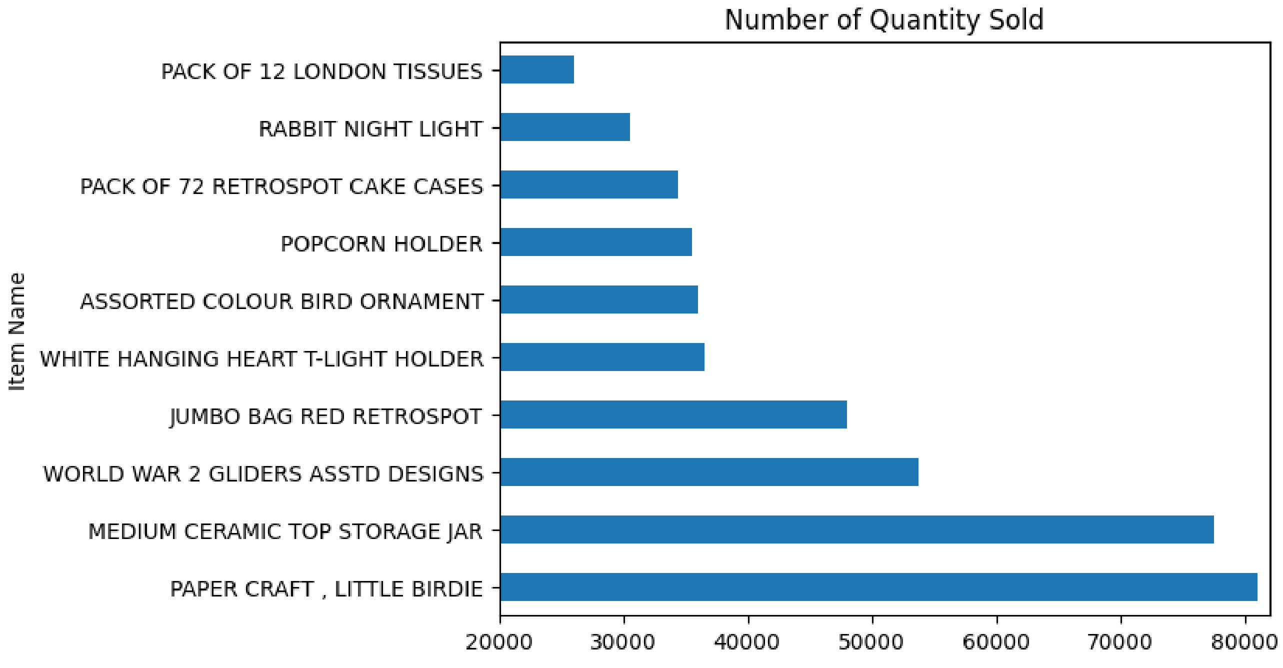
```
Timestamp('2011-12-10 17:19:00')
```

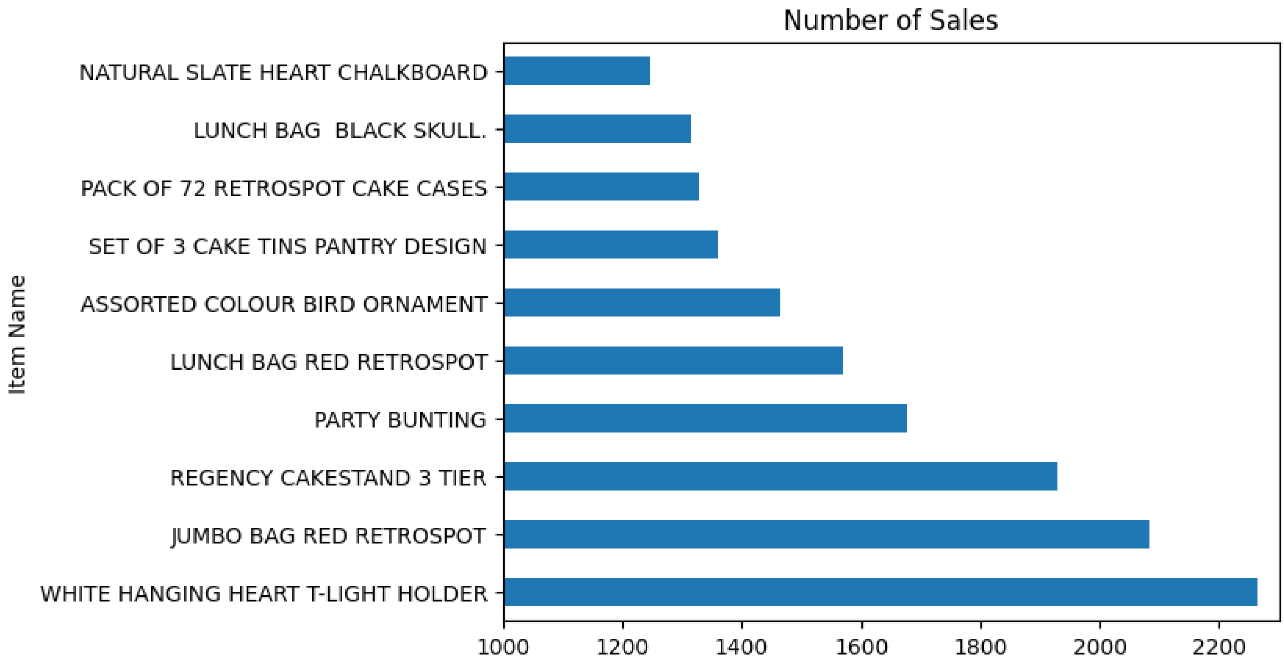
Based on the finding that the data is only available for 10 days in the last month, it becomes evident that the significant drop in sales observed during that period is likely due to the limited data rather than an actual decline in sales. The incomplete data for the last month may not provide a comprehensive representation of the sales performance during that period.

To gain a more accurate understanding of the sales trend, it is advisable to consider a broader time frame with complete data. Analyzing a more extended period that encompasses multiple months or years would provide a more reliable assessment of the sales performance and allow for more meaningful insights and conclusions.

```
# Plotting the top 10 most sold products by quantity
df.groupby('Itemname')['Quantity'].sum().sort_values(ascending=False)[:10].plot(kind='barh', title='Number of Quantity Sold')
plt.ylabel('Item Name')
plt.xlim(20000, 82000)
plt.show()

# Plotting the top 10 most sold products by count
df['Itemname'].value_counts(ascending=False)[:10].plot(kind='barh', title='Number of Sales')
plt.ylabel('Item Name')
plt.xlim(1000, 2300)
plt.show()
```





The code snippet above creates two horizontal bar plots to visualize the most sold products based on quantity and count, respectively.

In the first plot, the top 10 items are determined by summing the 'Quantity' column for each unique 'Itemname' and sorting them in descending order. The plot displays the number of quantities sold for each item.

The second plot showcases the top 10 items based on the count of sales for each unique 'Itemname'. The value_counts function counts the occurrences of each item and sorts them in descending order. The plot represents the number of times each item has been sold.

Observing the plots, we can infer that there are products that are sold more frequently (higher count) compared to others, despite having relatively lower quantities sold per transaction. This indicates the presence of items that are commonly purchased in larger quantities at once. These products might include items that are frequently bought in bulk or items that are typically sold in larger packages or quantities.

This insight highlights the importance of considering both the quantity sold and the count of sales when analyzing the popularity and demand for different products. It suggests that some items may have a higher turnover rate due to frequent purchases, while others may have a higher quantity per sale, leading to different sales patterns and customer behaviors. Understanding these dynamics can be valuable for inventory management, pricing strategies, and identifying customer preferences.

Conclusion :

In conclusion, the Market Basket Insights project has provided valuable and actionable information to help businesses understand customer behavior and improve their operations. By analyzing transaction data and identifying patterns and associations between product purchases.

Overall, the Market Basket Insights project empowers businesses with the tools to make data-driven decisions that enhance customer satisfaction, increase revenue, and drive sustainable growth. It is an essential component in the era of data-driven business intelligence and analytics.

Thank You