The data contains transaction details with 7501 rows. Each row is a grocery transaction from the market with the unique items list for each transaction.

1. Read the data and remove all null or empty values.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('Dataset_Day15.csv')
df.info()
# need to preprocess the data and convert it
into a list of lists (transactions) before
proceeding with the steps
# Convert each row into a single transaction (a
list of items)
transactions = df.values.tolist()
# Display the first few transactions to check
if they are correctly converted
print(transactions[:2])
# Remove all null or empty values from the list
of lists
transactions = [[item for item in transaction
if pd.notna(item)] for transaction in
transactions]
print(transactions[:2])
```

File - Day15Q1

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\tejas\PycharmProjects\pythonProject\START\Day15Q1.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7500 entries, 0 to 7499
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   shrimp             7500 non-null   object
 1   almonds            5746 non-null   object
 2   avocado            4388 non-null   object
 3   vegetables mix     3344 non-null   object
 4   green grapes       2528 non-null   object
 5   whole weat flour   1863 non-null   object
 6   yams               1368 non-null   object
 7   cottage cheese     980 non-null    object
 8   energy drink       653 non-null    object
 9   tomato juice       394 non-null    object
 10  low fat yogurt     255 non-null    object
 11  green tea          153 non-null    object
 12  honey              86 non-null     object
 13  salad              46 non-null     object
 14  mineral water      24 non-null     object
 15  salmon             7 non-null      object
 16  antioxydant juice  3 non-null      object
 17  frozen smoothie    3 non-null      object
 18  spinach            2 non-null      object
 19  olive oil          0 non-null      float64
dtypes: float64(1), object(19)
memory usage: 1.1+ MB
[['burgers', 'meatballs', 'eggs', nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan], ['chutney', nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]]
[['burgers', 'meatballs', 'eggs'], ['chutney']]

Process finished with exit code 0

```

2. Transform the data using TransactionEncoder to perform Market Basket Analysis.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from mlxtend.preprocessing import TransactionEncoder

# Load the dataset
df = pd.read_csv('Dataset_Day15.csv')
df.info()
# need to preprocess the data and convert it into a list of lists (transactions) before proceeding with the steps
# Convert each row into a single transaction (a list of items)
```

```
transactions = df.values.tolist()
# Display the first few transactions to check if
they are correctly converted
print(transactions[:2])
# Remove all null or empty values from the list
of lists
transactions = [[item for item in transaction if
pd.notna(item)] for transaction in transactions]
print(transactions[:2])
# converts the list of lists (transactions) into
a one-hot encoded DataFrame, where each column
represents an item, and each row represents a
transaction
# Initialize TransactionEncoder
te = TransactionEncoder()
# Fit and transform the transactions to one-hot
encoded DataFrame
onehot_encoded = te.fit_transform(transactions)
# Convert the one-hot encoded array to a
DataFrame
df_onehot = pd.DataFrame(onehot_encoded,
columns=te.columns_)
# Display the first few rows of the one-hot
encoded DataFrame
print(df_onehot.head())
```

File - Day15Q2

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day15Q2.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7500 entries, 0 to 7499
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   shrimp             7500 non-null   object
 1   almonds            5746 non-null   object
 2   avocado            4388 non-null   object
 3   vegetables mix     3344 non-null   object
 4   green grapes       2528 non-null   object
 5   whole weat flour   1863 non-null   object
 6   yams               1368 non-null   object
 7   cottage cheese     980 non-null    object
 8   energy drink       653 non-null    object
 9   tomato juice       394 non-null    object
 10  low fat yogurt     255 non-null    object
 11  green tea          153 non-null    object
 12  honey              86 non-null     object
 13  salad              46 non-null     object
 14  mineral water      24 non-null     object
 15  salmon             7 non-null      object
 16  antioxydant juice  3 non-null      object
 17  frozen smoothie    3 non-null      object
 18  spinach            2 non-null      object
 19  olive oil          0 non-null      float64
dtypes: float64(1), object(19)
memory usage: 1.1+ MB
[['burgers', 'meatballs', 'eggs', nan, nan, nan, nan, nan, nan, nan, nan, nan,
nan, nan, nan, nan, nan, nan, nan, nan], ['chutney', nan, nan, nan, nan, nan,
nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]]
[['burgers', 'meatballs', 'eggs'], ['chutney']]
   asparagus  almonds  antioxydant juice  ...   yams  yogurt cake  zucchini
0      False    False              False  ...  False        False     False
1      False    False              False  ...  False        False     False
2      False    False              False  ...  False        False     False
3      False    False              False  ...  False        False     False
4      False    False              False  ...  False        False     False

[5 rows x 120 columns]

Process finished with exit code 0
```

3. Use min_support = 0.02 to find frequent itemsets.

   INSIGHTS: Avocado is present in 3.3% of the transaction, Almonds are present is 2.0% of the transaction, so forth and so-on .

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from mlxtend.preprocessing import
TransactionEncoder
from mlxtend.frequent_patterns import apriori
```

```python
# Load the dataset
df = pd.read_csv('Dataset_Day15.csv')
df.info()
# need to preprocess the data and convert it into
a list of lists (transactions) before proceeding
with the steps
# Convert each row into a single transaction (a
list of items)
transactions = df.values.tolist()
# Display the first few transactions to check if
they are correctly converted
print(transactions[:2])
# Remove all null or empty values from the list
of lists
transactions = [[item for item in transaction if
pd.notna(item)] for transaction in transactions]
print(transactions[:2])
# converts the list of lists (transactions) into
a one-hot encoded DataFrame, where each column
represents an item, and each row represents a
transaction
# Initialize TransactionEncoder
te = TransactionEncoder()
# Fit and transform the transactions to one-hot
encoded DataFrame
onehot_encoded = te.fit_transform(transactions)
# Convert the one-hot encoded array to a
DataFrame
df_onehot = pd.DataFrame(onehot_encoded,
columns=te.columns_)
print(df_onehot.head())
# Find frequent item sets with minimum support of
0.02
frequent_itemsets = apriori(df_onehot,
min_support=0.02, use_colnames=True)
print(frequent_itemsets)
```

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\tejas\PycharmProjects\pythonProject\START\Day15Q3.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7500 entries, 0 to 7499
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   shrimp             7500 non-null   object
 1   almonds            5746 non-null   object
 2   avocado            4388 non-null   object
 3   vegetables mix     3344 non-null   object
 4   green grapes       2528 non-null   object
 5   whole weat flour   1863 non-null   object
 6   yams               1368 non-null   object
 7   cottage cheese     980 non-null    object
 8   energy drink       653 non-null    object
 9   tomato juice       394 non-null    object
 10  low fat yogurt     255 non-null    object
 11  green tea          153 non-null    object
 12  honey              86 non-null     object
 13  salad              46 non-null     object
 14  mineral water      24 non-null     object
 15  salmon             7 non-null      object
 16  antioxydant juice  3 non-null      object
 17  frozen smoothie    3 non-null      object
 18  spinach            2 non-null      object
 19  olive oil          0 non-null      float64
dtypes: float64(1), object(19)
memory usage: 1.1+ MB
[['burgers', 'meatballs', 'eggs', nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan], ['chutney', nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]]
[['burgers', 'meatballs', 'eggs'], ['chutney']]
    asparagus  almonds  antioxydant juice  ...   yams  yogurt cake  zucchini
0       False    False              False  ...  False        False     False
1       False    False              False  ...  False        False     False
2       False    False              False  ...  False        False     False
3       False    False              False  ...  False        False     False
4       False    False              False  ...  False        False     False

[5 rows x 120 columns]
      support                         itemsets
0    0.020267                        (almonds)
1    0.033200                        (avocado)
2    0.033733                       (brownies)
3    0.087200                        (burgers)
4    0.030133                         (butter)
..        ...                              ...
99   0.020133  (whole wheat rice, mineral water)
100  0.022933            (olive oil, spaghetti)
101  0.025200             (pancakes, spaghetti)
102  0.021200               (shrimp, spaghetti)
103  0.020933             (tomatoes, spaghetti)

[104 rows x 2 columns]

Process finished with exit code 0

```

4. Use the frequent itemsets to create association rules (take min_threshold as 15%) and evaluate the rules for the following metrics

   a. Find top 5 antecedent -> consequent rules based on 'conviction', 'leverage', 'lift'. Explain your findings.

   b. Find top 2 & bottom 2 antecedent -> consequent rules based on 'zhang's metric'. Explain your findings.

   INSIGHTS: Lift Metric-Customers who buy spaghetti are 2.29 times more likely to buy ground meat compared to when ground meat is purchased independently.

   Zhang's Metric: French fries and mineral water have a weak association as their Zhang's metric is -0.2 ,so forth and so on .

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from mlxtend.preprocessing import
TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import
association_rules

# Load the dataset
df = pd.read_csv('Dataset_Day15.csv')
df.info()
# need to preprocess the data and convert it into a
list of lists (transactions) before proceeding with
the steps
# Convert each row into a single transaction (a
list of items)
transactions = df.values.tolist()
# Display the first few transactions to check if
they are correctly converted
print(transactions[:2])
# Remove all null or empty values from the list of
lists
transactions = [[item for item in transaction if
pd.notna(item)] for transaction in transactions]
print(transactions[:2])
# converts the list of lists (transactions) into a
one-hot encoded DataFrame, where each column
represents an item, and each row represents a
```

```
transaction
# Initialize TransactionEncoder
te = TransactionEncoder()
# Fit and transform the transactions to one-hot
encoded DataFrame
onehot_encoded = te.fit_transform(transactions)
# Convert the one-hot encoded array to a DataFrame
df_onehot = pd.DataFrame(onehot_encoded,
columns=te.columns_)
print(df_onehot.head())
# Find frequent item sets with minimum support of
0.02
frequent_itemsets = apriori(df_onehot,
min_support=0.02, use_colnames=True)
print(frequent_itemsets)
# sort the values
frequent_itemsets.sort_values('support',
ascending=False)
# Generate association rules with minimum threshold
of 15%
rules = association_rules(frequent_itemsets,
metric='confidence', min_threshold=0.15)
# top 5 Sort rules based on conviction, leverage,
and lift
top_rules_conviction =
rules.sort_values(by='conviction',
ascending=False).head(5)
top_rules_leverage =
rules.sort_values(by='leverage',
ascending=False).head(5)
top_rules_lift = rules.sort_values(by='lift',
ascending=False).head(5)
# Find top 2 and bottom 2 rules based on Zhang's
metric
top_rules_zhangs =
rules.sort_values(by='zhangs_metric',
ascending=False).head(2)
bottom_rules_zhangs =
rules.sort_values(by='zhangs_metric').head(2)
print("Top 5 Antecedent -> Consequent Rules based
on Conviction:")
print(top_rules_conviction)
```

```
print("Top 5 Antecedent -> Consequent Rules based
on Leverage:")
print(top_rules_leverage)
print("Top 5 Antecedent -> Consequent Rules based
on Lift:")
print(top_rules_lift)
print("Top 2 Antecedent -> Consequent Rules based
on Zhang's Metric:")
print(top_rules_zhangs)
print("Bottom 2 Antecedent -> Consequent Rules
based on Zhang's Metric:")
print(bottom_rules_zhangs)
# Filter the association rules DataFrame based on
'ground meat' and 'spaghetti'
filtered_rules = rules[(rules['antecedents'] ==
{'ground meat'}) & (rules['consequents'] ==
{'spaghetti'})]
# Print the lift value
print("Lift Value for the rule (ground meat ->
spaghetti):")
print(filtered_rules['lift'].values[0])
```

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\tejas\PycharmProjects\pythonProject\START\Day15Q4.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7500 entries, 0 to 7499
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   shrimp             7500 non-null   object
 1   almonds            5746 non-null   object
 2   avocado            4388 non-null   object
 3   vegetables mix     3344 non-null   object
 4   green grapes       2528 non-null   object
 5   whole weat flour   1863 non-null   object
 6   yams               1368 non-null   object
 7   cottage cheese     980 non-null    object
 8   energy drink       653 non-null    object
 9   tomato juice       394 non-null    object
 10  low fat yogurt     255 non-null    object
 11  green tea          153 non-null    object
 12  honey              86 non-null     object
 13  salad              46 non-null     object
 14  mineral water      24 non-null     object
 15  salmon             7 non-null      object
 16  antioxydant juice  3 non-null      object
 17  frozen smoothie    3 non-null      object
 18  spinach            2 non-null      object
 19  olive oil          0 non-null      float64
dtypes: float64(1), object(19)
memory usage: 1.1+ MB
[['burgers', 'meatballs', 'eggs', nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan], ['chutney', nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]]
[['burgers', 'meatballs', 'eggs'], ['chutney']]
    asparagus  almonds  antioxydant juice  ...   yams  yogurt cake  zucchini
0       False    False              False  ...  False        False     False
1       False    False              False  ...  False        False     False
2       False    False              False  ...  False        False     False
3       False    False              False  ...  False        False     False
4       False    False              False  ...  False        False     False

[5 rows x 120 columns]
      support                             itemsets
0    0.020267                            (almonds)
1    0.033200                            (avocado)
2    0.033733                           (brownies)
3    0.087200                            (burgers)
4    0.030133                             (butter)
..        ...                                  ...
99   0.020133  (whole wheat rice, mineral water)
100  0.022933             (olive oil, spaghetti)
101  0.025200              (spaghetti, pancakes)
102  0.021200                (shrimp, spaghetti)
103  0.020933              (tomatoes, spaghetti)

[104 rows x 2 columns]
Top 5 Antecedent -> Consequent Rules based on Conviction:
      antecedents      consequents  ...  conviction  zhangs_metric
63         (soup)  (mineral water)  ...    1.401441       0.503458
54  (ground meat)      (spaghetti)  ...    1.373959       0.624888
```

```
60    (olive oil)  (mineral water)  ...    1.308483       0.460018
52  (ground meat)  (mineral water)  ...    1.305576       0.474647
68    (olive oil)      (spaghetti)  ...    1.268387       0.536127

[5 rows x 10 columns]
Top 5 Antecedent -> Consequent Rules based on Leverage:
        antecedents      consequents  ...  conviction  zhangs_metric
53      (spaghetti)    (ground meat)  ...    1.163699       0.682292
54    (ground meat)      (spaghetti)  ...    1.373959       0.624888
65      (spaghetti)  (mineral water)  ...    1.159468       0.369806
64  (mineral water)      (spaghetti)  ...    1.102184       0.400941
51  (mineral water)    (ground meat)  ...    1.088782       0.561883

[5 rows x 10 columns]
Top 5 Antecedent -> Consequent Rules based on Lift:
            antecedents      consequents  ...  conviction  zhangs_metric
54        (ground meat)      (spaghetti)  ...    1.373959       0.624888
53          (spaghetti)    (ground meat)  ...    1.163699       0.682292
68          (olive oil)      (spaghetti)  ...    1.268387       0.536127
63               (soup)  (mineral water)  ...    1.401441       0.503458
42  (frozen vegetables)           (milk)  ...    1.156758       0.526685

[5 rows x 10 columns]
Top 2 Antecedent -> Consequent Rules based on Zhang's Metric:
      antecedents    consequents  ...  conviction  zhangs_metric
53    (spaghetti)  (ground meat)  ...    1.163699       0.682292
54  (ground meat)    (spaghetti)  ...    1.373959       0.624888

[2 rows x 10 columns]
Bottom 2 Antecedent -> Consequent Rules based on Zhang's Metric:
       antecedents      consequents  ...  conviction  zhangs_metric
36  (french fries)  (mineral water)  ...    0.949021      -0.200059
38     (spaghetti)   (french fries)  ...    0.985224      -0.086750

[2 rows x 10 columns]
Lift Value for the rule (ground meat -> spaghetti):
2.2908567284695827

Process finished with exit code 0
```