a5

Graded

22 Hours, 27 Minutes Late

Group

Feven Tefera
Mihret Kemal
Tagore Kosireddy
...and 1 more
View or edit group

Total Points

53 / 53 pts

Autograder Score

53.0 / 53.0

Passed Tests

Public Tests
q0 (2/2)
q1a (19/19)
q1e (16/16)
q1g (16/16)

Autograder Results

Autograder Output

```
 _______      __       __
/ _____ \   __|  |__   __|  |__
| /    \ |  |__   __|  |__   __|  ______   _ _____
| |     | |    |  |       |  |    |  ___  |  | |/ ___|
| |     | |    |  |       |  |    | |__| |  |  /
| |     | |    |  |       |  |    | _____|  |  |
| \____/ |    |  |_      |  |_   | |____   |  |
\_______/     \__|      \__|   \_____|  |__|
                                  v5.2.3
```

---------- GRADING SUMMARY ----------

No discrepancies found while verifying scores against the log.

Successfully uploaded submissions for: fatefera@mtu.edu, mkemal@mtu.edu, trkosire@mtu.edu, mkngala@

Total Score: 53.000 / 53.000 (100.000%)

	name	score	max_score
0	Public Tests	NaN	NaN
1	q0	2.0	2.0
2	q1a	19.0	19.0
3	q1e	16.0	16.0
4	q1g	16.0	16.0

Public Tests

q0 results: All test cases passed!

q1a results: All test cases passed!

q1e results: All test cases passed!

q1g results: All test cases passed!

q0 (2/2)

q0 results: All test cases passed!

q1a (19/19)

q1a results: All test cases passed!

q1e (16/16)

q1e results: All test cases passed!

q1g (16/16)

q1g results: All test cases passed!

Submitted Files

A5 - Python

This assignment will cover topics of association analysis.

Make sure that you keep this notebook named as "a5.ipynb"

Any other packages or tools, outside those listed in the assignments or Canvas, should be cleared by Dr. Brown before use in your submission.

Q0 - Setup

The following code looks to see whether your notebook is run on Gradescope (GS), Colab (COLAB), or the linux Python environment you were asked to setup.

In [1]:
```
import re
import os
import platform
import sys

# flag if notebook is running on Gradescope
if re.search(r'amzn', platform.uname().release):
    GS = True
else:
    GS = False

# flag if notebook is running on Colaboratory
try:
  import google.colab
  COLAB = True
except:
  COLAB = False

# flag if running on Linux lab machines.
cname = platform.uname().node
if re.search(r'(guardian|colossus|c28|lebrown|rovernet)', cname):
    LLM = True
else:
    LLM = False

print("System: GS - %s, COLAB - %s, LLM - %s" % (GS, COLAB, LLM))
```

```
System: GS - False, COLAB - False, LLM - True
```

Notebook Setup

It is good practice to list all imports needed at the top of the notebook. You can import modules in later cells as needed, but listing them at the top clearly shows all which are needed to be available / installed.

If you are doing development on Colab, the otter-grader package is not available, so you will need to install it with pip (uncomment the cell directly below).

In [ ]:
```
# Only uncomment if you developing on Colab
# if COLAB == True:
#     print("Installing otter:")
#     !pip install otter-grader==4.2.0
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline


from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
from mlxtend.frequent_patterns import fpgrowth

import warnings
warnings.filterwarnings('ignore')

# Package for Autograder
import otter
grader = otter.Notebook()
```

In [3]:
```
grader.check("q0")
```

Out [3]:
```
q0 results: All test cases passed!
```

Q1 - Association Analysis

For this problem, you will analyze a portion of the Instacart Online Grocery Shopping Dataset from 2017. The full data set is available if you are interested. https://www.instacart.com/datasets/grocery-shopping-2017

The original dataset has 3 million orders. We will work with a smaller data set.

You will use the following files for this analysis:

- `orders_products.csv`
- `products.csv`

Structure of the files, `products_orders`:

- `order_id`: foreign key
- `product_id`: foreign key
- `add_to_cart_order`: order in which each product was added to cart
- `reordered`: 1 if this product has been ordered by this user in the past, 0 otherwise

File structure for `products`:

- `product_id`: product identifier
- `product_name`: name of the product
- `aisle_id`: foreign key
- `department_id`: foreign key

You can connect the `product_id` with the name of the product, `product_name` in the `products.csv` file.

Q1(a) - Load the Data

Load the 2 data files mentioned above.

You will need to transform this data into a boolean transaction DataFrame `orders`.

These boolean transaction DataFrame will have rows corresponding to orders / transaction with the `order_id` as the row index. The DataFrame will have columns corresponding to products with the `product_name` as the column names. The DataFrame is boolean and entry i, j is : "False" meaning this product, j, was not purchased in order, i, and "True" means product, j, was purchased in order, i.

The rows in the DataFrame should be in order of `order_id`. The columns should be ordered in alphanumeric ordering of the `product_name`.

Note, you can not use the `mlxtend.TransactionEncoder` function for this because it expects data as lists of lists.

Hint: several `pandas` functions such as `join`, `merge`, or `pivot` may be useful to construct the `orders` DataFrame.

Also, calculate the mean/max number of products per order for the `orders` dataset: `mean_num_prods` and `max_num_prods`

- `orders_num_rows`
- `orders_num_cols`
- `orders_col_names`

Then, save off a slice of the data frame, `orders_small`, the first 50 rows the first 100 items.

In [96]:

```
# Load and prepare the data

prods_orders = pd.read_csv("products_orders.csv")
prods = pd.read_csv("products.csv")

# Merge the two DataFrames on the column 'product_id' and create a pivot table
orders = pd.pivot_table(pd.merge(prods_orders, prods, on='product_id'),
                        index='order_id', columns='product_name', values='product_id',
                        aggfunc=lambda x: True, fill_value=False)

# Get the number of rows and columns
orders_num_rows = orders.shape[0]
orders_num_cols = orders.shape[1]
# Get the names of the columns (product names)
orders_col_names = orders.columns.values

# Calculate the mean and maximum number of products per order
mean_num_prods = orders.sum(axis=1).mean()
max_num_prods = orders.sum(axis=1).max()


# Create a smaller subset of the 'orders' DataFrame
orders_small = orders.iloc[:50, :100]

orders.head()

print("Mean number of products per order: ", mean_num_prods)
print("Max number of projucts per order:  ", max_num_prods)
```

```
# clean up unneeded raw data
del prods_orders, prods
```

Mean number of products per order: 10.67698259187621
Max number of projucts per order: 54

In [97]:
```
grader.check("q1a")
```

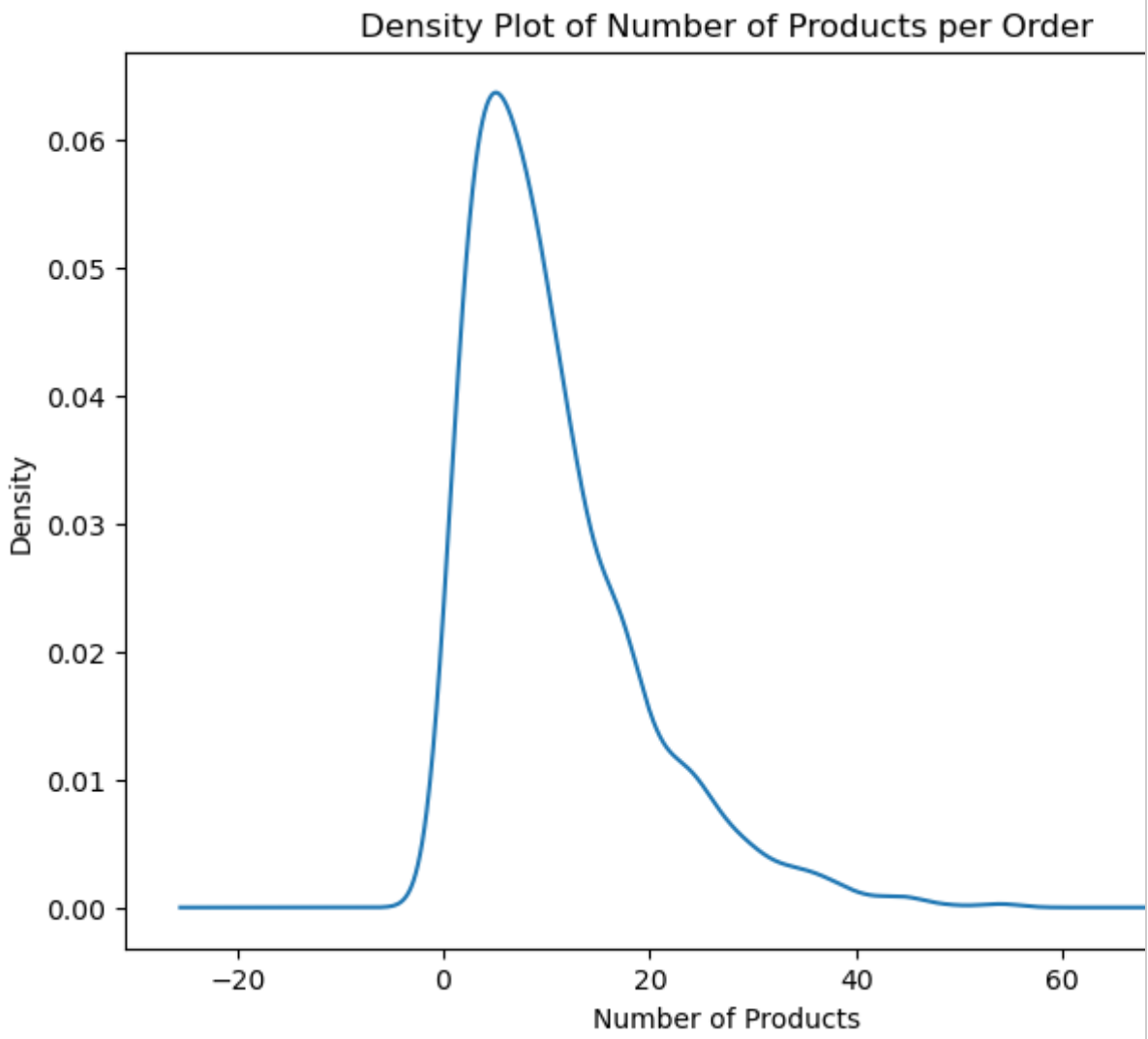
Out [97]:
q1a results: All test cases passed!

Q1(b) - Explore the Data

Create a density plot showing the number of products per order using the `orders` data set.

In [98]:
```
# Calculate the total number of products per order
num_products_per_order = orders.sum(axis=1)

# Plot the number of products per order
plt.figure(figsize=(8, 6))
num_products_per_order.plot(kind='density')
plt.title('Density Plot of Number of Products per Order')
plt.xlabel('Number of Products')
plt.ylabel('Density')
plt.show()
```



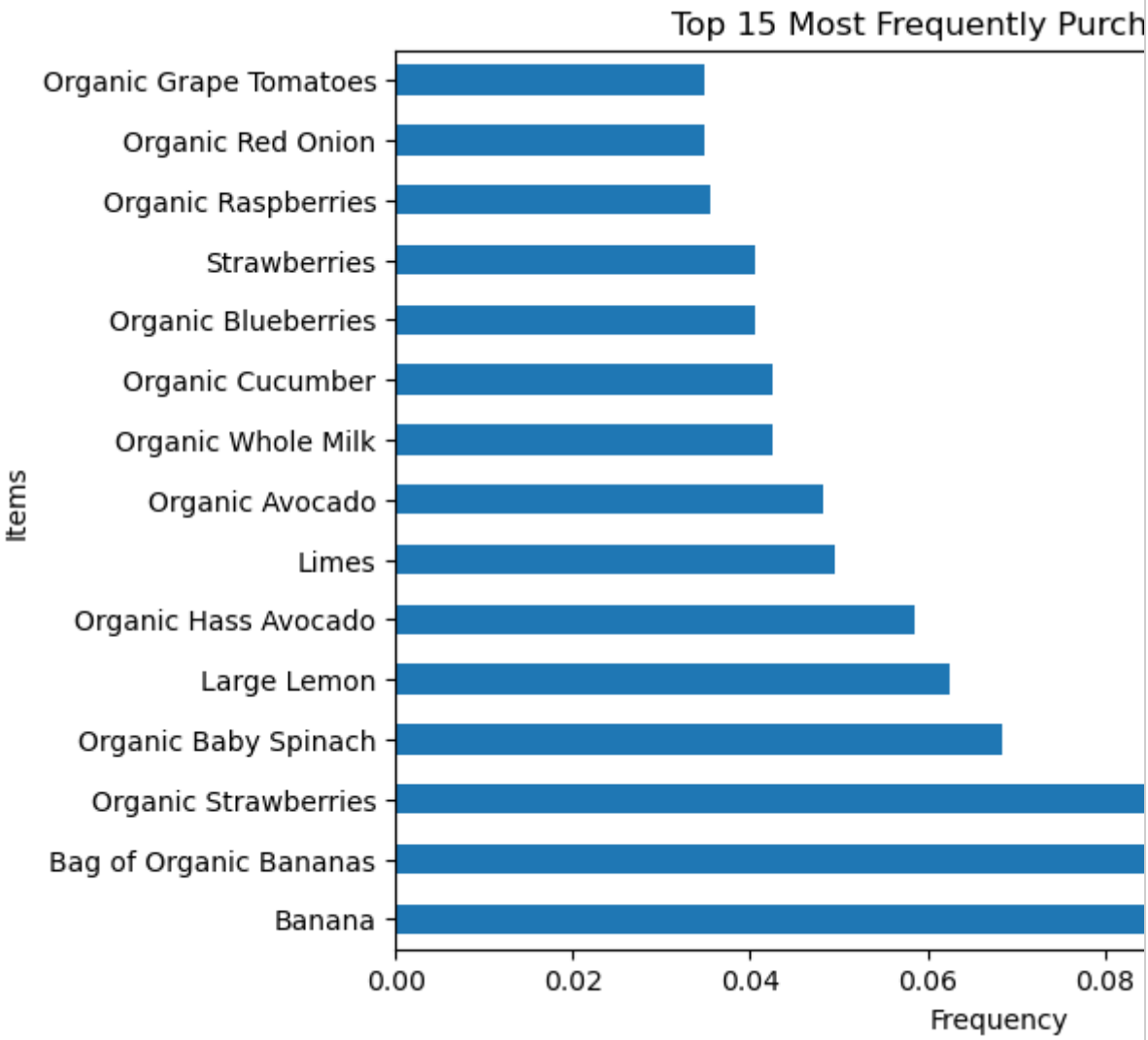
Q1(c) - Explore the Data, part 2

For the `orders` dataset, create an top 15 item frequency plot, that is plot the top 15 most frequently purchased items. This should be a bar plot with items vs. frequency (relative support).

In [99]:

```
# Calculate the mean frequency of each item and select the top 15 most frequently purchased items
top_15_items = orders.mean().sort_values(ascending = False).head(15)

# Plot top 15 most frequently purchased product (by relative support)
plt.figure(figsize=(8, 6))
top_15_items.plot(kind = 'barh')
plt.title('Top 15 Most Frequently Purchased Items')
plt.xlabel('Frequency')
plt.ylabel('Items')
plt.show()
```



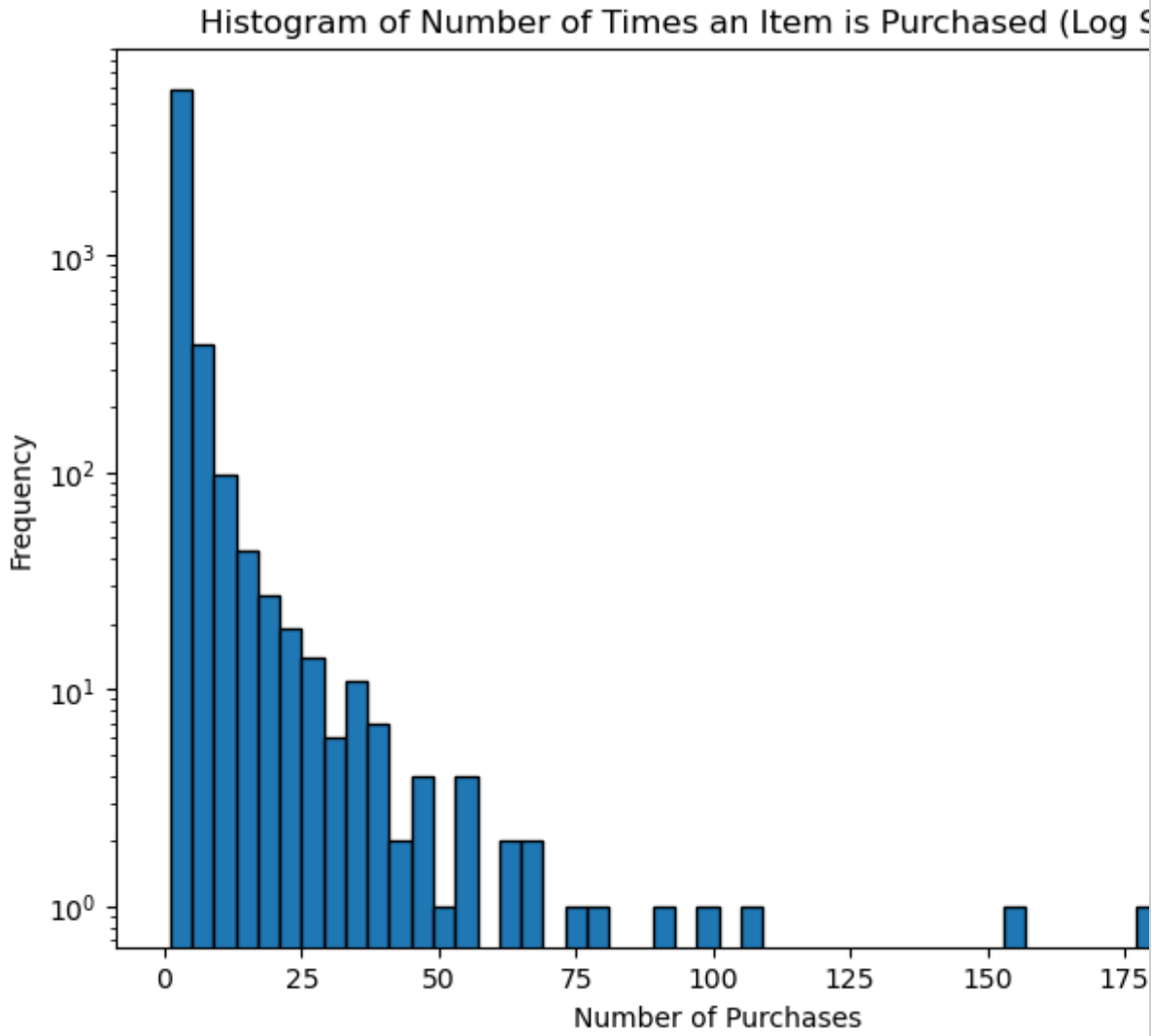
Q1(d) - Explore the Data, part 3

For the `orders` dataset, create an histogram of the number of times an item is purchased. You may want to consider using log scaling to view the data distribution more easily.

In [100]:

```
# Calculate the total number of times each item is purchased
num_item_purchased = orders.sum(axis=0)

# Plot histogram of number of purchases per item.
plt.figure(figsize=(8, 6))
plt.hist(num_item_purchased, bins = 50, edgecolor = 'black', log = True)
plt.title('Histogram of Number of Times an Item is Purchased (Log Scale)')
plt.xlabel('Number of Purchases')
plt.ylabel('Frequency')
plt.show()
```



Q1(e) - Apriori

For the `orders` dataset, use Apriori to find association rules, `rules` with a minimum relative support of 0.0035 and confidence of 0.5.

In `q1e_df` sort the rules by leverage (descending order), then by confidence (in descending order) and return the top 20 rules.

Note, the minimum support level is rather high given the information plotted in Q1(c) and Q1(d). However, this was done to avoid using too much memory (lower support values will require 15-20 GB memory).

In [101]:

```
# Run Apriori as instructed

# Generate association rules using Apriori
rules = association_rules(apriori(orders, min_support = 0.0035, use_colnames =
```

```
True),
                    metric = 'confidence', min_threshold = 0.5)

# Sort the rules based on leverage and confidence in descending order and select
the top 20
q1e_df = rules.sort_values(by = ['leverage', 'confidence'], ascending =[False,
False]).head(20)


q1e_df.iloc[0:10, [0, 1, 4, 5, 7]]
```

Out [101]:

	antecedents	consequents	support	confidence	leverage
5	(Honeycrisp Apple)	(Banana)	0.011605	0.500000	0.008597
18	(Organic Hass Avocado, Organic Strawberries)	(Bag of Organic Bananas)	0.007737	0.521739	0.006035
26	(Large Lemon, Organic Avocado)	(Limes)	0.005803	0.529412	0.005259
25	(Limes, Organic Avocado)	(Large Lemon)	0.005803	0.562500	0.005158
6	(Roma Tomato)	(Banana)	0.006447	0.526316	0.004860
32	(Organic Red Onion, Organic Avocado)	(Limes)	0.005158	0.800000	0.004838
33	(Limes, Organic Avocado)	(Organic Red Onion)	0.005158	0.500000	0.004799
31	(Organic Red Onion, Limes)	(Organic Avocado)	0.005158	0.533333	0.004690
20	(Organic Large Extra Fancy Fuji Apple, Bag of ...	(Organic Strawberries)	0.005803	0.500000	0.004658
19	(Organic Large Extra Fancy Fuji Apple, Organic...	(Bag of Organic Bananas)	0.005803	0.529412	0.004545

In [102]:

```
grader.check("q1e")
```

Out [102]:

```
q1e results: All test cases passed!
```

Q1(f) - Apriori, part 2

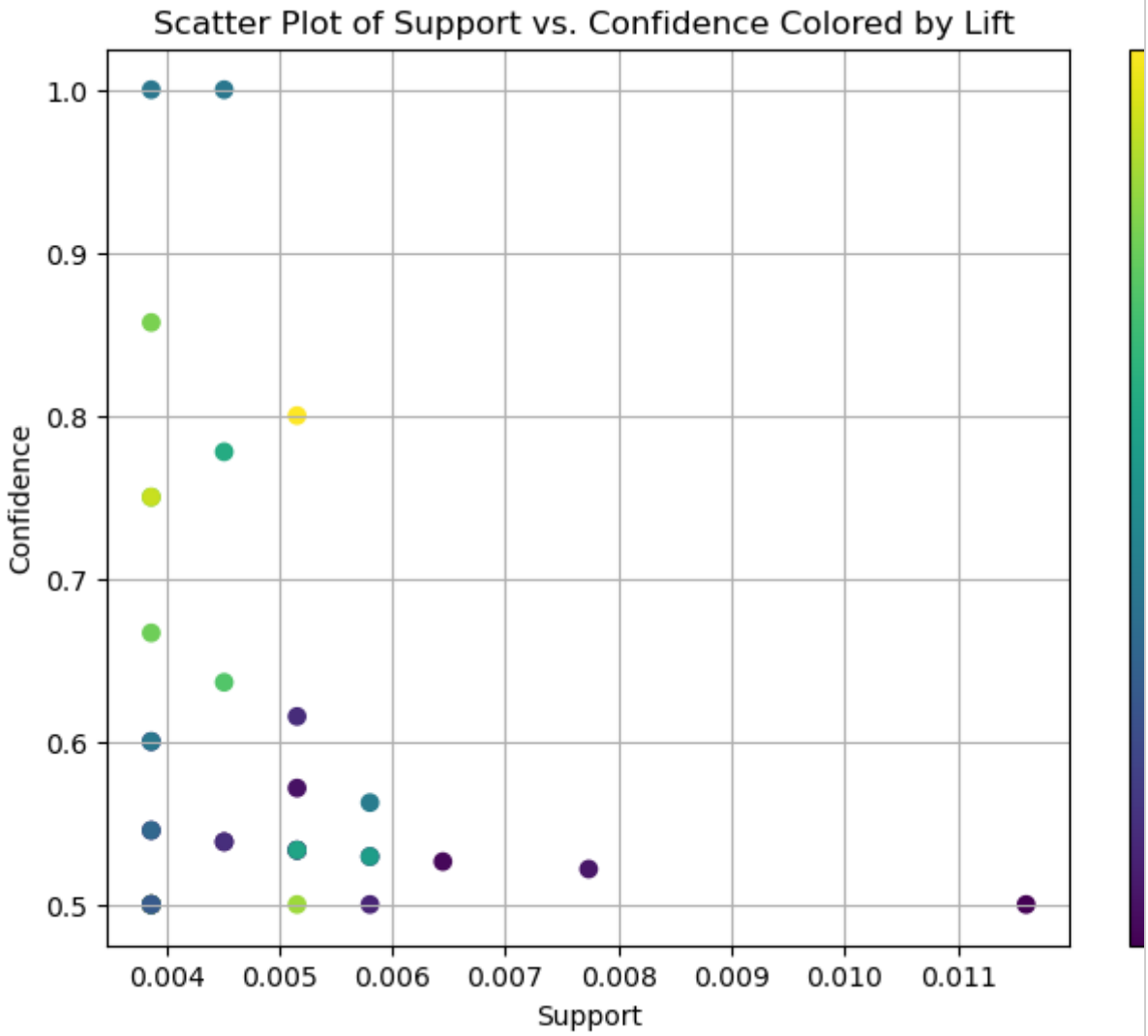
Create a scatterplot of the rules, plotting support vs. confidence colored by lift value.

```

```

In [103]:

```
# Plot the results of Apriori
plt.figure(figsize=(8, 6))
plt.scatter(rules['support'], rules['confidence'], c = rules['lift'], cmap = 'viridis')
plt.colorbar(label = 'Lift')
plt.title('Scatter Plot of Support vs. Confidence Colored by Lift ')
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.grid(True)
plt.show()
```



Q1(g) - FPGrowth

For the `orders` dataset, use FPGrowth to find association rules, `rules2` with a minimum support of 0.0035 and confidence of 0.5.

Sort the rules by conviction (descending order), then by support (descending order). Store the top 20 rules in `q1g_df`.

Note, the relative speed for FPGrowth over Apriori.

In [104]:
```
# Run FPGrowth as instructed

# Generate association rules using FP-Growth
rules2 = association_rules(fpgrowth(orders, min_support = 0.0035, use_colnames = True),
                           metric = 'confidence', min_threshold = 0.5)

# Sort the rules based on conviction and support in descending order and select the top 20
q1g_df = rules2.sort_values(by =['conviction', 'support'], ascending = [False, False]).head(20)


q1g_df.iloc[0:10, [0, 1, 4, 5, 8]]
```

Out [104]:

	antecedents	consequents	support	confidence	conviction
22	(Organic Large Extra Fancy Fuji Apple, Organic...	(Bag of Organic Bananas)	0.004513	1.000000	inf
25	(Organic Hass Avocado, Organic Plain Greek Who...	(Bag of Organic Bananas)	0.003868	1.000000	inf
4	(Limes, Asparagus)	(Large Lemon)	0.003868	0.857143	6.562218
33	(Organic Red Onion, Organic Avocado)	(Limes)	0.005158	0.800000	4.751773
3	(Limes, Organic Garnet Sweet Potato (Yam))	(Organic Baby Spinach)	0.004513	0.777778	4.192456
16	(Organic Cilantro, Banana)	(Limes)	0.003868	0.750000	3.801418
26	(Bag of Organic Bananas, Organic Plain Greek W...	(Organic Hass Avocado)	0.003868	0.750000	3.765313
5	(Asparagus, Large Lemon)	(Limes)	0.003868	0.666667	2.851064
31	(Organic Red Onion, Large Lemon)	(Limes)	0.004513	0.636364	2.613475
2	(Organic Hass Avocado, Organic Garnet Sweet Po...	(Organic Baby Spinach)	0.003868	0.600000	2.329142

In [105]:
```
grader.check("q1g")
```

Out [105]:
```
q1g results: All test cases passed!
```

Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

NOTE the submission must be run on the campus linux machines. See the instruction in the Canvas assignment.

In [106]:
```
# Save your notebook first, then run this cell to export your submission.
grader.export()
```

<IPython.core.display.HTML object>

a5.pdf Download

Your browser does not support PDF previews. You can download the file instead.

.OTTER_LOG Download

1	Binary file hidden. You can download it using the button above.

__zip_filename__ Download

1	a5_2024_04_18T22_25_50_450284.zip