Task 1 : Transactions Data Set :

Objective: To find out if there is any sesaonal pattern in purchase behaviour and to find seasonal Score

Table 1: Month-Wise Count for all Categories:

Count of product_id	Category				
Month- Year	**Casual Dress**	**Fleece Jacket**	**Pullover Sweater**	**Sleeveless Blouse**	**Grand Total**
Apr-18	60434	261	12886	14516	88097
Aug-18	9730	151	3072	2151	15104
Dec-17	52338	449	24862	10481	88130
Feb-18	91162	474	24109	22752	138497
Jan-18	66649	498	22481	14674	104302
Jul-18	62865	581	15219	14183	92848
Jun-18	63372	336	10356	15559	89623
Mar-18	82695	407	20045	18177	121324
May-18	80490	374	13732	19962	114558
Nov-17	54334	645	29354	10918	95251
Oct-17	56963	798	31239	11841	100841
Grand Total	**681032**	**4974**	**207355**	**155214**	**1048575**

Table 2 : Average of all Categories:

Month- Year	Casual Dress	Fleece Jacket	Pullover Sweater	Sleeveless Blouse
Apr-18	60434	261	12886	14516
Aug-18	9730	151	3072	2151
Dec-17	52338	449	24862	10481
Feb-18	91162	474	24109	22752
Jan-18	66649	498	22481	14674
Jul-18	62865	581	15219	14183
Jun-18	63372	336	10356	15559
Mar-18	82695	407	20045	18177
May-18	80490	374	13732	19962
Nov-17	54334	645	29354	10918
Oct-17	56963	798	31239	11841
Average	**61912.00**	**452.18**	**18850.45**	**14110.36**

Sample Calculation :

Category : Casual Dress

Formula to find Sesonal Index Score

Oct-17 = **56963/61912 = 0.920063962**

Table 3: Seasonal Index Score [0 - 1] Range for Categories:

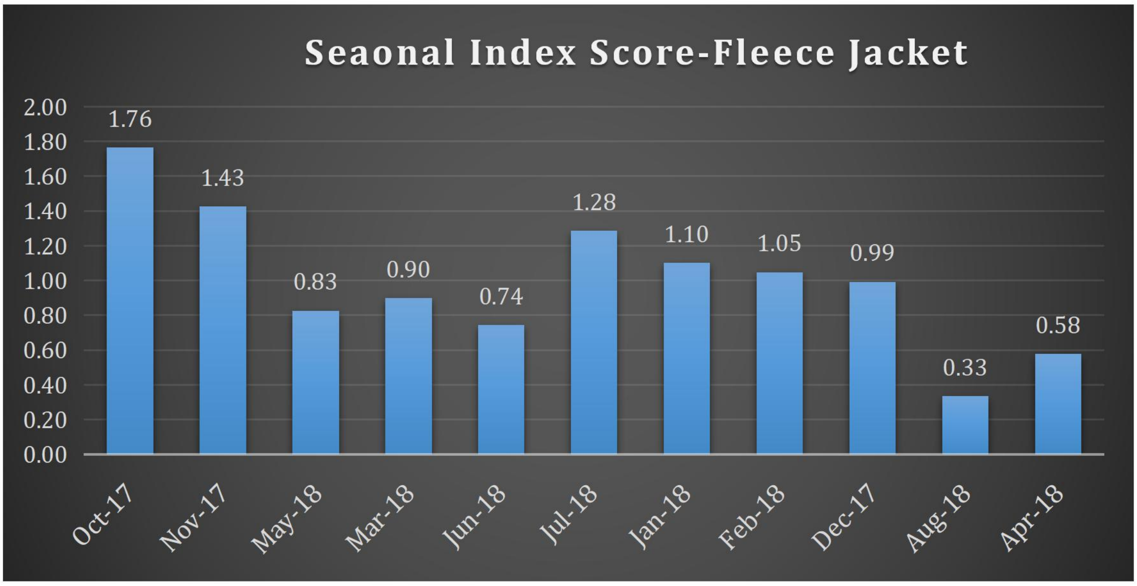
Month- Year	Casual Dress	Fleece Jacket	Pullover Sweater	Sleeveless Blouse
Oct-17	0.920063962	1.76477684	1.657201418	0.839170436
Nov-17	0.877600465	1.42641737	1.557203829	0.773757522
Dec-17	0.845361158	0.99296341	1.318907188	0.742787377
Jan-18	1.076511823	1.1013269	1.192597237	1.03994485
Feb-18	1.47244476	1.048250905	1.278961202	1.612431868
Mar-18	1.335686135	0.900080418	1.063369584	1.288202095
Apr-18	0.976127407	0.577201448	0.683590943	1.028747407
May-18	1.300071069	0.827100925	0.728470497	1.414704859
Jul-18	1.015392816	1.284881383	0.807354537	1.005147732
Jun-18	1.023581858	0.743063932	0.549376673	1.102664708
Aug-18	0.157158548	0.33393647	0.162966893	0.152441146

Table 4 : Rounded off Seasonal Index Score [0 - 1] Range for Categories:

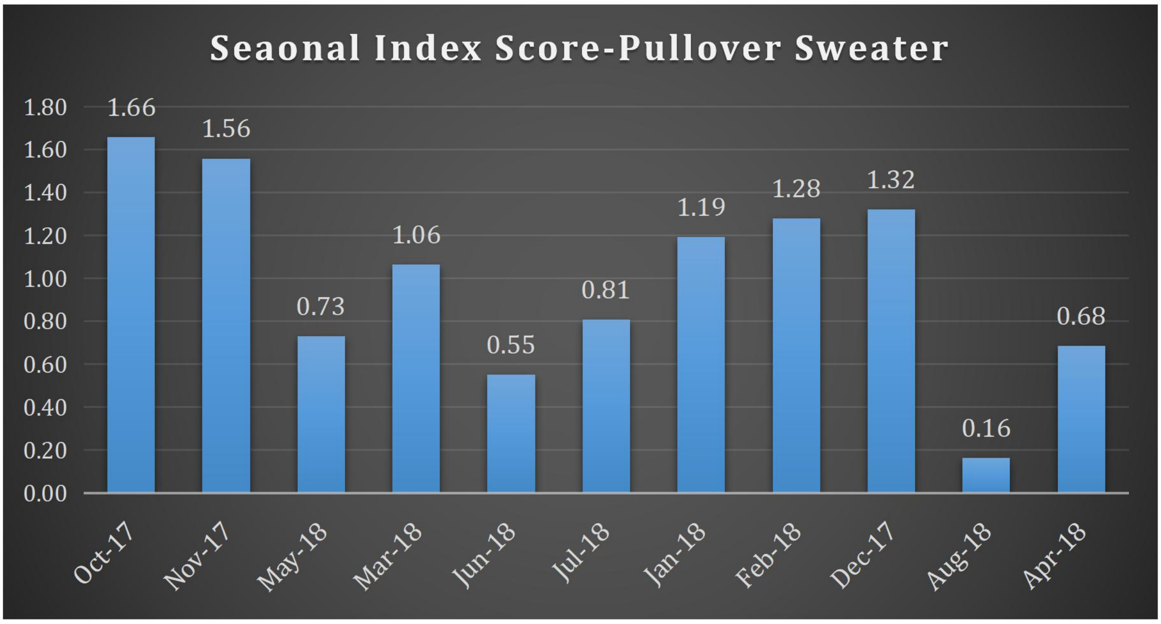
Month- Year	Casual Dress	Fleece Jacket	Pullover Sweater	Sleeveless Blouse
Oct-17	0.92	1.76	1.66	0.84
Nov-17	0.88	1.43	1.56	0.77
Dec-17	0.85	0.99	1.32	0.74
Jan-18	1.08	1.10	1.19	1.04
Feb-18	1.47	1.05	1.28	1.61
Mar-18	1.34	0.90	1.06	1.29
Apr-18	0.98	0.58	0.68	1.03
May-18	1.30	0.83	0.73	1.41
Jul-18	1.02	1.28	0.81	1.01
Jun-18	1.02	0.74	0.55	1.10
Aug-18	0.16	0.33	0.16	0.15

Charts:

Seaonal Index Score-Casual Dress
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.92
0.88
1.30
1.34
1.02
1.02
1.08
1.47
0.85
0.16
0.98
Oct-17
Nov-17
May-18
Mar-18
Jun-18
Jul-18
Jan-18
Feb-18
Dec-17
Aug-18
Apr-18



Seaonal Index Score-Fleece Jacket
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
1.76
1.43
0.83
0.90
0.74
1.28
1.10
1.05
0.99
0.33
0.58
Oct-17
Nov-17
May-18
Mar-18
Jun-18
Jul-18
Jan-18
Feb-18
Dec-17
Aug-18
Apr-18


Seaonal Index Score-Pullover Sweater
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
1.66
1.56
0.73
1.06
0.55
0.81
1.19
1.28
1.32
0.16
0.68
Oct-17
Nov-17
May-18
Mar-18
Jun-18
Jul-18
Jan-18
Feb-18
Dec-17
Aug-18
Apr-18

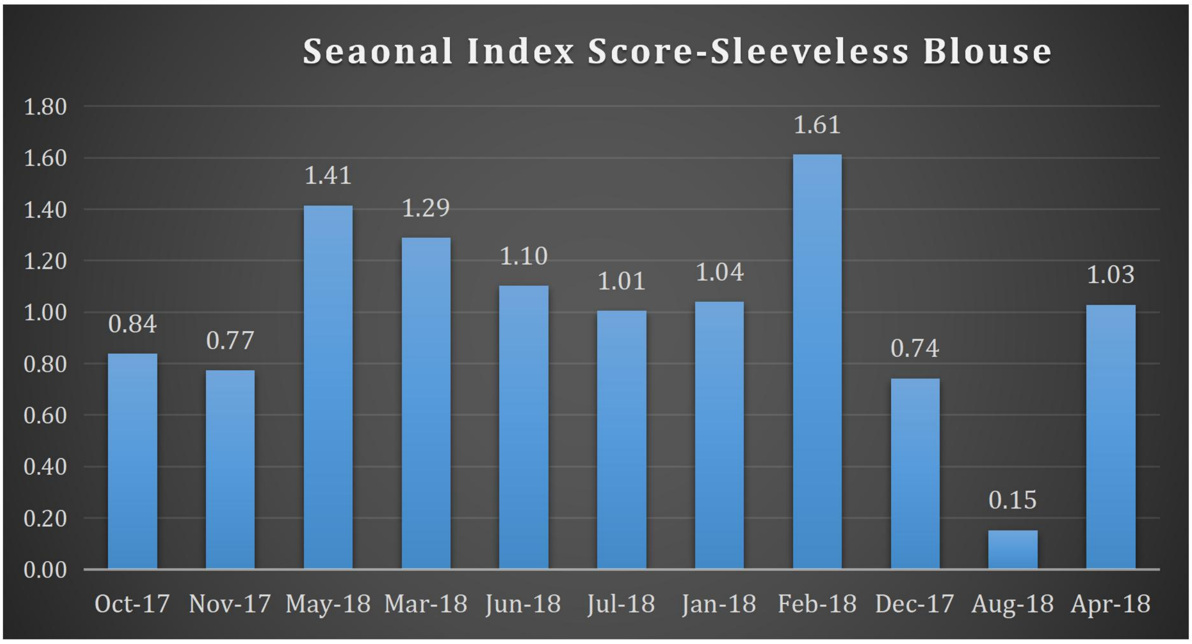
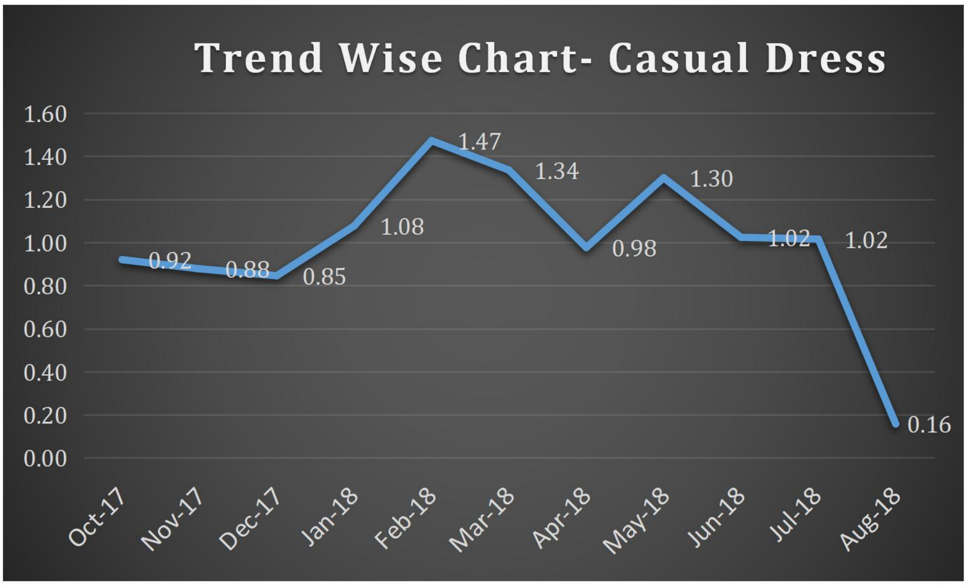
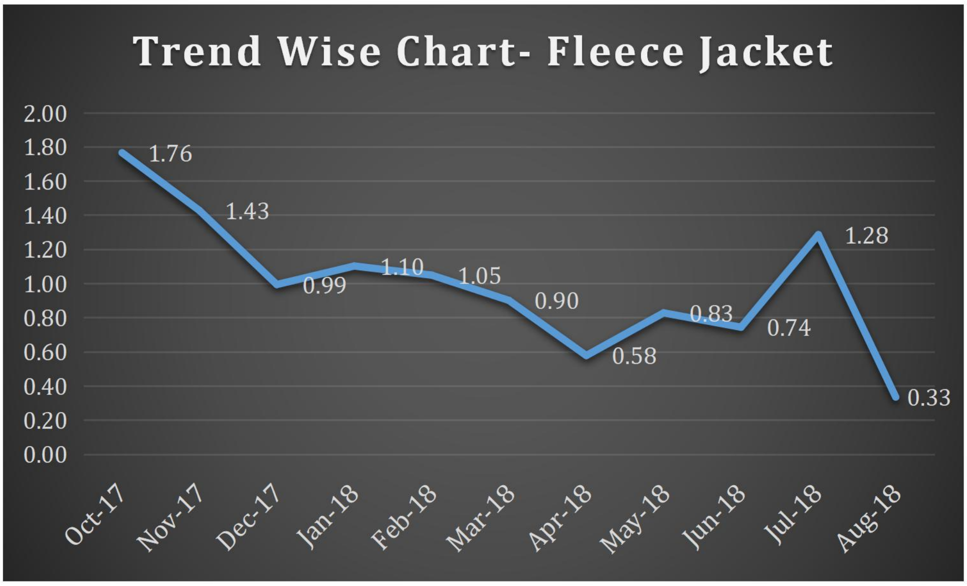
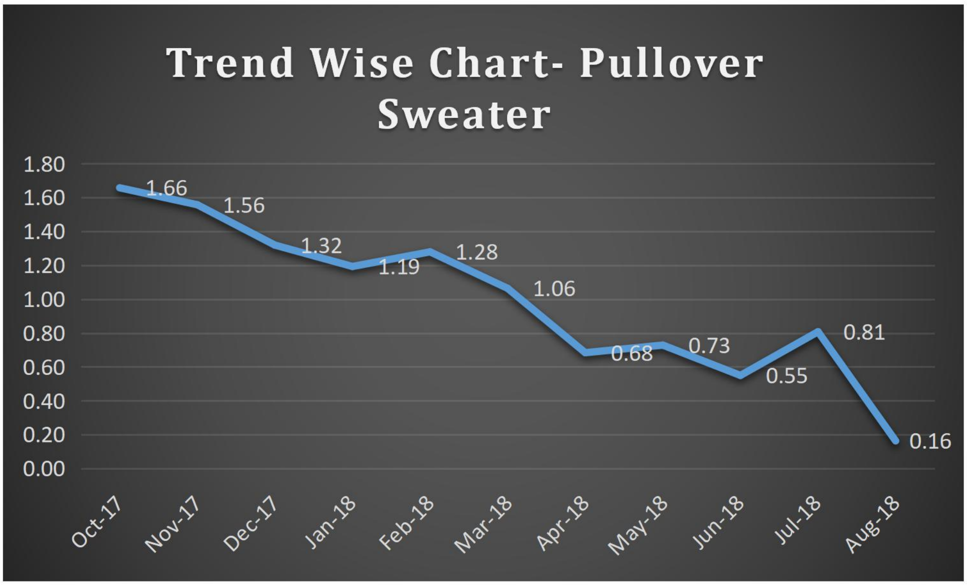
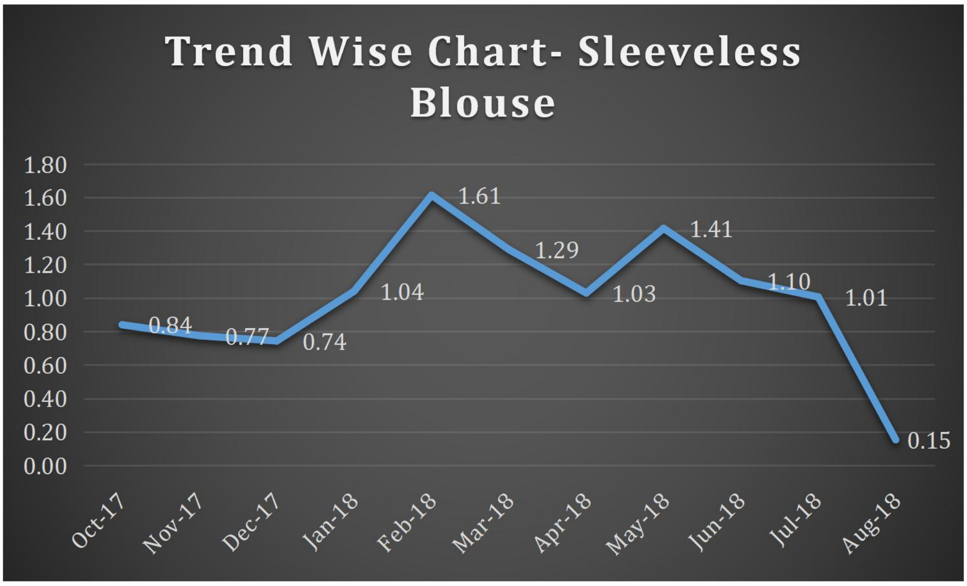

Seaonal Index Score-Sleeveless Blouse
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.84
0.77
1.41
1.29
1.10
1.01
1.04
1.61
0.74
0.15
1.03
Oct-17
Nov-17
May-18
Mar-18
Jun-18
Jul-18
Jan-18
Feb-18
Dec-17
Aug-18
Apr-18

Chart 2: Time -Series Chart:



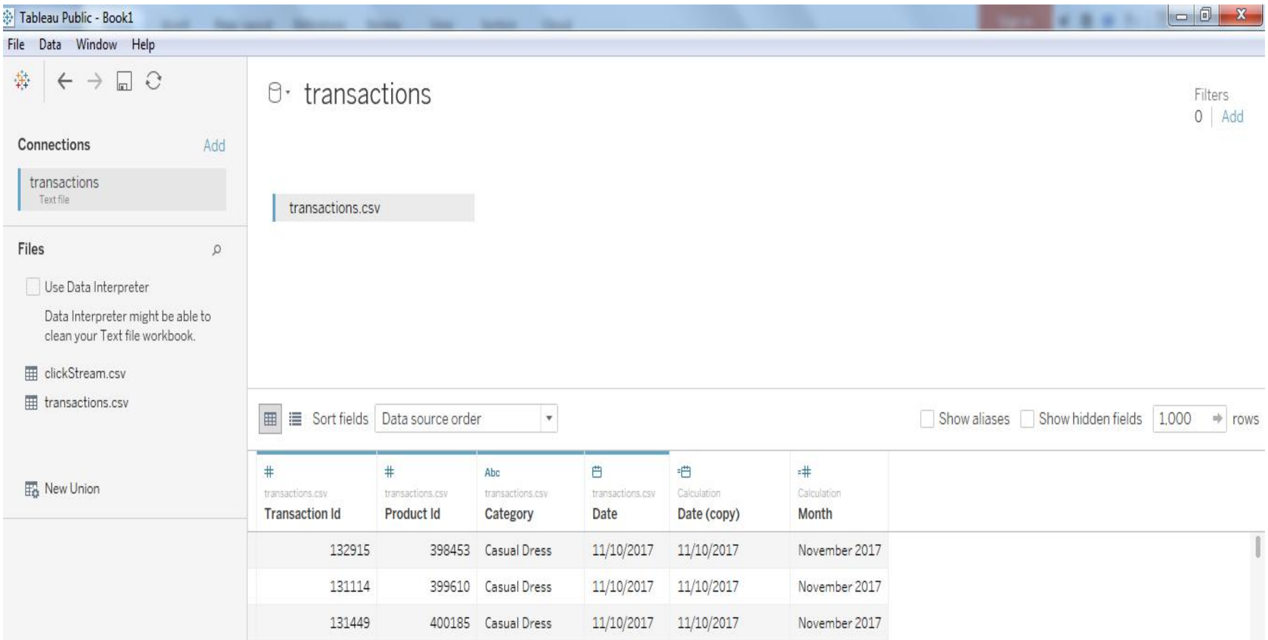



Trend Wise Chart- Pullover Sweater
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
1.66
1.56
1.32
1.19
1.28
1.06
0.68
0.73
0.55
0.81
0.16
Oct-17
Nov-17
Dec-17
Jan-18
Feb-18
Mar-18
Apr-18
May-18
Jun-18
Jul-18
Aug-18


Trend Wise Chart- Sleeveless Blouse
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.84
0.77
0.74
1.04
1.61
1.29
1.03
1.41
1.10
1.01
0.15
Oct-17
Nov-17
Dec-17
Jan-18
Feb-18
Mar-18
Apr-18
May-18
Jun-18
Jul-18
Aug-18

Task 1 : Using Tableau Public:

Step 1 : Loading of Data:



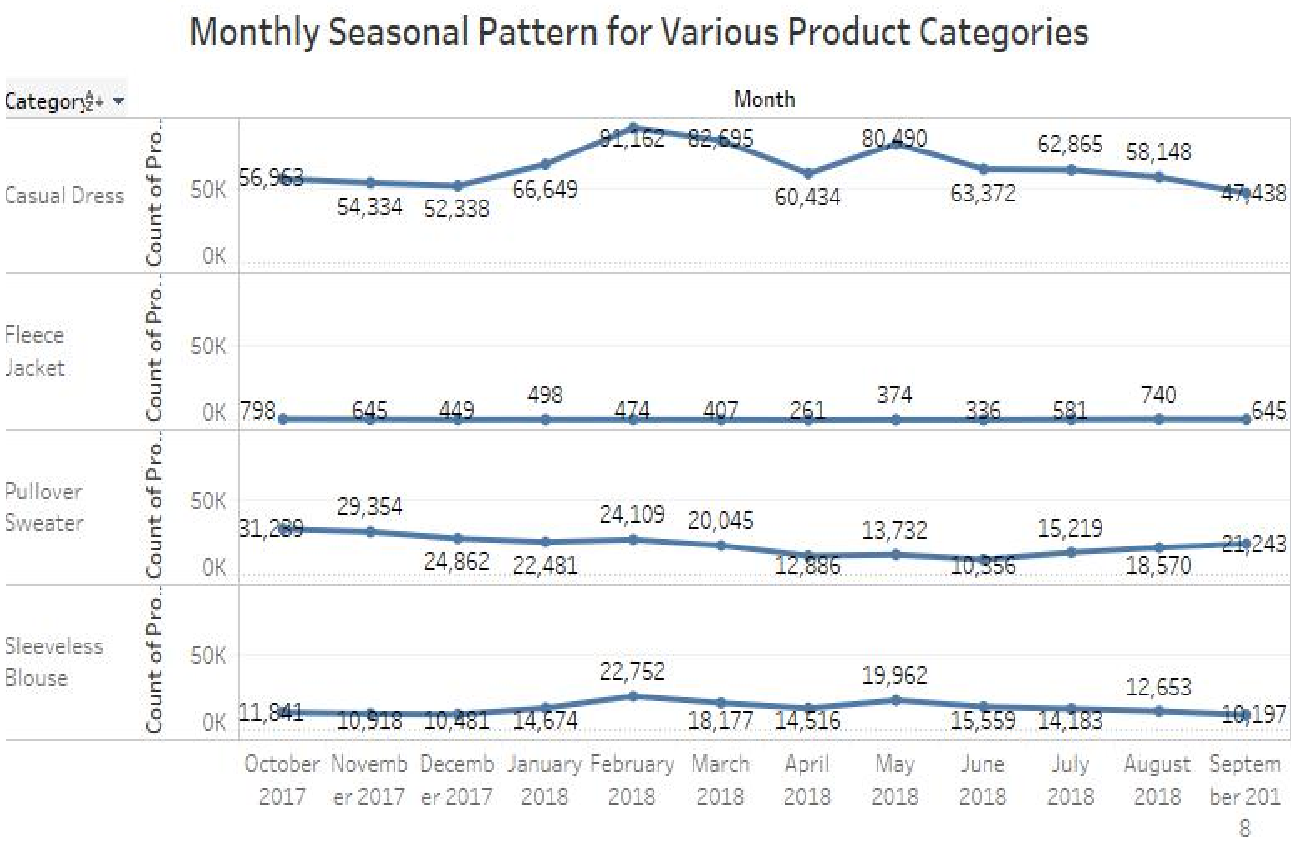
Step 2: Analyzing and Exploring Dataset:

Monthly Seasonal Pattern for Various Product Categories

Month

Category	October 2017	November 2017	December 2017	January 2018	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018	August 2018	September 2018
Casual Dress	56,963	54,334	52,338	66,649	91,162	82,695	60,434	80,490	63,372	62,865	58,148	47,438
Fleece Jack..	798	645	449	498	474	407	261	374	336	581	740	645
Pullover Sw..	31,239	29,354	24,862	22,481	24,109	20,045	12,886	13,732	10,356	15,219	18,570	21,243
Sleeveless ..	11,841	10,918	10,481	14,674	22,752	18,177	14,516	19,962	15,559	14,183	12,653	10,197

Step 3 : Seasonal Pattern Graph - Month-Wise for Various Categories:



Task 2 : To Predict the next Category that will be bought based on categories bought in the past:

Machine Learning Workflow

1. Reading of Click stream csv Data
2. Transformation of Data from Unstructured to Structured
3. Exploratory Data Analysis on the Data Set [To analyse data with visual methods to summarize the main characteristics.]
4. Building of Machine Learning Model using Apriori Algorithm
5. Performing Evaluvation Metrics for Machine Learning Model

Final Phase :

To Build a Rest-Api and check for New Predictions:

TASK 2 : E-Commerce Purchase Recommendation [Clickstream dataset]

Exploratory Data Analysis: Insights found out from Clickstream Dataset:

Loading of the Required Packages for doing the analysis:

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

Reading of Data:

```
df = pd.read_csv("C:/Users/Karthick/Desktop/clickStreams.csv")
```

```
In [42]: df
Out[42]:
        clicked_epoch      Time Stamp    uuid       date      price  \
0        1.496273e+09   6/1/2017 0:25  110971   6/1/2017     599.50
1        1.496273e+09   6/1/2017 1:23  110971   6/1/2017     599.50
2        1.496276e+09   6/1/2017 1:35   49864   6/1/2017    1349.10
3        1.496277e+09   6/1/2017 1:35   49864   6/1/2017    1124.10
4        1.496280e+09   6/1/2017 1:48   21453   6/1/2017     999.00
5        1.496281e+09   6/1/2017 1:53  120631   6/1/2017     999.00
6        1.496281e+09   6/1/2017 1:55  120631   6/1/2017     999.00
```

```
        product_id            category
0           122712      kurta & kurtis
1             3453      kurta & kurtis
2            13610               jeans
3            48309               jeans
4           133239      kurta & kurtis
5            78375      kurta & kurtis
6            62607      kurta & kurtis
```

Knowing the data types of all the variables present in the dataset:

```
In [44]: df.dtypes
Out[44]:
clicked_epoch     float64
Time Stamp         object
uuid                int64
date               object
price             float64
product_id          int64
category           object
dtype: object
```

Note: I have added a new column called Time Stamp and I have converted Unix time [Clicked_epoch] to the time format which excel uses.

Formula used:

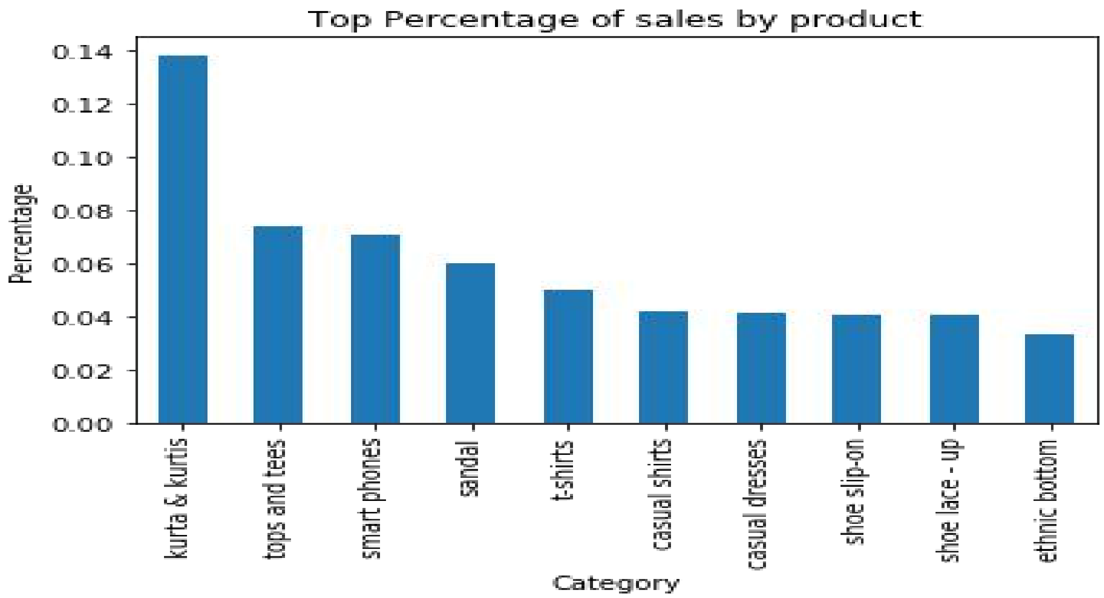
Converting Unix Time Stamp to Excel Date:

= (A2/86400)+DATE(1970,1,1)

The formula was used in excel Sheet and then the data is read in Python.

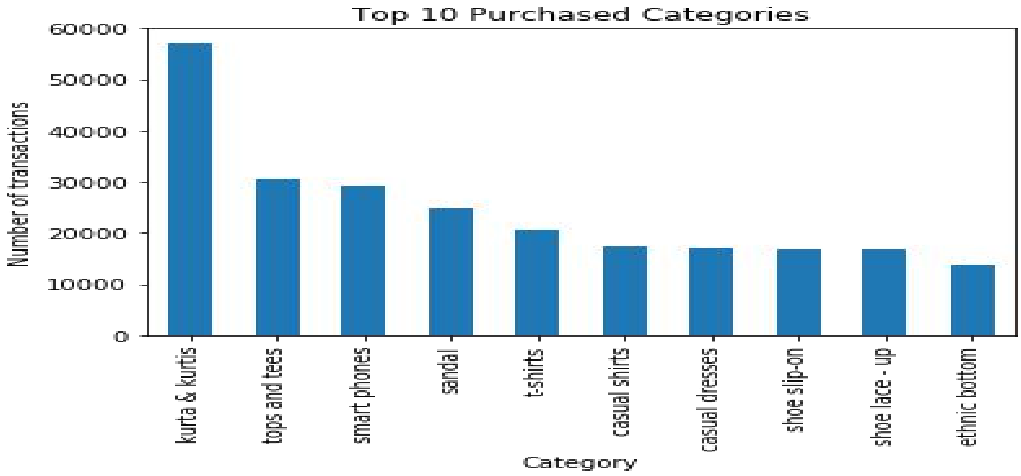
1) Visualizing Top Percentage of sales by product:

```
df.category.value_counts(normalize=True)[:10].plot(kind="bar",
title="Top Percentage of sales by product").set(xlabel="Category",ylabel="Percentage")
```



Note: From the bar charts above, we infer that Kurta & Kurtis contribute nearly 13.7% and it is the best-selling item in the E-commerce dataset, followed by tops & tees(7%) and smart phones (6.9%).

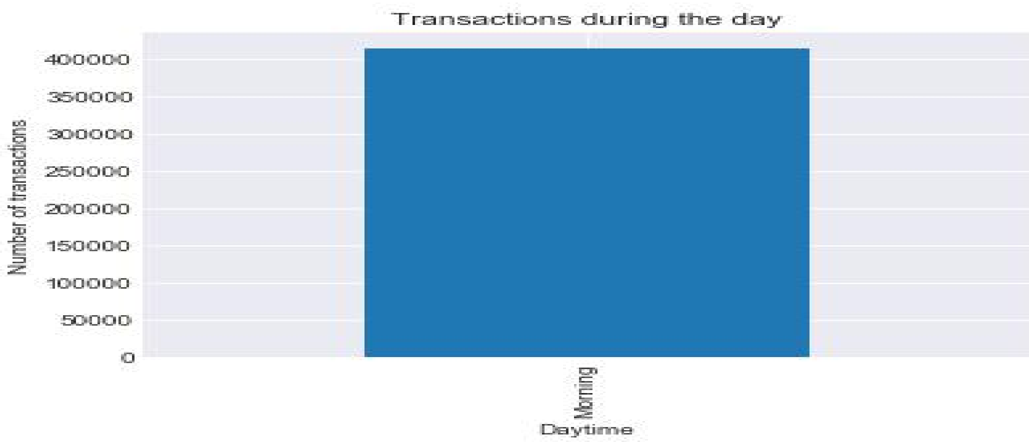
```
df['category'].value_counts().sort_values(ascending=False).head(10).plot(kind='bar')
plt.ylabel('Number of transactions')
plt.xlabel('Category')
plt.title('Top 10 Purchased Categories')
```



3) From Time Stamp[epoch] to check whether at what time during the day has more product purchase whether in the morning,evening or during night:

```
df.loc[(df['Datetime']<'12:00:00'),'Daytime']='Morning'
df.loc[(df['Datetime']>='12:00:00')&(df['Datetime']<'17:00:00'),'Daytime']='Afternoon'
df.loc[(df['Datetime']>='17:00:00')&(df['Datetime']<'21:00:00'),'Daytime']='Evening'
df.loc[(df['Datetime']>='21:00:00')&(df['Datetime']<'23:50:00'),'Daytime']='Night'
```

```
sns.set_style('darkgrid')
df.groupby('Daytime')['category'].count().sort_values().plot(kind='bar')
plt.ylabel('Number of transactions')
plt.title('Transactions during the day')
```



Conclusion:

Note : It is evident from above chart that almost all the E-commerce product purchases are done during Morning.

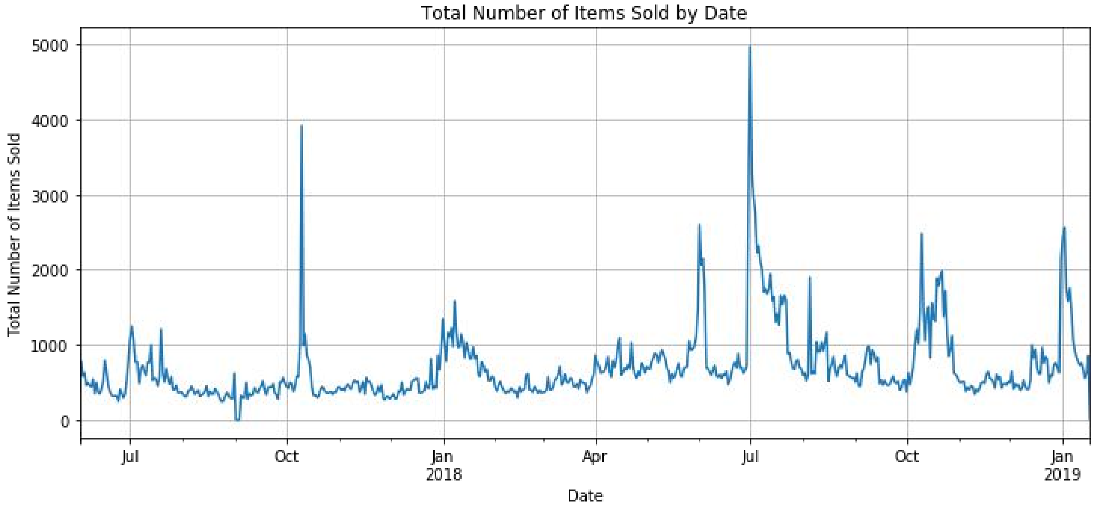
```
df['Datetime'] = pd.to_datetime(df['Time Stamp'])
df2 = df[["Datetime", "uuid", "product_id"]].set_index("Datetime")
df2.head(10)
```

```
total_items = len(df2)
total_days = len(np.unique(df2.index.date))
total_months = len(np.unique(df2.index.month))
average_items = total_items / total_days
unique_items = df2.product_id.unique().size


print("There are {} unique items sold ".format(unique_items))
print("Total {} items sold in {} days throughout {} months".format(total_items, total_days, total_months))
print("With an average of {} items sold daily".format(average_items))
```

```
There are 173030 unique items sold
Total 413913 items sold in 593 days throughout 12 months
With an average of 697.9983136593592 items sold daily
```

Total No of Items Sold by Date:

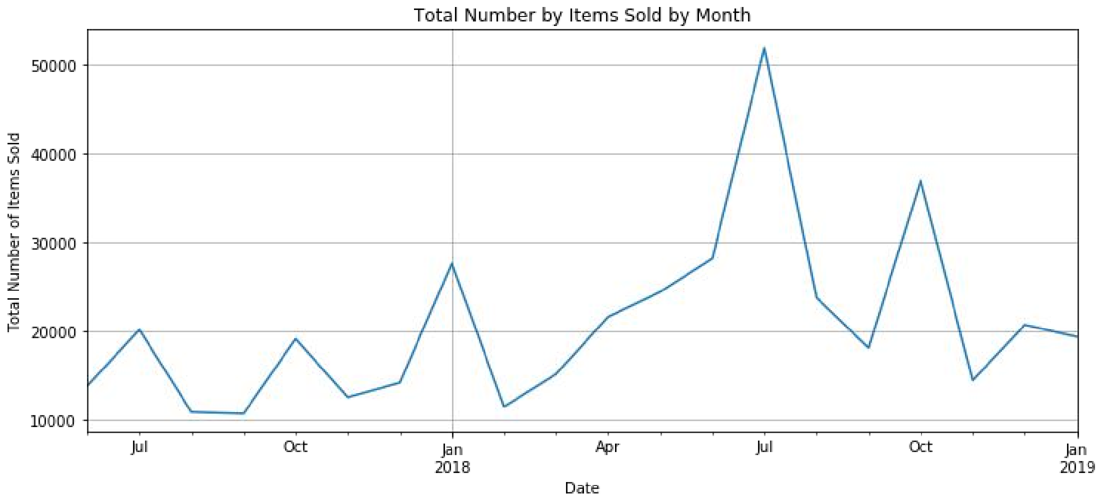
```
df2["product_id"].resample("D").count().plot(figsize=(12,5),
grid=True, title="Total Number of Items Sold by Date").set(xlabel="Date", ylabel="Total Number of Items Sold")
```



Note: Total Number of Items Sold and their fluctuations for the period of 593 days.

Total No of Items Sold by Month:

```
df2["product_id"].resample("M").count().plot(figsize=(12,5),
grid=True, title="Total Number by Items Sold by Month").set(xlabel="Date", ylabel="Total Number of Items Sold")
```



Note:

July Month is the one where the more no of transactions were done.

```
# import the libraries required
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import scipy
import scipy as sc
import pickle

import matplotlib.pyplot as plt
import matplotlib as mpl

from mlxtend.frequent_patterns import association_rules
import itertools

import arff
import pandas as pd

# from dotenv import load_dotenv
from flask import (
    Flask,
    render_template,
    redirect,
    request
)
from mlxtend.frequent_patterns import (
    apriori,
    association_rules,
)
from scipy.io import arff
import itertools
from flask import jsonify
```

```
app = Flask(__name__)


df = pd.read_csv("C:/Users/karth/OneDrive/Desktop/clickStreams.csv")

df.category.value_counts(normalize=True)[:10]
df = df.groupby(["uuid","category"]).size().reset_index(name="Count")
basket = (df.groupby(['uuid', 'category'])['Count']
          .sum().unstack().reset_index().fillna(0)
          .set_index('uuid'))
basket.head()

def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

basket_sets = basket.applymap(encode_units)
frequent_itemsets = apriori(basket_sets, min_support=0.01, use_colnames=True)
itemset_count=len(frequent_itemsets)

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules_count = len(rules)
rules.sort_values("confidence", ascending = False, inplace = True)
rules.head(10)

items = sorted(
    set(itertools.chain.from_iterable(frequent_itemsets.itemsets.values)))
basket = set()
```

Output for the Above Code Statement:

```
In [57]: rules
Out[57]:
        antecedents       consequents  antecedent support  \
7    (ethnic bottom)  (kurta & kurtis)            0.044743
0        (suit sets)  (kurta & kurtis)            0.025550
17    (tops and tees)  (kurta & kurtis)            0.077801
13  (casual dresses)  (kurta & kurtis)            0.060922
9   (casual dresses)   (tops and tees)            0.060922
15   (ethnic bottom)   (tops and tees)            0.044743
8    (tops and tees)  (casual dresses)            0.077801
6   (kurta & kurtis)   (ethnic bottom)            0.132917
5    (casual shirts)        (t-shirts)            0.061299
4         (t-shirts)   (casual shirts)            0.067084
16  (kurta & kurtis)   (tops and tees)            0.132917
10   (tops and tees)          (sandal)            0.077801
12  (kurta & kurtis)  (casual dresses)            0.132917
14   (tops and tees)   (ethnic bottom)            0.077801
3     (shoe slip-on)  (shoe lace - up)            0.083048
2   (shoe lace - up)    (shoe slip-on)            0.086196
11          (sandal)   (tops and tees)            0.111881
1   (kurta & kurtis)       (suit sets)            0.132917

    consequent support   support  confidence      lift  leverage  conviction
7             0.132917  0.027582    0.616449  4.637842  0.021635    2.260670
0             0.132917  0.012022    0.470511  3.539881  0.008626    1.637584
17            0.132917  0.023613    0.303502  2.283392  0.013272    1.244918
13            0.132917  0.018372    0.301568  2.268842  0.010275    1.241471
9             0.077801  0.017699    0.290526  3.734228  0.012960    1.299834
15            0.077801  0.010407    0.232597  2.989646  0.006926    1.201714
8             0.060922  0.017699    0.227497  3.734228  0.012960    1.215630
6             0.044743  0.027582    0.207511  4.637842  0.021635    1.205388
5             0.067084  0.012694    0.207090  3.087009  0.008582    1.176571
4             0.061299  0.012694    0.189230  3.087009  0.008582    1.157790
16            0.077801  0.023613    0.177650  2.283392  0.013272    1.121419
10            0.111881  0.011039    0.141894  1.268255  0.002335    1.034975
12            0.060922  0.018372    0.138222  2.268842  0.010275    1.089699
14            0.044743  0.010407    0.133766  2.989646  0.006926    1.102770
3             0.086196  0.010239    0.123289  1.430325  0.003080    1.042309
2             0.083048  0.010239    0.118786  1.430325  0.003080    1.040555
11            0.077801  0.011039    0.098671  1.268255  0.002335    1.023155
1             0.025550  0.012022    0.090444  3.539881  0.008626    1.071347
```

Interpretation for above Output:

- ❖ **Above Output Shows all categories which has support over 1% and lift value over 1.**
- ❖ **The first categories (i.e [Ethnic Bottom], [kurta & Kurtis]) have a support value of 0.044743 means nearly 4.4% of all transactions have this combination being bought together.**
- ❖ **Confidence that [Ethnic Bottom], [kurta & Kurtis]) purchase may happen together is 61.6%**
- ❖ **Lift Value of 4.637842 (greater than 1) shows that purchase of [kurta & Kurtis] is influenced by ethnic bottom**
- ❖ **Lift Value of 4.637842 means [Ethnic Bottom] purchase lifts the[kurta & Kurtis]) purchase by 4.637842 times.**

Conclusion:

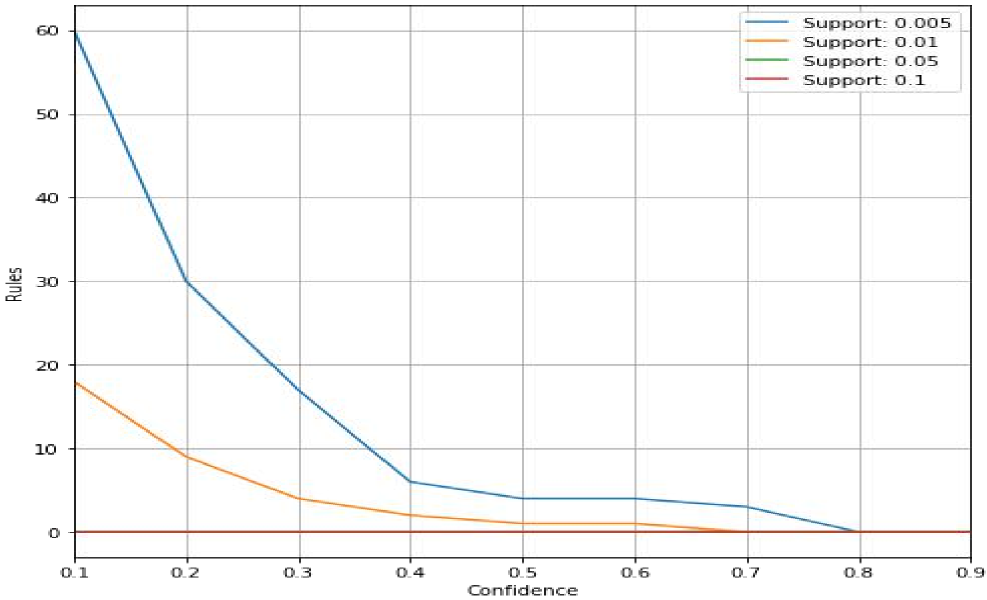
Therefore, we can conclude that there is indeed evidence to suggest that the purchase of **[Ethnic Bottom]** leads to the purchase of **[kurta & Kurtis]).**

Apriori Algorithm With Different Support Levels:

Objective:

To find the ideal threshold value between No of Rules and Confidence Lmit at various Support Intervals:

```
df = pd.read_csv("C:/Users/karth/OneDrive/Desktop/clickStreams.csv")
from mlxtend.preprocessing import TransactionEncoder

transactions=[]
item_sets = {}

for t,g in df.groupby('uuid')['category']:
    transactions.append(g.tolist())
item_sets[t] = g.tolist()
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
ap = pd.DataFrame(te_ary, columns=te.columns_)

ap_0_5 = {}
ap_1 = {}
ap_5 = {}
ap_1_0 = {}

confidence = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]

def gen_rules(df,confidence,support):
    ap = {}
    for i in confidence:
        ap_i =apriori(df,support,True)
        rule= association_rules(ap_i,min_threshold=i)
        ap[i] = len(rule.antecedents)
        return pd.Series(ap).to_frame("Support: %s"%support)

confs = []
for i in [0.005,0.01,0.05,0.1]:
    ap_i = gen_rules(ap,confidence=confidence,support=i)
confs.append(ap_i)

all_conf = pd.concat(confs,axis=1)

all_conf.plot(figsize=(8,8),grid=True)
plt.ylabel('Rules')
plt.xlabel('Confidence')
plt.show()
```



Analyzing Result:

- ❖ **Support level of 10%.** We only identify a few rules with very low confidence levels. This means that there are no relatively frequent associations in our data set. We can't choose this value, the resulting rules are unrepresentative.

- ❖ **Support level of 1%**. We started to get more no of rules,and also have a confidence of at least 50%.

- ❖ **Support level of 0.5%.** Too many rules to analyze.

Conclusion:

The ideal threshold value between No of Rules and Confidence Limit is evident from the above graph and We are going to use a support level of 1% and a confidence level of 50%.

Testing of Sample Data using Rest Api[Flask] and check whether there is any influence on any product

Itemset count: 40
Rules count: 18

Basket

- ethnic bottom

Reset basket

Recommendations

- kurta & kurtis
- tops and tees

Add to basket

Items available

Add to basket

- analog
- boot
- bras & bra sets
- casual dresses
- casual jackets and blazers
- casual shirts
- casual tops & tees
- casual trousers

```
In [8]: ap_final =  apriori(ap,0.01,True)
   ...: rules_final = association_rules(ap_final,min_threshold=.03,support_only=False)
   ...:
   ...: rules_final[rules_final['confidence'] > 0.5]
Out[8]:
       antecedents       consequents  antecedent support  consequent support  \
4  (ethnic bottom)  (kurta & kurtis)            0.044743            0.132917

    support  confidence      lift  leverage  conviction
4  0.027582    0.616449  4.637842  0.021635     2.26067
```

Conclusion:

On choosing Ethnic bottom We get Recommendations { Kurta & Kurtis and Tops & Tees]

The confidence for the above to happen is 61.64%.