Kingdome of Saudi Arabia
Ministry of Higher Education
Taibah University
College of Computer Science and Engineering
Department of Information Systems

المملكة العربية السعودية
وزارة التعليم العالي
جـامـعـة طـيـبـة
كلية علوم وهندسة الحاسب الآلي
قسم نظم المعلومات

IS372 – Data warehouse and data mining

Analysis sales transactions

Professor. Mohammed Al-Sarem

Team member :

Nosiabh Humod Alamri 4051085

Shahad Mansour Aljohani 4052322

Shahad Khaled Almuzaini 4052253

Haneen Mohammed Alzoughibi 4051033

Section:

IA8G

Date:

24/5/2022

Introduction

One of the problems facing the market is the inability to deal with huge data and thus not achieving the expected income, so the rules of association can help solve this problem by extracting data that frequently appear with each other, to improve marketing plans and analyze the purchasing habits of customers and the difficulties that we will face Finding relationships of association rules between different products.

We used the rules of association, which is a simple and suitable technique for analyzing purchase data and searching for groups of repeating elements. It consists of two steps: creating groups of repeating elements, creating the appropriate rule.

The type of data used is Transaction data obtained from the "kaggle" site and its name is Sales Product Data.

Libraries

The database was downloaded as an excel file and these libraries were imported:

Panada is one of the most important libraries in Python for data manipulation and analysis.

NumPy provides an extensive library of high-level mathematical functions

Seaborn Python data visualization library based on matplotlib

matplotlib.pyplot comprehensive library for creating static, animated, and interactive visualizations in Python.

datetime allow us to manipulate dates and times.

OS This module provides a portable way of using operating system-dependent functionality.

Warnings to warn the developer of situations that aren’t necessarily exceptions.

Pip install mlxtend To install mlxtend.

mlxtend.frequent_patterns import fpgrowth Function implementing FP-Growth to extract frequent itemsets for association rule mining.

mlxtend.frequent_patterns import association_rules Function to generate association rules from frequent itemsets.

Timeline

Task	Deadline	lead
Search For the Dataset	20/5/2022	All team member
Download It	22/5/2022	Shahad Khaled
Data Understanding and Preparations	22/5/2022	All
Model Building	23/5/2022	All
Evaluation And Testing	23/5/2022	All
Prepare Report	24/5/2022	All
Presentations	25/5/20200	All

Dataset description

Reviewing the Dataset

In [2]:

```
df = pd.read_csv(r'C:\Users\pc\Documents\shahad\python project\archive\all_data.csv'
df.head()
```

Out[2]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	236670	Wired Headphones	2	11.99	08/31/19 22:21	359 Spruce St, Seattle, WA 98101
1	236671	Bose SoundSport Headphones	1	99.99	08/15/19 15:11	492 Ridge St, Dallas, TX 75001
2	236672	iPhone	1	700.0	08/06/19 14:40	149 7th St, Portland, OR 97035
3	236673	AA Batteries (4-pack)	2	3.84	08/29/19 20:59	631 2nd St, Los Angeles, CA 90001
4	236674	AA Batteries (4-pack)	2	3.84	08/15/19 19:53	736 14th St, New York City, NY 10001

Statistical analysis of the dataset

Reviewing some basic statistical details

```
In [23]: df.describe()
```

Out[23]:

	Order ID	Quantity Ordered	Price Each
count	185688.000000	185688.000000	185688.000000
mean	230408.894522	1.124531	184.517268
std	51516.989791	0.443082	332.842597
min	0.000000	0.000000	0.000000
25%	185831.750000	1.000000	11.950000
50%	230354.000000	1.000000	14.950000
75%	275028.250000	1.000000	150.000000
max	319670.000000	9.000000	1700.000000

Attributes' classification

Reviewing data type to all attributes before pre-processing

```
# data type each column
df.dtypes
```

```
Out[4]:

Order ID            object
Product             object
Quantity Ordered    object
Price Each          object
Order Date          object
Purchase Address    object
dtype: object
```

Data pre-processing

Handling with missing values

```
df['Product'].fillna('Nan ', inplace=True)
df['Order Date'].fillna('Nan', inplace=True)
df['Purchase Address'].fillna('Nan', inplace=True)
df.isnull().sum()
```

```
Out[17]:

Order ID            0
Product             0
Quantity Ordered    0
Price Each          0
Order Date          0
Purchase Address    0
dtype: int64
```

drop identical duplicates rows

```
df.duplicated().sum()
```

Out[13]:

```
0
```

```
df.duplicated().values.any()
```

Out[14]:

```
False
```

convert columns to correct data types

```
#convert columns to correct data types
df["Order ID"] = pd.to_numeric(df["Order ID"], errors='coerce').fillna(0, downcast='infer')
df["Price Each"] = pd.to_numeric(df["Price Each"], errors='coerce').fillna(0, downcast='inf
df["Quantity Ordered"] = pd.to_numeric(df["Quantity Ordered"], errors='coerce').fillna(0, d
df['Product']= df['Product'].convert_dtypes()
df['Purchase Address']= df['Purchase Address'].convert_dtypes()

df.dtypes
```

Out[11]:

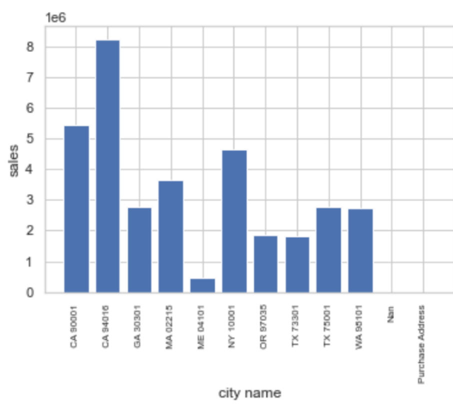
```
Order ID              int64
Product              string
Quantity Ordered      int64
Price Each          float64
Order Date           object
Purchase Address     string
dtype: object
```

Visualization

Reviewing most sales city

The highest sales were in San Francisco on Hill St, the lowest sales were in Portland on North St.

```
import matplotlib.pyplot as plt
cities = [city for city , df in df.groupby('city')]
plt.bar(cities,city_result['sales'])
plt.xticks(cities, rotation = 'vertical',size = '8')
plt.ylabel('sales')
plt.xlabel('city name')
plt.show()
```

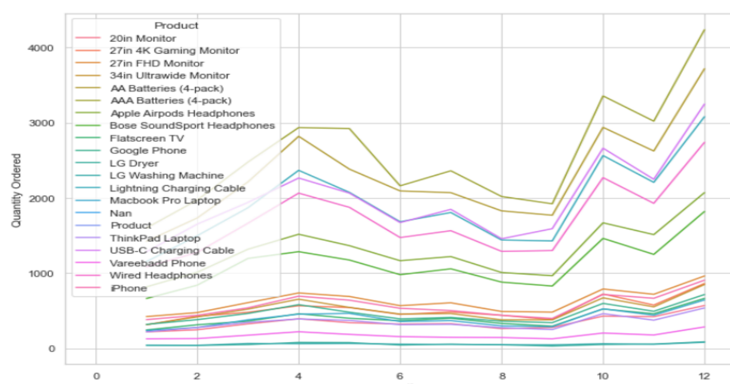


It shows that almost all products have the same sales pattern over the months.

```
fig, ax = plt.subplots()
fig.set_size_inches(11,8)
sns.lineplot(data = itemsMonthly, x='month',y='Quantity Ordered', hue='Product',ax=ax)
```

```
Out[62]:

<AxesSubplot:xlabel='month', ylabel='Quantity Ordered'>
```

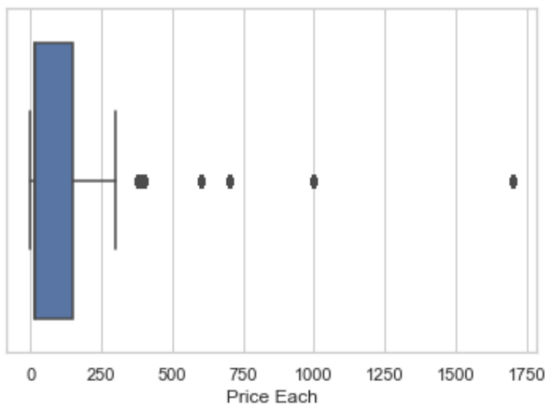


Box plot shows five numbers summary (Minimum, Q1, M, Q3, Maximum)

Price each

In [89]:
```
#df[['Price Each']].plot.box()
sns.boxplot(x=df['Price Each'])
```

Out[89]: <AxesSubplot:xlabel='Price Each'>



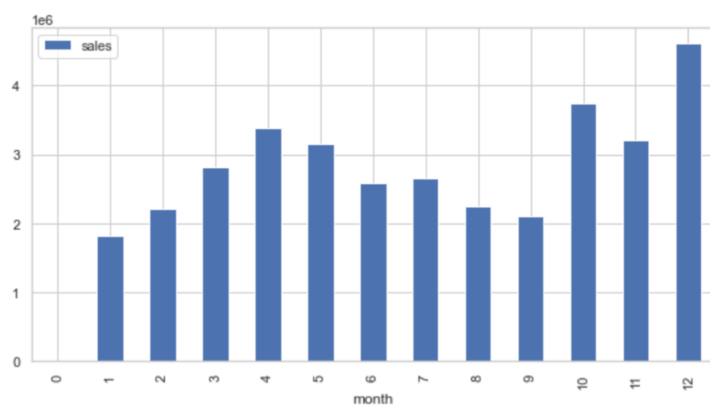
Reviewing the best month of sales

Bar Plot shows the best sales were in December, and the lowest sales were in January.

```
monthlySales = df.groupby('month').agg({'sales' : 'sum'})
monthlySales.plot.bar(figsize=(10,5))
```

Out[60]:

<AxesSubplot:xlabel='month'>



Box plot shows five numbers summary (Minimum, Q1, M, Q3, Maximum)

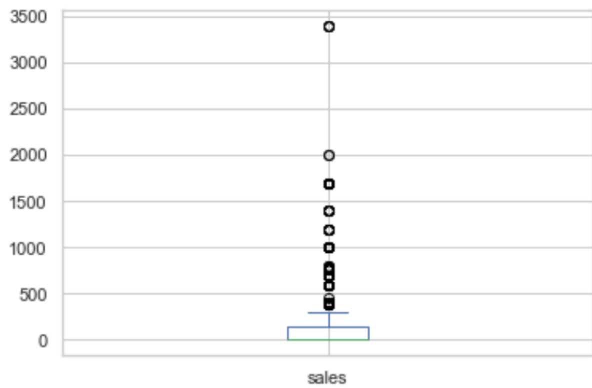
Sales.

In [91]:

```
df[['sales']].plot.box()
```

Out[91]:

```
<AxesSubplot:>
```



Data mining algorithm/s

Apriori

Algorithm for finding frequent items in a dataset

```
frequent_itemsets_fp=fpgrowth(Tran_sets,min_support=0.001,use_colnames=True)
frequent_itemsets_fp
```

Out[77]:

	support	itemsets
0	0.038333	(iPhone)
1	0.121073	(Lightning Charging Cable)
2	0.105622	(Wired Headphones)
3	0.042020	(27in FHD Monitor)
4	0.115407	(AAA Batteries (4-pack))
5	0.034886	(27in 4K Gaming Monitor)
6	0.122480	(USB-C Charging Cable)
7	0.074524	(Bose SoundSport Headphones)
8	0.087005	(Apple Airpods Headphones)
9	0.026457	(Macbook Pro Laptop)
10	0.026866	(Flatscreen TV)
11	0.011573	(Vareebadd Phone)
12	0.115121	(AA Batteries (4-pack))
13	0.030946	(Google Phone)
14	0.022966	(20in Monitor)
15	0.034600	(34in Ultrawide Monitor)

FP growth

An algorithm that stores the frequency of occurrence of itemsets


```
frequent_itemsets_fp=fpgrowth(Tran_sets,min_support=0.001,use_colnames=True)
frequent_itemsets_fp
```

Out[77]:

	support	itemsets
0	0.038333	(iPhone)
1	0.121073	(Lightning Charging Cable)
2	0.105622	(Wired Headphones)
3	0.042020	(27in FHD Monitor)
4	0.115407	(AAA Batteries (4-pack))
5	0.034886	(27in 4K Gaming Monitor)
6	0.122480	(USB-C Charging Cable)
7	0.074524	(Bose SoundSport Headphones)
8	0.087005	(Apple Airpods Headphones)
9	0.026457	(Macbook Pro Laptop)
10	0.026866	(Flatscreen TV)
11	0.011573	(Vareebadd Phone)
12	0.115121	(AA Batteries (4-pack))
13	0.030946	(Google Phone)
14	0.022966	(20in Monitor)
15	0.034600	(34in Ultrawide Monitor)

Model validation

Apriori

Testing the algorithm accuracy

```
from mlxtend.frequent_patterns import association_rules, apriori
rules = association_rules(frequent_items, metric = "lift", min_threshold = 0.001)
rules.sort_values('confidence', ascending = False, inplace = True)
rules
```

Out[74]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lever
4	(Google Phone)	(USB-C Charging Cable)	0.030946	0.122480	0.005587	0.180551	1.474128	0.001
10	(Vareebadd Phone)	(USB-C Charging Cable)	0.011573	0.122480	0.002062	0.178208	1.455004	0.000
8	(iPhone)	(Lightning Charging Cable)	0.038333	0.121073	0.005666	0.147807	1.220810	0.001
6	(Google Phone)	(Wired Headphones)	0.030946	0.105622	0.002365	0.076422	0.723538	-0.000
14	(iPhone)	(Wired Headphones)	0.038333	0.105622	0.002589	0.067544	0.639486	-0.001
1	(iPhone)	(Apple Airpods Headphones)	0.038333	0.087005	0.002090	0.054532	0.626770	-0.001
9	(Lightning Charging Cable)	(iPhone)	0.121073	0.038333	0.005666	0.046797	1.220810	0.001
5	(USB-C Charging Cable)	(Google Phone)	0.122480	0.030946	0.005587	0.045619	1.474128	0.001
3	(Google Phone)	(Bose SoundSport Headphones)	0.030946	0.074524	0.001278	0.041289	0.554038	-0.001
15	(Wired Headphones)	(iPhone)	0.105622	0.038333	0.002589	0.024513	0.639486	-0.001
0	(Apple Airpods Headphones)	(iPhone)	0.087005	0.038333	0.002090	0.024026	0.626770	-0.001

FP growth

Testing the algorithm accuracy

```
rules_fb = association_rules(frequent_itemsets_fp, metric = "confidence", min_threshold = 0
rules_fb
```

Out[78]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lever
0	(Apple Airpods Headphones)	(iPhone)	0.087005	0.038333	0.002090	0.024026	0.626770	-0.001
1	(iPhone)	(Apple Airpods Headphones)	0.038333	0.087005	0.002090	0.054532	0.626770	-0.001
2	(iPhone)	(Wired Headphones)	0.038333	0.105622	0.002589	0.067544	0.639486	-0.001
3	(Wired Headphones)	(iPhone)	0.105622	0.038333	0.002589	0.024513	0.639486	-0.001
4	(iPhone)	(Lightning Charging Cable)	0.038333	0.121073	0.005666	0.147807	1.220810	0.001
5	(Lightning Charging Cable)	(iPhone)	0.121073	0.038333	0.005666	0.046797	1.220810	0.001
6	(Wired Headphones)	(USB-C Charging Cable)	0.105622	0.122480	0.001138	0.010771	0.087941	-0.011
7	(USB-C Charging Cable)	(Wired Headphones)	0.122480	0.105622	0.001138	0.009288	0.087941	-0.011
8	(Vareebadd Phone)	(USB-C Charging Cable)	0.011573	0.122480	0.002062	0.178208	1.455004	0.000
9	(USB-C Charging Cable)	(Vareebadd Phone)	0.122480	0.011573	0.002062	0.016838	1.455004	0.000
10	(Google Phone)	(USB-C Charging Cable)	0.030946	0.122480	0.005587	0.180551	1.474128	0.001

Final result

As we can see from the evaluation result are similar to each other and that proof the accuracy of the algorithm..

References :

- Knightbearr. (2022, February 5). *Analysis: Sales data (knightbearr)*. Kaggle. Retrieved May 25, 2022, from https://www.kaggle.com/code/knightbearr/analysis-sales-data-knightbearr/data

- Lab sheet (Lab 4: Data Preparation- NumPy Library, Lab 5: Data Exploratory Analysis, Lab 6: Dealing with Missing Values & PCA, Lab 8: Association Rules)

- Ganguly, M. (2021, April 15). *This is how you can analyze any dataset with the CRISP-DM framework*. Medium. Retrieved May 25, 2022, from https://gangulym23.medium.com/this-is-how-you-can-analyze-any-dataset-with-the-crisp-dm-framework-cc9353f4dabe

- YouTube. (2021). *YouTube*. Retrieved May 25, 2022, from https://www.youtube.com/watch?v=Cryve9ZWbYk.