|  |  |
| --- | --- |
| **Practical 1** | |
| **Aim**: To perform data Pre-processing task and demonstrate Classification algorithm of K nearest neighbour for the given dataset. | |
| Name: Aryan Penikal | Roll No: KCTBCS050 |
| Performance date: | Sign: |

**Theory:**

Supervised learning is the type of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. Where, the labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

**How Supervised Learning Works?**

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

**Steps Involved in Supervised Learning:**

* First Determine the type of training dataset
* Collect/Gather the labelled training data.
* Split the training dataset into training dataset, test dataset, and validation dataset.
* Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
* Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
* Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
* Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

**Types of Supervised learning:**

Supervised learning can be separated into two types of problems when data mining—classification and regression:

* Classification uses an algorithm to accurately assign test data into specific categories. It recognizes specific entities within the dataset and attempts to draw some conclusions on how those entities should be labeled or defined. Common classification algorithms are linear classifiers, support vector machines (SVM), decision trees, k-nearest neighbor, and random forest, which are described in more detail below.
* Regression is used to understand the relationship between dependent and independent variables. It is commonly used to make projections, such as for sales revenue for a given business. Linear regression, logistical regression, and polynomial regression are popular regression algorithms.

## Advantages of Supervised learning:

* With the help of supervised learning, the model can predict the output on the basis of prior experiences.
* In supervised learning, we can have an exact idea about the classes of objects.
* Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering**, etc.

## Disadvantages of supervised learning:

* Supervised learning models are not suitable for handling the complex tasks.
* Supervised learning cannot predict the correct output if the test data is different from the training dataset.
* Training required lots of computation times.
* In supervised learning, we need enough knowledge about the classes of object.

**KNN algorithm.**

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
* It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
* KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
* **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

## How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

* **Step-1:** Select the number K of the neighbors
* **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
* **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
* **Step-4:** Among these k neighbors, count the number of the data points in each category.
* **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
* **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



* Firstly, we will choose the number of neighbors, so we will choose the k=5.
* Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



* By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



* As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

## How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

* There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
* A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
* Large values for K are good, but it may find some difficulties.

**3. Advantages and Disadvantages of KNN.**

## Advantages of KNN Algorithm:

* It is simple to implement.
* It is robust to the noisy training data
* It can be more effective if the training data is large.

## Disadvantages of KNN Algorithm:

* We always needs to determine the value of K which may be complex some time.
* The computation cost is high because of calculating the distance between the data points for all the training samples.
* Might not be as accurate as some other learning algorithms.

**Code**

* 1. **Dataset 1:**

# -\*- coding: utf-8 -\*-

"""

Created on Thu Sep 22 21:39:51 2022

@author: ARYAN

"""

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

#Importing the dataset

dataset = pd.read\_csv("Social\_Network\_Ads.csv")

x = dataset.iloc[:,[2,3]].values

y = dataset.iloc[:,-1].values

dataset.head()

dataset.describe()

dataset.info()

dataset.isnull().sum()

#Splitting the dataset into training and testing

from sklearn.model\_selection import train\_test\_split

xtrain, xtest, ytrain, ytest = train\_test\_split(x,y,test\_size=0.20, random\_state = 90)

print("Size of x-training data: ", xtrain.shape)

print("Size of y-training data: ", ytrain.shape)

print("Size of x-test data: ", xtest.shape)

print("Size of y-test data: ", ytest.shape)

#Feature scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

xtrain = sc.fit\_transform(xtrain)

xtest = sc.transform(xtest)

#Training the knn model on the training set

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 7, metric = "minkowski", p=2)

classifier.fit(xtrain,ytrain)

#Predict the test set result

ypred = classifier.predict(xtest)

#Making confusion matrix

from sklearn.metrics import confusion\_matrix , accuracy\_score

cm = confusion\_matrix(ytest, ypred)

ac = accuracy\_score(ytest, ypred)

print("\nConfusion Matrix: \n",cm)

print("Accuracy of the model: ",ac)

#plotting elbow method graph

neighbors = np.arange(1,9)

train\_accuracy = np.empty(len(neighbors))

test\_accuracy = np.empty(len(neighbors))

for i,k in enumerate(neighbors):

knn = KNeighborsClassifier(n\_neighbors = k)

knn.fit(xtrain, ytrain)

train\_accuracy[i] = knn.score(xtrain, ytrain)

test\_accuracy[i] = knn.score(xtest, ytest)

plt.plot(neighbors, train\_accuracy, label="Train Accuracy")

plt.plot(neighbors, test\_accuracy, label="Test Accuracy")

plt.legend()

plt.xlabel("n\_neighbors")

plt.ylabel("Accuracy")

plt.show()

**Output**

runfile('A:/aryan - Copy/aryan - Copy/DWM pracs/prac2\_decision\_tree.py', wdir='A:/aryan - Copy/aryan - Copy/DWM pracs')

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 400 entries, 0 to 399

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 User ID 400 non-null int64

1 Gender 400 non-null object

2 Age 400 non-null int64

3 EstimatedSalary 400 non-null int64

4 Purchased 400 non-null int64

dtypes: int64(4), object(1)

memory usage: 15.8+ KB

Size of x-training data: (320, 2)

Size of y-training data: (320,)

Size of x-test data: (80, 2)

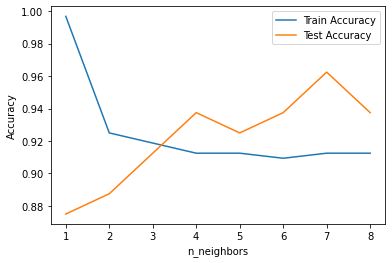
Size of y-test data: (80,)

Confusion Matrix:

[[52 2]

[ 1 25]]

Accuracy of the model: 0.9625



* 1. Dataset 2

**Code**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

dataset = pd.read\_csv("BankNote\_Authentication.csv")

x=dataset.iloc[:,[0,1,2,3]].values

y=dataset.iloc[:,-1].values

print(dataset.count)

from sklearn.model\_selection import train\_test\_split

xtrain, xtest,ytrain,ytest = train\_test\_split(x,y,test\_size = 0.20,random\_state=40)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

xtrain = sc.fit\_transform(xtrain)

xtest = sc.transform(xtest)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 3, metric = "minkowski", p=2)

classifier.fit(xtrain,ytrain)

ypred = classifier.predict(xtest)

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(ytest, ypred)

ac = accuracy\_score(ytest, ypred)

print("Confusion Matrix: ",cm)

print("Accuracy Score: ", ac)

#plotting elbow method graph

neighbors = np.arange(1,20)

train\_accuracy = np.empty(len(neighbors))

test\_accuracy = np.empty(len(neighbors))

for i,k in enumerate(neighbors):

knn = KNeighborsClassifier(n\_neighbors = k)

knn.fit(xtrain, ytrain)

train\_accuracy[i] = knn.score(xtrain, ytrain)

test\_accuracy[i] = knn.score(xtest, ytest)

plt.plot(neighbors, train\_accuracy, label="Train Accuracy")

plt.plot(neighbors, test\_accuracy, label="Test Accuracy")

plt.legend()

plt.xlabel("n\_neighbors")

plt.ylabel("Accuracy")

plt.show()

**Output**

runfile('A:/aryan - Copy/aryan - Copy/DWM pracs/prac2\_decision\_tree.py', wdir='A:/aryan - Copy/aryan - Copy/DWM pracs')

<bound method DataFrame.count of variance skewness curtosis entropy class

0 3.62160 8.66610 -2.8073 -0.44699 0

1 4.54590 8.16740 -2.4586 -1.46210 0

2 3.86600 -2.63830 1.9242 0.10645 0

3 3.45660 9.52280 -4.0112 -3.59440 0

4 0.32924 -4.45520 4.5718 -0.98880 0

... ... ... ... ...

1367 0.40614 1.34920 -1.4501 -0.55949 1

1368 -1.38870 -4.87730 6.4774 0.34179 1

1369 -3.75030 -13.45860 17.5932 -2.77710 1

1370 -3.56370 -8.38270 12.3930 -1.28230 1

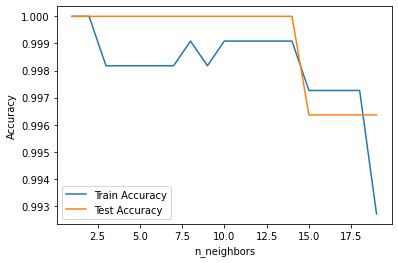
1371 -2.54190 -0.65804 2.6842 1.19520 1

[1372 rows x 5 columns]>

Confusion Matrix: [[147 0]

[ 0 128]]

Accuracy Score: 1.0



|  |  |
| --- | --- |
| **Practical 2** | |
| **Aim: To Demonstrate Classification algorithm of Decision Tree on the given Dataset.** | |
| **Name: Aryan Penikal** | **Roll no: KCTBCS050** |
| **Date of performance:** | **Sign:** |

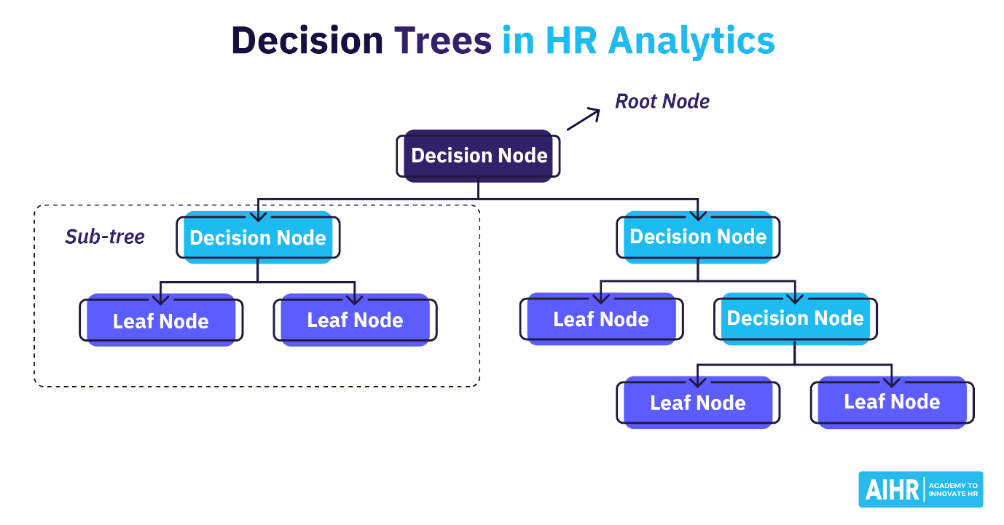
**Theory**

**Classification**

Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. Decision trees can handle both categorical and numerical data.

**Decision Tree Terminologies**

* **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
* **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
* **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
* **Branch/Sub Tree:** A tree formed by splitting the tree.
* **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
* **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.



**2. Algorithm along with formula**

The core algorithm for building decision trees called **ID3** by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking

ID3 uses Entropy and Information Gain to construct a decision tree.

The ID3 algorithm follows the below workflow in order to build a Decision Tree:

1. Calculate entropy for dataset.
2. For each attribute/feature.  
   2.1. Calculate entropy for all its categorical values.  
   2.2. Calculate information gain for the feature.
3. Find the feature with maximum information gain.
4. Repeat it until we get the desired tree.

Two measures are used to decide the best attribute:

1. Information Gain
2. Entropy

Entropy measures the impurity or uncertainty present in the data. It is used to decide how a Decision Tree can split the data.   
**Equation For Entropy:**

### Entropy formula - Decision Tree Algorithm - Edureka

Information Gain (IG) is the most significant measure used to build a Decision Tree. It indicates how much “information” a particular feature/ variable gives us about the final outcome.

Information Gain is important because it used to choose the variable that best splits the data at each node of a Decision Tree. The variable with the highest IG is used to split the data at the root node.

**Equation For Information Gain (IG):**

Information Gain formula - Decision Tree Algorithm - Edureka

**3. Advantages and Disadvantages**

|  |  |
| --- | --- |
| **Advantages** | Disadvantage |
| 1. Compared to other algorithms decision trees requires less effort for data preparation during pre-processing. | 1. A small change in the data can cause a large change in the structure of the decision tree causing instability. |
| 1. A decision tree does not require normalization of data. | 1. Decision tree often involves higher time to train the model. |
| 1. A decision tree does not require scaling of data as well. | 1. For a Decision tree sometimes calculation can go far more complex compared to other algorithms. |
| 1. Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent. | 1. The Decision Tree algorithm is inadequate for applying regression and predicting continuous values. |
| 1. A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders. | 1. Decision tree training is relatively expensive as the complexity and time has taken are more. |

**Code**

1. **DATASET 1**

# -\*- coding: utf-8 -\*-

"""

Created on Thu Sep 22 21:46:32 2022

@author: ARYAN

"""

import pandas as pd

#importing datasets

dataset= pd.read\_csv('./Social\_Network\_Ads.csv')

#Extracting Independent and dependent Variable

x= dataset.iloc[:, [2,3]].values

y= dataset.iloc[:, 4].values

dataset.count

# Splitting the dataset into training and test set.

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.25, random\_state=0)

#feature Scaling

from sklearn.preprocessing import StandardScaler

st\_x= StandardScaler()

x\_train= st\_x.fit\_transform(x\_train)

x\_test= st\_x.transform(x\_test)

#Fitting Decision Tree classifier to the training set

from sklearn.tree import DecisionTreeClassifier

classifier= DecisionTreeClassifier(criterion='entropy', random\_state=0)

classifier.fit(x\_train, y\_train)

#Predicting the test set result

y\_pred= classifier.predict(x\_test)

#Creating the Confusion matrix

from sklearn.metrics import confusion\_matrix, accuracy\_score

from sklearn import tree

cm = confusion\_matrix(y\_test, y\_pred)

ac = accuracy\_score(y\_test, y\_pred)

print ("Confusion Matrix : \n", cm)

print ("Accuracy : ", ac)

tree.plot\_tree(classifier)

**Output**

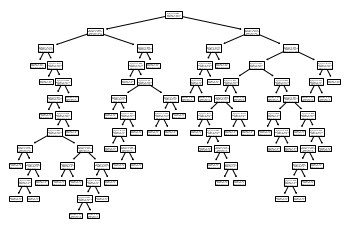
runfile('A:/aryan - Copy/aryan - Copy/DWM pracs/prac1\_knn.py', wdir='A:/aryan - Copy/aryan - Copy/DWM pracs')

Confusion Matrix :

[[62 6]

[ 3 29]]

Accuracy : 0.91



1. **DATASET 2**

**Code**

# -\*- coding: utf-8 -\*-

import pandas as pd

dataset = pd.read\_csv("UniversalBank.csv")

x=dataset.iloc[:,[1,2,3,5,6,7,8,9,10,11,12]].values

y=dataset.iloc[:,-1].values

print(dataset.count)

from sklearn.model\_selection import train\_test\_split

xtrain, xtest,ytrain,ytest = train\_test\_split(x,y,test\_size = 0.20, random\_state=40)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

xtrain = sc.fit\_transform(xtrain)

xtest = sc.transform(xtest)

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion='entropy', random\_state=40,max\_depth=3)

classifier.fit(xtrain, ytrain)

ypred = classifier.predict(xtest)

from sklearn.metrics import confusion\_matrix, accuracy\_score

from sklearn import tree

cm = confusion\_matrix(ytest, ypred)

ac = accuracy\_score(ytest, ypred)

print("Confusion Matrix: ",cm)

print("Accuracy Score: ", ac)

tree.plot\_tree(classifier)

**Output**

runfile('A:/aryan - Copy/aryan - Copy/DWM pracs/prac1\_knn.py', wdir='A:/aryan - Copy/aryan - Copy/DWM pracs')

<bound method DataFrame.count of ID Age Experience ... CD Account Online CreditCard

0 1 25 1 ... 0 0 0

1 2 45 19 ... 0 0 0

2 3 39 15 ... 0 0 0

3 4 35 9 ... 0 0 0

4 5 35 8 ... 0 0 1

... ... ... ... ... ... ...

4995 4996 29 3 ... 0 1 0

4996 4997 30 4 ... 0 1 0

4997 4998 63 39 ... 0 0 0

4998 4999 65 40 ... 0 1 0

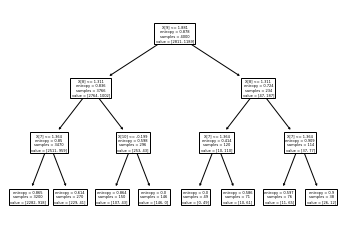
4999 5000 28 4 ... 0 1 1

[5000 rows x 14 columns]>

Confusion Matrix: [[712 7]

[233 48]]

Accuracy Score: 0.76



|  |  |
| --- | --- |
| **Practical 3** | |
| **Aim: To pre-process the data and demonstrate Clustering algorithm of K-means Clustering .** | |
| **Name: Aryan Penikal** | **Roll no: KCTBCS050** |
| **Date of performance** | **Sign:** |

**Practical 3**

Perform data clustering using clustering algorithm (K-Means Clustering )

**Theory**

**Clustering?**

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

* Determines the best value for K center points or centroids by an iterative process.
* Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

**Algorithm for K-means clustering**

The following are the steps involved in K-Means clustering:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be others from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

**Advantages of K-Means Clustering**

* Relatively simple to implement.
* Scales to large data sets.
* Guarantees convergence.
* Can warm-start the positions of centroids.
* Easily adapts to new examples.
* Generalizes to clusters of different shapes and sizes, such as elliptical clusters.

**Disadvantages of K-Means Clustering**

* Choosing k manually.
* Being dependent on initial values.
* Clustering data of varying sizes and density.
* Clustering outliers.
* Scaling with number of dimensions.

**Code**

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('Wholesale customers data(1).csv')

x = dataset.iloc[:, [3, 4]].values

#finding optimal number of clusters using the elbow method

from sklearn.cluster import KMeans

wcss\_list= [] #Initializing the list for the values of WCSS

#Using for loop for iterations from 1 to 10.

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++', random\_state= 42)

kmeans.fit(x)

wcss\_list.append(kmeans.inertia\_)

mtp.plot(range(1, 11), wcss\_list)

mtp.title('The Elobw Method Graph')

mtp.xlabel('Number of clusters(k)')

mtp.ylabel('wcss\_list')

mtp.show()

#training the K-means model on a dataset

kmeans = KMeans(n\_clusters=5, init='k-means++', random\_state= 42)

y\_predict= kmeans.fit\_predict(x)

#visulaizing the clusters

mtp.scatter(x[y\_predict == 0, 0], x[y\_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for first cluster

mtp.scatter(x[y\_predict == 1, 0], x[y\_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second cluster

mtp.scatter(x[y\_predict== 2, 0], x[y\_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for third cluster

mtp.scatter(x[y\_predict == 3, 0], x[y\_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #for fourth cluster

mtp.scatter(x[y\_predict == 4, 0], x[y\_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth cluster

mtp.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 300, c = 'yellow', label = 'Centroid')

mtp.title('Clusters of customers')

mtp.xlabel('Annual Income (k$)')

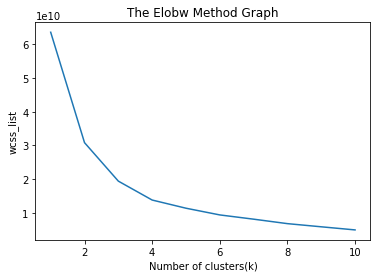
mtp.ylabel('Spending Score (1-100)')

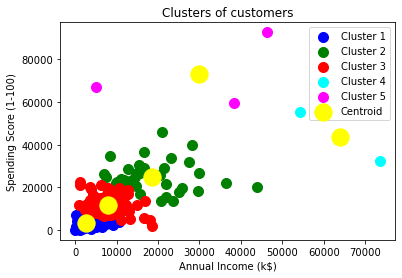
mtp.legend()

mtp.show()

**Output**

runfile('A:/aryan - Copy/aryan - Copy/DWM pracs/prac3a.py', wdir='A:/aryan - Copy/aryan - Copy/DWM pracs')





|  |  |
| --- | --- |
| **Data Warehousing and Mining Practical 4** | |
| **Practical 4: Hierarchical Clustering** | |
| **Name: Aryan Penikal** | **Roll no: KCTCS050** |
| **Date of performance: 06-08-22** | **Sign:** |

**Theory:**

**Agglomerative Hierarchical Clustering:** This method is also called a bottom-up approach. In this method, each node represents a single cluster at the beginning; eventually, nodes start merging based on their similarities until all nodes belong to the same cluster.

**Divisive Hierarchical Clustering**: This method is also called a top-down approach. Initially, all nodes belong to the same cluster; eventually, each node forms its own cluster. Divisive approach is less widely used due to its complexity compared with agglomerative approach.

**Dendrogram:** A Dendrogram is a tree-like diagram that records the sequences of merges / splits.

**Single Linkage**: It is the Shortest Distance between the closest points of the clusters.

**Complete Linkage**: It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.

**Average Linkage**: It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.

**Centroid Linkage**: It is the linkage method in which the distance between the centroid of the clusters is calculated.

**Hierarchical Clustering Algorithm Steps.**

Step-1: Compute the proximity matrix.

Step-2: Assign each data point as a single cluster.

Step-3: Merge two closest data points or clusters to form one cluster.

Step-4: Update the proximity matrix.

Step-5: Repeat Step 3 and Step 4 until only a single cluster remains.

Step-6: Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

**Advantages:**

* Dendrograms help us in clear visualization, which is practical and easy to understand.
* No prior information about the number of clusters is required.
* Easy to use and implement.

**Disadvantages:**

* Not suitable for large datasets due to high time and space complexity.
* There is no mathematical objective for Hierarchical clustering.
* The order of the data has an impact on the final results.
* It is very sensitive to outliers.

**Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Oct 18 23:32:32 2022

@author: ARYAN

"""

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering

x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]

y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

plt.scatter(x, y)

plt.show()

from scipy.cluster.hierarchy import dendrogram, linkage

data = list(zip(x, y))

print(data)

linkage\_data = linkage(data, method='ward', metric='euclidean')

dendrogram(linkage\_data)

plt.show()

hierarchical\_cluster = AgglomerativeClustering(n\_clusters=2, affinity='euclidean', linkage='ward')

labels = hierarchical\_cluster.fit\_predict(data)

plt.scatter(x, y, c=labels,cmap ='rainbow')

plt.show()

data = pd.read\_csv('Wholesale customers data.csv')

data.head()

from sklearn.preprocessing import normalize

data\_scaled = normalize(data)

data\_scaled = pd.DataFrame(data\_scaled, columns=data.columns)

data\_scaled.head()

import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))

plt.title("Dendrograms")

dend = shc.dendrogram(shc.linkage(data\_scaled, method='ward'))

from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n\_clusters=2, affinity='euclidean', linkage='ward')

cluster.fit\_predict(data\_scaled)

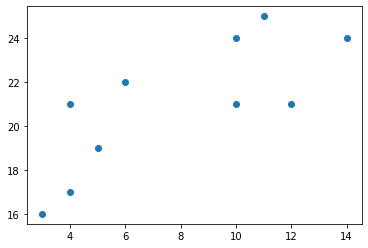
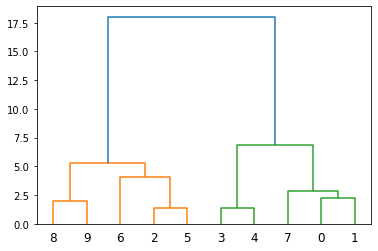
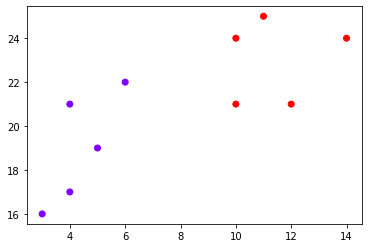
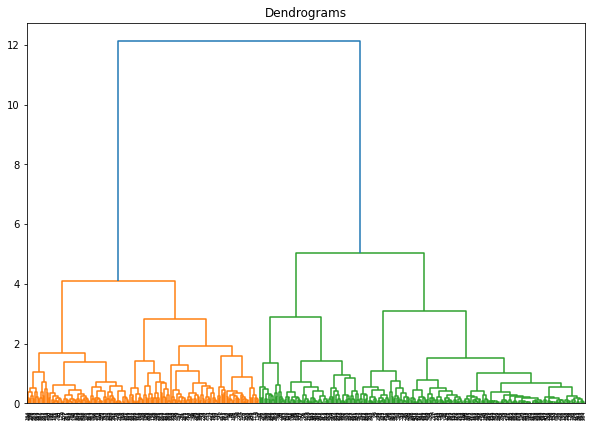
plt.figure(figsize=(10, 7))

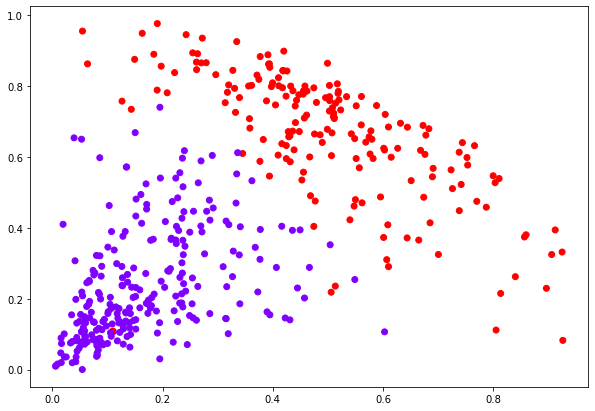
plt.scatter(data\_scaled['Milk'], data\_scaled['Grocery'], c=cluster.labels\_, cmap ='rainbow')

**Output:**

runfile('A:/aryan - Copy/aryan - Copy/DWM pracs/prac4.py', wdir='A:/aryan - Copy/aryan - Copy/DWM pracs')

[(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24), (6, 22), (10, 21), (12, 21)]



|  |  |
| --- | --- |
| **Data Warehousing and Mining Practical 5** | |
| **Practical 5: Market Basket Analysis** | |
| **Name: Aryan Penikal** | **Roll no: KCTCS050** |
| **Date of performance: 13-08-22** | **Sign:** |

**Theory:**

**Theory:**

A data mining technique that is used to uncover purchase patterns in any retail setting is known as Market Basket Analysis. In simple terms Basically, Market basket analysis in data mining is to analyze the combination of products which have been bought together.

This is a technique that gives the careful study of purchases done by a customer in a supermarket. This concept identifies the pattern of frequent purchase items by customers. This analysis can help to promote deals, offers, sales by the companies, and data mining techniques helps to achieve this analysis task. Example:

* Data mining concepts are in use for Sales and marketing to provide better customer service, to improve cross-selling opportunities, to increase direct mail response rates.
* Customer Retention in the form of pattern identification and prediction of likely defections is possible by Data mining.
* Risk Assessment and Fraud areas also use the data-mining concept for identifying inappropriate or unusual behavior etc.

With the help of the **Apriori Algorithm**, we can further classify and simplify the item sets which are frequently bought by the consumer.

There are three components in APRIORI ALGORITHM:

* SUPPORT
* CONFIDENCE
* LIFT

**Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Oct 18 23:38:09 2022

@author: ARYAN

"""

import numpy as np

import pandas as pd

from apyori import apriori

store\_data=pd.read\_csv('Day 1.csv',header=None)

store\_data

records=[]

for i in range(0,22):

records.append([str(store\_data.values[i,j]) for j in range(0,5)])

a\_rule=apriori(records, min\_supprt=0.3, min\_confidence=0.5, min\_lift=1, min\_length=2)

a\_results=list(a\_rule)

print(len(a\_results))

print(a\_results)

for i in a\_results:

print(i)

print('\n')

**Output:**

runfile('A:/aryan - Copy/aryan - Copy/DWM pracs/prac5.py', wdir='A:/aryan - Copy/aryan - Copy/DWM pracs')

33

[RelationRecord(items=frozenset({'Apple'}), support=0.7272727272727273, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Apple'}), confidence=0.7272727272727273, lift=1.0)]), RelationRecord(items=frozenset({'Bread'}), support=0.7272727272727273, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Bread'}), confidence=0.7272727272727273, lift=1.0)]), RelationRecord(items=frozenset({'Chips'}), support=0.7272727272727273, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Chips'}), confidence=0.7272727272727273, lift=1.0)]), RelationRecord(items=frozenset({'Milk'}), support=0.7727272727272727, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Milk'}), confidence=0.7727272727272727, lift=1.0)]), RelationRecord(items=frozenset({'Wine'}), support=0.7272727272727273, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Wine'}), confidence=0.7272727272727273, lift=1.0)]), RelationRecord(items=frozenset({'nan'}), support=0.7272727272727273, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'nan'}), confidence=0.7272727272727273, lift=1.0)]), RelationRecord(items=frozenset({'Bread', 'Apple'}), support=0.5454545454545454, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.5454545454545454, lift=1.0), OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Bread'}), confidence=0.7499999999999999, lift=1.0312499999999998), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Apple'}), confidence=0.7499999999999999, lift=1.0312499999999998)]), RelationRecord(items=frozenset({'Chips', 'Apple'}), support=0.5454545454545454, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Chips', 'Apple'}), confidence=0.5454545454545454, lift=1.0), OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Chips'}), confidence=0.7499999999999999, lift=1.0312499999999998), OrderedStatistic(items\_base=frozenset({'Chips'}), items\_add=frozenset({'Apple'}), confidence=0.7499999999999999, lift=1.0312499999999998)]), RelationRecord(items=frozenset({'Apple', 'Milk'}), support=0.5454545454545454, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.5454545454545454, lift=1.0)]), RelationRecord(items=frozenset({'Wine', 'Apple'}), support=0.5454545454545454, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.5454545454545454, lift=1.0), OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Wine'}), confidence=0.7499999999999999, lift=1.0312499999999998), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Apple'}), confidence=0.7499999999999999, lift=1.0312499999999998)]), RelationRecord(items=frozenset({'Chips', 'Bread'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5, lift=1.0)]), RelationRecord(items=frozenset({'Bread', 'Milk'}), support=0.5909090909090909, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Bread', 'Milk'}), confidence=0.5909090909090909, lift=1.0), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Milk'}), confidence=0.8125, lift=1.0514705882352942), OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7647058823529412, lift=1.0514705882352942)]), RelationRecord(items=frozenset({'Bread', 'Wine'}), support=0.5909090909090909, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Bread', 'Wine'}), confidence=0.5909090909090909, lift=1.0), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Wine'}), confidence=0.8125, lift=1.1171875), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Bread'}), confidence=0.8125, lift=1.1171875)]), RelationRecord(items=frozenset({'Chips', 'Milk'}), support=0.5454545454545454, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5454545454545454, lift=1.0)]), RelationRecord(items=frozenset({'Wine', 'Milk'}), support=0.6363636363636364, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Wine', 'Milk'}), confidence=0.6363636363636364, lift=1.0), OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.8235294117647058, lift=1.1323529411764706), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Milk'}), confidence=0.875, lift=1.1323529411764706)]), RelationRecord(items=frozenset({'Milk', 'nan'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Milk', 'nan'}), confidence=0.5, lift=1.0)]), RelationRecord(items=frozenset({'Chips', 'Bread', 'Apple'}), support=0.4090909090909091, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5625, lift=1.125), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Chips', 'Apple'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Chips'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Chips'}), confidence=0.7500000000000001, lift=1.0312500000000002), OrderedStatistic(items\_base=frozenset({'Chips', 'Apple'}), items\_add=frozenset({'Bread'}), confidence=0.7500000000000001, lift=1.0312500000000002), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Apple'}), confidence=0.8181818181818182, lift=1.125)]), RelationRecord(items=frozenset({'Bread', 'Apple', 'Milk'}), support=0.4090909090909091, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7500000000000001, lift=1.0312500000000002)]), RelationRecord(items=frozenset({'Bread', 'Wine', 'Apple'}), support=0.45454545454545453, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Bread', 'Wine'}), confidence=0.625, lift=1.0576923076923077), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.625, lift=1.1458333333333335), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.625, lift=1.1458333333333335), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Wine'}), confidence=0.8333333333333334, lift=1.1458333333333333), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Bread'}), confidence=0.8333333333333334, lift=1.1458333333333333), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine'}), items\_add=frozenset({'Apple'}), confidence=0.7692307692307692, lift=1.0576923076923075)]), RelationRecord(items=frozenset({'Chips', 'Apple', 'Milk'}), support=0.4090909090909091, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Chips'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Chips'}), confidence=0.7500000000000001, lift=1.0312500000000002), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Apple'}), confidence=0.7500000000000001, lift=1.0312500000000002)]), RelationRecord(items=frozenset({'Chips', 'Wine', 'Apple'}), support=0.36363636363636365, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5, lift=1.1), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Apple'}), confidence=0.8, lift=1.1)]), RelationRecord(items=frozenset({'Wine', 'Apple', 'Milk'}), support=0.45454545454545453, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.5882352941176471, lift=1.0784313725490198), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.625, lift=1.1458333333333335), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.8333333333333334, lift=1.1458333333333333), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Milk'}), confidence=0.8333333333333334, lift=1.0784313725490198)]), RelationRecord(items=frozenset({'Chips', 'Bread', 'Milk'}), support=0.4090909090909091, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5294117647058824, lift=1.0588235294117647), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Milk'}), confidence=0.8181818181818182, lift=1.0588235294117647), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7500000000000001, lift=1.0312500000000002)]), RelationRecord(items=frozenset({'Chips', 'Bread', 'Wine'}), support=0.36363636363636365, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5, lift=1.1), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Wine'}), confidence=0.7272727272727273, lift=1.0), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Bread'}), confidence=0.8, lift=1.1)]), RelationRecord(items=frozenset({'Bread', 'Wine', 'Milk'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Bread', 'Wine', 'Milk'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Wine', 'Milk'}), confidence=0.6875, lift=1.0803571428571428), OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Bread', 'Wine'}), confidence=0.6470588235294118, lift=1.0950226244343892), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Bread', 'Milk'}), confidence=0.6875, lift=1.1634615384615383), OrderedStatistic(items\_base=frozenset({'Bread', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.8461538461538461, lift=1.1634615384615383), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine'}), items\_add=frozenset({'Milk'}), confidence=0.8461538461538461, lift=1.0950226244343892), OrderedStatistic(items\_base=frozenset({'Wine', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7857142857142857, lift=1.0803571428571428)]), RelationRecord(items=frozenset({'Chips', 'Wine', 'Milk'}), support=0.4090909090909091, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5294117647058824, lift=1.1647058823529413), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.7500000000000001, lift=1.0312500000000002), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Milk'}), confidence=0.9000000000000001, lift=1.1647058823529415)]), RelationRecord(items=frozenset({'Wine', 'Milk', 'nan'}), support=0.36363636363636365, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Milk', 'nan'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Milk', 'nan'}), items\_add=frozenset({'Wine'}), confidence=0.7272727272727273, lift=1.0), OrderedStatistic(items\_base=frozenset({'Wine', 'nan'}), items\_add=frozenset({'Milk'}), confidence=0.8, lift=1.035294117647059)]), RelationRecord(items=frozenset({'Chips', 'Bread', 'Apple', 'Milk'}), support=0.3181818181818182, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5833333333333334, lift=1.0694444444444446), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5833333333333334, lift=1.1666666666666667), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.6363636363636364, lift=1.1666666666666667), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.5833333333333334, lift=1.0694444444444446), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Apple'}), items\_add=frozenset({'Milk'}), confidence=0.7777777777777777, lift=1.0065359477124183), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple', 'Milk'}), items\_add=frozenset({'Chips'}), confidence=0.7777777777777777, lift=1.0694444444444442), OrderedStatistic(items\_base=frozenset({'Chips', 'Apple', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7777777777777777, lift=1.0694444444444442), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Milk'}), items\_add=frozenset({'Apple'}), confidence=0.7777777777777777, lift=1.0694444444444442)]), RelationRecord(items=frozenset({'Chips', 'Bread', 'Wine', 'Apple'}), support=0.3181818181818182, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5833333333333334, lift=1.2833333333333334), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5833333333333334, lift=1.1666666666666667), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.6363636363636364, lift=1.1666666666666667), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.7000000000000001, lift=1.2833333333333337), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Apple'}), items\_add=frozenset({'Wine'}), confidence=0.7777777777777777, lift=1.0694444444444442), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Apple'}), items\_add=frozenset({'Bread'}), confidence=0.875, lift=1.203125), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Wine'}), items\_add=frozenset({'Apple'}), confidence=0.875, lift=1.203125)]), RelationRecord(items=frozenset({'Bread', 'Wine', 'Apple', 'Milk'}), support=0.36363636363636365, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Bread', 'Wine', 'Milk'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Wine', 'Apple', 'Milk'}), confidence=0.5, lift=1.1), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Bread', 'Apple', 'Milk'}), confidence=0.5, lift=1.222222222222222), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Wine', 'Milk'}), confidence=0.6666666666666667, lift=1.0476190476190477), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Bread', 'Wine'}), confidence=0.6666666666666667, lift=1.1282051282051282), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Bread', 'Milk'}), confidence=0.6666666666666667, lift=1.1282051282051282), OrderedStatistic(items\_base=frozenset({'Bread', 'Milk'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.6153846153846154, lift=1.1282051282051284), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.6153846153846154, lift=1.1282051282051284), OrderedStatistic(items\_base=frozenset({'Wine', 'Milk'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.5714285714285715, lift=1.047619047619048), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.8888888888888888, lift=1.222222222222222), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine', 'Apple'}), items\_add=frozenset({'Milk'}), confidence=0.8, lift=1.035294117647059), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.8, lift=1.1), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine', 'Milk'}), items\_add=frozenset({'Apple'}), confidence=0.7272727272727273, lift=1.0)]), RelationRecord(items=frozenset({'Chips', 'Wine', 'Apple', 'Milk'}), support=0.3181818181818182, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5833333333333334, lift=1.2833333333333334), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5833333333333334, lift=1.0694444444444446), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.5833333333333334, lift=1.0694444444444446), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.7000000000000001, lift=1.2833333333333337), OrderedStatistic(items\_base=frozenset({'Chips', 'Apple', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.7777777777777777, lift=1.0694444444444442), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Apple'}), items\_add=frozenset({'Milk'}), confidence=0.875, lift=1.1323529411764706), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Milk'}), items\_add=frozenset({'Apple'}), confidence=0.7777777777777777, lift=1.0694444444444442)]), RelationRecord(items=frozenset({'Chips', 'Bread', 'Wine', 'Milk'}), support=0.3181818181818182, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Wine', 'Milk'}), confidence=0.6363636363636364, lift=1.0), OrderedStatistic(items\_base=frozenset({'Bread', 'Milk'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5384615384615384, lift=1.1846153846153846), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Bread', 'Milk'}), confidence=0.7000000000000001, lift=1.1846153846153846), OrderedStatistic(items\_base=frozenset({'Wine', 'Milk'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.7777777777777777, lift=1.0694444444444442), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Wine'}), items\_add=frozenset({'Milk'}), confidence=0.875, lift=1.1323529411764706), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7777777777777777, lift=1.0694444444444442)]), RelationRecord(items=frozenset({'Chips', 'Wine', 'Milk', 'Bread', 'Apple'}), support=0.2727272727272727, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Chips', 'Wine', 'Milk'}), confidence=0.5, lift=1.222222222222222), OrderedStatistic(items\_base=frozenset({'Chips', 'Apple'}), items\_add=frozenset({'Bread', 'Wine', 'Milk'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Chips', 'Wine', 'Bread'}), confidence=0.5, lift=1.375), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Bread', 'Chips', 'Milk'}), confidence=0.5, lift=1.222222222222222), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Wine', 'Apple', 'Milk'}), confidence=0.5454545454545454, lift=1.2), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Bread', 'Wine', 'Apple'}), confidence=0.5, lift=1.1), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Bread', 'Apple', 'Milk'}), confidence=0.6, lift=1.4666666666666666), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Apple'}), items\_add=frozenset({'Wine', 'Milk'}), confidence=0.6666666666666666, lift=1.0476190476190477), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple', 'Milk'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.6666666666666666, lift=1.4666666666666666), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine', 'Apple'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.6, lift=1.1), OrderedStatistic(items\_base=frozenset({'Chips', 'Apple', 'Milk'}), items\_add=frozenset({'Bread', 'Wine'}), confidence=0.6666666666666666, lift=1.1282051282051282), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Apple'}), items\_add=frozenset({'Bread', 'Milk'}), confidence=0.7499999999999999, lift=1.269230769230769), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple', 'Milk'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.6, lift=1.2), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Milk'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.6666666666666666, lift=1.2222222222222223), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Wine'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.7499999999999999, lift=1.375), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine', 'Milk'}), items\_add=frozenset({'Chips', 'Apple'}), confidence=0.5454545454545454, lift=1.0), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Milk'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.6666666666666666, lift=1.2222222222222223), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Apple', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.8571428571428571, lift=1.1785714285714284), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Wine', 'Apple'}), items\_add=frozenset({'Milk'}), confidence=0.8571428571428571, lift=1.1092436974789917), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine', 'Apple', 'Milk'}), items\_add=frozenset({'Chips'}), confidence=0.7499999999999999, lift=1.0312499999999998), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Apple', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.8571428571428571, lift=1.1785714285714284), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Wine', 'Milk'}), items\_add=frozenset({'Apple'}), confidence=0.8571428571428571, lift=1.1785714285714284)])]

RelationRecord(items=frozenset({'Apple'}), support=0.7272727272727273, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Apple'}), confidence=0.7272727272727273, lift=1.0)])

RelationRecord(items=frozenset({'Bread'}), support=0.7272727272727273, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Bread'}), confidence=0.7272727272727273, lift=1.0)])

RelationRecord(items=frozenset({'Chips'}), support=0.7272727272727273, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Chips'}), confidence=0.7272727272727273, lift=1.0)])

RelationRecord(items=frozenset({'Milk'}), support=0.7727272727272727, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Milk'}), confidence=0.7727272727272727, lift=1.0)])

RelationRecord(items=frozenset({'Wine'}), support=0.7272727272727273, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Wine'}), confidence=0.7272727272727273, lift=1.0)])

RelationRecord(items=frozenset({'nan'}), support=0.7272727272727273, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'nan'}), confidence=0.7272727272727273, lift=1.0)])

RelationRecord(items=frozenset({'Bread', 'Apple'}), support=0.5454545454545454, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.5454545454545454, lift=1.0), OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Bread'}), confidence=0.7499999999999999, lift=1.0312499999999998), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Apple'}), confidence=0.7499999999999999, lift=1.0312499999999998)])

RelationRecord(items=frozenset({'Chips', 'Apple'}), support=0.5454545454545454, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Chips', 'Apple'}), confidence=0.5454545454545454, lift=1.0), OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Chips'}), confidence=0.7499999999999999, lift=1.0312499999999998), OrderedStatistic(items\_base=frozenset({'Chips'}), items\_add=frozenset({'Apple'}), confidence=0.7499999999999999, lift=1.0312499999999998)])

RelationRecord(items=frozenset({'Apple', 'Milk'}), support=0.5454545454545454, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.5454545454545454, lift=1.0)])

RelationRecord(items=frozenset({'Wine', 'Apple'}), support=0.5454545454545454, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.5454545454545454, lift=1.0), OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Wine'}), confidence=0.7499999999999999, lift=1.0312499999999998), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Apple'}), confidence=0.7499999999999999, lift=1.0312499999999998)])

RelationRecord(items=frozenset({'Chips', 'Bread'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5, lift=1.0)])

RelationRecord(items=frozenset({'Bread', 'Milk'}), support=0.5909090909090909, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Bread', 'Milk'}), confidence=0.5909090909090909, lift=1.0), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Milk'}), confidence=0.8125, lift=1.0514705882352942), OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7647058823529412, lift=1.0514705882352942)])

RelationRecord(items=frozenset({'Bread', 'Wine'}), support=0.5909090909090909, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Bread', 'Wine'}), confidence=0.5909090909090909, lift=1.0), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Wine'}), confidence=0.8125, lift=1.1171875), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Bread'}), confidence=0.8125, lift=1.1171875)])

RelationRecord(items=frozenset({'Chips', 'Milk'}), support=0.5454545454545454, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5454545454545454, lift=1.0)])

RelationRecord(items=frozenset({'Wine', 'Milk'}), support=0.6363636363636364, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Wine', 'Milk'}), confidence=0.6363636363636364, lift=1.0), OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.8235294117647058, lift=1.1323529411764706), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Milk'}), confidence=0.875, lift=1.1323529411764706)])

RelationRecord(items=frozenset({'Milk', 'nan'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Milk', 'nan'}), confidence=0.5, lift=1.0)])

RelationRecord(items=frozenset({'Chips', 'Bread', 'Apple'}), support=0.4090909090909091, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5625, lift=1.125), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Chips', 'Apple'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Chips'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Chips'}), confidence=0.7500000000000001, lift=1.0312500000000002), OrderedStatistic(items\_base=frozenset({'Chips', 'Apple'}), items\_add=frozenset({'Bread'}), confidence=0.7500000000000001, lift=1.0312500000000002), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Apple'}), confidence=0.8181818181818182, lift=1.125)])

RelationRecord(items=frozenset({'Bread', 'Apple', 'Milk'}), support=0.4090909090909091, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7500000000000001, lift=1.0312500000000002)])

RelationRecord(items=frozenset({'Bread', 'Wine', 'Apple'}), support=0.45454545454545453, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Bread', 'Wine'}), confidence=0.625, lift=1.0576923076923077), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.625, lift=1.1458333333333335), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.625, lift=1.1458333333333335), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Wine'}), confidence=0.8333333333333334, lift=1.1458333333333333), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Bread'}), confidence=0.8333333333333334, lift=1.1458333333333333), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine'}), items\_add=frozenset({'Apple'}), confidence=0.7692307692307692, lift=1.0576923076923075)])

RelationRecord(items=frozenset({'Chips', 'Apple', 'Milk'}), support=0.4090909090909091, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Chips'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Chips'}), confidence=0.7500000000000001, lift=1.0312500000000002), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Apple'}), confidence=0.7500000000000001, lift=1.0312500000000002)])

RelationRecord(items=frozenset({'Chips', 'Wine', 'Apple'}), support=0.36363636363636365, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5, lift=1.1), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Apple'}), confidence=0.8, lift=1.1)])

RelationRecord(items=frozenset({'Wine', 'Apple', 'Milk'}), support=0.45454545454545453, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.5882352941176471, lift=1.0784313725490198), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.625, lift=1.1458333333333335), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.8333333333333334, lift=1.1458333333333333), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Milk'}), confidence=0.8333333333333334, lift=1.0784313725490198)])

RelationRecord(items=frozenset({'Chips', 'Bread', 'Milk'}), support=0.4090909090909091, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5294117647058824, lift=1.0588235294117647), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Milk'}), confidence=0.8181818181818182, lift=1.0588235294117647), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7500000000000001, lift=1.0312500000000002)])

RelationRecord(items=frozenset({'Chips', 'Bread', 'Wine'}), support=0.36363636363636365, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5, lift=1.1), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Wine'}), confidence=0.7272727272727273, lift=1.0), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Bread'}), confidence=0.8, lift=1.1)])

RelationRecord(items=frozenset({'Bread', 'Wine', 'Milk'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_add=frozenset({'Bread', 'Wine', 'Milk'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Wine', 'Milk'}), confidence=0.6875, lift=1.0803571428571428), OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Bread', 'Wine'}), confidence=0.6470588235294118, lift=1.0950226244343892), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Bread', 'Milk'}), confidence=0.6875, lift=1.1634615384615383), OrderedStatistic(items\_base=frozenset({'Bread', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.8461538461538461, lift=1.1634615384615383), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine'}), items\_add=frozenset({'Milk'}), confidence=0.8461538461538461, lift=1.0950226244343892), OrderedStatistic(items\_base=frozenset({'Wine', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7857142857142857, lift=1.0803571428571428)])

RelationRecord(items=frozenset({'Chips', 'Wine', 'Milk'}), support=0.4090909090909091, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Milk'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5294117647058824, lift=1.1647058823529413), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5625, lift=1.03125), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.7500000000000001, lift=1.0312500000000002), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Milk'}), confidence=0.9000000000000001, lift=1.1647058823529415)])

RelationRecord(items=frozenset({'Wine', 'Milk', 'nan'}), support=0.36363636363636365, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Milk', 'nan'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Milk', 'nan'}), items\_add=frozenset({'Wine'}), confidence=0.7272727272727273, lift=1.0), OrderedStatistic(items\_base=frozenset({'Wine', 'nan'}), items\_add=frozenset({'Milk'}), confidence=0.8, lift=1.035294117647059)])

RelationRecord(items=frozenset({'Chips', 'Bread', 'Apple', 'Milk'}), support=0.3181818181818182, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5833333333333334, lift=1.0694444444444446), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5833333333333334, lift=1.1666666666666667), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.6363636363636364, lift=1.1666666666666667), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.5833333333333334, lift=1.0694444444444446), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Apple'}), items\_add=frozenset({'Milk'}), confidence=0.7777777777777777, lift=1.0065359477124183), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple', 'Milk'}), items\_add=frozenset({'Chips'}), confidence=0.7777777777777777, lift=1.0694444444444442), OrderedStatistic(items\_base=frozenset({'Chips', 'Apple', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7777777777777777, lift=1.0694444444444442), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Milk'}), items\_add=frozenset({'Apple'}), confidence=0.7777777777777777, lift=1.0694444444444442)])

RelationRecord(items=frozenset({'Chips', 'Bread', 'Wine', 'Apple'}), support=0.3181818181818182, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5833333333333334, lift=1.2833333333333334), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5833333333333334, lift=1.1666666666666667), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.6363636363636364, lift=1.1666666666666667), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.7000000000000001, lift=1.2833333333333337), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Apple'}), items\_add=frozenset({'Wine'}), confidence=0.7777777777777777, lift=1.0694444444444442), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Apple'}), items\_add=frozenset({'Bread'}), confidence=0.875, lift=1.203125), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Wine'}), items\_add=frozenset({'Apple'}), confidence=0.875, lift=1.203125)])

RelationRecord(items=frozenset({'Bread', 'Wine', 'Apple', 'Milk'}), support=0.36363636363636365, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple'}), items\_add=frozenset({'Bread', 'Wine', 'Milk'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Bread'}), items\_add=frozenset({'Wine', 'Apple', 'Milk'}), confidence=0.5, lift=1.1), OrderedStatistic(items\_base=frozenset({'Wine'}), items\_add=frozenset({'Bread', 'Apple', 'Milk'}), confidence=0.5, lift=1.222222222222222), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Wine', 'Milk'}), confidence=0.6666666666666667, lift=1.0476190476190477), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Bread', 'Wine'}), confidence=0.6666666666666667, lift=1.1282051282051282), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Bread', 'Milk'}), confidence=0.6666666666666667, lift=1.1282051282051282), OrderedStatistic(items\_base=frozenset({'Bread', 'Milk'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.6153846153846154, lift=1.1282051282051284), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.6153846153846154, lift=1.1282051282051284), OrderedStatistic(items\_base=frozenset({'Wine', 'Milk'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.5714285714285715, lift=1.047619047619048), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.8888888888888888, lift=1.222222222222222), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine', 'Apple'}), items\_add=frozenset({'Milk'}), confidence=0.8, lift=1.035294117647059), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.8, lift=1.1), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine', 'Milk'}), items\_add=frozenset({'Apple'}), confidence=0.7272727272727273, lift=1.0)])

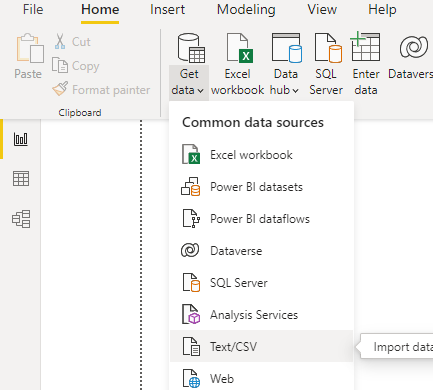
RelationRecord(items=frozenset({'Chips', 'Wine', 'Apple', 'Milk'}), support=0.3181818181818182, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5833333333333334, lift=1.2833333333333334), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.5833333333333334, lift=1.0694444444444446), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.5833333333333334, lift=1.0694444444444446), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.7000000000000001, lift=1.2833333333333337), OrderedStatistic(items\_base=frozenset({'Chips', 'Apple', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.7777777777777777, lift=1.0694444444444442), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Apple'}), items\_add=frozenset({'Milk'}), confidence=0.875, lift=1.1323529411764706), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Milk'}), items\_add=frozenset({'Apple'}), confidence=0.7777777777777777, lift=1.0694444444444442)])

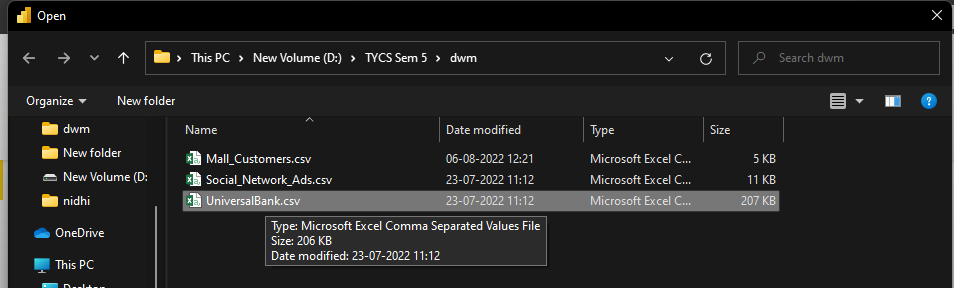
RelationRecord(items=frozenset({'Chips', 'Bread', 'Wine', 'Milk'}), support=0.3181818181818182, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Wine', 'Milk'}), confidence=0.6363636363636364, lift=1.0), OrderedStatistic(items\_base=frozenset({'Bread', 'Milk'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.5384615384615384, lift=1.1846153846153846), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Bread', 'Milk'}), confidence=0.7000000000000001, lift=1.1846153846153846), OrderedStatistic(items\_base=frozenset({'Wine', 'Milk'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.7777777777777777, lift=1.0694444444444442), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Wine'}), items\_add=frozenset({'Milk'}), confidence=0.875, lift=1.1323529411764706), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.7777777777777777, lift=1.0694444444444442)])

RelationRecord(items=frozenset({'Chips', 'Wine', 'Milk', 'Bread', 'Apple'}), support=0.2727272727272727, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Bread', 'Apple'}), items\_add=frozenset({'Chips', 'Wine', 'Milk'}), confidence=0.5, lift=1.222222222222222), OrderedStatistic(items\_base=frozenset({'Chips', 'Apple'}), items\_add=frozenset({'Bread', 'Wine', 'Milk'}), confidence=0.5, lift=1.0), OrderedStatistic(items\_base=frozenset({'Apple', 'Milk'}), items\_add=frozenset({'Chips', 'Wine', 'Bread'}), confidence=0.5, lift=1.375), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple'}), items\_add=frozenset({'Bread', 'Chips', 'Milk'}), confidence=0.5, lift=1.222222222222222), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread'}), items\_add=frozenset({'Wine', 'Apple', 'Milk'}), confidence=0.5454545454545454, lift=1.2), OrderedStatistic(items\_base=frozenset({'Chips', 'Milk'}), items\_add=frozenset({'Bread', 'Wine', 'Apple'}), confidence=0.5, lift=1.1), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine'}), items\_add=frozenset({'Bread', 'Apple', 'Milk'}), confidence=0.6, lift=1.4666666666666666), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Apple'}), items\_add=frozenset({'Wine', 'Milk'}), confidence=0.6666666666666666, lift=1.0476190476190477), OrderedStatistic(items\_base=frozenset({'Bread', 'Apple', 'Milk'}), items\_add=frozenset({'Chips', 'Wine'}), confidence=0.6666666666666666, lift=1.4666666666666666), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine', 'Apple'}), items\_add=frozenset({'Chips', 'Milk'}), confidence=0.6, lift=1.1), OrderedStatistic(items\_base=frozenset({'Chips', 'Apple', 'Milk'}), items\_add=frozenset({'Bread', 'Wine'}), confidence=0.6666666666666666, lift=1.1282051282051282), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Apple'}), items\_add=frozenset({'Bread', 'Milk'}), confidence=0.7499999999999999, lift=1.269230769230769), OrderedStatistic(items\_base=frozenset({'Wine', 'Apple', 'Milk'}), items\_add=frozenset({'Chips', 'Bread'}), confidence=0.6, lift=1.2), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Milk'}), items\_add=frozenset({'Wine', 'Apple'}), confidence=0.6666666666666666, lift=1.2222222222222223), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Wine'}), items\_add=frozenset({'Apple', 'Milk'}), confidence=0.7499999999999999, lift=1.375), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine', 'Milk'}), items\_add=frozenset({'Chips', 'Apple'}), confidence=0.5454545454545454, lift=1.0), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Milk'}), items\_add=frozenset({'Bread', 'Apple'}), confidence=0.6666666666666666, lift=1.2222222222222223), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Apple', 'Milk'}), items\_add=frozenset({'Wine'}), confidence=0.8571428571428571, lift=1.1785714285714284), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Wine', 'Apple'}), items\_add=frozenset({'Milk'}), confidence=0.8571428571428571, lift=1.1092436974789917), OrderedStatistic(items\_base=frozenset({'Bread', 'Wine', 'Apple', 'Milk'}), items\_add=frozenset({'Chips'}), confidence=0.7499999999999999, lift=1.0312499999999998), OrderedStatistic(items\_base=frozenset({'Chips', 'Wine', 'Apple', 'Milk'}), items\_add=frozenset({'Bread'}), confidence=0.8571428571428571, lift=1.1785714285714284), OrderedStatistic(items\_base=frozenset({'Chips', 'Bread', 'Wine', 'Milk'}), items\_add=frozenset({'Apple'}), confidence=0.8571428571428571, lift=1.1785714285714284)])

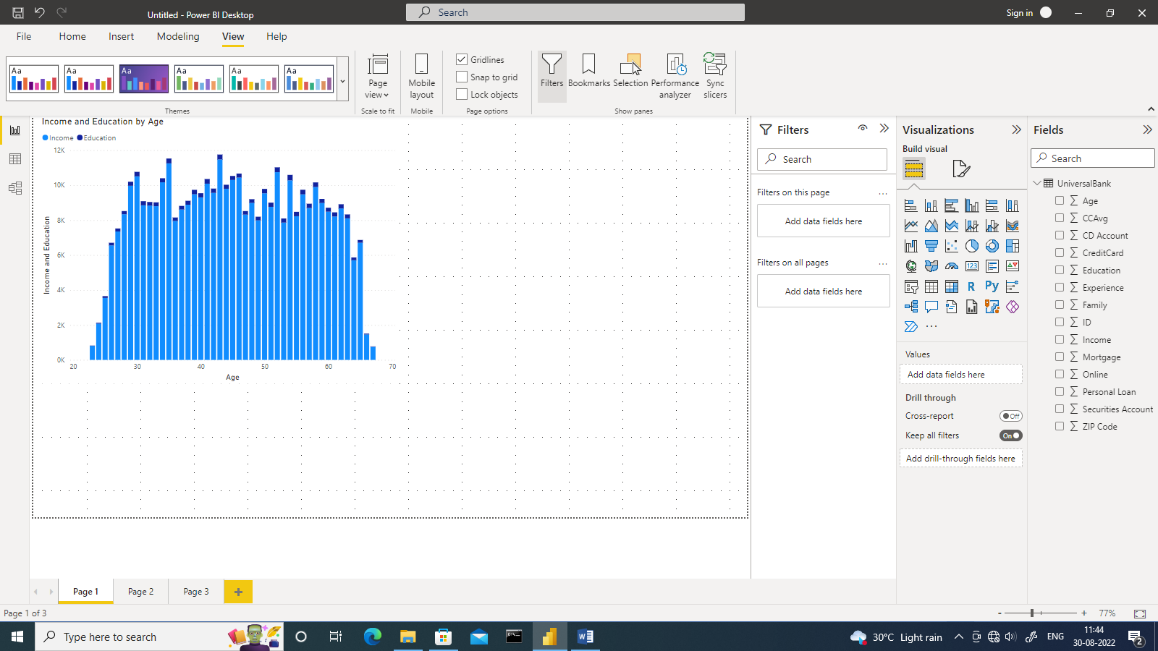
|  |  |
| --- | --- |
| **Practical 6** | |
| **Aim**: Power Bi Demonstration - I | |
| Name: Aryan Penikal | Roll No: KCTBCS050 |
| Performance date: | Sign: |

1. Import the Universal bank dataset.

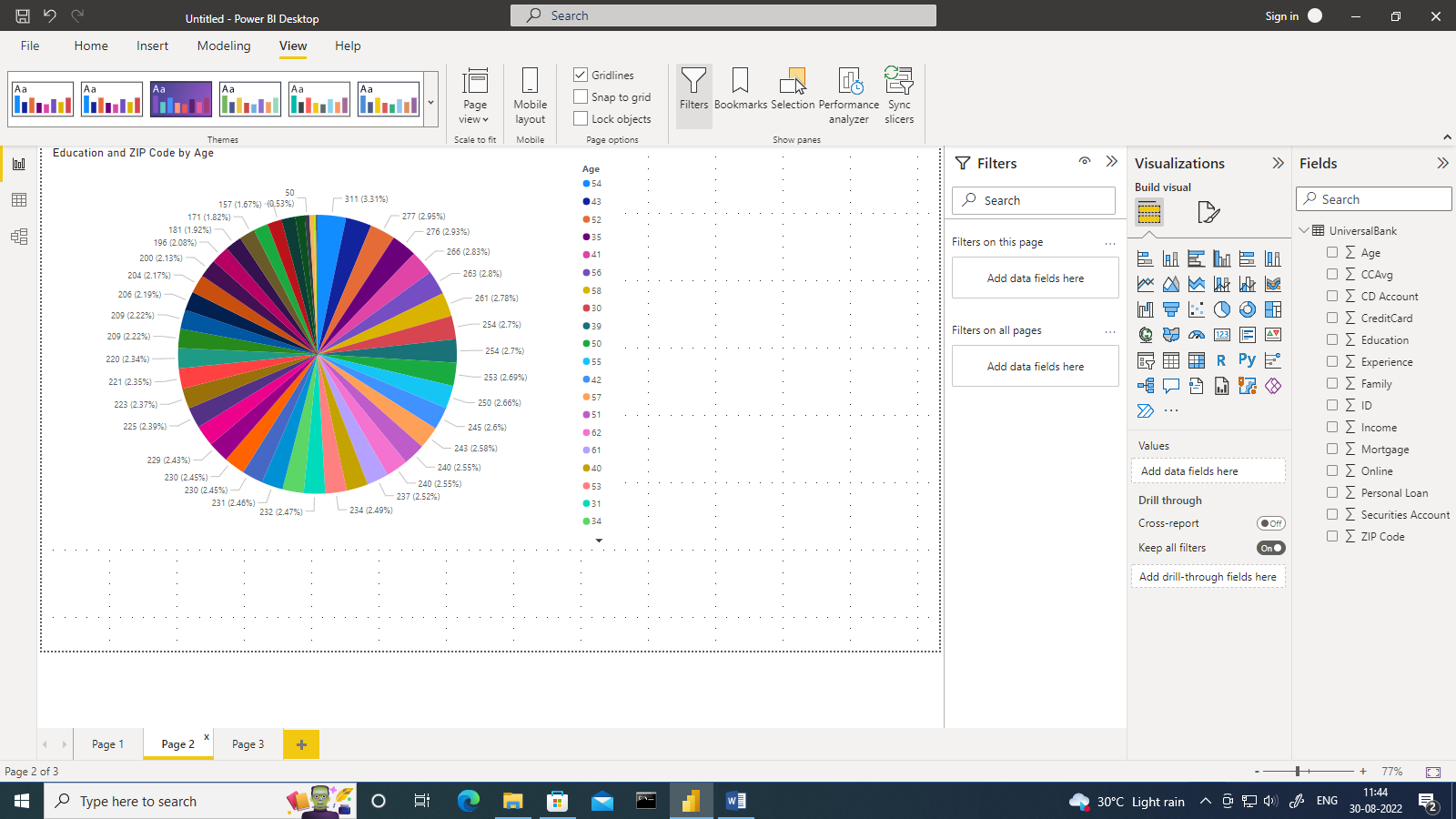
****

****

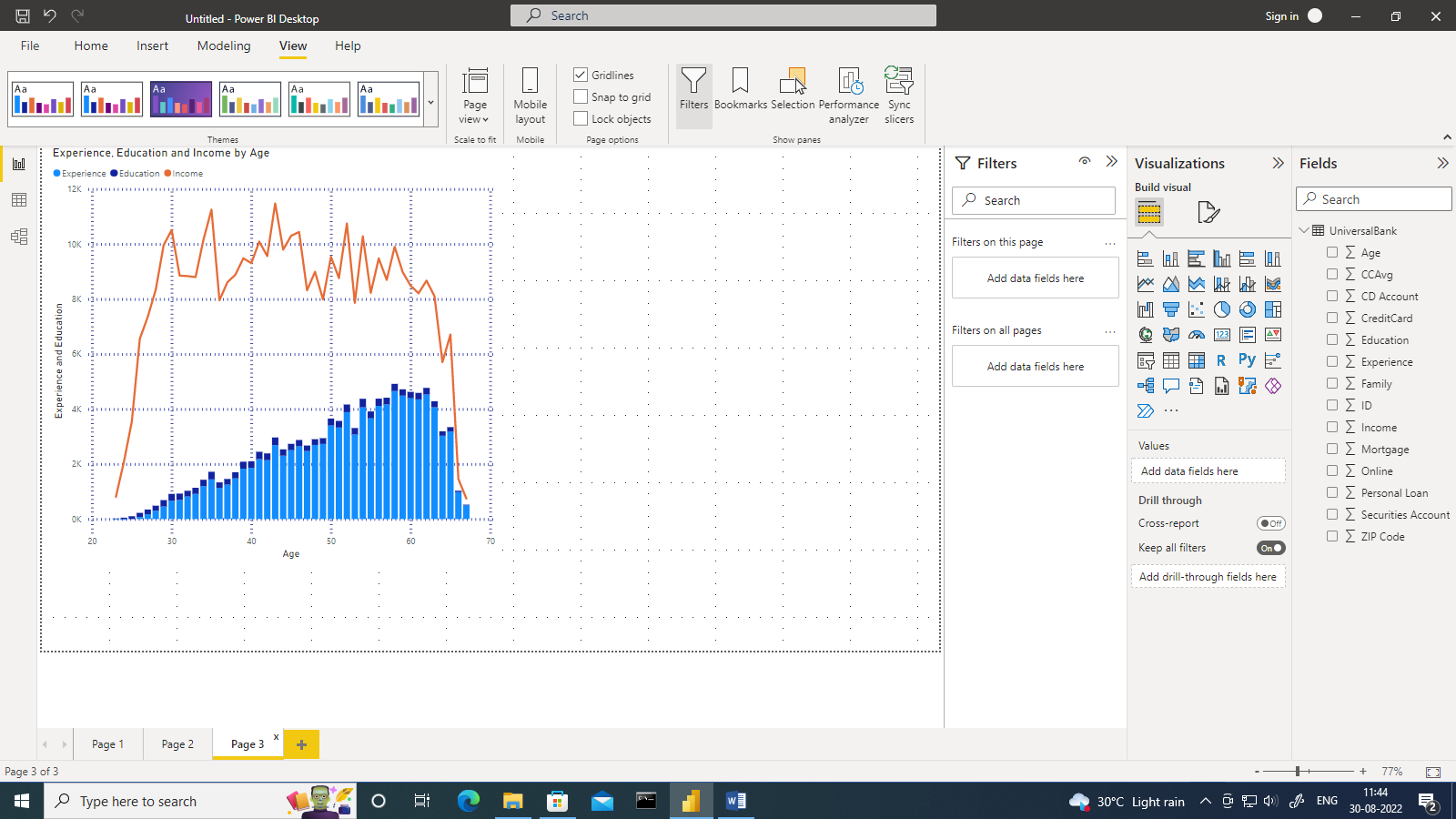
1. Plot a stacked column chart of Income and education w.r.t. age.



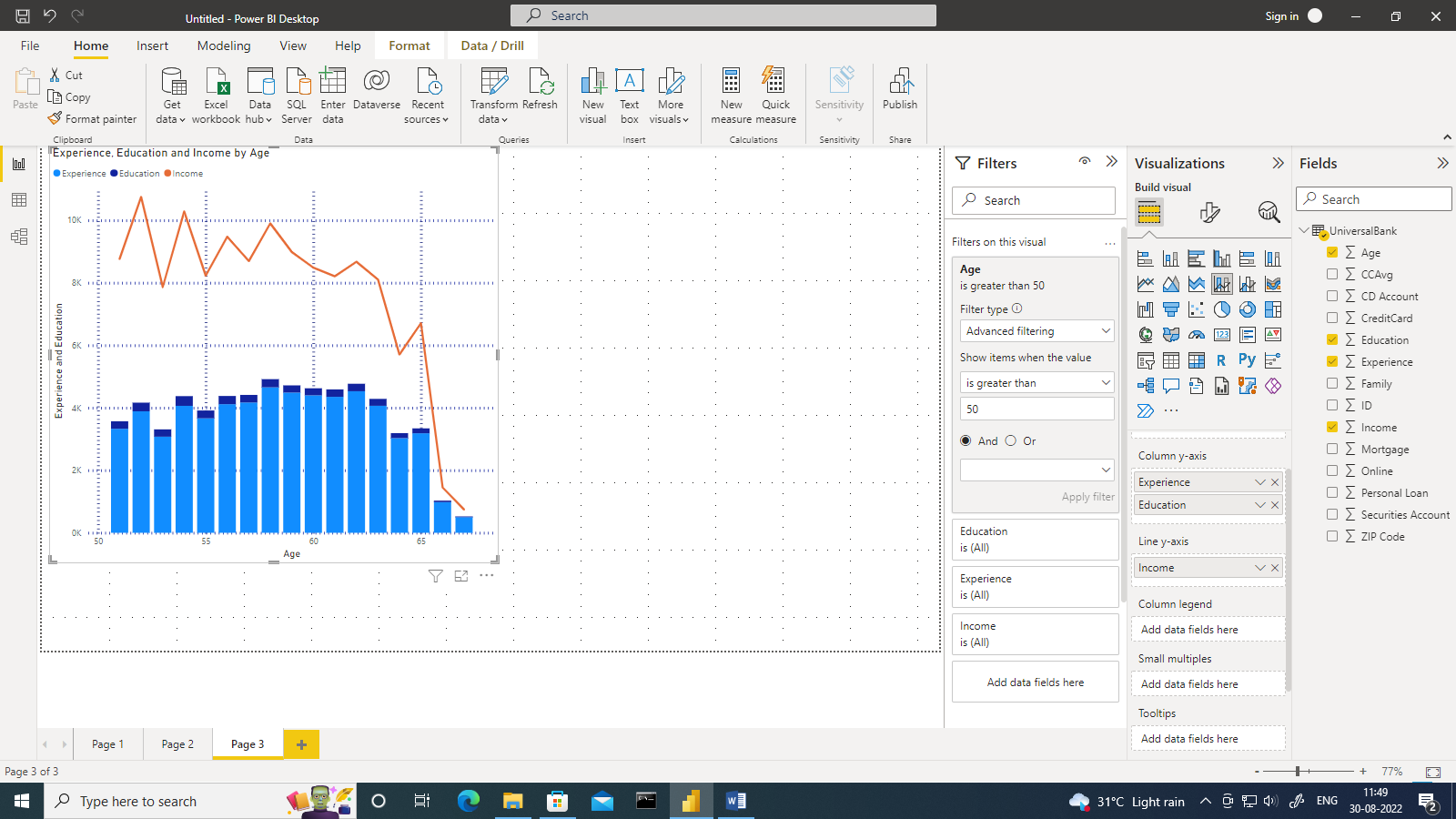
1. Plot the pie chart of education qualification with age. Provide the zip code details using tool tip.



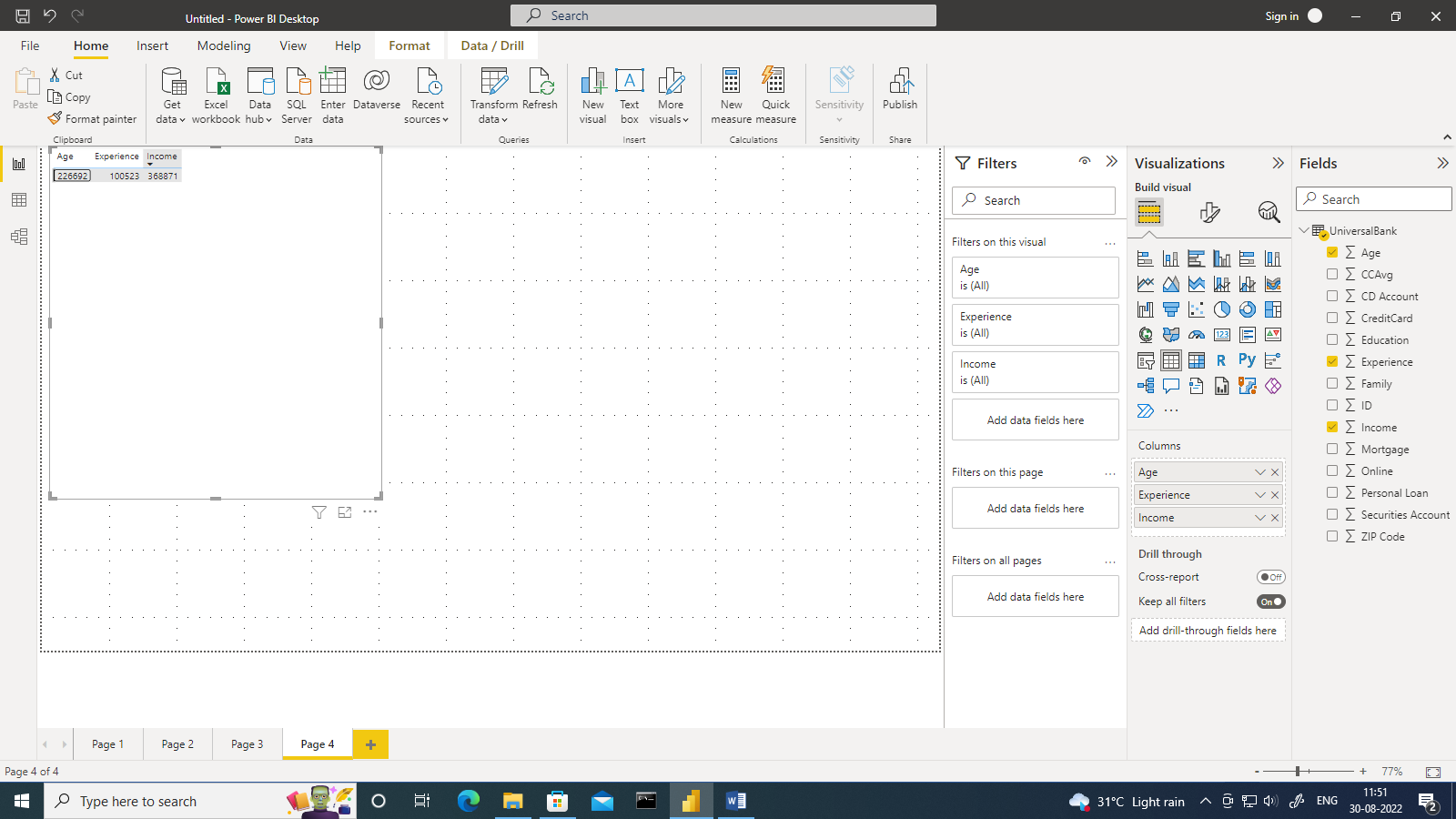
1. Plot the line and column stacked chart showing age and experience and education in column chart and income on line chart.



1. Modify the above column stacked chart for age group greater than 50.



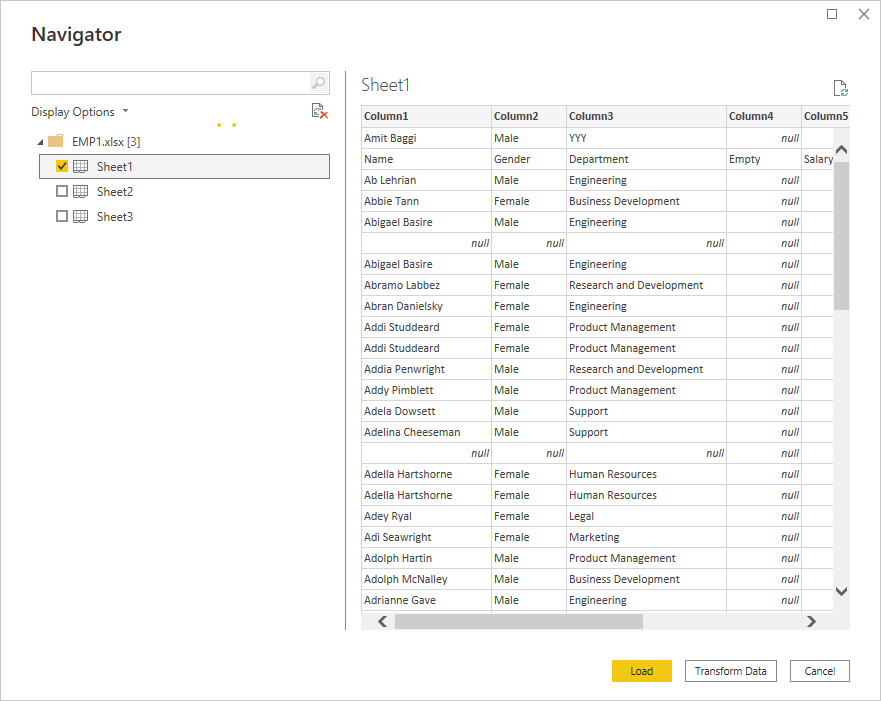
1. View age, experience, income data in tabular format using table.



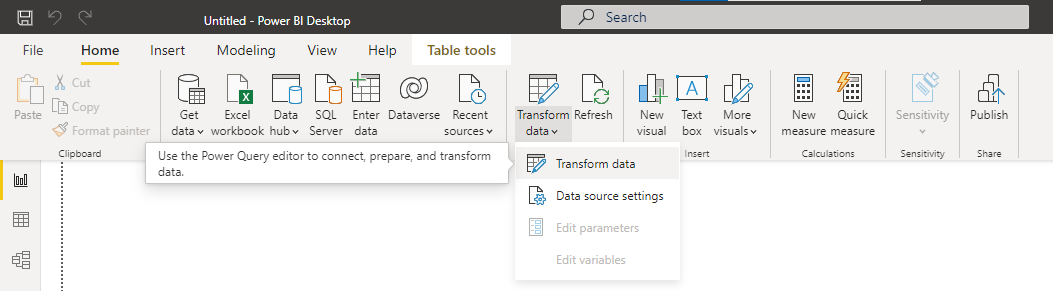
|  |  |
| --- | --- |
| **Practical 7** | |
| **Aim**: Perform the Extraction Transformation and Loading (ETL) process to construct the database using Power Bi. | |
| Name: Aryan Penikal | Roll No: KCTBCS050 |
| Performance date: | Sign: |

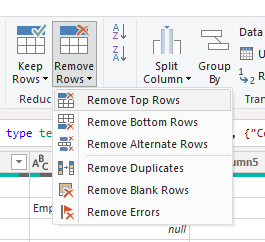
Perform the Extraction Transformation and Loading (ETL) process to construct the database

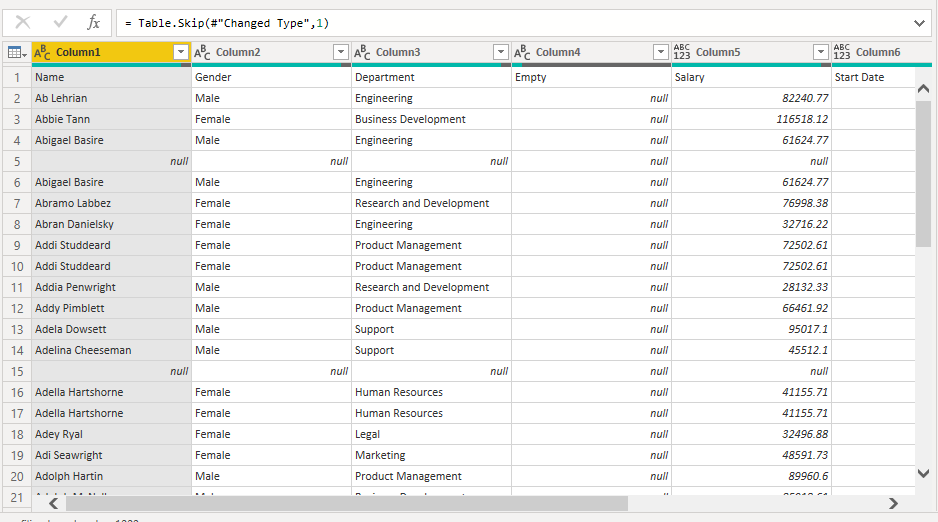
1. Load the Emp1.xlsx



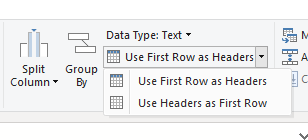
1. Remove the first row

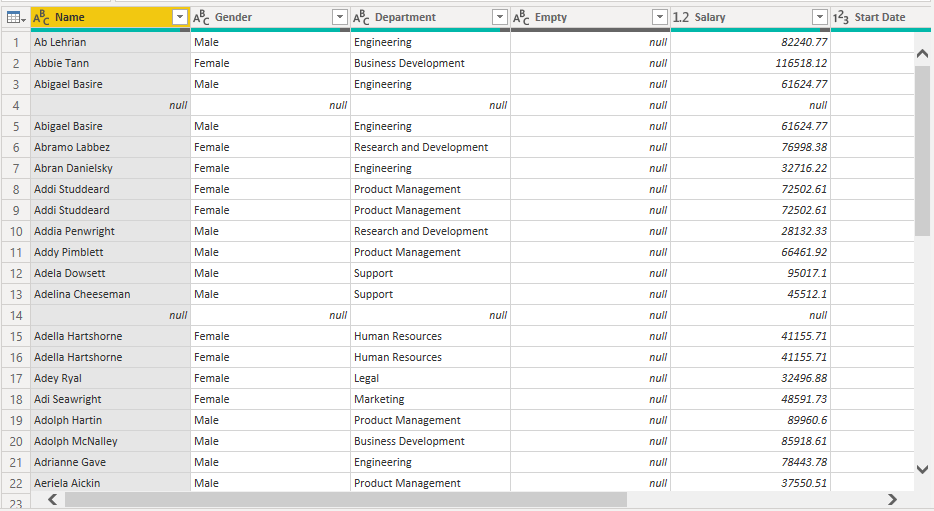




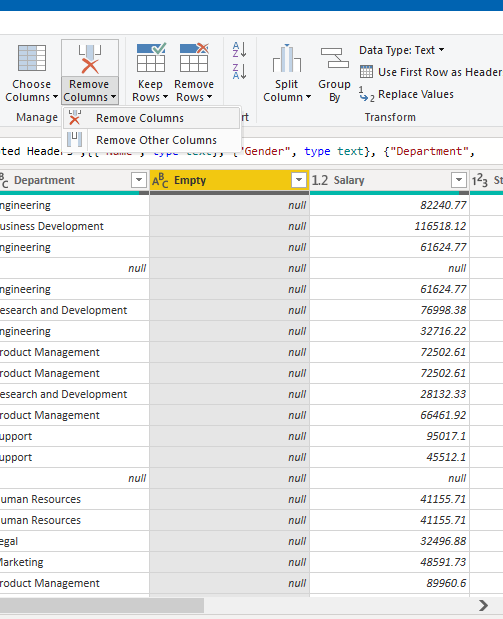


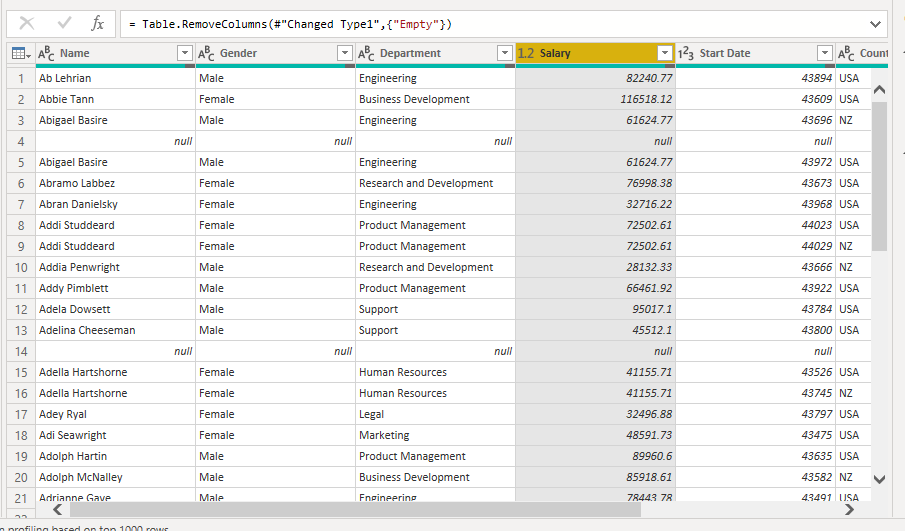
1. Promote the first row as header.



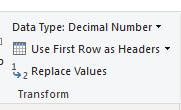


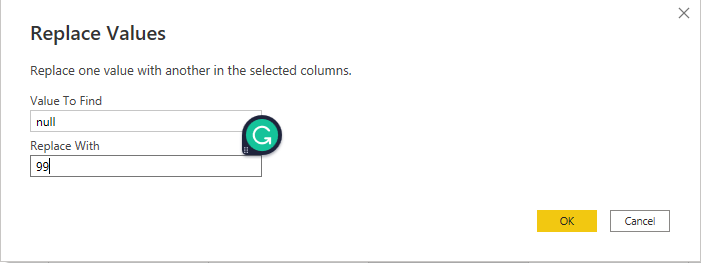
1. Remove the null column.

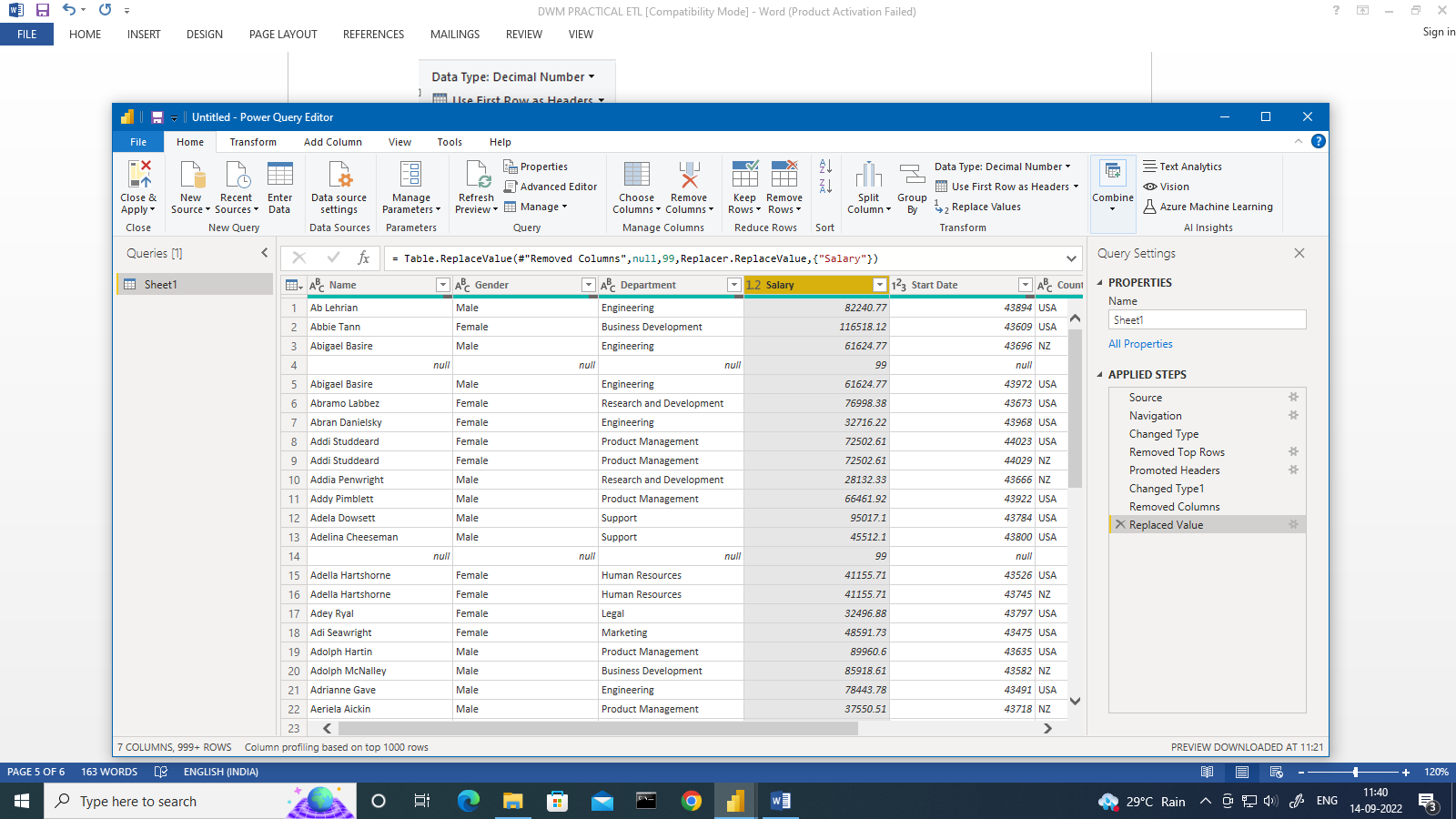




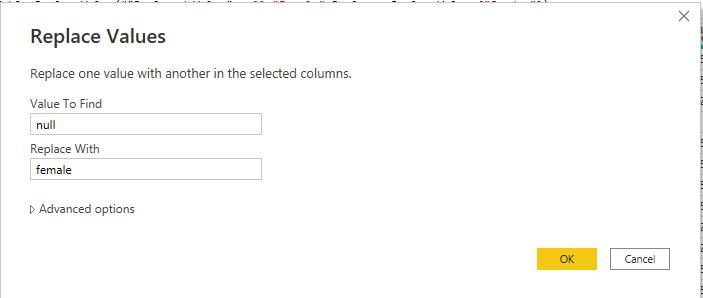
1. Replace the missing values in salary column with 99.

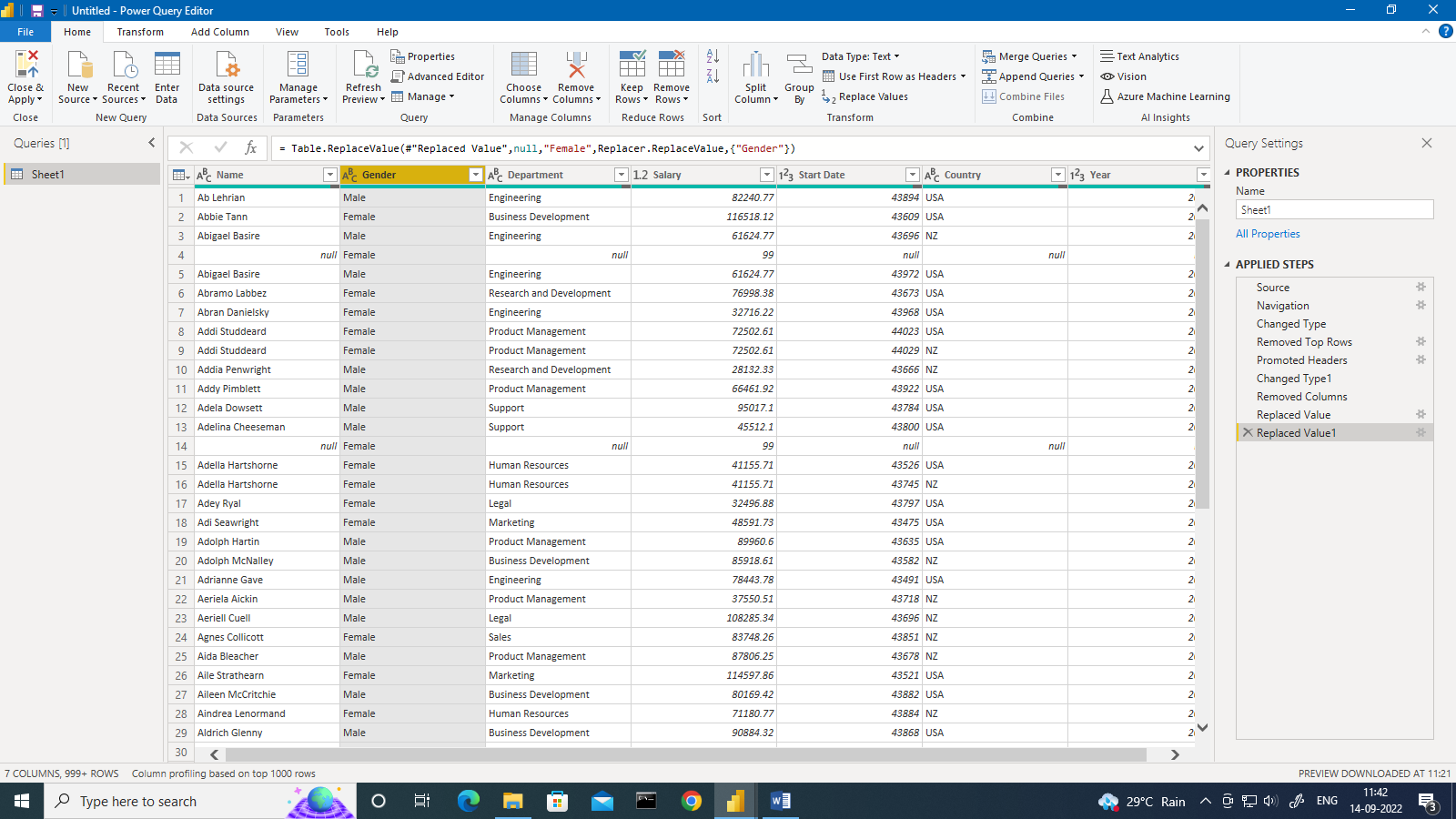




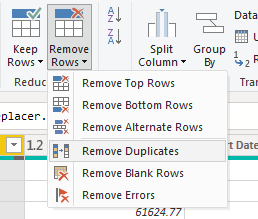


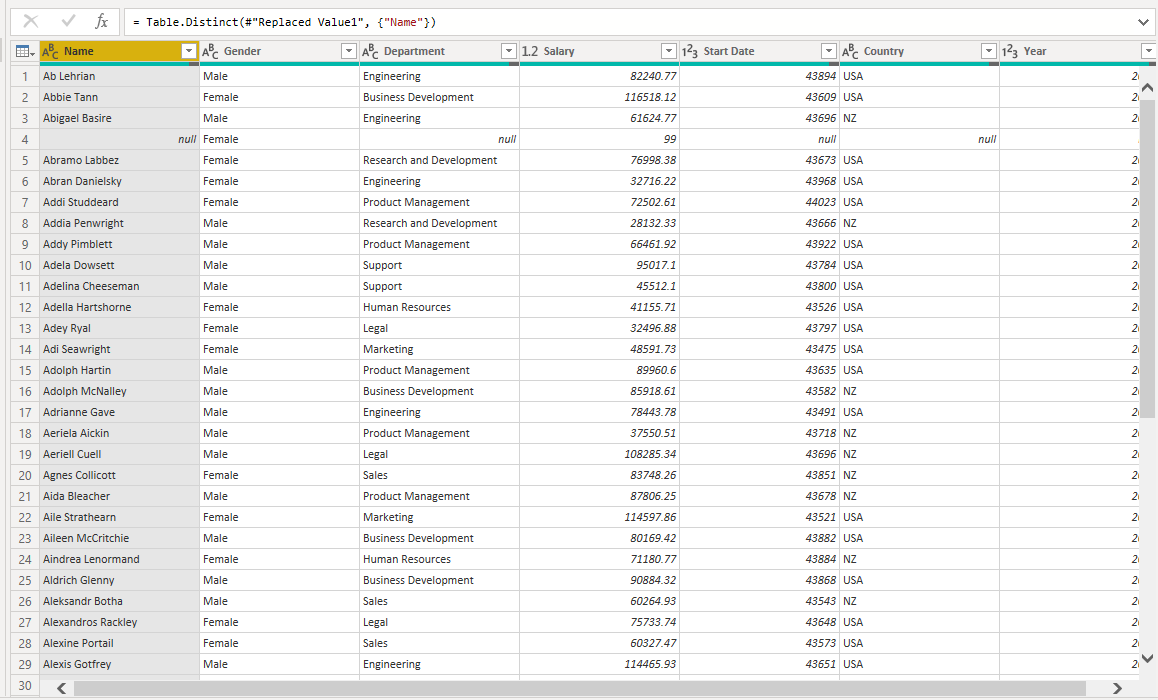
1. Replace the empty spaces in Gender column with Female.



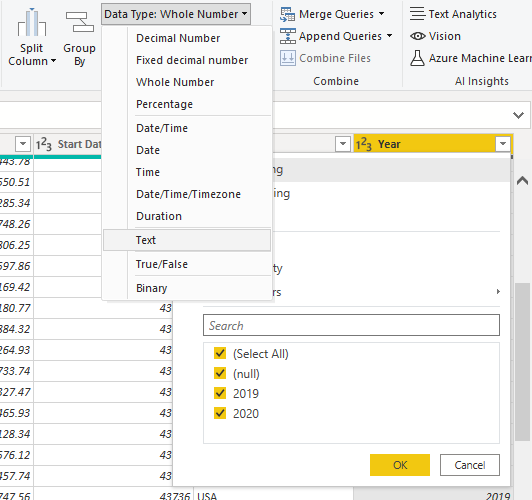


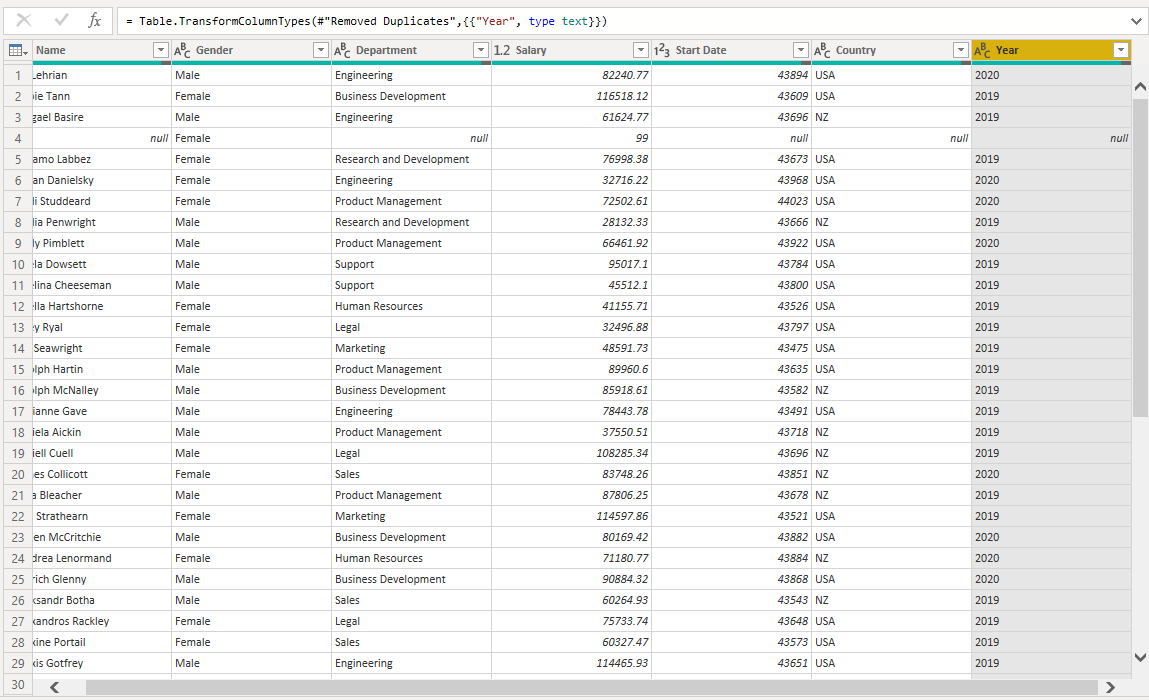
1. Remove duplicate data in Name.



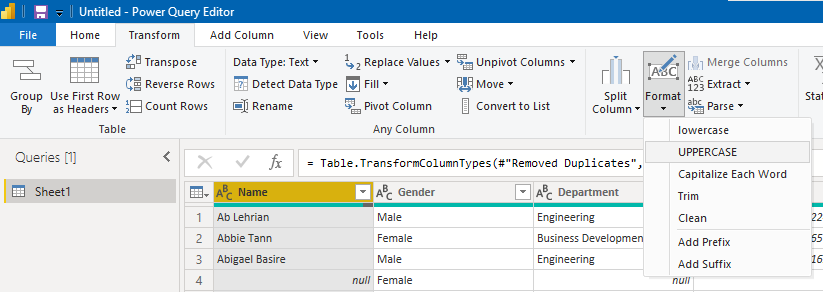


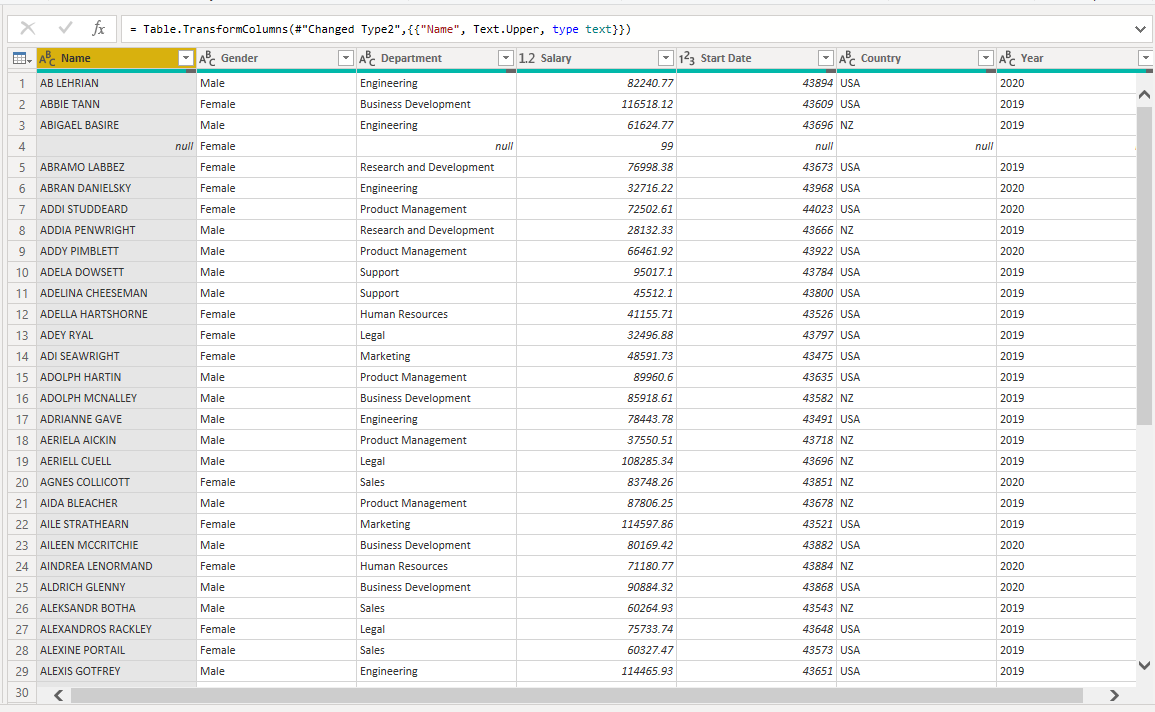
1. Change the data type of Year column as text.



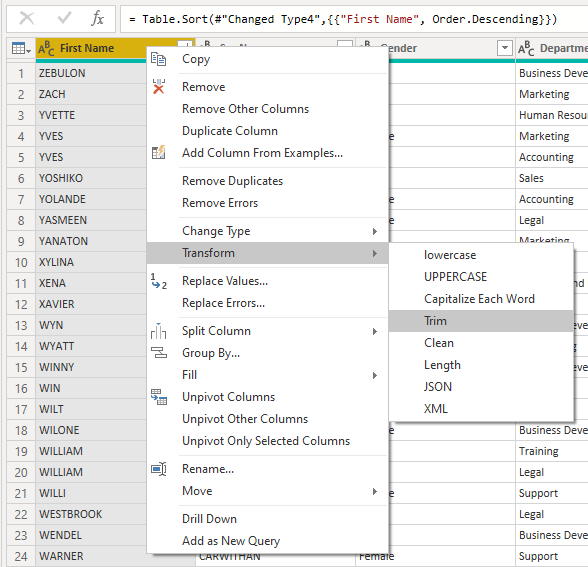


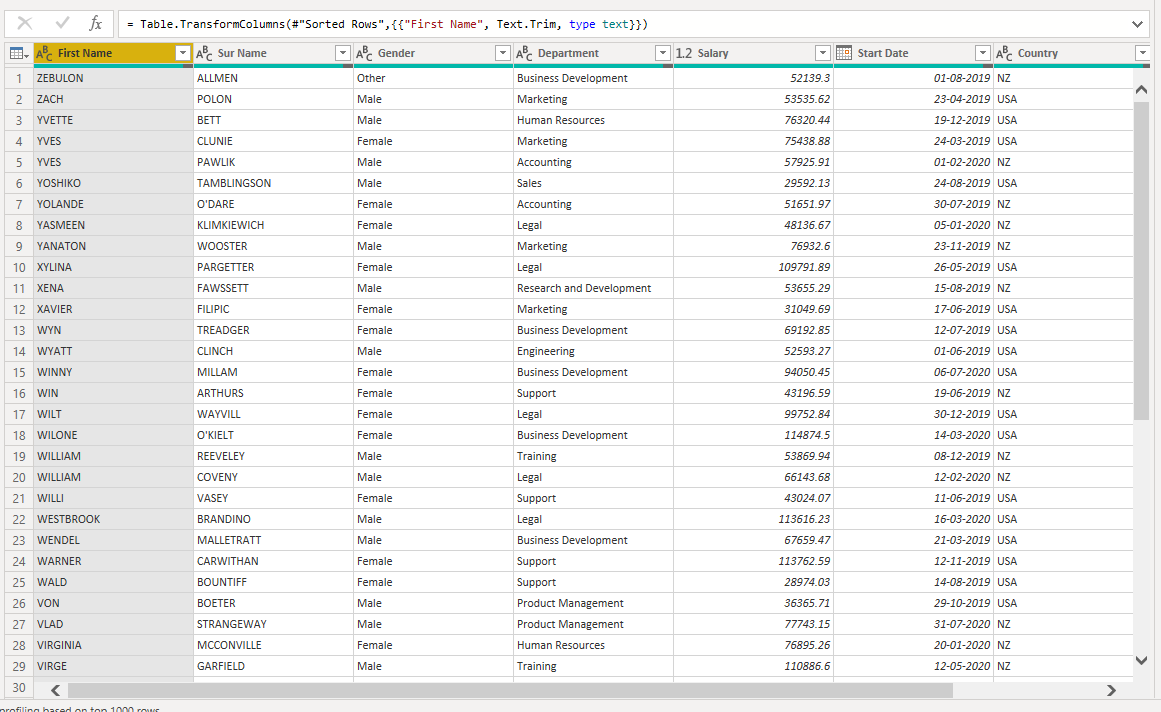
1. Convert the data in the Name column to Upper case.



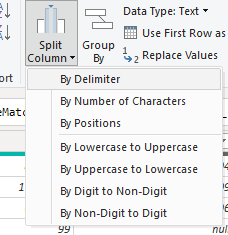


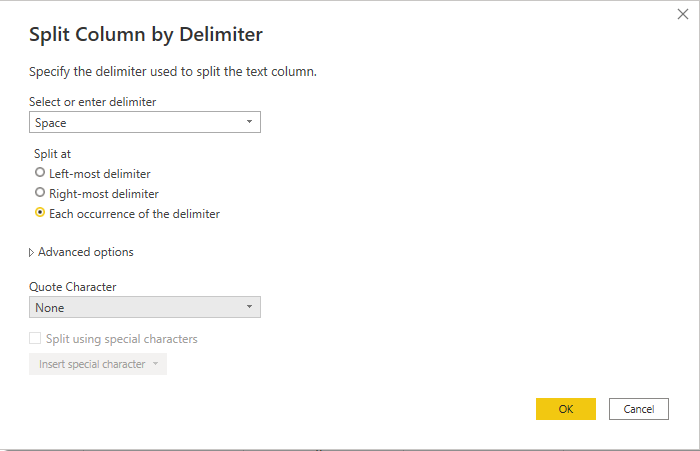
1. Remove all the extra whitespaces in the Name column.

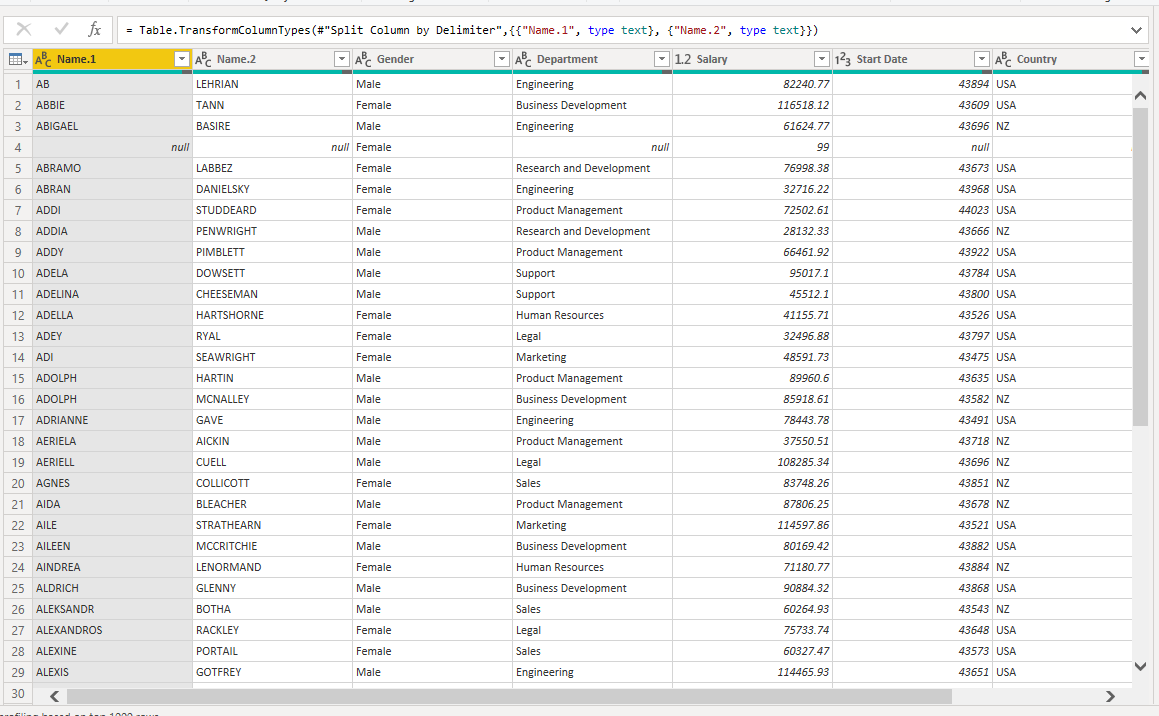




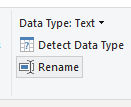
1. Split the Name column into two.

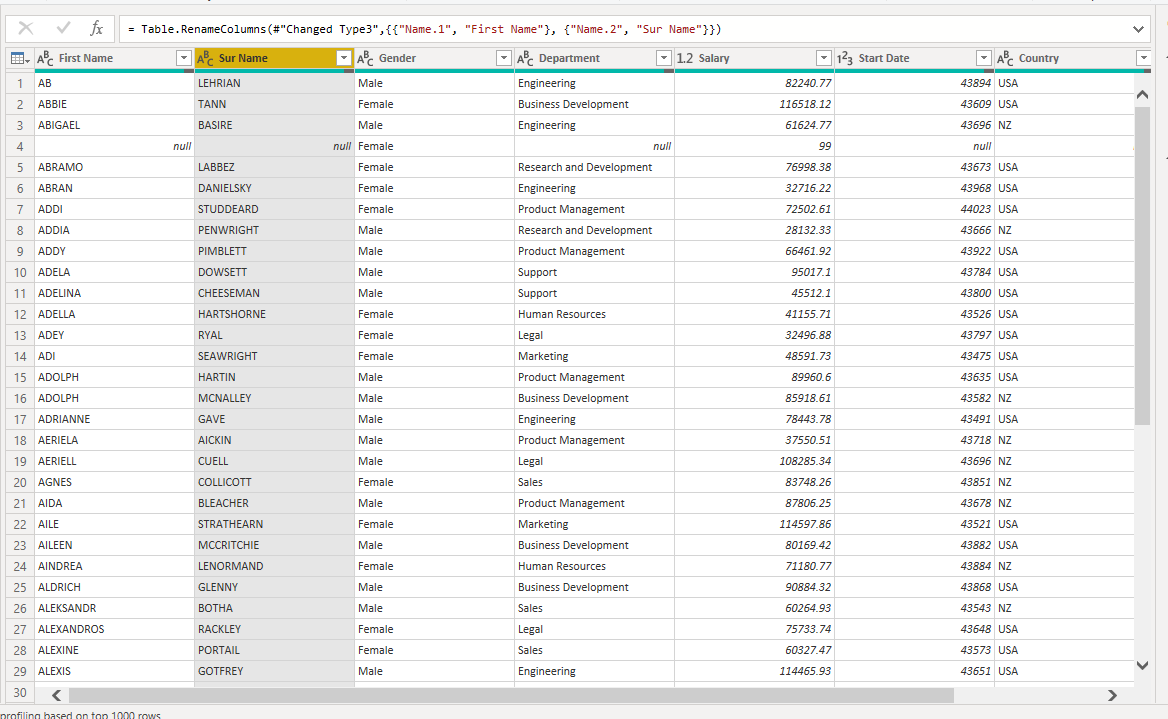




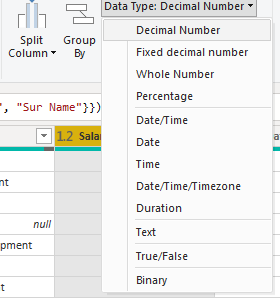


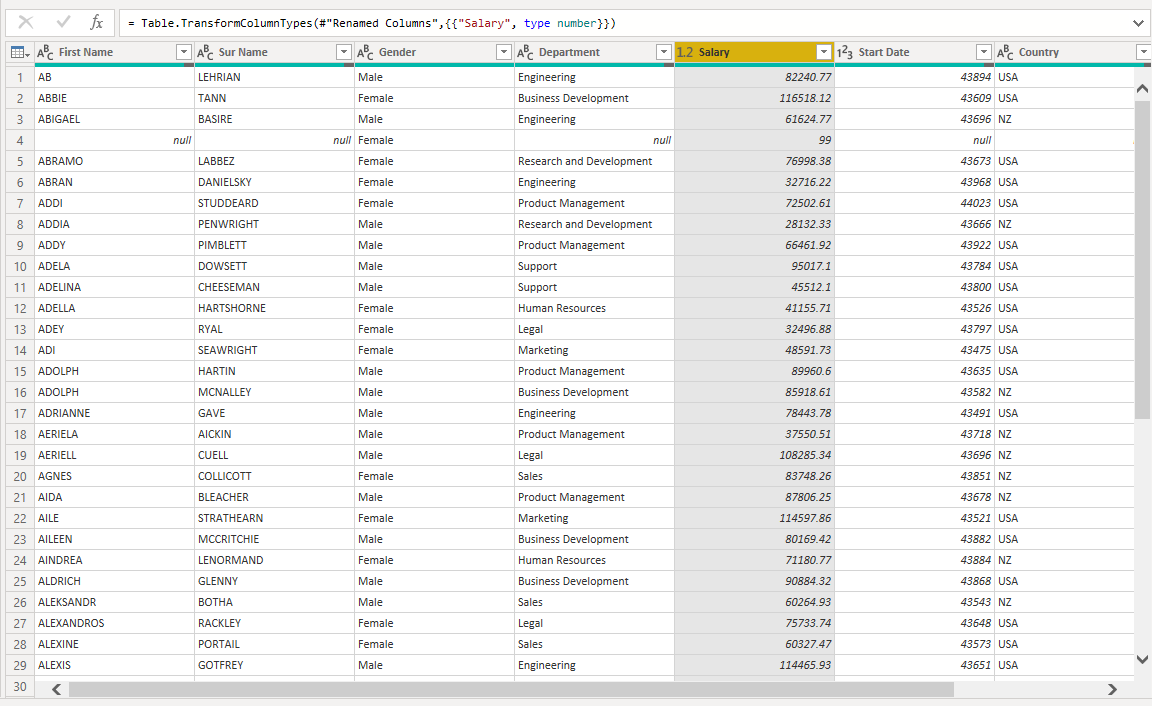
1. Rename the Name1 column in to First name and Name 2 as Surname.



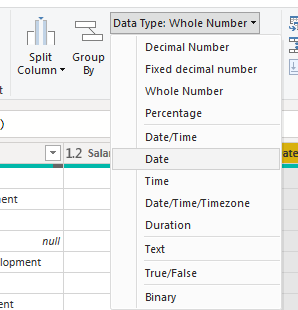


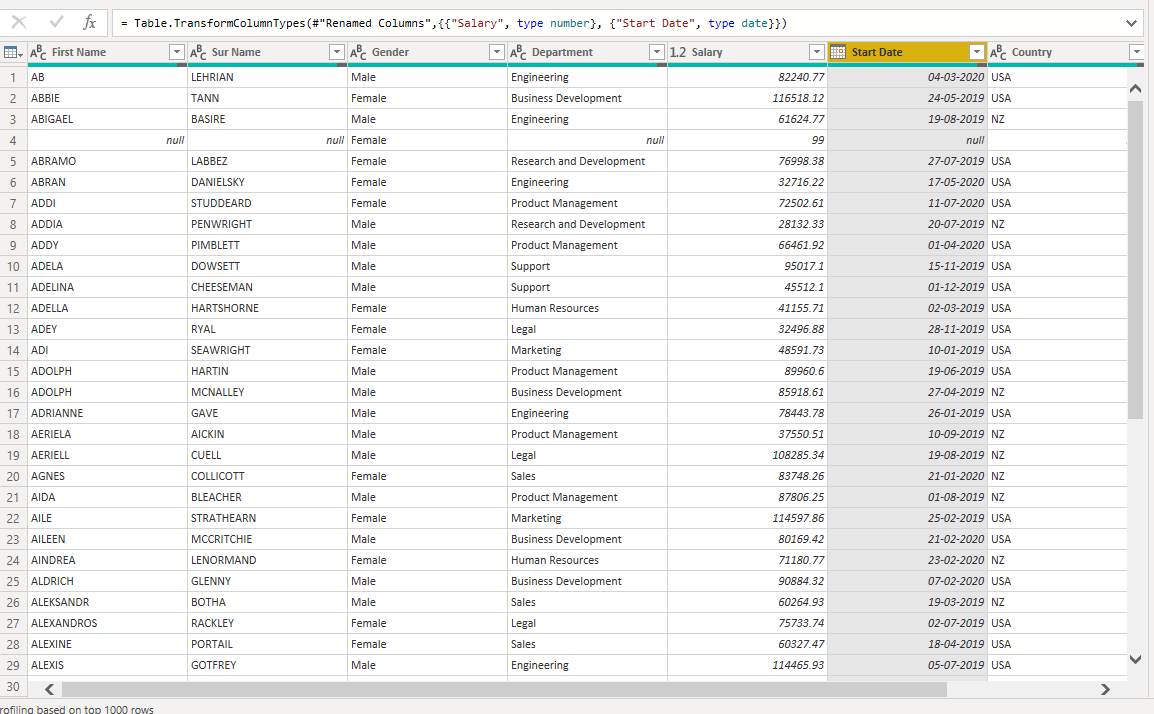
1. Change the data type of salary as decimal number.



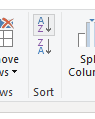


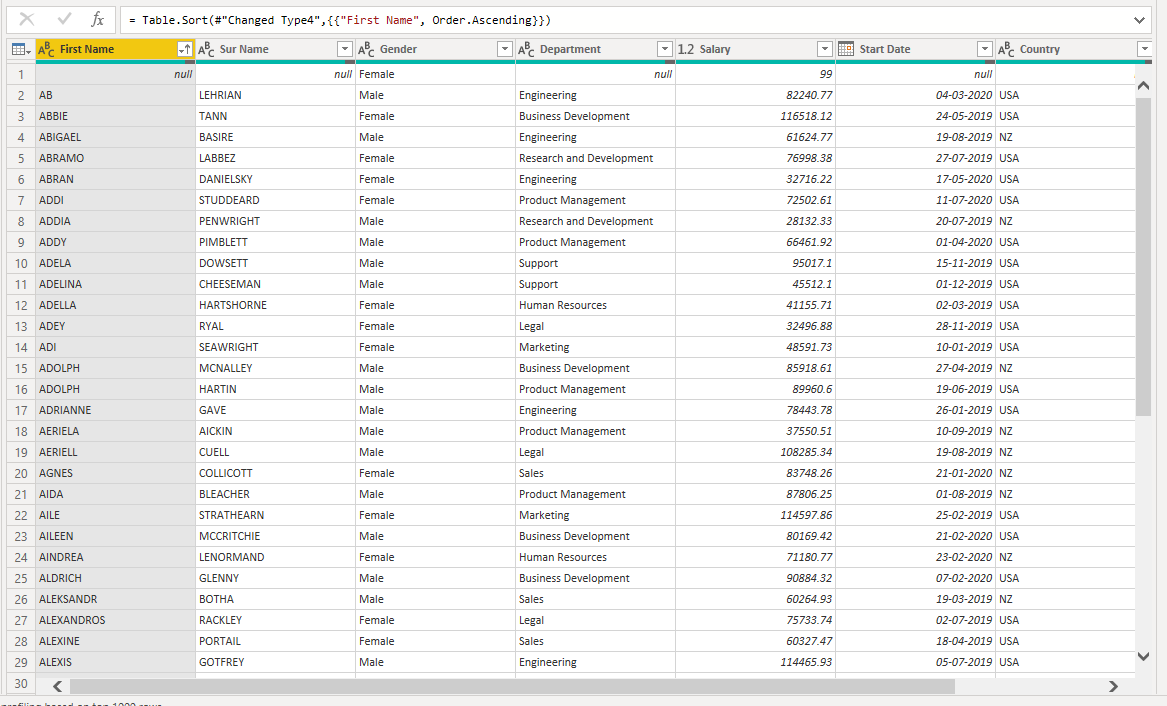
1. Change the data type of Start date column as date.



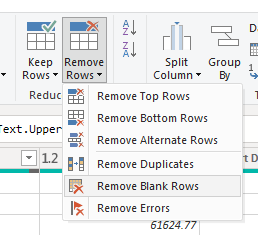


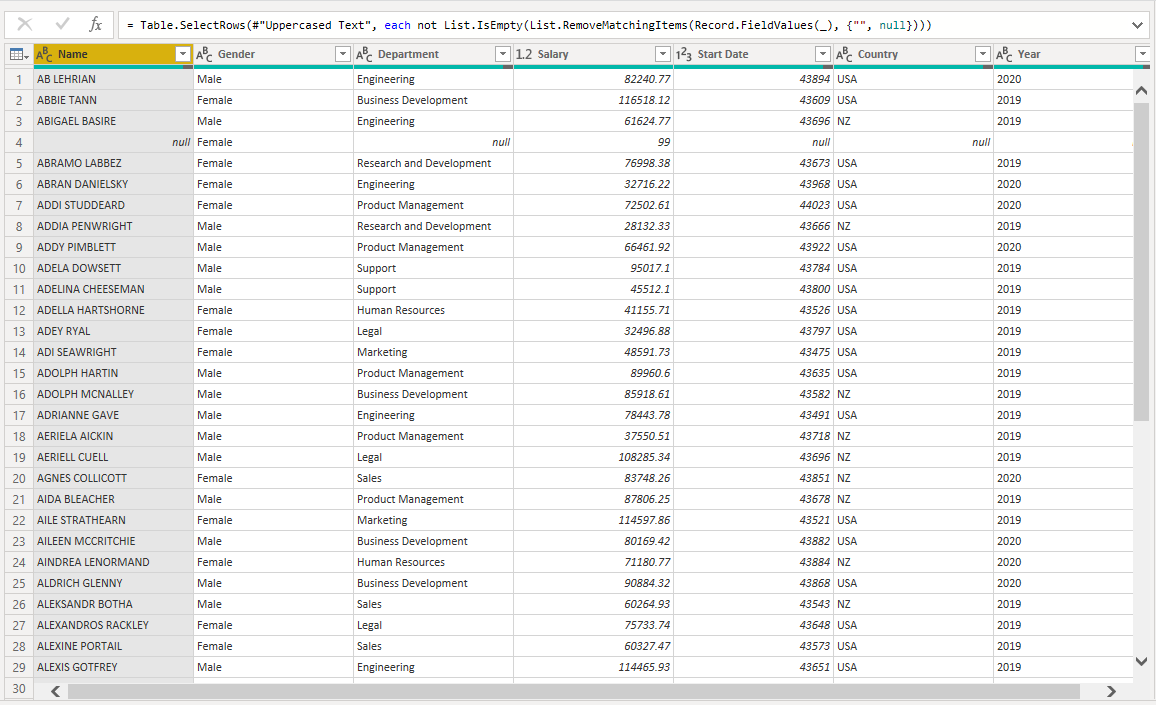
1. Sort the data in the ascending order of Name





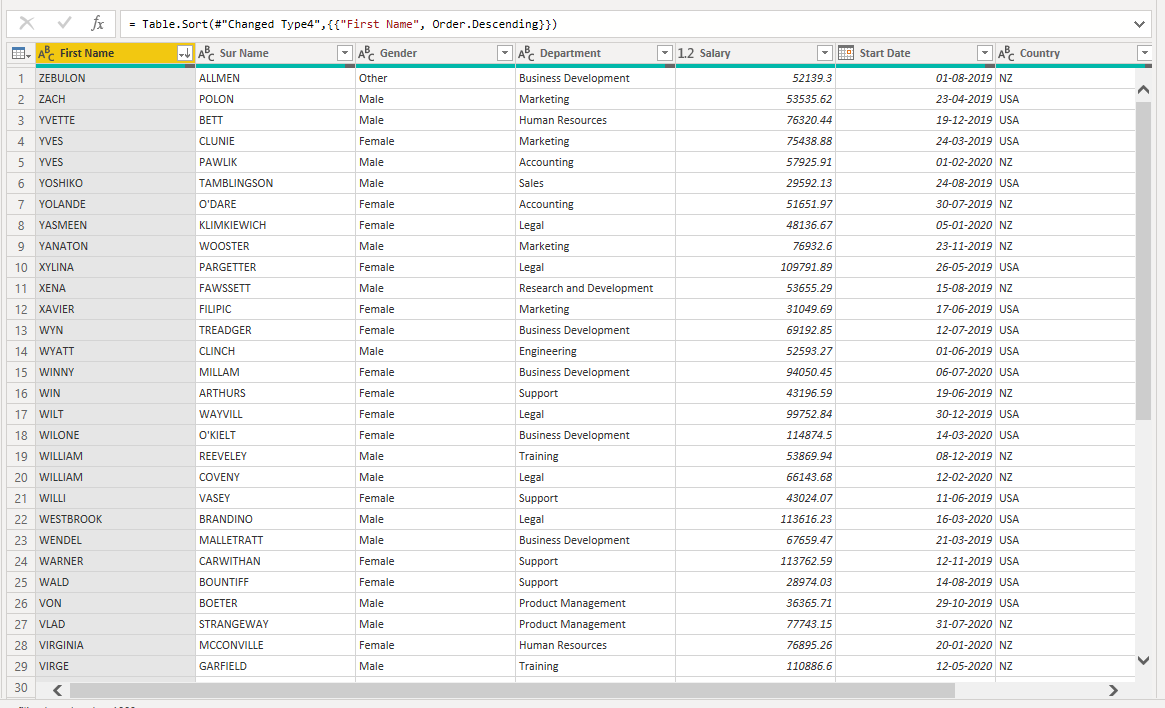
1. Remove the rows with null in name column.



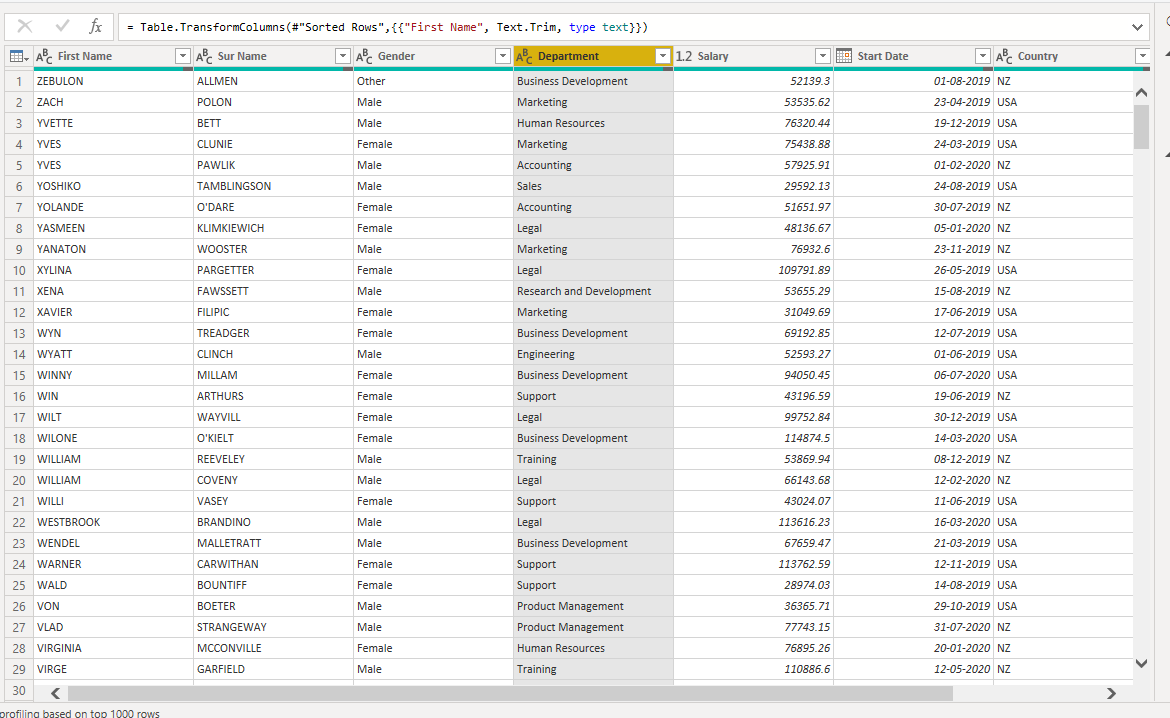


1. Sort by descending order of first name.

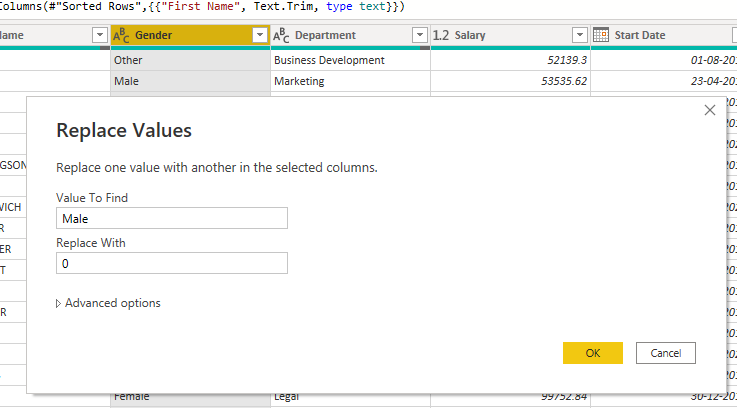




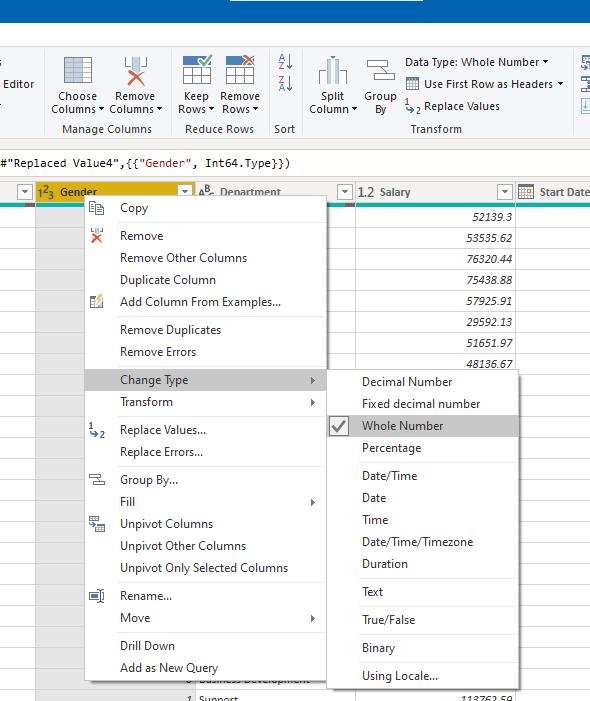
1. Remove the staff with department as null.



1. Change the field gender to 0 or 1 such that if the gender is male it should be 0 and if it is female it should be 1 and other should be 2.



1. Convert the column gender to numeric.

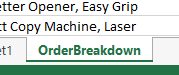


1. Save the file.

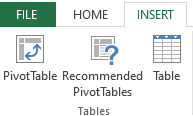
|  |  |
| --- | --- |
| **Practical 8** | |
| **Aim**: Create the Pivot table and Pivot Chart in Microsoft Excel. | |
| Name: Aryan Penikal | Roll No: KCTBCS050 |
| Performance date: | Sign: |

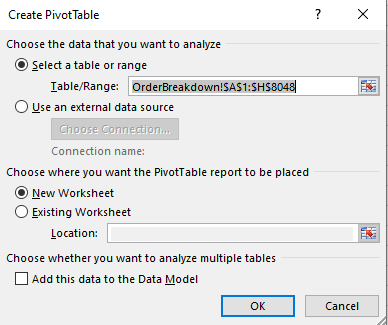
Open AmazingMart.xlsx

1. Select OrdersBreakdown Sheet

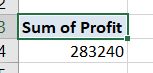


1. Insert a Pivot table in new worksheet.

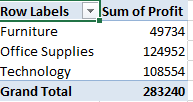




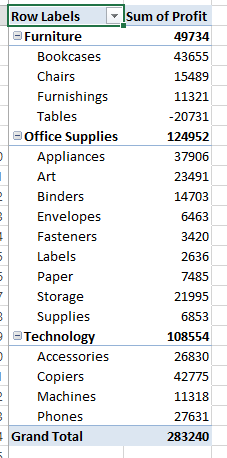
1. Display the total Profit incurred.



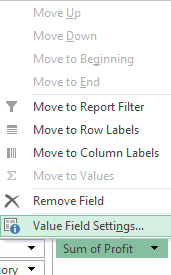
1. Get the total Profit of each category of items.

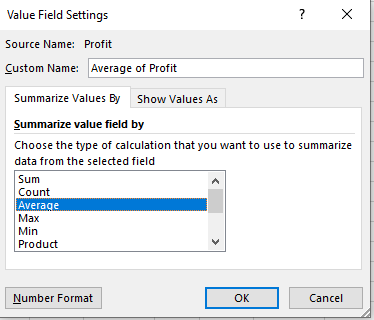


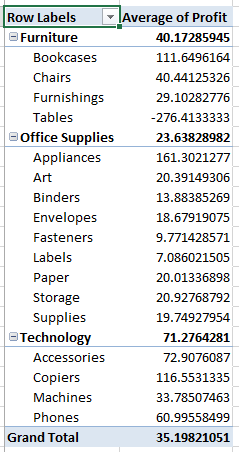
1. Display the total Profit distribution as per every subcategory inside category of items.



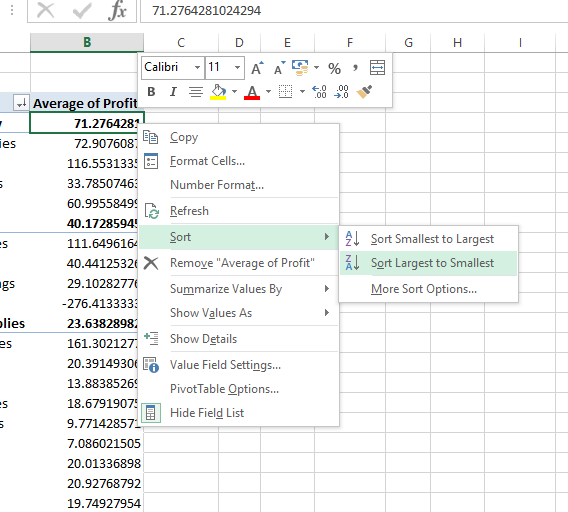
1. Display the AVERAGE Profit distribution as per every subcategory inside category of items.

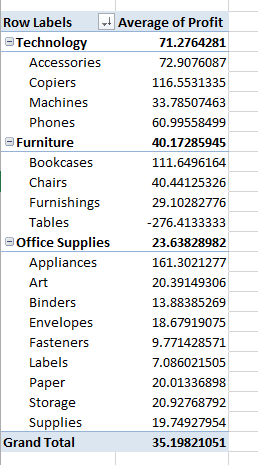




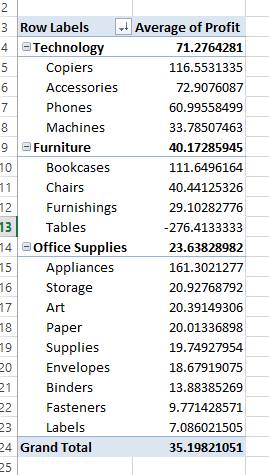


1. Sort the Profit in descending order.





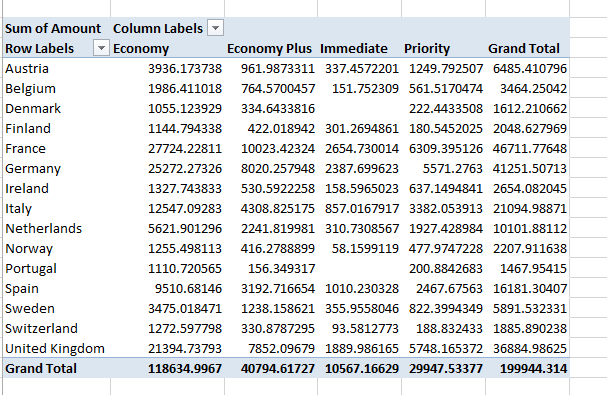
1. Sort the Category and subcategory in the descending order.

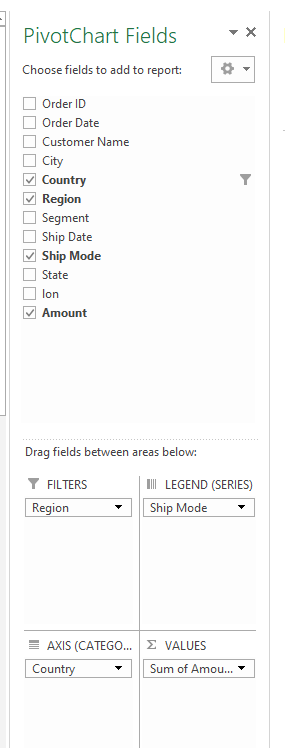


1. Use ListOfOrder sheet

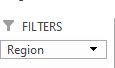


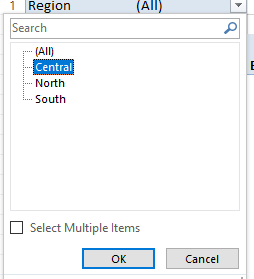
1. Insert 2 dimensional Pivot table such that countries are available along rows, shipping mode is available in columns along with total amount as values.

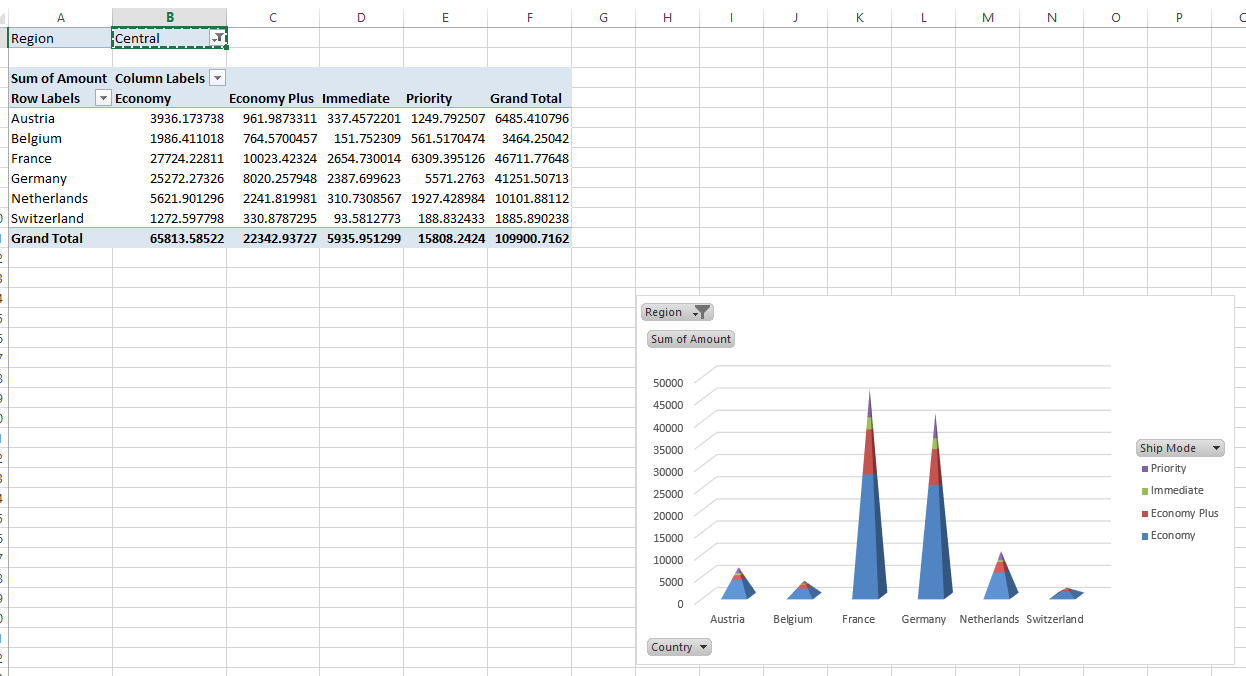




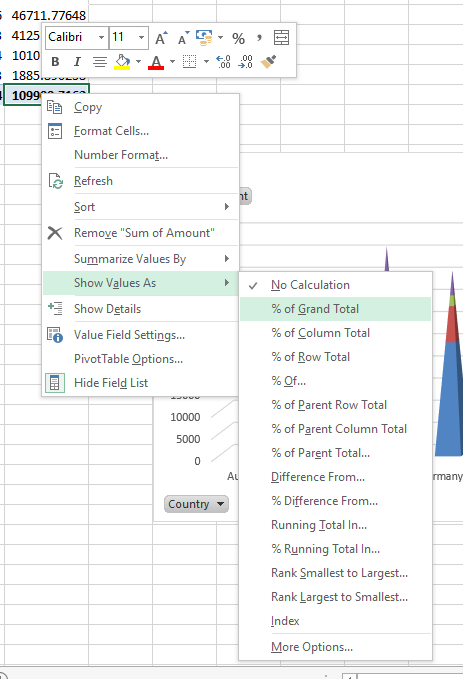
1. Apply the filter for central region.

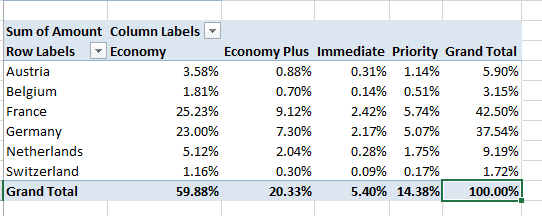




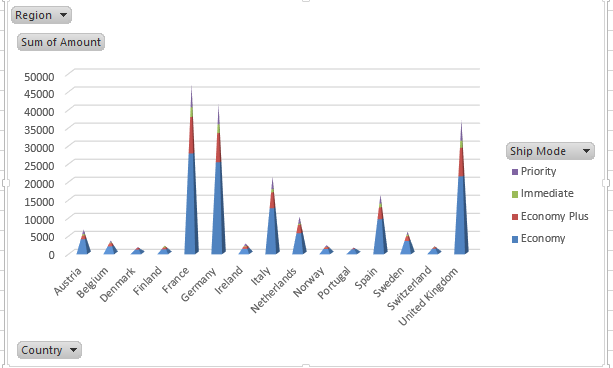


1. Display the data in the form of percentage of Grand total.





1. Insert a 3D stacked pyramid Pivot chart for the above table



1. Display the data of only Germany.

