**Practical** **No.** **01 : Introduction to Excel**

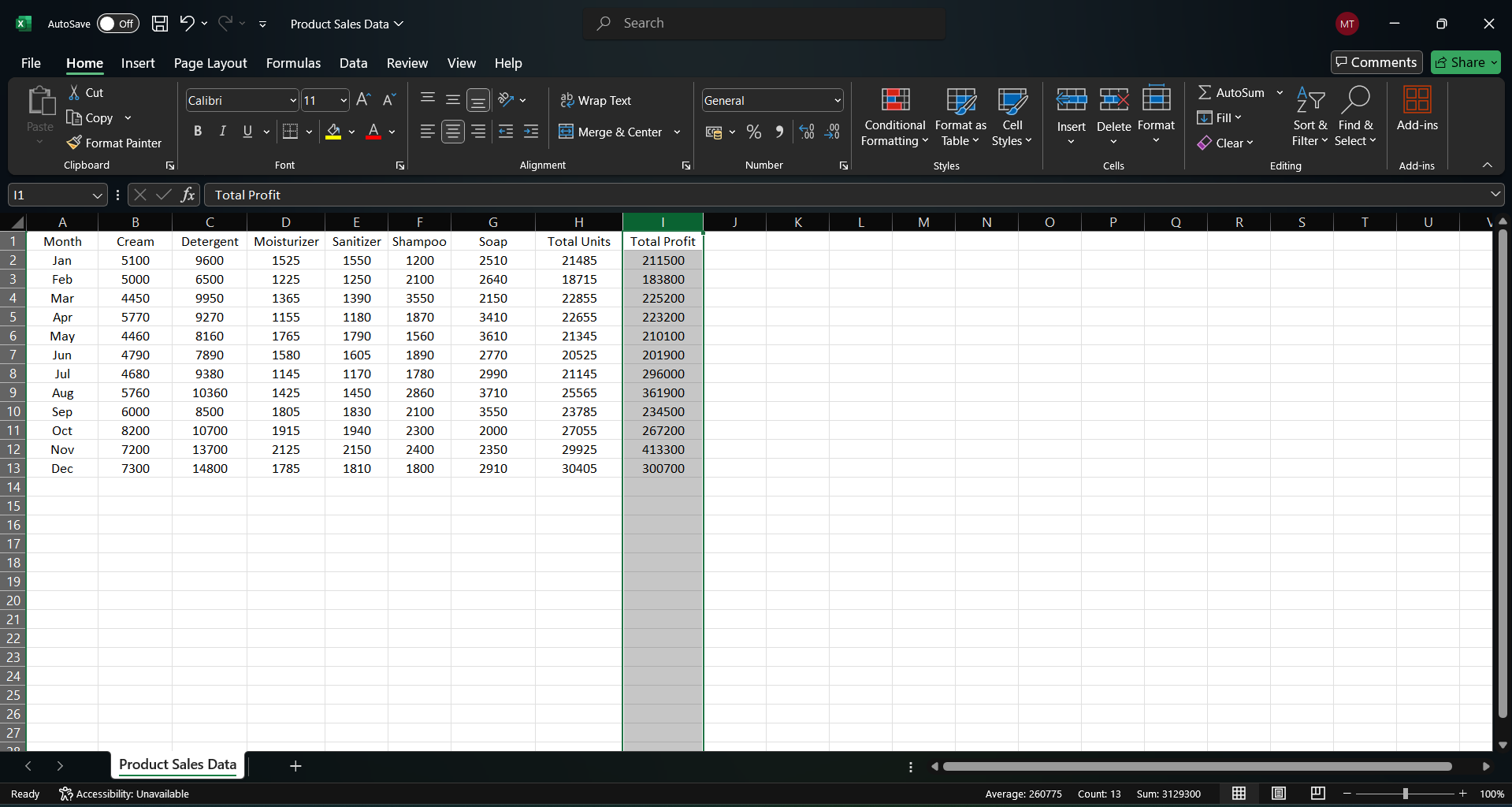
**Aim:** **Perform conditional formatting on a dataset using various criteria.**

* **Perform** **conditional** **formatting** **on** **a** **dataset** **using** **various** **criteria.**

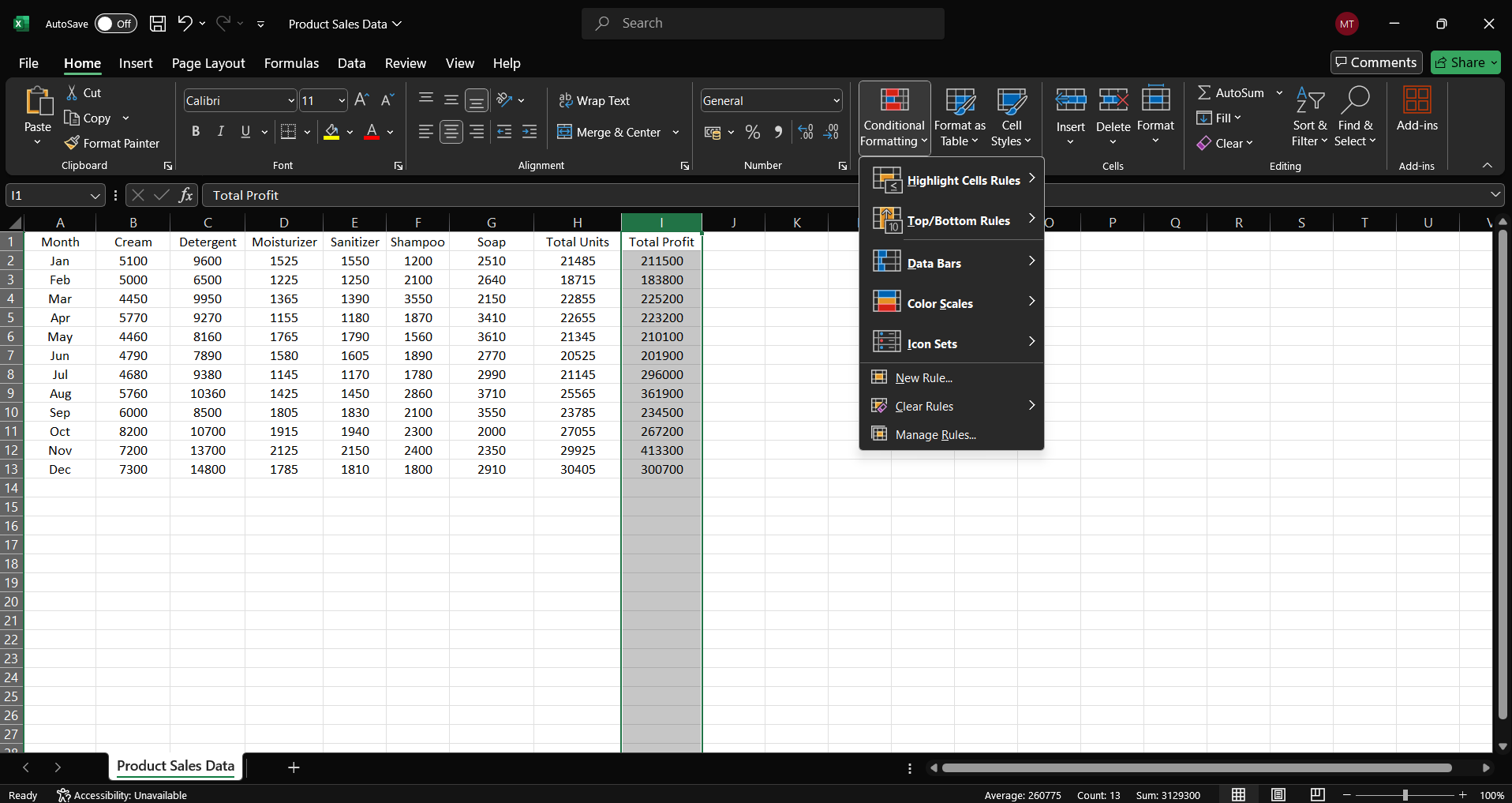
**We perform conditional formatting on the "Profit" column to highlight cells with a profit greater than 250000 using following steps:**

**Steps:**

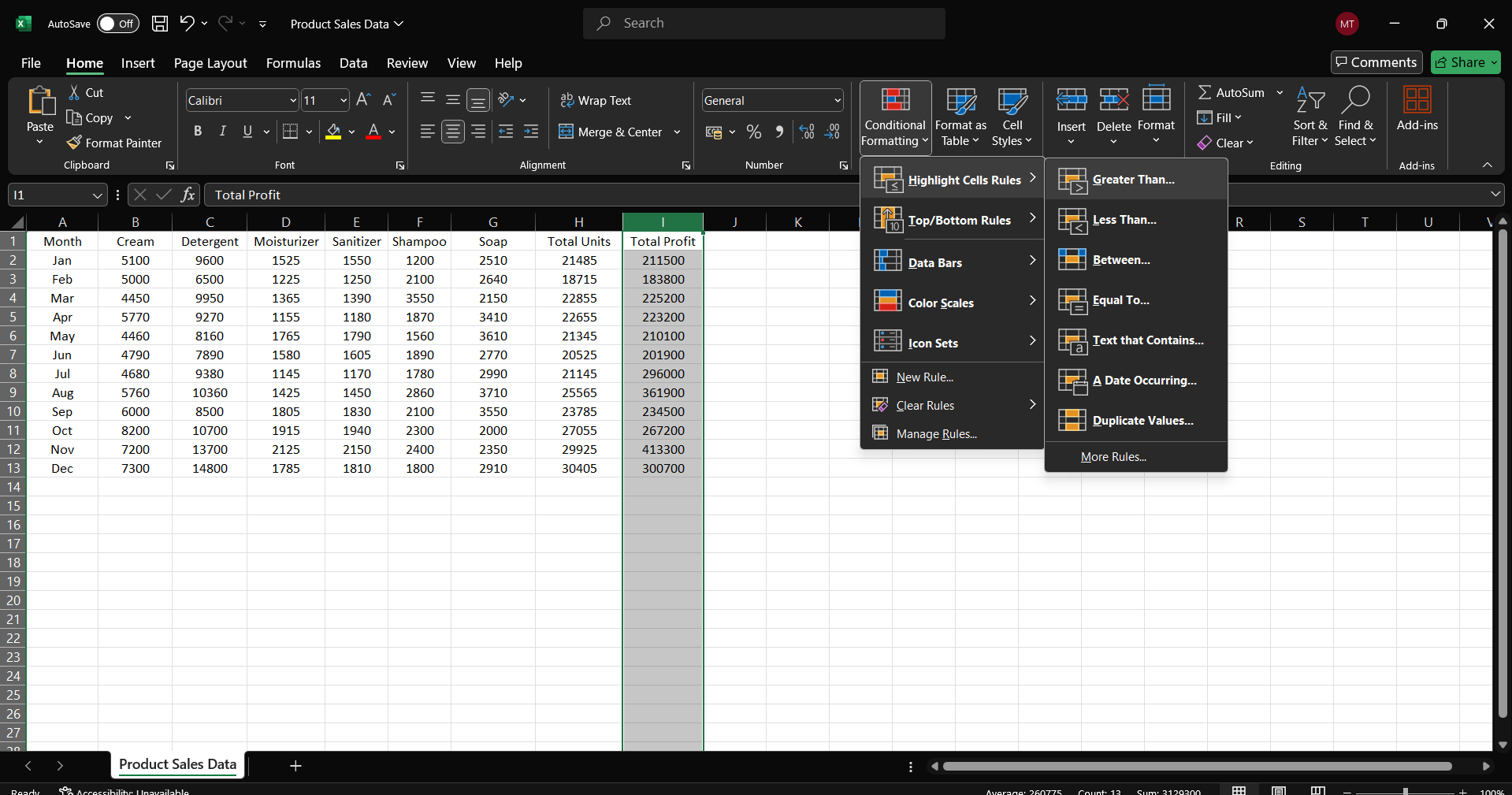
1. Select the "Profit" column (Column I).

****

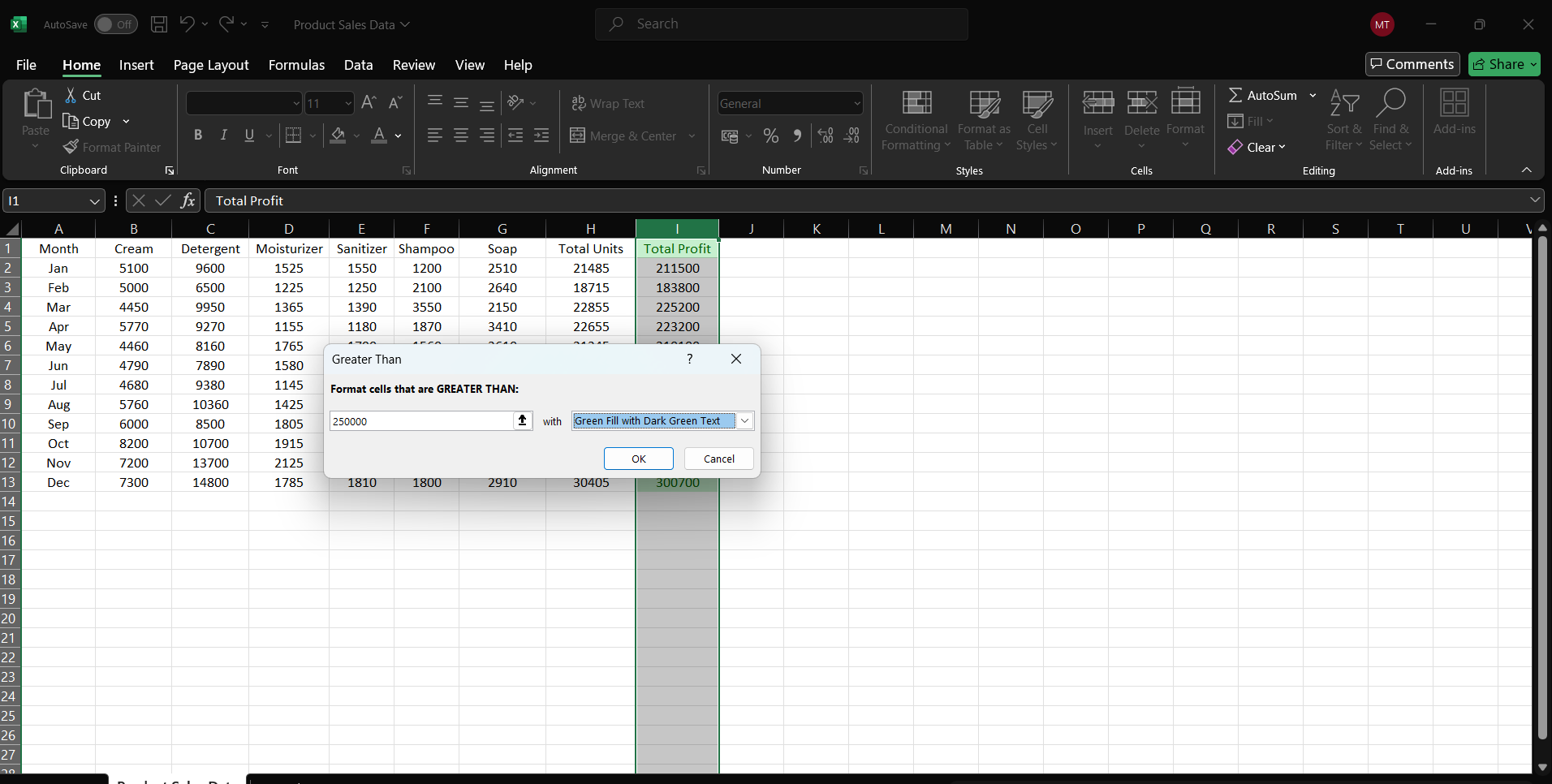
1. Go to the "Home" tab on the ribbon.
2. Click on "Conditional Formatting" in the toolbar.

****

1. Choose "Highlight Cells Rules" and then "Greater Than."

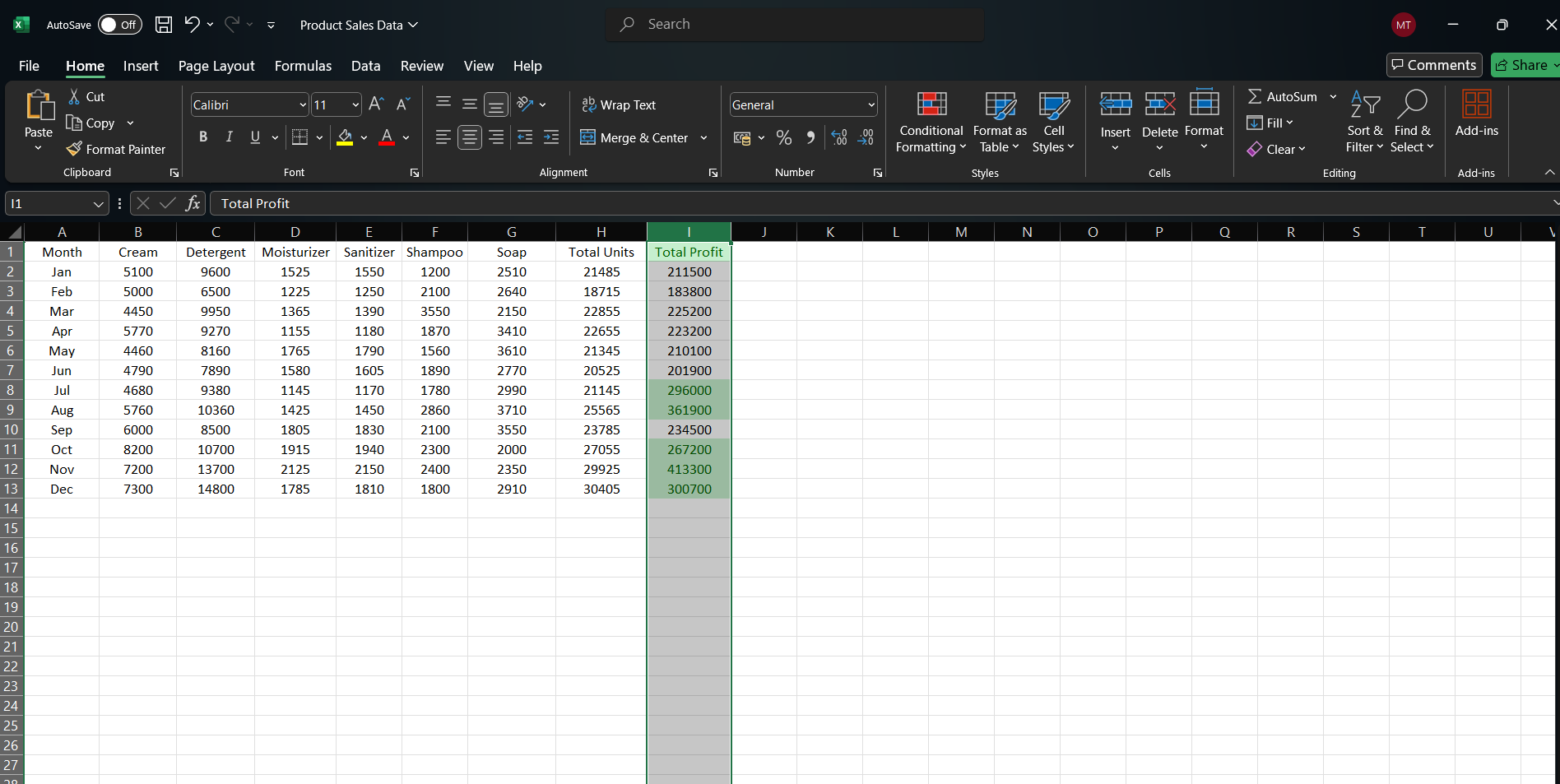
****

1. Enter the threshold value as 250000

****

6 : Customize the formatting option(e.g., choose a fill color).

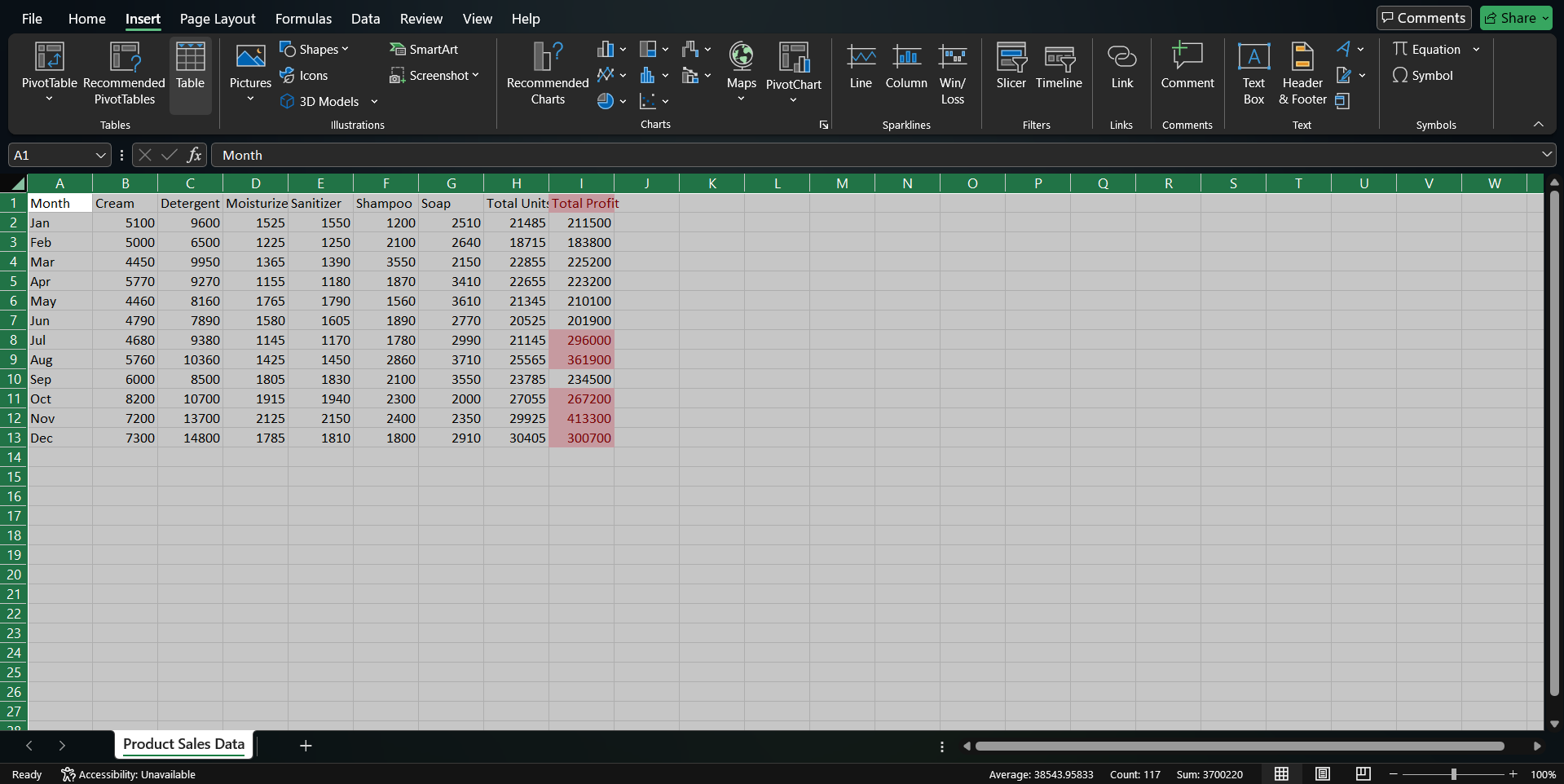
7 : Click “OK” to apply the rule

****

**Aim : Create a pivot table to analyze and summarize data.**

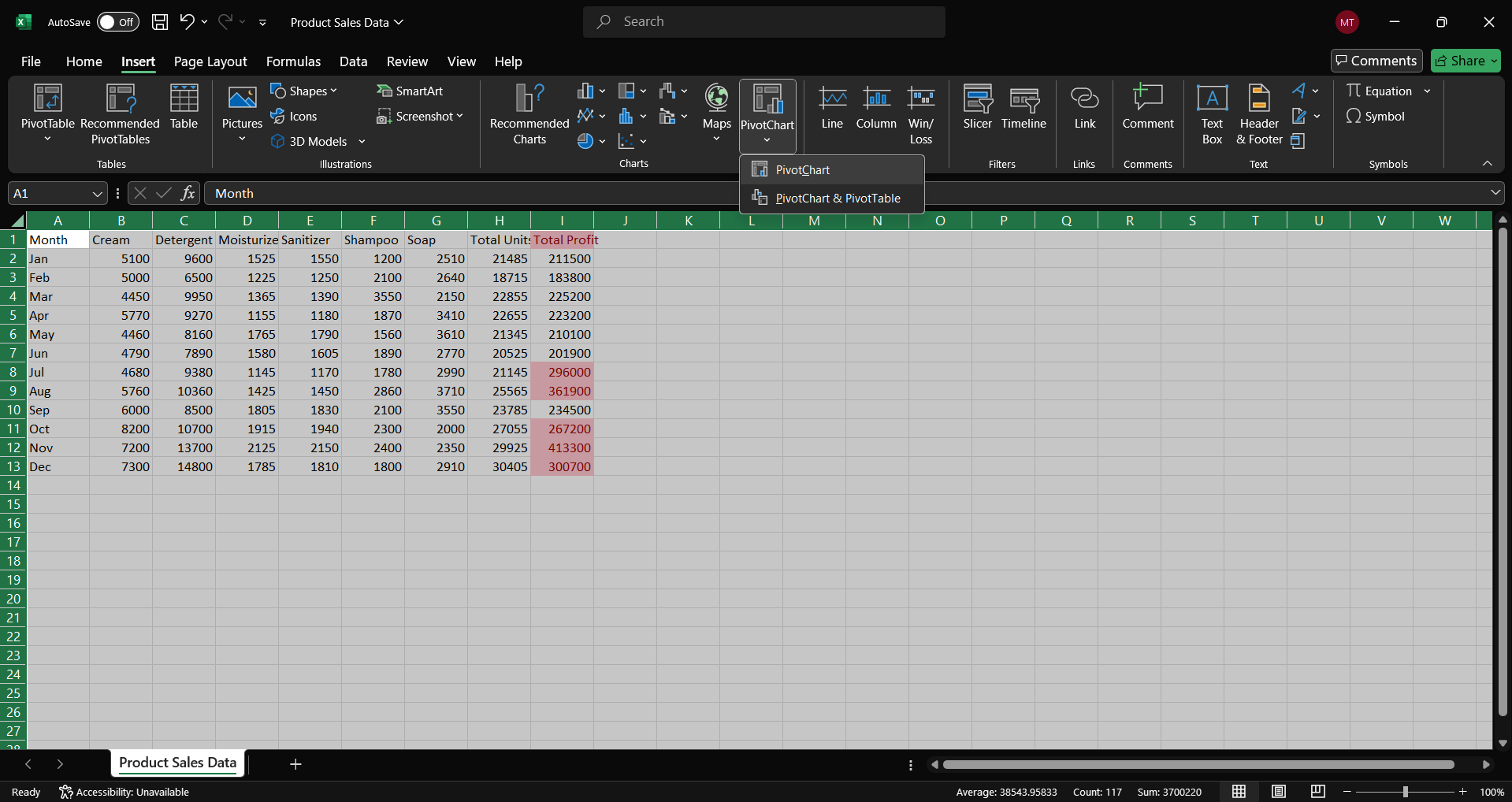
Steps :

1: Select the entire dataset including headers.

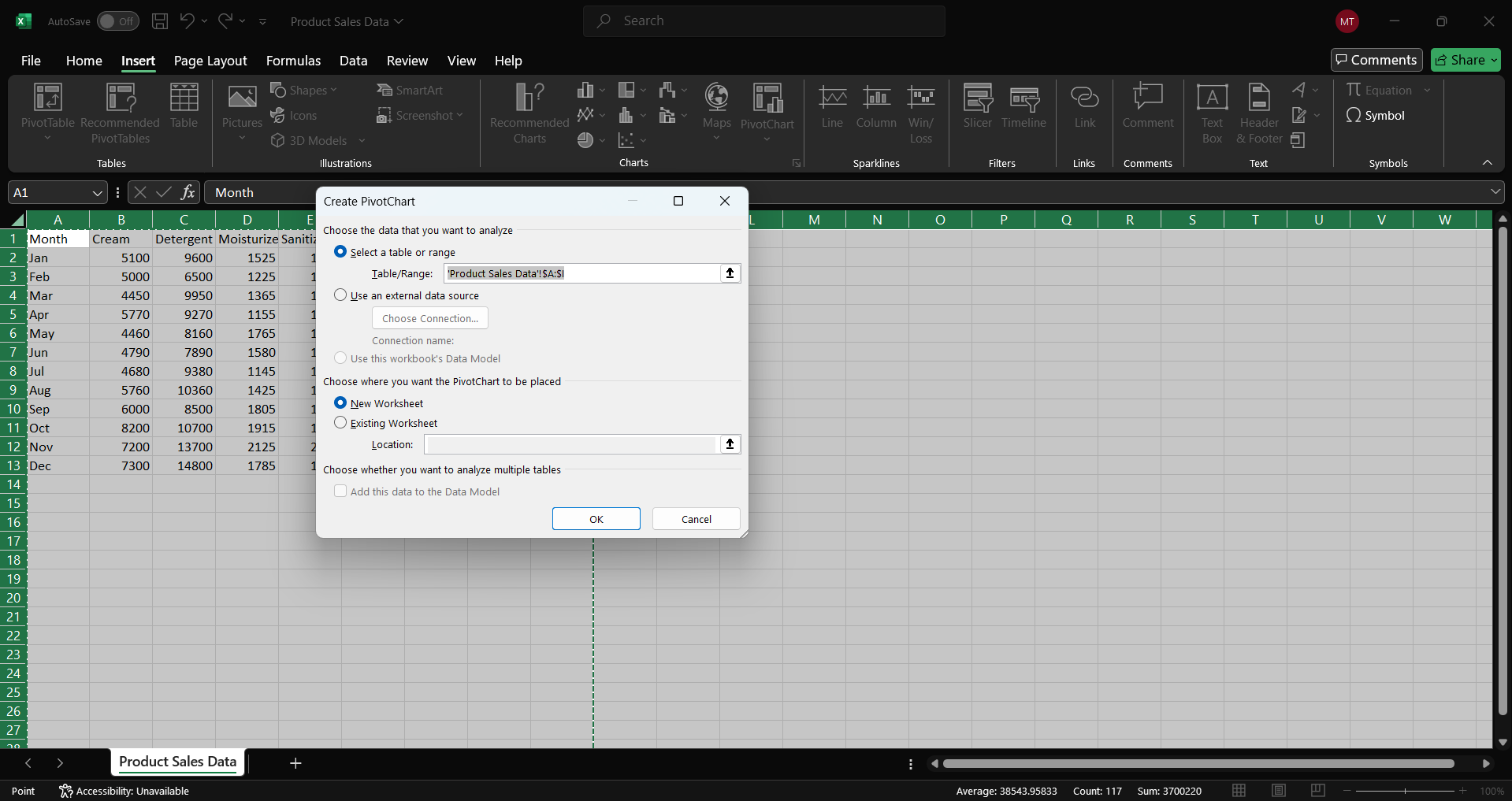
****

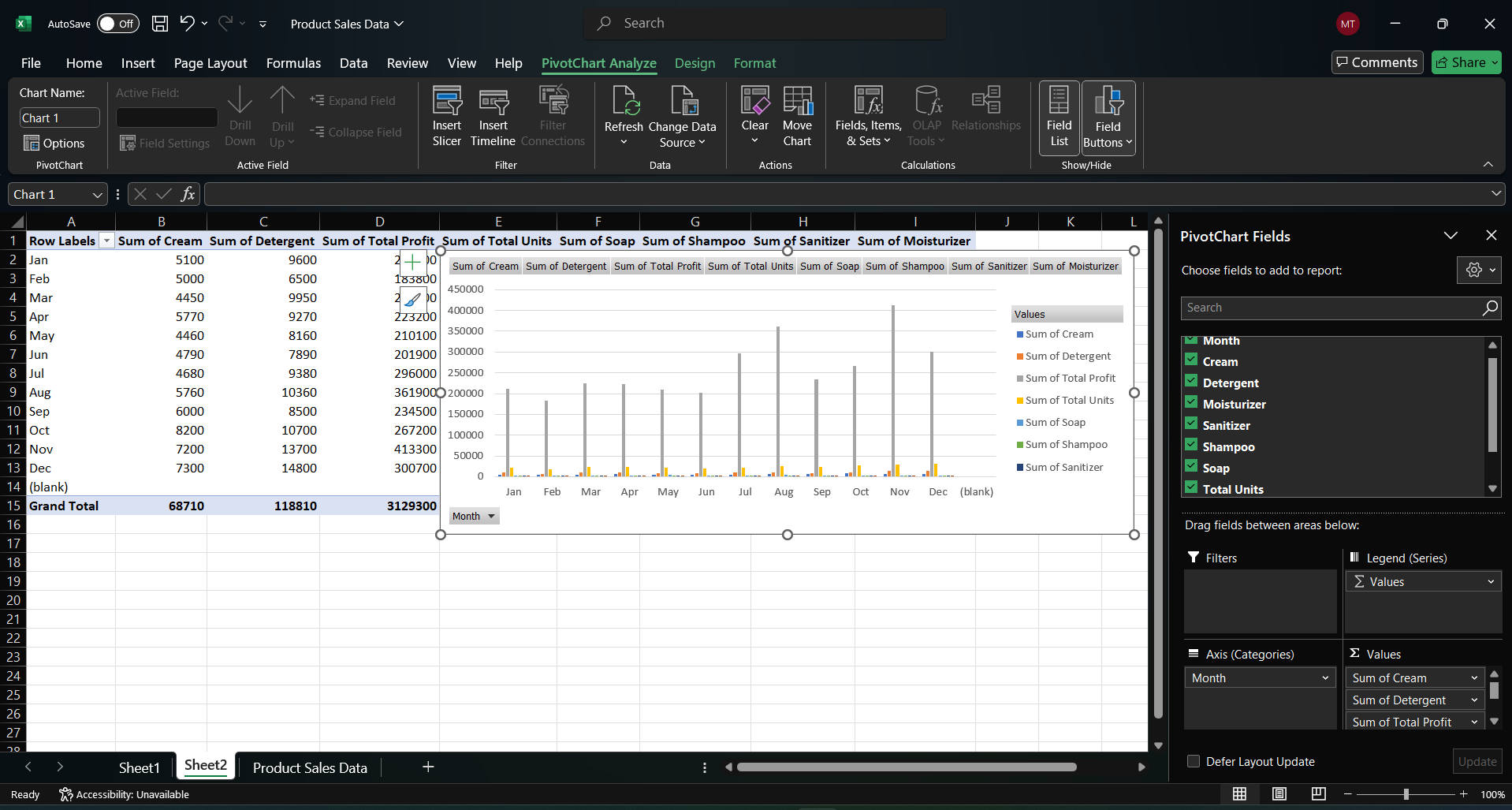
2 : Go to the “Insert ” tab on the ribbon

3 : Click on “PivotTable”



4 : Choose where you want to place the pivotable (e.g., new worksheet).



****

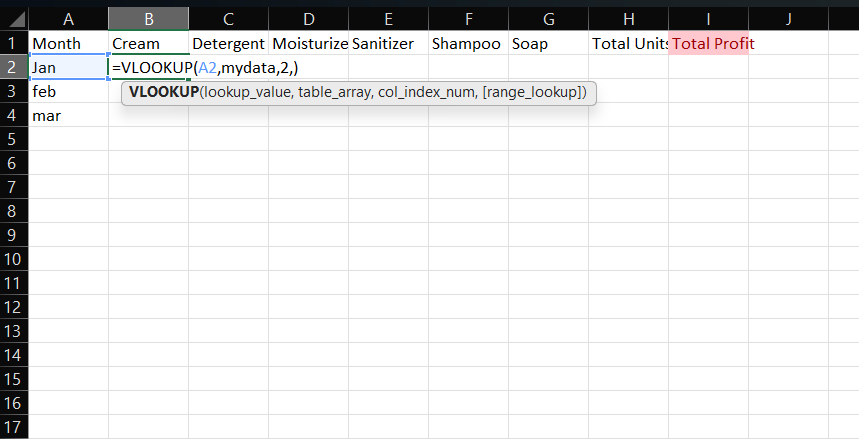
**Aim : Use VLOOKUP function to retrieve information from a different worksheet or table .**

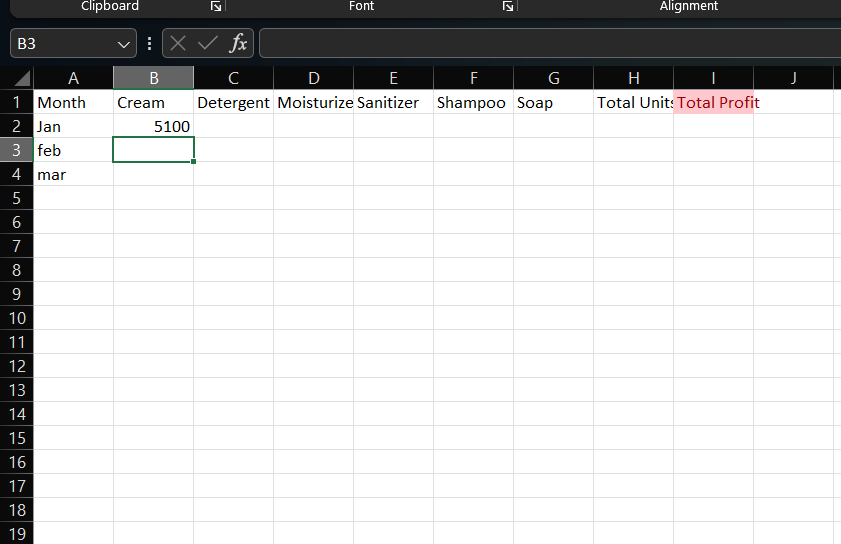
Steps :

1 : Assuming your “Product Table ” is in a different worksheet.

2 : In a cell in your main dataset, enter the formula :

=VLOOKUP(A2,mydata,2,0)





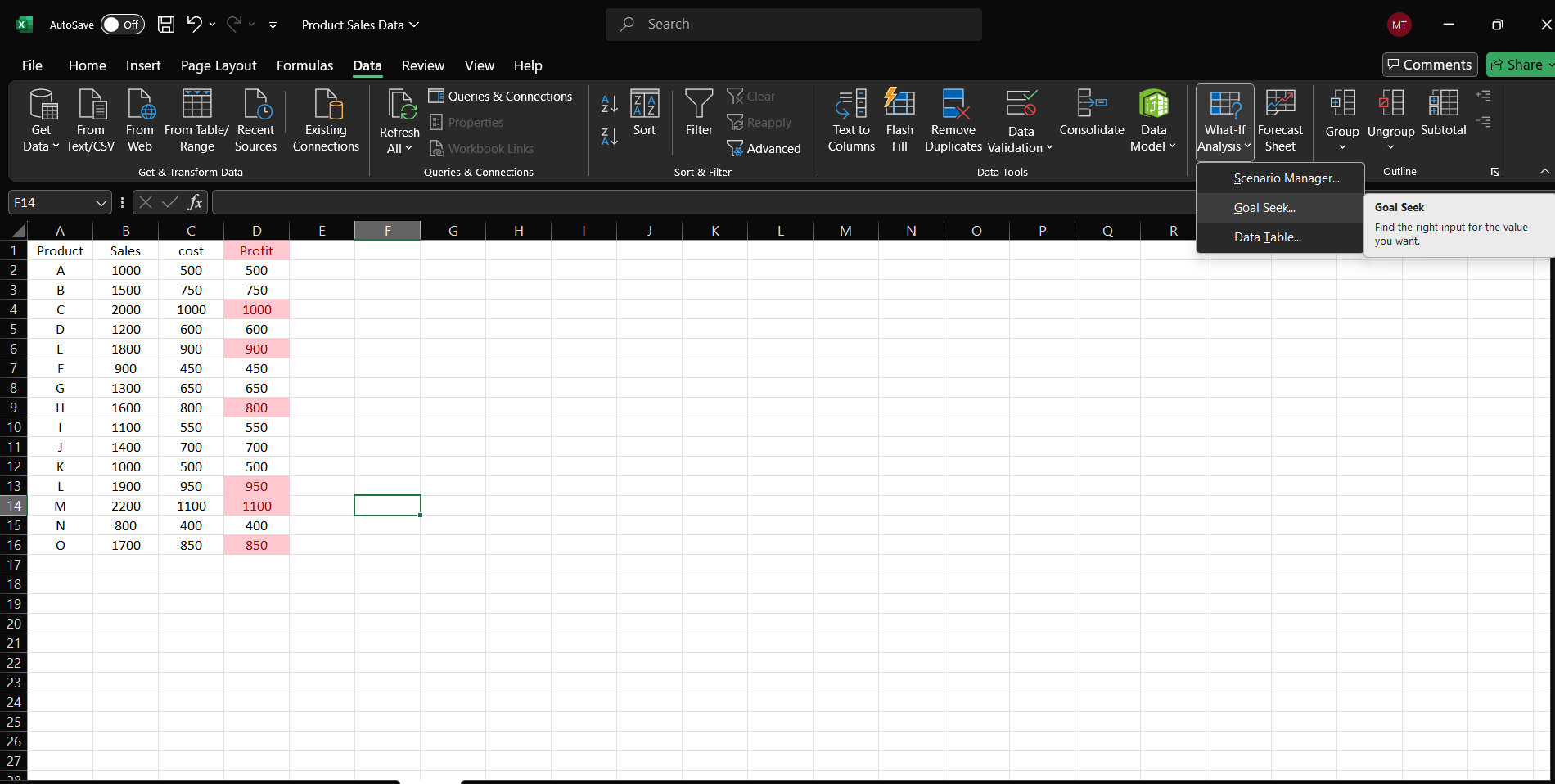
**Aim : Perform what-if analysis using Goal seek to determine input values for desired output**

Steps :

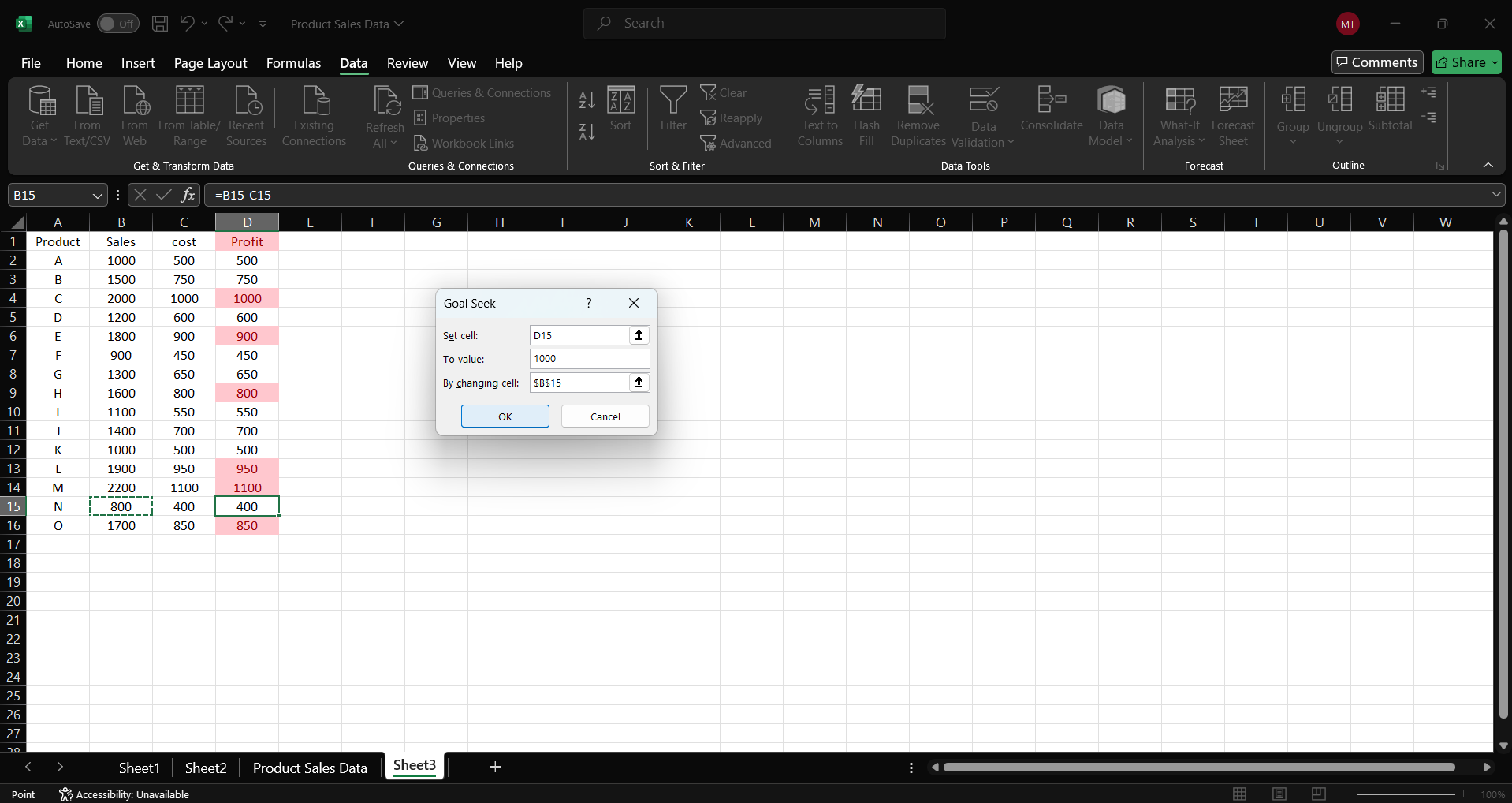
1: Identify the cell containing the formula for “Profit” for “Product P” (let’s assume it’s in cell E15)

2: Go to the “Data” tab on the ribbon.

3: Click on “What-If-Analysis” and select “Goal seek”.

****

4: Set “Set cell” to the profit cell (E15), “To value” to 1000 , and “By changing cell” to the sales cell (C15)

****

5: Click “OK” to let Excel determine the required sales

**Practical** **No.** **02**

**Aim:** Data Frames and Basic Data Pre-processing

* + Read data from CSV and JSON files into a data frame.
  + Perform basic data pre-processing tasks such as handling missing values and outliers.
  + Manipulate and transform data using functions like filtering, sorting, and grouping.

# Data pre-processing:

Data pre-processing is a crucial step in the data analysis pipeline, encompassing tasks such as reading data from various file formats, handling missing values, and managing outliers. This practical guide explores how to execute these tasks using the pandas library in Python.

# Steps:

**Step** **1:** **Reading** **from** **CSV** **and** **JSON** **Files**

1. Utilize pandas to read data from a CSV file ('student.csv') into a data frame.
2. Use pandas to read data from a JSON file ('ds.json') into a data frame.
3. Display the first few rows of each data frame to inspect the data.

**Step** **2:** **Handling** **Missing** **Values**

1. Drop rows with missing values from the CSV data frame.
2. Fill missing values with a specific value (e.g., 0) in the JSON data frame.

**Step** **3:** **Handling** **Outliers**

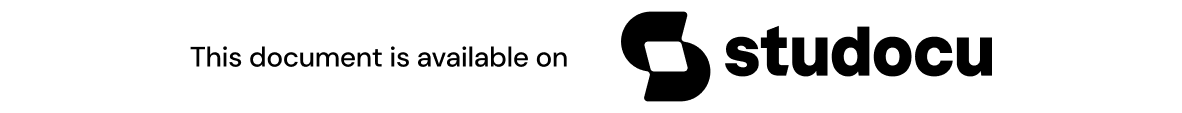
1. Identify outliers in the 'Sales' column of the CSV data frame.
2. Replace outliers with the median value.

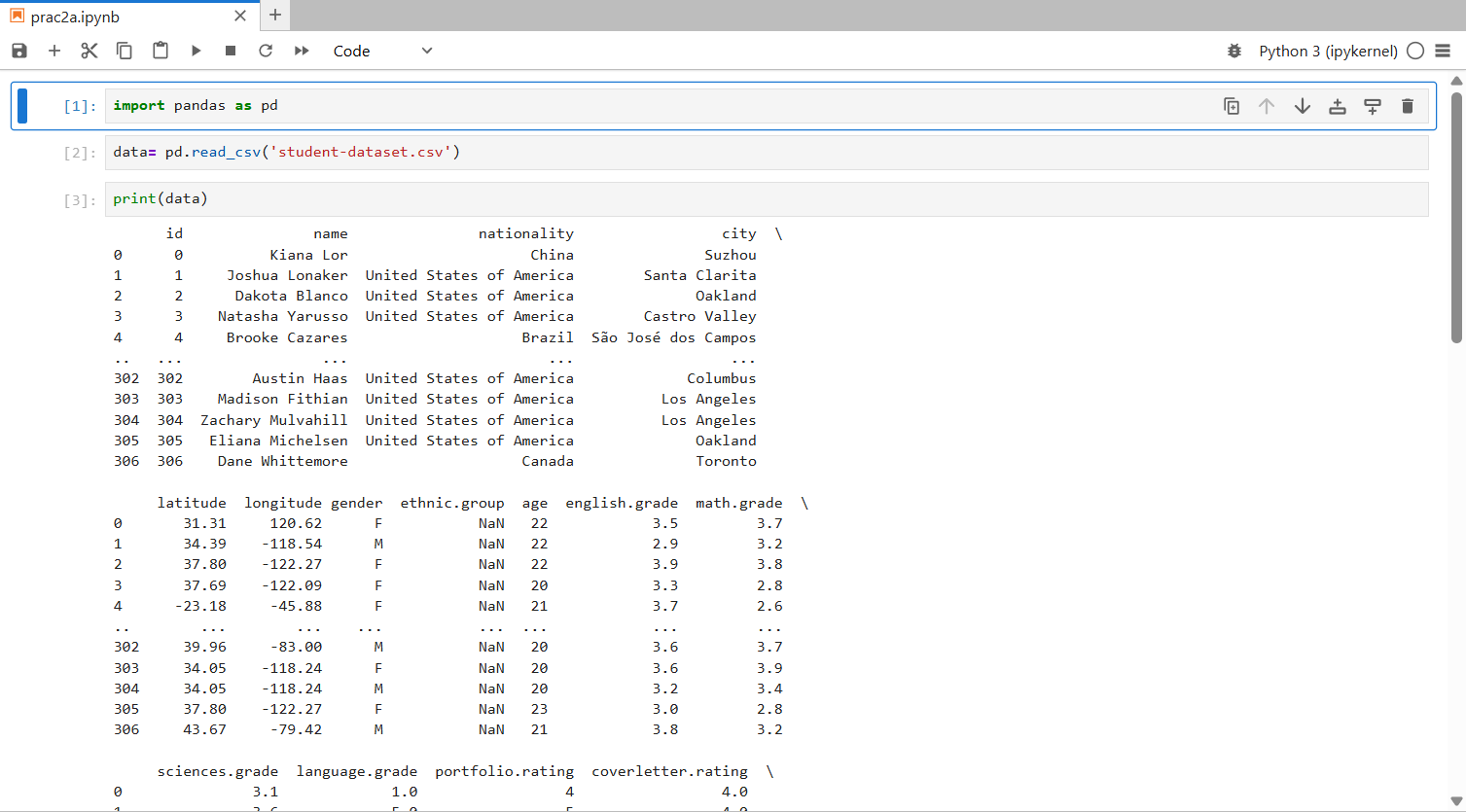
**Step** **4:** **Manipulating** **and** **Transforming** **Data**

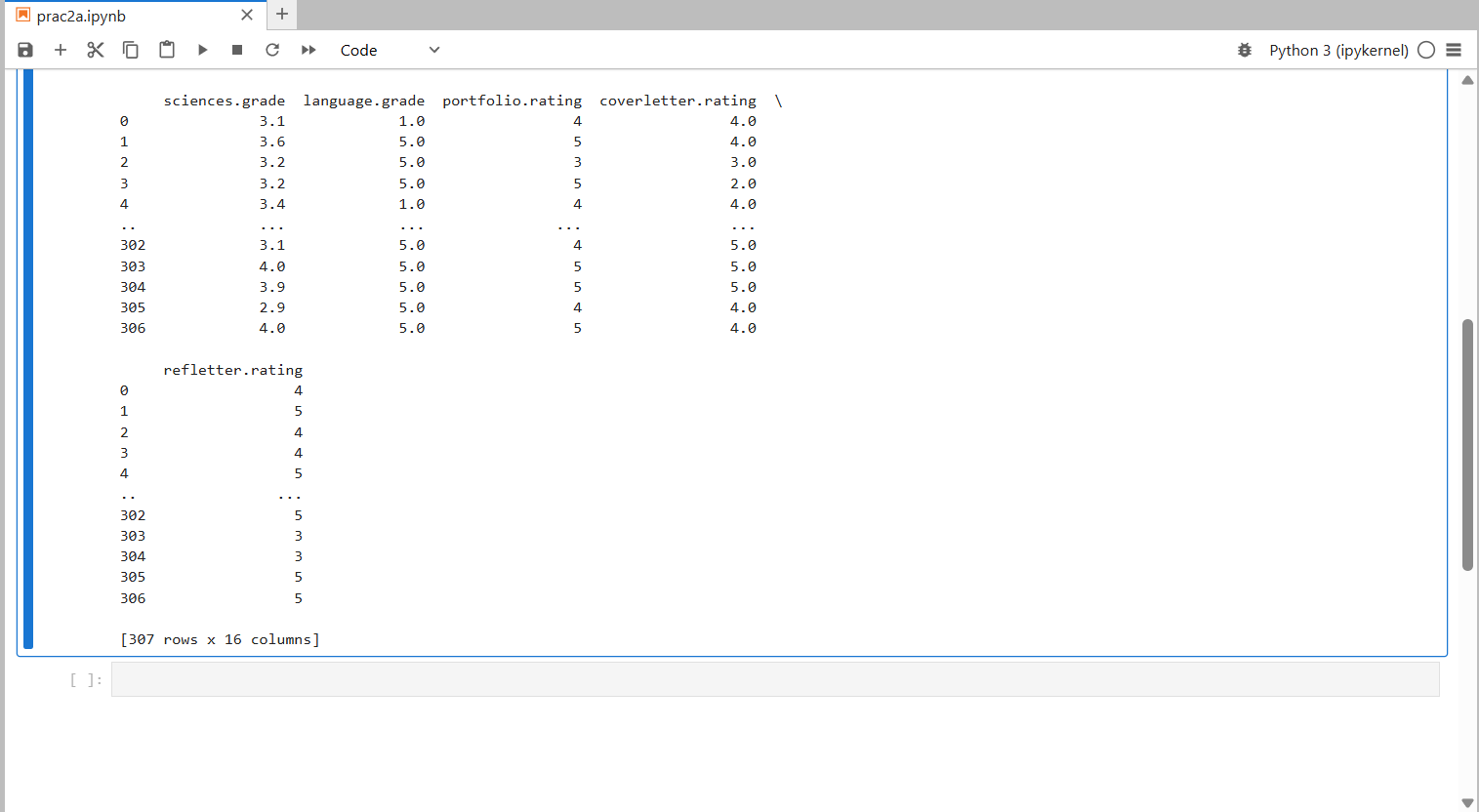
1. Filter the CSV data frame to include only rows where 'Sales' is greater than 10.
2. Sort the CSV data frame based on the 'Sales' column in descending order.
3. Group the CSV data frame by the 'Category' column and calculate the mean for numeric columns ('Sales', 'Cost', 'Profit').

**Step** **5:** **Displaying** **Results**

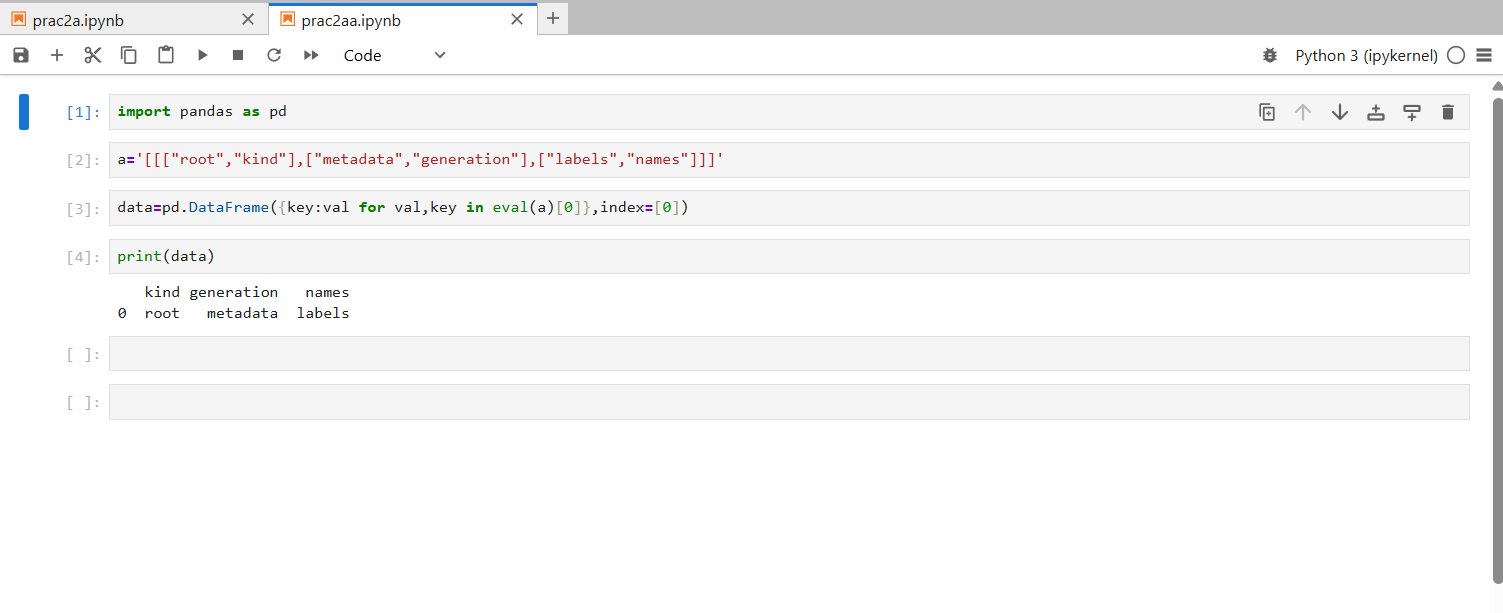
1. Display the cleaned CSV data frame after handling missing values.
2. Display the JSON data frame after filling missing values.
3. Display the filtered CSV data frame.
4. Display the sorted CSV data frame.
5. Display the grouped CSV data frame showing the mean values for numeric columns.

**Step** **1: Reading from CSV and JSON Files into a data frame :**



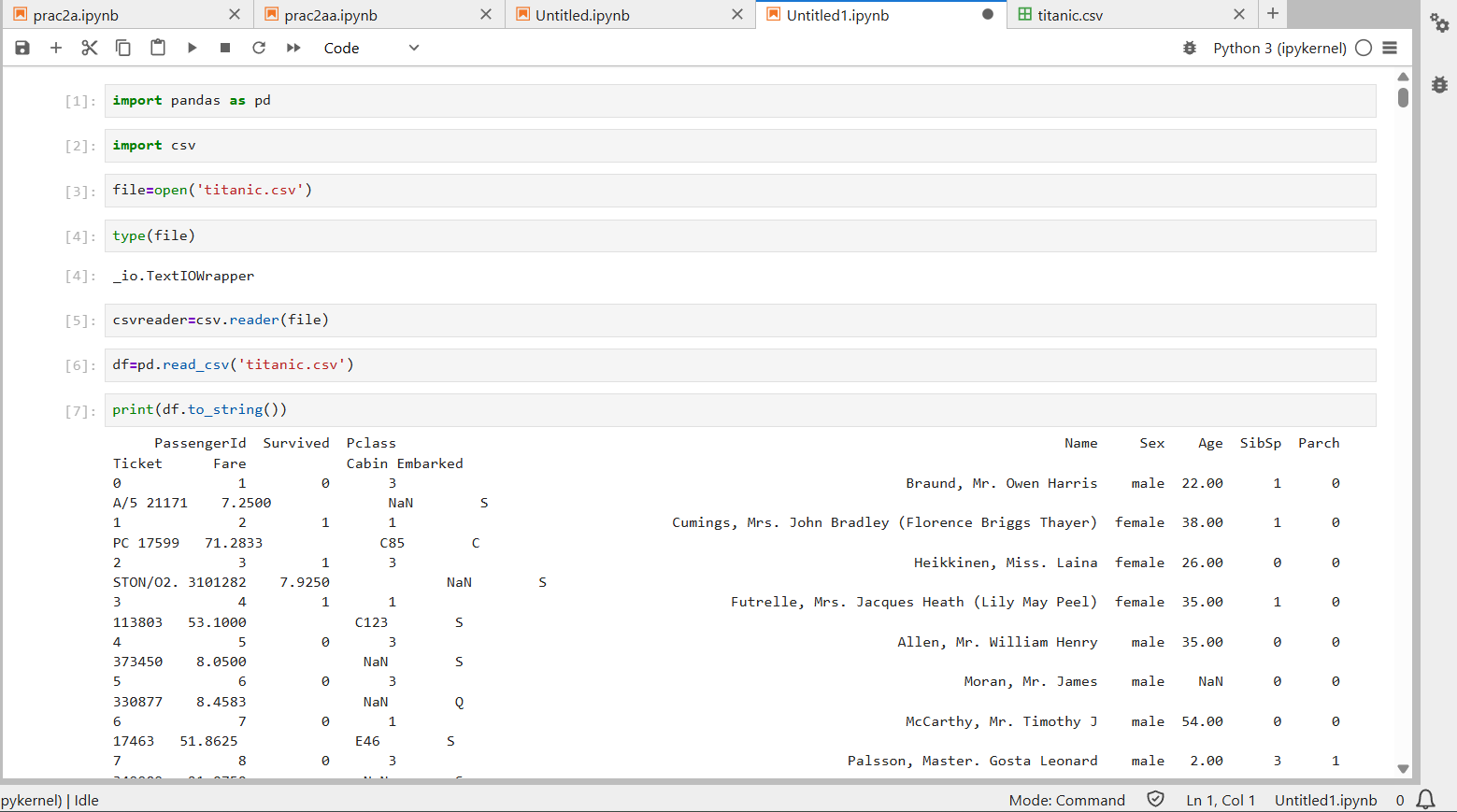
****

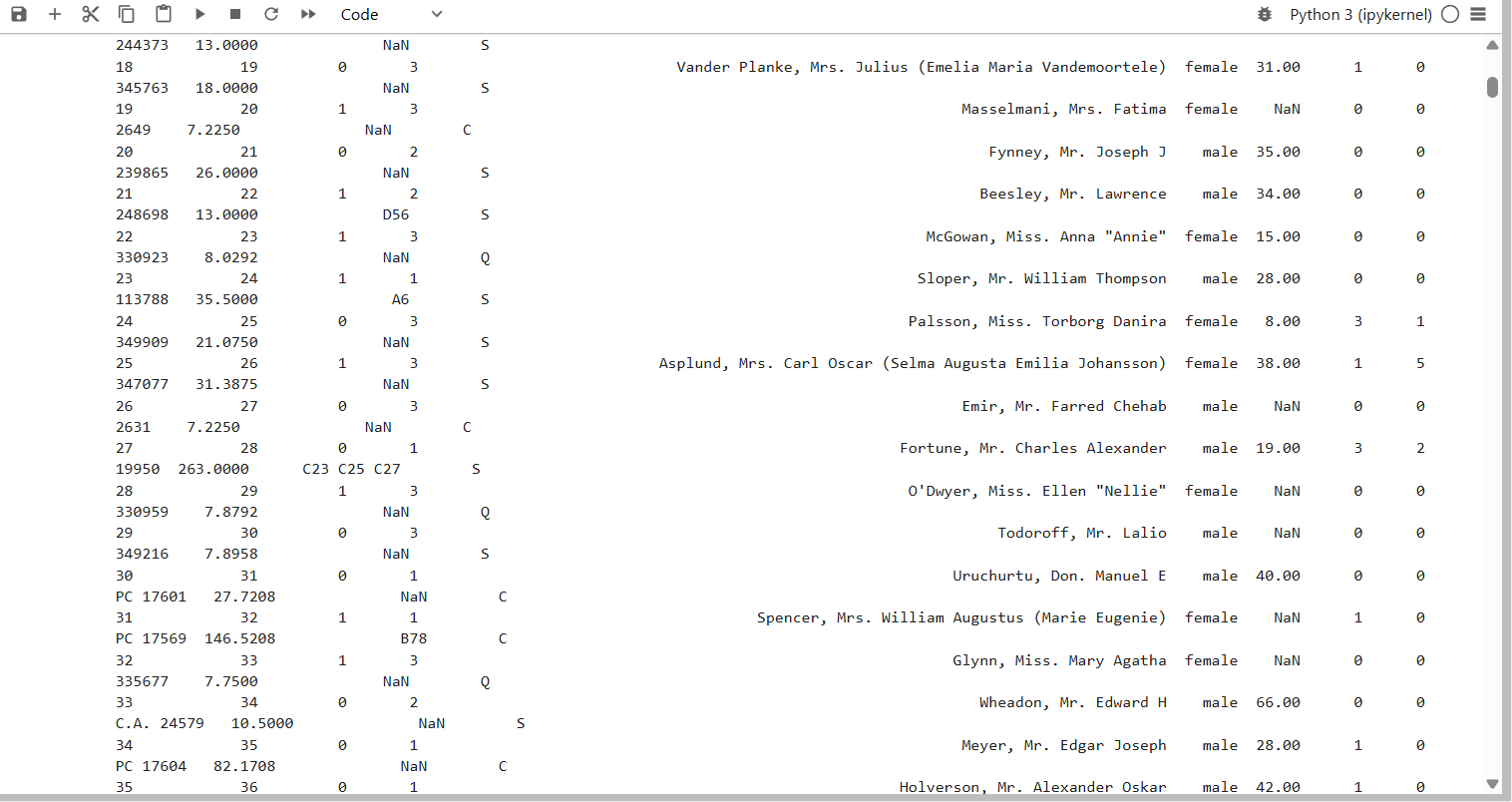
**JSON files into a data frame :**

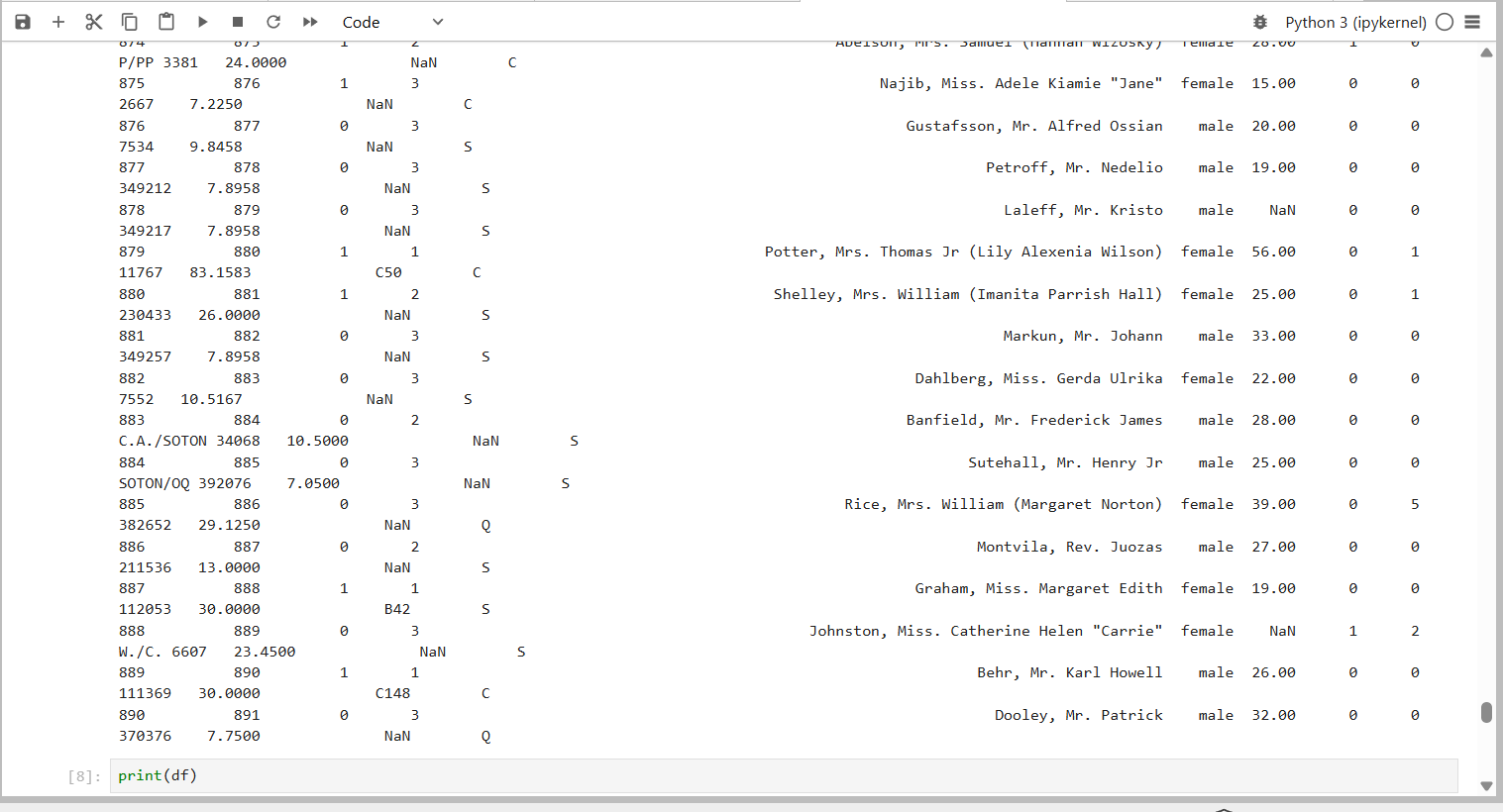
****

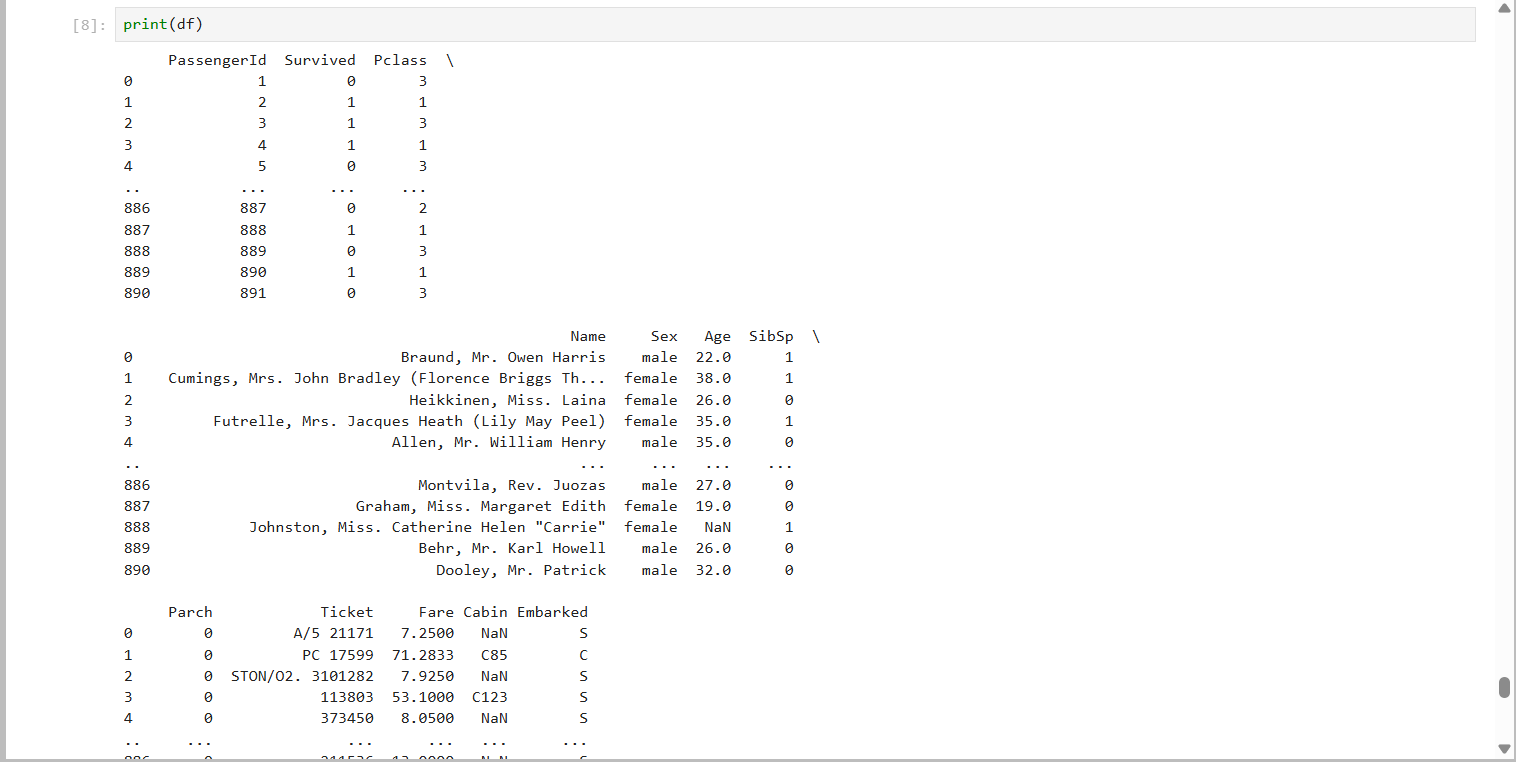
**Aim : Perform basic data pre-processing task such as handling missing value and outliners.**

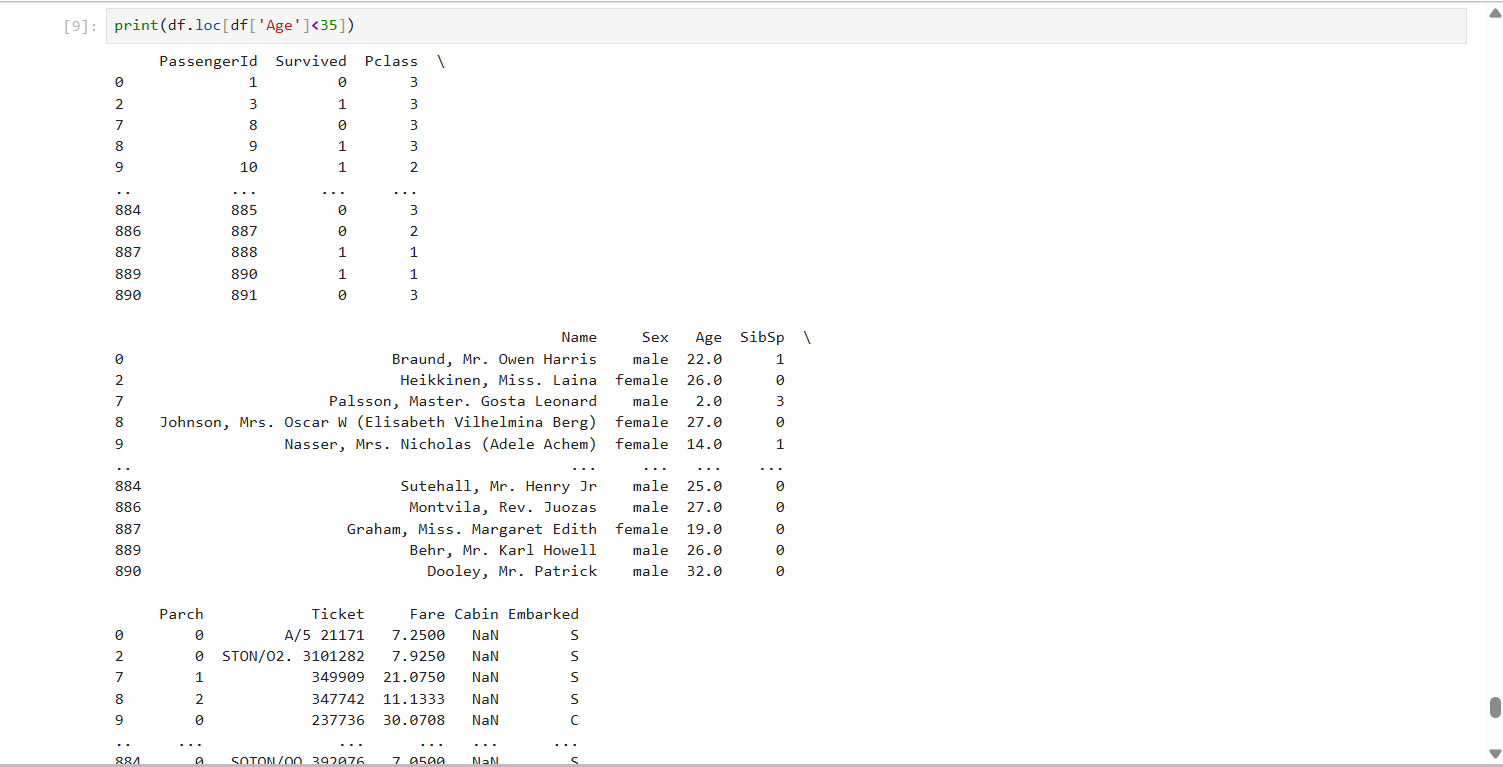
**Download Titanic.csv dataset from browser**

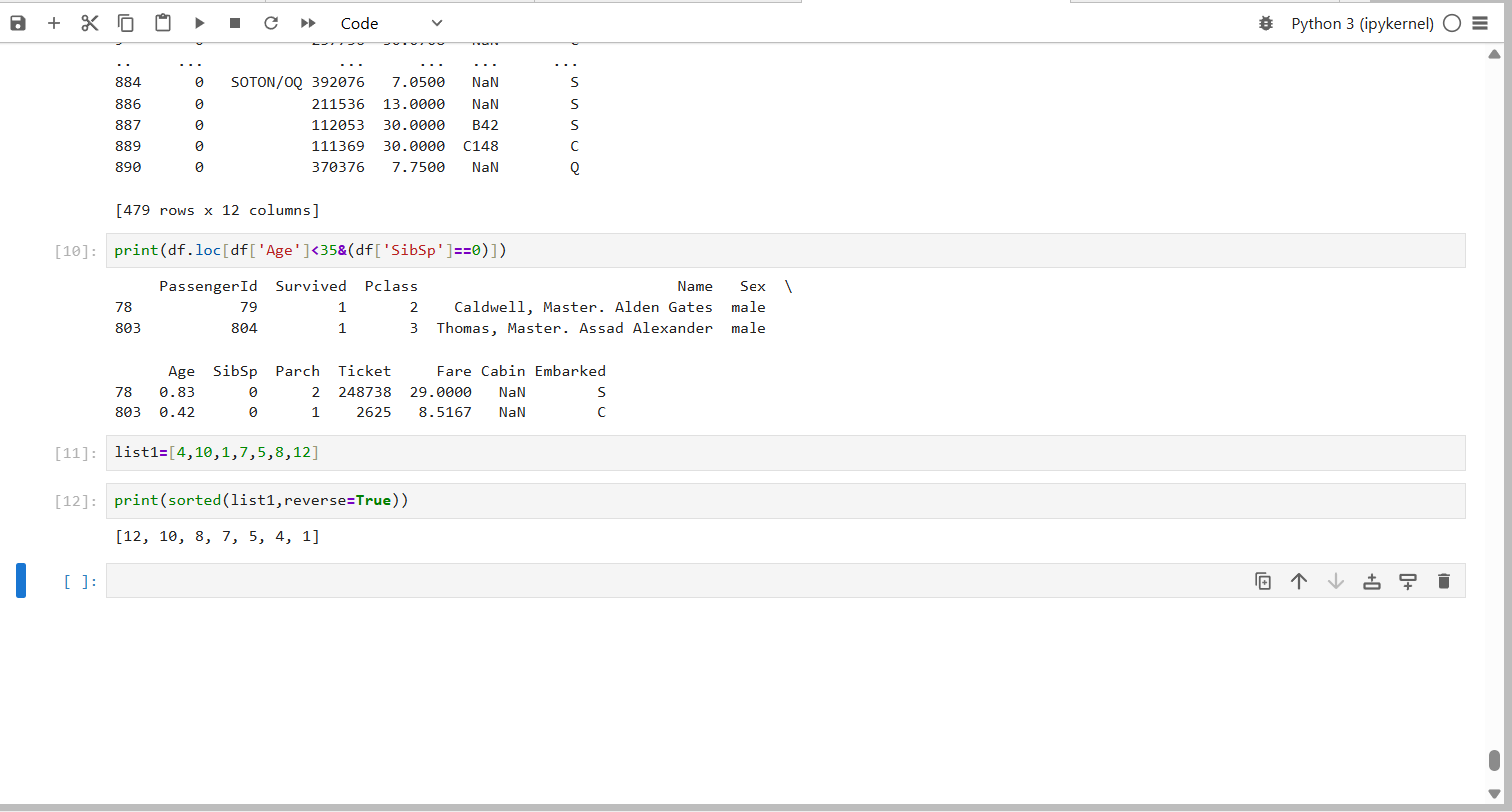
****

****

****

****

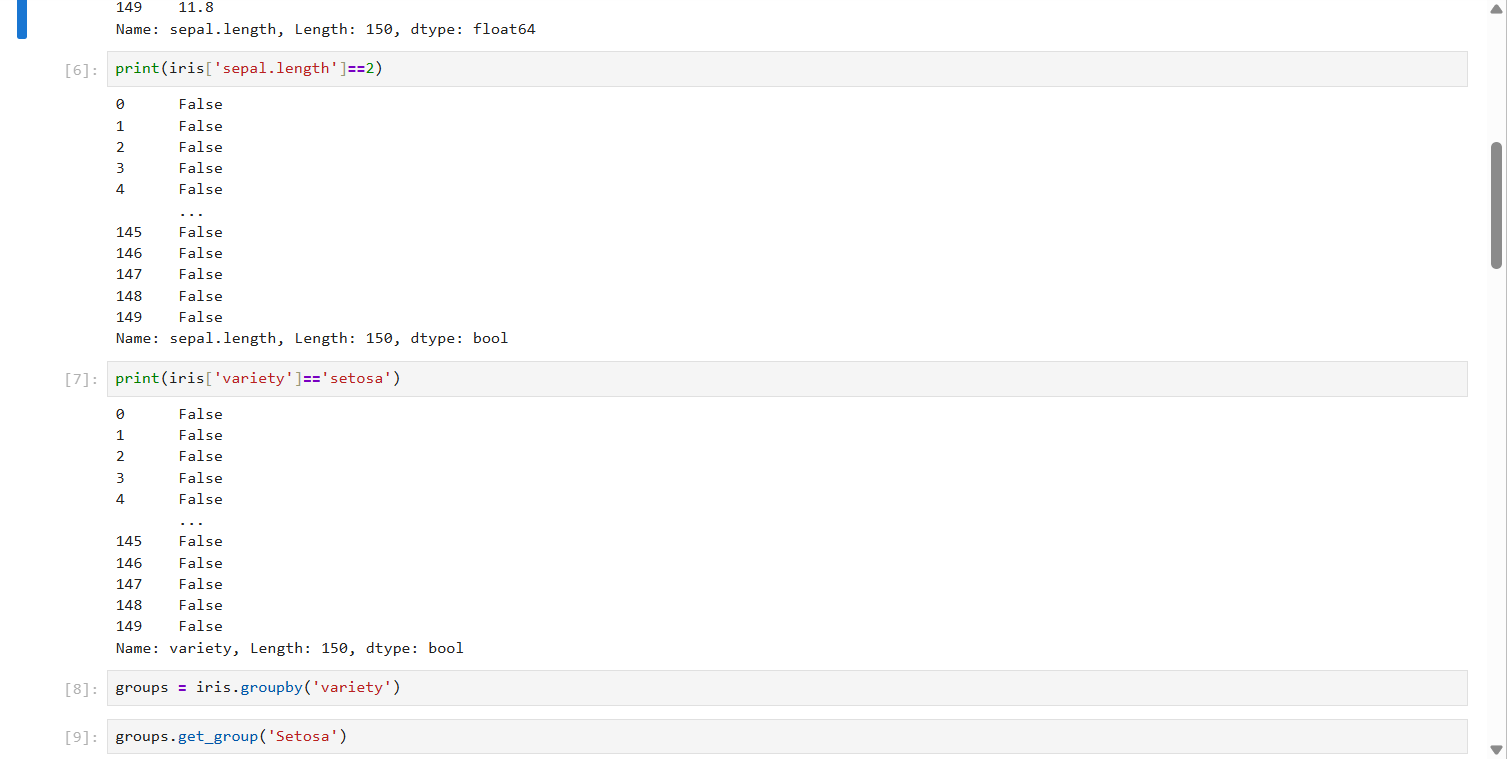
****

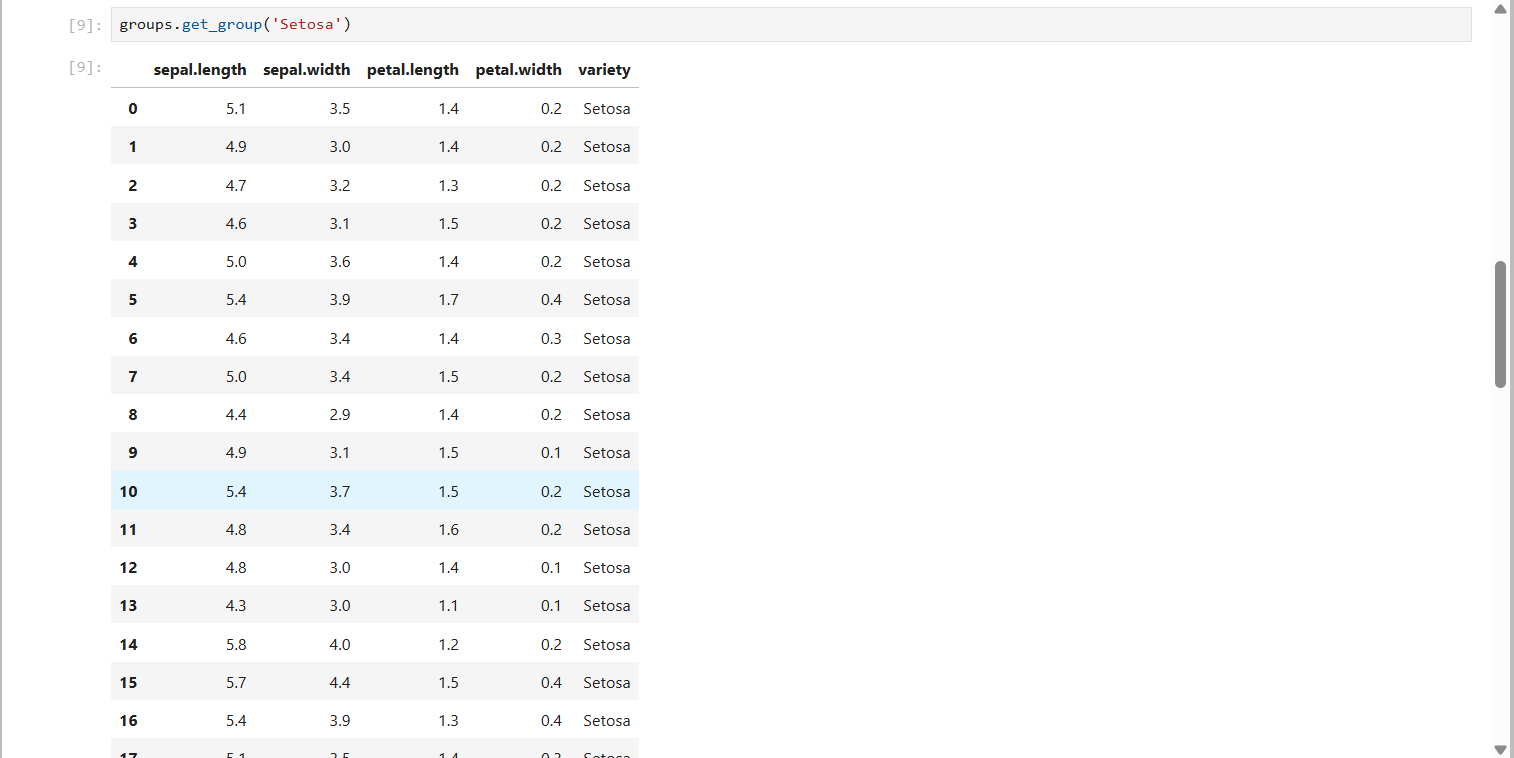
****

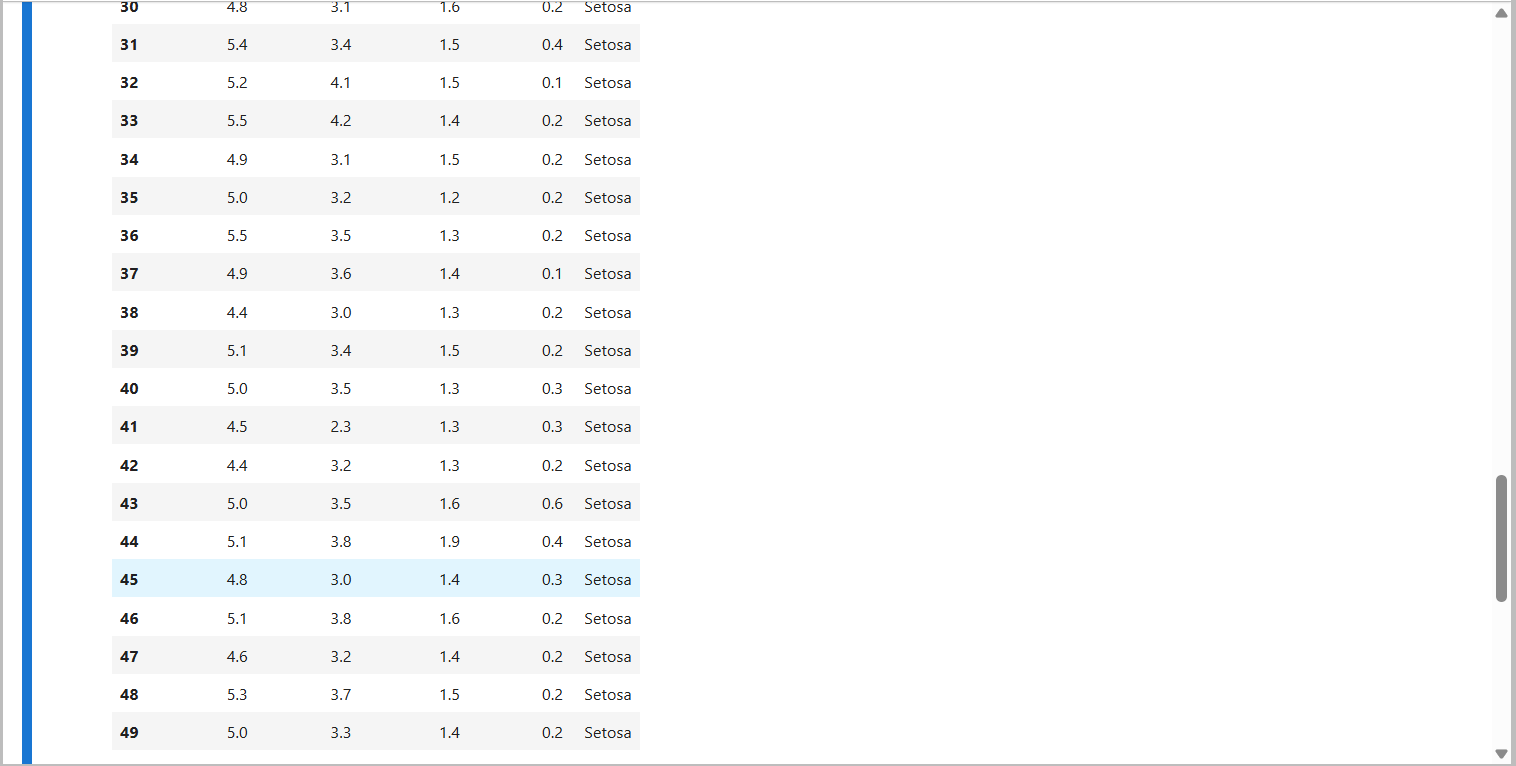
**Aim : Manipulate and transform data using function like filtering , Sorting and grouping.**

**Download iris.csv dataset from browser**

****

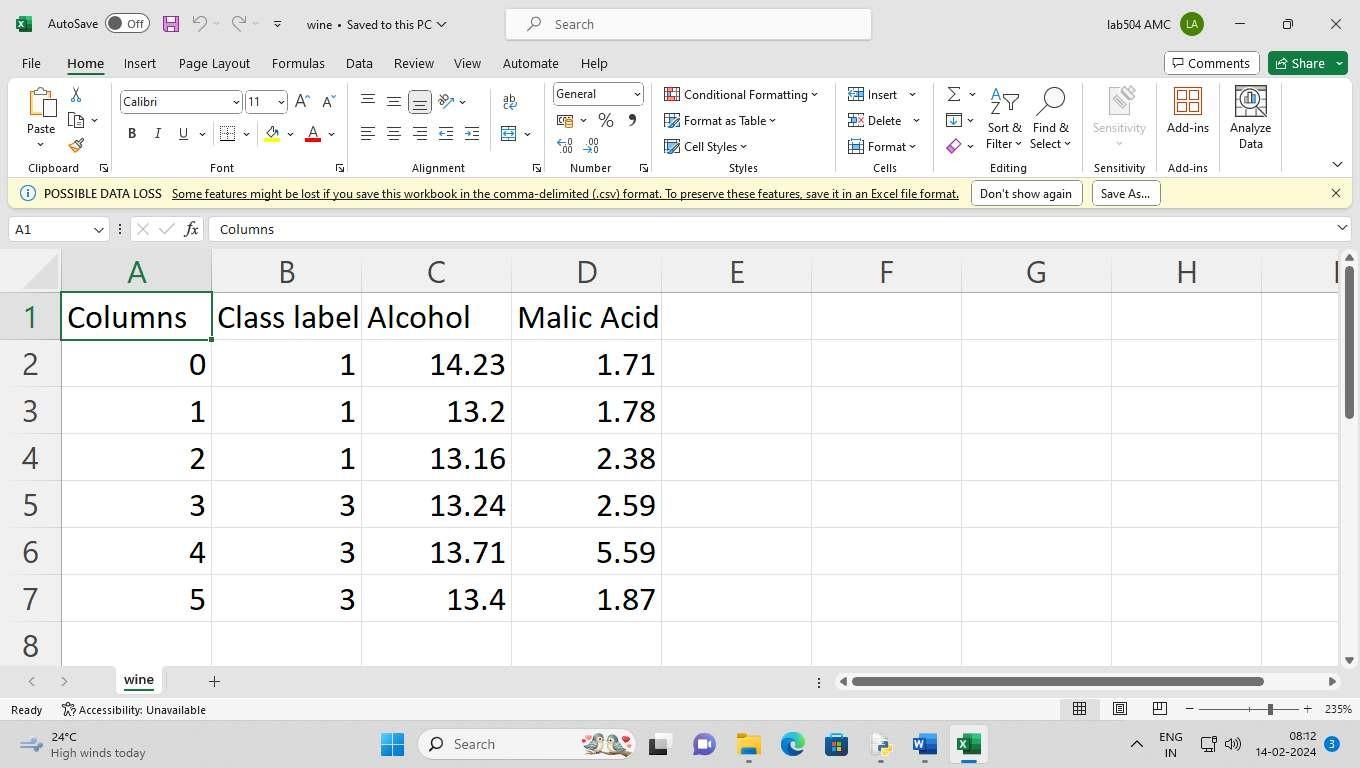
****

****

****

**Practical** **No** **03** **:** **Features** **scaling** **and** **Dummification**

**Practical** **No** **03** **(A):** **Apply** **feature** **scaling** **techniques** **like** **standardization** **normalization** **to** **numerically** **features.**



**save** **it:** **wine.cvs**

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

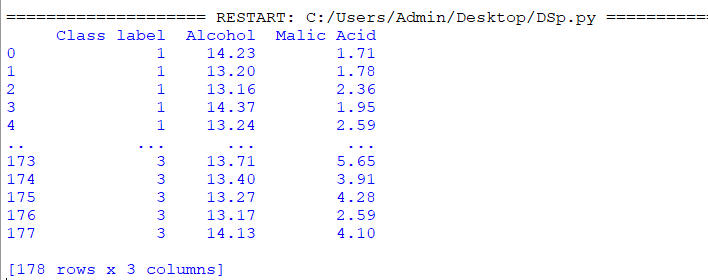
import seaborn as sns

df = pd.read\_csv('wine.csv',header=None, skiprows=1, usecols=[0,1,2])

df.columns=['Class label', 'Alcohol', 'Malic Acid']

print(df)

**output:**



**Code :**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import MinMaxScaler

df = pd.read\_csv('wine.csv',header=None, skiprows=1, usecols=[0,1,2])

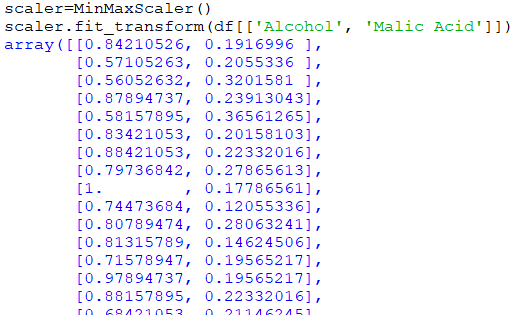
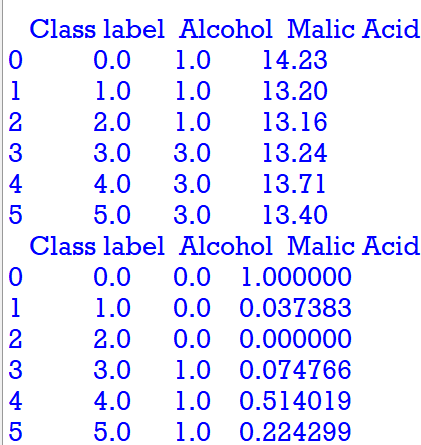
df.columns=['Class label', 'Alcohol', 'Malic Acid']

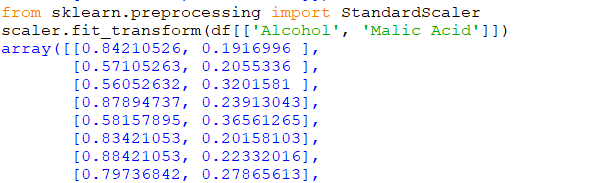
print(df)

scaler=MinMaxScaler()

df[['Alcohol','Malic Acid']] = scaler.fit\_transform(df[['Alcohol', 'Malic Acid']])

**Output :**



****

**Practical** **No** **03** **(B):Perform** **feature** **Dummification** **to** **convert** **categorical** **variables** **into** **numerical** **representations.**

**We** **use** **the** **Iris** **data** **set** **as** **it** **has** **categorical** **variables.**

**save** **it:** **iris.csv**



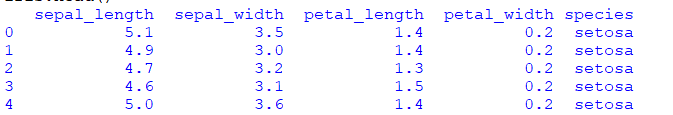
**Code :**

import pandas as pd

iris = pd.read\_csv('iris.csv')

iris.head()

**Output :**

****

**Code :**

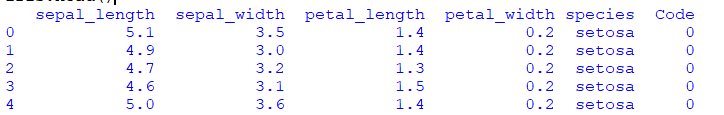
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

iris['Code']=le.fit\_transform(iris.species)

iris.head()

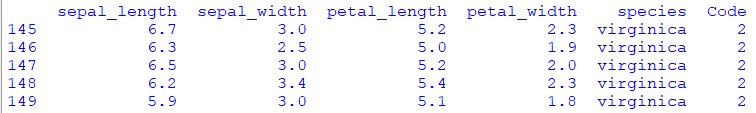
**Output :**

****

**Code :**

iris.tail()

**Output :**

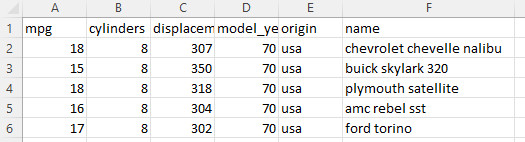


**Practical** **No** **04** **:** **Hypothesis** **Testing**

* To check if there is correlation among the given two categorical variables we apply the chi square test.
* Assumption for chi square test.
* Observations in each sample are independent and identically distributed
* Expected value of cells should be 5 or greater in at least 80% of cell
* The two variables should be categorical variables.

**Practical** **No** **04** **(A)** **:Formulate** **null** **and** **alternative** **hypotheses** **for** **a** **given** **problem**

**save** **it:** **mpg.csv**



**Code :**

import pandas as pd

import numpy as np

import seaborn as sb

import matplotlib.pyplot as plt

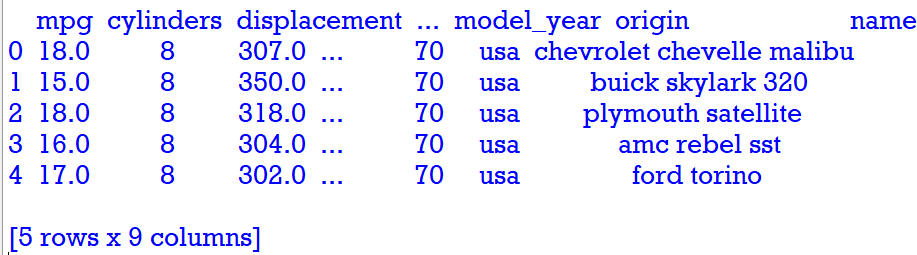
import warnings

warnings.filterwarnings('ignore')

df = sb.load\_dataset('mpg')

df.head()

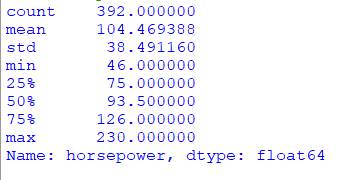
**Output :**



**#Convert** **horsepower** **and** **model** **year** **into** **numerical** **value.**

df['horsepower'].describe()

**Output :**

****

df['model\_year'].describe()

**Output :**

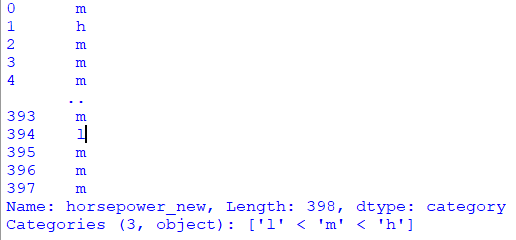


bins=[0,75,150,240]

df['horsepower\_new']=pd.cut(df['horsepower'],bins=bins,labels=['l','m','h'])

df['horsepower\_new']

**Output :**



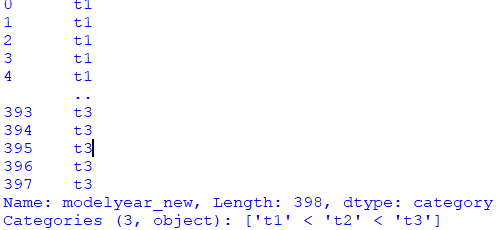
ybins=[69,72,74,84]

label=['t1','t2','t3']

df['modelyear\_new']=pd.cut(df['model\_year'],bins=ybins,labels=label)

df['modelyear\_new']

Output :



#create a matrix using crosstabs which will be giving us the frequencies which is occurring in both the classes

df\_chi=pd.crosstab(df['horsepower\_new'],df['modelyear\_new']) df\_chi

print(df\_chi)

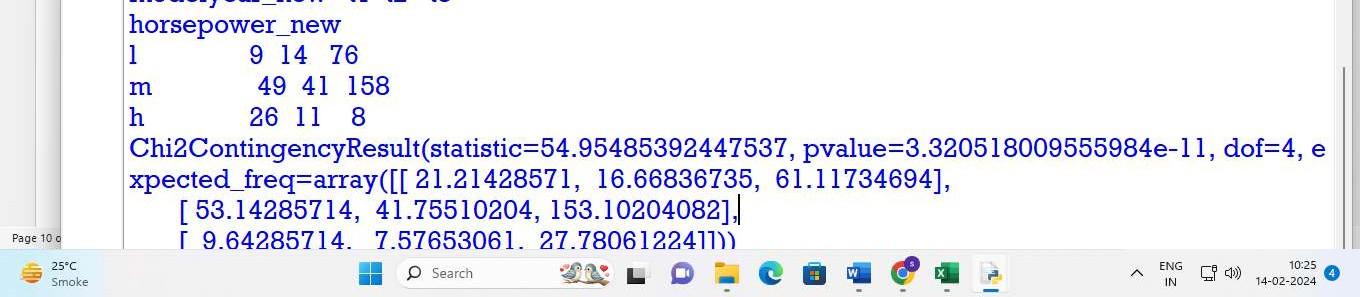
**output:**



Will create null and alternative hypotheses for our chi square test.

* H0 - variables are independent if p-> value> 0.05.
* H1- there is dependency in both the variables if p-value <0.05.

from scipy import stats stats.chi2\_contingency(df\_chi) print(stats.chi2\_contingency(df\_chi)) **output:**



* There is a correlation between the both variables, i.e. there is dependency between these two variables. The manufacturing of model and horsepower of model and it has got correlation as year changes the horsepower also changes.
* P value is less than level of significance null hypothesis is rejected.

**Code :**

import numpy as np

import scipy as scipy

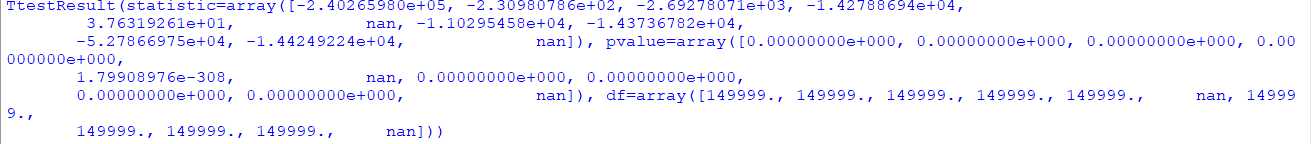
from scipy import stats

import pandas as pd

data1=pd.read\_csv('data.csv')

stats.ttest\_1samp(data1,155)

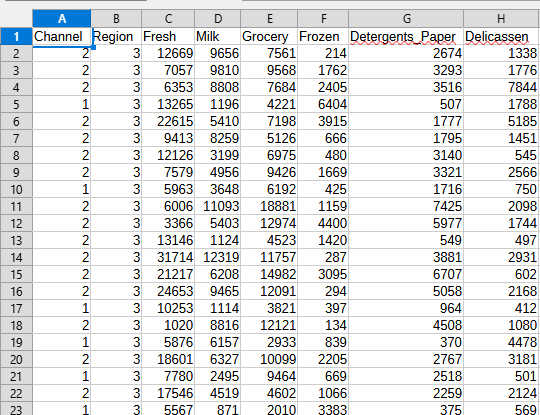
**Output :**

****

**Practical** **No** **05** **:** **ANOVA** **(Analysis** **of** **Variance)**

* **Practical** **No** **05** **(A)** **:Perform** **one-way** **ANOVA** **to** **compare** **means** **across** **multiple** **groups.**
* **Practical** **No** **05** **(B)** **:Conduct** **post-hoc** **tests** **to** **identify** **significant** **differences** **between** **group** **means.**

**Wholesale customers data.csv :**

****

**R Code :**

**> crop.data <- read.csv("C:\\Users\\admin\\Desktop\\wholesale customers data.csv")**

**> head(crop.data)**

**Channel Region Fresh Milk Grocery Frozen Detergents\_Paper Delicassen**

**1 2 3 12669 9656 7561 214 2674 1338**

**2 2 3 7057 9810 9568 1762 3293 1776**

**3 2 3 6353 8808 7684 2405 3516 7844**

**4 1 3 13265 1196 4221 6404 507 1788**

**5 2 3 22615 5410 7198 3915 1777 5185**

**6 2 3 9413 8259 5126 666 1795 1451**

**> str(crop.data)**

**'data.frame': 440 obs. of 8 variables:**

**$ Channel : int 2 2 2 1 2 2 2 2 1 2 ...**

**$ Region : int 3 3 3 3 3 3 3 3 3 3 ...**

**$ Fresh : int 12669 7057 6353 13265 22615 9413 12126 7579 5963 6006 ...**

**$ Milk : int 9656 9810 8808 1196 5410 8259 3199 4956 3648 11093 ...**

**$ Grocery : int 7561 9568 7684 4221 7198 5126 6975 9426 6192 18881 ...**

**$ Frozen : int 214 1762 2405 6404 3915 666 480 1669 425 1159 ...**

**$ Detergents\_Paper: int 2674 3293 3516 507 1777 1795 3140 3321 1716 7425 ...**

**$ Delicassen : int 1338 1776 7844 1788 5185 1451 545 2566 750 2098 …**

**> names (crop.data)**

**[1] "Channel" "Region" "Fresh" "Milk"**

**[5] "Grocery" "Frozen" "Detergents\_Paper" "Delicassen"**

**> one.way <- aov (Milk ~ Channel, data = crop.data)**

**> summary (one.way)**

**Df Sum Sq Mean Sq F value Pr(>F)**

**Channel 1 5.076e+09 5.076e+09 118 <2e-16 \*\*\***

**Residuals 438 1.884e+10 4.301e+07**

**---**

**Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1**

**> Fit <- aov(formula = Milk ~ Channel, data = crop.data)**

**> summary (Fit)**

**Df Sum Sq Mean Sq F value Pr(>F)**

**Channel 1 5.076e+09 5.076e+09 118 <2e-16 \*\*\***

**Residuals 438 1.884e+10 4.301e+07**

**---**

**Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1**

**> crop.data$Channel <- as.factor(crop.data$Channel)**

**> Fit <- aov(Milk ~ Channel, data = crop.data)**

**> summary(Fit)**

**Df Sum Sq Mean Sq F value Pr(>F)**

**Channel 1 5.076e+09 5.076e+09 118 <2e-16 \*\*\***

**Residuals 438 1.884e+10 4.301e+07**

**---**

**Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1**

**> tukey\_result <- TukeyHSD(Fit)**

**> print(tukey\_result)**

**Tukey multiple comparisons of means**

**95% family-wise confidence level**

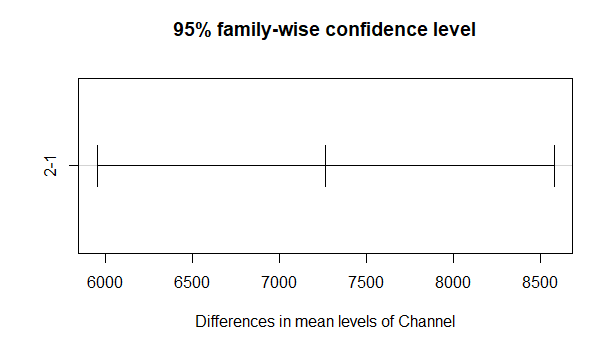
**Fit: aov(formula = Milk ~ Channel, data = crop.data)**

**$Channel**

**diff lwr upr p adj**

**2-1 7264.775 5950.494 8579.056 0**

**> plot(tukey\_result)**



**Practical** **No** **06** **:** **Regression** **and Its Type**

* **Implement simple linear regression using a dataset.**
* **Explore and interpret the regression model coefficients and goodness-of-fit measures.**
* **Extend the analysis to multiple linear regression and assess the impact of additional predictors.**

**R Code :**

**> LungCapData <- read.table(file.choose(),sep = ",",header = TRUE)**

**> attach(LungCapData)**

**> names(LungCapData)**

**[1] "Channel"**

**[2] "Region"**

**[3] "Fresh"**

**[4] "Milk"**

**[5] "Grocery"**

**[6] "Frozen"**

**[7] "Detergents\_Paper"**

**[8] "Delicassen"**

**> class(Fresh)**

**[1] "integer"**

**> class(Milk)**

**[1] "integer"**

**> class(Grocery)**

**[1] "integer"**

**> cor(Region,Channel)**

**[1] 0.06202762**

**> mod <- lm(Channel~Region)**

**> summary(mod)**

**Call:**

**lm(formula = Channel ~ Region)**

**Residuals:**

**Min 1Q Median 3Q**

**-0.3399 -0.3399 -0.3024 0.6601**

**Max**

**0.7351**

**Coefficients:**

**Estimate Std. Error**

**(Intercept) 1.22737 0.07663**

**Region 0.03750 0.02883**

**t value Pr(>|t|)**

**(Intercept) 16.016 <2e-16 \*\*\***

**Region 1.301 0.194**

**---**

**Signif. codes:**

**0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05**

**‘.’ 0.1 ‘ ’ 1**

**Residual standard error: 0.4677 on 438 degrees of freedom**

**Multiple R-squared: 0.003847, Adjusted R-squared: 0.001573**

**F-statistic: 1.692 on 1 and 438 DF, p-value: 0.1941**

| **> plot(Region,Channel,**  **+ main = "Scatterplot of Channel vs Region",**  **+ xlab = "Region",**  **+ ylab = "Channel",**  **+ col = "blue",**  **+ pch = 16)**  **> mod <- lm(Channel ~ Region)**  **> abline(mod,col = 'red',lwd = 2)**  **> library(ggplot2)**  **> ggplot(LungCapData,aes(x=Region, y = Channel))+**  **+ geom\_point(color = 'blue')+**  **+ geom\_smooth(method = "lm",color = 'red',se = FALSE)+**  **+ labs(title = "Scatterplot of channel vs Region", x = "Region",y = "Channel")+**  **+ theme\_minimal()**  **`geom\_smooth()` using formula = 'y ~**  **x'**    **> LungCapData$Channel <- as.factor(LungCapData$Channel)**  **> ggplot(LungCapData,aes(x=Region, y=as.numeric(Channel)))+**  **+ geom\_jitter(color = 'blue',width = 0.2)+**  **+ labs(title = "Scatterplot of Channel vs Region",x="Region",y="Channel")+**  **+ theme\_minimal()** |
| --- |
|  |
|  |

**Practical** **No** **07** **:** **Logistic** **Regression** **and** **Decision** **Tree**

* **Build a logistic regression and Decision Tree**
* **Evaluate the model’s performance using classification metrics (e.g., accuracy, precision, recall).**
* **Construct a Decision tree model and interpret the decision rules for classification**

**Code :**

**> library(datasets)**

**> head(ir\_data)**

**Sepal.Length Sepal.Width Petal.Length Petal.Width Species**

**1 5.1 3.5 1.4 0.2 setosa**

**2 4.9 3.0 1.4 0.2 setosa**

**3 4.7 3.2 1.3 0.2 setosa**

**4 4.6 3.1 1.5 0.2 setosa**

**5 5.0 3.6 1.4 0.2 setosa**

**6 5.4 3.9 1.7 0.4 setosa**

**> library(datasets)**

**> ir\_data <- iris**

**> head(ir\_data)**

**Sepal.Length Sepal.Width Petal.Length Petal.Width Species**

**1 5.1 3.5 1.4 0.2 setosa**

**2 4.9 3.0 1.4 0.2 setosa**

**3 4.7 3.2 1.3 0.2 setosa**

**4 4.6 3.1 1.5 0.2 setosa**

**5 5.0 3.6 1.4 0.2 setosa**

**6 5.4 3.9 1.7 0.4 setosa**

**> str(ir\_data)**

**'data.frame': 150 obs. of 5 variables:**

**$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...**

**$ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...**

**$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...**

**$ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...**

**$ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 …**

**> levels(ir\_data$Species)**

**[1] "setosa" "versicolor" "virginica"**

**> sum(is.na(ir\_data))**

**[1] 0**

**> ir\_data <- ir\_data[1:100,]**

**> set.seed(100)**

**> samp <- sample(1:100,80)**

**> ir\_test <- ir\_data[samp,]**

**> ir\_ctrl <- ir\_data[-samp,]**

**> library(ggplot2)**

**> library(GGally)**

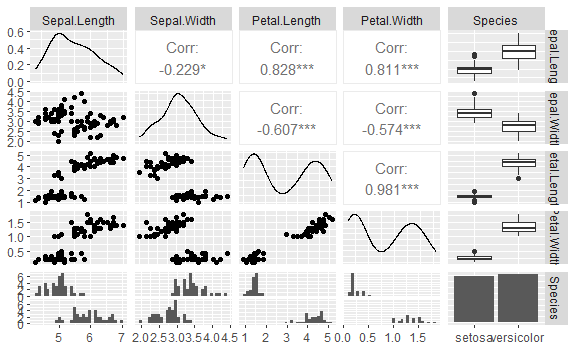
**> ggpairs(ir\_test)**

**plot: [5, 1] [===================================>-------] 84% est: 0s `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.**

**plot: [5, 2] [=====================================>-----] 88% est: 0s `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.**

**plot: [5, 3] [=======================================>---] 92% est: 0s `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.**

**plot: [5, 4] [========================================>--] 96% est: 0s `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.**

****

**> y <- ir\_test$Species;x<- ir\_test$Sepal.Length**

**> glfit <- glm(y ~ x,family = "binomial")**

**> summary(glfit)**

**Call:**

**glm(formula = y ~ x, family = "binomial")**

**Coefficients:**

**Estimate Std. Error z value Pr(>|z|)**

**(Intercept) -27.500 5.934 -4.634 3.59e-06 \*\*\***

**x 5.112 1.109 4.611 4.01e-06 \*\*\***

**---**

**Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1**

**(Dispersion parameter for binomial family taken to be 1)**

**Null deviance: 110.854 on 79 degrees of freedom**

**Residual deviance: 48.818 on 78 degrees of freedom**

**AIC: 52.818**

**Number of Fisher Scoring iterations: 6**

**> newdata <- data.frame(x= ir\_ctrl$Sepal.Length)**

**> predict\_val <- predict(glfit,newdata,type = "response")**

**> prediction <- data.frame(ir\_ctrl$Sepal.Length,ir\_ctrl$Species,predict\_val)**

**> prediction**

**ir\_ctrl.Sepal.Length ir\_ctrl.Species predict\_val**

**1 5.4 setosa 0.52665832**

**2 5.0 setosa 0.12584710**

**3 4.8 setosa 0.04923563**

**4 5.4 setosa 0.52665832**

**5 5.7 setosa 0.83759291**

**6 4.9 setosa 0.07948111**

**7 5.5 setosa 0.64975559**

**8 5.1 setosa 0.19357325**

**9 4.5 setosa 0.01104861**

**10 5.0 setosa 0.12584710**

**11 5.3 setosa 0.40023260**

**12 6.9 versicolor 0.99958015**

**13 5.7 versicolor 0.83759291**

**14 5.2 versicolor 0.28582944**

**15 5.6 versicolor 0.75569041**

**16 5.6 versicolor 0.75569041**

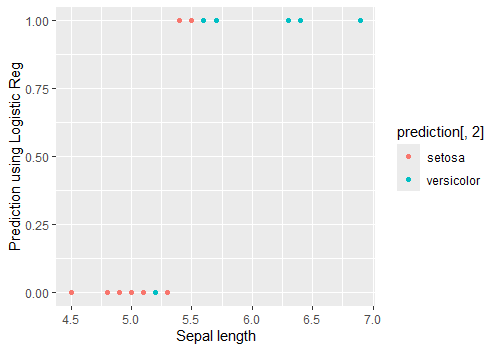
**17 6.3 versicolor 0.99105619**

**18 6.4 versicolor 0.99461661**

**19 5.7 versicolor 0.83759291**

**20 5.7 versicolor 0.83759291**

**> qplot(prediction[,1],round(prediction[,3]),col = prediction[,2],xlab = 'Sepal length',ylab = 'Prediction using Logistic Reg')**



* **So, from the above plot, we can see that our simple model is doing a Sor frogmod prediction for plant species. We can also see a blue dot in the bottom cluster.**
* **This blue dot is showing that although correct specie of this plant is 'versicolor' but our model is predicting it as 'setosa'.**

**Practical** **No** **08** **:** **K-Means Clustering**

* **Apply the K-Means algorithm to group similar data points into clusters.**
* **Determine the optimal number of clusters using elbow method or silhouette analysis.**
* **Visualize the clustering results and analyze the cluster characteristics.**

**Code :**

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

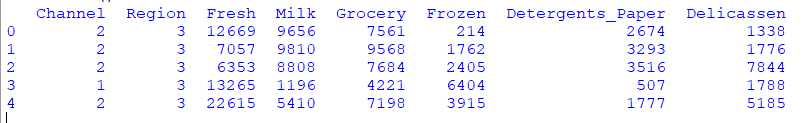
from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

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

data.head()

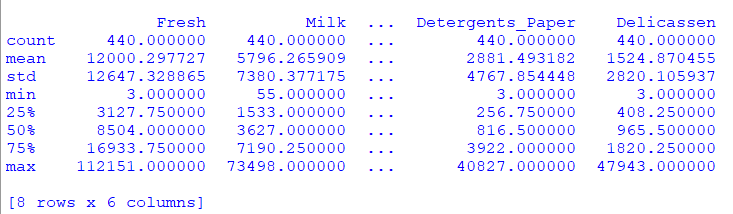
**Output :**

****

categorical\_features = ['Channel', 'Region ']

continuous\_features = ['Fresh', 'Milk', 'Grocery', 'Frozen','Detergents\_Paper', 'Delicassen']

data[continuous\_features].describe()



for col in categorical\_features:

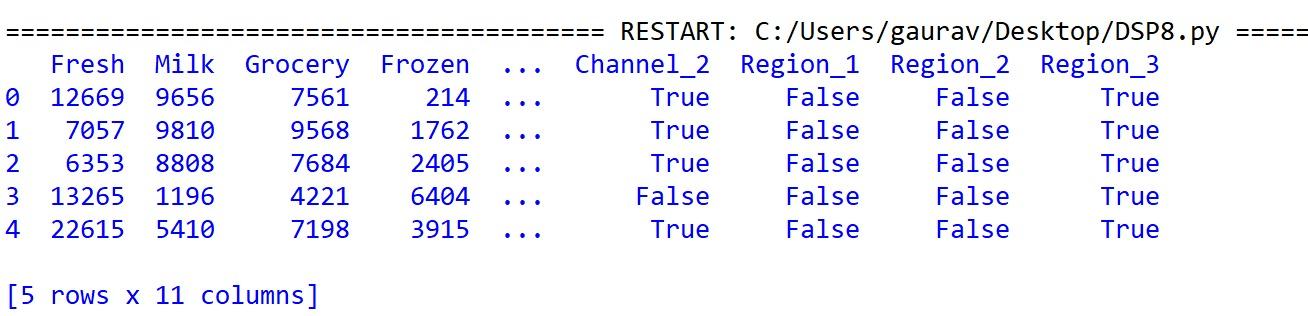
dummies = pd.get\_dummies (data [col], prefix=col)

data = pd.concat([data, dummies], axis=1)

data.drop(col, axis=1, inplace=True)

data.head()

**Output :**

****

**Code :**

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

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

scaler = MinMaxScaler()

data\_transformed = scaler.fit\_transform(data.select\_dtypes (include=['float64', 'int64']))

Sum\_of\_squared\_distances = []

K = range(1,15)

for k in K:

km = KMeans (n\_clusters=k, random\_state=42)

km.fit(data\_transformed)

Sum\_of\_squared\_distances.append(km.inertia\_)

plt.plot(K, Sum\_of\_squared\_distances, 'bx-')

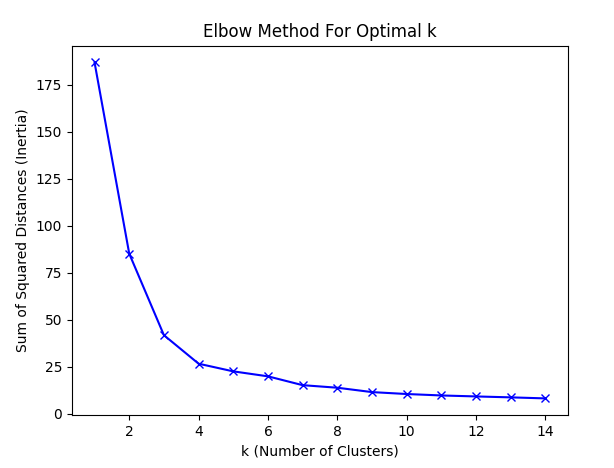
plt.xlabel('k (Number of Clusters)')

plt.ylabel('Sum of Squared Distances (Inertia)')

plt.title('Elbow Method For Optimal k')

plt.show()

**Output :**

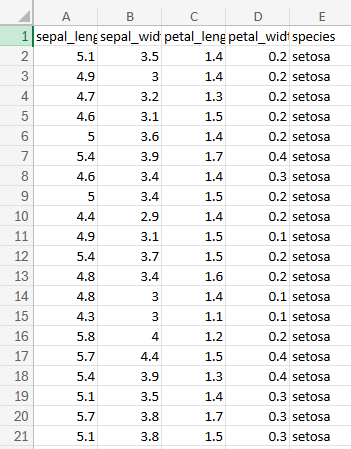
****

* **In the plot above the elbow is at k=5 indicating the optimal K for this dataset is 5**

**Practical 09 : Principal Component Analysis (PCA)**

* **Perform PCA on a dataset to reduce dimensionality.**
* **Evaluate the explained variance and select the appropriate number of principal components**
* **Visualize the data in the reduced-dimensional space**

**use iris.csv:**

****

**> data\_iris <- iris[1:4]**

**> Cov\_data <-cov(data\_iris)**

**> Eigen\_data<-eigen(Cov\_data)**

**> Eigen\_data$values**

**[1] 4.22824171 0.24267075 0.07820950 0.02383509**

**> PCA\_data <- princomp(data\_iris,cor = "False")**

**> PCA\_data$sdev^2**

**Comp.1 Comp.2 Comp.3 Comp.4**

**4.20005343 0.24105294 0.07768810 0.02367619**

**> PCA\_data$loadings[,1:4]**

**Comp.1 Comp.2 Comp.3 Comp.4**

**Sepal.Length 0.36138659 0.65658877 0.58202985 0.3154872**

**Sepal.Width -0.08452251 0.73016143 -0.59791083 -0.3197231**

**Petal.Length 0.85667061 -0.17337266 -0.07623608 -0.4798390**

**Petal.Width 0.35828920 -0.07548102 -0.54583143 0.7536574**

**> Eigen\_data$vectors**

**[,1] [,2] [,3] [,4]**

**[1,] 0.36138659 -0.65658877 0.58202985 0.3154872**

**[2,] -0.08452251 -0.73016143 -0.59791083 -0.3197231**

**[3,] 0.85667061 0.17337266 -0.07623608 -0.4798390**

**[4,] 0.35828920 0.07548102 -0.54583143 0.7536574**

**> summary(PCA\_data)**

**Importance of components:**

**Comp.1 Comp.2 Comp.3**

**Standard deviation 2.0494032 0.49097143 0.27872586**

**Proportion of Variance 0.9246187 0.05306648 0.01710261**

**Cumulative Proportion 0.9246187 0.97768521 0.99478782**

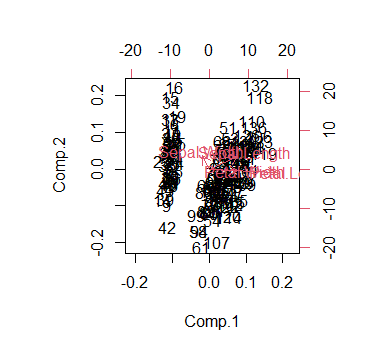
**Comp.4**

**Standard deviation 0.153870700**

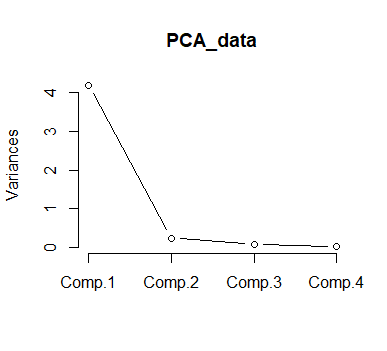
**Proportion of Variance 0.005212184**

**Cumulative Proportion 1.000000000**

**> biplot(PCA\_data)**



**> screeplot(PCA\_data,type="lines")**



**> library(class)**

**> install.packages("e1071")**

**> library(e1071)**

**> modl<-naiveBayes(iris[,1:4], iris[,5])**

**> table(predict(modl, iris[,1:4]), iris[,5])**

**setosa versicolor virginica**

**setosa 50 0 0**

**versicolor 0 47 3**

**virginica 0 3 47**

**Practical No. 10 : Data Visualization and Storytelling**

* **Create meaningful visualizations using data visualization tools.**
* **Combine multiple visualizations to tell a compelling data story.**
* **Present the findings and insights in a clear and concise manner.**

**Solution:**

**Consider an example scenario where we analyze customer churn data for a telecommunications company.**

**Step 1: Data Understanding and Preparation.**

**Assuming we have a dataset containing information about customer demographics, services subscribed, and whether they churned or not.**

**Step 2: Exploratory Data Analysis (EDA)**

**Perform exploratory data analysis to understand the characteristics of the dataset and identify patterns, trends, and correlations.**

**Step 3: Creating Meaningful Visualization**

* **Churn Rate by Gender:** A bar chart showing the churn rate by gender.
* **Churn Rate by Age Group:** A histogram or box plot showing the distribution of age among churned and retained customers.
* **Churn Rate by Service Type:** A pie chart or bar chart showing the percentage of churned customers based on the type of services subscribed.
* **Correlation Matrix:** A heatmap showing the correlation between different variables, highlighting factors that are strongly correlated with churn.
* **Customer Tenure vs. Churn:** A scatter plot showing the relationship between customer tenure and churn status.
* **Customer Segmentation:** Use clustering algorithms like K-Means to segment customers based on their behavior and visualize the clusters using a scatter plot with different colors for each cluster.

**Step 4: Combining Visualizations**

**Combine the above visualizations into a dashboard or a series of slides to tell a compelling data story.**

**For example:**

* **Start with an overview of the churn rate and its distribution across different demographics.**
* **Dive deeper into specific factors contributing to churn, such as age, services subscribed, and tenure.**
* **Highlight any correlations or patterns observed in the data.**
* **Conclude with actionable insights and recommendations for reducing churn.**

**Step 5: Presenting Findings**

**Present the findings and insights in a clear and concise manner, using storytelling techniques to engage the audience and emphasize key points. Use titles, labels, and annotations to guide the audience through the visualizations and explain the significance of each insight.**

**Code :**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

from sklearn.preprocessing import LabelEncoder

data = {

'gender': ['Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female'],

'age': [25, 40, 35, 50, 28, 45, 33, 38, 22, 55],

'churn': [1, 0, 1, 0, 1, 0, 0, 1, 1, 0],

'service\_type': ['A', 'B', 'A', 'B', 'C', 'A', 'C', 'B', 'C', 'A'],

'tenure': [12, 24, 6, 36, 18, 48, 30, 20, 15, 50],

'feature1': [1.5, 2.3, 3.1, 4.0, 2.7, 3.8, 1.9, 2.6, 3.3, 4.5],

'feature2': [2.1, 3.0, 1.8, 4.2, 3.6, 2.9, 1.5, 3.8, 2.7, 4.0],

'cluster': [0, 1, 0, 2, 1, 0, 1, 2, 1, 2]

}

df = pd.DataFrame(data)

label\_encoders = {}

for col in ['gender', 'service\_type']:

le = LabelEncoder()

df[col] = le.fit\_transform(df[col])

label\_encoders[col] = le

sns.set\_style("whitegrid")

plt.figure(figsize=(6, 4))

sns.countplot(data=df, x='gender', hue='churn')

plt.title('Churn Rate by Gender')

plt.show()

fig = px.histogram(df, x='age', color='churn', nbins=5, histnorm='percent')

fig.update\_layout(title='Churn Rate by Age Group', xaxis\_title='Age', yaxis\_title='% of Customers')

fig.show()

service\_churn = df.groupby('service\_type')['churn'].mean()

plt.figure(figsize=(6, 4))

plt.pie(service\_churn, labels=service\_churn.index, autopct='%1.1f%%', colors=['lightblue', 'orange', 'green'])

plt.title('Churn Rate by Service Type')

plt.show()

correlation\_matrix = df.corr()

plt.figure(figsize=(6, 4))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

plt.title('Correlation Matrix')

plt.show()

fig = px.scatter(df, x='tenure', y='churn', color=df['churn'].astype(str),

labels={'churn': 'Churn (1=Yes, 0=No)'})

fig.update\_layout(title='Customer Tenure vs. Churn', xaxis\_title='Tenure', yaxis\_title='Churn')

fig.show()

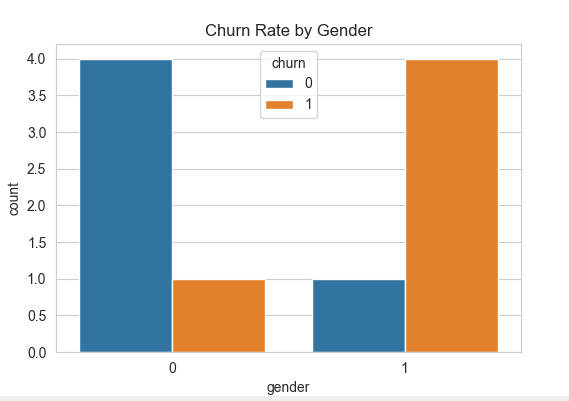
fig = px.scatter(df, x='feature1', y='feature2', color=df['cluster'].astype(str),

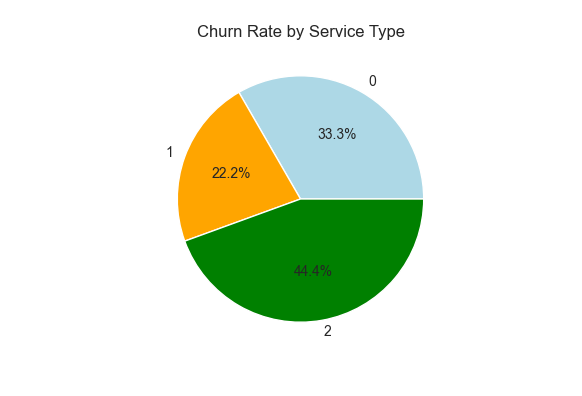
labels={'cluster': 'Customer Segment'})

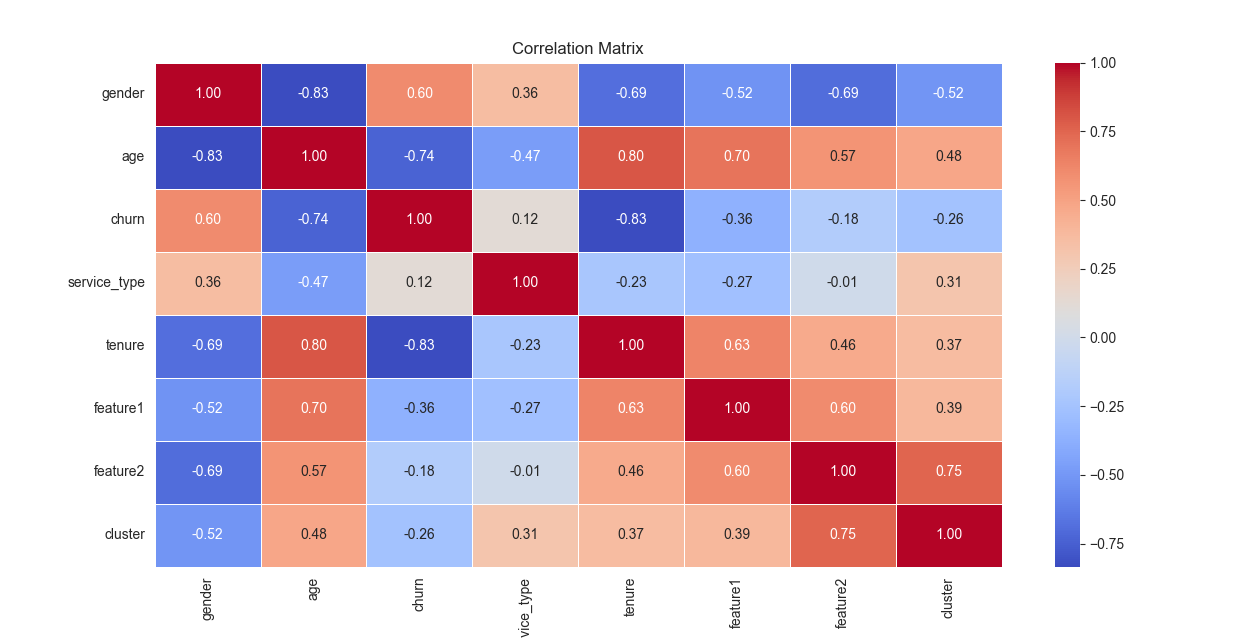
fig.update\_layout(title='Customer Segmentation')

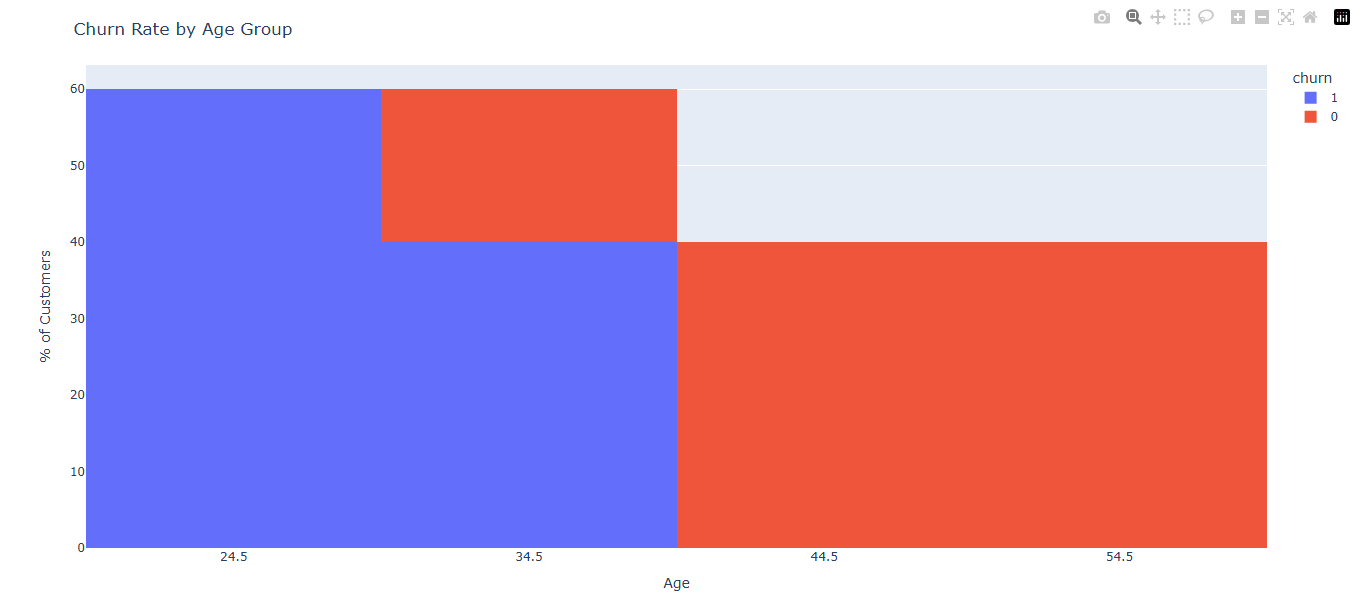
fig.show()

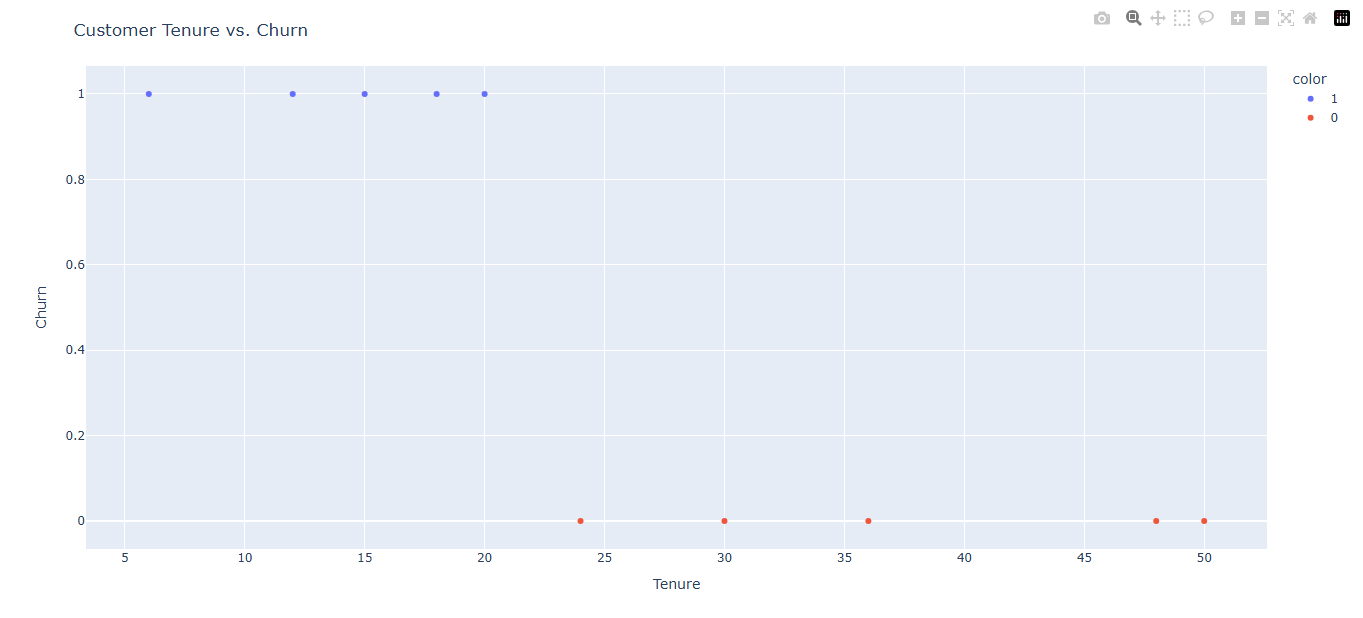
**Output :**

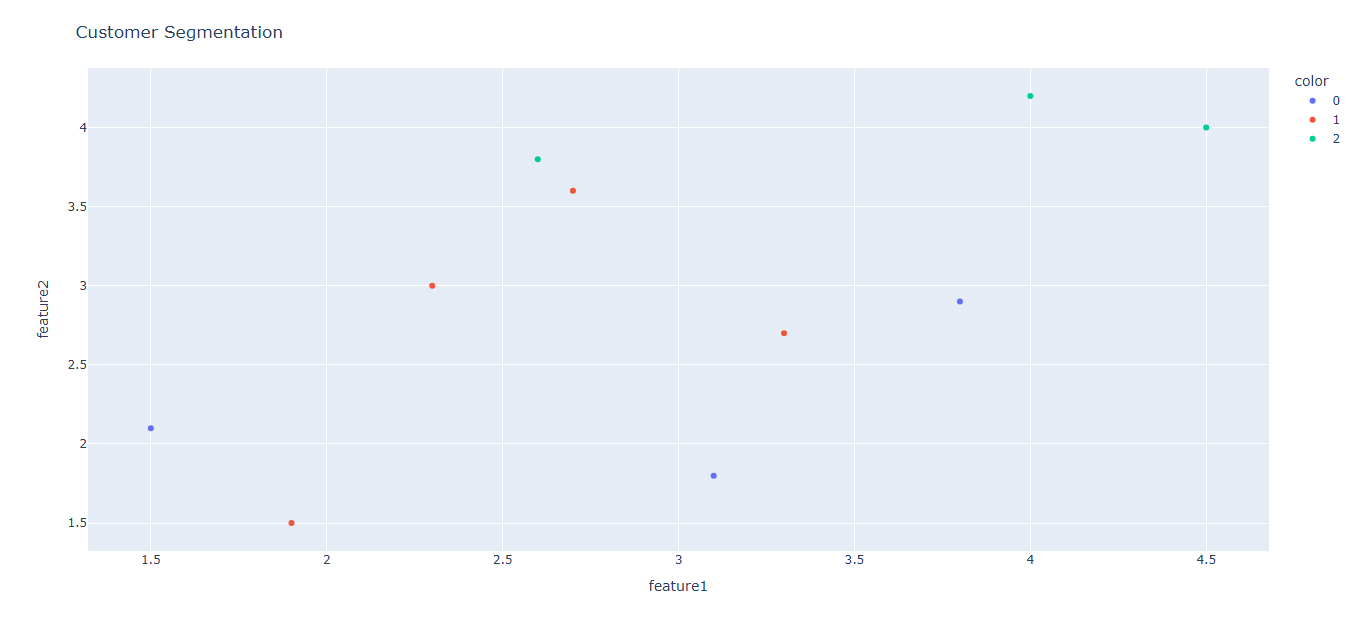
****

****

****

****

****

****