# RESEARCH IN COMPUTING

# INDEX

| Sr. No | Practical Aim | Signature |
|---|---|---|
| **1** | a. Write a program for obtaining descriptive statistics of data.<br>b. Import data from different data sources (from Excel, csv, mysql, sql server, oracle to R/Python/Excel) | |
| **2** | a. Design a survey form for a given case study, collect the primary data and analyze it<br>b. Perform suitable analysis of given secondary data. | |
| **3** | a. Perform testing of hypothesis using one sample t-test.<br>b. Perform testing of hypothesis using two sample t-test.<br>c. Perform testing of hypothesis using paired t- test. | |
| **4** | a. Perform testing of hypothesis using chi- squared goodness-of-fit test.<br>b. Perform testing of hypothesis using chi- squared Test of Independence | |
| **5** | a. Perform testing of hypothesis using Z-test. | |
| **6** | a. Perform testing of hypothesis using one-way ANOVA.<br>b. Perform testing of hypothesis using two-way ANOVA<br>c. Perform testing of hypothesis using multivariate ANOVA (MANOVA). | |
| **7** | a. Perform the Random sampling for the givendata and analyse it.<br>b. Perform the Stratified sampling for the givendata and analyse it. | |
| **8** | a. Compute different types of correlation. | |
| **9** | a. Perform linear regression for prediction.<br>b. Perform polynomial regression for prediction. | |
| **10** | a. Perform multiple linear regression.<br>b. Perform Logistic regression. | |

# Practical 1

## A. Write a program for obtaining descriptive statistics of data.

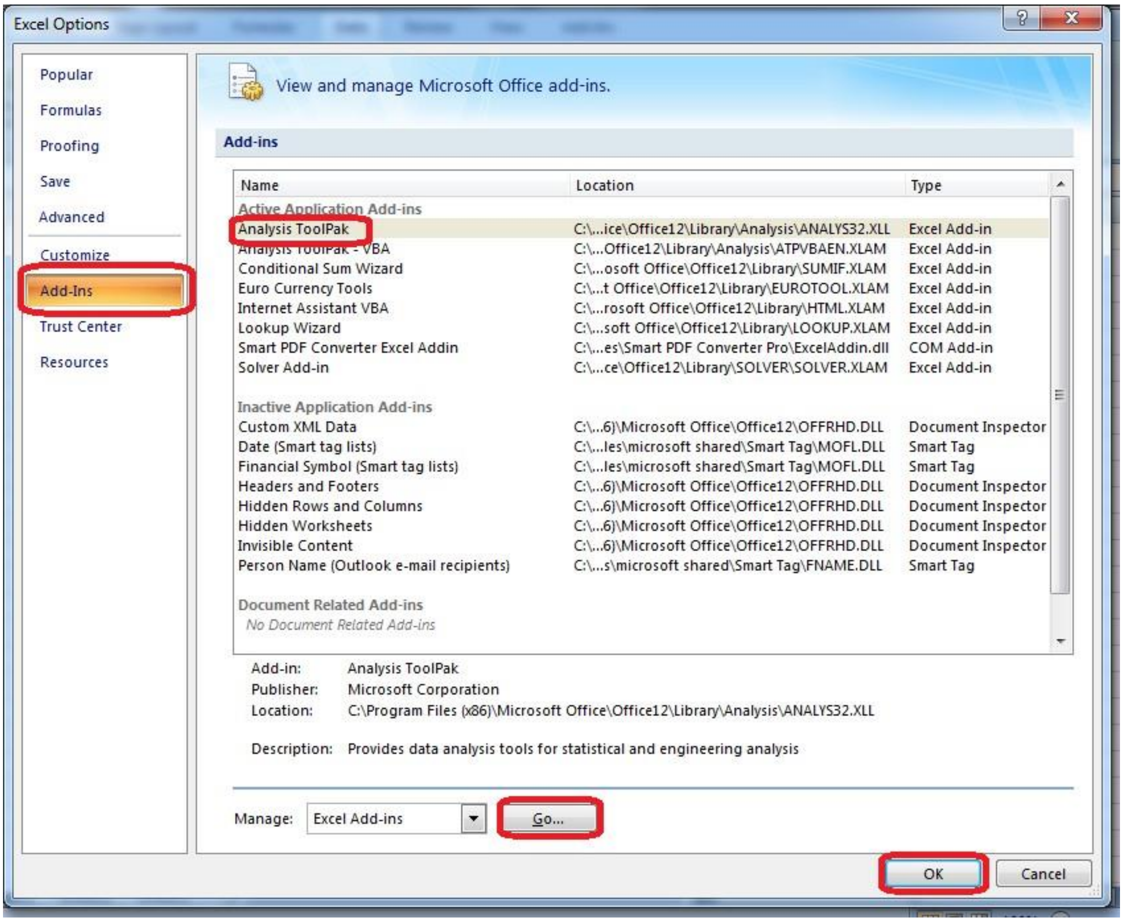
```
########################################################################
#Practical 1A: Write a python program on descriptive statistics analysis.
########################################################################
import pandas as pd
#Create a Dictionary of series
d = {'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}
#Create  a  DataFrame
df = pd.DataFrame(d)
print(df)
print('############ Sum ########## ')
print (df.sum())
print('############ Mean ########## ')
print (df.mean())
print('############ Standard Deviation ########## ')
print (df.std())
print('############ Descriptive Statistics ########## ')
print (df.describe())
```

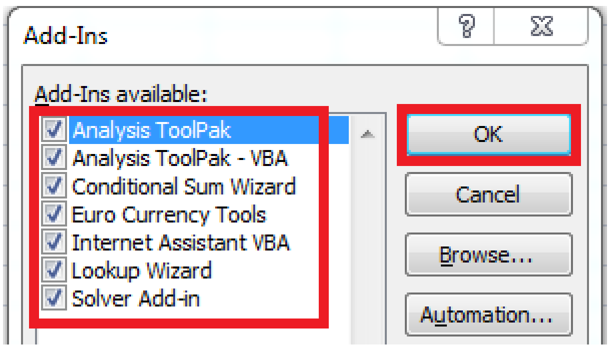
**Output:**

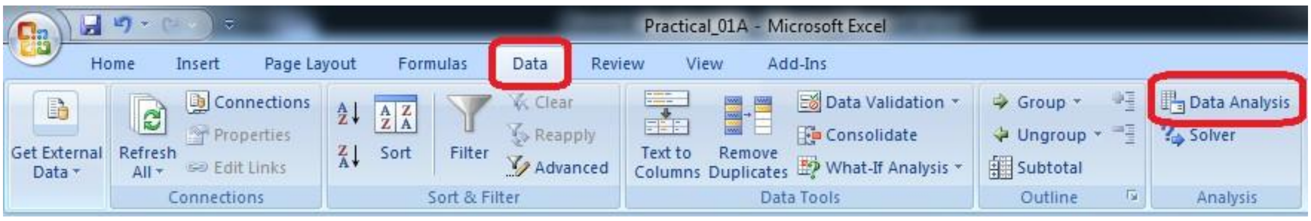
```
    Age  Rating
0    25    4.23
1    26    3.24
2    25    3.98
3    23    2.56
4    30    3.20
5    29    4.60
6    23    3.80
7    34    3.78
8    40    2.98
9    30    4.80
10   51    4.10
11   46    3.65
############  Sum ##########
Age       382.00
Rating     44.92
dtype: float64
############  Mean ##########
Age       31.833333
Rating     3.743333
dtype: float64
############  Standard Deviation ##########
Age       9.232682
Rating    0.661628
dtype: float64
############  Descriptive Statistics ##########
             Age     Rating
count  12.000000  12.000000
mean   31.833333   3.743333
std     9.232682   0.661628
min    23.000000   2.560000
25%    25.000000   3.230000
50%    29.500000   3.790000
75%    35.500000   4.132500
max    51.000000   4.800000
```

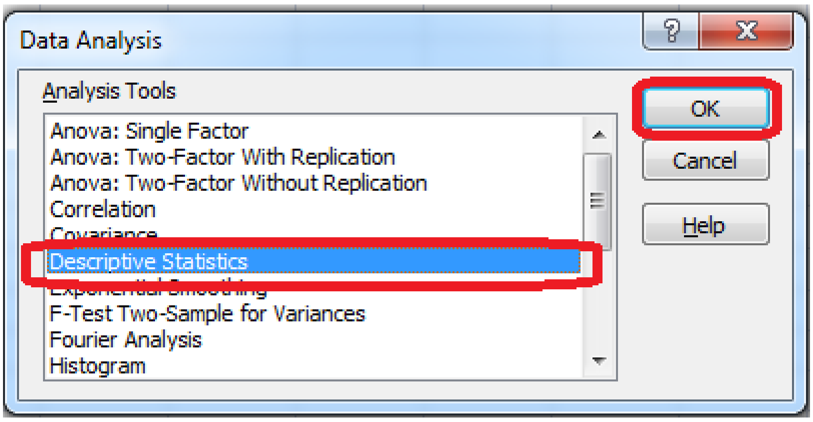
# Using Excel

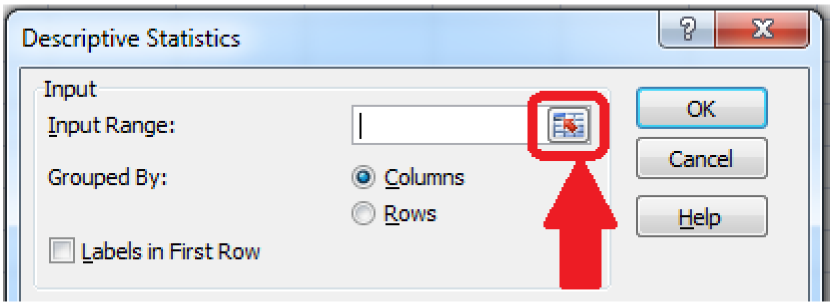
Go to File Menu □ Options □ Add-Ins□ Select Analysis ToolPak□ Press OK



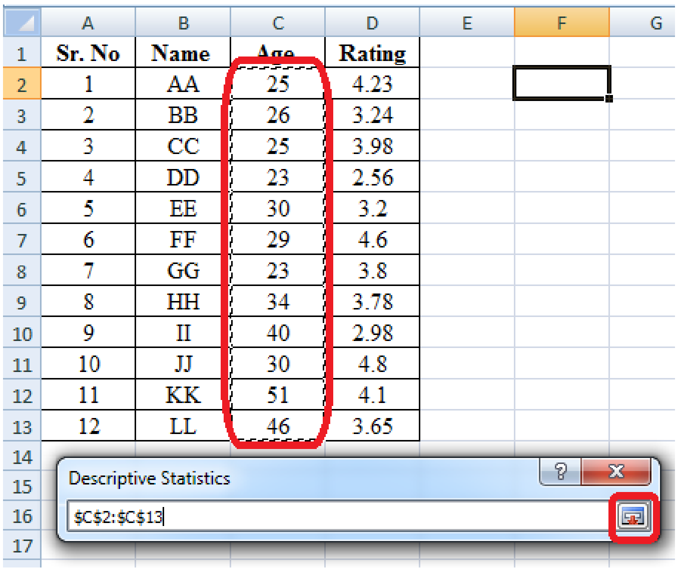



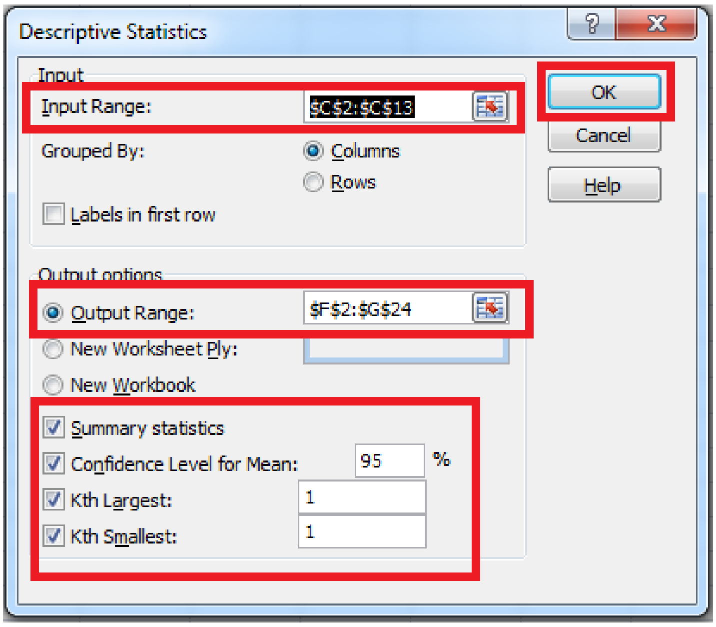






Select the data range from the excel worksheet.





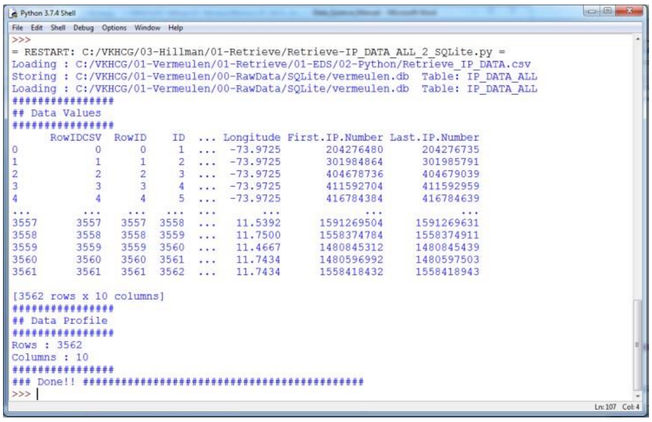

**Output:**

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Sr. No | Name | Age | Rating | | | |
| 2 | 1 | AA | 25 | 4.23 | | *Column1* | |
| 3 | 2 | BB | 26 | 3.24 | | | |
| 4 | 3 | CC | 25 | 3.98 | | Mean | 31.83333 |
| 5 | 4 | DD | 23 | 2.56 | | Standard Error | 2.665246 |
| 6 | 5 | EE | 30 | 3.2 | | Median | 29.5 |
| 7 | 6 | FF | 29 | 4.6 | | Mode | 25 |
| 8 | 7 | GG | 23 | 3.8 | | Standard Deviation | 9.232682 |
| 9 | 8 | HH | 34 | 3.78 | | Sample Variance | 85.24242 |
| 10 | 9 | II | 40 | 2.98 | | Kurtosis | 0.24931 |
| 11 | 10 | JJ | 30 | 4.8 | | Skewness | 1.135089 |
| 12 | 11 | KK | 51 | 4.1 | | Range | 28 |
| 13 | 12 | LL | 46 | 3.65 | | Minimum | 23 |
| 14 | | | | | | Maximum | 51 |
| 15 | | | | | | Sum | 382 |
| 16 | | | | | | Count | 12 |
| 17 | | | | | | Largest(1) | 51 |
| 18 | | | | | | Smallest(1) | 23 |
| 19 | | | | | | Confidence Level(95.0%) | 5.866167 |

## B. Import data from different data sources (from Excel, csv, mysql, sql server, oracle to R/Python/Excel)

# SQLite:

```
import sqlite3 as sq
import pandas as pd
################################################################
Base='C:/VKHCG'
sDatabaseName=Base + '/01-Vermeulen/00-RawData/SQLite/vermeulen.db'
conn = sq.connect(sDatabaseName)
################################################################
sFileName='C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-
Python/Retrieve_IP_DATA.csv'
print('Loading :',sFileName)
IP_DATA_ALL_FIX=pd.read_csv(sFileName,header=0,low_memory=False)
IP_DATA_ALL_FIX.index.names = ['RowIDCSV']
sTable='IP_DATA_ALL'
print('Storing :',sDatabaseName,' Table:',sTable)
IP_DATA_ALL_FIX.to_sql(sTable, conn, if_exists="replace")
print('Loading :',sDatabaseName,' Table:',sTable)
TestData=pd.read_sql_query("select * from IP_DATA_ALL;", conn)
print('################')
print('## Data Values')
print('################')
print(TestData)
print('################')
print('## Data Profile')
print('################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('################')
print('### Done!! ############################################')
```



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>>
= RESTART: C:/VKHCG/03-Hillman/01-Retrieve/Retrieve-IP_DATA_ALL_2_SQLite.py =
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv
Storing : C:/VKHCG/01-Vermeulen/00-RawData/SQLite/vermeulen.db  Table: IP_DATA_ALL
Loading : C:/VKHCG/01-Vermeulen/00-RawData/SQLite/vermeulen.db  Table: IP_DATA_ALL
################
## Data Values
################
      RowIDCSV  RowID    ID  ... Longitude First.IP.Number Last.IP.Number
0            0      0     1  ...  -73.9725       204276480      204276735
1            1      1     2  ...  -73.9725       301984864      301985791
2            2      2     3  ...  -73.9725       404678736      404679039
3            3      3     4  ...  -73.9725       411592704      411592959
4            4      4     5  ...  -73.9725       416784384      416784639
...        ...    ...   ...  ...       ...             ...            ...
3557      3557   3557  3558  ...   11.5392      1591269504     1591269631
3558      3558   3558  3559  ...   11.7500      1558374784     1558374911
3559      3559   3559  3560  ...   11.4667      1480845312     1480845439
3560      3560   3560  3561  ...   11.7434      1480596992     1480597503
3561      3561   3561  3562  ...   11.7434      1558418432     1558418943

[3562 rows x 10 columns]
################
## Data Profile
################
Rows : 3562
Columns : 10
################
### Done!! ############################################
>>>
Ln: 107 Col: 4
```

**MySQL:**

Open MySql

Create a database "DataScience"

Create a python file and add the following code:

################# Connection With MySQL #######################

importmysql.connector

conn = mysql.connector.connect(host='localhost',

database='DataScience',

user='root',

password='root')

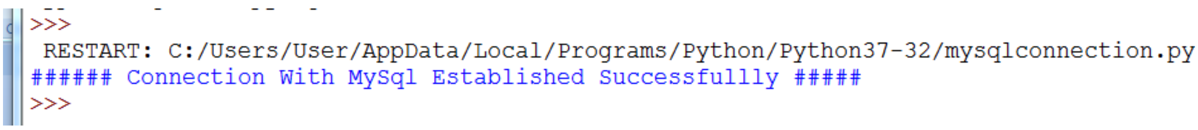
conn.connect

if(conn.is_connected):

print('###### Connection With MySql Established Successfullly ##### ')

else:

print('Not Connected -- Check Connection Properites')



```
>>>
 RESTART: C:/Users/User/AppData/Local/Programs/Python/Python37-32/mysqlconnection.py
###### Connection With MySql Established Successfullly #####
>>>
```

# Microsoft Excel

#################Retrieve-Country-Currency.py

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

importos

import pandas as pd

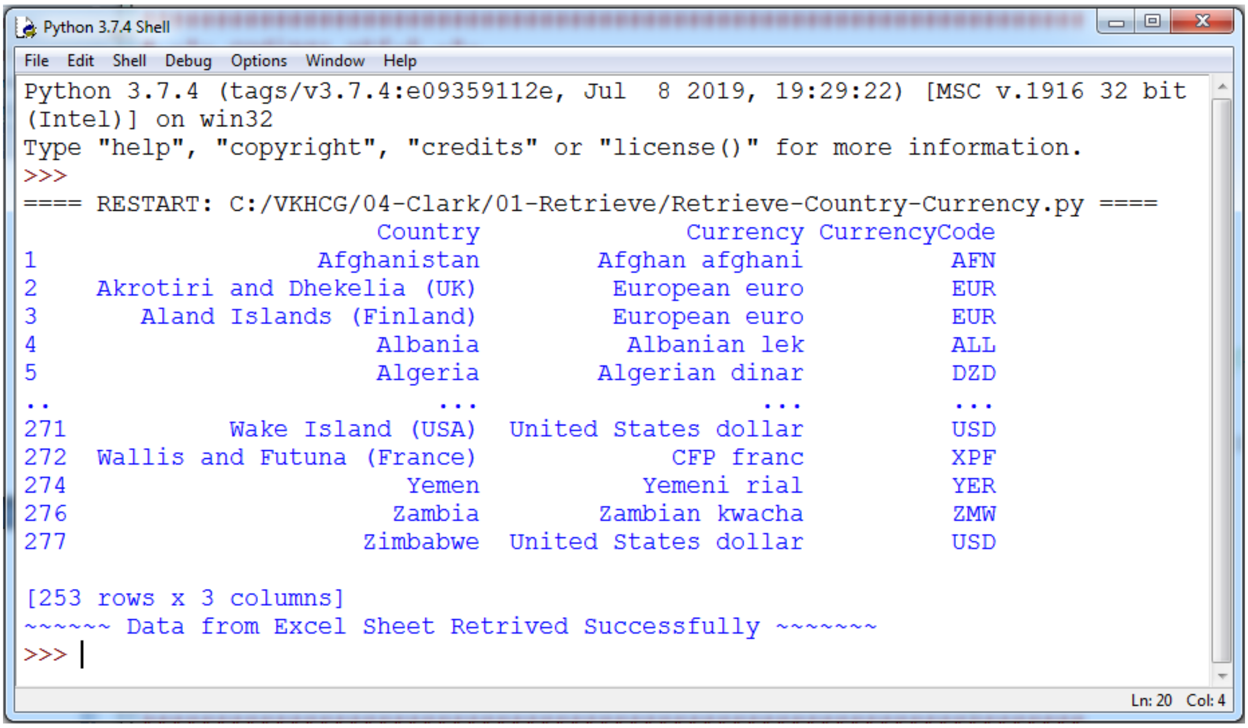
Base='C:/VKHCG'

sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'

#if not os.path.exists(sFileDir):

```
#os.makedirs(sFileDir)
CurrencyRawData = pd.read_excel('C:/VKHCG/01-Vermeulen/00-RawData/Country_Currency.xlsx')
sColumns = ['Country or territory', 'Currency', 'ISO-4217']
CurrencyData = CurrencyRawData[sColumns]
CurrencyData.rename(columns={'Country or territory': 'Country', 'ISO-4217':
'CurrencyCode'}, inplace=True)
CurrencyData.dropna(subset=['Currency'],inplace=True)
CurrencyData['Country'] = CurrencyData['Country'].map(lambda x: x.strip())
CurrencyData['Currency'] = CurrencyData['Currency'].map(lambda x:
x.strip())
CurrencyData['CurrencyCode'] = CurrencyData['CurrencyCode'].map(lambda x:
x.strip())
print(CurrencyData)
print('~~~~~~ Data from Excel Sheet Retrived Successfully ~~~~~~~ ')
sFileName=sFileDir + '/Retrieve-Country-Currency.csv'
CurrencyData.to_csv(sFileName, index = False)
```

**OUTPUT:**

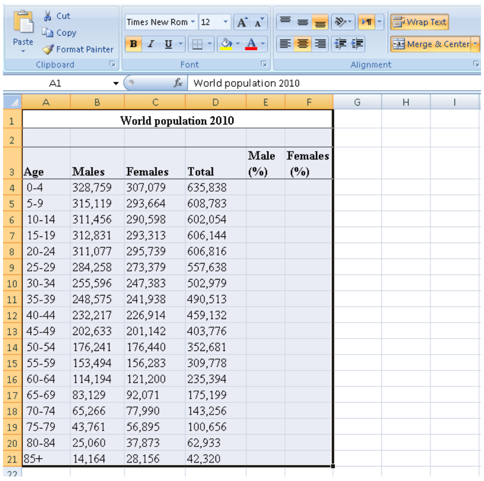

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/VKHCG/04-Clark/01-Retrieve/Retrieve-Country-Currency.py ====
                        Country               Currency CurrencyCode
1                   Afghanistan         Afghan afghani          AFN
2    Akrotiri and Dhekelia (UK)          European euro          EUR
3       Aland Islands (Finland)          European euro          EUR
4                       Albania           Albanian lek          ALL
5                       Algeria         Algerian dinar          DZD
..                          ...                    ...          ...
271           Wake Island (USA)   United States dollar          USD
272  Wallis and Futuna (France)              CFP franc          XPF
274                       Yemen           Yemeni rial          YER
276                      Zambia         Zambian kwacha          ZMW
277                    Zimbabwe   United States dollar          USD

[253 rows x 3 columns]
~~~~~~ Data from Excel Sheet Retrived Successfully ~~~~~~~
>>> |
Ln: 20  Col: 4
```

# Practical 2

## Perform analysis of given secondary data.

1. **Determine your research question** – Knowing exactly what you are looking for.
2. **Locating data**– Knowing what is out there and whether you can gain access to it. A quick Internet search, possibly with the help of a librarian, will reveal a wealth of options.
3. **Evaluating relevance of the data** – Considering things like the data's original purpose, when it was collected, population, sampling strategy/sample, data collection protocols, operationalization of concepts, questions asked, and form/shape of the data.
4. **Assessing credibility of the data** – Establishing the credentials of the original researchers, searching for full explication of methods including any problems encountered, determining how consistent the data is with data from other sources, and discovering whether the data has been used in any credible published research.
5. **Analysis –** This will generally involve a range of statistical processes.

**Example:**Analyze the given Population Census Data for Planning and Decision Making by using the size and composition of populations.

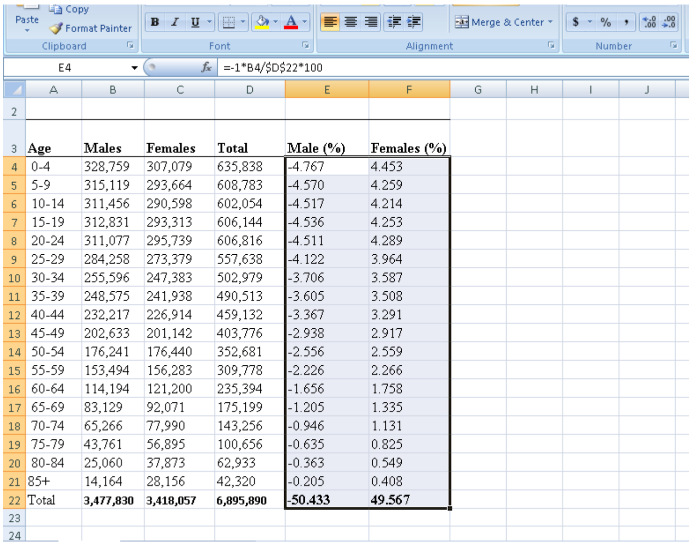


World population 2010

| Age | Males | Females | Total | Male (%) | Females (%) |
|---|---|---|---|---|---|
| 0-4 | 328,759 | 307,079 | 635,838 | | |
| 5-9 | 315,119 | 293,664 | 608,783 | | |
| 10-14 | 311,456 | 290,598 | 602,054 | | |
| 15-19 | 312,831 | 293,313 | 606,144 | | |
| 20-24 | 311,077 | 295,739 | 606,816 | | |
| 25-29 | 284,258 | 273,379 | 557,638 | | |
| 30-34 | 255,596 | 247,383 | 502,979 | | |
| 35-39 | 248,575 | 241,938 | 490,513 | | |
| 40-44 | 232,217 | 226,914 | 459,132 | | |
| 45-49 | 202,633 | 201,142 | 403,776 | | |
| 50-54 | 176,241 | 176,440 | 352,681 | | |
| 55-59 | 153,494 | 156,283 | 309,778 | | |
| 60-64 | 114,194 | 121,200 | 235,394 | | |
| 65-69 | 83,129 | 92,071 | 175,199 | | |
| 70-74 | 65,266 | 77,990 | 143,256 | | |
| 75-79 | 43,761 | 56,895 | 100,656 | | |
| 80-84 | 25,060 | 37,873 | 62,933 | | |
| 85+ | 14,164 | 28,156 | 42,320 | | |

Put the cursor in cell **B22** and click on the **AutoSum** and then click **Enter**. This will calculate the total population. Then copy the formula in cell **D22** across the row **22.**To calculate the percent of males in cell **E4**, enter the formula **=-1*100*B4/$D$22** . And copy the formula in cell **E4** down to cell **E21.**

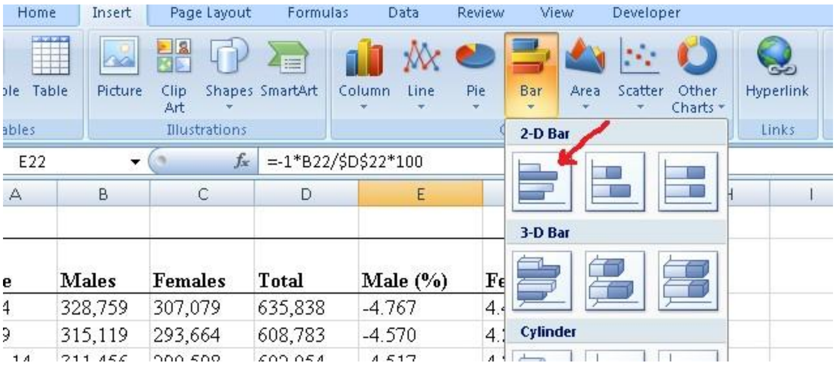
To calculate the percent of females in cell **F4**, enter the formula =**100*C4/$D$22**. Copy the formula in cell **F4** down to cell **F21.**



| Age | Males | Females | Total | Male (%) | Females (%) |
|---|---|---|---|---|---|
| 0-4 | 328,759 | 307,079 | 635,838 | -4.767 | 4.453 |
| 5-9 | 315,119 | 293,664 | 608,783 | -4.570 | 4.259 |
| 10-14 | 311,456 | 290,598 | 602,054 | -4.517 | 4.214 |
| 15-19 | 312,831 | 293,313 | 606,144 | -4.536 | 4.253 |
| 20-24 | 311,077 | 295,739 | 606,816 | -4.511 | 4.289 |
| 25-29 | 284,258 | 273,379 | 557,638 | -4.122 | 3.964 |
| 30-34 | 255,596 | 247,383 | 502,979 | -3.706 | 3.587 |
| 35-39 | 248,575 | 241,938 | 490,513 | -3.605 | 3.508 |
| 40-44 | 232,217 | 226,914 | 459,132 | -3.367 | 3.291 |
| 45-49 | 202,633 | 201,142 | 403,776 | -2.938 | 2.917 |
| 50-54 | 176,241 | 176,440 | 352,681 | -2.556 | 2.559 |
| 55-59 | 153,494 | 156,283 | 309,778 | -2.226 | 2.266 |
| 60-64 | 114,194 | 121,200 | 235,394 | -1.656 | 1.758 |
| 65-69 | 83,129 | 92,071 | 175,199 | -1.205 | 1.335 |
| 70-74 | 65,266 | 77,990 | 143,256 | -0.946 | 1.131 |
| 75-79 | 43,761 | 56,895 | 100,656 | -0.635 | 0.825 |
| 80-84 | 25,060 | 37,873 | 62,933 | -0.363 | 0.549 |
| 85+ | 14,164 | 28,156 | 42,320 | -0.205 | 0.408 |
| Total | 3,477,830 | 3,418,057 | 6,895,890 | -50.433 | 49.567 |

To build the population pyramid, we need to choose a horizontal bar chart with two series of data (% male and % female) and the age labels in column A as the **Category X-axis** labels. Highlight the range **A3:A21**, hold down the CTRL key and highlight the range **E3:F21**

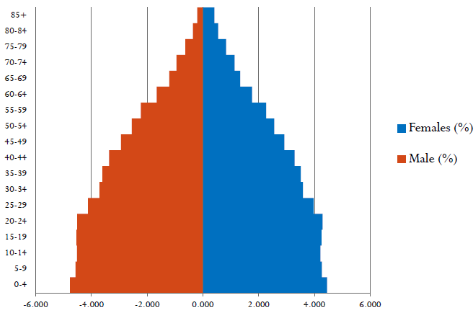
Under **inset** tab, under horizontal bar charts select **clustered bar chart**




Put the tip of your mouse arrow on the **Y-axis** (vertical axis) so it says “Category Axis”, right click and chose **Format Axis**

Choose **Axis options** tab and set the major and minor tick mark type to **None**, Axis labels to **Low**, and click **OK**.

Click on any of the bars in your pyramid, click right and select “format data series”. Set the **Overlap** to **100** and **Gap Width** to **0**. Click **OK**.



# Practical 3

## A. Perform testing of hypothesis using one sample t-test.

**One sample t-test** : The One Sample T Test determines whether the sample mean is statistically different from a known or hypothesised population mean. The One Sample T Test is a parametric test.

**Program Code:**

```
# -*- coding: utf-8 -*-
fromscipy.stats import ttest_1samp
importnumpy as np
ages = np.genfromtxt('ages.csv')
print(ages)
ages_mean = np.mean(ages)
print(ages_mean)
tset, pval = ttest_1samp(ages, 30)
print('p-values - ',pval)

if pval< 0.05:    # alpha value is 0.05
        print(" we are rejecting null hypothesis")
else:
        print("we are accepting null hypothesis")
```

## Output:

```
In [4]: runfile('K:/Research In Computing/Practical Material/Programs/
Practical_05/Prac_3A.py', wdir='K:/Research In Computing/Practical Material/
Programs/Practical_05')
[20. 30. 25. 13. 16. 17. 34. 35. 38. 42. 43. 45. 48. 49. 50. 51. 54. 55.
 56. 59. 61. 62. 18. 22. 29. 30. 31. 39. 52. 53. 67. 36. 47. 54. 40. 40.
 35. 22. 59. 58. 30. 43. 22. 45. 21. 59. 51. 47. 25. 58. 50. 23. 24. 45.
 37. 59. 28. 28. 48. 42. 54. 36. 36. 24. 26. 24. 50. 48. 34. 44. 56. 55.
 35. 33. 39. 53. 34. 28. 56. 24. 21. 29. 28. 58. 35. 57. 26. 25. 59. 56.
 22. 57. 48. 33. 23. 26. 57. 32. 53. 31. 35. 44. 54. 25. 31. 58. 26. 32.
 26. 50. 41. 49. 26. 33. 34. 24. 43. 42. 51. 36. 38. 38. 40. 38. 56. 39.
 23. 33. 53. 30. 38.]
39.47328244274809
p-values -  5.362905195437013e-14
 we are rejecting null hypothesis
```

## B. Write a program for t-test comparing two means for independent samples.

The T distribution provides a good way to perform one sample tests on the mean when the population variance is not known provided the population is normal or the sample is sufficiently large so that the Central Limit Theorem applies.

### Two Sample t Test

Example: A college Princiapal informed classroom teachers that some of their students showedunusual potential for intellectual gains. One months later the students identified to teachers ashaving potentional for unusual intellectual gains showed significiantly greater gains performanceon a test said to measure IQ than did students who were not so identified. Below are the data forthe students:

| **Experimental** | **Comparison** | |
|---|---|---|
| 35 | 2 | |
| 40 | 27 | |
| 12 | 38 | |
| 15 | 31 | |
| 21 | 1 | |
| 14 | 19 | |
| 46 | 1 | |
| 10 | 34 | |
| 28 | 3 | |
| 48 | 1 | |
| 16 | 2 | |
| 30 | 3 | |
| 32 | 2 | |
| 48 | 1 | |
| 31 | 2 | |
| 22 | 1 | |
| 12 | 3 | |
| 39 | 29 | |
| 19 | 37 | |
| 25 | 2 | |
| 27.15 | 11.95 | Mean |
| 12.51 | 14.61 | Sd |

Experimental Data

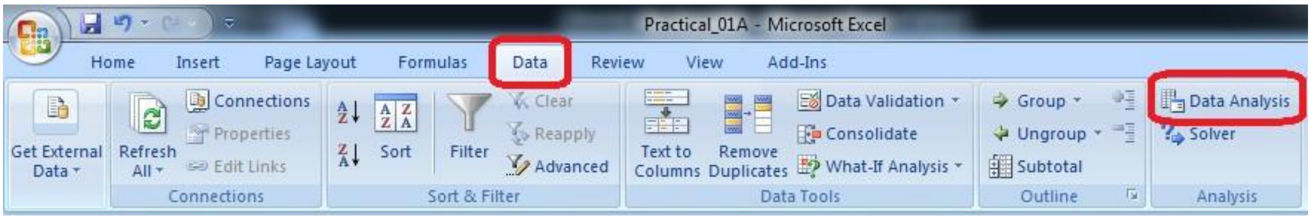
To calculate Standard Mean go to cell A22 and type =SUM(A2:A21)/20

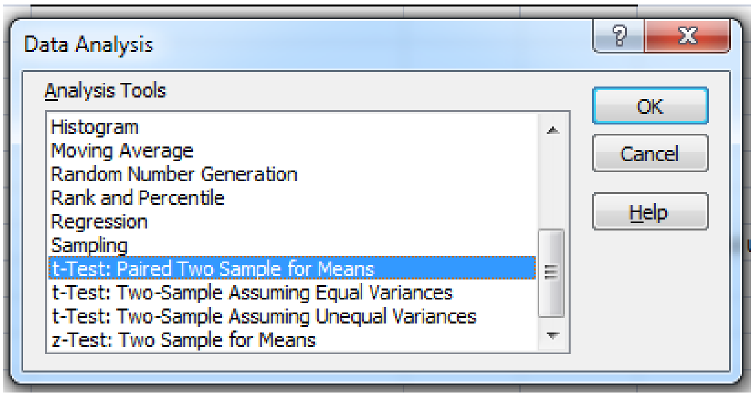
To calculate Standard Deviation go to cell A23 and type =STDEV(A2:A21)

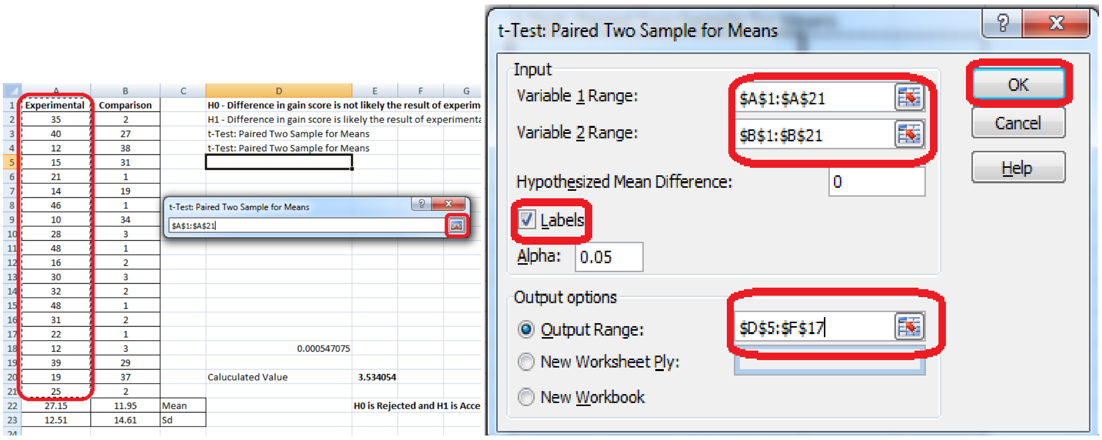
Comparison Data

To calculate Standard Mean go to cell B22 and type =SUM(B2:B21)/20

To calculate Standard Deviation go to cell B23 and type =STDEV(B2:B21) To find T-Test Statistics go to data □Data Analysis




To caluculate the T-Test square value go to cell E20 and type

=(A22-B22)/SQRT((A23*A23)/COUNT(A2:A21)+(B23*B23)/COUNT(A2:A21))

Now go to cell E20 and type

=IF(E20<E12,"H0 is Accepted", "H0 is Rejected and H1 is Accepted")

Our calculated value is larger than the tabled value at alpha = .01, so we reject the null hypothesisand accept the alternative hypothesis, namely, that the difference in gain scores is likely the resultof the experimental treatment and not the result of chance variation.

**Output:**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Experimental | Comparison | | H0 - Difference in gain score is not likely the result of experimental treatment. | | |
| 2 | 35 | 2 | | H1 - Difference in gain score is likely the result of experimental treatment and not the result of change variation. | | |
| 3 | 40 | 27 | | t-Test: Paired Two Sample for Means | | |
| 4 | 12 | 38 | | t-Test: Paired Two Sample for Means | | |
| 5 | 15 | 31 | | t-Test: Paired Two Sample for Means | | |
| 6 | 21 | 1 | | | | |
| 7 | 14 | 19 | | | *Experimental* | *Comparison* |
| 8 | 46 | 1 | | Mean | 27.15 | 11.95 |
| 9 | 10 | 34 | | Variance | 156.45 | 213.5236842 |
| 10 | 28 | 3 | | Observations | 20 | 20 |
| 11 | 48 | 1 | | Pearson Correlation | -0.395904927 | |
| 12 | 16 | 2 | | Hypothesized Mean Difference | 0 | |
| 13 | 30 | 3 | | df | 19 | |
| 14 | 32 | 2 | | t Stat | 2.996289153 | |
| 15 | 48 | 1 | | P(T<=t) one-tail | 0.003711226 | |
| 16 | 31 | 2 | | t Critical one-tail | 1.729132792 | |
| 17 | 22 | 1 | | P(T<=t) two-tail | 0.007422452 | |
| 18 | 12 | 3 | | t Critical two-tail | 2.09302405 | |
| 19 | 39 | 29 | | | | |
| 20 | 19 | 37 | | Caluculated Value | 3.534053898 | |
| 21 | 25 | 2 | | | | |
| 22 | 27.15 | 11.95 | Mean | | H0 is Rejected and H1 is Accepted | |
| 23 | 12.51 | 14.61 | Sd | | | |

**Using Python**

```
importnumpy as np
fromscipy import stats
fromnumpy.random import randn
N = 20
#a = [35,40,12,15,21,14,46,10,28,48,16,30, 32,48,31,22,12,39,19,25]
#b = [2,27,31,38,1,19,1,34,3,1,2,1,3,1,2,1,3,29,37,2]
a = 5 * randn(100) + 50
b = 5 * randn(100) + 51
var_a = a.var(ddof=1)
var_b = b.var(ddof=1)
s = np.sqrt((var_a + var_b)/2)
t = (a.mean() - b.mean())/(s*np.sqrt(2/N))
df = 2*N - 2
#p-value after comparison with the t
p = 1 - stats.t.cdf(t,df=df)
print("t = " + str(t))
print("p = " + str(2*p))
if t> p :
print('Mean of two distribution are differnt and significant')
else:
print('Mean of two distribution are same and not significant')
```

**Output:**

```
In [9]: runfile('E:/Research In Computing/Programs/
Practical_04/Program_4B.py', wdir='E:/Research In
Computing/Programs/Practical_04')
t = -1.051463820987354
p = 1.700313560478936
Mean of two distribution are same and not significant

In [10]: runfile('E:/Research In Computing/Programs/
Practical_04/Program_4B.py', wdir='E:/Research In
Computing/Programs/Practical_04')
t = 0.46409515960993775
p = 0.6452274090296801
Mean of two distribution are differnt and significant
```

## A. Perform testing of hypothesis using paired t-test.

The paired sample t-test is also called dependent sample t-test. It’s an univariate test that tests for a significant difference between 2 related variables. An example of this is if you where t

collect the blood pressure for an individual before and after some treatment, condition, or time point. The data set contains blood pressure readings before and after an intervention. These are variables “bp_before” and “bp_after”.

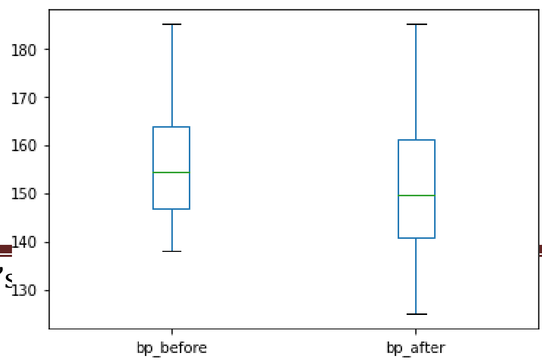
The hypothesis being test is:

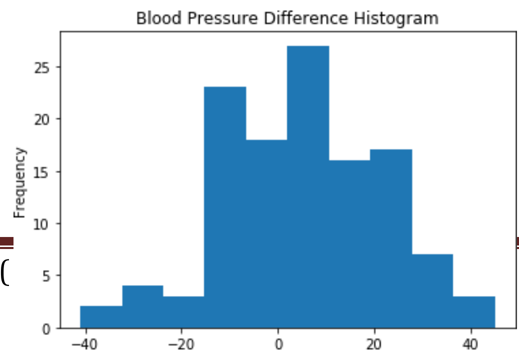
- **$H_0$** - The mean difference between sample 1 and sample 2 is equal to 0.
- **$H_0$** - The mean difference between sample 1 and sample 2 is not equal to 0

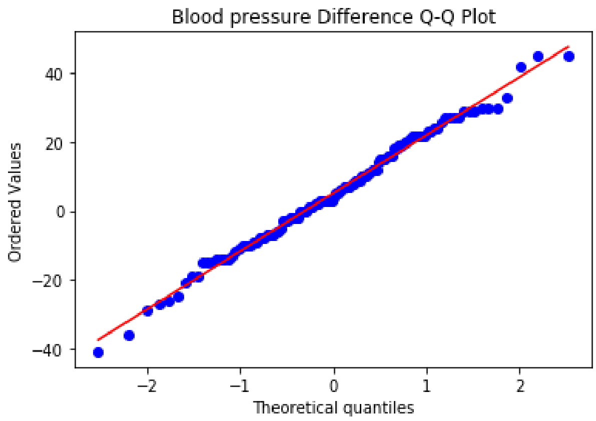
**Program Code:**

```
# -*- coding: utf-8 -*-
Created on Mon Dec 16 19:49:23 2019
from scipy import stats
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv("blood_pressure.csv")
print(df[['bp_before','bp_after']].describe())

#First let’s check for any significant outliers in
#each of the variables.
df[['bp_before', 'bp_after']].plot(kind='box')
# This saves the plot as a png file
plt.savefig('boxplot_outliers.png')

# make a histogram to differences between the two scores.
df['bp_difference'] = df['bp_before'] - df['bp_after']

df['bp_difference'].plot(kind='hist', title= 'Blood Pressure Difference Histogram')
#Again, this saves the plot as a png file
plt.savefig('blood pressure difference histogram.png')
stats.probplot(df['bp_difference'], plot= plt)
plt.title('Blood pressure Difference Q-Q Plot')
plt.savefig('blood pressure difference qq plot.png')
stats.shapiro(df['bp_difference'])
stats.ttest_rel(df['bp_before'], df['bp_after'])
```

**Output:**






```
In [21]: runfile('K:/Research In Computing/Practical
Material/Programs/Practical_03/Practical_03C.py', wdir='K:/
Research In Computing/Practical Material/Programs/
Practical_03')
        bp_before    bp_after
count  120.000000  120.000000
mean   156.450000  151.358333
std     11.389845   14.177622
min    138.000000  125.000000
25%    147.000000  140.750000
50%    154.500000  149.500000
75%    164.000000  161.000000
max    185.000000  185.000000
0.0011297914644840823
Ttest_relResult(statistic=3.3371870510833657,
pvalue=0.0011297914644840823)
reject null hypothesis
```

A paired sample t-test was used to analyze the blood pressure before and after the intervention to test if the intervention had a significant affect on the blood pressure. The blood pressure before the intervention was higher (156.45 ± 11.39 units) compared to the blood pressure post intervention (151.36 ± 14.18 units); there was a statistically significant decrease in blood pressure ($t(119)=3.34$, $p= 0.0011$) of 5.09 units.

# Practical 4

## A. Perform testing of hypothesis using chi-squared goodness-of-fit test. Problem

Ansystem administrator needs to upgrade the computers for his division. He wants to know what sort of computer system his workers prefer. He gives three choices: Windows, Mac, or Linux. Test the hypothesis or theory that an equal percentage of the population prefers each type of computer system .

| System | O | Ei | $\sum \frac{(O_i - E_i)^2}{Ei}$ |
|---|---|---|---|
| **Windows** | 20 | **33.33 %** | |
| **Mac** | 60 | **33.33 %** | |
| **Linux** | 20 | **33.33 %** | |

H0 : The population distribution of the variable is the same as the proposed distribution

HA : The distributions are different

To calculate the Chi –Squred value for Windows go to cell D2 and type =((B2-C2)*(B2-C2))/C2

To calculate the Chi –Squred value for Mac go to cell D3 and type =((B3-C3)*(B3-C3))/C3

To calculate the Chi –Squred value for Mac go to cell D3 and type =((B4-C4)*(B4-C4))/C4

Go to Cell D5 for $\sum \frac{(O_i - E_i)^2}{Ei}$ and type=SUM(D2:D4)

To get the table value for Chi-Square for α = 0.05 and dof = 2, go to cell D7 and type =CHIINV(0.05,2)

At cell D8 type =IF(D5>D7, "H0 Accepted","H0 Rejected")

**Output:**

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | System | O | Ei | $\sum \frac{(O_i - E_i)^2}{Ei}$ | | | | | | | | | | |
| 2 | Windows | 20 | 33.33 | 5.333333 | | H0 : The population distribution of the variable is the same as the proposed distribution | | | | | | | | |
| 3 | Mac | 60 | 33.33 | 21.33333 | | H1 - : The distributions are different | | | | | | | | |
| 4 | Linux | 20 | 33.33 | 5.333333 | | | | | | | | | | |
| 5 | Total | 100 | 100 | 32 | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | |
| 7 | | | Table Value | 5.991465 | | | | | | | | | | |
| 8 | | | H0 Accepted | | | | | | | | | | | |

**B. Perform testing of hypothesis using chi-squared test of independence.**

In a study to understatnd the permormacne of M. Sc. IT Part -1 class, a college selects a random sample of 100 students. Each student was asked his grade obtained in B. Sc. IT. The sample is as given below

| Sr. No | Roll No | Student's Name | Gen | Grade |
|---|---|---|---|---|
| 1 | 1 | Gaborone | m | O |
| 2 | 2 | Francistown | m | O |
| 3 | 5 | Niamey | m | O |
| 4 | 13 | Maxixe | m | O |
| 5 | 16 | Tema | m | O |
| 6 | 17 | Kumasi | m | O |
| 7 | 34 | Blida | m | O |
| 8 | 35 | Oran | m | O |
| 9 | 38 | Saefda | m | O |
| 10 | 42 | Constantine | m | O |
| 11 | 43 | Annaba | m | O |
| 12 | 45 | Bejaefa | m | O |
| 13 | 48 | Medea | m | O |
| 14 | 49 | Djelfa | m | O |
| 15 | 50 | Tipaza | m | O |
| 16 | 51 | Bechar | m | O |
| 17 | 54 | Mostaganem | m | O |
| 18 | 55 | Tiaret | m | O |
| 19 | 56 | Bouira | m | O |
| 20 | 59 | Tebessa | m | O |
| 21 | 61 | El Harrach | m | O |
| 22 | 62 | Mila | m | O |
| 23 | 65 | Fouka | m | O |
| 24 | 66 | El Eulma | m | O |
| 25 | 68 | SidiBel Abbes | m | O |
| 26 | 69 | Jijel | m | O |
| 27 | 70 | Guelma | m | O |
| 28 | 85 | Khemis El Khechna | m | O |
| 29 | 87 | Bordj El Kiffan | m | O |
| 30 | 88 | Lakhdaria | m | O |
| 31 | 6 | Maputo | m | D |
| 32 | 12 | Lichinga | m | D |
| 33 | 15 | Ressano Garcia | m | D |
| 34 | 19 | Accra | m | D |
| 35 | 27 | Wa | m | D |
| 36 | 28 | Navrongo | m | D |
| 37 | 37 | Mascara | m | D |
| 38 | 44 | Batna | m | D |
| 39 | 57 | El Biar | m | D |
| 40 | 60 | Boufarik | m | D |
| 41 | 63 | OuedRhiou | m | D |
| 42 | 64 | Souk Ahras | m | D |
| 43 | 71 | Dar El Befda | m | D |
| 44 | 86 | Birtouta | m | D |
| 45 | 18 | Takoradi | m | C |
| 46 | 22 | Cape Coast | m | C |
| 47 | 29 | Kwabeng | m | C |
| 48 | 30 | Algiers | m | C |

| Sr. No | Roll No | Student's Name | Gen | Grade |
|---|---|---|---|---|
| 62 | 3 | Maun | f | O |
| 63 | 7 | Tete | f | O |
| 64 | 9 | Chimoio | f | O |
| 65 | 11 | Pemba | f | O |
| 66 | 14 | Chibuto | f | O |
| 67 | 25 | Mampong | f | O |
| 68 | 36 | Tlemcen | f | O |
| 69 | 40 | Adrar | f | O |
| 70 | 41 | Tindouf | f | O |
| 71 | 46 | Skikda | f | O |
| 72 | 47 | Ouargla | f | O |
| 73 | 10 | Matola | f | D |
| 74 | 20 | Legon | f | D |
| 75 | 21 | Sunyani | f | D |
| 76 | 72 | Teenas | f | D |
| 77 | 73 | Kouba | f | D |
| 78 | 75 | HussenDey | f | D |
| 79 | 77 | Khenchela | f | D |
| 80 | 82 | HassiBahbah | f | D |
| 81 | 84 | Baraki | f | D |
| 82 | 91 | Boudouaou | f | D |
| 83 | 95 | Tadjenanet | f | D |
| 84 | 4 | Molepolole | f | C |
| 85 | 8 | Quelimane | f | C |
| 86 | 23 | Bolgatanga | f | C |
| 87 | 58 | Mohammadia | f | C |
| 88 | 83 | Merouana | f | C |
| 89 | 24 | Ashaiman | f | B |
| 90 | 76 | N'gaous | f | B |
| 91 | 90 | Bab El Oued | f | B |
| 92 | 92 | BordjMenael | f | B |
| 93 | 93 | Ksar El Boukhari | f | B |
| 94 | 74 | Reghaa | f | A |
| 95 | 78 | Cheria | f | A |
| 96 | 79 | Mouzaa | f | A |
| 97 | 80 | Meskiana | f | A |
| 98 | 81 | Miliana | f | A |
| 99 | 94 | Sig | f | A |
| 100 | 99 | Kadiria | f | A |

| | | | | |
|---|---|---|---|---|
| **49** | 31 | Laghouat | m | C |
| **50** | 39 | Relizane | m | C |
| **51** | 52 | Setif | m | C |
| **52** | 53 | Biskra | m | C |
| **53** | 67 | Kolea | m | C |
| **54** | 100 | AefnFakroun | m | C |
| **55** | 26 | Nima | m | B |
| **56** | 32 | TiziOuzou | m | B |
| **57** | 33 | Chlef | m | B |
| **58** | 89 | M'sila | m | A |
| **59** | 96 | Heliopolis | m | A |
| **60** | 97 | Berrouaghia | m | A |
| **61** | 98 | Sougueur | m | A |

**Null Hypothesis - H0 :** The performance of girls students is same as boys students.
**Alternate Hypothesis - H1 :** The performance of boys and girls students are different.
Open Excel Workbook

| | **O** | **A** | **B** | **C** | **D** | **Total** | $\sum \frac{(O_i - E_i)^2}{Ei}$ |
|---|---|---|---|---|---|---|---|
| **Girls** | 11 | 7 | 5 | 5 | 11 | **39** | 6.075 |
| **Boys** | 30 | 4 | 3 | 10 | 14 | **61** | 6.075 |
| **Total** | 41 | 11 | 8 | 15 | 25 | **100** | **12.150** |
| **Ei** | **20.5** | **5.5** | **4** | **7.5** | **12.5** | **50** | |

Prepare a contingency table as shown above.
To calculate Girls Students with ‘O’ Grade
Go to Cell N6 and type =COUNTIF($J$2:$K$40,"O")

To calculate Girls Students with ‘A’ Grade
Go to Cell O6 and type =COUNTIF($J$2:$K$40,"A")

To calculate Girls Students with ‘B’ Grade
Go to Cell P6 and type =COUNTIF($J$2:$K$40,"B")

To calculate Girls Students with ‘C’ Grade
Go to Cell Q6 and type =COUNTIF($J$2:$K$40,"C")

To calculate Girls Students with ‘D’ Grade
Go to Cell R6 and type =COUNTIF($J$2:$K$40,"D")

To calculate Boys Students with ‘O’ Grade
Go to Cell N7 and type =COUNTIF($D$2:$E$62,"O")
To calculate Boys Students with ‘A’ Grade
Go to Cell O7 and type =COUNTIF($D$2:$E$62,"A")
To calculate Boys Students with ‘B’ Grade
Go to Cell P7 and type =COUNTIF($D$2:$E$62,"B")
To calculate Boys Students with ‘C’ Grade
Go to Cell Q7 and type =COUNTIF($D$2:$E$62,"C")
To calculate Boys Students with ‘D’ Grade

Go to Cell R7 and type =COUNTIF($D$2:$E$62,"D")

**To calculated the expected value Ei**
Go to Cell N9 and type =N8/2
Go to Cell O9 and type =O8/2
Go to Cell P9 and type =P8/2
Go to Cell Q9 and type =Q8/2
Go to Cell R9 and type =R8/2
Go to Cell S6 and calculate total girl students = SUM(N6:R6)
Go to Cell S7 and calculate total girl students = SUM(N7:R7)

**Now Calculate** $\sum \frac{(O_i - E_i)^2}{Ei}$

Go to cell **T6** and type
=SUM((N6-$N$9)^2/$N$9,(O6-$O$9)^2/$O$9,(P6-$P$9)^2/$P$9,(Q6-Q$9)^2/$Q$9, (R6-$R$9)^2/$R$9)
Go to cell **T7** and type
=SUM((N7-$N$9)^2/$N$9,(O7-$O$9)^2/$O$9,(P7-$P$9)^2/$P$9,(Q7-Q$9)^2/$Q$9, (R7-$R$9)^2/$R$9)
To get the table value go to cell T11 and type **=CHIINV(0.05,4)**
Go to cell O13 and type =IF(T8>=T11," H0 is Accepted", "H0 is Rejected")

| M | N | O | P | Q | R | S | T |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| **H0 :** Performance of boys and girls are equal | | | | | | | |
| | | | | | | | |
| **Frequency Table** | | | | | | | $\frac{(O_i - E_i)^2}{}$ |
| | **O** | **A** | **B** | **C** | **D** | **Total** | **Ei** |
| **Girls** | 11 | 7 | 5 | 5 | 11 | **39** | 6.075 |
| **Boys** | 30 | 4 | 3 | 10 | 14 | **61** | 6.075 |
| **Total** | 41 | 11 | 8 | 15 | 25 | **100** | **12.150** |
| **Ei** | **20.5** | **5.5** | **4** | **7.5** | **12.5** | **50** | |
| | | | | | | | |
| **Critcal Value of α =0.05 for df = (2-1) * (5-1)** | | | | | | | 9.487729 |
| | | | | | | | |
| Decesion | | **H0 is Accepted** | | | | | |

## Using Python

```
importnumpy as np
import pandas as pd
importscipy.stats as stats
np.random.seed(10)
stud_grade = np.random.choice(a=["O","A","B","C","D"],
                              p=[0.20, 0.20 ,0.20, 0.20, 0.20], size=100)
stud_gen = np.random.choice(a=["Male","Female"], p=[0.5, 0.5], size=100)
mscpart1 = pd.DataFrame({"Grades":stud_grade, "Gender":stud_gen})
print(mscpart1)
stud_tab = pd.crosstab(mscpart1.Grades, mscpart1.Gender, margins=True)
stud_tab.columns = ["Male", "Female", "row_totals"]
stud_tab.index = ["O", "A", "B", "C", "D", "col_totals"]
observed = stud_tab.iloc[0:5, 0:2 ]
print(observed)
expected = np.outer(stud_tab["row_totals"][0:5],
stud_tab.loc["col_totals"][0:2]) / 100
print(expected)
chi_squared_stat = (((observed-expected)**2)/expected).sum().sum()
print('Calculated : ',chi_squared_stat)

crit = stats.chi2.ppf(q=0.95, df=4)
print('Table Value : ',crit)

ifchi_squared_stat>= crit:
print('H0 is Accepted ')
else:
print('H0 is Rejected ')
```

**Output :**

```
In [1]: runfile('E:/Research In Computing/Programs/
Practical_03/ChiSquaer.py', wdir='E:/Research In
Computing/Programs/Practical_03')
   Grades  Gender
0       C  Female
1       O  Female
2       C    Male
3       C    Male
4       B  Female
..    ...     ...
95      B    Male
96      D  Female
97      B  Female
98      A    Male
99      B    Male

[100 rows x 2 columns]
   Male  Female
O    11      12
A     9      13
B     7      11
C    10       8
D    12       7
[[11.27 11.73]
 [10.78 11.22]
 [ 8.82  9.18]
 [ 8.82  9.18]
 [ 9.31  9.69]]
Calculated :  3.158915138993211
Table Value :  9.487729036781154
H0 is Rejected
```

# Practical 5

## Perform testing of hypothesis using Z-test.

Use a Z test if:

- Your sample size is greater than 30. Otherwise, use a t test.
- Data points should be independent from each other. In other words, one data point isn't related or doesn't affect another data point.
- Your data should be normally distributed. However, for large sample sizes (over 30) this doesn't always matter.
- Your data should be randomly selected from a population, where each item has an equal chance of being selected.
- Sample sizes should be equal if at all possible.

**Ho -** Blood pressure has a mean of 156 units

**Program Code for one-sample Z test.**

```
from statsmodels.stats import weightstats as stests
import pandas as pd
from scipy import stats
df = pd.read_csv("blood_pressure.csv")
df[['bp_before','bp_after']].describe()
print(df)
ztest ,pval = stests.ztest(df['bp_before'], x2=None, value=156)
print(float(pval))

if pval<0.05:
   print("reject null hypothesis")
else:
   print("accept null hypothesis")
```

**Output:**

```
In [26]: runfile('K:/Research In Computing/Practical
Material/Programs/Practical_05/Z_Test_One_Sample.py',
wdir='K:/Research In Computing/Practical Material/Programs/
Practical_05')
     patient  gender agegrp  bp_before  bp_after
0          1    Male  30-45        143       153
1          2    Male  30-45        163       170
2          3    Male  30-45        153       168
3          4    Male  30-45        153       142
4          5    Male  30-45        146       141
..       ...     ...    ...        ...       ...
115      116  Female    60+        152       152
116      117  Female    60+        161       152
117      118  Female    60+        165       174
118      119  Female    60+        149       151
119      120  Female    60+        185       163

[120 rows x 5 columns]
0.6651614730255063
accept null hypothesis
```

**Two-sample Z test-** In two sample z-test , similar to t-test here we are checking two independent data groups and deciding whether sample mean of two group is equal or not.

**H0 : mean of two group is 0**

**H1 : mean of two group is not 0**

```
# -*- coding: utf-8 -*-
"""
```

Created on Mon Dec 16 20:42:17 2019
@author: MyHome """

```
import pandas as pd
from statsmodels.stats import weightstats as stests
df = pd.read_csv("blood_pressure.csv")
df[['bp_before','bp_after']].describe()
print(df)

ztest ,pval = stests.ztest(df['bp_before'], x2=df['bp_after'], value=0,alternative='two-sided')
print(float(pval))

if pval<0.05:
   print("reject null hypothesis")
else:
   print("accept null hypothesis")
```

```
In [29]: runfile('K:/Research In Computing/Practical
Material/Programs/Practical_05/Z_Test_Two_Sample.py',
wdir='K:/Research In Computing/Practical Material/Programs/
Practical_05')
     patient  gender agegrp  bp_before  bp_after
0          1    Male  30-45        143       153
1          2    Male  30-45        163       170
2          3    Male  30-45        153       168
3          4    Male  30-45        153       142
4          5    Male  30-45        146       141
..       ...     ...    ...        ...       ...
115      116  Female    60+        152       152
116      117  Female    60+        161       152
117      118  Female    60+        165       174
118      119  Female    60+        149       151
119      120  Female    60+        185       163

[120 rows x 5 columns]
0.002162306611369422
reject null hypothesis
```

# Practical 6

## A. Perform testing of hypothesis using One-way ANOVA.

ANOVA ASSUMPTIONS

- The dependent variable (SAT scores in our example) should be continuous.
- The independent variables (districts in our example) should be two or more categorical groups.
- There must be different participants in each group with no participant being in more than one group. In our case, each school cannot be in more than one district.
- The dependent variable should be approximately normally distributed for each category.
- Variances of each group are approximately equal.

From our data exploration, we can see that the average SAT scores are quite different for each district. Since we have five different groups, we cannot use the t-test, use the 1-way ANOVA test anyway just to understand the concepts.

**H0 - There are no significant differences between the groups' mean SAT scores.**

$$\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5$$

**H1 - There is a significant difference between the groups' mean SAT scores.**

If there is at least one group with a significant difference with another group, the null hypothesis will be rejected.

```
import pandas as pd
importnumpy as np
importmatplotlib.pyplot as plt
importseaborn as sns
fromscipy import stats

data = pd.read_csv("scores.csv")
data.head()
data['Borough'].value_counts()
############### There is no total score column, have to create it.
########In addition, find the mean score of the each district across all schools.
data['total_score'] = data['Average Score (SAT Reading)'] + \
data['Average Score (SAT Math)']    + \
data['Average Score (SAT Writing)']
data = data[['Borough', 'total_score']].dropna()
x = ['Brooklyn', 'Bronx', 'Manhattan', 'Queens', 'Staten Island']
district_dict = {}

#Assigns each test score series to a dictionary key
for district in x:
```

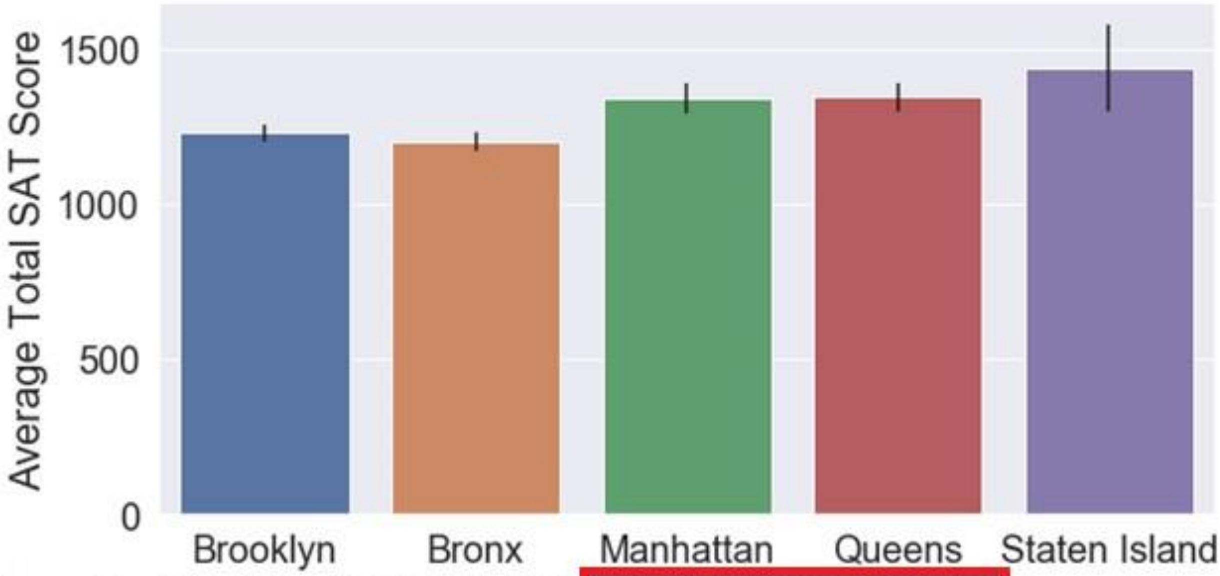
```
district_dict[district] = data[data['Borough'] == district]['total_score']
y = []
yerror = []
#Assigns the mean score and 95% confidence limit to each district
for district in x:
y.append(district_dict[district].mean())
yerror.append(1.96*district_dict[district].std()/np.sqrt(district_dict[district].shap
e[0]))
print(district + '_std : {}'.format(district_dict[district].std()))

sns.set(font_scale=1.8)
fig = plt.figure(figsize=(10,5))
ax = sns.barplot(x, y, yerr=yerror)
ax.set_ylabel('Average Total SAT Score')
plt.show()
```

**######################## Perform 1-way ANOVA**

```
print(stats.f_oneway(
district_dict['Brooklyn'], district_dict['Bronx'], \
district_dict['Manhattan'], district_dict['Queens'], \
district_dict['Staten Island']
))
districts = ['Brooklyn', 'Bronx', 'Manhattan', 'Queens', 'Staten Island']
ss_b = 0
for d in districts:
ss_b += district_dict[d].shape[0] * \
np.sum((district_dict[d].mean() - data['total_score'].mean())**2)
ss_w = 0
for d in districts:
ss_w += np.sum((district_dict[d] - district_dict[d].mean())**2)
msb = ss_b/4
msw = ss_w/(len(data)-5)
f=msb/msw
print('F_statistic: {}'.format(f))

ss_t = np.sum((data['total_score']-data['total_score'].mean())**2)
eta_squared = ss_b/ss_t
print('eta_squared: {}'.format(eta_squared))
```

## Output:

```
In [37]: runfile('E:/Research In Computing/Programs/Practical_05/Annova.py', wdir='E:/Research In
Computing/Programs/Practical_05')
Brooklyn_std : 154.8684270520867
Bronx_std : 150.39390071890668
Manhattan_std : 230.2941395363782
Queens_std : 195.25289850192115
Staten Island_std : 222.30359621222706
```




```
F_onewayResult(statistic=12.733085029201668, pvalue=1.0161974965566023e-09)
F_statistic: 12.733085029201687
eta_squared: 0.12099887621529214
```

Since theresulting pvalueis less than 0.05. The null hypothesis is rejected and conclude that there is a significant difference between the SAT scores for each district.

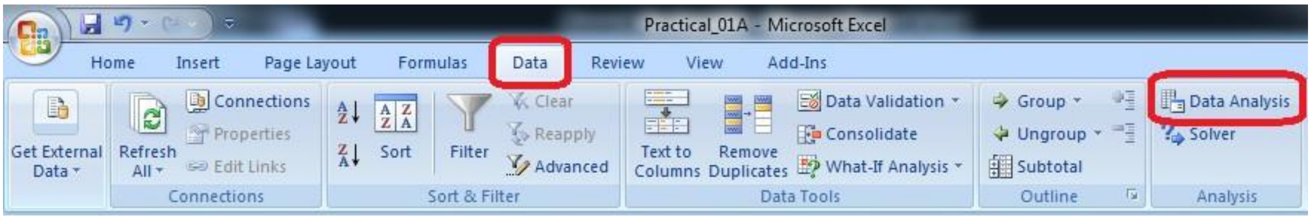
**Using Excel**

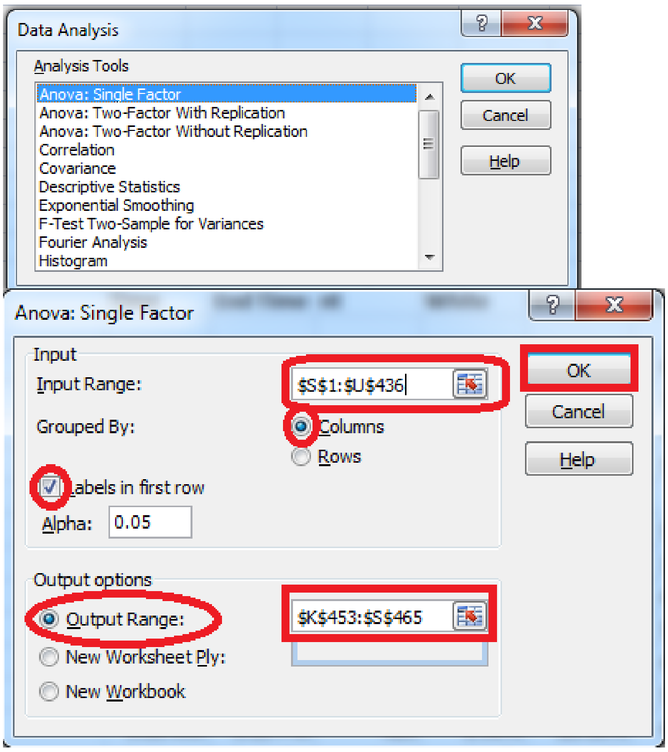
**H0 - There are no significant differences between the Subject's mean SAT scores.**

$$\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5$$

**H1 - There is a significant difference between the Subject's mean SAT scores.**

To perform ANOVA go to data □Data Analysis






**Input Range** : $S$1:$U$436*( Select columns to be analyzed in group)*

**Output Range** :$K$453:$S$465*( Can be any Range)*

| | | | | | | |
|---|---|---|---|---|---|---|
| Anova: Single Factor | | | | | | |
| | | | | | | |
| SUMMARY | | | | | | |
| *Groups* | *Count* | *Sum* | *Average* | *Variance* | | |
| Average Score (SAT Math) | 375 | 162354 | 432.944 | 5177.144 | | |
| Average Score (SAT Reading) | 375 | 159189 | 424.504 | 3829.267 | | |
| Average Score (SAT Writing) | 375 | 156922 | 418.4587 | 4166.522 | | |
| | | | | | | |
| | | | | | | |
| ANOVA | | | | | | |
| *Source of Variation* | *SS* | *df* | *MS* | *F* | *P-value* | *F crit* |
| Between Groups | 39700.57 | 2 | 19850.28 | 4.520698 | 0.01108 | 3.003745 |
| Within Groups | 4926677 | 1122 | 4390.977 | | | |
| | | | | | | |
| Total | 4966377 | 1124 | | | | |

Since theresulting pvalueis less than 0.05. The null hypothesis (H0) is rejected and conclude that there is a significant difference between the SAT scores for each subject.

## B:Perform testing of hypothesis using Two-way ANOVA.

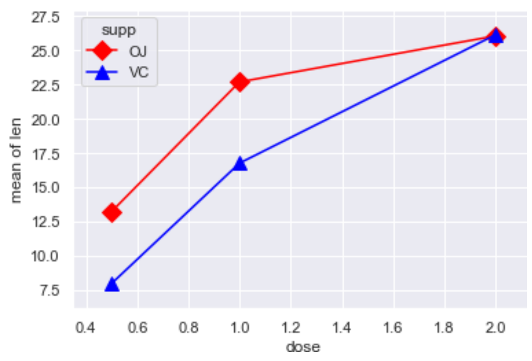
**Program Code:**

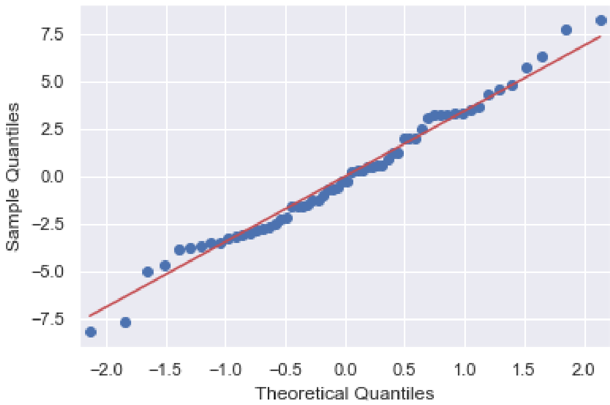
```
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
from statsmodels.graphics.factorplots import interaction_plot
import matplotlib.pyplot as plt
from scipy import stats

def eta_squared(aov):
    aov['eta_sq'] = 'NaN'
    aov['eta_sq'] = aov[:-1]['sum_sq']/sum(aov['sum_sq'])
    return aov

def omega_squared(aov):
    mse = aov['sum_sq'][-1]/aov['df'][-1]
    aov['omega_sq'] = 'NaN'
    aov['omega_sq'] = (aov[:-1]['sum_sq']-(aov[:-1]['df']*mse))/(sum(aov['sum_sq'])+mse)
    return aov

datafile = "ToothGrowth.csv"
data = pd.read_csv(datafile)
fig  =  interaction_plot(data.dose,  data.supp,  data.len,
            colors=['red','blue'], markers=['D','^'], ms=10)
N = len(data.len)
df_a = len(data.supp.unique()) - 1
df_b = len(data.dose.unique()) - 1
df_axb = df_a*df_b
df_w = N - (len(data.supp.unique())*len(data.dose.unique()))
grand_mean = data['len'].mean()
#Sum of Squares A – supp
ssq_a = sum([(data[data.supp ==l].len.mean()-grand_mean)**2 for l in data.supp])
#Sum of Squares B – supp
ssq_b = sum([(data[data.dose ==l].len.mean()-grand_mean)**2 for l in data.dose])
#Sum of Squares Total
ssq_t = sum((data.len - grand_mean)**2)
vc = data[data.supp == 'VC']
oj = data[data.supp == 'OJ']
vc_dose_means = [vc[vc.dose == d].len.mean() for d in vc.dose]
oj_dose_means = [oj[oj.dose == d].len.mean() for d in oj.dose]
ssq_w = sum((oj.len - oj_dose_means)**2) +sum((vc.len - vc_dose_means)**2)
ssq_axb = ssq_t-ssq_a-ssq_b-ssq_w
ms_a = ssq_a/df_a        #Mean Square A
ms_b = ssq_b/df_b        #Mean Square B
ms_axb = ssq_axb/df_axb        #Mean Square AXB
ms_w = ssq_w/df_w
f_a = ms_a/ms_w
f_b = ms_b/ms_w
```

```
f_axb = ms_axb/ms_w
p_a = stats.f.sf(f_a, df_a, df_w)
p_b = stats.f.sf(f_b, df_b, df_w)
p_axb = stats.f.sf(f_axb, df_axb, df_w)
results = {'sum_sq':[ssq_a, ssq_b, ssq_axb, ssq_w],
           'df':[df_a, df_b, df_axb, df_w],
           'F':[f_a, f_b, f_axb, 'NaN'],
            'PR(>F)':[p_a, p_b, p_axb, 'NaN']}
columns=['sum_sq', 'df', 'F', 'PR(>F)']

aov_table1 = pd.DataFrame(results, columns=columns,
                          index=['supp', 'dose',
                          'supp:dose', 'Residual'])
formula = 'len ~ C(supp) + C(dose) + C(supp):C(dose)'
model = ols(formula, data).fit()
aov_table = anova_lm(model, typ=2)
eta_squared(aov_table)
omega_squared(aov_table)
print(aov_table.round(4))
res = model.resid
fig = sm.qqplot(res, line='s')
plt.show()
```

**Output:**

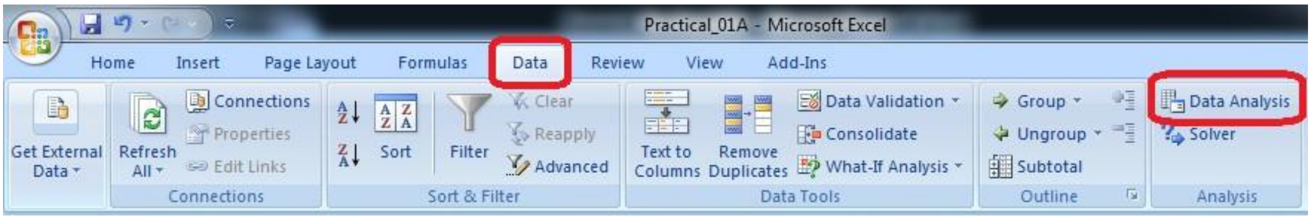
```
In [40]: runfile('K:/Research In Computing/Practical Material/Programs/
Practical_06/Annova_2_Way.py', wdir='K:/Research In Computing/Practical
Material/Programs/Practical_06')
                    sum_sq    df       F  PR(>F)  eta_sq  omega_sq
C(supp)           205.3500   1.0  15.572  0.0002  0.0595    0.0555
C(dose)          2426.4343   2.0  92.000  0.0000  0.7029    0.6926
C(supp):C(dose)   108.3190   2.0   4.107  0.0219  0.0314    0.0236
Residual          712.1060  54.0     NaN     NaN     NaN       NaN
```

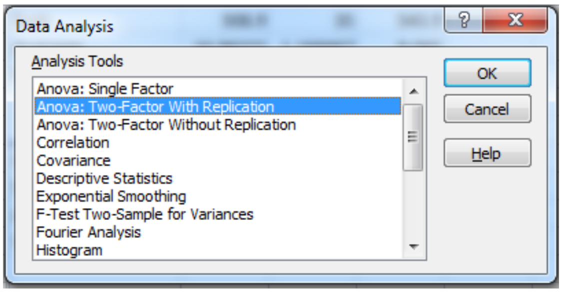


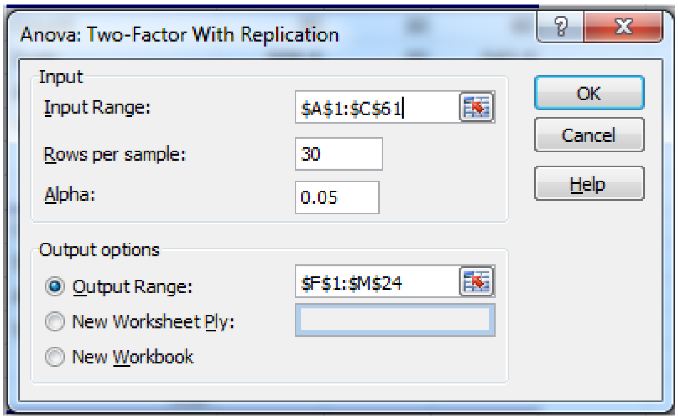




**Using Excel:**

Go to Data tab  Data Analysis




Input Range - $A$1:$C$61
Rows Per Sample – 30 (Beacause 30 Patients are given each dose)
Alpha – 0.05
Output Range - $F$1:$M$24

## Output:

| **Anova: Two-Factor With Replication** | | | | | | | |
|---|---|---|---|---|---|---|---|
| SUMMARY | len | dose | Total | | | | |
| *1* | | | | | | | |
| Count | 30 | 30 | 60 | | | | |
| Sum | 508.9 | 35 | 543.9 | | | | |
| Average | 16.96333 | 1.166667 | 9.065 | | | | |
| Variance | 68.32723 | 0.402299 | 97.22333 | | | | |
| *31* | | | | | | | |
| Count | 30 | 30 | 60 | | | | |
| Sum | 619.9 | 35 | 654.9 | | | | |
| Average | 20.66333 | 1.166667 | 10.915 | | | | |
| Variance | 43.63344 | 0.402299 | 118.2854 | | | | |
| *Total* | | | | | | | |
| Count | 60 | 60 | | | | | |
| Sum | 1128.8 | 70 | | | | | |
| Average | 18.81333 | 1.166667 | | | | | |
| Variance | 58.51202 | 0.39548 | | | | | |
| **ANOVA** | | | | | | | |

| *Source of Variation* | *SS* | *df* | *MS* | *F* | *P-value* | *F crit* |
|---|---|---|---|---|---|---|
| Sample | 102.675 | 1 | 102.675 | 3.642079 | 0.058808 | 3.922879 |
| Columns | 9342.145 | 1 | 9342.145 | 331.3838 | 8.55E-36 | 3.922879 |
| Interaction | 102.675 | 1 | 102.675 | 3.642079 | 0.058808 | 3.922879 |
| Within | 3270.193 | 116 | 28.19132 | | | |
| Total | 12817.69 | 119 | | | | |

P-value = 0.0588079 column in the ANOVA Source of Variation table at the bottom of the output. Because the p-values for both medicin dose and interaction are less than our significance level, these factors are statistically significant. On the other hand, the interaction effect is not significant because its p-value (0.0588) is greater than our significance level. Because the interaction effect is not significant, we can focus on only the main effects and not consider the interaction effect of the dose.

## B. Perform testing of hypothesis using MANOVA.

**Code:**

```
import pandas as pd
fromstatsmodels.multivariate.manova import MANOVA
df = pd.read_csv('iris.csv', index_col=0)
df.columns = df.columns.str.replace(".", "_")
df.head()
print('~~~~~~~~~ Data Set ~~~~~~~~~')
print(df)
maov = MANOVA.from_formula('Sepal_Length + Sepal_Width + \
Petal_Length + Petal_Width ~ Species', data=df)
print('~~~~~~~~~ MANOVA Test Result ~~~~~~~~~')
print(maov.mv_test())
```

**Output:**

```
In [42]: runfile('E:/Research In Computing/Programs/Practical_10/Manova_Test.py', wdir='E:/Research
In Computing/Programs/Practical_10')
~~~~~~~~~ Data Set ~~~~~~~~~
     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
..            ...          ...           ...          ...        ...
146           6.7          3.0           5.2          2.3  virginica
147           6.3          2.5           5.0          1.9  virginica
148           6.5          3.0           5.2          2.0  virginica
149           6.2          3.4           5.4          2.3  virginica
150           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
~~~~~~~~~ MANOVA Test Result ~~~~~~~~~
                  Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept         Value  Num DF  Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0170 4.0000 144.0000 2086.7720 0.0000
         Pillai's trace  0.9830 4.0000 144.0000 2086.7720 0.0000
 Hotelling-Lawley trace 57.9659 4.0000 144.0000 2086.7720 0.0000
    Roy's greatest root 57.9659 4.0000 144.0000 2086.7720 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
        Species          Value  Num DF  Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0234 8.0000 288.0000  199.1453 0.0000
         Pillai's trace  1.1919 8.0000 290.0000   53.4665 0.0000
 Hotelling-Lawley trace 32.4773 8.0000 203.4024  582.1970 0.0000
    Roy's greatest root 32.1919 4.0000 145.0000 1166.9574 0.0000
================================================================
```

# Practical 7

## A. Perform the Random sampling for the given data and analyse it.

**Example 1**: From a population of 10 women and 10 men as given in the table in Figure 1 on the left below, create a random sample of 6 people for Group 1 and a periodic sample consisting of every 3rd woman for Group 2.

You need to run the sampling data analysis tool twice, once to create Group 1 and again to create Group 2. For Group 1 you select all 20 population cells as the Input Range and Random as the Sampling Method with 6 for the Random Number of Samples. For Group 2 you select the 10 cells in the Women column as Input Range and Periodic with Period 3.

Open existing excel sheet with population data

Sample Sheet looks as given below:

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Sr. No** | **Roll No** | **Student's Name** | **Gender** | **Grade** | | **Sr. No** | **Roll No** | **Student's Name** | **Gender** | **Grade** |
| 2 | **1** | 1 | Gaborone | m | O | | **62** | 3 | Maun | f | O |
| 3 | **2** | 2 | Francistown | m | O | | **63** | 7 | Tete | f | O |
| 4 | **3** | 5 | Niamey | m | O | | **64** | 9 | Chimoio | f | O |
| 5 | **4** | 13 | Maxixe | m | O | | **65** | 11 | Pemba | f | O |
| 6 | **5** | 16 | Tema | m | O | | **66** | 14 | Chibuto | f | O |
| 7 | **6** | 17 | Kumasi | m | O | | **67** | 25 | Mampong | f | O |
| 8 | **7** | 34 | Blida | m | O | | **68** | 36 | Tlemcen | f | O |
| 9 | **8** | 35 | Oran | m | O | | **69** | 40 | Adrar | f | O |
| 10 | **9** | 38 | Saefda | m | O | | **70** | 41 | Tindouf | f | O |
| 11 | **10** | 42 | Constantine | m | O | | **71** | 46 | Skikda | f | O |
| 12 | **11** | 43 | Annaba | m | O | | **72** | 47 | Ouargla | f | O |
| 13 | **12** | 45 | Bejaefa | m | O | | **73** | 10 | Matola | f | D |
| 14 | **13** | 48 | Medea | m | O | | **74** | 20 | Legon | f | D |
| 15 | **14** | 49 | Djelfa | m | O | | **75** | 21 | Sunyani | f | D |
| 16 | **15** | 50 | Tipaza | m | O | | **76** | 72 | Teenas | f | D |
| 17 | **16** | 51 | Bechar | m | O | | **77** | 73 | Kouba | f | D |
| 18 | **17** | 54 | Mostaganem | m | O | | **78** | 75 | Hussen Dey | f | D |
| 19 | **18** | 55 | Tiaret | m | O | | **79** | 77 | Khenchela | f | D |
| 20 | **19** | 56 | Bouira | m | O | | **80** | 82 | Hassi Bahbah | f | D |
| 21 | **20** | 59 | Tebessa | m | O | | **81** | 84 | Baraki | f | D |
| 22 | **21** | 61 | El Harrach | m | O | | **82** | 91 | Boudouaou | f | D |
| 23 | **22** | 62 | Mila | m | O | | **83** | 95 | Tadjenanet | f | D |
| 24 | **23** | 65 | Fouka | m | O | | **84** | 4 | Molepolole | f | C |

## Output:

| O | P |
|---|---|
| **Male** | **Female** |
| A | A |
| A | A |
| A | A |
| B | A |
| C | B |
| C | C |
| D | C |
| D | C |
| D | C |
| D | C |
| D | D |
| D | A |
| D | B |
| D | B |
| O | D |
| O | D |

## B. Perform the Stratified sampling for the given data and analyse it.

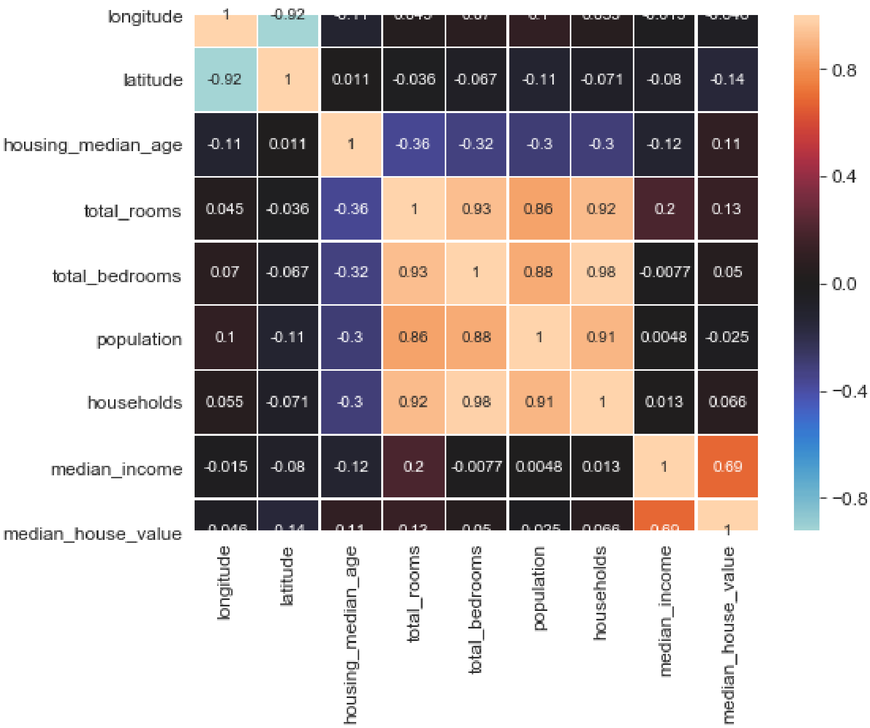
we are to carry out a **hypothetical** housing quality survey across Lagos state, Nigeria. And we looking at a total of 5000 houses (hypothetically). We don't just go to one local government and select 5000 houses, rather we ensure that the 5000 houses are a representative of the whole 20 local government areas Lagos state is comprised of. This is called stratified sampling. The population is divided into homogenous strata and the right number of instances is sampled from each stratum to guarantee that the test-set (which in this case is the 5000 houses) is a representative of the overall population. If we used random sampling, there would be a significant chance of having bias in the survey results.

**Program Code:**

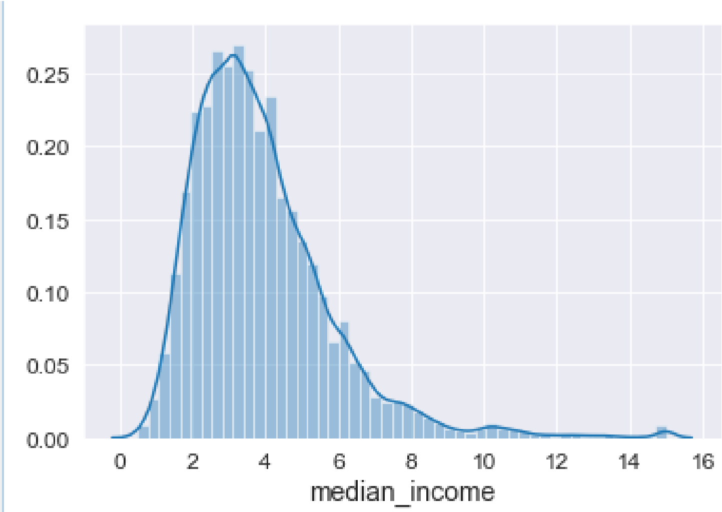
```
import pandas as pd
importnumpy as np

importmatplotlib
importmatplotlib.pyplot as plt

plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12

importseaborn as sns
color = sns.color_palette()
sns.set_style('darkgrid')

importsklearn
fromsklearn.model_selection import train_test_split

housing =pd.read_csv('housing.csv')
print(housing.head())
print(housing.info())

 #creating a heatmap of the attributes in the dataset
correlation_matrix = housing.corr()
plt.subplots(figsize=(8,6))
sns.heatmap(correlation_matrix, center=0, annot=True, linewidths=.3)


corr =housing.corr()
print(corr['median_house_value'].sort_values(ascending=False))

sns.distplot(housing.median_income)
plt.show()
```

**Output:**

```
In [28]: runfile('J:/Research In Computing/Practical Material/Programs/Practical_05/
Stratified_Sample.py', wdir='J:/Research In Computing/Practical Material/Programs/Practical_05')
   longitude  latitude  ...  median_house_value  ocean_proximity
0    -122.23     37.88  ...            452600.0         NEAR BAY
1    -122.22     37.86  ...            358500.0         NEAR BAY
2    -122.24     37.85  ...            352100.0         NEAR BAY
3    -122.25     37.85  ...            341300.0         NEAR BAY
4    -122.25     37.85  ...            342200.0         NEAR BAY

[5 rows x 10 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
None
median_house_value    1.000000
median_income         0.688075
total_rooms           0.134153
housing_median_age    0.105623
households            0.065843
total_bedrooms        0.049686
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Name: median_house_value, dtype: float64
```




There's a ton of information we can mine from the heatmap above, a couple of strongly positively correlated features and a couple of negatively correlated features. Take a look at the small bright box right in the middle of the heatmap from total_rooms on the left 'y-axis' till households and note how bright the box is as well as the highly positively correlated attributes, also note that median_income is the most correlated feature to the target which is median_house_value.




From the image above, we can see that most median incomes are clustered between $20,000 and $50,000 with some outliers going far beyond $60,000 making the distribution skew to the right.

# Practical 8

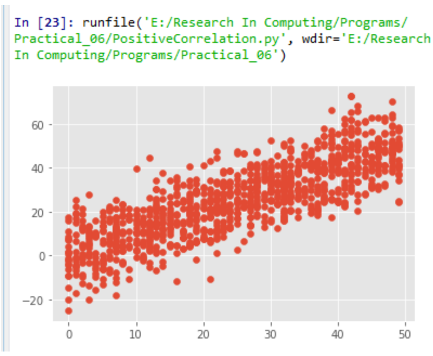
## Write a program for computing different correlation.

**Positive Correlation:**

**Code:**

```
importnumpy as np
importmatplotlib.pyplot as plt
np.random.seed(1)
# 1000 random integers between 0 and 50
x = np.random.randint(0, 50, 1000)
# Positive Correlation with some noise
y = x + np.random.normal(0, 10, 1000)
np.corrcoef(x, y)
matplotlib.style.use('ggplot')
plt.scatter(x, y)
plt.show()
```

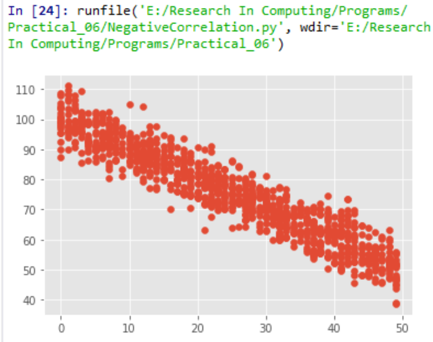
**Output:**




**Negative Correlation:**

```
importnumpy as np
importmatplotlib.pyplot as plt
np.random.seed(1)
# 1000 random integers between 0 and 50
x = np.random.randint(0, 50, 1000)
# Negative Correlation with some noise
y = 100 - x + np.random.normal(0, 5, 1000)
np.corrcoef(x, y)
plt.scatter(x, y)
plt.show()
```

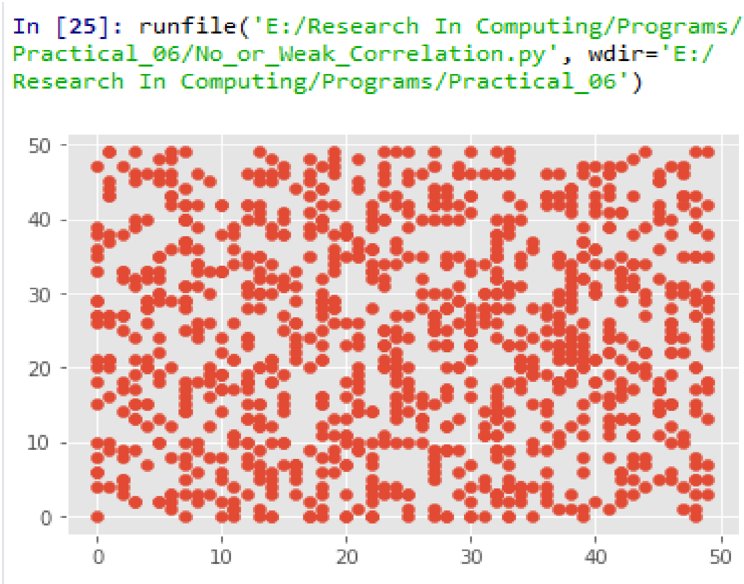
**Output:**




**No/Weak Correlation:**

importnumpy as np
importmatplotlib.pyplot as plt
np.random.seed(1)
x = np.random.randint(0, 50, 1000)
y = np.random.randint(0, 50, 1000)
np.corrcoef(x, y)
plt.scatter(x, y)
plt.show()

**Output:**



# Practical 9

## A. Write a program to Perform linear regression for prediction.

```
# -*- coding: utf-8 -*-
import Quandl, math
import numpy as np
import pandas as pd
from sklearn import preprocessing, cross_validation, svm
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from matplotlib import style
import datetime
style.use('ggplot')
df = Quandl.get("WIKI/GOOGL")
df = df[['Adj. Open', 'Adj. High', 'Adj. Low', 'Adj. Close', 'Adj. Volume']]
df['HL_PCT'] = (df['Adj. High'] - df['Adj. Low']) / df['Adj. Close'] * 100.0
df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open']) / df['Adj. Open'] * 100.0

df = df[['Adj. Close', 'HL_PCT', 'PCT_change', 'Adj. Volume']]
forecast_col = 'Adj. Close'
df.fillna(value=-99999, inplace=True)
forecast_out = int(math.ceil(0.01 * len(df)))
df['label'] = df[forecast_col].shift(-forecast_out)

X = np.array(df.drop(['label'], 1))
X = preprocessing.scale(X)
X_lately = X[-forecast_out:]
X = X[:-forecast_out]

df.dropna(inplace=True)
y = np.array(df['label'])
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.2)
clf = LinearRegression(n_jobs=-1)
clf.fit(X_train, y_train)
confidence = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)
df['Forecast'] = np.nan

last_date = df.iloc[-1].name
last_unix = last_date.timestamp()
one_day = 86400
next_unix = last_unix + one_day

for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += 86400
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

df['Adj. Close'].plot()
```

```
df['Forecast'].plot()
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

## Output:

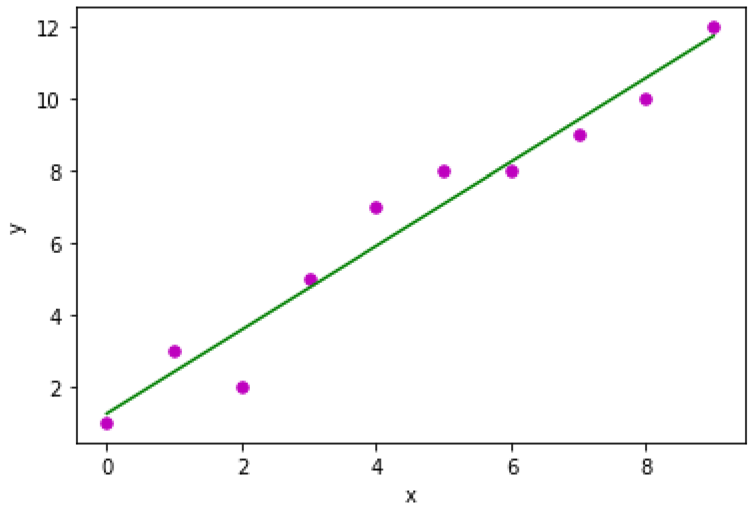

**B. Perform polynomial regression for prediction.**

```
importnumpy as np
importmatplotlib.pyplot as plt

defestimate_coef(x, y):
	# number of observations/points
	n = np.size(x)

	# mean of x and y vector
	m_x, m_y = np.mean(x), np.mean(y)

	# calculating cross-deviation and deviation about x
	SS_xy = np.sum(y*x) - n*m_y*m_x
	SS_xx = np.sum(x*x) - n*m_x*m_x

	# calculating regression coefficients
	b_1 = SS_xy / SS_xx
	b_0 = m_y - b_1*m_x

	return(b_0, b_1)

defplot_regression_line(x, y, b):
	# plotting the actual points as scatter plot
	plt.scatter(x, y, color = "m",
				marker = "o", s = 30)

	# predicted response vector
	y_pred = b[0] + b[1]*x
```

```
    # plotting the regression line
    plt.plot(x, y_pred, color = "g")

    # putting labels
    plt.xlabel('x')
    plt.ylabel('y')

    # function to show plot
    plt.show()

def main():
    # observations
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb_0 = {} b_1 = {}".format(b[0], b[1]))

    # plotting regression line
    plot_regression_line(x, y, b)

if__name__== "_main_":
    main()
```

**Output:**

```
In [22]: runfile('E:/Research In Computing/Programs/
Practical_07/Practical_7B.py', wdir='E:/Research In
Computing/Programs/Practical_07')
Estimated coefficients:
b_0 = 1.2363636363636363  b_1 = 1.1696969696969697
```



# Practical 10

## A. Write a program for multiple linear regression analysis.

**Step #1:** Data Pre Processing

a) Importing The Libraries.
b) Importing the Data Set.
c) Encoding the Categorical Data.
d) Avoiding the Dummy Variable Trap.
e) Splitting the Data set into Training Set and Test Set.

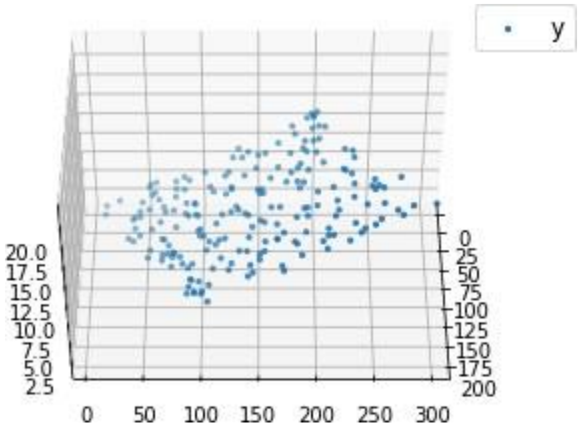
**Step #2:** Fitting Multiple Linear Regression to the Training set

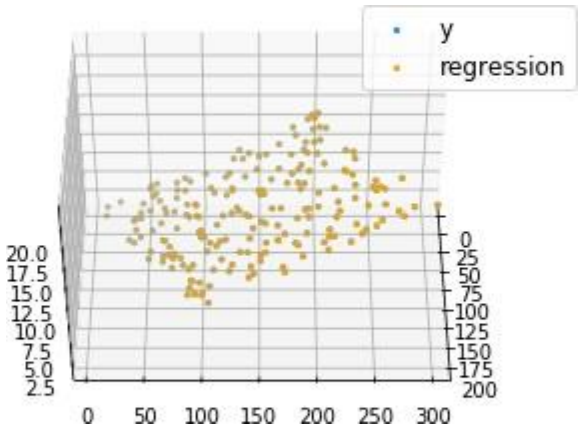
**Step #3:** Predicting the Test set results.

```
importnumpy as np
importmatplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
importmatplotlib.pyplot as plt
defgenerate_dataset(n):
        x = []
        y = []
        random_x1 = np.random.rand()
        random_x2 = np.random.rand()
        fori in range(n):
                x1 = i
                x2 = i/2 + np.random.rand()*n
                x.append([1, x1, x2])
                y.append(random_x1 * x1 + random_x2 * x2 + 1)
        returnnp.array(x), np.array(y)
x, y = generate_dataset(200)
mpl.rcParams['legend.fontsize'] = 12
fig = plt.figure()
ax = fig.gca(projection ='3d')
ax.scatter(x[:, 1], x[:, 2], y, label ='y', s = 5)
ax.legend()
ax.view_init(45, 0)
plt.show()
defmse(coef, x, y):
        returnnp.mean((np.dot(x, coef) - y)**2)/2
def gradients(coef, x, y):
        returnnp.mean(x.transpose()*(np.dot(x, coef) - y), axis = 1)
defmultilinear_regression(coef, x, y, lr, b1 = 0.9, b2 = 0.999, epsilon = 1e-8):
        prev_error = 0
        m_coef = np.zeros(coef.shape)
        v_coef = np.zeros(coef.shape)
        moment_m_coef = np.zeros(coef.shape)
        moment_v_coef = np.zeros(coef.shape)
        t = 0
        while True:
                error = mse(coef, x, y)
                if abs(error - prev_error) <= epsilon:
                        break
                prev_error = error
```

```
                    grad = gradients(coef, x, y)
                    t += 1
                    m_coef = b1 * m_coef + (1-b1)*grad
                    v_coef = b2 * v_coef + (1-b2)*grad**2
                    moment_m_coef = m_coef / (1-b1**t)
                    moment_v_coef = v_coef / (1-b2**t)

                    delta = ((lr / moment_v_coef**0.5 + 1e-8) *
                                        (b1 * moment_m_coef + (1-b1)*grad/(1-b1**t)))
                    coef = np.subtract(coef, delta)
          returncoef
coef = np.array([0, 0, 0])
c = multilinear_regression(coef, x, y, 1e-1)
fig = plt.figure()
ax = fig.gca(projection ='3d')
ax.scatter(x[:, 1], x[:, 2], y, label ='y',
                                        s = 5, color ="dodgerblue")
ax.scatter(x[:, 1], x[:, 2], c[0] + c[1]*x[:, 1] + c[2]*x[:, 2],
                                                  label ='regression', s = 5, color ="orange")
ax.view_init(45, 0)
ax.legend()
plt.show()
```

**Output:**



## B. Perform logistic regression analysis.

Logistic regression is a classification method built on the same concept as linear regression. With linear regression, we take linear combination of explanatory variables plus an intercept term to arrive at a prediction.
In this example we will use a logistic regression model to predict survival.

**Program Code:**

```
import os
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import scipy.stats as stats
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import metrics

matplotlib.style.use('ggplot')
plt.figure(figsize=(9,9))

def sigmoid(t):                     # Define the sigmoid function
   return (1/(1 + np.e**(-t)))

plot_range = np.arange(-6, 6, 0.1)

y_values = sigmoid(plot_range)

# Plot curve
plt.plot(plot_range, # X-axis range
        y_values,        # Predicted values
        color="red")
titanic_train = pd.read_csv("titanic_train.csv")    # Read the data
char_cabin = titanic_train["Cabin"].astype(str)     # Convert cabin to str
new_Cabin = np.array([cabin[0] for cabin in char_cabin]) # Take first letter

titanic_train["Cabin"] = pd.Categorical(new_Cabin) # Save the new cabin var

# Impute median Age for NA Age values
new_age_var = np.where(titanic_train["Age"].isnull(), # Logical check
                  28,                     # Value if check is true
                  titanic_train["Age"])     # Value if check is false

titanic_train["Age"] = new_age_var

label_encoder = preprocessing.LabelEncoder()

# Convert Sex variable to numeric
encoded_sex = label_encoder.fit_transform(titanic_train["Sex"])

# Initialize logistic regression model
log_model = linear_model.LogisticRegression()
```

```
# Train the model
log_model.fit(X = pd.DataFrame(encoded_sex),
              y = titanic_train["Survived"])

# Check trained model intercept
print(log_model.intercept_)

# Check trained model coefficients
print(log_model.coef_)

# Make predictions
preds = log_model.predict_proba(X= pd.DataFrame(encoded_sex))
preds = pd.DataFrame(preds)
preds.columns = ["Death_prob", "Survival_prob"]

# Generate table of predictions vs Sex
pd.crosstab(titanic_train["Sex"], preds.ix[:, "Survival_prob"])

# Convert more variables to numeric
encoded_class = label_encoder.fit_transform(titanic_train["Pclass"])
encoded_cabin = label_encoder.fit_transform(titanic_train["Cabin"])

train_features = pd.DataFrame([encoded_class,
                               encoded_cabin,
                               encoded_sex,
                               titanic_train["Age"]]).T

# Initialize logistic regression model
log_model = linear_model.LogisticRegression()

# Train the model
log_model.fit(X = train_features ,
              y = titanic_train["Survived"])

# Check trained model intercept
print(log_model.intercept_)

# Check trained model coefficients
print(log_model.coef_)

# Make predictions
preds = log_model.predict(X= train_features)

# Generate table of predictions vs actual
pd.crosstab(preds,titanic_train["Survived"])

log_model.score(X = train_features ,
                y = titanic_train["Survived"])
```

```
metrics.confusion_matrix(y_true=titanic_train["Survived"], # True labels
                    y_pred=preds) # Predicted labels

# View summary of common classification metrics
print(metrics.classification_report(y_true=titanic_train["Survived"],
                        y_pred=preds) )

# Read and prepare test data
titanic_test = pd.read_csv("titanic_test.csv")    # Read the data

char_cabin = titanic_test["Cabin"].astype(str)     # Convert cabin to str

new_Cabin = np.array([cabin[0] for cabin in char_cabin]) # Take first letter

titanic_test["Cabin"] = pd.Categorical(new_Cabin) # Save the new cabin var

# Impute median Age for NA Age values
new_age_var = np.where(titanic_test["Age"].isnull(), # Logical check
                  28,                       # Value if check is true
                  titanic_test["Age"])      # Value if check is false

titanic_test["Age"] = new_age_var

# Convert test variables to match model features
encoded_sex = label_encoder.fit_transform(titanic_test["Sex"])
encoded_class = label_encoder.fit_transform(titanic_test["Pclass"])
encoded_cabin = label_encoder.fit_transform(titanic_test["Cabin"])

test_features = pd.DataFrame([encoded_class,
                        encoded_cabin,encoded_sex,titanic_test["Age"]]).T

# Make test set predictions
test_preds = log_model.predict(X=test_features)

# Create a submission for Kaggle
submission = pd.DataFrame({"PassengerId":titanic_test["PassengerId"],
                      "Survived":test_preds})

# Save submission to CSV
submission.to_csv("tutorial_logreg_submission.csv",
              index=False)        # Do not save index values

print(pd)
```

## Output:

| Survival_prob | 0.193110906347 | 0.729443792051 |
|---|---|---|
| Sex | | |
| female | 0 | 312 |
| male | 577 | 0 |

The table shows that the model predicted a survival chance of roughly 19% for males and 73% for females.

```
             precision    recall  f1-score   support

          0       0.82      0.85      0.83       549
          1       0.74      0.70      0.72       340

avg / total       0.79      0.79      0.79       889
```

For the Titanic competition, accuracy is the scoring metric used to judge the competition, so we don't have to worry too much about other metrics.

| Survived | 0 | 1 |
|---|---|---|
| row_0 | | |
| 0 | 467 | 103 |
| 1 | 82 | 237 |

The table above shows the classes our model predicted vs. true values of the Survived variable.

This logistic regression model has an accuracy score of 0.75598 which is actually worse than the accuracy of the simplistic women survive, men die model (0.76555).

**Example 2:**

The dataset is related to direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict whether the client will subscribe (1/0) to a term deposit (variable y). The dataset provides the bank customers' information. It includes 41,188 records and 21 fields.

**Input variables**

1. **age** (numeric)
2. **job :** type of job (categorical: "admin", "blue-collar", "entrepreneur", "housemaid", "management", "retired", "self-employed", "services", "student", "technician", "unemployed", "unknown")
3. **marital :** marital status (categorical: "divorced", "married", "single", "unknown")

4. **education** (categorical: “basic.4y”, “basic.6y”, “basic.9y”, “high.school”, “illiterate”, “professional.course”, “university.degree”, “unknown”)
5. **default:** has credit in default? (categorical: “no”, “yes”, “unknown”)
6. **housing:** has housing loan? (categorical: “no”, “yes”, “unknown”)
7. **loan:** has personal loan? (categorical: “no”, “yes”, “unknown”)
8. **contact:** contact communication type (categorical: “cellular”, “telephone”)
9. **month:** last contact month of year (categorical: “jan”, “feb”, “mar”, …, “nov”, “dec”)
10. **day_of_week:** last contact day of the week (categorical: “mon”, “tue”, “wed”, “thu”, “fri”)
11. **duration:** last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y=’no’). The duration is not known before a call is performed, also, after the end of the call, y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model
12. **campaign:** number of contacts performed during this campaign and for this client (numeric, includes last contact)
13. **pdays:** number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14. **previous:** number of contacts performed before this campaign and for this client (numeric)
15. **poutcome:** outcome of the previous marketing campaign (categorical: “failure”, “nonexistent”, “success”)
16. **emp.var.rate:** employment variation rate — (numeric)
17. **cons.price.idx:** consumer price index — (numeric)
18. **cons.conf.idx**: consumer confidence index — (numeric)
19. **euribor3m:** euribor 3 month rate — (numeric)
20. **nr.employed:** number of employees — (numeric)

**Predict variable (desired target):**
**y — has the client subscribed a term deposit?**
**(binary: “1”, means “Yes”, “0” means “No”)**

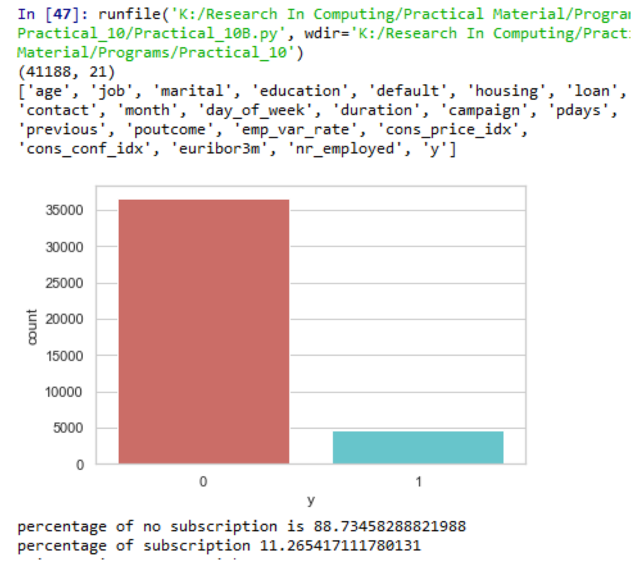
**Program Code:**

```
# -*- coding: utf-8 -*-
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
data = pd.read_csv('bank.csv', header=0)
data = data.dropna()
print(data.shape)
print(list(data.columns))
data['education'].unique()
data['education']=np.where(data['education'] =='basic.9y', 'Basic', data['education'])
```

```
data['education']=np.where(data['education'] =='basic.6y', 'Basic', data['education'])
data['education']=np.where(data['education'] =='basic.4y', 'Basic', data['education'])
data['education'].unique()
data['y'].value_counts()

sns.countplot(x='y', data=data, palette='hls')
plt.show();
plt.savefig('Practical10B-plot.jpeg')

count_no_sub = len(data[data['y']==0])
count_sub = len(data[data['y']==1])
pct_of_no_sub = count_no_sub/(count_no_sub+count_sub)
print("percentage of no subscription is", pct_of_no_sub*100)
pct_of_sub = count_sub/(count_no_sub+count_sub)
print("percentage of subscription", pct_of_sub*100)

data.groupby('y').mean()
data.groupby('job').mean()
data.groupby('marital').mean()
data.groupby('education').mean()

########### Purchase Frequency for Job Title
pd.crosstab(data.job,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Job Title')
plt.xlabel('Job')
plt.ylabel('Frequency of Purchase')
plt.savefig('purchase_fre_job')

###################### Marital Status vs Purchase
table=pd.crosstab(data.marital,data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Marital Status vs Purchase')
plt.xlabel('Marital Status')
plt.ylabel('Proportion of Customers')
plt.savefig('mariral_vs_pur_stack')

############       Education vs Purchase
table=pd.crosstab(data.education,data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Education vs Purchase')
plt.xlabel('Education')
plt.ylabel('Proportion of Customers')
plt.savefig('edu_vs_pur_stack')

pd.crosstab(data.day_of_week,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Frequency of Purchase')
plt.savefig('pur_dayofweek_bar')
```

```
############ Purchase Frequency for Month
pd.crosstab(data.month,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Month')
plt.xlabel('Month')
plt.ylabel('Frequency of Purchase')
plt.savefig('pur_fre_month_bar')

########### Age Purchase frequency pattern
data.age.hist()
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.savefig('hist_age')
```

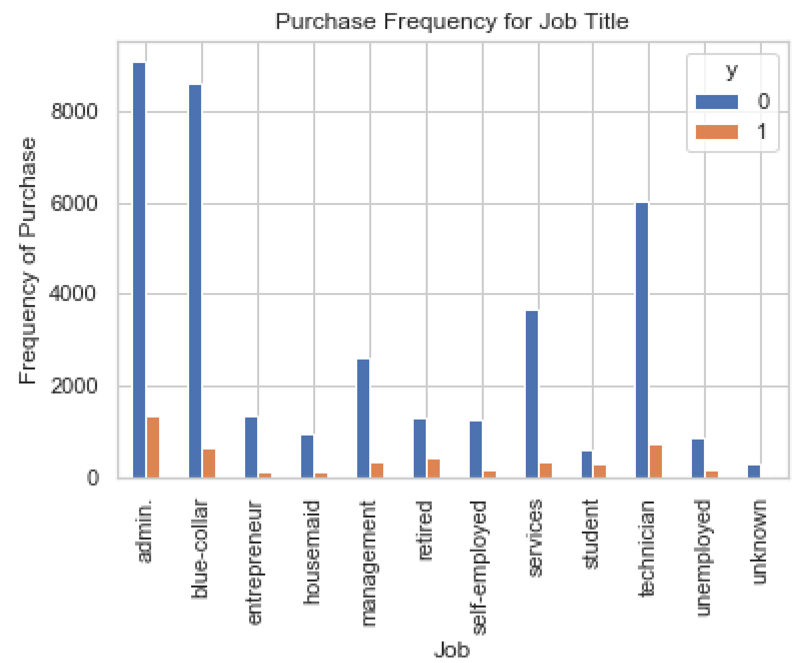
**Output: -**



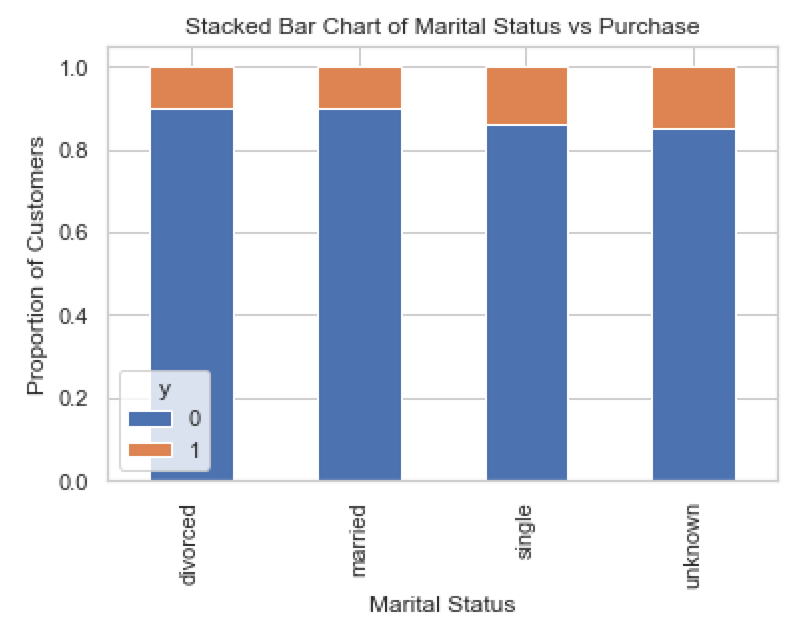

**PERCENTAGE OF NO SUBSCRIPTION IS 88.73458288821988**
**PERCENTAGE OF SUBSCRIPTION 11.265417111780131**

Our classes are imbalanced, and the ratio of no-subscription to subscription instances is 89:11.

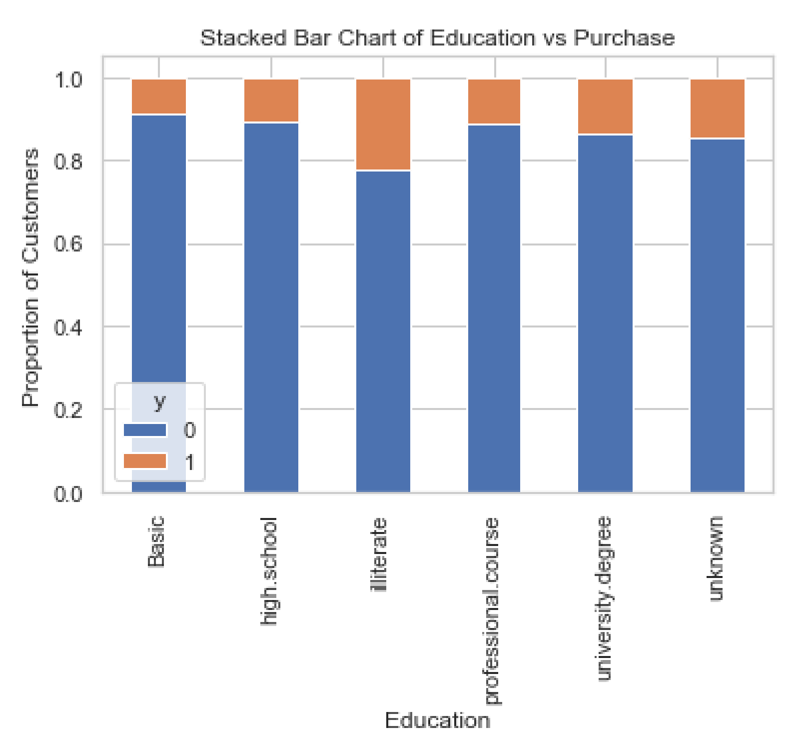
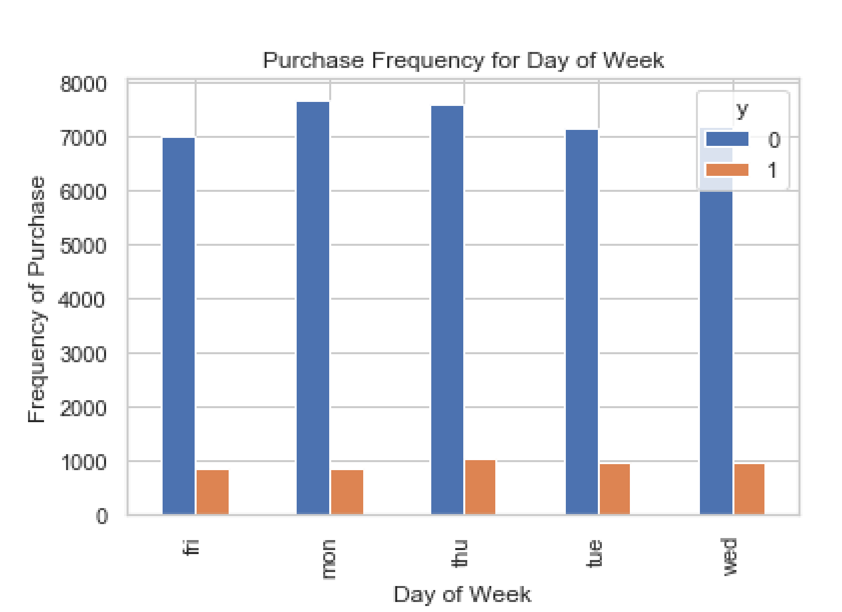
- The average age of customers who bought the term deposit is higher than that of the customers who didn’t.
- The pdays (days since the customer was last contacted) is understandably lower for the customers who bought it. The lower the pdays, the better the memory of the last call and hence the better chances of a sale.
- Surprisingly, campaigns (number of contacts or calls made during the current campaign) are lower for customers who bought the term deposit.

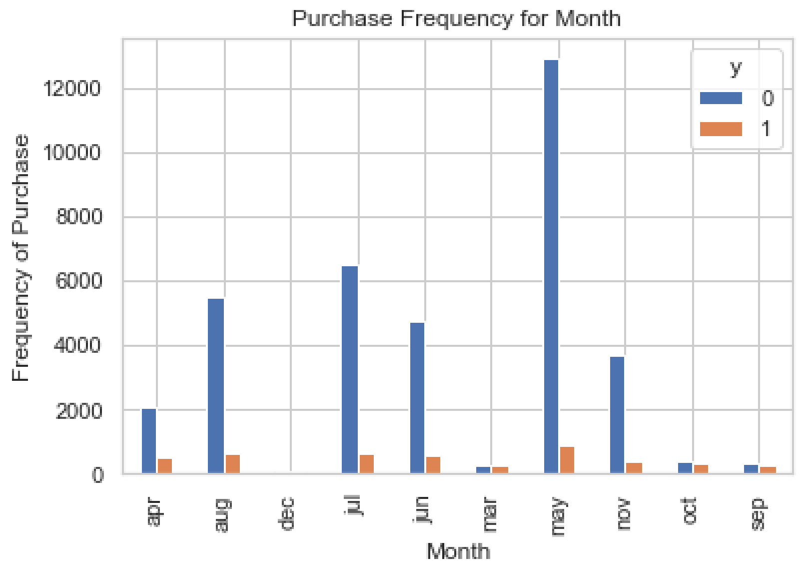
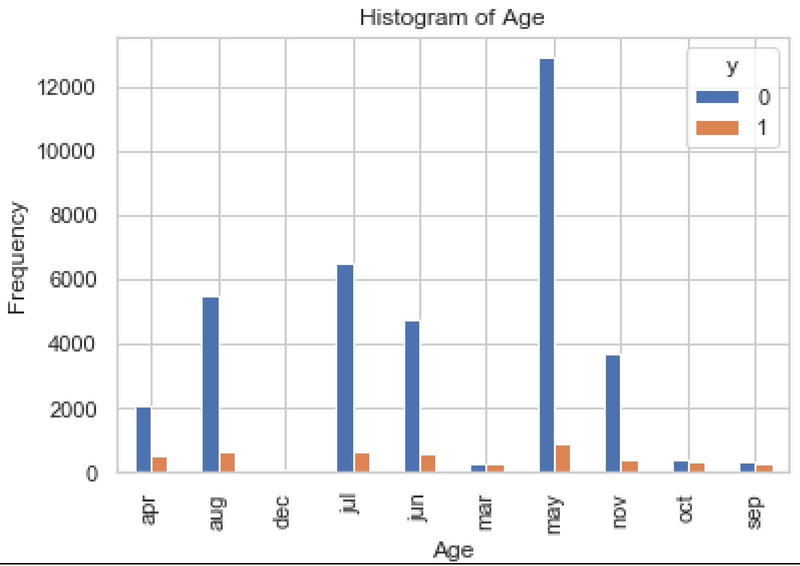



The frequency of purchase of the deposit depends a great deal on the job title. Thus, the job title can be a good predictor of the outcome variable.




The marital status does not seem a strong predictor for the outcome variable.

| | |
|---|---|
|  | Education seems a good predictor of the outcome variable. |
|  | Day of week may not be a good predictor of the outcome. |

| | |
|---|---|
|  | Month might be a good predictor of the outcome variable. |
|  | Most of the customers of the bank in this dataset are in the age range of 30–40. |