Bahria University,

Karachi Campus

COURSE CODE: CSL-487
Introduction to Data Science Lab
TERM: SPRING 2024, CLASS: BSCS- (A)

Submitted By:

Ahmed Farrukh	78993
(Name)	**(Reg. No.)**

Submitted To:

Engr. Rabia Amjad

Signed **Remarks:** **Score:**

INDEX

S NO	DATE	LAB NO	LAB OBJECTIVE	SIGN
1	23-2-24	1	Introduction to anaconda (Jupyter Notebook)	
2	23-2-24	2	Introduction to python programming	
3	1-3-24	3	Getting to know R basic concepts (Variable , Data Types , Vector , Dataframes , lists and matrices)	
4	8-3-24	4	Implement linear Regression	
5	15-3-24	5	Implementation of Logistic Regression	
6	22-3-24	6	Implementation of KNN	
7	29-3-24	7	Sample Code for spam. Apply filtering spam on different datasets.	
8	26-4-24	8	Learn and implement APISs and other for tools for the scrapping the web.	
9	3-5-24	9	Learn and implement crowdsourcing	
10	10-5-24	10	Apply feature selection datasets (PCA)	
11	17-5-24	11	Build your own recommendation system	
12	24-5-24	12	Financial Data Analytics	
13	31-5-24	13	Visualization of complex dataset	

Department of COMPUTER SCIENCE
Bahria University
Discovering Knowledge

CSL-487: Introduction to Data Science Lab

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

LAB 1

Exercise

Exercise 1

Write a Python program to replace the last element in a list with another list.

Expected output:

```
List 1 :  [86, 39, 58, 62, 9, 18]
List 2 :  [24, 46, 21, 8]
Replaced List :  [86, 39, 58, 62, 9, 24, 46, 21, 8]
```

CODE:

```
list1 = [86, 39, 50, 62, 9, 18]
list2 = [24, 46, 21, 8]
list1[-1:]=list2
print(list1)
```

OUTPUT:

```
In [2]: list1 = [86, 39, 50, 62, 9, 18]
        list2 = [24, 46, 21, 8]
        list1[-1:]=list2
        print(list1)

        [86, 39, 50, 62, 9, 24, 46, 21, 8]
```

Exercise 2

Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string

Expected output:

```
Actual String : Python Programming
Resultant String:  Pyng
```

CODE:

```
str1 = "latibulate"
print(str1[0:2]+str1[-2:])
```

OUTPUT:

```
In [4]: str1 = "latibulate"
        print(str1[0:2]+str1[-2:])

        late
```

Exercise 3

Write a python program that takes three dictionaries as input (user or static) and concatenates them into one dictionary, and prints it.

Expected output:

```
Dictionary 1
{1: 10, 2: 20}

Dictionary 2
{3: 30, 4: 40}

Dictionary 3
{5: 50, 6: 60}

After Concatenating all three
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

CODE:

```
dict_1={1:10,2:20}
dict_2={3:30,4:40}
dict_3={5:50,6:60}


merged_dict = {**dict_1, **dict_2, **dict_3}
print("Merged Dictionary: ", merged_dict
```

OUTPUT:

```
In [7]: dict_1={1:10,2:20}
        dict_2={3:30,4:40}
        dict_3={5:50,6:60}

        merged_dict = {**dict_1, **dict_2, **dict_3}
        print("Merged Dictionary: ", merged_dict)

        Merged Dictionary:  {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

Exercise 4

Create a python program that takes user input and prints square, square root of that number, number powered to 3 (num^3).

Expected output:

```
Enter a number please :36
Square of  36  is:  1296
Square root of  36  is:  6.0
Power 3 of  36  is:  46656
```

CODE:

```
import math
num = float(input("Enter a number: "))
sq = num ** 2
sq_root = math.sqrt(num)
cube = num** 3

print(f"The square of {num} is {sq}")
print(f"The square root of {num} is {sq_root}")
print(f"The cube of {num} is {cube}")
```

OUTPUT:

In [9]:
```
import math
num = float(input("Enter a number: "))
sq = num ** 2
sq_root = math.sqrt(num)
cube = num** 3

print(f"The square of {num} is {sq}")
print(f"The square root of {num} is {sq_root}")
print(f"The cube of {num} is {cube}")
```

```
Enter a number: 5
The square of 5.0 is 25.0
The square root of 5.0 is 2.23606797749979
The cube of 5.0 is 125.0
```

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Lab :02

EXERSICE:

Exercise 01:

Calculate Standard Deviation and Variance using python user-defined functions.
Hint:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{N}}$$

CODE:

```
import math
def variance(val):
    numb = len(val)
    m = sum(val) / numb
    devi = [(x - m) ** 2 for x in val]
    variance = sum(devi) / numb
    return variance

variance = variance([6, 6, 3, 9, 4, 3, 6, 9, 7, 8])
print("Variance: ",variance)

def stddev(val):
    stdev = math.sqrt(val)
    return stdev

print("Standard Deviation: ",stddev(variance))
```

OUTPUT:

In [21]:
```
import math
def variance(val):
    numb = len(val)
    m = sum(val) / numb
    devi = [(x - m) ** 2 for x in val]
    variance = sum(devi) / numb
    return variance

variance = variance([6, 6, 3, 9, 4, 3, 6, 9, 7, 8])
print("Variance: ",variance)

def stddev(val):
    stdev = math.sqrt(val)
    return stdev

print("Standard Deviation: ",stddev(variance))
```

```
Variance:  4.49
Standard Deviation:  2.118962010041709
```

Exercise 02:

Take input string name from the user and display the names having more than 5 letters.

CODE:

```
names = input("Enter names: ").split()

conditioned_names = [name for name in names if len(name)>5]

print("Names with more than 5 letters",conditioned_names)
```

OUTPUT:

In [16]:
```
names = input("Enter names: ").split()

conditioned_names = [name for name in names if len(name)>5]

print("Names with more than 5 letters",conditioned_names)
```

```
Enter names: Dasha Merdiya Zverev Sascha
Names with more than 5 letters ['Merdiya', 'Zverev', 'Sascha']
```

Exercise 03:

Write a Python function that checks whether a passed string is palindrome or not.

CODE:

```
str = input("Enter a palindrome: ")

if str == str[::-1]:
    print("The string is a palindrome")
else:
    print("The string is not a palindrome")
```

OUTPUT:

```
In [3]: str = input("Enter a palindrome: ")

        if str == str[::-1]:
            print("The string is a palindrome")
        else:
            print("The string is not a palindrome")

Enter a palindrome: level
The string is a palindrome
```

Exercise 04:

You input a number and the function created checks whether the number belongs to the Fibonacci sequence or not.

CODE:

```
num=int(input("Enter the number: "))
a=0
b=1
c=1
if num==0 or num==1:
    print(num," is a Fabionacci number.")
else:
    while a<num:
        a=b+c
        c=b
        b=a
    if a==num:
        print(num," is a Fabionacci number.")
    else:
        print(num," is not a Fabionacci number.")
```

OUTPUT:

In [6]:
```
num=int(input("Enter the number: "))
a=0
b=1
c=1
if num==0 or num==1:
    print(num," is a Fabionacci number.")
else:
    while a<num:
        a=b+c
        c=b
        b=a
    if a==num:
        print(num," is a Fabionacci number.")
    else:
        print(num," is not a Fabionacci number.")
```

```
Enter the number: 8
8  is a Fabionacci number.
```

Exercise 05:

Write a Python program to check whether the given integer is a multiple of 5 and 7 both using function.

CODE:

```
num=int(input("Enter a number: "))

def check(num):
  if num%5==0 and num%7==0:
    print("Number is multiple of 5 and 7.")
  else:
    print("Number is not multiple of 5 and 7.")

check(num)
```

OUTPUT:

```
In [15]: num=int(input("Enter a number: "))

         def check(num):
           if num%5==0 and num%7==0:
             print("Number is multiple of 5 and 7.")
           else:
             print("Number is not multiple of 5 and 7.")

         check(num)

         Enter a number: 70
         Number is multiple of 5 and 7.
```

Department of COMPUTER SCIENCE
Bahria University
Discovering Knowledge

CSL-487: Introduction to Data Science Lab

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Lab 03

Task 01:

Write a program to find minimum and maximum of number series in R.

CODE:

```
> find_min_max <- function(numbers) {
+     minimum <- numbers[1]
+     maximum <- numbers[1]
+
+     for (num in numbers) {
+         if (num < minimum) {
+             minimum <- num
+         }
+         if (num > maximum) {
+             maximum <- num
+         }
+     }
+
+     return(list(minimum = minimum, maximum = maximum))
+ }
>
> numbers <- c(5, 10, 2, 8, 15, 3)
> result <- find_min_max(numbers)
>
> cat("Minimum:", result$minimum, "\n")
> cat("Maximum:", result$maximum, "\n")
```

OUTPUT:

```
> find_min_max <- function(numbers) {
+     minimum <- numbers[1]
+     maximum <- numbers[1]
+
+     for (num in numbers) {
+         if (num < minimum) {
+             minimum <- num
+         }
+         if (num > maximum) {
+             maximum <- num
+         }
+     }
+
+     return(list(minimum = minimum, maximum = maximum))
+ }
>
> numbers <- c(5, 10, 2, 8, 15, 3)
> result <- find_min_max(numbers)
>
> cat("Minimum:", result$minimum, "\n")
Minimum: 2
> cat("Maximum:", result$maximum, "\n")
Maximum: 15
```

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Task 02:

Write a program to test whether the value of the element of a given vector greater than 10 or not. Return Statement in Logical (TRUE or FALSE)

CODE:

```
> check_condition <- function(vector) {
+     result <- vector > 10
+     return(result)
+ }
>
> my_vector <- c(5, 12, 8, 15, 3)
> result <- check_condition(my_vector)
>
> print(result)
```

OUTPUT:

```
> check_condition <- function(vector) {
+     result <- vector > 10
+     return(result)
+ }
>
> my_vector <- c(5, 12, 8, 15, 3)
> result <- check_condition(my_vector)
>
> print(result)
[1] FALSE  TRUE FALSE  TRUE FALSE
```

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Task 03:

Find Sum, Mean and Product of Vector in R Programming.

CODE:

```
> find_sum <- function(numbers) {
+     result <- 0
+     for (num in numbers) {
+         result <- result + num
+     }
+     return(result)
+ }
>
> find_mean <- function(numbers) {
+     sum <- find_sum(numbers)
+     mean <- sum / length(numbers)
+     return(mean)
+ }
>
> find_product <- function(numbers) {
+     result <- 1
+     for (num in numbers) {
+         result <- result * num
+     }
+     return(result)
+ }
>
> numbers <- c(5, 10, 2, 8, 15, 3)
>
> sum_result <- find_sum(numbers)
> mean_result <- find_mean(numbers)
> product_result <- find_product(numbers)
>
> cat("Sum:", sum_result, "\n")
> cat("Mean:", mean_result, "\n")
> cat("Product:", product_result, "\n")
```

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

OUTPUT:

```
> find_sum <- function(numbers) {
+     result <- 0
+     for (num in numbers) {
+         result <- result + num
+     }
+     return(result)
+ }
>
> find_mean <- function(numbers) {
+     sum <- find_sum(numbers)
+     mean <- sum / length(numbers)
+     return(mean)
+ }
>
> find_product <- function(numbers) {
+     result <- 1
+     for (num in numbers) {
+         result <- result * num
+     }
+     return(result)
+ }
>
> numbers <- c(5, 10, 2, 8, 15, 3)
>
> sum_result <- find_sum(numbers)
> mean_result <- find_mean(numbers)
> product_result <- find_product(numbers)
>
> cat("Sum:", sum_result, "\n")
Sum: 43
> cat("Mean:", mean_result, "\n")
Mean: 7.166667
> cat("Product:", product_result, "\n")
Product: 36000
>
```

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Task 04:

Write a program to find common elements from multiple vectors.

CODE:

```
> find_common_elements <- function(vectors) {
+     common_elements <- vectors[[1]]
+
+     for (i in 2:length(vectors)) {
+         common_elements <- intersect(common_elements, vectors[[i]])
+     }
+
+     return(common_elements)
+ }
>
> vector1 <- c(1, 2, 3, 4, 5)
> vector2 <- c(3, 4, 5, 6, 7)
> vector3 <- c(5, 6, 7, 8, 9)
>
> common_elements <- find_common_elements(list(vector1, vector2, vector3))
>
> cat("Common Elements:", common_elements, "\n")
```

OUTPUT:

```
> find_common_elements <- function(vectors) {
+     common_elements <- vectors[[1]]
+
+     for (i in 2:length(vectors)) {
+         common_elements <- intersect(common_elements, vectors[[i]])
+     }
+
+     return(common_elements)
+ }
>
> vector1 <- c(1, 2, 3, 4, 5)
> vector2 <- c(3, 4, 5, 6, 7)
> vector3 <- c(5, 6, 7, 8, 9)
>
> common_elements <- find_common_elements(list(vector1, vector2, vector3))
>
> cat("Common Elements:", common_elements, "\n")
Common Elements: 5
```

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Task 05:

Write a program to create three vectors numeric data, character data and logical data. Display the content of the vectors and their type.

CODE:

```
> numeric_vector <- c(1, 2, 3, 4, 5)
> character_vector <- c("apocalypse", "cardigan", "wrecked")
> logical_vector <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
>
> cat("Numeric Vector: ",numeric_vector,"\n")
> cat("Type:", class(numeric_vector), "\n\n")
>
> cat("Character Vector: ",character_vector,"\n")
> cat("Type:", class(character_vector), "\n\n")
>
> cat("Logical Vector: ",logical_vector,"\n")
> cat("Type:", class(logical_vector), "\n\n")
```

OUTPUT:

```
> numeric_vector <- c(1, 2, 3, 4, 5)
> character_vector <- c("apocalypse", "cardigan", "wrecked")
> logical_vector <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
>
> cat("Numeric Vector: ",numeric_vector,"\n")
Numeric Vector:  1 2 3 4 5
> cat("Type:", class(numeric_vector), "\n\n")
Type: numeric

>
> cat("Character Vector: ",character_vector,"\n")
Character Vector:  apocalypse cardigan wrecked
> cat("Type:", class(character_vector), "\n\n")
Type: character

>
> cat("Logical Vector: ",logical_vector,"\n")
Logical Vector:  TRUE FALSE TRUE TRUE FALSE
> cat("Type:", class(logical_vector), "\n\n")
Type: logical
```

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Task 06:

Write a program to create a vector of a specified type and length. Create vector of numeric, complex, logical and character types of length 6.

CODE:

```
> numeric_vector <- 1:6

> complex_vector <- as.complex(1:6)
> logical_vector <- c(TRUE, FALSE, TRUE, FALSE, TRUE, FALSE)
> character_vector <- c("apocalypse", "snap", "cardigan", "wrecked", "shots",
"roots")
>
> cat("Numeric vector:", numeric_vector, "\n")
> cat("Complex vector:", complex_vector, "\n")
> cat("Logical vector:", logical_vector, "\n")
> cat("Character vector:", character_vector, "\n")
>
> cat("Type of numeric vector:", typeof(numeric_vector), "\n")
> cat("Type of complex vector:", typeof(complex_vector), "\n")
> cat("Type of logical vector:", typeof(logical_vector), "\n")
> cat("Type of character vector:", typeof(character_vector), "\n")
```

OUTPUT:

```
> numeric_vector <- 1:6
> complex_vector <- as.complex(1:6)
> logical_vector <- c(TRUE, FALSE, TRUE, FALSE, TRUE, FALSE)
> character_vector <- c("apocalypse", "snap", "cardigan", "wrecked", "shots", "roots")
>
> cat("Numeric vector:", numeric_vector, "\n")
Numeric vector: 1 2 3 4 5 6
> cat("Complex vector:", complex_vector, "\n")
Complex vector: 1+0i 2+0i 3+0i 4+0i 5+0i 6+0i
> cat("Logical vector:", logical_vector, "\n")
Logical vector: TRUE FALSE TRUE FALSE TRUE FALSE
> cat("Character vector:", character_vector, "\n")
Character vector: apocalypse snap cardigan wrecked shots roots
>
> cat("Type of numeric vector:", typeof(numeric_vector), "\n")
Type of numeric vector: integer
> cat("Type of complex vector:", typeof(complex_vector), "\n")
Type of complex vector: complex
> cat("Type of logical vector:", typeof(logical_vector), "\n")
Type of logical vector: logical
> cat("Type of character vector:", typeof(character_vector), "\n")
Type of character vector: character
>
```

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Task 07:

https://www.kaggle.com/datasets/majyhain/height-of-male-and-female-by-country-2022

Write a program to read the .csv file and create data frame show 5 head and 5 tail records from the given dataset.

CODE:

```
> data <- read.csv(file="C:/Users/spring2018/Downloads/data_file.csv")
> head(data, n=5)
> tail(data, n=5)
>
```

OUTPUT:

```
> data <- read.csv(file="C:/Users/spring2018/Downloads/data_file.csv")
> head(data, n=5)
  X Brand       Model.Name Processor Operating.System Storage  RAM
1 0    HP     15s-fq5007TU   Core i3  Windows 11 Home  512 GB 8 GB
2 1    HP     15s-fy5003TU   Core i3  Windows 11 Home  512 GB 8 GB
3 2 Apple 2020 Macbook Air        M1   Mac OS Big Sur  256 GB 8 GB
4 3 Apple 2020 Macbook Air        M1   Mac OS Big Sur  256 GB 8 GB
5 4 Apple 2020 Macbook Air        M1   Mac OS Big Sur  256 GB 8 GB
           Screen.Size Touch_Screen     Price
1 39.62 cm (15.6 Inch)           No â,'38,990
2 39.62 cm (15.6 Inch)           No â,'37,990
3 33.78 cm (13.3 inch)           No â,'70,990
4 33.78 cm (13.3 inch)           No â,'70,990
5 33.78 cm (13.3 inch)           No â,'70,990
> tail(data, n=5)
      X   Brand              Model.Name         Processor Operating.System Storage
833 832      HP                  255 G8 Ryzen 5 Hexa Core  Windows 11 Home  512 GB
834 833    DELL           Inspiron 7430           Core i3  Windows 11 Home    1 TB
835 834     MSI Katana 17 B13UCXK-256IN           Core i7  Windows 11 Home    4 TB
836 835 Infinix                   XL25           Core i5  Windows 11 Home  512 GB
837 836      HP             13-BE0030AU Ryzen 5 Hexa Core  Windows 10 Home  512 GB
      RAM          Screen.Size Touch_Screen     Price
833  8 GB 39.62 cm (15.6 inch)           No â,'42,990
834  8 GB   35.56 cm (14 inch)          Yes â,'60,490
835 16 GB 43.94 cm (17.3 Inch)           No â,'88,990
836  8 GB 39.62 cm (15.6 Inch)           No â,'37,990
837 16 GB 33.78 cm (13.3 inch)           No â,'70,500
>
```

Department of COMPUTER SCIENCE
Bahria University
Discovering Knowledge

CSL-487: Introduction to Data Science Lab
Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

LAB 04

Task 01: Predicting House Prices

You are a real estate agent working in a competitive market. You want to develop a model to help you estimate the prices of houses quickly and accurately. You have collected data on various features of houses sold in the past year, including transaction date, house age, distance to the nearest MRT station, number of convenience stores, latitude, longitude and house price of unit area. Write a python program on the given dataset Real estate.csv and implement linear regression model to check best fit line and perform following task:

1. Data analysis

In [1]:
```
import pandas as pd
data=pd.read_csv("Real estate.csv")
data.head()
```

Out[1]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

In [2]:
```
data.tail()
```

Out[2]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
409	410	2013.000	13.7	4082.01500	0	24.94155	121.50381	15.4
410	411	2012.667	5.6	90.45606	9	24.97433	121.54310	50.0
411	412	2013.250	18.8	390.96960	7	24.97923	121.53986	40.6
412	413	2013.000	8.1	104.81010	5	24.96674	121.54067	52.5
413	414	2013.500	6.5	90.45606	9	24.97433	121.54310	63.9

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   No                                      414 non-null    int64  
 1   X1 transaction date                     414 non-null    float64
 2   X2 house age                            414 non-null    float64
 3   X3 distance to the nearest MRT station  414 non-null    float64
 4   X4 number of convenience stores         414 non-null    int64  
 5   X5 latitude                             414 non-null    float64
 6   X6 longitude                            414 non-null    float64
 7   Y house price of unit area              414 non-null    float64
dtypes: float64(6), int64(2)
memory usage: 26.0 KB
```

In [4]:
```
data.describe()
```

Out[4]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
count	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000
mean	207.500000	2013.148971	17.712560	1083.885689	4.094203	24.969030	121.533361	37.980193
std	119.655756	0.281967	11.392485	1262.109595	2.945562	0.012410	0.015347	13.606488
min	1.000000	2012.667000	0.000000	23.382840	0.000000	24.932070	121.473530	7.600000
25%	104.250000	2012.917000	9.025000	289.324800	1.000000	24.963000	121.528085	27.700000
50%	207.500000	2013.167000	16.100000	492.231300	4.000000	24.971100	121.538630	38.450000
75%	310.750000	2013.417000	28.150000	1454.279000	6.000000	24.977455	121.543305	46.600000
max	414.000000	2013.583000	43.800000	6488.021000	10.000000	25.014590	121.566270	117.500000

In [5]:
```
data.shape
```

Out[5]: (414, 8)

In [7]:
```
data.size
```

Out[7]: 3312

In [8]:
```
data.corr()
```

Out[8]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
No	1.000000	-0.048658	-0.032808	-0.013573	-0.012699	-0.010110	-0.011059	-0.028587
X1 transaction date	-0.048658	1.000000	0.017549	0.060880	0.009635	0.035058	-0.041082	0.087491
X2 house age	-0.032808	0.017549	1.000000	0.025622	0.049593	0.054420	-0.048520	-0.210567
X3 distance to the nearest MRT station	-0.013573	0.060880	0.025622	1.000000	-0.602519	-0.591067	-0.806317	-0.673613
X4 number of convenience stores	-0.012699	0.009635	0.049593	-0.602519	1.000000	0.444143	0.449099	0.571005
X5 latitude	-0.010110	0.035058	0.054420	-0.591067	0.444143	1.000000	0.412924	0.546307
X6 longitude	-0.011059	-0.041082	-0.048520	-0.806317	0.449099	0.412924	1.000000	0.523287
Y house price of unit area	-0.028587	0.087491	-0.210567	-0.673613	0.571005	0.546307	0.523287	1.000000

In [9]:
```
data.drop('No',axis=1)
```

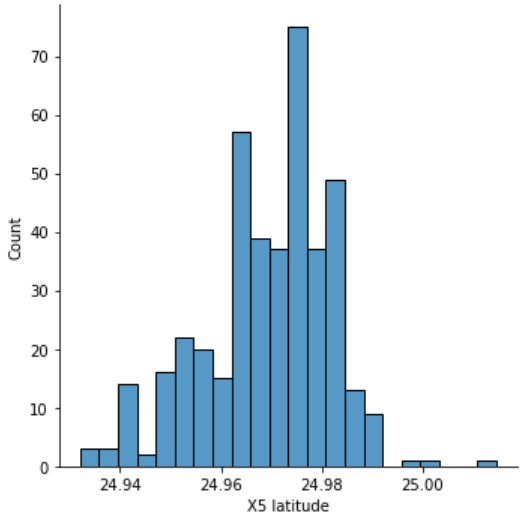
Out[9]:

	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1
...	...	...	...	...	...	...	...
409	2013.000	13.7	4082.01500	0	24.94155	121.50381	15.4
410	2012.667	5.6	90.45606	9	24.97433	121.54310	50.0
411	2013.250	18.8	390.96960	7	24.97923	121.53986	40.6
412	2013.000	8.1	104.81010	5	24.96674	121.54067	52.5
413	2013.500	6.5	90.45606	9	24.97433	121.54310	63.9

414 rows × 7 columns

In [14]:
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.displot(data['X5 latitude'])
```

Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x1ba2beef850>
```



2. Feature selection

```
In [23]: X=data[['X1 transaction date','X2 house age','X3 distance to the nearest MRT station','X4 number of convenience stores','X5 latit
         Y=data['Y house price of unit area']
```

3. Model implementation

```
In [24]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3)

In [25]: from sklearn.linear_model import LinearRegression
         model=LinearRegression()
         model.fit(x_train,y_train)

Out[25]: LinearRegression()
```

4. Predicting house prices

```
In [26]: y_predict=model.predict(x_test)
         print("Predicted values of Y on the basis of X_test",y_predict)

Predicted values of Y on the basis of X_test [47.87919496 47.71620402 35.14548465 36.51986093 16.6208     42.97171857
 43.32146431 46.5891701  48.09015703 47.32531071 47.499733   17.20211008
 33.8115883  42.83744397 47.2885124   9.45290229 13.58958989 45.15318374
 29.93276514  8.13613068 27.78318561 36.83736254 30.18551249 43.3968406
 12.86449304 43.24633014 37.70726615 41.83888165 40.76629937 44.67262739
 37.70726615 28.72333797 48.43223999 35.0089455  29.84461373 18.45921545
 51.05535567 45.74149756 30.01588702 35.32748699 44.29253281 36.86325886
 51.3700199  39.50603636 12.5150773  36.61295212 44.83180317 40.20514751
 32.09916884 11.96770134 38.76641828 49.006734   43.13557691 12.1122588
 40.62819756 46.73098251 44.86277108 46.33980121 35.29982013 30.48817891
 16.59597996 33.77480267 40.15502096 45.56144499 39.96854409 41.34008577
 32.49299589 32.88925721 43.37496524 30.10518465 50.86374968 51.03541527
 30.38635592 32.17395967 43.86924861 19.5585045  44.05693812 46.01712363
 45.99038246 36.9519965  33.10442304 48.38908304 42.89862863 24.7919965
 37.92386903 34.45181538 36.09803475 43.8363646  40.15502096 48.14161188
  7.59761803 23.4453339  44.57586671 44.06825253 13.50586548 39.71758923
 35.29815922 30.37346715 41.97095211 35.74525178 51.38106292 28.87668471
 12.31175023 38.40722819 52.14028688 35.88326898 44.81501327 40.81056832
 31.91055016 46.25547046 40.46423462 13.56388079 10.79003963 34.15443164
 15.14494447 41.27195188 43.24550244 31.69209317 41.41673057 14.73331748
 42.3950075  43.56215175 37.65377444 16.23687942 35.45201553]
```

5. Evaluate performance

In [28]:
```
from sklearn import metrics
print("Training accuracy",model.score(x_train,y_train))
print("Testing accuracy",model.score(x_test,y_test))
```

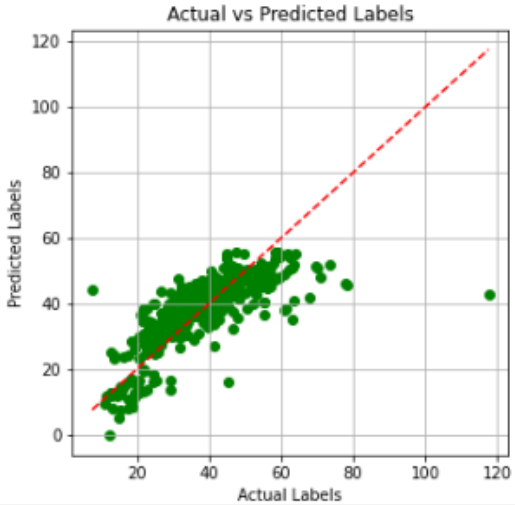
```
Training accuracy 0.5548772968335667
Testing accuracy 0.6231825713846785
```

In [30]:
```
print("Mean absolute error",metrics.mean_absolute_error(y_predict,y_test))
print("Mean squared error",metrics.mean_squared_error(y_predict,y_test))
```

```
Mean absolute error 6.155202620948804
Mean squared error 63.82217055129963
```

6. Visualize best fit line

In [34]:
```
import numpy as np
coefficients=model.coef_
predicted_values=np.dot(X,coefficients)+model.intercept_
plt.figure(figsize=(5,5))
plt.scatter(Y,predicted_values, color='green')
plt.plot([Y.min(),Y.max()],[Y.min(),Y.max()],linestyle='--',color='red')
plt.xlabel('Actual Labels')
plt.ylabel('Predicted Labels')
plt.title('Actual vs Predicted Labels')
plt.grid(True)
plt.show()
```



Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Task 02: Predicting salary

You are an HR manager at a tech company tasked with developing a salary prediction model to assist in determining appropriate compensation for new hires and promotions. Your company values transparency and fairness in salary negotiations, and they believe that a data-driven approach will help achieve these goals. You have been provided with a dataset containing information on employees' years of experience and their corresponding salaries. Write a python program on the given dataset and implement linear regression model to check best fit line and perform following task:

1. Data analysis

```
In [1]: import pandas as pd
        data=pd.read_csv("salaries.csv")
        data.head()
```

Out[1]:

	YearsOfExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891

```
In [2]: data.tail()
```

Out[2]:

	YearsOfExperience	Salary
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   YearsOfExperience  30 non-null     float64
 1   Salary             30 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 608.0 bytes
```

In [4]:
```
data.describe()
```

Out[4]:

	YearsOfExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [5]:
```
data.shape
```

Out[5]: (30, 2)

In [6]:
```
data.size
```

Out[6]: 60

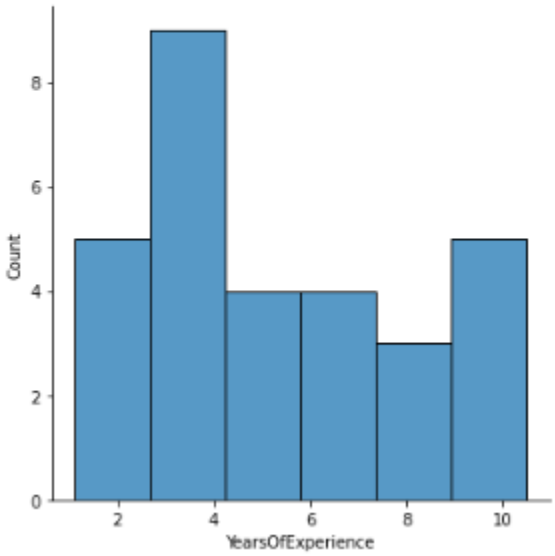
```
In [7]: data.corr()
```

Out[7]:

	YearsOfExperience	Salary
YearsOfExperience	1.000000	0.978242
Salary	0.978242	1.000000

```
In [9]: import matplotlib.pyplot as plt
        import seaborn as sns

        sns.displot(data['YearsOfExperience'])
```

```
Out[9]: <seaborn.axisgrid.FacetGrid at 0x17e13af0f40>
```



2. Feature selection

```
In [15]: X=data[['YearsOfExperience']]
         Y=data['Salary']
```

3. Model implementation

```
In [18]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3)

In [19]: from sklearn.linear_model import LinearRegression
         model=LinearRegression()
         model.fit(x_train,y_train)

Out[19]: LinearRegression()
```

4. Predict salary of a person having experience of **7.5** years.

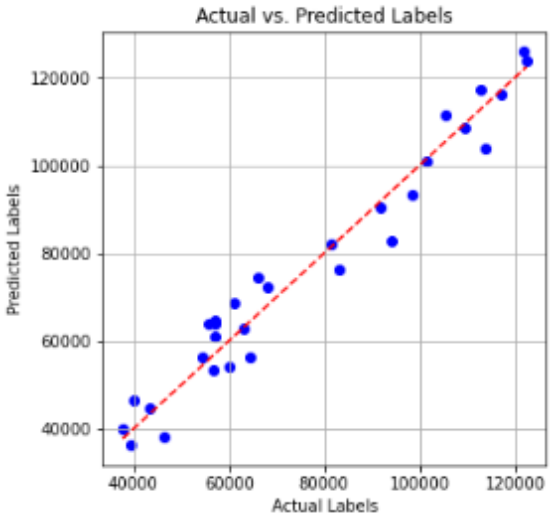
```
In [30]: y_predict2=model.predict([[7.5]])
         print("Predicted values of Y on the basis of X_test",y_predict2)

         Predicted values of Y on the basis of X_test [97224.71589501]
```

5. Evaluate performance

```
In [31]: from sklearn import metrics

         print("Training Accuracy", model.score(x_train,y_train))
         print("Testing Accuracy", model.score(x_test,y_test))

         Training Accuracy 0.934619640785993
         Testing Accuracy 0.983639809810399

In [32]: y_predict=model.predict(x_test)
         print("Predicted values of Y on the basis of X_test",y_predict)

         Predicted values of Y on the basis of X_test [ 46649.21107154  39969.42741561  38060.91779963 123943.85051874
          125852.36013472 108675.77359089  90544.93223908  56191.75915144
           72414.09088727]

In [33]: print("Mean absolute error",metrics.mean_absolute_error(y_predict,y_test))
         print("Mean squared error",metrics.mean_squared_error(y_predict,y_test))

         Mean absolute error 3427.2306166339768
         Mean squared error 17816110.597757563
```

6. Visualize best fit line

In [34]:
```
import numpy as np
coefficients = model.coef_

predicted_values = np.dot(X, coefficients) + model.intercept_

plt.figure(figsize=(5,5))
plt.scatter(Y, predicted_values, color='blue')
plt.plot([Y.min(), Y.max()], [Y.min(), Y.max()], linestyle='--', color='red')
plt.xlabel('Actual Labels')
plt.ylabel('Predicted Labels')
plt.title('Actual vs. Predicted Labels')
plt.grid(True)
plt.show()
```



Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

	A	B
1	YearsofExperience	Salary
2	1.1	39343
3	1.3	46205
4	1.5	37731
5	2	43525
6	2.2	39891
7	2.9	56642
8	3	60150
9	3.2	54445
10	3.2	64445
11	3.7	57189
12	3.9	63218
13	4	55794
14	4	56957
15	4.1	57081
16	4.5	61111
17	4.9	67938
18	5.1	66029
19	5.3	83088
20	5.9	81363
21	6	93940
22	6.8	91738
23	7.1	98273
24	7.9	101302
25	8.2	113812
26	8.7	109431
27	9	105582
28	9.5	116969
29	9.6	112635
30	10.3	122391
31	10.5	121872

Lab 05

Task 01: Credit Card Fraud

You work as a data scientist for a financial institution that issues credit cards to customers. The company is concerned about credit card fraud, which can lead to significant financial losses. To mitigate fraud risk, the company wants to develop a predictive model to identify fraudulent transactions accurately based on dataset: creditcard.csv. Write a Python program that implements logistic regression model to detect credit card fraud and perform following task:

1. Data analysis and preprocessing of data.

In [32]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("creditcard.csv")
data.head()
```

Out[32]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.1
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.1
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.3
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.6
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.2

5 rows × 31 columns

2. Perform data visualization

In [33]:
```
from sklearn.model_selection import train_test_split
X = data.drop('Class', axis=1)
y = data['Class']

X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.3)
```

In [34]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

In [35]:
```
print("X_train shape:", X_train.shape)
print("Y_train shape:", Y_train.shape)
print("X_test shape:", X_test.shape)
print("Y_test shape:", Y_test.shape)
```

```
X_train shape: (199364, 30)
Y_train shape: (199364,)
X_test shape: (85443, 30)
Y_test shape: (85443,)
```

In [36]:
```
X_train = X_train.values
Y_train = Y_train.values
X_test = X_test.values
Y_test = Y_test.values

X_train = X_train.T
Y_train = Y_train.reshape(1, X_train.shape[1])

print("X_train shape:", X_train.shape)
print("Y_train shape:", Y_train.shape)
print("X_test shape:", X_test.shape)
print("Y_test shape:", Y_test.shape)
```

```
X_train shape: (30, 199364)
Y_train shape: (1, 199364)
X_test shape: (85443, 30)
Y_test shape: (85443,)
```

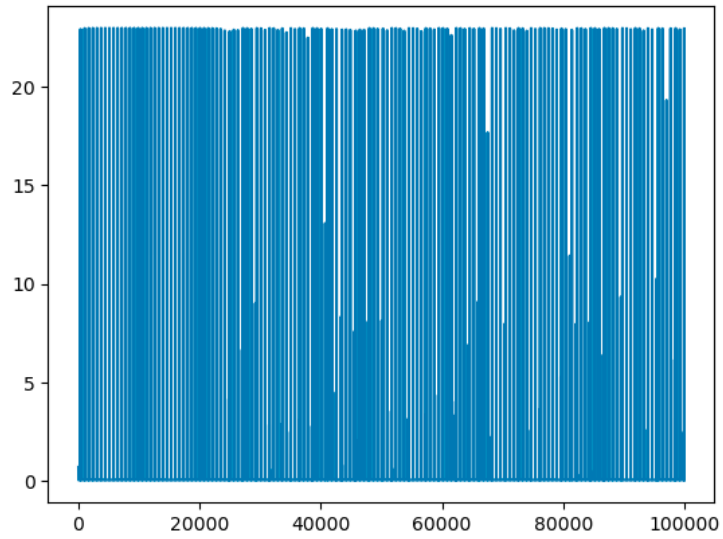
3. Implement Logistic regression using sigmoid function and Gradient decent.

In [37]:
```
def sigmoid(x):
    max_limit = 100
    min_limit = -100
    x = np.clip(x, min_limit, max_limit)
    return 1 / (1 + np.exp(-x))


def model(X, Y, learning_rate, iterations):
    m = X_train.shape[1]
    n = X_train.shape[0]

    W = np.zeros((n,1))
    B = 0

    cost_list = []
    for i in range(iterations):
        Z = np.dot(W.T, X) + B
        A = sigmoid(Z)

        epsilon = 1e-10


        cost = -(1/m) * np.sum(Y * np.log(A + epsilon) + (1 - Y) * np.log(1 - A + epsilon))


        dW = (1/m)*np.dot(A-Y, X.T)
        dB = (1/m)*np.sum(A - Y)
        W = W - learning_rate*dW.T
        B = B - learning_rate*dB
        cost_list.append(cost)

        if(i%(iterations/10) == 0):
            print("cost after ", i, "iteration is : ", cost)

    return W, B, cost_list
```

```
iterations = 100000
learning_rate = 0.0015
W, B, cost_list = model(X_train, Y_train, learning_rate, iterations)
```

```
cost after  0 iteration is :  0.6931471803599453
cost after  10000 iteration is :  0.03973936867455107
cost after  20000 iteration is :  0.040885772863602364
cost after  30000 iteration is :  0.04050330895911904
cost after  40000 iteration is :  0.041386533166516005
cost after  50000 iteration is :  0.041061797597881346
cost after  60000 iteration is :  0.04019279360224354
cost after  70000 iteration is :  0.04003740978192132
cost after  80000 iteration is :  0.039846304001362125
cost after  90000 iteration is :  0.039846304001362125
```

In [40]:
```
plt.plot(np.arange(iterations), cost_list)
plt.show()
```



4. Calculate accuracy score, precision, recall and fi-score

In [44]:
```
def accuracy(X, Y, W, B):
    W = W.reshape(-1, 1)

    Z = np.dot(W.T, X) + B
    A = sigmoid(Z)
    A = A > 0.5
    A = np.array(A, dtype='int64')

    acc = (np.sum(A == Y) / Y.shape[0]) * 100

    print("Accuracy of the model is : ", round(acc, 2), "%")

accuracy(X_test.T, Y_test, W, B)
```

```
Accuracy of the model is :  99.82 %
```

In [58]:
```
from sklearn.metrics import precision_score, recall_score

precision = precision_score(Y_test, Y_pred)

recall = recall_score(Y_test, Y_pred)

print("Precision:", precision)
print("Recall:", recall)
```

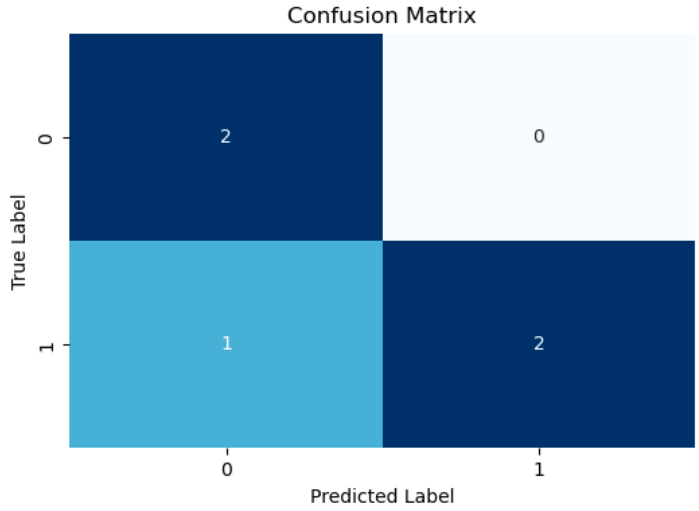
```
Precision: 1.0
Recall: 0.6666666666666666
```

In [59]:
```
from sklearn.metrics import f1_score

f1 = f1_score(Y_test, Y_pred)

print("F1-score:", f1)
```

```
F1-score: 0.8
```

5. Evaluate performance using confusion matrix

In [61]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

conf_matrix = confusion_matrix(Y_test, Y_pred)

plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



Task 02: Titanic Passengers Survival

You are a data scientist working for a research institute interested in analyzing the Titanic passenger dataset to uncover patterns and insights regarding the factors influencing survival rates during the tragic sinking of the ship. Your objective is to develop a logistic regression model to predict passenger survival based on dataset: Titanic.csv. Write a Python program that implements logistic regression model to detect credit card fraud and perform following task:

1. Data analysis and preprocessing of data.

In [26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("titanic.csv")
data.head()
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [27]:
```
data['Age'].fillna(data['Age'].mean(),inplace=True)
```

In [28]:
```
data['Age'].isna().sum()
```

Out[28]:
```
0
```

In [29]:
```
data.drop(['Cabin','PassengerId','Name','Ticket'],axis=1,inplace=True)
```

In [30]:
```
data.head()
```

Out[30]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

In [35]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
data['Sex'] = label_encoder.fit_transform(data['Sex'])
data['Embarked'] = label_encoder.fit_transform(data['Embarked'])

data.head()
```

Out[35]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.0	1	0	7.2500	2
1	1	1	0	38.0	1	0	71.2833	0
2	1	3	0	26.0	0	0	7.9250	2
3	1	1	0	35.0	1	0	53.1000	2
4	0	3	1	35.0	0	0	8.0500	2

In [36]:
```
from sklearn.model_selection import train_test_split
x = data.drop('Survived', axis=1)
y = data['Survived']

X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.3)
```

2. Perform data visualization

In [36]:
```
from sklearn.model_selection import train_test_split
x = data.drop('Survived', axis=1)
y = data['Survived']

X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.3)
```

3. Implement Logistic regression using SKLearn.

In [37]:
```
from sklearn.linear_model import LogisticRegression

logistic_model = LogisticRegression(max_iter=1000)

logistic_model.fit(X_train, Y_train)

Y_pred_train = logistic_model.predict(X_train)
Y_pred_test = logistic_model.predict(X_test)
```

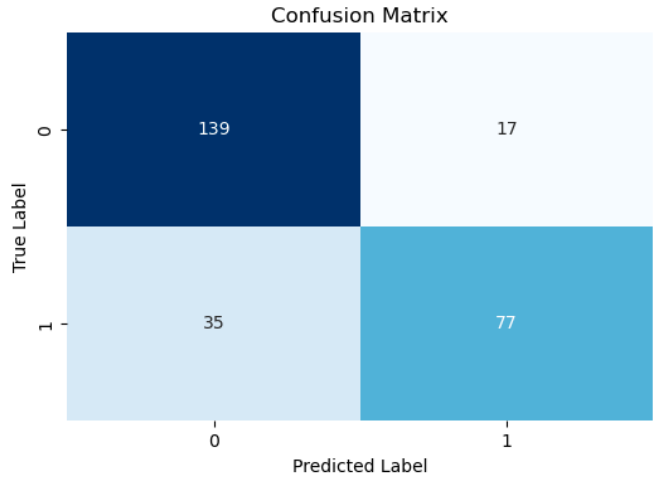
4. Calculate accuracy score, precision, recall and fi-score

In [41]:
```
from sklearn.metrics import classification_report

print(classification_report(Y_test,Y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       156
           1       0.82      0.69      0.75       112

    accuracy                           0.81       268
   macro avg       0.81      0.79      0.79       268
weighted avg       0.81      0.81      0.80       268
```

5. Evaluate performance using confusion matrix

In [43]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

conf_matrix = confusion_matrix(Y_test, Y_pred_test)

plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Lab 06

Task 01

You've been assigned a critical project at a leading healthcare analytics firm: analyzing a comprehensive dataset of diabetes patient records. Your primary objective is to leverage the power of machine learning to develop a predictive model that can identify patients at risk of developing complications associated with diabetes. Write a python program that implements KNN on dataset given below and perform following task:

1. Data analysis

```
In [46]: import pandas as pd
         data=pd.read_csv("diabetes.csv")

         data.head()
```

Out[46]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [47]: data.shape
```

```
Out[47]: (768, 9)
```

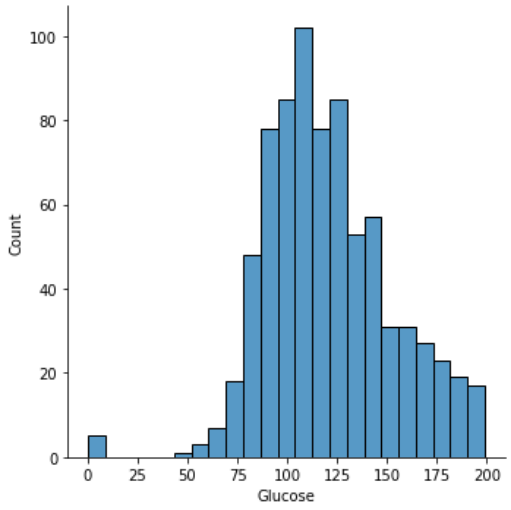
```
In [48]: data.info
```

```
Out[48]: <bound method DataFrame.info of      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]>
```

2. Perform data visualization.

```
In [49]: import matplotlib.pyplot as plt
         import seaborn as sns

         sns.displot(data['Glucose'])
```

```
Out[49]: <seaborn.axisgrid.FacetGrid at 0x1dc34acc8b0>
```



3. Feature selection

```
In [50]: X = data[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]
         Y = data['Outcome']
```

```
In [51]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test=train_test_split(X,Y,test_size=0.3)
```

4. Model implementation

```
In [52]: from sklearn.neighbors import KNeighborsClassifier

         model = KNeighborsClassifier(n_neighbors=3)

         model.fit(x_train,y_train)
```

```
Out[52]: KNeighborsClassifier(n_neighbors=3)
```

5. Predict possibilities of diabetes if a person has **0** Pregnancies, **10** Glucose concentration, **150** Blood pressure rate, **0** Skin Thickness, **200** insulin intake, **10.9** BMI, **0.009** Diabetes pedigree and having an age of **32**.

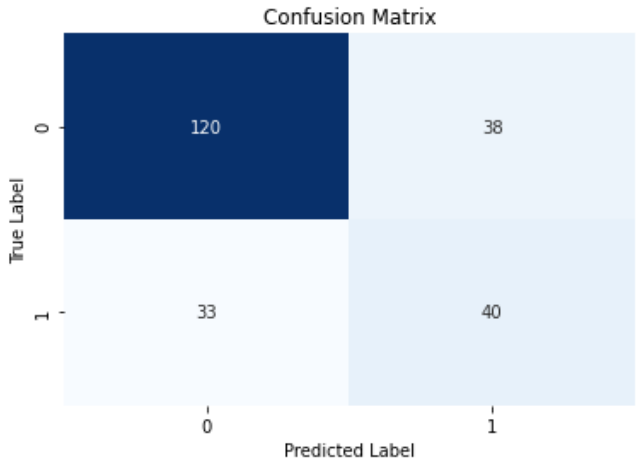
```
In [53]: predicted= model.predict([[0,10,150,0,200,10.9,0.009,32]])
         print(predicted)
```

```
[0]
```

6. Evaluate performance using accuracy score and confusion matrix.

```
In [43]: from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score, confusion_matrix

         y_pred = model.predict(x_test)

         accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy Score:", accuracy)

         conf_matrix = confusion_matrix(y_test, y_pred)

         plt.figure(figsize=(6,4))
         sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
         plt.xlabel('Predicted Label')
         plt.ylabel('True Label')
         plt.title('Confusion Matrix')
         plt.show()
```

```
Accuracy Score: 0.6926406926406926
```



https://pern-my.sharepoint.com/:x:/g/personal/rabiaamjad_bukc_bahria_edu_pk/EShS7y4SxrdIneBEHnMkF_oBGdi7EOd6qFiZpNosGTMxHg?e=cbaJzs

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Task 02

Implement KNN for above task without using Sklearn Library.

1. Data analysis.

```
In [46]: import pandas as pd
         data=pd.read_csv("diabetes.csv")

         data.head()
```

Out[46]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [47]: data.shape
```

```
Out[47]: (768, 9)
```

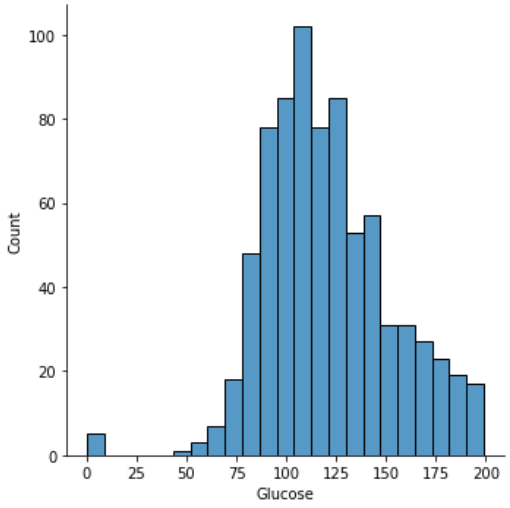
```
In [48]: data.info
```

```
Out[48]: <bound method DataFrame.info of      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
         0              6      148             72             35        0  33.6
         1              1       85             66             29        0  26.6
         2              8      183             64              0        0  23.3
         3              1       89             66             23       94  28.1
         4              0      137             40             35      168  43.1
         ..           ...      ...            ...            ...      ...   ...
         763           10      101             76             48      180  32.9
         764            2      122             70             27        0  36.8
         765            5      121             72             23      112  26.2
         766            1      126             60              0        0  30.1
         767            1       93             70             31        0  30.4

              DiabetesPedigreeFunction  Age  Outcome
         0                       0.627   50        1
         1                       0.351   31        0
         2                       0.672   32        1
         3                       0.167   21        0
         4                       2.288   33        1
         ..                        ...  ...      ...
         763                     0.171   63        0
         764                     0.340   27        0
         765                     0.245   30        0
         766                     0.349   47        1
         767                     0.315   23        0

         [768 rows x 9 columns]>
```

2. Perform data visualization.

In [49]:
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.displot(data['Glucose'])
```

Out[49]:
```
<seaborn.axisgrid.FacetGrid at 0x1dc34acc8b0>
```



3. Feature Selection.

In [68]:
```
X = data[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]
Y = data['Outcome']
```

In [69]:
```
split_index = int(0.7 * len(data))
x_train, x_test = X[:split_index], X[split_index:]
y_train, y_test = Y[:split_index], Y[split_index:]
```

4. Model Implementation.

In [72]:
```
import numpy as  np

def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2)**2))

def knn_predict(x_train, y_train, x_test, k):
    y_pred = []

    for test_point in x_test:
        distances = []

        for train_point in x_train:
            dist = euclidean_distance(test_point, train_point)
            distances.append(dist)

        nearest_indices = np.argsort(distances)[:k]

        k_nearest_labels = [y_train[i] for i in nearest_indices]

        pred_label = max(set(k_nearest_labels), key=k_nearest_labels.count)
        y_pred.append(pred_label)

    return y_pred

k = 3
```

5. Predict possibilities of diabetes if a person has **0** Pregnancies, **10** Glucose concentration, **150** Blood pressure rate, **0** Skin Thickness, **200** insulin intake, **10.9** BMI, **0.009** Diabetes pedigree and having an age of **32**.

In [73]:
```
input_data = np.array([[0, 10, 150, 0, 200, 10.9, 0.009, 32]], dtype=np.float64)

x_test = x_test.values if isinstance(x_test, pd.DataFrame) else x_test
predicted_label = knn_predict(x_train.values, y_train, input_data, k)
print("Predicted possibility of diabetes:", predicted_label[0])
```

```
Predicted possibility of diabetes: 1
```

6. Evaluate performance using accuracy score and confusion matrix.

In [74]:
```
y_pred = knn_predict(x_train.values, y_train, x_test, k)

accuracy = np.sum(y_pred == y_test) / len(y_test)
print("Accuracy Score:", accuracy)

conf_matrix = np.zeros((2, 2))
for true_label, pred_label in zip(y_test, y_pred):
    conf_matrix[true_label][pred_label] += 1
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy Score: 0.7186147186147186
Confusion Matrix:
[[118.  34.]
 [ 31.  48.]]
```

Department of COMPUTER SCIENCE
Bahria University
Discovering Knowledge

CSL-487: Introduction to Data Science Lab

Name: Ahmed Farrukh
Enrolment: 02-134212-030
BS(CS)- Semester 6-A

Lab 07

Task 01

As a data scientist at a telecommunications firm prioritizing secure communication, your task is to create an advanced SMS spam detection system. With increasing phishing scams and fraudulent activities via text, your company seeks a solution to accurately filter spam while ensuring genuine messages flow smoothly, vital for maintaining user trust amidst evolving cyber threats. Write a python program that implements Naïve Bayes model and perform following task:

1. Read and Preprocess data

In [50]:
```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
import string

data = pd.read_csv('spam_or_not_spam.csv')
data.head()
```

Out[50]:

	email	label
0	date wed NUMBER aug NUMBER NUMBER NUMBER NUMB...	0.0
1	martin a posted tassos papadopoulos the greek ...	0.0
2	man threatens explosion in moscow thursday aug...	0.0
3	klez the virus that won t die already the most...	0.0
4	in adding cream to spaghetti carbonara which ...	0.0

In [51]:
```
data.drop_duplicates(inplace=True)
```

In [54]:
```
data.dropna(subset=['email'], inplace=True)
data.dropna(subset=['label'], inplace=True)
```

In [55]:
```
def pre_process(text):
    nopunc=[char for char in text if char not in string.punctuation]
    nopunc=''.join(nopunc)
    clean_words=[word for word in nopunc.split() if word.lower() not in stopwords.words('english')]
    return clean_words
```

In [56]:
```
data['email'].head().apply(pre_process)
```

Out[56]:
```
0    [date, wed, NUMBER, aug, NUMBER, NUMBER, NUMBE...
1    [martin, posted, tassos, papadopoulos, greek, ...
2    [man, threatens, explosion, moscow, thursday, ...
3    [klez, virus, die, already, prolific, virus, e...
4    [adding, cream, spaghetti, carbonara, effect, ...
Name: email, dtype: object
```

2. Perform data analysis

```
In [52]: data.shape
Out[52]: (2889, 2)
```

```
In [53]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2889 entries, 0 to 3015
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   email   2888 non-null   object 
 1   label   2873 non-null   float64
dtypes: float64(1), object(1)
memory usage: 67.7+ KB
```

```
In [68]: data.tail()
```

Out[68]:

	email	label
3011	abc s good morning america ranks it the NUMBE...	1.0
3012	hyperlink hyperlink hyperlink let mortgage le...	1.0
3013	thank you for shopping with us gifts for all ...	1.0
3014	the famous ebay marketing e course learn to s...	1.0
3015	hello this is chinese traditional å□ ä»¶ NUM...	1.0

3. Perform feature selection and split the data into training testing with the ratio of 75/25

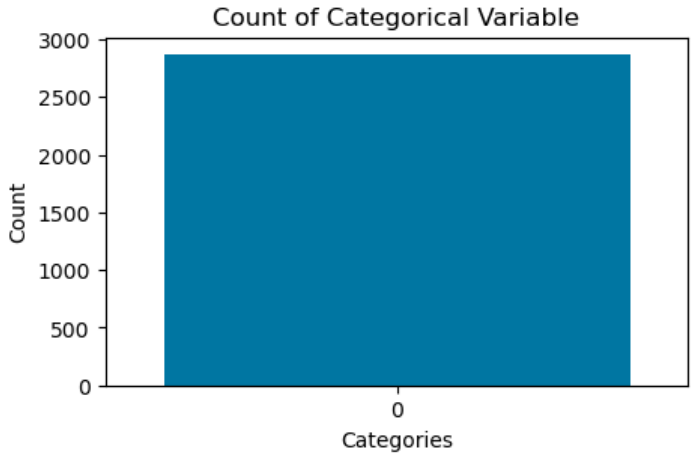
```
In [57]: from sklearn.feature_extraction.text import CountVectorizer

         messages_bow=CountVectorizer(analyzer=pre_process).fit_transform(data['email'])
```

```
In [58]: from sklearn.model_selection import train_test_split

         x_train, x_test, y_train, y_test=train_test_split(messages_bow,data['label'],test_size=0.25, random_state=0)
```

```
In [59]: messages_bow.shape
Out[59]: (2872, 28860)
```

4. Perform data visualization

In [62]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(5, 3))
sns.countplot(data['label'])
plt.title('Count of Categorical Variable')
plt.xlabel('Categories')
plt.ylabel('Count')
plt.show()
```



5. Model implementation using Naïve Bayes

In [63]:
```
from sklearn.naive_bayes import MultinomialNB

classifier=MultinomialNB().fit(x_train,y_train)
```

6. Model implementation using KNN

In [64]:
```
from sklearn.neighbors import KNeighborsClassifier
knc = KNeighborsClassifier(n_neighbors=49).fit(x_train,y_train)
```

7. Evaluate performance using accuracy score and confusion matrix.

In [65]:
```
from sklearn.metrics import *

pred_NB=classifier.predict(x_test)
pred_KNC=knc.predict(x_test)
print("Classification Report of NB: \n\n",classification_report(y_test,pred_NB))
print("Classification Report of KNN: \n\n",classification_report(y_test,pred_KNC))
```

```
Classification Report of NB:

              precision    recall  f1-score   support

         0.0       0.98      0.99      0.99       600
         1.0       0.94      0.92      0.93       118

    accuracy                           0.98       718
   macro avg       0.96      0.95      0.96       718
weighted avg       0.98      0.98      0.98       718

Classification Report of KNN:

              precision    recall  f1-score   support

         0.0       0.84      1.00      0.91       600
         1.0       1.00      0.03      0.07       118

    accuracy                           0.84       718
   macro avg       0.92      0.52      0.49       718
weighted avg       0.87      0.84      0.77       718
```

In [66]:
```
print("Confusion Matrix of NB: \n\n",confusion_matrix(y_test,pred_NB))
print("Confusion Matrix of KNN: \n\n",confusion_matrix(y_test,pred_KNC))
```

```
Confusion Matrix of NB:

 [[593   7]
 [ 10 108]]
Confusion Matrix of KNN:

 [[600   0]
 [114   4]]
```

In [67]:
```
print('Accuracy Score of NB: ',accuracy_score(y_test,pred_NB))
print('Accuracy Score of KNN: ',accuracy_score(y_test,pred_KNC))
```

```
Accuracy Score of NB:  0.9763231197771588
Accuracy Score of KNN:  0.841225626740947
```

8. Write a comparative analysis for both of the models created.

On the basis of the accuracy score, confusion matrix and other analysis it can be determined that Naïve Bayes model is more efficient and performs well compared to KNN model.

https://pern-my.sharepoint.com/:x:/g/personal/rabiaamjad_bukc_bahria_edu_pk/ESflM9HWCe5KmFAP6vAKipoBi7uaMKCLwqbGQ7ITH_sllw?e=PgKNSe

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Lab 08

Task 01

As a market analyst for a retail client, your task is to scrape a leading e-commerce website of your choice for product categories air pods and smart watch. This includes extracting details such as product names, prices, descriptions, and customer reviews. Through web scraping techniques, you aim to compile a comprehensive dataset for competitive analysis and pricing strategies. This initiative will offer valuable insights to inform strategic decisions and enhance your client's market position. Write a python program to perform following task:

1. Fetch the reviews for product categories air pods and smart watch.

```
In [34]: import pandas as pd
         import requests
         from bs4 import BeautifulSoup

In [35]: search_query="airpods+smart+watch"
         base_url="https://www.amazon.com/s?k="

         url=base_url+search_query
         url

Out[35]: 'https://www.amazon.com/s?k=airpods+smart+watch'

In [36]: header={'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64)

In [37]: search_response=requests.get(url,headers=header)

In [38]: search_response.status_code

Out[38]: 200
```

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

In [39]:
```
search_response.text
```

Out[39]:
```
'<!doctype html><html lang="en-us" class="a-no-js" data-19ax5a9jf="dir
r aPageStart = (new Date()).getTime();</script><meta charset="utf-8"/>
e:csm:head-open-part1 -->\n\n<script type=\'text/javascript\'>var ue_t
csm:head-open-part1 -->\n<!-- sp:feature:cs-optimization -->\n<meta ht
\n<link rel="dns-prefetch" href="https://images-na.ssl-images-amazon.c
-amazon.com">\n<link rel="dns-prefetch" href="https://completion.amazo
-- sp:feature:csm:head-open-part2 -->\n<script type=\'text/javascript\
|| 0) + 1;\nif (window.ue_ihb === 1) {\n\nvar ue_csm = window,\n    ue
{},f=Date.now||function(){return+new Date};e.d=function(b){return f()-
[];b[a]=function(){c.push([c.slice.call(arguments),e.d(),d.ue_id])};b[
a[1],a[2])};b[a].isStub=1}};e.exec=function(b,a){return function(){try
(c,{attribution:a||"undefined",logLevel:"WARN"})}}})(ue_csm);\n\n\n
nction h(f,b){if(!(a.ec>a.mxe)&&f){a.ter.push(f);b=b||{};var c=f.logLe
+;c&&c!=k||a.ecf++;b.pageURL=""+(e.location?e.location.href:"");b.logL
erl.push({ex:f,info:b})}}function l(a,b,c,e,g){d.ueLogError({m:a,f:b,l
ribution:g.attribution,logLevel:g.logLevel}:void 0);return!1}var k="FA
0,\npec:0,ts:0,erl:[],ter:[],buffer:[],mxe:50,startTimer:function(){a.
("at");a.pec=a.ec},1E4)}};l.skipTrace=1;h.skipTrace=1;h.isStub=1;d.ueL
\n\nvar ue_id = \'QDERP6Z81C4GW611JV9X\',\n    ue_url = \'/rd/uedata\'
```

In [40]:
```
search_response.cookies
```

Out[40]:
```
<RequestsCookieJar[Cookie(version=0, name='session-id', value='130-6899897-6873363', port=None, por
='.amazon.com', domain_specified=True, domain_initial_dot=True, path='/', path_specified=True, secu
77, discard=False, comment=None, comment_url=None, rest={}, rfc2109=False), Cookie(version=0, name=
='20827872011', port=None, port_specified=False, domain='.amazon.com', domain_specified=True, domai
='/', path_specified=True, secure=True, expires=1745824377, discard=False, comment=None, comment_ur
False), Cookie(version=0, name='i18n-prefs', value='USD', port=None, port_specified=False, domain=
ified=True, domain_initial_dot=True, path='/', path_specified=True, secure=False, expires=17458243
=None, comment_url=None, rest={}, rfc2109=False), Cookie(version=0, name='sp-cdn', value='"L5Z9:PK'
ied=False, domain='.amazon.com', domain_specified=True, domain_initial_dot=True, path='/', path_spe
expires=1745824376, discard=False, comment=None, comment_url=None, rest={'HttpOnly': None}, rfc210
```

In [43]:
```
cookie={}
def getAmazonSearch(search_query):
    url="https://www.amazon.com/s?k="+search_query
    print(url)
    page=requests.get(url,headers=header)
    if page.status_code==200:
        return page
    else:
        return"Error"
```

In [46]:
```
product_names=[]
response=getAmazonSearch('airpods+smart+watch')
soup=BeautifulSoup(response.content)
for i in soup.findAll("span",{'class':'a-size-medium a-color-base a-text-normal'}):
    product_names.append(i.text)
```

```
https://www.amazon.com/s?k=airpods+smart+watch
```

In [47]:
```
product_names
```

Out[47]:
```
['S5 Smart Watch (Answer/Make Calls), 1.43'' AMOLED Smart Watches for Men Women 100+ Sport Modes Fitne
ygen/Sleep/Heart Rate Monitor, IP68 Waterproof Smartwatch Black',
 'Smart Watch with Earbuds, 2 in 1 Bluetooth Watch buds Smart Watch for Android iPhone, Fitness Tracke
art Rate Sleep Monitor, Long Time Standby Sports Men Women Smart Watch',
 'S3 Smart Watch (Answer/Make Call) Bluetooth Fitness Tracker with Heart Rate, Blood Oxygen Monitor, S
rproof 1.83-inch HD Color for Men Women Compatible iPhone & Android',
 'Smart Watches for Men Women, Alexa Built in & Bluetooth Call(Answer/Make), 1.95" Touch Screen Fitnes
ate SpO2 Sleep Monitor Smartwatch for iPhone Android IP68 Waterproof',
 'Smart Watch, Bluetooth 5.3 Answer/Make Call, Alexa Built in, 1.8" Fitness Tracker with Heart Rate Sp
t Watches for Men Women iPhone Android Compatible IP68 Waterproof',
 'Watch Series 9 [GPS 45mm] Smartwatch with Midnight Aluminum Case with Midnight Sport Band S/M. Fitne
Always-On Retina Display, Water Resistant',
 'Smart Watch for Men Women, Smartwatch with Bluetooth Call(Answer/Make Call),  1.85" Touch Screen Fitr
h IP68 Waterproof, 100+Sports Compatible with Android iPhone iOS (Black)',
 'S5 Smartwatch (Answer/Make Calls) Sport Mode Fitness Watch, Black + A1 Wireless Bluetooth in-Ear Hea
 '2 in 1 Stand,Wrieless Charger Compatible with iPhone/Huawei/Samsung/Airpods 2 / Pro, Smart Watch Cha
h, Samsung Galaxy Watch, Huawei Watch, Garmin, LG, Google Smart Watch (White)',
 'S5 Smartwatch (Answer/Make Calls) Sport Mode Fitness Watch, Black + A2 Wireless Bluetooth in-Ear Hea
 'Upgraded Smart Watch Charger for Apple Watch Charger Charging Cable Compatible with iWatch Series SE,
hone 14/13/12/11/Pro/Max/XS/X/Airpods/Pad Series, 6.5ft/2m',
 '2023 New Smart Watch Charger USB C with 20W PD Fast Wall Charger for iWatch/iPhone/Airpods, 4 in 2 iW
or iWatch Series 3/4/5/6/7/8/SE/Ultra Android Phone Charger Cord 6FT/1.8M',
 'Watch Series 9 [GPS + Cellular 45mm] Smartwatch with Midnight Aluminum Case with Midnight Sport Band
Blood Oxygen & ECG Apps, Always-On Retina Display',
```

In [76]:
```
len(product_names)
```

Out[76]: 19

In [78]:
```
data_asin=[]
response=getAmazonSearch('airpods+smart+watch')
soup=BeautifulSoup(response.content)
for i in soup.findAll("div",{'class':"sg-col-20-of-24 s-result-item s-asin sg-col-0-of-12 sg-c
    data_asin.append(i['data-asin'])
```

https://www.amazon.com/s?k=airpods+smart+watch

In [79]:
```
data_asin
```

Out[79]:
```
['B07KY8MQGT',
 'B08LCX2M9J',
 'B07L1GWR66',
 'B0C4BF5BLR',
 'B0C4FXNFFX',
 'B0BPQRFHK7',
 'B0CJFFBDY1',
 'B0CSTY93PP',
 'B0D12GK38C',
 'B09PFXSRC2',
 'B0CHDPG4JH',
 'B0C4YDXT11',
 'B0D12JWP8W',
 'B084D6FSG1',
 'B07MDPC5NH',
 'B092XRJ3Q6']
```

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

In [80]:
```
def SearchAsin(asin):
    url="https://www.amazon.com/airpods+smart+watch/dp/"+asin
    print(url)
    page=requests.get(url,cookies=cookie,headers=header)
    if page.status_code==200:
        return page
    else:
        return "Error"
```

In [91]:
```
link=[]
for i in range(len(data_asin)):
    response=SearchAsin(data_asin[i])
    soup=BeautifulSoup(response.content)
    for i in soup.findAll("a",{'data-hook':"see-all-reviews-link-foot"}):
        link.append(i['href'])

len(link)
```

```
https://www.amazon.com/airpods+smart+watch/dp/B07KY8MQGT
https://www.amazon.com/airpods+smart+watch/dp/B08LCX2M9J
https://www.amazon.com/airpods+smart+watch/dp/B07L1GWR66
https://www.amazon.com/airpods+smart+watch/dp/B0C4BF5BLR
https://www.amazon.com/airpods+smart+watch/dp/B0C4FXNFFX
https://www.amazon.com/airpods+smart+watch/dp/B0BPQRFHK7
https://www.amazon.com/airpods+smart+watch/dp/B0CJFFBDY1
https://www.amazon.com/airpods+smart+watch/dp/B0CSTY93PP
https://www.amazon.com/airpods+smart+watch/dp/B0D12GK38C
https://www.amazon.com/airpods+smart+watch/dp/B09PFXSRC2
https://www.amazon.com/airpods+smart+watch/dp/B0CHDPG4JH
https://www.amazon.com/airpods+smart+watch/dp/B0C4YDXT11
https://www.amazon.com/airpods+smart+watch/dp/B0D12JWP8W
https://www.amazon.com/airpods+smart+watch/dp/B084D6FSG1
https://www.amazon.com/airpods+smart+watch/dp/B07MDPC5NH
https://www.amazon.com/airpods+smart+watch/dp/B092XRJ3Q6
```

Out[91]: 23

In [94]:
```
def SearchReviews(link):
    url="https://www.amazon.com"+link
    print(url)
    page=requests.get(url,cookies=cookie,headers=header)
    if page.status_code==200:
        return page
    else:
        return "Error"
```

In [95]:
```
reviews=[]
for j in range(len(link)):
    for k in range(100):
        response=SearchReviews(link[j]+'&pagenumber=' +str(k))
        soup=BeautifulSoup(response.content)
    for i in soup.findAll("span",{'data-hook':"review-body"}):
        reviews.append(i.text)
```

```
https://www.amazon.com/TOZO-S5-Watches-Waterproof-Smartwatch/product-reviews/B07KY8MQGT/
8&reviewerType=all_reviews&pagenumber=0
https://www.amazon.com/TOZO-S5-Watches-Waterproof-Smartwatch/product-reviews/B07KY8MQGT/
8&reviewerType=all_reviews&pagenumber=1
https://www.amazon.com/TOZO-S5-Watches-Waterproof-Smartwatch/product-reviews/B07KY8MQGT/
8&reviewerType=all_reviews&pagenumber=2
https://www.amazon.com/TOZO-S5-Watches-Waterproof-Smartwatch/product-reviews/B07KY8MQGT/
8&reviewerType=all_reviews&pagenumber=3
https://www.amazon.com/TOZO-S5-Watches-Waterproof-Smartwatch/product-reviews/B07KY8MQGT/
8&reviewerType=all_reviews&pagenumber=4
https://www.amazon.com/TOZO-S5-Watches-Waterproof-Smartwatch/product-reviews/B07KY8MQGT/
8&reviewerType=all_reviews&pagenumber=5
https://www.amazon.com/TOZO-S5-Watches-Waterproof-Smartwatch/product-reviews/B07KY8MQGT/
8&reviewerType=all_reviews&pagenumber=6
https://www.amazon.com/TOZO-S5-Watches-Waterproof-Smartwatch/product-reviews/B07KY8MQGT/
8&reviewerType=all_reviews&pagenumber=7
https://www.amazon.com/TOZO-S5-Watches-Waterproof-Smartwatch/product-reviews/B07KY8MQGT/
8&reviewerType=all_reviews&pagenumber=8
https://www.amazon.com/TOZO-S5-Watches-Waterproof-Smartwatch/product-reviews/B07KY8MQGT/
```

In [97]:
```
len(reviews)
```

Out[97]: 165

2. Store the fetch data in CSV file.

In [98]:
```
rev={'reviews':reviews}
review_data=pd.DataFrame.from_dict(rev)
pd.set_option('max_colwidth', 800)
```

In [99]:
```
review_data.head(5)
```

Out[99]:

	reviews
0	\nA Budget-Friendly Gem! I recently had the opportunity to test out the Tozo S5 Smart Watch. I am writing this review after the pleasantly surprised me, especially considering its affordable price point. I already owned the S4 version and was using budget-friendly wearable:Design and Build Quality:The Tozo S5 boasts a sleek and modern design that wouldn't look out of p make it comfortable to wear throughout the day, and the quality of materials used feels sturdy enough to withstand daily display is vibrant and responsive, making navigation a breeze.Feat
1	\nTozo Watch S5 is a sleek and stylish smartwatch that combines sophisticated design with advanced functionality, smartwatch market. Here's a detailed review of its features, performance, and overall value.Design: The Tozo Watch S5 bo profile and a high-quality stainless steel case that exudes durability and sophistication. Its 1.4-inch AMOLED touchscre providing an immersive user experience. The customizable watch faces add a personal touch, allowing users to tailor the devi with a wide range of features, the Tozo Watch S5 serves as a versatil
2	\nThe TOZO S5 is a full-featured smart watch that offers a lot of value for the money. It syncs well with my iPhone to answ health/workout features, but I like the way it tracks my daily steps and measures pulse rate and blood oxygen level. Its build qu face options attractive and functional, and it offers plenty of onboard apps. I also own an Apple Watch, which needs to be rech use from the first charge on my TOZO S5. For that reason alone, this is
3	\nSo I'm writing about this watch after using two weeks. In a nutshell, the quality and design the TOZO S5 smartwatch provid recommend it but let me point out a few things that I noticed in the last two weeks of using it.The design - the round face resistant metal, and the matched soft belt just blew me away. It's a beautiful watch that is super-comfortable to wear for a long going for formal/semi-formal or casual. I'm a 70" tall 35y Male and whatever the occasion and outfit I choose, the watch goes v bright and easy to read even under full sunlight
4	\nThis TOZO smart watch model S5 is an upgrade (in many ways) from my TOZO S4. Compering the two watches, the S5 is watch is something that you would wear to a big night out. Both have a great design but the S5 is larger with a heavier feel. T AMOLED touchscreen display, big colors and super sharp clarity. You can customize watch faces to make the watch to your p faces. It has great design, lots of features including oxygen, heart rate and tracking fitness. I've had no performance issue stands out as a strong contender in the vast world of smart watches.

In [100]:
```
review_data.shape
```
Out[100]: (165, 1)

In [101]:
```
review_data.to_csv('Scraping_reviews.csv')
```

In [102]:
```
review_data.to_excel('E:\\Ahmed\\University\\Semester 6\\Courses\\DS Lab\\Labs\\Lab 8\\amazon_
```

3. Pre-process the fetched data.

In [104]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))
import nltk
nltk.download('wordnet')
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer, PorterStemmer
from nltk.corpus import stopwords
import re
lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()
```
```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Ahmed\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Ahmed\AppData\Roaming\nltk_data...
```

In [109]:
```
import pandas as pd
import numpy as np
from textblob import TextBlob
import re
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
from wordcloud import WordCloud
```

In [115]:
```
def preprocess(sentence):
    sentence=str(sentence)
    sentence=sentence.lower()
    sentence=sentence.replace('{html}',"")
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, '', sentence)
    rem_url=re.sub(r'http\S+','',cleantext)
    rem_num=re.sub('[0-9]+','',rem_url)
    tokenizer=RegexpTokenizer(r'\w+')
    tokens=tokenizer.tokenize(rem_num)
    filtered_words=[w for w in tokens if len(w) > 2 if not w in stopwords.words('english')]
    stem_words=[stemmer.stem(w) for w in filtered_words]
    lemma_words=[lemmatizer.lemmatize(w) for w in stem_words]
    return " ".join(filtered_words)
```

In [116]:
```
review_data['cleanReviews']=review_data['reviews'].map(lambda s:preprocess(s))
```

In [117]:
```
review_data.to_csv(r'cleaned_amazon.csv', index = False)
```

4. Apply snetiment analysis over the data. Annotate each data with repsective sentimet.

In [118]:
```
def getSubjectivity(text):
    return TextBlob(text).sentiment.subjectivity
def getPolarity(text):
    return TextBlob(text).sentiment.polarity
```

In [119]:
```
review_data['Subjectivity']=review_data['cleanReviews'].apply(getSubjectivity)
```

In [120]:
```
review_data['Polarity']=review_data['cleanReviews'].apply(getPolarity)
```

In [121]:
```
def GetAnalysis(score):
    if score < 0:
        return 'Negative'
    elif score == 0:
        return 'Neutral'
    else:
        return 'Positive'
```

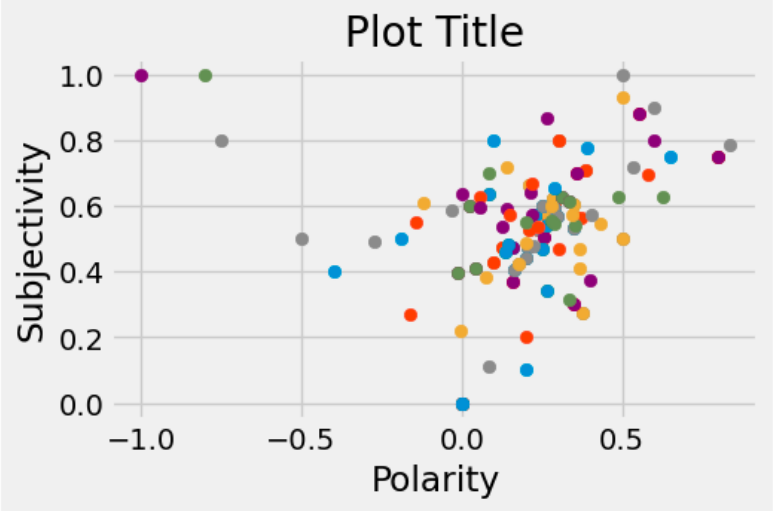
In [122]:
```
review_data['Sen_tag']=review_data['Polarity'].apply(GetAnalysis)
review_data.head()
```

Out[122]:

	reviews	cleanReviews	Subjectivity	Polarity
0	\nA Budget-Friendly Gem! I recently had the opportunity to test out the Tozo S5 Smart Watch. I am writing this review after the use of one month. Its features and performance pleasantly surprised me, especially considering its affordable price point. I already owned the S4 version and was using Here's a breakdown of my experience with this budget-friendly wearable:Design and Build Quality:The Tozo S5 boasts a sleek and modern design that wouldn't look out of place on any wrist. Its lightweight and slim profile make it comfortable to wear throughout the day, and the quality of materials used feels sturdy enough to withstand daily wear and tear. The 1.54-inch color touchscreen display is vibrant and responsive, making navigation a breeze.Features:Despite its budget-friendly price, the Tozo...	budget friendly gemi recently opportunity test tozo smart watch writing review use one month features performance pleasantly surprised especially considering affordable price point already owned version using breakdown experience budget friendly wearable design build quality tozo boasts sleek modern design look place wrist lightweight slim profile make comfortable wear throughout day quality materials used feels sturdy enough withstand daily wear tear inch color touchscreen display vibrant responsive making navigation breeze features despite budget friendly price tozo packed features rival expensive smartwatches market basic fitness tracking capabilities like step counting calorie monitoring advanced functions heart rate monitoring sleep tracking smartwatch covers essentials maintainin...	0.474291	0.202409

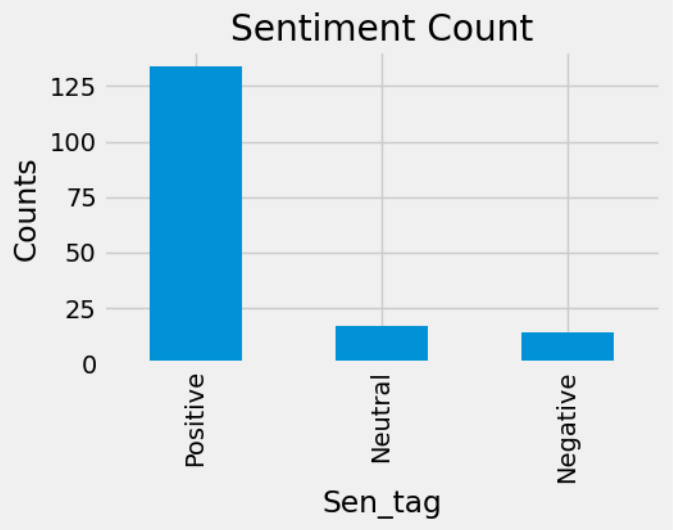
5. Display the sentiment count in bar graph against respective sentiment of each category.

In [138]:
```
plt.figure(figsize=(5,3))
for i in range(0, review_data.shape[0]):
    plt.scatter(review_data['Polarity'][i],review_data['Subjectivity'][i])
plt.title('Plot Title')
plt.xlabel('Polarity')
plt.ylabel('Subjectivity')
plt.show()
```



Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

In [137]:
```
review_data['Sen_tag'].value_counts()
plt.figure(figsize=(5,3))
plt.title('Sentiment Count')
plt.xlabel('Sentiments')
plt.ylabel('Counts')
review_data['Sen_tag'].value_counts().plot(kind='bar')
plt.show()
```



Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Lab 09

Task 01

In 2016, Kickstarter, PBC, an American public benefit corporation based in Brooklyn, New York, launched a groundbreaking crowd funding campaign. Their mission: to bring creative projects to life by democratizing education through a global online platform. Backers worldwide rallied behind the project, fueling a movement that transcended borders and empowered learners everywhere. Write a python program and implement following task:

1. EDA & Pre-processing
    - Dealing with missing values
    - Removing whitespace of col names

In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [7]:
```
data=pd.read_csv("ks-projects-201612.csv", encoding='latin_1',low_memory=False)
data.head()
```

Out[7]:

	ID	name	category	main_category	currency	deadline	goal	launched	pledged	state	backers	country	usd pledged	Unnamed: 13	Unr
0	1000002330	The Songs of Adelaide & Abullah	Poetry	Publishing	GBP	2015-10-09 11:36:00	1000	2015-08-11 12:12:28	0	failed	0	GB	0	NaN	
1	1000004038	Where is Hank?	Narrative Film	Film & Video	USD	2013-02-26 00:20:50	45000	2013-01-12 00:20:50	220	failed	3	US	220	NaN	
2	1000007540	ToshiCapital Rekordz Needs Help to Complete Album	Music	Music	USD	2012-04-16 04:24:11	5000	2012-03-17 03:24:11	1	failed	1	US	1	NaN	
3	1000011046	Community Film Project: The Art of Neighborhoo...	Film & Video	Film & Video	USD	2015-08-29 01:00:00	19500	2015-07-04 08:35:03	1283	canceled	14	US	1283	NaN	
4	1000014025	Monarch Espresso Bar	Restaurants	Food	USD	2016-04-01 13:38:27	50000	2016-02-26 13:38:27	52375	successful	224	US	52375	NaN	

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 323750 entries, 0 to 323749
Data columns (total 17 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   ID              323750 non-null  int64
 1   name            323746 non-null  object
 2   category        323745 non-null  object
 3   main_category   323750 non-null  object
 4   currency        323750 non-null  object
 5   deadline        323750 non-null  object
 6   goal            323750 non-null  object
 7   launched        323750 non-null  object
 8   pledged         323750 non-null  object
 9   state           323750 non-null  object
 10  backers         323750 non-null  object
 11  country         323750 non-null  object
 12  usd pledged     319960 non-null  object
 13  Unnamed: 13     625 non-null     object
 14  Unnamed: 14     12 non-null      object
 15  Unnamed: 15     4 non-null       object
 16  Unnamed: 16     1 non-null       float64
dtypes: float64(1), int64(1), object(15)
memory usage: 42.0+ MB
```

In [9]:
```
data=data.drop(['Unnamed: 13','Unnamed: 14','Unnamed: 15','Unnamed: 16'], axis=1)
```

In [10]:
```
data.head()
```

Out[10]:

	ID	name	category	main_category	currency	deadline	goal	launched	pledged	state	backers	country	usd pledged
0	1000002330	The Songs of Adelaide & Abullah	Poetry	Publishing	GBP	2015-10-09 11:36:00	1000	2015-08-11 12:12:28	0	failed	0	GB	0
1	1000004038	Where is Hank?	Narrative Film	Film & Video	USD	2013-02-26 00:20:50	45000	2013-01-12 00:20:50	220	failed	3	US	220
2	1000007540	ToshiCapital Rekordz Needs Help to Complete Album	Music	Music	USD	2012-04-16 04:24:11	5000	2012-03-17 03:24:11	1	failed	1	US	1
3	1000011046	Community Film Project: The Art of Neighborhoo...	Film & Video	Film & Video	USD	2015-08-29 01:00:00	19500	2015-07-04 08:35:03	1283	canceled	14	US	1283
4	1000014025	Monarch Espresso Bar	Restaurants	Food	USD	2016-04-01 13:38:27	50000	2016-02-26 13:38:27	52375	successful	224	US	52375

In [11]:
```
data[['goal ','pledged ','backers ','usd pledged ']] = data[['goal ','pledged ','backers ','usd pledged ']].apply(pd.to_numeric,
```

In [12]:
```
data.dropna(inplace=True)
```

In [13]:
```
data.isnull().sum()
```

Out[13]:
```
ID                0
-
```

In [15]:
```
newColnames=[]
for i in range (len(data.columns.values)):
    newColnames.append(data.columns.values[i][:len(data.columns.values[i])-1])

data.columns=newColnames
data.head()
print(data.columns)
```

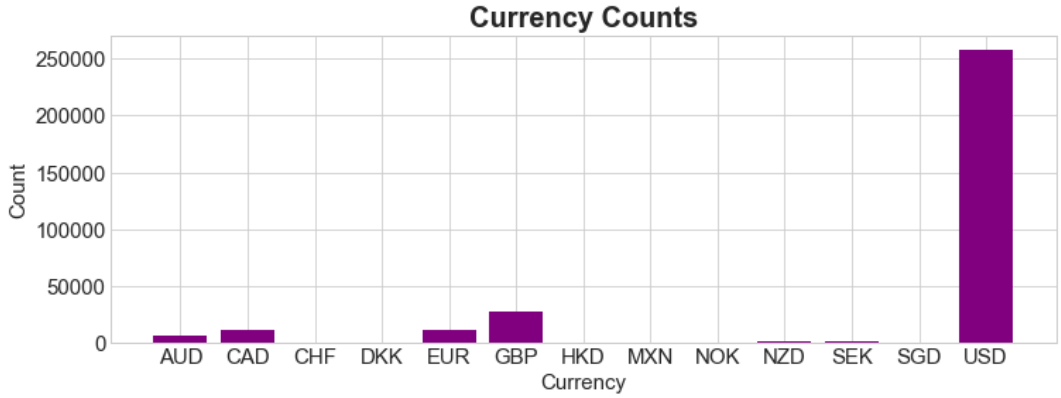
```
Index(['ID', 'name', 'category', 'main_category', 'currency', 'deadline',
       'goal', 'launched', 'pledged', 'state', 'backers', 'country',
       'usd pledged'],
      dtype='object')
usd pledged       0
dtype: int64
```

In [14]:
```
print(data.columns.values)
```

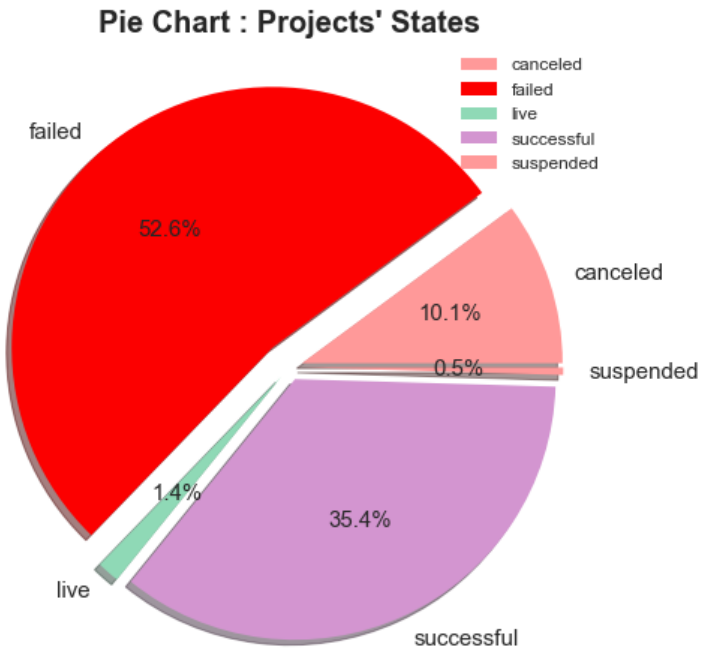
```
['ID ' 'name ' 'category ' 'main_category ' 'currency ' 'deadline '
 'goal ' 'launched ' 'pledged ' 'state ' 'backers ' 'country '
 'usd pledged ']
```

In [16]:
```
ks_cnt_cur=data.groupby('currency')['currency'].count()

plt.style.use('seaborn-whitegrid')
plt.figure(figsize=(12,4))

plt.bar(ks_cnt_cur.index.values,ks_cnt_cur.values,color='purple')

plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.title('Currency Counts', fontsize=20, fontweight='bold')
plt.xlabel('Currency', fontsize=15)
plt.ylabel('Count', fontsize=15)
```

Out[16]:
```
Text(0, 0.5, 'Count')
```



In [18]:
```
ks_cnt_st = data.groupby('state')['state'].count()


plt.style.use('seaborn-whitegrid')
plt.figure(figsize=(8,8))

colors = ['#ff9999', '#fc0000', '#8fd9b6', '#d395d0']
explode = [0.05,0.1,0.05,0.05,0.05]
plt.pie(ks_cnt_st.values, labels=ks_cnt_st.index.values,
        autopct='%.1f%%',
        explode=explode,
        shadow=True,
        colors=colors,
        textprops={'fontsize':15})
plt.title('Pie Chart : Projects\' States', fontsize=20, fontweight='bold')
plt.legend(loc='upper right', fontsize=12)

plt.show()
```



2. Generating dummies
   - Drop all the projects unfinished
   - Generate dummies
   - Convert the Dependent variable's values to [0,1]

In [19]:
```
df_ks_p = data
df_ks_p.drop(df_ks_p.loc[df_ks_p['state'] == 'live'].index, inplace=True)
df_ks_p.drop(df_ks_p.loc[df_ks_p['state'] == 'suspended'].index, inplace=True)
df_ks_p.drop(df_ks_p.loc[df_ks_p['state'] == 'canceled'].index, inplace=True)
print(df_ks_p['state'].value_counts())
```

```
failed        168113
successful    112976
Name: state, dtype: int64
```

In [20]:
```
df_ks_pd =pd.get_dummies(data=df_ks_p, columns=['main_category', 'country', 'currency' ], drop_first=True)
df_ks_pd.head()
```

Out[20]:

	ID	name	category	deadline	goal	launched	pledged	state	backers	usd pledged	...	currency_DKK	currency_EUR	currency_GBP
0	1000002330	The Songs of Adelaide & Abullah	Poetry	2015-10-09 11:36:00	1000.0	2015-08-11 12:12:28	0.0	failed	0.0	0.0	...	0	0	1
1	1000004038	Where is Hank?	Narrative Film	2013-02-26 00:20:50	45000.0	2013-01-12 00:20:50	220.0	failed	3.0	220.0	...	0	0	0
2	1000007540	ToshiCapital Rekordz Needs Help to Complete Album	Music	2012-04-16 04:24:11	5000.0	2012-03-17 03:24:11	1.0	failed	1.0	1.0	...	0	0	0
4	1000014025	Monarch Espresso Bar	Restaurants	2016-04-01 13:38:27	50000.0	2016-02-26 13:38:27	52375.0	successful	224.0	52375.0	...	0	0	0
5	1000023410	Support Solar Roasted Coffee & Green Energy! ...	Food	2014-12-21 18:30:44	1000.0	2014-12-01 18:30:44	1205.0	successful	16.0	1205.0	...	0	0	0

5 rows × 56 columns

In [21]:
```
df_ks_pd.loc[df_ks_pd['state'] == 'failed','state']= 0
df_ks_pd.loc[df_ks_pd['state'] == 'successful','state']= 1

df_ks_pd['state'] = df_ks_pd['state'].astype('int64')
print(df_ks_pd['state'].value_counts())
```

```
0    168113
1    112976
Name: state, dtype: int64
```

In [22]:
```
data=df_ks_pd.drop(['ID', 'name','goal', 'category','deadline',
        'launched'], axis=1)
print(data.columns.values)
```

```
['pledged' 'state' 'backers' 'usd pledged' 'main_category_Comics'
 'main_category_Crafts' 'main_category_Dance' 'main_category_Design'
 'main_category_Fashion' 'main_category_Film & Video' 'main_category_Food'
 'main_category_Games' 'main_category_Journalism' 'main_category_Music'
 'main_category_Photography' 'main_category_Publishing'
 'main_category_Technology' 'main_category_Theater' 'country_AU'
 'country_BE' 'country_CA' 'country_CH' 'country_DE' 'country_DK'
 'country_ES' 'country_FR' 'country_GB' 'country_HK' 'country_IE'
 'country_IT' 'country_LU' 'country_MX' 'country_NL' 'country_NO'
 'country_NZ' 'country_SE' 'country_SG' 'country_US' 'currency_CAD'
 'currency_CHF' 'currency_DKK' 'currency_EUR' 'currency_GBP'
 'currency_HKD' 'currency_MXN' 'currency_NOK' 'currency_NZD'
 'currency_SEK' 'currency_SGD' 'currency_USD']
```

3. Logistic Regression: Model Building
   - Splitting into train and test sets
   - Building Logistic Regression model

In [23]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(data.drop('state',axis=1),data['state'], test_size=0.25,random_state=101)
```

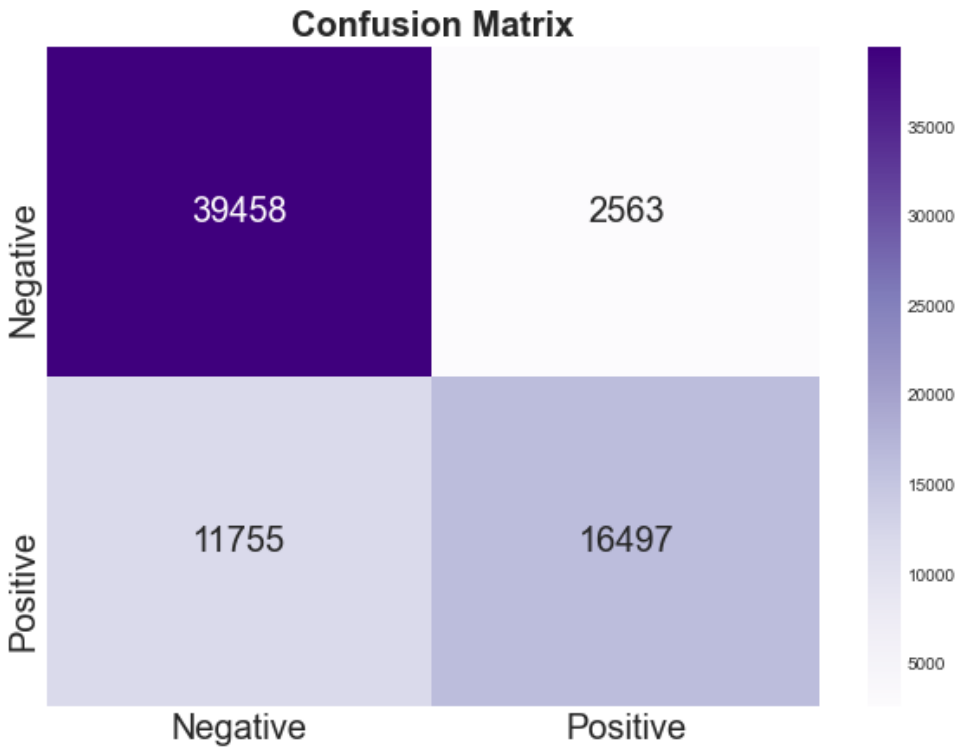
In [24]:
```
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression (solver='liblinear')
logmodel.fit(X_train, y_train)
predictions = logmodel.predict(X_test)
probs = logmodel.predict_proba(X_test)
```

4. Model Evaluation
   - Confusion Matrix
   - Accuracy & precision

In [25]:
```
from sklearn.metrics import confusion_matrix
cf_mat = confusion_matrix(y_test, predictions)
print(cf_mat)
```

```
[[39458  2563]
 [11755 16497]]
```

In [26]:
```
plt.figure(figsize=(10,7))
sns.heatmap(cf_mat, annot=True,
xticklabels=['Negative', 'Positive'],
yticklabels=['Negative', 'Positive'],
cmap='Purples',
fmt='g',
annot_kws={'size' :20})
plt.title('Confusion Matrix', fontsize=20, fontweight='bold')
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```



In [27]:
```
from sklearn.metrics import classification_report
print(classification_report (y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.77      0.94      0.85     42021
           1       0.87      0.58      0.70     28252

    accuracy                           0.80     70273
   macro avg       0.82      0.76      0.77     70273
weighted avg       0.81      0.80      0.79     70273
```

https://www.kaggle.com/datasets/kemical/kickstarter-projects?select=ks-projects-201612.csv

Lab 10

Task 01

Imagine overseeing a pizza restaurant meticulously tracking nutritional data such as calories, fat content, moisture levels etc. in various pizza brands. Despite the wealth of information, the dataset's complexity poses challenges for analysis. Introducing PCA and SVD (Singular Value Decomposition) provides a solution by condensing the dataset's dimensions while retaining its vital aspects. Leveraging PCA and SVD unveils hidden correlations in pizza composition, identifies crucial factors driving nutritional value, and streamlines decision-making processes like ingredient sourcing and recipe development. Download the dataset from given link Pizza.csv and perform following task:

1. Perform data analysis

In [20]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA, TruncatedSVD
from sklearn.preprocessing import StandardScaler
```

In [21]:
```
data = pd.read_csv("Pizza.csv")

data.head()
```

Out[21]:

	brand	id	mois	prot	fat	ash	sodium	carb	cal
0	A	14069	27.82	21.43	44.87	5.11	1.77	0.77	4.93
1	A	14053	28.49	21.26	43.89	5.34	1.79	1.02	4.84
2	A	14025	28.35	19.99	45.78	5.08	1.63	0.80	4.95
3	A	14016	30.55	20.15	43.13	4.79	1.61	1.38	4.74
4	A	14005	30.49	21.28	41.65	4.82	1.64	1.76	4.67

In [22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   brand   300 non-null    object
 1   id      300 non-null    int64
 2   mois    300 non-null    float64
 3   prot    300 non-null    float64
 4   fat     300 non-null    float64
 5   ash     300 non-null    float64
 6   sodium  300 non-null    float64
 7   carb    300 non-null    float64
 8   cal     300 non-null    float64
dtypes: float64(7), int64(1), object(1)
memory usage: 21.2+ KB
```

In [23]:
```
data.shape
```

Out[23]: (300, 9)

2. Split dataset into 70/30 ratio.

In [24]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
data['brand']=label_encoder.fit_transform(data['brand'])
```

In [25]:
```
data.head()
```

Out[25]:

	brand	id	mois	prot	fat	ash	sodium	carb	cal
0	0	14069	27.82	21.43	44.87	5.11	1.77	0.77	4.93
1	0	14053	28.49	21.26	43.89	5.34	1.79	1.02	4.84
2	0	14025	28.35	19.99	45.78	5.08	1.63	0.80	4.95
3	0	14016	30.55	20.15	43.13	4.79	1.61	1.38	4.74
4	0	14005	30.49	21.28	41.65	4.82	1.64	1.76	4.67

In [26]:
```
X=data.drop('brand', axis=1)
y=data['brand']
```

```
In [27]: x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=
```

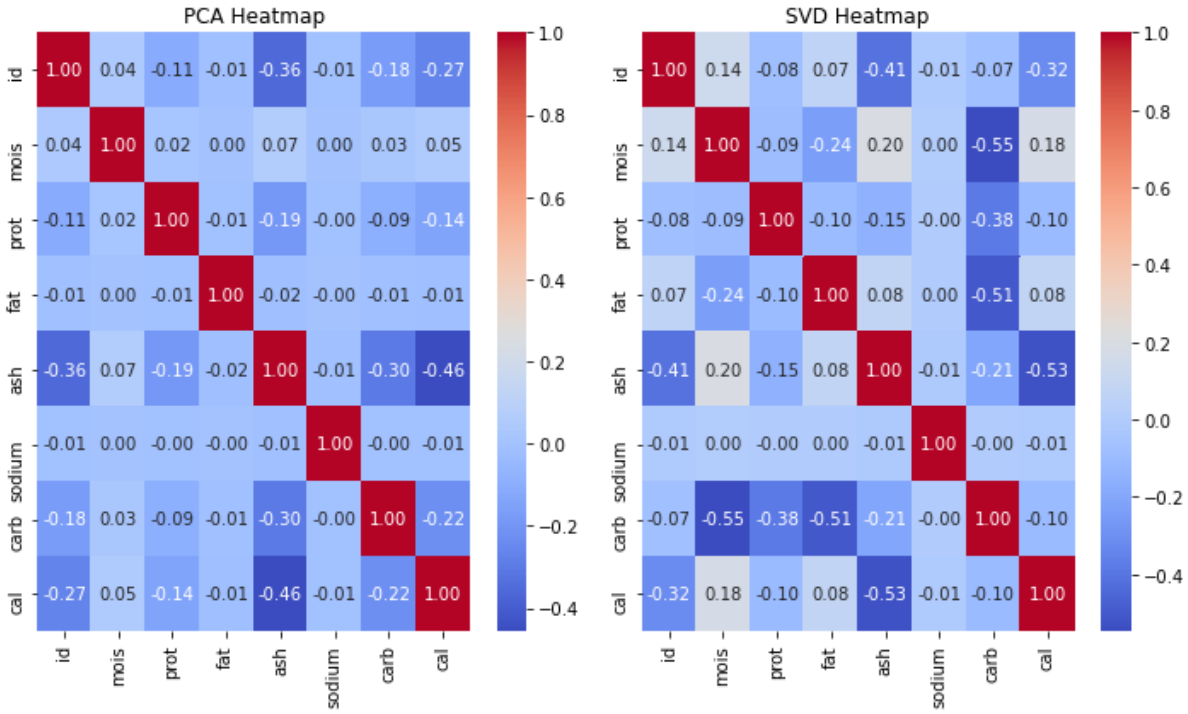
```
In [28]: scaler = StandardScaler()
         X_train_scaled = scaler.fit_transform(x_train)
         X_test_scaled = scaler.transform(x_test)
```

3. Implement KNN for 9 neighbors.

```
In [31]: from sklearn.metrics import accuracy_score

         knn = KNeighborsClassifier(n_neighbors=9)
         knn.fit(X_train_scaled, y_train)

         y_pred = knn.predict(X_test_scaled)

         accuracy_bfr = accuracy_score(y_test, y_pred)
```

4. Apply PCA & SVD and display the result by plotting Head map (data.corr).

```
In [35]: from sklearn.decomposition import PCA, TruncatedSVD
         import seaborn as sns

         pca = PCA()
         pca_result = pca.fit_transform(X_train_scaled)

         svd = TruncatedSVD(n_components=min(X.shape[0], X.shape[1])-1)
         svd_result = svd.fit_transform(X_train_scaled)

         plt.figure(figsize=(10, 6))

         plt.subplot(1, 2, 1)
         sns.heatmap(pd.DataFrame(pca.components_, columns=X.columns).corr(), annot=True, cmap='coolwarm', fmt=".2f")
         plt.title('PCA Heatmap')

         plt.subplot(1, 2, 2)
         sns.heatmap(pd.DataFrame(svd.components_, columns=X.columns).corr(), annot=True, cmap='coolwarm', fmt=".2f")
         plt.title('SVD Heatmap')

         plt.tight_layout()
         plt.show()
```



5. Evaluate the results by Mean Acc Score before and after PCA and SVD implementation.

In [42]:
```
X_test_pca = pca.transform(X_test_scaled)
X_test_svd = svd.transform(X_test_scaled)

knn_pca = KNeighborsClassifier(n_neighbors=9)
knn_svd = KNeighborsClassifier(n_neighbors=9)

knn_pca.fit(pca_result, y_train)
knn_svd.fit(svd_result, y_train)

y_pred_pca = knn_pca.predict(X_test_pca)
y_pred_svd = knn_svd.predict(X_test_svd)

accuracy_pca = accuracy_score(y_test, y_pred_pca)
accuracy_svd = accuracy_score(y_test, y_pred_svd)

print("Accuracy after PCA:", accuracy_pca)
print("Accuracy after SVD:", accuracy_svd)
```

```
Accuracy after PCA: 0.7666666666666667
Accuracy after SVD: 0.7666666666666667
```

Department of COMPUTER SCIENCE
Bahria University
Discovering Knowledge

CSL-487: Introduction to Data Science Lab
Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Lab 11

Task 01

You're a movie enthusiast who wants to create a personalized movie recommendation system. You've collected data on your favorite genres and ratings for a handful of films with the help of given dataset: filmtv_movies.csv.

Write a python program that implements cosine similarity to create a recommendation system and perform following task:

1. Data Analysis and Data visualization

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
```

In [2]:
```
df = pd.read_csv('filmtv_movies.csv', nrows=1000)
df.head()
```

Out[2]:

	filmtv_id	title	year	genre	duration	country	directors	actors	avg_vote	critics_vote	public_vote	total_votes	description	notes	humc
0	2	Bugs Bunny's Third Movie: 1001 Rabbit Tales	1982	Animation	76	United States	David Detiege, Art Davis, Bill Perez	NaN	7.7	8.00	7	22	With two protruding front teeth, a slightly sl...	These are many small independent stories, whic...	
1	3	18 anni tra una settimana	1991	Drama	98	Italy	Luigi Perelli	Kim Rossi Stuart, Simona Cavallari, Ennio Fant...	6.5	6.00	7	4	Samantha, not yet eighteen, leaves the comfort...	Luigi Perelli, the director of the "Piovra", o...	
2	17	Ride a Wild Pony	1976	Romantic	91	United States	Don Chaffey	Michael Craig, John Meillon, Eva Griffith, Gra...	5.7	6.00	5	10	In the Australia of the pioneers, a boy and a ...	"Ecological" story with a happy ending, not wi...	

In [4]:
```
df.shape
```

Out[4]:
```
(1000, 19)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 19 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   filmtv_id     1000 non-null   int64
 1   title         1000 non-null   object
 2   year          1000 non-null   int64
 3   genre         1000 non-null   object
 4   duration      1000 non-null   int64
 5   country       1000 non-null   object
 6   directors     1000 non-null   object
 7   actors        991 non-null    object
 8   avg_vote      1000 non-null   float64
 9   critics_vote  1000 non-null   float64
 10  public_vote   1000 non-null   int64
 11  total_votes   1000 non-null   int64
 12  description   1000 non-null   object
 13  notes         653 non-null    object
 14  humor         1000 non-null   int64
 15  rhythm        1000 non-null   int64
 16  effort        1000 non-null   int64
 17  tension       1000 non-null   int64
 18  erotism       1000 non-null   int64
dtypes: float64(2), int64(10), object(7)
```

2. Select important features

```
In [6]: columns=['title', 'genre', 'directors', 'actors']
```

```
In [7]: df[columns].head()
```

Out[7]:

	title	genre	directors	actors
0	Bugs Bunny's Third Movie: 1001 Rabbit Tales	Animation	David Detiege, Art Davis, Bill Perez	NaN
1	18 anni tra una settimana	Drama	Luigi Perelli	Kim Rossi Stuart, Simona Cavallari, Ennio Fant...
2	Ride a Wild Pony	Romantic	Don Chaffey	Michael Craig, John Meillon, Eva Griffith, Gra...
3	Diner	Comedy	Barry Levinson	Mickey Rourke, Steve Guttenberg, Ellen Barkin,...
4	A che servono questi quattrini?	Comedy	Esodo Pratelli	Eduardo De Filippo, Peppino De Filippo, Clelia...

```
In [8]: df[columns].isnull().values .any()
```

```
Out[8]: True
```

```
In [11]: df = df.dropna(subset=columns)
```

```
In [12]: df[columns].isnull().values .any()
```

```
Out[12]: False
```

3. Implement cosine similarity

```
In [20]: def get_imp_features(row):
             return f"{row['title']} {row['genre']} {row['directors']} {row['actors']}"
```

```
In [21]: df['imp_features'] = df.apply(get_imp_features, axis=1)
         df.head(2)
```

Out[21]:

	filmtv_id	title	year	genre	duration	country	directors	actors	avg_vote	critics_vote	public_vote	total_votes	description	notes	humor
1	3	18 anni tra una settimana	1991	Drama	98	Italy	Luigi Perelli	Kim Rossi Stuart, Simona Cavallari, Ennio Fant...	6.5	6.0	7	4	Samantha, not yet eighteen, leaves the comfort...	Luigi Perelli, the director of the "Piovra", o...	0
2	17	Ride a Wild Pony	1976	Romantic	91	United States	Don Chaffey	Michael Craig, John Meillon, Eva Griffith, Gra...	5.7	6.0	5	10	In the Australia of the pioneers, a boy and a ...	"Ecological" story with a happy ending, not wi...	1

```
In [22]: cm=CountVectorizer().fit_transform(df['imp_features'])
```

```
In [23]: cs=cosine_similarity(cm)
```

In [24]:
```
print(cs)
```

```
[[1.         0.         0.         ... 0.         0.         0.        ]
 [0.         1.         0.         ... 0.07071068 0.07559289 0.12510865]
 [0.         0.         1.         ... 0.03768892 0.         0.04445542]
 ...
 [0.         0.07071068 0.03768892 ... 1.         0.         0.07372098]
 [0.         0.07559289 0.         ... 0.         1.         0.        ]
 [0.         0.12510865 0.04445542 ... 0.07372098 0.         1.        ]]
```

In [27]:
```
title='Delusion'
rank=df[df.title==title]['filmtv_id'].values[0]
```

In [29]:
```
scores=list(enumerate(cs[rank]))
```

```
[(498, 0.1928791874526149), (984, 0.16329931618554525), (660, 0.13608276348795434), (341, 0.1276884796138123), (1, 0.12247448713915893), (228, 0.12247448713915893), (117, 0.1200961153538153 7), (424, 0.11785113019775793), (273, 0.11371470653683552), (448, 0.11371470653683552), (976, 0.11371470653683552), (633, 0.11180339887498895), (434, 0.10825317547305485), (499, 0.10540925533894598), (934, 0.10502100630210073), (663, 0.10350983390135315), (885, 0.10350983390135313), (12, 0.10206207261596577), (518, 0.10206207261596575), (64, 0.0936585811581694), (137, 0.09128709291752769), (24, 0.08908708063747481), (157, 0.08908708063747481), (174, 0.08908708063747481), (630, 0.08908708063747481), (155, 0.08703882797784893), (405, 0.08703882797784893), (881, 0.08703882797784893), (35, 0.08512565307587487), (370, 0.08512565307587487), (413, 0.08512565307587487), (610, 0.08512565307587487), (977, 0.08512565307587487), (990, 0.08512565307587487), (54, 0.08333333333333336), (90, 0.08333333333333336), (128, 0.08333333333333336), (491, 0.08333333333333336), (730, 0.08333333333333336), (759, 0.08333333333333336), (765, 0.08333333333333336), (856, 0.08333333333333336), (911, 0.08333333333333336), (364, 0.08164965809277262), (569, 0.08164965809277262), (576, 0.08164965809277262), (611, 0.08164965809277262), (79, 0.08006407690254358), (91, 0.08006407690254358), (124, 0.08006407690254358), (241, 0.08006407690254358), (261, 0.08006407690254358), (435, 0.08006407690254358), (528, 0.08006407690254358), (696, 0.08006407690254358), (755, 0.08006407690254358), (877, 0.08006407690254358), (958, 0.08006407690254358), (312, 0.07856742013183862), (451, 0.07856742013183862), (952, 0.07856742013183862), (628, 0.07715167498104596), (875, 0.07715167498104596), (806, 0.07580980435789035), (815, 0.07580980435789035), (924, 0.07580980435789035), (533, 0.07453559924999299), (543, 0.07453559924999299), (563, 0.07453559924999299), (847, 0.07453559924999299), (883, 0.07453559924999299), (551, 0.07332355575106 7667), (635, 0.07332355751067667), (657, 0.07332355751067667), (705, 0.07216878364870323), (903, 0.07106690545187017), (506, 0.06711560552140244), (619, 0.06375767130633384), (564, 0.06225728063646904), (302, 0.061545745489666376), (369, 0.055048188256318 04), (152, 0.04811252243246882), (20, 0.0468292905790847), (271, 0.0468292905790847), (770, 0.0468292905790847), (14
```

4. Recommend top 10 movies

In [38]:
```
sorted_score = sorted(sorted_score, key=lambda x: x[1], reverse=True)[:10]

print("Top 10 Recommended Movies:")

for i, item in enumerate(sorted_score):
    movie_id = item[0]

    filtered_df = df[df['filmtv_id'] == movie_id]

    if not filtered_df.empty:
        movie_title = filtered_df['title'].values[0]
        print(f"{i + 1}. {movie_title}")
    else:
        print(f"{i + 1}. Movie with ID {movie_id} not found")
```

```
Top 10 Recommended Movies:
1. Hope and Glory
2. Movie with ID 984 not found
3. Movie with ID 660 not found
4. An American in Paris
5. Movie with ID 1 not found
6. O Megalexandros
7. Goodbye Mr. Chips!
8. Amori in corso
9. Allemagne neuf zéro
10. Movie with ID 448 not found
```

Task 02

Fetch the data of any movie of your choice from any online platform using APIs or any web scraper module and create a ***'.csv'*** or ***'.xlsx'*** file. Data fetched must have: Movie id, Movie name, Movie genre, Movie actors, Movie directors, Movie description

Write a python program that implements cosine similarity to create a recommendation system and perform following task:

1. Read the data into pandas data frame and preprocess the fetched data

In [59]:

```
import requests
import pandas as pd

api_key = 'dd17d310'
movie_titles = [
    'Inception', 'The Matrix', 'Interstellar', 'The Dark Knight', 'Pulp Fiction',
    'The Godfather', 'The Shawshank Redemption', 'The Lord of the Rings: The Return of the King',
    'Forrest Gump', 'Fight Club', 'Star Wars: Episode IV - A New Hope', 'The Avengers',
    'Back to the Future', 'Gladiator', 'The Lion King', 'Jurassic Park', 'The Terminator',
    'Titanic', 'Avatar', 'The Silence of the Lambs', 'Saving Private Ryan', 'Schindler\'s List',
    'The Departed', 'The Prestige', 'The Usual Suspects', 'Se7en', 'The Sixth Sense',
    'The Green Mile', 'Braveheart', 'The Social Network', 'The Wolf of Wall Street',
    'Mad Max: Fury Road', 'Django Unchained', 'Inglourious Basterds', 'Shutter Island',
    'Memento', 'The Dark Knight Rises', 'Batman Begins', 'Blade Runner 2049', 'Guardians of the Galaxy',
    'Spider-Man: Into the Spider-Verse', 'The Incredibles', 'Finding Nemo', 'WALL-E',
    'Toy Story', 'Toy Story 3', 'Up', 'Coco', 'Ratatouille'
]
movies_data = []
for title in movie_titles:
    url = f'http://www.omdbapi.com/?t={title}&apikey={api_key}'
    response = requests.get(url)
    data = response.json()


    if data['Response'] == 'True':
        movies_data.append({
            'Movie ID': data.get('imdbID'),
            'Movie Name': data.get('Title'),
            'Movie Genre': data.get('Genre'),
            'Movie Actors': data.get('Actors'),
            'Movie Directors': data.get('Director'),
            'Movie Description': data.get('Plot')
        })

df = pd.DataFrame(movies_data)

df.to_csv('movies.csv', index=False)
```

In [60]:
```
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
```

In [61]:
```
df = pd.read_csv('movies.csv')
df.head()
```

Out[61]:

	Movie ID	Movie Name	Movie Genre	Movie Actors	Movie Directors	Movie Description
0	tt1375666	Inception	Action, Adventure, Sci-Fi	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellio...	Christopher Nolan	A thief who steals corporate secrets through t...
1	tt0133093	The Matrix	Action, Sci-Fi	Keanu Reeves, Laurence Fishburne, Carrie-Anne ...	Lana Wachowski, Lilly Wachowski	When a beautiful stranger leads computer hacke...
2	tt0816692	Interstellar	Adventure, Drama, Sci-Fi	Matthew McConaughey, Anne Hathaway, Jessica Ch...	Christopher Nolan	When Earth becomes uninhabitable in the future...
3	tt0468569	The Dark Knight	Action, Crime, Drama	Christian Bale, Heath Ledger, Aaron Eckhart	Christopher Nolan	When the menace known as the Joker wreaks havo...
4	tt0110912	Pulp Fiction	Crime, Drama	John Travolta, Uma Thurman, Samuel L. Jackson	Quentin Tarantino	The lives of two mob hitmen, a boxer, a gangst...

2. Data Analysis and Data visualization

In [62]:
```
df.shape
```

Out[62]: (49, 6)

In [63]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Movie ID           49 non-null     object
 1   Movie Name         49 non-null     object
 2   Movie Genre        49 non-null     object
 3   Movie Actors       49 non-null     object
 4   Movie Directors    49 non-null     object
 5   Movie Description  49 non-null     object
dtypes: object(6)
memory usage: 2.4+ KB
```

In [64]:
```
columns=['Movie Name', 'Movie Genre', 'Movie Directors', 'Movie Actors']
```

In [65]:
```
df[columns].head()
```

Out[65]:

	Movie Name	Movie Genre	Movie Directors	Movie Actors
0	Inception	Action, Adventure, Sci-Fi	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellio...
1	The Matrix	Action, Sci-Fi	Lana Wachowski, Lilly Wachowski	Keanu Reeves, Laurence Fishburne, Carrie-Anne ...
2	Interstellar	Adventure, Drama, Sci-Fi	Christopher Nolan	Matthew McConaughey, Anne Hathaway, Jessica Ch...
3	The Dark Knight	Action, Crime, Drama	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart
4	Pulp Fiction	Crime, Drama	Quentin Tarantino	John Travolta, Uma Thurman, Samuel L. Jackson

In [66]:
```
df[columns].isnull().values .any()
```

Out[66]: False

In [67]:
```
def get_imp_features(row):
    return f"{row['Movie Name']} {row['Movie Genre']} {row['Movie Directors']} {row['Movie Actors']}"
```

In [68]:
```
df['imp_features'] = df.apply(get_imp_features, axis=1)
df.head()
```

Out[68]:

	Movie ID	Movie Name	Movie Genre	Movie Actors	Movie Directors	Movie Description	imp_features
0	tt1375666	Inception	Action, Adventure, Sci-Fi	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellio...	Christopher Nolan	A thief who steals corporate secrets through t...	Inception Action, Adventure, Sci-Fi Christophe...
1	tt0133093	The Matrix	Action, Sci-Fi	Keanu Reeves, Laurence Fishburne, Carrie-Anne ...	Lana Wachowski, Lilly Wachowski	When a beautiful stranger leads computer hacke...	The Matrix Action, Sci-Fi Lana Wachowski, Lill...
2	tt0816692	Interstellar	Adventure, Drama, Sci-Fi	Matthew McConaughey, Anne Hathaway, Jessica Ch...	Christopher Nolan	When Earth becomes uninhabitable in the future...	Interstellar Adventure, Drama, Sci-Fi Christop...
3	tt0468569	The Dark Knight	Action, Crime, Drama	Christian Bale, Heath Ledger, Aaron Eckhart	Christopher Nolan	When the menace known as the Joker wreaks havo...	The Dark Knight Action, Crime, Drama Christoph...
4	tt0110912	Pulp Fiction	Crime, Drama	John Travolta, Uma Thurman, Samuel L. Jackson	Quentin Tarantino	The lives of two mob hitmen, a boxer, a gangst...	Pulp Fiction Crime, Drama Quentin Tarantino Jo...

3. Implement cosine similarity

In [69]:
```
cm=CountVectorizer().fit_transform(df['imp_features'])
```

In [70]:
```
cs=cosine_similarity(cm)
```

In [71]:
```
print(cs)
```

```
[[1.         0.18898224 0.37062466 ... 0.07142857 0.06900656 0.06681531]
 [0.18898224 1.         0.19611614 ... 0.         0.         0.        ]
 [0.37062466 0.19611614 1.         ... 0.07412493 0.14322297 0.06933752]
 ...
 [0.07142857 0.         0.07412493 ... 1.         0.13801311 0.20044593]
 [0.06900656 0.         0.14322297 ... 0.13801311 1.         0.12909944]
 [0.06681531 0.         0.06933752 ... 0.20044593 0.12909944 1.        ]]
```

In [72]:
```
title='The Dark Knight'
rank=df[df['Movie Name']==title].index[0]
```

In [73]:
```
scores=list(enumerate(cs[rank]))
```

In [74]:
```
sorted_score=sorted(scores, key=lambda x:x[1], reverse=True)
sorted_score=sorted_score[1:]

print(sorted_score)
```

```
[(36, 0.6210590034081188), (37, 0.5188745216627709), (23, 0.42857142857142866), (7, 0.27105237087157535), (19, 0.25197631533948484), (2, 0.2223747949983304), (5, 0.2223747949983304), (22, 0.2223747949983304), (0, 0.2142857142857143), (24, 0.2142857142857143), (27, 0.20701966780270625), (4, 0.15430334996209194), (6, 0.15430334996209194), (13, 0.15430334996209194), (25, 0.15430334996209194), (35, 0.15430334996209194), (29, 0.14824986333222026), (41, 0.14824986333222026), (11, 0.14285714285714288), (16, 0.14285714285714288), (26, 0.13801311186847084), (38, 0.13801311186847084), (39, 0.13801311186847084), (12, 0.13363062095621222), (30, 0.1336306209562122), (14, 0.12964074471043288), (1, 0.12598815766974242), (40, 0.11145564251507058), (9, 0.08058229640253803), (17, 0.08058229640253803), (8, 0.07715167498104597), (18, 0.07715167498104597), (32, 0.07715167498104597), (43, 0.07715167498104597), (21, 0.07412493166611013), (33, 0.07412493166611013), (34, 0.07412493166611013), (15, 0.07142857142857144), (20, 0.06900655593423542), (47, 0.06900655593423542), (28, 0.0668153104781061), (31, 0.066815310478106
```

4. Recommend top 5 movies

In [77]:
```
sorted_score = sorted_score[1:6]

movie_indices = [i[0] for i in sorted_score]
```

In [79]:
```
recommendations = df.iloc[movie_indices][['Movie Name', 'Movie Genre', 'Movie Actors', 'Movie Directors', 'Movie Description'

if not recommendations.empty:
    print("Top 5 Recommended Movies:")
    for i, row in recommendations.iterrows():
        print(f"\nMovie {i + 1}:")
        print(f"Name: {row['Movie Name']}")
```

```
Top 5 Recommended Movies:

Movie 24:
Name: The Prestige

Movie 8:
Name: The Lord of the Rings: The Return of the King

Movie 20:
Name: The Silence of the Lambs

Movie 3:
Name: Interstellar

Movie 6:
Name: The Godfather
```

Department of COMPUTER SCIENCE
Bahria University
Discovering Knowledge

CSL-487: Introduction to Data Science Lab
Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Lab 12

Task 01

As a member of financial analysis team at **InvestSmart Analytics,** you are tasked with conducting a comprehensive analysis of the financial indicators of US stocks over the period from 2014 to 2018 from dataset IDS_Lab12.zip. This analysis is crucial for their upcoming investment strategy review. Focus on key financial metrics such as revenue growth, quick ratio, current ratio, debt equity ratio, inventory turnover, short term assets, long term assets etc.

Write a python program and perform following task:

1. Perform EDA (Exploratory Data Analysis).
2. Create a machine learning model using logistic regression and KNN classifiation over the data.
3. Perfrom a complete comparative analysis based on the confusion matrix and acuracy score of the model.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df_2014=pd.read_csv('/2014_Financial_Data.csv')
df_2015=pd.read_csv('/2015_Financial_Data.csv')
df_2016=pd.read_csv('/2016_Financial_Data.csv')
df_2017=pd.read_csv('/2017_Financial_Data.csv')
df_2018=pd.read_csv('/2018_Financial_Data.csv')
df_2014.head()
```

	Unnamed: 0	Revenue	Revenue Growth	Cost of Revenue	Gross Profit	R&D Expenses	SG&A Expense	Operating Expenses	Operating Income	Inte Exp
0	PG	7.440100e+10	-0.0713	3.903000e+10	3.537100e+10	0.000000e+00	2.146100e+10	2.146100e+10	1.391000e+10	7.090000
1	VIPS	3.734148e+09	1.1737	2.805625e+09	9.285226e+08	1.083303e+08	3.441414e+08	7.939267e+08	1.345959e+08	1.214869
2	KR	9.837500e+10	0.0182	7.813800e+10	2.023700e+10	0.000000e+00	1.519600e+10	1.751200e+10	2.725000e+09	4.430000
3	RAD	2.552641e+10	0.0053	1.820268e+10	7.323734e+09	0.000000e+00	6.561162e+09	6.586482e+09	7.372520e+08	4.245910

```
[4] print("Shape of the data of year 2014 is: {}".format(df_2014.shape))
    print("Shape of the data of year 2015 is: {}".format(df_2015.shape))
    print("Shape of the data of year 2016 is: {}".format(df_2016.shape))
    print("Shape of the data of year 2017 is: {}".format(df_2017.shape))
    print("Shape of the data of year 2018 is: {}".format(df_2018.shape))
```

```
Shape of the data of year 2014 is: (3808, 225)
Shape of the data of year 2015 is: (4120, 225)
Shape of the data of year 2016 is: (4797, 225)
Shape of the data of year 2017 is: (4960, 225)
Shape of the data of year 2018 is: (4392, 225)
```

```
[6] df_2016.isna().sum()
```

```
Unnamed: 0                0
Revenue                 489
Revenue Growth          649
Cost of Revenue         640
Gross Profit            492
                       ...
R&D Expense Growth      772
SG&A Expenses Growth    756
Sector                    0
2017 PRICE VAR [%]        0
Class                     0
Length: 225, dtype: int64
```

```
[7] df_2014.fillna(0,inplace=True)
    df_2015.fillna(0,inplace=True)
    df_2016.fillna(0,inplace=True)
    df_2017.fillna(0,inplace=True)
    df_2018.fillna(0,inplace=True)
```

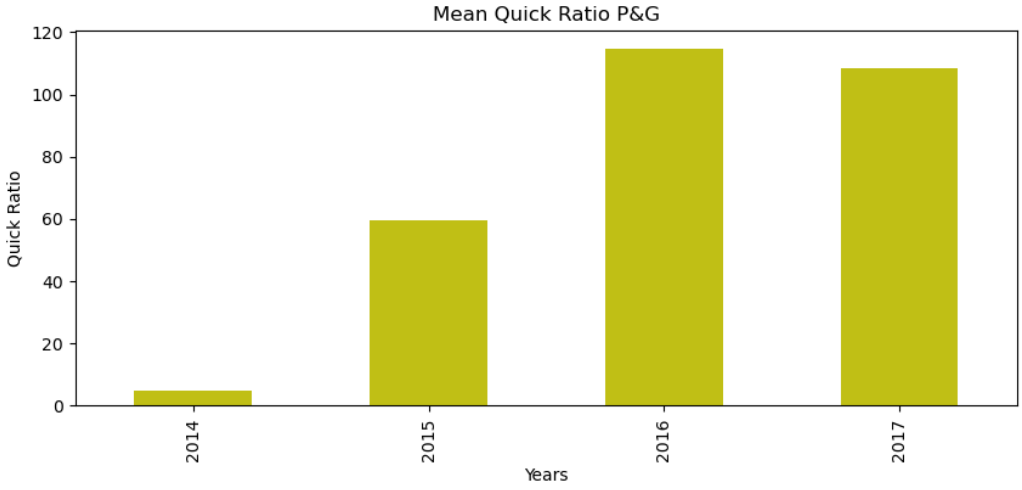
```
[8] df_2016.isna().sum()
```

```
Unnamed: 0              0
Revenue                 0
Revenue Growth          0
Cost of Revenue         0
Gross Profit            0
                       ..
R&D Expense Growth      0
SG&A Expenses Growth    0
Sector                  0
2017 PRICE VAR [%]      0
Class                   0
Length: 225, dtype: int64
```

```
[12] df_all = pd.concat([df_2014, df_2015, df_2016, df_2017, df_2018], axis=0)
     df_all.fillna(0, inplace=True)
     years = [2014, 2015, 2016, 2017, 2018]
     repeated_years = []

     for year, df in zip(years, [df_2014, df_2015, df_2016, df_2017, df_2018]):
         repeated_years.extend([year] * len(df))
     df_all.index = repeated_years
```

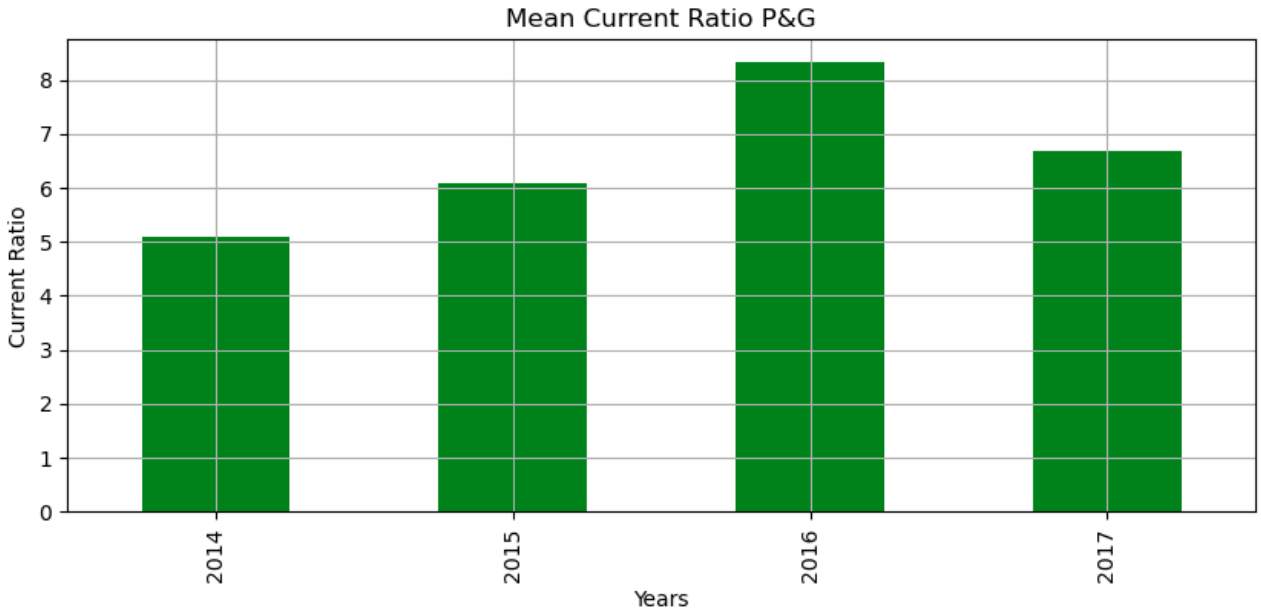
```
[16] if 'quickRatio' in df_all.columns:
         mean_quick_ratios = df_all.groupby(df_all.index)['quickRatio'].mean()

         print("Mean Quick Ratio for each year:")
         print(mean_quick_ratios)

         plt.figure(figsize=(15, 7))
         mean_quick_ratios.plot.bar(color='g')
         plt.xlabel('Years')
         plt.ylabel('Quick Ratio')
         plt.title('Mean Quick Ratio P&G')
         plt.grid(False)
         plt.show()
     else:
         print("The column 'quickRatio' does not exist in the dataset.")
```

```
Mean Quick Ratio for each year:
2014      4.695566
2015     59.389867
2016    114.712549
2017    108.315809
2018      3.842980
Name: quickRatio, dtype: float64
```



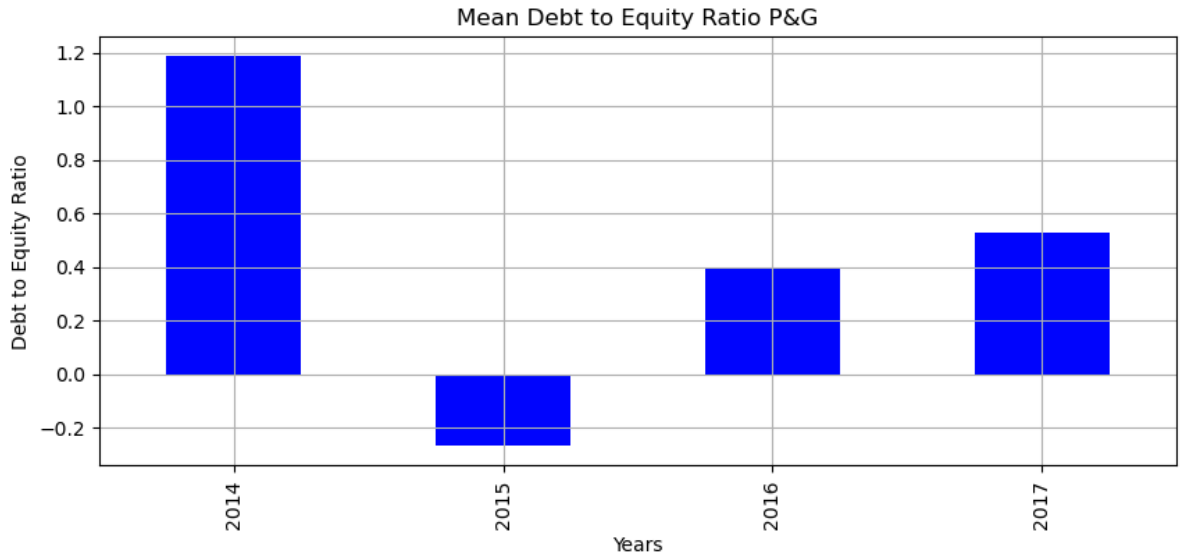
```
[17] if 'currentRatio' in df_all.columns:
        mean_current_ratios = df_all.groupby(df_all.index)['currentRatio'].mean()

        print("Mean Current Ratio for each year:")
        print(mean_current_ratios)

        plt.figure(figsize=(15, 7))
        mean_current_ratios.plot.bar(color='b')
        plt.xlabel('Years')
        plt.ylabel('Current Ratio')
        plt.title('Mean Current Ratio P&G')
        plt.grid(True)
        plt.show()
    else:
        print("The column 'currentRatio' does not exist in the dataset.")
```

```
Mean Current Ratio for each year:
2014    5.103073
2015    6.089222
2016    8.333480
2017    6.681410
2018    4.026999
Name: currentRatio, dtype: float64
```



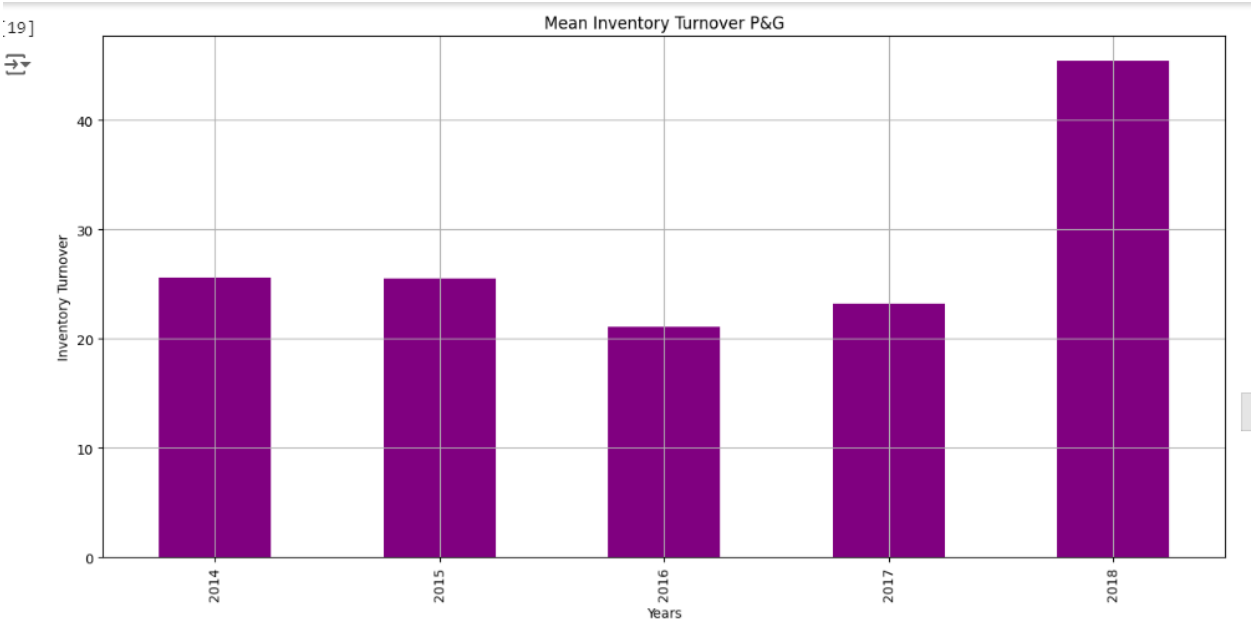
```
[18] if 'debtEquityRatio' in df_all.columns:

        mean_debt_equity_ratios = df_all.groupby(df_all.index)['debtEquityRatio'].mean()

        print("Mean Debt to Equity Ratio for each year:")
        print(mean_debt_equity_ratios)

        plt.figure(figsize=(15, 7))
        mean_debt_equity_ratios.plot.bar(color='r')
        plt.xlabel('Years')
        plt.ylabel('Debt to Equity Ratio')
        plt.title('Mean Debt to Equity Ratio P&G')
        plt.grid(True)
        plt.show()
    else:
        print("The column 'debtEquityRatio' does not exist in the dataset.")
```

```
Mean Debt to Equity Ratio for each year:
2014    1.187114
2015   -0.267735
2016    0.393225
2017    0.527340
2018    0.729422
Name: debtEquityRatio, dtype: float64
```



```
[19] if 'inventoryTurnover' in df_all.columns:

        mean_inventory_turnovers = df_all.groupby(df_all.index)['inventoryTurnover'].mean()

        print("Mean Inventory Turnover for each year:")
        print(mean_inventory_turnovers)

        plt.figure(figsize=(15, 7))
        mean_inventory_turnovers.plot.bar(color='purple')
        plt.xlabel('Years')
        plt.ylabel('Inventory Turnover')
        plt.title('Mean Inventory Turnover P&G')
        plt.grid(True)
        plt.show()
    else:
        print("The column 'inventoryTurnover' does not exist in the dataset.")
```

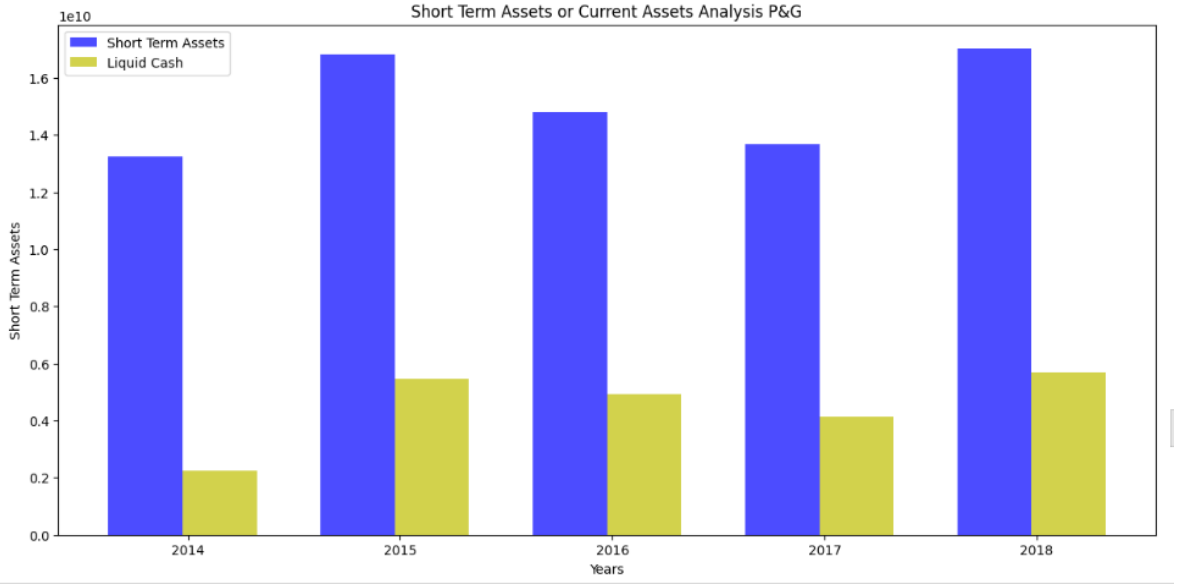
```
Mean Inventory Turnover for each year:
2014    25.650086
2015    25.557572
2016    21.079613
2017    23.206634
2018    45.430480
Name: inventoryTurnover, dtype: float64
```



Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

```
[21] mean_short_term_assets = df_all.groupby(df_all.index)['ShortTermAssets'].mean()

     print("Mean of Short Term Assets for P&G last 5 years is {:.2f}".format(mean_short_term_assets.mean()))

     n_years = 5
     index = np.arange(n_years)
     bar_width = 0.35
     opacity = 0.7

     plt.figure(figsize=(15, 7))
     plt.bar(index, mean_short_term_assets, bar_width, alpha=opacity, color='b', label='Short Term Assets')
     plt.bar(index + bar_width, df_all.groupby(df_all.index)['LiquidCash'].mean(), bar_width, alpha=opacity, color='y', label='L

     plt.xlabel('Years')
     plt.ylabel('Short Term Assets')
     plt.title('Short Term Assets or Current Assets Analysis P&G')
     plt.xticks(index + 0.20, mean_short_term_assets.index)
     plt.grid(False)
     plt.legend()
     plt.show()
```

```
[21] Mean of Short Term Assets for P&G last 5 years is 15110908342.73
```



Insert code cell below
Ctrl+M B

```
[28]: import numpy as np
      import pandas as pd
      from sklearn.model_selection import train_test_split
      from sklearn.preprocessing import StandardScaler
      from sklearn.linear_model import LogisticRegression
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn.metrics import accuracy_score

      features = [
          'Revenue', 'Revenue Growth', 'Cost of Revenue', 'Gross Profit',
          'R&D Expenses', 'SG&A Expense', 'Operating Expenses', 'Operating Income',
          'Interest Expense', 'Receivables growth', 'Inventory Growth', 'Asset Growth',
          'Book Value per Share Growth', 'Debt Growth', 'R&D Expense Growth',
          'SG&A Expenses Growth'
      ]
      target = '2015 PRICE VAR [%]'

      df_all.fillna(0, inplace=True)

      X = df_all[features]
      y = (df_all[target] > threshold).astype(int)

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

      scaler = StandardScaler()
```

```
df_all.fillna(0, inplace=True)

X = df_all[features]
y = (df_all[target] > threshold).astype(int)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

log_reg = LogisticRegression()
log_reg.fit(X_train_scaled, y_train)

knn = KNeighborsClassifier()
knn.fit(X_train_scaled, y_train)

y_pred_log_reg = log_reg.predict(X_test_scaled)
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
```

```
log_reg = LogisticRegression()
log_reg.fit(X_train_scaled, y_train)

knn = KNeighborsClassifier()
knn.fit(X_train_scaled, y_train)

y_pred_log_reg = log_reg.predict(X_test_scaled)
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
print("Accuracy of Logistic Regression:", accuracy_log_reg)

y_pred_knn = knn.predict(X_test_scaled)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
print("Accuracy of KNN Classification:", accuracy_knn)
```

```
Accuracy of Logistic Regression: 0.9252717391304348
Accuracy of KNN Classification: 0.9207427536231884
```

```
[26] from sklearn.metrics import confusion_matrix

     log_reg_cm = confusion_matrix(y_test, y_pred_log_reg)
     log_reg_accuracy = accuracy_score(y_test, y_pred_log_reg)

     knn_cm = confusion_matrix(y_test, y_pred_knn)
     knn_accuracy = accuracy_score(y_test, y_pred_knn)


     print("Confusion Matrix - Logistic Regression:")
     print(log_reg_cm)

     print("\nConfusion Matrix - KNN Classification:")
     print(knn_cm)
```

```
Confusion Matrix - Logistic Regression:
[[4086    1]
 [ 329    0]]

Confusion Matrix - KNN Classification:
[[4063   24]
 [ 326    3]]
```

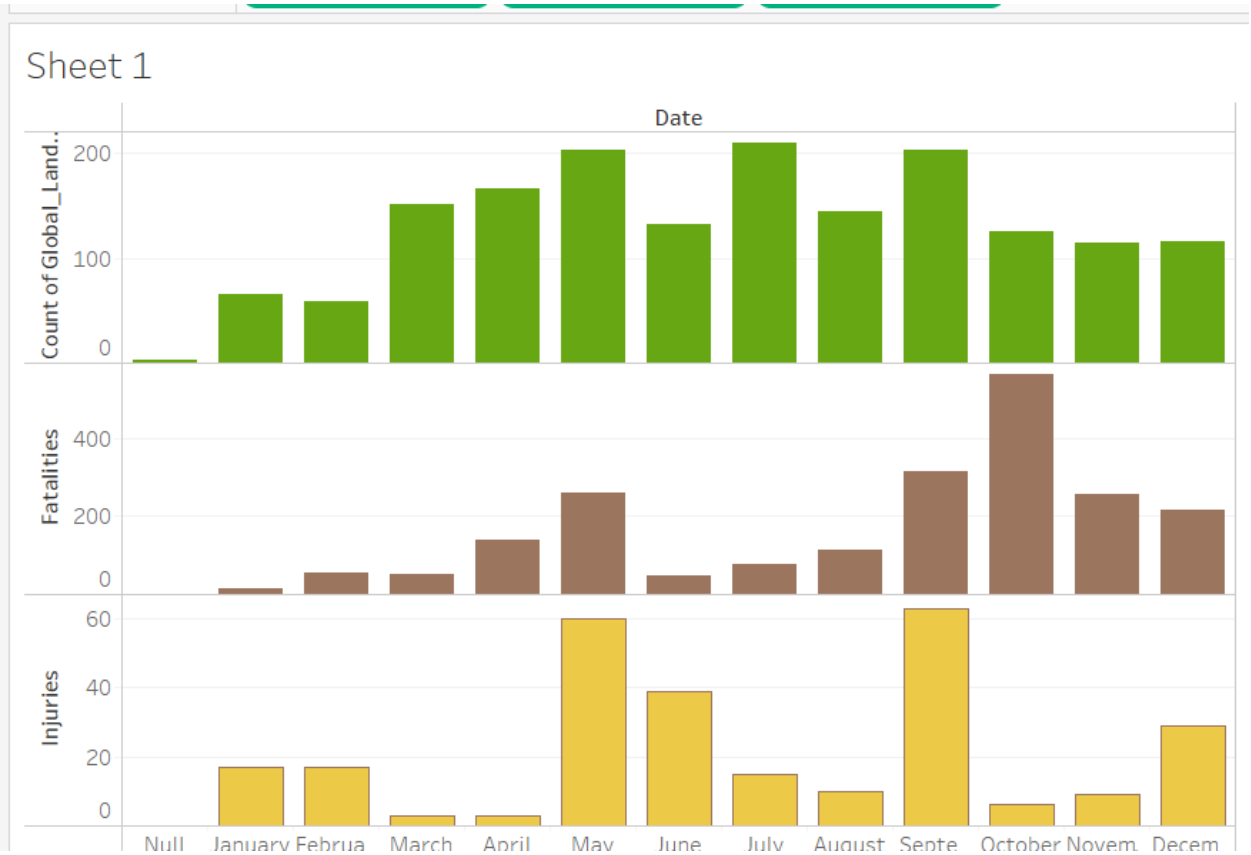
Department of COMPUTER SCIENCE
Bahria University
Discovering Knowledge

CSL-487: Introduction to Data Science Lab
Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Lab 13 Visualization in tableau

Exercise 01:

Which month is the **safest month**? Show the total number of landslides, the number of injured persons and the number fatalities.



1. From a **total number of events** points of view show the safest month.
   FEB

2. From a **number of injured persons** points of view show the safest months.
   MARCH April

3. From a **number of fatalities** point of view show the safest month**.**
   JANUARY

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A

Exercise 02:

Question: Which landslide types and how many of them happen in the Dominican Republic, Cuba and Puerto Rico?

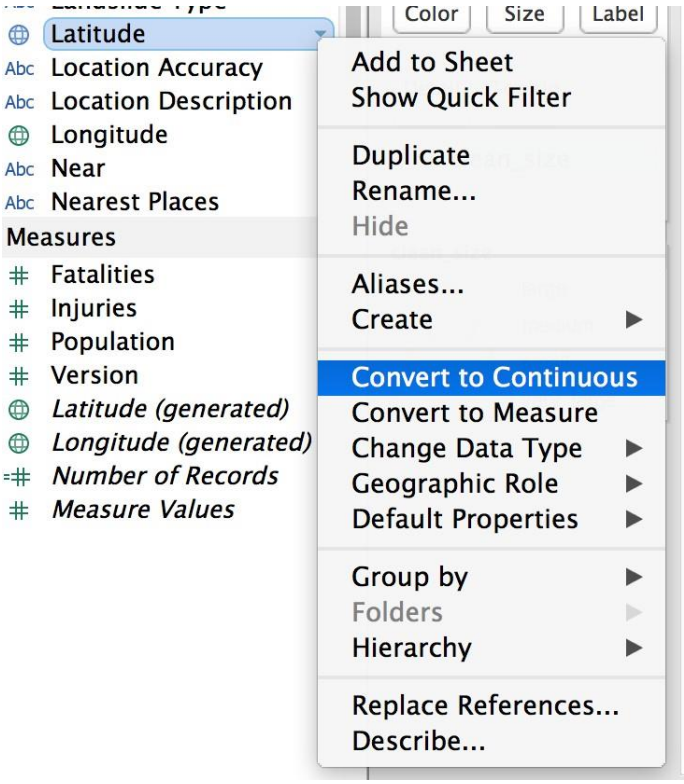
Tips: Use Filters (hint - **Quickfilter**) and check the dimensions data quality

Sheet 1

Countryname	Landslide Type: Complex	Landslide	Mudslide	Other
Cuba	1	2		
Dominican Republic	2	9	4	
Puerto Rico	1	5		1

Exercise 03:

Use the mapping functionality of Tableau and create an interactive Dashboard with at least 2 views, to explore the dataset further. Show us what you can find out about triggers, size, type, location, impact and other things!



To use the location data you can use the country information, but there are **null** values in this column. To get **all** data points, you have to **Convert Latitude** and **Longitude** to **Continuous**. After that double click both and the map is ready to go.

Department of COMPUTER SCIENCE
Bahria University
Discovering Knowledge

CSL-487: Introduction to Data Science Lab

Name: Ahmed Farrukh
Enrollment: 02-134212-030
BS(CS)- Semester 6-A