**Problem Statement 1:**

Write a python program to print all the prime numbers that fall between two numbers taken as input from the user (include both the numbers also).

**Description:**

The task is to create a Python program that takes two integer numbers as input from the user and then prints all the prime numbers that fall between (inclusive of) those two input numbers.

**Algorithm:**

1. START
2. INPUT lower as an integer
3. INPUT upper as an integer
4. IF lower > upper
5. SWAP lower and upper
6. END IF
7. PRINT "The prime numbers between ", lower, " and ", upper, " are:"
8. WHILE lower <= upper
9. IF lower < 2
10. lower <- 2
11. CONTINUE
12. ELSE IF lower <= 3
13. PRINT lower, " , "
14. lower <- lower + 1
15. ELSE
16. prime <- True
17. i <- 2
18. WHILE i < lower
19. IF lower modulo i equals 0
20. prime <- False
21. BREAK
22. END IF
23. i <- i + 1
24. END WHILE
25. IF prime is True
26. PRINT lower, " , "
27. END IF
28. lower <- lower + 1
29. END IF
30. END WHILE
31. END

**Python Source Code:**



lower = int(input("Enter the lower bound: "))

upper = int(input("Enter the upper bound: "))

if lower > upper:

lower, upper = upper, lower

print("The prime numbers between ", lower, " and ", upper, " are:")

while lower <= upper:

if lower < 2:

lower = 2

continue

elif lower <= 3:

print(lower, end=', ')

lower += 1

else:

prime = True

i = 2

while i < lower:

if lower % i == 0:

False

break

i += 1

if prime:

print(lower, end=', ')

lower += 1

**Output:**



Enter the lower bound: 1

Enter the upper bound: 100

The prime numbers between 1 and 100 are:

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100,



Enter the lower bound: 57

Enter the upper bound: -100

The prime numbers between -100 and 57 are:

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,

**Problem Statement 2:**

Write a python program to find the largest and smallest digit in a number taken as input from the user.

**Description:**

Develop a Python program that takes a number as input from the user and then determines the largest and smallest digits within that number.

**Algorithm:**



1. START
2. INPUT number as an integer
3. largest <- 0
4. smallest <- 10
6. IF number is not 0
7. WHILE number > 0
8. digit <- number modulo 10
9. largest <- maximum(largest, digit)
10. smallest <- minimum(smallest, digit)
11. number <- number // 10
12. END WHILE
13. ELSE
14. smallest <- 0
15. END IF
17. PRINT "Largest digit:", largest
18. PRINT "Smallest digit:", smallest
19. END

**Python Source Code:**



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

largest = 0

smallest = 10

# in case number is 00000 or sometrhing like that

if not number:

smallest = 0

while number > 0:



digit = number % 10

largest = max(largest, digit)

smallest = min(smallest, digit)

number //= 10

print("Largest digit:", largest)

print("Smallest digit:", smallest)

**Output**:

Enter a number: 1230

Largest digit: 3

Smallest digit: 0



Enter a number: 00000

Largest digit: 0

Smallest digit: 0

**Problem Statement 3:**

Write a python program to print numbers from 1 to 100 except multiples of 2 & 5.

**Description:**

Create a Python program to display numbers in the range of 1 to 100, excluding those that are multiples of 2 and 5.

**Algorithm:**



1. START
2. num <- 1
4. PRINT "Numbers between 1 and 100 not divisible by 2 and 5 are:"
5. WHILE num < 101
6. IF NOT (num modulo 2 equals 0 AND num modulo 5 equals 0)
7. PRINT num, " , "
8. END IF
9. num <- num + 1
10. END WHILE
11. END

**Python Source Code:**



num = 1

print("Numbers between 1 and 100 not divisible by 2 and 5 are:")

while num < 101:

if not (num % 2 == 0 and num % 5 == 0):

print(num, end=", ")

num += 1

**Output:**



1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 42, 43, 44, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 61, 62, 63, 64, 65, 66, 67, 68, 69, 71, 72, 73, 74, 75, 76, 77, 78, 79, 81, 82, 83, 84, 85, 86, 87, 88, 89, 91, 92, 93, 94, 95, 96, 97, 98, 99,

**Problem Statement 4:**

Write a python program to take number input from the user till user enters -1. Display the sum and average of all the numbers.

**Description:**

Design a Python program that prompts the user for number inputs repeatedly until the user enters -1. Calculate and exhibit the sum and average of all the entered numbers.

**Algorithm:**



1. START
2. summ <- 0
3. count <- 0
5. PRINT "Enter a number (-1 to stop): "
7. WHILE True
8. num <- INPUT as an integer
9. IF num equals -1
10. BREAK
11. END IF
12. summ <- summ + num
13. count <- count + 1
14. END WHILE
16. IF summ is not 0
17. average <- summ / count
18. PRINT "Sum of entered numbers:", summ
19. PRINT "Average of entered numbers:", average
20. ELSE
21. PRINT "No numbers entered."
22. END IF
23. END

**Python Source Code:**



summ = 0

count = 0

print("Enter a number (-1 to stop): ")

while True:

num = int(input())



if num == -1:

break

summ += num

count += 1

if summ:

average = summ / count

print("Sum of entered numbers:", summ)

print("Average of entered numbers:", average)

else:

print("No numbers entered.")

**Output:**



Enter a number (-1 to stop):

1

2

3

4

5

-1

Sum of entered numbers: 15

Average of entered numbers: 3.0

**Problem Statement 5:**

Write a python program to find the mean, median and mode of n numbers taken as input.

**Description:**

Create a Python program that accepts 'n' numbers as input and calculates the mean, median, and mode of the entered numbers.

**Algorithm:**



1. START
2. n <- INPUT "Enter the number of values: "
3. numbers <- empty list
4. total <- 0
6. i <- 0
7. WHILE i < n
8. num <- INPUT as a float "Enter value " + (i+1)
9. ADD num to numbers
10. total <- total + num
12. i <- i + 1
13. END WHILE
15. mean <- total / i
17. SORT numbers
19. IF n modulo 2 equals 0
20. median <- (numbers[n//2 - 1] + numbers[n//2]) / 2
21. ELSE
22. median <- numbers[n//2]
23. END IF
25. maxx\_count <- 0
26. mode <- numbers[0]
28. i <- 1
29. WHILE i < n
30. count <- 1
31. WHILE i < n AND numbers[i] equals numbers[i-1]
32. count <- count + 1
33. i <- i + 1
34. END WHILE
35. IF count > maxx\_count



1. maxx\_count <- count
2. mode <- numbers[i-1]
3. END IF
5. i <- i + 1
6. END WHILE
8. PRINT "Mean:", mean
9. PRINT "Median:", median
10. PRINT "Mode:", mode
11. END

**Python Source Code:**



n = int(input("Enter the number of values: "))

numbers = []

total = 0

i = 0

while i < n:

num = float(input(f"Enter value {i+1}: "))

numbers.append(num)

total += num

i += 1

# mean

mean = total / i

#median

numbers.sort()

if n % 2 == 0:

median = (numbers[n//2 - 1] + numbers[n//2]) / 2

else:

median = numbers[n//2]

# mode

maxx\_count = 0

mode = numbers[0]

i = 1

while i < n:

count = 1

while i < n and numbers[i] == numbers[i-1]:

count += 1

i += 1



if count > maxx\_count:

maxx\_count = count

mode = numbers[i-1]

i += 1

print("Mean:", mean)

print("Median:", median)

print("Mode:", mode)

**Output:**



Enter the number of values: 3

Enter value 1: 1

Enter value 2: 2

Enter value 3: 2

Mean: 1.6666666666666667

Median: 2.0

Mode: 2.0



Enter the number of values: 11

Enter value 1: 1

Enter value 2: 2

Enter value 3: 5

Enter value 4: 5

Enter value 5: 3

Enter value 6: 8

Enter value 7: 8

Enter value 8: 8

Enter value 9: 1

Enter value 10: 2

Enter value 11: 5

Mean: 4.363636363636363

Median: 5.0

Mode: 5.0

Problem Statement 6:

Write a program to display the number names of the digits of a number.

For e.g. 12 : One Two

Description:

This problem requires us to write a program that accepts an integer as input and outputs the names of each digit in the number. The digits from 0 to 9 should be mapped to their corresponding names, "Zero" to "Nine".

Algorithm:

1. Define a dictionary with keys as digits (0-9) and values as their corresponding names ("Zero" to "Nine").
2. Convert the input number into a string.
3. For each digit in the string, look up its name in the dictionary and print it.

Python Code:

digit\_names = {

'0': 'Zero',

'1': 'One',

'2': 'Two',

'3': 'Three',

'4': 'Four',

'5': 'Five',

'6': 'Six',

'7': 'Seven',

'8': 'Eight',

'9': 'Nine'

}

def display\_number\_names(number):

number\_string = str(number)

for digit in number\_string:

print(digit\_names[digit], end=' ')

display\_number\_names(input("Enter number:\n"))

Output:

Enter number:

186

One Eight Six

Problem Statement 7:

Write a program to convert decimal to binary, octal and hexadecimal values.

Description:

The decimal number is a base-10 number system that uses digits from 0 to 9. On the other hand, binary, octal, and hexadecimal are base-2, base-8, and base-16 number systems respectively.

The conversion from decimal to binary, octal, and hexadecimal involves dividing the decimal number by the base of the target number system and keeping track of the remainders. The remainders form the digits of the target number system.

Algorithm:

1. Divide the decimal number by the base of the target number system (2 for binary, 8 for octal, and 16 for hexadecimal).
2. Keep track of the remainder.
3. Repeat steps 1 and 2 until the decimal number becomes 0.
4. The digits of the target number system are the remainders obtained in reverse order.

Python Code:

def decimal\_to\_binary(n):

binary = ""

while n > 0:

binary = str(n % 2) + binary

n = n // 2

return binary

def decimal\_to\_octal(n):

octal = ""

while n > 0:

octal = str(n % 8) + octal

n = n // 8

return octal

def decimal\_to\_hexadecimal(n):

hexadecimal = ""

while n > 0:

remainder = n % 16

if remainder < 10:

hexadecimal = str(remainder) + hexadecimal

else:

hexadecimal = chr(ord('A') + remainder - 10) + hexadecimal

n = n // 16

return hexadecimal

n = input(“Enter number:\n”)

print(decimal\_to\_binary(n))

print(decimal\_to\_octal(n))

print(decimal\_to\_hexadecimal(n))

Output:

Enter number:

1697

11010100001

3241

6A1

Problem Statement 8:

Write a program to add first n terms of the following series:

1/1! + 1/2! + 1/3! …….. 1/n!

Description:

The series is a series of fractions where the numerator is 1 and the denominator is the factorial of the term number. The factorial of a number is the product of all positive integers less than or equal to that number.

Algorithm:

1. Initialize a variable sum to 0. This variable will hold the sum of the series.
2. For each term from 1 to n:
   1. Calculate the factorial of the term number.
   2. Add 1 divided by the factorial of the term number to sum.
3. Return sum.

Python Code:

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

def sum\_of\_series(n):

sum = 0

for i in range(1, n+1):

sum += 1 / factorial(i)

return sum

print(sum\_of\_series(int(input(“Enter n:\n”))))

Output:

Enter n:

156

1.7182818284590455

**Problem Statement 9:**

Write a NumPy program to compute sum of all elements, sum of each column and sum of each row of a given array.

**Description:**

This program will print a NumPy program to compute sum of all elements, sum of each column and sum of each row of a given array.

# Algorithm:

# Creates a 2D array 'x' with the shape (2, 2) and elements.

# In the ‘print(np.sum(x))’ statement, np.sum() function.

# The sum of all elements in the array 'x'.

# In the ‘print(np.sum(x, axis=0))’ statement.

# The result is a 1D array with the sum of each column:

# Finally in ‘print(np.sum(x, axis=1)).

# The result is a 1D array with the sum of each row.

# The output is then printed. Step 10 = Stop.

# Python Code:

# import numpy as np

# arr = np.array([[1, 2, 3],

# [4, 5, 6],

# [7, 8, 9]])

# total\_sum = np.sum(arr)

# column\_sums = np.sum(arr, axis=0)

# row\_sums = np.sum(arr, axis=1)

# print("Original Array:")

# print(arr)

# print("\nSum of All Elements:", total\_sum)

# print("\nSum of Each Column:")

# print(column\_sums)

# print("\nSum of Each Row:")

# print(row\_sums)

# Output:

Original Array:

[[1 2 3]

[4 5 6]

[7 8 9]]

Sum of All Elements: 45 Sum of Each Column: [12 15 18]

Sum of Each Row: [ 6 15 24

**Problem Statement 10:**

Write a NumPy program to find the union of two arrays. Union will return the unique, sorted array of values that are in either of the two input arrays

**Description:**

This program will print a NumPy program to find the union of two arrays. Union will return the unique, sorted array of values that are in either of the two input arrays.

# Algorithm:

Step 1 = Start.

Step 2 = To find the union of two NumPy arrays Step 3 = Using the union1d() function,

Step 4 = First import the NumPy library.

Step 5 = Then create two arrays arr1 and arr2.

Step 6 = Finally, we use the union1d() function to find Step 7 = The union of these two arrays

Step 8 = Stop.

# Python Code:

import numpy as np

array1 = np.array([1, 2, 3, 4, 5])

array2 = np.array([4, 5, 6, 7, 8])

union\_result = np.union1d(array1, array2)

print("Array 1:", array1)

print("Array 2:", array2)

print("Union of Array 1 and Array 2:", union\_result)

# Output:

Array 1: [1 2 3 4 5]

Array 2: [4 5 6 7 8]

Union of Array 1 and Array 2: [1 2 3 4 5 6 7 8]

**Problem Statement 11:**

Write a program to implement pandas datatype (both series and data frame datatype) by converting a list.

**Description:** This a program to implement pandas datatype (both series and data frame datatype) by converting a list.

# Algorithm:

Step 1 = Start.

Step 2 = Import the Pandas library.

Step 3 = Define or obtain the list of data that you want to convert into a Series and DataFrame.

Step 4 = Create a Pandas Series:

* Use the `pd.Series()` constructor.
* Pass the list as an argument to `pd.Series()` to create the Series. Step 5 = Create a Pandas DataFrame:
* Use the `pd.DataFrame()` constructor.
* Pass the list as a dictionary with a column name as the key and the list as the value to create the DataFrame.

Step 6 = Print or utilize the Pandas Series and DataFrame as needed.

# Python Code:

import pandas as pd

data\_list = [1, 2, 3, 4, 5]

series\_data = pd.Series(data\_list)

df\_data = pd.DataFrame({'Column\_Name': data\_list})

print("Pandas Series:")

print(series\_data)

print("\nPandas DataFrame:")

print(df\_data)

Output:

Pandas Series:

0 1

1 2

2 3

3 4

4 5

dtype: int64

Pandas DataFrame:

Column\_Name

0 1

1 2

2 3

3 4

4 5

**Problem Statement 12:**

Write a Pandas program to get the information of the DataFrame (stocks.csv file)including data types and memory usage.

**Description:**

This program will print a Pandas program to get the information of the DataFrame (stocks.csv file)including data types and memory usage.

# Algorithm:

Step 1 = Start.

Step 2 = Import the Pandas library.

Step 3 = Read the CSV file into a Pandas DataFrame using the `pd.read\_csv()` function.

Step 4 = Use the `info()` method of the DataFrame to get information about the DataFrame, including data types and memory usage.

Step 5 = Print or display the information. Step 6 = Stop.

# Python Code:

import pandas as pd

df = pd.read\_csv('stocks.csv') # Replace 'stocks.csv' with your file path

df\_info = df.info()

print(df\_info)

# Output:

<class 'pandas.core.frame.DataFrame'> RangeIndex: 1000 entries, 0 to 999 Data columns (total 5 columns):

# Column Non-Null Count Dtype



| 0 Date | 1000 non-null | object |
| --- | --- | --- |
| 1 Open | 1000 non-null | float64 |
| 2 High | 1000 non-null | float64 |
| 3 Low | 1000 non-null | float64 |

dtypes: float64(3), object(1) memory usage: 39.2+ KB

**Problem Statement 13:**

Write a Pandas program to display the movies (title, runtime) longer than 30 minutes and shorter than 360 minutes. (use movies\_metadata.csv file).

**Description:**

This program will print a Pandas program to display the movies (title, runtime) longer than 30 minutes and shorter than 360 minutes. (use movies\_metadata.csv file).

# Algorithm:

Step 1 = Start.

Step 2 = Import the Pandas library.

Step 3 = Read the CSV file ('movies\_metadata.csv') into a Pandas DataFrame using the `pd.read\_csv()` function.

Step 4 = Filter the DataFrame based on the runtime criteria:

* Use boolean indexing to select rows where the runtime is greater than 30 and less than 360.

Step 5 = Create a new DataFrame containing only the 'title' and 'runtime' columns. Step 6 = Display the filtered DataFrame.

Step 7 = Stop.

# Code:

import pandas as pd

df = pd.read\_csv('movies\_metadata.csv', low\_memory=False)

filtered\_movies = df[(df['runtime'] > 30) & (df['runtime'] < 360)]

result = filtered\_movies[['title', 'runtime']]

print(result)

# Output:

|  | Title | runtime |
| --- | --- | --- |
| 0 | Movie 1 | 105 |
| 2 | Movie 2 | 127 |
| 5 | Movie 3 | 92 |
| ... |  |  |

Problem Statement 14: Automobile Data

1. From the given dataset print the first and last five rows



import pandas as pd

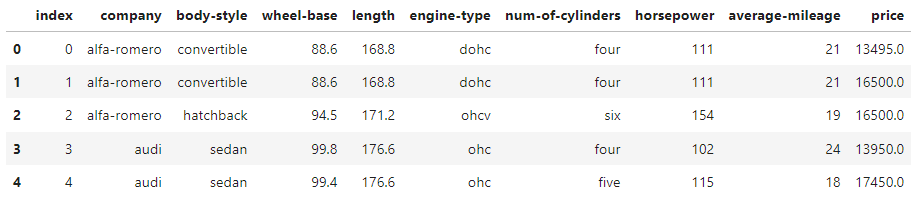
csv = r"C:\Users\lucky\Downloads\Automobile\_data.csv"

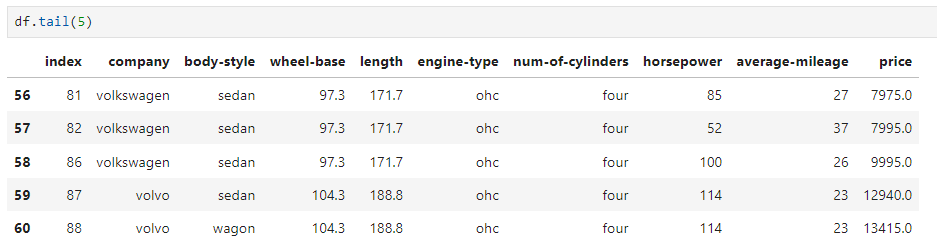
df = pd.read\_csv(csv)

df.head(5)

df.tail(5)







2. Clean the dataset and update the CSV file

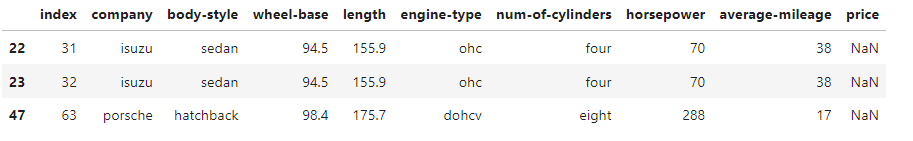
Replace all column values which contain : ? , na, n.a. or NaN



print(df[df.isnull().any(axis=1)])

df.dropna(inplace=True)





3. Find the most expensive car company name



max\_p = df.price.max()

df[df['price']==max\_p].company

df.loc[df['price'].idxmax(), 'company']

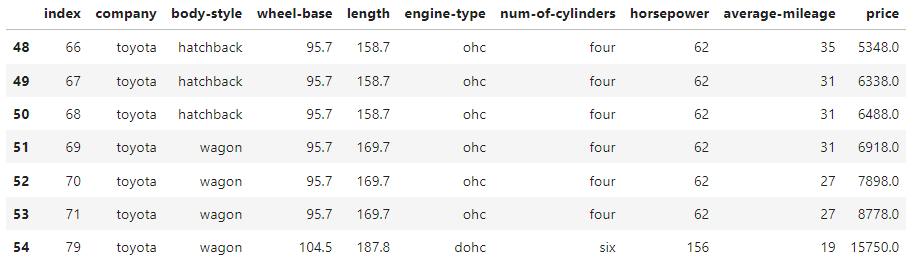




4. Print All Toyota Cars details



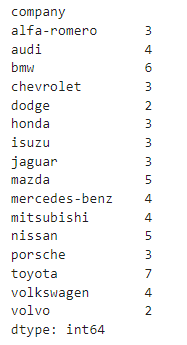
df[df['company']=='toyota']



5. Count total cars per company



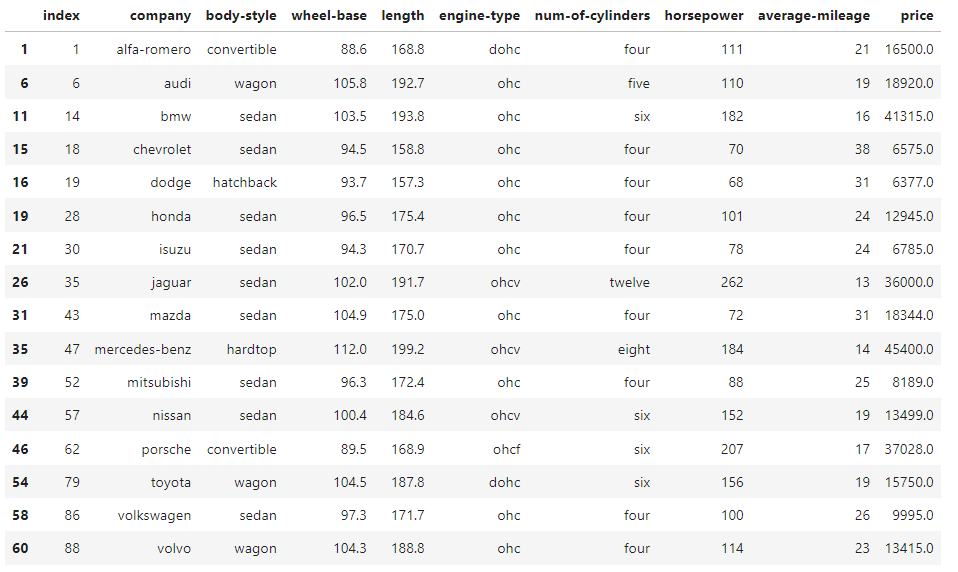
df.groupby('company').size()



6. Find each company’s Highest price car



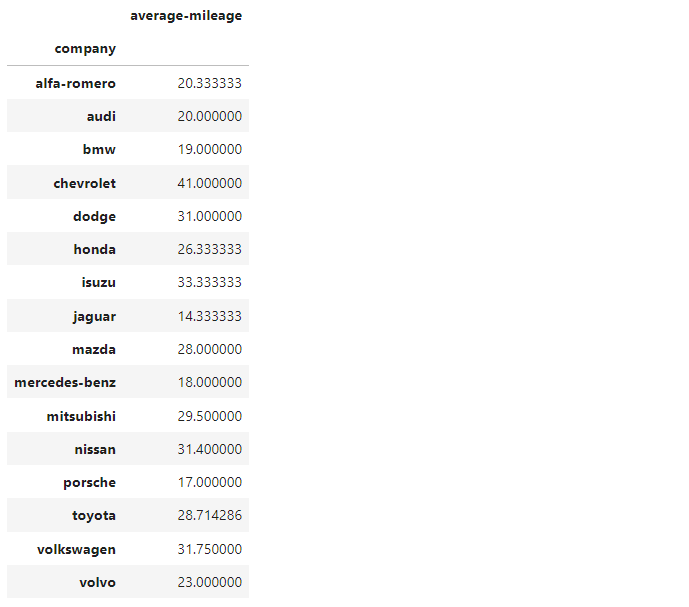
df.loc[df.groupby('company')['price'].idxmax()]



7. Find the average mileage of each car making company



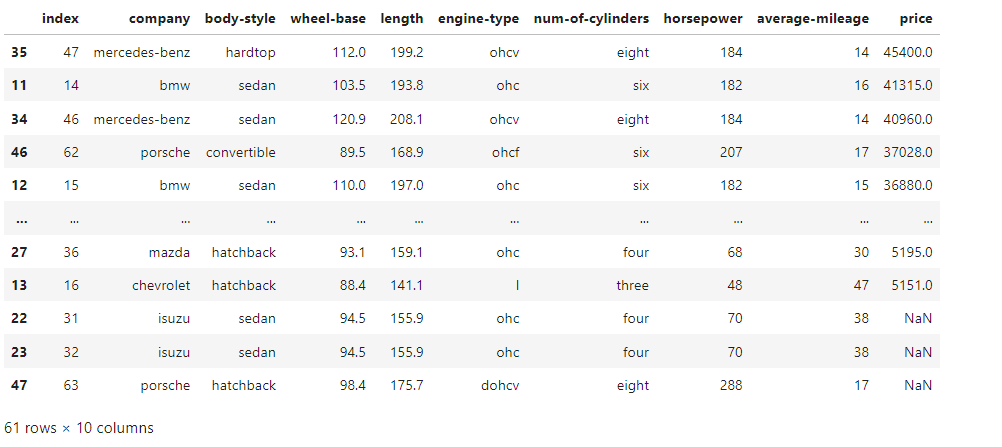
df.groupby('company')[['average-mileage']].mean()



8. Sort all cars by Price column



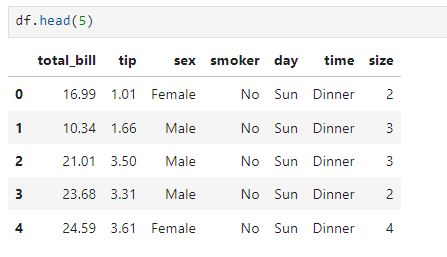
df.sort\_values('price', ascending=False)



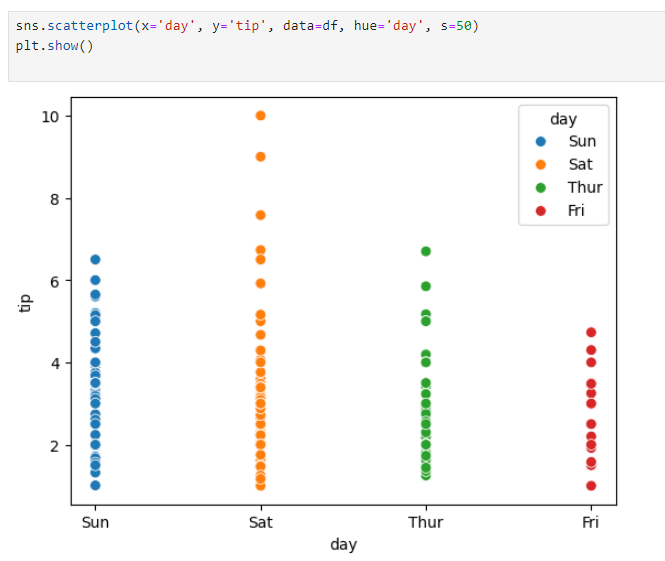
Problem Statement 15:

Using the given dataset perform the following :

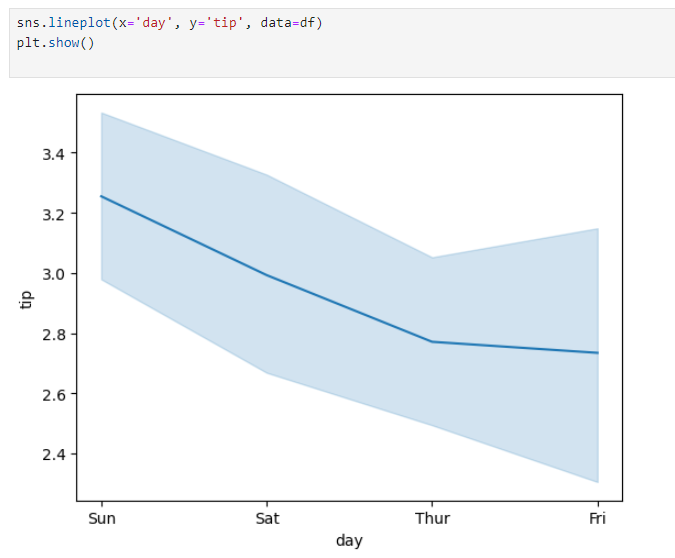
1. Display the top 5 rows of the dataset



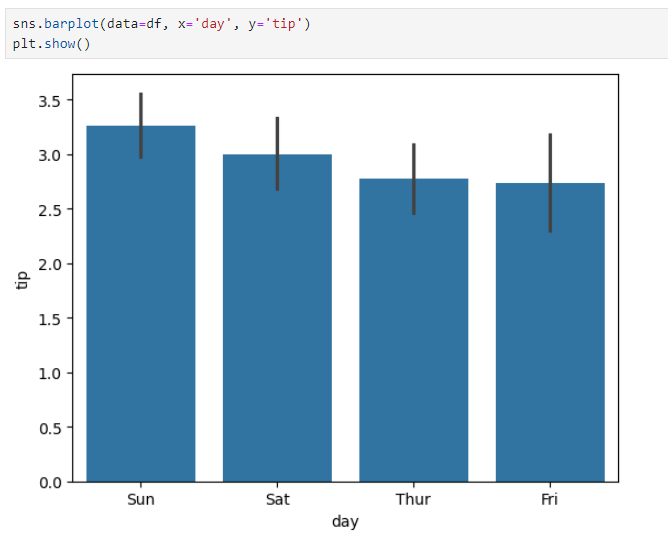
2. Create Scatter plot with day against the tip.



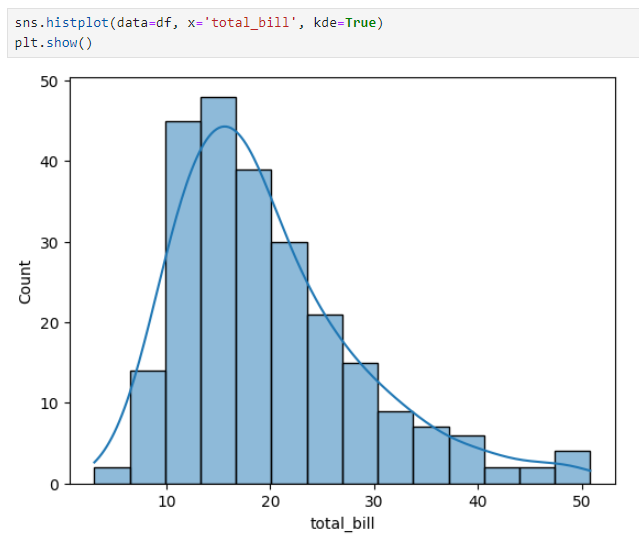
3. Create Line plot with day against tip.



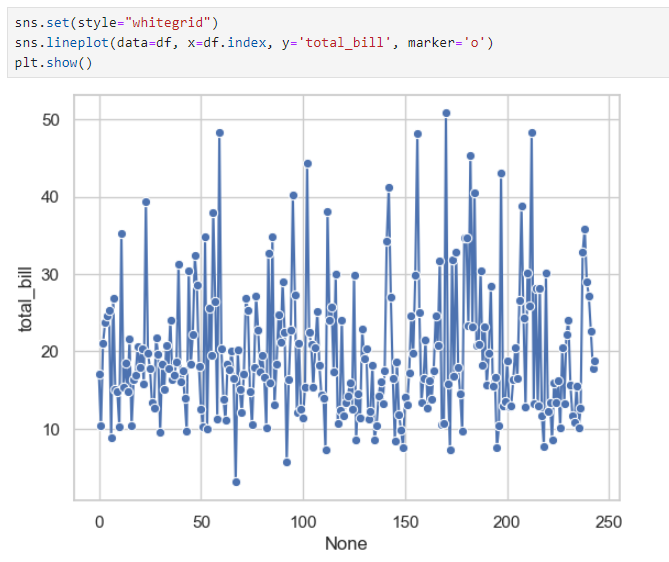
4. Create Bar chart with day against tip.



5. Create histogram of total\_bills



6. Draw line plot for total\_bills



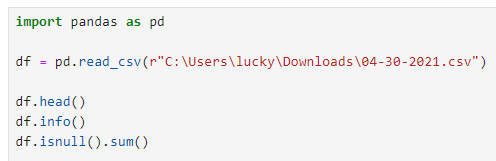
Problem Statement 16:

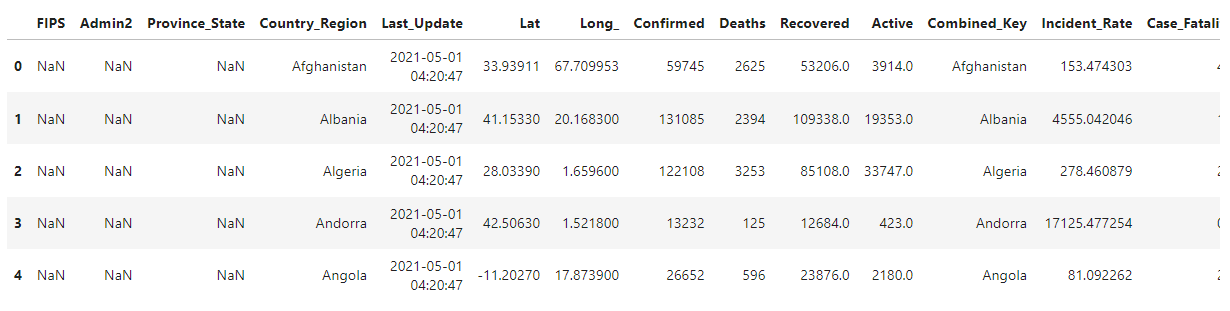
Case Study 1:

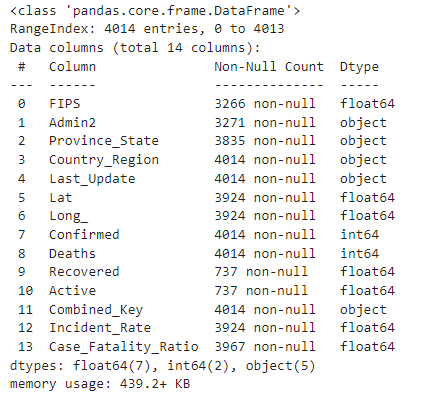
Coronavirus disease 2019 (COVID-19) is an infectious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The disease was first identified in 2019 in Wuhan, China, and has since spread globally, resulting in the 2019–20 coronavirus pandemic. Common symptoms include fever, cough and shortness of breath. Muscle pain, sputum production and sore throat are less common. The rate of deaths per number of diagnosed cases is on average 3.4%, ranging from 0.2% in those less than 20 to approximately 15% in those over 80 years old.

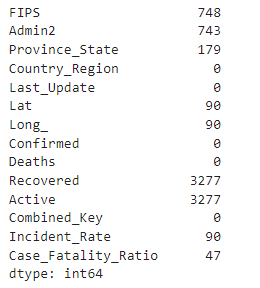
Link for dataset : https://github.com/CSSEGISandData/COVID-19/tree/master/csse\_covid\_19\_data/csse\_covid\_19\_daily\_reports

1. Write a Python program to display first 5 rows from COVID-19 dataset. Also print the dataset information and check the missing values.





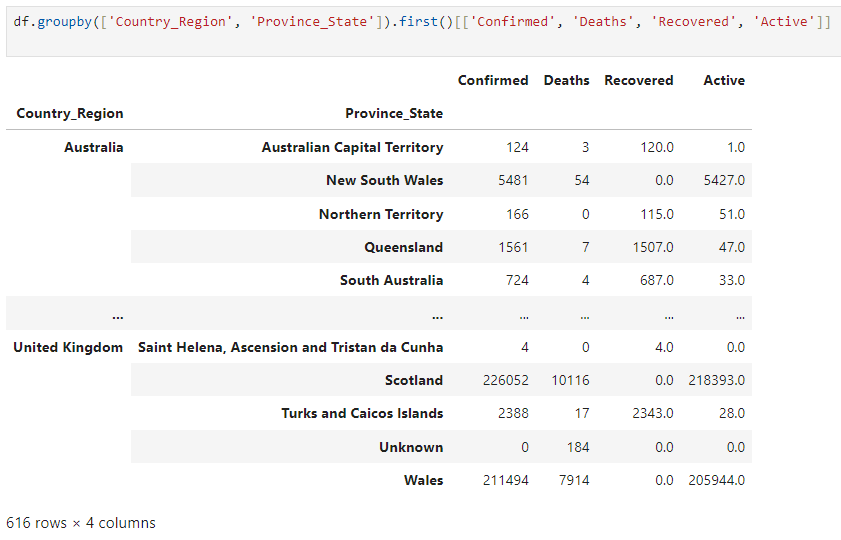




2. Get the latest number of confirmed, deaths, recovered and active cases of Novel Coronavirus (COVID-19) Country wise.



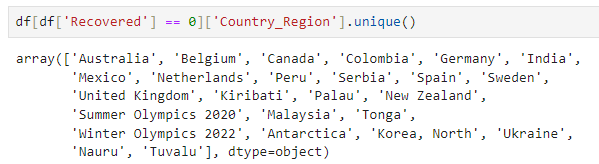
3. get the latest number of confirmed deaths and recovered people of Novel Coronavirus (COVID-19) cases Country/Region - Province/State wise.



4. get the top 10 countries data (Last Update, Country/Region, Confirmed, Deaths, Recovered) of Novel Coronavirus (COVID-19).



5. list countries with no cases of Novel Coronavirus (COVID-19) recovered.



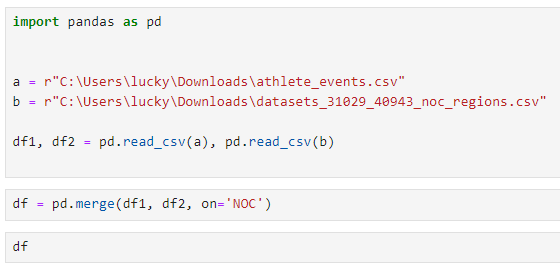
6. get the latest country wise deaths cases of Novel Coronavirus (COVID- 19).



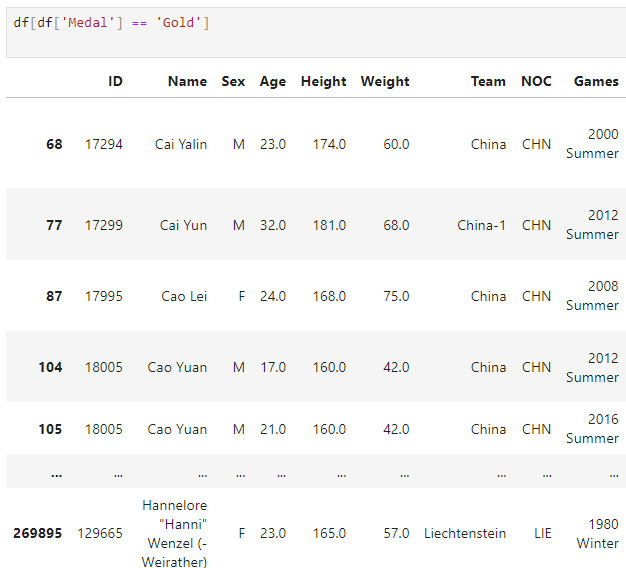
Problem Statement 17:

The contemporary Olympic Games, sometimes known as the Olympics, are major international sporting events that feature summer and winter sports contests in which thousands of participants from all over the world compete in a range of disciplines. With over 200 nations competing, the Olympic Games are regarded as the world's premier sporting event. We have two CSV files when dealing with Olympic data. One detailing the total sports-related expenses of all Olympic Games. Another has information on athletes from all years who competed with information.

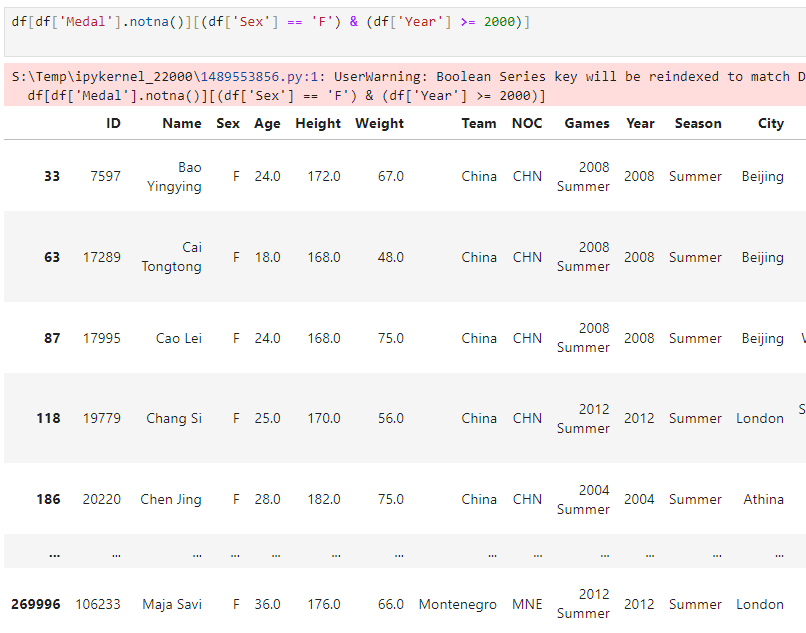
1. Write a python code to merge both datasets.



2. Create a new data frame including only gold medallists.



3. List medals we have only for women in the recent history of the Summer Games.



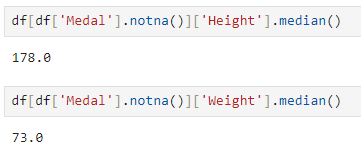
4. Display top 5 Gold medallist countries



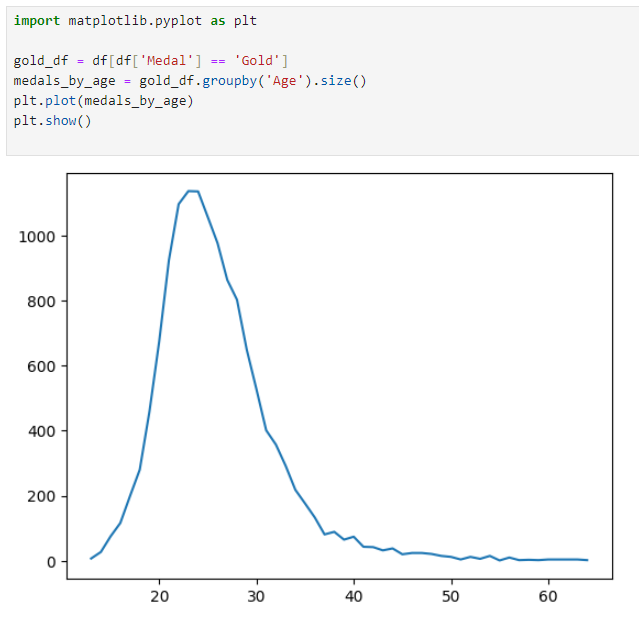
5. count the medals per discipline for country taken as input



6. What is the median height/weight of an Olympic medallist?

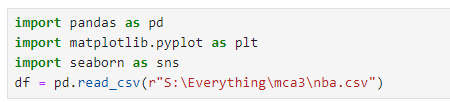


7. make a graph showing the number of gold medals in relation to age.



Problem Statement 18:

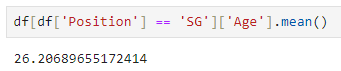
From the given data set perform the following :

Read Data:  
  


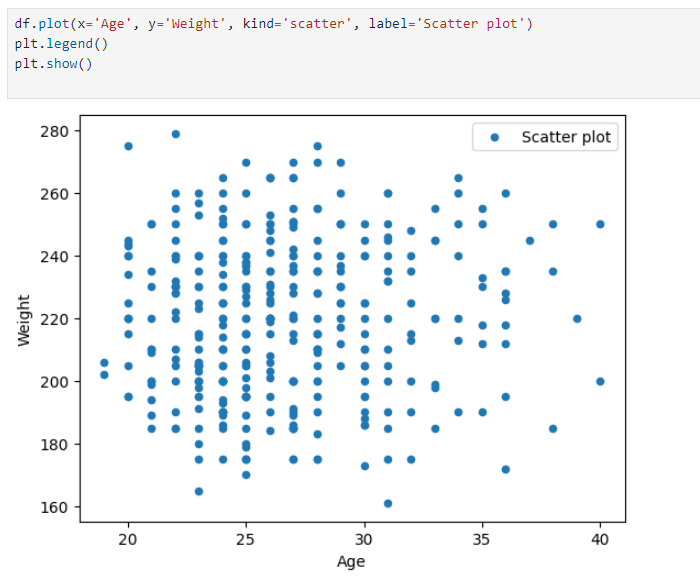
1. Remove all the missing values and NaN from the dataset.

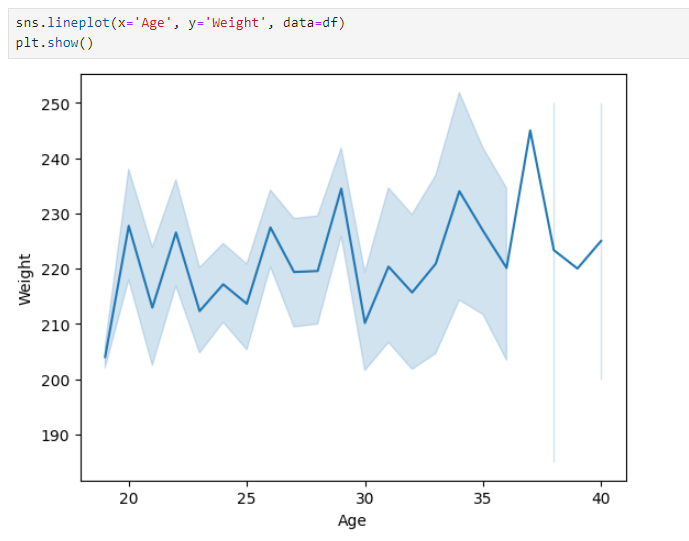


2. Find the average age for SG position.

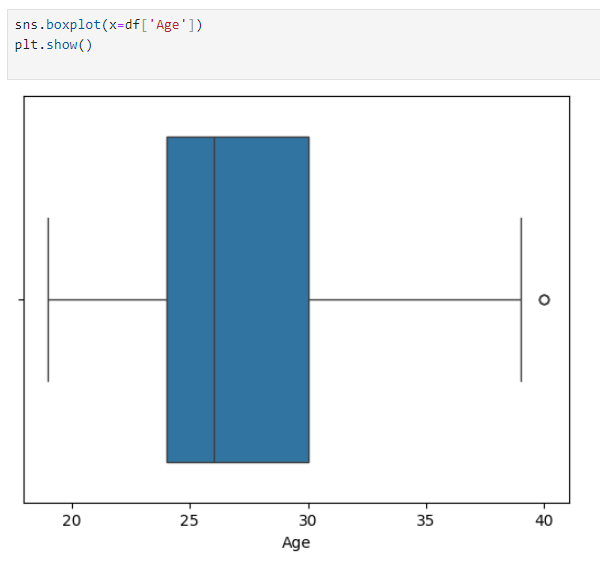


3. Plot the age vs weight graph using line and scatter plot

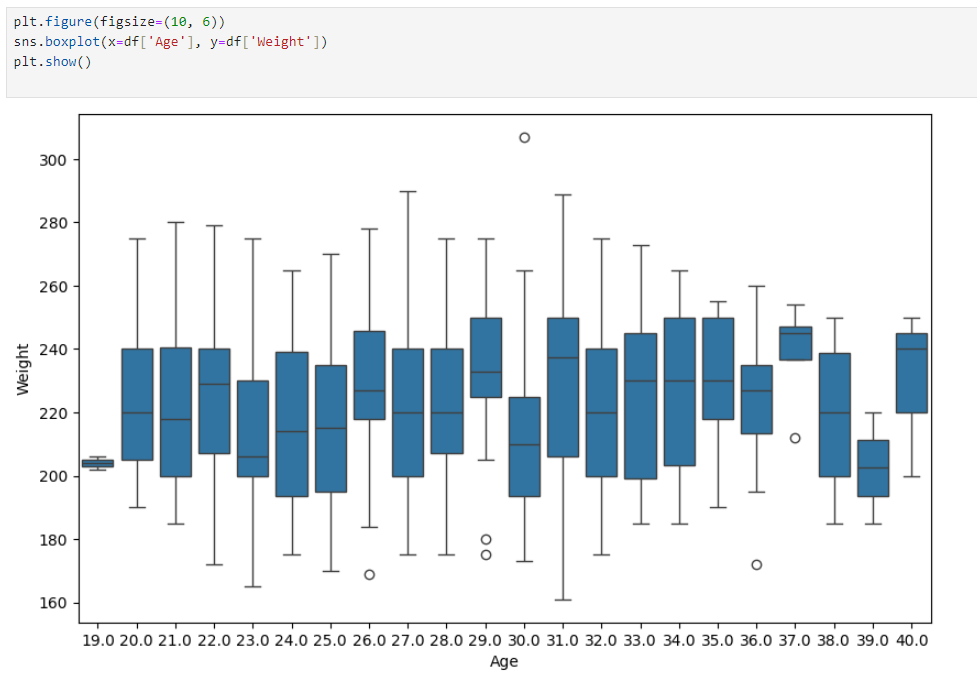




4. Display the box plot for age value using seaborn and pyplot



5. Display the box plot for age and weight value using seaborn and pyplot



Problem Statement 19:

From the given data set perform the following :

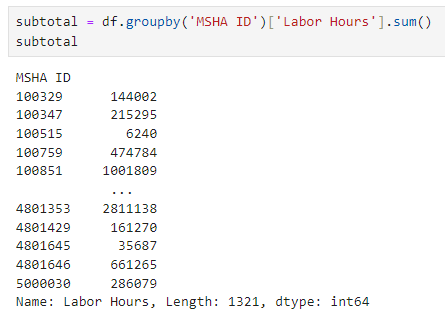
1. Write a Pandas program to find the sum, mean, max, min value of 'Production (short tons)' column of coalpublic2013.xlsx file.



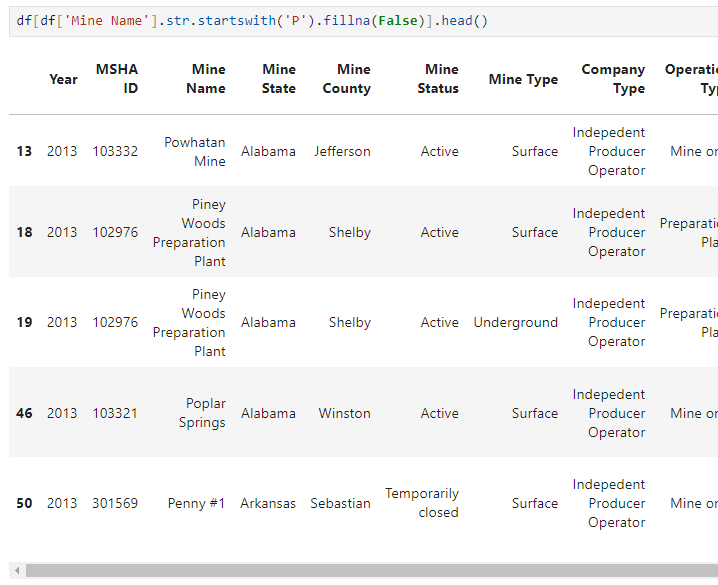
2. Write a Pandas program to import some excel data (coalpublic2013.xlsx ) skipping first twenty rows into a Pandas dataframe.



3. Write a Pandas program to create a subtotal of "Labor Hours" against MSHA ID from the given excel data



4. Write a Pandas program to import excel data (coalpublic2013.xlsx ) into a dataframe and find details where "Mine Name" starts with "P".



Problem Statement 20:

Uber Technologies, Inc., commonly known as Uber, is an American multinational ride-hailing company offering services that include peer-to-peer ridesharing, ride service hailing, food delivery (Uber Eats), and a micromobility system with electric bikes and scooters. The company is based in San Francisco and has operations in over 785 metropolitan areas worldwide. Its platforms can be accessed via its websites and mobile apps. There are more than 75 million active Uber riders accross the world. Uber is available in more than 80 countries worldwide. Uber has completed more than 5 billion rides till now. Over 3 million people drive for Uber. In the United States, Uber fulfills 40 million rides per month. The average Uber driver earns $364 per month.

https://www.kaggle.com/code/mohamed08/exploratory-data-analysis-for-uber-trips/input

This dataset contains 7 columns and 1156 entries.

1. Drop/Remove the null values from the dataset and represent it in a heatmap.



df = pd.read\_csv('My Uber Drives - 2016.csv')

df.dropna(inplace=True)

df['START\_DATE\*'] = pd.to\_datetime(df['START\_DATE\*'], format="%m/%d/%Y %H:%M").dt.strftime('%Y/%m/%d %H:%M')

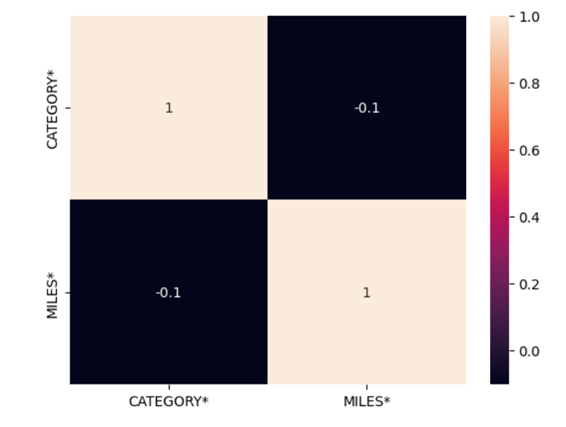
df['END\_DATE\*'] = pd.to\_datetime(df['END\_DATE\*'], format='%m/%d/%Y %H:%M').dt.strftime('%Y/%m/%d %H:%M')

df['START\_DATE\*'] = pd.to\_datetime(df['START\_DATE\*'], format='%Y/%m/%d %H:%M')

df['END\_DATE\*'] = pd.to\_datetime(df['END\_DATE\*'], format='%Y/%m/%d %H:%M')

df['CATEGORY\*'].replace({'Business' : 1, 'Personal' : 0}, inplace=True)

sns.heatmap(df.corr(numeric\_only=True), annot=True)



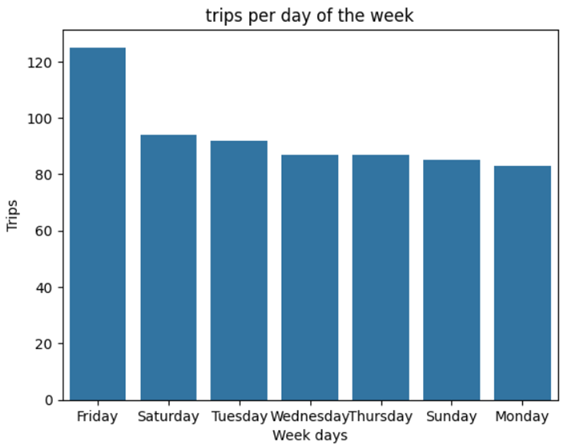
2. Plot the trips per day of the week.



df['month'] = df['START\_DATE\*'].dt.month\_name()

df['week'] = df['START\_DATE\*'].dt.day\_name()

sns.barplot(x=df['week'].unique(), y=df['week'].value\_counts()).set(xlabel='Week days', ylabel='Trips', title='trips per day of the week')

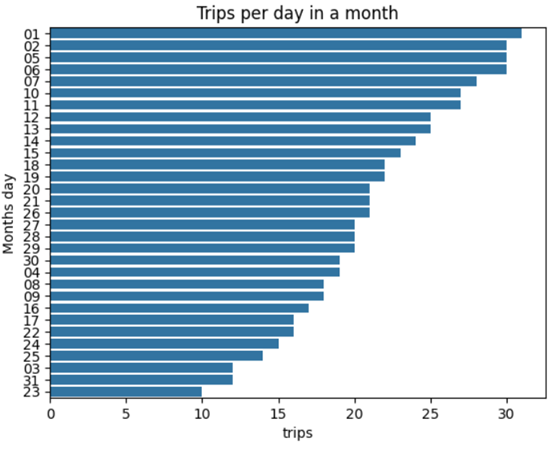


3. Plot the trips per day in a month.



df['day'] = df['START\_DATE\*'].dt.strftime('%d')

sns.barplot(y=df['day'].unique(), x=df['day'].value\_counts()).set(xlabel='trips', ylabel='Months day', title='Trips per day in a month')

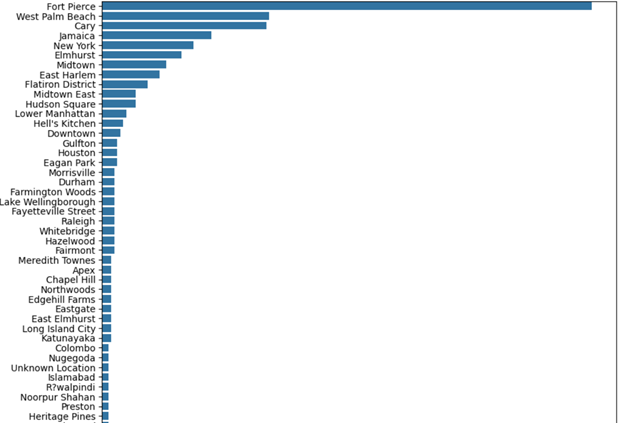


4. Plotting the starting point of the trip



plt.figure(figsize=(10,25))

sns.barplot(y=df['START\*'].unique(), x=df['START\*'].value\_counts())



5. Comparing the overall purpose with miles, hour, day of week, month, travel tim and speed.



df['travel(min)'] = (df['END\_DATE\*']-df['START\_DATE\*']).dt.total\_seconds()/60

df['speed(Mph)'] = df['MILES\*']/(df['travel(min)']/60)

df.head()

df['hour'] = df['START\_DATE\*'].dt.strftime('%H')

df.head()

df.groupby('PURPOSE\*')[['MILES\*', 'hour', 'week', 'month', 'travel(min)', 'speed(Mph)']].agg({'MILES\*' : 'mean', 'hour' : 'unique', 'week' : 'unique', 'month' : 'unique', 'travel(min)' : 'sum', 'speed(Mph)' : 'mean'}).reset\_index()

