University of Mumbai

DEPARTMENT OF COMPUTER SCIENCE

मुंबई विद्यापीठ
University of Mumbai
Re-accredited with A++ Grade
(CGPA 3.65) by NAAC (3rd Cycle 2021)

M.Sc. Data Science – Semester I

Essential Technologies for Data Science

JOURNAL

2024-2025

SUBMITTED BY

Vinamra Vijay Mishra

SEAT NO: 1300101

University of Mumbai

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the work entered in this journal was done in the **Department of Computer Science, University of Mumbai** by Mr./Ms.**Vinamra Vijay Mishra** Seat No. **1300101** for the course of **M.Sc. (Data Science) (NEP) Semester-I** during the academic year **2024-25** in a satisfactory manner.

_____ _____

Subject In-charge Head of Department

_____

External Examiner

Date:_____

INDEX

Page No:

Practical 1

Write a Python program to accept inputs from users and perform arithmetic operations

Practical 01

Code:

```
num1 = int(input("enter first number \n"))

num2 = int(input("enter second number \n"))


print("add:", num1+num2)

print("sub:", num1+num2)

print("div:", num1/num2)

print("mul:", num1*num2)

print("floordiv:", num1//num2)

print("exp:", num1**num2)

print("mod:",num1 % num2)
```

Output:

add: 110

sub: 110

div: 0.6923076923076923

mul: 2925

floordiv: 0

exp:28761623399675838629303884395468489341991082545373604947270415263266968297628
523032472003251314163208007812 5 mod: 45

Page No:

Practical 2

Write a program to accept the Shopping amount and apply discount on it

Practical 02

Code:

```
# Accept the shopping amount from the user
amount = float(input("Enter the shopping amount: "))

# Apply discount based on the amount entered
if amount >= 5000:
    discount = 0.50  # 50% discount for amounts 5000 or more
elif amount >= 2000:
    discount = 0.25  # 25% discount for amounts 2000 or more
elif amount >= 500:
    discount = 0.10  # 10% discount for amounts 500 or more
else:
    discount = 0  # No discount for amounts less than 500

# Calculate the final amount to be paid after applying the discount
final_amount = amount - (amount * discount)

# Display the final amount to the user
print("Amount to be paid after discount:", final_amount)
```

Output:

```
Amount to be paid after discount: 3332.5
```

Page No:

Practical 3

Demonstrate the use of data structures list, sets, dictionary

Practical 03

Code:

```
-> List Operations
my_list = [10, 20, 25, 30, 40, 50]
# Append
my_list.append(60)
print("After append:", my_list)
# Clear
my_list.clear()
print("After clear:", my_list)
# Copy
my_list = [10, 25, 20, 30, 40, 50]
copy_of_my_list = my_list.copy()
print("Copy of list:", copy_of_my_list)
# Pop
last_item = my_list.pop()
print("Popped item:", last_item)
print("After pop:", my_list)
# Remove
my_list.remove(25)
print("After remove:", my_list)
```

Output:

After append: [10, 20, 25, 30, 40, 50, 60]

After clear: []

Copy of list: [10, 25, 20, 30, 40, 50]

Popped item: 50

After pop: [10, 25, 20, 30, 40]

After remove: [10, 20, 30, 40]

Code:

```
-> Tuple Operations
Tuple1 = ('a', 'b', 'c', 'd')
Tuple2 = ('e', 'f', 'g', 'h')
# Repetition
repeated_Tuple1 = Tuple1 * 2
print("Repeated Tuple1:", repeated_Tuple1)
# Concatenation
concatenated_Tuples = Tuple1 + Tuple2
print("Concatenated Tuple1 and Tuple2:", concatenated_Tuples)
# Membership
is_a_in_Tuple1 = 'a' in Tuple1
print("Is 'a' in Tuple1?", is_a_in_Tuple1)
# Iteration
print("Iterating over Tuple1:")
for item in Tuple1:
    print(item)
```

Output:

Repeated Tuple1: ('a', 'b', 'c', 'd', 'a', 'b', 'c', 'd')

Concatenated Tuple1 and Tuple2: ('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h')

Is 'a' in Tuple1? True

Iterating over Tuple1:

a

b

c

d

Code:

-> Set Operations

```
Set1 = {1, 4, 2, 4, 5, 6, 3, 5, 4, 6, 77, 8, 7, 7, 876}

Set2 = {3, 432, 5, 6, 4, 6, 7, 6, 5, 6, 54, 567, 5}

# Union

union_Set1_Set2 = Set1 | Set2

print("Union of Set1 and Set2:", union_Set1_Set2)

# Intersection

intersection_Set1_Set2 = Set1 & Set2

print("Intersection of Set1 and Set2:", intersection_Set1_Set2)

# Difference

difference_Set1_Set2 = Set1 - Set2

print("Difference of Set1 and Set2 (Set1 - Set2):", difference_Set1_Set2)

# Symmetric difference

symmetric_difference_Set1_Set2 = Set1 ^ Set2

print("Symmetric Difference of Set1 and Set2:", symmetric_difference_Set1_Set2)
```

Output:

Union of Set1 and Set2: {1, 2, 3, 4, 5, 6, 7, 8, 876, 77, 432, 54, 567}

Intersection of Set1 and Set2: {3, 4, 5, 6, 7}

Difference of Set1 and Set2 (Set1 - Set2): {1, 2, 8, 876, 77}

Symmetric Difference of Set1 and Set2: {1, 2, 8, 876, 77, 432, 54, 567}

Code:

-> Dictionary Operations

```
dict = {'Name': Om Thakur', 'Age': 22}
# Length
length_given_dict = len(dict)
print("Length of Given Dictionary:", length_given_dict)
# String
string_given_dict = str(dict)
print("String representation of Given Dictionary:", string_given_dict)
dictionaries = {0:" Data",1: "GREAT", 2: "LEARNING",3:"Python",4:"Happy"}
# Copy
copied_dictionaries = dictionaries.copy()
print("Copied Given Dictionary:", copied_dictionaries)
# Key
keys = ['a', 'b', 'c']
new_dict = dict.fromkeys(keys, "Default Value")
print("New Dictionary from Keys:", new_dict)
# Dict Value
values_given = dictionaries.values()
print("Values in Given Dictionary:", list(values_given))
```

Output:

Length of Given Dictionary: 2

String representation of Given Dictionary: {'Name': Om Thakur, 'Age': 22}

Copied Given Dictionary: {0: ' Data', 1: 'GREAT', 2: 'LEARNING', 3: 'Python', 4: 'Happy'}

New Dictionary from Keys: {'a': 'Default Value', 'b': 'Default Value', 'c': 'Default Value'}

Values in Given Dictionary: [' Data', 'GREAT', 'LEARNING', 'Python', 'Happy']

Page No:

Practical 4

Demonstrate the use higher ordered function

Practical 04

Code:

```
# 1. Write a Python function to multiply all the numbers in a list.
def multiply_numbers(numbers):
    result = 1
    for num in numbers:
        result *= num
    return result

# Example usage
numbers_list = [7, 8, 9]
print("Product of all numbers:", multiply_numbers(numbers_list))
```

Output:

Product of all numbers: 504

Code:

```
#2. Write a Python function to reverse a string use the while loop also?
def reverse_string(text):
    reversed_text = ""
    i = len(text)
    while i:
        i -= 1
        reversed_text += text[i]
    return reversed_text
print(reverse_string("Hello Python"))
```

Output:

nohtyP olleH

Code:

 # 3. Write a function to add and subtract two variables

```
def add_and_subtract(x, y):
    return x + y, x - y
a, b = 10, 5
sum_result, diff_result = add_and_subtract(a, b)
print("Sum:", sum_result)
print("Difference:", diff_result)
```

Output:

Sum: 15

Difference: 5

 # 4. Write a function to check the number is divisible by 12

```
def is_divisible_by_12(number):
    if number % 12 == 0:
        return True
    else:
        return False
num = 36
if is_divisible_by_12(num):
    print(num, "is divisible by 12")
else:
    print(num, "is not divisible by 12")
```

Output:

36 is divisible by 12

 # 5. Write a function to calculate the number of days and weeks

```
def calculate_weeks_and_days(total_days):
    weeks = total_days // 7  # Calculate the number of weeks
    days = total_days % 7    # Calculate the remaining days
    return weeks, days

# Example usage
number_of_days = 15
weeks, days = calculate_weeks_and_days(number_of_days)
print("Weeks:", weeks)
print("Days:", days)
```

Output:

Weeks: 2

Days: 1

 # 6. Write a Python function to Find the 5!?

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result = result * i
    return result
print("5! =", factorial(5))
```

Output:

5! = 120

 # 7. Write a Python function Find the unique elements of the first list = [1,2,3,3,3,3,4,5,4,2,4,2,4,4,2,4,5,4,34,654,5,7,6,5,4,3,]?

```
def find_unique_elements(input_list):
    unique_elements = []
    for item in input_list:
        if item not in unique_elements:
            unique_elements.append(item)
    return unique_elements
first_list = [1, 2, 3, 3, 3, 3, 4, 5, 4, 2, 4, 2, 4, 4, 2, 4, 5, 4, 34, 654, 5, 7, 6, 5, 4, 3]
unique_values = find_unique_elements(first_list)
print("Unique elements:", unique_values)
```

Output:

Unique elements: [1, 2, 3, 4, 5, 34, 654, 7, 6]

 # 8. Required arguments: (the function simple_interest accepts three arguments and returns the simple interest accordingly)

```
def simple_interest(principal, rate, time):
    interest = (principal * rate * time) / 100
    return interest
p = 1000
r = 5
t = 3
interest = simple_interest(p, r, t)
print("Simple Interest:", interest)
```

Output:

Simple Interest: 150.0

 # 9. Keyword arguments:(Function is called with the name and message as the keyword arguments)

```
def greet(name, message):
    print("Hello,", name + "!")
    print(message)

# Example usage with keyword arguments
greet(name="Gagan", message="Happy Diwali!")
```

Output:

```
Hello, Gagan!
Happy Diwali!
```

 # 10. Default Arguments:()

```
def greet(name, message="Welcome to the program!"):
    print("Hello,", name + "!")
    print(message)

# Example usage
greet("Gagan")  # Calls the function with the default message
greet("Om", "Hope you have a great day!")  # Calls the function with a custom message
```

Output:

```
Hello, Gagan!
Welcome to the program!
Hello, Om!
Hope you have a great day!
```

 # 11. Write a lambda function to find the sum of two numbers

```
add = lambda x, y: x + y
result = add(5, 3)
print("Sum:", result)
```

Output:

Sum: 8

 # 12. Adding two lists using map lis1 = [12, 24, 36] and lis2 = [41, 54, 69]

```
# Given lists
lis1 = [12, 24, 36]
lis2 = [41, 54, 69]
result = list(map(lambda x, y: x + y, lis1, lis2))
print("Sum of two lists:", result)
```

Output:

Sum of two lists: [53, 78, 105]

 # 13. Filter the lis1 = [3,12, 24, 36,43,654,65432,2,654,455,43,543] which is not divided by 2?

```
lis1 = [3, 12, 24, 36, 43, 654, 65432, 2, 654, 455, 43, 543]
not_divisible_by_2 = list(filter(lambda x: x % 2 != 0, lis1))
print("Numbers not divisible by 2:", not_divisible_by_2)
```

Output:

Numbers not divisible by 2: [3, 43, 455, 43, 543]

Page No:

Practical 5

Demonstrate Univariate Analysis

Practical 05

Code:

```
# 1. Univarite Analysis using Pandas

# Import libraries and dataset

import pandas as pd, numpy as np, matplotlib.pyplot as plt, seaborn as sns

df_boston = pd.read_csv("Boston.csv", index_col=0)


# Univarite Function

def univariate_analysis(data, column):

  print("----------------------------------------")

  print(f"Univariate analysis of column {column}:")

  print("----------------------------------------")

  # Descriptive stats

  print(data[column].describe(include='all'))

  print(f"\nMissing values in column {column}: {data[column].isnull().sum()}")


  print("----------------------")

  # Check Skewness of the data

  print(f"Skewness of {column}: {data[column].skew()}")

  # Check Kurtosis of the data

  print(f"Kurtosis of {column}: {data[column].kurt()}")


  print("----------------------")

  print("IQR Range, Lower Fence, Upper Fence")

  # Get the IQR range from 75 percentile of data by minus with first quartile

  iqr_range = data[column].quantile(0.75) - data[column].quantile(0.25)

  # Store the lower and upper bound using the following formula

  lower_bound = data[column].quantile(0.25) - (1.5 * iqr_range)

  upper_bound = data[column].quantile(0.75) + (1.5 * iqr_range)

  print(f"IQR: {iqr_range}, Lower Bound: {lower_bound}, Upper Bound: {upper_bound}")


  # Z-score based outlier detection

  # Z-score formula to get the z-score value of all individual points

  z_scores = (data[column] - data[column].mean()) / data[column].std()
```

Output:

univariate_analysis(data=df_boston, column='crim')

Univariate analysis of column crim:

---

count 506.000000

mean 3.613524

std 8.601545

min 0.006320

25% 0.082045

50% 0.256510

75% 3.677083

max 88.976200

Name: crim, dtype: float64

Missing values in column crim: 0

Skewness of crim: 5.223148798243851

Kurtosis of crim: 37.13050912952203

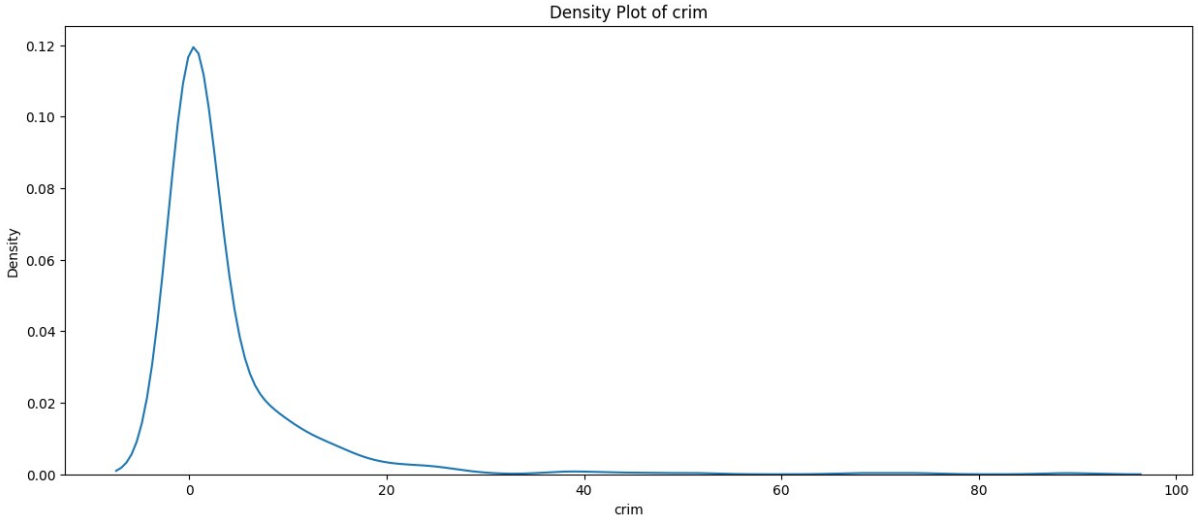
IQR Range, Lower Fence, Upper Fence

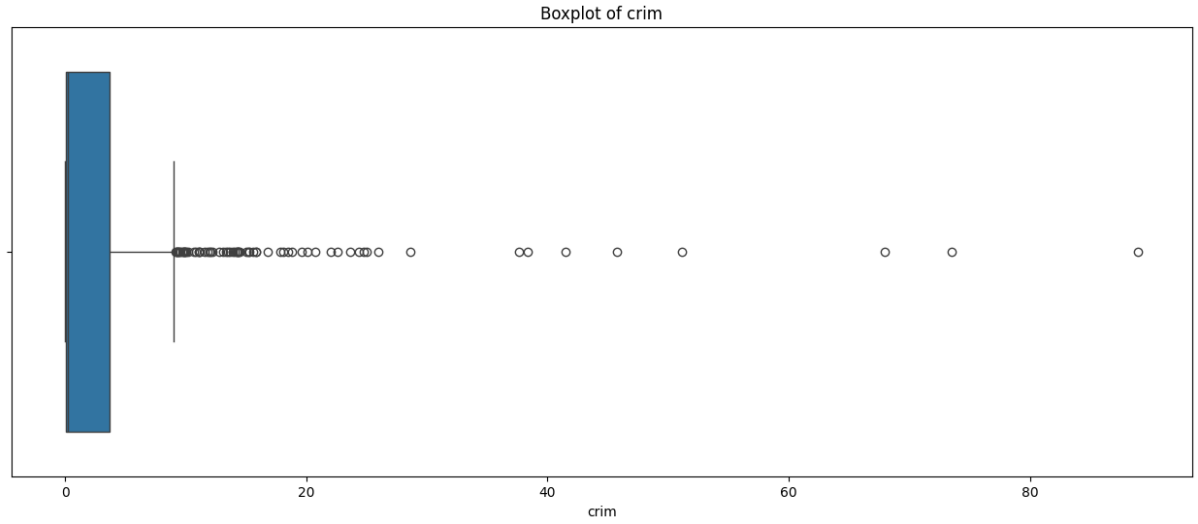
IQR: 3.5950375**, Lower Bound:** -5.31051125**, Upper Bound:** 9.06963875

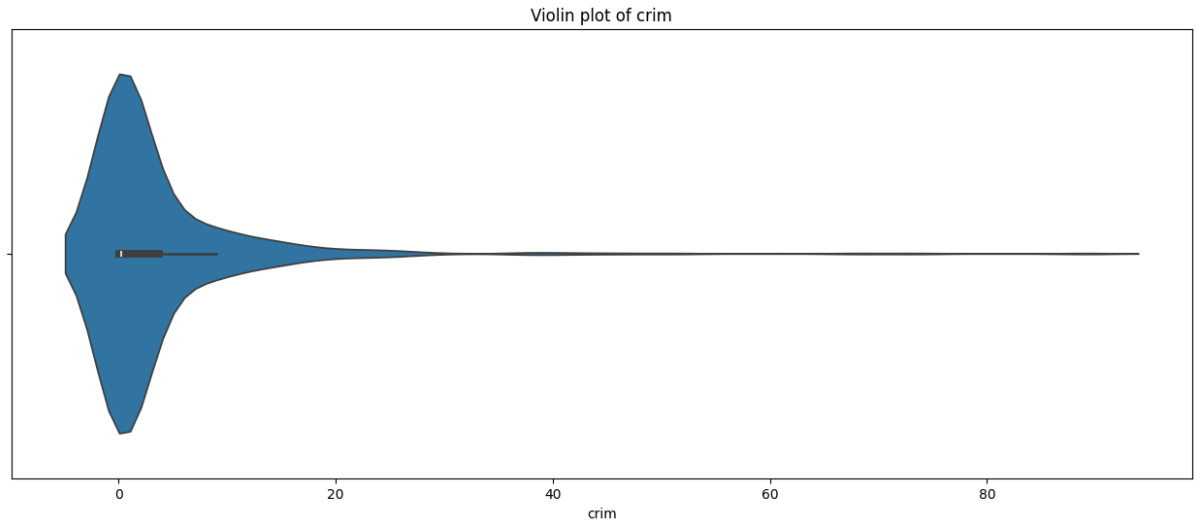
Number of outliers based on Z-score: 16

Range: 88.96988**, Variance:** 73.98657819906931**, Coefficient of Variation:** 238.03760979851853%

Univariate Analysis using Matplotlib & Seaborn







Page No:

Practical 6

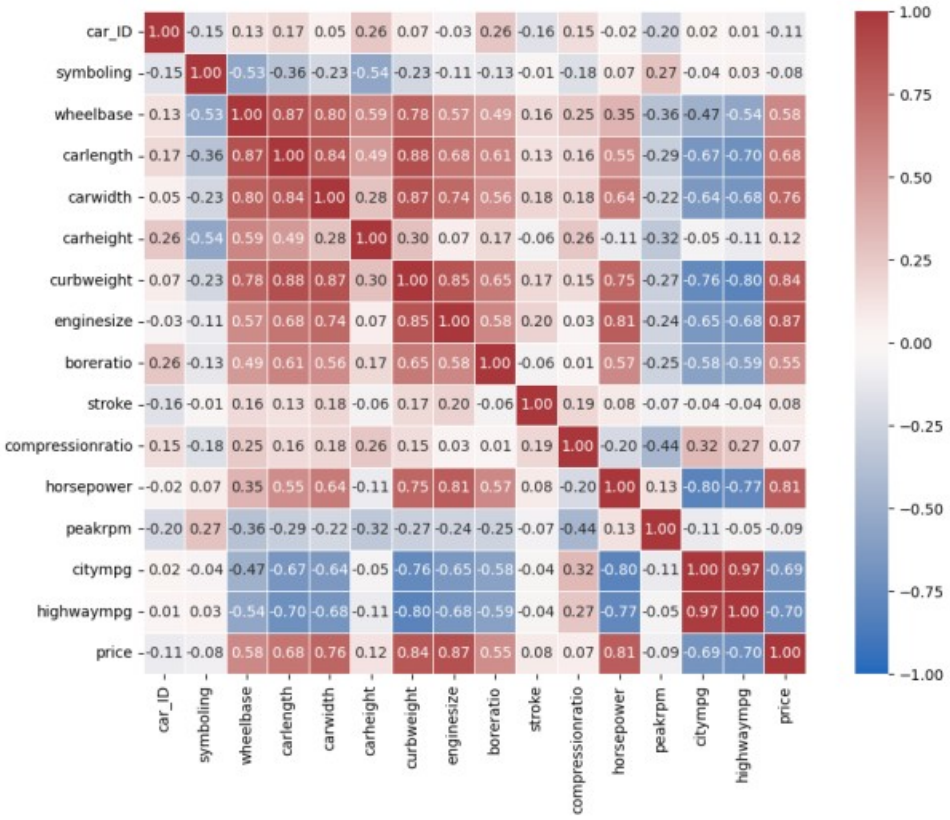
Demonstrate correlation analysis. Use heatmap for visualization. Write inferences.

Practical 06

Code:

```
# 1. Show Co-relation using HeatMap

# Import libraries and dataset

import pandas as pd, numpy as np, matplotlib.pyplot as plt, seaborn as sns

car_df = pd.read_csv(r"c:\Users\gagan\Downloads\CarPrice.csv")


# Create co-relation

filtered_df = car_df.select_dtypes(['int64', 'float64'])

correlation_matrix = filtered_df.corr()

correlation_matrix


# Create the heatmap

plt.figure(figsize = (12,8))

plt.rcParams.update({'font.size': 10})

sns.heatmap(correlation_matrix, cmap = 'vlag', vmin=-1, vmax=1, annot=True, fmt=".2f",
square=True, linewidths=.50)

plt.show()
```

Output:



Page No:

Practical 7

Import a csv or Excel dataset and demonstrate data wrangling

Practical 07

Code:

 # 1. Youtube Channel Analysis

```
import pandas as pd

df = pd.read_csv("c:\Users\gagan \Downloads\youtube.csv")
```

 # Assign Correct Datatypes

```
df["Subscriptions"]=df["Subscriptions"].str.extract('(\d+)')

df["Subscriptions"]=pd.to_numeric(df["Subscriptions"])


df["Uploads"]=df["Uploads"].str.replace(',',")

df["Uploads"]=pd.to_numeric(df["Uploads"])


df['Views'] = df['Views'].str.replace(",", "", regex=True).replace("--", "", regex=True).replace("", None).astype("Int64")
```

 # Print datatypes

Rank	object
Grade	object
Ch_name	object
Uploads	float64
Subscriptions	int64
Views	Int64

 # Question 01

```
df[(df['Ch_name'] == 'T-Series') | (df['Ch_name'] == 'SAB TV') | (df['Ch_name'] == 'Zee TV')]
```

	Rank	**Grade**	**Ch_name**	**Uploads**	**Subscriptions**	**Views**
0	1st	A++	T-Series	14297.0	135	104724369854
7	8th	A++	Zee TV	98621.0	43	41544259461
10	11th	A++	SAB TV	23812.0	29	25597492503

 # Question 02

	Rank	**Grade**	**Ch_name**	**Uploads**	**Subscriptions**	**Views**
1	2nd	A++	Cocomelon - Nursery	517.0	78	57054290512

			Rhymes			
2	3rd	A++	✤ Kids Diana Show	691.0	50	24157678368
3	4th	A++	Like Nastya	400.0	52	30591257306
4	5th	A++	SET India	37017.0	69	52149505781
18	19th	A+	WWE	47028.0	57	41299497565
25	26th	A+	Zee Music Company	4693.0	53	25075527967
75	76th	A	Canal KondZilla	1387.0	56	29414628881
100	101st	A	Go Turkey	405.0	52	550553577
218	219th	A	FlyntofRWBY	64.0	94	104182735
220	221st	A	5-Minute Crafts	4011.0	65	17487779993
240	241st	A	Walls Thailand	117.0	55	102481881
266	267th	A	PhonePe	160.0	90	1513957671
390	391st	A	Dude Perfect	229.0	50	10311702172
400	401st	A	뮤지컬웨딩 MusicalWedding	3055.0	56	73795144
488	489th	A	lester villegas	34.0	72	63700113

 # Question 03

df.groupby(['Ch_name','Subscriptions', 'Views']).agg(Count=('Subscriptions', 'count')).sort_values(by="Subscriptions", ascending=False).head(10)

	Rank	Grade	Ch_name	Uploads	Subscriptions	Views
176	177th	A	ببجي بالعربي	113.0	991	141803352
401	402nd	A	Телеканал Звезда	2663.0	990	181693400
308	309th	A	3D Music India	245.0	960	85756622
339	340th	A	VSRAP	147.0	947	84033455
257	258th	A	Siyah Giyen Genç	599.0	947	212758305
61	62nd	A	MUSIC BANGLA TV	328.0	943	227019598
149	150th	A	SO LY DA	27.0	908	227038807
265	266th	A	BillieEilishVEVO	56.0	876	4369175365
38	39th	A+	Odia E News	1466.0	838	265026133
437	438th	A	tvN D CLASSIC	9343.0	825	1396808540

 # Question 04

	Rank	Grade	Ch_name	Uploads	Subscriptions	Views
97	98th	A	LETRAS RD	2.0	250	180383784
272	273rd	A	Dynoro	6.0	249	125983776
321	322nd	A	NS Tv Show	6.0	445	774863
142	143rd	A	DuckDuck Kids TV	8.0	771	72747787
108	109th	A	theplatformfilmmake production	9.0	221	147483344

 # Question 05

df.groupby('Grade').agg(Average_subscriber=('Subscriptions', "mean"))

Grade	Average_subscriber
A	75.40459770114943

A+	40.06122448979592
A++	48.81818181818182

 # Question 06

df[df['Grade'] == 'A++'].sort_values(by='Views', ascending=False).head(5)

	Rank	**Grade**	**Ch_name**	**Uploads**	**Subscriptions**	**Views**
0	1st	A++	T-Series	14297.0	135	104724369854
1	2nd	A++	Cocomelon	517.0	78	57054290512
4	5th	A++	SET India	37017.0	69	52149505781
7	8th	A++	Zee TV	98621.0	43	41544259461
6	7th	A++	Movieclips	35226.0	36	35055807085

Page No:

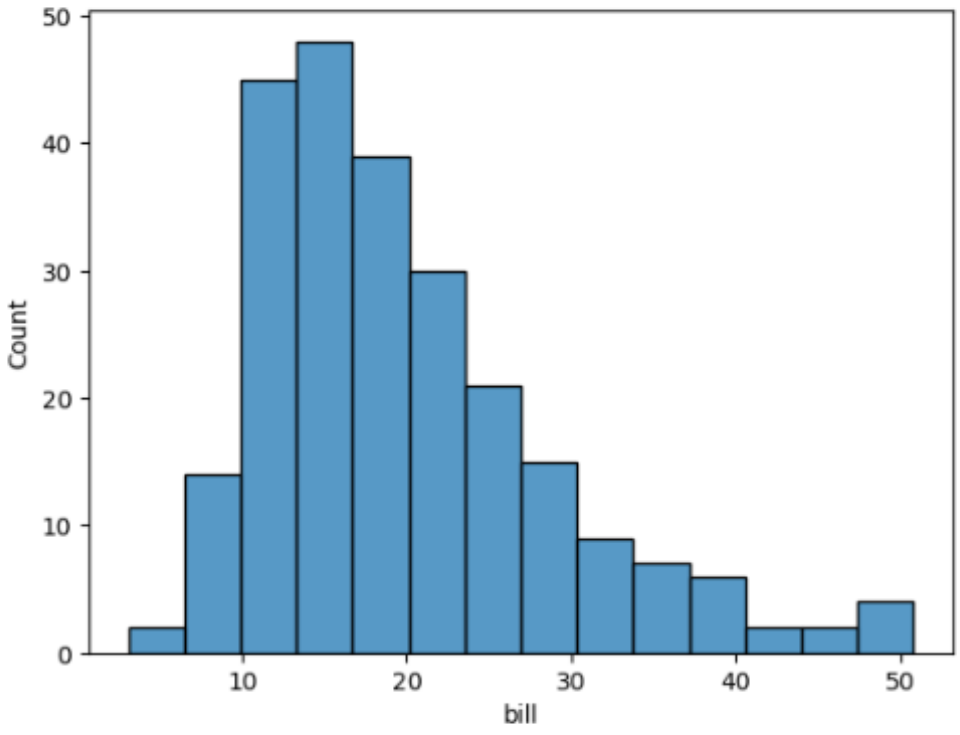
Practical 8

Perform univariate, bivariate and multivariate analysis using visualization techniques in Python

Practical 08

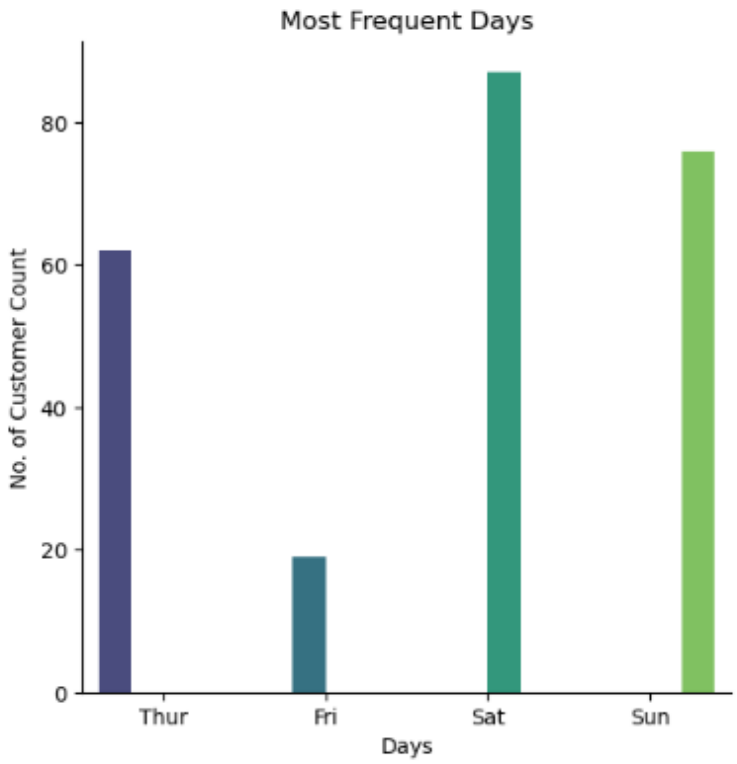
Code:

```
# Import Libraries and datasets

import pandas as pd, matplotlib.pyplot as plt, seaborn as sns

df_tip = pd.read_csv("c:\Users\gagan\Downloads\tips2.csv")


# Gender

df_tip['gender'].replace({0: 'Male', 1: 'Female'}).value_counts()


# Bill

sns.histplot(data=df_tip, x="bill")

plt.show()
```

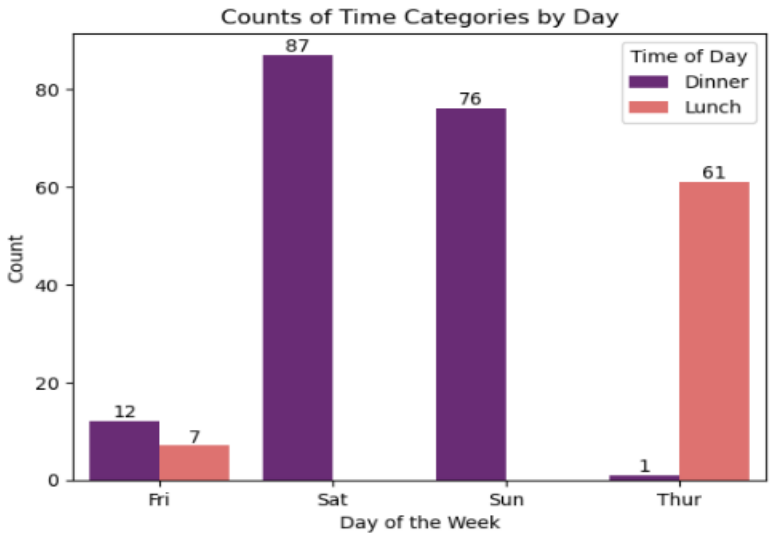


 # Tip Count

```
sns.catplot(data=df_tip['day'].value_counts().reindex(['Thur', 'Fri', 'Sat', 'Sun']).reset_index(), x='day', y='count', kind="bar", hue="day", palette="viridis")

plt.title("Most Frequent Days")

plt.xlabel("Days")

plt.ylabel("No. of Customer Count")

plt.show()
```



```
# Count occurrences of time for each day
grouped_counts = df_tip.groupby(['day', 'time']).size().reset_index(name='count').reindex()
# Create a grouped bar plot
ax = sns.barplot(data=grouped_counts, x='day', y='count', hue='time', palette='magma')
for row in ax.containers:
    ax.bar_label(row)
plt.title("Counts of Time Categories by Day")
plt.xlabel("Day of the Week")
plt.ylabel("Count")
plt.legend(title="Time of Day")
plt.show()
```



Tip range

```
df_tip['tip_range'] = pd.cut(x=df_tip['tip'], bins=[0, 2, 4, 6, 8, 10], labels=['0-2', '2-4', '4-6', '6-8', '8-10'], include_lowest=True)
df_tip['tip_range'].value_counts()
```

tip_range	count
2-4	125
0-2	78
4-6	34
6-8	5
8-10	2

Page No:

Practical 9

Perform Group by operations and sorting techniques

Practical 09

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

df = pd.read_csv("Bank.csv")
df = df.drop(df.columns[0], axis=1)

# Create Bin
bins = [18, 28, 38, 48, 58, 68, 78, 88]
labels = ['18-28', '28-38', '38-48', '48-58', '58-68', '68-78', '78-87']
df['age_group'] = pd.cut(df['age'], bins=bins, labels=labels, include_lowest=True)
```

age_group	age
18-28	385
28-38	1775
38-48	1221
48-58	895
58-68	178
68-78	49
78-87	18

```
df.groupby('job').agg(Total_Job=('job', 'count')).head()
```

job	Total_Job
admin.	478
blue-collar	946
entrepreneur	168
housemaid	112
management	969

```
df.groupby(['age_group','job']).agg({
    'job':'count', 'marital':'count'}).sort_index(ascending=True)
```

age_group	job	job	marital
18-28	admin.	53	53
18-28	blue-collar	78	78

18-28	entrepreneur	7	7
18-28	housemaid	4	4
18-28	management	48	48

df[df['job'] == 'unemployed'].groupby('marital').agg(Total_No=('job', 'count'))

marital	**Total_No**
divorced	22
married	75
single	31

df.groupby(['marital', 'job']).agg(Total_No=('job', 'count')).sort_values(ascending=False, by='Total_No').head()

marital	**job**	**Total_No**
married	blue-collar	693
married	management	557
married	technician	411
single	management	293
single	technician	268

df.groupby(["education", "job"]).agg(Count=("education", "count")).sort_values(by="Count", ascending=False).head(5)

education	**job**	**Count**
tertiary	management	787
secondary	blue-collar	524
secondary	technician	520
secondary	admin.	393
primary	blue-collar	369

df.groupby(["job", "default"]).agg(total_no=('age', 'count')).sort_values(by=['default', 'total_no'], ascending=False).head()

job	**default**	**total_no**
technician	yes	15
blue-collar	yes	14
management	yes	14
entrepreneur	yes	7
services	yes	7

Page No:

Practical 10

Demonstrate Hypothesis testing, and ANOVA using a dataset [use Python]

Practical 10

Code:

```
import pandas as pd, seaborn as sns

from scipy.stats import ttest_ind, ttest_1samp

df = pd.read_csv(r" c:\Users\gagan \Downloads\crop_rec.csv")


# A. Two-Sample T-Test for Temperature and Humidity

rice = df[df['label'] == 'rice']

jute = df[df['label'] == 'jute']

banana = df[df['label'] == 'banana']

grapes = df[df['label'] == 'grapes']


# 1. Rice and Jute

t_temp_rice_jute, p_temp_rice_jute = ttest_ind(rice['temperature'], jute['temperature'],
equal_var=False)

t_humid_rice_jute, p_humid_rice_jute = ttest_ind(rice['humidity'], jute['humidity'], equal_var=False)


# 2. Banana and Grapes

t_temp_banana_grapes, p_temp_banana_grapes = ttest_ind(banana['temperature'],
grapes['temperature'], equal_var=False)

t_humid_banana_grapes, p_humid_banana_grapes = ttest_ind(banana['humidity'], grapes['humidity'],
equal_var=False)
```

Output

```
# B. One-Sample T-Test for pH of Mango

mango = df[df['label'] == 'mango']

t_ph_mango, p_ph_mango = ttest_1samp(mango['ph'], 7.5)

# Display Results

print("A. Two-Sample T-Test Results")

print("Rice vs Jute (Temperature): t-stat =", t_temp_rice_jute, "p-value =", p_temp_rice_jute)

print("Rice vs Jute (Humidity): t-stat =", t_humid_rice_jute, "p-value =", p_humid_rice_jute)
```

```
print("Banana vs Grapes (Temperature): t-stat =", t_temp_banana_grapes, "p-value =", p_temp_banana_grapes)

print("Banana vs Grapes (Humidity): t-stat =", t_humid_banana_grapes, "p-value =", p_humid_banana_grapes)
```

Output

```
A. Two-Sample T-Test Results

Rice vs Jute (Temperature): t-stat = -5.3962205928971 p-value = 2.423334234277455e-07

Rice vs Jute (Humidity): t-stat = 4.6293767138756845 p-value = 9.9305191846967e-06

Banana vs Grapes (Temperature): t-stat = 3.583542334338258 p-value = 0.000519057029018083

Banana vs Grapes (Humidity): t-stat = -4.986508831451448 p-value = 1.8856468716438267e-06
```

```
print("\nB. One-Sample T-Test for Mango pH")

print("Mango (pH): t-stat =", t_ph_mango, "p-value =", p_ph_mango)
```

Output

```
B. One-Sample T-Test for Mango pH

Mango (pH): t-stat = -24.637475315453745 p-value = 5.047616652632044e-44
```

Declare the Null Hypothesis

```
#1. Rice vs Jute (Temperature and Humidity)

print("Hypothesis Testing for Rice vs Jute (Temperature):")

print("Null Hypothesis (H0): The mean temperature of rice and jute are the same.")

print("Alternative Hypothesis (H1): The mean temperature of rice and jute are different.")

print("\nHypothesis Testing for Rice vs Jute (Humidity):")

print("Null Hypothesis (H0): The mean humidity of rice and jute are the same.")

print("Alternative Hypothesis (H1): The mean humidity of rice and jute are different.")
```

Output

Hypothesis Testing for Rice vs Jute (Temperature):

Null Hypothesis (H0): The mean temperature of rice and jute are the same.

Alternative Hypothesis (H1): The mean temperature of rice and jute are different.

Hypothesis Testing for Rice vs Jute (Humidity):

Null Hypothesis (H0): The mean humidity of rice and jute are the same.

Alternative Hypothesis (H1): The mean humidity of rice and jute are different.

#2. Banana vs Grapes (Temperature and Humidity)

```
print("Hypothesis Testing for Banana vs Grapes (Temperature):")
print("Null Hypothesis (H0): The mean temperature of banana and grapes are the same.")
print("Alternative Hypothesis (H1): The mean temperature of banana and grapes are different.")
print("\nHypothesis Testing for Banana vs Grapes (Humidity):")
print("Null Hypothesis (H0): The mean humidity of banana and grapes are the same.")
print("Alternative Hypothesis (H1): The mean humidity of banana and grapes are different.")
```

Output

Hypothesis Testing for Banana vs Grapes (Temperature):

Null Hypothesis (H0): The mean temperature of banana and grapes are the same.

Alternative Hypothesis (H1): The mean temperature of banana and grapes are different.

Hypothesis Testing for Banana vs Grapes (Humidity):

Null Hypothesis (H0): The mean humidity of banana and grapes are the same.

Alternative Hypothesis (H1): The mean humidity of banana and grapes are different.

 # Mango (pH)

```
print("\nHypothesis Testing for Mango pH:")
print("Null Hypothesis (H0): The mean pH of mango is 7.5.")
print("Alternative Hypothesis (H1): The mean pH of mango is not equal to 7.5.")
```

Output

Hypothesis Testing for Mango pH:

Null Hypothesis (H0): The mean pH of mango is 7.5.

Alternative Hypothesis (H1): The mean pH of mango is not equal to 7.5.

Inferences

#1) Inference based on Temperature

```
if p_temp_rice_jute < 0.05:
    print("Inference: Reject the Null Hypothesis. The mean temperature of rice and jute are significantly different.")
else:
    print("Inference: Fail to reject the Null Hypothesis. There is no significant difference in temperature between rice and jute.")
```

Output

Inference: Reject the Null Hypothesis. The mean temperature of rice and jute are significantly different.

#2) Inference based on Humidity

```
if p_humid_rice_jute < 0.05:
    print("Inference: Reject the Null Hypothesis. The mean humidity of rice and jute are significantly different.")
else:
    print("Inference: Fail to reject the Null Hypothesis. There is no significant difference in humidity between rice and jute.")
```

Output

Inference: Reject the Null Hypothesis. The mean humidity of rice and jute are significantly different.

#2) Banana vs Grapes

##1) Inference based on Temperature

```
if p_temp_banana_grapes < 0.05:
    print("Inference: Reject the Null Hypothesis. The mean temperature of banana and grapes are significantly different.")
else:
```

print("Inference: Fail to reject the Null Hypothesis. There is no significant difference in temperature between banana and grapes.")

Output

Inference: Reject the Null Hypothesis. The mean temperature of banana and grapes are significantly different.

#2) Inference based on Humidity

if p_humid_banana_grapes < 0.05:

print("Inference: Reject the null hypothesis. The humidity requirements of banana and grapes are significantly different.")

else:

print("Inference: Fail to reject the null hypothesis. The humidity requirements of banana and grapes are not significantly different.")

Output

Inference: Reject the null hypothesis. The humidity requirements of banana and grapes are significantly different.

 # Inference based on p-value

if p_ph_mango < 0.05:

print("Inference: Reject the Null Hypothesis. The pH of mango is significantly different from 7.5.")

else:

print("Inference: Fail to reject the Null Hypothesis. The pH of mango is not significantly different from 7.5.")

Output

Inference: Reject the Null Hypothesis. The pH of mango is significantly different from 7.5.