DATA SCIENCE RECORD

1. **Write programs to parse Text, CSV, JSON files/documents & extract relevant data. After retrieving data check any anomalies in the data, missing values etc.**

TEXT:

```
file_path = r"C:\Users\pras\Desktop\TXT.txt"

try:
  with open(file_path, 'r') as file:
    contents = file.read()
    print(contents)
except FileNotFoundError:
  print("File does not exist")
except IOError:
  print("Unable to open file")
```

CSV:

```
import csv
file_path=r"C:\Users\pras\Desktop\student.csv"
data=[]
with open(file_path,newline='')as file:
  csv_reader=csv.reader(file)
  for row in csv_reader:
    data.append(row)
  for row in data:
    print(row)
```

JSON:

```
import json
file_path = "C:/users/pras/Desktop/stud.json"

with open(file_path, 'r') as file:
  data = json.load(file)
print(data)
```

2. **Write programs to Write and Read Binary files in python**

Writing

```
content = b'\x48\x65\x6c\x6c\x6f'  # binary data -> "Hello"
file_path = r"C:/Users/pras/Desktop/binrite.bin"

with open(file_path, 'wb') as file:
  file.write(content)
```

Reading

```
file_path=r"C:/Users/pras/Desktop/binarite.bin"
try:
  with open(file_path)as file:
    contents=file.read()
    print(contents)
except FileNotFoundError:
  print("file foes not exist")
except IOError:
  print("Unable to open file")
```

3. **Write programs for searching splitting and replacing strings based on pattern matching using regular expressions**

Searching

```
Import re
Text=”Data Engg With Python”
x=re.search(‘/s’,text)
print(x)
```

Splitting

```
import re
text='Data Engg With Python'
x=re.split('/s',text)
print(x)
```

Replacing

```
import re
text='Data Engg With Python'
x=re.sub('/s','5',text)
print(x)
```

4. **Design relational database for a small application and populate the database using sql to the crude (create, read, update and delete operations)**

```
mysql> USE company;
Database changed

mysql> CREATE TABLE student (studentname TINYTEXT, Marks FLOAT,
joining_date DATE);
Query OK, 0 rows affected

mysql> INSERT INTO student VALUES ('saritha', 400, '2001-05-22');
Query OK, 1 row affected

mysql> INSERT INTO student VALUES ('Ramesh', 500, '2001-06-21');
Query OK, 1 row affected
```

5. **Write programs to create numpy arrays of different slopes and from different sources, reshape and slice arrays, add array indexes and apply arithmetic logic and aggregation functions to same on all array elements**

Creating Numpy Arrays:

```
import numpy as np

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

arr_from_tup = np.array((1, 2, 3, 4, 5))
print(arr_from_tup)

zeros_array = np.zeros((3, 4))
print(zeros_array)
```

```
ones_array = np.ones((3, 4))
print(ones_array)

identity_matrix = np.eye(2)
print(identity_matrix)

fill_matrix = np.diag([1, 2, 3])
print(fill_matrix)

diag_matrix = np.diag([1, 2, 3])
print(diag_matrix)

empty_array = np.empty((2, 2))
print(empty_array)
```

Reshaping Arrays:

```
import numpy as np
arr=np.arange(12)
print('Original Array : ',arr)
reshaped_arr=arr.reshape(3,4)
print(reshaped_arr)
```

Slicing Arrays:

```
import numpy as np
arr_2d=np.array([[1,2,3],
        [4,5,6],
        [7,8,9]])
print(arr_2d[0:2,1:3])
```

Array indising:

```
import numpy as np
arr_2d=np.array([[1,2,3],[4,5,6]])
print(arr_2d[1,2])
arr=np.array([[1,2,3],[4,5,6],[7,8,9]])
cond=arr>5
print(arr[cond])
```

```
arr=np.array([10,20,30,40,50])
indices=[1,3]
print(arr[indices])
```

Arithmetic operations:

```
import numpy as np
arr1=np.array([1,2,3])
arr2=np.array([4,5,6])
print('Sum: ',arr1+arr2)
print('Sub: ',arr1-arr2)
print('Mul: ',arr1*arr2)
print('Div: ',arr1/arr2)
```

Logical functions:

```
import numpy as np
arr=np.array([1,10,100])
print('Natural Log:',np.log(arr))
print('Log base to the 10:',np.log(arr))
```

Aggregation functions:

```
import numpy as np
arr=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(np.sum(arr))
print(np.sum(arr,axis=0))
print(np.sum(arr,axis=1))
print(np.mean(arr))
print(np.mean(arr,axis=0))
print(np.mean(arr,axis=1))
print(np.max(arr,axis=0))
print(np.max(arr,axis=0))
print(np.prod)
```

6. **Write programs to use pandas data structures: Frames and series as storage containers and for a variety of data wrangling operations such as**

- Singled level and hierarchical indexing
- Handling missing data
- Arithmetic and Boolean operations on entire columns and tables
- Database type operations (such as merging and aggregation)
- Plotting individual columns and whole tables
- Reading data from files and writing to files

Single handed indexing:

```
import pandas as pd
data=[10,20,30,40]
index=['a','b','c','d']
series=pd.Series(data,index=index)
print(series)
```

Hierotical indexing:

```
import pandas as pd
data={
  'Name':['Alice','Bob','Carry'],
  'Age':[20,21,24],
  'City':['Newyork','Austalia','Chicago']
  }
df=pd.DataFrame(data)
print(df)
```

Handling missing data:

```
import pandas as pd
import numpy as np


df = pd.DataFrame({
  'A': [1, np.nan, 3],
  'B': [4, 5, np.nan],
  'C': [np.nan, 7, 8]
})
```

```
print(df.isna())
print(df.isna().sum())
```

Arithmetic and Boolean operations:

Arithmetic:

```
import pandas as pd

data = {
   'A': [10, 20, 30, 40],
   'B': [5, 10, 15, 20],
}

df = pd.DataFrame(data)
print("Original Table:\n", df)

df['A+B'] = df['A'] + df['B']
df['A-B'] = df['A'] - df['B']
df['A*B'] = df['A'] * df['B']
df['A/B'] = df['A'] / df['B']

print("\nAfter Arithmetic Operations:\n",df)
```

Boolean:

```
import pandas as pd

data = {
   'A': [10, 20, 30, 40],
   'B': [5, 10, 15, 20],
   'C': [2, 4, 6, 8]
}

df = pd.DataFrame(data)
print("Original Table:\n", df)

df['A>15'] = df['A'] > 15
df['B=10'] = df['B'] == 10
```

```
df['C_even'] = df['C'] % 2 == 0

print("\nBoolean Columns Added:\n", df)
```

Merging :

```
import pandas as pd
df1=pd.DataFrame({'ID':[1,2],'Value':['A','B']})
df2=pd.DataFrame({'ID':[1,3],'Description':['Desc1','Desc3']})
result=pd.merge(df1,df2,on='ID')
print(result)
```

Aggregation:

```
import pandas as pd
df=pd.DataFrame({
  'A':[1,2,3,4,5],
  'B':[10,20,30,40,50],
  'C':['a','b','c','d','e']
  })
description=df.describe()
print(description)
```

Plotting individual columns and whole tables :

```
import pandas as pd
import matplotlib.pyplot as plt

data = {
  'Month': ['Jan', 'Feb', 'Mar', 'Apr'],
  'Sales': [200, 250, 300, 280],
  'Profit': [50, 70, 90, 80]
}

df = pd.DataFrame(data)

print("Data:")
print(df)
```

```
plt.figure(figsize=(6,4))
plt.plot(df['Month'], df['Sales'], marker='o', color='blue')
plt.title('Sales Over Months')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.show()

df.plot(x='Month', kind='line', marker='o', figsize=(7,5), title='Sales and
Profit Over Months')
plt.show()
```

Reading data and writing data

Reading :

```
import pandas as pd
df = pd.read_csv('students.csv')

print("Data read from 'students.csv':")
print(df)
```

Writing:

```
import pandas as pd

data = {
   'Name': ['Ravi', 'Sita', 'Amit', 'Priya'],
   'Age': [25, 30, 22, 28],
   'City': ['Delhi', 'Mumbai', 'Chennai', 'Kolkata']
}

df = pd.DataFrame(data)
df.to_csv('students.csv', index=False)
print(" Data written successfully to 'students.csv'")
```