PAI ASSIGNMENT 2

Muhammad Baqar Laghari
24K-0006
BAI-3A

Task 1:

```
import requests

def getLiveScore():
    print("Fetching live cricket scores...")

    try:
        with open('api.txt', 'r') as file:
            apiUrl = file.read()
    except:
        print("Error: Could not read API.")
        return
    try:
        response = requests.get(apiUrl)

        if response.status_code != 200:
            print(f"HTTP Error: {response.status_code} \n Raw response:
{response.text}")
            return

        data = response.json()

        for key, value in data.items():
            if key != "data":
                print(f" {key}: {value}")

        print("\n")

        if data.get("status") != "success":
            reason = data.get("API returned failure status")
            print(f"API Error: {reason}")
            return

        if "data" not in data or not data["data"]:
```

```
            print("No matches found")

        matchList = data["data"]
        print(f"Found {len(matchList)} matches.\n")

        for match in matchList:
            print(f"Match: {match.get('name', 'N/A')}")
            print(f"Status: {match.get('status', 'N/A')}")
            print(f"Venue: {match.get('venue', 'N/A')}")
            print(f"Date: {match.get('date', 'N/A')}")

            if "score" in match and match["score"]:
                for inning in match["score"]:
                    print(f" Inning: {inning.get('inning', 'N/A')}")
                    print(f" Runs:   {inning.get('r', 'N/A')}")
                    print(f" Wickets:{inning.get('w', 'N/A')}")
                    print(f" Overs:  {inning.get('o', 'N/A')}")

            else:
                print("No score available.")

            print("\n")

    except requests.exceptions.RequestException as e:
        print(f"Error: {e}")
```

Output:

```
Fetching live cricket scores...
 apikey: de638428-9c2a-479d-a412-07dbdc5073d8
 status: success
 info: {'hitsToday': 68, 'hitsUsed': 1, 'hitsLimit': 100, 'credits': 0,
'server': 11, 'offsetRows': 0, 'totalRows': 13196, 'queryTime': 269.4098,
's': 0, 'cache': 0}


Found 25 matches.

Match: Nepal vs United States of America, 86th Match
Status: United States of America won by 106 runs
Venue: Dubai International Cricket Stadium, Dubai
Date: 2025-10-26
 Inning: United States of America Inning 1
```

```
 Runs:    262
 Wickets:6
 Overs:   50
 Inning: Nepal Inning 1
 Runs:    156
 Wickets:10
 Overs:   39.1


Match: Brazil vs Mexico, 3rd Match
Status: Mexico won by 54 runs
Venue: Sao Fernando Polo and Cricket Club, Seropedica
...
Wickets:2
 Overs:   40.1
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Task 2:

```python
import numpy as np
print("Importing CSV file")
dataList = []
with open("sensor_data.csv", "r") as file:
    for line in file:
        row = line.strip().split(",")
        rowFloat = [float(value) for value in row]
        dataList.append(rowFloat)
dataArray = np.array(dataList)
print(dataArray.shape)
print(dataArray.dtype)
print("Cleaning data")
dataArray[dataArray == -999] = np.nan
condition = (dataArray > 100 ) | (dataArray < 0)
dataArray[condition] = np.nan
dataArray
columnsMean = np.nanmean(dataArray, axis = 0)
print(f"Mean for each sensor: {columnsMean}")
rowMedian = np.nanmedian(dataArray, axis = 1)
print(f"Median moisture for each hour: {rowMedian}")
```

```
nanCount = np.sum(np.isnan(dataArray), axis = 0)
sensorMostInvalid = np.argmax(nanCount)
maxInvalidCount = nanCount[sensorMostInvalid]
print(f"Invalid readings per senor: {nanCount}")
print(f"Sensor with highest inavlid readings: {sensorMostInvalid}
with {maxInvalidCount} Invalid readings")
maxVal = np.nanmax(dataArray)
minVal = np.nanmin(dataArray)
dataNormalised = (dataArray - minVal) /(maxVal - minVal)

print(f"Max valid reading: {maxVal}, Min valid reading: {minVal}")
print(f"Normalised Array Shape: {dataNormalised.shape}")
np.savetxt("Cleaaned_data.csv", dataNormalised, delimiter= ",",
fmt='%.6f')
print("Cleeaned data saved to Cleaaned_data.csv")
```

Output:

```
Importing CSV file
(720, 100)
Float64
Cleaning data
array([[21.21, 66.86, 47.73, ..., 60.63, 10.51, 83.58],
       [47.24, 71.7 , 68.44, ..., 68.91, 29.68, 38.72],
       [39.07, 21.85, 31.86, ..., 12.77, 12.38, 16.5 ],
       ...,
       [18.6 , 64.89, 41.9 , ..., 16.49, 32.04, 36.97],
       [22.03, 43.26, 55.61, ..., 13.56, 83.69, 61.68],
       [45.06,   nan, 20.37, ..., 86.32, 35.03, 34.74]], shape=(720, 100))
Mean for each sensor: [50.50262763 50.04455505 49.80035222 48.85437309
49.83437879 49.80668175
 51.28129771 50.26348872 50.69630031 51.20293313 50.28427035 50.8485303
 49.43291291 51.46549467 49.20422901 49.37936652 49.21208524 49.25009023
 49.85362687 52.42956781 50.4398806  48.23953383 49.07180451 49.94535769
 49.15453055 49.97029985 50.44724551 49.37434978 49.48810319 50.87231689
 50.35599696 49.49472393 50.13304805 50.21394216 51.98691244 51.39295008
 49.18030257 49.96240122 48.82482389 49.26535769 50.51614693 49.93198519
 50.07898678 49.23602134 51.73808937 49.37856707 50.58920181 48.45622901
 50.09266667 49.26336364 49.62663126 51.06783866 49.32479576 49.64679641
 49.45808383 51.01671733 48.88019118 50.20588235 51.6394494  48.58670641
 50.50378743 51.22426426 50.40354449 50.87356707 49.725997   49.84435045
 51.07342814 50.26929851 48.81086053 50.17495399 48.14147465 48.31043609
 50.424003   49.96523308 48.97831563 49.5033678  50.91120846 50.41424514
 50.32633333 50.99456086 51.59852853 50.85981595 49.85289318 50.09880249
```

```
 49.68850904 49.61254464 47.89051437 49.12515648 47.2306535  49.45568862
 48.53630402 50.5400597  49.3694697  50.96949025 49.15935484 49.45331343
 50.28570776 49.92805175 51.26437596 49.95143722]
Median moisture for each hour: [54.355 55.72  52.275 53.19  53.28  53.
45.65  50.21  55.63  50.34
 47.79  53.92  54.29  46.26  48.69  52.43  55.645 45.59  50.12  41.81
 51.47  50.82  50.5   53.01  45.215 54.195 52.02  37.38  43.3   60.39
 54.1   57.96  47.165 52.69  50.55  51.56  45.29  50.58  59.73  52.945
 53.32  43.015 54.22  53.1   47.37  44.51  45.975 47.78  48.23  52.375
 42.49  49.62  53.17  55.78  56.03  53.995 58.59  46.76  45.455 48.09
 50.51  48.32  53.23  48.375 57.66  46.625 48.32  49.285 43.41  53.13
 53.43  52.05  56.18  55.11  53.09  52.34  50.565 58.46  52.68  52.73
 44.35  50.24  51.65  49.34  53.44  52.53  46.66  53.39  54.33  49.01
 53.085 57.4   56.275 51.35  47.19  50.825 48.625 51.27  53.29  51.92
 47.52  50.05  55.685 55.93  44.63  48.81  50.035 45.05  58.81  50.09
 50.31  60.22  40.845 49.145 54.15  43.1   58.475 52.54  45.14  55.87
 47.85  53.13  50.98  49.16  46.715 47.74  51.415 55.61  46.6   41.79
 49.97  45.89  48.34  44.11  44.82  54.79  52.385 52.1   52.59  49.21
 53.485 49.1   49.835 50.53  53.05  48.19  50.95  42.48  46.26  51.11
 48.08  51.175 52.23  47.44  50.66  46.09  53.81  54.99  49.185 54.19
 52.71  52.19  62.83  57.58  51.19  44.285 50.92  48.39  50.25  57.49
 41.18  51.78  55.485 59.61  52.615 52.66  46.64  48.05  47.97  47.355
 45.665 50.89  45.28  52.18  49.87  52.83  56.03  51.67  56.77  47.77
 49.69  48.355 51.82  55.105 53.44  44.12  51.3   47.42  43.35  54.09
 44.73  45.46  55.985 57.085 51.21  46.28  61.16  53.105 49.92  42.755
 47.85  44.89  45.8   59.18  37.46  50.34  47.65  50.33  43.5   48.56
 48.24  50.43  50.45  47.86  52.78  55.575 51.5   55.54  47.79  50.535
 47.965 53.81  42.89  58.31  54.87  47.6   50.985 46.62  47.235 48.91
 49.28  49.42  50.02  50.8   53.22  46.41  53.76  51.11  44.525 54.45
 57.88  48.93  54.59  46.565 46.725 48.295 48.58  47.38  48.4   50.755
 58.45  52.74  51.675 45.46  53.08  49.355 50.55  48.555 46.14  48.77
 52.16  49.12  44.33  48.965 46.42  50.95  43.975 50.5   51.65  51.62
 60.345 56.17  45.65  47.55  45.51  47.41  48.635 46.72  44.4   52.07
 48.315 52.63  45.795 54.19  57.82  57.735 50.46  54.36  50.24  50.56
 52.09  50.16  51.32  54.375 42.    49.52  51.06  49.04  51.77  49.06
 53.13  52.055 47.79  43.24  46.34  48.26  50.54  49.19  50.57  51.31
 51.88  44.18  41.99  50.32  47.71  44.71  47.57  48.685 48.35  52.37
 51.72  54.67  48.37  50.19  44.78  57.18  50.695 47.04  39.53  60.25
 54.22  50.265 51.87  49.1   55.785 48.675 48.52  46.6   58.08  45.21
 55.22  51.55  56.195 41.855 55.04  52.775 51.66  45.21  49.46  42.205
 53.455 47.86  50.49  49.23  51.64  47.64  45.34  47.89  41.175 52.94
 49.67  48.31  50.45  49.095 53.51  43.455 45.62  49.595 55.26  53.24
 47.68  55.74  51.61  48.74  54.3   55.17  46.28  53.73  49.47  50.735
 41.5   41.88  52.07  57.525 44.46  50.935 54.16  53.88  47.42  50.9
 49.35  53.61  48.67  57.26  45.67  49.89  58.51  49.235 53.085 49.7
 49.02  53.425 45.525 43.6   52.385 46.075 50.22  52.44  44.6   47.995
 49.13  46.84  54.07  46.935 49.845 55.64  54.385 54.65  53.22  56.365
```

```
 50.18  43.77  51.47  46.035 48.115 49.7   51.05  53.16  44.74  53.76
 53.88  55.49  47.52  51.53  50.905 53.72  46.155 51.845 43.45  54.11
 61.165 52.36  43.145 43.28  50.21  48.32  52.33  48.57  49.49  50.1
 56.395 41.51  48.65  49.16  45.6   56.88  46.305 53.23  46.31  47.335
 43.235 44.91  44.59  50.14  43.66  52.74  48.53  50.425 53.85  54.42
 49.185 53.44  49.    49.435 48.98  51.32  43.65  48.03  51.69  43.89
 50.59  40.745 50.83  57.685 50.91  48.9   54.7   51.56  43.08  49.09
 51.64  51.79  54.33  60.13  53.73  40.93  44.865 52.86  53.33  48.8
 55.755 52.89  53.39  49.69  48.65  46.25  43.64  51.95  54.36  44.13
 49.88  50.83  47.72  50.86  44.435 52.13  49.27  49.96  49.57  49.045
 54.43  55.24  55.735 50.9   51.275 51.48  48.72  50.95  41.86  48.5
 53.32  47.385 51.98  52.46  49.02  41.185 51.55  41.67  51.395 49.605
 46.41  55.63  47.615 53.46  44.45  48.17  46.07  41.55  54.16  47.4
 49.38  52.68  49.38  48.495 49.4   50.38  52.43  56.91  56.35  46.75
 47.24  55.53  45.43  54.76  46.55  53.625 47.51  53.58  46.24  45.17
 51.3   47.17  49.215 49.745 50.98  46.72  45.2   53.605 47.64  43.31
 50.75  57.645 45.89  51.93  51.45  50.725 47.08  45.77  44.33  46.93
 47.09  52.39  51.57  50.83  51.53  50.34  49.875 43.1   48.18  49.22
 55.745 49.62  41.32  53.71  49.66  56.76  45.2   47.045 46.645 50.05
 54.34  49.86  51.8   46.94  53.66  47.04  54.38  49.98  50.71  47.315
 61.34  53.11  51.98  52.35  48.79  58.07  50.73  48.565 46.84  47.73
 51.185 46.62  49.98  52.255 46.89  50.88  51.7   54.98  53.32  48.045
 42.42  49.84  48.565 47.745 52.76  51.17  53.105 55.38  44.765 45.84
 48.23  43.045 55.81  50.465 46.49  50.225 45.91  46.88  52.24  44.1
 54.305 53.17  50.91  51.76  42.48  39.8   49.62  49.2   47.22  53.69
 50.04  41.42  49.83  53.81  50.49  48.15  48.22  45.73  42.66  48.24
 44.195 54.545 48.43  48.89  54.83  51.405 54.71  45.46  48.36  51.91
 48.93  49.42  55.605 45.69  50.505 53.46  47.68  49.99  47.1   51.49
 57.1   49.65  46.445 36.775 51.05  53.31  46.355 55.83  51.64  41.76 ]
Invalid readings per senor: [54 57 67 66 60 57 65 55 74 62 69 60 54 63 65
57 63 55 50 49 50 55 55 63
 49 53 52 51 61 51 63 68 54 63 69 59 59 62 67 63 53 45 39 64 71 64 56 65
 45 60 61 63 59 52 52 62 40 57 48 49 52 54 57 64 53 58 52 50 46 68 69 55
 53 55 61 43 58 51 60 71 54 68 46 77 56 48 59 49 62 52 49 50 60 53 69 50
 63 63 71 59]
Sensor with highest inavlid readings: 83 with 77 Invalid readings
Max valid reading: 90.0, Min valid reading: 10.0
Normalised Array Shape: (720, 100)
Cleeaned data saved to Cleaaned_data.csv
```

Task 3:

```python
import pandas as pd

df = pd.read_csv("Titanic-Dataset.csv")

reportLines = []
```

```
for col in df.columns:
    dtype = df[col].dtype
    missingPart = df[col].isna().mean() * 100
    reportLines.append(f'{col}: {dtype}, missing: {missingPart}')
with open("reportFile.txt", "w") as file:
    for line in reportLines:
        file.write(line + '\n')

print("Reported saved to reportFile.txt.")
df["Age"] = df.groupby(['Pclass', 'Sex'])["Age"].transform(lambda
g: g.fillna(g.median()))
embarkedMode = df["Embarked"].mode()[0]
df["Embarked"] = df["Embarked"].fillna(embarkedMode)
if "Cabin" in df.columns:
    df = df.drop(columns=["Cabin"])

print("All Imputations done.")
df['FamilySize'] = df["SibSp"] + df["Parch"]
df["isAlone"] = (df["FamilySize"] == 0).astype(int)
df["Age"] = df["Age"].astype('int64')
df.to_csv("Cleaned_Titanic_Dataset.csv", index=False)
df
```

Output:

```
Reported saved to reportFile.txt.
All Imputations done.
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	FamilySize	isAlone
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	S	1	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	C	1	0
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	S	0	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	S	1	0
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.0500	S	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27	0	0	211536	13.0000	S	0	1
887	888	1	1	Graham, Miss. Margaret Edith	female	19	0	0	112053	30.0000	S	0	1
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	21	1	2	W./C. 6607	23.4500	S	3	0
889	890	1	1	Behr, Mr. Karl Howell	male	26	0	0	111369	30.0000	C	0	1
890	891	0	3	Dooley, Mr. Patrick	male	32	0	0	370376	7.7500	Q	0	1

891 rows × 13 columns

Task 4:

```
import pandas as pd
titanic = pd.read_csv("Cleaned_Titanic_Dataset.csv")
fares = pd.read_csv("ticket_fares.csv")
df = pd.merge(titanic, fares, on="Ticket", how="left")
df
bins = [0, 12, 19, 30, 61, 120]
labels = ['Child', 'Teenager', 'Young Adult', 'Adult', 'Senior']
df['AgeGroup'] = pd.cut(df['Age'], bins = bins, labels =  labels,
include_lowest=True)
df
survivedGenderAge = df.groupby(['Age',
'AgeGroup'])['Survived'].mean().reset_index()
print(survivedGenderAge)
survivedPClass = df.groupby("Pclass")["Survived"].mean().reset_index()
print(survivedPClass)

df["FareBin"] = pd.qcut(df['Fare_x'], q = 4, labels = ["Low", "Medium",
"High", "VeryHigh"])
survivedFareBin = df.groupby("FareBin")["Survived"].mean().reset_index()
print(survivedFareBin)
paragraph1 = """Survival rates grouped y Gender and age show that Women
and Children
```

```
                had more survival rate than Men and Adults and Senior.
This proves the fact that evacuation
                of Women and Children was given more priority than any
other."""

paragraph2 = """Passengers in 1st Class had much more higher survival
rates than those in 2nd
                and 3rd class. When using fare-based quantile bins,
Survival rates also increase
                with increase in higher fares. This confirms wealth was
strongly related to survival chances"""
with open('reportOfTitanicSurvival.txt', 'w') as file:
    file.write(paragraph1.strip() + "\n" + paragraph2.strip())
```

Output:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare_x	Embarked	FamilySize	isAlone	Fare_y
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	S	1	0	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	C	1	0	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	S	0	1	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	S	1	0	53.1000
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	S	1	0	53.1000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1588	888	1	1	Graham, Miss. Margaret Edith	female	19	0	0	112053	30.0000	S	0	1	30.0000
1589	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	21	1	2	W./C. 6607	23.4500	S	3	0	23.4500
1590	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	21	1	2	W./C. 6607	23.4500	S	3	0	23.4500
1591	890	1	1	Behr, Mr. Karl Howell	male	26	0	0	111369	30.0000	C	0	1	30.0000
1592	891	0	3	Dooley, Mr. Patrick	male	32	0	0	370376	7.7500	Q	0	1	7.7500

1593 rows × 14 columns

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare_x	Embarked	FamilySize	isAlone	Fare_y	AgeGroup
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	S	1	0	7.2500	Young Adult
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	C	1	0	71.2833	Adult
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	S	0	1	7.9250	Young Adult
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	S	1	0	53.1000	Adult
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	S	1	0	53.1000	Adult
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1588	888	1	1	Graham, Miss. Margaret Edith	female	19	0	0	112053	30.0000	S	0	1	30.0000	Teenager
1589	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	21	1	2	W./C. 6607	23.4500	S	3	0	23.4500	Young Adult
1590	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	21	1	2	W./C. 6607	23.4500	S	3	0	23.4500	Young Adult
1591	890	1	1	Behr, Mr. Karl Howell	male	26	0	0	111369	30.0000	C	0	1	30.0000	Young Adult
1592	891	0	3	Dooley, Mr. Patrick	male	32	0	0	370376	7.7500	Q	0	1	7.7500	Adult

1593 rows × 15 columns

	Age	AgeGroup	Survived
0	0	Child	1.0
1	0	Teenager	NaN
2	0	Young Adult	NaN
3	0	Adult	NaN

```
4      0       Senior       NaN
..   ...          ...       ...
350   80        Child       NaN
351   80     Teenager       NaN
352   80  Young Adult       NaN
353   80        Adult       NaN
354   80       Senior       1.0
```

	Pclass	Survived
0	1	0.689055
1	2	0.480000
2	3	0.239057

	FareBin	Survived
0	Low	0.229426
1	Medium	0.441687
2	High	0.336449
3	VeryHigh	0.609418