Hands-on Activity 6.1 Introduction to Data Analysis and Tools

Name: Abon, Benedict Aldous A. Section: CPE22S3 Performed on: 02/12/2026 Submitted on: 02/12/2026 Submitted to: Engr. Neil Barton James Matira

Exercise 1

Run the given code below for exercises 1 and 2, perform the given tasks without using any Python modules.

```
import random
random.seed(0)
salaries = [round(random.random()*1000000, -3) for _ in range(100)]
```

Using the data generated above, calculate the following statistics without importing anything from the statistics module in the standard library (https://docs.python.org/3/library/statistics.html) and then confirm your results match up to those that are obtained when using the statistics module (where possible):

- Mean
- Median
- Mode (hint: check out the Counter in the collections module of the standard library at https://docs.python.org/3/library/collections.html#collections.Counter)
- Sample variance
- Sample standard deviation

```
salaries
```

```
[844000.0,
 758000.0,
 421000.0,
 259000.0,
 511000.0,
 405000.0,
 784000.0,
 303000.0,
 477000.0,
 583000.0,
 908000.0,
 505000.0,
 282000.0,
 756000.0,
 618000.0,
 251000.0,
 910000.0,
```

```
983000.0,
810000.0,
902000.0,
310000.0,
730000.0,
899000.0,
684000.0,
472000.0,
101000.0,
434000.0,
611000.0,
913000.0,
967000.0,
477000.0,
865000.0,
260000.0,
805000.0,
549000.0,
14000.0,
720000.0,
399000.0,
825000.0,
668000.0,
1000.0,
494000.0,
868000.0,
244000.0,
325000.0,
870000.0,
191000.0,
568000.0,
239000.0,
968000.0,
803000.0,
448000.0,
80000.0,
320000.0,
508000.0,
933000.0,
109000.0,
551000.0,
707000.0,
547000.0,
814000.0,
540000.0,
964000.0,
603000.0,
588000.0,
445000.0,
```

```
 596000.0,
 385000.0,
 576000.0,
 290000.0,
 189000.0,
 187000.0,
 613000.0,
 657000.0,
 477000.0,
 90000.0,
 758000.0,
 877000.0,
 923000.0,
 842000.0,
 898000.0,
 923000.0,
 541000.0,
 391000.0,
 705000.0,
 276000.0,
 812000.0,
 849000.0,
 895000.0,
 590000.0,
 950000.0,
 580000.0,
 451000.0,
 660000.0,
 996000.0,
 917000.0,
 793000.0,
 82000.0,
 613000.0,
 486000.0]
```

```
# Mean
salaries_mean = sum(salaries) / len(salaries)
print(f'Mean Salary: {salaries_mean}')
```

```
Mean Salary: 585690.0
```

```
# Median
salaries_median = salaries[len(salaries)//2]
print(f'Median Salary: {salaries_median}')
```

```
Median Salary: 803000.0
```

```
# Mode
salaries_mode = max(salaries, key=lambda x: salaries.count(x))
salaries_mode
```

```
477000.0
```

```
# Sample variance
average = salaries_mean
len_minus_one = len(salaries)-1

x_mean_squared = []
for salary in salaries:
  temp = (salary - average)**2
  x_mean_squared.append(temp)

summation = sum(x_mean_squared)
salaries_variance = summation/len_minus_one

print(f'Salaries Sample Variance: {salaries_variance:.2f}')
```

```
Salaries Sample Variance: 70664054444.44
```

```
# Sample standard deviation
salaries_std = salaries_variance ** 0.5

print(f'Salaries Standard Deviation: {salaries_std:.2f}')
```

```
Salaries Standard Deviation: 265827.11
```

Exercise 2

Using the same data, calculate the following statistics using the functions in the statistics module where appropriate:

- Range
- Coefficient of variation
- Interquartile range
- Quartile coefficient of dispersion

```
# Range
salaries_range = max(salaries) - min(salaries)
print(f'Salaries Range: {salaries_range}')
```

```
Salaries Range: 995000.0
```

```
# Coefficient of variation
salaries_cov = salaries_std / salaries_mean
print(f'Salaries Coefficient of Variation: {salaries_cov}')
```

```
Salaries Coefficient of Variation: 0.45386998894439035
```

```
# Interquartile range
salaries.sort()
q1 = .25 * (len(salaries)+1)
```

```
q3 = .75 * (len(salaries)+1)
salaries_iqr = q3 - q1

print(f'Salaries Interquartile Range: {salaries_iqr}')
```

```
Salaries Interquartile Range: 50.5
```

```
# Quartile coefficient of dispersion
salaries_qcd = (q3-q1)/(q3+q1)

print(f'Salaries Quartile Coefficient of Dispersion: {salaries_qcd}')
```

```
Salaries Quartile Coefficient of Dispersion: 0.5
```

Exercise 3: Pandas for Data Analysis

Load the diabetes.csv file. Convert the diabetes.csv into dataframe

```
import pandas as pd

diabetes = pd.read_csv('diabetes.csv')
diabetes.head()
```


```
{"summary":"{\n  \"name\": \"diabetes\",\n  \"rows\": 768,\n
\"fields\": [\n    {\n      \"column\": \"Pregnancies\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 17,\n
\"num_unique_values\": 17,\n        \"samples\": [\n          6,\n
1,\n          3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Glucose\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 31,\n        \"min\": 0,\n        \"max\": 199,\n
\"num_unique_values\": 136,\n        \"samples\": [\n          151,\n
101,\n          112\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BloodPressure\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 19,\n        \"min\": 0,\n
\"max\": 122,\n        \"num_unique_values\": 47,\n
\"samples\": [\n          86,\n          46,\n          85\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SkinThickness\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 15,\n        \"min\": 0,\n
\"max\": 99,\n        \"num_unique_values\": 51,\n        \"samples\":
[\n          7,\n          12,\n          48\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Insulin\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 115,\n
```

```
\"min\": 0,\n        \"max\": 846,\n        \"num_unique_values\":
186,\n        \"samples\": [\n          52,\n          41,\n
183\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BMI\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7.8841603203754405,\n        \"min\": 0.0,\n        \"max\":
67.1,\n        \"num_unique_values\": 248,\n        \"samples\": [\n
19.9,\n          31.0,\n          38.1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DiabetesPedigreeFunction\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.33132859501277484,\n        \"min\": 0.078,\n        \"max\": 2.42,\
n        \"num_unique_values\": 517,\n        \"samples\": [\n
1.731,\n          0.426,\n          0.138\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 11,\n        \"min\": 21,\n
\"max\": 81,\n        \"num_unique_values\": 52,\n        \"samples\":
[\n          60,\n          47,\n          72\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Outcome\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"diabetes"}
```


Perform the following tasks in the diabetes dataframe:

1. Identify the column names
2. Identify the data types of the data
3. Display the total number of records
4. Display the first 20 records
5. Display the last 20 records
6. Change the Outcome column to Diagnosis
7. Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes"
8. Create a new dataframe "withDiabetes" that gathers data with diabetes
9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes
10. Create a new dataframe "Pedia" that gathers data with age 0 to 19
11. Create a new dataframe "Adult" that gathers data with age greater than 19
12. Use numpy to get the average age and glucose value.
13. Use numpy to get the median age and glucose value.
14. Use numpy to get the middle values of glucose and age.
15. Use numpy to get the standard deviation of the skinthickness.

```
# 1. Identify the column names
diabetes.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

```
# 2. Identify the data types of the data
diabetes.dtypes
```

```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

```
# 3. Display the total number of records
len(diabetes)
```

```
768
```

```
# 4. Display the first 20 records
diabetes.head(20)
```


```
{"summary":"{\n  \"name\": \"diabetes\",\n  \"rows\": 768,\n
\"fields\": [\n    {\n      \"column\": \"Pregnancies\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 17,\n
\"num_unique_values\": 17,\n        \"samples\": [\n          6,\n
1,\n          3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Glucose\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 31,\n        \"min\": 0,\n        \"max\": 199,\n
\"num_unique_values\": 136,\n        \"samples\": [\n          151,\n
101,\n          112\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BloodPressure\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 19,\n        \"min\": 0,\n
\"max\": 122,\n        \"num_unique_values\": 47,\n
\"samples\": [\n          86,\n          46,\n          85\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SkinThickness\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 15,\n        \"min\": 0,\n
\"max\": 99,\n        \"num_unique_values\": 51,\n        \"samples\":
```

[\n   7,\n   12,\n   48\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Insulin\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 115,\n
\"min\": 0,\n   \"max\": 846,\n   \"num_unique_values\":
186,\n   \"samples\": [\n   52,\n   41,\n
183\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"BMI\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 7.8841603203754405,\n   \"min\": 0.0,\n   \"max\":
67.1,\n   \"num_unique_values\": 248,\n   \"samples\": [\n
19.9,\n   31.0,\n   38.1\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"DiabetesPedigreeFunction\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.33132859501277484,\n   \"min\": 0.078,\n   \"max\": 2.42,\
n   \"num_unique_values\": 517,\n   \"samples\": [\n
1.731,\n   0.426,\n   0.138\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 11,\n   \"min\": 21,\n
\"max\": 81,\n   \"num_unique_values\": 52,\n   \"samples\":
[\n   60,\n   47,\n   72\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Outcome\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 0,\n
\"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   0,\n   1\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"diabetes"}

```
# 5. Display the last 20 records
diabetes.tail(20)
```

{"summary":"{\n \"name\": \"diabetes\",\n \"rows\": 20,\n
\"fields\": [\n   {\n   \"column\": \"Pregnancies\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
3,\n   \"min\": 0,\n   \"max\": 10,\n
\"num_unique_values\": 11,\n   \"samples\": [\n   8,\n
3,\n   10\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Glucose\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 32,\n   \"min\": 88,\n   \"max\": 190,\n
\"num_unique_values\": 19,\n   \"samples\": [\n   187,\n
181,\n   190\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"BloodPressure\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 10,\n   \"min\": 58,\n
\"max\": 92,\n   \"num_unique_values\": 11,\n   \"samples\":
[\n   72,\n   70,\n   74\n   ],\n

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"SkinThickness\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
17,\n   \"min\": 0,\n   \"max\": 48,\n
\"num_unique_values\": 12,\n   \"samples\": [\n   27,\n
48,\n   22\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Insulin\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 123,\n   \"min\": 0,\n   \"max\": 510,\n
\"num_unique_values\": 8,\n   \"samples\": [\n   0,\n
16,\n   200\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"BMI\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 5.896946132042459,\n   \"min\": 22.5,\n   \"max\":
44.0,\n   \"num_unique_values\": 20,\n   \"samples\": [\n
36.4,\n   26.2,\n   32.9\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"DiabetesPedigreeFunction\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.2848823348978349,\n   \"min\": 0.142,\n   \"max\": 1.182,\
n   \"num_unique_values\": 20,\n   \"samples\": [\n
0.408,\n   0.245,\n   0.171\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 13,\n   \"min\": 22,\n
\"max\": 66,\n   \"num_unique_values\": 18,\n   \"samples\":
[\n   36,\n   50,\n   39\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Outcome\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 0,\n
\"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   0,\n   1\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

```
# 6. Change the Outcome column to Diagnosis
diabetes = diabetes.rename(columns={'Outcome':'Diagnosis'})
diabetes.head()
```

{"summary":"{\n \"name\": \"diabetes\",\n \"rows\": 768,\n
\"fields\": [\n   {\n   \"column\": \"Pregnancies\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
3,\n   \"min\": 0,\n   \"max\": 17,\n
\"num_unique_values\": 17,\n   \"samples\": [\n   6,\n
1,\n   3\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Glucose\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 31,\n   \"min\": 0,\n   \"max\": 199,\n
\"num_unique_values\": 136,\n   \"samples\": [\n   151,\n
101,\n   112\n   ],\n   \"semantic_type\": \"\",\n

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"BloodPressure\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 19,\n   \"min\": 0,\n
\"max\": 122,\n   \"num_unique_values\": 47,\n
\"samples\": [\n   86,\n   46,\n   85\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"SkinThickness\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 15,\n   \"min\": 0,\n
\"max\": 99,\n   \"num_unique_values\": 51,\n   \"samples\":
[\n   7,\n   12,\n   48\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Insulin\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 115,\n
\"min\": 0,\n   \"max\": 846,\n   \"num_unique_values\":
186,\n   \"samples\": [\n   52,\n   41,\n
183\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"BMI\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 7.8841603203754405,\n   \"min\": 0.0,\n   \"max\":
67.1,\n   \"num_unique_values\": 248,\n   \"samples\": [\n
19.9,\n   31.0,\n   38.1\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"DiabetesPedigreeFunction\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.33132859501277484,\n   \"min\": 0.078,\n   \"max\": 2.42,\
n   \"num_unique_values\": 517,\n   \"samples\": [\n
1.731,\n   0.426,\n   0.138\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 11,\n   \"min\": 21,\n
\"max\": 81,\n   \"num_unique_values\": 52,\n   \"samples\":
[\n   60,\n   47,\n   72\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Diagnosis\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 0,\n   \"max\": 1,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Classification\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 1,\n   \"samples\":
[\n   \"No Diabetes\"\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"diabetes"}

```
# 7. Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes"
diabetes['Classification'] = diabetes['Diagnosis'].apply(lambda x: 'Diabetes' if x == 1 else 'No Diabetes')
```

```
diabetes.head()
```


```
{"summary":"{\n  \"name\": \"diabetes\",\n  \"rows\": 768,\n
\"fields\": [\n    {\n      \"column\": \"Pregnancies\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 17,\n
\"num_unique_values\": 17,\n        \"samples\": [\n          6,\n
1,\n          3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Glucose\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 31,\n        \"min\": 0,\n        \"max\": 199,\n
\"num_unique_values\": 136,\n        \"samples\": [\n          151,\n
101,\n          112\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BloodPressure\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 19,\n        \"min\": 0,\n
\"max\": 122,\n        \"num_unique_values\": 47,\n
\"samples\": [\n          86,\n          46,\n          85\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SkinThickness\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 15,\n        \"min\": 0,\n
\"max\": 99,\n        \"num_unique_values\": 51,\n        \"samples\":
[\n          7,\n          12,\n          48\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Insulin\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 115,\n
\"min\": 0,\n        \"max\": 846,\n        \"num_unique_values\":
186,\n        \"samples\": [\n          52,\n          41,\n
183\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BMI\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7.8841603203754405,\n        \"min\": 0.0,\n        \"max\":
67.1,\n        \"num_unique_values\": 248,\n        \"samples\": [\n
19.9,\n          31.0,\n          38.1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DiabetesPedigreeFunction\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.33132859501277484,\n        \"min\": 0.078,\n        \"max\": 2.42,\
n        \"num_unique_values\": 517,\n        \"samples\": [\n
1.731,\n          0.426,\n          0.138\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 11,\n        \"min\": 21,\n
\"max\": 81,\n        \"num_unique_values\": 52,\n        \"samples\":
[\n          60,\n          47,\n          72\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Diagnosis\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
```

```
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Classification\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"No Diabetes\",\n          \"Diabetes\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"diabetes"}
```


```
# 8. Create a new dataframe "withDiabetes" that gathers data with
diabetes
withDiabetes = diabetes[diabetes['Classification'] == 'Diabetes']
withDiabetes.head()
```


```
{"summary":"{\n  \"name\": \"withDiabetes\",\n  \"rows\": 268,\n
\"fields\": [\n    {\n      \"column\": \"Pregnancies\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 17,\n
\"num_unique_values\": 17,\n        \"samples\": [\n          6,\n
8,\n          10\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Glucose\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 31,\n        \"min\": 0,\n        \"max\": 199,\n
\"num_unique_values\": 104,\n        \"samples\": [\n          131,\n
85,\n          151\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BloodPressure\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 21,\n        \"min\": 0,\n
\"max\": 114,\n        \"num_unique_values\": 39,\n
\"samples\": [\n          58,\n          100,\n          70\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SkinThickness\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 17,\n        \"min\": 0,\n
\"max\": 99,\n        \"num_unique_values\": 43,\n        \"samples\":
[\n          48,\n          18,\n          46\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Insulin\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 138,\n
\"min\": 0,\n        \"max\": 846,\n        \"num_unique_values\":
93,\n        \"samples\": [\n          176,\n          99,\n
249\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BMI\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7.262967242346376,\n        \"min\": 0.0,\n        \"max\":
67.1,\n        \"num_unique_values\": 148,\n        \"samples\": [\n
34.4,\n          35.4,\n          46.1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DiabetesPedigreeFunction\",\n
```

\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.372354483554611,\n   \"min\": 0.088,\n   \"max\": 2.42,\n
\"num_unique_values\": 231,\n   \"samples\": [\n   0.378,\
n   0.578,\n   0.484\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 10,\n   \"min\": 21,\n
\"max\": 70,\n   \"num_unique_values\": 45,\n   \"samples\":
[\n   35,\n   61,\n   23\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Diagnosis\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 1,\n   \"max\": 1,\n
\"num_unique_values\": 1,\n   \"samples\": [\n   1\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Classification\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 1,\n   \"samples\": [\n
\"Diabetes\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"withDiabetes"}

```
# 9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes
noDiabetes = diabetes[diabetes['Classification'] == 'No Diabetes']
noDiabetes.head()
```

{"summary":"{\n \"name\": \"noDiabetes\",\n \"rows\": 500,\n
\"fields\": [\n   {\n   \"column\": \"Pregnancies\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
3,\n   \"min\": 0,\n   \"max\": 13,\n
\"num_unique_values\": 14,\n   \"samples\": [\n   7,\n
0,\n   1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Glucose\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 26,\n   \"min\": 0,\n   \"max\": 197,\n
\"num_unique_values\": 111,\n   \"samples\": [\n   193,\n
145,\n   110\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"BloodPressure\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 18,\n   \"min\": 0,\n
\"max\": 122,\n   \"num_unique_values\": 43,\n
\"samples\": [\n   94,\n   65,\n   68\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"SkinThickness\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 14,\n   \"min\": 0,\n
\"max\": 60,\n   \"num_unique_values\": 46,\n   \"samples\":
[\n   43,\n   54,\n   40\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\

```
n    },\n    {\n      \"column\": \"Insulin\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 98,\n
\"min\": 0,\n        \"max\": 744,\n        \"num_unique_values\":
137,\n        \"samples\": [\n          79,\n          193,\n
342\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BMI\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7.689855011650112,\n        \"min\": 0.0,\n        \"max\":
57.3,\n        \"num_unique_values\": 210,\n        \"samples\": [\n
46.8,\n          35.2,\n          25.2\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DiabetesPedigreeFunction\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.29908530435741093,\n        \"min\": 0.078,\n        \"max\":
2.329,\n        \"num_unique_values\": 372,\n        \"samples\": [\n
0.133,\n          0.304,\n          0.512\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 11,\n        \"min\": 21,\n
\"max\": 81,\n        \"num_unique_values\": 51,\n        \"samples\":
[\n          81,\n          49,\n          66\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Diagnosis\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Classification\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 1,\n        \"samples\": [\n          \"No
Diabetes\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"noDiabetes"}
```


```
# 10. Create a new dataframe "Pedia" that gathers data with age 0 to
19
Pedia = diabetes[diabetes['Age'] <= 19]
Pedia.head()
```


```
{"repr_error":"Out of range float values are not JSON compliant:
nan","type":"dataframe","variable_name":"Pedia"}
```


```
# 11. Create a new dataframe "Adult" that gathers data with age
greater than 19
Adult = diabetes[diabetes['Age'] > 19]
Adult.head()
```


```
{"summary":"{\n  \"name\": \"Adult\",\n  \"rows\": 768,\n  \"fields\":
[\n    {\n      \"column\": \"Pregnancies\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 3,\n        \"min\":
```

```
0,\n        \"max\": 17,\n        \"num_unique_values\": 17,\n
\"samples\": [\n          6,\n          1,\n          3\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Glucose\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 31,\n
\"min\": 0,\n        \"max\": 199,\n        \"num_unique_values\":
136,\n        \"samples\": [\n          151,\n          101,\n
112\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BloodPressure\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 19,\n        \"min\": 0,\n
\"max\": 122,\n        \"num_unique_values\": 47,\n
\"samples\": [\n          86,\n          46,\n          85\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SkinThickness\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 15,\n        \"min\": 0,\n
\"max\": 99,\n        \"num_unique_values\": 51,\n        \"samples\":
[\n          7,\n          12,\n          48\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Insulin\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 115,\n
\"min\": 0,\n        \"max\": 846,\n        \"num_unique_values\":
186,\n        \"samples\": [\n          52,\n          41,\n
183\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BMI\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7.8841603203754405,\n        \"min\": 0.0,\n        \"max\":
67.1,\n        \"num_unique_values\": 248,\n        \"samples\": [\n
19.9,\n          31.0,\n          38.1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DiabetesPedigreeFunction\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.33132859501277484,\n        \"min\": 0.078,\n        \"max\": 2.42,\
n        \"num_unique_values\": 517,\n        \"samples\": [\n
1.731,\n          0.426,\n          0.138\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 11,\n        \"min\": 21,\n
\"max\": 81,\n        \"num_unique_values\": 52,\n        \"samples\":
[\n          60,\n          47,\n          72\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Diagnosis\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Classification\",\n      \"properties\": {\n        \"dtype\":
```

```
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"No Diabetes\",\n          \"Diabetes\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"Adult"}
```

```
# 12. Use numpy to get the average age and glucose value.
import numpy as np

avg_age = diabetes['Age'].mean()
avg_glucose = diabetes['Glucose'].mean()

print(f'Average age: {avg_age}\nAverage glucose: {avg_glucose}')
```

```
Average age: 33.240885416666664
Average glucose: 120.89453125
```

```
# 13. Use numpy to get the median age and glucose value.
diabetes.sort_values(by='Age')

median_age = diabetes['Age'].median()
median_glucose = diabetes['Glucose'].median()

print(f'Median age: {median_age}\nMedian glucose: {median_glucose}')
```

```
Median age: 29.0
Median glucose: 117.0
```

```
# 14. Use numpy to get the middle values of glucose and age.
median_age = diabetes['Age'].median()
median_glucose = diabetes['Glucose'].median()

print(f'Middle age: {median_age}\nMiddle glucose: {median_glucose}')
```

```
Middle age: 29.0
Middle glucose: 117.0
```

```
# 15. Use numpy to get the standard deviation of the skinthickness.
std_skin = diabetes['SkinThickness'].std()
print(f'Skin Thickness STD: {median_glucose}')
```

```
Skin Thickness STD: 117.0
```

Conclusion

The activity demonstrates the use of pandas and numpy in data analysis and exploration. I was able to familiarize myself with the steps involved in data analysis and syntaxes associated with them. In the first and second exercise, I used raw Python code in finding the means of central tendency of the given dataset, testing my Python knowledge and logical skills. In the third exercise, I used pandas to analyze the following dataset. I was able to explore its data, alter the

column names, and created sub DataFrames from the original DataFrame. I also used numpy syntaxes to find the means of central tendency of the DataFrame.