Name: Apuyan, Viktor Angelo

Section: CPE22S3

Performed on: 03/07/2024

Submitted on: 03/07/2024

Submitted to: Engr. Roman M. Richard

Exercise 1

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

Out[ ]:
```
[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]
```

In [ ]:
```
#MEAN
n = len(salaries)

get_sum = sum(salaries)
mean = get_sum / n

print("Mean / Average is: " + str(mean))
```

```
Mean / Average is: 585690.0
```

In [ ]:
```
#Median
salaries.sort()
median_form = salaries[49] + salaries[50]
med = median_form / 2

print("The Median is: " + str(med))
```

```
The Median is: 589000.0
```

In [ ]:
```
#Mode
salaries.sort()
list_1=[] # Create an empty list to store the frequency of each salary

# Iterate through the salary list
i=0
while i<len(salaries):
  list_1.append(salaries.count(salaries[i]))  # Count the occurrences of the curren
  i+=1 # increment the counter

dic = dict(zip(salaries, list_1)) # Create a dictionary to map salaries to their fr
dic2 = {k for (k,v)in dic.items() if v== max(list_1)} # Create a set to store salar

print("The mode for this dataset: " + str(dic2))
```

```
The mode for this dataset: {477000.0}
```

In [ ]:
```
#Sample Variance
summation = 0
for x in salaries:
  summation += (x - mean)**2

sample_var = summation/(len(salaries)-1)

print("Sample Variance for this dataset: " + str(sample_var))
```

```
Sample Variance for this dataset: 70664054444.44444
```

In [ ]:
```
#Sample Standard Deviation
stdev = sample_var**0.5
print("Standard Deviation for this dataset: " + str(stdev))
```

```
Standard Deviation for this dataset: 265827.11382484
```

Exercise 2

In [ ]:
```
#Range
range_of_data = max(salaries) - min(salaries)
print("The range of the dataset is: " + str(range_of_data))
```

```
The range of the dataset is: 995000.0
```

In [ ]:
```
#Coefficient of variation Interquartile range
q1 = np.percentile(salaries, 25)
q3 = np.percentile(salaries, 75)
iqr = q3 - q1
```

```
cov = np.std(salaries) / np.mean(salaries) * 100
print("COV: " + str(cov))
print("IQR: " + str(iqr))
```

```
COV: 45.15949370793889
IQR: 413250.0
```

In [ ]:
```
#Quartile coefficient of dispersion
qcd = (q3 - q1) / (2 * med)
print("Quartile Coefficient of Dispersion:", qcd)
```

```
Quartile Coefficient of Dispersion: 0.35080645161290325
```

Exercise 3: Pandas for Data Analysis

In [ ]:
```
import pandas as pd
import numpy as np

diabetes = pd.read_csv(("diabetes.csv"))
diabetes
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-3-7662f8a01cc7> in <cell line: 4>()
      2 import numpy as np
      3
----> 4 diabetes = pd.read_csv(("diabetes.csv"))
      5 diabetes

/usr/local/lib/python3.10/dist-packages/pandas/util/_decorators.py in wrapper(*args,
**kwargs)
    209                 else:
    210                     kwargs[new_arg_name] = new_arg_value
--> 211             return func(*args, **kwargs)
    212
    213         return cast(F, wrapper)

/usr/local/lib/python3.10/dist-packages/pandas/util/_decorators.py in wrapper(*args,
**kwargs)
    329                     stacklevel=find_stack_level(),
    330                 )
--> 331             return func(*args, **kwargs)
    332
    333         # error: "Callable[[VarArg(Any), KwArg(Any)], Any]" has no

/usr/local/lib/python3.10/dist-packages/pandas/io/parsers/readers.py in read_csv(fil
epath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, prefix,
mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinitials
pace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose, s
kip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, day
first, cache_dates, iterator, chunksize, compression, thousands, decimal, linetermin
ator, quotechar, quoting, doublequote, escapechar, comment, encoding, encoding_error
s, dialect, error_bad_lines, warn_bad_lines, on_bad_lines, delim_whitespace, low_mem
ory, memory_map, float_precision, storage_options)
    948     kwds.update(kwds_defaults)
    949
--> 950     return _read(filepath_or_buffer, kwds)
    951
    952

/usr/local/lib/python3.10/dist-packages/pandas/io/parsers/readers.py in _read(filepa
th_or_buffer, kwds)
    603
    604     # Create the parser.
--> 605     parser = TextFileReader(filepath_or_buffer, **kwds)
    606
    607     if chunksize or iterator:

/usr/local/lib/python3.10/dist-packages/pandas/io/parsers/readers.py in __init__(sel
f, f, engine, **kwds)
   1440
   1441         self.handles: IOHandles | None = None
-> 1442         self._engine = self._make_engine(f, self.engine)
   1443
   1444     def close(self) -> None:

/usr/local/lib/python3.10/dist-packages/pandas/io/parsers/readers.py in _make_engine
```

```
(self, f, engine)
   1733                 if "b" not in mode:
   1734                     mode += "b"
-> 1735             self.handles = get_handle(
   1736                 f,
   1737                 mode,

/usr/local/lib/python3.10/dist-packages/pandas/io/common.py in get_handle(path_or_bu
f, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    854         if ioargs.encoding and "b" not in ioargs.mode:
    855             # Encoding
--> 856             handle = open(
    857                 handle,
    858                 ioargs.mode,

FileNotFoundError: [Errno 2] No such file or directory: 'diabetes.csv'
```

1. Identify Column Names

In [ ]:
```
column_names = list(diabetes.columns.values)
print("Column Names :", column_names)
```

```
Column Names : ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
n', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
```

2. Identify the data types of the data

In [ ]:
```
step_counts = pd.Series(salaries, name='salaries')
step_counts.dtype
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-46050d306ca6> in <cell line: 1>()
----> 1 step_counts = pd.Series(salaries, name='salaries')
      2 step_counts.dtype

NameError: name 'pd' is not defined
```

3. Display the total number of records

In [ ]:
```
text = "Total Number of Records: "
total_records = len(diabetes)
print(text + str(total_records))
```

```
Total Number of Records: 768
```

4. Display the first 20 records

In [ ]:
```
diabetes.head(20)
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFur
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
5	5	116	74	0	0	25.6	
6	3	78	50	32	88	31.0	
7	10	115	0	0	0	35.3	
8	2	197	70	45	543	30.5	
9	8	125	96	0	0	0.0	
10	4	110	92	0	0	37.6	
11	10	168	74	0	0	38.0	
12	10	139	80	0	0	27.1	
13	1	189	60	23	846	30.1	
14	5	166	72	19	175	25.8	
15	7	100	0	0	0	30.0	
16	0	118	84	47	230	45.8	
17	7	107	74	0	0	29.6	
18	1	103	30	38	83	43.3	
19	1	115	70	30	96	34.6	

5. Display the last 20 records

In [ ]:
```
diabetes.tail(20)
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
748	3	187	70	22	200	36.4	
749	6	162	62	0	0	24.3	
750	4	136	70	0	0	31.2	
751	1	121	78	39	74	39.0	
752	3	108	62	24	0	26.0	
753	0	181	88	44	510	43.3	
754	8	154	78	32	0	32.4	
755	1	128	88	39	110	36.5	
756	7	137	90	41	0	32.0	
757	0	123	72	0	0	36.3	
758	1	106	76	0	0	37.5	
759	6	190	92	0	0	35.5	
760	2	88	58	26	16	28.4	
761	9	170	74	31	0	44.0	
762	9	89	62	0	0	22.5	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

6. Change the Outcome column to Diagnosis

In [ ]:
```
#before renaming
diabetes
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-60e0e959e0f9> in <cell line: 2>()
      1 #before renaming
----> 2 diabetes

NameError: name 'diabetes' is not defined
```

In [ ]:
```
#after renaming
diabetes.rename(columns = {'Outcome':'Diagnosis'}, inplace = True)
diabetes.head()
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0
2	8	183	64	0	0	23.3	0
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2

7. Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes"

In [ ]:
```
diabetes['Classification'] = np.where(diabetes['Diagnosis'] == 1, 'Diabetes', 'No D
diabetes
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-135d8b250265> in <cell line: 1>()
----> 1 diabetes['Classification'] = np.where(diabetes['Diagnosis'] == 1, 'Diabete
s', 'No Diabetes')
      2 diabetes

NameError: name 'diabetes' is not defined
```

8. Create a new dataframe "withDiabetes" that gathers data with diabetes

In [ ]:
```
diabetes = pd.DataFrame(diabetes)
withDiabetes = diabetes[diabetes['Diagnosis'] == 1].copy()

withDiabetes
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	6	148	72	35	0	33.6	
2	8	183	64	0	0	23.3	
4	0	137	40	35	168	43.1	
6	3	78	50	32	88	31.0	
8	2	197	70	45	543	30.5	
...	...	...	...	...	...	...	
755	1	128	88	39	110	36.5	
757	0	123	72	0	0	36.3	
759	6	190	92	0	0	35.5	
761	9	170	74	31	0	44.0	
766	1	126	60	0	0	30.1	

268 rows × 10 columns

9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes

In [ ]:
```
diabetes = pd.DataFrame(diabetes)
noDiabetes = diabetes[diabetes['Diagnosis'] == 0].copy()

noDiabetes
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
1	1	85	66	29	0	26.6	
3	1	89	66	23	94	28.1	
5	5	116	74	0	0	25.6	
7	10	115	0	0	0	35.3	
10	4	110	92	0	0	37.6	
...	...	...	...	...	...	...	
762	9	89	62	0	0	22.5	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
767	1	93	70	31	0	30.4	

500 rows × 10 columns

10. Create a new dataframe "Pedia" that gathers data with age 0 to 19

In [ ]:
```
pedia = diabetes[diabetes['Age'] <= 19]
pedia
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct

11. Create a new dataframe "Adult" that gathers data with age greater than 19

In [ ]:
```
Adult = diabetes[diabetes['Age'] > 19]
Adult
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 10 columns

12. Use numpy to get the average age and glucose value.

In [ ]:
```
mean = np.mean(diabetes['Age']), np.mean(data['Glucose'])
for x in mean:
  print(x)
```

```
33.240885416666664
120.89453125
```

13. Use numpy to get the median age and glucose value.

In [ ]:
```
get_median = np.median(diabetes['Glucose']), np.median(diabetes['Age'])
get_median
```

Out[ ]: (117.0, 29.0)

14. Use numpy to get the middle values of glucose and age.

In [ ]:
```
median = np.median(data['Age']), np.median(data['Glucose'])
for x in median:
  print(x)
```

```
29.0
117.0
```

15. Use numpy to get the standard deviation of the skinthickness.

In [ ]:
```
stdev_skinthick = np.std(diabetes['SkinThickness']) #
print("Standard Deviation for this dataset:", stdev_skinthick)
nump_median = np.median(diabetes['Age']), np.median(diabetes['Glucose'])
for x in nump_median:
  print(x)
```

```
Standard Deviation for this dataset: 15.941828626496939
29.0
117.0
```

6.4 Conclusion

What I've learned in this HOA, is that I was able to learn a bit of an overview about data analysis. It kind of reminds of our DBMS course last semester wherein we would display a table and manipulate certain things for us to display important information. Here in this HOA, it is exactly the same but we are using python programming. I was glad that there are built-in modules for us to make our lives easier, because when I was doing exercise 1 without using the statistic modules, it is difficult and at the same time it lengthens the code.Furthermore, I got to learn how to use different