Statistical Methods for Data Science

Course Code: MEAD-609

Lab File

Submitted to: **Dr. Sunita Prasad**

Submitted by

**Shantanu Shukla**

**Enrollment No. - 01211805424**

in partial fulfillment for the award of the degree

of

*Master of Technology - AI/DS*



Center for Development of Advanced Computing, Noida

*Affiliated to Guru Gobind Singh Indraprastha University*

**Index**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Title** | **Page No.** |
| 1. | Find the mean of all valid columns in the given Data | 3 |
| 2. | Generate the data summary for the given data set | 6 |
| 3. | Descriptive Statistics – Graphical Display | 10 |
| 4. | Exercise-4 | 15 |

* **EXERCISE-1**

**Objective:** Find the mean of all valid columns in the given DATA.

**DATA:**

A screenshot of a table

Description automatically generated

**CODE:**

1. import os

2. import pandas as pd

3.

4. # Read the data from the file gapminder.csv

5. current\_dir = os.path.dirname(\_\_file\_\_)

6. data = pd.read\_csv(current\_dir + "/dataset/gapminder.csv")

7.

8. # Verify that the data has been loaded correctly

9. print(data.head())

10.

11. # Replace missing values in 'lifeExp' with 0

12. data["lifeExp"].fillna(0, inplace=True)

13.

14. # Calculate the mean of lifeExp from all the data

15. mean\_all = data["lifeExp"].mean()

16. print("Mean of lifeExp from all the data:", mean\_all)

17.

18. # Calculate the mean of lifeExp by country

19. mean\_country = data.groupby("country")["lifeExp"].mean()

20. print("\nMean of lifeExp by country:")

21. print(mean\_country)

22.

23. # Calculate the mean of lifeExp by year

24. mean\_year = data.groupby("year")["lifeExp"].mean()

25. print("\nMean of lifeExp by year:")

26. print(mean\_year)

27.

28. # Calculate the mean of lifeExp by continent

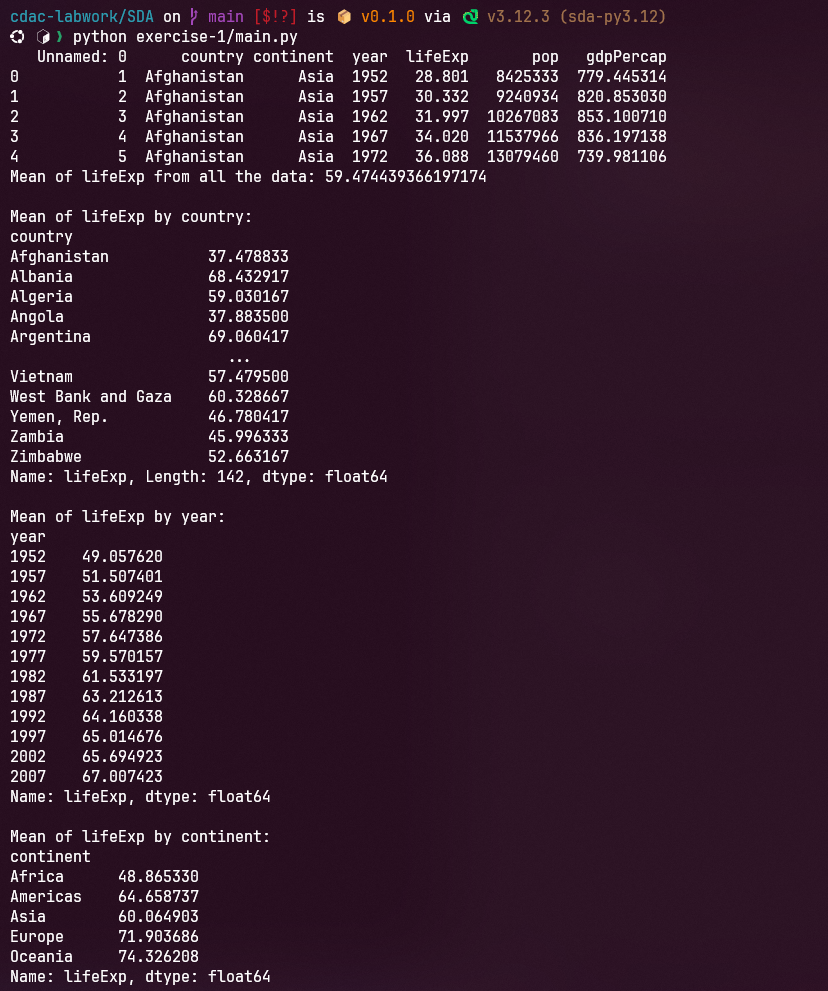
29. mean\_continent = data.groupby("continent")["lifeExp"].mean()

30. print("\nMean of lifeExp by continent:")

31. print(mean\_continent)

32.

**OUTPUT:**

****

* **EXERCISE-2**

**Objective:** Generate the data summary for the given data set.

**DATA:**

**A screenshot of a computer

Description automatically generated**

**CODE:**

1. import pandas as pd

2. import os

3.

4. # Read the data from the file

5. current\_dir = os.path.dirname(\_\_file\_\_)

6. data = pd.read\_csv(current\_dir + "/dataset/online-retail.csv")

7.

8. for column in data.select\_dtypes(include=['object']).columns:

9. data[column] = data[column].astype('string')

10.

11. # Print Number of missing values in each column

12. missing\_val = data.isnull().sum()

13. print("\nMissing values in the dataframe:")

14. print(missing\_val)

15.

16. # Drop Description column

17. data\_drop = data.drop(columns=['Description'])

18.

19. # remove rows with missing values in any column

20. data\_drop = data\_drop.dropna()

21.

22. # Print Number of missing values in each column after imputing CustomerID column

23. print("\nMissing values in the dataframe after removal: ")

24. missing\_val = data\_drop.isnull().sum()

25. print(missing\_val)

26.

27. # Print the number of duplicate rows in the dataframe

28. dup\_rows = data\_drop.duplicated().sum()

29. print("\nNumber of duplicate rows in the dataframe:", dup\_rows)

30.

31. # Remove duplicate rows from the dataframe

32. data\_drop\_dup = data\_drop.drop\_duplicates()

33.

34. # Print the number of duplicate rows in the dataframe after removing duplicates

35. dup\_rows = data\_drop\_dup.duplicated().sum()

36. print("\nNumber of duplicate rows in the dataframe after removing duplicates: ", dup\_rows)

37.

38. # Remove non-numeric columns

39. data\_drop\_dup\_num\_only = data\_drop.select\_dtypes(include=['number'])

40.

41. # Detect and print the outliers in the dataframe

42. Q1 = data\_drop\_dup\_num\_only.quantile(0.25)

43. Q3 = data\_drop\_dup\_num\_only.quantile(0.75)

44. IQR = Q3 - Q1

45. outliersNum = ((data\_drop\_dup\_num\_only < (Q1 - 1.5 \* IQR)) | (data\_drop\_dup\_num\_only > (Q3 + 1.5 \* IQR))).sum()

46. print("\nOutliers in the dataframe:")

47. print(outliersNum)

48.

49. # Remove the outliers from the dataframe

50. data\_no\_outliers = data\_drop\_dup\_num\_only[~((data\_drop\_dup\_num\_only < (Q1 - 1.5 \* IQR)) | (data\_drop\_dup\_num\_only > (Q3 + 1.5 \* IQR))).any(axis=1)]

51.

52. # Generate summary statistics of the dataframe

53. summary\_statistics = data\_no\_outliers.describe()

54. print("\nSummary statistics of the dataframe:")

55. print(summary\_statistics)

56.

57. # export data\_no\_outliers to a new csv file

58. data\_no\_outliers.to\_csv("cleaned\_dataset", index=False)

59.

60. # Plot histogram for numerical features and bar plot for categorical features

61. import matplotlib.pyplot as plt

62.

63. # Plot histogram for numerical features

64. data\_no\_outliers.hist()

65. plt.show()

66.

67. # Plot bar plot for categorical features

68. cat\_cols = data\_drop\_dup.select\_dtypes(include=['string']).columns

69. for column in cat\_cols:

70. plt.figure(figsize=(15, 10))

71. data\_drop\_dup[column].value\_counts().plot(kind='bar')

72. plt.title(column)

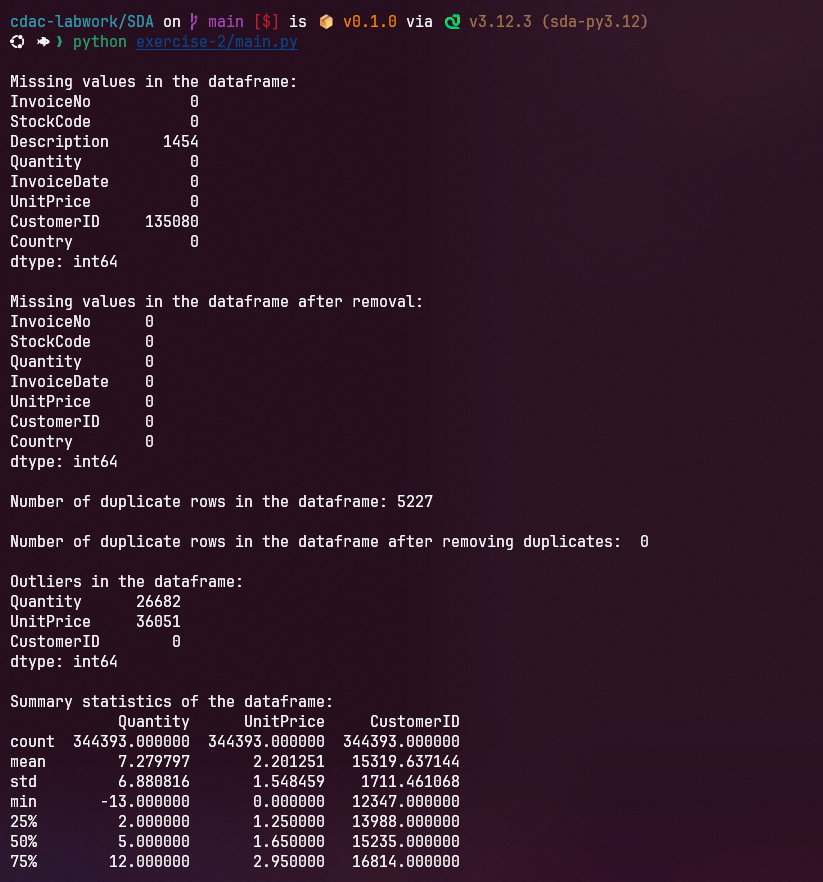
73. plt.show()

74.

**OUTPUT:**

A graph of different sizes and colors

Description automatically generated with medium confidence

****

* **EXERCISE-3**

**Objective:** Descriptive Statistics – Graphical Display

**Data Description:**

In 1798, H. Cavendish set out to determine the density of the earth relative to that of water, using a torsion balance. He conducted 29 very precise experiments assessing this relative density. A working assumption to begin with is that the values of his measurements were taken from a normal distribution centered around the true relative density. The data from all 29 experiments are given below (from Stigler, S., "Do Robust Estimators Work with Real Data?", *Annals of Statistics*, 5, 1977).

5.50 5.55 5.57 5.42 5.30 5.34

5.61 5.36 5.53 5.79 5.47 5.75

4.88 5.29 5.62 5.10 5.63 5.68

5.07 5.58 5.29 5.27 5.34 5.85

5.26 5.65 5.44 5.39 5.46

1. Enter the above data in column one & name it CAVEND.
2. Construct a boxplot of the data.
3. Construct a histogram with a suitable binwidth.
4. Based on the two plots above, does the data seem symmetric or skewed? Also, comment on the modality. Are there outliers in the data?
5. Guesstimate the sample mean & standard deviation from the histogram.
6. Calculate & record the basic summary statistics, such as the mean, standard deviation, quartiles, minimum, and maximum.
7. Find the proportion of the observations within plus or minus one standard deviation of the mean. Repeat for plus or minus two standard deviations. Compare with the rule of thumb from lecture.

**CODE:**

1. # Import necessary libraries

2. import numpy as np

3. import pandas as pd

4. import matplotlib.pyplot as plt

5. import seaborn as sns

6.

7. # Data: Cavendish's measurements

8. cavendish\_data = [

9. 5.50,

10. 5.55,

11. 5.57,

12. 5.42,

13. 5.30,

14. 5.34,

15. 5.61,

16. 5.36,

17. 5.53,

18. 5.79,

19. 5.47,

20. 5.75,

21. 4.88,

22. 5.29,

23. 5.62,

24. 5.10,

25. 5.63,

26. 5.68,

27. 5.07,

28. 5.58,

29. 5.29,

30. 5.27,

31. 5.34,

32. 5.85,

33. 5.26,

34. 5.65,

35. 5.44,

36. 5.39,

37. 5.46,

38. ]

39.

40. # Convert to a Pandas DataFrame for easy handling

41. df = pd.DataFrame(cavendish\_data, columns=["CAVEND"])

42.

43. # a. Enter data in column one and name it CAVEND

44. print("\n=== Step a: Data in 'CAVEND' Column ===")

45. print(df)

46.

47. # b. Construct a boxplot

48. plt.figure(figsize=(8, 5))

49. sns.boxplot(x=df["CAVEND"], color="skyblue")

50. plt.title("Boxplot of Cavendish Measurements")

51. plt.xlabel("CAVEND")

52. plt.grid(axis="x", linestyle="--", alpha=0.7)

53. plt.show()

54.

55. # c. Construct a histogram with a suitable binwidth

56. plt.figure(figsize=(8, 5))

57. plt.hist(df["CAVEND"], bins=8, color="blue", alpha=0.7, edgecolor="black")

58. plt.title("Histogram of Cavendish Measurements")

59. plt.xlabel("CAVEND")

60. plt.ylabel("Frequency")

61. plt.grid(axis="y", linestyle="--", alpha=0.7)

62. plt.show()

63.

64. # d. Check for symmetry, skewness, modality, and outliers

65. print("\n=== Step d: Data Analysis from Plots ===")

66. boxplot\_analysis = """

67. The boxplot indicates there may be some outliers on the lower side (values < 5.0).

68. The histogram appears roughly symmetric but slightly skewed to the left.

69. The data is unimodal, as there's one clear peak in the histogram.

70. """

71. print(boxplot\_analysis)

72.

73. # e. Guesstimate mean and standard deviation from the histogram

74. print("\n=== Step e: Guesstimates ===")

75. print(

76. "From the histogram, the mean appears to be around 5.45, and the standard deviation around 0.15."

77. )

78.

79. # f. Calculate basic summary statistics

80. mean = df["CAVEND"].mean()

81. std\_dev = df["CAVEND"].std()

82. quartiles = df["CAVEND"].quantile([0.25, 0.5, 0.75])

83. minimum = df["CAVEND"].min()

84. maximum = df["CAVEND"].max()

85.

86. print("\n=== Step f: Basic Summary Statistics ===")

87. print(f"Mean: {mean:.2f}")

88. print(f"Standard Deviation: {std\_dev:.2f}")

89. print(f"Quartiles:\n{quartiles}")

90. print(f"Minimum: {minimum}")

91. print(f"Maximum: {maximum}")

92.

93. # g. Proportions within 1 and 2 standard deviations

94. within\_1\_std = df[(df["CAVEND"] >= mean - std\_dev) & (df["CAVEND"] <= mean + std\_dev)]

95. within\_2\_std = df[

96. (df["CAVEND"] >= mean - 2 \* std\_dev) & (df["CAVEND"] <= mean + 2 \* std\_dev)

97. ]

98.

99. proportion\_1\_std = len(within\_1\_std) / len(df)

100. proportion\_2\_std = len(within\_2\_std) / len(df)

101.

102. print("\n=== Step g: Proportions within Standard Deviations ===")

103. print(f"Proportion within ±1 SD: {proportion\_1\_std:.2%}")

104. print(f"Proportion within ±2 SD: {proportion\_2\_std:.2%}")

105.

106. # Rule of thumb comparison

107. rule\_of\_thumb = """

108. The empirical rule states:

109. - About 68% of the data should fall within ±1 standard deviation.

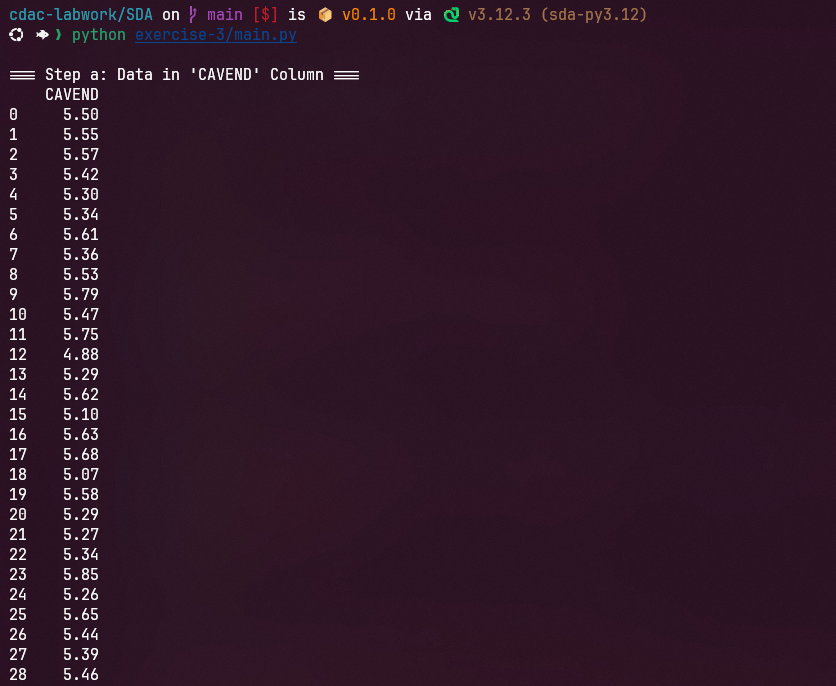
110. - About 95% of the data should fall within ±2 standard deviations.

111. """

112. print(rule\_of\_thumb)

113.

**OUTPUT:**



A graph with a rectangle

Description automatically generated

A graph with blue squares

Description automatically generated

A screenshot of a computer program

Description automatically generated