| Experiment No. 11 |
| --- |
| Name : Sumit Yesade |
|  |
| To study Exploratory Data Analysis using Pandas Library |
| Date of Performance: |
| Date of Submission: |

**Experiment No. 11**

**Title:** Program to demonstrate data frame creation and Manipulation using Pandas

**Aim:** To study and implement EDA using Pandas

**Objective:** To introduce Pandas package for python

**Theory:**

**Pandas** is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.

**`read\_csv`** is a function that reads data from a CSV file into a format that can be easily used for analysis in programming.

**syntax:-** *df = pd.read\_csv('filename.csv')*

**`df.head`** is a command that shows the first few rows of a table or dataset in Python. **syntax:***- df.head(n)*

**`df.shape`** is a property in pandas that returns a tuple representing the dimensions of a DataFrame. The tuple contains two values: the number of rows and the number of columns in the DataFrame.

**syntax:***- print(df.shape) # Output: (rows, columns)*

**`df.info()`** is a method in pandas used to provide a concise summary of a DataFrame's structure and contents. It displays information such as the data types of each column, the number of non-null values, and memory usage.

**syntax:***- df.info()*

**`df.describe()`** is a method in pandas used to generate descriptive statistics summarizing the central tendency, dispersion, and shape of a DataFrame's distribution.

**syntax:***- df.describe()*

**`df.columns.tolist()`** is a method in pandas used to convert the column labels of a DataFrame into a list.

**syntax:***- df.columns.tolist()*

**`df.isnull()`** is a method in pandas used to check for missing (null) values in a DataFrame. It returns a DataFrame of the same shape as the original, where each element is either True or False indicating whether the corresponding element in the original DataFrame is missing or not.

**syntax:***- df.isnull()*

**`df.isnull().sum()`** is a pandas method used to calculate the sum of missing (null) values for each column in a DataFrame. It returns a Series where the index represents column names and the values represent the count of missing values in each column.

**syntax:***- df.isnull().sum()*

**`df.nunique()`** is a method in pandas used to calculate the number of unique values for each column in a DataFrame. It returns a Series where the index represents column names and the values represent the count of unique values in each column.

**syntax:***- df.nunique()*

**`df.value\_counts()`** is a method in pandas used to count the occurrences of unique values in a Series or DataFrame column. It returns a new Series containing counts of unique values, sorted in descending order by default.

**syntax:***- df.value\_counts()*

**swarmplot** is a function provided by the Seaborn library in Python for visualizing categorical data along with their distribution. It displays individual data points along the categorical axis, avoiding overlapping points by adjusting their positions. This plot is useful for visualizing the distribution of data points within different categories.

**syntax:***- sns.swarmplot(x='x\_column', y='y\_column', data=df)*

**pairplot** is a function provided by the Seaborn library in Python for creating a grid of scatterplots to show the relationship between pairs of variables in a dataset. It displays pairwise relationships between different numerical variables along with the distribution of each variable on the diagonal.

**syntax:***-sns.pairplot(df)*

**`df.min()`** is a pandas method used to find the minimum value for each column in a DataFrame. It returns a Series where the index represents column names and the values represent the minimum value in each column.

**syntax:***-df.min()*

**`df.max()`** is a pandas method used to find the maximum value for each column in a DataFrame. It returns a Series where the index represents column names and the values represent the maximum value in each column.

**syntax:***-df.max()*

**`df.std()`** is a pandas method used to calculate the standard deviation for each column in a DataFrame. It returns a Series where the index represents column names and the values represent the standard deviation of each column's values.

**syntax:***-df.std()*

**`df.var()`** is a pandas method used to calculate the variance for each column in a DataFrame. It returns a Series where the index represents column names and the values represent the variance of each column's values.

**syntax:***-df.std()*

**`df.mean()`** is a pandas method used to calculate the mean (average) for each column in a DataFrame. It returns a Series where the index represents column names and the values represent the mean of each column's values.

**syntax:***-df.mean()*

**`df.median()`** is a pandas method used to calculate the median for each column in a DataFrame. It returns a Series where the index represents column names and the values represent the median of each column's values.

**syntax:***-df.median()*

**`df.mode()`** is a pandas method used to calculate the mode (most frequent value) for each column in a DataFrame. It returns a DataFrame where each row represents a column, and the mode(s) for that column are listed along with their respective frequencies.

**syntax:***-df.mode()*

#importing library

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings as wr

wr.filterwarnings('ignore')

df = pd.read\_csv("dataset11.csv")

print(df.head())

# by default head is first 5 rows

fixed acidity volatile acidity citric acid residual sugar chlorides \ 0 7.4 0.70 0.00 1.9 0.076 1 7.8 0.88 0.00 2.6 0.098 2 7.8 0.76 0.04 2.3 0.092 3 11.2 0.28 0.56 1.9 0.075 4 7.4 0.70 0.00 1.9 0.076

free sulfur dioxide total sulfur dioxide density pH sulphates \ 0 11.0 34.0 0.9978 3.51 0.56 1 25.0 67.0 0.9968 3.20 0.68 2 15.0 54.0 0.9970 3.26 0.65 3 17.0 60.0 0.9980 3.16 0.58 4 11.0 34.0 0.9978 3.51 0.56

alcohol quality

0 9.4 5

1 9.8 5

2 9.8 5

3 9.8 6

4 9.4 5

df.shape

# will give the number of rows and column in your dataset

(1599, 12)

df.info()

# basic info display hoga like datatype,null etc

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1599 entries, 0 to 1598

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 fixed acidity 1599 non-null float64

1 volatile acidity 1599 non-null float64

2 citric acid 1599 non-null float64

3 residual sugar 1599 non-null float64

4 chlorides 1599 non-null float64

5 free sulfur dioxide 1599 non-null float64

6 total sulfur dioxide 1599 non-null float64

7 density 1599 non-null float64

8 pH 1599 non-null float64

9 sulphates 1599 non-null float64

10 alcohol 1599 non-null float64

11 quality 1599 non-null int64

dtypes: float64(11), int64(1)

memory usage: 150.0 KB

df.describe()

# max mean count etc

**fixed acidity**

**volatile acidity**

**citric acid**

**residual**

**sugarchlorides**

**free**

**sulfur dioxide**

**s**

**di**

**count** 1599.000000 1599.000000 1599.000000 1599.000000 1599.000000 1599.000000 1599.0 **mean** 8.319637 0.527821 0.270976 2.538806 0.087467 15.874922 46.4 **std** 1.741096 0.179060 0.194801 1.409928 0.047065 10.460157 32.8 **min** 4.600000 0.120000 0.000000 0.900000 0.012000 1.000000 6.0 **25%** 7.100000 0.390000 0.090000 1.900000 0.070000 7.000000 22.0 **50%** 7.900000 0.520000 0.260000 2.200000 0.079000 14.000000 38.0 **75%** 9.200000 0.640000 0.420000 2.600000 0.090000 21.000000 62.0 **max** 15.900000 1.580000 1.000000 15.500000 0.611000 72.000000 289.0

df.columns.tolist()

# will convert the columns to list..

# to change the column name use list operation

['fixed acidity',

'volatile acidity',

'citric acid',

'residual sugar',

'chlorides',

'free sulfur dioxide',

'total sulfur dioxide',

'density',

'pH',

'sulphates',

'alcohol',

'quality']

df.isnull()

**fixed acidity**

**volatile acidity**

**citric acid**

**residual**

**sugarchlorides**

**free**

**sulfur dioxide**

**total**

**sulfur dioxide**

**density pH s**

**0** False False False False False False False False False **1** False False False False False False False False False **2** False False False False False False False False False **3** False False False False False False False False False **4** False False False False False False False False False **...** ... ... ... ... ... ... ... ... ...

**1594** False False False False False False False False False **1595** False False False False False False False False False **1596** False False False False False False False False False **1597** False False False False False False False False False **1598** False False False False False False False False False

df.isnull().sum()

# sum dega total no of mull values

fixed acidity 0

volatile acidity 0

citric acid 0

residual sugar 0

chlorides 0

free sulfur dioxide 0

total sulfur dioxide 0

density 0

pH 0

sulphates 0

alcohol 0

quality 0

dtype: int64

df.nunique()

# no of unique values not repeated

fixed acidity 96

volatile acidity 143

citric acid 80

residual sugar 91

chlorides 153

free sulfur dioxide 60

total sulfur dioxide 144

density 436

pH 89

sulphates 96

alcohol 65

quality 6

dtype: int64

quality\_counts=df['quality'].value\_counts()

plt.figure(figsize=(8,6))

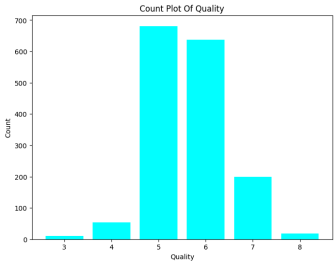
plt.bar(quality\_counts.index,quality\_counts,color='cyan')

plt.title('Count Plot Of Quality')

plt.xlabel('Quality')

plt.ylabel('Count')

plt.show()



# the outlyer is the point at the top excluded

plt.figure(figsize=(10,8))

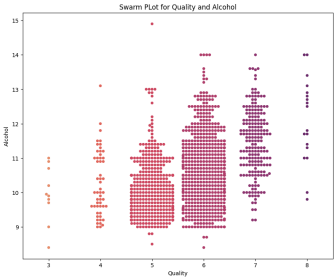
sns.swarmplot(x="quality", y="alcohol", data=df, palette='flare')

plt.title('Swarm PLot for Quality and Alcohol')

plt.xlabel('Quality')

plt.ylabel('Alcohol')

plt.show()



sns.set\_palette('icefire')

plt.figure(figsize=(10,6))

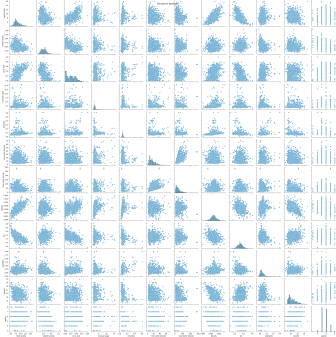
sns.pairplot(df)

plt.suptitle('Pair plot for dataframe') plt.show()

https://colab.research.google.com/drive/1ahi-k9pVwkaVCy3z7Svc7pUteYMCORa-#scrollTo=SSr7zkO0h4ou&printMode=true 4/6

3/27/24, 1:23 PM exp11.ipynb - Colaboratory

<Figure size 1000x600 with 0 Axes>



print(df.min())

fixed acidity 4.60000

volatile acidity 0.12000

citric acid 0.00000

residual sugar 0.90000

chlorides 0.01200

free sulfur dioxide 1.00000

total sulfur dioxide 6.00000

density 0.99007

https://colab.research.google.com/drive/1ahi-k9pVwkaVCy3z7Svc7pUteYMCORa-#scrollTo=SSr7zkO0h4ou&printMode=true 5/6

3/27/24, 1:23 PM exp11.ipynb - Colaboratory

pH 2.74000 sulphates 0.33000 alcohol 8.40000 quality 3.00000 dtype: float64

print(df.max())

fixed acidity 15.90000 volatile acidity 1.58000 citric acid 1.00000 residual sugar 15.50000 chlorides 0.61100 free sulfur dioxide 72.00000 total sulfur dioxide 289.00000 density 1.00369 pH 4.01000 sulphates 2.00000 alcohol 14.90000 quality 8.00000 dtype: float64

print(df.std())

fixed acidity 1.741096 volatile acidity 0.179060 citric acid 0.194801 residual sugar 1.409928 chlorides 0.047065 free sulfur dioxide 10.460157 total sulfur dioxide 32.895324 density 0.001887 pH 0.154386 sulphates 0.169507 alcohol 1.065668 quality 0.807569 dtype: float64

print(df.var())

fixed acidity 3.031416 volatile acidity 0.032062 citric acid 0.037947 residual sugar 1.987897 chlorides 0.002215 free sulfur dioxide 109.414884 total sulfur dioxide 1082.102373 density 0.000004 pH 0.023835 sulphates 0.028733 alcohol 1.135647 quality 0.652168 dtype: float64

print(df.mean())

fixed acidity 8.319637 volatile acidity 0.527821 citric acid 0.270976 residual sugar 2.538806 chlorides 0.087467 free sulfur dioxide 15.874922 total sulfur dioxide 46.467792

**Conclusion:** Dataframes have been created and manipulated using Pandas