05/03

WEEK - 00

TO-DO EXERCISES

① Initializing values directly into DataFrame - five rows of data with column heading as USN, Name, Marks.

```
import pandas as pd
data = { "USN" : ["234", "235", "236", "237", "238"],
         "Name" : ["Asha", "Cathy", "Rosa", "Amy", "Gina"],
         "Marks" : [85, 90, 78, 92, 88]
       }
df = pd.DataFrame(data)
print(df)
```

o/p

	USN	Name	Marks
0	234	Asha	85
1	235	Cathy	90
2	236	Rosa	78
3	237	Amy	92
4	238	Gina	88

② Importing datasets from sklearn.datasets

```
from sklearn.datasets import load_diabetes
import pandas as pd
diabetes = load_diabetes()
df_diabetes = pd.DataFrame(diabetes.data, columns = diabetes.feature_names)
print(df_diabetes)
```

o/p

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.03807	0.50680	0.061691	0.0218	-0.044	-0.034	-0.043		-0.01764	
1	-0.001882	-0.04472	-0.051474		-0.05					
2	0.085299	0.050680								
3										
4										

③ Import datasets from a specific .csv file

```
import pandas as pd
df_csv = pd.read_csv("/sample_sales_data.csv")
print(df_csv.head())
```

o/p

	Product	Quantity	Price	Sales	Region
0	Laptop	5	1000	5000	North
1	Mouse	15	20	300	West
2	Keyboard	10	50	500	East
3	Monitor	8	200	1600	South
4	Laptop	12	950	11400	North

④ Downloading datasets from existing dataset repository like Kaggle..

```
import pandas as pd
df_kaggle = pd.read_csv("/Dataset of Diabetes.csv")
print(df_kaggle.head())
```

o/p

	ID	No_Patient	Gender	AGE	Urea	Cr
0	502	17975	F	50	4.7	46
1	735	34221	M	26	4.5	62
2	420	47975	F	50	4.7	

	BMI	CLASS
0	24.0	N
1	23.0	N
2	24.0	N
3	24.0	N
4	21.0	N

TO - DO

Stock Market Data Analysis

1. ["HDFCBANK.NS", "ICICIBANK.NS", "KOTAKBANK.NS"]
2. start date and end date.
3. Plot the closing price and daily return for all three bank.

```
import yfinance as yf
import panda as pd
import pmatplotlib.pyplot as plt
tickers = ["HDFCBANK.NS", "ICICIBANK.NS", "KOTAKBANK.NS"]
data = yf.download(tickers, start = "2024-01-01", end = '2024-12-30",
    group-by = 'ticker')
print ("First 5 rows of the dataset")
print ( data.head (1)
print ("\n Shape of dataset")
print (data.shape)
print ("\n Column names")
print ( data.columns)
hdfc_data = data ['HDFCBANK.NS']
print ("\n Summary statistics for HDFC Industries")
print (hdfc_data.describe ())
hdfc_data ['Daily Return'] = hdfc_data ['Close'].pct_change ()
plt.figure ( figsize = (12, 6))
plt.subplot (2, 1, 1)
hdfc_data ['Daily_Return'] = hdfc_data.plot (title = "HDFC Industries - Daily
  Returns", color - orang)
plt.tight_layout ()
plt.show ()
```

WEEK 1

① To load .csv file into dataframe

```
import pandas as pd
df = pd.read_csv('/content/housing.csv')
print(df.head())
print(df.columns)
print(df.info)
```

② To display information of all columns

```
print(df.columns())
```

	population	households	median_income	median_house_value	ocean_proximity
0	322.0	126.0	322.0		
1	2401.0	1138.0	8.304		
2	496.0		8.754		
3	558.0	219.0	5.6471		
4	565.0	219.0	3.8462		

③ To display statistical information of all numerical

```
print(df.info)
```

④ To display the count of unique label for "Ocean Proximity" column

```
print(df.counter('ocean_proximity'))
```

⑤ To display which attributes in a dataset have missing values.

```
print(df.isnull())
```

	ocean_proximity
0	NEAR BAY
1	NEAR BAY
2	INLAND
⋮	

05-03-24

For Diabetes dataset, apply data preprocessing technique —

```
import pandas as pd
import numpy as np
import os
import sklearn.preprocessing import LabelEncoder, MinMaxScaler, StandardScaler

diabetes_df = pd.read_csv("/content/Dataset_of_Diabetes.csv")

# Check for missing value
print("Missing values in Diabetes dataset :\n", diabetes_df.isnull().sum())

# Handle missing values with mean
numeric_cols = diabetes_df.select_dtypes(include=np.number).columns
diabetes_df[numeric_cols] = diabetes_df[numeric_cols].fillna(diabetes_df[numeric_cols].
                                                             mean())

# To fill missing categorical values with mode
for col in diabetes_df.select_dtypes(include=['object']).columns:
    diabetes_df[col].fillna(diabetes_df[col].mode()[0], inplace=True)

# Identify categorical values and encode them.
categorical_cols = ['Gender', 'CLASS']
label_enc = LabelEncoder()
for col in categorical_cols:
    diabetes_df[col] = label_enc.fit_transform(diabetes_df[col])

# Apply min max scaling
min_max_scaler = MinMaxScaler()
diabetes_df_scaled = pd.DataFrame(min_max_scaler.fit_transform(diabetes_df),
                                  columns=diabetes_df.columns)
```

Apply standardization

std_scaler = StandardScaler()

① diabetes_df_standardized = pd.DataFrame(std_scaler.fit_transform(diabetes_df), columns = diabetes_df.columns)

Save the preprocessed datasets

diabetes_df_scaled.to_csv("/content/diabetes_preprocessed.csv", index = False)

os.makedirs("content", exist_ok = True)

diabetes_df_standardized.to_csv("/content/diabetes_standardized.csv", index = False)

print("Diabetes dataset preprocessing completed. Preprocessed datasets saved.")

②

Q1. Which columns in the dataset had missing values? How did you handle them?

- Numeric columns such as glucose levels, blood pressure, insulin.
- Categorical columns such as Gender and CLASS

To handle numeric data, mean of the respective column was used.

categorical data, mode

Q2. Which categorical columns did you identify in the dataset? How did you encode?

Gender and CLASS are the two categorical columns.

③ Encoding method - Label encoding Male → 0, Female → 1

Q3. What is the difference between min-max scaling and standardization?

④ • Min Max Scaling - transforms data to fixed range [0,1]

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

⑤ • Standardization - transforms data so that it has a mean of 0 and standard deviation of 1

$$x' = \frac{x - \mu}{\sigma}$$