Data Profiling:

```
df.info()
df.describe()
df.isnull().sum()
df.nunique()
df.duplicated()
df.duplicated('Column_name')
df.drop_duplicates(subset=[Column_name'])
```

Data profiling is the process of examining, analyzing, and creating useful summaries of data.

In [3]:
```
# Data Profiling: Pandas
```

In [4]:
```
import pandas as pd
```

In [5]:
```
df=pd.read_csv('/Users/pragatigupta/Documents/AI And ML/Linkedin Post/
```

In [6]:
```
#df.head()
# to see the full dataset '=
pd.set_option("display.max_rows",None)
df
```

Out[6]:

	ID	Student_ID	Gender	AGE	Score	CLASS
0	1.0	17975	F	15	6.7	y
1	2.0	34221	M	16	6.5	y
2	3.0	47975	F	17	5.5	y
3	4.0	87656	F	14	6.8	y
4	5.0	34223	M	15	7.1	y
5	6.0	34224	F	16	2.3	N
6	7.0	34225	F	17	2.0	n
7	8.0	34227	M	15	4.7	N
8	9.0	34229	M	16	2.6	N
9	10.0	34230	F	17	6.7	y
10	11.0	34231	F	14	6.5	Y
11	NaN	87656	F	14	6.8	y
12	2.0	34221	M	16	6.5	y
13	14.0	34224	F	16	2.3	N

14	15.0	34235	F	14	3.5	N
15	16.0	34236	M	15	5.5	y
16	17.0	34237	F	16	5.9	y
17	18.0	87654	F	17	6.7	y
18	19.0	34238	F	15	6.5	y
19	20.0	34239	F	16	5.5	Y
20	21.0	Null	F	17	6.8	Y
21	22.0	12744	F	14	7.1	y
22	23.0	34302	F	15	6.5	y
23	24.0	NaN	M	16	5.5	Y
24	25.0	34242	F	17	6.8	y
25	26.0	46675	F	15	6.7	y
26	27.0	45566	M	16	6.5	y
27	28.0	34309	M	17	5.5	y
28	29.0	87664	M	14	6.8	Y
29	30.0	34245	F	15	7.1	y

```
# DATA TYPES
```

In [7]:
```
info=df.info()
info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   ID          29 non-null     float64
 1   Student_ID  29 non-null     object 
 2   Gender      30 non-null     object 
 3   AGE         30 non-null     int64  
 4   Score       30 non-null     float64
 5   CLASS       30 non-null     object 
dtypes: float64(2), int64(1), object(3)
memory usage: 1.5+ KB
```

In [8]:
```
# Convert the float ID column to int
#df['ID'] = df['ID'].astype(int) >>>>>>>>> wll give error bcz we havnt



# Replace NaN values with a specific integer (e.g., 0)
df['ID'] = df['ID'].fillna(0).astype(int)
info=df.info()
info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   ID          30 non-null     int64  
 1   Student_ID  29 non-null     object 
 2   Gender      30 non-null     object 
 3   AGE         30 non-null     int64  
 4   Score       30 non-null     float64
 5   CLASS       30 non-null     object 
dtypes: float64(1), int64(2), object(3)
memory usage: 1.5+ KB
```

In [9]:
```
# Get basic statistics
summary = df.describe()
summary
```

Out[9]:

	ID	AGE	Score
count	30.000000	30.000000	30.000000
mean	14.733333	15.566667	5.730000
std	9.530110	1.072648	1.578334
min	0.000000	14.000000	2.000000
25%	6.250000	15.000000	5.500000
50%	15.500000	16.000000	6.500000
75%	22.750000	16.000000	6.775000
max	30.000000	17.000000	7.100000

In [10]:
```
# Count unique values for each column
unique_counts = df.nunique()
unique_counts
```

Out[10]:
```
ID            29
Student_ID    26
Gender         2
AGE            4
Score         11
CLASS          8
dtype: int64
```

In [11]:
```
# Check for missing values
missing_values = df.isnull().sum()
missing_values
```

Out[11]:
```
ID            0
Student_ID    1
Gender        0
AGE           0
Score         0
CLASS         0
dtype: int64
```

In [12]:
```
# Check for duplicate rows
duplicates = df[df.duplicated()]
duplicates
```

Out[12]:

	ID	Student_ID	Gender	AGE	Score	CLASS
12	2	34221	M	16	6.5	y

In [13]:
```
# Check for duplicate rows
duplicates = df[df['Student_ID'].duplicated()]
duplicates
```

Out[13]:

	ID	Student_ID	Gender	AGE	Score	CLASS
11	0	87656	F	14	6.8	y
12	2	34221	M	16	6.5	y
13	14	34224	F	16	2.3	N

In [14]:
```
# Drop duplicates based on the ''Student ID'' column
df_no_duplicates_Student_ID = df.drop_duplicates(subset=['Student_ID'])
print(df_no_duplicates_Student_ID)
```

```
    ID Student_ID Gender  AGE  Score CLASS
0    1      17975      F   15    6.7     y
1    2      34221      M   16    6.5     y
2    3      47975      F   17    5.5     y
3    4      87656      F   14    6.8     y
4    5      34223      M   15    7.1     y
5    6      34224      F   16    2.3     N
6    7      34225      F   17    2.0     n
7    8      34227      M   15    4.7     N
8    9      34229      M   16    2.6    N
9   10      34230      F   17    6.7     y
10  11      34231      F   14    6.5     Y
14  15      34235      F   14    3.5     N
15  16      34236      M   15    5.5     y
16  17      34237      F   16    5.9     y
17  18      87654      F   17    6.7     y
18  19      34238      F   15    6.5     y
19  20      34239      F   16    5.5     Y
20  21       Null      F   17    6.8    Y
21  22      12744      F   14    7.1     y
22  23      34302      F   15    6.5     y
23  24        NaN      M   16    5.5     Y
24  25      34242      F   17    6.8     y
25  26      46675      F   15    6.7     y
26  27      45566      M   16    6.5     y
27  28      34309      M   17    5.5     y
28  29      87664      M   14    6.8     Y
29  30      34245      F   15    7.1     y
```

Pyspark

In [15]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, isnan
```

```
Data profiling is the process of examining, analyzing, and creating
useful summaries of data.
```

In [16]:
```
# Initialize a Spark session
spark = SparkSession.builder.appName("DataProfiling").getOrCreate()
```

```
23/09/28 12:41:35 WARN Utils: Your hostname, Pragatis-MacBook-Air.loc
al resolves to a loopback address: 127.0.0.1; using 10.0.0.145 instea
d (on interface en0)
23/09/28 12:41:35 WARN Utils: Set SPARK_LOCAL_IP if you need to bind
to another address
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.
properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
23/09/28 12:41:37 WARN NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applica
ble
```

In [17]:
```
DataFrame
agatigupta/Documents/AI And ML/Linkedin Post/Student Dataset/Student_Sc
```

In [18]:
```
df.head(5)
```

Out[18]:
```
[Row(ID=1, Student_ID='17975', Gender='F', AGE=15, Score=6.7, CLASS='
y '),
 Row(ID=2, Student_ID='34221', Gender='M', AGE=16, Score=6.5, CLASS='
y '),
 Row(ID=3, Student_ID='47975', Gender='F', AGE=17, Score=5.5, CLASS='
y '),
 Row(ID=4, Student_ID='87656', Gender='F', AGE=14, Score=6.8, CLASS='
y '),
 Row(ID=5, Student_ID='34223', Gender='M', AGE=15, Score=7.1, CLASS='
y ')]
```

In [19]:
```
# Get basic statistics
summary = df.describe()
summary.show()
```

```
+-------+-----------------+-----------------+------+--------------
---+-----------------+-----+
|summary|               ID|       Student_ID|Gender|
AGE|            Score|CLASS|
+-------+-----------------+-----------------+------+--------------
---+-----------------+-----+
|  count|               29|               29|    30|
30|               30|   30|
|   mean|15.241379310344827| 41860.82142857143|  null|15.566666666666
666|             5.73| null|
| stddev| 9.276141332987368|20171.267078682085|  null| 1.072648457158
112|1.5783339098665028| null|
|    min|                1|            12744|     F|
14|              2.0|    Y|
|    max|               30|             Null|     M|
17|              7.1|   y |
+-------+-----------------+-----------------+------+--------------
---+-----------------+-----+
```

In [20]:
```
# Get data types and missing values
info = df.printSchema()
info
```

```
root
 |-- ID: integer (nullable = true)
 |-- Student_ID: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- AGE: integer (nullable = true)
 |-- Score: double (nullable = true)
 |-- CLASS: string (nullable = true)
```

In [21]:
```
# Find missing values in the specified column
missing_values = df.filter(col("ID").isNull() | isnan(col("ID")))
missing_values.show()
```

```
+----+----------+------+---+-----+-----+
|  ID|Student_ID|Gender|AGE|Score|CLASS|
+----+----------+------+---+-----+-----+
|null|     87656|     F| 14|  6.8|   y |
+----+----------+------+---+-----+-----+
```

In [22]:
```
null_count = df.filter(col("Student_ID").isNull()).count()
null_count
```

Out[22]: 1

In [23]:
```
# Check for duplicate rows
duplicates = df.groupBy(df.ID).count().filter("count > 1")
duplicates.show()
```

```
+---+-----+
| ID|count|
+---+-----+
|  2|    2|
+---+-----+
```

In [25]:
```
# Print results
summary.show()
info
missing_values.show
duplicates.show()
```

```
+-------+-----------------+-----------------+------+--------------
---+-----------------+-----+
|summary|               ID|       Student_ID|Gender|
AGE|            Score|CLASS|
+-------+-----------------+-----------------+------+--------------
---+-----------------+-----+
|  count|               29|               29|    30|
30|               30|   30|
|   mean|15.241379310344827| 41860.82142857143|  null|15.566666666666
666|             5.73| null|
| stddev| 9.276141332987368|20171.267078682085|  null| 1.072648457158
112|1.5783339098665028| null|
|    min|                1|            12744|     F|
14|              2.0|    Y|
|    max|               30|             Null|     M|
17|              7.1|   y |
+-------+-----------------+-----------------+------+--------------
---+-----------------+-----+

+---+-----+
| ID|count|
+---+-----+
|  2|    2|
+---+-----+
```

SQL

In [39]:
```
#!pip install sqlite3
```

In [40]:
```
import pandas as pd
from pandasql import sqldf
```

In [42]:
```
data=pd.read_csv('/Users/pragatigupta/Documents/AI And ML/Linkedin Pos
```

```
df = sqldf("SELECT * FROM data");
df.head()
```

In [44]:
```
# Find the data types of columns in the DataFrame
column_data_types = df.dtypes
# Print or display the data types
print(column_data_types)
```

```
[('ID', 'int'), ('Student_ID', 'string'), ('Gender', 'string'), ('AG
E', 'int'), ('Score', 'double'), ('CLASS', 'string')]
```

```
# Total Count
Total_IDs = sqldf("SELECT count() As Total_IDs From df");
Total_IDs
```

```
#Duplicates
Total_IDs = sqldf("SELECT count() As Total_IDs From df Group By ID
");
Total_IDs
```

In [ ]:
```
Score_Avg = sqldf("SELECT Avg(Score) AS Score_Avg From df");
Score_Avg
```

In [ ]:
```
missing_count=sqldf("SELECT ID, COUNT(*) AS missing_count FROM df GROU
missing_count
```

In [ ]:
```
duplicate_count=sqldf("SELECT ID, COUNT(*) AS duplicate_count FROM df
duplicate_count
```