NAME : Amaraiah.C

REGISTER NO. : 2411021240002

CLASS : BCA "A" sec

SUBJECT : INTRODUCTION TO DATA SCIENCE

GUTHUB LINK : https://github.com/Amaraiah11/IDS-ASSIGNMENT

What is Data Science?

Data Science is an interdisciplinary field that combines statistics, mathematics, programming, domain expertise, and machine learning to extract meaningful insights from data. It involves collecting, cleaning, analysing, visualizing, and interpreting data to help in decision-making and predictions.

Key Components of Data Science

Data Collection – Gathering structured and unstructured data from various sources (databases, APIs, web scraping, etc.).

Data Cleaning & Processing – Removing errors, handling missing values, and preparing data for analysis.

Exploratory Data Analysis (EDA) – Understanding patterns, trends, and relationships in data using visualization techniques.

Machine Learning (ML) & AI – Using algorithms to train models for predictions and automation.

Data Visualization – Representing data insights using charts, graphs, and dashboards.

Big Data & Cloud Computing – Working with big data by using tools such as Hadoop, Spark, AWS, or Google Cloud.

Applications of Data Science

Healthcare – Prediction of diseases, medical image analysis, and personalized medicines

Finance – Detection of fraud in transactions, stock market prediction, and risk analysis

E-commerce – Customer recommendations, demand forecasting

Social Media – Sentiment analysis and user behaviour analysis

Autonomous Vehicles – Image recognition, path planning.

Key components and the CRISP-DM process.

CRISP-DM Process:

CRISP-DM (Cross-Industry Standard Process for Data Mining) is a widely used framework for data science projects. It consists of six phases:

1**. Business Understanding**

- Define project objectives and requirements

- Identify key stakeholders and their needs

- Determine the scope and timeline of the project

2. Data Understanding

- Collect and document data sources and metadata

- Explore and visualize data to understand its structure and quality

- Identify data quality issues and develop a plan to address them

3. Data Preparation

- Clean and preprocess data by handling missing values, outliers, and data transformations

- Integrate data from multiple sources and formats

- Develop a data pipeline to support repeatable and scalable data processing

4. Modelling

- Select and apply machine learning algorithms or statistical models to the prepared data

- Train and evaluate models using techniques such as cross-validation and hyperparameter tuning

- Compare and select the best-performing model

5. Evaluation

- Assess the performance of the selected model using metrics such as accuracy, precision, and recall

- Evaluate the model's interpretability and explainability

- Identify potential biases and limitations of the model

6. Deployment

- Deploy the model in a production-ready environment

- Develop a plan for ongoing model maintenance, monitoring, and updates

- Communicate results and insights to stakeholders and support decision-making

By following the CRISP-DM process, data scientists can ensure that their projects are well-structured, efficient, and effective in delivering valuable insights and business outcomes.

CRISP-DM framework is applied in solving real-world problems:

1. Business Understanding: Define the problem, identify key stakeholders, and determine the project's objectives and scope.

2. Data Understanding: Collect and analyse data to understand trends, patterns, and relationships.

3. Data Preparation: Clean, transform, and prepare data for modelling.

4. Modelling: Apply machine learning algorithms to solve the problem.

5. Evaluation: Assess the model's performance using relevant metrics.

6. Deployment: Implement the model in a production-ready environment and monitor its performance.

Predicting Customer Churn:

- Identify customers at risk of churning

- Analyze customer information, transaction history, and service usage

- Use machine learning models like logistic regression, decision trees, or neural networks

- Evaluate the model's accuracy using precision, recall, and F1-score

Movie Recommendation System:

- Increase user engagement by providing personalized movie recommendations

- Gather ratings, watch history, and genre preferences of users

- Apply collaborative filtering or content-based filtering algorithms

- Evaluate the model's performance using metrics like Mean Absolute Error (MAE) and precision-recall

What is the main business objective of the Netflix Recommendation System?

Netflix Recommendation System:

Overview:

A complex system utilizing machine learning and data processing to provide personalized recommendations.

Architecture:

1. Data Ingestion: Collects user interaction data and content metadata.

2. Data Processing: Processes and transforms data.

3. Model Training: Trains machine learning models.

4. Model Serving: Deploys trained models.

5. Recommendation Generation: Combines model outputs.

Algorithms and Techniques:

1. Collaborative Filtering (CF)

2. Content-Based Filtering (CBF)

3. Matrix Factorization

4. Neural Networks

5. Natural Language Processing (NLP)

Data Storage and Processing:

1. Hadoop

2. Spark

3. NoSQL databases (Cassandra, MongoDB)

4. Cloud infrastructure (AWS)

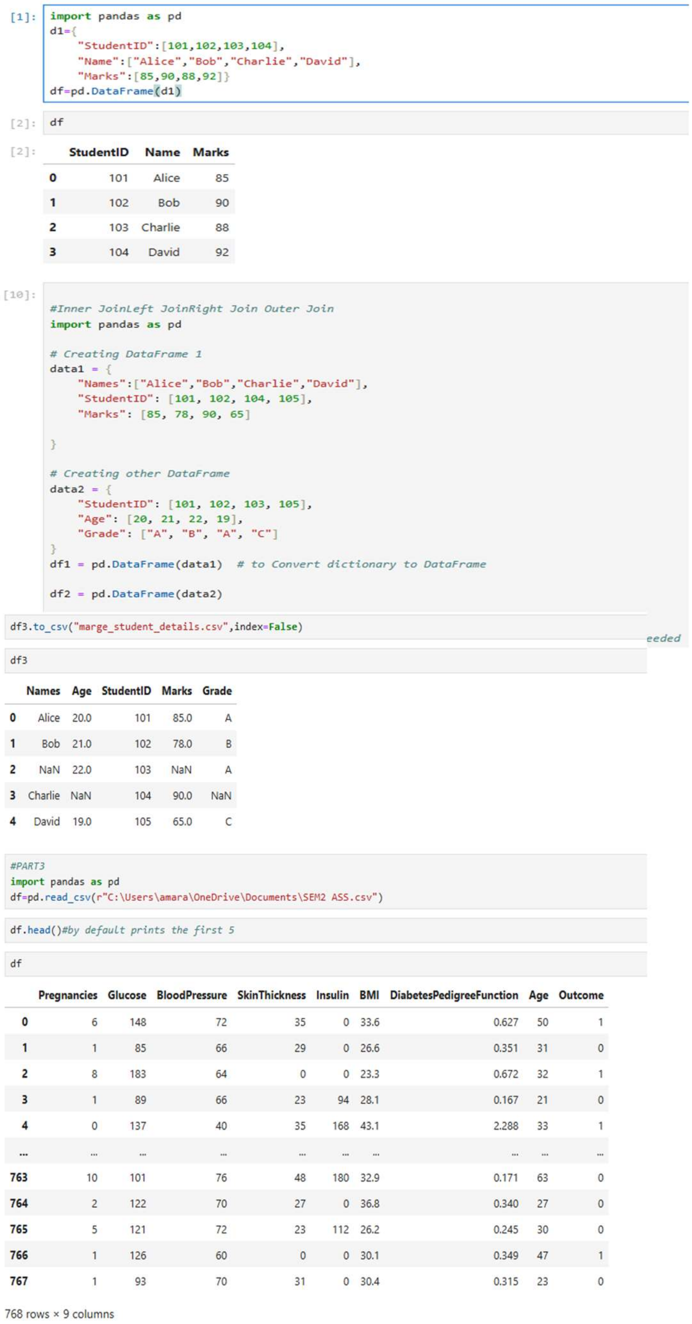
Challenges and Optimizations

1. Scalability

2. Diversity

3. Cold start

Optimizations: model pruning, data sampling, hybrid approaches.



```
[1]: import pandas as pd
     d1={
         "StudentID":[101,102,103,104],
         "Name":["Alice","Bob","Charlie","David"],
         "Marks":[85,90,88,92]}
     df=pd.DataFrame(d1)
```

```
[2]: df
```

[2]:

	StudentID	Name	Marks
0	101	Alice	85
1	102	Bob	90
2	103	Charlie	88
3	104	David	92

```
[10]:
     #Inner JoinLeft JoinRight Join Outer Join
     import pandas as pd

     # Creating DataFrame 1
     data1 = {
         "Names":["Alice","Bob","Charlie","David"],
         "StudentID": [101, 102, 104, 105],
         "Marks": [85, 78, 90, 65]

     }

     # Creating other DataFrame
     data2 = {
         "StudentID": [101, 102, 103, 105],
         "Age": [20, 21, 22, 19],
         "Grade": ["A", "B", "A", "C"]
     }
     df1 = pd.DataFrame(data1)  # to Convert dictionary to DataFrame

     df2 = pd.DataFrame(data2)
```

```
df3.to_csv("marge_student_details.csv",index=False)
```

eeded

```
df3
```

	Names	Age	StudentID	Marks	Grade
0	Alice	20.0	101	85.0	A
1	Bob	21.0	102	78.0	B
2	NaN	22.0	103	NaN	A
3	Charlie	NaN	104	90.0	NaN
4	David	19.0	105	65.0	C

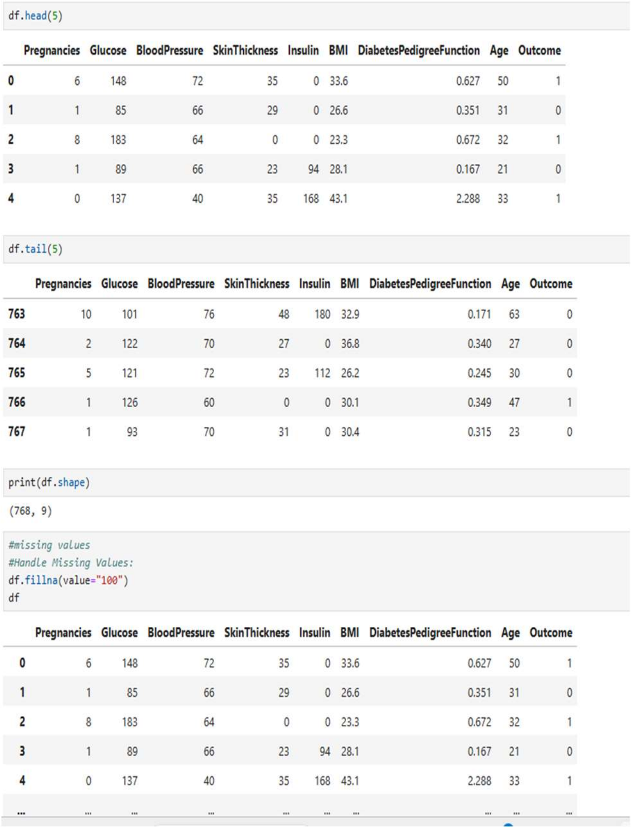
```
#PART3
import pandas as pd
df=pd.read_csv(r"C:\Users\amara\OneDrive\Documents\SEM2 ASS.csv")
```

```
df.head()#by default prints the first 5
```

```
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns



```
df.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
df.tail(5)
```

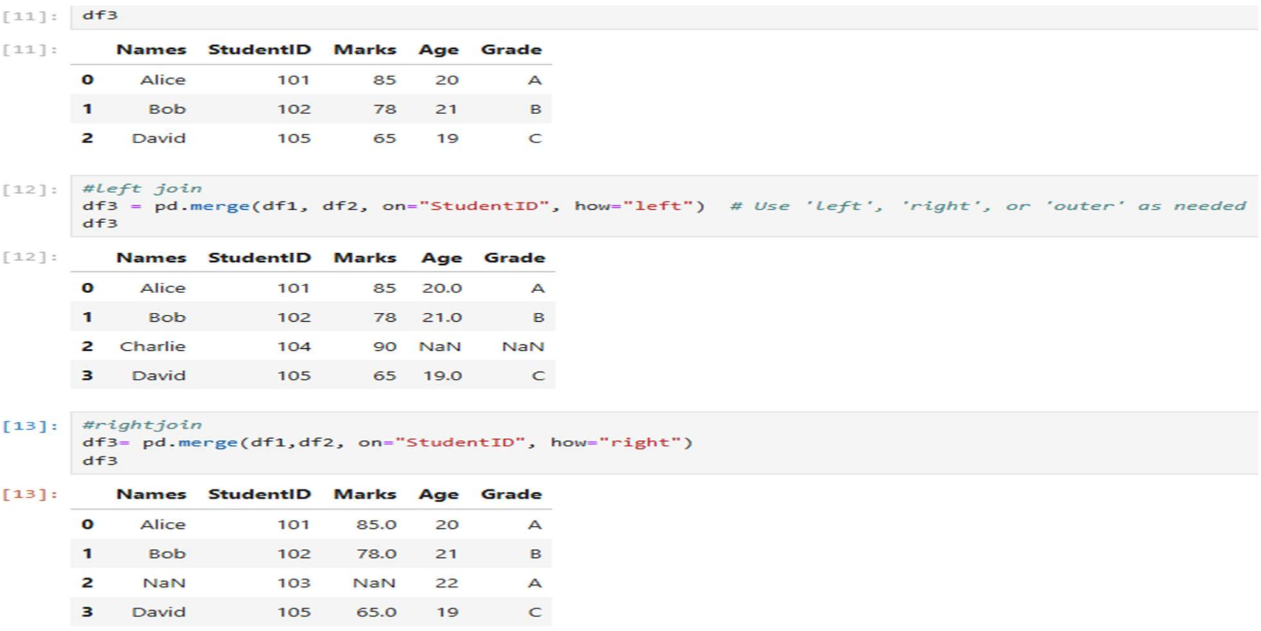
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

```
print(df.shape)
```

```
(768, 9)
```

```
#missing values
#Handle Missing Values:
df.fillna(value="100")
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...



[11]:
```
df3
```

[11]:

	Names	StudentID	Marks	Age	Grade
0	Alice	101	85	20	A
1	Bob	102	78	21	B
2	David	105	65	19	C

[12]:
```
#left join
df3 = pd.merge(df1, df2, on="StudentID", how="left")  # Use 'left', 'right', or 'outer' as needed
df3
```

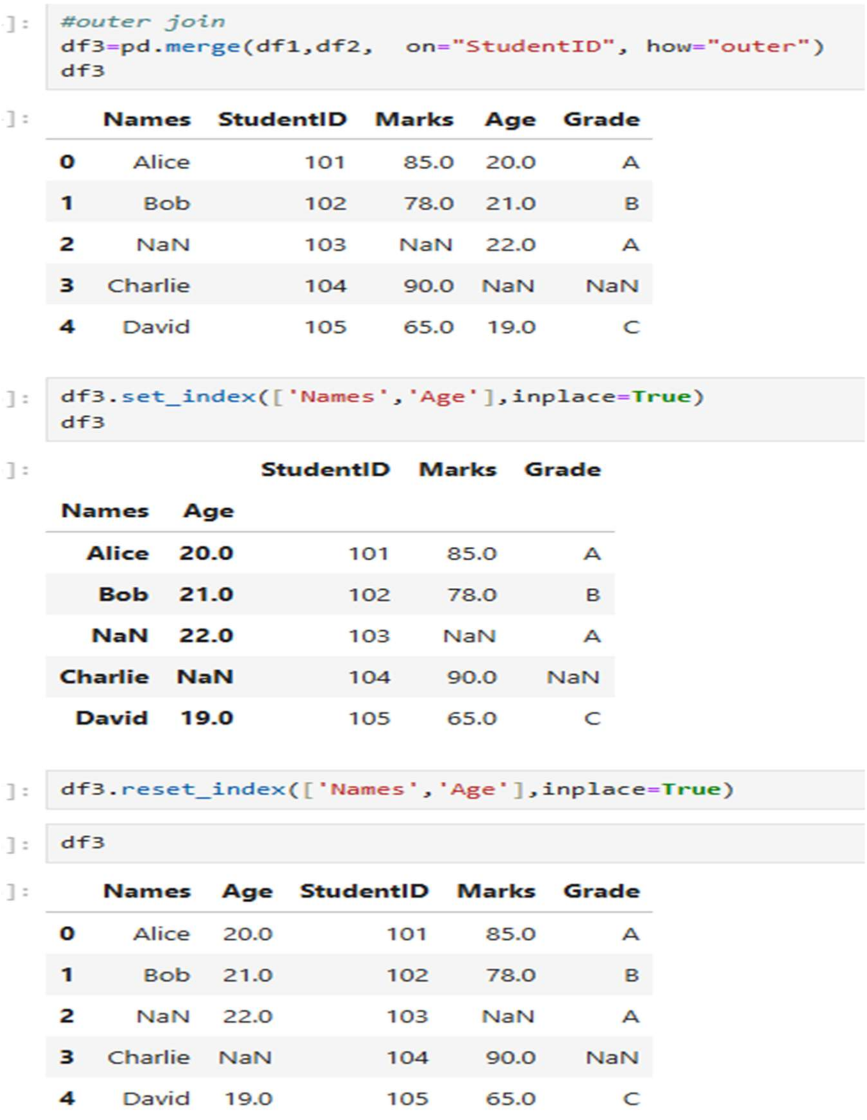
[12]:

	Names	StudentID	Marks	Age	Grade
0	Alice	101	85	20.0	A
1	Bob	102	78	21.0	B
2	Charlie	104	90	NaN	NaN
3	David	105	65	19.0	C

[13]:
```
#rightjoin
df3= pd.merge(df1,df2, on="StudentID", how="right")
df3
```

[13]:

	Names	StudentID	Marks	Age	Grade
0	Alice	101	85.0	20	A
1	Bob	102	78.0	21	B
2	NaN	103	NaN	22	A
3	David	105	65.0	19	C



]:
```
#outer join
df3=pd.merge(df1,df2,  on="StudentID", how="outer")
df3
```

]:

	Names	StudentID	Marks	Age	Grade
0	Alice	101	85.0	20.0	A
1	Bob	102	78.0	21.0	B
2	NaN	103	NaN	22.0	A
3	Charlie	104	90.0	NaN	NaN
4	David	105	65.0	19.0	C

]:
```
df3.set_index(['Names','Age'],inplace=True)
df3
```

]:

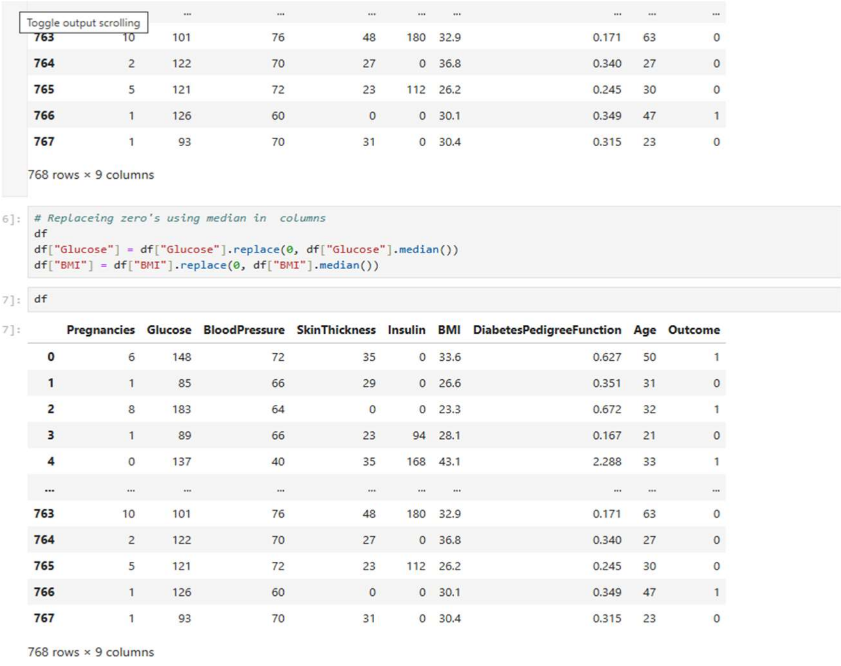
		StudentID	Marks	Grade
Names	**Age**			
Alice	**20.0**	101	85.0	A
Bob	**21.0**	102	78.0	B
NaN	**22.0**	103	NaN	A
Charlie	**NaN**	104	90.0	NaN
David	**19.0**	105	65.0	C

]:
```
df3.reset_index(['Names','Age'],inplace=True)
```

]:
```
df3
```

]:

	Names	Age	StudentID	Marks	Grade
0	Alice	20.0	101	85.0	A
1	Bob	21.0	102	78.0	B
2	NaN	22.0	103	NaN	A
3	Charlie	NaN	104	90.0	NaN
4	David	19.0	105	65.0	C



...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

6]:

```
# Replaceing zero's using median in  columns
df
df["Glucose"] = df["Glucose"].replace(0, df["Glucose"].median())
df["BMI"] = df["BMI"].replace(0, df["BMI"].median())
```

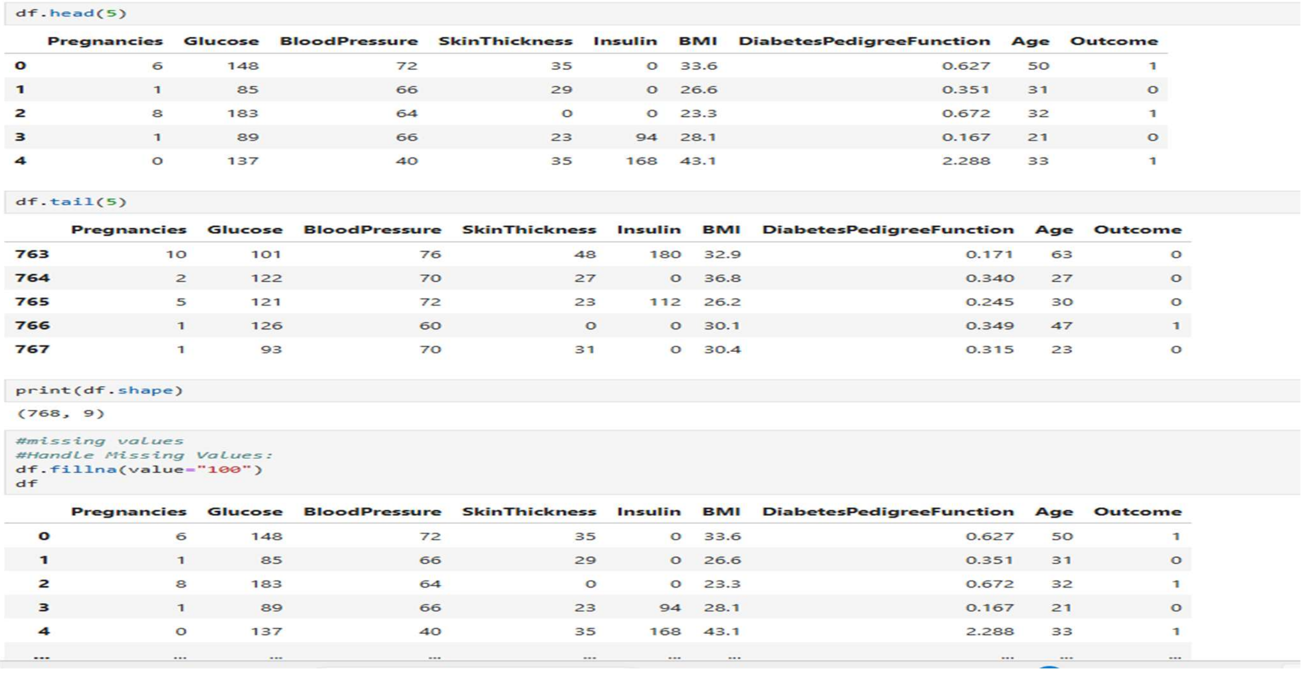
7]:

```
df
```

7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns



```
df.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
df.tail(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

```
print(df.shape)
```

```
(768, 9)
```

```
#missing values
#Handle Missing Values:
df.fillna(value="100")
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...