QUESTIONS:

Q1. How do you load a CSV file into a Pandas DataFrame?

Q2. How do you check the data type of a column in a Pandas DataFrame?

Q3. How do you select rows from a Pandas DataFrame based on a condition?

Q4. How do you rename columns in a Pandas DataFrame?

Q5. How do you drop columns in a Pandas DataFrame?

Q6. How do you find the unique values in a column of a Pandas DataFrame?

Q7. How do you find the number of missing values in each column of a Pandas DataFrame?

Q8. How do you fill missing values in a Pandas DataFrame with a specific value?

Q9. How do you concatenate two Pandas DataFrames?

Q10. How do you merge two Pandas DataFrames on a specific column?

Q11. How do you group data in a Pandas DataFrame by a specific column and apply an aggregation function?

Q12. How do you pivot a Pandas DataFrame?

Q13. How do you change the data type of a column in a Pandas DataFrame?

Q14. How do you sort a Pandas DataFrame by a specific column?

Q15. How do you create a copy of a Pandas DataFrame?

Q16. How do you filter rows of a Pandas DataFrame by multiple conditions?

Q17. How do you calculate the mean of a column in a Pandas DataFrame?

Q18. How do you calculate the standard deviation of a column in a Pandas DataFrame?

Q19. How do you calculate the correlation between two columns in a Pandas DataFrame?

Q20. How do you select specific columns in a DataFrame using their labels?

Q21. How do you select specific rows in a DataFrame using their indexes?

Q22. How do you sort a DataFrame by a specific column?

Q23. How do you create a new column in a DataFrame based on the values of another column?

Q24. How do you remove duplicates from a DataFrame?

Q25. What is the difference between .loc and .iloc in Pandas?

ANSWERS:

Q1. To load a CSV file into a Pandas DataFrame, you can use the read_csv() function in Pandas. Here’s an example:

```
import pandas as pd


df = pd.read_csv('filename.csv')
```

Q2. To check the data type of a column in a Pandas DataFrame, you can use the dtype attribute of the column. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

print(df['column_name'].dtype)
```

Q3. To select rows from a Pandas DataFrame based on a condition, you can use boolean indexing. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

new_df = df[df['column_name'] > 10]
```

Q4. To rename columns in a Pandas DataFrame, you can use the rename() method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

df = df.rename(columns={'old_column_name': 'new_column_name'})
```

Q5. To drop columns in a Pandas DataFrame, you can use the drop() method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')
df = df.drop(columns=['column_name'])
```

Q6. To find the unique values in a column of a Pandas DataFrame, you can use the unique() method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

unique_values = df['column_name'].unique()
```

Q7. To find the number of missing values in each column of a Pandas DataFrame, you can use the isnull() method and then use the sum() method to count the number of missing values. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

missing_values_count = df.isnull().sum()
```

Q8. To fill missing values in a Pandas DataFrame with a specific value, you can use the fillna() method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

df = df.fillna(value=0)
```

Q9. To concatenate two Pandas DataFrames, you can use the concat() function. Here’s an example:

```
import pandas as pd

df1 = pd.read_csv('filename1.csv')
df2 = pd.read_csv('filename2.csv')

new_df = pd.concat([df1, df2])
```

Q10. To merge two Pandas DataFrames on a specific column, you can use the merge() function. Here’s an example:

```
import pandas as pd

df1 = pd.read_csv('filename1.csv')
df2 = pd.read_csv('filename2.csv')

new_df = pd.merge(df1, df2, on='column_name')
```

Q11. To group data in a Pandas DataFrame by a specific column and apply an aggregation function, you can use the groupby() method and then apply an aggregation function such as sum(), mean(), etc. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

grouped_data = df.groupby(['column_name']).sum()
```

Q12. To pivot a Pandas DataFrame, you can use the pivot() method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')
```

```
pivoted_data = df.pivot(index='column_name1', columns='column_name2', values='column_name3')
```

Q13. To change the data type of a column in a Pandas DataFrame, you can use the astype() method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

df['column_name'] = df['column_name'].astype(float)
```

Q14. To sort a Pandas DataFrame by a specific column, you can use the sort_values() method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

sorted_df = df.sort_values(by='column_name')
```

Q15. To create a copy of a Pandas DataFrame, you can use the copy() method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

new_df = df.copy()
```

Q16. To filter rows of a Pandas DataFrame by multiple conditions, you can use boolean indexing with the & operator for “and” and the | operator for “or”. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

new_df = df[(df['column_name1'] > 10) & (df['column_name2'] < 5)]
```

Q17. To calculate the mean of a column in a Pandas DataFrame, you can use the mean() method. Here's an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

mean_value = df['column_name'].mean()
```

Q18. To calculate the standard deviation of a column in a Pandas DataFrame, you can use the std() method. Here's an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

std_value = df['column_name'].std()
```

Q19. To calculate the correlation between two columns in a Pandas DataFrame, you can use the corr() method. Here's an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

correlation_value = df['column_name1'].corr(df['column_name2'])
```

Q20. To select specific columns in a DataFrame using their labels, you can use the loc[] method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

new_df = df.loc[:, ['column_name1', 'column_name2']]
```

Q21. To select specific rows in a DataFrame using their indexes, you can use the iloc[] method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

new_df = df.iloc[[0, 2, 4]]
```

Q22. To sort a DataFrame by a specific column, you can use the sort_values() method. Here’s an example:

```
import pandas as pd

df = pd.read_csv('filename.csv')

sorted_df = df.sort_values(by='column_name')
```

Q23. To create a new column in a DataFrame based on the values of another column, you can use the following syntax:

```
import pandas as pd

df = pd.read_csv('filename.csv')
```

```
df['new_column_name'] = df['column_name1'] + df['column_name2']
```

Q24. To remove duplicates from a DataFrame, you can use the drop_duplicates() method. Here’s an example:

```
import pandas as pd


df = pd.read_csv('filename.csv')


new_df = df.drop_duplicates()
```

Q25. The main difference between .loc and .iloc is that .loc is label-based while .iloc is integer-based. .loc uses labels to index rows and columns while .iloc uses integers to index rows and columns.