PANDAS INTERVIEW

Pandas-Interview-Questions

PHANI RAJENDRA

1. Question: What is Pandas and why is it used in data analysis?

Answer: Pandas is a powerful open-source data manipulation and analysis library for Python. It's widely used in data analysis because it provides high-performance, easy-to-use data structures and tools for working with structured data.

Key features of Pandas include:

• DataFrame and Series data structures

• Ability to handle large datasets efficiently

• Built-in functions for data cleaning and transformation

• Easy data import/export from various file formats

• Powerful data aggregation and grouping capabilities

```
import pandas as pd

# Create a simple DataFrame
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
print(df)
```

2. Question: Explain the difference between a Pandas Series and DataFrame.

Answer: A Series is a one-dimensional labeled array that can hold data of any type, while a DataFrame is a two-dimensional labeled data structure with columns of potentially different types.

• Series: Think of it as a single column of data with an index.

• DataFrame: Think of it as a table with rows and columns, similar to a spreadsheet.

```
import pandas as pd

# Create a Series
s = pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])
print("Series:")
print(s)

# Create a DataFrame
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
print("\nDataFrame:")
print(df)
```

3. Question: How do you create a DataFrame from a dictionary?

Answer: You can create a DataFrame from a dictionary by passing the dictionary to the pd.DataFrame() constructor. The keys of the dictionary become the column names, and the values become the data in those columns.

```
import pandas as pd

data = {
    'Name': ['John', 'Emma', 'Alex'],
    'Age': [28, 32, 25],
    'City': ['New York', 'London', 'Paris']
}

df = pd.DataFrame(data)
print(df)
```

4. Question: What are the primary data types in Pandas?

Answer: Pandas has several built-in data types, including:

• object: for string data

• int64: for integer data

• float64: for floating-point data

• bool: for boolean data

• datetime64: for date and time data

• category: for categorical data

```
import pandas as pd

df = pd.DataFrame({
    'A': [1, 2, 3],
    'B': [1.1, 2.2, 3.3],
    'C': ['a', 'b', 'c'],
    'D': [True, False, True],
    'E': pd.date_range('20210101', periods=3)
})

print(df.dtypes)
```

5. Question: How do you handle missing data in Pandas?

Answer: Pandas provides several methods to handle missing data:

• dropna(): Remove rows or columns with missing data

• fillna(): Fill missing values with a specific value or method

• interpolate(): Interpolate missing values based on various methods

```python
import pandas as pd
import numpy as np

df = pd.DataFrame({
    'A': [1, 2, np.nan, 4],
    'B': [5, np.nan, np.nan, 8],
    'C': [9, 10, 11, 12]
})

print("Original DataFrame:")
print(df)

print("\nAfter dropping rows with NaN:")
print(df.dropna())

print("\nAfter filling NaN with 0:")
print(df.fillna(0))

print("\nAfter interpolating NaN:")
print(df.interpolate())
```

6. Question: How do you select a single column from a DataFrame?

Answer: You can select a single column from a DataFrame using either square bracket notation or dot notation.

```python
import pandas as pd

df = pd.DataFrame({
    'Name': ['John', 'Emma', 'Alex'],
    'Age': [28, 32, 25],
    'City': ['New York', 'London', 'Paris']
})

# Using square bracket notation
age_column = df['Age']

# Using dot notation
name_column = df.Name

print("Age column:")
print(age_column)
print("\nName column:")
print(name_column)
```

7. Question: How do you select multiple columns from a DataFrame?

Answer: You can select multiple columns from a DataFrame by passing a list of column names to the DataFrame using square bracket notation.

```
import pandas as pd

df = pd.DataFrame({
    'Name': ['John', 'Emma', 'Alex'],
    'Age': [28, 32, 25],
    'City': ['New York', 'London', 'Paris'],
    'Salary': [50000, 60000, 55000]
})

# Select multiple columns
selected_columns = df[['Name', 'Age', 'Salary']]

print(selected_columns)
```

8. Question: How do you add a new column to a DataFrame?

Answer: You can add a new column to a DataFrame by assigning values to a new column name.

```
import pandas as pd

df = pd.DataFrame({
    'Name': ['John', 'Emma', 'Alex'],
    'Age': [28, 32, 25]
})

# Add a new column
df['City'] = ['New York', 'London', 'Paris']

# Add a new column based on existing columns
df['Age in 5 years'] = df['Age'] + 5

print(df)
```

9. Question: How do you rename columns in a DataFrame?

Answer: You can rename columns in a DataFrame using the rename() method or by assigning new names to the columns attribute.

```python
import pandas as pd

df = pd.DataFrame({
    'A': [1, 2, 3],
    'B': [4, 5, 6],
    'C': [7, 8, 9]
})

# Using rename() method
df_renamed = df.rename(columns={'A': 'X', 'B': 'Y', 'C': 'Z'})

# Using columns attribute
df.columns = ['P', 'Q', 'R']

print("Using rename():")
print(df_renamed)
print("\nUsing columns attribute:")
print(df)
```

10. Question: How do you sort a DataFrame by a specific column?

Answer: You can sort a DataFrame by a specific column using the sort_values() method.

```python
import pandas as pd

df = pd.DataFrame({
    'Name': ['John', 'Emma', 'Alex', 'Sarah'],
    'Age': [28, 32, 25, 30],
    'Salary': [50000, 60000, 55000, 62000]
})

# Sort by Age in ascending order
df_sorted_asc = df.sort_values('Age')

# Sort by Salary in descending order
df_sorted_desc = df.sort_values('Salary', ascending=False)

print("Sorted by Age (ascending):")
print(df_sorted_asc)
print("\nSorted by Salary (descending):")
print(df_sorted_desc)
```

11.How do you filter rows in a DataFrame based on a condition?

Answer: You can filter rows in a DataFrame using boolean indexing, where you pass a boolean condition to the DataFrame.

```python
import pandas as pd

df = pd.DataFrame({
    'Name': ['John', 'Emma', 'Alex', 'Sarah'],
    'Age': [28, 32, 25, 30],
    'City': ['New York', 'London', 'Paris', 'Tokyo']
})

# Filter rows where Age is greater than 28
filtered_df = df[df['Age'] > 28]

# Filter rows where City is either London or Tokyo
city_filter = df['City'].isin(['London', 'Tokyo'])
filtered_cities = df[city_filter]

print("Filtered by Age > 28:")
print(filtered_df)
print("\nFiltered by City (London or Tokyo):")
print(filtered_cities)
```

12. Question: How do you handle duplicate rows in a DataFrame?

Answer: You can handle duplicate rows in a DataFrame using the drop_duplicates() method.

```python
import pandas as pd

df = pd.DataFrame({
    'Name': ['John', 'Emma', 'John', 'Sarah', 'Emma'],
    'Age': [28, 32, 28, 30, 32],
    'City': ['New York', 'London', 'New York', 'Tokyo', 'London']
})

print("Original DataFrame:")
print(df)

# Remove all duplicate rows
df_no_duplicates = df.drop_duplicates()

print("\nDataFrame with all duplicates removed:")
print(df_no_duplicates)

# Remove duplicates based on specific columns
df_unique_names = df.drop_duplicates(subset=['Name'])

print("\nDataFrame with duplicates removed based on 'Name' column:")
print(df_unique_names)
```

13. Question: How do you merge two DataFrames?

Answer: You can merge two DataFrames using the merge() function, which allows you to combine DataFrames based on common columns or indices.

```python
import pandas as pd

df1 = pd.DataFrame({
    'ID': [1, 2, 3, 4],
    'Name': ['John', 'Emma', 'Alex', 'Sarah']
})

df2 = pd.DataFrame({
    'ID': [1, 2, 3, 5],
    'Age': [28, 32, 25, 35]
})

# Merge on 'ID' column
merged_df = pd.merge(df1, df2, on='ID')

print("Merged DataFrame:")
print(merged_df)

# Left join
left_join = pd.merge(df1, df2, on='ID', how='left')

print("\nLeft Join:")
print(left_join)
```

14. Question: How do you group data in a DataFrame?

Answer: You can group data in a DataFrame using the groupby() method, which allows you to split the data into groups based on some criteria and then apply a function to each group.

```python
import pandas as pd

df = pd.DataFrame({
    'Name': ['John', 'Emma', 'Alex', 'Sarah', 'Mike', 'Emily'],
    'Department': ['IT', 'HR', 'IT', 'Finance', 'HR', 'Finance'],
    'Salary': [50000, 60000, 55000, 65000, 58000, 62000]
})

# Group by Department and calculate mean salary
grouped = df.groupby('Department')['Salary'].mean()

print("Mean salary by Department:")
print(grouped)

# Group by Department and get multiple statistics
multi_stats = df.groupby('Department').agg({
    'Salary': ['mean', 'min', 'max'],
    'Name': 'count'
})

print("\nMultiple statistics by Department:")
print(multi_stats)
```

15. Question: How do you melt a DataFrame?

Answer: You can melt a DataFrame using the melt() method, which is used to unpivot a DataFrame from wide format to long format.

```python
import pandas as pd

df = pd.DataFrame({
    'Name': ['John', 'Emma'],
    'Math': [90, 85],
    'Science': [95, 92],
    'History': [88, 89]
})

# Melt the DataFrame
melted = pd.melt(df, id_vars=['Name'], var_name='Subject', value_name='Score')

print("Original DataFrame:")
print(df)
print("\nMelted DataFrame:")
print(melted)
```

16. Question: How do you handle date and time data in Pandas?

Answer: Pandas provides powerful tools for working with date and time data, including the to_datetime() function and various date/time-related methods.

```
import pandas as pd

df = pd.DataFrame({
    'Date': ['2021-01-01', '2021-01-15', '2021-02-01', '2021-02-15'],
    'Value': [100, 150, 120, 180]
})

# Convert 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])

# Extract various components
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day

# Add 7 days to each date
df['Date_plus_7'] = df['Date'] + pd.Timedelta(days=7)

print(df)
```

17. Question: How do you handle categorical data in Pandas?

Answer: Pandas provides a special dtype called ‘category’ for handling categorical data, which can improve performance and memory usage for certain operations.

```
import pandas as pd

df = pd.DataFrame({
    'ID': range(1, 1001),
    'Color': ['Red', 'Blue', 'Green', 'Yellow'] * 250
})

# Convert 'Color' to categorical
df['Color'] = df['Color'].astype('category')

print("DataFrame info:")
df.info()

# Get category codes
print("\nCategory codes:")
print(df['Color'].cat.codes.head())

# Get category names
print("\nCategory names:")
print(df['Color'].cat.categories)
```

18. Question: How do you handle outliers in a Pandas DataFrame?

Answer: There are several ways to handle outliers in a Pandas DataFrame, including detection using statistical methods and removal or transformation of outlier values.

```
import pandas as pd
import numpy as np

np.random.seed(0)
df = pd.DataFrame({
    'Value': np.random.normal(0, 1, 1000)
})

# Add some outliers
df.loc[df.index[-5:], 'Value'] = np.random.uniform(10, 20, 5)

# Detect outliers using Z-
```

All the Best.