INTERVIEW QUESTIONS

- What is Pandas, and how does it differ from NumPy?

  (Google)

- Answer: Pandas is a Python library designed for data manipulation and analysis. Unlike NumPy, which handles homogenous data (mainly numerical arrays), Pandas provides tools for working with heterogeneous data, especially in tabular form. Pandas offers Series (1D) and DataFrame (2D) structures with labeled indexing.

- How can you create a DataFrame in Pandas from a dictionary or a list?
  (Amazon)

- Answer: To create a DataFrame from a dictionary:

```
import pandas as pd
data = {'Name': ['Alice', 'Bob'], 'Age': [24, 27]}
df = pd.DataFrame(data)
print(df)
```

```
    Name  Age
0  Alice   24
1    Bob   27
```

- From a list:

```
data= [['Alice', 24], ['Bob', 27]]
df = pd.DataFrame(data, columns=['Name', 'Age'])
print(df)
```

```
    Name  Age
0  Alice   24
1    Bob   27
```

- Explain the difference between a Series and a DataFrame in Pandas.
  (Meta)

- Answer A Series is a one-dimensional labeled array capable of holding any data type. A DataFrame, on the other hand, is a two-dimensional table-like structure consisting of rows and columns, where each column is a Series. The DataFrame can hold data of different types (e.g., integers, strings, floats).

- How do you read a CSV file into a Pandas DataFrame?
  (Netflix)

- **Answer:** You can use the pd.read_csv() function to read a CSV file into a DataFrame

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

- How can you write a Pandas DataFrame to a CSV file?
  (Microsoft)

- **Answer**: You can use the to_csv() function to export a DataFrame to a CSV file

  df.to_csv('output.csv', index=False

- How do you select subsets of data from a DataFrame using loc[] and iloc[]?
  (Google)

Answer:

loc[] is label-based: you select rows and columns by their labels.

iloc[] is index-based: you select rows and columns by integer positions.

df.loc[0:2, ['Name', 'Age']]

df.iloc[0:2, 0:2]

- How do you filter rows in a DataFrame based on a condition in a column?
  (Amazon)

Answer:

You can filter rows by applying a condition to a DataFrame column

filtered_df = df[df['Age'] > 25]

- How do you handle missing data in a Pandas DataFrame? (Meta)

Answer:

To handle missing data:

Drop missing values: df.dropna()

Fill missing values: df.fillna(value)

Fill with forward/backward values: df.fillna(method='ffill') or df.fillna(method='bfill')

- What is the groupby() function in Pandas, and how is it used?
  (Netflix)

Answer:

groupby() is used to split the data into groups based on some criteria, apply a function to each group, and combine the results**. For example**:

df.groupby('Age')['Name'].count()

- How do you merge two DataFrames, and what are the different types of joins available?
  (Microsoft)

Answer:

You can merge two DataFrames using the merge() function.

Types of joins:

Inner Join: df1.merge(df2, on='key')

Left Join: df1.merge(df2, on='key', how='left')

Right Join: df1.merge(df2, on='key', how='right')

Outer Join: df1.merge(df2, on='key', how='outer')

- How do you concatenate DataFrames along rows or columns?
  (Google)

Answer:

Use the concat() function

 # Concatenate along rows:

pd.concat([df1, df2], axis=0)

 # Concatenate along columns:

pd.concat([df1, df2], axis=1)

- What is the difference between merge(), join(), and concat()?
  (Amazon)

  Answer

- **merge()** joins DataFrames based on key columns or indexes.
- **join()** is used for joining DataFrames on their index or a key column.
- **concat()** is used to concatenate DataFrames along a specific axis (rows or columns).

- How do you add a new column to a DataFrame?
  (Meta)

  Answer:
  To add a new column, assign a value to the new column name:

  df['New_Column'] = [1, 2, 3]

- How do you remove rows or columns from a DataFrame?
  (Netflix)

  Answer:
  To remove columns: df.drop('column_name', axis=1)
  To remove rows: df.drop(index)

  df.drop('Age', axis=1)
  df.drop(0, axis=0)

- What is vectorization in Pandas, and why is it useful?
  (Microsoft)

Answer:
Vectorization allows you to perform operations on entire arrays or DataFrames instead of looping through individual elements. It is faster and more efficient.

df['New_Column'] = df['Column1'] + df['Column2']

- How do you apply a function to each element in a DataFrame or Series using apply() and map()?
(Google)

Answer:

- apply() applies a function to rows or columns of a DataFrame.
- map() is used to map values from a Series.

```
df['Age'] = df['Age'].apply(lambda x: x + 1)
df['Name'] = df['Name'].map(lambda x: x.upper())
print(df['Age'])
print(df['Name'])
```

```
0    25
1    28
Name: Age, dtype: int64
0    ALICE
1      BOB
Name: Name, dtype: object
```

- How do you sort a DataFrame by the values of one or more columns?
(Amazon)

Answer:
Use sort_values() to sort by a column:

```
df_sorted = df.sort_values(by=['Age'], ascending=False)
print(df_sorted)
```

```
    Name  Age
1    Bob   27
0  Alice   24
```

- What is the difference between pivot() and pivot_table() in Pandas?
(Meta)

Answer:

- pivot() reshapes data based on column values but does not perform aggregation.
- pivot_table() is similar but allows aggregation (e.g., sum, mean).

```
df.pivot(index='Date', columns='Product', values='Sales')
```

```
df.pivot_table(index='Date', columns='Product', values='Sales', aggfunc='sum')
```

- How do you handle duplicates in a DataFrame? How can you remove duplicates? (Netflix)

  Answer:
  You can remove duplicates using drop_duplicates():

  ```
  df_clean = df.drop_duplicates()
  ```

- What is the crosstab() function in Pandas, and how is it used? (Microsoft)

  Answer:
  crosstab() computes a cross-tabulation of two or more factors. It's useful for counting combinations of categorical variables.

  ```
  pd.crosstab(df['Gender'], df['Product'])
  ```

- How do you perform time-series analysis with Pandas? (Google)

  Answer:
  Pandas supports time series analysis using datetime objects. You can use functions like resample(), rolling(), and shift().

  ```
  df['Date'] = pd.to_datetime(df['Date'])
  df.set_index('Date', inplace=True)
  ```

- How do you compute summary statistics like mean, median, and standard deviation for a DataFrame?
  (Amazon)

  Answer:
  Use the respective functions

  df['Age'].mean()
  df['Age'].median()
  df['Age'].std()

- What is reindexing in Pandas, and why is it useful?
  (Meta)

  Answer:
  Reindexing allows you to change the row/column labels in a DataFrame, or to conform DataFrames to a common index.

```
df_reindexed = df.reindex([0, 2, 1])
print(df_reindexed)
```

```
    Name   Age
0  Alice  24.0
2    NaN   NaN
1    Bob  27.0
```

- How can you visualize data from a Pandas DataFrame using Matplotlib or Seaborn?
  (Microsoft)

  Answer:
  You can use the plot() function from Matplotlib:

```
import matplotlib.pyplot as plt
df['Age'].plot(kind='bar')
plt.show()
```