Pandas Interview Questions & Answers

Basic Level Questions

Q1. What is Pandas? Why is it used?
A: Pandas is an open-source Python library for data manipulation and analysis. It provides data structures like Series and DataFrame, making it easy to handle structured data efficiently.

Q2. Explain the difference between Series and DataFrame.

- Series → One-dimensional labeled array (like a column).
- DataFrame → Two-dimensional labeled structure (like a table).

Q3. How do you read a CSV file using Pandas?

```
import pandas as pd
df = pd.read_csv("file.csv")
```

Q4. How can you select a column or multiple columns from a DataFrame?

```
df['column_name']       # single column
df[['col1', 'col2']]    # multiple columns
```

Q5. What is the difference between loc[] and iloc[]?

- loc[] → Label-based indexing.
- iloc[] → Integer position-based indexing.

Q6. How do you check for missing values in Pandas?

```
df.isnull().sum()
```

Q7. How can you remove duplicate rows in Pandas?

```
df.drop_duplicates(inplace=True)
```

Q8. How do you sort a DataFrame by column values?

```
df.sort_values(by='column_name', ascending=True)
```

Q9. What is the difference between head() and tail()?

- head(n) → First n rows (default 5).
- tail(n) → Last n rows (default 5).

Q10. How do you get summary statistics of a DataFrame?

```
df.describe()
```

Intermediate / Advanced (3+ Years Experience)

Q11. How do you handle missing data in Pandas?
Options:

- df.fillna(value)
- df.dropna()
- df.interpolate()

Q12. What are the different ways to merge/join DataFrames?

- merge()
- join()
- concat()

Q13. Explain the difference between apply(), map(), and applymap().

- map() → Works on Series.
- apply() → Works on DataFrame row/column.
- applymap() → Works on entire DataFrame element-wise.

Q14. How do you group and aggregate data in Pandas?

```
df.groupby('category')['sales'].mean()
df.groupby('category').agg({'sales': 'sum', 'profit': 'mean'})
```

Q15. Explain vectorization and why it is faster than loops in Pandas.
Vectorization uses NumPy's optimized C operations, avoiding Python-level loops.

Q16. What are MultiIndex DataFrames?

MultiIndex allows hierarchical indexing on rows/columns.

```
df.set_index(['city','date'], inplace=True)
```

Q17. How do you pivot and unpivot (melt) data in Pandas?

```
df.pivot(index='id', columns='month', values='sales')
pd.melt(df, id_vars=['id'], value_vars=['sales'])
```

Q18. How do you optimize memory usage in large Pandas DataFrames?

- Convert float64 → float32
- Convert object → category
- Use df.memory_usage(deep=True)

Q19. Explain difference between copy() and view() in Pandas.

- view() → Reference (changes affect original).
- copy() → Independent deep copy.

Q20. How do you handle time-series data in Pandas?

- Convert with pd.to_datetime()
- Use .resample(), .rolling() for operations

Q21. Scenario: You have a dataset with 10 million rows. Some columns are categorical with only 5–10 unique values. How would you optimize it?

- Convert to category dtype
- Use chunked reading (pd.read_csv(chunksize=...))
- Prefer vectorized operations
- Use distributed libraries like Dask or Modin