Pandas EDA Interview-Ready Guide

---

1. Loading Excel/CSV Data

Theory: Load datasets using Pandas. Excel/CSV are most common.

```
import pandas as pd

# Load CSV
df = pd.read_csv('employee_data.csv')

# Load Excel
df = pd.read_excel('employee_data.xlsx', sheet_name='Employees')

print(df.head())
```

Expected Output:

```
| Name  | Age | Department | Salary | JoiningDate |
|-------|-----|------------|--------|------------|
| John  | 28  | IT         | 55000  | 2020-01-15 |
| Alice | 32  | HR         | 60000  | 2019-05-23 |
| Bob   | 25  | IT         | 49500  | 2021-03-10 |
| Emma  | 29  | Finance    | 70000  | 2018-07-12 |
| Liam  | 35  | HR         | 65000  | 2017-11-30 |
```

---

2. Basic Info & Summary

Theory: Understand dataset size, columns, data types, and missing values.

```
# Info and data types
df.info()

# Shape of DataFrame
print(df.shape)

# Summary statistics
print(df.describe())
```

Expected Output: Shows columns, non-null count, data types, shape, and summary statistics.

---

3. Selecting Columns & Rows

Theory: Access columns and rows using `[]`, `.loc`, `.iloc`.

```
# Select column
print(df['Salary'].head())

# Select multiple columns
print(df[['Name','Salary']].head())

# Select row by index
print(df.iloc[0])

# Select row by label
print(df.loc[0,['Name','Salary']])
```

Expected Output: Shows values of selected rows/columns.

---

4. Filtering & Conditional Selection

Theory: Filter rows using conditions, multiple conditions, `.isin()`.

```
# Single condition
print(df[df['Salary'] > 60000][['Name','Salary']])

# Multiple conditions
print(df[(df['Age']>30) & (df['Department']=='IT')])

# Using isin
print(df[df['Department'].isin(['IT','HR'])])
```

Expected Output: Rows that satisfy the condition.

---

5. Aggregation & Grouping

Theory: Summarize data with `groupby` and `agg`.

```
# Average Salary by Department
print(df.groupby('Department')['Salary'].mean())

# Multiple aggregations
print(df.groupby('Department').agg({'Salary':['mean','max'],'Age':'min'}))
```

Expected Output: Shows aggregated metrics.

6. Apply & Lambda Functions

Theory: Transform data row-wise or column-wise using `apply()` and lambda.

```
# Experience in years
df['Experience'] = df.apply(lambda x: (pd.Timestamp('2026-02-06') -
x['JoiningDate']).days // 365, axis=1)

# Conditional Salary Increase
df['Salary'] = df.apply(lambda x: x['Salary']*1.1 if x['Department']=='IT'
else x['Salary'], axis=1)
```

Expected Output: New columns with calculated values.

---

7. DateTime Operations

Theory: Extract year, month, day; calculate durations.

```
# Extract Year/Month
df['JoinYear'] = df['JoiningDate'].dt.year

# Calculate experience in days
df['DaysSinceJoining'] = (pd.Timestamp('2026-02-06') -
df['JoiningDate']).dt.days
```

Expected Output: Columns with year, month, and days since joining.

---

8. Correlation & Covariance

Theory: Understand relationships between numerical variables.

```
# Correlation
print(df[['Age','Salary','Experience']].corr())

# Covariance
print(df[['Age','Salary','Experience']].cov())
```

Expected Output: Correlation and covariance matrices.

---

9. Pivot Tables & Multi-level Aggregation

Theory: Summarize data like Excel pivot tables.

```
# Average Salary by Department
print(df.pivot_table(values='Salary', index='Department', aggfunc='mean'))

# Count & Average Salary by Department
print(df.pivot_table(values='Salary', index='Department',
aggfunc=['mean','count']))
```

Expected Output: Aggregated pivot tables.

---

10. Duplicates & Conditional Columns

Theory: Detect/remove duplicates and create new columns based on conditions.

```
# Drop duplicates
df.drop_duplicates(inplace=True)

# HighSalary flag
import numpy as np
df['HighSalary'] = np.where(df['Salary']>60000,'Yes','No')

# Seniority based on Experience
df['Seniority'] = df['Experience'].apply(lambda x: 'Senior' if x>=5 else
'Junior')
```

Expected Output: Columns HighSalary and Seniority created.

---

11. Advanced Filtering & Multi-condition Selection

Theory: Use `&`, `|`, `.isin()` for complex filters.

```
# Age > 30 and Salary > 60000
filtered = df[(df['Age']>30) & (df['Salary']>60000)]

# Department in IT or HR
filtered_dept = df[df['Department'].isin(['IT','HR'])]
```

Expected Output: Rows satisfying complex conditions.

---

12. Combining, Merging & Concatenating DataFrames

Theory: Stack datasets vertically/horizontally or merge like SQL JOIN.

```
# Vertical Concatenation
df_combined = pd.concat([df1, df2])

# Merge DataFrames
merged_df = pd.merge(df1, df2, on='Name', how='inner')
```

Expected Output: Combined DataFrames.

---

13. Handling Outliers & Data Scaling

Theory: Detect outliers using IQR/Z-score, scale features using MinMax or StandardScaler.

```
# IQR Method
Q1 = df['Salary'].quantile(0.25)
Q3 = df['Salary'].quantile(0.75)
IQR = Q3-Q1
outliers = df[(df['Salary']<(Q1-1.5*IQR)) | (df['Salary']>(Q3+1.5*IQR))]

# MinMax Scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df['Salary_Scaled'] = scaler.fit_transform(df[['Salary']])
```

Expected Output: Outliers identified, Salary_Scaled column added.

---

14. Visualization & Basic Plots

Theory: Histogram for distributions, bar plots for categorical counts.

```
import matplotlib.pyplot as plt

# Histogram
df['Age'].hist(bins=10)
plt.show()

# Bar Plot
df['Department'].value_counts().plot(kind='bar')
plt.show()
```

Expected Output: Distribution plots.

15. Advanced Plots (Boxplot, Scatter, Pairplot)

Theory: Detect outliers and relationships between variables.

```
import seaborn as sns

# Boxplot
df.boxplot(column='Salary', by='Department')
plt.show()

# Scatter
df.plot(kind='scatter', x='Experience', y='Salary')
plt.show()

# Pairplot
sns.pairplot(df[['Age','Salary','Experience']], diag_kind='hist')
plt.show()
```

Expected Output: Boxplot, scatter, and pairplot figures.

16. Saving, Exporting Data & Excel Operations

Theory: Save cleaned or analyzed data to CSV/Excel.

```
# Save to CSV
df.to_csv('cleaned_data.csv', index=False)

# Save to Excel
df.to_excel('employee_data.xlsx', sheet_name='Employees', index=False)

# Multiple sheets
with pd.ExcelWriter('report.xlsx') as writer:
    df.to_excel(writer, sheet_name='AllEmployees', index=False)
    df[['Department','Salary']].to_excel(writer, sheet_name='SalaryReport',
index=False)
```

Expected Output: Files created.

17. Common Pandas Interview Tricks & Questions

Theory: Frequently asked shortcuts.

```python
# Top/Bottom N values
print(df.nlargest(3,'Salary'))
print(df.nsmallest(2,'Age'))

# Rename columns
df.rename(columns={'Salary':'MonthlySalary','Age':'EmployeeAge'},
inplace=True)

# Memory optimization
df['Department'] = df['Department'].astype('category')
df['MonthlySalary'] = df['MonthlySalary'].astype('float32')
```

Expected Output: Top salaries, renamed columns, optimized memory.

---

18. Miscellaneous Useful Functions & Shortcuts

Theory: Quick functions for unique values, missing data, string ops, mapping.

```python
# Unique & value counts
print(df['Department'].unique())
print(df['Department'].value_counts())

# Check missing values
print(df.isnull().sum())

# String operations
df['Name'] = df['Name'].str.upper()

# Mapping/Replace
df['DeptCode'] =
df['Department'].map({'IT':'I','HR':'H','Finance':'F','Marketing':'M'})
df['HighSalaryNum'] = df['HighSalary'].replace({'Yes':1,'No':0})
```

Expected Output: Unique values, counts, uppercase names, mapped codes.

This notebook covers all major Pandas operations for an interview-ready EDA workflow, including: - Loading & inspecting data
- Cleaning & transformation
- Filtering & grouping
- Aggregation & pivoting
- DateTime & outlier handling
- Visualization & export
- Shortcuts & interview tricks