**1. Python Basics for Data Analytics**

**Q1: What are the core features of Python that make it suitable for data analytics?**  
**A:**

* **Ease of Use:** Simple syntax, easy to learn and implement.
* **Rich Ecosystem:** Libraries like pandas, numpy, and matplotlib facilitate analysis.
* **Scalability:** Suitable for both small and large datasets.
* **Integration:** Works with SQL, Hadoop, and various databases.
* **Automation:** Enables task automation with scripting.

**Q2: How do you load and inspect large datasets in Python?**  
**A:**

* Use pandas functions such as:

python

CopyEdit

import pandas as pd

df = pd.read\_csv('data.csv')

print(df.info()) # Overview of dataset

print(df.head()) # First few rows

print(df.describe()) # Summary statistics

* Inspect memory usage using df.memory\_usage(deep=True).

**2. Data Wrangling and Cleaning**

**Q3: How do you handle missing and duplicate data in Python?**  
**A:**

* **Handling Missing Data:**

python

CopyEdit

df.dropna(inplace=True) # Remove missing values

df.fillna(df.mean(), inplace=True) # Fill with mean

* **Handling Duplicates:**

python

CopyEdit

df.drop\_duplicates(inplace=True)

**Q4: Explain the difference between apply(), map(), and lambda functions in pandas.**  
**A:**

* **apply():** Used for row/column-wise operations on DataFrames.

python

CopyEdit

df['new\_col'] = df['col'].apply(lambda x: x \* 2)

* **map():** Element-wise transformation for Series.

python

CopyEdit

df['col'] = df['col'].map(lambda x: x.upper())

* **lambda:** Anonymous function useful for one-time operations.

**3. Data Manipulation Techniques**

**Q5: How do you filter and sort data in pandas?**  
**A:**

* **Filtering:**

python

CopyEdit

filtered\_df = df[df['Age'] > 30]

* **Sorting:**

python

CopyEdit

df.sort\_values(by=['Salary'], ascending=False)

**Q6: Explain the difference between loc and iloc in pandas.**  
**A:**

* loc: Access rows/columns by label.

python

CopyEdit

df.loc[2, 'Name']

* iloc: Access rows/columns by index position.

python

CopyEdit

df.iloc[2, 1]

**4. Data Visualization**

**Q7: How do you visualize categorical vs numerical data using Python?**  
**A:**

* **Categorical Data:** Use bar charts.

python

CopyEdit

sns.countplot(x='Category', data=df)

* **Numerical Data:** Use histograms or boxplots.

python

CopyEdit

sns.histplot(df['Salary'])

sns.boxplot(x=df['Salary'])

**Q8: What are some best practices for creating effective visualizations?**  
**A:**

* Keep it simple and readable.
* Use appropriate chart types for the data.
* Label axes and include legends.
* Use color coding to distinguish categories.
* Avoid clutter and unnecessary elements.

**5. Statistical Analysis and A/B Testing**

**Q9: Explain A/B testing and its significance in data analytics.**  
**A:**  
A/B testing is a statistical method used to compare two versions (A and B) of a product or feature to determine which performs better based on a key metric (e.g., conversion rate). It helps in data-driven decision-making.

**Steps in A/B Testing:**

1. Define hypothesis.
2. Split users into control and test groups.
3. Apply the changes to the test group.
4. Analyze the results statistically.

**Q10: How do you perform A/B testing in Python?**  
**A:**  
Using scipy.stats for hypothesis testing:

python

CopyEdit

from scipy.stats import ttest\_ind

control = df[df['group'] == 'control']['conversion']

experiment = df[df['group'] == 'experiment']['conversion']

t\_stat, p\_val = ttest\_ind(control, experiment)

if p\_val < 0.05:

print("Significant difference detected")

else:

print("No significant difference")

**6. Working with SQL in Python**

**Q11: How do you connect and retrieve data from an SQL database using Python?**  
**A:**  
Using sqlite3 or SQLAlchemy for database interactions.

python

CopyEdit

import sqlite3

conn = sqlite3.connect('database.db')

df = pd.read\_sql\_query("SELECT \* FROM sales", conn)

conn.close()

**Q12: What are common SQL operations used in data analysis?**  
**A:**

* Data retrieval: SELECT, WHERE, JOIN.
* Aggregation: GROUP BY, SUM(), AVG().
* Filtering: HAVING, ORDER BY.
* Data merging using joins.

**7. Performance Optimization**

**Q13: How do you optimize slow-running Python scripts?**  
**A:**

* Use vectorized operations instead of loops.
* Use efficient data types (astype()).
* Profile the script with cProfile.
* Minimize memory usage with df.memory\_usage().

**Example Optimization:**

python

CopyEdit

df['new\_col'] = df['col1'] + df['col2'] # Vectorized

**Q14: What are the best practices for memory management in pandas?**  
**A:**

* Use appropriate data types (category for categorical values).
* Drop unnecessary columns with df.drop().
* Convert columns to datetime format when applicable.

**8. Data Transformation and Aggregation**

**Q15: How do you aggregate data based on multiple columns in pandas?**  
**A:**  
Using groupby() and agg() functions.

python

CopyEdit

df.groupby(['Department', 'Region'])['Sales'].agg(['sum', 'mean'])

**9. Automation in Python for Data Analytics**

**Q16: How can Python be used to automate repetitive data tasks?**  
**A:**

* Automating reporting with scheduled scripts (cron jobs, task scheduler).
* Using pandas for automated data cleaning.
* Emailing reports with smtplib.

**Example:**

python

CopyEdit

import pandas as pd

df = pd.read\_csv('data.csv')

df.to\_excel('output.xlsx')

**10. Python Coding Scenarios**

**Q17: How do you handle large datasets in Python?**  
**A:**

* Use chunking while reading files:

python

CopyEdit

for chunk in pd.read\_csv('large\_data.csv', chunksize=10000):

process(chunk)

* Consider using Dask for parallel processing.

**Q18: Describe a real-world data analytics problem you solved using Python.**  
**A:**  
Example Answer:  
"I worked on optimizing a customer segmentation analysis for an e-commerce platform by cleaning the data, performing exploratory data analysis (EDA), and building customer profiles using clustering techniques in Python. The insights helped the business tailor marketing strategies and increase sales by 15%."