Python

APIs & Databases

Alish Bista - 11/11/2025

API

API

Application Programming Interface

- Way to request data from services
- Like ordering from a menu
- Send request & Get data back
- Returns data in JSON format

```
import requests

url = "https://
restcountries.com/v3.1/all"
response = requests.get(url)
data = response.json()
```

API

Handling API data

- Check response status first
- Convert JSON to DataFrame
- Extract needed fields
- Handle nested structures
- Ready for analysis

```
response = requests.get(url)
response.raise_for_status()


countries = []
for c in response.json():
    countries.append({
        'name': c['name']
['common'],
        'population':
c['population']
    })
df = pd.DataFrame(countries)
```

Databases

PostgreSQL

Connecting to PostgreSQL

- Use SQLAlchemy for connections
- Connection string has all details
- Host, database name, credentials
- Create engine once, reuse it
- Close connection when done

```
from sqlalchemy import create_engine

# Connection string
engine = create_engine(
    'postgresql://
postgres:password@localhost:5432/
analytics_db'
)

# Test connection
with engine.connect() as conn:
    print("Connected!")
```

Fetching Data

The SQL way

- Write SQL queries as strings
- Full control over query logic
- Use JOINs, WHERE, GROUP BY
- Good for complex operations
- Returns pandas DataFrame

```
# Read with SQL query
query = """
    SELECT product_name, sales_amount,
sale_date
    FROM sales
    WHERE sale_date >= '2024-01-01'
"""

df = pd.read_sql(query, engine)
```

Fetching Data

The Pandas Way

- Read entire table without SQL
- Filter in pandas after reading
- Good for small to medium tables

```
# Read entire table

df = pd.read_sql_table('sales',
engine)


# Filter in pandas

recent_sales = df[df['sale_date'] >=
'2024-01-01']


# Or read with simple query

df = pd.read_sql('sales', engine)
```

Write Back

Writing Back to Database

- Use ***to_sql()*** method
- Choose: replace or append data
- Creates table if doesn’t exist
- Specify table name

```
# Write DataFrame to database
df.to_sql(
    'new_sales_data',
    engine,
    if_exists='replace',  # or 'append'
    index=False
)

# Verify
check = pd.read_sql_table('new_sales_data',
engine)
```

Multiple Tables

Joining Multiple Tables

- SQL JOIN combines tables in database
- Pandas also has merge/join methods
- SQL: Better for large data (faster)
- Pandas: Better for small data (easier)
- Choose based on data size

```
# SQL approach
query = """
    SELECT s.sale_date, s.amount, p.product_name
    FROM sales s
    JOIN products p ON s.product_id = p.id
"""
df = pd.read_sql(query, engine)

# Pandas approach
sales = pd.read_sql_table('sales', engine)
products = pd.read_sql_table('products', engine)
df = sales.merge(products, on='product_id')
```

Q&A