Program 1: Load a CSV, clean it, and plot a histogram

```
# Concepts: ingestion, handling missing values, visualization

import pandas as pd
import matplotlib.pyplot as plt

# Ingest
df = pd.read_csv("sales.csv")  # columns: product, price, quantity

# Transform
df["revenue"] = df["price"] * df["quantity"]
df = df.dropna(subset=["revenue"])  # remove missing values

# Visualize
plt.hist(df["revenue"], bins=20, color="skyblue")
plt.title("Revenue Distribution")
plt.xlabel("Revenue")
plt.ylabel("Frequency")
plt.show()
```

Program 2: Read a JSON file, normalize nested fields, and bar-plot results

```
# Concepts: JSON ingestion, flattening, grouping, bar chart

import pandas as pd
import seaborn as sns

# Ingest
data = pd.read_json("users.json")
# Example structure: {"users":[{"name":"A","age":30,"city":"NY"}, ...]}

# Transform
df = pd.json_normalize(data["users"])     # flatten
city_counts = df["city"].value_counts()

# Visualize
sns.barplot(x=city_counts.index, y=city_counts.values)
plt.title("Users per City")
plt.xticks(rotation=45)
plt.show()
```

Program 3: Read an Excel file, compute summary stats, and plot a line chart

```
# Concepts: Excel ingestion, date parsing, resampling, line plot

import pandas as pd
import matplotlib.pyplot as plt

# Ingest
df = pd.read_excel("web_traffic.xlsx")  # columns: date, visits
df["date"] = pd.to_datetime(df["date"])

# Transform
monthly = df.resample("M", on="date")["visits"].sum()

# Visualize
plt.plot(monthly.index, monthly.values, marker="o")
plt.title("Monthly Web Traffic")
plt.xlabel("Month")
plt.ylabel("Visits")
plt.grid(True)
plt.show()
```

Program 4: Load a public dataset (URL), filter data, scatter-plot relationships

```
# Concepts: URL ingestion, filtering, correlation exploration

import pandas as pd
import seaborn as sns

# Ingest
url = "https://raw.githubusercontent.com/mwaskom/seaborn-
data/master/iris.csv"
df = pd.read_csv(url)

# Transform
filtered = df[df["species"] == "setosa"]

# Visualize
sns.scatterplot(data=filtered, x="sepal_length", y="petal_length")
plt.title("Setosa: Sepal vs Petal Length")
plt.show()
```

Program 5: Load multiple files, merge them, and visualize results

```
# Concepts: multi-file ingestion, merging, aggregation, boxplot

import pandas as pd
import seaborn as sns
import glob

# Ingest (read all CSVs into one DataFrame)
files = glob.glob("data/quarter_*.csv")
df = pd.concat([pd.read_csv(f) for f in files], ignore_index=True)

# Transform
# Example: compute total sales by quarter
quarter_sales = df.groupby("quarter")["sales"].sum().reset_index()

# Visualize
sns.boxplot(data=df, x="quarter", y="sales")
plt.title("Sales Distribution per Quarter")
plt.show()
```

Program 6: Ingesting SQL Data and Computing Revenue KPIs

```
import pandas as pd
import matplotlib.pyplot as plt
import sqlite3

# 1. Create a sample SQLite database
conn = sqlite3.connect("sales.db")

# Create sample tables
products_data = {
    "product_id": [1, 2, 3, 4],
    "category": ["Electronics", "Clothing", "Clothing", "Kitchen"],
    "price": [500, 40, 60, 120]
}

orders_data = {
    "order_id": [101, 102, 103, 104],
    "product_id": [1, 2, 3, 1],
    "qty": [2, 5, 3, 1]
}
```

```
pd.DataFrame(products_data).to_sql("products", conn,
if_exists="replace", index=False)
pd.DataFrame(orders_data).to_sql("orders", conn, if_exists="replace",
index=False)

# 2. Ingest (read tables from SQL)
products = pd.read_sql("SELECT * FROM products", conn)
orders = pd.read_sql("SELECT * FROM orders", conn)

# 3. Transform (join + revenue calculation)
df = orders.merge(products, on="product_id")
df["revenue"] = df["qty"] * df["price"]

category_revenue = df.groupby("category")["revenue"].sum()

# 4. Visualize (bar chart)

category_revenue.plot(kind="bar")
plt.title("Revenue by Category")
plt.xlabel("Category")
plt.ylabel("Revenue")
plt.show()
```

Program 7: Chunk-read a large CSV and plot daily transaction totals.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Create a sample large CSV
df = pd.DataFrame({
    "txn_date": pd.date_range("2023-01-01", periods=300000, freq="min"),
    "amount": np.random.randint(100, 1000, 300000)
})
df.to_csv("large.csv", index=False)

# Chunk processing
daily = {}
for chunk in pd.read_csv("large.csv", chunksize=50000,
parse_dates=["txn_date"]):
    d = chunk.groupby(chunk["txn_date"].dt.date)["amount"].sum()
    for k, v in d.items():
        daily[k] = daily.get(k, 0) + v

# Plot
```

```
s = pd.Series(daily).sort_index()
plt.plot(s.index, s.values)
plt.title("Daily Transaction Volume")
plt.show()
```

Program 8: Read Parquet data, split it by platform, and visualize duration differences.

```
import pandas as pd
import numpy as np
import pathlib
import matplotlib.pyplot as plt

# Create sample parquet
df = pd.DataFrame({
    "platform": np.random.choice(["Android", "iOS", "Web"], 150),
    "duration": np.random.randint(20, 400, 150)
})
df.to_parquet("data.parquet")

# Read
df = pd.read_parquet("data.parquet")

# Partition
path = pathlib.Path("parts")
path.mkdir(exist_ok=True)
for p, g in df.groupby("platform"):
    g.to_parquet(path / f"{p}.parquet", index=False)

# Read back and plot
d = pd.concat([pd.read_parquet(f) for f in path.glob("*.parquet")])
plt.boxplot([d[d["platform"] == p]["duration"] for p in
d["platform"].unique()],
            labels=d["platform"].unique())
plt.title("Session Duration by Platform")
plt.show()
```