Pandas-based (no visualization) mini case study titles and case study statements:

1. **Customer Purchase Behavior Analysis Using Transactional Data**
2. **Employee Salary Data Cleaning and Aggregation for HR Insights**
3. **E-Commerce Order Data Analysis with Missing Value Handling**
4. **Airline Flight Delay Dataset: Filtering and Grouping Operations**
5. **Student Academic Performance Evaluation with Pandas Joins**
6. **Healthcare Patient Admission Records Data Transformation**
7. **Bank Loan Application Data Preprocessing and Outlier Detection**
8. **Retail Store Sales Trend Analysis Using Time-Series Data**
9. **Movie Ratings Dataset Exploration with Pivot Tables**
10. **Telecom Call Records Analysis Using GroupBy and Aggregation**

1. Customer Purchase Behavior Analysis Using Transactional Data

📌 Project: *Customer Purchase Behavior Analysis Using Transactional Data*

- Pandas-based

Problem Statement

A retail store wants to analyze customer purchase behavior using its transactional sales dataset. The goal is to clean and preprocess the data, identify patterns such as most purchased products, high-spending customers, and frequency of purchases, and prepare insights purely using **Pandas operations** (no visualization).

Objectives

1. Clean the transactional dataset (handle missing values, duplicates, and data types).

2. Perform exploratory analysis using Pandas methods.

3. Identify:

    - Top 5 most purchased products.
    - Customers with the highest total spending.
    - Average purchase amount per transaction.
    - Purchase frequency of each customer.
    - Product categories contributing the most to revenue.
4. Export aggregated results back to CSV for reporting.

Dataset Structure (`transactions.csv`)

```
TransactionID,CustomerID,ProductID,ProductCategory,Quantity,Price,TransactionDate
T001,C101,P501,Electronics,1,500,2023-01-05
T002,C102,P601,Grocery,5,20,2023-01-06
T003,C101,P701,Clothing,2,150,2023-01-06
T004,C103,P601,Grocery,10,20,2023-01-07
T005,C104,P801,Electronics,1,1200,2023-01-07
T006,C102,P701,Clothing,3,150,2023-01-08
T007,C105,P901,Furniture,1,2000,2023-01-09
T008,C101,P601,Grocery,2,20,2023-01-09
T009,C106,P701,Clothing,4,150,2023-01-10
T010,C102,P801,Electronics,1,1200,2023-01-10
T011,C107,P601,Grocery,8,20,2023-01-11
T012,C108,P502,Electronics,2,750,2023-01-12
T013,C109,P702,Clothing,1,200,2023-01-12
T014,C110,P903,Furniture,1,2500,2023-01-13
T015,C101,P601,Grocery,6,20,2023-01-13
T016,C111,P802,Electronics,1,1100,2023-01-14
T017,C112,P701,Clothing,5,150,2023-01-14
T018,C113,P601,Grocery,3,20,2023-01-15
T019,C114,P904,Furniture,2,1800,2023-01-15
T020,C115,P503,Electronics,1,950,2023-01-16
T021,C102,P702,Clothing,2,200,2023-01-16
T022,C116,P601,Grocery,4,20,2023-01-17
T023,C117,P701,Clothing,1,150,2023-01-17
T024,C118,P803,Electronics,3,600,2023-01-18
T025,C119,P905,Furniture,1,2200,2023-01-18
T026,C120,P601,Grocery,7,20,2023-01-19
T027,C121,P703,Clothing,2,175,2023-01-19
T028,C122,P804,Electronics,1,1300,2023-01-20
T029,C123,P906,Furniture,1,2100,2023-01-20
T030,C124,P601,Grocery,5,20,2023-01-21
T031,C125,P701,Clothing,2,150,2023-01-21
T032,C126,P805,Electronics,2,850,2023-01-22
T033,C127,P907,Furniture,1,2400,2023-01-22
T034,C128,P601,Grocery,9,20,2023-01-23
T035,C129,P704,Clothing,3,160,2023-01-23
T036,C130,P806,Electronics,1,1400,2023-01-24
T037,C131,P908,Furniture,2,1950,2023-01-24
T038,C132,P601,Grocery,4,20,2023-01-25
```

```
T039,C133,P701,Clothing,6,150,2023-01-25
T040,C134,P807,Electronics,1,1250,2023-01-26
T041,C135,P909,Furniture,1,2300,2023-01-26
T042,C136,P601,Grocery,3,20,2023-01-27
T043,C137,P705,Clothing,2,180,2023-01-27
T044,C138,P808,Electronics,2,950,2023-01-28
T045,C139,P910,Furniture,1,2600,2023-01-28
T046,C140,P601,Grocery,5,20,2023-01-29
T047,C141,P701,Clothing,1,150,2023-01-29
T048,C142,P809,Electronics,1,1350,2023-01-30
T049,C143,P911,Furniture,1,2000,2023-01-30
T050,C144,P601,Grocery,6,20,2023-01-31
```

Tasks to Perform (Using Pandas Only)

1. Load the dataset (`pd.read_csv`).
2. Inspect data (`head()` , `info()` , `describe()`).
3. Clean data (handle duplicates, missing values if any).
4. Add a new column: `TotalAmount = Quantity * Price` .
5. Group data by **ProductID** and calculate total quantity sold.
6. Group data by **CustomerID** to find top spenders.
7. Find average transaction value.
8. Analyze contribution of each `ProductCategory` to total revenue.
9. Sort results and export summaries to CSV.

📂 Deliverables

- Input file: `transactions.csv` (sample above, can be extended).
- Output files (via Pandas `.to_csv()`):
  - `top_products.csv`
  - `top_customers.csv`
  - `category_revenue.csv`
  - `summary_stats.csv`

```
```

2. Employee Salary Data Cleaning and Aggregation for HR Insights

📌 Project: *Employee Salary Data Cleaning and Aggregation for HR Insights*

Problem Statement

The HR department of a company maintains employee records, including salaries, departments, and job titles. However, the dataset contains duplicates, missing values, and inconsistent data formats. The goal is to clean the employee dataset using **Pandas**, perform aggregation to extract useful HR insights, and export the results for reporting.

Objectives

1. **Clean the dataset**:
    - Handle missing values in salaries and job titles.
    - Remove duplicate employee records.
    - Standardize department names and job titles.
2. **Perform Aggregations**:
    - Calculate **average salary per department**.
    - Identify **highest-paid employee** in each department.
    - Find **total salary expenditure per department**.
    - Group employees by job titles and compute their **average salaries**.
    - Count employees per department.
3. **Export cleaned and aggregated results** into CSV files.

Dataset Structure (`employees.csv`)

EmpID	Name	Department	JobTitle	Salary	JoiningDate
E001	Alice Wong	HR	HR Manager	60000	2019-03-15
E002	Bob Smith	IT	Software Engg	75000	2020-06-20
E003	Carol Lee	Finance	Accountant	55000	2018-11-12
E004	David Kim	IT	Data Scientist	95000	2021-02-10
E005	Eva Brown	Sales	Sales Executive	50000	2019-07-01

Full Dataset (50 Rows, CSV Format)

```
EmpID,Name,Department,JobTitle,Salary,JoiningDate
E001,Alice Wong,HR,HR Manager,60000,2019-03-15
E002,Bob Smith,IT,Software Engineer,75000,2020-06-20
E003,Carol Lee,Finance,Accountant,55000,2018-11-12
E004,David Kim,IT,Data Scientist,95000,2021-02-10
E005,Eva Brown,Sales,Sales Executive,50000,2019-07-01
E006,Frank White,Finance,Financial Analyst,67000,2020-01-25
E007,Grace Green,IT,Software Engineer,77000,2019-08-14
E008,Henry Adams,HR,Recruiter,45000,2021-04-01
E009,Ivy Chen,Sales,Sales Executive,52000,2020-09-12
E010,Jack Black,IT,System Administrator,68000,2018-05-19
E011,Karen Davis,Finance,Accountant,56000,2019-11-07
E012,Liam Johnson,IT,Data Scientist,98000,2021-06-30
E013,Mia Lopez,HR,HR Executive,48000,2020-02-28
E014,Noah Brown,Sales,Sales Manager,82000,2018-12-05
E015,Olivia Wilson,Finance,Financial Analyst,64000,2019-07-23
E016,Paul Harris,IT,Software Engineer,73000,2021-08-10
E017,Quinn Taylor,HR,Recruiter,47000,2019-04-17
E018,Ryan Clark,Sales,Sales Executive,51000,2021-09-01
E019,Sophia Lewis,Finance,Accountant,59000,2020-03-11
E020,Tom Hall,IT,Software Engineer,76000,2018-10-30
E021,Ursula Scott,Sales,Sales Executive,54000,2019-06-12
E022,Vincent King,Finance,Senior Accountant,72000,2021-01-19
E023,Wendy Lee,HR,HR Manager,61000,2020-05-05
E024,Xavier Young,IT,Data Scientist,97000,2019-08-28
E025,Yara Patel,Finance,Financial Analyst,66000,2018-11-15
E026,Zach Miller,Sales,Sales Manager,83000,2021-07-09
E027,Aaron Evans,IT,System Administrator,70000,2020-04-21
E028,Bella Turner,Finance,Accountant,58000,2019-02-16
E029,Carlos Gomez,Sales,Sales Executive,53000,2020-10-20
E030,Diana Ross,HR,HR Executive,49000,2018-09-25
E031,Ethan Carter,IT,Software Engineer,78000,2021-03-14
E032,Fiona Brooks,Finance,Senior Accountant,71000,2019-06-06
E033,George Hill,Sales,Sales Executive,55000,2021-11-22
E034,Hannah Moore,HR,Recruiter,46000,2020-07-18
E035,Ian Thomas,IT,Software Engineer,74000,2018-12-10
E036,Julia Adams,Finance,Financial Analyst,68000,2021-08-30
E037,Kyle Martin,Sales,Sales Executive,50000,2019-05-13
E038,Lily Perez,HR,HR Manager,62000,2018-06-27
E039,Marcus Allen,IT,Data Scientist,99000,2021-10-02
E040,Nina White,Finance,Accountant,57000,2020-09-17
E041,Omar Rivera,Sales,Sales Manager,81000,2019-03-20
E042,Pamela Scott,IT,System Administrator,69000,2018-07-08
E043,Robert Green,Finance,Senior Accountant,73000,2021-12-11
E044,Sara Clark,HR,Recruiter,48000,2020-11-29
E045,Tim Hughes,IT,Software Engineer,75000,2019-01-04
E046,Umar Shah,Finance,Financial Analyst,66000,2021-05-21
E047,Vanessa Cole,Sales,Sales Executive,52000,2018-08-15
E048,William Baker,HR,HR Executive,50000,2021-09-09
E049,Ximena Diaz,IT,Data Scientist,96000,2019-10-26
E050,Yusuf Khan,Finance,Senior Accountant,74000,2020-12-14
```

Tasks to Perform (Using Pandas Only)

1. Load dataset (`pd.read_csv`).
2. Inspect dataset (`info()` , `describe()`).
3. Remove duplicates (if any).
4. Handle missing salaries/job titles (if we simulate).
5. Standardize department and job titles (e.g., "Software Engg" → "Software Engineer").
6. Add derived columns (e.g., YearsOfService = CurrentDate – JoiningDate).
7. Group by **Department**:
   - Average salary.
   - Total salary cost.
   - Employee count.
8. Group by **JobTitle**:
   - Average salary.
   - Highest-paid employee.
9. Export summaries to CSV (`dept_summary.csv` , `job_summary.csv`).

3. E-Commerce Order Data Analysis with Missing Value Handling

📌 Project: *E-Commerce Order Data Analysis with Missing Value Handling*

Problem Statement

An e-commerce company maintains order data containing customer purchases, but the dataset has **missing values, duplicate entries, and inconsistent formats**. The goal is to clean the dataset using **Pandas**, handle missing values effectively, and perform order-level analysis to provide business insights.

Objectives

1. **Clean & Preprocess Data**
    - Handle missing values in `CustomerID`, `ProductCategory`, and `Price`.
    - Remove duplicate order entries.
    - Convert `OrderDate` to proper datetime format.
2. **Perform Analysis**
    - Calculate **total revenue per product category**.
    - Identify **top 5 customers by total spending**.
    - Find **average order value**.
    - Count orders per product category.
    - Detect orders with missing data and decide imputation or removal strategy.
3. **Export Results** into cleaned dataset and summary CSVs.

Dataset Structure (`orders.csv`)

OrderID	CustomerID	ProductID	ProductCategory	Quantity	Price	OrderDate
O001	C101	P201	Electronics	1	500	2023-01-05
O002	C102	P301	Grocery	5	20	2023-01-06
O003	NaN	P401	Clothing	2	150	2023-01-07
O004	C104	P201	Electronics	1	NaN	2023-01-08
O005	C105	P501	Furniture	1	2000	2023-01-09

Full Dataset (50 Rows, CSV Format with some missing values & duplicates)

```
OrderID,CustomerID,ProductID,ProductCategory,Quantity,Price,OrderDate
O001,C101,P201,Electronics,1,500,2023-01-05
O002,C102,P301,Grocery,5,20,2023-01-06
O003,,P401,Clothing,2,150,2023-01-07
O004,C104,P201,Electronics,1,,2023-01-08
O005,C105,P501,Furniture,1,2000,2023-01-09
O006,C106,P301,Grocery,3,20,2023-01-10
O007,C101,P401,Clothing,1,160,2023-01-11
```

```
O008,C107,P201,Electronics,2,520,2023-01-12
O009,C108,P301,Grocery,10,20,2023-01-12
O010,C109,P501,Furniture,1,2100,2023-01-13
O011,C110,P201,Electronics,1,480,2023-01-13
O012,C111,P401,Clothing,3,150,2023-01-14
O013,C112,P301,Grocery,2,,2023-01-15
O014,C113,P201,Electronics,1,530,2023-01-16
O015,C114,P502,Furniture,2,2200,2023-01-17
O016,C115,P301,Grocery,4,20,2023-01-18
O017,C116,P401,Clothing,1,155,2023-01-18
O018,C117,P201,Electronics,1,510,2023-01-19
O019,C118,P301,Grocery,6,20,2023-01-19
O020,C119,P501,Furniture,1,2050,2023-01-20
O021,C101,P401,Clothing,2,150,2023-01-21
O022,C120,P201,Electronics,1,500,2023-01-21
O023,C121,P301,Grocery,5,20,2023-01-22
O024,,P401,Clothing,3,NaN,2023-01-23
O025,C122,P201,Electronics,2,520,2023-01-23
O026,C123,P502,Furniture,1,2300,2023-01-24
O027,C124,P301,Grocery,4,20,2023-01-25
O028,C125,P401,Clothing,2,150,2023-01-25
O029,C126,P201,Electronics,1,540,2023-01-26
O030,C127,P301,Grocery,3,20,2023-01-27
O031,C128,P501,Furniture,1,2000,2023-01-28
O032,C129,P201,Electronics,1,500,2023-01-28
O033,C130,P401,Clothing,2,160,2023-01-29
O034,C131,P301,Grocery,6,20,2023-01-30
O035,C132,P201,Electronics,1,NaN,2023-01-30
O036,C133,P502,Furniture,2,2100,2023-01-31
O037,C134,P401,Clothing,3,150,2023-02-01
O038,C135,P301,Grocery,4,20,2023-02-02
O039,C136,P201,Electronics,2,500,2023-02-02
O040,C137,P401,Clothing,1,NaN,2023-02-03
O041,C138,P301,Grocery,5,20,2023-02-03
O042,C139,P501,Furniture,1,2150,2023-02-04
O043,C140,P201,Electronics,1,510,2023-02-04
O044,C141,P301,Grocery,2,20,2023-02-05
O045,C142,P401,Clothing,2,155,2023-02-05
O046,C143,P201,Electronics,1,500,2023-02-06
O047,C144,P301,Grocery,7,20,2023-02-06
O048,C145,P502,Furniture,1,2250,2023-02-07
O049,C146,P401,Clothing,3,150,2023-02-07
O050,C147,P201,Electronics,1,505,2023-02-08
```

Tasks to Perform (Using Pandas Only)

1. Load dataset (`pd.read_csv`).
2. Explore data (`info()` , `isnull().sum()` , `describe()`).

3. Handle missing values:
   - `CustomerID` : drop or mark as "Unknown".
   - `Price` : impute using category mean/median.
4. Remove duplicate orders (if any).
5. Add a column: `TotalAmount = Quantity * Price` .
6. Group by **ProductCategory**:
   - Total revenue.
   - Average order value.
   - Number of orders.
7. Find **top 5 customers by spending**.
8. Export:
   - `cleaned_orders.csv`
   - `category_summary.csv`
   - `top_customers.csv`

```
```

4. Airline Flight Delay Dataset: Filtering and Grouping Operations

📌 Project: *Airline Flight Delay Dataset: Filtering and Grouping Operations*

Problem Statement

An airline company maintains flight records including departure/arrival times and delays. The dataset contains delays caused by weather, airline operations, or technical issues. The HR Analytics team needs to analyze delays by **filtering** and **grouping operations** using Pandas (no visualization).

Objectives

1. **Clean & Explore Data**
   - Check for missing or duplicate entries.
   - Convert date/time columns to proper formats.

2. **Filtering Operations**
    - Extract flights delayed more than 30 minutes.
    - Filter flights operated by a specific airline.
    - Find all flights departing from a specific airport.
3. **Grouping Operations**
    - Group by **Airline**: average delay, total flights.
    - Group by **OriginAirport**: total flights, % delayed.
    - Group by **Date**: daily average delay.
    - Identify **top 5 routes** (Origin–Destination) with highest average delay.
4. **Export Results** into summary CSV files.

Dataset Structure (`flights.csv`)

FlightID	Airline	OriginAirport	DestinationAirport	Date	DepartureDelay	A
F001	Delta	JFK	LAX	2023-01-05	15	1
F002	United	ORD	SFO	2023-01-05	45	5
F003	Delta	ATL	MIA	2023-01-06	-5	0
F004	American	DFW	LAX	2023-01-06	60	5
F005	Southwest	LAX	LAS	2023-01-07	10	5

Full Dataset (50 Rows, CSV Format)

```
FlightID,Airline,OriginAirport,DestinationAirport,Date,DepartureDelay,ArrivalDelay,Dis
F001,Delta,JFK,LAX,2023-01-05,15,10,2475
F002,United,ORD,SFO,2023-01-05,45,50,1846
F003,Delta,ATL,MIA,2023-01-06,-5,0,595
F004,American,DFW,LAX,2023-01-06,60,55,1235
F005,Southwest,LAX,LAS,2023-01-07,10,5,236
F006,Delta,JFK,ORD,2023-01-07,25,20,740
F007,United,SFO,JFK,2023-01-08,75,80,2586
F008,American,ORD,LAX,2023-01-08,20,15,1744
F009,Southwest,LAS,PHX,2023-01-09,5,0,255
F010,Delta,LAX,JFK,2023-01-09,90,85,2475
```

```
F011,United,ATL,ORD,2023-01-10,30,25,606
F012,American,MIA,DFW,2023-01-10,15,10,1121
F013,Southwest,LAX,SFO,2023-01-11,40,35,337
F014,Delta,JFK,BOS,2023-01-11,-10,-5,187
F015,United,LAX,SEA,2023-01-12,60,55,954
F016,American,DFW,JFK,2023-01-12,20,25,1391
F017,Southwest,PHX,LAS,2023-01-13,0,0,255
F018,Delta,ATL,LAX,2023-01-13,35,30,1946
F019,United,SFO,ORD,2023-01-14,25,20,1846
F020,American,LAX,DFW,2023-01-14,50,45,1235
F021,Southwest,LAS,LAX,2023-01-15,15,10,236
F022,Delta,BOS,JFK,2023-01-15,5,0,187
F023,United,ORD,ATL,2023-01-16,10,15,606
F024,American,JFK,MIA,2023-01-16,65,60,1090
F025,Southwest,SFO,LAX,2023-01-17,35,30,337
F026,Delta,LAX,ATL,2023-01-17,55,50,1946
F027,United,JFK,SFO,2023-01-18,20,25,2586
F028,American,MIA,ORD,2023-01-18,15,10,1197
F029,Southwest,LAX,PHX,2023-01-19,5,0,370
F030,Delta,JFK,LAX,2023-01-19,80,75,2475
F031,United,ORD,SEA,2023-01-20,40,35,1721
F032,American,DFW,LAX,2023-01-20,25,30,1235
F033,Southwest,LAS,LAX,2023-01-21,10,5,236
F034,Delta,BOS,ATL,2023-01-21,0,-5,946
F035,United,SFO,JFK,2023-01-22,70,65,2586
F036,American,JFK,ORD,2023-01-22,30,25,740
F037,Southwest,LAX,LAS,2023-01-23,5,0,236
F038,Delta,ATL,MIA,2023-01-23,10,15,595
F039,United,ORD,LAX,2023-01-24,50,45,1744
F040,American,DFW,SEA,2023-01-24,20,15,1660
F041,Southwest,PHX,LAX,2023-01-25,15,10,370
F042,Delta,LAX,JFK,2023-01-25,100,95,2475
F043,United,ATL,SFO,2023-01-26,35,30,2139
F044,American,MIA,LAX,2023-01-26,25,20,2342
F045,Southwest,LAX,SFO,2023-01-27,30,25,337
F046,Delta,JFK,DFW,2023-01-27,45,50,1391
F047,United,SEA,ORD,2023-01-28,15,10,1721
F048,American,LAX,JFK,2023-01-28,85,80,2475
F049,Southwest,LAS,PHX,2023-01-29,0,-5,255
F050,Delta,ATL,LAX,2023-01-29,20,15,1946
```

Tasks to Perform (Using Pandas Only)

1. Load dataset (`pd.read_csv`).
2. Explore (`head()` , `info()` , `describe()`).
3. Filtering:

- Flights delayed more than 30 mins.
    - Flights from `JFK`.
    - Flights by `Delta`.

4. Grouping:

    - By **Airline** → average delay, total flights.
    - By **OriginAirport** → total flights, % delayed.
    - By **Date** → daily average delay.
    - By **Origin–Destination route** → avg delay.

5. Export summaries:

    - `airline_summary.csv`
    - `airport_summary.csv`
    - `route_summary.csv`

5. Student Academic Performance Evaluation with Pandas Joins

📌 Project Definition

Title: "Student Academic Performance Evaluation with Pandas Joins"

Type: Pandas-based (no visualization) mini case study

Objective: This project focuses on evaluating students' academic performance using multiple datasets that contain student details, course enrollments, and grades. The primary goal is to demonstrate the use of **Pandas joins (merge, join, concat)** to combine datasets and generate meaningful insights such as student GPA, performance trends, and subject-wise average scores.

Key Pandas Operations Used:

- `pd.merge()` with different join types (inner, left, right, outer)
- `groupby()` for aggregations
- `fillna()` to handle missing grades
- `sort_values()` for ranking students
- `drop_duplicates()` to ensure clean student-course relationships

Expected Outcomes:

- Student GPA calculation.
- Identifying top-performing students across courses.

- Subject-wise performance statistics.
- Detecting students missing grades in certain subjects.

🗂️ Datasets

1. `students.csv`

```
StudentID,Name,Department
S001,Alice,Computer Science
S002,Bob,Mathematics
S003,Charlie,Physics
S004,Diana,Computer Science
S005,Ethan,Mathematics
```

2. `courses.csv`

```
CourseID,CourseName,Department
C101,Data Structures,Computer Science
C102,Algorithms,Computer Science
C201,Calculus,Mathematics
C202,Linear Algebra,Mathematics
C301,Quantum Mechanics,Physics
```

3. `enrollments.csv`

```
StudentID,CourseID,Semester
S001,C101,Fall2023
S001,C102,Fall2023
S002,C201,Fall2023
S002,C202,Fall2023
S003,C301,Fall2023
S004,C101,Fall2023
S005,C202,Fall2023
```

4. `grades.csv`

```
StudentID,CourseID,Grade
S001,C101,85
S001,C102,90
S002,C201,78
S002,C202,
S003,C301,88
```

```
S004,C101,92
S005,C202,80
```

⚡ With these datasets, you can demonstrate:

- Joining `students` with `enrollments` to see which students took which courses.
- Further joining with `grades` to analyze performance.
- Handling missing values in grades (like Bob's missing grade in Linear Algebra).
- Aggregating grades to compute GPA or department-wise performance.

✅ Step-by-Step Pandas Tasks (Problem Statements):

Step 1 – Load Data

1. Load `students.csv` and `marks.csv` into Pandas DataFrames.
2. Inspect the first 5 rows of each dataset.
3. Check the shape (rows, columns) of each dataset.

Step 2 – Data Cleaning

4. Check for missing values in both datasets.
5. Fill missing marks with `0` (assuming absent students).
6. Drop duplicate student entries (if any).

Step 3 – Joins & Merging

7. Perform an **inner join** on `StudentID` between `students.csv` and `marks.csv` to combine student details with their marks.
8. Perform a **left join** to list all students (even if they don't have marks recorded).
9. Perform a **right join** to see only students who have marks data but may not exist in the student records.

Step 4 – Aggregations & Grouping

10. Group data by `Name` and calculate **total marks across subjects**.
11. Group data by `Name` and calculate **average marks** per student.
12. Group data by `Subject` and calculate the **highest, lowest, and average marks**.

Step 5 – Filtering & Ranking

13. Find students who scored **above 90 in Math**.
14. Find students with **total marks greater than 250**.
15. Rank students by their **average marks** (highest to lowest).

Step 6 – Derived Columns

16. Create a new column `Result` → mark students as `"Pass"` if average marks ≥ 40, otherwise `"Fail"`.
17. Create a new column `Grade` based on average marks:

- ≥ 90 → `"A"`
- 75–89 → `"B"`
- 60–74 → `"C"`
- 40–59 → `"D"`
- < 40 → `"F"`

Step 7 – Final Insights

18. Display **top 3 performers** by average marks.
19. Count how many students got `"Pass"` and `"Fail"`.
20. Find the subject with the **highest overall average score**.

6. Healthcare Patient Admission Records Data Transformation

📌 Project Definition

Title: *Healthcare Patient Admission Records Data Transformation*

Type: *Pandas-based (no visualization) mini case study*

Description: The project focuses on transforming and cleaning patient admission records in a healthcare system. Using **Pandas**, you will perform data wrangling tasks like handling missing values, formatting dates, filtering based on conditions, grouping for analysis, and generating new derived columns. The goal is to prepare the dataset for insights such as patient stay duration, department utilization, and admission trends.

📝 Step-by-Step Pandas Tasks (Problem Statements)

1. **Load Dataset:** Import the patient admission dataset into a Pandas DataFrame.
2. **Check Data Quality:** Identify missing values, incorrect data types, and duplicates.
3. **Handle Missing Values:**
   - Fill missing values in `Diagnosis` with `"Unknown"`.
   - Fill missing values in `DischargeDate` with today's date (simulate ongoing admission).
4. **Data Type Conversion:** Convert `AdmissionDate` and `DischargeDate` to datetime objects.
5. **Create Derived Column:** Add a new column `StayDuration` = `DischargeDate - AdmissionDate` (in days).
6. **Filter Data:** Find patients admitted to `"Cardiology"` with stay duration greater than 5 days.
7. **Group and Aggregate:**
   - Count number of admissions per `Department`.
   - Find the average stay duration per `Department`.
8. **Sort Data:** Sort patients by longest `StayDuration`.
9. **Remove Duplicates:** Ensure no duplicate patient admissions exist (same `PatientID`, `AdmissionDate`).
10. **Export Transformed Data:** Save the cleaned and transformed dataset into a new CSV file `transformed_admissions.csv`.

📂 Sample Dataset (CSV Format)

Filename: `patient_admissions.csv`

```
PatientID,Name,Age,Gender,Department,Diagnosis,AdmissionDate,DischargeDate
P001,John Doe,45,M,Cardiology,Heart Attack,2023-03-01,2023-03-10
P002,Jane Smith,34,F,Neurology,Stroke,2023-03-05,2023-03-12
P003,Michael Lee,29,M,Orthopedics,Fracture,2023-03-08,
P004,Sarah Kim,52,F,Cardiology,,2023-03-10,2023-03-15
P005,David Chen,61,M,Oncology,Cancer,2023-03-11,2023-03-25
P006,Linda Brown,47,F,Neurology,Epilepsy,2023-03-12,2023-03-14
P007,Robert Wilson,39,M,Cardiology,Arrhythmia,2023-03-14,2023-03-20
P008,Emily Davis,26,F,Orthopedics,Dislocation,2023-03-15,2023-03-18
P009,Daniel White,58,M,Oncology,,2023-03-16,2023-03-28
P010,Anna Scott,40,F,Cardiology,Hypertension,2023-03-18,
```

7. Bank Loan Application Data Preprocessing and Outlier Detection

📌 Project Definition

- **Title:** "*Bank Loan Application Data Preprocessing and Outlier Detection*"
- **Type:** *Pandas-based (no visualization) mini case study*

Objective

The project focuses on cleaning, preprocessing, and analyzing a loan application dataset to detect inconsistencies, missing values, and outliers. The final goal is to prepare the dataset for further modeling (not included here).

✅ Step-by-Step Pandas Tasks (Problem Statements)

1. **Load the dataset** into a Pandas DataFrame.
2. **Inspect basic info**: Check shape, column names, and data types.
3. **Identify missing values**: Count missing values in each column.
4. **Handle missing values**:
    - Fill missing `Income` with median.
    - Fill missing `CreditScore` with mean.
    - Drop rows where `LoanAmount` is missing.
5. **Remove duplicate records** based on `ApplicationID`.
6. **Standardize categorical values** (e.g., ensure `LoanStatus` is either "Approved" or "Rejected").
7. **Detect outliers**:
    - For `Income` and `LoanAmount`, calculate IQR (Interquartile Range) and detect extreme values.
    - Mark them for review.
8. **Filter records** where `LoanAmount` > 2 × `Income` (potentially high risk).
9. **Group & aggregate**: Find average `LoanAmount` by `EmploymentStatus`.
10. **Export cleaned dataset** into a new CSV file named `cleaned_loan_data.csv`.

📁 Sample Dataset (CSV Format)

`bank_loan_applications.csv`

```
ApplicationID,Name,Age,EmploymentStatus,Income,CreditScore,LoanAmount,LoanStatus
A001,John Smith,29,Employed,45000,720,15000,Approved
A002,Sarah Johnson,35,Self-Employed,60000,680,25000,Approved
A003,Michael Lee,42,Employed,52000,,18000,Rejected
A004,Emily Davis,28,Unemployed,,650,8000,Rejected
A005,David Wilson,50,Employed,85000,710,40000,Approved
A006,Linda Martinez,31,Employed,47000,600,25000,Rejected
A007,James Anderson,38,Self-Employed,120000,730,60000,Approved
A008,Mary Thomas,45,Employed,95000,710,100000,Approved
A009,Robert Jackson,52,Retired,30000,500,20000,Rejected
A010,Patricia White,33,Employed,48000,650,25000,Approved
A011,John Smith,29,Employed,45000,720,15000,Approved
```

```

```

8. Retail Store Sales Trend Analysis Using Time-Series Data

📌 Project Definition

Title: "Retail Store Sales Trend Analysis Using Time-Series Data"

Type: Pandas-based (no visualization) mini case study

Objective: The project aims to analyze daily sales records of a retail store to identify trends, seasonality, and store performance. Using **Pandas**, you will perform data cleaning, time-series manipulations, filtering, grouping, and aggregations to generate actionable insights.

📝 Step-by-Step Pandas Tasks (Problem Statements)

Step 1 – Load Data

1. Load the dataset `retail_sales.csv` into a Pandas DataFrame.
2. Inspect the first 5 rows and data types.
3. Check the number of rows and columns (`shape`).

Step 2 – Data Cleaning

4. Identify missing values in `Sales` and `StoreID`.
5. Fill missing `Sales` with `0`.
6. Drop duplicates if any.
7. Convert `Date` column to datetime format.

Step 3 – Time-Series Manipulation

8. Set `Date` as the index of the DataFrame.
9. Create a new column `Weekday` representing the day of the week.
10. Create a new column `Month` representing the month number.

Step 4 – Filtering & Conditional Operations

11. Filter data for `StoreID = 101`.
12. Find all days where `Sales > 5000`.
13. Identify weekends (`Saturday` and `Sunday`) with sales above 4000.

Step 5 – Grouping & Aggregations

14. Group by `StoreID` and calculate **total sales**.
15. Group by `Month` and calculate **average daily sales**.
16. Group by `Weekday` and find **average sales per day**.
17. Identify the **top 3 stores by total sales**.

Step 6 – Derived Columns

18. Create a new column `CumulativeSales` per store (cumulative sum).
19. Create a new column `SalesCategory` where:

- Sales ≥ 5000 → `"High"`
- Sales 3000–4999 → `"Medium"`
- Sales < 3000 → `"Low"`

Step 7 – Export Results

20. Export the cleaned dataset as `cleaned_retail_sales.csv`.
21. Export **store-wise total sales** as `store_sales_summary.csv`.
22. Export **weekday-wise average sales** as `weekday_sales_summary.csv`.

📂 Sample Dataset (CSV Format)

Filename: `retail_sales.csv`

```
Date,StoreID,Sales
2023-01-01,101,4500
2023-01-01,102,5200
2023-01-01,103,3100
2023-01-02,101,4700
2023-01-02,102,5000
2023-01-02,103,2800
2023-01-03,101,5200
2023-01-03,102,5300
2023-01-03,103,3000
2023-01-04,101,4800
2023-01-04,102,4900
2023-01-04,103,3500
2023-01-05,101,6000
2023-01-05,102,6200
2023-01-05,103,4000
2023-01-06,101,5500
2023-01-06,102,5800
2023-01-06,103,4200
2023-01-07,101,5000
2023-01-07,102,5100
2023-01-07,103,3900
2023-01-08,101,4800
2023-01-08,102,4950
2023-01-08,103,3600
2023-01-09,101,4700
2023-01-09,102,5050
2023-01-09,103,3700
2023-01-10,101,4900
2023-01-10,102,5200
2023-01-10,103,3800
```

This dataset contains **10 days of sales data for 3 stores** and includes opportunities to practice:

- Time-series handling (`Date` as datetime index)
- Filtering (`Sales > threshold`)
- Grouping (`StoreID`, `Weekday`, `Month`)
- Cumulative sum and derived columns

9. Movie Ratings Dataset Exploration with Pivot Tables

📌 Project Definition

Title: "Movie Ratings Dataset Exploration with Pivot Tables"

Type: Pandas-based (no visualization) mini case study

Objective: This project focuses on analyzing a movie ratings dataset to extract insights about user preferences, movie popularity, and genre performance. The goal is to practice **Pandas operations**, especially **pivot tables**, joins, grouping, and aggregation.

📝 Step-by-Step Pandas Tasks (Problem Statements)

Step 1 – Load Data

1. Load the datasets (`movies.csv` , `ratings.csv` , `users.csv`) into Pandas DataFrames.
2. Inspect the first 5 rows and check data types.
3. Check dataset shapes and column names.

Step 2 – Data Cleaning

4. Check for missing values in all datasets.
5. Fill or handle missing ratings appropriately (e.g., leave as `NaN` or fill with 0 for analysis).
6. Remove duplicate entries if any exist.

Step 3 – Merge Datasets

7. Merge `ratings` with `movies` using `MovieID` .
8. Merge the result with `users` using `UserID` .
9. Verify that all merges are successful (check row counts).

Step 4 – Pivot Table Analysis

10. Create a pivot table showing **average rating per movie**.
11. Create a pivot table showing **average rating per genre**.
12. Create a pivot table showing **average rating per user**.

13. Create a pivot table showing **count of ratings per movie** to identify most rated movies.

Step 5 – Filtering & Sorting

14. Find movies with **average rating ≥ 4.0**.
15. Find users who rated **more than 5 movies**.
16. Find top 5 movies by **number of ratings**.
17. Identify movies with **highest and lowest average ratings**.

Step 6 – Derived Columns

18. Add a column `RatingCategory`:

- Rating ≥ 4 → `"High"`
- Rating 3–3.9 → `"Medium"`
- Rating < 3 → `"Low"`

19. Add a column `IsPopular` for movies with **more than 10 ratings** → `"Yes"` / `"No"`.

Step 7 – Export Results

20. Export pivot tables for:

- `movie_avg_ratings.csv`
- `genre_avg_ratings.csv`
- `user_avg_ratings.csv`

21. Export the cleaned and merged dataset as `cleaned_movie_ratings.csv`.

📂 Sample Datasets (CSV Format)

1. `movies.csv`

```
MovieID,Title,Genre
M001,Inception,Sci-Fi
M002,Titanic,Romance
M003,The Godfather,Crime
M004,Avengers,Action
M005,Interstellar,Sci-Fi
```

2. `ratings.csv`

```
UserID,MovieID,Rating,Timestamp
U001,M001,5,2023-01-01
U002,M001,4,2023-01-02
U003,M002,5,2023-01-03
U001,M003,3,2023-01-04
U002,M004,4,2023-01-05
U003,M005,5,2023-01-06
U004,M001,4,2023-01-07
U004,M003,2,2023-01-08
U005,M002,3,2023-01-09
```

3. `users.csv`

```
UserID,Name,Age,Gender
U001,Alice,25,F
U002,Bob,30,M
U003,Charlie,22,M
U004,Diana,28,F
U005,Ethan,35,M
```

This setup allows you to practice:

- **Merging multiple datasets**
- **Pivot tables** for aggregation
- **Filtering & ranking**
- **Derived columns** for categorization
- **Exporting results**

```

```

10. Telecom Call Records Analysis Using GroupBy and Aggregation

📌 Project Definition

Title: "Telecom Call Records Analysis Using GroupBy and Aggregation"

Type: Pandas-based (no visualization) mini case study

Objective: This project focuses on analyzing telecom call records to gain insights about call durations, frequent callers, and network usage patterns. The goal is to practice **Pandas data wrangling**, **groupby operations**, and **aggregations** without visualization.

📝 Step-by-Step Pandas Tasks (Problem Statements)

Step 1 – Load Data

1. Load the dataset `call_records.csv` into a Pandas DataFrame.
2. Inspect the first 5 rows and check data types.
3. Check dataset shape and column names.

Step 2 – Data Cleaning

4. Identify missing values in any column.
5. Fill missing `CallDuration` values with 0.
6. Drop duplicates based on `CallID`.
7. Convert `CallStartTime` and `CallEndTime` to datetime objects.

Step 3 – Derived Columns

8. Create a new column `CallDurationMinutes` = (`CallEndTime - CallStartTime`) in minutes.

9. Create a new column `CallTypeCategory` based on `CallType`:

    - `'Local'` → `"Domestic"`
    - `'STD'` or `'ISD'` → `"International"`

Step 4 – Filtering

10. Filter calls where `CallDurationMinutes > 10`.
11. Filter calls made by a specific `CustomerID` (e.g., `C001`).
12. Find all international calls (`CallTypeCategory = "International"`) with duration > 30 mins.

Step 5 – Grouping & Aggregation

13. Group by `CustomerID` and calculate:

    - Total call duration
    - Average call duration
    - Number of calls

14. Group by `CallTypeCategory` and calculate total and average call duration.

15. Group by `CallDate` and calculate **daily total call duration**.

16. Identify the **top 5 customers by total call duration**.

Step 6 – Export Results

17. Export customer-wise summary as `customer_call_summary.csv`.
18. Export call type summary as `calltype_summary.csv`.
19. Export daily call summary as `daily_call_summary.csv`.

🗂️ Sample Dataset (CSV Format)

Filename: `call_records.csv`

```
CallID,CustomerID,CallType,CallStartTime,CallEndTime,CallDuration
CL001,C001,Local,2023-08-01 09:00:00,2023-08-01 09:12:00,12
CL002,C002,STD,2023-08-01 10:05:00,2023-08-01 10:35:00,30
CL003,C001,ISD,2023-08-01 11:00:00,2023-08-01 11:50:00,50
CL004,C003,Local,2023-08-01 12:30:00,2023-08-01 12:40:00,10
CL005,C002,Local,2023-08-02 09:15:00,2023-08-02 09:45:00,30
CL006,C004,STD,2023-08-02 10:20:00,2023-08-02 10:50:00,30
CL007,C001,Local,2023-08-02 11:10:00,2023-08-02 11:30:00,20
CL008,C003,ISD,2023-08-03 08:00:00,2023-08-03 08:50:00,50
CL009,C004,Local,2023-08-03 09:40:00,2023-08-03 09:55:00,15
CL010,C002,STD,2023-08-03 10:30:00,2023-08-03 11:00:00,30
```

This setup allows you to practice:

- **Datetime conversions** (`CallStartTime` / `CallEndTime`)
- **Derived columns** (`CallDurationMinutes`, `CallTypeCategory`)
- **Filtering based on conditions**
- **Grouping and aggregation** for summary statistics
- **Exporting results** to CSV