Lecture 16 | pandas Part II – Exercises

Guided exercise | Two laboratory exercises | Three independent exercises

Working with the exercise materials

Purpose and scope

These problems develop the Part II skills introduced in class: sorting, grouping, named aggregations, merging related data, reshaping tables, covariance and CSV export

Your files

Open the folder of one exercise and run its script from that folder so relative paths can locate the input CSV files. The first three folders include starter.py; independent exercises provide the data and task description

Expected output

Check printed intermediate tables, interpret their meaning and verify any requested CSV files. Existing input files must remain unchanged

Guided exercise | Portfolio volatility

From long daily return records to a covariance matrix and a risk report

Guided | Context, data and objective

Scenario

A small fund holds two assets, A and B, with portfolio weights 60% and 40%. Its analyst receives one recorded return per asset and trading day in `daily_returns.csv`

The problem

The records are stored in **long format** (date, asset, return). For a portfolio calculation, we need both assets as columns, aligned by date. We want the sample covariance matrix and the resulting portfolio variance and volatility

Input example

```
date,asset,return
2026-03-02,A,0.01
2026-03-02,B,0.02
2026-03-03,A,-0.02
2026-03-03,B,0.01
```

Interpretation

A return of 0.01 represents +1%; negative values indicate losses. The file contains eight records: two assets observed on four dates

Guided | From input records to a wide table

Read and inspect

Read `daily_returns.csv` with `pd.read_csv()` and print the raw data. Confirm that each pair (date, asset) has exactly one return: this makes `.pivot()` appropriate instead of `.pivot_table()`

Reshape

Apply `.pivot(index="date", columns="asset", values="return")` to create one row per date and one column per asset. Keep the values unchanged

Expected intermediate table

date	A	B
2026-03-02	0.01	0.02
2026-03-03	-0.02	0.01
2026-03-04	0.03	-0.01
2026-03-05	0.00	0.02

Important check

Explicitly select the columns in the order ["A", "B"] before moving to NumPy: the weight vector [0.60, 0.40] must refer to the same asset order

Guided | Calculate and export portfolio risk

Covariance

Use `.cov()` on the wide DataFrame. The diagonal entries describe the sample variance of each asset; the off-diagonal entries describe their sample covariance

Matrix calculation

Use `.to_numpy()` to obtain the covariance array Sigma. With weights `w = np.array([0.60, 0.40])`, compute `variance = w @ Sigma @ w`, then `volatility = np.sqrt(variance)`

Report

Build a one-row DataFrame with columns `portfolio_variance` and `portfolio_volatility`, and save it to `portfolio_risk.csv` using `.to_csv(index=False)`

Check before finishing

The variance is nonnegative; the volatility is its square root and is expressed in return units, not in monetary units. Explain why changing the order of the asset columns without changing the weights would invalidate the calculation

Guided | Starter and completion checks

Prepared for you

`starter.py` already imports pandas and NumPy, reads the CSV and defines `asset_order` and weights. It deliberately leaves the reshaping, risk calculation and export operations for you

Complete these operations

Create wide; select its columns; compute the covariance table and NumPy array; compute variance and volatility; create and export the risk report

Self-check

The resulting wide table must have shape (4, 2). The covariance matrix must have shape (2, 2). Your exported file must contain one row of numerical results with two named columns

Laboratory exercises | Independent with support

Use the hints and intermediate checks, but write the pandas operations yourself

Lab 1 | Daily trading activity: the problem

Scenario

A trading desk receives individual transaction records. Several trades may involve the same asset on the same date. The desk needs a compact daily volume matrix and a ranking of assets by total volume

Input file

`trades.csv` contains date, asset and volume. Example: on 2026-04-01 asset A appears twice, with volumes 10 and 5, while B has volume 8

Required results

Produce `daily_volume.csv` with one row per date and one column per asset; produce `asset_totals.csv` with one row per asset and its total traded volume, sorted from largest to smallest

Key interpretation

A missing asset/date combination means no recorded trade for this dataset. It is therefore reasonable here, unlike for missing asset prices, to represent the daily traded volume with zero

Lab 1 | Implementation requirements and hints

Reshape and aggregate

Read `trades.csv`. A plain `.pivot()` is unsuitable because asset A on `2026-04-01` has repeated observations. Use `.pivot_table()` with date as index, asset as columns, volume as values and `aggfunc="sum"`

Handle absent combinations

Set `fill_value=0`. Check that the cell for (`2026-04-01, A`) equals 15 and that missing asset/date combinations are shown as 0

Rank assets

Independently use `.groupby("asset")` and `.sum()` to compute volumes across all days, followed by `.sort_values(..., ascending=False)`. Keep asset as an ordinary column in the ranking

Export carefully

When saving the daily matrix, retain the date index; when saving the asset ranking, exclude its default integer index. Check that the original input CSV has not been modified

Lab 1 | Expected checks

Daily matrix: selected rows

date	A	B	C
2026-04-01	15	8	0
2026-04-02	7	0	10
2026-04-03	0	3	0

Totals by asset

A: 22 units; C: 10 units; B: 11 units. The sorted ranking must therefore be A, B, C

Explain

Why would `.pivot()` fail on the raw file? Which aggregation rule would be appropriate for total traded volume? Why should missing volumes not automatically be treated as missing prices?

Lab 2 | Portfolio valuation: the problem

Scenario

A wealth-management desk has holdings by client account. A separate asset catalog stores the sector and unit price for each known asset. One holding refers to an asset that is not present in the catalog

Inputs

`holdings.csv` contains account, asset, units. The starter creates a second DataFrame, assets, with the reference columns asset, sector, price

Required outputs

Create `unknown_assets.csv` with unmatched holdings and `account_values.csv` showing the total value and number of valid positions per account, ordered by descending value

Important rule

Never assume an unknown price is zero. Flag the unmatched record separately and exclude it only from the valuation summary

Lab 2 | Operations to implement

Join the tables

Use a `left .merge()` on asset to preserve every holding. Print the joined DataFrame and identify which rows have a missing reference price

Detect exceptions

Use the Boolean mask produced by `.isna()` to select unmatched holdings. For unknown_assets.csv retain only account, asset and units

Calculate and summarize

For matched holdings, calculate `market_value = units * price`. Group by account and use named `.agg()` to produce `total_value` (sum of `market_value`) and positions (count of asset)

Sort and export

Use `.sort_values()` on `total_value`, highest first, and save both reports with `index=False`

Lab 2 | Example and verification

Example of reference matching

Account P1 owns 4 units of A and 3 of B. With catalog prices 100 and 80, its matched market value is 4×100 + 3×80 = 640

Unknown record

P2 contains 2 units of asset X, absent from the catalog. This holding belongs in unknown_assets.csv but must not be counted in the matched valuation

Expected account totals

P1: 640 with 2 valid positions; P2: 300 with 1 valid position; P3: 260 with 2 valid positions

Discussion

Why is a left join preferable to an inner join when preparing an exception report? What would be lost if unmatched holdings were discarded immediately?

Independent exercises | Increasing difficulty

Use what you learned in Part II and consolidate earlier Python fundamentals

Independent 1 | Bond inventory: context and task

Scenario

An analyst has a simple inventory of bonds. Each record lists a bond identifier, number of units held, market price per unit, and annual coupon rate expressed as a percentage of a face value of 100

Input

`bonds.csv` with columns `bond, units, price, coupon_pct`. For example B1 has 10 units, price 98 and coupon_pct 3.0

Required calculations

For each bond compute market_value = units × price and annual_coupon = units × 100 × coupon_pct / 100. Add both as new DataFrame columns

Output

Sort the full table by `market_value` (descending), display it and save `bond_inventory_report.csv` with all original and calculated columns

Independent 1 | Implementation and result checks

Python fundamentals

Perform the two calculations using an ordinary Python for loop, collecting results in lists before assigning them as new DataFrame columns. You may use `.iloc[]` to access individual rows

Example

For B1, market_value = 10 × 98 = 980; annual_coupon = 10 × 100 × 3.0/100 = 30

Verification

The exported file contains four bonds, all six columns and no unwanted numerical index. Explain why the coupon calculation uses face value 100 while market valuation uses the observed price

Independent 2 | Sector risk limits

Scenario

A risk team evaluates the exposure of five portfolio positions. It uses simplified sector multipliers: Tech 1.5, Energy 1.2 and Finance 1.0

Data and formula

`positions.csv` contains asset, sector, units and price. Compute market_value = units × price, then risk_exposure = market_value × risk_factor

Required data construction

Define a Python dictionary of sector multipliers, then use a loop to construct a separate reference DataFrame containing sector and `risk_factor`. Merge it with the positions table

Required report

Group by sector and calculate `total_market_value` and `total_risk_exposure`. Add a Boolean column `limit_exceeded`, True if the aggregated risk exposure is strictly greater than 1500, then sort descending by risk exposure

Independent 2 | Deliverable and checks

Output file

Save `sector_risk_report.csv` with exactly these columns: sector, `total_market_value, total_risk_exposure, limit_exceeded`

Example calculation

Asset A belongs to Tech and has 10 units at price 100. Its market value is 1000 and its risk exposure is 1000 × 1.5 = 1500

Verification

The Tech sector contains two assets, so its final total risk exposure must combine both records before testing the limit. No single-position limit is requested

Design choice

The report must keep the sector name as an ordinary column and must not include an artificial CSV index

Independent 3 | Trade fees: exam-style problem

Scenario

A brokerage firm must prepare daily fee reports from individual orders. Orders can repeat the same asset on one date. A separate asset catalog provides a unit price and sector for every asset in the order file

Input files

`orders.csv` contains `date, asset, units`. `asset_catalog.csv` contains `asset, sector, unit_price`. All identifiers in `orders.csv` are present in the catalog

Fee policy

The transaction value is units × unit_price. Fees are Tech: 1.0%, Energy: 1.5%, Finance: 1.2% of transaction value

Task

Implement a Python function `fee_rate(sector)` that returns the appropriate decimal rate. Use a `for` loop to calculate the fee of every joined order and add a fee column

Independent 3 | Reports and acceptance criteria

Daily report

Build `daily_fees.csv` with one row per date, one column per sector and total fees for that date/sector combination. Repeated records must be summed; missing combinations represent zero fees

Sector report

Build `sector_fees.csv` with columns sector and fee. Each row gives the total fee across all input dates for that sector, sorted from largest to smallest fee

Example

A Tech order of 3 units priced at 100 produces a transaction value of 300 and a fee of 3.00. A second Tech order on the same date must contribute to that date's Tech total

Constraints and completion

Use the two supplied CSV inputs, your own `fee_rate()` function, pandas operations and basic Python iteration. Save both reports without modifying the input files. Aim to complete the problem in about 20 minutes

Independent 3 | Output structure

Illustrative daily report

```
date                Energy    Finance      Tech
2026-05-04          [total]      0         [total]
2026-05-05          [total]   [total]      [total]
```

Illustrative sector report

```
sector,fee
Tech,[aggregated fee]
Energy,[aggregated fee]
Finance,[aggregated fee]
```

Read carefully

The placeholders above illustrate the required file structure, not numerical answers. Your program must compute the actual numbers using the supplied CSV data

Important distinction

The daily table groups records simultaneously by date and sector; the sector report groups all dates together for each sector