Advanced Pandas Homework: Analyzing NYC Flights 2013 Dataset

September 15, 2025

1 Overview

This homework assignment focuses on advanced pandas techniques using the real-world NYC Flights 2013 dataset (∼336,000 records of flights from NYC airports). The tasks build progressively, requiring data loading, cleaning, outlier detection, grouping/aggregation, reshaping, window functions, and complex filtering.

2 Learning Objectives

- Efficient data handling with categorical types and vectorized operations.
- Custom aggregations and lambdas in groupby.
- Reshaping with pivot/melt.
- Window functions for ranking and cumulatives.
- Scalable analysis on large datasets (∼300k rows after cleaning).

3 Requirements

- Python 3.8+ with pandas (≥1.5), numpy (≥1.21).
- Dataset: Download from `https://www.openintro.org/data/csv/nycflights.csv` (or use `pd.read_csv` in code).
- Submit
  - A Python file ({student_id}.py based on nycflights_skeleton.py) containing your code **- due Sep. 23**
  - A report ({student_id}.pdf) with outputs from your code, comparisons with reference code, and explanations for each task **- due Sep. 27**
- Grading: Code (50%) + Report (50%).

4 Dataset Columns (Key Ones)

- `year`, `month`, `day`: Flight date.
- `dep_time`, `arr_time`, `sched_dep_time`, `sched_arr_time`: Times as HHMM integers.
- `dep_delay`, `arr_delay`: Delays in minutes (NaN = cancelled).
- `carrier`: Airline code (e.g., 'UA', 'AA').
- `origin`, `dest`: Airports (e.g., 'JFK', 'LGA').

- air_time: Flight duration (minutes).
- distance: Miles.
- Others: flight, tailnum, hour, minute.

NaNs in arr_delay indicate cancelled/diverted flights—handle as specified.

5 Task

5.1 Task 1: Data Loading, Type Conversion, and Initial Cleaning

Load the CSV. Convert dep_time and arr_time to hours-of-day (float, e.g., 14.5 for 14:30) using vectorized ops. Make carrier and origin categorical. Drop rows with NaN arr_delay and reset index.

- Compute: Shape after cleaning, memory usage (MB), dtypes.
- **Expected:** Shape (x, 18) [after adding time cols]; Memory ∼45 MB; carrier/origin as category.

5.2 Task 2: Outlier Detection and Robust Cleaning

Use IQR to detect outliers in arr_delay (lower = Q1 - 1.5×IQR, upper = Q3 + 1.5×IQR). Filter to non-outliers.

- Add delay_category using pd.cut: bins=$[-\infty, 15, 60, \infty]$, labels=['short', 'medium', 'long'].
- Compute: Outliers removed, delay_category value_counts.
- **Expected:** ∼18k outliers removed; short: ∼200k, medium: ∼85k, long: ∼42k.

5.3 Task 3: Advanced Grouping and Custom Aggregation

Group by origin and carrier. Aggregate:

- arr_delay: mean.
- air_time: median.
- Proportion long delays: lambda $(x > 60)$.mean().
- distance: 95th percentile (np.percentile).

Use agg dict with multi-functions. Sort by mean arr_delay desc. Show top 5 rows.

- **Expected:** Table with columns like mean_arr_delay, prop_long_delay, median_air_time, p95_distance. Sort by mean_arr_delay descending.

5.4 Task 4: Reshaping and Pivot Analysis

Pivot: rows=month, cols=carrier, values=mean dep_delay (fill 0, margins=True).

- Melt back to long format (add metric='mean_dep_delay' col). Filter to top 3 carriers by flight count.
- Compute correlation between mean_dep_delay and month.
- **Expected:** Melted shape; Corr rounded to 2 decimals.

5.5 Task 5: Window Functions and Ranking

- Rank `carriers` by mean `arr_delay` (asc) within each `month-origin` partition using `rank(method='dense')`.
- Compute cumulative avg `arr_delay` per `carrier` over months (expanding mean).
- Find worst `carrier` (highest mean delay) per `month` using `idxmax`.
- Output: Worst carriers series; top 3 final cum_avgs.
- **Expected:** 12 worst carriers (e.g., month1: 'UA'); Top3 e.g., UA=18.2.

5.6 Task 6: Complex Filtering and Derived Insights

Filter: `distance > 1000` AND `air_time > 90th percentile`.

- Compute score: (`dep_delay` + `arr_delay`)/`distance` × 100 (min delay per 100 miles)—vectorized.
- Group by `dest`: mean score, count. Filter count>1000, sort desc mean score. Top 5 worst dests.
- **Expected:** Top5 includes dest, mean_score, count.