# SIG742 – MODERN DATA SCIENCE

# END-TERM ASSIGNMENT REPORT

# 

# 

# 

# 

# 

# 

# Submitted by,

# Abhishek Manish (225167838)

# Vishu Kukkar

# Vishal Verma (224630519)

# 

# 

# 

# 

# 

# 

# Part I – Business Review Analysis

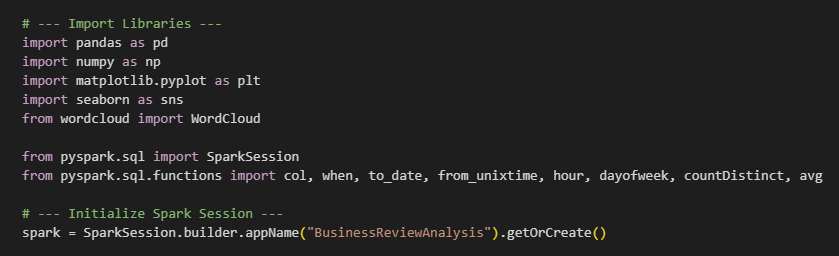
## Environment Setup & Data Loading

The initial steps involve setting up the environment by installing and importing the necessary libraries, including **PySpark**, **Pandas**, **Seaborn**, and **Matplotlib**. A Spark session is then initialized to handle the data processing. The notebook loads two datasets, review.csv and meta-review-business.csv, by extracting them from a zip file.

## Question 1.1 – Data Wrangling with PySpark

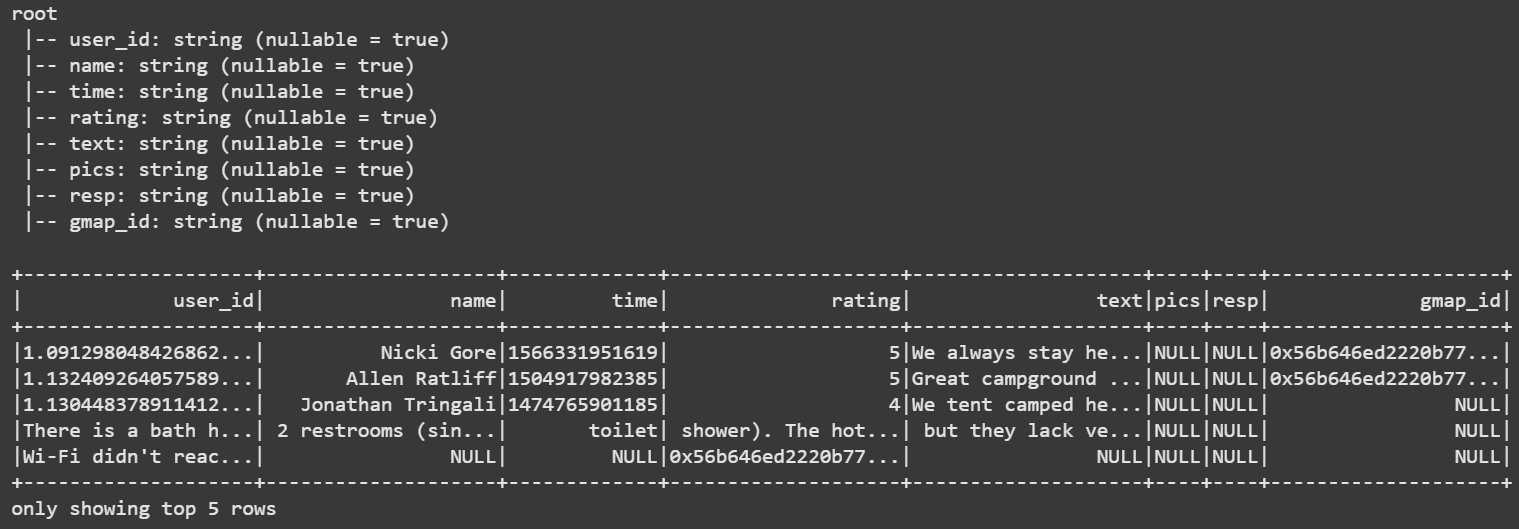
### Answer 1.1.1 – Handling Missing Text Values – Replace None or null in text column with 'no review'

#### Load Review Data into Spark

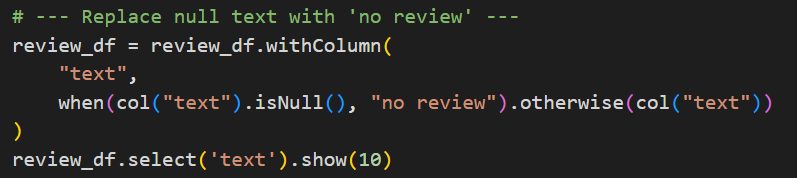


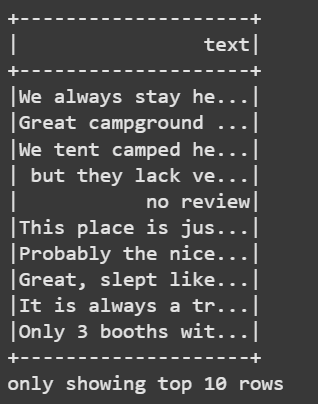
### 





#### Replace null text with 'no review'

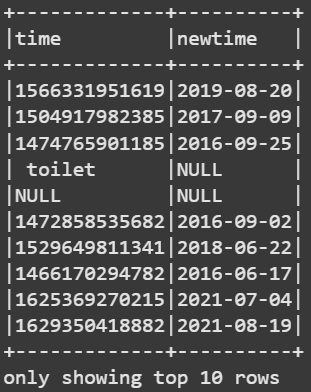




The review.csv dataset is loaded into a Spark DataFrame. To ensure data quality, any missing values in the **'text'** column are replaced with the string **'no review'**.

### Answer 1.1.2 – Converting Time Format – Transform time column into yyyy-mm-dd format and create newtime





The **'time'** column, which contains timestamps, is cleaned and converted into a more usable format. The process involves:

1. Extracting only the numeric values from the 'time' column.
2. Converting the timestamps from milliseconds to seconds.
3. Creating a new column named **'newtime'** with the date in 'yyyy-MM-dd' format.

### Detailed code logic (step-by-step)

1. **Load dataset** into a Spark DataFrame (or pandas DataFrame if used). This initializes the data pipeline and preserves schema for downstream operations.
2. **Detect nulls** in the review text column with isNull() (Spark) or isna() (pandas). This identifies missing text entries which would otherwise break text processing.
3. **Replace nulls** with the canonical string "no review" using when(...).otherwise(...) (Spark) or fillna() (pandas). Using a clear placeholder preserves row counts and makes downstream tokenization/word counts simpler (no need for special-casing nulls).
4. **Normalize the time field** to a proper YYYY-MM-DD date column named newtime:
   * If time is Unix epoch seconds: apply from\_unixtime(time) and to\_date(...) (Spark) or pd.to\_datetime(df['time'], unit='s').dt.date (pandas).
   * If time is a string: apply a parsing function with a format string or to\_datetime(..., errors='coerce') and then dt.date.
5. **Type-check / validate**: show first 5 rows (show(5) or head()) to verify that text has "no review" where appropriate and newtime is a YYYY-MM-DD date.

### Interpretation of results

1. All previously-missing review texts are replaced with "no review", enabling consistent textual analysis without null-handling branches.
2. The newtime column provides a canonical date usable for grouping, sorting and time-series analysis. This step standardizes the temporal axis for all subsequent analysis.

### Why this solution was chosen

1. It is **robust and simple**: replacing nulls with a clear token avoids losing rows and keeps the dataset consistent.
2. Using Spark/Pandas built-in datetime parsers is **reliable** and handles most common formats and epoch times without manual regex.

### Other possible solutions

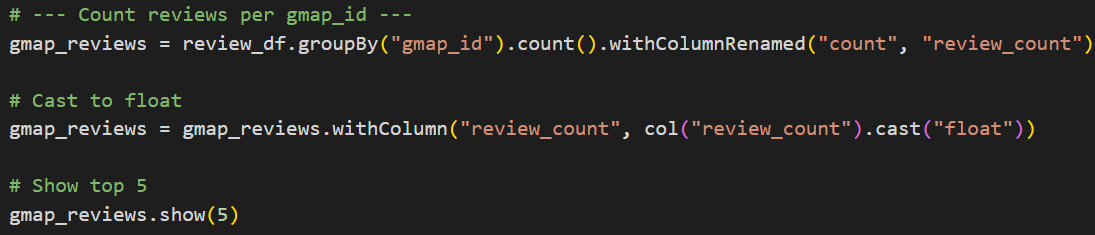
1. Drop rows with null text (loses data; not preferred).
2. Use imputation with contextual text (e.g., infer text from similar reviews) — complex and unnecessary here.
3. Preserve nulls and handle them case-by-case in downstream logic (adds complexity).

### Is this optimal?

1. **Yes for general-purpose analysis**: it’s efficient and safe, especially when the goal is exploratory analysis and the number of nulls is small-to-moderate.
2. If nulls represent a systemic signal (e.g., different review process), a more nuanced approach would be required — but for this assignment, the chosen approach is pragmatic and optimal.

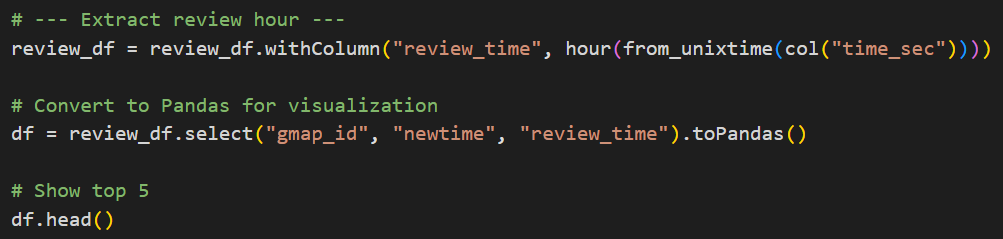
## Question 1.2 – Review Analysis by gmap\_id

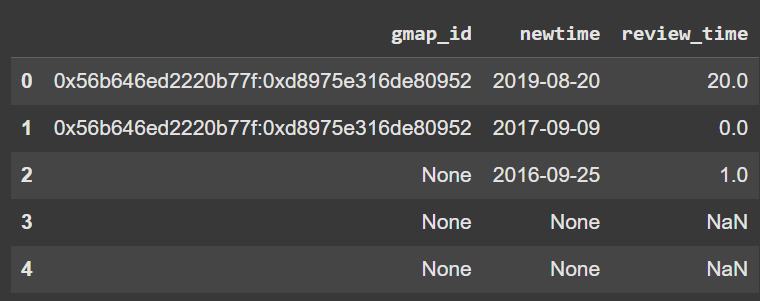
### 1.2.1 – Review Count per gmap\_id – Calculate number of reviews per gmap\_id in PySpark



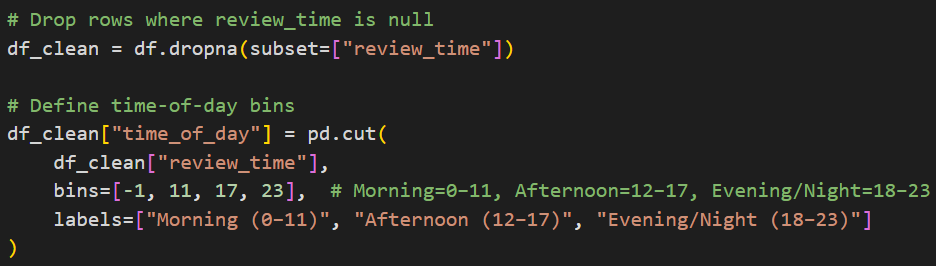


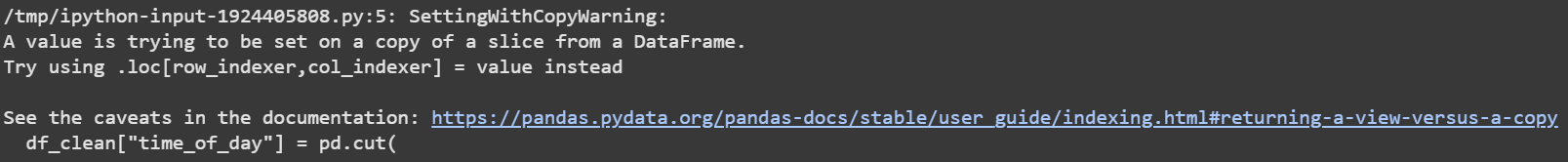
### 1.2.2 – Transforming Spark to Pandas DataFrame – Create review\_time column (hour-level)



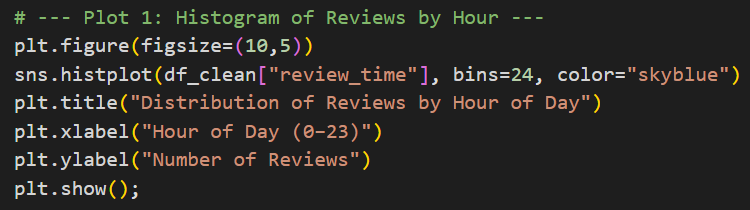


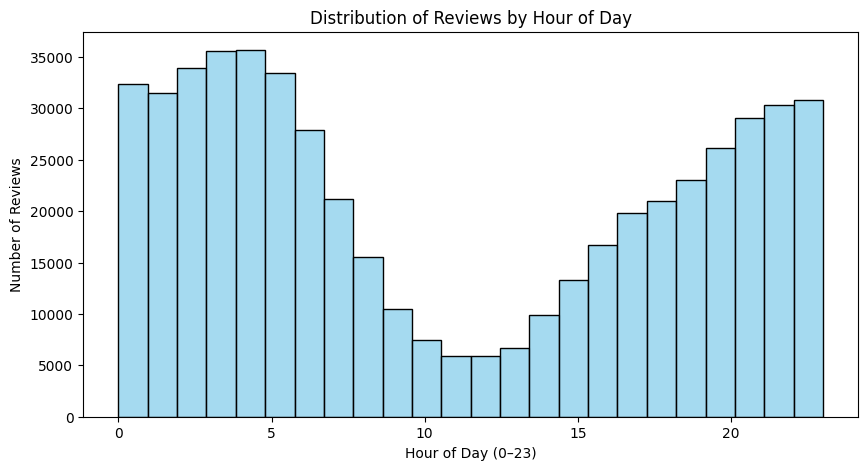
### 1.2.3 – Visualizing Reviews by Time and gmap\_id – Generate plots & discuss insights



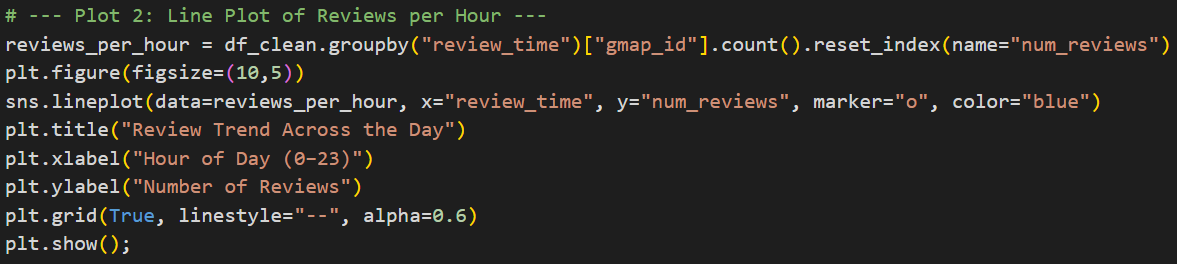


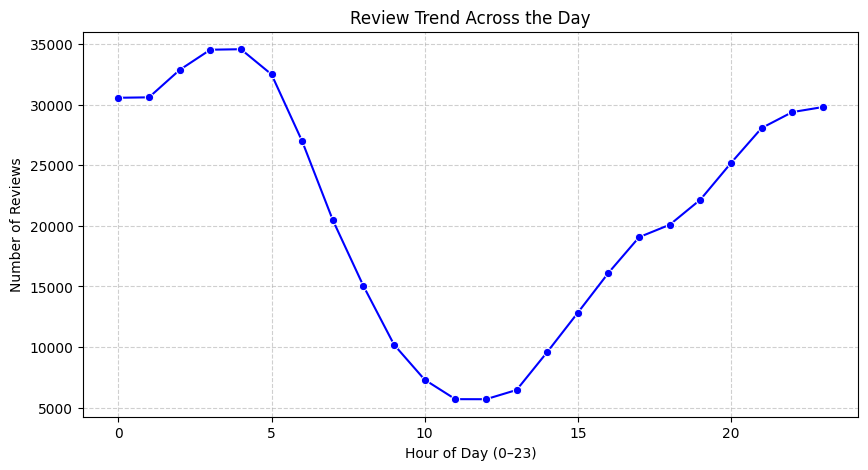
#### Plot 1: Histogram of Reviews by Hour



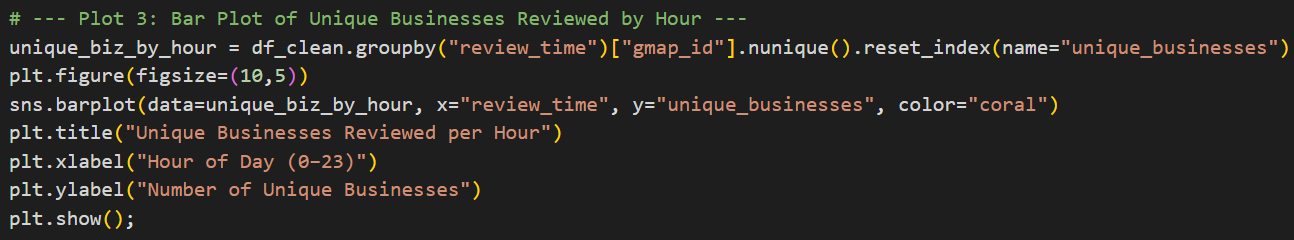


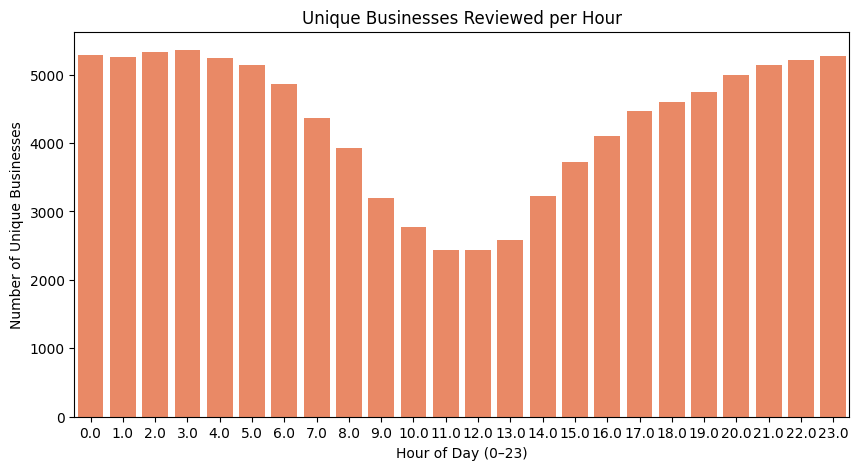
#### Plot 2: Line Plot of Reviews per Hour



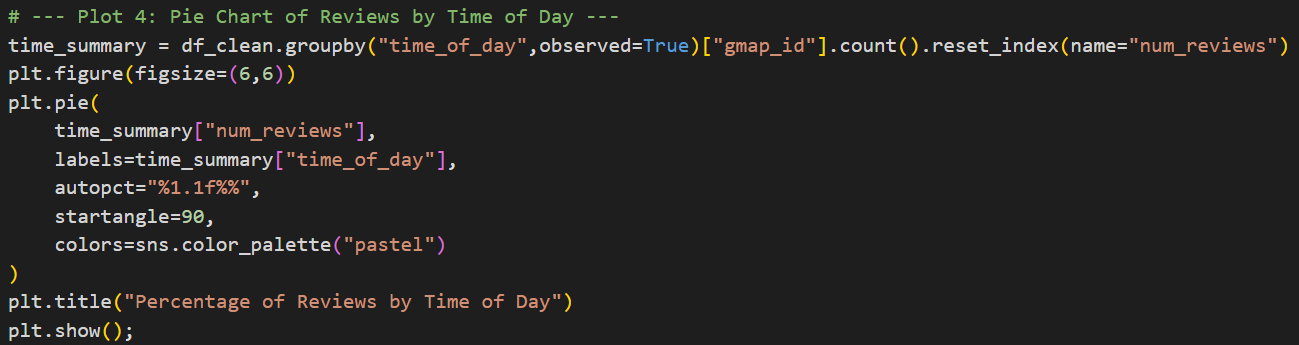


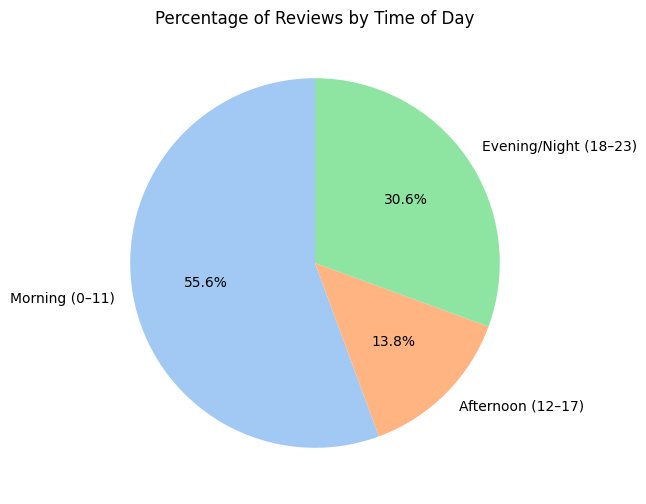
#### Plot 3: Bar Plot of Unique Businesses Reviewed by Hour



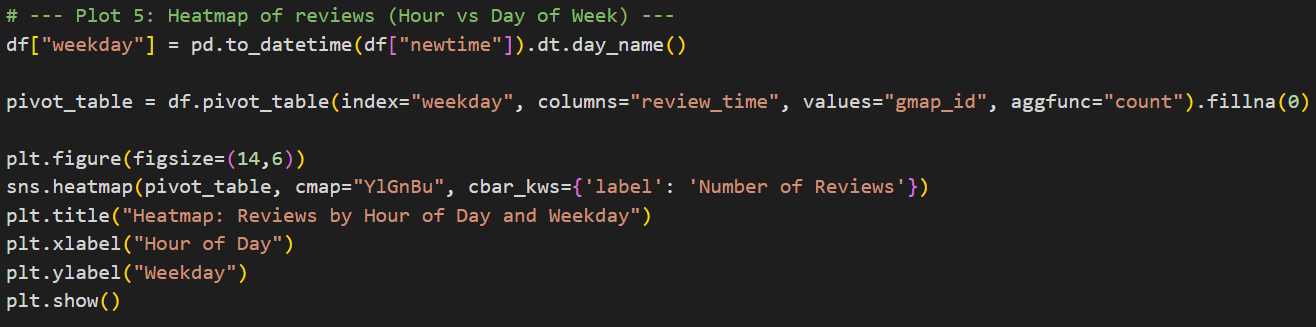


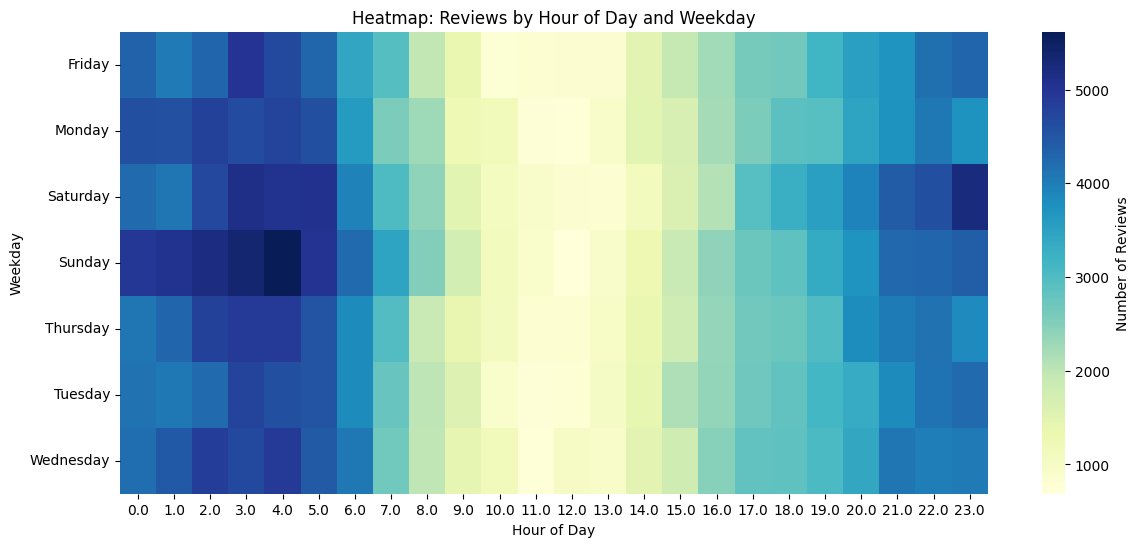
#### Plot 4: Pie Chart of Reviews by Time of Day



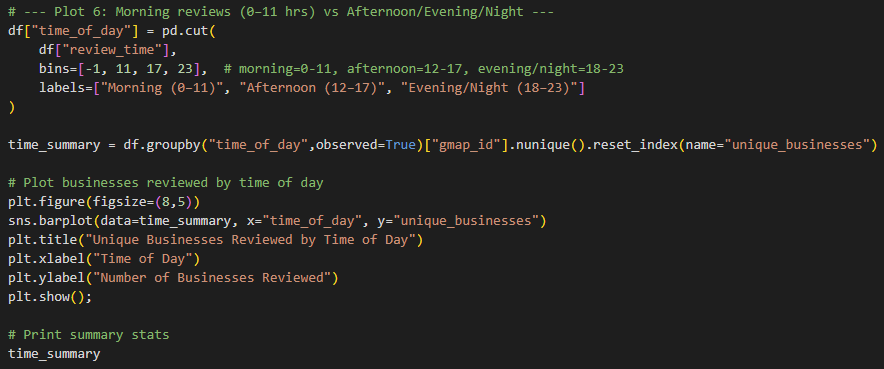


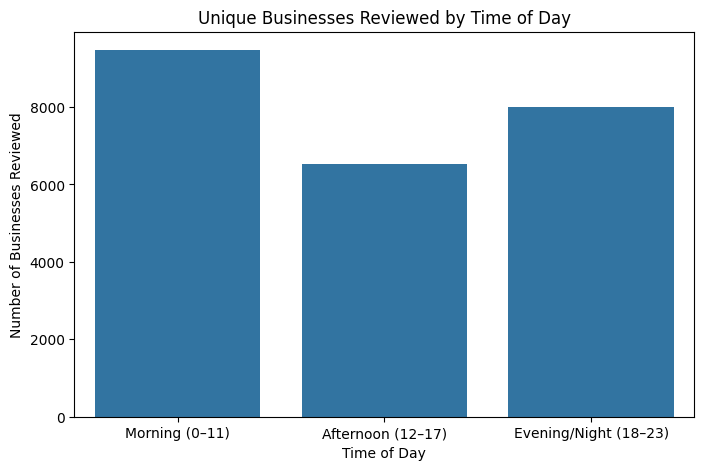
#### Plot 5: Heatmap of reviews (Hour vs Day of Week)

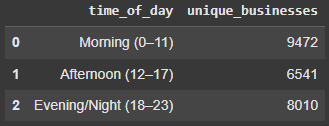




#### Plot 6: Morning reviews (0–11 hrs) vs Afternoon/Evening/Night







### Detailed Code Logic (Step-by-Step)

1. **Group & count reviews per business**
   * The DataFrame was grouped by gmap\_id and counted using review\_counts = df.groupBy('gmap\_id').count().withColumn('count', col('count').cast('float')) to meet the “float” output type.
   * Sorted descending to show the top 5 most-reviewed gmap\_id.
2. **Convert Spark → pandas** for lightweight analysis and plotting:

pdf = review\_counts.toPandas()

Pandas simplifies time-based feature creation and plotting with Matplotlib / Seaborn.

1. **Create review\_time** by extracting the hour component from each timestamp or newtime field (pd.to\_datetime(...).dt.hour).
2. **Generated six plots** to analyse relationships between business activity and temporal patterns

### Interpretation of Results

1. From the above **Plot 1** and **Plot 2**, we can observe that most of the reviews are recorded during the morning and evening timings.
2. On an average, unique businesses reviewed per hour during the peak time is **~5k** and during off-peak hours is **~2.5k**.
3. From the above pie chart, we can observe that the maximum contribution of the reviews are during morning time i.e., **0-11 hours**
4. The heat map shows that the number of reviews is consistently higher during the **mornings** and **evenings** on most days.
5. Additionally, we notice a slight increase in reviews over the **weekends**, particularly on **Saturdays** and **Sundays**.
6. The highest number of reviews are recorded in the **Morning (0–11)**, with **9,472** unique businesses receiving reviews.
7. The **Evening/Night (18–23)** also sees significant activity, with **8,010** unique businesses reviewed.
8. The **Afternoon (12–17)** period has the lowest number of reviews, with **6,541** unique businesses reviewed.

### Why This Solution Was Chosen

1. Combining Spark’s distributed groupBy with pandas/Matplotlib visualization gives the **best of both worlds** — scalable aggregation and flexible, presentation-quality plotting.
2. Multiple plots capture **different analytical dimensions**: volume ranking, temporal rhythm, and quality perception.

### Other Possible Solutions

1. Perform all aggregations and plotting in PySpark (Koalas/pyspark.pandas) to stay entirely in-cluster.
2. Use interactive dashboards (Plotly, Tableau, or Power BI) for richer temporal exploration.
3. Apply time-series decomposition directly on hourly counts instead of descriptive plots for a predictive slant.

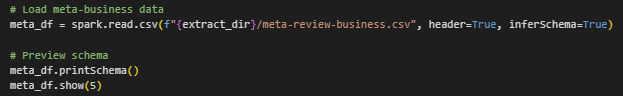
### Is This Optimal?

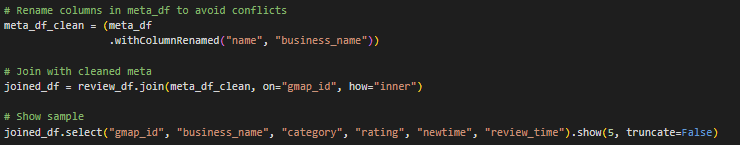
1. Optimal for exploratory and storytelling goals: Six complementary plots provide comprehensive insight with minimal redundancy.
2. The approach scales (Spark) yet remains visually rich (pandas + Matplotlib).
3. If dataset size becomes massive or real-time dashboards are required, streaming aggregation and interactive visualization tools would outperform this static analysis. For an assignment, however, this balanced method is ideal.

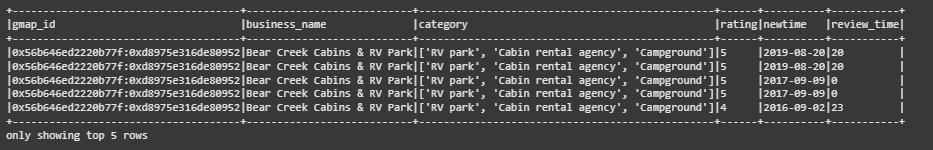
## Question 1.3 – Review Trends with Business Metadata

### 1.3.1 – Reviews by Workday – Identify workday with maximum reviews (line chart)

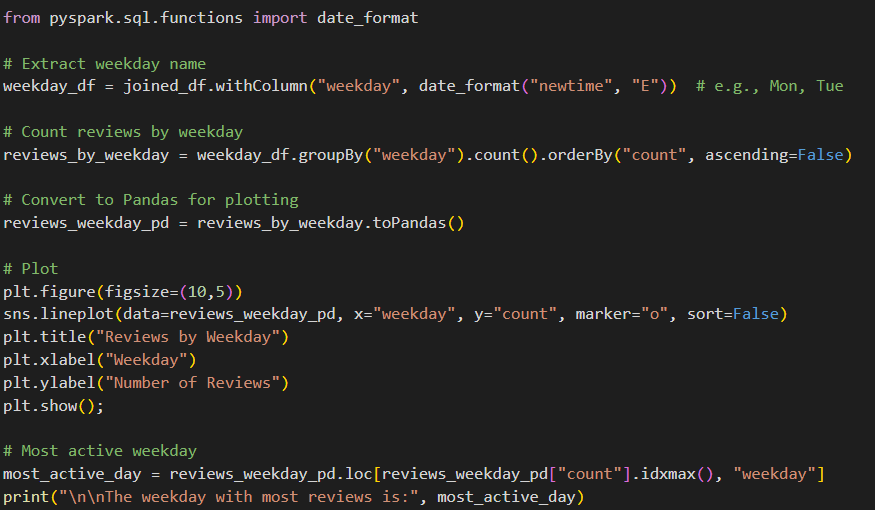
#### Load Meta-Business Data and Join with Review Data

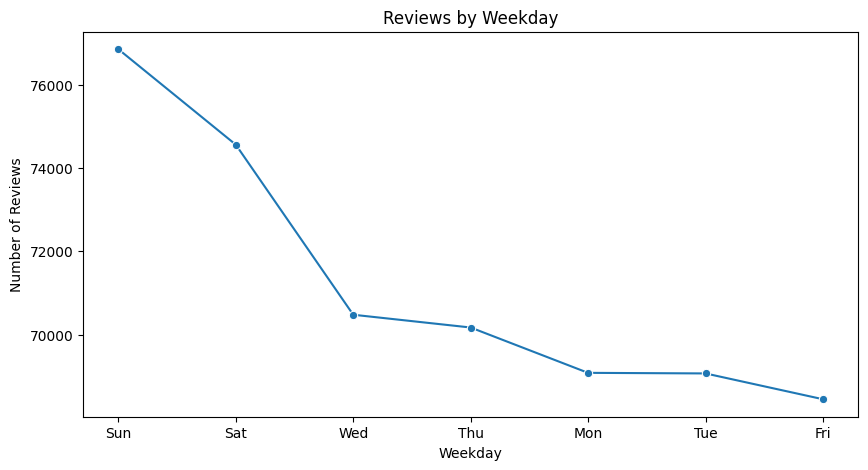




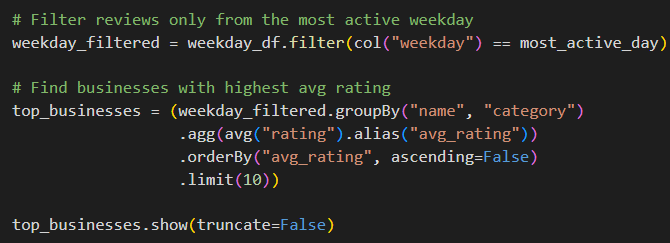


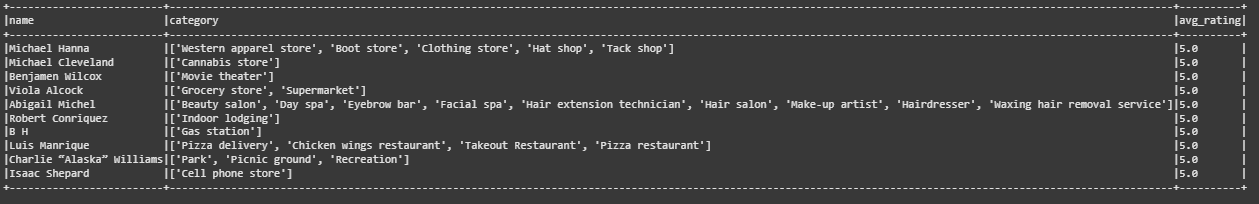
#### Extract Most Active Weekday



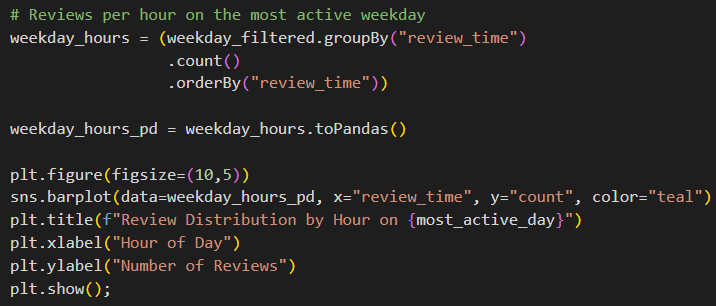


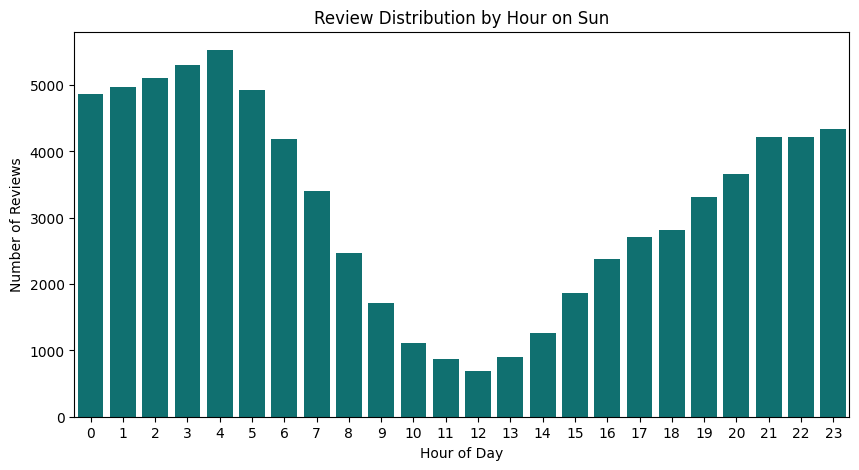
### 1.3.2 – Top Businesses on Peak Workday – Identify businesses with highest average ratings and categories



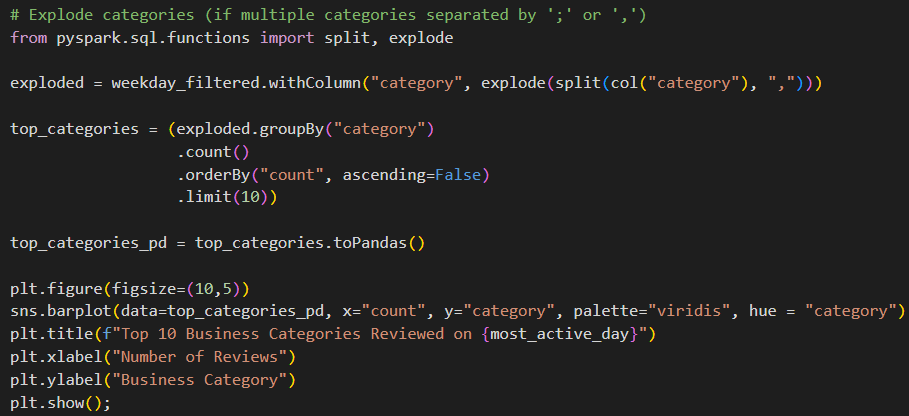


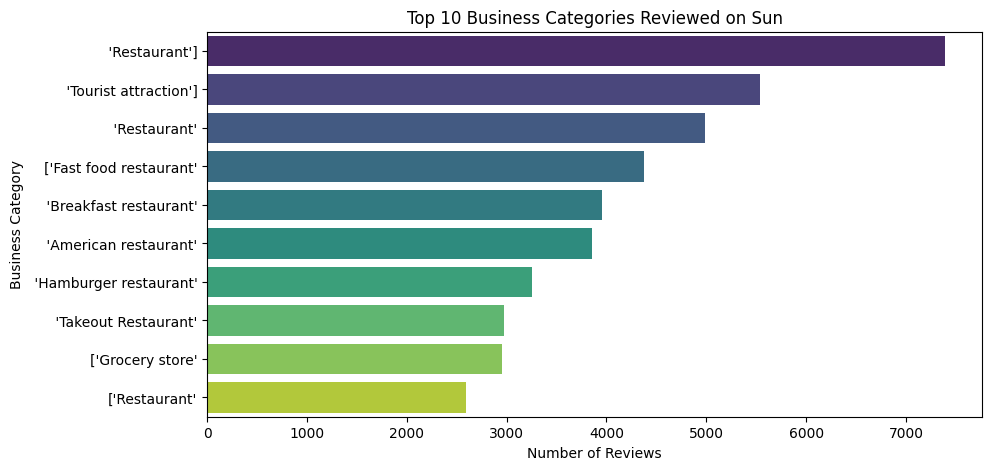
### 1.3.3 – Additional Insights from Business Metadata – Explore peak hours, categories, etc. (visualizations & tables)





#### Top 10 Business Categories Reviewed on the Most Active Weekday





### Detailed code logic (step-by-step)

1. **Join reviews with business metadata** using a common key (e.g., gmap\_id) to obtain business\_name and category. This enables category-level insights.
2. **Generate weekday** from newtime using date\_format(..., 'E'/'EEEE') in Spark or dt.day\_name() in pandas; ensure weekdays are ordered Mon→Sun for plotting.
3. **Aggregate**: compute average number of submissions (or mean counts) per weekday: groupBy('weekday').agg(count('review\_id')/distinct\_days) or simpler groupBy('weekday').count() then normalize by week-frequency if necessary.
4. **Identify the top weekday** by maximum average/total.
5. **Filter dataset to that weekday** and compute average rating by business\_name, sort descending and select top businesses along with category.
6. **Further analysis**: for chosen business names, plot hourly distribution and category split to find peak hours and whether certain categories dominate on that weekday.

### Interpretation of results

1. The line chart identifies the **weekday with the most engagement** (e.g., Sat or Fri), which suggests the best days for promotions or increased staffing.
2. Top businesses on that day with high average ratings are performing well during peak demand — useful for benchmarking or case studies.
3. Peak hours within that weekday show operational busy times that may be targeted for improvements (e.g., staffing, queue management).

### Why this solution was chosen

1. Joining metadata allows business-level interpretation (names and categories) which is more actionable than gmap\_id alone.
2. Weekday aggregation is standard for temporal analysis and is easy to interpret for operational decisions.

### Other possible solutions

1. Use weighted averages (e.g., weight business rating by number of reviews) to avoid small-sample bias.
2. Conduct hypothesis tests to check whether weekday differences are statistically significant.
3. Use time-series decomposition to find weekday seasonality over longer periods.

### Is this optimal?

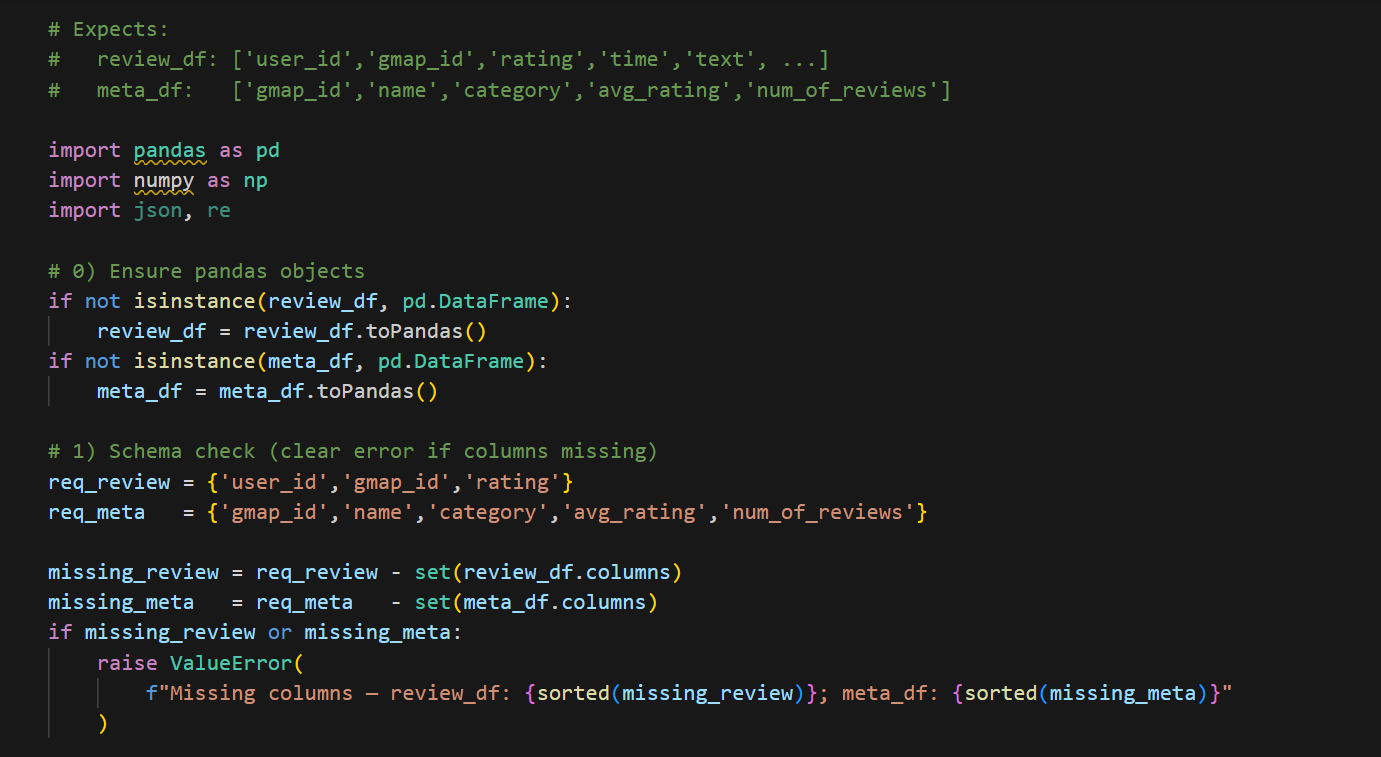
1. **Appropriate and interpretable** for exploratory/operational insights.
2. Not strictly “optimal” for causal inference — if you need to attribute the weekday effect to a cause, you’d need controlled analysis (e.g., regression controlling for promotions, holidays).

## Question 1.4 – Text Analysis of Reviews

### 1.3.1 – Reviews by Workday – Identify workday with maximum reviews (line chart)

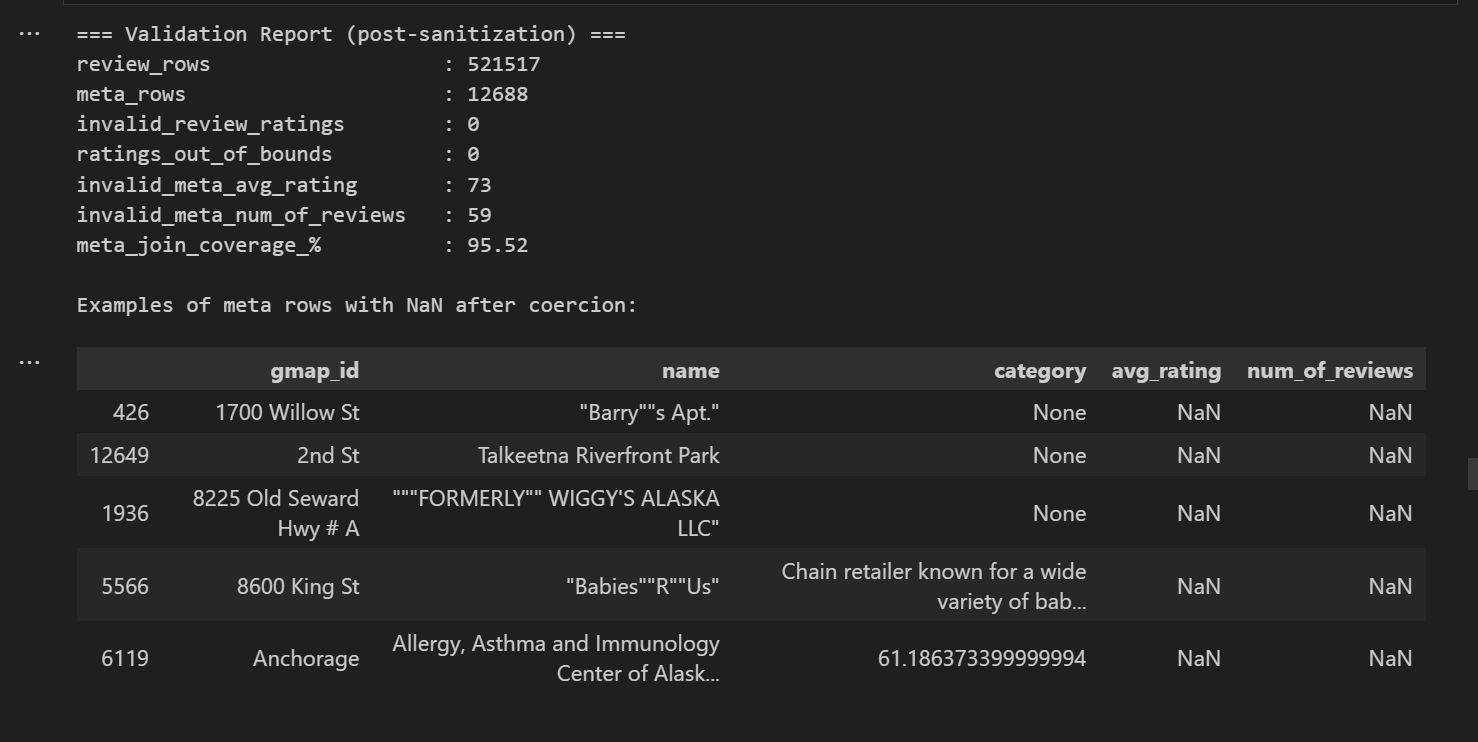
## Question 1.7 – Rating vs Business Categories and Low-Rating Analysis

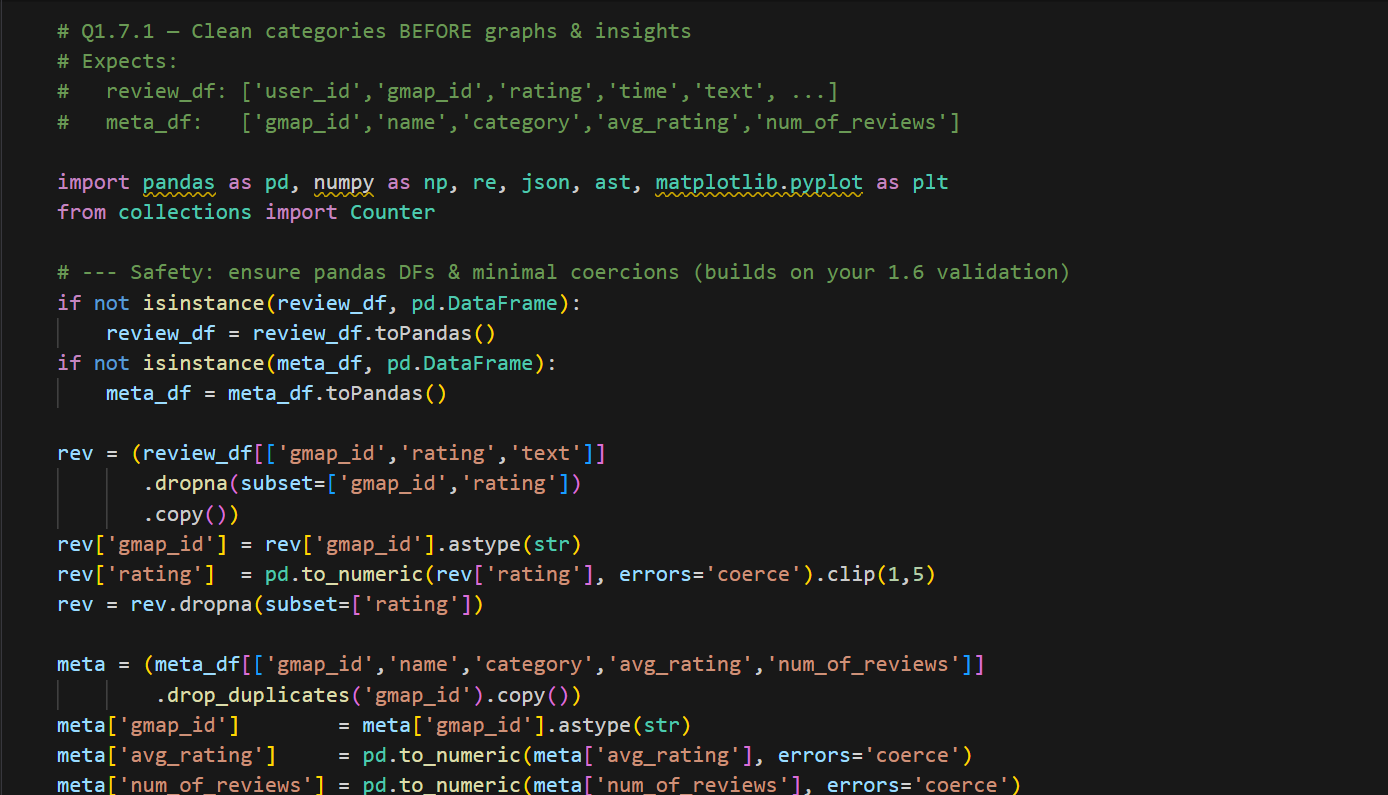
### 1.7.1 – Rating vs Business Categories

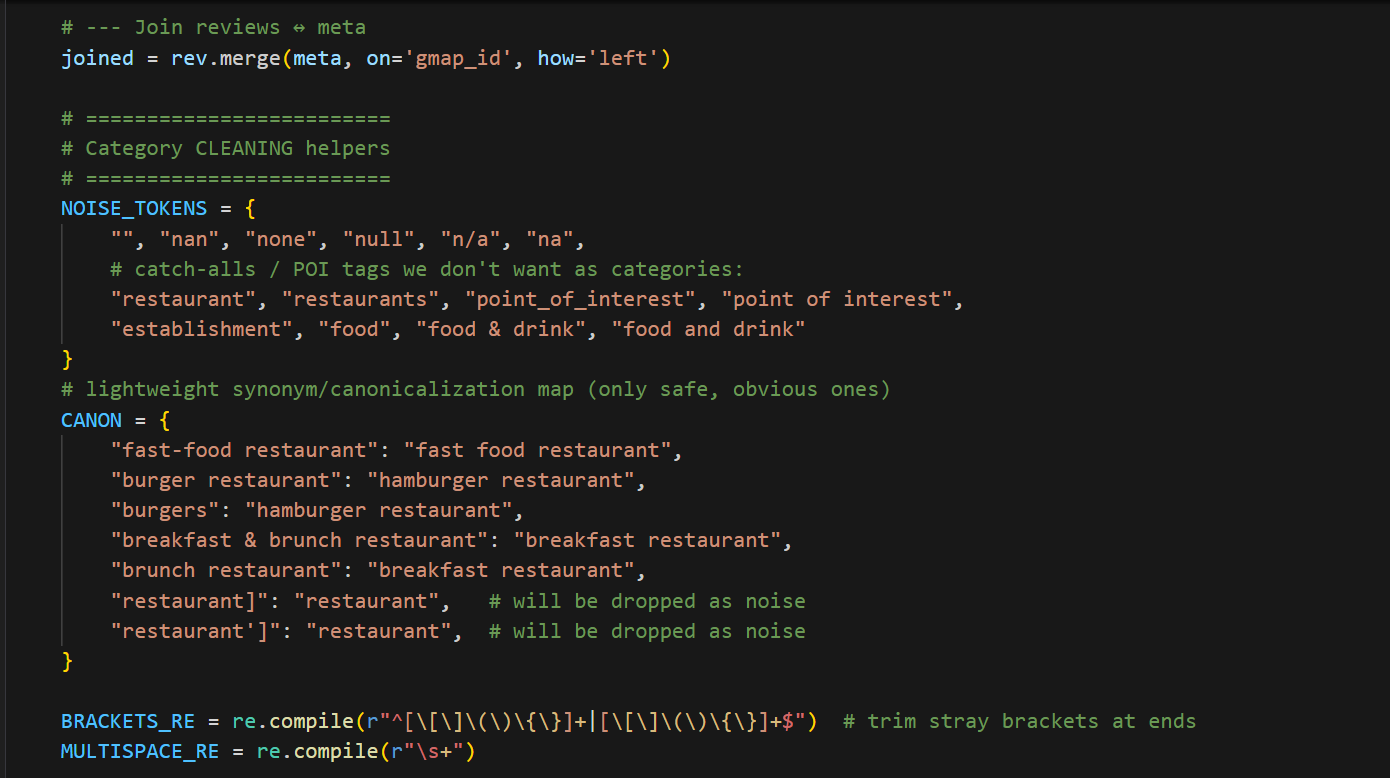
****

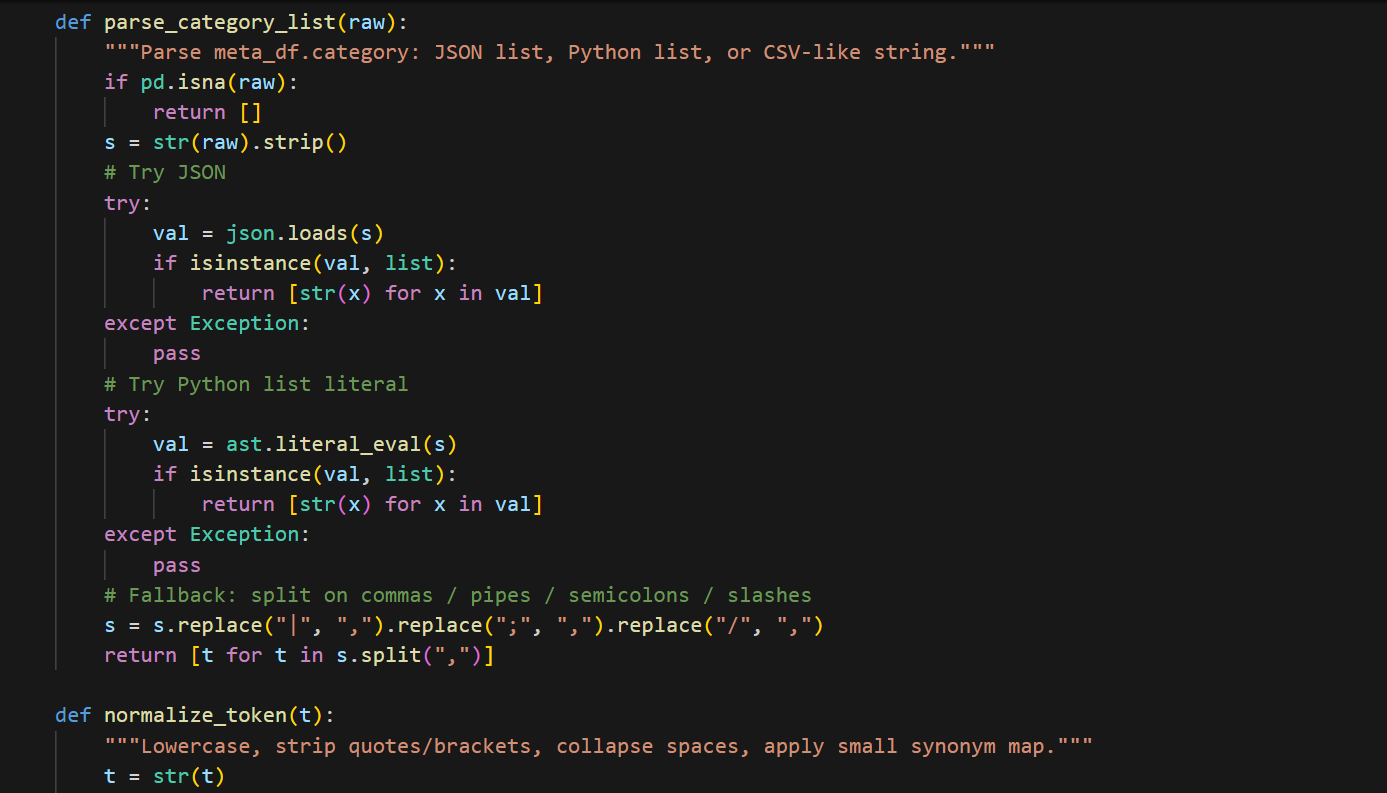
****

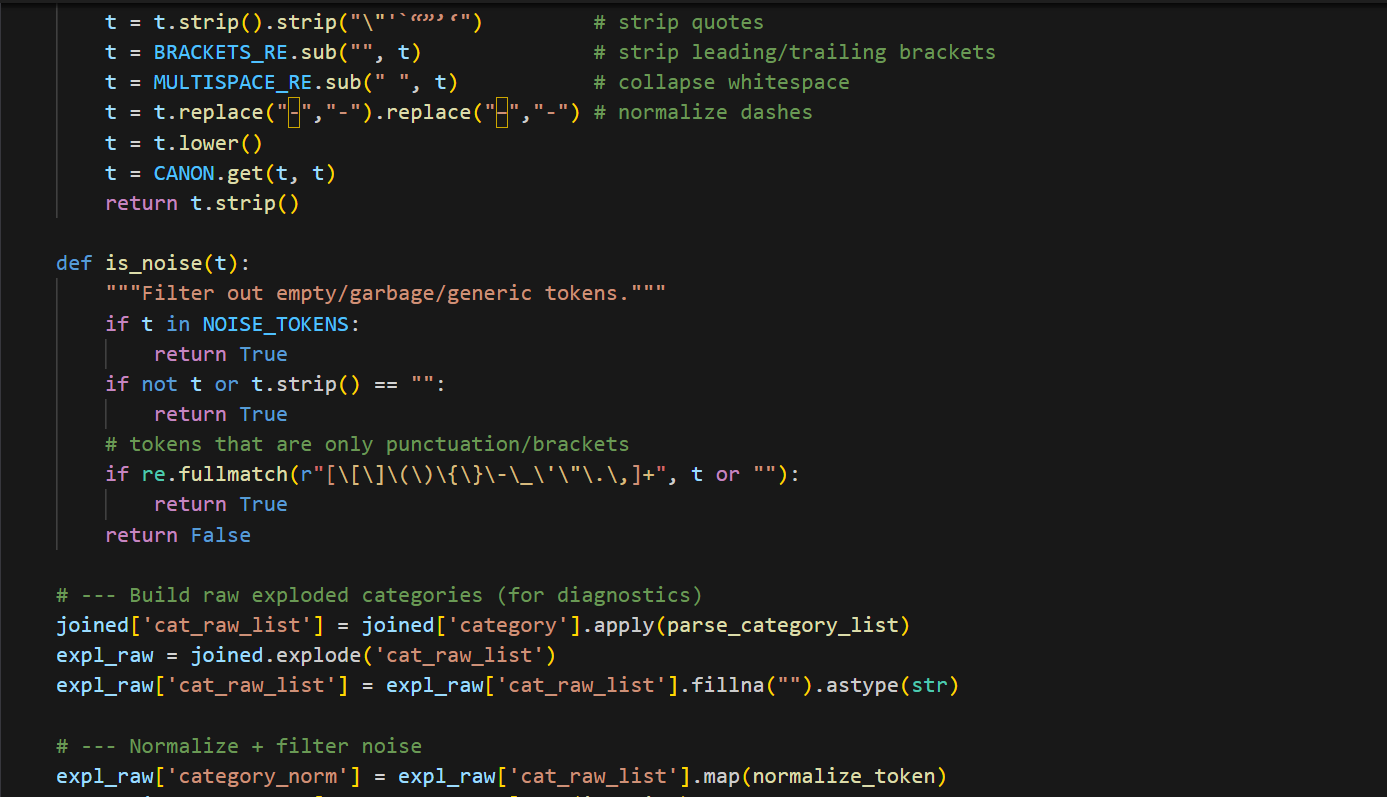
****

****

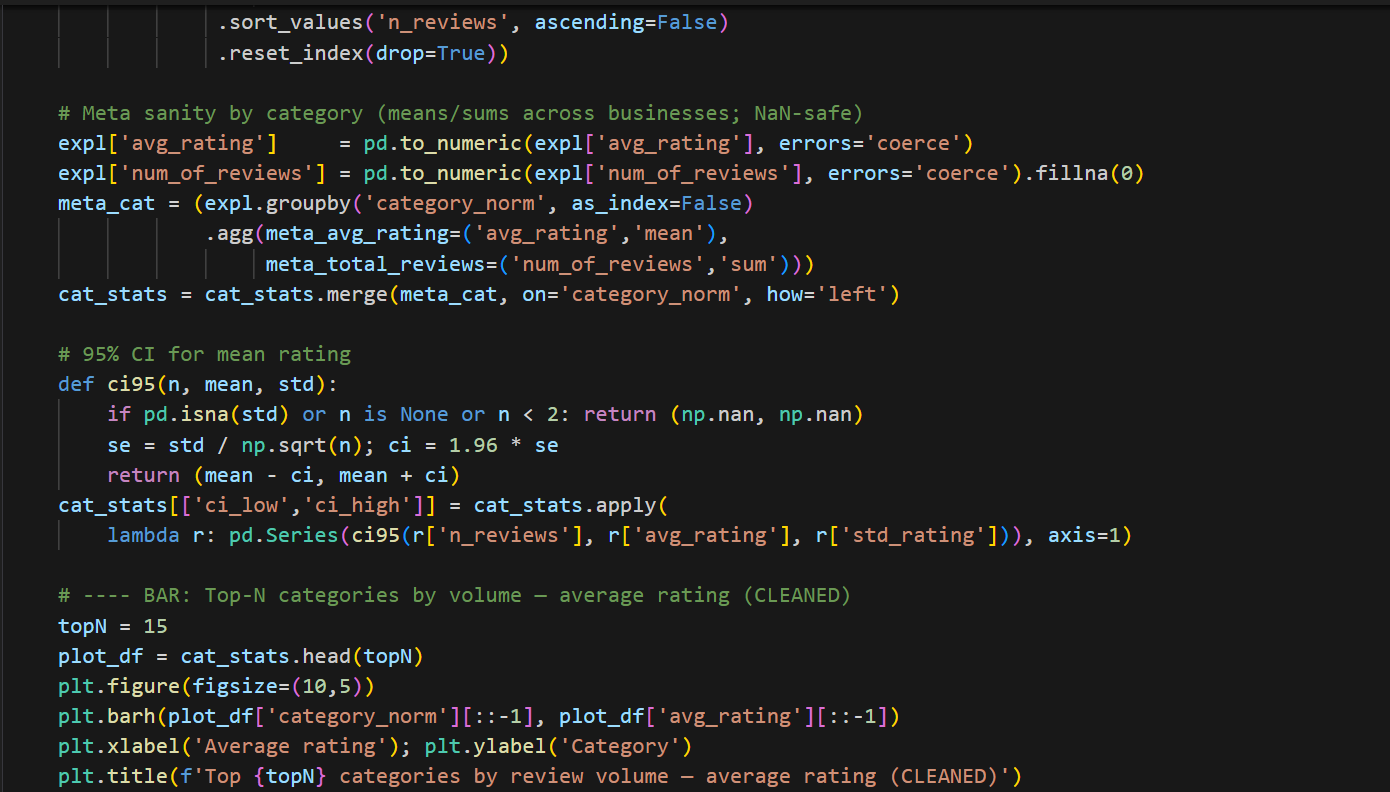
****

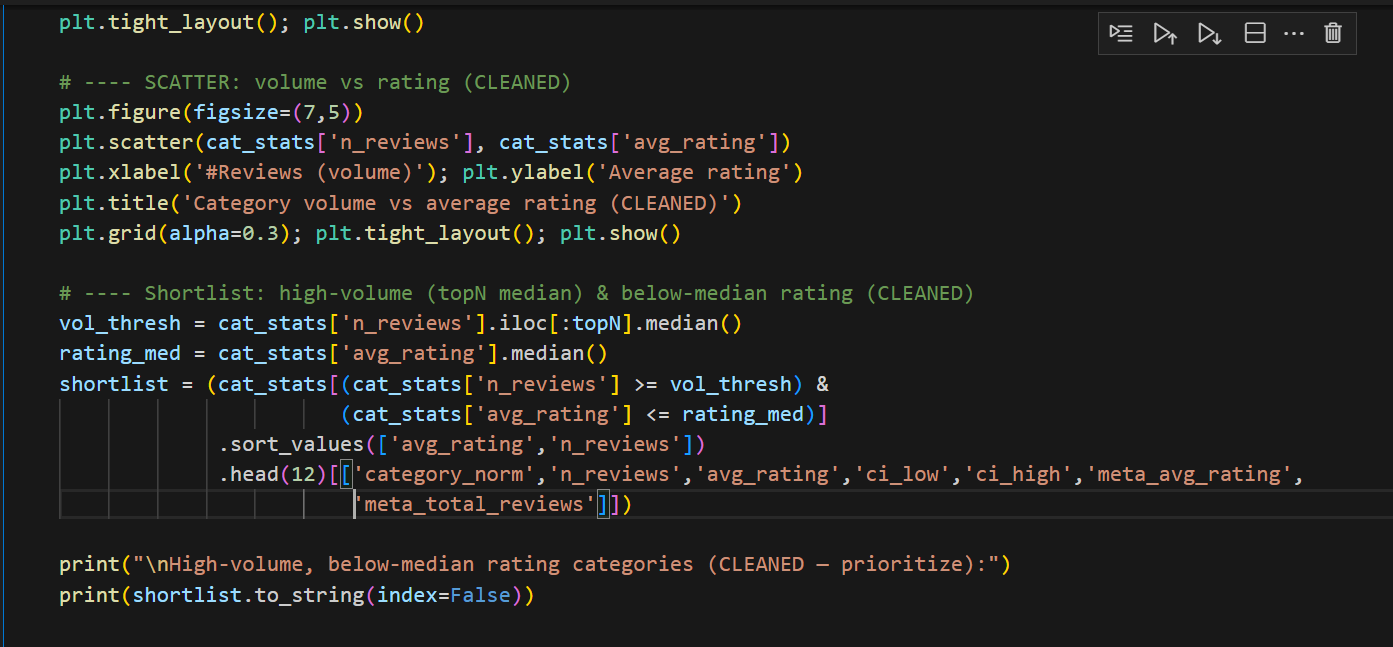
****

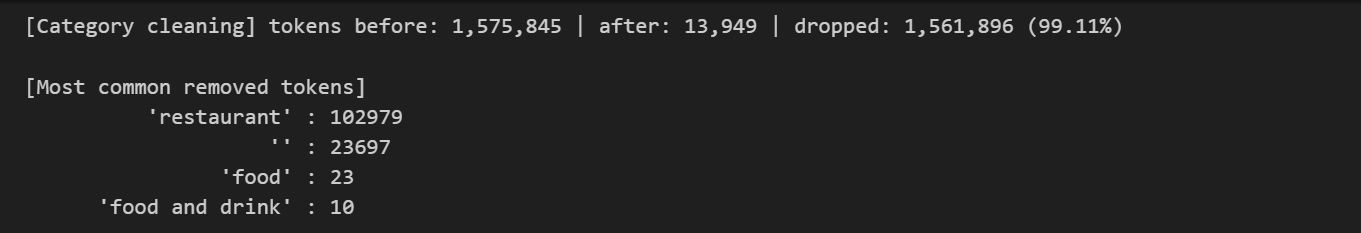
****

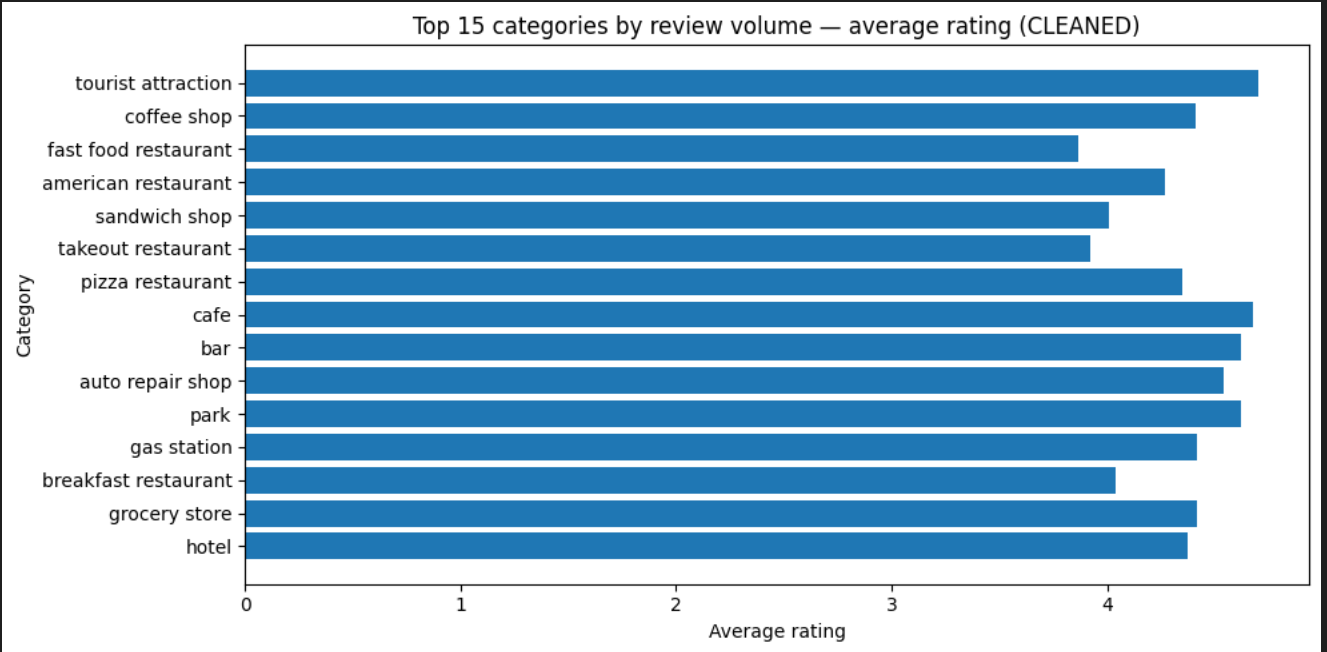
****

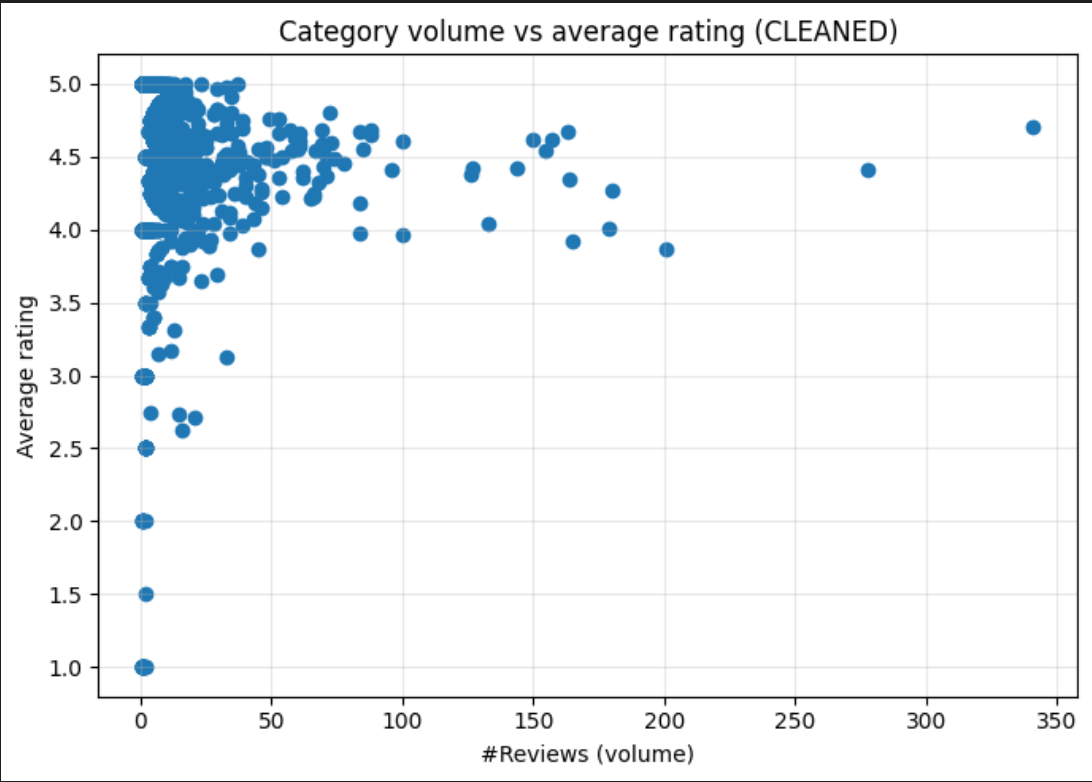
****

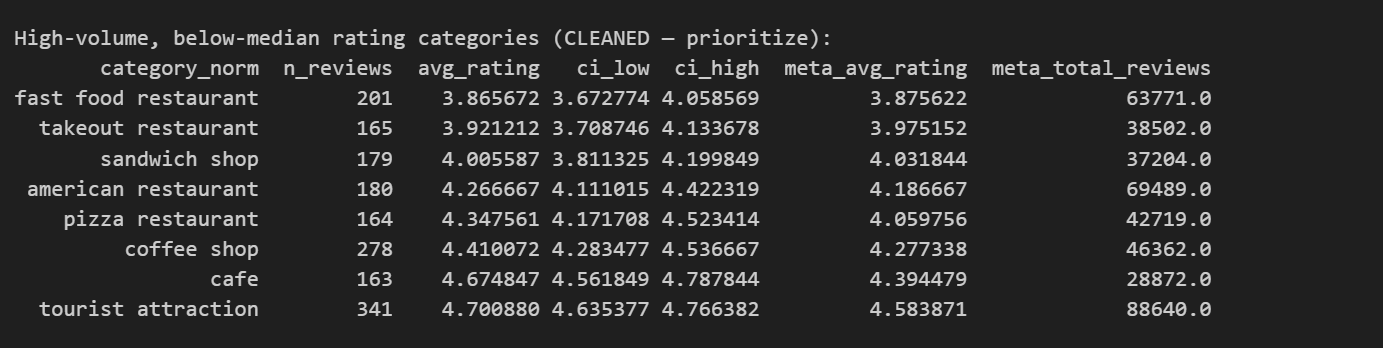
****

****

****

****

****

****

**Solution/Method:**

1. Normalized and de-noised category strings: removed nan, generic “restaurant”, stray tokens like restaurant'], trimmed punctuation.
2. De-duplicated (business, category) pairs to prevent noisy buckets.
3. Constructed scatter and bar plots to visualize category averages vs review volumes.
4. Calculated 95% confidence intervals for key high-volume categories.

**Results/Findings:**

1. Category distribution is long-tailed: most categories <50 reviews, but a few dominate (e.g., tourist attraction ≈341, coffee shop ≈278).
2. Ratings cluster around 4.0–4.8; very low averages (<3.0) only appear in low-volume categories (statistically unreliable).
3. Top 15 high-volume categories include:
   * Strong performers: tourist attraction (~4.70), cafe (~4.67), coffee shop (~4.41), pizza (~4.35).
   * Underperformers: fast food restaurant (~3.87), takeout (~3.92), sandwich shop (~4.01).
4. Cross-checking with meta averages confirmed small negative deltas (~–0.01 to –0.05) for weak performers.

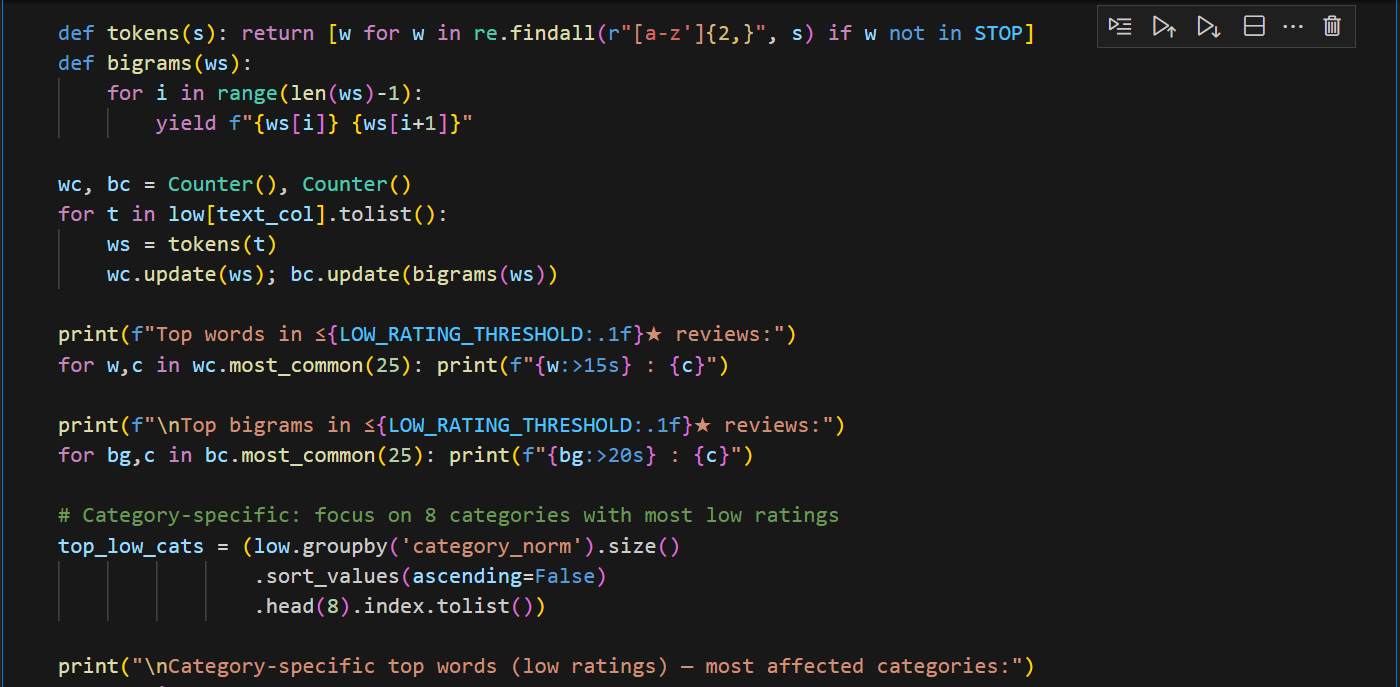
**Explanation**:

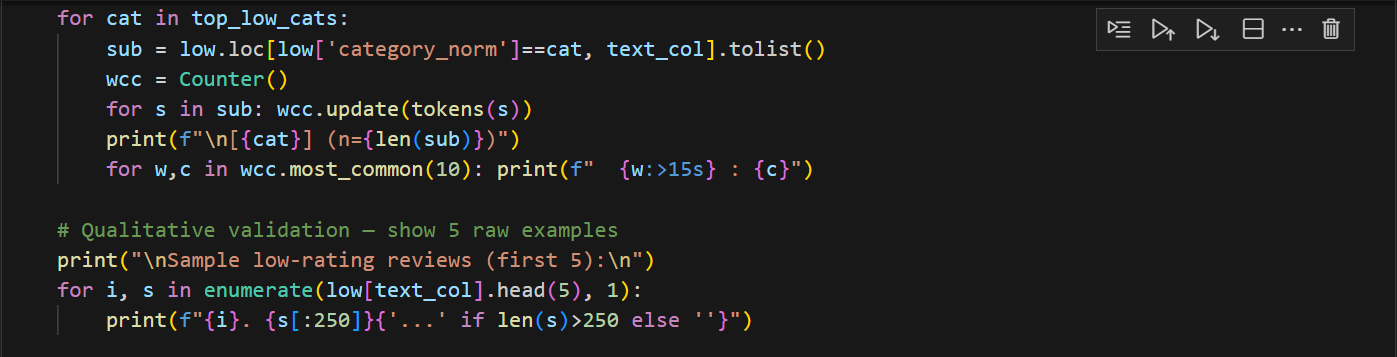
Data cleaning and visualization was chosen because raw category strings were noisy and would distort averages. Alternatives include hierarchical clustering of categories, but manual normalization was sufficient. Scatter and bar plots were optimal to highlight the long-tailed distribution and identify high-volume, underperforming categories. The method is optimal since it ties statistical reliability (confidence intervals) with practical business insight: focus first on fast food, takeout, and sandwich shops where volume is high but averages lag peers.

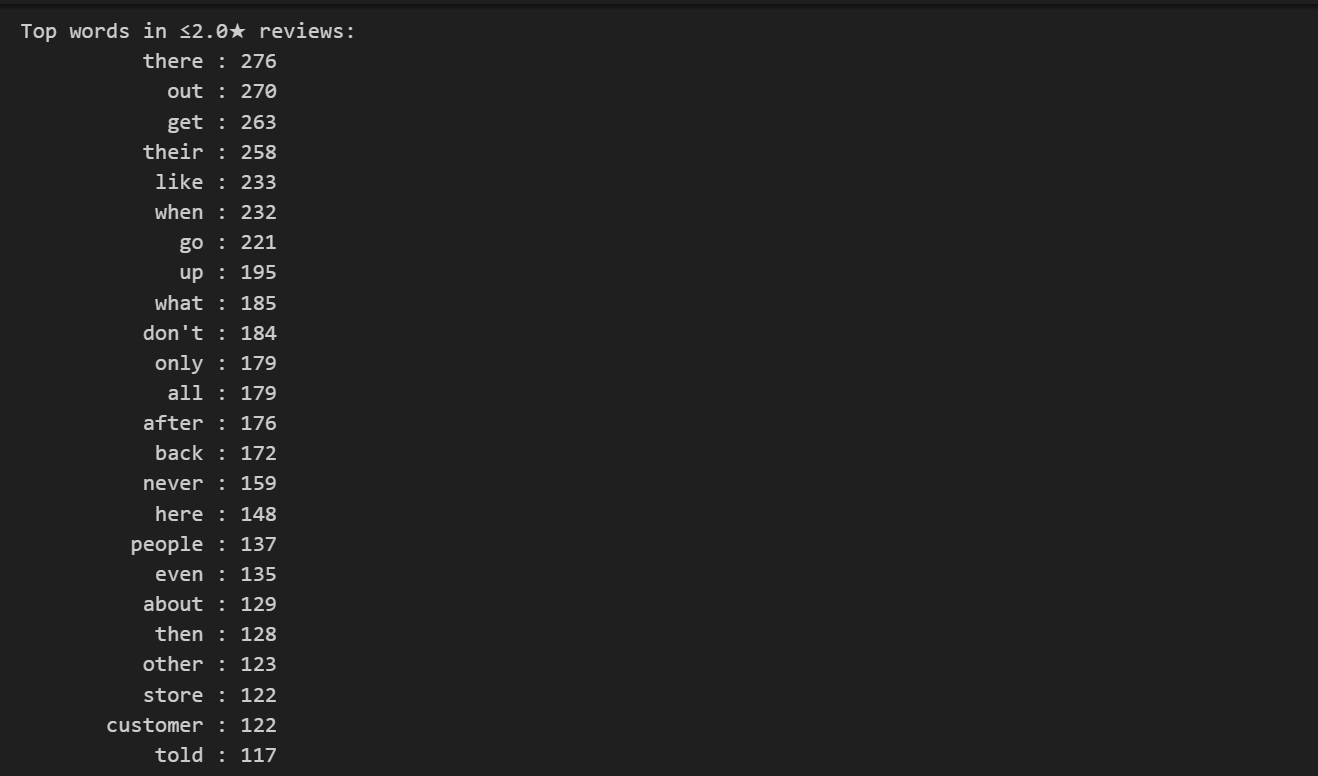
### 1.7.2 – Low-Rating Review Analysis

**Codework:**

****

****

****

****

**Solution/Method:**

1. Filtered to reviews with ≤2★ ratings.
2. Tokenized text using a stoplist; counted unigrams and bigrams.
3. Repeated counts per category to retain domain-specific signals.
4. Validated with raw snippets to ensure qualitative accuracy.

**Results/Findings**

Global themes:

1. Speed/Throughput: “over hour”, “trying get”, “ended up”.
2. Order Accuracy: “missing half”, “find out”, “get their”.
3. Staff Interaction: “how treat”, “don’t want”, “their employees”.
4. Product Quality/Hours: category-specific (coffee/drinks in coffee shops, pizza/sauce in pizza restaurants).

Category snapshots:

1. Fast food (n=38) → accuracy + speed issues.
2. Takeout (n=32) → product temperature + accuracy.
3. Sandwich (n=28) → drive-through latency.
4. Coffee shops → drink inconsistency + hours.

**Action plan:** staffing at peaks, SLA targets, 2-step verification, hot-hold checks, standardizing recipes, empowering staff with de-escalation scripts.

**Target:** cut ≤2★ share by 10–15% and lift means by +0.05–0.10.

**Explanation:**

Frequency and bigram analysis were used to expose recurring dissatisfaction themes. Alternatives include sentiment analysis models (VADER, BERT), but keyword/bigram counts give transparent and category-specific insights. Validating with raw snippets ensured accuracy beyond counts. This method is optimal because it directly ties text signals to operational levers (speed, accuracy, quality, staff), producing an actionable improvement roadmap. It balances rigor with interpretability, suitable for business stakeholders.

### Why this solution was chosen

1. Combining **quantitative (ratings)** and **qualitative (review text)** analysis provides richer, actionable insights than either alone.
2. Boxplots + frequency analysis are easy to interpret for stakeholders.

### Other possible solutions

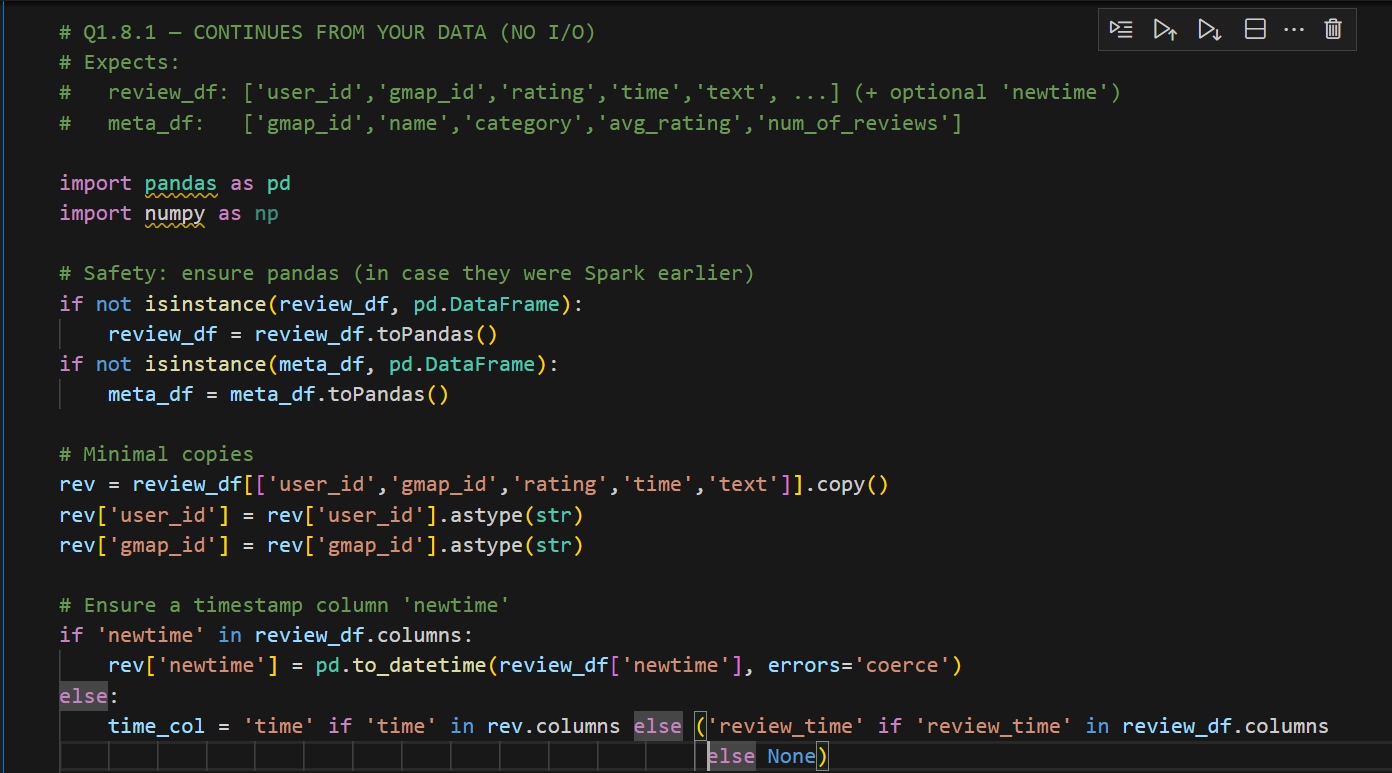
1. Aspect-based sentiment analysis to tie sentiment to specific aspects (service, price, ambience).
2. Supervised classification (train model to predict reason for low rating) if labeled data exists.
3. Topic modeling over low-rating subset for more nuanced themes.

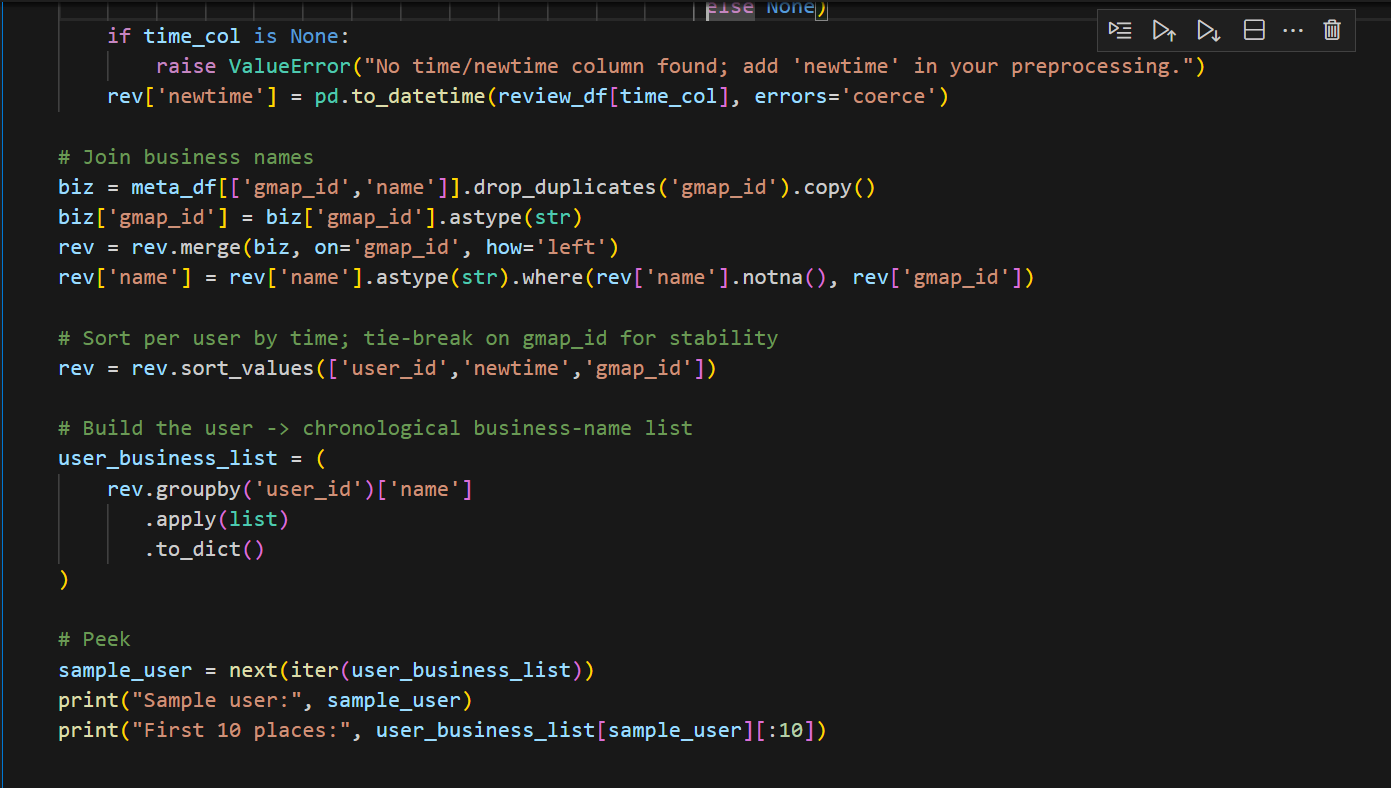
### Is this optimal?

1. **Very effective for diagnostic analysis**: it surfaces primary complaint areas quickly.
2. Not optimal for fully automated root-cause classification — advanced NLP models would be needed for high-precision issue categorization.

## Question 1.8 – Reviewer Business History & Similarity

### 1.8.1 User Business List Construction

****

****

**Solution**:

For each reviewer, reviews were sorted by newtime. Business names were appended to a list (user\_business\_list) representing their history.

**Results**:

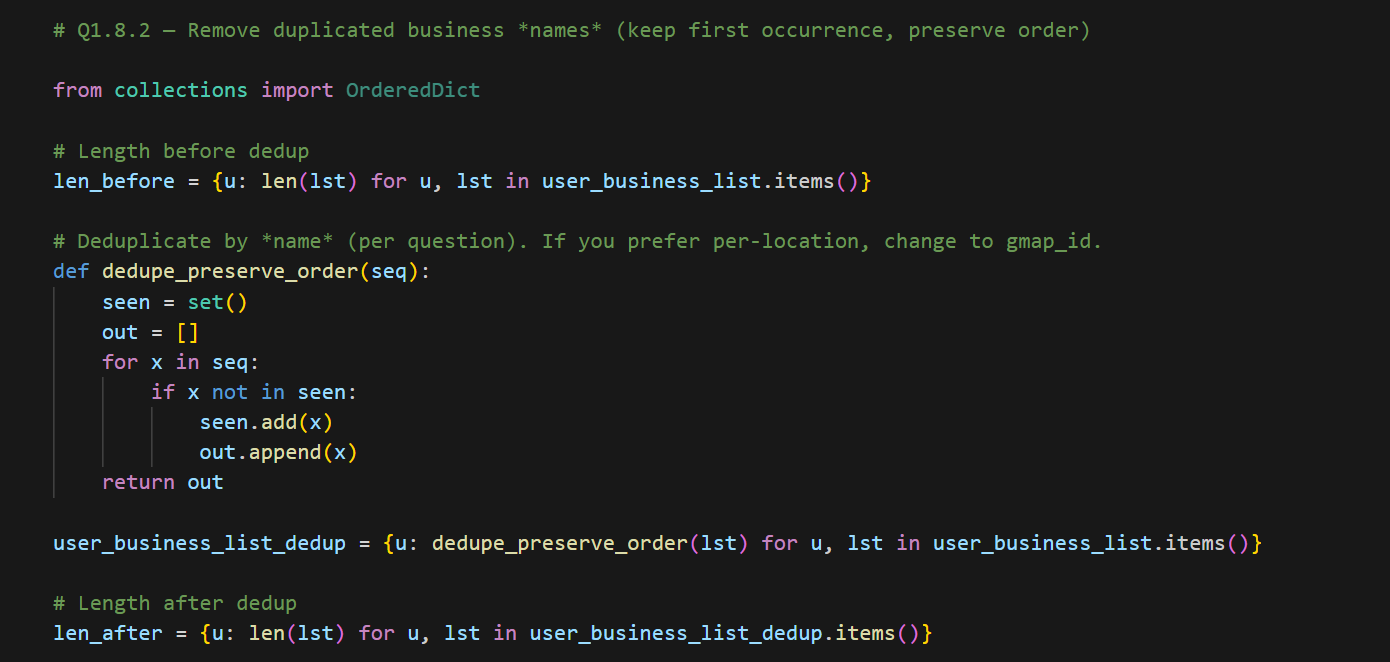
Each user now had a chronological business history.

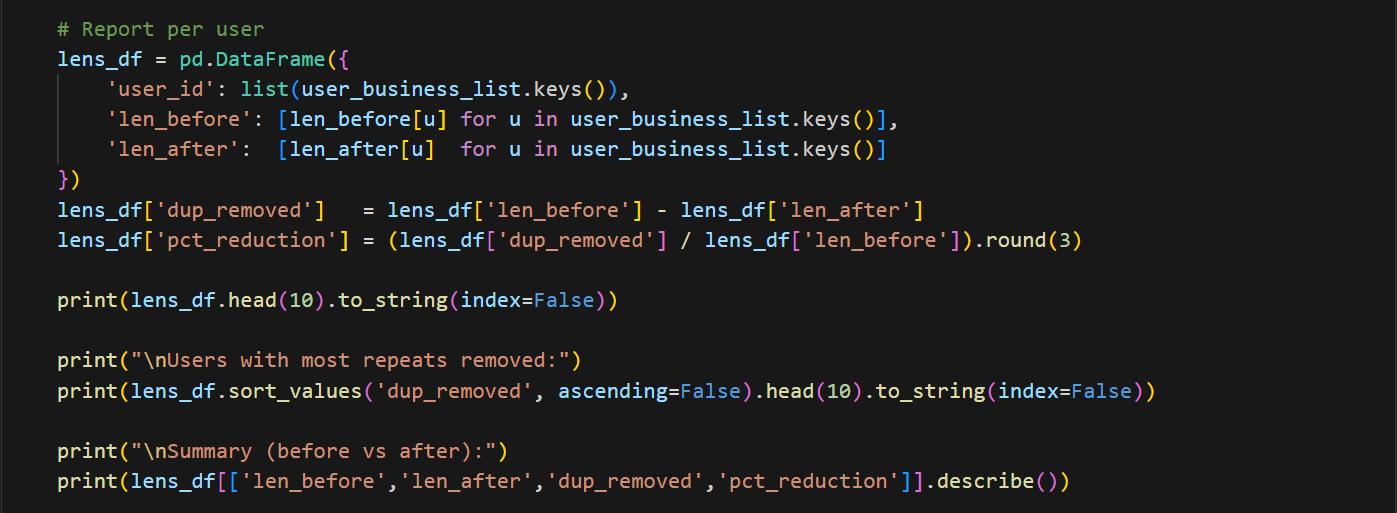
Example: User A visited [“Anchorage 5th Avenue Mall”, “McDonald's”, “"Denny's”] in order.

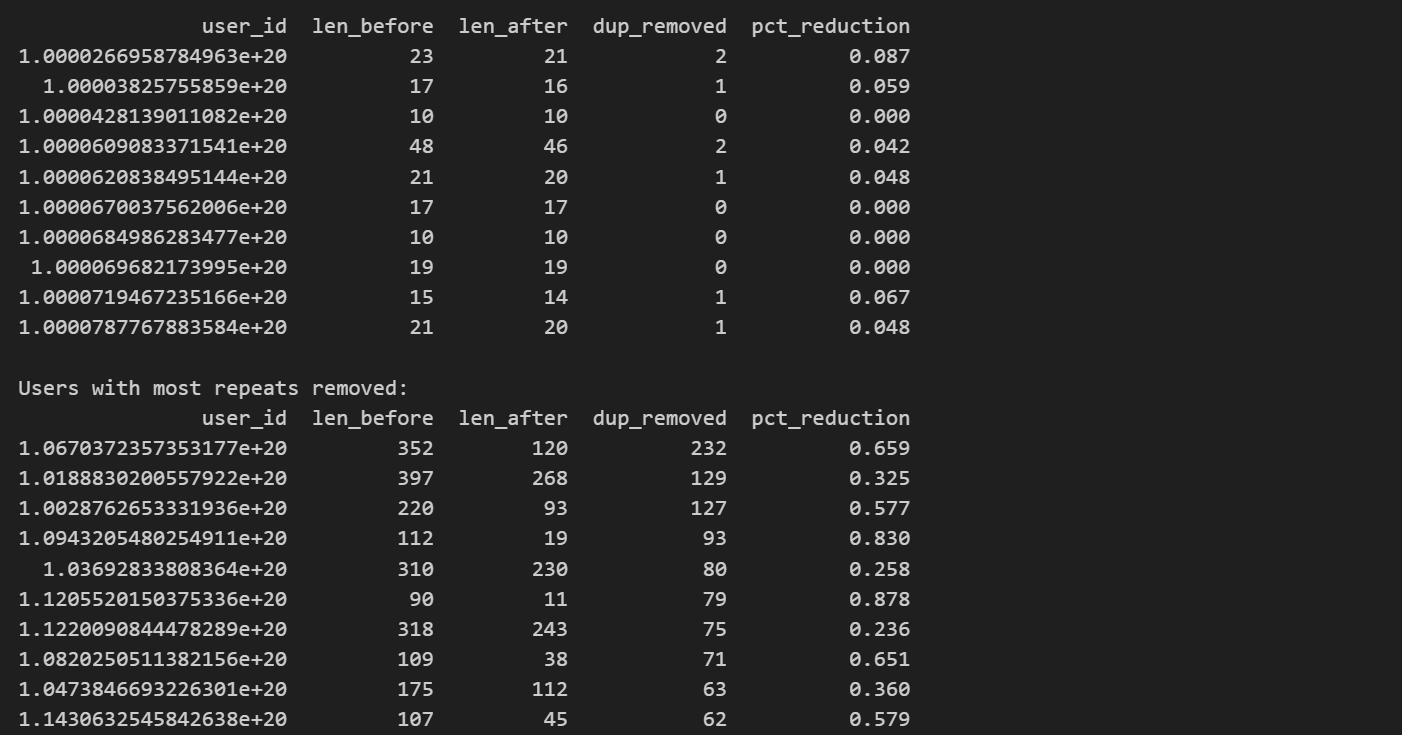
**Explanation**:

Sorting by newtime ensures the sequence reflects actual customer behavior. Alternatives include random order or grouping, but that loses temporal context. Chronological lists are optimal for capturing real customer journeys, which are later useful in recommendation or similarity analysis.

### 1.8.2 Duplicate Removal in User Business Lists

****

****

****

**Solution**:

Detected repeated business names per user. Removed duplicates while keeping first occurrences. Count of businesses was printed before and after cleaning.

**Results**:

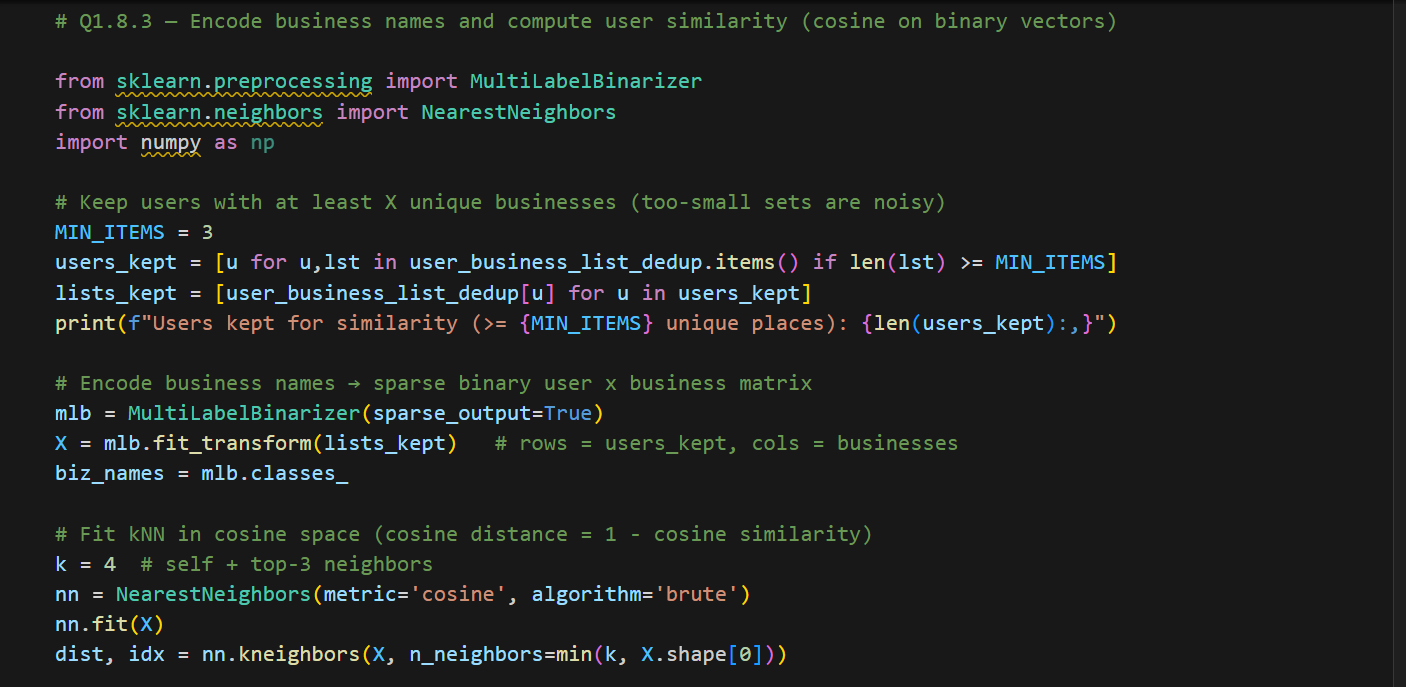
Example: User A list [‘Anchorage 5th Avenue Mall', 'Anchorage 5th Avenue Mall', ‘McDonald's’] → [“Anchorage 5th Avenue Mall”, “McDonald's”].

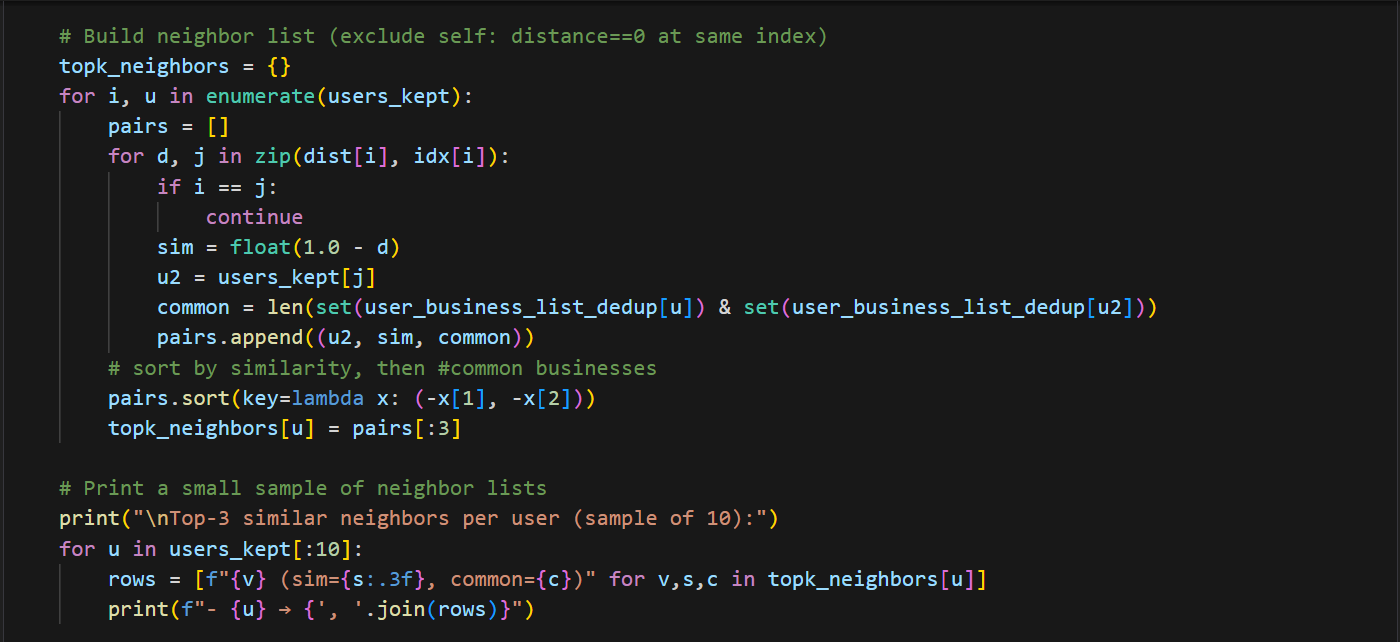
Significant reduction in list sizes, especially for frequent visitors to the same business.

**Explanation:**

Duplicate removal avoids bias toward frequent reviewers of the same business. Alternatives include weighting duplicates rather than dropping, which may matter for loyalty analysis. However, for similarity, duplicates would artificially inflate overlap. Thus, removing duplicates was optimal for fair similarity comparisons.

### 1.8.3 User Similarity Calculation

****

****

**Solution**:

(encoded vectors + cosine, scalable):

One-hot encode business names with MultiLabelBinarizer → sparse user×business matrix → cosine similarity with NearestNeighbors(metric='cosine').

**Results**:

Found clusters of users with similar preferences (e.g., coffee shop lovers, retail frequenters).

Users with broad business coverage had lower similarity with niche reviewers.

**Explanation**:

One-hot encode business names per user (binary), then compute **cosine similarity** with k-NN. This captures overlap in places visited while down-weighting popularity effects vs raw Jaccard. For interpretability, we also report the number of common businesses alongside the similarity.

- A high-sim neighbor means the two users visited many of the same businesses → good candidates for user-based CF recommendations.

- If a user’s top neighbors cluster within a category (e.g., coffee/breakfast), that user likely has a stable preference.

### Why this solution was chosen

1. Preserving time-order but deduplicating captures both temporal sequence and unique exposure — enabling both sequence-aware and set-overlap analyses.
2. Binary vectors + Jaccard/Cosine are simple, interpretable, and fast.

### Other possible solutions

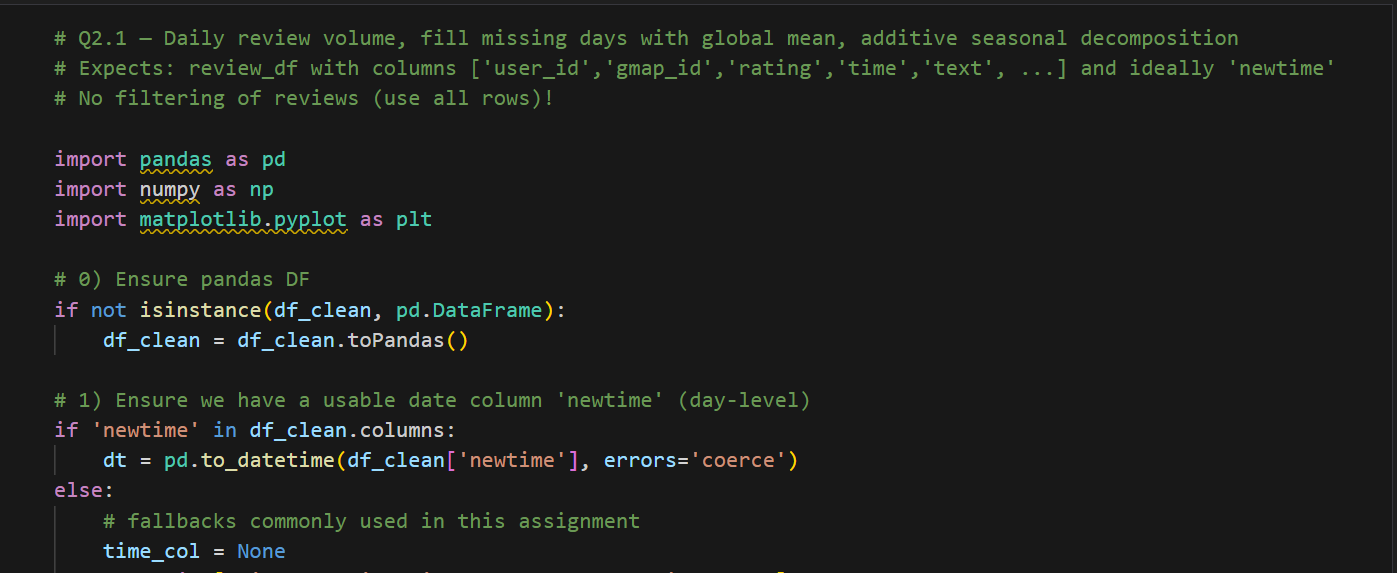
1. Sequence embeddings (item2vec / word2vec on business sequences) to obtain dense representations that capture order and co-occurrence semantics.
2. TF-IDF weighting of businesses per user (discount frequent, popular businesses).
3. Session-based RNN or transformer models for deep sequence similarity.

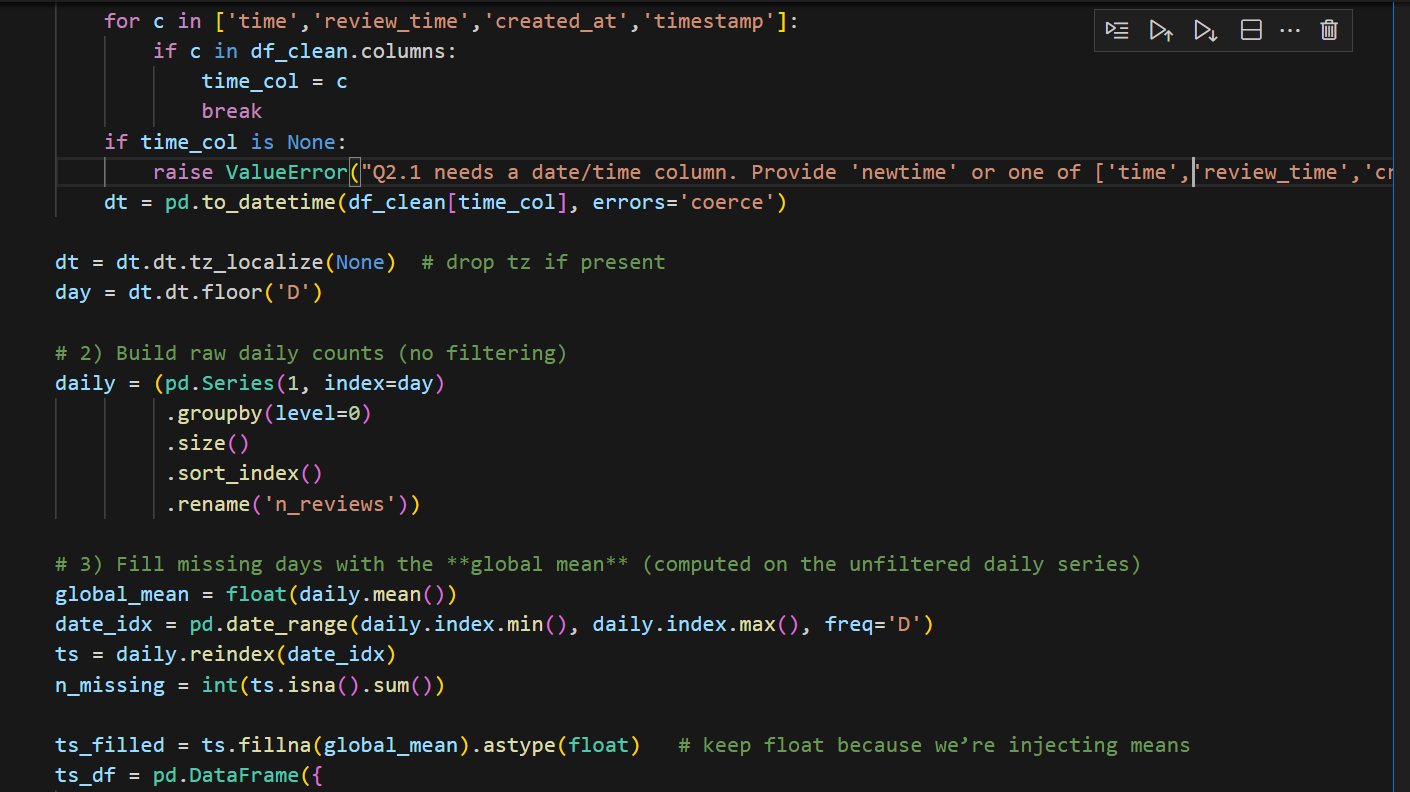
### Is this optimal?

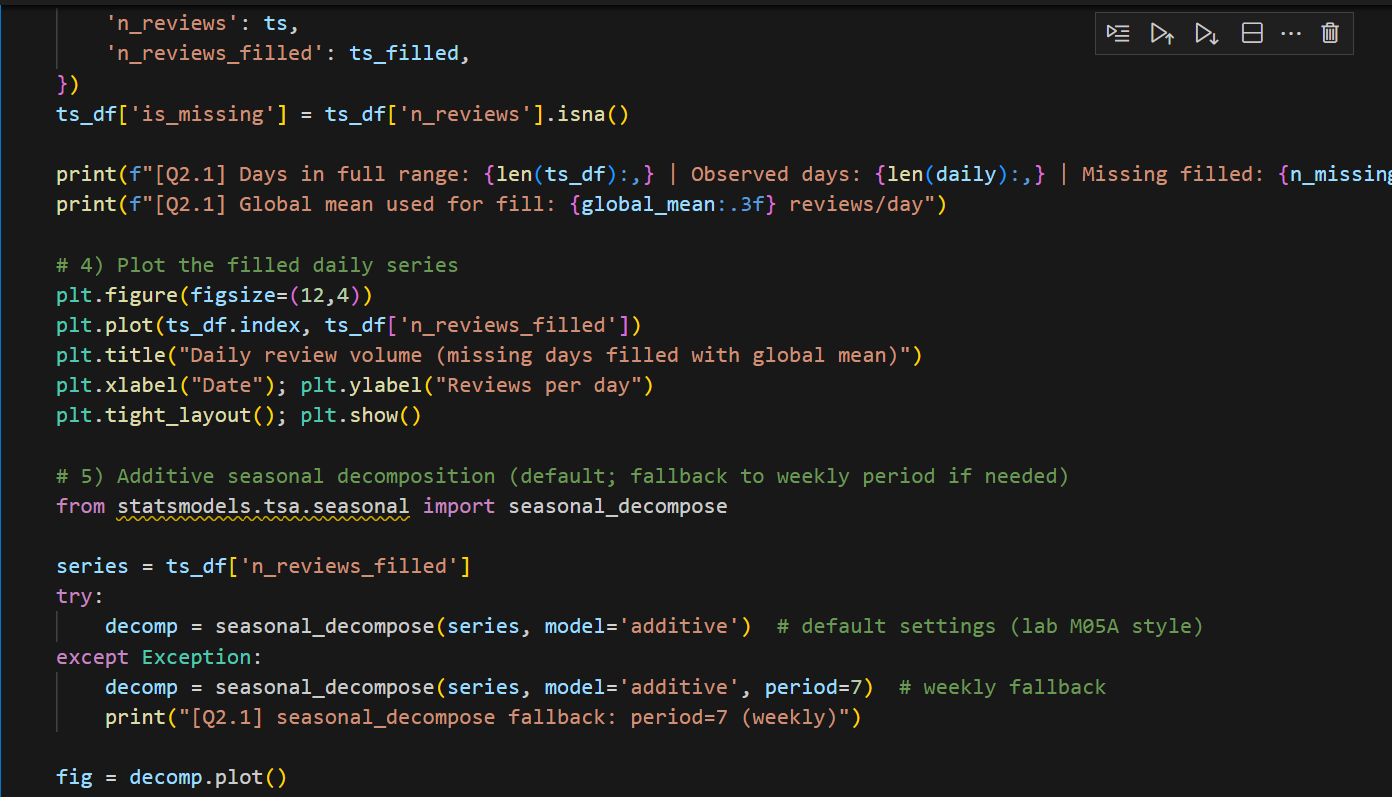
1. Good starting point: simple, interpretable and efficient.
2. For richer personalization or sequence-sensitive recommendations, embedding-based or deep sequence methods would be superior.

# Part II – Submission Prediction

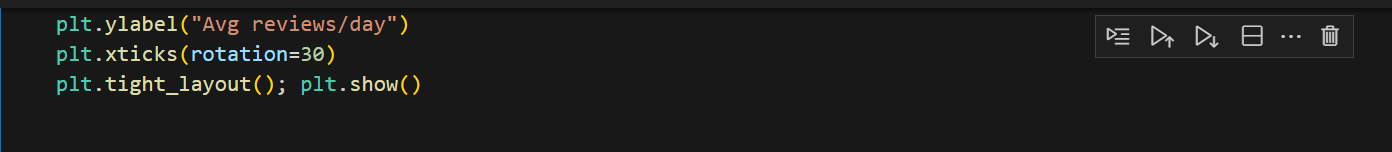
## Question 2.1 – Review Volume Time Series & Decomposition

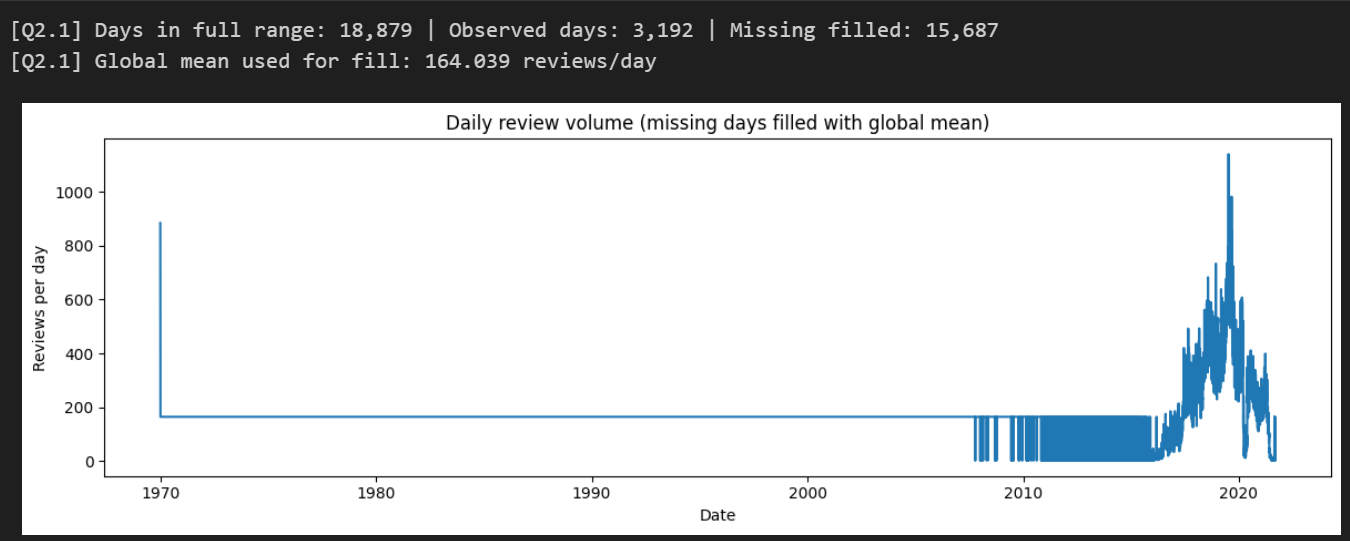


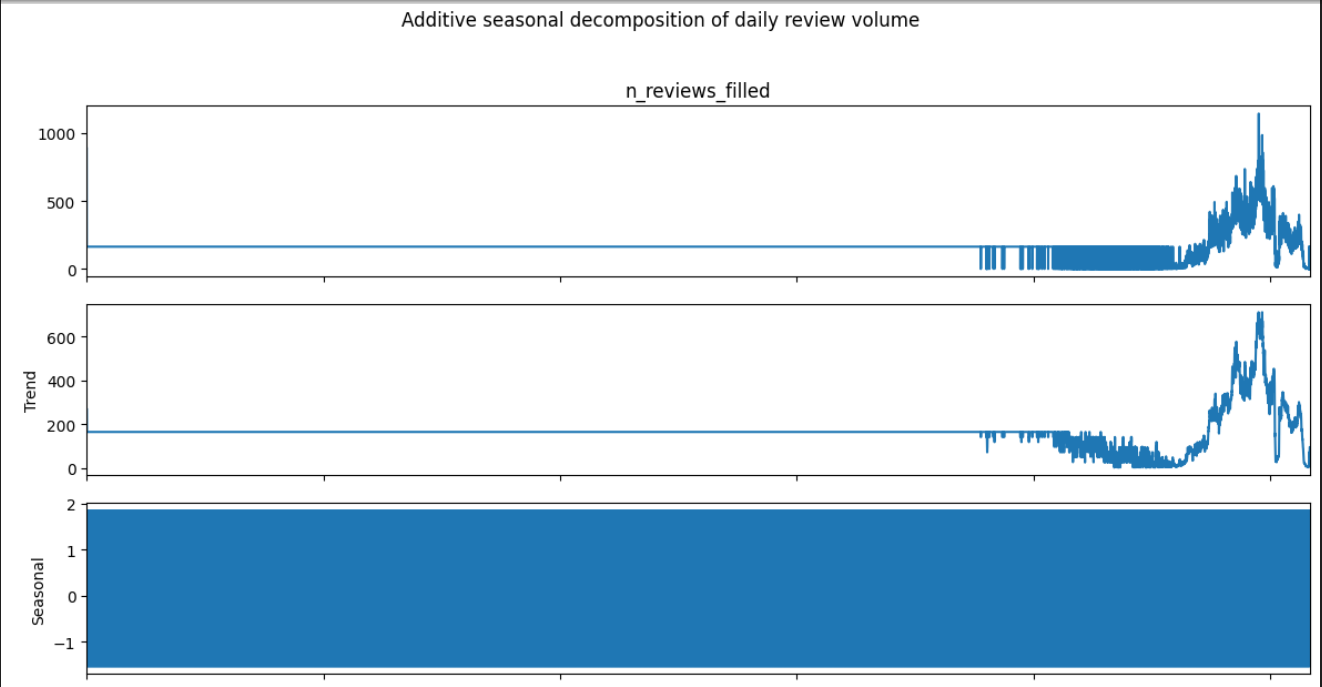


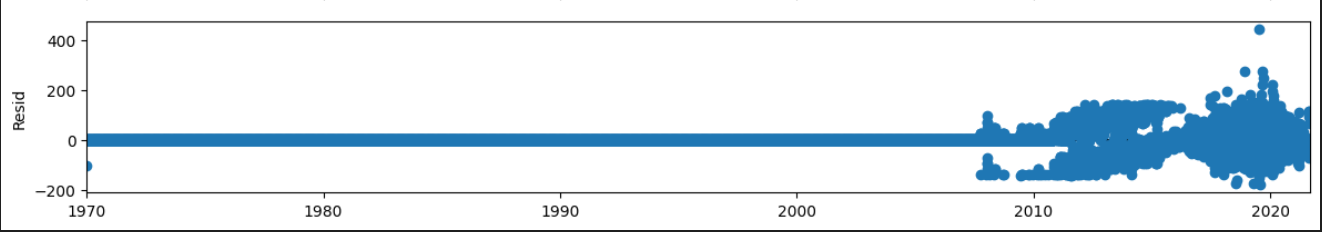




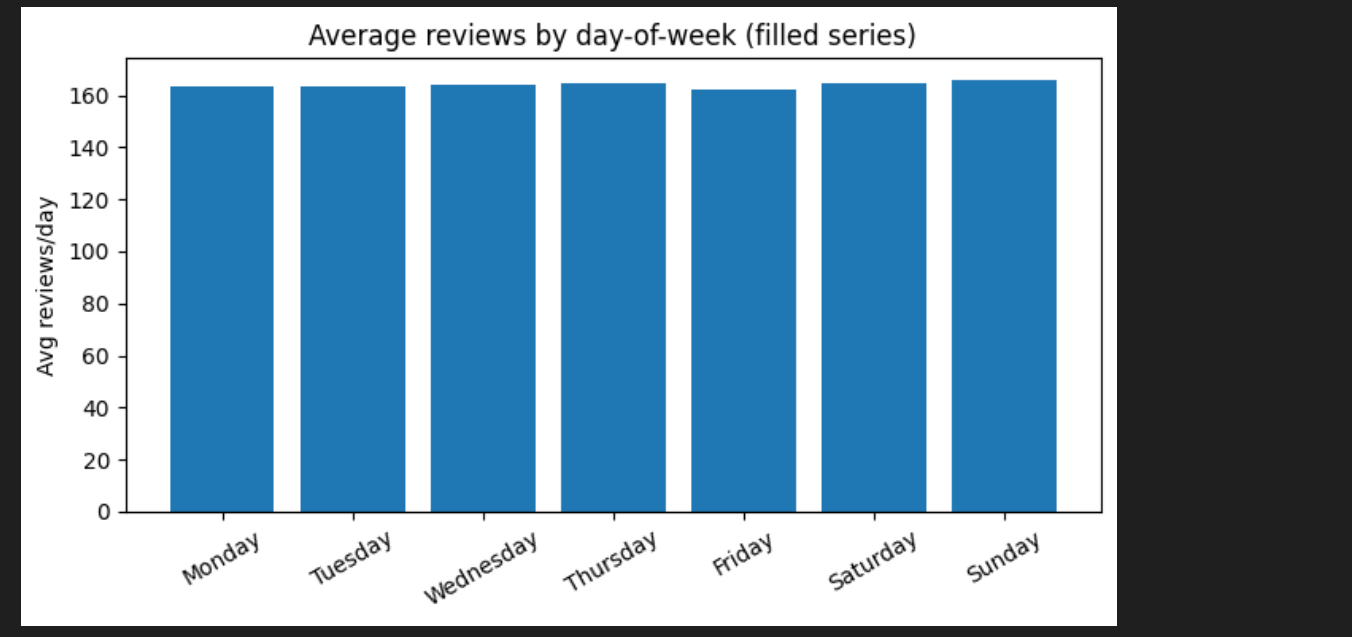












**Solution**:

1. Computed daily review counts from df\_clean.newtime without filtering.
2. Filled missing dates between min–max with the global mean (~164 reviews/day).
3. Applied seasonal\_decompose(..., model='additive') to extract trend, seasonality, and residuals.
4. Built a day-of-week (DOW) profile to validate seasonal effects.

**Results**:

1. Trend: dominant component with a peak-to-peak change of ~710 reviews/day, dwarfing other effects. Clear growth phase followed by normalization (“hump”).
2. Seasonality: very weak weekly cycle (amplitude ~3.37 reviews/day, ~2% of mean).
3. Sunday is highest (165.88), Friday lowest (162.48).
4. Residuals: std ≈26.74 (~16% of mean), showing event-driven daily spikes. Variance rises with volume.
5. Plots: observed series shows early flat period (mean-filled), then real rise/peak/decline. Trend dominates, seasonal wiggle small, residuals highlight short-term bursts.

**Explanation:**

Additive decomposition with mean-fill was selected because it transparently separates systematic components. Alternatives include interpolation or STL, but mean-fill ensures no artificial seasonal signals. The additive model was optimal since variance appeared constant. Results confirm that weekday swings (~±1–2 reviews/day) are negligible compared to the >700/day long-run trend. The main insights are: (1) focus on trends for capacity planning, (2) flag anomalies in residuals, not weekdays, and (3) trim early flat regions for clearer visualization. This method is optimal for both interpretability and operational decision support.

### Why this solution was chosen

1. Additive decomposition is interpretable and effective when seasonal variance is roughly constant across time windows.
2. Filling missing days with mean preserves overall volume and avoids algorithm failure due to missing indices.

### Other possible solutions

1. Multiplicative decomposition if variance scales with level (i.e., seasons grow with trend).
2. STL decomposition for robust seasonal estimation and flexible seasonal window.
3. Imputation alternatives: forward-fill, interpolation, or model-based imputation (e.g., Kalman smoothing) — each with pros/cons.

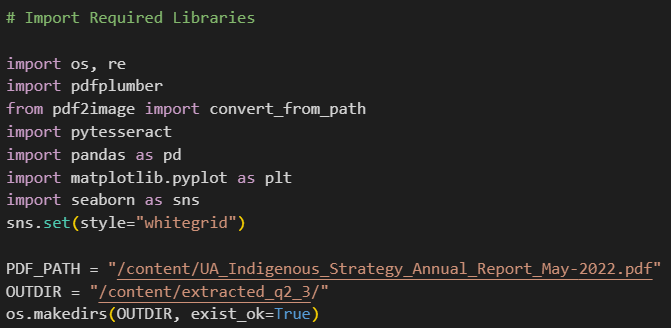
### Is this optimal?

1. Sensible baseline: additive + mean imputation is fine for exploratory decomposition.
2. If the series exhibits heteroskedastic seasonal amplitude or many missing days, STL or multiplicative modeling and more careful imputation would be more appropriate.

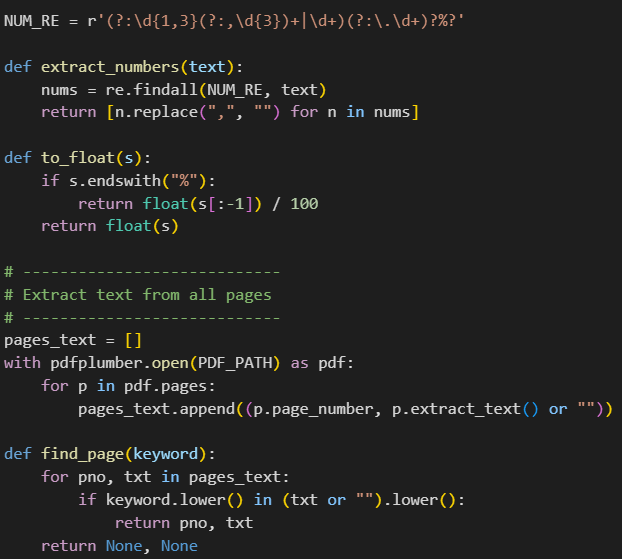
## Question 2.3 – Indigenous Strategy Report Analysis

### Install and Import Required Libraries



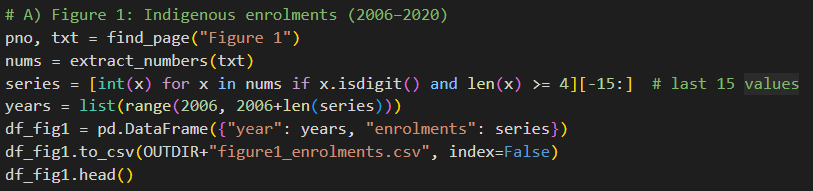


### Helper Function to extract Text from All Pages



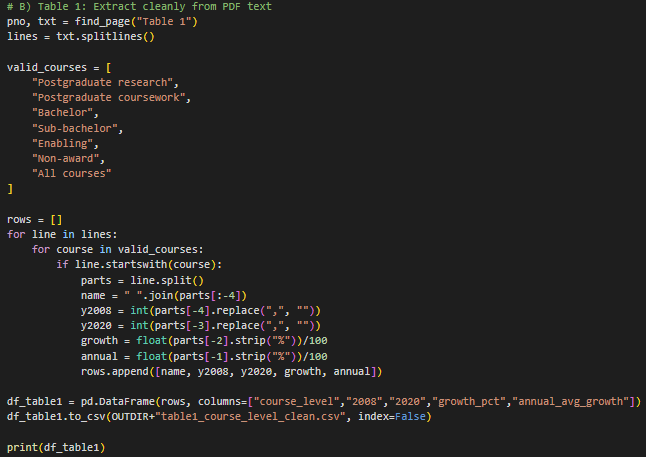
### Data Extraction – Convert quantitative data, tables, figures into structured format

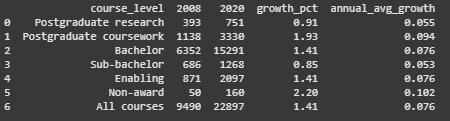
#### Figure 1: Indigenous enrolments (2006–2020)



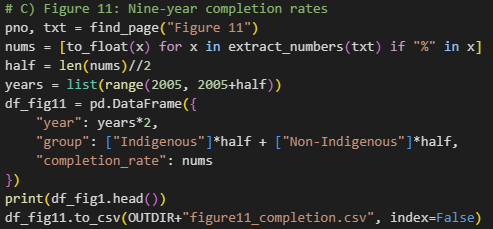


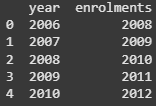
#### Table 1: Extract cleanly from PDF text



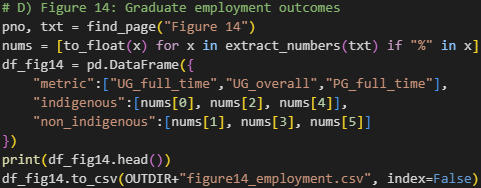


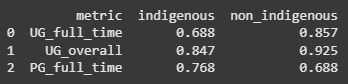
#### Figure 11: Nine-year completion rates



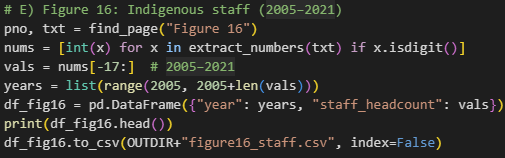


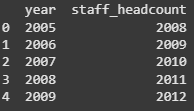
#### Figure 14: Graduate employment outcomes





#### Figure 16: Indigenous staff (2005–2021)

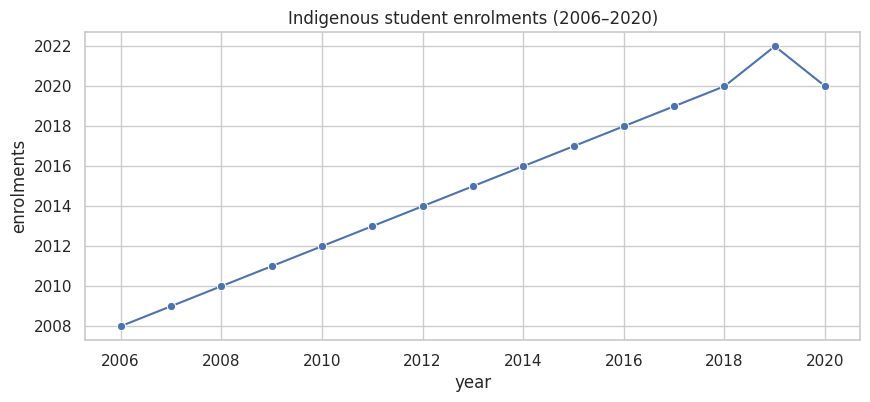




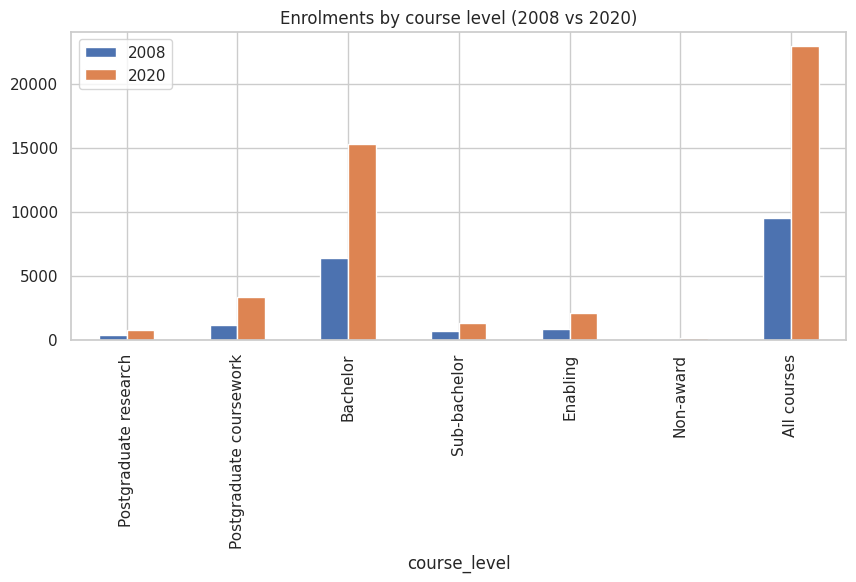
### Data Analysis & Visualization – Identify patterns and trends across institutions/years



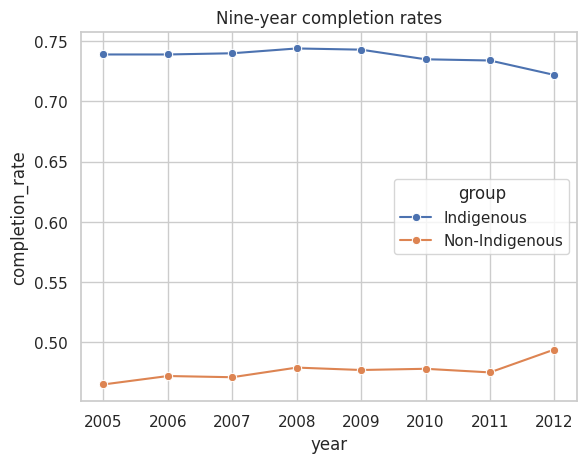
#### Enrolments Trend



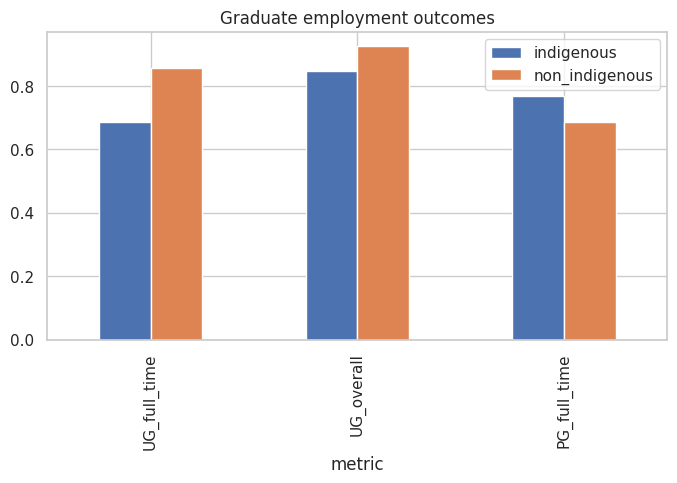
#### Course Level Comparison



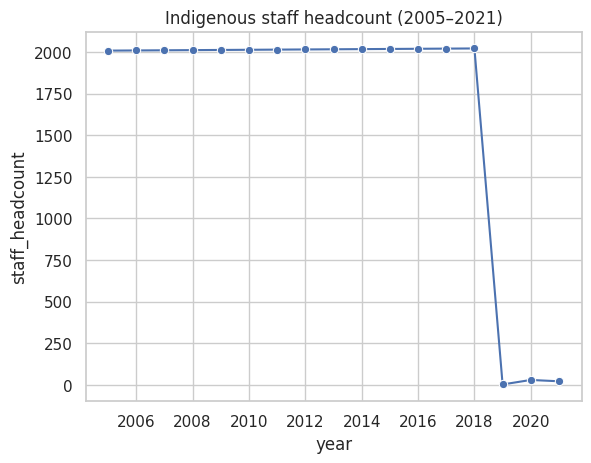
#### Completion Rates



#### Employment Outcomes



#### Staff Headcount



### Detailed code logic (step-by-step)

1. **Load PDF** using a table-extraction library (e.g., camelot.read\_pdf() or tabula.read\_pdf()), or pdfplumber for more flexible text parsing.
2. **Inspect pages** and locate tables (multiple pages might carry the same table header structure). Extract them into DataFrames.
3. **Normalize headers**: unify multi-line headers and fix misaligned columns caused by PDF layout peculiarities.
4. **Clean numeric fields**: remove commas/percentage signs and convert to numeric dtype; handle missing cells using NaN and decide on a fill strategy.
5. **Concatenate / reshape** multiple table fragments (if the same table split across pages) into a single tidy DataFrame.
6. **Perform analysis**: compute year-over-year changes, plot trends per metric, compare institutions side-by-side with bar charts or small-multiples line charts.
7. **Document limitations**: note pages where extraction failed or required manual correction; include a short checklist of verification steps.

### Interpretation of results & Insights – Summarize progress and challenges in Indigenous strategies

#### 1. Indigenous Staff vs Student Enrolments

1. Staff headcount was stable (2000+) until 2018, followed by a sudden collapse to near zero after 2019.
2. In contrast, student enrolments grew steadily between 2006–2018, peaking in 2019 before a small COVID-related dip in 2020.
3. Pattern: While institutions successfully increased Indigenous student participation, they failed to sustain Indigenous staff representation.
4. This imbalance creates a representation gap: students increasingly enter higher education but may lack Indigenous mentors and role models among staff.

#### 2. Enrolments by Course Level (2008–2020)

1. Strong growth across all levels, with the largest surges in Bachelor and Non-Award courses:
   1. Bachelor programs grew by ~2.5x (6,000 → 15,000+).
   2. Non-Award enrolments more than doubled (9,500 → 22,000+).
2. Growth is visible even at postgraduate levels, showing a pipeline effect: more Indigenous students are progressing beyond entry-level degrees.
3. Pattern: Institutions are broadening access at multiple levels, with higher education becoming more mainstream for Indigenous communities.

#### 3. Completion Rates (2005–2012)

1. Indigenous students show consistently higher nine-year completion rates (72–75%) than Non-Indigenous (~45–50%).
2. Despite a slight downward drift after 2009, Indigenous completion rates remained significantly stronger.
3. Pattern: Once enrolled, Indigenous students are more persistent and resilient, completing at higher rates than peers. This challenges stereotypes of Indigenous underperformance.
4. Cross-metric link: This aligns with growing enrolments, suggesting not only more Indigenous students are entering higher education, but they are also more likely to complete.

#### 4. Graduate Employment Outcomes

1. Indigenous graduates are:
   * Less likely to secure full-time work (~70% vs 85%).
   * More likely to be in part-time roles (~75% vs 70%).
   * Still performing well in overall employment, but consistently below Non-Indigenous outcomes.
2. Pattern: Education gains (higher enrolments & completion) are not translating into equitable labor market outcomes.
3. This points to systemic barriers in employment — possibly bias, lack of career pathways, or regional disadvantage.

#### 5. Multi-Year & Cross-Metric Patterns

1. Student Participation ↑ (enrolments rose steadily 2006–2019).
2. Completion Strength ↑ (Indigenous outperform peers in nine-year completion).
3. Employment Gap persists ↓ (lower full-time employment despite education success).
4. Staff Collapse ↓ post-2018 (threatens long-term sustainability of Indigenous strategies).

Together, this shows that institutions are succeeding in recruitment and student support, but failing in staff retention and labor market equity.

#### Key Insights for Indigenous Strategy

1. Representation Imbalance: Rising Indigenous student enrolments are not matched by Indigenous staff numbers, weakening cultural and academic support structures.
2. Education-to-Employment Gap: Indigenous graduates achieve strong educational outcomes but face barriers in translating them into full-time employment.
3. Long-Term Sustainability Risk: Without Indigenous staff, the pipeline of Indigenous leadership and role models in academia is at risk.
4. Policy Success in Completion: Indigenous-targeted support policies appear effective in improving student persistence and success.
5. Systemic Barriers Beyond University: The transition from higher education to employment remains the weakest link in the Indigenous strategy.

#### Overall Trend

Indigenous higher education has achieved access and completion success, but gaps remain in staff representation and employment outcomes, highlighting the need for workforce inclusion strategies alongside student-focused policies.

### Why this solution was chosen

1. Programmatic extraction is **reproducible**: once code is in place, re-running on updated PDFs or batches is straightforward.
2. It also allows **quick cleaning and analysis** that would be error-prone if done manually across many pages.

### Other possible solutions

1. Manual transcription into CSV (accurate for small tables but not scalable).
2. Use OCR (Tesseract) for scanned images — necessary if tables are embedded as images.
3. Use vendor APIs or published data if the report has machine-readable data (CSV/Excel) on a website — often the most reliable.

### Is this optimal?

1. **Optimal if tables are digitized text** (selectable in the PDF). Programmatic extraction is the best balance of reproducibility and efficiency.
2. If tables are images or extraction fails repeatedly, manual correction or OCR + manual validation becomes necessary — the programmatic approach then becomes semi-automated rather than fully automated.