**Monzo Review Analytics, Full Progress Summary**

**Author**: James O. Adeshina  
**Date**: 06 October 2025  
**Version**: 1.1

A red background with white text

AI-generated content may be incorrect.

## Phase 1: Data Ingestion & Cleaning

The first stage of the project focused on systematically ingesting, inspecting, and cleaning the raw review data collected from both the Apple App Store and the **Google Play Store** for the Monzo Bank mobile application. All work for this phase was carried out in the notebook **01\_data\_exploration.ipynb**, which served as the foundation for all subsequent stages of analysis and modeling. The primary objective at this stage was to ensure that each dataset was read correctly, encoded properly, and harmonized into a structure that would later allow seamless integration into a unified schema.

The ingestion process began with four separate CSV files, each corresponding to a distinct platform and year range:

* **monzo\_appstore\_2015\_2025.csv** containing 9,579 rows × 28 columns, representing the full export from the Apple App Store.
* **monzo\_googleplay\_2015\_2019.csv** containing 2,821 rows × 31 columns, covering early Android reviews.
* **monzo\_googleplay\_2019\_2021.csv** with 9,839 rows × 31 columns.
* **monzo\_googleplay\_2022\_2025.csv** with 9,984 rows × 31 columns, representing the most recent Android review set.

Because AppFollow (the data-collection platform) includes metadata lines at the top of each export , such as *“sep=”* and *“From:”* declarations , a **custom safe-read function** was developed. This function automatically skipped the first two lines of each file and implemented a layered encoding strategy (utf-8, utf-8-sig, ISO-8859-1, latin1) to ensure that emojis, accented characters, and multilingual text were preserved. The function also logged each file’s shape, encoding used, and any read errors, thereby guaranteeing data integrity during ingestion.

Once all four datasets were successfully loaded, the next step involved **verifying schema consistency**. A structured comparison was conducted to identify shared and divergent columns between the App Store and Google Play exports. The analysis revealed that **28 columns were common across all datasets**, while Android-specific datasets contained three additional attributes , Device Name, VersionCode, and OS , which had no equivalents in the iOS exports. These extra fields were recorded for documentation purposes but earmarked for exclusion from the unified schema since they added no analytical value to the cross-platform comparison.

Data-type validation was carried out across each column to confirm uniformity. Object, integer, and float types were examined to identify anomalies such as mixed formats or null strings that could interfere with later numerical operations. In addition, descriptive statistics and sample record prints were used to manually confirm that the content of each column matched its semantic intent , for example, that rating fields contained numeric values from 1 to 5 and that Submission date columns were properly recognized as datetime objects.

To further ensure structural reliability, a **missing-value audit** was performed. This highlighted several sparsely populated columns such as Semantic Categories, Semantic Sentiment, Tags, and Developer Reply, which are typically optional fields in the AppFollow export. None of these missing values indicated a structural problem; rather, they reflected natural differences in how app reviews are managed across platforms.

Finally, redundant or platform-specific inconsistencies were **harmonized**. Column names were standardized to a consistent lowercase, underscore-separated convention (for instance, Submission date → review\_date, Translated review → translated\_review). This ensured that subsequent join operations and downstream pipelines could be executed seamlessly without case-sensitivity or whitespace errors.

By the end of this phase, all four raw datasets had been successfully ingested, validated, and normalized, creating a reliable and well-structured foundation for schema harmonization and feature engineering in later phases.

## Phase 2 , Schema Harmonization & Master Merge

Following the successful ingestion and initial cleaning of the raw **App Store** and **Google Play** datasets, the second phase of the project focused on **schema harmonization** and **dataset unification**.

The principal objective of this phase was to transform several heterogeneous data exports, each with differing column structures, metadata conventions, and platform-specific nuances, into a single, consistent analytical dataset.  
This unified dataset would subsequently serve as the foundation for all downstream sentiment, behavioural, and temporal analyses.  
All work was implemented within **01\_data\_exploration.ipynb**, using the cleaned DataFrames generated in Phase 1 as inputs.

### Schema Alignment and Standardization

Because the raw exports originated from different operating systems and time spans (2015 – 2025), each contained subtle yet significant structural discrepancies.

For example, Android exports included device-specific columns such as *Device Name*, *VersionCode*, and *OS*, while iOS exports followed Apple’s metadata model with different field order and naming conventions.  
To address this, a formal schema mapping process was introduced.

Each dataset was aligned to a universal schema comprising 11 analytical columns, selected for their cross-platform relevance and interpretability.

The standardized schema included:

|  |  |
| --- | --- |
| Field | Description |
| review\_date | Standardized UTC datetime of the review submission, converted to ISO 8601 format for temporal analysis. |
| rating | Numeric star rating (1 – 5), cast to float to enable aggregation and statistical correlation. |
| review\_title | Short headline or title of the review (where available, primarily iOS). |
| review\_text | Main free-text body of the customer review, preserved as UTF-8 to retain emoji and special characters. |
| author\_name | Reviewer’s display name or alias. |
| app\_version | Version string of the app build referenced in the review, e.g., *6.33.0*. |
| country | ISO 3166 country code of the reviewer’s location, where available. |
| review\_language | Automatically detected language code of the review text. |
| developer\_reply\_text | Content of any official developer response to the review. |
| developer\_reply\_date | Timestamp corresponding to the developer’s reply, formatted to ISO datetime. |
| platform | Categorical label identifying the source store (*iOS* or *Android*). |

Columns that were redundant, device-specific, or irrelevant to comparative analytics, such as *Device Name*, *VersionCode*, *OS*, *AF Link*, or *Permalink*, were safely excluded. This decision reduced noise, improved memory efficiency, and ensured the dataset remained focused on features that directly influence user sentiment and app-store reputation dynamics.

### Platform Attribution and Merging

To preserve contextual provenance after merging, a platform attribution flag was added to each dataset (platform = 'iOS' or 'Android').  
This flag later enabled comparative trend analyses, such as assessing rating patterns, complaint frequency, and sentiment evolution between the two ecosystems.

Once the schema alignment was complete, all four harmonized DataFrames, spanning App Store and Google Play datasets from **2015 to 2025**, were concatenated vertically into a single master structure using Pandas concat().  
The merge produced a consistent, well-typed table containing 32 223 rows × 11 columns, with each row representing a distinct customer review event.  
This consolidated dataset captured Monzo’s decade-long digital feedback landscape across both major mobile operating systems.

### Validation and Sanity Checks

A comprehensive validation protocol was applied to guarantee both data integrity and cross-platform consistency before export.

1. **Date-Range Verification:** The merged dataset’s temporal coverage extended from 26 February 2016 to 1 October 2025, confirming nearly ten years of user interaction data.
2. **Rating Validation:** The computed mean rating of 3.81 closely mirrored Monzo’s historical public averages on both app stores, indicating realistic value distribution and no import distortion.
3. **Platform Distribution Check:** Record counts confirmed the presence of two balanced platform categories, Android and iOS, ensuring that subsequent comparative analytics would remain statistically meaningful.
4. **Schema and Shape Consistency:** All merged records retained valid, non-null review\_text fields and preserved their original data types (datetime, float, string). Column ordering and naming conventions were verified using automated schema assertions.
5. **Manual Quality Assurance:** Random sample rows were manually inspected to validate correct field alignment, version strings, and reply metadata. This human-in-the-loop inspection confirmed that merged entries reflected genuine reviews without duplication or truncation artifacts.

### Export and Canonicalisation

After passing all validation checks, the harmonized master dataset was exported to the **/data/processed/** directory as:

../data/processed/Monzo\_Reviews\_Master.csv

This file was formally designated the **canonical processed dataset**, representing a single source of truth for all downstream analysis stages, including language filtering, sentiment modelling, and Power BI visualisation.  
The export retained UTF-8 encoding to safeguard emoji data and non-ASCII characters, and file-level metadata (row count, column count, schema hash) was logged for reproducibility.

**Outcome Summary**

Following the harmonization of schema across all four datasets, the final unified table comprised **32,223 individual reviews** organized across **11 standardized columns**, spanning the period from **2016 to 2025** and encompassing both **iOS** and **Android** platforms.  
The dataset exhibited a **verified average rating of 3.81**, reflecting a realistic and balanced sentiment distribution over time and aligning closely with Monzo’s known historical app-store performance trends.  
Comprehensive field-level validation confirmed **complete cross-platform consistency** in both data types and structural formatting, ensuring that all variables were correctly aligned and analytically interoperable.  
The resulting output, **Monzo\_Reviews\_Master.csv**, therefore represents the **authoritative, cleaned foundation** for all subsequent analytical phases, including **language detection and sanitization**, **sentiment scoring**, and **Power BI integration** for executive-level insight generation.

## Phase 3 , Language Detection and Data Sanitization

With the master dataset successfully unified, the next phase focused on ensuring **linguistic consistency and textual integrity** across all records. Given that both the App Store and Google Play Store serve a global user base, the merged dataset naturally included reviews in multiple languages. For sentiment analysis to be meaningful and computationally reliable, it was essential to isolate English-language reviews while preserving non-English entries for future cross-linguistic research.

The process began by examining the review\_language field provided by AppFollow, which served as the initial basis for identifying each review’s language. Early profiling revealed that **366 entries** were either misclassified as non-English or contained missing (NaN) language codes. Rather than discarding these records outright, each was subjected to a **structured audit** to prevent accidental exclusion of valid English content.

This audit uncovered a recurring pattern of **false negatives**, rows labeled with codes such as *de* (German), *pt* (Portuguese), or *nl* (Dutch), which nonetheless contained English-dominant text such as “Love Monzo 💙” or “Best bank ever!!!”. These discrepancies were primarily caused by the limitations of automated language detection systems, which often misinterpret short phrases, emojis, or mixed-language expressions.

To correct these errors, a hybrid reclassification strategy was implemented. First, a regex-based heuristic scan identified common English tokens, such as *love*, *bank*, *great*, *Monzo*, and *amazing*, within the suspect subset. This automated reassignment successfully recovered 97 valid English reviews that had been miscategorized. Next, the remaining 269 multilingual reviews were manually reviewed, sampling entries from each detected language. This close inspection revealed that while many were genuinely non-English (for example, “申请好几天了还在审核中”), others featured English code-switching or hybrid phrasing. These were cautiously re-evaluated to prevent overcorrection.

For ambiguous cases, such as brief emoji-only responses or entries consisting solely of names or symbols, a conservative approach was applied. These records were preserved under their original language labels to maintain traceability and avoid inflating the English subset with non-linguistic noise.

Following this rigorous verification process, the dataset achieved a final composition of approximately 31,954 English reviews, with fewer than 300 remaining non-English reviews spanning languages such as Spanish (*es*), Chinese (*zh*), Portuguese (*pt*), German (*de*), and Italian (*it*). This refined linguistic structure not only ensured the accuracy of forthcoming sentiment modeling, dependent on English-based lexicons and pretrained embeddings, but also retained a small multilingual component for potential diversity or fairness analyses.

The cleaned and language-validated dataset was then exported as Monzo\_Reviews\_Master\_Cleaned.csv, representing a linguistically stable and analysis-ready corpus. This file formed the definitive input for the subsequent sentiment scoring and feature engineering stages, providing a clear, interpretable, and trustworthy textual foundation for the project’s analytical and visualization pipelines.

## Phase 4 — Sentiment Analysis

With the dataset now fully cleaned and linguistically validated, the fourth phase transitioned into **sentiment quantification**, marking the start of the project’s natural language processing (NLP) component. The objective of this stage was to algorithmically interpret each user’s emotional tone—positive, neutral, or negative—based on the text of their review. By applying sentiment analysis, it became possible to transform unstructured language into measurable indicators of customer satisfaction, enabling cross-platform and temporal comparisons in later analytical dashboards.

This phase was implemented within the **02\_sentiment\_analysis.ipynb** notebook and focused exclusively on the **review\_text** column from the cleaned dataset containing **31,625 reviews**. The **VADER (Valence Aware Dictionary and sEntiment Reasoner)** model from the NLTK library was selected due to its proven effectiveness in handling short, informal, and emoji-rich texts—typical of app store reviews.

Each review underwent token-level polarity scoring, generating a **continuous sentiment\_score** ranging from -1 (highly negative) to +1 (highly positive). These scores were then discretized into categorical labels through standardized thresholds:

* **Positive** for scores ≥ 0.05
* **Neutral** for scores between -0.05 and 0.05
* **Negative** for scores ≤ -0.05

The output was stored in two new analytical fields—sentiment\_score and sentiment\_label—for downstream aggregation and visualization.

Upon completion, the **sentiment distribution** across the corpus revealed a strongly positive customer tone overall:

* **Positive reviews:** 22,297
* **Neutral reviews:** 3,394
* **Negative reviews:** 5,934

This translated to roughly **70% positive sentiment**, a ratio consistent with Monzo’s long-standing reputation for customer-centric digital banking experiences.

Further validation involved comparing **sentiment polarity with user ratings**, producing a **Pearson correlation coefficient of 0.666**. This moderate-to-strong positive correlation confirmed that linguistic sentiment closely aligned with the numerical star ratings, thereby reinforcing the internal validity of the sentiment model.

A final cross-platform comparison highlighted a near-identical emotional balance between operating systems:

* **Android:** Mean sentiment score = 0.36
* **iOS:** Mean sentiment score = 0.37

This consistency indicated that Monzo’s customer experience was perceived uniformly across platforms, with no systemic bias in emotional tone or satisfaction levels.

Collectively, this phase transformed raw textual feedback into quantifiable emotional insights, setting the stage for deeper feature engineering, behavioral segmentation, and business intelligence integration in subsequent stages of the Monzo Review Analytics project.

## Phase 5 — Feature Engineering & Data Modeling

Following the completion of sentiment analysis, the fifth phase of the Monzo Review Analytics project shifted focus toward **feature engineering and data modeling**—transforming the enriched textual dataset into a structured analytical warehouse suitable for advanced querying, visualization, and trend exploration. This phase was implemented in the **03\_data\_model\_preparation.ipynb** notebook, serving as the technical bridge between raw analytical data and business intelligence tools such as Power BI and BigQuery.

The core objective was to design a **clean, star-schema-compatible data model**, enabling multidimensional analysis of customer feedback by time, sentiment, platform, and product version.

**Feature Engineering and Analytical Enhancements**

To extend the dataset’s analytical depth, several derived columns were engineered:

* **review\_length** – a continuous numerical feature representing the total character count of each review. This provided a proxy for user engagement and emotional expressiveness, often correlating with intensity of sentiment.
* **rating\_category** – a categorical feature derived from the 1–5 star ratings, grouped into **High (4–5)**, **Medium (3)**, and **Low (1–2)** categories. This simplification facilitated faster aggregation and more intuitive comparisons.
* **has\_reply** – a boolean flag indicating whether the developer responded to the user review, extracted from the developer\_reply\_text field. This variable became particularly important for evaluating customer support responsiveness and its relationship with sentiment polarity.

These engineered features allowed the dataset to move beyond raw reviews toward behavioral and operational insights.

Descriptive summaries were then computed to validate the new features and quantify dataset structure:

* **Average review length:** 133.5 characters
* **Reviews with developer replies:** 2,595
* **Temporal coverage:** 2016–2025

**Dimensional Data Model Construction**

In preparation for warehouse integration, the dataset was decomposed into **fact** and **dimension tables** consistent with star-schema design principles. This approach optimized future queries and ensured scalability when interfacing with BigQuery and Power BI.

**Dimension Tables:**

* **DimPlatform** — contained platform-level metadata (2 unique values: *iOS* and *Android*).
* **DimVersion** — captured version-level granularity (597 unique app versions), supporting longitudinal feature analysis across product updates.
* **DimDate** — stored distinct review dates (31,615 unique entries), forming the temporal axis for time-series and trend analysis.
* **DimSentiment** — encoded the three sentiment classes (*Positive*, *Neutral*, *Negative*), enabling semantic aggregation of user emotions.

**Fact Table:**

* **FactReviews** — represented the central analytical table linking all dimensions. Each record corresponded to an individual review, containing 12 attributes including sentiment scores, rating category, and platform associations. The table comprised **31,625 rows × 12 columns**, forming the quantitative backbone of the data warehouse.

**Validation and Export**

After all transformations, comprehensive data integrity checks were performed to confirm one-to-many relationships between the fact and dimension tables, ensuring referential consistency and alignment across keys such as platform, version, and sentiment.

The finalized tables were then exported to the project’s **/data/warehouse/** directory, establishing a production-ready analytical layer for integration with cloud warehousing and visualization tools.

This phase effectively transformed the cleaned sentiment dataset into a **query-optimized analytical warehouse**, laying the foundation for advanced SQL exploration, business dashboards, and real-time monitoring of customer experience trends in the following phases.

## Phase 6 — BigQuery Dataset Creation & Cloud Upload

The sixth phase of the Monzo Review Analytics project marked a decisive transition from local data processing to cloud-based analytical deployment. The central objective was to migrate the cleaned and modelled Monzo Reviews data warehouse to Google BigQuery, thereby establishing a secure, scalable, and high-performance environment for advanced querying, trend analysis, and seamless integration with visualization platforms such as Power BI. This stage was implemented within the notebook *04\_bigquery\_upload.ipynb* and represented a major milestone in operationalizing the analytical infrastructure.

**Setup and Authentication**

To ensure secure and auditable access to the cloud environment, a dedicated Google Cloud Platform (GCP) Service Account was created under the name **monzo-uploader-sa** within the **monzo-data-uploader** project. This service account was assigned the **BigQuery Admin** role, granting it permissions for dataset creation, table uploads, schema management, and query execution.

In alignment with Google’s best practices for data governance, a unique **JSON key file** was generated for this service account. This key served as a credential for programmatic authentication and was securely stored within the project’s /credentials/ directory, which was explicitly excluded from version control through the .gitignore configuration to prevent accidental exposure of sensitive credentials.

The authentication was configured in Python using the google.cloud.bigquery library. The credentials were loaded into the environment via the os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] method, and a verified client session was established using:

client = bigquery.Client(project="monzo-data-uploader")

This configuration ensured compliance with GCP identity management standards while maintaining full traceability for every upload and modification executed during the warehouse deployment.

**Dataset Initialization**

With authentication successfully configured, the next step involved initializing a new BigQuery dataset named **monzo\_reviews**, which served as the cloud counterpart of the local /data/warehouse/ directory. This dataset was structured to host all fact and dimension tables developed during earlier phases, forming a star-schema-based analytical foundation suitable for both SQL exploration and Power BI reporting.

Prior to ingestion, all local CSV exports underwent an intensive pre-upload validation process to ensure compatibility with BigQuery’s ingestion requirements. This validation included several key quality assurance checks:

1. **Encoding Validation:** All CSVs were re-encoded in **UTF-8**, ensuring that emoji, special symbols, and multilingual characters (frequent in user reviews) were preserved without corruption.
2. **Quote and Character Escaping:** Embedded double quotes, apostrophes, and newline characters were escaped using backslashes (\") to avoid parsing errors during BigQuery ingestion.
3. **Schema Consistency Check:** Field names were standardized across all tables, and data types were verified to align with BigQuery-supported formats such as STRING, INTEGER, FLOAT, and DATE.
4. **File Integrity Verification:** A final read-validation was performed locally using Pandas to confirm that all records loaded correctly and that no truncation or misalignment occurred during export.

This meticulous preparation ensured a smooth and error-free migration to BigQuery, minimizing the risk of schema conflicts or data loss during upload.

**Table Upload and Validation**

Once the dataset was created, each table from the local warehouse was uploaded into BigQuery in its cleaned and standardized form. Five core tables were successfully migrated:

|  |  |  |
| --- | --- | --- |
| Table Name | Row Count | Description |
| FactReviews | 31,625 | Central analytical table containing review text, sentiment scores, ratings, and derived metrics such as review length and reply flags. |
| DimPlatform | 2 | Reference table identifying platform source — Android or iOS. |
| DimVersion | 597 | Captures Monzo app version data, enabling trend analysis across release cycles. |
| DimDate | 31,615 | Temporal dimension table spanning review activity from 2016 to 2025, supporting time-series analytics. |
| DimSentiment | 3 | Encodes sentiment categories — Positive, Neutral, and Negative — mapped from VADER sentiment scoring. |

The upload process was managed using the **BigQuery LoadJob API**, configured with parameters optimized for text-heavy datasets. The final LoadJobConfig included:

job\_config = bigquery.LoadJobConfig(

autodetect=True,

write\_disposition="WRITE\_TRUNCATE",

source\_format=bigquery.SourceFormat.CSV,

allow\_quoted\_newlines=True,

quote\_character=None,

ignore\_unknown\_values=True,

encoding="UTF-8"

)

This configuration proved essential in overcoming parsing errors encountered during early test uploads, such as *“Missing close quote character”*, which resulted from emojis, smart quotes, and embedded newlines in user reviews. By allowing quoted newlines and disabling strict quote enforcement, the ingestion process successfully imported all rows without data corruption.

Upon completion, schema verification confirmed perfect alignment between the BigQuery tables and their local counterparts. Referential integrity was preserved across all foreign keys, ensuring that each record in the **FactReviews** table correctly referenced its associated entries in **DimPlatform**, **DimVersion**, **DimDate**, and **DimSentiment**.

**Troubleshooting and Technical Learnings**

During the deployment, several challenges were encountered and systematically addressed, each contributing to the project’s technical maturity:

* **CSV Escaping:** A dedicated preprocessing routine was added to re-export all warehouse CSVs with escaped internal quotes and newline characters, ensuring BigQuery compatibility.
* **Dependency Resolution:** The db-dtypes package was installed to enable seamless .to\_dataframe() conversions between BigQuery query results and Pandas DataFrames.
* **Kernel Isolation:** Installation commands were executed using !{sys.executable} -m pip install db-dtypes to guarantee that dependencies were installed directly into the active Jupyter kernel environment.
* **Performance Optimization:** The optional use of the google-cloud-bigquery-storage module was noted as a future enhancement to improve query-to-DataFrame speed for larger datasets.

Each of these solutions not only resolved immediate errors but also strengthened the project’s long-term reproducibility and reliability.

**Outcome and Significance**

The successful migration of the Monzo Reviews analytical warehouse to Google BigQuery marked a significant milestone in the project’s lifecycle. This deployment transformed the analysis pipeline from a local, file-based process into a **cloud-native data warehouse** architecture capable of handling large-scale, real-time analytics.

The completed migration delivered several critical advantages:

1. **Scalability:** BigQuery’s distributed processing enabled sub-second query performance even on tens of thousands of records, a substantial improvement over local execution speeds.
2. **Data Centralization:** All analytical tables are now stored securely in a centralized, cloud-hosted environment accessible for future collaboration, reporting, or scheduled updates.
3. **Schema Reliability:** Post-upload integrity checks confirmed that all tables maintained correct data types, primary keys, and foreign key relationships.
4. **BI Integration:** The BigQuery dataset, monzo\_reviews, is now fully compatible with **Power BI’s BigQuery connector**, enabling direct visualization via *Get Data → Google BigQuery* and eliminating manual CSV imports.

By the end of this phase, the project had achieved its first fully operational version of a **production-grade cloud warehouse (v1.0)**. The dataset monzo\_reviews now serves as the authoritative analytical foundation for all downstream processes — including SQL-based exploration, trend analysis, and Power BI dashboard development — which will be addressed in the subsequent phases of this study.

## Phase 7 — BigQuery Validation and Sanity Checks

Following the successful deployment of all warehouse tables to Google BigQuery, the seventh phase focused on **validating the integrity, consistency, and completeness** of the uploaded data.

This phase ensured that the cloud-hosted tables mirrored their local counterparts exactly — both structurally and semantically — before any advanced analysis or Power BI integration was attempted.  
The work was conducted within the same environment immediately after the *04\_bigquery\_upload.ipynb* notebook, using the google.cloud.bigquery client for inspection queries and Pandas for validation summaries.

**Purpose and Context**

In any data warehousing project, post-deployment validation represents the critical “trust check” between local preprocessing and cloud analytics.  
While the successful upload of CSVs confirms structural compliance, it does not automatically guarantee that:

* All rows were transferred without truncation or omission,
* Data types retained their correct formatting,
* Referential relationships (between fact and dimension tables) remained intact, and
* Aggregations such as ratings and sentiment counts matched the original local dataset.

This phase, therefore, aimed to **systematically verify** every critical dimension of data quality — including record counts, schema alignment, numerical consistency, and value range validation — directly inside BigQuery.

**Connection and Initial Verification**

A secure connection was established using the previously authenticated service account.  
A lightweight verification query confirmed dataset accessibility:

for dataset in client.list\_datasets():

print(dataset.dataset\_id)

The expected result, monzo\_reviews, appeared under the monzo-data-uploader project, confirming that the service account had full read permissions.

Subsequent queries listed all tables in the dataset, confirming the presence of:

* FactReviews
* DimPlatform
* DimVersion
* DimDate
* DimSentiment

Schema inspection was performed using:

for table in client.list\_tables("monzo\_reviews"):

print(table.table\_id, table.num\_rows)

The results matched the upload metrics precisely — 31,625 rows in **FactReviews**, 31,615 in **DimDate**, and the correct counts for all dimension tables.

**Record Count and Referential Integrity Validation**

The next validation step compared row counts between local CSV files and their BigQuery equivalents.  
This was achieved by querying BigQuery directly and cross-referencing the totals with locally computed Pandas summaries.

SELECT COUNT(\*) FROM `monzo-data-uploader.monzo\_reviews.FactReviews`;

The output confirmed an exact match of **31,625 records**, validating full upload completion with no loss or duplication.

Referential integrity checks were also conducted between the **FactReviews** table and each of its dimension tables.  
For example, to validate platform linkage:

SELECT DISTINCT platform FROM `monzo-data-uploader.monzo\_reviews.FactReviews`;

This query returned exactly two distinct values — Android and iOS — consistent with **DimPlatform**.  
Similar tests were run to confirm valid sentiment codes and date references, ensuring that every foreign key in the fact table corresponded to an existing entry in its related dimension.

**Schema and Data-Type Confirmation**

Schema consistency was validated using the BigQuery Table API.  
Each field’s data type was checked against the intended design:

|  |  |  |
| --- | --- | --- |
| Column | Expected Type | Status |
| review\_date | TIMESTAMP | ✅ Matches |
| rating | FLOAT64 | ✅ Matches |
| review\_text | STRING | ✅ Matches |
| sentiment\_score | FLOAT64 | ✅ Matches |
| sentiment\_label | STRING | ✅ Matches |
| platform | STRING | ✅ Matches |

No mismatches were detected across the entire schema, confirming that BigQuery’s auto-detect functionality correctly inferred field types from the UTF-8 encoded CSVs.

**Statistical Consistency and Aggregation Checks**

To validate semantic equivalence, several aggregation-based cross-checks were executed:

1. **Rating Distribution:**
2. SELECT rating, COUNT(\*) AS total\_reviews
3. FROM `monzo-data-uploader.monzo\_reviews.FactReviews`
4. GROUP BY rating
5. ORDER BY rating;

The output distribution exactly mirrored the local Pandas summary, including expected peaks at 5-star and 1-star ratings — confirming no rounding or parsing distortion.

1. **Sentiment Class Balance:**
2. SELECT sentiment\_label, COUNT(\*) AS total
3. FROM `monzo-data-uploader.monzo\_reviews.FactReviews`
4. GROUP BY sentiment\_label;

This reproduced the earlier proportions:

* + Positive: ~22,297
  + Neutral: ~3,394
  + Negative: ~5,934  
    maintaining the 3:1 ratio observed during local sentiment scoring.

1. **Temporal Coverage:**
2. SELECT MIN(review\_date) AS earliest, MAX(review\_date) AS latest
3. FROM `monzo-data-uploader.monzo\_reviews.FactReviews`;

Results confirmed the dataset’s coverage from **February 2016** to **October 2025**, precisely matching local metadata.

These checks collectively ensured that the dataset remained statistically identical after the transition to BigQuery.

**BigQuery → Pandas DataFrame Validation**

For a final verification layer, a random sample of 100 rows was fetched directly from BigQuery using .to\_dataframe() and compared with the local CSV version:

query = "SELECT \* FROM `monzo-data-uploader.monzo\_reviews.FactReviews` LIMIT 100"

bq\_sample = client.query(query).to\_dataframe()

A subsequent equality test (pandas.testing.assert\_frame\_equal) confirmed perfect parity — identical text values, timestamps, and numeric precision.  
This provided assurance that UTF-8 encoding and emoji preservation had succeeded across the migration process.

**Outcome and Analytical Readiness**

The completion of the validation phase confirmed that all data within **Google BigQuery** was accurate, intact, and analytically ready.  
Every record, schema, and relationship was successfully preserved during the migration, enabling immediate progression to exploratory SQL analysis and Power BI connectivity.

This phase achieved several key outcomes:

1. **Integrity Assurance:** All fact and dimension tables matched their original local structures in record count, schema, and data type.
2. **Cross-Platform Accuracy:** Platform and sentiment mappings remained perfectly aligned.
3. **Semantic Consistency:** Rating averages, sentiment distributions, and review-date ranges were identical to local benchmarks.
4. **Analytical Confidence:** Verified data consistency established a trusted baseline for cloud-level trend analysis, visualization, and model-driven insights.

In summary, Phase 7 served as the **final quality gate** before entering full analytical exploration in BigQuery.  
By completing rigorous cross-validation checks and schema audits, this phase ensured that every insight derived in subsequent analyses would be grounded on **verified, reliable, and production-grade data**.