Data Pipeline Transformations and Changes Documentation

The purpose of this document is to provide a detailed record of all transformations, data cleaning operations, filters, enrichments, and other modifications applied to the data throughout the pipeline. This ensures traceability, reproducibility, and better understanding of the data processing logic.

1. Pipeline Overview

- Pipeline Name: mobile_customers_messy_dataset
- Source: Qversity public data from S3
- Target: Postgres (localhost)
- Technologies Used: Airflow, DBT, Python, PostgreSQL

2. Customer Data Cleaning and Standardization transformations.

I started by uploading the raw data to postgres so I could perform the necessary transformations using DBT, then I created a staging table called ‘cleanedmobile_customers_messy_dataset’ to extract the non-nested data to be able to change formats and eliminate nulls and duplicated data which I will describe below:

Customers Table:

A comprehensive data cleaning and standardization process was applied to the customer records, with the following main actions:

- First and Last Name Cleaning:
  Names were normalized to address inconsistencies such as blank spaces, numeric characters, and incomplete entries. Based on common prefixes (For Example, "car", "lau"), some names were imputed with standardized values like "Carlos" or "Laura".

- Email Validation:
  Blank or malformed email addresses were removed using a regex pattern to ensure only valid formats were retained.

- Phone Number Standardization:
  Phone numbers were cleaned by removing country codes (For Example, +57), special characters, and spaces. Only 10-digit Colombian numbers were retained using regular expressions.

- Age Cleaning:
  Age values were converted to integers using FLOOR(), and negative or invalid values were set to NULL.

- Country and City Standardization:
  Based on text patterns, country and city names were normalized to a standard set of values (For Example, “bog” → “Bogotá”, “co” → “Colombia”). A new structured reference table was created to manage this location data more consistently.

- Operator and Plan Type Mapping:
  Telecom operator names and plan types were cleaned and standardized (For Example, “cla” → “Claro”, “pre” → “Prepago”).

- Device Information Cleaning:
  Device brands and models were normalized. Special characters and unnecessary symbols in the device_model field were removed using regular expressions.

- Date Format Unification:
  Registration and payment dates were parsed from multiple formats (DD/MM/YYYY and YYYY-MM-DD) and converted to standard SQL date types.

- Status Normalization:
  Status values such as "active", "inactive", and "suspended" were inferred from partial matches and standardized accordingly.

- Invalid or Corrupt Records:
  Specific known corrupt values (For Example., an invalid UUID) were set to NULL, and any records missing critical fields like first_name, record_uuid, credit_score, or outside a valid age range (0–100) were filtered out.

- Deduplication:
  A two-step deduplication was applied:

    1. By email: To keep one entry per unique email address.

    2. By customer_id: To remove residual duplicates after city joins and filtering.

- Geolocation Rounding:
  Latitude and longitude values were rounded to six decimal places to ensure consistency.

- Metadata Tracking:
  A new column transformation_time was added to log the timestamp when transformations were applied.

Locations Table:

To address inconsistencies and inaccuracies in the geographical data, a new locations dimension table was created. This transformation was necessary due to frequent

mismatches between the country and city fields in the source data. For example, some records listed Peru as the country with Medellín as the city—an invalid combination.

Actions Performed:

- Country field ignored:
  Given the high rate of incorrect country assignments, the country column was excluded from the main customer dataset. Instead, focus was placed on the city field, which provided a more granular and reliable location reference.

- Standardization of city names:
  Cities were cleaned and normalized using pattern matching (ILIKE) to group similar variants under a standard name (For Example, bog, bogota → Bogotá, cdmx, ciudad → Ciudad de México).

- Country inference based on city:
  A set of rules was defined to infer the correct country for each city. This mapping was hardcoded and assigned a unique numeric location_id.

- Dimension table creation:
  The cleaned cities and their inferred countries were stored in a new reference table named locations, which includes three fields:

    - id: Unique identifier (location_id)

    - country: Standardized country name

    - city: Standardized city name

- Foreign key relationship:
  The location_id was then used as a foreign key to join this dimension table with the main customer data. This helped ensure that only valid, known city-country combinations were used, and improved data consistency.

- Filtering for valid locations:
  Only valid location_ids (less than 17) were included to avoid fallback or unidentified entries (For Example, id 999 for unknown combinations).

Payment_history Table:

In the original dataset, each customer record included a column named payment, which stored an array of historical payment records in JSONB format. To make this information queryable, analyzable, and aligned with best practices for relational databases, a normalization process was carried out.

Actions Performed:

- Filtering valid JSONB arrays:
  Only records where the payment field was a valid JSON array

(jsonb_typeof(payment) = 'array') were considered for transformation.

- Unnesting the JSON array:
  Using jsonb_array_elements, the nested structure was flattened so that each payment entry became a separate row, while retaining the customer_id as a foreign key.

- Field extraction and casting:
  From each payment object:

    - The date field was cast to a proper SQL DATE type as payment_date.

    - The amount field was converted to a numeric FLOAT, with invalid values (For Example., "unknown") converted to NULL.

    - The status field was extracted as plain text (payment_status).

- Creation of payment_history table:
  The result of this process is a new normalized table, payment_history, with the following fields:

    - customer_id: References the customer making the payment

    - payment_date: The date of the payment

    - payment_amount: The amount paid, or NULL if unknown

    - payment_status: The status of the transaction (For Example, completed, failed, pending)

- Ordering and validation:
  The output was ordered by customer_id and payment_date to validate chronological integrity and facilitate further time-series analysis.

Contrated_services Table:

The source data included a column named contracted_service, which stored a list of services subscribed to by each customer as a JSONB array of strings. To bring this data into a relational form and enable proper querying and analysis, the following transformation was performed.

Actions Performed:

- Validation and filtering:
  Only records with contracted_service fields recognized as valid JSON arrays

(jsonb_typeof(contracted_service) = 'array') were selected for processing.

- Array unnesting:
  Using jsonb_array_elements_text, each service within the array was expanded into its own row, while preserving the customer_id to maintain the relationship.

- Standardization:
  The extracted service names were converted to lowercase to ensure consistency and prevent duplication caused by casing differences (For Example, Roaming vs roaming).

- Creation of contracted_services table:
  This transformation resulted in a new table called contracted_services, with a simple and normalized structure:

    - customer_id: Foreign key referencing the main customer table

    - service: Name of the contracted service in lowercase

Calendar Table:
Following academic best practices and recommendations provided by professors at the university, a calendar (date dimension) table was created to improve the handling of date-type fields and support data scientists in performing robust time-based analysis.

Motivation:

- Handling inconsistent date formats:
  The source dataset included multiple date fields (registration_date, last_payment_date, and payment_history.date), which appeared in varying formats (DD/MM/YYYY and YYYY-MM-DD). These inconsistencies required normalization before any meaningful temporal analysis could be performed.

- Enabling time-based analysis:
  A dedicated calendar table simplifies common analytical tasks such as time aggregation, trend analysis, cohort grouping, and seasonality detection.

Actions Performed:

- Normalization of date fields:
  Three date sources were considered:

    - registration_date

    - last_payment_date

    - payment.date from the payment_history JSONB field
      All were converted to a consistent DATE type using format detection and

parsing logic.

- Union of all unique dates:
  A unified set of unique dates was compiled from all three sources to ensure complete coverage of relevant customer events.

- Enrichment with calendar attributes:
  Each date was enhanced with the following derived fields:

    - year, month, day
    - weekday (name of the day)
    - month_name (name of the month)
    - week_start (start of the week)
    - month_start (start of the month)

- Creation of calendar table:
  The final output is a comprehensive calendar dimension table, which supports easier and more efficient joins in analytical models and dashboards.