Submission for the Cloud Case Study

by

Tom Dobson

‘We help brands leverage data’

fifty-five UK Ltd.

A take-home assignment submitted for
the role of Analytics Engineer.

August 2024

Part 1: High-Level Solution

1.1 Conditions for Abandonment

To alert customers who have added items to their basket and then abandoned it, we must first define the conditions of basket abandonment. The following conditions can be used to form a basic definition of basket abandonment:

- **Condition 1:** During a session, one or more items are added to the basket.
- **Condition 2:** There has been a window of 45 minutes since the last session activity.
    - Accounting for the 15-minute delay in log data population, this provides users with a 45-minute to 1-hour window to complete the purchase before an abandonment alert is triggered.
    - Calculating the abandonment window relative to the last session activity is preferable to using the last basket addition time, as it reduces the likelihood of false positives for users who are still interacting with the store and considering their purchase.
- **Condition 3:** The session is not associated with a purchase. Any session with a purchase event will not be considered for abandonment. For this case study, it is assumed that a purchase can only be for all items in the basket.

> Although not tracked in this analysis due to time and data limitations, other conditions that could be considered include whether items were removed from the basket and whether they were purchased by the same user from a different device/session.

1.2 Events Tracking

Our next task is to establish which fields and values from `events_12_2020` can be used as indicators for our conditions, what supplementary information must also be captured for our analysis, and how these can be combined. The following events are logged in the `event_name` field and will be tracked to identify basket abandonment:

- **add_to_cart:** Occurrences of `add_to_cart` will determine if items were added to the basket during a session.
- **purchase:** An occurrence of `purchase` will determine if items were purchased during a session.
- **click, scroll, page_view, etc.:** Since the data only contains logs for the Google Merchandise Store, any other event can be used to determine if a user is still interacting with the store.

In addition to these events, the following dimensions will be included to enable us to create the Actionable Dataset and conduct our analysis:

- **event_timestamp:** This field contains the UNIX timestamp of the action performed (e.g., 1612091260872757) and is required to calculate the time of the last session activity. This field is at the same granularity as the main data and requires no transformation.
- **items.item_id:** This field from the `items` array contains the unique identifier of an item associated with an activity (e.g., 9190330). Since `items` is a nested and repeated field, it has a different level of granularity compared to the main table, which is at the event level. To access each individual `item_id` and enable us to concatenate them to create the list of abandoned products, we need to flatten this array. We can do this by using the `UNNEST` function, which effectively changes the granularity of the main data from an event level to the item level, allowing us to work with each item individually. Finally, we can use `STRING_AGG` to concatenate item IDs across rows.
- **email:** This field contains the email associated with each `user_pseudo_id` and is required to enable us to contact customers following abandonment. This field is from an external data source and must be joined to `events_12_2020` on the shared `user_pseudo_id` field. Because we can only alert users with a registered email, we will use an `INNER JOIN` to exclude sessions without an email.

1.3 Automation and Maintenance

Automation: Given that `events_12_2020` and our external source of user emails are updated automatically, we need to do the following to automate our analysis:

- Schedule our transformation to run in BigQuery every 15 minutes, seven days per week. While a view could also be used for our Actionable Dataset, a scheduled query is preferable as it allows for monitoring and alerts on failed runs, and can be more cost-effective if the dataset is queried frequently.

  > Given that our transformation is relatively simple and runs frequently, I've opted to execute it as a single scheduled query rather than breaking it down into a modular dbt project. This approach enhances performance and improves cost-efficiency by reducing the overhead associated with materialising multiple consecutive queries.

- Set up an API connection between Adobe Campaign and BigQuery to import data from our Actionable Dataset every 15 minutes, ensuring that the most recent data is available for email triggers.

Maintenance: There are several potential issues that could cause our analysis to fail or behave unexpectedly:

- **Scheduled Run Failure:** Solution: Implement run failure alerts to notify us of any issues with the scheduled transformations.

- **Upstream Errors:** Errors or null values in fields such as `user_pseudo_id` in either the events or email lookup tables could cause the join between these tables to fail, preventing email delivery for affected users. Solution: Regular data quality scans to identify and rectify such issues.

Part 2: Technical Specifications

2.1 Datasets and Data Fields

Source Data: The source data for this project comes from Google Analytics 4 (GA4) event logs, specifically from the `events` table in BigQuery. Additionally, a separate table containing user email addresses linked by `user_pseudo_id` will be used.

> The task description specifies that the data should be for December 2020. However, the provided link only includes data for January 1, 2021. To address this, I created a custom table by unioning data for all days in December 2020 (`events_12_2020`), which I then used as the source for the events data.

Fields Used in the Actionable Dataset: The following fields from the source data will be used to generate the Actionable Dataset:

Field Name	Source	Description
`email`	`email_lookup`	The email address of the visitor, used for triggering abandonment emails.
`user_pseudo_id`	`events_12_2020`	A pseudonymous identifier for the user.
`session_id`	`events_12_2020`	The session ID, created by concatenating `user_pseudo_id` and the `ga_session_id`.
`item_id`	`events_12_2020`, `items` array	The ID of the item added to the basket.
`event_timestamp`	`events_12_2020`	The timestamp of the event in microseconds, used to determine the abandonment time.
`normal_timestamp`	`events_12_2020`	A human-readable timestamp converted from `event_timestamp`.
`abandoned_products`	Derived	A concatenated string of item IDs that were abandoned in the basket.
`abandon_timestamp`	Derived	The timestamp at which the basket was considered abandoned.
`abandon_count`	Derived	The count of abandonments by the same user.

2.2 Temporary Datasets and Data Aggregation

Temporary Datasets: The following Common Table Expressions (CTEs) are used to create the final Actionable Dataset:

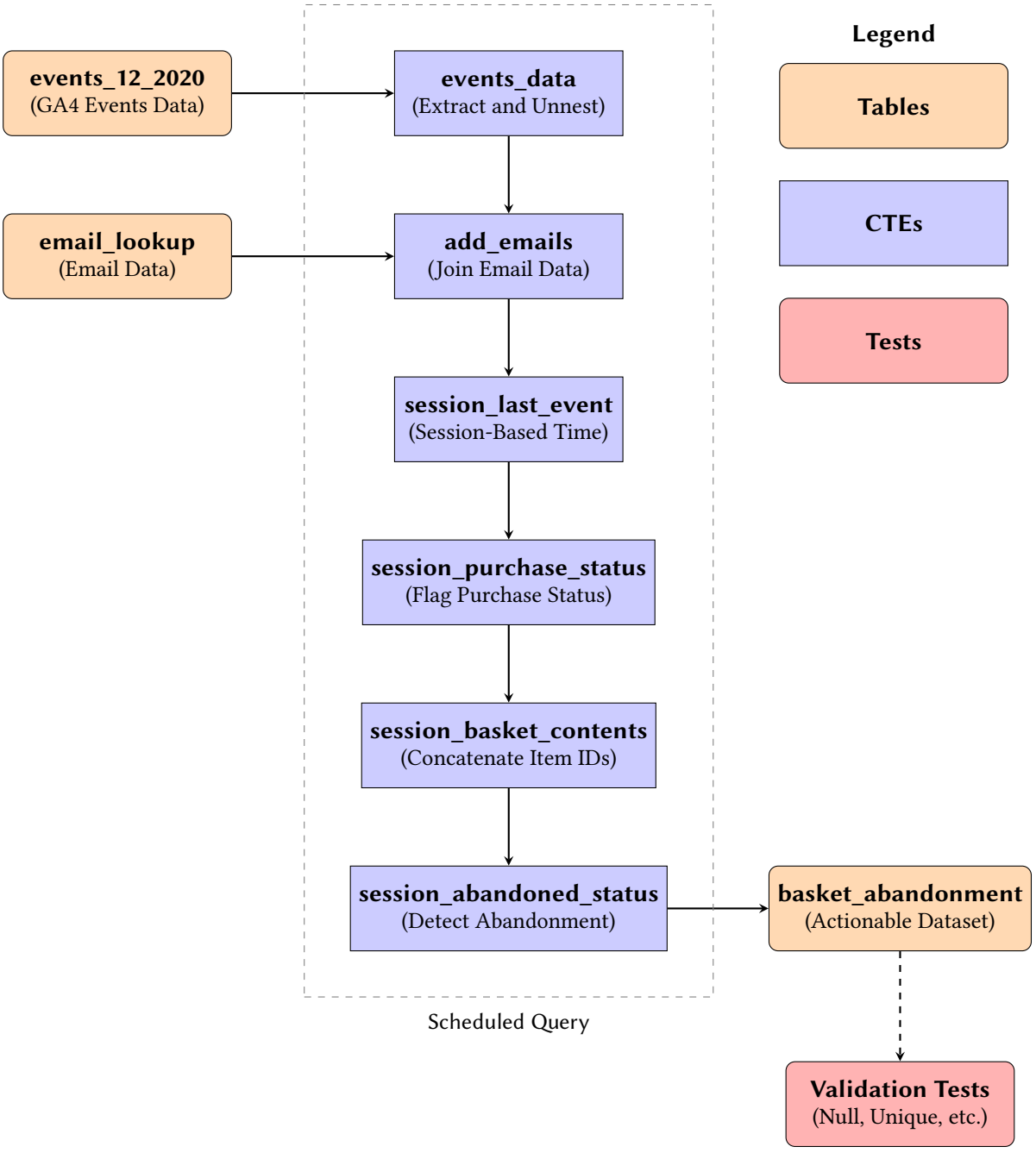
CTE Name	Purpose	Description
`events_data`	Initial extraction and processing	Extracts relevant fields from the `events_12_2020` table and unnests the `items` array.
`add_emails`	Email enrichment	Joins the `events_data` CTE with the email lookup table to associate users with their emails.
`session_last_event`	Session-based time calculations	Calculates the first and last event times for each session.
`session_purchase_status`	Purchase flagging	Determines if a purchase occurred in a session and flags it accordingly.
`session_basket_contents`	Abandoned products aggregation	Aggregates item IDs for products added to the basket during a session.
`session_abandoned_status`	Abandonment detection	Identifies whether a session is abandoned based on the absence of a purchase and the time elapsed since the last event.
`final_output`	Final dataset preparation	Produces the final Actionable Dataset with all required fields.

Data Aggregation Methods: The following methods are used to aggregate and transform the data:

- `STRING_AGG` : Used to concatenate multiple item IDs into a single string representing the products in the abandoned basket.
- `ROW_NUMBER` : Used to assign a sequential count of abandonment instances per user.
- `MAX` , `MIN` : Used to determine the first and last events in a session for time-based calculations.

2.3 Automation and Refresh Schedule

Query Scheduling: The query will be scheduled to run in BigQuery every 15 minutes. This ensures that the Actionable Dataset is updated regularly to reflect the latest user activity on the Google Merchandise Store.


events_12_2020
(GA4 Events Data)
email_lookup
(Email Data)
events_data
(Extract and Unnest)
add_emails
(Join Email Data)
session_last_event
(Session-Based Time)
session_purchase_status
(Flag Purchase Status)
session_basket_contents
(Concatenate Item IDs)
session_abandoned_status
(Detect Abandonment)
basket_abandonment
(Actionable Dataset)
Validation Tests
(Null, Unique, etc.)
Scheduled Query
Legend
Tables
CTEs
Tests

Adobe Campaign Integration: An API connection will be established between BigQuery and Adobe Campaign. Every 15 minutes, Adobe Campaign will query the Actionable Dataset in BigQuery to retrieve the latest abandoned basket data and trigger email notifications to users.

Part 3: Creating the Actionable Dataset

Attached to this submission, you will find the following files:

- `union_events.sql`: The SQL script used to union data for all days in December 2020.
- `basket_abandonment.sql`: The SQL script used to generate the final Actionable Dataset.
- `tests.sql`: SQL scripts to test and validate the Actionable Dataset.
- `actionable_dataset.csv`: The final Actionable Dataset for December 2020.

The SQL scripts used to create and test the Actionable Dataset are also available on GitHub.