Progress Checklist (short, living plan)

| Step | Item | Status | Notes |
| --- | --- | --- | --- |
| 1 | Choose dataset & explore raw GA tables | ✅ Done | We ran sanity queries and understood nesting (STRUCT/ARRAY). |
| 2 | Create dbt project scaffolding (dbt\_project.yml, folders) | ✅ Done | Project in C:\Projects\ga\_retail\_project. |
| 3 | Auth & profile (profiles.yml, service account) | ✅ Done | dbt debug connection OK. |
| 4 | Create BigQuery datasets (stg, mart) | ✅ Done | Created via bq CLI. |
| 5 | Create stg\_sessions.sql (this action) | 🔄 In progress | You’re creating the file now. |
| 6 | Run & test stg\_sessions | ⏳ Next | dbt run --models staging.stg\_sessions then dbt test. |
| 7 | Build stg\_hits\_products (UNNEST) | ⏳ Later today | Needed for item-level fact table. |
| 8 | Fact/dim models in mart | ⏳ Day 2 | Star schema (facts/dims). |
| 9 | Looker dashboard | ⏳ Day 3 | KPIs and story. |

GA Retail Analytics Project Documentation

# Introduction

This document captures the end-to-end progress of building a professional data analytics project using Google Cloud Platform (GCP), BigQuery, dbt, Python, and Looker. The purpose of the project is to showcase advanced analytics engineering skills by transforming and modeling Google Analytics e-commerce data into business-ready insights.

# Step 1: Project Framing and Dataset Exploration

Actions Taken:

- Selected the Google Analytics e-commerce dataset from BigQuery Public Datasets.  
- Justification: Dataset contains sessions, transactions, traffic source, and nested product data, making it ideal for modeling an e-commerce funnel end-to-end.  
- Explored dataset structure using SQL queries to understand nested STRUCTs (totals, trafficSource) and repeated arrays (hits, product).

Rationale:

BigQuery’s native handling of nested and repeated fields allows us to demonstrate advanced SQL skills (dot notation, UNNEST). Choosing this dataset highlights realistic e-commerce analytics scenarios (conversion, AOV, repeat purchases, ROI).

# Step 2: Environment Setup

Actions Taken:

- Installed Python 3.13 and created a dedicated virtual environment for isolation.  
- Installed dbt-bigquery within the virtual environment.  
- Justification: Virtual environments are industry best practice, preventing dependency conflicts and ensuring reproducibility.

Rationale:

Using isolated environments demonstrates professional engineering hygiene and avoids permission issues. It also ensures the project can be replicated consistently across machines.

# Step 3: BigQuery Dataset Preparation

Actions Taken:

- Created two datasets in BigQuery via CLI: `stg` (staging) and `mart` (analytics-ready tables).  
- Justification: Separating raw/staging from marts is standard practice in modern data engineering.

Rationale:

This layering approach ensures clean boundaries between data transformations and presentation. `stg` holds cleaned and standardized raw data, while `mart` contains fact and dimension tables optimized for BI dashboards.

# Step 4: dbt Connection Setup

Actions Taken:

- Created a `.dbt` directory under the user home folder.  
- Added `profiles.yml` with configuration for BigQuery (project ID, dataset, service account JSON key).  
- Created a service account in GCP, assigned BigQuery roles, and generated a JSON key.  
- Stored the JSON key securely and referenced its absolute path in `profiles.yml`.

Rationale:

Service accounts with JSON keys are the professional way to authenticate dbt with BigQuery. They avoid reliance on personal logins, support automation, and enforce the principle of least privilege.

# Step 5: Initialize Local dbt Project (Fixing dbt debug path)

Observation: Running `dbt debug` inside C:\Users\<user>\.dbt succeeded for connection, but failed with 'dbt\_project.yml not found'. This is expected because `.dbt` is only for credentials (profiles.yml). The dbt project (SQL models, dbt\_project.yml) must live in a separate project folder (e.g., C:\Projects\ga\_retail\_project).

Actions Taken / To Take:

- Ensure a dedicated project directory exists: C:\Projects\ga\_retail\_project  
- Create a local dbt project file `dbt\_project.yml` inside that folder.  
- Create the `models` folder structure and first staging model.  
- Re-run `dbt debug` from the project folder.

Minimal dbt\_project.yml:

name: ga\_retail\_project  
version: 1.0.0  
config-version: 2  
profile: ga\_retail\_project  
model-paths: ["models"]  
target-path: "target"  
clean-targets: ["target", "dbt\_packages"]  
models:  
 ga\_retail\_project:  
 +materialized: view

Rationale:

dbt separates credentials (profiles.yml in the user home .dbt folder) from the project code (dbt\_project.yml + models in your repository). This keeps secrets out of version control and lets the dbt project be portable and shareable.