Customer Purchase Trends Pipeline

Project Overview

The Customer Purchase Trends Pipeline is a cloud-based data pipeline designed to process and analyze retail transaction data. The pipeline uses **Google Cloud Storage (GCS)** and **BigQuery** to process over 3,900 retail transactions and generate actionable insights.

The primary goal of the pipeline is to demonstrate the end-to-end flow of data from ingestion to reporting, including data cleaning, transformation, and aggregation

Architecture and Components

The pipeline follows a standard extract, transform, load (ETL) process using several Google Cloud services.

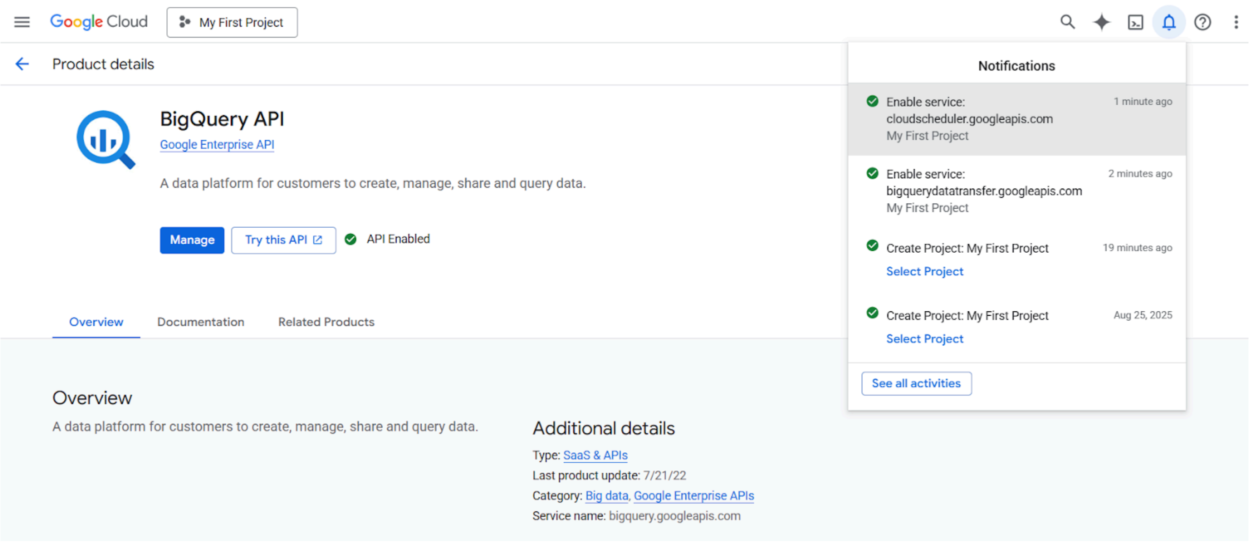
- **Ingestion (Bronze Layer)**: Daily raw CSV files are ingested and stored in a GCS bucket. This acts as the bronze layer, holding the raw, unprocessed data.
- **Transformation (Silver Layer)**: Data is transformed in BigQuery to clean missing values and standardize categories.
- **Aggregation (Gold Layer)**: The transformed data is aggregated to create fact tables in BigQuery, such as "purchases by age," "gender," and "season". These aggregated tables form the gold layer, which is optimized for analysis and reporting.
- **Visualization**: A Looker Studio dashboard is used to visualize key metrics like top product categories and seasonal sales trends.

Technical Implementation

Step 1: Project Preparation

First, several APIs must be enabled within the Google Cloud project to support the pipeline's functionality.

- **BigQuery API**: For creating, managing, and querying data.
- **BigQuery Data Transfer API**: Required for scheduled queries to automate data processing.
- **Cloud Storage JSON API**: For interacting with GCS buckets.
- **Cloud Scheduler API (Optional)**: Can be enabled later for scheduling tasks.



BigQuery API

Google Enterprise API

A data platform for customers to create, manage, share and query data.

API Enabled

Notifications

- Enable service: cloudscheduler.googleapis.com — My First Project — 1 minute ago
- Enable service: bigquerydatatransfer.googleapis.com — My First Project — 2 minutes ago
- Create Project: My First Project — Select Project — 19 minutes ago
- Create Project: My First Project — Select Project — Aug 25, 2025

Overview

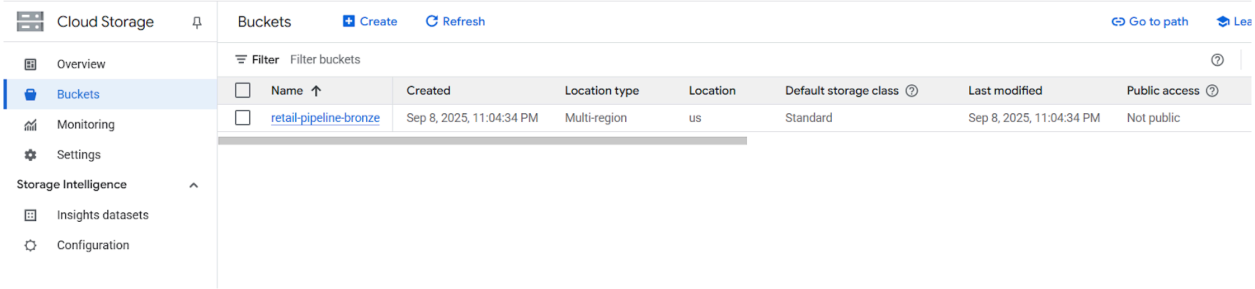
A data platform for customers to create, manage, share and query data.

Additional details

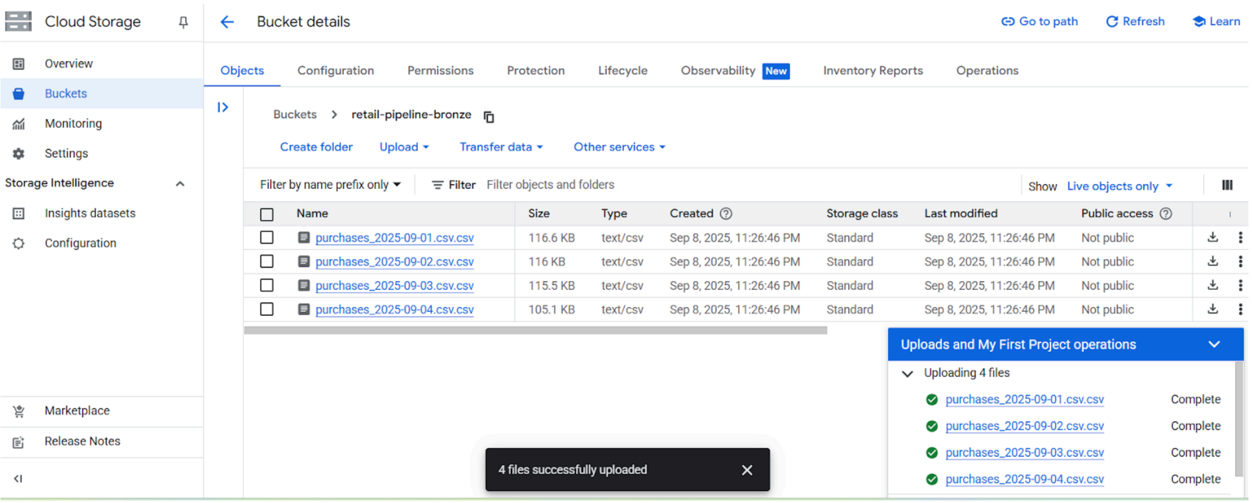
Type: SaaS & APIs
Last product update: 7/21/22
Category: Big data, Google Enterprise APIs
Service name: bigquery.googleapis.com

Step 2: Data Storage

A GCS bucket, named retail-pipeline-bronze, was created to store the raw daily CSV files.



Name	Created	Location type	Location	Default storage class	Last modified	Public access
retail-pipeline-bronze	Sep 8, 2025, 11:04:34 PM	Multi-region	us	Standard	Sep 8, 2025, 11:04:34 PM	Not public



Buckets > retail-pipeline-bronze

Name	Size	Type	Created	Storage class	Last modified	Public access
purchases_2025-09-01.csv.csv	116.6 KB	text/csv	Sep 8, 2025, 11:26:46 PM	Standard	Sep 8, 2025, 11:26:46 PM	Not public
purchases_2025-09-02.csv.csv	116 KB	text/csv	Sep 8, 2025, 11:26:46 PM	Standard	Sep 8, 2025, 11:26:46 PM	Not public
purchases_2025-09-03.csv.csv	115.5 KB	text/csv	Sep 8, 2025, 11:26:46 PM	Standard	Sep 8, 2025, 11:26:46 PM	Not public
purchases_2025-09-04.csv.csv	105.1 KB	text/csv	Sep 8, 2025, 11:26:46 PM	Standard	Sep 8, 2025, 11:26:46 PM	Not public

Uploads and My First Project operations

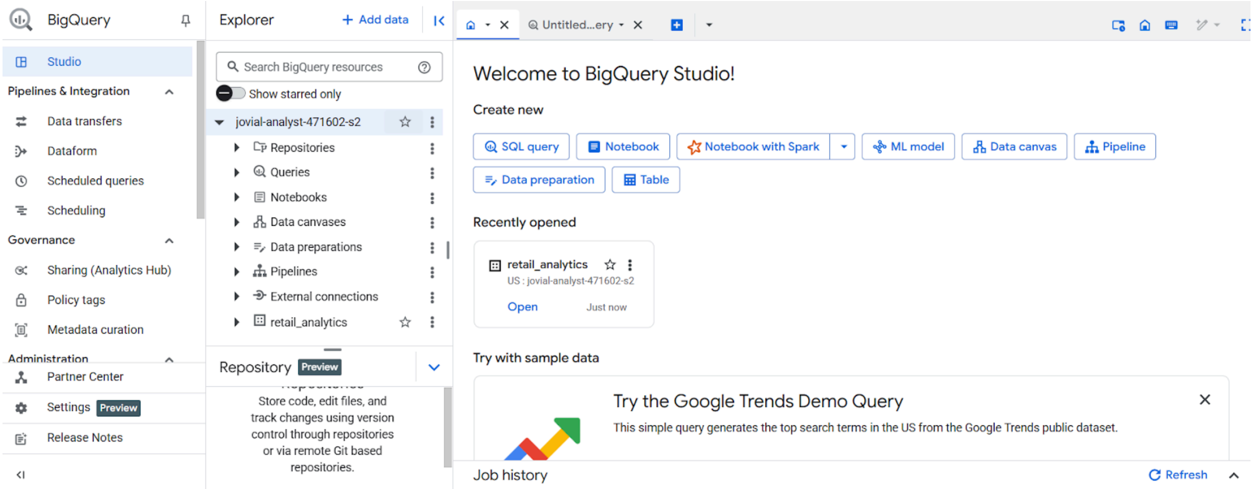
Uploading 4 files

purchases_2025-09-01.csv.csv	Complete
purchases_2025-09-02.csv.csv	Complete
purchases_2025-09-03.csv.csv	Complete
purchases_2025-09-04.csv.csv	Complete

4 files successfully uploaded

Step 3: BigQuery Setup

A BigQuery dataset, such as retail_analytics, was created to house the transformed and aggregated tables. External tables were created in BigQuery to point to the raw CSV data in GCS, allowing it to be queried directly without loading it first.



Welcome to BigQuery Studio!

Store code, edit files, and track changes using version control through repositories or via remote Git based repositories.

Try the Google Trends Demo Query

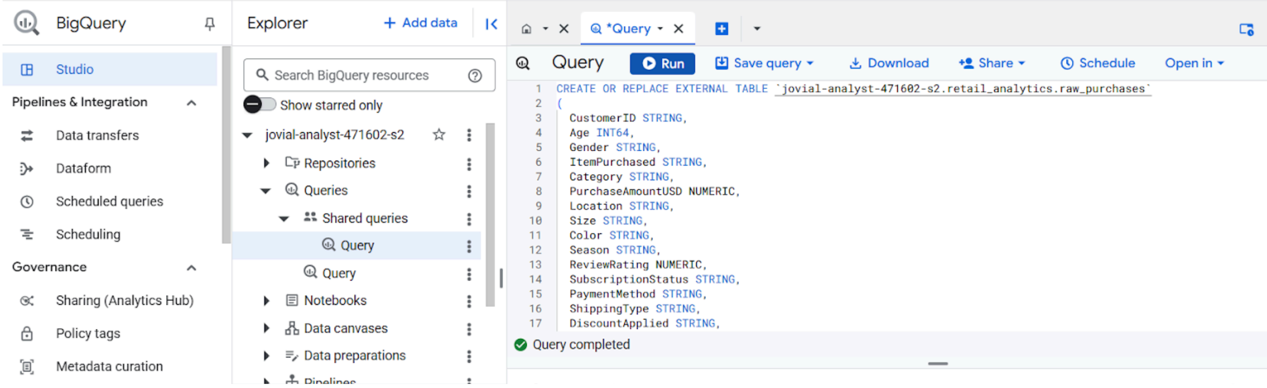
This simple query generates the top search terms in the US from the Google Trends public dataset.

Step 4: Data Transformation and Aggregation

SQL queries were used in BigQuery to transform the raw data into aggregated fact tables. Examples include:

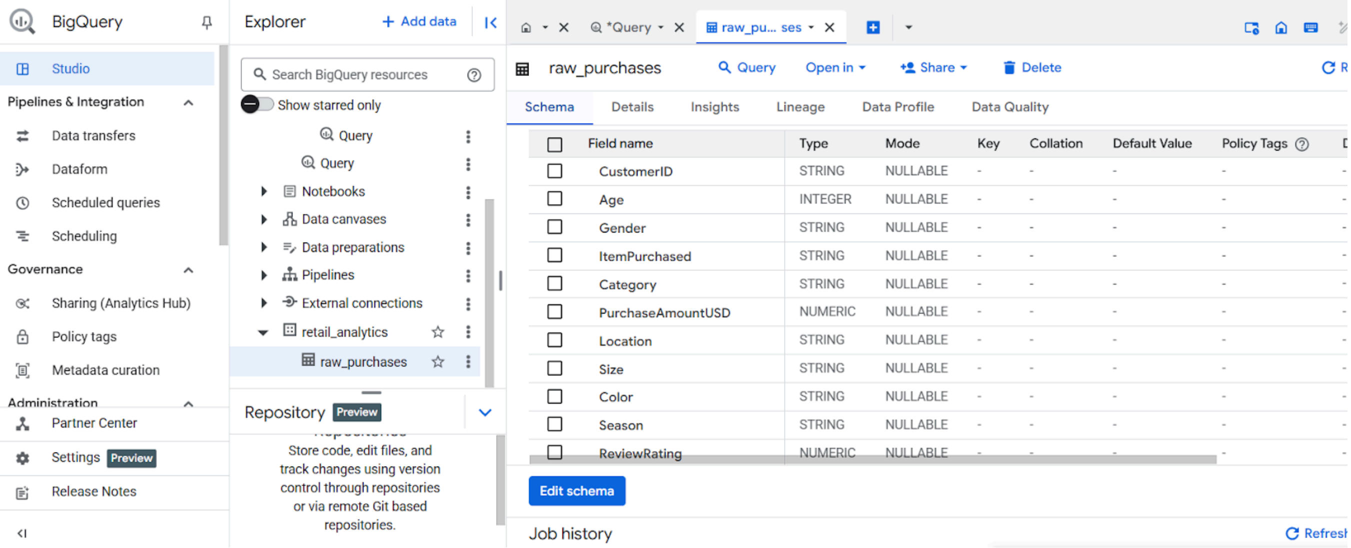
- Seasonal and Category Sales: A table named agg_sales_by_season_category was created by grouping purchases by season and category to calculate total sales and average order value.
- Demographic Sales: A table named agg_sales_by_age_gender was created to show sales trends segmented by age and gender.

The queries were scheduled to run on a set frequency (e.g., every 8 hours) using the BigQuery Data Transfer API to ensure the fact tables are regularly updated with new data.

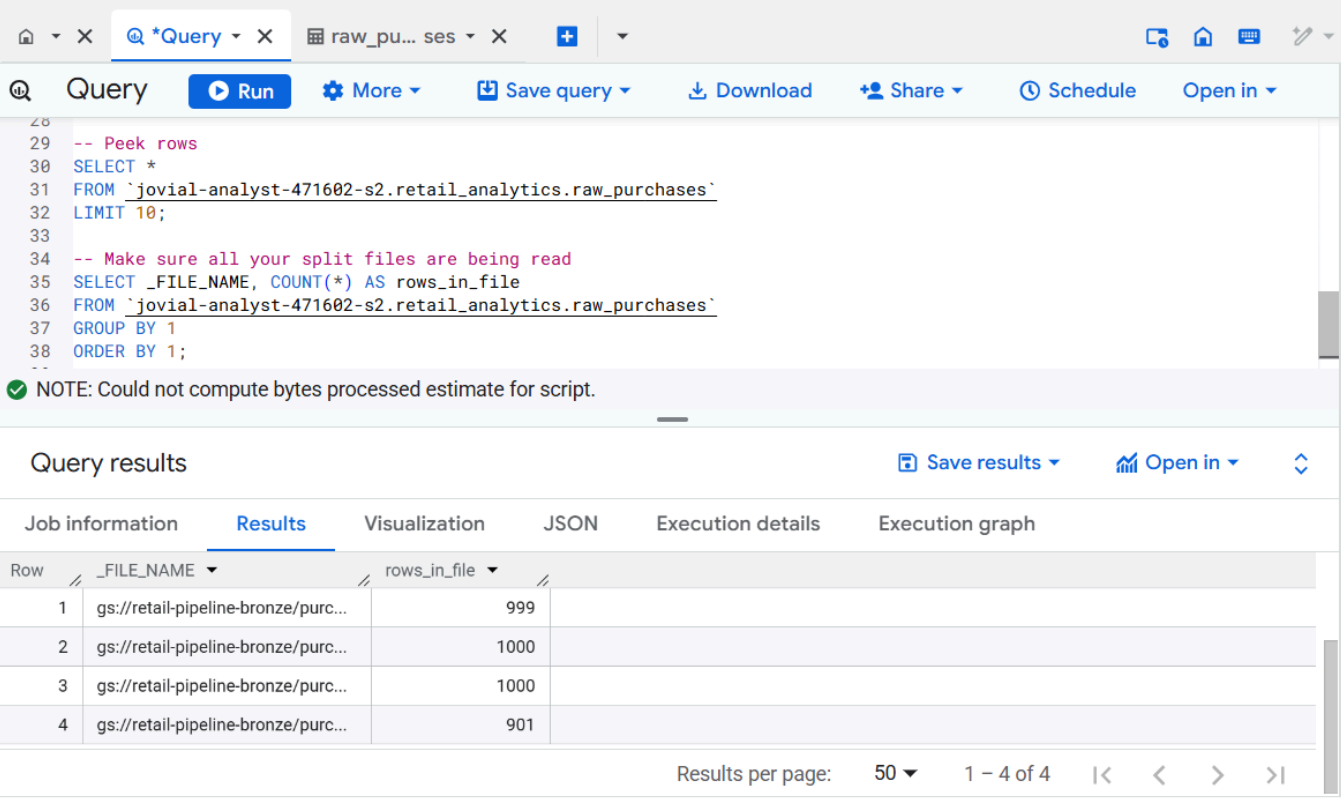


```
CREATE OR REPLACE EXTERNAL TABLE `jovial-analyst-471602-s2.retail_analytics.raw_purchases`
(
  CustomerID STRING,
  Age INT64,
  Gender STRING,
  ItemPurchased STRING,
  Category STRING,
  PurchaseAmountUSD NUMERIC,
  Location STRING,
  Size STRING,
  Color STRING,
  Season STRING,
  ReviewRating NUMERIC,
  SubscriptionStatus STRING,
  PaymentMethod STRING,
  ShippingType STRING,
  DiscountApplied STRING,
```

Query completed



□	Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	D
□	CustomerID	STRING	NULLABLE	-	-	-	-	-
□	Age	INTEGER	NULLABLE	-	-	-	-	-
□	Gender	STRING	NULLABLE	-	-	-	-	-
□	ItemPurchased	STRING	NULLABLE	-	-	-	-	-
□	Category	STRING	NULLABLE	-	-	-	-	-
□	PurchaseAmountUSD	NUMERIC	NULLABLE	-	-	-	-	-
□	Location	STRING	NULLABLE	-	-	-	-	-
□	Size	STRING	NULLABLE	-	-	-	-	-
□	Color	STRING	NULLABLE	-	-	-	-	-
□	Season	STRING	NULLABLE	-	-	-	-	-
□	ReviewRating	NUMERIC	NULLABLE	-	-	-	-	-



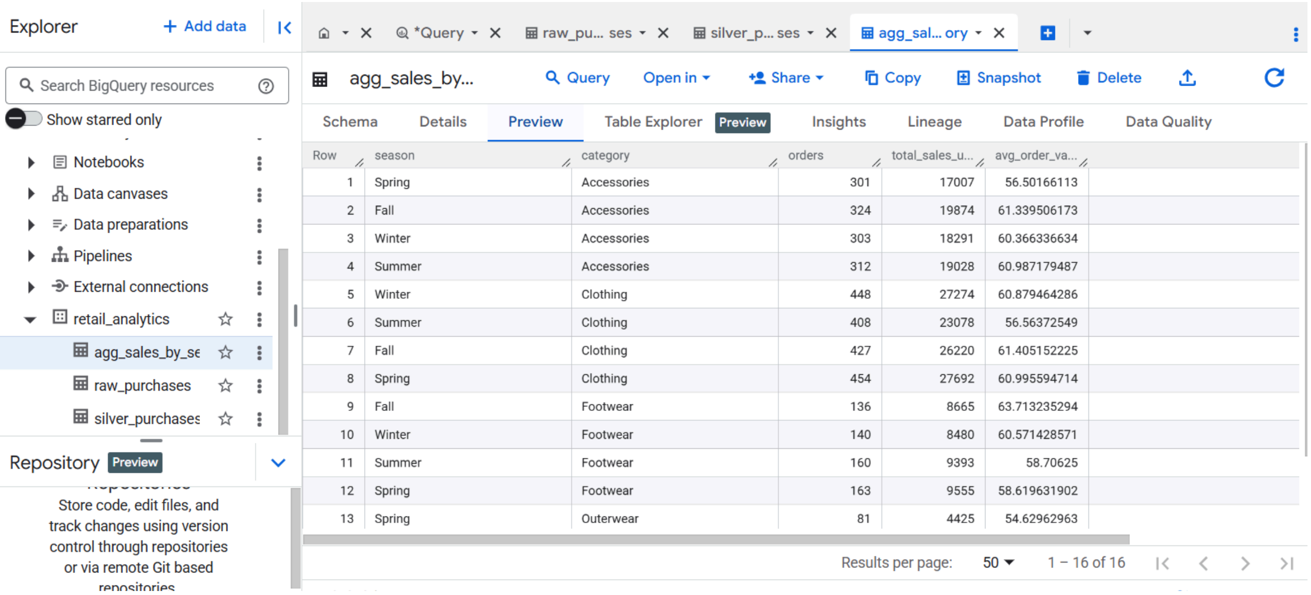
```
-- Peek rows
SELECT *
FROM `jovial-analyst-471602-s2.retail_analytics.raw_purchases`
LIMIT 10;

-- Make sure all your split files are being read
SELECT _FILE_NAME, COUNT(*) AS rows_in_file
FROM `jovial-analyst-471602-s2.retail_analytics.raw_purchases`
GROUP BY 1
ORDER BY 1;
```

NOTE: Could not compute bytes processed estimate for script.

Row	_FILE_NAME	rows_in_file
1	gs://retail-pipeline-bronze/purc...	999
2	gs://retail-pipeline-bronze/purc...	1000
3	gs://retail-pipeline-bronze/purc...	1000
4	gs://retail-pipeline-bronze/purc...	901

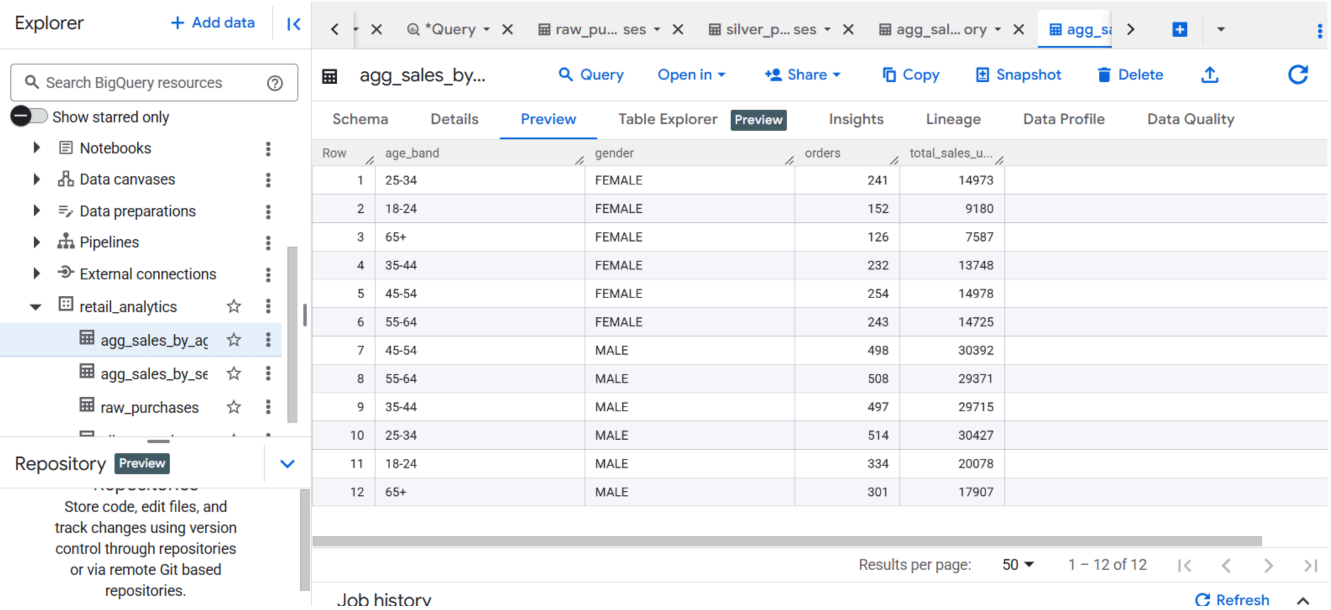
Results per page: 50 1 – 4 of 4



Store code, edit files, and track changes using version control through repositories or via remote Git based repositories

Row	season	category	orders	total_sales_u...	avg_order_va...
1	Spring	Accessories	301	17007	56.50166113
2	Fall	Accessories	324	19874	61.339506173
3	Winter	Accessories	303	18291	60.366336634
4	Summer	Accessories	312	19028	60.987179487
5	Winter	Clothing	448	27274	60.879464286
6	Summer	Clothing	408	23078	56.56372549
7	Fall	Clothing	427	26220	61.405152225
8	Spring	Clothing	454	27692	60.995594714
9	Fall	Footwear	136	8665	63.713235294
10	Winter	Footwear	140	8480	60.571428571
11	Summer	Footwear	160	9393	58.70625
12	Spring	Footwear	163	9555	58.619631902
13	Spring	Outerwear	81	4425	54.62962963

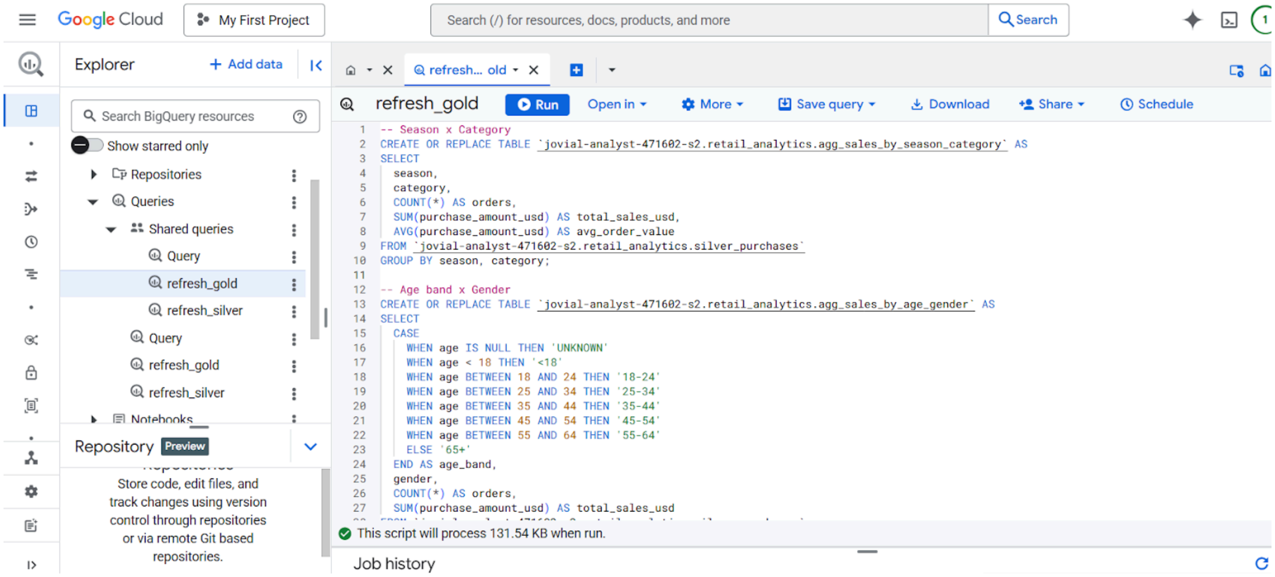
Results per page: 50 1 – 16 of 16



Store code, edit files, and track changes using version control through repositories or via remote Git based repositories.

Row	age_band	gender	orders	total_sales_u...
1	25-34	FEMALE	241	14973
2	18-24	FEMALE	152	9180
3	65+	FEMALE	126	7587
4	35-44	FEMALE	232	13748
5	45-54	FEMALE	254	14978
6	55-64	FEMALE	243	14725
7	45-54	MALE	498	30392
8	55-64	MALE	508	29371
9	35-44	MALE	497	29715
10	25-34	MALE	514	30427
11	18-24	MALE	334	20078
12	65+	MALE	301	17907

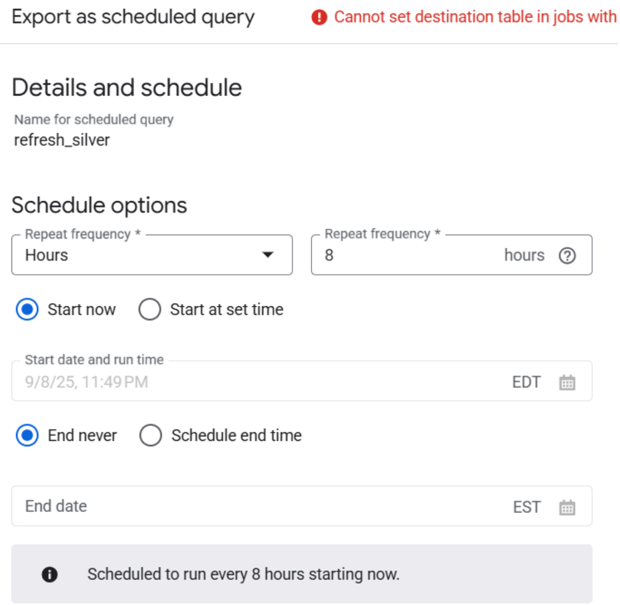
Results per page: 50 1 – 12 of 12



```
-- Season x Category
CREATE OR REPLACE TABLE `jovial-analyst-471602-s2.retail_analytics.agg_sales_by_season_category` AS
SELECT
  season,
  category,
  COUNT(*) AS orders,
  SUM(purchase_amount_usd) AS total_sales_usd,
  AVG(purchase_amount_usd) AS avg_order_value
FROM `jovial-analyst-471602-s2.retail_analytics.silver_purchases`
GROUP BY season, category;

-- Age band x Gender
CREATE OR REPLACE TABLE `jovial-analyst-471602-s2.retail_analytics.agg_sales_by_age_gender` AS
SELECT
  CASE
    WHEN age IS NULL THEN 'UNKNOWN'
    WHEN age < 18 THEN '<18'
    WHEN age BETWEEN 18 AND 24 THEN '18-24'
    WHEN age BETWEEN 25 AND 34 THEN '25-34'
    WHEN age BETWEEN 35 AND 44 THEN '35-44'
    WHEN age BETWEEN 45 AND 54 THEN '45-54'
    WHEN age BETWEEN 55 AND 64 THEN '55-64'
    ELSE '65+'
  END AS age_band,
  gender,
  COUNT(*) AS orders,
  SUM(purchase_amount_usd) AS total_sales_usd
```

This script will process 131.54 KB when run.



Export as scheduled query

Cannot set destination table in jobs with

Details and schedule

Name for scheduled query
refresh_silver

Schedule options

Repeat frequency *: Hours

Repeat frequency *: 8 hours

● Start now ○ Start at set time

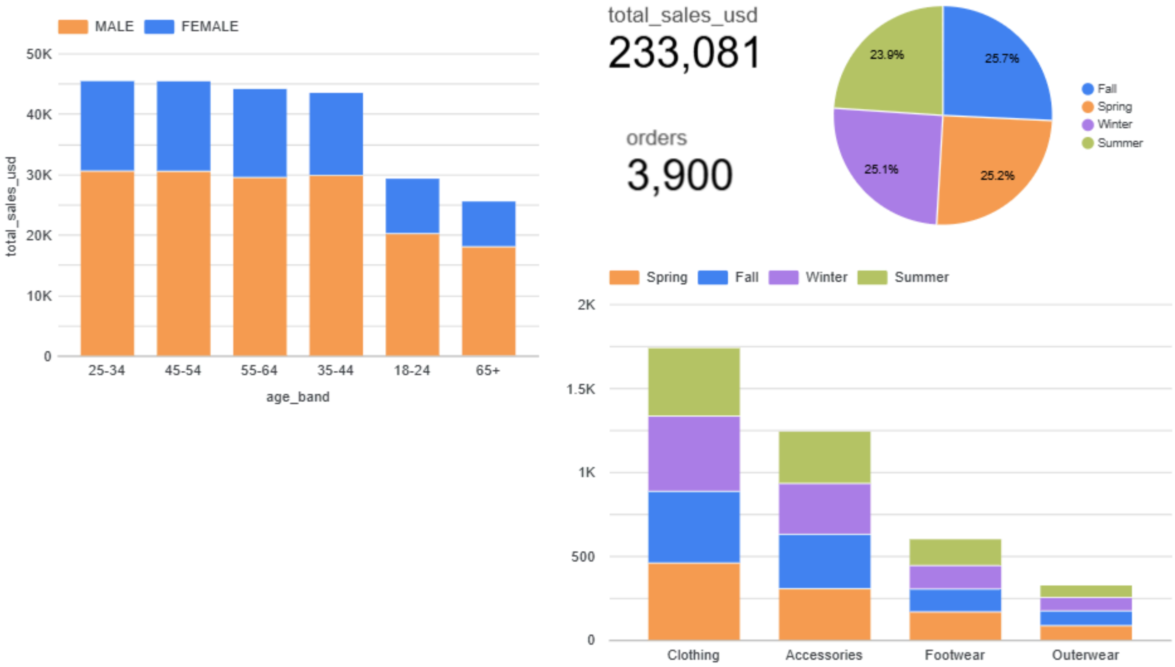
Start date and run time: 9/8/25, 11:49 PM EDT

● End never ○ Schedule end time

End date EST

Scheduled to run every 8 hours starting now.

Step 5. Visualization & Reporting
Connect the aggregated BigQuery tables to a Looker Studio dashboard. This dashboard provides a visual representation of the processed data, allowing for easy analysis of sales by demographics, season, and product category.



total_sales_usd
233,081

orders
3,900