XCCELERATE

BIG QUERY

Agenda

Overview

- What is BigQuery and its features
- Pricing
- Comparison to other GCP technologies
- BigQuery Data Model

  Datasets, tables and views

Demos

- Setting Up and Accessing BigQuery Web UI
- Creating Datasets and Tables in BigQuery
- Loading Local Data to BigQuery
- Accessing BigQuery from Jupyter notebooks

What is Big Query

Query as a Service

- Runs on Google's cloud infrastructure

Massively Scalable

- Process trillions of records in seconds
- Terabytes in seconds, petabytes in minutes

Queries are based on SQL

BigQuery as an analytical data warehouse

- Serverless operations
- Pay for storage and queries
- Visualizing data with Data Studio

Varied requirements for analysis

Order Management Support

John is responsible for tracking and delivering orders on time

Revenue Analyst

Anna is responsible for tracking and monitoring revenues

Transactional and Analytical Processing

Transactional Processing

Analytical Processing

Transactional and Analytical Processing

Transactional Processing

Analyzes individual entries

Access to recent data, from the last few hours or days

Updates data

Fast real-time access

Usually a single data source

Analytical Processing

Analyzes large batches of data

Access to older data going back months, or even years

Mostly reads data

Long running jobs

Multiple data sources

Transactional and Analytical Processing

Small Data

Both these objectives could be achieved using the same database system

Small Data

Single machine with backup

Structured, well-defined data

Can access individual records or the entire dataset

No replication, updated data available instantaneously

Different tables store data from different sources

Transactional and Analytical Processing

Big Data

Very hard to meet all requirements with
the same database system

Big Data

Data distributed on a cluster with multiple machines

Semi-structured or unstructured data

Data replicated, propagation of updates take time

Different sources may have different formats

3 Vs of Big Data

Volume: Amount of data

Variety: Number and type of sources

Velocity: Batch and streaming

Transactional and Analytical Processing

Transactional Processing

Traditional
RDBMS

Analytical Processing

Data Warehouse

BigQuery is a Data Warehouse that is fullly managed like a RDBMS

BigQuery for Analytics and Data Warehousing

BigQuery vs. Traditional Data Warehouses

BigQuery	Traditional Data Warehouse
Complex analytical queries	Complex analytical queries
Scales to Petabytes	Scales to Petabytes
Both reads and updates	Mostly reads
Fast real-time or batch access	Long running jobs
Multiple data sources	Multiple data sources
Streaming as well as batch	Often more focus on batch

BigQuery vs. Traditional RDBMS

BigQuery	Traditional RDBMS
Access using SQL	Access using SQL
Scales to Petabytes	Usually top out at Terabytes
No ACID or transaction support	Strong emphasis on ACID and transaction support
Serverless	Classic example of server
No indices, no provisioning	Heavy administrative overhead

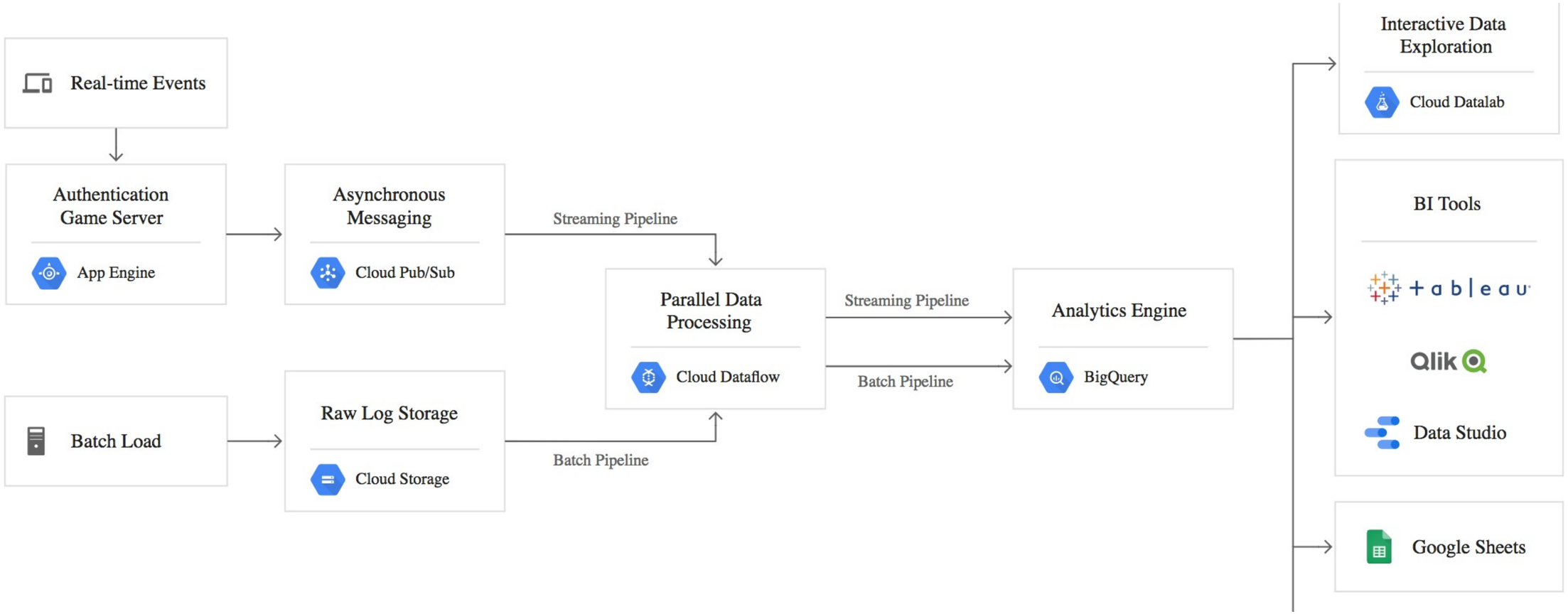
BigQuery Features

Serverless: No cluster, no provisioning

Autoscaling

Automatic high availability

Solution architecture - GCP



Support for the 3Vs

Volume: Scales to petabytes

Variety: Federated data sources

- Cloud storage
- BigTable
- Google Drive spreadsheets

Velocity

- Streaming ingestion
- Real-time queries

SQL Support

Standard SQL

- ANSI:2011 compliant
- Extensions for nested/repeated fields

Legacy SQL

Data Locality

Some support for specific locations

US, Japan, EU

BigQuery ML

Currently in Beta

Simple model building and use

All in SQL from within BigQuery

Cloud Storage options

Product	Type	Similar Products
Cloud Storage	Object Storage	S3, HDFS
Cloud SQL	RDBMS	MySQL, PostgreSQL
Cloud Spanner	RDBMS	Oracle
Cloud BigTable	NoSQL – Columnar	Hbase, Cassandra
Cloud DataStore	NoSQL – Document	MongoDB, ElasticSearch
Cloud MemoryStore	NoSQL – Key Value	Redis
BigQuery	Data Warehouse	Hive, HBase

GCP Storage Options

Choosing BigQuery

BigQuery

GCP data warehousing solution

Serverless - no provisioning needed

Autoscaling - no control over compute

No operations needed

AWS Redshift

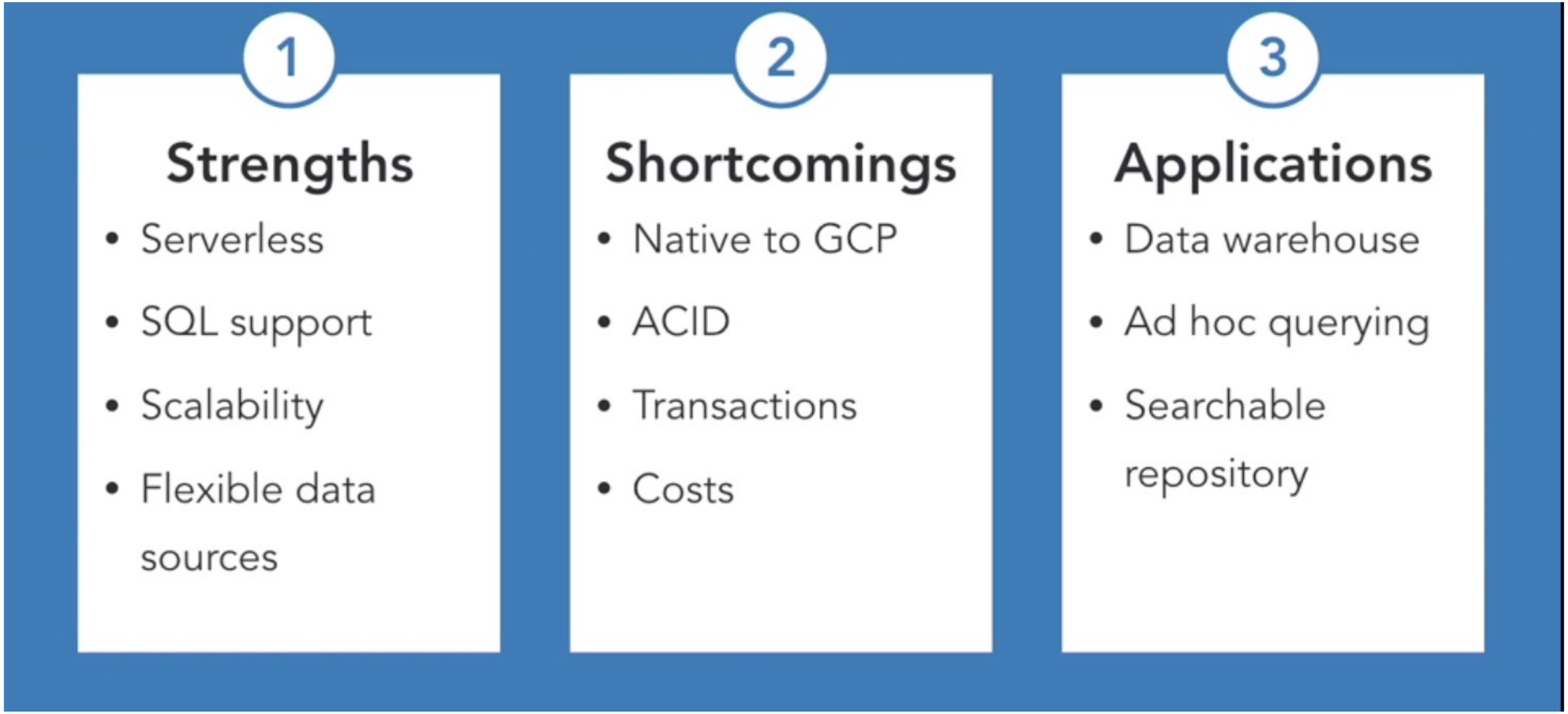
AWS data warehousing solution

Provisioning needed - more like BigTable

Scale up by adding nodes

Some operations and cluster maintenance indeed needed

Big Query - Overview



BigQuery Pricing

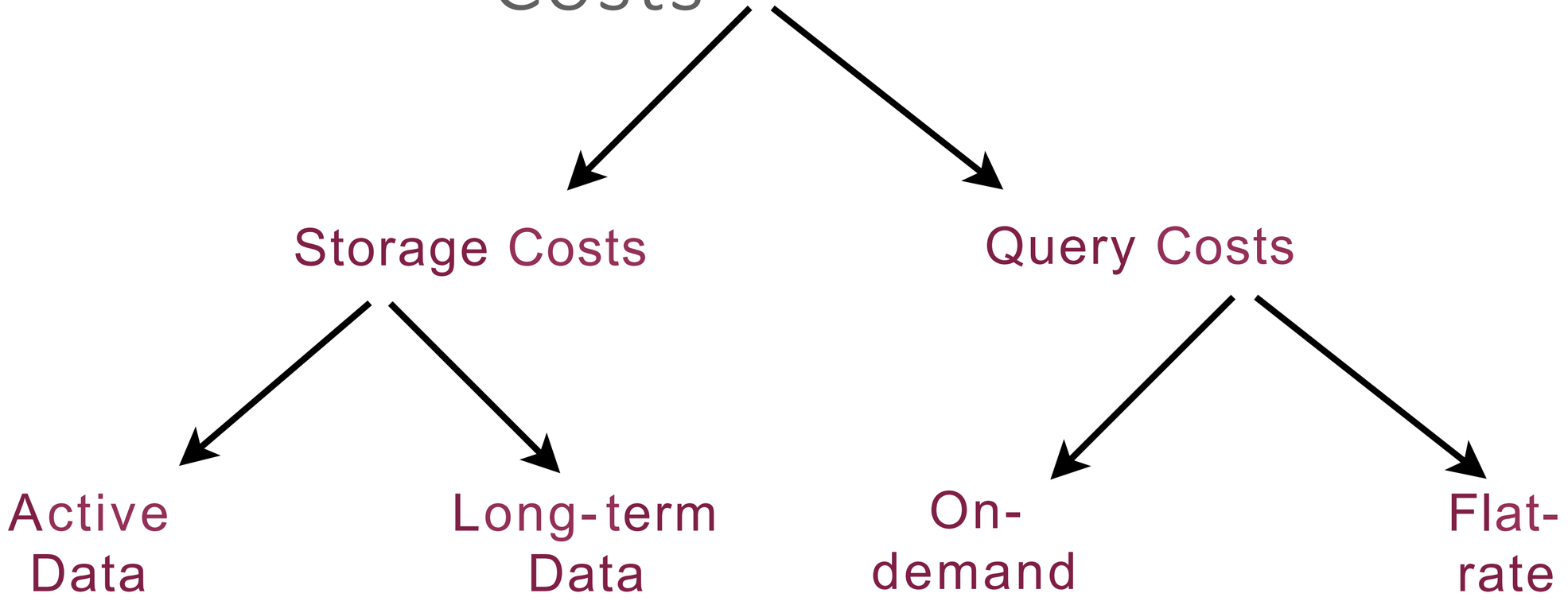
BigQuery
Pricing

Storage costs

Query costs

Free operations

BigQuery Costs



Storage Costs

Active

Data in tables modified in last 90 days

Currently approximately 2 cents/GB/month

First 10 GB is free

When table is edited, pricing reverts to active

Long-term

Data in tables not modified in last 90 days

About 50% lower; currently about 1cent/GB/month

First 10 GB is free

When table is not edited, pricing automatically drops to long-term

Query Costs

On-demand	Flat-rate
Based solely on usage	Predictable, fixed monthly costs
$5 per TB/month; First 1TB/month free	$40,000/month for 2000 slots*; $10,000 per 500 additional slots

BigQuery Slot

Unit of computational capacity required to execute SQL queries. BigQuery automatically calculates how many slots are required by each query, depending on query size and complexity.

Free Operations

Loading data

- (Streaming inserts not free)

Copying data

Exporting data

Deleting datasets

Deleting tables, views, partitions

Metadata operations

Minimizing Costs

Query only columns you need

Use table preview to explore data

Don’t run queries just to explore

Calculate query price before running

Summary

BigQuery is a cloud data warehouse

Standard SQL and ODBC/JDBC drivers

No clusters, no servers

“No-ops” - not even indices

Autoscaling right to petabytes

Streaming and real-time analytics

BigQuery Data Model

BigQuery Data Model

Datasets

Top-level container used to organize and control access to tables and views. A table or view must belong to a dataset.

Tables

Contains individual records organized in rows. Each record is composed of columns (also called fields)

Views

Virtual table defined by a SQL query. Whenever a user queries the view, the underlying view-query is executed

Data Location

Geographic location can be specified at create-time

After creation, location becomes immutable

US, EU, Asia (Japan)

Dataset Features

Access control at dataset level

Labels can be assigned

Expiration time for new tables

Public Datasets

Stored in BigQuery

Google pays for storage

Available for general usage via a project

User pays for queries

Advantages of Views

Reduce query complexity

Restrict access to data

Construct different logical tables from the same physical table

XCCELERATE

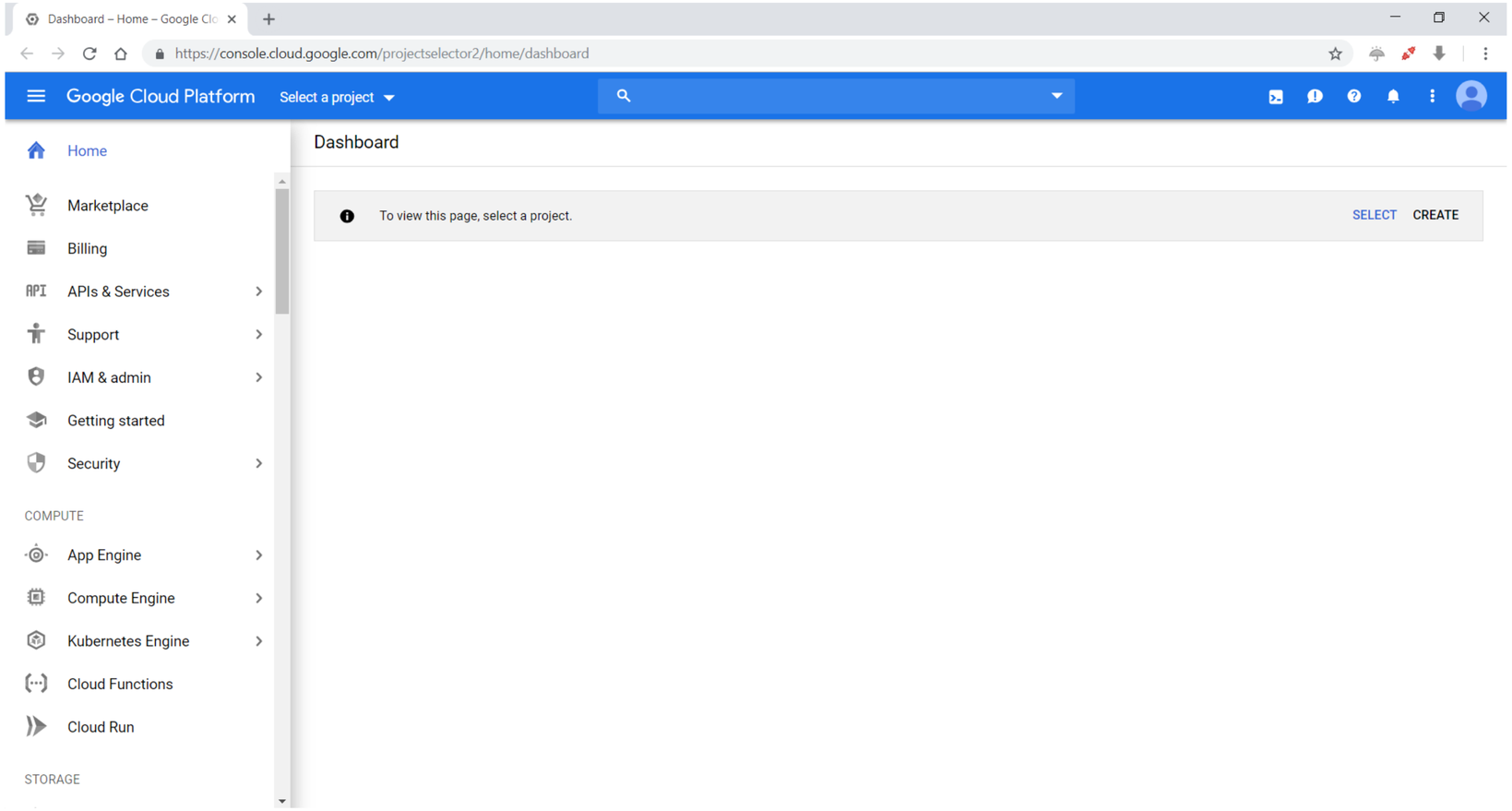
DEMOS

XCCELERATE

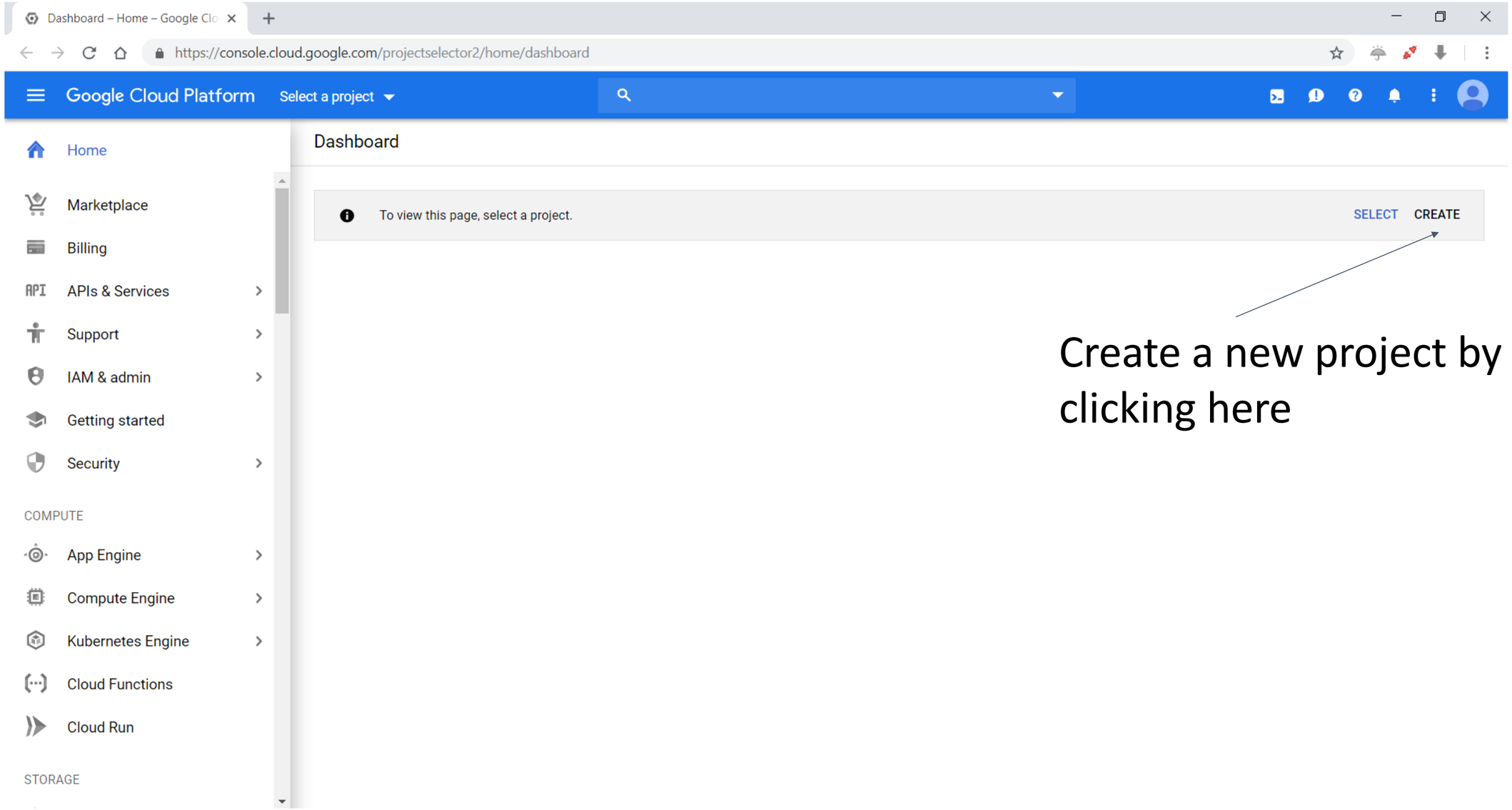
SETTING UP AND ACCESSING BIGQUERY WEB UI

Creating a new project

Go to this link:
https://console.cloud.google.com/projectselector2/home/dashboard



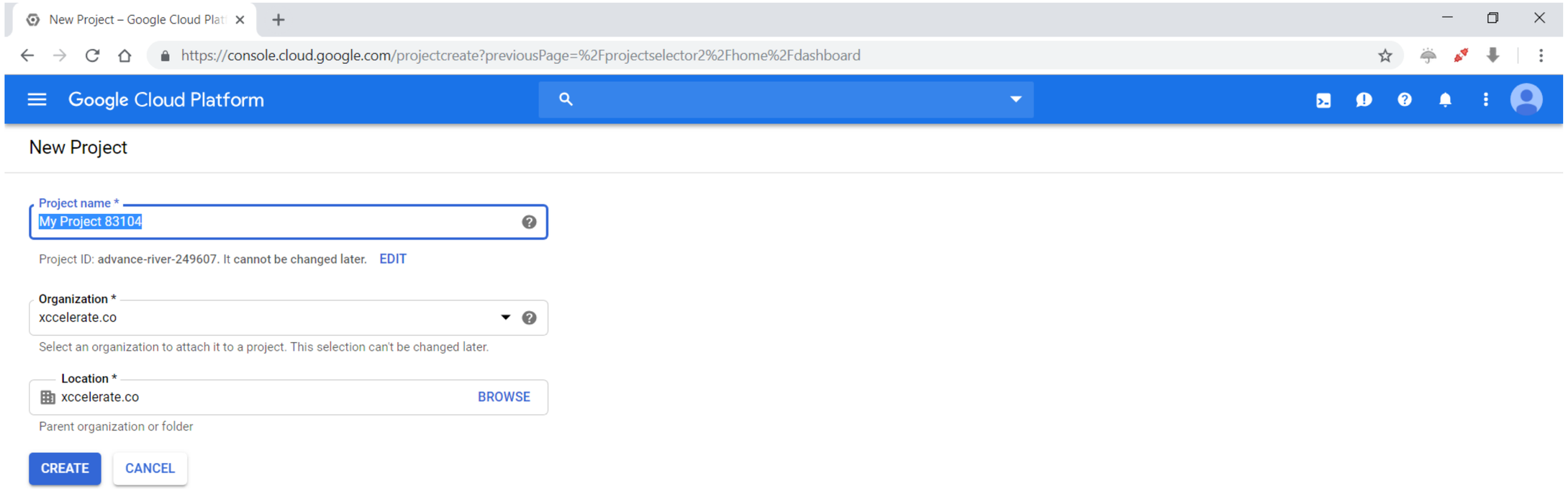
Creating a new project



Create a new project by clicking here

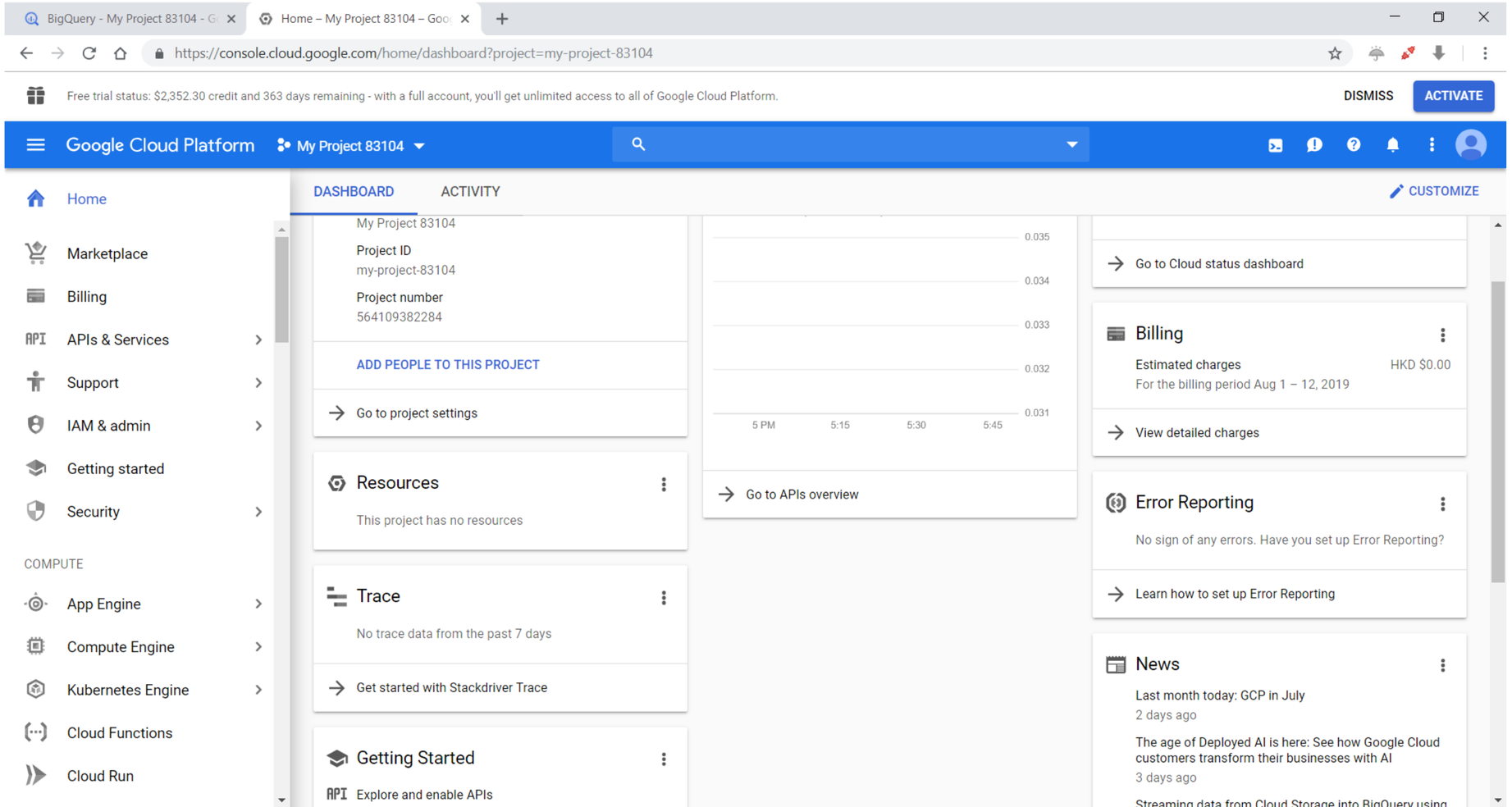
Creating a new project

Type in a project name and press Create



Creating a new project

It will redirect you to this window, the Google Cloud Platform (GCP) Console

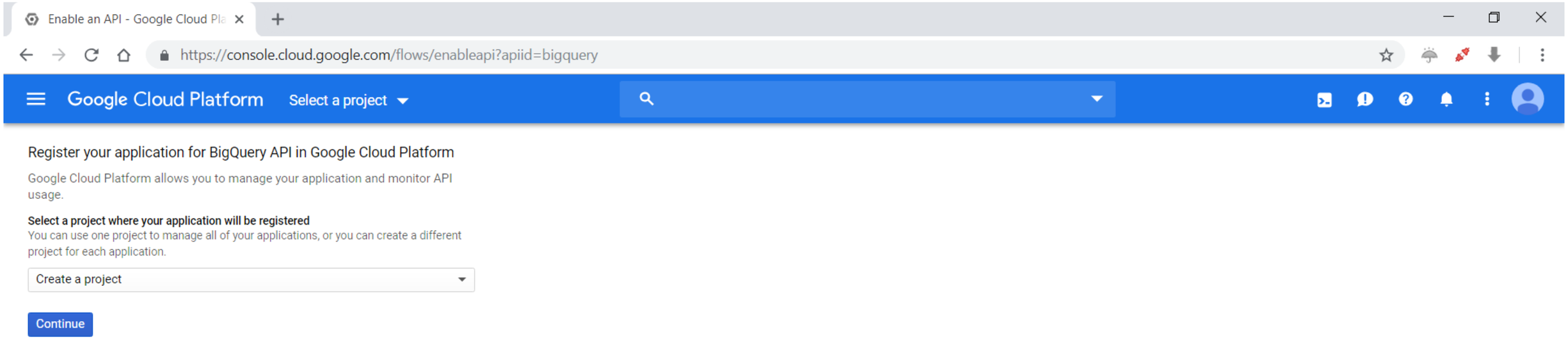


Enable billing

Next, we need to enable the billing. The steps are outlined here, under *Enable billing for a project*:
https://cloud.google.com/billing/docs/how-to/manage-billing-account

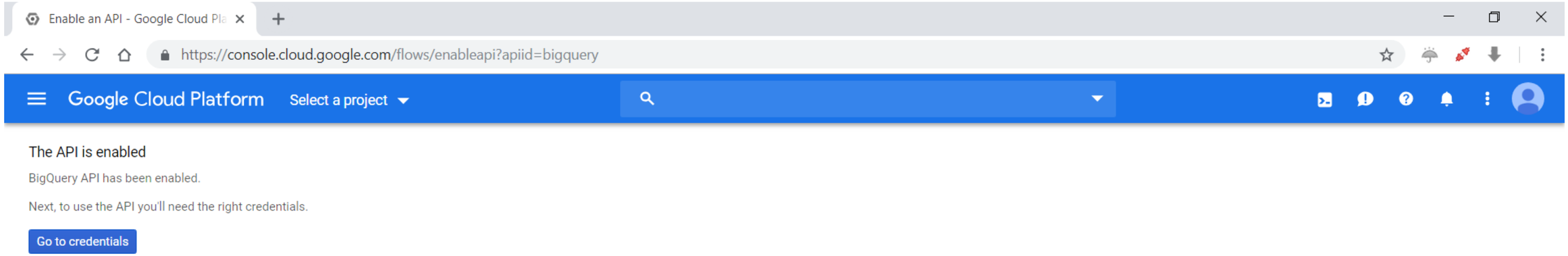
Enable API

Now, we need to enable the BigQuery API for our project. Go this link: https://console.cloud.google.com/flows/enableapi?apiid=bigquery and the window below will appear. Choose your created project and press Continue



Enable API

You should see this screen if enabling is successful:



Accessing BigQuery web UI

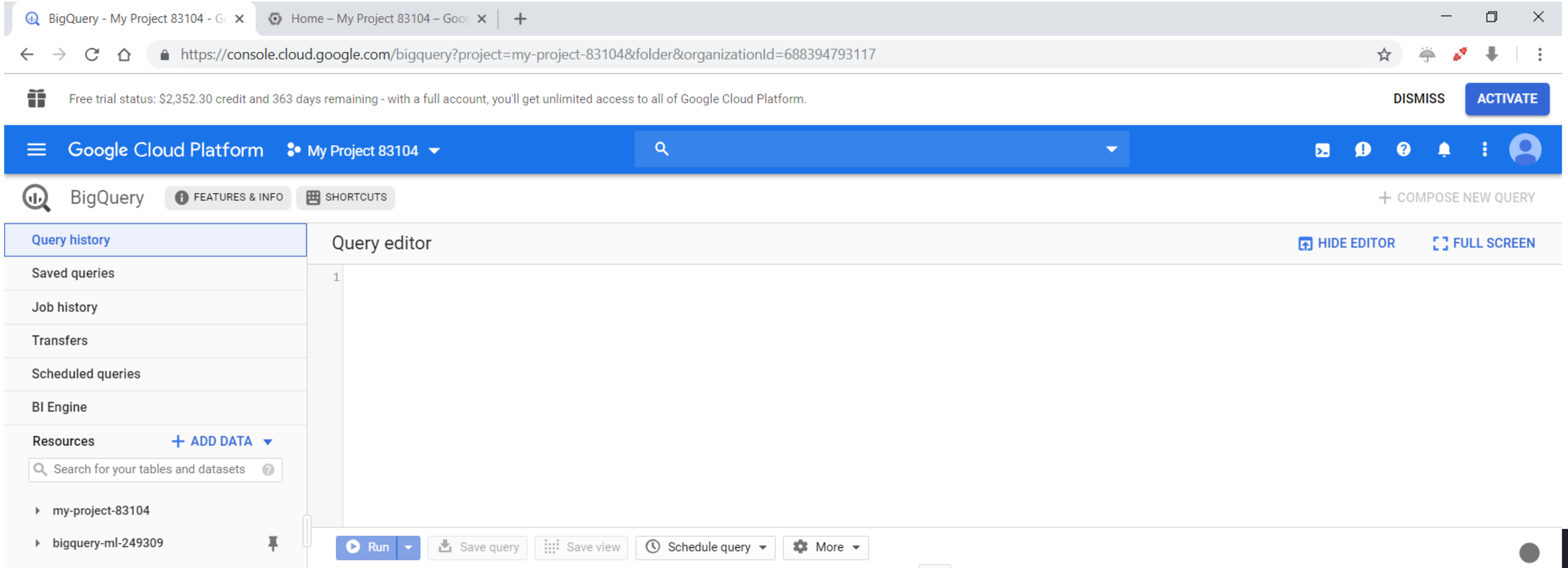
Access the BigQuery web UI through this link:
https://console.cloud.google.com/bigquery

XCCELERATE

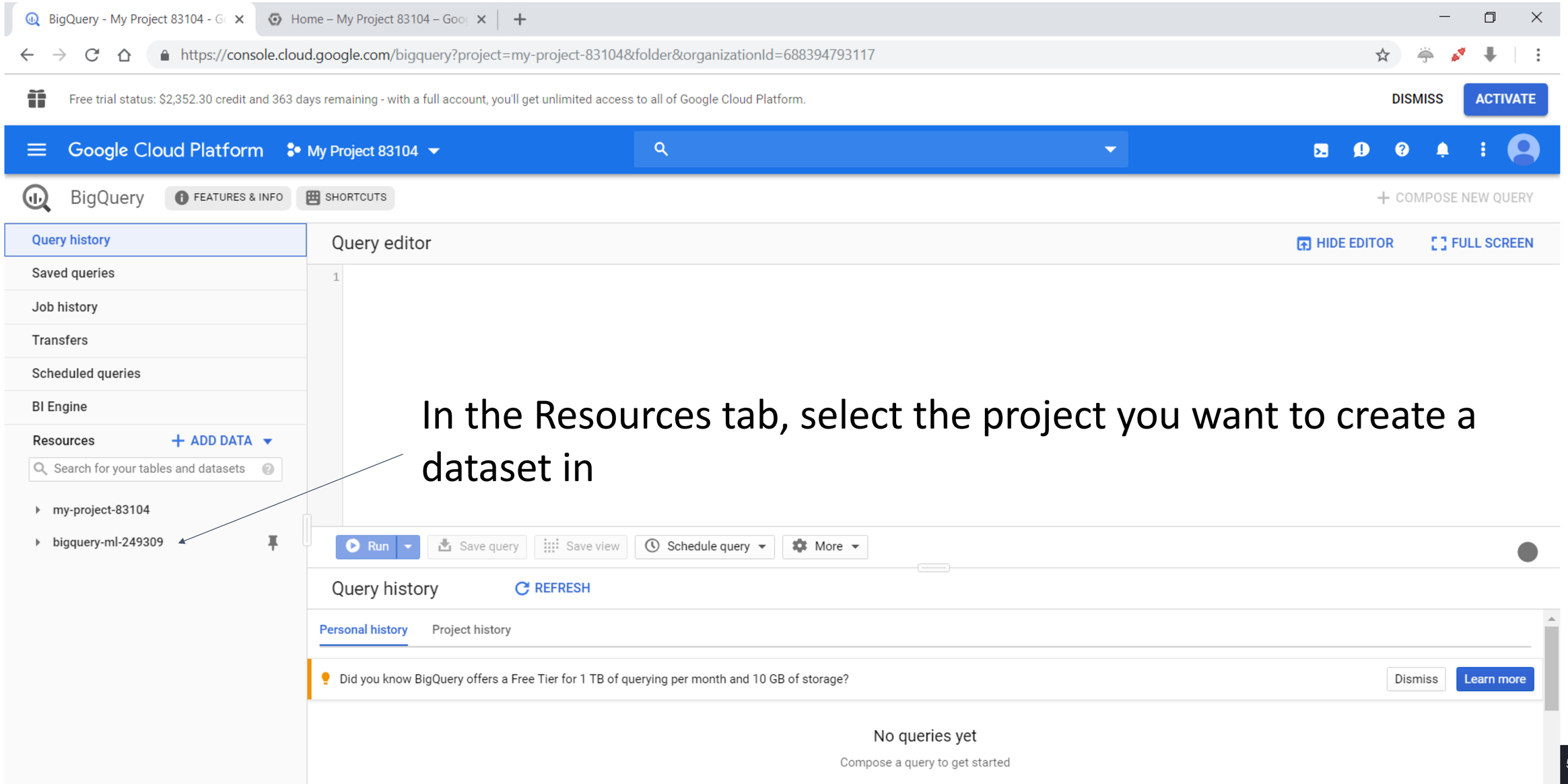
CREATING DATASETS AND TABLES IN BIGQUERY

Accessing BigQuery web UI

Access the BigQuery web UI through this link:
https://console.cloud.google.com/bigquery

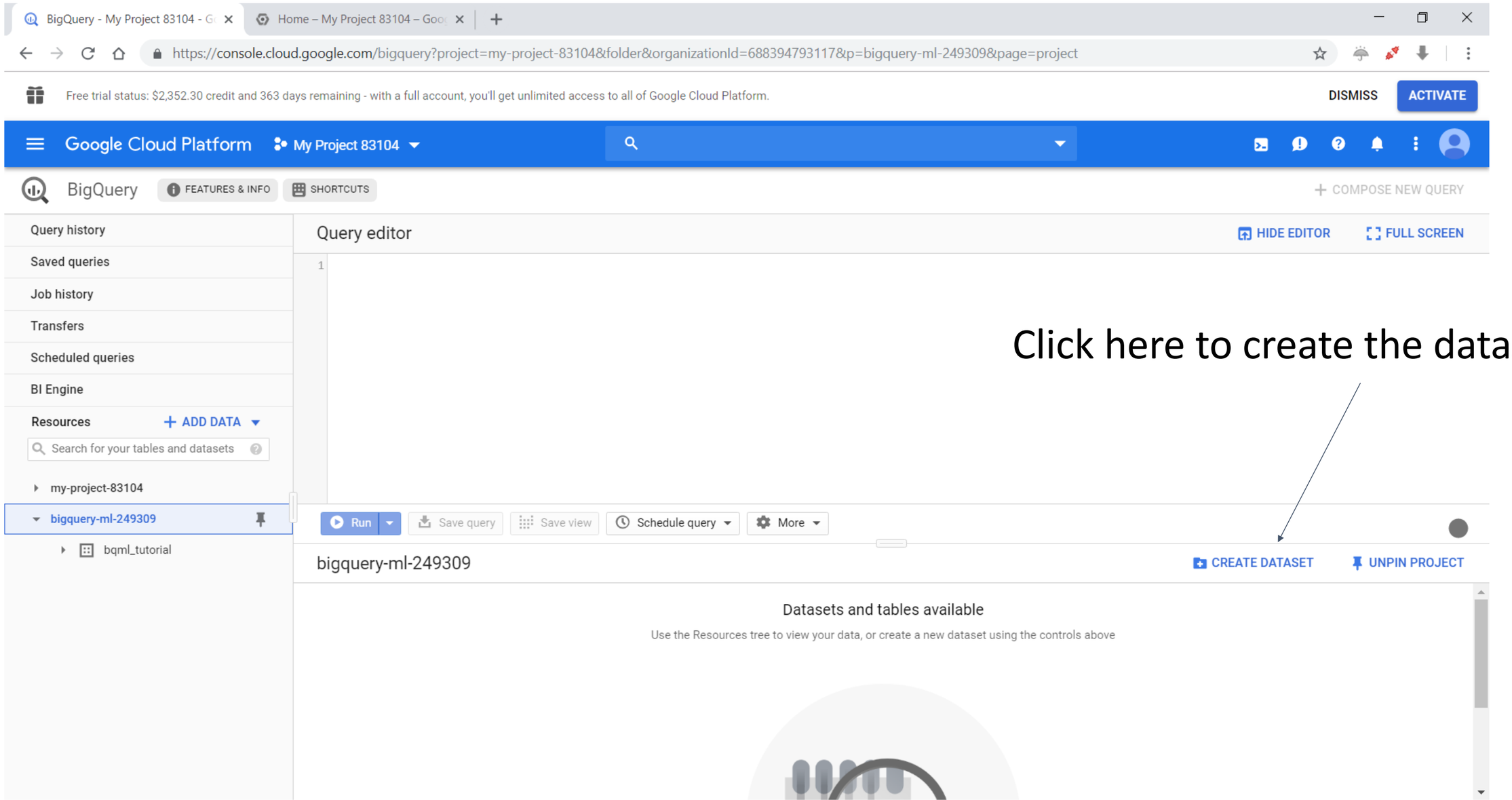


Resources



In the Resources tab, select the project you want to create a dataset in

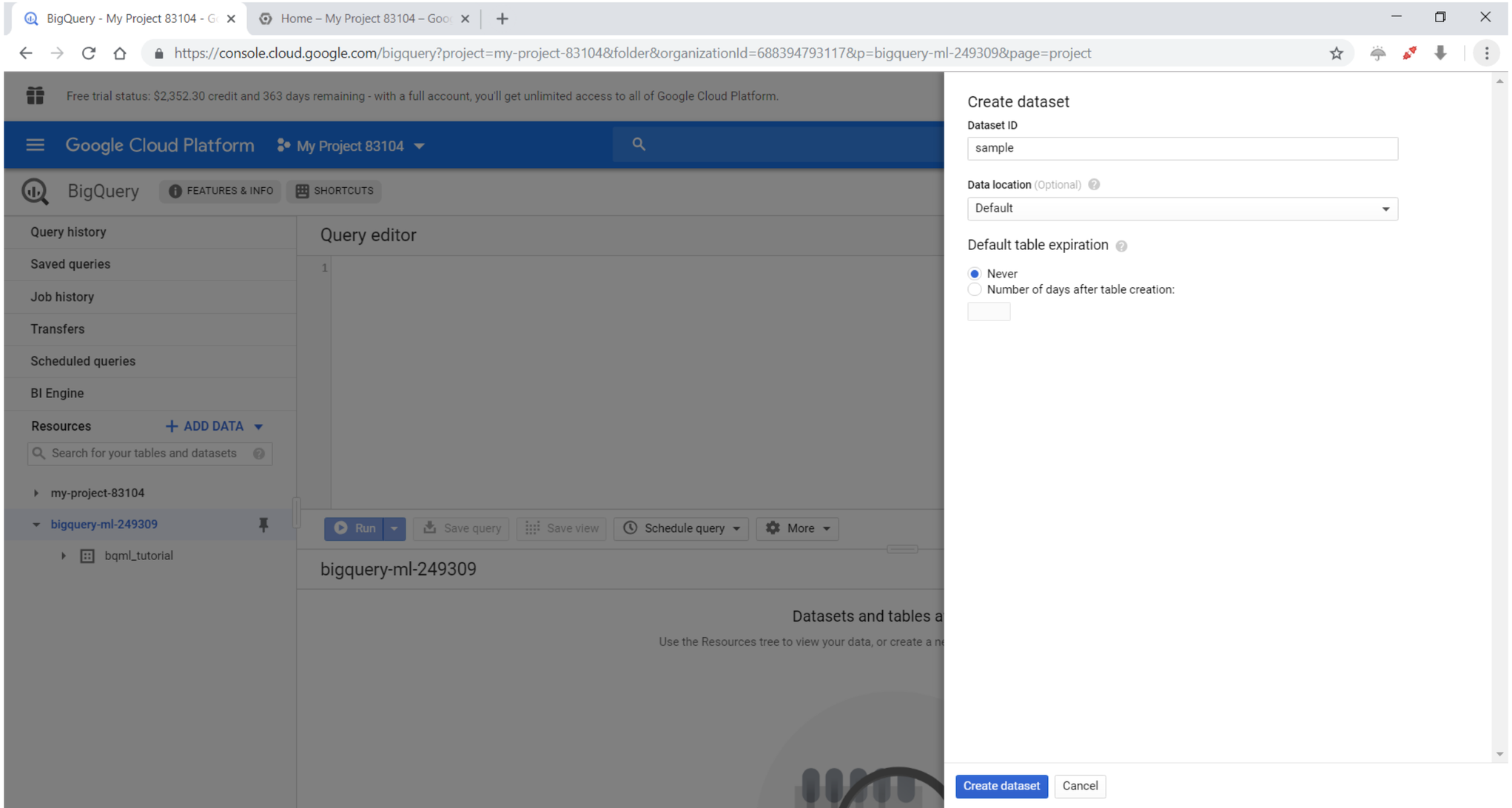
Creating a new dataset



Click here to create the dataset

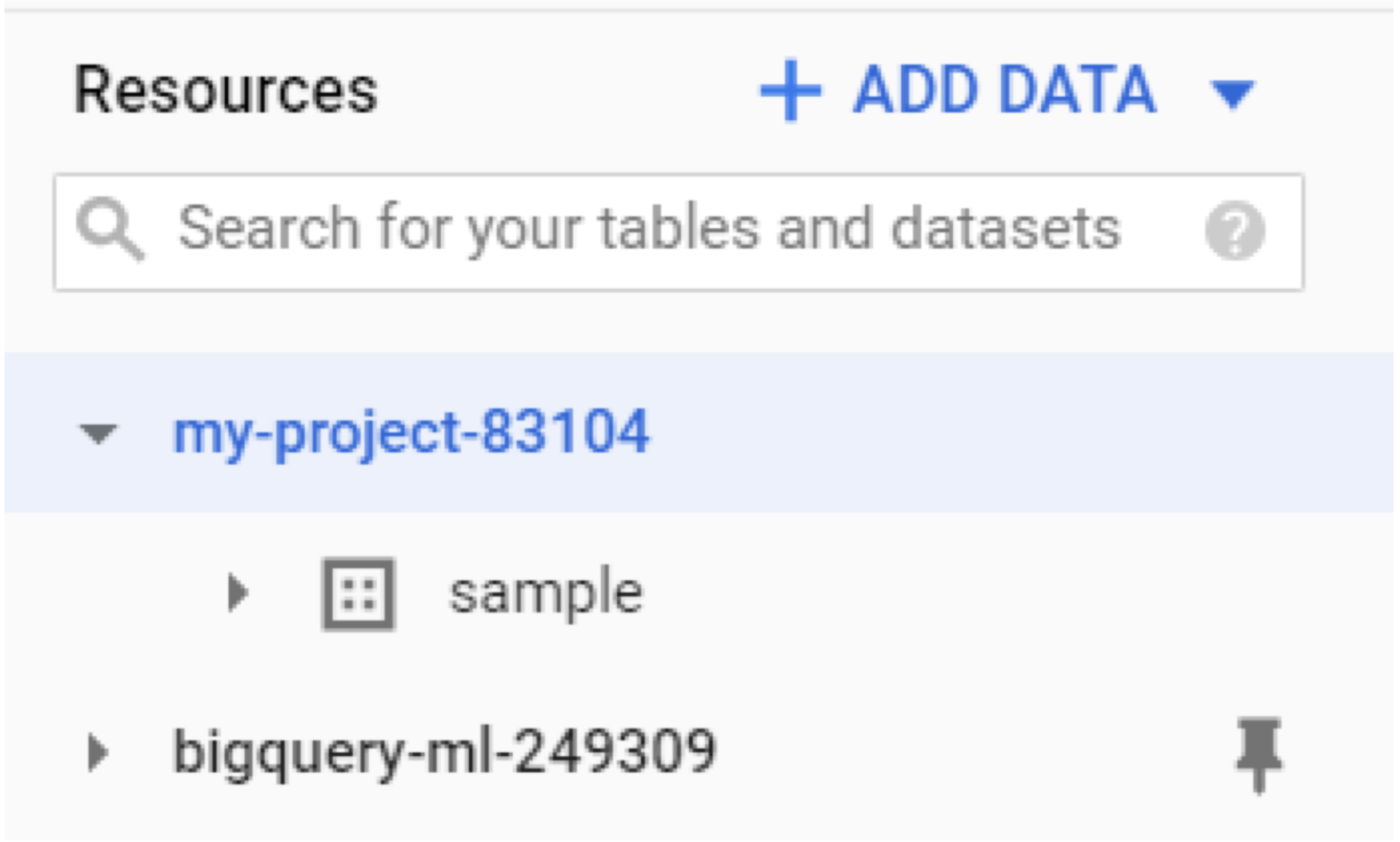
Creating a new dataset

Input dataset ID and press Create dataset



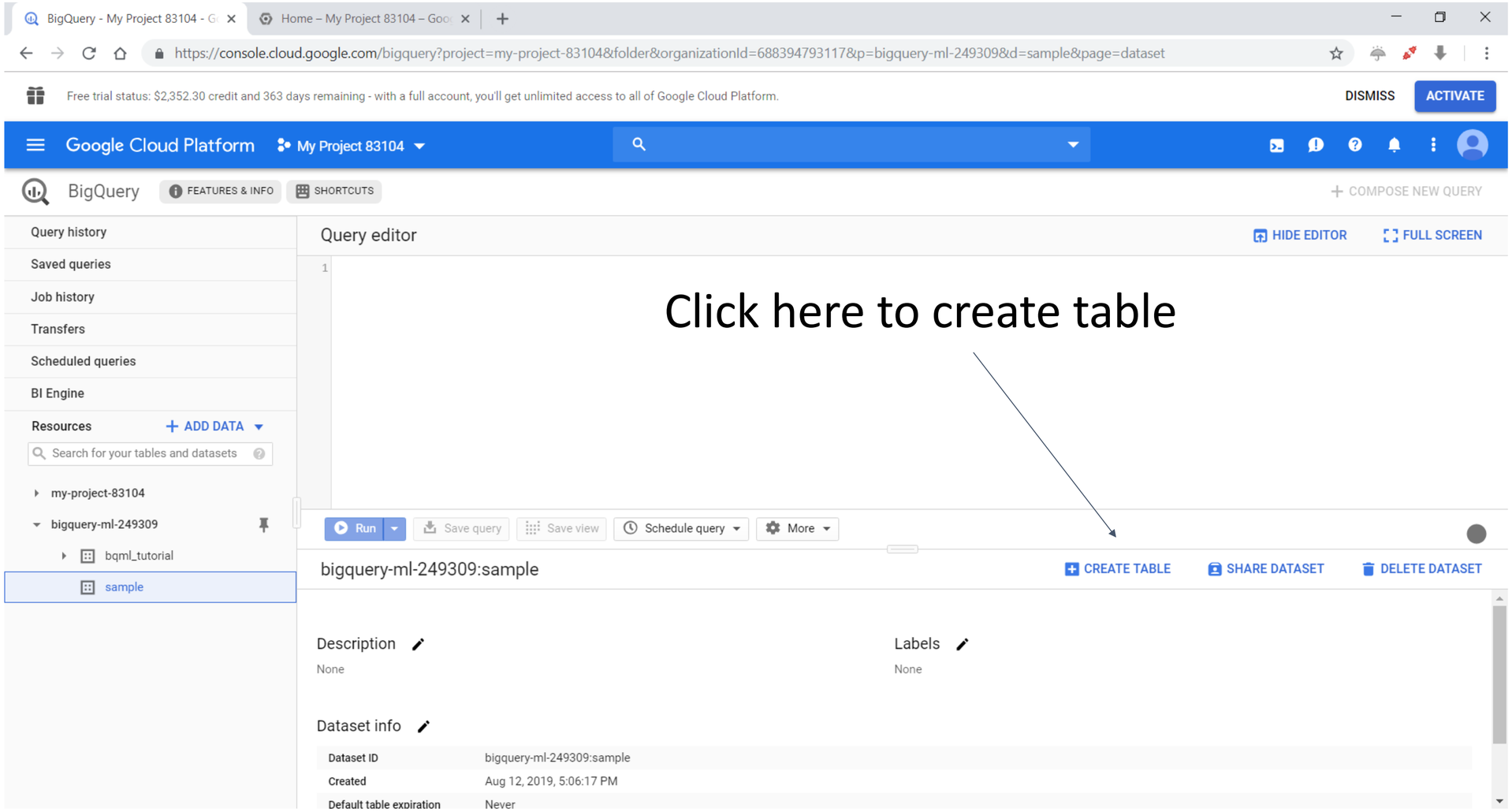
Creating a new dataset

You should see your newly-created dataset in Resources under your project:



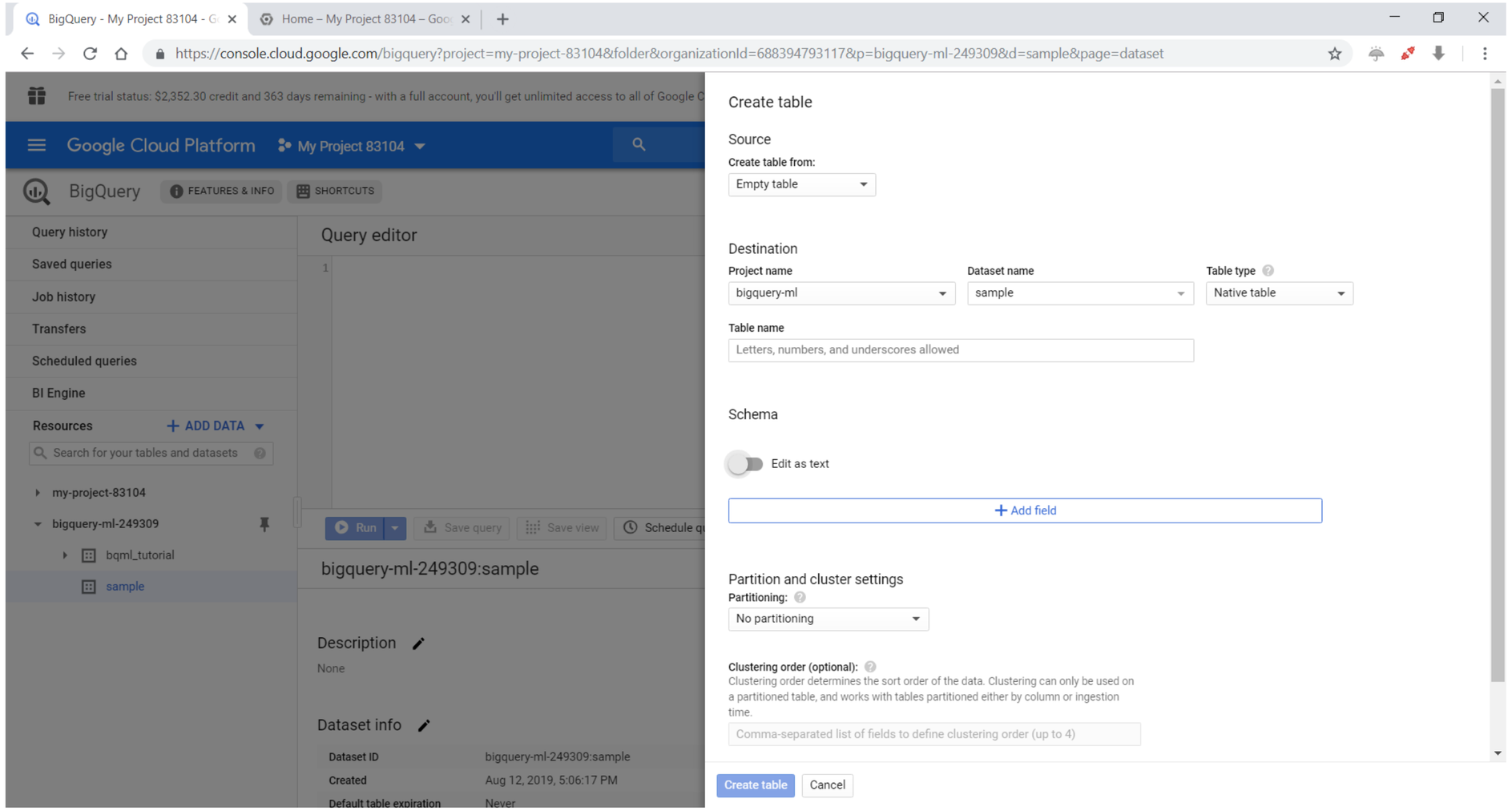
Creating a new table

Click your newly-created dataset and there should be a Create table option for it



Creating a new table

This will take you to the table creation window where you will need to specify different settings for your table. Once you're done, click Create table

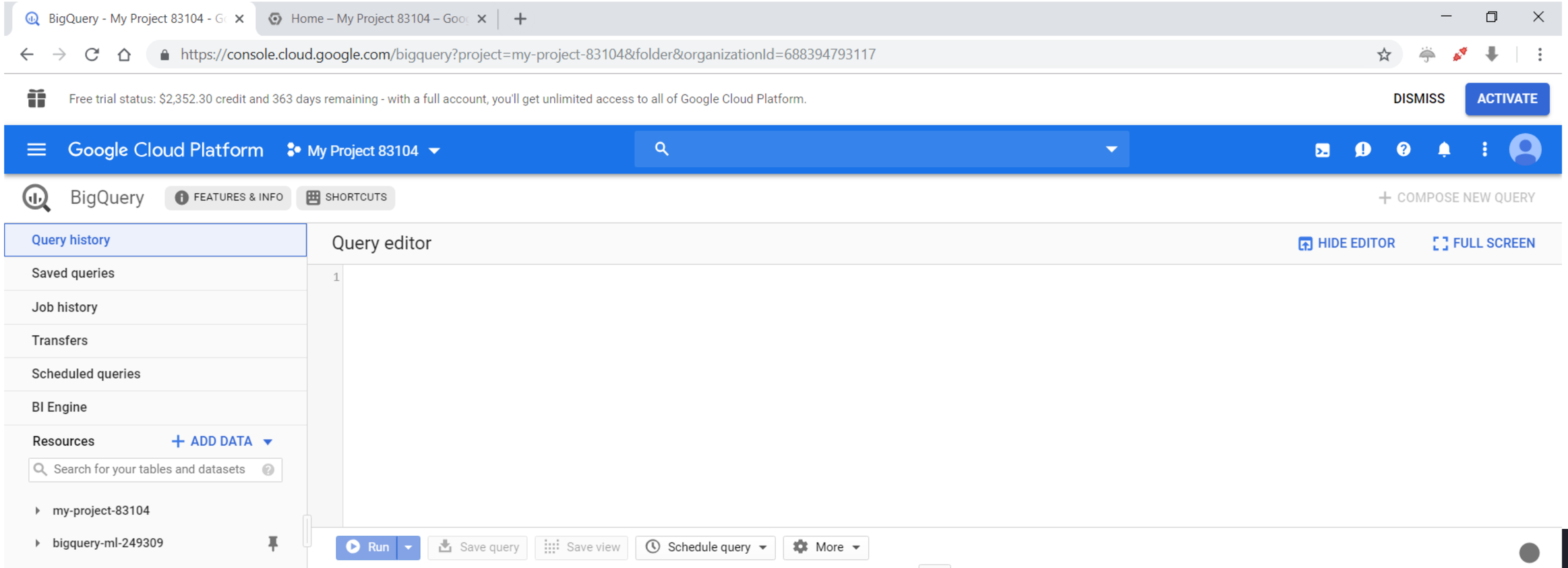


XCCELERATE

LOADING LOCAL DATA TO BIGQUERY

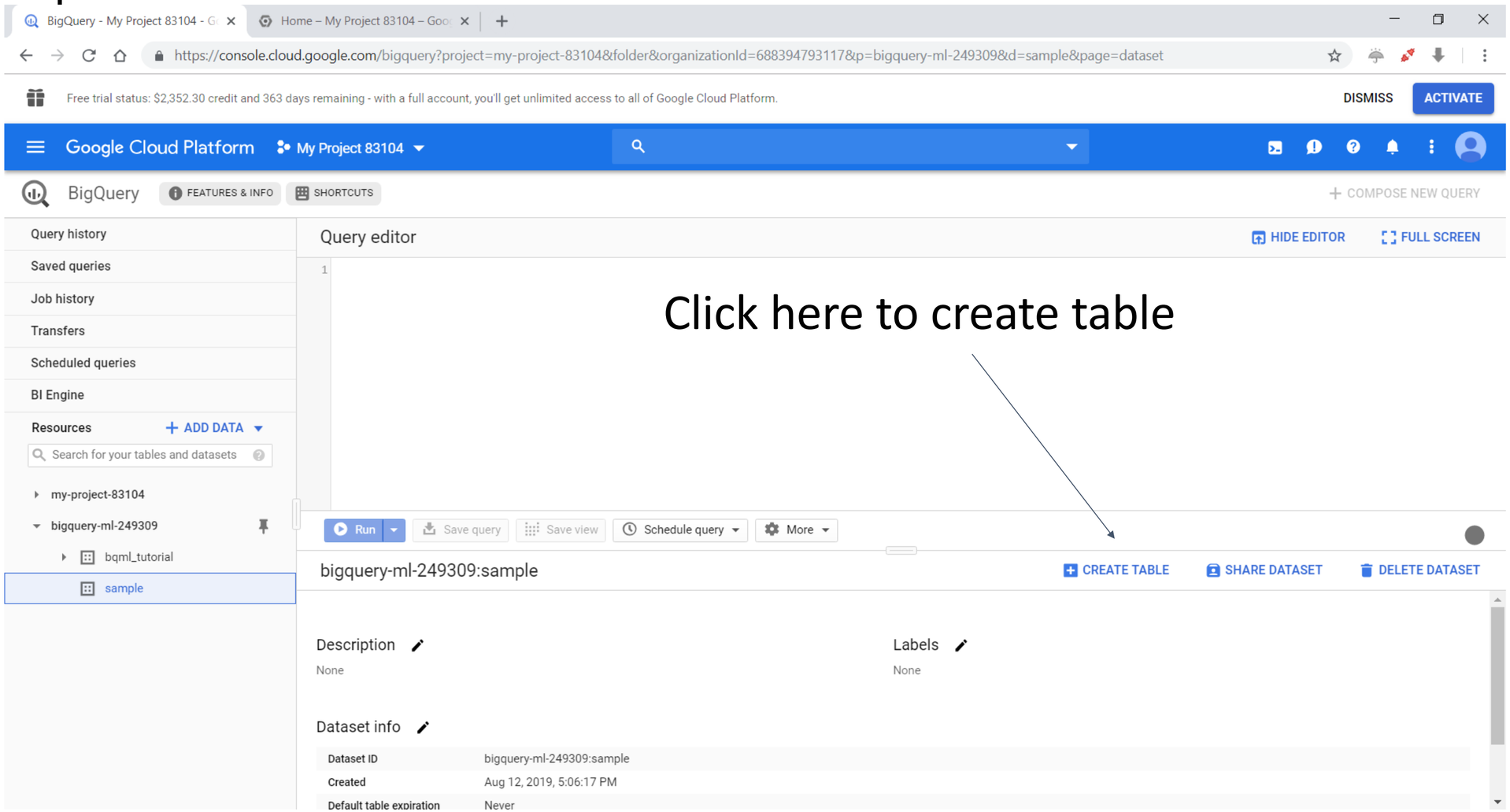
Accessing BigQuery web UI

Access the BigQuery web UI through this link:
https://console.cloud.google.com/bigquery



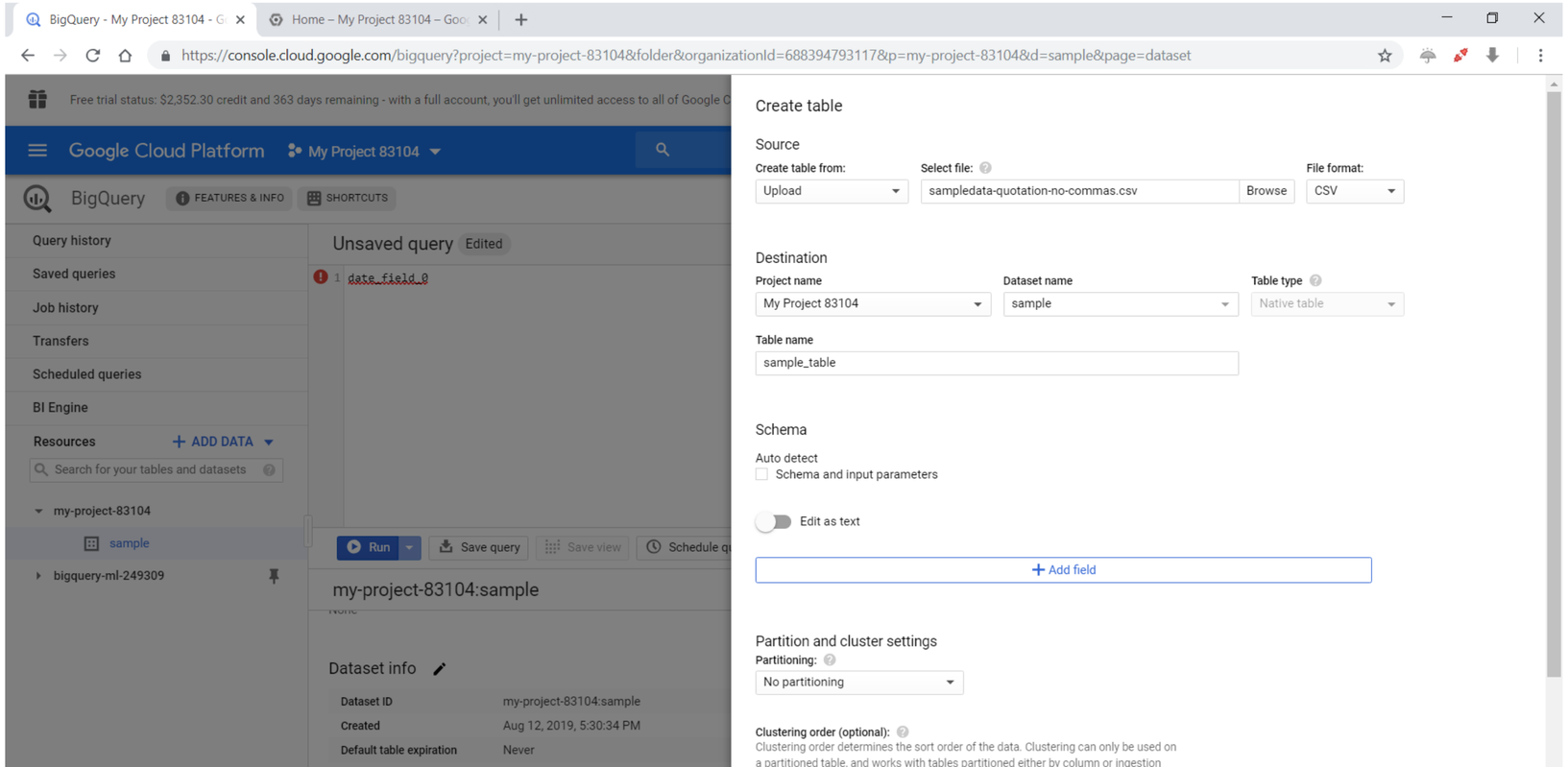
Creating a new table

Click on the dataset you want to upload local data in and there should be a Create table option for it



Upload local data

This will take you to the table creation window. In this window under Source, choose upload under *Create table from* and *Upload*. Select the file (csv) you want to upload and set the file format to the appropriate one. Make sure you input a name for the table, schema and input parameters. Once you're done, click Create table.



XCCELERATE

RUNNING ANALYTIC QUERIES IN BIGQUERY

XCCELERATE

CONNECTING TO BIGQQUERY IN JUPYTER NOTEBOOK

XCCELERATE

BIGQUERYML

XCCELERATE

ADDITIONAL INFORMATION – DENORMALIZED STORAGE IN BIGQUERY

Normalized Storage in Traditional Databases

Traditional Database Design

Normalization

Minimizes redundancy, optimizes storage

Foreign keys to ensure valid joins

Updates in one location, no duplication of data

Employee Information

Employee Details

Id	Name	Department	Grade
1	Emily	Finance	6

Employee Subordinates

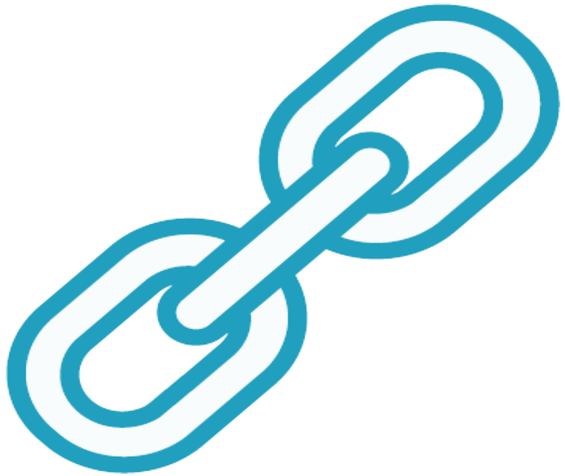
Id	Subordinate Id
1	2
1	3

Employee Address

Id	City	Zip Code
1	Palo Alto	94305
2	Seattle	98101

Denormalized Storage in BigQuery

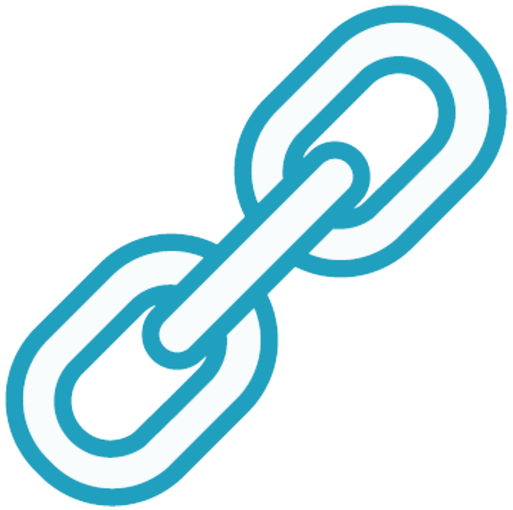
Denormalized Storage



Denormalized data

Data is compressed into one table to be read in a single operation

Denormalized Storage

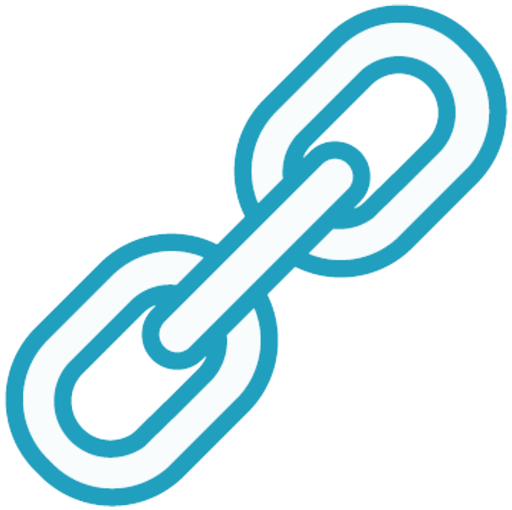


Disk space is very cheap

No foreign key constraints

Read operations, no data updates

Denormalized Storage



- Optimize the number of disk seeks
- Store data for an entity in one location
- Ignore redundancy, minimize joins

Denormalized Storage

Id	Name	Function	Grade
1	Emily	Finance	6
2	John	Finance	3
3	Ben	Finance	4

Id	Subordinate Id
1	2
1	3

Id	Name	Function	Grade	Subordinates
1	Emily	Finance	6	<ARRAY>
2	John	Finance	3	
3	Ben	Finance	4	

Denormalized Storage

Id	Name	Function	Grade
1	Emily	Finance	6
2	John	Finance	3
3	Ben	Finance	4

Id	Subordinate Id
1	2
1	3

Id	Name	Function	Grade	Subordinates
1	Emily	Finance	6	2,3
2	John	Finance	3	
3	Ben	Finance	4	

Denormalized Storage

Id	Name	Function	Grade
1	Emily	Finance	6
2	John	Finance	3
3	Ben	Finance	4

Id	City	Zip Code
1	Palo Alto	94305
2	Seattle	98101

Id	Name	Function	Grade	Subordinates	Address
1	Emily	Finance	6	**2,3**	**<STRUCT>**
2	John	Finance	3		
3	Ben	Finance	4		

Denormalized Storage

Id	Name	Function	Grade
1	Emily	Finance	6
2	John	Finance	3
3	Ben	Finance	4

Id	City	Zip Code
1	Palo Alto	94305
2	Seattle	98101

Id	Name	Function	Grade	Subordinates	Address
1	Emily	Finance	6	**2,3**	**Palo Alto, 9430 5**
2	John	Finance	3		
3	Ben	Finance	4		

Denormalized Storage

Id	Name	Function	Grade	Subordinates	Address
1	Emily	Finance	6	**2,3**	**Palo Alto, 94305**
2	John	Finance	3		
3	Ben	Finance	4		

Store everything related to an employee in the same table

Denormalized Storage

Id	Name	Function	Grade	Subordinates	Address
1	Emily	Finance	6	**2,3**	**Palo Alto, 94305**
2	John	Finance	3		
3	Ben	Finance	4		

Get all details about an employee in one read operation

Google DataStudio

Interactive dashboards and reports

Connectors to GCP services and data stores

Easy to share with teams in your organization