



Google Cloud Platform



BigQuery

## **Google BigQuery: Introduction, Deep-Dive & Applications**

Harshit Arora, Patrick He, Halle Steinberg, Sean Yu

Team 1

ISOM 671 Managing Big Data

Emory University

February 17th, 2020

# Table of Contents

1. Introduction	3
2. Conceptual Foundation	3
3. Working with Google Cloud and BigQuery	6
4. BigQuery Application and Demo	10
5. Conclusion and Final Remarks	22
6. References	23

## Introduction

### *What is this project about?*

This report will provide an overview of Google BigQuery, a fully-managed and cloud-based interactive query service for massive datasets. We will provide an introduction to show how the platform can be used, including how to create a database, how to load data into a database, and how to perform queries in order to retrieve and modify useful information from a database. We hope to cover the following steps in detail:

- Creating a database
- Adding tables to the database
- Loading data into a table
- Performing queries on a dataset
- Connecting to BigQuery with outside tools such as Python

### *What is our motivation for this project?*

We were motivated to explore how Google BigQuery works because it is the main data storage and retrieval tool used by The Home Depot, which is our team's Capstone partner. We wanted to explore how the platform works and how we can maximize its functionality in order to best perform various data retrieval tasks and analysis for our project. In fact, THD is one of Google Cloud's leading partners, as BigQuery has allowed for THD to efficiently solve problems and sustain its customer-first service mentality, empowering associates to access timely data and keep over 50,000 items stocked worldwide ([cloud.google.com/the-home-depot](https://cloud.google.com/the-home-depot)). Our ability to perform certain tasks for THD is contingent on us becoming BigQuery knowledge experts, so we hope to achieve this and share our findings in this report.

### *What will we accomplish in this report?*

We will provide a step-by-step overview of how BigQuery works through a demonstration using one of Google's publicly available datasets. The dataset contains information on how much money is spent by verified advertisers on political advertising across Google Ad Services. We will explain how to connect to BigQuery using Python and perform queries using Python in order to retrieve the data from our database and perform some simple analytical tasks.

## Conceptual Foundation

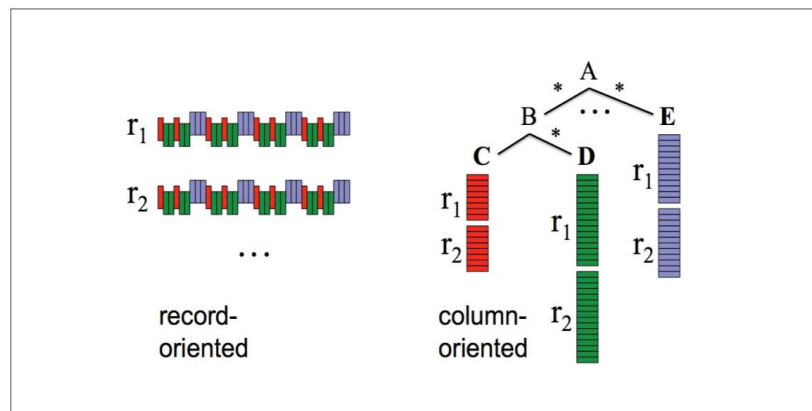
### *What is Google BigQuery?*

Google BigQuery is part of Google's Cloud Platform and allows users to perform queries and retrieve data fast at very low costs (Sato, 2012). Google BigQuery is Google's external implementation, accessible to third party developers and companies, with a level of usability that engineers and non-engineers alike are able to understand and handle. Internally, Google engineers use an implementation called Dremel, the core technology that Google BigQuery is built on. BigQuery is the public implementation of Dremel, providing the same core set of features to users

outside of Google. Dremel was first designed by Google to overcome issues faced due to MapReduce, such as the performance rate when attempting to handle large distributed datasets (Sato, 2012).

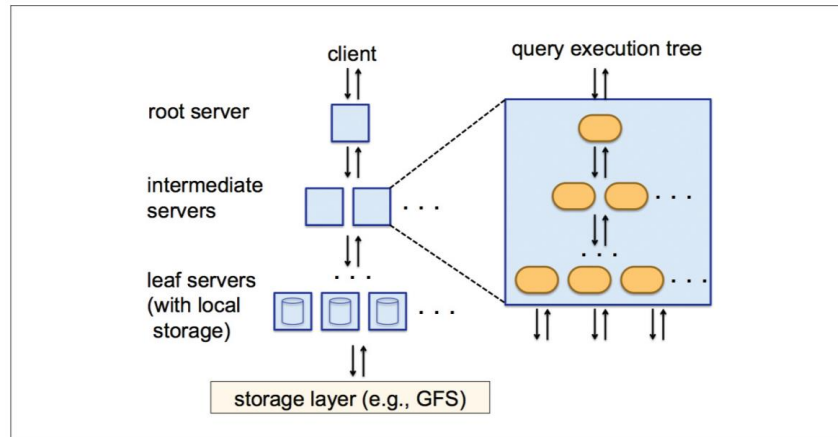
In order to fully capture BigQuery's role in big data, we first need to understand what Dremel is and how it works. Dremel is Google's internal query service that allows users to run SQL queries on very large datasets and retrieve results in just seconds (Sato, 2012). Dremel parallelizes each query it receives and runs it on tens of thousands of servers at the same time, making it clear just how powerful its scalability is. Because the platform requires only basic knowledge of SQL, Dremel (and BigQuery) can be used by both analysts and engineers alike to perform analytical tasks (Melnik et al., 2010).

As we can see, Dremel's scalability and query performance are two key factors that make it so attractive to users. Two core technologies are responsible for Dremel's performance: columnar storage and tree architecture. Data within Dremel is stored in a columnar storage manner which allows for a very high compression ratio and scan throughput. This means that Dremel separates each record into separate column values and stores each of these values on a different storage volume (Sato, 2012). Most traditional databases store the entire record together on one volume. By storing data on different storage volumes, only required column values in a query will be scanned and executed, minimizing the flow of traffic within the system. One drawback of columnar storage, however, is that it does not efficiently allow for updates to records. Because of this, columnar storage frameworks are typically only used in OLAP systems that store historical records that won't need to be updated (Melnik et al., 2010).



*Dremel Columnar Storage (Sato, 2012)*

Because Dremel stores data in a columnar fashion, one of the challenges Google faced in its design was how to collect query results across thousands of machines, which is resolved by its tree architecture. With this framework, Dremel is able to quickly send out queries and receive results by forming parallel distributed trees. These two factors are the reason behind Dremel's outstanding speed and low-cost performance (Sato, 2012).



*Dremel Tree Architecture (Sato, 2012)*

In addition to BigQuery’s columnar storage paradigm and tree architecture that allow for fast data scanning and more efficient querying and retrieval, BigQuery is also a serverless technology. In order to successfully manage a serverless platform, BigQuery leverages Google Cloud’s cloud architecture and data ingest models such as batch processing that help it to more dynamically store and warehouse data (sisense.com).

We now understand the underlying technologies that make Dremel so attractive to users, and now that BigQuery is publicly available for outside businesses and developers, Google outsiders can now utilize the power of Dremel for all necessary big data storage, processing, and analysis. BigQuery is basically the public version of Dremel, as it provides the same core features via a REST API, command line interface, Web UI, access control, data schema management, and an integration with Google Cloud Storage (Sato, 2012). In both Dremel and BigQuery technologies, we see valuable features such as high scalability and incredibly fast speed.

One of the most popular big data programming models is MapReduce, a paradigm that performs map and reduce tasks, which are automatically parallelized and executed on large clusters of hardware. It has many benefits in cloud computing such as flexibility and cost savings, and it has been used by many large corporations including Google for many years. When Google introduced BigQuery to the public, many developers and business analysts wondered how it differs from the popular Hadoop implementation of MapReduce. The main difference lies in the framework: BigQuery is designed to provide an interactive data analysis user experience for large datasets, and MapReduce instead uses a batch processing technique to manage large datasets, making it less suitable for iterative data analysis since turnaround time is so slow (stackshare.io). If instantaneous results are needed, MapReduce will not perform well. MapReduce takes much longer than BigQuery, on the scale of hours or days for MapReduce versus seconds or minutes for BigQuery (Sato, 2012). For this reason, Google created Dremel and eventually released it to the public as BigQuery.

Even though speed is BigQuery's most notable feature especially compared to MapReduce, there are some features that MapReduce capitalizes on that BigQuery does not. For example, BigQuery does not efficiently allow updates, but MapReduce has no problem with it. MapReduce is also much better at dealing with unstructured data than BigQuery. BigQuery can only handle structured data, making it much more suitable for OLAP or BI tasks, where most queries are done through quick aggregation and need to be performed quickly (Sato 2012).

Below is a summary of the key differences between BigQuery and MapReduce, as detailed in the Google BigQuery White Paper (Sato, 2012).

Key Differences	Big Query	MapReduce
OLAP/BI	Yes	No
Data mining	Partially	Yes
Speed	Very fast (seconds, minutes)	Slow (hours, days)
Easy to use for non-programmers	Yes	No
Complex programming/data processing logic	No	Yes
Unstructured data	No	Yes
Handling large results/ joins	No	Yes
Updates	No	Yes

We can see that BigQuery's biggest advantage is its processing speed and performance. It can process large amounts of data in a fraction of the time that MapReduce can, and it is also much more user-friendly, especially for non-technical users. Because data can be retrieved in BigQuery simply by writing SQL queries, anyone that is familiar with this language can easily use it, whereas MapReduce requires much more extensive technical knowledge. BigQuery is clearly the best choice for any necessary analytical tasks that require fast results at a highly cost-effective rate.

## Working with Google Cloud and BigQuery

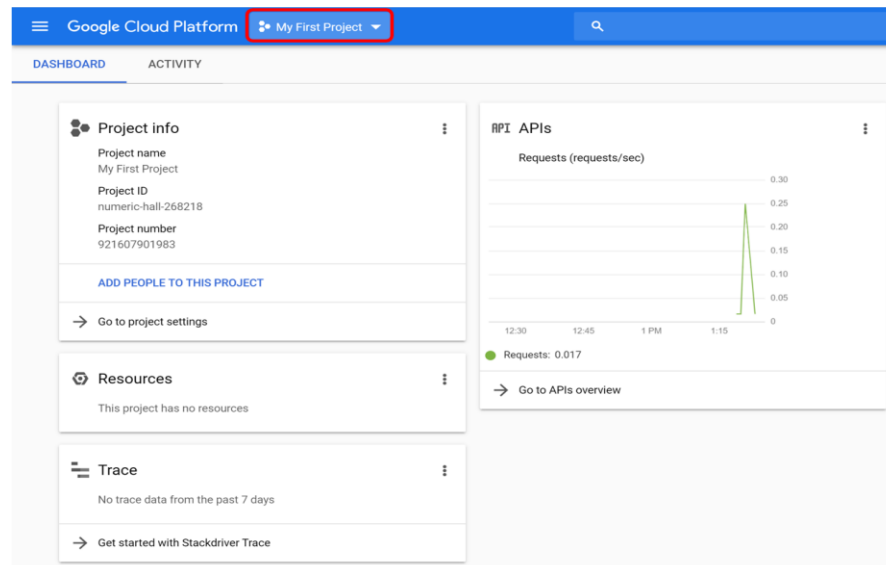
Registering for a Google Cloud account is free, and Google provides a \$300 credit for the first 12 months of setup. In the simplest case, this credit will be utilized for storing, retrieving, and querying data on BigQuery.

## Setting Up a Google Cloud Account

1. Visit <https://cloud.google.com/>
2. Click on “Sign-In” OR “Get Started for Free” (depending on whether you already have an account).
3. Provide your billing information (Credit Card or Bank Account - *You are not billed unless you activate automatic billing*).
4. Once signed in, click on Console to visit your Google Cloud dashboard.

## Creating a Project on Google Cloud’s BigQuery

1. You should automatically see a project called “My First Project” as shown. Click on the drop-down button next to it.




2. In the pop-up, click on “New Project” and type a name for your Project. Keep in mind that each project can contain multiple datasets, and each dataset is designed to store multiple tables.



3. Type a project name. You will see a default Project ID assigned to your Project. We recommend renaming this to something you can easily remember, as this will be the unique identifier used in order to pin your projects or load them into Python/R. You can see this later as well, but you cannot change it.

## New Project

 You have 10 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)  
[MANAGE QUOTAS](#)

Project name \*

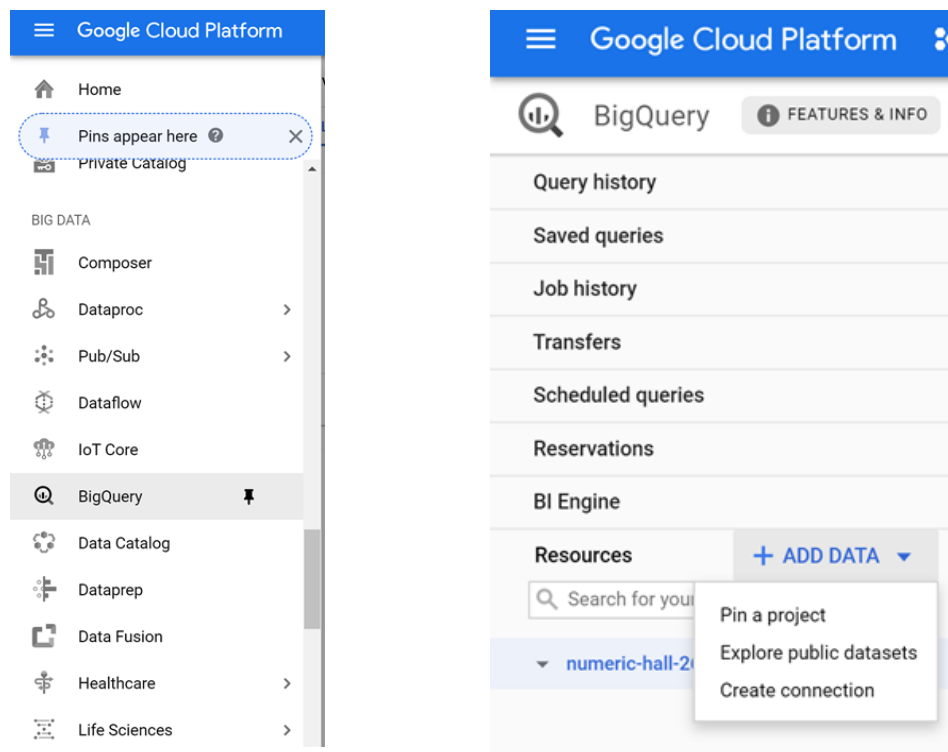
Project ID: managing-big-data-268218. It cannot be changed later. [EDIT](#)

Location \*  
 [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

- Open BigQuery by using the drop-down option and selecting BigQuery under “Big Data”.



- You are now ready to use this project in BigQuery. For convenience, it is always recommended to pin both BigQuery and the Project within BigQuery, as shown.



## Pin a project

Select a project from the list below to add it to the Resources tree. It will be pinned for easy access going forward.

☐ Enter a project name

☒ Search for a project

Select a project

All projects

Managing Big Data

managing-big-data-pr...

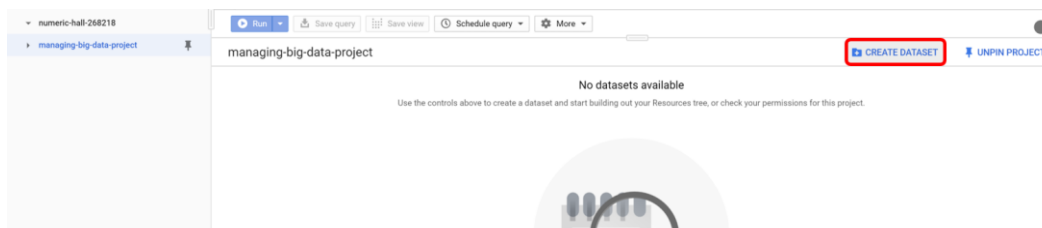
My First Project

numeric-hall-268218

PIN

## Creating a Dataset in a BigQuery Project

1. As mentioned earlier, it is easy to set up a database that can hold multiple tables and views for querying. To do this, select the project that you pinned earlier and click on 'CREATE DATASET'.



2. Setup a Database-ID. This is a unique database identifier within your project.
3. You should now be able to see the database within the project.

## Adding Tables to the Database

1. There are multiple options to add tables within a database, as shown. You can work with an empty table and add data manually using SQL. Alternatively, if you want to work with an already existing dataset (with multiple tables), you can either upload it from your local machine or load it from your Google Drive.

Create table

Source

Create table from:

Empty table

Google Cloud Storage

Upload

Drive

Google Cloud Bigtable

Destination

Project name

Managing Big Data

Dataset name

Political\_Ads

Table type

Native table

Table name

Letters, numbers, and underscores allowed

2. The most common way of loading large tables (each sized hundreds of GBs) is to load them from Google Cloud Storage (GCS). GCS is similar to an S3 Bucket on Amazon Web Services, in that it can store large datasets permanently (until deleted manually or expired

based on a pre-specified time). You can load datasets stored on the GCS by using their unique address, as shown. *Details on how to use GCS are provided later.*

Create table

Source

Create table from: Google Cloud Storage

Select file from GCS bucket: ☒ https://storage.cloud.google.com/test\_bucket\_test Browse

File format: CSV

Destination

Project name: Managing Big Data

Dataset name: test

Table type: Native table

Table name: Letters, numbers, and underscores allowed

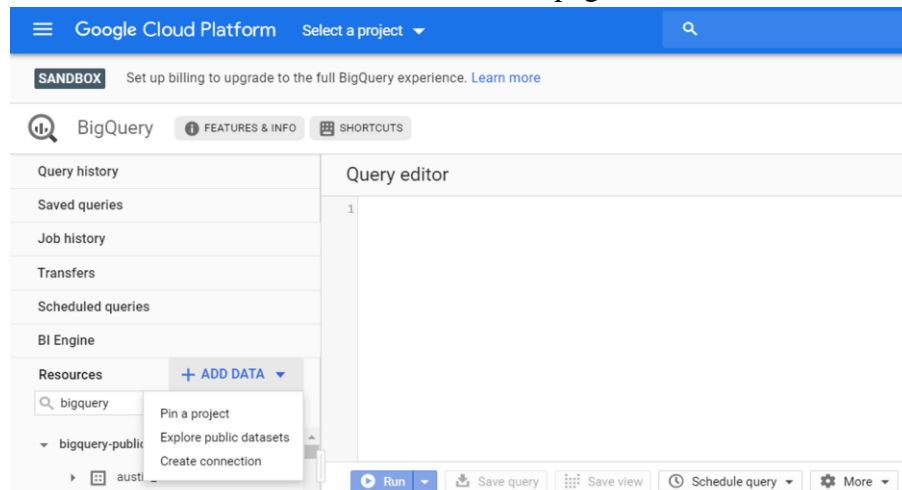
3. Finally, it is possible to work with tables without uploading them yourself. For instance, there are public datasets available under the project 'bigquery-public-data' which can be accessed by anyone. You can also use external datasets or shared datasets. In the case of external datasets, you can create a table using this external data source.

## BigQuery Application and Demo

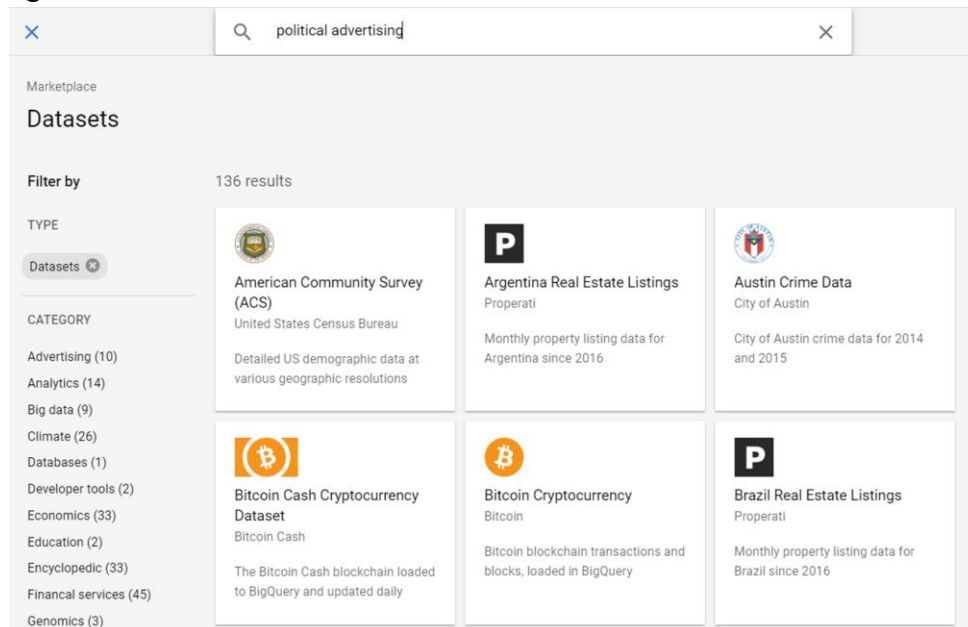
For our demonstration of Google BigQuery, we will be using a publicly available dataset that is hosted within BigQuery. The dataset we will be using is *Political Advertising on Google*, a dataset that contains information on how much money is spent by verified advertisers on political advertising across Google Ad Services. We plan to use this dataset to demonstrate BigQuery's use cases, including how we can connect to the data using Python. Since this data is hosted in BigQuery already, some of the steps will be different than what we've outlined above.

### Copying Public Data to Our Project Folder

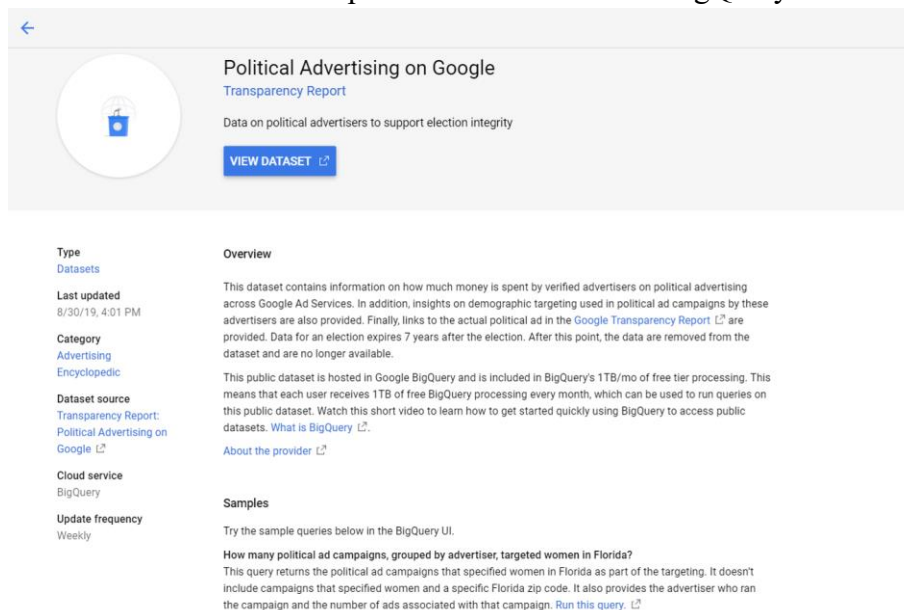
1. Since this data is hosted in BigQuery, we need to find it and put it into our Project folder that we've created. To first search for the specific dataset in GCP, click on the 'ADD DATA' button that is on the left-hand side of the webpage.



2. Click 'Explore public dataset'. There will be a new window pop out and we can search the name of the dataset we are looking for. In our case, we can simply search 'political advertising'.

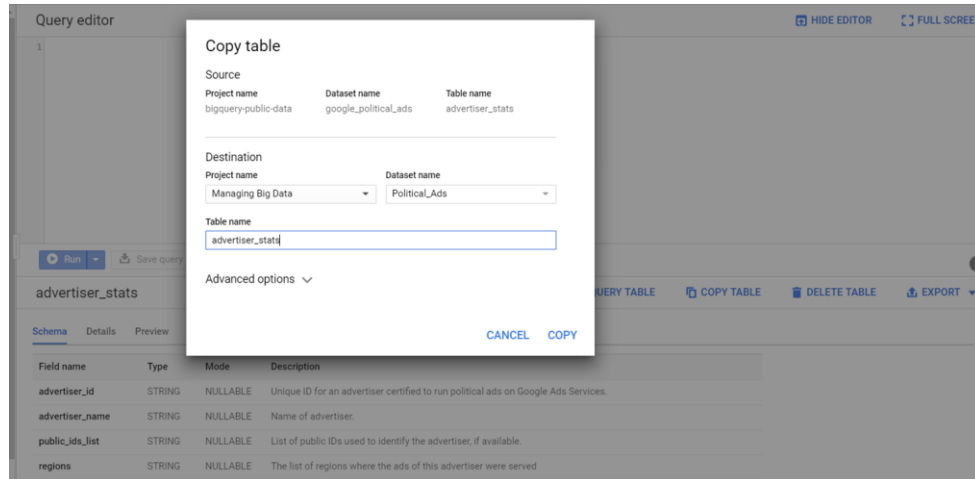


3. Click on the dataset we want to use to see the data overview and information.
4. Click on 'VIEW DATASET' to open the dataset folder in BigQuery.

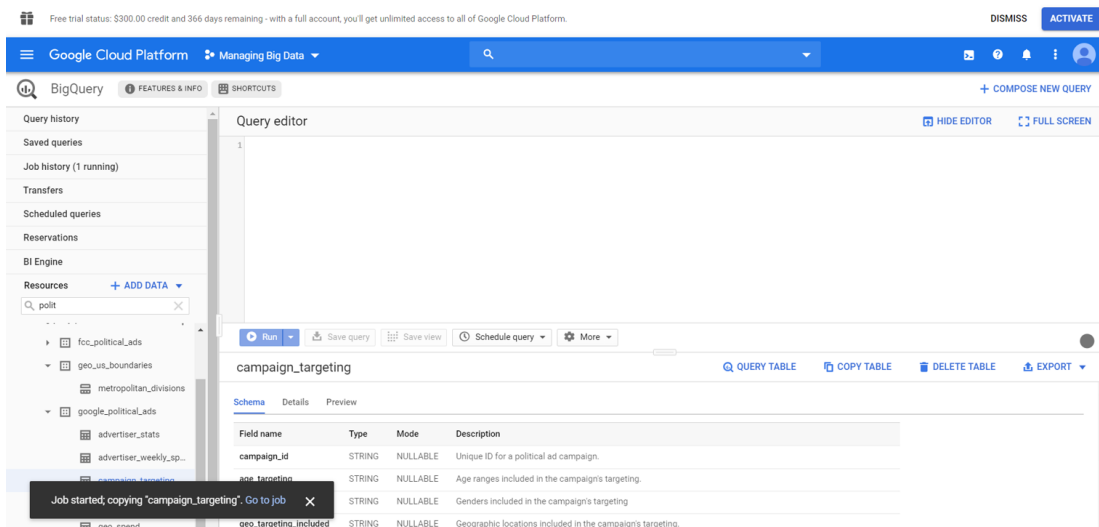


BigQuery is a big data database, where we can query, copy, create or even delete data. Without actually writing SQL commands, BigQuery can help us accomplish what we want with just a few clicks. To move the data we just found to our project folder for future analytical tasks, BigQuery provides a convenient way.

1. To add GCP public data into our Project folder, first locate the table we are trying to use in the public data.
2. Click on the 'COPY TABLE' button and a new window will pop out. Select the Project name we are using and database name that we want the table to be in. In our case, we will be using the 'Managing Big Data' Project and 'Political\_Ads' which we just created.



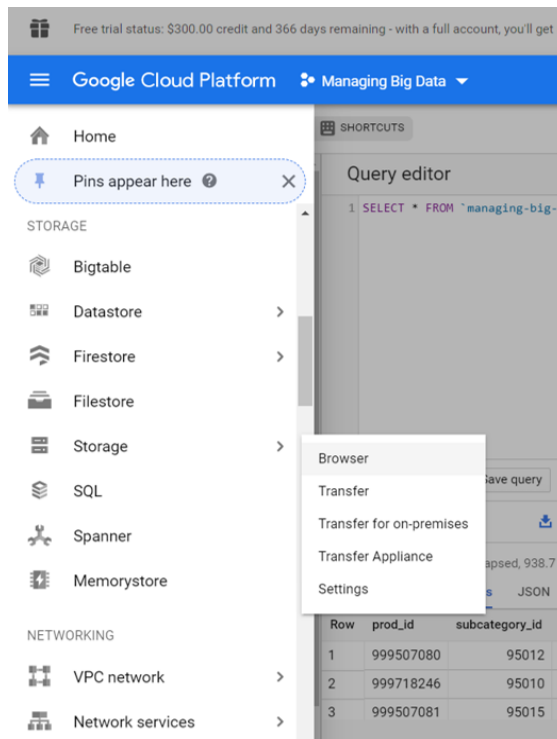
3. Create a name for the table. It can be the same as the original table since we are creating the table in a different location.



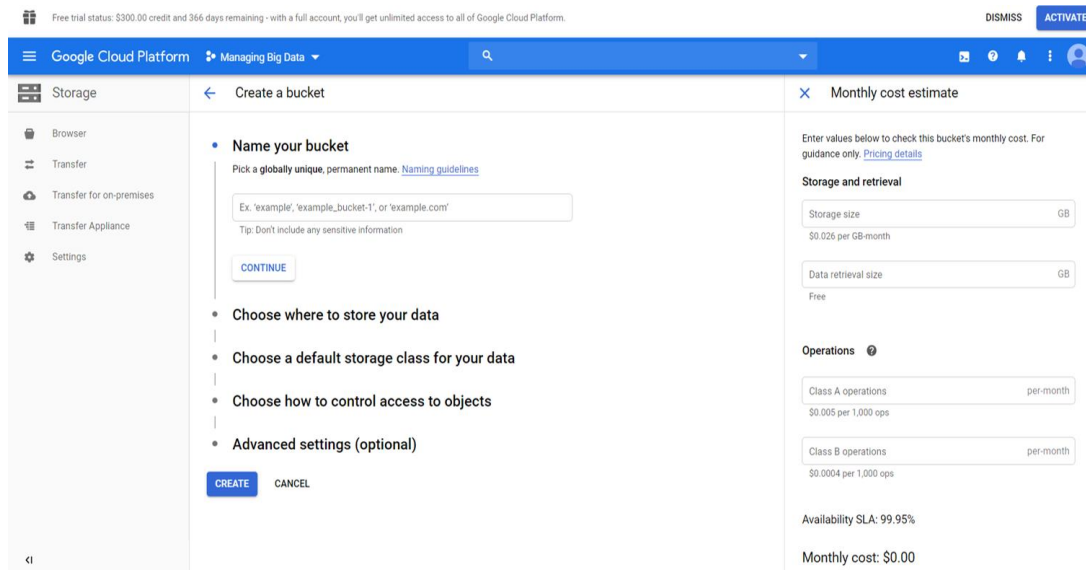
Another popular way to create tables is to fetch data from Google Cloud Storage, where you can upload a dataset from your local machine or other sources. For our demonstration, we will explain how to upload a dataset from the local machine to create a table with datasets that exist in Google Cloud Storage.

### *Uploading Files to Google Cloud Storage*

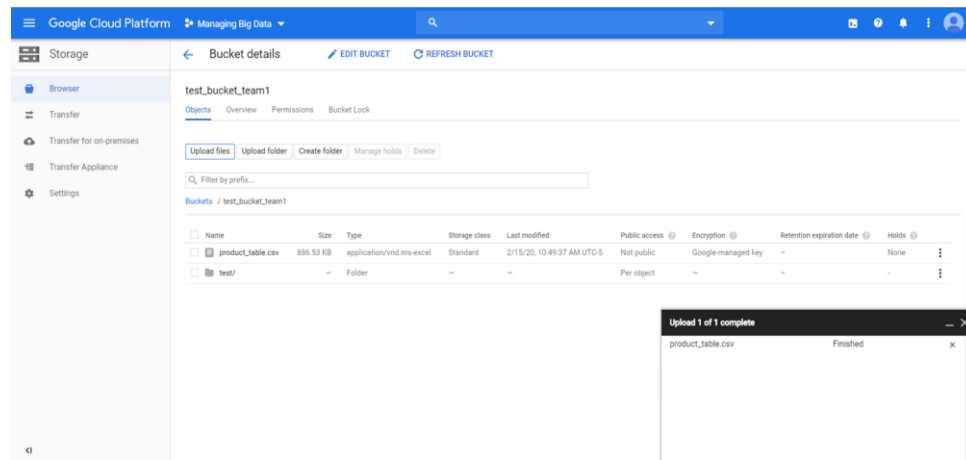
1. Click on the drop-down button on the top left of the web page and find 'Storage'.



2. Click 'Browser'. If you are a new user of GCP, you might need to create a bucket to store your data.
3. Create your bucket name and choose your bucket setting based on your needs.

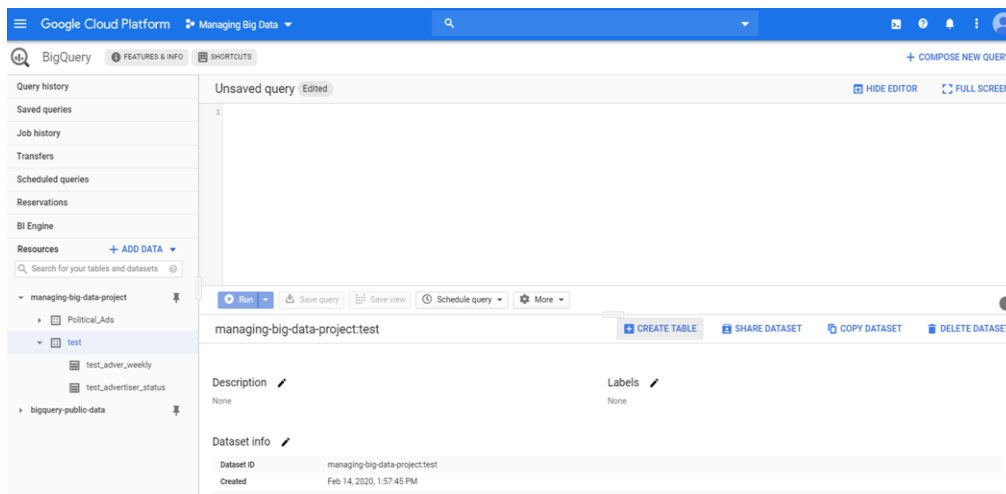


4. You will see the bucket you just created on the web page. Click on the bucket name to access the contents of your bucket.
5. Click 'Upload files' to upload files from your machine.



## Creating Tables from Google Cloud Storage

1. Navigate back to BigQuery. Find the project and dataset we want to use, then click 'CREATE TABLE'.



2. From 'Source', click Google Cloud Storage, then choose the bucket name and locate to the file we just uploaded.

Create table

Source

Create table from: Google Cloud Storage Select file from GCS bucket:  File format: CSV

Destination

Project name: Managing Big Data Dataset name: test Table type: Native table

Table name

Letters, numbers, and underscores allowed

Schema

Auto detect

☐ Schema and input parameters

☐ Edit as text

[+ Add field](#)

3. Next, choose the Project and dataset name that we want to create our table in, and create a new table name.
4. GCP can automatically create a schema for your data, or you can create the schema manually.
5. Click 'Create table'. Now we can see the new table.

### *Accessing Data in BigQuery with Colab*

Colab is a Python development environment that runs in your browser using Google Cloud. It supports TensorFlow, BigQuery, and Google Drive. We can directly access the datasets on BigQuery by providing the proper credentials and Project ID. One advantage of Colab is that we don't have to set up an API or provide the access key locally every time we want to query data from Google Cloud.

1. Provide credentials by logging into your Google Account.

▼ Provide your credentials to the runtime

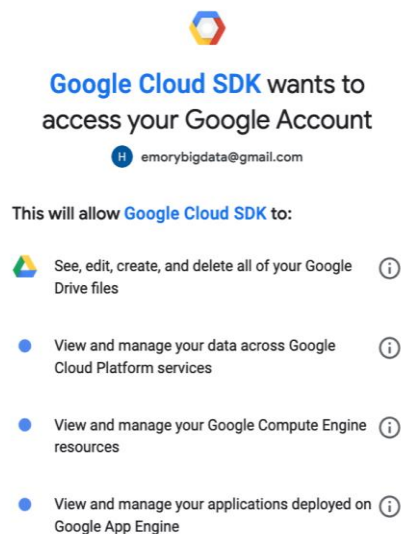
```
from google.colab import auth
auth.authenticate_user()
print('Authenticated')
```

... Go to the following link in your browser:

[https://accounts.google.com/o/oauth2/auth?code\\_challenge=0iuC097uwyjhdPsnv2E882KnL\\_7KOicLqG9dzGtUm3o&prompt=select\\_account&code\\_challenge\\_method=S256&access\\_ty](https://accounts.google.com/o/oauth2/auth?code_challenge=0iuC097uwyjhdPsnv2E882KnL_7KOicLqG9dzGtUm3o&prompt=select_account&code_challenge_method=S256&access_ty)

Enter verification code:

2. By logging into your Google Account related to the service, you are automatically directed to a page that asks your permission to let Google SDK access your account and provide your access key to the Google Cloud service.



3. Next, you are able to use the Project ID and the table name to access the data you want to get and customize the query through Google BigQuery.

```
[ ] from google.colab import auth
auth.authenticate_user()
print('Authenticated')
```

Authenticated

4. We can then access the database via Magic, google-big-query, and pandas-gbq.
  - a. On Colab, we can access BigQuery through Magic, a command which runs a query and either displays the result or saves it to a variable as a dataframe, or Python packages like google-cloud-bigquery and pandas-gbq.
  - b. For example, we can specify the Project ID and table name to run a query via Magic. We can save the query result to a dataframe, which can be edited and further processed in Python.

```
[11] # Save output in a variable `df`

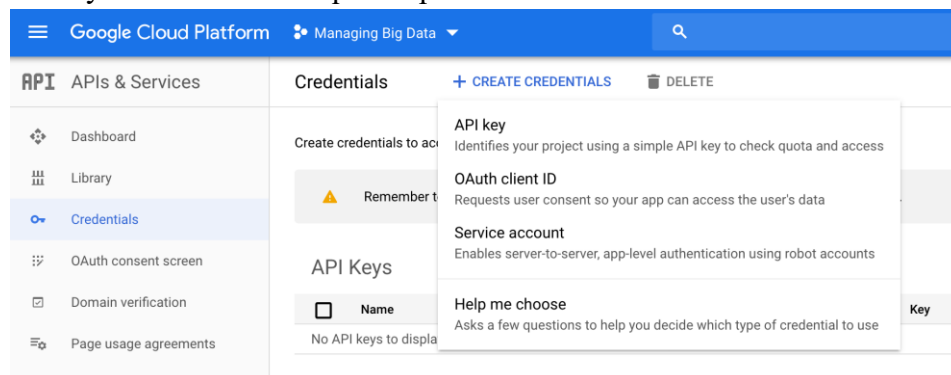
%%bigquery --project managing-big-data-project df
SELECT *
FROM `managing-big-data-project.Political_Ads.advertiser_stats`
LIMIT 10
```

[12] df

1 to 10 of 10 entries Filter ?															
index	advertiser_id	advertiser_name	public_ids_list	regions	elections	total_creatives	spend_usd	spend_eur	spend_inr	spend_bgn	spend_hrk	spend_czk	spend_dkk	spend_huf	spend_pln
0	AR106804085620998144	EXACT DRIVE, INC.	EIN ID 27-1797458	US	US-Federal	6	0	0	250	0	0	0	0	0	0
1	AR130306833859477504	FRIENDS OF DANIEL SPARKMAN	EIN ID 83-4323123	US	US-Federal	1	0	0	0	0	0	0	0	0	0
2	AR131651811458154496	Jayprakash Murli Prajapati		IN	IN-General	1	0	0	0	0	0	0	0	0	0
3	AR131714208743030784	RATLIFF ENTERPRISES	EIN ID 86-0859865	US	US-Federal	1	0	0	250	0	0	0	0	0	0
4	AR141456019204079616	Birchwood & Saks Investment Finland Oy		EU, FI	EU-Parliament	1	0	0	0	0	0	0	0	0	0
5	AR166715924462698496	Asociația Platforma Inițiativa România		EU, RO	EU-Parliament	5	0	0	0	0	0	0	0	0	0

## Creating Credentials for Google Cloud API

1. Next, we need to create credentials in order to use the Google Cloud API. First, navigate to the API & Services page to create credentials. For our project, we will create a Service Account key which doesn't require a password.



2. Type in a Service Account name and description.



1 Service account details — 2 Grant this service account access to project (optional)

### Service account details

Service account name

BigData

Display name for this service account

Service account ID

bigdata-805

@managing-big-data-project.iam.gserviceaccount.co



Service account description

Managing Big Data

Describe what this service account will do

CREATE

CANCEL

3. Set account permissions role as Project Owner so that you have full access to the database. You can also assign viewing or editing rights to each service account depending on the level of authentication that needs to be provided without compromising the data.

✓ Service account details — 2 Grant this service account access to project (optional)

### Service account permissions (optional)

Grant this service account access to Managing Big Data so that it has permission to complete specific actions on the resources in your project. [Learn more](#)

Role

Owner



Condition

[Add condition](#)



Full access to all resources.

[+ ADD ANOTHER ROLE](#)

CONTINUE

CANCEL

4. Next, create a JSON key that will be stored locally. This is a private key containing your credentials and can be generated only once. Make sure you save it in a place you will remember.

### Create key (optional)

Download a file that contains the private key. Store the file securely because this key can't be recovered if lost. However, if you are unsure why you need a key, skip this step for now.

Key type

☒ JSON

Recommended

☐ P12

For backward compatibility with code using the P12 format

CREATE

CANCEL

## Connecting to BigQuery with Python on Local Machine

Sometimes we may want to use Python on a local console to access/manipulate data via Jupyter Notebooks. It is relatively easy, efficient, and has a bunch of convenient applications. For instance, if we store TBs of data through multiple tables of a schema, we might not want to load the complete dataset into our memory in order to perform manipulation and computations. One of the key advantages of using BigQuery with Python is that you can pass complex queries to load data into a Pandas dataframe depending on your requirements. If we want to filter some rows, or perform joins & aggregations, we can use SQL queries within Python to only retrieve what we need from the database.

1. Ensure you have the private key we just saved in the form of a JSON file stored on your local machine. This JSON key will provide the credentials and ensure you are authenticated to access the data on Google Cloud. This is the same step as creating credentials by setting up a service account.
2. We will use the bigquery library from the google.cloud package. Before using this package, you need to ensure it is installed on your machine. If it is not, or to check that it is, execute the following command in your Terminal/Command Prompt:

**`pip install --upgrade google-cloud-bigquery`**

3. Ensure that the necessary packages are loaded in:

```
import pandas as pd
from google.cloud import bigquery
from google.oauth2 import service_account
credentials = service_account.Credentials.from_service_account_file(
    'C:/Users/harsh/Desktop/MSBA/Spring/Big Data/Project/key.json')
project_id = 'managing-big-data-project'
client = bigquery.Client(credentials=credentials, project=project_id)
```

- a. Your credentials are located in the JSON file - use service\_account.Credentials to provide the location of this JSON file as shown above (key.json in our case).
- b. The other component of the authentication request is the *unique* Project ID. Users who have access to one project may not have access to another. Pass the Project ID to BigQuery along with the JSON file.
- c. Use the bigquery.Client function to provide the credentials and Project ID. You can now access the data from Google Cloud by passing SQL queries.

4. To query data from BigQuery, use a simple SQL query as shown:

```
sql = """
SELECT * FROM `managing-big-data-project.Political_Ads.advertiser_stats` LIMIT 1000"""
```

```
df = client.query(sql).to_dataframe()
```

- a. The client.query function and to\_dataframe methods are used to pass the query to BigQuery, the result of which is stored as a Pandas dataframe called df.
- b. You are now ready to use this dataframe for anything that you can do in Python!

```
df.head()
```

	advertiser_id	advertiser_name	public_ids_list	regions	elections	total_creatives	spend_usd	spend_eur	spend_inr	spend_bgn	spend_hrk
0	AR106804085620998144	EXACT DRIVE, INC.	EIN ID 27-1797458	US	US-Federal	6	0	0	250	0	0
1	AR130306833859477504	FRIENDS OF DANIEL SPARKMAN	EIN ID 83-4323123	US	US-Federal	1	0	0	0	0	0
2	AR131651811458154496	Jayprakash Murli Prajapati	None	IN	IN-General	1	0	0	0	0	0
3	AR131714208743030784	RATLIFF ENTERPRISES	EIN ID 86-0859865	US	US-Federal	1	0	0	250	0	0
4	AR141456019204079616	Birchwood & Saks Investment Finland Oy	None	EU, FI	EU-Parliament	1	0	0	0	0	0

### *Application Dataset: Political Advertising on Google*

This dataset contains information about how much money is spent by verified advertisers on political advertising across Google Ad Services. In addition, insights on demographic targeting used in political ad campaigns by these advertisers are also provided. Data for an election expires seven years after the election.

This public dataset is hosted in Google BigQuery and is included in BigQuery's 1TB/month of free tier processing. This means that each user receives 1TB of free BigQuery processing every month, which can be used to run queries on this public dataset. The dataset contains the following six tables.

1. *Advertiser\_stats: contains spending for each advertiser in various currencies*
2. *Advertiser\_weekly\_spend: similar to Advertiser\_stats, but contains spending at a weekly level rather than election aggregate*
3. *Campaign\_targeting: contains spending for each political campaign, its location, and the population targeted (gender, age groups), start and end date*
4. *Creative\_stats: contains spending for online ads, their URLs, potential spending, impressions, and duration*
5. *Geo\_stats: spending in each country in local currencies*
6. *Top\_keywords\_history: contains keywords with the maximum spending in each election*

*Which advertisers spend the most on political ad campaigns?*

1. We want to find the top advertisers by spending in each election region. Since the spending is calculated in different currencies, we first need to convert all spendings into US dollars using the 2019 annual average exchange rate.
2. Next, we use the Partition and Row\_Number function to pick the top 5 advertisers in each region and change the spending format to US dollars. Also, there are many advertisers with multiple advertiser\_id, so we need to group rows by advertiser name.

Query editor HIDE EDITOR FULL SCREEN

```

1 SELECT * FROM(
2 SELECT advertiser_name,elections, CONCAT("$",CAST(Total_Spending AS INT64)) as Spending,
3 ROW_NUMBER() OVER (PARTITION BY elections ORDER BY Total_Spending DESC) AS rank
4 FROM(
5 SELECT advertiser_name, elections, sum(Total_Spending_USD) as Total_Spending
6 FROM(
7 SELECT advertiser_name,elections, regions, (spend_usd + spend_eur/0.893 + spend_inr/70.394 + spend_bgn/1.66 + spend_hrk/6.72 + spend_czk/21.49 + spend_dkk/6.61
8 + spend_huf/285.81 + spend_pln/3.78 + spend_ron/4.18 + spend_sek/9.41 + spend_gbp/0.784) as Total_Spending_USD
9 FROM `managing-big-data-project.Political_Ads.advertiser_stats`
10 WHERE elections IS NOT NULL
11 )
12 GROUP BY advertiser_name,elections
13 ORDER BY sum(Total_Spending_USD) DESC))
14 WHERE rank <= 5
15 ORDER BY elections DESC

```

3. From the query results, we can see that MIKE BLOOMBERG 2020 INC is the largest political advertiser in the US with a total spending of \$459,665,851, while, an advertiser supporting TRUMP has the second largest spending of \$152,565,798.

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.6 sec elapsed, 269.7 KB processed)

Job information Results JSON Execution details

Row	advertiser_name	elections	Spending	rank
1	MIKE BLOOMBERG 2020 INC	US-Federal	\$459665851	1
2	TRUMP MAKE AMERICA GREAT AGAIN COMMITTEE	US-Federal	\$152565798	2
3	TOM STEYER 2020	US-Federal	\$87737257	3
4	PETE FOR AMERICA, INC.	US-Federal	\$77930512	4
5	DONALD J. TRUMP FOR PRESIDENT, INC.	US-Federal	\$65597912	5
6	Bharatiya Janata Party	IN-General	\$24027440	1
7	DRAVIDA MUNNETRA KAZHAGAM	IN-General	\$6935651	2
8	Auburn Digital Solutions Private Limited	IN-General	\$5901308	3
9	Indian National Congress	IN-General	\$5216979	4
10	Yuvajana Sramika Rythu Congress Party	IN-General	\$2774190	5

4. With the same query and few lines of code, we are able to process the query in Python and save the query result as a dataframe.

```

In [12]: sql = """
SELECT * FROM(
SELECT advertiser_name,elections, CONCAT("$",CAST(Total_Spending AS INT64)) as Spending,
ROW_NUMBER() OVER (PARTITION BY elections ORDER BY Total_Spending DESC) AS rank
FROM(
SELECT advertiser_name, elections, sum(Total_Spending_USD) as Total_Spending
FROM(
SELECT advertiser_name,elections, regions, (spend_usd + spend_eur/0.893 + spend_inr/70.394 + spend_bgn/1.66 + spend_hrk/6.72 + spend_czk/21.49 + spend_dkk/6.61
+ spend_huf/285.81 + spend_pln/3.78 + spend_ron/4.18 + spend_sek/9.41 + spend_gbp/0.784) as Total_Spending_USD
FROM `managing-big-data-project.Political_Ads.advertiser_stats`
WHERE elections IS NOT NULL
)
GROUP BY advertiser_name,elections
ORDER BY sum(Total_Spending_USD) DESC))
WHERE rank <= 5
ORDER BY elections DESC"""

In [13]: df = client.query(sql).to_dataframe()

In [14]: df

Out[14]:

```

	advertiser_name	elections	Spending	rank
0	MIKE BLOOMBERG 2020 INC	US-Federal	\$459665851	1
1	TRUMP MAKE AMERICA GREAT AGAIN COMMITTEE	US-Federal	\$152565798	2
2	TOM STEYER 2020	US-Federal	\$87737257	3
3	PETE FOR AMERICA, INC.	US-Federal	\$77930512	4
4	DONALD J. TRUMP FOR PRESIDENT, INC.	US-Federal	\$65597912	5
5	Bharatiya Janata Party	IN-General	\$24027440	1
6	DRAVIDA MUNNETRA KAZHAGAM	IN-General	\$6935651	2
7	Auburn Digital Solutions Private Limited	IN-General	\$5901308	3
8	Indian National Congress	IN-General	\$5216979	4
9	Yuvajana Sramika Rythu Congress Party	IN-General	\$2774190	5
10	The Conservative & Unionist Party	EU-Parliament	\$14311676	1
11	Labour Party	EU-Parliament	\$9484028	2
12	NEA DEMOKRATIA	EU-Parliament	\$8838720	3
13	Ciudadanos - Partido de la Ciudadanía	EU-Parliament	\$5225758	4
14	Freiheitliche Partei Oesterreichs	EU-Parliament	\$3911628	5

Which advertisers produce the most successful advertisements in the US?

1. For our second application, we want to identify the advertisers that produce the most successful advertisements in terms of impressions. In this case, we are only looking at those advertisements with impressions over 1M. We first locate the advertisements with 1M impressions and above by selecting value “1M-10M” and “> 10M” in the impressions column. Then, we group the data by advertiser name and count the number of successful advertisements for each advertiser.

```
Query editor HIDE EDITOR FULL SCREEN
1 SELECT Advertiser, Ads_Impression_Over_1M, CONCAT("$",CAST(Total_Spending AS INT64)) as Spending,
2 CONCAT("$",CAST(Total_Spending/Ads_Impression_Over_1M AS INT64)) as Per_Ad_Spending
3 FROM(
4 SELECT Advertiser,COUNT(*) as Ads_Impression_Over_1M,
5 FROM(
6 SELECT
7   CS.ad_id,
8   CS.impressions AS Impressions,
9   CS.advertiser_name AS Advertiser,
10  CS.ad_campaigns_list AS AD_Campaigns,
11  regions
12 FROM `bigquery-public-data.google_political_ads.creative_stats` AS CS
13 WHERE CS.impressions = "1M-10M" OR CS.impressions = "> 10M")
14 WHERE regions = "US"
15 GROUP BY Advertiser)
16 JOIN (SELECT advertiser_name, elections, sum(Total_Spending_USD) as Total_Spending
17 FROM(SELECT advertiser_name,elections, regions, (spend_usd + spend_eur/0.893 + spend_inr/70.394 + spend_bgn/1.66 + spend_hrk/6.72 + spend_czk/21.49 +
18 spend_dkk/6.61 + spend_huf/285.81 + spend_pln/3.78 + spend_ron/4.18 + spend_sek/9.41 + spend_gbp/0.784) as Total_Spending_USD
19 FROM `managing-big-data-project.Political_Ads.advertiser_stats`
20 WHERE elections IS NOT NULL)
21 GROUP BY advertiser_name,elections) as spd
22 ON Advertiser = spd.advertiser_name
23 ORDER BY Ads_Impression_Over_1M DESC
```

2. The query result shows TRUMP MAKE AMERICA GREAT AGAIN COMMITTEE has the most successful ads in the 2018 federal election, but from our example above, we know that BLOOMBERG 2020 INC spent the most on advertising.
3. We also created a simple ratio, which divides the total spending by number of impressions to measure if the advertisers use their budget efficiently. It turns out TRUMP’s campaign advertisement is 5 times more efficient than BLOOMBERG’s advertisement.

Query results <span>SAVE RESULTS</span> <span>EXPLORE DATA</span>					
Query complete (0.7 sec elapsed, 17.3 MB processed)					
Job information <span>Results</span> <span>JSON</span> <span>Execution details</span>					
Row	Advertiser	Ads_Impression_Over_1M	Spending	Per_Ad_Spending	
1	TRUMP MAKE AMERICA GREAT AGAIN COMMITTEE	443	\$152565798	\$344392	
2	CONSERVATIVE BUZZ LLC	355	\$57548620	\$162109	
3	MIKE BLOOMBERG 2020 INC	279	\$459665851	\$1647548	
4	TOM STEYER 2020	132	\$87737257	\$664676	
5	NRCC	92	\$42653585	\$463626	
6	Need to Impeach	78	\$43945492	\$563404	
7	SENATE LEADERSHIP FUND	67	\$60449069	\$902225	
8	ASCENSION MARKETING GROUP, INC.	58	\$6878639	\$118597	
9	INDEPENDENCE USA PAC	46	\$26428581	\$574534	
10	NATIONAL RIFLE ASSOCIATION OF AMERICA POLITICAL VICTORY FUND	41	\$6213970	\$151560	

```
In [23]: sql = """
SELECT Advertiser, Ads_Impression_Over_1M, CONCAT("$",CAST(Total_Spending AS INT64)) as Spending,
CONCAT("$",CAST(Total_Spending/Ads_Impression_Over_1M AS INT64)) as Per_Ad_Spending
FROM(
SELECT Advertiser,COUNT(*) as Ads_Impression_Over_1M,
FROM(
SELECT
    CS.ad_id,
    CS.impressions AS Impressions,
    CS.advertiser_name AS Advertiser,
    CS.ad_campaigns_list AS AD_Campaigns,
    regions
FROM `bigquery-public-data.google_political_ads.creative_stats` AS CS
WHERE CS.impressions = "1M-10M" OR CS.impressions = "> 10M")
WHERE regions = "US"
GROUP BY Advertiser)
JOIN (SELECT advertiser_name, elections, sum(Total_Spending_USD) as Total_Spending
FROM(SELECT advertiser_name,elections, regions, (spend_usd + spend_eur/0.893 + spend_inr/70.394 + spend_bgn/1.66 + spend_rub/70.394) as Total_Spending_USD
FROM `managing-big-data-project.Political_Ads.advertiser_stats`
WHERE elections IS NOT NULL)
GROUP BY advertiser_name,elections) as spd
ON Advertiser = spd.advertiser_name
ORDER BY Ads_Impression_Over_1M DESC
LIMIT 10"""
```

```
In [24]: df = client.query(sql).to_dataframe()
```

```
In [25]: df
```

```
Out[25]:
```

	Advertiser	Ads_Impression_Over_1M	Spending	Per_Ad_Spending
0	TRUMP MAKE AMERICA GREAT AGAIN COMMITTEE	443	\$152565798	\$344392
1	CONSERVATIVE BUZZ LLC	355	\$57548620	\$162109
2	MIKE BLOOMBERG 2020 INC	279	\$459665851	\$1647548
3	TOM STEYER 2020	132	\$87737257	\$664676
4	NRCC	92	\$42653585	\$463626
5	Need to Impeach	78	\$43945492	\$563404
6	SENATE LEADERSHIP FUND	67	\$60449069	\$902225
7	ASCENSION MARKETING GROUP, INC.	58	\$6878639	\$118597
8	INDEPENDENCE USA PAC	46	\$26428581	\$574534
9	PRIORITIES USA ACTION & SMP	41	\$32871734	\$801750

## Conclusion and Final Remarks

In this report, we took a deep-dive into the foundation of Google BigQuery and explored use cases and applications through the *Political Advertising* dataset. BigQuery is a highly functional and scalable tool that both engineers and non-engineers can use due to its user-friendly interface and SQL query language. We learned that BigQuery is attractive for its high-speed performance when dealing with large datasets, something that is critical in today's world of big data and high accessibility. We have discussed a lot in our Managing Big Data class about MapReduce and its benefits and use cases, and we can now see that BigQuery has many advantages of its own, including processing speed and ease of use.

As cloud computing and big data continues to grow in use and popularity around the world, we can expect that BigQuery will continue to expand its presence and toolset to more uses cases and applications. We have seen already that BigQuery can be accessed through commonly used tools such as Python & R, allowing us to perform queries and retrieve information to be used directly on our local machines. As a team, we have found this research to be incredibly insightful, as it is information we will carry with us throughout our Capstone project. Because The Home Depot houses all of their data in Google Cloud, having this knowledge and skills will allow us to access the data we need with ease and connect to it with tools we are already familiar with such as Python and R.

## References

- Google BigQuery vs Hadoop: What are the differences? (n.d.). Retrieved from <https://stackshare.io/stackups/google-bigquery-vs-hadoop>
- Google Colaboratory. (n.d.). Retrieved from <https://colab.research.google.com/notebooks/bigquery.ipynb>
- Markou, E. (2019, November 7). Access Your Data in Google BigQuery with Python and R. Retrieved from <https://www.blendo.co/blog/access-data-google-bigquery-python-r/>
- Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., & Vassilakis, T. (2010). Dremel: Interactive Analysis of Web-Scale Datasets. Retrieved from <https://storage.googleapis.com/pub-tools-public-publication-data/pdf/36632.pdf>
- Political Advertising on Google. (n.d.). Retrieved from <https://console.cloud.google.com/marketplace/details/transparency-report/google-political-ads?filter=solution-type:dataset&q=google+political&id=0c422e45-5809-4373-b2f4-ce31204c2f4c&pli=1>
- Sato, Kazunori. (2012). An Inside Look at Google BigQuery. Google. Retrieved from <https://cloud.google.com/files/BigQueryTechnicalWP.pdf>
- The Home Depot Hub | Customers | Google Cloud | Google Cloud. (n.d.). Retrieved from <https://cloud.google.com/customers/featured/the-home-depot>
- What is BigQuery Data Warehouse? (n.d.). Retrieved from <https://www.sisense.com/glossary/google-big-query-data-warehouse/>