

Exploring the Public Cryptocurrency Datasets Available in BigQuery

GSP665



Google Cloud Self-Paced Labs

Overview

This lab lets you explore the six cryptocurrency blockchain datasets released publically in BigQuery. This was introduced in the blog post [Introducing six new cryptocurrencies in BigQuery Public Datasets—and how to analyze them](#)

This is a challenge lab. This means you will not be given all the parts to tasks that are marked. You must have working knowledge of SQL.

The following is from the blog post, please read to understand the background of the lab:

Since they emerged in 2009, cryptocurrencies have experienced their share of volatility—and are a continual source of fascination. In the past year, as part of the BigQuery Public Datasets program, Google Cloud released datasets consisting of the blockchain transaction history for Bitcoin and Ethereum, to help you better understand cryptocurrency. Today, we're releasing an additional six cryptocurrency blockchains.

We are also including a set of queries and views that map all blockchain datasets to a double-entry book data structure that enables multi-chain meta-analyses, as well as integration with conventional financial record processing systems.

Additional blockchain datasets

The six cryptocurrency blockchain datasets we're releasing today are Bitcoin Cash, Dash, Dogecoin, Ethereum Classic, Litecoin, and Zcash.

Five of these datasets, along with the previously published Bitcoin dataset now follow a common schema that enables comparative analyses. We are releasing this group of Bitcoin-like datasets (Bitcoin, Bitcoin Cash, Dash, Dogecoin, Litecoin and Zcash) together because they all have similar implementations, i.e., their source code is derived from Bitcoin's. Similarly, we're also releasing the Ethereum Classic dataset alongside the previously published Ethereum dataset, and Ethereum Classic is also using the same common schema.

A unified data ingest architecture

All datasets update every 24 hours via a common codebase, the Blockchain ETL ingestion framework (built with Cloud Composer, previously described here), to accommodate a variety of Bitcoin-like cryptocurrencies. While this means higher latency for loading Bitcoin blocks into BigQuery, it also means that:

We are able to ingest additional BigQuery datasets with less effort, meaning additional datasets can be onboarded more quickly in the future. We can implement a low-latency loading solution once that can be used to enable real-time streaming transactions for all blockchains.

Unified schema and views

Since we provided the original Bitcoin dataset last year, we've learned how users want to access data, and restructured the dataset accordingly. Some of these changes address performance and convenience concerns, yielding faster and lower cost queries (commonly accessed nested data are denormalized; each table is partitioned by time).

We've also included more data, such as script op-codes. Most Bitcoin transactions describe transfers of value not simply as a debit/credit pair, but rather as a series of functions that describe both simple transfers and more complex transactions.

Having these scripts available for Bitcoin-like datasets enables more advanced analyses similar to this smart contract analyzer that Tomasz Kolinko recently built on top of the BigQuery Ethereum dataset. For example, we can now identify and report on patterns of activity involving multi-signature wallets. This is particularly important for analyzing privacy-oriented cryptocurrencies like Zcash.

For analytics interoperability, we designed a unified schema that allows all Bitcoin-like datasets to share queries. To further interoperate with Ethereum and ERC-20 token transactions, we also created some views that abstract the blockchain ledger to be presented as a double-entry accounting ledger.

What you'll do

- Explore and perform a simple SQL query on the BigQuery public cryptocurrency datasets.
- Verify that the BigQuery public cryptocurrency dataset is correct.
- Perform a complex query that calculates the gini coefficient for Litecoin and Dash, on a week by week basis.
- Use your SQL skills to complete two interesting queries on the bitcoin dataset.

Prerequisites

To get the best out of this lab you should have some:

- Familiarity with cryptocurrencies.
- Ability to write basic SQL statements.

Task 0: Setup

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.


Note: If you are using a Pixelbook, open an Incognito window to run this lab.


How to start your lab and sign in to the Google Cloud Console


1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.

[Open Google Console](#)

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)


Username
google2727032_student@qwiklabs.n 

Password
k68CZxsxMZ 

GCP Project ID
qwiklabs-gcp-4fbfecac8667e457 

[New to labs? View our introductory video!](#)


- Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Sign in
Use your Google Account


[Forgot email?](#)


Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another**


Choose an account

 Your.Email@gmail.com

 google1381214_student@qwiklabs.net
Signed out

 **Use another account**

Account.

3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

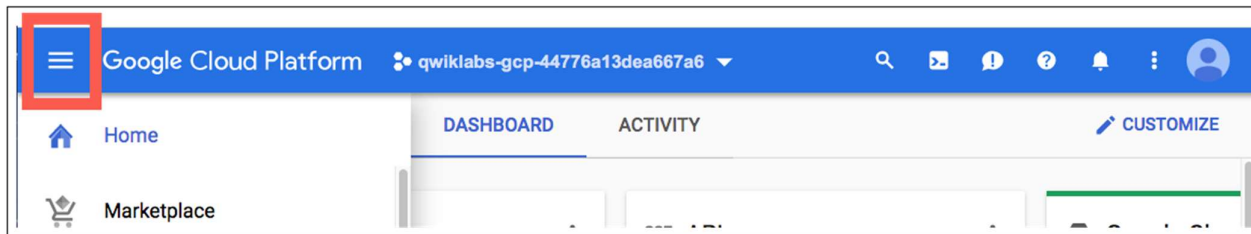
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



Task 1: View the cryptocurrencies in the public dataset

1. Open **Navigation Menu > BigQuery**.
2. Click **+ ADD DATA > Explore public datasets** .
3. In **Search for solutions**, type `bitcoin` and press **enter**.
4. Click **Bitcoin Cash Cryptocurrency Dataset**.
5. Click **VIEW DATASET**.

A new tab will open with BigQuery, and you should be on the `bigquery-public-data:crypto_bitcoin_cash` dataset.

All the public datasets are visible.

6. In **Type to search**, type `crypto`.

Only the public datasets starting with `crypto` will be displayed. Expand the datasets so you can see they all share the same structure. This makes performing queries across the different cryptocurrencies easy as the tables, views, and fields are identical in each cryptocurrency dataset.

Task 2: Perform a simple query

In this task you will view the famous 10,000 bitcoin pizza purchase transaction. You can [read about what happened](#).

1. Copy and paste this query into the query window and then press **Run**.

```
SELECT * FROM `bigquery-public-data.crypto_bitcoin.transactions` as transactions WHERE transactions.hash = 'a1075db55d416d3ca199f55b6084e2115b9345e16c5cf302fc80e9d5fbf5d48d'
```

You can see the raw data returned.

Points to note:

- Values for input, output and fee are in satoshis. The [satoshi](#) is currently the smallest unit of the bitcoin currency recorded on the block chain. It is a one hundred millionth of a single bitcoin (0.00000001 BTC).
- The sent amount is via multiple inputs all coming from the same address.
- The output amount is sent as a single transaction to the one address.

Return the 10,000 BTC pizza purchase transaction.

Check my progress

Task 3: Validate the data

In this task you will check that you can access the cryptocurrency datasets by performing simple validation queries on two cryptocurrencies.

Double-entry book query of Bitcoin Cash

To motivate an initial exploration of these new datasets, let's start with a simple example, comparing the way to query both payments and receipts across multiple cryptocurrencies. This comparison is the simplest way to verify that a cryptocurrency is operating as intended, and at least operationally, is a mathematically correct store of value.

1. Copy and paste this query into the query window and then press **Run**.

```
-- SQL source from https://cloud.google.com/blog/products/data-analytics/introducing-
six-new-cryptocurrencies-in-bigquery-public-datasets-and-how-to-analyze-them
WITH double_entry_book AS (
  -- debits
  SELECT
    array_to_string(inputs.addresses, ",") as address
  , inputs.type
  , -inputs.value as value
  FROM `bigquery-public-data.crypto_bitcoin.inputs` as inputs
  UNION ALL
  -- credits
  SELECT
    array_to_string(outputs.addresses, ",") as address
  , outputs.type
  , outputs.value as value
  FROM `bigquery-public-data.crypto_bitcoin.outputs` as outputs
)
SELECT
  address
, type
, sum(value) as balance
FROM double_entry_book
GROUP BY 1,2
ORDER BY balance DESC
LIMIT 100
```

Verify the Bitcoin values returned are accurate

2. In another browser tab open the website <https://www.blockchain.com/search?> It will say Oops! - this is ok.
3. From the **All Blockchains** dropdown, under Mainnet select **Bitcoin**.
4. Copy the top address returned in BigQuery results.

5. Paste into the search box and click **Search**.
6. Check the final balance returned - it should be the same as the balance listed in BigQuery results for that address.

NOTE: If it is out, that it will either be due to a rounding difference in the adding (small fraction), or that the 24 hour delay in the dataset is the issue.

Double-entry book query of Dogecoin

7. Modify the query to use dogecoin, and run the query.
Calculate the balance for dogecoin.

Check my progress

Verify the Dogecoin values returned are accurate

8. In another browser tab open the website <https://dogechain.info/>
9. Copy the top address returned in BigQuery results.
10. Paste into the search box and click search.
11. Note the balance returned, it should be the same as the balance listed in BigQuery results for that address.

NOTE: If it is out, that it will either be due to a rounding difference in the adding (small fraction), or that the 24 hour delay in the dataset is the issue.

Note that the only difference between them is the name of the data location. You can swap in Bitcoin Cash, Dash, Litecoin, and Zcash in a similar fashion.

Task 4: Plot the Gini coefficient for cryptocurrency

Beyond quality control and auditing applications, presenting cryptocurrency in a traditional format enables integration with other financial data management systems. As an example, let's consider a common economic measure, the [Gini Coefficient](#). In the field of macroeconomics, the Gini Coefficient is a member of a family of [econometric measures of wealth inequality](#). Values range between 0.0 and 1.0, with completely distributed wealth (all members have the same amount) mapping to a value of 0.0 and completely accumulated wealth (one member has everything) mapping to 1.0.

Typically, the Gini Coefficient is estimated for a specific country's economy based on data sampling or imputation. For crypto-economies, we have complete transparency of the data at the highest possible resolution.

In addition to data transparency, one of the purported benefits of cryptocurrencies is that they allow the implementation of money to more closely resemble the implementation of digital information. It follows that a fully digitized money network will come to resemble the internet, with reduced transactional friction and fewer barriers that impede capital flow. Frequently, implicit in this narrative is that capital will distribute more equally. But we don't always observe that particular outcome, and the crypto-assets presented here display a broad spectrum of distribution patterns over time. You can read more about using the Gini coefficient to reason about crypto-economic network performance in [Quantifying Decentralization](#).

To set a baseline to interpret our findings, consider how resources are distributed in traditional, non-crypto economies. According to a [World Bank analysis in 2013](#), recent Gini coefficients for world economies have a mean value of 39.6 (with a standard deviation of 9.6).

Create the query

In this next query you will calculate the gini coefficient of dash, on a week-by-week basis. This query will take about a minute to execute. Once you have the data you can easily visualize the graph and compare it with the one generated for the source article.

1. Copy and paste this query into the query window and then press **Run**.

```
-- SQL source from https://gist.github.com/allenday/1500cc268f24ae89b7adfc25c74967b0
WITH double_entry_book AS (
  -- debits
  SELECT
    array_to_string(inputs.addresses, ",") as address
  , inputs.type
  , -inputs.value as value
  , block.timestamp
  FROM `bigquery-public-data.crypto_dash.inputs` as inputs

  UNION ALL

  -- credits
```

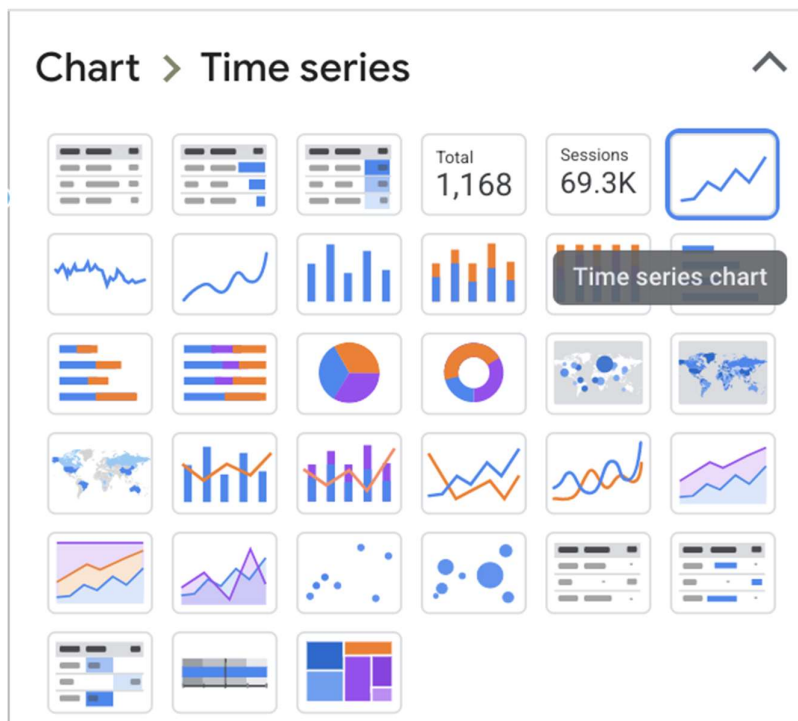
```

SELECT
  array_to_string(outputs.addresses, ",") as address
, outputs.type
, outputs.value as value
, block_timestamp
FROM `bigquery-public-data.crypto_dash.outputs` as outputs
)
,double_entry_book_by_date as (
  select
    date(block_timestamp) as date,
    address,
    sum(value / POWER(10,0)) as value
  from double_entry_book
  group by address, date
)
,daily_balances_with_gaps as (
  select
    address,
    date,
    sum(value) over (partition by address order by date) as balance,
    lead(date, 1, current_date()) over (partition by address order by date) as
next_date
  from double_entry_book_by_date
)
,calendar as (
  select date from unnest(generate_date_array('2009-01-12', current_date())) as date
)
,daily_balances as (
  select address, calendar.date, balance
  from daily_balances_with_gaps
  join calendar on daily_balances_with_gaps.date <= calendar.date and calendar.date <
daily_balances_with_gaps.next_date
)
,supply as (
  select
    date,
    sum(balance) as daily_supply
  from daily_balances
  group by date
)
,ranked_daily_balances as (
  select
    daily_balances.date,
    balance,
    row_number() over (partition by daily_balances.date order by balance desc) as
rank
  from daily_balances
  join supply on daily_balances.date = supply.date
  where safe_divide(balance, daily_supply) >= 0.0001
  ORDER BY safe_divide(balance, daily_supply) DESC
)
select
  date,
  -- (1 - 2B) https://en.wikipedia.org/wiki/Gini\_coefficient
  1 - 2 * sum((balance * (rank - 1) + balance / 2)) / count(*) / sum(balance) as gini
from ranked_daily_balances
group by date
order by date asc

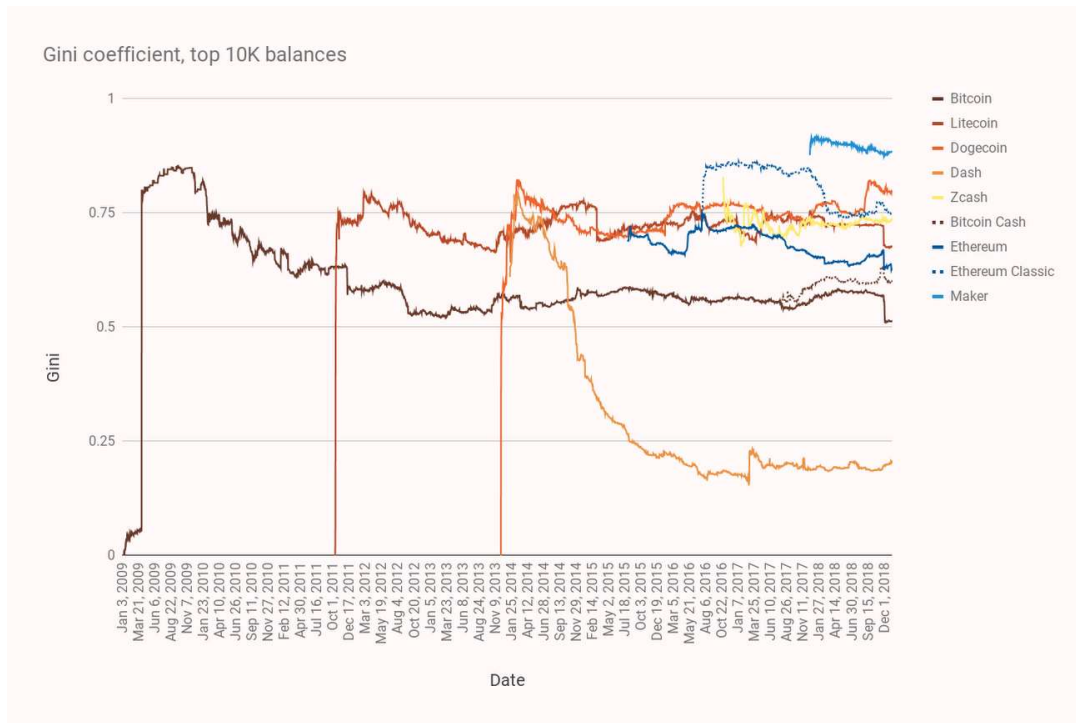
```

Use Data Studio to visualize the query

2. In the Query Results ribbon, click **EXPLORE DATA > Explore with Data Studio** and authorize BigQuery.
3. Click **Time Series Chart**.



4. Change the metric to **gini**.
5. Compare the graph with the original source article graph (reproduced below). In the chart below you are looking for the brown line. Starting in December 2019, see how it's changed since then.



Generate the gini coefficient for litecoin

6. Perform the same steps as you did for the Dash cryptocurrency to get the gini coefficient, but this time perform it for Litecoin.

Task 5: Explore two famous cryptocurrency events

There is a [blog post on the Blockchain site](#) that describes a number of memorable events in bitcoin. You will perform two queries on the bitcoin public dataset to retrieve the data in a couple of those events.

November 22, 2013: This bitcoin transaction sparks mystery and speculation

In the fall of 2013, a 194,993 bitcoin transaction hit the network, which caused many to wonder who was behind this very large transaction. The value at the time of the transaction in USD was over \$149 million. CoinDesk wrote "unsurprisingly, a transaction of that size has prompted the bitcoin community to do some analysis and detective work. The transaction involved a large number of sending addresses, with some of them from blocks mined in February 2010 or even earlier, prompting excited speculation they might be from Satoshi Nakamoto, bitcoin's absent (and likely pseudonymous) founder."

1. Write a SQL query to find the `transaction_hash` for transfer of 194993 bitcoins (BTC) and write that data to a table called 51 in the dataset lab. Below is SQL to help you get started:

```
CREATE OR REPLACE TABLE lab.51 (transaction_hash STRING) as
SELECT -- write the rest of the select statement (remember to use where) ...
```

This will require you to understand the structure of the **transactions** table, or you can use the **outputs** view. Your query must return the transaction hash, other fields are not important.

Hint: The unique amount makes this easy to query for that value. You will need to convert the value to satoshis. Below is

Store the transaction hash of the large mystery transfer of 194993 BTC in the table 51 inside the lab dataset.

Check my progress

After all this time, what is the balance of the address that purchased the two pizzas for 10,000 BTC?

2. Write a SQL query to write the balance for the account that paid 10,000 BTC for the two pizzas in 2010, into the table 52 in the lab dataset.

The output must have the balance for the address.

Hint: Modify the query from task 3 to select only rows with the purchaser address (using a WHERE clause). To get the purchaser address you can perform the query from task 2 and copy the inputs address (you do not need to include that query as part of the solution for this activity).

Here is some SQL to start you off:

```
CREATE OR REPLACE TABLE lab.52 (balance NUMERIC) as
WITH double_entry_book AS (
  -- debits
  SELECT
    array_to_string(inputs.addresses, ",") as address
  , -inputs.value as value
  FROM `bigquery-public-data.crypto_bitcoin.inputs` as inputs
  UNION ALL
  -- credits
  SELECT
    array_to_string(outputs.addresses, ",") as address
  , outputs.value as value
  FROM `bigquery-public-data.crypto_bitcoin.outputs` as outputs
)
SELECT -- write the rest of the select statement (remember to use where) ...
```

Store the balance of the pizza purchase address in the table 52 inside the lab dataset.

Check my progress

Congratulations!

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated: February 01, 2021

Lab Last Tested: February 01, 2021

Copyright 2021 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.

solution : <https://github.com/GirishSharma5956/---SQL-source-from-https-cloud.google.com-blog-product...-CREATE-OR-REPLACE-TABLE-lab.52-balance-/blob/master/httpsgithub.comGirishSharma5956---SQL-source-from-https-cloud.google.com-blog-product...-CREATE-OR-REPLACE-TABLE-lab.52-balance.txt>