WORKING WITH BIG DATA IN POLITICAL SCIENCE

An introduction to using Google’s BigQuery with R

WHAT PROBLEM ARE WE SOLVING?

- Truly big data
  - Need to work outside of your computer’s memory
- Need a tool that helps you store, manage, and work with these data
- Don’t want to learn a whole new language

BIGQUERY

What is Google’s BigQuery?

- Serverless data warehouse
- Built-in query engine

bigrquery

R package that allows you to work with data stored in BigQuery through R.

To install:

```r
install.packages(c("bigrquery", "DBI", "dplyr"))
```

To load into your current session:

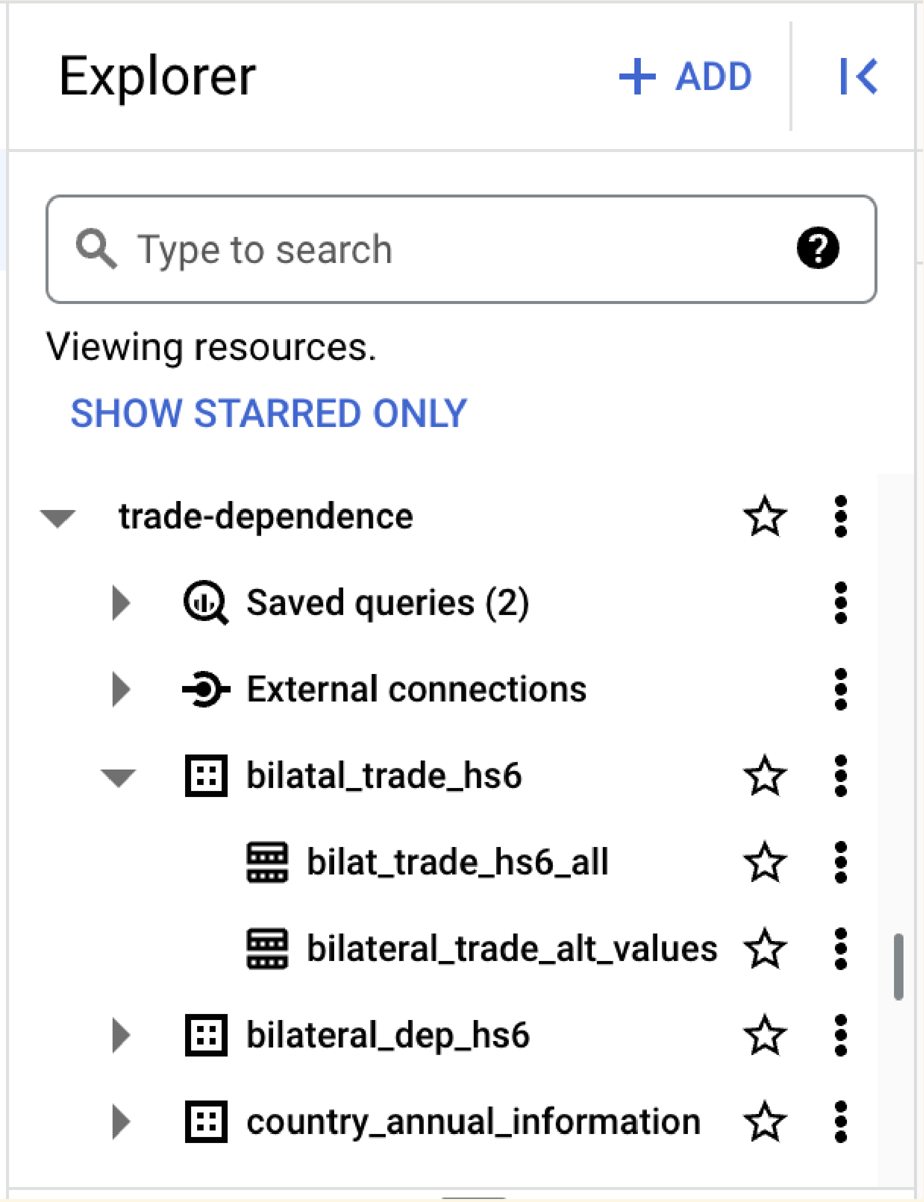
```r
library(bigrquery)
library(DBI)
library(dplyr)
```

HOW TO STORE YOUR DATA

BigQuery is hierarchical:

- **Tables** are stored in:
  - **Datasets**, which are stored in:
    - **Projects.**

HOW TO STORE YOUR DATA



Explorer

+ ADD

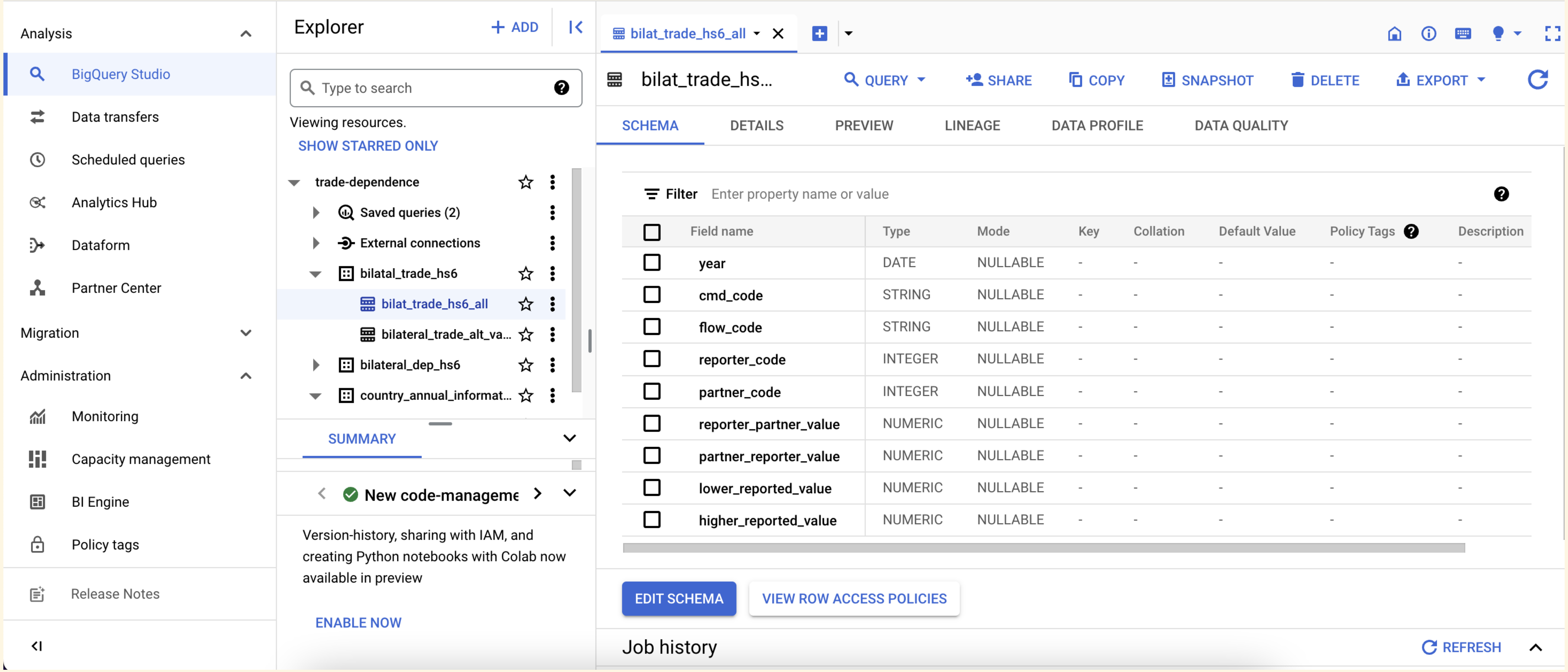
Type to search

Viewing resources.

SHOW STARRED ONLY

- trade-dependence
  - Saved queries (2)
  - External connections
  - bilatal_trade_hs6
    - bilat_trade_hs6_all
    - bilateral_trade_alt_values
  - bilateral_dep_hs6
  - country_annual_information

TABLES

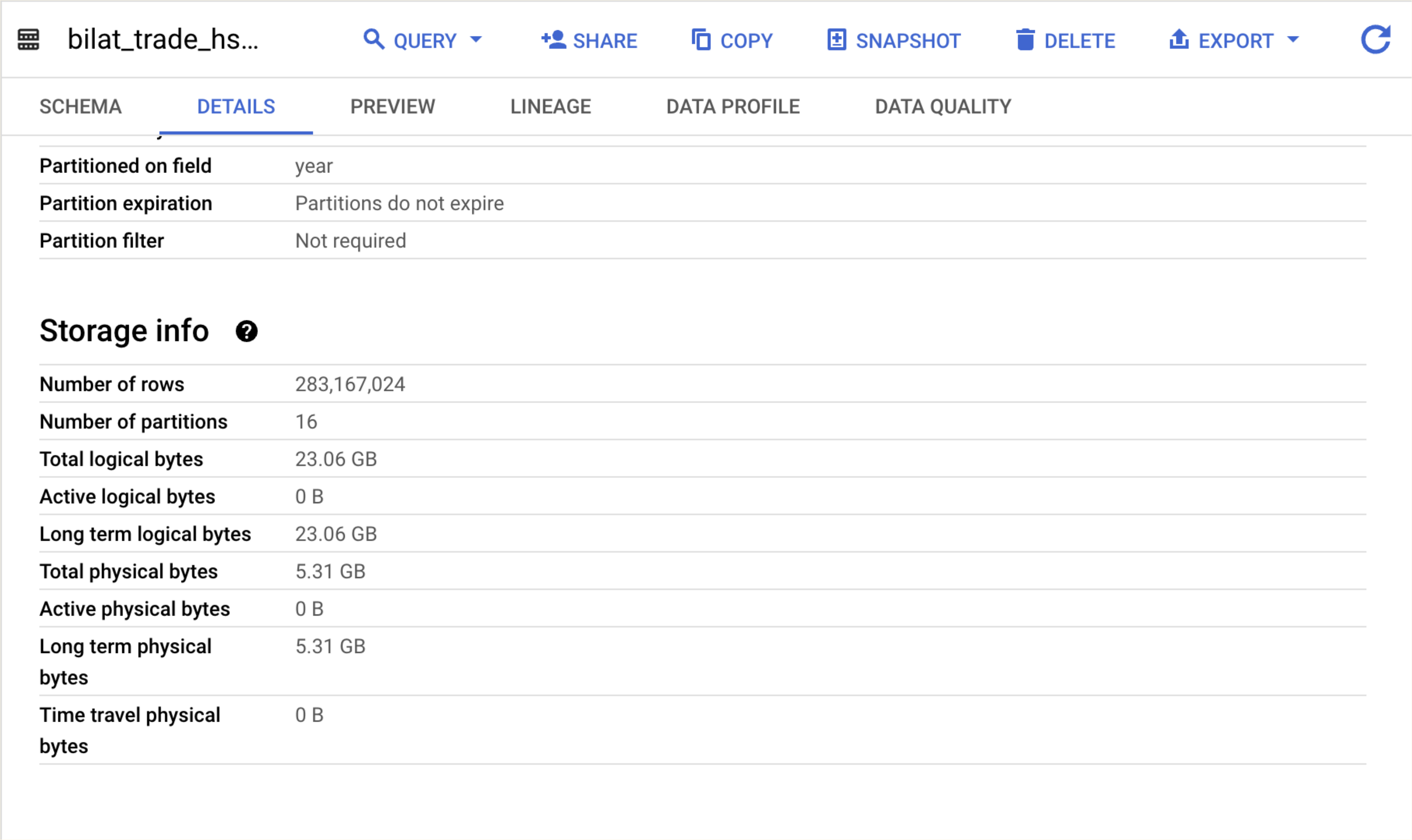


Version-history, sharing with IAM, and creating Python notebooks with Colab now available in preview

ENABLE NOW

□	Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
□	year	DATE	NULLABLE	-	-	-	-	-
□	cmd_code	STRING	NULLABLE	-	-	-	-	-
□	flow_code	STRING	NULLABLE	-	-	-	-	-
□	reporter_code	INTEGER	NULLABLE	-	-	-	-	-
□	partner_code	INTEGER	NULLABLE	-	-	-	-	-
□	reporter_partner_value	NUMERIC	NULLABLE	-	-	-	-	-
□	partner_reporter_value	NUMERIC	NULLABLE	-	-	-	-	-
□	lower_reported_value	NUMERIC	NULLABLE	-	-	-	-	-
□	higher_reported_value	NUMERIC	NULLABLE	-	-	-	-	-

TABLES

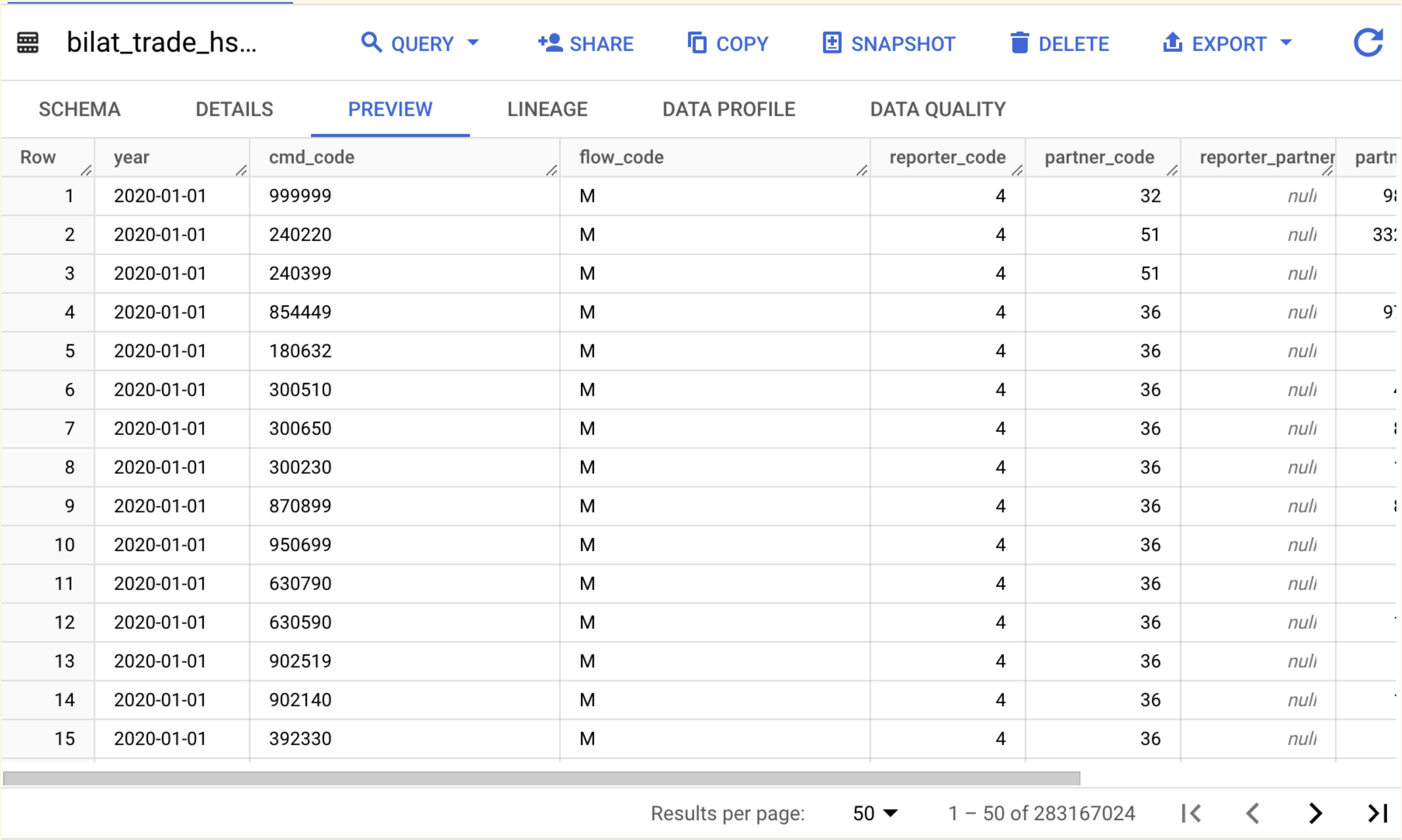


Partitioned on field	year
Partition expiration	Partitions do not expire
Partition filter	Not required

Storage info

Number of rows	283,167,024
Number of partitions	16
Total logical bytes	23.06 GB
Active logical bytes	0 B
Long term logical bytes	23.06 GB
Total physical bytes	5.31 GB
Active physical bytes	0 B
Long term physical bytes	5.31 GB
Time travel physical bytes	0 B

TABLES



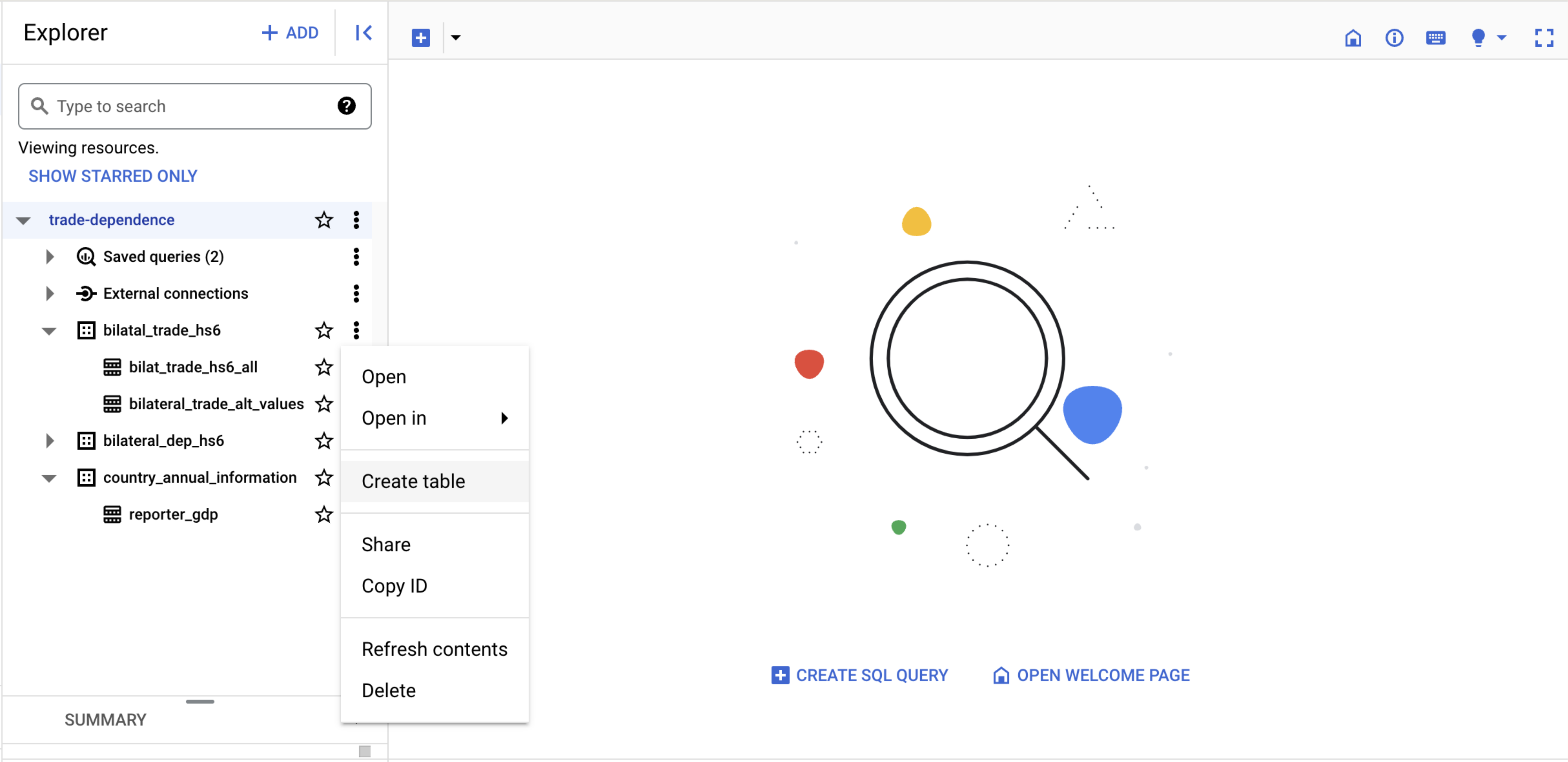
Row	year	cmd_code	flow_code	reporter_code	partner_code	reporter_partner	partn
1	2020-01-01	999999	M	4	32	null	9
2	2020-01-01	240220	M	4	51	null	33
3	2020-01-01	240399	M	4	51	null	
4	2020-01-01	854449	M	4	36	null	9
5	2020-01-01	180632	M	4	36	null	
6	2020-01-01	300510	M	4	36	null	[illegible]
7	2020-01-01	300650	M	4	36	null	[illegible]
8	2020-01-01	300230	M	4	36	null	[illegible]
9	2020-01-01	870899	M	4	36	null	[illegible]
10	2020-01-01	950699	M	4	36	null	
11	2020-01-01	630790	M	4	36	null	
12	2020-01-01	630590	M	4	36	null	[illegible]
13	2020-01-01	902519	M	4	36	null	
14	2020-01-01	902140	M	4	36	null	[illegible]
15	2020-01-01	392330	M	4	36	null	

Results per page: 50 1 – 50 of 283167024

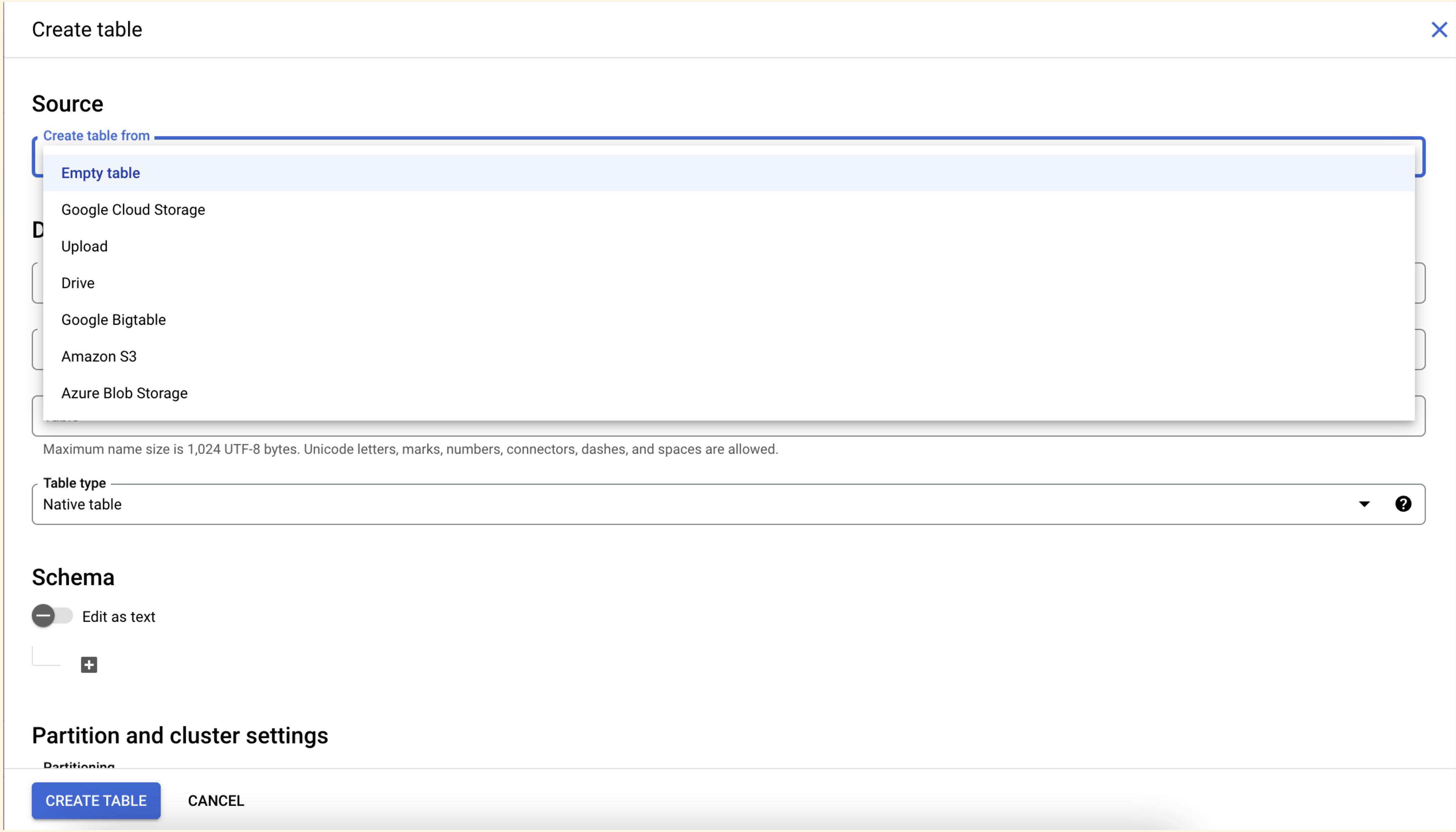
UPLOADING YOUR DATA

We will step through uploading your data to BigQuery

UPLOADING DATA MANUALLY



UPLOADING DATA MANUALLY



Create table

Source

Create table from

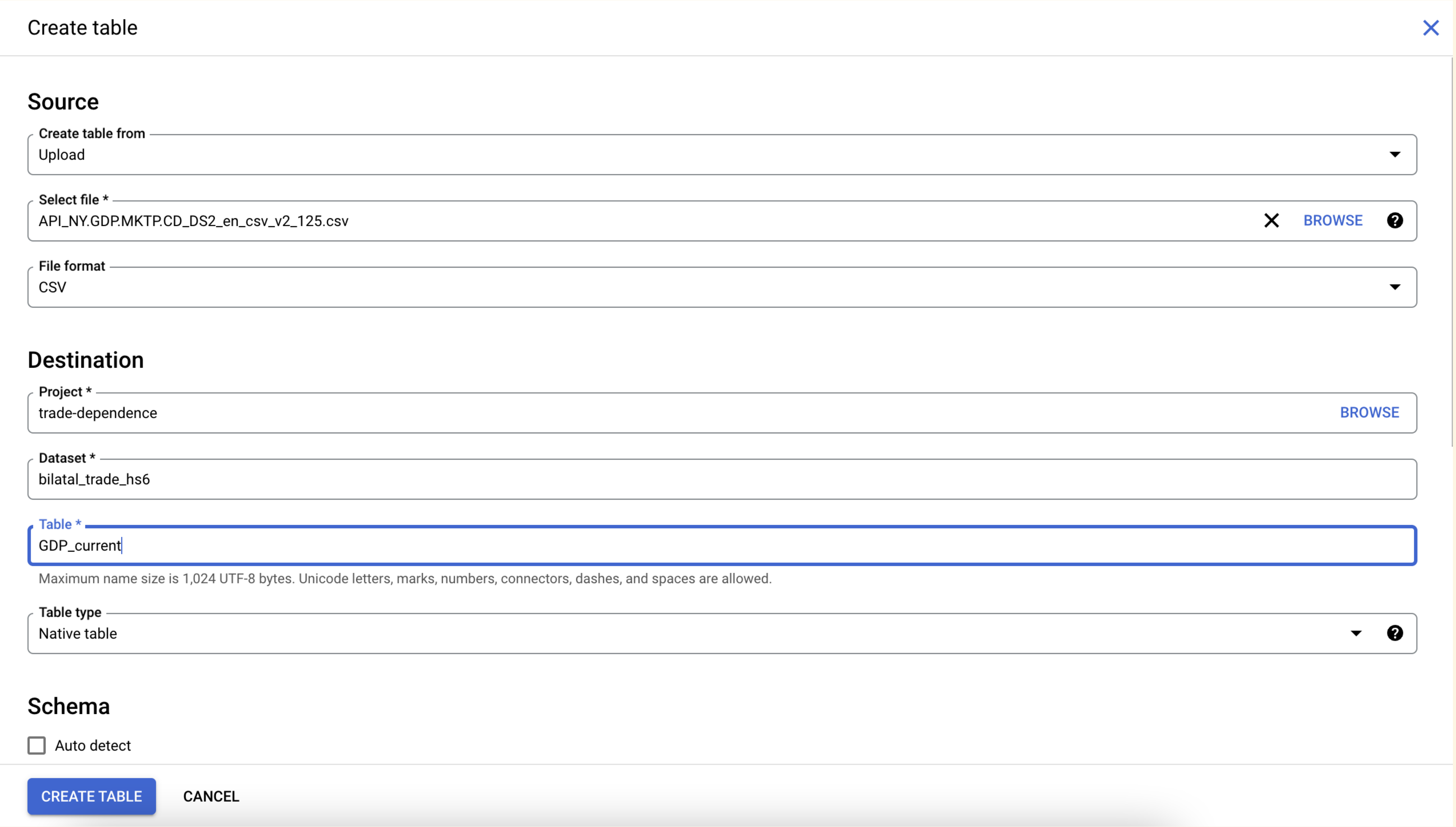
Table type: Native table

Maximum name size is 1,024 UTF-8 bytes. Unicode letters, marks, numbers, connectors, dashes, and spaces are allowed.

Schema

Partition and cluster settings

UPLOADING DATA MANUALLY



Create table

Source

Create table from: Upload

Select file *: API_NY.GDP.MKTP.CD_DS2_en_csv_v2_125.csv

File format: CSV

Destination

Project *: trade-dependence

Dataset *: bilatal_trade_hs6

Table *: GDP_current

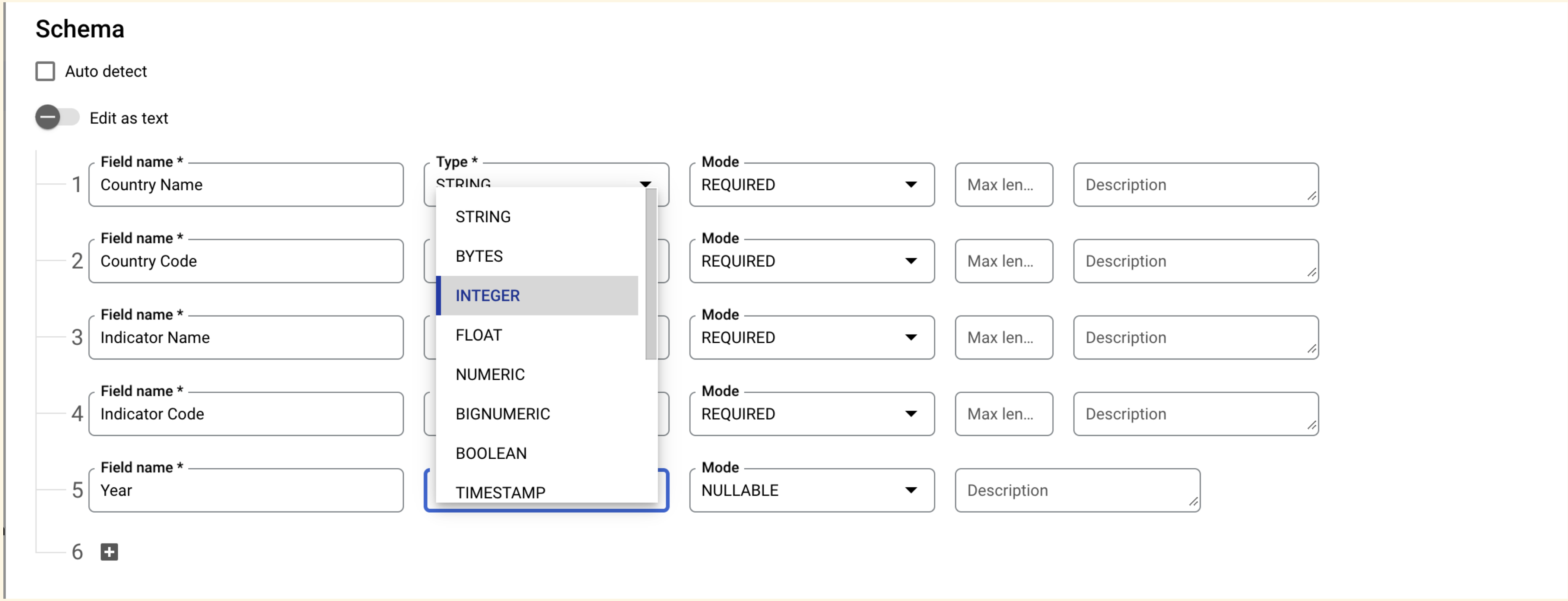
Maximum name size is 1,024 UTF-8 bytes. Unicode letters, marks, numbers, connectors, dashes, and spaces are allowed.

Table type: Native table

Schema

□ Auto detect

UPLOADING DATA MANUALLY



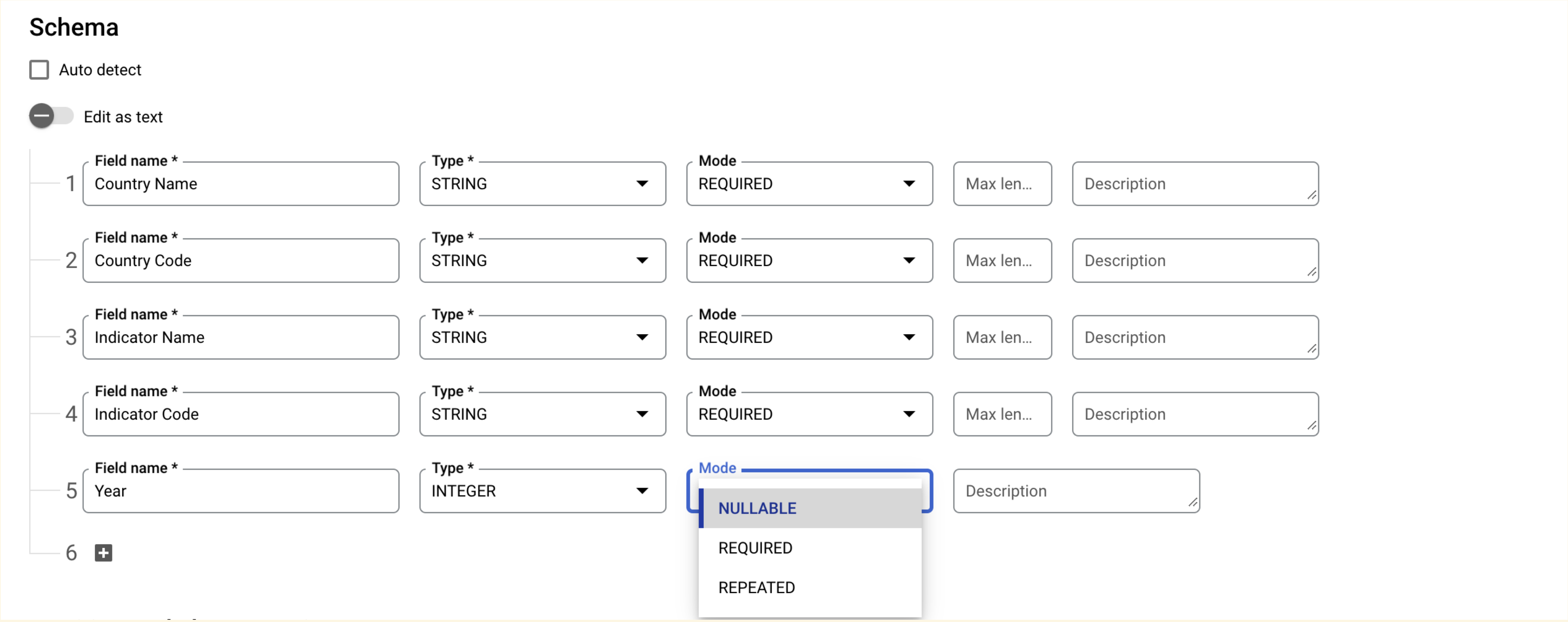
Schema

□ Auto detect

Edit as text

	Field name *	Type *	Mode		
1	Country Name	STRING	REQUIRED	Max len...	Description
2	Country Code		REQUIRED	Max len...	Description
3	Indicator Name		REQUIRED	Max len...	Description
4	Indicator Code		REQUIRED	Max len...	Description
5	Year		NULLABLE	Description	
6					

UPLOADING DATA MANUALLY



Schema

□ Auto detect

Edit as text

	Field name *	Type *	Mode		
1	Country Name	STRING	REQUIRED	Max len…	Description
2	Country Code	STRING	REQUIRED	Max len…	Description
3	Indicator Name	STRING	REQUIRED	Max len…	Description
4	Indicator Code	STRING	REQUIRED	Max len…	Description
5	Year	INTEGER		Description	
6	⊞				

UPLOADING DATA USING `bigrquery`

```r
library(bigrquery)
library(DBI)
library(dplyr)
```

List useful information:

```r
selected_project <- "trade-dependence"
selected_dataset <- "bilatal_trade_hs6"
```

UPLOADING DATA USING `bigrquery`

First, check that the table does not already exist:

```r
bq_gdp_current <- bq_table(project = selected_project,
                           dataset = selected_dataset,
                           table = "gdp_current")

bq_table_exists(bq_gdp_current)
```

```
[1] FALSE
```

The first time you do this, you will need to authorize `bigrquery`'s access to your

UPLOADING DATA USING bigrquery

Next, create the (empty) table:

```r
bq_table_create(
  bq_gdp_current,
  fields = gdp_current,
  friendly_name = "GDP (current USD)",
  description = "The data was extracted from the World Bank."
)

bq_table_exists(bq_gdp_current)
```

```
[1] TRUE
```

UPLOADING DATA USING `bigrquery`

Next, upload your data to that empty table:

```r
bq_table_upload(bq_gdp_current, gdp_df)
```

WORKING WITH YOUR DATA

We will now step through how to work with big data out of your computer's memory

STARTING IN R WITH `dplyr`

Let's collect data on Australia's GDP.

STARTING IN R WITH dplyr

These data are stored in the `trade-dependence` project and the `country_annual_information` dataset.

```r
selected_project <- "trade-dependence"
selected_dataset <- "country_annual_information"

con <- dbConnect(
  bigrquery::bigquery(),
  project = selected_project,
  dataset = selected_dataset,
  billing = selected_project
)

con
```

```
<BigQueryConnection>
  Dataset: trade-dependence.country_annual_information
  Billing: trade-dependence
```

STARTING IN R WITH dplyr

Create the connection to the reporter_gdp table:

```r
gdp_df <- tbl(con, "reporter_gdp")
gdp_df
```

```
# Source:   table<reporter_gdp> [?? x 3]
# Database: BigQueryConnection
   year       reporter_code reporter_gdp_current
   <date>             <int>                <dbl>
 1 2003-01-01            92                   NA
 2 2003-01-01           136                   NA
 3 2003-01-01           531                   NA
 4 2003-01-01           292                   NA
 5 2003-01-01           408                   NA
 6 2003-01-01            NA                   NA
 7 2003-01-01           520                   NA
 8 2003-01-01           534                   NA
 9 2003-01-01           706                   NA
10 2003-01-01           728                   NA
# ℹ more rows
```

STARTING IN R WITH dplyr

Query that table:

```r
aus_gdp <- gdp_df |>
  filter(reporter_code == 36) |>
  collect()

aus_gdp
```

```
# A tibble: 23 × 3
   year       reporter_code reporter_gdp_current
   <date>             <dbl>                <dbl>
 1 2002-01-01            36              3.96e11
 2 2018-01-01            36              1.43e12
 3 2009-01-01            36              9.29e11
 4 2011-01-01            36              1.40e12
 5 2004-01-01            36              6.14e11
 6 2021-01-01            36              1.55e12
 7 2017-01-01            36              1.33e12
 8 2008-01-01            36              1.06e12
 9 2001-01-01            36              3.79e11
10 2012-01-01            36              1.55e12
# ℹ 13 more rows
```

MOVING BETWEEN R AND BIGQUERY

R will write your SQL queries for you:

```
gdp_df |>
  filter(reporter_code == 36) |>
  show_query()
```

```
<SQL>
SELECT `reporter_gdp`.*
FROM `reporter_gdp`
WHERE (`reporter_code` = 36.0)
```

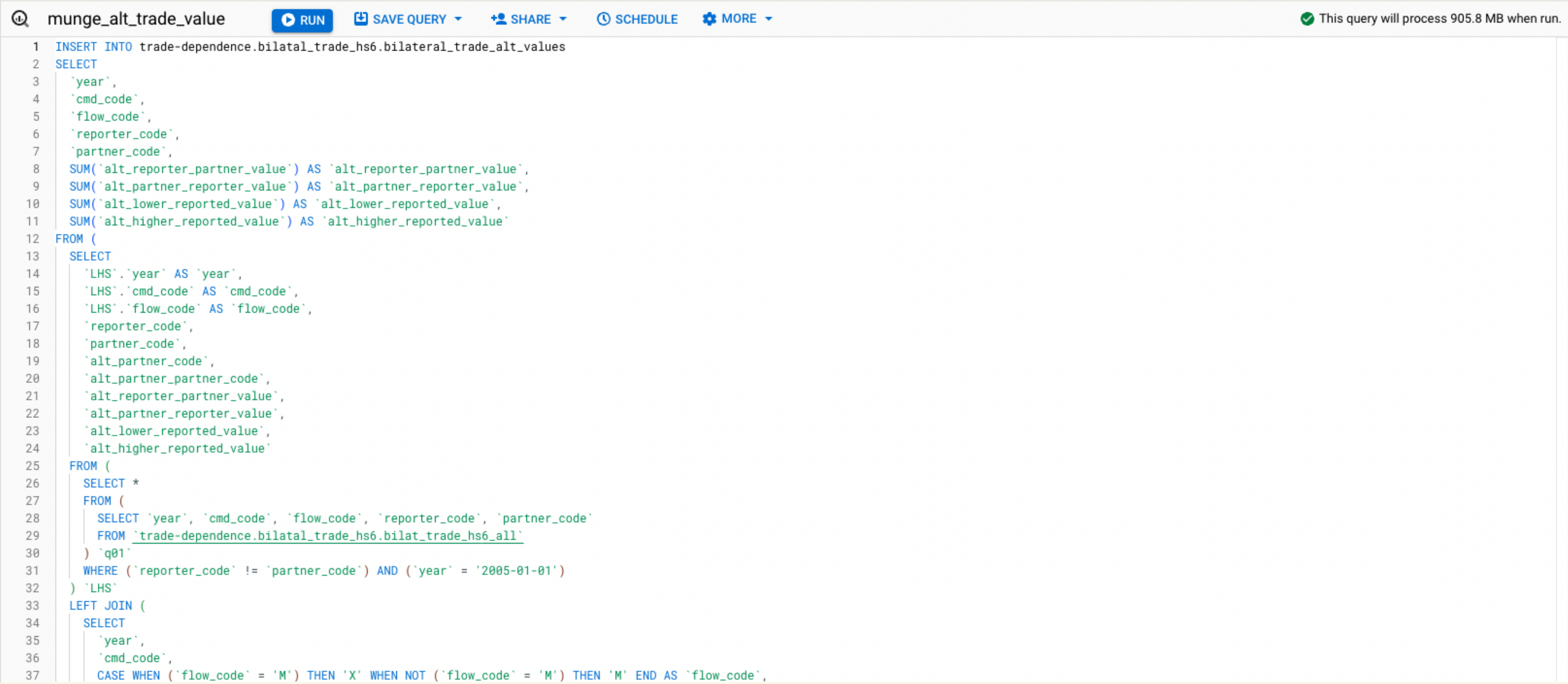
MOVING BETWEEN R AND BIGQUERY

You can perform that query in BigQuery’s in-built query engine:

MOVING BETWEEN R AND BIGQUERY

You can perform that query in BigQuery’s in-built query engine:

PERFORMING COMPLEX QUERIES



This query will process 905.8 MB when run.

```sql
INSERT INTO trade-dependence.bilatal_trade_hs6.bilateral_trade_alt_values
SELECT
  `year`,
  `cmd_code`,
  `flow_code`,
  `reporter_code`,
  `partner_code`,
  SUM(`alt_reporter_partner_value`) AS `alt_reporter_partner_value`,
  SUM(`alt_partner_reporter_value`) AS `alt_partner_reporter_value`,
  SUM(`alt_lower_reported_value`) AS `alt_lower_reported_value`,
  SUM(`alt_higher_reported_value`) AS `alt_higher_reported_value`
FROM (
  SELECT
    `LHS`.`year` AS `year`,
    `LHS`.`cmd_code` AS `cmd_code`,
    `LHS`.`flow_code` AS `flow_code`,
    `reporter_code`,
    `partner_code`,
    `alt_partner_code`,
    `alt_partner_partner_code`,
    `alt_reporter_partner_value`,
    `alt_partner_reporter_value`,
    `alt_lower_reported_value`,
    `alt_higher_reported_value`
  FROM (
    SELECT *
    FROM (
      SELECT `year`, `cmd_code`, `flow_code`, `reporter_code`, `partner_code`
      FROM `trade-dependence.bilatal_trade_hs6.bilat_trade_hs6_all`
    ) `q01`
    WHERE (`reporter_code` != `partner_code`) AND (`year` = '2005-01-01')
  ) `LHS`
  LEFT JOIN (
    SELECT
      `year`,
      `cmd_code`,
      CASE WHEN (`flow_code` = 'M') THEN 'X' WHEN NOT (`flow_code` = 'M') THEN 'M' END AS `flow_code`,
```

NEXT STEPS

- Partitioning and clustering your datasets
  - Great for yearly, country-level data
- Integrated ML model-building