Scaling Data Analysis

Google Cloud Platform Fundamentals: Big Data and Machine Learning

Google Cloud

Version #1.1

Agenda

1 Introduction

2 Fundamentals of GCP

3 Data Analysis on the Cloud

4 Scaling Data Analysis

5 Machine Learning

6 Data Processing Architecture

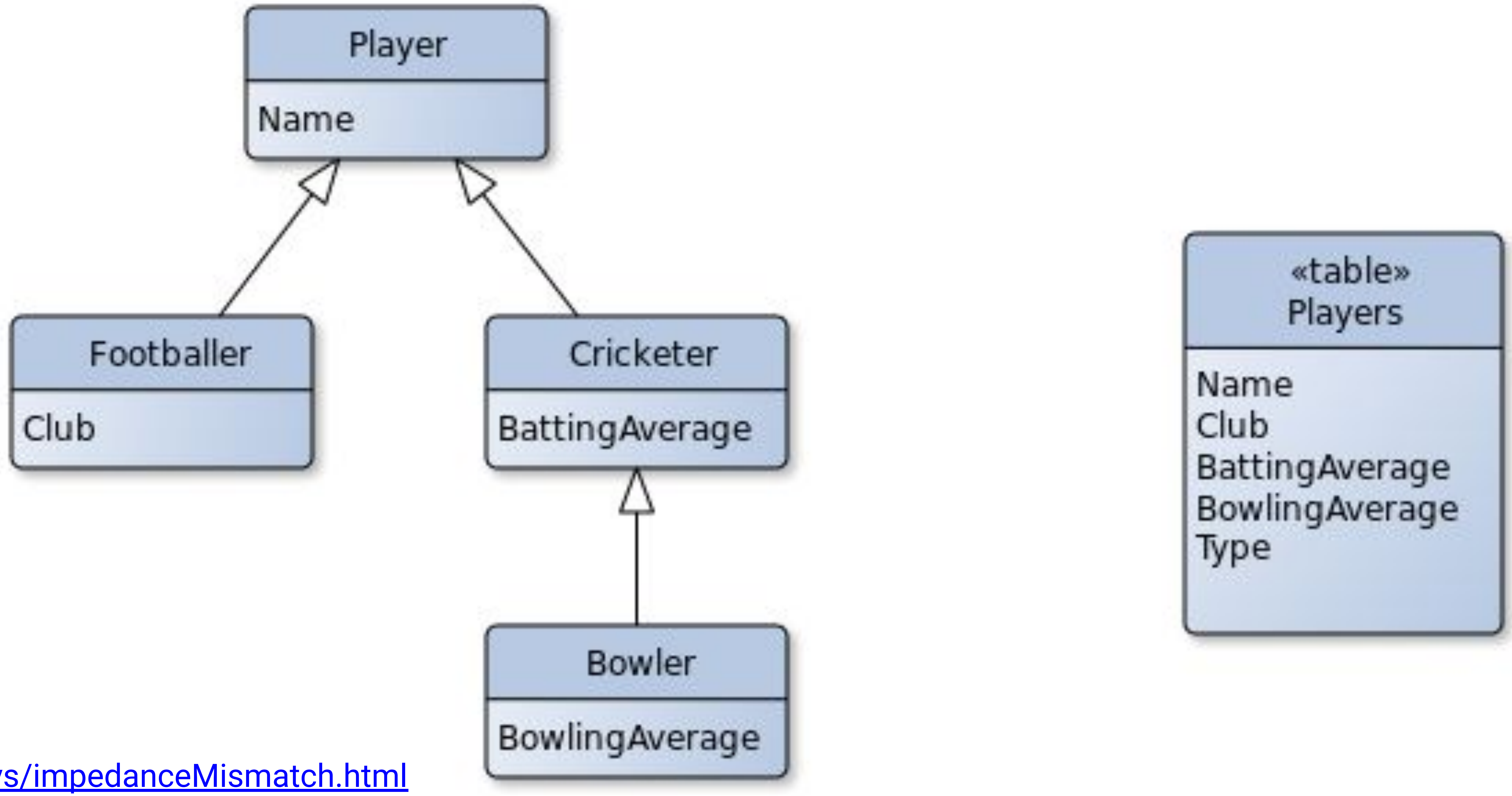
Agenda

Fast random access

Warehouse and interactively query petabytes

Interactive, iterative development + Lab

Relational tables are hard to use from object-oriented programs



http://www.agiledata.org/essays/impedanceMismatch.html
https://en.wikipedia.org/wiki/Object_database

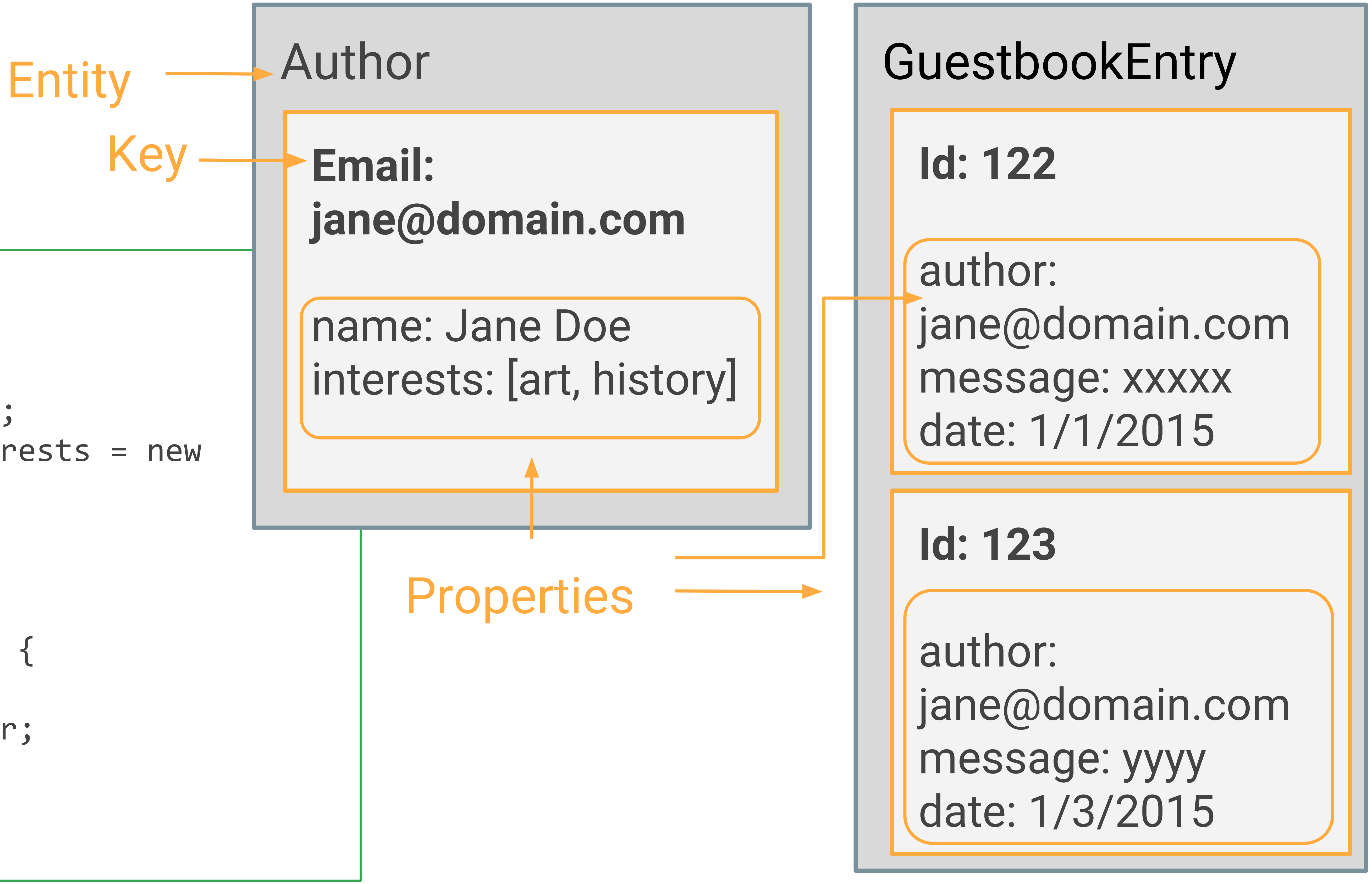
Choose storage based on access pattern

	Cloud Storage	Cloud SQL	Datastore	Bigtable	BigQuery
Capacity	Petabytes +	Gigabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Persistent Hashmap	Key-value(s), HBase API	Relational
Read	Have to copy to local disk	SELECT rows	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row	put object	put row	Batch/stream
Update granularity	An object (a “file”)	Field	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud	Structured data from AppEngine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

Datastore is like a persistent HashMap

```
@Entity
public class Author {
  @Id public String email;
  @Index public String name;
  public List<String>  interests = new
ArrayList<>();
}

@Entity
public class GuestbookEntry {
  @Id public Long id;
  @Parent Key<Author> author;
  public String message;
  @Index public Date date;
}
```



CRUD operations are easily implemented on Datastore

```
@Entity
public class Author {
  @Id public String email;
  @Index public String name;
  public List<String>  interests =
new ArrayList<>();

}
```

```
// CREATE

Author xjin = new Author("xjin@bu.edu", "Ha Jin");

xjin.interests.add("Misty Poetry");

ofy.save().entity(xjin);

// READ

Iterable<Author> authors =
ofy().load().type(Author.class).filter("name", "Ha Jin");

Author jh = ofy.load().type(Author.class).id("xjin@bu.edu").now();

// UPDATE

jh.name = "Jīn Xuěfēi (金雪飞)";

ofy().save().entity(jh).now();

// DELETE

ofy().delete().entity(jh).now();
```

Choose storage product based on access pattern

	Cloud Storage	Cloud SQL	Datastore	Bigtable	BigQuery
Capacity	Petabytes +	Gigabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Persistent Hashmap	Key-value(s), HBase API	Relational
Read	Have to copy to local disk	SELECT rows	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row	put object	put row	Batch/stream
Update granularity	An object (a “file”)	Field	Attribute	Row (write new row instead)	Field
Usage	Store blobs	No-ops SQL database on the cloud	Structured data from AppEngine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

Bigtable is meant for high throughput data where access is primarily for a range of Row Key prefixes

Design row key with most common query in mind

Row key	Column data				
NASDAQ#ZXZZT#1426535612045	MD:SYMBOL: ZXZZT	MD:LASTSALE: 600.58	MD:LASTSIZE: 300	MD:TRADETIME: 1426535612045	MD:EXCHANGE: NASDAQ

Design row key to minimize hotspots

Bigtable is meant for high throughput data where access is primarily for a range of Row Key prefixes

Row key	Column data				
NASDAQ#ZXZZT#1426535612045	MD:SYMBOL: ZXZZT	MD:LASTSALE: 600.58	MD:LASTSIZE: 300	MD:TRADETIME: 1426535612045	MD:EXCHANGE: NASDAQ
...	...	...	...	...	...

Tables should be tall and narrow
Store changes as new rows

Bigtable will automatically compact the table

Short meaningful column names reduce storage and RPC overhead

Design row key with most common query in mind

Column families is a quick way to get some hierarchy

Row key	Column data				
NASDAQ#ZXZZT#1426535612045	MD:SYMBOL: ZXZZT	MD:LASTSALE: 600.58	MD:LASTSIZE: 300	MD:TRADETIME: 1426535612045	MD:EXCHANGE: NASDAQ

Design row key to minimize hotspots

Use short column names
Designed for sparse tables

Google Cloud

Can work with Bigtable using the HBase API

```
import org.apache.hadoop.hbase.*;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.*;

byte[] CF = Bytes.toBytes("MD");  // column family
Connection connection = ConnectionFactory.createConnection(...)
Table table = null;
try {
      table = connection.getTable(TABLE_NAME);
      Put p = new Put(Bytes.toBytes("NASDAQ#GOOG  #1234561234561"));
      p.addColumn(CF, Bytes.toBytes("SYMBOL"), Bytes.toBytes("GOOG"));

      p.addColumn(CF, Bytes.toBytes("LASTSALE"), Bytes.toBytes(742.03d));

      ...
      table.put(p);
} finally {
      if (table != null) table.close();
}
```

Google Cloud

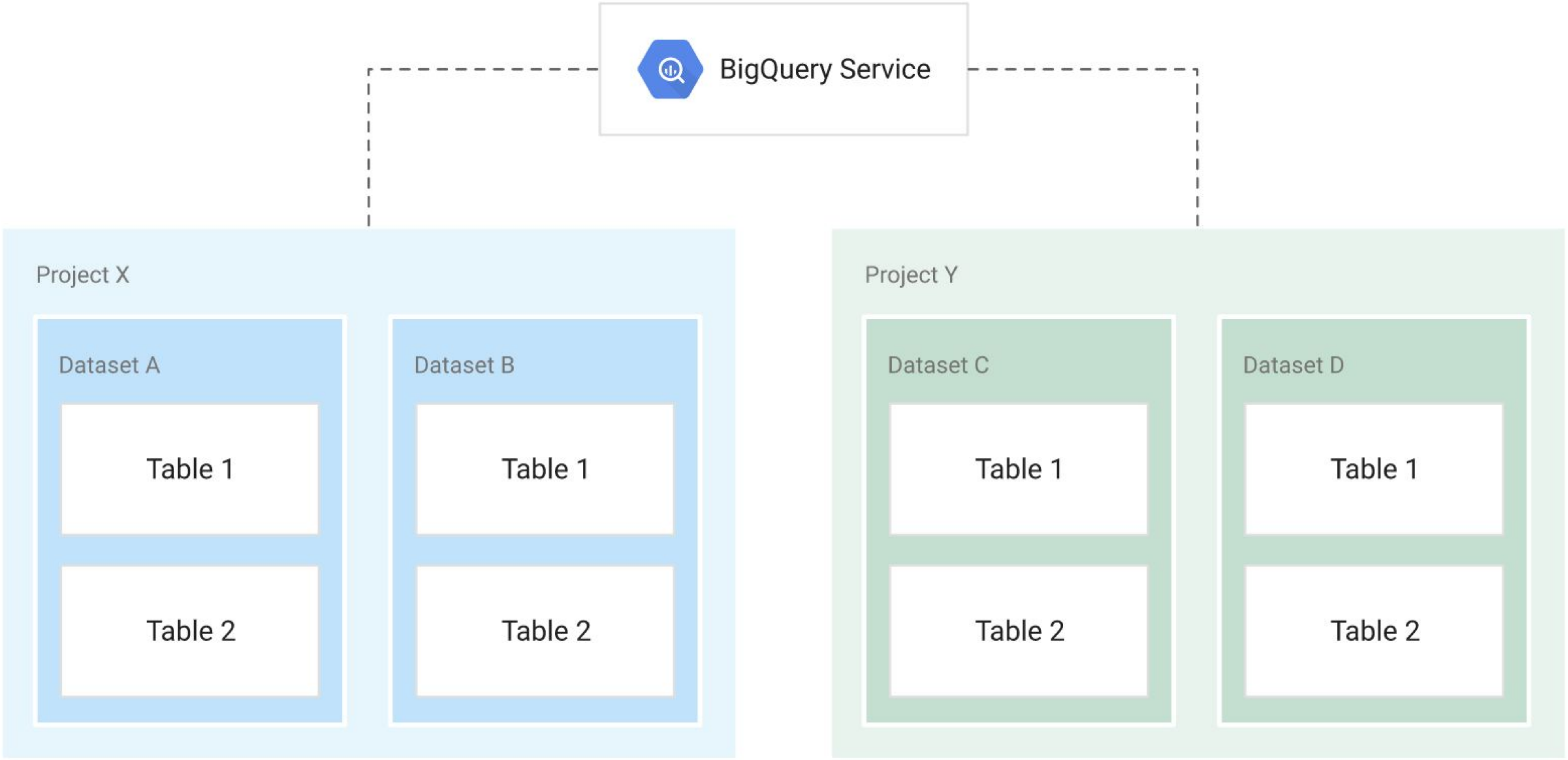
Agenda

Fast random access

Warehouse and interactively query petabytes

Interactive, iterative development + Lab

BigQuery is a fully managed data warehouse that lets you do ad-hoc SQL queries on massive volumes of data



Google Cloud

A demo of BigQuery on a 10 billion-row dataset shows what it is and what it can do

```
#standardsql
SELECT
    language, SUM(views) as views
FROM `bigquery-samples.wikipedia_benchmark.Wiki10B`
WHERE
    title like "%google%"
GROUP by language
ORDER by views DESC
```

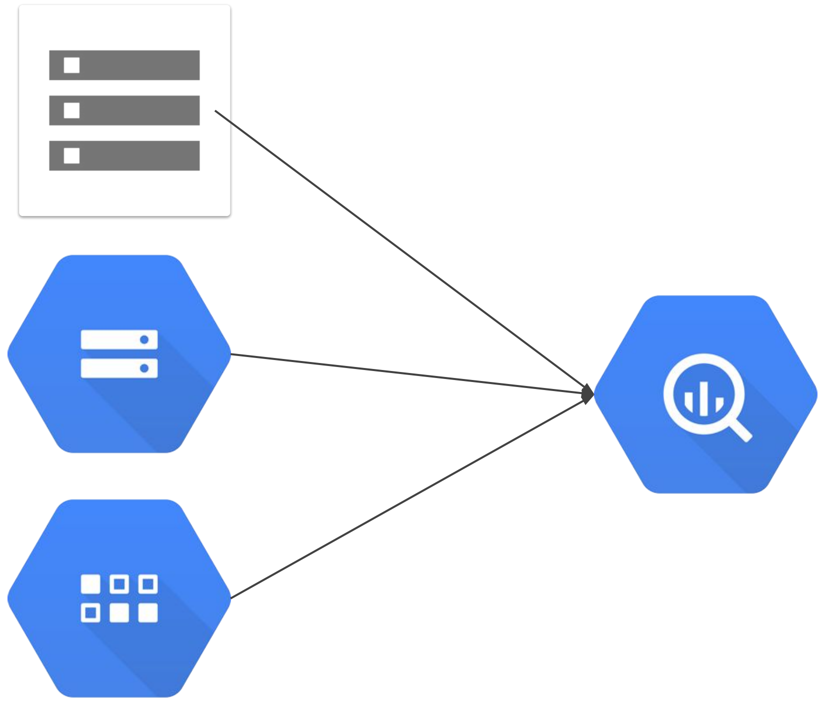
Familiar, SQL 2011 query language

Interactive ad-hoc analysis of petabyte-scale databases

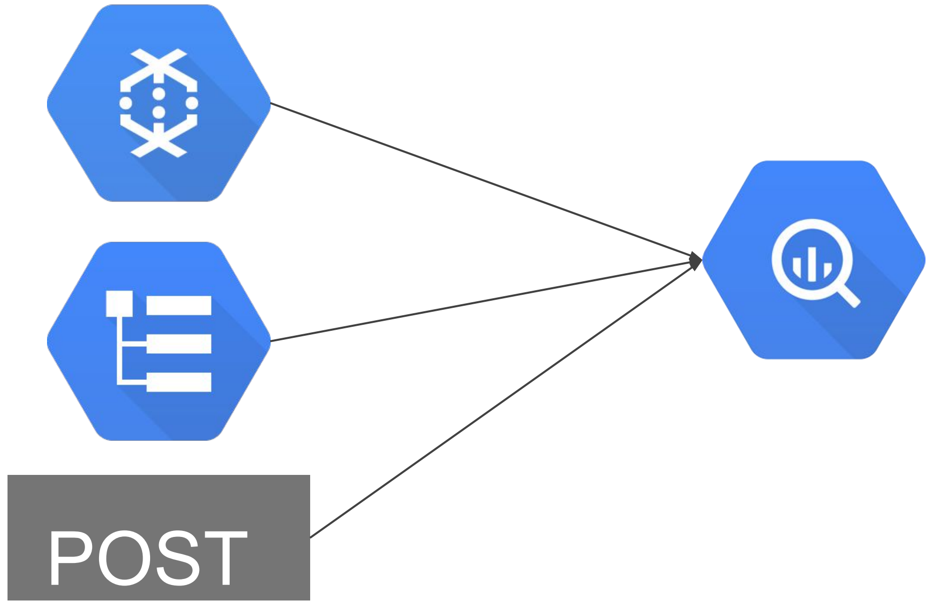
No need to provision clusters

Three ways of loading data into BigQuery

Files on disk or Cloud Storage

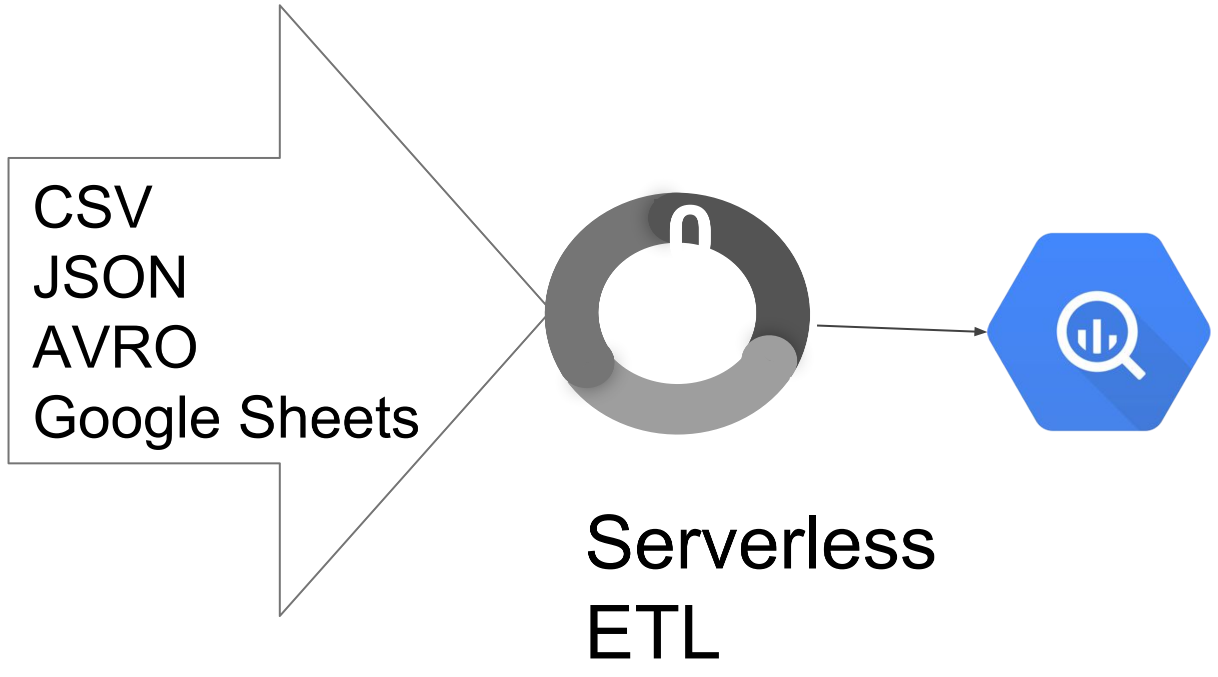


Stream Data



POST

Federated data source

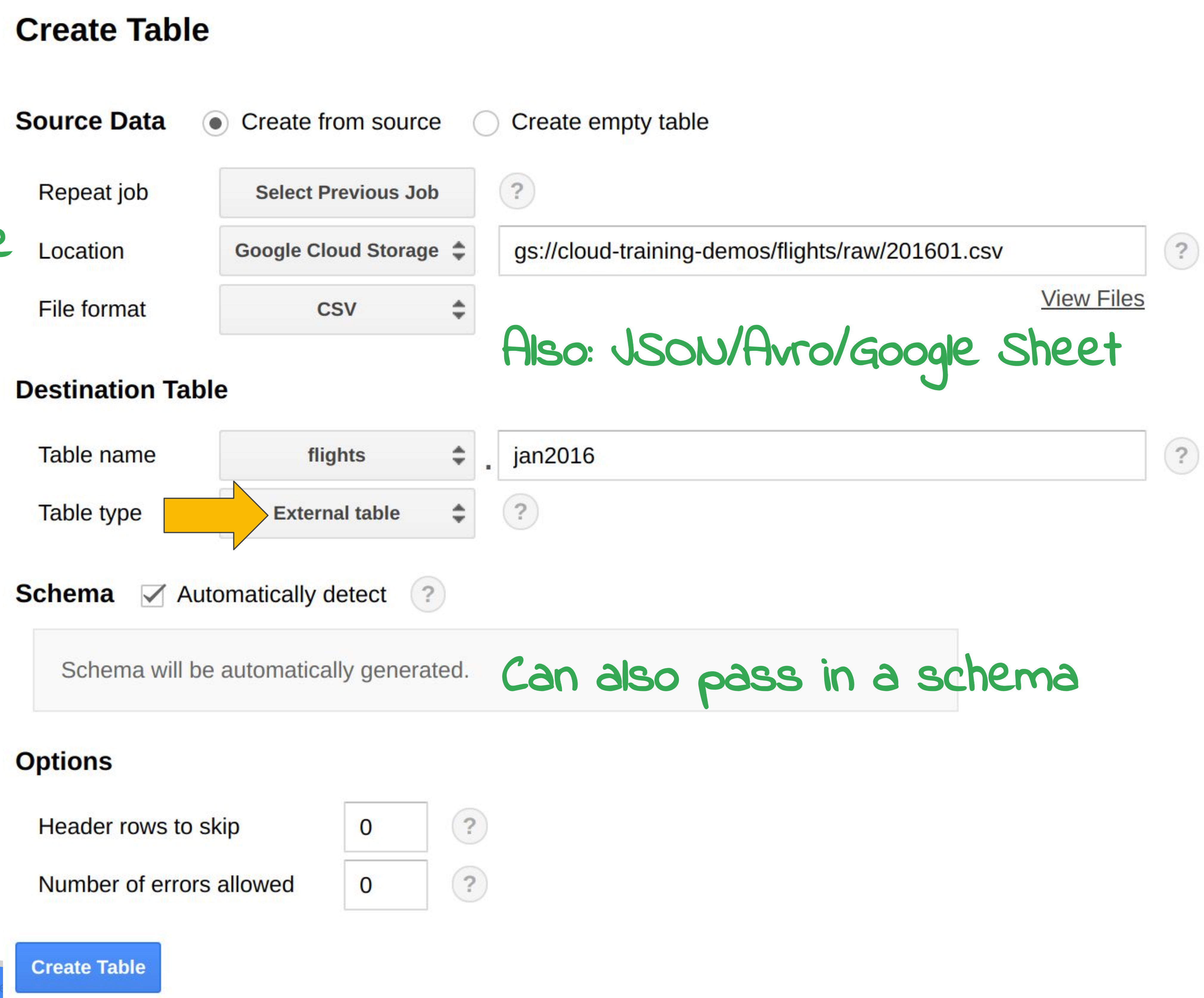


CSV
JSON
AVRO
Google Sheets

Serverless
ETL

With Federated data sources, you can directly query files on Cloud Storage, without having to ingest them into BigQuery

Also: Google Drive, Bigtable



Create Table

Source Data ● Create from source ○ Create empty table

Repeat job	Select Previous Job	
Location	Google Cloud Storage	gs://cloud-training-demos/flights/raw/201601.csv
File format	CSV	

Also: JSON/Avro/Google Sheet

Destination Table

Table name	flights	. jan2016
Table type	External table	

Schema ☑ Automatically detect

Schema will be automatically generated.

Can also pass in a schema

Options

Header rows to skip	0
Number of errors allowed	0

Agenda

Fast random access

Warehouse and interactively query petabytes

Interactive, iterative development + Lab

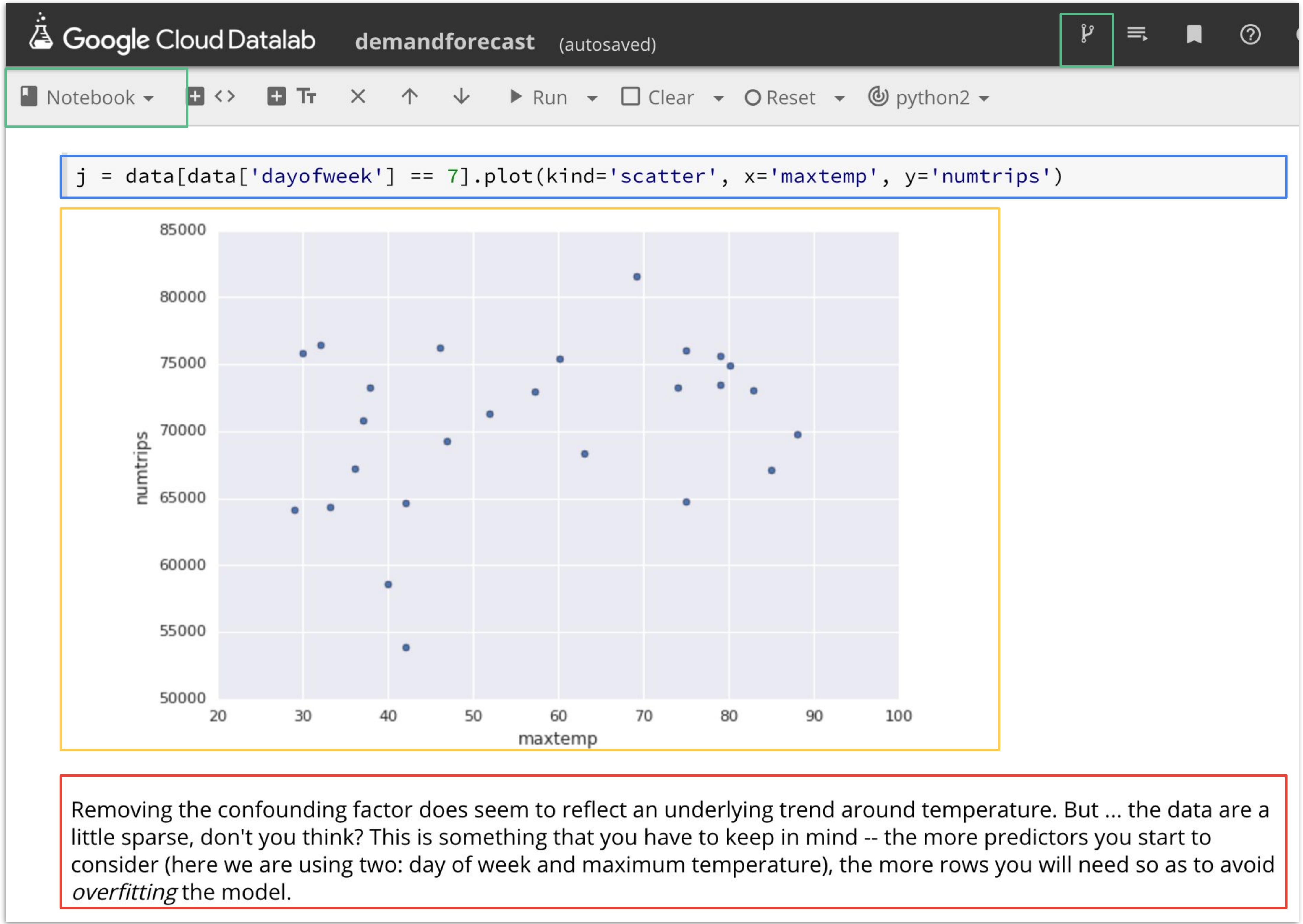
Increasingly, data analysis and machine learning are carried out in self-descriptive, shareable, executable notebooks

Share

Code

Output

Markup



```
j = data[data['dayofweek'] == 7].plot(kind='scatter', x='maxtemp', y='numtrips')
```

Removing the confounding factor does seem to reflect an underlying trend around temperature. But ... the data are a little sparse, don't you think? This is something that you have to keep in mind -- the more predictors you start to consider (here we are using two: day of week and maximum temperature), the more rows you will need so as to avoid *overfitting* the model.

A typical notebook contains code, charts, and explanations

Image Source:
Git Logo from Wikipedia

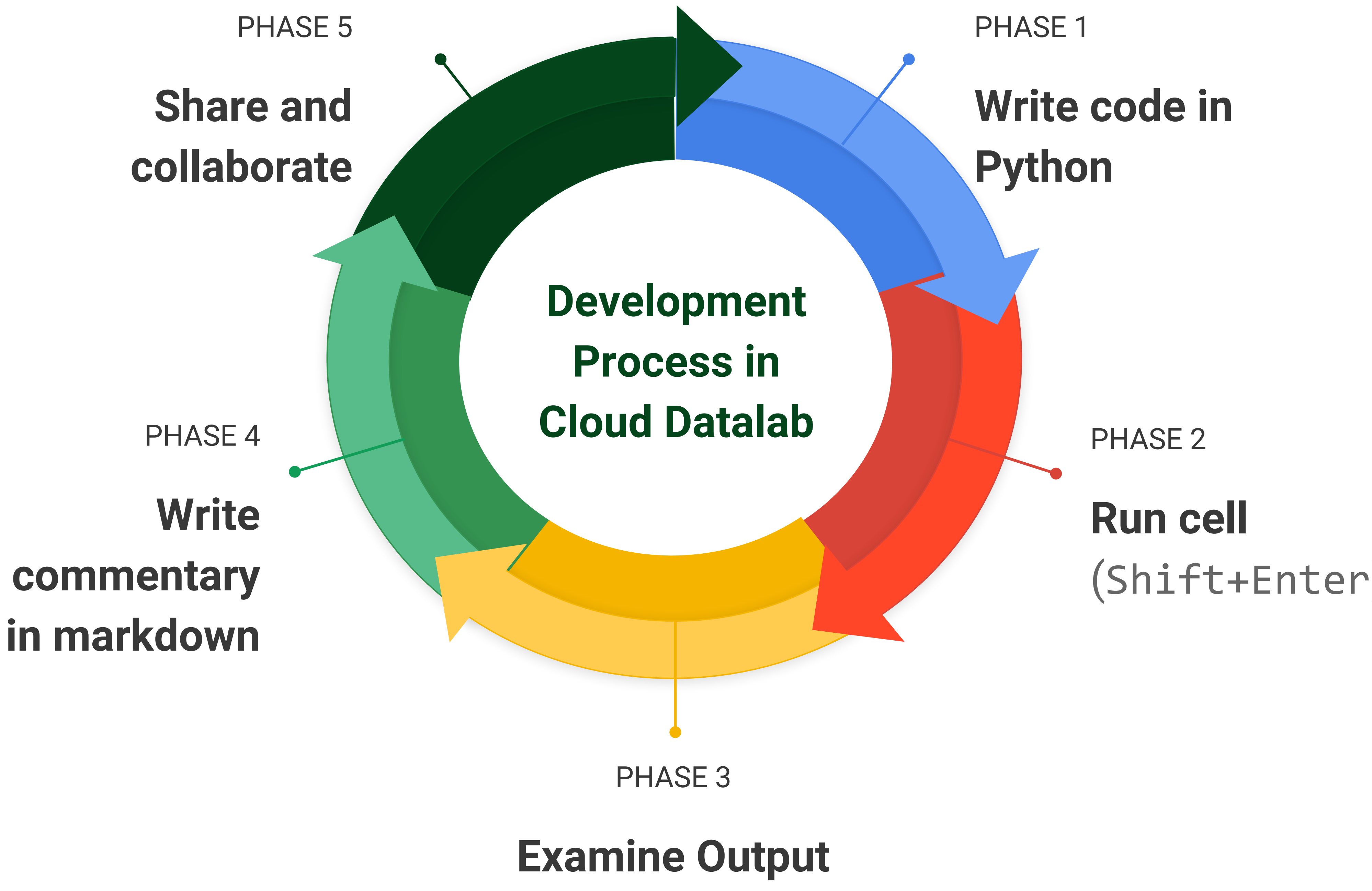
Datalab is an open-source notebook built on Jupyter (IPython)

Analyze data in BigQuery, Compute Engine or Cloud Storage

Datalab is free—just pay for Google Cloud resources

Use existing Python packages

Datalab notebooks are developed in an iterative, collaborative process



Development Process in Cloud Datalab

PHASE 1

Write code in Python

PHASE 2

Run cell

(Shift+Enter)

PHASE 3

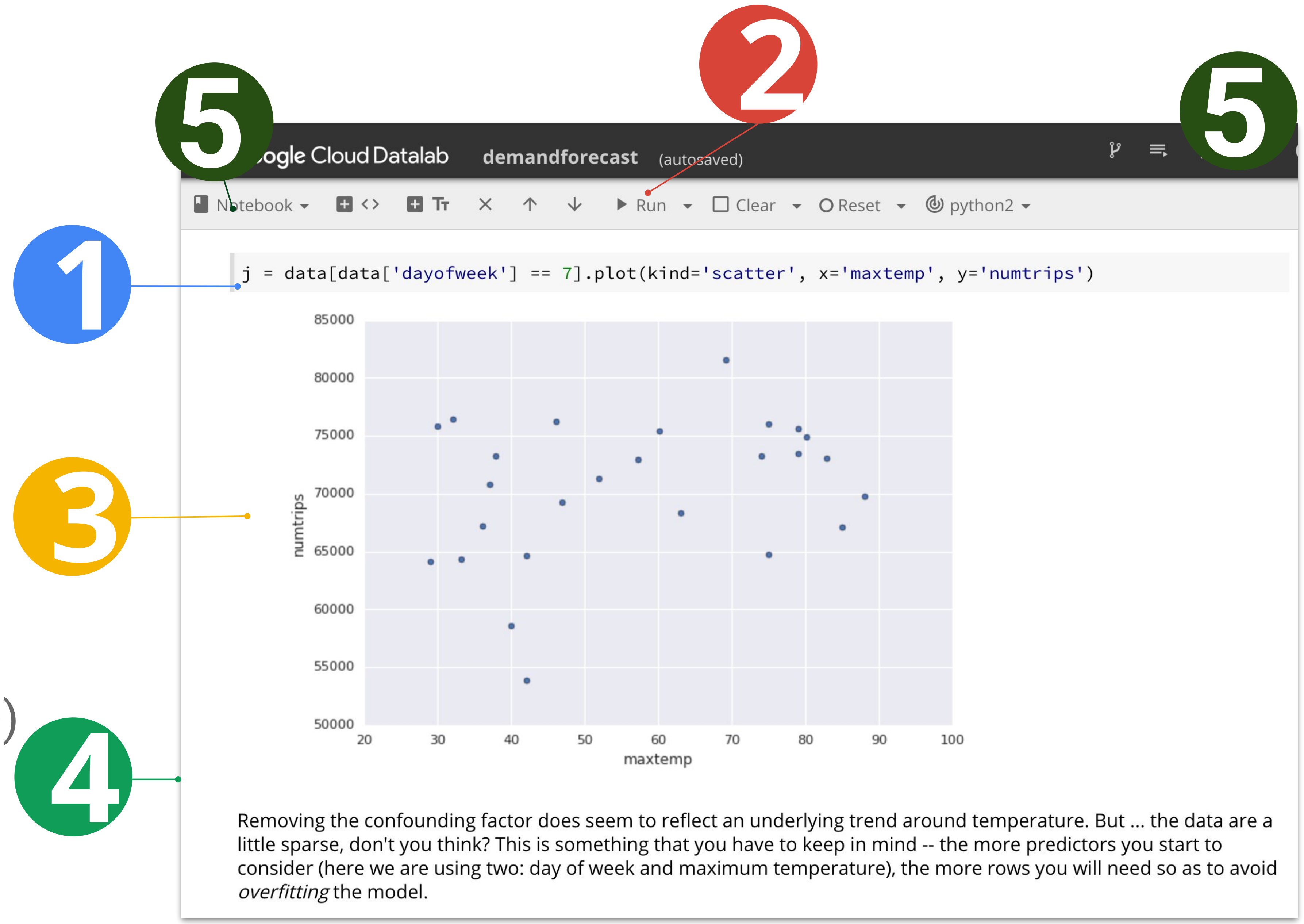
Examine Output

PHASE 4

Write commentary in markdown

PHASE 5

Share and collaborate

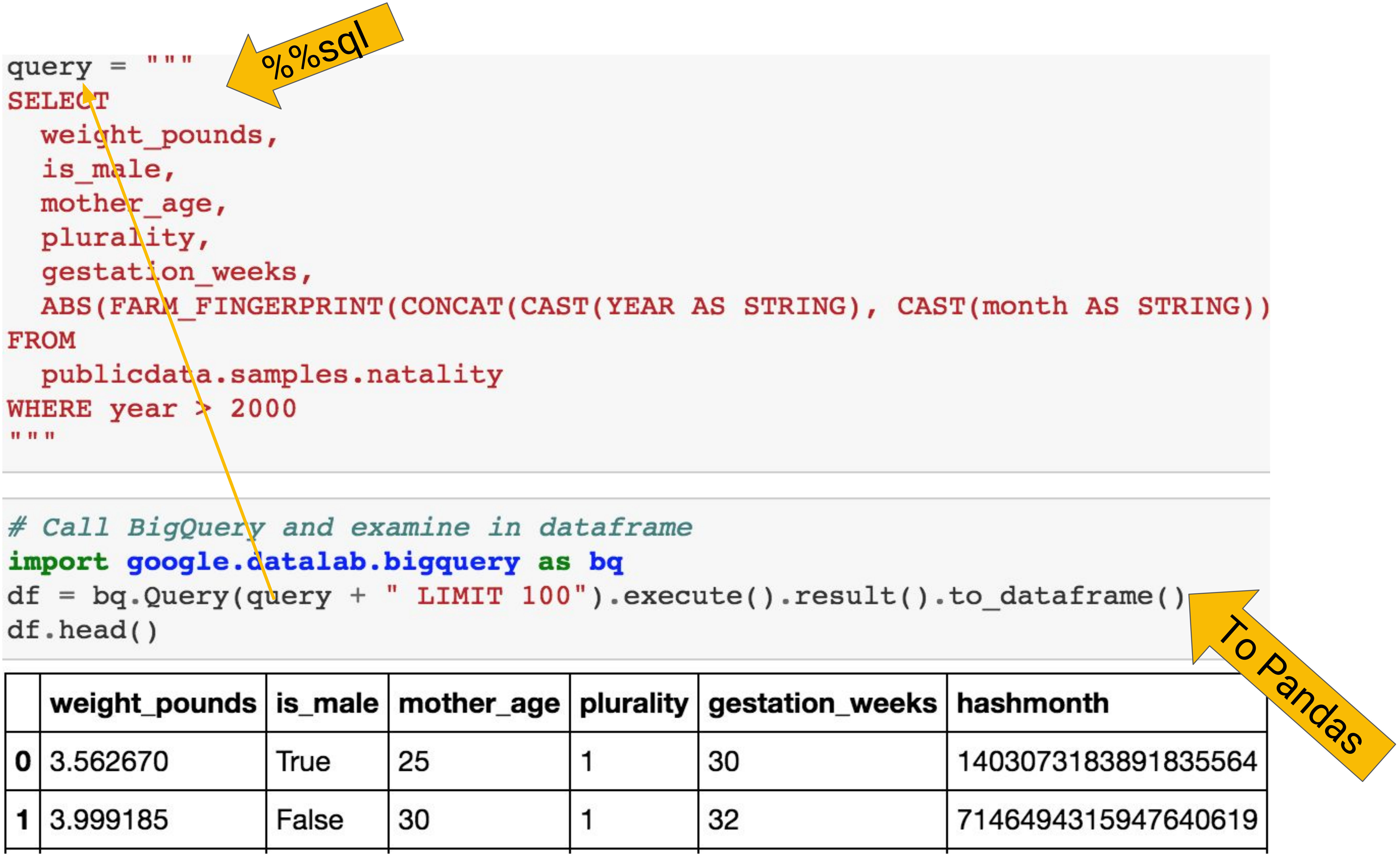


```
j = data[data['dayofweek'] == 7].plot(kind='scatter', x='maxtemp', y='numtrips')
```

Removing the confounding factor does seem to reflect an underlying trend around temperature. But ... the data are a little sparse, don't you think? This is something that you have to keep in mind -- the more predictors you start to consider (here we are using two: day of week and maximum temperature), the more rows you will need so as to avoid *overfitting* the model.

Google Cloud

Datalab supports BigQuery



%%sql

```
query = """
SELECT
  weight_pounds,
  is_male,
  mother_age,
  plurality,
  gestation_weeks,
  ABS(FARM_FINGERPRINT(CONCAT(CAST(YEAR AS STRING), CAST(month AS STRING))
FROM
  publicdata.samples.natality
WHERE year > 2000
"""
```

```
# Call BigQuery and examine in dataframe
import google.datalab.bigquery as bq
df = bq.Query(query + " LIMIT 100").execute().result().to_dataframe()
df.head()
```

	weight_pounds	is_male	mother_age	plurality	gestation_weeks	hashmonth
0	3.562670	True	25	1	30	1403073183891835564
1	3.999185	False	30	1	32	7146494315947640619

To Pandas

Lab: Create ML dataset with BigQuery

Lab 5: Create ML dataset with BigQuery

In this lab, we use BigQuery to create a dataset that we later use to build a taxi demand forecast system using Machine Learning.

- What kinds of things affect taxi demand?
- What are some ways to measure "demand"?

Google Cloud

Lab 5: Create ML dataset with BigQuery

In this lab, we use BigQuery to create a dataset that we later use to build a taxi demand forecast system using Machine Learning.

1. Use BigQuery and Datalab to explore and visualize data

2. Build a Pandas dataframe that will be used as the training dataset for machine learning using TensorFlow

Resources

Cloud Datastore	https://cloud.google.com/datastore/
Cloud Bigtable	https://cloud.google.com/bigtable/
Google BigQuery	https://cloud.google.com/bigquery/
Cloud Datalab	https://cloud.google.com/datalab/

Google Cloud

cloud.google.com

Images by Connie Zhou