**Introduction to Data Engineering.**

**Module introduction.**

In this introduction to data engineering module, we'll describe the role of a data engineer and motivate the claim why data engineering should be done in the Cloud.

A data engineer is someone who builds data pipelines.

And so we'll start by looking at what this means, what kinds of pipelines a data engineer builds and their purpose.

We'll look at the challenges associated with the practice of data engineering, and how many of those challenges are easier to address when you build your data pipelines in the Cloud.

Next, we'll introduce you to BigQuery, Google Cloud's petabyte scale serverless data warehouse.

Having defined what data lakes and data warehouses are, we'll then discuss these in more detail.

Data engineers may be responsible for both the backend transactional database systems that support a company's applications, and the data warehouses that support their analytic workloads.

In this lesson, we'll explore the differences between databases and data warehouses, and the Google Cloud's solutions for each of these workloads.

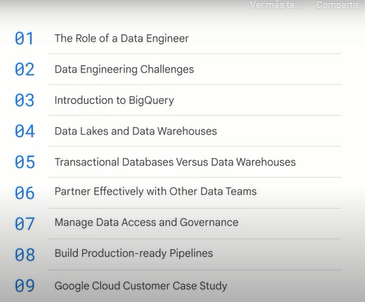
Since a data warehouse also serves other teams, it's crucial to learn how to partner effectively with them.

As part of being an effective partner, your engineering team will be asked to set up data access policies and overall governance of how data is to be used and not used by your users.

We'll discuss how to provide access to the data warehouse while keeping to data governance best practices.

We'll also discuss productionizing the whole operation and automating and monitoring as much of it as possible.

Finally, we'll look at a case study of how a Google Cloud customer solved a specific business problem before you complete a hands-on lab where you will use BigQuery to analyze data.



**The role of a data engineer.**

Let's start by exploring the role of a data engineer in a little more detail.

**What does a data engineer do?**

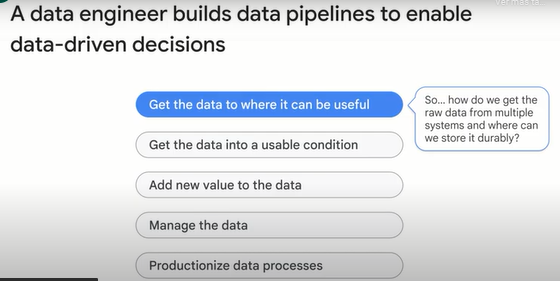
A data engineer builds data pipelines.

**Why does the data engineer build data pipelines?**

Because they want to get their data into a place, such as a dashboard or report or machine learning model, from where the business can make data-driven decisions.

The data has to be in a usable condition so that someone can use this data to make decisions.

Many times, the raw data is, by itself, not very useful.



One term you will hear a lot when you do data engineering is the concept of a data lake.

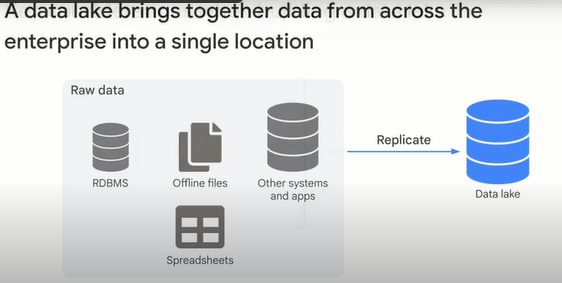
**A data lake brings together data from across the enterprise into a single location**.

So you might get the data from a relational database or from a spreadsheet and store the raw data in a data lake.

One option for this single location to store the raw data is to store it in a Cloud Storage bucket.

What are the key considerations when deciding between data lake options?

What do you think?



There are some considerations that you need to keep in mind as you build a data lake.

Does your data lake handle all the types of data you have?

Can it all fit into a Cloud Storage bucket?

If you have an RDBMS, you might need to put the data in Cloud SQL, a managed database, rather than Cloud Storage.

Can it elastically scale to meet the demand?

As your data collected increases, will you run out of disk space?

This is more a problem with on-premises systems than with Cloud.

Does it support high-throughput ingestion?

What is the network bandwidth?

Do you have edge points of presence?

Is there fine-grained access control to objects?

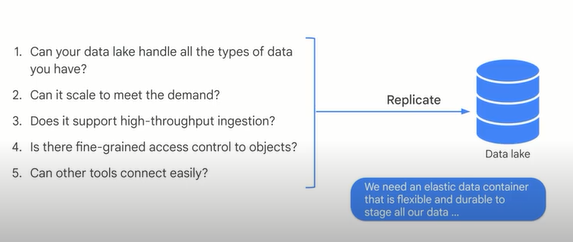
Do users need to seek within a file, or is it enough to get a file as a whole?

Cloud Storage is blob storage, so you might need to think about the granularity of what you store.

Can other tools connect easily?

How do they access the store?

Don't lose sight of the fact that the purpose of a data lake is to make data accessible for analytics.



We mentioned our first Google Cloud product, the Cloud Storage bucket, which is a good option for staging all of your raw data in one place before building transformation pipelines into your data warehouse.

Why choose Google Cloud Storage?

Commonly, businesses use Cloud Storage as a backup and archival utility for their businesses.

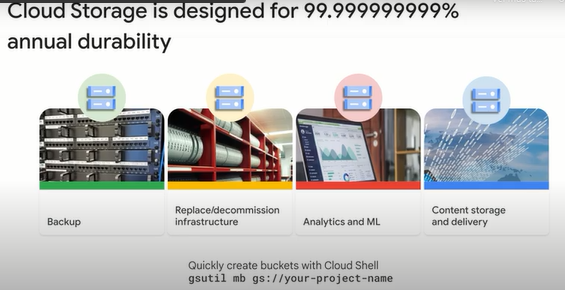
Because of Google's many data center locations and high network availability, storing data in a Cloud Storage bucket is durable and performant.

For a data engineer, you will often use a Cloud Storage bucket as part of your data lake to store many different raw data files, such as CSV, JSON, or Avro.

You could then load or query them directly from BigQuery as a data warehouse.

Later in the course, you'll create Cloud Shell buckets using the Cloud console and command line, like you see here.

Other Google Cloud products and services can easily query and integrate with your bucket once you've got it set up and loaded with data.



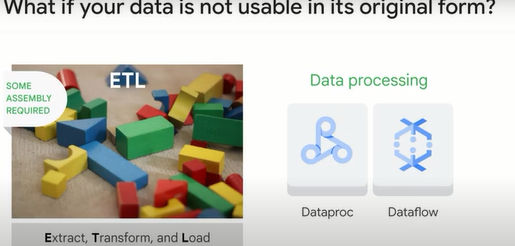
Speaking of loading data, what if your raw data needs additional processing?

You may need to extract the data from its original location, transform it, and then load it in.

One option is to carry out data processing.

This is often done using Dataproc or Dataflow.

We'll discuss using these products to carry out batch pipelines later in this course.

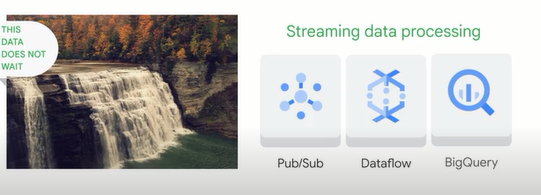


But what if batch pipelines are not enough?

What if you need real-time analytics on data that arrives continuously and endlessly?

In that case, you might receive the data in Pub/Sub, transform it using Dataflow, and stream it into BigQuery.

We'll discuss streaming pipelines later in this course.

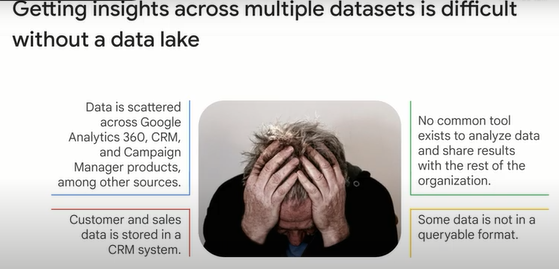


**Data engineering challenges.**

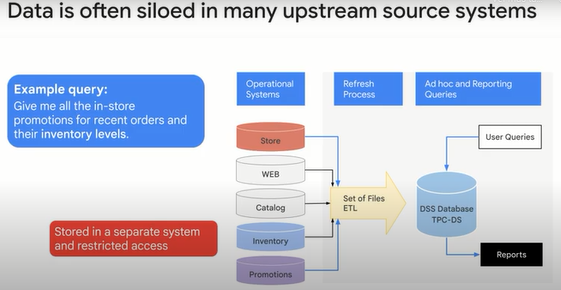
Let's look at some of the challenges that a data engineer faces. As a data engineer, you'll usually encounter a few problems when building data pipelines. You might find it difficult to access the data that you need, you might find that the data, even after you access it, doesn't have the quality that's required by the analytics or machine learning model. You plan to build a model, and even if the data quality exists, you might find that the transformations require computational resources that might not be available to you. And finally, you might run into challenges around query performance, and being able to run all of the queries and all of the transformations that you need with the computational resources that you have.



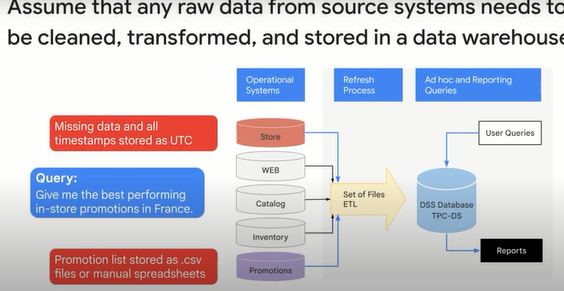
Let's take **the first challenge** of **consolidating disparate datasets, data formats and managing access at scale**. For example, you want to compute the customer acquisition cost, how much does it cost in terms of marketing and promotions and discounts to acquire a customer? That data might be scattered across a variety of marketing products and customer relationship management software. And finding a tool that can analyze all of this data might be difficult, because it might come from different organizations, different tools, and different schemas, and maybe some of that data is not even structured. So in order to find something as essential to your business as how much getting a new customer costs so that you can figure out what kind of discounts to offer to keep them from turning, you can't have your data exist in silos.



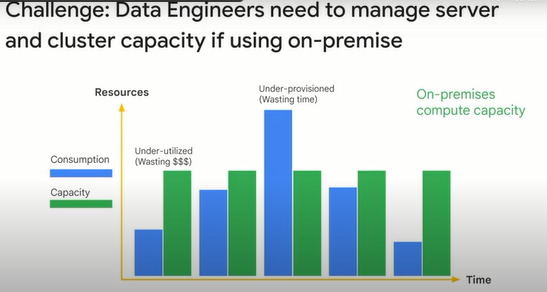
So what makes data access so difficult? Primarily, this is because data in many businesses is siloed by departments, and each department creates its own transactional systems to support its own business processes. So for example, you might have operational systems that correspond to store systems, have a different operational system maintained by your product warehouses that manages your inventory, and have a marketing department that manages all the promotions given that you need to do an analytic query on, such as, give me all the in store promotions for recent orders and their inventory levels. You need to know how to combine data from the stores, from the promotions, and from the inventory levels. And because these are all stored in separate systems, some of which have restricted access, building an analytic system that uses all three of these datasets to answer an ad hoc query like this can be very difficult.



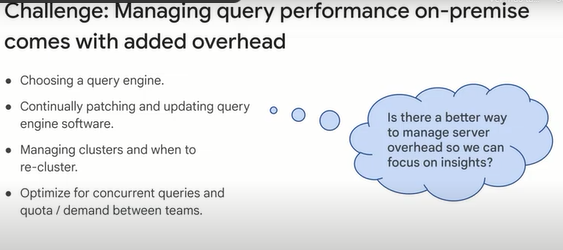
**The second challenge** is that **cleaning, formatting, and getting the data ready for insights** requires you to build ETL pipelines. ETL pipelines are usually necessary to ensure **data accuracy and quality**. The cleaned and transformed data are typically stored not in a data lake but in a data warehouse. A data warehouse is a consolidated place to store the data and all the data are easily joinable and queryable. Unlike a data lake where the data is in the raw format, in the data warehouse, the data is stored in a way that makes it efficient to query. Because data becomes useful only after you clean it up, you should assume that any raw data that you collect from source systems need to be cleaned and transformed. And if you are transforming it, you might as well transform it into a format that makes it efficient to query. In other words, ETL the data and store it in a data warehouse. Let's say you're a retailer, and you have to consolidate data from multiple source systems. Think about what the use case is. Suppose the use case is to get the best performing in store promotions in France, you need to get the data from the stores and you have to get the data from the promotions. But perhaps the stored data is missing information. Maybe some of the transactions are in cash. And for those, perhaps there is no information on who the customer is. Or some transactions might be spread over multiple receipts, and you might need to combine those transactions because they come from the same customer. Or perhaps the timestamps of the products are stored in local time, whereas you have to spread across the globe. And so before you do anything, you need to convert everything into UTC. Similarly, the promotions may not be stored in the transaction database at all. They might be just a text file that somebody loads on their web page and has a list of codes that are used by the web application to apply discounts. It can be extremely difficult to do a query like finding the best performing in store promotions because the data has so many problems. Whenever you have data like this, you need to get the raw data and transform it into a form with which you can actually carry out the necessary analysis. It is obviously best if you can do this sort of cleanup and consolidation just once and store the resulting data to make further analysis easy. That's the point of a data warehouse.



If you need to do so much consolidation and cleanup, a common problem that arises is where to carry out this compute. **The availability of computation resources can be a challenge**. If you're on an on-premises system, data engineers will need to manage server and cluster capacity and make sure that enough capacity exists to carry out the ETL jobs. The problem is that the compute needed by these ETL jobs is not constant over time. Very often it varies week to week, and depending on factors like holidays and promotional sales. This means that when traffic is low, you're wasting money, and when traffic is high, your jobs are taking way too long.



Once your data is in your data warehouse, you need to **optimize the queries** your users are running to make the most efficient use of your compute resources. If you're managing an on-premise data analytics cluster, you will be responsible for choosing a query engine and installing the query engine software and keeping it up to date as well as provisioning any more servers for additional capacity. Isn't there a better way to manage server overhead so we can focus on insights?



**Introduction to BigQuery.**

There is a much better way to manage server overhead so we can focus on insights, it is to use a serverless data warehouse. (*Hay una manera mucho mejor de administrar la sobrecarga del servidor para que podamos centrarnos en los conocimientos, es usar un almacén de datos sin servidor*.)

BigQuery is Google Cloud's petabyte scale serverless data warehouse, you don't have to manage clusters, just focus on insights. The BigQuery service replaces the typical hardware setup for a traditional data warehouse. That is, it serves as a collective home for all analytical data in an organization.

Datasets are collections of tables that can be divided along business lines, or a given analytical domain. Each dataset is tied to a Google Cloud project.

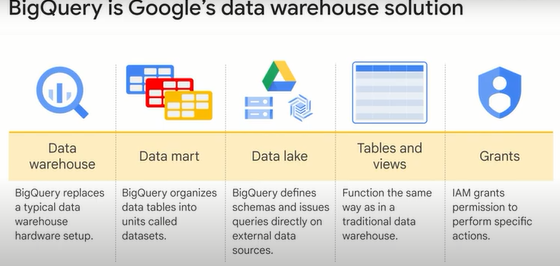
A data lake might contain files in Cloud storage or Google Drive, or even transactional data from Cloud Bigtable.

BigQuery can define a schema and issue queries directly on external data as federated data sources.

Database tables and views function the same way in BigQuery as they do in a traditional data warehouse, allowing BigQuery to support queries written in a standard SQL dialect, which is ANSI 2011 compliant.

Identity and Access Management is used to grant permission to perform specific actions in BigQuery.

This replaces the SQL Grant and Revoke statements that are used to manage access permissions in traditional SQL databases.



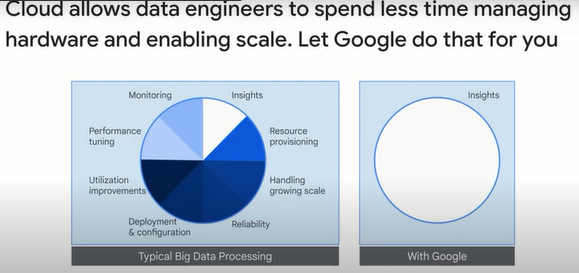
A key consideration behind agility is being able to do more with less, and it's important to make sure you're not doing things that don't add value.

If you do work that is common across multiple industries, it's probably not something that your business wants to pay for.

The Cloud lets you, the data engineer, spend less time managing hardware and more time doing things that are much more customized and specific to the business.

You don't have to be concerned about provisioning and reliability, and utilization improvements, and performance or tuning on the Cloud.

So you can spend all your time thinking about how to get better insights from your data.



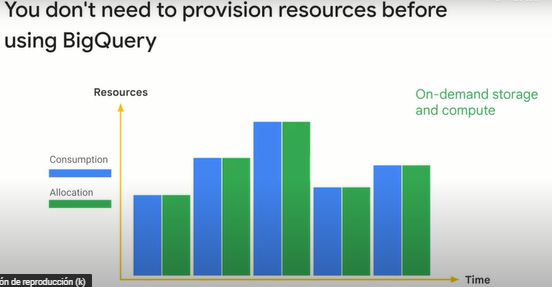
You don't need to provision resources before using BigQuery.

Unlike many RDBMS systems, BigQuery allocates storage and query resources dynamically based on your usage patterns.

Storage resources are allocated as you consume them and de-allocated as you remove data or drop tables.

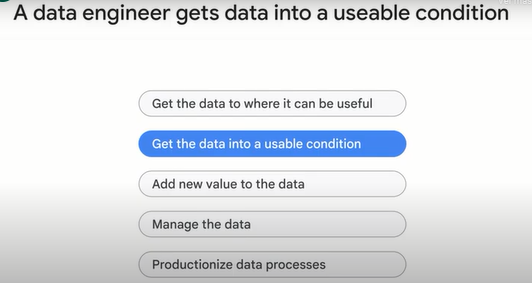
Query resources are allocated according to query type and complexity.

Each query uses some number of slots, which are units of computation that comprise a certain amount of CPU and RAM.

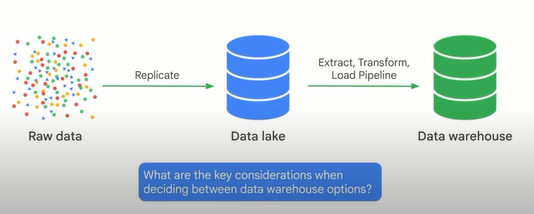


**Data lakes and data warehouses.**

We've defined what a data lake is and what a data warehouse is. Let's look at these in a bit more detail. Recall that we emphasized that the data has to be in a usable condition so that someone can use this data to make decisions. Many times, the raw data is by itself not very useful.



We said that raw data gets replicated and stored in a data lake. In order to make that data usable, you will use Extract Transform Load or ETL pipelines to make the data usable and store this more usable data in a data warehouse. Let's consider what are the key considerations when deciding between data warehouse options?



We need to ask ourselves these questions.

The data warehouse is going to definitely serve as a sink (pileta, repositorio), you're going to store data in it.

But will it be fed by a batch pipeline or by a streaming pipeline?

Does the warehouse need to be accurate up to the minute?

Or is it enough to load data into it once a day or once a week?

Will the data warehouse scale to meet my needs?

Many cluster based data warehouses will set per cluster concurrent query limits.

Will those query limits cause a problem?

Will the cluster size be large enough to store and traverse your data?

How is the data organized, cataloged and access controlled?

Will you be able to share access to the data to all your stakeholders?

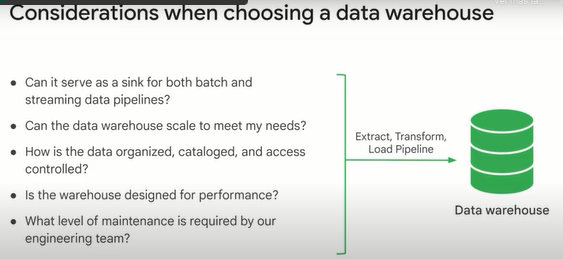
What happens if they want to query the data?

Who will pay for the querying?

Is the warehouse designed for performance?

Again, carefully consider concurrent query performance and whether that performance comes out of the box, or whether you need to go around creating indexes and tuning the data warehouse.

Finally, what level of maintenance is required by your engineering team?



Traditional data warehouses are hard to manage and operate. They were designed for a batch paradigm of data analytics and for operational reporting needs. The data in the data warehouse was meant to be used by only a few management folks for reporting purposes. BigQuery is a modern data warehouse that changes the conventional mode of data warehousing. Here, we can see some of the key comparisons between a traditional data warehouse and BigQuery. BigQuery provides mechanisms for automated data transfer, empowers business applications using technology that teams already know and use, so everyone has access to data insights.

You can create read-only shared data sources that both internal and external users can query, and make query results accessible for anyone through user-friendly tools such as Looker, Google Sheets, Tableau, or Google Data Studio. BigQuery lays the foundation for AI, it's possible to train TensorFlow and Google Cloud Machine Learning models directly with datasets stored in BigQuery, and BigQuery ML can be used to build and train machine learning models with simple SQL.

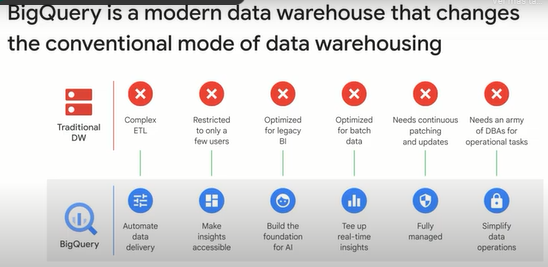
Another extended capability is BigQuery GIS, which allows organizations to analyze geographic data in BigQuery essential to many critical business decisions that revolve around location data.

BigQuery also allows organizations to analyze business events real time as they unfold by automatically ingesting data and making it immediately available to query in their data warehouse.

This is supported by the ability of BigQuery to ingest up to 100,000 rows of data per second and for petabytes of data to be queried at lightning fast speeds.

Due to Google's fully managed serverless infrastructure and globally available network, BigQuery eliminates the work associated with provisioning and maintaining a traditional data warehousing infrastructure.

BigQuery also simplifies data operations through the use of identity and access management to control user access to resources, creating roles and groups and assigning permissions for running jobs and queries in a project and also providing automatic data backup and replications.



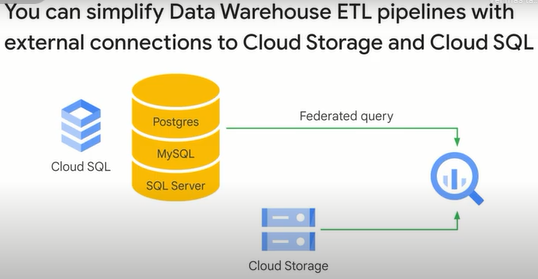
Even though we talked about getting data into BigQuery by running ETL pipelines, there is another option, that is to treat BigQuery as a query engine, and allow it to query the data in place.

For example, you can use BigQuery to directly query database data in Cloud SQL, that is, managed relational databases like Postgres SQL, MySQL, and SQL Server.

You can also use BigQuery to directly query files on Cloud Storage as long as these files are in formats like CSV or Parquet.

The real power comes when you can leave your data in place and still join it against other data in the data warehouse.

Let's take a look.



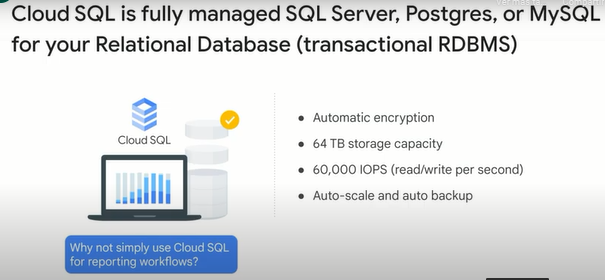
**Transactional databases versus data warehouses.**

Data engineers may be responsible for both the backend transactional database systems that support your company's applications AND the data warehouses that support your analytic workloads. In this lesson, you'll explore the differences between databases and data warehouses and the Google Cloud solutions for each workload. If you have SQL Server MySQL or PostgresSQL as your relational database, you can migrate it to Cloud SQL, which is Google Cloud's fully managed relational database solution.

Cloud SQL delivers high performance and scalability with up to 64 terabytes of storage capacity, 60,000 IOPS and 624 gigabytes of RAM per instance.

You can take advantage of storage auto-scale to handle growing database needs with zero downtime.

One question you might get asked is: "Why not simply use Cloud SQL for reporting workflows? You can run SQL directly on the database, right?"



This is a great question and will be answered in greater detail in the "Building a Data Warehouse" module.

Google Cloud has several options for RDBMS's, including Spanner and Cloud SQL.

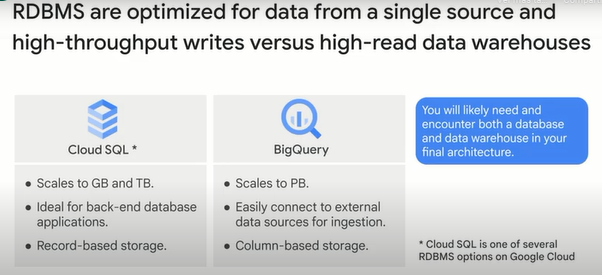
When considering Cloud SQL, be aware that Cloud SQL is optimized to be a database for transactions or rights, and BigQuery is a data warehouse optimized for reporting workloads (mostly reads).

The fundamental architecture of these data storage options is quite different.

Cloud SQL databases are RECORD-based storage, meaning the entire record must be opened on disk.

Even if you just selected a single column in your query.

BigQuery is COLUMN based storage, which as you might guess, allows for really wide reporting schemas, since you can simply read individual columns out from disk.



This isn't to say RDBMS's aren't as performant as Data Warehouses.

They serve two different purposes.

RDBMS helps your business manage new transactions.

Take this point of sale terminal at a storefront.

Each order and product is likely written out as new records in a relational database somewhere.

This database may store all of the orders received from their website.

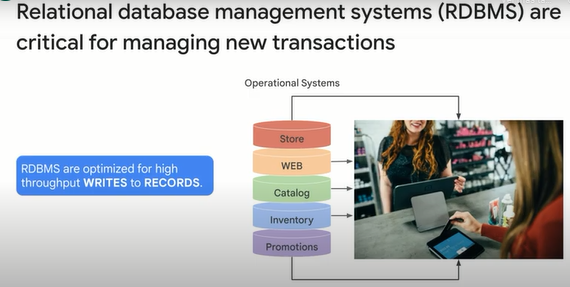
All of the products listed in the catalog or the number of items in their inventory.

This is so that when an existing order is changed, it can be quickly updated in the database.

Transactional systems allow for a single row in a database table to be modified in a consistent way.

They also are built on certain relational database principles like referential integrity to guard against cases like a customer ordering a product that doesn't exist in the product table.

So where does all this raw data end up In our data lake and data warehouse discussion?



What's the complete picture?

Here it is.

Our operational systems, like our relational databases that store online orders, inventory and promotions, are our raw data sources on the left.

Note that this isn't exhaustive, you could have other source systems that are manual like CSV files or spreadsheets too.

These upstream data sources get gathered together in a single consolidated location in our Data Lake, which is designed for durability and high availability.

Once in the data lake, the data often needs to be processed via transformations that then output the data into our data warehouse, where it is ready for use by downstream teams.

Here are three quick examples of other teams that often build pipelines on our data warehouse.

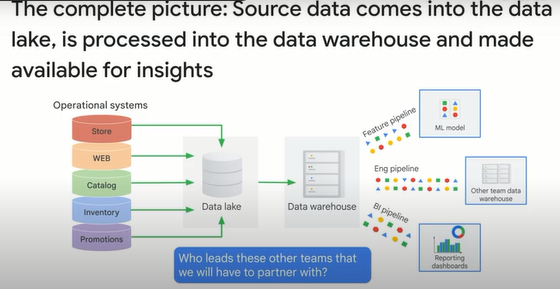
An ML team may build a pipeline to get features for their models.

04:13

An engineering team may be using our data as part of their data warehouse.

And a BI team may want to build dashboards using some of our data.

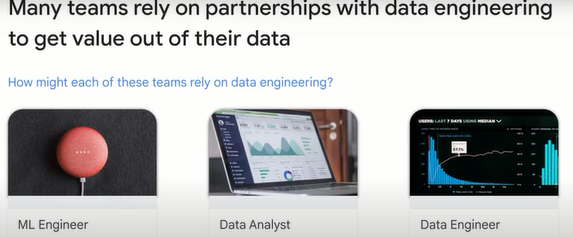
So who works on these teams and how do they partner with our data engineering team?



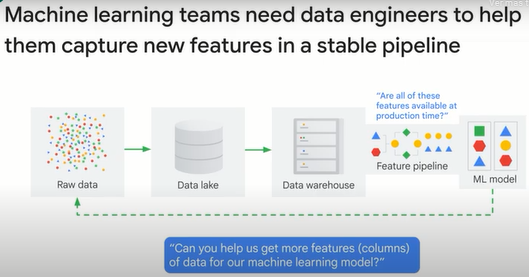
**Partner effectively with other data teams.**

Since a data warehouse also serves other teams, it is crucial to learn how to partner effectively with them. Remember that once you've got data where it can be useful, and it's in a usable condition, we need to add new value to the data through analytics and machine learning. What teams might rely (confiar) on our data?

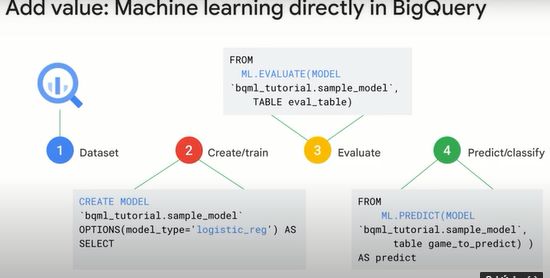
There are many data teams that rely on your data warehouse and partnerships with data engineering to build and maintain new data pipelines. The three most common clients are the machine learning engineer, the data or BI analyst, and other data engineers.



Let's examine how each of these roles interacts with your new data warehouse and how data engineers can best partner with them. As you'll see in our course on machine learning, an ML team's models rely on having lots of high-quality input data to create, train, test, evaluate, and serve their models. They will often partner with data engineering teams to build pipelines and data sets for use in their models. Two common questions you may get asked are: "How long does it take for a transaction to make it from raw data all the way into the data warehouse?" They're asking this because any data that they train their models on must also be available at prediction time as well. If there is a long delay in collecting and aggregating the raw data, it will impact the ML team's ability to create useful models. A second question that you will definitely get asked is how difficult it would be to add more columns or rows of data into certain data sets. Again, the ML team relies on teasing out relationships between the columns of data and having a rich history to train models on.(*Nuevamente, el equipo de ML se basa en descubrir las relaciones entre las columnas de datos y tener un historial completo para entrenar modelos.*) You will earn the trust of your partner ML teams by making your data sets easily discoverable, documented, and available to ML teams to experiment on quickly.



A unique feature of BigQuery is that you can create high-performing machine learning models directly in BigQuery using just SQL by using BigQuery ML. Here is the actual model code to create a model, evaluate it, and then make predictions. You'll see this again in our lectures on machine learning later on.



Other critical stakeholders are your business intelligence and data analyst teams that rely on good, clean data to query for insights and build dashboards. These teams need data sets that have clearly defined schema definitions, the ability to quickly preview rows, and the performance to scale to many concurrent dashboard users. One of the Google Cloud products that helps manage the performance of dashboards is BigQuery BI Engine. BI Engine is a fast in-memory analysis service that is built directly into BigQuery and available to speed up your business intelligence applications. Historically, BI teams would have to build, manage, and optimize their own BI servers and OLAP cubes to support reporting applications. Now, with BI Engine, you can get sub-second query response time on your BigQuery data sets without having to create your own cubes. BI Engine is built on top of the same BigQuery storage and compute architecture and servers as a fast in-memory intelligent caching service that maintains state.

One last group of stakeholders are other data engineers that rely on the uptime and performance of your data warehouse and pipelines for their downstream data lakes and data warehouses. They will often ask, "How can you ensure that the data pipeline we depend on will always be available when we need it?" or "We are noticing high demand for certain really popular data sets. How can you monitor and scale the health of your entire data ecosystem?" One popular way is to use the built-in cloud monitoring of all resources on Google Cloud. Since Google Cloud Storage and BigQuery are resources, you can set up alerts and notifications for metrics like query count or bytes of data processed so you can better track usage and performance, another two reasons why cloud monitoring is used for tracking spending of all the different resources used and what the billing trends are for your team or organization. And lastly, you can use the Cloud Audit Logs to view actual query job information to see granular-level details about which queries were executed and by whom. This is useful if you have sensitive data sets that you need to monitor closely, a topic we will discuss more next.

**Manage data access and governance.**

As part of being an effective partner, your engineering team will be asked to set up data access policies and overall governance of how data is to be used and not used by your users.

This is what we mean when we say a data engineer must manage the data.

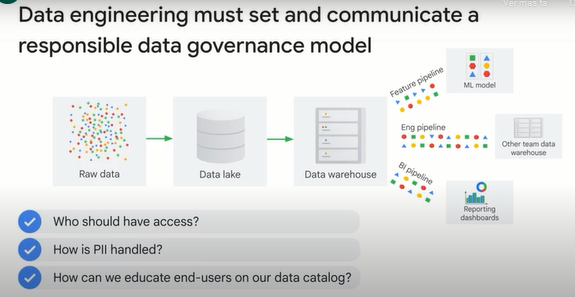
This includes critical topics such as privacy and security.

What are some key considerations when managing certain datasets?

Clearly communicating a data governance model for who should and should not have access.

How is personally identifiable information like phone numbers or email addresses handled?

And even more basic tasks like how will our end users discover the different datasets we have for analysis?



One solution for data governance is the Cloud data catalog and the data loss prevention API.

The data catalog makes all the metadata about your datasets available to search for your users.

You group datasets together with tags, flag certain columns as sensitive et cetera.

Why is this useful?

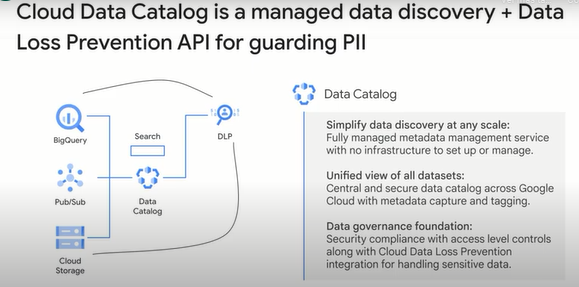
If you have many different datasets with many different tables, which different users have different access levels to?

The data catalog provides a single unified user experience for discovering those datasets quickly.

No more hunting for specific table names and SQL first.

Often used in conjunction with data catalog is the Cloud Data Loss Prevention API, or DLP API, which helps you better understand and manage sensitive data.

It provides fast, scalable classification and reduction for sensitive data elements like credit card numbers, names, social security numbers, US and selected international identifier numbers, phone numbers and Google Cloud credentials.



**Demo: Finding PII in your dataset with the DLP API.**

In this demo, we're going to use the Cloud Data Loss Prevention API to redact personally identifiable information like email addresses, phone numbers, that sort of thing, PII data, at scale.

Again, find all these demos under the data engineering demos course folder in our public repository.

This one is pretty short, we're just going to be using the web tool, and you can invoke the API and experiment with that yourself.

So navigating to the web demo, I'm just going to copy that link.

So imagine in BigQuery, you've got like email addresses or something like that inside of your data.

How do we actually proactively identify it?

So this demo will actually take you to a page that has just a tax code file inside of here.

Inside of the file, it actually explains a little bit about the DLP API itself.

You can have this, it identifies this, it just looks through here, this unstructured data, and it structures it and basically says, "Hey, I'm reading through here, yeah, it's all good. Hey, somebody put a phone number inside of their comments, I don't want to have this go to anybody."

I'm going to actually flag that as very high likelihood that it's this type, it is a phone number, here is the string, here is where it is.

And you can inform the model whether or not that's a good result or a poor result.

So what we can do is we can hide that welcome text, and we can paste in our own examples where we know this is a personally identifiable information, and we can see if it catches it.

Boom, immediately.

Credit card number, very high.

US driver's license, very high.

Email address, part of the email address, you also have the domain and where the data is as well.

So you imagine this is a trivial demo.

This is just four lines here, but imagine your inherited dataset that is terabytes.

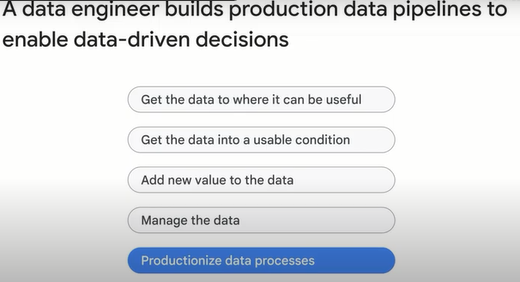
How can you automatically and at scale, run through, identify it, and then also use the different components of the API to run hash functions or obfuscation functions to redact that data so that PII doesn't leak out there?

All right, that's DLP, the API.

**Build production-ready pipelines.**

Once your data lakes and data warehouses are set up and your governance policy is in place, it's time to productionalize the whole operation and automate and monitor as much of it as we can.

That's what we mean when we say productionalize the data process.

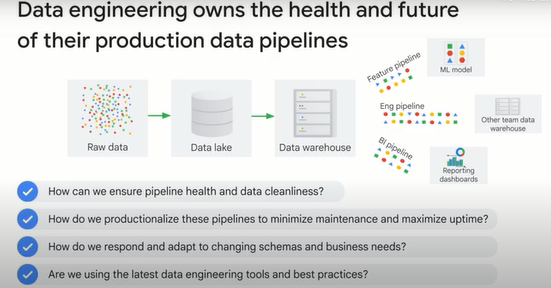


It has to be an end-to-end and scalable data processing system.

Your data engineering team is responsible for the health of the plumbing-- that is, the pipelines-- and ensuring that the data is available and up to date for analytic and ML workloads.

Common questions that you should ask at this phase are: "How can we ensure pipeline health and data cleanliness?" "How do we productionalize these pipelines to minimize maintenance and maximize uptime?" "How do we respond and adapt to changing schemas and business needs?"

And, "Are we using the latest data engineering tools and best practices?"



One common workflow orchestration tool used by enterprises is Apache Airflow.

Google Cloud has a fully-managed version of Airflow called "Cloud Composer."

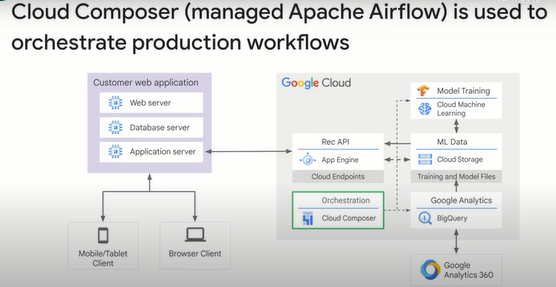
Cloud Composer helps your data engineering team orchestrate all the pieces to the data engineering puzzle that we have discussed so far and even more that you haven't come across yet.

For example, when a new CSV file gets dropped into Cloud Storage, you can automatically have that trigger an event that kicks off a data processing workflow and puts that data directly into your data warehouse.

The power of this tool comes from the fact that Google Cloud big data products and services have API endpoints that you can call.

A Cloud Composer job can then run every night or every hour and kick off your entire pipeline from raw data to the data lake and into the data warehouse for you.

We'll discuss workflow orchestration in greater detail in later modules, and you'll do a lab on Cloud Composer as well.



**Google Cloud customer case study.**

We have looked at a lot of different aspects of what a data engineer has to do.

Let's look at a case study of how a Google Cloud customer solves a specific business problem.

That will help tie all these different aspects together.

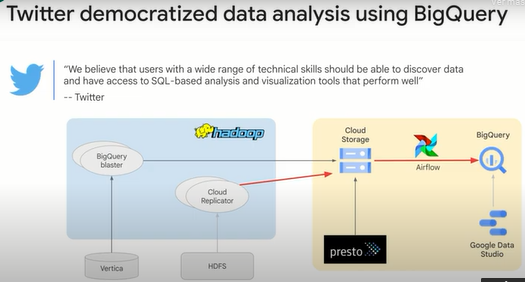
Twitter has large amounts of data and they also have high-powered sales teams and marketing teams which, for a long time, did not have access to the data and couldn't use that data for carrying out the analysis that they wanted to be able to do.

Much of the data was stored on Hadoop clusters that were completely overtaxed.

So Twitter replicated some of that data from HDFS onto Cloud Storage, loaded it into BigQuery, and provided BigQuery to the rest of the organization.

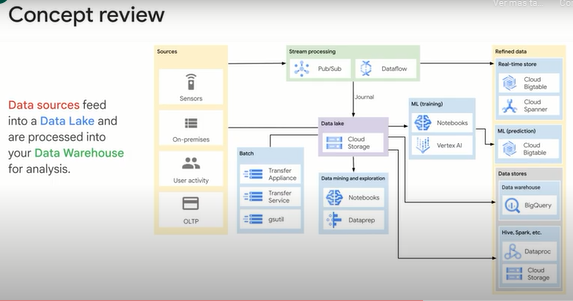
These were some of the most frequently requested data sets within Twitter, and they discovered that with ready access to the data, many people who were not data analysts are now analyzing data and making better decisions as a result.

For more information, a link to the blog post is available in the PDF version of this content under "course resources.".

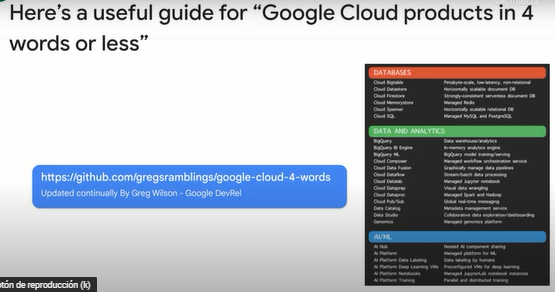


**Recap.**

Let's summarize the major topics we covered so far in this introduction. Recall that your data sources are: your upstream systems, like RDBMS and other raw data that comes from your business in different formats; data lakes, your consolidated location for raw data that is durable and highly available--in this example, our data lake is Cloud Storage-- and data warehouses, which are the end result of preprocessing the raw data in your data lake and getting it ready for analytic and ML workloads-- you'll notice a lot of other Google Cloud product icons here, like Batch--and streaming data into your lake and running ML on your data. We'll cover those topics in detail later in this course.



A useful cheat sheet to bookmark as a reference is the "Google Cloud products in 4 words or less," which is actively maintained on GitHub by our Google Developer Relations team. It's also a great way to stay on top of new products and services that come out just by following the GitHub commits.



**Using BigQuery to do Analysis.**

**Prerequisites**

This is a fundamental level lab and assumes some experience with BigQuery and SQL.

**Introduction**

This lab uses two public datasets in BigQuery: weather data from the US National Oceanic and Atmospheric Administration (NOAA), and bicycle rental data from New York City.

You will encounter, for the first time, several aspects of Google Cloud Platform that are of great benefit to scientists:

Serverless. No need to download data to your machine in order to work with it - the dataset will remain on the cloud.

Ease of use. Run ad-hoc SQL queries on your dataset without having to prepare the data, like indexes, beforehand. This is invaluable for data exploration.

Scale. Carry out data exploration on extremely large datasets interactively. You don't need to sample the data in order to work with it in a timely manner.

Shareability. You will be able to run queries on data from different datasets without any issues. BigQuery is a convenient way to share datasets. Of course, you can also keep your data private, or share them only with specific persons -- not all data need to be public.

The end-result is that you will find if there are lesser bike rentals on rainy days.

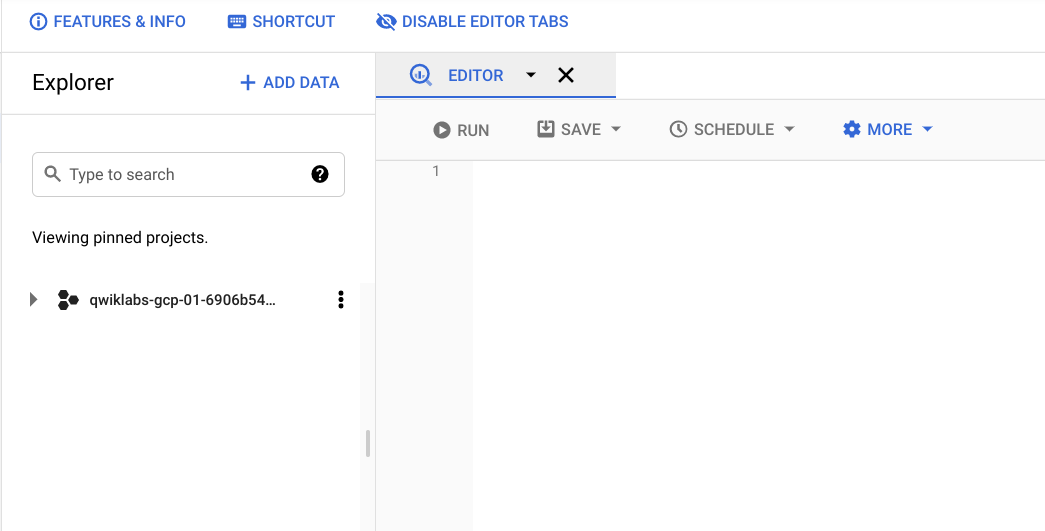
## Explore bicycle rental data

### **Open BigQuery Console**

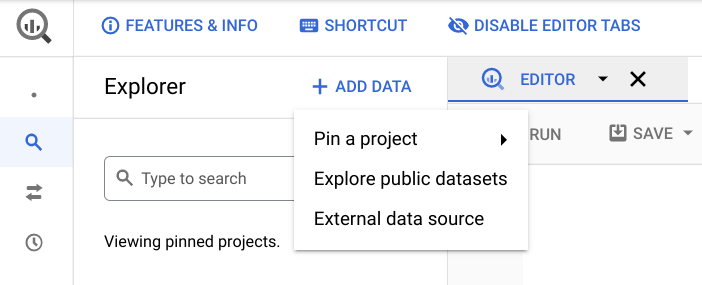
1. In the Google Cloud Console, select **Navigation menu** > **BigQuery**.

The **Welcome to BigQuery in the Cloud Console** message box opens. This message box provides a link to the quickstart guide and lists UI updates.

1. Click **Done**.



1. In the left pane, click **ADD DATA** > **Explore public datasets**.



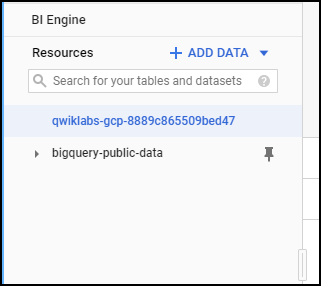
The Datasets window opens.

1. In the **Search** bar, type "NYC bike" then press **Enter**.
2. One result **NYC Citi Bike Trips** is returned. Click on the dataset name and then **View Dataset**.

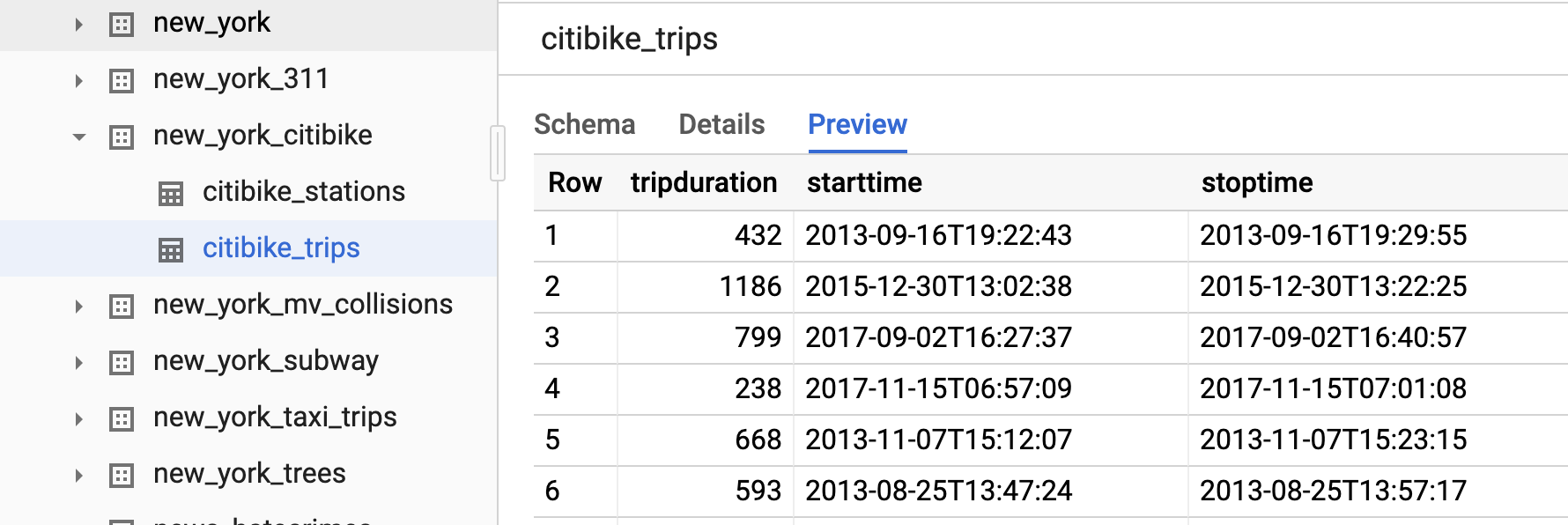
Your Google BigQuery console opens in a new browser tab.To keep your workspace organized, close this new browser tab and refresh the first tab.

**Note**: If the new project bigquery-public-data doesn't appear to the Explorer panel, then click on **+ ADD DATA** -> **Pin a project** -> **Enter project name** (bigquery-public-data) and **Pin**.

1. In the BigQuery console (in the first browser tab) you see two projects in the left pane, one named your Qwiklabs project ID, and one named **bigquery-public-data**.



1. In the left pane of the BigQuery console, select **bigquery-public-data** > **new\_york\_citibike** > **citibike\_trips** table.
2. In the Table (citibike\_trips) window, click the Preview tab.



1. Examine the columns and some of the data values.

Click **Compose New Query** and enter the following:

SELECT

MIN(start\_station\_name) AS start\_station\_name,

MIN(end\_station\_name) AS end\_station\_name,

APPROX\_QUANTILES(tripduration, 10)[OFFSET (5)] AS typical\_duration,

COUNT(tripduration) AS num\_trips

FROM

`bigquery-public-data.new\_york\_citibike.citibike\_trips`

WHERE

start\_station\_id != end\_station\_id

GROUP BY

start\_station\_id,

end\_station\_id

ORDER BY

num\_trips DESC

LIMIT

10

Click **Run**. Look at the result and try to determine what this query does ? (Hint: typical duration for the 10 most common one-way rentals)

1. Next, run the below to find another interesting fact: total distance travelled by each bicycle in the dataset. Note that the query limits the results to only top 5.

WITH

trip\_distance AS (

SELECT

bikeid,

ST\_Distance(ST\_GeogPoint(s.longitude,

s.latitude),

ST\_GeogPoint(e.longitude,

e.latitude)) AS distance

FROM

`bigquery-public-data.new\_york\_citibike.citibike\_trips`,

`bigquery-public-data.new\_york\_citibike.citibike\_stations` as s,

`bigquery-public-data.new\_york\_citibike.citibike\_stations` as e

WHERE

start\_station\_id = s.station\_id

AND end\_station\_id = e.station\_id )

SELECT

bikeid,

SUM(distance)/1000 AS total\_distance

FROM

trip\_distance

GROUP BY

bikeid

ORDER BY

total\_distance DESC

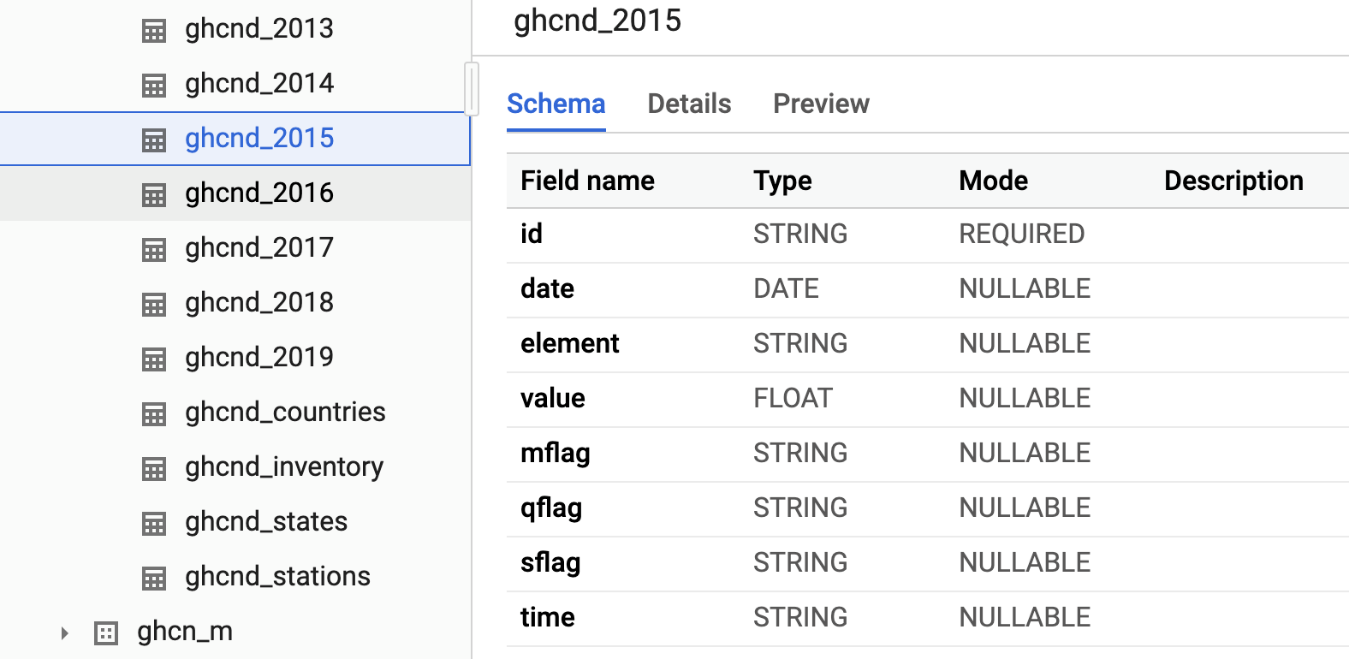
LIMIT

5

**Note:** For this query, we also used the other table in the dataset called **citibike\_stations** to get bicycle station information.

## Explore the weather dataset

In the left pane of the BigQuery Console, select the newly added bigquery-public-data project and select **ghcn\_d** > **ghcnd\_2015**. Then click on the **Preview** tab. Your console should resemble the following:



Examine the columns and some of the data values.

Click **Compose New Query** and enter the following:

SELECT

wx.date,

wx.value/10.0 AS prcp

FROM

`bigquery-public-data.ghcn\_d.ghcnd\_2015` AS wx

WHERE

id = 'USW00094728'

AND qflag IS NULL

AND element = 'PRCP'

ORDER BY

wx.date

Copied!

content\_copy

Click **Run**.

This query will return rainfall (in mm) for all days in 2015 from a weather station in New York whose id is provided in the query (the station corresponds to NEW YORK CNTRL PK TWR )

## Find correlation between rain and bicycle rentals

How about joining the bicycle rentals data against weather data to learn whether there are fewer bicycle rentals on rainy days?

Click **Compose New Query** and enter the following:

WITH bicycle\_rentals AS (

SELECT

COUNT(starttime) as num\_trips,

EXTRACT(DATE from starttime) as trip\_date

FROM `bigquery-public-data.new\_york\_citibike.citibike\_trips`

GROUP BY trip\_date

),

rainy\_days AS

(

SELECT

date,

(MAX(prcp) > 5) AS rainy

FROM (

SELECT

wx.date AS date,

IF (wx.element = 'PRCP', wx.value/10, NULL) AS prcp

FROM

`bigquery-public-data.ghcn\_d.ghcnd\_2015` AS wx

WHERE

wx.id = 'USW00094728'

)

GROUP BY

date

)

SELECT

ROUND(AVG(bk.num\_trips)) AS num\_trips,

wx.rainy

FROM bicycle\_rentals AS bk

JOIN rainy\_days AS wx

ON wx.date = bk.trip\_date

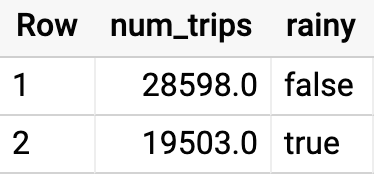
GROUP BY wx.rainy

Copied!

content\_copy

Click **Run**.

Now you can see the results of joining the bicycle rental dataset with a weather dataset that comes from a completely different source.



Running the query yields that, yes, New Yorkers ride the bicycle 47% fewer times when it rains.

## Summary

In this lab you did ad-hoc queries on two datasets. You were able to query the data without setting up any clusters, creating any indexes, etc. You were also able to mash up the two datasets and get some interesting insights. All without ever leaving your browser!

## Congratulations!

**Quizz.**

1. Which of the following are the jobs of a data engineer? (Choose all that apply)

**Get the data into a usable condition**

**Get the data to where it can be useful**

**Add new value to the data**

**Productionize data processes**

**Manage the data**

2. Which of the following statements are true? (Choose TWO)

**BigQuery is optimized for high-read data**

**Cloud SQL is optimized for high-throughput writes**

BigQuery is a row-based storage

Cloud SQL is optimized for high-read data