**Data Engineering for Streaming Data.**

**Introduction.**

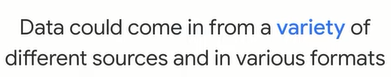
in the previous section of this course you learned about the different layers of the google cloud infrastructure including the categories of big data and machine learning products in this second section you'll explore data engineering for streaming data with the goal of building a real-time data solution with google cloud products and services this includes how to ingest streaming data using pub sub process the data with data flow and visualize the results with google data studio and looker coming up in this section you'll start by examining some of the big data challenges faced by today's data engineers when setting up and managing pipelines next you'll learn about message oriented architecture this includes ways to capture streaming messages globally reliably and at scale so they can be fed into a pipeline from there you'll see how to design streaming pipelines with apache beam and then implement them with dataflow you'll explore how to visualize data insights on a dashboard with looker and data studio and finally you'll get hands-on practice building an end-to-end data pipeline that handles real-time data ingestion with pub sub processing with data flow and visualization with data studio before we get too far let's take a moment to explain what streaming data is how it differs from batch processing and why it's important batch processing is when the processing and analysis happens on a set of stored data an example is payroll and billing systems that have to be processed on either a weekly or monthly basis streaming data is a flow of data records generated by various data sources the processing of streaming data happens as the data flows through a system this results in the analysis and reporting of events as they happen an example would be fraud detection or intrusion detection streaming data processing means that the data is analyzed in near real time and that actions will be taken on the data as quickly as possible modern data processing has progressed from legacy batch processing of data toward working with real-time data streams an example of this is streaming music and movies no longer is it necessary to download an entire movie or album to a local device data streams are a key part in the world of big data.

**Big data challenges.**

Building scalable and reliable pipelines is a core responsibility of data engineers however in modern organizations data engineers and data scientists are facing four major challenges these are collectively known as the four v's they are variety volume velocity and veracity



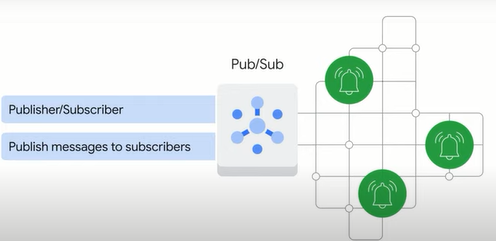
first data could come in from a variety of different sources and in various formats



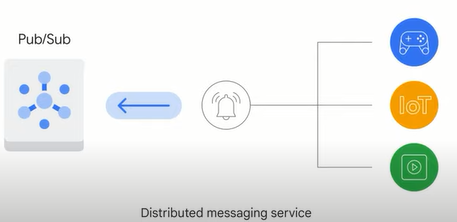
imagine hundreds of thousands of sensors for self-driving cars on roads around the world the data is returned in various formats such as number image or even audio now consider point of sale data from a thousand different stores how do we alert our downstream systems of new transactions in an organized way with no duplicates next let's increase the magnitude of the challenge to handle not only an arbitrary variety of input sources but a volume of data that can vary from gigabytes to petabytes you'll need to know whether your pipeline code and infrastructure can scale with those changes or whether it will grind to a halt or even crash the third challenge concerns velocity data often needs to be processed in near real time as soon as it reaches the system you'll probably also need a way to handle data that arrives late has bad data in the message or needs to be transformed mid-flight because it's streamed into a data warehouse and the fourth major challenge is veracity which refers to the data quality because big data involves a multitude of data dimensions resulting from different data types and sources there's a possibility that gathered data will come with some inconsistencies and uncertainties challenges like these are common considerations for pipeline developers by the end of this section the goal is for you to better understand the tools available to help successfully build a streaming data pipeline and avoid these challenges.

**Message-oriented architecture.**

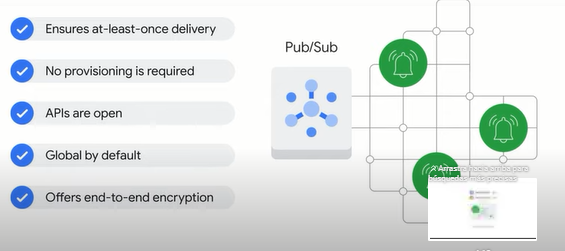
one of the early stages in a data pipeline is data ingestion which is where large amounts of streaming data are received data however may not always come from a single structured database instead the data might stream from a thousand or even a million different events that are all happening asynchronously a common example of this is data from iot or internet of things applications these can include sensors on taxes that send out location data every 30 seconds or temperature sensors around a data center to help optimize heating and cooling these iot devices present new challenges to data ingestion which can be summarized in four points the first is that data can be streamed from many different methods and devices many of which might not talk to each other and might be sending bad or delayed data the second is that it can be hard to distribute event messages to the right subscribers event messages are notifications a method is needed to collect the streaming messages that come from iot sensors and broadcast them to the subscribers as needed the third is that data can arrive quickly and at high volumes services must be able to support this and the fourth challenge is ensuring services are reliable secure and perform as expected google cloud has a tool to handle distributed message-oriented architectures at scale and that's pub/sub the name is short for publisher subscriber or published messages to subscribers



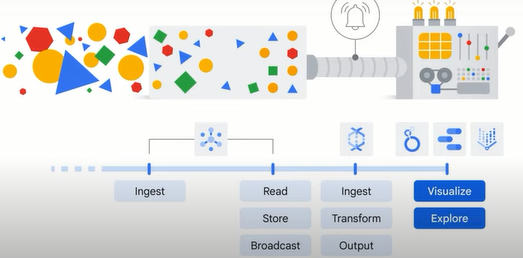
pub/sub is a distributed messaging service that can receive messages from a variety of device streams such as gaming events iot devices and application streams



it ensures at least once delivery of received messages to subscribing applications with no provisioning required pub subs apis are open the service is global by default and it offers end-to-end encryption



let's explore the end-to-end pub sub architecture upstream source data comes in from devices all over the globe and is ingested into pub sub which is the first point of contact within the system pub sub reads stores and broadcasts to any subscribers of this data topic that new messages are available as a subscriber of pub sub dataflow can ingest and transform those messages in an elastic streaming pipeline and output the results into an analytics data warehouse like bigquery finally you can connect a data visualization tool like looker or data studio to visualize and monitor the results of a pipeline or an ai or ml tool such as vertex ai to explore the data to uncover business insights or help with predictions



a central element of pub sub is the topic you can think of a topic like a radio antenna whether your radio is playing music or it's turned off the antenna itself is always there if music is being broadcast in a

03:02

frequency that nobody's listening to the stream of music still exists similarly a publisher can send data to a topic that has no subscriber to receive it or a subscriber can be waiting for data from a topic that isn't getting data sent to it like listening to static from

03:17

a bad radio frequency or you could have a fully operational pipeline where the publisher is sending data to a topic that an application is subscribed to that means there can be zero one or more publishers and zero one or more subscribers related to a topic and they're completely decoupled so

03:37

they're free to break without affecting their counterparts it's helpful to describe this using an example say you've got a human resources topic a new employee joins your company and several applications across the company need to be updated adding a new employee can be an event that generates a notification to the

03:56

other applications that are subscribed to the topic and they'll receive the message about the new employees starting now let's assume that there are two different types of employee a full-time employee and a contractor both sources of employee data could have no knowledge of the other but still published their events saying this

04:15

employee joined into the pub sub hr topic after pub sub receives a message downstream applications like the directory service facilities system account provisioning and badge activation systems can all listen and process their own next steps independent of one another pubsub is a good solution to buffer changes for lightly coupled

04:37

architectures like this one that have many different sources and sinks pubsub supports many different inputs and outputs and you can even publish a pubsub event from one topic to another the next task is to get these messages reliably into our data warehouse and we'll need a pipeline that can match

04:55

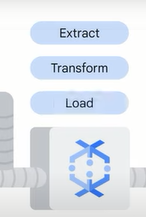
pub subs scale and elasticity to do it

**Designing streaming pipelines with Apache Beam.**

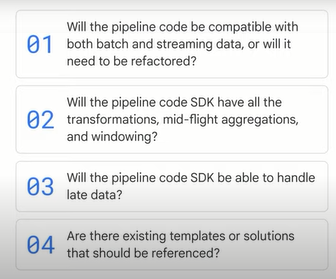
after messages have been captured from the streaming input sources you need a way to pipe that data into a data warehouse for analysis this is where dataflow comes in dataflow creates a pipeline to process both streaming data and batch data process



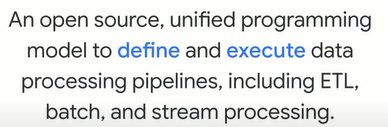
in this case refers to the steps to extract transform and load data or etl



when building a data pipeline data engineers often encounter challenges related to coding the pipeline design and implementing and serving the pipeline at scale during the pipeline design phase there are a few questions to consider will the pipeline code be compatible with both batch and streaming data or will it need to be refactored will the pipeline code software development kit or sdk being used have all the transformations mid-flight aggregations and windowing and be able to handle late data are there existing templates or solutions that should be referenced



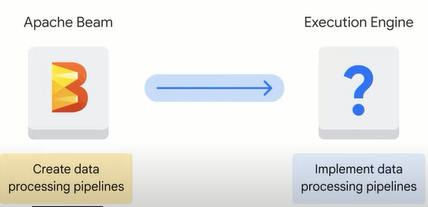
a popular solution for pipeline design is **apache beam** it's an open source unified programming model to define and execute data processing pipelines including etl batch and stream processing



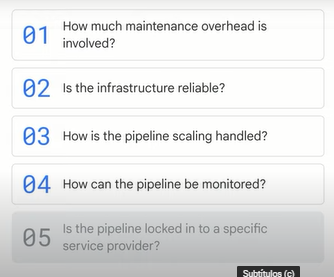
apache beam is unified which means it uses a single programming model for both batch and streaming data it's portable which means it can work on multiple execution environments like dataflow and apache spark among others and it's extensible which means it allows you to write and share your own connectors and transformation libraries apache beam provides pipeline templates so you don't need to build a pipeline from nothing and it can write pipelines in java python or go the apache beam software development kit sdk is a collection of software development tools in one installable package it provides a variety of libraries for transformations and data connectors to sources and sinks apache beam creates a model representation from your code that's portable across many runners runners pass off your model for execution on a variety of different possible engines with dataflow being a popular choice.

**Implementing streaming pipelines on Cloud Dataflow.**

as covered in the previous video apache beam can be used to create data processing pipelines the next step is to identify an execution engine to implement those pipelines

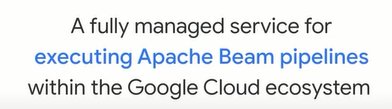


when choosing an execution engine for your pipeline code it may be helpful to consider the following questions how much maintenance overhead is involved is the infrastructure reliable how is the pipeline scaling handled how can the pipeline be monitored is the pipeline locked into a specific service provider

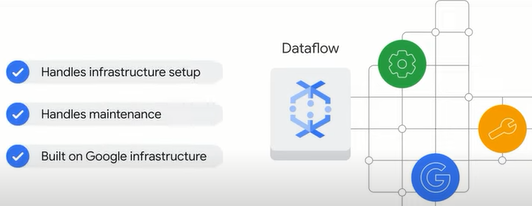


this brings us to **dataflow**

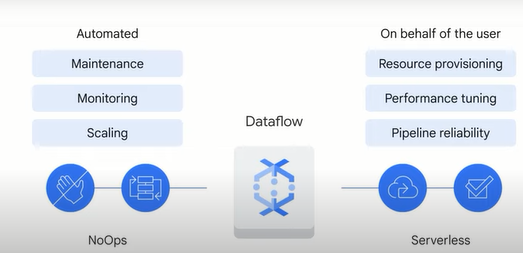
**dataflow** is a fully managed service for executing apache beam pipelines within the google cloud ecosystem



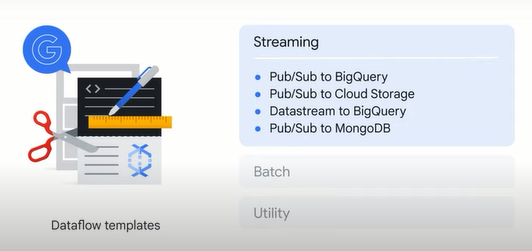
dataflow handles much of the complexity relating to infrastructure setup and maintenance and is built on google's infrastructure

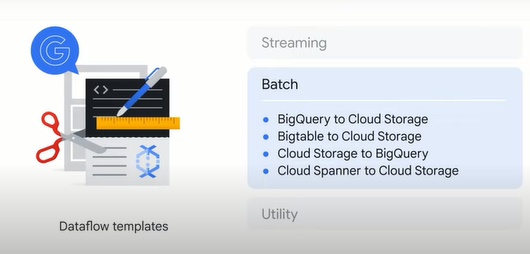


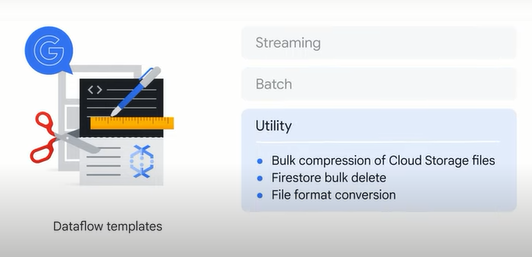
this allows for reliable auto scaling to meet data pipeline demands dataflow is serverless and no ops which means no operations but what does that mean exactly a no-ops environment is one that doesn't require management from an operations team because maintenance monitoring and scaling are automated serverless computing is a cloud computing execution model this is when google cloud for example manages infrastructure tasks on behalf of the users this includes tasks like resource provisioning performance tuning and ensuring pipeline reliability



using a serverless and knob solution like dataflow means that you can spend more time analyzing the insights from your data sets and less time provisioning resources to ensure that your pipeline will successfully complete its next cycles it's designed to be low maintenance let's explore the tasks dataflow performs when a job is received it starts by optimizing a pipeline model's execution graph to remove any inefficiencies next it schedules out distributed work to new workers and scales as needed after that it auto heals any worker faults from there it automatically rebalances efforts to most efficiently use its workers and finally it outputs data to produce a result bigquery is one of many options that data can be outputted to you'll get some more practice using bigquery later in this course so by design you don't need to monitor all of the compute and storage resources that dataflow manages to fit the demand of a streaming data pipeline even experienced java or python developers will benefit from using dataflow templates which cover common use cases across google cloud products the list of templates is continuously growing they can be broken down into three categories streaming templates batch templates and utility templates streaming templates are for processing continuous or real-time data for example pub sub to bigquery pubs up to cloud storage data stream to bigquery and pubsub to mongodb batch templates are for processing bulk data or batch load data for example bigquery to cloud storage bigtable to cloud storage cloud storage to bigquery and cloud spanner to cloud storage finally utility templates address activities related to bulk compression deletion and conversion for a complete list of templates please refer to the reading list.







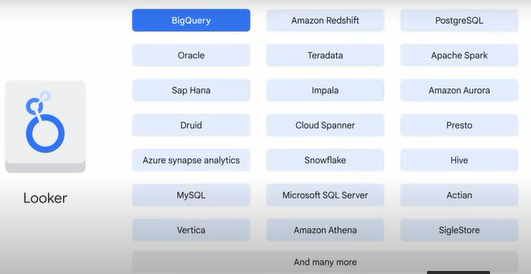
**Visualization with Looker.**

telling a good story with data through a dashboard can be critical to the success of a data pipeline because data that's difficult to interpret or draw insights from might be useless after data is in bigquery a lot of skill and effort can still be required to uncover insights to help create an environment where stakeholders can easily interact with and visualize data google cloud offers two solutions looker and google data studio



let's explore both of them starting with **looker**

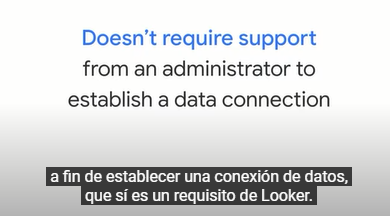
**looker** supports bigquery as well as more than 60 different types of sql database products commonly referred to as dialects



it allows developers to define a semantic modeling layer on top of databases using looker modeling language or lookml lookml defines logic and permissions independent from a specific database or a sql language which frees a data engineer from interacting with individual databases to focus more on business logic across an organization the looker platform is 100 web-based which makes it easy to integrate into existing workflows and share with multiple teams at an organization there's also a looker api which can be used to embed looker reports in other applications let's explore some of looker's features starting with dashboards dashboards like the business pulse dashboard for example can visualize data in a way that makes insights easy to understand for a sales organization it shows figures that many might want to see at the start of the week like the number of new users acquired monthly sales trends and even the number of year-to-date orders information like this can help align teams identify customer frustrations and maybe even uncover lost revenue based on the metrics that are important to your business you can create looker dashboards that provide straightforward presentations to help you and your colleagues quickly see a high level business status looker has multiple data visualization options including area charts line charts sankey diagrams funnels and liquid fill gauges to share a dashboard with your team you schedule delivery through storage services like google drive slack and dropbox let's explore another look at dashboard this time one that monitors key metrics related to new york city taxes over a period of time this dashboard displays total revenue total numbers of passengers and total number of rides looker displays this information through a time series to help monitor metrics over time looker also lets you plot data on a map to see ride distribution busy areas and peak hours the purpose of these features is to help you draw insights to make business decisions for more training on looker please refer to cloud.google.com forward slash training

**Visualization with Data Studio.**

another popular data visualization tool offered by google is data studio data studio is integrated into bigquery which makes data visualization possible with just a few clicks this means that leveraging data studio doesn't require support from an administrator to establish a data connection which is a requirement with looker

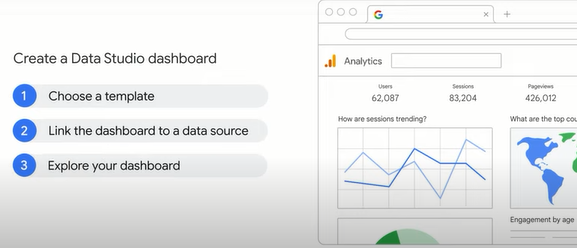


beta studio dashboards are widely used across many google products and applications for example beta studio is integrated into google analytics to help visualize in this case a summary of a marketing website this dashboard visualizes the total number of visitors through a map compares month-over-month trends and even displays visitor distribution by age another data studio integration is the google cloud billing dashboard you might be familiar with this from your account maybe you've already used it to monitor spending for example you'll soon have hands-on practice with data studio but in preparation for the lab let's explore the three steps needed to create a data studio dashboard

first choose a template you can start with either a pre-built template or a blank report,

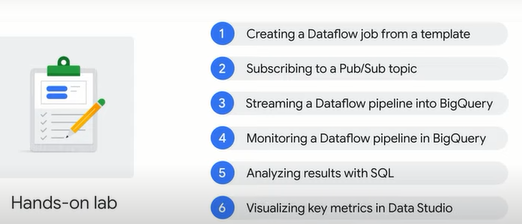
second link the dashboard to a data source this might come from bigquery a local file or a google application like google sheets or google analytics or a combination of any of these sources,

and third explore your dashboard



**Lab introduction: Creating a streaming data pipeline for a Real-Time dashboard with Dataflow.**

now it's time for hands-on practice with some of the tools you learned about in this section of the course in the lab that follows this video you build a streaming data pipeline to monitor sensor data and then visualize the data set through a dashboard your practice creating a data flow job from a pre-existing template and subscribing to a pub sub topic streaming and monitoring a data flow pipeline into bigquery analyzing results with sql and visualizing key metrics in data studio please note that though you will use some sql commands in this lab the lab doesn't actually require strong sql knowledge we'll explore bigquery in more detail later in this course you'll have multiple attempts at each lab so if you don't complete it the first time or if you want to experiment more with it later on you can return and start a new instance.



**Lab: Creating a streaming data pipeline for a Real-Time dashboard with Dataflow.**

**Overview**

In this lab, you own a fleet of New York City taxi cabs and are looking to monitor how well your business is doing in real-time. You will build a streaming data pipeline to capture taxi revenue, passenger count, ride status, and much more and visualize the results in a management dashboard.

## **Task 1. Source a public Pub/Sub topic and create a BigQuery dataset**

[Pub/Sub](https://cloud.google.com/pubsub/) is an asynchronous global messaging service. By decoupling senders and receivers, it allows for secure and highly available communication between independently written applications. Pub/Sub delivers low-latency, durable messaging.

In Pub/Sub, publisher applications and subscriber applications connect with one another through the use of a shared string called a **topic**. A publisher application creates and sends messages to a topic. Subscriber applications create a subscription to a topic to receive messages from it.

Google maintains a few public Pub/Sub streaming data topics for labs like this one. We'll be using the [NYC Taxi & Limousine Commission’s open dataset](https://data.cityofnewyork.us/).

[BigQuery](https://cloud.google.com/bigquery/) is a serverless data warehouse. Tables in BigQuery are organized into datasets. In this lab, messages published into Pub/Sub will be aggregated and stored in BigQuery.

To create a new BigQuery dataset:

### **Option 1: The command-line tool**

1. Open **Cloud Shell** (Cloud Shell icon) and run the below command to create the taxirides dataset.

bq --location=us-west1 mk taxirides

1. Run this command to create the taxirides.realtime table (empty schema that you will stream into later).

bq --location=us-west1 mk \

--time\_partitioning\_field timestamp \

--schema ride\_id:string,point\_idx:integer,latitude:float,longitude:float,\

timestamp:timestamp,meter\_reading:float,meter\_increment:float,ride\_status:string,\

passenger\_count:integer -t taxirides.realtime

### **Option 2: The BigQuery Console UI**

**Note:** Skip these steps if you created the tables using the command line.

1. In the Google Cloud Console, select **Navigation menu** > **Analytics** > **BigQuery**:
2. 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.
3. Click on the **View actions** icon next to your Project ID and click **Create dataset**.
4. Set the **Dataset ID** as **taxirides**, for **Data location**, select **us-west1 (Oregon)** leave all the other fields the way they are, and click **CREATE DATASET**.
5. If you look at the left-hand resources menu, you should see your newly created dataset.
6. Click on the **View actions** icon next to the **taxirides** dataset and click **Open in current tab**.
7. Click **CREATE TABLE**.
8. Name the table **realtime**
9. For the schema, click **Edit as text** and paste in the below:

ride\_id:string,

point\_idx:integer,

latitude:float,

longitude:float,

timestamp:timestamp,

meter\_reading:float,

meter\_increment:float,

ride\_status:string,

passenger\_count:integer

Copied!

content\_copy

1. Under **Partition and cluster settings**, select the **timestamp** option for the Partitioning field.
2. Click the **CREATE TABLE** button.

## Task 2. Create a Cloud Storage bucket

[Cloud Storage](https://cloud.google.com/storage/) allows world-wide storage and retrieval of any amount of data at any time. You can use Cloud Storage for a range of scenarios including serving website content, storing data for archival and disaster recovery, or distributing large data objects to users via direct download. In this lab, you use Cloud Storage to provide working space for your Dataflow pipeline.

1. In the Cloud Console, go to **Navigation menu** > **Cloud Storage**.
2. Click **CREATE BUCKET**.
3. For **Name**, paste in your **GCP Project ID** and then click **Continue**.
4. For **Location type**, click **Multi-region** if it is not already selected.
5. Click **CREATE**.

## Task 3. Set up a Dataflow Pipeline

[Dataflow](https://cloud.google.com/dataflow/) is a serverless way to carry out data analysis. In this lab, you set up a streaming data pipeline to read sensor data from Pub/Sub, compute the maximum temperature within a time window, and write this out to BigQuery.

Restart the connection to the Dataflow API.

1. In the Cloud Console, enter **Dataflow API** in the top search bar.
2. Click on the result for **Dataflow API**.
3. Click **Manage**.
4. Click **Disable API**.
5. If asked to confirm, click **Disable**.
6. Click **Enable**.

To create a new streaming pipeline:

1. In the Cloud Console, go to **Navigation menu** > **Dataflow**.
2. In the top menu bar, click **CREATE JOB FROM TEMPLATE**.
3. Enter **streaming-taxi-pipeline** as the Job name for your Dataflow job.
4. Under **Regional endpoint**, select **us-west1 (Oregon)**.
5. Under **Dataflow template**, select the **Pub/Sub Topic to BigQuery** template.
6. Under **Input Pub/Sub topic**, click **Enter topic Manually**, enter projects/pubsub-public-data/topics/taxirides-realtime
7. Click **Save**.
8. Under **BigQuery output table**, enter <myprojectid>:taxirides.realtime

**Note**: There is a colon : between the project and dataset name and a dot . between the dataset and table name.

1. Under **Temporary location**, enter gs://<mybucket>/tmp/.
2. Click **Show Optional Parameters** and input the following values as listed below:

* **Max workers:** 2
* **Number of workers:** 2
* **Worker region:** us-west1

1. Click the **RUN JOB** button.

A new streaming job has started! You can now see a visual representation of the data pipeline.

**Note**: If the dataflow job fails for the first time then re-create a new job template with new job name and run the job.

## Task 4. Analyze the taxi data using BigQuery

To analyze the data as it is streaming:

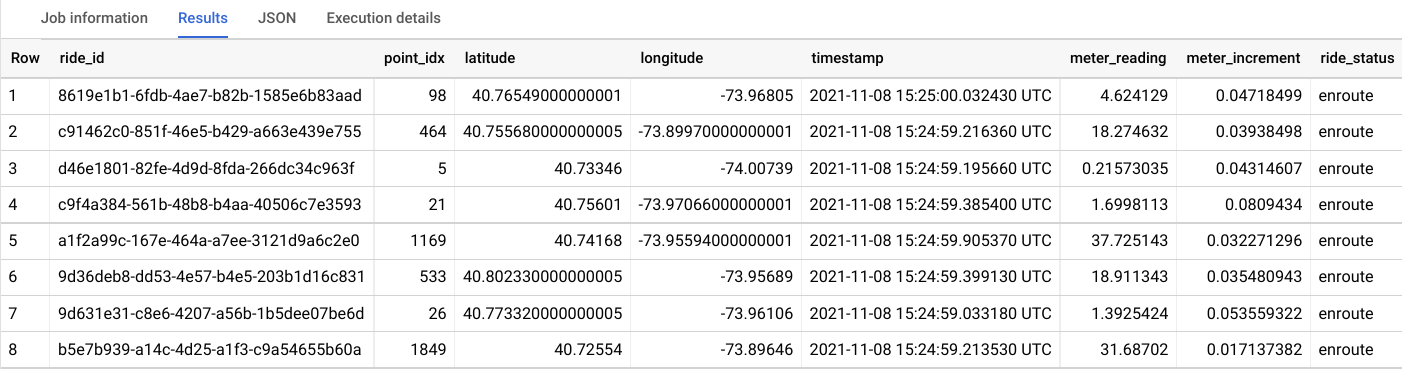
1. In the Cloud Console, select **Navigation menu** > **BigQuery**.
2. Enter the following query in the query **EDITOR** and click **RUN**:

SELECT \* FROM taxirides.realtime LIMIT 10

Copied!

content\_copy

1. If no records are returned, wait another minute and re-run the above query (Dataflow takes 3-5 minutes to setup the stream). You will receive a similar output:



## Task 5. Perform aggregations on the stream for reporting

1. Copy and paste the below query and click **RUN**.

WITH streaming\_data AS (

SELECT

timestamp,

TIMESTAMP\_TRUNC(timestamp, HOUR, 'UTC') AS hour,

TIMESTAMP\_TRUNC(timestamp, MINUTE, 'UTC') AS minute,

TIMESTAMP\_TRUNC(timestamp, SECOND, 'UTC') AS second,

ride\_id,

latitude,

longitude,

meter\_reading,

ride\_status,

passenger\_count

FROM

taxirides.realtime

WHERE ride\_status = 'dropoff'

ORDER BY timestamp DESC

LIMIT 1000

)

# calculate aggregations on stream for reporting:

SELECT

ROW\_NUMBER() OVER() AS dashboard\_sort,

minute,

COUNT(DISTINCT ride\_id) AS total\_rides,

SUM(meter\_reading) AS total\_revenue,

SUM(passenger\_count) AS total\_passengers

FROM streaming\_data

GROUP BY minute, timestamp

Copied!

content\_copy

**Note**: Ensure dataflow is registering data in BigQuery before proceeding to the next task.

The result shows key metrics by the minute for every taxi drop-off.

## Task 6. Stop the Dataflow job

1. Navigate back to **Dataflow**.
2. Click the **streaming-taxi-pipeline** or the new job name.
3. Click **STOP** and select **Cancel** > **STOP JOB**.

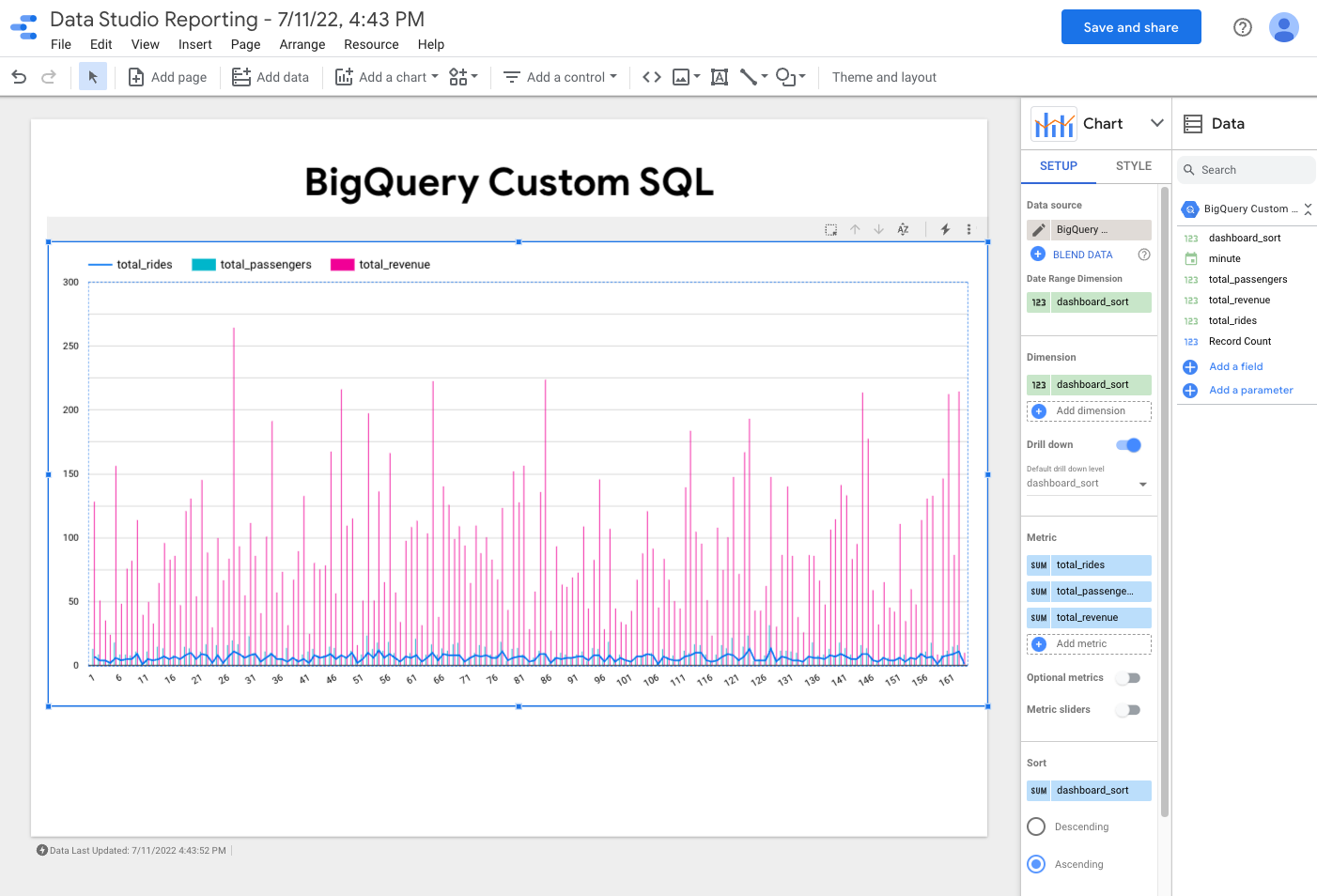
This will free up resources for your project.

## Task 7. Create a real-time dashboard

1. Open this [Google Data Studio link](https://datastudio.google.com/) in a new incognito browser tab.
2. On the **Reports** page, in the **Start with a Template** section, click the **[+] Blank Report** template.
3. To get started complete account setup, select your **Country** from the drop down, enter **Company** if applicable.
4. Check the checkbox to acknowledge the Google Data Studio Additional Terms, and click **Continue**.
5. Select **No** to all the questions, then click **Continue**.
6. Switch back to the **BigQuery** Console.
7. Click **EXPLORE DATA > Explore with Data Studio** in BigQuery page.
8. Specify the below settings:

* **Chart type:** Combo chart
* **Date range Dimension:** dashboard\_sort
* **Dimension:** dashboard\_sort
* **Drill Down:** dashboard\_sort (Make sure that Drill down option is turned ON)
* **Metric:** SUM() total\_rides, SUM() total\_passengers, SUM() total\_revenue
* **Sort:** dashboard\_sort, Ascending (latest rides first)

Your chart should look similar to this:



**Note:** Visualizing data at a minute-level granularity is currently not supported in Data Studio as a timestamp. This is why we created our own dashboard\_sort dimension.

1. When you're happy with your dashboard, click **Save and share** to save this data source.
2. If prompted with the **Review data access before saving** window, click **Acknowledge and save**.
3. Click **Add to report**.
4. Whenever anyone visits your dashboard, it will be up-to-date with the latest transactions. You can try it yourself by clicking on the **More option** and **Refresh data**.

## Task 8. Create a time series dashboard

1. Click this [Google Data Studio link](https://datastudio.google.com/) to open Data Studio in a new browser tab.
2. On the **Reports** page, in the **Start with a Template** section, click the **[+] Blank Report** template.
3. A new, empty report opens with **Add data to report**.
4. From the list of **Google Connectors**, select the **BigQuery** tile.
5. Under **CUSTOM QUERY**, click **qwiklabs-gcp-xxxxxxx** > **Enter Custom Query**, add the following query.

SELECT

\*

FROM

taxirides.realtime

WHERE

ride\_status='dropoff'

Copied!

content\_copy

1. Click **Add > ADD TO REPORT**.

### **Create a time series chart**

1. In the **Data** panel, click **ADD A FIELD**. Click **All Fields** on the left corner.
2. Change the field **timestamp** type to **Date & Time > Date Hour Minute (YYYYMMDDhhmm)**.
3. Click **Continue** and then click **Done**.
4. Click **Add a chart**.
5. Choose **Time series chart**.
6. Position the chart in the bottom left corner - in the blank space.
7. In the **Data** panel on the right, change the following:

* **Dimension:** timestamp
* **Metric:** meter\_reading(SUM)

Your time series chart should look similar to this:



**Note:** if Dimension is timestamp(Date), then click on calendar icon next to timestamp(Date), and select type to **Date & Time > Date Hour Minute**.

## Congratulations!

**Summary.**

well done on completing the lab on building a data pipeline for streaming data before you move on in the course let's do a quick recap in this section you explore the streaming data workflow from ingestion to visualization you started by learning about the four common big data challenges or the four v's volume data size variety data format velocity data speed and veracity data accuracy these challenges can be especially common when dealing with streaming data you then learned how the streaming data workflow provided by google can help address these challenges you started with pub sub which can be

data from diverse resources in various formats after that you explore dataflow a serverless no op service to process the data process here refers to etl extract transform and load and finally you're introduced to two google visualization tools looker and data studio.

**Quiz**

1.When you build scalable and reliable pipelines, data often needs to be processed in near-real time, as soon as it reaches the system. Which type of challenge might this present to data engineers?

Variety

Volume

**Velocity**

Veracity

2.Select the correct streaming data workflow.

**Ingest the streaming data, process the data, and visualize the results.**

Visualize the data, process the data, and ingest the streaming data.

Ingest the streaming data, visualize the data, and process the data.

Process the data, visualize the data, and ingest the data.

3. Which Google Cloud product acts as an execution engine to process and implement data processing pipelines?

Apache Beam

Looker

Data Studio

**Dataflow**

4. Which Google Cloud product is a distributed messaging service that is designed to ingest messages from multiple device streams such as gaming events, IoT devices, and application streams?

Looker

**Pub/Sub**

Data Studio

Apache Beam

5.Due to several data types and sources, big data often has many data dimensions. This can introduce data inconsistencies and uncertainties. Which type of challenge might this present to data engineers?

Variety

Volume

Velocity

**Veracity**