Perform Foundational Data, ML, and AI Tasks in Google Cloud Challenge Lab

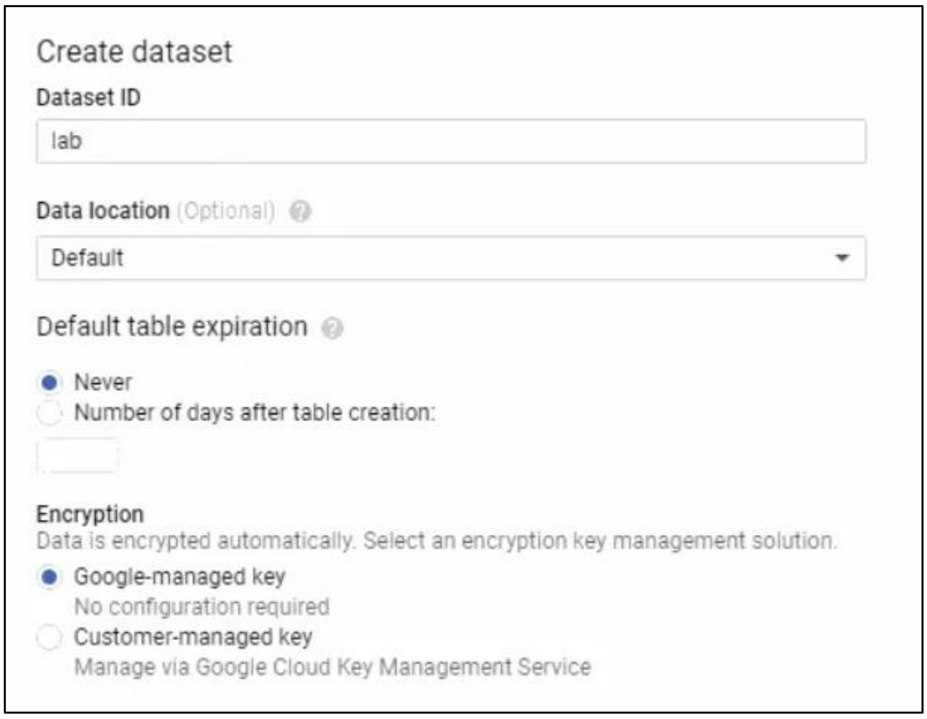
Author: Vedant Kakde | **GitHub Profile:** github.com/vedant-kakde | **LinkedIn Profile:** linkedin.com/in/vedant-kakde/

Task 1: Run a simple Dataflow job

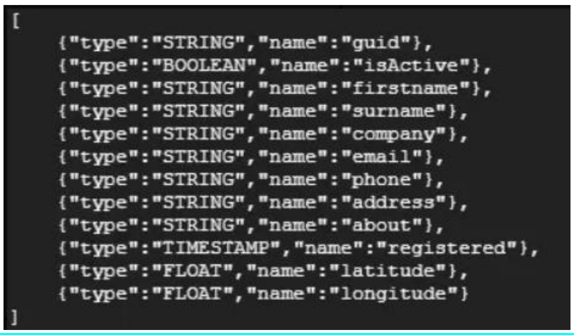
In this task, you have to transfer the data in a CSV file to BigQuery using Dataflow via Pub/Sub. First of all, you need to create a BigQuery dataset called `lab` and a Cloud Storage bucket called with your project ID.

1.1 Created a BigQuery dataset called `lab`

1. In the Cloud Console, click on **Navigation Menu** > **BigQuery**.
2. Select your project in the left pane.
3. Click **CREATE DATASET**.
4. Enter `lab` in the Dataset ID, then click **Create dataset**.

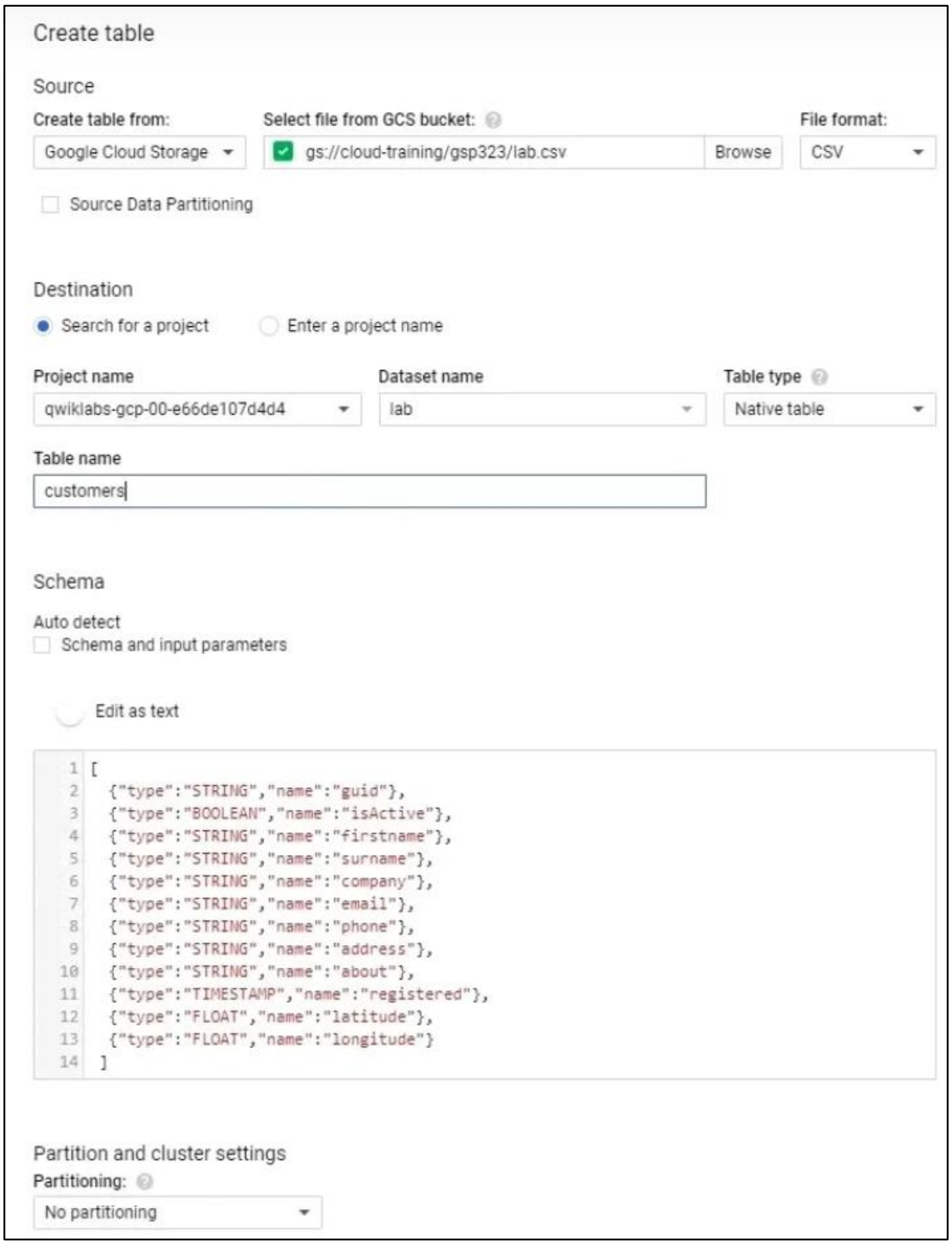

Create dataset
Dataset ID
lab
Data location (Optional)
Default
Default table expiration
● Never
○ Number of days after table creation:
Encryption
Data is encrypted automatically. Select an encryption key management solution.
● Google-managed key
No configuration required
○ Customer-managed key
Manage via Google Cloud Key Management Service

5. Run `gsutil cp gs://cloud-training/gsp323/lab.schema .` in the Cloud Shell to download the schema file.
6. View the schema by running `cat lab.schema`.


```
[
    {"type":"STRING","name":"guid"},
    {"type":"BOOLEAN","name":"isActive"},
    {"type":"STRING","name":"firstname"},
    {"type":"STRING","name":"surname"},
    {"type":"STRING","name":"company"},
    {"type":"STRING","name":"email"},
    {"type":"STRING","name":"phone"},
    {"type":"STRING","name":"address"},
    {"type":"STRING","name":"about"},
    {"type":"TIMESTAMP","name":"registered"},
    {"type":"FLOAT","name":"latitude"},
    {"type":"FLOAT","name":"longitude"}
]
```

Copy this schema into notepad. We will use it in further step.

7. Go back to the Cloud Console, select the new dataset **lab** and click **Create Table**.
8. In the Create table dialog, select **Google Cloud Storage** from the dropdown in the Source section.
9. Copy `gs://cloud-training/gsp323/lab.csv` to **Select file from GCS bucket**.
10. Enter `customers` to "Table name" in the Destination section.
11. Enable **Edit as text** and copy the JSON data from the `lab.schema` file to the textarea in the Schema section.
12. Click **Create table**.



```
[
  {"type":"STRING","name":"guid"},
  {"type":"BOOLEAN","name":"isActive"},
  {"type":"STRING","name":"firstname"},
  {"type":"STRING","name":"surname"},
  {"type":"STRING","name":"company"},
  {"type":"STRING","name":"email"},
  {"type":"STRING","name":"phone"},
  {"type":"STRING","name":"address"},
  {"type":"STRING","name":"about"},
  {"type":"TIMESTAMP","name":"registered"},
  {"type":"FLOAT","name":"latitude"},
  {"type":"FLOAT","name":"longitude"}
 ]
```

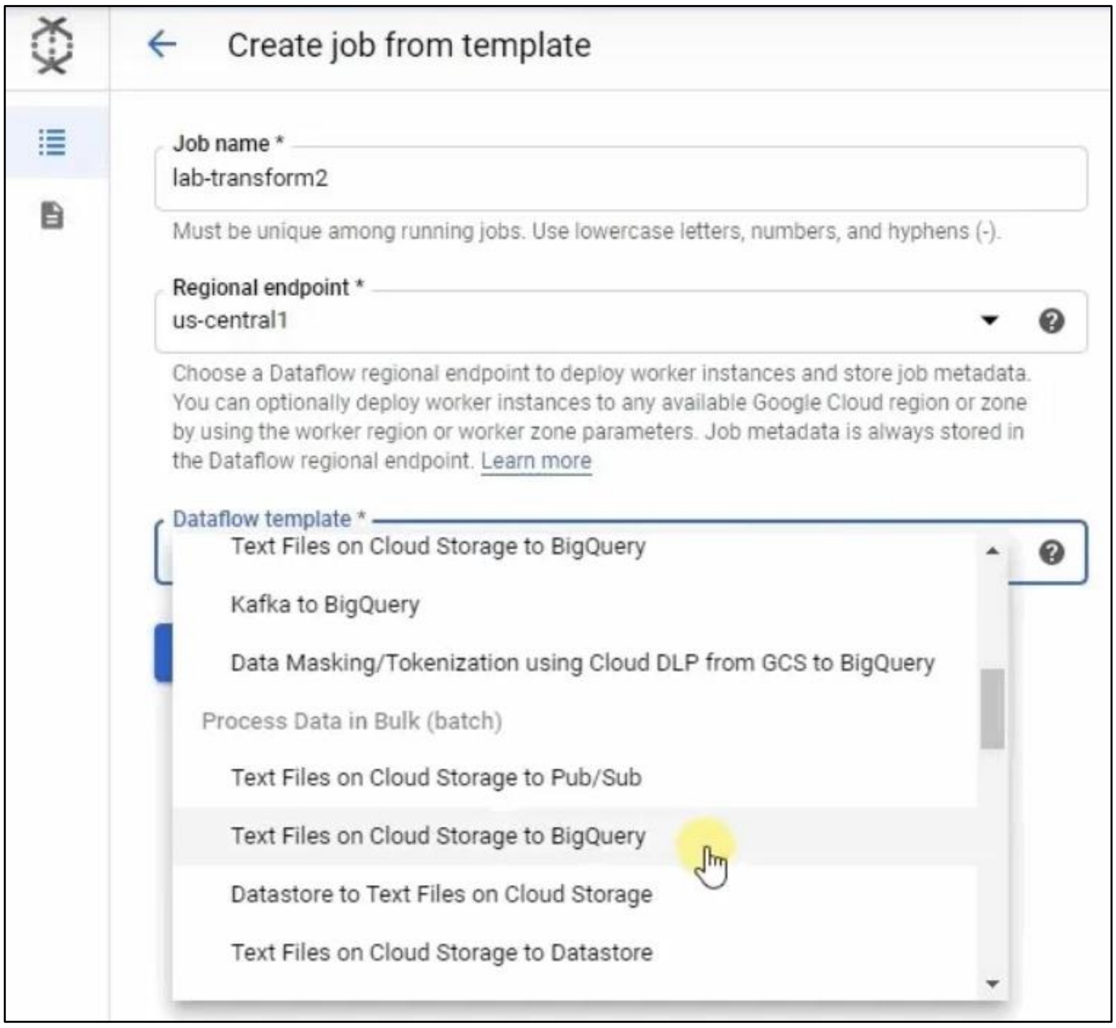
1.2 Create a Cloud Storage bucket

1. In the Cloud Console, click on **Navigation Menu** > **Storage**.
2. Click **CREATE BUCKET**.
3. Copy your GCP Project ID to Name your bucket.

4. Click **CREATE**.

1.3 Create a Dataflow job

1. In the Cloud Console, click on **Navigation Menu** > **Dataflow**.
2. Click **CREATE JOB FROM TEMPLATE**.
3. In Create job from template, give an arbitrary job name.
4. From the dropdown under Dataflow template, select **Text Files on Cloud Storage Pub/Sub** under “Process Data in Bulk (batch)”. (**DO NOT** select the item under “Process Data Continuously (stream)”).



Create job from template

Job name *
lab-transform2
Must be unique among running jobs. Use lowercase letters, numbers, and hyphens (-).

Regional endpoint *
us-central1
Choose a Dataflow regional endpoint to deploy worker instances and store job metadata. You can optionally deploy worker instances to any available Google Cloud region or zone by using the worker region or worker zone parameters. Job metadata is always stored in the Dataflow regional endpoint. Learn more

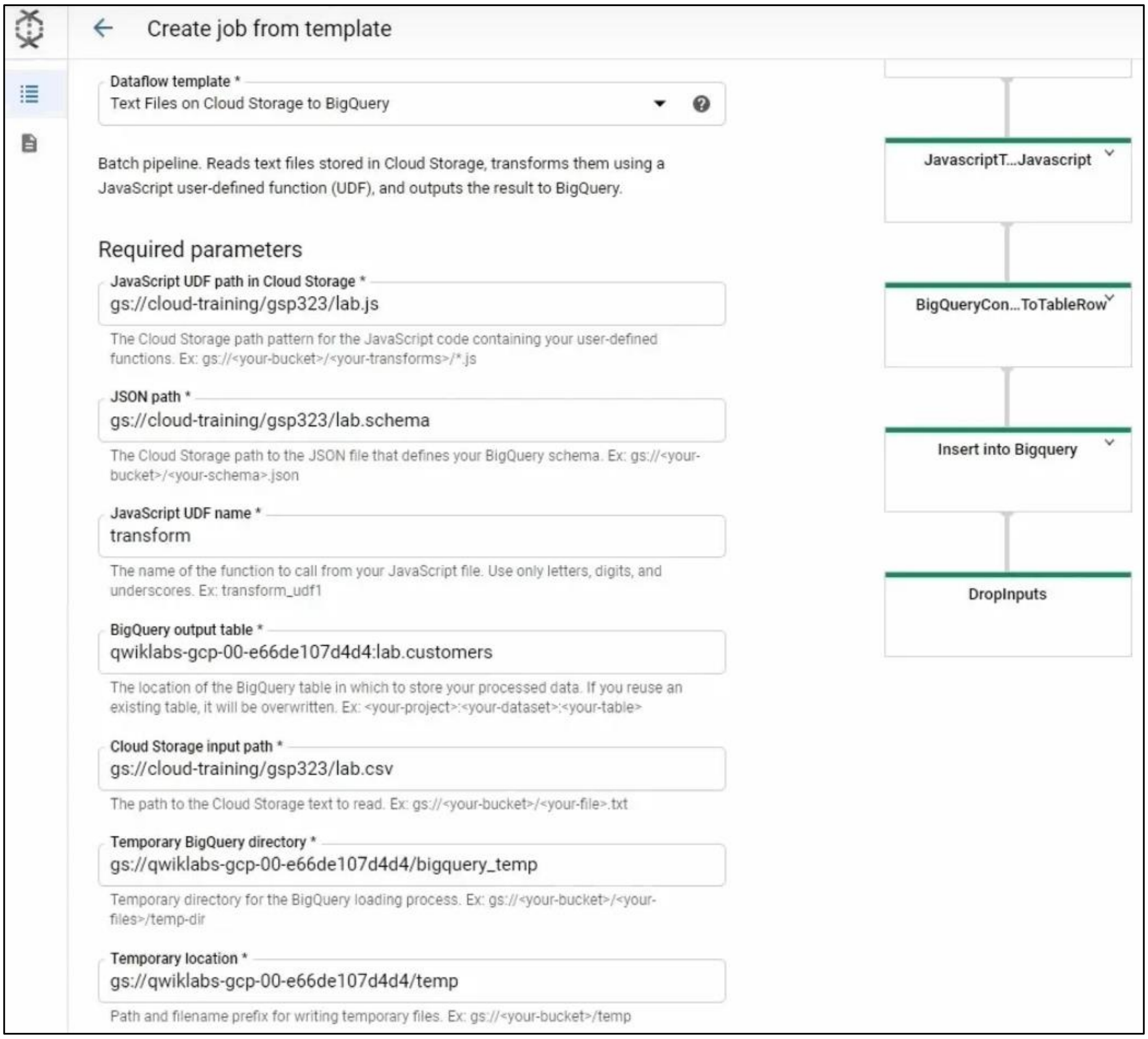
Dataflow template *
Text Files on Cloud Storage to BigQuery
Kafka to BigQuery
Data Masking/Tokenization using Cloud DLP from GCS to BigQuery
Process Data in Bulk (batch)
Text Files on Cloud Storage to Pub/Sub
Text Files on Cloud Storage to BigQuery
Datastore to Text Files on Cloud Storage
Text Files on Cloud Storage to Datastore

5. Under the Required parameters, enter the following values:

Field	**Value**
JavaScript UDF path in Cloud Storage	`gs://cloud-training/gsp323/lab.js`
JSON path	`gs://cloud-training/gsp323/lab.schema`
JavaScript UDF name	`transform`
BigQuery output table	`YOUR_PROJECT:lab.customers`

Field	Value
Cloud Storage input path	`gs://cloud-training/gsp323/lab.csv`
Temporary BigQuery directory	`gs://YOUR_PROJECT/bigquery_temp`
Temporary location	`gs://YOUR_PROJECT/temp`

Replace `YOUR_PROJECT` with your project ID.



Dataflow template *
Text Files on Cloud Storage to BigQuery

Batch pipeline. Reads text files stored in Cloud Storage, transforms them using a JavaScript user-defined function (UDF), and outputs the result to BigQuery.

Required parameters

JavaScript UDF path in Cloud Storage *
gs://cloud-training/gsp323/lab.js
The Cloud Storage path pattern for the JavaScript code containing your user-defined functions. Ex: gs://<your-bucket>/<your-transforms>/*.js

JSON path *
gs://cloud-training/gsp323/lab.schema
The Cloud Storage path to the JSON file that defines your BigQuery schema. Ex: gs://<your-bucket>/<your-schema>.json

JavaScript UDF name *
transform
The name of the function to call from your JavaScript file. Use only letters, digits, and underscores. Ex: transform_udf1

BigQuery output table *
qwiklabs-gcp-00-e66de107d4d4:lab.customers
The location of the BigQuery table in which to store your processed data. If you reuse an existing table, it will be overwritten. Ex: <your-project>:<your-dataset>:<your-table>

Cloud Storage input path *
gs://cloud-training/gsp323/lab.csv
The path to the Cloud Storage text to read. Ex: gs://<your-bucket>/<your-file>.txt

Temporary BigQuery directory *
gs://qwiklabs-gcp-00-e66de107d4d4/bigquery_temp
Temporary directory for the BigQuery loading process. Ex: gs://<your-bucket>/<your-files>/temp-dir

Temporary location *
gs://qwiklabs-gcp-00-e66de107d4d4/temp
Path and filename prefix for writing temporary files. Ex: gs://<your-bucket>/temp

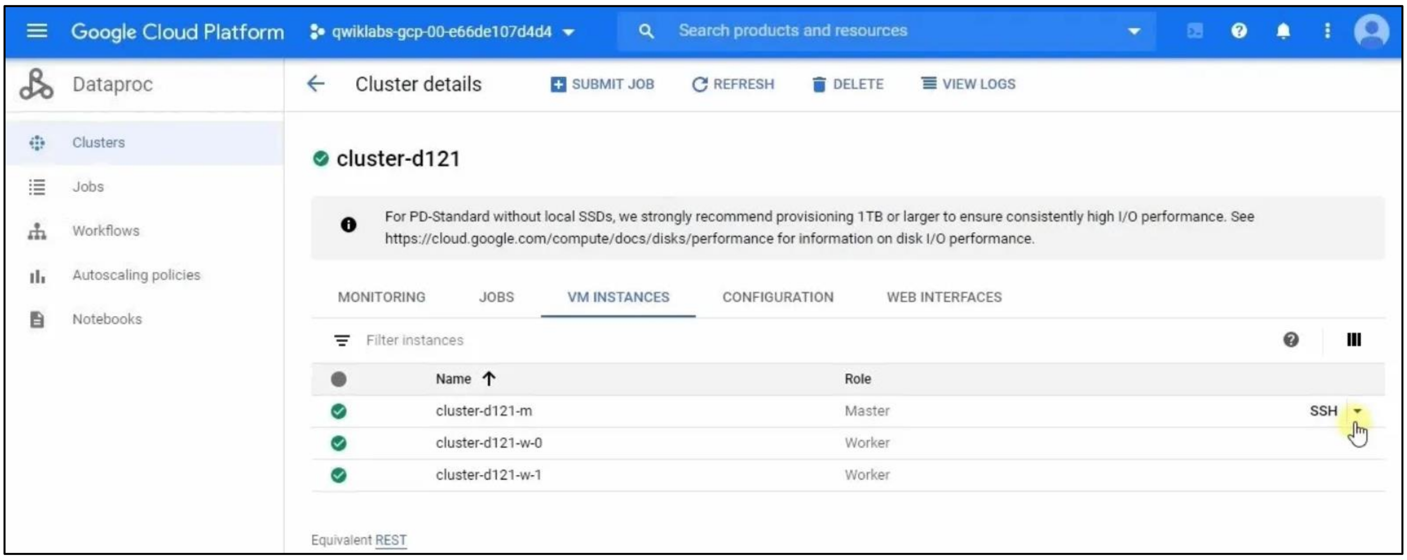
6. Click **RUN JOB**.

Task 2: Run a simple Dataproc job

Create a Dataproc cluster

1. In the Cloud Console, click on **Navigation Menu** > **Dataproc** > **Clusters**.
2. Click **CREATE CLUSTER**.
3. Make sure the cluster is going to create in the region **us-central1**.
4. Click **Create**.

5. After the cluster has been created, click the **SSH** button in the row of the master instance.



cluster-d121

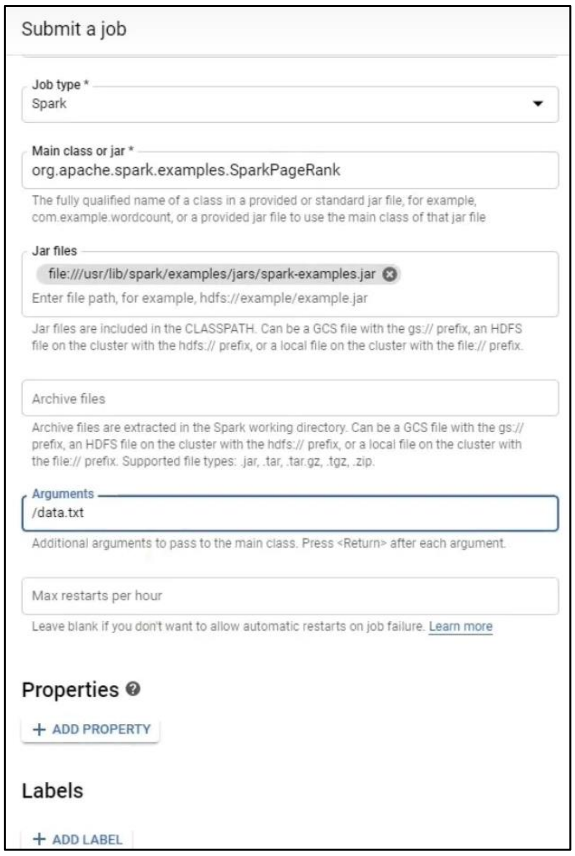
For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. See https://cloud.google.com/compute/docs/disks/performance for information on disk I/O performance.

Name	Role
cluster-d121-m	Master
cluster-d121-w-0	Worker
cluster-d121-w-1	Worker

6. In the SSH console, run the following command:

```
hdfs dfs -cp gs://cloud-training/gsp323/data.txt /data.txt
```

7. Close the SSH window and go back to the Cloud Console.
8. Click **SUBMIT JOB** on the cluster details page.
9. Select **Spark** from the dropdown of “Job type”.
10. Copy `org.apache.spark.examples.SparkPageRank` to “Main class or jar”.
11. Copy `file:///usr/lib/spark/examples/jars/spark-examples.jar` to “Jar files”.
12. Enter `/data.txt` to “Arguments”.



Submit a job

Job type *: Spark

Main class or jar *: org.apache.spark.examples.SparkPageRank

The fully qualified name of a class in a provided or standard jar file, for example, com.example.wordcount, or a provided jar file to use the main class of that jar file

Jar files: file:///usr/lib/spark/examples/jars/spark-examples.jar

Enter file path, for example, hdfs://example/example.jar

Jar files are included in the CLASSPATH. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix.

Archive files

Archive files are extracted in the Spark working directory. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix. Supported file types: .jar, .tar, .tar.gz, .tgz, .zip.

Arguments: /data.txt

Additional arguments to pass to the main class. Press <Return> after each argument.

Max restarts per hour

Leave blank if you don't want to allow automatic restarts on job failure. Learn more

Properties

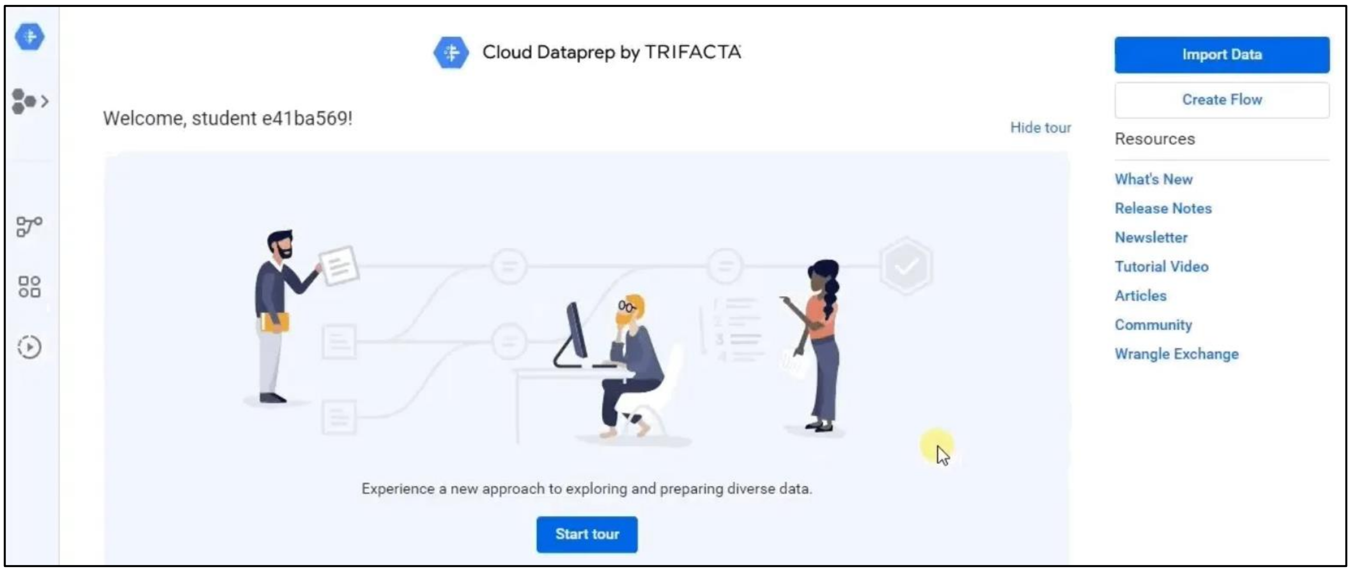
Labels

13. Click **CREATE**.

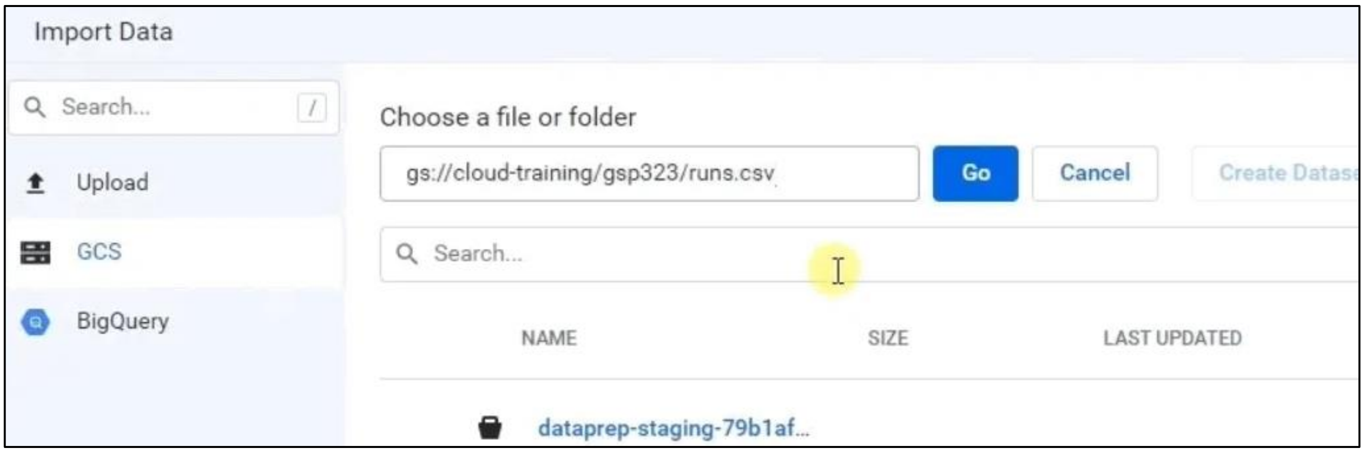
Task 3: Run a simple Dataprep job

Import runs.csv to Dataprep

1. In the Cloud Console, click on **Navigation menu** > **Dataprep**.
2. After entering the home page of Cloud Dataprep, click the **Import Data** button.



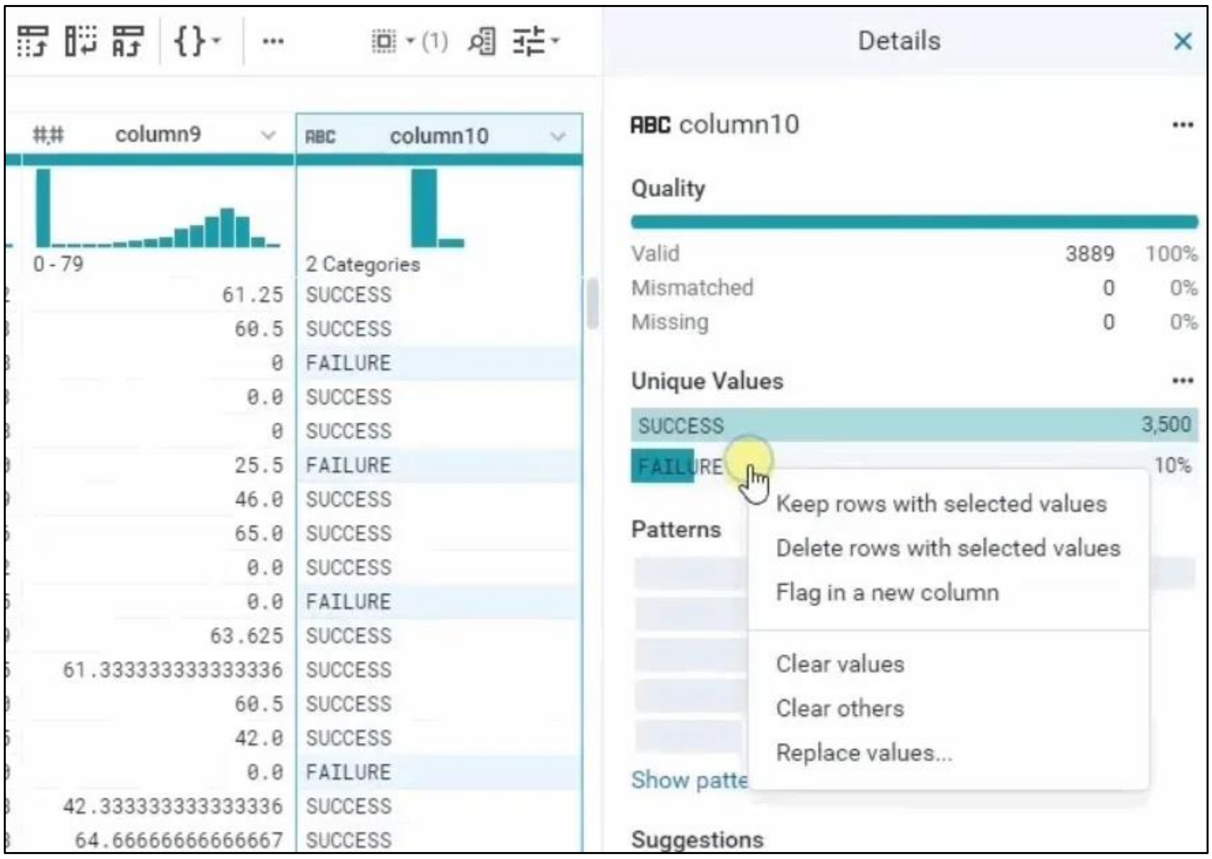
3. In the Import Data page, select **GCS** in the left pane.
4. Click on the pencil icon under Choose a file or folder.
5. Copy `gs://cloud-training/gsp323/runs.csv` to the textbox, and click the **Go** button next to it.



6. After showing the preview of runs.csv in the right pane, click on the **Import & Wrangle** button.

Transform data in Dataprep

1. After launching the Dataperop Transformer, scroll right to the end and select **column10**.
2. In the Details pane, click **FAILURE** under Unique Values to show the context menu.
3. Select **Delete rows with selected values** to Remove all rows with the state of “FAILURE”.



column9	column10
0 - 79	2 Categories
61.25	SUCCESS
60.5	SUCCESS
0	FAILURE
0.0	SUCCESS
0	SUCCESS
25.5	FAILURE
46.0	SUCCESS
65.0	SUCCESS
0.0	SUCCESS
0.0	FAILURE
63.625	SUCCESS
61.333333333333336	SUCCESS
60.5	SUCCESS
42.0	SUCCESS
0.0	FAILURE
42.333333333333336	SUCCESS
64.66666666666667	SUCCESS

Details

ABC column10

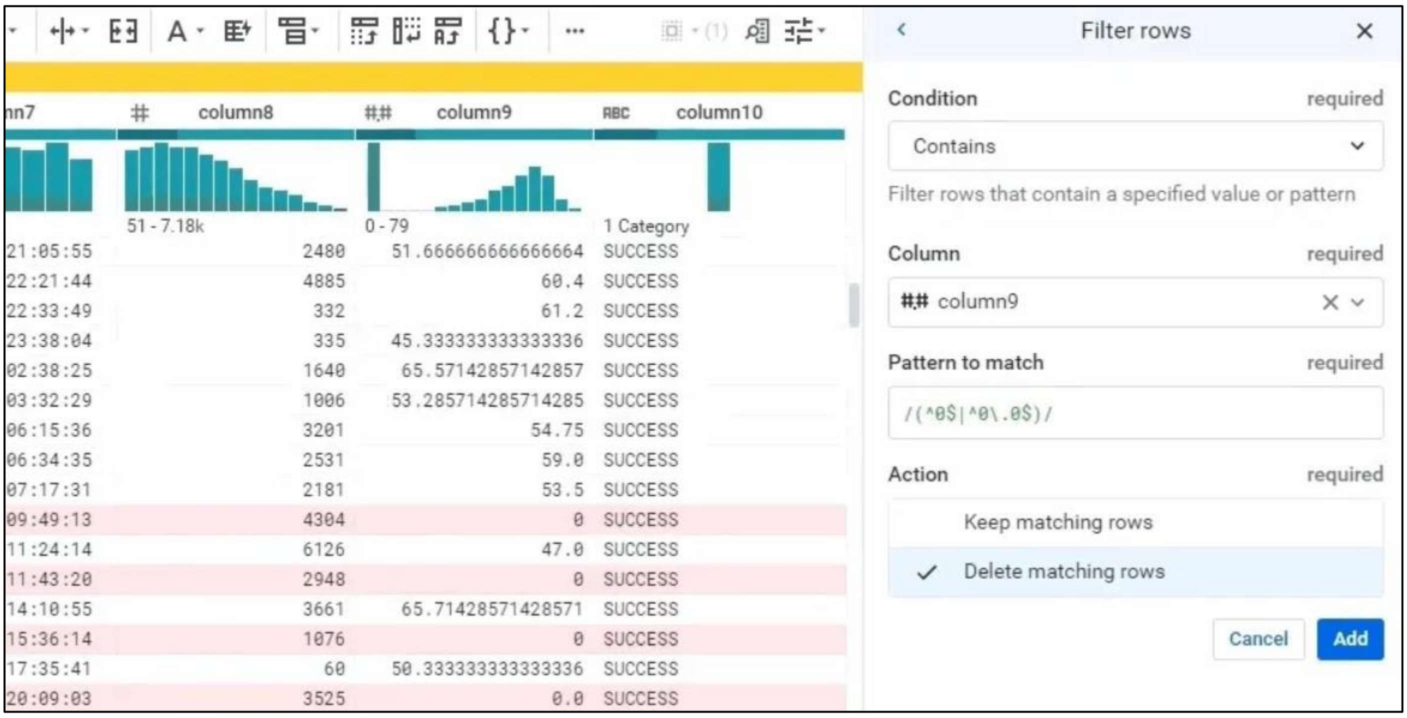
Quality

Valid	3889	100%
Mismatched	0	0%
Missing	0	0%

Unique Values

SUCCESS	3,500
FAILURE	10%

4. Click the downward arrow next to **column9**, choose **Filter rows** > **On column value** > **Contains**.
5. In the Filter rows pane, enter the regex pattern `/(^0$|^0\.0$)/` to “Pattern to match”.
6. Select **Delete matching rows** under the Action section, then click the **Add** button.



nn7	column8	column9	column10
	51 - 7.18k	0 - 79	1 Category
21:05:55	2480	51.666666666666664	SUCCESS
22:21:44	4885	60.4	SUCCESS
22:33:49	332	61.2	SUCCESS
23:38:04	335	45.333333333333336	SUCCESS
02:38:25	1640	65.57142857142857	SUCCESS
03:32:29	1006	53.285714285714285	SUCCESS
06:15:36	3201	54.75	SUCCESS
06:34:35	2531	59.0	SUCCESS
07:17:31	2181	53.5	SUCCESS
09:49:13	4304	0	SUCCESS
11:24:14	6126	47.0	SUCCESS
11:43:20	2948	0	SUCCESS
14:10:55	3661	65.71428571428571	SUCCESS
15:36:14	1076	0	SUCCESS
17:35:41	60	50.333333333333336	SUCCESS
20:09:03	3525	0.0	SUCCESS

Filter rows

Condition required
Contains
Filter rows that contain a specified value or pattern

Column required
column9

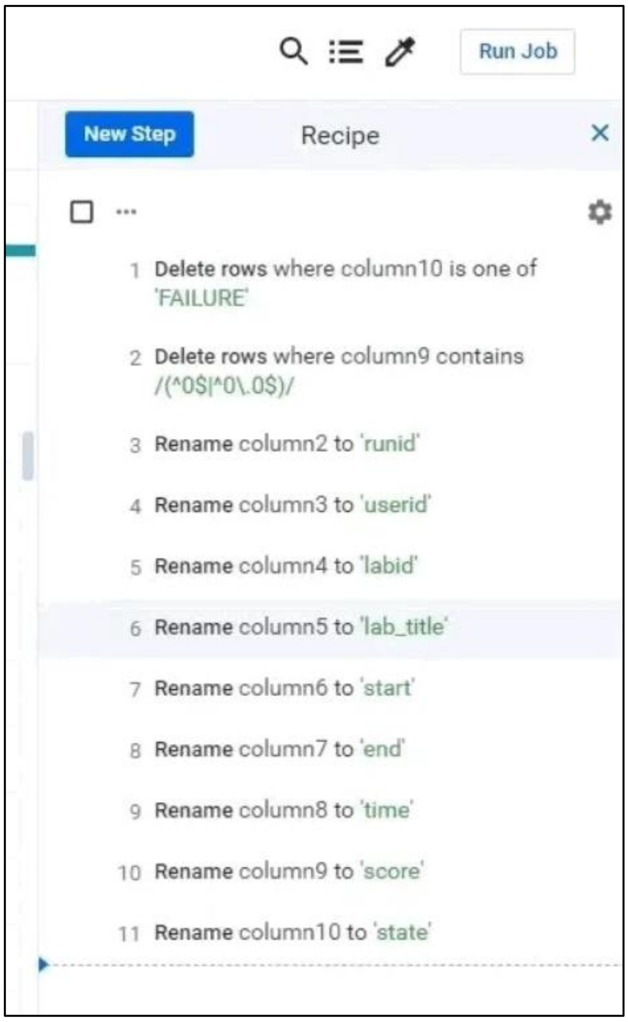
Pattern to match required
/(^0$|^0\.0$)/

Action required
Keep matching rows
✓ Delete matching rows

7. Rename the columns to be:
   - runid
   - userid
   - labid
   - lab_title

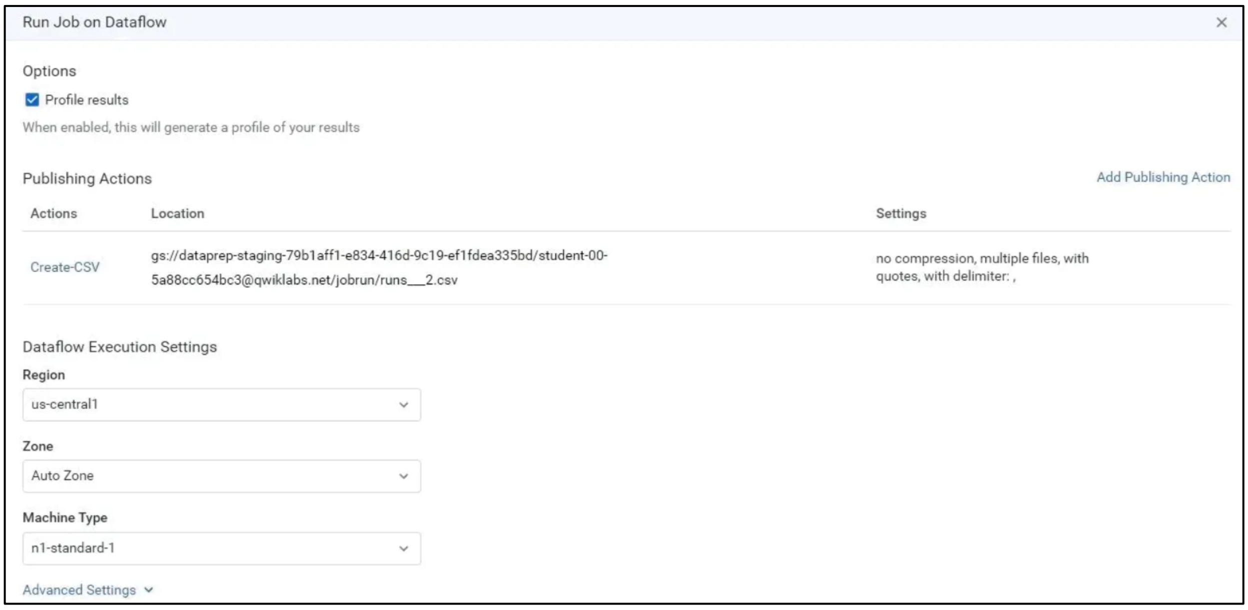
- start
- end
- time
- score
- state

8. Confirm the recipe. It should like the screenshot below.



1 Delete rows where column10 is one of 'FAILURE'
2 Delete rows where column9 contains /(^0$|^0\.0$)/
3 Rename column2 to 'runid'
4 Rename column3 to 'userid'
5 Rename column4 to 'labid'
6 Rename column5 to 'lab_title'
7 Rename column6 to 'start'
8 Rename column7 to 'end'
9 Rename column8 to 'time'
10 Rename column9 to 'score'
11 Rename column10 to 'state'

9. Click **Run Job**.


Run Job on Dataflow

Options

☑ Profile results

When enabled, this will generate a profile of your results

Publishing Actions

Add Publishing Action

Actions	Location	Settings
Create-CSV	gs://dataprep-staging-79b1aff1-e834-416d-9c19-ef1fdea335bd/student-00-5a88cc654bc3@qwiklabs.net/jobrun/runs__2.csv	no compression, multiple files, with quotes, with delimiter: ,

Dataflow Execution Settings

Region: us-central1

Zone: Auto Zone

Machine Type: n1-standard-1

Advanced Settings

Task 4: AI

```
TASK 4 - PART 1 -
CLOUD NATURAL
LANGUAGE:

gcloud iam service-accounts create my-natlang-sa \
  --display-name "my natural language service account"

gcloud iam service-accounts keys create ~/key.json \
  --iam-account my-natlang-sa@${GOOGLE_CLOUD_PROJECT}.iam.gserviceaccount.com

export GOOGLE_APPLICATION_CREDENTIALS="/home/$USER/key.json"

gcloud auth activate-service-account my-natlang-
sa@${GOOGLE_CLOUD_PROJECT}.iam.gserviceaccount.com --key-
file=$GOOGLE_APPLICATION_CREDENTIALS

gcloud ml language analyze-entities --content="Old Norse texts portray Odin as
one-eyed and long-bearded, frequently wielding a spear named Gungnir and wearing a
cloak and a broad hat." > result.json

gcloud auth login
(Copy the token from the link provided)

gsutil cp result.json gs://YOUR_PROJECT-marking/task4-cnl.result
```

Use Google Cloud Speech API to analyze the audio file

1. In the Cloud Console, click on **Navigation menu** > **APIs & Services** > **Credentials**.
2. In the Credentials page, click on **+ CREATE CREDENTIALS** > **API key**.
3. Copy the API key to the clipboard, then click **RESTRICT KEY**.
4. Open the Cloud Shell, store the API key as an environment variable by running the following command:

```
export API_KEY=<YOUR-API-KEY>
```

5. **Replace** `<YOUR-API-KEY>` with the copied key value.

```
nano
request.json
```

```
{
  "config": {
      "encoding":"FLAC",
      "languageCode": "en-US"
  },
  "audio": {
      "uri":"gs://cloud-training/gsp323/task4.flac"
  }
}

curl -s -X POST -H "Content-Type: application/json" --data-binary @request.json \
"https://speech.googleapis.com/v1/speech:recognize?key=${API_KEY}" > result.json

gsutil cp result.json gs://YOUR_PROJECT-marking/task4-gcs.result
```

```
# TASK 4
- PART 3
- VIDEO
INTELLIGE
NCE:
```

```
gcloud iam service-accounts create quickstart

gcloud iam service-accounts keys create key.json --iam-account
quickstart@${GOOGLE_CLOUD_PROJECT}.iam.gserviceaccount.com

gcloud auth activate-service-account --key-file key.json

export ACCESS_TOKEN=$(gcloud auth print-access-token)


nano request.json

{
   "inputUri":"gs://spls/gsp154/video/chicago.mp4",
   "features": [
       "TEXT_DETECTION"
   ]
}



curl -s -H 'Content-Type: application/json' \
    -H "Authorization: Bearer $ACCESS_TOKEN" \
    'https://videointelligence.googleapis.com/v1/videos:annotate' \
    -d @request.json
```

```
curl -s -H 'Content-Type: application/json' -H "Authorization:
Bearer $ACCESS_TOKEN"
'https://videointelligence.googleapis.com/v1/operations/OPERATION_FR
OM_PREVIOUS_REQUEST' > result1.json


gsutil cp result1.json gs://YOUR_PROJECT-marking/task4-gvi.result
```

Congratulations! You completed this challenge lab.