



GCP Data Engineering Bootcamp

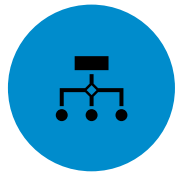
Version 1.01

Nov 2021

Content

- 00** Commandments
- 01** GCP Basics
- 03** Network & Compute
- 04** Storage & Databases
- 05** Data Ingestion
- 06** Data Exploration & Consumption

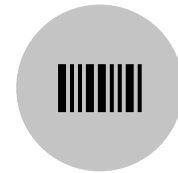
00 Commandments



All code & commands need to be stored in **organized code repository**
<https://docs.github.com/en/get-started/quickstart/create-a-repo>



Create **Unit Tests** for your code



Evaluate code on RRR (**readability**, **reusability**, **repeatability**)



Evaluate code on **Error** and **Exception** handling.



Least the total **cost** for bootcamp more the score for your evaluation



Make sure you have **logging** enabled in your code and logs for each activity could be found through cloud logging and **monitoring**.

Refresh The Concepts

Reading & Reference Material

Read about Cloud Computing basics

- <https://searchcloudcomputing.techtarget.com/definition/public-cloud>

Reference sites for Python

- <https://www.tutorialspoint.com/python/index.htm>
- <https://realpython.com/tutorials/basics/>
- <https://docs.python.org/3/tutorial/venv.html>
- <https://www.python.org/dev/peps/pep-0008/>

SQL

- <https://www.tutorialspoint.com/sql/index.htm>
- <https://www.tutorialspoint.com/sql/sql-select-query.htm>
- <https://www.tutorialspoint.com/sql/sql-where-clause.htm>
- <https://www.tutorialspoint.com/sql/sql-group-by.htm>

Learn about GitHub

- [Git and GitHub for Beginners - Crash Course - YouTube](#)

01 **GCP Basics**

GCP Basics

Setting Up GCP Account

Free Google Cloud Account

- Create your free google cloud account & claim your \$300 credit for 90 days
- <https://cloud.google.com/free/>

Paid Google Account (In case you are not eligible for Free account)

- Check here to setup billing
<https://cloud.google.com/billing/docs/concepts>
- You don't need Organization, Folders
- <https://cloud.google.com/billing/docs/how-to/payment-methods>
- <https://console.cloud.google.com/>

GCP Basics

Create Google Cloud Project

Create a separate project for this bootcamp. At the end of bootcamp assessment this project would be deleted to avoid any accidental charges.

Create a Project, name it “firstname-lastname-bootcamp” replace first and last name with your details.

<https://cloud.google.com/resource-manager/docs/creating-managing-projects#console>

GCP Basics

Understanding IAM

- IAM overview
 - <https://cloud.google.com/iam/docs/overview>
- Read about basic roles
 - <https://cloud.google.com/iam/docs/understanding-roles/#basic>
- Read about Service Accounts
 - <https://cloud.google.com/iam/docs/service-accounts>
- Check IAM in your project
 - <https://console.cloud.google.com/home>
 - <https://console.cloud.google.com/iam-admin/iam>
- Create two service accounts from console
 - Google Cloud Storage Admin and name it as “gcsadminsa”
 - Bigquery Admin and name it as “bqadminsa”
- Use them in your subsequent activities for bootcamp.

GCP Basics

Activate Google Cloud Shell

<https://cloud.google.com/shell/docs>

<https://cloud.google.com/shell>

<https://cloud.google.com/compute/docs/regions-zones>

<https://cloud.google.com/about/locations>

Set up default project, region, zones.

Choose

us-central1 as default region

and us-central1-a as default zone

Set up Environment Variables

PROJECT_ID = your_project_id,

ZONE = us-central1-a

GCP Basics

Cloud SDK – gcloud & python

- Install Python 3 on your Google Cloud Shell VM if its not already present.
- Setup gcloud & python sdk on your Google Cloud Shell VM
- Check & setup gcloud in your Cloud Shell

<https://cloud.google.com/sdk/docs/install>

<https://cloud.google.com/sdk/gcloud>

<https://cloud.google.com/sdk/docs/initializing>

GCP Basics

Cloud Logging and Monitoring

- Make sure all your Python Code has logging.
- For all activities that you performed on Google Cloud, find out your logs in cloud logging.
- <https://cloud.google.com/products/operations>
- <https://cloud.google.com/products/operations#section-6>
- <https://cloud.google.com/blog/products/management-tools/cloud-monitoring-improves-custom-dashboard-creation>

02 **Network and Compute**

02 | Network and Compute

VPC

<https://cloud.google.com/vpc/docs/concepts>

Use Web Console

Create a VPC Network named “vpc-bootcamp” and two subnets “subnet1” and “subnet2”

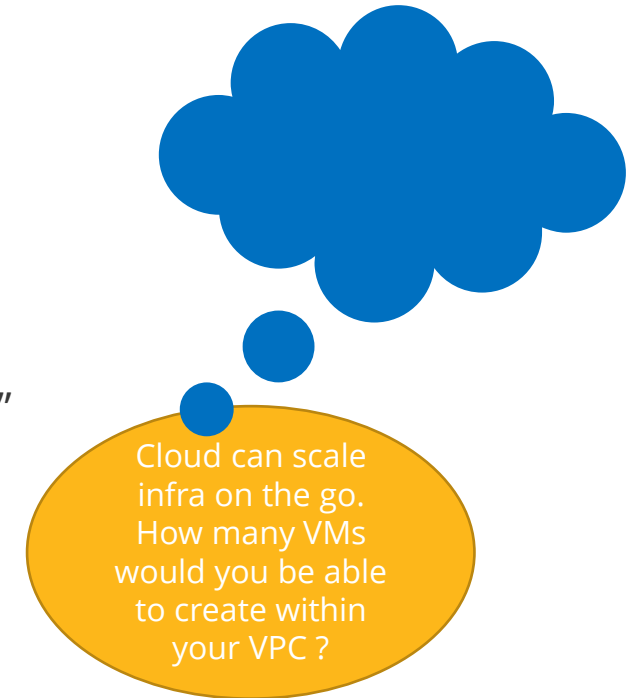
Subnet 1 IP Address Range 10.0.0.0/13

Subnet 2 IP Address Range 10.8.0.0/13

Region: US Central 1

Other settings as default

Create a Firewall rule no https, http allowed



02 | Network and Compute

Set up Bastion Host

<https://cloud.google.com/solutions/connecting-securely#bastion>

https://en.wikipedia.org/wiki/Bastion_host

Create a Bastion Host in “subnet1”

Follow the below post to set it up

<https://medium.com/@0d6e/creating-an-ssh-bastion-host-in-google-cloud-vpc-86d65509eb42>

02 | Network and Compute

Compute Engine

<https://cloud.google.com/vpc/docs/concepts>

Use gcloud and shell script on cloud shell VM –

Create one f1-micro Debian VM named “vm1ce” in subnet1, No external IP

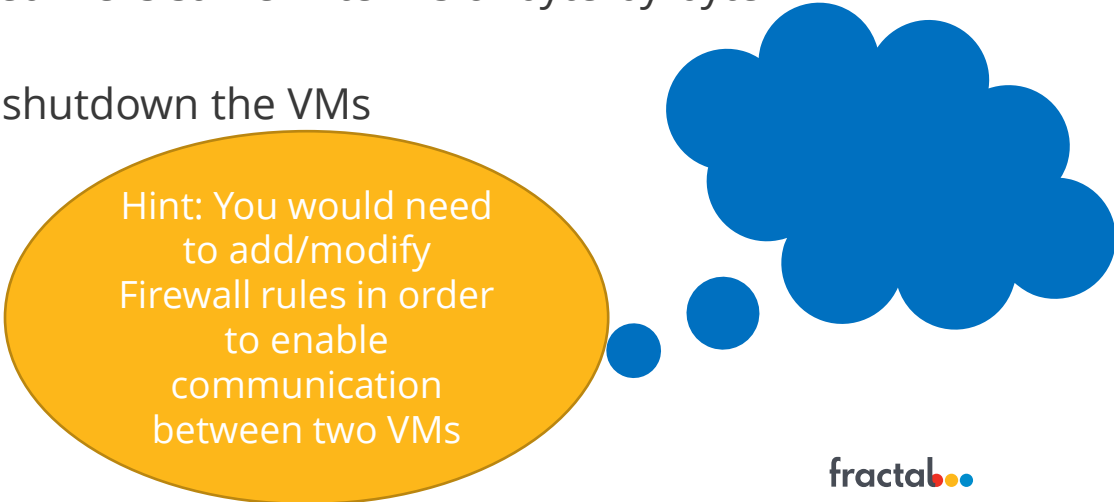
Create one f1-micro Debian VM named “vm2ce” in subnet2, No external IP

SSH to VM “vm1ce” , create a CSV file “vmreccp.csv” with 25 columns and 5000 rows.

Copy “vmreccp.csv” from “vm1ce” to “vm2ce”

SSH to VM “vm2ce” and verify (use any command /tools) the copied file is same in terms of byte-by-byte comparison.

Preserve the screenshots of your tasks and verification and then shutdown the VMs



Hint: You would need to add/modify Firewall rules in order to enable communication between two VMs

03 **Storage & Databases**

03 | Storage & Databases

Files

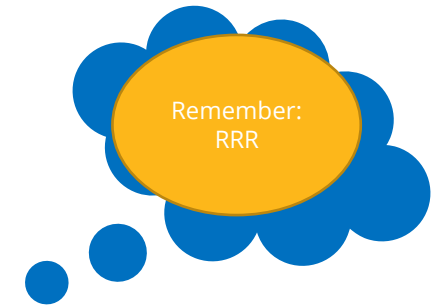
<https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files>

- Files created in this activity would be needed in next set of activities
- Create a python program on your Google Cloud Shell VM to generate a CSV file "sourcefile1.csv" with 10 columns and 10000 rows.
- File should be created in a folder named "workingfile".
- Columns need to be a mix of number, text, , Boolean, date and timestamp types.
- At least one of each type should be present.
- File should be created in a folder named "workingfile".
- Copy the "sourcefile1.csv" file in folder named "truefile"
- Modify your program to add another 5000 rows to your file "sourcefile1.csv" in the folder "workingfile".
- You should now have two files 1st with 10K rows in "truefile" folder and 2nd with 15K rows in "workingfile" folder. Files should be having same name.



03 | Storage & Databases

Google Cloud Storage



<https://cloud.google.com/storage/docs/concepts>

- Create a shell script and use gcloud and or gsutil - in your Google Cloud Shell VM.
- Use the service account created in GCP Basics section.
- Create a standard class bucket "firstname-lastname-fagcpbcmp" in region us-central1
- Enable object versioning on the bucket.
- Set object lifecycle – objects to be deleted in 15 Days
- Upload "sourcefile1.csv" file in folder "truefile" (10K rows created in Files section) to bucket
- Upload the modified file "sourcefile1.csv" in the folder "workingfile" (15K rows created in Files section) to bucket
- Download the 1st version of the "sourcefile1.csv" file from bucket into a folder named "gcsfile" on your Google Cloud Shell VM
- Verify the files in folder "truefile" and "gcsfile" are same. Use any command/tools.
- Need to make sure files are byte by byte same.

03 | Storage & Databases

BigQuery Part 1

This should be default settings throughout the bootcamp whenever you create dataset

<https://cloud.google.com/bigquery/docs>

Use bq from Cloud Shell VM for completing this activity.

Create a dataset named "partition_comp": location: US, default table expiration: 30 Days

Create a table "version1" by uploading version 1 of "sourcefile1.csv" (10K rows) of file in GCS bucket "firstname-lastname-fagcpbcmp". Make sure table is partitioned on one of your datetime columns

Create another table "version2" by uploading version 2 of "sourcefile1.csv" (15K rows) of file in GCS bucket "firstname-lastname-fagcpbcmp". Make sure table is partitioned on same datetime column as "version1" table.

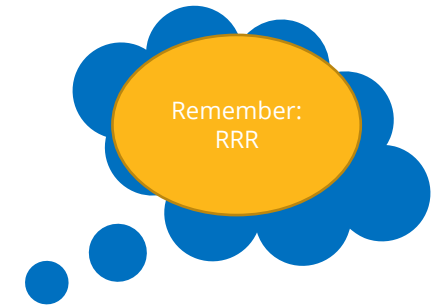
Write an SQL to generate columns and result like below:

Partition	NumberOfRowsinVersion1	NumberOfRowsinVersion2	DifferenceOfRowsVer2-Ver1
-----------	------------------------	------------------------	---------------------------

What if you had to do this check on a daily basis ?

03 | Storage & Databases

BigQuery Part 2



<https://cloud.google.com/bigquery/docs>

Create a python program on your Google Cloud Shell VM, use service account "bqadmins@sa"

Source Data: bigquery-public-data, Dataset->austin_bikeshare, table->bikeshare_trips

Create a dataset named "**bikeshare**"

Create a table "**hourly_summary_trips**" partitioned on trip date and clustered on station name in your dataset which would contain following information: **Hourly summary for trip duration and number of trips by station**

"hourly_summary_trips" Table Columns: trip_date, trip_start_hour, start_station_name, trip_count, total_trip_duration_minutes

Create a view "**busiest_stations_by_hour**" on top of above table to show following information: **Highest trips in hour and its station name**

"busiest_stations_by_hour" view columns: trip_date, trip_start_hour, station_name, max_trips,

03 | Storage & Databases

BigQuery Part 3

<https://cloud.google.com/bigquery/docs>

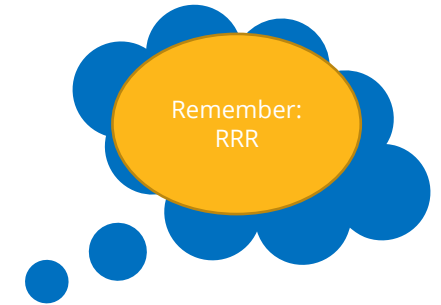
Use Bigquery Web Console

Source Data: bigquery-public-data, Dataset->baseball, table->schedules

Create a dataset named "baseball"

Create a Bigquery view "games_attendance" to generate following result: **For all closed games, find average attendance for weekday and weekend for each home team**

"games_attendance" View Columns : home_team, avg_weekday_attendance, avg_weekend_attendance



03 | Storage & Databases

BigQuery Part 4



<https://cloud.google.com/bigquery/docs>

Download data from: <https://www.kaggle.com/ashishgup/netflix-rotten-tomatoes-metacritic-imdb> to your Google Cloud Shell VM

Use GCS and BigQuery Service accounts created previously.

Load raw data to GCS bucket named "firstname-lastname-netflix" using Python program.

Create a BigQuery dataset named "netflix". AND

Load data from GCS to BigQuery table "netflix-raw-data", identify right data types to represent each column into BigQuery and load data using Python.

Write an SQL & create View: Find out the number of titles in each Country Availability grouped by Runtime

Write an SQL & create View : Find out Number of Titles against each actor. Should cover all actors available in data.

Write an SQL & create View : Find out the number of Titles for each Genre. Should cover all genres available in data.

Write an SQL & create View : Find out the number of Titles available in each country by Genre.

Write an SQL & create View : Find out top 3 Box Office grossers for each year: Release Year, Title, Box Office, Actors, Genre

03 | Storage & Databases

Pub Sub Part 1

<https://cloud.google.com/pubsub/docs>

Create a service account named “pubsubadmin” with pub sub admin role and use it

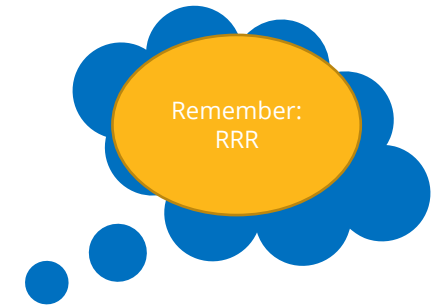
Create a Shell script, use gcloud on your Cloud Shell VM

Create a topic “topic1” and two pull subscriptions “sub1” and “sub2” for this topic

Task:

Publish a message : { "id": 1, "name": "abc" } to the topic “topic1”

Publish 4 more similar messages by incrementing “id” and changing the value of “name”.



03 | Storage & Databases

Pub Sub Part 2

<https://cloud.google.com/pubsub/docs>

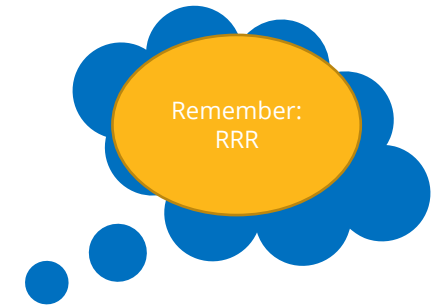
Write a Python program, use Service Account “pubsubadmin”

Use topics and subscriptions created earlier in Part 1

Execute your Python program from Cloud Shell

Task:

Read messages from “topic1” using “sub1” subscription and write them into a file “msg_from_sub1.json”



03 | Storage & Databases

Pub Sub Part 3

<https://cloud.google.com/pubsub/docs>

Create a Python program, use Service Account “pubsubadmin”

Use topic “topic1” created earlier

Execute your Python program from Cloud Shell

Task:

Publish five messages to the “topic1” by incrementing “id” and changing the value of “name”.

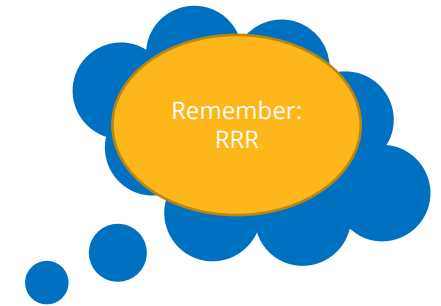
```
{ "id": 6, "name": "xxx"}
```

```
{ "id": 7, "name": "yyy"}
```

```
{ "id": 8, "name": "ccc"}
```

```
{ "id": 9, "name": "zzz"}
```

```
{ "id": 10, "name": "bbb"}
```



03 | Storage & Databases

Pub Sub Part 4

<https://cloud.google.com/pubsub/docs>

Use Python program, use Service Account “pubsubadmin”

Use topic1 and subscription sub2 created earlier

You can execute your Python program from Cloud Shell

Task:

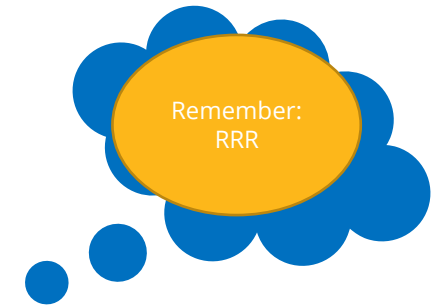
Read messages from “topic1” using “sub2” subscription and write them into a file “msg_from_sub2.json”

What is the difference between “msg_from_sub1.json” (created in part 2) and “msg_from_sub1.json” ? Why?



03 | Storage & Databases

CloudSQL - PostgreSQL



<https://cloud.google.com/sql/docs/postgres>

Use Web Console

Create a PostgreSQL-12 Instance "**mypginstance**", *us-central1 region, single zone, Lightweight, 1 vCPU, 3.75GB, SSD Storage, 20GB Capacity, Backups Disabled*, Note down your password.

Create a database named **myorg**

Create following tables and insert data into these tables. **Hint:** In your Cloud Shell use `gcloud sql connect & psql`

Employee: emp_id, name, dept_id, salary, joining_date, leaving_date, is_active

Deptarment: dept_id, dept_name, dept_head_id

Project: proj_id, proj_name, dept_id, proj_start_date, proj_end_date

Project_staff: proj_id, emp_id, role_name, start_date, end_date

Populate at least 15 employee, 3 department, 3 project, 10 project_staff rows. All data should follow primary key & foreign key constraints

03 | Storage & Databases

Bigtable

Video Learning Only

<https://www.youtube.com/watch?v=P4q4nqJAamo>

<https://www.youtube.com/watch?v=MglAys6RxKE>

Handle massive workloads with Cloud Bigtable database service → <https://goo.gle/3kkbRYU>

QuickStart for Bigtable using Cloud Shell → <https://goo.gle/3klbzkv>

Cloud Bigtable in action (NEXT 2017) → <https://goo.gle/2WmRe6d>

Cloud Bigtable Documentation → <https://goo.gle/3kCTNLa>

Bigtable vs. BigQuery - what's the difference? → <https://goo.gle/3lRHnhQ>

04 Ingestion

04 | Ingestion

Dataflow Part 1

<https://cloud.google.com/dataflow/docs>

<https://cloud.google.com/dataflow/docs/samples>

Use your earlier python code to generate a CSV file “f100k.csv” with 100K rows and at least 50 columns on your Cloud Shell VM.

Create a new regional (us-central1) GCS bucket “firstname-lastname-dfpart” and Upload file “f100k” to GCS bucket using gsutil.

Create a new BigQuery dataset “dfbatch” with location US, default table expiration 30 Days, create a table “t100k” with schema for your file “f100k.csv”

Create a dataflow batch pipeline using Python to load data from file “f100k.csv” in GCS bucket to BigQuery Table “t100k”. Use VPC network created at the start of the bootcamp and machine type as n1-standard-2 and not more than 3 instances while launching pipeline.

04 | Ingestion

Dataflow Part 2

<https://cloud.google.com/dataflow/docs>

<https://cloud.google.com/dataflow/docs/samples>

Use Shell script from Pub Sub Part 1 and create a pubsub topic “**dftopic**” and subscription “**dfsub**”

Use your python program from PubSub – Part 3 to publish Json messages to this topic.

Create a new BigQuery dataset “**dfstream**” with location US, default table expiration 30 Days.

Create a new table “**tstream**” with schema for your messages.

Create a python streaming dataflow pipeline to consume data from subscription “**dfsub**” and write to BigQuery Table “**tstream**”.

Launch your pipeline and check for successful running.

Execute your python program to publish 5 messages every 5 seconds for 5 minutes.

Monitor the pipeline for pub sub messages ack, unack, received etc

After 5 minutes ensure your pipeline processes all messages before you shutdown (drain) your pipeline.

Explore pipeline shutdown options.

04 | Ingestion

Compute Engine with Python

Use Shell script from Compute Engine Activity to Launch a ubuntu/Linux/centos compute engine named **"vm-python-etl"** n1-standard2 in your VPC. Make sure you have Python 3 and python sdk and related packages for postgresql installed

Create a BigQuery dataset named **"pgdataset"** with location US, default table expiration 30 Days.

Write Python programs

- To read all 4 tables from database "myorg" in Cloud SQL Instance "mypginstance" and create tables in Bigquery dataset "pgdataset" with data from those tables. Plan for incremental data load as well i.e. your program should be able to pick up new data when you run in next time.
- To create following additional tables
 - Table Name: "dept_summary"; Columns: "dept_id", "dept_name", "emp_count", "total_cost", "proj_count"; total_cost is sum of salaries of all employees in that dept
 - Table Name: "project_staff_count"; Columns: "proj_id", "staff_count"

04 | Ingestion

Cloud Composer

<https://cloud.google.com/composer/docs/quickstart>

<https://avro.apache.org/docs/1.10.2/gettingstartedpython.html>

Create an Avro file with 20K records and upload it on GCS bucket “firstname-lastname-composer” using python program. Use Service Accounts only.

Create a DAG to upload Avro file data from GCS bucket to a BigQuery table. Dataset name “dag-dataset” and table name “composer_data”

04 | Ingestion

Cloud Function

<https://cloud.google.com/functions/docs>

Create a separate bucket “firstname-lastname-functions” and BigQuery dataset “cfunc” for this activity using your previous code and correct service accounts.

Create a python Cloud Function named “gcs_to_bigquery”

to read data from GCS bucket file and load data into a BigQuery table. Create a separate bucket and BigQuery dataset for this activity.

Cloud function should trigger on a file arrival at GCS bucket “firstname-lastname-functions” .

Upload at least 5 different files (same format but different data), each with rows in the range 20K - 30K rows, simultaneously to your GCS bucket. Your function instance should get triggered for each of the file.

Check the row count in your table with row count of all the files, they should match.

04 | Ingestion

Dataproc

<https://cloud.google.com/dataproc/docs>

Create a Dataproc spark cluster from console

Name **bootcamp,**

Region us-central1,

Zone us-central1-c

Configure nodes

Master node - Machine type 2 vCPUs (n1-standard-2),

2 Worker nodes - Machine type 2 vCPUs (n1-standard-2),

Choose VPC that you created earlier for your spark cluster.

Submit an example spark job. Spark examples are usually at file:///usr/lib/spark/examples/jars/spark-examples.jar

Find out the details of Job what it has done and download/screengrab the Job Details/Logs

04 | Ingestion

Data Fusion

<https://cloud.google.com/data-fusion/docs>

Video Learning Only

<https://www.youtube.com/watch?v=xjsNWh1TLKo>

<https://www.youtube.com/watch?v=kehG0CJw2wo> (start from 17th Minute)

<https://cloud.google.com/data-fusion/docs/quickstart>

05 Exploration & Consumption

05 | Exploration & Consumption

Dataprep

<https://cloud.google.com/dataprep/docs/concepts>

Use the table from dataset created in BigQuery – Part 4

Setup your Dataprep environment - It may take a while to get the setup running.

Create a new flow – **“Netflix Data Exploration”**

Import Data - use table from BigQuery – Part 4

Edit Recipe

1. *Explore to find out number of columns, data types*
2. *What is the most common value in Genre*
3. *Explore top 3 “Country Availability” with number of titles*
4. *Explore “how many distinct values in each column”*
5. *What is the Maximum and Minimum Box Office Amount.*
6. *Can you change data types of columns? How ?*
7. *Can you remove duplicates from data ? How ?*

05 | Exploration & Consumption

DataStudio Part 1

<https://developers.google.com/datastudio>

Refer to the **two slides mydukan – Tables & mydukan – Invoice**

Create 4 tables **customer_master**, **product_master**, **order_details**, **order_quantity** tables in a new BigQuery dataset named “**mydukan**”

Generate data using Python where possible. Generate data for 200 Customers, 50 Products, 5000 Orders, 50000 order_quantity. Save it in CSV or JSON

Load data to Bigquery tables from above files using Python.

Create an Invoice report in Data Studio – refer to slide **mydukan – Invoice** for details. If there are 1000 Orders, then report should have 1000 Order Details. **Hint:** Try preparing your data on BigQuery itself.

Data Studio Report should have **ability to filter on Order Status and Customer Name**

05 | Exploration & Consumption

DataStudio Part 2

<https://developers.google.com/datastudio>

Reference to the next **two slides mydukan – Tables & mydukan – Invoice**

Pro Level :

Generate 10 orders and Order Quantity (related to orders) files upload it to GCS. Make sure these files have customer and products from your master data.

Write a Python Cloud Function to be triggered on file upload to GCS and loads that data in BigQuery tables.

View the same data in your Data Studio Report and get the PDF.

Modify your program to generate files every 2 mins (for 10 mins) and load to GCS and see results in your Data Studio report and get the PDFs for your new orders

Mydukan - Tables

customer_master

customerid	name	address	city	state	pincode
1	Akash Gupta	Shantivan, Borivali	Mumbai	Maharashtra	400066

product_master

productid	productcode	productname	sku	rate	isactive
1	A01	Ashirvad Ata	5KG	150	TRUE
2	A02	Ashirvad Ata	1KG	17	TRUE

order_details

orderid	customerid	orderplaceddatetime	Ordercompletiondatetime	orderstatus
1	1	12/3/2021 19:03		InProgress

order_quantity

orderid	productid	quantity
1	1	5

