

Tony Nguyen

Dr. Shawn Bowers

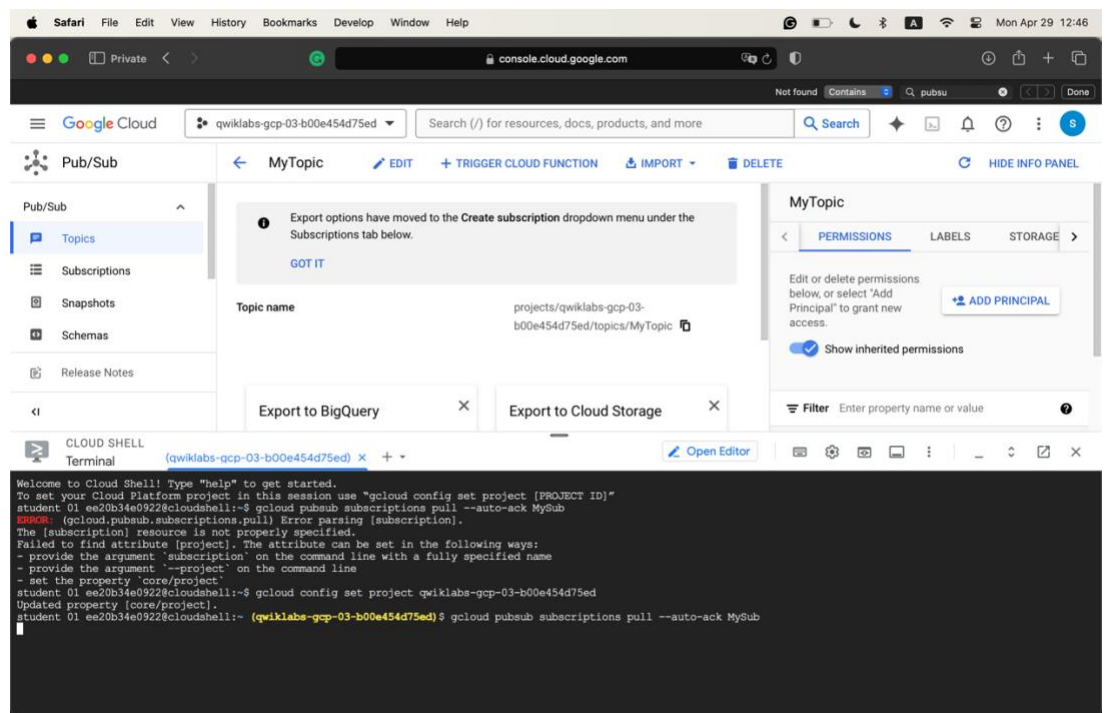
CPSC 324 01

27 April 2024

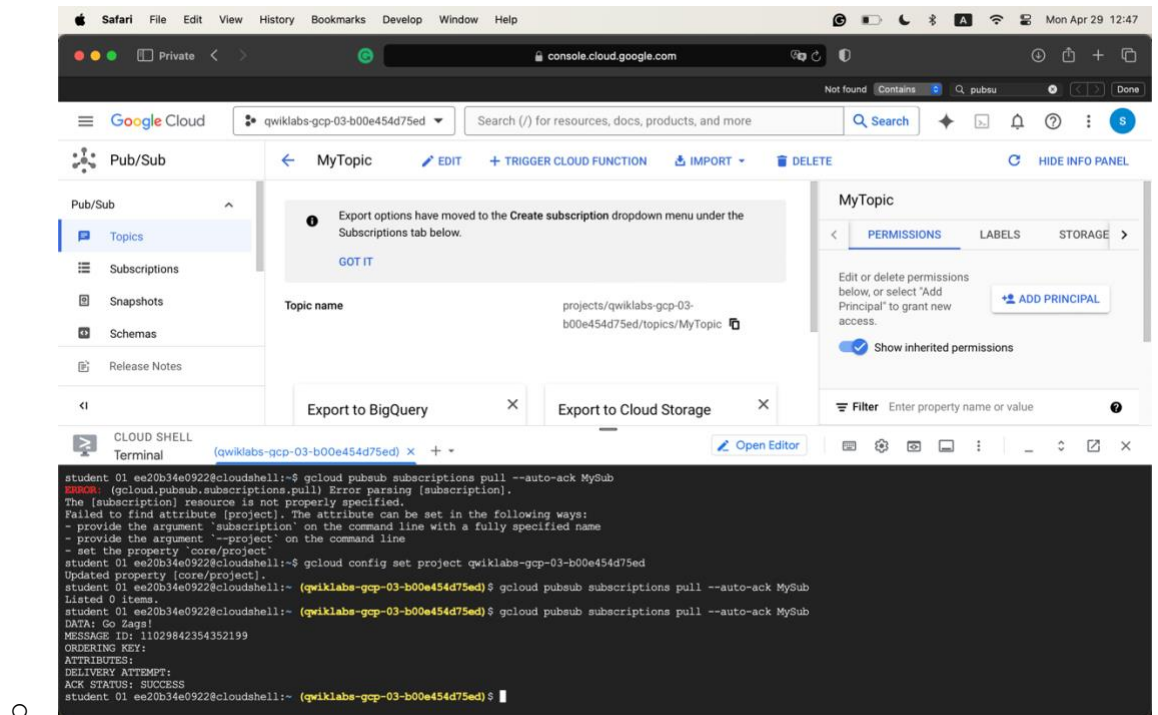
Homework 6

1. Question 1

- Hello world



- Go Zags



2. Question 2

- Write down CLI commands:
 - Create a topic
 - `gcloud pubsub topics create myTopic`
 - To see three topics
 - `gcloud pubsub topics list`
 - Clean up
 - `gcloud pubsub topics delete Test1`
 - Create a subscription
 - `gcloud pubsub subscriptions create --topic myTopic mySubscription`
 - To see the subscriptions
 - `gcloud pubsub topics list-subscriptions myTopic`
 - To delete the two tests

- `gcloud pubsub subscriptions delete Test1`
 - To publish a message
 - `gcloud pubsub topics publish myTopic --message "Hello"`
 - To pull a message
 - `gcloud pubsub subscriptions pull mySubscription --auto-ack`
 - Run several times to see more results
 - `gcloud pubsub topics publish myTopic --message "Publisher is starting to get the hang of Pub/Sub"`
 - limit flag
 - `gcloud pubsub subscriptions pull mySubscription --auto-ack --limit=3`
- Four messages order
 - "Hello"
 - "Publisher's name is Tony"
 - "Publisher likes to eat BunBO"
 - "Publisher thinks Pub/Sub is awesome"
- Results

```

student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub topics publish myTopic --message "Hey"
messageId:
- '11030459366727553'
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub topics publish myTopic --message "It's been lam"
messageId:
- '11030628511296197'
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub topics publish myTopic --message "I'm kinda sleepy"
messageId:
- '11030564652060273'
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub topics publish myTopic --message "So I guess I'm gonna stop here"
messageId:
- '11030401204869743'
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub topics publish myTopic --message "And finish the remaining part later"
messageId:
- '11030435652149256'
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub topics publish myTopic --message "Good night"
messageId:
- '1103058188019481'
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub topics publish myTopic --message "See you tomorrow"
messageId:
- '11030593176887167'
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub subscriptions pull mySubscription --auto-ack --limit=3
DATA: Hey
MESSAGE ID: 11030459366727553
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK STATUS: SUCCESS

DATA: It's been lam
MESSAGE ID: 11030628511296197
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK STATUS: SUCCESS

DATA: I'm kinda sleepy
MESSAGE ID: 11030564652060273
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK STATUS: SUCCESS
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub subscriptions pull mySubscription --auto-ack --limit=2
DATA: So I guess I'm gonna stop here
MESSAGE ID: 11030401204869743

```

```

DELIVERY ATTEMPT:
ACK STATUS: SUCCESS

DATA: It's been lam
MESSAGE ID: 11030628511296197
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK STATUS: SUCCESS

DATA: I'm kinda sleepy
MESSAGE ID: 11030564652060273
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK STATUS: SUCCESS
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub subscriptions pull mySubscription --auto-ack --limit=2
DATA: So I guess I'm gonna stop here
MESSAGE ID: 11030401204869743
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK STATUS: SUCCESS

DATA: And finish the remaining part later
MESSAGE ID: 11030435652149256
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK STATUS: SUCCESS
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub subscriptions pull mySubscription --auto-ack --limit=1
DATA: Good night
MESSAGE ID: 1103058188019481
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK STATUS: SUCCESS
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $ gcloud pubsub subscriptions pull mySubscription --auto-ack --limit=1
DATA: See you tomorrow
MESSAGE ID: 11030593176887167
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK STATUS: SUCCESS
student 01 18b4f2ebd4f2@cloudshell:~ (qwiklabs-gcp-00-cad8fff72214) $

```

3. Question 3

- Notes
- Pub/Sub is an asynchronous global messaging service. Three terms

- Topics
 - Publishing
 - Subscribing
- create_topic() in Step 4

```
def create_topic(project id: str, topic id: str) -> None:
    """Create a new Pub/Sub topic."""
    # [START pubsub quickstart create topic]
    # [START pubsub create topic]
    from google.cloud import pubsub v1

    # TODO(developer)
    # project id = "your-project-id"
    # topic id = "your-topic-id"

    publisher = pubsub v1.PublisherClient()
    topic path = publisher.topic path(project id, topic id)

    topic = publisher.create topic(request={"name": topic path})

    print(f"Created topic: {topic.name}")
    # [END pubsub quickstart create topic]
    # [END pubsub create topic]
```

- - This code creates a Publisher object. Then, it creates a topic from that object.
- It does not seem to pass it into any function.

- list_topic() in Step 4

```
def list_topics(project id: str) -> None:
    """Lists all Pub/Sub topics in the given project."""
    # [START pubsub list topics]
    from google.cloud import pubsub v1

    # TODO(developer)
    # project id = "your-project-id"

    publisher = pubsub v1.PublisherClient()
    project path = f"projects/{project id}"

    for topic in publisher.list topics(request={"project": project path}):
        print(topic)
    # [END pubsub list topics]
```

-
- This code creates a PublisherClient object and then finds the project path that the user passes in. Then, it sends a request to query the data, then print it out.

- To create a topic: python publisher.py \$GOOGLE_CLOUD_PROJECT create MyTopic
- Create a subscription
 - Create a subscription: python subscriber.py \$GOOGLE_CLOUD_PROJECT create MyTopic MySub

```
def create_subscription(project id: str, topic id: str, subscription id: str) -> None:
    """Create a new pull subscription on the given topic."""
    # [START pubsub create pull subscription]
    from google.cloud import pubsub v1

    # TODO(developer)
    # project id = "your-project-id"
    # topic id = "your-topic-id"
    # subscription id = "your-subscription-id"

    publisher = pubsub v1.PublisherClient()
    subscriber = pubsub v1.SubscriberClient()
    topic path = publisher.topic path(project id, topic id)
    subscription path = subscriber.subscription path(project id, subscription id)

    # Wrap the subscriber in a 'with' block to automatically call close() to
    # close the underlying gRPC channel when done.
    with subscriber:
        subscription = subscriber.create_subscription(
            request={"name": subscription path, "topic": topic path}
        )

    print(f"Subscription created: {subscription}")
    # [END pubsub create pull subscription]
```

- For this function, it creates two Publisher and Subscriber objects, which are then used to obtain a topic path and a subscription path. Finally, those two paths are used to create a subscription using the subscriber object.
- Create pull messages

```
def receive_messages(
    project_id: str, subscription_id: str, timeout: Optional[float] = None
) -> None:
    """Receives messages from a pull subscription."""
    # [START pubsub subscriber async pull]
    # [START pubsub quickstart subscriber]
    from concurrent.futures import TimeoutError
    from google.cloud import pubsub v1

    # TODO(developer)
    # project_id = "your-project-id"
    # subscription_id = "your-subscription-id"
    # Number of seconds the subscriber should listen for messages
    # timeout = 5.0

    subscriber = pubsub v1.SubscriberClient()
    # The `subscription_path` method creates a fully qualified identifier
    # in the form `projects/{project_id}/subscriptions/{subscription_id}`
    subscription_path = subscriber.subscription_path(project_id, subscription_id)

    def callback(message: pubsub v1.subscriber.message.Message) -> None:
        print(f"Received {message}.")
        message.ack()

    streaming_pull_future = subscriber.subscribe(subscription_path, callback=callback)
    print(f"Listening for messages on {subscription_path}...\n")

    # Wrap subscriber in a 'with' block to automatically call close() when done.
    with subscriber:
        try:
            # When `timeout` is not set, result() will block indefinitely,
            # unless an exception is encountered first.
            streaming_pull_future.result(timeout=timeout)
        except TimeoutError:
            streaming_pull_future.cancel() # Trigger the shutdown.
            streaming_pull_future.result() # Block until the shutdown is complete.

    # [END pubsub subscriber async pull]
    # [END pubsub quickstart subscriber]
```

-
- This function creates a subscriber object, which is then used to get the subscription_path. This path is later passed to obtain a streaming pull future (I'm not sure what this is used for).

4. Question 4

- send_messages.py

```
student_01_1a50d9dfad53@cloudshell:~$ python send_messages.py
Published a message to partition 0 and offset 0.
student_01_1a50d9dfad53@cloudshell:~$ gcloud config set project qwiklabs-gcp-01-1af8635687ae
Updated property [core/project].
student_01_1a50d9dfad53@cloudshell:~ (qwiklabs-gcp-01-1af8635687ae)$ python send_messages.py
Published a message to partition 0 and offset 1.
student_01_1a50d9dfad53@cloudshell:~ (qwiklabs-gcp-01-1af8635687ae)$ python send_messages.py
Published a message to partition 0 and offset 2.
student_01_1a50d9dfad53@cloudshell:~ (qwiklabs-gcp-01-1af8635687ae)$ python send_messages.py
Published a message to partition 0 and offset 3.
student_01_1a50d9dfad53@cloudshell:~ (qwiklabs-gcp-01-1af8635687ae)$ python send_messages.py
Published a message to partition 0 and offset 4.
student_01_1a50d9dfad53@cloudshell:~ (qwiklabs-gcp-01-1af8635687ae)$ python send_messages.py
Published a message to partition 0 and offset 5.
student_01_1a50d9dfad53@cloudshell:~ (qwiklabs-gcp-01-1af8635687ae)$ python send_messages.py
Published a message to partition 0 and offset 6.
```

○

- receive_message.py

```
student_01_1a50d9dfad53@cloudshell:~ (qwiklabs-gcp-01-1af8635687ae)$ python receive_messages.py
Listening for messages on projects/370412727899/locations/us-east1-b/subscriptions/my-lite-subscription1...
Received Hello world! of ordering key .
Received Hello world! of ordering key .
Received Hello world! of ordering key .
Received Hello world! of ordering key .
Received Hello world! of ordering key .
Received Hello world! of ordering key .
Received Hello world! of ordering key .
```

- The difference between Pub/Sub and Pub/Sub Lite
 - Pub/Sub: an asynchronous and scalable messaging service that decouples services producing messages from services processing those messages.
 - Pub/Sub Lite: A separate but similar messaging service built for lower cost. It offers lower reliability compared to Pub/Sub.

5. Question 5

```
student 01 bb77cb7dia8d@cloudshell:~ (qwiklabs-gcp-01-9a5bela6b66d)$ gcloud pubsub topics create cron-topic
Created topic [projects/qwiklabs-gcp-01-9a5bela6b66d/topics/cron-topic].
student 01 bb77cb7dia8d@cloudshell:~ (qwiklabs-gcp-01-9a5bela6b66d)$ gcloud pubsub subscriptions create cron-sub --topic cron-topic
Created subscription [projects/qwiklabs-gcp-01-9a5bela6b66d/subscriptions/cron-sub].
```

```
student 01 bb77cb7dia8d@cloudshell:~ (qwiklabs-gcp-01-9a5bela6b66d)$ gcloud pubsub subscriptions pull cron-sub --limit 5
DATA: hello tony!
MESSAGE ID: 11069168017453272
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK ID: UAYWLF1GSFE3GQhoUQ5PX1M NSAoRRECCBQPFfH1wQVF1X1h8aFENGXJ9YH08WRaCV0VaffZRGwdoTm11H-235PNLQ1RrWxMHC0RVd19aHAltX15y8Xm00puyt6G3egk9Ovmpg95t04G81cJGZIM9XxJLLD5-NSxPQV
5AEkw-GURJuytDCypYEU4E1SE-MD5FU0Q

DATA: hello tony!
MESSAGE ID: 11069632808529400
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK ID: UAYWLF1GSFE3GQhoUQ5PX1M NSAoRRECCBQPFfH1wQVF1X15zaFENGXJ9YH08WRaCVBEGeltRGwdoTm11H-235PNLQ1RrWxMHC0NQffdbEwhqVWh1B3m00puyt6G3egk9Ovmpg95t04G81cJGZIM9XxJLLD5-NSxPQV
5AEkw-GURJuytDCypYEU4E1SE-MD5FU0Q

DATA: hello tony!
MESSAGE ID: 11069431488404707
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK ID: UAYWLF1GSFE3GQhoUQ5PX1M NSAoRRECCBQPFfH1wQVF1X15zaFENGXJ9YH08WRaCVBEGeltRGwdoTm11H-235PNLQ1RrWxMHC0FQf1tTBw1oWft1AHm00puyt6G3egk9Ovmpg95t04G81cJGZIM9XxJLLD5-NSxPQV
5AEkw-GURJuytDCypYEU4E1SE-MD5FU0Q

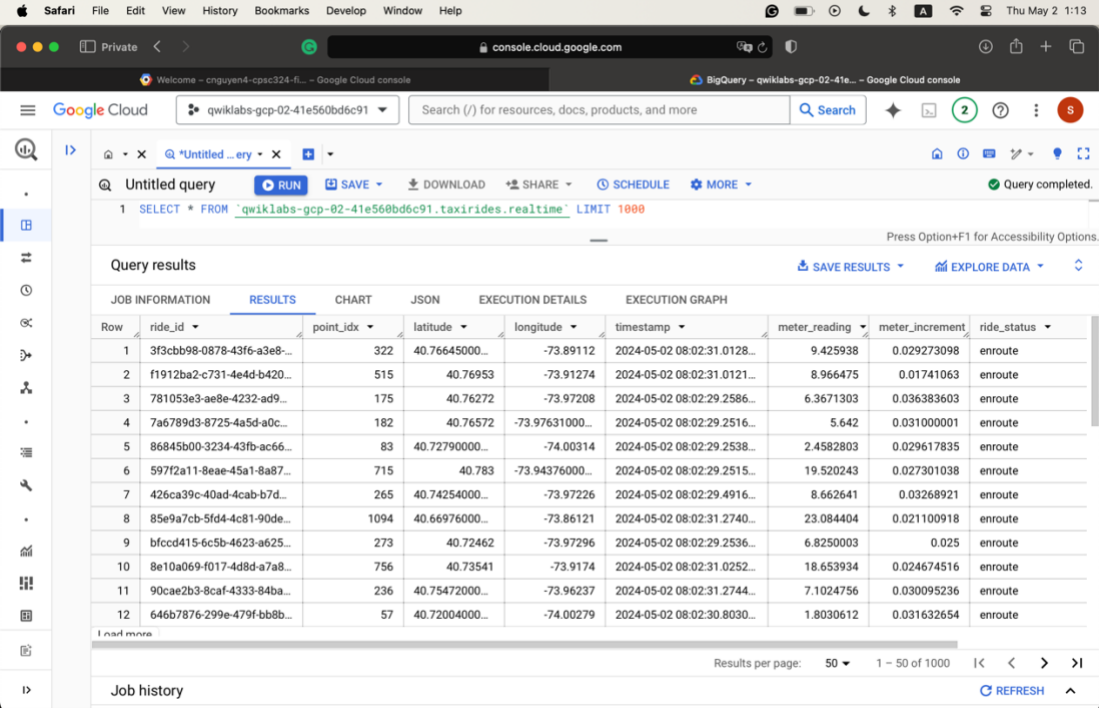
DATA: hello tony!
MESSAGE ID: 11069601731360556
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK ID: UAYWLF1GSFE3GQhoUQ5PX1M NSAoRRECCBQPFfH1wQVF1X15zaFENGXJ9YH08WRaCVBEGeltRGwdoTm11H-235PNLQ1RrWxMHC0NTf1hYGg5uXFlwAXm00puyt6G3egk9Ovmpg95t04G81cJGZIM9XxJLLD5-NSxPQV
5AEkw-GURJuytDCypYEU4E1SE-MD5FU0Q

DATA: hello tony!
MESSAGE ID: 11069593501253525
ORDERING KEY:
ATTRIBUTES:
DELIVERY ATTEMPT:
ACK ID: RFAGf1xdRKhRnXk1aFEOT14jPzUgKEURAgg0BXx9cUFRdV5dGdRDRlyfGkjaVIQVFRGAH5VWxNem1cbvvr8JhEX0B0alovCAdMUHtfWWhk1allecC SxvC9q7WoSkAvOb7 rv9pe5fZocJvZiA9XxJLLD5-NSxPQV
5AEkw-GURJuytDCypYEU4E1SE-MD5FU0Q
```

6. Question 6

- gcloud commands
 - Create a dataset
 - bq mk taxirides

- Create a table
 - `bq 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`
- Create a cloud storage
 - `export BUCKET_NAME=[PROJECT_ID]`
 - `gsutil mb gs://$BUCKET_NAME/`
- Deploy the dataflow template
 - `gcloud dataflow jobs run iotflow \`
`--gcs-location gs://dataflow-templates-europe-`
`west4/latest/PubSub_to_BigQuery \`
`--region europe-west4 \`
`--worker-machine-type e2-medium \`
`--staging-location gs://qwiklabs-gcp-02-41e560bd6c91/temp \`
`--parameters inputTopic=projects/pubsub-public-`
`data/topics/taxirides-realtime,outputTableSpec=qwiklabs-gcp-02-`
`41e560bd6c91:taxirides.realtime`
- Screenshot of query result



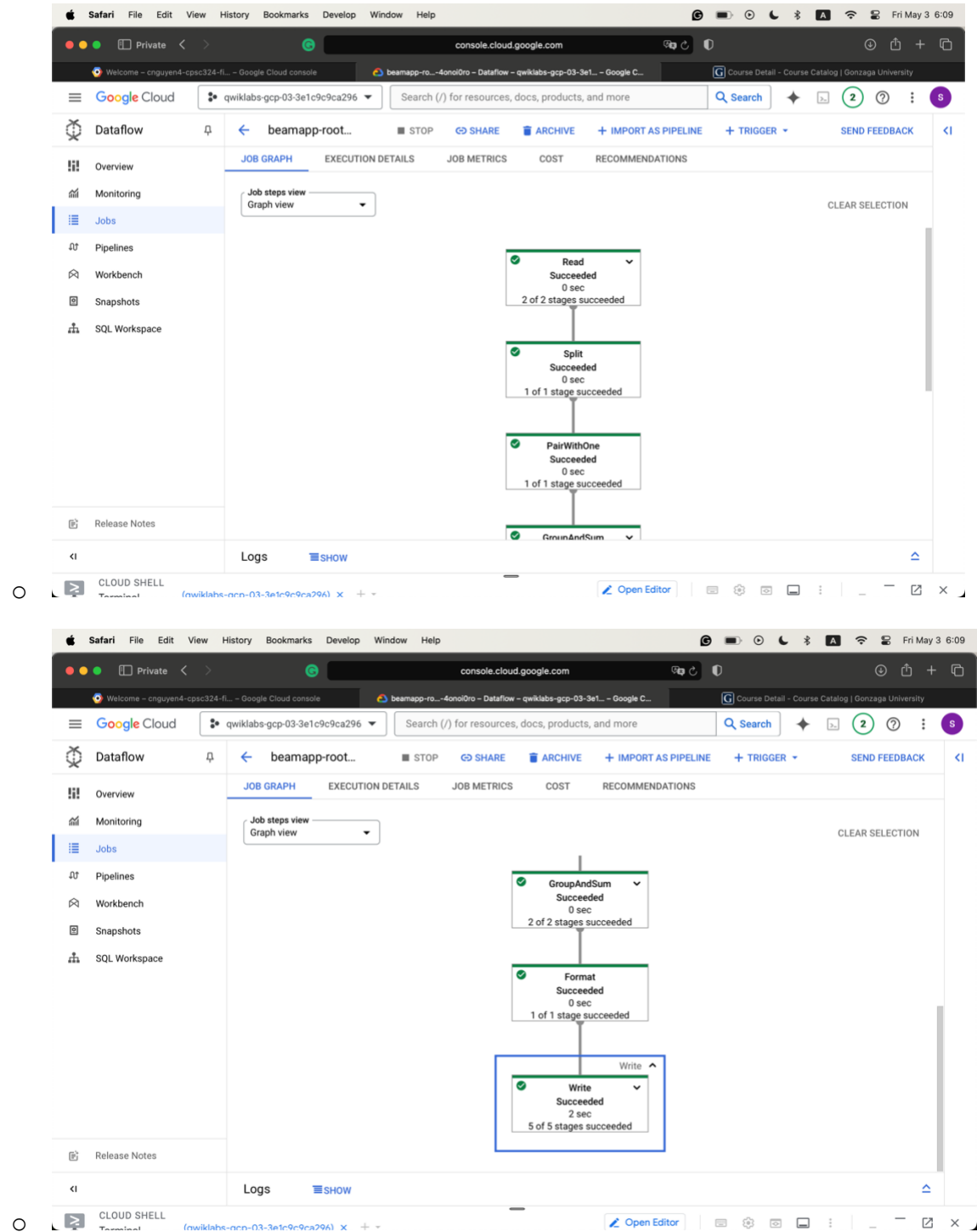
Query results

Row	ride_id	point_idx	latitude	longitude	timestamp	meter_reading	meter_increment	ride_status
1	3f3cb98-0878-43f6-a3e8...	322	40.76645000...	-73.89112	2024-05-02 08:02:31.0128...	9.425938	0.029273098	enroute
2	f1912ba2-c731-4e4d-b420...	515	40.76953	-73.91274	2024-05-02 08:02:31.0121...	8.966475	0.01741063	enroute
3	781053e3-ae8e-4232-ad9...	175	40.76272	-73.97208	2024-05-02 08:02:29.2586...	6.3671303	0.036383603	enroute
4	7a6789d3-8725-4a5d-a0c...	182	40.76572	-73.97631000...	2024-05-02 08:02:29.2516...	5.642	0.031000001	enroute
5	86845b00-3234-43fb-ac66...	83	40.72790000...	-74.00314	2024-05-02 08:02:29.2538...	2.4582803	0.029617835	enroute
6	597f2a11-8eae-45a1-8a87...	715	40.783	-73.94376000...	2024-05-02 08:02:29.2515...	19.520243	0.027301038	enroute
7	426ca39c-40ad-4cab-b7d...	265	40.74254000...	-73.97226	2024-05-02 08:02:29.4916...	8.662641	0.03268921	enroute
8	85e9a7cb-5fd4-4c81-90de...	1094	40.66976000...	-73.86121	2024-05-02 08:02:31.2740...	23.084404	0.021100918	enroute
9	bfcc415-6c5b-4623-a625...	273	40.72462	-73.97296	2024-05-02 08:02:29.2536...	6.8250003	0.025	enroute
10	8e10a069-f017-4d8d-a7a8...	756	40.73541	-73.9174	2024-05-02 08:02:31.0252...	18.653934	0.024674516	enroute
11	90cae2b3-8caf-4333-84ba...	236	40.75472000...	-73.96237	2024-05-02 08:02:31.2744...	7.1024756	0.030095236	enroute
12	646b7876-299e-479f-bb8b...	57	40.72004000...	-74.00279	2024-05-02 08:02:30.8030...	1.8030612	0.031632654	enroute

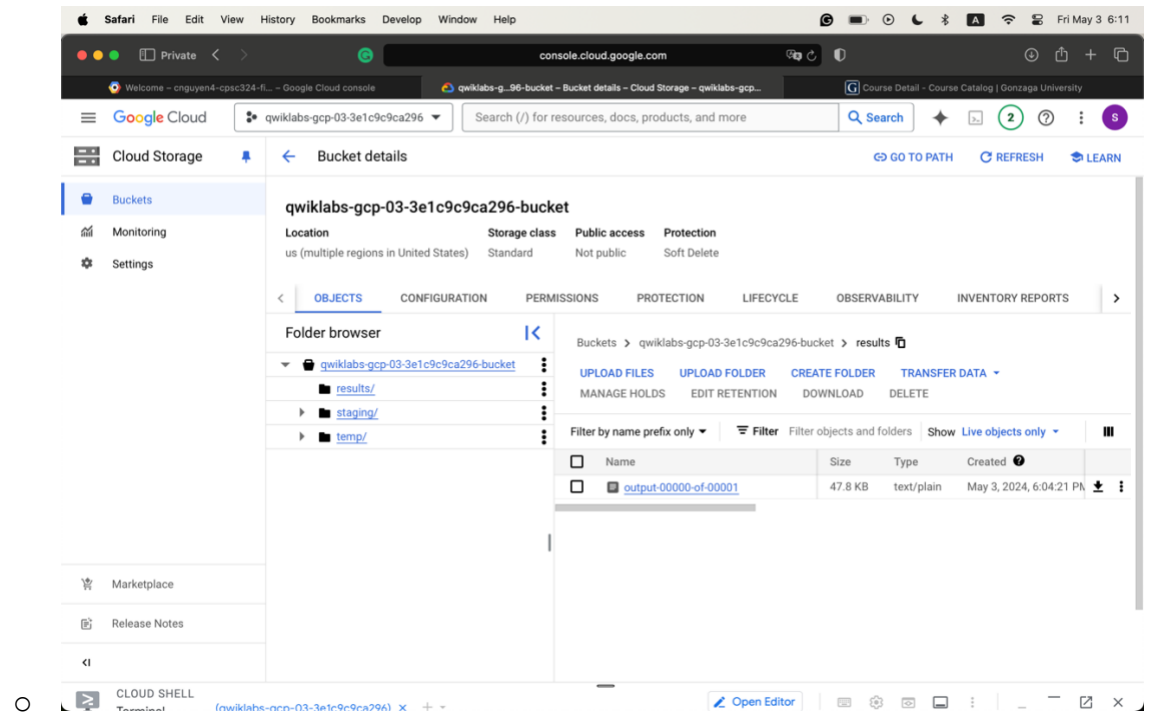
7. Question 7

- JOB_MESSAGE_DETAILED similar items
 - Fusing: Refers to combining multiple operations into a single operation to optimize pipeline execution.
 - Autoscaling: Involves adjusting the number of worker resources automatically based on the workload.
 - Expanding: Pertains to transforming user-defined pipelines into detailed execution graphs for processing.

•



- Directory screenshot



8. Question 8

- Files copied to the bucket
 - usa_names.csv
 - head_usa_names.csv
 - They both have the same schemas
 - state
 - gender
 - year
 - name
 - number
 - created_date
- Files 8
 - How is the pipeline p initialized?

- `p = beam.Pipeline(options=PipelineOptions(pipeline_args))`

- What is the first step of the pipeline? What does it do?

- Read a line as the source of the pipeline. Skip the first line

- `| 'Read from a File' >> beam.io.ReadFromText(known_args.input,`
`skip_header_lines=1)`

- What is the second step?

- Translates from a CSV file single row input as a string to a dictionary object consumable by BigQuery

- `| 'String To BigQuery Row' >>`
`beam.Map(lambda s: data_ingestion.parse_method(s))`

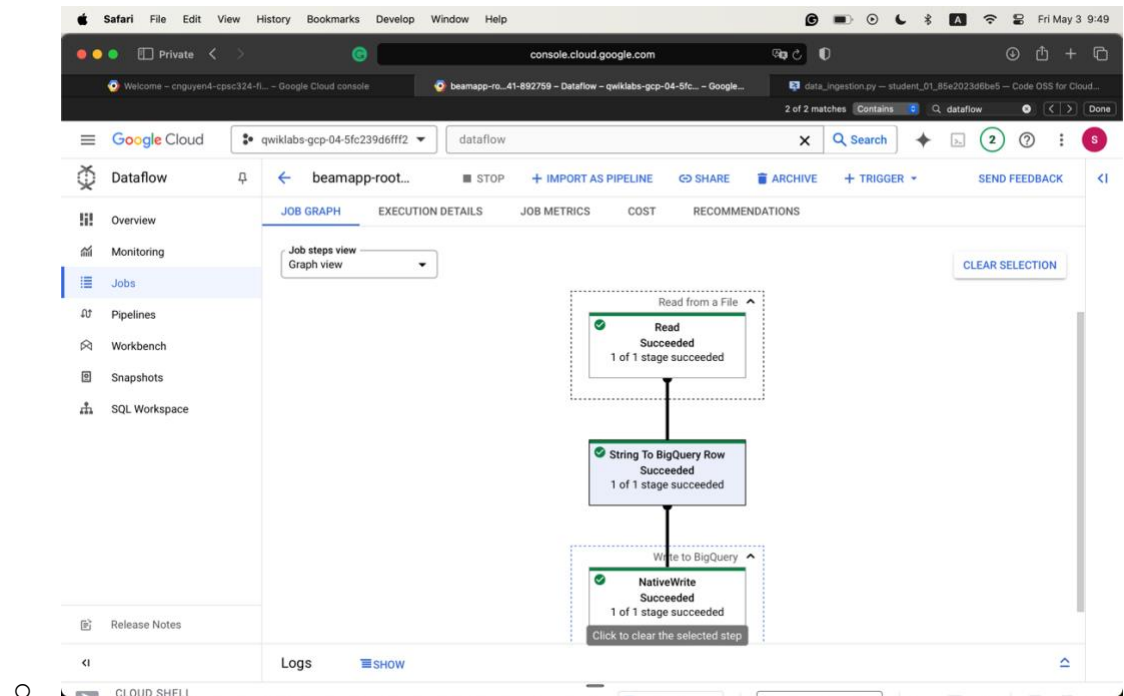
- What is the third step?

- Write the parsed result to Big Query

- `| 'Write to BigQuery' >> beam.io.Write(`
`beam.io.BigQuerySink(`

`schema='state:STRING,gender:STRING,year:STR`
`ING,name:STRING,`
`'number:STRING,created_date:STRING',`
`create_disposition=beam.io.BigQueryDisposition.C`
`REATE_IF_NEEDED,`
`write_disposition=beam.io.BigQueryDisposition.W`
`RITE_TRUNCATE))))`

- Job graph in step 9



- Data pipeline comparison in Steps 9 and 11
 - In step 9, the pipeline converts the input into dictionary objects after taking input. However, in step 11, after converting into dictionary objects, the pipeline transforms the data that contains the year to a format understandable to Big Query.
- Steps for the pipeline in task 13
 - Initiate the command line
 - `p = beam.Pipeline(options=PipelineOptions(pipeline_args))`
 - `schema = parse_table_schema_from_json(data_ingestion.schema_str)`
 - Read the file
 - `| 'Read From Text' >> beam.io.ReadFromText(known_args.input, skip_header_lines=1)`
 - Translates the CSV input row into a dictionary object
 - `| 'String to BigQuery Row' >> beam.Map(lambda s:`

`data_ingestion.parse_method(s))`

- `| 'Write to BigQuery' >> beam.io.Write(`
`beam.io.BigQuerySink(`
`known_args.output,`
`schema=schema,`
`create_disposition=beam.io.BigQueryDisposition.CREATE_IF_NEEDED,`
`write_disposition=beam.io.BigQueryDisposition.WRITE_TRUNCATE)))`
- Steps for the pipeline in task 13
 - Initiate the pipeline using the pipeline arguments
 - `p = beam.Pipeline(options=PipelineOptions(pipeline_args))`
 - `schema = parse_table_schema_from_json(data_ingestion.schema_str)`
 - Fetch in the second source of data (which is Big Query)
 - `p | 'Read from BigQuery' >> beam.io.Read(`
`beam.io.BigQuerySource(query=read_query,`
`use_standard_sql=True))`
`| 'Abbreviation to Full Name' >> beam.Map(`
`lambda row: (row['state_abbreviation'], row['state_name'])))`
 - Read the file
 - `| 'Read From Text' >> beam.io.ReadFromText(known_args.input,`
`skip_header_lines=1)`
 - Translates from the raw string data in the CSV to a dictionary.

- | 'String to BigQuery Row' >> beam.Map(lambda s:

data_ingestion.parse_method(s))
 - Join the data
 - | 'Join Data' >> beam.Map(add_full_state_name, AsDict(

state_abbreviations))
 - Writing to BigQuery
 - | 'Write to BigQuery' >> beam.io.Write(

beam.io.BigQuerySink(

known_args.output,

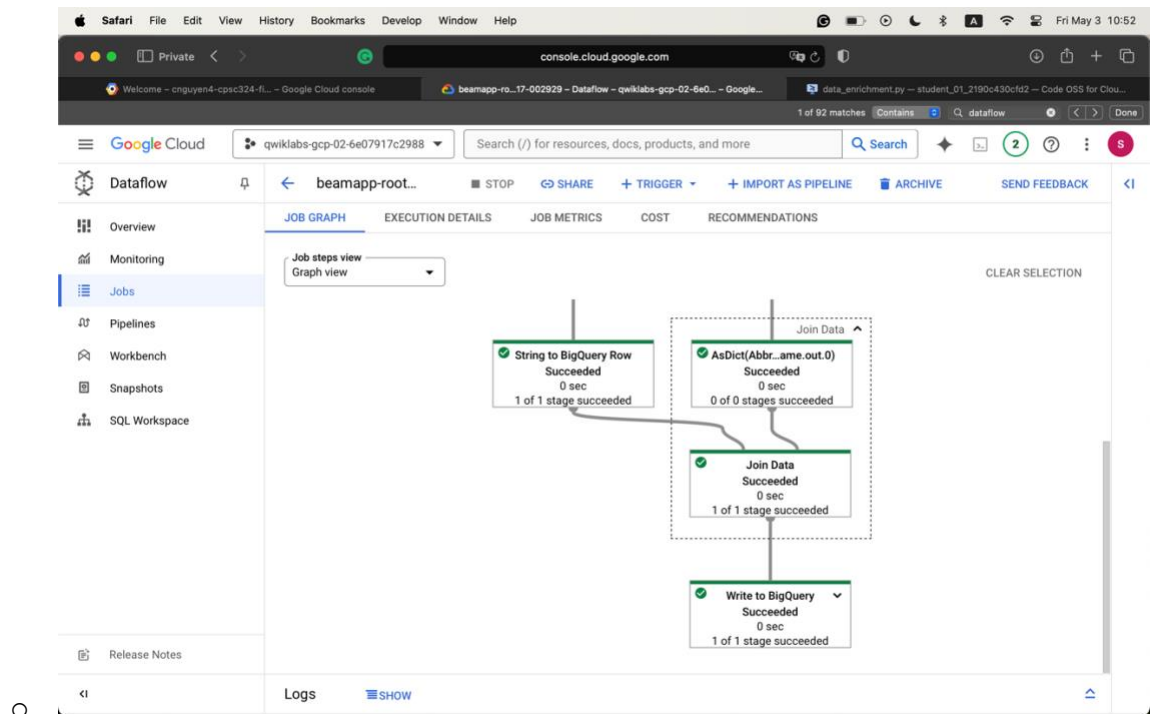
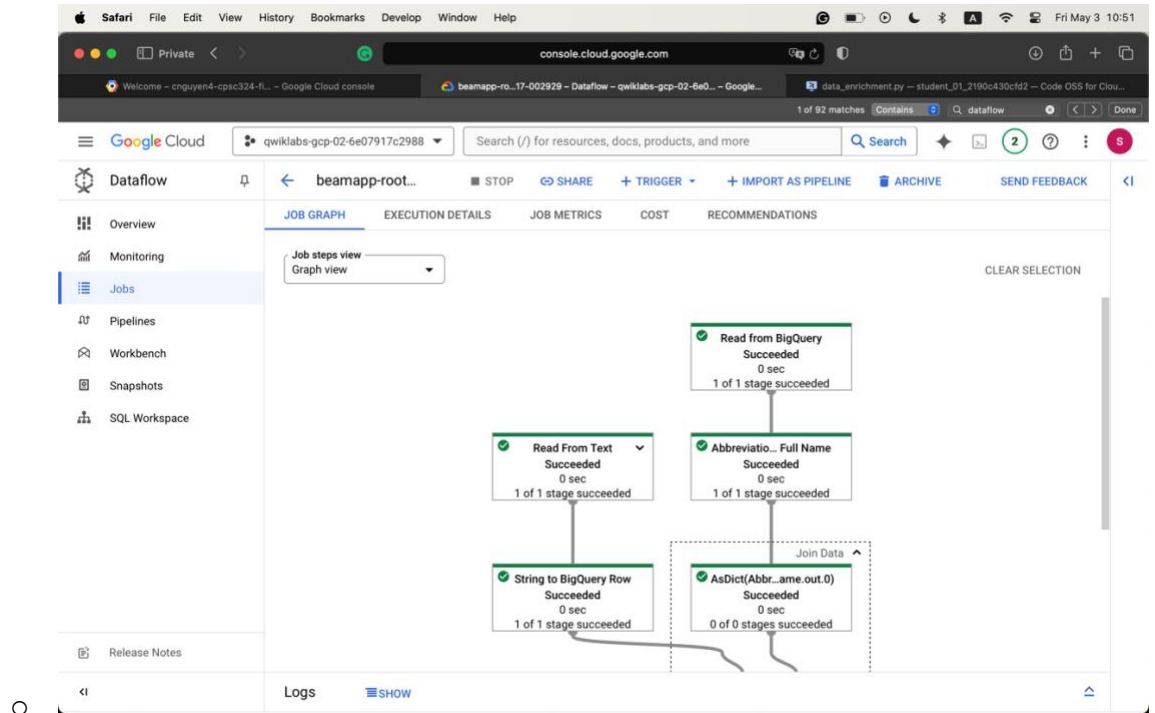
schema=schema,

create_disposition=beam.io.BigQueryDisposition.CREATE_IF_NEE

DED,

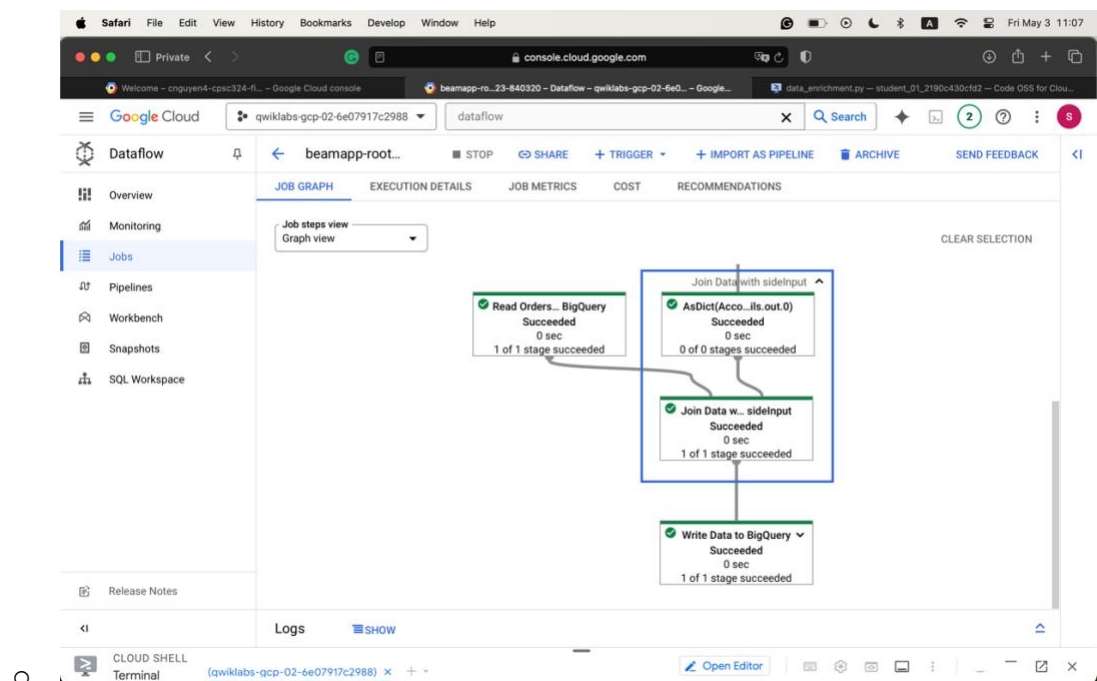
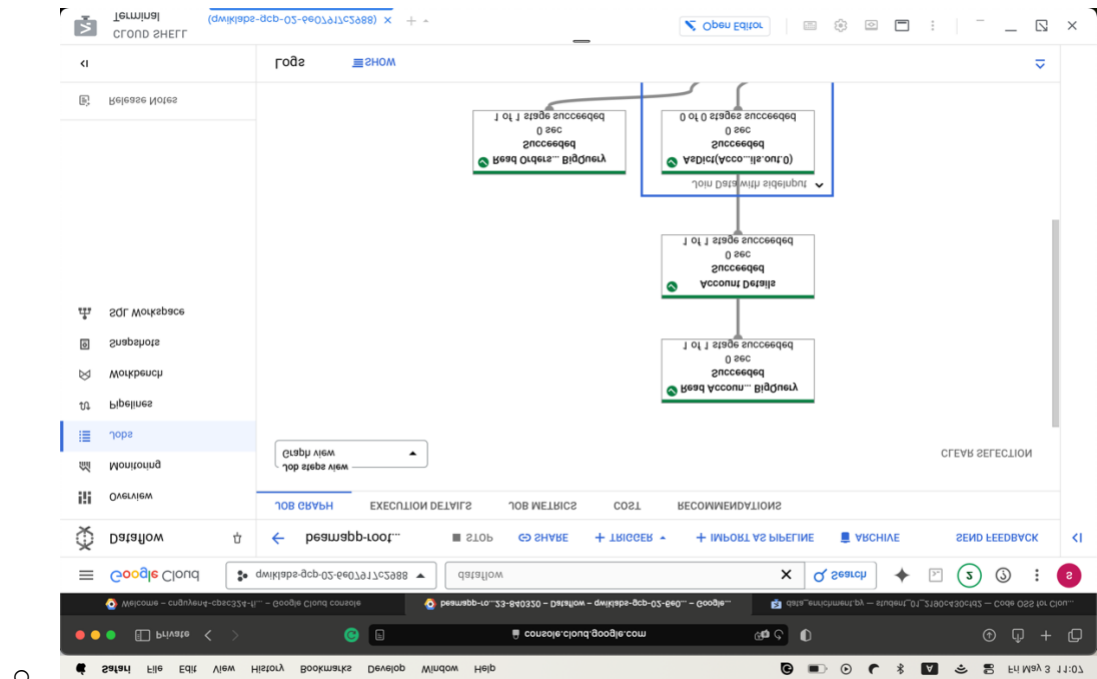
write_disposition=beam.io.BigQueryDisposition.WRITE_TRUNCAT

E)))
- Graph for task 14



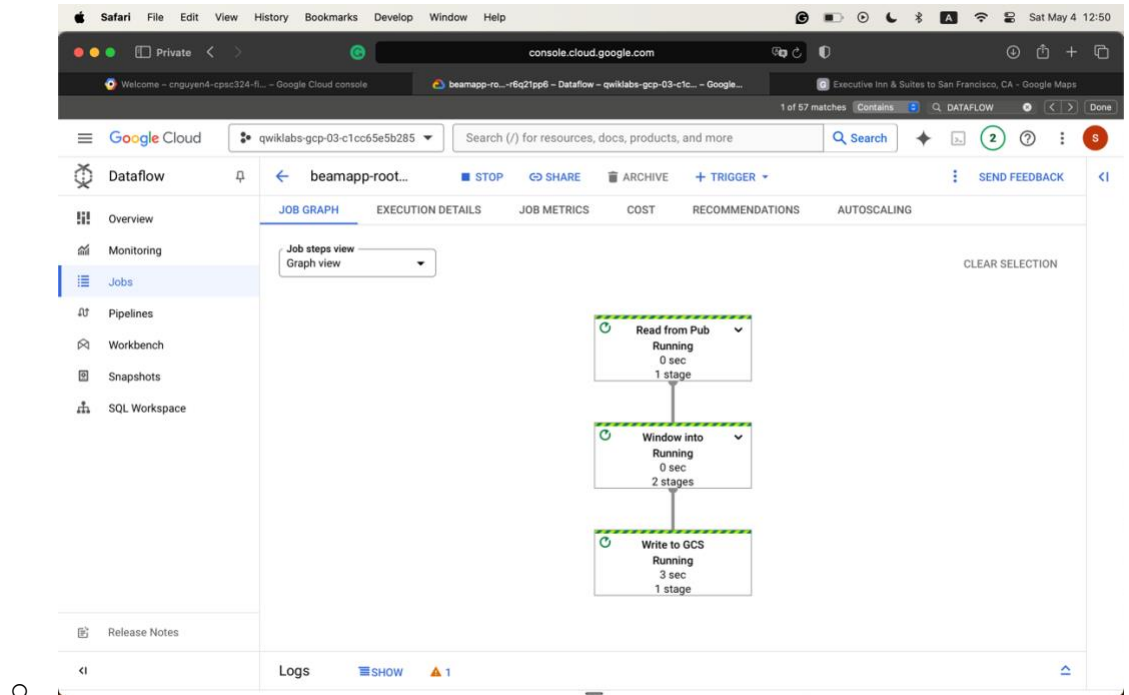
- Steps for the pipeline in task 15
 - Get data from two BigQuery sources
 - Join two data sources

- Filter out the header row
 - Convert the lines read to dictionary objects
 - Output the rows to BigQuery
- Graph for task 16



9. Question 9

- Graph photo



- Files generated in part 4

The screenshot shows the Google Cloud Storage console interface. The left sidebar contains navigation options: Buckets (selected), Monitoring, and Settings. The main panel displays the 'Bucket details' for a bucket named 'samples'. The table below lists the files generated in part 4.

Name	Size	Type	Created	Storage class	Last modified
output-07-34-07-36-1	34 B	application/octet-stream	May 4, 2024, 12:38:48 AM	Standard	May 4, 2024
output-07-36-07-38-1	68 B	application/octet-stream	May 4, 2024, 12:38:48 AM	Standard	May 4, 2024
output-07-38-07-40-0	34 B	application/octet-stream	May 4, 2024, 12:40:07 AM	Standard	May 4, 2024
output-07-38-07-40-1	34 B	application/octet-stream	May 4, 2024, 12:40:07 AM	Standard	May 4, 2024
output-07-40-07-42-0	34 B	application/octet-stream	May 4, 2024, 12:42:07 AM	Standard	May 4, 2024
output-07-40-07-42-1	34 B	application/octet-stream	May 4, 2024, 12:42:07 AM	Standard	May 4, 2024
output-07-42-07-44-0	34 B	application/octet-stream	May 4, 2024, 12:44:07 AM	Standard	May 4, 2024
output-07-42-07-44-1	34 B	application/octet-stream	May 4, 2024, 12:44:07 AM	Standard	May 4, 2024
output-07-44-07-46-1	68 B	application/octet-stream	May 4, 2024, 12:46:37 AM	Standard	May 4, 2024
output-07-46-07-48-0	34 B	application/octet-stream	May 4, 2024, 12:48:36 AM	Standard	May 4, 2024
output-07-46-07-48-1	34 B	application/octet-stream	May 4, 2024, 12:48:36 AM	Standard	May 4, 2024
output-07-48-07-50-0	34 B	application/octet-stream	May 4, 2024, 12:50:04 AM	Standard	May 4, 2024
output-07-48-07-50-1	34 B	application/octet-stream	May 4, 2024, 12:50:04 AM	Standard	May 4, 2024

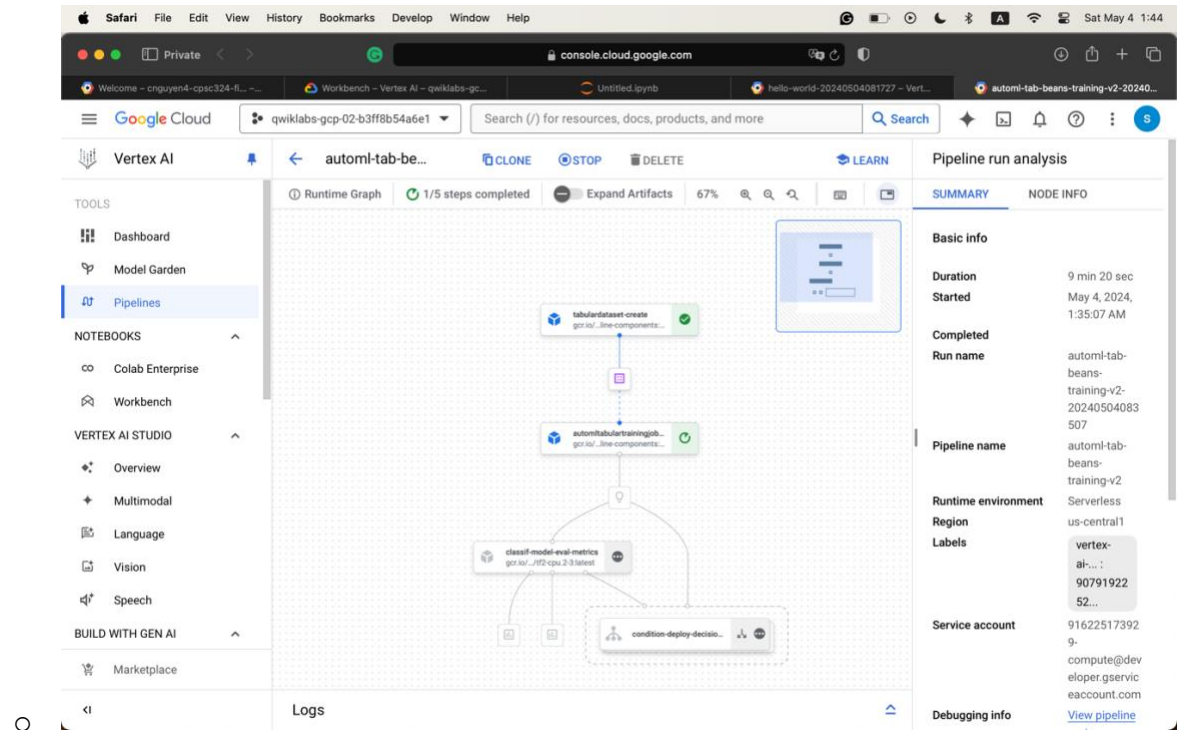
10. Question 10

- Three pipeline components
 - product_name: take the product name as input and return that string as output
 - emoji: take the text description of an emoji and convert it to an emoji
 - build_sentence: consume the output of the previous two to build a sentence that uses the emoji
- The pipeline
 - To set up the pipeline, use the `@dsl.pipeline` decorator
 - Write the intro_pipeline function
 - product_task: takes a product name as input
 - emoji_task: takes the text code for an emoji as input
 - consume_task: to execute the task. Has 3 inputs
 - output of product_task
 - output of emoji_task
 - the output of the previous two
- Screenshot of the pipeline running

The screenshot displays the Google Cloud Vertex AI console interface. On the left, a sidebar shows navigation options like Dashboard, Model Garden, Pipelines, and Notebooks. The main area shows a pipeline run analysis for a pipeline named 'hello-world-20...'. The pipeline graph consists of three tasks: 'emoji' (python3.9), 'product_name' (python3.9), and 'build_sentence' (python3.9). The 'build_sentence' task is the final step, receiving inputs from the other two. The right panel shows the 'Pipeline run analysis' summary, including basic info like run name, pipeline name, and runtime environment, and a table of run parameters.

Parameter	Type	Value
emoji_str	string	sparkles
text	string	Vertex Pipelines

- Screenshot of the pipeline from task 4



- The purpose of task 4
 - Custom evaluation: Evaluates model performance and decides whether to deploy.
 - Tabular Dataset Creation: Prepares data for training.
 - AutoML Tabular Training Job: Trains the model.
 - Model Deployment: Deploys the trained model.
 - Conditional Logic: Determines deployment based on evaluation metrics

11. Question 11

- Example screenshots
 - 500 epochs

```

Welcome - cnguyend-cpsc324-rl - Google Cloud console
model.py - code-server

home > ide-dev > model.py > ...
9 cloud_logger.addHandler(logging.StreamHandler())
10
11 # Import TensorFlow
12 import tensorflow as tf
13
14 # Import numpy
15 import numpy as np
16
17 xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
18 ys = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)
19
20 model = tf.keras.Sequential([tf.keras.layers.Dense(units=1, input_shape=[1])])
21
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Epoch 491/500
1/1 [=====] - 0s 4ms/step - loss: 3.7739e-06
Epoch 492/500
1/1 [=====] - 0s 4ms/step - loss: 3.6965e-06
Epoch 493/500
1/1 [=====] - 0s 3ms/step - loss: 3.6205e-06
Epoch 494/500
1/1 [=====] - 0s 3ms/step - loss: 3.5457e-06
Epoch 495/500
1/1 [=====] - 0s 4ms/step - loss: 3.4728e-06
Epoch 496/500
1/1 [=====] - 0s 5ms/step - loss: 3.4016e-06
Epoch 497/500
1/1 [=====] - 0s 4ms/step - loss: 3.3322e-06
Epoch 498/500
1/1 [=====] - 0s 5ms/step - loss: 3.2635e-06
Epoch 499/500
1/1 [=====] - 0s 5ms/step - loss: 3.1965e-06
Epoch 500/500
1/1 [=====] - 0s 3ms/step - loss: 3.1309e-06
[101.005165]
Waiting up to 5 seconds.
Sent all pending logs.
04:19:49 ide-dev@cloudlearningservices ~ -
ide-service-rew2q4l4ca-uc-a.run.app Python 3.7.3 64-bit 0 0 0
Ln 26, Col 46 Spaces: 4 UTF-8 LF Python Layout: U.S.

```

- 400 epochs

```

Welcome - cnguyend-cpsc324-rl - Google Cloud console
model.py - code-server

home > ide-dev > model.py > ...
21
22 model.compile(optimizer=tf.keras.optimizers.SGD(), loss=tf.keras.losses.MeanSquaredError())
23
24 model.fit(xs, ys, epochs=400)
25
26 cloud_logger.info(str(model.predict([10.0])))
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Epoch 391/400
1/1 [=====] - 0s 3ms/step - loss: 2.8423e-06
Epoch 392/400
1/1 [=====] - 0s 3ms/step - loss: 2.7841e-06
Epoch 393/400
1/1 [=====] - 0s 4ms/step - loss: 2.7268e-06
Epoch 394/400
1/1 [=====] - 0s 3ms/step - loss: 2.6709e-06
Epoch 395/400
1/1 [=====] - 0s 3ms/step - loss: 2.6162e-06
Epoch 396/400
1/1 [=====] - 0s 3ms/step - loss: 2.5626e-06
Epoch 397/400
1/1 [=====] - 0s 3ms/step - loss: 2.5096e-06
Epoch 398/400
1/1 [=====] - 0s 3ms/step - loss: 2.4582e-06
Epoch 399/400
1/1 [=====] - 0s 4ms/step - loss: 2.4077e-06
Epoch 400/400
1/1 [=====] - 0s 4ms/step - loss: 2.3581e-06
[131.004402]
Waiting up to 5 seconds.
Sent all pending logs.
04:21:19 ide-dev@cloudlearningservices ~ -
ide-service-rew2q4l4ca-uc-a.run.app Python 3.7.3 64-bit 0 0 0
Ln 24, Col 27 Spaces: 4 UTF-8 LF Python Layout: U.S.

```

- 1000 epochs


```

21 model.compile(optimizer=tf.keras.optimizers.SGD(), loss=tf.keras.losses.MeanSquaredError())
22
23 model.fit(xs, ys, epochs=1000)
24
25
26 cloud_logger.info(str(model.predict([10.0])))

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Epoch 991/1000
1/1 [=====] - 0s 3ms/step - loss: 5.4750e-11
Epoch 992/1000
1/1 [=====] - 0s 3ms/step - loss: 5.4300e-11
Epoch 993/1000
1/1 [=====] - 0s 3ms/step - loss: 5.2658e-11
Epoch 994/1000
1/1 [=====] - 0s 3ms/step - loss: 5.2477e-11
Epoch 995/1000
1/1 [=====] - 0s 4ms/step - loss: 5.2296e-11
Epoch 996/1000
1/1 [=====] - 0s 3ms/step - loss: 5.4845e-11
Epoch 997/1000
1/1 [=====] - 0s 3ms/step - loss: 5.0195e-11
Epoch 998/1000
1/1 [=====] - 0s 3ms/step - loss: 5.0010e-11
Epoch 999/1000
1/1 [=====] - 0s 3ms/step - loss: 4.9842e-11
Epoch 1000/1000
1/1 [=====] - 0s 3ms/step - loss: 4.9402e-11
1/1 [=====] - 0s 110ms/step
[31.000027]
Waiting up to 5 seconds.
Sent all pending logs.
04:22:01 ide-dev@cloudlearningservices ~ -

```

- 100 epochs

```

21 model.compile(optimizer=tf.keras.optimizers.SGD(), loss=tf.keras.losses.MeanSquaredError())
22
23 model.fit(xs, ys, epochs=100)
24
25
26 cloud_logger.info(str(model.predict([10.0])))

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Epoch 91/100
1/1 [=====] - 0s 3ms/step - loss: 0.0096
Epoch 92/100
1/1 [=====] - 0s 4ms/step - loss: 0.0094
Epoch 93/100
1/1 [=====] - 0s 3ms/step - loss: 0.0092
Epoch 94/100
1/1 [=====] - 0s 4ms/step - loss: 0.0090
Epoch 95/100
1/1 [=====] - 0s 4ms/step - loss: 0.0088
Epoch 96/100
1/1 [=====] - 0s 3ms/step - loss: 0.0086
Epoch 97/100
1/1 [=====] - 0s 3ms/step - loss: 0.0084
Epoch 98/100
1/1 [=====] - 0s 3ms/step - loss: 0.0083
Epoch 99/100
1/1 [=====] - 0s 3ms/step - loss: 0.0081
Epoch 100/100
1/1 [=====] - 0s 3ms/step - loss: 0.0079
1/1 [=====] - 0s 110ms/step
[31.259002]
Waiting up to 5 seconds.
Sent all pending logs.
04:25:10 ide-dev@cloudlearningservices ~ -

```

- The lower the epoch number, the less accurate the result is
- Script
 - `import logging`
 - `import google.cloud.logging as cloud_logging`

```

○ from google.cloud.logging.handlers import CloudLoggingHandler
○ from google.cloud.logging_v2.handlers import setup_logging
○
○ cloud_logger = logging.getLogger('cloudLogger')
○ cloud_logger.setLevel(logging.INFO)
○ cloud_logger.addHandler(CloudLoggingHandler(cloud_logging.Client()))
○ cloud_logger.addHandler(logging.StreamHandler())
○
○ # Import TensorFlow
○ import tensorflow as tf
○
○ # Import numpy
○ import numpy as np
○
○ xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
○ ys = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)
○
○ model = tf.keras.Sequential([tf.keras.layers.Dense(units=1, input_shape=[1])])
○
○ model.compile(optimizer=tf.keras.optimizers.SGD(), loss=tf.keras.losses.MeanSquaredError())
○
○ model.fit(xs, ys, epochs=100)
○
○ cloud_logger.info(str(model.predict([10.0])))

```

12. Question 12

- Describe the structure of the code
 - A sequence of task
 - Flatten the image from a square matrix shape to a 1-D vector
 - Dense: adds a layer of neurons
 - There are two calls on Dense
 - Relu: if $X > 0$, return X , else 0. It passes 0 or greater to the next layer in the network

- Softmax: takes a set of values and effectively picks the biggest one so that we won't have to sort
- Evaluation at the end of step 4

The screenshot shows a web-based IDE interface with a code editor and a terminal. The code editor displays a Python script for training a Keras model. The terminal shows the execution output, including training progress and evaluation results.

```

36 image_batch, labels_batch = next(iter(ds_train))
37 print("After normalization ->", np.min(image_batch[0]), np.max(image_batch[0]))
38
39 # Define the model
40 model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),
41                                     tf.keras.layers.Dense(64, activation=tf.nn.relu),
42                                     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
43
44 # Compile the model
45 model.compile(optimizer = tf.keras.optimizers.Adam(),
46               loss = tf.keras.losses.SparseCategoricalCrossentropy(),
47               metrics=[tf.keras.metrics.SparseCategoricalAccuracy()])
48
49 model.fit(ds_train, epochs=5)
50
51 cloud_logger.info(model.evaluate(ds_test))

```

The terminal output shows the following training progress and evaluation results:

```

Epoch 2/5
1875/1875 [=====] - 5s 3ms/step - loss: 0.3895 - sparse_categorical_accuracy: 0.8619
Epoch 3/5
1875/1875 [=====] - 5s 3ms/step - loss: 0.3509 - sparse_categorical_accuracy: 0.8739
Epoch 4/5
1875/1875 [=====] - 5s 3ms/step - loss: 0.3256 - sparse_categorical_accuracy: 0.8817
Epoch 5/5
1875/1875 [=====] - 5s 3ms/step - loss: 0.3074 - sparse_categorical_accuracy: 0.8883
313/313 [=====] - 1s 4ms/step - loss: 0.3657 - sparse_categorical_accuracy: 0.8702
[0.36572569608688354, 0.870199978351593]
INFO:cloudLogger:[0.36572569608688354, 0.870199978351593]
Waiting up to 5 seconds.
Sent all pending logs.
04:47:32 ide-dev@cloudlearningservices ~ -

```

- Evaluation in step 7

```

home > ide-dev > model.py > ...
47 | | | metrics=[tf.keras.metrics.SparseCategoricalAccuracy()]
48 |
49 | model.fit(ds_train, epochs=5)
50 |
51 | cloud_logger.info(model.evaluate(ds_test))
52 |
=====
Layer (type)                Output Shape                Param #
=====
flatten (Flatten)           (None, 784)                 0
dense (Dense)               (None, 64)                 50240
dense_1 (Dense)             (None, 10)                 650
=====
Total params: 50,890
Trainable params: 50,890
Non-trainable params: 0

Model: "sequential"
=====
Layer (type)                Output Shape                Param #
=====
flatten (Flatten)           (None, 784)                 0
dense (Dense)               (None, 64)                 50240
dense_1 (Dense)             (None, 10)                 650
=====
Total params: 50,890
Trainable params: 50,890
Non-trainable params: 0

Waiting up to 5 seconds.
Sent all pending logs.
04:50:19 ide-dev@cloudlearningservices ~ -

```

- Screenshot of the evaluation exercise 1

```

home > ide-dev > updated_model.py > ...
11 | import tensorflow_datasets as tfds
12 | # Import numpy
13 | import numpy as np
14 | # Import TensorFlow
15 | import tensorflow as tf
16 |
17 | # Define, load and configure data
18 | (ds_train, ds_test), info = tfds.load('fashion_mnist', split=['train', 'test'], with_info=True, as_supervised=True)
19 | # Define batch size
20 | BATCH_SIZE = 32
21 | # Normalizing and batch processing of data
22 | ds_train = ds_train.map(lambda x, y: (tf.cast(x, tf.float32)/255.0, y)).batch(BATCH_SIZE)
23 | ds_test = ds_test.map(lambda x, y: (tf.cast(x, tf.float32)/255.0, y)).batch(BATCH_SIZE)
24 | # Define the model
=====
1875/1875 [=====] - 8s 4ms/step - loss: 0.3743 - sparse_categorical_accuracy: 0.8647
Epoch 3/5
1875/1875 [=====] - 7s 4ms/step - loss: 0.3348 - sparse_categorical_accuracy: 0.8772
Epoch 4/5
1875/1875 [=====] - 7s 4ms/step - loss: 0.3117 - sparse_categorical_accuracy: 0.8856
Epoch 5/5
1875/1875 [=====] - 7s 4ms/step - loss: 0.2919 - sparse_categorical_accuracy: 0.8920
INFO:upLogger:Model: "sequential"
INFO:upLogger:
INFO:upLogger:Layer (type)                Output Shape                Param #
INFO:upLogger:=====
INFO:upLogger:flatten (Flatten)           (None, 784)                 0
INFO:upLogger:
INFO:upLogger:dense (Dense)               (None, 128)                100480
INFO:upLogger:
INFO:upLogger:dense_1 (Dense)             (None, 10)                 1290
INFO:upLogger:=====
INFO:upLogger:Total params: 101,770
INFO:upLogger:Trainable params: 101,770
INFO:upLogger:Non-trainable params: 0
INFO:upLogger:
Program shutting down, attempting to send 14 queued log entries to Cloud Logging...
Waiting up to 5 seconds.
Sent all pending logs.
04:59:23 ide-dev@cloudlearningservices ~ -

```

- Screenshot of the evaluation exercise 2

```

home > ide-dev > updated_model.py > ...
14 # Import TensorFlow
15 import tensorflow as tf
16
17 # Define, load and configure data
18 (ds_train, ds_test), info = tfds.load('fashion_mnist', split=['train', 'test'], with_info=True, as_supervised=True)
19 # Define batch size
20 BATCH_SIZE = 32
21 # Normalizing and batch processing of data
22 ds_train = ds_train.map(lambda x, y: (tf.cast(x, tf.float32)/255.0, y)).batch(BATCH_SIZE)
23 ds_test = ds_test.map(lambda x, y: (tf.cast(x, tf.float32)/255.0, y)).batch(BATCH_SIZE)
24 # Define the model
25 model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),
26                                     tf.keras.layers.Dense(64, activation=tf.nn.relu),
27                                     tf.keras.layers.Dense(64, activation=tf.nn.relu),
28                                     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
29 # Compile data

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Epoch 5/5
1875/1875 [=====] - 7s 4ms/step - loss: 0.2990 - sparse_categorical_accuracy: 0.8894
INFO:upLogger:Model: "sequential"
INFO:upLogger:
INFO:upLogger: Layer (type) Output Shape Param #
INFO:upLogger:=====
INFO:upLogger: Flatten (Flatten) (None, 784) 0
INFO:upLogger:
INFO:upLogger: dense (Dense) (None, 64) 50240
INFO:upLogger:
INFO:upLogger: dense_1 (Dense) (None, 64) 4160
INFO:upLogger:
INFO:upLogger: dense_2 (Dense) (None, 10) 650
INFO:upLogger:
INFO:upLogger:=====
INFO:upLogger:Total params: 55,050
INFO:upLogger:Trainable params: 55,050
INFO:upLogger:Non-trainable params: 0
INFO:upLogger:
Program shutting down, attempting to send 7 queued log entries to Cloud Logging...
Waiting up to 5 seconds.
Sent all pending logs.
05:02:32 ide-dev@cloudlearningservices ~ -

```

- Screenshot of the evaluation exercise 3

```

home > ide-dev > updated_model.py > ...
29 tf.keras.layers.Dense(64, activation=tf.nn.relu),
30 tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
31 # Compile data
32 model.compile(optimizer = tf.keras.optimizers.Adam(),
33               loss = tf.keras.losses.SparseCategoricalCrossentropy(),
34               metrics=[tf.keras.metrics.SparseCategoricalAccuracy()])
35 model.fit(ds_train, epochs=5)
36 # Logs model summary
37 model.summary(print_fn=up_logger.info)
38
39 # Print out max value to see the changes
40 image_batch, labels_batch = next(iter(ds_train))
41 t_image_batch, t_labels_batch = next(iter(ds_test))
42 up_logger.info("training images max " + str(np.max(image_batch[0])))
43 up_logger.info("test images max " + str(np.max(t_image_batch[0])))

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Epoch 5/5
1875/1875 [=====] - 4s 2ms/step - loss: 0.5522 - sparse_categorical_accuracy: 0.8085
INFO:upLogger:Model: "sequential"
INFO:upLogger:
INFO:upLogger: Layer (type) Output Shape Param #
INFO:upLogger:=====
INFO:upLogger: flatten (Flatten) (None, 784) 0
INFO:upLogger:
INFO:upLogger: dense (Dense) (None, 64) 50240
INFO:upLogger:
INFO:upLogger: dense_1 (Dense) (None, 10) 650
INFO:upLogger:
INFO:upLogger:=====
INFO:upLogger:Total params: 50,890
INFO:upLogger:Trainable params: 50,890
INFO:upLogger:Non-trainable params: 0
INFO:upLogger:
INFO:upLogger:training images max 255
INFO:upLogger:test images max 255
Program shutting down, attempting to send 16 queued log entries to Cloud Logging...
Waiting up to 5 seconds.
Sent all pending logs.
05:06:32 ide-dev@cloudlearningservices ~ -

```

13. Question 13

- I choose to work with the Introduction to Convolutions with TensorFlow lab

- Purpose of the lab
 - This lab shows how to train an image classifier using a deep neural network.
 - It also teaches the fundamentals of convolutional neural network
 - It shows how convolutions operate on images to extract features, using filters to identify patterns and pooling to reduce dimensionality while preserving important features.
- Outcome of the lab
 - Understand the basic concept.
 - Understand how these operations affect image processing and utilize these techniques to build an image classifier.
- What did I learn?
 - The basics of convolutional neural network. This is, in fact, my first time working with neural networks despite having to talk about it so many times.

14. Question 14

- I choose to work with the Classify Images with TensorFlow Convolutional Neural Networks lab.
- Purpose of the lab
 - Show how to train and deploy a custom Convolutional Neural Network (CNN) for image classification using TensorFlow on Vertex AI.
- Outcome of the lab
 - Training a custom CNN model using the Vertex AI.
 - Deploying the trained model for online prediction.

- Understanding how to manipulate TensorFlow datasets, build CNN models, and make predictions.
 - Deploying and making predictions with the model
- What did I learn
 - This lab allows me to learn many important things about this technology, such as how to use it, how to set it up, and more.
 - It also shows me techniques to manipulate and process the data, especially with vision computing.