5

Informe Big Data Sumativa Nro. 3

* **ASIGNATURA: Big Data**
* **SECCIÓN: 002D**
* **DOCENTE: Leonardo Hernandez**
* **INTEGRANTES: Matias Bello - Bastian Olivares**
* **GRUPO 9**

Índice

[**1. URLS DE DATOS DIARIOS Y ESTRUCTURA JSON DE CADA UNA 1**](#_heading=h.30j0zll)

[**2. ARQUITECTURA REAL-TIME Y DESCRIPCIÓN DE SERVICIOS 1**](#_heading=h.1fob9te)

[**3. CÓDIGOS Y EXPLICACION PROCESOS AUTOMATIZADOS 2**](#_heading=h.3znysh7)

[**4. PASOS REALIZADOS 5**](#_heading=h.2et92p0)

[**5. GENERE 3 PREGUNTAS A PARTIR DE LOS DATOS QUE CONTRIBUYAN MEJORAR EL SERVICIO DE TRANSPORTE 10**](#_heading=h.28edg693bcwa)

[**6. CONSTRUYA LAS CONSULTAS SQL EN BIGQUERY QUE RESPONDA A LAS PREGUNTAS FORMULADAS 11**](#_heading=h.pqqk53fz21kp)

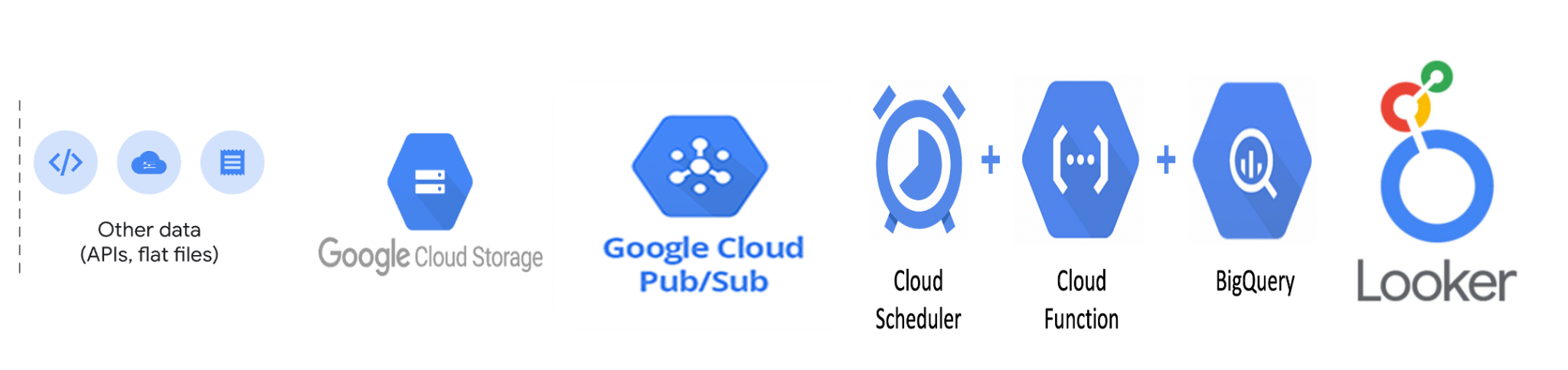
[**7. CONSTRUCCIÓN DE REPORTES Y VISUALIZACIONES 12**](#_heading=h.sy3u8cfd3qq2)

[**8. CONCLUSIONES 13**](#_heading=h.4d34og8)

# URLS DE DATOS DIARIOS Y ESTRUCTURA JSON DE CADA UNA

* 1. <https://www.red.cl/restservice_v2/rest/getservicios/all>
  2. [https://red.diego0alonso.me/restservice\_v2/rest/conocerecorrido](https://red.diego0alonso.me/restservice_v2/rest/conocerecorrido?codsint=101)

# ARQUITECTURA REAL-TIME Y DESCRIPCIÓN DE SERVICIOS



# CÓDIGOS Y EXPLICACION PROCESOS AUTOMATIZADOS

**Main.py:**

import requests

from google.cloud import bigquery

import logging

# Configura el logging

logging.basicConfig(level=logging.INFO)

# Función que obtiene las rutas diarias

def rutas\_diarias(event, context):

try:

url = "https://www.red.cl/restservice\_v2/rest/getservicios/all"

response = requests.get(url)

response.raise\_for\_status() # Verifica si la solicitud fue exitosa

services = response.json()

for service in services:

service\_code = service

detalles\_rutas(service\_code)

except requests.exceptions.RequestException as e:

logging.error(f"Error al obtener las rutas diarias: {e}")

# Función que obtiene los detalles de las rutas

def detalles\_rutas(service\_code):

try:

url = f"https://red.diego0alonso.me/restservice\_v2/rest/conocerecorrido?codsint={service\_code}"

response = requests.get(url)

response.raise\_for\_status() # Verifica si la solicitud fue exitosa

data = response.json()

client = bigquery.Client()

dataset\_id = "transporte-publico-423221.transporte"

# Insertar datos en la tabla negocio

negocio\_row = [

{

"id": data.get("negocio", {}).get("id"),

"nombre": data.get("negocio", {}).get("nombre"),

"color": data.get("negocio", {}).get("color"),

"url": data.get("negocio", {}).get("url"),

}

]

insert\_to\_bq(client, f"{dataset\_id}.negocio", negocio\_row)

# Insertar datos en la tabla ida

ida\_id = data.get("ida", {}).get("id")

ida\_row = [

{

"id": ida\_id,

"negocio\_id": data.get("negocio", {}).get("id"),

"destino": data.get("ida", {}).get("destino"),

"itinerario": data.get("ida", {}).get("itinerario"),

}

]

insert\_to\_bq(client, f"{dataset\_id}.ida", ida\_row)

# Insertar datos en la tabla horarios\_ida

horarios\_ida\_rows = [

{

"ida\_id": ida\_id,

"tipoDia": horario.get("tipoDia"),

"inicio": horario.get("inicio"),

"fin": horario.get("fin"),

}

for horario in data.get("ida", {}).get("horarios", [])

]

insert\_to\_bq(client, f"{dataset\_id}.horarios\_ida", horarios\_ida\_rows)

# Insertar datos en la tabla path\_ida

path\_ida\_rows = [

{"ida\_id": ida\_id, "latitud": coord[0], "longitud": coord[1]}

for coord in data.get("ida", {}).get("path", [])

]

insert\_to\_bq(client, f"{dataset\_id}.path\_ida", path\_ida\_rows)

# Insertar datos en la tabla paraderos y servicios\_paraderos

for paradero in data.get("ida", {}).get("paraderos", []):

paradero\_id = paradero.get("id")

paradero\_row = [

{

"id": paradero\_id,

"cod": paradero.get("cod"),

"num": paradero.get("num"),

"latitud": (

paradero.get("pos", [])[1] if paradero.get("pos") else None

),

"longitud": (

paradero.get("pos", [])[0] if paradero.get("pos") else None

),

"name": paradero.get("name"),

"comuna": paradero.get("comuna"),

"type": paradero.get("type"),

"eje": paradero.get("eje"),

"codSimt": paradero.get("codSimt"),

"distancia": paradero.get("distancia"),

"stop\_id": paradero.get("stop", {}).get("id"),

}

]

insert\_to\_bq(client, f"{dataset\_id}.paraderos", paradero\_row)

servicios\_paraderos\_rows = [

{

"paradero\_id": paradero\_id,

"id": servicio.get("id"),

"cod": servicio.get("cod"),

"destino": servicio.get("destino"),

"orden": servicio.get("orden"),

"color": servicio.get("color"),

"negocio\_nombre": servicio.get("negocio", {}).get("nombre"),

"negocio\_color": servicio.get("negocio", {}).get("color"),

"recorrido\_destino": servicio.get("recorrido", {}).get("destino"),

"codigo": servicio.get("codigo"),

}

for servicio in paradero.get("servicios", [])

]

insert\_to\_bq(

client, f"{dataset\_id}.servicios\_paraderos", servicios\_paraderos\_rows

)

# Insertar datos en la tabla regreso

regreso\_id = data.get("regreso", {}).get("id")

regreso\_row = [

{

"id": regreso\_id,

"destino": data.get("regreso", {}).get("destino"),

"itinerario": data.get("regreso", {}).get("itinerario"),

}

]

insert\_to\_bq(client, f"{dataset\_id}.regreso", regreso\_row)

# Insertar datos en la tabla horarios\_regreso

horarios\_regreso\_rows = [

{

"regreso\_id": regreso\_id,

"tipoDia": horario.get("tipoDia"),

"inicio": horario.get("inicio"),

"fin": horario.get("fin"),

}

for horario in data.get("regreso", {}).get("horarios", [])

]

insert\_to\_bq(client, f"{dataset\_id}.horarios\_regreso", horarios\_regreso\_rows)

# Insertar datos en la tabla path\_regreso

path\_regreso\_rows = [

{"regreso\_id": regreso\_id, "latitud": coord[0], "longitud": coord[1]}

for coord in data.get("regreso", {}).get("path", [])

]

insert\_to\_bq(client, f"{dataset\_id}.path\_regreso", path\_regreso\_rows)

logging.info(

f"Datos del servicio {service\_code} insertados exitosamente en BigQuery"

)

except requests.exceptions.RequestException as e:

logging.error(f"Error al obtener los detalles de la ruta {service\_code}: {e}")

except Exception as e:

logging.error(

f"Error al insertar los datos en BigQuery para la ruta {service\_code}: {e}"

)

def insert\_to\_bq(client, table\_id, rows):

if rows:

errors = client.insert\_rows\_json(table\_id, rows)

if errors:

logging.error(

f"Errores encontrados mientras se insertaban las filas en {table\_id}: {errors}"

)

else:

logging.info(f"Filas insertadas exitosamente en {table\_id}")

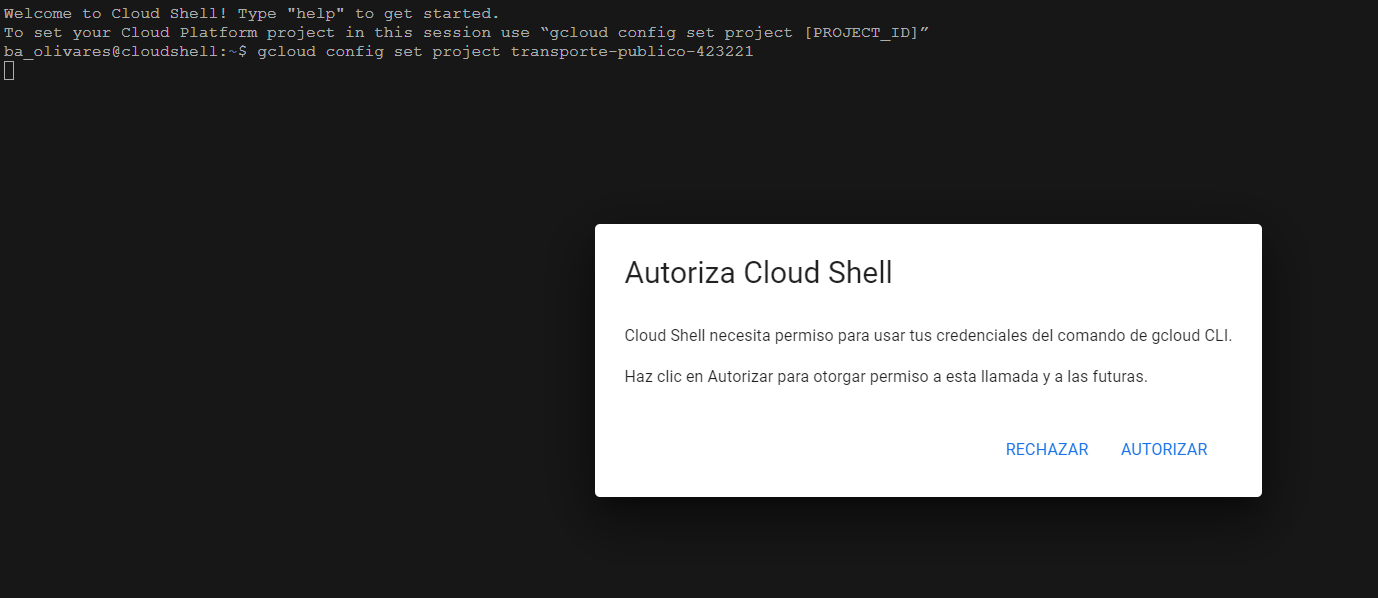
if \_\_name\_\_ == "\_\_main\_\_":

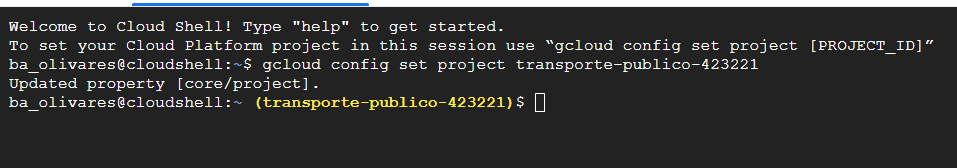
rutas\_diarias(None, None)

**Explicación del código:** El código obtiene datos de rutas de transporte público desde una API, los procesa y los inserta en BigQuery utilizando “requests” para las solicitudes y el cliente de BigQuery para las inserciones. Además, configura el logging para registrar el progreso y los errores.

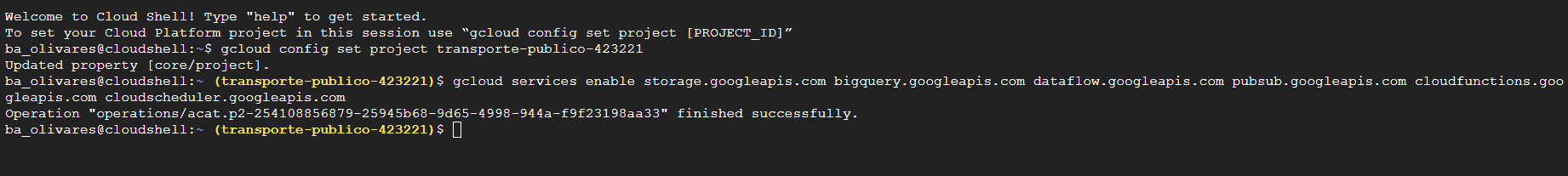
# PASOS REALIZADOS

1. El siguiente comando configura nuestro proyecto en la herramienta gcloud, simplificando su uso al evitar la necesidad de especificar el proyecto en cada comando. Esto mejora la eficiencia y precisión al interactuar con los servicios de Google Cloud Platform.

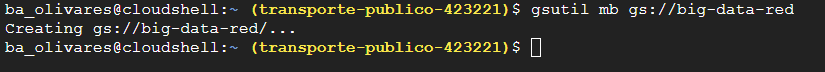




2. Habilitación de servicios de Google Cloud necesarios para el desarrollo y despliegue de aplicaciones que utilizan múltiples funcionalidades de la nube.



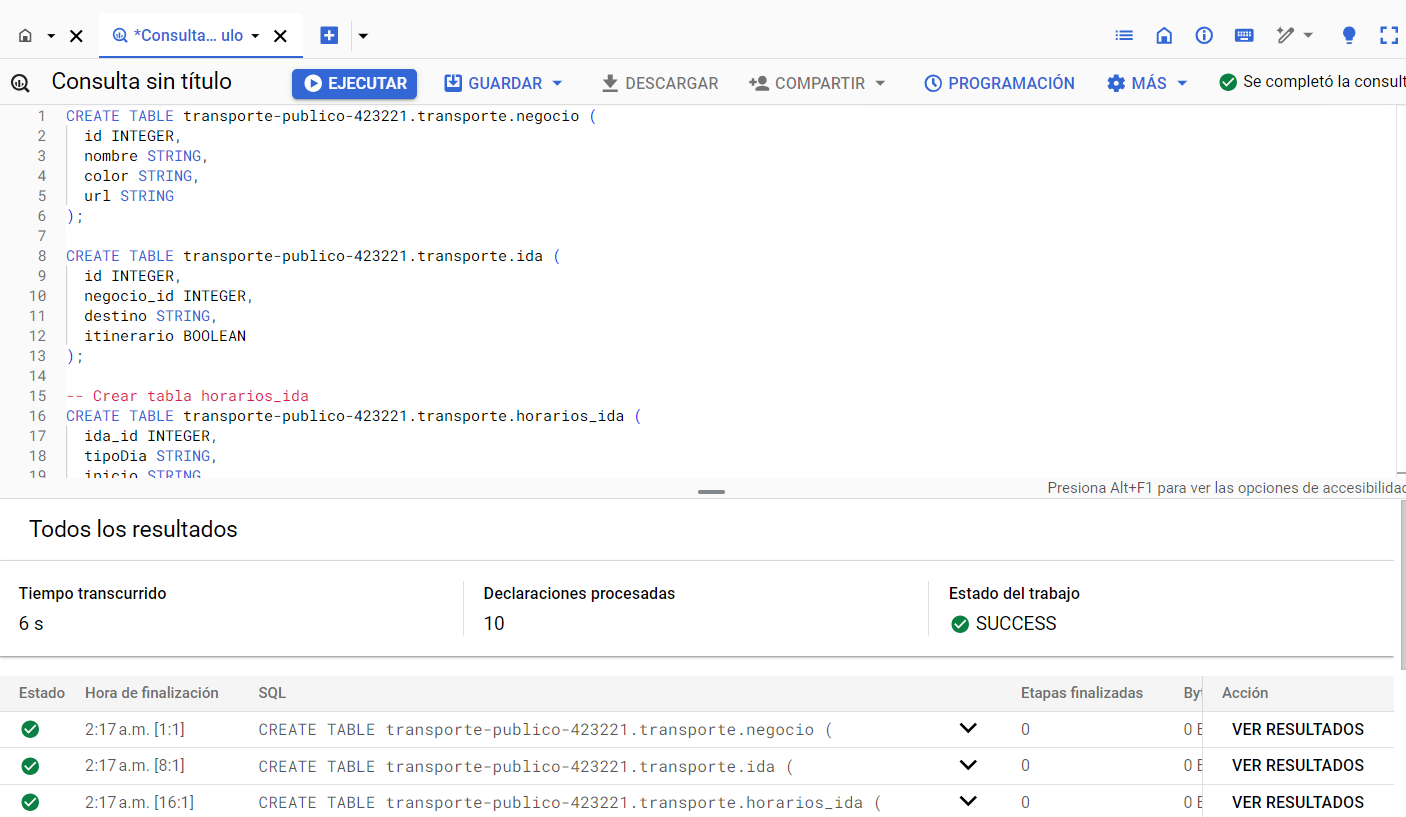
3. Creación del Bucket con el nombre de “big-data-red”.



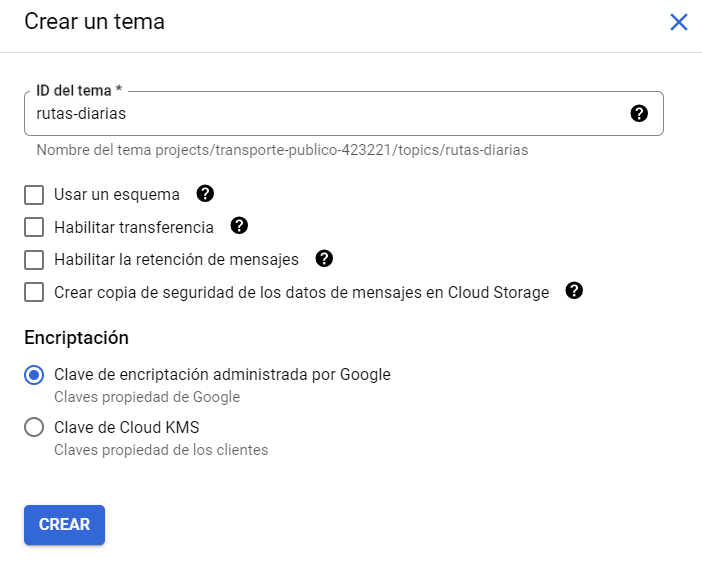
4. Creación de la base de datos del proyecto llamada “transporte”, ubicada en EE.UU, cuya descripción es “Dataset para datos de transporte”.



5. Creación de las tablas de la base de datos del proyecto.

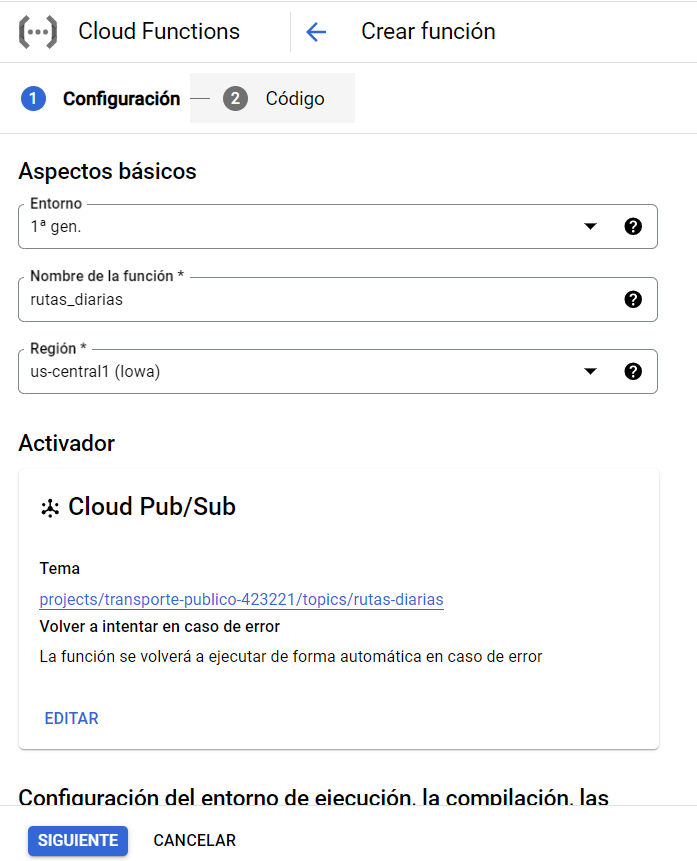


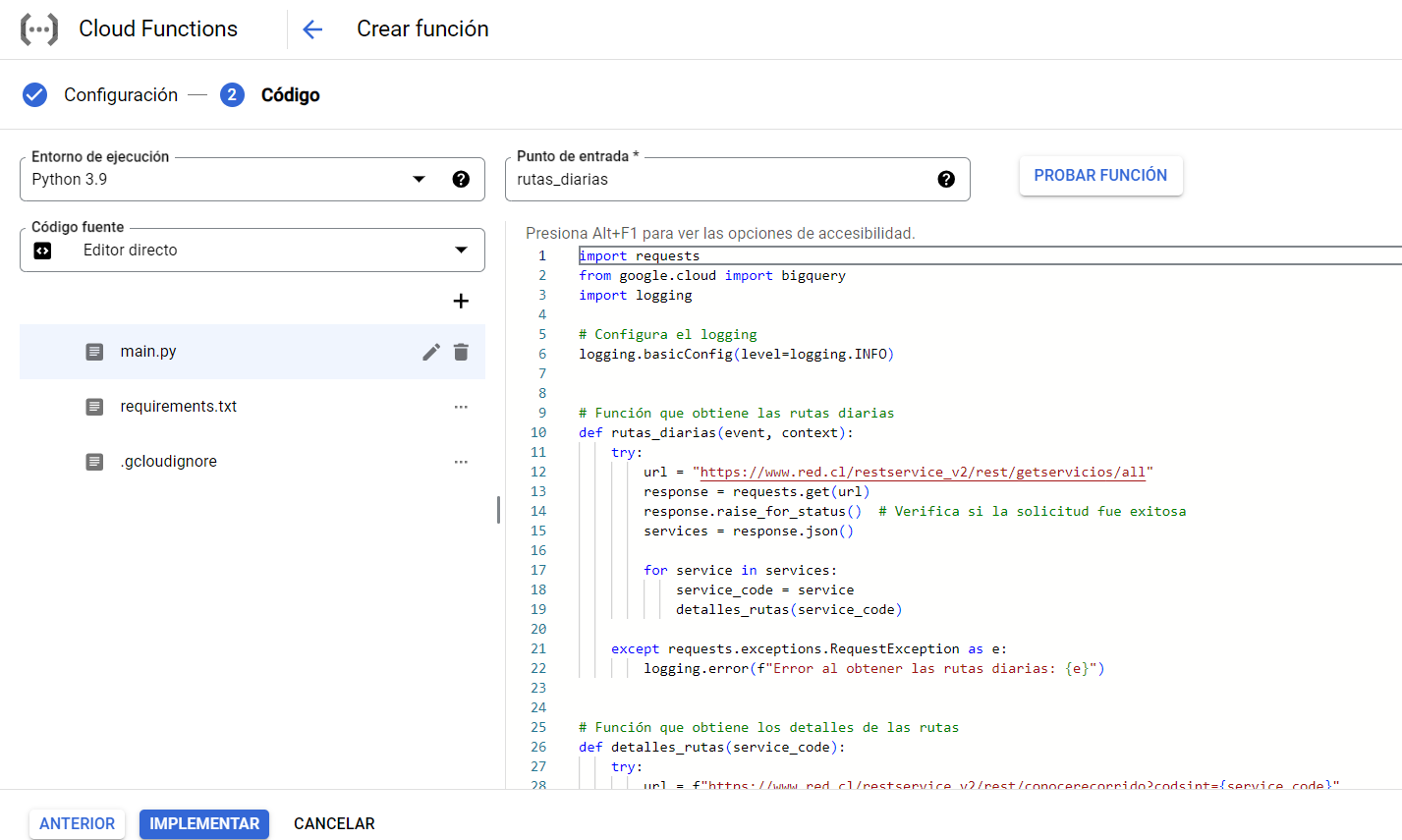
6.Creación de un tema, que es un canal de comunicación para la transmisión de mensajes. Marcando la casilla “Clave de encriptación administrada por google” para garantizar la seguridad de los datos en tránsito.



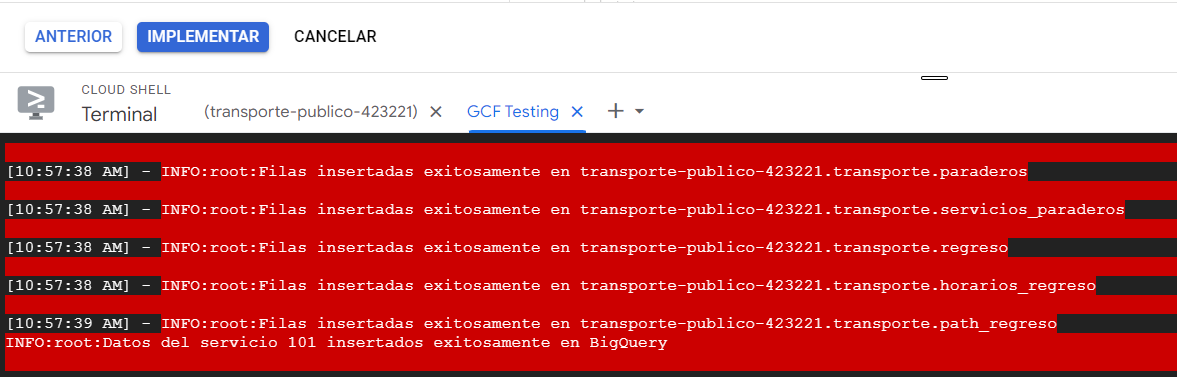
7. Creación de una función cloud, bajo el lenguaje de programación “Python 3.9”

Como resultado se insertan filas en el bucket

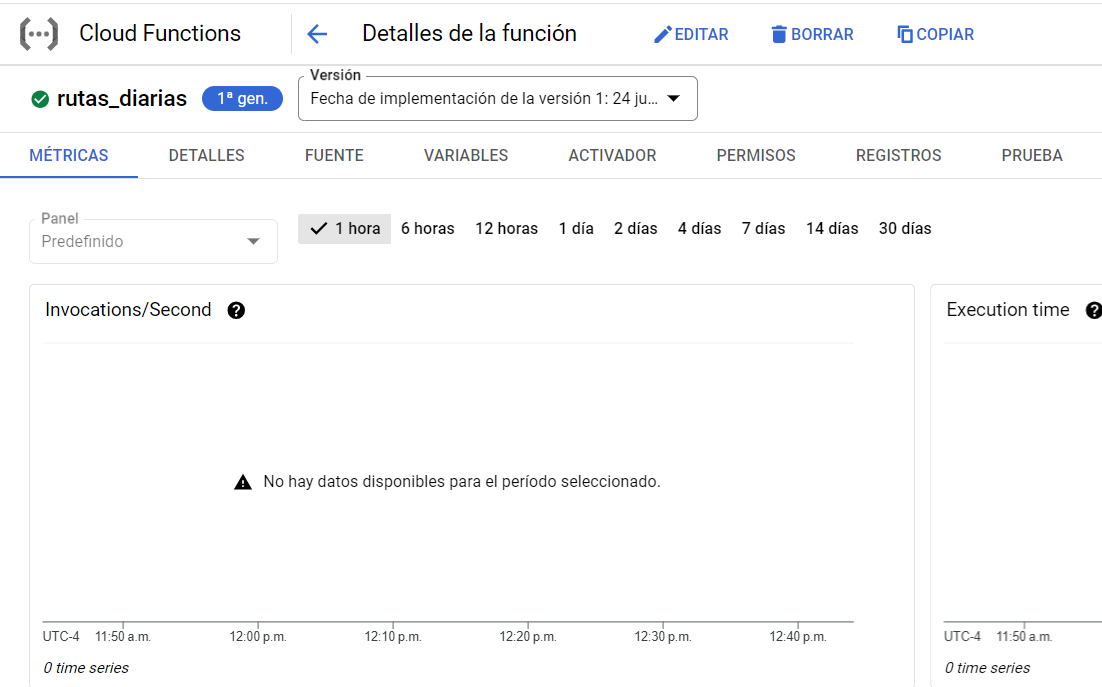




8. Inserción de los datos en las tablas



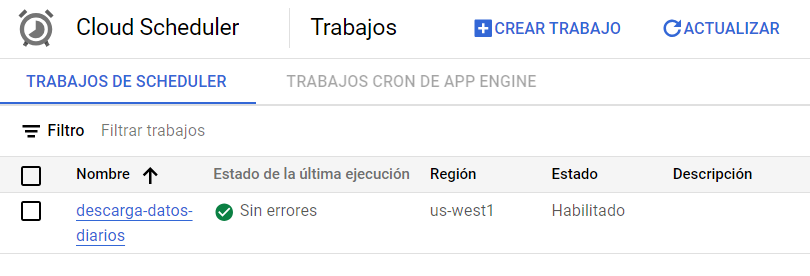
9. Implementación de la rutas\_diarias

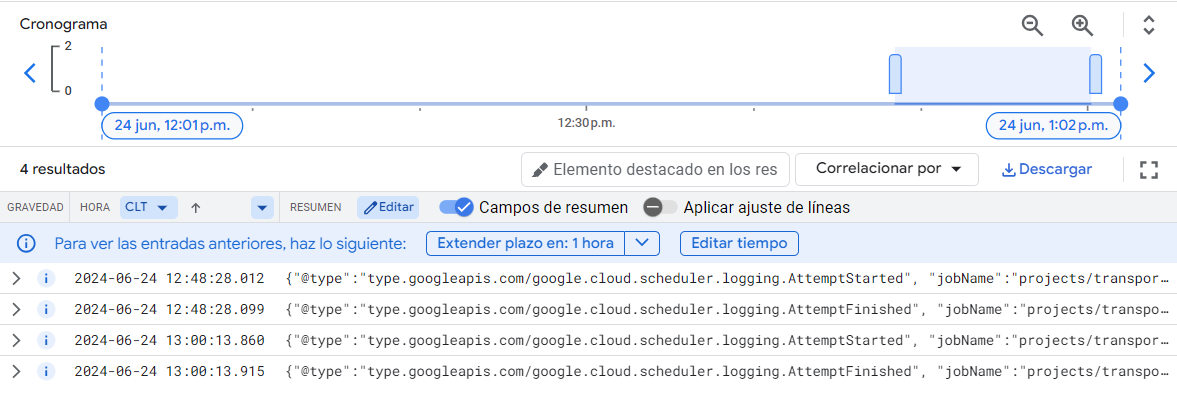


10. El comando crea un trabajo en Google Cloud Scheduler llamado descarga-datos-diarios, que se ejecuta diariamente a medianoche en la zona horaria de Santiago de Chile. Este trabajo publica un mensaje vacío en el tema de Pub/Sub rutas-diarias y está configurado en la región us-west1.



11. Automatización de la ejecución de las funciones.



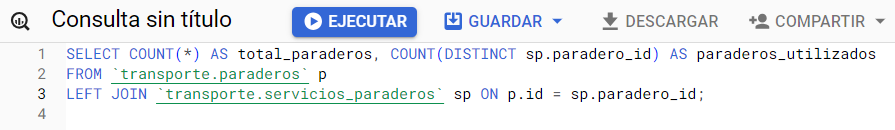
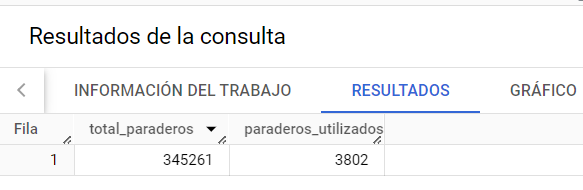
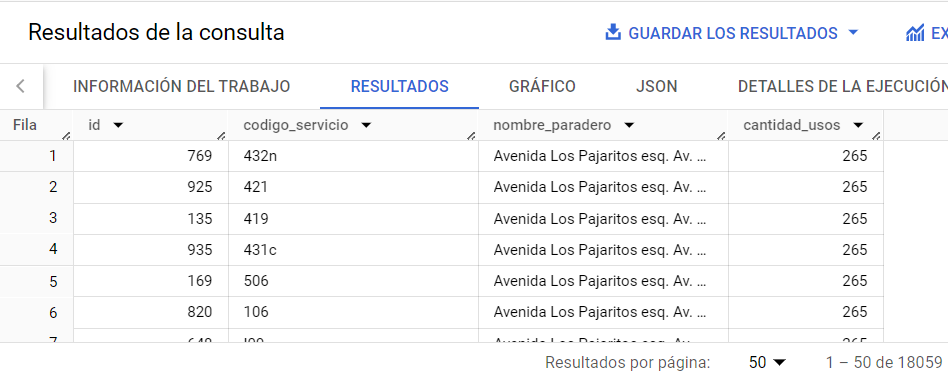
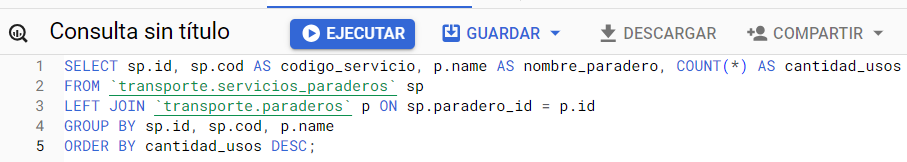


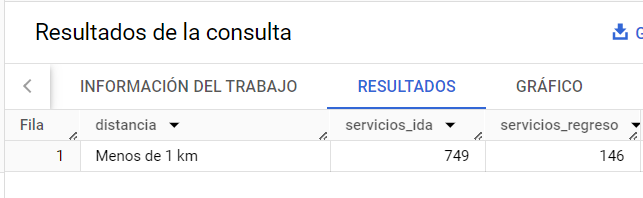
# GENERE 3 PREGUNTAS A PARTIR DE LOS DATOS QUE CONTRIBUYAN MEJORAR EL SERVICIO DE TRANSPORTE

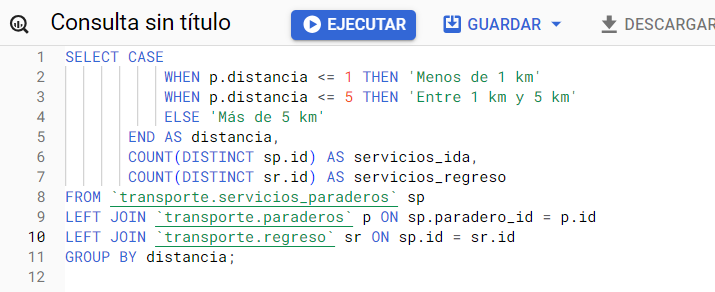
**Pregunta 1:** ¿Cuántos paraderos están registrados y cuántos de ellos se utilizan en servicios de ida y regreso?  
**Pregunta 2:** ¿Cuáles son los paraderos más utilizados por cada servicio de ida y regreso?

**Pregunta 3:** ¿Cuántos servicios de ida y regreso están asociados a paraderos que tienen una distancia específica desde un punto de referencia?

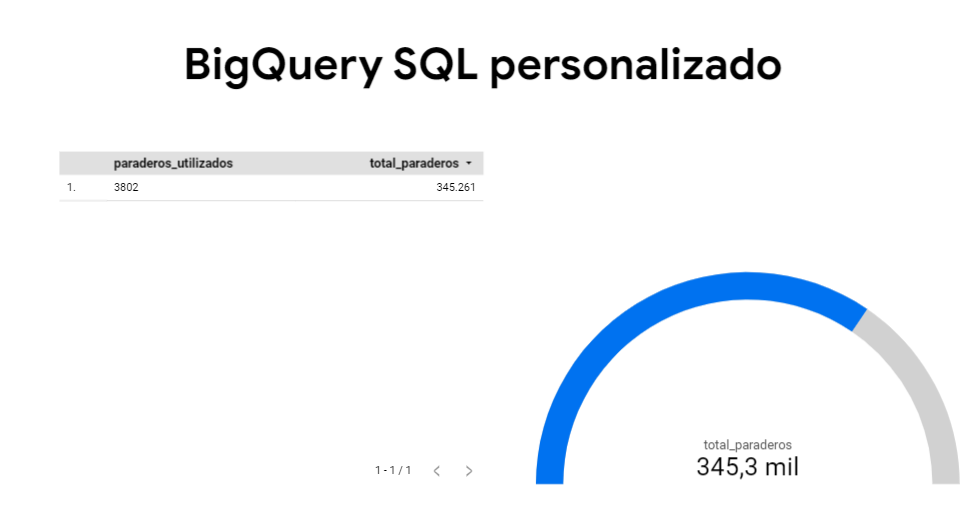
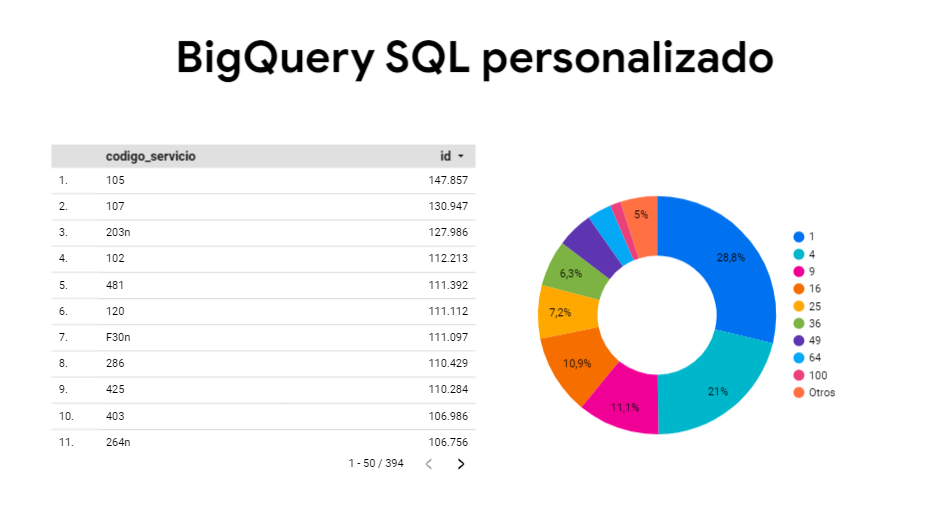
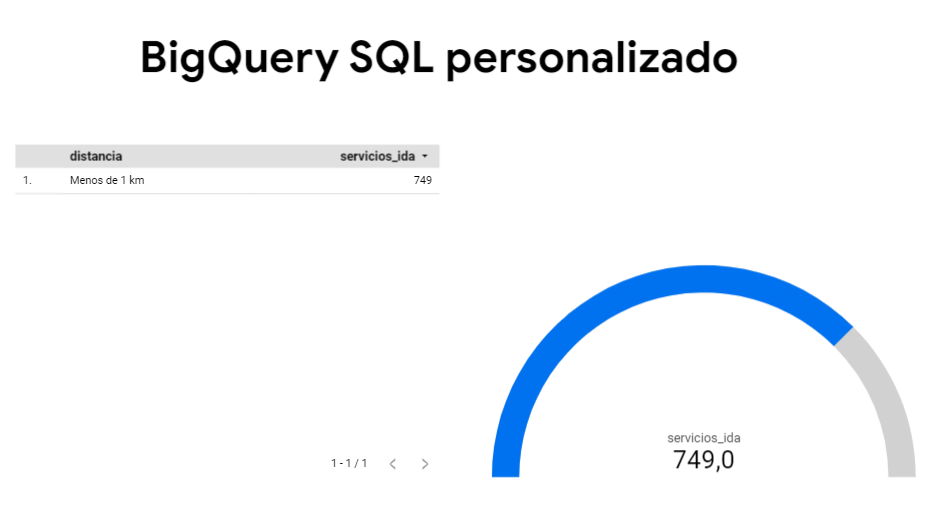
# CONSTRUYA LAS CONSULTAS SQL EN BIGQUERY QUE RESPONDA A LAS PREGUNTAS FORMULADAS

**Pregunta 1:** ¿Cuántos paraderos están registrados y cuántos de ellos se utilizan en servicios de ida y regreso? **  
Pregunta 2:** ¿Cuáles son los paraderos más utilizados por cada servicio de ida y regreso?  
  
**Pregunta 3:** ¿Cuántos servicios de ida y regreso están asociados a paraderos que tienen una distancia superior a un kilómetro uno del otro?





# CONSTRUCCIÓN DE REPORTES Y VISUALIZACIONES

Visualización pregunta 1:   
  
Visualización pregunta 2:  
  
Visualización pregunta 3:  


# CONCLUSIONES

El informe describe el desarrollo e implementación de una arquitectura de procesamiento de datos en tiempo real para mejorar la gestión del transporte público. Utilizando Python 3.9 y Google Cloud Platform, se automatizó la recolección y almacenamiento de datos de rutas de transporte, integrándolos en BigQuery. Se llevaron a cabo varios pasos clave, como la configuración de Google Cloud y la implementación de funciones en la nube. Las consultas SQL en BigQuery permitieron responder a preguntas sobre el uso y eficiencia de los paraderos, proporcionando información valiosa para mejorar el sistema. La visualización de datos ayudó a interpretar los resultados y tomar decisiones informadas. El proyecto demuestra la efectividad de las tecnologías de Big Data y computación en la nube para optimizar servicios públicos.