

GCP Lambda Data Pipeline

Scalable Batch and Real-Time Data Architecture

Technical Report

Author: Miguel Mancilla

Role: Data Engineering Project

Date: July 2025

Technologies:

Google Cloud Platform · Pub/Sub · Dataflow · BigQuery · Apache Beam

1. Introducción

Este informe presenta el diseño y avance de un proyecto Big Data que integra dos fuentes de datos: archivos Parquet (batch) con registros de viajes de taxis amarillos de Nueva York (enero 2022 – diciembre 2024) y un flujo de eventos JSON (streaming) de subastas electrónicas. La solución propuesta cubre todo el ciclo de vida del dato: recolección, procesado, almacenamiento, análisis y consumo.

2. Justificación del uso de Big Data

El proyecto requiere una solución Big Data debido a la magnitud, complejidad y velocidad de los datos involucrados, lo cual hace inviable el uso de métodos tradicionales como bases de datos relacionales o scripts manuales de procesamiento.

El proyecto cumple con las 5 V del Big Data:

Volumen: Millones de registros mensuales en archivos Parquet durante 3 años.

Velocidad: Ingesta de eventos cada ~2 minutos en streaming.

Variedad: Dos formatos (Parquet y JSON) y atributos heterogéneos (temporales, geográficos, transaccionales).

Veracidad: Se aplican procesos de validación, limpieza y normalización para asegurar calidad.

Valor: Permite análisis de movilidad urbana y comportamiento de subastas, optimizando operaciones y prediciendo tendencias.

Por estas razones, Big Data no solo es conveniente, sino esencial. La diversidad de fuentes, el alto volumen y la necesidad de análisis tanto en tiempo real como histórico hacen inviable una solución tradicional basada únicamente en bases de datos relacionales o scripts de procesamiento secuencial.

3. Objetivos técnicos del Proyecto

- Diseñar una arquitectura unificada que soporte procesamiento batch y streaming.
- Seleccionar herramientas gestionadas en Google Cloud Platform (GCP) para minimizar operación.
- Automatizar pipelines de ingestión, transformación y carga a un data warehouse.
- Desplegar dashboards interactivos en Looker Studio.
- Asegurar gobernanza, seguridad y trazabilidad de los datos.

4. Gobierno de datos y ciclo de vida del dato

En este Proyecto el gobierno de datos consiste en buenas prácticas y políticas que garantizan la calidad, seguridad, disponibilidad y cumplimiento normativo a lo largo de todo el ciclo de vida del dato.

Políticas y prácticas de gestión de datos

Uso: Los datos son utilizados exclusivamente para análisis de comportamiento urbano y subastas electrónicas, respetando la finalidad para la cual fueron recolectados.

Acceso: Se gestionan mediante IAM de GCP, asegurando acceso granular basado en roles (principio de mínimo privilegio).

Retención: Se definen reglas de retención usando políticas de ciclo de vida en Cloud Storage y vencimiento automático de particiones en BigQuery (por ejemplo, retención de 10 años).

Calidad de los datos

- **Integridad:** Definición de reglas que los datos deben cumplir para ser validos: formatos obligatorios, campos que no pueden estar vacíos y relaciones lógicas entre los atributos.
- **Consistencia:** Se validan los esquemas y formatos desde la ingesta (JSON y Parquet) y se normalizan atributos para unificación de estructuras.
- **Compleitud:** Se monitorean registros faltantes o con valores nulos.
- **Exactitud:** Se estandarizan fechas y horas ajustándolas a un mismo formato evitando confusión al analizar.
- **Disponibilidad:** Los datos procesados se almacenan en BigQuery, lo que permite acceso inmediato a usuarios autorizados, sin interrupciones. Además, se cuenta con alta disponibilidad gracias a la infraestructura de GCP.
- **Redundancia mínima:** Se evita la duplicación de datos mediante claves únicas. Manteniendo la eficiencia del almacenamiento y la coherencia.

Seguridad

- **Protección contra acceso no autorizado**
 - **Registros de acceso y monitoreo:** Se utilizan herramientas como **Cloud Audit Logs** para registrar quién accede a los datos y cuándo. Esto permite detectar intentos sospechosos o accesos fuera de horario.
- **Cifrado en tránsito y en reposo:**
 - **En tránsito:** Los datos se transfieren usando el protocolo TLS que proporciona comunicación segura a través de la red.
 - **En reposo:** Los datos almacenados están cifrados automáticamente con claves gestionadas por GCP
- **Normativas y cumplimiento:**
 - Se garantizará que el uso de los datos de los taxis amarillos y los registros de las subastas cumplan plenamente con la Ley 19.628 de Chile, asegurando que la privacidad de los individuos se respete al manejar información sensible proporcionada
 - Asimismo, se documentará y analizará el cumplimiento de las regulaciones aplicables

Creación y obtención

- **Generación de datos:**

Los datos se generan desde dos flujos principales. Por una parte, los datos batch provienen de los viajes realizados por los conductores de taxis amarillos, registrando información como rutas, tarifas y horarios. Por otra parte, los datos en tiempo real (*streaming*) se originan desde subastas electrónicas que capturan eventos conforme ocurren en plataformas digitales.
- **Ingesta de datos:**

La ingestá se realiza mediante mecanismos batch (archivos Parquet) o en tiempo real (eventos JSON), utilizando herramientas como Cloud Storage y Pub/Sub, asegurando la captura sin pérdidas.
- **Clasificación inicial:**

Al momento de ingresar los datos, se clasifican según sensibilidad, criticidad y requisitos de acceso, con base en políticas de seguridad y gobernanza.

Almacenamiento

- **Estructurado:** Datos organizados en tablas y columnas almacenados en sistemas BigQuery
- **No estructurado:** Archivos en formato libre (JSON, imágenes, texto plano) gestionados mediante Cloud Storage.
- **Datos en frío y caliente:** Se distingue entre datos de acceso frecuente (calientes, almacenados en sistemas de alta disponibilidad como BigQuery) y datos de consulta eventual (fríos, almacenados en Cloud Storage o archivados).

Transformación

- **Limpieza:** Se eliminan duplicados, se corrigen errores y se rellenan valores nulos.
- **Integración:** Se consolidan datos de distintas fuentes para construir una vista unificada.
- **Enriquecimiento:** Se añaden variables derivadas o externas que aportan mayor valor al análisis.
- **ETL/ELT:** Se utilizan pipelines automatizados (Dataflow, Dataproc) para extraer, transformar y cargar datos hacia el data warehouse.

Uso y explotación

- **Análisis de datos:** Se realiza mediante consultas analíticas en BigQuery
- **Visualización:** A través de dashboards interactivos en Looker Studio, facilitando la toma de decisiones informadas.
- **Toma de decisiones:** La información generada permiten optimizar operaciones, predecir tendencias y establecer políticas de negocio basadas en datos.

Retención y eliminación

- **Políticas de retención:** Se aplican reglas automatizadas para conservar los datos por 10 años, según normativas legales y necesidades del negocio.
- **Archivado:** Datos históricos no consultados frecuentemente son enviados a almacenamiento de bajo costo.
- **Eliminación segura:** Cuando se supera el plazo de retención, los datos se eliminan cumpliendo estándares de seguridad para prevenir recuperación no autorizada.

5. Propuesta de arquitectura y herramientas

Arquitectura Lambda

La solución implementada sigue el patrón de arquitectura **Lambda**, que **combina procesamiento en tiempo real y por lotes** para obtener tanto resultados inmediatos como análisis históricos robustos.

- **Ingesta en tiempo real:** Utiliza servicios como **Cloud Pub/Sub** para capturar eventos conforme ocurren, por ejemplo, subastas en vivo.
- **Procesamiento en tiempo real:** Se apoya en **Dataflow** para realizar transformaciones y análisis de los datos a medida que llegan.
- **Procesamiento batch:** Implementa herramientas como **Dataproc** (o en este caso, se reutiliza Dataflow) para analizar grandes volúmenes de datos históricos almacenados.
- **Almacenamiento:** Los datos procesados en tiempo real se almacenan en **BigQuery**, mientras que los datos históricos procesados por lotes pueden mantenerse en **Cloud Storage** o **Bigtable**, según el caso.

¿Cuándo usar esta arquitectura?

Cuando se necesita obtener *insights* inmediatos para toma de decisiones en tiempo real, sin perder la capacidad de hacer análisis profundo sobre datos históricos.

Herramientas utilizadas en GCP

La arquitectura fue desarrollada íntegramente en **Google Cloud Platform**, seleccionando herramientas óptimas para cada etapa del flujo de datos:

1. Recolección de datos (Ingesta)

- **Streaming:** Se utiliza **Google Cloud Pub/Sub** para eventos en tiempo real generados desde las subastas electrónicas DUOC UC. La API que captura estos eventos corre en **Cloud Run**, publicando mensajes a Pub/Sub.
- **Batch:** Para datos históricos como los de “Taxis NYC” en formato **.parquet**, se emplea **Google Cloud Storage** como repositorio inicial.

Justificación:

Cloud Pub/Sub facilita la distribución y escalabilidad de eventos, ideal para escenarios de alta frecuencia. Por otro lado, Cloud Storage es el estándar para ingestá de archivos estructurados o semiestructurados en escenarios batch.

2. Procesamiento en tiempo real

- **Herramienta elegida: Google Dataflow**
- **Alternativa considerada:** Google Dataproc

Justificación:

Dataflow es un servicio completamente gestionado y autoescalable, lo que permite realizar transformaciones complejas y análisis de datos sin necesidad de gestionar clústeres.

3. Procesamiento batch

- **Herramienta utilizada: Google Dataflow**
- **Alternativa considerada:** Google Dataproc

Justificación:

Aunque Dataproc (con Spark o Hadoop) es muy útil para procesamiento batch, se decidió reutilizar Dataflow también para el procesamiento por lotes. Esto permite unificar el desarrollo y simplificar la arquitectura general.

4. Almacenamiento

- **Herramienta elegida: Google BigQuery**
- **Alternativa considerada:** Google Bigtable

Justificación:

BigQuery es ideal para análisis interactivo y exploración de grandes volúmenes de datos estructurados. Su rendimiento y costo-efectividad lo convierten en la elección natural para análisis posteriores en Looker Studio.

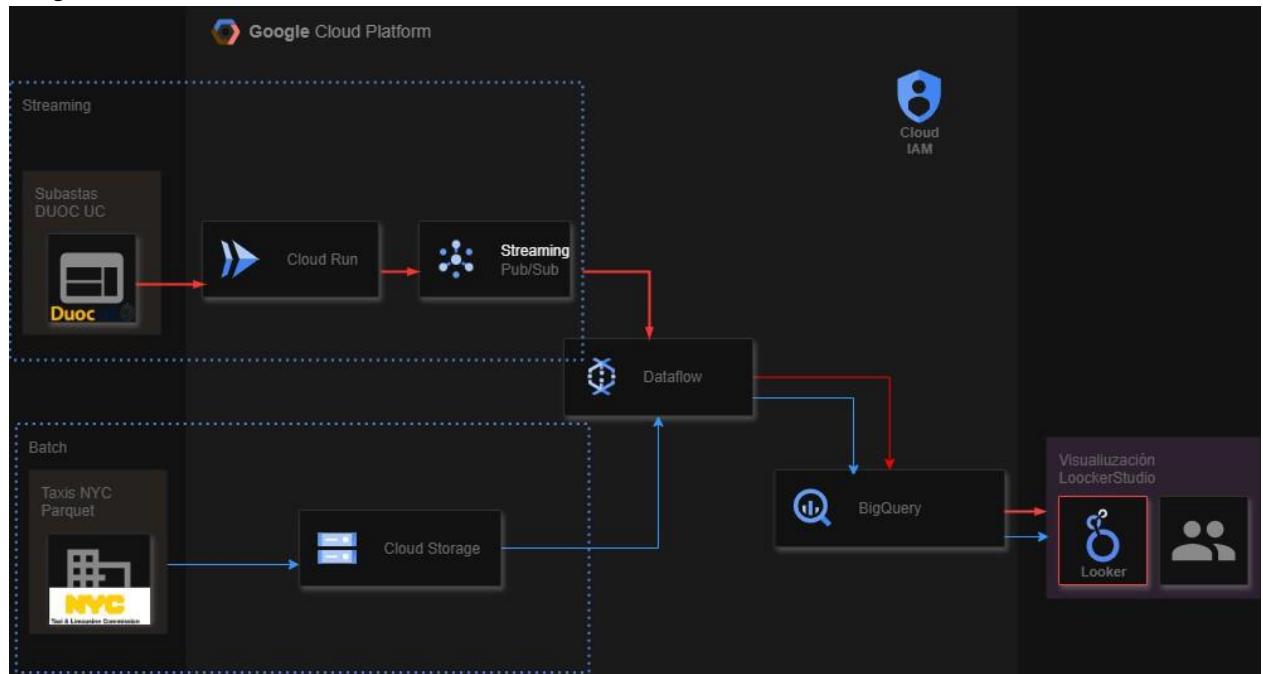
5. Visualización

- **Herramienta utilizada: Looker Studio**
- **Alternativas consideradas:** Power BI, Tableau, Grafana

Justificación:

Looker Studio permite construir dashboards personalizados e interactivos, directamente conectados a BigQuery, sin necesidad de herramientas intermedias. Es especialmente útil para entregar valor visual y analítico de forma rápida al equipo o stakeholders.

Diagrama Final



Implementación Bash

Creación e Importación de Datos

Para almacenar los datos de forma segura y accesible, se utilizó **Cloud Storage**, una herramienta que permite guardar grandes volúmenes de información en la nube. Esta estrategia es especialmente útil para proteger los datos en caso de que la fuente original sufra caídas o pérdida de información.

Configuración del bucket:

- Se definió un nombre identificador para el bucket, el cual permite acceder a los datos almacenados.
- Se seleccionó una región de almacenamiento que garantice disponibilidad y rapidez en las operaciones posteriores.

Los archivos en formato .parquet fueron organizados en carpetas dentro del bucket, con el fin de mantener un orden lógico que facilite su identificación y uso.

The screenshot shows the Google Cloud Storage interface for the project 'My First Project'. The left sidebar includes sections for 'Buckets', 'Supervisión', 'Configuración', 'Storage Intelligence', and 'Conjuntos de datos de es...'. The main content area has tabs for 'Descripción general' (selected), 'Guías', and 'Ver lista de buckets'. A welcome message says 'Te damos la bienvenida a Cloud Storage, Miguel'. It features a 'Crear bucket' button and links to 'Ir a una ruta específica' and 'Ver lista de buckets'. The 'Buckets fijados' section shows a list of recently viewed buckets: 'deep-bluoic-461003-m9-taxismarillos/debug_output/' (marked as 'Nuevo'), 'deep-bluoic-461003-m9-taxismarillos/temp_draflow/beamapp-contactocommandilla-060106130-665593...', 'deep-bluoic-461003-m9-taxismarillos/2024/', and 'deep-bluoic-461003-m9-taxismarillos/2023/'. The 'Visitados recientemente' section shows the same list. The 'Actividad de transferencia reciente' section indicates 'No hay actividad de transferencia reciente.' Below it is a link to 'Create a transfer job'. At the bottom, there are 'Funciones que podrían gustarte' (Recommended functions) cards for 'Prueba introductoria de 30 días de Storage Intelligence' and 'Realiza acciones de administración en millones de objetos a la vez con las operaciones por lotes'.

Creando el bucket se tiene que definir el nombre este nombre se usara para poder acceder a los datos, se necesita elegir una región para poder acceder a los datos posteriormente y no tener problemas al intentar comunicarse al solicitar la información. Click en crear bucket.

Crear un bucket

- Empezar**

Selecciona un nombre permanente globalmente único. [Lineamientos para asignar nombre](#)

Sugerencia: No incluyas información sensible
- Etiquetas (opcional)**

Continuar

- Elige dónde almacenar tus datos**

Ubicación: us (múltiples regiones en Estados Unidos)
Tipo de ubicación: Multi-region
- Elige cómo almacenar tus datos**

Clase de almacenamiento predeterminada: Standard
Espacio de nombres jerárquico: Inhabilitado
- Elige cómo controlar el acceso a los objetos**

Prevención del acceso público: Sí
Control de acceso: Uniforme
- Elige cómo proteger los datos de objeto**

Política de eliminación no definitiva: Predeterminada
Controles de versiones de objetos: Inhabilitado
Política de retención del bucket: Inhabilitado
Retención de objetos: Inhabilitado
Tipo de encriptación: Administrada por Google

Información útil

Precios de ubicación
Las tarifas de almacenamiento varían según la clase de almacenamiento de los datos y la ubicación de los buckets. [Detalles de las tarifas](#)

Configuración actual: Multi-region/Standard

Elemento	Costo
us (múltiples regiones en Estados Unidos)	\$0.026 por GB al mes
Con replicación predeterminada	\$0.020 por GB escrito

[Estimar costo mensual](#)

Cloud Storage | My First Project

Buckets

Filtro	Nombre	Fecha de creación	Tipo de ubicación	Ubicación	Clase de almacenamiento predeterminada	Última modificación	Acceso público	Control de acceso
<input type="checkbox"/>	[REDACTED]	1 jun 2025 06:09:51	Region	us-east1	Standard	1 jun 2025 06:09:51	Sujeto a LCA de objeto	Detalizado
<input type="checkbox"/>	[REDACTED]	25 may 2025 23:28:04	Region	us-east1	Standard	25 may 2025 23:28:04	No público	Uniforme

Marketplace

Notas de versión

Los archivos .parquet fueron descargados, organizados y cargados en carpetas específicas dentro del bucket en Cloud Storage. Esta estructura facilita la identificación de los archivos, mejora el orden del repositorio y optimiza el acceso a los datos durante las etapas de procesamiento. Seleccionamos el bucket y en el interior podemos crear carpetas y subir archivos.



The screenshot shows the Google Cloud Storage console. At the top, there are navigation links: 'Crear carpeta', 'Subir', 'Transferir los datos', and 'Otros servicios'. Below the header, there are two search bars: 'Filtrar solo por prefijo de nombre' and 'Filtro Filtrar objetos y carpetas'. A dropdown menu 'Mostrar' is set to 'Solo objetos activos'. The main area displays a table with columns: 'Nombre', 'Tamaño', 'Tipo', 'Fecha de creación', 'Clase de almacenamiento', 'Última modificación', and 'Acceso pùl'. Three empty folders are listed under 'Nombre': '2022/' (Carpeta), '2023/' (Carpeta), and '2024/' (Carpeta). Each folder has a small checkbox icon to its left. To the right of each folder name are three vertical dots, likely for more options.

Se utilizó **Cloud Shell**, un entorno de máquina virtual temporal que permite interactuar con los recursos de Google Cloud. Debido a su naturaleza transitoria, fue necesario descargar los archivos desde el bucket para trabajar localmente.

Como parte de las buenas prácticas, se creó una carpeta específica para almacenar los archivos necesarios:

```
# Crear el directorio del proyecto
mkdir my_dataflow_project
```

```
# Entrar al directorio
cd my_dataflow_project/
```

```
# Copiar el script de Python desde el bucket de Cloud Storage
gsutil cp gs://YOUR_PROJECT_ID-taxisamarillos/test_limpieza_taxis.py .
```

```
# Copiar el archivo de requerimientos
gsutil cp gs://YOUR_PROJECT_ID-taxisamarillos/requirements.txt .
```

Librerías necesarias:

1. **apache-beam[gcp]==2.57.0**
 - Permite crear y ejecutar pipelines de procesamiento de datos en GCP (usando Dataflow).
 - Incluye conectores para BigQuery, Cloud Storage, Pub/Sub, etc.
2. **pyarrow==16.0.0**
 - Permite leer archivos .parquet, un formato eficiente para datos tabulares.
 - Apache Beam no puede leer Parquet directamente sin pyarrow.

Para instalarlas se debe crear un archivo llamado requirements.txt y posterior ejecutar el siguiente comando.

```
gsutil cp gs://YOUR_PROJECT_ID -taxisamarillos/requirements.txt .
```



En la consola de cloud Shell esta la siguiente opción que permite editar los archivos y es más amigable para diseñar el código que requerimos para poder insertar en una tabla de un dataset en bigquery .

```
gsutil cp gs://YOUR_PROJECT_ID -taxisamarillos/test_limpieza_taxis.py .
```

```
pip install -r requirements.txt
```

Estas instrucciones permitieron instalar las dependencias necesarias y preparar el entorno para el procesamiento de los datos

[LINK CODIGO](#)

Explicación del código

Este script implementa un pipeline de procesamiento de datos utilizando Apache Beam. Su propósito principal es leer archivos en formato Parquet, transformar los datos limpiando y formateando los campos y luego cargarlos en una tabla de BigQuery. A continuación, se describe brevemente el funcionamiento de cada sección:

1. Importación de librerías

Se importan las bibliotecas necesarias, incluyendo apache_beam para definir el pipeline, argparse para manejar argumentos desde la línea de comandos y datetime para el manejo de fechas.

2. Clase DataIngestion y método parse_method

Esta clase define el método que limpia y transforma cada fila del archivo Parquet. La función realiza las siguientes tareas:

Verifica el tipo de cada valor y lo convierte si es necesario.

Elimina valores nulos, vacíos o cadenas como "null".

Convierte fechas a formato ISO 8601 (por ejemplo, 2025-07-14T10:00:00).

Estandariza los datos para que sean compatibles con el esquema de destino en BigQuery.

3. Función run()

Esta función principal define y ejecuta el pipeline. Las etapas incluidas son:

Lectura de argumentos: Recibe las rutas del archivo de entrada (--input) y del destino en BigQuery (--output).

Inicialización del pipeline: Se configura el entorno de ejecución (local o en la nube) mediante PipelineOptions.

Definición del flujo de datos:

Lectura del archivo Parquet mediante ReadFromParquet.

Transformación de cada fila usando parse_method.

Escritura de los datos procesados en BigQuery, con un esquema predefinido.

4. Ejecución del pipeline

El método p.run().wait_until_finish() inicia la ejecución del pipeline y espera a que finalice.

Utilidad del código

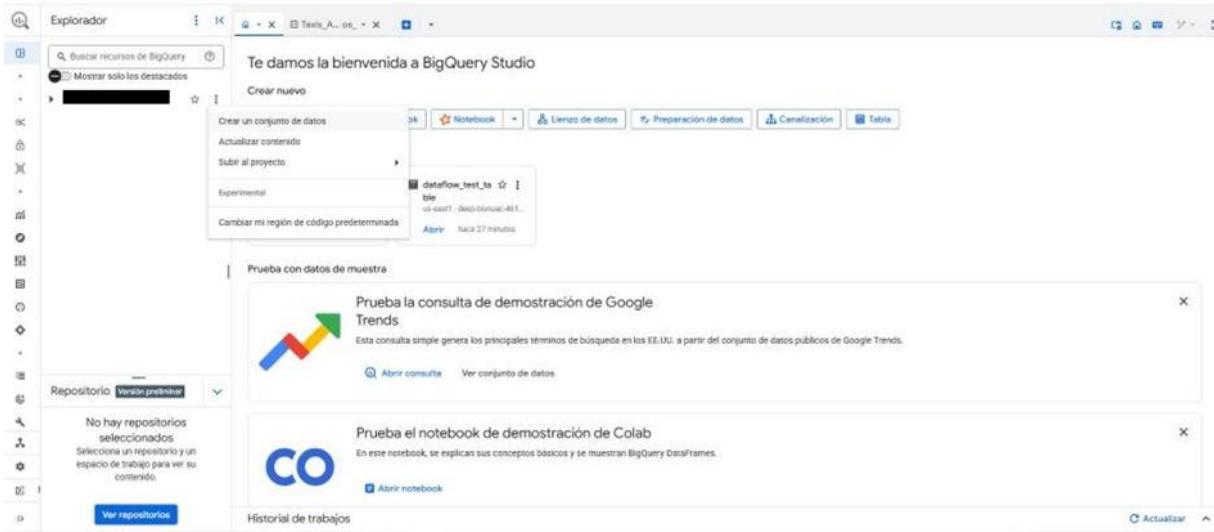
Este pipeline permite automatizar un proceso de extracción, transformación y carga (ETL) de datos estructurados. Es especialmente útil en contextos donde se requiere:

Procesar grandes volúmenes de información de manera eficiente.

Garantizar la calidad de los datos antes de su análisis.

Integrar datos desde Cloud Storage hacia BigQuery de forma programada y escalable.

Nota: El **dataset en BigQuery debe ser creado previamente** para evitar errores durante la ejecución del job. Además, es necesario habilitar previamente las **APIs de Dataflow, BigQuery y Cloud Storage**, al igual que con el resto de herramientas utilizadas en este proyecto.



Crea un conjunto de datos

ID del proyecto *

Cambio

ID de conjunto de datos *

Puede incluir letras, números y guiones bajos

Tipo de ubicación [?](#)

Región

Especifica una región para colocar tus conjuntos de datos con otros servicios de Google Cloud.

Multirregional

Permite que BigQuery seleccione una región de un grupo para alcanzar límites de cuota más altos.

Multirregional *

US (múltiples regiones en Estados Unidos)



Conjunto de datos externo

The selected region supports the following external dataset types: Cloud Spanner

Vínculo a un conjunto de datos externo [?](#)

Etiquetas



Opciones avanzadas



[Crear conjunto de datos](#)

[Cancelar](#)

Nombre del conjunto “DatosBash” , no se realizó modificaciones adicionales.

Ejecución de pipelines de Apache Beam para la ingestión de datos

Los comandos proporcionados permiten ejecutar el script pipeline.py, que contiene el pipeline de procesamiento previamente explicado, con el objetivo de **cargar archivos Parquet a BigQuery**. Cada comando corresponde a un año distinto (2022, 2023 y 2024).

Comando general (formato base):

```
python3 pipeline.py \
--input [ruta_de_archivos_Parquet] \
--output [tabla_de_BigQuery] \
--temp_location [ruta_de_archivos_temporales] \
--project [ID_del_proyecto_de_GCP] \
--region [región_de_Dataflow] \
--runner DataflowRunner \
--staging_location [ruta_para_stage_de_ejecución] \
--save_main_session
```

python3 pipeline.py	Ejecuta el script Python que contiene el pipeline.
--input	Ruta en Cloud Storage donde se encuentran los archivos .parquet a procesar. El comodín * permite leer múltiples archivos del año correspondiente.
--output	Tabla de destino en BigQuery donde se cargarán los datos transformados. El formato es proyecto:dataset.tabla.
--temp_location	Ubicación en Cloud Storage donde Dataflow almacenará archivos temporales durante la ejecución
--project	ID del proyecto de Google Cloud donde se ejecutará el pipeline.
--region	Región de GCP donde se ejecutará el servicio de Dataflow (por ejemplo, us-central1).
--runner DataflowRunner	Indica que el pipeline se ejecutará en el entorno gestionado de Dataflow (no localmente).
--staging_location	Carpeta donde Dataflow subirá archivos intermedios y dependencias necesarias para ejecutar el pipeline.
--save_main_session	Permite que Dataflow mantenga el contexto de la sesión principal de Python, útil para evitar errores con funciones o clases definidas fuera del main.

Ejecuciones específicas

Ingesta del año 2024:

```
python3 pipeline.py \
--input gs://bucketdatarealduoc/2024/yellow_tripdata_2024-*.parquet \
--output your-project:DatosBash.YellowTaxis \
...
```

Ingesta del año 2023:

```
python3 pipeline.py \
--input gs://bucketdatarealduoc/2023/yellow_tripdata_2023-*.parquet \
--output your-project -m8:DatosBash.YellowTaxis \
...
```

Ingesta del año 2022:

```
python3 pipeline.py \
--input gs://bucketdatarealduoc/2022/yellow_tripdata_2022-*.parquet \
--output your-project:DatosBash.YellowTaxis \
...
```

Cada uno de estos comandos ejecuta el mismo pipeline, pero con los archivos correspondientes al año especificado.

Al ejecutar el script pipeline.py mediante cualquiera de los comandos proporcionados, se crea un job en el servicio de Dataflow de Google Cloud. Este job pasa automáticamente por los siguientes estados:

- Creación (Created): Se configura el entorno de ejecución.
- En ejecución (Running): El pipeline comienza a procesar los datos.
- Finalizado con éxito (Succeeded): El procesamiento y la carga en BigQuery terminan sin errores.

El tiempo total que tarda el pipeline en completarse depende directamente de la cantidad de datos que se estén procesando. Por ejemplo, el procesamiento de datos correspondientes a un año completo (como los archivos Parquet de 2022, 2023 o 2024) puede tardar varios minutos o incluso horas, dependiendo del volumen de registros y la disponibilidad de recursos asignados por Dataflow.

Además, dado que el pipeline está configurado con las opciones CREATE_IF_NEEDED y WRITE_APPEND, la tabla de BigQuery se crea automáticamente si no existe y los datos se agregan a los existentes, evitando sobrescrituras. Sin embargo, el dataset (DatosBash) debe estar creado previamente en BigQuery antes de ejecutar el pipeline.

071	Batch	12 Jul 2025, 16:17:13	1 hr 24 min	12 Jul 2025, 14:52:29	Succeeded	2.65.0	2025-07-12_11_52_29-4947255078587379897	us-central1
071	Batch	12 Jul 2025, 14:42:21	30 min 59 sec	12 Jul 2025, 14:11:22	Succeeded	2.65.0	2025-07-12_11_11_21-16748275555242360716	us-central1
065	Batch	30 Jun 2025, 11:54:07	33 min 15 sec	30 Jun 2025, 11:20:52	Succeeded	2.65.0	2025-06-30_08_20_51-11693855853369280584	us-central1

Una vez finalizada la ejecución del pipeline, es posible validar que los datos han sido cargados correctamente en BigQuery realizando una consulta SQL. Para ello, se puede utilizar la instrucción: `SELECT * FROM `your-project.DatosBash.YellowTaxis``

Query results								
Job information		Results	Chart	JSON	Execution details	Execution graph		
Row	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID
1	2	2024-09-29 22:06:40 UTC	2024-09-29 22:29:40 UTC	1.0	15.97	1.0	N	138
2	2	2022-03-29 03:06:37 UTC	2022-03-29 03:27:06 UTC	2.0	8.2	1.0	N	230
3	1	2022-03-28 15:49:33 UTC	2022-03-28 17:01:44 UTC	3.0	19.5	1.0	N	132
4	2	2023-04-29 16:34:24 UTC	2023-04-29 17:02:38 UTC	1.0	13.03	1.0	N	132
5	2	2024-05-13 22:49:29 UTC	2024-05-13 23:20:29 UTC	1.0	7.38	1.0	N	170
6	2	2022-12-12 09:16:16 UTC	2022-12-12 09:52:40 UTC	1.0	9.53	1.0	N	138

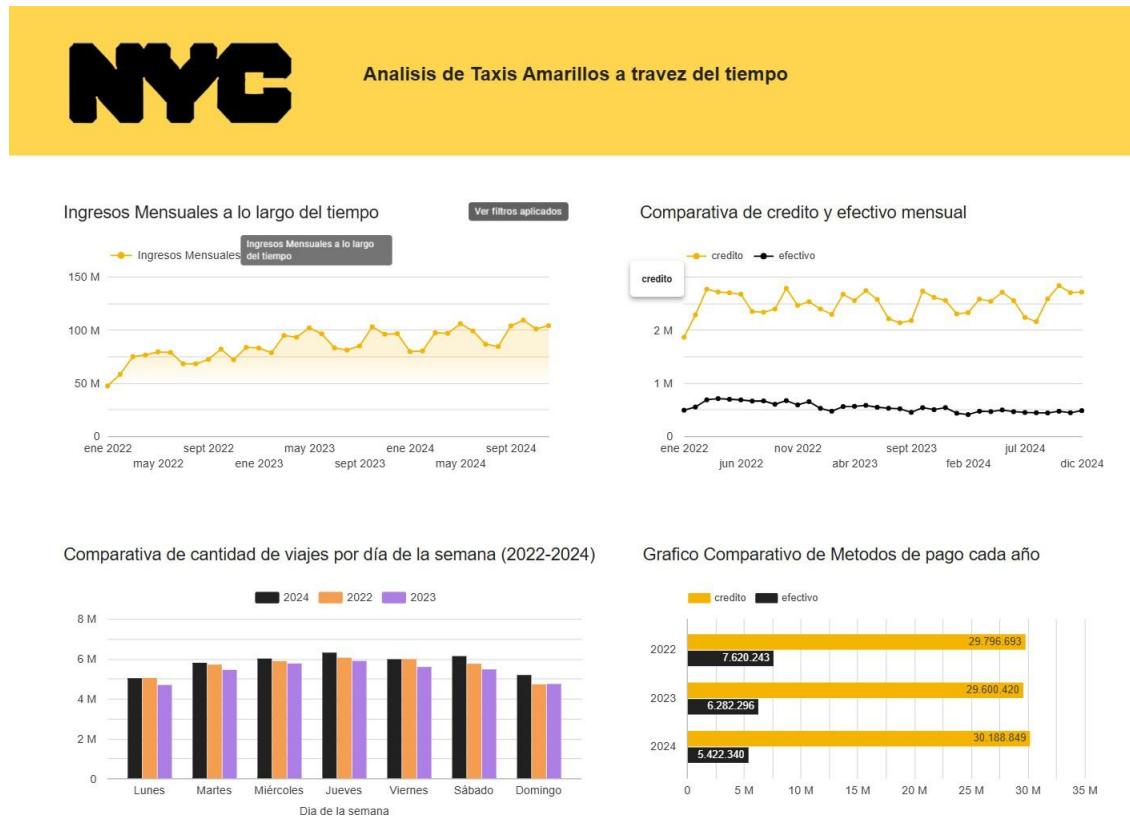
Results per page: 50 ▾ 1 – 50 of 119136044 | < < > >|

Esta consulta mostrará todos los registros almacenados en la tabla YellowTaxis del dataset DatosBash, permitiendo verificar visualmente que los datos han sido procesados y cargados exitosamente desde los archivos Parquet originales.

Visualización de los datos en Looker Studio

Con el objetivo de presentar de forma clara y comprensible los resultados del análisis, se desarrolló un dashboard en Looker Studio conectado directamente a BigQuery. Este panel visual resume los principales hallazgos del análisis de los viajes de taxis amarillos en la ciudad de Nueva York entre los años 2022 y 2024.

[Dashboard NYC taxis amarillos](#)



Descripción de los gráficos:

Ingresos Mensuales a lo largo del tiempo

Este gráfico de línea muestra la evolución de los ingresos generados mes a mes desde enero de 2022 hasta diciembre de 2024. Se observa un crecimiento progresivo con ciertas fluctuaciones estacionales, lo que permite identificar meses con mayor actividad económica.

Detalle de construcción del gráfico: *Ingresos mensuales a lo largo del tiempo*

Para elaborar este gráfico de líneas en Looker Studio, se utilizó una conexión directa con la tabla YellowTaxis almacenada en BigQuery. El objetivo fue representar la evolución de los ingresos generados por los viajes de taxis amarillos en un periodo de tres años.

- **Dimensión utilizada:** tpep_pickup_datetime transformada al formato **Año y Mes**, lo que permite agrupar los datos mensualmente.
- **Métrica utilizada:** **Ingresos mensuales**, definidos como la suma del campo total_amount, el cual representa el valor total de cada viaje (incluyendo tarifa, propina, peajes, etc.).
- **Rango de fechas:** Se definió un **periodo personalizado** desde el **1 de enero de 2022 hasta el 31 de diciembre de 2024** para cubrir los tres años analizados.

Esta configuración permitió generar una visualización clara del comportamiento de los ingresos en el tiempo, facilitando la detección de patrones estacionales, aumentos progresivos y posibles anomalías en ciertos meses.

Comparativa de crédito y efectivo mensual

También representado mediante líneas, este gráfico compara el uso de los métodos de pago crédito y efectivo a lo largo del tiempo. Se evidencia una clara preferencia por el pago con tarjeta de crédito, cuya tendencia se mantiene estable o en aumento, en contraste con el pago en efectivo, que se mantiene bajo y relativamente constante.

Detalle de construcción del gráfico: Comparativa de crédito y efectivo mensual

Este gráfico de líneas tiene como objetivo comparar mensualmente la cantidad de viajes pagados con tarjeta de crédito y con efectivo durante el periodo 2022–2024. Para su construcción en Looker Studio, se utilizaron los siguientes elementos:

Dimensión utilizada: tpep_pickup_datetime transformada al formato Año y Mes, lo que permite agrupar los viajes por mes calendario.

Métricas utilizadas:

Crédito: campo calculado que contabiliza solo aquellos viajes realizados (`trip_distance > 0`) y pagados con tarjeta de crédito (`payment_type = 1`).

Efectivo: campo calculado que contabiliza los viajes realizados y pagados en efectivo (`payment_type = 2`).

Para crear estas métricas personalizadas en Looker Studio, se utilizaron expresiones CASE que filtran y cuentan los viajes válidos según el método de pago:

Crédito:

CASE

WHEN `payment_type = 1 AND trip_distance > 0` THEN 1

ELSE 0

END

Efectivo:

CASE

WHEN payment_type = 2 AND trip_distance > 0 THEN 1

ELSE 0

END

Rango de fechas (Dimensión de periodo): Se utilizó el campo tpep_pickup_datetime como fecha base para establecer un periodo desde enero de 2022 hasta diciembre de 2024.

Gracias a esta configuración, el gráfico permite visualizar la evolución y preferencia de los usuarios por cada método de pago a lo largo del tiempo, revelando patrones de comportamiento mensuales.

Comparativa de cantidad de viajes por día de la semana (2022–2024)

En este gráfico de barras verticales se muestra la distribución de viajes según el día de la semana, comparando los tres años analizados. Los datos indican que los días laborales (especialmente martes, miércoles y jueves) concentran una mayor cantidad de viajes en comparación con los fines de semana.

Detalle de construcción del gráfico: Comparativa de cantidad de viajes por día de la semana (2022–2024)

Este gráfico de barras verticales permite analizar el comportamiento de los viajes de taxis amarillos según el día de la semana, comparando los años 2022, 2023 y 2024.

Para construir esta visualización en Looker Studio, se utilizó como fuente una vista guardada en BigQuery bajo el nombre ViajesPorDiaDeLaSemana, la cual fue creada a partir de la siguiente consulta SQL:

```
SELECT  
    EXTRACT(YEAR FROM tpep_pickup_datetime) AS anio,  
    CASE EXTRACT(DAYOFWEEK FROM tpep_pickup_datetime)  
        WHEN 1 THEN 'Domingo'  
        WHEN 2 THEN 'Lunes'  
        WHEN 3 THEN 'Martes'  
        WHEN 4 THEN 'Miércoles'  
        WHEN 5 THEN 'Jueves'  
        WHEN 6 THEN 'Viernes'  
        WHEN 7 THEN 'Sábado'  
    END AS dia_semana,  
    CASE EXTRACT(DAYOFWEEK FROM tpep_pickup_datetime)  
        WHEN 1 THEN 7  
        WHEN 2 THEN 1  
        WHEN 3 THEN 2  
        WHEN 4 THEN 3  
        WHEN 5 THEN 4  
        WHEN 6 THEN 5  
        WHEN 7 THEN 6  
    END AS orden_semana,  
    COUNT(*) AS total_viajes  
FROM  
    `your-project.DatosBash.YellowTaxis`  
WHERE  
    trip_distance > 0
```

AND EXTRACT(YEAR FROM tpep_pickup_datetime) BETWEEN 2022 AND 2024

GROUP BY

anio, dia_semana, orden_semana

ORDER BY

anio, orden_semana;

Detalles de configuración en Looker Studio:

- Dimensión principal: dia_semana (ordenada usando el campo orden_semana para respetar la secuencia tradicional de lunes a domingo).
- Métrica: Suma de total_viajes, correspondiente al total de viajes registrados por día.
- Dimensión de periodo: anio, lo que permite comparar visualmente los tres años de forma paralela mediante colores diferenciados.

Esta visualización permite identificar qué días concentran mayor cantidad de viajes a lo largo de la semana, y cómo ese patrón se ha mantenido o ha cambiado entre los distintos años analizados.

Gráfico Comparativo de Métodos de Pago cada Año

Se presenta un gráfico de barras horizontales que muestra el total de viajes pagados con crédito y efectivo en cada año. La visualización permite comparar fácilmente cómo ha evolucionado la preferencia de los usuarios por cada método de pago. Se aprecia que el uso de crédito ha aumentado ligeramente con los años, mientras que el efectivo ha ido disminuyendo.

Detalle de construcción del gráfico: Gráfico Comparativo de Métodos de Pago cada Año

Este gráfico de barras horizontales permite comparar el uso de los métodos de pago crédito y efectivo en los servicios de taxis amarillos, diferenciando los valores totales por año (2022, 2023 y 2024). Su objetivo es mostrar cómo ha evolucionado la preferencia de los pasajeros respecto a la forma de pago durante el periodo analizado.

Configuración utilizada en Looker Studio:

Dimensión principal: año, extraída desde el campo tpep_pickup_datetime.

Métricas: se utilizaron dos campos calculados:

Crédito:

CASE

WHEN payment_type = 1 AND trip_distance > 0 THEN 1

ELSE 0

END

Efectivo:

CASE

WHEN payment_type = 2 AND trip_distance > 0 THEN 1

ELSE 0

END

Estas métricas contabilizan únicamente los viajes efectivamente realizados ($\text{trip_distance} > 0$), separados por el tipo de pago registrado (payment_type = 1 para crédito y payment_type = 2 para efectivo).

Dimensión de periodo: se utilizó año, tanto para segmentar los datos como para establecer el filtro de tiempo aplicado.

Filtro de periodo personalizado: se definió un rango desde el 1 de enero de 2022 hasta el 31 de diciembre de 2024, lo que asegura que se incluyan únicamente los registros pertenecientes al periodo en análisis.

Este gráfico facilita la comparación anual de los volúmenes de viajes por tipo de pago y refuerza las conclusiones sobre las tendencias de uso en los diferentes años.

Etapa Streaming

Creación del tópico en Pub/Sub:

Se accede al servicio Google Cloud Pub/Sub y se crea un tópico denominado registros mediante la opción “Create Topic”. Este tópico se utilizará como canal principal para la recepción de los datos provenientes del sistema de subastas en tiempo real.

[←](#) Create topic

Topic ID * [?](#)

Topic name: projects/qwiklabs-gcp-01-a612354f0d1a/topics/registros

Add a default subscription [?](#)

Use a schema [?](#)

Enable ingestion [?](#)

Enable message retention [?](#)

Export message data to BigQuery [?](#)

Backup message data to Cloud Storage [?](#)

Transforms

Optionally manipulate and filter message data

[ADD A TRANSFORM](#)

Encryption

Google-managed encryption key
Keys owned by Google

Cloud KMS key
Keys owned by customers

Implementación del endpoint de recepción en Cloud Run:

Se desarrolla una función en Google Cloud Run que actúa como endpoint para recibir las solicitudes HTTP del sistema de subastas y enviarlas al tópico de Pub/Sub.

Pasos realizados:

1. Ingreso a Cloud Run > Create Service > Write a Function.

The screenshot shows the Google Cloud Platform dashboard. At the top left, there's a navigation bar with a 'Google Cloud' logo and a search bar. Below the navigation bar, a sidebar titled 'Products' lists various Google services: Billing, IAM & Admin, Marketplace, APIs & Services, Compute Engine, Kubernetes Engine, Cloud Storage, Security, BigQuery, Monitoring, Cloud Run, and VPC Network. The 'Cloud Run' item is highlighted with a red box. At the bottom of the sidebar is a 'View all products' button. The main content area has tabs for 'Services', '+ Deploy container', 'Connect repo', and a red-boxed '(-) Write a function'. Below these tabs, a message states: 'A service exposes a unique endpoint and automatically scales the underlying infrastructure to handle incoming requests. Deploy a container image, source code or a function to create a service.' There's also a 'Filter' section with dropdowns for 'Name' (set to '↑') and 'Deployment type', and buttons for 'Req/sec', 'Region', 'Authentication', and 'Ingress'.

2. Asignación de nombre al servicio (ejemplo: webhookpubsub) y selección de la región.
3. Configuración de entorno en Python 3.10.

Configure

Service name *

i Some locations have been restricted due to a policy set by your organization. [Learn more about restricting locations](#).

Region * ▼

[How to pick a region?](#)

Endpoint URL ? i

Runtime * ▼

4. Habilitación de autenticación y definición de solicitudes.

Authentication *

Use IAM to authenticate incoming requests
All invocations of this service's endpoint will be authorized by IAM.

- Allow unauthenticated invocations
Configures an IAM policy that allows access without a credential.
- Require authentication
Manage authorized users with IAM.

Ingress ?

- Internal
Allow traffic from your project, shared VPC, and VPC service controls perimeter. Traffic from another Cloud Run service must be routed through a VPC. Limitations apply. [Learn more](#)
- Allow traffic from external Application Load Balancers
- All
Allow direct access to your service from the internet.

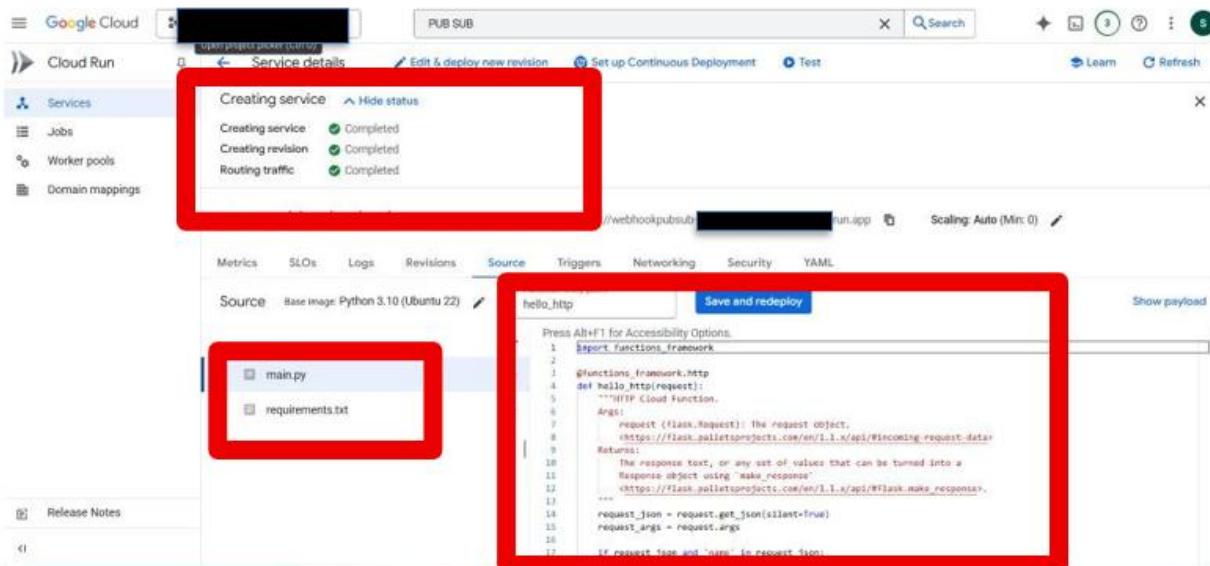
Requests

- Request timeout — seconds
Time within which a response must be returned (maximum 3600 seconds).
- Maximum concurrent requests per instance —
The maximum number of concurrent requests that can reach each instance. [What is concurrency?](#)

Execution environment
The execution environment your container runs in. [Learn More](#)

- Default
Cloud Run will select a suitable execution environment for you.
- First generation
Faster cold starts.
- Second generation
Network file system support, full Linux compatibility, faster CPU and network performance.

5. Configuración de la función editamos el código que viene por defecto con el código facilitado en el **informe de descarga de data streaming** para realizar las solicitudes.
 - Definición del punto de entrada de la función como main.
 - Incorporación de un archivo requirements.txt con las dependencias necesarias.
 - Ingreso de código que permite recibir datos JSON y publicarlos en Pub/Sub.



Explicación del código para publicación en Pub/Sub (Cloud Run)

Importación de librerías

Se importan las librerías necesarias:

google-cloud-pubsub para publicar mensajes en Google Pub/Sub.

functions-framework para crear la función HTTP (si usas Cloud Functions o Cloud Run).

google-cloud-bigquery para futuras interacciones con BigQuery (aunque no se usa directamente en el código actual).

json para manipular datos JSON.

Inicialización del cliente Pub/Sub

Se crea una instancia del cliente PublisherClient() para conectarse al servicio Pub/Sub.

Definición de proyecto y tópico

Se asignan variables para:

El ID del proyecto de Google Cloud.

El ID del tópico de Pub/Sub donde se enviarán los mensajes.

Se construye la ruta completa del tópico usando publisher.topic_path.

Función principal main(request)

Es la función que procesa las solicitudes HTTP entrantes.

Obtención y validación del JSON recibido

Dentro de la función, se extrae el JSON enviado en el cuerpo de la solicitud HTTP.

Si no existe o no es válido, retorna un error 400.

Publicación del mensaje en Pub/Sub

Si el JSON es una lista (varios objetos), se itera y se publica cada elemento como un mensaje independiente.

Si es un objeto individual, se publica directamente.

Conversión a bytes

Cada mensaje JSON se convierte a una cadena y luego a bytes, que es el formato que requiere Pub/Sub.

Publicación síncrona

Se espera la confirmación (`future.result()`) para asegurar que el mensaje se publicó correctamente antes de continuar.

Respuesta HTTP exitosa

Si todo sale bien, retorna el mensaje "Completado" con código 200.

Manejo de errores

Si ocurre alguna excepción durante el proceso, se captura y devuelve un mensaje de error con código 500.

Requerimientos (requirements.txt)

Para que este código funcione, debes instalar estas librerías en tu entorno:

```
google-cloud-pubsub  
functions-framework  
google-cloud-bigquery
```

6. Permitir invocaciones no autenticadas, en la consola Shell de cloud

```
[in this session is set to [REDACTED]]  
Use `gcloud config set project [PROJECT_ID]` to change to a different project.  
$ gcloud run services add-iam-policy-binding webhookpubsub \  
--region=us-central1 \  
--member="allUsers" \  
--role="roles/run.invoker"  
Updated IAM policy for service [webhookpubsub].  
bindings:  
- members:  
  - allUsers  
    role: roles/run.invoker  
etag: BwY4pazVRxA=  
version: 1  
[REDACTED]
```

7. Asignación del rol “Pub/Sub-Publisher” a la cuenta de servicio.

The screenshot shows the Google Cloud IAM interface. On the left, a sidebar lists various IAM-related options like PAM, Principal Access Boundaries, Organizations, and Policy Troubleshooter. The main area is titled 'Permissions for project [REDACTED]'. It shows two entries under 'Grant access': one for 'compute@developer.gserviceaccount.com' with the role 'Editor' and another for 'Qwiklabs User Service' with the role 'Owner'. A red box highlights the edit icon for the 'Editor' role entry. Below this, there's a section titled 'Edit access to "compute@developer.gserviceaccount.com"' where a 'Pub/Sub Publisher' role is assigned to the project.

8. Volver a lanzar para cargar con los nuevos datos.

The screenshot shows the Cloud Run service details page for 'webhookpubsub'. The 'Source' tab is selected, showing the code: 'Base Image: Python 3.10 (Ubuntu 22)' and 'Function entry point: main'. Below the code is a 'Save and redeploy' button, which is highlighted with a red box. Other tabs include Metrics, SLOs, Logs, Revisions, Triggers, Networking, Security, and YAML.

Recordar en cada paso que se solicite ejecutar o guardar aplicar el botón azul, para no tener problemas con la ejecución ya que no guardara los cambios o simplemente quedaran sin haberse ejecutados

The screenshot shows the Google Cloud Run 'Service details' page for a service named 'webhookpubsub'. The top navigation bar includes 'Cloud Run', 'Search', 'Edit & deploy new revision', 'Set up Continuous Deployment', 'Test', 'Learn', and 'Refresh'. On the left, a sidebar lists 'Services', 'Jobs', 'Worker pools', and 'Domain mappings'. The main content area has tabs for 'Metrics', 'SLOs', 'Logs', 'Revisions', 'Source', 'Triggers', 'Networking', 'Security', and 'YAML'. The 'Source' tab is selected, showing a base image of 'Python 3.10 (Ubuntu 22.04)' and a function entry point of 'main'. A red box highlights the 'Edit source' button. Below it, a red box highlights the file structure showing 'main.py' and 'requirements.txt'. Another red box highlights the code editor window containing Python code for publishing to Pub/Sub. The code uses the `google.cloud` library to import `pubsub_v1` and handle JSON data from a request to publish to a topic.

```
1 from google.cloud import pubsub_v1
2 import json
3 # Inicializa el cliente de Pub/Sub una vez
4 publisher = pubsub_v1.PublisherClient()
5 project_id = "proyecto" # MODIFICAR DE ACUERDO AL PROYECTO QUE USEN EN GCP o BSB
6 topic_id = "registro"
7 topic_path = publisher.topic_path(project_id, topic_id)
8 def main(request):
9     try:
10         # Obtener el JSON desde el body del request
11         data = request.get_json()
12         if not data:
13             return "Solicitud sin JSON válido", 400
14         # Convertir a bytes para Pub/Sub
15         message_bytes = json.dumps(data).encode("utf-8")
16         # Publicar mensaje en Pub/Sub
```

9. Despliegue de la función y obtención de la URL para el sistema de subastas, con la cuenta facilitada se ingresa la url generada por la función y mostrara la lista que confirma como “enviado” cada solicitud en caso de tener algún problema indicara “error”

Etapa 2: Almacenamiento de datos procesados

Herramienta utilizada:

Google BigQuery

Cloud Storage (Bucket, carpeta temporal)

Descripción y procedimiento:

Se crea un dataset denominado DatosRealTime y una tabla llamada DatosTR con el siguiente esquema:

- id_cliente (STRING)
- cliente (STRING)
- genero (STRING)
- id_producto (STRING)
- producto (STRING)
- precio (FLOAT)
- cantidad (INTEGER)
- monto (FLOAT)
- forma_pago (STRING)
- fecreg (TIMESTAMP)

Los datos procesados se insertan en tiempo real en esta tabla, permitiendo su consulta inmediata para análisis posteriores.

Dimensiones de los datos:

- Registros con 10 campos relevantes.

Estrategia de recolección de datos

Para recopilar la información necesaria, utilicé una cuenta gratuita en Google Cloud, lo que me permitió disponer de un entorno con mayor capacidad y flexibilidad para la recolección y almacenamiento de datos. Esta estrategia fue fundamental porque, durante los laboratorios, el margen para obtener datos era limitado.

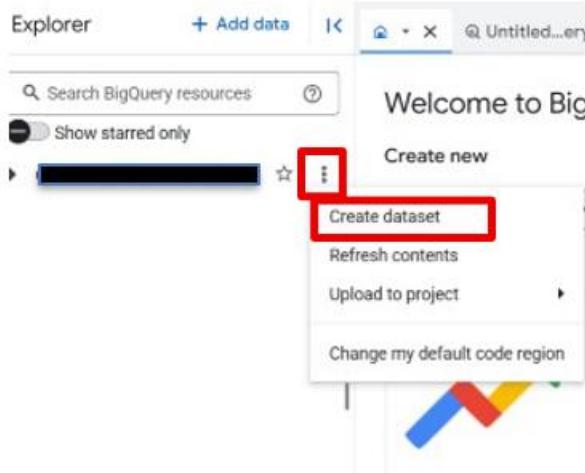
Mientras avanzaba en el desarrollo de las demás partes del informe, simultáneamente generaba y acumulaba datos hasta la fecha de término establecida, optimizando así el tiempo y los recursos disponibles.

Volumen de datos recolectados

- **Cantidad de registros:** 39,107
- **Espacio de almacenamiento ocupado:** 2.78 MB
- **Periodo de recolección:** Desde el 29-06-2025 hasta el 12-07-2025

Este volumen de datos fue suficiente para realizar un análisis representativo y robusto en el contexto del proyecto.

Pasos que realizar para poder crear nuestro almacenamiento en bigquery



- 1- Creamos un dataset con la siguiente configuración estándar, para evitar algún problema con la ubicación utilizaremos multi-region en caso de tener todo en la misma región se puede utilizar la región específica.

Create dataset

Project ID * ██ Change

Dataset ID * DatosRealTime Change

Letters, numbers, and underscores allowed

Location type ?

Region
Specify a region to colocate your datasets with other Google Cloud services.

Multi-region
Allow BigQuery to select a region within a group to achieve higher quota limits.

Info Some locations have been restricted due to a policy set by your organization. [Learn more about restricting locations.](#) ?

Multi-region * US (multiple regions in United States) ▼

External Dataset

The selected region supports the following external dataset types: Cloud Spanner

Link to an external dataset ?

Tags ▼

Create dataset Cancel

2- Creamos una tabla con la siguiente estructura y poder recibir los datos json.

The screenshot shows the BigQuery interface with a context menu open over a dataset named 'DatosRealTime'. The 'Create table' option is highlighted with a red box. Below the menu, a 'Create table' dialog is displayed:

- Source**:
 - Create table from
 - Empty table
- Destination**:
 - Project * [redacted]
 - Dataset * DatosRealTime
 - Table * DatosTR
- Schema**:
 - Edit as text (highlighted with a red box)
 - Press Alt+F1 for Accessibility Options.

At the bottom of the dialog are 'Create table' and 'Cancel' buttons.

- Hay que elegir la opción Edit as text para poder insertar la siguiente estructura

```
[  
  { "name": "id_cliente", "type": "STRING", "mode": "NULLABLE" },  
  { "name": "cliente", "type": "STRING", "mode": "NULLABLE" },  
  { "name": "genero", "type": "STRING", "mode": "NULLABLE" },  
  { "name": "id_producto", "type": "STRING", "mode": "NULLABLE" },  
  { "name": "producto", "type": "STRING", "mode": "NULLABLE" },  
  { "name": "precio", "type": "FLOAT", "mode": "NULLABLE" },  
  { "name": "cantidad", "type": "INTEGER", "mode": "NULLABLE" },  
  { "name": "monto", "type": "FLOAT", "mode": "NULLABLE" },  
  { "name": "forma_pago", "type": "STRING", "mode": "NULLABLE" },  
  { "name": "fecreg", "type": "TIMESTAMP", "mode": "NULLABLE" }  
]
```

- Creamos la tabla revisamos si se guardaron los cambios verificando que dentro del dataset se allá creado la tabla.

Dimensiones de los datos procesados

Los datos que se almacenan en BigQuery provienen del sistema de subastas en tiempo real, y cada registro corresponde a una transacción individual. El esquema definido en la tabla DatosTR contempla **10 campos** que permiten describir completamente cada operación de venta. A continuación, se describen las dimensiones principales:

- **Volumen de datos:** los datos se insertan de manera continua, lo que implica que el volumen puede crecer en función del número de subastas activas durante el periodo de observación.
- **Número de atributos por registro:** cada registro cuenta con 10 atributos estructurados, que incluyen información del cliente, del producto, de la transacción y de la forma de pago.
- **Estructura de los datos:**
 - id_cliente (STRING)
 - cliente (STRING)
 - genero (STRING)
 - id_producto (STRING)
 - producto (STRING)
 - precio (FLOAT)
 - cantidad (INTEGER)
 - monto (FLOAT)
 - forma_pago (STRING)
 - fecreg (TIMESTAMP)
- **Crecimiento dinámico:** la tabla está diseñada para escalar automáticamente en función del número de registros, lo que permite mantener la performance sin pérdida de datos ni necesidad de intervención manual.
- **Calidad y completitud:** al usar un flujo automatizado con Dataflow, los datos llegan en tiempo real sin intervención manual, lo que mejora la integridad y reduce errores de carga.

Estas dimensiones permiten un análisis eficiente y flexible, facilitando la posterior visualización en dashboards con herramientas como Looker Studio.

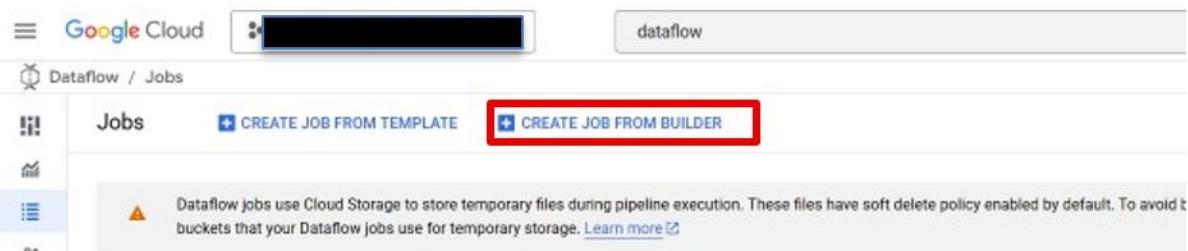
Etapa 3: Procesamiento de datos

Herramienta utilizada:

- Google Cloud Dataflow

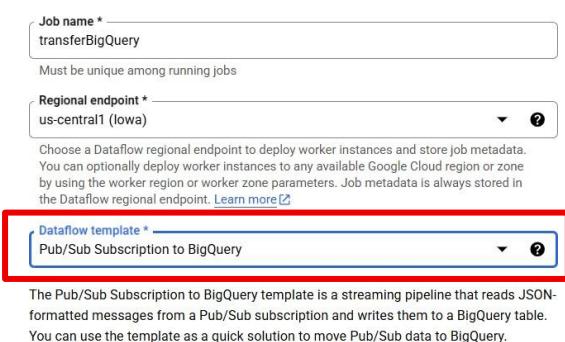
Descripción y procedimiento:

- Se utiliza Google Cloud Dataflow para procesar los datos mediante una plantilla predefinida que permite transferir datos desde Pub/Sub hacia BigQuery.
- Pasos realizados:
 1. Acceso a Dataflow y selección de “Create Job from Template”.



The screenshot shows the Google Cloud Dataflow interface. At the top, there's a navigation bar with the Google Cloud logo and a search bar containing 'dataflow'. Below it, the main title is 'Dataflow / Jobs'. On the left, there's a sidebar with icons for 'Jobs', 'Logs', and 'Metrics'. The main area is titled 'Jobs' and contains two buttons: '+ CREATE JOB FROM TEMPLATE' and '+ CREATE JOB FROM BUILDER'. The 'CREATE JOB FROM BUILDER' button is highlighted with a red box. A warning message below the buttons states: 'Dataflow jobs use Cloud Storage to store temporary files during pipeline execution. These files have soft delete policy enabled by default. To avoid buckets that your Dataflow jobs use for temporary storage. Learn more'.

2. Selección de la plantilla “Pub/Sub Subscription to BigQuery”, también elegimos un nombre con que identificaremos nuestro job y definimos también la region en este caso como esta todo configurado en us-central1(Iowa) debemos obligatoriamente elegirla para no generar errores.



The screenshot shows a configuration form for creating a Dataflow job. It includes fields for 'Job name' (set to 'transferBigQuery'), 'Regional endpoint' (set to 'us-central1 (Iowa)'), and a 'Dataflow template' dropdown. The 'Dataflow template' dropdown is set to 'Pub/Sub Subscription to BigQuery' and is highlighted with a red box. Below the dropdown, there's a brief description of the template: 'The Pub/Sub Subscription to BigQuery template is a streaming pipeline that reads JSON-formatted messages from a Pub/Sub subscription and writes them to a BigQuery table. You can use the template as a quick solution to move Pub/Sub data to BigQuery.' At the bottom, there's a link 'OPEN TUTORIAL'.

Configuración de la suscripción de Pub/Sub y de la tabla de destino en BigQuery.

3.

Required Parameters

BigQuery output table *	DatosRealTime.DatosTR	BROWSE
Pub/Sub input subscription *	projects/[REDACTED]/subscriptions/registros-sub	▼

4. Especificación de un bucket temporal en Cloud Storage.

Streaming Engine ?

Enable Streaming Engine

Temporary location *

gs://[REDACTED]/temp	BROWSE
Path and filename prefix for writing temporary files. E.g. gs://yourbucket/temp	

5. Ejecución del job y monitoreo hasta el estado “Running”.

The screenshot shows the Google Cloud Dataflow Jobs interface. At the top, there's a navigation bar with 'Google Cloud' and 'dataflow'. Below it, a banner displays a warning about temporary storage usage. The main area is titled 'Jobs' and shows a table of running jobs. One job is listed:

Name	Type	End time	Elapsed time	Start time	Status	SDK version	ID	Region	Insights
transferBigQuery	Streaming		14 sec	Jun 28, 2025, 2:43:32PM	Starting...	2.65.0	2025-06-28_11_43_22-18263910946104903245	us-central1	

6. Verificamos que todo este ejecutando sin errores

This screenshot shows the same Dataflow Jobs page after the job has completed. The status column now shows 'Running' for the entire row. The job details remain the same as in the previous screenshot.

Name	Type	End time	Elapsed time	Start time	Status	SDK version	ID	Region	Insights
transferBigQuery	Streaming		2 min 58 sec	Jun 28, 2025, 2:54:30PM	Running	2.65.0	2025-06-28_11_54_27-388751467614665228	us-central1	

7. Posteriormente al pasar un tiempo de ejecución se estará rellenando la tabla de datos en tiempo real con los datos que envía la página de subastas.

The screenshot shows a BigQuery query results page. At the top, there are navigation buttons for Run, Save, Download, Share, Schedule, Open in, and More. Below that, a SQL query is displayed:

```
1 SELECT * FROM [REDACTED].DataflowTime.Datastore LIMIT 1000
```

The main area is titled "Query completed" and shows a "Results" table. The table has the following columns and data:

Row	id_cliente	cliente	genero	id_producto	producto	precio	cantidad	monto	forma_pago	fecreg
101	1	Raul Ospino	H	10	Cisco	49.74	86	4277.64	Efectivo	2025-06-29 14:13:01.900000 UTC
102	8	Paula Rojas	M	4	Alphabet	141.25	77	10878.25	Credito	2025-06-29 14:17:01.750000 UTC
103	7	Elizabeth Cordero	M	1	Amazon	151.48	75	11361.0	Debito	2025-06-29 14:32:01.907000 UTC
104	2	Alejandro Perez	H	3	Meta	309.09	77	27649.93	Debito	2025-06-29 14:32:02.349000 UTC
105	9	Marién Soto	M	9	Disney	90.24	61	5504.64	Credito	2025-06-29 14:39:01.861000 UTC
106	7	Elizabeth Cordero	M	10	Cisco	49.74	81	3994.14	Debito	2025-06-29 14:39:01.861000 UTC
107	10	Luisa Torres	M	3	Meta	309.09	93	33995.37	Credito	2025-06-29 14:45:01.745000 UTC
108	4	Eric Aracena	H	10	Cisco	49.74	100	4974.0	Debito	2025-06-29 14:55:01.695000 UTC
109	1	Raul Ospino	H	1	Amazon	151.48	95	14390.6	Debito	2025-06-29 15:54:02.330000 UTC
110	2	Alejandro Perez	H	3	Meta	309.09	99	35549.91	Efectivo	2025-06-29 14:02:01.916000 UTC
111	6	Carolina Lopez	M	7	Apple	184.85	26	4806.1	Efectivo	2025-06-29 14:12:01.743000 UTC

Transformaciones realizadas

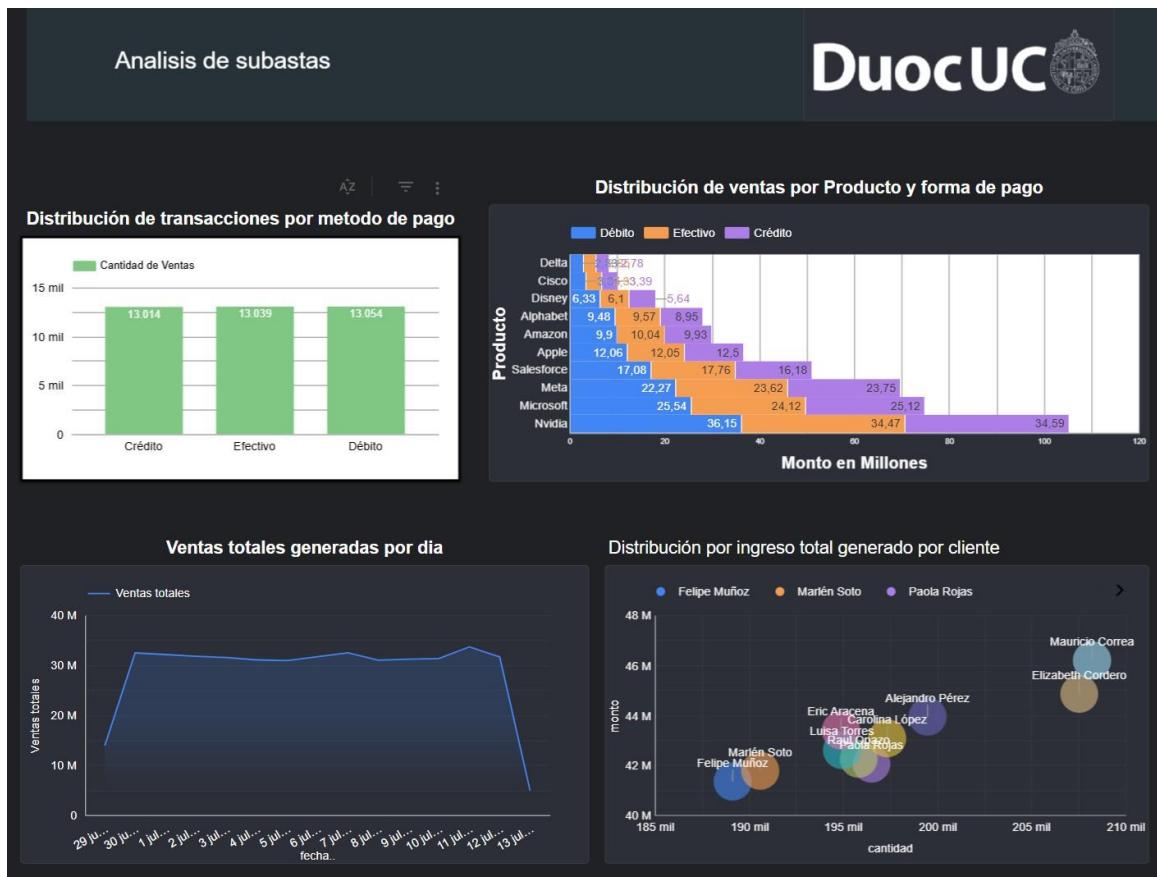
En este proyecto se utilizó el **template preconfigurado “Pub/Sub Subscription to BigQuery”** de Google Cloud Dataflow, el cual permite enrutar mensajes JSON desde un tópico de Pub/Sub directamente hacia una tabla en BigQuery. Este template **no aplica transformaciones personalizadas** por defecto, sino que transfiere los datos en su estado original.

Sin embargo, existen ciertas **transformaciones implícitas** que se deben tener en cuenta:

- **Validación de formato JSON:** los mensajes recibidos deben ser válidos y correctamente estructurados para que puedan insertarse sin errores en BigQuery.
- **Mapeo automático de campos:** los campos del mensaje deben coincidir con los nombres y tipos definidos en el esquema de la tabla de destino.
- **Rechazo de registros inválidos:** en caso de mensajes incompletos o con errores de tipado, el sistema los descarta o los reintenta según la configuración del job.

Aunque en esta versión del pipeline no se aplicaron transformaciones adicionales, se deja abierto el diseño para futuras mejoras que contemplen limpieza de datos, enriquecimiento o validación personalizada utilizando un pipeline de Apache Beam propio.

Etapa 4: Visualización de datos



Herramienta utilizada:

Google Looker Studio

Descripción y procedimiento:

Para la etapa de visualización de datos, se conectó la tabla DatosRealTime.DatosTR desde BigQuery a Google Looker Studio. A partir de esta conexión, se diseñaron dashboards interactivos que permiten el análisis visual de las ventas en tiempo real. Se enfocaron principalmente en indicadores clave de rendimiento (KPI), distribución por género, métodos de pago y temporalidad de las ventas.

1. Distribución de transacciones por método de pago

Este gráfico de barras representa la cantidad total de transacciones realizadas según el método de pago utilizado. Se observa una distribución bastante equilibrada entre Crédito (13.014), Efectivo (13.039) y Débito (13.054), lo que indica que no hay una preferencia clara por un método de pago en particular entre los clientes.

Configuración técnica:

- **Dimensión utilizada:**
forma_pago
(Esta columna agrupa las transacciones según el método de pago utilizado).
- **Métrica utilizada:**
cantidad_registros
(Cuenta cuántas transacciones hay por cada categoría de forma de pago).
- **Tipo de visualización:**
Gráfico de barras verticales.
- **Etiquetas:**
Se mostraron los valores exactos sobre cada barra para facilitar la lectura comparativa.

Resultado obtenido:

- Las tres formas de pago (Crédito, Efectivo y Débito) se encuentran bastante equilibradas en volumen de uso, con más de 13,000 transacciones cada una.
-

2. Distribución de ventas por Producto y forma de pago

El gráfico de barras apiladas muestra el monto total en millones generado por cada producto, desglosado según el método de pago. Por ejemplo, se destaca que productos como Microsoft, Meta y Nvidia generan mayores ingresos con pagos con tarjeta de crédito. Esta visualización permite identificar tanto los productos más vendidos como la forma de pago más común asociada a cada uno.

Configuración :

- **Dimensión principal:**
producto
(Categoriza las ventas por cada producto subastado).

- **Dimensión de desglose (o segmentación):**
forma_pago
(Permite descomponer las ventas por método de pago dentro de cada producto).
- **Métrica utilizada:**
monto_en_millones
Es un **campo calculado** creado a partir de la siguiente fórmula:
 $\text{SUM}(\text{monto}) / 1000000$

Esto permite expresar los montos en **millones de unidades monetarias**, facilitando la visualización a gran escala.

- **Tipo de visualización:**
Gráfico de barras apiladas horizontales.
Cada barra representa un producto y se divide según el método de pago.

Resultado obtenido:

- Productos como **Nvidia, Microsoft y Meta** alcanzan los montos más altos.
 - La forma de pago más frecuente en los productos con mayores ventas tiende a ser el **crédito**, como se aprecia en la proporción de color.
 - Permite comparar no solo cuánto se vende por producto, sino también **qué forma de pago predomina** en cada uno.
-

3. Ventas totales generadas por día

Este gráfico de línea ilustra cómo han evolucionado las ventas diarias durante el periodo analizado (del 29 de junio al 13 de julio). Se evidencia una tendencia estable, con montos diarios de venta que oscilan en torno a los 30 millones, destacándose un pequeño aumento el 12 de julio. Esta información es clave para identificar patrones de comportamiento y días de mayor actividad comercial.

Configuración:

- **Dimensión utilizada:**
fecha
(Representa el día en que se realizó cada transacción. Se utiliza para agrupar las ventas por día).
- **Métrica utilizada:**
monto

Se usó la **auto suma del campo monto** (función de agregación automática del sistema de visualización), lo que calcula el total de ventas por cada día.

- **Tipo de visualización:**

Gráfico de líneas.

Muestra la evolución diaria de las ventas de forma continua.

Resultado obtenido:

- Se observa un volumen de ventas relativamente estable entre el 30 de junio y el 12 de julio, con un leve aumento hacia los días finales.
 - La gráfica permite detectar días con menor o mayor actividad comercial, útil para el análisis de comportamiento del usuario o efectividad de campañas.
-

4. Distribución por ingreso total generado por cliente

Este gráfico de dispersión (bubble chart) relaciona el monto total generado por cada cliente con la cantidad de compras realizadas. El tamaño de la burbuja representa el volumen de ingreso. Se identifican clientes como Mauricio Correa y Elizabeth Cordero como los de mayor valor, tanto por volumen como por número de compras, lo cual permite reconocer a los compradores más influyentes en las subastas.

Configuración técnica:

- **Dimensión utilizada:**

cliente

(Identifica a cada cliente individualmente).

- **Métricas:**

- **Eje X:**

cantidad con agregación **SUM()**

(Representa el número total de transacciones realizadas por cada cliente).

- **Eje Y:**

monto con agregación **SUM()**

(Indica el monto total gastado por cada cliente).

- **Tamaño de burbuja:**
Campo calculado con la siguiente fórmula:

$\text{SUM(monto)} / \text{SUM(cantidad)}$

Esto calcula el **promedio de gasto por transacción** (ticket promedio), lo que permite identificar a clientes con compras más costosas, independientemente de la cantidad de transacciones.

- **Tipo de visualización:**

Gráfico de burbujas.

Cada burbuja representa a un cliente, posicionada según su comportamiento y con un tamaño proporcional al ticket promedio.

Resultado obtenido:

- Clientes como **Mauricio Correa** y **Elizabeth Cordero** se ubican en la parte superior derecha, indicando alto volumen de compras y alto monto total.
- Clientes con menor frecuencia de compra y bajo gasto aparecen en la zona inferior izquierda.
- El tamaño de burbuja ayuda a distinguir entre quienes compran mucho con montos bajos y quienes hacen menos compras, pero de alto valor.

[Link Looker Dashboard](#)

Propuesta de Valor

Pregunta de datos en Bash

"¿Qué patrones históricos de comportamiento urbano y financiero podrían ayudarme a decidir en qué sectores lanzar una flota de taxis eléctricos o servicios on-demand, optimizando los medios de pago según la demanda estacional?"

Gracias al uso de **Bash y Apache Beam**, se ejecutó un pipeline de ETL que procesó más de 3 años de viajes históricos de taxis de NYC en formato Parquet. Esto permitió cargar a BigQuery datos limpios y confiables, analizados luego en Looker Studio.

Los dashboards responden a esta pregunta mostrando:

- Tendencias de **ingresos mensuales** para identificar estacionalidades y picos de actividad.
- **Comparación por método de pago** (crédito vs efectivo) mes a mes.
- Días de la semana con más actividad (**martes, miércoles y jueves** como días de alta demanda).
- Evolución anual de las **preferencias de pago**, donde el **uso de crédito crece** sostenidamente.

Valor estratégico generado:

Una empresa externa podría usar esta información para:

- Detectar zonas y períodos del año con alta rentabilidad.
- Estimar qué tipo de método de pago habilitar según la zona o el día.
- Decidir si introducir flotas alternativas (como taxis eléctricos o apps on-demand) con foco en horarios rentables.

Pregunta de datos en Streaming

"¿Qué tan rentable es el comportamiento de compra de los usuarios en subastas electrónicas en tiempo real y cómo podría adaptar mis productos o promociones para competir o ingresar al mercado?"

La arquitectura streaming captura datos en tiempo real desde el sistema de subastas a través de Cloud Run, los enruta con Pub/Sub y los carga a BigQuery mediante Dataflow.

Los dashboards desarrollados en Looker Studio permiten ver:

- **Ventas diarias** en tiempo real (con tendencias por fecha).
- Productos más vendidos y **métodos de pago usados por producto**.
- Identificación de **clientes clave** según ticket promedio y frecuencia de compra.
- Equilibrio en el uso de **crédito, débito y efectivo**, con volúmenes similares.

Valor estratégico generado:

Una empresa externa interesada en ingresar al mercado podría:

- Detectar qué productos generan más ingresos y en qué formas de pago.
- Identificar posibles segmentos objetivo (por ejemplo: mujeres que compran más productos tecnológicos).
- Ajustar estrategias de precios y promociones basándose en picos de compra diaria.

Ambas preguntas reflejan **necesidades reales del mercado** y son respondidas eficazmente mediante una arquitectura híbrida que combina:

- **Batch (con Bash + Beam)** para analizar el pasado y entender comportamientos consolidados.
- **Streaming (con GCP Serverless)** para reaccionar al presente y anticipar decisiones.

Esto convierte tu solución en un sistema **accionable, escalable y competitivo**, ideal para empresas que desean ingresar o destacar en mercados basados en movilidad urbana y subastas digitales.

Conclusión General

El desarrollo de este proyecto se fundamentó desde su origen en una propuesta de valor clara: **proveer una solución analítica híbrida que combine datos históricos y en tiempo real**, permitiendo a las empresas tomar decisiones informadas, anticiparse a comportamientos y detectar oportunidades comerciales de forma dinámica y escalable. Cada etapa del flujo desde la ingesta, el procesamiento y almacenamiento, hasta la visualización fue diseñada para responder preguntas reales de negocio que surgen al observar los datos:

- ¿Qué patrones históricos pueden ayudarme a definir estrategias comerciales futuras?
- ¿Cómo reaccionar en tiempo real al comportamiento de clientes en subastas para maximizar ingresos?

Para responder estas interrogantes, se implementó una arquitectura Lambda sobre Google Cloud Platform. El uso de herramientas como **Cloud Run, Pub/Sub, Dataflow y BigQuery** permitió capturar, procesar y almacenar datos tanto batch como streaming de forma eficiente y automatizada. Por su parte, **Looker Studio** brindó una capa de visualización accionable que transformó los datos en conocimiento inmediato.

Gracias a esta integración:

- El procesamiento **batch** con Bash y Apache Beam sobre archivos Parquet históricos habilitó el análisis de tendencias urbanas, formas de pago y estacionalidades clave.
- El procesamiento **streaming** capturó en segundos el comportamiento del usuario en subastas electrónicas, permitiendo visualizar en tiempo real las ventas por cliente, producto y forma de pago.

En conjunto, se construyó un flujo de datos **robusto, flexible y orientado a resultados**, que permite:

- Analizar el pasado para planificar con precisión.
- Visualizar el presente para reaccionar con agilidad.
- Y conectar ambos mundos para **anticipar decisiones estratégicas** que impacten directamente la rentabilidad y competitividad de una empresa.

Esta solución no solo responde a las necesidades del cliente actual, sino que **representa una ventaja competitiva real frente a empresas de la industrias**, al ofrecer automatización, escalabilidad y análisis accionable en una sola plataforma unificada.