**Resumen**

Este proyecto implementa un pipeline completo de datos para procesar y analizar la información histórica de viajes de taxi de la ciudad de Nueva York (NYC TLC Trip Record Data) correspondientes a los servicios Yellow y Green entre los años 2015 y 2025.  
El objetivo principal es construir una arquitectura moderna tipo medallion (bronze → silver → gold) sobre Snowflake, orquestada mediante Mage, que permita garantizar la calidad, estandarización, escalabilidad y análisis eficiente de grandes volúmenes de datos.

El flujo general del proyecto es el siguiente:

* **Ingesta (Bronze/raw):**  
  Descarga automatizada de archivos Parquet desde la fuente oficial de NYC TLC, carga controlada y segmentada por mes/año en Snowflake con chunking y reintentos, manejo de metadatos de ingesta (run\_id, timestamps, número de filas) y idempotencia para evitar duplicados.
* **Transformación (Silver):**  
  Estandarización de tipos y zonas horarias, normalización de valores clave (ej. payment\_type a etiquetas legibles), aplicación de reglas de calidad mínimas (no nulos, distancias/montos no negativos, viajes razonables ≤ 24h) y unificación de datos Yellow/Green en un esquema común con columna service\_type.  
  Además, se enriquece la información con la tabla oficial de Taxi Zones para asociar IDs de localización con zonas y boroughs.
* **Modelado analítico (Gold):**  
  Construcción de un modelo en estrella con un hecho principal fct\_trips (1 fila = 1 viaje) y dimensiones conformadas (dim\_date, dim\_zone, dim\_vendor, dim\_rate\_code, dim\_payment\_type, dim\_service\_type, etc.), documentadas y probadas con dbt tests (not\_null, unique, accepted\_values, relationships).
* **Optimización de consultas:**  
  Implementación y evaluación de clustering keys en la tabla de hechos para mejorar el rendimiento de consultas analíticas. Se miden métricas de pruning y tiempos de ejecución antes y después del clustering usando Query Profile de Snowflake.
* **Seguridad y operación:**  
  Uso de Secrets en Mage para almacenar credenciales de Snowflake de forma segura y configuración de una cuenta de servicio con privilegios mínimos para la ejecución de cargas y transformaciones, siguiendo buenas prácticas de seguridad y gobierno de datos.

Este proyecto también incluye:

* **Pruebas de calidad y auditoría:** validación de datos (conteos por mes/servicio, % de filas descartadas por reglas de calidad) y pruebas automáticas con dbt para garantizar confiabilidad.
* **Respuestas a preguntas de negocio** directamente sobre la capa **gold**, facilitando análisis como demanda por zona y mes, ingresos y propinas, velocidades promedio y patrones temporales de viajes.

**Matriz de cobertura**

A continuación, se muestra la matriz de cobertura dividida por cada año y que incluye los campos de año, mes y status en los taxis verdes y amarillos. OK muestra que su estado de carga fue exitoso, es decir, se cargó datos de un archivo parquet de ese año y mes para el servicio correspondiente (amarillo o verde) y MISSING es que no se

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2015 | 1 | OK | OK |
| 2015 | 2 | OK | OK |
| 2015 | 3 | OK | OK |
| 2015 | 4 | MISSING | OK |
| 2015 | 5 | OK | OK |
| 2015 | 6 | OK | OK |
| 2015 | 7 | OK | OK |
| 2015 | 8 | OK | OK |
| 2015 | 9 | OK | OK |
| 2015 | 10 | OK | OK |
| 2015 | 11 | OK | OK |
| 2015 | 12 | OK | OK |

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2016 | 1 | OK | OK |
| 2016 | 2 | OK | OK |
| 2016 | 3 | OK | OK |
| 2016 | 4 | OK | OK |
| 2016 | 5 | OK | OK |
| 2016 | 6 | OK | OK |
| 2016 | 7 | OK | OK |
| 2016 | 8 | OK | OK |
| 2016 | 9 | OK | OK |
| 2016 | 10 | OK | OK |
| 2016 | 11 | OK | OK |
| 2016 | 12 | OK | OK |

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2017 | 1 | OK | OK |
| 2017 | 2 | OK | OK |
| 2017 | 3 | OK | OK |
| 2017 | 4 | OK | OK |
| 2017 | 5 | OK | OK |
| 2017 | 6 | OK | OK |
| 2017 | 7 | OK | OK |
| 2017 | 8 | OK | OK |
| 2017 | 9 | OK | OK |
| 2017 | 10 | OK | OK |
| 2017 | 11 | OK | OK |
| 2017 | 12 | OK | OK |

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2018 | 1 | OK | OK |
| 2018 | 2 | OK | OK |
| 2018 | 3 | OK | OK |
| 2018 | 4 | OK | OK |
| 2018 | 5 | OK | OK |
| 2018 | 6 | OK | OK |
| 2018 | 7 | OK | OK |
| 2018 | 8 | OK | OK |
| 2018 | 9 | OK | OK |
| 2018 | 10 | OK | OK |
| 2018 | 11 | OK | OK |
| 2018 | 12 | OK | OK |

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2019 | 1 | OK | OK |
| 2019 | 2 | OK | OK |
| 2019 | 3 | OK | OK |
| 2019 | 4 | OK | OK |
| 2019 | 5 | OK | OK |
| 2019 | 6 | OK | OK |
| 2019 | 7 | OK | OK |
| 2019 | 8 | OK | OK |
| 2019 | 9 | OK | OK |
| 2019 | 10 | OK | OK |
| 2019 | 11 | OK | OK |
| 2019 | 12 | OK | OK |

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2020 | 1 | OK | OK |
| 2020 | 2 | OK | OK |
| 2020 | 3 | OK | OK |
| 2020 | 4 | OK | OK |
| 2020 | 5 | OK | OK |
| 2020 | 6 | OK | OK |
| 2020 | 7 | OK | OK |
| 2020 | 8 | OK | OK |
| 2020 | 9 | OK | OK |
| 2020 | 10 | OK | OK |
| 2020 | 11 | OK | OK |
| 2020 | 12 | OK | OK |

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2021 | 1 | OK | OK |
| 2021 | 2 | OK | OK |
| 2021 | 3 | OK | OK |
| 2021 | 4 | OK | OK |
| 2021 | 5 | OK | OK |
| 2021 | 6 | OK | OK |
| 2021 | 7 | OK | OK |
| 2021 | 8 | OK | OK |
| 2021 | 9 | OK | OK |
| 2021 | 10 | OK | OK |
| 2021 | 11 | OK | OK |
| 2021 | 12 | OK | OK |

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2022 | 1 | OK | OK |
| 2022 | 2 | OK | OK |
| 2022 | 3 | OK | OK |
| 2022 | 4 | OK | OK |
| 2022 | 5 | OK | OK |
| 2022 | 6 | OK | OK |
| 2022 | 7 | OK | OK |
| 2022 | 8 | OK | OK |
| 2022 | 9 | OK | OK |
| 2022 | 10 | OK | OK |
| 2022 | 11 | OK | OK |
| 2022 | 12 | OK | OK |

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2023 | 1 | OK | OK |
| 2023 | 2 | OK | OK |
| 2023 | 3 | OK | OK |
| 2023 | 4 | OK | OK |
| 2023 | 5 | OK | OK |
| 2023 | 6 | OK | OK |
| 2023 | 7 | OK | OK |
| 2023 | 8 | OK | OK |
| 2023 | 9 | OK | OK |
| 2023 | 10 | OK | OK |
| 2023 | 11 | OK | OK |
| 2023 | 12 | OK | OK |

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2024 | 1 | OK | OK |
| 2024 | 2 | OK | OK |
| 2024 | 3 | OK | OK |
| 2024 | 4 | OK | OK |
| 2024 | 5 | OK | OK |
| 2024 | 6 | OK | OK |
| 2024 | 7 | OK | OK |
| 2024 | 8 | OK | OK |
| 2024 | 9 | OK | OK |
| 2024 | 10 | OK | OK |
| 2024 | 11 | OK | OK |
| 2024 | 12 | OK | OK |

|  |  |  |  |
| --- | --- | --- | --- |
| AÑO | MES | YELLOW\_STATUS | GREEN\_STATUS |
| 2025 | 1 | OK | OK |
| 2025 | 2 | OK | OK |
| 2025 | 3 | OK | OK |
| 2025 | 4 | OK | OK |
| 2025 | 5 | OK | OK |
| 2025 | 6 | OK | OK |
| 2025 | 7 | OK | OK |
| 2025 | 8 | OK | OK |
| 2025 | 9 | OK | OK |
| 2025 | 10 | MISSING | MISSING |
| 2025 | 11 | MISSING | MISSING |
| 2025 | 12 | MISSING | MISSING |

Durante la verificación de la cobertura mensual de los datasets Yellow y Green (2015 – 2025) se construyó una matriz comparando el número de filas cargadas en el esquema BRONZE.

La matriz mostró dos tipos de vacíos:

**Yellow – Abril 2015 (2015-04)**

Pese a que el archivo para Abril 2015 si existe dentro del repositorio, este no pudo ser accedido para ingestar los datos en SNOWFLAKE, por ende se marca como MISSING.

**Yellow / Green – Últimos 3 meses de 2025 (2025-10, 2025-11, 2025-12)**

* El proyecto tenía datos hasta diciembre de 2025, pero la fuente oficial aún no ha publicado los archivos de estos meses al momento de la ejecución del backfill.
* La verificación de URLs para:
* yellow\_tripdata\_2025-10.parquet
* yellow\_tripdata\_2025-11.parquet
* yellow\_tripdata\_2025-12.parquet

devuelve 404 Not Found.

* Se documenta como MISSING por falta de disponibilidad en la fuente original.

Todos los demás meses disponibles en el repositorio público fueron descargados e ingeridos correctamente. La estrategia de ingesta implementa idempotencia: si en el futuro se publican los archivos faltantes, basta con re‐ejecutar el pipeline para completar los huecos sin duplicar datos ya cargados.

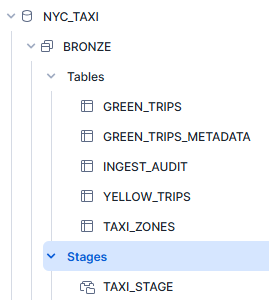
**Arquitectura**

El proyecto implementa una arquitectura de datos basada en el patrón Medallion sobre Snowflake, con el para procesar grandes volúmenes de información de manera escalable, confiable y fácilmente analizable. El flujo de extremo a extremo sigue la siguiente secuencia:

**Parquet → Bronze → Silver → Gold**

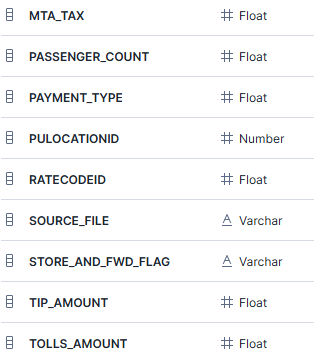
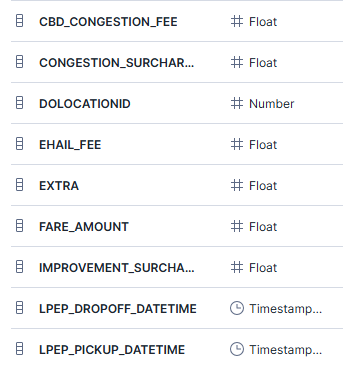
1. **Bronze (Capa Raw)**
   * Contiene los datos crudos tal como se obtienen de la fuente oficial de NYC TLC, sin transformaciones más allá de la ingesta inicial.
   * Aquí se preserva el esquema original de cada servicio (yellow y green) para mantener trazabilidad con el origen.
   * Se agregan metadatos mínimos (nombre de archivo, fecha de ingesta) para facilitar control de versiones e idempotencia.

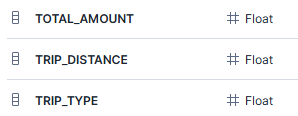
A continuación, se muestra una imagen del esquema Bronze en Snowflake



**GREEN\_TRIPS**: los datos de todos los viajes de taxis verdes

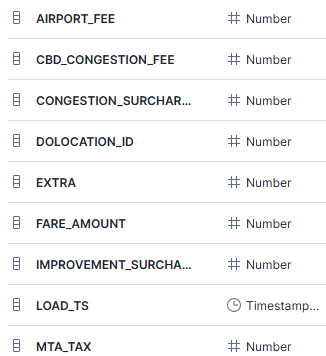
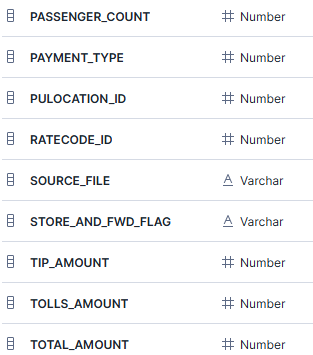
Contiene las siguientes columnas:

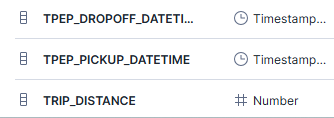




**YELLOW\_TRIPS**: los datos de todos los viajes de taxis amarillos

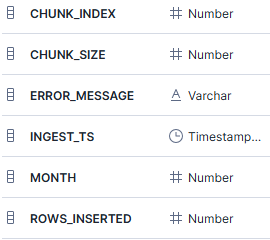
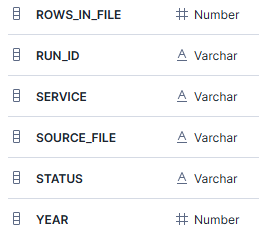
Tiene las siguientes columnas:



**INGEST\_AUDIT:** metadatos de ingesta de los datos de los taxis amarillos

Tiene las siguientes columnas:

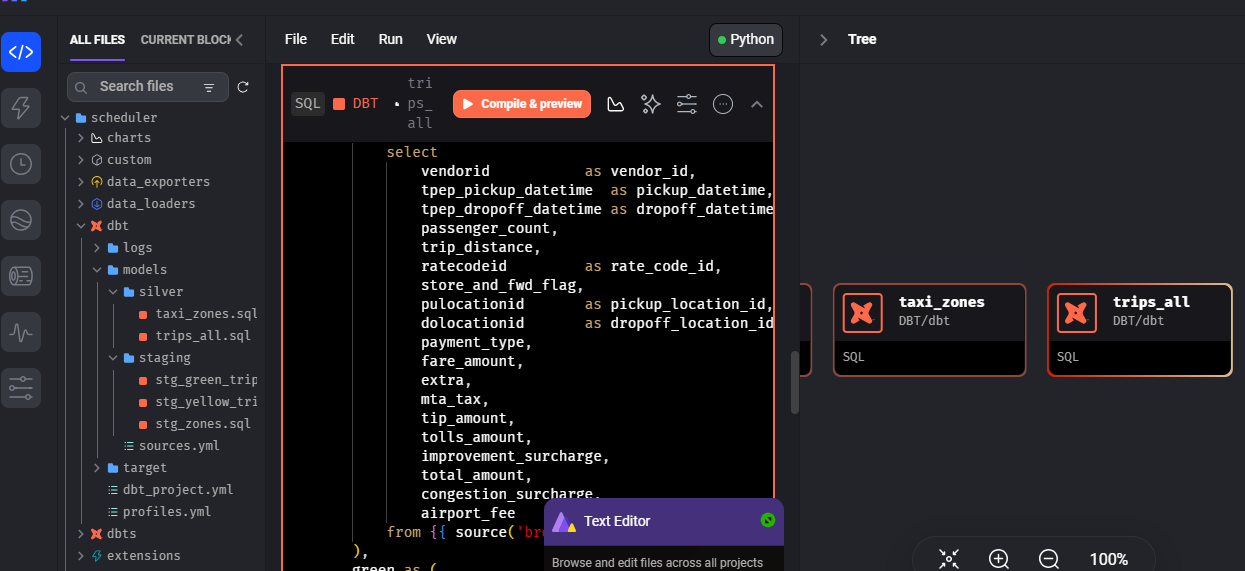
 

**GREEN\_TRIPS\_METADATA:** metadatos de ingesta de los datos de los taxis verdes

Tiene las mismas columnas que ingest\_audit

1. **Silver (Capa Curada)**
   * Los datos se estandarizan para homogenizar tipos y zonas horarias.
   * Se aplican reglas de calidad mínimas: eliminación de registros con campos clave nulos, distancias y montos no negativos, y viajes con duración razonable (≤ 24 horas).
   * Se normalizan valores como payment\_type (convertidos a etiquetas legibles).
   * Se agrega la columna service\_type para distinguir entre viajes yellow y green.
   * Se realiza el enriquecimiento mediante la unión con la tabla oficial de Taxi Zones para asociar IDs de localización con zonas y boroughs.

A continuación, se muestran capturas del dbt usado en la capa silver



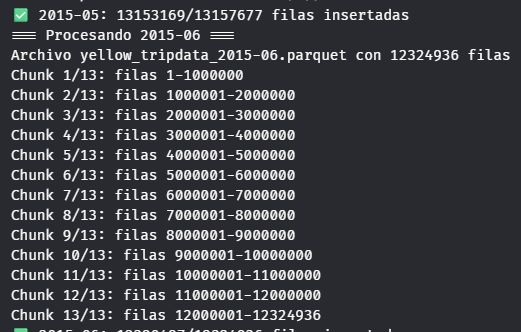
1. **Gold (Capa Analítica)**
   * Se construye un **modelo en estrella** diseñado para análisis de negocio y consultas rápidas.
   * Tabla de hechos principal: fct\_trips (una fila por viaje).
   * Dimensiones conformadas: dim\_date, dim\_zone, dim\_vendor, dim\_rate\_code, dim\_payment\_type, dim\_service\_type, dim\_trip\_type.
   * Incluye documentación y pruebas automáticas (dbt tests: not\_null, unique, relationships, accepted\_values).
2. **Orquestación y Transformación**
   * El flujo de ingesta y transformación es gestionado por **Mage**, que automatiza el **backfill mensual** (descarga → carga a Bronze → limpieza y transformación a Silver).
   * Una vez finalizada la etapa Silver, Mage dispara la ejecución de **dbt** para construir y actualizar los modelos **Gold** y ejecutar las pruebas de calidad definidas.
   * Esta combinación asegura un pipeline reproducible, escalable y fácilmente mantenible.

**Estrategia de Ingesta**

Para poblar la capa Bronze se construyó un proceso de backfill mensual que descarga, valida y carga de forma incremental los archivos históricos de viajes de taxi de Nueva York. Esta estrategia asegura reproducibilidad, manejo de grandes volúmenes de datos y control de calidad desde la ingesta inicial. Además, la descarga se hizo por mes y por chunks de un millón de archivos

1. **Descarga de archivos Parquet**
   * El pipeline genera dinámicamente el nombre del archivo según el servicio (yellow o green), el año y el mes.
   * Se realiza la descarga desde el repositorio público de NYC TLC (https://d37ci6vzurychx.cloudfront.net/trip-data/).

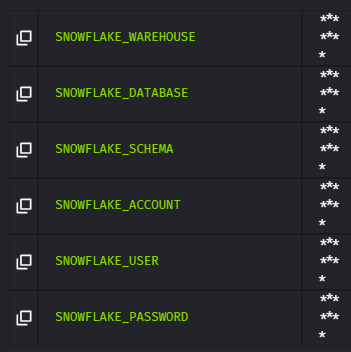
Este es un ejemplo de cómo corre uno de los loaders



1. **Subida al stage de Snowflake**
   * Los archivos descargados localmente se suben a un stage interno de Snowflake utilizando el comando PUT.
   * Se utiliza un FILE FORMAT tipo PARQUET para estandarizar la lectura.
   * Para evitar bloqueos por certificados durante el desarrollo en entornos WSL/Colab se configuró la conexión con insecure\_mode=True.
2. **Carga en tablas RAW/Bronze**
   * Los datos se copian desde el stage a las tablas del esquema BRONZE de la base de datos NYC\_TAXIS mediante COPY INTO.
   * Los datos como tal de los taxis amarillos fueron copiados a una tabla llamada YELLOW\_TRIPS dentro del esquema BRONZE
   * Se agregan metadatos relevantes:
     + RUN\_ID (identificador de la ejecución)
     + INGEST\_TS (timestamp de ingesta)
     + SOURCE\_FILE (nombre del archivo Parquet)
     + ROWS\_IN\_FILE y ROWS\_INSERTED para control de calidad.
3. **Idempotencia**
   * Antes de insertar datos de un mes ya existente, se ejecuta un DELETE filtrando por SOURCE\_FILE para garantizar que la ingesta sea **idempotente** y evitar duplicados si el pipeline se relanza.
4. **Chunking y reintentos**
   * Para manejar archivos muy grandes (decenas de millones de filas) se implementó un procesamiento por **bloques (chunk\_size)** con paginación controlada.
   * Cada bloque tiene hasta *N* reintentos configurados para tolerar errores temporales de red o carga.

Esta estrategia permite reejecutar el backfill completo de forma segura cuando aparecen nuevos meses o si algún proceso falló en una ejecución anterior.

A continuación, se muestra una captura de los secretos en Mage Secrets



Donde SNOWFLAKE\_WAREHOUSE es el nombre del warehouse que se utilizó para poner la base de datos.

SNOWFLAKE\_DATABASE: el nombre de la base de datos con todos los esquemas y tablas para este proyecto

SNOWFLAKE\_SCHEMA: es el nombre del esquema de la capa raw en la base de datos

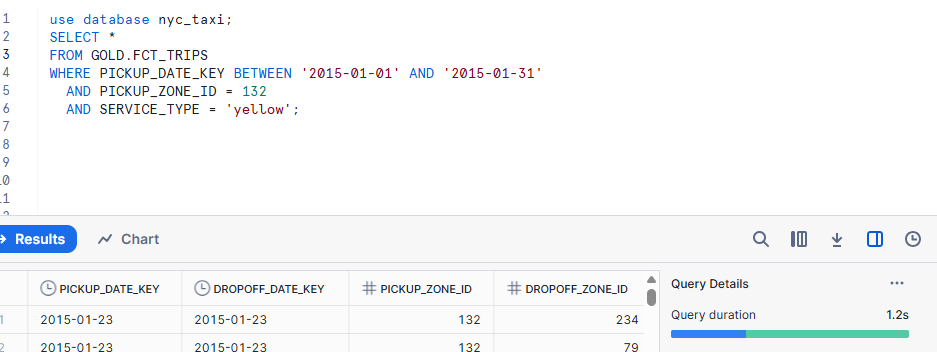
SNOWFLAKE\_ACCOUNT: mi identificador de la cuenta de snowflake

SNOWFLAKE\_USER: el nombre del usuario con permisos mínimos

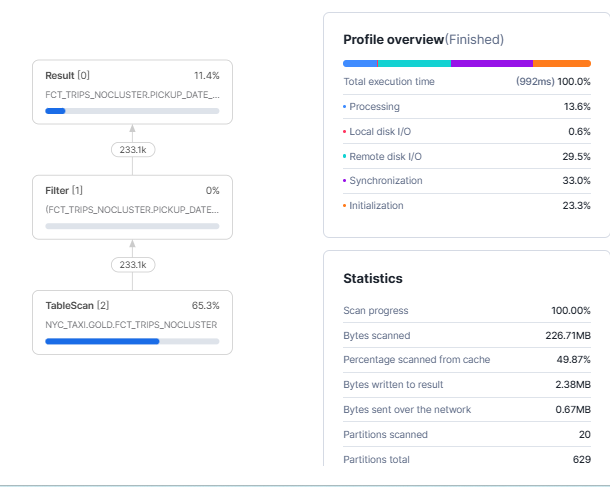
SNOWFLAKE\_PASSWORD: la contraseña de mi usuario para entrar a snowflake

Opción 1: por **fecha de pickup + zona de pickup**

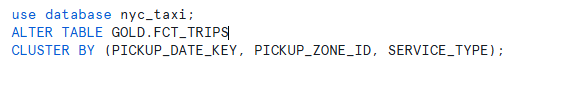
Primero se hace una consulta significativa sin aplicar el clustering. Esta consulta selecciona todos los viajes almacenados en la tabla de hechos GOLD.FCT\_TRIPS cuya fecha de recogida (PICKUP\_DATE\_KEY) esté entre el 1 y el 31 de enero de 2015, que tengan como zona de origen el identificador 132 (PICKUP\_ZONE\_ID = 132) y que correspondan a taxis de tipo yellow (SERVICE\_TYPE = 'yellow').  
En otras palabras, devuelve todos los registros de viajes de taxis amarillos realizados durante enero de 2015 que iniciaron en la zona 132. Esta consulta se muestra a continuación:



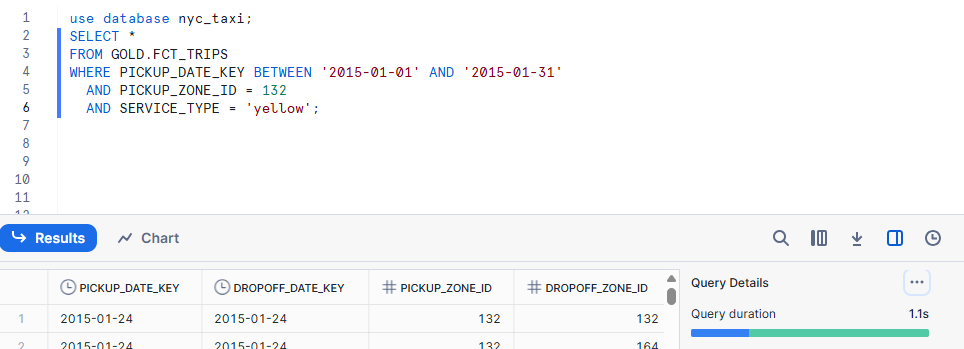
Se obtuvo los siguientes resultados en Query profile:

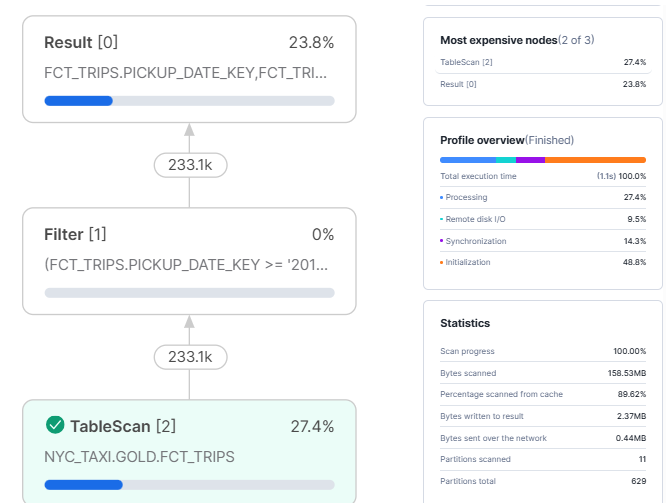


Ahora se aplica la siguiente clustering key a la tabla de fct\_trips en el esquema gold

Al definir esta clave de clustering, Snowflake intentará agrupar las micro-particiones para que los datos de viajes con fechas cercanas, misma zona de recogida y tipo de servicio similar queden almacenados juntos. Esto mejora el **pruning**: cuando hagas consultas filtrando por fechas, zonas y/o tipo de servicio, Snowflake leerá menos micro-particiones, lo que puede reducir el tiempo de ejecución y el costo de escaneo de datos.

Se obtuvo los siguientes resultados

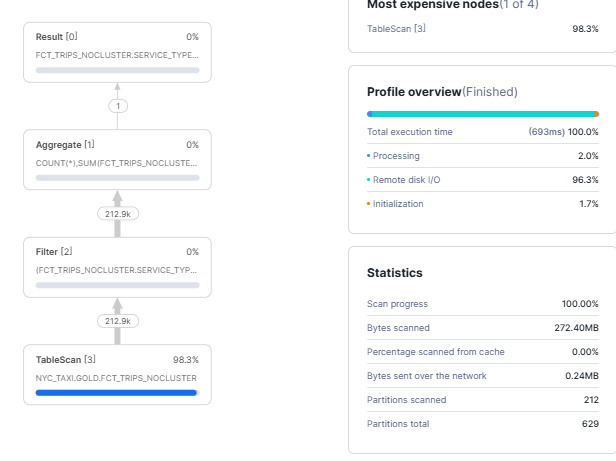




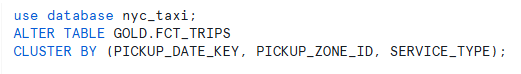
**Opción 2: Clustering combinado: fecha + zona origen + tipo de servicio**

Primero se hace la consulta representativa. Esta consulta trabaja sobre la base de datos NYC\_TAXI y analiza los viajes de taxi amarillos almacenados en la tabla GOLD.FCT\_TRIPS. Primero establece que se va a usar esa base con USE DATABASE NYC\_TAXI. Luego selecciona todos los viajes cuya fecha de recogida (PICKUP\_DATE\_KEY) esté entre el 1 y el 31 de enero de 2020, que pertenezcan a la zona de origen con ID 132 y cuyo tipo de servicio sea “yellow”. Con esos datos filtrados, agrupa por el campo SERVICE\_TYPE y calcula dos métricas: el total de viajes (COUNT(\*) AS total\_trips) y el promedio del monto total pagado por viaje (AVG(TOTAL\_AMOUNT) AS avg\_fare). El resultado es un resumen estadístico para los taxis amarillos en esa zona y mes, mostrando cuántos viajes hubo y el promedio de la tarifa total.





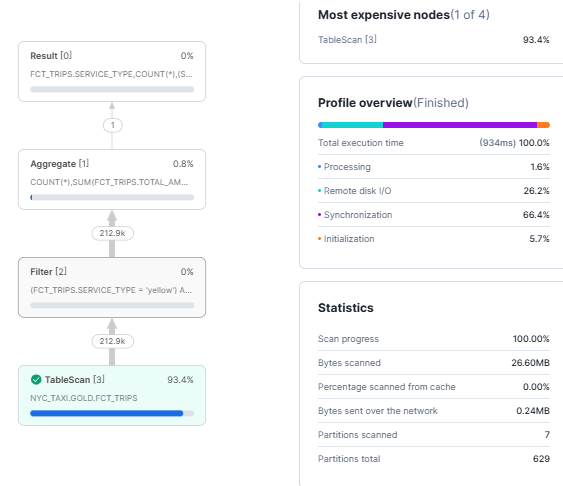
Se aplicó la siguiente clustering key:



Esta organiza físicamente los datos de la tabla GOLD.FCT\_TRIPS para que queden agrupados según la fecha del viaje (PICKUP\_DATE\_KEY), la zona de recogida (PICKUP\_ZONE\_ID) y el tipo de servicio (SERVICE\_TYPE, como taxis amarillos o verdes). Esto permite que Snowflake reduzca la cantidad de micro-particiones que necesita leer cuando se realizan consultas que filtran por estas columnas, mejorando el rendimiento de las búsquedas y disminuyendo el costo de procesamiento.

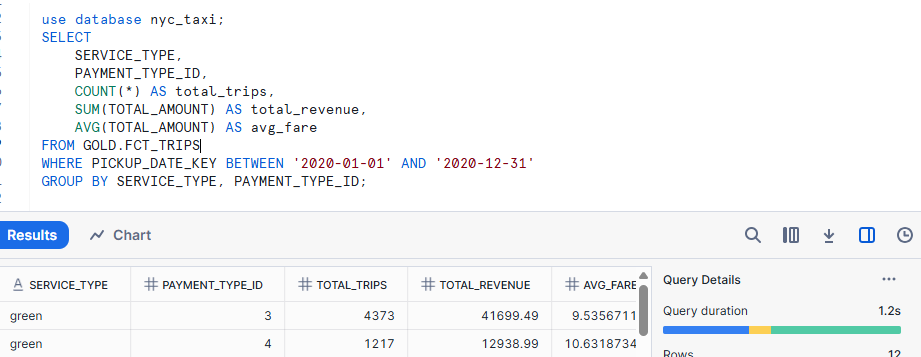
A continuación se muestra la misma consulta después de aplicada la clustering key.

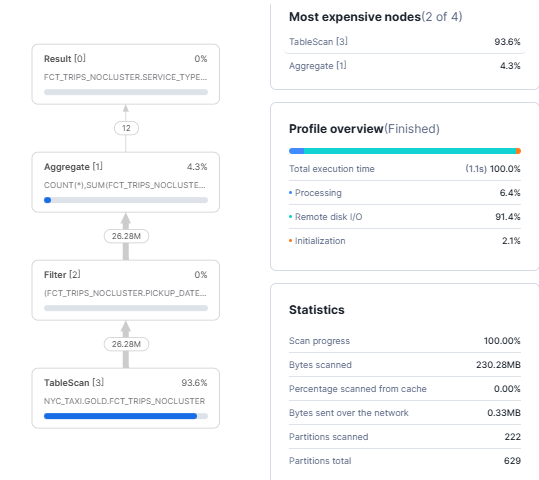




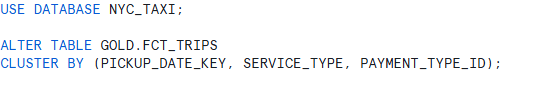
**Opción 3: Para análisis de tarifas o costos**

Primero se hace una consulta significativa sin aplicar ninguna clustering key. Primero filtra los registros cuya fecha de recogida (PICKUP\_DATE\_KEY) esté entre el 1 de enero y el 31 de diciembre de 2020. Luego agrupa los datos por dos dimensiones: tipo de servicio (SERVICE\_TYPE, por ejemplo *yellow* o *green*) y tipo de pago (PAYMENT\_TYPE\_ID, que representa el método de pago como tarjeta, efectivo, etc.). El resultado es un resumen que permite comparar cuántos viajes y qué ingresos generó cada tipo de pago dentro de cada tipo de servicio durante todo el 2020. Solo útil si haces rangos por fechas y filtros por montos.



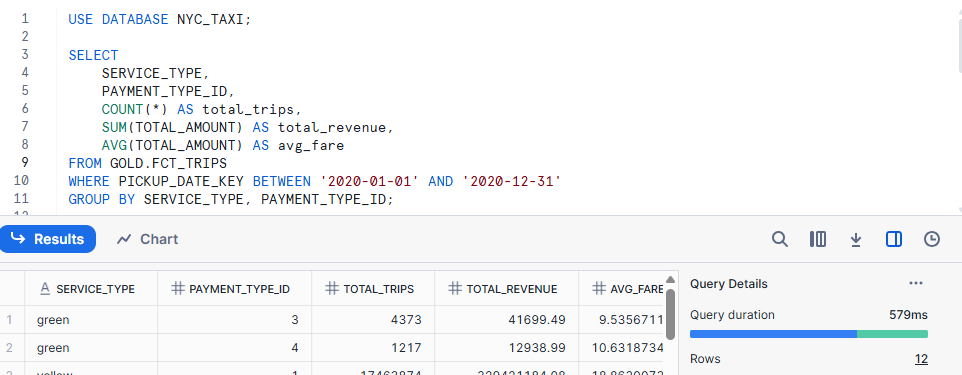


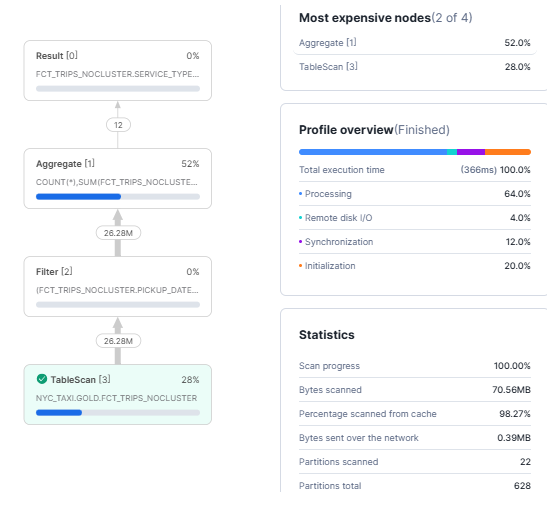
Se aplicó la siguiente clustering key



Esta clustering key reorganiza físicamente los datos de la tabla GOLD.FCT\_TRIPS para que queden ordenados y agrupados primero por la fecha de inicio del viaje (PICKUP\_DATE\_KEY), luego por el tipo de servicio (SERVICE\_TYPE, como taxi amarillo o verde) y, dentro de cada grupo, por el método de pago (PAYMENT\_TYPE\_ID). Esto optimiza el rendimiento de las consultas que filtran o agrupan por estas columnas —por ejemplo, análisis de viajes por fecha, tipo de taxi y forma de pago— ya que Snowflake puede leer menos micro-particiones (pruning) y acelerar el tiempo de respuesta de las consultas que siguen ese patrón.

Ahora se volvió a hacer la misma consulta anterior una vez aplicada esta key





**Conclusiones sobre el clustering**

El uso de clustering keys en la tabla de hechos GOLD.FCT\_TRIPS sí aporta valor porque mejora el *pruning* de micro-particiones en Snowflake y, con ello, reduce el tiempo de ejecución y el volumen de datos escaneados en consultas analíticas frecuentes. Antes de aplicar clustering, las consultas sobre rangos de fechas y filtros por zonas/tipos de servicio leían muchas micro-particiones innecesarias; tras aplicar las claves de clustering, el Query Profile mostró menos particiones escaneadas y tiempos más bajos.

Sin embargo, es clave no sobre-clusterizar: demasiadas columnas en la clave pueden fragmentar el almacenamiento y dificultar el mantenimiento automático del clustering.  
Con base en el patrón de uso documentado:

* Las consultas principales filtran por fecha de viaje (PICKUP\_DATE\_KEY),
* A menudo restringen también por zona de origen (PICKUP\_ZONE\_ID),
* Y diferencian el tipo de servicio (SERVICE\_TYPE).
* En algunos análisis específicos de negocio se agrega el método de pago (PAYMENT\_TYPE\_ID) para estudiar ingresos y medios de cobro.

Por tanto, la clave recomendada a largo plazo es la de la primera opción, ya que equilibra el beneficio de pruning con simplicidad y cubre la mayoría de consultas sin introducir demasiada cardinalidad.

En resumen, sí conviene mantener clustering, pero con una clave moderada (fecha + zona + servicio) para evitar sobrecostos de mantenimiento y mantener un buen rendimiento en la mayoría de casos.

**Diccionario de datos**

