Universidad San Francisco de Quito

Data Mining

Proyecto 03

---

1) Resumen

Levantarás una **infraestructura con Docker Compose** que incluya **Jupyter+Spark**, **Snowflake**. Replicarás con Spark la ingesta del P2: **2015–2025** (**Yellow** y **Green**) del dataset **NYC TLC Trips** hacia **Snowflake** en un **esquema `raw`**. Luego construirás una **tabla analítica única (One Big Table, OBT)** en el esquema `analytics`, desde la cual responderás **20 preguntas de negocio** mediante Spark. Todas las **credenciales** y parámetros se manejarán con **variables de ambiente**.

Si se les acabó la cuenta gratis, creen otra con otro email.

---

2) Objetivos de aprendizaje

1. Operar **Spark en Jupyter** para ingesta masiva y transformación ligera de **Parquet**.
2. Diseñar un **aterrizaje RAW** y una **OBT** desnormalizada para analítica directa.
3. Practicar un **modelo alternativo** al dimensional (P2): **One Big Table**.
4. Gestionar **seguridad** y **reproducibilidad** con **Docker Compose** y **variables de ambiente**.
5. Implementar **controles de calidad**, **idempotencia** y **auditoría** de cargas.

---

3) Alcance y restricciones

- **Fuente**: NYC TLC Trip Record Data en **Parquet**.
- **Cobertura obligatoria**: **todos los meses 2015–2025** para **Yellow** y **Green** (si falta un Parquet, documéntalo).
- **Destino**: **Snowflake** con dos esquemas:
    - `raw`: aterrizaje espejo del origen (con metadatos de ingesta).

    - `analytics`: **`obt_trips`** (One Big Table) para consumo.
- **Procesamiento**: **Spark** (Jupyter) → Snowflake.
- **Infra**: **Docker Compose** con **spark-notebook**
- **Seguridad**: **variables de ambiente** para todas las credenciales y parámetros.

---

4) Requisitos técnicos

- Docker y Docker Compose.
- Contenedor **Jupyter+Spark** con conector JDBC a Snowflake.
- Conocimientos de Spark DataFrames, SQL, Jupyter y modelado analítico.

---

5) Arquitectura esperada (alto nivel)

- **spark-notebook**: ejecuta notebooks de ingesta y construcción OBT.
- **Snowflake**: almacena `raw` y `analytics`.
- **Flujo**: Parquet (2015–2025, Yellow/Green) → Spark (backfill mensual) → Snowflake `raw` → enriquecimiento/unificación → `analytics.obt_trips` (OBT).

---

6) Seguridad y variables de ambiente (obligatorio)

- Define **.env** y **.env.example** (este último sin credenciales reales).
- Variables mínimas:
    - **Snowflake**: host (nombre de servicio), puerto, base, usuario, contraseña, esquemas (`raw`, `analytics`).
    - **Parquet**: rutas/URL origen, años/meses/servicios a procesar.
    - **Parámetros**: chunk (mes), `RUN_ID`, flags de validación.
- **Prohibido** hardcodear credenciales en notebooks o Compose.

---

7) Conoce el dataset (guía mínima)

- **Yellow**: `tpep_pickup_datetime`, `tpep_dropoff_datetime`, `PULocationID`, `DOLocationID`, `passenger_count`, `trip_distance`, tarifas itemizadas, `payment_type`, `RatecodeID`, `VendorID`.
- **Green**: análogo con `lpep_*_datetime` y **`trip_type`** (1 street-hail, 2 dispatch).
- **Taxi Zone Lookup**: mapea `LocationID` → `zone` / `borough`.
- **service_type**: derivado del origen (yellow/green).

---

8) ¿Qué es One Big Table (OBT)? (definición y reglas del P3)

Definición.
One Big Table es un **modelo analítico desnormalizado** donde concentras en **una sola tabla** todas las columnas necesarias para responder preguntas de negocio: hechos, descriptores (antes en dimensiones), derivadas y metadatos. En este proyecto, esa tabla es `analytics.obt_trips`.

Objetivos: Maximizar la **simplicidad de consulta** (JOIN-free o JOIN-lite) y acelerar el prototipado/descubrimiento, sacrificando algo de normalización.

Buenas prácticas (obligatorias para este P3):

- **Grano**: 1 fila = 1 viaje (consistencia absoluta).
- **Desnormalización deliberada**: incluir nombres legibles (zona, borough, vendor, rate, payment), además de los IDs.
- **Derivadas documentadas**: `trip_duration_min`, `avg_speed_mph`, `tip_pct` con reglas de cálculo y manejo de nulos/ceros.
- **Metadatos de lineage**: `run_id`, `source_year`, `source_month`, `service_type`.
- **Idempotencia**: reingestar el mismo mes no duplica filas (define tu clave natural + estrategia de upsert).
- **Calidad mínima**: rangos lógicos (duración/distancia/montos), nulos en claves esenciales, consistencia de fechas PU/DO.
- **Indexación enfocada** (si decides crear índices): prioriza columnas típicas de filtro (fecha/hora, `pu_location_id`, `service_type`), y documenta el impacto medido.

Ventajas (por qué lo practicamos):

- **Simplicidad** para analistas/BI: consultas directas y rápidas de escribir.
- **Menos JOINs** ⇒ riesgo menor de errores de cardinalidad.
- **Rápido para iterar** hipótesis y validar métricas.

Limitaciones y mitigaciones:

- **Duplicación de datos** → más almacenamiento. *Mitiga*: columnas realmente útiles; evita redundancia innecesaria.
- **Actualizaciones/anomalías** (cambios en catálogos) → hay que **recalcular** OBT. *Mitiga*: notebooks reproducibles y "rebuild by partition" (por mes).
- **Escalabilidad**: OBT gigantes pueden requerir **particionado lógico** (por mes/año) y **vacuum/analyze** periódicos en Snowflake. *Mitiga*: plan de mantenimiento y segmentación por lotes.

Idea clave: OBT **no sustituye** al modelo dimensional del P2; es **otra técnica** útil para exploración rápida y workloads específicos.

---

9) Diseño de esquemas en Snowflake

9.1 Esquema `raw` (aterrizaje)

- **Tablas espejo** por servicio y/o partición lógica año/mes (elige y documenta).
- **Metadatos obligatorios**: `run_id`, `service_type`, `source_year`, `source_month`, `ingested_at_utc`, `source_path`, conteos por lote (en tabla de auditoría o reporte).
- **Idempotencia**: definiciones claras de clave natural (p. ej., timestamps + PU/DO + VendorID) y estrategia de no-duplicación.

9.2 Esquema `analytics` (One Big Table)

- **Tabla**: `analytics.obt_trips`.
- **Grano**: 1 fila = 1 viaje.
- **Columnas mínimas**:
  - **Tiempo**: `pickup_datetime`, `dropoff_datetime`, `pickup_date`, `pickup_hour`, `dropoff_date`, `dropoff_hour`, `day_of_week`, `month`, `year`.
  - **Ubicación**: `pu_location_id`, `pu_zone`, `pu_borough`, `do_location_id`, `do_zone`, `do_borough`.
  - **Servicio y códigos**: `service_type` (yellow/green), `vendor_id`, `vendor_name`, `rate_code_id`, `rate_code_desc`, `payment_type`, `payment_type_desc`, `trip_type` (green).
  - **Viaje**: `passenger_count`, `trip_distance`, `store_and_fwd_flag`.
  - **Tarifas**: `fare_amount`, `extra`, `mta_tax`, `tip_amount`, `tolls_amount`, `improvement_surcharge`, `congestion_surcharge`, `airport_fee`, `total_amount`.
  - **Derivadas**: `trip_duration_min`, `avg_speed_mph`, `tip_pct`.

- **Lineage/Calidad**: `run_id`, `ingested_at_utc`, `source_service`, `source_year`, `source_month`.

---

10) Notebooks (orden y propósito)

1. **01_ingesta_parquet_raw.ipynb**
   a. Lee Parquet **2015–2025** (Yellow/Green) **mes a mes**.
   b. Estandariza tipos y timestamps mínimos.
   c. Escribe hacia `raw` + registra **conteos por lote** y metadatos.
2. **02_enriquecimiento_y_unificacion.ipynb**
   a. Integra **Taxi Zones** (nombres/borough).
   b. Unifica yellow/green, normaliza catálogos (`payment_type`, `rate_code`, `vendor`).
3. **03_construccion_obt.ipynb**
   a. Ensambla `analytics.obt_trips` con derivadas y metadatos.
   b. Verifica **idempotencia** reingestando un mes.
4. **04_validaciones_y_exploracion.ipynb**
   a. Valida: nulos, rangos, coherencia de fechas, conteos por mes/servicio.
5. **05_data_analysis.ipynb:** Contesta las siguientes preguntas con la tabla `analytics.obt_trips` con Spark.
   a. Top 10 zonas de pickup por volumen mensual.
   b. Top 10 zonas de dropoff por volumen mensual.
   c. Evolución mensual de total_amount y tip_pct por borough.
   d. Ticket promedio (avg `total_amount`) por service_type y mes.
   e. Viajes por hora del día y día de semana (picos).
   f. p50/p90 de `trip_duration_min` por borough de pickup.
   g. `avg_speed_mph` por franja horaria (6–9, 17–20) y borough.
   h. Participación por `payment_type_desc` y su relación con `tip_pct`.
   i. ¿Qué `rate_code_desc` concentran mayor `trip_distance` y `total_amount`?
   j. Mix yellow vs green por mes y borough.
   k. Top 20 flujos PU→DO por volumen y su ticket promedio.
   l. Distribución de `passenger_count` y efecto en `total_amount`.
   m. Impacto de `tolls_amount` y `congestion_surcharge` por zona.
   n. Proporción de viajes cortos vs largos por borough y estacionalidad.
   o. Diferencias por vendor en `avg_speed_mph` y `trip_duration_min`.
   p. Relación método de pago ↔ `tip_amount` por hora.
   q. Zonas con percentil 99 de duración/distancia fuera de rango (posible congestión/eventos).

r. Yield por milla (`total_amount/trip_distance`) por borough y hora.
s. Cambios YoY en volumen y ticket promedio por service_type.
t. Días con alta `congestion_surcharge`: efecto en `total_amount` vs días "normales"

Todas las notebooks deben parametrizar **años/meses/servicios** y leer **variables de ambiente**.

---

11) Infraestructura con Docker Compose

- **Servicios**:
  - `spark-notebook`: Jupyter+Spark (puerto expuesto, volumen para notebooks).
- **Evidencias**: capturas de Spark ejecutandose en el puerto designado, al igual que Servidor Jupyter
- Ejemplo del docker (**no** esta completo):
  - services:
  - pyspark-notebook:
  - image: jupyter/pyspark-notebook:latest
  - container_name: pyspark-notebook
  - ports:
  - - "8888:8888"
  - - "4040:4040"
  - volumes:
  - - ./work:/home/jovyan/work
  - environment:
  - - SPARK_LOCAL_IP=0.0.0.0
  - - PYSPARK_PYTHON=python

---

12) Calidad, auditoría y performance

- **Calidad**: reglas mínimas (no nulos esenciales; distancias/duraciones ≥0; montos coherentes); documenta filas descartadas si filtras outliers.
- **Auditoría**: tabla o reporte con **conteos por servicio/año/mes**, tiempos de carga y `run_id`.
- **Performance**: documenta tiempos estimados de consultas típicas; si decides, crea **índices** focalizados (fecha, `pu_location_id`, `service_type`) y reporta el impacto.

---

13) Entregables (en GitHub)

1. **README** con:
    - Arquitectura (diagrama/tabla): Spark/Jupyter → Snowflake (`raw` → `analytics.obt_trips`).
    - **Matriz de cobertura 2015–2025** por servicio/mes (ok/falta/fallido).
    - Pasos para **Docker Compose** y ejecución de notebooks (orden y parámetros).
    - **Variables de ambiente**: listado y propósito; guía para `.env`.
    - Diseño de `raw` y **OBT** (columnas, derivadas, metadatos, supuestos).
    - **Calidad/auditoría**: qué se valida y dónde se ve.
2. **Carpeta `notebooks/`** con los 5 cuadernos limpios y ejecutables.
3. **Carpeta de evidencias**: capturas de Compose corriendo, servicios ejecutándose y conectados, conteos por lote, snapshot de OBT.

---

14) Rúbrica de evaluación (100 pts)

- **Infraestructura (20 pts)**: Compose con 1 servicio; red compartida; variables de ambiente bien usadas.
- **Ingesta RAW (25 pts)**: cobertura real **2015–2025** (Yellow/Green), metadatos por lote, idempotencia.
- **OBT (25 pts)**: diseño correcto (grano 1 viaje), desnormalización útil, derivadas claras, soporte integral a preguntas.
- **Calidad & Auditoría (15 pts)**: reglas aplicadas, conteos por lote, documentación.
- **README & Evidencias (15 pts)**: claridad, pasos reproducibles, matriz, capturas.

---

15) Checklist de aceptación (para tu README)

- ☐ Docker Compose levanta **Spark y Jupyter Notebook**.
- ☐ Todas las credenciales/parámetros provienen de **variables de ambiente** (`.env`).
- ☐ **Cobertura 2015–2025** (Yellow/Green) cargada en **`raw`** con **matriz** y **conteos** por lote.
- ☐ **`analytics.obt_trips`** creada con columnas mínimas, derivadas y metadatos.
- ☐ **Idempotencia** verificada reingestando al menos un mes.
- ☐ **Validaciones** básicas documentadas (nulos, rangos, coherencia).
- ☐ **20 preguntas** respondidas (texto) usando la OBT.
- ☐ README claro: pasos, variables, esquema, decisiones, troubleshooting.

---

16) Recomendaciones finales

- Parametriza notebooks (años/meses/servicios) y evita `SELECT *` en verificaciones.
- Estandariza nombres **antes** de consolidar en OBT.
- Mantén la OBT **enfocada** (columnas útiles, sin redundancia).
- Documenta **supuestos** y **derivadas** (cómo calculas `tip_pct`, duración, velocidad).
- Si notas cuellos, segmenta cargas por mes y considera índices focalizados (documenta impacto).