Universidad San Francisco de Quito

Data Mining

PSet 02

1) Resumen

Construir un **data pipeline orquestado 100% con Mage** que ingeste datos del dataset **NYC TLC Trip Record Data** (Yellow y Green), aterrice la capa **bronze** en **PostgreSQL** (Docker), estandarice y depure en **silver**, y modele un **esquema estrella** en **gold** usando **dbt ejecutado desde Mage**.

Además, deberás implementar **particionamiento declarativo en PostgreSQL**:

- **Hechos (`gold.fct_trips`)**: particionamiento **por rango** (RANGE) por fecha de pickup.
- **Dimensiones**: particionamiento **por hash** y **por lista** donde corresponda.

Finalmente, deberás configurar **triggers en Mage** para automatizar la ejecución end-to-end y responder **20 preguntas de negocio** usando **exclusivamente la capa gold** en un notebook.

Referencia oficial del dataset y diccionarios:
https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

---

2) Objetivos de aprendizaje

Al finalizar este PSet, el estudiante debe poder:

1. Orquestar un pipeline completo con **Mage** (ingesta → transformaciones → calidad) sobre **PostgreSQL**.
2. Implementar arquitectura de medallas (**bronze/silver/gold**) con **dbt dentro de Mage**, incluyendo documentación y tests.
3. Diseñar un **modelo dimensional en estrella** (1 fila = 1 viaje) con dimensiones conformadas.
4. Implementar y justificar **particionamiento en PostgreSQL**:
    - RANGE en hechos
    - HASH y LIST en dimensiones
    - evidenciar **partition pruning** con `EXPLAIN (ANALYZE, BUFFERS)`

5. Operar credenciales mediante **Mage Secrets** y variables en `.env`, sin exponer secretos en el repo.
6. Entregar un repositorio reproducible con evidencias, tests y un notebook de análisis basado solo en gold.

---

3) Alcance y restricciones

Fuente de datos

- NYC TLC Trip Record Data (página oficial).
- Dataset complementario obligatorio: **Taxi Zone Lookup** (para enriquecer zonas).

Cobertura de datos

- **Ingesta de datos 2022, 2023, 2024, 2025 completa (todos los meses)**: Yellow + Green.
- Debe existir columna `service_type` que diferencia si fue yellow o green

Formato y carga

- El aterrizaje de los debe quedar tabulado en Postgres.
- Si falta un mes o formato, debes documentarlo con una **tabla de cobertura** en el README (ver Sección 8).

Destino

- PostgreSQL en Docker.
- Esquemas obligatorios:
    - `bronze` (raw)
    - `silver` (curated)
    - `gold` (marts)

Orquestación (obligatoria)

- Mage debe orquestar:
    - ingesta a bronze
    - ejecución de dbt (silver y gold)
    - creación de particiones
    - ejecución de tests
- No se aceptan entregas que dependan de ejecutar dbt “a mano” fuera de Mage.

Seguridad (obligatoria)

- Credenciales solo en **Mage Secrets** y `.env`.
- `.env` real no se sube al repo (solo `.env.example`).

---

4) Requisitos técnicos

- Docker + Docker Compose
- PostgreSQL (contenedor)
- Mage (contenedor, proyecto versionado)
- dbt-postgres instalado **dentro del contenedor de Mage**
- GitHub repo con estructura y documentación

---

5) Arquitectura esperada

Bronze (raw) — PostgreSQL

- Tablas raw (sin lógica de negocio) que reflejan el origen.
- Deben incluir metadatos de ingesta mínimos por fila:
  - `ingest_ts`
  - `source_month` (YYYY-MM)
  - `service_type` (`yellow`/`green`)

Silver (curated)

- Estandarización y limpieza:
  - Tipificación correcta (timestamps, numerics)
  - Reglas de calidad mínimas (Sección 9)
  - Enriquecimiento con Taxi Zones (PU/DO)

Materialización obligatoria: `silver` como **views**.

Gold (marts)

- Modelo en estrella:
  - Hecho principal: `gold.fct_trips` (1 fila = 1 viaje)
  - Dimensiones conformadas: `dim_date`, `dim_zone`, `dim_service_type`, `dim_payment_type`, `dim_vendor` (si aplica)

Materialización obligatoria: `gold` como **tablas**.

---

6) Seguridad y acceso (obligatorio)

Secrets (Mage)

Todas las credenciales de Postgres y parámetros de conexión usados por pipelines y por dbt deben vivir en **Mage Secrets**.

Prohibido:

- Credenciales hardcodeadas en código.
- Subir `.env` real al repo.
- Guardar passwords en `profiles.yml` con valores explícitos.

Evidencia obligatoria:

- Capturas de pantalla de Mage Secrets con **nombres visibles** y **valores ocultos**.

`.env` y `.env.example`

- `.env.example` debe existir y estar versionado.
- `.env` debe estar en `.gitignore`.
- `.env` solo puede contener valores de infraestructura local (host/puerto/db), nunca secretos expuestos públicamente.

---

7) Conocer el dataset (antes de modelar)

- Yellow/Green incluyen típicamente:
  - pickup/dropoff datetime
  - PULocationID / DOLocationID
  - trip_distance, passenger_count
  - fare/tip/tolls/total
  - payment_type, RatecodeID, VendorID (según servicio y año)
- Taxi Zone Lookup incluye zonas y boroughs.

Granularidad objetivo en Gold: 1 fila = 1 viaje.

---

8) Ingesta con Mage → PostgreSQL (Bronze)

Debes implementar en Mage un pipeline llamado **`ingest_bronze`**.

8.1 Tabla de cobertura (obligatoria)

En el README debes incluir una tabla con:

- `year_month`
- `service_type`
- `status` (`loaded` / `missing` / `failed`)
- `row_count`

8.2 Backfill por mes (obligatorio)

- La ingesta debe ser **mensual** (chunking por año/mes).
- Debe ser **idempotente**: reejecutar un mes no debe duplicar filas.
  - Solución aceptada: `DELETE FROM bronze... WHERE source_month = ... AND service_type=...` antes de insertar.
  - Debe quedar evidenciado en el código y README.

8.3 Metadatos de ingesta (obligatorio)

Cada fila en la tabla de viajes bronze debe incluir:

- `ingest_ts` (timestamp de ejecución)
- `source_month` (YYYY-MM)
- `service_type`

8.4 Evidencia obligatoria

- Logs de ejecución en Mage del pipeline `ingest_bronze`.
- Query o tabla de auditoría con conteos por mes y servicio.

---

9) Transformaciones con dbt (medallion) — ejecutado desde Mage

Debes ejecutar dbt desde Mage con **2 pipelines** separados:

1. `dbt_build_silver`

- Ejecuta: `dbt run --select silver`

2. `dbt_build_gold`
- Ejecuta, en orden:
    1. scripts SQL de particionamiento (Sección 10)
    2. `dbt run --select gold`
3. `quality_checks`
- Ejecuta: `dbt test`
- Debe fallar si un test falla.

Reglas mínimas de limpieza en Silver (obligatorias)

- `pickup_ts` y `dropoff_ts` no nulos
- `pickup_ts <= dropoff_ts`
- `trip_distance >= 0`
- `total_amount >= 0`
- `PULocationID` y `DOLocationID` deben mapear a una zona válida (validado por tests)

Unificación de servicios (si aplica)

- Debes incluir `service_type` (`yellow`/`green`) en Silver y Gold.

---

10) Particionamiento en PostgreSQL (práctica obligatoria)

El particionamiento es parte del pipeline `dbt_build_gold` y debe implementarse con **declarative partitioning**.

10.1 Hechos — RANGE (obligatorio)

- `gold.fct_trips` debe ser:
    - `PARTITION BY RANGE (pickup_date)`
- Particiones **mensuales** para el rango cubierto (mínimo Q1 2024).

Evidencia obligatoria:

- salida `\d+ gold.fct_trips` mostrando `Partition key: RANGE`
- lista de particiones creadas

10.2 Dimensiones — HASH (obligatorio)

- `gold.dim_zone` debe ser:

    - `PARTITION BY HASH (zone_key)`
- Debe tener **4 particiones**.

Evidencia obligatoria:

- `\d+ gold.dim_zone` mostrando `Partition key: HASH`

10.3 Dimensiones — LIST (obligatorio)

Debes particionar por LIST al menos estas dos dimensiones:

- `gold.dim_service_type` particionada por `service_type` con particiones:
    - `yellow`, `green`
- `gold.dim_payment_type` particionada por `payment_type` con particiones:
    - `cash`, `card`, `other/unknown` (o el dominio que definas, pero debe estar documentado y testeado)

Evidencia obligatoria:

- `\d+` de ambas tablas mostrando `Partition key: LIST`

10.4 Evidencia de partition pruning (obligatorio)

En el README debes incluir:

- 2 consultas con `EXPLAIN (ANALYZE, BUFFERS)`:
    1. Filtro por mes en `fct_trips` (debe tocar 1 partición)
    2. Búsqueda por `zone_key` en `dim_zone`

Debe explicarse en 2–4 líneas qué evidencia muestra el pruning.

---

11) Calidad y documentación

Tests en dbt (obligatorios)

Mínimo:

- `unique` + `not_null`:
    - `fct_trips.trip_key`
    - `dim_zone.zone_key`
    - `dim_date.date_key`

- `relationships`:
  - `fct_trips.pu_zone_key` → `dim_zone.zone_key`
  - `fct_trips.do_zone_key` → `dim_zone.zone_key`
  - `fct_trips.pickup_date_key` → `dim_date.date_key`
- `accepted_values`:
  - `dim_service_type.service_type in ('yellow','green')`
  - `dim_payment_type.payment_type in (...)` (dominio documentado)

`dbt test` debe pasar ejecutado desde Mage (`quality_checks`).

Documentación mínima obligatoria

- README (Sección 12)
- Diagrama textual del flujo (bronze → silver → gold)
- Tabla de cobertura de meses

---

12) Entregables (GitHub)

12.1 README (obligatorio)

Debe contener, en este orden:

1. Arquitectura (bronze/silver/gold) + diagrama textual
2. Tabla de cobertura por mes y servicio
3. Cómo levantar el stack (`docker compose up`)
4. Nombres de Mage pipelines y qué hace cada uno
5. Nombres de triggers y qué disparan
6. Gestión de secretos: lista de secretos (nombres y propósito) sin valores
7. Particionamiento:
   - `\d+` evidencias
   - `EXPLAIN (ANALYZE, BUFFERS)` evidencias
8. dbt:
   - materializations
   - `dbt run` y `dbt test` logs (capturas o snippet)
9. Troubleshooting (mínimo 3 problemas comunes y solución)

12.2 Código y proyectos versionados

- Docker Compose
- Proyecto Mage (pipelines + triggers)

- Proyecto dbt (models bronze/silver/gold + tests)
- Scripts SQL de particionamiento
- Notebook `data_analysis.ipynb` con 20 preguntas (solo Gold)

12.3 Evidencias (capturas)

- Mage Secrets (valores ocultos)
- Triggers configurados
- Ejecución end-to-end (logs)
- Particiones y pruning
- `dbt test` passing

---

13) Rúbrica de evaluación (100 pts)

- Ingesta + cobertura + idempotencia (20 pts)
- Arquitectura de medallas + materializations (20 pts)
- Modelo estrella (20 pts)
- Particionamiento (RANGE + HASH + LIST) + pruning evidence (25 pts)
- Seguridad (Mage Secrets + `.env` correcto) (10 pts)
- Calidad + documentación + notebook (5 pts)

---

14) 20 preguntas de negocio (desde capa gold)

Debes responder en `data_analysis.ipynb` usando **solo tablas `gold.*`**.
Cada respuesta debe incluir:

- query SQL
- 1–3 líneas de interpretación
- tablas usadas (en un comentario o markdown)

Demanda / estacionalidad

1. Viajes por mes (2024): `count(*)` por `month`.
2. Viajes por `service_type` y mes.
3. Top 10 zonas de pickup (total 2024).
4. Top 10 zonas de dropoff (total 2024).
5. Top 5 boroughs por mes (pickup).
6. Horas pico (top 5 horas) para cada día de semana.
7. Distribución de viajes por día de semana (ranking).

Ingresos / tarifas / propinas

8. Ingreso total (`total_amount`) por mes.
9. Ingreso total por `service_type` y mes.
10. `tip %` promedio por mes (`avg(tip_amount / nullif(fare_amount,0))`).
11. `tip %` por borough y mes.
12. Top 10 zonas por ingreso total (pickup).
13. Top 10 zonas por tip % (pickup) con mínimo N viajes (define N y documenta).
14. Comparación cash vs card: viajes, ingreso total, tip %.

Duración / distancia / eficiencia

15. Duración promedio (min) por mes.
16. Distancia promedio por mes.
17. Velocidad promedio (mph) por borough y franja horaria.
18. Percentiles p50 y p90 de duración por borough.
19. Top 10 zonas (pickup) por p90 de duración.

Rutas / patrones

20. Top 10 rutas borough→borough (pickup borough to dropoff borough) por número de viajes.

---

15) Triggers en Mage (obligatorio)

Debes configurar **2 triggers mínimos**:

1. `ingest_monthly` (Schedule trigger)

- Ejecuta `ingest_bronze` con frecuencia diaria o semanal (documentar).

2. `dbt_after_ingest` (Event trigger/pipeline chaining)

- Dispara cuando `ingest_bronze` termina exitosamente.
- Ejecuta en orden:
    1. `dbt_build_silver`
    2. `dbt_build_gold`
    3. `quality_checks`

Evidencia obligatoria: capturas de los triggers y logs de una corrida completa.

---

16) Checklist de aceptación (copiar al README y marcar)

- [ ] Docker Compose levanta Postgres + Mage
- [ ] Credenciales en Mage Secrets y `.env` (solo `.env.example` en repo)
- [ ] Pipeline `ingest_bronze` mensual e idempotente + tabla de cobertura
- [ ] dbt corre dentro de Mage: `dbt_build_silver`, `dbt_build_gold`, `quality_checks`
- [ ] Silver materialized = views; Gold materialized = tables
- [ ] Gold tiene esquema estrella completo
- [ ] Particionamiento: RANGE en `fct_trips`, HASH en `dim_zone`, LIST en `dim_service_type` y `dim_payment_type`
- [ ] README incluye `\d+` y `EXPLAIN (ANALYZE, BUFFERS)` con pruning
- [ ] `dbt test` pasa desde Mage
- [ ] Notebook responde 20 preguntas usando solo `gold.*`
- [ ] Triggers configurados y evidenciados