Instacart: Data Architecture & Analytics

1. Preparación del entorno de desarrollo

1. **Entorno de trabajo**
    - Desarrollo en un ambiente en local.
    - Uso de ambientes virtuales para manejo de dependencias.
2. **Instalación de herramientas necesarias**
    - **Python 3.x** (Version necesaria para las dependencias)
    - **MySQL** (Servidor en local)
    - **Cliente de Snowflake** (Libreria de python)
    - **MageAI** (para la orquestación de la tubería de datos).
    - Cualquier librería adicional para análisis (pandas, numpy, matplotlib) y visualizaciones (seaborn, plotly) si se necesitan.
3. **Organización del proyecto**

Estructura del proyecto:

```
├── data/           # Carpeta donde se guardan temporalmente los CSVs
├── scripts/        # Scripts de Python para la carga inicial
├── notebooks/      # Notebooks para EDA e insights discovery
├── data_pipeline_engine/  # Configuración y archivos de MageAI
├── docs/           # Documentación del proyecto
├── README.md       # Descripción general
└── requirements.txt # Dependencias de Python
```

---

2. Ingesta de datos a MySQL (Script en Python)

1. **Descarga de los datasets**
    - Los estudiantes deberán descargar los archivos CSV proporcionados (instacart_orders.csv, products.csv, order_products.csv, aisles.csv, departments.csv) desde la plataforma D2L.
2. **Creación de la base de datos y tablas en MySQL**
    - Elaborar un **script en Python** que:
        - Se conecte a MySQL (mediante `pymysql`, `mysql-connector-python` u otra librería).
        - Crear la **base de datos** si no existe: “instacart_db”.
        - Cree las **tablas** necesarias (siguiendo la estructura de cada CSV).
        - Inserte los datos de cada CSV en la tabla correspondiente.

- Debe manejar:
    - Tipos de datos adecuados (INT, VARCHAR, FLOAT, etc.).
    - Posibles claves primarias, foráneas, etc.

3. **Ejecución del script**
    - Verificar que, **con un solo comando**, se ejecute todo el proceso de creación y carga de tablas, sin intervención adicional del usuario.

---

3. Creación de la tubería de datos a Snowflake con MageAI

1. **Configuración de MageAI**
    - Crear un nuevo proyecto de MageAI.
    - Ajustar la conexión a la base de datos MySQL (fuente) y la conexión a Snowflake (destino).
2. **Construcción de la tubería (ELT)**
    - Definir un pipeline donde se extraen los datos de MySQL y se cargan directamente a Snowflake en el **schema "RAW" en una base de datos "INSTACART_DB"**.
    - En esta etapa, se mueven los datos **tal cual** se encuentran en MySQL al entorno de Snowflake, sin transformaciones, con la finalidad de tener una "copia fiel" de los datos originales.
3. **Validaciones iniciales**
    - Asegurarse de que en Snowflake queden creadas las tablas de "RAW" con los mismos nombres y columnas que en MySQL
    - Verificar la consistencia en número de filas cargadas y ids presentes, data tests para verificar que ningun dato se perdio en la tuberia entre la fuente y raw.

---

4. Exploratory Data Analysis (EDA) en el schema RAW (Notebook)

1. **Creación de un Notebook**
    - Utilizar un Jupyter Notebook que se conecte a Snowflake que se llame: "eda.ipynb".
2. **Análisis exploratorio**
    - Inspeccionar cada tabla:
        - **Dimensión de los datos**: cuántas filas y columnas.
        - **Estadísticos descriptivos básicos**: promedio, conteo de valores únicos, minimos, maximos, desviacion estandar.

        - **Distribuciones**: histogramas para valores numéricos.
    - Identificación de **problemas de calidad**:
        - Datos ausentes (`NaN`, `NULL`).
        - Registros duplicados.
        - Inconsistencias en tipos de datos.
        - Posibles valores atípicos.
3. **Conclusiones y plan de acción**
    - Documentar los hallazgos de la data sucia o problemáticas detectadas.
    - Proponer un **plan de transformación y curación**:
        - Limpieza de valores ausentes (imputación, eliminación).
        - Eliminación o consolidación de duplicados.
        - Conversión de tipos de datos o normalización.

---

5. Diseño de la tubería de transformación y modelado dimensional (star-schema)

1. **Modelado dimensional**
    - Decidir cuáles tablas serán **dimensiones** (por ejemplo, `dim_products`, etc.) y cuál/es serán la(s) **tabla(s) de hechos** (`fact_orders`, etc.).
    - Definir claves primarias, foráneas y la relación en estrella (star-schema).
    - Planificar las columnas que se van a incluir en cada dimensión y cada hecho (por ejemplo: `order_id`, `user_id`, métricas, etc).
2. **Implementación de la transformación**
    - Crear una nueva tubería en MageAI o un flujo dentro de la misma, que:
        1. **Extraiga** datos del schema RAW.
        2. **Transforme** y limpie conforme al plan de acción del EDA:
            - Manejar valores faltantes.
            - Eliminar duplicados.
            - Generar columnas derivadas (por ejemplo, formato de fecha/hora).
            - Mapear a los tipos de datos correctos.
            - Modelar la tabla segun si es dimension o hecho (dimension o fact)
            - Cargar la tabla con el nombre correspondiente.
        3. **Cargue** los datos en las tablas finales del **schema CLEAN** en Snowflake, siguiendo el star-schema definido.
3. **Validación de la tabla final**
    - Revisar que el número de registros y las relaciones concuerden con lo esperado después de la limpieza.
    - Opcional: incluir métricas de calidad de datos (por ejemplo: cuántos registros fueron descartados, cuántos se corrigieron, etc.).

---

6. Análisis final para obtención de insights

Una vez los datos están en la capa **CLEAN** con un modelo dimensional, se debe realizar un Notebook llamado: “insights.ipynb” para responder a las siguientes preguntas:

1. **Comportamiento de compra según día de la semana**
    - Analizar la distribución de órdenes por cada día (0 = domingo, 1 = lunes, etc.).
2. **Comportamiento de compra según hora del día**
    - Evaluar la hora de las compras y ver la frecuencia por cada hora (0–23).
3. **Comportamiento según hora del día *y* día de la semana**
    - Cruzar las dos variables para ver si hay días en que la compra por horas difiera del patrón general.
4. **Distribución de las órdenes hechas por los clientes**
    - ¿Hay clientes que hacen más órdenes que otros? ¿Cuántas órdenes hace un cliente en promedio?
5. **Top 20 productos más frecuentes**
    - Contar la frecuencia con que aparecen los productos en las órdenes.
6. **¿Cuántos artículos se compran generalmente en un pedido?**
    - Distribución de la cantidad de artículos por orden.
7. **Top 20 artículos que se vuelven a pedir con más frecuencia**
    - Productos con mayor índice de reorder.
8. **Proporción de pedidos que se vuelven a pedir para cada producto**
    - Para cada producto, calcular cuántas veces es “reordenado” respecto al total de pedidos del mismo.
9. **Proporción de productos pedidos que se vuelven a pedir para cada cliente**
    - Cuántos productos vuelven a pedir los clientes en relación a la cantidad total de productos comprados por cliente.
10. **Top 20 artículos que la gente pone primero en el carrito**
    - Orden por la columna `add_to_cart_order` = 1 para identificar los productos más comunes en primera posición.

---

7. Presentación y documentación

1. **Codigo fuente:**
    - **Versionado**: Se debe usar Git/GitHub para el control de versiones del proyecto, subir el proyecto a un repositorio y publicar la URL en la tarea del D2L. Un unico repositorio para todo el proyecto.
        - Se revisara el ultimo commit hecho hasta antes de la hora de entrega
2. **Reporte final, como si fuera para los directivos de la empresa**

    - Elaborar una presentación con los principales hallazgos y conclusiones.
    - Destacar los puntos críticos del proyecto:
        - Cómo fue el proceso de limpieza y por qué.
        - Por qué se eligió un cierto diseño de star-schema.
        - Qué consideraciones de negocio influyeron en las transformaciones de datos.
    - Incluir gráficos y tablas que ilustren las respuestas a las preguntas anteriores.
3. **Buenas prácticas**
    - Mantener la **reproducibilidad** del proyecto:
        - Scripts claros y bien comentados.
        - Documentar cada paso del pipeline en MageAI.
        - Añadir un README con instrucciones claras para correr cada componente.

---

8. Puntos extras (opcional)

- **Seguridad**: Asegurar que las credenciales de MySQL y Snowflake se manejen con cuidado (idealmente con variables de entorno, no en texto plano, y con privilegios limitados).
- **Automatización**: En la medida de lo posible, configurar la tubería de MageAI para que se ejecute de manera automática (por schedule), simulando un entorno de producción.