Informe: Decisiones Tomadas y Consultas Realizadas

- **Nombre:** Jorge Pardo Cutillas
- **Tarea:** LMSG05

Ejercicio 1: Decisiones Tomadas para la Transformación

1. Limpieza de Datos

1. **Eliminación de duplicados:**
    - Razonamiento: Se identificaron registros duplicados en la fuente de datos original. Estos fueron eliminados para evitar inconsistencias en los resultados.
    - Implementación: Uso de comandos SQL como DELETE con cláusulas ROW_NUMBER() para identificar duplicados.
2. **Normalización de nombres de columnas:**
    - Razonamiento: Los nombres de las columnas no seguían una convención uniforme, lo que dificultaba la legibilidad y el uso.
    - Implementación: Cambio de nombres como "User ID" a user_id, siguiendo el estándar snake_case.
3. **Gestión de valores nulos:**
    - Razonamiento: Se encontraron valores nulos en columnas clave como email y phone_number. Esto podría afectar la calidad de los análisis.
    - Implementación: Sustitución de valores nulos por datos predeterminados o eliminación de registros dependiendo del caso.

2. Transformación de Datos

1. **Creación de nuevas columnas derivadas:**
    - Razonamiento: Se requirió calcular métricas adicionales como "ingresos totales" a partir de varias columnas relacionadas.
    - Implementación: Uso de expresiones SQL como SUM(column1 + column2) AS total_income.
2. **Normalización de datos categóricos:**
    - Razonamiento: Las categorías como "activo/inactivo" tenían variantes como "activo", "Act", "INACT".
    - Implementación: Uso de la función CASE en SQL para unificar estas categorías.
3. **Indexación de datos:**
    - Razonamiento: Mejorar el rendimiento de las consultas frecuentes sobre tablas grandes.

- Implementación: Creación de índices en columnas clave como user_id y order_id.

Ejercicio 2: Consultas Realizadas

1. Consulta para Filtrar Usuarios Activos

- **Propósito:** Identificar usuarios que se encuentran activos en el sistema.
- **Consulta:**

```
SELECT user_id, name, email
FROM users
WHERE status = 'activo';
```

2. Consulta para Calcular Ingresos Totales por Usuario

- **Propósito:** Obtener los ingresos totales generados por cada usuario.
- **Consulta:**

```
SELECT user_id, SUM(order_total) AS total_income
FROM orders
GROUP BY user_id
ORDER BY total_income DESC;
```

3. Consulta para Identificar Productos Más Vendidos

- **Propósito:** Listar los productos con mayores ventas.
- **Consulta:**

```
SELECT product_id, COUNT(*) AS sales_count
FROM order_items
GROUP BY product_id
ORDER BY sales_count DESC
LIMIT 10;
```

4. Consulta para Detectar Anomalías en los Datos

- **Propósito:** Identificar órdenes con valores inconsistentes.
- **Consulta:**

```
SELECT order_id, user_id, order_total
FROM orders
WHERE order_total < 0;
```