Analista de datos: Laura Yadira González Olivares
Fecha: 27 de Marzo de 2025

Sección 1: Preguntas Teóricas (11 puntos)

SQL (4 puntos)

1. **¿Cuál es la diferencia entre un JOIN y un UNION en SQL? Proporcione un ejemplo de cada uno. (1 punto)**

1.1 La principal diferencia entre estos operadores radica en que el JOIN mediante relaciones entre columnas de diferentes tablas cruza información por filas, mientas que el UNION combina los resultados de múltiples sentencias SELECT.

1.2 JOIN combina filas horizontalmente (añadiendo columnas), mientras que UNION combina filas verticalmente (añadiendo filas).

1.3 JOIN se utiliza para cruzar datos relacionados, mientras que UNION se utiliza para consolidar datos.

Ejemplo JOIN

Tablas de Ejemplo

- **Identificacion_1:**

Id	Nombre	Teléfonos
1	Jorge	3112013143
2	Liliana	3219589564
3	Edward	3108676354

- **Identificacion_2**

Id	Apellido	Email
1	Rojas	Jr.123@gmail.com
2	Camacho	Liliana24@yahoo.es
3	López	Edd007@gmail.com

Consulta

SELECT Identificacion_1.Nombre**,** Identificacion_2.Apellido **FROM** Identificacion_1 **INNER JOIN** Identificacion_2 **ON** Identificacion_1.id = Identificacion_2.id;

Resultado:

Nombre	Apellido
Jorge	Rojas
Liliana	Camacho
Edward	López

Ejemplo UNION

- **Clientes_AAA:**

Id	Nombre	Móvil
1	Jorge	3112013143
2	Liliana	3219589564
3	Edward	3108676354

- **Clientes_AA:**

Id	Nombre	Móvil
1	Camilo	3195242658
2	Daniel	3219574612
3	Pedro	3148542659

Consulta

SELECT Nombre, Movil **FROM** Clientes_AAA
UNION
SELECT Nombre, Movil **FROM** Clientes_AA

Resultado:

Nombre	Móvil
Jorge	3112013143
Liliana	3219589564
Edward	3108676354
Camilo	3195242658
Daniel	3219574612
Pedro	3148542659

2. Explique qué es una consulta CTE (Common Table Expression) y mencione un caso de uso. (1 punto)

   Una CTE es una forma de crear una "tabla temporal" dentro de una consulta, para hacerla más fácil de leer y manejar. También se entiende como una consulta más pequeña que se realiza antes de la consulta principal. A esta consulta se le asigna un nombre y luego se usa como si fuera una tabla en la consulta principal.

   Ejemplo

Producto ID	Nombre Producto	Precio
1	Camiseta	25
2	Pantalón	50
3	Zapatos	75
4	Gorra	15
5	Chaqueta	100

   Query

```
WITH Promedio Precios AS (
        SELECT AVG (Precio) AS Precio Promedio
        FROM Productos
)
SELECT NombreProducto, Precio
FROM Productos, PromedioPrecios
WHERE Precio > PrecioPromedio;
```

53

   Resultado:

Nombre Producto	Precio
Zapatos	75
Chaqueta	100

3. ¿Qué es la cláusula HAVING en SQL y en qué se diferencia de WHERE? (1 punto)

   HAVING te permite filtrar resultados que ya han sido agrupados y resumidos.

   Ejemplo

   SELECT Ciudad, SUM (Ventas)

   FROM Ventas

   GROUP BY Ciudad

   HAVING SUM (Ventas) > 1000;

   A diferencia del WHERE que filtra filas individuales antes de agrupar.

   Ejemplo

   SELECT * FROM Productos WHERE Precio > 50;

4. Ejercicio Practico – Tabla ventas

   Consulta

   SELECT cliente_id, SUM(monto) **AS total_ventas**
   FROM ventas
   GROUP BY cliente_id
   HAVING SUM (monto) > 250;

Python (3 puntos)

¿Cuál es la diferencia entre una lista y un diccionario en Python? Dé un ejemplo de cada uno. (1 punto)

Listas: es una colección ordenada de elementos, los elementos pueden ser de cualquier tipo de dato (números, cadenas, otros objetos, etc.).Los elementos se acceden mediante un índice numérico, que comienza en 0.Algunas características son: Ordenadas: los elementos tienen un orden específico. Mutables: puedes modificar los elementos de una lista después de su creación. Permiten duplicados: puedes tener elementos repetidos en una lista.

Diccionarios: es una colección de pares clave-valor. Las claves deben ser únicas e inmutables.Los valores pueden ser de cualquier tipo de dato, los elementos se acceden mediante sus claves.

Diferencias clave

- **Acceso a elementos**:
  - Listas: mediante índices numéricos.
  - Diccionarios: mediante claves.
- **Orden**:
  - Listas: ordenadas.
  - Diccionarios: no ordenados
- **Estructura**:
  - Listas: colección de elementos individuales.
  - Diccionarios: colección de pares clave-valor.
- **Casos de uso**:
  - Listas: para almacenar colecciones ordenadas de elementos.
  - Diccionarios: para almacenar datos relacionados mediante claves únicas.

Ejemplo – Listas

Query

lista compras = ["leche", "huevos", "pan", "manzanas"]

print(lista compras[0])="leche"

print(lista compras[2])= "pan"

Ejemplo Diccionario

directorio_telefonico = {"Juan": "123-456-7890", "María": "987-654-3210", "Pedro": "555-123-4567"}

Print(directorio_telefonico ["Juan"])="123-456-7890"

Print (directorio_telefonico ["María"])= "987-654-3210"

2. ¿Qué librerías en Python se utilizan comúnmente para análisis de datos? Mencione al menos tres. (1 punto)

NumPy para la manipulación numérica, Pandas para la gestión de datos tabulares, y Matplotlib para la visualización de los resultados.

3. ¿Qué salida generará el siguiente código en Python?
data = [5, 3, 9, 1]
Print (sorted (data) [-2])
5

Visualización de datos - Tableau (4 puntos)

1. ¿Cuál es la diferencia entre una **medida** y una **dimensión** en Tableau? (1 punto)

- Las dimensiones responden a la pregunta "¿qué?", mientras que las medidas responden a la pregunta "¿cuánto?".
- Las dimensiones proporcionan el contexto para el análisis, mientras que las medidas proporcionan los valores que se analizan.
- Las dimensiones son cualitativas, mientras que las medidas son cuantitativas.

2. ¿Qué tipo de gráfico es más adecuado para comparar porcentajes de una variable categórica? (1 punto)

   Gráfico de Barras: Permite comparar claramente las proporciones entre las diferentes categorías. Es útil tanto para un número pequeño como grande de categorías. Cada barra representa una categoría. La altura de la barra corresponde al porcentaje de esa categoría

3. ¿Cómo se puede crear un **filtro dinámico** en Tableau para que el usuario pueda seleccionar un rango de fechas? (1 punto)

   Se deben seguir varios pasos:

   3.1 Crear parámetros de fecha
   3.2 Crear un campo calculado
   3.3 Mostrar los controles de parámetros
   3.4 Usar el campo calculado como filtro

- Los parámetros de fecha permiten al usuario seleccionar las fechas de inicio y fin del rango.
- El campo calculado compara las fechas de los datos con los parámetros y devuelve "Verdadero" si la fecha está dentro del rango seleccionado.
- Al usar el campo calculado como filtro, solo se muestran los datos que cumplen con la condición de "Verdadero".

4. ¿Qué es un archivo **hyper** en tableau, y en qué caso de uso se puede aplicar? (1 punto)

   Hyper es la tecnología subyacente que impulsa el motor de datos de Tableau. Fue introducido para mejorar el rendimiento de las extracciones de datos, permitiendo un procesamiento más rápido y eficiente de grandes conjuntos de datos.

   Caso de uso: Análisis de grandes conjuntos de datos
   Cuando se trabaja con volúmenes masivos de datos, Hyper mejora significativamente el rendimiento en comparación con los archivos ".tde" anteriores.

Sección 2: Prueba práctica SQL (55 puntos)

```
SELECT departamento, AVG (salario) AS promedio_salario
FROM empleados
WHERE fecha_contratacion < '2020-01-01'
GROUP BY departamento;
```

Resultados

departamento	promedio_salario
IT	6500

2.2 Escribe una consulta para obtener los 5 clientes con mayor monto total de ventas en los últimos 6 meses. (5 puntos)

```
SELECT c.nombre, c.apellido, SUM (v.monto) AS total_ventas

FROM clientes c

JOIN ventas v ON c.id = v.cliente_id

WHERE v.fecha >= CURRENT_DATE - INTERVAL '6 months'

GROUP BY c.id

ORDER BY total_ventas DESC

LIMIT 5;
```

2.3 Escribe una consulta para obtener el nombre completo de los clientes y su monto total de ventas.

```
SELECT c.nombre, c.apellido, SUM(v.monto) AS mt_ventas

FROM clientes c

JOIN ventas v ON c.id = v.cliente_id

GROUP BY c.id;
```

2.4 Escribe una consulta para obtener el ingreso promedio de ventas por mes. (10 puntos)

```
SELECT DATE_TRUNC ('month', v.fecha) AS mes, AVG (v.monto) AS promedio ventas

FROM ventas v

GROUP BY mes

ORDER BY mes;
```

2.5 Escribe una consulta para calcular el ranking de clientes por ventas en el último año.

```
SELECT

    c.nombre,

    c.apellido,

    SUM(v.monto) AS total_ventas,

    RANK() OVER (ORDER BY SUM(v.monto) DESC) AS ranking_ventas

FROM clientes c

JOIN ventas v ON c.id = v.cliente_id

WHERE v.fecha >= CURRENT_DATE - INTERVAL '1 year'

GROUP BY c.id

ORDER BY total_ventas DESC;
```

2.6 Escribe una consulta para calcular el total de ventas por cliente y luego selecciona solo los clientes cuyo total de ventas es superior al promedio general.

```
SELECT

    c.nombre,

    c.apellido,

    SUM(v.monto) AS total_ventas

FROM clientes c

JOIN ventas v ON c.id = v.cliente_id

GROUP BY c.id

HAVING SUM(v.monto) > (

    SELECT AVG(total_ventas)

    FROM (

        SELECT SUM (v.monto) AS total_ventas

        FROM ventas v

        GROUP BY v.cliente_id

    ) AS v.procliente

);
```

Sección 3: Python (15 puntos)

Resultado

Lea el archivo CSV.

```
Import csv

def leer_datos_ventas (nombre archivo):

    datos_ventas = []

    try:
        with open(nombre_archivo, 'r', newline='', encoding='utf-8') as archivo_csv:
            lector_csv = csv.DictReader(archivo_csv)
            for fila in lector_csv:
                datos_ventas.append(fila)

    return datos_ventas
```

Calcule el total de ventas por producto.

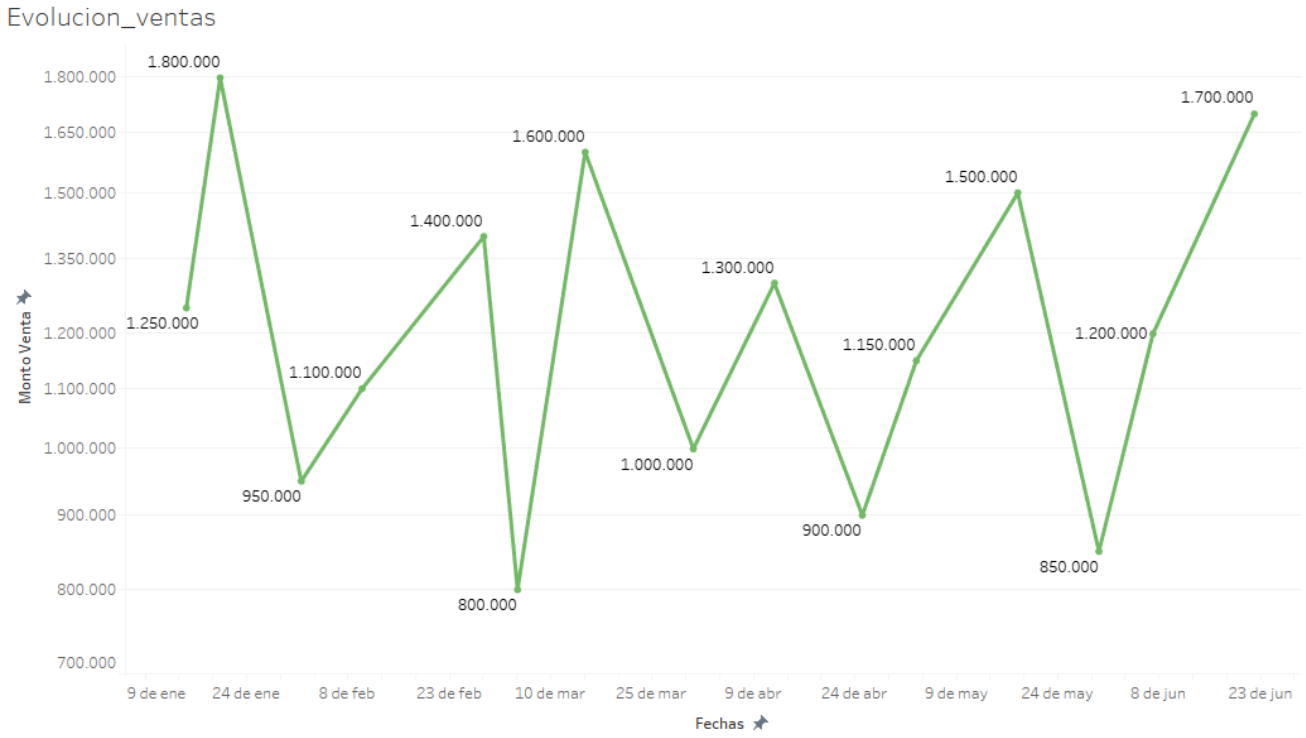
```
import csv

def calcular_ventas_producto(nombre_archivo):
    ventas_por_producto = {}
    with open(nombre_archivo, 'r') as archivo:
        lector = csv.DictReader(archivo)
        for fila in lector:
            producto = fila ['producto']
            cantidad = int(fila['cantidad'])
            precio = float(fila['precio_unitario'])
            total = cantidad * precio
            if producto in ventas_por_producto:
                ventas_por_producto [producto] += total
            else:
                ventas_por_producto[producto] = total
    return ventas_por_producto
```

Genere un archivo resumen_ventas.csv con los resultados.

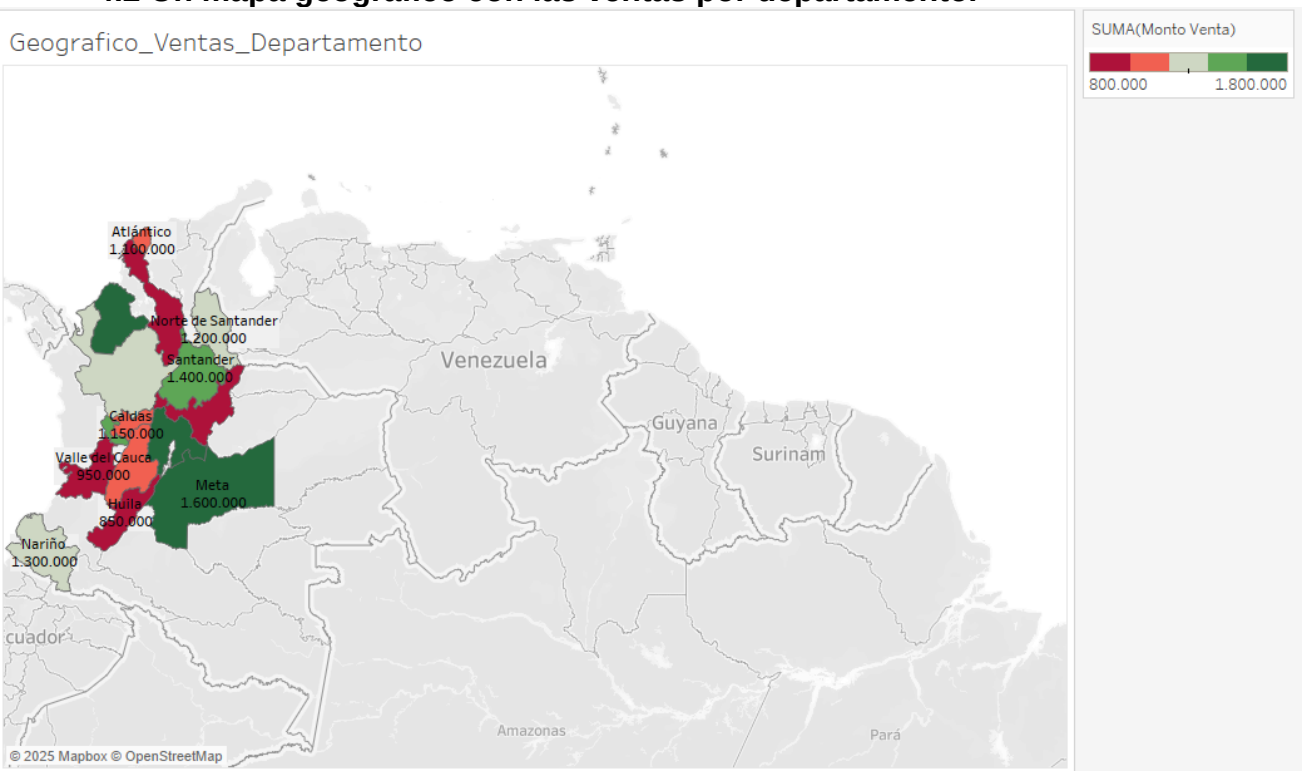
```
def guardar_resumen(ventas, nombre_archivo_resumen):
                    with open(nombre_archivo_resumen, 'w', newline='') as archivo_resumen:
                                escritor = csv.writer(archivo_resumen)
                                escritor.writerow(['Producto', 'Total Ventas'])
                                for producto, total in ventas.items():
                                            escritor.writerow([producto, total])
ventas = calcular_ventas_producto('datos_ventas.csv')
guardar_resumen(ventas, 'resumen_ventas.csv')
```

Sección 4: Visualización de datos - Tableau (30 puntos)

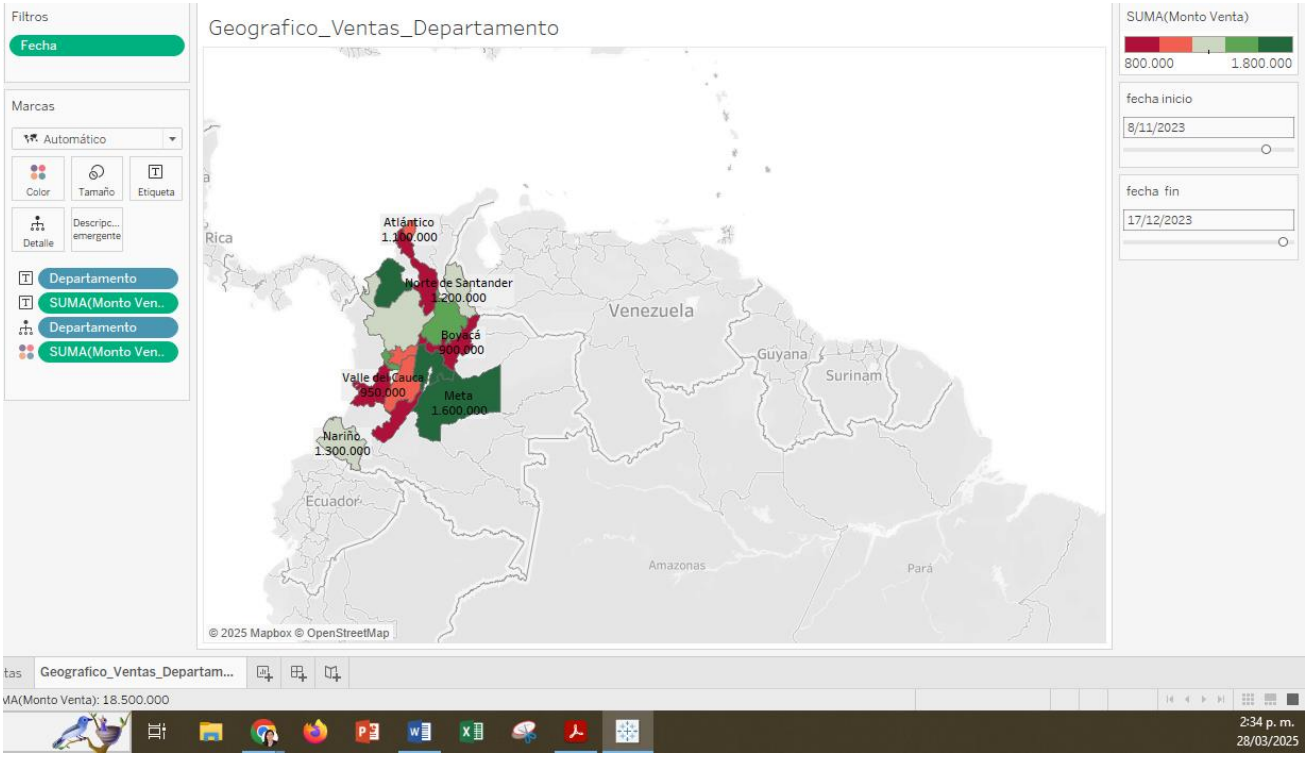
4.1 Un gráfico de líneas mostrando la evolución de las ventas en el tiempo.



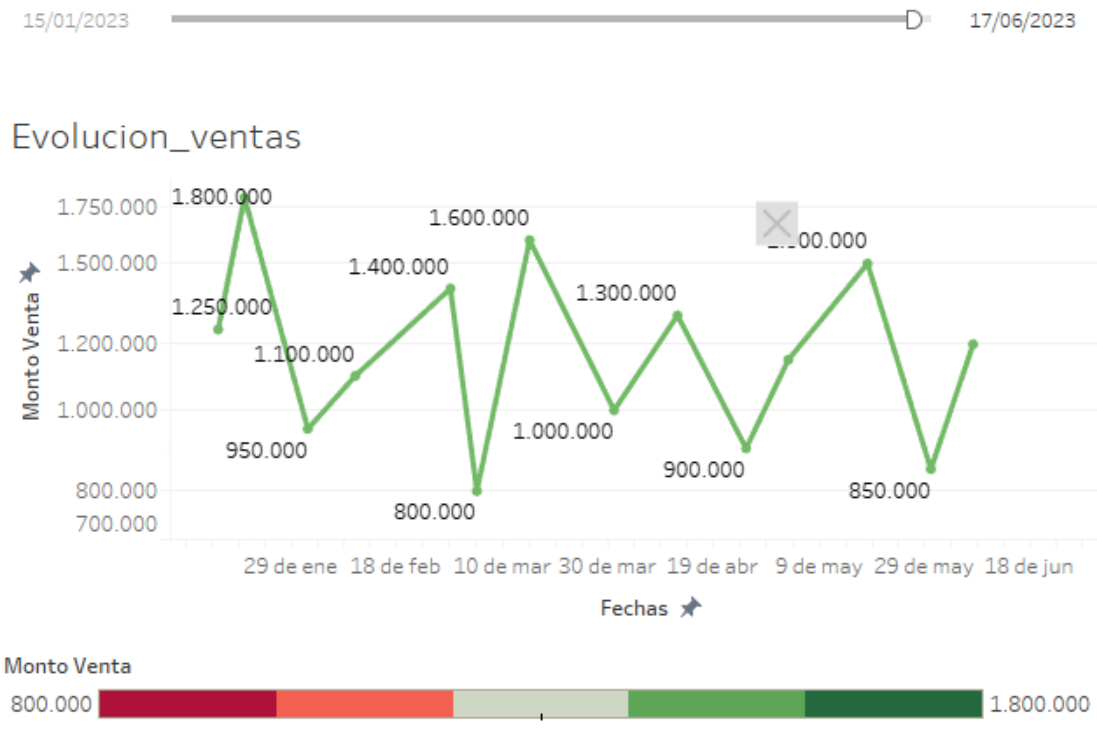
4.2 Un mapa geográfico con las ventas por departamento.

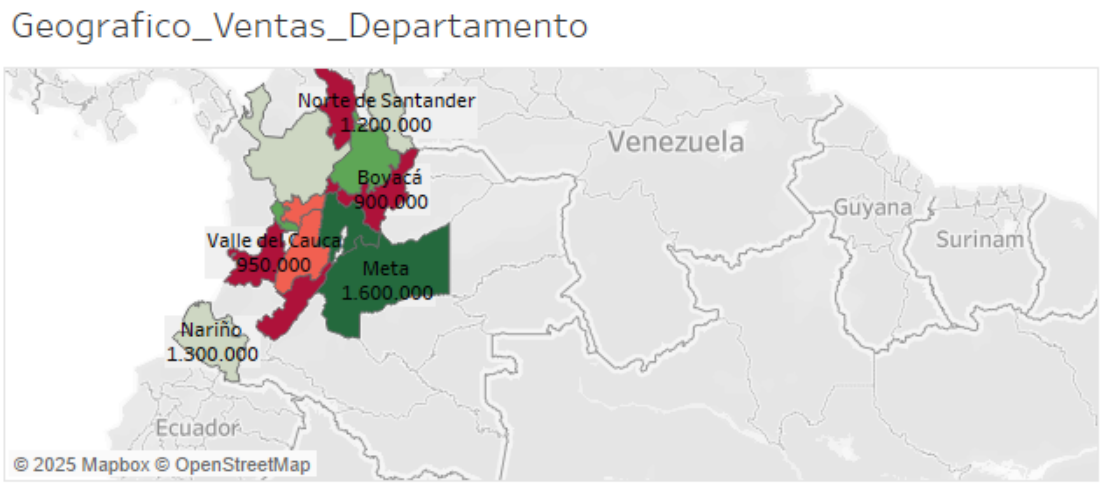


4.3 Un filtro interactivo para seleccionar un rango de fechas.



Dashboard Final





Dashboard- Final