Bancolombia

Solución Prueba Técnica Rol Analítico Cumplimiento

Autor

Cristian David Salcedo

Fecha

14/10/2024

Cristian David Salcedo
crsalced@bancolombia.com.co

Grupo Bancolombia

1. Modelo de Datos:

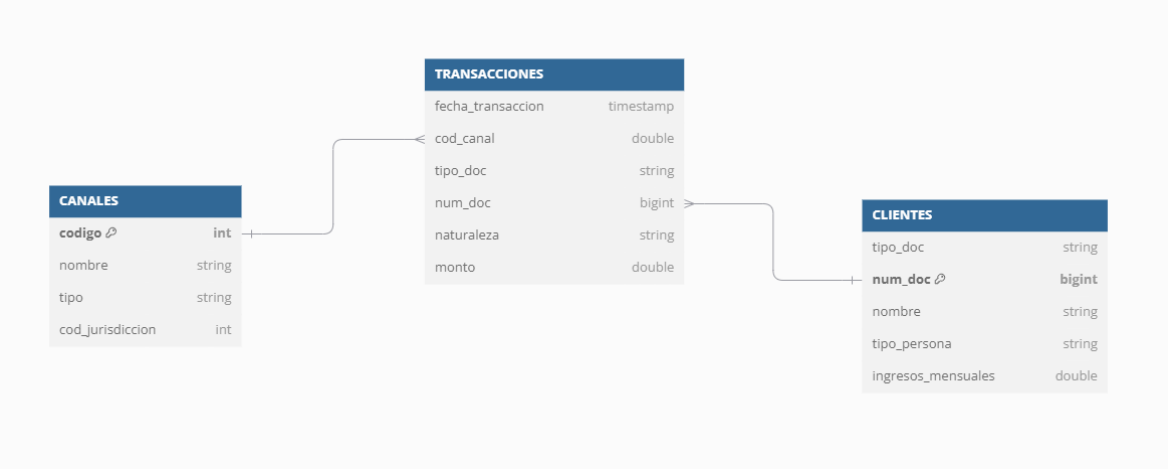

Ilustración 1. Modelo relacional.

El modelo este compuesto por 3 tablas Clientes, Canales y transacciones siguiendo una estructura relacional donde las tablas están conectadas mediante sus llaves, las relaciones existentes son:
Clientes a Transacciones: (Relación de 1 a muchos) cada cliente puede tener varias transacciones, pero una transacción solo está asociada a un cliente.
Canales a Transacciones: (Relación de 1 a muchos) cada canal puede tener varias transacciones asociadas a él, pero cada transacción ocurre en un único canal.

2. Implementación del modelo en Impala SQL:

Se comienza cargando los archivos CSV en el motor de bases de datos seleccionado, en este caso Impala SQL, mediante Sparky, utilizando el orquestador 2.0:

```python
class SubirBases(Step):
    def ejecutar(self):

        directorio_etl = os.path.dirname(os.path.abspath(__file__))

        file_1 = os.path.join(directorio_etl, '..','..', 'CANALES.csv')
        base_1 = pd.read_csv( file_1, header = 0)
        sp = self.getSparky()
        sp.subir_df(df = base_1, nombre_tabla='canales',zona='proceso_canales')

        file_2 = os.path.join(directorio_etl, '..','..', 'CLIENTES.csv')
        base_2 = pd.read_csv( file_2,header = 0)
        sp.subir_df(df = base_2, nombre_tabla='clientes',zona='proceso_canales')

        file_3 = os.path.join(directorio_etl, '..','..', 'TRANSACCIONES.csv')
        base_3 = pd.read_csv( file_3,header = 0)
        sp.subir_df(df = base_3, nombre_tabla='transacciones',zona='proceso_canales')

        file_4 = os.path.join(directorio_etl, '..','..', 'DIVIPOLA_Municipios.xslx')
        base_4 = pd.read_excel( file_4)
        sp = self.getSparky()
        sp.subir_df(df = base_4, nombre_tabla='base_municipios_col',zona='proceso_canales')
```

Ilustración 2. Función para subir las fuentes a la LZ

Para la construcción del modelo se utiliza Impala SQL. Lo primero es evaluar la calidad de los datos. En la tabla de clientes se observó más de un registro por cliente. Estos registros duplicados pueden generar errores en el futuro, por lo que el criterio de eliminación fue conservar, para cada cliente, el registro con el mayor ingreso mensual reportado:

```sql
DROP TABLE PROCESO_CANALES.crsalced_data_clientes PURGE;
CREATE TABLE PROCESO_CANALES.crsalced_data_clientes STORED AS PARQUET AS
WITH
S0 AS (
SELECT  tipo_doc,
        num_doc,
        nombre,
        tipo_persona,
        ingresos_mensuales,
        COUNT(*) OVER(PARTITION BY num_doc ORDER BY ingresos_mensuales DESC ) AS RN
FROM proceso_canales.clientes)
SELECT tipo_doc,
        num_doc,
        nombre,
        tipo_persona,
        ingresos_mensuales
FROM S0
WHERE  RN = 1;
```

Ilustración 3. Creación de la tabla con clientes únicos

Ya con clientes únicos se procedió a realizar la unión de la tabla de transacciones con clientes y canales:

```
DROP TABLE proceso_canales.crsalced_union_bases PURGE;
CREATE TABLE proceso_canales.crsalced_union_bases STORED AS PARQUET AS
SELECT   T1.fecha_transaccion,
         T1.cod_canal,
         T1.tipo_doc,
         T1.num_doc,
         T1.naturaleza,
         T1.monto,
         CASE WHEN T2.tipo_persona = 'NATURAL' THEN 'PERSONA NATURAL' ELSE T2.tipo_persona END AS tipo_persona,
         T2.ingresos_mensuales,
         T3.tipo,
         T3.cod_jurisdiccion
FROM proceso_canales.transacciones AS T1
INNER JOIN PROCESO_CANALES.crsalced_data_clientes AS T2
ON T1.num_doc = T2.num_doc
LEFT JOIN proceso_canales.canales AS T3
ON T1.cod_canal = T3.codigo ;
```

Ilustración 4. Unión de las bases y transformaciones.

Adicionalmente se creó una transformación para la columna “tipo_persona”, con esto ya se tiene la base final.

3. Análisis descriptivo de la información (python):

1. Como primer paso se importan las librerías necesarias para este análisis:

1. Importar Librerias

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from helper import Helper
import seaborn as sns
import openpyxl
```

Ilustración 5. Importar librerías

2. Por medio de impala helper se realiza la conexión a la LZ para traer los datos:

```
consulta= """
                SELECT *
                FROM proceso_canales.crsalced_union_bases
          """
df_o = h.obtener_dataframe(consulta)
```

Ilustración 6. Conexión a la LZ

3. Se realiza una visualización inicial de los datos previamente procesados y unidos:

```
df_o.head(10)
```

	fecha_transaccion	cod_canal	tipo_doc	num_doc	naturaleza	monto	tipo_persona	ingresos_mensuales	tipo	cod_jurisdiccion
0	2024-07-06 05:00:00	1000331.0	CEDULA DE CIUDADANIA	-9223285194817728665	ENTRADA	1500000.0	PERSONA NATURAL	1742864.0	SUCURSAL	5001.0
1	2024-07-06 05:00:00	1000331.0	CEDULA DE CIUDADANIA	-9223285194817728665	ENTRADA	2500000.0	PERSONA NATURAL	1742864.0	SUCURSAL	5001.0
2	2024-07-23 05:00:00	2069703.0	CEDULA DE CIUDADANIA	-9223285194817728665	ENTRADA	550000.0	PERSONA NATURAL	1742864.0	CORRESPONSAL	54405.0
3	2024-09-20 05:00:00	2023594.0	CEDULA DE CIUDADANIA	-9223285194817728665	ENTRADA	2000000.0	PERSONA NATURAL	1742864.0	CORRESPONSAL	76001.0
4	2024-09-03 05:00:00	2028672.0	CEDULA DE CIUDADANIA	-9223285194817728665	ENTRADA	3000000.0	PERSONA NATURAL	1742864.0	CORRESPONSAL	76001.0
5	2024-09-03 05:00:00	2028672.0	CEDULA DE CIUDADANIA	-9223285194817728665	ENTRADA	3000000.0	PERSONA NATURAL	1742864.0	CORRESPONSAL	76001.0

Ilustración 7. Vistazo general del DataFrame.

4. Gráficamente se observa cómo están distribuidas las variables del modelo de datos:

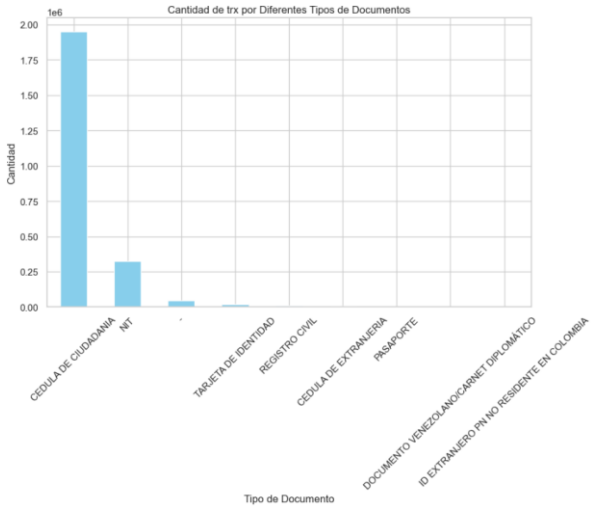

Ilustración 8. Distribución transaccional por Tipo de Documento

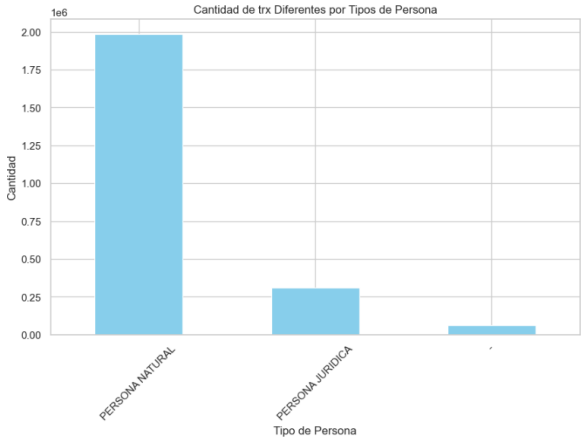

Ilustración 9. Distribución transaccional por Tipo de Persona

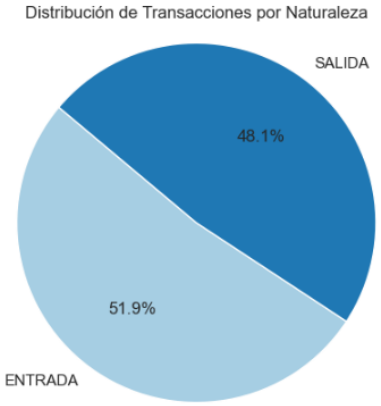

Ilustración 10. Distribución de la naturaleza de los montos transados

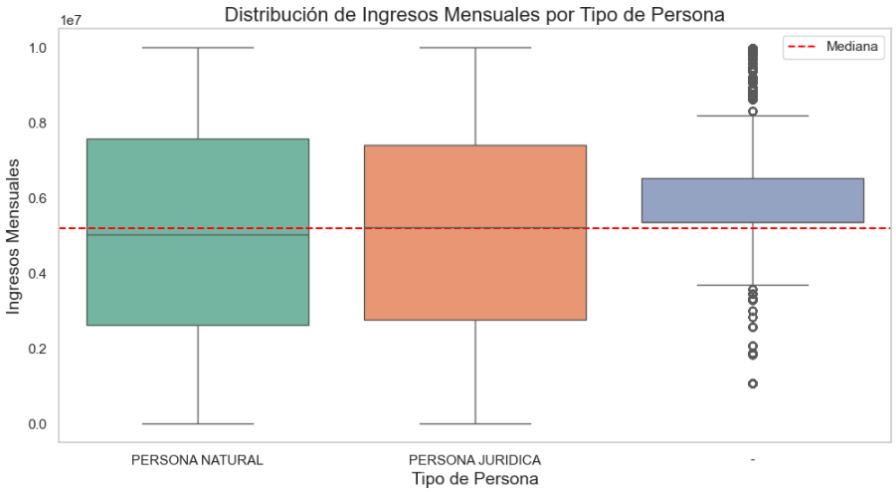

Ilustración 11. Distribuciones de los ingresos mensuales por cada tipo de Persona

4. Tablero en Power BI:

Para la construcción del tablero, se hace uso del modelo de datos y adicional se obtiene información del *Geoportal del DANE - Codificación Divipola*, para obtener las coordenadas, nombre municipio y departamento de los canales físicos, esta información se descarga del portal en formato .xlsx y se lleva a la LZ para realizar los cruces :

```
-------------------------------------------------------------------------
-- UNION DEL MODELO ANTERIOR CON LA INFORMACION DEL GEO-PORTAL DANE
-------------------------------------------------------------------------
DROP TABLE PROCESO_CANALES.CRSALCED_BASE_FINAL PURGE;
CREATE TABLE PROCESO_CANALES.CRSALCED_BASE_FINAL STORED AS PARQUET AS ;
WITH
UBI AS (
        SELECT *
        FROM  proceso_canales.base_municipios_col)
SELECT *
FROM proceso_canales.crsalced_union_bases   T1
LEFT JOIN UBI T2
ON CAST(T1.cod_jurisdiccion AS BIGINT) = CAST(T2.codigo AS BIGINT);
```

Ilustración 12. cruce de la información del modelo con los datos del Geoportal DANE

Para la construcción de la georreferenciación se usa el visual de la herramienta ArcGIS Maps, usando los campos, longitud y latitud previamente hallados:

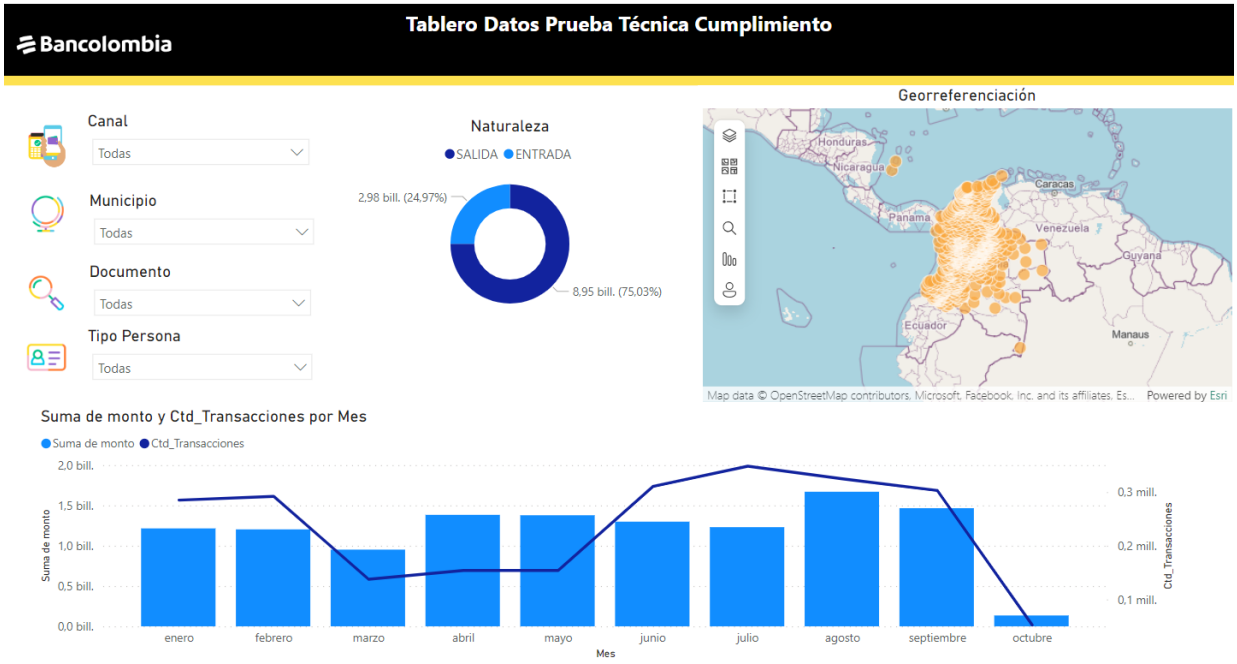

Ilustración 13. Vista del tablero en Power BI

5. Reporte:

Para la construcción del reporte, se procedió a agrupar y sumar el monto transado de cada cliente en los últimos 6 meses, adición se obtuvo el campo 'Canales_Usados' el cual concatena los canales que el cliente uso, todo esto se construyó en una expresión de tabla común (S0):

```
WITH
S0 AS (
        SELECT tipo_persona,
               num_doc,
               ingresos_mensuales,
               GROUP_CONCAT(DISTINCT TIPO," | ") AS canales_usados,
               SUM(monto) AS Suma_monto
        FROM proceso_canales.crsalced_union_bases
        WHERE fecha_transaccion >= ADD_months (NOW(),-6)
        GROUP BY 1,2,3),
```

Para obtener los percentiles se hizo uso de la función de ventana NTILE la cual nos permite dividir los datos en N grupos, se realiza el filtro de la primer condición:

```
S1 AS (
            SELECT tipo_persona,
                   num_doc,
                   ingresos_mensuales,
                   canales_usados,
                   Suma_monto,
                   NTILE(100) OVER (PARTITION BY tipo_persona ORDER BY ingresos_mensuales) AS percentil
            FROM S0
            WHERE suma_monto >= (ingresos_mensuales * 2))
```

Finalmente se realiza la proyección de los campos que cumplen la segunda condición:

```
SELECT  tipo_persona,
        num_doc,
        suma_monto,
        ingresos_mensuales,
        canales_usados
FROM S1
WHERE percentil > 95;
```


6. Conclusiones:

- La falta en la calidad de los datos, en especial en la base de clientes, con valores duplicados, puede generar errores, dado que para un mismo número de documentos se tenían diferentes tipos de documento, en especial atención, cuando era '-'.
- Se observo que los ingresos mensuales para clientes sin tipo de documento son muy elevados, como se ve en la gráfica número 11.
- Al incluir la georreferenciación, se pudo observar las zonas con alta densidad de canales físicos, esto resulta útil, para conocer mas a fondo a los clientes.