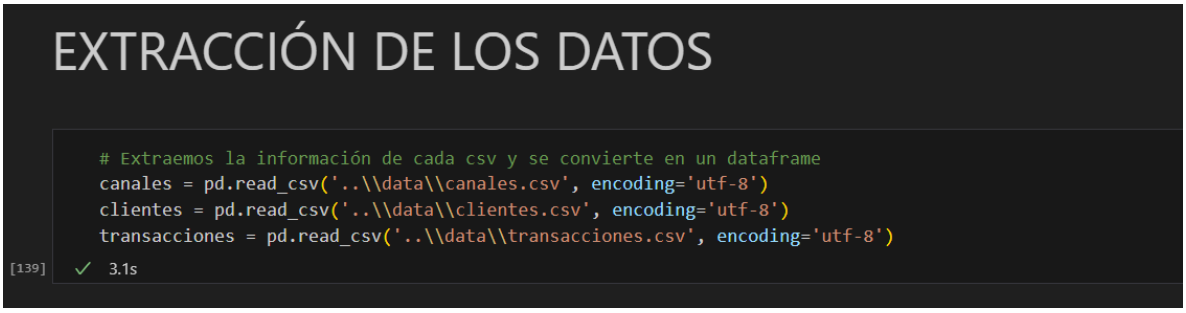
Prueba técnica Bancolombia: Ingeniero de datos Andrés Camilo Gómez Orozco

PROCESO DE ETL

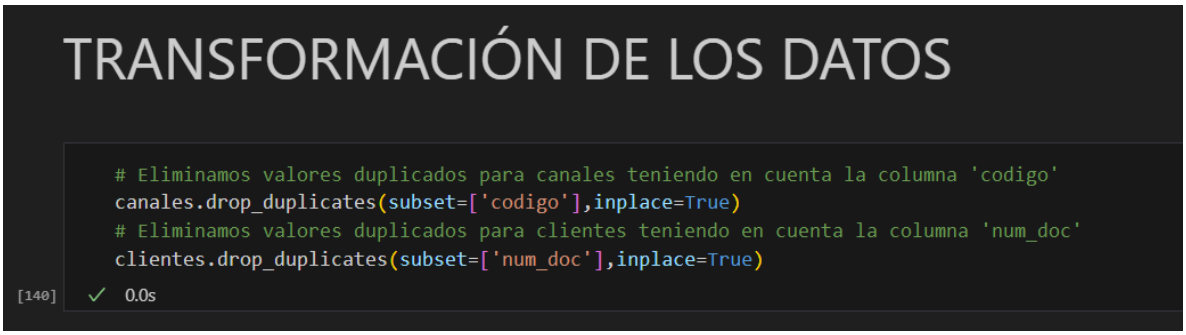
Extracción de los datos

El primer paso para la extracción de los datos, fue descargar los datos de kaggle, luego de tener los archivos .csv se hizo la extracción de ellos y se convierten en dataframes con la librería pandas para el manejo de datos.



Transformación de los datos

En la transformación de los datos primeramente hacemos una limpieza de los valores duplicados, en el caso del dataframe de canales no se deberían tener valores duplicados en la columna de 'codigo', y para el dataframe de clientes no se deberían tener valores duplicados en la columna 'num_doc'.



Lo próximo que se hizo en el proceso de transformación de los datos fue cambiar el tipo de dato a el indicado para subirlo a la base de datos, la columna 'fecha_transacción' se convirtió a tipo de dato datetime. Las columnas que se cargaron como Object ('tipo_doc', 'naturaleza', 'nombre', 'tipo_persona', 'tipo') se pasaron a tipo de dato string. Algunos de las columnas que se cargaron como tipo de dato numérico ('num_doc', 'cod_canal', 'cod_jurisdiccion', 'codigo') se pasaron a tipo de dato string.

Nota: La columna 'cod canal primero se convirtió a tipo de dato int y luego a tipo de dato string', debido a que al extraer los datos se cargó como tipo de dato double, por ende, al convertirse

directamente a tipo de dato string, el valor quedaba con un decimal extra y al hacer la comparación con la columna ‘cod_canal’ de la tabla canales no coincidía, ya que ese valor se cargo como tipo de dato int y no quedaba con decimales extra al convertirse a string.

```
# Convertimos la columna de 'fecha_transaccion' de Object a datetime
transacciones['fecha_transaccion'] = pd.to_datetime(transacciones['fecha_transaccion'])

# Convertimos los columnas de datos correspondientes de Object a string
transacciones['tipo_doc'] = transacciones['tipo_doc'].astype("string")
transacciones['naturaleza'] = transacciones['naturaleza'].astype("string")
clientes['tipo_doc'] = clientes['tipo_doc'].astype("string")
clientes['nombre'] = clientes['nombre'].astype("string")
clientes['tipo_persona'] = clientes['tipo_persona'].astype("string")
canales['nombre'] = canales['nombre'].astype("string")
canales['tipo'] = canales['tipo'].astype("string")

# Convertimos la columna de double a int para eliminar los decimales
transacciones['cod_canal'] = transacciones['cod_canal'].astype(int)

# Convertimos las columnas que se pueden convertir numero a string
transacciones['num_doc'] = transacciones['num_doc'].astype("string")
transacciones['cod_canal'] = transacciones['cod_canal'].astype("string")
clientes['num_doc'] = clientes['num_doc'].astype("string")
canales['cod_jurisdiccion'] = canales['cod_jurisdiccion'].astype("string")
canales['codigo'] = canales['codigo'].astype("string")
```

Carga de los datos

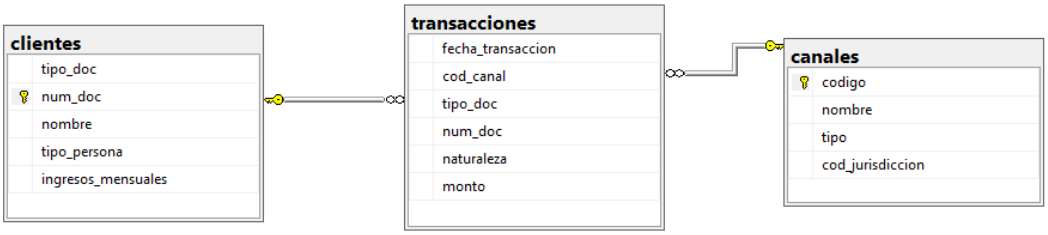
Inicialmente para este proceso de carga de datos creamos una base de datos SQL server y creamos las tablas previas a subir los datos, con el siguiente query:

```
CREATE TABLE clientes (
    tipo_doc VARCHAR(50),
    num_doc VARCHAR(50) PRIMARY KEY,
    nombre VARCHAR(255),
    tipo_persona VARCHAR(50),
    ingresos_mensuales DECIMAL,
);

CREATE TABLE canales (
    codigo VARCHAR(50) PRIMARY KEY,
    nombre VARCHAR(255),
    tipo VARCHAR(50),
    cod_jurisdiccion VARCHAR(50)
);

CREATE TABLE transacciones (
    fecha_transaccion DATETIME,
    cod_canal VARCHAR(50),
    tipo_doc VARCHAR(50),
    num_doc VARCHAR(50),
    naturaleza VARCHAR(50),
    monto DECIMAL,
    FOREIGN KEY (num_doc) REFERENCES clientes (num_doc),
    FOREIGN KEY (cod_canal) REFERENCES canales (codigo)
);
```

El diagrama de relación de las tablas quedaría de la siguiente forma:



Luego se inició con el proceso de carga de los datos a la base de datos, en este caso se usó la biblioteca sqlalchemy. Como era un servidor local en mi computadora lo único que hice para indicar la ruta del servidor fue indicar el nombre del equipo, en este caso el server, y el nombre de la database, y finalmente se indica el driver usado para la conexión.

CARGA DE LOS DATOS

```
import pyodbc
from sqlalchemy import create_engine

server = 'LAPTOP-E360B96B\SQLEXPRESS'
database = 'Prueba_bancolombia'

engine = create_engine(f"mssql+pyodbc://{server}/{database}?driver=ODBC+Driver+17+for+SQL+Server")
```

✓ 0.0s

Finalmente, para terminar el proceso de carga subimos los dataframes a la base de datos convirtiéndolos directamente en tablas SQL, dándoles como argumento ‘append’ para que se añadan los valores en las tablas ya creadas en la base de datos.

```
# Insertar datos en la tabla clientes
clientes.to_sql('clientes', engine, index=False, if_exists='append')

# Insertar datos en la tabla canales
canales.to_sql('canales', engine, index=False, if_exists='append')

# Insertar datos en la tabla transacciones
transacciones.to_sql('transacciones', engine, index=False, if_exists='append')
```

Análisis descriptivo

Principalmente hacemos un análisis descriptivo general de la tabla de clientes donde podemos un resumen estadístico de las columnas numéricas, en este caso la columna ingresos_mensuales, esto se hace con el método .describe() de pandas:

```
# Resumen estadístico general de tabla clientes
print("Resumen estadístico general de clientes:")
print(clientes.describe())
```

```
Resumen estadístico general de clientes:
       ingresos_mensuales
count        6.980220e+05
mean         5.037177e+06
std          2.883255e+06
min          1.850000e+03
25%          2.575704e+06
50%          5.032722e+06
75%          7.528875e+06
max          9.995531e+06
```

Aquí podemos ver valores como la media, la desviación estándar, el valor mínimo, valor máximo y los cuartiles de 25%,50% y 75%.

Para la tabla de transacciones hacemos lo mismo.

```
# Resumen estadístico general de la tabla transacciones
print("Resumen estadístico general de transacciones:")
print(transacciones.describe())
```

```
Resumen estadístico general de transacciones:
                  fecha_transaccion         monto
count                       2360594  2.360594e+06
mean   2024-05-28 22:52:07.997105664  5.055470e+06
min              2024-01-01 00:00:00  1.000000e+00
25%              2024-03-04 00:00:00  1.000000e+05
50%              2024-06-16 00:00:00  3.000000e+05
75%              2024-08-09 00:00:00  1.000000e+06
max              2024-10-07 00:00:00  4.775500e+10
std                              NaN  1.633161e+08
```

Podemos también ver el numero total de transacciones por cliente

```
    # Número total de transacciones por cliente
    transacciones_por_cliente = transacciones.groupby('num_doc').size()
    print("\nNúmero total de transacciones por cliente:")
    print(transacciones_por_cliente)
```

```
Número total de transacciones por cliente:
num_doc
-1000076379879933291     9
-1000147930647869782    55
-1000460014616510321    13
-1000951368134169935     9
-100101085567707252     12
                        ..
999050126532234206      13
999075441145575820      78
999126550605799708      33
```

El monto total de transacciones por cliente

```
    # Monto total de transacciones por cliente
    monto_total_por_cliente = transacciones.groupby('num_doc')['monto'].sum()
    print("\nMonto total de transacciones por cliente:")
    print(monto_total_por_cliente)
```

```
Monto total de transacciones por cliente:
num_doc
-1000076379879933291      2230000.0
-1000147930647869782     21215000.0
-1000460014616510321     10390000.0
-1000951368134169935      5070000.0
-100101085567707252       6120000.0
                             ...
999050126532234206      345542384.0
999075441145575820        6270000.0
999126550605799708       60770000.0
```

Podemos ver media y mediana de ingresos mensuales por tipo de persona

```python
# Media y mediana de ingresos mensuales por tipo de persona
ingresos_por_tipo_persona = clientes.groupby('tipo_persona')['ingresos_mensuales'].agg(['mean', 'median'])
print("\nMedia y mediana de ingresos mensuales por tipo de persona:")
print(ingresos_por_tipo_persona)
```

```
Media y mediana de ingresos mensuales por tipo de persona:
                         mean     median
tipo_persona
-                5.180662e+06  5177504.0
NATURAL          4.814551e+06  5079649.5
PERSONA JURIDICA 5.072947e+06  5082839.0
PERSONA NATURAL  5.035913e+06  5028636.0
```

Podemos también ver la media y mediana del monto total de transacciones por tipo de persona

```python
# Media y mediana del monto total de transacciones por tipo de persona
monto_total_por_tipo_persona = transacciones.groupby('tipo_doc')['monto'].agg(['mean', 'median'])
print("\nMedia y mediana del monto total de transacciones por tipo de persona:")
print(monto_total_por_tipo_persona)
```

```
Media y mediana del monto total de transacciones por tipo de persona:
                                                 mean     median
tipo_doc
-                                       1.417000e+08  2511847.5
CEDULA DE CIUDADANIA                    1.053876e+06   300000.0
CEDULA DE EXTRANJERIA                   1.368191e+06   300000.0
DOCUMENTO VENEZOLANO/CARNET DIPLOMÁTICO 1.228372e+06   400000.0
ID EXTRANJERO PN NO RESIDENTE EN COLOMBIA 9.313810e+05   550000.0
NIT                                     1.013541e+07   865535.0
PASAPORTE                               1.149261e+06   400000.0
REGISTRO CIVIL                          7.113704e+05   250000.0
TARJETA DE IDENTIDAD                    6.556301e+05   200000.0
```

¿Qué clientes han realizado transacciones en los últimos 6 meses por un monto total superior al 200% de sus ingresos mensuales y superiores al percentil 95 del total de la población por tipo de persona?

Para responder a esta pregunta se desarrolló el siguiente query:

```sql
WITH TransaccionesUltimos6Meses AS (
    SELECT
        t.num_doc,
        SUM(t.monto) AS total_transacciones
    FROM
        transacciones t
    WHERE
        t.fecha_transaccion >= DATEADD(MONTH, -6, GETDATE())
    GROUP BY
        t.num_doc
),
ClientesConTransacciones AS (
    SELECT
        cl.tipo_doc,
        cl.num_doc,
        cl.nombre,
        cl.tipo_persona,
        cl.ingresos_mensuales,
        t.total_transacciones
    FROM
        clientes cl
    JOIN
        TransaccionesUltimos6Meses t ON cl.num_doc = t.num_doc
    WHERE
        t.total_transacciones > 2 * cl.ingresos_mensuales
),
Percentil95 AS (
    SELECT
        tipo_persona,
        PERCENTILE_CONT(0.95) WITHIN GROUP (ORDER BY total_transacciones)
            OVER (PARTITION BY tipo_persona) AS percentil_95
    FROM
        ClientesConTransacciones
)
SELECT DISTINCT
    c.tipo_doc,
    c.num_doc,
    c.nombre,
    c.tipo_persona,
    c.ingresos_mensuales,
    c.total_transacciones
FROM
    ClientesConTransacciones c
JOIN
    (SELECT DISTINCT tipo_persona, percentil_95 FROM Percentil95) p
    ON c.tipo_persona = p.tipo_persona
WHERE
    c.total_transacciones > p.percentil_95;
```

Esta primera parte del query filtramos las transacciones de los últimos 6 meses y calculamos el monto total de las transacciones por cliente.

```
WITH TransaccionesUltimos6Meses AS (
    SELECT
        t.num_doc,
        SUM(t.monto) AS total_transacciones
    FROM
        transacciones t
    WHERE
        t.fecha_transaccion >= DATEADD(MONTH, -6, GETDATE())
    GROUP BY
        t.num_doc
),
```

- WITH TransaccionesUltimos6Meses AS: Definimos una CTE llamada TransaccionesUltimos6Meses.
- SELECT t.num_doc, SUM(t.monto) AS total_transacciones: Seleccionamos el número de documento del cliente y suma el monto de las transacciones.
- FROM transacciones t: tomamos los datos provenientes de la tabla transacciones.
- WHERE t.fecha_transaccion >= DATEADD(MONTH, -6, GETDATE()): Filtramos las transacciones que se hicieron en los últimos 6 meses.
- GROUP BY t.num_doc: Agrupamos los resultados por el número de documento del cliente.

En esta parte unimos los datos de clientes con las transacciones filtradas y calculamos si el monto total de las transacciones es superior al 200% de los ingresos mensuales del cliente.

```
ClientesConTransacciones AS (
    SELECT
        cl.tipo_doc,
        cl.num_doc,
        cl.nombre,
        cl.tipo_persona,
        cl.ingresos_mensuales,
        t.total_transacciones
    FROM
        clientes cl
    JOIN
        TransaccionesUltimos6Meses t ON cl.num_doc = t.num_doc
    WHERE
        t.total_transacciones > 2 * cl.ingresos_mensuales
),
```

- ClientesConTransacciones AS: Definimos una CTE llamada ClientesConTransacciones.

- SELECT cl.tipo_doc, cl.num_doc, cl.nombre, cl.tipo_persona, cl.ingresos_mensuales, t.total_transacciones: Seleccionamos varias columnas de la tabla clientes y la columna total_transacciones de TransaccionesUltimos6Meses.
- FROM clientes cl: Indicamos que los datos provienen de la tabla clientes.
- JOIN TransaccionesUltimos6Meses t ON cl.num_doc = t.num_doc: Unimos la tabla clientes con TransaccionesUltimos6Meses usando num_doc.
- WHERE t.total_transacciones > 2 * cl.ingresos_mensuales: Filtramos los clientes cuyo monto total de transacciones es superior al 200% de sus ingresos mensuales.

En siguiente parte calculamos el percentil 95 del monto total de las transacciones por tipo de persona.

```
Percentil95 AS (
    SELECT
        tipo_persona,
        PERCENTILE_CONT(0.95) WITHIN GROUP (ORDER BY total_transacciones)
            OVER (PARTITION BY tipo_persona) AS percentil_95
    FROM
        ClientesConTransacciones
)
```

- Percentil95 AS: Definimos una CTE llamada Percentil95.
- SELECT tipo_persona, PERCENTILE_CONT(0.95) WITHIN GROUP (ORDER BY total_transacciones) OVER (PARTITION BY tipo_persona) AS percentil_95: Calculamos el percentil 95 del monto total de las transacciones dentro de cada grupo de tipo_persona.
- FROM ClientesConTransacciones: Indicamos que los datos provienen de la CTE ClientesConTransacciones.

En la ultima parte de la consulta filtramos los clientes que cumplen con ambas condiciones y seleccionamos las columnas requeridas.

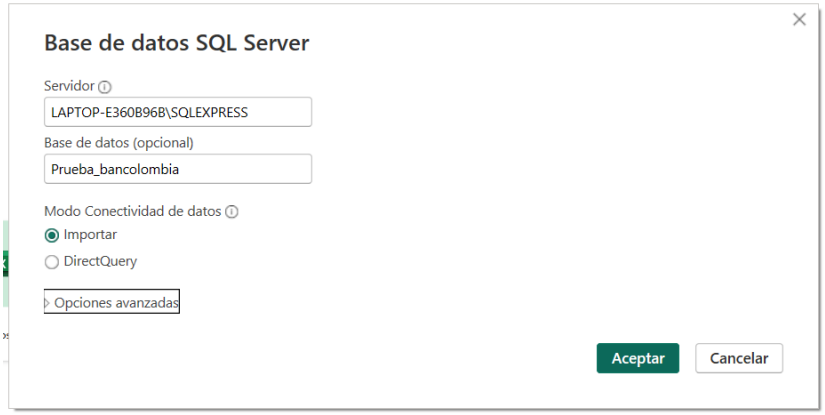
```sql
SELECT DISTINCT
    c.tipo_doc,
    c.num_doc,
    c.nombre,
    c.tipo_persona,
    c.ingresos_mensuales,
    c.total_transacciones
FROM
    ClientesConTransacciones c
JOIN
    (SELECT DISTINCT tipo_persona, percentil_95 FROM Percentil95) p
    ON c.tipo_persona = p.tipo_persona
WHERE
    c.total_transacciones > p.percentil_95;
```

- SELECT DISTINCT: Seleccionamos filas únicas para evitar duplicados.
- c.tipo_doc, c.num_doc, c.nombre, c.tipo_persona, c.ingresos_mensuales, c.total_transacciones: Seleccionamos las columnas requeridas de ClientesConTransacciones.
- FROM ClientesConTransacciones c: Indicamos que los datos provienen de la CTE ClientesConTransacciones.
- JOIN (SELECT DISTINCT tipo_persona, percentil_95 FROM Percentil95) p ON c.tipo_persona = p.tipo_persona:
  Unimos ClientesConTransacciones con Percentil95 usando tipo_persona.
- WHERE c.total_transacciones > p.percentil_95: Filtra los clientes cuyo monto total de transacciones es superior al percentil 95 de su tipo de persona.

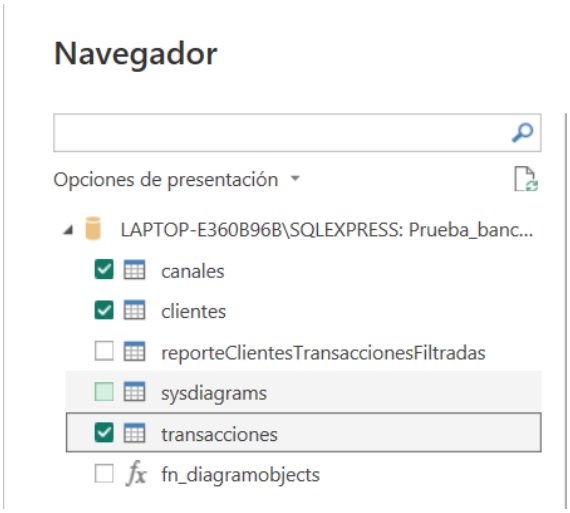
Finalmente se guardo el resultado en un Excel llamado ‘reporteClientesTransaccionesFiltradas.xls’ respondiendo así a la pregunta realizada.

Visualización

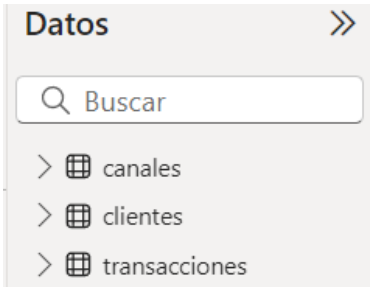
La visualización de los datos las hice en PowerBI. Iniciamos cargando los datos en powerBI desde la base de datos. Indicamos el servidor y la base de datos.



Luego seleccionamos las tablas que queremos cargar, que en este caso son canales, clientes y transacciones.



De esa forma ya tendríamos los datos cargados y podemos trabajar con ellos.



El dashboard lo iniciamos con una portada con una pequeña descripción de el objetivo del mismos.

Cada botón nos llevara a ver el comportamiento transaccional en el tiempo de un cliente, un canal o una jurisdicción. El botón de cliente nos llevara a ver el siguiente panel, donde podremos ver dos gráficos de línea, el primero para ver monto total de transacciones de un cliente por meses, el segundo nos permitirá ver el total de transacciones por mes, además hay unos paneles que nos muestran información adicional como total de transacciones del cliente, monto total y monto promedio.

Para jurisdicción y canales, se tiene las mismas visualizaciones.

Conclusiones:

La extracción de datos desde Kaggle y la transformación utilizando Pandas manejando sólidamente las herramientas permitió una buena manipulación de los datos. La limpieza de datos duplicados y la conversión de tipos son pasos esenciales que se realizaron correctamente.

El análisis descriptivo permitió obtener un análisis básico de los clientes, incluyendo valores de ingreso y características de las transacciones. Esto permite tener una buena base para entender el perfil general de los clientes y sus patrones de transacción.

La consulta SQL la cual buscaba identificar clientes con transacciones que superan el 200% de sus ingresos mensuales y el percentil 95 permitió filtrar datos de clientes con alta relevancia transaccional.