

Desarrollo de prueba

FASE 1- Unificar tablas con Python

1. Importar librerías a utilizar

```
In [2]: import numpy as np
import matplotlib.pyplot as plt
import math
import seaborn as sns
import pandas as pd
```

2. Se nos proporcionaron 3 tablas

```
3]: BARRIOS = pd.read_csv(r'C:\Users\geosh\Documents\peoplesoft_test\exa_barrios.csv')
DISPOSITIVOS = pd.read_csv(r'C:\Users\geosh\Documents\peoplesoft_test\exa_dispositivos.csv')
CLIENTES = pd.read_csv(r'C:\Users\geosh\Documents\peoplesoft_test\exa_trx_clientes.csv')
```

A continuación, las siguientes tablas creadas que luego unificaremos para exportar y analizar en SQL

Tabla de Barrios

In [5]:

BARRIOS

Out[5]:

	id_barrio	nombre_barrio
0	737998257	La Campina
1	737999435	Guayaquil
2	737998219	Vipasa
3	737996837	Lourdes
4	737997031	Villanueva
...
332	737998799	Ignacio Rengifo
333	738000408	Prados del Limonar
334	737998040	Pizamos III-Las Dalias
335	737998078	Brisas de Mayo
336	737998035	Ciudadela del Río

337 rows × 2 columns

Tabla de Dispositivo

```
In [4]: DISPOSITIVOS
```

Out[4]:

	tipo_dispositivo	codigo_dispositivo	latitud	longitud	id_barrio
0	POS	1024702	3.451135	-76.530893	737998832
1	POS	1076402	3.446585	-76.517672	738000381
2	POS	1077002	3.446585	-76.517672	738000381
3	POS	3342404	3.476909	-76.485286	737998905
4	POS	3378003	3.485575	-76.516418	737998174
...
37279	POS	74486603	3.452831	-76.523627	737998935
37280	POS	74632603	3.450660	-76.533361	737998832
37281	POS	4023000120	3.450063	-76.530697	737998832
37282	POS	4408002741	3.434776	-76.543102	737999403
37283	POS	4023002563	3.414809	-76.546871	737998095

37284 rows × 5 columns

Tabla de clientes

```
In [6]: CLIENTES
```

Out[6]:

	num_cliente	tipo_doc	tipo_dispositivo	codigo_dispositivo	num_trx	mnt_total_trx
0	6.861790e+18	1	POS	14812028	3	22218042.0
1	1.552970e+18	1	POS	14388938	3	21583407.0
2	-4.075900e+18	1	POS	12370698	1	6185349.0
3	3.539340e+18	1	POS	10313583	2	10313583.0
4	5.595550e+18	1	POS	14358071	4	28716142.0
...
93441	-5.638570e+18	2	POS	13809496	1	6904748.0
93442	-6.859290e+18	2	DISPENSADOR	1116	1	558.0
93443	-1.762470e+18	2	SAI	745	1	372.5
93444	-2.708240e+18	2	DISPENSADOR	4090	1	2045.0
93445	4.476350e+18	2	POS	15409667	2	15409667.0

93446 rows × 6 columns

3. Unificación de tablas

Luego de analizar las tablas, podemos unificarlas ya que algunos “Keys” se repiten y utilizaremos estos para la unificación.

- La tabla de dispositivos y barrios se pueden unir por “ID_BARRIO” con la función MERGE de Pandas. Por lo cual crearemos un nuevo dataframe llamado NEWDF

```
NEWDF = pd.merge(DISPOSITIVOS, BARRIOS, on='id_barrio')
```

- Por último, unificamos ‘NEWDF’ con la tabla ‘CLIENTES’ usando ‘codigo_dispositivo’ y ‘tipo_dispositivo’ ya que se repite en ambos dataframes.

```
TRANSACCIONES = pd.merge(NEWDF, CLIENTES, on=['codigo_dispositivo', 'tipo_dispositivo' ])
```

```
In [8]: TRANSACCIONES = pd.merge(NEWDF, CLIENTES, on=['codigo_dispositivo', 'tipo_dispositivo' ])  
TRANSACCIONES
```

```
Out[8]:
```

	tipo_dispositivo	codigo_dispositivo	latitud	longitud	id_barrio	nombre_barrio	num_cliente	tipo_doc	num_trx	mnt_total_trx
0	POS	11304805	3.451187	-76.531933	737998832	San Pedro	9.026910e+18	1	1	5652402.5
1	POS	11304805	3.451187	-76.531933	737998832	San Pedro	2.844930e+18	1	2	11304805.0
2	POS	11304805	3.451187	-76.531933	737998832	San Pedro	-1.583450e+18	1	1	5652402.5
3	POS	11304805	3.451187	-76.531933	737998832	San Pedro	1.122050e+18	1	1	5652402.5
4	POS	11316429	3.451244	-76.530667	737998832	San Pedro	9.006950e+18	1	1	5658214.5
...
93441	POS	11896917	3.477901	-76.482795	737998907	Petecuy II	4.633210e+18	1	2	11896917.0
93442	POS	14587661	3.441552	-76.546751	737998684	Santa Bárbara	7.943360e+18	1	1	7293830.5
93443	POS	14107627	3.434907	-76.549859	737998112	Altos de Santa Isabel La Morelia	-5.277160e+18	1	1	7053813.5
93444	POS	15528193	3.428579	-76.495799	737998953	Villa Blanca	-3.579070e+18	1	1	7764096.5
93445	POS	15383565	3.428883	-76.495888	737998953	Villa Blanca	-3.832090e+18	1	1	7691782.5

93446 rows × 10 columns

4. Análisis exploratorio en Python para entender mejor nuestra tabla para posterior análisis en sql.
- a. Quería saber los tipos de dispositivos y la cantidad de veces que aparecía en la tabla

```
TRANSACCIONES['tipo_dispositivo'].value_counts()
```

```
DISPENSADOR    37090
POS            37067
SAI            8810
CB             7076
MF            3386
PAC             17
Name: tipo_dispositivo, dtype: int64
```

- b. Cantidad de barrios en la tabla

```
TRANSACCIONES['nombre_barrio'].value_counts()
```

```
San Pedro                4704
Urbanización San Joaquín 4352
Santa Mónica Residencial 3811
Chipichape              3437
Unicentro Cali          3157
...
Polvorines                1
San Benito                1
Santa Bárbara            1
Altos de Santa Isabel La Morelia 1
La Esperanza             1
Name: nombre_barrio, Length: 277, dtype: int64
```

5. Por último, exportamos archivo con el siguiente código.

```
TRANSACCIONES.to_excel("TRANSACCIONES.xlsx",
                        sheet_name='TABLA')
```

FASE 2 – ANÁLISIS EN SQL

1. *¿Cuáles son los barrios en los cuales se realizaron los movimientos de al menos el 51% del dinero total trizado por cada cliente? Considere que los clientes pueden tener más de un barrio.*

a. Analicé el monto total de las transacciones de la tabla.

```
SELECT SUM(mnt_total_trx) as 'Monto Total de transacciones' from project..tabla
```

b. La siguiente consulta muestra el monto total de las transacciones realizadas por cada cliente en la tabla, el porcentaje que representa ese monto en relación con el total de la tabla, y los agrupa por barrio y número de cliente, ordenando el porcentaje descendentemente.

```
select nombre_barrio, num_cliente, SUM(mnt_total_trx) as 'Monto Transacciones por cliente',  
ROUND(100* SUM(mnt_total_trx) / (SELECT SUM(mnt_total_trx)FROM Project..tabla),2) AS 'Porcentaje'  
from Project..tabla  
group by nombre_barrio, num_cliente  
order by SUM(mnt_total_trx) DESC
```

c. Se creó una tabla nueva llamada “TABLA2” para analizar la cantidad de transacciones según barrio. Una vez creada se realizó la siguiente consulta.

```
-- AGRUPAR CANTIDAD TOTAL DE TRANSACCIONES POR BARRIO
```

```
SELECT DISTINCT BARRIO, SUM(TRX) AS 'SUMA DE TRANSACCIONES', ROUND(100 * SUM(TRX) / (SELECT  
SUM(TRX) FROM TABLA2),2) AS 'PORCENTAJE'  
FROM TABLA2  
group by BARRIO  
ORDER BY 'PORCENTAJE' DESC
```

Esto permitió saber en que barrios se concentra la mayor cantidad de transacciones.

2. ¿Cuáles son los dispositivos con transacciones de al menos 100 clientes diferentes?

a. Se realizó la siguiente consulta para obtener los valores: la consulta muestra los primeros 100 clientes todos distintos y su respectivo tipo_dispositivo.

```
select distinct TOP 100 num_cliente, tipo_dispositivo
from project..tabla
order by tipo_dispositivo
```

3. ¿Cuáles son los 5 barrios donde la mayor cantidad de clientes únicos realizan transacciones en dispositivos tipo POS? La respuesta debe incluir la cantidad de clientes asociados a estos barrios.

a. Se realizó la consulta que muestra la cantidad de clientes distintos utilizando la función COUNT POR Barrio, filtrando solamente donde se hayan utilizado los dispositivos 'POS'.

```
SELECT nombre_barrio, COUNT( DISTINCT num_cliente) as 'Clientes Distintos'
FROM project..tabla
where tipo_dispositivo = 'POS'
group by nombre_barrio
ORDER BY COUNT(DISTINCT num_cliente) DESC;
```

4. ¿Cuáles son las 10 distancias únicas (en kilómetros) de los dispositivos más alejados entre sí del barrio Sucre?

a. Primero analizamos los diferentes barrios con sus respectivas latitud y longitud.

```
SELECT DISTINCT(nombre_barrio), latitud, longitud from project..tabla order by nombre_barrio
```

b. Utilizaremos el método **geography::Point** para calcular la distancia de cada barrio del barrio sucre. Para esto declaramos la variable @Sucre como punto de comparación.

Ordenaremos en la consulta de mayor a menor el resultado en KM y seleccionaremos los 10 primeros barrios, que serían los barrios más alejados.

```
DECLARE @sucre geography;
SET @sucre = geography::Point(3.446585795, -76.52194317, 4326);
SELECT TOP 10 nombre_barrio, ROUND(AVG(geography::Point(Latitud, Longitud,
4326).STDistance(@sucre)/1000),1) AS 'Distancia en KM del Barrio Sucre'
FROM project..tabla
GROUP BY nombre_barrio
ORDER BY ROUND(AVG(geography::Point(Latitud, Longitud, 4326).STDistance(@sucre)/1000),1) DESC;
```

Código completo en el siguiente link:

<https://github.com/ninoshkavega/EXPLORATORY-PROJECT/tree/main>