Descripción y Análisis Prueba Técnica Analista ll

Nota: Este script realiza análisis de datos utilizando Python y SQL. Las versiones del código en Python y SQL se encuentran versionadas en el repositorio https://github.com/juanmgart92/analisis-datos-bancolombia.git y los resultados de salida se almacenan en la carpeta **outputs.**

Ejercicio 1: Obtener listado de clientes masculinos con transacciones en 2 o más dispositivos.

```python
def obtener_clientes_masculinos_con_dispositivos(master_df, itc_df, dispositivos=2):
    """
    Obtiene un listado de clientes masculinos con transacciones en al menos la
    cantidad de dispositivos especificada.
    """
    clientes_masculinos = master_df[master_df['genero_cli'] == 'M']

    master_itc = pd.merge(clientes_masculinos, itc_df, left_on='num_doc', right_on='documento', how='left')

    resultado = master_itc.groupby(['num_doc', 'genero_cli']).disposit.nunique().reset_index()

    resultado = resultado[resultado['disposit'] >= dispositivos]

    resultado.to_csv('./outputs/sol_ej_1.csv', index=False)

    return resultado
```

Python:

La función **obtener_clientes_masculinos_con_dispositivos** realiza la siguiente operación:

1. Se filtran los clientes masculinos del DataFrame master_df.
2. Se realiza una fusión (merge) entre el DataFrame filtrado y el DataFrame itc_df utilizando las columnas **'num_doc'** y **'documento'** respectivamente.
3. Se agrupa el DataFrame resultante por el número de documento **'num_doc'** y el género del cliente **'genero_cli'**.
4. Se calcula la cantidad de dispositivos únicos **'disposit'** asociados a cada cliente.
5. Se filtran aquellos clientes que tienen transacciones en al menos la cantidad de dispositivos especificada.
6. El resultado se guarda en un archivo CSV llamado 'sol_ej_1.csv'

SQL ejercicio1_SQL.sql:

1. La consulta inicia con una vista llamada **'MasterItc'** donde se selecciona las columnas genero_cli, num_doc de la tabla master y disposit de la tabla itc donde se hace un left join con llave de unión el documento del cliente de las dos tablas adicional se filtra por genero masculino.
2. Se realiza una consulta a la vista **'MasterItc'** donse traemos los clientes masculinos y adicional hacemos un count a la variable disposit añadiéndole la clausula distinct para evitar la replicación de ellos y así traemos el conteo de dispositivos, finalmente agrupamos los clientes para no traer duplicados y que el conteo de los dispositivos sea mayor o igual a 2.

Ejercicio 2: Definir el canal con mayor suma de valor de transacciones de cada cliente.

```python
def obtener_canal_con_mayor_suma(itc_df):
    """
    Obtiene el canal con la mayor sumatoria de valor de transacciones de cada cliente.
    """
    total_canales_cliente = itc_df.groupby(['documento', 'canal'])['vlrtran'].sum().reset_index()

    orden_costos = total_canales_cliente.sort_values('vlrtran', ascending=False) \
                                        .groupby('documento') \
                                        .head(1) \
                                        .reset_index(drop=True)

    orden_costos = orden_costos[['documento', 'canal', 'vlrtran']]

    orden_costos.to_csv('./outputs/sol_ej_2.csv', index=False)

    return orden_costos
```

Python:

La función **obtener_canal_con_mayor_suma** realiza la siguiente operación:

1. Se agrupa el DataFrame que recibe el parametro itc_df por las columnas 'documento' y 'canal', sumando el valor de transacciones ('vlrtran').
2. Se ordenan los resultados de manera descendente según la suma de valor de transacciones.
3. Se selecciona el canal con la mayor suma para cada cliente utilizando el método head(1) por grupo.
4. El resultado se guarda en un archivo CSV llamado 'sol_ej_2.csv'.

SQL ejercicio2_SQL.sql:

1. Crea una vista temporal llamada **totalCanalesCliente** que agrupa las transacciones en el DataFrame itc por 'documento' y 'canal', calculando la suma de 'vlrtran' para cada grupo.
2. Crea otra vista temporal llamada ordenCostos que asigna un número de fila (orden) a cada grupo de 'documento', ordenado por la suma de 'vlrtran' de manera descendente.
3. La consulta final selecciona el 'documento', 'canal' y 'suma_vlrtran' de la vista ordenCostos donde el orden es igual a 1, lo que significa que se selecciona el canal con el mayor valor de transacciones para cada cliente.

Ejercicio 3: Identificar transacciones exitosas donde cada cliente realiza al menos el 70% del monto total transferido

```python
def obtener_transacciones_exitosas_con_porcentaje(itc_df, porcentaje=0.7):
    """
    Obtiene transacciones exitosas donde cada cliente realiza al menos el porcentaje
    especificado del monto total transferido.
    """
    transacciones_exitosas = itc_df[itc_df['cdgrpta'] == 0]

    total_por_cliente = transacciones_exitosas.groupby('documento')['vlrtran'].sum().reset_index()

    resultado = pd.merge(transacciones_exitosas, total_por_cliente, on='documento', suffixes=('', '_total'))

    resultado = resultado[resultado['vlrtran'] >= resultado['vlrtran_total'] * porcentaje]

    resultado = resultado[['documento', 'cdgrpta', 'vlrtran', 'vlrtran_total']]

    resultado.to_csv('./outputs/sol_ej_3.csv', index=False)

    return resultado
```

Python:

La función **obtener_transacciones_exitosas_con_porcentaje** realiza la siguiente operación:

1. Se filtran las transacciones exitosas del DataFrame itc_df.
2. Se calcula la suma total de valor de transacciones por cliente.
3. Se realiza una fusión entre las transacciones exitosas y la suma total por cliente.
4. Se filtran las transacciones donde el valor es al menos el 70% del monto total transferido por cliente.
5. El resultado se guarda en un archivo CSV llamado 'sol_ej_3.csv'.

SQL ‘ejercicio3_SQL.sql’:

1. Crea una vista temporal llamada **TotalPorCliente** que agrupa las transacciones en el DataFrame ITC por 'documento', calculando la suma de 'vlrtran' solo para las transacciones con 'cdgrpta' igual a 0.
2. La consulta final selecciona las columnas 'documento', 'cdgrpta', 'vlrtran', y 'suma_vlrtran' de los DataFrames ITC y TotalPorCliente, respectivamente, utilizando una condición de unión en la columna 'documento'.
3. Filtra los resultados para incluir solo aquellas transacciones donde 'vlrtran' es mayor o igual al 70% de la suma total de 'vlrtran' para el respectivo cliente y donde 'cdgrpta' es igual a 0.

```python
if __name__ == "__main__":
    try:
        master = pd.read_csv("./src/master.csv", sep=",")
        itc = pd.read_csv("./src/ITC.csv", sep=",")

        # Ejercicio 1
        obtener_clientes_masculinos_con_dispositivos(master, itc)

        # Ejercicio 2
        obtener_canal_con_mayor_suma(itc)

        # Ejercicio 3
        obtener_transacciones_exitosas_con_porcentaje(itc)

    except Exception as e:
        print(f"Ocurrió un error: {e}")
```

Por último, se invoca tres funciones: obtener_clientes_masculinos_con_dispositivos, obtener_canal_con_mayor_suma, y obtener_transacciones_exitosas_con_porcentaje con los DataFrames correspondientes. Cualquier excepción durante la ejecución se maneja, y se imprime un mensaje de error en caso de un problema.

Power Bi: Se realiza un tablero con 4 gráficos, donde tenemos el top5 por ciudad/país con mas clientes, cantidad de respuesta exitosa y fallida por transacción, un grafico de pastel donde vemos la distribución del valor total de transacciones agrupadas por genero del cliente, un grafico del promedio del valor de la transacción con respuestas exitosas, como solo tenemos información del año 2024 y del mes de enero se realizó filtro por días de dicho mes el archivo que contiene la solución es reporte_bi.pbix