

## **Prueba Técnica Ingeniero De Datos**

**Presentada por:** Laura Camila Fernandez Ospina

**Cargo al que se postula:** Ingeniero de Datos

**Fecha de Aplicación:** 03 de marzo de 2025

**Empresa:** Banco de Bogotá

**Descripción:** Se hace el desarrollo de la prueba técnica para el cargo de Ingeniero de Datos, obteniendo como resultado algunos entregables que se detallan a continuación, adicionalmente estos entregables los encontrarán en la URL del proyecto en GIT y estarán nombrados de la misma manera como se detalla en este documento.

**URL GitHub:** [https://github.com/Laura020/PruebaBanco\\_Bogota.git](https://github.com/Laura020/PruebaBanco_Bogota.git)

**Entregables de documentos a tener en cuenta:**

1. [datos\\_transacciones.csv](#)
2. [datos\\_transacciones\\_limpio.csv](#)
3. [datos\\_transacciones\\_final.csv](#)

## TABLA DE CONTENIDO

Parte 1. Modelado y Arquitectura de datos (Teórico – Diseño).....	3
1.1.    Diseño del Modelo Dimensional .....	3
1.2.    Justificación Modelo .....	4
1.3.    Estrategia de Particionamiento y Clustering en BigQuery.....	4
1.4.    Calidad y Gobernanza de los datos .....	4
1.5.    Trazabilidad y Auditoria de Datos .....	5
Parte 2. Construcción de Pipeline ETL/ELT (Práctico - Implementación) .....	6
2.1.    Cargue de información .....	6
2.2.    Aplicación de reglas de limpieza y transformación .....	7
2.3.    Cargue de datos finales.....	8
Parte 3: Explotación y Acceso a los Datos (Práctico - Análisis).....	10

## TABLA DE ILUSTRACIONES

Ilustración 1: Modelo Dimensional Estrella.....	3
Ilustración 2: Creación Bucket .....	6
Ilustración 3: Cargue documento .....	6
Ilustración 4: Codificación limpieza de datos.....	7
Ilustración 5: Creación de conjunto de datos .....	8
Ilustración 6: Visualización de campos.....	8
Ilustración 7: Creación Particionamiento .....	9
Ilustración 8: Creación Clustering .....	9
Ilustración 9: Confirmación creación tablas .....	9
Ilustración 10: Consulta para conocer los clientes más rentables .....	10
Ilustración 11: Consulta para conocer los clientes más rentables por movimientos y transacciones .....	11
Ilustración 13: Rentabilidad por cliente .....	11

## TABLAS

Tabla 1: Clientes más rentables .....	10
Tabla 2: Clientes por movimientos y transacción.....	11

## Parte 1. Modelado y Arquitectura de datos (Teórico – Diseño)

### 1.1. Diseño del Modelo Dimensional

Para estructurar el Data Warehouse (DWH) de la entidad bancaria, se adopta un **modelo dimensional tipo estrella**, ya que proporciona un equilibrio óptimo entre rendimiento, facilidad de consulta y escalabilidad.

#### Modelo propuesto:

- **Tabla de hechos: transacciones** Contiene información detallada de cada operación financiera realizada por los clientes, incluyendo montos, tipos de transacción y productos asociados.
- Tablas de dimensión:
  - **dim\_cliente:** Información del cliente (ID, nombre, edad, género, segmento de ingresos, historial de crédito).
  - **dim\_producto:** Detalles de los productos financieros (tipo: cuenta de ahorro, tarjeta de crédito, crédito).
  - **dim\_tiempo:** Estructura temporal (fecha, mes, trimestre, año).
  - **dim\_riesgo\_credicio:** Score crediticio y nivel de riesgo de los clientes.

A continuación, se presenta la diagramación del modelo estrella. Este se eligió un modelo estrella porque organiza los datos en una tabla central de hechos y tablas de dimensiones relacionadas. Esto optimiza las consultas y facilita análisis rápidos y eficientes.

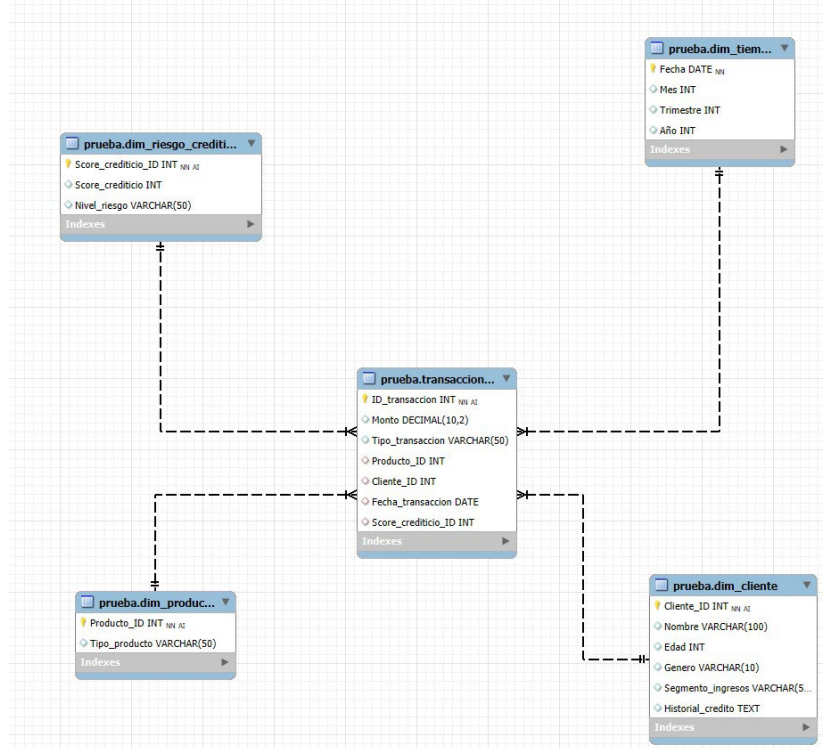


Ilustración 1: Modelo Dimensional Estrella

## 1.2. Justificación Modelo

El modelo estrella fue seleccionado por sus siguientes ventajas:

- Eficiencia en consultas analíticas: Reducción de la cantidad de JOINS complejos, mejorando la latencia en entornos de Big Data.
- Facilidad de integración con herramientas de BI y reporting: La estructura clara permite que analistas accedan a los datos sin conocimiento profundo de SQL.
- Escalabilidad: Permite la incorporación de nuevas dimensiones sin alterar significativamente la estructura central.

Este modelo facilita análisis rápidos, evita complejidad y optimiza la conexión con herramientas de negocio sin afectar el rendimiento.

## 1.3. Estrategia de Particionamiento y Clustering en BigQuery

Para optimizar el rendimiento en BigQuery, se implementará:

- **Particionamiento por fecha\_transaccion (DATE/TIMESTAMP):**
  - Facilita consultas sobre períodos específicos sin escanear toda la tabla
  - Reduce costos operativos al disminuir el volumen de datos analizados.
- **Clustering por id\_cliente y tipo\_producto:**
  - Acelera consultas segmentadas por cliente o producto.
  - Mejora la compresión y organización interna de los datos.

Esta estrategia divide los datos en bloques por fecha y agrupa la información similar, optimizando la velocidad de consulta.

## 1.4. Calidad y Gobernanza de los datos

Para asegurar calidad y gobernanza en el pipeline, se implementarán:

- **Control de calidad:**
  - Validación de datos en la ingesta con reglas de integridad (verificación de NULL, claves foráneas).
  - Eliminación de duplicados en la ETL mediante técnicas de deduplicación.
- **Gobernanza de datos:**
  - Políticas RBAC en BigQuery para controlar el acceso por roles.
  - Catalogación y documentación de metadatos con Google Data Catalog.
  - Registro de auditoría con Cloud Audit Logs.
  - Esto asegura la integridad de los datos, controla el acceso y facilita la auditoría de cambios.

Estas medidas aseguran la integridad de los datos, controlan el acceso adecuado y permiten una auditoría efectiva.

### 1.5. Trazabilidad y Auditoria de Datos

Para garantizar la trazabilidad de los datos a lo largo del pipeline, se implementarán los siguientes mecanismos:

- **Registro de Logs con Cloud Logging:**
  - Captura de errores en cada etapa del proceso ETL.
  - Almacenamiento de eventos de ejecución para análisis forense.
- **Metadatos de procesamiento:**
  - Inclusión de columnas de auditoría (created\_at, updated\_at, source\_system) en las tablas.
  - Uso de Google Data Catalog para etiquetar datasets y facilitar la trazabilidad.
- **Versionamiento de datos:**
  - Implementación de Time Travel en BigQuery para mantener versiones históricas.

Estos mecanismos aseguran un seguimiento detallado de los datos, permiten auditar cambios y recuperar versiones anteriores en caso de error.

## Parte 2. Construcción de Pipeline ETL/ELT (Práctico - Implementación)

### 2.1. Cargue de información

El primer paso en el proceso fue cargar los datos en un **Data Lake** usando **Google Cloud Storage**. A continuación, se detallan las acciones realizadas:

**2.1.1 Acceso a la Consola de Google Cloud:** Ingrese a la **Consola de Google Cloud** para iniciar la gestión de los recursos.

#### 2.1.2 Creación del Bucket:

- Seleccione la opción para crear un nuevo depósito en **Google Cloud Storage**.
- Asigné el nombre data-lake-transacciones al bucket

A continuación, podrán observar la **Ilustración #01**, que deben **Almacenamiento en la nube de Google**.

Buckets

+ CREAR

ACTUALIZAR

IR A RUTA

MÁS INFORMACIÓN

Filtro

Filtrar depósitos

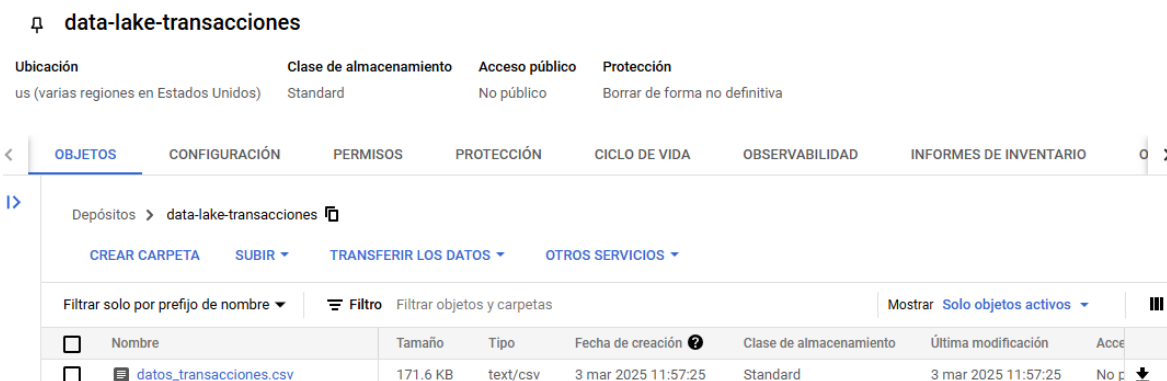
<input type="checkbox"/>	Nombre ↑	Fecha de creación	Tipo de ubicación	Ubicación	Clase de almacenamiento predeterminada ?	Última modificación
<input type="checkbox"/>	<a href="#">data-lake-transacciones</a>	3 mar 2025 09:08:13	Multi-region	us	Standard	3 mar 2025 09:08:13

Ilustración 2: Creación Bucket

#### 2.1.3 Cargar archivo CSV :

- Una vez creado el bucket, procedí a cargar el archivo CSV con los datos de las transacciones
- Utilicé la opción de "Subir archivos" en el depósito para cargar los datos directamente desde mi sistema local a **Google Cloud Storage**.

Posteriormente podrán visualizar la ilustración #02 en la que se visualiza el cargue del documento:



data-lake-transacciones							
Ubicación	Clase de almacenamiento	Acceso público	Protección				
us (varias regiones en Estados Unidos)	Standard	No público	Borrar de forma no definitiva				
<a href="#">OBJETOS</a> CONFIGURACIÓN PERMISOS PROTECCIÓN CICLO DE VIDA OBSERVABILIDAD INFORMES DE INVENTARIO							
Depósitos > data-lake-transacciones							
<a href="#">CREAR CARPETA</a> <a href="#">SUBIR</a> <a href="#">TRANSFERIR LOS DATOS</a> <a href="#">OTROS SERVICIOS</a>							
Filtrar solo por prefijo de nombre		Filtro Filtrar objetos y carpetas			Mostrar Solo objetos activos		
<input type="checkbox"/>	Nombre	Tamaño	Tipo	Fecha de creación ⓘ	Clase de almacenamiento	Última modificación	Acceso
<input type="checkbox"/>	<a href="#">datos_transacciones.csv</a>	171.6 KB	text/csv	3 mar 2025 11:57:25	Standard	3 mar 2025 11:57:25	No p

Ilustración 3: Cargue documento

## 2.2. Aplicación de reglas de limpieza y transformación

Una vez cargados los datos en el Data Lake (limpieza, transformación y control de calidad a los datos suministrados. La ejecución de esta tarea se realizó desde la consola de Cloud Shell de Google Cloud:

### 2.2.1. Limpieza de datos:

- Eliminación de Valores Nulos: Se identificaron columnas que se encontraban con información vacía y estas se omitieron.
- Corregir errores de formato: Se realiza de caracteres especiales.
- Eliminación de Duplicados: Se realiza la eliminación de valores duplicados que se puedan presentar.

### 2.2.2. Transformación de Datos:

- Conversión de tipos de datos: Se realizó la transformación del campo monto en tipo FLOAT.
- Se realizó la revisión de que la fecha se encontrara en formato fecha.

A screenshot of a code editor window titled 'dataflow\_transformation.py'. The code is a Python script for a Dataflow pipeline. It imports 'apache\_beam as beam' and 'PipelineOptions' from 'apache\_beam.options.pipeline\_options'. A class 'TransformarDatos' inherits from 'beam.DoFn'. The 'process' method takes 'self' and 'row' as arguments. It splits the row into values. If there are fewer than 4 values, it returns an empty list to skip incomplete rows. Otherwise, it extracts 'id\_transaccion', 'monto', 'fecha', and 'cliente\_id'. It then tries to convert 'monto' to a float and strip the 'fecha'. If successful, it returns a list with these four values. If there's an exception, it returns an empty list to skip corrupted data. Finally, it configures 'PipelineOptions' with 'DataflowRunner', the project name 'dataflow\_transformation.py', and a temporary location 'gs://data-lake-transacciones/temp'.

```
1 import apache_beam as beam
2 from apache_beam.options.pipeline_options import PipelineOptions
3
4
5 class TransformarDatos(beam.DoFn):
6     def process(self, row):
7         values = row.split(',')
8         if len(values) < 4:
9             return # Omitir filas incompletas
10
11         id_transaccion, monto, fecha, cliente_id = values
12         try:
13             monto = float(monto) # Convertir el monto a número
14             fecha = fecha.strip() # Asegurar formato correcto de fecha
15             return [(id_transaccion, monto, fecha, cliente_id)]
16         except:
17             return # Omitir datos corruptos
18
19 # Configuración del pipeline de Dataflow
20 options = PipelineOptions([
21     runner='DataflowRunner',
22     project='dataflow_transformation.py', # Reemplaza esto con el ID de tu proyecto GCP
23     temp_location='gs://data-lake-transacciones/temp'
24 ])
```

Ilustración 4: Codificación limpieza de datos

## Realice uso de las herramientas de Google Cloud para la Limpieza y Transformación:

- **BigQuery:** Es muy útil para realizar consultas SQL avanzadas que permitan transformar y limpiar los datos en grandes volúmenes, aplicando funciones como COALESCE para reemplazar valores nulos o TRIM para eliminar espacios en blanco.
- **Cloud Dataflow:** Utilizado para la ejecución de flujos de trabajo de transformación de datos en tiempo real o por lotes. Permite aplicar transformaciones complejas como el agrupamiento, filtrado y la transformación de tipos de datos.

## 2.3. Cargue de datos finales

**2.3.1. Acceso a la Consola de Google Cloud:** Ingrese a Big Query para iniciar con la gestión de la creación de conjunto de datos

### 2.3.2. Creación del conjunto de datos:

- Seleccione la opción para crear el conjunto de datos.
- Asigné el nombre dwh\_transacciones
- Posteriormente realice la creación de la tabla dentro del conjunto de datos.
- Se crea la tabla 'mi\_Tabla'.

A continuación, podrán observar la **Ilustración #04**, donde se observará el conjunto de datos y la tabla creada.

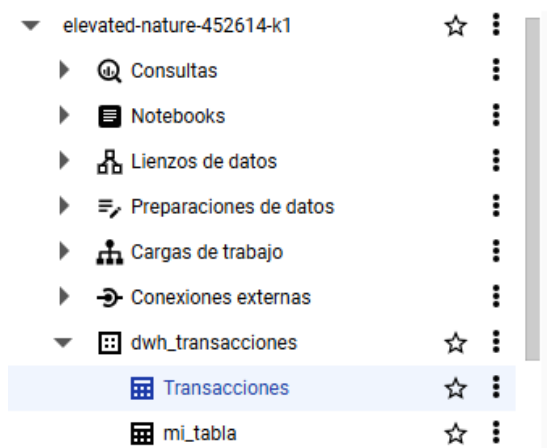


Ilustración 5: Creación de conjunto de datos

Se realiza la confirmación de la creación de las columnas. A continuación, podrán observar la **Ilustración #05**, donde se observará las columnas e información que contiene la tabla:

Consulta sin título [EJECUTAR] [GUARDAR] [DESCARGAR] [COMPARTIR] [Completada]

1 SELECT \* FROM `elevated-nature-452614-k1.dwh\_transacciones.mi\_tabla` LIMIT 1000

Presiona Alt+F1 para ver las opciones de accesibilidad.

Resultados de la consulta [GUARDAR LOS RESULTADOS] [ABRIR EN]

	INFORMACIÓN DEL TRABAJO	RESULTADOS	GRÁFICO	JSON	DETALLES DE LA EJECUCIÓN	GRÁFICO DE
Fila	id_transaccion	monto	fecha	cliente_id		
1	7708300906	2418887	2024-02-28 00:01:12 UTC	Cliente710 Méndez		
2	7113152307	4352281	2024-02-28 00:05:28 UTC	Cliente487 Torres		
3	4404409953	2372382	2024-02-28 00:06:39 UTC	Cliente614 Méndez		
4	3230746595	3108478	2024-02-28 00:09:15 UTC	Cliente287 Ramírez		
5	7963360136	3438835	2024-02-28 00:11:07 UTC	Cliente542 Méndez		
6	5585072061	1422996	2024-02-28 00:19:05 UTC	Cliente25 Gómez		
7	9613935886	925256	2024-02-28 00:19:18 UTC	Cliente155 Pérez		
8	8655821105	4415588	2024-02-28 00:27:17 UTC	Cliente955 Pérez		
9	4405883856	581832	2024-02-28 00:30:32 UTC	Cliente279 Gómez		
10	6595029391	2497427	2024-02-28 00:31:40 UTC	Cliente602 Pérez		

Resultados por página: 50 1 - 50 de 480

Ilustración 6: Visualización de campos



## 2.4. Optimización del Rendimiento Mediante Particionamiento y Clustering en la Base de Datos

Para garantizar que el proceso sea escalable y eficiente, implemente la técnica de particionamiento y clustering en la base de datos. Estas estrategias mejoran significativamente el rendimiento de las consultas y optimizan los costos operativos.

### 2.4.1. Particionamiento por Fecha

Se segmentaron las tablas grandes en partes más pequeñas, basadas en una clave de partición (por ejemplo, fecha). Esto mejora la consulta de grandes volúmenes de datos, ya que solo se accede a las particiones relevantes, reduciendo así el tiempo de procesamiento y los costos asociados.

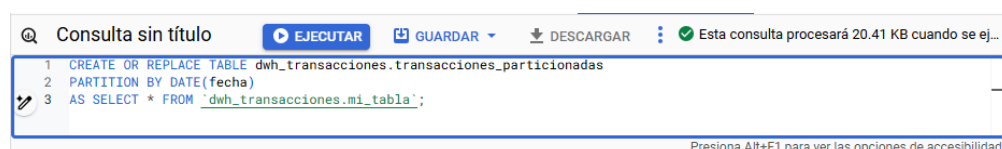


Ilustración 7: Creación Particionamiento

### 2.4.2. Clustering por Cliente

Se agrupan los datos dentro de cada partición según ciertas columnas clave, lo que optimiza las búsquedas y mejora la eficiencia de las consultas al reducir la cantidad de datos leídos por la base de datos.

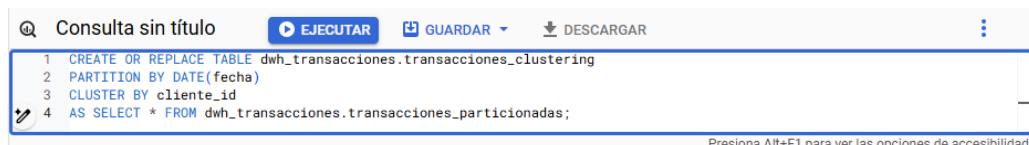


Ilustración 8: Creación Clustering

Con estas técnicas, conseguimos no solo un procesamiento más rápido, sino también un control más preciso sobre los costos, optimizando la infraestructura en la nube a medida que aumentan los volúmenes de datos. A continuación, podrá observar las transacciones anteriormente relacionadas:

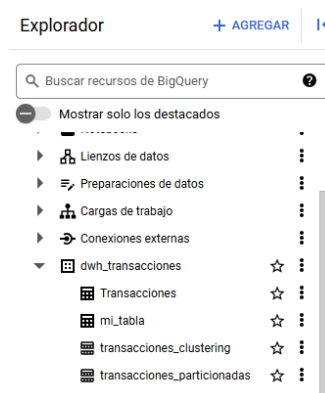


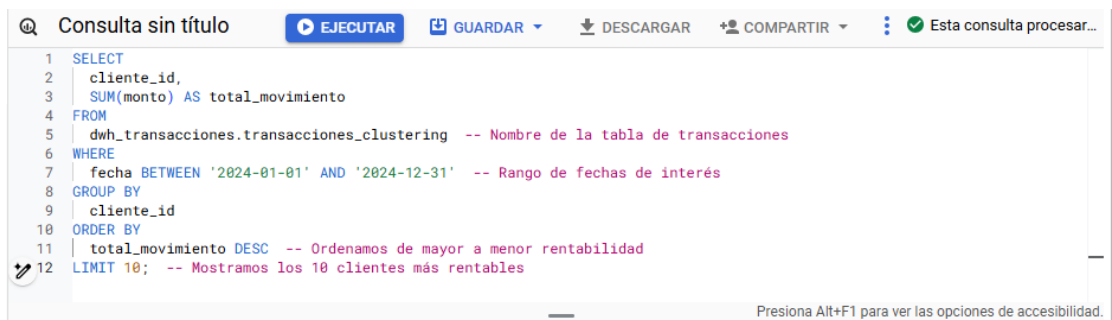
Ilustración 9: Confirmación creación tablas

### Parte 3: Explotación y Acceso a los Datos (Práctico - Análisis)

Para desarrollar una consulta en el Data Warehouse (DWH) que identifique a los clientes más rentables basándose en su historial transaccional, es necesario sumar las transacciones de cada cliente y calcular el monto total que ha movido o gastado. En este caso se tendrá en cuenta lo siguiente:

- **cliente\_id**: Identificador único del cliente.
- **monto**: Monto de cada transacción (positivo para depósitos, negativo para retiros).
- **fecha**: Fecha de la transacción.

El objetivo es calcular el **total de las transacciones** de cada cliente en un período específico y ordenar los resultados de mayor a menor rentabilidad.



```
1 SELECT
2   cliente_id,
3   SUM(monto) AS total_movimiento
4 FROM
5   dwh_transacciones.transacciones_clustering -- Nombre de la tabla de transacciones
6 WHERE
7   fecha BETWEEN '2024-01-01' AND '2024-12-31' -- Rango de fechas de interés
8 GROUP BY
9   cliente_id
10 ORDER BY
11   total_movimiento DESC -- Ordenamos de mayor a menor rentabilidad
12 LIMIT 10; -- Mostramos los 10 clientes más rentables
```

Ilustración 10: Consulta para conocer los clientes más rentables

A continuación, presento los clientes más rentables de acuerdo con sus transacciones:

Posición	Cliente	Total, Movido (\$)
1	Cliente 449 Pérez	4,996,841
2	Cliente 727 Gómez	4,995,527
3	Cliente 161 Torres	4,984,686
4	Cliente 497 Ramírez	4,974,210
5	Cliente 519 Gómez	4,974,122
6	Cliente 989 Torres	4,969,307
7	Cliente 267 Gómez	4,932,390
8	Cliente 767 Ramírez	4,927,921
9	Cliente 527 Gómez	4,912,134
10	Cliente 675 Méndez	4,898,674

Tabla 1: Clientes más rentables

- El Cliente 449 Pérez es el cliente con el mayor monto total movido, con \$4,996,841.
- Los clientes son listados de mayor a menor según el monto total de las transacciones realizadas en el periodo de tiempo analizado.

- Estos datos son clave para identificar a los clientes que generan más volumen financiero y, por lo tanto, pueden ser considerados como los más rentables.

A continuación, presento otro ejemplo donde se tienen en cuenta el total de las transacciones y el total de movimientos. Es importante observar que, en este caso, todos los clientes tienen solo una transacción, por lo que el promedio de transacciones es igual al total de la transacción para cada cliente.

```

1 SELECT
2   cliente_id,
3   SUM(monto) AS total_movimiento,
4   COUNT(*) AS total_transacciones,
5   AVG(monto) AS promedio_transacciones
6 FROM
7   dwh_transacciones.transacciones_clustering
8 WHERE
9   fecha BETWEEN '2024-01-01' AND '2024-12-31'
10 GROUP BY
11   cliente_id
12 ORDER BY
13   total_movimiento DESC
14 LIMIT 10;

```

Ilustración 11: Consulta para conocer los clientes más rentables por movimientos y transacciones

A continuación, presento los clientes más rentables de acuerdo con sus transacciones y sus movimientos:

Posición	Cliente	Total, Movimiento (\$)	Total, Transacciones
1	Cliente 449 Pérez	4,996,841	1
2	Cliente 727 Gómez	4,995,527	1
3	Cliente 161 Torres	4,984,686	1
4	Cliente 497 Ramírez	4,974,210	1
5	Cliente 519 Gómez	4,974,122	1
6	Cliente 989 Torres	4,969,307	1
7	Cliente 267 Gómez	4,932,390	1
8	Cliente 767 Ramírez	4,927,921	1
9	Cliente 527 Gómez	4,912,134	1
10	Cliente 675 Méndez	4,898,674	1

Tabla 2: Clientes por movimientos y transacción

Finalmente, presento un ejemplo de optimización con clustering, donde se analiza la rentabilidad de un único cliente durante el año 2024.

```

1 SELECT
2   cliente_id,
3   SUM(monto) AS total_rentabilidad
4 FROM
5   dwh_transacciones.transacciones_clustering
6 WHERE
7   fecha BETWEEN '2024-01-01' AND '2024-12-31'
8   AND cliente_id = 'Cliente449 Pérez' -- Filtro que aprovecha el clustering
9 GROUP BY
10  cliente_id
11 ORDER BY
12  total_rentabilidad DESC;

```

Ilustración 12: Rentabilidad por cliente

Los datos sobre los **clientes más rentables** pueden ser utilizados en productos financieros de varias maneras clave:

- **Personalización de productos:** Ofrecer productos financieros adaptados a las necesidades de los clientes rentables, como **créditos personalizados** o **tarjetas premium**.
- **Beneficios exclusivos:** Proporcionar **descuentos** o **recompensas** exclusivas para retener a los clientes más valiosos.
- **Segmentación de clientes:** Crear productos financieros específicos basados en el comportamiento de los clientes rentables, como **tarjetas internacionales** o **cuentas de ahorro globales**.
- **Desarrollo de nuevos productos:** Identificar patrones de comportamiento para diseñar nuevos productos que atraigan a clientes similares.
- Esta optimización mejora la eficiencia en el acceso a los datos, facilitando el análisis detallado y permitiendo tomar decisiones estratégicas basadas en el comportamiento transaccional de dicho cliente.

Esto ayuda a mejorar la **retención de clientes**, **optimizar ofertas** y **maximizar la rentabilidad** de la institución financiera.