

PROYECTO DE EVALUACIÓN - DATA ANALYST JR - GRUPO 02

URL DE GITHUB: https://github.com/hral-work/PROYECTO_RSM-KODIGO_DAJ10.git

ALUMNOS:

Katia Elizabeth Martínez - k00002733.

Rafael Alexander Martínez - k00002735.

Javier Antonio Valle - k00002730.

Hugo Robin Aparicio - k00002729.

ENTREGABLES DEL AVANCE 2: 28 de noviembre de 2024.

Análisis exploratorio de datos | Creación de dashboard.

1. GENERALIDADES DEL PROYECTO Y REQUERIMIENTOS

Como proyecto final, los alumnos del bootcamp tienen que realizar un proyecto que consiste en optimizar las ventas de una tienda en línea mediante el análisis de datos, se debe diseñar una base de datos y analizar la información de ventas, clientes y productos.

El proyecto se divide en varias fases, como se indica a continuación:

1. **Diseño de Base de Datos:**
Crear un diagrama entidad-relación (ERD), definir atributos clave y relaciones, y transformar el ERD en sentencias SQL para implementar las tablas en un DBMS.
2. **Extracción y Manipulación de Datos:**
Importar datos de archivos CSV a la base de datos, realizar validaciones y escribir consultas SQL para obtener información relevante.
3. **Análisis Exploratorio de Datos:**
Usar estadísticas descriptivas y técnicas de análisis para identificar patrones, tendencias y anomalías en las ventas.
4. **Creación de Dashboard:**
Desarrollar un dashboard interactivo en Power BI para visualizar KPIs, tendencias de ventas, y análisis de cestas de compra.
5. **Modelo Predictivo (Opcional):**
Construir y evaluar un modelo de machine learning para prever tendencias de ventas futuras.
6. **Reporte y Presentación:**
Preparar un informe escrito y una presentación para resumir los hallazgos, métodos utilizados y recomendaciones basadas en el análisis de datos. Además, se incluyen buenas prácticas de codificación y optimización de consultas SQL, así como la documentación y entrega del proyecto en GitHub.

El objetivo final es proporcionar estrategias basadas en datos para mejorar las ventas del cliente.

2. DATOS RELEVANTES, DESCRIPCION Y REVISION DE LOS INSUMOS

Por tanto, iniciando de lo más básico, en la tabla número 1, se describen los datos y el contenido de los tres Data Set que han sido proporcionados para proyecto (la revisión se hizo explorando el archivo en texto plano).

2.1 Descripción del contenido

Nombre del Data Set	Descripción
Data Set Clientes	Este dataset contiene información sobre los clientes de una empresa, con los siguientes campos: <ul style="list-style-type: none">• ClienteID: Identificador único del cliente.• NombreCliente: Nombre del cliente.• Email: Correo electrónico del cliente.• Telefono: Número de teléfono del cliente.• Direccion: Dirección del cliente.
Data Set Productos	Este dataset incluye información sobre los productos ofrecidos por la empresa: <ul style="list-style-type: none">• ProductoID: Identificador único del producto.• NombreProducto: Nombre del producto.• Categoria: Categoría a la que pertenece el producto.• PrecioUnitario: Precio unitario del producto.
Data Set Ventas	Este dataset contiene registros de ventas, con los siguientes campos: <ul style="list-style-type: none">• VentaID: Identificador único de la venta.• ClienteID: Identificador del cliente que realizó la compra.• ProductoID: Identificador del producto comprado.• Cantidad: Cantidad del producto comprado.• FechaVenta: Fecha en que se realizó la venta.• Region: Región donde se realizó la venta.

Tabla 1, Descripción del contenido de los Data Set del proyecto.

2.2 Relevancia de cada Data Set

Clientes:

Este dataset es crucial para entender quiénes son los clientes de la empresa. Es la base para cualquier análisis de comportamiento del cliente, segmentación de mercado, y estrategias de marketing personalizadas.

Productos:

Es fundamental para el análisis de inventarios, planificación de estrategias de precios, y análisis de ventas por categoría de producto.

Ventas:

Este dataset es esencial para el análisis de rendimiento de ventas, tendencias de compra, y evaluación del impacto de las campañas de marketing.

2.3 Hallazgos Clave y Errores a Revisar

Clientes:

- Duplicados: Verificar si hay clientes duplicados, especialmente en el campo Email o Teléfono.
- Campos Vacíos o Nulos: Revisar que todos los campos estén completos.
- Formatos: Asegurar que los correos electrónicos y números de teléfono tengan el formato correcto.

Productos:

- Precios Negativos o Inválidos: Verificar que todos los precios sean positivos y válidos.
- Categorías Correctas: Asegurar que cada producto esté correctamente categorizado.
- Duplicados: Revisar que no haya productos duplicados.

Ventas:

- Fechas Inválidas: Verificar que todas las fechas sean válidas y en el formato correcto.
- Inconsistencias en IDs: Asegurar que ClienteID y ProductoID correspondan a registros válidos en los otros datasets.
- Regiones Correctas: Verificar que las regiones sean válidas y consistentes.

3. DISEÑO DE LA BASE DE DATOS

3.1 Diagrama Entidad-Relación

Para la creación del diagrama entidad-relación (ERD) con base en los datasets proporcionados ("clientes", "productos" y "ventas"), identificamos las entidades principales, sus atributos, y las relaciones entre ellas. A continuación, se detalla cómo se puede construir el ERD y qué elementos clave se deben considerar.

Entidades y Atributos

1. Clientes

- o ClienteID (llave primaria)
- o NombreCliente
- o Email
- o Telefono
- o Direccion

2. Productos

- o ProductoID (llave primaria)
- o NombreProducto
- o Categoria
- o PrecioUnitario

3. Ventas

- o VentaID (llave primaria)
- o ClienteID (llave foránea)
- o ProductoID (llave foránea)
- o Cantidad
- o FechaVenta
- o Región

Relaciones

1. Relación entre Clientes y Ventas:
 - o Un cliente puede tener asociadas muchas ventas.
 - o Cardinalidad: Uno a Muchos (1:N) desde Clientes a Ventas.
 - o ClienteID en la tabla Ventas es una llave foránea que referencia a ClienteID en la tabla Clientes.
2. Relación entre Productos y Ventas:
 - o Un producto puede estar asociado con muchas ventas.
 - o Cardinalidad: Uno a Muchos (1:N) desde Productos a Ventas.
 - o ProductoID en la tabla Ventas es una llave foránea que referencia a ProductoID en la tabla Productos.

Llaves y Cardinalidad

- Llaves Primarias: ClienteID en Clientes, ProductoID en Productos, y VentaID en Ventas son llaves primarias que identifican de manera única a cada registro en sus respectivas tablas.
- Llaves Foráneas: ClienteID y ProductoID en la tabla Ventas son llaves foráneas que crean relaciones entre las tablas, permitiendo la integridad referencial y conexiones lógicas entre datos.
- Cardinalidad: Definida por las relaciones 1:N, ya que un cliente puede tener múltiples ventas y un producto puede ser vendido en múltiples transacciones.

Diagrama Visual

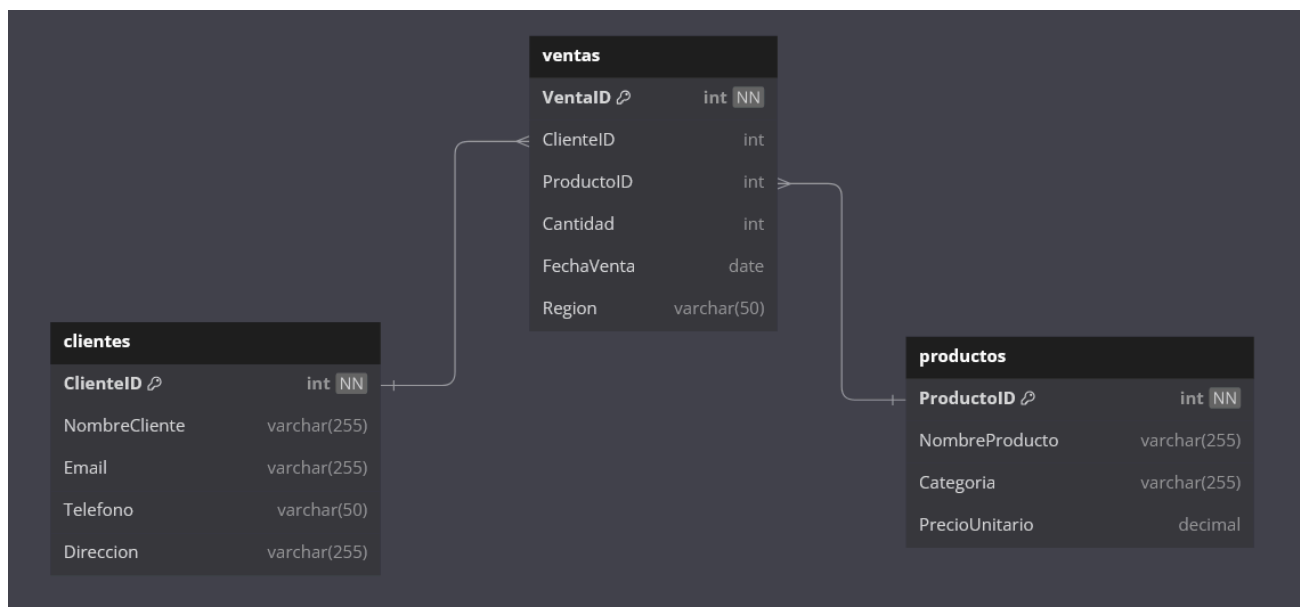


Diagrama 1, diagrama Entidad-Relación.

Para generar el diagrama anterior se utilizó el lenguaje DBML (Database Markup Language):

https://dbdiagram.io/d/PROYECTO_FINAL-673c1b28e9daa85acaea168c

Unset

```
Table clientes {
  ClienteID int [pk, not null]
  NombreCliente varchar(255)
  Email varchar(255)
  Telefono varchar(50)
  Direccion varchar(255)
}

Table productos {
  ProductoID int [pk, not null]
  NombreProducto varchar(255)
  Categoria varchar(255)
  PrecioUnitario decimal
}

Table ventas {
  VentaID int [pk, not null]
  ClienteID int [ref: > clientes.ClienteID]
  ProductoID int [ref: > productos.ProductoID]
  Cantidad int
  FechaVenta date
  Region varchar(50)
}
```

3.2 Implementación en el DBMS

Previo al análisis y normalización de datos es necesario que se ejecute la creación de la base de datos y tablas, una vez lista esa parte, se procede a cargar la información de los Datasets en el ambiente habilitado. Se decidió utilizar como motor de base datos SQLServer 2019 por la facilidad que se tiene en cuanto a disponibilidad y acceso al uso de un ambiente de desarrollo.

Los pasos requeridos en esta fase pueden resumirse así:

1. Crear la Base de Datos: Crear la base de datos RSMDB.
2. Crear Usuario y Roles: Crear un usuario rsmuser y asignarle roles de lectura y escritura.
3. Crear las Tablas: Crear las tablas clientes, productos y ventas con sus respectivas relaciones.
4. Verificar: Asegurar que las tablas y relaciones se han creado correctamente mediante consultas de verificación.

SCRIPT creacion_db_user_tablas.sql:

```
Unset
-- 1. Creación de la Base de Datos
CREATE DATABASE RSMDB;
GO

-- 2. Creación del Usuario y Asignación de Permisos
USE RSMDB;
GO

-- Crear usuario
CREATE USER rsmuser WITH PASSWORD = 'password';
GO

-- Crear roles
CREATE ROLE db_datareader;
CREATE ROLE db_datawriter;
GO

-- Asignar permisos al usuario
ALTER ROLE db_datareader ADD MEMBER rsmuser;
ALTER ROLE db_datawriter ADD MEMBER rsmuser;
GO

-- 3. Creación de las Tablas
CREATE TABLE clientes (
    ClienteID int PRIMARY KEY NOT NULL,
    NombreCliente varchar(255),
    Email varchar(255),
    Telefono varchar(50),
    Direccion varchar(255)
);
GO

CREATE TABLE productos (
    ProductoID int PRIMARY KEY NOT NULL,
    NombreProducto varchar(255),
    Categoria varchar(255),
```

```
PrecioUnitario decimal(10, 2)
);
GO

CREATE TABLE ventas (
    VentaID int PRIMARY KEY NOT NULL,
    ClienteID int,
    ProductoID int,
    Cantidad int,
    FechaVenta date,
    Region varchar(50),
    FOREIGN KEY (ClienteID) REFERENCES clientes (ClienteID),
    FOREIGN KEY (ProductoID) REFERENCES productos (ProductoID)
);
GO
```

4. **EXTRACCION Y MANIPULACION DE DATOS**

4.1 Importación de datos (paso inicial necesario)

SQL Server ofrece un asistente de Importación y Exportación para cargar datos desde archivos CSV a las tablas de SQL Server.

1. Abrir SQL Server Management Studio (SSMS).
2. Conectarse a la instancia de SQL Server.
3. Hacer clic derecho en la base de datos RSMDB y seleccionar Tasks -> Import Data.
4. Elegir el origen de datos:
 - o Source: Flat File Source
 - o File name: Seleccionamos el archivo CSV correspondiente (clientes.csv, productos.csv, o ventas.csv).
 - o Configuramos los delimitadores y otros parámetros según sea necesario.
5. Elegir el destino de datos:
 - o Destination: SQL Server Native Client
 - o Server name: SQLSRV.
 - o Database: RSMDB.
6. Configurar las opciones de mapeo de columnas para asegurar que las columnas de los archivos CSV correspondan correctamente a las columnas de las tablas SQL.
7. Ejecutar el proceso de importación.

Una vez que los datos se hayan importado, verifica que estén correctamente cargados ejecutando algunas consultas básicas en la base de datos:

–Verificar que las tablas estén pobladas

Unset

```
SELECT * FROM clientes;  
SELECT * FROM productos;  
SELECT * FROM ventas;
```

Como buenas prácticas, durante y posteriormente a la importación de datos se recomienda aplicar estos controles:

- Limpieza de Datos: Asegúrate de eliminar duplicados, manejar valores nulos y corregir errores en los datos.
- Integridad Referencial: Garantizar que todas las llaves foráneas correspondan a registros válidos en sus tablas de origen.
- Normalización: Mantener una estructura de datos normalizada para evitar redundancias y asegurar un almacenamiento eficiente.
- Indexación: Crear índices en las columnas utilizadas frecuentemente en las consultas para mejorar el rendimiento.

4.2 Consultas SQL para Extracción de Información

Las siguientes, son consultas SQL para extraer la información solicitada, basadas en los datos de los datasets y el procedimiento de importación descritos anteriormente:

1. Ventas Totales por Categoría de Producto (ventas_totales_por_categoria.sql)

Esta consulta calcula la suma de las ventas agrupadas por categoría de producto.

Unset

```
SELECT  
    p.Categoria,  
    SUM(v.Cantidad * p.PrecioUnitario) AS VentasTotales  
FROM  
    ventas v  
JOIN  
    productos p ON v.ProductoID = p.ProductoID  
GROUP BY  
    p.Categoria;
```


2. Clientes con Mayor Valor de Compra (clientes_mayor_valor_compra.sql)

Esta consulta calcula el total gastado por cada cliente y ordena los resultados de mayor a menor.

```
Unset
SELECT
    c.ClienteID,
    c.NombreCliente,
    SUM(v.Cantidad * p.PrecioUnitario) AS TotalGastado
FROM
    ventas v
JOIN
    clientes c ON v.ClienteID = c.ClienteID
JOIN
    productos p ON v.ProductoID = p.ProductoID
GROUP BY
    c.ClienteID, c.NombreCliente
ORDER BY
    TotalGastado DESC;
```

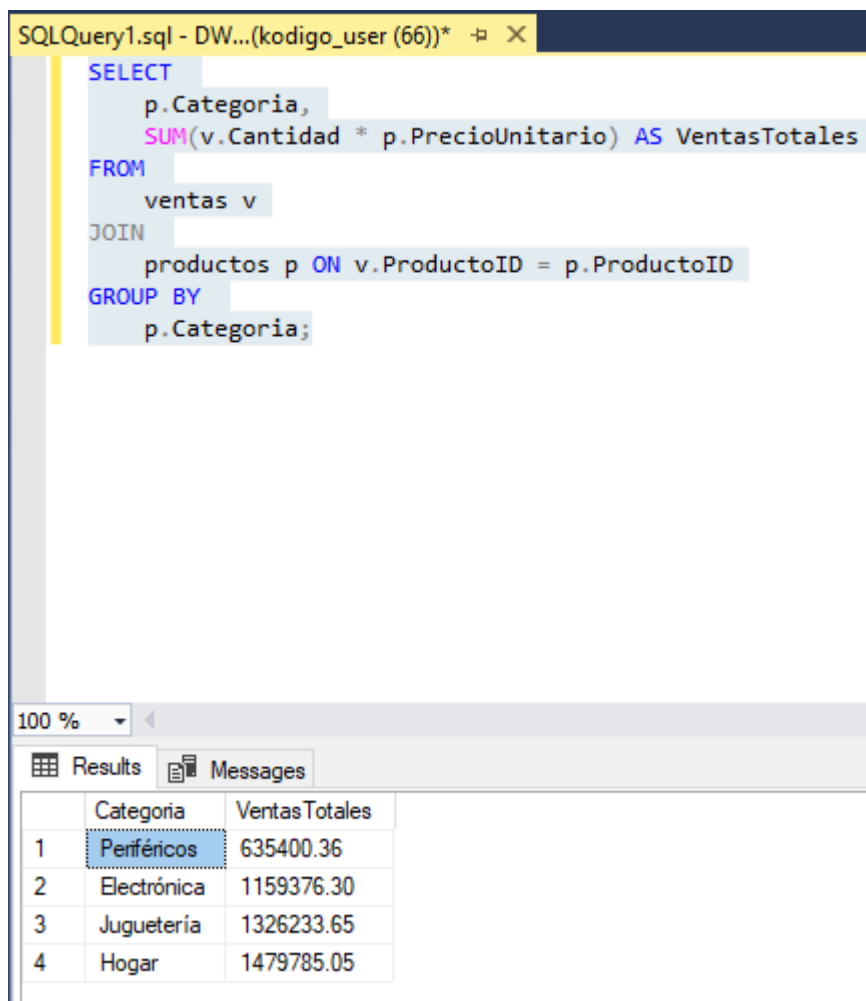
3. Productos Más Vendidos por Región (productos_mas_vendidos_por_region.sql)

Esta consulta determina los productos más populares en cada región, basándose en la cantidad total vendida.

```
Unset
SELECT
    Region,
    NombreProducto,
    CantidadTotalVendida
FROM (
    SELECT
        v.Region,
        p.NombreProducto,
        SUM(v.Cantidad) AS CantidadTotalVendida,
        ROW_NUMBER() OVER (PARTITION BY v.Region ORDER BY SUM(v.Cantidad)
        DESC) AS rn
    FROM
```

```
        ventas v
    JOIN
        productos p ON v.ProductoID = p.ProductoID
    GROUP BY
        v.Region, p.NombreProducto
) AS ventas_rankeadas
WHERE
    ventas_rankeadas.rn <= 3
ORDER BY
    Region, CantidadTotalVendida DESC;
```

Resultado de consulta ventas_totales_por_categoria.sql:



The screenshot shows a SQL Query Editor window titled "SQLQuery1.sql - DW...(kodigo_user (66))". The query is as follows:

```
SELECT
    p.Categoria,
    SUM(v.Cantidad * p.PrecioUnitario) AS VentasTotales
FROM
    ventas v
JOIN
    productos p ON v.ProductoID = p.ProductoID
GROUP BY
    p.Categoria;
```

Below the query editor, the "Results" tab is active, displaying a table with the following data:

	Categoria	VentasTotales
1	Periféricos	635400.36
2	Electrónica	1159376.30
3	Juguetería	1326233.65
4	Hogar	1479785.05

Resultado de la consulta clientes_mayor_valor_compra.sql:

```
--Esta consulta calcula el total gastado por cada cliente y ordena los resultados de mayor a menor.
SELECT
    c.ClienteID,
    c.NombreCliente,
    SUM(v.Cantidad * p.PrecioUnitario) AS TotalGastado
FROM
    ventas v
JOIN
    clientes c ON v.ClienteID = c.ClienteID
JOIN
    productos p ON v.ProductoID = p.ProductoID
GROUP BY
    c.ClienteID, c.NombreCliente
ORDER BY
    TotalGastado DESC;
```

100 %

Results Messages

	ClienteID	NombreCliente	TotalGastado
1	1043	Cliente 43	110127.57
2	1063	Cliente 63	94780.25
3	1005	Cliente 5	85356.86
4	1070	Cliente 70	82935.08
5	1030	Cliente 30	81736.62
6	1095	Cliente 95	81139.11
7	1024	Cliente 24	75599.88
8	1058	Cliente 58	75347.86
9	1075	Cliente 75	72330.36
10	1098	Cliente 98	71991.52
11	1010	Cliente 10	69773.41
12	1061	Cliente 61	68949.07
13	1041	Cliente 41	68728.42
14	1073	Cliente 73	68687.97
15	1022	Cliente 22	68026.40
16	1027	Cliente 27	66659.18
17	1082	Cliente 82	65104.78
18	1035	Cliente 35	63724.44
19	1085	Cliente 85	63545.22
20	1029	Cliente 29	62798.98
21	1006	Cliente 6	61443.68
22	1031	Cliente 31	61443.18
23	1091	Cliente 91	61242.86
24	1062	Cliente 62	61117.13
25	1037	Cliente 37	60240.61
26	1078	Cliente 78	60234.20
27	1066	Cliente 66	60035.61
28	1011	Cliente 11	59817.61
29	1016	Cliente 16	58591.13
30	1039	Cliente 39	57414.60

Query executed successfully.

Resultado de la consulta productos_mas_vendidos_por_region.sql:

```
SELECT
    Region,
    NombreProducto,
    CantidadTotalVendida
FROM (
    SELECT
        v.Region,
        p.NombreProducto,
        SUM(v.Cantidad) AS CantidadTotalVendida,
        ROW_NUMBER() OVER (PARTITION BY v.Region ORDER BY SUM(v.Cantidad) DESC) AS rn
    FROM
        ventas v
    JOIN
        productos p ON v.ProductoID = p.ProductoID
    GROUP BY
        v.Region, p.NombreProducto
) AS ventas_rankeadas
WHERE
    ventas_rankeadas.rn <= 3
ORDER BY
    Region, CantidadTotalVendida DESC;
```

100 %

Results Messages

	Region	NombreProducto	CantidadTotalVendida
1	Este	Producto 5	75
2	Este	Producto 6	69
3	Este	Producto 10	69
4	Norte	Producto 1	72
5	Norte	Producto 15	71
6	Norte	Producto 11	64
7	Oeste	Producto 13	73
8	Oeste	Producto 28	73
9	Oeste	Producto 1	63
10	Sur	Producto 12	84
11	Sur	Producto 9	62
12	Sur	Producto 23	61

✓ Query executed successfully.

5. ANÁLISIS EXPLORATORIO DE DATOS

5.1 Análisis Estadístico Descriptivo

Resumen del proceso que realiza el código:

1. **Preparar el Entorno:** Instalamos las librerías necesarias.
2. **Cargar los Datos:** Leer los archivos CSV para generar DataFrames.
3. **Calcular Estadísticas Básicas:** Calcula medias, medianas, desviaciones estándar y otras estadísticas descriptivas.
4. **Identificar Variables Importantes:** Calcular el número de transacciones y ventas promedio por cliente.
5. **Analizar la Distribución de las Ventas:** Analizar distribuciones de ventas.

Nombre del script: 5_1_analisis_estadistico_descriptivo_v4.py

Python

Paso 1: Preparar el Entorno

Se debe tener las librerías necesarias instaladas. Pueden ser instaladas usando pip:

pip install pandas numpy matplotlib

import pandas as pd # Importa la librería de manipulación de datos Pandas

import numpy as np # Importa la librería NumPy para operaciones numéricas

import matplotlib.pyplot as plt # Importa la librería de visualización

Matplotlib

Paso 2: Cargar los Datos

Leer los archivos CSV en DataFrames de pandas

Cargar los archivos CSV en un DataFrame

clientes = pd.read_csv('clientes.csv')

productos = pd.read_csv('productos.csv')

ventas = pd.read_csv('ventas.csv')

Paso 3: Calcular Estadísticas Básicas

Calcular estadísticas básicas como medias, medianas y desviaciones estándar

```
# Estadísticas descriptivas para ventas
ventas_descriptivas = ventas.describe().round(2) # Genera estadísticas
descriptivas para el DataFrame 'ventas' y redondea a 2 decimales

# Media y mediana
media_ventas = ventas['Cantidad'].mean() # Calcula la media de la columna
'Cantidad' en 'ventas'
mediana_ventas = ventas['Cantidad'].median() # Calcula la mediana de la
columna 'Cantidad' en 'ventas'
desviacion_estandar_ventas = ventas['Cantidad'].std() # Calcula la
desviación estándar de la columna 'Cantidad' en 'ventas'

print("Paso 3: Calcular Estadísticas Básicas")
print("Estadísticas Básicas de Ventas:")
print(ventas_descriptivas)
print(f"Media de Ventas: {media_ventas}")
print(f"Mediana de Ventas: {mediana_ventas}")
print(f"Desviación Estándar de Ventas: {desviacion_estandar_ventas}")

# Paso 4: Identificar Variables Importantes
# Calcular el número de transacciones, ventas promedio por cliente, etc.

# Número de transacciones
num_transacciones = ventas.shape[0] # Calcula el número de transacciones
(filas) en el DataFrame 'ventas'

# Ventas promedio por cliente
ventas_por_cliente = ventas.groupby('ClienteID')['Cantidad'].sum() #
Agrupar las ventas por 'ClienteID' y suma la 'Cantidad' vendida
ventas_promedio_cliente = ventas_por_cliente.mean() # Calcula la media de
las ventas por cliente

print("Paso 4: Identificar Variables Importantes")
```

```
print(f"Número de Transacciones: {num_transacciones}")
print(f"Ventas Promedio por Cliente: {ventas_promedio_cliente}")

# Paso 5: Analizar la Distribución de las Ventas
# Analizar la distribución de las ventas para detectar patrones
# estacionales o concentraciones en determinados productos

# Distribución de ventas por producto
ventas_por_producto = ventas.groupby('ProductoID')['Cantidad'].sum() #
Agrupar las ventas por 'ProductoID' y suma la 'Cantidad' vendida

# Gráfica de distribución de ventas por producto
plt.figure(figsize=(10, 6)) # Crea una nueva figura con un tamaño
específico
ventas_por_producto.plot(kind='bar') # Genera un gráfico de barras con
los datos de ventas por producto
plt.title('Paso 5 Analizar la Distribución de las Ventas - Distribución de
Ventas por Producto') # Añade un título al gráfico
plt.xlabel('ProductoID') # Etiqueta el eje x como 'ProductoID'
plt.ylabel('Cantidad Vendida (Unidades)') # Etiqueta el eje y como
'Cantidad Vendida (Unidades)'
plt.show() # Muestra el gráfico

# Análisis de ventas estacionales (FechaVenta está en formato
'YYYY-MM-DD')
ventas['FechaVenta'] = pd.to_datetime(ventas['FechaVenta']) # Convierte
la columna 'FechaVenta' a un tipo de dato datetime
ventas['Mes'] = ventas['FechaVenta'].dt.month # Crea una nueva columna
'Mes' basada en el mes de 'FechaVenta'
ventas_por_mes = ventas.groupby('Mes')['Cantidad'].sum() # Agrupa las
ventas por 'Mes' y suma la 'Cantidad' vendida

# Gráfica de ventas por mes
```

```
plt.figure(figsize=(10, 6)) # Crea una nueva figura con un tamaño
específico
ventas_por_mes.plot(kind='line') # Genera un gráfico de líneas con los
datos de ventas por mes
plt.title('Paso 5: Analizar la Distribución de las Ventas - Ventas por
Mes') # Añade un título al gráfico
plt.xlabel('Mes') # Etiqueta el eje x como 'Mes'
plt.ylabel('Cantidad Vendida (Unidades)') # Etiqueta el eje y como
'Cantidad Vendida (Unidades)'
plt.show() # Muestra el gráfico

# Paso 6: Resumir y Guardar Resultados
# Finalmente, guarda los resultados del análisis para su inclusión en el
informe final

# Guardar estadísticas básicas en un archivo CSV
ventas_descriptivas.to_csv('estadisticas_basicas_ventas.csv') # Guarda el
DataFrame 'ventas_descriptivas' en un archivo CSV

# Guardar resumen de ventas por cliente
ventas_por_cliente.to_csv('ventas_por_cliente.csv') # Guarda el DataFrame
'ventas_por_cliente' en un archivo CSV

# Guardar análisis de ventas por mes
ventas_por_mes.to_csv('ventas_por_mes.csv') # Guarda el DataFrame
'ventas_por_mes' en un archivo CSV

print("Análisis completado y resultados guardados.")
```

El resultado de este script es el siguiente:

Paso 3: Calcular Estadísticas Básicas

Estadísticas Básicas de Ventas:

	VentaID	ClienteID	ProductoID	Cantidad
count	1000.00	1000.00	1000.00	1000.00
mean	600.50	1049.97	515.24	4.94
std	288.82	29.01	8.71	2.58
min	101.00	1001.00	501.00	1.00
25%	350.75	1025.00	507.75	3.00
50%	600.50	1050.00	515.00	5.00
75%	850.25	1074.00	523.00	7.00
max	1100.00	1100.00	530.00	9.00

Media de Ventas: 4.936

Mediana de Ventas: 5.0

Desviación Estándar de Ventas: 2.576925405312413

Paso 4: Identificar Variables Importantes

Número de Transacciones: 1000

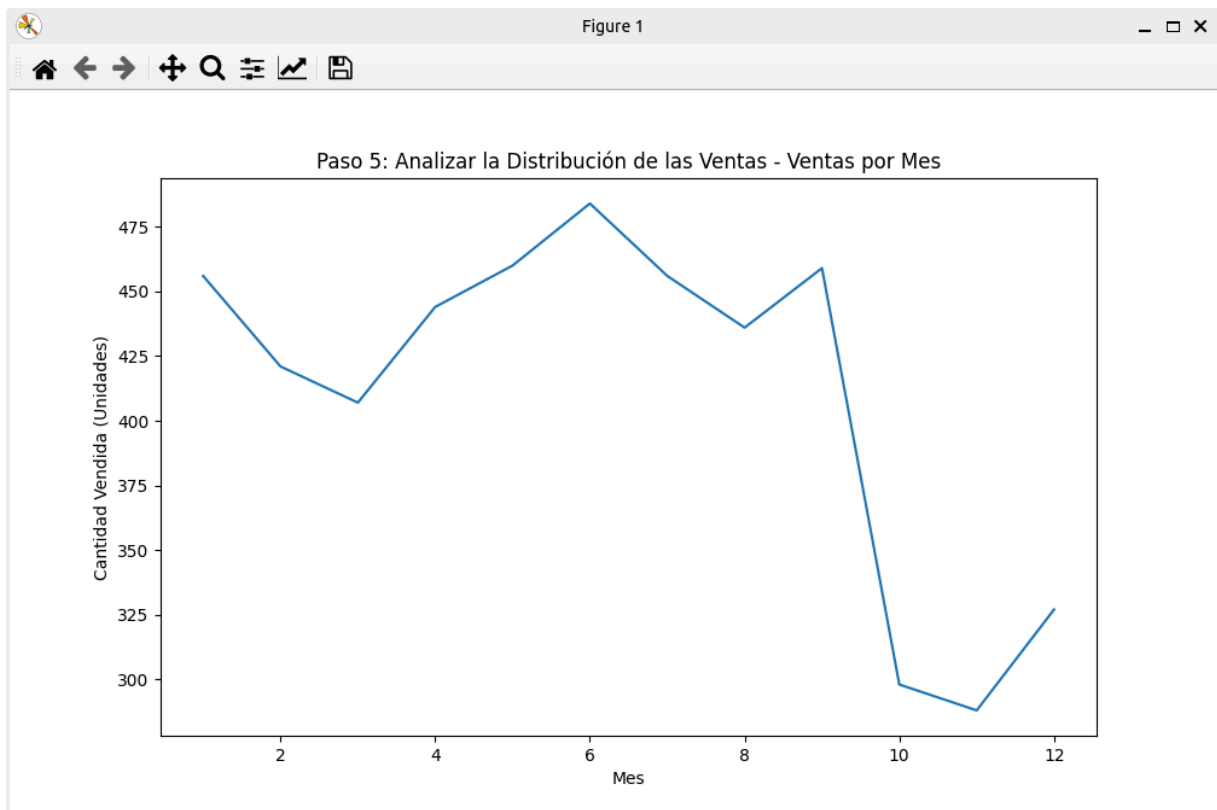
Ventas Promedio por Cliente: 49.36

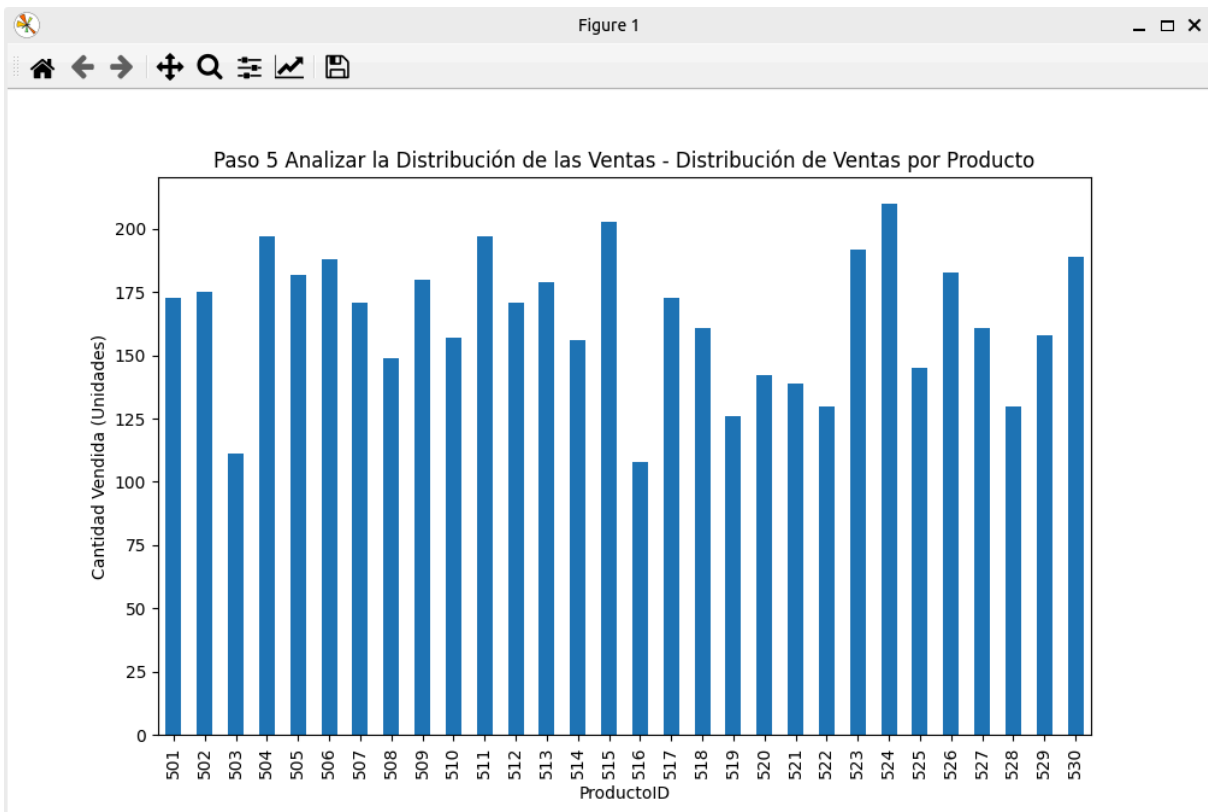
QSocketNotifier: Can only be used with threads started with QThread

qt.qpa.wayland: Wayland does not support QWindow::requestActivate()

qt.qpa.wayland: Wayland does not support QWindow::requestActivate()

Análisis completado y resultados guardados.





Interpretar los resultados.

1. Estadísticas Básicas de Ventas:

- **count:** El número de elementos que se tienen en las tablas es 1000 en todos los campos. Esto indica que hay 1000 registros de ventas.
- **mean:** La media (promedio) para cada campo, la media de la cantidad vendida es 4.936.
- **std:** La desviación estándar, mide la dispersión de datos respecto a la media. La desviación estándar es 2.576, lo que indica que las ventas varían alrededor de esta cantidad.
- **min:** El valor mínimo en cada campo. Por ejemplo, la venta más baja (en cantidad) es 1.
- **25% (1er Cuartil):** El valor por debajo del cual se encuentra el 25% de los datos. Para Cantidad, el 25% de las ventas son de 3 o menos unidades.
- **50% (Mediana):** El valor central en el conjunto de datos, por debajo y por encima del cual se encuentra el 50% de los datos. Para Cantidad, la mediana es 5, lo que significa que la mitad de las ventas son de 5 o menos unidades.
- **75% (3er Cuartil):** El valor por debajo del cual se encuentra el 75% de los datos es 7 o menos unidades.
- **Max:** El valor máximo en cada campo, la venta más alta (en cantidad) es 9.

2. Media de Ventas: 4.93

- Esta es la cantidad promedio de unidades vendidas por transacción. Indica que, en promedio, cada venta incluye aproximadamente 5 unidades de un producto.

3. Mediana de Ventas: 5.0

- La mediana nos dice que la mitad de las transacciones implican la venta de 5 unidades o menos, y la otra mitad implica la venta de más de 5 unidades. Esto puede ser útil para entender la distribución de ventas.

4. Desviación Estándar de Ventas: 2.57

- La desviación estándar nos muestra la variabilidad de las ventas. Una desviación estándar de 2.57 sugiere que la mayoría de las transacciones varían en torno a 2.57 unidades alrededor de la media (4.93 unidades).

5. Número de Transacciones: 1000

- Este es el número total de ventas registradas. Es un dato importante para entender el volumen de transacciones realizadas.

6. Ventas Promedio por Cliente: 49.36

- Esta estadística calcula el promedio de ventas totales por cliente. Indica que, en promedio, cada cliente ha comprado aproximadamente 49.36 unidades.

Qué significan estos resultados.

- **Distribución de Ventas:** La distribución de las ventas por cantidad muestra que la mayoría de las transacciones están concentradas alrededor de la media de 4.93 unidades, con una variabilidad moderada.
- **Ventas por Cliente:** La media de ventas por cliente puede ayudar a identificar qué tan valiosos son los clientes en promedio, lo que es crucial para estrategias de marketing y retención.
- **Variabilidad de Ventas:** La desviación estándar ayuda a entender la dispersión en los datos de ventas, lo que es útil para detectar ventas anómalas o patrones inusuales.
- **Tendencias de Ventas:** Conocer la media, mediana y la distribución de las ventas puede ayudar a planificar el inventario y optimizar las estrategias de precios y promociones.

Estos insights pueden proporcionar una base para tomar decisiones de negocio informadas y estratégicas.

5.2 Identificación de patrones, tendencias.**Procedimiento para este requerimiento:**

- 1. Preparar el Entorno:** Instalamos las librerías necesarias.
- 2. Cargar los Datos:** Lee los archivos CSV en DataFrames.
- 3. Analizar Series Temporales de Ventas:** Agrupar ventas por fecha y generar gráfica.
- 4. Análisis de Comportamiento de Compra de Clientes:** Ventas totales por cliente y su gráfica.
- 5. Productos con Bajas Ventas:** Imprime lista de productos con bajas ventas.

Nombre del script: 5_2_identificacion_patrones_v5.py

Python

```
# Paso 1: Preparar el Entorno
# Instala las librerías necesarias con el siguiente comando:
# pip install pandas numpy matplotlib
```

```
import pandas as pd # Importa la librería de manipulación de datos
Pandas
import numpy as np # Importa la librería NumPy para operaciones
numéricas
import matplotlib.pyplot as plt # Importa la librería de visualización
Matplotlib

# Paso 2: Cargar los Datos
# Leer los archivos CSV en DataFrames de pandas

# Cargar los archivos CSV
clientes = pd.read_csv('clientes.csv') # Carga el archivo CSV
'clientes.csv' en un DataFrame
productos = pd.read_csv('productos.csv') # Carga el archivo CSV
'productos.csv' en un DataFrame
ventas = pd.read_csv('ventas.csv') # Carga el archivo CSV 'ventas.csv'
en un DataFrame

# Paso 3: Analizar Series Temporales de Ventas
# Identificar tendencias ascendentes o descendentes

# Convertir FechaVenta a formato de fecha
ventas['FechaVenta'] = pd.to_datetime(ventas['FechaVenta']) #
Convierte la columna 'FechaVenta' a tipo datetime

# Agrupar ventas por fecha y sumar solo la columna 'Cantidad'
ventas_diarias =
ventas.groupby('FechaVenta')['Cantidad'].sum().reset_index() # Agrupa
las ventas por fecha y suma las cantidades
```

```
# Graficar ventas diarias con gráfico de barras
plt.figure(figsize=(15, 6)) # Tamaño de la figura
plt.bar(ventas_diarias['FechaVenta'], ventas_diarias['Cantidad'],
width=1.0) # Genera un gráfico de barras para las ventas diarias
plt.title('Ventas Diarias') # Añade un título al gráfico
plt.xlabel('Fecha') # Etiqueta el eje x como 'Fecha'
plt.ylabel('Cantidad Vendida (Unidades)') # Etiqueta el eje y como
'Cantidad Vendida (Unidades)'
plt.xticks(rotation=45) # Rota las etiquetas del eje x para mayor
legibilidad
plt.tight_layout() # Ajusta el diseño para que todo encaje bien
plt.show() # Muestra el gráfico
```

```
# Paso 4: Análisis de Comportamiento de Compra de Clientes
# Calcular y graficar el comportamiento de compra de los clientes
```

```
# Calcular ventas totales por cliente
ventas_por_cliente =
ventas.groupby('ClienteID')['Cantidad'].sum().reset_index() # Agrupa
las ventas por 'ClienteID' y suma las cantidades
```

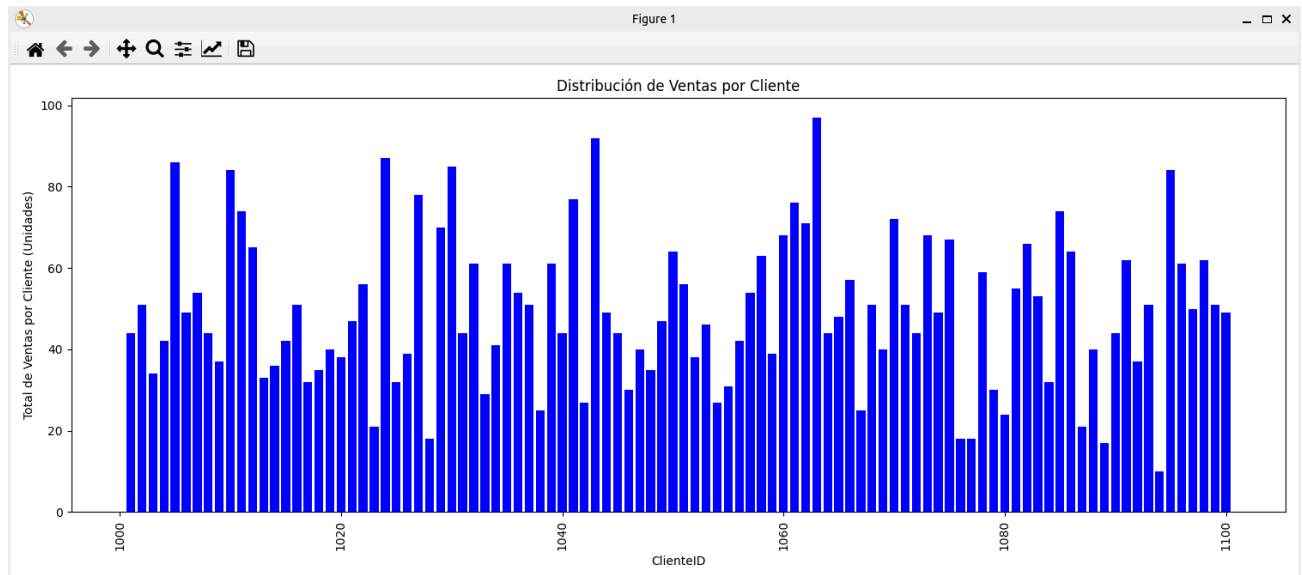
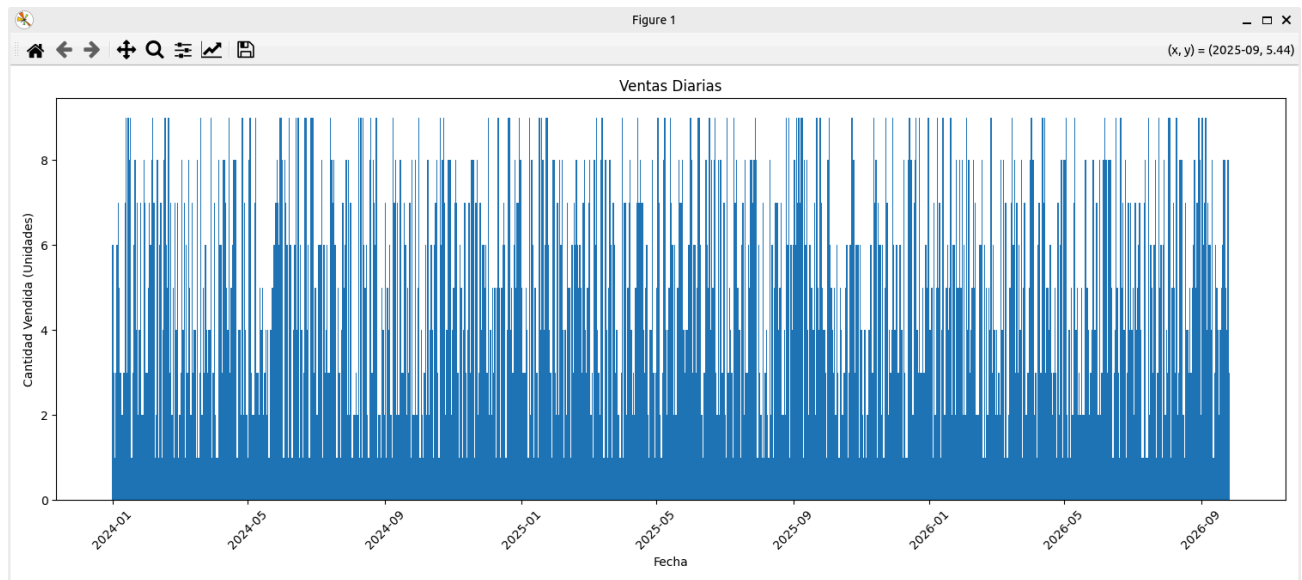
```
# Graficar la distribución de ventas por cliente con gráfico de barras
plt.figure(figsize=(15, 6)) # Tamaño de la figura
plt.bar(ventas_por_cliente['ClienteID'],
ventas_por_cliente['Cantidad'], color='blue') # Genera un gráfico de
barras para la distribución de ventas por cliente
plt.title('Distribución de Ventas por Cliente') # Añade un título al
gráfico
plt.xlabel('ClienteID') # Etiqueta el eje x como 'ClienteID'
```

```
plt.ylabel('Total de Ventas por Cliente (Unidades)') # Etiqueta el eje
y como 'Total de Ventas por Cliente (Unidades)'
plt.xticks(rotation=90) # Rota las etiquetas del eje x para mayor
legibilidad
plt.tight_layout() # Ajusta el diseño para que todo encaje bien
plt.show() # Muestra el gráfico

# Productos con bajas ventas
# Selecciona solo columnas numéricas antes de agrupar y sumar
ventas_por_producto =
ventas.select_dtypes(include=[np.number]).groupby('ProductoID').sum().r
eset_index() # Agrupa las ventas por 'ProductoID' y suma las
cantidades
productos_bajas_ventas =
ventas_por_producto[ventas_por_producto['Cantidad'] <
ventas_por_producto['Cantidad'].quantile(0.1)] # Identifica productos
con ventas por debajo del décimo percentil

print("Productos con bajas ventas:")
print(productos_bajas_ventas) # Imprime la lista de productos con
bajas ventas
```

El resultado de este script se muestra a continuación:



Productos con bajas ventas:

	ProductoID	VentaID	ClienteID	Cantidad
2	503	16767	29472	111
15	516	14291	27064	108
18	519	14971	26130	126

Resumen de Resultados y sus Implicaciones:**1. Análisis de Ventas Diarias:**

Gráfico de Barras: Visualización de las ventas diarias. Permite identificar tendencias ascendentes o descendentes en las ventas, ayudando a comprender los patrones de demanda y detectar posibles estacionalidades.

2. Comportamiento de Compra de Clientes:

Ventas Totales por Cliente: Calcula y grafica las ventas totales por cada cliente. Ayuda a identificar los clientes más valiosos y analizar su comportamiento de compra, lo que puede ser útil para estrategias de marketing y retención de clientes.

3. Productos con Bajas Ventas:

Identificación de Productos con Bajas Ventas: Generaliza los productos con ventas por debajo del décimo percentil. Permite identificar productos que no están vendiendo bien, lo cual es crucial para la optimización del inventario y las estrategias de producto.

Implicaciones Generales:

- **Optimización de Estrategias:** La visualización y el análisis de datos proporcionan información para la toma de decisiones en áreas como marketing, gestión de clientes y gestión de productos.
- **Detección de Tendencias:** Se pueden identificar patrones de ventas y comportamientos de compra ayuda a prever la demanda futura y a planificar.
- **Mejora de la Eficiencia:** Detectar productos con bajas ventas permite reducir costos de almacenamiento y enfocar recursos en productos más rentables.

5.3 Generación de visualizaciones**Resumen del procedimiento para este requerimiento:****1. Preparar el Entorno:**

- Instalar las librerías necesarias: matplotlib, seaborn, pandas, numpy.

2. Cargar los Datos:

- Leer los archivos CSV en DataFrames: clientes.csv, productos.csv, ventas.csv.

3. Crear Visualizaciones:

- Ventas Totales por Categoría de Producto: Agrupar ventas por categoría de producto y generar un gráfico de barras.
- Tendencia de Ventas Diarias: Agrupar ventas por fecha y generar un gráfico de líneas.
- Distribución de Ventas por Producto: Generar un histograma con línea para visualizar la distribución de ventas.

- Ventas por Cliente: Agrupar ventas por cliente y generar un gráfico de dispersión.
- Matriz de Correlación: Generar un mapa de calor para visualizar la correlación entre las variables de ventas.

Nombre del script: 5_3_generacion_visualizaciones_v2.py

Python

Paso 1: Preparar el Entorno

Instala las librerías necesarias si aún no las tienes:

pip install matplotlib seaborn pandas numpy

import pandas as pd # Importa la librería de manipulación de datos Pandas

import matplotlib.pyplot as plt # Importa la librería de visualización

Matplotlib

import seaborn as sns # Importa la librería de visualización Seaborn

Paso 2: Cargar los Datos

Leer los archivos CSV en DataFrames de pandas

Cargar los archivos CSV

clientes = pd.read_csv('clientes.csv') # Carga el archivo CSV

'clientes.csv' en un DataFrame

productos = pd.read_csv('productos.csv') # Carga el archivo CSV

'productos.csv' en un DataFrame

ventas = pd.read_csv('ventas.csv') # Carga el archivo CSV 'ventas.csv' en un DataFrame

Paso 3: Crear Visualizaciones

1. Gráfico de Barras: Ventas Totales por Categoría de Producto

Verificar que la columna 'Categoría' existe en productos

if 'Categoría' in productos.columns:

Calcular ventas totales por categoría de producto

```
ventas_productos = ventas.merge(productos, on='ProductoID') # Une las
tablas 'ventas' y 'productos' en 'ProductoID'
ventas_categoria =
ventas_productos.groupby('Categoria')['Cantidad'].sum().reset_index() #
Agrupar las ventas por 'Categoria' y suma las cantidades

# Crear gráfico de barras
plt.figure(figsize=(10, 6)) # Crea una nueva figura con un tamaño
específico
sns.barplot(x='Categoria', y='Cantidad', data=ventas_categoria) #
Genera un gráfico de barras con los datos de ventas por categoría
plt.title('Ventas Totales por Categoría de Producto') # Añade un
título al gráfico
plt.xlabel('Categoría') # Etiqueta el eje x como 'Categoría'
plt.ylabel('Cantidad Vendida') # Etiqueta el eje y como 'Cantidad
Vendida'
plt.xticks(rotation=45) # Rota las etiquetas del eje x para mejor
legibilidad
plt.show() # Muestra el gráfico

# Análisis
print("Este gráfico de barras muestra las ventas totales agrupadas por
la categoría de producto. Se puede observar cuál categoría tiene el mayor
volumen de ventas.")
else:
    print("La columna 'Categoria' no existe en el DataFrame 'productos'.")

# 2. Gráfico de Barras: Tendencia de Ventas Diarias

# Convertir FechaVenta a formato de fecha
ventas['FechaVenta'] = pd.to_datetime(ventas['FechaVenta']) # Convierte
la columna 'FechaVenta' a tipo datetime
```

```
# Agrupar ventas por fecha
ventas_diarias =
ventas.groupby('FechaVenta')['Cantidad'].sum().reset_index() # Agrupa las
ventas por fecha y suma las cantidades

# Graficar ventas diarias con gráfico de barras
plt.figure(figsize=(15, 6)) # Tamaño de la figura
plt.bar(ventas_diarias['FechaVenta'], ventas_diarias['Cantidad'],
width=1.0) # Genera un gráfico de barras para las ventas diarias
plt.title('Ventas Diarias') # Añade un título al gráfico
plt.xlabel('Fecha') # Etiqueta el eje x como 'Fecha'
plt.ylabel('Cantidad Vendida (Unidades)') # Etiqueta el eje y como
'Cantidad Vendida (Unidades)'
plt.xticks(rotation=45) # Rota las etiquetas del eje x para mayor
legibilidad
plt.tight_layout() # Ajusta el diseño para que todo encaje bien
plt.show() # Muestra el gráfico

# Análisis
print("Este gráfico de barras muestra la tendencia de ventas diarias. Se
pueden identificar patrones y fluctuaciones en las ventas a lo largo del
tiempo.")

# 3. Histograma: Distribución de Ventas por Producto

# Crear histograma
plt.figure(figsize=(10, 6)) # Crea una nueva figura con un tamaño
específico
sns.histplot(ventas['Cantidad'], bins=30, kde=True) # Genera un
histograma para la distribución de ventas por producto con una línea KDE
plt.title('Distribución de Ventas por Producto') # Añade un título al
gráfico
```

```
plt.xlabel('Cantidad Vendida') # Etiqueta el eje x como 'Cantidad
Vendida'
plt.ylabel('Frecuencia') # Etiqueta el eje y como 'Frecuencia'
plt.show() # Muestra el gráfico

# Análisis
print("Este histograma muestra la distribución de la cantidad de ventas
por producto. La línea KDE ayuda a visualizar la densidad de las ventas.")

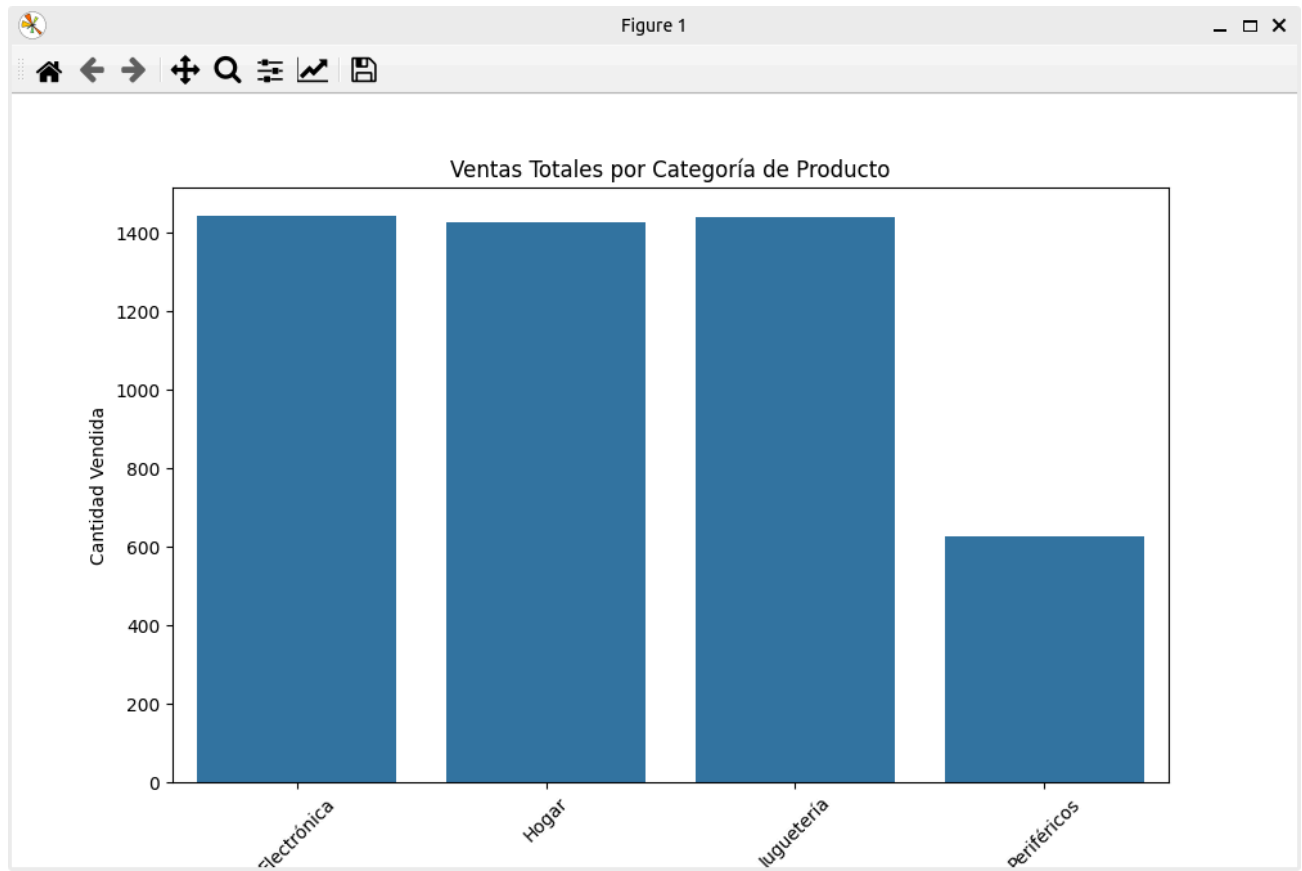
# 4. Diagrama de Dispersión: Ventas por Cliente

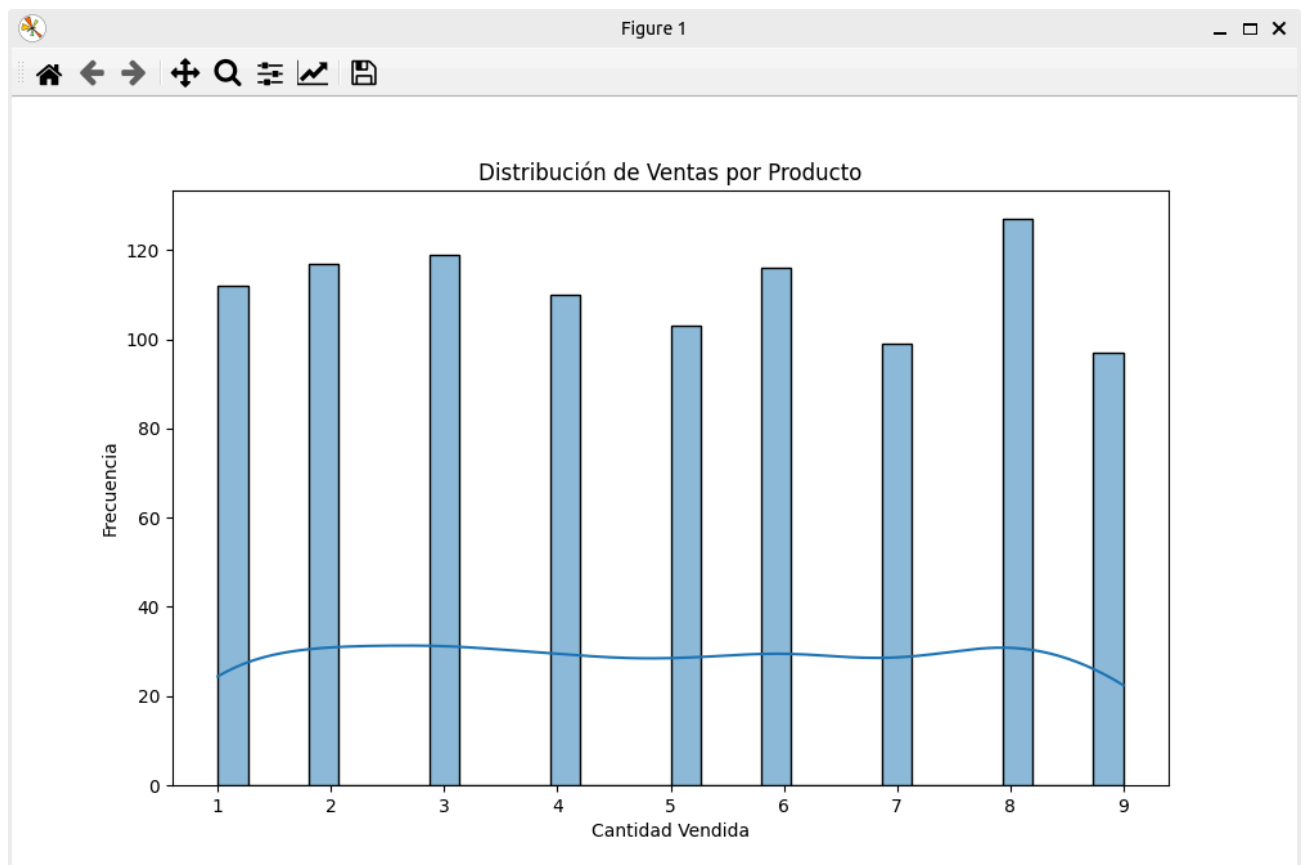
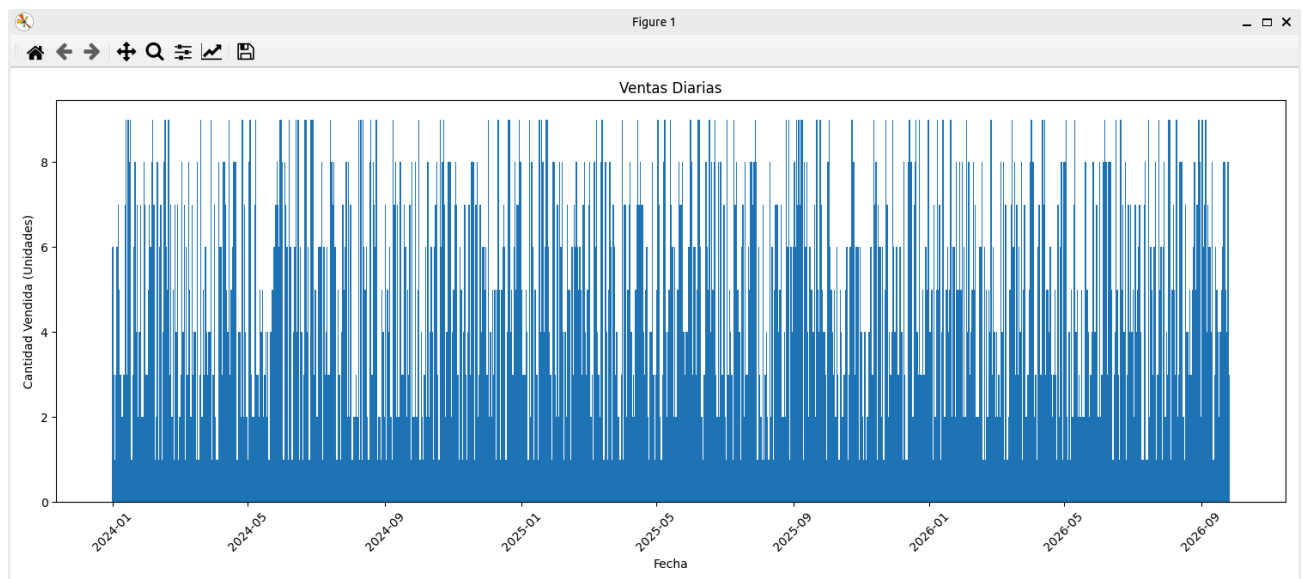
# Calcular ventas totales por cliente
ventas_cliente =
ventas.groupby('ClienteID')['Cantidad'].sum().reset_index() # Agrupa las
ventas por 'ClienteID' y suma las cantidades

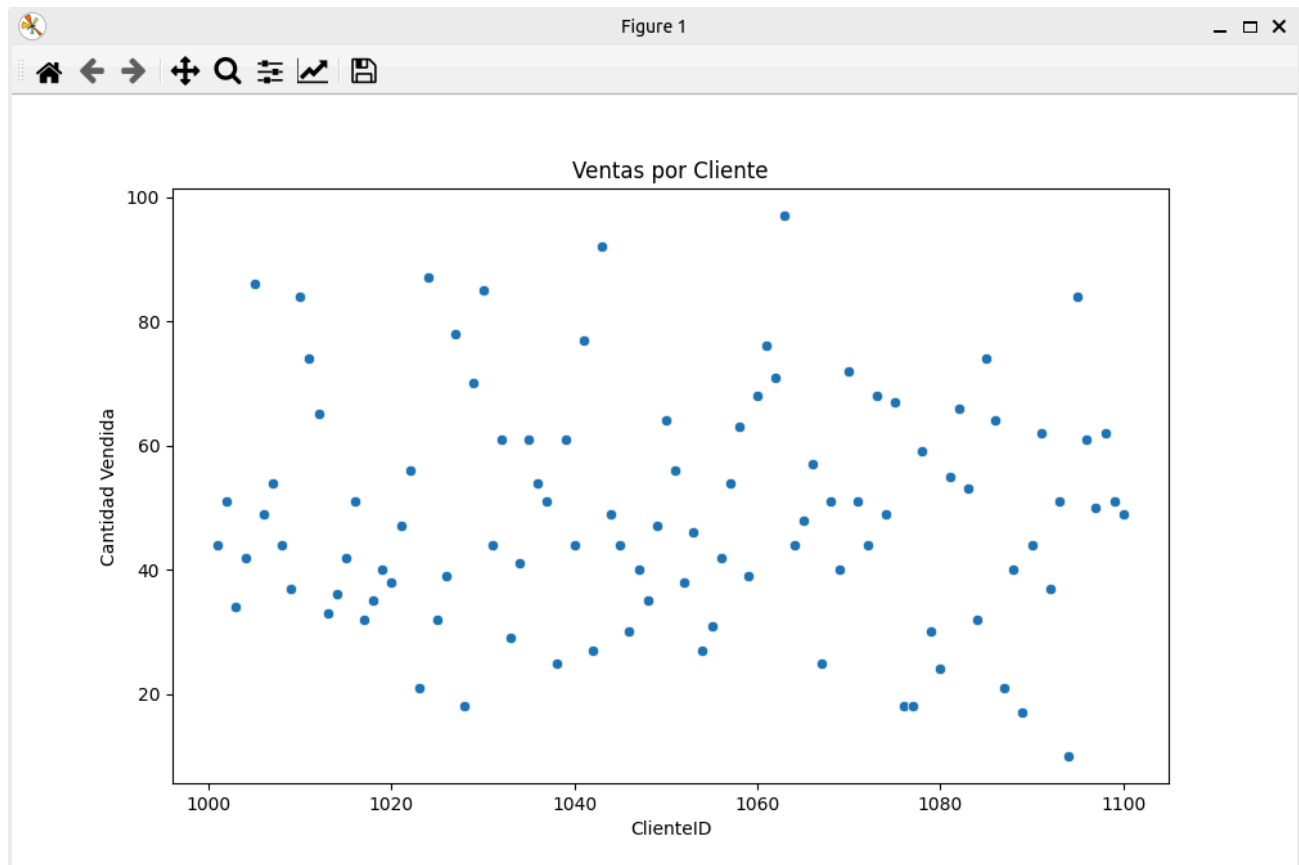
# Crear diagrama de dispersión
plt.figure(figsize=(10, 6)) # Crea una nueva figura con un tamaño
específico
sns.scatterplot(x='ClienteID', y='Cantidad', data=ventas_cliente) #
Genera un gráfico de dispersión para las ventas por cliente
plt.title('Ventas por Cliente') # Añade un título al gráfico
plt.xlabel('ClienteID') # Etiqueta el eje x como 'ClienteID'
plt.ylabel('Cantidad Vendida') # Etiqueta el eje y como 'Cantidad
Vendida'
plt.show() # Muestra el gráfico

# Análisis
print("Este diagrama de dispersión muestra las ventas totales por cliente.
Se pueden identificar clientes con compras altas y bajas.")
```

El resultado de este otro script es el siguiente:







Resumen de Resultados y Contribución al Negocio:

1. **Ventas por Categoría de Producto:**
 - **Resultados:** Identificación de categorías con mayores ventas.
 - **Contribución:** Mejora en la planificación de inventario y marketing.
2. **Tendencia de Ventas Diarias:**
 - **Resultados:** Identificación de patrones y fluctuaciones en ventas diarias.
 - **Contribución:** Mejora en la previsión de demanda y gestión de inventarios.
3. **Distribución de Ventas por Producto:**
 - **Resultados:** Variabilidad en ventas de productos.
 - **Contribución:** Optimización del portafolio de productos.
4. **Ventas por Cliente:**
 - **Resultados:** Identificación de clientes clave.
 - **Contribución:** Personalización de estrategias de marketing y fidelización.
5. **Matriz de Correlación:**
 - **Resultados:** Relación entre variables de ventas.
 - **Contribución:** Optimización de estrategias de marketing y promociones.

Este análisis integral apoya la toma de decisiones estratégicas, mejora la planificación y aumenta la eficiencia operativa.

6. CREACIÓN DE DASHBOARD

6.1 Desarrollo del Dashboard en Power BI

Resumen del Procedimiento:

1. **Importar Datos a Power BI:** Cargar los datos procesados para utilizarlos en el dashboard.
2. **Resumen de KPIs Clave:** Crear un panel de control con indicadores clave de rendimiento para una visión rápida del negocio.
3. **Gráficos de Tendencias de Ventas:** Visualizar las tendencias de ventas a lo largo del tiempo para identificar patrones.

Paso 1: Importar Datos a Power BI

En esta etapa, estamos cargando los datos procesados en Power BI para que podamos usarlos en nuestras visualizaciones.

1. Lo primero que se realiza es abrir Power BI Desktop para la creación de los Dashboard
2. Seleccionar en la ventana de inicio, "Obtener datos" para este caso se realiza directamente con la Base de datos de SQLServer.
3. Se establecen las credenciales definidas en esa base de datos, se buscan las tablas de clientes, productos y ventas cargados anteriormente.
4. Se finaliza con transformar datos en Power BI.

Paso 2: Desarrollar un Resumen de KPIs Clave

Aquí, estamos creando un panel de control con indicadores clave de rendimiento (KPIs) que proporcionan una visión rápida del rendimiento del negocio.

1. Se procede a realizar la creación de los KPIs según los requerimientos que se necesitan en el proyecto.
2. Ventas Totales: Usa una medida que sume la cantidad vendida.
3. Número de Transacciones: Cuenta el número total de ventas.
4. Clientes Únicos: Cuenta el número distinto de clientes.

Paso 3: Implementar Gráficos de Tendencias de Ventas

1. Se realizan gráficos de tendencias, en donde se mostrará las ventas a lo largo del tiempo.
2. Agrega el campo de fecha al eje x y la cantidad de ventas al eje y.
3. Segmentar los gráficos por diferentes criterios como producto o región usando filtros.
4. Visualizaciones de segmentación para permitir a los usuarios filtrar los datos por criterios diferentes.

Proyecto de Evaluación

Análisis de Datos para Optimización de Ventas en e-commerce

VENTA TOTAL

\$4,600,795

TRANSACCIONES

1000

CLIENTES UNIC...

100

MES

Todas

AÑO

Todas

REGIÓN

Este

Norte

Oeste

Sur

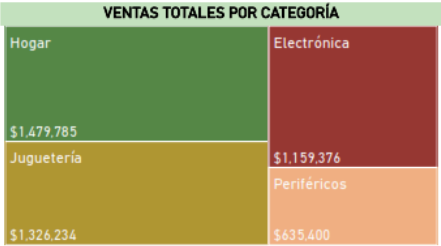
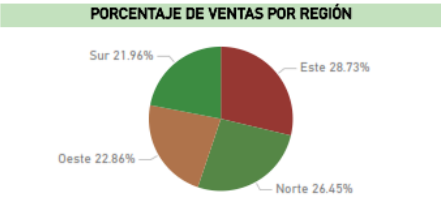
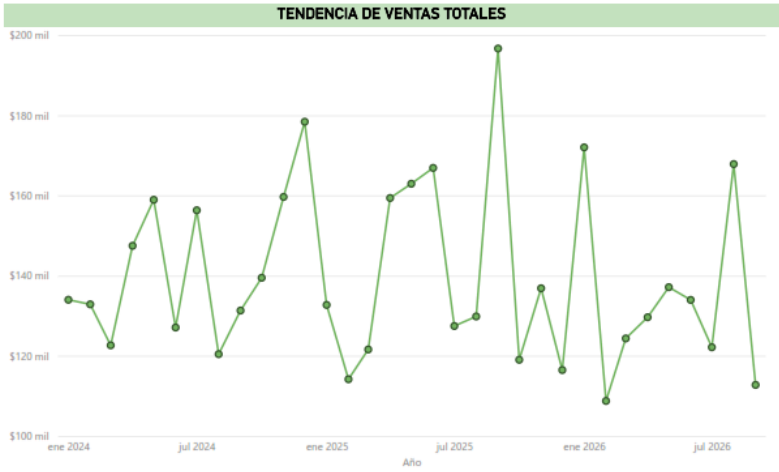
CATEGORÍA

Electrónica

Hogar

Juguetería

Periféricos



Proyecto de Evaluación

Análisis de Datos para Optimización de Ventas en e-commerce

VENTA TOTAL

\$4,600,795

TRANSACCIONES

1000

CLIENTES UNIC...

100

MES

Todas

AÑO

Todas

REGIÓN

Este

Norte

Oeste

Sur

CATEGORÍA

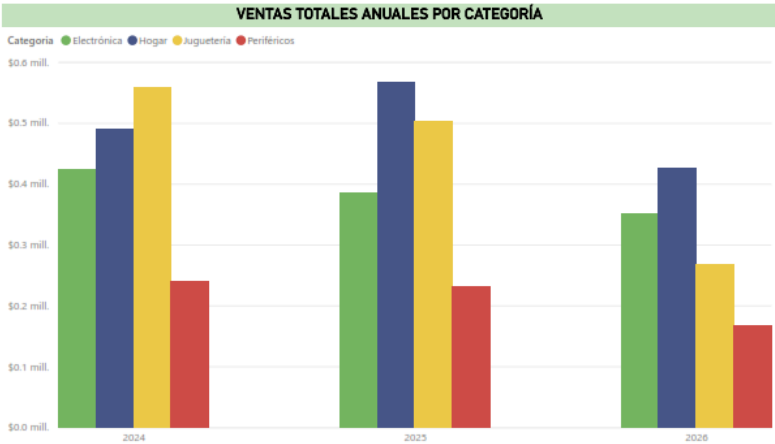
Electrónica

Hogar

Juguetería

Periféricos

Region	2024	2025	2026	Total
<input type="checkbox"/> Este	\$444,836	\$505,422	\$371,517	\$1,321,775
<input type="checkbox"/> Electrónica	\$96,791	\$155,029	\$136,638	\$388,458
<input type="checkbox"/> Hogar	\$182,239	\$157,428	\$98,931	\$438,598
<input type="checkbox"/> Juguetería	\$123,435	\$129,414	\$75,542	\$328,391
<input type="checkbox"/> Periféricos	\$42,371	\$63,551	\$60,406	\$166,328
<input type="checkbox"/> Norte	\$456,881	\$398,176	\$362,052	\$1,217,110
<input type="checkbox"/> Electrónica	\$93,299	\$89,662	\$94,811	\$277,772
<input type="checkbox"/> Hogar	\$96,772	\$144,248	\$141,455	\$382,475
<input type="checkbox"/> Juguetería	\$211,740	\$77,810	\$70,838	\$360,387
<input type="checkbox"/> Periféricos	\$55,071	\$86,457	\$54,947	\$196,475
<input type="checkbox"/> Oeste	\$386,270	\$440,233	\$225,274	\$1,051,777
<input type="checkbox"/> Electrónica	\$99,665	\$71,063	\$69,108	\$239,836
<input type="checkbox"/> Hogar	\$117,507	\$161,448	\$49,751	\$328,705
<input type="checkbox"/> Juguetería	\$119,660	\$164,090	\$86,099	\$369,849
<input type="checkbox"/> Periféricos	\$49,437	\$43,632	\$20,316	\$113,386
<input type="checkbox"/> Sur	\$420,475	\$340,023	\$249,636	\$1,010,134
<input type="checkbox"/> Electrónica	\$133,345	\$69,345	\$50,620	\$253,310
<input type="checkbox"/> Hogar	\$92,744	\$102,262	\$135,001	\$330,008
<input type="checkbox"/> Juguetería	\$102,039	\$131,628	\$33,939	\$267,606
<input type="checkbox"/> Periféricos	\$92,347	\$36,789	\$30,076	\$159,212
Total	\$1,708,462	\$1,683,854	\$1,208,479	\$4,600,795



Explicación: Estos gráficos de líneas nos ayudan a visualizar las tendencias de ventas a lo largo del tiempo, identificando patrones y estacionalidades.

Resumen de Resultados y Análisis de Insights

Descripción de los datos graficados.

- **KPIs Clave:** Proporcionan una visión rápida del rendimiento del negocio, permitiendo a los directivos tomar decisiones informadas basadas en los principales indicadores.
- **Gráficos de Tendencias:** Ayudan a identificar patrones y fluctuaciones en las ventas a lo largo del tiempo, facilitando la planificación de inventarios y estrategias de marketing.

Breve explicación de las visualizaciones.

1. **Gráfico de Líneas:** La tendencia de ventas revela patrones estacionales y fluctuaciones en las ventas a lo largo del tiempo.
2. **TreeMap:** Muestra una representación visual de las ventas totales por categoría la cual permite identificar patrones y valores atípicos según el tamaño y color de los rectángulos.
3. **Gráfico Circular:** Muestra el porcentaje de ingreso por ventas según la región.
4. **Gráfico de Barras:** Las ventas totales por categoría de producto agrupadas por año, muestran qué categorías tienen el mayor volumen de ventas y permiten una rápida comparación visual respecto a los años anteriores y posteriores.
5. **Matriz:** Muestra el detalle de las ventas por región y año permitiendo hacer drill down a categoría de producto y producto específico.

Insights Clave

1. **Identificación de Productos y Categorías Populares:** Conocer qué productos y categorías tienen mayores ventas permite enfocar esfuerzos en mantener un inventario óptimo y dirigir las promociones hacia esos productos.
2. **Análisis de Tendencias:** Identificar patrones estacionales y tendencias en las ventas diarias facilita una mejor planificación y gestión del inventario, así como el diseño de campañas de marketing más efectivas.
3. **Segmentación y Fidelización de Clientes:** Entender el comportamiento de compra de los clientes ayuda a segmentarlos eficazmente, permitiendo personalizar estrategias de retención y aumentar la fidelización.
4. **Optimización de Precios y Promociones:** Analizar la distribución de ventas por producto y la correlación entre variables puede influir en las decisiones de precios y estrategias promocionales, mejorando la eficiencia y efectividad.

Estos insights proporcionan una base sólida para tomar decisiones estratégicas informadas, optimizar operaciones y mejorar la satisfacción del cliente, impulsando así el rendimiento general del negocio.