Informe Técnico

Análisis de Datos de Ventas utilizando MongoDB y Python

1. Introducción

En este proyecto, se desarrolló una solución práctica para analizar datos de ventas utilizando MongoDB y Python. Se seleccionó la temática de "Análisis de Ventas" debido a su relevancia en la toma de decisiones empresariales.

El objetivo principal fue diseñar una base de datos, poblarla con información realista y desarrollar consultas avanzadas mediante pipelines de agregación, permitiendo obtener información valiosa como ingresos por producto, ventas por categoría, y el producto más vendido.

2. Diseño de la Base de Datos

La base de datos consta de dos colecciones principales:

- **Productos**: Contiene información sobre los productos en venta, como su nombre, categoría y precio.
- **Ventas**: Registra las transacciones realizadas, asociando cada venta con un producto mediante su producto_id.

Estructura de las colecciones:

Colección productos**:**

Campo	**Tipo**	**Descripción**
_id	Integer	Identificador único del producto.
nombre	String	Nombre del producto.
categoria	String	Categoría a la que pertenece el producto.
precio	Decimal	Precio del producto.

Colección ventas**:**

Campo	**Tipo**	**Descripción**
_id	Integer	Identificador único de la venta.
producto_id	Integer	Relación con el _id de la colección productos.
cantidad	Integer	Cantidad de productos vendidos.

Campo	Tipo	Descripción
fecha	Date	Fecha de la transacción.

Esta estructura garantiza la integridad y la facilidad de realizar operaciones complejas mediante $lookup.

3. Desarrollo del Trabajo

Conexión a MongoDB

Se utilizó la librería pymongo para conectar Python con MongoDB. Se crearon funciones para poblar la base de datos con datos realistas y ejecutar consultas con pipelines de agregación.

Consultas Realizadas

1. **Consulta 1: Total de ingresos por producto**
    - **Propósito**: Calcular el ingreso generado por cada producto, multiplicando su precio por la cantidad vendida.
    - **Pipeline**:

```
pipeline = [
  {"$lookup": {"from": "productos", "localField": "producto_id", "foreignField": "_id", "as": "producto"}},
  {"$unwind": "$producto"},
  {"$group": {"_id": "$producto.nombre", "total_ingresos": {"$sum": {"$multiply": ["$cantidad", "$producto.precio"]}}}},
  {"$sort": {"total_ingresos": -1}}
]
```

    - **Resultado**:
        - Laptop: $2,400
        - Auriculares: $250
        - Zapatillas: $240
2. **Consulta 2: Ventas totales por categoría**
    - **Propósito**: Determinar el número total de productos vendidos por categoría.

```
pipeline = [
  {"$lookup": {"from": "productos", "localField": "producto_id", "foreignField": "_id", "as": "producto"}},
  {"$unwind": "$producto"},
  {"$group": {"_id": "$producto.categoria", "total_ventas": {"$sum": "$cantidad"}}},
  {"$sort": {"total_ventas": -1}}
]
```

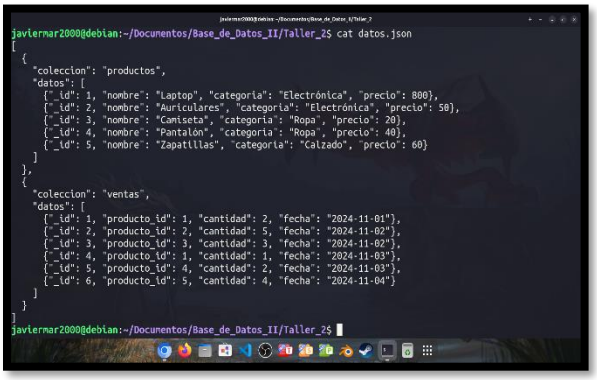
- **Resultado**:
    - Electrónica: 8
    - Ropa: 5
    - Calzado: 4

3. **Consulta 3: Producto más vendido**
    - **Propósito**: Identificar el producto con mayor cantidad de ventas.
    - **Pipeline**:

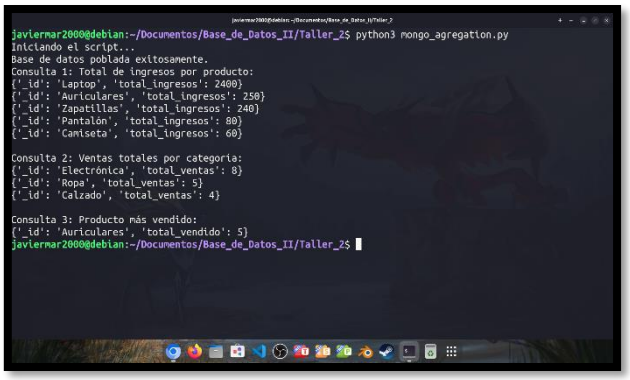
```
pipeline = [
  {"$lookup": {"from": "productos", "localField": "producto_id", "foreignField": "_id", "as": "producto"}},
  {"$unwind": "$producto"},
  {"$group": {"_id": "$producto.nombre", "total_vendido": {"$sum": "$cantidad"}}},
  {"$sort": {"total_vendido": -1}},
  {"$limit": 1}
]
```

- **Resultado**: Auriculares (5 unidades).

Capturas de Pantalla:

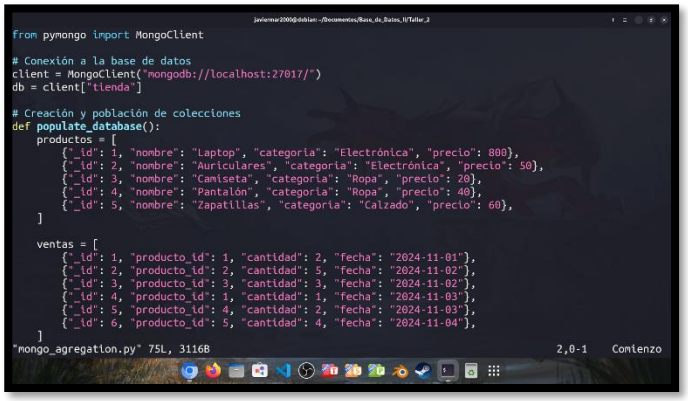


Captura 1: "Estructura de los datos iniciales en formato JSON, organizados en las colecciones productos y ventas, listos para ser importados a MongoDB."



Captura 2: "Resultados obtenidos al ejecutar el script mongo_agregation.py, mostrando el total de ingresos por producto, ventas por categoría y el producto más vendido."

Captura 3: "Fragmento del script mongo_agregation.py que conecta Python con MongoDB y pobla la base de datos con las colecciones productos y ventas."

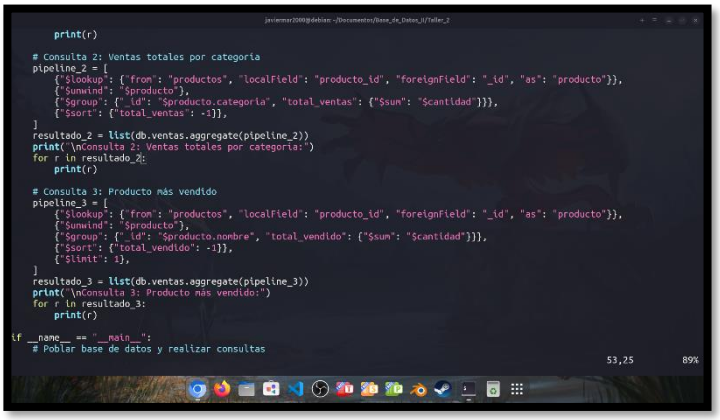


```
from pymongo import MongoClient

# Conexión a la base de datos
client = MongoClient("mongodb://localhost:27017/")
db = client["tienda"]

# Creación y población de colecciones
def populate_database():
    productos = [
        {"_id": 1, "nombre": "Laptop", "categoria": "Electrónica", "precio": 800},
        {"_id": 2, "nombre": "Auriculares", "categoria": "Electrónica", "precio": 50},
        {"_id": 3, "nombre": "Camiseta", "categoria": "Ropa", "precio": 20},
        {"_id": 4, "nombre": "Pantalón", "categoria": "Ropa", "precio": 40},
        {"_id": 5, "nombre": "Zapatillas", "categoria": "Calzado", "precio": 60},
    ]

    ventas = [
        {"_id": 1, "producto_id": 1, "cantidad": 2, "fecha": "2024-11-01"},
        {"_id": 2, "producto_id": 2, "cantidad": 5, "fecha": "2024-11-02"},
        {"_id": 3, "producto_id": 3, "cantidad": 3, "fecha": "2024-11-02"},
        {"_id": 4, "producto_id": 1, "cantidad": 1, "fecha": "2024-11-03"},
        {"_id": 5, "producto_id": 4, "cantidad": 2, "fecha": "2024-11-03"},
        {"_id": 6, "producto_id": 5, "cantidad": 4, "fecha": "2024-11-04"},
    ]
"mongo_agregation.py" 75L, 3116B                                2,0-1     Comienzo
```



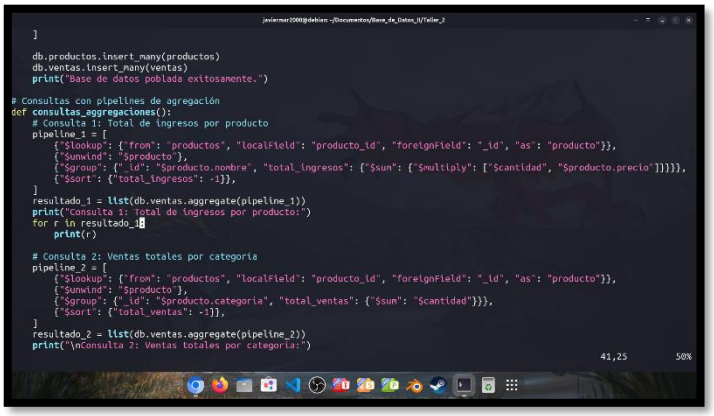
```
    print(r)

# Consulta 2: Ventas totales por categoría
pipeline_2 = [
    {"$lookup": {"from": "productos", "localField": "producto_id", "foreignField": "_id", "as": "producto"}},
    {"$unwind": "$producto"},
    {"$group": {"_id": "$producto.categoria", "total_ventas": {"$sum": "$cantidad"}}},
    {"$sort": {"total_ventas": -1}},
]
resultado_2 = list(db.ventas.aggregate(pipeline_2))
print("\nConsulta 2: Ventas totales por categoría:")
for r in resultado_2:
    print(r)

# Consulta 3: Producto más vendido
pipeline_3 = [
    {"$lookup": {"from": "productos", "localField": "producto_id", "foreignField": "_id", "as": "producto"}},
    {"$unwind": "$producto"},
    {"$group": {"_id": "$producto.nombre", "total_vendido": {"$sum": "$cantidad"}}},
    {"$sort": {"total_vendido": -1}},
    {"$limit": 1},
]
resultado_3 = list(db.ventas.aggregate(pipeline_3))
print("\nConsulta 3: Producto más vendido:")
for r in resultado_3:
    print(r)

if __name__ == "__main__":
    # Poblar base de datos y realizar consultas
                                                    53,25         89%
```

Captura 4: "Definición del pipeline para calcular el total de ingresos por producto, utilizando operadores avanzados como $lookup, $unwind, $group y $sort para realizar operaciones de agregación sobre las colecciones productos y ventas."



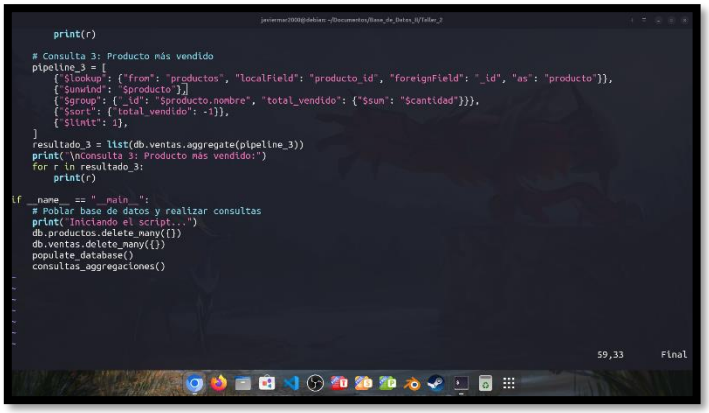
```
]

db.productos.insert_many(productos)
db.ventas.insert_many(ventas)
print("Base de datos poblada exitosamente.")

# Consultas con pipelines de agregación
def consultas_aggregaciones():
    # Consulta 1: Total de ingresos por producto
    pipeline_1 = [
        {"$lookup": {"from": "productos", "localField": "producto_id", "foreignField": "_id", "as": "producto"}},
        {"$unwind": "$producto"},
        {"$group": {"_id": "$producto.nombre", "total_ingresos": {"$sum": {"$multiply": ["$cantidad", "$producto.precio"]}}}},
        {"$sort": {"total_ingresos": -1}},
    ]
    resultado_1 = list(db.ventas.aggregate(pipeline_1))
    print("Consulta 1: Total de ingresos por producto:")
    for r in resultado_1:
        print(r)

    # Consulta 2: Ventas totales por categoría
    pipeline_2 = [
        {"$lookup": {"from": "productos", "localField": "producto_id", "foreignField": "_id", "as": "producto"}},
        {"$unwind": "$producto"},
        {"$group": {"_id": "$producto.categoria", "total_ventas": {"$sum": "$cantidad"}}},
        {"$sort": {"total_ventas": -1}},
    ]
    resultado_2 = list(db.ventas.aggregate(pipeline_2))
    print("\nConsulta 2: Ventas totales por categoría:")
                                                    41,25         58%
```

Captura 5: "Definición del pipeline para calcular las ventas totales por categoría, aprovechando operadores como $lookup, $unwind, $group y $sort para agrupar los datos por categorías y sumar las cantidades vendidas."



```
    print(r)

# Consulta 3: Producto más vendido
pipeline_3 = [
    {"$lookup": {"from": "productos", "localField": "producto_id", "foreignField": "_id", "as": "producto"}},
    {"$unwind": "$producto"},
    {"$group": {"_id": "$producto.nombre", "total_vendido": {"$sum": "$cantidad"}}},
    {"$sort": {"total_vendido": -1}},
    {"$limit": 1},
]
resultado_3 = list(db.ventas.aggregate(pipeline_3))
print("\nConsulta 3: Producto más vendido:")
for r in resultado_3:
    print(r)

if __name__ == "__main__":
    # Poblar base de datos y realizar consultas
    print("Iniciando el script...")
    db.productos.delete_many({})
    db.ventas.delete_many({})
    populate_database()
    consultas_aggregaciones()
~
~
~
~
~
~
~
                                                    59,33         Final
```

Captura 6: "Definición del pipeline para identificar el producto más vendido, utilizando operadores como $lookup, $unwind, $group, $sort y $limit para encontrar el producto con mayor cantidad de ventas."

4. Resultados y Análisis

Los resultados obtenidos a partir de las consultas muestran tendencias importantes:

- El producto más vendido son los auriculares, posiblemente debido a su precio accesible.
- La categoría "Electrónica" genera más ingresos y tiene mayor número de ventas.
- Productos con precios elevados como la Laptop contribuyen significativamente a los ingresos totales.

Este análisis podría ser utilizado para optimizar el inventario, priorizar promociones y maximizar las ganancias.

5. Conclusiones

Este proyecto permitió explorar la integración de Python con MongoDB y desarrollar habilidades en el uso de pipelines de agregación. Los principales aprendizajes incluyen:

- La eficacia de $lookup para unir colecciones y enriquecer datos.
- La importancia de las operaciones de $group y $project para realizar cálculos avanzados.
- Cómo un diseño adecuado de la base de datos facilita el análisis de datos.

Desafíos superados:

- Configuración inicial de MongoDB.
- Ajuste de pipelines para obtener resultados precisos.

Futuras mejoras:

- Incorporar visualizaciones interactivas para facilitar la interpretación de los datos.
- Ampliar las consultas para incluir análisis de tendencias temporales.

6. Anexos

Código Python Completo:

mongo_agregation.py

Archivo JSON:

datos.json