Postgrado en Análisis y Predicción de Datos

Ciencia de Datos en Python

Ing. Preng Biba Solares

Proyecto Final

Saul Alexander López Contreras
Carné: 24000138

Josias Emanuel Chocooj Azmitia
Carné: 24000705

Erick Manuel Andrade Zuñiga
Carné: 24000165

Sección: U

Ciudad de Guatemala, 12 de abril 2024

Índice

1. Introducción 2

2. Objetivos 2

   2.1. Objetivo general 2
   2.2. Objetivos específicos 2

3. Contenido 3

   3.1. Infraestructura de AWS 3
   3.2. Base de Datos Transaccional 3
   3.3. Data Warehouse 4
   3.4. Proceso ETL 5
   3.5. Visualización y Análisis 6
   3.6. Código y Tecnología Utilizadas 6
   3.7. Cumplimiento de Requerimientos 7
   3.8. Preguntas de Negocio 8

4. Conclusiones 8

5. Anexos 9

1. Introducción

Como proyecto final del curso Ciencia de Datos en Python, se desarrolla el presente sistema de ingeniería de datos. Este consta de crear una infraestructura de base de datos transaccional, cargar datos de prueba y emplear una estructura de Datawarehouse para realizar análisis de datos del mismo y responder preguntas de negocio. Se utiliza Python en cada uno de estos pasos como principal sistema de programación. ema
Además de Python, se emplea el sistema de cloud computing de AWS para desarrollar la base de datos y SQL para generar la estructura del DataWarehouse y el proceso de ETL para carga de datos, respectivamente. Para la exploración y análisis de datos se emplean paquetes de Numpy, Pandas, Matplotlib y Seaborn, dentro del entorno de Python.

El Proyecto cumple con los requerimientos de uso de AWS para generación de infraestructura, separación de sistema transaccional y de Datawarehouse, procesamiento en EC2 a través de Python, resolución de preguntas de negocio. No se utiliza SQL para generación de ninguna estructura.

2. Objetivos

2.1. Objetivos Generales

- Desarrollar un sistema de ingeniería de datos que integre Python, SQL, y AWS para automatizar el flujo de datos desde la ingestión hasta el análisis y la visualización.

2.2. Objetivos Específicos

- Implementar un sistema transaccional automatizado en AWS.
- Crear y poblar una base de datos utilizando Python.
- Diseñar y construir un data warehouse que responda a preguntas de negocio específicas.
- Desarrollar un proceso ETL para integrar, transformar y cargar datos.
- Realizar análisis de datos y visualización para responder a preguntas de negocio.

3. Contenido

3.1. Infraestructura de AWS

Para lograr el objetivo del proyecto, se utilizó la infraestructura de capa gratuita de AWS, utilizando RDS para el uso de MySQL y Postgres, EC2, VPC y la respectiva configuración para únicamente utilizar los puertos asignados con credenciales previamente configuradas para acceder a recursos predefinidos

```
aws_rds_conn = boto3.client('rds',
                            aws_access_key_id=config.get('IAM', 'ACCESS_KEY'),
                            aws_secret_access_key=config.get('IAM', 'SECRET_ACCESS_KEY'),
                            region_name=config.get('IAM', 'REGION'))

aws_ec2_conn = boto3.client('ec2',
                            aws_access_key_id=config.get('IAM', 'ACCESS_KEY'),
                            aws_secret_access_key=config.get('IAM', 'SECRET_ACCESS_KEY'),
                            region_name=config.get('IAM', 'REGION'))
```

3.2. Base de Datos Transaccional

La base de datos transaccional fue diseñada para gestionar las operaciones diarias y facilitar la recolección de datos en tiempo real. Se utilizó un gestor de base de datos SQL disponible en Amazon RDS, lo que permitió un manejo eficiente y seguro de las transacciones. La estructura de esta base se definió para optimizar la rapidez y la eficiencia en las operaciones de inserción y actualización de datos, cruciales para el manejo transaccional.

- Creación de la Base de Datos: Utilizando el lenguaje SQL, se definieron las tablas necesarias con su correspondiente estructura (`database.sql`), incluyendo claves primarias, claves foráneas, índices, y restricciones para asegurar la integridad y el rendimiento de la base de datos.

```
# Crear la base de datos MySQL de transacciones
try:
    response = aws_rds_conn.create_db_instance(
        DBInstanceIdentifier=config.get('TRANSACC', 'DB_INSTANCE_ID'),
        DBName=config.get('TRANSACC', 'DB_NAME'),
        DBInstanceClass='db.t2.micro',
        Engine='mysql',
        MasterUsername=config.get('TRANSACC', 'DB_USER'),
        MasterUserPassword=config.get('TRANSACC', 'DB_PASSWORD'),
        Port=int(config.get('TRANSACC', 'DB_PORT')),
        PubliclyAccessible=True,
        VpcSecurityGroupIds=[config.get('VPC', 'SECURITY_GROUP')],
        AllocatedStorage=20
    )
    print(response)
except aws_rds_conn.exceptions.DBInstanceAlreadyExistsFault as ex:
    print("La instancia de base de datos MySQL ya existe.")
except Exception as ex:
    print("Error al crear la base de datos MySQL:", ex)
```

- Automatización de la Ingesta de Datos: Se desarrollaron scripts en Python que automatizan la inserción de datos a la base de datos transaccional. Estos scripts utilizan bibliotecas como `psycopg2` o `sqlalchemy` para establecer una conexión segura y eficiente con la base de datos, y ejecutar operaciones de inserción de datos masivos de forma regular.

```python
from sqlalchemy import create_engine, Column, Integer, String, DECIMAL, DateTime, ForeignKey
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, relationship
from faker import Faker
import random

db_url = 'mysql+pymysql://root:password@localhost/transactional_db'
engine = create_engine(db_url)
Session = sessionmaker(bind=engine)
session = Session()
Base = declarative_base()

fake = Faker()
```

3.3. Data Warehouse

El data warehouse se estructuró para apoyar el análisis de datos y la toma de decisiones, almacenando datos históricos estructurados de forma que faciliten la ejecución de consultas complejas y análisis multidimensionales.

- Estructura del Data Warehouse: Se diseñó una esquema en estrella o en copo de nieve, según las necesidades analíticas del proyecto. Este diseño implicó la creación de una tabla de hechos central, que almacena métricas clave y claves foráneas que se relacionan con dimensiones específicas.

```sql
CREATE TABLE IF NOT EXISTS categoria(
    idcategoria INT PRIMARY KEY,
    nombre VARCHAR(50) UNIQUE,
    descripcion VARCHAR(255),
    estado BIT
);

CREATE TABLE IF NOT EXISTS rol(
    idrol INT PRIMARY KEY,
    nombre VARCHAR(30) UNIQUE,
    descripcion VARCHAR(255),
    estado BIT
);
```

- Definición del Esquema: Utilizando Python para definir el código DDL (`estructura_dwh.py`), se crearon las tablas de hechos y dimensiones, asegurando que soporten los tipos de análisis requeridos por las preguntas de negocio.

```python
from sqlalchemy import create_engine, Column, Integer, String, Numeric, DateTime, ForeignKey
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, relationship

db_url = 'postgresql://usuario:contraseña@localhost/dwh_db'
engine = create_engine(db_url)
Session = sessionmaker(bind=engine)
session = Session()
Base = declarative_base()

# Tabla de dimensión para Categoría (SCD Tipo 1)
class DimCategoria(Base):
    __tablename__ = 'dim_categoria'
    idcategoria = Column(Integer, primary_key=True)
    nombre = Column(String(50))
    descripcion = Column(String(255))
    estado = Column(Integer)
    fecha_inicio = Column(DateTime)
    fecha_fin = Column(DateTime)

    def __repr__(self):
        return f"<DimCategoria(idcategoria={self.idcategoria}, nombre='{self.nombre}')>"
```

3.4. Proceso ETL

El proceso ETL (Extract, Transform, Load) se implementó para extraer datos del sistema transaccional, transformarlos según los requerimientos del data warehouse y cargarlos en la estructura definida del data warehouse.

- Extracción: Los datos se extrajeron de la base de datos transaccional utilizando consultas SQL optimizadas para minimizar el impacto en el rendimiento operativo.
- Transformación: En esta etapa, se aplicaron técnicas de limpieza, normalización, y agregación a los datos, utilizando Python para transformar los datos en un formato adecuado para el análisis. Esto incluyó la conversión de tipos de datos, el manejo de valores faltantes, la derivación de nuevas columnas, y la agrupación de datos.
- Carga: Finalmente, los datos transformados se cargaron en el data warehouse. Este proceso se automatiza para ejecutarse en intervalos regulares, asegurando que el data warehouse estuviera actualizado con los datos más recientes para análisis.

```python
# Extracción, transformación y carga de la dimensión Categoría
def etl_dim_categoria():
    # Extracción
    df_categoria = pd.read_sql('SELECT * FROM categoria', transactional_engine)

    # Transformación
    df_categoria_dim = df_categoria.rename(columns={'idcategoria': 'idcategoria', 'nombre': 'nombre',
'descripcion': 'descripcion', 'estado': 'estado'})
    df_categoria_dim['fecha_inicio'] = pd.to_datetime('today')
    df_categoria_dim['fecha_fin'] = pd.to_datetime('2999-12-31')

    # Carga
    df_categoria_dim.to_sql('dim_categoria', dwh_engine, if_exists='append', index=False)
```

3.5. Visualización y Análisis

El análisis de datos se llevó a cabo utilizando herramientas como Pandas, NumPy, Matplotlib, y Seaborn en Python, para responder a las preguntas de negocio planteadas.

- Preparación de Datos: Utilizando Pandas, se prepararon los conjuntos de datos para el análisis, realizando operaciones como selección de columnas, filtrado de filas, y manejo de datos faltantes.
- Análisis Exploratorio de Datos (EDA): Se realizaron análisis exploratorios para entender las tendencias, patrones y anomalías en los datos. Esto incluyó la generación de estadísticas descriptivas, la visualización de distribuciones de datos y la identificación de relaciones entre variables.
- Análisis Profundo: Se realizaron análisis más profundos para responder a preguntas específicas de negocio, aplicando técnicas de análisis estadístico, modelado predictivo, o minería de datos según fuera necesario.
- Visualización de Datos: Se crearon visualizaciones interactivas y estáticas para representar los resultados de los análisis, facilitando la interpretación y la toma de decisiones basada en datos.

3.6. Código y Tecnologías Utilizadas

El proyecto se apoyó en una variedad de tecnologías y lenguajes de programación para construir un sistema de ingeniería de datos completo y robusto.

- Python: Se utilizó como el lenguaje principal para el desarrollo del proyecto debido a su flexibilidad y las amplias bibliotecas

disponibles para el manejo de datos, tales como Pandas para el procesamiento de datos y NumPy para operaciones numéricas.

```
import boto3
import pandas as pd
import numpy as np
import pymysql
import psycopg2
import configparser
from sqlalchemy import create_engine
```

- SQL: Empleado para la consulta de datos en las bases de datos transaccionales y el data warehouse alojados en Amazon RDS. Se utilizó principalmente para extraer datos que luego serían procesados y analizados en Python.
- Amazon Web Services (AWS): Se usaron varios servicios de AWS, incluyendo:
    - Amazon RDS: Para alojar las bases de datos transaccionales y de data warehouse.
    - Amazon EC2: Para ejecutar las aplicaciones de procesamiento y análisis de datos.
- Bibliotecas de Visualización: Matplotlib y Seaborn se emplearon para crear visualizaciones de datos que ayudaron a interpretar los análisis y presentar los hallazgos de manera efectiva.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

3.7. Cumplimiento de Requerimientos

El proyecto cumplió con los siguientes requisitos técnicos y funcionales:

- Uso de AWS para Infraestructura: Se configuraron y utilizaron servicios de AWS, cumpliendo con el requisito de implementar una infraestructura escalable y confiable. Amazon RDS se utilizó para alojar las bases de datos, mientras que EC2 proporciona una plataforma para el procesamiento y análisis de datos.

- Separación de Sistemas Transaccionales y Data Warehouse: Cumpliendo con las mejores prácticas de diseño de sistemas de información, se emplearon distintas bases de datos en RDS para el sistema transaccional y el data warehouse, asegurando la separación de las operaciones y análisis.
- Procesamiento de Datos en Python: Se cumplió con la restricción de no utilizar SQL para la construcción de estructuras o transformaciones de datos complejas. Python se utilizó extensivamente para el procesamiento de datos, desde la extracción y transformación en el proceso ETL hasta el análisis y visualización final.
- Análisis y Visualización de Datos: Se utilizaron Python y sus bibliotecas para el análisis y visualización de datos, logrando un enfoque profundo y consistente en el análisis. Esto permitió responder a las preguntas de negocio planteadas al principio del proyecto.

Este enfoque aseguró que todas las fases del proyecto, desde la ingestión de datos hasta el análisis y visualización, se realizarán de manera coherente, eficiente y alineada con los objetivos del proyecto.

3.8. **Preguntas de negocio**

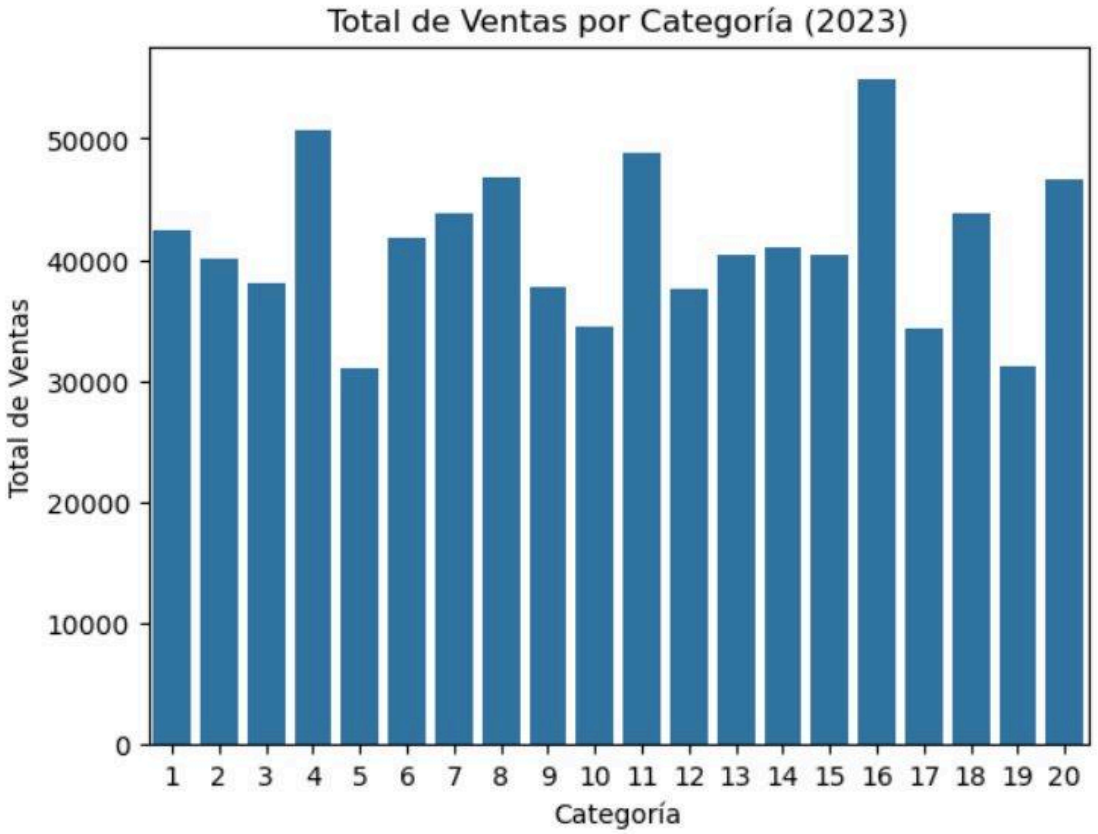
- ¿Cuál es el total de ventas por categoría de artículo en un período específico?
- ¿Quiénes son los clientes que generan el mayor volumen de ventas y cuál es su contribución al total de ingresos?
- ¿Cuál es el rendimiento de ventas de cada vendedor y cómo se compara con el promedio de ventas por vendedor?
- ¿Cuáles son los artículos más vendidos y menos vendidos en términos de cantidad y monto total de ventas?
- ¿Cómo ha evolucionado el volumen de ventas a lo largo del tiempo (por mes, trimestre, año)?

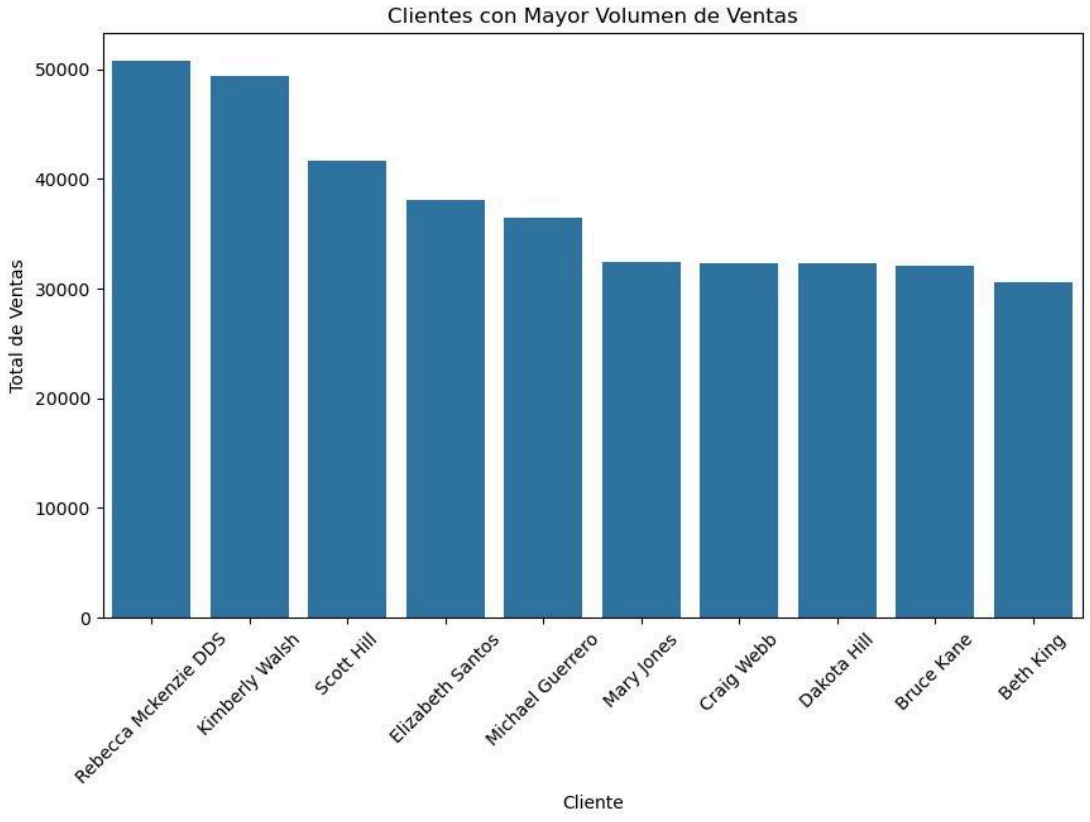
4. Conclusiones

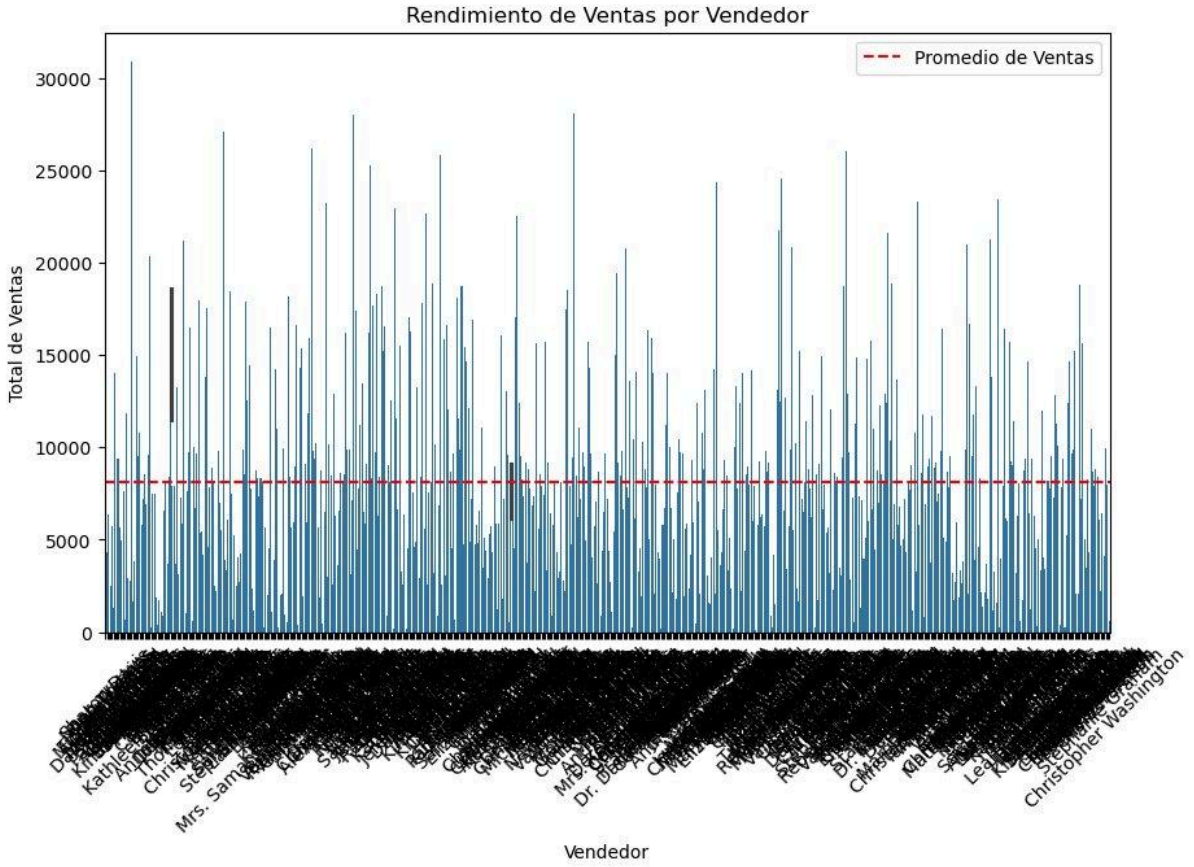
4.1. Cumplimiento de Objetivos: Resumir cómo el proyecto cumplió los objetivos y requisitos planteados.

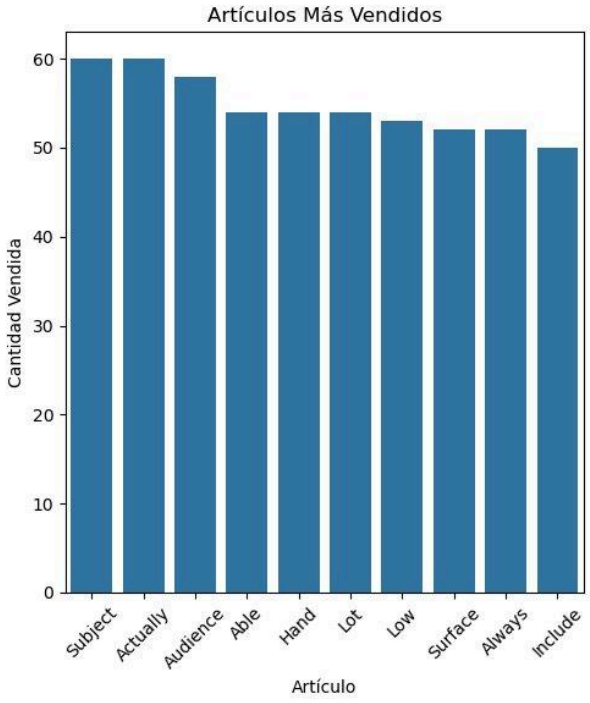
4.2. El correcto desarrollo del proyecto responde a una secuencia lógica adecuada en la generación de cada una de sus partes, desde la infraestructura hasta el análisis de información.

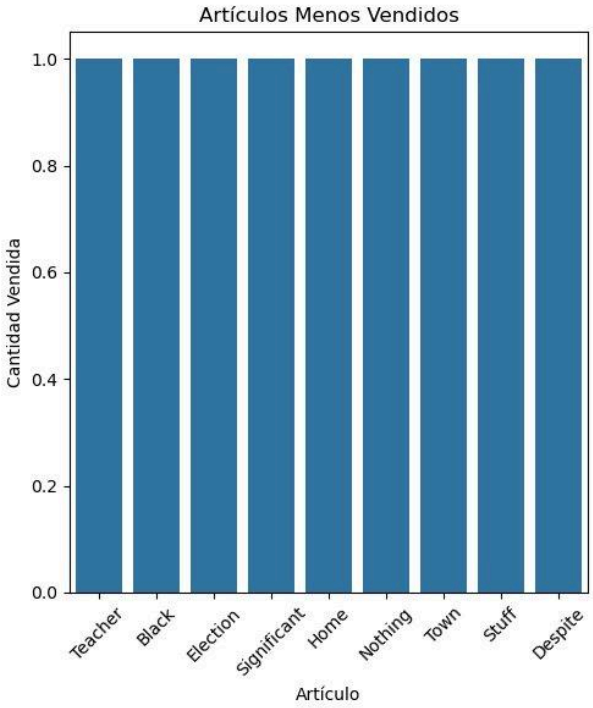
5. Anexos

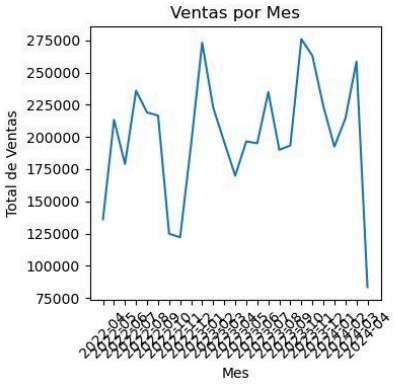

Total de Ventas por Categoría (2023)
Total de Ventas
0
10000
20000
30000
40000
50000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Categoría

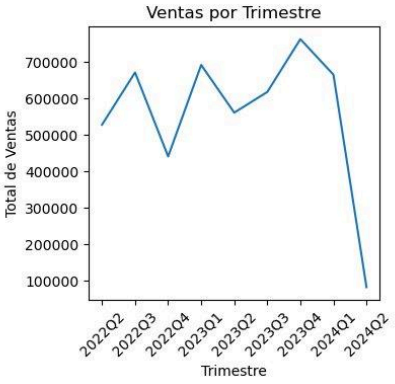

Clientes con Mayor Volumen de Ventas
Total de Ventas
0
10000
20000
30000
40000
50000
Rebecca Mckenzie DDS
Kimberly Walsh
Scott Hill
Elizabeth Santos
Michael Guerrero
Mary Jones
Craig Webb
Dakota Hill
Bruce Kane
Beth King
Cliente

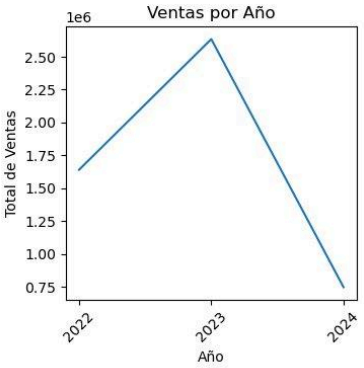

Rendimiento de Ventas por Vendedor
Promedio de Ventas
Total de Ventas
30000
25000
20000
15000
10000
5000
0
Vendedor


Artículos Más Vendidos
Cantidad Vendida
60
50
40
30
20
10
0
Subject
Actually
Audience
Able
Hand
Lot
Low
Surface
Always
Include
Artículo


Artículos Menos Vendidos
Cantidad Vendida
1.0
0.8
0.6
0.4
0.2
0.0
Teacher
Black
Election
Significant
Home
Nothing
Town
Stuff
Despite
Artículo


Ventas por Mes
Total de Ventas
Mes


Ventas por Trimestre
Total de Ventas
2022Q2
2022Q3
2022Q4
2023Q1
2023Q2
2023Q3
2023Q4
2024Q1
2024Q2
Trimestre


Ventas por Año
1e6
Total de Ventas
2022
2023
2024
Año