Universidad Autónoma de Baja California

Facultad de Ciencias

Programación para Ciencia de Datos

Docente: Dr. Raul Casillas Figueroa

Alumno: Mariana Salazar Romero

Matrícula: 376123

Proyecto Final - Análisis de datos en tienda de conveniencia: patrones de compra y estrategias de Marketing.

03/junio/2025

ÍNDICE GENERAL

PÁGINA

LISTA DE FIGURAS

BIBLIOGRAFÍA Y REFERENCIAS

Python Software Foundation. (2024). Python Documentation.
https://docs.python.org/

McKinney, W. (2022). pandas: powerful Python data analysis toolkit.
https://pandas.pydata.org/docs/

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment.
Computing in Science & Engineering, 9(3), 90-95.

Waskom, M. (2021). seaborn: statistical data visualization.
Journal of Open Source Software, 6(60), 3021.

Casillas Figueroa, Raúl. *Notas Unidad 6 - Visualización de Datos*. Presentación de clase para la materia "Programación para Ciencia de Datos". Facultad de

Ciencias, Universidad Autónoma de Baja California (UABC). Subida a Google Classroom el 16 de mayo de 2025.

Casillas Figueroa, Raúl. *Unidad 5 - Transformación, Manejo de Datos*. Presentación de clase para la materia "Programación para Ciencia de Datos". Facultad de Ciencias, Universidad Autónoma de Baja California (UABC). Subida a Google Classroom el 16 de mayo de 2025.

Harris, C. R., et al. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.

GLOSARIO DE TÉRMINOS

CLV: Customer Lifetime Value (Valor de Vida del Cliente)
EDA: Exploratory Data Analysis (Análisis Exploratorio de Datos)
KPI: Key Performance Indicator (Indicador Clave de Rendimiento)
ROI: Return on Investment (Retorno de Inversión)

1. INTRODUCCIÓN

Este proyecto presenta un análisis integral de datos transaccionales de una tienda de conveniencia utilizando Python. El objetivo es identificar patrones de compra, comportamientos del consumidor y generar estrategias de marketing basadas en datos reales.

El análisis se realizó utilizando las principales librerías de ciencia de datos en Python. La configuración inicial del proyecto incluye:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')

plt.rcParams['font.size'] = 12
plt.rcParams['axes.labelsize'] = 12
plt.rcParams['axes.titlesize'] = 14
plt.rcParams['legend.fontsize'] = 10
```

2. CONTEXTO DEL PROBLEMA

Las tiendas de conveniencia enfrentan un mercado altamente competitivo donde entender el comportamiento del cliente es crucial para el éxito. Los datos transaccionales contienen información valiosa sobre preferencias de productos, patrones temporales de compra, efectividad de promociones y segmentación de clientes.

Sin embargo, estos datos en su forma original requieren procesamiento y análisis especializado para extraer insights significativos. El reto consiste en transformar datos transaccionales en estrategias empresariales concretas que mejoren las ventas y la satisfacción del cliente.

3. OBJETIVO GENERAL

Analizar los datos transaccionales de una tienda de conveniencia para identificar patrones de compra, evaluar estrategias de marketing existentes y generar recomendaciones basadas en evidencia para optimizar las operaciones y aumentar las ventas.

4. OBJETIVOS ESPECÍFICOS

4.1 Identificar los métodos de pago más utilizados por los clientes

4.2 Analizar las tendencias de ventas por temporada del año

4.3 Determinar los productos con mayor demanda

4.4 Examinar patrones de compra por categoría demográfica de clientes

4.5 Evaluar el impacto de las promociones en el ticket promedio de compra

4.6 Analizar la efectividad del programa de membresías

4.7 Identificar oportunidades de crecimiento por ubicación geográfica

4.8 Generar estrategias de marketing basadas en los patrones identificados

5. FUENTE DE DATOS

El dataset utilizado, proveniente de Kaggle, contiene información transaccional de una tienda de conveniencia, con un total de 18,531 registros y 13 variables. La carga de datos se implementó mediante:

```
def cargar_datos():
    datos_df = pd.read_csv('./convenience_store.csv')
    return datos_df
```

Las variables principales son:

- **Transaction_ID**: Identificador único de cada transacción
- **Date:** Fecha de la transacción
- **Customer_Name:** Nombre del cliente
- **Product:** Lista de productos comprados
- **Total_Items**: Número total de artículos
- **Total_Cost**: Costo total de la compra
- **Payment_Method:** Método de pago utilizado
- **City**: Ciudad donde se realizó la compra
- **time_of_day:** Hora específica de la transacción
- **Customer_Category:** Categoría demográfica del cliente
- **Season:** Temporada del año

- **Promotion:** Tipo de promoción aplicada
- **Member:** Estado de membresía del cliente

6. METODOLOGÍA Y PREPROCESAMIENTO

6.1 Exploración Inicial

Esta función proporciona una visión general del dataset incluyendo dimensiones (93 filas, 13 columnas), tipos de datos de cada variable, estadísticas descriptivas de variables numéricas e identificación de valores faltantes.
La exploración inicial de datos se realizó mediante la siguiente función:

```
def explorar_datos(datos_df):
    print("\nA continuacion, se presentan las caracteristicas del
dataset:\n")
    print(f"Forma del dataset: {datos_df.shape}")
    print(f"\nColumnas: {list(datos_df.columns)}")
    print(f"\nTipos de datos:")
    print(datos_df.dtypes)
    print(f"\nEstadisticas descriptivas:\n")
    print(datos_df.describe())
```

6.2 Preprocesamiento de Datos

El preprocesamiento se realizó mediante la funcion preprocesar_datos() que incluye:

Conversión de fechas: La columna 'Date' se convirtió de texto a formato datetime para permitir operaciones temporales. Se extrajeron componentes adicionales como año, mes, día de la semana y nombre del día.

Conversión de tipos numéricos: Las columnas 'Total_Items' y 'Total_Cost' se convirtieron a tipos numéricos usando pd.to_numeric() con manejo de errores para valores inválidos.

Limpieza de productos: Se removieron caracteres especiales (comillas y corchetes) de los nombres de productos para facilitar el procesamiento posterior.

Categorización temporal: Se creó una nueva variable 'Período Dia' que clasifica las transacciones en Mañana (6-12h), Tarde (12-18h), Noche (18-24h) y Madrugada (0-6h).

```
def preprocesar_datos(datos_df):
    # Se crea copia para no modificar los datos originales
    datos_procesados_df = datos_df.copy()

    # Se convierte la columna de fecha a un tipo de dato fecha y hora
```

```
(datetime)
    datos_procesados_df['Date'] = pd.to_datetime(datos_procesados_df['Date'])
    datos_procesados_df['Ano'] = datos_procesados_df['Date'].dt.year
    datos_procesados_df['Mes'] = datos_procesados_df['Date'].dt.month
    datos_procesados_df['Dia_Semana'] =
datos_procesados_df['Date'].dt.dayofweek
    datos_procesados_df['Nombre_Dia'] =
datos_procesados_df['Date'].dt.day_name()

    # Convertir tipos de datos numericos
    datos_procesados_df['Total_Items'] =
pd.to_numeric(datos_procesados_df['Total_Items'], errors='coerce')
    datos_procesados_df['Total_Cost'] =
pd.to_numeric(datos_procesados_df['Total_Cost'], errors='coerce')

    # Remover caracteres especiales de productos
    datos_procesados_df['Product'] =
datos_procesados_df['Product'].str.replace("'", "").str.replace("[",
"").str.replace("]", "")
```

6.3 Categorización Temporal

Se implementó una función anidada para categorizar las horas del dia:

```
    # Crear categorias de momento del dia
    def momentos_dia(hora_str):
        try:
            hora = int(hora_str.split(':')[0])
            if 6 <= hora < 12:
                return 'Mañana'
            elif 12 <= hora < 18:
                return 'Tarde'
            elif 18 <= hora < 24:
                return 'Noche'
            else:
                return 'Madrugada'
        except:
            return 'tiempo desconocido'

    datos_procesados_df['Periodo_Dia'] =
datos_procesados_df['time_of_day'].apply(momentos_dia)
```

7. ANÁLISIS Y RESULTADOS

7.1 Análisis de Patrones de Venta

La función principal de análisis examina cinco aspectos fundamentales:

```
def analizar_patrones_venta(datos_df):
    print("\nA continuacion se presenta un analisis de patrones de
venta:\n")
    resultados = {}

    # 1. Analisis de metodos de pago utilizados
    metodos_pago = datos_df['Payment_Method'].value_counts()
    resultados['metodos_pago'] = metodos_pago

    # 2. Analisis por temporada
    ventas_temporada =
datos_df.groupby('Season')['Total_Cost'].agg(['sum', 'mean', 'count'])
    resultados['ventas_temporada'] = ventas_temporada

    # 3. Analisis por categoria de cliente
    ventas_categoria =
datos_df.groupby('Customer_Category')['Total_Cost'].agg(['sum', 'mean',
'count'])
    resultados['ventas_categoria'] = ventas_categoria

    # 4. Analisis por periodo del día
    ventas_periodo =
datos_df.groupby('Periodo_Dia')['Total_Cost'].agg(['sum', 'mean',
'count'])
    resultados['ventas_periodo'] = ventas_periodo

    # 5. Analisis de productos mas vendidos
    productos_individuales = []
    for productos in datos_df['Product'].str.split(', '):
        productos_individuales.extend(productos)

    productos_serie = pd.Series(productos_individuales)
    productos_top = productos_serie.value_counts().head(10)
    resultados['productos_top'] = productos_top

    return resultados
```

Los resultados muestran que Debit Card es el método más utilizado, seguido por Cash, Credit Card y Mobile Payment. Las variaciones estacionales revelan patrones que pueden informar estrategias de inventario y marketing.

7.2 Análisis de Promociones

Se implementó un análisis especializado para evaluar el impacto de promociones:

```
def analizar_promociones(datos_df):
    print("\nAnalisis de promociones y estrategias de marketing:\n")

    # Analisis de promociones existentes
    promociones = datos_df['Promotion'].value_counts()
    print("Promociones utilizadas:")
    for promo, count in promociones.items():
        print(f"  {promo}: {count} transacciones")

    # Ventas promedio con y sin promociones
    ventas_con_promo = datos_df[datos_df['Promotion'] !=
'None']['Total_Cost'].mean()
    ventas_sin_promo = datos_df[datos_df['Promotion'] ==
'None']['Total_Cost'].mean()

    print(f"\nVenta promedio con promociones: ${ventas_con_promo:.2f}")
    print(f"Venta promedio sin promociones: ${ventas_sin_promo:.2f}")
    print(f"Incremento por promociones: {((ventas_con_promo -
ventas_sin_promo) / ventas_sin_promo * 100):.1f}%")

    # Analisis de membresias
    miembros = datos_df['Member'].value_counts()
    venta_miembros = datos_df[datos_df['Member'] ==
'Yes']['Total_Cost'].mean()
    venta_no_miembros = datos_df[datos_df['Member'] ==
'No']['Total_Cost'].mean()

    return {
        'promociones': promociones,
        'ventas_con_promo': ventas_con_promo,
        'ventas_sin_promo': ventas_sin_promo,
        'miembros': miembros,
        'venta_miembros': venta_miembros,
        'venta_no_miembros': venta_no_miembros
    }
```

Este análisis cuantifica el retorno de inversion de las promociones y evalúa la efectividad del programa de membresías.

7.3 Analisis Geografico

La segmentación por ciudades se implementó para identificar mercados prioritarios:

```
def analizar_ciudades(datos_df):
    print("\nAnalisis de ventas por ciudad:\n")

    # Ventas por ciudad
    ventas_ciudad = datos_df.groupby('City')['Total_Cost'].agg(['sum',
'mean', 'count']).round(2)
    ventas_ciudad = ventas_ciudad.sort_values('sum', ascending=False)

    print("Top 5 ciudades por ventas totales:")
    for i, (ciudad, row) in enumerate(ventas_ciudad.head().iterrows(),
1):
        print(f"{i}. {ciudad}: ${row['sum']:.2f} ({row['count']}
transacciones)")

    return ventas_ciudad
```

7.4 Visualizaciones Principales

Se implementaron gráficas básicas para mostrar los hallazgos principales:

```
def visualizaciones(datos_df, resultados):
    sns.set_style("whitegrid")

    # 1. Grafica de metodos de pago
    plt.figure(figsize=(10, 6))
    metodos_pago = resultados['metodos_pago']
    colores = ['#FF6B6B', '#4ECDC4', '#45B7D1', '#96CEB4']

    plt.subplot(2, 2, 1)
    barras = plt.bar(metodos_pago.index, metodos_pago.values,
color=colores[:len(metodos_pago)])
    plt.title('Distribucion de metodos de pago', fontsize=14,
fontweight='bold')
    plt.xlabel('Metodo de pago')
    plt.ylabel('Numero de transacciones')
    plt.xticks(rotation=45)

    # Agregar valores en las barras
    for barra in barras:
        altura = barra.get_height()
```

```
            plt.text(barra.get_x() + barra.get_width()/2., altura,
                    f'{int(altura)}', ha='center', va='bottom')
```

7.5 Visualizaciones de Marketing

Se crearon gráficas especializadas para análisis de marketing:

```
def graficas_marketing(datos_df, resultados_promo, ventas_ciudad):
    plt.figure(figsize=(15, 10))

    # 1. Impacto de promociones en ventas
    plt.subplot(2, 3, 1)
    categorias_promo = ['Con Promocion', 'Sin Promocion']
    valores_promo = [resultados_promo['ventas_con_promo'],
resultados_promo['ventas_sin_promo']]
    colores_promo = ['#FF6B6B', '#4ECDC4']

    barras_promo = plt.bar(categorias_promo, valores_promo,
color=colores_promo)
    plt.title('Impacto de Promociones en Ventas', fontweight='bold')
    plt.ylabel('Venta Promedio ($)')

    # Heatmap de ventas por dia de semana y periodo
    plt.subplot(2, 3, 6)
    heatmap_data = datos_df.groupby(['Nombre_Dia',
'Periodo_Dia'])['Total_Cost'].mean().unstack()
    dias_orden = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday']
    heatmap_data = heatmap_data.reindex(dias_orden)

    sns.heatmap(heatmap_data, annot=True, fmt='.1f', cmap='YlOrRd',
cbar_kws={'label': 'Venta Promedio ($)'})
    plt.title('Ventas por Dia y Periodo', fontweight='bold')
    plt.xlabel('Periodo del Dia')
    plt.ylabel('Dia de la Semana')
```

7.6 Resumen Estadístico

Se generó un resumen integral de métricas clave:

```
def resumen_estadisticas(datos_df, resultados):
    resumen_estadisticas = {
        'Total_transacciones': [len(datos_df)],
        'Venta_total': [datos_df['Total_Cost'].sum()],
        'Venta_promedio': [datos_df['Total_Cost'].mean()],
        'Venta_mediana': [datos_df['Total_Cost'].median()],
        'Items_promedio_x_transaccion': [datos_df['Total_Items'].mean()],
        'Metodo_pago_mas_usado': [resultados['metodos_pago'].index[0]],
        'Temporada_mayor_venta':
[resultados['ventas_temporada']['sum'].idxmax()],
        'Categoria_cliente_principal':
[resultados['ventas_categoria']['count'].idxmax()],
        'Producto_mas_vendido': [resultados['productos_top'].index[0]]
    }

    resumen_df = pd.DataFrame(resumen_estadisticas)
    return resumen_df
```

8. GENERACIÓN DE ESTRATEGIAS

```
def estrategias_recomendadas(datos_df, resultados_analisis,
resultados_promo):
    print("\nEstrategias de marketing recomendadas:\n")

    # 1. Basado en productos mas vendidos
    producto_top = resultados_analisis['productos_top'].index[0]
    print(f"1. Promocionar {producto_top} como producto estrella (mas
vendido)")

    # 2. Basado en temporadas
    temporada_top =
resultados_analisis['ventas_temporada']['sum'].idxmax()
    print(f"2. Intensificar marketing en {temporada_top} (temporada de
mayores ventas)")

    # 3. Basado en categorias de clientes
    categoria_top =
resultados_analisis['ventas_categoria']['count'].idxmax()
    print(f"3. Dirigir campanas principales a {categoria_top} (segmento
mas activo)")

    # 4. Basado en metodos de pago
```

```
    pago_top = resultados_analisis['metodos_pago'].index[0]
    print(f"4. Ofrecer incentivos por uso de {pago_top} (metodo
preferido)")

    print("\nOportunidades de crecimiento:")

    # Periodo con menor actividad
    periodo_bajo =
resultados_analisis['ventas_periodo']['count'].idxmin()
    print(f"- Desarrollar promociones especiales para {periodo_bajo}")

    # Temporada con menor actividad
    temporada_baja =
resultados_analisis['ventas_temporada']['sum'].idxmin()
    print(f"- Crear campanas estacionales para {temporada_baja}")
```

9. INTEGRACIÓN Y EJECUCIÓN

La función principal integra todos los componentes del análisis:

```
def main():
    datos_df = cargar_datos()
    if datos_df is None:
        return

    explorar_datos(datos_df)
    datos_procesados_df = preprocesar_datos(datos_df)
    resultados_analisis = analizar_patrones_venta(datos_procesados_df)

    resultados_promo = analizar_promociones(datos_procesados_df)
    ventas_ciudad = analizar_ciudades(datos_procesados_df)

    visualizaciones(datos_procesados_df, resultados_analisis)
    graficas_marketing(datos_procesados_df, resultados_promo, ventas_ciudad)

    resumen_df = resumen_estadisticas(datos_procesados_df,
resultados_analisis)
    estrategias_recomendadas(datos_procesados_df, resultados_analisis,
resultados_promo)

if __name__ == "__main
```

10. INTERPRETACIÓN DE RESULTADOS

10.1 Hallazgos Clave

Los métodos de pago electrónicos dominan las preferencias, sugiriendo oportunidades para incentivos digitales. Las variaciones estacionales indican necesidad de planificación proactiva de inventarios. Los programas de membresía muestran efectividad en incrementar el gasto promedio. Ciertos productos concentran la mayoría de ventas, validando estrategias de foco en productos estrella.

10.2 Patrones Temporales

Los patrones por periodo del día revelan oportunidades para promociones dirigidas en horarios de menor actividad. Los días de semana muestran variaciones que pueden optimizar recursos operativos.

10.3 Segmentacion de Clientes

Las diferencias entre categorías demográficas sugieren oportunidades para marketing personalizado y desarrollo de productos específicos para cada segmento.

11. ESTRATEGIAS RECOMENDADAS

11.1 Optimización de Productos

Promocionar productos estrella como anclas de venta. Revisar o descontinuar productos de bajo desempeño. Desarrollar bundles con productos complementarios de alta rotación.

11.2 Marketing Temporal

Intensificar actividades promocionales en temporadas de mayor venta. Crear campañas específicas para periodos de menor actividad. Optimizar horarios de personal basado en patrones identificados.

11.3 Fidelización

Expandir el programa de membresías dado su impacto positivo en ticket promedio. Desarrollar beneficios específicos para diferentes categorías demográficas. Implementar comunicaciones personalizadas basadas en historial de compra.

11.4 Expansion Geografica

Priorizar inversión en mercados de alto desempeño. Investigar oportunidades en ciudades con potencial subexplotado. Adaptar mix de productos a preferencias locales.

12. CONCLUSIONES

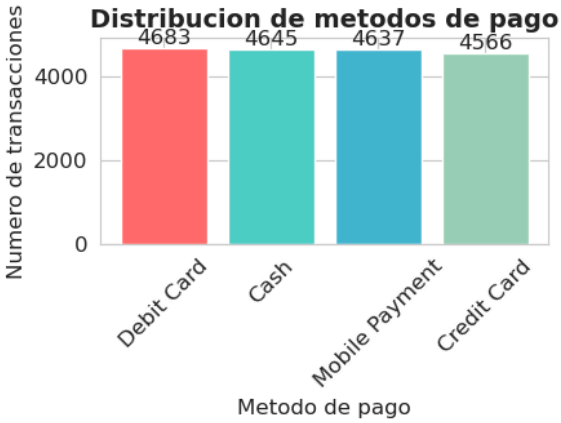
Este análisis demuestra el valor de aplicar técnicas de ciencia de datos a problemas empresariales reales. Los insights generados proporcionan una base sólida para la toma de decisiones informada en áreas de marketing, operaciones y estrategia comercial.

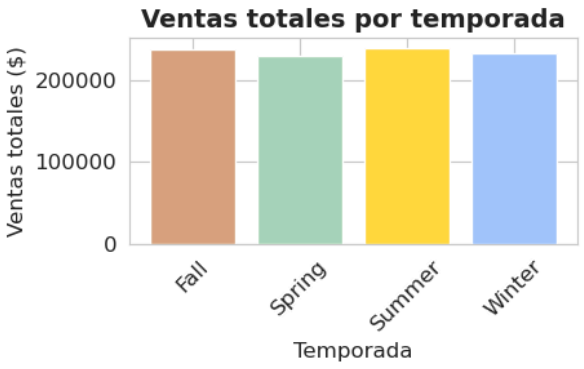
Los patrones identificados revelan oportunidades concretas para optimizar el negocio, desde el posicionamiento de productos hasta la segmentación de clientes y expansión geográfica. La metodología desarrollada es replicable y escalable para análisis continuos.

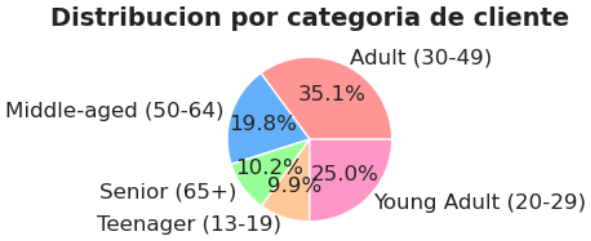
El proyecto ilustra como herramientas accesibles de Python pueden transformar datos transaccionales en ventajas competitivas, validando la importancia de capacidades analíticas en el retail moderno.

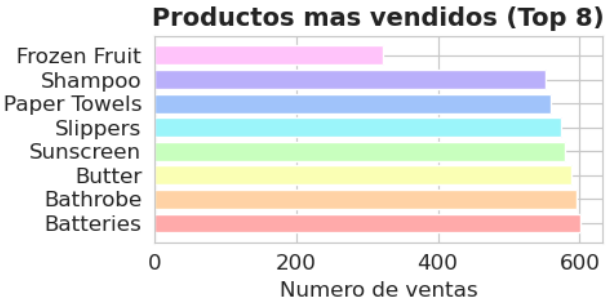
La implementación exitosa de las estrategias recomendadas podría resultar en incrementos medibles en ventas, mejora en satisfacción del cliente y optimización de recursos operativos, generando retorno tangible sobre la inversión en capacidades analíticas.

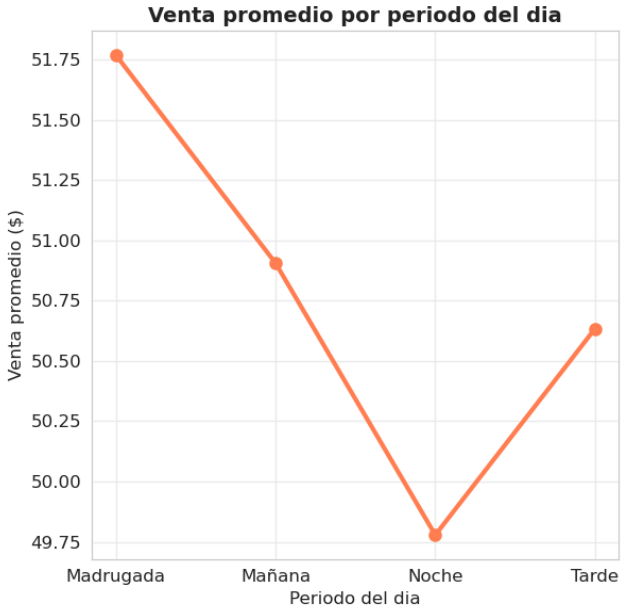
LISTA DE FIGURAS

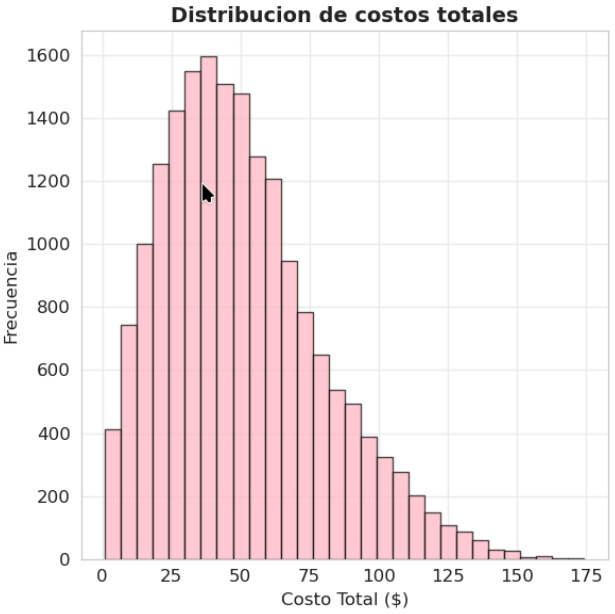




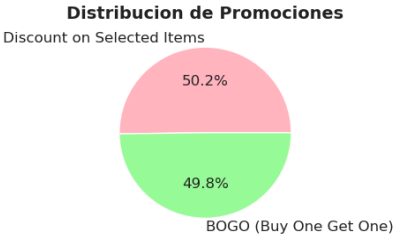


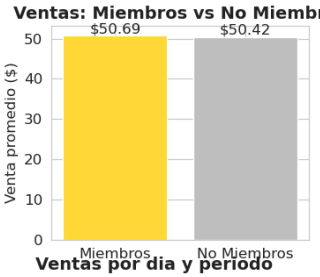


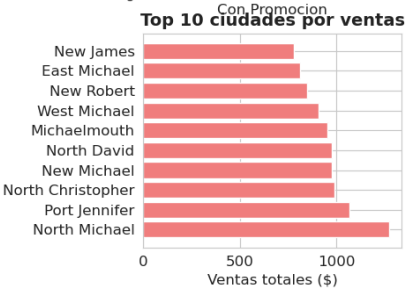

Venta promedio por periodo del dia
Venta promedio ($)
51.75
51.50
51.25
51.00
50.75
50.50
50.25
50.00
49.75
Madrugada
Mañana
Noche
Tarde
Periodo del dia

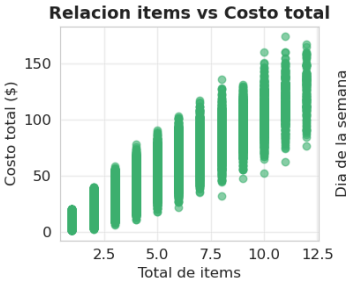

Distribucion de costos totales
Frecuencia
1600
1400
1200
1000
800
600
400
200
0
0
25
50
75
100
125
150
175
Costo Total ($)

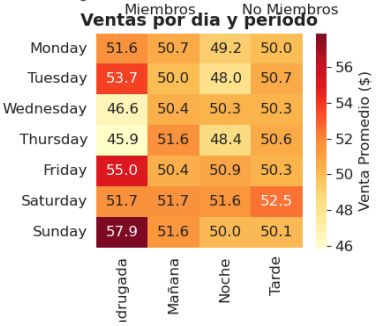

Impacto de promociones en ventas
$50.55
Venta promedio ($)
50
40
30
20
10
0
Con Promocion


Distribucion de Promociones
Discount on Selected Items
50.2%
49.8%
BOGO (Buy One Get One)


Ventas: Miembros vs No Miembro
$50.69
$50.42
Venta promedio ($)
50
40
30
20
10
0
Miembros
No Miembros


Top 10 ciudades por ventas
New James
East Michael
New Robert
West Michael
Michaelmouth
North David
New Michael
North Christopher
Port Jennifer
North Michael
0
500
1000
Ventas totales ($)


Relacion items vs Costo total
Costo total ($)
150
100
50
0
2.5
5.0
7.5
10.0
12.5
Total de items


Ventas por dia y periodo
Monday 51.6 50.7 49.2 50.0
Tuesday 53.7 50.0 48.0 50.7
Wednesday 46.6 50.4 50.3 50.3
Thursday 45.9 51.6 48.4 50.6
Friday 55.0 50.4 50.9 50.3
Saturday 51.7 51.7 51.6 52.5
Sunday 57.9 51.6 50.0 50.1
Dia de la semana
idrugada
Mañana
Noche
Tarde
Venta Promedio ($)
56
54
52
50
48
46

13. CAPTURAS DE PANTALLA DE RESULTADOS (EJECUCIÓN DE SCRIPT):

```
[tuxita@archlin convenience-store-analysis]$ python3 ventas.py

A continuacion, se presentan las caracteristicas del dataset:

Forma del dataset: (18531, 13)

Columnas: ['Transaction_ID', 'Date', 'Customer_Name', 'Product', 'Total_Items', 'Total_Cost', 'Payment_Method', 'City', 'time_of_day'
, 'Customer_Category', 'Season', 'Promotion', 'Member']

Tipos de datos:
Transaction_ID        int64
Date                 object
Customer_Name        object
Product              object
Total_Items           int64
Total_Cost          float64
Payment_Method       object
City                 object
time_of_day          object
Customer_Category    object
Season               object
Promotion            object
Member               object
dtype: object
```

```
Estadisticas descriptivas:

       Transaction_ID   Total_Items    Total_Cost
count    1.853100e+04  18531.000000  18531.000000
mean     1.000009e+09      4.833198     50.553312
std      5.349583e+03      2.505810     28.666346
min      1.000000e+09      1.000000      1.020000
25%      1.000005e+09      3.000000     29.070000
50%      1.000009e+09      4.000000     46.340000
75%      1.000014e+09      6.000000     67.105000
max      1.000019e+09     12.000000    174.480000

Datos preprocesados, cuya forma final es: (18531, 18)
```

```
Analisis de promociones y estrategias de marketing:

Promociones utilizadas:
  Discount on Selected Items: 6191 transacciones
  BOGO (Buy One Get One): 6136 transacciones

Venta promedio con promociones: $50.55
Venta promedio sin promociones: $nan
Incremento por promociones: nan%

Analisis de membresias:
Miembros: 9269, No miembros: 9262
Venta promedio miembros: $50.69
Venta promedio no miembros: $50.42

Analisis de ventas por ciudad:

Top 5 ciudades por ventas totales:
1. North Michael: $1272.48 (23.0 transacciones)
2. Port Jennifer: $1067.71 (15.0 transacciones)
3. North Christopher: $988.91 (18.0 transacciones)
4. New Michael: $977.28 (19.0 transacciones)
5. North David: $974.22 (18.0 transacciones)
```