Benemérita Universidad
Autónoma de Puebla

Facultad de Ciencias de la Computación

Reporte Fase 2: Limpieza de la base de datos

Ximena Axel Martínez Pelayo
21 de octubre de 2024

Introducción a la ciencia de datos

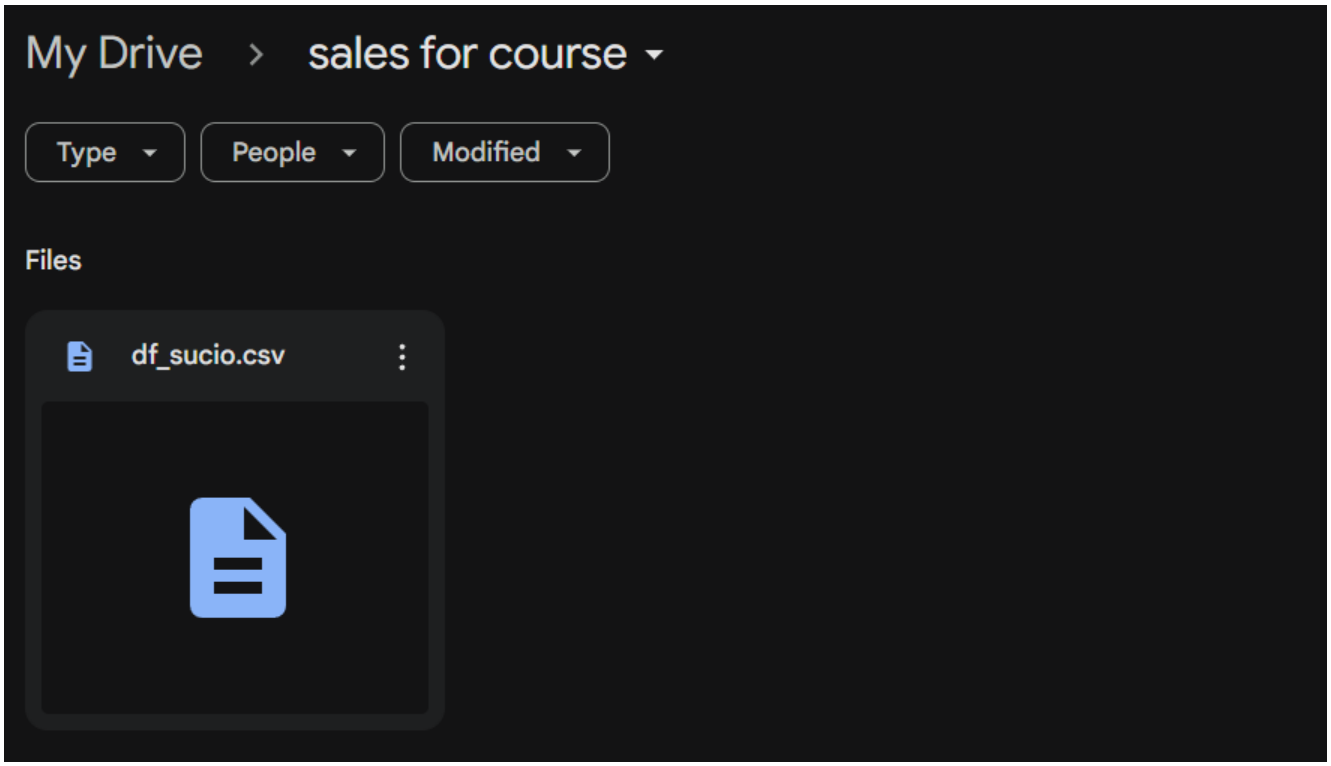
M.C. Jaime A. Romero Sierra

Manejo de la base de datos

El manejo adecuado de la base de datos es esencial para garantizar un análisis efectivo y confiable. A continuación, se describe el proceso detallado utilizado para trabajar con un archivo .csv obtenido de Kaggle.

Organización y Almacenamiento

Es importante almacenar el archivo .csv en un lugar accesible y seguro que soporte su tamaño en megabytes (MB). Para este proyecto, se utiliza Google Drive, una herramienta que facilita el almacenamiento en la nube y permite integrarse fácilmente con Google Colab. El archivo .csv se ubicará en una carpeta dedicada exclusivamente al proyecto, lo que asegura un entorno organizado.



Cargar la Base de Datos en Google Colab

Para trabajar con la base de datos en Google Colab, es necesario establecer una conexión entre la notebook y Google Drive. Este paso permite acceder al archivo almacenado en la nube. Para ello, se utiliza el siguiente código:

Este fragmento de código enlaza Google Drive en la notebook, otorgando acceso a los archivos almacenados en la carpeta del proyecto. Una vez completado este paso, es posible navegar por las carpetas de Google Drive desde el entorno de Colab y así cargar la base de datos con el siguiente código:

```
[125] #Cargar la base de datos
      df = pd.read_csv('/content/drive/MyDrive/sales for course/df_sucio.csv')
      df
```

Importar las Bibliotecas necesarias

Para trabajar con datos en Python, es necesario importar las bibliotecas adecuadas. Afortunadamente, Google Colab ya incluye muchas de las bibliotecas esenciales, como Pandas y NumPy, pero se deben importar explícitamente para garantizar su disponibilidad en el código.

Ejemplo de importación de bibliotecas:

```
[2]  #importar las bibliotecas
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

Si se estuviera trabajando en otro entorno, como Visual Studio Code, sería obligatorio instalar estas bibliotecas utilizando el administrador de paquetes pip. Esto asegura la compatibilidad del código en distintos entornos.

Identificación de Datos Sucios

Es fundamental identificar y corregir datos faltantes o incorrectos antes de realizar un análisis estadístico. Los valores faltantes o erróneos pueden alterar el orden natural de los datos, distorsionando los resultados y, en última instancia, influir negativamente en la toma de decisiones. Por ello es esencial revisar y validar las bases de datos antes de considerarlas como definitivas. Los principales tipos de problemas que se deben evitar incluyen:

- Datos nulos: Valores faltantes que pueden distorsionar cálculos como promedios, sumas y demás operaciones estadísticas.
- Datos duplicados: Registros redundantes que inflan los resultados y generan interpretaciones incorrectas.
- Datos incorrectos: Columnas que contienen valores que no coinciden con el tipo de dato esperado, como texto en una columna numérica.
- Valores inválidos: Datos que no tienen sentido dentro del contexto, como edades negativas o precios con valores cero.
- Datos atípicos: Valores extremos o inusuales que podrían ser errores o casos excepcionales. Si no se manejan adecuadamente, pueden sesgar el análisis.
- Inconsistencias en texto: Errores tipográficos o variaciones de formato, como diferencias en el uso de mayúsculas y minúsculas en categorías similares.

Evaluación Inicial

El primer paso en el análisis es inspeccionar las filas, también llamadas registros del archivo para entender su estructura y detectar posibles problemas. Para ellos, es recomendable obtener un resumen general de las columnas, tipos de datos y valores nulos presentes:

```
#Información de la base
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44611 entries, 0 to 44610
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   index             43273 non-null  float64
 1   Date              43272 non-null  object
 2   Year              43272 non-null  float64
 3   Month             0 non-null      float64
 4   Customer Age      42407 non-null  float64
 5   Customer Gender   43272 non-null  object
 6   Country           43272 non-null  object
 7   State             43272 non-null  object
 8   Product Category  43272 non-null  object
 9   Sub Category      43272 non-null  object
 10  Quantity          43272 non-null  float64
 11  Unit Cost         43272 non-null  object
 12  Unit Price        43272 non-null  object
 13  Cost              43272 non-null  float64
 14  Revenue           43273 non-null  float64
 15  Column1           3191 non-null   float64
dtypes: float64(8), object(8)
memory usage: 5.4+ MB
```

Tras evaluar la base de datos, se determinó que contiene 44,611 registros categorizados en 16 columnas. A continuación, se detalla el análisis de cada columna, su tipo de dato y las observaciones que se pueden hacer con la información obtenida:

Columna	Tipo de Dato	Notas de análisis
index	float64	Este es un tipo de dato numérico.
Date	object	Es una columna de tipo texto. Probablemente debiera convertida a tipo fecha.
Year	float64	Este es un tipo de dato numérico.
Month	float64	Tiene solo valores nulos. Debería ser eliminada o revisada.
Customer Age	float64	Este es un tipo numérico, pero tiene valores nulos que deberían ser tratados.
Customer Gender	object	Tipo de dato texto (categoría). Tiene valores nulos.
Country	object	Tipo de dato texto (categoría). Sin valores nulos.
State	object	Tipo de dato texto (categoría). Sin valores nulos.
Product Category	object	Tipo de dato texto (categoría). Sin valores nulos.

Sub Category	object	Tipo de dato texto (categoría). Sin valores nulos.
Quantity	float64	Tipo numérico. Sin valores nulos.
Unit Cost	object	Aunque es texto, debería ser numérico. Necesita conversión.
Unit Price	object	Aunque es texto, debería ser numérico. Necesita conversión.
Cost	float64	Tipo numérico. Sin valores nulos.
Revenue	float64	Tipo numérico. Sin valores nulos.
Column1	float64	La mayoría de los valores son nulos. Debería ser revisada o eliminada.

De acuerdo con la evaluación, los principales problemas identificados en la base de datos incluyen:

- Datos incorrectos: Columnas como Unit Cost y Unit Price están en formato texto y deben ser convertidas a numérico.
- Datos nulos: Columnas como Customer Age, Customer Gender, y Month contienen valores faltantes que requieren imputación o eliminación.
- Datos atípicos: Es necesario identificar y manejar valores extremos en columnas como Revenue y Cost.

Describir Estadísticas iniciales

La descripción estadística inicial de una base de datos es un paso fundamental en el análisis de datos. Este proceso no solo proporciona un panorama general del contenido, sino que también permite identificar tendencias, rangos, valores atípicos y posibles problemas que podrían afectar el análisis posterior.

```
[128] #Describir las estadísticas
      print(df.describe(include='all'))
```

```
                index     Date          Year  Month  Customer Age  \
count   43273.000000    43272  43272.000000    0.0  42407.000000
unique           NaN      576           NaN    NaN           NaN
top              NaN  invalid           NaN    NaN           NaN
freq             NaN      857           NaN    NaN           NaN
mean    17435.637603      NaN   2015.568428    NaN     36.390077
std     10051.100359      NaN      0.495301    NaN     11.122280
min         0.000000      NaN   2015.000000    NaN     17.000000
25%      8732.000000      NaN   2015.000000    NaN     28.000000
50%     17401.000000      NaN   2016.000000    NaN     35.000000
75%     26144.000000      NaN   2016.000000    NaN     44.000000
max     34866.000000      NaN   2016.000000    NaN     87.000000

       Customer Gender        Country       State Product Category  \
count            43272          43272       43272            43272
unique               2              4          46                4
top                  M  United States  California      Accessories
freq             22169          22458       12665            27428
mean               NaN            NaN         NaN              NaN
std                NaN            NaN         NaN              NaN
min                NaN            NaN         NaN              NaN
25%                NaN            NaN         NaN              NaN
50%                NaN            NaN         NaN              NaN
75%                NaN            NaN         NaN              NaN
max                NaN            NaN         NaN              NaN

           Sub Category      Quantity Unit Cost Unit Price          Cost  \
count             43272  43272.000000     43272      43272  43272.000000
unique               18           NaN       882       5111           NaN
top     Tires and Tubes           NaN   invalid    invalid           NaN
freq              13461           NaN       868        858           NaN
mean                NaN      2.001271       NaN        NaN    573.863815
std                 NaN      0.813315       NaN        NaN    687.566174
min                 NaN      1.000000       NaN        NaN      2.000000
25%                 NaN      1.000000       NaN        NaN     85.000000
50%                 NaN      2.000000       NaN        NaN    261.000000
75%                 NaN      3.000000       NaN        NaN    769.000000
max                 NaN      3.000000       NaN        NaN   3600.000000

             Revenue      Column1
count   43273.000000  3191.000000
unique           NaN          NaN
top              NaN          NaN
freq             NaN          NaN
mean      638.790297   682.708145
std       734.480650   772.447090
min         2.000000     2.000000
25%       102.000000   103.500000
50%       318.000000   389.000000
75%       901.000000   956.500000
max      5082.000000  3681.000000
```

A continuación, se presentan las observaciones obtenidas de la base de datos:

1. Análisis de Columnas Numéricas

   Las columnas numéricas (*Customer Age, Quantity, Cost, Revenue*) revelan las siguientes características:

   - Rangos razonables: Los valores están dentro de márgenes esperados, permitiendo realizar cálculos estadísticos confiables.

   - Distribución adecuada**:** La mayoría de las columnas muestran una distribución homogénea, aunque ciertas métricas (*Cost* y *Revenue*) presentan posibles valores extremos que deberán ser revisados más a fondo para determinar si se trata de outliers reales o errores en los datos.

2. Análisis de Columnas Categóricas

   Las columnas categóricas (*Product Category, Sub Category, Country*) tienen características que facilitan la segmentación y el análisis:

   - Distribuciones claras: Existen categorías predominantes, lo que puede guiar la creación de segmentos significativos para análisis específicos.
   - Inconsistencias detectadas**:** Algunas columnas, como *Unit Cost* y *Unit Price*, contienen valores inválidos o etiquetas incorrectas. Estas deben ser revisadas y corregidas para asegurar la calidad del análisis y permitir su conversión a tipos numéricos.

3. Identificación de Datos Faltantes

   El análisis confirma y detalla la presencia de datos faltantes:

   - Valores nulos en columnas clave: *Customer Age* y *Customer Gender* presentan valores nulos que deben ser tratados para evitar sesgos en los análisis posteriores.
   - Columnas con proporción alta de nulos: *Month* y *Column1* contienen demasiados valores faltantes, lo que podría justificar su eliminación o una evaluación más profunda de su relevancia.

Con base en este análisis preliminar, se puede concluir que la base de datos tiene un diseño funcional y un contenido valioso para el análisis, pero requiere ciertos ajustes para optimizar su calidad.

Limpieza de Datos Sucios Identificados

Una vez identificados los problemas presentes en la base de datos, podemos abordar su limpieza de manera más efectiva.

Cambio de formato

Para poder trabajar correctamente con los datos, es necesario asegurarse de que estén en el formato adecuado. En este paso, se verifica que los valores que no corresponden al formato esperado sean corregidos. Para identificar los datos con formato incorrecto, se utiliza el siguiente comando:

```
[129] #Se establece el formato óptimo de dato

      from datetime import datetime

      expected_types = {
          'index': float,
          'Date': datetime,
          'Year': float,
          'Month': float,
          'Customer Age': float,
          'Customer Gender': str,
          'Country': str,
          'State': str,
          'Product Category': str,
          'Sub Category': str,
          'Quantity': float,
          'Unit Cost': float,
          'Unit Price': float,
          'Revenue': float,
          'Column1': float
      }

      # Verifica que todos los valores en cada columna cumplan con el tipo esperado
      type_check_results = {
          column: df[column].apply(lambda x: isinstance(x, dtype) or pd.isna(x)).all()
          for column, dtype in expected_types.items()
      }

      # Muestra los resultados
      for column, is_correct_type in type_check_results.items():
          print(f"Columna '{column}': {'Correcta' if is_correct_type else 'Incorrecta'}")
```

```
Columna 'index': Correcta
Columna 'Date': Incorrecta
Columna 'Year': Correcta
Columna 'Month': Correcta
Columna 'Customer Age': Correcta
Columna 'Customer Gender': Correcta
Columna 'Country': Correcta
Columna 'State': Correcta
Columna 'Product Category': Correcta
Columna 'Sub Category': Correcta
Columna 'Quantity': Correcta
Columna 'Unit Cost': Incorrecta
Columna 'Unit Price': Incorrecta
Columna 'Revenue': Correcta
Columna 'Column1': Correcta
```

Como se puede observar, existen dos tipos de datos incorrectos. Estos deben ser convertidos al formato adecuado: numérico para algunas columnas y de tipo fecha para otras, como se muestra a continuación:

```
[130] #Convertir formato a valor numérico
      df['Unit Cost'] = pd.to_numeric(df['Unit Cost'], errors='coerce')
      df['Unit Price'] = pd.to_numeric(df['Unit Price'], errors='coerce')

[131] #Convertir fechas
      df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
```

Una vez realizadas las conversiones, se debe verificar que todo esté en orden. Para ello, se vuelve a ejecutar el comando inicial y, si los datos están correctamente formateados, se continúa con el siguiente paso:

```
# Verifica que todos los valores en cada columna cumplan con el tipo esperado
type_check_results = {
    column: df[column].apply(lambda x: isinstance(x, dtype) or pd.isna(x)).all()
    for column, dtype in expected_types.items()
}

# Muestra los resultados
for column, is_correct_type in type_check_results.items():
    print(f"Columna '{column}': {'Correcta' if is_correct_type else 'Incorrecta'}")
```

```
Columna 'index': Correcta
Columna 'Date': Correcta
Columna 'Year': Correcta
Columna 'Month': Correcta
Columna 'Customer Age': Correcta
Columna 'Customer Gender': Correcta
Columna 'Country': Correcta
Columna 'State': Correcta
Columna 'Product Category': Correcta
Columna 'Sub Category': Correcta
Columna 'Quantity': Correcta
Columna 'Unit Cost': Correcta
Columna 'Unit Price': Correcta
Columna 'Revenue': Correcta
Columna 'Column1': Correcta
```

Eliminación de Registros Duplicados

Ya se han identificado los registros duplicados. Para asegurar que el método utilizado para eliminarlos sea eficiente, es necesario conocer cuántos registros duplicados existen en la base de datos. Un registro se considera duplicado cuando los datos de todas las columnas coinciden exactamente con los de otro registro. Estos registros deben eliminarse para evitar duplicar información innecesaria. Para obtener el total de registros duplicados, se utiliza el siguiente comando:

```
[133] # Filas duplicadas
      print(df.duplicated().sum())

      3367
```

Para eliminar los registros duplicados, se utiliza el siguiente comando:

```
[134] #Eliminar los registros duplicados
      df.drop_duplicates(inplace = True)
```

Una vez eliminados, podemos verificar que los duplicados hayan sido eliminados correctamente con el siguiente comando:

```
[135] #Filas duplicadas corroboración
      print(df.duplicated().sum())

      0
```

Valores faltantes

Contar con datos completos es fundamental para realizar un análisis efectivo. En este paso, se identifican las columnas que contienen valores faltantes. Dependiendo del volumen de datos faltantes por columna, se evalúa si la columna es relevante o si debe eliminarse. En el caso de columnas con pocos datos faltantes, estos pueden ser imputados utilizando valores aproximados como la media o el promedio. Sin embargo, si la cantidad de datos faltantes es considerable, la columna se elimina, ya que podría sesgar el análisis.

Para identificar los valores faltantes en cada columna, se utiliza el siguiente comando:

```
[136] # Columnas con total de valores nulos
      print(df.isnull().sum())

      index                1336
      Date                 2185
      Year                 1336
      Month               41244
      Customer Age         2197
      Customer Gender      1338
      Country              1332
      State                1335
      Product Category     1337
      Sub Category         1334
      Quantity             1338
      Unit Cost            2193
      Unit Price           2189
      Cost                 1335
      Revenue              1336
      Column1             38292
      dtype: int64
```

```
[137] #Comprobar la extensión
      df.shape

      (41244, 16)
```

Es importante verificar la extensión de la base de datos, ya que los valores faltantes modifican el tamaño de la base, ya sea al disminuir o incrementar los registros. Tras revisar la extensión y analizar los datos faltantes, se observa que las columnas *Month* y *Column1* tienen un número excesivo de valores nulos. Por lo tanto, se opta por eliminar estas columnas con el siguiente comando:

```
# Eliminar las columnas sin datos
df = df.drop(columns=['Month','Column1'])
```

Después de eliminar las columnas, se verifica nuevamente la extensión de la base para asegurarse de que los registros eliminados no hayan afectado otras áreas de la base de datos:

```
[139] #Comprobar la nueva extensión
      df.shape

      (41244, 14)
```

En los casos restantes, en los que se imputarán valores, se procede con la imputación de los valores faltantes. En el caso de la columna *Date*, se decide imputar los valores faltantes como 'Unknown', lo cual mejora la calidad del registro, como se muestra en el siguiente código:

```
# Imputar con un valor fijo
df['Date'].fillna('Unknown', inplace=True)
df['Year'].fillna('Unknown', inplace=True)
```

Para la columna *Customer Age*, que es una variable continua, se elige la mediana como valor de imputación, ya que se espera que la edad tenga una distribución normal. El código para imputar la mediana es el siguiente:

```
[142] #Imputar Customer Age con mediana
      df['Customer Age'].fillna(df['Customer Age'].median(), inplace=True)
```

Similar a la edad, para la columna *Customer Gender*, se utiliza la moda para imputar los valores faltantes, ya que es más probable que el género siga una distribución que se ajuste a la moda. En este caso, aunque factores como el país o la categoría de producto podrían influir, la moda es una estimación razonable. El código para imputar la moda es el siguiente:

```
[143] #Imputar Customer Gender mediante la moda
      df['Customer Gender'].fillna(df['Customer Gender'].mode()[0], inplace=True)
```

Para las columnas *Country* y *State*, se puede aprovechar la posible relación con otras variables, como la región de ventas. Se utiliza la siguiente línea de código para imputar estos valores:

```
#Imputar Country y State mediante su relación
df['Country'].fillna(df['Country'].mode()[0], inplace=True)
df['State'].fillna(df['State'].mode()[0], inplace=True)
```

De manera similar, *Product Category* y *Sub Category* podrían estar relacionadas con otros datos, como *Cost* o *Quantity*, por lo que también se realiza la imputación con el siguiente código:

```
[145] #Imputar Product Category y Subcategory
      df['Product Category'].fillna(df['Product Category'].mode()[0], inplace=True)
      df['Sub Category'].fillna(df['Sub Category'].mode()[0], inplace=True)
```

En cuanto a los valores numéricos, se debe tener especial cuidado al imputar. Para la columna *Quantity*, que es una variable discreta, se utiliza la mediana para la imputación, ya que se espera que los valores no sean negativos y que su dispersión sea limitada. El código utilizado para imputar la mediana es el siguiente:

```
[146] #Imputar Quantity
      df['Quantity'].fillna(df['Quantity'].median(), inplace=True)
```

Para columnas como *Unit Cost, Unit Price, Cost* y *Revenue*, que son valores numéricos interdependientes, si falta uno de los valores, se puede calcular utilizando los demás. Si no es posible realizar este cálculo, se utiliza la mediana como alternativa para imputar el valor faltante.

```
df['Unit Cost'].fillna(df['Unit Cost'].median(), inplace=True)
df['Unit Price'].fillna(df['Unit Price'].median(), inplace=True)
df['Cost'].fillna(df['Cost'].median(), inplace=True)
df['Revenue'].fillna(df['Revenue'].median(), inplace=True)
```

Finalmente, para la columna *Index*, se puede reemplazar los valores faltantes por un rango numérico secuencial, lo que reiniciará el índice basado en el orden de las filas. Sin embargo, si el índice no es secuencial, como en este caso, la única opción disponible es eliminar los registros con valores faltantes. El código para hacerlo es el siguiente:

```
[151] #Índice
      df = df.dropna(axis=1)
```

Verificación Final

Una vez realizados todos los pasos anteriores, se verifica que ya no haya datos faltantes en la base de datos. Para ello, se utiliza la siguiente línea de código:

```
[152] # Columnas con total de valores nulos
      print(df.isnull().sum())

Date               0
Year               0
Customer Age       0
Customer Gender    0
Country            0
State              0
Product Category   0
Sub Category       0
Quantity           0
Unit Cost          0
Unit Price         0
Cost               0
Revenue            0
dtype: int64
```

Finalmente, se comprueba la extensión final de la base de datos para confirmar que todos los cambios se hayan aplicado correctamente:

```
[153] df.shape

(41244, 13)
```

Y se muestran las estadísticas finales:

```
#Describir las estadísticas de forma final
print(df.describe(include='all'))

              Date     Year  Customer Age Customer Gender        Country  \
count        41244  41244.0  41244.000000           41244          41244
unique         576      3.0           NaN               2              4
top        Unknown   2016.0           NaN               M  United States
freq          2185  22735.0           NaN           21790          22025
mean           NaN      NaN     36.319368             NaN            NaN
std            NaN      NaN     10.825552             NaN            NaN
min            NaN      NaN     17.000000             NaN            NaN
25%            NaN      NaN     28.000000             NaN            NaN
50%            NaN      NaN     35.000000             NaN            NaN
75%            NaN      NaN     43.000000             NaN            NaN
max            NaN      NaN     87.000000             NaN            NaN
```

	State	Product Category	Sub Category	Quantity	\
count	41244	41244	41244	41244.000000	
unique	46	4	18	NaN	
top	California	Accessories	Tires and Tubes	NaN	
freq	12965	26635	13769	NaN	
mean	NaN	NaN	NaN	2.001649	
std	NaN	NaN	NaN	0.799945	
min	NaN	NaN	NaN	1.000000	
25%	NaN	NaN	NaN	1.000000	
50%	NaN	NaN	NaN	2.000000	
75%	NaN	NaN	NaN	3.000000	
max	NaN	NaN	NaN	3.000000	

	Unit Cost	Unit Price	Cost	Revenue
count	41244.000000	41244.000000	41244.000000	41244.000000
unique	NaN	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN
mean	338.063005	376.903311	564.107167	629.049984
std	475.917397	511.912443	679.589018	726.201449
min	0.670000	0.666667	2.000000	2.000000
25%	48.000000	57.500000	90.000000	106.000000
50%	150.000000	178.000000	261.000000	319.000000
75%	418.000000	487.000000	769.000000	873.000000
max	3240.000000	5082.000000	3600.000000	5082.000000

Conclusiones mediante la estadística

El resultado muestra que el conjunto de datos describe transacciones de clientes con información sobre el año, edad, género, ubicación geográfica, y detalles financieros de productos como cantidad, costo y precio. La mayoría de las transacciones corresponden a 2016, con una edad promedio de 36 años y una mayor presencia masculina. Los productos más vendidos son de la categoría "Accesorios" y la subcategoría "Neumáticos y Tubos", lo que sugiere que estos productos son clave en las ventas.

Los precios unitarios varían considerablemente, lo que refleja una oferta de productos desde económicos hasta de mayor precio. La mayoría de las compras son pequeñas, con 1 a 3 unidades adquiridas por transacción. Esto indica que los clientes compran productos según necesidades específicas.

Se puede predecir que los "Accesorios" seguirán siendo populares, pero también es importante explorar estrategias para productos más caros, como los "Neumáticos", adaptándose a la diversidad de clientes en edad, ubicación y preferencias.