Icono

El contenido generado por IA puede ser incorrecto. Logotipo

El contenido generado por IA puede ser incorrecto.



TECNICATURA SUPERIOR EN

**Ciencia de Datos e Inteligencia Artificial**

Módulo Innovación de Datos - Evidencia de Aprendizaje 3



Estudiantes:

* Fava Perez, Maria Pia: 30722626 | [piafava@gmail.com](mailto:piafava@gmail.com)
* Molina Pasi, Lorenzo: 41712829 | [molinaplorenzo@gmail.com](mailto:molinaplorenzo@gmail.com)
* Ontiveros, Manuel Alberto: 28127248 | manu.ontiveros023@gmail.com
* Ortega, Santiago Andrés: 32739392 | ortegasantiagoandres@gmail.com
* Rivas, Mariela Yanina: 34677549 | [myantequera@gmail.com](mailto:myantequera@gmail.com)

Profesores:

* Charletti, Carlos Ignacio
* Mainero, Alejandro

Cohorte: 2025.

Contenido

[Evidencia de Aprendizaje nº 3: Exploración, Transformación y Limpieza de Datos utilizando Pandas 2](#_heading=h.klzjsqg7l847)

[1. Introducción 3](#_heading=h.azm0k3htuxiu)

[2. Proceso ETL – Python y Pandas 3](#_heading=h.g9gx4w9socgf)

[3. Carga de datos y consultas SQL 15](#_heading=h.zbv9q55nl4et)

[4. Análisis y síntesis del estudio 17](#_heading=h.ammnq1hgv1kc)

[5. Conclusiones 20](#_heading=h.z9jjk7u1553o)

[6. Entregables 21](#_heading=h.3sj0pzmtpan5)

[7. Bibliografía 21](#_heading=h.obtrm7ywt6iq)

# Evidencia de Aprendizaje nº 3: Exploración, Transformación y Limpieza de Datos utilizando Pandas

# 

## Introducción

El presente trabajo práctico tiene como objetivo adquirir experiencia en el proceso ETL (Extracción, Transformación y Limpieza) mediante el tratamiento de un conjunto de datos que combina información de ventas y datos de clientes. A través de la implementación de un flujo de trabajo estructurado, se busca integrar, depurar y enriquecer los datasets utilizando herramientas como la librería Pandas en Python, con el fin de preparar los datos para su posterior análisis en un sistema de gestión de bases de datos relacional (MySQL).

Durante el desarrollo del proyecto, se aplican técnicas de limpieza, estandarización y categorización de variables, incluyendo la creación de rangos etarios y segmentos de precios. Además, se incorpora información geográfica y socioeconómica para ampliar el contexto del análisis. Este enfoque permite construir un DataFrame listo para consultas SQL y generación de insights relevantes sobre el comportamiento de compra de los clientes.

A lo largo del informe se consignan los principales bloques de código utilizados en cada etapa del proceso. Para acceder al código completo, se debe consultar el archivo fuente del notebook: **Evidencia3\_con\_conexion.ipynb**. El mismo contiene comentarios explicativos en cada bloque.

## Proceso ETL – Python y Pandas

Describimos las operaciones de Extracción, Transformación y Limpieza realizadas en los datos de ventas y de clientes, usando Python y la librería Pandas.

*Extracción de datos – Extract*

Se cargaron los archivos sales\_data.csv y customer\_data.csv en dos DataFrames distintos (df\_sales y df\_customers). Se procedió a una inspección inicial de cada uno para determinar sus dimensiones, encabezados, tipos de datos y la presencia de valores nulos. Ambos datasets contienen 99.457 registros y no presentan valores nulos.

*Código Python para la extracción:*

| *import* pandas *as* pd  *import* matplotlib.pyplot *as* plt    *#Importamos los data set*  df\_sales *=* pd.read\_csv('sales\_data.csv')  df\_customers *=* pd.read\_csv('customer\_data.csv')    *#Análisis Exploratorio de df\_customer#*  • Dimensiones *del* Data Frame (DF en adelante).  • Header *del* DF.  • Info *del* DF (Permite saber si existen valores nulos, el tipo de los  datos contenidos y una vista con más detalle de las columnas).  print('--- Características del DataFrame Customers ---')  print(df\_customers.shape)  print(df\_customers.head())  print(df\_customers.info())    *# Analisis Exploratorio de df\_sales*  • Dimensiones *del* Data Frame.  • Header *del* DF.  • Info *del* DF.  print('--- Características del DataFrame Sales ---')  print(df\_sales.shape)  print(df\_sales.head())  print(df\_sales.info()) |
| --- |

*Concatenar los dos DataFrames anteriores en uno final*

Renombramos las columnas y realizamos la unión (merge) de los dos DataFrames usando la columna común COSTUMER ID (renombrada en mayúscula para tener consistencia).

| *### Cambiar el tipo de datos de object a string*  df\_compradores *=* df\_customers  df\_compradores *=* df\_compradores.astype({'customer\_id':'str','gender':'str', 'payment\_method':'str'})  print(df\_compradores.dtypes)  '''En Pandas object almacena cadenas de texto (str) razón por la cual el print anterior los sigue mostrando como object, induciendonos al error de que no tuve efecto el .astype. Para corroborarlo, debemos pedir el tipo de dato, columna por columna.'''  print(type(df\_compradores['customer\_id'].iloc[0]))  print(type(df\_compradores['gender'].iloc[1]))  print(type(df\_compradores['payment\_method'].iloc[3]))    *### Reordenar el DF en base a la columna costumer\_id*  df\_compradores *=* df\_compradores.sort\_values(by*=*'customer\_id')  print(df\_compradores.head())    *### Poner los nombres de las columnas en mayúsculas*  df\_compradores.columns *=* df\_compradores.columns.str.upper()  print(df\_compradores.head())    *### Chequear si existen valores duplicados para customer\_id*  duplicados *=* df\_compradores['CUSTOMER\_ID'].duplicated(keep*=*False)  total\_duplicados *=* duplicados.sum()  print(*f*"Número total de filas con un 'customer\_id' duplicado: {total\_duplicados}")    *#Modificaciones al DF Sales*  *### Cambiar tipos de datos a str y datetime*  df\_ventas *=* df\_sales  df\_ventas *=* df\_ventas.astype({'invoice\_no':'str','customer\_id':'str','category':'str', 'shopping\_mall':'str'})  print(df\_ventas.dtypes)    *#Cambiamos invoice\_date a DATETIME*  df\_ventas['invoice\_date'] *=* pd.to\_datetime(df\_ventas['invoice\_date'], dayfirst*=*True)  df\_ventas['invoice\_date'] *=* df\_ventas['invoice\_date'].dt.strftime('%d-%m-%Y')  print(df\_ventas.head())    *# Verifica el nuevo formato de las fechas*  print(df\_ventas.head())  print(type(df\_ventas['invoice\_no'].iloc[0]))  print(type(df\_ventas['customer\_id'].iloc[1]))  print(type(df\_ventas['category'].iloc[2]))  print(type(df\_ventas['shopping\_mall'].iloc[6]))  print(type(df\_ventas['invoice\_date'].iloc[5]))    *### Reordenar el DF en base a la columna costumer\_id*  df\_ventas *=* df\_ventas.sort\_values(by*=*'customer\_id')  print(df\_ventas.head())    *### Poner los nombres de las columnas en mayusculas*  df\_ventas.columns *=* df\_ventas.columns.str.upper()  print(df\_ventas.head())    *### Chequear si existen valores duplicados para customer\_id e invoice\_no*  duplicados *=* df\_ventas['CUSTOMER\_ID'].duplicated(keep*=*False)  total\_duplicados *=* duplicados.sum()  print(*f*"Número total de filas con un 'customer\_id' duplicado: {total\_duplicados}")  duplicados\_in *=* df\_ventas['INVOICE\_NO'].duplicated(keep*=*False)  total\_duplicados\_in *=* duplicados\_in.sum()  print(*f*"Número total de filas con un 'invoice\_no' duplicado: {total\_duplicados\_in}")  duplicados\_cust *=* df\_ventas['CUSTOMER\_ID'].duplicated().sum()  duplicados\_inv *=* df\_ventas['INVOICE\_NO'].duplicated().sum()  print(*f*"N° de customer\_id duplicados: {duplicados\_cust}")  print(*f*"N° de invoice\_no duplicados: {duplicados\_inv}")    *# Unir ambos DF mediante la columna CUSTOMER\_ID*  df\_compradores\_ventas *=* pd.merge(df\_compradores, df\_ventas, on*=*'CUSTOMER\_ID', how*=*'inner')  print(df\_compradores\_ventas)    Para la concatenación de los dos DataFrames usamos la función pd.merge() aplicando un INNER JOIN (how='inner') sobre la clave común CUSTOMER ID. Esto asegura que el DataFrame final (df\_analisis\_FINAL) solo contenga a los clientes que tienen registros de ventas coincidentes, eliminando cualquier cliente sin historial transaccional para fines de este análisis. |
| --- |

*Transformación de los datos*

Las operaciones de transformación se centraron en dos ejes principales: el manejo de valores faltantes (NaN) y la ingeniería de características, con el objetivo de preparar el DataFrame para análisis exploratorios y consultas SQL.

#### Manejo de Valores Nulos y Estandarización

* Limpieza de Fechas: La columna INVOICE DATE fue convertida a tipo datetime.
* Geolocalización: Las columnas LATITUD y LONGITUD contenían nulos (NaN) que fueron producto de la imputación de datos geográficos. Se decidió imputar estos valores faltantes con 0.0 para permitir la carga a la base de datos, aunque el valor cero claramente indica un dato faltante o inválido.

*Transformaciones adicionales, ingeniería de características.* Con el fin de facilitar el análisis del comportamiento de compra por grupos, creamos dos variables categóricas: INTERVALOS\_AGE y INTERVALOS\_PRICE.

* Rangos Etarios **(**INTERVALOS\_AGE**):** Se definieron intervalos de edad en función de la distribución observada en el dataset.

| bins *=* [18, 28, 38, 48, 58, 70]  labels *=* ['18 - 27 Años', '28 - 37 Años', '38 - 47 Años', '48 - 57 Años', '58 - 70 Años']  df\_compradores\_ventas['INTERVALOS\_AGE'] *=* pd.cut(  df\_compradores\_ventas['AGE'], bins *=* bins,  labels *=* labels, include\_lowest *=* True,  right *=* True  )  print("#--- DF con la nueva columna de rango ---#")  print(df\_compradores\_ventas[['AGE', 'INTERVALOS\_AGE']].head()) |
| --- |

* Segmentos de Precio **(**INTERVALOS\_PRICE**):**Agrupamos los valores de PRICE en rangos definidos por el comportamiento de compra observado.

| bins *=* [ df\_compradores\_ventas['PRICE'].min() *-* 0.01,  45.45,  203.30,  1200.32,  df\_compradores\_ventas['PRICE'].max() *+* 0.01  ]  labels *=* ['Bajo (5-45)', 'Medio (45-203)', 'Alto (203-1200)', 'Premium (>1200)']  df\_compradores\_ventas['INTERVALOS\_PRICE'] *=* pd.cut(  df\_compradores\_ventas['PRICE'], bins *=* bins,  labels *=* labels, include\_lowest *=* True,  right *=* True  ) |
| --- |

Estas nuevas variables permiten realizar análisis segmentados por edad y nivel de gasto, facilitando la detección de patrones de consumo y preferencias de pago.

*Carga de datos (Python a SQL)*

El DataFrame limpio y transformado (df\_analisis\_final) fue cargado directamente a la base de datos relacional MySQL (schema innovacion1) utilizando el conector SQLAlchemy y el método .to\_sql() de Pandas.

#### Sobre las restricciones de integridad, en este paso (carga de Pandas a la tabla temporal ventas1\_temp), se garantiza la integridad de la estructura al usar el parámetro if\_exists='replace', que sobrescribe la tabla asegurando que el esquema coincida con el DataFrame más reciente. Las conversiones de tipos y las validaciones finales de integridad las hacemos en el paso siguiente, directamente con SQL.

| *#Instalación del paquete que permite la conexión con MySQL*  ***!***pip install pymysql  ***!***pip install mysql*-*connector*-*python    *#Importar módulos*  *import* pandas *as* pd  *from* sqlalchemy *import* create\_engine  *import* mysql.connector    *# Datos de conexión*  USUARIO *=* "root"  CONTRASEÑA *=* "Argentina2025!!"  HOST *=* "localhost"  PUERTO *=* "3306"  BASE\_DE\_DATOS *=* "innovacion1"  TABLA\_DESTINO *=* "ventas1\_temp"    *# Crear la URL de conexión*  DATABASE\_URL *=* *f*"mysql+mysqlconnector://{USUARIO}:{CONTRASEÑA}@{HOST}:{PUERTO}/{BASE\_DE\_DATOS}"    *# El motor de conexión está creado, lo redefinimos por si acaso*  *try*:  engine *=* create\_engine(DATABASE\_URL)  print("Conexión con SQLAlchemy establecida.")  *except* Exception *as* e:  print(*f*"Error al conectar: {e}")  exit()    *#Carga del dataframe*  *try*:  print(*f*"Iniciando carga del DataFrame en {BASE\_DE\_DATOS}.{TABLA\_DESTINO}...")    *# Usamos df\_analisis\_final*  df\_analisis\_final.to\_sql(  name*=*TABLA\_DESTINO,  con*=*engine,  if\_exists*=*'replace',  index*=*False,  )  *# len(df\_analisis\_final) cuenta las filas*  print(*f*"\n✔ ¡Carga exitosa! Se cargaron {len(df\_analisis\_final)} registros en {BASE\_DE\_DATOS}.{TABLA\_DESTINO}")    *except* Exception *as* e:  print(*f*"\n✖ Error durante la carga a SQL: {e}") |
| --- |

## Carga de datos y consultas SQL

*Esquema de la base de datos y sentencias SQL*

Llevamos los datos a la base de datos relacional MySQL. Para la carga final, utilizamos una tabla temporal (ventas1\_temp) para recibir los datos tal como los exportó Python (con todos los datos como texto y los encabezados con espacios). Posteriormente, se usó la sentencia INSERT INTO para crear la tabla final (ventas1) con los tipos de datos correctos (DATETIME, DOUBLE) y aplicando la transformación final de la columna de fecha.

El proceso en MySQL se centró en definir el esquema final con tipos de datos optimizados y en la validación de integridad.

El script completo de la creación de la base de datos, la tabla de ventas y la inserción de datos, se encuentra en el archivo SQL\_create\_ventas1.sql

*Creación del esquema y carga*

El script SQL\_create\_ventas1 muestra la definición del esquema innovación y crea la tabla ventas1 (CREATE TABLE) donde se estandarizan los nombres de columna y se definen tipos de datos específicos (DATETIME, DOUBLE) para el análisis. La sentencia INSERT INTO es fundamental, ya que ejecuta la conversión final del formato de fecha utilizando la función STR\_TO\_DATE() y aplica los casteos necesarios.

*Trigger: implementación de integridad*

Para garantizar la integridad lógica de las transacciones futuras, implementamos un trigger (TRG\_validar\_cantidad\_positiva).

Este procedimiento se ejecuta antes de cualquier intento de inserción (BEFORE INSERT) en la tabla ventas1 y aplica una regla de negocio fundamental: la cantidad vendida (QUANTITY) debe ser siempre mayor a cerodeteniendo la transacción si la condición no se cumple.

Archivo que lo contiene: SQL\_queries\_ventas1.sql

*Stored Procedure: automatización del análisis*

Creamos el Stored Procedure Generar\_Informe\_Tendencias\_Pago para encapsular y automatizar la ejecución de las consultas analíticas clave sobre tendencias de pago. Esto optimiza la ejecución (código precompilado) y facilita el acceso a los hallazgos sin necesidad de volver a escribir las consultas.

Archivo que lo contiene: SQL\_queries\_ventas1.sql

*Sentencias SQL para análisis de datos*

Ejecutaron consultas SQL básicas sobre la table ventas1 para responder a las preguntas planteadas en la fase de transformación.

Se encuentran en el archivo SQL\_queries\_ventas1.sql y contienen los bloques de consultas:

* Modo de pago más frecuente (general y por género): consultas para determinar el medio de pago dominante.
* Métodos de pago por rango etario: segmentación por el intervalo de edad clave (28-37 años).
* Comportamiento de compra (precios por categoría): cruce para identificar el rango de precio dominante por categoría (ej., Ropa).
* Distribución de transacciones por rango de precio: medición de la frecuencia porcentual de los tickets (bajo, medio, *premium*) para identificar el patrón de consumo medio-bajo (consulta utilizada para obtener el 50.21% de tickets).
* Jerarquía de ventas geográficas: consulta utilizada para obtener la distribución porcentual de transacciones por centro comercial (por ejemplo, Mall of Istanbul y Kanyon).

## Análisis y síntesis del estudio

El proceso ETL concluyó de manera correcta, generando una tabla analítica (innovacion1.ventas1) con 99,457 registros limpios y tipificados. La etapa de transformación en Pandas y la posterior tipificación y aplicación de un trigger de integridad en MySQL (TRG\_validar\_cantidad\_positiva) aseguraron la calidad y robustez del *dataset*. El análisis exploratorio se realizó mediante consultas SQL (incluyendo el stored procedure Generar\_Informe\_Tendencias\_Pago), lo cual reveló patrones clave en la composición demográfica, el comportamiento de compra y la cultura financiera de los consumidores en los centros comerciales de Estambul.

a. Análisis demográfico y comportamiento del consumidor

*Perfil del cliente y patrones de consumo por género*

Encontramos una base de clientes con mayor proporción femenina (59.81% mujeres vs. 40.19% hombres). El análisis demográfico destaca una distribución etaria equilibrada: los centros comerciales atraen por igual a segmentos extremos (18-27 años y 58-70 años), sugiriendo un espacio de consumo amplio.

A pesar de la diferencia numérica, el comportamiento de compra dentro de los grupos es uniforme: la cantidad de artículos adquiridos por transacción es prácticamente idéntica entre géneros. La diferencia en el volumen total de ventas se debe exclusivamente a la mayor proporción de clientas.

| Categoría | Mayor Inclinación |
| --- | --- |
| Femenina | Ropa (34.72%), Cosméticos (15.25%), Souvenirs y Juguetes. |
| Masculina | Calzado (10.17%), Comida y Bebida (14.94%), Libros y Tecnología. |

*Segmentación por Precio y Rango Etario*

La distribución de tickets muestra un patrón de consumo predominantemente medio-bajo: el 50.21% de las transacciones son tickets bajos o medios, mientras que los tickets premium solo representan el 19.14%. Esto indica que la mayoría de los consumidores en Estambul busca productos accesibles o de precio moderado.

El grupo etario 28-37 años (segmento clave) confirma esta tendencia, mostrando una fuerte preferencia por el uso de la tarjeta de crédito para sus transacciones, lo cual confirma su estabilidad financiera y dependencia de instrumentos bancarios para el consumo.

b. Análisis espacial y socioeconómico

*Distribución y jerarquía por centros comerciales*

El análisis revela una clara jerarquía en el ecosistema *retail*, con una concentración significativa en cuatro ubicaciones que capturan el 64.49% de las ventas. Mall of Istanbul (20.05%) y Kanyon (19.93%) son los líderes indiscutibles.

El entrecruzamiento de datos confirma la coherencia en algunos segmentos: Emaar Square Mall (perfil Alto/Clase Media-Alta) lidera en tickets *premium* (19.91%), mientras que Forum Istanbul (Medio/Masivo) domina los tickets medios (24.80%). Sin embargo, centros de lujo como İstinye Park y Zorlu Center no muestran la dominancia esperada en el segmento *premium*, sugiriendo que atraen a un público más diverso o que la muestra no es del todo representativa.

*Medios de Pago y Cultura Financiera*

El dato más revelador es el dominio absoluto del efectivo (*cash*), que representa el 44.69% de las transacciones, superando la suma de tarjetas de crédito (35.12%) y débito (20.19%). Este patrón es consistente en todos los centros comerciales y grupos etarios, reflejando una cultura de pago en efectivo arraigada en la región.

Las mujeres muestran una preferencia ligeramente mayor por tarjetas de crédito (35.32% vs. 34.82% hombres), un indicio que puede ser explotado en programas de fidelización dirigidos.

*Inteligencia de negocios y estrategias de escalamiento*

La aplicación de los hallazgos en inteligencia de negocios genera oportunidades estratégicas para la gestión de productos, inventario y fidelización. A continuación compartimos algunas sugerencias:

* *Optimización del mix de productos y estrategia de ropa*.La categoría Ropa es el principal motor de volumen (34.68% de las ventas). El 79.88% de las ventas de ropa se concentra en el segmento medio-alto ($203-$1200). Estrategia Recomendada: la expansión de la oferta de marcas en el rango de precio $203-$1200 maximizará el volumen de ventas al capturar el segmento que ya demuestra el mayor gasto. Esto implica priorizar marcas de *fast fashion premium* y marcas locales competitivas en este rango.
* *Estrategia Diferenciada por Centro Comercial*.La estrategia de *retail* debe ser adaptada al perfil sociodemográfico de cada ubicación:

| Centros Comerciales | Perfil y Enfoque Estratégico |
| --- | --- |
| Kanyon y Metrocity | Ejecutivos/Medio-Alto: fortalecer la oferta de tecnología y calzado premium. Implementar servicios para capturar tráfico post-laboral. |
| Mall of Istanbul y Forum Istanbul | Medio/Masivo: Ampliar la oferta de cosméticos y juguetes (dominantes en segmento medio). Desarrollar promociones fuertes orientadas al pago en efectivo. |
| İstinye Park y Zorlu Center | Lujo (Híbrido): Mantener los productos de gama alta y diversificar la oferta: introducir marcas de affordable luxury en el rango $203-$1200 para capturar al público diverso que realmente visita estos comercios. |

* *Estrategia de pagos y fidelización (cash-to-digital)*.El dominio del efectivo (44.69%) es el principal desafío y la mayor oportunidad para mejorar la calidad de los datos y la fidelización. Se podría implementar un puesto de conversión de efectivo a a tarjeta prepaga del centro comercial. Al incentivar digitalmente estos pagos, se logra capturar el 44% de las transacciones "ciegas", mejorando radicalmente la calidad del dato imposible de rastrear con el efectivo y la fidelización dentro del ecosistema del centro.

Los datos revelan un mercado con segmentación clara por género, edad y ubicación, pero con amplias oportunidades en digitalización de pagos, optimización de mix de productos por perfil de centro comercial, y estrategias de fidelización basadas en los segmentos identificados. La clave del escalamiento exitoso radica en adaptar el modelo operativo al perfil socioeconómico específico del distrito, evitando una aproximación genérica.

## Conclusiones

El trabajo finaliza con la validación del proceso ETL que transformó datos brutos en una fuente estructurada de inteligencia de negocios. El proyecto aseguró la calidad del dato mediante la correcta tipificación en MySQL y la implementación de herramientas de integridad (el Trigger TRG\_validar\_cantidad\_positiva y el Stored Procedure Generar\_Informe\_Tendencias\_Pago), confirmando la capacidad para sostener un análisis de gran escala.

Los hallazgos extraídos de los 99.457 registros revelan un mercado con oportunidades y desafíos específicos que deben guiar la estrategia futura:

* El dominio del efectivo como desafío central: el principal desafío para la estrategia de retail es la fuerte cultura de pago en efectivo (cash), que abarca el 44.69% de las transacciones. Esta dependencia limita la capacidad de análisis y fidelización. La conclusión estratégica más urgente es la adopción de la iniciativa cash-to-digital para capturar las transacciones no trazables y monetizar la información de consumo.
* Motor de ingresos y optimización por precio: La categoría Ropa es el principal motor de volumen del negocio (34.68% de las ventas). El insight clave es que el volumen real de ventas es impulsado por el segmento medio-alto ($203-$1200). La estrategia de inventario y stock puede centrarse en este rango de precios, priorizando marcas competitivas.
* Segmentación estratégica geográfica**:** la concentración alta de ventas, el 64.49% en solo cuatro centros comerciales, exige que el modelo operativo sea bien específico. El negocio necesita abandonar las estrategias generalizadas y adaptar la oferta al perfil sociodemográfico de cada distrito, como se sugirió con la estrategia Lujo (Híbrido) para Zorlu Center y el enfoque masivo para Forum Istanbul.

Como conclusión, este estudio permite la transición de una gestión de retail basada en la intuición a una analítica. El valor final del proyecto radica en garantizar que cada decisión de negocio, desde la gestión de inventario hasta los programas de fidelización, esté directamente alineada con el comportamiento real, medible y geolocalizado del consumidor en Estambul, maximizando así el retorno de la inversión.

## Entregables

| Nombre archivo | Descripción |
| --- | --- |
| Docu\_Evidencia3\_Innovacion.docx | Informe con descripción detallada |
| df\_analisis\_FINAL\_con\_geo.csv | DataFrame final limpio |
| Evidencia3\_con\_conexion.ipynb | Código fuente Python con Pandas |
| SQL\_create\_ventas1.sql | Esquema MySQL create y populate |
| SQL\_queries\_ventas1.sql | Queries para análisis de datos |

## Bibliografía

DataCEO. (2020). *Sales and Customer Data* [Conjunto de datos]. Kaggle. <https://www.kaggle.com/datasets/dataceo/sales-and-customer-data>

El equipo de desarrollo de pandas. (s.f.). *pandas.DataFrame.to\_sql*. Recuperado de <https://docs.kanaries.net/es/topics/Pandas/pandas-to-sql>

SQLAlchemy. (s.f.). *Using the Engine: Engine and Connection*. Recuperado de <https://docs.sqlalchemy.org/>

Oracle y/o sus afiliados. (s.f.). *MySQL 8.0 Reference Manual: 14.7 Date and Time Functions - STR\_TO\_DATE*. Recuperado de <https://www.datacamp.com/es/doc/mysql/mysql-str-to-date>

Anaconda.org. (s. f.). User guides – Anaconda.org. Recuperado de <https://www.anaconda.com/docs/tools/anaconda-org/user-guide/main>

Jupyter Documentation. (s. f.). Jupyter Documentation. Recuperado de <https://docs.jupyter.org/en/latest/>

Pandas Development Team. (s. f.). pandas documentation. Recuperado de <https://pandas.pydata.org/docs/>