Análisis de Datos

Junior Sesario Huanca Acebo

Primer Informe

1. Importanción de Librerias

Como primer paso, se importan las librerias necesarias, para utilizar las funciones que necesarias.

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

2. Lectura de Datos

A continuación, se crean variables que contengan las direcciones de los archivos.

```
DATA_PATH="/Users/Usuario/Desktop/ANÁLISIS DE
DATOS/international_orders_dataset/"

FILE_CUSTOMERS_ADDRESSES = 'customer_addresses.csv'
FILE_CUSTOMERS = 'customers.csv'
FILE_EMPLOYEES = 'employees.csv'
FILE_INVENTORY = 'inventory.csv'
FILE_ITEMS = 'order_items.csv'
FILE_ORDERS = 'orders.csv'
FILE_PAYMENTS = 'payments.csv'
FILE_CATEGORIES = 'product_categories.csv'
FILE_PRODUCTS = 'products.csv'
FILE_EVENTS = 'shipment_events.csv'
FILE_SHIPMENTS = 'shipments.csv'
FILE_SUPPLIERS = 'suppliers.csv'
FILE_WAREHOUSES = 'warehouses.csv'
```

2.1. Leemos con pandas todos los csv

Luego, procederemos a leer todos los archivos csv con la librerias de pandas.

```python
customer_addresses = pd.read_csv(
    os.path.join(DATA_PATH, FILE_CUSTOMERS_ADDRESSES),
    dtype={'address_id': int}
    )

customers = pd.read_csv(
    os.path.join(DATA_PATH, FILE_CUSTOMERS),
    dtype={'customer_id': int}
    )

employees = pd.read_csv(
    os.path.join(DATA_PATH, FILE_EMPLOYEES),
    dtype={'employee_id': int}
    )

inventory = pd.read_csv(
    os.path.join(DATA_PATH, FILE_INVENTORY ),
    dtype={'inventory_id': int}
    )

items = pd.read_csv(
    os.path.join(DATA_PATH, FILE_ITEMS),
    dtype={'order_item_id': int}
    )

orders = pd.read_csv(
    os.path.join(DATA_PATH, FILE_ORDERS),
    dtype={'order_id': int}
    )

payments = pd.read_csv(
    os.path.join(DATA_PATH, FILE_PAYMENTS),
    dtype={'payment_id': int}
    )

product_categorias = pd.read_csv(
    os.path.join(DATA_PATH, FILE_CATEGORIES),
    dtype={'category_id': int}
    )

products = pd.read_csv(
    os.path.join(DATA_PATH, FILE_PRODUCTS),
    dtype={'product_id': int}
    )

shipment_events = pd.read_csv(
    os.path.join(DATA_PATH, FILE_EVENTS),
    dtype={'shipment_event_id': int}
    )
```

```python
shipments = pd.read_csv(
    os.path.join(DATA_PATH, FILE_SHIPMENTS ),
    dtype={'shipment_id': int}
    )

suppliers = pd.read_csv(
    os.path.join(DATA_PATH, FILE_SUPPLIERS),
    dtype={'supplier_id': int}
    )

warehouses = pd.read_csv(
    os.path.join(DATA_PATH, FILE_WAREHOUSES),
    dtype={'warehouse_id': int}
    )
```

3. Exploración Inicial

Realizamos una verificación que todos los archivos se hayan leido correctamente, usando funciones como: head(), sample(), info() y describe ().

```python
customer_addresses.head()
```

```
   address_id  customer_id      type        line1        city
state_province  \
0           1            1  shipping  Calle 3006  Ciudad 384
SP-62
1           2            2  shipping   Calle 972  Ciudad 231
SP-36
2           3            2  shipping  Calle 6418  Ciudad 393
SP-20
3           4            2   billing  Calle 3090  Ciudad 489
SP-15
4           5            3   billing  Calle 9511  Ciudad 260
SP-91

   postal_code               country country_code
0        78417                 Italy           IT
1        82349                Canada           CA
2        93627  United Arab Emirates           AE
3        74915           Netherlands           NL
4        53438           Netherlands           NL
```

```python
customers.sample(10)
```

```
     customer_id     customer_name                         email  \
168          169      Elijah Popov       elijahpopov306@demo.org
186          187       Lucía Patel      lucapatel8660@company.net
158          159  Valentina Ivanov  valentinaivanov8257@mail.com
209          210          Liam Kim          liamkim6742@mail.com
112          113       Chen Flores     chenflores2671@example.com
```

```
134          135       Zoe Sanchez       zoesanchez1379@mail.com
75            76    Sofia Williams    sofiawilliams7114@mail.com
30            31      Camila Perez   camilaperez3762@example.com
170          171   Isabella Miller   isabellamiller7592@mail.com
185          186       Emily Smith    emilysmith2432@example.com

                 phone         country country_code preferred_currency
\
168  +13-861-420-7151           China           CN                CNY

186   +3-837-414-4827        Paraguay           PY                PYG

158  +96-752-803-8230           Italy           IT                EUR

209  +46-385-749-2974           Japan           JP                JPY

112  +98-527-543-3500    South Africa           ZA                ZAR

134  +88-331-425-8390    South Africa           ZA                ZAR

75   +68-825-428-7046           Spain           ES                EUR

30   +56-630-207-6984     Netherlands           NL                EUR

170  +82-582-995-4885  United Kingdom           GB                GBP

185  +16-611-551-2176         Uruguay           UY                UYU


                    created_at
168  2024-10-13 04:43:37+00:00
186  2024-02-21 01:53:12+00:00
158  2022-12-01 14:07:02+00:00
209  2024-08-11 01:48:37+00:00
112  2025-07-07 01:57:08+00:00
134  2024-08-14 07:55:54+00:00
75   2022-11-10 18:42:47+00:00
30   2024-02-06 07:23:03+00:00
170  2023-05-07 17:30:19+00:00
185  2023-03-27 10:48:57+00:00
```

```
employees.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 80 entries, 0 to 79
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   employee_id  80 non-null     int64
 1   full_name    80 non-null     object
 2   role         80 non-null     object
 3   email        80 non-null     object
 4   country      80 non-null     object
 5   hired_at     80 non-null     object
 6   active       80 non-null     int64
dtypes: int64(2), object(5)
memory usage: 4.5+ KB
```

```
inventory.describe()
```

```
       inventory_id  warehouse_id   product_id    stock_qty
reorder_point
count   2400.000000   2400.000000  2400.000000  2400.000000
2400.00000
mean    1200.500000      6.500000   247.530833   125.179167
34.40875
std      692.964646      3.452772   144.535284    57.408698
14.19004
min        1.000000      1.000000     1.000000     5.000000
5.00000
25%      600.750000      3.750000   120.750000    85.000000
25.00000
50%     1200.500000      6.500000   246.000000   123.500000
35.00000
75%     1800.250000      9.250000   371.000000   165.000000
44.00000
max     2400.000000     12.000000   500.000000   349.000000
88.00000
```

```
items.head()
```

```
   order_item_id  order_id  line_number  product_id  qty
unit_price_usd  \
0              1         1            1         316    2
14.83
1              2         1            2          75    3
37.44
2              3         1            3         440    1
33.70
3              4         1            4         236    4
21.90
4              5         1            5         159    5
45.86

   discount
0      0.00
1      0.00
2      0.00
3      0.00
4      0.07
```

```
orders.sample(10)
```

```
     order_id  customer_id                 order_date     status  \
644       645           49  2025-07-16 07:25:02+00:00  fulfilled
351       352           67  2024-11-24 10:03:14+00:00  fulfilled
480       481          207  2023-02-18 23:27:59+00:00  cancelled
204       205           58  2023-11-20 05:26:12+00:00  fulfilled
286       287           18  2023-05-10 18:09:58+00:00  fulfilled
264       265          187  2024-08-05 17:48:40+00:00  confirmed
899       900           51  2024-01-11 22:49:28+00:00  confirmed
130       131           96  2025-04-15 14:50:37+00:00  confirmed
51         52           97  2023-12-14 17:42:52+00:00  confirmed
220       221            9  2022-11-05 14:17:09+00:00  fulfilled

     sales_rep_id  shipping_address_id currency
644            80                  102      USD
351            64                  141      USD
480            11                  406      CLP
204            21                  120      USD
286            46                   40      EUR
264            31                  373      PYG
899            55                  106      ZAR
130            58                  200      AUD
51             33                  201      BOB
220            19                   23      UYU
```

```
payments.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 921 entries, 0 to 920
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   payment_id    921 non-null    int64
 1   order_id      921 non-null    int64
 2   payment_date  921 non-null    object
 3   amount_usd    921 non-null    float64
 4   method        921 non-null    object
 5   status        921 non-null    object
dtypes: float64(1), int64(2), object(3)
memory usage: 43.3+ KB
```

```
product_categorias.describe()
```

```
       category_id
count     20.00000
mean      10.50000
std        5.91608
min        1.00000
25%        5.75000
```

```
50%       10.50000
75%       15.25000
max       20.00000
```

```
products.head()
```

```
   product_id        sku   product_name  category_id  supplier_id  \
0           1  SKU-00001  Producto 0001            5           34
1           2  SKU-00002  Producto 0002           17            2
2           3  SKU-00003  Producto 0003           15           12
3           4  SKU-00004  Producto 0004           20            1
4           5  SKU-00005  Producto 0005            5           12

   unit_price_usd  weight_kg  is_active
0          193.34       0.88          0
1           53.87       3.81          1
2           49.96       0.25          1
3           34.14       2.22          1
4           74.91       1.04          1
```

```
shipment_events.sample(5)
```

```
      shipment_event_id  shipment_id                 event_time  \
1544               1545          440  2025-07-05 03:04:53+00:00
1459               1460          419  2023-09-16 18:29:06+00:00
1720               1721          487  2025-03-28 05:16:12+00:00
2445               2446          697  2024-08-07 15:43:15+00:00
1654               1655          470  2025-05-04 12:55:56+00:00

         event_type     location
1544     in_transit         Peru
1459  label_created        India
1720  label_created    Australia
2445  label_created     Paraguay
1654     in_transit  Netherlands
```

```
shipments.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 741 entries, 0 to 740
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   shipment_id          741 non-null    int64  
 1   order_id             741 non-null    int64  
 2   ship_date            741 non-null    object 
 3   carrier              741 non-null    object 
 4   tracking_number      741 non-null    object 
 5   shipping_cost_usd    741 non-null    float64
 6   origin_warehouse_id  741 non-null    int64  
```

```
dtypes: float64(1), int64(3), object(3)
memory usage: 40.7+ KB

suppliers.describe()

       supplier_id
count    40.000000
mean     20.500000
std      11.690452
min       1.000000
25%      10.750000
50%      20.500000
75%      30.250000
max      40.000000

warehouses.head()

   warehouse_id warehouse_name   country country_code currency
0             1       WH-BR-01    Brazil           BR      BRL
1             2       WH-PY-02  Paraguay           PY      PYG
2             3       WH-JP-03     Japan           JP      JPY
3             4       WH-CA-04    Canada           CA      CAD
4             5       WH-PE-05      Peru           PE      PEN
```

4. Conversión de Fechas

Previamente se analizo todas las tablas y buscamos columnas que contengan fechas para luego convertirlo a formato fecha para hacer análisis de tiempo.

```
## convierte order_purchase_timestamp

orders['order_date'] = pd.to_datetime(orders['order_date'],
errors='coerce' )
customers['created_at']  = pd.to_datetime(customers['created_at'],
errors='coerce' )
employees['hired_at'] = pd.to_datetime(employees['hired_at'],
errors='coerce' )
inventory['last_restocked_at'] =
pd.to_datetime(inventory['last_restocked_at'], errors='coerce' )
payments['payment_date'] = pd.to_datetime(payments['payment_date'],
errors='coerce' )
shipment_events['event_time'] =
pd.to_datetime(shipment_events['event_time'], errors='coerce' )
shipments[ 'ship_date'] = pd.to_datetime(shipments[ 'ship_date'],
errors='coerce' )
```

5. Agrupación y Agregación

Realizamos agrupaciones y agregaciones necesarias para el buen análisis.

```
## agrupamos por cada orden, cuantos productos se vendieron y el total
de los productos

items_agg = items.groupby(
     ['order_id']).agg(
     {'order_item_id': 'count',
    'unit_price_usd': 'sum'}
     ).reset_index()

## Se agrupa aqui, para que veamos el total de lo que se pago por cada
orden

payments_agg = payments.groupby('order_id').agg({
    'amount_usd':'sum'
}).reset_index()

 items_agg.head()

   order_id  order_item_id  unit_price_usd
0         1              5          153.73
1         2              6          392.07
2         3              1          153.86
3         4              1            8.44
4         5              3          254.69

payments_agg.head()

   order_id  amount_usd
0         2      803.68
1         3      769.30
2         4       42.20
3         5      820.58
4         6     1133.76
```

6. Limpieza de Datos

6.1. Filtrar filas: df.query("columna == 'valor'")

Filtraremos algunos datos, con la función del .query().

```
customer_addresses.query("postal_code == 78417")

   address_id  customer_id      type       line1        city
state_province  \
0           1            1  shipping  Calle 3006  Ciudad 384
SP-62

   postal_code country country_code
0        78417   Italy           IT
```

```
customers.query("customer_id == 7")

   customer_id customer_name                     email
phone  \
6            7     Mei Novak  meinovak6398@company.net  +94-951-350-
6943

  country country_code preferred_currency                 created_at
6  Poland           PL                PLN 2023-05-03 06:58:52+00:00

orders.query("order_id == 2")

   order_id  customer_id                order_date     status
sales_rep_id  \
1         2          179 2024-03-22 10:38:19+00:00  confirmed
33

   shipping_address_id currency
1                  355      EUR

items.query("order_item_id == 10")

   order_item_id  order_id  line_number  product_id  qty
unit_price_usd  \
9             10         2            5         157    2
116.76

   discount
9      0.02

products.query("product_id == 10")

   product_id        sku   product_name  category_id  supplier_id  \
9          10  SKU-00010  Producto 0010           17           36

   unit_price_usd  weight_kg  is_active
9           15.86       1.12          1

product_categorias.query("category_id == 10")

   category_id category_name
9           10        Jardín
```

6.2. Renombrar columnas: df.rename(columns={'viejo':'nuevo'}, inplace=True)

Comenzamos a renombrar algunas columnas de algunas tablas, para un mejor entendimiento.

```
items_agg.head()
```

```
   order_id  order_item_id  unit_price_usd
0         1              5          153.73
1         2              6          392.07
2         3              1          153.86
3         4              1            8.44
4         5              3          254.69
```

```
items_agg.rename(
    columns={'order_item_id': 'total_products', 'unit_price_usd':
'total_sales'},
    inplace=True
    )

items_agg.head()
```

```
   order_id  total_products  total_sales
0         1               5       153.73
1         2               6       392.07
2         3               1       153.86
3         4               1         8.44
4         5               3       254.69
```

6.3. Eliminar duplicados: df.drop_duplicates()

Eliminamos algunos datos que estan duplicados en algunas tablas.

```
## Solamente tomamos una dirección de cada cliente

unique_customer_addresse = customer_addresses.drop_duplicates(
    subset = ['customer_id']
    )
```

## 7. Uniones de Tablas

Realizamos la uniones de varias tablas con una relazión especifico entre ambas, esto se hace , para que el análisis se muestre en una sola plana.

7.1. Clientes + direcciones

```
customers_geo = pd.merge(
    customers,
    unique_customer_addresse,
    on="customer_id",
    how="left"
)

customers_geo.head(1)
```

```
   customer_id customer_name                    email
phone  \
0            1   Jacob Novak  jacobnovak7841@demo.org  +75-895-315-
7153

  country_x country_code_x preferred_currency
created_at  \
0  Colombia             CO                COP 2023-07-28
22:32:38+00:00

   address_id      type       line1        city state_province
postal_code  \
0           1  shipping  Calle 3006  Ciudad 384          SP-62
78417

  country_y country_code_y
0     Italy             IT
```

7.2. Clientes + direcciones + pedidos

```
customers_orders = pd.merge(
    customers_geo,
    orders,
    on="customer_id",
    how="left"
)

customers_orders.head(1)

   customer_id customer_name                    email
phone  \
0            1   Jacob Novak  jacobnovak7841@demo.org  +75-895-315-
7153

  country_x country_code_x preferred_currency
created_at  \
0  Colombia             CO                COP 2023-07-28
22:32:38+00:00

   address_id      type  ... state_province postal_code country_y  \
0           1  shipping  ...          SP-62       78417     Italy

   country_code_y order_id                order_date     status
sales_rep_id  \
0              IT     50.0 2023-07-04 10:37:53+00:00  confirmed
8.0

  shipping_address_id  currency
0                 1.0       COP

[1 rows x 22 columns]
```

7.3. Pedidos + items (ventas)

```
orders_items = pd.merge(
    customers_orders,
    ## Aqui quise agregarle lo que agrupe en items_agg, pero como mas
abajo necesito el id del producto, no fue posible
    ## Asi que se puso items, con todas sus columnas, como esta en el
csv

    items,
    on='order_id',
    how='left'
)

orders_items.head(1)
```

```
   customer_id customer_name                    email
phone  \
0            1   Jacob Novak  jacobnovak7841@demo.org  +75-895-315-
7153

  country_x country_code_x preferred_currency
created_at  \
0  Colombia             CO                COP 2023-07-28
22:32:38+00:00

   address_id      type  ...     status sales_rep_id
shipping_address_id  \
0           1  shipping  ...  confirmed          8.0
1.0

   currency order_item_id line_number  product_id  qty unit_price_usd
\
0       COP         182.0         1.0        70.0  4.0          16.86


   discount
0       0.0

[1 rows x 28 columns]
```

7.4. Agregar detalle de productos

```
orders_items_products = pd.merge(
    orders_items,
    products,
    on="product_id",
    how="left"
)

orders_items_products.head(1)
```

```
   customer_id customer_name                   email
phone  \
0            1   Jacob Novak  jacobnovak7841@demo.org  +75-895-315-
7153

  country_x country_code_x preferred_currency
created_at  \
0  Colombia             CO                COP 2023-07-28
22:32:38+00:00

   address_id      type  ...  qty unit_price_usd_x discount        sku
\
0           1  shipping  ...  4.0            16.86      0.0  SKU-00070


    product_name category_id  supplier_id unit_price_usd_y
weight_kg  \
0  Producto 0070         1.0         22.0            16.86      2.09


   is_active
0        1.0

[1 rows x 35 columns]
```

7.5. Unir con pagos

```
df_final = pd.merge(
    orders_items_products,
    payments,
    on="order_id",
    how="left"
)

df_final.head(1)
```

```
   customer_id customer_name                   email
phone  \
0            1   Jacob Novak  jacobnovak7841@demo.org  +75-895-315-
7153

  customer_country customer_country_code preferred_currency  \
0         Colombia                    CO                COP

                 created_at  address_id      type  ... category_id  \
0 2023-07-28 22:32:38+00:00           1  shipping  ...         1.0

  supplier_id unit_price_product  weight_kg is_active payment_id  \
0        22.0              16.86       2.09       1.0       42.0

               payment_date amount_usd method  payment_status
```

```
0 2023-10-31 12:22:47+00:00     877.29   card        captured

[1 rows x 40 columns]
```

7.6. Renombrar las columnas

Se está renombrando las columnas que tienen el mismo nombre, pero los diferencia un 'X' o un 'Y' y se está colocando algo especifico, para saber que significa esa columna

```
df_final.rename(
    columns={
        'status_x': 'order_status',
        'status_y': 'payment_status',
        'unit_price_usd_x': 'unit_price_order',
        'unit_price_usd_y': 'unit_price_product',
        'country_x': 'customer_country',
        'country_y': 'address_country',
        'country_code_x': 'customer_country_code',
        'country_code_y': 'address_country_code'
    },
    inplace=True
)

df_final.head(1)
```

```
   customer_id customer_name                    email
phone  \
0            1   Jacob Novak  jacobnovak7841@demo.org  +75-895-315-
7153

  customer_country customer_country_code preferred_currency  \
0         Colombia                    CO                COP

                 created_at  address_id      type  ... category_id  \
0 2023-07-28 22:32:38+00:00           1  shipping  ...         1.0

  supplier_id unit_price_product  weight_kg is_active payment_id  \
0        22.0              16.86       2.09       1.0       42.0

               payment_date amount_usd method  payment_status
0 2023-10-31 12:22:47+00:00     877.29   card        captured

[1 rows x 40 columns]
```

8. Estadísticas y Visualización

Realizamos algunas estadísticas, ya sea de manera gráfica o no.

8.1. Estados de los pedidos (value_counts)

```
## Cantidad de pedidos por estado (Completed, Cancelled, Pending,
etc.)

df_final['order_status'].value_counts()
```

```
order_status
confirmed    1839
fulfilled    1481
pending       380
cancelled     258
returned      131
Name: count, dtype: int64
```

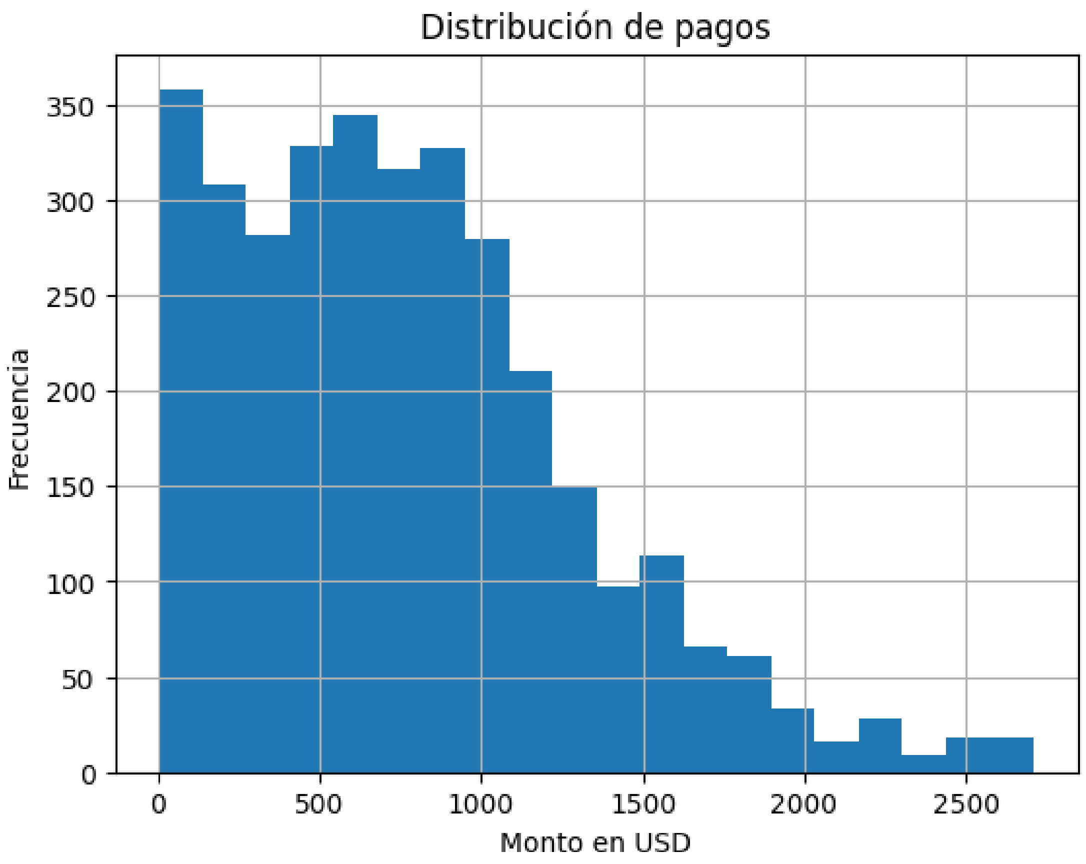
8.2. Cantidad de pedidos por mes (ejemplo mensual)

```
## Agrupar por mes y contar pedidos

pedidos_por_mes =
df_final.groupby(df_final['order_date'].dt.month).size()
print(pedidos_por_mes)
```

```
order_date
1.0     307
2.0     345
3.0     268
4.0     318
5.0     311
6.0     416
7.0     337
8.0     298
9.0     330
10.0    405
11.0    382
12.0    372
dtype: int64
```

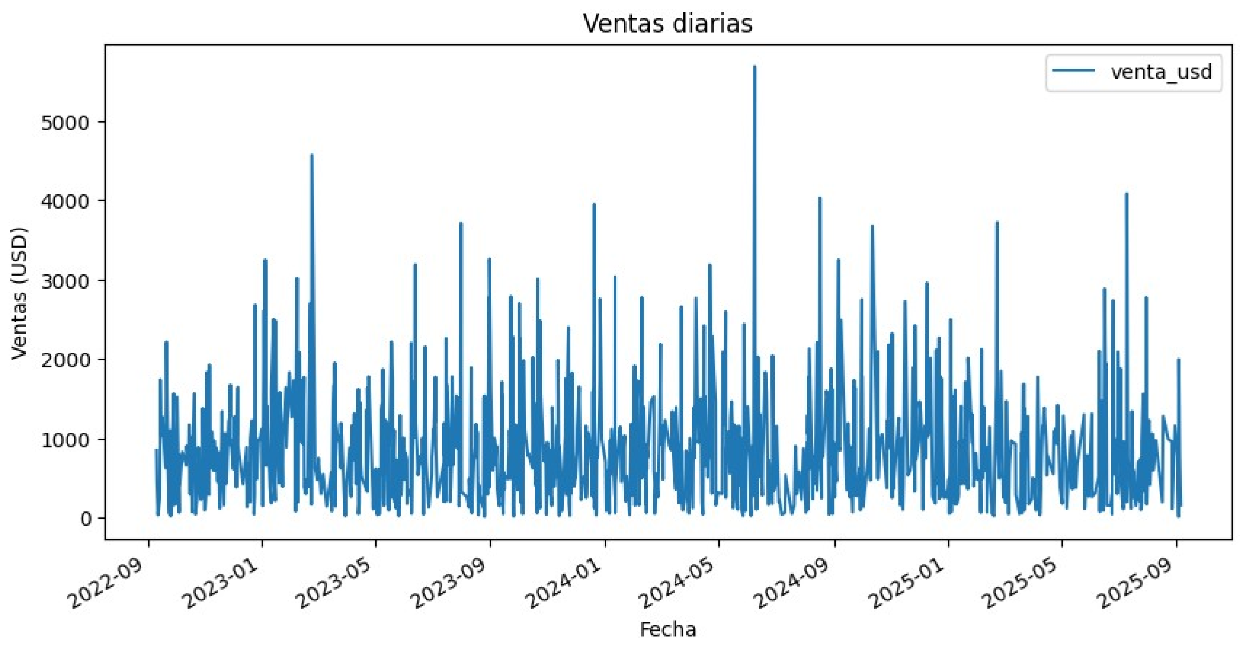
8.3. Histograma

```
## Distribución de montos de pago

df_final['amount_usd'].hist(bins=20)
plt.title("Distribución de pagos")
plt.xlabel("Monto en USD")
plt.ylabel("Frecuencia")
plt.show()
```



8.4. Ventas totales por fecha de pedido (línea de ventas)

```
## Crear columna de ventas usando el precio del pedido
df_final['venta_usd'] = (df_final['qty'] *
df_final['unit_price_order']) - df_final['discount']

## Agrupar por fecha y sumar ventas
ventas_diarias = df_final.groupby('order_date')
['venta_usd'].sum().reset_index()

## Graficar
ventas_diarias.plot(x='order_date', y='venta_usd', kind='line',
figsize=(10,5))
plt.title("Ventas diarias")
plt.xlabel("Fecha")
plt.ylabel("Ventas (USD)")
plt.show()
```



9. Archivo entregable

Finalmente, todo lo que se realizó anteriormente, lo guardamos en un nuevo .csv y eso seria todo.

```
df_final.to_csv(
    'results.csv',
    index=False
    )
```