Portafolio - Proyecto 1

April 20, 2025

1 Customer Purchase Behavior Analysis – Instacart Dataset

Victor Uriel Leyva
Analista de Datos Jr.

1.0.1 Objetivo del Proyecto:

Analizar los patrones de compra de clientes en una tienda en línea (Instacart) mediante técnicas de análisis exploratorio de datos y limpieza de datos, con el fin de identificar hábitos de consumo, productos más populares y comportamientos de recompra.

1.0.2 Herramientas utilizadas:

- Python
- Pandas
- Matplotlib
- Seaborn
- Jupyter Notebook

1.0.3 Principales habilidades aplicadas:

- Limpieza y preprocesamiento de datos
- Manejo de valores ausentes y duplicados
- Análisis de variables categóricas y numéricas
- Visualización de datos
- Detección de patrones de comportamiento de clientes

2 Introducción

Instacart es una plataforma de entregas de comestibles donde la clientela puede registrar un pedido y hacer que se lo entreguen, similar a Uber Eats y Door Dash. El conjunto de datos que se ha proporcionado tiene modificaciones del original. Se redujo el tamaño del conjunto para que los cálculos se hicieran más rápido y se introdujeron valores ausentes y duplicados. Se tuvo cuidado de conservar las distribuciones de los datos originales cuando se hicieron los cambios.

Se deben completar tres pasos. Para cada uno de ellos, se escribirá una breve introducción que refleje con claridad cómo se pretende resolver cada paso, y se escribirán párrafos explicatorios que justifiquen las decisiones al tiempo que se avanza en la solución. También se redactará una conclusión que resuma los hallazgos y elecciones.

2.1 Diccionario de datos

Hay cinco tablas en el conjunto de datos, y se tendran que usar todas para hacer el preprocesamiento de datos y el análisis exploratorio de datos. A continuación se muestra un diccionario de datos que enumera las columnas de cada tabla y describe los datos que contienen.

- `instacart_orders.csv`: cada fila corresponde a un pedido en la aplicación Instacart.
  - `'order_id'`: número de ID que identifica de manera única cada pedido.
  - `'user_id'`: número de ID que identifica de manera única la cuenta de cada cliente.
  - `'order_number'`: el número de veces que este cliente ha hecho un pedido.
  - `'order_dow'`: día de la semana en que se hizo el pedido (0 si es domingo).
  - `'order_hour_of_day'`: hora del día en que se hizo el pedido.
  - `'days_since_prior_order'`: número de días transcurridos desde que este cliente hizo su pedido anterior.
- `products.csv`: cada fila corresponde a un producto único que pueden comprar los clientes.
  - `'product_id'`: número ID que identifica de manera única cada producto.
  - `'product_name'`: nombre del producto.
  - `'aisle_id'`: número ID que identifica de manera única cada categoría de pasillo de víveres.
  - `'department_id'`: número ID que identifica de manera única cada departamento de víveres.
- `order_products.csv`: cada fila corresponde a un artículo pedido en un pedido.
  - `'order_id'`: número de ID que identifica de manera única cada pedido.
  - `'product_id'`: número ID que identifica de manera única cada producto.
  - `'add_to_cart_order'`: el orden secuencial en el que se añadió cada artículo en el carrito.
  - `'reordered'`: 0 si el cliente nunca ha pedido este producto antes, 1 si lo ha pedido.
- `aisles.csv`
  - `'aisle_id'`: número ID que identifica de manera única cada categoría de pasillo de víveres.
  - `'aisle'`: nombre del pasillo.
- `departments.csv`
  - `'department_id'`: número ID que identifica de manera única cada departamento de víveres.
  - `'department'`: nombre del departamento.

3 Paso 1. Descripción de los datos

Leeré los archivos de datos (`/datasets/instacart_orders.csv`, `/datasets/products.csv`, `/datasets/aisles.csv`, `/datasets/departments.csv` y `/datasets/order_products.csv`) con `pd.read_csv()` usando los parámetros adecuados para leer los datos correctamente. Verificando la información para cada DataFrame creado.

3.1 Plan de solución

En nuestro plan de solución para la descripción de los datos, comenzaremos examinando brevemente los datos, proseguiremos importando las librerias que vamos a necesitar en nuestro código, para después utilizar las funciones con los parametros adecuados para leer nuestros datos correctamente, finalmente verificaremos la información de nuestros datos, asegurandonos que los importamos de manera correcta.

[1]:
```
# importar librerías
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

[2]:
```
# leer conjuntos de datos en los DataFrames
df_instacart_orders = pd.read_csv('/datasets/instacart_orders.csv', sep=';')
df_products = pd.read_csv('/datasets/products.csv', sep=';')
df_order_products = pd.read_csv('/datasets/order_products.csv', sep=';')
df_aisles = pd.read_csv('/datasets/aisles.csv', sep=';')
df_departments = pd.read_csv('/datasets/departments.csv', sep=';')
```

[3]:
```
# mostrar información del DataFrame instacart_orders
print(df_instacart_orders.info(show_counts=True))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 478967 entries, 0 to 478966
Data columns (total 6 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   order_id                478967 non-null  int64  
 1   user_id                 478967 non-null  int64  
 2   order_number            478967 non-null  int64  
 3   order_dow               478967 non-null  int64  
 4   order_hour_of_day       478967 non-null  int64  
 5   days_since_prior_order  450148 non-null  float64
dtypes: float64(1), int64(5)
memory usage: 21.9 MB
None
```

[4]:
```
# mostrar información del DataFrame products
print(df_products.info(show_counts=True))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49694 entries, 0 to 49693
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   product_id     49694 non-null  int64 
 1   product_name   48436 non-null  object
 2   aisle_id       49694 non-null  int64 
```

```
 3   department_id  49694 non-null  int64
dtypes: int64(3), object(1)
memory usage: 1.5+ MB
None
```

[5]:
```
# mostrar información del DataFrame order_products
print(df_order_products.info(show_counts=True))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4545007 entries, 0 to 4545006
Data columns (total 4 columns):
 #   Column             Non-Null Count    Dtype
---  ------             --------------    -----
 0   order_id           4545007 non-null  int64
 1   product_id         4545007 non-null  int64
 2   add_to_cart_order  4544171 non-null  float64
 3   reordered          4545007 non-null  int64
dtypes: float64(1), int64(3)
memory usage: 138.7 MB
None
```

[6]:
```
# mostrar información del DataFrame aisles
print(df_aisles.info(show_counts=True))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134 entries, 0 to 133
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   aisle_id  134 non-null    int64
 1   aisle     134 non-null    object
dtypes: int64(1), object(1)
memory usage: 2.2+ KB
None
```

[7]:
```
# mostrar información del DataFrame departments
print(df_departments.info(show_counts=True))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   department_id  21 non-null     int64
 1   department     21 non-null     object
dtypes: int64(1), object(1)
memory usage: 464.0+ bytes
None
```

3.2 Conclusiones

Podemos comprobar que los DataFrames pueden utilizarse para su análisis; existen DataFrames con una gran cantidad de datos, mientras que otros tienen muy pocas filas. También contamos con la existencia de valores nulos en los primeros tres dataframes. Tomando en cuenta la información de las columnas que se nos proporcionó al inicio, estamos listos para el preprocesamiento de nuestros datos.

4 Paso 2. Preprocesamiento de los datos

Se procesarán los datos de la siguiente manera:

- Verificar y corrigir los tipos de datos (por ejemplo, asegúrate de que las columnas de ID sean números enteros).
- Identificar y completar los valores ausentes.
- Identificar y eliminar los valores duplicados.

Asegurando de explicar qué tipos de valores ausentes y duplicados se encontraron, cómo se completaron o eliminaron y por qué se usaronesos métodos. ¿Por qué crees que estos valores ausentes y duplicados pueden haber estado presentes en el conjunto de datos?

4.1 Plan de solución

El plan de solución para el preprociesamiento de nuestros datos inciará inspeccionando que tipo de columnas (variables) estamos manejando, para así determinar el mejor tipo de datos para nuestras columnas. Después haremos una limpieza de los datos identificando y corrigiendo los valores ausentes, así como los valores duplicados, para finalmente hacer nuestro análisis.

4.2 Encuentra y elimina los valores duplicados (y describe cómo tomaste tus decisiones).

4.2.1 `instacart_orders` data frame

[8]:
```
# Revisa si hay pedidos duplicados

print(df_instacart_orders.duplicated().sum())
print(df_instacart_orders[df_instacart_orders.duplicated()])
```

```
15
        order_id  user_id  order_number  order_dow  order_hour_of_day  \
145574    794638    50898            24          3                  2
223105   2160484   107525            16          3                  2
230807   1918001   188546            14          3                  2
266232   1782114   106752             1          3                  2
273805   1112182   202304            84          3                  2
284038   2845099    31189            11          3                  2
311713   1021560    53767             3          3                  2
321100    408114    68324             4          3                  2
323900   1919531   191501            32          3                  2
345917   2232988    82565             1          3                  2
```

```
371905    391768    57671            19          3                  2
394347    467134    63189            21          3                  2
411408   1286742   183220            48          3                  2
415163   2282673    86751            49          3                  2
441599   2125197    14050            48          3                  2

        days_since_prior_order
145574                     2.0
223105                    30.0
230807                    16.0
266232                     NaN
273805                     6.0
284038                     7.0
311713                     9.0
321100                    18.0
323900                     7.0
345917                     NaN
371905                    10.0
394347                     2.0
411408                     4.0
415163                     2.0
441599                     3.0
```

Se cuenta con líneas duplicadas, lo que tienen en común es el dia de la semana y la hora en la que se realizarón 15 pedidos.

[9]:
```
# Basándote en tus hallazgos,
# Verifica todos los pedidos que se hicieron el miércoles a las 2:00 a.m.
print(df_instacart_orders.query("order_dow == 3 and order_hour_of_day == 2"))
```

```
        order_id  user_id  order_number  order_dow  order_hour_of_day  \
4838     2766110   162084            41          3                  2
5156     2190225   138285            18          3                  2
15506     553049    58599            13          3                  2
18420     382357   120200            19          3                  2
24691     690242    77357             2          3                  2
…            …        …             …          …                  …
457013   3384021    14881             6          3                  2
458816    910166   164782            18          3                  2
459635   1680532   106435             6          3                  2
468324    222962    54979            59          3                  2
477526   2592344    46860            38          3                  2

        days_since_prior_order
4838                      16.0
5156                      11.0
15506                      7.0
18420                     11.0
24691                      9.0
```

```
…                       …
457013                  30.0
458816                   4.0
459635                  21.0
468324                   3.0
477526                   3.0

[121 rows x 6 columns]
```

A pesar de compartir el día de la semana y la hora del día, son pedidos distintos (tienen un número de ID diferente).

[10]:
```
# Elimina los pedidos duplicados
df_instacart_orders = df_instacart_orders.drop_duplicates()
```

[11]:
```
# Vuelve a verificar si hay filas duplicadas
print(df_instacart_orders.duplicated().sum())
```

```
0
```

[12]:
```
# Vuelve a verificar si hay IDs duplicados de pedidos
print(df_instacart_orders['order_id'].duplicated().sum())
```

```
0
```

En un principio se utilizo el método `duplicated()` para identificar valores duplicados, se logró identificar 15 valores duplicados, los cuales tenían el mismo día de la semana (miercoles), y la misma hora del dia (02:00). Pudimos observarlos filtrando los valores que cumplen ciertas condiciones en nuestro DataFrame, utilizando la función `query()`. Para después observar que los duplicados comparten el mismo horario de pedido más no el mismo ID de orden. Finalmente eliminamos los valores duplicados utilizando el método `drop_duplicates()` para así poder comprobar que no contamos con pedidos duplicados en nuestro DataFrame.

4.2.2 products data frame

[13]:
```
# Verifica si hay filas totalmente duplicadas
print(df_products.duplicated().sum())
```

```
0
```

[14]:
```
# Verifica si hay IDs duplicadas de productos
print(df_products['product_id'].duplicated().sum())
```

```
0
```

[15]:
```
# Revisa si hay nombres duplicados de productos (convierte los nombres a letras␣
  ↪mayúsculas para compararlos mejor)
print(df_products['product_name'].str.upper().duplicated().sum())
```

```
1361
```

[16]:
```
# Revisa si hay nombres duplicados de productos no faltantes
print(df_products.product_name.isna().sum())
```

```
1258
```

En un principio se verificó la existencia de filas duplicadas, nuevamente se utilizó el método `duplicated()` junto con `sum()`), para después encontrar que no existen filas ni IDs de productos duplicados. Finalmente comprobamos la existencia de nombres duplicados de productos utilizando el mismo método que usamos anteriormente pero esta vez cambiamos los nombres a mayusculas con el método `upper()` para compararlos de mejor manera; se identificó la presencía de 1361 nombres duplicados. Para comprobar que dichos dupilcados fuesen valores ausentes utilizamos los métodos `isna()` junto con `sum()`, para encontrar que 1258 nombres de productos están ausentes.

4.2.3 `departments` data frame

[17]:
```
# Revisa si hay filas totalmente duplicadas
print(df_departments.duplicated().sum())
```

```
0
```

[18]:
```
# Revisa si hay IDs duplicadas de productos
print(df_departments.department_id.duplicated().sum())
```

```
0
```

Se inspeccionó el DataFrame de `departmens`, y no se encontraron filas duplicadas; así como IDs de productos duplicados. Se utilizó el método `duplicated()`junto con `sum()`para ello.

4.2.4 `aisles` data frame

[19]:
```
# Revisa si hay filas totalmente duplicadas
print(df_aisles.duplicated().sum())
```

```
0
```

[20]:
```
# Revisa si hay IDs duplicadas de productos
print(df_aisles.aisle_id.duplicated().sum())
```

```
0
```

Se inspeccionó el DataFrame de `aisles`, y no se encontraron filas duplicadas; así como IDs duplicados de productos. Se utilizó el método `duplicated()`junto con `sum()`para ello.

4.2.5 `order_products` data frame

[21]:
```
# Revisa si hay filas totalmente duplicadas
print(df_order_products.duplicated().sum())
```

```
0
```

[22]:
```
# Vuelve a verificar si hay cualquier otro duplicado engañoso
print(df_order_products['order_id'].value_counts())
print(df_order_products.loc[df_order_products.loc[:,'order_id'] == 61355])
```

```
61355      127
3308010    115
2136777    108
171934     104
1959075     98
          ...
54659        1
210431       1
1717983      1
1136563      1
8196         1
Name: order_id, Length: 450046, dtype: int64
         order_id  product_id  add_to_cart_order  reordered
57981       61355        5322                NaN          0
149404      61355       39475               45.0          0
247893      61355        8594                NaN          0
269840      61355       14233                5.0          0
345399      61355       24010               51.0          0
...           ...         ...                ...        ...
4426514     61355       29270               15.0          0
4482168     61355       24964               33.0          0
4515077     61355        6760               27.0          0
4521208     61355       16262               62.0          0
4525165     61355       24561               54.0          0

[127 rows x 4 columns]
```

En un principio no encontramos filas completamente duplicadas, pero después logramos visualizar mediante el método `value_counts()` que contamos con IDs de ordenes que se repiten en multiples ocaciones; visualizamos una muestra de el ID que más se duplicaba y encontramos que algunas celdas de la columna `add_to_cart_order` cuentan con valores ausentes, o como no se han pedido anteriormente ciertos productos la columna de `reordered` se mantiene con el valor 0, quizás por ello los cuenta como valores duplicados.

4.3 Encuentra y elimina los valores ausentes

Al trabajar con valores duplicados, pudimos observar que también nos falta investigar valores ausentes:

- La columna `'product_name'` de la tabla products.
- La columna `'days_since_prior_order'` de la tabla instacart_orders.
- La columna `'add_to_cart_order'` de la tabla order_products.

4.3.1 products data frame

[23]:
```
# Encuentra los valores ausentes en la columna 'product_name'
print(df_products['product_name'].isna().sum())
print(df_products[df_products['product_name'].isna()])
```

```
1258
       product_id product_name  aisle_id  department_id
37             38          NaN       100             21
71             72          NaN       100             21
109           110          NaN       100             21
296           297          NaN       100             21
416           417          NaN       100             21
…             …            …         …               …
49552       49553          NaN       100             21
49574       49575          NaN       100             21
49640       49641          NaN       100             21
49663       49664          NaN       100             21
49668       49669          NaN       100             21

[1258 rows x 4 columns]
```

Utilizando el método `isna()` junto con `sum()` logramos identificar la presencia de 1258 nombres de productos ausentes, dichos productos comparten el mismo pasillo y el mismo departamento.

[24]:
```
#  ¿Todos los nombres de productos ausentes están relacionados con el pasillo␣
 ↪con ID 100?
print(df_products[df_products['product_name'].isna()]['aisle_id'].unique())
```

```
[100]
```

Se identificó que los valores ausentes de nombres de productos están relacionados al mismo pasillo, el cual contiene el ID número cien.

[25]:
```
# ¿Todos los nombres de productos ausentes están relacionados con el␣
 ↪departamento con ID 21?
print(df_products[df_products['product_name'].isna()]['department_id'].unique())
```

```
[21]
```

Se identificó que los nombres ausentes de productos están relacionados al mismo departamento, el cual contiene el ID número 21.

[26]:
```
# Usa las tablas department y aisle para revisar los datos del pasillo con ID␣
 ↪100 y el departamento con ID 21.
print(df_aisles[df_aisles['aisle_id'] == 100])
print(df_departments[df_departments['department_id'] == 21])
```

```
    aisle_id    aisle
99       100  missing
```

```
    department_id department
20             21    missing
```

En un principio se comprobó la existencia valores ausentes o nulos en la columna `'product_name'`, utilizando los métodos `isna()` y `sum()`. Después, fué curioso que los nombres de productos ausentes, estaban relacionados a un pasillo y un departamento en especificio, el 100 y el 21, respectivamente. Comprobamos que fuese así usando los métodos `isna()` y `unique()`. Finalmente filtramos nuestros DataFrames de `aisles` y `departments` para comprobar que el departamento 21 y el pasillo 100 no existen.

[27]:
```
# Completa los nombres de productos ausentes con 'Unknown'
df_products['product_name'] = df_products['product_name'].fillna('Unknown')
#Se comprobará si continua la presencia de valores ausentes en el nombre de los␣
 ↪productos
print(df_products['product_name'].isna().sum())
```

```
0
```

En el DataFrame `products`, en la columna `'product_name'` notamos que contabamos con valores ausentes en la información proporcionada en nuestro método `info()`. Es por ello que lo comprobamos aplicando el método `isna()` y `sum()`. Para después notar que dichos valores ausentes estabán relacionados a un pasillo y un departamento. Finalmente se completaron los valores ausentes con la palabra `'Unknown'`, para corroborar que ya no contamos con nombres de productos duplicados.

4.3.2 orders data frame

[28]:
```
# Encuentra los valores ausentes
print(df_instacart_orders.isna().sum())
```

```
order_id                      0
user_id                       0
order_number                  0
order_dow                     0
order_hour_of_day             0
days_since_prior_order    28817
dtype: int64
```

[29]:
```
# ¿Hay algún valor ausente que no sea el primer pedido del cliente?
print(df_instacart_orders[df_instacart_orders['days_since_prior_order'].isna()])
#Se cambiará el valor ausente por el número 0
df_instacart_orders['days_since_prior_order'] =␣
 ↪df_instacart_orders['days_since_prior_order'].fillna(0)
```

```
        order_id  user_id  order_number  order_dow  order_hour_of_day  \
28        133707   182261             1          3                 10
96        787445    25685             1          6                 18
100       294410   111449             1          0                 19
103      2869915   123958             1          4                 16
104      2521921    42286             1          3                 18
…            …        …             …          …                  …
```

```
478895   2589657   205028            1          0                16
478896   2222353   141211            1          2                13
478922   2272807   204154            1          1                15
478926   2499542    68810            1          4                19
478945   1387033    22496            1          5                14

        days_since_prior_order
28                         NaN
96                         NaN
100                        NaN
103                        NaN
104                        NaN
…                          …
478895                     NaN
478896                     NaN
478922                     NaN
478926                     NaN
478945                     NaN

[28817 rows x 6 columns]
```

En el DataFrame `instacart_orders`, notamos que contabamos con valores ausentes en la columna `days_since_prior_order`, utilizando el método `isna()` y `sum()` a todo nuestro DataFrame. Después nos dimos cuenta que todos los valores ausentes de esta columna comparten el mismo número de pedido del cliente, el primero; y esto tiene todo el sentido del mundo, ya que nuestra columna `days_since_prior_order` no puede registrar número de días transcurridos desde que este cliente hizo su pedido anterior ya que está realizando su primer pedido. Es por esto que considero que no deberíamos eliminar estas filas ya que son datos valiosos, en su lugar remplazamos los NaN por el valor 0.

4.3.3 `order_products` data frame

[30]:
```
# Encuentra los valores ausentes
print(df_order_products.isna().sum())
```

```
order_id               0
product_id             0
add_to_cart_order    836
reordered              0
dtype: int64
```

[31]:
```
# ¿Cuáles son los valores mínimos y máximos en esta columna?
print(df_order_products['add_to_cart_order'].min())
print(df_order_products['add_to_cart_order'].max())
```

```
1.0
64.0
```

Notamos que tenemos valores ausentes en la colúmna `add_to_cart_orders`, utilizando los métodos `isna()`y `sum()` en todo nuestro DataFrame `order_products`. Posteriormente encontramos los val-

ores mínimos y máximos de dicha columna, haciendo uso de los métodos `min()` y `max()`, dándonos a entender que el valor mínimo en el carrito de un pedido es de 1 y el valor máximo es de 64 artículos.

[32]:
```
# Guarda todas las IDs de pedidos que tengan un valor ausente en␣
 ↪'add_to_cart_order'
df_order_products_nan =␣
 ↪df_order_products[df_order_products['add_to_cart_order'].isna()]['order_id']
```

[33]:
```
# ¿Todos los pedidos con valores ausentes tienen más de 64 productos?
# Agrupa todos los pedidos con datos ausentes por su ID de pedido.
# Cuenta el número de 'product_id' en cada pedido y revisa el valor mínimo del␣
 ↪conteo.
print(df_order_products[df_order_products['order_id'].
 ↪isin(df_order_products_nan)].groupby('order_id')['product_id'].count().min())
```

```
65
```

Tras notar que el valor máximo en `add_to_cart_orders` es de 64 artículos, pudimos suponer que quizás alguien pudo agregar más de esa cantidad y no se registro el dato. Para ello creamos un dataframe con solo los IDs de pedidos (`order_id`) que contaran con valores ausentes en `add_to_cart_orders`. Después utilizamos este DataFrame filtrado para obtener el valor mínimo del ID del producto en nuestro DataFrame que contiene solo IDs de valores ausentes. Arrojandonos que el valor mínimo es el 65, comprobando que los pedidos con un mayor número de 64 artículos, tendrían valores ausentes en la columna `add_to_cart_orders`.

[34]:
```
# Remplaza los valores ausentes en la columna 'add_to_cart_orders' con 999 y␣
 ↪convierte la columna al tipo entero.
df_order_products['add_to_cart_order'] = df_order_products['add_to_cart_order'].
 ↪fillna(999).astype(int)
```

Tras notar el error en los datos se decidió por rellenar los valores ausentes con la cifra 999, haciendo referencia a que se añadió una cantidad mayor de 64 artículos. También se convirtió el tipo de datos a entero (`int`) ya que era decimal y para el contexto de la variable no tendremos datos flotantes (decimales).

4.4 Conclusiones

Este preprocesamiento de los datos comenzó identificando valores duplicados, para así poder corregirlos, después se comprobó si existen valores ausentes y se gestionarón conforme la raíz del problema, la mayoría de las ocaciones se remplazaó el valor ausente por una cifra o una cadena. Sin duda alguna fue una tarea difícil, creo que este arte de preprocesar los datos es más que simples codigos de limpieza, hay que entender el contexto de la actividad para así poder responder a ¿por qué tenemos estos valores ausentes y/o duplicados? ¿Son valores que ya nos son útiles para nuestro análsis? ¿O simplemente son errores humanos, situaciones inesperadas o problemas de variables? De cualquier u otro modo, creo que la práctica ayudará mucho a la hora de poder abordar con más facilidad este tipo de actividades.

5 Paso 3. Análisis de los datos

Una vez los datos estén procesados y listos, haz el siguiente análisis:

6 [A] Fácil (deben completarse todos para aprobar)

1. Verifica que los valores en las columnas `'order_hour_of_day'` y `'order_dow'` en la tabla instacart_orders sean razonables (es decir, `'order_hour_of_day'` oscile entre 0 y 23 y `'order_dow'` oscile entre 0 y 6).
2. Crea un gráfico que muestre el número de personas que hacen pedidos dependiendo de la hora del día.
3. Crea un gráfico que muestre qué día de la semana la gente hace sus compras.
4. Crea un gráfico que muestre el tiempo que la gente espera hasta hacer su siguiente pedido, y comenta sobre los valores mínimos y máximos.

6.0.1 [A1] Verifica que los valores sean sensibles

[35]:
```
# Para los valores de la columna 'order_hour_of_day'
print(df_instacart_orders.order_hour_of_day.min())
print()
print(df_instacart_orders.order_hour_of_day.max())
```

```
0

23
```

[36]:
```
# Para los valores de la columna 'order_dow'
print(df_instacart_orders.order_dow.min())
print()
print(df_instacart_orders.order_dow.max())
```

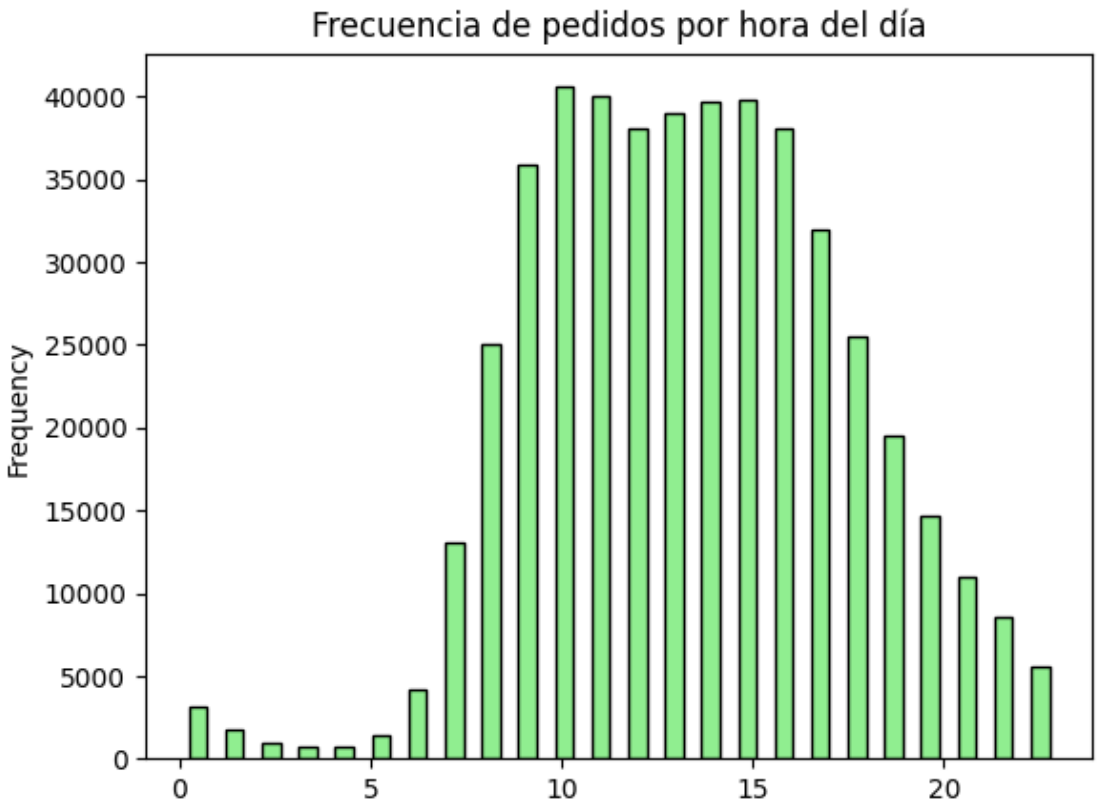
```
0

6
```

Se comprobó que los valores son razonables, es decir, nuestra columna `order_hour_of_day` no registra días de más de 24 horas, o la columna `order_dow` no registra semanas con mas de 7 días. Para ello buscamos los valores mínimos y máximos en nuestras columnas correspondientes, utilizando los métodos `min()` y `max()`.

6.0.2 [A2] Para cada hora del día, ¿cuántas personas hacen órdenes?

[37]:
```
df_instacart_orders['order_hour_of_day'].plot(kind= 'hist',
                                              bins=24,
                                              title='Frecuencia de pedidos por␣
 ↪hora del día',
                                              color='lightgreen',
                                              ec='black',
                                              rwidth=0.5
```

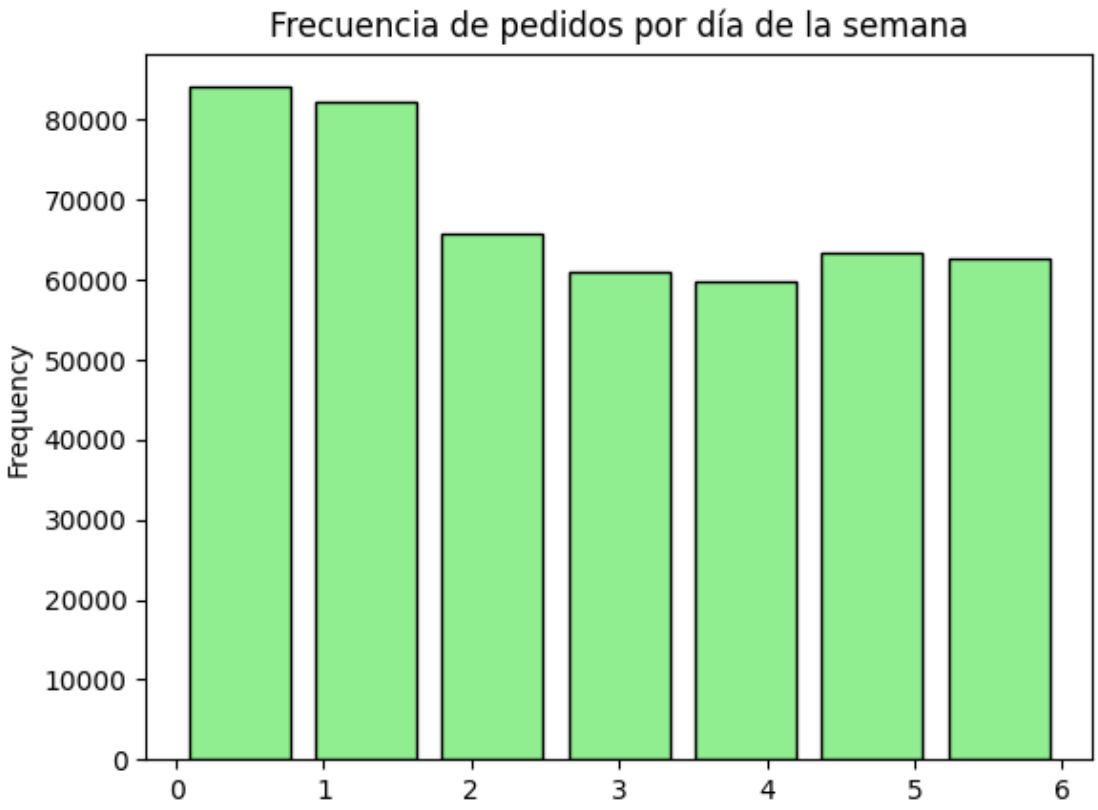
```
                                        )

plt.show()
```



Podemos observar que la hora con mayor frecuencia de pedidos es a las 10:00 am, manteniendose una alta demananda hasta las 15:00 donde comienza a disminuir.

6.0.3 [A3] ¿Qué día de la semana compran víveres las personas?

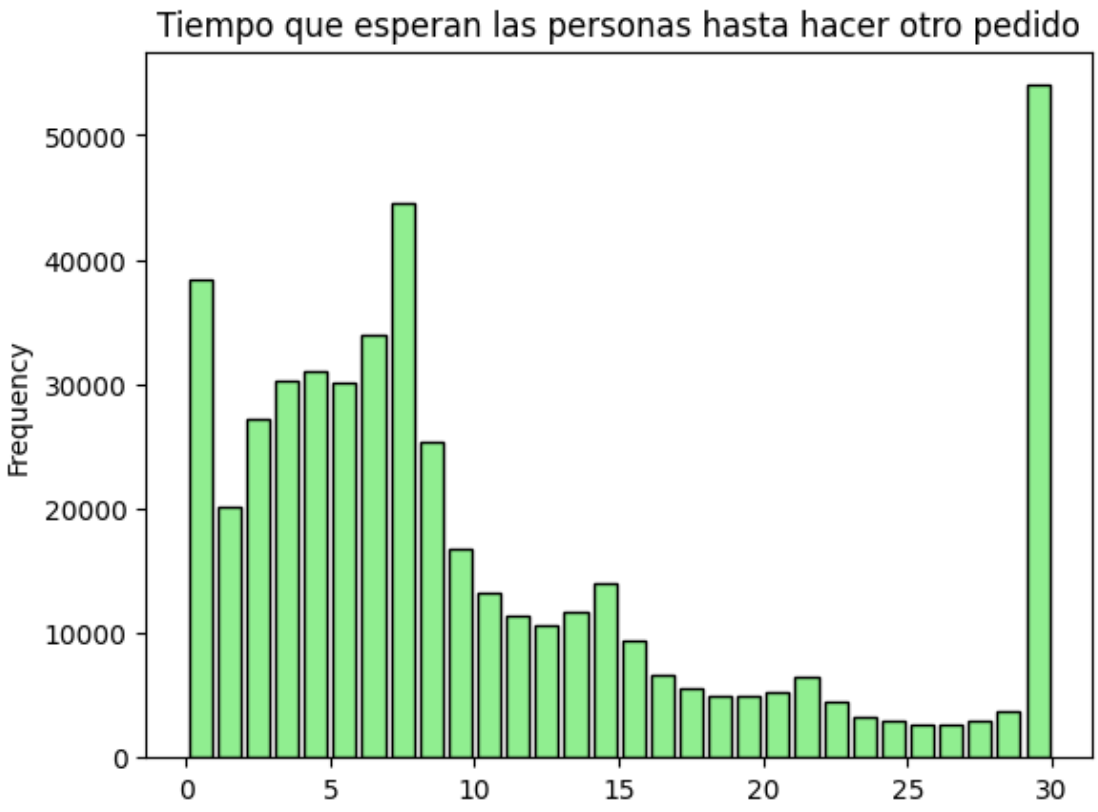
[38]:
```
df_instacart_orders['order_dow'].plot(kind= 'hist',
                                      bins=7,
                                      title= 'Frecuencia de pedidos por día de␣
 ↪la semana',
                                      color='lightgreen',
                                      ec='black',
                                      rwidth=0.8
                                     )
plt.show()
```



Es evidente que el día domingo es el día con una mayor frecuencia de pedidos, seguido del día lunes. Por otra parte los días con menor frecuencia de pedidos son jueves y miercoles.

6.0.4 [A4] ¿Cuánto tiempo esperan las personas hasta hacer otro pedido? Comenta sobre los valores mínimos y máximos.

[39]:
```
df_instacart_orders['days_since_prior_order'].plot(kind='hist',
                                                   bins=30,
                                                   title='Tiempo que esperan␣
 ↪las personas hasta hacer otro pedido',
                                                   color='lightgreen',
                                                   ec='black',
                                                   rwidth=0.8
                                                  )

plt.show()
```



Un gran número de personas tardan 30 días en adquirir otro artículo, aunque es posible que exista sesgo en esta cifra, ya que es posible que clientes que realizaron más de un pedido en un mismo día se hayan registrado con este valor 0; caso contrario con el valor máximo, clientes que quizás tardaron más de 30 días en volver a hacer un pedido se registraron con esta cifra (30).

7 [B] Intermedio (deben completarse todos para aprobar)

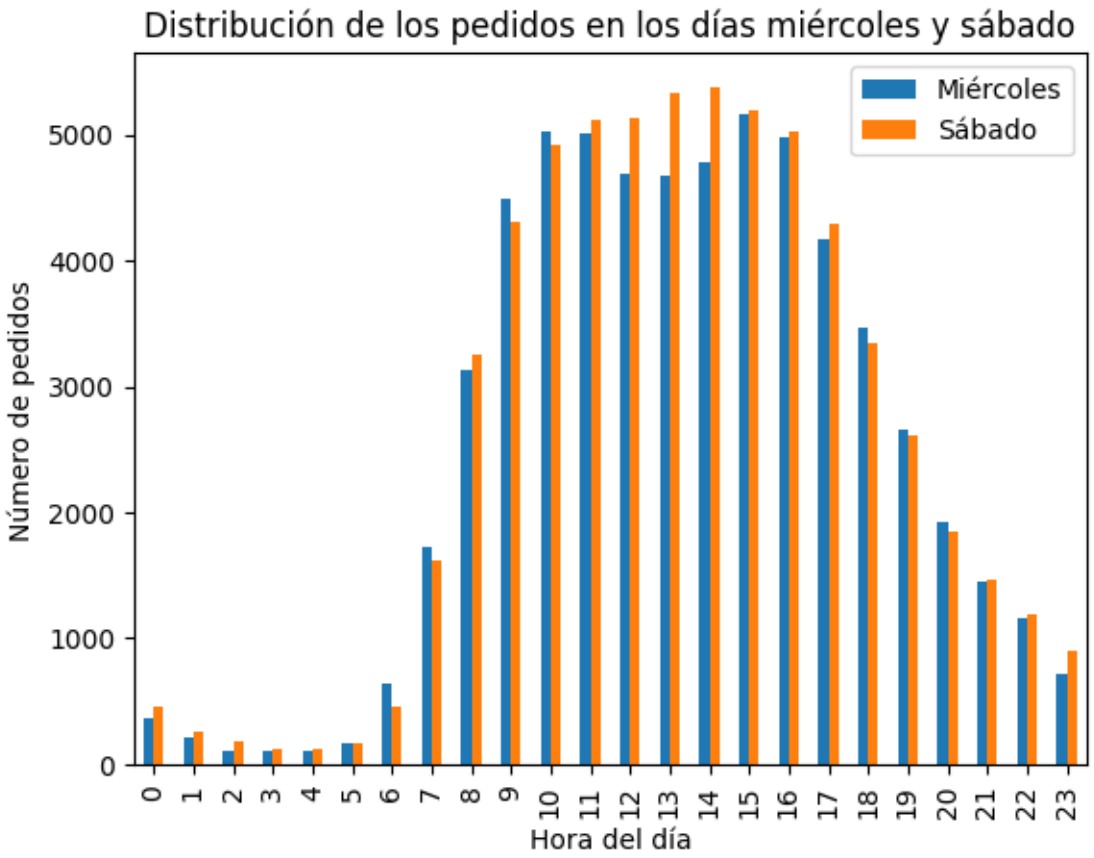
1. ¿Existe alguna diferencia entre las distribuciones `'order_hour_of_day'` de los miércoles y los sábados? Traza gráficos de barra de `'order_hour_of_day'` para ambos días en la misma figura y describe las diferencias que observes.
2. Grafica la distribución para el número de órdenes que hacen los clientes (es decir, cuántos clientes hicieron solo 1 pedido, cuántos hicieron 2, cuántos 3, y así sucesivamente…).
3. ¿Cuáles son los 20 principales productos que se piden con más frecuencia (muestra su identificación y nombre)?

7.0.1 [B1] Diferencia entre miércoles y sábados para 'order_hour_of_day'. Traza gráficos de barra para los dos días y describe las diferencias que veas.

[40]:
```
df_instacart_orders['order_wednesday'] =␣
↪df_instacart_orders[df_instacart_orders['order_dow'] ==␣
↪3]['order_hour_of_day']
df_instacart_orders['order_saturday'] =␣
↪df_instacart_orders[df_instacart_orders['order_dow'] ==␣
↪6]['order_hour_of_day']
df_instacart_orders['order_hour_of_day'] =␣
↪df_instacart_orders['order_hour_of_day'].astype('category')
```

[41]:
```
df_orders_filtered = df_instacart_orders.loc[:,['order_hour_of_day',␣
↪'order_wednesday', 'order_saturday']]
df_orders_filtered = df_orders_filtered.groupby(by='order_hour_of_day').count().
↪reset_index()
```

[42]:
```
df_orders_filtered.plot(x='order_hour_of_day',
                        kind= 'bar',
                        title='Distribución de los pedidos en los días␣
↪miércoles y sábado',
                        xlabel='Hora del día',
                        ylabel='Número de pedidos')

plt.legend(['Miércoles', 'Sábado'])
plt.show()
```

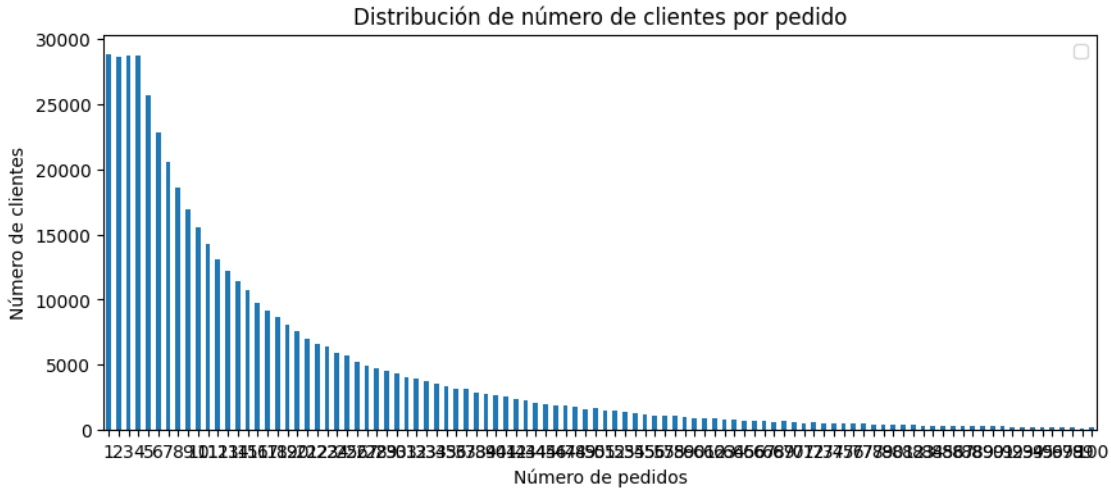


Ambos días de la semana cuentan con una distribución muy similar a lo largo del día, siendo de las 10:00 a las 16:00 el horario con mayor demanda. Si acaso se puede resaltar que los días sabado suele habér un poco más de pedidos en el transcurso del día (11:00 a 14:00).

7.0.2 [B2] ¿Cuál es la distribución para el número de pedidos por cliente?

[43]:
```
df_orders_number = df_instacart_orders.groupby(by='order_number').count().
  ↪reset_index()
df_orders_number = df_orders_number.loc[:,['user_id','order_number']]
```

[44]:
```
df_orders_number.plot(x='order_number',
                      kind='bar',
                      title='Distribución de número de clientes por pedido',
                      xlabel='Número de pedidos',
                      ylabel='Número de clientes',
                      figsize=[10,4],
                      rot=0)

plt.legend('')
plt.show()
```



La gran mayoría de los clientes adquiere entre 1 a 4 artículos, ya que la distribución descendiente, es decir, que entre más artículos se adquieran la frecuencia de clientes será menor.

7.0.3 [B3] ¿Cuáles son los 20 productos más populares (muestra su ID y nombre)?

[45]:
```
# Filtraremos el número de pedidos del dataframe order_products
df_order_products_filtered = df_order_products.loc[:,['order_id','product_id']]
```

[46]:
```
# Filtraremos el ID y nombre de los productos del dataframe products
df_products_filtered = df_products.loc[:,['product_id','product_name']]
```

[47]:
```
# Uniremos los dataframes para identificar por ID y nombre, los 20 productos␣
 ↪más populares
df_popular_products = df_order_products_filtered.
 ↪merge(df_products_filtered,on='product_id')
df_popular_products = df_popular_products.
 ↪groupby(['product_id','product_name']).count()
df_popular_products = df_popular_products.sort_values(by='order_id',␣
 ↪ascending=False)
df_popular_products.columns = ['orders_number']
print(df_popular_products.head(20))
```

```
                                    orders_number
product_id product_name
24852      Banana                           66050
13176      Bag of Organic Bananas           53297
21137      Organic Strawberries             37039
21903      Organic Baby Spinach             33971
47209      Organic Hass Avocado             29773
47766      Organic Avocado                  24689
47626      Large Lemon                      21495
```

```
16797      Strawberries                 20018
26209      Limes                        19690
27845      Organic Whole Milk           19600
27966      Organic Raspberries          19197
22935      Organic Yellow Onion         15898
24964      Organic Garlic               15292
45007      Organic Zucchini             14584
39275      Organic Blueberries          13879
49683      Cucumber Kirby               13675
28204      Organic Fuji Apple           12544
5876       Organic Lemon                12232
8277       Apple Honeycrisp Organic     11993
40706      Organic Grape Tomatoes       11781
```

¿Quién lo diría? La banana es el producto más popular entre los clientes, aunque no me sorprende, no por nada es la fruta más famosa. Los 20 productos más pedidos son frutas y verduras orgánicas, quizás suelen ser de agrado de los clientes.

8 [C] Difícil (deben completarse todos para aprobar)

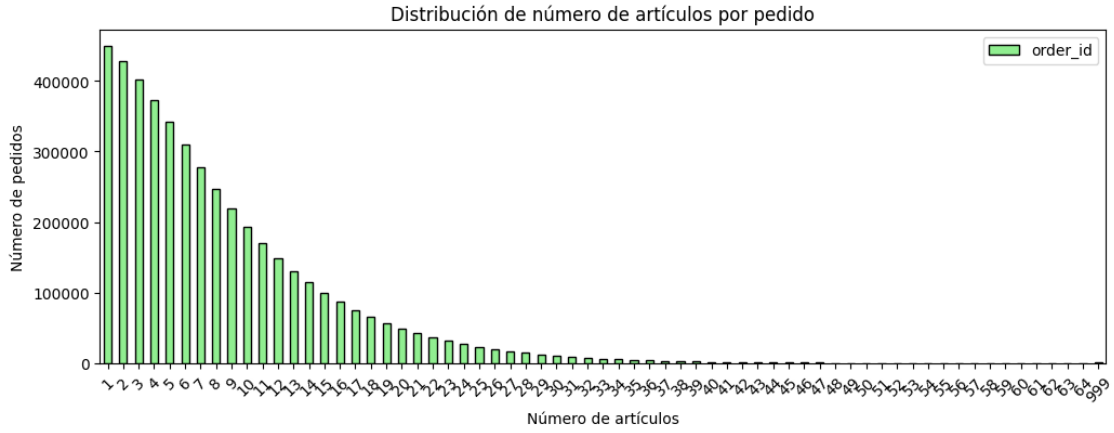
1. ¿Cuántos artículos suelen comprar las personas en un pedido? ¿Cómo es la distribución?
2. ¿Cuáles son los 20 principales artículos que vuelven a pedirse con mayor frecuencia (muestra sus nombres e IDs de los productos)?
3. Para cada producto, ¿cuál es la tasa de repetición del pedido (número de repeticiones de pedido/total de pedidos?
4. Para cada cliente, ¿qué proporción de los productos que pidió ya los había pedido? Calcula la tasa de repetición de pedido para cada usuario en lugar de para cada producto.
5. ¿Cuáles son los 20 principales artículos que la gente pone primero en sus carritos (muestra las IDs de los productos, sus nombres, y el número de veces en que fueron el primer artículo en añadirse al carrito)?

8.0.1 [C1] ¿Cuántos artículos compran normalmente las personas en un pedido? ¿Cómo es la distribución?

[48]:
```
# Agruparemos el dataframe por el número de artículos que añaden a su pedido␣
↪los clientes
df_filtered = df_order_products.groupby(by='add_to_cart_order').count().
↪reset_index()
df_filtered = df_filtered.loc[:,['add_to_cart_order','order_id']]
```

[49]:
```
# Graficaremos el dataframe para poder visualizar la distribución de la␣
↪cantidad de artículos que compran las personas
df_filtered.plot(x='add_to_cart_order',
                 kind='bar',
                 title='Distribución de número de artículos por pedido',
                 xlabel='Número de artículos',
                 ylabel='Número de pedidos',
```

```
                  color='lightgreen',
                  ec='black',
                  figsize=[12,4],
                  rot=45)
plt.show()
```



A mayor número de artículos una menor frecuencia de pedidos, esto quiere decir que nuestra distribución es descendiente a medida que un mayor numero de artículos son añadidios a un pedido.

8.0.2 [C2] ¿Cuáles son los 20 principales artículos que vuelven a pedirse con mayor frecuencia (muestra sus nombres e IDs de los productos)?

[50]:
```
# Filtraremos los productos que han vuelto a pedirse del dataframe␣
 ↪order_products
df_reordered_filtered = df_order_products[df_order_products['reordered']==1]
```

[51]:
```
# Uniremos los dataframes products y order_products
df_reordered_products = df_reordered_filtered.merge(df_products,␣
 ↪on='product_id')
```

[52]:
```
# Agruparemos el dataframe por nombre e ID los productos que más veces se han␣
 ↪vuelto a ordenar
df_reordered_products = df_reordered_products.
 ↪groupby(['product_id','product_name']).count()
df_reordered_products = df_reordered_products.sort_values(by='reordered',␣
 ↪ascending=False)
print(df_reordered_products.head(20))
```

```
                                    order_id  add_to_cart_order  reordered  \
product_id product_name
24852      Banana                      55763              55763      55763
13176      Bag of Organic Bananas      44450              44450      44450
```

21137	Organic Strawberries	28639	28639	28639
21903	Organic Baby Spinach	26233	26233	26233
47209	Organic Hass Avocado	23629	23629	23629
47766	Organic Avocado	18743	18743	18743
27845	Organic Whole Milk	16251	16251	16251
47626	Large Lemon	15044	15044	15044
27966	Organic Raspberries	14748	14748	14748
16797	Strawberries	13945	13945	13945
26209	Limes	13327	13327	13327
22935	Organic Yellow Onion	11145	11145	11145
24964	Organic Garlic	10411	10411	10411
45007	Organic Zucchini	10076	10076	10076
49683	Cucumber Kirby	9538	9538	9538
28204	Organic Fuji Apple	8989	8989	8989
8277	Apple Honeycrisp Organic	8836	8836	8836
39275	Organic Blueberries	8799	8799	8799
5876	Organic Lemon	8412	8412	8412
49235	Organic Half & Half	8389	8389	8389

product_id	product_name	aisle_id	department_id
24852	Banana	55763	55763
13176	Bag of Organic Bananas	44450	44450
21137	Organic Strawberries	28639	28639
21903	Organic Baby Spinach	26233	26233
47209	Organic Hass Avocado	23629	23629
47766	Organic Avocado	18743	18743
27845	Organic Whole Milk	16251	16251
47626	Large Lemon	15044	15044
27966	Organic Raspberries	14748	14748
16797	Strawberries	13945	13945
26209	Limes	13327	13327
22935	Organic Yellow Onion	11145	11145
24964	Organic Garlic	10411	10411
45007	Organic Zucchini	10076	10076
49683	Cucumber Kirby	9538	9538
28204	Organic Fuji Apple	8989	8989
8277	Apple Honeycrisp Organic	8836	8836
39275	Organic Blueberries	8799	8799
5876	Organic Lemon	8412	8412
49235	Organic Half & Half	8389	8389

Que curioso, la gran mayoría de los productos más populares son los que más han sido reordenados, eso comprueba que las frutas y verduras orgánicas son de agrado de los clientes.

8.0.3 [C3] Para cada producto, ¿cuál es la proporción de las veces que se pide y que se vuelve a pedir?

[53]:
```
# Contaremos el número total de pedidos por producto
total_orders = df_order_products.groupby('product_id').size()

# Contarmeos el número de veces que vuelve a ser reordenado cada producto
reordered_products = df_order_products[df_order_products['reordered'] == 1].
 ↪groupby('product_id').size()
```

[54]:
```
# Calculamos la tasa de repetición del pedido
reordered_rate = reordered_products / total_orders

# Convertimos al formato tabla para mejorar su presentación
df_reordered_rate = reordered_rate.reset_index()
df_reordered_rate.columns = ['product_id', 'reordered_rate']
```

[55]:
```
# Uniremos con la tabla de products para obtener los nombres
df_reordered_rate_info = df_reordered_rate.merge(df_products[['product_id',␣
 ↪'product_name']],
                                                 on='product_id',
                                                 how='left')
# Mostramos la tabla resultante
display(df_reordered_rate_info)
```

```
       product_id  reordered_rate  \
0               1        0.564286
1               2             NaN
2               3        0.738095
3               4        0.510204
4               7        0.500000
…             …              …
45568       49690        0.800000
45569       49691        0.430556
45570       49692        0.416667
45571       49693        0.440000
45572       49694        0.333333

                                            product_name
0                              Chocolate Sandwich Cookies
1                                       All-Seasons Salt
2                    Robust Golden Unsweetened Oolong Tea
3      Smart Ones Classic Favorites Mini Rigatoni Wit…
4                          Pure Coconut Water With Orange
…                                                     …
45568                       HIGH PERFORMANCE ENERGY DRINK
45569                       ORIGINAL PANCAKE & WAFFLE MIX
45570    ORGANIC INSTANT OATMEAL LIGHT MAPLE BROWN SUGAR
```

```
45571                              SPRING WATER BODY WASH
45572                            BURRITO- STEAK & CHEESE

[45573 rows x 3 columns]
```

8.0.4 [C4] Para cada cliente, ¿qué proporción de sus productos ya los había pedido?

[56]:
```
# Vamos a crear una tabla que contenga la tabla de pedidos y la tabla␣
↪df_order_products usando 'order_id'
df_orders_id = df_instacart_orders.merge(df_order_products, on='order_id',␣
↪how='left')
```

[57]:
```
# Contaremos el número total de productos pedidos por cada cliente (usando␣
↪'user_id')
total_orders_per_client = df_orders_id.groupby('user_id').size()

# Contaremos cuantos pedidos fueron reordenados por cada cliente (usando la␣
↪condición 'reordered'==1)
reordered_rate_per_client = df_orders_id[df_orders_id['reordered'] == 1].
↪groupby('user_id').size()
```

[58]:
```
# Convertimos al formato tabla para mejorar su presentación
df_reordered_rate_per_client = reordered_rate_per_client.reset_index()
df_reordered_rate_per_client.columns = ['user_id', 'reordered_rate']

# Mostramos el resultado
display(df_reordered_rate_per_client)
```

```
        user_id  reordered_rate
0             2               1
1             5               8
2             7              13
3            11               3
4            12               3
…             …               …
132986   206203               6
132987   206206              15
132988   206207              41
132989   206208              87
132990   206209               8

[132991 rows x 2 columns]
```

8.0.5 [C5] ¿Cuáles son los 20 principales artículos que las personas ponen primero en sus carritos?

[59]:
```
# Filtraremos los principales artículos que los clientes han puesto en sus␣
 ↪carritos
first_in_order = df_order_products[df_order_products['add_to_cart_order'] == 1]

# Uniremos los dataframes first_in_orders y order_products
df_first_in_order = first_in_order.merge(df_products, on='product_id', how=␣
 ↪'left')
```

[60]:
```
# Agruparemos el dataframe por nombre e ID los productos que se ponen primero␣
 ↪en los carritos
df_first_in_order = df_first_in_order.groupby(['product_id','product_name']).
 ↪size()

# Mostraremos los 20 productos que las personas ponen primero en sus carritos
df_popular_products.columns = ['first_in_order']
print(df_popular_products.head(20))
```

		first_in_order
product_id	product_name	
24852	Banana	66050
13176	Bag of Organic Bananas	53297
21137	Organic Strawberries	37039
21903	Organic Baby Spinach	33971
47209	Organic Hass Avocado	29773
47766	Organic Avocado	24689
47626	Large Lemon	21495
16797	Strawberries	20018
26209	Limes	19690
27845	Organic Whole Milk	19600
27966	Organic Raspberries	19197
22935	Organic Yellow Onion	15898
24964	Organic Garlic	15292
45007	Organic Zucchini	14584
39275	Organic Blueberries	13879
49683	Cucumber Kirby	13675
28204	Organic Fuji Apple	12544
5876	Organic Lemon	12232
8277	Apple Honeycrisp Organic	11993
40706	Organic Grape Tomatoes	11781

8.0.6 Conclusion general del proyecto:

Como conclusión, este proyecto comenzó con la etapa de exploración de datos, en donde se cargó y examinó el conjunto de datos proporcionado; después se abordó la etapa de preparación de datos, en donde se corrigierón errores de valores duplicados y ausentes, en otras palabras, se limpiarón

los datos; para finalmente hacer nuestro análisis de los datos, obteniendo gráficos para comprender mejor las tendencías de nuestros clientes, sus preferencias y hábitos.