



## **Unidad 2: Procesamiento y Limpieza de Datos**

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

MARTÍNEZ RUGERIO LUCÍA

INTRODUCCIÓN A LA CIENCIA DE DATOS

M.C. JAIME ALEJANDRO ROMERO SIERRA

21 DE OCTUBRE DE 2024

# Reporte Fase 1

## 1. Título del Proyecto

- Nombre del proyecto: “Análisis del precio histórico de las acciones de McDonald 's.”

## 2. Objetivo del Proyecto

- Definir si las acciones de McDonald 's tienen un comportamiento descrito por patrones.
  - Si es así, encontrar el periodo en que las acciones de McDonalds tienen su precio mínimo y máximo.

## 3. Descripción del Problema

Aunque por lo general las acciones de diversas empresas muestran patrones en su comportamiento, hay que reconocer que en realidad, estas pueden verse afectadas de último momento por múltiples factores como pueden ser el sentimiento del mercado, cambios en la economía como la inflación o la disposición del consumidor a gastar, la disponibilidad de las materias primas; que para empresas como McDonald's es de suma importancia, ya que de no contar con los insumos necesarios para su producción, podría suponer una pérdida; y en general cualquier evento inesperado.

Conocer sobre el posible comportamiento de estas acciones ayuda a los inversores a anticipar movimientos y les permite tomar decisiones informadas sobre cuándo comprar o vender las acciones, asimismo reduce el riesgo que conlleva invertir, y prepara al inversor para enfrentar distintos escenarios e identificar futuras oportunidades con estrategias que maximicen las ganancias y minimicen las pérdidas.

Por lo tanto, este proyecto pretende ofrecer una guía para aquellos que estén interesados en la compra de acciones de McDonald 's. Se analizará el precio histórico de las acciones y se utilizarán herramientas como Tradingview para tener una visión más clara con el fin de comprender su evolución y lo más importante, su comportamiento.

## 4. Recursos Disponibles

- **Tecnología y Herramientas:**

1. Python

2. Pandas
3. TradingView

- **Datos:**

1. Date: La fecha de negociación.
2. Open: El precio al que se abrió la acción en un día determinado.
3. High: El precio más alto de la acción durante la sesión de negociación.
4. Low: El precio más bajo de la acción durante la sesión de negociación.
5. Close: El precio de la acción al cierre del mercado.
6. Adj close: el precio de cierre ajustado por dividendos y divisiones de acciones.
7. Volume: El número de acciones negociadas durante el día.

## 5. Hipótesis Iniciales

- **Hipótesis 1:** La demanda de las acciones de McDonald 's subirán y consigo el precio de estas.
- **Hipótesis 2:** El precio de las acciones se ve afectado por el sentimiento del mercado.
- **Hipótesis 3:** Si volume tiene un valor alto, es porque ese día convino comprar acciones de McDonald 's .

## 6. Definición de Stakeholders Clave

- **Inversores:** Utilizarán la información para saber cuándo invertir
- **Accionistas:** Utilizarán la información para conocer el valor de sus inversiones y su rendimiento
- **Directivos:** Usarán la información para tomar decisiones estratégicas en el momento adecuado.
- **Marketing:** Utilizarán la información para crear proyectos que mejoren el sentimiento del mercado.

## 7. Preguntas Clave

1. ¿Qué eventos económicos han afectado el precio de las acciones?
2. ¿Cómo son las acciones de McDonalds en comparación con sus competidores?
3. ¿Qué impacto tiene el sentimiento del mercado?
4. ¿Cuál es un patrón estacional que se puede identificar?
5. ¿Cómo influye la disponibilidad de materia prima?
6. ¿Cómo se ven afectadas las acciones por las tendencias de consumo?
7. ¿Cómo responden las acciones de McDonalds ante eventos inesperados?
8. ¿Qué estrategias de marketing son efectivas para mejorar el rendimiento de las acciones?
9. ¿Cuál es el futuro de los precios de las acciones de McDonald 's?
10. ¿Qué papel juegan las innovaciones del menú?
11. ¿Cuál es la relación entre las acciones y la satisfacción al cliente?

## **8. Fuentes de Datos Identificadas**

Datos históricos de los precios de las acciones

Modelos de predicción de sentimiento

## **9. Justificación del Proyecto**

El analizar el comportamiento de las acciones de McDonald 's es relevante ya que al ser una de las cadenas de comida rápida más conocidas, su desempeño tiene un impacto significativo en la industria, comprender su comportamiento implica conocer sobre tendencias en este sector y de esta forma ayudar a los inversores a tomar decisiones informadas.

Actualmente, nuestro entorno económico se encuentra en constante cambio, lo que hace que la predicción sea cada vez más necesaria para anticipar aquellos movimientos en el mercado, son estas investigaciones las que nos sirven como base para futuras investigaciones sobre el comportamiento del mercado en general.

Un proyecto como este no solo sirve para inversores experimentados sino que también ayuda a que mediante la identificación de patrones, aquellos que están empezando en el mundo de las acciones, vayan por un camino menos riesgoso y logren desarrollar estrategias propias. También las decisiones que una empresa toma nos dan una visión de cómo vive la comunidad en general.

## **11.- ¿Cuántos datos y que tipo son?**

```
import pandas as pd
url='https://raw.githubusercontent.com/LuciaMaRu/df_sucio/refs/heads/main/df_sucio.csv'
df=pd.read_csv(url)
df
```

Python

	Date	Open	High	Low	Close	Adj Close	Volume
0	1966-07-05	0.000000	0.2736630141735077	invalid	0.2695469856262207	NaN	388800.0
1	1966-07-06	0.000000	0.2839510142803192	0.2674899995326996	0.2839510142803192	0.1221895366907119	692550.0
2	1966-07-07	0.000000	0.2911520004272461	0.2716050148010254	0.2736630141735077	0.1177623867988586	1858950.0
3	1966-07-08	0.000000	0.2767490148544311	0.2674899995326996	0.2767490148544311	0.1190903931856155	1239300.0
4	1966-07-11	0.000000	0.2839510142803192	0.2726339995861053	0.2757200002670288	0.1186476200819015	656100.0
...	...	...	...	...	...	...	...
18769	2022-03-04	234.619995	236.38999938964844	231.63999938964844	235.80999755859372	NaN	4184100.0
18770	2001-09-27	26.950001	27.0	26.43000030517578	27.0	14.856624603271484	10036500.0
18771	1995-02-09	16.500000	16.5625	16.4375	16.5	8.753664016723633	2089200.0
18772	1993-08-17	13.562500	13.59375	13.46875	13.53125	7.093704700469971	2456000.0
18773	1969-04-28	0.000000	0.534978985786438	0.5267490148544312	0.534978985786438	0.2308428138494491	911250.0

18774 rows × 7 columns

# 11.1.- Análisis Inicial de la Base de Datos

```
# resumen estadístico de los datos.
df.describe()
```

[4]

Python

...

	Open	Volume
count	17448.000000	1.746400e+04
mean	50.652481	4.267833e+06
std	73.632531	3.120997e+06
min	0.000000	4.860000e+04
25%	1.589506	2.357650e+06
50%	17.137500	3.579100e+06
75%	66.074999	5.405400e+06
max	300.950012	8.698130e+07

```
#Valores faltantes
df.isnull().sum()
```

[5] Python

```
... Date      908
Open      1326
High      946
Low       962
Close     914
Adj Close  960
Volume    1310
dtype: int64
```

```
#Filas duplicadas
df.duplicated().sum()
```

[6] Python

```
... np.int64(1572)
```

```
#Tipos de datos
df.info()
```

[7] Python

```
... <class 'pandas.core.frame.DataFrame'>
RangeIndex: 18774 entries, 0 to 18773
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Date        17866 non-null  object
1   Open        17448 non-null  float64
2   High        17828 non-null  object
3   Low         17812 non-null  object
4   Close       17860 non-null  object
5   Adj Close   17814 non-null  object
6   Volume      17464 non-null  float64
dtypes: float64(2), object(5)
memory usage: 1.0+ MB
```

1. Los datos están como object
2. Poner date en formato de fecha
3. Cambiar formatos a float
4. Muchas filas duplicadas
5. Valores NaN

### 3.- Limpieza de Datos

1. Eliminar todas las filas duplicadas
2. Eliminar los 'invalid'
3. Eliminar todos los NaN de 'Date' porque considero que podría causar anomalías si se llena con una fecha promedio.
4. Cambiar los formatos a fecha y float
5. Cuando ya estén en float, llenar los demás NaN con el promedio de los datos ya que son precios y considero que es factible.
6. Revisar que no haya más datos NaN
7. Ordenar los datos
8. Reindexar
9. Guardar el CSV

#Eliminar filas duplicadas  
df1=df.drop\_duplicates()  
df1

[11] ✓ 0.0s Python

...

	Date	Open	High	Low	Close	Adj Close	Volume
0	1966-07-05	0.000000	0.2736630141735077	invalid	0.2695469856262207	NaN	388800.0
1	1966-07-06	0.000000	0.2839510142803192	0.2674899995326996	0.2839510142803192	0.1221895366907119	692550.0
2	1966-07-07	0.000000	0.2911520004272461	0.2716050148010254	0.2736630141735077	0.1177623867988586	1858950.0
3	1966-07-08	0.000000	0.2767490148544311	0.2674899995326996	0.2767490148544311	0.1190903931856155	1239300.0
4	1966-07-11	0.000000	0.2839510142803192	0.2726339995861053	0.2757200002670288	0.1186476200819015	656100.0
...	...	...	...	...	...	...	...
18764	1976-03-29	1.540123	1.558642029762268	1.5401229858398438	1.5493830442428589	0.6685570478439331	741150.0
18767	1967-11-03	0.000000	0.5843619704246521	0.5596709847450256	invalid	0.2408370822668075	1129950.0
18769	2022-03-04	234.619995	236.38999938964844	231.63999938964844	235.80999755859372	NaN	4184100.0
18770	2001-09-27	26.950001	27.0	26.43000030517578	27.0	14.856624603271484	10036500.0
18771	1995-02-09	16.500000	16.5625	16.4375	16.5	8.753664016723633	2089200.0

17202 rows x 7 columns

#Eliminar invalid  
lista\_col=df1.columns  
df2=df1  
for i in lista\_col:  
| df2=df2[df2[i] != 'invalid']  
df2

[31] ✓ 0.1s Python

...

	Date	Open	High	Low	Close	Adj Close	Volume
1	1966-07-06	0.000000	0.2839510142803192	0.2674899995326996	0.2839510142803192	0.1221895366907119	692550.0
2	1966-07-07	0.000000	0.2911520004272461	0.2716050148010254	0.2736630141735077	0.1177623867988586	1858950.0
3	1966-07-08	0.000000	0.2767490148544311	0.2674899995326996	0.2767490148544311	0.1190903931856155	1239300.0
4	1966-07-11	0.000000	0.2839510142803192	0.2726339995861053	0.2757200002670288	0.1186476200819015	656100.0
5	1966-07-12	0.000000	0.2736630141735077	0.270576000213623	0.2716050148010254	0.1168767735362052	303750.0
...	...	...	...	...	...	...	...
18763	2005-09-21	32.490002	32.619998931884766	31.309999465942383	31.420000076293945	18.28680229187012	13572300.0
18764	1976-03-29	1.540123	1.558642029762268	1.5401229858398438	1.5493830442428589	0.6685570478439331	741150.0
18769	2022-03-04	234.619995	236.38999938964844	231.63999938964844	235.80999755859372	NaN	4184100.0
18770	2001-09-27	26.950001	27.0	26.43000030517578	27.0	14.856624603271484	10036500.0
18771	1995-02-09	16.500000	16.5625	16.4375	16.5	8.753664016723633	2089200.0

15413 rows x 7 columns

#Valores faltantes  
df3=df2.dropna(subset=['Date'])  
df3

[32] ✓ 0.0s Python

...

	Date	Open	High	Low	Close	Adj Close	Volume
1	1966-07-06	0.000000	0.2839510142803192	0.2674899995326996	0.2839510142803192	0.1221895366907119	692550.0
2	1966-07-07	0.000000	0.2911520004272461	0.2716050148010254	0.2736630141735077	0.1177623867988586	1858950.0
3	1966-07-08	0.000000	0.2767490148544311	0.2674899995326996	0.2767490148544311	0.1190903931856155	1239300.0
4	1966-07-11	0.000000	0.2839510142803192	0.2726339995861053	0.2757200002670288	0.1186476200819015	656100.0
5	1966-07-12	0.000000	0.2736630141735077	0.270576000213623	0.2716050148010254	0.1168767735362052	303750.0
...	...	...	...	...	...	...	...
18763	2005-09-21	32.490002	32.619998931884766	31.309999465942383	31.420000076293945	18.28680229187012	13572300.0
18764	1976-03-29	1.540123	1.558642029762268	1.5401229858398438	1.5493830442428589	0.6685570478439331	741150.0
18769	2022-03-04	234.619995	236.38999938964844	231.63999938964844	235.80999755859372	NaN	4184100.0
18770	2001-09-27	26.950001	27.0	26.43000030517578	27.0	14.856624603271484	10036500.0
18771	1995-02-09	16.500000	16.5625	16.4375	16.5	8.753664016723633	2089200.0

14592 rows x 7 columns

```
> #Cambiar formatos
df3.columns
[33] ✓ 0.0s Python

... Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
+ Code + Markdown

df3['Date'] = pd.to_datetime(df3['Date'])
lista2_col=['High', 'Low', 'Close', 'Adj Close']
for i in lista2_col:
    df3[i]=df3[i].astype(float)
df3.info()

[34] ✓ 0.2s Python

... C:\Users\marul\AppData\Local\Temp\ipykernel_27764\215297052.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df3['Date'] = pd.to_datetime(df3['Date'])
C:\Users\marul\AppData\Local\Temp\ipykernel_27764\215297052.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df3[i]=df3[i].astype(float)
C:\Users\marul\AppData\Local\Temp\ipykernel_27764\215297052.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df3[i]=df3[i].astype(float)
C:\Users\marul\AppData\Local\Temp\ipykernel_27764\215297052.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df3[i]=df3[i].astype(float)
C:\Users\marul\AppData\Local\Temp\ipykernel_27764\215297052.py:4: SettingWithCopyWarning:
...
Try using .loc[row_indexer,col_indexer] = value instead

In [34]: pandas.core.frame.DataFrame
Index: 14592 entries, 1 to 18771
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Date        14592 non-null  datetime64[ns]
1   Open        13478 non-null  float64
2   High        13775 non-null  float64
3   Low         13760 non-null  float64
4   Close       13811 non-null  float64
5   Adj Close   13757 non-null  float64
6   Volume      13491 non-null  float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 912.0 KB
```



```
#Valores faltantes
df4=df3
lista1=['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']
for i in lista1:
    df3[i].fillna(df3[i].mean(), inplace=True)
df4
```

[45] ✓ 0.0s Python

... C:\Users\marul\AppData\Local\Temp\ipykernel\_27764\1316582885.py:5: FutureWarning: A value is trying to be set on a copy of a Data  
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are sett:  
  
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df|  
  
df3[i].fillna(df3[i].mean(), inplace=True)  
C:\Users\marul\AppData\Local\Temp\ipykernel\_27764\1316582885.py:5: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame  
  
See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-ver](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver):  
df3[i].fillna(df3[i].mean(), inplace=True)

	Date	Open	High	Low	Close	Adj Close	Volume
1	1966-07-06	0.000000	0.283951	0.267490	0.283951	0.122190	692550.0
2	1966-07-07	0.000000	0.291152	0.271605	0.273663	0.117762	1858950.0
3	1966-07-08	0.000000	0.276749	0.267490	0.276749	0.119090	1239300.0
4	1966-07-11	0.000000	0.283951	0.272634	0.275720	0.118648	656100.0
5	1966-07-12	0.000000	0.273663	0.270576	0.271605	0.116877	303750.0
...	...	...	...	...	...	...	...
18763	2005-09-21	32.490002	32.619999	31.309999	31.420000	18.286802	13572300.0
18764	1976-03-29	1.540123	1.558642	1.540123	1.549383	0.668557	741150.0
18769	2022-03-04	234.619995	236.389999	231.639999	235.809998	41.474216	4184100.0
18770	2001-09-27	26.950001	27.000000	26.430000	27.000000	14.856625	10036500.0
18771	1995-02-09	16.500000	16.562500	16.437500	16.500000	8.753664	2089200.0

14592 rows x 7 columns

```
df4.describe()
```

[48] ✓ 0.1s Python

...

	Date	Open	High	Low	Close	Adj Close	Volume
count	14592	14592.000000	14592.000000	14592.000000	14592.000000	14592.000000	1.459200e+04
mean	1995-09-23 03:34:38.289473792	51.010934	51.539300	50.502152	51.160889	41.474216	4.270142e+06
min	1966-07-06 00:00:00	0.000000	0.156379	0.152263	0.154321	0.066407	4.860000e+04
25%	1981-02-15 18:00:00	1.770834	1.716049	1.672840	1.694444	0.762684	2.452775e+06
50%	1995-10-31 12:00:00	23.812500	23.375000	22.812500	23.062500	12.546503	3.817300e+06
75%	2010-04-23 18:00:00	58.522500	60.502500	59.124999	60.312501	41.474216	5.208300e+06
max	2024-09-18 00:00:00	300.950012	302.390015	299.309998	300.529999	296.690002	8.698130e+07
std	NaN	70.932223	72.375330	71.027183	71.872657	67.220943	2.979676e+06

```
df4.isnull().sum()
```

[47] ✓ 0.0s Python

...

```
Date      0
Open      0
High      0
Low       0
Close     0
Adj Close 0
Volume    0
dtype: int64
```

#Ordenar los datos por fecha  
df5=df4.sort\_values(by=['Date'])  
df5

[58] ✓ 0.0s Python

...

	Date	Open	High	Low	Close	Adj Close	Volume
1	1966-07-06	0.000000	0.283951	0.267490	0.283951	0.122190	692550.0
2	1966-07-07	0.000000	0.291152	0.271605	0.273663	0.117762	1858950.0
3	1966-07-08	0.000000	0.276749	0.267490	0.276749	0.119090	1239300.0
4	1966-07-11	0.000000	0.283951	0.272634	0.275720	0.118648	656100.0
5	1966-07-12	0.000000	0.273663	0.270576	0.271605	0.116877	303750.0
...	...	...	...	...	...	...	...
14646	2024-09-12	290.170013	292.690002	288.260010	292.350006	292.350006	2274700.0
14647	2024-09-13	294.489990	296.739990	292.619995	296.529999	296.529999	2205500.0
14648	2024-09-16	297.420013	300.109985	295.040009	296.690002	296.690002	2916200.0
14649	2024-09-17	297.000000	297.390015	292.170013	293.750000	293.750000	2997400.0
14650	2024-09-18	293.880005	295.100006	290.369995	292.029999	292.029999	1788200.0

14592 rows × 7 columns

#Reindexar  
df6 = df5.reset\_index(drop=True)  
df6

[59] ✓ 0.0s Python

	Date	Open	High	Low	Close	Adj Close	Volume
0	1966-07-06	0.000000	0.283951	0.267490	0.283951	0.122190	692550.0
1	1966-07-07	0.000000	0.291152	0.271605	0.273663	0.117762	1858950.0
2	1966-07-08	0.000000	0.276749	0.267490	0.276749	0.119090	1239300.0
3	1966-07-11	0.000000	0.283951	0.272634	0.275720	0.118648	656100.0
4	1966-07-12	0.000000	0.273663	0.270576	0.271605	0.116877	303750.0
...	...	...	...	...	...	...	...
14587	2024-09-12	290.170013	292.690002	288.260010	292.350006	292.350006	2274700.0
14588	2024-09-13	294.489990	296.739990	292.619995	296.529999	296.529999	2205500.0
14589	2024-09-16	297.420013	300.109985	295.040009	296.690002	296.690002	2916200.0
14590	2024-09-17	297.000000	297.390015	292.170013	293.750000	293.750000	2997400.0
14591	2024-09-18	293.880005	295.100006	290.369995	292.029999	292.029999	1788200.0

14592 rows × 7 columns

#Guardar CSV  
df6.to\_csv("Base\_limpia.csv", index=True)

[60] ✓ 0.2s Python

# Comparación

df

✓ 0.7s

Python

	Date	Open	High	Low	Close	Adj Close	Volume
0	1966-07-05	0.000000	0.2736630141735077	invalid	0.2695469856262207	NaN	388800.0
1	1966-07-06	0.000000	0.2839510142803192	0.2674899995326996	0.2839510142803192	0.1221895366907119	692550.0
2	1966-07-07	0.000000	0.2911520004272461	0.2716050148010254	0.2736630141735077	0.1177623867988586	1858950.0
3	1966-07-08	0.000000	0.2767490148544311	0.2674899995326996	0.2767490148544311	0.1190903931856155	1239300.0
4	1966-07-11	0.000000	0.2839510142803192	0.2726339995861053	0.2757200002670288	0.1186476200819015	656100.0
...	...	...	...	...	...	...	...
18769	2022-03-04	234.619995	236.38999938964844	231.63999938964844	235.80999755859372	NaN	4184100.0
18770	2001-09-27	26.950001	27.0	26.43000030517578	27.0	14.856624603271484	10036500.0
18771	1995-02-09	16.500000	16.5625	16.4375	16.5	8.753664016723633	2089200.0
18772	1993-08-17	13.562500	13.59375	13.46875	13.53125	7.093704700469971	2456000.0
18773	1969-04-28	0.000000	0.534978985786438	0.5267490148544312	0.534978985786438	0.2308428138494491	911250.0

18774 rows × 7 columns

df6

✓ 0.0s

Python

	Date	Open	High	Low	Close	Adj Close	Volume
0	1966-07-06	0.000000	0.283951	0.267490	0.283951	0.122190	692550.0
1	1966-07-07	0.000000	0.291152	0.271605	0.273663	0.117762	1858950.0
2	1966-07-08	0.000000	0.276749	0.267490	0.276749	0.119090	1239300.0
3	1966-07-11	0.000000	0.283951	0.272634	0.275720	0.118648	656100.0
4	1966-07-12	0.000000	0.273663	0.270576	0.271605	0.116877	303750.0
...	...	...	...	...	...	...	...
14587	2024-09-12	290.170013	292.690002	288.260010	292.350006	292.350006	2274700.0
14588	2024-09-13	294.489990	296.739990	292.619995	296.529999	296.529999	2205500.0
14589	2024-09-16	297.420013	300.109985	295.040009	296.690002	296.690002	2916200.0
14590	2024-09-17	297.000000	297.390015	292.170013	293.750000	293.750000	2997400.0
14591	2024-09-18	293.880005	295.100006	290.369995	292.029999	292.029999	1788200.0

14592 rows × 7 columns

#Valores faltantes

df.isnull().sum()

✓ 0.1s

...  
Date 908  
Open 1326  
High 946  
Low 962  
Close 914  
Adj Close 960  
Volume 1310  
dtype: int64

#Filas duplicadas

df.duplicated().sum()

✓ 0.0s

...  
np.int64(1572)

#Tipos de datos

df.info()

✓ 0.0s

...  
<class 'pandas.core.frame.DataFrame'  
RangeIndex: 18774 entries, 0 to 18773  
Data columns (total 7 columns):  
# Column Non-Null Count Dtype  
--- ---  
0 Date 17866 non-null object  
1 Open 17448 non-null float64  
2 High 17828 non-null object  
3 Low 17812 non-null object  
4 Close 17860 non-null object  
5 Adj Close 17814 non-null object  
6 Volume 17464 non-null float64  
dtypes: float64(2), object(5)  
memory usage: 1.0+ MB

df6.isnull().sum()

✓ 0.0s

...  
Date 0  
Open 0  
High 0  
Low 0  
Close 0  
Adj Close 0  
Volume 0  
dtype: int64

df6.info()

✓ 0.0s

...  
<class 'pandas.core.frame.DataFrame'  
RangeIndex: 14592 entries, 0 to 14591  
Data columns (total 7 columns):  
# Column Non-Null Count Dtype  
--- ---  
0 Date 14592 non-null datetime64[ns]  
1 Open 14592 non-null float64  
2 High 14592 non-null float64  
3 Low 14592 non-null float64  
4 Close 14592 non-null float64  
5 Adj Close 14592 non-null float64  
6 Volume 14592 non-null float64  
dtypes: datetime64[ns](1), float64(6)  
memory usage: 798.1 KB

df6.duplicated().sum()

✓ 0.0s

...  
np.int64(0)