Proceso: Para limpieza de datos

SAMUEL DE MARÍA CABRERA FLORES

MATERIA: Introducción a la Ciencia de Datos

Nombre del profesor: JAIME ALEJANDRO ROMERO – SIERRA

Fecha de entrega: 20/10/2025

Link al repositorio de GitHub:

https://github.com/SamuelCFlores/LIMPIEZA_DATOS

2. Descripción inicial de la base de datos

Contexto inicial

Data set procedente de una plataforma de casino o apuestas online que registra actividad de gaming, transacciones y resultados de partidas. Contiene aproximadamente 65,000 registros de sesiones de juego.

para mi objetivo final demostraré cual es el verdadero margen de ganancia del casino, aun sabiendo que su manera de operar es diferente a la de los casinos en línea convencionales

Descripción general

El data set es un registro de apuestas registradas en un casino en línea con libre socket, donde los principales datos son el ID del juego, el coeficiente sobre el que se apuesta (tick), y el pago (outpay), para mi objetivo final demostraré cual es el verdadero margen de ganancia del casino, aun sabiendo que su manera de operar es diferente a la de los casinos en línea convencionales

Significado por columna

Campo	Descripción	Ejemplo
ID	Identificador único de cada juego.	-
players	Número de jugadores en el juego.	4 jugadores participan
items		100 items apostados

	Cantidad de objetos apostados por los jugadores en el juego.	
money	Equivalente de los objetos en dinero real (dólares).	Valor del item
ticks	Coeficiente del sitio para este juego.	Coeficiente : 1.8
peopleLost	Cantidad de dinero que las personas perdieron en este juego (dólares).	$100 x (1.8 -1) = 80$
outpay	Cantidad de dinero que las personas reciben de este juego (dólares). Calculado como su apuesta multiplicada por el coeficiente.	$100 x 1.8 180$
time	Fecha y hora en que ocurrió el juego (formato YY-MM-DD hh-mm).	Fecha y hora
moderator		False o true

	Indica si el moderador participó en el juego.	

3. Proceso de limpieza

Inicio cargando el data set sucio

```
import pandas as pd
import numpy as np
```
[318] ✓ 0.0s Python

```
df = pd.read_csv('df_sucio.csv')
df
```
[319] ✓ 0.2s Python

	ID	gamers	skins	money	ticks	peopleWin	peopleLost	outpay	time	moderator
0	NaN	144.0	174.0	283.57000	14.30	125.459984	0.25	408.78	2021-08-25 16:43	False
1	2091104.0	134.0	182.0	279.30000	1.14	5.810003	177.04	108.07001	2021-08-25 16:44	False
2	2091105.0	139.0	179.0	282.87000	3.91	125.01	5.96	401.91992	2021-08-25 16:44	False
3	2091106.0	139.0	169.0	271.44000	NaN	6.900001	181.85	96.490005	2021-08-25 16:45	False
4	2091107.0	142.0	177.0	304.88000	1.00	0.0	304.88	0.0	2021-08-25 16:45	False
...	...	...	...	...	...	...	...	...	...	...
65010	2143256.0	133.0	163.0	277.65002	7.10	117.300026	5.17	474.36993	2021-09-13 15:51	False
65011	2114948.0	109.0	148.0	247.16994	1.23	20.520004	81.36001	289.3299	2021-09-03 07:40	False
65012	2132428.0	116.0	158.0	299.85000	1.11	6.2600007	Auto%#	178.95004	2021-09-09 16:42	False
65013	2131341.0	NaN	144.0	185.78000	2.15	50.739986	NaN	308.36002	2021-09-09 07:14	False
65014	2125051.0	91.0	118.0	167.83000	1.29	7.430003	89.06002	192.12001	2021-09-07 00:00	False

65015 rows × 10 columns

```
# COPIA DE SEGURIDAD DEL DATAFRAME
df_respaldo = df.copy()
```
[320]

Hice un “diagnóstico” general o analisis para poder filtrar y tener un orden para limpiar la base (datos faltantes)

```
# 0 - IDENTIFICAR o hacer DIAGNÓSTICO general

Para este data base:
--VALORES NULOS
--VALORES INVALIDOS
```
markdown

```
# INFORMACIÓN GENERAL, para hacer un diagnóstico inicial
df.info()
```
[322] ✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65015 entries, 0 to 65014
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   ID          61803 non-null  float64
 1   gamers      63065 non-null  float64
 2   skins       63065 non-null  float64
 3   money       63065 non-null  float64
 4   ticks       63065 non-null  float64
 5   peopleWin   63065 non-null  object
 6   peopleLost  63065 non-null  object
 7   outpay      63065 non-null  object
 8   time        63055 non-null  object
 9   moderator   63065 non-null  object
dtypes: float64(5), object(5)
memory usage: 5.0+ MB
```

Seguimiento del análisis

```python
#Con este ciclo for es posible identificar la cantidad de auto%# y cuales son los valores en cada columna

lista_col= df.columns
for p in lista_col:
    print(f"En la Columna {p} hay: ")
    print(f"Valor: Auto%# hay: {df[df[p] == 'Auto%#'].shape[0]} ")
    print(f"Hay {df[p].nunique()} valores en la columna")
    print(df[p].unique())
    print(f"_____")
```

Salida del código: Nos damos cuenta de que para los valores str, no hay en columnas numéricas pero si en las que son tipo object

```
En la Columna money hay:
Valor: Auto%# hay: 0
Hay 47198 valores en la columna
[283.57    279.3     282.87    ... 652.44995 317.86008 291.54993]

_____
En la Columna ticks hay:
Valor: Auto%# hay: 0
Hay 3904 valores en la columna
[ 14.3    1.14   3.91 ...  18.88 179.1  185.9 ]

_____
En la Columna peopleWin hay:
Valor: Auto%# hay: 1271
Hay 42160 valores en la columna
['125.459984' '5.810003' '125.01' ... '39.11' '19.270002' '140.19003']

_____
En la Columna peopleLost hay:
Valor: Auto%# hay: 1261
Hay 29294 valores en la columna
['0.25' '177.04' '5.96' ... '167.38997' '116.25' '87.15001']

_____
En la Columna outpay hay:
Valor: Auto%# hay: 1258
Hay 48106 valores en la columna
['408.78' '108.07001' '401.91992' ... '386.32' '291.08005' '209.53001']

_____
En la Columna time hay:
Valor: Auto%# hay: 1270
Hay 27787 valores en la columna
['2021-08-25 16:43' '2021-08-25 16:44' '2021-08-25 16:45' ...
 '2021-08-26 19:09' '2021-08-27 18:54' '2021-09-11 07:38']
```

```
_____________________________
En la Columna moderator hay:
Valor: Auto%# hay: 0
Hay 2 valores en la columna
[False nan True]
```

Seguimiento

Muestra los valores nulos (NaN) para cada una de las columnas del data frame original

```python
# CICLO FOR PARA IDENTIFICAR VALORES NULOS DE CADA COLUMNA

for columna in df.columns:
    nulos = df[columna].isnull().sum()
    print(f'Columna {columna} tiene {nulos} valores nulos')
```

```
Columna ID tiene 3212 valores nulos
Columna gamers tiene 1950 valores nulos
Columna skins tiene 1950 valores nulos
Columna money tiene 1950 valores nulos
Columna ticks tiene 1950 valores nulos
Columna peopleWin tiene 1950 valores nulos
Columna peopleLost tiene 1950 valores nulos
Columna outpay tiene 1950 valores nulos
Columna time tiene 1960 valores nulos
Columna moderator tiene 1950 valores nulos
```

Detecta los valores de cada columna: Aunque sean variados y no todos necesariamente “inconsistentes” nos da una pista de lo que se tiene que hacer

```python
# CICLO FOR PARA IDENTIFICAR VALORES UNICOS DE CADA COLUMNA
for columna in df.columns:
    unicos = df[columna].nunique()
    print(f'Columna {columna} tiene {unicos} valores únicos')
```

```
Columna ID tiene 53272 valores únicos
Columna gamers tiene 273 valores únicos
Columna skins tiene 308 valores únicos
Columna money tiene 47198 valores únicos
Columna ticks tiene 3904 valores únicos
Columna peopleWin tiene 42160 valores únicos
Columna peopleLost tiene 29294 valores únicos
Columna outpay tiene 48106 valores únicos
Columna time tiene 27787 valores únicos
Columna moderator tiene 2 valores únicos
```

Reemplazando los valores nulos, dentro de las columnas numéricas: Se usa una lista de las columnas numéricas exceptuando el ID (por motivos de coherencia) para después reemplazar los valores nulos por el promedio evitando distorsionar tanto los datos, de igual forma se redondea para evitar tantos decimales

```python
# CREAR COPIA DEL DATAFRAME ORIGINAL
columnas_numericas_filtradas = ['gamers', 'skins', 'money', 'ticks']
df_filtro1_numericas = df.copy()

# Aplicar los cambios solo al nuevo DataFrame
for columna in columnas_numericas_filtradas:
    promedio = df_filtro1_numericas[columna].mean()
    promedio_redondeado = round(promedio, 2) # (ROUND) Redondea el promedio para mantener consistencia y evitar decimales largos
    df_filtro1_numericas[columna] = df_filtro1_numericas[columna].fillna(promedio_redondeado)
    print(f'Para columna: {columna} /// Valores nulos reemplazados por: {promedio_redondeado}')
```

Salida del código y verificación de los cambios

```
Para columna: gamers /// Valores nulos reemplazados por: 123.05
Para columna: skins /// Valores nulos reemplazados por: 158.3
Para columna: money /// Valores nulos reemplazados por: 284.82
Para columna: ticks /// Valores nulos reemplazados por: 11.38
```

NOTA: La columna ID no corresponde a este proceso aun si es númerica debido a que este se ve determinado en el momento y no puede ser reemplazado en este caso por el promedio

```
# Verificación de forma visual de los de los registros finales después del reemplazo de valores nulos en las columnas númericas
df_filtro1_numericas.tail()
```

✓ 0.0s Python

	ID	gamers	skins	money	ticks	peopleWin	peopleLost	outpay	time	moderator
65010	2143256.0	133.00	163.0	277.65002	7.10	117.300026	5.17	474.36993	2021-09-13 15:51	False
65011	2114948.0	109.00	148.0	247.16994	1.23	20.520004	81.36001	289.3299	2021-09-03 07:40	False
65012	2132428.0	116.00	158.0	299.85000	1.11	6.2600007	Auto%#	178.95004	2021-09-09 16:42	False
65013	2131341.0	123.05	144.0	185.78000	2.15	50.739986	NaN	308.36002	2021-09-09 07:14	False
65014	2125051.0	91.00	118.0	167.83000	1.29	7.430003	89.06002	192.12001	2021-09-07 00:00	False

```
# VEERICACIÓN contundente de VALORES NULOS después del reemplazo en el nuevo Data Frame

for columna in df_filtro1_numericas.columns:
    nulos = df_filtro1_numericas[columna].isnull().sum()
    print(f'Columna {columna} tiene {nulos} valores nulos')
```

[328] ✓ 0.0s

```
Columna ID tiene 3212 valores nulos
Columna gamers tiene 0 valores nulos
Columna skins tiene 0 valores nulos
Columna money tiene 0 valores nulos
Columna ticks tiene 0 valores nulos
Columna peopleWin tiene 1950 valores nulos
Columna peopleLost tiene 1950 valores nulos
Columna outpay tiene 1950 valores nulos
Columna time tiene 1960 valores nulos
Columna moderator tiene 1950 valores nulos
```

Se puede ver que este “filtro” se guardó en un nuevo df para no trabajar sobre el original y ser más práctico (corrección de datos, isnull()

NOTAS IMPORTANTES:

```
NOTA: (Suponiendo que se trata de un caso real) Se encuentra que el ID del usuario DEBE ser determinado por el juego que
estaba en ocurriendo en ese instante, por lo que en la "base de datos sucia" no hay forma de demostar cual era en realidad
ese juego y seria un error del casino al tener inconsistencias en los registros, por lo que se opta de tratar a este tipo de
"juego" con ID nulo como juego/sesión tipo "Demo" (simulación) cosa que llega a ser común en casinos y se tomaría comom una
forma incompleta de llenar los registros de los datos. Si se trata de un casino "profesional" las demo tienen las mismas
probabilidades de ganar o perder que los demas juegos

Por lo que: Caso 1: ERROR INTERNO en el registro de sesiones del casino
            Caso 2: Razones "Reales" o fallos en la página: Sesiones incompletas, usuario sin conexión, almacenamiento local
            (cache), pruebas internas
            Caso 3: Sesión tipo "DEMO" o simulaciones (Caso a utilizar)
```

Reemplazando

Se reemplazan los valores en el nuevo df o “filtro” (corrección de datos .fillna())

Antes hecho y guardado, por la palabra “demo” esto explicado a profundidad en la nota anterior, donde ahora se toma como una sesión simulada

```
#REEMPLAZANDO los valores nulos para los ID por "Invitado"

df_filtro2_id = df_filtro1_numericas.copy()

df_filtro2_id['ID'] = df_filtro2_id['ID'].fillna('Demo')
df_filtro2_id
```

✓ 0.0s

	ID	gamers	skins	money	ticks	peopleWin	peopleLost	outpay	time	moderator
0	Demo	144.00	174.0	283.57000	14.30	125.459984	0.25	408.78	2021-08-25 16:43	False
1	2091104.0	134.00	182.0	279.30000	1.14	5.810003	177.04	108.07001	2021-08-25 16:44	False
2	2091105.0	139.00	179.0	282.87000	3.91	125.01	5.96	401.91992	2021-08-25 16:44	False
3	2091106.0	139.00	169.0	271.44000	11.38	6.900001	181.85	96.490005	2021-08-25 16:45	False
4	2091107.0	142.00	177.0	304.88000	1.00	0.0	304.88	0.0	2021-08-25 16:45	False
...	...	...	...	...	...	...	...	...	...	...
65010	2143256.0	133.00	163.0	277.65002	7.10	117.300026	5.17	474.36993	2021-09-13 15:51	False
65011	2114948.0	109.00	148.0	247.16994	1.23	20.520004	81.36001	289.3299	2021-09-03 07:40	False
65012	2132428.0	116.00	158.0	299.85000	1.11	6.2600007	Auto%#	178.95004	2021-09-09 16:42	False
65013	2131341.0	123.05	144.0	185.78000	2.15	50.739986	NaN	308.36002	2021-09-09 07:14	False
65014	2125051.0	91.00	118.0	167.83000	1.29	7.430003	89.06002	192.12001	2021-09-07 00:00	False

65015 rows × 10 columns

Verificación

```
#Para más practicidad se vuelven a verificar los valores nulos en todas las columnas del último Data Frame que es en el que se trabajará
# CICLO FOR PARA IDENTIFICAR VALORES NULOS DE CADA COLUMNA

for columna in df_filtro2_id.columns:
    nulos = df_filtro2_id[columna].isnull().sum()
    print(f'Columna {columna} tiene {nulos} valores nulos')
```

✓ 0.0s

```
Columna ID tiene 0 valores nulos
Columna gamers tiene 0 valores nulos
Columna skins tiene 0 valores nulos
Columna money tiene 0 valores nulos
Columna ticks tiene 0 valores nulos
Columna peopleWin tiene 1950 valores nulos
Columna peopleLost tiene 1950 valores nulos
Columna outpay tiene 1950 valores nulos
Columna time tiene 1960 valores nulos
Columna moderator tiene 1950 valores nulos
```

De antes se sabía que las columnas numéricas no contenían valores tipo string por lo

que ahora se busca reemplazarlos por valores NaN para un manejo más fácil, usando de guía que los valores son numéricos y pueden ser transformados en tipo float. (Conversión de datos)

En la segunda imagen se crea un tercer filtro esta vez para los valores tipo string

```
# REEMPLZANDO los tipo string
encontrar_str = ['peopleWin', 'peopleLost','outpay', 'time', 'moderator']
for columna in encontrar_str:
    df_filtro2_id[columna] = df_filtro2_id[columna].replace(['Auto%#'], np.nan)
    # Los valores unicos se reemplazan por NaN para un mejor manejo de datos y se les asigna valor NaN
```

✓ 0.0s

```
#Verificación de los valores únicos después del reemplazo
df_filtro3_str = df_filtro2_id.copy()

lista_col2 = ['peopleWin', 'peopleLost','outpay', 'time', 'moderator'] # Se tomaron los valores que habian demostrado tener valores tipo objeto (string)
for p in lista_col2:
    print(f"En la Columna {p} hay: ")
    print(f"Valor: Auto%# hay: {df_filtro3_str[df_filtro3_str[p] == 'Auto%#'].shape[0]} ")
    print(f"Hay {df_filtro3_str[p].nunique()} valores en la columna")
    print(df_filtro3_str[p].unique())
    print(f"_"*25)
```

✓ 0.0s

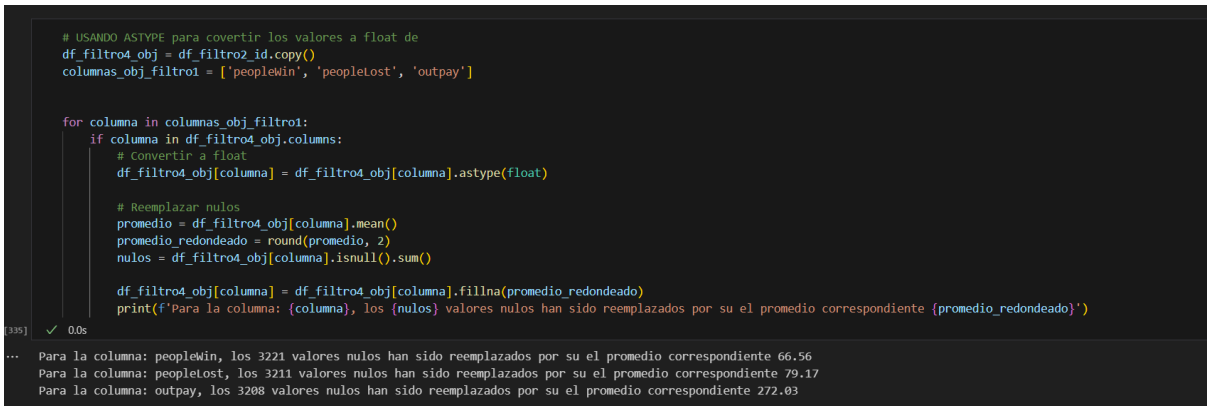
Salida del último código, con esto comprobamos que los valores tipo string ya han sido reemplazados por valores NaN

```
En la Columna peopleWin hay:
Valor: Auto%# hay: 0
Hay 42159 valores en la columna
['125.459984' '5.810003' '125.01' ... '39.11' '19.270002' '140.19003']
_________________________
En la Columna peopleLost hay:
Valor: Auto%# hay: 0
Hay 29293 valores en la columna
['0.25' '177.04' '5.96' ... '167.38997' '116.25' '87.15001']
_________________________
En la Columna outpay hay:
Valor: Auto%# hay: 0
Hay 48105 valores en la columna
['408.78' '108.07001' '401.91992' ... '386.32' '291.08005' '209.53001']
_________________________
En la Columna time hay:
Valor: Auto%# hay: 0
Hay 27786 valores en la columna
['2021-08-25 16:43' '2021-08-25 16:44' '2021-08-25 16:45' ...
 '2021-08-26 19:09' '2021-08-27 18:54' '2021-09-11 07:38']
_________________________
En la Columna moderator hay:
Valor: Auto%# hay: 0
Hay 2 valores en la columna
[False nan True]
_________________________
```

Después de todos los cambios se crea un cuarto filtro para cambiar donde antes había valores tipo string que posteriormente fueron cambiados por valores NaN, serán

reemplazados por el promedio de cada columna correspondiente, se elige el promedio ya que no distorsionaría el análisis de datos a la hora de buscar el objetivo de mi proyecto

Se puede observar el cambio de tipo de columna usando .astype() , paso clave en este paso



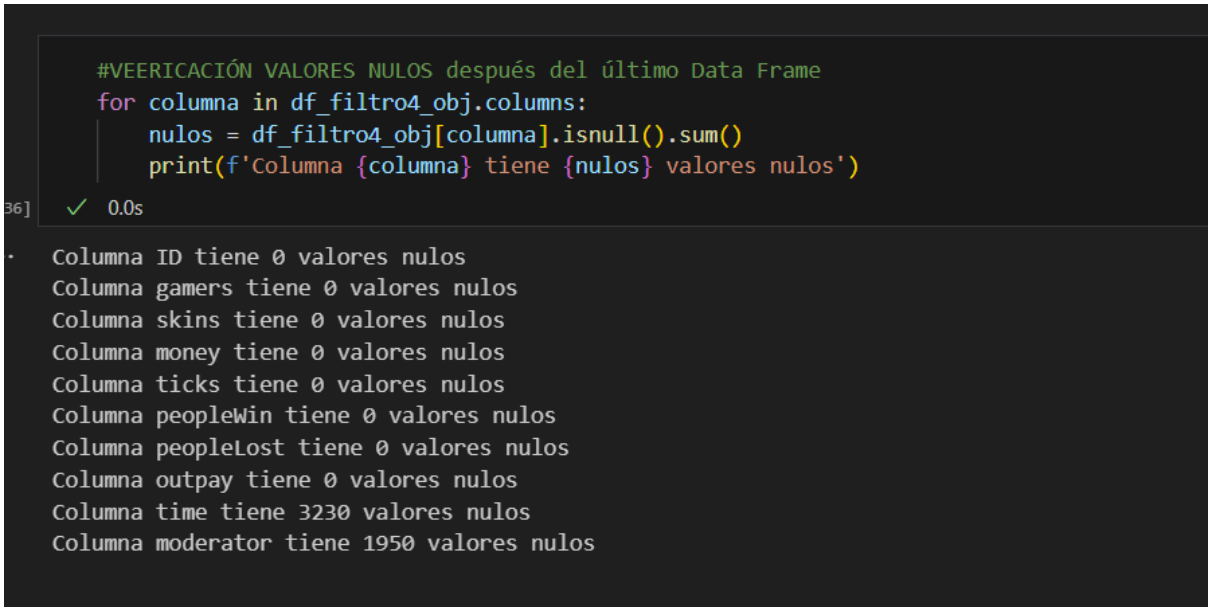
```
# USANDO ASTYPE para covertir los valores a float de
df_filtro4_obj = df_filtro2_id.copy()
columnas_obj_filtro1 = ['peopleWin', 'peopleLost', 'outpay']


for columna in columnas_obj_filtro1:
    if columna in df_filtro4_obj.columns:
        # Convertir a float
        df_filtro4_obj[columna] = df_filtro4_obj[columna].astype(float)

        # Reemplazar nulos
        promedio = df_filtro4_obj[columna].mean()
        promedio_redondeado = round(promedio, 2)
        nulos = df_filtro4_obj[columna].isnull().sum()

        df_filtro4_obj[columna] = df_filtro4_obj[columna].fillna(promedio_redondeado)
        print(f'Para la columna: {columna}, los {nulos} valores nulos han sido reemplazados por su el promedio correspondiente {promedio_redondeado}')
[335]  ✓ 0.0s
...  Para la columna: peopleWin, los 3221 valores nulos han sido reemplazados por su el promedio correspondiente 66.56
     Para la columna: peopleLost, los 3211 valores nulos han sido reemplazados por su el promedio correspondiente 79.17
     Para la columna: outpay, los 3208 valores nulos han sido reemplazados por su el promedio correspondiente 272.03
```

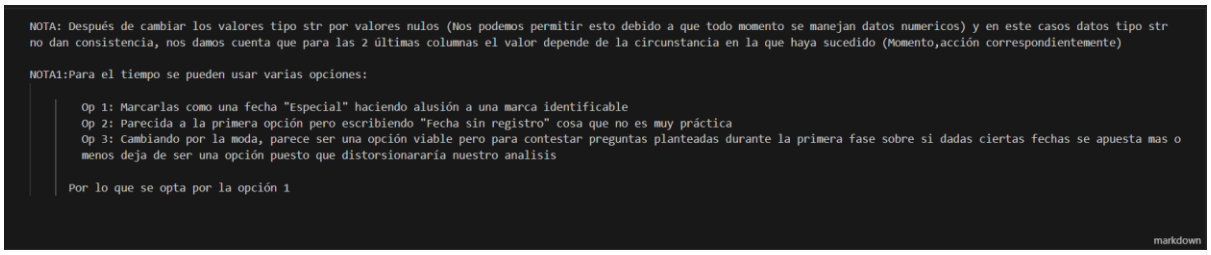
Nuevamente se verifican que los cambios se hayan hecho correctamente y guardado en el nuevo ”filtro”



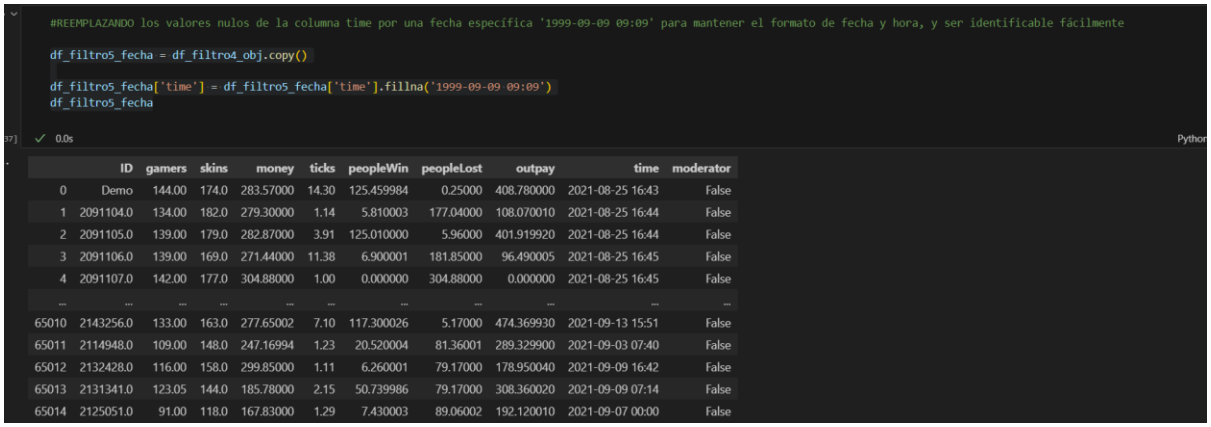
```
#VEERICACIÓN VALORES NULOS después del último Data Frame
for columna in df_filtro4_obj.columns:
    nulos = df_filtro4_obj[columna].isnull().sum()
    print(f'Columna {columna} tiene {nulos} valores nulos')
36]  ✓ 0.0s
·  Columna ID tiene 0 valores nulos
   Columna gamers tiene 0 valores nulos
   Columna skins tiene 0 valores nulos
   Columna money tiene 0 valores nulos
   Columna ticks tiene 0 valores nulos
   Columna peopleWin tiene 0 valores nulos
   Columna peopleLost tiene 0 valores nulos
   Columna outpay tiene 0 valores nulos
   Columna time tiene 3230 valores nulos
   Columna moderator tiene 1950 valores nulos
```

Limpieza para datos independientes

NOTA IMPORTANTE:



Para este paso: Se muestra la solución al problema planteado por una inconsistencia de datos , nuevamente guardando los cambios en un nuevo filtro asegurando un retorno en caso de error (Corrección de valores inconsistentes)



	ID	gamers	skins	money	ticks	peopleWin	peopleLost	outpay	time	moderator
0	Demo	144.00	174.0	283.57000	14.30	125.459984	0.25000	408.780000	2021-08-25 16:43	False
1	2091104.0	134.00	182.0	279.30000	1.14	5.810003	177.04000	108.070010	2021-08-25 16:44	False
2	2091105.0	139.00	179.0	282.87000	3.91	125.010000	5.96000	401.919920	2021-08-25 16:44	False
3	2091106.0	139.00	169.0	271.44000	11.38	6.900001	181.85000	96.490005	2021-08-25 16:45	False
4	2091107.0	142.00	177.0	304.88000	1.00	0.000000	304.88000	0.000000	2021-08-25 16:45	False
...	...	...	...	...	...	...	...	...	...	...
65010	2143256.0	133.00	163.0	277.65002	7.10	117.300026	5.17000	474.369930	2021-09-13 15:51	False
65011	2114948.0	109.00	148.0	247.16994	1.23	20.520004	81.36001	289.329900	2021-09-03 07:40	False
65012	2132428.0	116.00	158.0	299.85000	1.11	6.260001	79.17000	178.950040	2021-09-09 16:42	False
65013	2131341.0	123.05	144.0	185.78000	2.15	50.739986	79.17000	308.360020	2021-09-09 07:14	False
65014	2125051.0	91.00	118.0	167.83000	1.29	7.430003	89.06002	192.120010	2021-09-07 00:00	False

Nota importante para entender mi decisión sobre el reemplazo de los valores

Podemos notar que hay más valores falsos que verdaderos, y dentro del rango máximo el veradero representa un 5.2% aproximadamente del total de datos, por lo que nos lleva a elegir a falso como la moda de esta columna

```
# Reemplazando los valores nulos de la columna moderator por la moda de la columna
df_filtro6_mod = df_filtro5_fecha.copy()

df_filtro6_mod['moderator'] = df_filtro6_mod['moderator'].fillna('False')
df_filtro6_mod
```

✓ 0.0s

	ID	gamers	skins	money	ticks	peopleWin	peopleLost	outpay	time	moderator
0	Demo	144.00	174.0	283.57000	14.30	125.459984	0.25000	408.780000	2021-08-25 16:43	False
1	2091104.0	134.00	182.0	279.30000	1.14	5.810003	177.04000	108.070010	2021-08-25 16:44	False
2	2091105.0	139.00	179.0	282.87000	3.91	125.010000	5.96000	401.919920	2021-08-25 16:44	False
3	2091106.0	139.00	169.0	271.44000	11.38	6.900001	181.85000	96.490005	2021-08-25 16:45	False
4	2091107.0	142.00	177.0	304.88000	1.00	0.000000	304.88000	0.000000	2021-08-25 16:45	False
...	...	...	...	...	...	...	...	...	...	...
65010	2143256.0	133.00	163.0	277.65002	7.10	117.300026	5.17000	474.369930	2021-09-13 15:51	False
65011	2114948.0	109.00	148.0	247.16994	1.23	20.520004	81.36001	289.329900	2021-09-03 07:40	False
65012	2132428.0	116.00	158.0	299.85000	1.11	6.260001	79.17000	178.950040	2021-09-09 16:42	False
65013	2131341.0	123.05	144.0	185.78000	2.15	50.739986	79.17000	308.360020	2021-09-09 07:14	False
65014	2125051.0	91.00	118.0	167.83000	1.29	7.430003	89.06002	192.120010	2021-09-07 00:00	False

65015 rows × 10 columns

Se reemplaza por los valores NaN que se cambiaron anteriormente usando los filtros de manera correcta así llevando un orden entre datos y filtros

Ya para terminar se vuelven a verificar o se hace el diagnostico nuevamente del dat set

Cumpliendo con la verificación o validación final de los datos

```
# DIAGNOSTICO FINAL

df_filtro6_mod.info()

print("\n"+"-"*30+"\n")

df_filtro6_mod.describe()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65015 entries, 0 to 65014
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   ID          65015 non-null  object
 1   gamers      65015 non-null  float64
 2   skins       65015 non-null  float64
 3   money       65015 non-null  float64
 4   ticks       65015 non-null  float64
 5   peopleWin   65015 non-null  float64
 6   peopleLost  65015 non-null  float64
 7   outpay      65015 non-null  float64
 8   time        65015 non-null  object
 9   moderator   65015 non-null  object
dtypes: float64(7), object(3)
memory usage: 5.0+ MB

------------------------------
```

	gamers	skins	money	ticks	peopleWin	peopleLost	outpay
count	65015.000000	65015.000000	65015.000000	65015.000000	65015.000000	65015.000000	65015.000000
mean	123.046351	158.303484	284.823779	11.376624	66.555495	79.173404	272.032278
std	25.755042	30.480599	191.009964	223.254983	109.422823	180.569355	211.921156
min	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	103.000000	136.000000	216.635025	1.310000	21.545003	3.760000	166.189990
50%	122.000000	156.000000	257.529970	2.010000	62.270016	36.159996	272.030000
75%	137.000000	174.000000	306.030030	4.270000	95.659973	105.349990	355.654900
max	491.000000	552.000000	5696.327000	23522.650000	21681.592000	4573.601600	25943.201000

Notamos que los valores toman sentido, no hay columnas que tengan más o menos valores, el margen de ganancia del casino sigue siendo positivo.

Sondeo para valores nulosss

```python
# DIAGNOSTICO FINAL

#VALORES NULOS
for columna in df_filtro6_mod.columns:
    nulos = df_filtro6_mod[columna].isnull().sum()
    print(f'Columna {columna} tiene {nulos} valores nulos')

print("\n"+"-"*30+"\n")

df_filtro6_mod_cols = df_filtro6_mod.columns # Se tomaron los valores que habian demostrado tener valores tipo objeto (string)
for p in df_filtro6_mod_cols:
    print(f"En la Columna {p} hay: ")
    print(f"Valor: Auto%# hay: {df_filtro3_str[df_filtro3_str[p] == 'Auto%#'].shape[0]} ")
    print(f"Hay {df_filtro3_str[p].nunique()} valores en la columna")
    print(df_filtro3_str[p].unique())
    print(f"_"*25)
```

Salida del código:

```
Columna ID tiene 0 valores nulos
Columna gamers tiene 0 valores nulos
Columna skins tiene 0 valores nulos
Columna money tiene 0 valores nulos
Columna ticks tiene 0 valores nulos
Columna peopleWin tiene 0 valores nulos
Columna peopleLost tiene 0 valores nulos
Columna outpay tiene 0 valores nulos
Columna time tiene 0 valores nulos
Columna moderator tiene 0 valores nulos

------------------------------

En la Columna ID hay:
Valor: Auto%# hay: 0
Hay 53273 valores en la columna
['Demo' 2091104.0 2091105.0 ... 2097953.0 2103773.0 2131341.0]
_________________________
En la Columna gamers hay:
Valor: Auto%# hay: 0
Hay 274 valores en la columna
[144.   134.   139.   142.   161.   157.   160.   155.   129.   122.
 127.   121.   125.   118.   120.   128.   117.   119.   116.   123.05
 126.   113.   123.   141.   130.   137.   159.   149.   138.   197.
 164.   167.   156.   145.   140.   148.   154.   153.   172.   158.
 146.   152.   110.   143.   133.   136.   151.   115.   150.   132.
 131.   163.   187.   185.   191.   182.   165.   135.   124.   184.
 170.   205.   169.   114.   147.   201.   177.   292.   234.   209.
 180.   219.   194.   193.   222.   217.   212.   190.   189.   223.
 229.   192.   252.   225.   270.   250.   314.   232.   221.   207.
```

```
_____________________________
En la Columna moderator hay:
Valor: Auto%# hay: 0
Hay 2 valores en la columna
[False nan True]
_____________________________
```

```
_____________________________
En la Columna money hay:
Valor: Auto%# hay: 0
Hay 47198 valores en la columna
[283.57    279.3     282.87    ... 652.44995 317.86008 291.54993]
_____________________________
En la Columna ticks hay:
Valor: Auto%# hay: 0
Hay 3904 valores en la columna
[ 14.3    1.14   3.91 ...  18.88 179.1  185.9 ]
_____________________________
En la Columna peopleWin hay:
Valor: Auto%# hay: 0
Hay 42159 valores en la columna
['125.459984' '5.810003' '125.01' ... '39.11' '19.270002' '140.19003']
_____________________________
En la Columna peopleLost hay:
Valor: Auto%# hay: 0
Hay 29293 valores en la columna
['0.25' '177.04' '5.96' ... '167.38997' '116.25' '87.15001']
_____________________________
En la Columna outpay hay:
Valor: Auto%# hay: 0
Hay 48105 valores en la columna
['408.78' '108.07001' '401.91992' ... '386.32' '291.08005' '209.53001']
_____________________________
```

De último paso se traducen las columnas para cuando sean trabajadas después sea más versátil su manejo y se guarda el Data Frame como nuevo csv (Traducción de textos)

```python
#TRADUCCIÓN AL ESPAÑOL DE LOS NOMBRES DE LAS COLUMNAS
df_filtro6_mod.rename(columns={
    'ID': 'Identificación',
    'gamers': 'Jugadores',
    'skins': 'Apariencias',
    'money': 'Dinero',
    'peopleWin': 'PersonasGanaron',
    'peopleLost': 'PersonasPerdieron',
    'outpay': 'PagoTotal',
    'ticks': 'TiemposDeJuego',
    'time': 'FechaYHora',
    'moderator': 'ModeradorActivo'
}, inplace=True)
```

✓ 0.0s

```python
# Para guardar el Data Frame limpio en un nuevo archivo CSV
df_filtro5_fecha.to_csv('casino_datos_limpios.csv', index=False)
```

✓ 0.4s

4. Conclusiones

Que problemas se presentaron: Los primeros problemas fueron a la hora de saber como empezar a trabajar, la manera en la que me tenía que organizar yo y mi forma de limpiar la base de datos, después reemplazando los datos había unos que no podían ser reemplazados por otros tan fácil pues son datos independientes por lo que opte darle una explicación “real” de lo que pudieron haber sucedido y asi encontrando la forma de no distorsionar los datos y no complicar el análisis que se hará después

Técnicas para solución de problemas: Primeramente investigué sobre como operaban casinos reales y como manejaban los registros, después aumenté mi conocimiento en lo que se trata de estructuras y problemas de lógica.

Que se aprendió: Aprendí que todo lleva un orden y dentro de ese orden se requiere más orden en los pasos y paciencia