UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

MAESTRÍA EN CIENCAS DE DATOS

BASES DE DATOS RELACIONALES

MAESTRO: JOSÉ ALBERTO BENAVIDES VÁZQUEZ

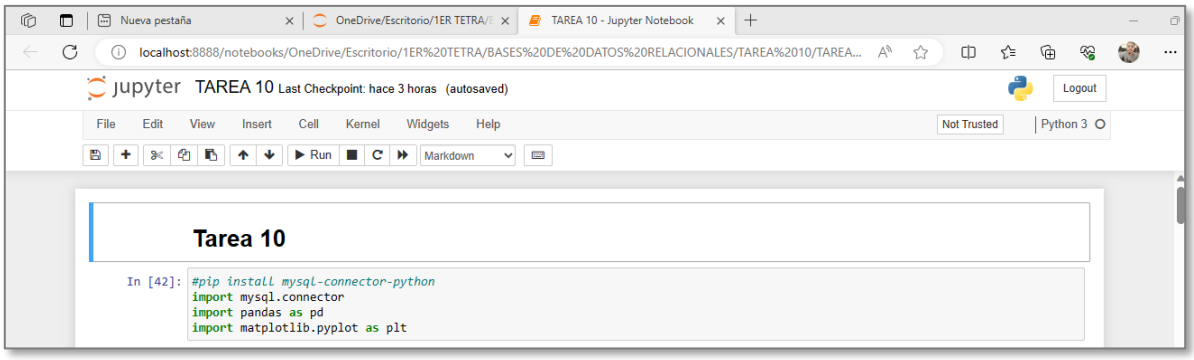
TAREA #10

ALUMNO: EDWIN MARTÍN ROMERO SILVA

MATRÍCULA: 1731276

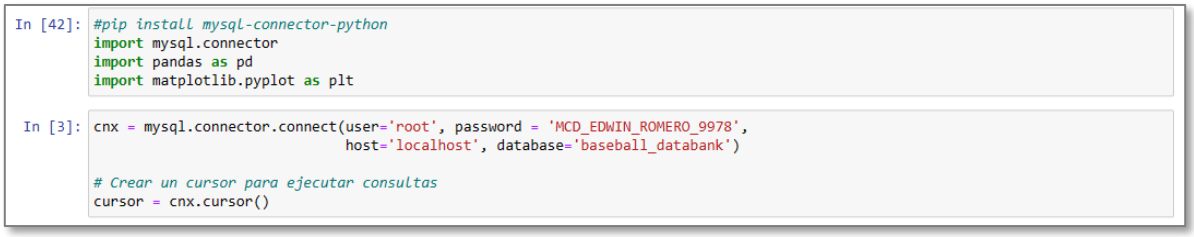
Tarea 10

1.- Crea un cuaderno de Jupyter



2.-Realiza un reporte donde:

a) Hagas una conexión desde Python a tu base de datos.



b) Carga tu base de datos en DataFrames de pandas.

	playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	deathMonth	deathDay	...	nameFirst	nameLast	nameGiven
0	aardsda01	1981.0	12.0	27.0	USA	CO	Denver	NaN	NaN	NaN	...	David	Aardsma	David Allan
1	aaronha01	1934.0	2.0	5.0	USA	AL	Mobile	NaN	NaN	NaN	...	Hank	Aaron	Henry Louis
2	aaronto01	1939.0	8.0	5.0	USA	AL	Mobile	1984.0	8.0	16.0	...	Tommie	Aaron	Tommie Lee
3	aasedo01	1954.0	9.0	8.0	USA	CA	Orange	NaN	NaN	NaN	...	Don	Aase	Donald William

Repetí el paso anterior en múltiples ocasiones, pero con otras tablas:

In [5]:
```
# Ejecutar una consulta
query = "SELECT * FROM batting"
cursor.execute(query)

# Obtener los resultados de la consulta
results = cursor.fetchall()

# Obtener los nombres de las columnas
column_names = [desc[0] for desc in cursor.description]

# Crear un DataFrame a partir de los resultados y los nombres de las columnas
batting = pd.DataFrame(results, columns=column_names)

# Mostrar el DataFrame
batting
```

Out[5]:

	playerID	yearID	stint	teamID	lgID	Games	AB	R	H	2B	...	BB	SO	IBB	HBP	SH	SF	GIDP	playerIDf	year_rangos	hr_ratio
0	aardsda01	2004	1	SFN	NL	11	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	aardsda01	7)>2000	0.000000
1	aardsda01	2006	1	CHN	NL	45	2.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	aardsda01	7)>2000	0.000000
2	aardsda01	2007	1	CHA	AL	25	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	aardsda01	7)>2000	0.000000

In [6]:
```
# Ejecutar una consulta
query = "SELECT * FROM pitching"
cursor.execute(query)

# Obtener los resultados de la consulta
results = cursor.fetchall()

# Obtener los nombres de las columnas
column_names = [desc[0] for desc in cursor.description]

# Crear un DataFrame a partir de los resultados y los nombres de las columnas
pitching = pd.DataFrame(results, columns=column_names)

# Mostrar el DataFrame
pitching
```

Out[6]:

	playerID	yearID	stint	teamID	lgID	W	L	G	GS	CG	...	WP	HBP	BK	BFP	GF	R	SH	SF	GIDP	playerIDf
0	aardsda01	2004	1	SFN	NL	1	0	11	0	0	...	0.0	2.0	0	61.0	5.0	8	0.0	1.0	NaN	aardsda01
1	aardsda01	2006	1	CHN	NL	3	0	45	0	0	...	1.0	1.0	0	225.0	9.0	25	1.0	3.0	NaN	aardsda01
2	aardsda01	2007	1	CHA	AL	2	1	25	0	0	...	2.0	1.0	0	151.0	7.0	24	2.0	1.0	NaN	aardsda01

In [35]:
```
# Ejecutar una consulta
query = "SELECT * FROM fielding"
cursor.execute(query)

# Obtener los resultados de la consulta
results = cursor.fetchall()

# Obtener los nombres de las columnas
column_names = [desc[0] for desc in cursor.description]

# Crear un DataFrame a partir de los resultados y los nombres de las columnas
fielding = pd.DataFrame(results, columns=column_names)

# Mostrar el DataFrame
fielding
```

Out[35]:

	playerID	yearID	stint	teamID	lgID	POS	G	GS	InnOuts	PO	A	E	DP	PB	WP	SB	CS	ZR	playerIDf
0	aardsda01	2004	1	SFN	NL	P	11	NaN	33.0	0.0	0.0	0.0	0.0	NaN	NaN	NaN	NaN	NaN	aardsda01
1	aardsda01	2006	1	CHN	NL	P	45	NaN	159.0	1.0	5.0	0.0	1.0	NaN	NaN	NaN	NaN	NaN	aardsda01
2	aardsda01	2007	1	CHA	AL	P	25	NaN	96.0	2.0	4.0	1.0	0.0	NaN	NaN	NaN	NaN	NaN	aardsda01

In [26]:
```
# Ejecutar una consulta
query = "SELECT * FROM teams"
cursor.execute(query)

# Obtener los resultados de la consulta
results = cursor.fetchall()

# Obtener los nombres de las columnas
column_names = [desc[0] for desc in cursor.description]

# Crear un DataFrame a partir de los resultados y los nombres de las columnas
teams = pd.DataFrame(results, columns=column_names)

# Mostrar el DataFrame
teams
```

Out[26]:

	yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	...	DP	FP	name	park	attendance	BPF	PPF	teamIDBR	teamIDla
0	1871	NA	BS1	BNA		3	31	NaN	20	10	...	NaN	0.830	Boston Red Stockings	South End Grounds I	NaN	103	98	BOS	
1	1871	NA	CH1	CNA		2	28	NaN	19	9	...	NaN	0.820	Chicago White Stockings	Union Base-Ball Grounds	NaN	104	102	CHI	
2	1871	NA	CL1	CFC		8	29	NaN	10	19	...	NaN	0.810	Cleveland Forest Citys	National Association Grounds	NaN	96	100	CLE	

c) Genere consultas relevantes y expliques su importancia.

d) Realices algunas operaciones de agregación directamente en Python.

e) Muestres resultados mediante gráficas.

Mostraré estos 3 incisos juntos ya que con un mismo código abarco los 3, genero las consultas incluyendo funciones de agregación y al final imprimo el gráfico que creo mas conveniente.

Primera consulta

Quise revisar el comportamiento de la cantidad de bateadores por equipo, a lo largo de los últimos 10 años. Nota. La tabla esta actualizada hasta 2015.

Primero a la tabla 'teams', la agrupé por año y calculé la función de agregación 'count' para saber cuantos equipos tengo por año. Noté que ese valor es constante, por año solo tenemos información de 30 equipos, me parece lógico, seguramente en el torneo solo participan 30 equipos y no es normal que el número varie.

Luego a la tabla 'batting', la agrupé por año y de igual forma le calculé un 'count' para saber cuantos bateadores tengo por año. Este valor no es constante.

Luego apliqué un merge con how = left, (esto es prácticamente lo mismo que un LEFT JOIN de sql) y uní las tablas.

Luego calculé Conteo_Bateadores/Conteo_Equipos.

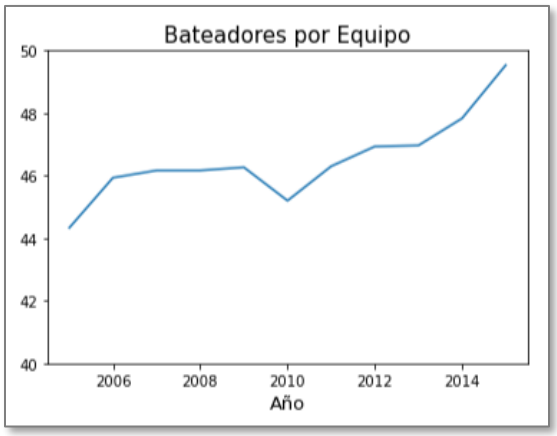
```
In [89]: #Conteo equipos
         equipos = teams[['yearID', 'teamID']].drop_duplicates()
         conteo_equipos = equipos.groupby(['yearID'])[['yearID']].count()
         conteo_equipos.columns = ['Conteo_Equipos']
         conteo_equipos = conteo_equipos.reset_index()
         conteo_equipos.columns = ['Año', 'Conteo_Equipos']
         conteo_equipos = conteo_equipos[conteo_equipos['Año'] >= 2005]

         #Conteo Bateadores
         bateadores = batting[['yearID', 'playerID']]
         conteo_bateadores = bateadores.groupby(['yearID'])[['yearID']].count()
         conteo_bateadores.columns = ['Conteo_Bateadores']
         conteo_bateadores = conteo_bateadores.reset_index()
         conteo_bateadores.columns = ['Año', 'Conteo_Bateadores']
         conteo_bateadores = conteo_bateadores[conteo_bateadores['Año'] >= 2005]

         bateadores_equipos = pd.merge(conteo_equipos, conteo_bateadores, how = 'left', on = 'Año' )
         bateadores_equipos['Bateadores/Equipos'] = bateadores_equipos['Conteo_Bateadores']/bateadores_equipos['Conteo_Equipos']

         plt.plot(bateadores_equipos['Año'], bateadores_equipos['Bateadores/Equipos'])
         plt.title('Bateadores por Equipo', size = 15)
         plt.xlabel('Año', size = 13)
         plt.ylim(40, 50)
```

Gráficamente podemos notar que el valor ha ido creciendo ligeramente, cada año los equipos cuentan con mas bateadores en su plantilla.



	Año	Conteo_Equipos	Conteo_Bateadores	Bateadores/Equipos
0	2005	30	1330	44.333333
1	2006	30	1378	45.933333
2	2007	30	1385	46.166667
3	2008	30	1385	46.166667
4	2009	30	1388	46.266667
5	2010	30	1356	45.200000
6	2011	30	1389	46.300000
7	2012	30	1408	46.933333
8	2013	30	1409	46.966667
9	2014	30	1435	47.833333
10	2015	30	1486	49.533333

Segunda consulta

Con esta consulta, quise analizar el comportamiento de los Doubles y de los HomeRuns a lo largo de los años, para ello utilicé 3 variables: Doubles, HomeRuns y year_rangos, esta última la cree en una tarea anterior.

Con la tabla de bateadores hice 1 vista, en donde calculé la función de agregación 'suma' para las 3 variables antes mencionadas. Para cada rango de año, sume el número de juegos, el número de doubles y el número de homeruns.

Posteriormente calculé con una división la cantidad de Doubles que realizaban por juego, es decir Suma_Dobles/Suma_Juegos.

Hice lo mismo con los homeruns: Suma_Homeruns/Suma_Juegos.

```
In [18]: sumas     = batting.groupby(['year_rangos'])[['Games', '2B', 'Homeruns']].sum()
         sumas['2B/Games'] = sumas['2B'] / sumas['Games']
         sumas['HR/Games'] = sumas['Homeruns'] / sumas['Games']
         sumas
```

Out[18]:

year_rangos	Games	2B	Homeruns	2B/Games	HR/Games
1)0-1900	382629	55511.0	9135.0	0.145078	0.023874
2)1901-1920	521197	61206.0	8008.0	0.117434	0.015365
3)1921-1940	554873	84600.0	24277.0	0.152467	0.043752
4)1941-1960	598227	70517.0	34375.0	0.117877	0.057461
5)1961-1980	948999	99084.0	56671.0	0.104409	0.059717
6)1981-2000	1165231	137503.0	76897.0	0.118005	0.065993
7)>2000	1037320	129952.0	74310.0	0.125277	0.071637

Luego mediante análisis gráfico podemos notar que la cantidad de homeruns por juego (o cada cuantos juegos se hace un homerun) se ha incrementado con el paso de los años, por lo que podríamos inferir que la técnica ha estado aumentando.

No se podría decir lo mismo sobre la cantidad de Doubles por juego, ya que incluso parece que este valor ha disminuido.

```
In [19]: plt.figure(figsize=(10, 5))
         plt.plot(sumas.index, sumas['HR/Games'], color = 'green', label = 'Homeruns/Games')
         plt.plot(sumas.index, sumas['2B/Games'], color = 'blue', label = 'Doubles/Games')
         plt.ylim(0,.2)
         plt.xlabel('Años', size = 13)
         plt.ylabel('Homeruns/Doubles por juego', size = 13)
         plt.title('Desempeño de los Bateadores', size = 20)
         plt.legend()
```

Out[19]: <matplotlib.legend.Legend at 0x2717935e4c0>