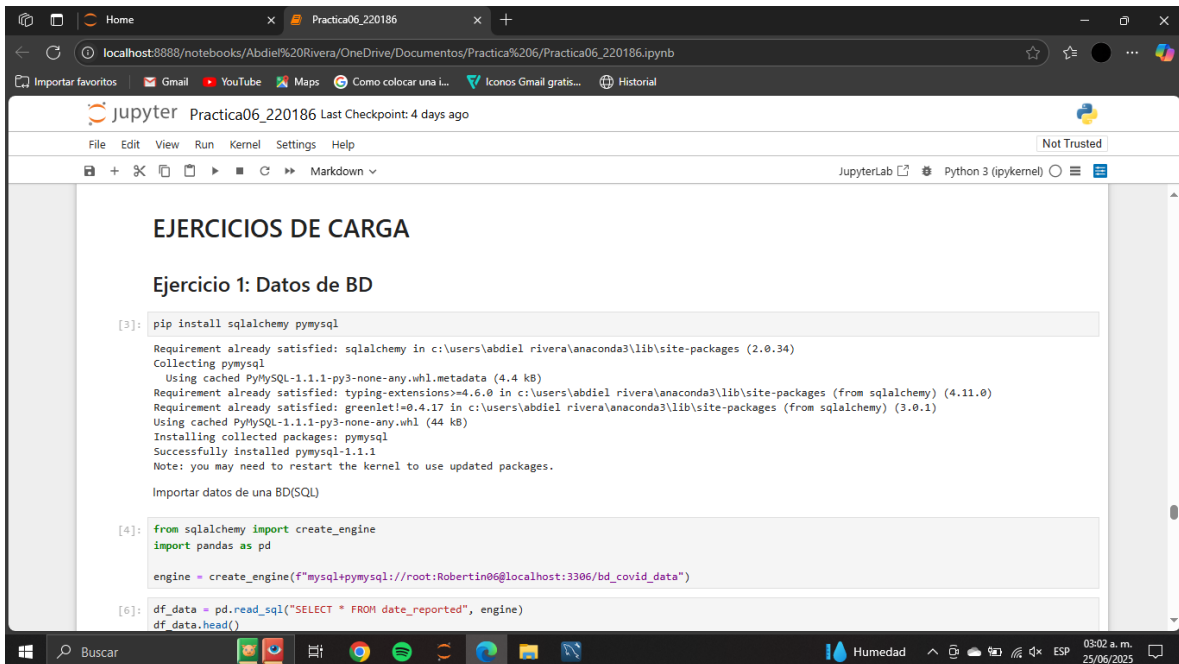


# Evidencias de la Tarea 05

## Ejercicio 1: Datos de una BD



The screenshot shows a Jupyter Notebook interface with the following code cells:

```
[3]: pip install sqlalchemy pymysql
```

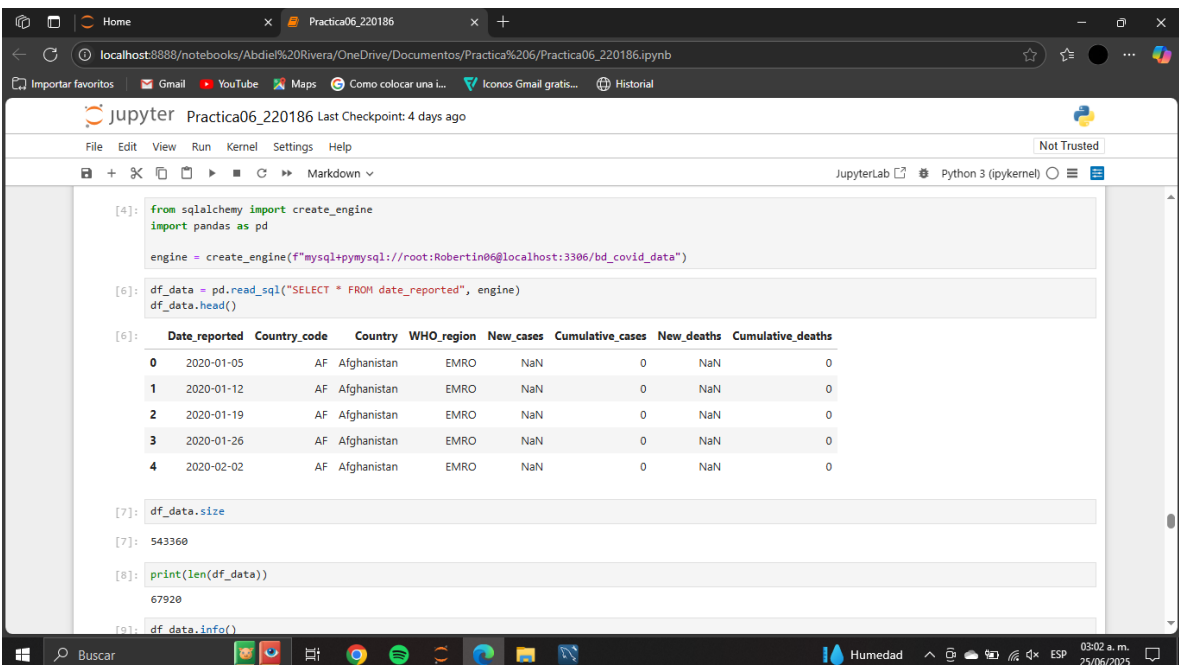
Requirement already satisfied: sqlalchemy in c:\users\abdiel rivera\anaconda3\lib\site-packages (2.0.34)  
Collecting pymysql  
Using cached PyMySQL-1.1.1-py3-none-any.whl.metadata (4.4 kB)  
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\abdiel rivera\anaconda3\lib\site-packages (from sqlalchemy) (4.11.0)  
Requirement already satisfied: greenlet!=0.4.17 in c:\users\abdiel rivera\anaconda3\lib\site-packages (from sqlalchemy) (3.0.1)  
Using cached PyMySQL-1.1.1-py3-none-any.whl (44 kB)  
Installing collected packages: pymysql  
Successfully installed pymysql-1.1.1  
Note: you may need to restart the kernel to use updated packages.

```
Importar datos de una BD(SQL)
```

```
[4]: from sqlalchemy import create_engine
import pandas as pd

engine = create_engine(f"mysql+pymysql://root:Robertin06@localhost:3306/bd_covid_data")
```

```
[6]: df_data = pd.read_sql("SELECT * FROM date_reported", engine)
df_data.head()
```



The screenshot shows the same Jupyter Notebook interface with the following code cells:

```
[4]: from sqlalchemy import create_engine
import pandas as pd

engine = create_engine(f"mysql+pymysql://root:Robertin06@localhost:3306/bd_covid_data")
```

```
[6]: df_data = pd.read_sql("SELECT * FROM date_reported", engine)
df_data.head()
```

```
[6]:
```

|   | Date_reported | Country_code | Country     | WHO_region | New_cases | Cumulative_cases | New_deaths | Cumulative_deaths |
|---|---------------|--------------|-------------|------------|-----------|------------------|------------|-------------------|
| 0 | 2020-01-05    | AF           | Afghanistan | EMRO       | NaN       | 0                | NaN        | 0                 |
| 1 | 2020-01-12    | AF           | Afghanistan | EMRO       | NaN       | 0                | NaN        | 0                 |
| 2 | 2020-01-19    | AF           | Afghanistan | EMRO       | NaN       | 0                | NaN        | 0                 |
| 3 | 2020-01-26    | AF           | Afghanistan | EMRO       | NaN       | 0                | NaN        | 0                 |
| 4 | 2020-02-02    | AF           | Afghanistan | EMRO       | NaN       | 0                | NaN        | 0                 |

```
[7]: df_data.size
```

```
[7]: 543360
```

```
[8]: print(len(df_data))
```

```
67920
```

```
[9]: df_data.info()
```

JupyterLab interface showing a notebook titled "Practica06\_220186". The notebook contains the following code and output:

```
[9]: df_data.info()
```

<class 'pandas.core.frame.DataFrame'  
RangeIndex: 67920 entries, 0 to 67919  
Data columns (total 8 columns):  
# Column Non-Null Count Dtype  
---  
0 Date\_reported 67920 non-null object  
1 Country\_code 67637 non-null object  
2 Country 67920 non-null object  
3 WHO\_region 62826 non-null object  
4 New\_cases 47299 non-null float64  
5 Cumulative\_cases 67920 non-null int64  
6 New\_deaths 35070 non-null float64  
7 Cumulative\_deaths 67920 non-null int64  
dtypes: float64(2), int64(2), object(4)  
memory usage: 4.1+ MB

Limpiar datos

```
[10]: df_data = df_data.dropna(subset=['New_cases'])  
[11]: print(len(df_data))
```

47299

Transformación de Datos

JupyterLab interface showing the same notebook "Practica06\_220186". The notebook contains the following code and output:

```
[11]: print(len(df_data))
```

47299

Transformación de Datos

```
[12]: df_data = df_data[['Date_reported', 'Country_code', 'Country', 'WHO_region']]  
[13]: print(len(df_data))  
df_data
```

47299

|       | Date_reported | Country_code | Country     | WHO_region |
|-------|---------------|--------------|-------------|------------|
| 8     | 2020-03-01    | AF           | Afghanistan | EMRO       |
| 10    | 2020-03-15    | AF           | Afghanistan | EMRO       |
| 11    | 2020-03-22    | AF           | Afghanistan | EMRO       |
| 12    | 2020-03-29    | AF           | Afghanistan | EMRO       |
| 13    | 2020-04-05    | AF           | Afghanistan | EMRO       |
| ...   | ...           | ...          | ...         | ...        |
| 67911 | 2025-04-06    | ZW           | Zimbabwe    | AFRO       |
| 67912 | 2025-04-13    | ZW           | Zimbabwe    | AFRO       |
| 67913 | 2025-04-20    | ZW           | Zimbabwe    | AFRO       |

Practica06\_220186 Last Checkpoint: 4 days ago

File Edit View Run Kernel Settings Help

Not Trusted

JupyterLab Python 3 (ipykernel)

| 67911 | 2025-04-06 | ZW | Zimbabwe | AFRO |  |
|-------|------------|----|----------|------|--|
| 67912 | 2025-04-13 | ZW | Zimbabwe | AFRO |  |
| 67913 | 2025-04-20 | ZW | Zimbabwe | AFRO |  |
| 67914 | 2025-04-27 | ZW | Zimbabwe | AFRO |  |
| 67915 | 2025-05-04 | ZW | Zimbabwe | AFRO |  |

47299 rows x 4 columns

Exportar datos a una tabla temporal de SQL

```
[15]: from sqlalchemy import text

with engine.connect() as conn:
    conn.execute(text("""
        CREATE TEMPORARY TABLE cases_tempo (
            'Date_reported' text,
            'Country_code' text,
            'Country' text,
            'WHO_region' text
        ));
    """))
```

03:02 a.m. 25/06/2025

Practica06\_220186 Last Checkpoint: 4 days ago

File Edit View Run Kernel Settings Help

Not Trusted

JupyterLab Python 3 (ipykernel)

Exportar datos a una tabla temporal de SQL

```
[15]: from sqlalchemy import text

with engine.connect() as conn:
    conn.execute(text("""
        CREATE TEMPORARY TABLE cases_tempo (
            'Date_reported' text,
            'Country_code' text,
            'Country' text,
            'WHO_region' text
        ));
    """))
```

```
[17]: df_data.to_sql(
    name='cases_tempo',
    con=engine,
    if_exists='append',
    index=False
)
```

[17]: 47299

```
[18]: verification_query = f"SELECT * FROM cases_tempo LIMIT 15"
result = pd.read_sql(verification_query, engine)
print(result)
```

18°C Parc. nublado 03:02 a.m. 25/06/2025

Practica06\_220186 Last Checkpoint: 4 days ago

File Edit View Run Kernel Settings Help

Not Trusted

JupyterLab Python 3 (ipykernel)

```
[17]: 47299
```

```
[18]: verification_query = f"SELECT * FROM cases_tempo LIMIT 15"
      result = pd.read_sql(verification_query, engine)
      print(result)
```

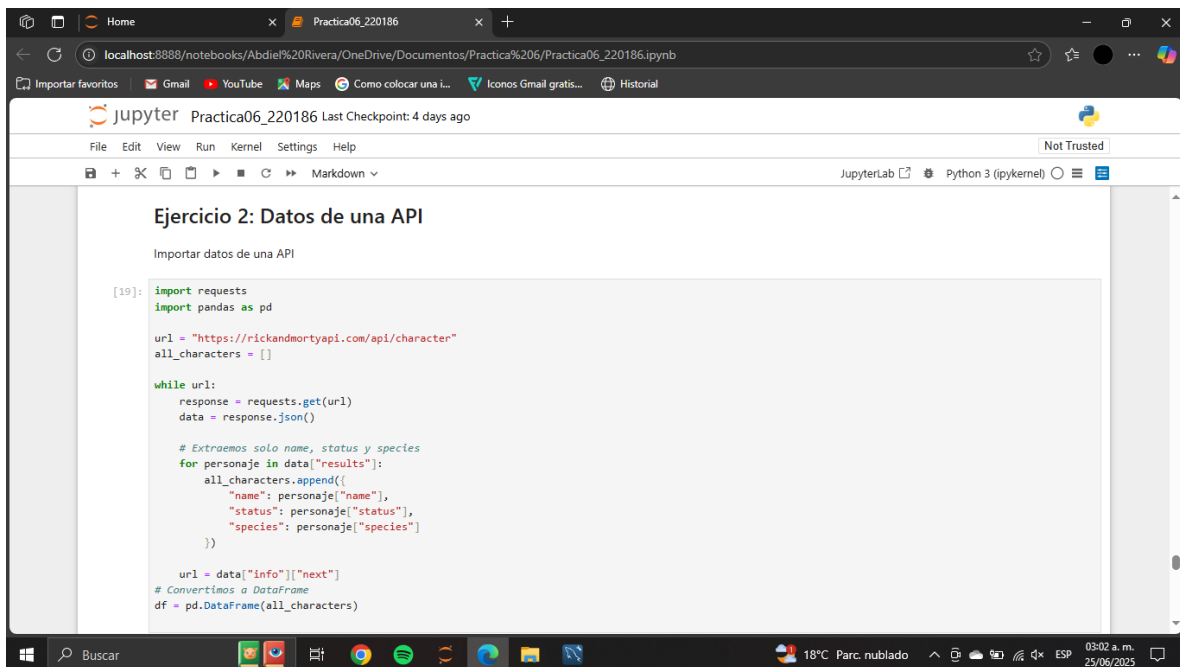
|    | Date_reported | Country_code | Country     | WHO_region |
|----|---------------|--------------|-------------|------------|
| 0  | 2020-03-01    | AF           | Afghanistan | EHIRO      |
| 1  | 2020-03-15    | AF           | Afghanistan | EHIRO      |
| 2  | 2020-03-22    | AF           | Afghanistan | EHIRO      |
| 3  | 2020-03-29    | AF           | Afghanistan | EHIRO      |
| 4  | 2020-04-05    | AF           | Afghanistan | EHIRO      |
| 5  | 2020-04-12    | AF           | Afghanistan | EHIRO      |
| 6  | 2020-04-19    | AF           | Afghanistan | EHIRO      |
| 7  | 2020-04-26    | AF           | Afghanistan | EHIRO      |
| 8  | 2020-05-03    | AF           | Afghanistan | EHIRO      |
| 9  | 2020-05-10    | AF           | Afghanistan | EHIRO      |
| 10 | 2020-05-17    | AF           | Afghanistan | EHIRO      |
| 11 | 2020-05-24    | AF           | Afghanistan | EHIRO      |
| 12 | 2020-05-31    | AF           | Afghanistan | EHIRO      |
| 13 | 2020-06-07    | AF           | Afghanistan | EHIRO      |
| 14 | 2020-06-14    | AF           | Afghanistan | EHIRO      |

## Ejercicio 2: Datos de una API

Importar datos de una API

18°C Parc. nublado 03:02 a. m. 25/06/2025

## Ejercicio 2: Datos de una API



The screenshot shows a JupyterLab notebook titled "Practica06\_220186". The code in cell [19] imports the 'requests' and 'pandas' libraries. It defines a URL for the Rick and Morty API and initializes an empty list 'all\_characters'. A while loop fetches data from the API, extracts character names and species, and appends them to the list. The loop continues until the 'next' key in the response is null. Finally, the data is converted into a pandas DataFrame named 'df'.

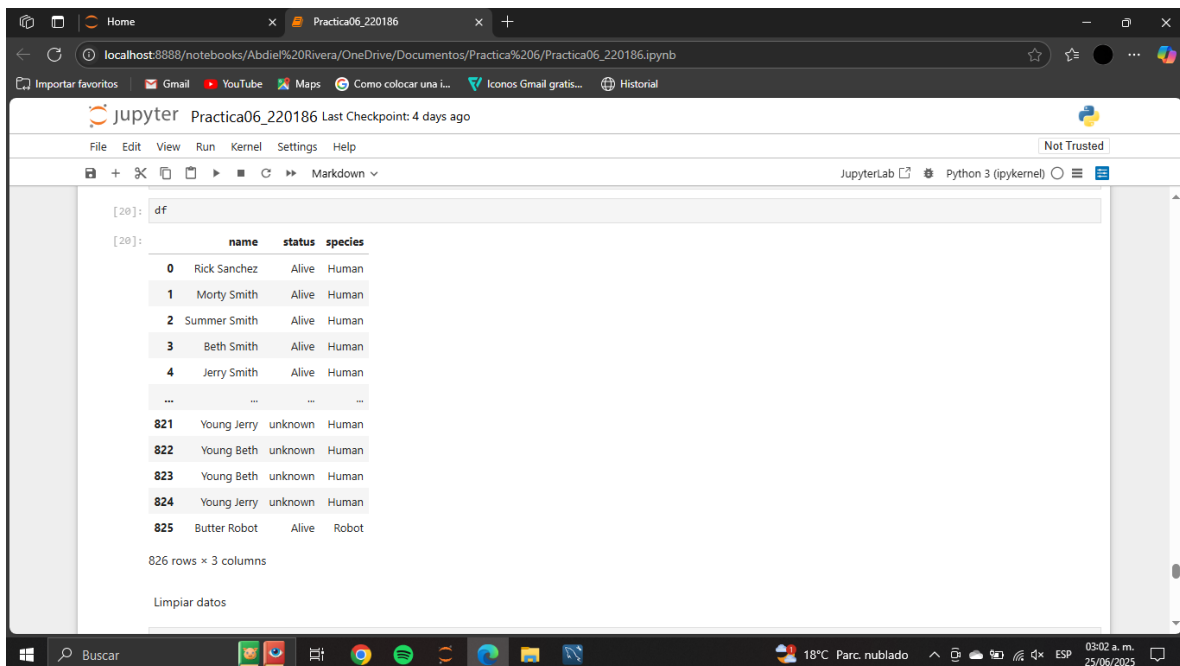
```
[19]: import requests
import pandas as pd

url = "https://rickandmortyapi.com/api/character"
all_characters = []

while url:
    response = requests.get(url)
    data = response.json()

    # Extraemos solo name, status y species
    for personaje in data["results"]:
        all_characters.append({
            "name": personaje["name"],
            "status": personaje["status"],
            "species": personaje["species"]
        })

    url = data["info"]["next"]
# Convertimos a DataFrame
df = pd.DataFrame(all_characters)
```



The screenshot shows the same JupyterLab notebook, but now displaying the output of the code. Cell [20] shows the DataFrame 'df' with 826 rows and 3 columns: 'name', 'status', and 'species'. The output is displayed as a table with alternating row colors.

|     | name         | status  | species |
|-----|--------------|---------|---------|
| 0   | Rick Sanchez | Alive   | Human   |
| 1   | Morty Smith  | Alive   | Human   |
| 2   | Summer Smith | Alive   | Human   |
| 3   | Beth Smith   | Alive   | Human   |
| 4   | Jerry Smith  | Alive   | Human   |
| ... | ...          | ...     | ...     |
| 821 | Young Jerry  | unknown | Human   |
| 822 | Young Beth   | unknown | Human   |
| 823 | Young Beth   | unknown | Human   |
| 824 | Young Jerry  | unknown | Human   |
| 825 | Butter Robot | Alive   | Robot   |

826 rows x 3 columns

Limpiar datos

JupyterLab interface showing a notebook titled "Practica06\_220186". The notebook is running on a local host (localhost:8888) and displays the output of a data cleaning operation.

The code cell shows:

```
[21]: df.dropna()
```

The output displays a DataFrame with 826 rows and 3 columns:

|     | name         | status  | species |
|-----|--------------|---------|---------|
| 0   | Rick Sanchez | Alive   | Human   |
| 1   | Morty Smith  | Alive   | Human   |
| 2   | Summer Smith | Alive   | Human   |
| 3   | Beth Smith   | Alive   | Human   |
| 4   | Jerry Smith  | Alive   | Human   |
| ... | ...          | ...     | ...     |
| 821 | Young Jerry  | unknown | Human   |
| 822 | Young Beth   | unknown | Human   |
| 823 | Young Beth   | unknown | Human   |
| 824 | Young Jerry  | unknown | Human   |
| 825 | Butter Robot | Alive   | Robot   |

826 rows x 3 columns

JupyterLab interface showing a notebook titled "Practica06\_220186". The notebook is running on a local host (localhost:8888) and displays the output of a data transformation operation.

The code cell shows:

```
[22]: df = df[['name', 'species']]
```

The output displays a DataFrame with 826 rows and 2 columns:

|     | name         | species |
|-----|--------------|---------|
| 0   | Rick Sanchez | Human   |
| 1   | Morty Smith  | Human   |
| 2   | Summer Smith | Human   |
| 3   | Beth Smith   | Human   |
| 4   | Jerry Smith  | Human   |
| ... | ...          | ...     |
| 821 | Young Jerry  | Human   |
| 822 | Young Beth   | Human   |
| 823 | Young Beth   | Human   |
| 824 | Young Jerry  | Human   |
| 825 | Butter Robot | Robot   |

826 rows x 2 columns

Practica06\_220186 Last Checkpoint: 4 days ago

File Edit View Run Kernel Settings Help

Not Trusted

JupyterLab Python 3 (ipykernel)

Transformar datos

```
[22]: df = df[['name', 'species']]
```

```
[23]: df
```

```
[23]:
```

|     | name         | species |
|-----|--------------|---------|
| 0   | Rick Sanchez | Human   |
| 1   | Morty Smith  | Human   |
| 2   | Summer Smith | Human   |
| 3   | Beth Smith   | Human   |
| 4   | Jerry Smith  | Human   |
| ... | ...          | ...     |
| 821 | Young Jerry  | Human   |
| 822 | Young Beth   | Human   |
| 823 | Young Beth   | Human   |
| 824 | Young Jerry  | Human   |
| 825 | Butter Robot | Robot   |

826 rows x 2 columns

18°C Parc. nublado 03:03 a. m. 25/06/2025

Practica06\_220186 Last Checkpoint: 4 days ago

File Edit View Run Kernel Settings Help

Not Trusted

JupyterLab Python 3 (ipykernel)

Exportar datos

```
[24]: from sqlalchemy import text
```

```
with engine.connect() as conn:
```

```
conn.execute(text("""
```

```
CREATE TEMPORARY TABLE characters_rick (
```

```
name VARCHAR(100),
```

```
species VARCHAR(50)
```

```
));
```

```
"""))
```

```
[25]: df.to_sql(
```

```
name='characters_rick',
```

```
con=engine,
```

```
if_exists='append',
```

```
index=False
```

```
)
```

```
[25]: 826
```

```
[26]: verification_query = f"SELECT * FROM characters_rick LIMIT 20"
```

```
result = pd.read_sql(verification_query, engine)
```

```
print(result)
```

|   | name         | species |
|---|--------------|---------|
| 0 | Rick Sanchez | Human   |

18°C Parc. nublado 03:03 a. m. 25/06/2025

Practica06\_220186

localhost:8888/notebooks/Abdie%20Rivera/OneDrive/Documentos/Practica%206/Practica06\_220186.ipynb

Importar favoritos Gmail YouTube Maps Como colocar una I... Iconos Gmail gratis... Historial

Jupyter Practica06\_220186 Last Checkpoint: 4 days ago

File Edit View Run Kernel Settings Help Not Trusted

JupyterLab Python 3 (ipykernel)

```
[26]: verification_query = f"SELECT * FROM characters_rick LIMIT 20"
      result = pd.read_sql(verification_query, engine)
      print(result)
```

|    | name                      | species |
|----|---------------------------|---------|
| 0  | Rick Sanchez              | Human   |
| 1  | Morty Smith               | Human   |
| 2  | Summer Smith              | Human   |
| 3  | Beth Smith                | Human   |
| 4  | Jerry Smith               | Human   |
| 5  | Abadango Cluster Princess | Alien   |
| 6  | Abradolf Lincler          | Human   |
| 7  | Adjudicator Rick          | Human   |
| 8  | Agency Director           | Human   |
| 9  | Alan Ralls                | Human   |
| 10 | Albert Einstein           | Human   |
| 11 | Alexander                 | Human   |
| 12 | Alien Googah              | Alien   |
| 13 | Alien Morty               | Alien   |
| 14 | Alien Rick                | Alien   |
| 15 | Amish Cyborg              | Alien   |
| 16 | Annie                     | Human   |
| 17 | Antenna Morty             | Human   |
| 18 | Antenna Rick              | Human   |
| 19 | Ants in my Eyes Johnson   | Human   |

18°C Parc. nublado 03:03 a.m. 25/06/2025