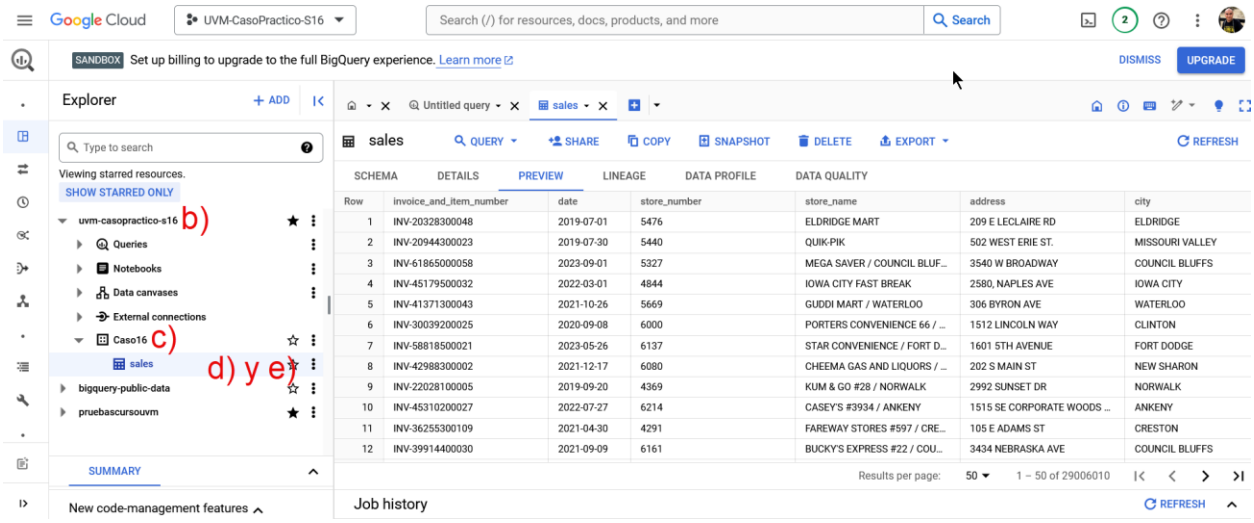
Bootcamp Data Science - Caso Práctico S16 - Gerardo Rodríguez.docx

Paso 1: Preparación del Dataset y Entorno de Trabajo

1. Agregar un Dataset y una tabla a BigQuery:

a) Ve a la consola de Google Cloud.
b) Crea un nuevo proyecto y navega a BigQuery.
c) En el panel de recursos, crea un nuevo conjunto de datos.
d) Carga un dataset público disponible en BigQuery o un dataset personal
e) subiéndolo a tu espacio de trabajo.

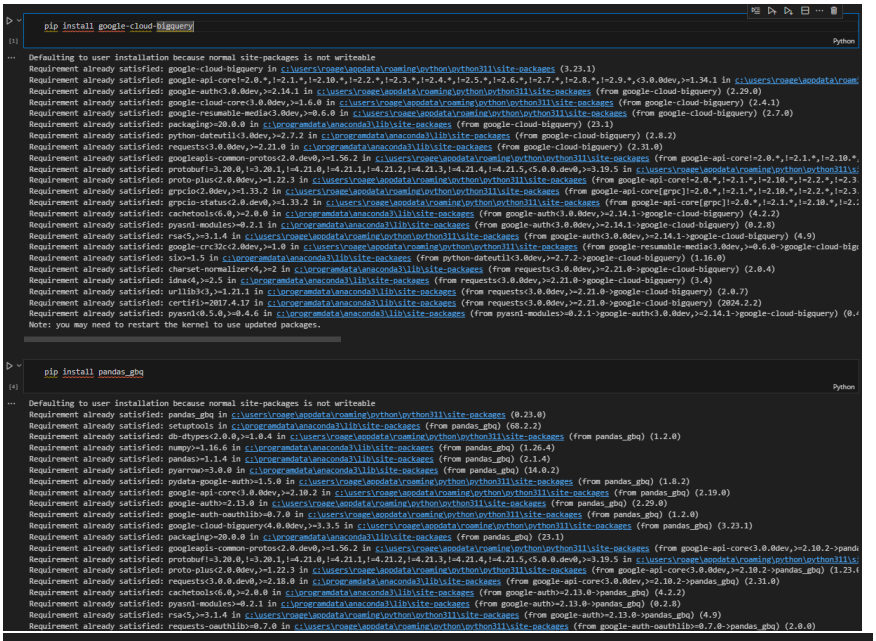


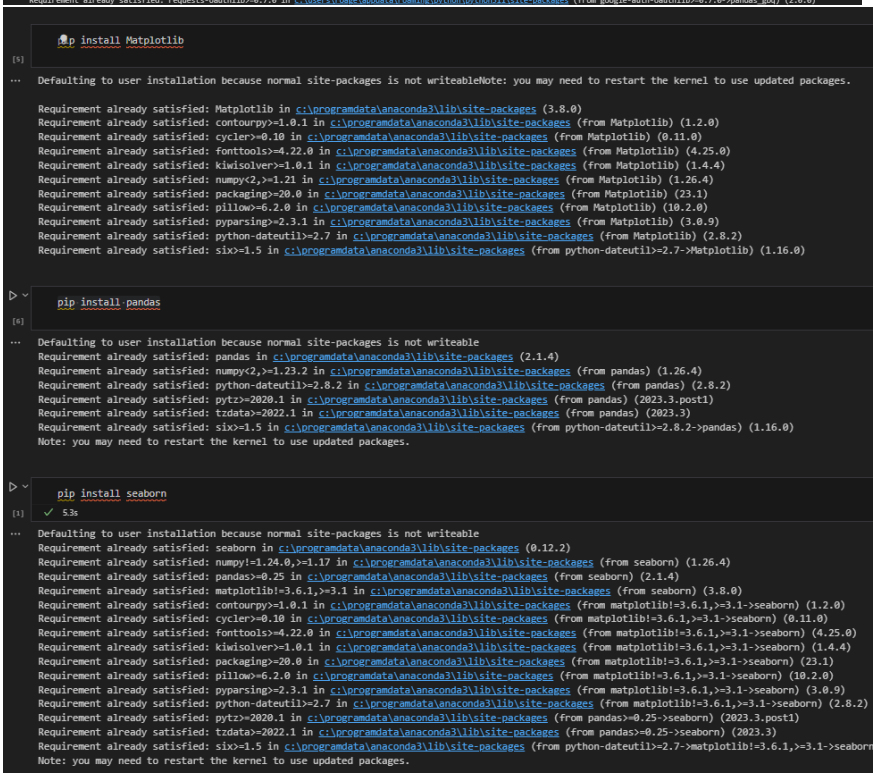
2. Configurar el Entorno en VSCode:

• Verificar tener las extensiones de Python, Jupyter y BigQuery para VSCode.

• Instala las bibliotecas necesarias, como google-cloud-bigquery y pandas, utilizando pip:

• pip install google-cloud-bigquery pandas matplotlib seaborn

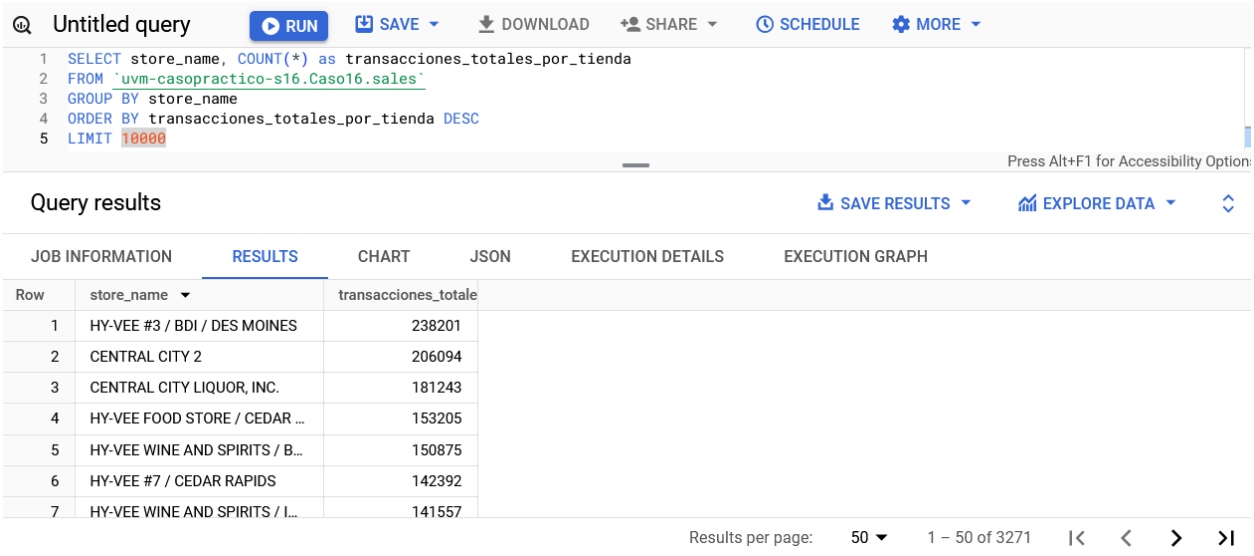




Paso 2: Consultas SQL en BigQuery (Sugeridas: 2)

A continuación, se presentan 4 ejemplos de consultas a ejecutar en BigQuery:

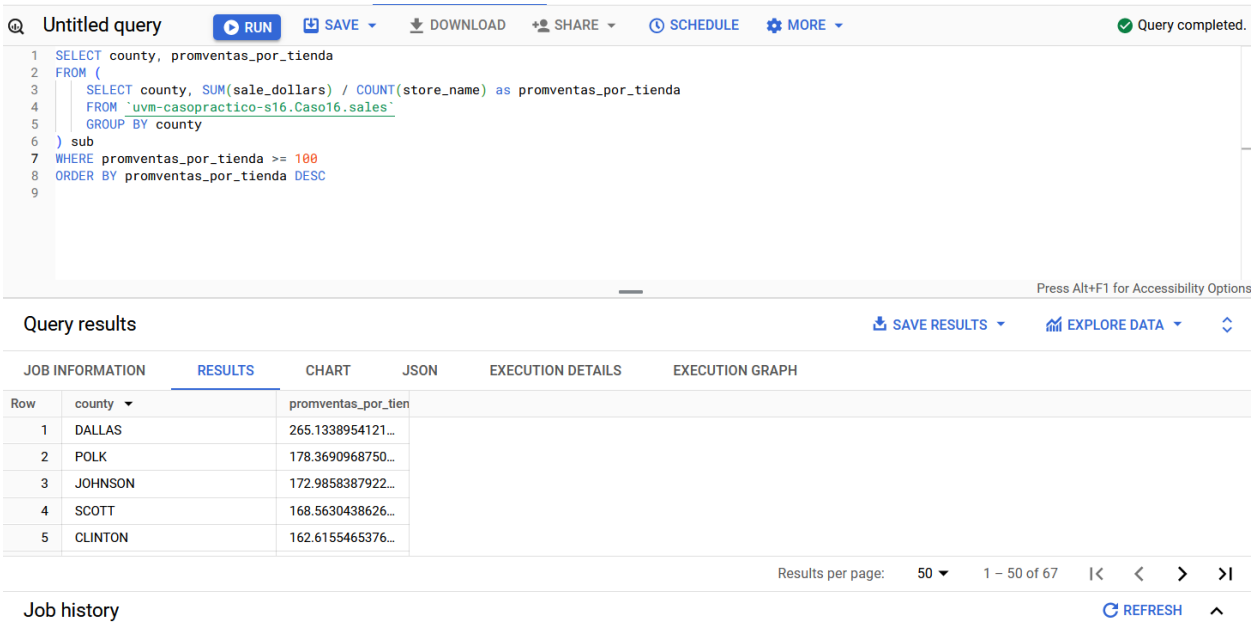
a) Consulta de datos agregados	**Transacciones totales por tienda**
SELECT category, COUNT(*) as total_category FROM `project.dataset.table` GROUP BY category ORDER BY total_category DESC	SELECT store_name, COUNT(*) as transacciones_totales_por_tienda FROM `uvm-casopractico-s16.Caso16.sales` GROUP BY store_name ORDER BY transacciones_totales_por_tienda DESC LIMIT 10000



Se muestran las tiendas mostrando las transacciones que han entregado.

Se identifican 3271 tiendas diferentes.

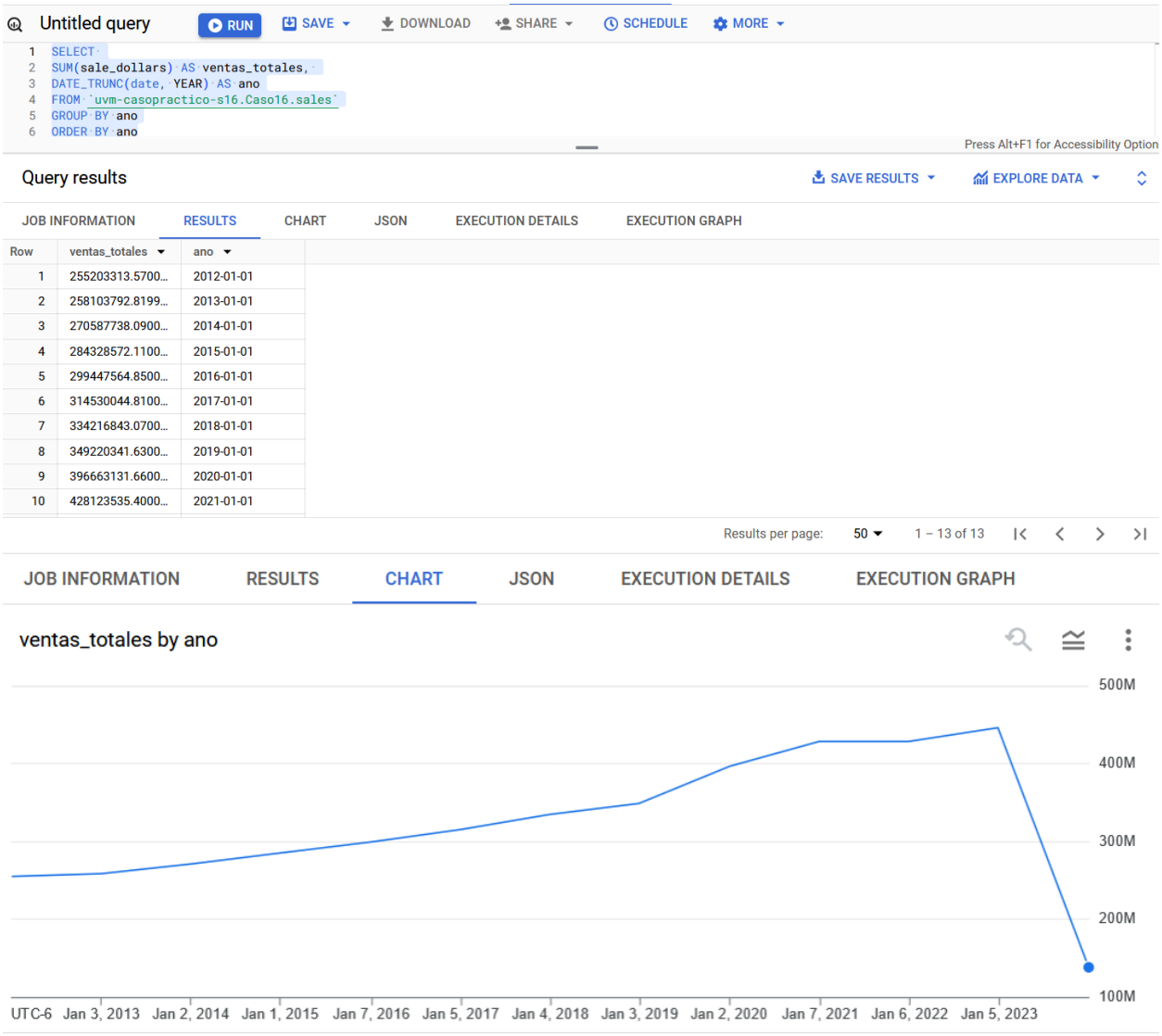
b) Consulta de Datos con Filtro:	Promedio de ventas de tiendas por Condado
SELECT name, number FROM `project.dataset.table` WHERE number > 100.0	SELECT county, promventas_por_tienda FROM (SELECT county, SUM(sale_dollars) / COUNT(store_name) as promventas_por_tienda FROM `uvm-casopractico-s16.Caso16.sales` GROUP BY county) sub WHERE promventas_por_tienda >= 100 ORDER BY promventas_por_tienda DESC



Solo 67 de los 101 condados tienen un promedio de ventas por tienda superior a 100.

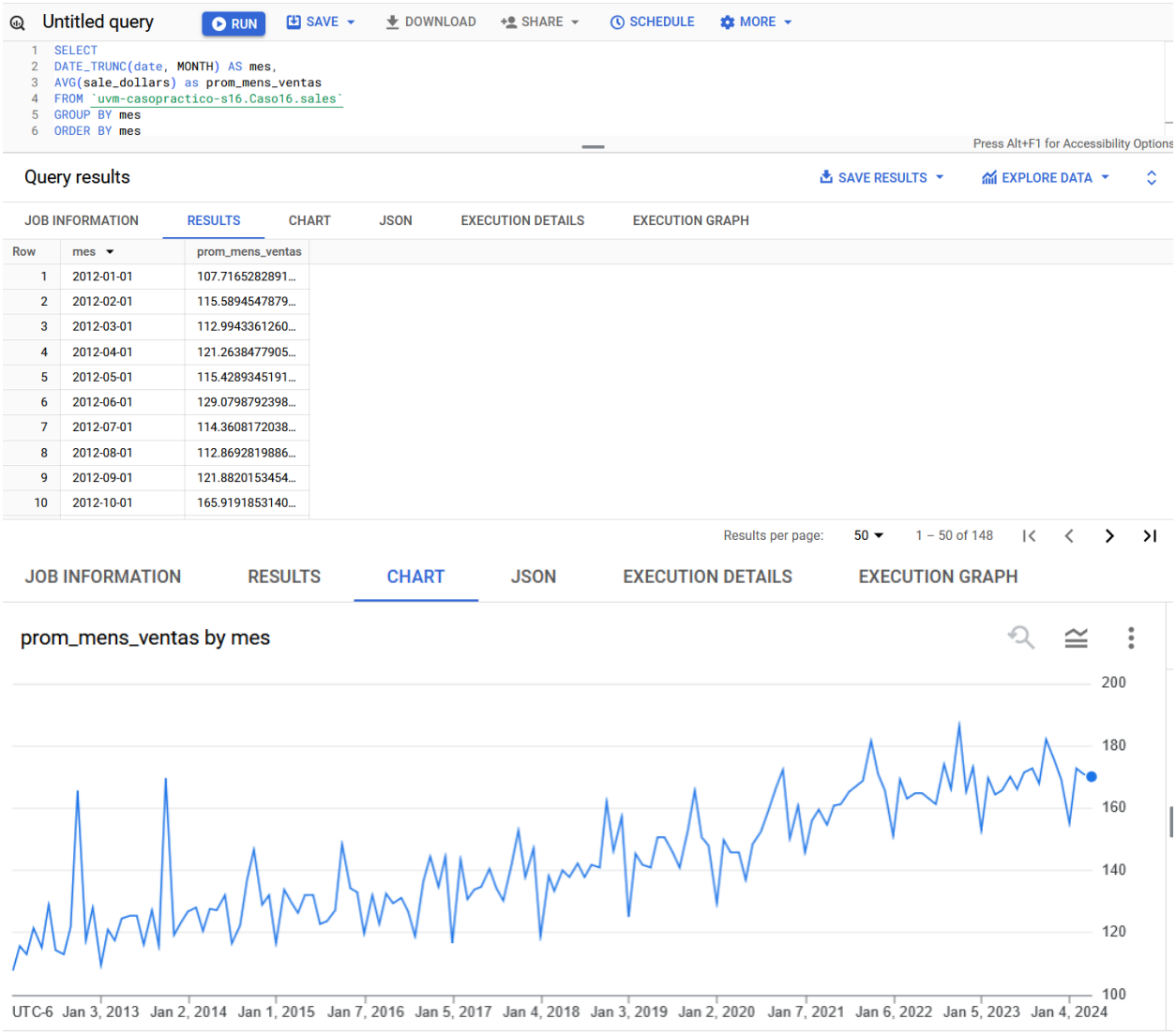
c) Consulta de Tendencias a lo Largo del Tiempo:	Sumatoria de las ventas totales por año
SELECT year, number FROM `project.dataset.table` GROUP BY year ORDER BY year	SELECT SUM(sale_dollars) AS ventas_totales, DATE_TRUNC(date, YEAR) AS ano FROM `uvm-casopractico-s16.Caso16.sales` GROUP BY ano ORDER BY ano

Se muestran la sumatoria de las ventas totales por año.



SELECT DATE_TRUNC(date, MONTH) as month, AVG(sales) as avg_sales FROM `project.dataset.table` GROUP BY month ORDER BY month	SELECT DATE_TRUNC(date, MONTH) AS mes, AVG(sale_dollars) as prom_mens_ventas FROM `uvm-casopractico-s16.Caso16.sales` GROUP BY mes ORDER BY mes

Se muestra el promedio de las ventas por mes



Paso 3: Consultas SQL en Python (Sugeridas: 2, pueden ser las del paso anterior)

1. Conexión a BigQuery desde Python:

Crea un archivo Jupyter Notebook en VSCode.
Utiliza la biblioteca google-cloud-bigquery para conectarte a BigQuery y ejecutar tus consultas.

a) Cargamos localmente nuevo set de datos “Jugadores de Basketball” que consta de 4 tablas

```
import pandas as pd
df1 = pd.read_csv('players.csv')
df2 = pd.read_csv('players_teams.csv')
df3 = pd.read_csv('teams.csv')
df4 = pd.read_csv('awards_players.csv')
```
✓ 0.9s

b) Creamos nuevo proyecto y cuenta de servicio

Service accounts for project "UVM-CasoPractico-S16"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts.

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. Learn more about service account organization policies.

Filter Enter property name or value

Email	Status	Name	Description	Key ID	Key creation date	OAuth 2 Client ID	Actions
uvm-casopractico-s16@uvm-casopractico-s16.iam.gserviceaccount.com	Enabled	UVM-CasoPractico-S16	UVM-CasoPractico-S16	7619a96886b9006a469b85a67826381ec6acf96a	May 30, 2024	103465380243737	

c) Creamos conexión y subimos los archivos locales a BigQuery

```
# Objetivo: Vamos a crear la conexión con BigQuery y vamos a leer mediante consulta SQL desde Python.
# Generamos conexion a BQ
from google.oauth2 import service_account # para generar conexion
bq_cred = service_account.Credentials.from_service_account_file('uvm-casopractico-s16-7619a96886b9.json')
```
[1] ✓ 0.1s

```
# Enviamos el df a BQ - Podemos crear o reemplazar una tabla
import pandas_gbq # para interactuar con BQ
pandas_gbq.to_gbq(df1, 'Caso16b.players', project_id= 'uvm-casopractico-s16', if_exists= 'replace', credentials = bq_cred)
```
[3] ✓ 8.8s

```
100%|██████████| 1/1 [00:00<?, ?it/s]
```

```
pandas_gbq.to_gbq(df2, 'Caso16b.players_teams', project_id= 'uvm-casopractico-s16', if_exists= 'replace', credentials = bq_cred)
```
[4] ✓ 7.4s

```
100%|██████████| 1/1 [00:00<?, ?it/s]
```

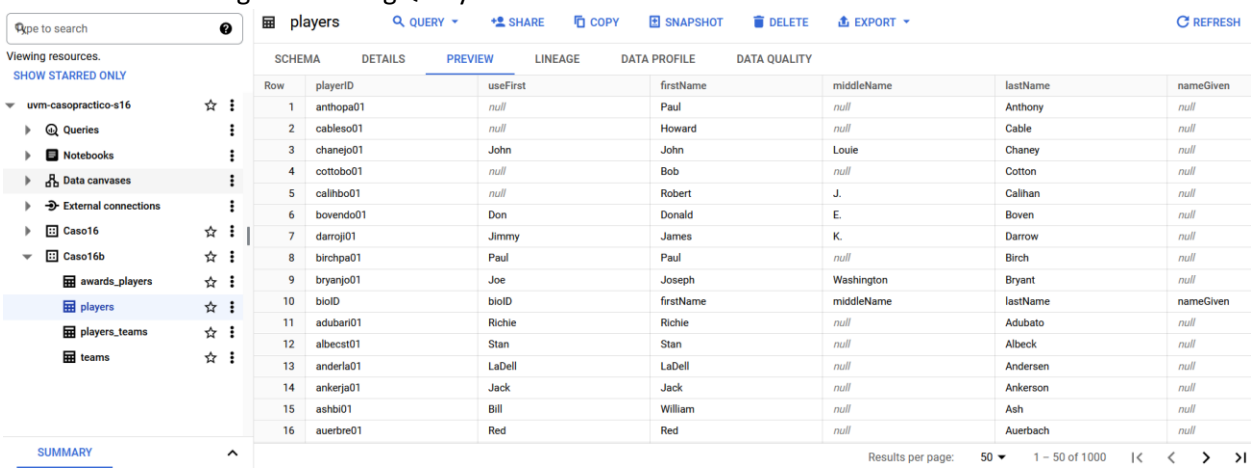
```
pandas_gbq.to_gbq(df3, 'Caso16b.teams', project_id= 'uvm-casopractico-s16', if_exists= 'replace', credentials = bq_cred)
```
[5] ✓ 7.8s

```
100%|██████████| 1/1 [00:00<?, ?it/s]
```

```
pandas_gbq.to_gbq(df4, 'Caso16b.awards_players', project_id= 'uvm-casopractico-s16', if_exists= 'replace', credentials = bq_cred)
```
[6] ✓ 6.8s

```
100%|██████████| 1/1 [00:00<?, ?it/s]
```

d) Validamos datos migrados en BigQuery

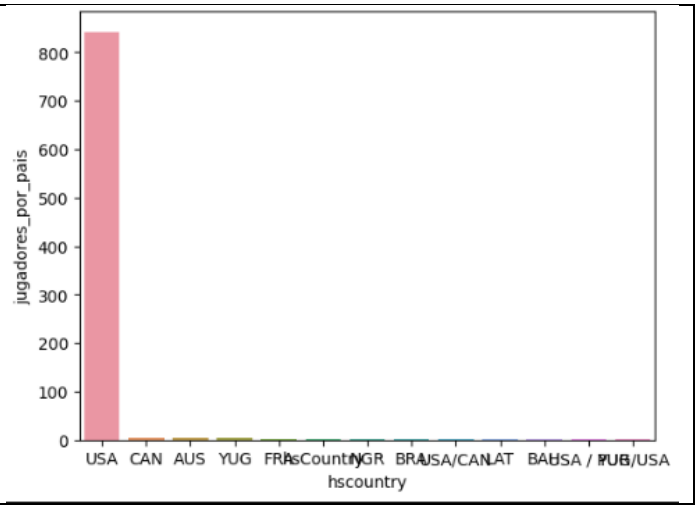


Paso 2: Consultas SQL en BigQuery (Sugeridas: 2)

A continuación, se presentan 4 ejemplos de consultas a ejecutar en BigQuery:

a) Consulta de datos agregados	**Contar jugadores únicos por país**
SELECT category, COUNT(*) as total_category FROM `project.dataset.table` GROUP BY category ORDER BY total_category DESC	`sql = """` `SELECT hscountry, COUNT(DISTINCT playerID) as jugadores_por_pais` `FROM `uvm-casopractico-s16.Caso16b.players` `GROUP BY hscountry` `ORDER BY jugadores_por_pais DESC` `"""` `df_bq = pd.read_gbq(sql, project_id='uvm-casopractico-s16', credentials = bq_cred, dialect='standard') #Dialect standard: para usar BigQuery's standard SQL dialect` `print(df_bq)`

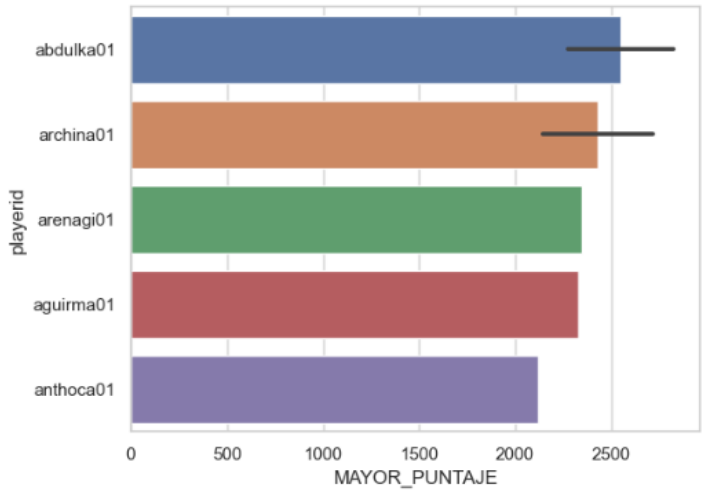
	hscountry	jugadores_por_pais
0	USA	842
1	None	138
2	CAN	4
3	AUS	3
4	YUG	3
5	FRA	2
6	hsCountry	1
7	NGR	1
8	BRA	1
9	USA/CAN	1
10	LAT	1
11	BAH	1
12	USA / PUR	1
13	YUG/USA	1



Se muestran los jugadores por país en la Base de datos, identificando que Estados Unidos es el que tienen más jugadores.

b) Consulta de Datos con Filtro:	Jugadores con mas de 2000 puntos

SELECT name, number
FROM `project.dataset.table`
WHERE number > 100.0
	`sql = """`
	`SELECT`
	`  tmID,  playerid,`
	`  MAX(points) AS MAYOR_PUNTAJE`
	`FROM `uvm-casopractico-s16.Caso16b.players_teams``
	`WHERE points >= 2000`
	`GROUP BY tmID, playerID`
	`ORDER BY MAYOR_PUNTAJE DESC`
	`"""`
	`df_bq = pd.read_gbq(sql, project_id='uvm-casopractico-s16', credentials = bq_cred, dialect='standard') #Dialect standard: para usar BigQuery's standard SQL dialect`
	`print(df_bq)`

tmID playerid MAYOR_PUNTAJE 0 MIL abdulka01 2822 1 KCO archina01 2719 2 WAS arenagi01 2346 3 DAL aguirma01 2330 4 LAL abdulka01 2275 5 CIN archina01 2145 6 DEN anthoca01 2122	

Abdulka01 del equipo MIL es el jugador con mayor puntaje en la Base de datos.

c) Consulta de Tendencias a lo Largo del Tiempo:	Jugadores con más de un premio por año

SELECT year, number
FROM `project.dataset.table`
GROUP BY year
ORDER BY year
	sql = """
	SELECT playerID, year, COUNT(playerID) AS MVP
	FROM `uvm-casopractico-s16.Caso16b.awards_players`
	GROUP BY playerID, year
	HAVING COUNT(playerID) > 1
	ORDER BY year DESC;
	"""
	df_bq = pd.read_gbq(sql, project_id='uvm-casopractico-s16', credentials = bq_cred, dialect='standard') #Dialect standard: para usar BigQuery's standard SQL dialect
	print(df_bq)

```
      playerID  year  MVP
0    jamesle01  2011    4
1    irvinky01  2011    2
2    chandty01  2011    3
3    howardw01  2011    2
4    bryanko01  2011    2
..         ...   ...  ...
171  chambwi01  1965    2
172  lucasje01  1963    2
173  chambwi01  1959    3
174  bayloel01  1958    2
175  cousybo01  1956    2

[176 rows x 3 columns]
```

Mostramos los jugadores con más premios agrupados por año.

Premios por año

```
sql = """
SELECT year, COUNT(playerID) AS MVP
FROM `uvm-casopractico-
s16.Caso16b.awards_players`
GROUP BY year
ORDER BY year DESC;
"""
df_bq = pd.read_gbq(sql,
project_id='uvm-casopractico-s16',
credentials = bq_cred,
dialect='standard') #Dialect standard:
para usar BigQuery's standard SQL
dialect
print(df_bq)
```

```
...      year  MVP
     0   2011   30
     1   2010   27
     2   2009   30
     3   2008   27
     4   2007   25
     ..   ...  ...
     67  1942    1
     68  1941    2
     69  1939    1
     70  1938    1
     71  1937    2

     [72 rows x 2 columns]
```

```
# Para una serie temporal
import matplotlib.pyplot as plt
# Asegúrate de que 'year' y 'number' están en el formato correcto
# df['year'] debe ser de tipo datetime, para asegurarlos
df_bq['year'] = pd.to_datetime(df_bq['year'], format='%Y')
# Asegúrate de que los datos están ordenados por 'year'
df_bq = df_bq.sort_values('year')
# Crea la figura y el eje
plt.figure(figsize=(10, 5))
plt.plot(df_bq['year'], df_bq['MVP'], marker='', color='blue', linewidth=2)
# Título y etiquetas
plt.title('Serie de Tiempo de JAMES a través del tiempo')
plt.xlabel('year')
plt.ylabel('MVP')
# Mostrar la gráfica
plt.show()
```

d) Consulta de Datos Relacionales (Join):	Lista de Jugadores por Equipo
SELECT a.name, b.purchase_date, b.amount FROM `project.dataset.customers` a JOIN `project.dataset.purchases` b ON a.customer_id = b.customer_id	sql = """ SELECT DISTINCT t.name, p.firstName, p.lastName FROM `uvm-casopractico- s16.Caso16b.players` p JOIN `uvm-casopractico- s16.Caso16b.players_teams` pt ON p.playerID = pt.playerID JOIN `uvm-casopractico- s16.Caso16b.teams` t ON pt.lgID = t.lgID ORDER BY t.name,p.lastName ; """ df_bq = pd.read_gbq(sql, project_id='uvm-casopractico-s16', credentials = bq_cred, dialect='standard') #Dialect standard: para usar BigQuery's standard SQL dialect print(df_bq)

```
...                              name firstName    lastName
     0      Akron Firestone Non-Skids      John   Abramovic
     1      Akron Firestone Non-Skids    Howard       Adams
     2      Akron Firestone Non-Skids   Raymond       Adams
     3      Akron Firestone Non-Skids     Glenn       Adams
     4      Akron Firestone Non-Skids    Willie       Adams
     ...                          ...       ...         ...
     11289          Youngstown Bears      Carl        Baer
     11290          Youngstown Bears     Frank       Baird
     11291          Youngstown Bears       Bob       Baker
     11292          Youngstown Bears       Art  Bakeraitis
     11293          Youngstown Bears   Herbert        Ball

     [11294 rows x 3 columns]
```

Se muestra la lista de jugadores por equipo.