Sprint 3 - Manejo eficiente de datos al programar aplicaciones distribuidas

Desarrolla

1. **¿Cuál es el top 5 de aerolíneas con más vuelos? (query_1.hql y script_query_1.py),**

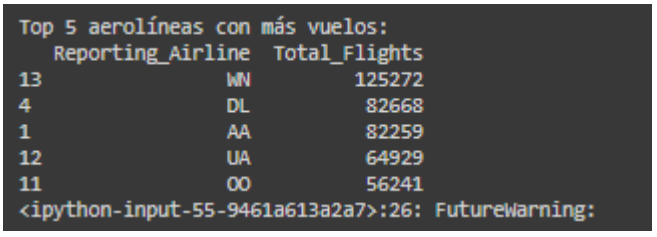
HiveQL

```
-- query_1.hql
USE flights_db;

SELECT reporting_airline AS Airline, COUNT(*) AS Total_Flight
FROM raw_flights_data
GROUP BY reporting_airline
ORDER BY Total_Flights DESC
LIMIT 5;
```

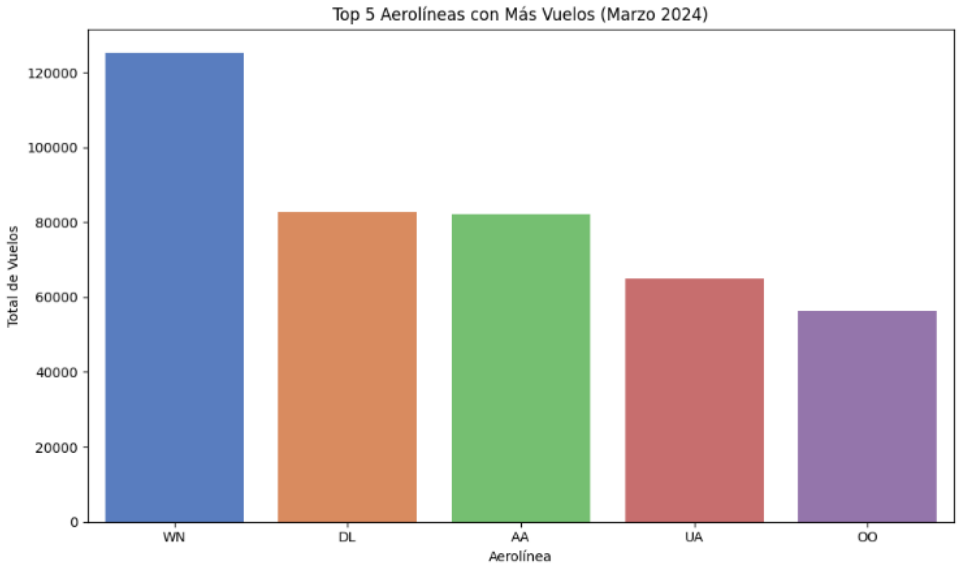
Python

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Cargar el archivo CSV con el argumento low_memory=False par
csv_file = '/data/airlines/2024_3 - copia.csv'

# Leer el archivo CSV en un DataFrame
df = pd.read_csv(csv_file, low_memory=False)

# Agrupar por la columna 'Reporting_Airline' y contar el núme
top_airlines = df.groupby('Reporting_Airline').size().reset_i

# Ordenar por el número de vuelos en orden descendente
```

```
top_airlines = top_airlines.sort_values(by='Total_Flights', a

# Seleccionar el top 5 de aerolíneas con más vuelos
top_5_airlines = top_airlines.head(5)

# Mostrar el resultado
print("Top 5 aerolíneas con más vuelos:")
print(top_5_airlines)

# Crear la visualización con seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x='Reporting_Airline', y='Total_Flights', data=to

# Agregar etiquetas y título
plt.title('Top 5 Aerolíneas con Más Vuelos (Marzo 2024)')
plt.xlabel('Aerolínea')
plt.ylabel('Total de Vuelos')

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```



```
Top 5 aerolíneas con más vuelos:
   Reporting_Airline  Total_Flights
13                WN         125272
4                 DL          82668
1                 AA          82259
12                UA          64929
11                OO          56241
<ipython-input-55-9461a613a2a7>:26: FutureWarning:
```



2. **¿Cuál es el top 10 de aerolíneas que más llegan en tiempo (en porcentaje de efectividad respecto a todos vuelos que realiza)?(query_2.hql y script_query_2.py).**

HiveQL

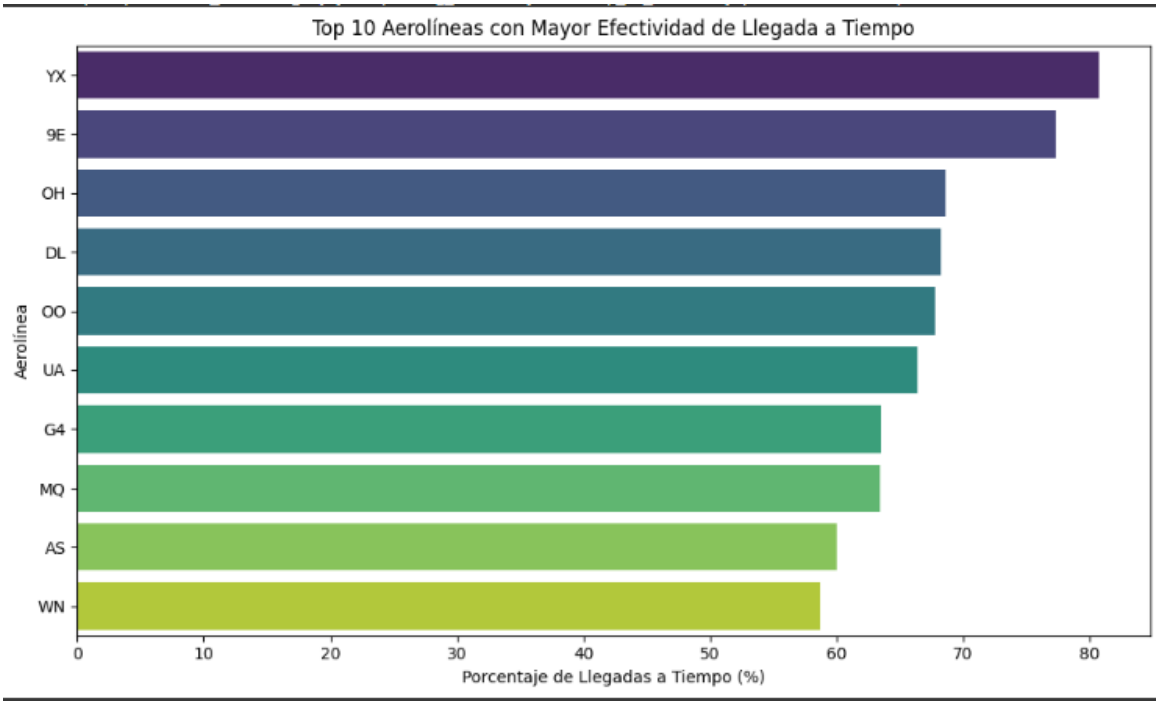
```
USE flights_db;

-- Consulta para obtener el top 10 de aerolíneas con mayor ef
SELECT Reporting_Airline,
       SUM(CASE WHEN ArrDelay <= 0 THEN 1 ELSE 0 END) * 100.0
FROM raw_flights_data
GROUP BY Reporting_Airline
ORDER BY OnTime_Percentage DESC
LIMIT 10;
```

Python

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Cargar el archivo CSV
csv_file = '/data/airlines/2024_3 - copia.csv'
df = pd.read_csv(csv_file, low_memory=False)
```

```
# Filtrar las filas con la columna de retraso de llegada ('Ar
df['OnTime'] = df['ArrDelay'].apply(lambda x: 1 if x <= 0 els

# Agrupar por 'Reporting_Airline' y calcular el porcentaje de
on_time_percentage = df.groupby('Reporting_Airline').agg(
    Total_Flights=('OnTime', 'size'),
    OnTime_Flights=('OnTime', 'sum')
).reset_index()

# Calcular el porcentaje de vuelos a tiempo
on_time_percentage['OnTime_Percentage'] = (
    on_time_percentage['OnTime_Flights'] / on_time_percentage
) * 100

# Ordenar por el porcentaje de vuelos a tiempo en orden desce
top_10_airlines = on_time_percentage.sort_values(by='OnTime_P

# Mostrar el resultado
print("Top 10 aerolíneas con mayor efectividad de llegada a t
print(top_10_airlines[['Reporting_Airline', 'OnTime_Percentag

# Crear la visualización con seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x='OnTime_Percentage', y='Reporting_Airline', dat

# Agregar etiquetas y título
plt.title('Top 10 Aerolíneas con Mayor Efectividad de Llegada
plt.xlabel('Porcentaje de Llegadas a Tiempo (%)')
plt.ylabel('Aerolínea')

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```

```
Top 10 aerolíneas con mayor efectividad de llegada a tiempo:
   Reporting_Airline  OnTime_Percentage
14                YX          80.782816
0                 9E          77.318017
10                OH          68.624741
4                 DL          68.280350
11                OO          67.811739
12                UA          66.361718
6                 G4          63.520986
8                 MQ          63.463146
2                 AS          59.992871
13                WN          58.747366
<ipython-input-57-dc05966f0ada>:32: FutureWarning:
```



3. **¿Cuál es el tiempo promedio de demora de los vuelos en cada aerolínea, para las 5 aerolíneas con más vuelos realizados en éste periodo? (query_3.hql y script_query_3.py`),**

HiveQL

```
-- query_3.hql

USE flights_db;

-- Consulta para obtener el tiempo promedio de demora de las 
WITH top_5_airlines AS (
  SELECT Reporting_Airline, COUNT(*) AS Total_Flights
  FROM raw_flights_data
  GROUP BY Reporting_Airline
  ORDER BY Total_Flights DESC
```

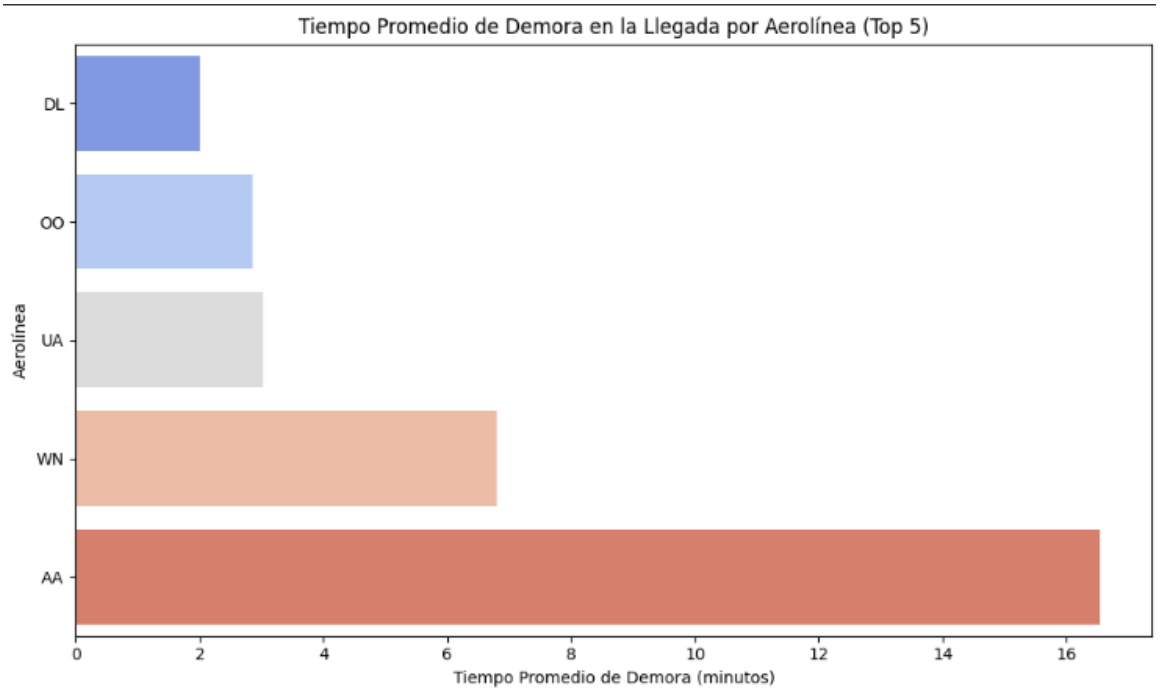
```
  LIMIT 5
)

SELECT rf.Reporting_Airline,
       AVG(ArrDelay) AS Avg_Arrival_Delay
FROM raw_flights_data rf
JOIN top_5_airlines ta ON rf.Reporting_Airline = ta.Reporting
GROUP BY rf.Reporting_Airline
ORDER BY Avg_Arrival_Delay;
```

Python

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Cargar el archivo CSV
csv_file = '/data/airlines/2024_3 - copia.csv'
df = pd.read_csv(csv_file, low_memory=False)

# Agrupar por aerolínea y contar el número de vuelos
top_5_airlines = (
    df.groupby('Reporting_Airline')
    .size()
    .reset_index(name='Total_Flights')
    .sort_values(by='Total_Flights', ascending=False)
    .head(5)
)

# Filtrar el dataframe original para incluir solo las 5 aerol
filtered_df = df[df['Reporting_Airline'].isin(top_5_airlines[

# Calcular el tiempo promedio de demora en la llegada por aer
avg_delay_by_airline = (
    filtered_df.groupby('Reporting_Airline')['ArrDelay']
    .mean()
    .reset_index(name='Avg_Arrival_Delay')
)
```

```
# Ordenar por el tiempo promedio de demora
avg_delay_by_airline = avg_delay_by_airline.sort_values(by='A

# Mostrar el resultado
print("Tiempo promedio de demora de las 5 aerolíneas con más
print(avg_delay_by_airline)

# Crear la visualización con seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x='Avg_Arrival_Delay', y='Reporting_Airline', dat

# Agregar etiquetas y título
plt.title('Tiempo Promedio de Demora en la Llegada por Aerolí
plt.xlabel('Tiempo Promedio de Demora (minutos)')
plt.ylabel('Aerolínea')

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```

```
Tiempo promedio de demora de las 5 aerolíneas con más vuelos:
  Reporting_Airline  Avg_Arrival_Delay
1                DL           2.003276
2                OO           2.856946
3                UA           3.019359
4                WN           6.795795
0                AA          16.548122
<ipython-input-59-608c6e99a5ce>:37: FutureWarning:
```



4. **¿Cómo es la distribución del performance de llegada de los vuelos(véase el siguiente reporte)? (query_4.hql y script_query_4.py)**

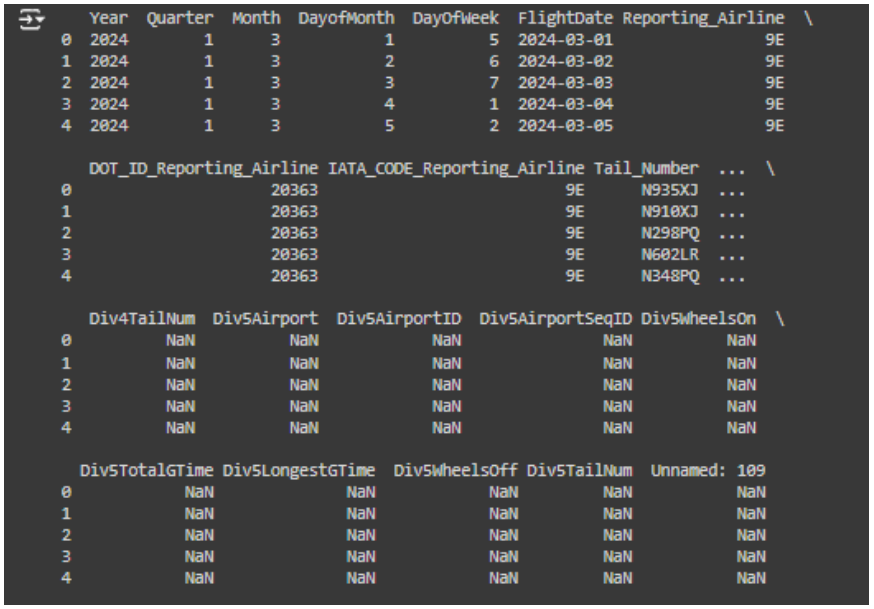
HiveQL

```
-- query_4.hql

USE flights_db;

SELECT
    SUM(CASE WHEN ArrDelay <= 0 THEN 1 ELSE 0 END) * 100.0 / (
    SUM(CASE WHEN CarrierDelay > 0 THEN 1 ELSE 0 END) * 100.0
    SUM(CASE WHEN WeatherDelay > 0 THEN 1 ELSE 0 END) * 100.0
    SUM(CASE WHEN NASDelay > 0 THEN 1 ELSE 0 END) * 100.0 / C(
    SUM(CASE WHEN SecurityDelay > 0 THEN 1 ELSE 0 END) * 100.(
    SUM(CASE WHEN LateAircraftDelay > 0 THEN 1 ELSE 0 END) * 1
    SUM(CASE WHEN Cancelled = 1 THEN 1 ELSE 0 END) * 100.0 / (
    SUM(CASE WHEN Diverted = 1 THEN 1 ELSE 0 END) * 100.0 / C(
FROM
    raw_flights_data
WHERE
    FlightDate BETWEEN '2024-03-01' AND '2024-03-31'; -- Rang(
```

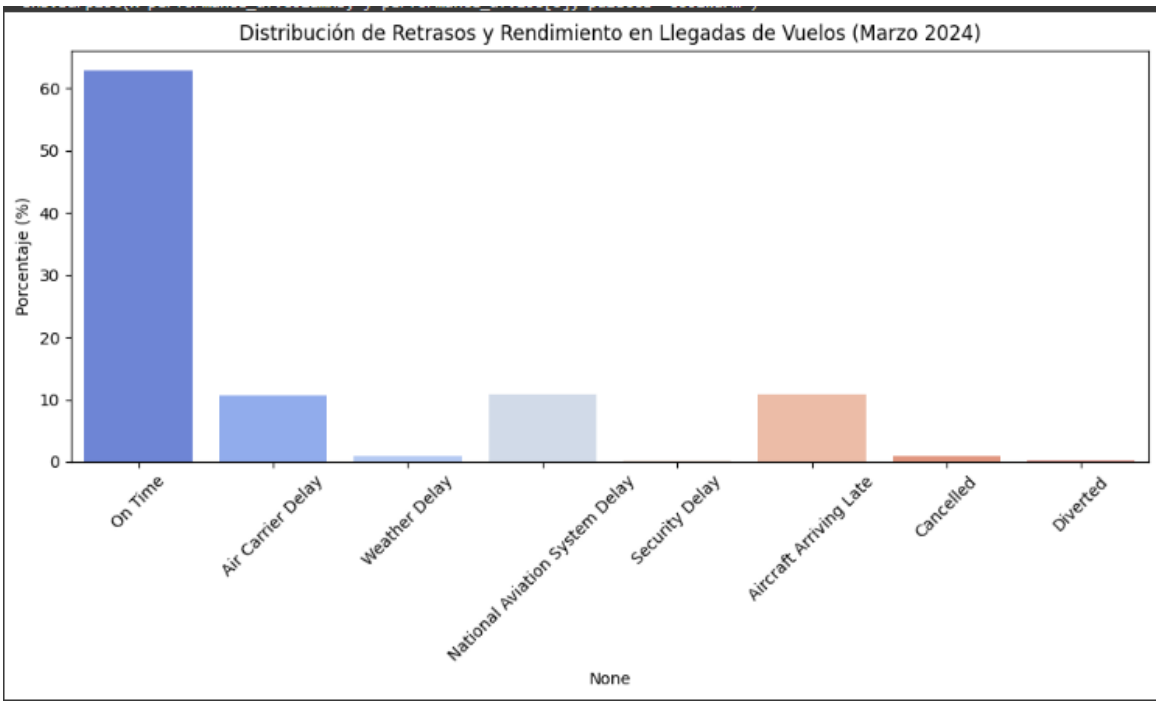
Python

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Especificar la ruta del archivo
csv_file = '/data/airlines/2024_3 - copia.csv'

# Cargar los datos desde el archivo CSV
df = pd.read_csv(csv_file, low_memory=False)

# Verificar las primeras filas del DataFrame
print(df.head())

# Filtrar los datos para un rango de fechas específico (opcio
df['FlightDate'] = pd.to_datetime(df['FlightDate'])
df = df[(df['FlightDate'] >= '2024-03-01') & (df['FlightDate'

# Calcular la distribución de rendimiento de llegada
on_time = (df['ArrDelay'] <= 0).mean() * 100
air_carrier_delay = (df['CarrierDelay'] > 0).mean() * 100
weather_delay = (df['WeatherDelay'] > 0).mean() * 100
nas_delay = (df['NASDelay'] > 0).mean() * 100
security_delay = (df['SecurityDelay'] > 0).mean() * 100
aircraft_arriving_late = (df['LateAircraftDelay'] > 0).mean()
cancelled = (df['Cancelled'] == 1).mean() * 100
diverted = (df['Diverted'] == 1).mean() * 100

# Crear un diccionario con los resultados
performance_distribution = {
    'On Time': on_time,
    'Air Carrier Delay': air_carrier_delay,
    'Weather Delay': weather_delay,
    'National Aviation System Delay': nas_delay,
    'Security Delay': security_delay,
    'Aircraft Arriving Late': aircraft_arriving_late,
    'Cancelled': cancelled,
    'Diverted': diverted
}
```

```
# Convertir el diccionario a un DataFrame para mejor visualiz
performance_df = pd.DataFrame(performance_distribution, index

# Mostrar los resultados
print(performance_df)

# Crear la visualización con seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x=performance_df.columns, y=performance_df.loc[0]

# Agregar etiquetas y título
plt.title('Distribución de Retrasos y Rendimiento en Llegadas
plt.ylabel('Porcentaje (%)')
plt.xticks(rotation=45)

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```



```
   Year  Quarter  Month  DayofMonth  DayOfWeek  FlightDate Reporting_Airline  \
0  2024        1      3           1          5  2024-03-01               9E
1  2024        1      3           2          6  2024-03-02               9E
2  2024        1      3           3          7  2024-03-03               9E
3  2024        1      3           4          1  2024-03-04               9E
4  2024        1      3           5          2  2024-03-05               9E

   DOT_ID_Reporting_Airline IATA_CODE_Reporting_Airline Tail_Number  ...  \
0                     20363                         9E      N935XJ  ...
1                     20363                         9E      N910XJ  ...
2                     20363                         9E      N298PQ  ...
3                     20363                         9E      N602LR  ...
4                     20363                         9E      N348PQ  ...

   Div4TailNum  Div5Airport  Div5AirportID  Div5AirportSeqID Div5WheelsOn  \
0          NaN          NaN            NaN               NaN          NaN
1          NaN          NaN            NaN               NaN          NaN
2          NaN          NaN            NaN               NaN          NaN
3          NaN          NaN            NaN               NaN          NaN
4          NaN          NaN            NaN               NaN          NaN

  Div5TotalGTime Div5LongestGTime  Div5WheelsOff Div5TailNum  Unnamed: 109
0            NaN              NaN            NaN         NaN           NaN
1            NaN              NaN            NaN         NaN           NaN
2            NaN              NaN            NaN         NaN           NaN
3            NaN              NaN            NaN         NaN           NaN
4            NaN              NaN            NaN         NaN           NaN
```



5. **¿Cuáles son los aeropuertos de los que parte más vuelos? (query_5.hql y script_query_5.py)**

HiveQL

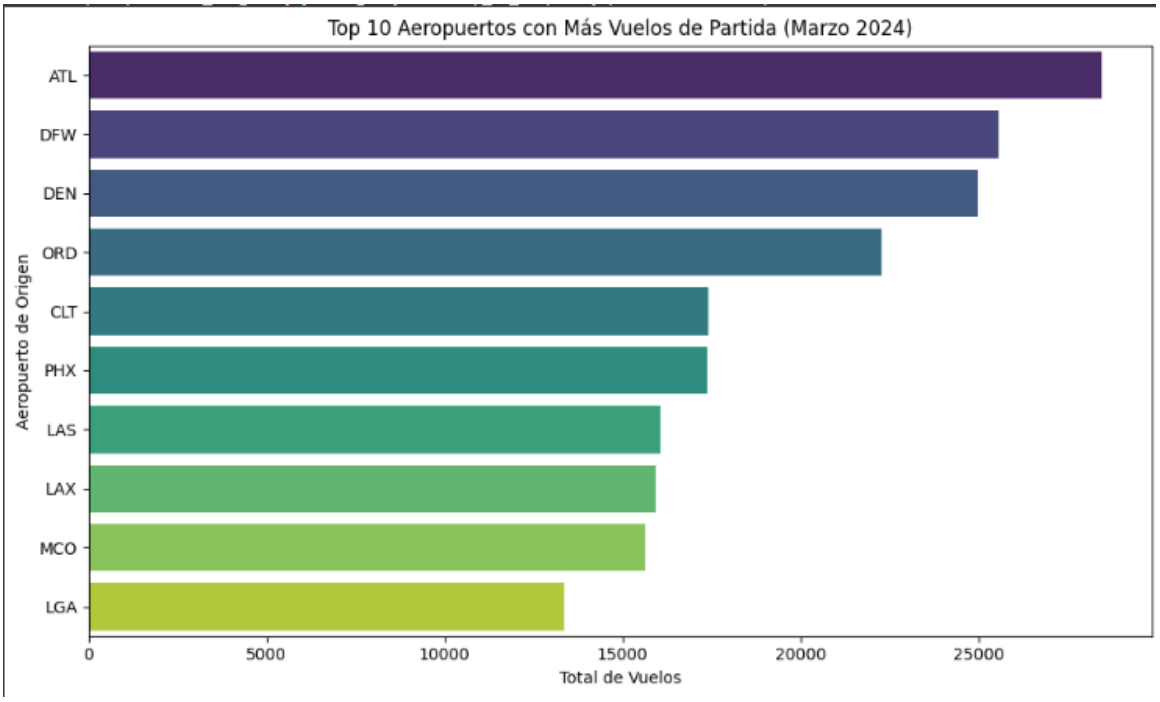
```
-- query_5.hql

USE flights_db;

-- Consulta para obtener los aeropuertos con más vuelos de pa
SELECT Origin, COUNT(*) AS Total_Flights
FROM raw_flights_data
GROUP BY Origin
ORDER BY Total_Flights DESC;
```

Python

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Cargar el archivo CSV
csv_file = '/data/airlines/2024_3 - copia.csv'
```

```
df = pd.read_csv(csv_file, low_memory=False)

# Agrupar por aeropuerto de origen y contar el número de vuel
top_airports = df.groupby('Origin').size().reset_index(name='

# Ordenar por el número de vuelos en orden descendente
top_airports = top_airports.sort_values(by='Total_Flights', a

# Seleccionar los 10 aeropuertos con más vuelos
top_10_airports = top_airports.head(10)

# Mostrar los resultados
print("Top 10 aeropuertos con más vuelos de partida:")
print(top_10_airports)

# Crear una visualización de los 10 aeropuertos con más vuelo
plt.figure(figsize=(10, 6))
sns.barplot(x='Total_Flights', y='Origin', data=top_10_airpor

# Agregar etiquetas y título
plt.title('Top 10 Aeropuertos con Más Vuelos de Partida (Marz
plt.xlabel('Total de Vuelos')
plt.ylabel('Aeropuerto de Origen')

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```

```
Top 10 aeropuertos con más vuelos de partida:
    Origin  Total_Flights
18     ATL          28459
87     DFW          25567
86     DEN          24976
225    ORD          22288
67     CLT          17400
236    PHX          17375
171    LAS          16062
173    LAX          15916
192    MCO          15650
182    LGA          13366
<ipython-input-63-b12285c73510>:24: FutureWarning:
```



6. **¿Cuáles son las rutas de vuelo que sufren mayores demoras? (query_6.hql y script_query_6.py),**

HiveQL

```
-- query_6.hql

USE flights_db;

-- Consulta para obtener las rutas de vuelo con mayores demor
SELECT origin, dest, AVG(depdelay) AS Avg_Departure_Delay
FROM raw_flights_data
WHERE depdelay IS NOT NULL
GROUP BY origin, dest
ORDER BY Avg_Departure_Delay DESC
LIMIT 10;
```

Python

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Cargar el archivo CSV
```

```
csv_file = '/data/airlines/2024_3 - copia.csv'
df = pd.read_csv(csv_file)

# Filtrar filas donde 'DepDelay' no es nulo
df_filtered = df[df['DepDelay'].notnull()]

# Crear una nueva columna para la ruta (combinando origen y d
df_filtered['Route'] = df_filtered['Origin'] + ' - ' + df_fil

# Calcular la demora promedio por ruta
average_delay = df_filtered.groupby('Route')['DepDelay'].mean

# Ordenar por la demora promedio en orden descendente y selec
top_delayed_routes = average_delay.sort_values(by='DepDelay',

# Mostrar el resultado
print("Top 10 rutas con mayor demora promedio en la salida:")
print(top_delayed_routes)

# Visualización de las 10 rutas con más retraso
plt.figure(figsize=(12, 6))
sns.barplot(x='DepDelay', y='Route', data=top_delayed_routes,

# Agregar etiquetas y título
plt.title('Top 10 Rutas con Mayor Demora Promedio en la Salid
plt.xlabel('Demora Promedio de Salida (minutos)')
plt.ylabel('Ruta')

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```

```
Top 10 rutas con mayor demora promedio en la salida:
          Route    DepDelay
1658  DFW - STT  532.200000
2232  HDN - BOS  215.600000
2382  HTS - PIE  214.875000
345   AUS - TYS  180.000000
5658  TYS - AUS  172.000000
2237  HDN - FLL  156.733333
4635  RIC - PIE  156.111111
2140  GJT - LAS  154.000000
5716  XNA - FLL  139.444444
2811  LAS - GJT  136.111111
<ipython-input-65-1748cfe49068>:27: FutureWarning:
```