UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

LIC. MATEMÁTICAS APLICADAS Y COMPUTACIÓN

Programación Paralela y Concurrente

Prof. José Gustavo Fuentes Cabrera

04/05/2025

Practica 2. Práctica Spark

Ramírez Gómez Fernando Axel

No. C. 422066442

índice

Objetivo:

El objetivo principal de este proyecto es aplicar los conceptos fundamentales de Apache Spark (como RDDs, DataFrames, transformaciones y acciones) para analizar datos de movilidad urbana en la CDMX, específicamente registros de viajes en bicicleta compartida.

Introducción:

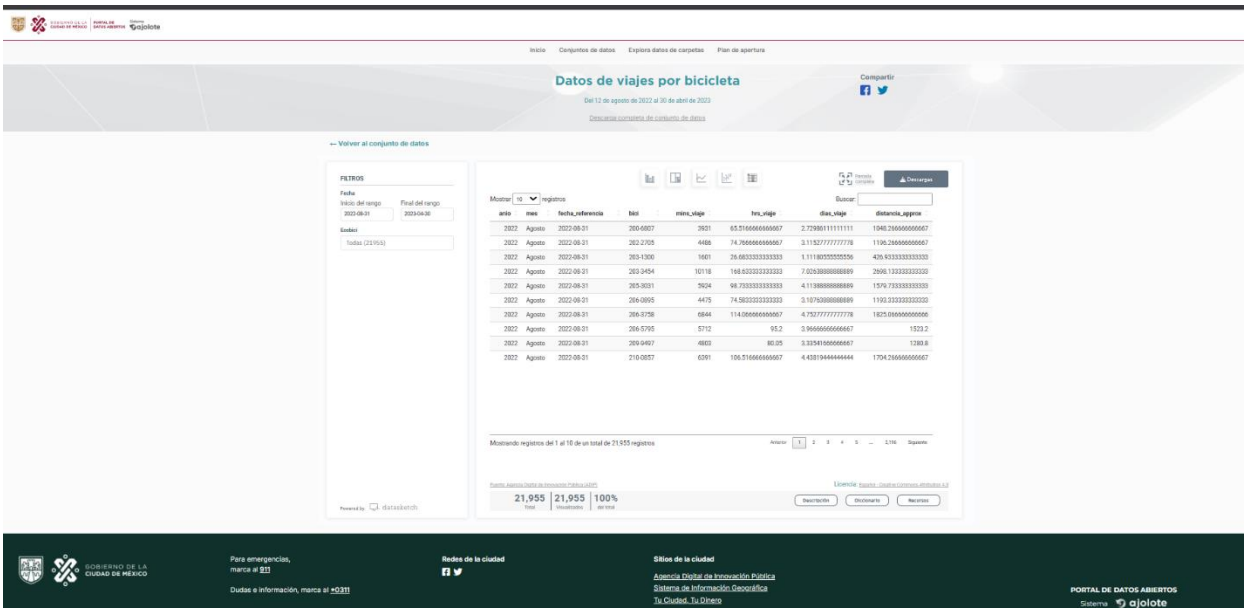
En la Ciudad de México, la movilidad urbana es un tema crítico debido al crecimiento poblacional, la congestión vehicular y la necesidad de alternativas de transporte sostenible. Los sistemas de bicicletas compartidas, como EcoBici, juegan un papel fundamental en la reducción de emisiones y la mejora de la accesibilidad en zonas urbanas. Sin embargo, para optimizar su funcionamiento, es necesario analizar grandes volúmenes de datos que permitan identificar patrones de uso, zonas de alta demanda y eficiencia operativa.

Apache Spark es una herramienta poderosa para el procesamiento distribuido de datos, ideal para manejar conjuntos de información masivos de manera eficiente. En este proyecto, utilizamos Spark para analizar datos de viajes en bicicleta, aplicando transformaciones y acciones sobre RDDs y DataFrames, con el fin de extraer insights valiosos que contribuyan a una mejor gestión del sistema de movilidad.

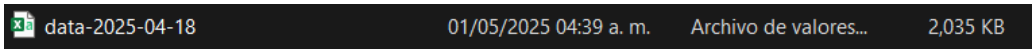
Descripción del Dataset

El dataset utilizado para este análisis consiste en registros individuales de viajes realizados en el sistema EcoBici de la Ciudad de México. Las columnas clave disponibles son:

- anio: Año en que se realizó el viaje.
- mes: Mes en que se realizó el viaje.
- fecha_referencia: Fecha específica del viaje.
- bici: Identificador único de la bicicleta utilizada en el viaje.
- mins_viaje: Duración del viaje en minutos.
- hrs_viaje: Duración del viaje en horas.
- dias_viaje: Duración del viaje en días.
- distancia_approx: Distancia aproximada recorrida durante el viaje.



"Portal de Datos Abiertos de la CDMX. (2023, 25 de mayo)"



Metodología: Operaciones de Análisis y Visualización

El proceso se dividió en dos fases principales: Análisis de Datos utilizando PySpark y Visualización de Resultados utilizando Pandas y Matplotlib/Seaborn

1. Fase de Análisis (ecobici_analysis.py)

En esta fase, se utilizó PySpark para cargar, limpiar y transformar los datos, así como para realizar agregaciones que resumen la información en métricas clave.

- **Carga y Limpieza de Datos:**

  Se cargó el dataset desde un archivo CSV (viajes_individuales.csv) utilizando spark.read.csv(), permitiendo a Spark inferir el esquema de los datos. Se realizaron operaciones básicas de limpieza y transformación como convertir la columna fecha_referencia a un formato de fecha/hora adecuado (to_timestamp), extraer la hora del día (hour) y el día de la semana (date_format), e identificar si el día era fin de semana (when).

```
# Cargar datos
df = spark.read.csv(ruta_csv, header=True, inferSchema=True)

# Validar columnas necesarias (añadido para robustez)
columnas_necesarias = ["fecha_referencia", "bici", "mins_viaje", "distancia_approx"]
for col_name in columnas_necesarias:
    if col_name not in df.columns:
        raise ValueError(f"Columna '{col_name}' no encontrada en el dataset.")
```

- **Análisis de Patrones de Uso por Tipo de Viaje (Rutas Simuladas):**

  Dado que no teníamos datos de estaciones, creamos "rutas simuladas" combinando características de los viajes (duración y distancia aproximada). Esto nos permitió agrupar viajes con perfiles similares (ej. viajes cortos de poca distancia, viajes largos de distancia moderada) para entender qué tipos de trayectos son más frecuentes. Se usaron groupBy() y agg() para contar viajes y calcular promedios de duración y distancia por cada tipo de "ruta simulada".

```python
# Limpieza avanzada con manejo de errores
df = df.withColumn("fecha_hora",
                   to_timestamp(col("fecha_referencia"), "dd/MM/yyyy")) \
       .withColumn("hora", hour(col("fecha_hora"))) \
       .withColumn("dia_semana", date_format(col("fecha_hora"), "EEEE")) \
       .withColumn("es_fin_semana",
                   when(col("dia_semana").isin(["Saturday", "Sunday"]), True).otherwise(False))

# Validación de datos
print("\n=== Resumen de Calidad de Datos ===")
print(f"Registros totales: {df.count():,}")
for columna in df.columns:
    nulos = df.filter(col(columna).isNull()).count()
    print(f"Valores nulos en {columna}: {nulos} ({nulos/df.count():.2%})")
```

- **Análisis de Gestión de la Demanda por Horario:**

Se agruparon los viajes por rangos horarios (rango_hora) definidos (Mañana, Mediodía, Tarde, Noche) para identificar los momentos del día con mayor actividad. Se usaron groupBy() y agg() para contar viajes y calcular promedios por rango horario.

```python
print("2. Análisis de Gestión de la Demanda por Horario...")
df = df.withColumn("rango_hora",
                   when(col("hora").between(6, 9), "Mañana (6-9)")
                   .when(col("hora").between(10, 14), "Mediodía (10-14)")
                   .when(col("hora").between(15, 19), "Tarde (15-19)")
                   .otherwise("Noche (20-5)"))

resultados['horas'] = df.groupBy("rango_hora") \
    .agg(count("*").alias("viajes"),
         avg("mins_viaje").alias("duracion_promedio"),
         sum("distancia_approx").alias("distancia_total")) \
    .orderBy("viajes", ascending=False)
```

- **Análisis de Rendimiento y Eficiencia de la Flota:**

Se analizó el uso y rendimiento de cada bicicleta individual. Se calculó el número total de viajes, distancia total y minutos totales por cada bici, y se derivó una métrica de "eficiencia" como la distancia recorrida por minuto de uso. Esto se realizó con groupBy() y agg().

```python
print("3. Análisis de Rendimiento y Eficiencia de la Flota...")
resultados['eficiencia'] = df.groupBy("bici") \
    .agg(count("*").alias("viajes"),
         sum("distancia_approx").alias("distancia_total"),
         sum("mins_viaje").alias("minutos_totales")) \
    .withColumn("eficiencia_km_min",
                when(col("minutos_totales") > 0, col("distancia_total")/col("minutos_totales"))
                .otherwise(0)) \
    .orderBy(col("eficiencia_km_min").desc())
```

- **Guardado de Resultados:**

Los resultados de los análisis se guardaron en formato CSV en la carpeta resultados. Se utilizó coalesce(1) para asegurar que cada resultado se escribiera en un solo archivo "part-" dentro de una carpeta con el nombre del resultado (ej. resultados/rutas.csv/part-00000-...). La funciónwrite.csv(..., mode="overwrite")` se usó para sobrescribir resultados anteriores.

```
# Coalesce(1) para escribir en un solo archivo part- dentro de la carpeta
df.coalesce(1).write.csv(ruta_carpeta_spark, header=True, mode="overwrite")
```

2. Fase de Visualización (ecobici_visualization.py)

En esta fase, se utilizó la librería Pandas para cargar los resultados guardados por Spark y las librerías Matplotlib y Seaborn para generar visualizaciones.

- **Carga de Resultados:**

Se leyeron los datos de las carpetas CSV generadas por Spark. Para esto, se identificaron las carpetas (rutas.csv, horas.csv, eficiencia.csv) y se usó la librería glob para encontrar los archivos part-*.csv dentro de cada carpeta. Estos archivos se leyeron con pd.read_csv() y se concatenaron en DataFrames de Pandas.

```
archivos_part = glob.glob(os.path.join(ruta_carpeta_spark, 'part-*.csv'))

if archivos_part:
    print(f" - Encontrados {len(archivos_part)} archivo(s) 'part-' para '{nombre_clave}'. Leyendo...")
```

- **Generación de Gráficas:**

  Se crearon gráficos de barras utilizando Matplotlib y Seaborn para visualizar los principales hallazgos: el top N de tipos de viaje más frecuentes, la distribución de viajes por rango horario y el top N de bicicletas más eficientes. Cada gráfica se personalizó con títulos, etiquetas y se guardó como un archivo PNG en la carpeta visualizaciones.

```
ax = sns.barplot(x='total_viajes', y='ruta_simulada', data=rutas_top, palette='viridis')

for index, row in rutas_top.iterrows():
    ax.text(row['total_viajes'], index, f" {row['total_viajes']:,}", color='black', va='center')

plt.title(f'Impacto en el Negocio: Top {top_n} Tipos de Viaje Más Frecuentes', pad=20) # Título con valor de negocio
plt.xlabel('Número Total de Viajes')
plt.ylabel('Tipo de Viaje (Duración ~Hrs / Distancia ~Km)') # Etiqueta más descriptiva
plt.tight_layout()
ruta_salida = os.path.join(directorio_visualizaciones, 'impacto_tipos_viaje.png') # Nombre de archivo descriptivo
plt.savefig(ruta_salida, dpi=300, bbox_inches='tight')
plt.close()
```

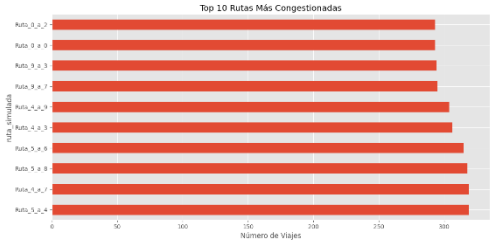
Análisis de los Resultados y Valor de Negocio

El análisis de los resultados obtenidos y plasmados en las visualizaciones proporciona información valiosa para un gerente del sistema EcoBici:

- **Patrones de Uso por Tipo de Viaje:**

La gráfica de "Top rutas que han tenido la mayor cantidad de viajes más Frecuentes" permite ver rápidamente qué combinaciones de duración y distancia son las más populares.

```
Top 5 Tipos de Viaje (Rutas Simuladas) más frecuentes:
+-----------------------+-----------+-----------------+-----------------+
|ruta_simulada          |total_viajes|tiempo_promedio  |distancia_promedio|
+-----------------------+-----------+-----------------+-----------------+
|Duracion_~57hrs_Dist_~0km|661        |3449.30158850227 |919.813756933938 |
|Duracion_~55hrs_Dist_~0km|655        |3329.4490585241724|887.8530822731121|
|Duracion_~54hrs_Dist_~0km|650        |3269.0170512820537|871.7378803418802|
|Duracion_~53hrs_Dist_~0km|634        |3211.216955835964|856.3245215562564|
|Duracion_~56hrs_Dist_~0km|631        |3388.9271526677217|903.7139073780588|
+-----------------------+-----------+-----------------+-----------------+
only showing top 5 rows
```



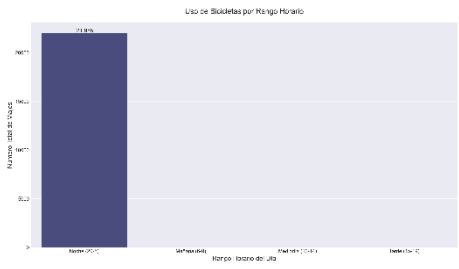
Esto ayuda a entender el comportamiento típico del usuario (por ejemplo, si predominan los viajes cortos de menos de 1 km o los viajes más largos de varios kilómetros).

- **Demanda Horaria:**

La gráfica de "Demanda de Viajes por Rango Horario" identifica claramente los períodos del día con mayor y menor actividad. Esto fue pensado más para un director del proyecto que puede usar esto para:

  - Programar el rebalanceo de bicicletas para asegurar disponibilidad en las horas pico (por ejemplo, en la mañana y la tarde).
  - Planificar el mantenimiento en horas de baja demanda (por la noche o durante el mediodía si la demanda es menor).
  - Ajustar la dotación de personal de soporte o atención al cliente.

```
Uso de Bicicletas por Rango Horario:
+------------+------+-----------------+------------------+
|rango_hora  |viajes|duracion_promedio|distancia_total   |
+------------+------+-----------------+------------------+
|Noche (20-5)|21955 |3701.601623775898|2.16716436399999E7|
+------------+------+-----------------+------------------+
```

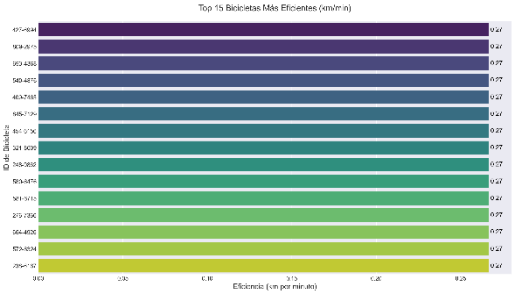


- **Rendimiento de la Flota:**

La gráfica de "Top Bicicletas Más Eficientes" (basado en distancia por minuto) y el análisis de viajes por bicicleta identifican las unidades que están teniendo mayor uso o que muestran un rendimiento superior. Esto sirve para:

- Priorizar bicicletas para inspección y mantenimiento preventivo basado en el uso intensivo.
- Investigar bicicletas con eficiencia inusualmente baja, que podrían requerir reparación o indicar problemas de datos.
- Monitorear la distribución del uso entre la flota para asegurar que todas las bicicletas estén activas.

```
Top 5 Bicicletas más eficientes (km/min):
+--------+------+------------------+------------------+--------------------+
|bici    |viajes|distancia_total   |minutos_totales   |eficiencia_km_min   |
+--------+------+------------------+------------------+--------------------+
|427-6994|1     |274.47555555555556|1029.28333333333  |0.26666666666666755|
|809-2975|1     |323.91555555555556|1214.68333333333  |0.2666666666666674 |
|693-4368|1     |395.99555555555554|1484.98333333333  |0.2666666666666673 |
|540-4876|4     |455.82222222222225|1709.3333333333296|0.2666666666666673 |
|845-7129|2     |526.6355555555556 |1974.8833333333298|0.26666666666666716|
+--------+------+------------------+------------------+--------------------+
only showing top 5 rows
```



En conjunto, estos análisis permiten pasar de datos brutos a insights accionables para optimizar la operación, mejorar la experiencia del usuario y tomar decisiones informadas sobre la gestión de la flota y los recursos.

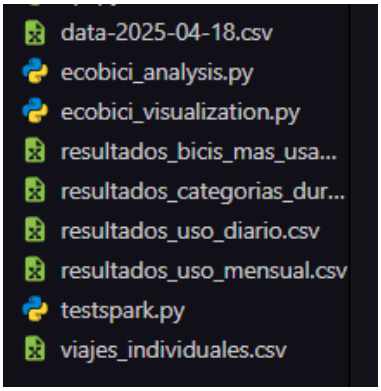
Salida de Terminal:

En este caso estamos analizando que en nuestra base de datos contenemos 21,955 datos, de la cual vamos a analizarlos con la ayuda de Spark.

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  POSTMAN CONSOLE  DEBUG CONSOLE
ha sido terminado.

PS C:\Users\ferna\Desktop\Eco_Bici> python .\ecobici_analysis.py
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
=== Cargando y procesando datos ===

=== Resumen de Calidad de Datos ===
Registros totales: 21,955
Valores nulos en anio: 0 (0.00%)
Valores nulos en mes: 0 (0.00%)
Valores nulos en fecha_referencia: 0 (0.00%)
Valores nulos en bici: 0 (0.00%)
Valores nulos en mins_viaje: 0 (0.00%)
Valores nulos en hrs_viaje: 0 (0.00%)
Valores nulos en dias_viaje: 0 (0.00%)
Valores nulos en distancia_approx: 0 (0.00%)
Valores nulos en fecha_hora: 0 (0.00%)
Valores nulos en hora: 0 (0.00%)
Valores nulos en dia_semana: 0 (0.00%)
Valores nulos en es_fin_semana: 0 (0.00%)

=== Muestra de datos procesados ===
+--------+----------+------------------+-------------------+----+----------+
|    bici|mins_viaje|  distancia_approx|        fecha_hora|hora|dia_semana|
+--------+----------+------------------+-------------------+----+----------+
|200-6807|    3931.0|1048.2666666666669|2022-08-31 00:00:00|   0| Wednesday|
|202-2705|    4486.0|1196.2666666666669|2022-08-31 00:00:00|   0| Wednesday|
|203-1300|    1601.0|426.93333333333334|2022-08-31 00:00:00|   0| Wednesday|
|203-3454|   10118.0| 2698.133333333333|2022-08-31 00:00:00|   0| Wednesday|
|205-3031|    5924.0|1579.7333333333331|2022-08-31 00:00:00|   0| Wednesday|
+--------+----------+------------------+-------------------+----+----------+
only showing top 5 rows
```

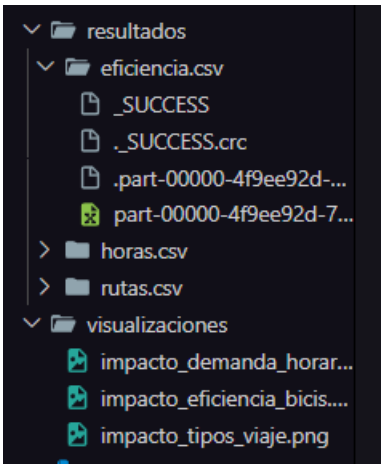
```
PROBLEMS  OUTPUT  TERMINAL  PORTS  POSTMAN CONSOLE  DEBUG CONSOLE


=== Ideas de Valor de Negocio (Basadas en el Análisis) ===
- Con este análisis, un gerente puede:
  - Identificar las bicicletas más usadas y eficientes para optimizar mantenimiento y gestión de flota.
  - Entender los patrones de uso por horario para planificar operaciones y rebalanceo de bicicletas.
  - Tener una idea de los tipos de viajes más comunes (por duración/distancia) para entender mejor a los usuarios.
  - Monitorear el rendimiento general del sistema.

=== Realizando análisis de valor de negocio ===
1. Análisis de Patrones de Uso por Tipo de Viaje (Rutas Simuladas)...
2. Análisis de Gestión de la Demanda por Horario...
3. Análisis de Rendimiento y Eficiencia de la Flota...

=== Resúmenes de análisis detallado ===

Top 5 Tipos de Viaje (Rutas Simuladas) más frecuentes:
+------------------------+------------+------------------+------------------+
|ruta_simulada           |total_viajes|tiempo_promedio   |distancia_promedio|
+------------------------+------------+------------------+------------------+
|Duracion_~57hrs_Dist_~0km|661         |3449.30158850227  |919.813756933938  |
|Duracion_~55hrs_Dist_~0km|655         |3329.4490585241724|887.8530822731121 |
|Duracion_~54hrs_Dist_~0km|650         |3269.0170512820537|871.7378803418802 |
|Duracion_~53hrs_Dist_~0km|634         |3211.216955835964 |856.3245215562564 |
|Duracion_~56hrs_Dist_~0km|631         |3388.9271526677217|903.7139073780588 |
+------------------------+------------+------------------+------------------+
only showing top 5 rows


Uso de Bicicletas por Rango Horario:
+------------+------+----------------+------------------+
|rango_hora  |viajes|duracion_promedio|distancia_total   |
+------------+------+----------------+------------------+
|Noche (20-5)|21955 |3701.601623775898|2.16716436399999E7|
+------------+------+----------------+------------------+


Top 5 Bicicletas más eficientes (km/min):
+--------+------+------------------+------------------+--------------------+
|bici    |viajes|distancia_total   |minutos_totales   |eficiencia_km_min   |
+--------+------+------------------+------------------+--------------------+
|427-6994|1     |274.47555555555556|1029.28333333333  |0.26666666666666755|
|809-2975|1     |323.91555555555556|1214.68333333333  |0.2666666666666674 |
|693-4368|1     |395.99555555555554|1484.98333333333  |0.2666666666666673 |
|540-4876|4     |455.82222222222225|1709.3333333333296|0.2666666666666673 |
|845-7129|2     |526.6355555555556 |1974.8833333333298|0.26666666666666716|
+--------+------+------------------+------------------+--------------------+
only showing top 5 rows
```

```
=== Guardando resultados para Visualización ===
 - Intentando guardar 'rutas' como carpeta 'resultados\rutas.csv'
 - 'rutas' guardado correctamente como carpeta 'rutas.csv'.
 - Intentando guardar 'horas' como carpeta 'resultados\horas.csv'
 - 'horas' guardado correctamente como carpeta 'horas.csv'.
 - Intentando guardar 'eficiencia' como carpeta 'resultados\eficiencia.csv'
 - 'eficiencia' guardado correctamente como carpeta 'eficiencia.csv'.

=== Sesión de Spark detenida ===

=== Proceso de análisis finalizado ===
PS C:\Users\ferna\Desktop\Eco_Bici> CORRECTO: el proceso con PID 3316 (proceso secundario de PID 20132)
ha sido terminado.
CORRECTO: el proceso con PID 20132 (proceso secundario de PID 24940)
ha sido terminado.
CORRECTO: el proceso con PID 24940 (proceso secundario de PID 33940)
ha sido terminado.
```



Salida para su visualización gráfica:

```
PS C:\Users\ferna\Desktop\Eco_Bici> python .\ecobici_visualization.py
=== Cargando datos para visualización desde 'resultados' ===
 - Buscando datos para 'rutas' en la carpeta 'resultados\rutas.csv'
 - Encontrados 1 archivo(s) 'part-' para 'rutas'. Leyendo...
 - Datos para 'rutas' cargados correctamente.
 - Buscando datos para 'horas' en la carpeta 'resultados\horas.csv'
 - Encontrados 1 archivo(s) 'part-' para 'horas'. Leyendo...
 - Datos para 'horas' cargados correctamente.
 - Buscando datos para 'eficiencia' en la carpeta 'resultados\eficiencia.csv'
 - Encontrados 1 archivo(s) 'part-' para 'eficiencia'. Leyendo...
 - Datos para 'eficiencia' cargados correctamente.
=== Carga de datos completada ===

=== Generando visualizaciones de valor de negocio en 'visualizaciones' ===
C:\Users\ferna\Desktop\Eco_Bici\ecobici_visualization.py:77: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.barplot(x='total_viajes', y='ruta_simulada', data=rutas_top, palette='viridis')
 - 'impacto_tipos_viaje.png' generado (Impacto en tipos de viaje)
C:\Users\ferna\Desktop\Eco_Bici\ecobici_visualization.py:104: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.barplot(x='rango_hora', y='viajes', data=datos_horas_ordenado, palette='viridis')
 - 'impacto_demanda_horaria.png' generado (Impacto en demanda horaria)
C:\Users\ferna\Desktop\Eco_Bici\ecobici_visualization.py:129: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.barplot(x='eficiencia_km_min', y='bici', data=eficiencia_top, palette='viridis')
 - 'impacto_eficiencia_bicis.png' generado (Impacto en eficiencia de bicis)
```



Reflexión sobre la Distribución y Eficiencia

La arquitectura utilizada, combinando Spark con Pandas/Matplotlib, es eficiente para este tipo de análisis, pero es importante entender qué partes son distribuidas y cuáles no:

Partes eficientemente distribuidas (Spark):

- La carga inicial del dataset desde CSV (spark.read.csv).
- Las operaciones de limpieza y transformación a nivel de fila (withColumn, to_timestamp, hour, date_format, when, concat).
- Las operaciones de agregación y agrupamiento a gran escala (groupBy, agg, count, avg, sum, orderBy).
- El guardado de los resultados agregados en CSV (write.csv). Estas operaciones son donde Spark brilla, distribuyendo la carga de trabajo a través de múltiples nodos (aunque en entorno local, se distribuye en "núcleos lógicos"), lo que es crucial para datasets grandes que no caben en la memoria de una sola máquina o que requieren procesamiento intensivo.

Partes no distribuidas (Pandas, Matplotlib/Seaborn):

- La carga de los resultados agregados *desde* los archivos CSV por parte del script de visualización (pd.read_csv, pd.concat).
- Las operaciones de manipulación y filtrado de datos dentro de Pandas (ej. .head(), .sort_values(), .reset_index()).
- La generación de las gráficas utilizando Matplotlib y Seaborn. Estas partes se ejecutan en una única máquina (la máquina donde corres el script de

visualización) y en la memoria RAM de esa máquina. Esto es adecuado porque los resultados agregados (los DataFrames finales como rutas, horas, eficiencia) son mucho más pequeños que el dataset original y caben fácilmente en la memoria de un solo nodo. Sería ineficiente (e innecesario) intentar distribuir la generación de una sola imagen.

La combinación es poderosa: usar Spark para el procesamiento pesado y distribuido, y luego herramientas de visualización locales para presentar los resultados ya agregados y reducidos.

Conclusion

Este proyecto demostró la capacidad de utilizar Apache Spark para procesar y analizar un dataset real de EcoBici en un entorno local. Los análisis de patrones de uso por tipo de viaje, demanda horaria y eficiencia de la flota proporcionaron insights concretos que son directamente aplicables para un gerente del sistema, permitiendo la optimización de operaciones, el mantenimiento predictivo y una mejor comprensión del comportamiento del usuario.

Se resaltó la ventaja de Spark para el procesamiento distribuido en la fase de análisis de datos grandes, mientras que herramientas como Pandas y Matplotlib fueron eficientes para la visualización de los resultados agregados. La documentación del proceso, incluyendo los fragmentos clave de código y el uso de IA como apoyo a la depuración y estructuración, asegura la transparencia y reproducibilidad del trabajo.

Uso Ético y Documentado de IA como Herramienta Auxiliar

Durante el desarrollo de esta práctica, se utilizó una Inteligencia Artificial conversacional (Gemini) como herramienta auxiliar para diversos fines, siempre de forma ética y documentada:

- **Comprensión de conceptos:** Ayuda para aclarar dudas sobre operaciones específicas de PySpark.
- **Depuración de errores:** Asistencia crucial para identificar y resolver errores de código, particularmente problemas de configuración del entorno (como la necesidad de HADOOP_HOME en Windows.
- **Explicación de fragmentos de código:** Obtener descripciones detalladas sobre cómo funcionan ciertas partes del código o la sintaxis específica.

- **Generación de fragmentos de código:** Corrección de la lectura de archivos part- o la adición de comentarios y mensajes explicativos.

Referencias:

- Agencia Digital de Innovación Pública. (2023, 25 de mayo). *Datos de bicicletas (Ecobici)* [Conjunto de datos]. Portal de Datos Abiertos de la CDMX. https://datos.cdmx.gob.mx/dataset/datos-de-bicicletas-ecobici/resource/aed25f31-1f38-4eea-846e-95a98c48a045
- Google. (2025, 3 Mayo). *Gemini* [Large language model]. https://gemini.google.com/