Tarea #2 - Big Data

Objetivo

Introducir a los estudiantes al procesamiento de tipos de datos complejos, agregaciones y métricas con Apache Spark.

Resultados esperados

Para esta asignación se espera que los estudiantes concluyan dos entregables relacionados:

- Un programa principal que recibirá como entrada un patrón de archivos tipo JSON que contienen los detalles de viajes relacionados por los conductores de Diber, una plataforma digital ficticia de transporte de personas. El programa principal generará varios archivos de salida con información procesada de los viajes.
- Una serie de pruebas unitarias que permitan corroborar la correctitud de las diferentes funciones internas al programa.

Entrega: Archivo comprimido con código y PDF en TEC Digital a más tardar el 11 de abril de 2021 a las 11:00 PM

Datos de entrada

Cada archivo de entrada será un objeto JSON que describe todos los viajes realizados por cada conductor(a) de Diber. La expectativa es que cada corrida del programa cargará un número posiblemente grande de estos archivos.

El formato de cada archivo es:

- Un atributo **identificador** numérico que sirve como valor número único para identificar a la persona
- Un nodo **viajes** que contiene la lista de cada uno de los viajes realizados.
- **kilometros** que representa, incluso de forma decimal, la cantidad de kilómetros recorridos en un viaje
- **precio_kilometro** que corresponde a precio por del viaje específico

El siguiente es un ejemplo del contenido que tendría un archivo:

```
{
	"identificador":	"78625",
	"viajes": [
		{
			"codigo_postal_origen": "11504",
			"codigo_postal_destino": "11501",
			"kilometros": "2.8",
			"precio_kilometro": "550"
		},
		{
			"codigo_postal_origen": "10101",
			"codigo_postal_destino": "60101",
			"kilometros": "96.3",
			"precio_kilometro": "300"
		}
	]
}
```

Para la ejecución del programa principal, los estudiantes deberán proveer 5 archivos de prueba con al menos 10 viajes diferentes cada uno. Los archivos deben contener JSON válido.

Programa principal (25 puntos)

Se espera que los estudiantes entreguen un manual en PDF con las instrucciones para ejecutar el programa principal. Idealmente esto debería realizarse con una simple llamada a "spark-submit programaestudiante.py persona*.json"

Cualquier detalle necesario para la ejecución debe agregarse en este documento. La imposibilidad de ejecución del programa impedirá la obtención de los puntos.

El producto de la ejecución del programa será una serie de archivos de texto:

- total_viajes.csv: contiene 3 columnas que representan 1) el código postal, 2) si es **origen** o **destino** y 3) la cantidad total de viajes para ese código postal como destino u origen.
- total_ingresos.csv: contiene 3 columnas que representan 1) el código postal, 2) si es **origen** o **destino** y 3) la cantidad de dinero generado en ingresos para ese código postal como destino u origen.
- metricas.csv: contiene 2 columnas que representan el tipo de métrica y su valor. En particular deberá generarse las siguientes métricas
    - persona_con_mas_kilometros: identificador de la persona con más kilómetros recorridos

    - persona_con_mas_ingresos: identificador de la persona con más dinero generado
    - percentil_25: si se ordenan todas las personas de menor cantidad de ventas a mayor, cuál valor monetario representa el percentil 25
    - percentil_50
    - percentil_75
    - codigo_postal_origen_con_mas_ingresos: el número que identifica el código postal de origen que generó la mayor cantidad de ingresos a las personas
    - codigo_postal_destino_con_mas_ingresos: el número que identifica el código postal de destino que generó la mayor cantidad de ingresos a las personas

Pruebas esperadas

Para realizar las pruebas unitarias se espera que los estudiantes piensen en las diferentes partes necesarias para conseguir el objetivo final. Éstas deberán arrancar de datos que se encuentren en memoria. En este caso, pueden arrancar de dataframes en los que cada fila es el string del JSON para una caja.

Los estudiantes deberán diseñar sus propias pruebas unitarias, utilizando la discusión en clase como base para guiar su diseño. Para efectos de evaluación se espera que haya suficientes pruebas para probar las diferentes áreas funcionales. Se espera que cada área funcional tenga su propia función de entrada:

- **Total de viajes: 20 puntos**
- **Total de ingresos: 20 puntos**
- **Métricas: 5 puntos cada una**

Las pruebas deben cubrir casos excepcionales. Tanto el profesor y asistente se reservan el derecho de agregar pruebas unitarias adicionales en cada apartado para asegurar el correcto funcionamiento.

Se recuerda a los estudiantes que la nota será completamente derivada de las pruebas unitarias. Deberá ser posible ejecutar las pruebas simplemente al correr el comando pytest en la carpeta que se entrega con el código.

Agregados en Base de Datos (15 Puntos extra)

En un ambiente de producción se esperaría que los resultados no sean almacenados en archivos CSV, sino en algún servicio sencillo de consultar. Como punto opcional de la

asignación se plantea a los estudiantes reutilizar el código hecho para generar el archivo CSV de métricas y expandirlo. Los requerimientos son los siguientes:

- Permitir que cada viaje tenga un atributo opcional **fecha**, con formato "YYYY/MM/DD" (año, mes y día). Dado que es opcional, los estudiantes deben asegurarse que los archivos de prueba pedidos previamente puedan servir como entrada al programa principal. Esto es, no será válido que la información para los puntos extra tenga que ser provista para revisar la tarea base.
- Debe existir código que produzca un dataframe con tres columnas: las dos columnas originales de métricas (nombre de métrica y valor) además de una tercera que representará la fecha, extraída del atributo JSON opcional, mencionado en el punto anterior. Éste debe ser probado con pytest.
- Debe proveerse un script de SQL para crear una tabla en postgresql donde se insertarán los datos.
- Crear un programa principal, aparte del solicitado para la tarea primaria, que reciba por parámetros de consola el nombre de los archivos JSON y los parámetros de conexión necesarios para insertar en la base de datos (dirección de la máquina o host, usuario, password y nombre de la base de datos). Se pide que se parametrice ya que, para la revisión, se ejecutará el script de SQL para crear la tabla en la base de datos y, posteriormente, se dará al programa principal la información de la base de datos donde se creó la tabla.
- Agregar una sección adicional en el manual explicando el uso del programa y cambios realizados.

Condiciones para puntos extra:

- Debe ser posible cumplir con los requerimientos básicos, aparte de los puntos extra. Por ejemplo, los estudiantes deben proveer los archivos JSON sin atributos de fechas y toda la infraestructura debe funcionar igual. Lo anterior a manera de ejemplo. En general, no se podrá justificar errores en la tarea primaria debido a código para puntos extra
- Los estudiantes deben proveer archivos JSON, adicionales, que sí incluyan fechas.
- Se debe crear un archivo "main" aparte.
- Los estudiantes deben documentar en detalle en el archivo PDF qué diferencias de implementación hubo, además de documentar cómo ejecutar el programa primario adicional.
- Deben proveer pruebas unitarias que revisen los dataframes de 3 columnas mencionadas para poder optar por los puntos extra.