Python and MongoDB Project for Beginners with Source Code - Part 2

Business Overview

MongoDB is a document database used to develop highly accessible and scalable web applications. Its flexible schema technique is popular among agile development teams. With drivers for all popular programming languages, MongoDB enables you to launch your application immediately without wasting time setting up a database.

MongoDB's document database architecture makes it simple for programmers to store both structured and unstructured data. It stores documents in a format akin to JSON. Since developers don't have to worry about normalizing data, this format naturally maps to native objects in most contemporary programming languages. MongoDB can scale vertically and horizontally to support heavy data loads and handle the huge volume.

This is the second project in the MongoDB project series. In the first project, we studied about MongoDB database, which is essentially a NoSQL service, its basic architecture, the data modeling used behind MongoDB, and how querying and aggregations can be performed using MongoDB. We also understood the different tools supported by MongoDB that can be leveraged to perform various tasks like Data Analytics, ETL, and Data Visualization. In this project, we will understand how to use PySpark connector with MongoDB using Apache Sedona. We will also perform some advanced analysis using Ranks and Window functions.

Dataset Description:

We will use the transportation dataset, which contains information about the driver details, route details, city details, truck details, etc. The dataset has various tables in the following manner:

1. **driver_details:** The driver details table contains information about 1300 truck drivers involved in the study across nine fields.
    1.1. 'driver_id': unique identification for each driver
    1.2. 'name': name of the truck driver
    1.3. 'gender': gender of the truck driver
    1.4. 'age': age of the truck driver
    1.5. 'experience': experience of the truck driver in years
    1.6. 'driving_style': driving style of the truck driver, conservative or proactive
    1.7. 'ratings': average rating of the truck driver on a scale of 1 to 10
    1.8. 'vehicle_no': the number of the driver's truck
    1.9. 'average_speed_mph': average speed the truck driver in miles per hour

2. **truck_details:** The truck details table contains information about 1300 trucks in the study across five fields.
    2.1. 'truck_id': the unique identification number of the truck

2.2. 'truck_age': age of the truck in years
2.3. 'load_capacity_pounds': loading capacity of the truck in years
2.4. 'mileage_mpg': mileage of the truck in miles per gallon
2.5. 'fuel_type': fuel type of the truck

3. **city_details:** The city details table contains the information for 49 cities.
   3.1. 'city_id': the unique identification number of the city
   3.2. 'city_name': name of the city
   3.3. 'lat': latitude
   3.4. 'lon': longitude

4. **route_details:** The route details table contains information for 2352 different routes followed by the trucks
   4.1. 'route_id': the unique identifier of the routes
   4.2. 'origin_city_id': the city identification number for the origin city
   4.3. 'destination_city_id': the city identification number for the destination
   4.4. 'distance(Miles)': the distance between the origin and destination cities in miles
   4.5. 'average_hours': average time needed to travel from the origin to the destination in hours

5. **truck_schedule_data:** The truck schedule data contains historical information of the trucks scheduled in the period of two months 2019-01-01 to 2019-02-28 and if arrival was delayed. It contains 16600 rows
   5.1. 'truck_id': the unique identifier of the truck
   5.2. 'route_id': the unique identifier of the route
   5.3. 'departure_date': departure DateTime of the truck
   5.4. 'estimated_arrival': estimated arrival DateTime of the truck
   5.5. 'delay': binary variable if the truck's arrival was delayed, 0 for on-time arrival and 1 for delayed arrival

Tech Stack

➔
Language: PySpark
➔
Package: PyMongo
➔
Services: MongoDB Atlas, MongoDB Compass

Key Takeaways

- Understanding the NoSQL Database
- Understanding the importance of MongoDB Database
- Understanding the Mongo-Spark connector
- Loading data to MongoDB using MongoDB Compass
- Setting up the Mongo-Spark environment
- Understanding the dataset
- Understanding the Apache Sedona
- Understanding the need for Apache Sedona
- Perform queries on data
- Understanding the different CRUD operations
- Implementation of And, Or operators in Mongo query
- Understanding the use of Regex in Mongo query
- Perform EDA on the dataset using PySpark
- Perform Sedona Spatial Analysis
- Understanding the Rank and Window functions
- Perform Advanced Analysis using PySpark