Telecommunications Fraud Detection Using MongoDB and Python

Project Deliverable

- A GitHub repository with a python file (.py) with your solution.

Problem Statement

Telecommunications companies need to detect fraudulent activities such as unauthorized use of premium services or fake billing. Building a data pipeline with MongoDB and Python could help identify suspicious activity by extracting data from billing systems, call logs, and other sources, transforming the data to identify patterns or anomalies, and storing it in MongoDB for further analysis.

Background Information

Telecommunications companies generate a vast amount of data daily, which can be used to detect fraud. Fraudulent activity can lead to substantial financial losses and damage the company's reputation. With the help of data pipelines, companies can detect fraud before it escalates.

Guidelines

We will build a data pipeline with three main functions: extraction, transformation, and loading. The pipeline will extract data from CSV files, transform it to identify suspicious activities, and load it into MongoDB. We will use Python as the programming language, and Pymongo as the driver to interact with MongoDB.

- **Sample Datasets for Extraction:** We will use sample call log data in CSV format as the dataset for extraction. The dataset will include fields such as call duration, call type, phone number, and time stamp.
- **Extraction Function:** The extraction function will read data from CSV files and insert it into MongoDB. To optimize the data pipeline, we can use connection pooling to maintain open connections to the database. To secure the pipeline, we can use SSL encryption to encrypt the data transmitted between the client and server. We can also use logging to monitor the extraction process for errors and potential security breaches. Here are some guidelines for the extraction function:
    - Use the pandas library to read the input CSV files.

    - Clean and preprocess the data by removing duplicates, handling missing values, and converting data types.
    - Use connection pooling to optimize performance.
    - Use SSL encryption to secure the pipeline.
    - Log errors and activities using the Python logging module.
- **Transformation Function:** The transformation function will identify suspicious activities based on the data extracted from CSV files. We can use techniques such as aggregation and grouping to identify patterns and anomalies in the data. To optimize the data pipeline, we can use indexes to speed up the data retrieval process. To secure the pipeline, we can use data masking to protect sensitive data. Here are some guidelines for the transformation function:
    - Clean the data and handle missing values.
    - Group and aggregate the data by customer, location, time, and other relevant parameters.
    - Identify patterns in the data to detect suspicious activity, such as unauthorized use of premium services, fake billing, or international calls.
    - Use data compression techniques to optimize performance and reduce storage requirements.
    - Use the Python logging module to log errors and activities.
- **Loading Function:** The loading function will insert the transformed data into MongoDB. We can use batch inserts to improve performance and reduce network traffic. To optimize the data pipeline, we can use sharding to distribute the data across multiple shards and balance the load. To secure the pipeline, we can use authentication and authorization to restrict access to the data. Here are some guidelines for the loading function:
    - Use the pymongo library to connect to the MongoDB instance.
    - Create a new MongoDB collection for each data source.
    - Create indexes on the collection to optimize queries and performance.
    - Use bulk inserts to optimize performance.
    - Use the write concern option to ensure that data is written to disk.
    - Use the Python logging module to log errors and activities.

You can use the guiding file (https://bit.ly/3lV9fW3) as a starting point for your data pipeline.

Sample Datasets for Data Extraction

Here are some sample datasets (https://bit.ly/3YRn7z4) you can use for extraction:

- Call logs
- Billing systems