Activity 5 Documentation

Introduction:

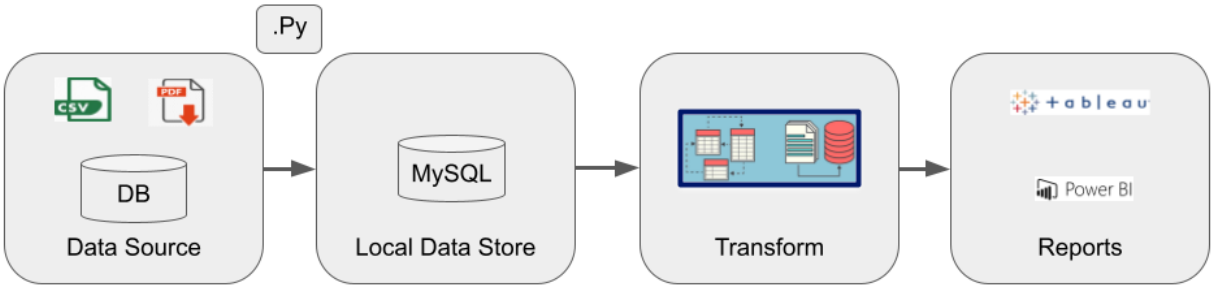
This document lists the requirements for Activity5. This activity is an introduction to the basics of ELT(Extract, Load and Transform) operations done by data engineers using python. We will be considering various data sources for our activity, based on which, we will be creating connectors in python. These connectors will be used to query data from data sources and then store them in our in-memory database(we have considered MySQL to be the internal data store). From MySQL, we can either internally do transformations on the loaded data or implement various python scripts to do the same.

For the initial scope of this activity, it is to create the Database connectors in python. The following databases are selected for implementation:

1. MySQL
2. pgSQL
3. MongoDB
4. Elasticsearch
5. Cassandra
6. CouchDB

Developers will be installing a database in their local machine and will be creating the db connector class in python.

Architecture:



The above architecture diagram explains the ELT process i.e Extract, Load and Transform. For the Extract process to be done, we need a data source, which can be a flat file(csv, txt, pdf) or a database. This is one of the reasons why the connectors are essential. From the data source, we will be loading the data into our in-memory database (MySQL). After this part, we will be expanding the use-case on a need-only basis.

Methodology:

Stage1:

To implement the connectors, use python classes with asynchronous methods for the basic CRUD operations with the database, be it singular or batch mode. Use the below mentioned python script as abstract classes and implement the database connectors with the similar structure.

```
import mysql.connector
from engine.connectors.connector import Connector
from engine.extensions import es
# from sqlalchemy import create_engine
# from sqlalchemy import inspect
# import pymysql

class MysqlConnector(Connector):
    """Implememtation of the Connector:class."""

    def __init__(self):
        """constructor function."""
        self.connection = None


    def initialize(self,credentials):
        """Validate the credentials with the instance."""
        try:
            dbconfig = {
                'host' : credentials['host'],
                'user' : credentials['user'],
                'password' : credentials['passwd'],
                'database' : credentials['database']
            }
            self.connection = mysql.connector.connect(**dbconfig)

        except ConnectionError as e:
            return False,str(e)

    def connect(self,credentials):
        try:
            MysqlConnector.initialize(self,credentials)
```

```
            self.connection.close()
            return True,"mysql connected successfully"
        except Exception as e:
            return False,str(e)

    def list_tables(self,credentials):
        try:
            MysqlConnector.initialize(self,credentials)
            table_names = []
            cursor = self.connection.cursor()
            cursor.execute("show tables")
            tables = cursor.fetchall()
            for table in tables:
                for table_name in table:
                    table_names.append(table_name)
            self.connection.close()
            return table_names
        except Exception as e:
            print(str(e))
            return []

    def list_fields(self, credentials,table_name):
        try:
            MysqlConnector.initialize(self,credentials)
            field_details = []
            cursor = self.connection.cursor()
            query = 'show columns from '+table_name
            cursor.execute(query)
            fields = cursor.fetchall()
            for field in fields:
                field_data = {
                    'name' : field[0],
                    'type' : field[1],
                    'isNull' : field[2]
                }
                field_details.append(field_data)
            self.connection.close()
            return field_details
        except Exception as e:
            return []
```

Stage2:

Once the connector classes are completed in all aspects, load the following data into the datasource DB.

- Using connectors and python scripts, load the data to MySQL using asynchronous python frameworks for fast implementation.
- Use logging wherever possible and ensure that the code doesn't break when migrating 10gb of data.
- Keep in mind that there can pause and resume kind-of features be implemented when migrating this dataset.
- Have some output shown by the python script in command-prompt so that we can see the current status of migration.

DataSource:

https://openlibrary.org/data/ol_dump_latest.txt.gz

Resources:

Use the following gitlab repository for code management:
https://code.suyatitech.com/analytics-practice/training/training-activity-2.git