API for Database Query Execution and Data Extraction: Oracle SQL and MongoDB

Abstract

This document outlines a process of executing two Flask APIs: one for Oracle database interaction and another for MongoDB. The Oracle API enables users to execute SQL queries, fetch the results, convert them into an Excel file, and allow for download. Meanwhile, the MongoDB API retrieves specific data from MongoDB, processes it into a DataFrame, and exports it to an Excel file. Both APIs demonstrate Flask integration with database connectivity and data manipulation for Excel export.

Prepared By

Vaishnavi Kulkarni, under the guidance of Mohit Vaishnav.

Contents

Problem Statement 3
Data Needed 3
Solution 3
Team/Members Involved 6
Data Collection 6
Observations 9

Problem Statement

Given two different databases for SQL (Oracle) and NoSQL (MongoDB) , the requirement was to connect to the respective databases , perform some queries on the datasets and design a python program that can extract these databases from Oracle and MongoDB respectively and export only the resultant data obtained from executing the queries into a excel sheet, that can be used as a report for further analysis. Then develop APIs to extract data from these Oracle and MongoDB databases. The Oracle API executes custom SQL queries, converts results into Excel files, and enables download. The MongoDB API filters movie data based on criteria, processes it into Excel format, and allows exporting.

Data Needed

1. A dataset for Oracle and a dataset for MongoDB.

2. Oracle database credentials (oracle_username, oracle_password, oracle_host, oracle_port, oracle_service_name) stored in a JSON file (db_creds.json) for the Oracle API.

3. Credentials for connecting to the MongoDB database, including the database name and connection URL.

4. The SQL query to be executed on the Oracle SQL database to retrieve specific information.

5. The MongoDB aggregation pipeline query to be executed on the MongoDB database to retrieve specific information.

6. The filename and path for the Excel file where the extracted data will be saved.

7. URI for both Oracle and MongoDB databases.

Solution

The solution proposed involves two codes, each tailored to interact with a specific type of database:

1. Oracle Database API

The Oracle Database API developed using Flask facilitates data extraction and export tasks from an Oracle database. This API allows users to enter custom SQL queries for data extraction. Upon receiving a POST request, the API establishes a connection to the Oracle database using credentials stored securely in a JSON file (db_creds.json). The API then executes the SQL query, retrieves the query results, and converts them into a structured format using the pandas library. The data is subsequently exported to an Excel file.

URI used :- http://127.0.0.1:5000/api/movies

The code snippet for this API is as follows -

```
app = Flask(__name__)
@app.route('/api/download_excel', methods=['POST'])
def download_excel():
    data = request.get_json()
    sql_query = data.get('sql_query')
    excel_filename = data.get('excel_filename')
    excel_file = fetch_from_oracle(sql_query, excel_filename)
    message = f"File '{excel_filename}' has been initiated for download."
    response_data = {
        "message": message,
        "file_url": f"/downloads/{excel_filename}"
    }
    response = make_response(json.dumps(response_data))
    response.headers['Content-Type'] = 'application/json'
    return response

def fetch_from_oracle(sql_query, excel_filename):

    with open('db_creds.json') as f:
        db_creds = json.load(f)
    oracle_username = db_creds["oracle_username"]
    oracle_password = db_creds["oracle_password"]
    oracle_host = db_creds["oracle_host"]
    oracle_port = db_creds["oracle_port"]
```

```
    oracle_service_name = db_creds["oracle_service_name"]
    connection = cx_Oracle.connect(
        user=oracle_username,
        password=oracle_password,
        dsn=f"{oracle_host}:{oracle_port}/{oracle_service_name}"
    )
    cursor = connection.cursor()
    cursor.execute(sql_query)
    columns = [col[0] for col in cursor.description]
    result = cursor.fetchall()
    df = pd.DataFrame(result, columns=columns)
    cursor.close()
    connection.close()
    df.to_excel(excel_filename, index=False)
    return excel_filename
```

2. MongoDB Database API:

The MongoDB API implemented with Flask is designed to retrieve and process data from a MongoDB database. This API applies predefined aggregation pipeline conditions to filter data based on certain criteria. Upon successful data retrieval, the API transforms the MongoDB query results into a pandas DataFrame for structured data representation. Then the processed data is exported to an Excel file.

URI used :- http://localhost:8080/api/download_excel

The code snippet for this API is as follows -

```
app = Flask(__name__)
MONGO_URI = "mongodb://localhost:27017/"
DATABASE_NAME = "MOVIE"
client = MongoClient(MONGO_URI)
db = client[DATABASE_NAME]

@app.route('/api/movies', methods=['GET'])
def get_movies():
```

```
    pipeline = [
        {"$match": {"year": {"$gte": 2000}, "imdb.rating": {"$gte": 6.0}}},
        {"$project": {"_id": 1, "plot": 1, "cast": 1, "directors": 1, "awards": 1, "imdb": 1}}
    ]
    result = db.movies.aggregate(pipeline)
    data = list(result)
    df = pd.DataFrame(data)
    df['wins'] = df['awards'].apply(lambda x: x.get('wins', None))
    df['nominations'] = df['awards'].apply(lambda x: x.get('nominations', None))
    df['imdb rating'] = df['imdb'].apply(lambda x: x.get('rating', None))
    df['cast'] = df['cast'].apply(lambda x: ', '.join(x) if isinstance(x, list) else x)
    df['directors'] = df['directors'].apply(lambda x: ', '.join(x) if isinstance(x, list) else x)
    df.drop(['awards', 'imdb'], axis=1, inplace=True)
    output_file = "data_movie3.xlsx"
    df.to_excel(output_file, index=False)

    return jsonify({"message": "Data exported to Excel successfully!"})
```

Team/Members Involved

K V Gopalakrishna

Deepak K Dubey

Mohit Vaishnav

Vaishnavi Kulkarni

Data Collection

1. The sample database used in Oracle is a ‘Sales_History’ database.

2. The sample database used in MongoDB is a ‘MOVIE’ database.

Sample input SQL query for Oracle database

```
SELECT C.CUST_ID, C.CUST_FIRST_NAME, C.CUST_LAST_NAME, SUM(S.AMOUNT_SOLD) AS
TOTAL_SALES
  FROM CUSTOMER C
  JOIN SALES S ON C.CUST_ID = S.CUST_ID
  GROUP BY C.CUST_ID, C.CUST_FIRST_NAME, C.CUST_LAST_NAME
  ORDER BY TOTAL_SALES DESC
  FETCH FIRST 10 ROWS ONLY
```

Sample output

CUST_ID	CUST_FIRST_NAME	CUST_LAST_NAME	TOTAL_SALES
11407	Dora	Rice	103412.66
10747	Lolita	Katz	99578.09
42167	Tesia	Eppling	98585.96
4974	Xerxes	Abbassi	98006.16
12783	Rose	Lehman	97573.55
6395	Mitchel	Polk	97010.48
2994	Ronald	Adams	94862.61
429	Meriel	Fairman	94819.41
1743	Harold	Allis	94786.13
4759	Roderica	Lavin	93644.32

Sample input MongoDB query for MongoDB database

```
[
    {
      "$match": {
        "year": { "$gte": 2000 },
        "imdb.rating": { "$gte": 6.0 }
      }
    },
    {
      "$project": {
        "_id": 1,
        "plot": 1,
        "cast": 1,
        "directors": 1,
        "awards": 1,
        "imdb": 1
      }
    }
  ]
```

Sample output

_id	plot	cast	directors	wins	nominations	imdb
573a	Kate and …	Meg Ryan, Hugh Jackman	James Mangold	2	4	6.3
573b	A modern day..	Crispin Glover, Vanessa	Menahem Golan	2	0	6.4

Observations

- The use of a JSON file to store database credentials ensures secure connectivity to Oracle SQL and MongoDB databases.
- Credentials are programmatically accessed, reducing the risk of exposure and enhancing security.
- The use of Flask endpoints enables users to initiate and track data export tasks easily.