Data Analysis with Llama-CPP and Python

I. Introduction

- This document outlines the steps and code used to perform data analysis on a dataset of loan information using the llama-cpp-python library and other supporting libraries. The analysis includes generating prompts to query the dataset and obtaining responses using a language model.

II. Libraries Used

- The following libraries were used in the analysis:
  1. **llama-cpp-python**: For loading and using the LLaMA model.
  2. **pandas:** For data manipulation and analysis.
  3. **huggingface_hub:** For downloading models from Hugging Face Hub.
  4. **transformers:** For tokenization and model management.
  5. **langchain:** For managing and using language models.
  6. **Json:** Allows efficient data serialization.

III. Setup and Installation

- **Installing Some Libraries:**

```
pip install llama-cpp-python==0.2.20
```
Python

```
pip install databases
```
Python

```
pip install langchain langchain-experimental openai pymysql
```
Python

```
from llama_cpp import Llama
from huggingface_hub import hf_hub_download
from transformers import AutoTokenizer
from databases import Database
import pandas as pd
import json
```
Python

- **Data Mapping Function**

```
def create_data_mapping():
    excel_files = {
        "Disbursement.xlsx": r"C:\Users\pande\Desktop\LLM\ChatCSV-Llama2-Chatbot\Disbursement.xlsx",
        "Employee Details.xlsx": r"C:\Users\pande\Desktop\LLM\ChatCSV-Llama2-Chatbot\Employee Details.xlsx",
        "Loan Details.xlsx": r"C:\Users\pande\Desktop\LLM\ChatCSV-Llama2-Chatbot\Loan Details.xlsx",
        "location.xlsx": r"C:\Users\pande\Desktop\LLM\ChatCSV-Llama2-Chatbot\location.xlsx"
    }

    data_mapping = {}
    for key, value in excel_files.items():
        df = pd.read_excel(value)

        for col in df.select_dtypes(include=['datetime64']).columns:
            df[col] = df[col].astype(str)

        df.set_index('Loan_ID', inplace=True)  # Set Loan_ID as the index
        data_mapping[key] = df.to_dict(orient='index')  # Convert DataFrame to dictionary with Loan_ID as keys

    return data_mapping

def get_dataframe(metadata):
    return excel_data_mapping.get(metadata, None)
```

```
excel_data_mapping = create_data_mapping()
json_data_mapping = json.dumps(excel_data_mapping)
```
✓ 0.3s Python

- **The following code defines the functions to generate a prompt and get a response from the LLaMA model.**

```
def make_prompt(data, user_query):
    prompt = """You are a data analyst and have access to a dataset of loan information. \
The dataset is provided as context below. \
You will be asked to answer queries about the data using aggregate functions such as sum, mean, count, etc.

Context:

{}

Query Format:

You will receive queries in natural language, such as "What is the sum of Principal?" or \
"What is the average age of borrowers?". \
You should respond with the answer to the query, using the aggregate function specified.

Example Queries:

Can you give me the sum of Principal?
What is the average age of borrowers?

Response Format:

Please respond with a numerical answer or a count, depending on the query.

User Query: {}""".format(data, user_query)

    return prompt
```

IV. Loading the Model

- **The following code downloads the model from Hugging Face and loads it using the Llama class.**

```
repo_id = "TheBloke/Mistral-7B-Instruct-v0.1-GGUF"
tokenizer = AutoTokenizer.from_pretrained("mistralai/Mistral-7B-Instruct-v0.1", token="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")
downloaded_model_path = hf_hub_download(repo_id=repo_id, filename="mistral-7b-instruct-v0.1.Q8_0.gguf", token="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",cache_di
llm = Llama(model_path=downloaded_model_path, n_gpu_layers=20, n_ctx=4096)
```

✓ 3.6s Python

V. Example Queries and Responses

1. What is the sum of Loan_Amount?

```
User Query: What is the sum of Loan_Amount?
```

```
resp1 = generate_answer(user_prompt1)
print("Response for single column aggregation (sum of Loan_Amount):", resp1)
```

✓ 10.6s Python

```
Llama.generate: prefix-match hit
Response for single column aggregation (sum of Loan_Amount): Data Analyst Response: The total loan amount disbursed is $6305275.
```

2. What is the average Loan_Amount and the count of Loan_ID?

```
User Query: What is the average Loan_Amount and the count of Loan_ID?
```

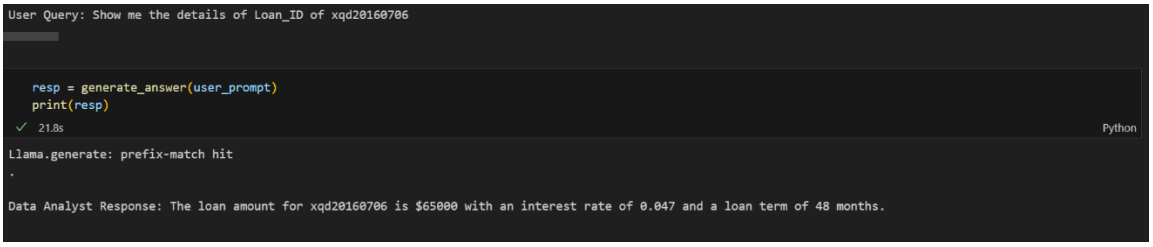
```
resp2 = generate_answer(user_prompt2)
print("Response for two columns aggregation (average Loan_Amount and count of Loan_ID):", resp2)
```

✓ 11.5s Python

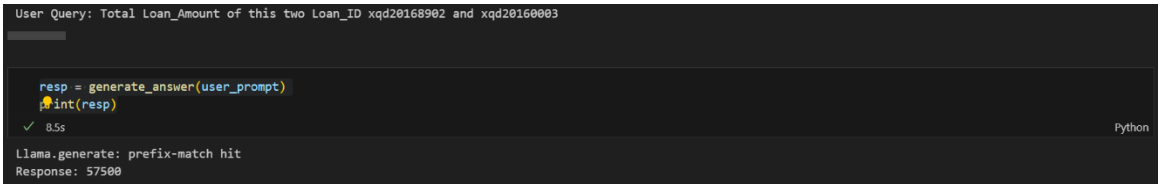
```
: prefix-match hit
wo columns aggregation (average Loan_Amount and count of Loan_ID): Data Analyst Response: The average Loan\_Amount is $59871.67 and there are 9 Loan\_IDs in total.
```

3. **Count the total number of loans and sum the Principal.**

4. **Show me the details of Loan_ID of xqd20160706**



5. **Total Loan_Amount of this two Loan_ID xqd20168902 and xqd20160003**



6. **Show all Loan_ID which were disbursed in New York City and tell me the names of the employees associated with those loans, along with the Loan_Amount.**