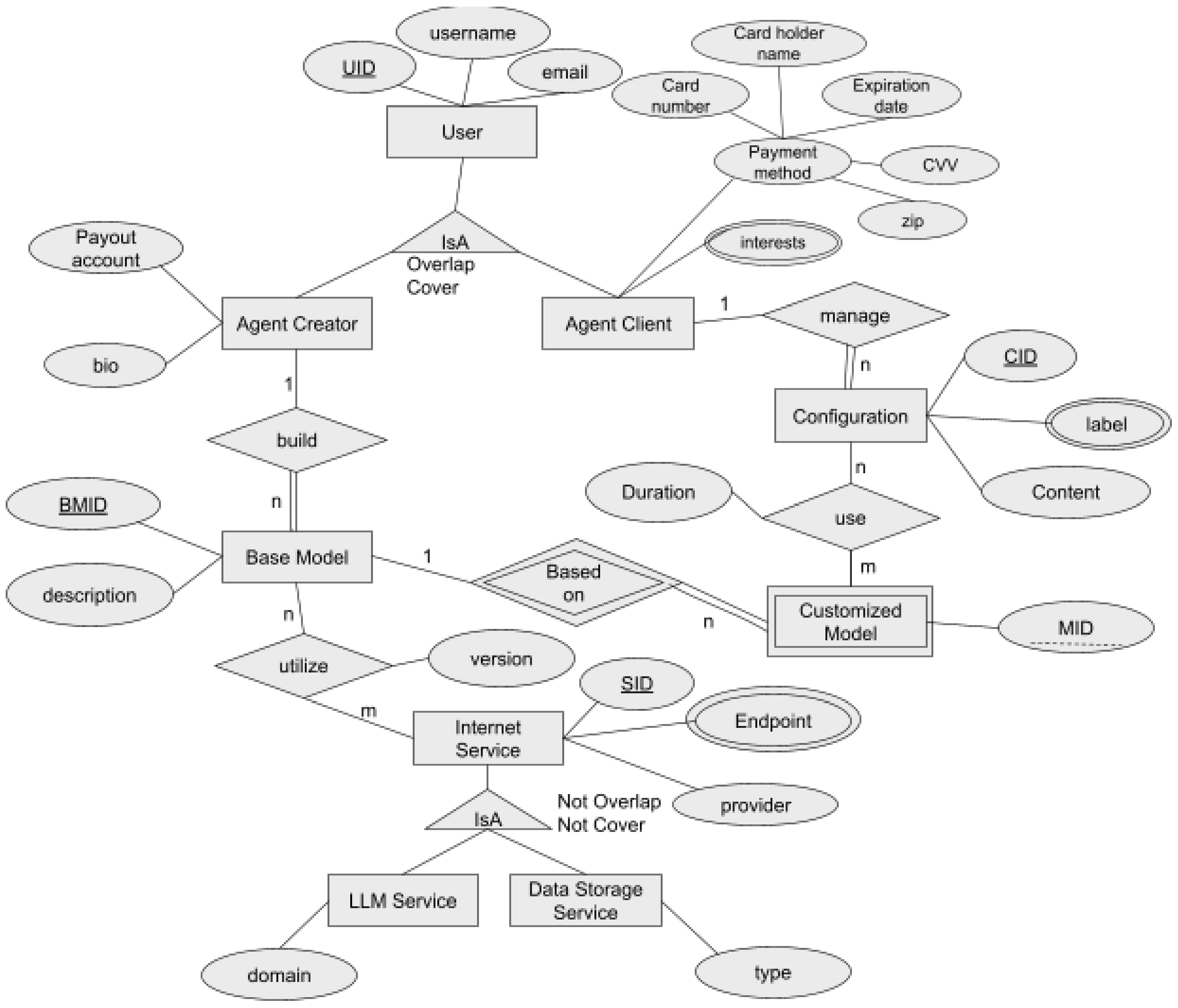
CS 122A Project

Introduction

In this project, you will implement a **command-line program** to manage the Agent platform. You are required to use the Python MySQL connector to access and manipulate the database.

The Python program should
1) accept command-line arguments as the inputs,
2) parse the inputs into SQL statements,
3) execute the statements in the MySQL server,
4) handle and print the results.

ER diagram (please refer to the HW2 solution for the DDLs) :



Dataset

There will be one CSV file for each table. Each line represents one record, and the columns are separated by a comma ‘,’. The order of columns follows the order of attributes in the DDL.

The **provided dataset** is for testing only, so you can make any changes to cover more cases. A hidden dataset (for evaluating Q1-Q8 only) will be used during the grading. ~~In the zip file is an~~ ~~**instructions text file** which contains commands to run in MySQL in order to load CSV files into your database **after having run the DDL statements.**~~

Q9 is open question, so no hidden dataset will be used to test the answer. Check the instructions for Q9 in the last two pages.

Regulations and Assumptions

1. You can have any number of Python files, but the entry/main file must be named **“project.py”**
2. The command to run the program will be

   python3 project.py <function name> [param1] [param2] …

   The list of function names and their parameters is in the function requirements section.
3. You can assume that the command-line input is always correct IN FORMAT ONLY. There won’t be a nonexistent function name as input, and the parameters will be given in the correct order and format. So you don’t need to handle unexpected input. However, input content can be faulty - e.g., given a duplicate netID for insertion.
4. You can assume that the dataset files are always correct IN FORMAT AND CONTENT. So there won’t be errors when parsing the file, or when inserting the records to DB.
5. Every date has the format YYYY-MM-DD, e.g. 2025-02-29, and every datetime has the format YYYY-MM-DD hh:mm:ss, e.g. 2025-02-29 14:10:34.
6. Strings that contain spaces will be wrapped in quotation marks when calling the command, (e.g. "The Matrix") whereas strings with no spaces will not have quotation marks (e.g. Wicked).
7. If the input is NULL, treat it as the None type in Python, not a string called “NULL”.
8. **If the output is boolean**, print “Success” or “Fail”.
9. **If the output is a result table**, print each record in one line and separate columns with ‘,’ - just like the format of the dataset file.
10. You must use **Python 3**. The standard Python libraries and mysql-connector-python will be installed in the autograder — other third-party packages are not allowed.

Setup Instructions

- Install MySQL server and test if you can run queries. (instructions)
- Install mysql-connector-python and try running SQL queries from python. (guide)

Function Requirements

1. **Import data**
   Delete existing tables and create new tables. Then read the CSV files in the given folder and import data into the database. You can assume that the folder always contains all the necessary CSV files and that the files are correct.

Input:

```
python3 project.py import [folderName:str]

EXAMPLE: python3 project.py import test_data
```

Output:

Boolean

2. **Insert Agent Client**
   Insert a new agent client into the related tables.

Input:

```
python3 project.py insertAgentClient [uid:int] [username:str] [email:str] [card_number:int]
[card_holder:str] [expiration_date:date] [cvv:int] [zip:int] [interests:str]

EXAMPLE: python3 project.py insertAgentClient 1 “awong” “test@uci.edu” 12345 “Alice
Wong” “2020-03-09” 321 92612 “finance;data analysis”
```

Output:

Boolean

3. **Add a customized model**
   Add a new customized model to the tables.

Input:

```
python3 project.py addCustomizedModel [mid:int] [bmid:int]

EXAMPLE: python3 project.py addCustomizedModel 130 30
```

Output:

Boolean

4. **Delete a base model**
   Delete a base model from the tables.

Input:

```
python3 project.py deleteBaseModel [bmid:int]

EXAMPLE: python3 project.py deleteBaseModel 30
```

Output:

Boolean

5. **List internet service**

Given a base model id, list all the internet services that the model is utilizing, sorted by provider's name in **ascending** order.

Input:

python3 project.py listInternetService [bmid:int]

EXAMPLE: python3 project.py listInternetService 30

Output:

Table: sid, endpoint, provider

6. **Count customized model**

Given a list of base model id, for each base model id, count on the numbers of customized models that build from it. Sort the results in **ascending** order of base model id.

Input:

~~python3 project.py countCustomizedModel [bmid:int]~~

~~EXAMPLE: python3 project.py countCustomizedModel 30~~

~~python3 project.py countCustomizedModel list: [bmid1:int, bmid2:int, bmid3:int]~~

python3 project.py countCustomizedModel [bmid1:int] [bmid2:int] [bmid3:int]

([bmid1:int] [bmid2:int] [bmid3:int] should be taken as list type in your function)

EXAMPLE: python3 project.py countCustomizedModel 40 30 50

Output:

Table: bmid, description, customizedModelCount

7. **Find Top-N longest duration configuration**

Given an agent client id, list the top-N longest duration configurations with the longest duration managed by that client. Sort the configurations by duration in descending order.

Input:

python3 project.py topNDurationConfig [uid:int] [N:int]

EXAMPLE: python3 project.py topNDurationConfig 1 5

Output:

Table: uid, cid, label, content, duration

8. **Keyword search**

List 5 base models that are utilizing LLM services whose domain contains the keyword "video". (If there are fewer than 5 base models that satisfy the condition, list them all.) Sort the results by bmid in ascending order.

Input:

```
python3 project.py listBaseModelKeyWord [keyword:str]

EXAMPLE: python3 project.py listBaseModelKeyWord “video”
```

Output:

Table: bmid,sid, provider, domain

9. Experiment: Solving NL2SQL with LLM

NL2SQL, or text-to-SQL, is a task that translates natural language queries into SQL queries. NL2SQL is an interdisciplinary study between NLP (natural language processing) and database systems. In this project, the previously required functions also fall under the NL2SQL category — students play the role of converting natural language (NL) into SQL queries.

LLM is now widely used for NL2SQL applications, due to its strong ability to understand natural language and perform domain-specific tasks such as code generation.

Your goal in the next task is to explore how effectively an LLM can perform NL2SQL conversions.

NL2SQL Task Description

A single natural language question can often be expressed by multiple valid SQL queries. For each provided question:

1) Using LLM to **generate 2-5 different SQL queries**.
    - You may choose to use an online free version of any LLM (e.g., ChatGPT, Gemini, Claude, etc.), or access through an API key (which comes with a cost).
    - Experiment with **different prompt strategies**, deciding what information should or should not be included in the prompt to improve correctness (note that we do not expect that all SQL queries that are generated are going to be fully correct or different. Part of this project is to analyze the outcomes as we detail in the rest o this question).

      Prompt Construction Hint:
        a. Basic prompt could be: “Translate this natural language query into SQL format: [NL query] ";
        b. A more sophisticated prompt could include a related table schema: “Achieve this NL2SQL task from the given natural language query and table schema: [NL query] [table schema]”
        c. Tuning prompt with different job descriptions, context (e.g., table schema), or even data samples (e.g., few-shot learning strategies).
2) Execute each generated SQL query and record your results
    - For each question, write down the **prompt** that you used and the generated SQL query. (if the generated SQL is the same for different prompts, it’s ok.)
    - Execute the SQL query in the database. We provide the ground-truth answers for comparison.
    - If the query result matches the ground truth, mark it as ‘correct’. If not, analyze the source of error, e.g., incorrect column name, syntax error, incorrect table name, etc. (you can include as many errors as you identified)

3) Save your results into a CSV file, and write a Python function to read and print its contents.
   Note that the following table format is not fixed. You must include the first seven columns, while the remaining columns for the error types can be adjusted based on specific errors you identify. If every prompt you used with the LLM generated correct SQL queries, you can exclude the error-type columns from your table—one function (table) for printing all three NL questions.

NL2SQL questions:

Q1. For each Internet Service(sid), find the most frequently used version among all Base Models that utilize it. If multiple versions have the same highest frequency, you may return any one of them.
Q2. Find all the Internet Service(sid) that are utilized by every Base Model built by the agent creator **user_iuwrh**.
Q3. Find all customized models(mid) that do not appear in model configurations.

Input: python3 project.py printNL2SQLresult
Output:
Print a table:

```
{
NLquery_id int,
NLquery str,
LLM_model_name str,
prompt str,
LLM_returned_SQL_id int,
LLM_returned_SQL_query str,
SQL_correct bool,
Error_name1 bool,
Error_name2 bool,
Error_name3 bool,
…}
```

Ground Truth for NL2SQL questions:

Ans of Q1: (result with frequency:) ; without frequency(last column) can also be counted as ‘correct’

```
+-----+---------+-------------+
| sid | version | usage_count |
+-----+---------+-------------+
|   1 |       2 |           3 |
|   2 |       5 |           2 |
|   3 |       2 |           2 |
|   4 |       5 |           2 |
|   5 |       4 |           2 |
|   6 |       3 |           2 |
|   7 |       1 |           2 |
|   8 |       1 |           2 |
|   9 |       4 |           3 |
|  10 |       2 |           2 |
|  11 |       3 |           2 |
|  12 |       5 |           1 |
+-----+---------+-------------+
```

Ans of Q2:

```
+-----+
| sid |
+-----+
|   8 |
|   3 |
|   7 |
|   9 |
+-----+
```

Ans Q3:

```
+-----+
| mid |
+-----+
|   2 |
|   3 |
|   4 |
|   7 |
|   8 |
|   9 |
|  11 |
|  12 |
|  14 |
|  17 |
|  23 |
|  24 |
|  26 |
|  31 |
|  35 |
|  39 |
+-----+
```