databaseUpload.py

1.0 Introduction

databaseUpload.py is the final component of the backend and is meant to be the bridge between the data formatter and the database where the formatted data is validated and is packaged into SQL statements where it is then uploaded to the database.

2.0 Architectural and component-level design

1. db_uploadFunction(dbup_table)
    a. db_uploadFunction takes in a dictionary of formatted data and constructs SQL statements from the given data, then executes all SQL statements.
    b. Accepts a dictionary of the following format:
       Note: See the section on data_validation() for more information on the attributes
       Note: yoe = years of experience
        jobID: string
        table: string
        degreeTitle: list of strings
        industry: list of strings
        educationLevel: list of characters
        languages: list of strings
        seniority: string
        skills:list of strings
        yoe: integer
    c. Uses the class DBConnection to handle the connection to the database and to execute SQL statements.
    d. All SQL statements are appended one after the other and executed in sequence at the end.
        i. This is so that if there are errors during execution then the process can be halted before anything is uploaded to the database.
    e. First constructs a SQL statement insert into the jobIDTable.
        i. An insertion must be made into the jobIDTable first because all other tables reference a jobID in the jobIDTable

   f. Next a SQL statement is made to make an insertion into the appropriate job table.
      i. If either seniority or years of experience have been found to be invalid then an error message is printed and execution is halted.
   g. After that insertions are made into the attribute tables.
      i. The attribute tables are as follows:
         1. degrees
         2. education
         3. industries
         4. languages
         5. skills
      ii. If the attribute for one of those tables is found to be invalid then execution is not halted.
         1. Instead the insertion into the infringing attributes' table is skipped and an error message is displayed to signify this.
   h. Finally all SQL statements are executed in sequential order.

2. data_validation(table)
   a. Validates the given data before constructing the SQL statements in order to check for invalid and NULL values.
   b. Returns a list of values signifying the status of each attribute given in the dictionary.
      i. Validation Results Format:
         1. Values:
            a. None = NULL
            b. 0 = No error
            c. -1 = Error
         2. Attributes:

            Note: Global variables are defined to easily access the validation status of each attribute.

            a. result[0] = seniority
            b. result[1] = industry
            c. result[2] = degreeTitle
            d. result[3] = educationLevel
            e. result[4] = skills

        f. result[5] = languages
        g. result[6] = yoe
    c. Methods of Validation:
        i. seniority
            1. If the value is -1 then this signifies that the value should be interpreted as a NULL.
            2. If the value is not null then it must be one of six values:
                a. internship
                b. entry level
                c. associate
                d. mid-senior level
                e. director
                f. executive
            3. If the value does not match one of those six values then it will be marked as invalid.
        ii. industry
            1. If the given list is empty then it will be interpreted as NULL.
        iii. degreeTitle
            1. If the given list is empty then it will be interpreted as NULL.
        iv. educationLevel
            1. If the given list is empty then it will be interpreted as NULL.
            2. If the value is not null then it must be one of 4 character values:
                a. ‘a’ (associate’s degree)
                b. ‘b’ (bachelor's degree)
                c. ‘m’ (master’s degree)
                d. ‘p’ (PhD)
            3. If the value does not match one of those six values then it will be marked as invalid.
        v. skills
            1. If the given list is empty then it will be interpreted as NULL.
            2. Skills are validated in the data formatter from a list of predetermined values.
        vi. languages

            1. If the given list is empty then it will be interpreted as NULL.
            2. Languages are validated in the data formatter from a list of predetermined values.

    vii. yoe

            1. If the value is -1 then this signifies that the value should be interpreted as a NULL.
            2. If the years of experience of value is something unrealistic or impossible like -2 years of experience or 30 years of experience then it will be flagged as invalid.

3. table_validation(db, table_name)
    a. Queries the database to get a list of job tables in the database.
    b. If the given table is not in the database then a new table will be created.
    c. This allows our backend to be scalable as all we need to do to start collecting data on a new job is to add a new search term to the web scraper.

3.0 Testing

1. Primary Testing Method: Whole-System Testing
    a. The reason whole-system testing was used is because the functionality of databaseUpload.py depends on data given by the data formatter which depends on data given by the web scraper.

4.0 Known Bugs

1. It is possible for a situation to arise where all values given are null and basically a useless insertion is made into the database.