Design Document:

Problem Statements:

1. Data Migration:

Write a script for migration of data from Excel File to Database such that all the sheets of that Excel file is stored in the PostgreSQL database as table schema and excel records is inserted to that table.

Tools & Technologies Used:

- Node.js for writing script for migrating data from excel sheets.
- NPM (Node Package Manager) used to acquire required modules for implementation i.e. ‘xlsx’, ‘pg’ and ‘prompt-sync’.
- PostgreSQL Database for storing the migrated data.
- Git/GitHub for version control.
- VS Code Editor

Proposed Solution:

Step 1: Import required modules:

- Import the xlsx module for reading Excel files.
- Import the pg module for connecting to PostgreSQL.
- Import the fs module for file system operations.
- Import the prompt-sync module for user input.

Step 2: Define PostgreSQL server information:

- Create a Pool object with the necessary properties for connecting to the PostgreSQL server.

Step 3: Define the main() function:

- This is the main function that orchestrates the overall process.
- It calls the connectToPostgreSQL() function to establish a connection to the PostgreSQL database.
- It calls the readExcel() function to read the Excel file, create tables, and insert data.
- It calls the disconnectFromPostgreSQL() function to disconnect from the PostgreSQL database.
- Step 4: Call the main() function to start the process.

Step 5: Connect to the PostgreSQL database:

- Use the pool.connect() function to establish a connection to the PostgreSQL database.
- If the connection is successful, log a success message.
- If there is an error, log the error message.

Step 6: Read data from an Excel file:

- Use the xlsx.readFile() function to read the Excel file specified by the file path.
- Retrieve the names of all the sheets in the Excel file.
- Iterate through each sheet:
    - Get the range of cells in the sheet using xlsx.utils.decode_range().

    - Iterate through each column in the range:
        - Retrieve the value of the cell and convert it to a column name.
        - Convert the column name to lowercase and replace spaces with underscores.
        - Store the column names in an array for each sheet.
        - Log the table name and column names for each sheet.

    - Store the array of column names for each sheet in another array.
- Log the array containing column names for all sheets in the Excel file.

Step 7: Ask the user for data types and constraints:

- Create an empty array to store the data types and constraints for each column in each table.
- Iterate through each table:
    - Get the table name and column names.
    - Ask the user to input the data type and constraints for each column.
    - Store the input in an array and push it into the main array.
    - Log the array containing the data types and constraints for each column in the table.

Step 8: Create table schema in the PostgreSQL database:

- Iterate through each table:
    - Get the table name and column names with their corresponding data types.
    - Use the createTable() function to execute a SQL query for creating the table with the specified columns and data types.

Step 9: Insert data into the tables:

- Iterate through each table:
    - Get the table name, column names, and corresponding worksheet data.
    - Use the sheet_to_json() function to convert the worksheet data into an array of objects representing rows.
    - Remove the first element of the array containing column names.
    - Use the insertData() function to insert the data into the PostgreSQL database.
    - Log the number of records inserted.

Step 10: Define the createTable() function:

- This function creates a table in the PostgreSQL database if it doesn't already exist.
- It takes the table name and column names with their data types as parameters.
- It uses the pool.connect() function to establish a connection.
- It executes a SQL query to create the table with the specified columns and data types.
- If there is an error, it logs the error message.

Step 11: Define the insertData() function:

- This function inserts data into a specified table in the PostgreSQL database.
- It takes the table name, column names, and the records (an array of arrays representing rows) as parameters.
- It uses the pool.connect() function to establish a connection.
- It maps and sanitizes the records to handle empty cell values.
- It executes a SQL query to insert the data into the table.
- If there is an error, it logs the error message.

Step 12: Disconnect from the PostgreSQL database:

- Use the pool.end() function to disconnect from the PostgreSQL database.
- If the disconnection is successful, log a success message.
- If there is an error, log the error message.