

Mini project report on

ELECTRICITY MANAGEMENT SYSTEM

*Submitted in partial fulfilment of the requirements for the award of degree of*

**Bachelor of Technology**

**in**

**Computer Science & Engineering**

**UE21CS351A – DBMS Project**

***Submitted by:***

|  |  |
| --- | --- |
| **Melvin Jojee Joseph**  **L Sai Tejas** | **PES2UG21CS294**  **PES2UG21CS250** |

Under the guidance of

**Prof. Mannan J Mannan**

Associate Professor,

|  |
| --- |
| CSE  PES University |

AUG - DEC 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India



PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

**CERTIFICATE**

*This is to certify that the mini project entitled*

**Electricity Management System**

*is a bonafide work carried out by*

|  |  |
| --- | --- |
| **Melvin Jojee Joseph**  **L Sai Tejas** | **PES2UG21CS294**  **PES2UG21CS250** |

In partial fulfilment for the completion of fifth semester DBMS Project (UE20CSS301) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2022 – DEC. 2023. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

|  |  |
| --- | --- |
| Signature  Prof. Mannan J Mannar  Assistant Professor |  |

**DECLARATION**

We hereby declare that the DBMS Project entitled **Electricity management system** has been carried out by us under the guidance of **Prof. Mannan J Mannar , Assistant Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2023.

|  |  |  |
| --- | --- | --- |
| **Melvin Jojee Joseph**  **L Sai Tejas** | **PES2UG21CS294**  **PES2UG21CS250** |  |

**ACKNOWLEDGEMENT**

I would like to express my gratitude to Prof. Mannan J Mannar, Department of Computer Science and Engineering, PES University, for his continuous guidance, assistance, and encouragement throughout the development of this UE21CS351 - DBMS Project.

I take this opportunity to thank Dr. Sandesh B J, C, Professor, Chair Person, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr.Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this DBMS Project could not have been completed without the continual support and encouragement I have received from my family and friends.

**ABSTRACT**

The constant evolution of technology and the increasing demand for energy necessitate efficient and intelligent systems for managing electricity consumption. This abstract introduces an innovative Electricity Management System (EMS) designed for seamless integration with Database Management Systems (DBMS). The proposed system aims to provide a user-friendly platform for managing and tracking electricity consumption, billing, and payment processes efficiently. This system aims to streamline the billing process enhance user experience by providing seamless access to the users to their monthly consumption.

# 

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page No.** |
|  | **INTRODUCTION** | **9** |
|  | **PROBLEM DEFINITION** | **11** |
|  | **ER MODEL** | **12** |
|  | **ER TO RELATIONAL MAPPING** | **13** |
|  | **DDL STATEMENTS** | **15** |
|  | **DML STATEMENTS** | **19** |
|  | **QUERIES (SIMPLE QUERY AND UPDATE AND DELETE OPERATION, CORRELATED QUERY AND NESTED QUERY)** | **22** |
|  | **STORED PROCEDURE, FUNCTIONS AND TRIGGERS** | **25** |
|  | **FRONT END DEVELOPMENT** | **27** |
| **REFERENCES/BIBLIOGRAPHY** | | **34** |
|  | |  |
|  | |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| **1** | **ER Model** | **12** |
| **2** | **Complete Diagram of relational mapping** | **14** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# INTRODUCTION

In the dynamic landscape of modern energy consumption, efficient and organized management of electricity is paramount. To address the complexities and demands of electricity distribution and billing, the Electricity Management System (EMS) emerges as a robust solution. This system leverages the power of a well-designed Database Management System (DBMS) to streamline operations, enhance data accuracy, and improve overall efficiency.

The key entities within the Electricity Management System include customers, administrators, bills, tariff structures and the electricity board. Each entity plays a crucial role in maintaining a seamless flow of information and services throughout the electricity distribution network.

1. **Customer:**
   * Represents the end-users who consume electricity.
   * Captures and manages customer details such as contact information, meter readings, and consumption history.
2. **Admin:**
   * Empowers system administrators to oversee and control the entire EMS.
   * Manages user accounts, system configurations, and ensures the security and integrity of the database.
3. **Bills:**
   * Encompasses the billing information associated with each customer.
   * Stores details like billing period, meter readings, and calculates the total consumption cost.
4. **Tariff:**
   * Defines the pricing structures for electricity consumption.
   * Encompasses different rates based on factors such as usage patterns, time of day, and any applicable discounts.
5. **Electricity Board:**
   * Represents the governing body responsible for the overall management and regulation of electricity distribution.
   * Monitors system performance, implements policies, and ensures compliance with regulatory standards.

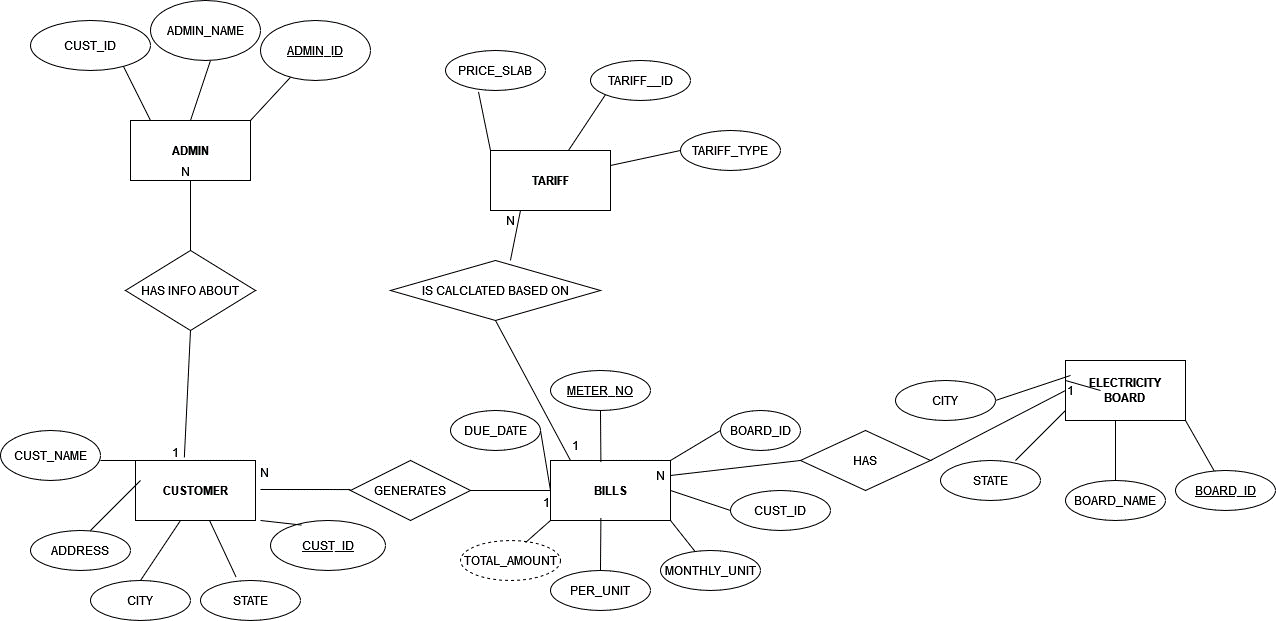
The integration of a DBMS into the Electricity Management System not only enhances data organization but also facilitates real-time data retrieval and analysis. This system provides a user-friendly interface for both administrators and customers, allowing for seamless interaction and efficient decision-making.

By centralizing and optimizing the management of entities like customers, admins, bills, tariffs, and the electricity board, the Electricity Management System becomes a pivotal tool in fostering a reliable, transparent, and economically viable electricity distribution network. This introduction sets the stage for exploring the various functionalities and benefits offered by the EMS powered by a robust DBMS.

# PROBLEM DEFINITION:

The Electricity Management System using a Database Management System aims to address these challenges by providing a structured and efficient platform. It centralizes data, streamlines operations, enhances user interaction, ensures accurate billing, facilitates financial transaction management, and promotes regulatory compliance. By identifying and articulating these challenges, the problem definition sets the stage for the development and implementation of a comprehensive solution to improve the overall electricity management process.

# ER MODEL:

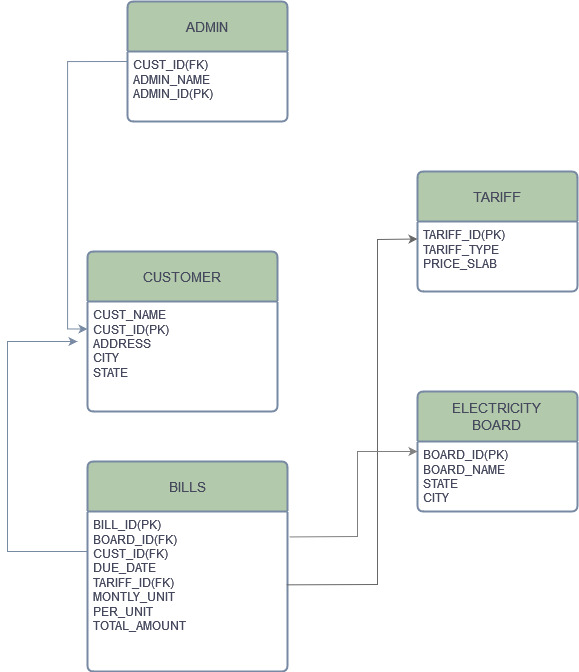


# ER TO RELATIONAL MAPPING:

**4.1 STEPS OF ALGORITHM FOR CHOOSEN PROBLEM :**

1. **Identify Entities:**
   * Entities in the ER diagram are translated into tables in the relational model.
   * In the provided script, tables such as customer, admin, electricity\_board, tariff, and bill represent the entities.
2. **Define Attributes:**
   * Attributes of entities become columns in the corresponding tables.
   * For example, in the customer table, attributes like cust\_id, name, address, city, and state are columns.
3. **Identify Primary Keys:**
   * Primary keys in the ER diagram become primary key constraints in the relational model.
   * In the script, the PRIMARY KEY constraints are applied to columns like cust\_id, admin\_id, board\_id, tariff\_id, and bill\_id.
4. **Handle Relationships:**
   * Relationships between entities are represented using foreign keys.
   * For instance, in the admin table, cust\_id is a foreign key referencing the customer table's cust\_id.
   * In the bill table, board\_id and cust\_id are foreign keys referencing the electricity\_board and customer tables, respectively.
5. **Handle Cardinality:**
   * If there are one-to-one, one-to-many, or many-to-many relationships, ensure that foreign keys are appropriately placed to maintain referential integrity.
6. **Translate Weak Entities:**
   * If there are weak entities, they might not have a primary key of their own. In such cases, a composite primary key involving the owner entity's primary key may be used.
7. **Create Indexes:**
   * Depending on the requirements and query patterns, you may need to create indexes on certain columns for better performance.
8. **Data Types and Constraints:**
   * Choose appropriate data types for each column (e.g., INT, VARCHAR(255), DATE) based on the nature of the data.
   * Apply constraints such as NOT NULL where necessary.

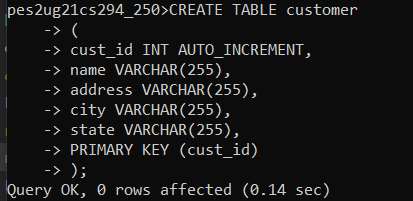
**4.2 COMPLETE DIAGRAM OF RELATIONAL MAPPING:**



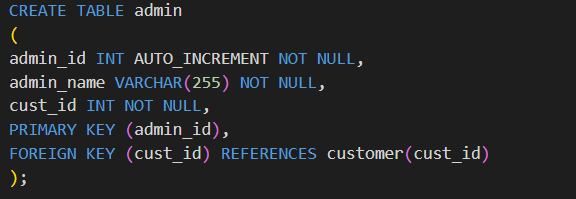
# DDL STATEMENTS:

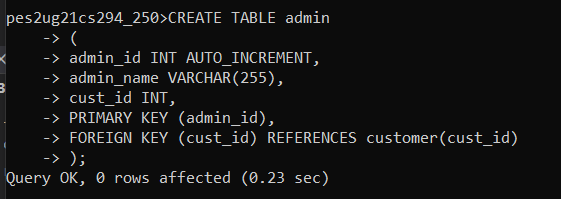
**Creation of table “customer”:**

# 

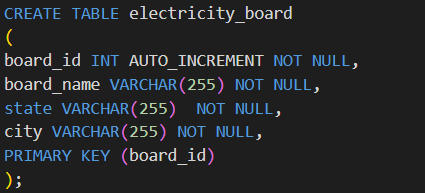


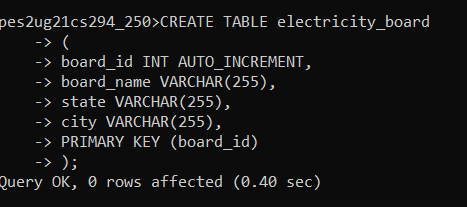
**Creation of table “admin”:**

****

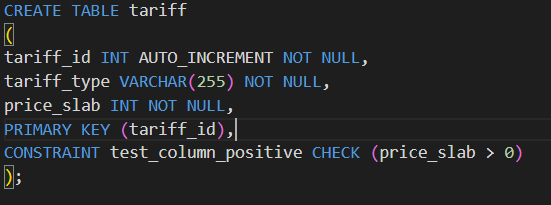
****

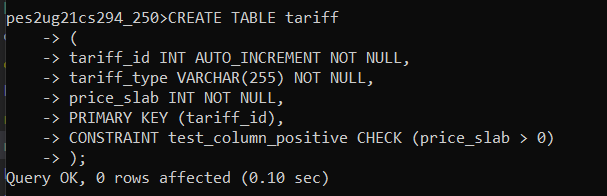
**Creation of table “electricity\_board”:**

****

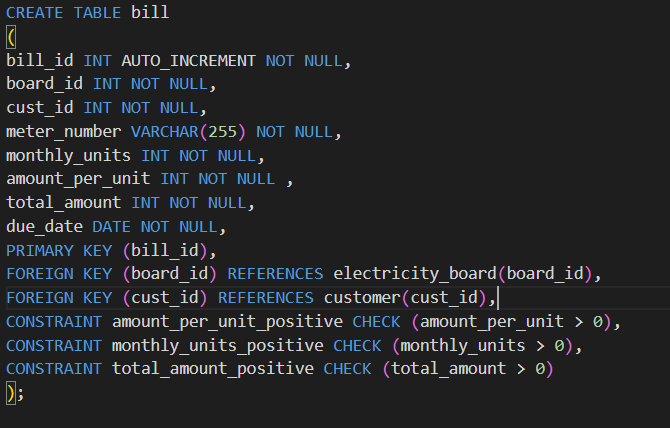


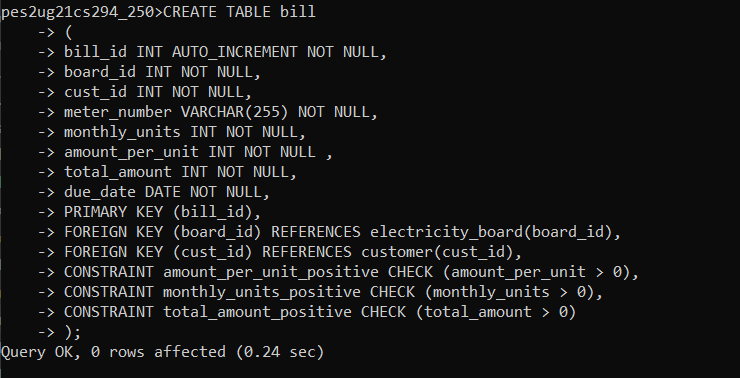
**Creation of the table “tariff”:**

****

****

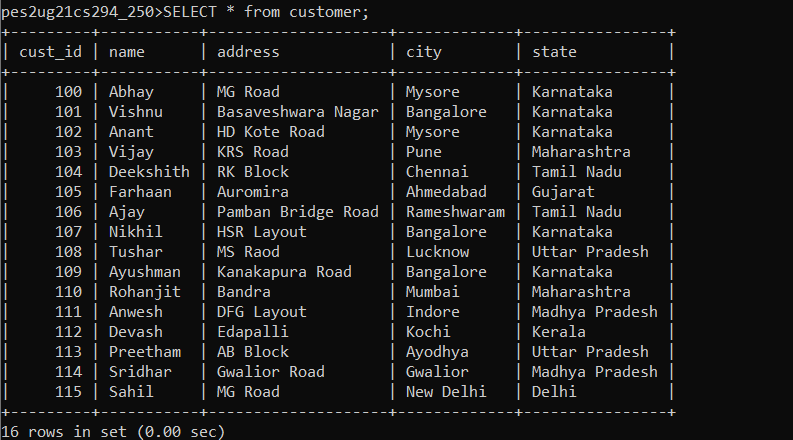
**Creation of the table “bill”:**

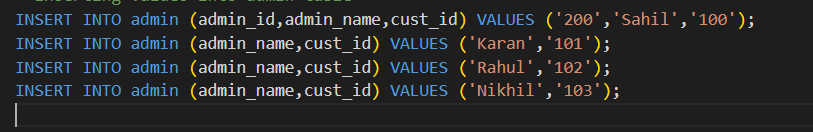
****

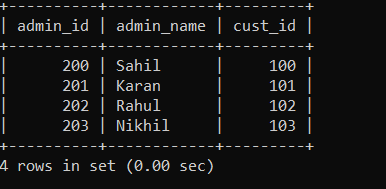
****

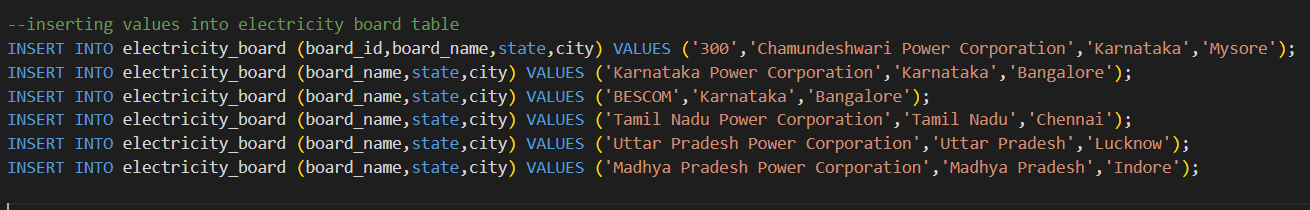
# 6. DML STATEMENTS

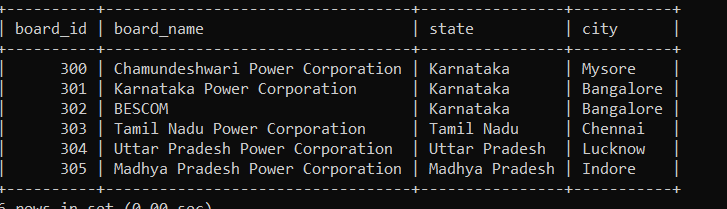
****

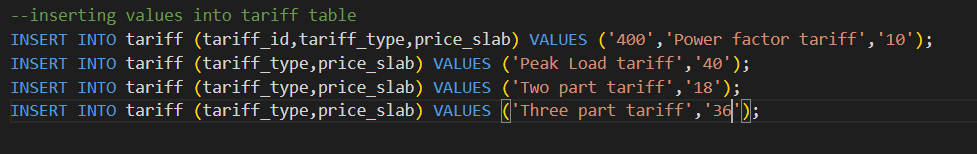
****

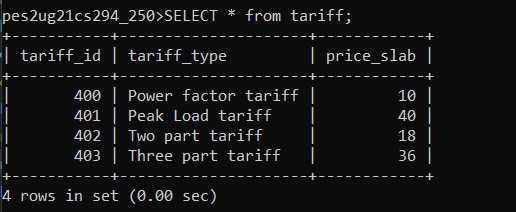
****

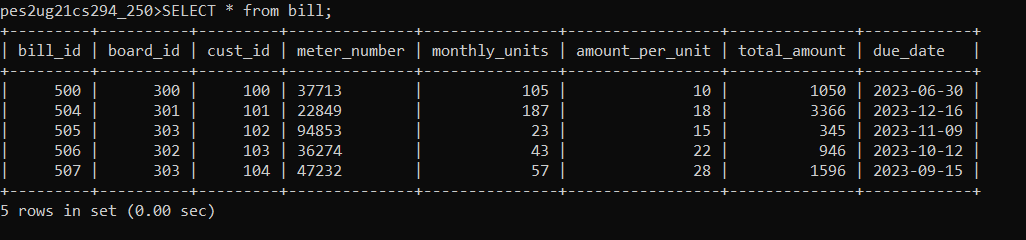
****

****

****

****

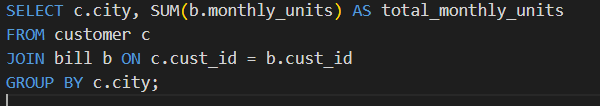
****

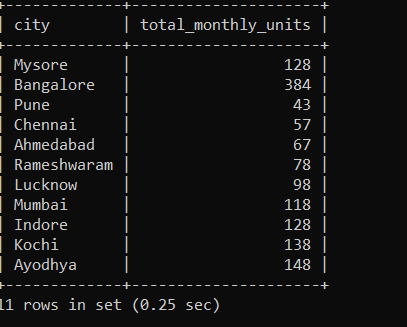
****

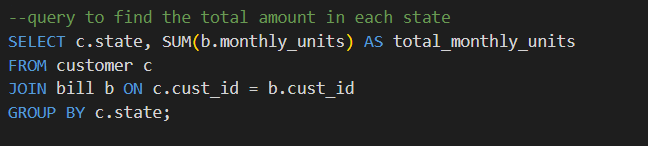
# 7. QUERIES

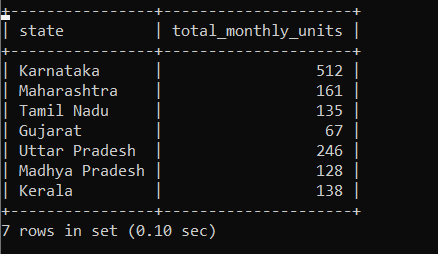
**7.1 SIMPLE QUERY WITH GROUP BY, AGRREGATE**

#### Query to find the total monthly units consumed by customers in each city:

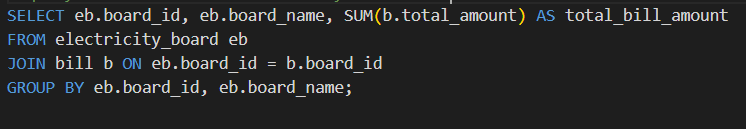
****

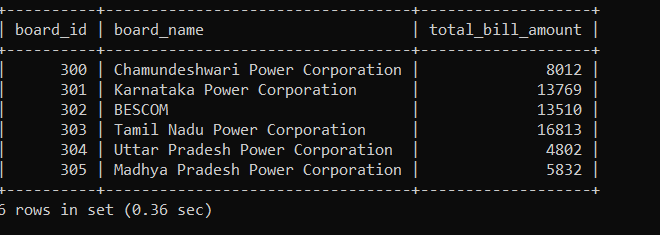
****





Query to find the bill amount collected by each electricity board





**7.2 UPDATE OPERATION**

--query to update the address of a customer

UPDATE customer

SET address = 'New Address'

WHERE cust\_id = 101;

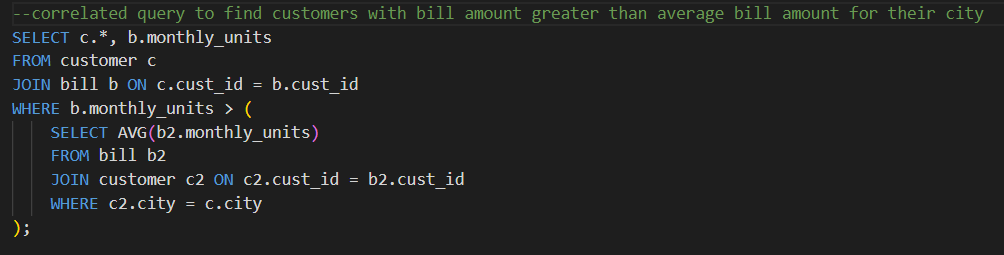
**7.3 DELETE OPERATION**

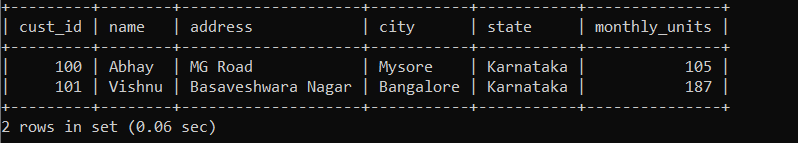
--query to delete a customer

DELETE FROM customer

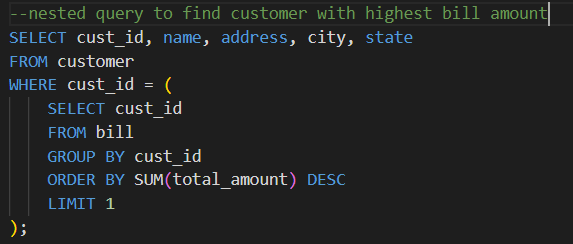
WHERE cust\_id = 101;

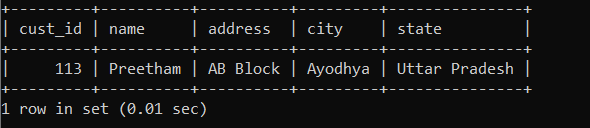
**7.4 CORRELATED QUERY**

****

****

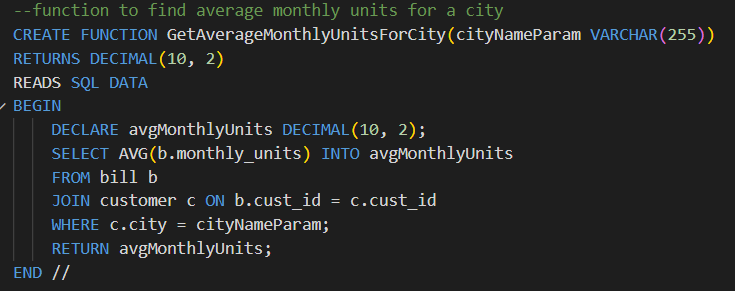
**7.5 NESTED QUERY**

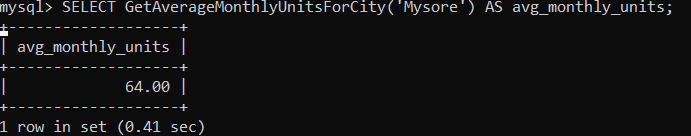
****

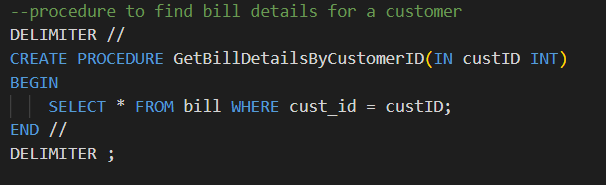
****

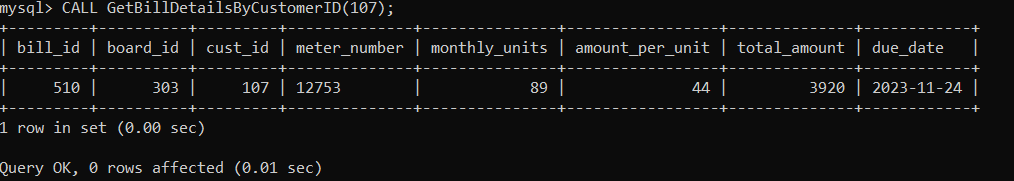
# 8. STORED PROCEDURES, FUCNTIONS AND TRIGGERS

**8.1 STORED PROCEDURES OR FUNCTIONS**

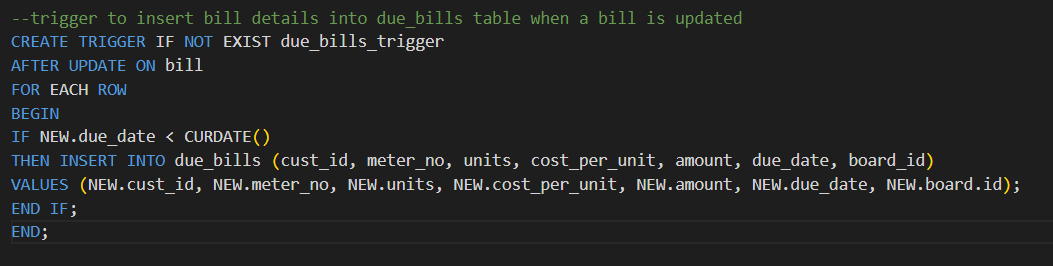
****

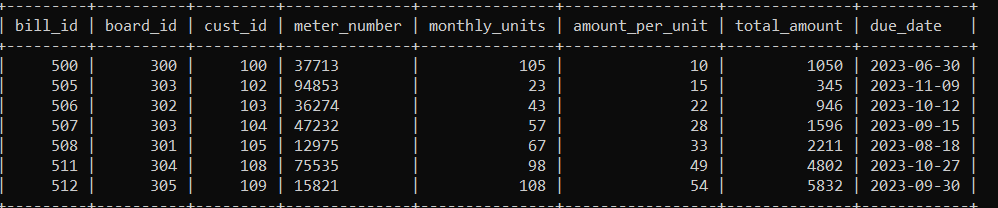
****

****

****

**8.2 TRIGGERS**

****

****

# 9. FRONT END DEVELOPMENT

The frontend for this electricity bill management system has been created with streamlit.

Some parts of the frontend code have been added since the entire code cannot be added here due to its size. The entire source code for this project can be found on the [Project's Github Repository](https://github.com/melvinjjoseph/dbms-project).

main.py

import streamlit as st

import mysql.connector

from database import \*

from customer import \*

from admin import \*

from bill import \*

from tariff import \*

from board import \*

def main():

    st.title("Electricity management system")

    choose=st.sidebar.radio("Select whether you are customer or admin",["Customer","Admin"])

    if choose=="Customer":

        menu=["Home","Customer", "Billing"]

        choice=st.sidebar.selectbox("Menu",menu)

    if choose=="Admin":

        menu=["Home","Customer","Admin", "Billing", "Tariff", "Electricity Boards", "Show Due Bills", "Custom Query"]

        choice=st.sidebar.selectbox("Menu",menu)

    if choice=="Home":

        st.subheader("Home")

        st.header("Welcome to the electricity bill management system")

        st.write("This is a simple electricity bill management system")

        st.write("Please select a menu option from the sidebar")

    if choice == "Customer":

        st.subheader("Customer details")

        customer\_menu=["Add","View","Update","Delete"]

        customer\_choice=st.selectbox("Menu",customer\_menu)

        if customer\_choice=="Add":

            st.subheader("Enter details")

            create\_customer()

        elif customer\_choice=="View" :

            view\_customer()

        elif customer\_choice=="Update" :

            update\_customer()

        elif customer\_choice=="Delete" :

            delete\_customer()

    if choice == "Admin":

        st.subheader("Admin")

        admin\_menu=["Add","View","Update","Delete"]

        admin\_choice=st.selectbox("Menu",admin\_menu)

        if admin\_choice=="Add":

            st.subheader("Enter details")

            create\_admin()

        elif admin\_choice=="View" :

            read\_admin()

        elif admin\_choice=="Update" :

            update\_admin()

        elif admin\_choice=="Delete" :

            delete\_admin()

    if choice == "Billing":

        st.subheader("Billing")

        billing\_menu=["Add Bill","View Bills","Update Bills","Delete Bills"]

        billing\_choice=st.selectbox("Menu",billing\_menu)

        if billing\_choice=="Add Bill":

            create\_bill()

        elif billing\_choice=="View Bills" :

            read\_bill()

        elif billing\_choice=="Update Bills" :

            update\_bill()

        elif billing\_choice=="Delete Bills" :

            delete\_bill()

    if choice == "Tariff":

        st.subheader("Tariff")

        tariff\_menu=["Add","View","Update","Delete"]

        tariff\_choice=st.selectbox("Menu",tariff\_menu)

        if tariff\_choice=="Add":

            create\_tariff()

        elif tariff\_choice=="View" :

            read\_tariff()

        elif tariff\_choice=="Update" :

            update\_tariff()

        elif tariff\_choice=="Delete" :

            delete\_tariff()

    if choice == "Electricity Boards":

        st.subheader("Electricity Boards")

        eb\_menu=["Add","View","Update","Delete"]

        eb\_choice=st.selectbox("Menu",eb\_menu)

        if eb\_choice=="Add":

            create\_eb()

        elif eb\_choice=="View" :

            read\_eb()

        elif eb\_choice=="Update" :

            update\_eb()

        elif eb\_choice=="Delete" :

            delete\_eb()

    if choice =="Show Due Bills":

        st.subheader("Due Bills")

        due\_bills()

    if choice == "Custom Query":

        st.subheader("Custom Query")

        query=st.text\_input("Enter query")

        submit=st.button("Submit")

        if submit:

            mycursor.execute(query)

            myresult = mycursor.fetchall()

            for x in myresult:

                st.write(x)

main()

database.py

import streamlit as st

import mysql.connector

import pandas as pd

mydb = mysql.connector.connect(user="root", password="246810", host="localhost")

mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE IF NOT EXISTS electricity")

mycursor.execute("USE electricity")

def add\_customer(name,address,city,state):

    mycursor.execute("CREATE TABLE IF NOT EXISTS customer (cust\_id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), address VARCHAR(255), city VARCHAR(255), state VARCHAR(255))")

    sql = "INSERT INTO customer (name, address, city, state) VALUES (%s, %s, %s, %s)"

    val = (name, address, city, state)

    mycursor.execute(sql, val)

    mydb.commit()

    return mycursor.lastrowid

def read\_customer():

    mycursor.execute("SELECT \* FROM customer")

    myresult = mycursor.fetchall()

    st.write("Customer details")

    df=pd.DataFrame(myresult,columns=['cust\_id','name','address','city','state'])

    st.dataframe(df)

def delete\_customer\_db(cust\_id):

    #check if cust\_id exists

    mycursor.execute("SELECT \* FROM customer WHERE cust\_id = %s",(cust\_id,))

    myresult = mycursor.fetchall()

    if not myresult:

        st.write("cust\_id does not exist")

        return

    sql = "DELETE FROM customer WHERE cust\_id = %s"

    val = (cust\_id,)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Customer deleted successfully")

def update\_customer\_db(cust\_id,name,address,city,state):

    sql = "UPDATE customer SET name = %s, address = %s, city = %s, state = %s WHERE cust\_id = %s"

    val = (name, address, city, state, cust\_id)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Customer updated successfully")

def add\_admin(name,Customer\_id):

    mycursor.execute("CREATE TABLE IF NOT EXISTS admin (admin\_id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), Customer\_id INT NOT NULL, FOREIGN KEY (Customer\_id) REFERENCES customer(cust\_id))")

    sql = "INSERT INTO admin (name, Customer\_id) VALUES (%s, %s)"

    val = (name, Customer\_id)

    mycursor.execute(sql, val)

    mydb.commit()

    return mycursor.lastrowid

def read\_admin():

    mycursor.execute("SELECT \* FROM admin")

    myresult = mycursor.fetchall()

    st.write("Admin details")

    df=pd.DataFrame(myresult,columns=['admin\_id','name','Customer\_id'])

    st.dataframe(df)

def update\_admin\_db(admin\_id,name,Customer\_id):

    sql = "UPDATE admin SET name = %s, Customer\_id = %s WHERE admin\_id = %s"

    val = (name, Customer\_id, admin\_id)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Admin updated successfully")

def delete\_admin\_db(admin\_id):

    #check if admin\_id exists

    mycursor.execute("SELECT \* FROM admin WHERE admin\_id = %s",(admin\_id,))

    myresult = mycursor.fetchall()

    if not myresult:

        st.write("admin\_id does not exist")

        return

    sql = "DELETE FROM admin WHERE admin\_id = %s"

    val = (admin\_id,)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Admin deleted successfully")

def add\_bill(cust\_id,meter\_no,units,cost\_per\_unit,due\_date,board\_id):

    mycursor.execute("CREATE TABLE IF NOT EXISTS bill (bill\_id INT AUTO\_INCREMENT PRIMARY KEY, cust\_id INT NOT NULL, meter\_no INT NOT NULL, units INT NOT NULL, cost\_per\_unit INT NOT NULL, amount INT NOT NULL, due\_date DATE NOT NULL board\_id INT NOT NULL, FOREIGN KEY (cust\_id) REFERENCES customer(cust\_id), FOREIGN KEY (board\_id) REFERENCES Board(eb\_id))")

    sql = "INSERT INTO bill (cust\_id, meter\_no, units, cost\_per\_unit, amount, due\_date) VALUES (%s, %s, %s, %s, %s, %s)"

    val = (cust\_id, meter\_no, units, cost\_per\_unit, int(units)\*int(cost\_per\_unit), due\_date, board\_id)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Bill added successfully")

    return mycursor.lastrowid

def read\_bill():

    mycursor.execute("SELECT \* FROM bill")

    myresult = mycursor.fetchall()

    st.write("Bill details")

    df=pd.DataFrame(myresult,columns=['bill\_id','cust\_id','meter\_no','units','cost\_per\_unit','amount','due\_date', 'board\_id'])

    st.dataframe(df)

def update\_bill\_db(bill\_id,cust\_id,meter\_no,units,cost\_per\_unit, due\_date, board\_id):

    sql = "UPDATE bill SET cust\_id = %s, meter\_no = %s, units = %s, cost\_per\_unit = %s, amount = %s, due\_date=%s, board\_id=%s WHERE bill\_id = %s"

    val = (cust\_id, meter\_no, units, cost\_per\_unit, int(units)\*int(cost\_per\_unit), due\_date, board\_id, bill\_id)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Bill updated successfully")

def delete\_bill\_db(bill\_id):

    #check if bill\_id exists

    mycursor.execute("SELECT \* FROM bill WHERE bill\_id = %s",(bill\_id,))

    myresult = mycursor.fetchall()

    if not myresult:

        st.write("bill\_id does not exist")

        return

    sql = "DELETE FROM bill WHERE bill\_id = %s"

    val = (bill\_id,)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Bill deleted successfully")

def add\_tariff(tariff\_type,tariff\_cost):

    mycursor.execute("CREATE TABLE IF NOT EXISTS tariff (tariff\_id INT AUTO\_INCREMENT PRIMARY KEY, tariff\_type VARCHAR(255), tariff\_cost INT NOT NULL)")

    sql = "INSERT INTO tariff (tariff\_type, tariff\_cost) VALUES (%s, %s)"

    val = (tariff\_type, tariff\_cost)

    mycursor.execute(sql, val)

    mydb.commit()

    return mycursor.lastrowid

def read\_tariff():

    mycursor.execute("SELECT \* FROM tariff")

    myresult = mycursor.fetchall()

    st.write("Tariff details")

    df=pd.DataFrame(myresult,columns=['tariff\_id','tariff\_type','tariff\_cost'])

    st.dataframe(df)

def update\_tariff\_db(tariff\_id,tariff\_type,tariff\_cost):

    sql = "UPDATE tariff SET tariff\_type = %s, tariff\_cost = %s WHERE tariff\_id = %s"

    val = (tariff\_type, tariff\_cost, tariff\_id)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Tariff updated successfully")

def delete\_tariff\_db(tariff\_id):

    sql = "DELETE FROM tariff WHERE tariff\_id = %s"

    val = (tariff\_id,)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Tariff deleted successfully")

def add\_eb(name,city,state):

    mycursor.execute("CREATE TABLE IF NOT EXISTS Board (eb\_id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), city VARCHAR(255), state VARCHAR(255))")

    # mycursor.execute("INSERT INTO Board (eb\_id, name, city, state) VALUES ('100','Karnataka Power Transmission Corporation Limited', 'Bengaluru', 'Karnataka')")

    sql = "INSERT INTO Board (name, city, state) VALUES (%s, %s, %s)"

    val = (name, city, state)

    mycursor.execute(sql, val)

    mydb.commit()

    return mycursor.lastrowid

def read\_eb():

    mycursor.execute("SELECT \* FROM Board")

    myresult = mycursor.fetchall()

    st.write("Electricity board details")

    df=pd.DataFrame(myresult,columns=['eb\_id','name','city','state'])

    st.dataframe(df)

def update\_eb\_db(eb\_id,name,city,state):

    sql = "UPDATE Board SET name = %s, city = %s, state = %s WHERE eb\_id = %s"

    val = (name, city, state, eb\_id)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Electricity board updated successfully")

def delete\_eb\_db(eb\_id):

    sql = "DELETE FROM Bpard WHERE eb\_id = %s"

    val = (eb\_id,)

    mycursor.execute(sql, val)

    mydb.commit()

    st.write("Electricity board deleted successfully")

# REFERENCES

[1] <https://docs.streamlit.io/>

[2] <https://dev.mysql.com/doc/connector-python/en/>