Retailer Sales Company

Sudheer Daggubati

GIT HUB URL: https://github.com/sudheer34/Mycode.git  
December 15’ 2022

A department store's general operation and function depend heavily on the administration and management of the store. There will be many products and categories, especially in the case of larger stores. Therefore, efficient management and good resource management are essential for success and smooth operation. So, to start our company that makes toys for kids and sells them on our website and mobile app, we must have a plan to develop and start our company in the city. Gradually we have plan to open more and more branches of it to enhance the company size and generate more revenue, we are planning to open our company into Mexico state. The initial size of our company is about 7-10 employees and next we will increase and enhance our company with time. We will be selling toy products named cartoons or characters to make kids indulge into them. Like ben10, Pokémon, Avengers, Superman, Batman, Animal Kingdom, Doraemon etc. As far as annual sales are concerned, we are planning to generate 1 million dollars.

Target is the one retailer company which we found like our plan. They have categorized the popular and exciting face characters of the kids to sell their clothes. We are doing the same, the different thing is that we are making their toys to sell. Target has generated 106$ billion in 2022 and has 1938 stores around the US. They employee 400,000+ members in their company. (<https://corporate.target.com/about>)

For our system we are going to use the 3-tier architecture like below. This is the generic architecture from which we have our client interaction with presentation layer (UI layer) which will send request to backend business logic which follows the data retrieval from the data component layer and make it in some presentable state to present back it on presentation layer.

Diagram

Description automatically generated

For our business we need to store inventory, prices, quantities, product names, etc. So, for this purpose we may need following data types:

varchar, int, number, bool and varchar

To translate data from one system to another system we can use Celigo, Cleo or Cyclr etc.

Customers of these are: Amazon, Amazon web services, Braintree, asana, 3dcart, ADP etc. (https://solutionsreview.com/data-integration/the-best-data-transformation-tools-and-software/)

The common data formats that computer systems use to transfer product data between systems are CSV and JSON and XML.

Entities:

**Customer:** Describes the information of customer. Customer, Name, Email, Address.

**Product:** Describes the information of products. Product\_ID, Name, Price.

**Stock:** Describes the information of stock in hand. Stock\_ID, Product\_ID, Available\_Quantity.

**Sales:** Describes the information of sales record. Sales\_ID, Product\_ID, Customer\_ID, Quantity, Total\_Amount.

**Payment:** Describes the information of payments and bills. Payment\_ID, Sales\_ID, Payment\_Method, Date.

Tables:

|  |  |
| --- | --- |
| **Customer** | |
| Customer | Integer |
| Name | Varchar(255) |
| Email | Varchar(255) |
| Address | Varchar(255) |

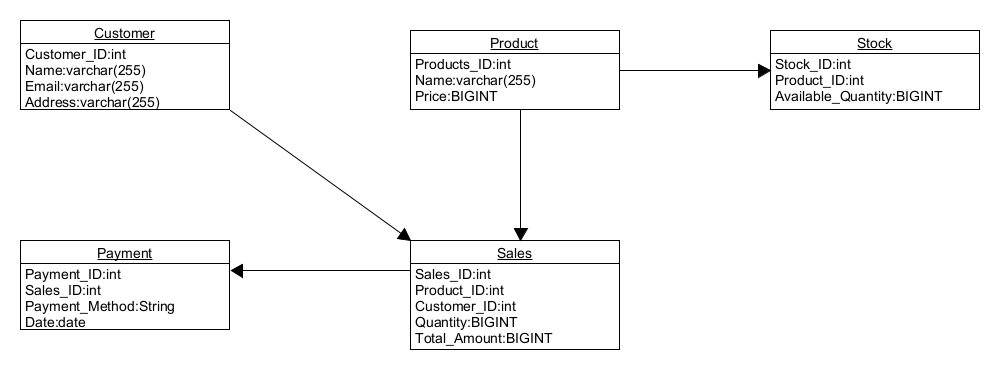
|  |  |
| --- | --- |
| **Product** | |
| Product\_ID | Integer |
| Name | Varchar(255) |
| Price | BIGINT |

|  |  |
| --- | --- |
| **Sales** | |
| Sales\_ID | Integer |
| Product\_ID | Integer |
| Customer\_ID | Integer |
| Quantity | BIGINT |
| Total\_Amount | BIGINT |

|  |  |
| --- | --- |
| **Payment** | |
| Payment\_ID | Integer |
| Sales\_ID | Integer |
| Payment\_Method | String |
| Date | date |

|  |  |
| --- | --- |
| **Stock** | |
| Stock\_ID | Integer |
| Product\_ID | Integer |
| Available\_Quantity | BIGINT |

UMLet:



Crow Foot’s Notation:

**Diagram

Description automatically generated**

Relationships between tables:

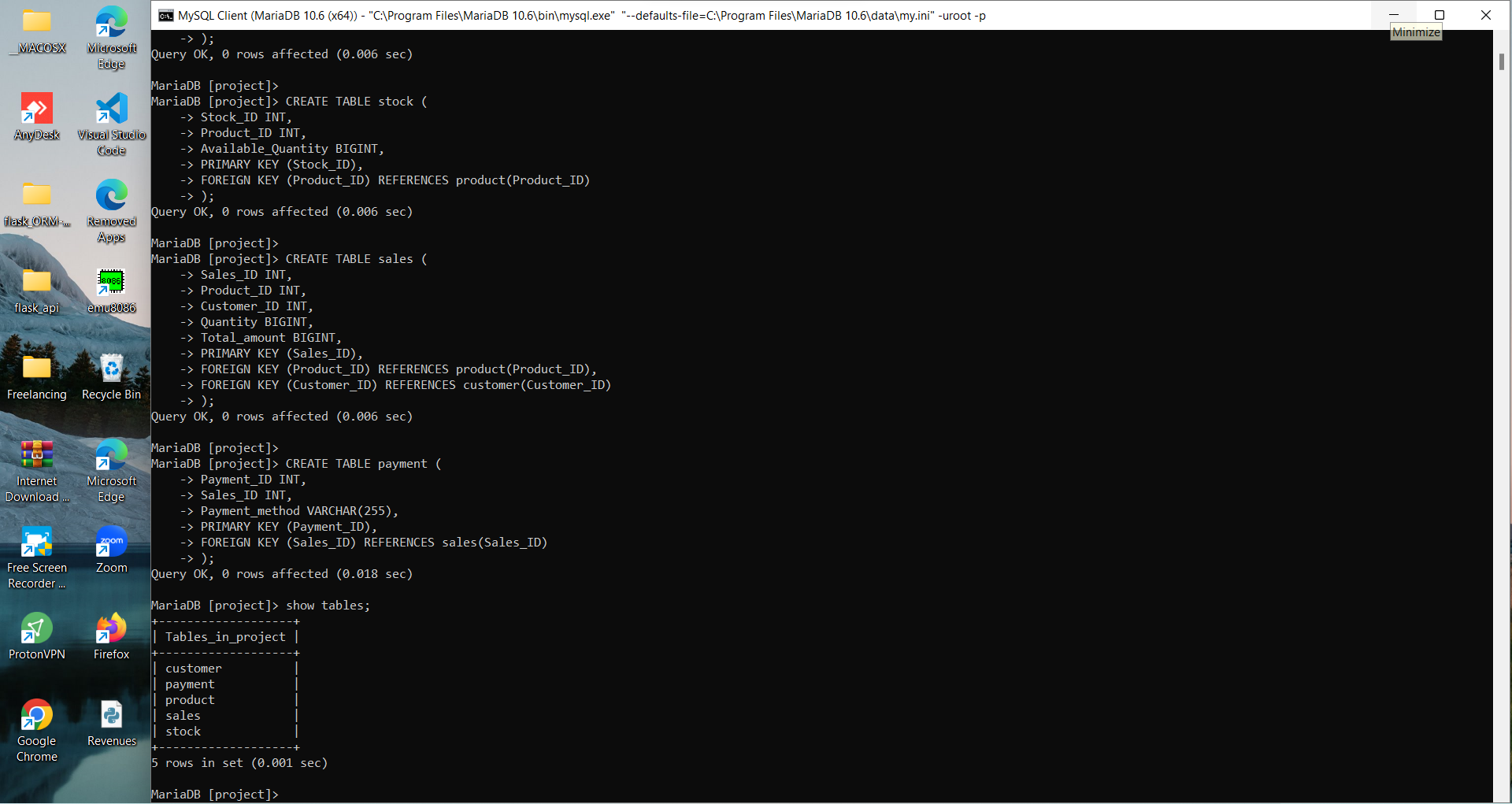
Product-Sales: Product 🡨🡪 Sales (one to many relationship)

Customer-Sales: Customer 🡨🡪 Sales (one to many relationship).

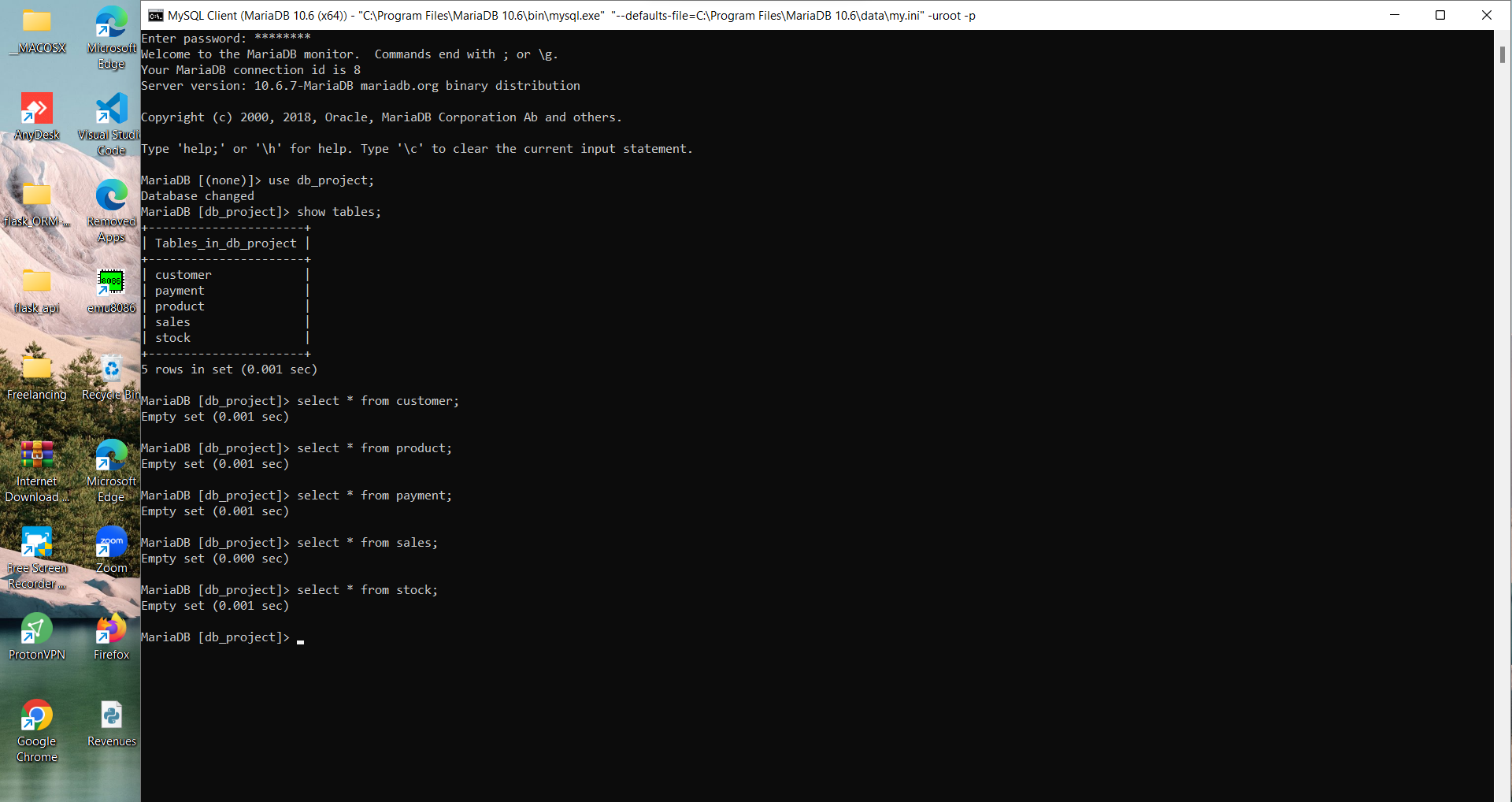
Product-Stock: Product 🡨🡪 Stock (one to one relationship).

Sales-Payment: Sales 🡨🡪 Payment (one to one relationship).

1. Screenshot showing the table creations:



1. Screenshot showing empty tables:



1. Screenshot showing table populating queries and populated tables:

Text

Description automatically generated

Twelve SQL Queries:

1,2 Two insert queries

Text

Description automatically generated

3,4 Two update queries

Text

Description automatically generated

5 One delete query

Text

Description automatically generated

6 One select subset of table query

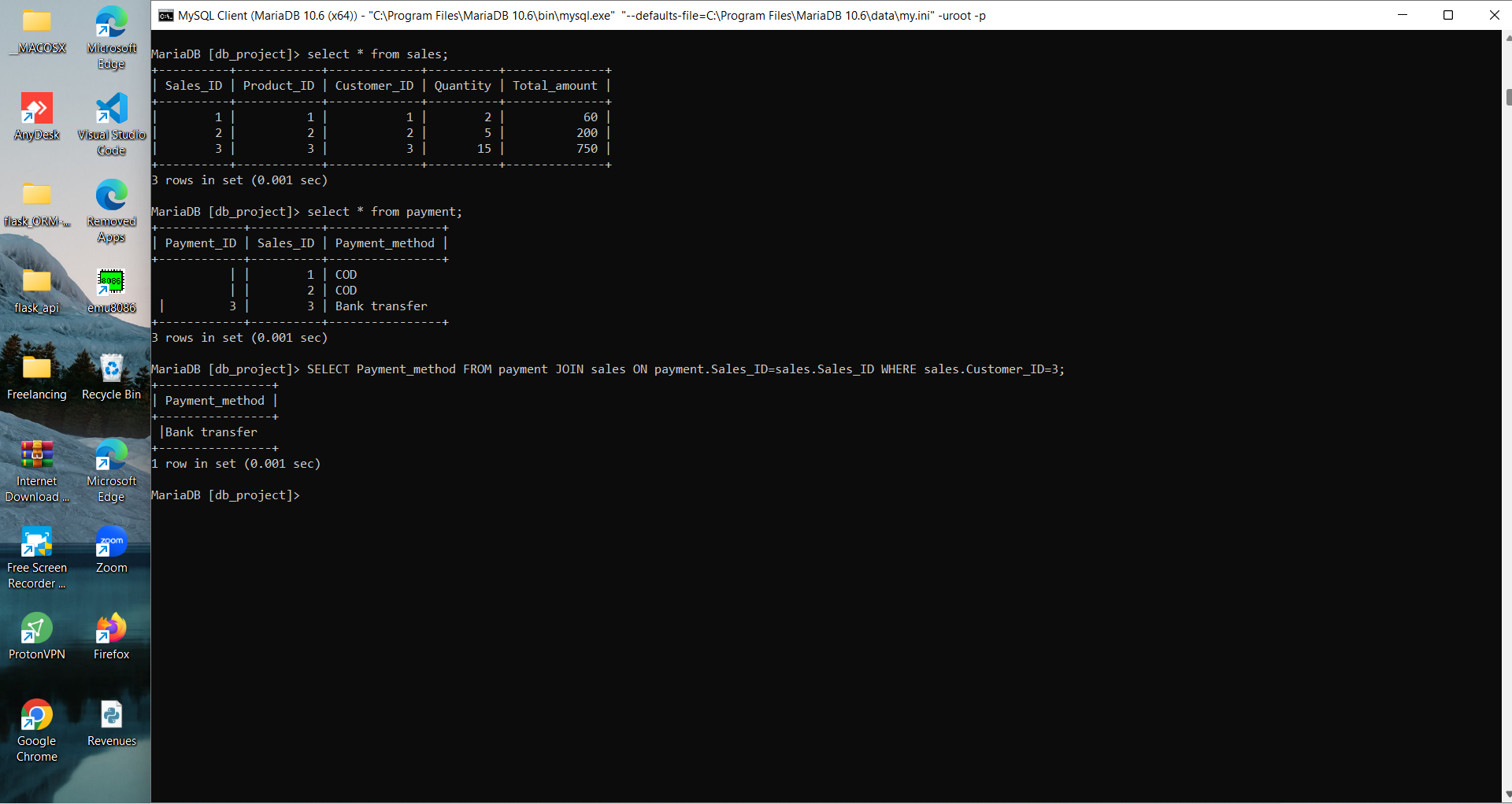
Text

Description automatically generated

7,8 Two select queries using join

Text

Description automatically generated



9,10,11 Three summary function queries

Text

Description automatically generated

12 One Multi-table Query

Text

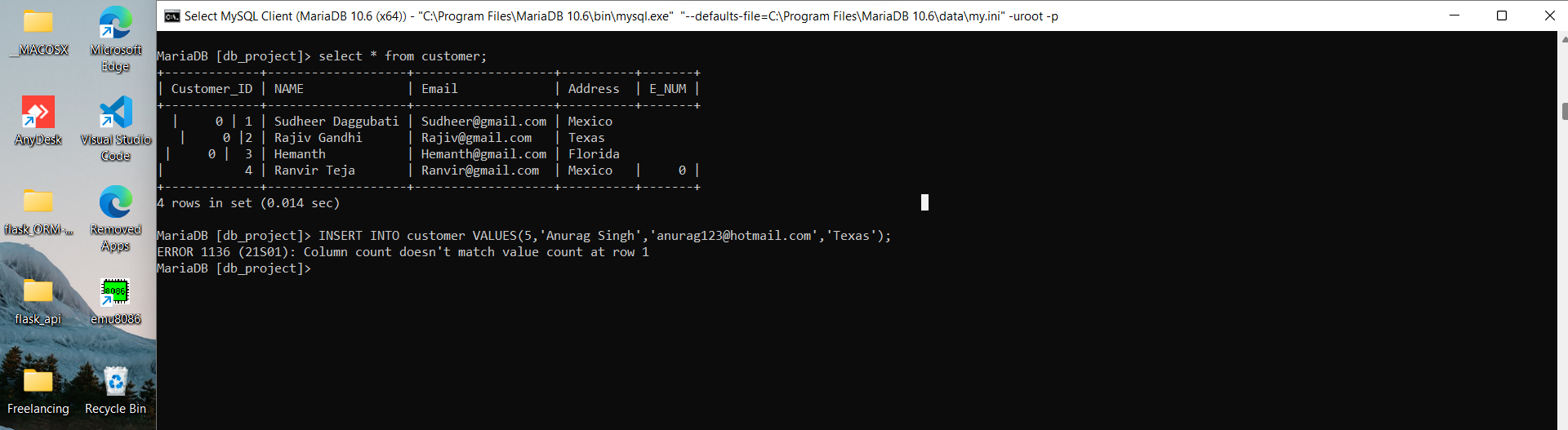
Description automatically generated

Part 1: ADD E\_NUM column:

Text

Description automatically generated

Part 2a: Error:



Part 2b: Modified Insert Query:

Text

Description automatically generated

Part 3: Update Query to add all values of E\_NUM:

To update all the null values for the E\_NUM column, I used the UPDATE statement to all the Customer\_ID where it is null. It updates all the records of E\_NUM null values into customer table.

Text

Description automatically generated with medium confidence

Part 4: DELETE Query to delete recent record of E\_NUM:

Text

Description automatically generated

Part 5: Auto\_increment column Query:

Auto increment is the constraint on a column which will be added to a key column and for that key column it will automatically add next value of the ID as accordance with the previous serial IDs.

Text

Description automatically generated

Part 6: default value for column Query:

Text

Description automatically generated

**Python Programming**

import mysql.connector

from mysql.connector import errorcode

try:

    cm\_connection = mysql.connector.connect(

        user="root",

        password="12345",

        port="3306",

        host="127.0.0.1",

        database="nation")

except mysql.connector.Error as err:

    if err.errno == errorcode.ER\_ACCESS\_DENIED\_ERROR:

        print("Invalid credentials")

    elif err.errno == errorcode.ER\_BAD\_DB\_ERROR:

        print("Database not found")

    else:

        print("Cannot connect to database:", err)

else:

    customer\_cursor = cm\_connection.cursor()

    customer\_cursor.execute("CREATE TABLE customers (id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), address VARCHAR(255))")

    sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

    val = ("John", "Highway 21")

    customer\_cursor.execute(sql, val)

    sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

    val = ("Alex", "Highway 15")

    customer\_cursor.execute(sql, val)

    sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

    val = ("James", "Highway # 11 Street 5")

    customer\_cursor.execute(sql, val)

    sql = "DELETE FROM customers WHERE address = 'Highway 15'"

    customer\_cursor.execute(sql)

    cm\_connection.commit()

    customer\_query = ("SELECT name, address FROM Customers")

    customer\_cursor.execute(customer\_query)

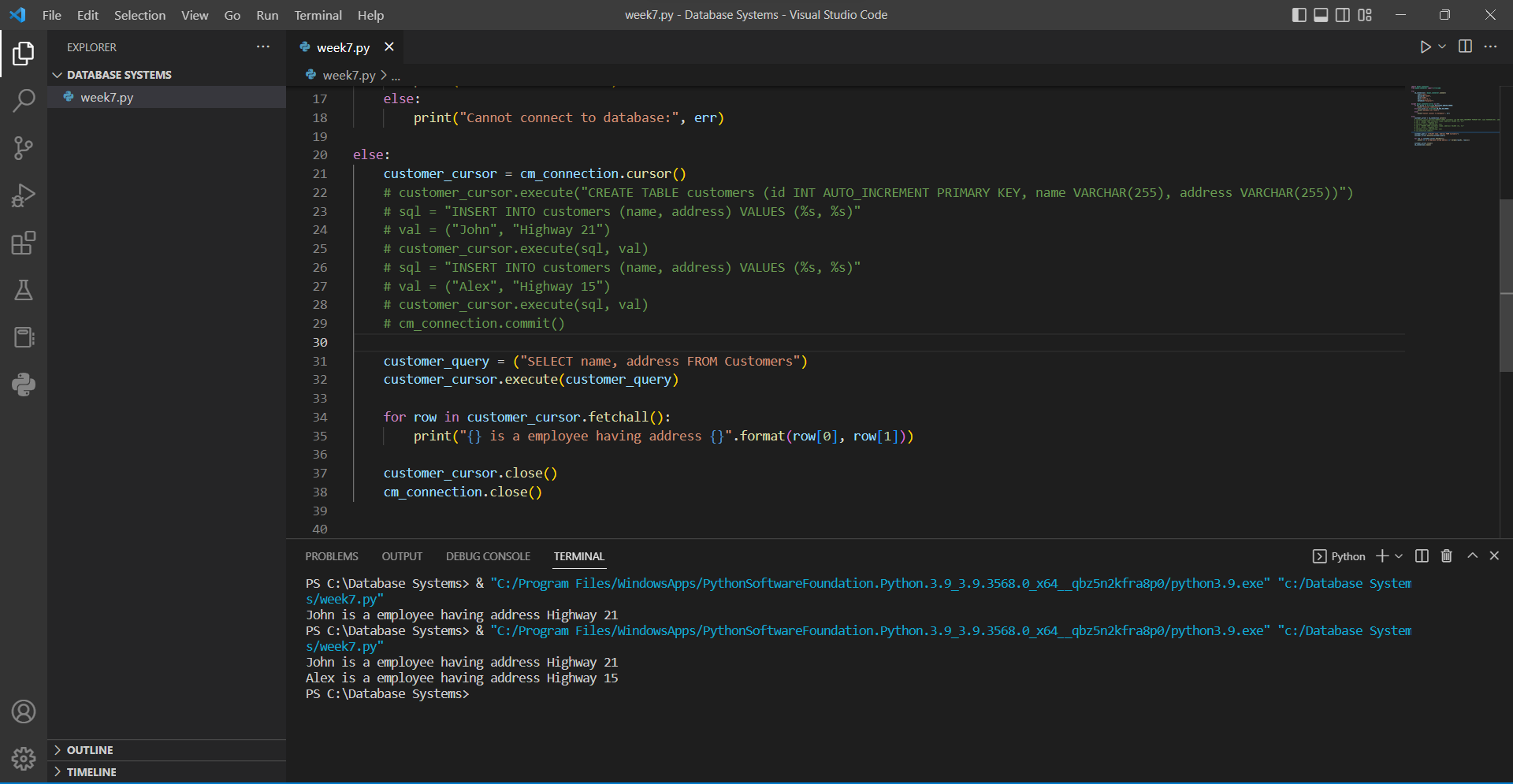
    for row in customer\_cursor.fetchall():

        print("{} is a employee having address {}".format(row[0], row[1]))

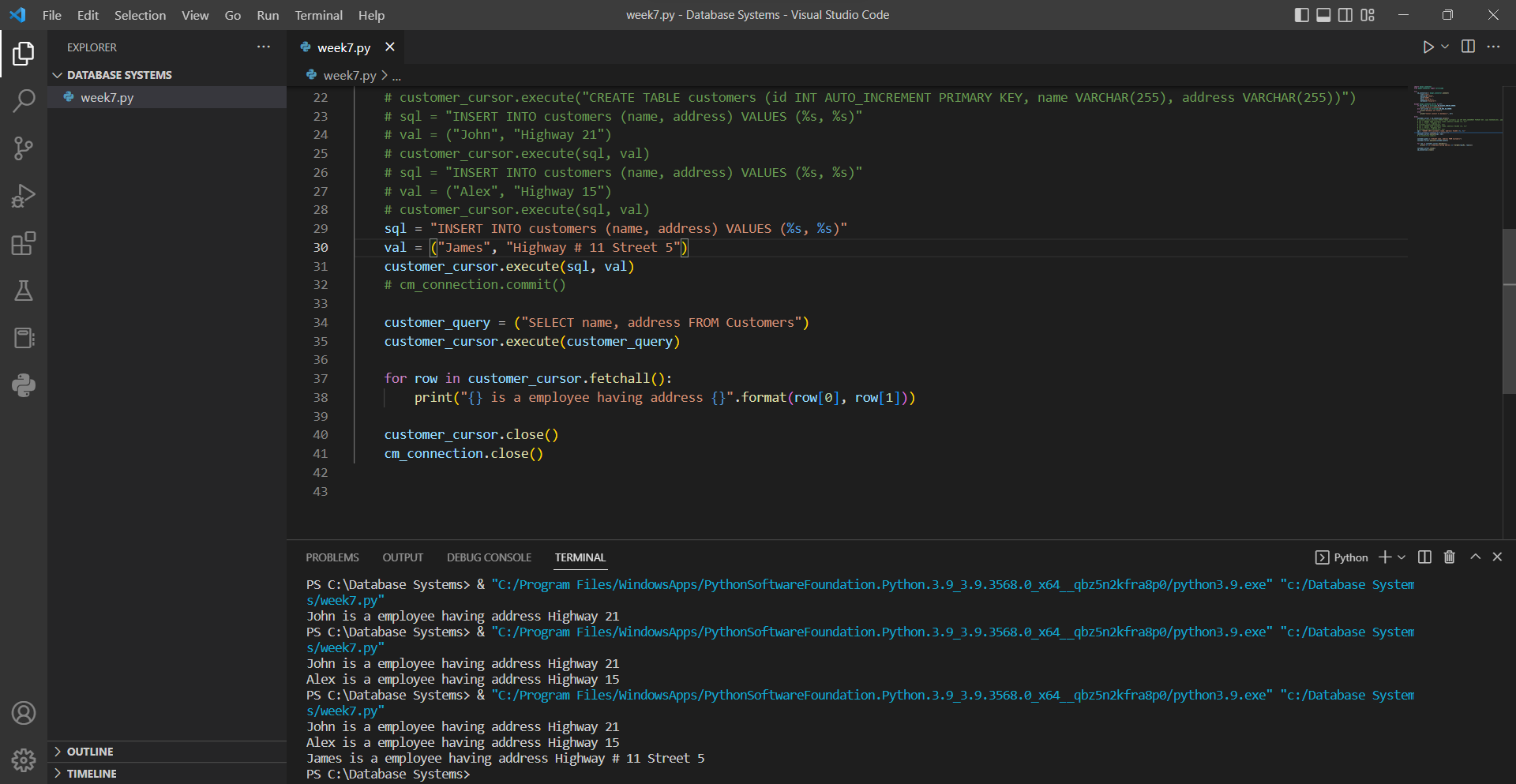
    customer\_cursor.close()

    cm\_connection.close()

* For Part 1: Display entire table



* For Part 2: Insert a new row and display the updated table



* For Part 3: Delete a row and display

