Database Systems
Project: Design, development and implementation of a relational database
Professor Venessa Aguiar
By: Spencer Terwilliger, Julian Cantillo, Andrew Shatsky
GitHub Link: https://github.com/Jncantillo/CSC423Project

The given case study for this project was the Super Maids cleaning company. Starting off by developing a conceptual data model, we first need to satisfy certain requirements such as: main entity types, main relationship types, multiplicity constraints, identifying attributes and associate them with entity or relationship types, and determining candidate and primary keys for each entity. For the Super Maids Cleaning Company, we have determined that the main entity types are: Client, Employee, Service Requirements, and Equipment. The following table shows the relationship types, and multiplicity constraints for each of the main entities:

Entity 1	**Relationship**	**Entity 2**	**Particip-ation**	**Cardinality**	**Multiplicity**	**Type of relation-ship**
Client	**Has**	Service Requirements	1	*	1..*	1:*
Service Requirements	Relates to	Client	1	1	1..1	
Staff	**Assigned to**	Client	0	*	0..*	*:*
Client	Hosts	Staff	1	*	1..*	
Service Requirements	**Uses**	Equipment	1	*	1..*	*:*
Equipment	Needed for	Service Requirements	1	*	1..*	

Staff	**Utilizes**	Equipment	1	*	1..*	*:*
Equipment	Used by	Staff	1	*	1..*	

We assume that the client and equipment do not have a direct relationship because they are accessed through the use of foreign and primary keys. We identified every attribute for each entity and relationship as well as identifying the foreign and primary keys:

Client:
-Client Number (Primary Key)
-First Name
-Last Name
-Address
-Telephone Number

Employee:
-Staff Number (Primary Key)
-First Name
-Last Name
-Address
-Salary
-Telephone Number

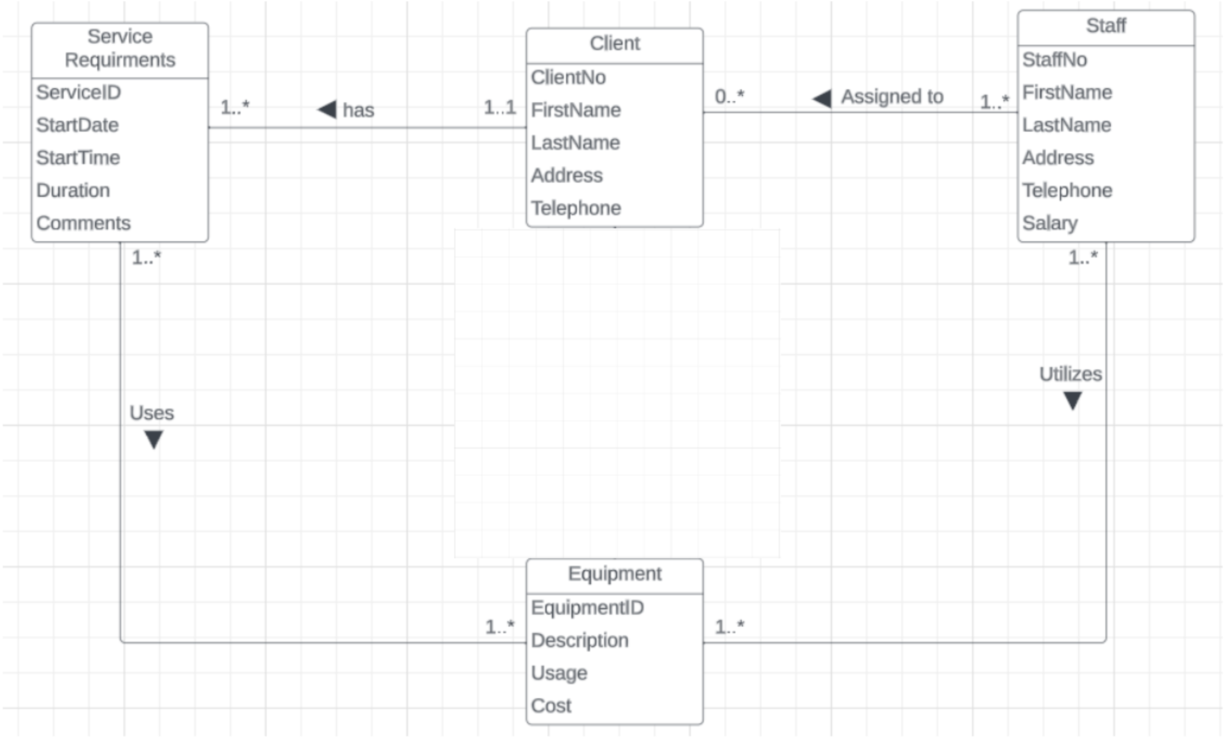
Service Requirements:
-Requirement ID (Primary Key) (Foreign Key)
-Start Date
-Start Time
-Duration
-Comments
-Client Number (Foreign Key)

Equipment:
-Equipment ID (Primary Key) (Foreign Key)
-Description
-Usage
-Cost

Assignment (Relationship):
-Staff Number (Foreign Key)
-Requirement ID (Foreign Key)

Requirement Equipment (Relationship):
-Equipment ID (Foreign Key)
-Requirement ID (Foreign Key)
-Usage Frequency

Now looking at an ER (Entity Relationship) diagram at the conceptual level using the data above without any foreign keys as attributes:



Now developing a logical data model following the Super Maids Cleaning company, we first need to start off by deriving the relations from the conceptual model. We did list by creating a list of relations using the conceptual model leaving us with the lists:

Client Relation

clientNo (primary key)	firstName	LastName	address	PhoneNumber

Requirement Relation

RequirmentID (primary key)	ClientNo (foreign key)	StartDate	StartTime	Duration	Comments

Equipment Relation

EquipmentID (primary key)	Description	usage	Cost

Employee Relation

StaffNo (primary key)	FirstName	LastName	Address	Salary	Phone Number

Equipment_Requirment Relation

EquipmentID (foreign key)(Primary key 1/2)	RequirmentID (foreign key) (primary key 2/2)	Quantity

Employee_Requirement Relation

StaffNo(foreign key) (Primary Key 1/2)	RequirmentID (foreign key) (Primary Key 2/2)

Proceeding from deriving relations from the conceptual model we need to validate the logical model using normalization to 3NF as well as validating the logical model against user transactions. We start we create a list of functional dependencies using the conceptual data model, leaving us with:

Client:
clientNumber → firstName, lastName, address, phoneNumber
Requirement:
requirementID → clientNo, startDate, startTime, duration, comments
Equipment:
equipmentID → description, usage, cost
Employee:
staffNo → firstName, lastName, address, salary
Equipment_Requirement:
{requirementID,equipmentID} → quantity
requirementID → equipmentID (Partial Dependency)
Employee_Requirement:
requirementID → staffNo (Partial Dependency)

To validate the model in 3NF we first must validate it in 1NF first, but because each row in each relation only contains a single value, the relations are in first normal form. Now we need to convert the 1NF normalization in 2NF, but because the Client, Requirement, Equipment, and Employee tables have no partial dependencies. The partial dependencies for the Equipment_Requirement and Employee_Requirement are acceptable since they make up the composite primary key. Since the table is also already in 1NF and has no problematic partial dependencies, it is also in Second Normal Form. Finally, because there are no transitive dependencies present, and the relations are in 2NF, the relations are also in Third Normal Form. Following validating the logical model using normalization we created a list of user transactions and solutions in order to validate the logical model against user transactions, the following list:

List of User Transactions and Solutions

1) Add/View Client/s
   a) This works since only the Client Relation would need to be accessed to add all of the information required for a new client or view all clients
2) Add/View Cleaning Requirement/s
   a) This requires access to just the Requirement Relation
3) Add Equipment
   a) This requires access to just the Equipment Relation
4) Add Employee
   a) This requires access to just the Employee Relation
5) Assign Equipment to Requirement / View equipment allocation status
   a) Joining the Equipment Relation to the Requirement Relation through the Equipment_Requirement relation would allow these transactions to happen.
   b) Equipment (eq) and Equipment_Requirement (eqr) would be joined by eq.staffNo = eqr.staffNo.
   c) Requirement (r) and Equipment_Requirement (eqr) would be joined by eqr.requirementID = r.requirementID
6) Assign Employee to Requirement / View status of all employees
   a) Joining the Employee Relation to the Requirement Relation through the Employee_Requirement relation would allow these transactions to happen.
   b) Employee (em) and Employee_Requirement (emr) would be joined by em.staffNo = emr.staffNo.
   c) Requirement (r) and Employee_Requirement (emr) would be joined by emr.requirementID = r.requirementID
7) Delete Cleaning Requirement
   a) Deleting a cleaning requirement would need the Requirement, Employee_Requirement, and Equipment_Requirement relations to be joined.
   b) Requirement (r) and Employee_Requirement (emr) would be joined by emr.requirementID = r.requirementID
   c) Requirement (r) and Equipment_Requirement (eqr) would be joined by eqr.requirementID = r.requirementID
8) Delete Client
   a) Since deleting a client could lead to the deletion of one or more cleaning requirements, it requires the steps outlined for the deletion of a cleaning requirement (#7).
   b) Deleting a client would also need the Client and Requirement relations to be joined
   c) Client (c) and Requirement (r) would be joined by c.clientNo = r.clientNo

Following the validation of logical model against user transaction, we determined and defined the integrity constraints. We defined the following integrity constraints, Primary key constraints, Foreign key constraints, Alternate key constraints, required data, Attribute domain constraints, and finally General Constraints. We determined these constraints as:

I. Primary key constraints:

- Client Relation

    - ClientNumber
- Requirments Relation
    - RequirmentsID
- Equipment Relation
    - EquipmentID
- Employee Relation
    - StaffNo

II. Foreign key constraints:

- Equipment_Requirment Relation
    - EquipmentID
- Equipment_Requirment Relation
    - RequirmentID
- Requirments Relation
    - clientNo
- Employee_Requirement Relation
    - staffNo
- Employee_Requirement Relation
    - RequirmentID

III. Alternate key constraints

- No alternate key constraints

IV. Required data

- Client Relation:
    - First_name
    - last_name
    - Address
    - telephone_number
- Equipment Relation:
    - Description
    - usage
    - Cost
- Employee Relation:
    - First_name
    - last_name
    - address
    - salary
    - telephone_number
- Requirement Relation:
    - start_date
    - start_time
    - duration

V. Attribute domain constraints

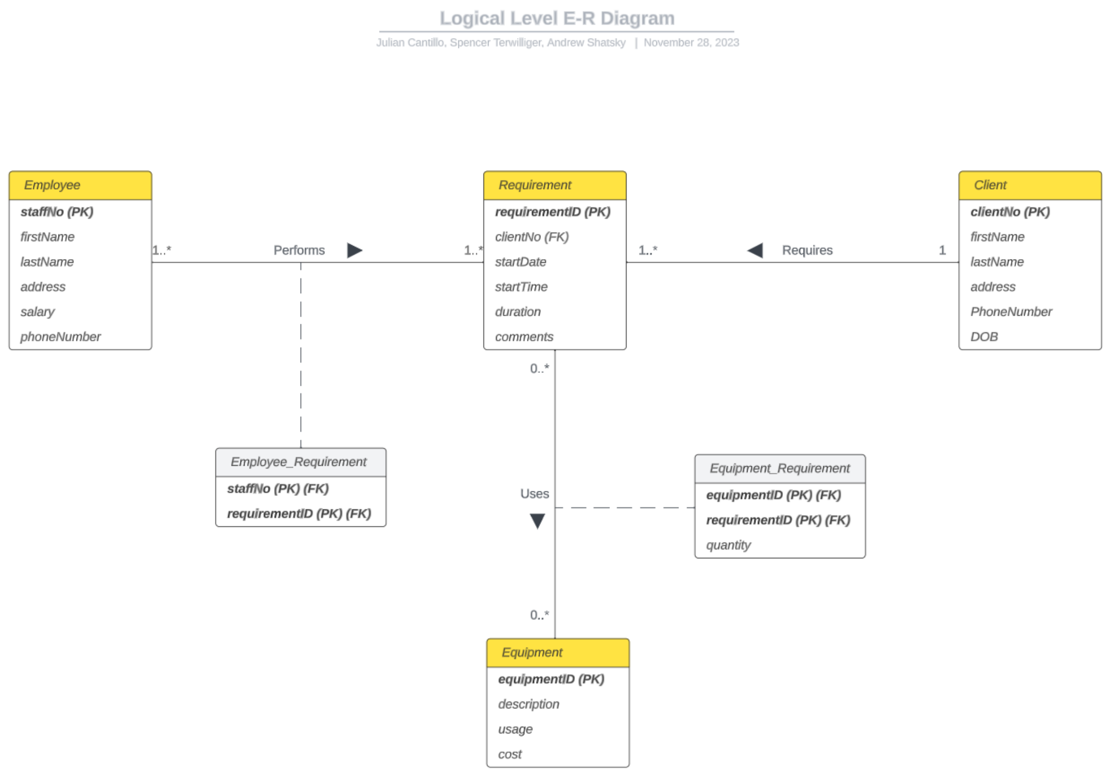
- Client: client number must be a unique identifier
- Client: first name, last name must be strings

- Employee: staff number must be a unique identifier
- Employee: first name, last name must be strings
- Equipment: description must be a string
- Equipment: equipment identifier must be a unique identifier
- Requirement: start date, start time, duration must be dates and times
- Requirement: comments must be strings
- Requirement: requirement identifier must be a unique identifier

VI. General constraints

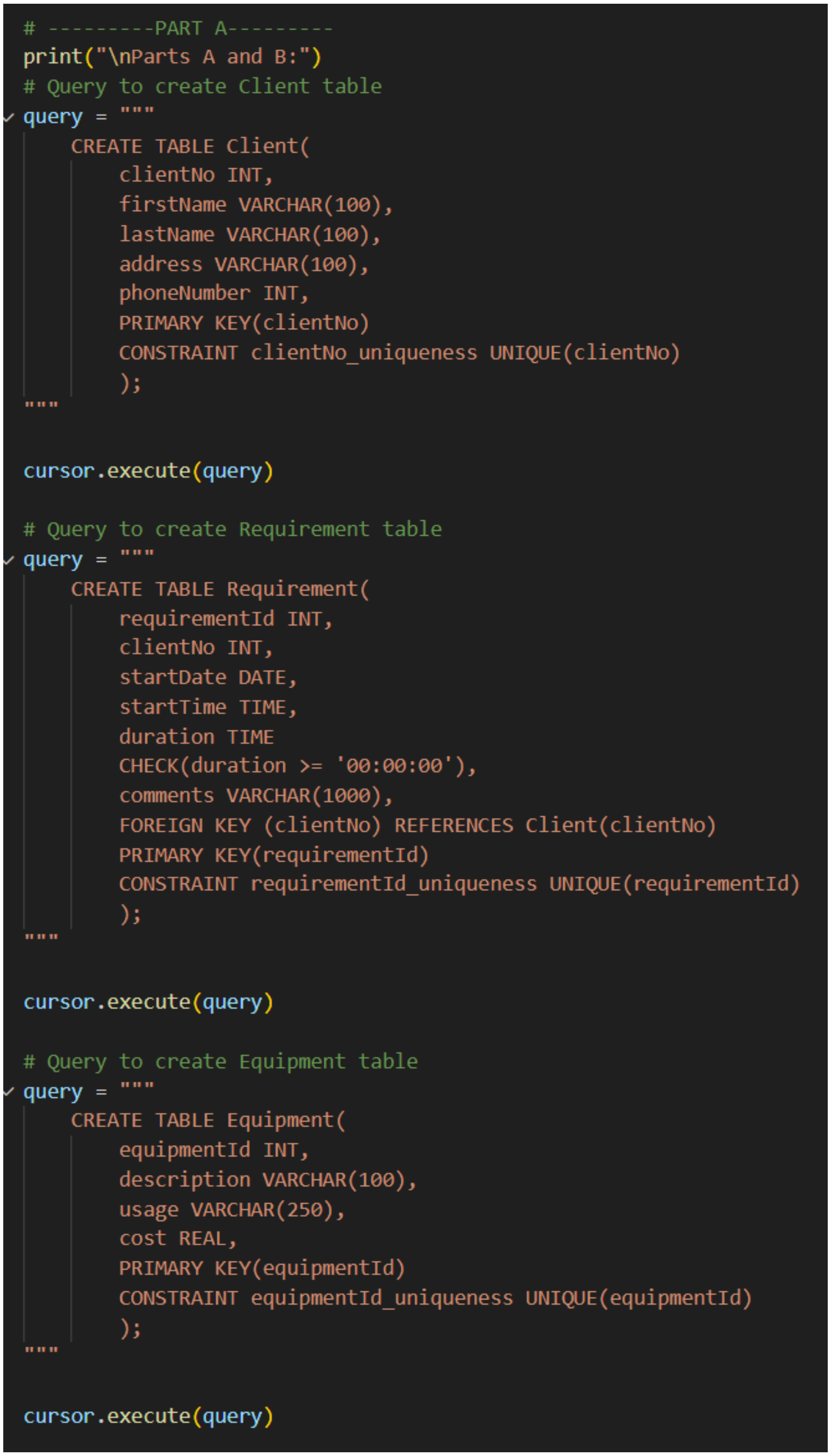
- End time needs to be later then start time
- End date needs to be later then start date
- Duration must be positive

After deriving the relations from the conceptual model, Validate the logical model against normalization to 3NF and against user transactions, and defining all of the integrity constraints. We developed another entity relationship diagram on the logical data model containing foreign keys as attribute:

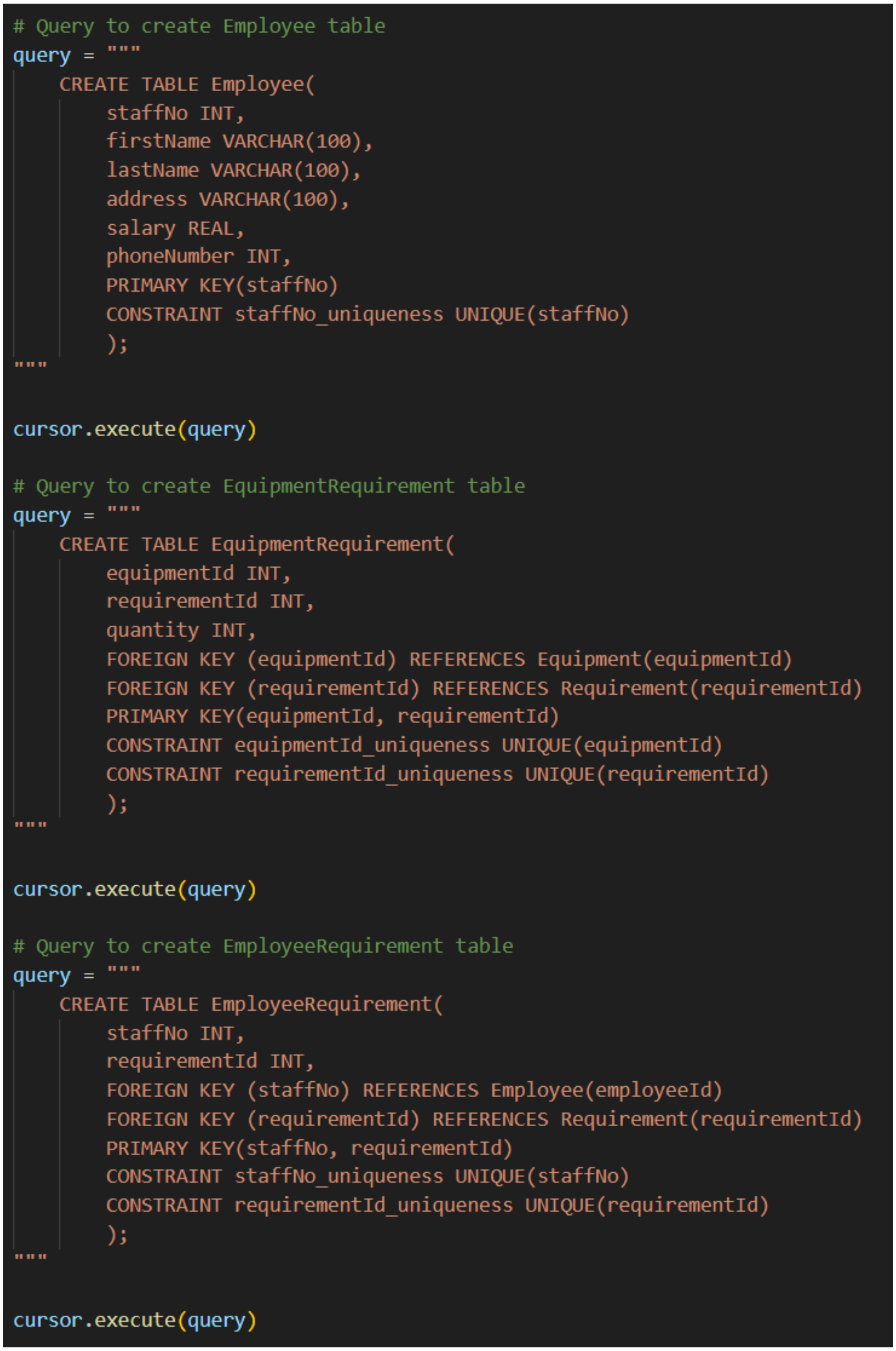


After completely normalizing the case studies requirements into 3NF and defining all the integrity constraints we are able to develop a SQL code to create the entire database schema, reflecting the constraints

Identified earlier in the report. Using the prior knowledge, we were taught in the class using SQL we developed the following code to create the entire database schema reflecting all the constraints:

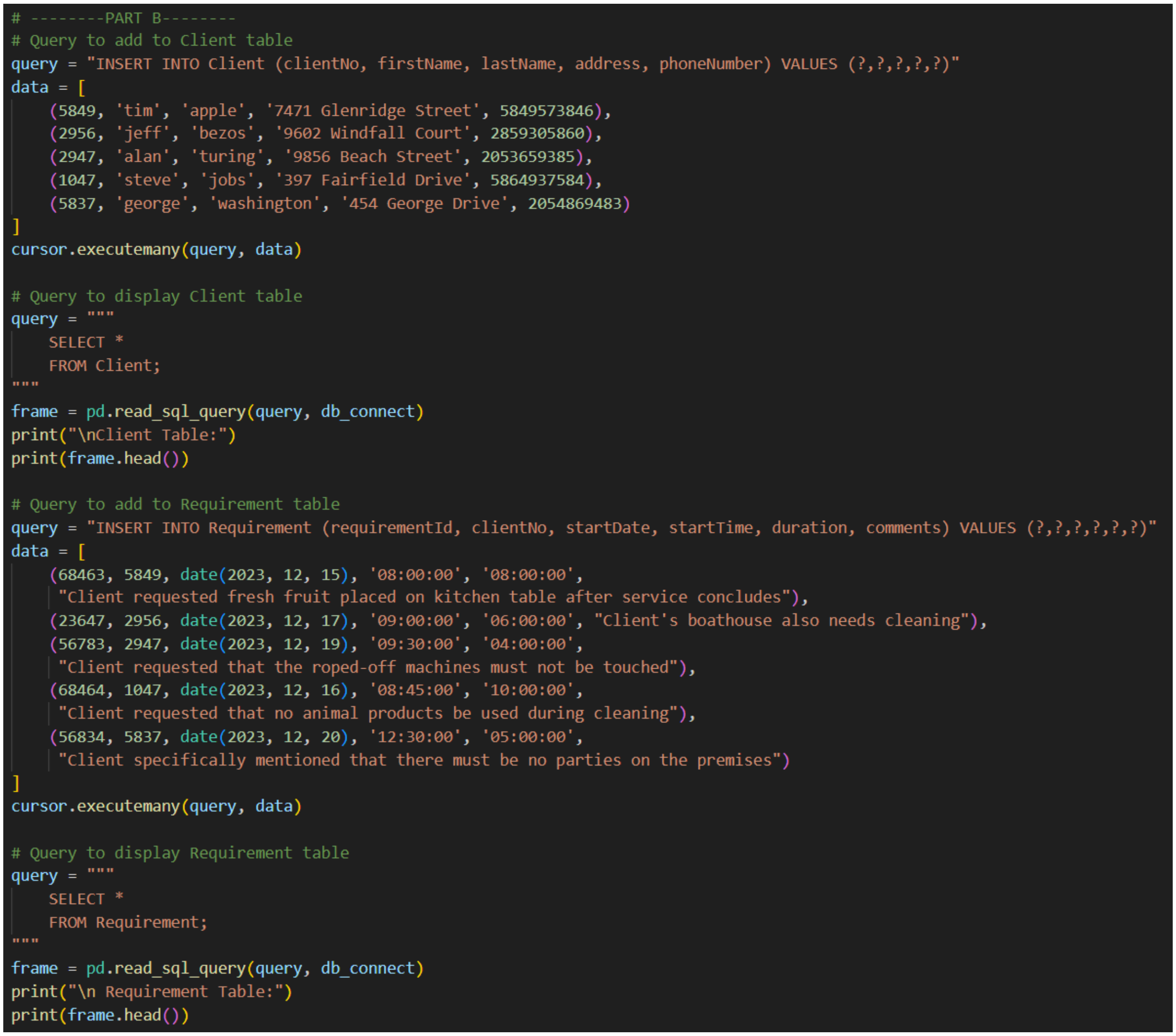


```
# ---------PART A---------
print("\nParts A and B:")
# Query to create Client table
query = """
    CREATE TABLE Client(
        clientNo INT,
        firstName VARCHAR(100),
        lastName VARCHAR(100),
        address VARCHAR(100),
        phoneNumber INT,
        PRIMARY KEY(clientNo)
        CONSTRAINT clientNo_uniqueness UNIQUE(clientNo)
        );
"""

cursor.execute(query)

# Query to create Requirement table
query = """
    CREATE TABLE Requirement(
        requirementId INT,
        clientNo INT,
        startDate DATE,
        startTime TIME,
        duration TIME
        CHECK(duration >= '00:00:00'),
        comments VARCHAR(1000),
        FOREIGN KEY (clientNo) REFERENCES Client(clientNo)
        PRIMARY KEY(requirementId)
        CONSTRAINT requirementId_uniqueness UNIQUE(requirementId)
        );
"""

cursor.execute(query)

# Query to create Equipment table
query = """
    CREATE TABLE Equipment(
        equipmentId INT,
        description VARCHAR(100),
        usage VARCHAR(250),
        cost REAL,
        PRIMARY KEY(equipmentId)
        CONSTRAINT equipmentId_uniqueness UNIQUE(equipmentId)
        );
"""

cursor.execute(query)
```

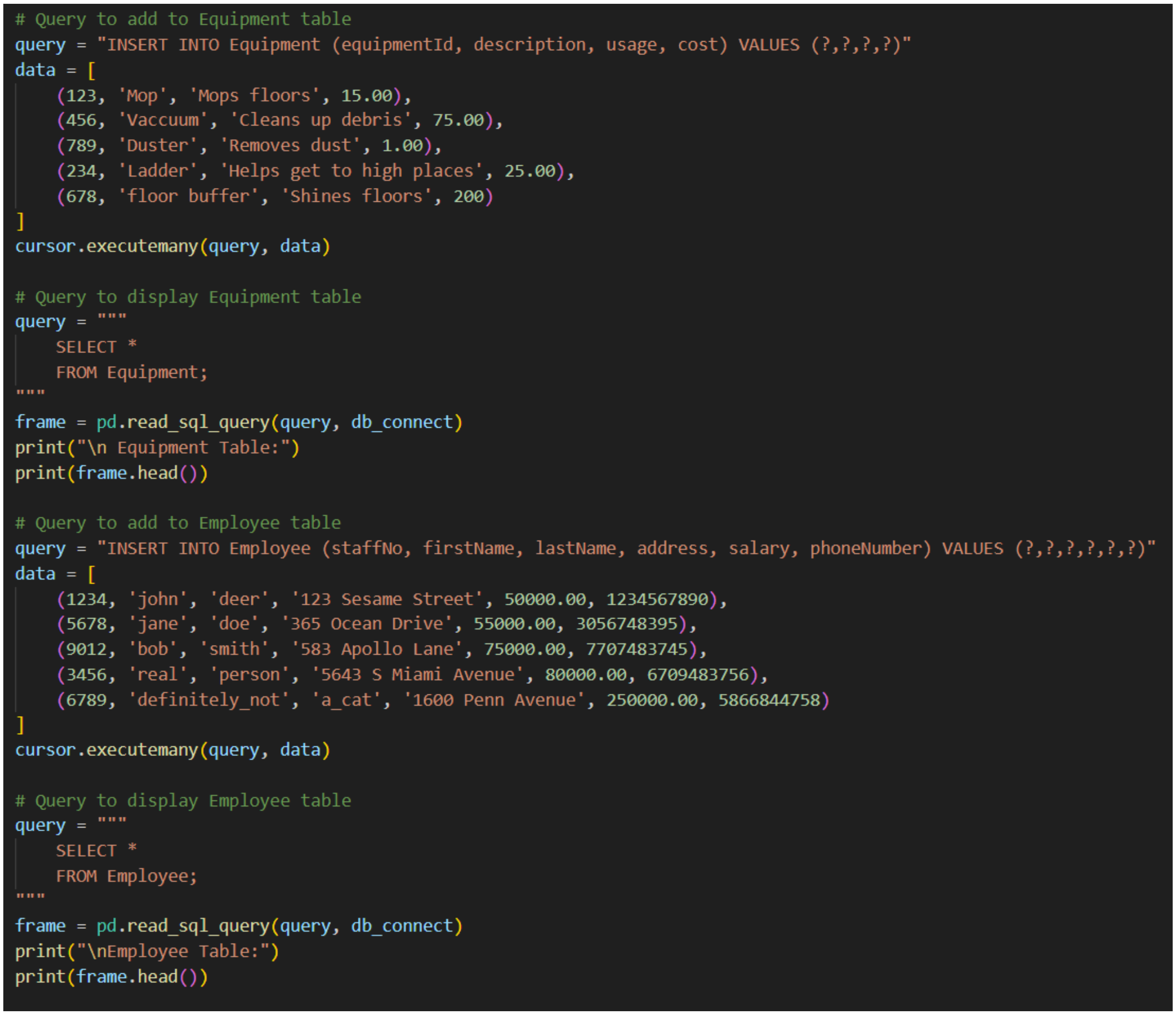


```
# Query to create Employee table
query = """
    CREATE TABLE Employee(
        staffNo INT,
        firstName VARCHAR(100),
        lastName VARCHAR(100),
        address VARCHAR(100),
        salary REAL,
        phoneNumber INT,
        PRIMARY KEY(staffNo)
        CONSTRAINT staffNo_uniqueness UNIQUE(staffNo)
        );
"""

cursor.execute(query)

# Query to create EquipmentRequirement table
query = """
    CREATE TABLE EquipmentRequirement(
        equipmentId INT,
        requirementId INT,
        quantity INT,
        FOREIGN KEY (equipmentId) REFERENCES Equipment(equipmentId)
        FOREIGN KEY (requirementId) REFERENCES Requirement(requirementId)
        PRIMARY KEY(equipmentId, requirementId)
        CONSTRAINT equipmentId_uniqueness UNIQUE(equipmentId)
        CONSTRAINT requirementId_uniqueness UNIQUE(requirementId)
        );
"""

cursor.execute(query)

# Query to create EmployeeRequirement table
query = """
    CREATE TABLE EmployeeRequirement(
        staffNo INT,
        requirementId INT,
        FOREIGN KEY (staffNo) REFERENCES Employee(employeeId)
        FOREIGN KEY (requirementId) REFERENCES Requirement(requirementId)
        PRIMARY KEY(staffNo, requirementId)
        CONSTRAINT staffNo_uniqueness UNIQUE(staffNo)
        CONSTRAINT requirementId_uniqueness UNIQUE(requirementId)
        );
"""

cursor.execute(query)
```

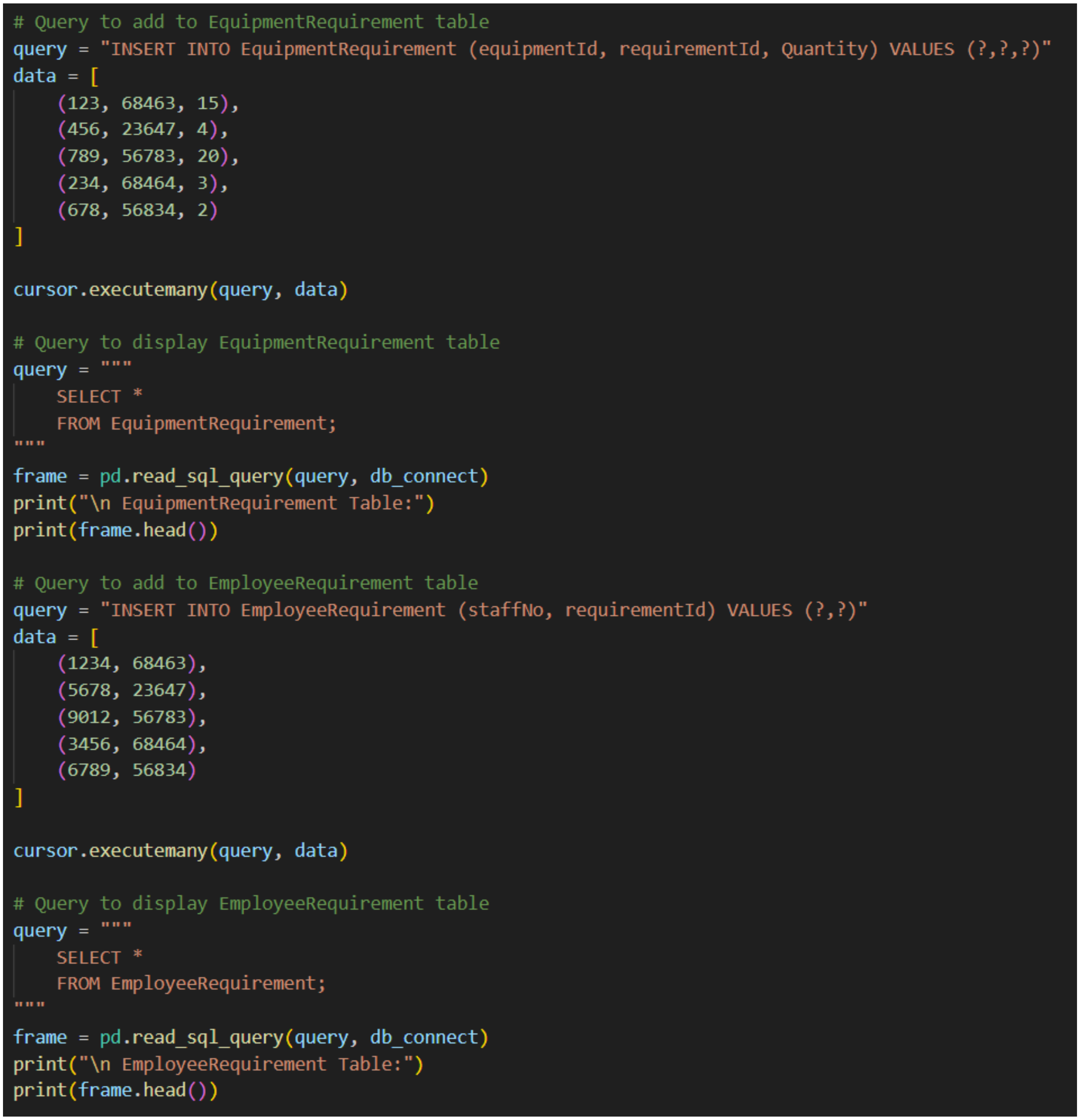
After fully completing our Schema of the Super Maids Cleaning Company we created 5 tuples for each relation we in our database which are: client, requirement, equipment, employee, equipment_requirement, and employee_requirement. The following code shows the tuples we created:



```
# --------PART B--------
# Query to add to Client table
query = "INSERT INTO Client (clientNo, firstName, lastName, address, phoneNumber) VALUES (?,?,?,?,?)"
data = [
    (5849, 'tim', 'apple', '7471 Glenridge Street', 5849573846),
    (2956, 'jeff', 'bezos', '9602 Windfall Court', 2859305860),
    (2947, 'alan', 'turing', '9856 Beach Street', 2053659385),
    (1047, 'steve', 'jobs', '397 Fairfield Drive', 5864937584),
    (5837, 'george', 'washington', '454 George Drive', 2054869483)
]
cursor.executemany(query, data)

# Query to display Client table
query = """
    SELECT *
    FROM Client;
"""
frame = pd.read_sql_query(query, db_connect)
print("\nClient Table:")
print(frame.head())

# Query to add to Requirement table
query = "INSERT INTO Requirement (requirementId, clientNo, startDate, startTime, duration, comments) VALUES (?,?,?,?,?,?)"
data = [
    (68463, 5849, date(2023, 12, 15), '08:00:00', '08:00:00',
     "Client requested fresh fruit placed on kitchen table after service concludes"),
    (23647, 2956, date(2023, 12, 17), '09:00:00', '06:00:00', "Client's boathouse also needs cleaning"),
    (56783, 2947, date(2023, 12, 19), '09:30:00', '04:00:00',
     "Client requested that the roped-off machines must not be touched"),
    (68464, 1047, date(2023, 12, 16), '08:45:00', '10:00:00',
     "Client requested that no animal products be used during cleaning"),
    (56834, 5837, date(2023, 12, 20), '12:30:00', '05:00:00',
     "Client specifically mentioned that there must be no parties on the premises")
]
cursor.executemany(query, data)

# Query to display Requirement table
query = """
    SELECT *
    FROM Requirement;
"""
frame = pd.read_sql_query(query, db_connect)
print("\n Requirement Table:")
print(frame.head())
```

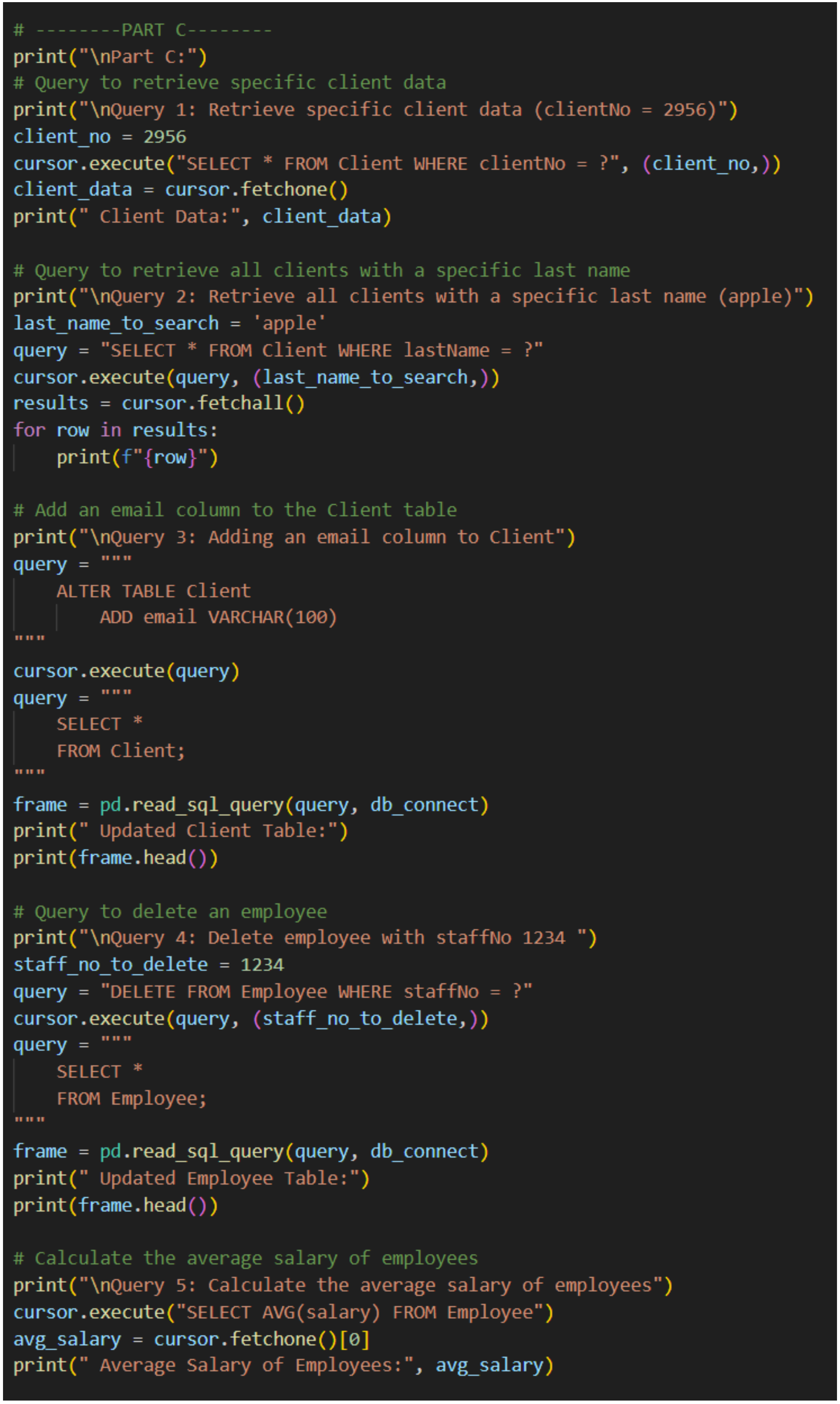


```python
# Query to add to Equipment table
query = "INSERT INTO Equipment (equipmentId, description, usage, cost) VALUES (?,?,?,?)"
data = [
    (123, 'Mop', 'Mops floors', 15.00),
    (456, 'Vaccuum', 'Cleans up debris', 75.00),
    (789, 'Duster', 'Removes dust', 1.00),
    (234, 'Ladder', 'Helps get to high places', 25.00),
    (678, 'floor buffer', 'Shines floors', 200)
]
cursor.executemany(query, data)

# Query to display Equipment table
query = """
    SELECT *
    FROM Equipment;
"""
frame = pd.read_sql_query(query, db_connect)
print("\n Equipment Table:")
print(frame.head())

# Query to add to Employee table
query = "INSERT INTO Employee (staffNo, firstName, lastName, address, salary, phoneNumber) VALUES (?,?,?,?,?,?)"
data = [
    (1234, 'john', 'deer', '123 Sesame Street', 50000.00, 1234567890),
    (5678, 'jane', 'doe', '365 Ocean Drive', 55000.00, 3056748395),
    (9012, 'bob', 'smith', '583 Apollo Lane', 75000.00, 7707483745),
    (3456, 'real', 'person', '5643 S Miami Avenue', 80000.00, 6709483756),
    (6789, 'definitely_not', 'a_cat', '1600 Penn Avenue', 250000.00, 5866844758)
]
cursor.executemany(query, data)

# Query to display Employee table
query = """
    SELECT *
    FROM Employee;
"""
frame = pd.read_sql_query(query, db_connect)
print("\nEmployee Table:")
print(frame.head())
```



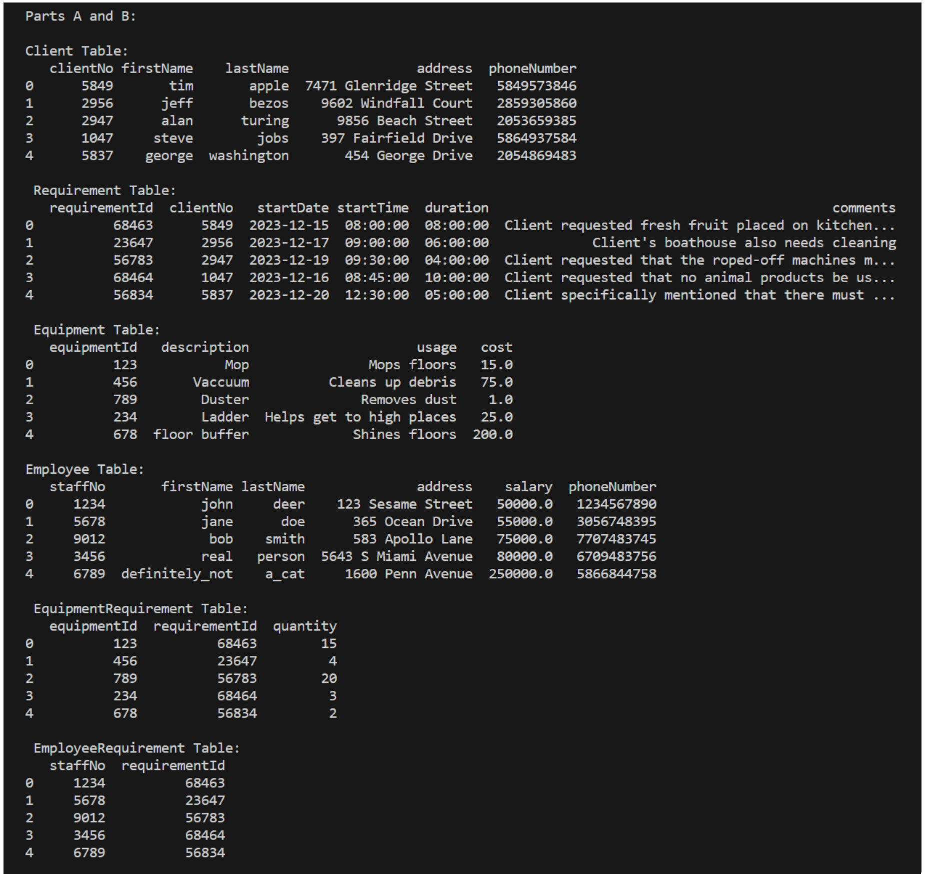
```
# Query to add to EquipmentRequirement table
query = "INSERT INTO EquipmentRequirement (equipmentId, requirementId, Quantity) VALUES (?,?,?)"
data = [
    (123, 68463, 15),
    (456, 23647, 4),
    (789, 56783, 20),
    (234, 68464, 3),
    (678, 56834, 2)
]

cursor.executemany(query, data)

# Query to display EquipmentRequirement table
query = """
    SELECT *
    FROM EquipmentRequirement;
"""
frame = pd.read_sql_query(query, db_connect)
print("\n EquipmentRequirement Table:")
print(frame.head())

# Query to add to EmployeeRequirement table
query = "INSERT INTO EmployeeRequirement (staffNo, requirementId) VALUES (?,?)"
data = [
    (1234, 68463),
    (5678, 23647),
    (9012, 56783),
    (3456, 68464),
    (6789, 56834)
]

cursor.executemany(query, data)

# Query to display EmployeeRequirement table
query = """
    SELECT *
    FROM EmployeeRequirement;
"""
frame = pd.read_sql_query(query, db_connect)
print("\n EmployeeRequirement Table:")
print(frame.head())
```

Finally, after creating 5 tuples for each of our relations in our database we created 5 queries specific to our database using embedded SQL. The following code shows our 5 queries:

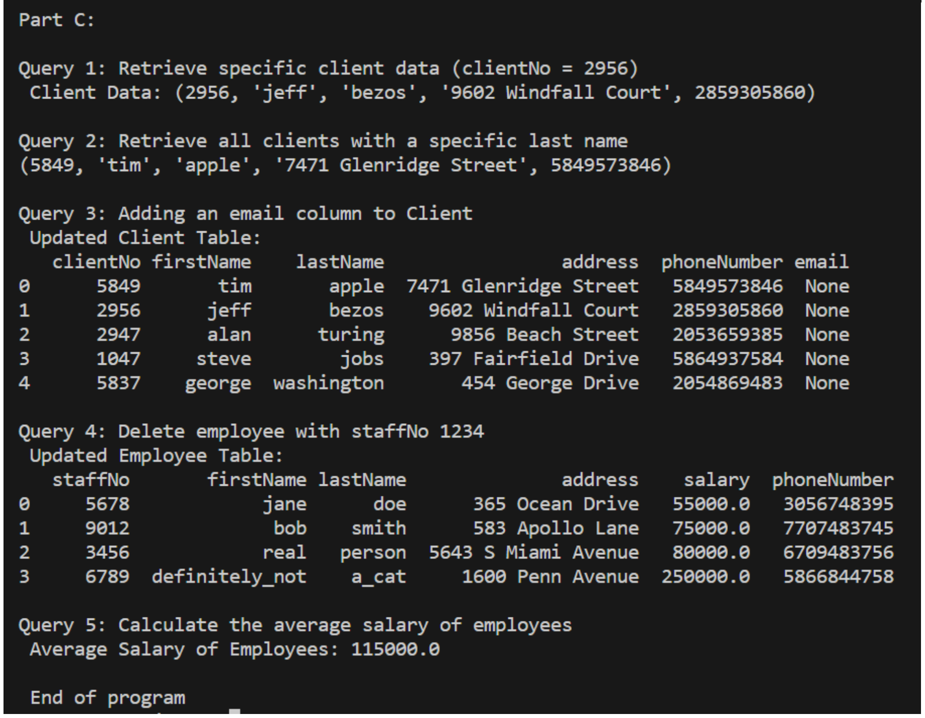


```
# --------PART C--------
print("\nPart C:")
# Query to retrieve specific client data
print("\nQuery 1: Retrieve specific client data (clientNo = 2956)")
client_no = 2956
cursor.execute("SELECT * FROM Client WHERE clientNo = ?", (client_no,))
client_data = cursor.fetchone()
print(" Client Data:", client_data)

# Query to retrieve all clients with a specific last name
print("\nQuery 2: Retrieve all clients with a specific last name (apple)")
last_name_to_search = 'apple'
query = "SELECT * FROM Client WHERE lastName = ?"
cursor.execute(query, (last_name_to_search,))
results = cursor.fetchall()
for row in results:
    print(f"{row}")

# Add an email column to the Client table
print("\nQuery 3: Adding an email column to Client")
query = """
    ALTER TABLE Client
        ADD email VARCHAR(100)
"""
cursor.execute(query)
query = """
    SELECT *
    FROM Client;
"""
frame = pd.read_sql_query(query, db_connect)
print(" Updated Client Table:")
print(frame.head())

# Query to delete an employee
print("\nQuery 4: Delete employee with staffNo 1234 ")
staff_no_to_delete = 1234
query = "DELETE FROM Employee WHERE staffNo = ?"
cursor.execute(query, (staff_no_to_delete,))
query = """
    SELECT *
    FROM Employee;
"""
frame = pd.read_sql_query(query, db_connect)
print(" Updated Employee Table:")
print(frame.head())

# Calculate the average salary of employees
print("\nQuery 5: Calculate the average salary of employees")
cursor.execute("SELECT AVG(salary) FROM Employee")
avg_salary = cursor.fetchone()[0]
print(" Average Salary of Employees:", avg_salary)
```

Finally, after completing all of the following steps in the objective we have a completed embedded SQL program. The following program output



```
Parts A and B:

Client Table:
   clientNo firstName    lastName                address  phoneNumber
0      5849       tim       apple  7471 Glenridge Street   5849573846
1      2956      jeff       bezos    9602 Windfall Court   2859305860
2      2947      alan      turing      9856 Beach Street   2053659385
3      1047     steve        jobs    397 Fairfield Drive   5864937584
4      5837    george  washington       454 George Drive   2054869483

 Requirement Table:
   requirementId  clientNo   startDate startTime  duration                                           comments
0          68463      5849  2023-12-15  08:00:00  08:00:00  Client requested fresh fruit placed on kitchen...
1          23647      2956  2023-12-17  09:00:00  06:00:00             Client's boathouse also needs cleaning
2          56783      2947  2023-12-19  09:30:00  04:00:00  Client requested that the roped-off machines m...
3          68464      1047  2023-12-16  08:45:00  10:00:00  Client requested that no animal products be us...
4          56834      5837  2023-12-20  12:30:00  05:00:00  Client specifically mentioned that there must ...

 Equipment Table:
   equipmentId   description                   usage   cost
0          123           Mop             Mops floors   15.0
1          456       Vaccuum         Cleans up debris   75.0
2          789        Duster            Removes dust    1.0
3          234        Ladder  Helps get to high places   25.0
4          678  floor buffer            Shines floors  200.0

Employee Table:
   staffNo       firstName lastName              address    salary  phoneNumber
0     1234            john     deer    123 Sesame Street   50000.0   1234567890
1     5678            jane      doe      365 Ocean Drive   55000.0   3056748395
2     9012             bob    smith      583 Apollo Lane   75000.0   7707483745
3     3456            real   person  5643 S Miami Avenue   80000.0   6709483756
4     6789  definitely_not    a_cat     1600 Penn Avenue  250000.0   5866844758

 EquipmentRequirement Table:
   equipmentId  requirementId  quantity
0          123          68463        15
1          456          23647         4
2          789          56783        20
3          234          68464         3
4          678          56834         2

 EmployeeRequirement Table:
   staffNo  requirementId
0     1234          68463
1     5678          23647
2     9012          56783
3     3456          68464
4     6789          56834
```



```
Part C:

Query 1: Retrieve specific client data (clientNo = 2956)
 Client Data: (2956, 'jeff', 'bezos', '9602 Windfall Court', 2859305860)

Query 2: Retrieve all clients with a specific last name
(5849, 'tim', 'apple', '7471 Glenridge Street', 5849573846)

Query 3: Adding an email column to Client
 Updated Client Table:
   clientNo firstName    lastName                address  phoneNumber email
0      5849       tim       apple  7471 Glenridge Street   5849573846  None
1      2956      jeff       bezos    9602 Windfall Court   2859305860  None
2      2947      alan      turing      9856 Beach Street   2053659385  None
3      1047     steve        jobs    397 Fairfield Drive   5864937584  None
4      5837    george  washington       454 George Drive   2054869483  None

Query 4: Delete employee with staffNo 1234
 Updated Employee Table:
   staffNo       firstName lastName              address    salary  phoneNumber
0     5678            jane      doe      365 Ocean Drive   55000.0   3056748395
1     9012             bob    smith      583 Apollo Lane   75000.0   7707483745
2     3456            real   person  5643 S Miami Avenue   80000.0   6709483756
3     6789  definitely_not    a_cat     1600 Penn Avenue  250000.0   5866844758

Query 5: Calculate the average salary of employees
 Average Salary of Employees: 115000.0

 End of program
```