# DATABASE MANAGEMENT SYSTEM
# UE18CS252

# PROJECT REPORT
# GYM MANAGEMENT SYSTEM

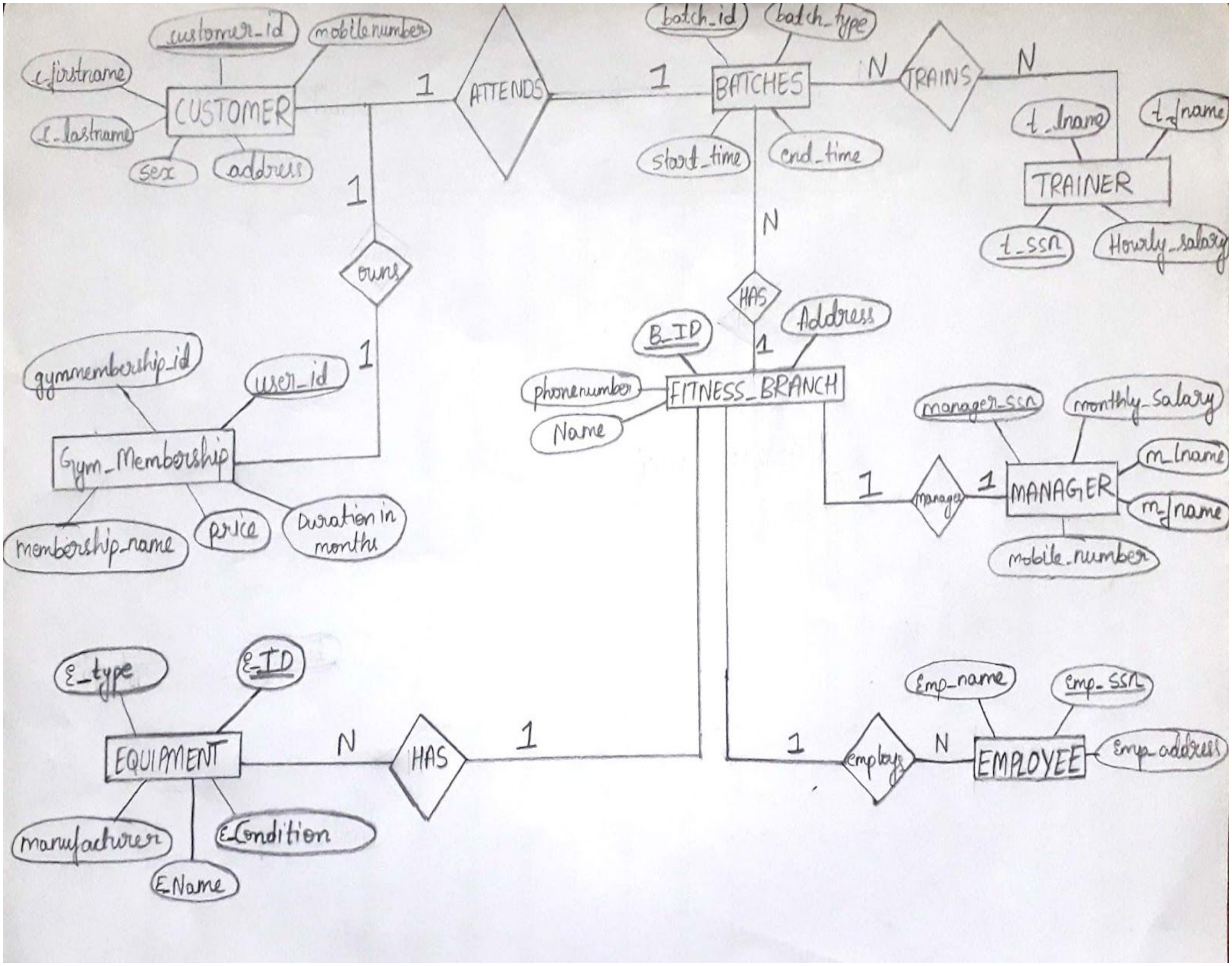
NAME: Harshapriya C.R
SRN: PES1201801774
SECTION: 4B

**TABLE OF CONTENTS:**

## PROJECT DESCRIPTION:

- In today's world, health and fitness plays an important role.
- In this busy schedule, it is very important to take care of our health and stay fit.
- The miniworld chosen is the Gym management system.
- This project considers all the details of a fitness center.
- The database is designed for a multi branch fitness center, which houses exercise equipment for the purpose of physical exercise.

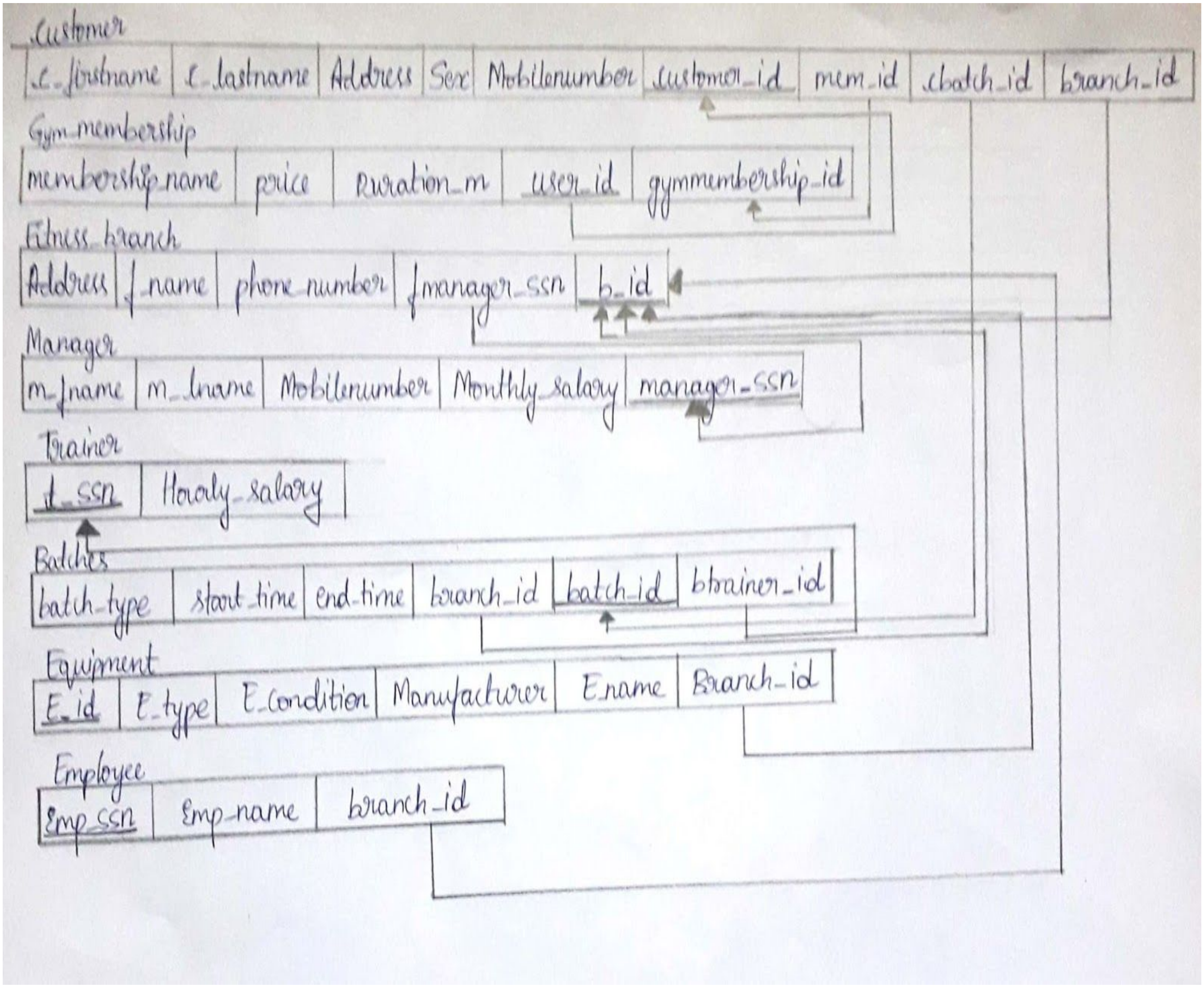
**ER DIAGRAM:**



## ENTITIES AND RELATIONSHIPS IN THE ABOVE ER DIAGRAM:

- Customer: A customer entity represents a particular customer.Its primary key is customer_id to distinguish different customers in the database system.Customer can register at the preferred fitness branch, while registering he/she will mention preferred batch, gym membership, to sign up.
- Gym membership: A customer has to own a gym membership level to use the facilities like cardio, jumba, self defence.user_id is the primary key in this entity.One customer can register for one membership.Customer can upgrade their membership.
- Fitness branch: Customer registers at a fitness branch or center.One fitness branch will have multiple equipments.Branch employs employees like manager,trainer.Fitness branch is managed by Manager.Each fitness branch can have one manager.It will store the details of branch like address,phone number,name and uniquely identified by branch id.
- Employee: Employee entity represents employee in the fitness center. Employees are classified into manager,trainer.Fitness branch employs employees.It stores the details like employee name,and uniquely identified by emp ssn.
- Manager: Manager is the one who manages the entire fitness branch.Manager will manage trainers, and assign work to them.It's the responsibility of the manager to create batches.They will have monthly salary.
- Trainer: Trainer will run the batches and also provides personal training to customers.A Trainer can train multiple batches.
- Batches: This entity has information of batches uniquely identified by batch id, their start time and the end time , batch type.A batch is trained by a trainer.A customer can register to attend only one batch.
- Equipment: This entity represents various equipment in the fitness center.It has details like Type,Manufacturer, and name and the condition of the equipment.

## RELATIONAL SCHEMA:



Customer

| c_firstname | c_lastname | Address | Sex | Mobilenumber | customer_id | mem_id | cbatch_id | branch_id |
|---|---|---|---|---|---|---|---|---|

Gym-membership

| membership_name | price | Duration_m | user_id | gymmembership_id |
|---|---|---|---|---|

Fitness branch

| Address | f_name | phone number | fmanager_ssn | b_id |
|---|---|---|---|---|

Manager

| m_fname | m_lname | Mobilenumber | Monthly_salary | manager_ssn |
|---|---|---|---|---|

Trainer

| t_ssn | Hourly_salary |
|---|---|

Batches

| batch_type | start_time | end_time | branch_id | batch_id | btrainer_id |
|---|---|---|---|---|---|

Equipment

| E_id | E_type | E_condition | Manufacturer | E_name | Branch_id |
|---|---|---|---|---|---|

Employee

| Emp_ssn | Emp_name | branch_id |
|---|---|---|

## FUNCTIONAL DEPENDENCIES , IDENTIFYING KEYS:

Candidate key(CK) is a minimal set of attributes of a relation, which uniquely identifies a tuple in the relation.
If the relation has more than one candidate key,then that set of attributes can be a Key. One of those is made a Primary key(PK).
The functional dependency(FD) is a relationship that exists between two attributes.
It typically exists between the primary key and non-key attribute within a table.

Customer:
**FD**:
Customer_id -> {c_Firstname,c_Lastname, Address, Sex, Mobilenumber, membership_id, cbatch_id, branch_id}
**PK**:customer_id

Gym Membership:
**FD**:User_id -> { gymmembership_id, membership_name, price, duration_month}
**PK**:user_id

Fitness_branch:
**FD**: {b_id, fmanager_s} -> {address, f_name, phone_number}
**CK**:b_id, fmanager_ssn
**PK**: b_id
**SK:** fmanager_ssn

Manager:
**FD:** {manager_ssn} -> {m_fname, m_lname, mobile number, Monthly_salary}
**PK:** manager_ssn

Trainer:
**FD:** t_ssn ->{Hourly_salary}
**PK:** t_ssn

Batches:
**FD**: {batch_id} -> { Batch_type, start_time, end_time, branch_id,trainer_id}
{trainer_id} ->{Batch_type, start_time, end_time, branch_id,batch_id}
**CK**:batch_id,trainer_id
**PK**: batch_id
**SK:** trainer_id

Equipment:
**FD:**E_id -> { e_type, e_condition, Manufacturer, E_name, branch_id}
**PK:** E_id

Employee:
**FD:**Emp_ssn -> { emp_name, branch_id}
**PK:** Emp_ssn

**NORMALIZATION:**
**Proposed by codd.**

1) First Normal form:

First normal form states that domain of an attribute must include only atomic (simple,indivisible) values.It disallows having a tuple of values for an attribute.
In the above relational schema, the domain of all attributes is simple,atomic.
Hence,the database is in first normal form.

2) Second normal form:

A relation R is in Second normal form, if every non-prime attribute in the relation is fully(not partially) functional dependent on the primary key of that relation.

In the table customer, all the non-prime attributes are FD on the primary key customer_id.

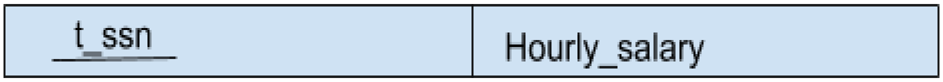
In table fitness_branch, all the non-prime attributes are FD on branch_id.

Hence,the database is in second normal form.

3) Third Normal form:

A relation is in 3NF if it satisfies 2NF and no non-prime attributes are transitively dependent on the primary key.
Hence,the database is in third normal form.

A relation schema R is in BCNF if whenever a FD X->A holds in,then X is a superkey of R.
In table trainer, t_ssn->Hourly_salary where t_ssn is the super key.



| t_ssn | Hourly_salary |
|---|---|

BCNF  is considered a stronger form of 3NF.Therefore Database is also in BCNF.

**TESTING FOR LOSSLESS JOIN PROPERTY:**

Decomposing a relation R into sub-relations R1 and R2,satisfies lossless join property if it holds the following conditions:

1) R1 UNION R2 = R
2) R1 INTERSECTION R2 ≠ Φ
3) R1 INTERSECTION R2   OR   R2 INTERSECTION R1 = CANDIDATE_KEY(at least one)

- In the relation **fitness_branch**,

fitness_branch( b_id, fmanager_ssn  ,address, f_name, phone_number}
Decomposing the relation into
F1(b_id,f_name)
F2(b_id,address,phone_number)
F3(b_id,fmanager_ssn)

->Union of all sub relations F1, F2, F3 has all the attributes of the relation fitness_branch.
->F1 ∩ F2 =b_id
F2 ∩ F3=b_id ,which is not equal to null
->F1 ∩ F2 =b_id
F2 ∩ F3=b_id ,where b_id is the primary key.
Hence the decomposition of the relation fitness_branch satisfies lossless join property.

- In the relation **batches**,

batches(batch_id,trainer_id , Batch_type, start_time, end_time, branch_id}
Decomposing the relation into
B1((batch_id,trainer_id )
B2(batch_id,Batch_type, start_time, end_time, branch_id)

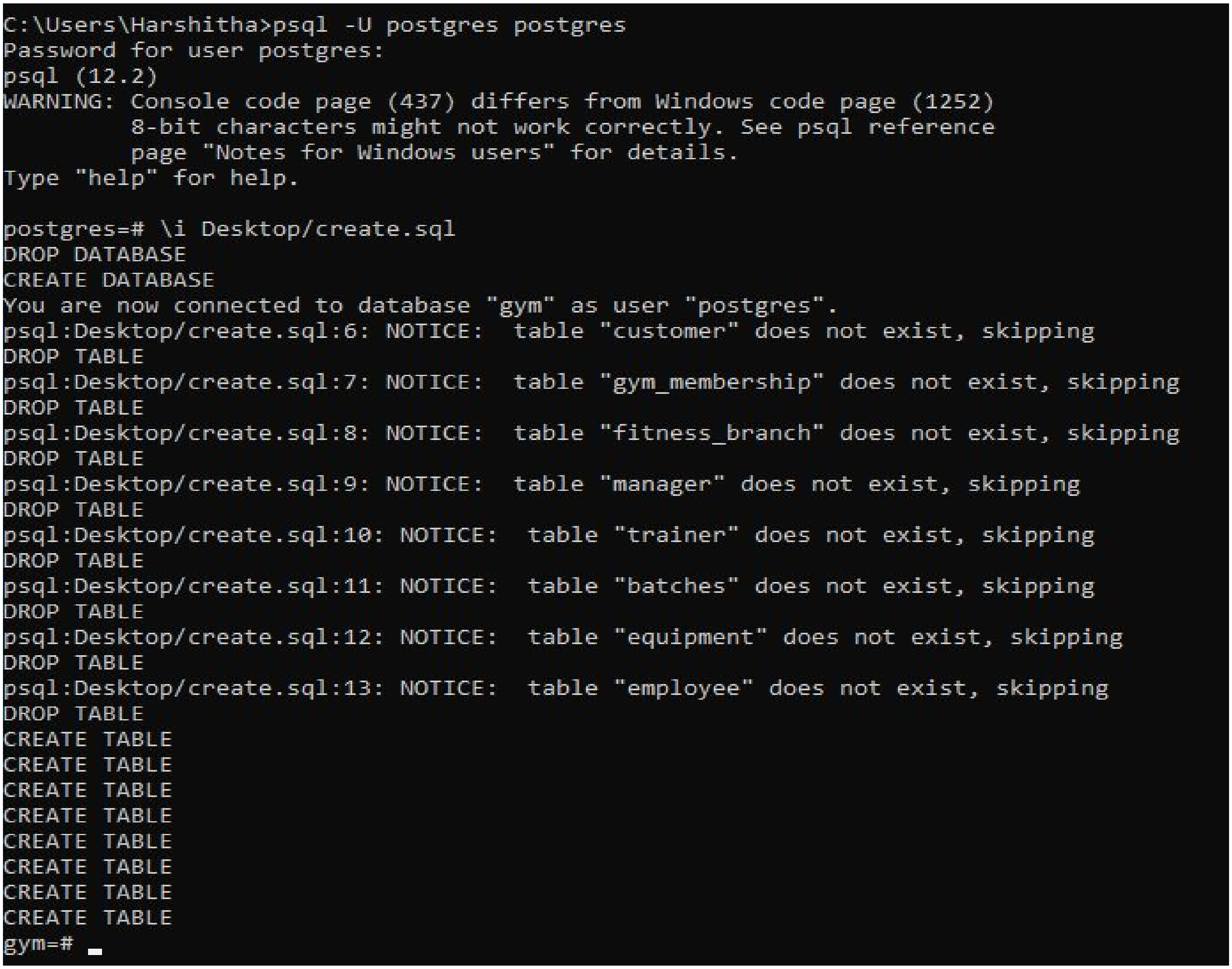
->Union of all sub relations B1, B2has all the attributes of the relation BATCHES.
->B1 ∩ B2 =batch_id
B2 ∩ B1=batch_id ,which is not equal to null
->B1 ∩ B2 =batch_id
B2 ∩ B1=batch_id ,where branch_id is the primary key.

Hence the decomposition of the relation BATCHES satisfies lossless join property.

All the relations in the database can be decomposed into sub relations, and holds this property.

**DDL STATEMENTS FOR CREATION OF TABLES:**

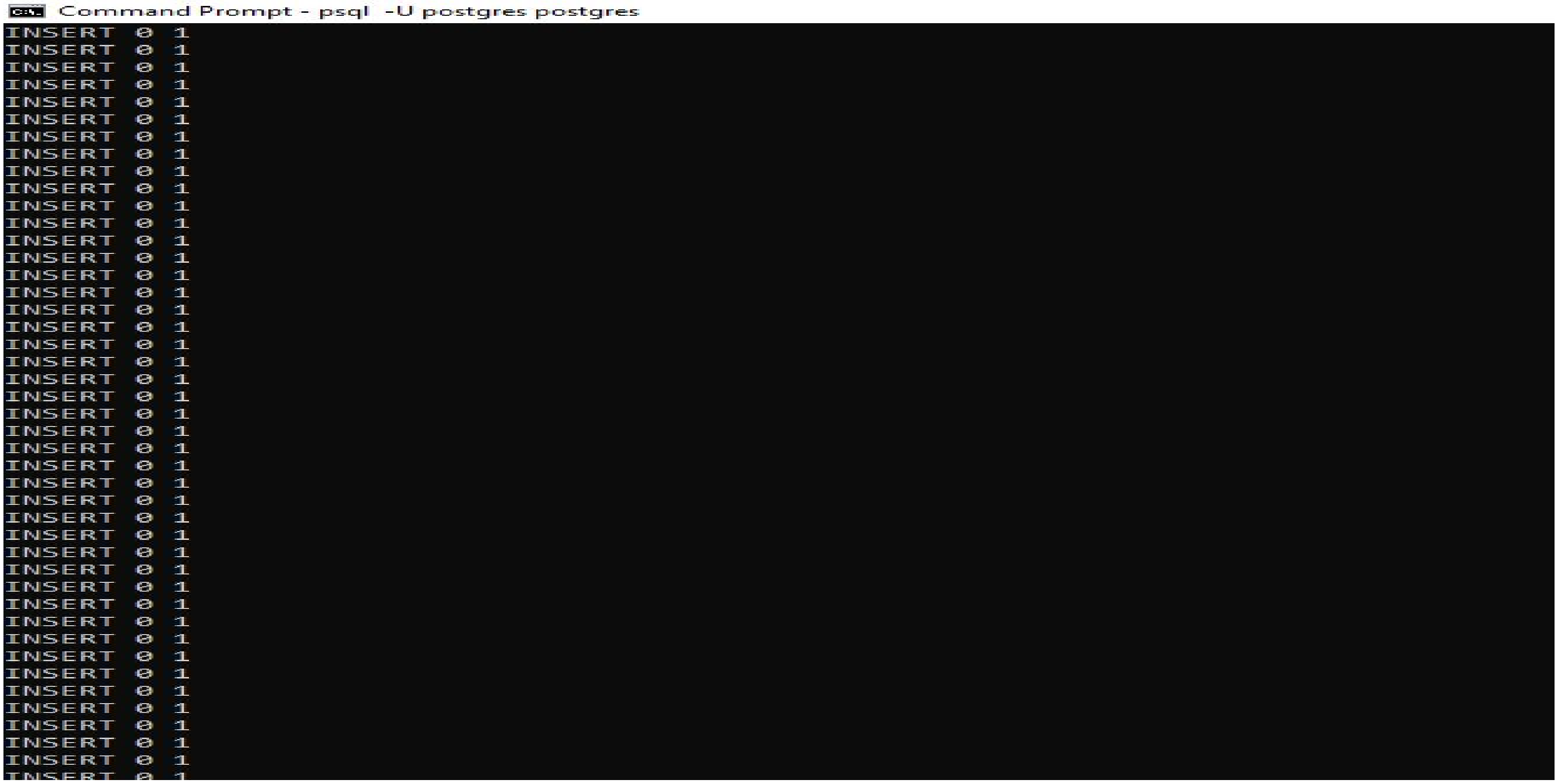
```
Create table customer(
                        Customer_id CHAR(6) NOT NULL,
                        c_Firstname VARCHAR(15) NOT NULL,
                        c_Lastname VARCHAR(15) NOT NULL,
                        Address VARCHAR(20),
                        Sex CHAR,
                        Mobile_Number VARCHAR(50) NOT NULL,
                        membership_id CHAR(4),
                        cbatch_id VARCHAR(4)  ,
                        branch_id VARCHAR(4),
                        UNIQUE(Mobilenumber),
                        PRIMARY KEY(Customer_id)
                );


Create table gym_membership(
                                gymmembership_id CHAR(4) NOT NULL,
                                membership_name VARCHAR(20) NOT NULL,
                                Price DECIMAL(10,2),
                                Duration_month  INT,
                                user_id VARCHAR(6) NOT NULL,
                                PRIMARY KEY(user_id),
                                FOREIGN KEY(user_id) REFERENCES
        customer(customer_id)
                );

create table fitness_branch(
                        b_id VARCHAR(5),
                        address VARCHAR(30) NOT NULL,
                        f_name VARCHAR(25) NOT NULL,
                        Phone_number VARCHAR(50)  NOT NULL ,
                        fmanager_ssn CHAR(6),
                        PRIMARY KEY(b_id)
                        );



create table manager(
                        m_fname VARCHAR(15) NOT NULL,
                        m_lname VARCHAR(15) NOT NULL,
                        Mobile_Number VARCHAR(50),
                        Monthly_salary DECIMAL(10,2),
                        manager_ssn CHAR(6) NOT NULL,
                        UNIQUE(Mobilenumber),
                        PRIMARY KEY(manager_ssn)
                );

create table  trainer(
                        t_ssn CHAR(6) NOT NULL,
                        Hourly_salary DECIMAL(5,2),
                        PRIMARY KEY(t_ssn)
                );

create table batches(
                        batch_id VARCHAR(4) NOT NULL,
                        Batch_type VARCHAR(50) NOT NULL,
                        start_time VARCHAR(5) NOT NULL,
                        end_time VARCHAR(5),
                        branch_id  VARCHAR(5) NOT NULL,
                        btrainer_id VARCHAR(6),
                        PRIMARY KEY(batch_id)
                        );




Create table equipment(
                                E_id VARCHAR(5) NOT NULL,
                                E_type  VARCHAR(20) NOT NULL,
                                E_condition VARCHAR(20),
                                Manufacturer  VARCHAR(20),
                                E_name  VARCHAR(15) NOT NULL,
                                Branch_id  VARCHAR(5) NOT NULL,
                                PRIMARY KEY(E_id)
);

create table employee(
                                emp_ssn CHAR(6) NOT NULL,
                                emp_fname VARCHAR(20),
                                emp_lname VARCHAR(20),
                                branch_id VARCHAR(5) NOT NULL,
                                PRIMARY KEY(emp_ssn),
                                FOREIGN KEY(branch_id) references  fitness_branch(b_id)
                        );
```



```
C:\Users\Harshitha>psql -U postgres postgres
Password for user postgres:
psql (12.2)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# \i Desktop/create.sql
DROP DATABASE
CREATE DATABASE
You are now connected to database "gym" as user "postgres".
psql:Desktop/create.sql:6: NOTICE:  table "customer" does not exist, skipping
DROP TABLE
psql:Desktop/create.sql:7: NOTICE:  table "gym_membership" does not exist, skipping
DROP TABLE
psql:Desktop/create.sql:8: NOTICE:  table "fitness_branch" does not exist, skipping
DROP TABLE
psql:Desktop/create.sql:9: NOTICE:  table "manager" does not exist, skipping
DROP TABLE
psql:Desktop/create.sql:10: NOTICE:  table "trainer" does not exist, skipping
DROP TABLE
psql:Desktop/create.sql:11: NOTICE:  table "batches" does not exist, skipping
DROP TABLE
psql:Desktop/create.sql:12: NOTICE:  table "equipment" does not exist, skipping
DROP TABLE
psql:Desktop/create.sql:13: NOTICE:  table "employee" does not exist, skipping
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
gym=#
```

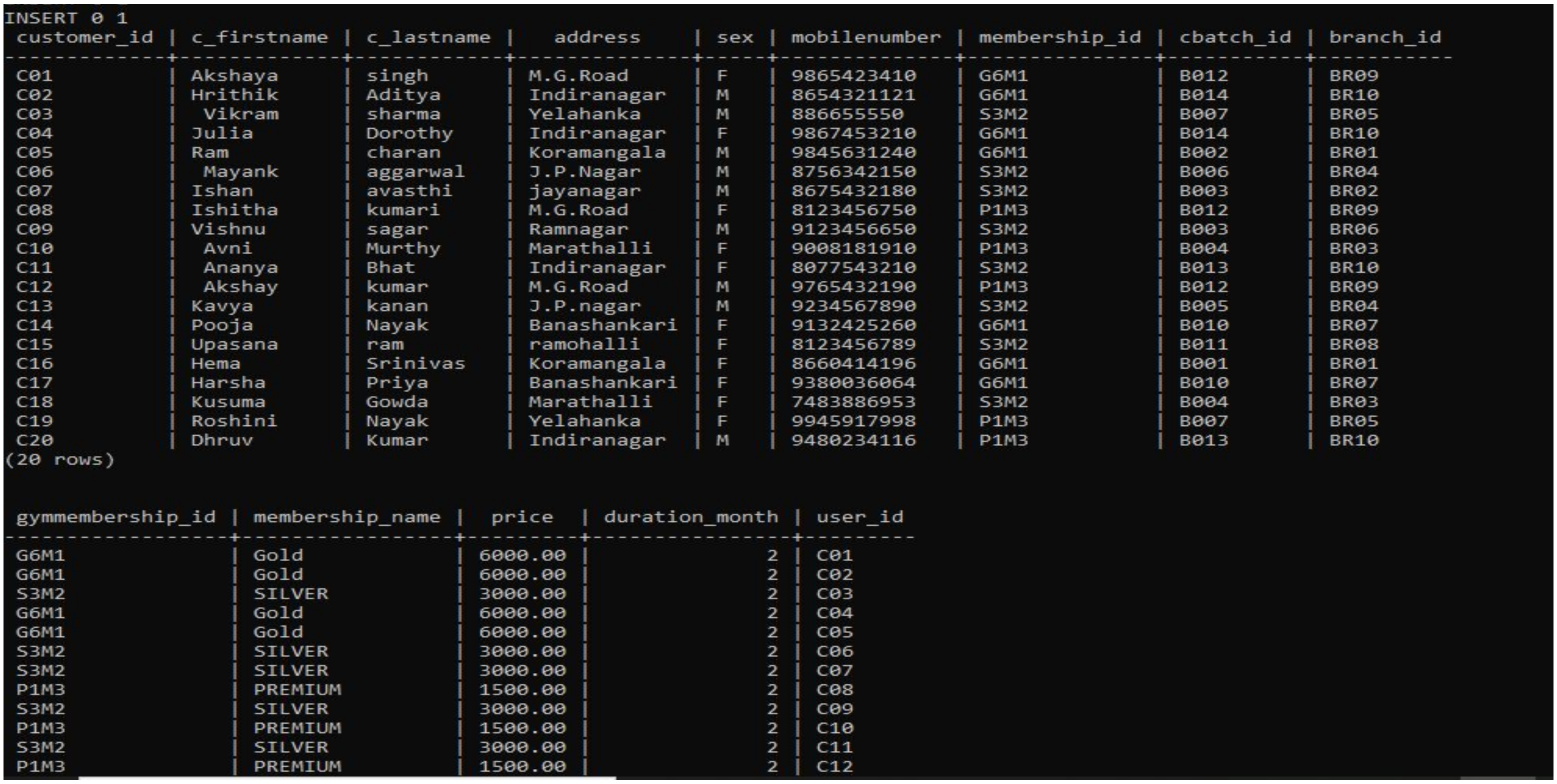
**DML STATEMENTS:**

**AFTER INSERTING VALUES INTO TABLES:**



Command Prompt - psql  -U postgres postgres

```
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
```



```
INSERT 0 1
 customer_id | c_firstname | c_lastname |   address    | sex | mobilenumber | membership_id | cbatch_id | branch_id
-------------+-------------+------------+--------------+-----+--------------+---------------+-----------+-----------
 C01         | Akshaya     | singh      | M.G.Road     | F   | 9865423410   | G6M1          | B012      | BR09
 C02         | Hrithik     | Aditya     | Indiranagar  | M   | 8654321121   | G6M1          | B014      | BR10
 C03         |  Vikram     | sharma     | Yelahanka    | M   | 886655550    | S3M2          | B007      | BR05
 C04         | Julia       | Dorothy    | Indiranagar  | F   | 9867453210   | G6M1          | B014      | BR10
 C05         | Ram         | charan     | Koramangala  | M   | 9845631240   | G6M1          | B002      | BR01
 C06         |  Mayank     | aggarwal   | J.P.Nagar    | M   | 8756342150   | S3M2          | B006      | BR04
 C07         | Ishan       | avasthi    | jayanagar    | M   | 8675432180   | S3M2          | B003      | BR02
 C08         | Ishitha     | kumari     | M.G.Road     | F   | 8123456750   | P1M3          | B012      | BR09
 C09         | Vishnu      | sagar      | Ramnagar     | M   | 9123456650   | S3M2          | B003      | BR06
 C10         |  Avni       | Murthy     | Marathalli   | F   | 9008181910   | P1M3          | B004      | BR03
 C11         |  Ananya     | Bhat       | Indiranagar  | F   | 8077543210   | S3M2          | B013      | BR10
 C12         |  Akshay     | kumar      | M.G.Road     | M   | 9765432190   | P1M3          | B012      | BR09
 C13         | Kavya       | kanan      | J.P.nagar    | M   | 9234567890   | S3M2          | B005      | BR04
 C14         | Pooja       | Nayak      | Banashankari | F   | 9132425260   | G6M1          | B010      | BR07
 C15         | Upasana     | ram        | ramohalli    | F   | 8123456789   | S3M2          | B011      | BR08
 C16         | Hema        | Srinivas   | Koramangala  | F   | 8660414196   | G6M1          | B001      | BR01
 C17         | Harsha      | Priya      | Banashankari | F   | 9380036064   | G6M1          | B010      | BR07
 C18         | Kusuma      | Gowda      | Marathalli   | F   | 7483886953   | S3M2          | B004      | BR03
 C19         | Roshini     | Nayak      | Yelahanka    | F   | 9945917998   | P1M3          | B007      | BR05
 C20         | Dhruv       | Kumar      | Indiranagar  | M   | 9480234116   | P1M3          | B013      | BR10
(20 rows)


 gymmembership_id | membership_name |  price  | duration_month | user_id
------------------+-----------------+---------+----------------+---------
 G6M1             | Gold            | 6000.00 |              2 | C01
 G6M1             | Gold            | 6000.00 |              2 | C02
 S3M2             | SILVER          | 3000.00 |              2 | C03
 G6M1             | Gold            | 6000.00 |              2 | C04
 G6M1             | Gold            | 6000.00 |              2 | C05
 S3M2             | SILVER          | 3000.00 |              2 | C06
 S3M2             | SILVER          | 3000.00 |              2 | C07
 P1M3             | PREMIUM         | 1500.00 |              2 | C08
 S3M2             | SILVER          | 3000.00 |              2 | C09
 P1M3             | PREMIUM         | 1500.00 |              2 | C10
 S3M2             | SILVER          | 3000.00 |              2 | C11
 P1M3             | PREMIUM         | 1500.00 |              2 | C12
```

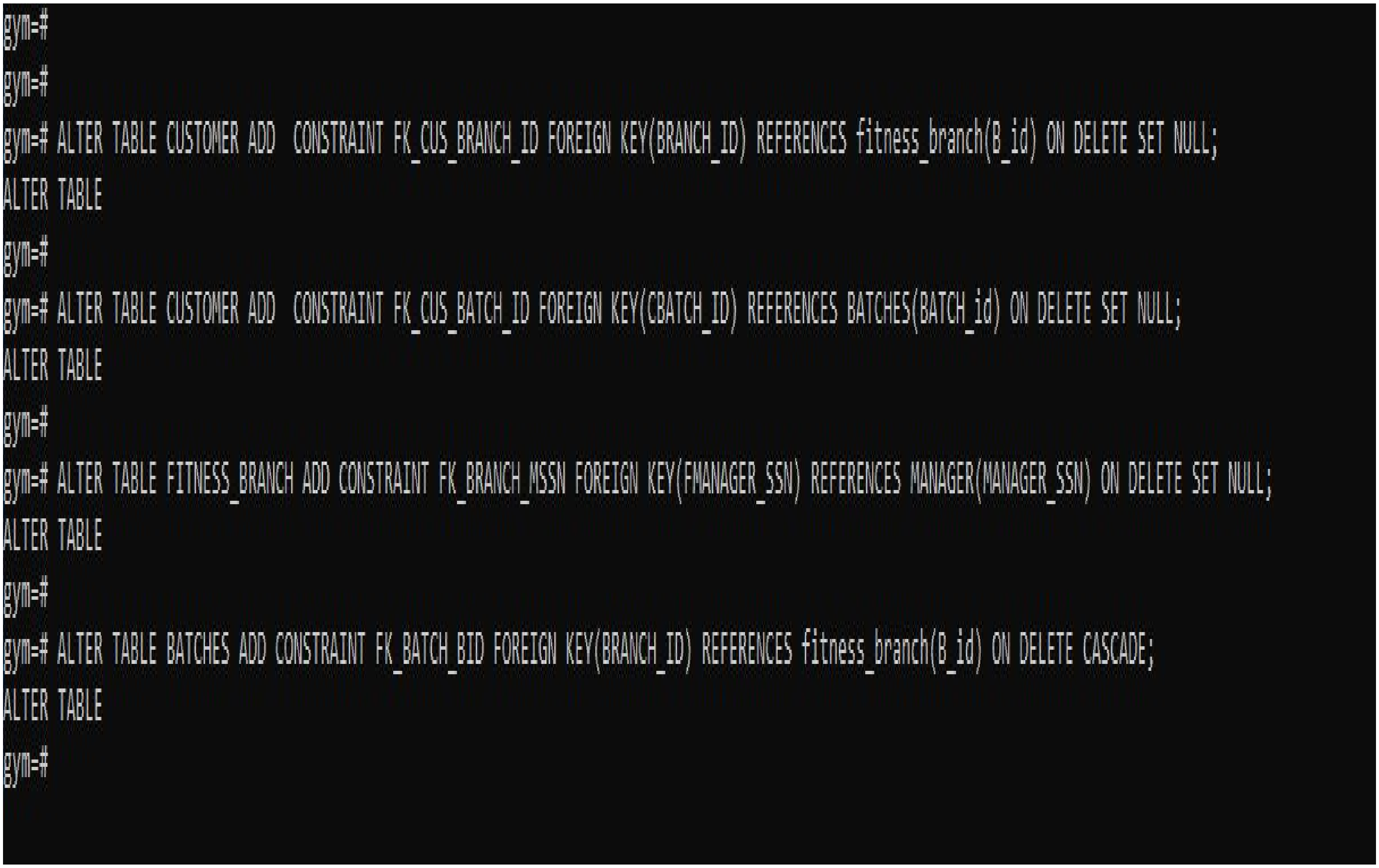
**DDL:**
**Creating check integrity constraints:**

```
ALTER TABLE MANAGER
ADD CONSTRAINT check_salary
CHECK (monthly_salary > 0);

ALTER TABLE TRAINER
ADD CONSTRAINT check_salary
CHECK (hourly_salary > 0);

ALTER TABLE TRAINER
ADD CONSTRAINT check_trainer_ssn
CHECK (t_ssn BETWEEN '400000' and '500000');
```

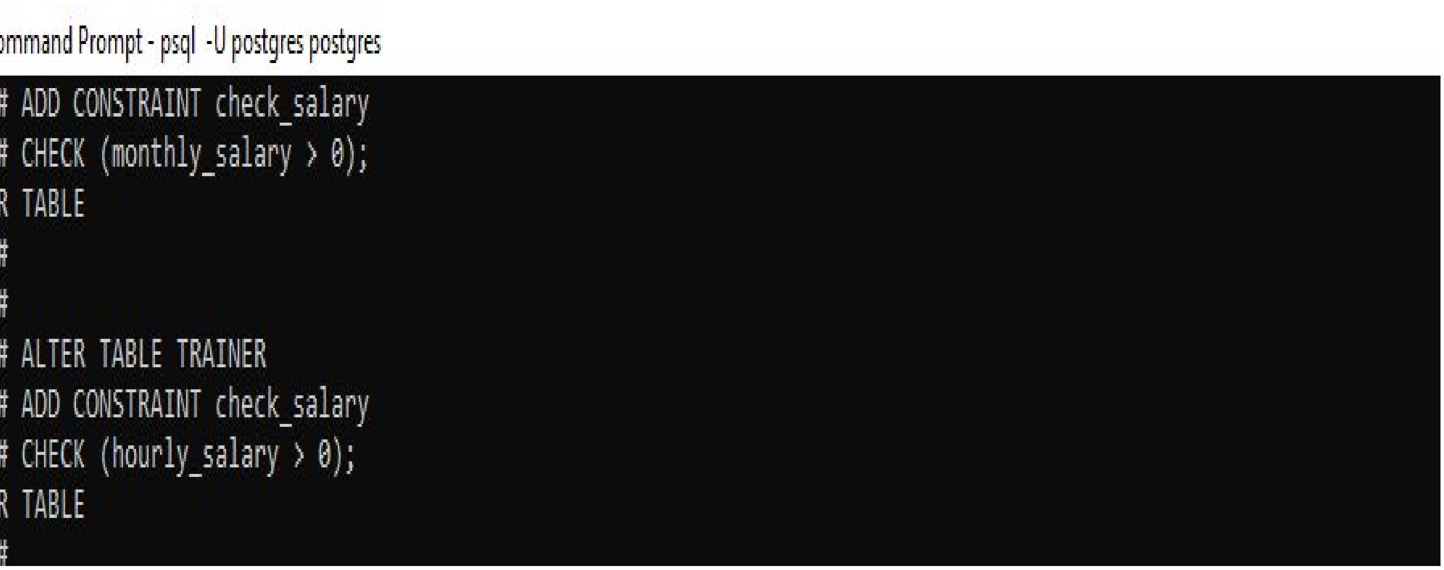
**Creating referential integrity constraints:**

- ALTER TABLE CUSTOMER ADD  CONSTRAINT FK_CUS_BRANCH_ID FOREIGN KEY(BRANCH_ID) REFERENCES fitness_branch(B_id) ON DELETE SET NULL;
  If a particular branch is deleted,then branch_id is set to NULL.

- ALTER TABLE CUSTOMER ADD  CONSTRAINT FK_CUS_BATCH_ID FOREIGN KEY(CBATCH_ID) REFERENCES BATCHES(BATCH_id) ON DELETE SET NULL;
- ALTER TABLE FITNESS_BRANCH ADD CONSTRAINT FK_BRANCH_MSSN FOREIGN KEY(FMANAGER_SSN) REFERENCES MANAGER(MANAGER_SSN) ON DELETE SET NULL;
- ALTER TABLE BATCHES ADD CONSTRAINT FK_BATCH_BID FOREIGN KEY(BRANCH_ID) REFERENCES fitness_branch(B_id) ON DELETE CASCADE;
  If a particular Fitness branch is deleted then all of its associated batches are also deleted.
- ALTER TABLE MANAGER ADD CONSTRAINT FK_MANAGER_SSN FOREIGN KEY(MANAGER_SSN) REFERENCES EMPLOYEE(EMP_SSN) ON DELETE CASCADE;



```
gym=#
gym=#
gym=# ALTER TABLE CUSTOMER ADD  CONSTRAINT FK_CUS_BRANCH_ID FOREIGN KEY(BRANCH_ID) REFERENCES fitness_branch(B_id) ON DELETE SET NULL;
ALTER TABLE
gym=#
gym=# ALTER TABLE CUSTOMER ADD  CONSTRAINT FK_CUS_BATCH_ID FOREIGN KEY(CBATCH_ID) REFERENCES BATCHES(BATCH_id) ON DELETE SET NULL;
ALTER TABLE
gym=#
gym=# ALTER TABLE FITNESS_BRANCH ADD CONSTRAINT FK_BRANCH_MSSN FOREIGN KEY(FMANAGER_SSN) REFERENCES MANAGER(MANAGER_SSN) ON DELETE SET NULL;
ALTER TABLE
gym=#
gym=# ALTER TABLE BATCHES ADD CONSTRAINT FK_BATCH_BID FOREIGN KEY(BRANCH_ID) REFERENCES fitness_branch(B_id) ON DELETE CASCADE;
ALTER TABLE
gym=#
```



```
# ADD CONSTRAINT check_salary
# CHECK (monthly_salary > 0);
R TABLE
#
#
# ALTER TABLE TRAINER
# ADD CONSTRAINT check_salary
# CHECK (hourly_salary > 0);
R TABLE
#
```

**TRIGGER:**

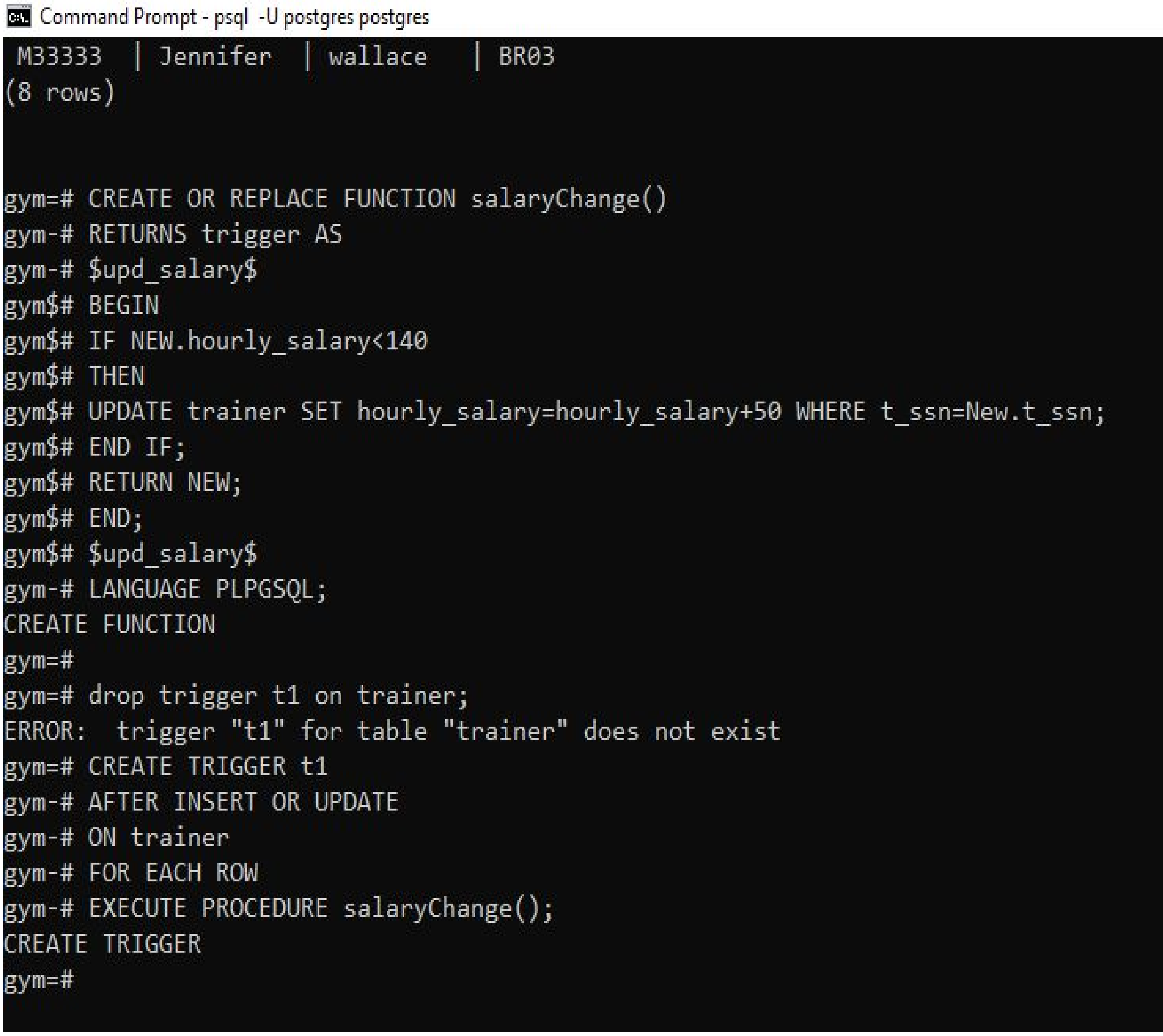
1. If the entered value of hourly_salary in the trainer table is less than 140, 50 is added to the entered value and then updated.

```
CREATE OR REPLACE FUNCTION salaryChange()
RETURNS trigger AS
$upd_salary$
BEGIN
IF NEW.hourly_salary<140
THEN
UPDATE trainer SET hourly_salary=hourly_salary+50 WHERE t_ssn=New.t_ssn;
END IF;
RETURN NEW;
END;
$upd_salary$
LANGUAGE PLPGSQL;
```

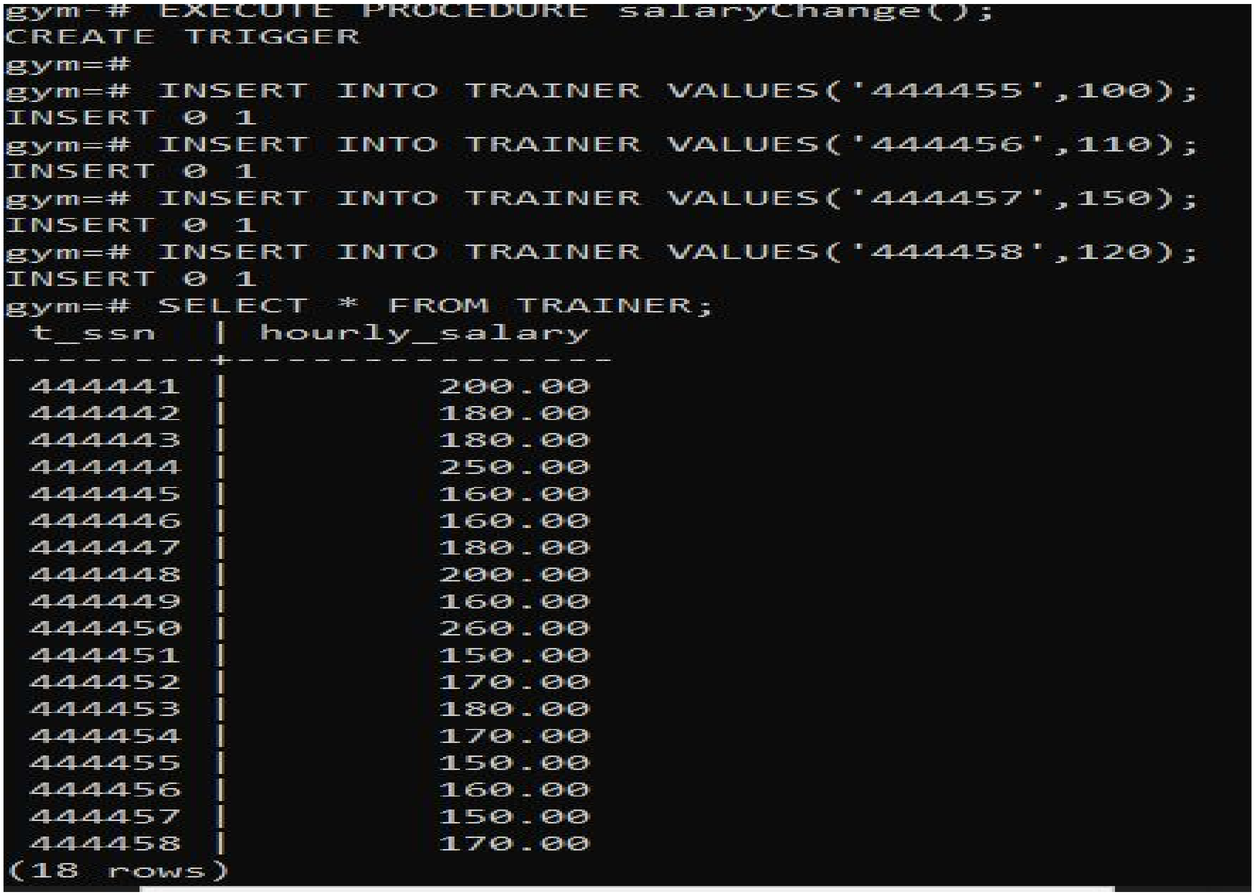
Execute the function salarychange() whenever a row of the table trainer is about to be updated or inserted

```
drop trigger t1 on trainer;
CREATE TRIGGER t1
```

```
AFTER INSERT OR UPDATE
ON trainer
FOR EACH ROW
EXECUTE PROCEDURE salaryChange();
```



```
 M33333  | Jennifer  | wallace   | BR03
(8 rows)


gym=# CREATE OR REPLACE FUNCTION salaryChange()
gym-# RETURNS trigger AS
gym-# $upd_salary$
gym$# BEGIN
gym$# IF NEW.hourly_salary<140
gym$# THEN
gym$# UPDATE trainer SET hourly_salary=hourly_salary+50 WHERE t_ssn=New.t_ssn;
gym$# END IF;
gym$# RETURN NEW;
gym$# END;
gym$# $upd_salary$
gym-# LANGUAGE PLPGSQL;
CREATE FUNCTION
gym=#
gym=# drop trigger t1 on trainer;
ERROR:  trigger "t1" for table "trainer" does not exist
gym=# CREATE TRIGGER t1
gym-# AFTER INSERT OR UPDATE
gym-# ON trainer
gym-# FOR EACH ROW
gym-# EXECUTE PROCEDURE salaryChange();
CREATE TRIGGER
gym=#
```



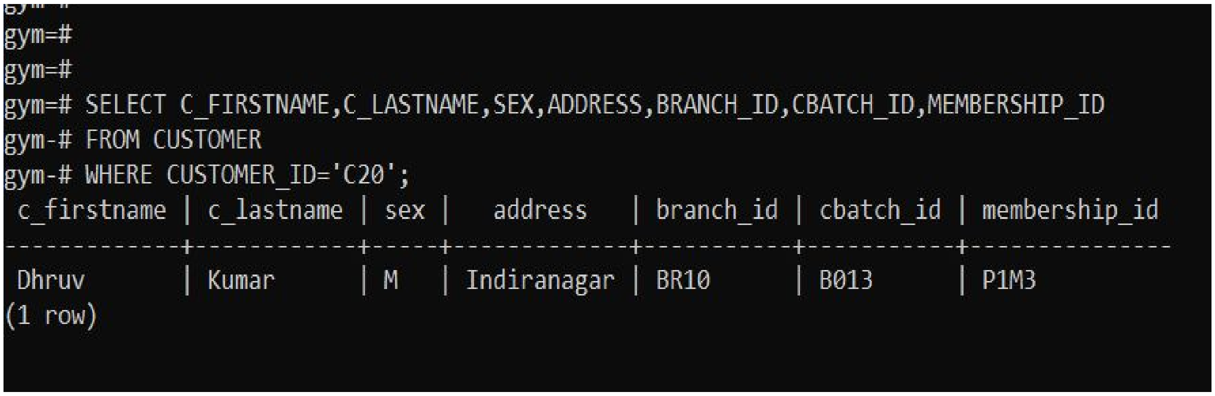
```
gym-# EXECUTE PROCEDURE salaryChange();
CREATE TRIGGER
gym=#
gym=# INSERT INTO TRAINER VALUES('444455',100);
INSERT 0 1
gym=# INSERT INTO TRAINER VALUES('444456',110);
INSERT 0 1
gym=# INSERT INTO TRAINER VALUES('444457',150);
INSERT 0 1
gym=# INSERT INTO TRAINER VALUES('444458',120);
INSERT 0 1
gym=# SELECT * FROM TRAINER;
 t_ssn  | hourly_salary
--------+---------------
 444441 |        200.00
 444442 |        180.00
 444443 |        180.00
 444444 |        250.00
 444445 |        160.00
 444446 |        160.00
 444447 |        180.00
 444448 |        200.00
 444449 |        160.00
 444450 |        260.00
 444451 |        150.00
 444452 |        170.00
 444453 |        180.00
 444454 |        170.00
 444455 |        150.00
 444456 |        160.00
 444457 |        150.00
 444458 |        170.00
(18 rows)
```

**QUERIES:**

**SIMPLE QUERIES:**

- Retrieve firstname,lastname,sex,address,branch,batch,membership details in customer table where customer id is C20

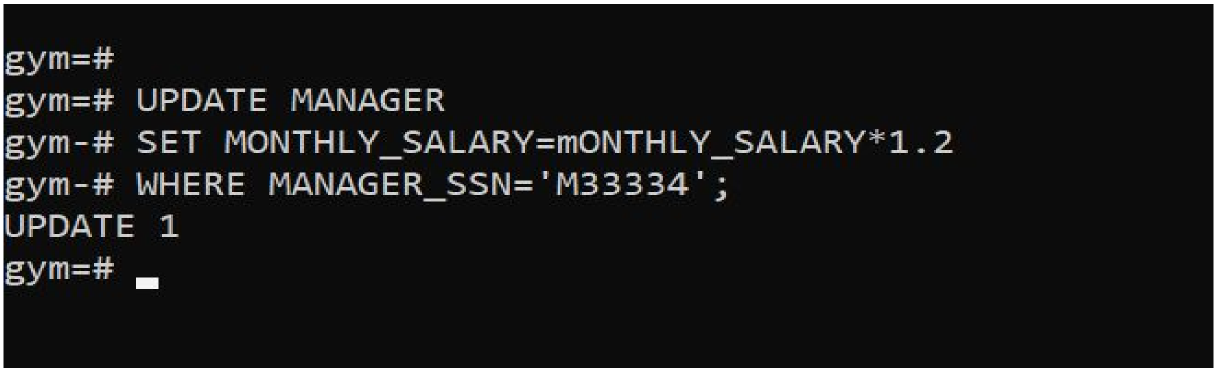
```
SELECT
C_FIRSTNAME,C_LASTNAME,SEX,ADDRESS,BRANCH_ID,CBATCH_ID,MEMBERSHIP_ID
FROM CUSTOMER
WHERE CUSTOMER_ID='C20';
```



```
gym=#
gym=#
gym=# SELECT C_FIRSTNAME,C_LASTNAME,SEX,ADDRESS,BRANCH_ID,CBATCH_ID,MEMBERSHIP_ID
gym-# FROM CUSTOMER
gym-# WHERE CUSTOMER_ID='C20';
 c_firstname | c_lastname | sex |   address   | branch_id | cbatch_id | membership_id
-------------+------------+-----+-------------+-----------+-----------+---------------
 Dhruv       | Kumar      | M   | Indiranagar | BR10      | B013      | P1M3
(1 row)
```

- Increase the salary by 20% for manager with id 'M33334'

UPDATE MANAGER
SET MONTHLY_SALARY=mONTHLY_SALARY*1.2
WHERE MANAGER_SSN='M33334';

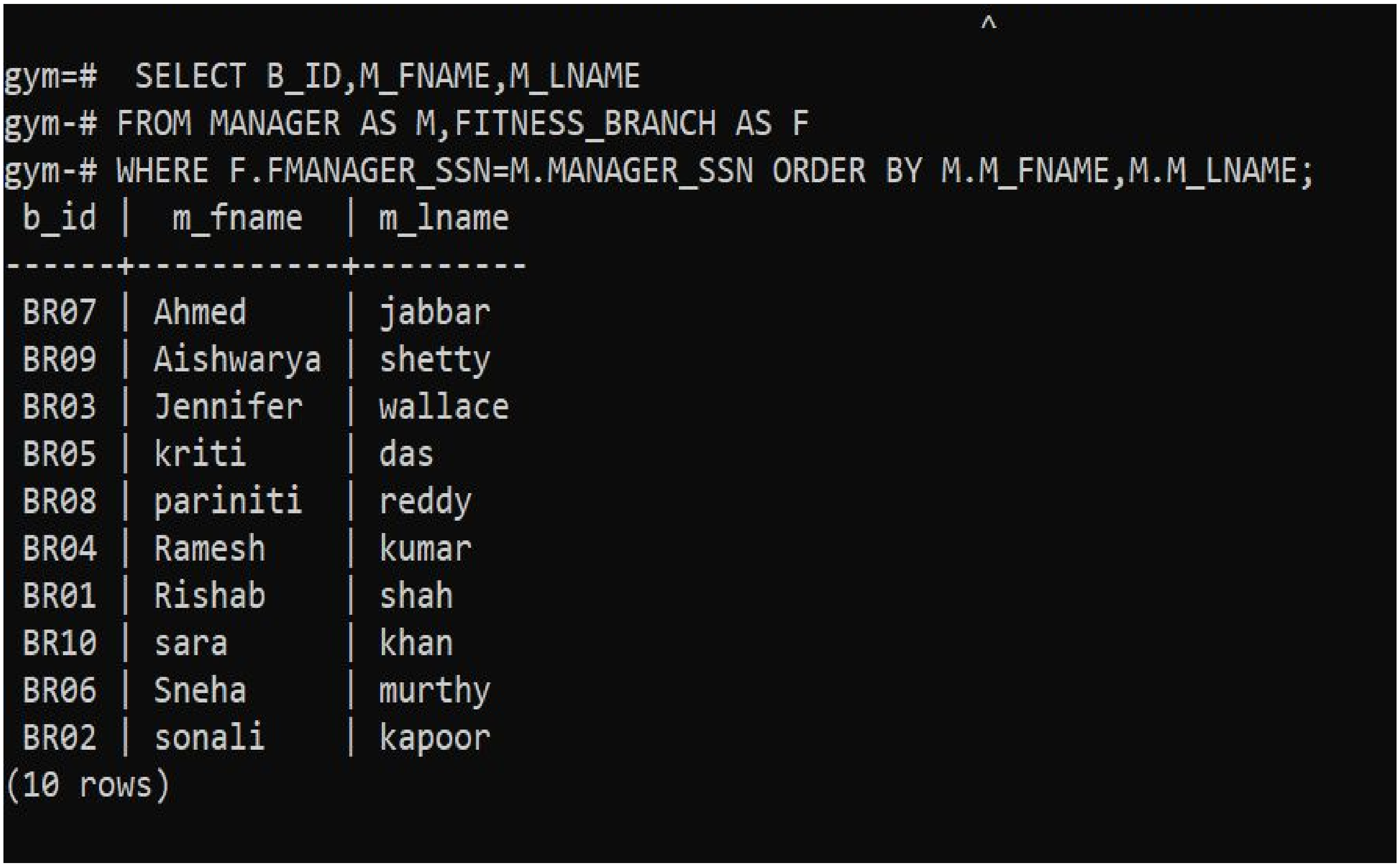


```
gym=#
gym=# UPDATE MANAGER
gym-# SET MONTHLY_SALARY=mONTHLY_SALARY*1.2
gym-# WHERE MANAGER_SSN='M33334';
UPDATE 1
gym=#
```

- Retrieve the list of managers managing a particular fitness branch order by manager fname,lname.

ALIASING USING AS KEYWORD
ORDER BY keyword sorts the records in ascending order by default.

SELECT B_ID,M_FNAME,M_LNAME
FROM MANAGER AS M,FITNESS_BRANCH AS F
WHERE F.FMANAGER_SSN=M.MANAGER_SSN
ORDER BY M.M_FNAME,M.M_LNAME;



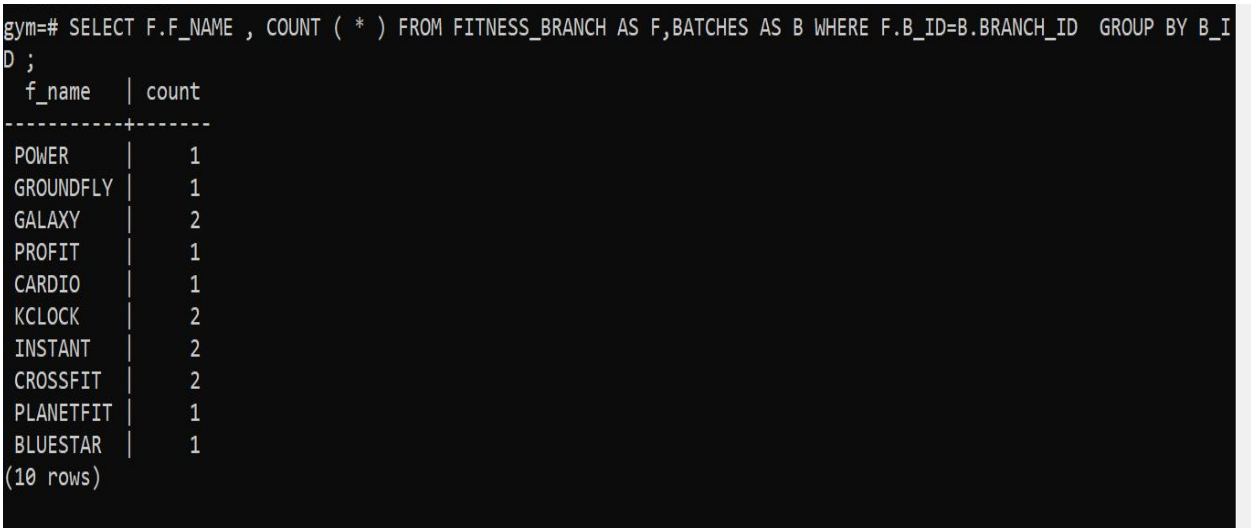
```
                                                    ^
gym=#  SELECT B_ID,M_FNAME,M_LNAME
gym-# FROM MANAGER AS M,FITNESS_BRANCH AS F
gym-# WHERE F.FMANAGER_SSN=M.MANAGER_SSN ORDER BY M.M_FNAME,M.M_LNAME;
 b_id |  m_fname  | m_lname
------+-----------+---------
 BR07 | Ahmed     | jabbar
 BR09 | Aishwarya | shetty
 BR03 | Jennifer  | wallace
 BR05 | kriti     | das
 BR08 | pariniti  | reddy
 BR04 | Ramesh    | kumar
 BR01 | Rishab    | shah
 BR10 | sara      | khan
 BR06 | Sneha     | murthy
 BR02 | sonali    | kapoor
(10 rows)
```

**COMPLEX QUERIES:**

1. For each fitness branch, retrieve the total number of batches.

ALIASING USING AS KEYWORD
GROUP BY STATEMENT TO GROUP ALL THE BATCHES

```
SELECT F.F_NAME , COUNT ( * )
FROM FITNESS_BRANCH AS F,BATCHES AS B
WHERE F.B_ID=B.BRANCH_ID
GROUP BY B_ID ;
```

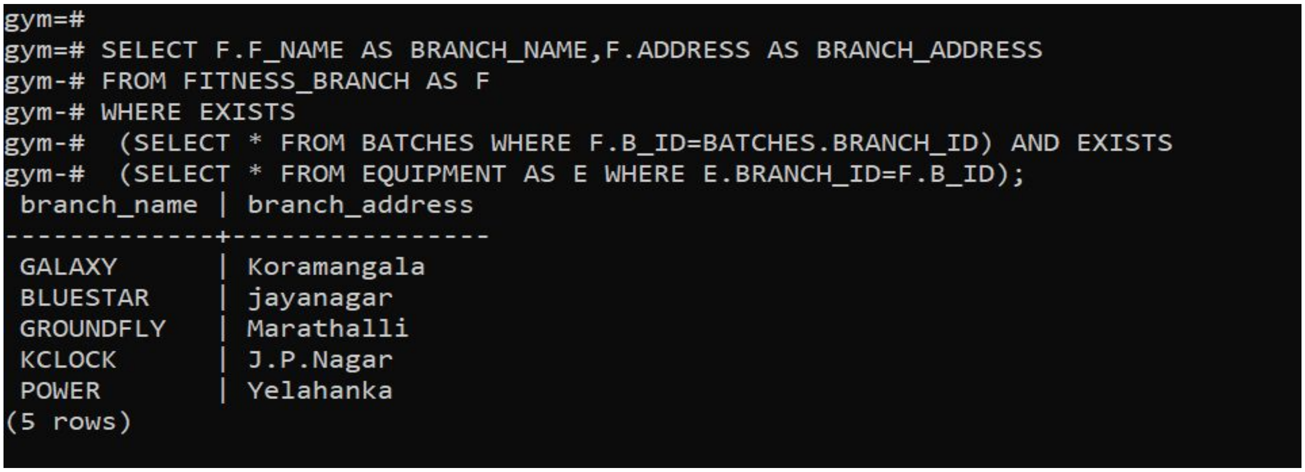


```
gym=# SELECT F.F_NAME , COUNT ( * ) FROM FITNESS_BRANCH AS F,BATCHES AS B WHERE F.B_ID=B.BRANCH_ID  GROUP BY B_I
D ;
  f_name   | count
-----------+-------
 POWER     |     1
 GROUNDFLY |     1
 GALAXY    |     2
 PROFIT    |     1
 CARDIO    |     1
 KCLOCK    |     2
 INSTANT   |     2
 CROSSFIT  |     2
 PLANETFIT |     1
 BLUESTAR  |     1
(10 rows)
```

2. List the name,address of fitness branches which has at least one equipment and at least one batch.

The EXISTS operator returns true if the subquery returns one or more records.

```
SELECT F.F_NAME AS BRANCH_NAME,F.ADDRESS AS BRANCH_ADDRESS
FROM FITNESS_BRANCH AS F
WHERE EXISTS
                  (SELECT * FROM BATCHES WHERE F.B_ID=BATCHES.BRANCH_ID)
         AND EXISTS
                 (SELECT * FROM EQUIPMENT AS E WHERE E.BRANCH_ID=F.B_ID);
```

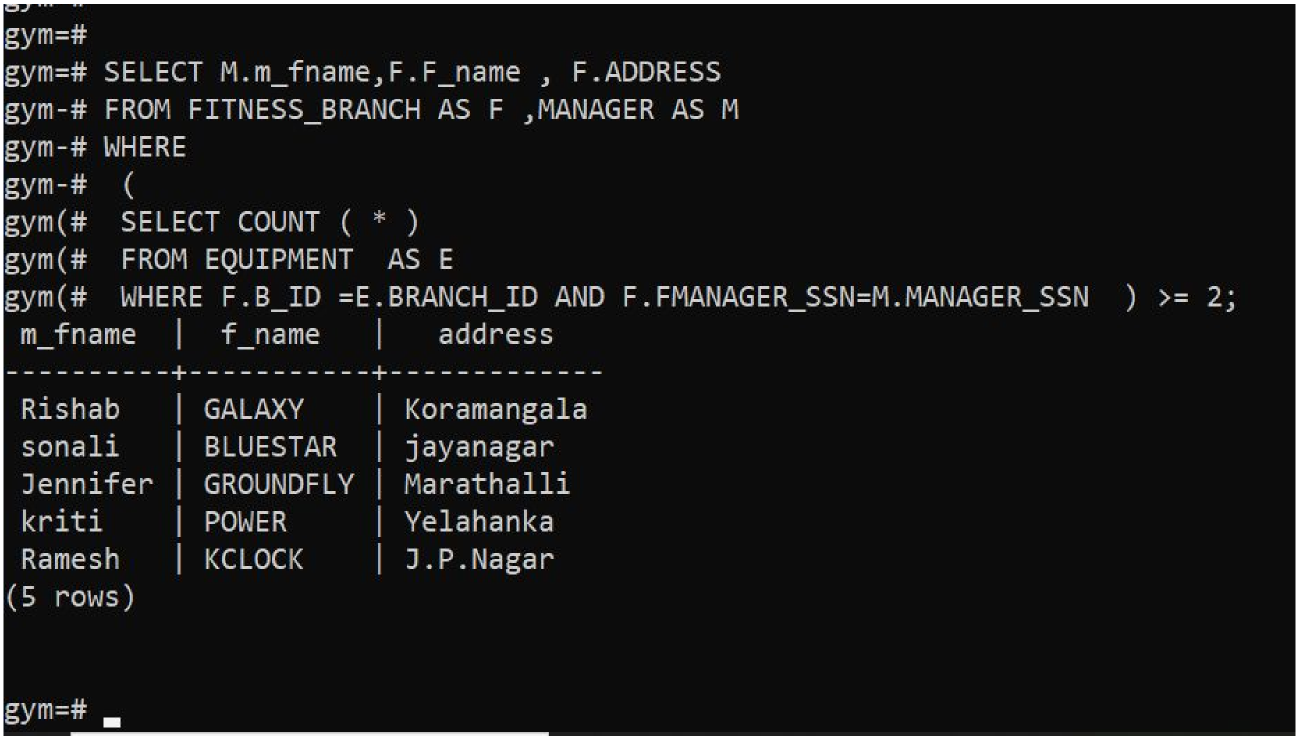


```
gym=#
gym=# SELECT F.F_NAME AS BRANCH_NAME,F.ADDRESS AS BRANCH_ADDRESS
gym-# FROM FITNESS_BRANCH AS F
gym-# WHERE EXISTS
gym-#  (SELECT * FROM BATCHES WHERE F.B_ID=BATCHES.BRANCH_ID) AND EXISTS
gym-#  (SELECT * FROM EQUIPMENT AS E WHERE E.BRANCH_ID=F.B_ID);
 branch_name | branch_address
-------------+----------------
 GALAXY      | Koramangala
 BLUESTAR    | jayanagar
 GROUNDFLY   | Marathalli
 KCLOCK      | J.P.Nagar
 POWER       | Yelahanka
(5 rows)
```

3. Retrieve the names of all branches and its manager which have two or more equipments

```
SELECT M.m_fname,F.F_name , F.ADDRESS
FROM FITNESS_BRANCH AS F ,MANAGER AS M
```

```
WHERE
 (
        SELECT COUNT ( * )
        FROM EQUIPMENT  AS E
        WHERE F.B_ID =E.BRANCH_ID AND
F.FMANAGER_SSN=M.MANAGER_SSN  ) >= 2;
```

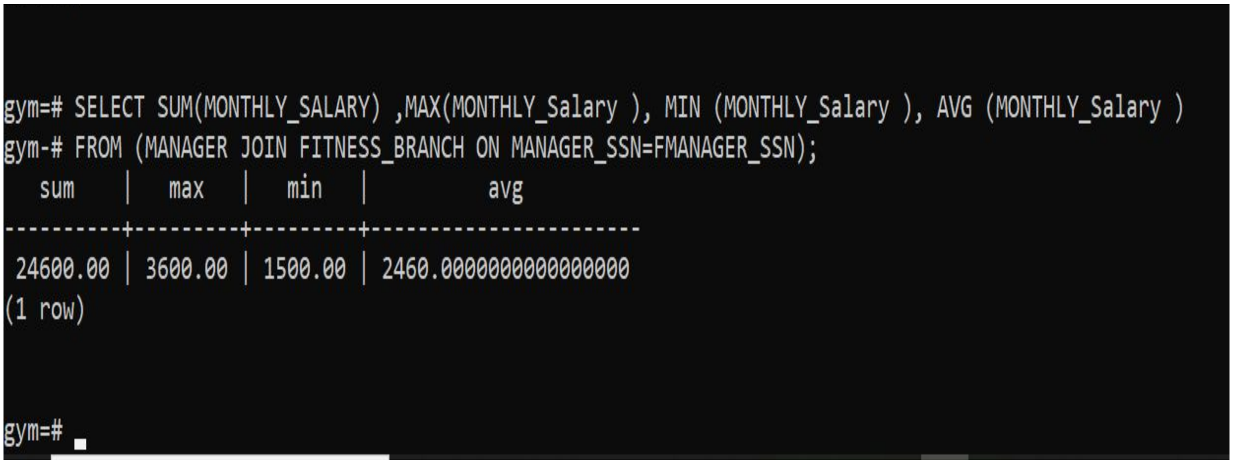


```
gym=#
gym=# SELECT M.m_fname,F.F_name , F.ADDRESS
gym-# FROM FITNESS_BRANCH AS F ,MANAGER AS M
gym-# WHERE
gym-#  (
gym(#  SELECT COUNT ( * )
gym(#  FROM EQUIPMENT  AS E
gym(#  WHERE F.B_ID =E.BRANCH_ID AND F.FMANAGER_SSN=M.MANAGER_SSN  ) >= 2;
 m_fname  |  f_name   |   address
----------+-----------+------------
 Rishab   | GALAXY    | Koramangala
 sonali   | BLUESTAR  | jayanagar
 Jennifer | GROUNDFLY | Marathalli
 kriti    | POWER     | Yelahanka
 Ramesh   | KCLOCK    | J.P.Nagar
(5 rows)


gym=#
```

4. Find the sum of the salaries of all managers, the maximum salary, the minimum salary, and the average salary.

AGGREGATE FUNCTIONS

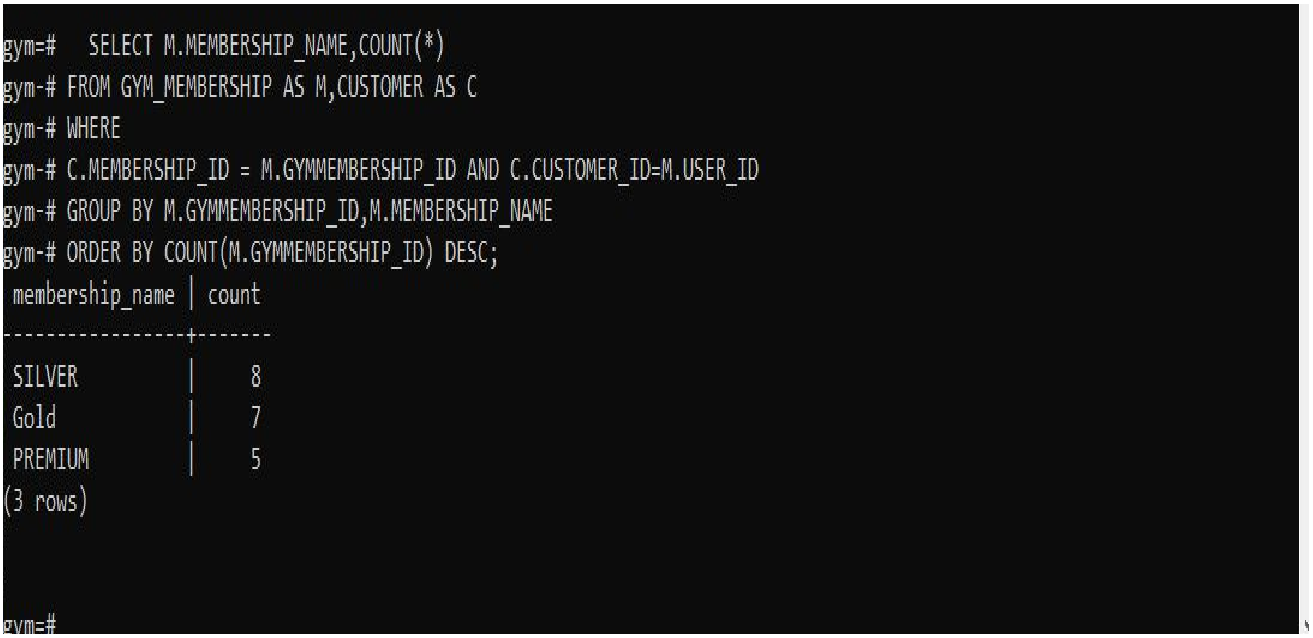
```
SELECT SUM(MONTHLY_SALARY) ,
        MAX(MONTHLY_Salary ), MIN (MONTHLY_Salary ),
        AVG (MONTHLY_Salary )
FROM
(MANAGER JOIN FITNESS_BRANCH ON
MANAGER_SSN=FMANAGER_SSN);
```



```
gym=# SELECT SUM(MONTHLY_SALARY) ,MAX(MONTHLY_Salary ), MIN (MONTHLY_Salary ), AVG (MONTHLY_Salary )
gym-# FROM (MANAGER JOIN FITNESS_BRANCH ON MANAGER_SSN=FMANAGER_SSN);
   sum    |   max   |   min   |          avg
----------+---------+---------+-----------------------
 24600.00 | 3600.00 | 1500.00 | 2460.0000000000000000
(1 row)


gym=#
```

5. For each membership retrieve the membership name and total number of customers owned that membership.

```
SELECT M.MEMBERSHIP_NAME,COUNT(*)
FROM GYM_MEMBERSHIP AS M,CUSTOMER AS C
WHERE
C.MEMBERSHIP_ID = M.GYMMEMBERSHIP_ID AND
C.CUSTOMER_ID=M.USER_ID
GROUP BY M.GYMMEMBERSHIP_ID,M.MEMBERSHIP_NAME
ORDER BY COUNT(M.GYMMEMBERSHIP_ID) DESC;
```



```
gym=#   SELECT M.MEMBERSHIP_NAME,COUNT(*)
gym-# FROM GYM_MEMBERSHIP AS M,CUSTOMER AS C
gym-# WHERE
gym-# C.MEMBERSHIP_ID = M.GYMMEMBERSHIP_ID AND C.CUSTOMER_ID=M.USER_ID
gym-# GROUP BY M.GYMMEMBERSHIP_ID,M.MEMBERSHIP_NAME
gym-# ORDER BY COUNT(M.GYMMEMBERSHIP_ID) DESC;
 membership_name | count
-----------------+-------
 SILVER          |     8
 Gold            |     7
 PREMIUM         |     5
(3 rows)


gym=#
```

## CONCLUSION:

By working on this project I have gained a better understanding of developing an conceptual schema,developing database relations,writing queries,normalization.Learning of the concepts have become stronger .