A Project Report On Club Management System

DEVELOPED BY:
IT118 – NAND RABADIYA
IT116 – PREET BRAHMBHATT

Guided By
Internal Guide:
Prof. Shweta Jambukia

Department of Information Technology
Faculty of Technology
DD University

Department of Information Technology Faculty of Technology, Dharmsinh Desai University College Road, Nadiad-387001
March-2024

<u>CERTIFICATE</u>

This is to certify that the project entitled “Blood Donation Management system” is a bonafide report of the work carried out by

1) **NAND RABADIYA** Student ID No: **22ITUOS003**
2) **PREET BRAMBHATT** Student ID No: **22ITUON075**

of Department of Information Technology, Semester IV, under the guidance and supervision for the subject Database Management System. They were involved in Project training during the academic year 2023-2024.

Prof. Shweta Jambukia
Project Guide, Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:04/03/2022

Prof. Vipul Dabhi
Head, Department of Information Technology

INDEX

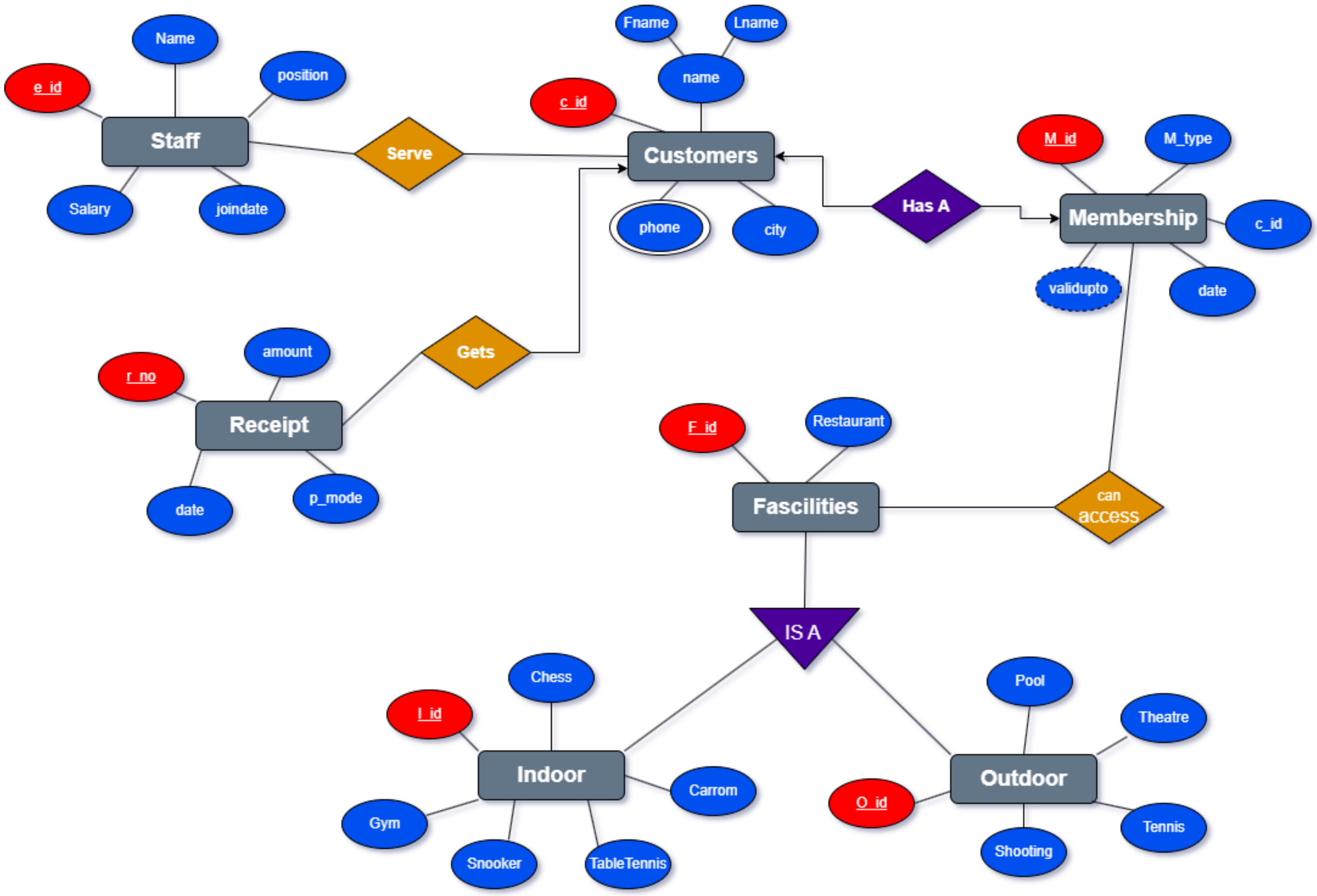
1.SYSTEM OVERVIEW

Data is the code word of the computer industry. Data refers to a collection of facts usually collected as a result of observation and experiment or processes within a computer system. This may consist of numbers, words or images or observations of a set of variables. Data are often viewed as a lowest level of abstraction from which information and knowledge are derived.

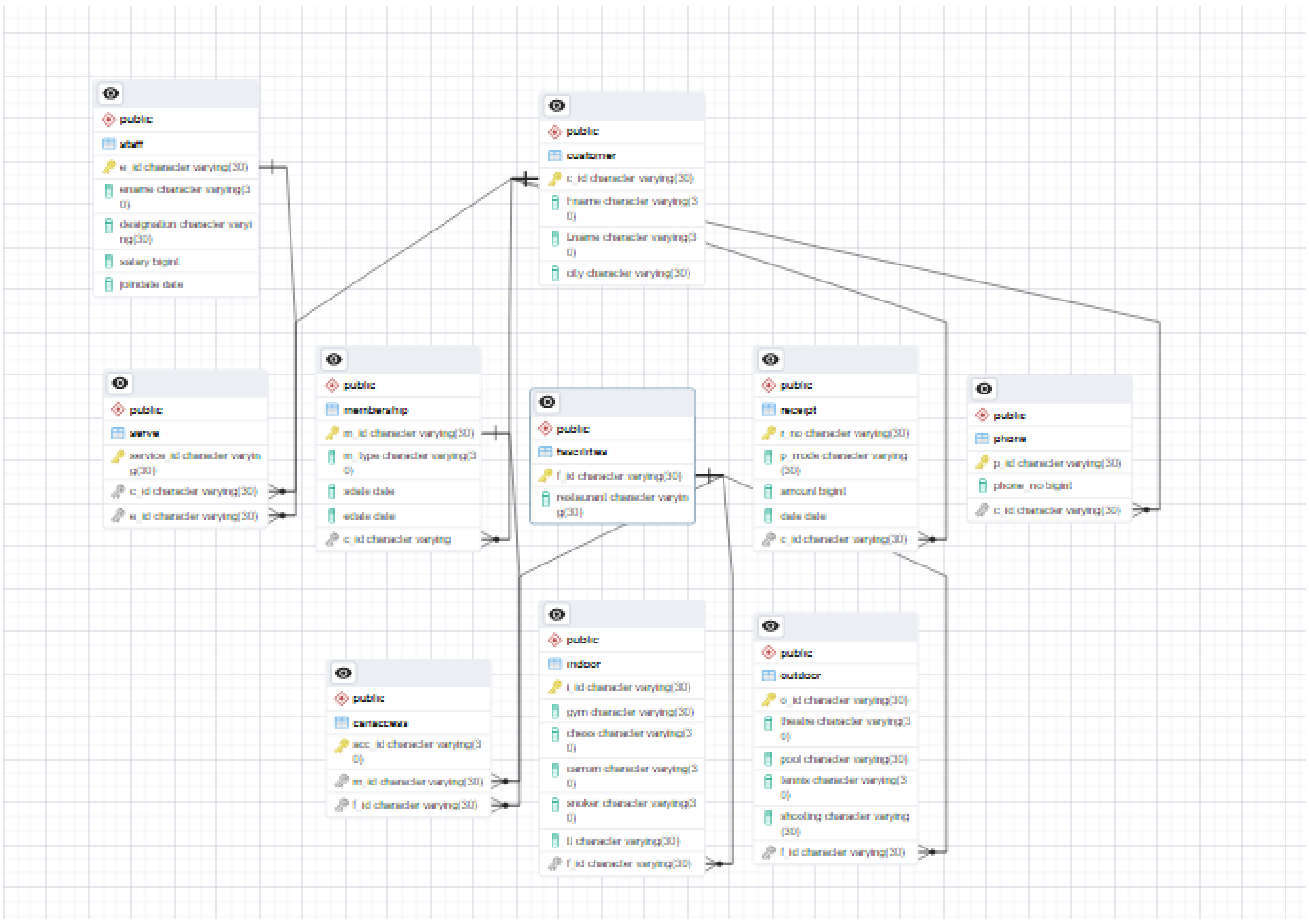
Club Management System (BBMS) is a web based system that can assists the information of the club for easy operations and management. With this system, the user of this system can key in the required data for managing the club and its guests and members.

From this system, we can retrieve the information regarding any customer, their membership, amenities, staff and payment details. Hence, DBMS will make the Club data more systematic and manageable.

2. ENTITY-RELATIONSHIP MODEL



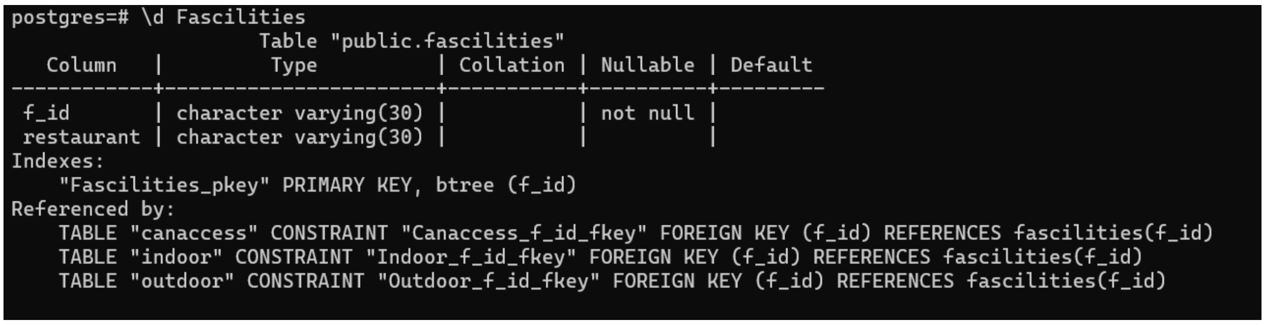
3. RELATIONAL SCHEMA



public
staff
e_id character varying(30)
ename character varying(30)
designation character varying(30)
salary bigint
joindate date
public
customer
c_id character varying(30)
Fname character varying(30)
Lname character varying(30)
city character varying(30)
public
serve
service_id character varying(30)
c_id character varying(30)
e_id character varying(30)
public
membership
m_id character varying(30)
m_type character varying(30)
sdate date
edate date
c_id character varying
public
facilities
f_id character varying(30)
restaurant character varying(30)
public
receipt
r_no character varying(30)
p_mode character varying(30)
amount bigint
date date
c_id character varying(30)
public
phone
p_id character varying(30)
phone_no bigint
c_id character varying(30)
public
canaccess
acc_id character varying(30)
m_id character varying(30)
f_id character varying(30)
public
indoor
i_id character varying(30)
gym character varying(30)
chess character varying(30)
carrom character varying(30)
snoker character varying(30)
tt character varying(30)
f_id character varying(30)
public
outdoor
o_id character varying(30)
theatre character varying(30)
pool character varying(30)
tennis character varying(30)
shooting character varying(30)
f_id character varying(30)

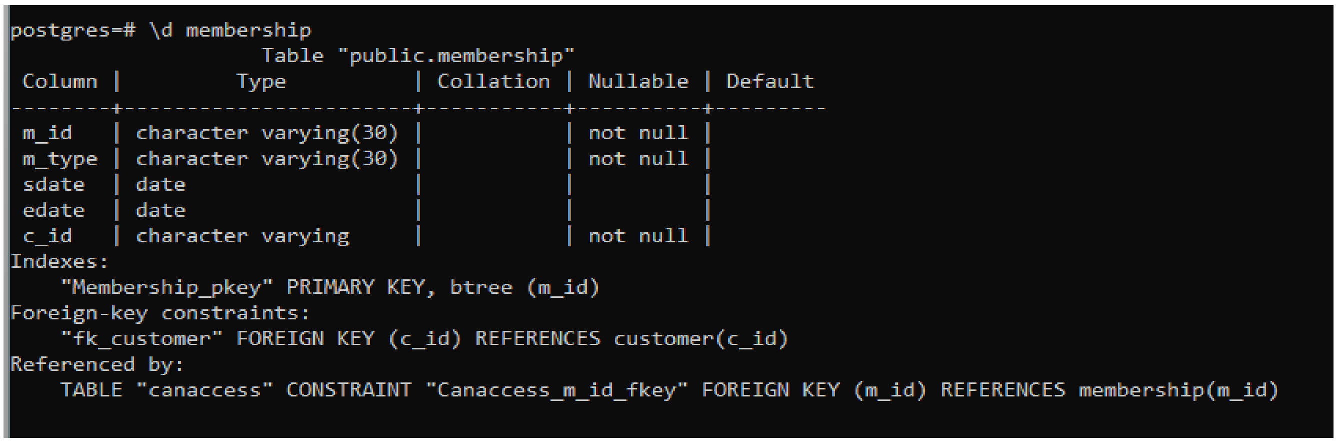
4. Data Dictionary

4.1 Fascilities



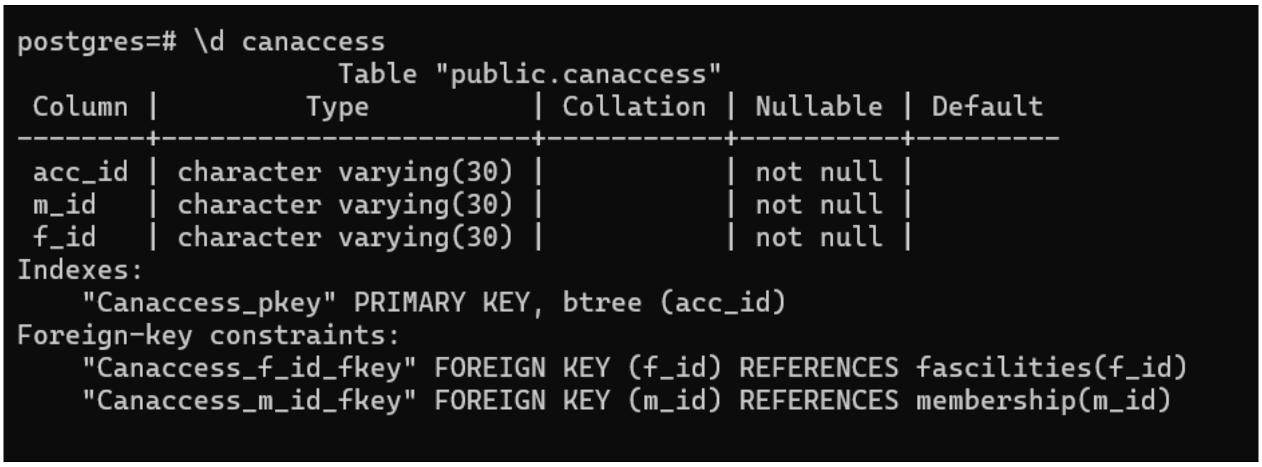
```
postgres=# \d Fascilities
                    Table "public.fascilities"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 f_id       | character varying(30) |           | not null |
 restaurant | character varying(30) |           |          |
Indexes:
    "Fascilities_pkey" PRIMARY KEY, btree (f_id)
Referenced by:
    TABLE "canaccess" CONSTRAINT "Canaccess_f_id_fkey" FOREIGN KEY (f_id) REFERENCES fascilities(f_id)
    TABLE "indoor" CONSTRAINT "Indoor_f_id_fkey" FOREIGN KEY (f_id) REFERENCES fascilities(f_id)
    TABLE "outdoor" CONSTRAINT "Outdoor_f_id_fkey" FOREIGN KEY (f_id) REFERENCES fascilities(f_id)
```

4.2 Membership



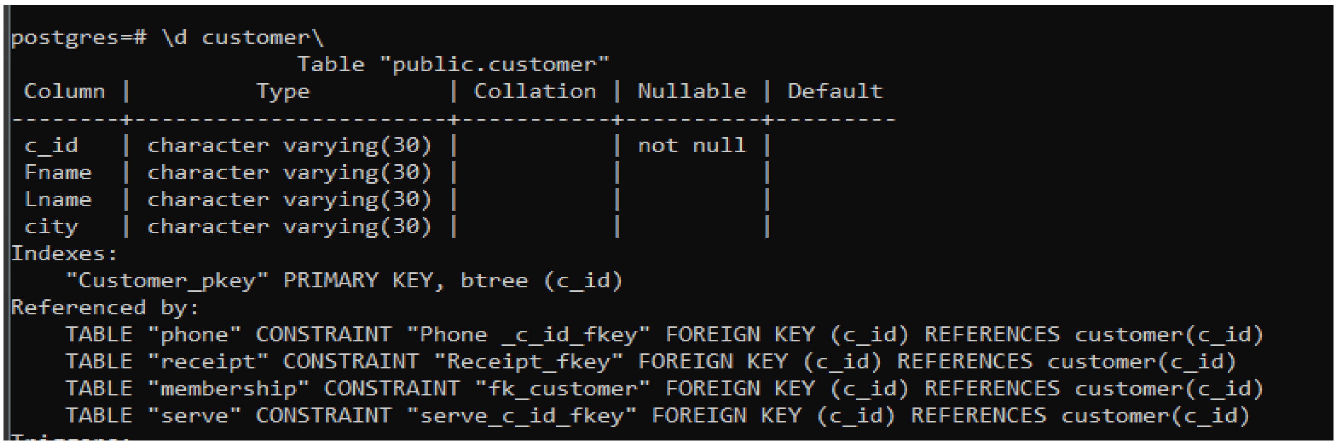
```
postgres=# \d membership
                    Table "public.membership"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 m_id   | character varying(30) |           | not null |
 m_type | character varying(30) |           | not null |
 sdate  | date                  |           |          |
 edate  | date                  |           |          |
 c_id   | character varying     |           | not null |
Indexes:
    "Membership_pkey" PRIMARY KEY, btree (m_id)
Foreign-key constraints:
    "fk_customer" FOREIGN KEY (c_id) REFERENCES customer(c_id)
Referenced by:
    TABLE "canaccess" CONSTRAINT "Canaccess_m_id_fkey" FOREIGN KEY (m_id) REFERENCES membership(m_id)
```

4.3 Canaccess



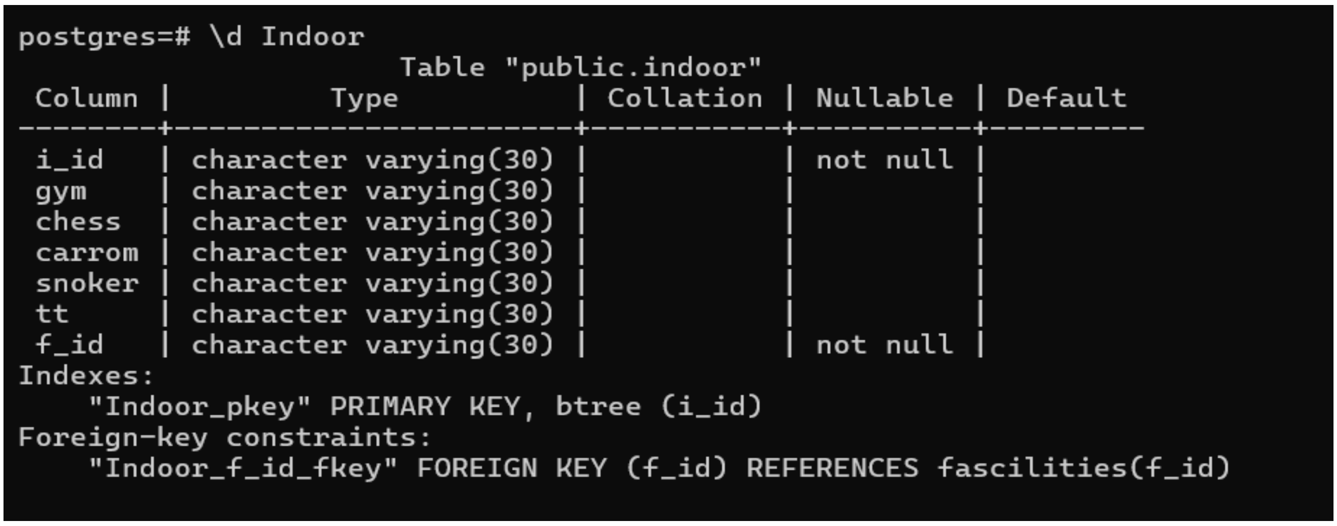
```
postgres=# \d canaccess
                    Table "public.canaccess"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 acc_id | character varying(30) |           | not null |
 m_id   | character varying(30) |           | not null |
 f_id   | character varying(30) |           | not null |
Indexes:
    "Canaccess_pkey" PRIMARY KEY, btree (acc_id)
Foreign-key constraints:
    "Canaccess_f_id_fkey" FOREIGN KEY (f_id) REFERENCES fascilities(f_id)
    "Canaccess_m_id_fkey" FOREIGN KEY (m_id) REFERENCES membership(m_id)
```

4.4 Customer



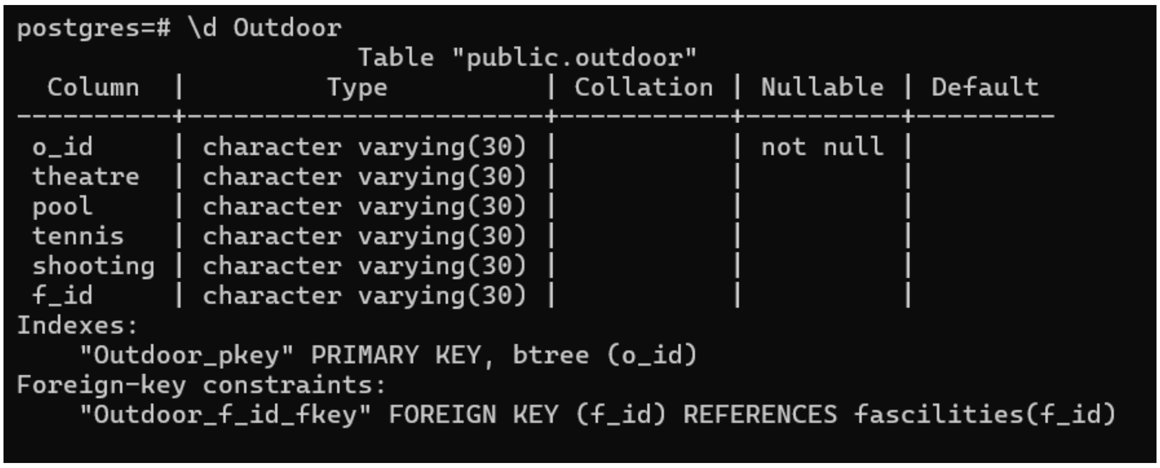
```
postgres=# \d customer\
                    Table "public.customer"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 c_id   | character varying(30) |           | not null |
 Fname  | character varying(30) |           |          |
 Lname  | character varying(30) |           |          |
 city   | character varying(30) |           |          |
Indexes:
    "Customer_pkey" PRIMARY KEY, btree (c_id)
Referenced by:
    TABLE "phone" CONSTRAINT "Phone _c_id_fkey" FOREIGN KEY (c_id) REFERENCES customer(c_id)
    TABLE "receipt" CONSTRAINT "Receipt_fkey" FOREIGN KEY (c_id) REFERENCES customer(c_id)
    TABLE "membership" CONSTRAINT "fk_customer" FOREIGN KEY (c_id) REFERENCES customer(c_id)
    TABLE "serve" CONSTRAINT "serve_c_id_fkey" FOREIGN KEY (c_id) REFERENCES customer(c_id)
```

4.5 Indoor



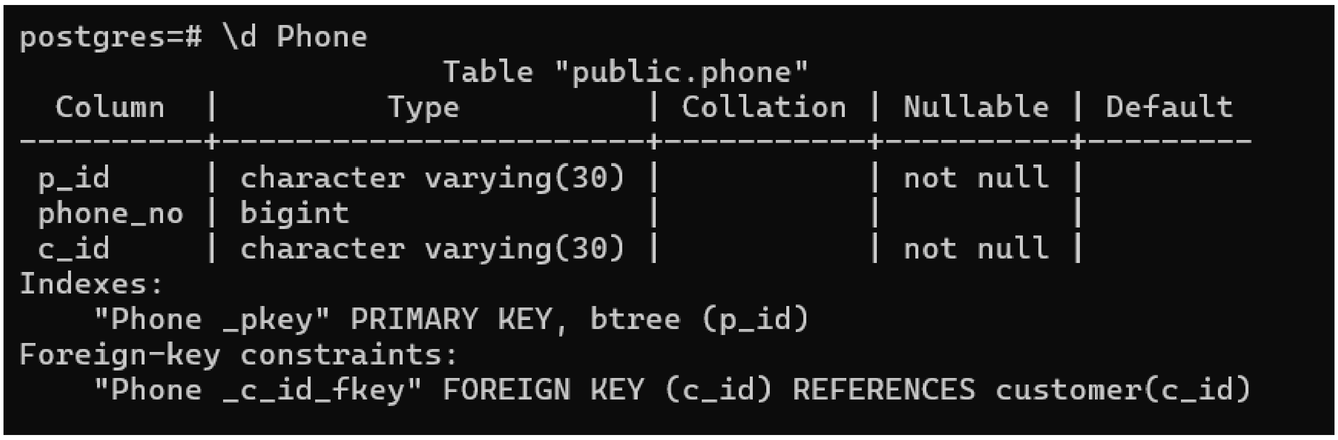
```
postgres=# \d Indoor
                      Table "public.indoor"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 i_id   | character varying(30) |           | not null |
 gym    | character varying(30) |           |          |
 chess  | character varying(30) |           |          |
 carrom | character varying(30) |           |          |
 snoker | character varying(30) |           |          |
 tt     | character varying(30) |           |          |
 f_id   | character varying(30) |           | not null |
Indexes:
    "Indoor_pkey" PRIMARY KEY, btree (i_id)
Foreign-key constraints:
    "Indoor_f_id_fkey" FOREIGN KEY (f_id) REFERENCES fascilities(f_id)
```

4.6 Outdoor

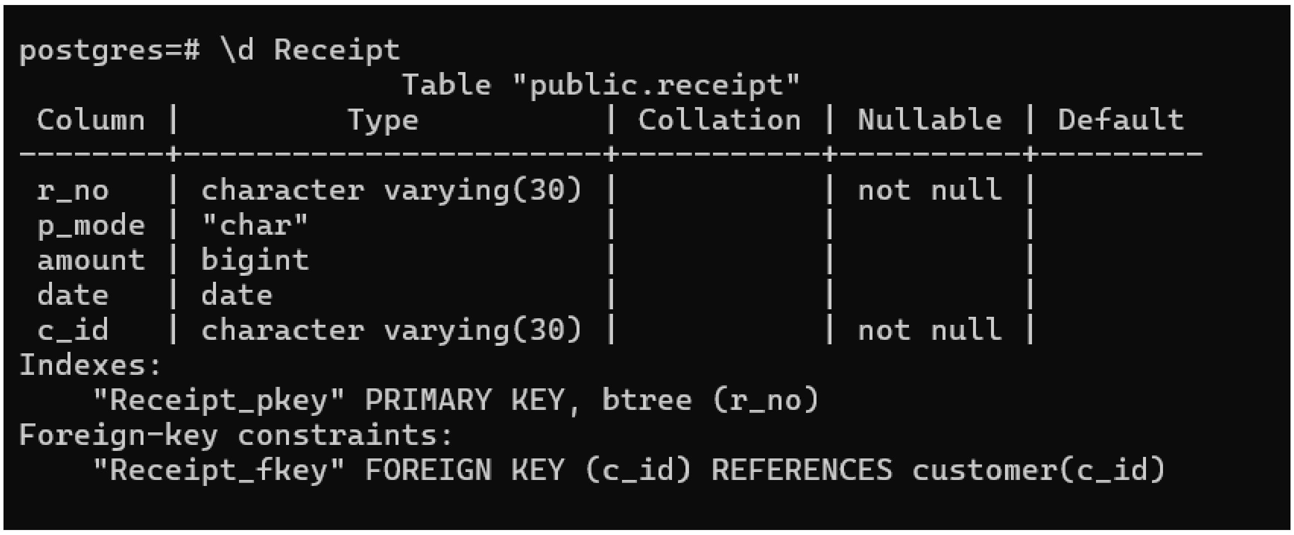


```
postgres=# \d Outdoor
                      Table "public.outdoor"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 o_id     | character varying(30) |           | not null |
 theatre  | character varying(30) |           |          |
 pool     | character varying(30) |           |          |
 tennis   | character varying(30) |           |          |
 shooting | character varying(30) |           |          |
 f_id     | character varying(30) |           |          |
Indexes:
    "Outdoor_pkey" PRIMARY KEY, btree (o_id)
Foreign-key constraints:
    "Outdoor_f_id_fkey" FOREIGN KEY (f_id) REFERENCES fascilities(f_id)
```

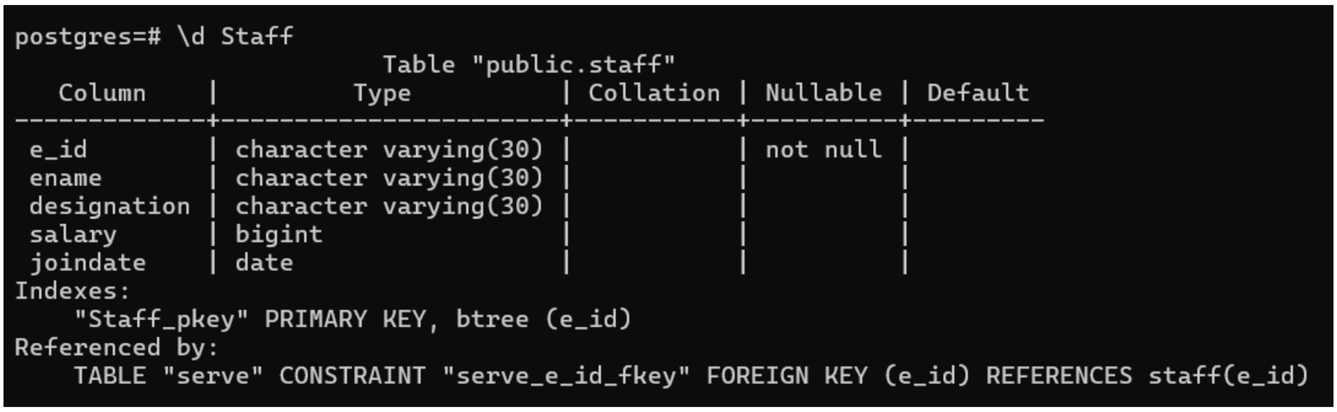
4.7 Phone


```
postgres=# \d Phone
                       Table "public.phone"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 p_id     | character varying(30) |           | not null |
 phone_no | bigint                |           |          |
 c_id     | character varying(30) |           | not null |
Indexes:
    "Phone _pkey" PRIMARY KEY, btree (p_id)
Foreign-key constraints:
    "Phone _c_id_fkey" FOREIGN KEY (c_id) REFERENCES customer(c_id)
```

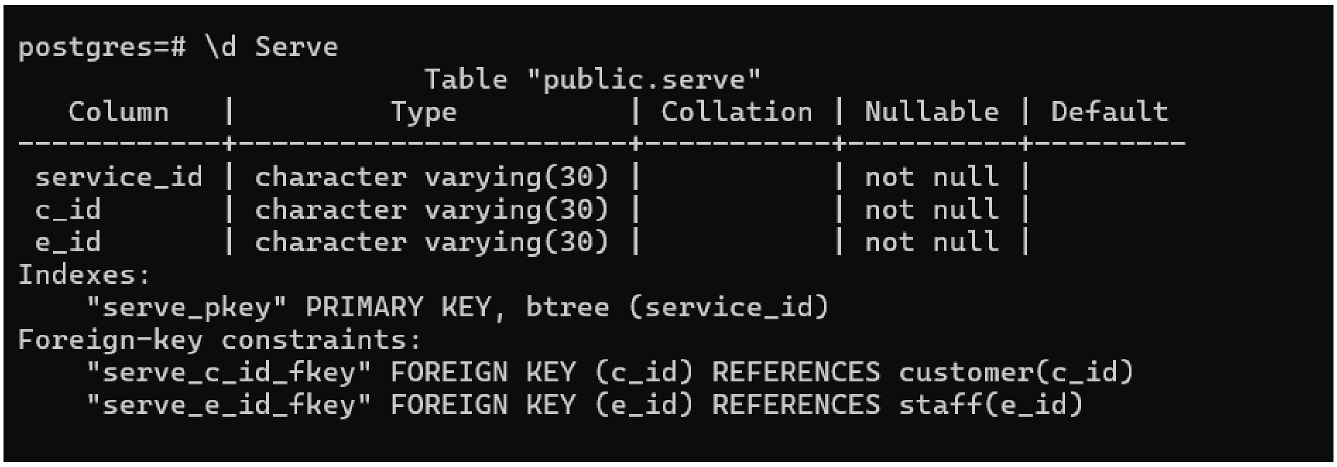
4.8 Receipt


```
postgres=# \d Receipt
                      Table "public.receipt"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 r_no   | character varying(30) |           | not null |
 p_mode | "char"                |           |          |
 amount | bigint                |           |          |
 date   | date                  |           |          |
 c_id   | character varying(30) |           | not null |
Indexes:
    "Receipt_pkey" PRIMARY KEY, btree (r_no)
Foreign-key constraints:
    "Receipt_fkey" FOREIGN KEY (c_id) REFERENCES customer(c_id)
```

4.9 Staff


```
postgres=# \d Staff
                        Table "public.staff"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 e_id        | character varying(30) |           | not null |
 ename       | character varying(30) |           |          |
 designation | character varying(30) |           |          |
 salary      | bigint                |           |          |
 joindate    | date                  |           |          |
Indexes:
    "Staff_pkey" PRIMARY KEY, btree (e_id)
Referenced by:
    TABLE "serve" CONSTRAINT "serve_e_id_fkey" FOREIGN KEY (e_id) REFERENCES staff(e_id)
```

4.10 Serve



```
postgres=# \d Serve
                    Table "public.serve"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 service_id | character varying(30) |           | not null |
 c_id       | character varying(30) |           | not null |
 e_id       | character varying(30) |           | not null |
Indexes:
    "serve_pkey" PRIMARY KEY, btree (service_id)
Foreign-key constraints:
    "serve_c_id_fkey" FOREIGN KEY (c_id) REFERENCES customer(c_id)
    "serve_e_id_fkey" FOREIGN KEY (e_id) REFERENCES staff(e_id)
```

5. DATA IMPLEMENTATION

A) SCHEMA

5.1.1 Fascilities

```
CREATE TABLE Fascilities
(
  f_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
  restaurant varchar(30) COLLATE pg_catalog."default",
  CONSTRAINT "Fascilities_pkey" PRIMARY KEY (f_id)
);
```

5.1.2 Membership

```
CREATE TABLE Membership
(
  m_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
  m_type varchar(30) COLLATE pg_catalog."default" NOT NULL,
  sdate date,
  edate date,
  CONSTRAINT "Membership_pkey" PRIMARY KEY (m_id),
  CONSTRAINT "c_id_fkey" FOREIGN KEY (c_id)
    REFERENCES Customer (c_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
);
```

5.1.3 Canaccess

```
CREATE TABLE Canaccess
(
  acc_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
  m_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
  f_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
  CONSTRAINT "Canaccess_pkey" PRIMARY KEY (acc_id),
  CONSTRAINT "Canaccess_f_id_fkey" FOREIGN KEY (f_id)
    REFERENCES Fascilities (f_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
  CONSTRAINT "Canaccess_m_id_fkey" FOREIGN KEY (m_id)
    REFERENCES Membership (m_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
```

```
        NOT VALID
);
```

5.1.4 Customer

```
CREATE TABLE Customer
(
    c_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
    "Fname" varchar(30) COLLATE pg_catalog."default",
    "Lname" varchar(30)COLLATE pg_catalog."default",
    city varchar(30) COLLATE pg_catalog."default",
    CONSTRAINT "Customer_pkey" PRIMARY KEY (c_id)
);
```

5.1.5 Indoor

```
CREATE TABLE Indoor
(
    i_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
    gym varchar(30) COLLATE pg_catalog."default",
    chess varchar(30) COLLATE pg_catalog."default",
    carrom varchar(30) COLLATE pg_catalog."default",
    snoker varchar(30) COLLATE pg_catalog."default",
    tt varchar(30) COLLATE pg_catalog."default",
    f_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Indoor_pkey" PRIMARY KEY (i_id),
    CONSTRAINT "Indoor_f_id_fkey" FOREIGN KEY (f_id)
        REFERENCES Fascilities (f_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

5.1.6 Outdoor

```
CREATE TABLE Outdoor
(
    o_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
    theatre varchar(30) COLLATE pg_catalog."default",
    pool varchar(30) COLLATE pg_catalog."default",
    tennis varchar(30) COLLATE pg_catalog."default",
    shooting varchar(30) COLLATE pg_catalog."default",
    f_id varchar(30) COLLATE pg_catalog."default",
    CONSTRAINT "Outdoor_pkey" PRIMARY KEY (o_id),
    CONSTRAINT "Outdoor_f_id_fkey" FOREIGN KEY (f_id)
        REFERENCES Fascilities (f_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
  );
```

5.1.7 Phone

```
CREATE TABLE Phone
(
    p_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
    phone_no bigint,
    c_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Phone _pkey" PRIMARY KEY (p_id),
    CONSTRAINT "Phone _c_id_fkey" FOREIGN KEY (c_id)
        REFERENCES Customer (c_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

5.1.8 Receipt

```
CREATE TABLE Receipt
(
    r_no varchar(30) COLLATE pg_catalog."default" NOT NULL,
    p_mode "char",
    amount bigint,
    date date,
    c_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Receipt_pkey" PRIMARY KEY (r_no),
    CONSTRAINT "Receipt_fkey" FOREIGN KEY (c_id)
        REFERENCES Customer (c_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

5.1.9 Staff

```
CREATE TABLE Staff
(
    e_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
    ename varchar(30) COLLATE pg_catalog."default",
    designation varchar(30) COLLATE pg_catalog."default",
    salary bigint,
    joindate date,
    CONSTRAINT "Staff_pkey" PRIMARY KEY (e_id)
);
```

5.1.10 Serve

```
CREATE TABLE serve
(
   service_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
   c_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
   e_id varchar(30) COLLATE pg_catalog."default" NOT NULL,
   CONSTRAINT serve_pkey PRIMARY KEY (service_id),
   CONSTRAINT serve_c_id_fkey FOREIGN KEY (c_id)
      REFERENCES Customer (c_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
      NOT VALID,
   CONSTRAINT serve_e_id_fkey FOREIGN KEY (e_id)
      REFERENCES Staff (e_id) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
      NOT VALID
);
```

B) DATA INSERTION

5.2.1 Fascilities

```
INSERT INTO Fascilities (f_id, restaurant) VALUES
('201', 'A'),
('202', 'B'),
('203', 'C');
```

5.2.2 Membership

```
INSERT INTO Membership (m_id, m_type, sdate, edate)
VALUES
('401', 'Monthly', '2023-01-01', '2023-01-31'),
('402', 'Annual', '2023-02-15', '2024-02-14'),
('403', 'Monthly', '2023-03-10', '2023-03-31'),
('404', 'Annual', '2023-04-20', '2024-04-19'),
('405', 'Monthly', '2023-05-05', '2023-05-31'),
('406', 'Annual', '2023-06-15', '2024-06-14')
('407', 'Monthly', '2023-07-01', '2023-07-31'),
('408', 'Annual', '2023-08-15', '2024-08-14'),
('409', 'Guest', '2023-09-10', '2023-09-30'),
('410', 'Guest', '2023-10-20', '2023-11-19');
```

5.2.3 Canaccess

```
INSERT INTO Canaccess (acc_id, m_id, f_id)
VALUES
('1', '401', '201'),
('2', '402', '202'),
('3', '403', '203'),
('4', '404', '201'),
('5', '405', '202'),
('6', '406', '203')
('7', '407', '201'),
('8', '408', '202'),
('9', '409', '203'),
('10', '404', '201'),
('11', '405', '202'),
('12', '402', '203'),
('13', '407', '203');
```

5.2.4 Customer

```
INSERT INTO Customer (c_id, "Fname", "Lname", city)
VALUES
('201', 'Robert', 'Johnson', 'San Francisco'),
('202', 'Mary', 'Wilson', 'Seattle'),
('203', 'Christopher', 'Brown', 'Dallas'),
('204', 'Jennifer', 'Davis', 'Boston'),
('205', 'William', 'Miller', 'Philadelphia')
('206', 'Emily', 'Anderson', 'New York'),
('207', 'Daniel', 'Martinez', 'Los Angeles'),
('208', 'Olivia', 'Garcia', 'Chicago'),
('209', 'James', 'Lopez', 'Houston');
```

5.2.5 Indoor

```
INSERT INTO Indoor (i_id, gym, chess, carrom, snoker, tt, f_id)
VALUES
('301', 'Yes', 'Yes', 'No', 'No', 'Yes', '201'),
('302', 'Yes', 'No', 'Yes', 'No', 'No', '202'),
('303', 'No', 'Yes', 'No', 'Yes', 'Yes', '203')
('304', 'Yes', 'Yes', 'No', 'No', 'Yes', '201'),
('305', 'Yes', 'No', 'Yes', 'No', 'No', '202'),
('306', 'No', 'Yes', 'No', 'Yes', 'Yes', '203'),
('307', 'Yes', 'Yes', 'No', 'No', 'No', '201');
```

5.2.6 Outdoor

```
INSERT INTO Outdoor (o_id, theatre, pool, tennis, shooting, f_id)
VALUES
('501', 'Yes', 'No', 'Yes', 'No', '201'),
('502', 'No', 'Yes', 'No', 'Yes', '202'),
('503', 'Yes', 'Yes', 'Yes', 'No', '203')
('504', 'Yes', 'No', 'Yes', 'No', '201'),
('505', 'No', 'Yes', 'No', 'Yes', '202'),
('506', 'Yes', 'Yes', 'Yes', 'No', '203'),
('507', 'No', 'No', 'Yes', 'Yes', '201');
```

5.2.7 Phone

```
INSERT INTO Phone (p_id, phone_no, c_id)
 VALUES
 ('601', 1234567890, '201'),
 ('602', 9876543210, '202'),
 ('603', 5555555555, '203'),
 ('604', 1112223333, '204'),
 ('605', 9998887777, '205')
 ('606', 1111111111, '206'),
 ('607', 2222222222, '207'),
 ('608', 3333333333, '208'),
 ('609', 4444444444, '209'),
 ('610', 1123678911, '206'),
 ('611', 2235522222, '202'),
 ('612', 3332579333, '208');
```

5.2.8 Receipt

```
INSERT INTO Receipt (r_no, p_mode, amount, date, c_id)
VALUES
('701', 'Cash', 50, '2023-01-05', '201'),
('702', 'Card', 100, '2023-02-10', '202'),
('703', 'Cash', 75, '2023-03-15', '203'),
('704', 'Card', 200, '2023-04-20', '204'),
('705', 'Cash', 150, '2023-05-25', '205')
('706', 'Cash', 80, '2023-06-30', '206'),
('707', 'Card', 120, '2023-07-05', '207'),
('708', 'Cash', 90, '2023-08-10', '208'),
('709', 'Card', 180, '2023-09-15', '209');
```

5.2.9 Staff

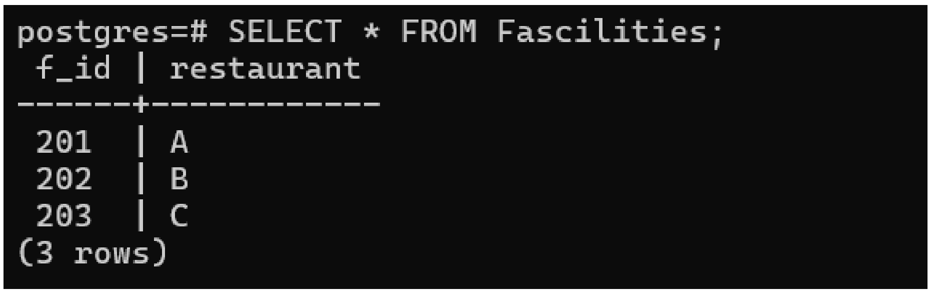
```
INSERT INTO Staff (e_id, ename, designation, salary, joindate)
VALUES
('801', 'Jessica', 'Manager', 60000, '2022-12-15'),
('802', 'Michael', 'Receptionist', 35000, '2023-01-20'),
('803', 'Christopher', 'Trainer', 45000, '2023-02-25'),
('804', 'Sarah', 'Manager', 55000, '2023-03-15'),
('805', 'David', 'Receptionist', 38000, '2023-04-20');
```

5.2.10 Serve

```
INSERT INTO serve (service_id, c_id, e_id)
VALUES
('901', '201', '801'),
('902', '202', '802'),
('903', '203', '803'),
('904', '204', '804'),
('905', '205', '805')
('906', '206', '801'),
('907', '207', '802'),
('908', '208', '803'),
('909', '209', '804');
```

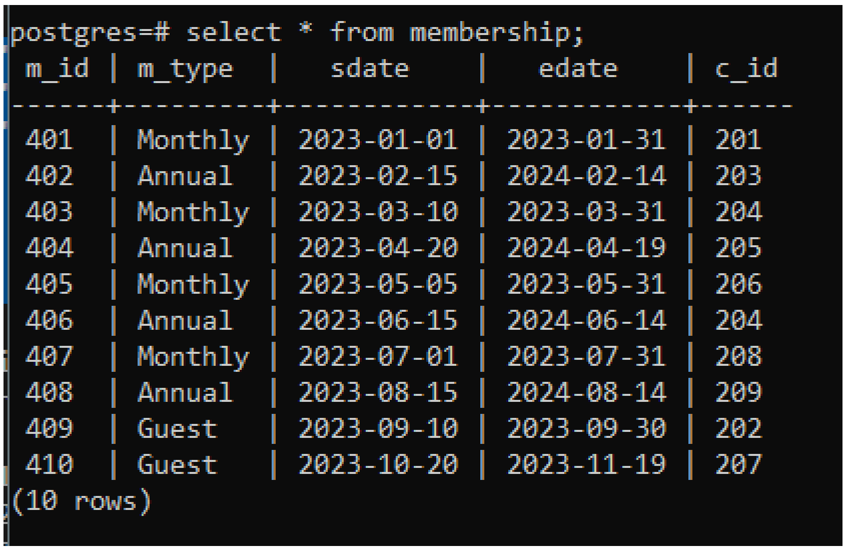
INSERTION OUTPUT:

5.2.1 Fascilities



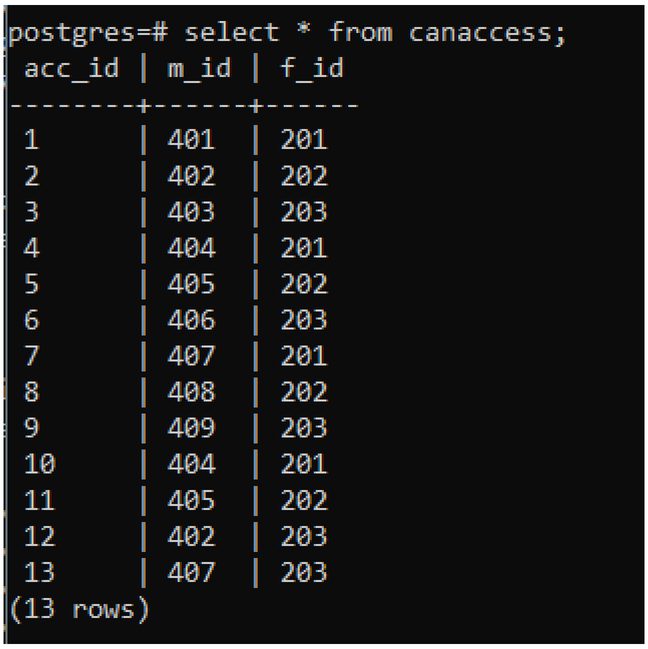
```
postgres=# SELECT * FROM Fascilities;
 f_id | restaurant
------+------------
 201  | A
 202  | B
 203  | C
(3 rows)
```

5.2.2 Membership



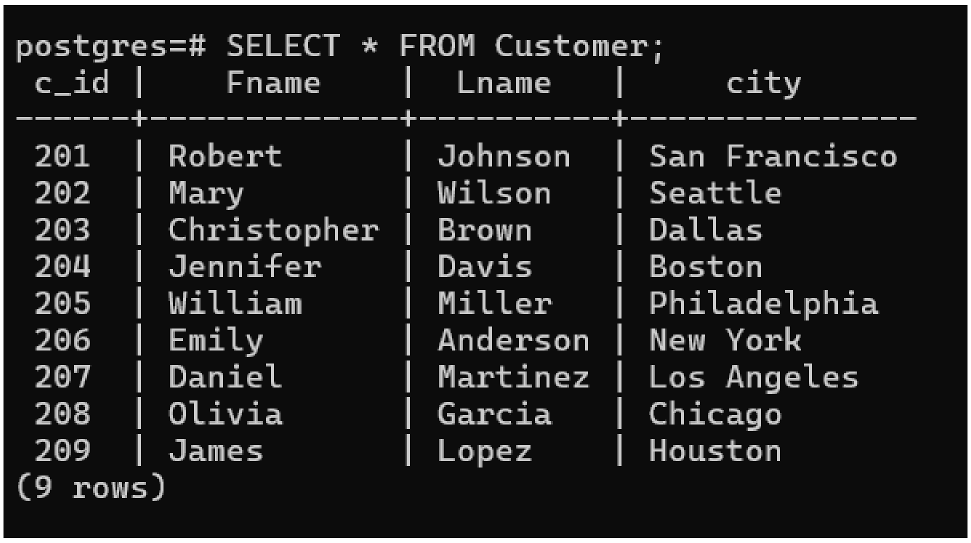
```
postgres=# select * from membership;
 m_id | m_type  |   sdate    |   edate    | c_id
------+---------+------------+------------+------
 401  | Monthly | 2023-01-01 | 2023-01-31 | 201
 402  | Annual  | 2023-02-15 | 2024-02-14 | 203
 403  | Monthly | 2023-03-10 | 2023-03-31 | 204
 404  | Annual  | 2023-04-20 | 2024-04-19 | 205
 405  | Monthly | 2023-05-05 | 2023-05-31 | 206
 406  | Annual  | 2023-06-15 | 2024-06-14 | 204
 407  | Monthly | 2023-07-01 | 2023-07-31 | 208
 408  | Annual  | 2023-08-15 | 2024-08-14 | 209
 409  | Guest   | 2023-09-10 | 2023-09-30 | 202
 410  | Guest   | 2023-10-20 | 2023-11-19 | 207
(10 rows)
```

5.2.3 Canaccess



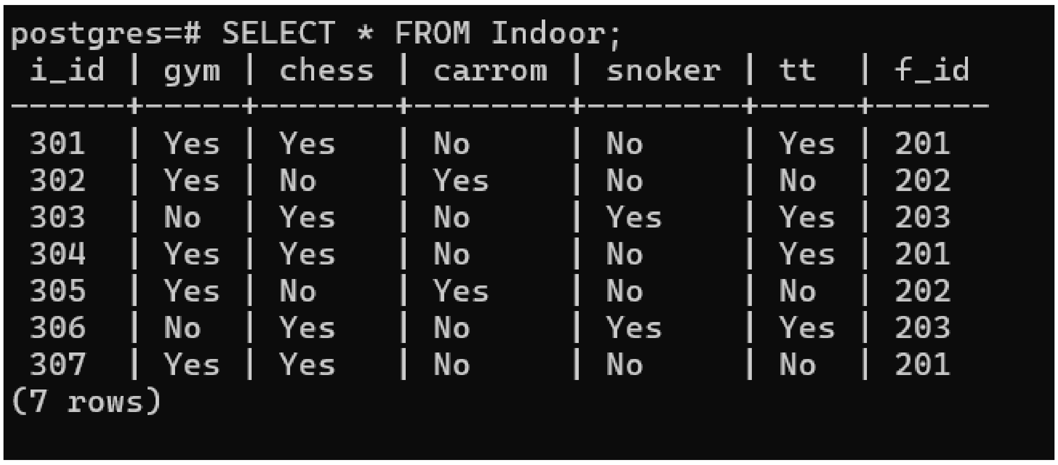
```
postgres=# select * from canaccess;
 acc_id | m_id | f_id
--------+------+------
 1      | 401  | 201
 2      | 402  | 202
 3      | 403  | 203
 4      | 404  | 201
 5      | 405  | 202
 6      | 406  | 203
 7      | 407  | 201
 8      | 408  | 202
 9      | 409  | 203
 10     | 404  | 201
 11     | 405  | 202
 12     | 402  | 203
 13     | 407  | 203
(13 rows)
```

5.2.4 Customer



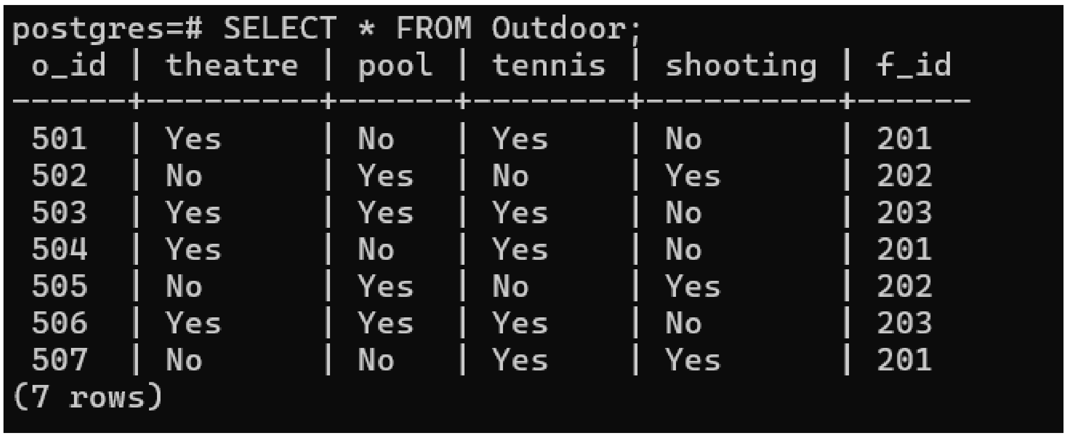
```
postgres=# SELECT * FROM Customer;
 c_id |     Fname     |  Lname    |     city
------+-------------+----------+---------------
 201  | Robert        | Johnson  | San Francisco
 202  | Mary          | Wilson   | Seattle
 203  | Christopher | Brown    | Dallas
 204  | Jennifer      | Davis    | Boston
 205  | William       | Miller   | Philadelphia
 206  | Emily         | Anderson | New York
 207  | Daniel        | Martinez | Los Angeles
 208  | Olivia        | Garcia   | Chicago
 209  | James         | Lopez    | Houston
(9 rows)
```

5.2.5 Indoor



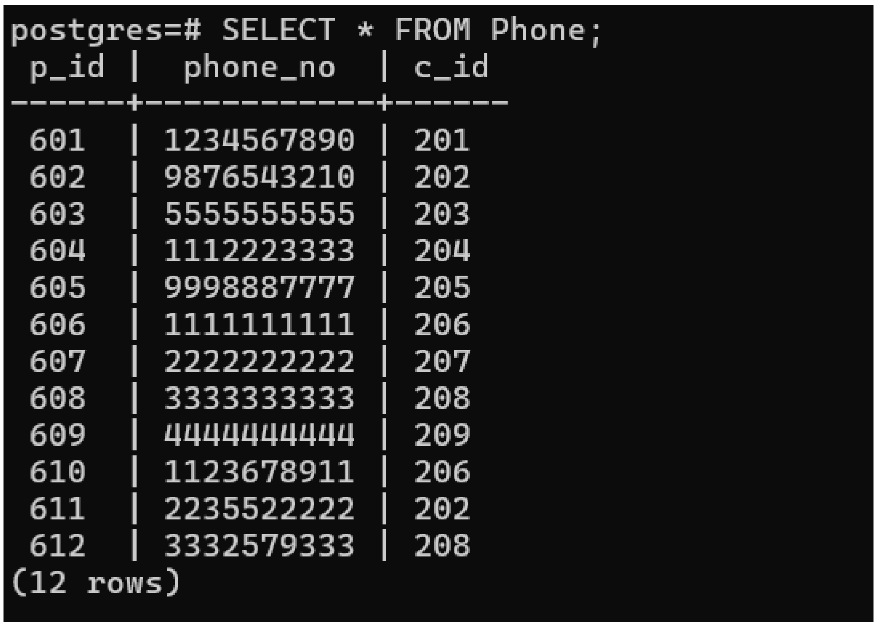
```
postgres=# SELECT * FROM Indoor;
 i_id | gym | chess | carrom | snoker | tt  | f_id
------+-----+-------+--------+--------+-----+------
 301  | Yes | Yes   | No     | No     | Yes | 201
 302  | Yes | No    | Yes    | No     | No  | 202
 303  | No  | Yes   | No     | Yes    | Yes | 203
 304  | Yes | Yes   | No     | No     | Yes | 201
 305  | Yes | No    | Yes    | No     | No  | 202
 306  | No  | Yes   | No     | Yes    | Yes | 203
 307  | Yes | Yes   | No     | No     | No  | 201
(7 rows)
```

5.2.6 Outdoor

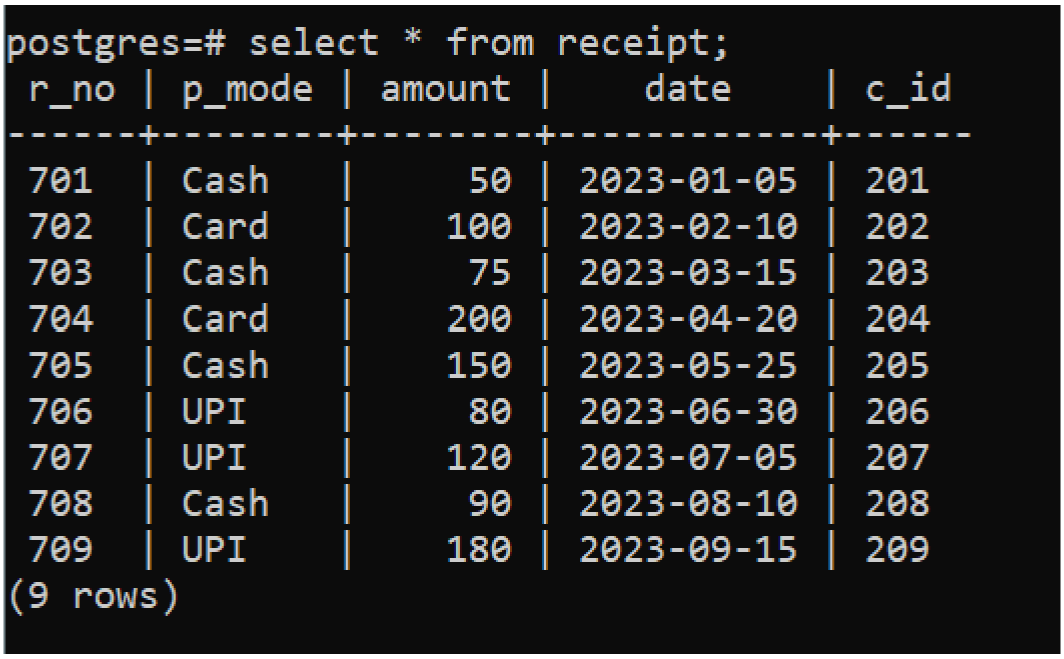


```
postgres=# SELECT * FROM Outdoor;
 o_id | theatre | pool | tennis | shooting | f_id
------+---------+------+--------+----------+------
 501  | Yes     | No   | Yes    | No       | 201
 502  | No      | Yes  | No     | Yes      | 202
 503  | Yes     | Yes  | Yes    | No       | 203
 504  | Yes     | No   | Yes    | No       | 201
 505  | No      | Yes  | No     | Yes      | 202
 506  | Yes     | Yes  | Yes    | No       | 203
 507  | No      | No   | Yes    | Yes      | 201
(7 rows)
```

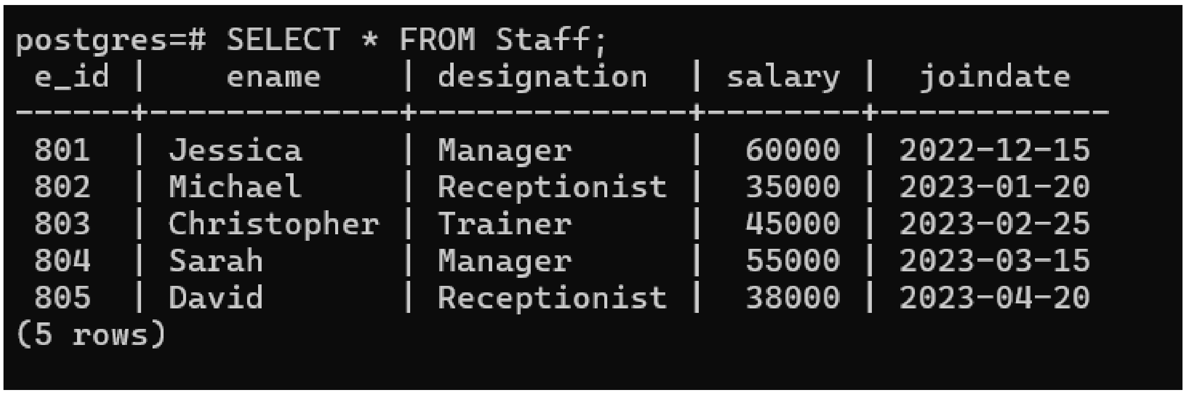
5.2.7 Phone


```
postgres=# SELECT * FROM Phone;
 p_id |  phone_no  | c_id
------+------------+------
 601  | 1234567890 | 201
 602  | 9876543210 | 202
 603  | 5555555555 | 203
 604  | 1112223333 | 204
 605  | 9998887777 | 205
 606  | 1111111111 | 206
 607  | 2222222222 | 207
 608  | 3333333333 | 208
 609  | 4444444444 | 209
 610  | 1123678911 | 206
 611  | 2235522222 | 202
 612  | 3332579333 | 208
(12 rows)
```

5.2.8 Receipt

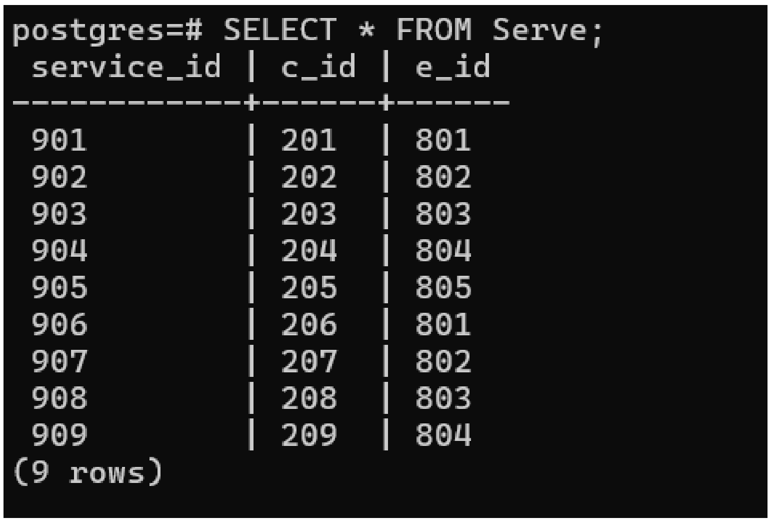

```
postgres=# select * from receipt;
 r_no | p_mode | amount |    date    | c_id
------+--------+--------+------------+------
 701  | Cash   |     50 | 2023-01-05 | 201
 702  | Card   |    100 | 2023-02-10 | 202
 703  | Cash   |     75 | 2023-03-15 | 203
 704  | Card   |    200 | 2023-04-20 | 204
 705  | Cash   |    150 | 2023-05-25 | 205
 706  | UPI    |     80 | 2023-06-30 | 206
 707  | UPI    |    120 | 2023-07-05 | 207
 708  | Cash   |     90 | 2023-08-10 | 208
 709  | UPI    |    180 | 2023-09-15 | 209
(9 rows)
```

5.2.9 Staff



```
postgres=# SELECT * FROM Staff;
 e_id |    ename     | designation  | salary |  joindate
------+-------------+--------------+--------+------------
 801  | Jessica     | Manager      |  60000 | 2022-12-15
 802  | Michael     | Receptionist |  35000 | 2023-01-20
 803  | Christopher | Trainer      |  45000 | 2023-02-25
 804  | Sarah       | Manager      |  55000 | 2023-03-15
 805  | David       | Receptionist |  38000 | 2023-04-20
(5 rows)
```

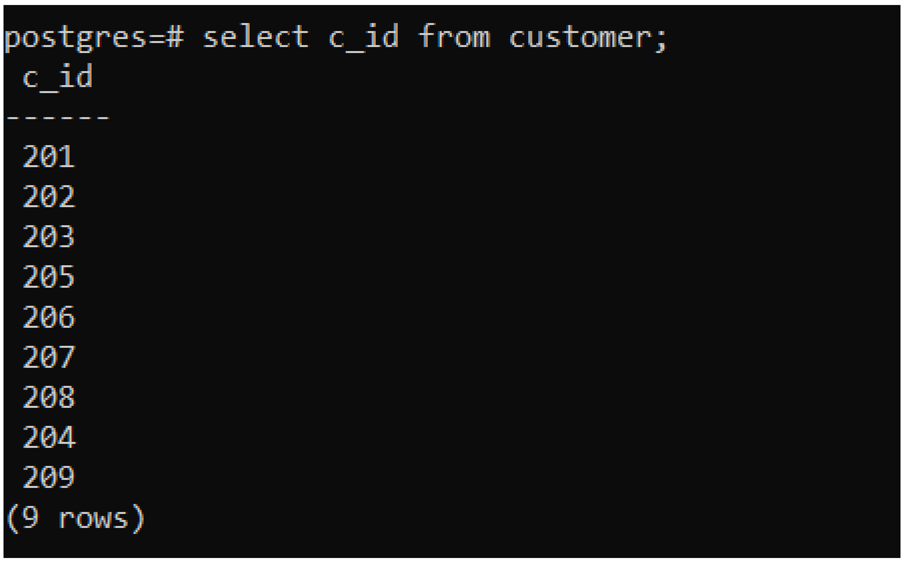
5.2.10 Serve



```
postgres=# SELECT * FROM Serve;
 service_id | c_id | e_id
------------+------+------
 901        | 201  | 801
 902        | 202  | 802
 903        | 203  | 803
 904        | 204  | 804
 905        | 205  | 805
 906        | 206  | 801
 907        | 207  | 802
 908        | 208  | 803
 909        | 209  | 804
(9 rows)
```

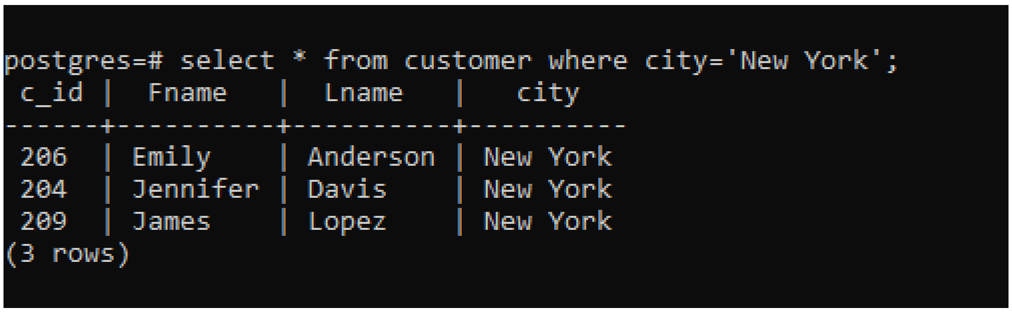
5.3 QUERIES USING BASIC DBMS CONSTRUCTS JOIN & SUBQUERIES:

5.3.1 Display Customer id of all customer



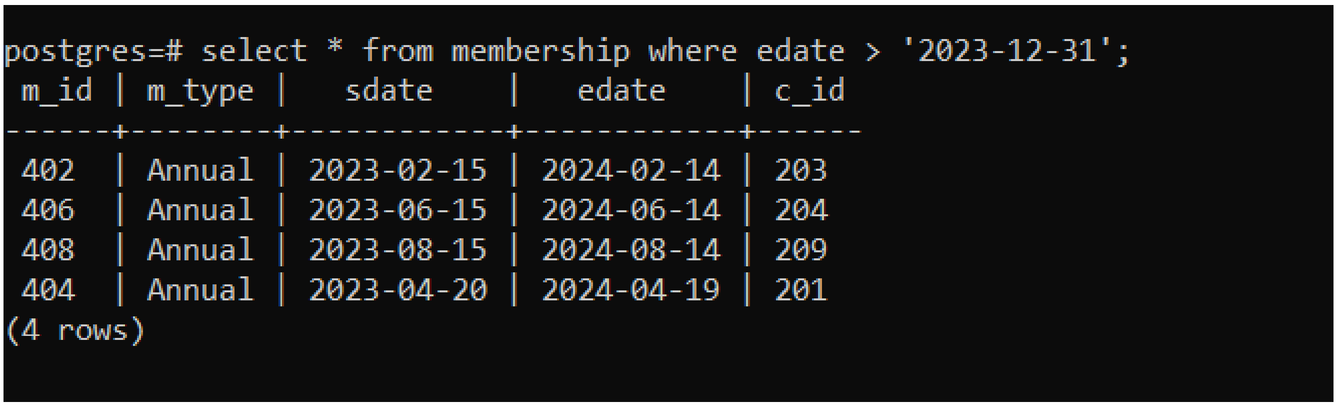
```
postgres=# select c_id from customer;
 c_id
------
 201
 202
 203
 205
 206
 207
 208
 204
 209
(9 rows)
```

5.3.2 Display customer name who lives in New York.



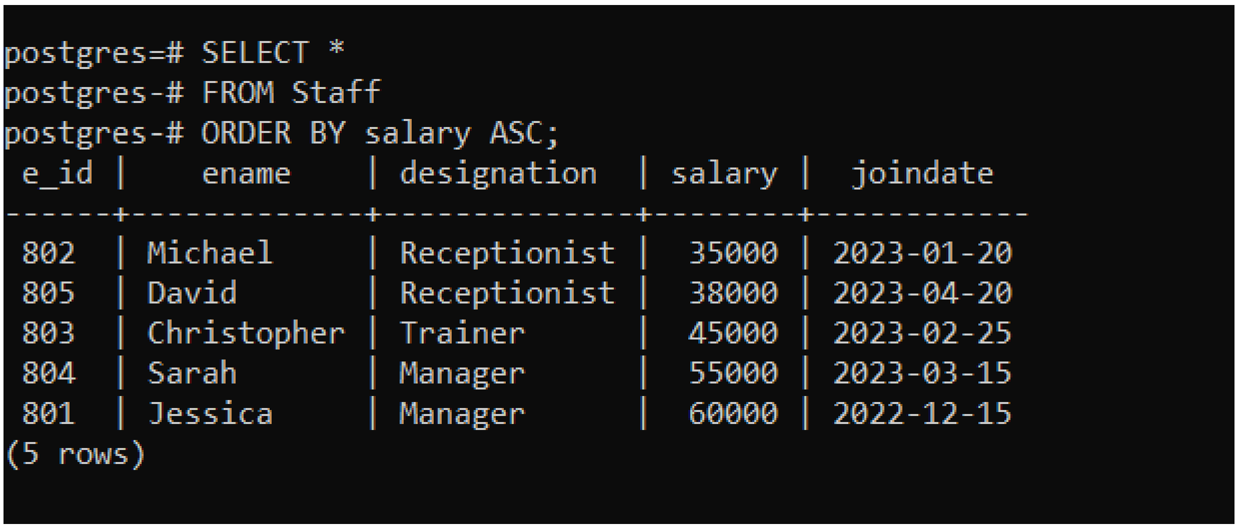
```
postgres=# select * from customer where city='New York';
 c_id |  Fname    |  Lname    |   city
------+----------+----------+----------
 206  | Emily    | Anderson | New York
 204  | Jennifer | Davis    | New York
 209  | James    | Lopez    | New York
(3 rows)
```

5.3.3 Display members whoes membership will be continued in 2024



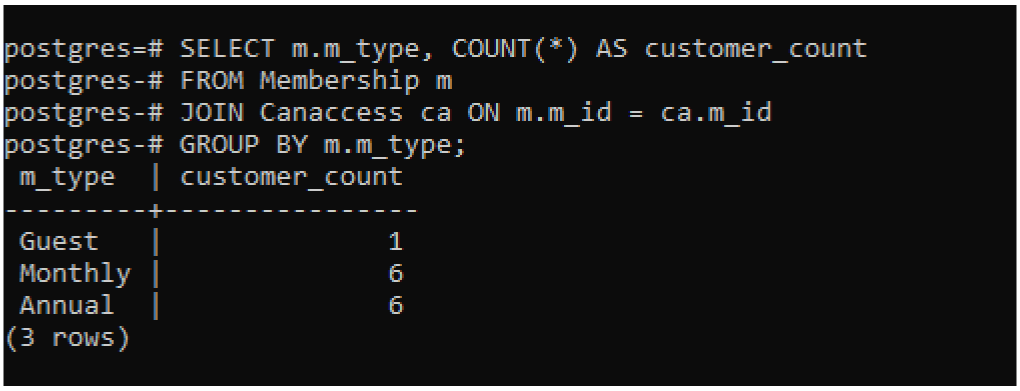
```
postgres=# select * from membership where edate > '2023-12-31';
 m_id | m_type |   sdate    |   edate    | c_id
------+--------+------------+------------+------
 402  | Annual | 2023-02-15 | 2024-02-14 | 203
 406  | Annual | 2023-06-15 | 2024-06-14 | 204
 408  | Annual | 2023-08-15 | 2024-08-14 | 209
 404  | Annual | 2023-04-20 | 2024-04-19 | 201
(4 rows)
```

5.3.4 Display employee details in ascending order by their salary.

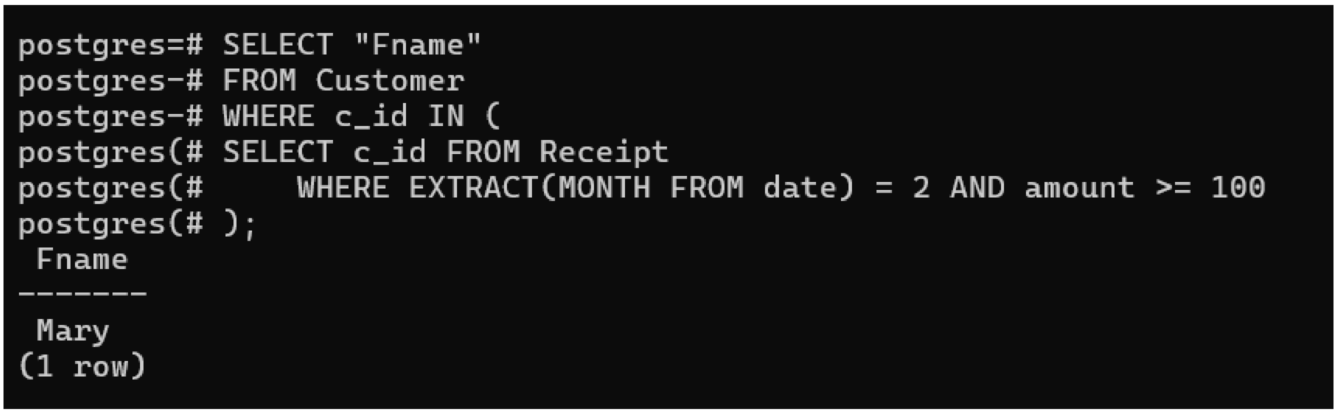


```
postgres=# SELECT *
postgres-# FROM Staff
postgres-# ORDER BY salary ASC;
 e_id |    ename     | designation  | salary |  joindate
------+--------------+--------------+--------+------------
 802  | Michael      | Receptionist |  35000 | 2023-01-20
 805  | David        | Receptionist |  38000 | 2023-04-20
 803  | Christopher  | Trainer      |  45000 | 2023-02-25
 804  | Sarah        | Manager      |  55000 | 2023-03-15
 801  | Jessica      | Manager      |  60000 | 2022-12-15
(5 rows)
```

<u>5.3.5</u> Display the count of customers with different membership types.

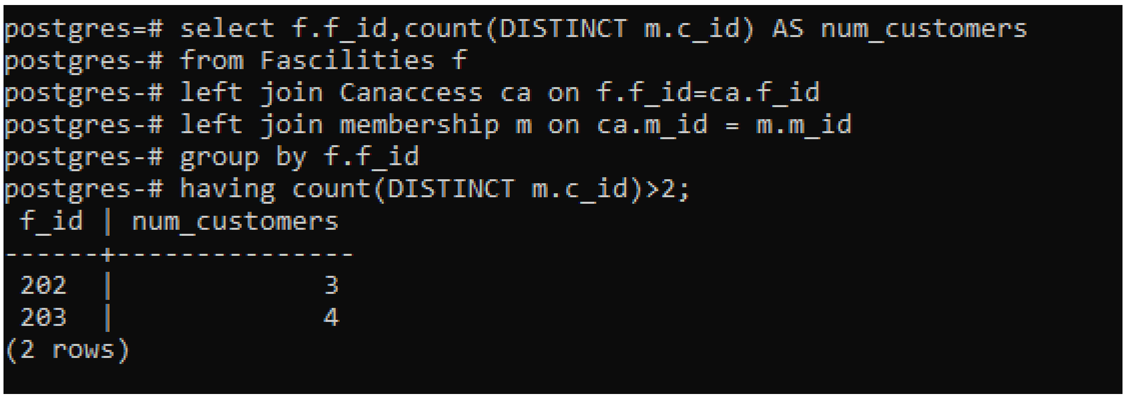

```
postgres=# SELECT m.m_type, COUNT(*) AS customer_count
postgres-# FROM Membership m
postgres-# JOIN Canaccess ca ON m.m_id = ca.m_id
postgres-# GROUP BY m.m_type;
 m_type  | customer_count
---------+----------------
 Guest   |              1
 Monthly |              6
 Annual  |              6
(3 rows)
```

5.3.6 Display name of customer who had done payment in February with payment amount more than 100.



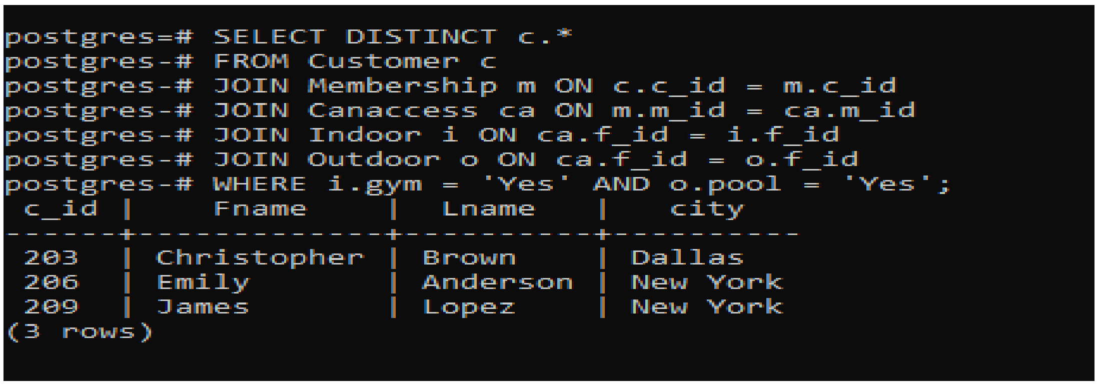
```
postgres=# SELECT "Fname"
postgres-# FROM Customer
postgres-# WHERE c_id IN (
postgres(# SELECT c_id FROM Receipt
postgres(#     WHERE EXTRACT(MONTH FROM date) = 2 AND amount >= 100
postgres(# );
 Fname
-------
 Mary
(1 row)
```

5.3.7 Display facilities used by more than 2 customers.



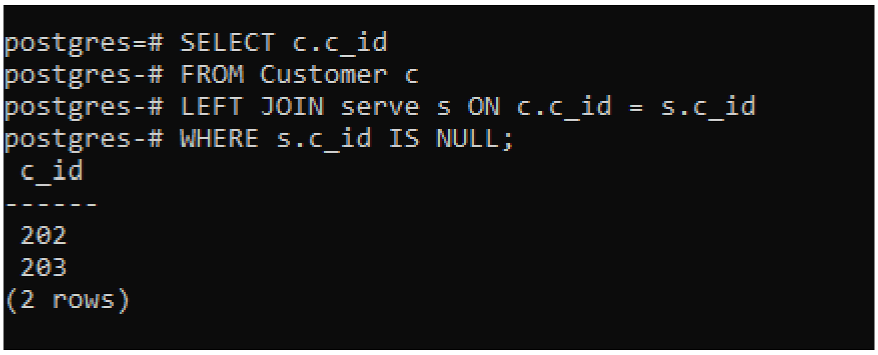
```
postgres=# select f.f_id,count(DISTINCT m.c_id) AS num_customers
postgres-# from Fascilities f
postgres-# left join Canaccess ca on f.f_id=ca.f_id
postgres-# left join membership m on ca.m_id = m.m_id
postgres-# group by f.f_id
postgres-# having count(DISTINCT m.c_id)>2;
 f_id | num_customers
------+---------------
 202  |             3
 203  |             4
(2 rows)
```

5.3.8 Display customer details who can access gym and pool both.



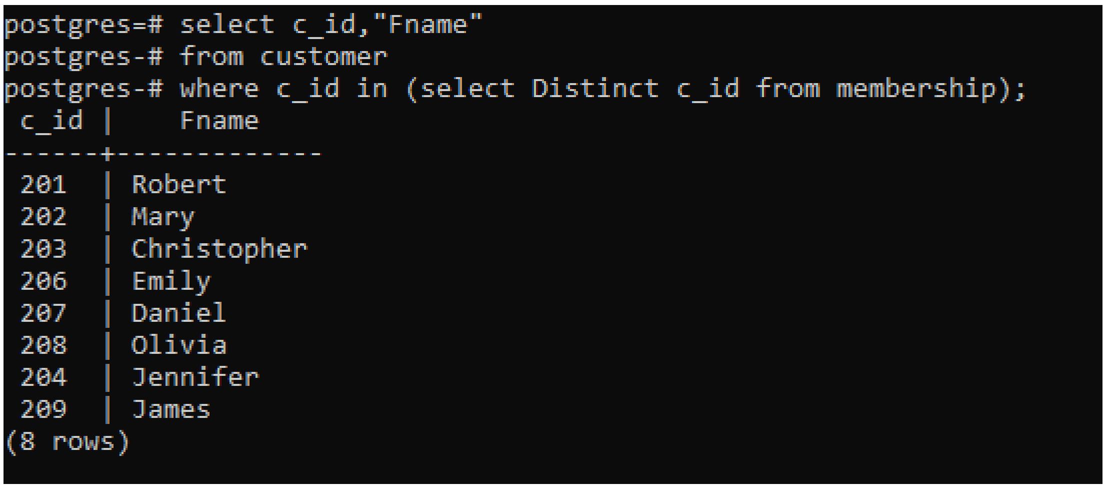
```
postgres=# SELECT DISTINCT c.*
postgres-# FROM Customer c
postgres-# JOIN Membership m ON c.c_id = m.c_id
postgres-# JOIN Canaccess ca ON m.m_id = ca.m_id
postgres-# JOIN Indoor i ON ca.f_id = i.f_id
postgres-# JOIN Outdoor o ON ca.f_id = o.f_id
postgres-# WHERE i.gym = 'Yes' AND o.pool = 'Yes';
 c_id |     Fname     |   Lname    |    city
------+---------------+------------+-----------
 203  | Christopher   | Brown      | Dallas
 206  | Emily         | Anderson   | New York
 209  | James         | Lopez      | New York
(3 rows)
```

5.4.9 Display customers detail which is not served yet by any staff members.



```
postgres=# SELECT c.c_id
postgres-# FROM Customer c
postgres-# LEFT JOIN serve s ON c.c_id = s.c_id
postgres-# WHERE s.c_id IS NULL;
 c_id
------
 202
 203
(2 rows)
```

5.4.10 Display all customers who have got their membership.



```
postgres=# select c_id,"Fname"
postgres-# from customer
postgres-# where c_id in (select Distinct c_id from membership);
 c_id |    Fname
------+-------------
 201  | Robert
 202  | Mary
 203  | Christopher
 206  | Emily
 207  | Daniel
 208  | Olivia
 204  | Jennifer
 209  | James
(8 rows)
```

5.5 FUNCTION & TRIGGERS:

5.5.1 Create a trigger for changes in the customer details

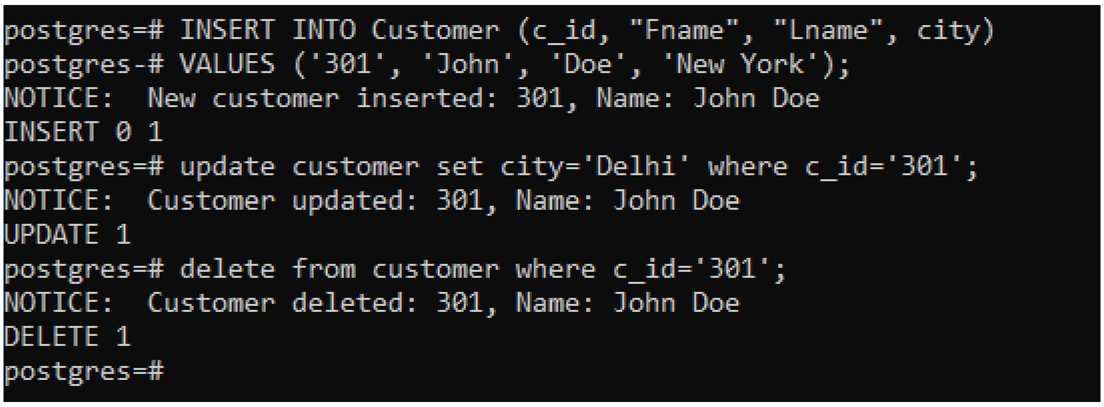
Function:

```
CREATE OR REPLACE FUNCTION customer_trigger_function()
RETURNS TRIGGER AS
$$
BEGIN
  IF TG_OP = 'INSERT' THEN
    -- Log insertion
    RAISE NOTICE 'New customer inserted: %, Name: % %', NEW.c_id, NEW."Fname", NEW."Lname";
  ELSIF TG_OP = 'UPDATE' THEN
    -- Log update
    RAISE NOTICE 'Customer updated: %, Name: % %', NEW.c_id, NEW."Fname", NEW."Lname";
  ELSIF TG_OP = 'DELETE' THEN
    -- Log deletion
    RAISE NOTICE 'Customer deleted: %, Name: % %', OLD.c_id, OLD."Fname", OLD."Lname";
  END IF;
  RETURN NULL;
END;
$$
LANGUAGE plpgsql;
```

Trigger:

```
CREATE TRIGGER customer_trigger
AFTER INSERT OR UPDATE OR DELETE
ON Customer
FOR EACH ROW
EXECUTE FUNCTION customer_trigger_function();
```

Checking



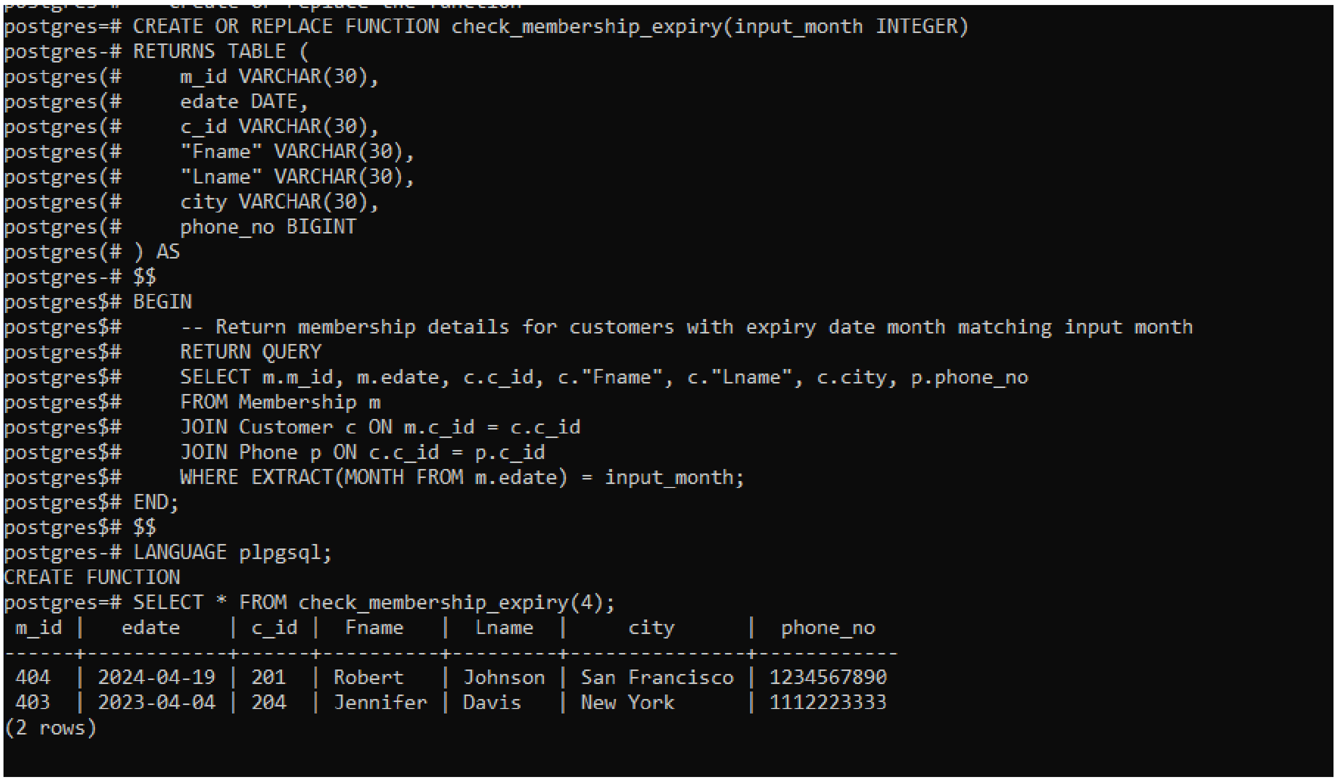
```
postgres=# INSERT INTO Customer (c_id, "Fname", "Lname", city)
postgres-# VALUES ('301', 'John', 'Doe', 'New York');
NOTICE:  New customer inserted: 301, Name: John Doe
INSERT 0 1
postgres=# update customer set city='Delhi' where c_id='301';
NOTICE:  Customer updated: 301, Name: John Doe
UPDATE 1
postgres=# delete from customer where c_id='301';
NOTICE:  Customer deleted: 301, Name: John Doe
DELETE 1
postgres=#
```

5.5.2 Create a function to show customer details whose membership expires in user defined month .

Function:

```
CREATE OR REPLACE FUNCTION check_membership_expiry(input_month INTEGER)
RETURNS TABLE (
   m_id VARCHAR(30),
   edate DATE,
   c_id VARCHAR(30),
   "Fname" VARCHAR(30),
   "Lname" VARCHAR(30),
   city VARCHAR(30),
   phone_no BIGINT
) AS
$$
BEGIN
   -- Return membership details for customers with expiry date month matching input month
   RETURN QUERY
   SELECT m.m_id, m.edate, c.c_id, c."Fname", c."Lname", c.city, p.phone_no
   FROM Membership m
   JOIN Customer c ON m.c_id = c.c_id
   JOIN Phone p ON c.c_id = p.c_id
   WHERE EXTRACT(MONTH FROM m.edate) = input_month;
END;
$$
LANGUAGE plpgsql;
```

Check: SELECT * FROM check_membership_expiry(4);



```
postgres=# CREATE OR REPLACE FUNCTION check_membership_expiry(input_month INTEGER)
postgres-# RETURNS TABLE (
postgres(#     m_id VARCHAR(30),
postgres(#     edate DATE,
postgres(#     c_id VARCHAR(30),
postgres(#     "Fname" VARCHAR(30),
postgres(#     "Lname" VARCHAR(30),
postgres(#     city VARCHAR(30),
postgres(#     phone_no BIGINT
postgres(# ) AS
postgres-# $$
postgres$# BEGIN
postgres$#     -- Return membership details for customers with expiry date month matching input month
postgres$#     RETURN QUERY
postgres$#     SELECT m.m_id, m.edate, c.c_id, c."Fname", c."Lname", c.city, p.phone_no
postgres$#     FROM Membership m
postgres$#     JOIN Customer c ON m.c_id = c.c_id
postgres$#     JOIN Phone p ON c.c_id = p.c_id
postgres$#     WHERE EXTRACT(MONTH FROM m.edate) = input_month;
postgres$# END;
postgres$# $$
postgres-# LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# SELECT * FROM check_membership_expiry(4);
 m_id |   edate    | c_id |  Fname   |  Lname  |     city      |  phone_no
------+------------+------+----------+---------+---------------+------------
 404  | 2024-04-19 | 201  | Robert   | Johnson | San Francisco | 1234567890
 403  | 2023-04-04 | 204  | Jennifer | Davis   | New York      | 1112223333
(2 rows)
```

5.5.3 Create a trigger for updation of employ salary which shows the difference of salary.

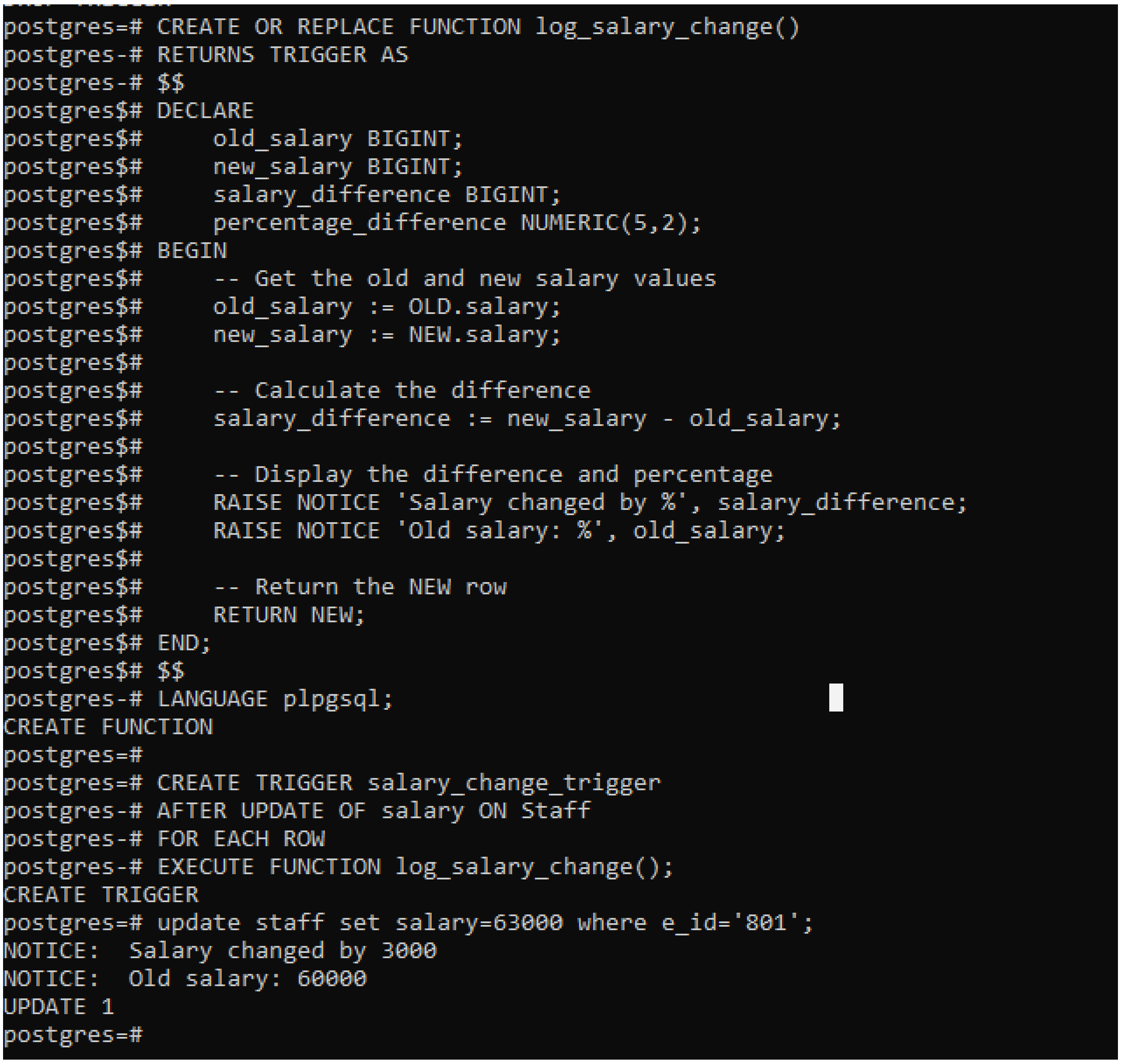
Function :

```
CREATE OR REPLACE FUNCTION log_salary_change()
RETURNS TRIGGER AS
$$
DECLARE
  old_salary BIGINT;
  new_salary BIGINT;
  salary_difference BIGINT;
  percentage_difference NUMERIC(5,2);
BEGIN
  -- Get the old and new salary values
  old_salary := OLD.salary;
  new_salary := NEW.salary;

  -- Calculate the difference
  salary_difference := new_salary - old_salary;

  -- Display the difference and percentage
  RAISE NOTICE 'Salary changed by %', salary_difference;
  RAISE NOTICE 'Old salary: %', old_salary;

  -- Return the NEW row
  RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Trigger:

```
CREATE TRIGGER salary_change_trigger
AFTER UPDATE OF salary ON Staff
FOR EACH ROW
EXECUTE FUNCTION log_salary_change();
```

Checking



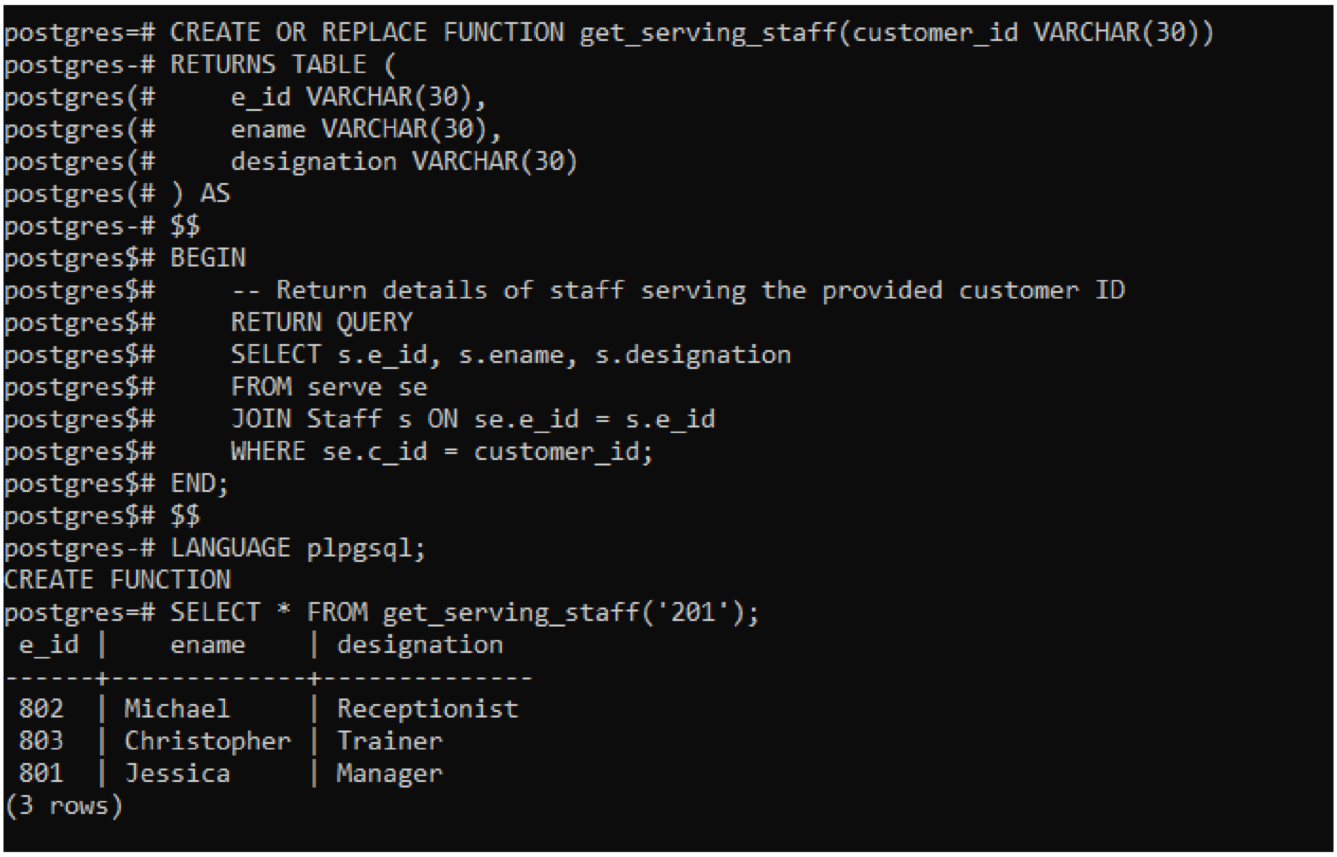
```
postgres=# CREATE OR REPLACE FUNCTION log_salary_change()
postgres-# RETURNS TRIGGER AS
postgres-# $$
postgres$# DECLARE
postgres$#     old_salary BIGINT;
postgres$#     new_salary BIGINT;
postgres$#     salary_difference BIGINT;
postgres$#     percentage_difference NUMERIC(5,2);
postgres$# BEGIN
postgres$#     -- Get the old and new salary values
postgres$#     old_salary := OLD.salary;
postgres$#     new_salary := NEW.salary;
postgres$#
postgres$#     -- Calculate the difference
postgres$#     salary_difference := new_salary - old_salary;
postgres$#
postgres$#     -- Display the difference and percentage
postgres$#     RAISE NOTICE 'Salary changed by %', salary_difference;
postgres$#     RAISE NOTICE 'Old salary: %', old_salary;
postgres$#
postgres$#     -- Return the NEW row
postgres$#     RETURN NEW;
postgres$# END;
postgres$# $$
postgres-# LANGUAGE plpgsql;
CREATE FUNCTION
postgres=#
postgres=# CREATE TRIGGER salary_change_trigger
postgres-# AFTER UPDATE OF salary ON Staff
postgres-# FOR EACH ROW
postgres-# EXECUTE FUNCTION log_salary_change();
CREATE TRIGGER
postgres=# update staff set salary=63000 where e_id='801';
NOTICE:  Salary changed by 3000
NOTICE:  Old salary: 60000
UPDATE 1
postgres=#
```

5.5.4 Create a function which list the staff details which served the customer with id determined by user.

Function :

```
CREATE OR REPLACE FUNCTION get_serving_staff(customer_id VARCHAR(30))
RETURNS TABLE (
   e_id VARCHAR(30),
   ename VARCHAR(30),
   designation VARCHAR(30)
) AS
$$
BEGIN
   -- Return details of staff serving the provided customer ID
   RETURN QUERY
   SELECT s.e_id, s.ename, s.designation
   FROM serve se
   JOIN Staff s ON se.e_id = s.e_id
   WHERE se.c_id = customer_id;
END;
$$
LANGUAGE plpgsql;
```

Check: SELECT * FROM get_serving_staff('201');



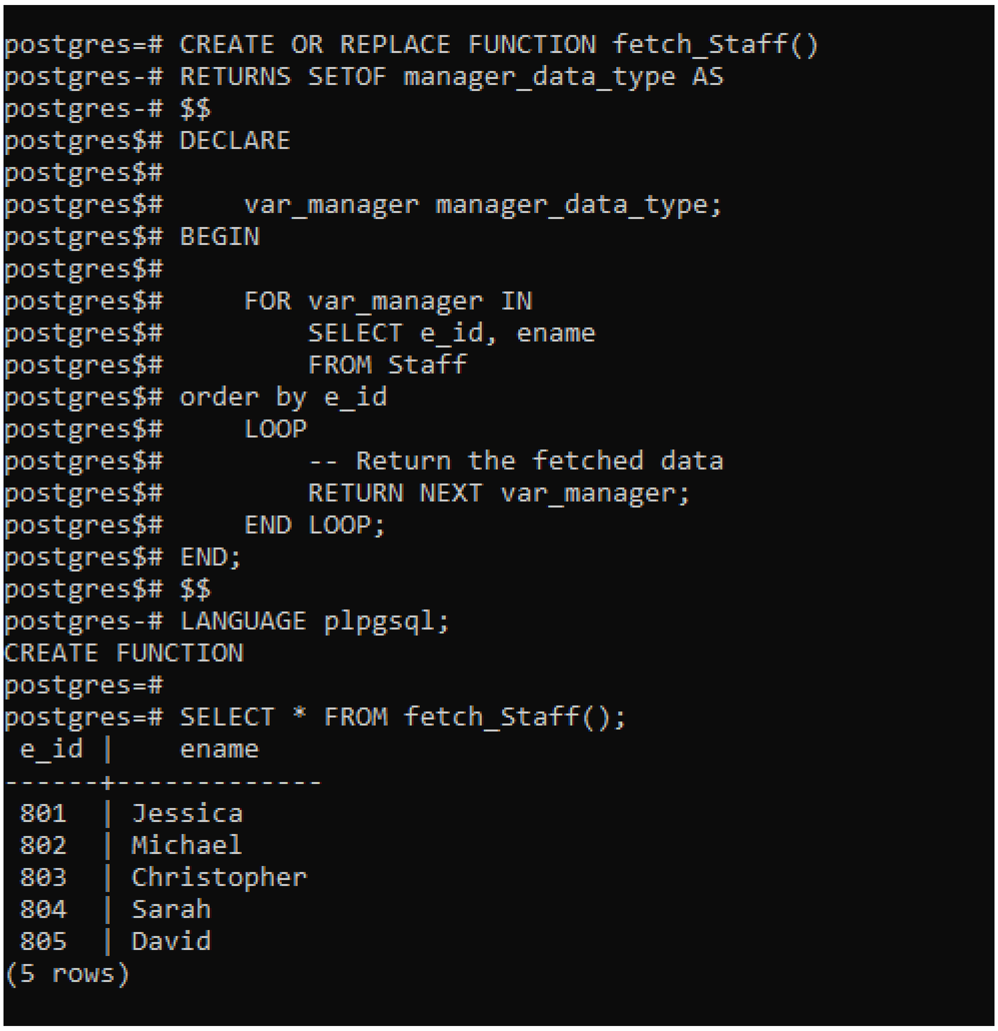
```
postgres=# CREATE OR REPLACE FUNCTION get_serving_staff(customer_id VARCHAR(30))
postgres-# RETURNS TABLE (
postgres(#     e_id VARCHAR(30),
postgres(#     ename VARCHAR(30),
postgres(#     designation VARCHAR(30)
postgres(# ) AS
postgres-# $$
postgres$# BEGIN
postgres$#     -- Return details of staff serving the provided customer ID
postgres$#     RETURN QUERY
postgres$#     SELECT s.e_id, s.ename, s.designation
postgres$#     FROM serve se
postgres$#     JOIN Staff s ON se.e_id = s.e_id
postgres$#     WHERE se.c_id = customer_id;
postgres$# END;
postgres$# $$
postgres-# LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# SELECT * FROM get_serving_staff('201');
 e_id |    ename    | designation
------+-------------+--------------
 802  | Michael     | Receptionist
 803  | Christopher | Trainer
 801  | Jessica     | Manager
(3 rows)
```

5.6 CURSOR:

Create a cursor to display the names of employes.

```
CREATE OR REPLACE FUNCTION fetch_Staff()
RETURNS SETOF manager_data_type AS
$$
DECLARE

   var_manager manager_data_type;
BEGIN

   FOR var_manager IN
      SELECT e_id, ename
      FROM Staff
          order by e_id
   LOOP
      -- Return the fetched data
      RETURN NEXT var_manager;
   END LOOP;
END;
$$
LANGUAGE plpgsql;
```

Check: SELECT * FROM fetch_Staff();



```
postgres=# CREATE OR REPLACE FUNCTION fetch_Staff()
postgres-# RETURNS SETOF manager_data_type AS
postgres-# $$
postgres$# DECLARE
postgres$#
postgres$#     var_manager manager_data_type;
postgres$# BEGIN
postgres$#
postgres$#     FOR var_manager IN
postgres$#         SELECT e_id, ename
postgres$#         FROM Staff
postgres$# order by e_id
postgres$#     LOOP
postgres$#         -- Return the fetched data
postgres$#         RETURN NEXT var_manager;
postgres$#     END LOOP;
postgres$# END;
postgres$# $$
postgres-# LANGUAGE plpgsql;
CREATE FUNCTION
postgres=#
postgres=# SELECT * FROM fetch_Staff();
 e_id |    ename
------+-------------
 801  | Jessica
 802  | Michael
 803  | Christopher
 804  | Sarah
 805  | David
(5 rows)
```