PES UNIVERSITY

100 feet Ring Road, BSK 3rd Stage, Bengaluru 560085

Department of Computer Science and Engineering
Jan – May 2020

UE18CS252
Database Management Systems

Project Report

BEAUTY PARLOR MANAGEMENT SYSTEM

PROJECT SUMMARY

Problem statement: **Beautyparlor database management system**.
This project is beneficial to beauty parlours in keeping customer data and promoting the services provided by them to public. There will be separate branch for men and women. Its functionality includes hiring staffs, offering different packages, taking care of appointments booked by customers and handling payments.
I have created 10 relational tables to distribute the data in the database based on relationships among them. I have modelled an e r diagram and then converted it to relational schema. I had to go through normalization process to overcome data insertion, deletion and update anomalies to decide the tables to be created. I have created 2 triggers for update operation considering staff and customer table of the database. There are functional dependencies among the keys in every tables. And every table contains a unique called primary key. I have made use of general data types like integers, date and time, characters and strings.
 I do believe that this model, with proper improvisations, will be beneficial to a lot of new parlour management systems.

Introduction

The scenario here is of a beauty parlour management system.

Relational tables:

There are 10 relational tables which describe the data model. First is the **parlour table**, which has 3 attributes namely branch code, branch venue and branch type. Branch code will be unique for every branch, and acts as primary key for the table. branch type refers to two different arena that is for men and women. And branch venue refers to the place where particular branch is located. Second table is the **owner's table** which contains details of the owner belonging to particular branch. Owner has attributes like first and last name, city, gender, and a unique attribute called o_id which designates to the branch which he owns. Similarly there are tables for **packages, services, staffs, staff skills, customers** and **appointments.**

Relationships:

Parlour owned by owner. this is a 1 to 1 relationship. Here, primary key of parlour acts as foreign key to owner attribute.

Parlour offers packages, is a 1 to many relationship. Since 1 parlour offers many packages. Here, primary key of parlour acts as foreign key for packages. Similarly, Parlour provides services is a 1 to many relationship. Since 1 parlour provides many services. Here, primary key of parlour acts as foreign key for services.

Parlour hires staffs. This is a 1 to n relationship since 1 parlour can hire as many staffs as it wants. Here, primary key of parlour acts as foreign key for staffs.

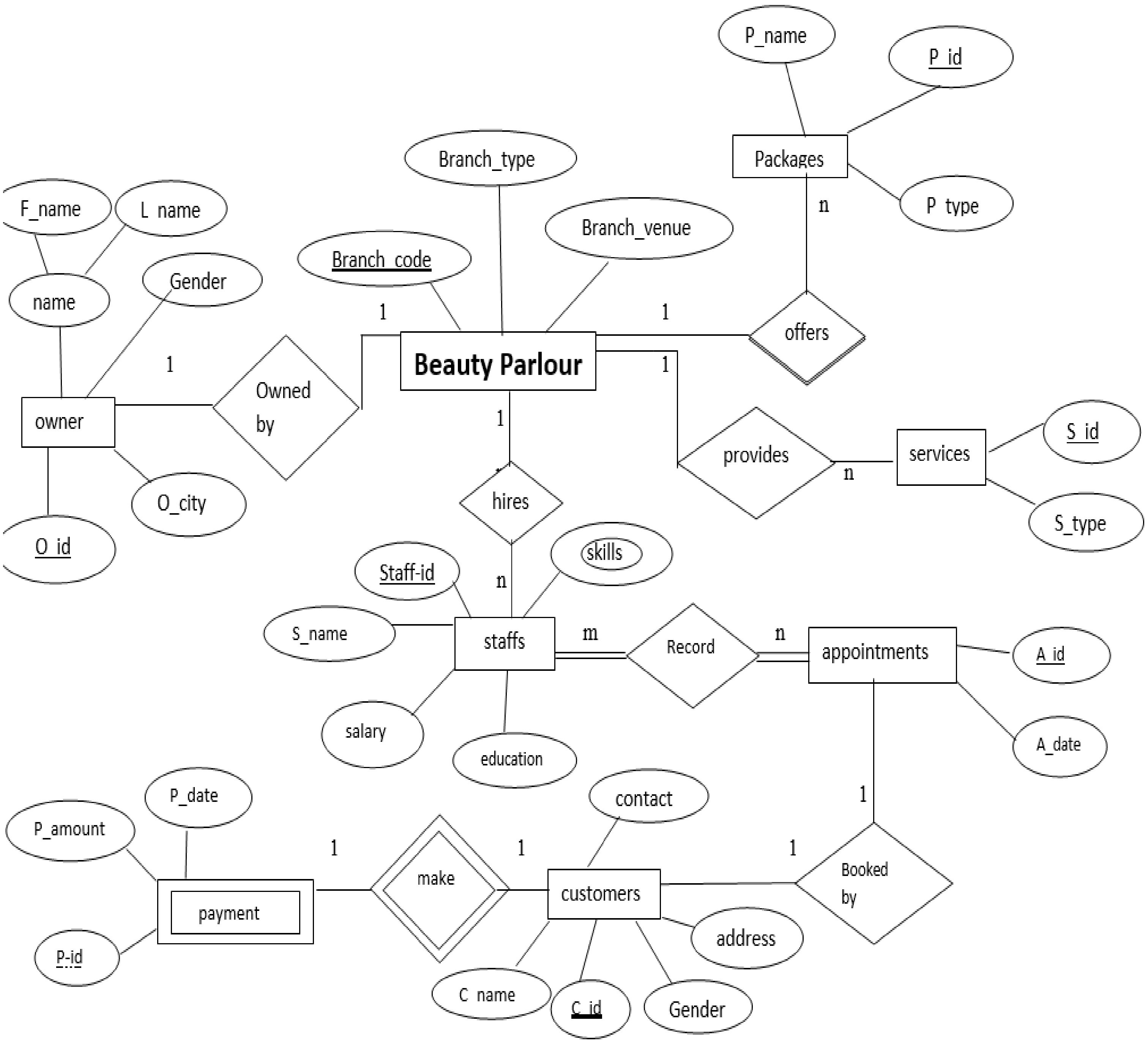
Staffs record appointments. This is a many to many relationship. Which has total participation from both entities. So we create a separate table with primary key of both the entities being considered. One major thing to note here is, staff has a multivalued attribute called skills, which requires a separate table with the primary key of staffs, and foreign key branch_code.

Appointments booked by customers. This is a 1 to 1 relationship. Here, primary key of customers act as foreign key for appointments.

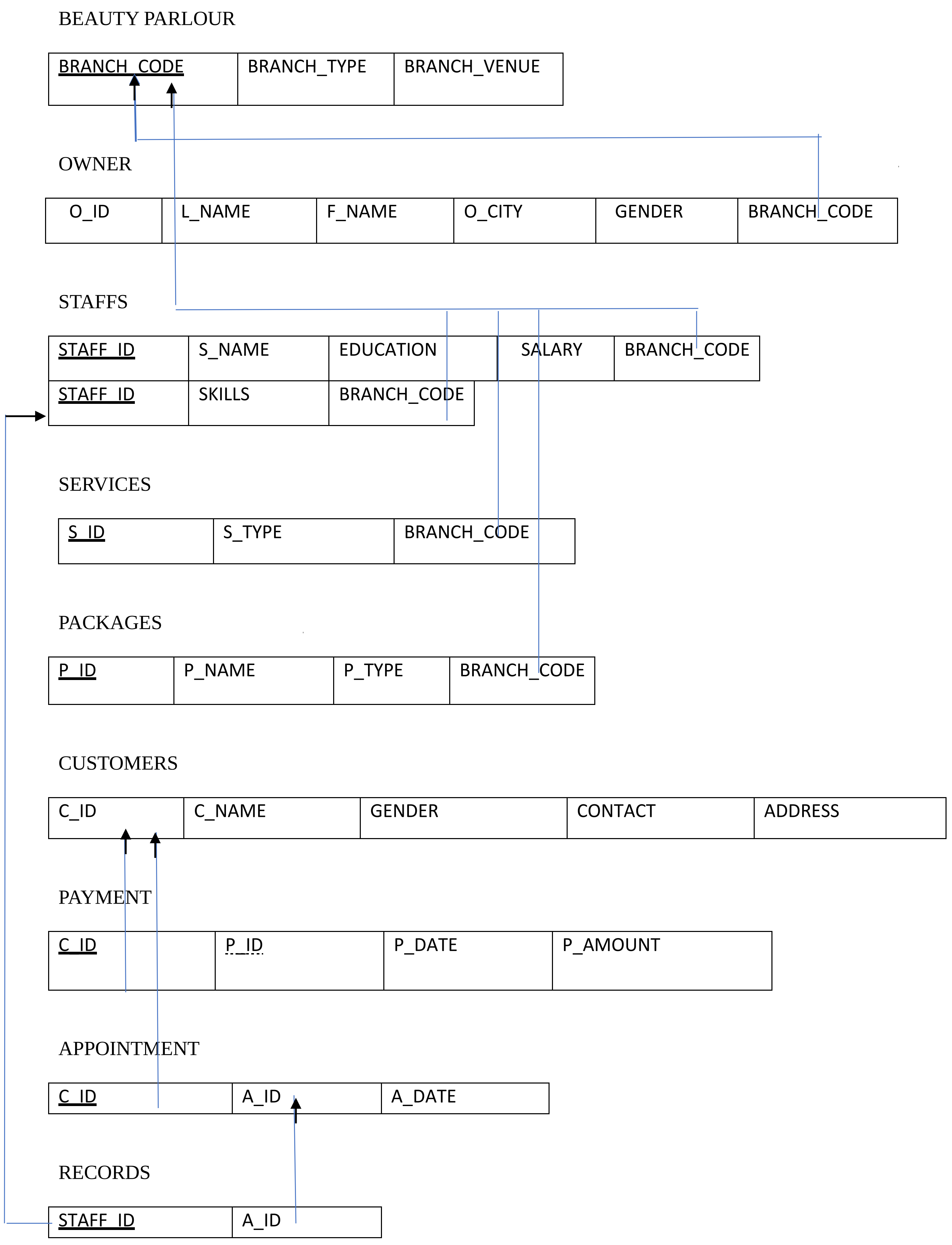
Customers make payments. This is a 1 to 1 relationship. And payment is a weak entity which does not contain any primary key. So customer id is being considered as primary key for the same.

Data Model

E R DIAGRAM



RELATIONAL SCHEMA



BEAUTY PARLOUR

BRANCH_CODE	BRANCH_TYPE	BRANCH_VENUE

OWNER

O_ID	L_NAME	F_NAME	O_CITY	GENDER	BRANCH_CODE

STAFFS

STAFF_ID	S_NAME	EDUCATION	SALARY	BRANCH_CODE

STAFF_ID	SKILLS	BRANCH_CODE

SERVICES

S_ID	S_TYPE	BRANCH_CODE

PACKAGES

P_ID	P_NAME	P_TYPE	BRANCH_CODE

CUSTOMERS

C_ID	C_NAME	GENDER	CONTACT	ADDRESS

PAYMENT

C_ID	P_ID	P_DATE	P_AMOUNT

APPOINTMENT

C_ID	A_ID	A_DATE

RECORDS

STAFF_ID	A_ID

Discussion on keys and datatypes used

Candidate keys

A Candidate Key is a column, or set of columns, in a table that can uniquely identify any database record without referring to any other data. Each table may have one or more candidate keys, but one candidate key is unique, and it is called the **primary key**. In our table beauty parlour, **branch code** is the primary key as it contains unique value.

Similarly **Staff id, customer id, service id, owner id, appointment id,** and **package id** are other primary keys present in our tables. The candidate key can be simple (having only one attribute) or composite as well.

Data types

A database data type refers to the format of data storage that can hold a distinct type or range of values. When computer stores data in variables, each variable must be designated a distinct data type. Some of the data types used in this beauty parlour database are, **integers, characters & strings, date and time** data types.

FD and Normalization

Identifying Functional Dependencies

Attribute of a table is said to be dependent on each other when an attribute of a table uniquely identifies another attribute of the same table. There are different types for functional dependencies like trivial, non-trivial, multi valued and transitive. We identify it depending on the context in which it is being used. For example, in this beauty parlour table**,**

branch_code →{ branch_venue, branch_type }**PRIMARY KEY** because the non key attributes branch_venue and branch_type are functionally dependent on branch_code.

Similarly, other functional dependencies are,
owner table:

- O_id→{fname, lname, gender, city}PRIMARY KEY
- Branch_code→{fname, lname, gender, city}FOREIGN KEY

Packages table:

- P_id→{p_name, p_type} PRIMARY KEY
- Branch_code→{p_name,p_type} FOREIGN KEY

Services table:

- S_id→ {s_type} PRIMARY KEY
- Branch_code→{s_type} FOREIGN KEY

Staffs table:

- Staff_id → {s_name, education , salary}PRIMARY KEY

- Staff_id→{skills} PRIMARY KEY
- Branch_code→{s_name,education,salary}FOREIGN KEY
- Branch_code→{skills} FOREIGN KEY

Customers table:

- C_id→{c_name, c_address, contact, gender,}PRIMARY KEY
- C_id→{p_id,p_amount,p_date} FOREIGN KEY

Appointments table:

- A_id→{a_date} PRIMARY KEY
- C_id→{a_date} FOREIGN KEY

So in general, functional dependency is a relation between the primary key and other non-key attributes within a table.

Normalization

Normalization is a process of organizing the data in a database. It is used to minimize the redundancy from a relation or set of relations.it is also used to eliminate undesirable characteristics like insertion, deletion and update anomalies.
Rules to define normal forms:

First normal form (1NF):

- each cell must be atomic valued.
- Entries in a column are of same type.
- Rows are uniquely identified.

Second normal form (2NF):

- It must be in first normal form.
- All non key columns are dependent on key attribute.

Third normal form (3NF):

- It must be in 1NF, 2NF.
- all fields can be determined only by the key in the table, not from any other columns.

Fourth normal form (4NF):

- It must be in 1NF,2NF,3NF.
- no non trivial multivalued dependencies other than candidate key.

Since I have developed E R model and converted them to relational schema, the relations I have obtained are normalized set of relations. So I will mention how these relations might have violated the normal form if not followed the rules. For example, for a relation **staffs record appointments**, 1NF would have been violated if staffs and appointments did not have a unique key called staff id and appointment id respectively.
Similarly, 2NF would have been violated if there was a column called customer name in appointments attribute, Which is not dependent on the key attribute appointment id. In the same way, 3NF would have been violated if there was another key called staff manager for the staffs, along with staff id. This causes the fields to be determined by two keys in the same column.

Some of the normal forms of our relation include,

Owned by(branch_code,o_id,branch_type,branch_venue,fname,lname,gender,city)
Normal form:
(branch_code,branch_type,branch_venue),
(o_id,fname, lname, gender, city,branch_code)

Offers(branch_code,p_id,branch_type,branch_venue,p_type,p_name)
Normal form:
(branch_code,branch_type,branch_venue),
(p_id,p_type,p_name,branch_code)

provides(branch_code,s_id,branch_type,branch_venue,s_type)
Normal form:
(branch_code,branch_type,branch_venue),
(s_id,s_type,branch_code)

records(staff_id,a_id,s_name,education,salary,skills,a_date)

Normal form:

(staff_id, s_name,education,salary,branch_code)

(staff_id,skills,branch_code)

(a_id,a_date,c_id)

(staff_id, a_id)

Makes(c_id,c_name,gender,contact,c_address,p_id,p_amount,p_date)

Normal form:

(c_id,p_id,p_amount,p_date)

(c_id,c_name,c_address,gender,contact)

booked by(c_id,c_name,c_address,gender,contact,a_id,a_date)

Normal form:

(c_id,c_name,c_address,gender,contact)

(a_id,a_date,c_id)

Lossless join property

if a relation R can be splited into 2 relationships R1 and R2 , then the natural join of R1 and R2 can get back relation R again. Therefore, there will be no loss of data or rows.

For example, In this beauty parlour database, if we apply join operation on table named beautyparlor and packages, we can see that no information will be lost as the table beautyparlor(branch_code,branch_venue,branch_type) and Packages(branch_code,p_id,p_type,p_name) have an attribute which is common and unique. Therefore there will be no loss of information.

This can be proved using NJB:

A decomposition D ={ R1,R2} of R has the lossless join property with respect to a set of functional dependencies F on R if and only if

The **FD((R1 ∩ R2) → (R1 – R2)) is in F.**

In our case, R1 is beautyparlor(branch_code,branch_venue,branch_type)

R2 is Packages(branch_code,p_id,p_type,p_name)

R1 ∩ R2 is branch_code

R2 – R1 is branch_venue, branch_type.

So clearly, branch_code→ branch_venue, branch_type . hence this decomposition has a lossless join property according to NJB.

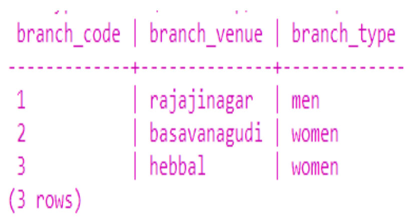
DDL

CREATE DATABASE beautyparlor;

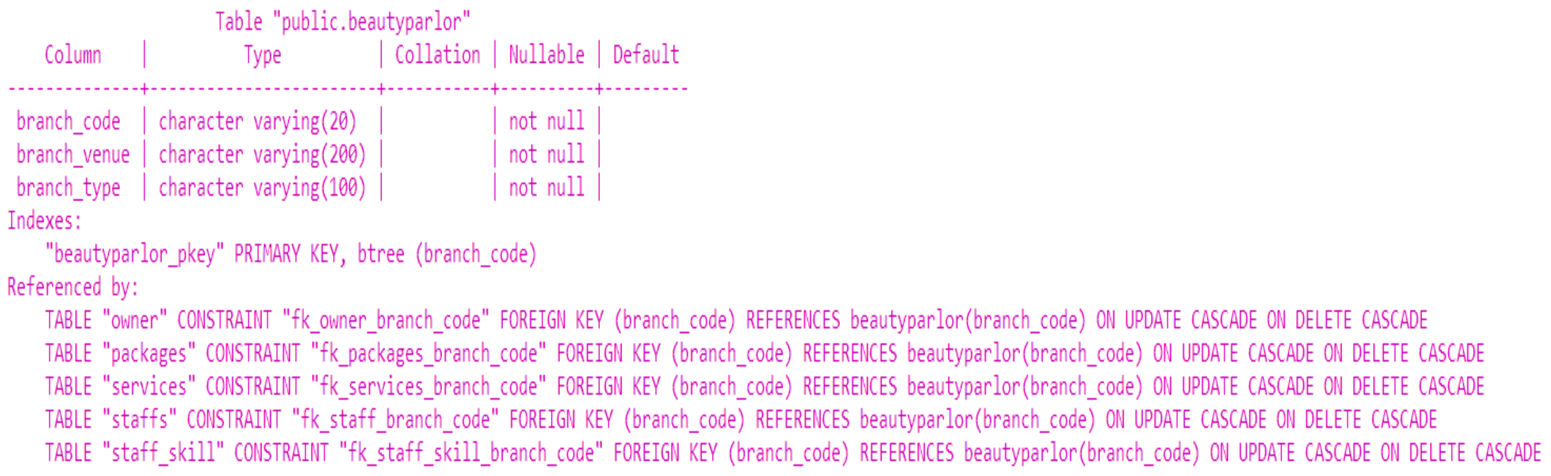
1. beautyparlor table

```
CREATE TABLE beautyparlor(
  branch_code varchar(20) PRIMARY KEY not null,
  branch_venue varchar(200) NOT NULL,
  branch_type varchar(100) NOT NULL
);

INSERT INTO beautyparlor(branch_code,branch_venue,branch_type)
VALUES
('1','rajajinagar','men'),
('2','basavanagudi','women'),
('3','hebbal','women');
```

OUTPUT:



```
 branch_code | branch_venue | branch_type
-------------+--------------+-------------
 1           | rajajinagar  | men
 2           | basavanagudi | women
 3           | hebbal       | women
(3 rows)
```



```
                  Table "public.beautyparlor"
    Column    |          Type          | Collation | Nullable | Default
--------------+------------------------+-----------+----------+---------
 branch_code  | character varying(20)  |           | not null |
 branch_venue | character varying(200) |           | not null |
 branch_type  | character varying(100) |           | not null |
Indexes:
    "beautyparlor_pkey" PRIMARY KEY, btree (branch_code)
Referenced by:
    TABLE "owner" CONSTRAINT "fk_owner_branch_code" FOREIGN KEY (branch_code) REFERENCES beautyparlor(branch_code) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "packages" CONSTRAINT "fk_packages_branch_code" FOREIGN KEY (branch_code) REFERENCES beautyparlor(branch_code) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "services" CONSTRAINT "fk_services_branch_code" FOREIGN KEY (branch_code) REFERENCES beautyparlor(branch_code) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "staffs" CONSTRAINT "fk_staff_branch_code" FOREIGN KEY (branch_code) REFERENCES beautyparlor(branch_code) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "staff_skill" CONSTRAINT "fk_staff_skill_branch_code" FOREIGN KEY (branch_code) REFERENCES beautyparlor(branch_code) ON UPDATE CASCADE ON DELETE CASCADE
```

2. owner table

```
CREATE TABLE owner(
   o_id varchar(20) primary key not null,
   fname varchar(40) NOT NULL,
   lname varchar(40)  not null,
   gender varchar(50) NOT NULL,
   o_city varchar(50) NOT NULL,
   branch_code varchar(20) not null
);

INSERT INTO owner(o_id,fname,lname,gender,o_city,branch_code)
VALUES
('100','john','ledger','male','mumbai','1'),
('101','reena','chopra','female','kolkata','3'),
('102','akshata','sharma','female','delhi','2');

ALTER TABLE owner
ADD CONSTRAINT fk_owner_branch_code
FOREIGN KEY (branch_code)
REFERENCES beautyparlor(branch_code)
ON DELETE CASCADE ON UPDATE CASCADE;
```

OUTPUT:

```
 o_id |  fname  | lname  | gender | o_city  | branch_code
------+---------+--------+--------+---------+-------------
 100  | john    | ledger | male   | mumbai  | 1
 101  | reena   | chopra | female | kolkata | 3
 102  | akshata | sharma | female | delhi   | 2
(3 rows)


                    Table "public.owner"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 o_id        | character varying(20) |           | not null |
 fname       | character varying(40) |           | not null |
 lname       | character varying(40) |           | not null |
 gender      | character varying(50) |           | not null |
 o_city      | character varying(50) |           | not null |
 branch_code | character varying(20) |           | not null |
Indexes:
    "owner_pkey" PRIMARY KEY, btree (o_id)
Foreign-key constraints:
    "fk_owner_branch_code" FOREIGN KEY (branch_code) REFERENCES beautyparlor(branch_code) ON UPDATE CASCADE ON DELETE CASCADE
```

3. staffs table

```
CREATE TABLE staffs(
    staff_id varchar(20) primary key not null,
    s_name varchar(50) not null,
    education varchar(60) not null,
    Salary varchar(50) not null,
    branch_code varchar(20) not null
 );

INSERT INTO staffs(staff_id,s_name,education,Salary,branch_code)
VALUES
('10','hithesh','b.sc','20,000','2'),
('20','heema','diploma','40,000','1'),
('30','bhushan','puc','10,000','1'),
('40','selena','B.E','80,000','3'),
('50','sushmita','bca','20,000','3'),
('60','imran','b.sc','20,000','3');

ALTER TABLE staffs
ADD CONSTRAINT fk_staff_branch_code
FOREIGN KEY (branch_code)
REFERENCES beautyparlor(branch_code)
ON DELETE CASCADE ON UPDATE CASCADE;
```

OUTPUT:

```
 staff_id |  s_name  | education | salary | branch_code
----------+----------+-----------+--------+-------------
 20       | heema    | diploma   | 40,000 | 1
 30       | bhushan  | puc       | 10,000 | 1
 40       | selena   | B.E       | 80,000 | 3
 50       | sushmita | bca       | 20,000 | 3
 60       | imran    | b.sc      | 20,000 | 3
 10       | hithesh  | b.sc      | 60000  | 2
(6 rows)


                        Table "public.staffs"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 staff_id    | character varying(20) |           | not null |
 s_name      | character varying(50) |           | not null |
 education   | character varying(60) |           | not null |
 salary      | character varying(50) |           | not null |
 branch_code | character varying(20) |           | not null |
Indexes:
    "staffs_pkey" PRIMARY KEY, btree (staff_id)
Foreign-key constraints:
    "fk_staff_branch_code" FOREIGN KEY (branch_code) REFERENCES beautyparlor(branch_code) ON UPDATE CASCADE ON DELETE CASCADE
Referenced by:
    TABLE "records" CONSTRAINT "fk1_records_staff_id" FOREIGN KEY (staff_id) REFERENCES staffs(staff_id) ON UPDATE CASCADE ON DELETE CASCADE
Triggers:
    staff_salary_change BEFORE UPDATE ON staffs FOR EACH ROW EXECUTE FUNCTION staff_salary_change()
```

4. staff skill table

```
CREATE TABLE staff_skill(
    staff_id varchar(20) primary key not null,
    skills varchar(200) not null,
    branch_code varchar(20) not null
 );


INSERT INTO staff_skill(staff_id,skills,branch_code)
VALUES
('10','modelling','2'),
('20','makeup artist','1'),
('30','hair stylist','1'),
('40','nail artist','3'),
('50','modelling','3'),
('60','makeup artist','3');

ALTER TABLE staff_skill
ADD CONSTRAINT fk_staff_skill_branch_code
FOREIGN KEY (branch_code)
REFERENCES beautyparlor(branch_code)
ON DELETE CASCADE ON UPDATE CASCADE;
```

OUTPUT:

```
 staff_id |    skills     | branch_code
----------+---------------+-------------
 10       | modelling     | 2
 20       | makeup artist | 1
 30       | hair stylist  | 1
 40       | nail artist   | 3
 50       | modelling     | 3
 60       | makeup artist | 3
(6 rows)


                    Table "public.staff_skill"
   Column    |          Type          | Collation | Nullable | Default
-------------+------------------------+-----------+----------+---------
 staff_id    | character varying(20)  |           | not null |
 skills      | character varying(200) |           | not null |
 branch_code | character varying(20)  |           | not null |
Indexes:
    "staff_skill_pkey" PRIMARY KEY, btree (staff_id)
Foreign-key constraints:
    "fk_staff_skill_branch_code" FOREIGN KEY (branch_code) REFERENCES beautyparlor(branch_code) ON UPDATE CASCADE ON DELETE CASCADE
```

5. packages table

```
CREATE TABLE packages(
    p_id varchar(20) primary key not null,
    p_name varchar(60) NOT NULL,
    p_type varchar(60) NOT NULL,
    branch_code varchar(20) not null
 );

INSERT INTO packages(p_id,p_name,p_type,branch_code)
 VALUES('12','loreal','wholesales','1'),
 ('13','himalayas','new arrivals','1'),
 ('15','lakme','branded sales','3'),
 ('14','herbals','discounts','2');

ALTER TABLE packages
ADD CONSTRAINT fk_packages_branch_code
FOREIGN KEY (branch_code)
REFERENCES beautyparlor(branch_code)
ON DELETE CASCADE ON UPDATE CASCADE;
```

OUTPUT:

```
 p_id |  p_name   |    p_type     | branch_code
------+-----------+---------------+-------------
 12   | loreal    | wholesales    | 1
 13   | himalayas | new arrivals  | 1
 15   | lakme     | branded sales | 3
 14   | herbals   | discounts     | 2
(4 rows)


                       Table "public.packages"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 p_id        | character varying(20) |           | not null |
 p_name      | character varying(60) |           | not null |
 p_type      | character varying(60) |           | not null |
 branch_code | character varying(20) |           | not null |
Indexes:
    "packages_pkey" PRIMARY KEY, btree (p_id)
Foreign-key constraints:
    "fk_packages_branch_code" FOREIGN KEY (branch_code) REFERENCES beautyparlor(branch_code) ON UPDATE CASCADE ON DELETE CASCADE
```

6. services table

```
CREATE TABLE services(
    s_id varchar(20) primary key not null,
    s_type varchar(60) NOT NULL,
    branch_code varchar(20) not null
 );

INSERT INTO services(s_id,s_type,branch_code)
 VALUES('1000','facial','1'),
 ('2000','pedicure','1'),
 ('3000','manicure','1'),
 ('4000','massages','2'),
 ('5000','waxing','2'),
 ('6000','threading','3'),
 ('7000','makeup','3');

ALTER TABLE services
ADD CONSTRAINT fk_services_branch_code
FOREIGN KEY (branch_code)
REFERENCES beautyparlor(branch_code)
ON DELETE CASCADE ON UPDATE CASCADE;
```

OUTPUT:

```
 s_id |  s_type   | branch_code
------+-----------+-------------
 1000 | facial    | 1
 2000 | pedicure  | 1
 3000 | manicure  | 1
 4000 | massages  | 2
 5000 | waxing    | 2
 6000 | threading | 3
 7000 | makeup    | 3
(7 rows)
```

```
                       Table "public.services"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 s_id        | character varying(20) |           | not null |
 s_type      | character varying(60) |           | not null |
 branch_code | character varying(20) |           | not null |
Indexes:
    "services_pkey" PRIMARY KEY, btree (s_id)
Foreign-key constraints:
    "fk_services_branch_code" FOREIGN KEY (branch_code) REFERENCES beautyparlor(branch_code) ON UPDATE CASCADE ON DELETE CASCADE
```

7. customers table

```
CREATE TABLE customers(
   c_id varchar(20) primary key not null,
   c_name varchar(200) not null,
   c_address varchar(200) not null,
   contact bigint not null,
   gender varchar(20) not null
);
```

```
INSERT INTO customers(c_id,c_name,c_address,contact,gender)
VALUES
('123','ayush','hosakerehalli road,3rd cross, 5th main, banashankari, 560050','9449331740','male'),
('124','ayesha','r.k gupta road, 4th main, chandigarh 450060','9432567432','female'),
('125','rohan','indiranagar, 6th cross ,layout bangalore 560030','9345678653','male'),
('126','prajaktha','shankar guru circle, 4th main, new bellore, kolkata 400500','8132453465','female'),
('127','apoorva','shivanagar, 4th road, new delhi 405060','7123634567','female'),
('128','barkha','r r nagar, 7th cross gurgauv, punjab 505060','8934236543','female'),
('129','carry','h s koppa road, 4th cross, basavani shringeri 278312','7786546534','male'),
('130','bhuvan','new brigade road, 5th main manipal, udupi 276210','9823547688','male');
```

OUTPUT:

```
 c_id |  c_name   |                          c_address                           |  contact   | gender
------+-----------+--------------------------------------------------------------+------------+--------
 124  | ayesha    | r.k gupta road, 4th main, chandigarh 450060                  | 9432567432 | female
 125  | rohan     | indiranagar, 6th cross ,layout bangalore 560030              | 9345678653 | male
 126  | prajaktha | shankar guru circle, 4th main, new bellore, kolkata 400500   | 8132453465 | female
 127  | apoorva   | shivanagar, 4th road, new delhi 405060                       | 7123634567 | female
 128  | barkha    | r r nagar, 7th cross gurgauv, punjab 505060                  | 8934236543 | female
 129  | carry     | h s koppa road, 4th cross, basavani shringeri 278312         | 7786546534 | male
 130  | bhuvan    | new brigade road, 5th main manipal, udupi 276210             | 9823547688 | male
 123  | ayush     | hosakerehalli road,3rd cross, 5th main, banashankari, 560050 | 9480763682 | male
(8 rows)
```

```
                      Table "public.customers"
  Column   |          Type          | Collation | Nullable | Default
-----------+------------------------+-----------+----------+---------
 c_id      | character varying(20)  |           | not null |
 c_name    | character varying(200) |           | not null |
 c_address | character varying(200) |           | not null |
 contact   | bigint                 |           | not null |
 gender    | character varying(20)  |           | not null |
Indexes:
    "customers_pkey" PRIMARY KEY, btree (c_id)
Referenced by:
    TABLE "appointments" CONSTRAINT "fk_appointments_c_id" FOREIGN KEY (c_id) REFERENCES customers(c_id) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "payment" CONSTRAINT "fk_payment_c_id" FOREIGN KEY (c_id) REFERENCES customers(c_id) ON UPDATE CASCADE ON DELETE CASCADE
Triggers:
    customer_contact_change BEFORE UPDATE ON customers FOR EACH ROW EXECUTE FUNCTION customer_contact_change()
```

8. appointment table

```
CREATE TABLE appointments(
   a_id varchar(20) primary key not null,
   a_date DATE not null,
   c_id varchar(20) not null
   );


INSERT INTO appointments(a_id,a_date,c_id)
VALUES('1732',DATE '3/3/2018','123'),
('1733',DATE '4/3/2018','125'),
('1734',DATE '5/3/2018','127'),
('1735',DATE '6/3/2018','130');

ALTER TABLE appointments
ADD CONSTRAINT fk_appointments_c_id
FOREIGN KEY (c_id)
REFERENCES customers(c_id)
ON DELETE CASCADE ON UPDATE CASCADE;
```

OUTPUT:

```
 a_id |   a_date   | c_id
------+------------+------
 1732 | 2018-03-03 | 123
 1733 | 2018-03-04 | 125
 1734 | 2018-03-05 | 127
 1735 | 2018-03-06 | 130
(4 rows)


                  Table "public.appointments"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 a_id   | character varying(20) |           | not null |
 a_date | date                  |           | not null |
 c_id   | character varying(20) |           | not null |
Indexes:
    "appointments_pkey" PRIMARY KEY, btree (a_id)
Foreign-key constraints:
    "fk_appointments_c_id" FOREIGN KEY (c_id) REFERENCES customers(c_id) ON UPDATE CASCADE ON DELETE CASCADE
Referenced by:
    TABLE "records" CONSTRAINT "fk2_appointments_a_id" FOREIGN KEY (a_id) REFERENCES appointments(a_id) ON UPDATE CASCADE ON DELETE CASCADE
```

9. payment table

```
CREATE TABLE payment(
    p_id varchar(20)  not null,
    p_amount varchar(50) not null,
    p_date date not null,
    c_id varchar(20) primary key not null
);

INSERT INTO payment(p_id,p_amount,p_date,c_id)
VALUES
('1670','50,000/-',date '5/3/2018','123'),
('1671','30,000/-',date '7/3/2018','125'),
('1672','10,000/-',date '9/3/2018','127'),
('1673','40,000/-',date '12/3/2018','130');


ALTER TABLE payment
ADD CONSTRAINT fk_payment_c_id
FOREIGN KEY (c_id)
REFERENCES customers(c_id)
ON DELETE CASCADE ON UPDATE CASCADE;
```

OUTPUT:

```
 p_id | p_amount |   p_date    | c_id
------+----------+------------+------
 1670 | 50,000/- | 2018-03-05 | 123
 1671 | 30,000/- | 2018-03-07 | 125
 1672 | 10,000/- | 2018-03-09 | 127
 1673 | 40,000/- | 2018-03-12 | 130
(4 rows)


                     Table "public.payment"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 p_id     | character varying(20) |           | not null |
 p_amount | character varying(50) |           | not null |
 p_date   | date                  |           | not null |
 c_id     | character varying(20) |           | not null |
Indexes:
    "payment_pkey" PRIMARY KEY, btree (c_id)
Foreign-key constraints:
    "fk_payment_c_id" FOREIGN KEY (c_id) REFERENCES customers(c_id) ON UPDATE CASCADE ON DELETE CASCADE
```

10. records table

```
CREATE TABLE records(
    staff_id varchar(20) not null,
    a_id varchar(20) not null,
    primary key(staff_id,a_id)
);

INSERT INTO records(staff_id,a_id)
VALUES('10','1732'),
('20','1733'),
('30','1734'),
('40','1735');

ALTER TABLE records
ADD CONSTRAINT fk1_records_staff_id
FOREIGN KEY (staff_id)
REFERENCES staffs(staff_id)
ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT fk2_appointments_a_id
FOREIGN KEY (a_id)
REFERENCES appointments(a_id)
ON DELETE CASCADE ON UPDATE CASCADE;
```

OUTPUT:

```
 staff_id | a_id
----------+------
 10       | 1732
 20       | 1733
 30       | 1734
 40       | 1735
(4 rows)


                      Table "public.records"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 staff_id | character varying(20) |           | not null |
 a_id     | character varying(20) |           | not null |
Indexes:
    "records_pkey" PRIMARY KEY, btree (staff_id, a_id)
Foreign-key constraints:
    "fk1_records_staff_id" FOREIGN KEY (staff_id) REFERENCES staffs(staff_id) ON UPDATE CASCADE ON DELETE CASCADE
    "fk2_appointments_a_id" FOREIGN KEY (a_id) REFERENCES appointments(a_id) ON UPDATE CASCADE ON DELETE CASCADE
```

Triggers

A trigger is a stored procedure in database which automatically invokes whenever a special event in database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

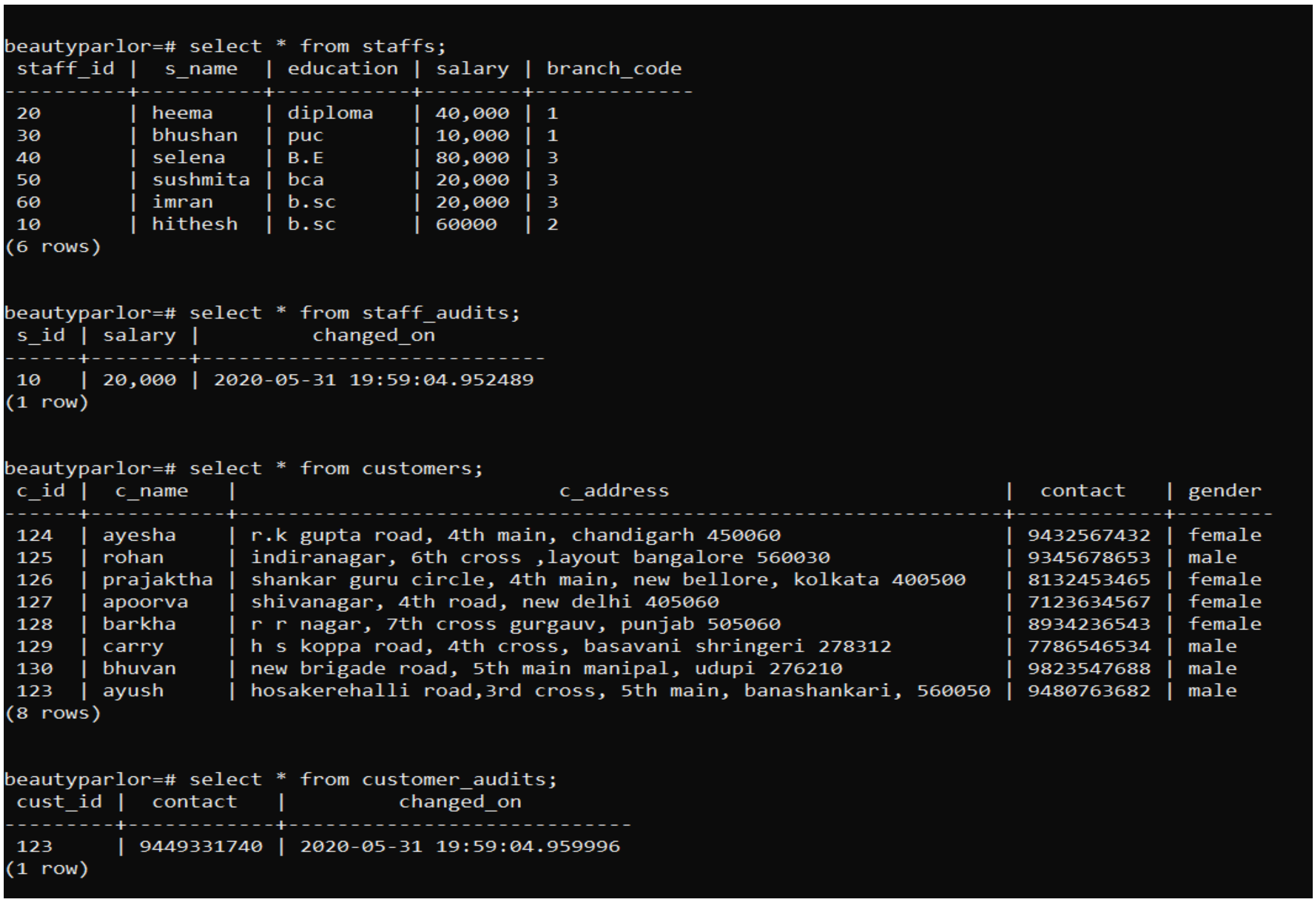
Here I have created **2 triggers for update operation** considering staff and customer table of the database. In the first case, trigger is invoked when newly updated salary of a staff is not equal to old salary. And in the second case, trigger is invoked when newly inserted contact number of the customer is not same as the old contact number.

SCRIPT TO CREATE TRIGGER:

```
CREATE TABLE STAFF_AUDITS
(
      s_id VARCHAR(15) NOT NULL,
      Salary VARCHAR(50) NOT NULL,
      changed_on TIMESTAMP(6) NOT NULL,
      PRIMARY KEY(s_id)

);

CREATE TABLE CUSTOMER_AUDITS
(
      cust_id VARCHAR(15) NOT NULL,
      contact VARCHAR(10) DEFAULT NULL,
      changed_on TIMESTAMP(6) NOT NULL,
      PRIMARY KEY(cust_id)

);


CREATE OR REPLACE FUNCTION STAFF_Salary_change()
  RETURNS trigger AS
$BODY$
BEGIN
```

```
        IF NEW.Salary <> OLD.Salary THEN
                INSERT INTO staff_audits(s_id,Salary,changed_on)
                VALUES(OLD.staff_id,OLD.Salary,now());
        END IF;
        RETURN NEW;
END;
$BODY$
language PLPGSQL;



CREATE OR REPLACE FUNCTION CUSTOMER_contact_change()
  RETURNS trigger AS
$BODY$
BEGIN
        IF NEW.contact <> OLD.contact THEN
                INSERT INTO customer_audits(cust_id,contact,changed_on)
                VALUES(OLD.c_id,OLD.contact,now());
        END IF;
        RETURN NEW;
END;
$BODY$
language PLPGSQL;


CREATE TRIGGER STAFF_Salary_change
  BEFORE UPDATE
  ON staffs
  FOR EACH ROW
  EXECUTE PROCEDURE STAFF_Salary_change();

CREATE TRIGGER CUSTOMER_contact_change
  BEFORE UPDATE
  ON CUSTOMERS
  FOR EACH ROW
  EXECUTE PROCEDURE CUSTOMER_contact_change();


UPDATE staffs
SET salary = '60,000'
WHERE staff_id = '10';

UPDATE customers
SET contact='9480763682'
WHERE c_id='123';
```

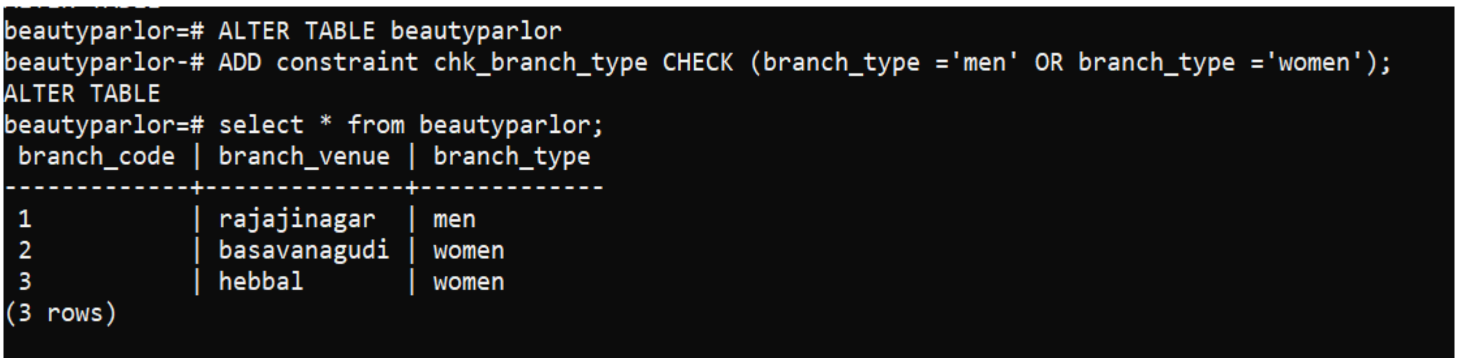
OUTPUT:



```
beautyparlor=# select * from staffs;
 staff_id |  s_name   | education | salary | branch_code
----------+----------+-----------+--------+-------------
 20       | heema    | diploma   | 40,000 | 1
 30       | bhushan  | puc       | 10,000 | 1
 40       | selena   | B.E       | 80,000 | 3
 50       | sushmita | bca       | 20,000 | 3
 60       | imran    | b.sc      | 20,000 | 3
 10       | hithesh  | b.sc      | 60000  | 2
(6 rows)


beautyparlor=# select * from staff_audits;
 s_id | salary |         changed_on
------+--------+----------------------------
 10   | 20,000 | 2020-05-31 19:59:04.952489
(1 row)


beautyparlor=# select * from customers;
 c_id |  c_name   |                          c_address                           |  contact   | gender
------+-----------+--------------------------------------------------------------+------------+--------
 124  | ayesha    | r.k gupta road, 4th main, chandigarh 450060                  | 9432567432 | female
 125  | rohan     | indiranagar, 6th cross ,layout bangalore 560030              | 9345678653 | male
 126  | prajaktha | shankar guru circle, 4th main, new bellore, kolkata 400500   | 8132453465 | female
 127  | apoorva   | shivanagar, 4th road, new delhi 405060                       | 7123634567 | female
 128  | barkha    | r r nagar, 7th cross gurgauv, punjab 505060                  | 8934236543 | female
 129  | carry     | h s koppa road, 4th cross, basavani shringeri 278312         | 7786546534 | male
 130  | bhuvan    | new brigade road, 5th main manipal, udupi 276210             | 9823547688 | male
 123  | ayush     | hosakerehalli road,3rd cross, 5th main, banashankari, 560050 | 9480763682 | male
(8 rows)


beautyparlor=# select * from customer_audits;
 cust_id |  contact   |         changed_on
---------+------------+----------------------------
 123     | 9449331740 | 2020-05-31 19:59:04.959996
(1 row)
```

CHECK CONSTRAINT:

ALTER TABLE beautyparlor
ADD constraint chk_branch_type CHECK (branch_type ='men' OR branch_type ='women');
OUTPUT:



```
beautyparlor=# ALTER TABLE beautyparlor
beautyparlor-# ADD constraint chk_branch_type CHECK (branch_type ='men' OR branch_type ='women');
ALTER TABLE
beautyparlor=# select * from beautyparlor;
 branch_code | branch_venue | branch_type
-------------+--------------+-------------
 1           | rajajinagar  | men
 2           | basavanagudi | women
 3           | hebbal       | women
(3 rows)
```

SQL Queries

1.SIMPLE AND AGGREGATE QUERIES

```
--To count number of entries in staffs --
select count(*) from staffs;

--to count the number of staffs in each branch--
select count(*) from staffs group by branch_code;

--to display customer id, name and gender of the customer in ascending order of their names--
select c_id,c_name,gender from customers
order by c_name;

--to display name and city of female owners of the parlour in ascending order of their name--
select fname,lname,o_city from owner
where gender='female' order by fname;

--to display appointment id recorded by staffs along with all details of staffs--
select * from staffs, records
where staffs.staff_id = records.staff_id;

--to display amount paid by male and female customers along with payment date--
select c_name,gender,p_amount,p_date
from customers,payment
where customers.c_id = payment.c_id;

--to display details of owners of all the branch with branch details--
select * from owner natural join beautyparlor;

--to display name of the staffs having minimum salary of 20000--
select s_name,min(Salary) from staffs group by staffs.s_name
having min(Salary)>'20,000';
```

--*to display name of the staff who has highest salary*--

```
select s_name from staffs where Salary>= ALL(select Salary from staffs);
```

NESTED QUERIES

--*to display name, contact number and address of customers whose customer id is in the tuple (123,124,125,126)*--

```
select c_name,contact,c_address from customers
where c_id in ('123','124','125','126');
```

--*to display name, salary and education of the staffs where their staff id is same as the id of staffs with skills having branch code 1,2,or 3*--

```
select s_name,Salary,education from staffs where
staff_id in (select staff_id from staff_skill where branch_code in ('1','2','3'));
```

CORRELATED NESTED QUERIES

--*to display appointment id which are recorded by staffs*—

```
select a_id from appointments where
exists(select a_id from records where appointments.a_id = records.a_id);
```

--*to display details of beauty parlour when branch codes of the parlour are same as branch codes of the branches that provide packages and services*--

```
select * from beautyparlor
where branch_code in
(select branch_code from packages intersect select branch_code from services);
```

TYPES OF SQL JOINS

```
--Inner join to join customer names and date of the appointment booked by them.—
select customers.c_name,appointments.a_date
from customers
inner join appointments on customers.c_id = appointments.c_id;

--left join to join customer name and amount paid by them--
select customers.c_name,payment.p_amount
from customers
left join payment on customers.c_id = payment.c_id;

--right join to join customer name and amount paid by them--
select customers.c_name,payment.p_amount
from customers
right join payment on customers.c_id = payment.c_id;

--full outer join to join branch code and type of packages offers by that branch with
the type of services provided by the same branch--
select packages.branch_code,packages.p_type,services.s_type
from packages
full outer join services on packages.branch_code = services.branch_code;
```

Conclusion

In this project, I have made use of PostgreSQL platform to execute sql queries.
I have modelled an E r diagram and then converted it into relational schema.
normalization process has made my work easier. I have shown all the queries to create tables, to insert values to the tables, to add foreign key constraints to it, and to create triggers on update operation. I have written few sql queries to carry out simple operations like select, join etc.
My goal was to develop a database with minimalistic approach which should be easy convenient yet portable and effective. this model can be further improvised in future to keep track of customer data , and to promote the services provided them to the public.
I hope this model, with proper improvisation, will be beneficial to new and upcoming beauty parlour database management systems.
Thank you.