LONDON METROPOLITAN UNIVERSITY

islington college

(इस्लिङ्टन कलेज)

Module Code & Module Title

CC5051NA Database Systems

Assessment Weightage & Type

50% Individual Coursework

Year and Semester

2018-19 Autumn

Student Name: Bibhu Manandhar

London Met ID: 18029955

College ID: NP01CP4A180301

Assignment Due Date: 30th December 2019

Assignment Submission Date: 29th December 2019

Title: Patient Record System

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded

Acknowledgement

I would like to humbly express my special thanks of gratitude towards all of my module teachers Ms. Yunisha Bajracharya and Mr. Yudhir Gurung, who gave me the golden opportunity to carry out this important assignment of Database Systems, who also helped us in completing this assignment with ease and conviction. From this assignment, I got to learn so many new things regarding the module and I am really grateful towards all of my teachers.

Table of Contents

Table of Contents

1. Introduction

1.1. Introduction of the hospital

The hospital chosen for this coursework is known as Grande International Hospital. It was established in February 2010 with the motive of “Care to cure”. The main aim for the creation of the hospital was to fill the existing void that existed in the Nepali healthcare industry and also be the leading healthcare provider in the South Asia region. It is also devoted to provide quality, patient-centric healthcare at an affordable cost. The main goal for this hospital is to establish, in Nepal, a culture of continuous improvement in healthcare by doing different health related programs. The objective for Grande International Hospital is also to be the destination for the best healthcare services the country has to offer.

The hospital currently is a 200-bed, best in class human services facility offering a wide scope of medicinal, careful and indicative administrations. (© Grande International Hospital, 2019)

1.2. Current Business Activities and Operations

There are a lot of patients both new and old that visits the hospital for a checkup or treatment for a disease they may have. Patients need to make a reservation for an appointment in order to be checked or diagnosed. Even the certified staff which are doctor/nurse/assistant can make an appointment as a patient which is free of cost whereas other beside the staff that are uncertified need to pay a certain fee like any other patient according to their treatment charge and ward charge.

The database records, for each person, all his/her address in which the address consists of country, province/state/zone, city, street, street number, and a list of phone numbers. Cell phone number and email address are also kept in the record books. Each person that the database can have are a regular patient, a new patient, a certified doctor/nurse/assistant, an uncertified doctor/nurse/assistant or any mixture of these. The database also stores appointment details that includes all the details of the treatment undergone while also store the data of the room/ward where the appointment was carried out.

1.3. Current Business Rules

1. A person can make multiple appointments.
2. An employee can handle only one appointment at a time.
3. An employee can have more than one appointment but not in a single day.
4. Certified employees of the hospital can be admitted and do not need to pay for their treatment charge meaning it is free of cost.
5. Patients and uncertified employees, who themselves are admitted as patients, need to pay accordingly to their treatments.
6. Employee get paid according to their treatment information.
7. One treatment room can be used only once in a day.

1.4. Identification of Entities and Attributes

Person = Person_ID, Person_Name, Person_Type, Person_Gender, Person_Age, Country, Zone, City, Street, Street_No, Phone_No, Cell_No, Email, Fax_No, Patient_ID, Employee_ID, Patient_Type, Employee_Type, Employee_Certification.

Appointment = Appointment_ID, Appointment_Type, Appointment_Date.

Treatment = Treatment_ID, Treatment_Date, Treatment_Charge, Ward_No, Ward_Type.

1.5. Initial E-R Diagram

1.5.1. ER Model

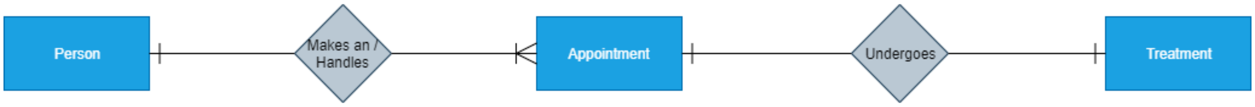


Figure 1: Initial ER Model

1.5.2. Assumption

1. Patient can make multiple appointments for different treatment.
2. Certified doctors, nurses and assistants do not need to pay for their treatment.
3. Patient type can be old or new or can also be an employee.
4. Employee type are doctors, nurses and assistant.
5. One appointment can only take one room and treatment.
6. At least one doctor should be assigned.
7. Employee type and Patient type cannot be null.
8. The address of a person is not unique.

2. Normalization

2.1. UNF

In the un-normalized form we take all the attributes and represent it by a single entity. The repeating groups are placed inside the curly braces.

Patient = (Person_ID, Person_Name, Person_Type, Person_Gender, Person_Age, Country, Zone, City, Street, Street_No, Phone_No, Cell_Phone_No, Email, Fax_No, Patient_ID, Patient_Type, Employee_ID, Employee_Type, Emp_Certification {Appointment_ID, Appointment_Type, Appointment_Date}, {Treatment_ID, Treatment_Date, Ward_No, Ward_Type, Treatment_Report, Treatment_Charge}).

2.2. 1NF

In the first normalization form we take the repeating groups and place them in a separate entity.

Person = (Person_ID, Person_Name, Person_Type, Person_Gender, Person_Age, Country, Zone, City, Street, Street_No, Phone_No, Cell_Phone_No, Email, Fax_No, Patient_ID, Patient_Type, Employee_ID, Employee_type, Employee_certification)

Appointment = (Person_ID, Appointment_ID, Appointment_Type, Appointment_Date)

Treatment = (Person_ID, Treatment_ID, Treatment_Date, Ward_No, Ward_Type, Treatment_Report, Treatment_Charge)

2.3. 2NF

In the second normalization form we check for any partial dependency and if a non-key attribute is found to be dependent on only one primary attribute, then it is placed on a separate entity. We can use the rule 2^n-1.

Appointment:

Person_ID →

Appointment_ID → Appointment_Type, Appointment_Date

Treatment:

Person_ID →

Treatment_ID → Treatment_ID, Treatment__Date, Ward_No, Ward_type

Entities after removing partial dependencies are:

Person = (Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age, Phone_No, Cell_Phone_No, Email, Fax_No, Patient_ID, Patient_Type, Employee_ID, Employee_Type, Employee_Certification).

Address = (Person_ID*, Country, Zone, City, Street, Street_No)

Appointment = (Appointment_ID*, Patient_ID, Employee_ID, Appointment_Type, Appointment_Date)

Treatment = (Treatment_ID*, Appointment_ID, Treatment_Date, Ward_No, Ward_Type)

Treatment_Info = (Treatment_ID, Treatment_Report, Treatment_Charge)

2.4. 3NF

In the third normalization form check for transitive dependencies and remove them. If a non-key is dependent on another non-key, then they are placed in a separate entity.

Person:

Patient:

Person→ Patient_ID → Patient_Type

Employee:

Person → Employee_ID → Employee_Type, Employee_Certification

Treamtment_Info:

Treatment_ID→ Treatment_Report, Treatment_Charge

The entities after removing the transitive dependencies are:

Person = (Person_ID, Person_Type, Person_Name, Person_Gender, Person_Age, Phone_No, Cell_Phone_No, Email, Fax_No)

Patient = (Person_ID*, Patient_ID, Patient_Type)

Employee = (Person_ID*, Employee_ID, Employee_Type, Employee_Certification)

Address = (Person_ID*, Country, Zone, City, Street, Street_No)

Patient, Employee and Address are subtype of the supertype Person Entity.

Appointment = (Appointment_ID*, Patient_ID, Employee_ID, Appointment_Type, Appointment_Date)

Treatment = (Treatment_ID*, Appointment_ID, Treatment_Date, Ward_No, Ward_Type)

Treatment_Info = (Treatment_ID, Treatment_Report, Treatment_Charge)

3. Entity Relation Diagram

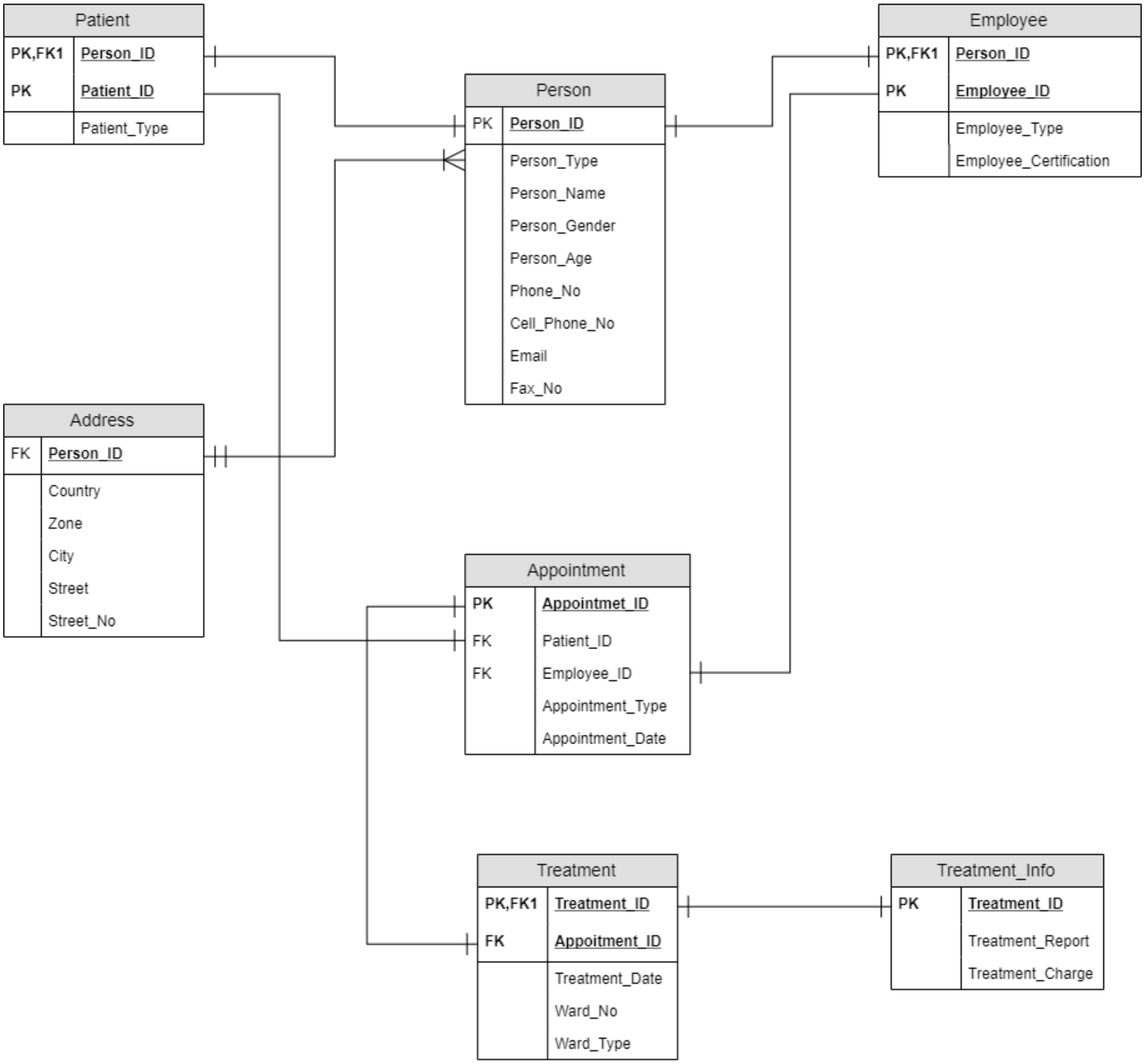


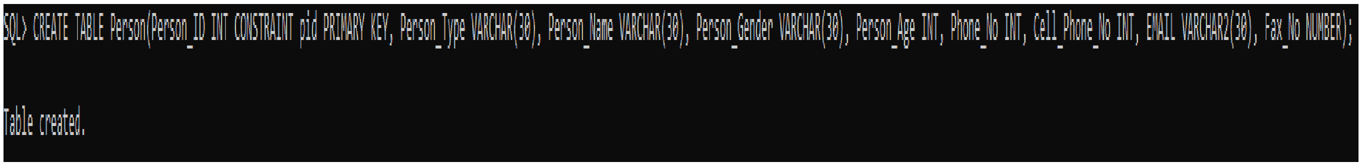
Figure 2: Normalized ER Diagram

4. Database Implementation

4.1. Tables Generation (DDL Scripts)

4.1.1. Person Table

CREATE TABLE Person (Person_ID INT CONSTRAINT pid PRIMARY KEY, Person_Type VARCHAR(30), Person_Name VARCHAR(30), Person_Gender VARCAHR(30), Person_Age INT, Phone_No INT, Cell_Phone_No, EMAIL VARCHAR2(30),Fax_No NUMBER);

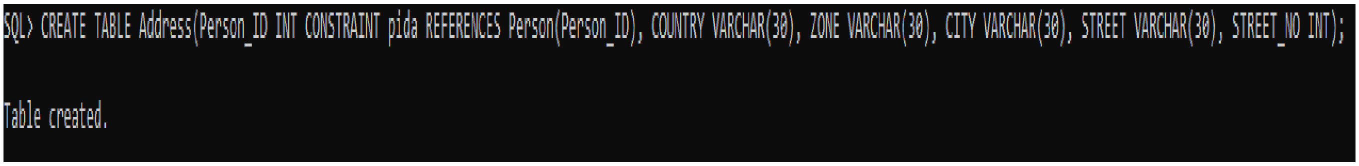


```
SQL> CREATE TABLE Person(Person_ID INT CONSTRAINT pid PRIMARY KEY, Person_Type VARCHAR(30), Person_Name VARCHAR(30), Person_Gender VARCHAR(30), Person_Age INT, Phone_No INT, Cell_Phone_No INT, EMAIL VARCHAR2(30), Fax_No NUMBER);

Table created.
```

Figure 3: Create Person Table

4.1.2. Address Table

CREATE TABLE Address(Person_ID INT CONSTRAINT pida REFERENCES Person(Person_ID), Country VARCHAR(30), Zone VARCHAR(30), City VARCHAR(30), Street VARCHAR(30), Street_No INT);

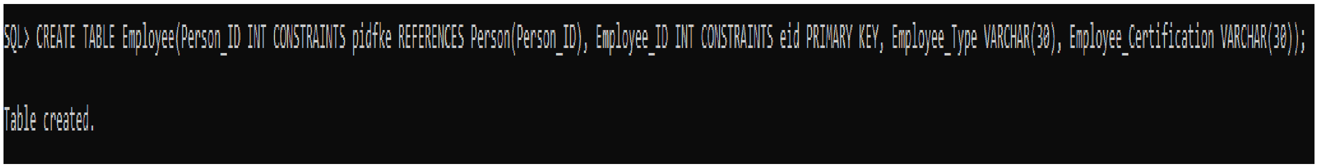


```
SQL> CREATE TABLE Address(Person_ID INT CONSTRAINT pida REFERENCES Person(Person_ID), COUNTRY VARCHAR(30), ZONE VARCHAR(30), CITY VARCHAR(30), STREET VARCHAR(30), STREET_NO INT);

Table created.
```

Figure 4: Create Address Table

4.1.3. Employee Table

CREATE TABLE Employee(Person_ID INT CONSTRAINTS pidfke REFERENCES Person(Person_ID), Employee_ID INT CONSTRAINTS eid PRIMARY KEY, Employee_Type VARCHAR(30), Employee_Certification VARCHAR(30));

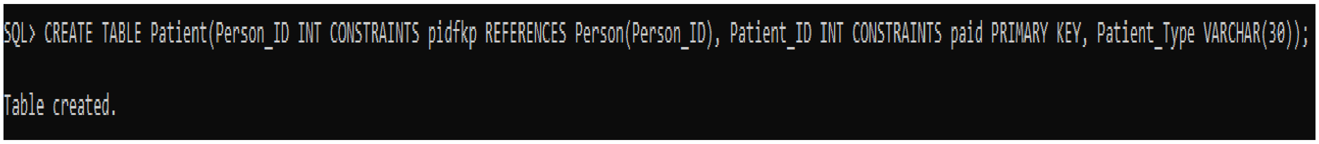


```
SQL> CREATE TABLE Employee(Person_ID INT CONSTRAINTS pidfke REFERENCES Person(Person_ID), Employee_ID INT CONSTRAINTS eid PRIMARY KEY, Employee_Type VARCHAR(30), Employee_Certification VARCHAR(30));

Table created.
```

Figure 5: Create Employee Table

4.1.4. Patient Table

CREATE TABLE Patient (Person_ID INT CONSTRAINTS pidfkp REFERENCES Person (Person_ID), Patient_ID INT CONSTRAINTS paid PRIMARY KEY, Patient_Type VARCHAR (30));

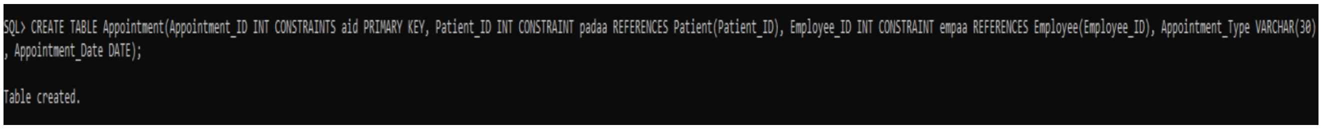


```
SQL> CREATE TABLE Patient(Person_ID INT CONSTRAINTS pidfkp REFERENCES Person(Person_ID), Patient_ID INT CONSTRAINTS paid PRIMARY KEY, Patient_Type VARCHAR(30));

Table created.
```

Figure 6: Create Patient Table

4.1.5. Appointment Table

CREATE TABLE Appointment (Appointment_ID CONSTRAINTS aid PRIMARY KEY, Patient_ID INT CONSTRAINTS padaa REFERENCES Patient (Patient_ID), Employee_ID INT CONSTRAINT empaa REFERENCES Employee (Employee_ID), Appointment_Type VARCHAR (30), Appointment_Date DATE);

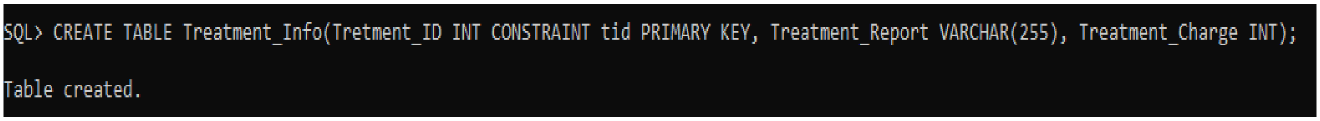


```
SQL> CREATE TABLE Appointment(Appointment_ID INT CONSTRAINTS aid PRIMARY KEY, Patient_ID INT CONSTRAINT padaa REFERENCES Patient(Patient_ID), Employee_ID INT CONSTRAINT empaa REFERENCES Employee(Employee_ID), Appointment_Type VARCHAR(30)
, Appointment_Date DATE);

Table created.
```

Figure 7: Create Appointment Table

4.1.6. Treatment_Info Table

CREATE TABLE Treatment_Info (Treatment_ID INT CONSTRAINT tid PRIMARY KEY, Treatment_Report VARCHAR (255), Treatment_Charge INT);

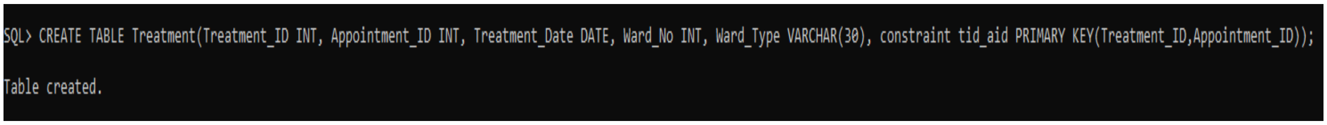


```
SQL> CREATE TABLE Treatment_Info(Tretment_ID INT CONSTRAINT tid PRIMARY KEY, Treatment_Report VARCHAR(255), Treatment_Charge INT);

Table created.
```

Figure 8: Create Treatment_Info Table

4.1.7. Treatment Table

CREATE TABLE Treatment(Treatment_ID INT, Appointment_ID INT, Treatment_Date DATE, Ward_No INT, Ward_Type VARCHAR(30), constraint tid_aid PRIMARY KEY(Treatment_ID,Appointment_ID));

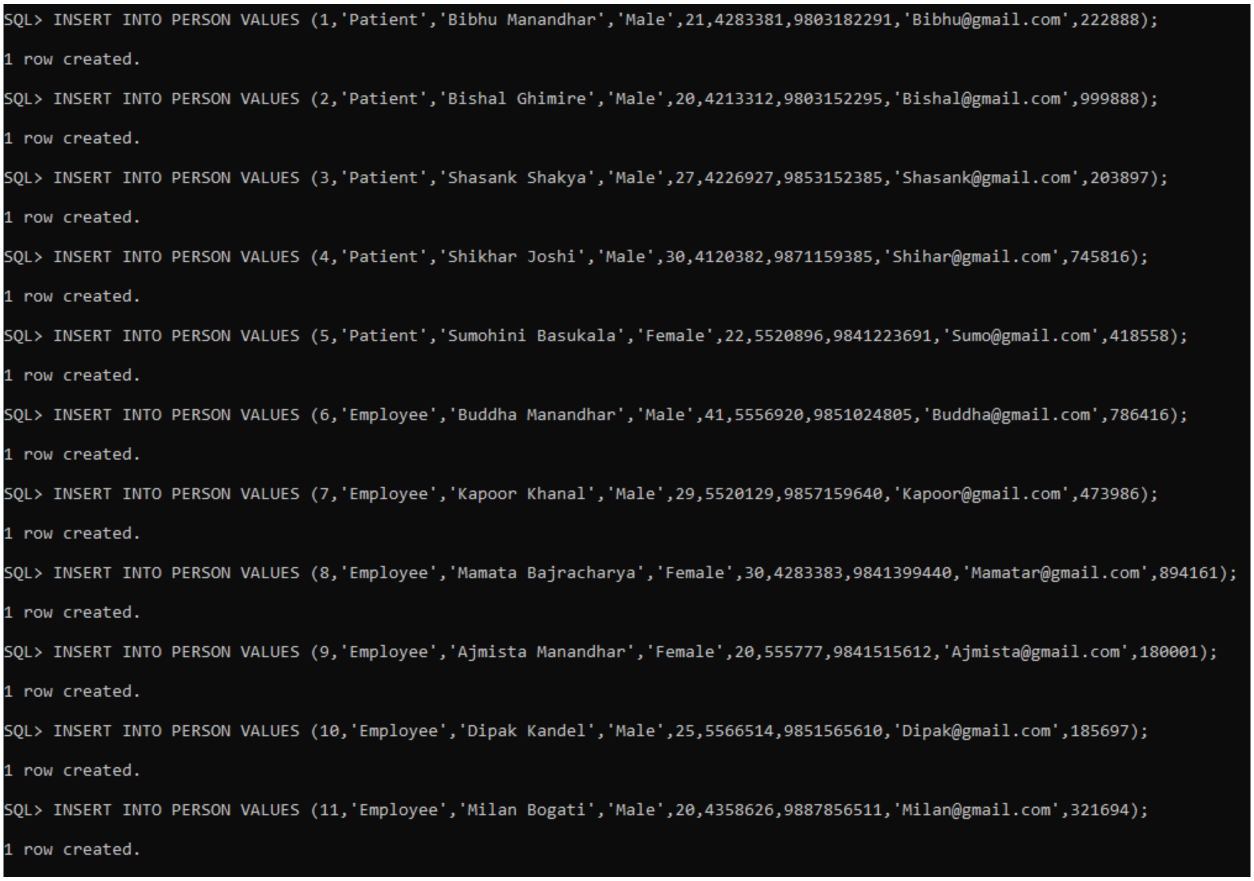


```
SQL> CREATE TABLE Treatment(Treatment_ID INT, Appointment_ID INT, Treatment_Date DATE, Ward_No INT, Ward_Type VARCHAR(30), constraint tid_aid PRIMARY KEY(Treatment_ID,Appointment_ID));

Table created.
```

Figure 9: Create Treatment Table

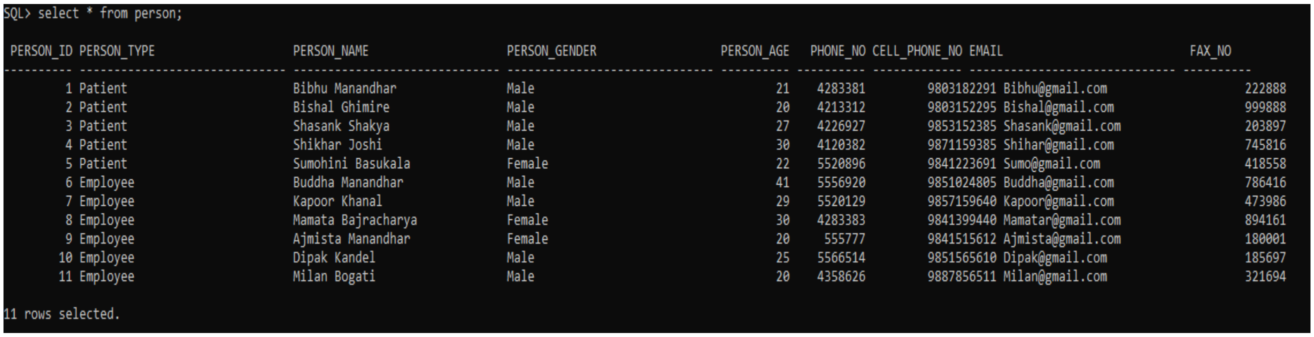
4.2. Populate DB tables

4.2.1. Inserting values to Person Table



```
SQL> INSERT INTO PERSON VALUES (1,'Patient','Bibhu Manandhar','Male',21,4283381,9803182291,'Bibhu@gmail.com',222888);

1 row created.

SQL> INSERT INTO PERSON VALUES (2,'Patient','Bishal Ghimire','Male',20,4213312,9803152295,'Bishal@gmail.com',999888);

1 row created.

SQL> INSERT INTO PERSON VALUES (3,'Patient','Shasank Shakya','Male',27,4226927,9853152385,'Shasank@gmail.com',203897);

1 row created.

SQL> INSERT INTO PERSON VALUES (4,'Patient','Shikhar Joshi','Male',30,4120382,9871159385,'Shihar@gmail.com',745816);

1 row created.

SQL> INSERT INTO PERSON VALUES (5,'Patient','Sumohini Basukala','Female',22,5520896,9841223691,'Sumo@gmail.com',418558);

1 row created.

SQL> INSERT INTO PERSON VALUES (6,'Employee','Buddha Manandhar','Male',41,5556920,9851024805,'Buddha@gmail.com',786416);

1 row created.

SQL> INSERT INTO PERSON VALUES (7,'Employee','Kapoor Khanal','Male',29,5520129,9857159640,'Kapoor@gmail.com',473986);

1 row created.

SQL> INSERT INTO PERSON VALUES (8,'Employee','Mamata Bajracharya','Female',30,4283383,9841399440,'Mamatar@gmail.com',894161);

1 row created.

SQL> INSERT INTO PERSON VALUES (9,'Employee','Ajmista Manandhar','Female',20,555777,9841515612,'Ajmista@gmail.com',180001);

1 row created.

SQL> INSERT INTO PERSON VALUES (10,'Employee','Dipak Kandel','Male',25,5566514,9851565610,'Dipak@gmail.com',185697);

1 row created.

SQL> INSERT INTO PERSON VALUES (11,'Employee','Milan Bogati','Male',20,4358626,9887856511,'Milan@gmail.com',321694);

1 row created.
```

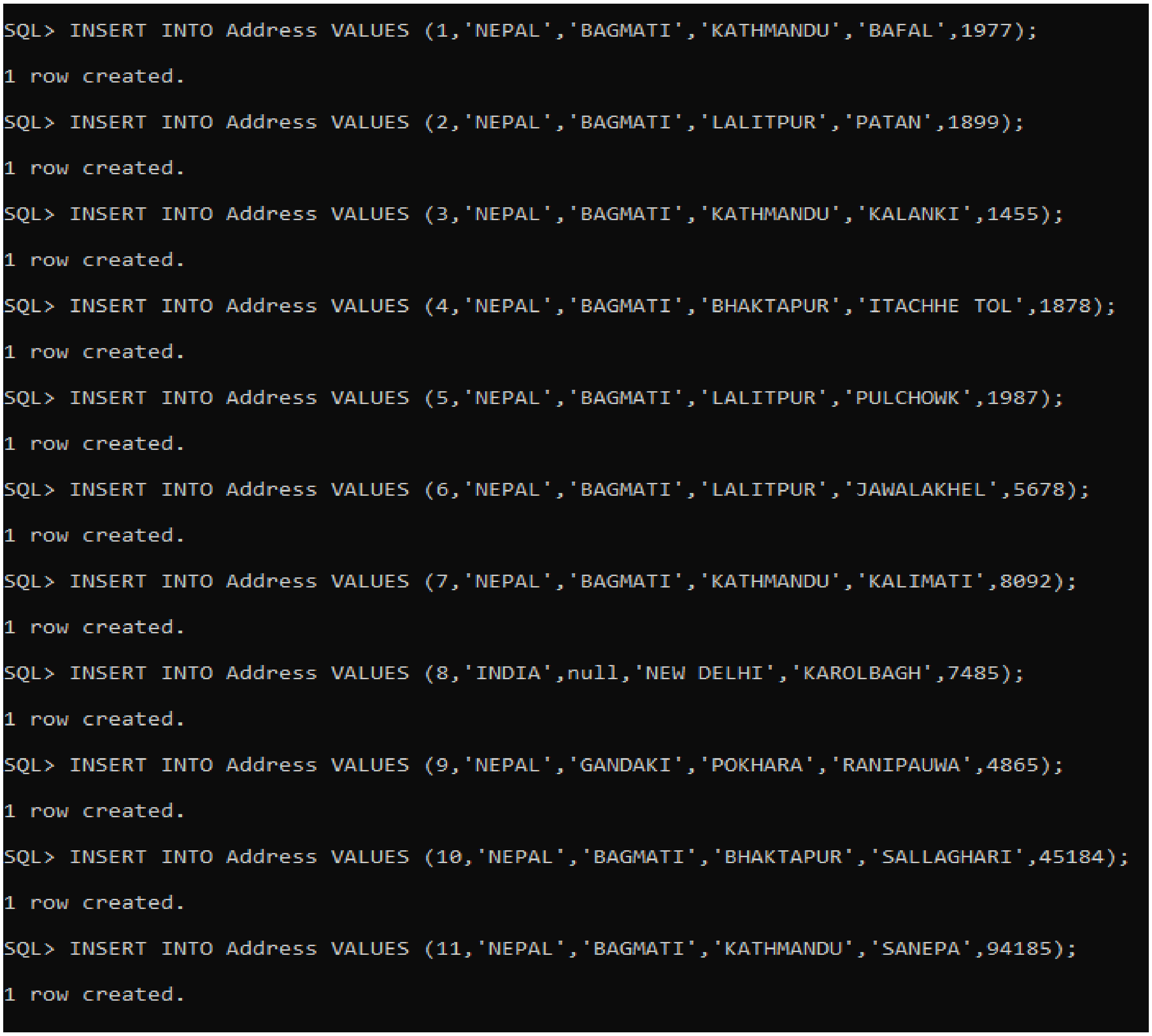
Figure 10: Insert to Patient Table



```
SQL> select * from person;

 PERSON_ID PERSON_TYPE                    PERSON_NAME                    PERSON_GENDER                  PERSON_AGE   PHONE_NO CELL_PHONE_NO EMAIL                          FAX_NO
---------- ------------------------------ ------------------------------ ------------------------------ ---------- ---------- ------------ ------------------------------ ----------
         1 Patient                        Bibhu Manandhar                Male                                   21    4283381   9803182291 Bibhu@gmail.com                    222888
         2 Patient                        Bishal Ghimire                 Male                                   20    4213312   9803152295 Bishal@gmail.com                   999888
         3 Patient                        Shasank Shakya                 Male                                   27    4226927   9853152385 Shasank@gmail.com                  203897
         4 Patient                        Shikhar Joshi                  Male                                   30    4120382   9871159385 Shihar@gmail.com                   745816
         5 Patient                        Sumohini Basukala              Female                                 22    5520896   9841223691 Sumo@gmail.com                     418558
         6 Employee                       Buddha Manandhar               Male                                   41    5556920   9851024805 Buddha@gmail.com                   786416
         7 Employee                       Kapoor Khanal                  Male                                   29    5520129   9857159640 Kapoor@gmail.com                   473986
         8 Employee                       Mamata Bajracharya             Female                                 30    4283383   9841399440 Mamatar@gmail.com                  894161
         9 Employee                       Ajmista Manandhar              Female                                 20     555777   9841515612 Ajmista@gmail.com                  180001
        10 Employee                       Dipak Kandel                   Male                                   25    5566514   9851565610 Dipak@gmail.com                    185697
        11 Employee                       Milan Bogati                   Male                                   20    4358626   9887856511 Milan@gmail.com                    321694

11 rows selected.
```

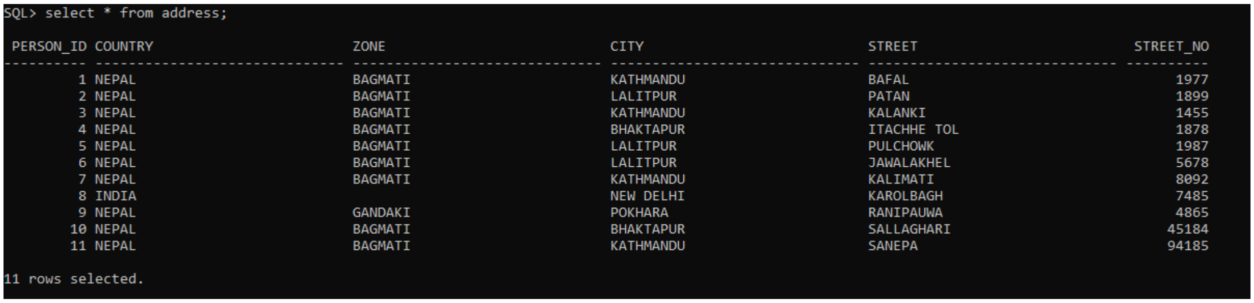
Figure 11: Patient Table Result

4.2.2. Inserting values to Address Table



```
SQL> INSERT INTO Address VALUES (1,'NEPAL','BAGMATI','KATHMANDU','BAFAL',1977);

1 row created.

SQL> INSERT INTO Address VALUES (2,'NEPAL','BAGMATI','LALITPUR','PATAN',1899);

1 row created.

SQL> INSERT INTO Address VALUES (3,'NEPAL','BAGMATI','KATHMANDU','KALANKI',1455);

1 row created.

SQL> INSERT INTO Address VALUES (4,'NEPAL','BAGMATI','BHAKTAPUR','ITACHHE TOL',1878);

1 row created.

SQL> INSERT INTO Address VALUES (5,'NEPAL','BAGMATI','LALITPUR','PULCHOWK',1987);

1 row created.

SQL> INSERT INTO Address VALUES (6,'NEPAL','BAGMATI','LALITPUR','JAWALAKHEL',5678);

1 row created.

SQL> INSERT INTO Address VALUES (7,'NEPAL','BAGMATI','KATHMANDU','KALIMATI',8092);

1 row created.

SQL> INSERT INTO Address VALUES (8,'INDIA',null,'NEW DELHI','KAROLBAGH',7485);

1 row created.

SQL> INSERT INTO Address VALUES (9,'NEPAL','GANDAKI','POKHARA','RANIPAUWA',4865);

1 row created.

SQL> INSERT INTO Address VALUES (10,'NEPAL','BAGMATI','BHAKTAPUR','SALLAGHARI',45184);

1 row created.

SQL> INSERT INTO Address VALUES (11,'NEPAL','BAGMATI','KATHMANDU','SANEPA',94185);

1 row created.
```

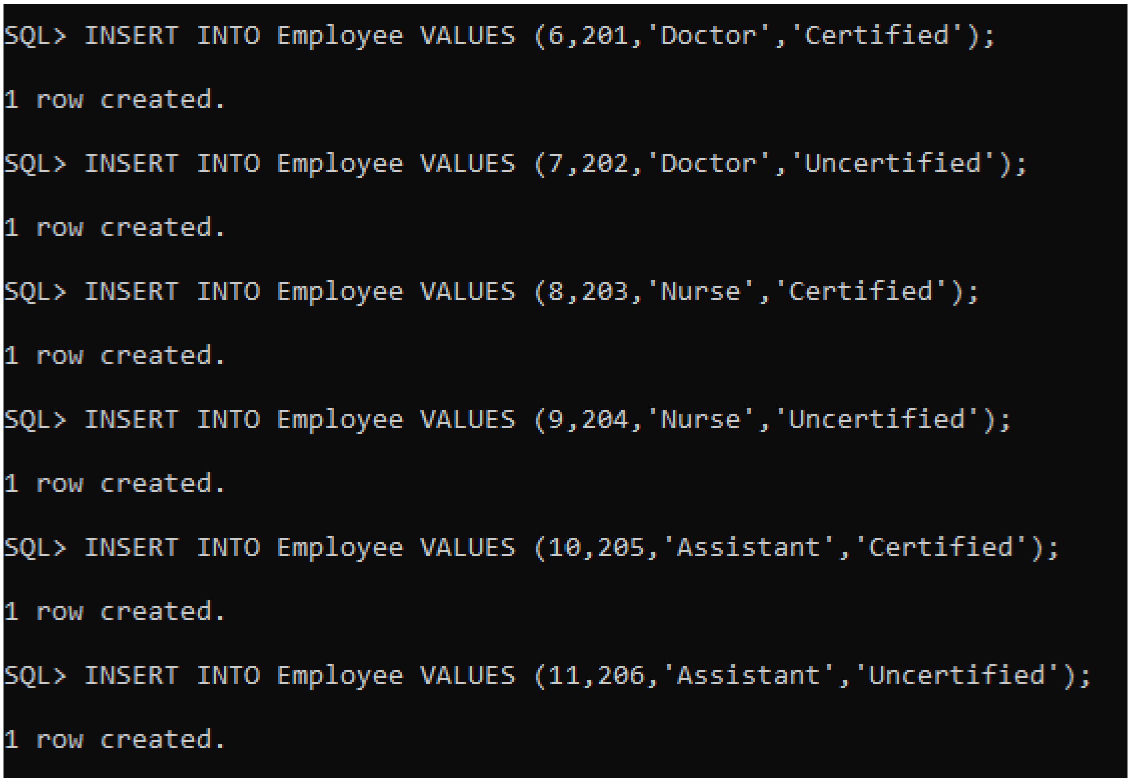
Figure 12: Insert to Address Table



```
SQL> select * from address;

 PERSON_ID COUNTRY                        ZONE                           CITY                           STREET                          STREET_NO
---------- ------------------------------ ------------------------------ ------------------------------ ------------------------------ ----------
         1 NEPAL                          BAGMATI                        KATHMANDU                      BAFAL                                1977
         2 NEPAL                          BAGMATI                        LALITPUR                       PATAN                                1899
         3 NEPAL                          BAGMATI                        KATHMANDU                      KALANKI                              1455
         4 NEPAL                          BAGMATI                        BHAKTAPUR                      ITACHHE TOL                          1878
         5 NEPAL                          BAGMATI                        LALITPUR                       PULCHOWK                             1987
         6 NEPAL                          BAGMATI                        LALITPUR                       JAWALAKHEL                           5678
         7 NEPAL                          BAGMATI                        KATHMANDU                      KALIMATI                             8092
         8 INDIA                                                         NEW DELHI                      KAROLBAGH                            7485
         9 NEPAL                          GANDAKI                        POKHARA                        RANIPAUWA                            4865
        10 NEPAL                          BAGMATI                        BHAKTAPUR                      SALLAGHARI                          45184
        11 NEPAL                          BAGMATI                        KATHMANDU                      SANEPA                              94185

11 rows selected.
```

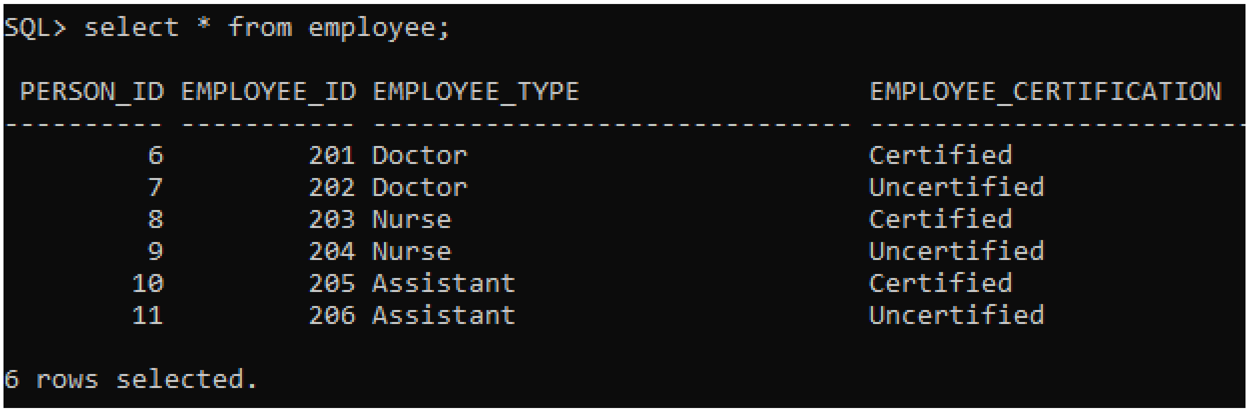
Figure 13: Address Table Result

4.2.3. Inserting values to Employee Table



```
SQL> INSERT INTO Employee VALUES (6,201,'Doctor','Certified');

1 row created.

SQL> INSERT INTO Employee VALUES (7,202,'Doctor','Uncertified');

1 row created.

SQL> INSERT INTO Employee VALUES (8,203,'Nurse','Certified');

1 row created.

SQL> INSERT INTO Employee VALUES (9,204,'Nurse','Uncertified');

1 row created.

SQL> INSERT INTO Employee VALUES (10,205,'Assistant','Certified');

1 row created.

SQL> INSERT INTO Employee VALUES (11,206,'Assistant','Uncertified');

1 row created.
```

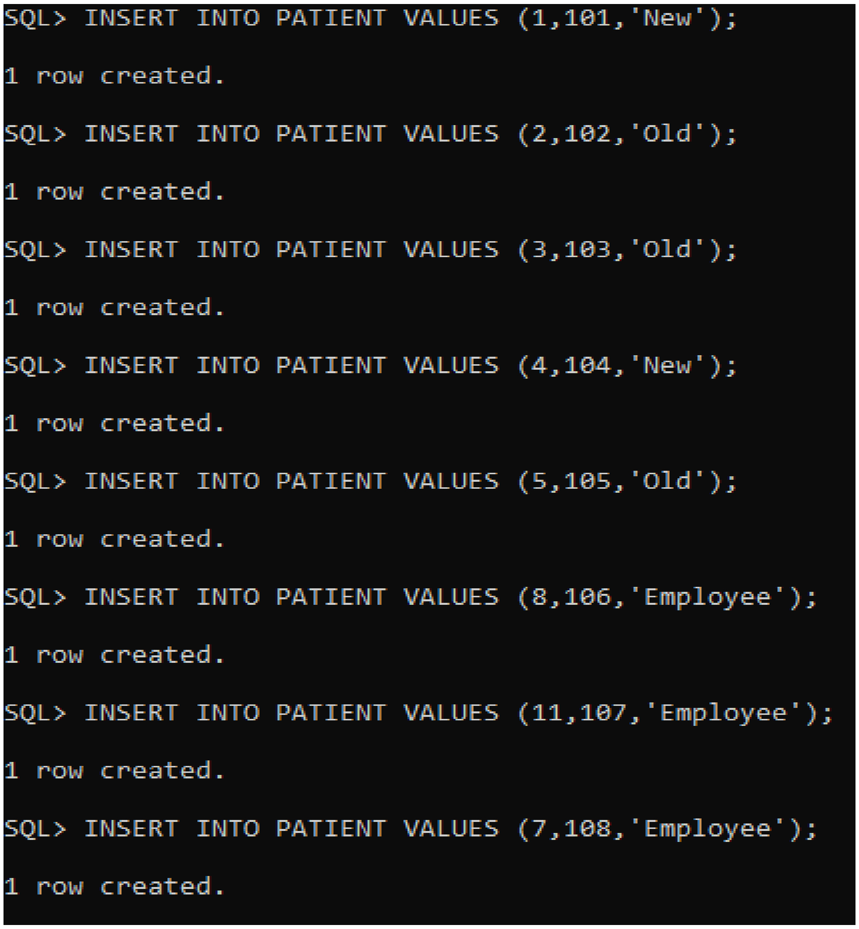
Figure 14: Insert to Employee Table



```
SQL> select * from employee;

 PERSON_ID EMPLOYEE_ID EMPLOYEE_TYPE                  EMPLOYEE_CERTIFICATION
---------- ----------- ------------------------------ -----------------------
         6         201 Doctor                         Certified
         7         202 Doctor                         Uncertified
         8         203 Nurse                          Certified
         9         204 Nurse                          Uncertified
        10         205 Assistant                      Certified
        11         206 Assistant                      Uncertified

6 rows selected.
```

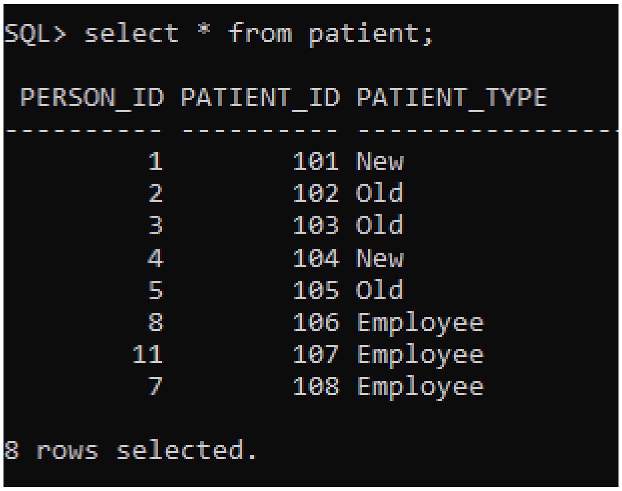
Figure 15: Employee Table Result

4.2.4. Inserting values to Patient Table



```
SQL> INSERT INTO PATIENT VALUES (1,101,'New');

1 row created.

SQL> INSERT INTO PATIENT VALUES (2,102,'Old');

1 row created.

SQL> INSERT INTO PATIENT VALUES (3,103,'Old');

1 row created.

SQL> INSERT INTO PATIENT VALUES (4,104,'New');

1 row created.

SQL> INSERT INTO PATIENT VALUES (5,105,'Old');

1 row created.

SQL> INSERT INTO PATIENT VALUES (8,106,'Employee');

1 row created.

SQL> INSERT INTO PATIENT VALUES (11,107,'Employee');

1 row created.

SQL> INSERT INTO PATIENT VALUES (7,108,'Employee');

1 row created.
```

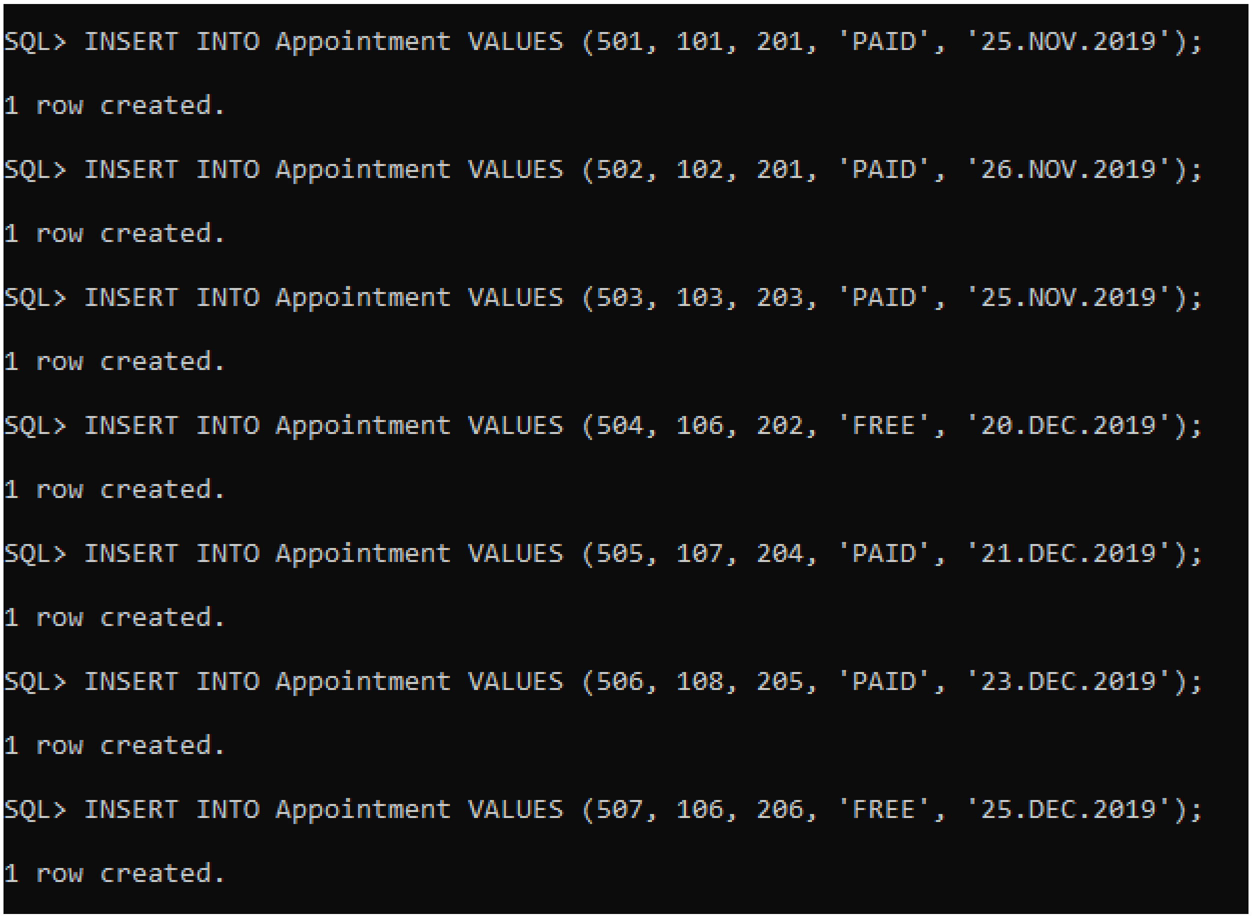
Figure 16: Insert to Patient Table



```
SQL> select * from patient;

 PERSON_ID PATIENT_ID PATIENT_TYPE
---------- ---------- ----------------
         1        101 New
         2        102 Old
         3        103 Old
         4        104 New
         5        105 Old
         8        106 Employee
        11        107 Employee
         7        108 Employee

8 rows selected.
```

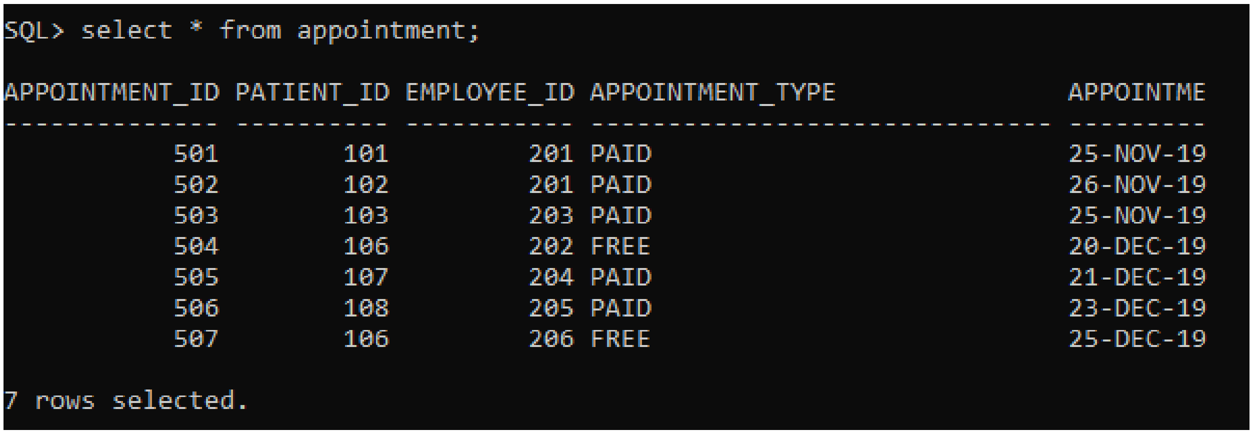
Figure 17: Patient Table Result

4.2.5. Inserting values to Appointment Table



```
SQL> INSERT INTO Appointment VALUES (501, 101, 201, 'PAID', '25.NOV.2019');

1 row created.

SQL> INSERT INTO Appointment VALUES (502, 102, 201, 'PAID', '26.NOV.2019');

1 row created.

SQL> INSERT INTO Appointment VALUES (503, 103, 203, 'PAID', '25.NOV.2019');

1 row created.

SQL> INSERT INTO Appointment VALUES (504, 106, 202, 'FREE', '20.DEC.2019');

1 row created.

SQL> INSERT INTO Appointment VALUES (505, 107, 204, 'PAID', '21.DEC.2019');

1 row created.

SQL> INSERT INTO Appointment VALUES (506, 108, 205, 'PAID', '23.DEC.2019');

1 row created.

SQL> INSERT INTO Appointment VALUES (507, 106, 206, 'FREE', '25.DEC.2019');

1 row created.
```

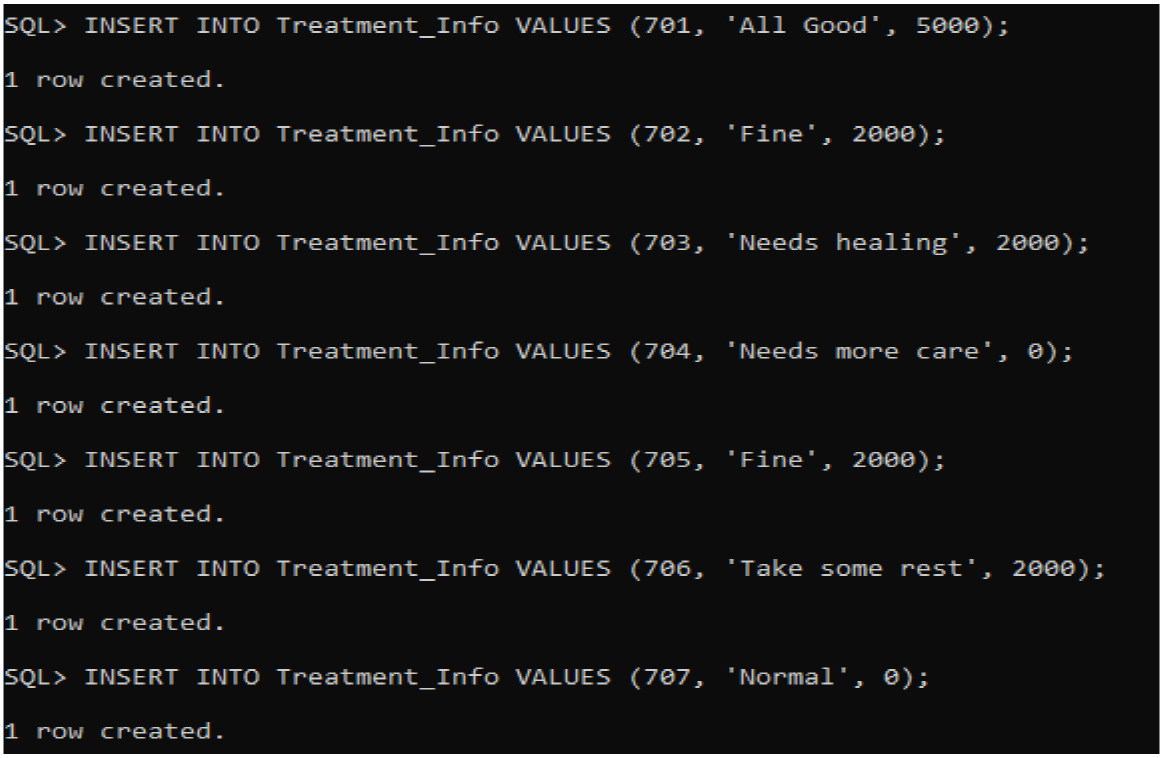
Figure 18: Insert to Appointment Table



```
SQL> select * from appointment;

APPOINTMENT_ID PATIENT_ID EMPLOYEE_ID APPOINTMENT_TYPE               APPOINTME
-------------- ---------- ----------- ------------------------------ ---------
           501        101         201 PAID                           25-NOV-19
           502        102         201 PAID                           26-NOV-19
           503        103         203 PAID                           25-NOV-19
           504        106         202 FREE                           20-DEC-19
           505        107         204 PAID                           21-DEC-19
           506        108         205 PAID                           23-DEC-19
           507        106         206 FREE                           25-DEC-19

7 rows selected.
```

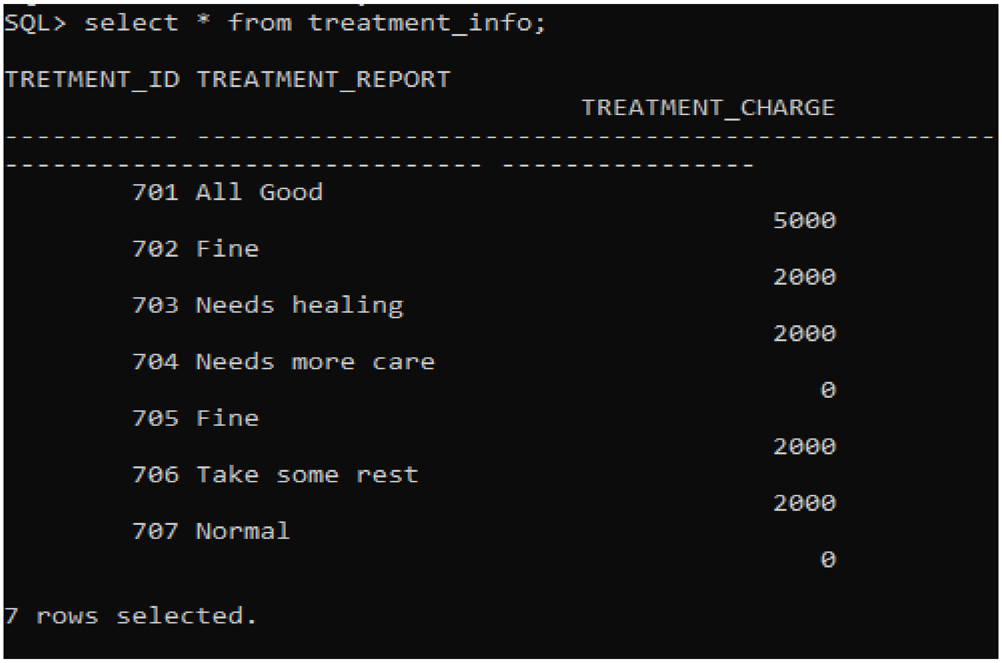
Figure 19: Appointment Table Result

4.2.6. Inserting values to Treatment_Info Table



```
SQL> INSERT INTO Treatment_Info VALUES (701, 'All Good', 5000);

1 row created.

SQL> INSERT INTO Treatment_Info VALUES (702, 'Fine', 2000);

1 row created.

SQL> INSERT INTO Treatment_Info VALUES (703, 'Needs healing', 2000);

1 row created.

SQL> INSERT INTO Treatment_Info VALUES (704, 'Needs more care', 0);

1 row created.

SQL> INSERT INTO Treatment_Info VALUES (705, 'Fine', 2000);

1 row created.

SQL> INSERT INTO Treatment_Info VALUES (706, 'Take some rest', 2000);

1 row created.

SQL> INSERT INTO Treatment_Info VALUES (707, 'Normal', 0);

1 row created.
```

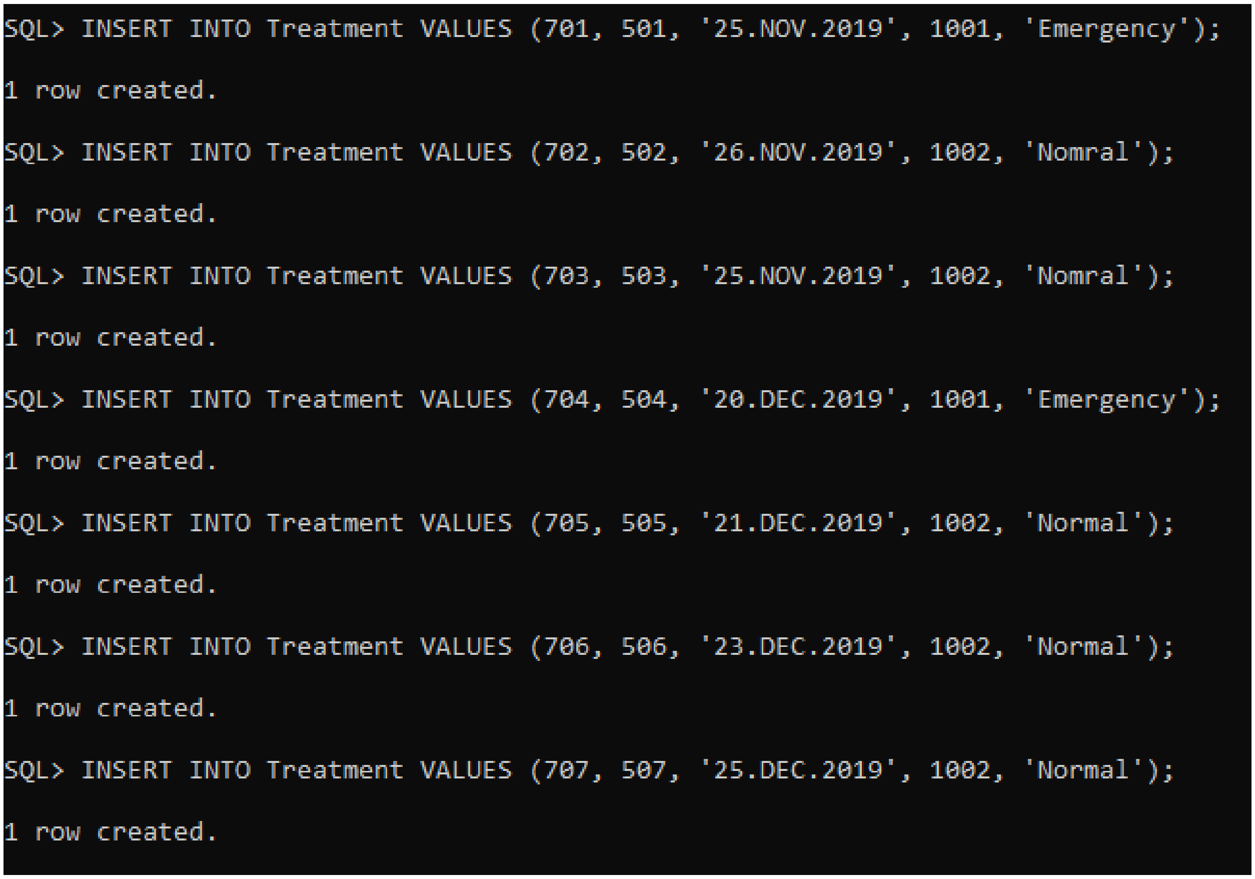
Figure 20: Insert to Appointment_Info Table



```
SQL> select * from treatment_info;

TRETMENT_ID TREATMENT_REPORT
                                    TREATMENT_CHARGE
----------- --------------------------------------------------
------------------------------ ----------------
        701 All Good
                                                5000
        702 Fine
                                                2000
        703 Needs healing
                                                2000
        704 Needs more care
                                                   0
        705 Fine
                                                2000
        706 Take some rest
                                                2000
        707 Normal
                                                   0

7 rows selected.
```

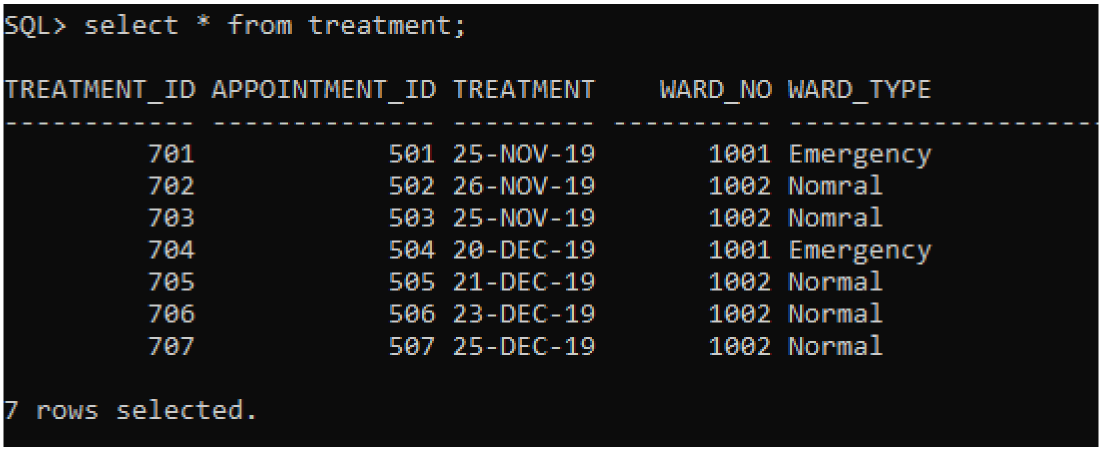
Figure 21: Appointment_Info Result

4.2.7. Inserting values to Treatment Table



```
SQL> INSERT INTO Treatment VALUES (701, 501, '25.NOV.2019', 1001, 'Emergency');

1 row created.

SQL> INSERT INTO Treatment VALUES (702, 502, '26.NOV.2019', 1002, 'Nomral');

1 row created.

SQL> INSERT INTO Treatment VALUES (703, 503, '25.NOV.2019', 1002, 'Nomral');

1 row created.

SQL> INSERT INTO Treatment VALUES (704, 504, '20.DEC.2019', 1001, 'Emergency');

1 row created.

SQL> INSERT INTO Treatment VALUES (705, 505, '21.DEC.2019', 1002, 'Normal');

1 row created.

SQL> INSERT INTO Treatment VALUES (706, 506, '23.DEC.2019', 1002, 'Normal');

1 row created.

SQL> INSERT INTO Treatment VALUES (707, 507, '25.DEC.2019', 1002, 'Normal');

1 row created.
```

Figure 22: Insert to Treatment Table



```
SQL> select * from treatment;

TREATMENT_ID APPOINTMENT_ID TREATMENT    WARD_NO WARD_TYPE
------------ -------------- --------- ---------- --------------------
         701            501 25-NOV-19       1001 Emergency
         702            502 26-NOV-19       1002 Nomral
         703            503 25-NOV-19       1002 Nomral
         704            504 20-DEC-19       1001 Emergency
         705            505 21-DEC-19       1002 Normal
         706            506 23-DEC-19       1002 Normal
         707            507 25-DEC-19       1002 Normal

7 rows selected.
```

Figure 23: Treatment Table Result

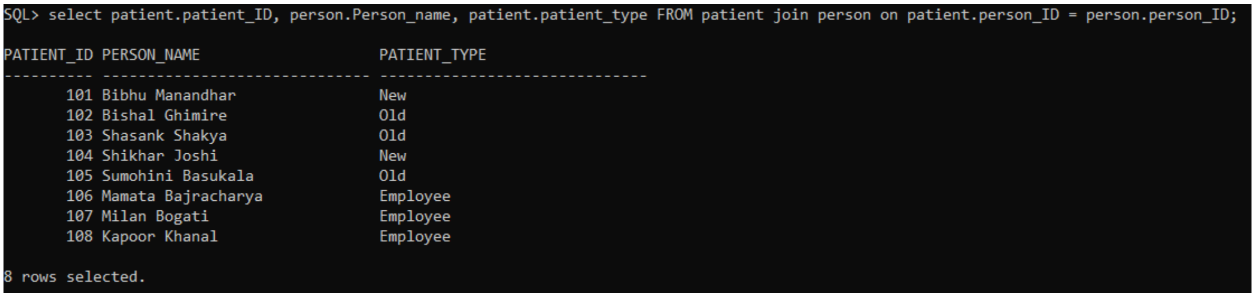
5. Database Querying

5.1. 4 SQL Information Queries

5.1.1. List all patients, regular, new and employee (Query 1)

This query selects columns Patient_ID and Patient_Type from Patient table and Person_name from Person table and then show results joining the two tables with condition where Person_ID from both tables have the same ID.

SQL> select patient.patient_ID, person.Person_name, patient.patient_type FROM patient join person on patient.personID = person.person_ID;



```
SQL> select patient.patient_ID, person.Person_name, patient.patient_type FROM patient join person on patient.person_ID = person.person_ID;

PATIENT_ID PERSON_NAME                    PATIENT_TYPE
---------- ------------------------------ ------------------------------
       101 Bibhu Manandhar                New
       102 Bishal Ghimire                 Old
       103 Shasank Shakya                 Old
       104 Shikhar Joshi                  New
       105 Sumohini Basukala              Old
       106 Mamata Bajracharya             Employee
       107 Milan Bogati                   Employee
       108 Kapoor Khanal                  Employee

8 rows selected.
```

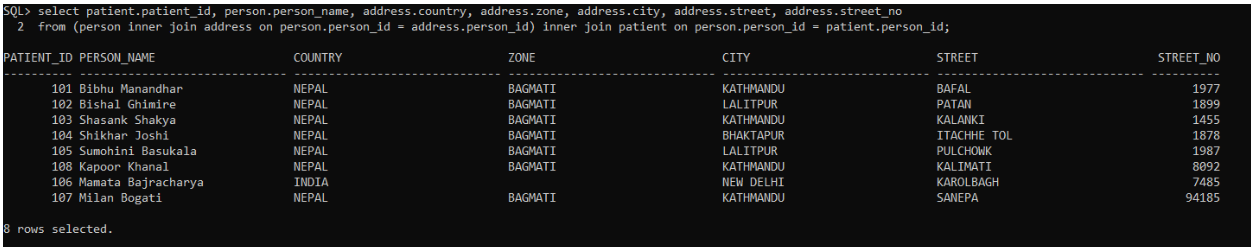
Figure 24: Query 1

5.1.2. List all patients with all their addresses. (Query 2)

This query selects columns patient_ID from patient table and person_name, country, zone, city, street, and street_no from person table and then show the result joining first the person_id of both person and address table and later joining this combined table with person_id of patient table.

SQL>select patient.patient_id, person.person_name, address.country, address.zone, address.city, address.street, address.street_no

2 from (person inner join address on person.person_id = address.person_id) inner join patient on person.person_id = patient.person_id;



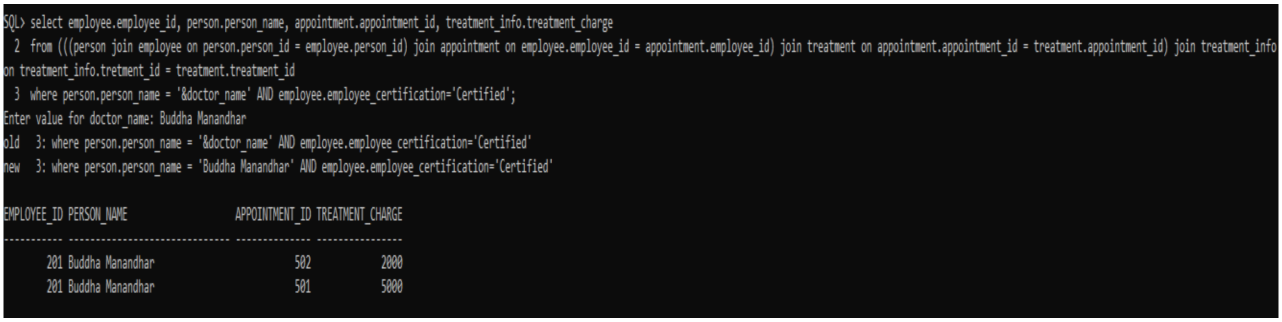
```
SQL> select patient.patient_id, person.person_name, address.country, address.zone, address.city, address.street, address.street_no
  2  from (person inner join address on person.person_id = address.person_id) inner join patient on person.person_id = patient.person_id;

PATIENT_ID PERSON_NAME                    COUNTRY                        ZONE                           CITY                           STREET                          STREET_NO
---------- ------------------------------ ------------------------------ ------------------------------ ------------------------------ ------------------------------ ----------
       101 Bibhu Manandhar                NEPAL                          BAGMATI                        KATHMANDU                      BAFAL                                1977
       102 Bishal Ghimire                 NEPAL                          BAGMATI                        LALITPUR                       PATAN                                1899
       103 Shasank Shakya                 NEPAL                          BAGMATI                        KATHMANDU                      KALANKI                              1455
       104 Shikhar Joshi                  NEPAL                          BAGMATI                        BHAKTAPUR                      ITACHHE TOL                          1878
       105 Sumohini Basukala              NEPAL                          BAGMATI                        LALITPUR                       PULCHOWK                             1987
       108 Kapoor Khanal                  NEPAL                          BAGMATI                        KATHMANDU                      KALIMATI                             8092
       106 Mamata Bajracharya             INDIA                                                         NEW DELHI                      KAROLBAGH                            7485
       107 Milan Bogati                   NEPAL                          BAGMATI                        KATHMANDU                      SANEPA                              94185

8 rows selected.
```

Figure 25: Query 2

5.1.3. For a given certified doctor, find all the appointments he/she have conducted and the amount he/she got for conducting the appointment. (Query 3)

This query selects columns employee_id, person_name, appointment_id and treatment_charge and first joining two tables person and employee with the same person.id and then joining the resulting table to appointment in which employee_id for both the employee and appointment table are the same, again this resulting table joins to treatment table on appointment_id of both appointment and treatment table, then finally again joins this table and treatment_info table in which treatment_id for treatment_info and treatment tables is the equal. Then the rows that are selected are restricted by the where clause that indicates person_name from person table must be a doctor and employee_certification from employee table must be certified.

```
SQL> select employee.employee_id, person.person_name, appointment.appointment_id, treatment_info.treatment_charge

2 from (((person join employee on person.person_id = employee.person_id) join appointment on employee.employee_id = appointment.employee_id) join treatment on appointment.appointment_id = treatment.appointment_id) join treatment_info on treatment_info.treatment_id = treatment.treatment_id

3 where person.person_name = ‘&doctor_name’ AND employee.employee_certification = ‘Certified’;

Enter value for doctor_name: Buddha Manandhar

Old 3: where person.person_name = ‘&doctor_name’ AND employee.employee_certification=’Certified’

New 3: where person.person_name = ‘Buddha Manandhar’ AND employee.employee_certification=’Certified’
```



```
SQL> select employee.employee_id, person.person_name, appointment.appointment_id, treatment_info.treatment_charge
  2  from (((person join employee on person.person_id = employee.person_id) join appointment on employee.employee_id = appointment.employee_id) join treatment on appointment.appointment_id = treatment.appointment_id) join treatment_info
on treatment_info.tretment_id = treatment.treatment_id
  3  where person.person_name = '&doctor_name' AND employee.employee_certification='Certified';
Enter value for doctor_name: Buddha Manandhar
old   3: where person.person_name = '&doctor_name' AND employee.employee_certification='Certified'
new   3: where person.person_name = 'Buddha Manandhar' AND employee.employee_certification='Certified'

EMPLOYEE_ID PERSON_NAME                    APPOINTMENT_ID TREATMENT_CHARGE
----------- ------------------------------ -------------- ----------------
        201 Buddha Manandhar                          502             2000
        201 Buddha Manandhar                          501             5000
```

Figure 26: Query 3

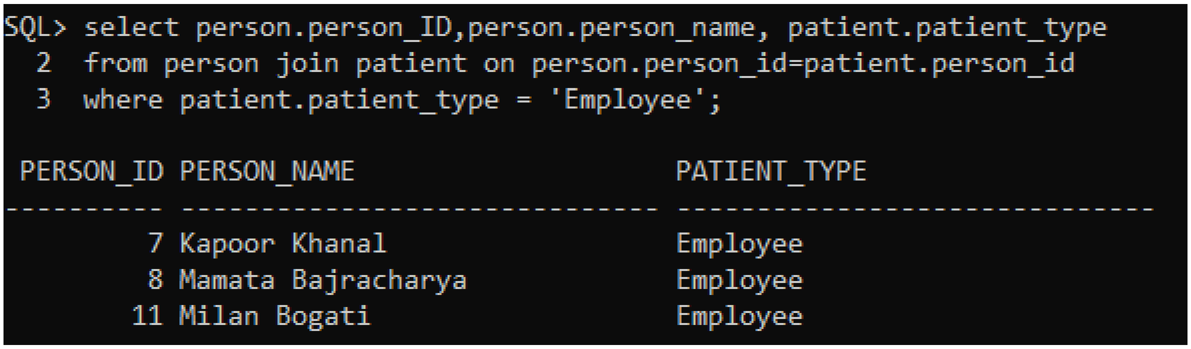
5.1.4. List all staffs that are also a patient. (Query 4)

This query selects columns person_id and person_name from person table, and patient_type from patient table and from the person table joins patient in which person_id of both tables are equal. Then the rows that are selected are restricted by the where clause that indicates patient_type from the patient table must be an Employee.

SQL> select person.person_ID, person.person_name, patient.patient_type

2 from person join patient on person.person_id = patient.person_id

3 where patient.patient_type = ‘Employee’;



```
SQL> select person.person_ID,person.person_name, patient.patient_type
  2  from person join patient on person.person_id=patient.person_id
  3  where patient.patient_type = 'Employee';

 PERSON_ID PERSON_NAME                    PATIENT_TYPE
---------- ------------------------------ ------------------------------
         7 Kapoor Khanal                  Employee
         8 Mamata Bajracharya             Employee
        11 Milan Bogati                   Employee
```

Figure 27: Query 4

5.2. 4 SQL Transaction Queries

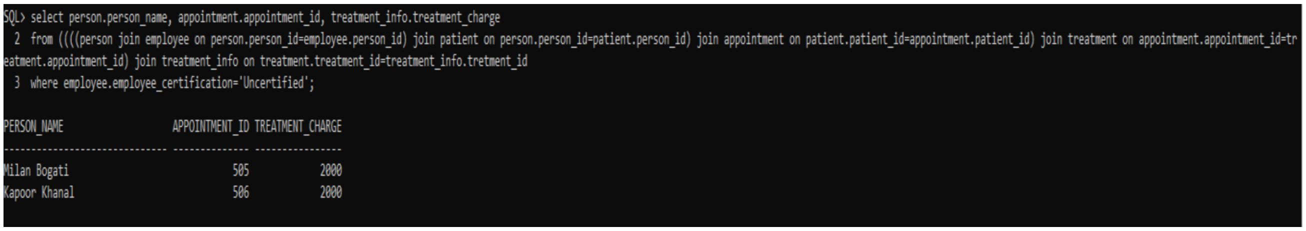
5.2.1. List all uncertified doctors who have been attended an appointment for a treatment and the amount he/she have paid. (Query 5)

This query selects columns person_name, appointment_id and treatment_charge and first from person table joins employee table in which person_id of both tables are equal, then this table joins patient table in which person_id of both tables person and patient table, again this table joins appointment table where patient_id is equal for both tables patient and appointment, furthermore the resulting table joins to treatment table in which also appointment_id is the same for both tables appointment and treatment, finally this table is joined onto treatment_info table where treatment_id for treatment table and treatment_info table is equal. Then the rows that are selected are restricted by the where clause that indicates employee_certification from employee table must be uncertified.

SQL> select person.person_name, appointment.appointment_id, treatment_info.treatment_charge

2 from ((((person join employee on person.person_id=employee.person_id) join patient on person.person_id=patient.person_id) join appointment on patient.patient_id=appointment.patient_id) join treatment on appointment.appointment_id = treatment.appointment_id) join treatment_info on treatment.treatment_id=treatment_info.treatment_id

3 where employee.employee_certification='Uncertified';



```
SQL> select person.person_name, appointment.appointment_id, treatment_info.treatment_charge
  2  from ((((person join employee on person.person_id=employee.person_id) join patient on person.person_id=patient.person_id) join appointment on patient.patient_id=appointment.patient_id) join treatment on appointment.appointment_id=tr
eatment.appointment_id) join treatment_info on treatment.treatment_id=treatment_info.tretment_id
  3  where employee.employee_certification='Uncertified';

PERSON_NAME                  APPOINTMENT_ID TREATMENT_CHARGE
---------------------------- -------------- ---------------
Milan Bogati                            505            2000
Kapoor Khanal                           506            2000
```

Figure 28: Query 5

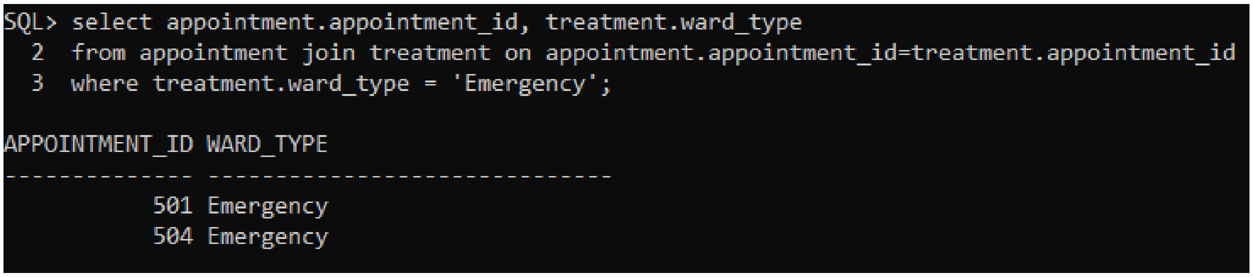
5.2.2. List all the appointments that have been conducted in an emergency ward. (Query 6)

This query selects columns appointment_id from appointment table, and ward_type from treatment table and from the appointment table joins treatment table in which appointment_id of both tables are equal. Then the rows that are selected are restricted by the where clause that indicates ward_type from the treatment table must be an Emergency ward.

SQL> select appointment.appointment_id, treatment.ward_type

2 from appointment join treatment on appointment.appointment_id=treatment.appointment_id

3 where treatment.ward_type = ‘Emergency’;



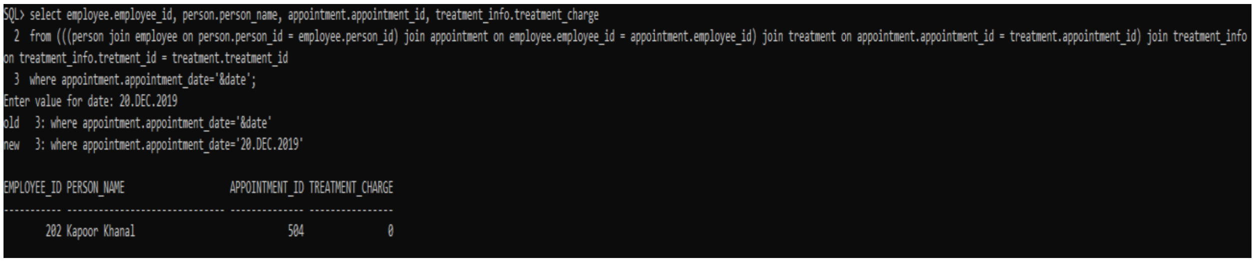
```
SQL> select appointment.appointment_id, treatment.ward_type
  2  from appointment join treatment on appointment.appointment_id=treatment.appointment_id
  3  where treatment.ward_type = 'Emergency';

APPOINTMENT_ID WARD_TYPE
-------------- ------------------------------
           501 Emergency
           504 Emergency
```

Figure 29: Query 6

5.2.3. List all staffs (certified and uncertified) who have conducted or will conduct an appointment on a given date. (Query 7)

This query selects columns employee_id, person_name, appointment_id and treatment_charge and first from person table joins employee table in which person_id of both tables are equal, then this table joins appointment table in which employee_id of both tables employee and appointment table is equal, again this table joins treatment table where appointment_id is equal for both tables appointment and treatment, finally the resulting table joins to treatment_info table in which also treatment_id is the same for both tables treatment_info and treatment. Then the rows that are selected are restricted by the where clause that indicates appointment_date from appointment table must be the same as provided by the user which in this case is ‘20.DEC.2019’.

```
SQL> select employee.employee_id, person.person_name, appointment.appointment_id, treatment_info.treatment_charge

2 from (((person join employee on person.person_id = employee.person_id) join appointment on employee.employee_id = appointment.employee_id) join treatment on appointment.appointment_id = treatment.appointment_id) join treatment_info on treatment_info.treatment_id = treatment.treatment_id

3 where appointment.appointment_date=’&date’;

Enter value for date: 20.DEC.2019

Old 3: where appointment.appointment_date=’&date’

New 3: where appointment.appointment_date = ’20.DEC.2019’
```



```
SQL> select employee.employee_id, person.person_name, appointment.appointment_id, treatment_info.treatment_charge
  2  from (((person join employee on person.person_id = employee.person_id) join appointment on employee.employee_id = appointment.employee_id) join treatment on appointment.appointment_id = treatment.appointment_id) join treatment_info
on treatment_info.tretment_id = treatment.treatment_id
  3  where appointment.appointment_date='&date';
Enter value for date: 20.DEC.2019
old   3: where appointment.appointment_date='&date'
new   3: where appointment.appointment_date='20.DEC.2019'

EMPLOYEE_ID PERSON_NAME                    APPOINTMENT_ID TREATMENT_CHARGE
----------- ------------------------------ -------------- ----------------
        202 Kapoor Khanal                             504                0
```

Figure 30: Query 7

5.2.4. List all patients booked for an appointment on a given date. (Query 8)

This query selects columns patient_id, person_name, appointment_id, and appointment_date and first from person table joins patient table in which person_id of both tables are equal, then this table joins appointment table in which patient_id of both tables patient and appointment table is equal. Then the rows that are selected are restricted by the where clause that indicates appointment_date from appointment table must be the same as provided by the user which in this case is ‘20.DEC.2019’.

SQL> select patient.patient_id, person.person_name, appointment.appointment_id, appointment.appointment_date

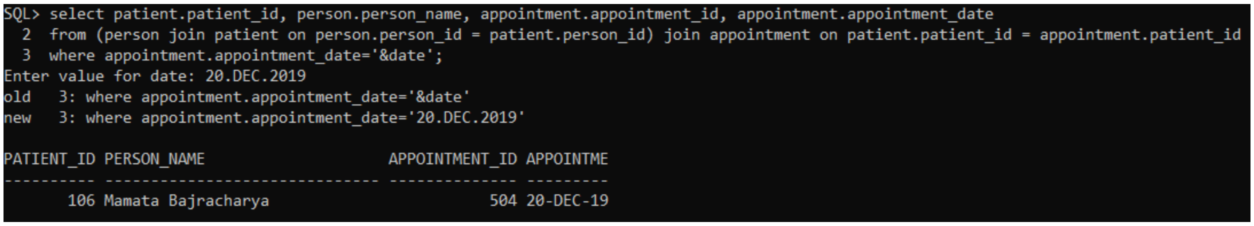
2 from (person join patient on person.person_id = patient.person_id) join appointment on patient.patient_id = appointment.patient_id

3 where appointment.appointment_date=’&date’;

Enter value for date: 20.DEC.2019

Old 3: where appointment.appointment_date=’&date’

New 3: where appointment.appointment_date = ’20.DEC.2019’

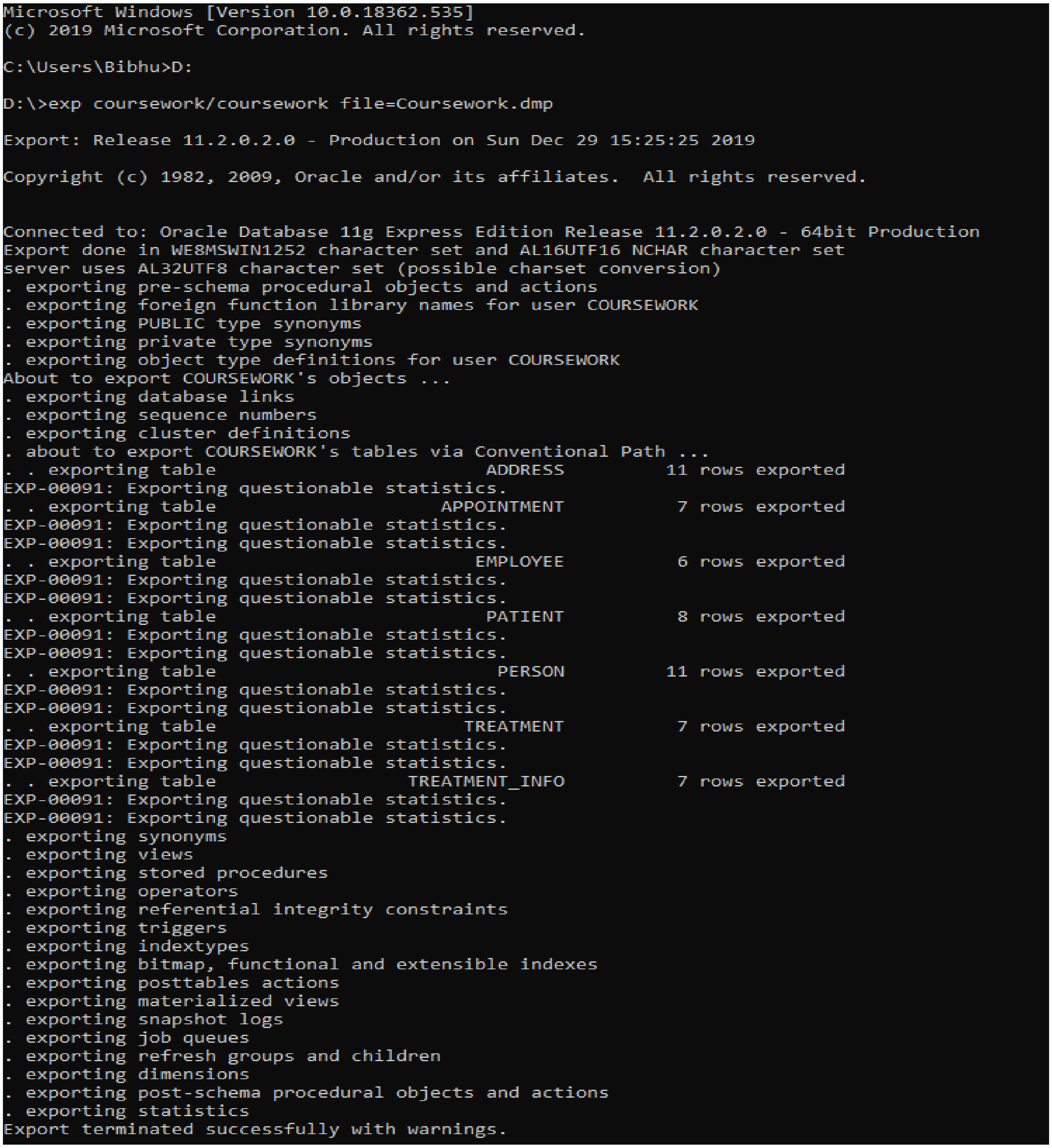


```
SQL> select patient.patient_id, person.person_name, appointment.appointment_id, appointment.appointment_date
  2  from (person join patient on person.person_id = patient.person_id) join appointment on patient.patient_id = appointment.patient_id
  3  where appointment.appointment_date='&date';
Enter value for date: 20.DEC.2019
old   3: where appointment.appointment_date='&date'
new   3: where appointment.appointment_date='20.DEC.2019'

PATIENT_ID PERSON_NAME                    APPOINTMENT_ID APPOINTME
---------- ------------------------------ -------------- ---------
       106 Mamata Bajracharya                        504 20-DEC-19
```

Figure 31: Query 8

5.3. Dump file creation.

Dump file was created using the command prompt with the name Coursework.dmp.

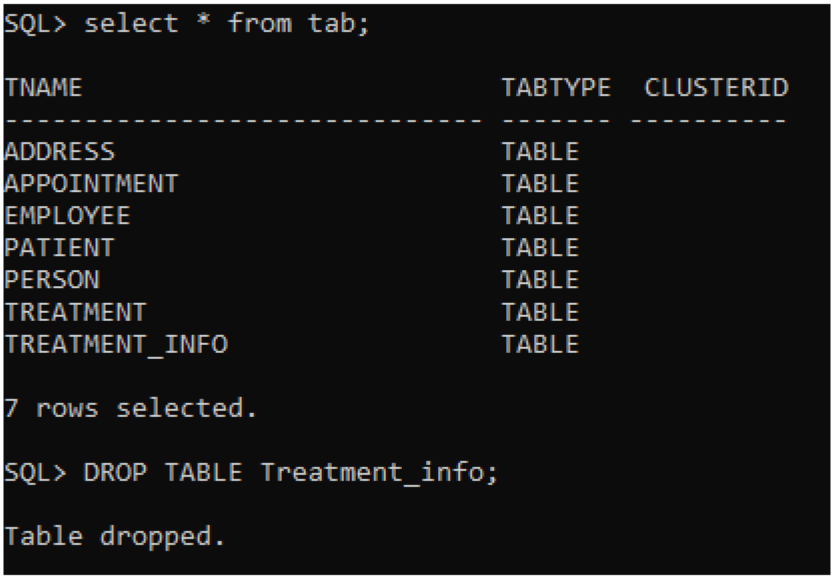


```
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Bibhu>D:

D:\>exp coursework/coursework file=Coursework.dmp

Export: Release 11.2.0.2.0 - Production on Sun Dec 29 15:25:25 2019

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user COURSEWORK
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user COURSEWORK
About to export COURSEWORK's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export COURSEWORK's tables via Conventional Path ...
. . exporting table                        ADDRESS         11 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                    APPOINTMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                       EMPLOYEE          6 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                        PATIENT          8 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         PERSON         11 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                      TREATMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                 TREATMENT_INFO          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting materialized views
. exporting snapshot logs
. exporting job queues
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully with warnings.
```

Figure 32: Dump File Creation

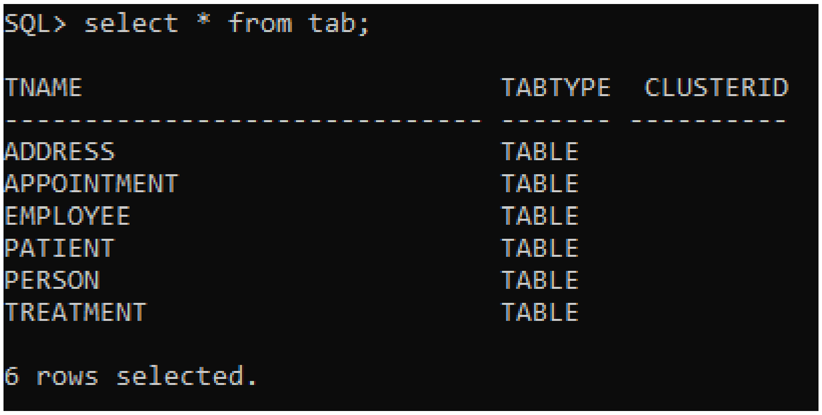
5.4. Deleting all tables in the database in sequential order.

5.4.1. Dropping table Treatment_Info.



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
APPOINTMENT                    TABLE
EMPLOYEE                       TABLE
PATIENT                        TABLE
PERSON                         TABLE
TREATMENT                      TABLE
TREATMENT_INFO                 TABLE

7 rows selected.

SQL> DROP TABLE Treatment_info;

Table dropped.
```

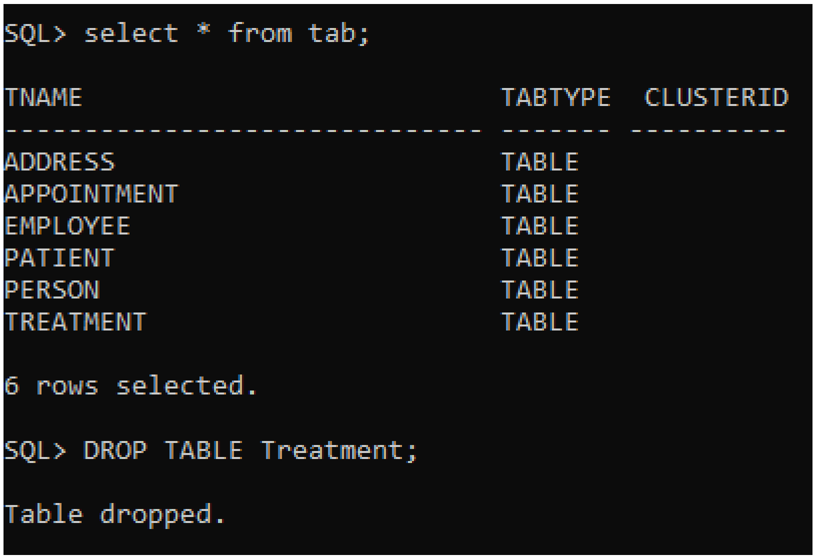
Figure 33: Drop Treatment_Info



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
APPOINTMENT                    TABLE
EMPLOYEE                       TABLE
PATIENT                        TABLE
PERSON                         TABLE
TREATMENT                      TABLE

6 rows selected.
```

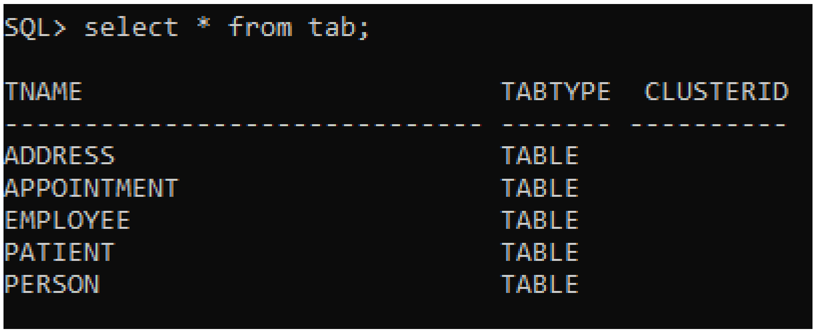
Figure 34: Drop Treatment_Info Result

5.4.2. Dropping table Treatment.



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
APPOINTMENT                    TABLE
EMPLOYEE                       TABLE
PATIENT                        TABLE
PERSON                         TABLE
TREATMENT                      TABLE

6 rows selected.

SQL> DROP TABLE Treatment;

Table dropped.
```

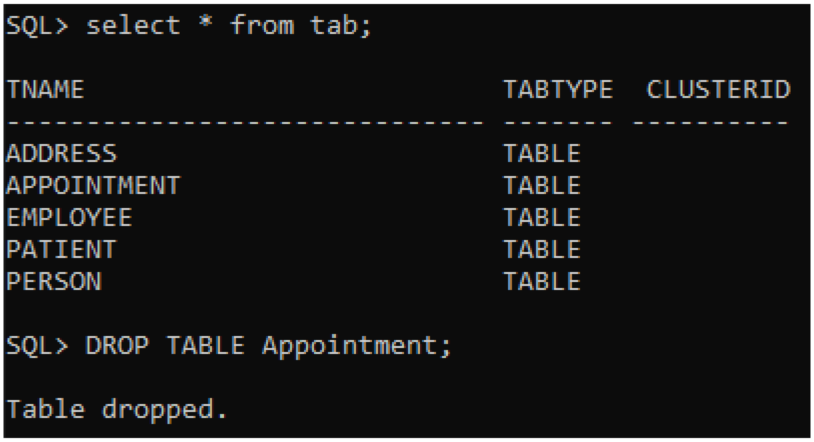
Figure 35: Drop Treatment



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
APPOINTMENT                    TABLE
EMPLOYEE                       TABLE
PATIENT                        TABLE
PERSON                         TABLE
```

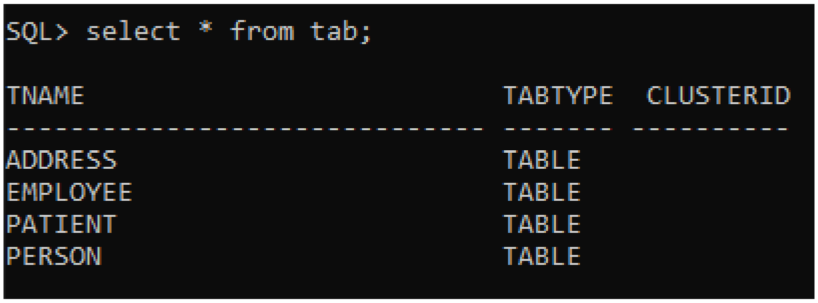
Figure 36: Drop Treatment Result

5.4.3. Dropping table Appointment.



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
APPOINTMENT                    TABLE
EMPLOYEE                       TABLE
PATIENT                        TABLE
PERSON                         TABLE

SQL> DROP TABLE Appointment;

Table dropped.
```

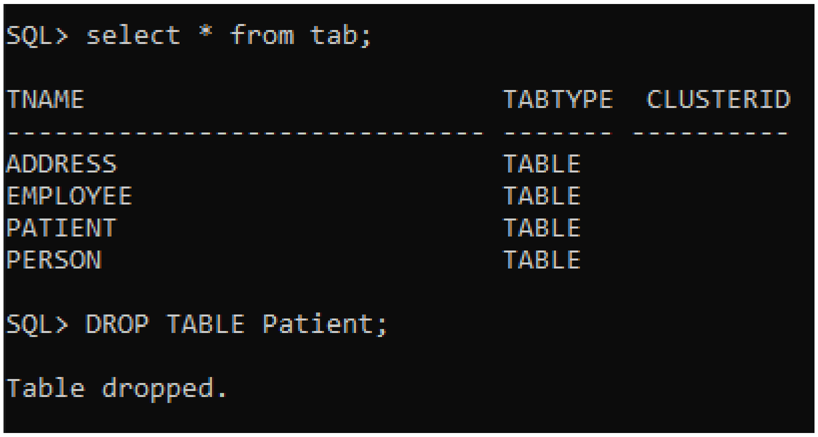
Figure 37: Drop Appointment



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
EMPLOYEE                       TABLE
PATIENT                        TABLE
PERSON                         TABLE
```

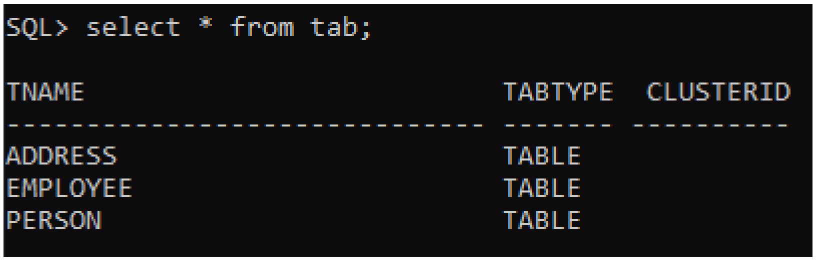
Figure 38: Drop Appointment Result

5.4.4. Dropping table Patient.



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
EMPLOYEE                       TABLE
PATIENT                        TABLE
PERSON                         TABLE

SQL> DROP TABLE Patient;

Table dropped.
```

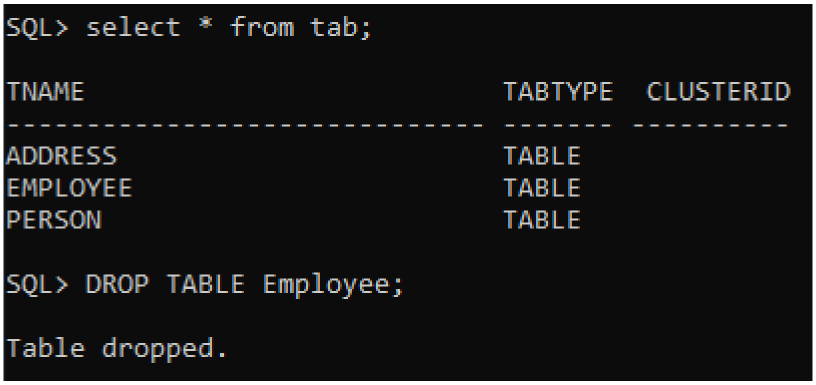
Figure 39: Drop Patient



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
EMPLOYEE                       TABLE
PERSON                         TABLE
```

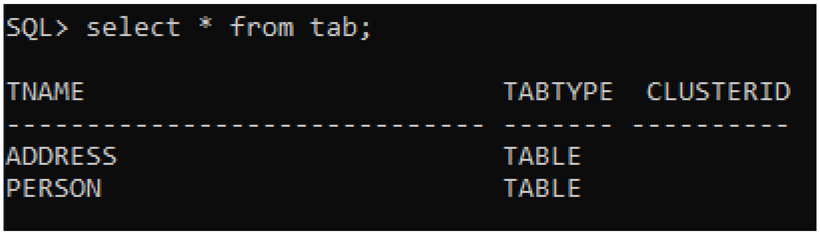
Figure 40: Drop Patient Result

5.4.5. Dropping table Employee.



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
EMPLOYEE                       TABLE
PERSON                         TABLE

SQL> DROP TABLE Employee;

Table dropped.
```

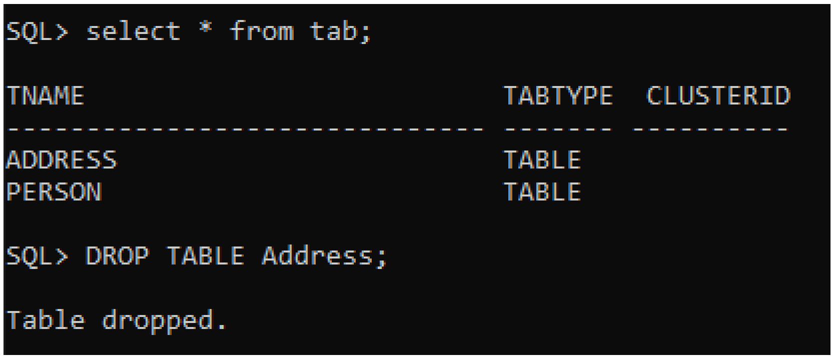
Figure 41: Drop Employee



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
PERSON                         TABLE
```

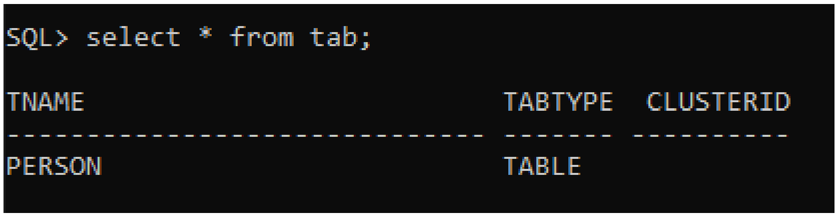
Figure 42: Drop Employee Result

5.4.6. Dropping table Address.



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADDRESS                        TABLE
PERSON                         TABLE

SQL> DROP TABLE Address;

Table dropped.
```

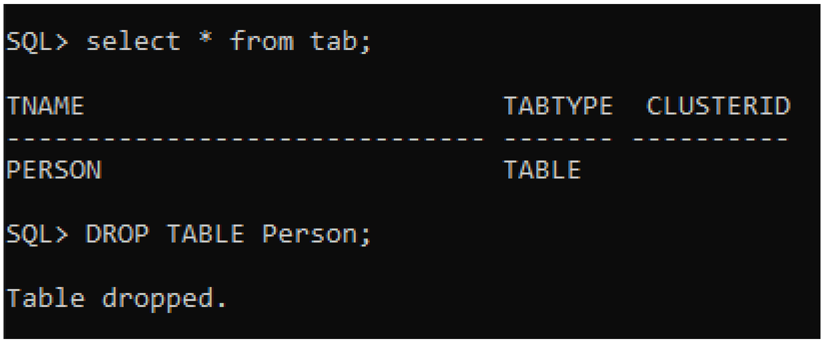
Figure 43: Drop Address



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
PERSON                         TABLE
```

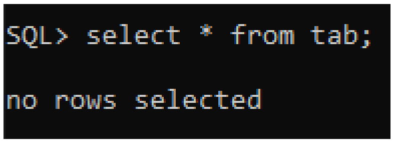
Figure 44: Drop Address Result

5.4.7. Dropping table Person.



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
PERSON                         TABLE

SQL> DROP TABLE Person;

Table dropped.
```

Figure 45: Drop Person



```
SQL> select * from tab;

no rows selected
```

Figure 46: Drop Person Result

6. Critical Evaluation

6.1. Critical Evaluation

From my experience this coursework was really quite tough comparing to what we learned and did our coursework of database in our first year. The knowledge and experience we got from completing the coursework of database in our first year only covered a small amount in assisting us do our current year's coursework. The main objective that I gained was a lot of experience regarding normalization of raw data.

The scenario given to us was about a Patient Record System in a hospital. Hospital seemed to be a vast topic for me which made identifying the entities, attributes and an initial ER-Diagram to be an issue. Upon reading the scenario carefully and reading the guidelines multiple times given in the question while also consulting our teachers I realized it was not that difficult. So finally the issues regarding the entities, attributes and the ER-Diagram was gone.

The main difficulty that fell upon us was the part Normalization. As we lacked the understanding of normalization from UNF to 3NF it became a huge problem. Without Normalization we could not even move forward towards other questions which made us stressed out. But multiple visits to our module leaders and teachers helped a lot in understanding the concept of normalizing the data which eventually helped us complete the part of normalization. As we had experience regarding the creation of tables, data insertion and SQL statements for the given queries, this part did not become a huge complication as it seemed.

Upon completion of the task given in our coursework I can now clearly identify entities, attributes and relations but I still have my doubt regarding normalization which I will remove upon doing more research and consult our teachers. I also expanded the knowledge and experience I had before regarding sql or databases than before.

6.2. Critical Assessment of Coursework

On completing this assessment I learned to create a database for a relatively small sized company and also have gained the confidence that I could create a database even for a medium sized company or organization. I also realized the importance of database and how it can help us in the current emerging digitization of Information all around the world.

The database module also relates to our other two modules which are 'Emerging Programming Platforms and Technologies' and 'Software Engineering'. In the coursework for Emerging Programming Platforms and Technologies we need to store data in a database in which we need to extract data from it which is like a query in some type of way. Whereas on the other hand, the coursework for Software Engineering also requires an ER-Diagram and a database system for a Dental Home Application. This coursework would also help us a lot in completing our other modules coursework with less complexity.

As discussed earlier, I learned a lot about identifying entities, attributes, creating a simple ER-Diagram and normalization raw data from UNF to 3NF. Before this module, I only had the capacity to create a database which were normalized or were few and simple raw data but after the module my capabilities has significantly increased. In conclusion, with this module and coursework I have acquired sufficient skills to successfully create fully functional and working database for my future jobs regarding databases.

References

© Grande International Hospital. (2019, December 20). *Grande International Hospital, Kathmandu, Nepal*. Retrieved from GRANDE INTERNATIONAL HOSPITAL: https://www.grandehospital.com/