HOSPITAL DATABASE MANAGEMENT SYSTEM

Complied by: Sapinderjeet Singh

Student Id: C0824372

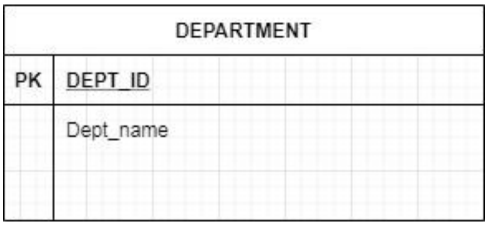
Table of Contents

Introduction

The purpose of this project is to create a Hospital Management System. This project involves various entities related to hospital for instance, the patients and employees. Further, there are different type of employees that is doctor, nurse, patient etc. Moreover, each entity has various attributes which gives a clear view of how the information relating to the hospital will be stored according to the concepts of Database Management System. Also, i have included a discussion of how the relationship between the various entities exists and also the various keys have been identified accordingly. After designing the ERD the normalization will be implemented upto 3rd normal form. Therefore, as an overall structure the designed system would be good enough to store the information as per the required standards. Lastly, the tables will be implemented in the Oracle and to check the working, various meaningful queries are implemented and relevant screenshots are also provided.

ENTITIES

1) Department: The department entity shows the different departments to which doctors belong.



DEPARTMENT	
PK	DEPT_ID
	Dept_name

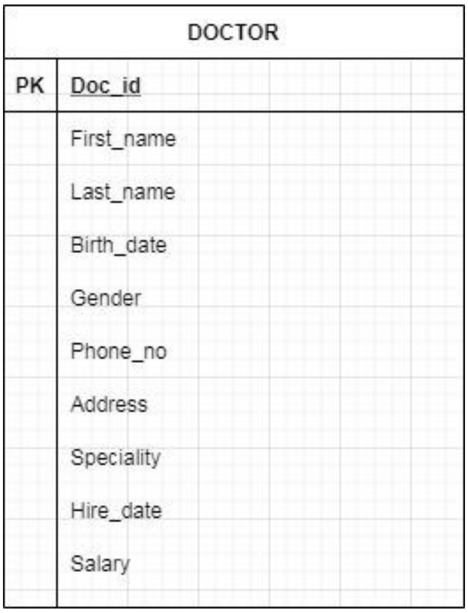
Primary Key: Dept_id

Mandatory attributes: Dept_name

Optional attributes: None

Description of Department attributes
Dept_id: Dept_id is the primary key.
Dept_name: The name of the department whether General doctor or Surgeon

2) Doctor: The Doctor is one of the main entity of the Hospital database system.



DOCTOR	
PK	Doc_id
	First_name Last_name Birth_date Gender Phone_no Address Speciality Hire_date Salary

Primary Key: Doc_id

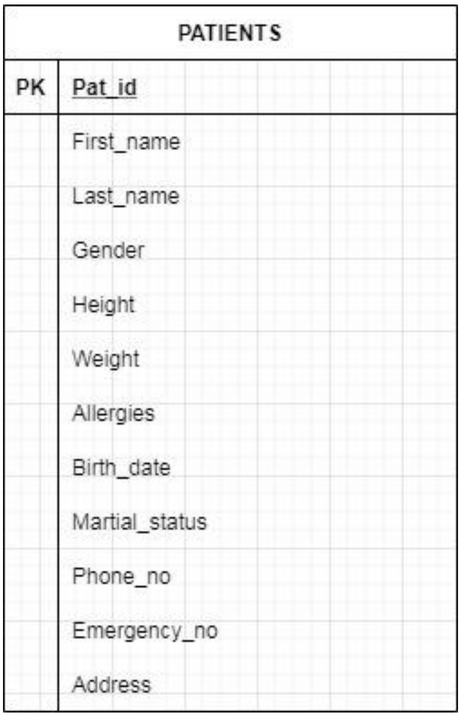
Mandatory attributes: First_name, Phone_no

Optional attributes: Address, Gender

Description of the DOCTOR entity is shown below:

Attributes	Data type	Comments
Doc_id	int	Unique id for a Doctor
First_name	varchar(20)	Doctor’s first name
Last_name	varchar(20)	Doctor’s last name
Speciality	varchar(20)	Doctor’s specialty(for e.g. Surgeon)
Birth_date	Date	Doctor’s date of birth
Gender	Varchar(20)	Whether Male or Female
Address	varchar(20)	Doctor’s address
Phone_no	int	Mobile Number
Hire_date	Date	Doctor’s experience in years
Salary	int	Salary of a doctor

3) **Patient:** This is the primary entity of the hospital database as all other entities like doctor provide service to this entity. It will act as the centre of the database,



PATIENTS	
PK	Pat_id
	First_name
	Last_name
	Gender
	Height
	Weight
	Allergies
	Birth_date
	Martial_status
	Phone_no
	Emergency_no
	Address

Primary Key: Pat_id

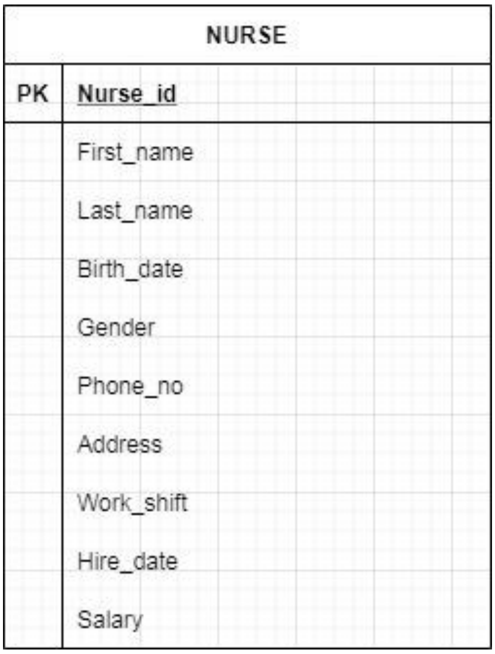
Mandatory attributes: First_name, Emergency_no, Phone_no

Optional attributes: Martial_status, Gender

Description of the Patient entity is shown below:

Attributes	Data type	Comments
Pat_id	int	Unique id for a Patient
First_name	varchar(20)	Patient’s first name
Last_name	Varchar(20)	Patient’s last name
Height	Int	Patient’s height
Weight	Int	Patient’s weight
Allergies	Varchar(20)	Allergy if any
Gender	varchar(20)	Patient is Male or Female
Birth_date	Date	Date of Birth
Marital Status	Varchar(20)	Whether married or unmarried
Phone_no	int	Mobile Number
Emergency Contact no	Int	Emergency no should be different from the Phone_no
Address	varchar(20)	Patients address

4) Nurse: The Nurse is an employee in the hospital which assist Doctor during an operation and also checks on patient.



	NURSE
PK	**Nurse_id**
	First_name
	Last_name
	Birth_date
	Gender
	Phone_no
	Address
	Work_shift
	Hire_date
	Salary

PRIMARY key: Nurse_id

Mandatory attributes: First_name, Work_shift Phone_no

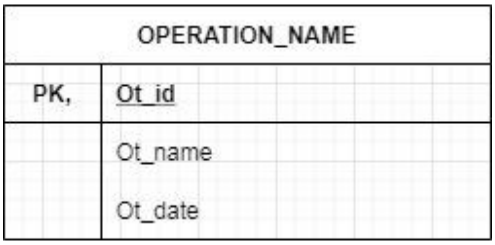
Optional attributes: Gender

Description of the Nurse entity is shown below:

Attribute	Data type	Comments
Nurse_id	int	Unique id for a Nurse
First_name	varchar(20)	Nurse’s first Name
Last_name	Varchar(20)	Nurse’s Last name
Gender	Varchar(10)	Whether male or female
Birth_date	Date	Nurse’s Date of birth
Phone_no	int	Mobile Number
Address	varchar(20)	Nurse’s Address
Work_shift	varchar(20)	Shift e.g. = morning, evening, night
Hire_date	Date	Hire date
salary	int	Salary of a Nurse

5) Operation: Operations are conducted in the hospital and therefore it is necessary to store all data related to an operation.

The entity diagram for OPERATION is shown below:



OPERATION_NAME	
PK,	Ot_id
	Ot_name Ot_date

PRIMARY key: Ot_id

Mandatory attributes: Ot_date, Ot_name

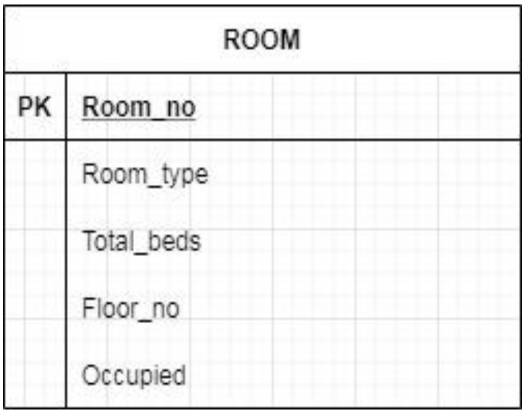
Optional attributes: None

Description of the OPERATION entity is shown below:

Attributes	Data type	Comments
Ot_id	int	Unique id for an Operation Theatre (OT)
Ot_name	Varchar(20)	Name of the operation which are conducted in the hospital for e.g. Dialysis.
Ot_date	date	Date of the operation

6) Room: Room is also an important entity as sick patient are admitted here.

The entity diagram for Room is shown below:



ROOM	
PK	Room_no
	Room_type
	Total_beds
	Floor_no
	Occupied

Primary key: Room_no

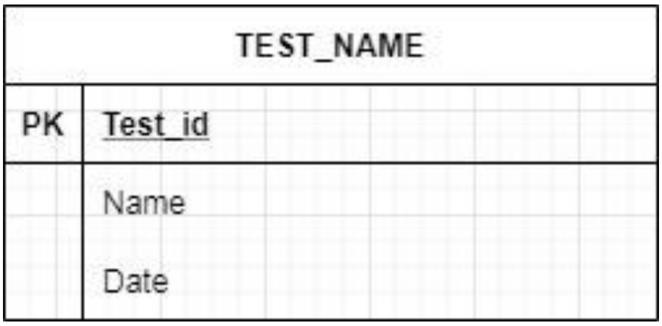
Mandatory attributes: Total_beds, Occupied, Floor_no, Room_type

Optional attributes: None

Description of the ROOM entity is shown below:

Attributes	Data type	Comments
Room_no	int	Room number which will act as the primary key.
Room_type	varchar(20)	Room is VIP or Normal
Floor_no	int	On which floor the room is.
Total_beds	Int	Total beds in a room.
Occupied	int	Total beds occupied which are not vacant.

7) Test: In a hospital various types of test are conducted to assess the health of a patient.



TEST_NAME	
PK	Test_id
	Name Date

Primary key: Test_id

Mandatory attributes: Name, Date

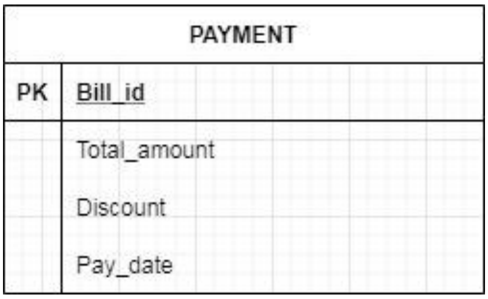
Optional attributes: None

Description of the Test entity is shown below:

Attributes	Data type	Comments
Test_id	int	Unique id for a Test
Name	varchar(20)	Name of the Test. For e.g. Blood test etc.
Date	Date	Date and time of Test

8) Payment: Whenever a patient is discharged from the hospital has to clear the bill and therefore, it is necessary to have a record of all the transactions.

The entity diagram for Payment is shown below:



PAYMENT	
PK	Bill_id
	Total_amount Discount Pay_date

Primary key: Bill_id

Mandatory attributes: Total_amount, Pay_date

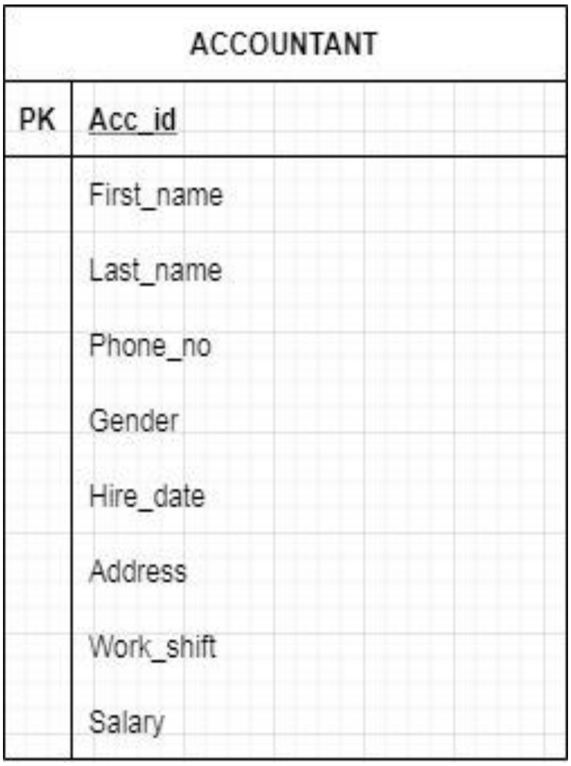
Optional attributes: Discount

Description of the Test entity is shown below:

Attributes	Data type	Comments
Bill_id	int	Unique id for a payment
Total_amount	Int	Total bill of a patient.
Discount	Int	Discount offered
Pay_date	Date	Date of the payment

9) Accountant: In a hospital accountant takes care of all the transactions.

The entity diagram for ORDER is shown below:



ACCOUNTANT	
PK	Acc_id
	First_name Last_name Phone_no Gender Hire_date Address Work_shift Salary

Primary key: Acc_id

Mandatory attributes: First_name,
Phone_no,
Work_shift.

Optional attributes: Address

Description of the Nurse entity is shown below:

Attribute	Data type	Comments
Acc_id	int	Unique id for an Accountant
First_name	varchar(20)	Accountant first Name
Last_name	Varchar(20)	Accountant’s Last name
Gender	Varchar(10)	Whether male or female
Phone_no	int	Mobile Number
Address	varchar(20)	Accountant’s Address
Work_shift	varchar(20)	Shift e.g. = morning, evening, night
Hire_date	Date	Hire date
salary	int	Salary of an Accountant

NOTE: It is understandable that all employees should have a single table but to reduce complexity the Doctor and Nurse are taken as separate entities as they connect with almost all the entities present in this database and after Doctor and Nurse only one Accountant table was left which works in a hospital therefore it is also taken as a separate table.

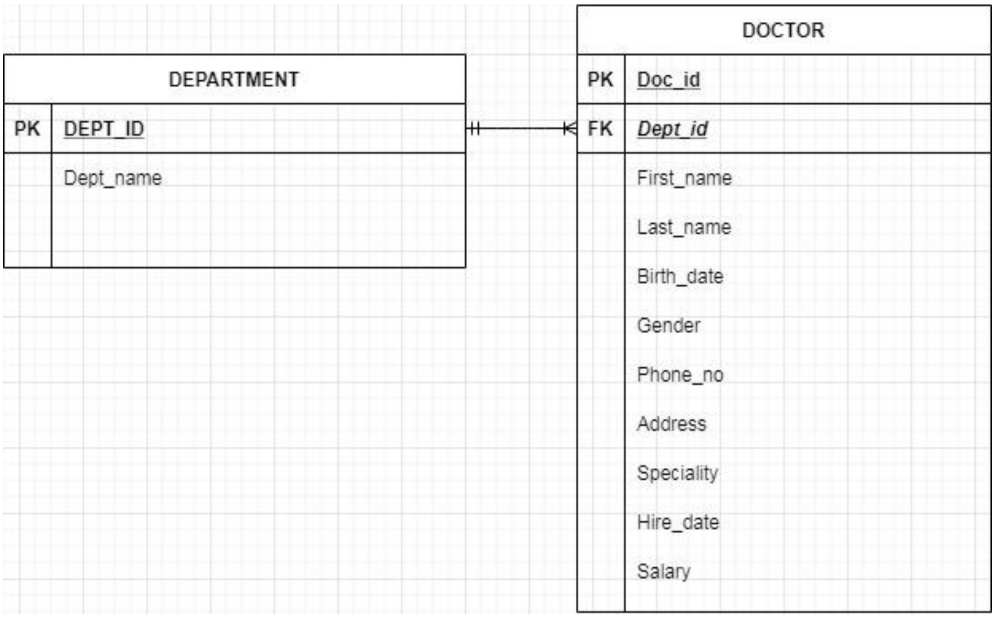
NOTE: Other entities which should also be present in a hospital database but were ignored to reduce complexity are:

- Ambulance
- Driver
- Medicine
- Prescription
- Ward boy
- Receptionist

RELATIONSHIPS

I. Relationship between Department and Doctor:

- One doctor can belong to one department only.
- One department can have one or more doctors.



II. Relationship between Doctor and Patient:

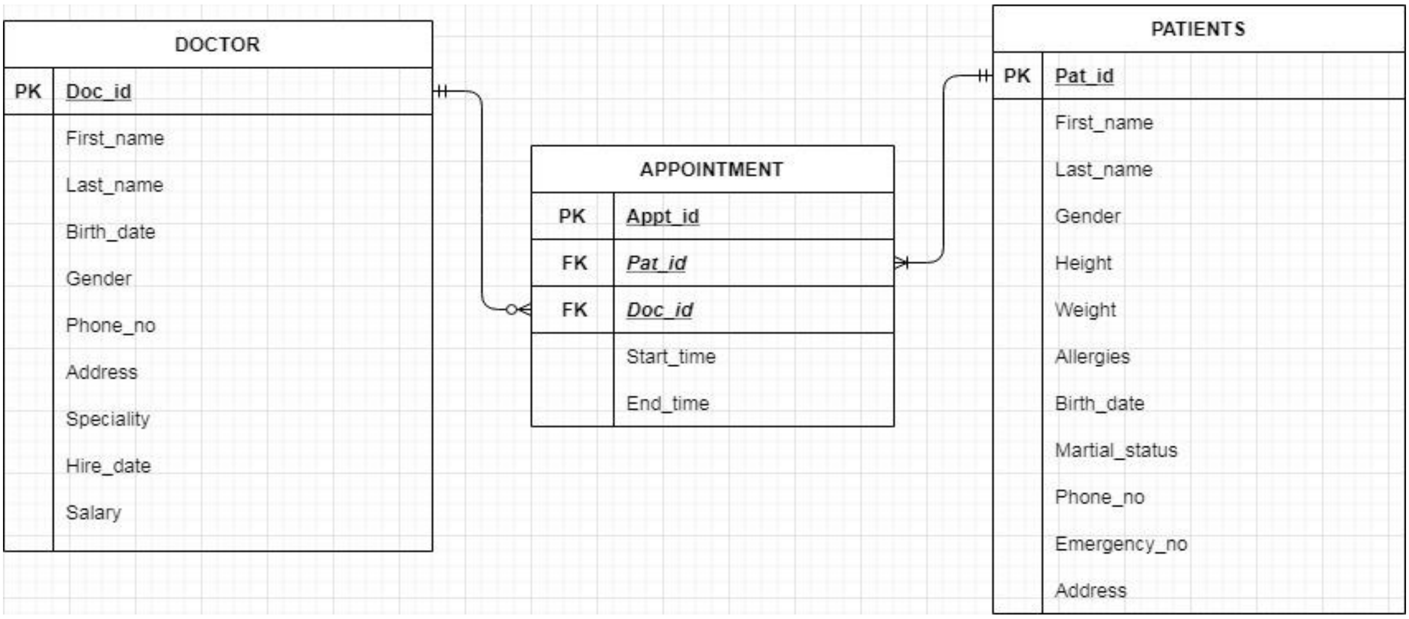
- One doctor can examine zero, one or many patients.
- One patient can see one or many doctors in a day.

Therefore, to resolve this many to many relationships a junction table APPOINTMENT is created.

Appointment table: Primary key: Appt_id

Foreign key: Pat_id and Doc_id

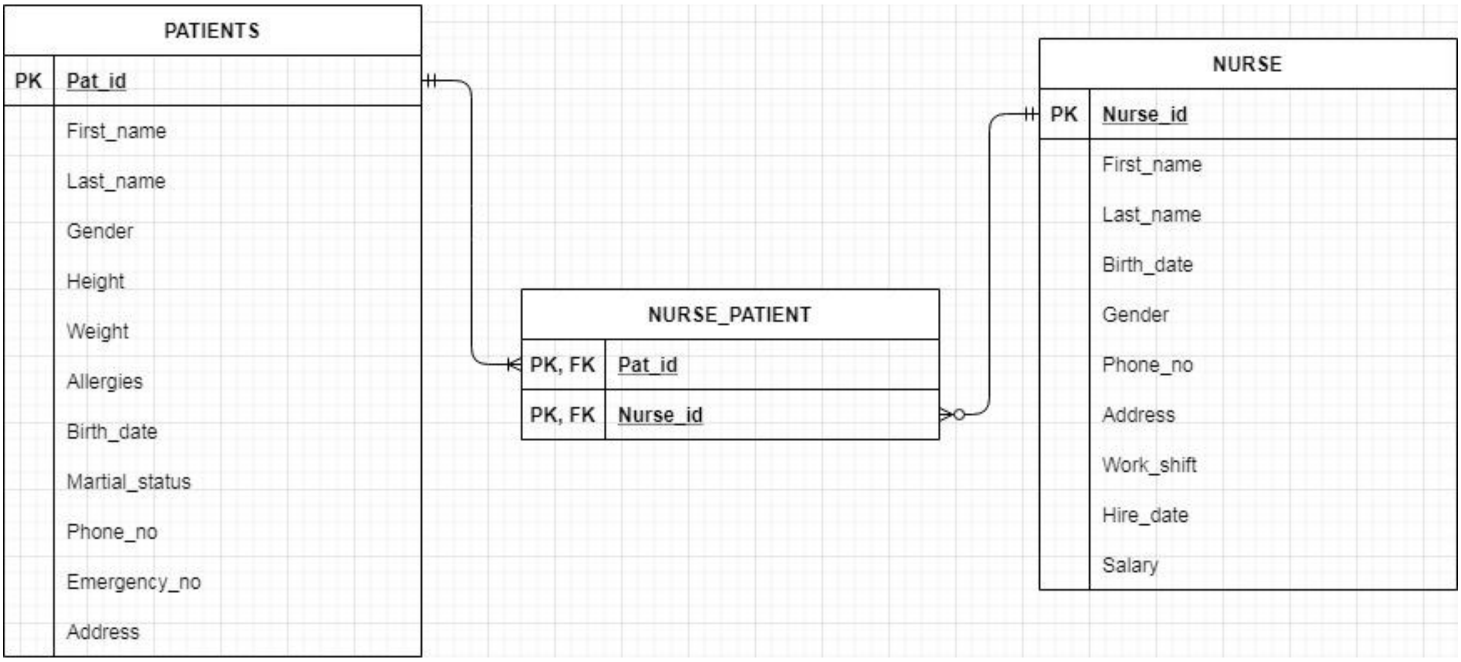
Other attributes: Start_time and End_time (of appointment)



III. Relationship between Patient and nurse:

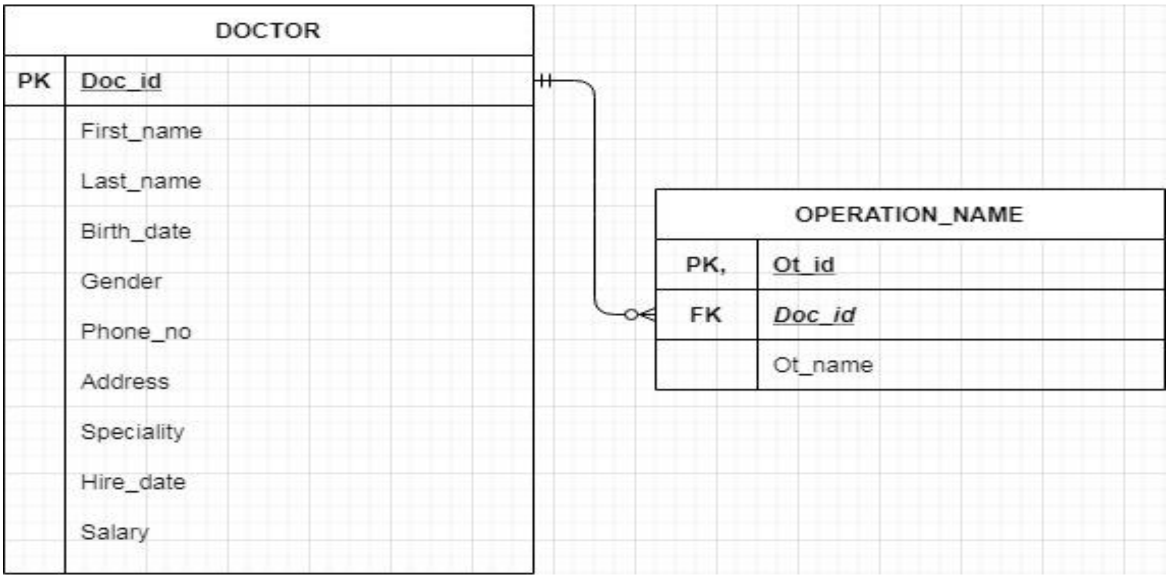
- One nurse can attend zero, one or many patients.
- One patient can be checked by one or many nurses.

Therefore, a junction table is created to resolve this many to many relationships.



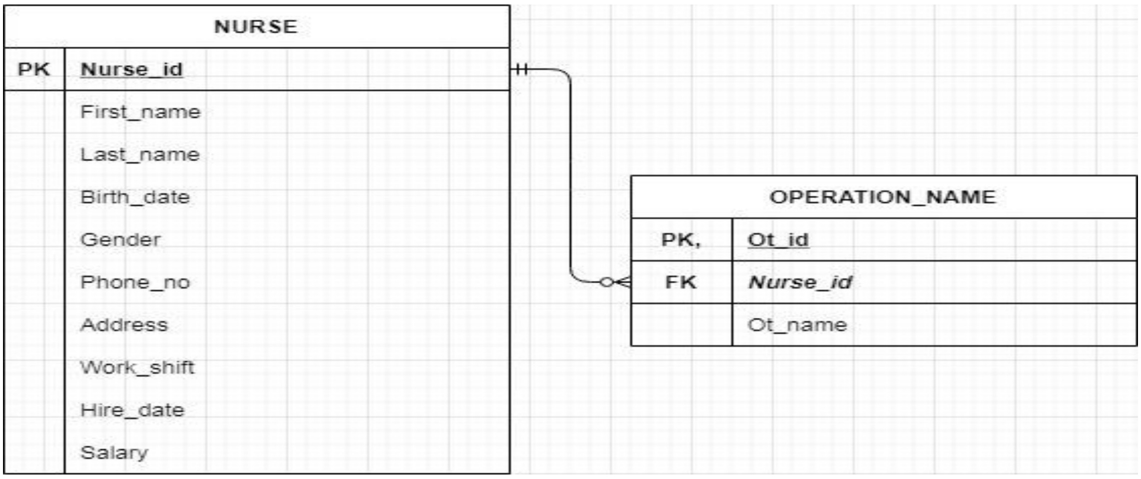
IV. Relationship between Operation and Doctor:

- One Doctor can do zero, one or more operation.
- One operation is done by one doctor.
  (Note: to reduce complexity)



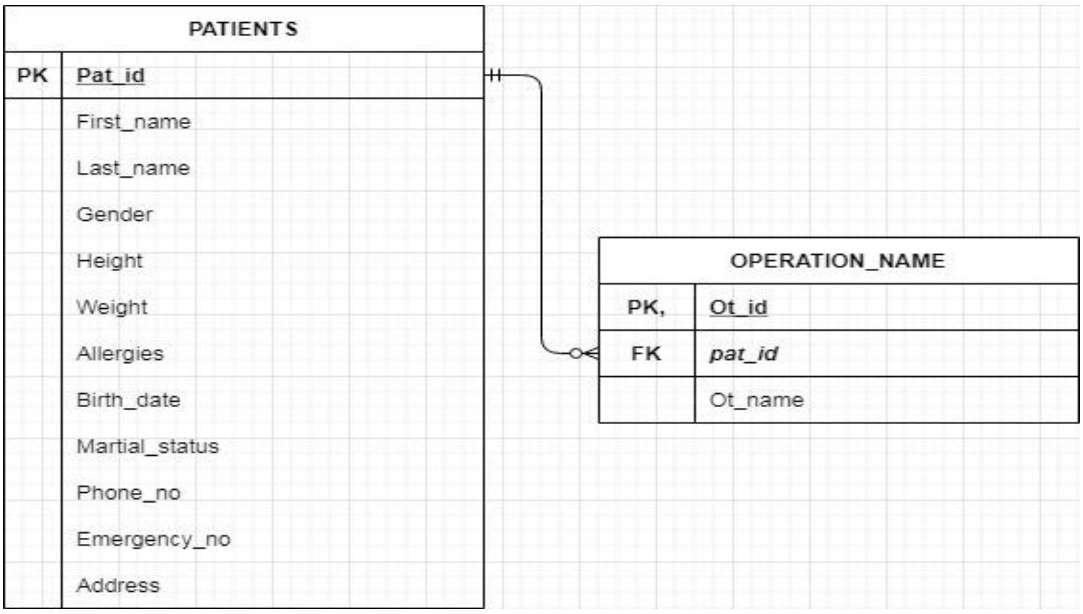
V. Relationship between Operation and Nurse:

- One Nurse can assist in zero, one or many operations.
- One operation can be assisted by one nurse.
  (Note: to reduce complexity)



VI. Relationship between Operation and Patient:

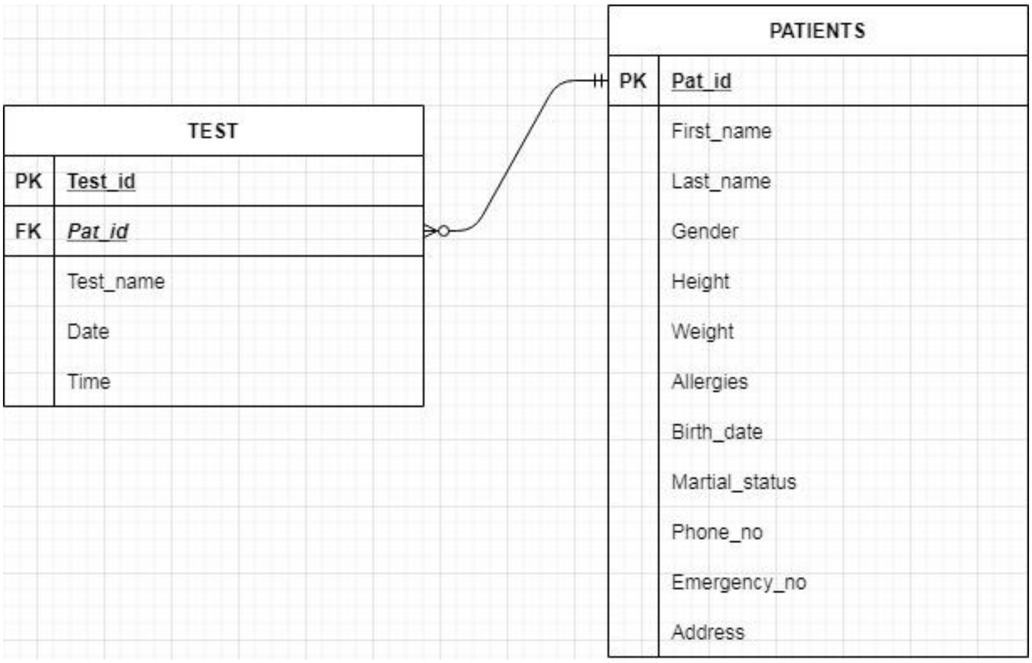
- One patient can have zero, one or more operation in a day.
- One operation is done on one patient.



PATIENTS	
PK	Pat_id
	First_name
	Last_name
	Gender
	Height
	Weight
	Allergies
	Birth_date
	Martial_status
	Phone_no
	Emergency_no
	Address

OPERATION_NAME	
PK,	Ot_id
FK	*pat_id*
	Ot_name

VII. Relationship between Test and Patient



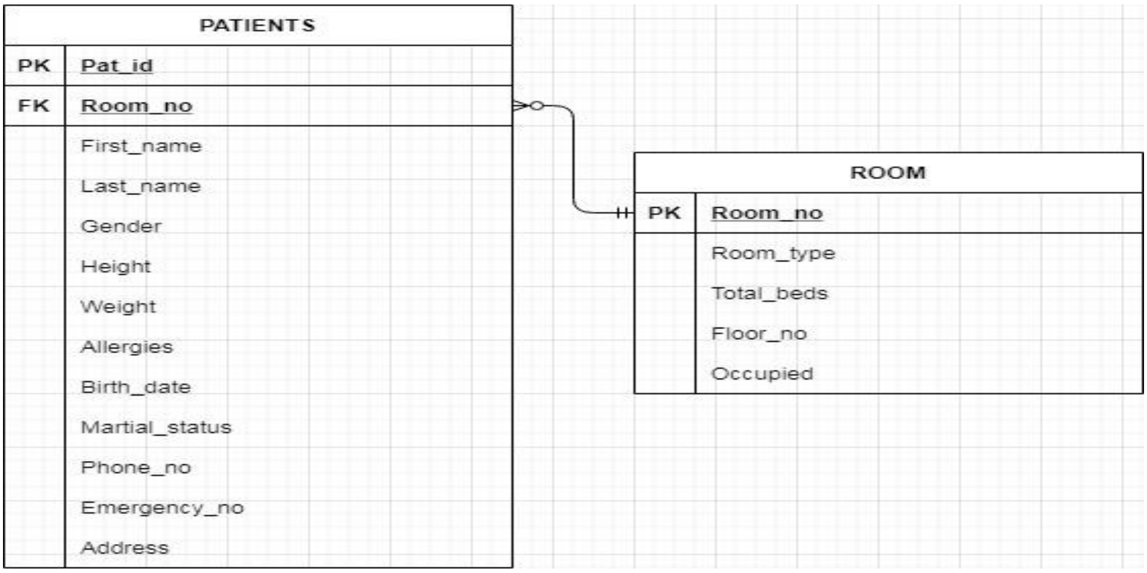
TEST	
PK	Test_id
FK	*Pat_id*
	Test_name
	Date
	Time

PATIENTS	
PK	Pat_id
	First_name
	Last_name
	Gender
	Height
	Weight
	Allergies
	Birth_date
	Martial_status
	Phone_no
	Emergency_no
	Address

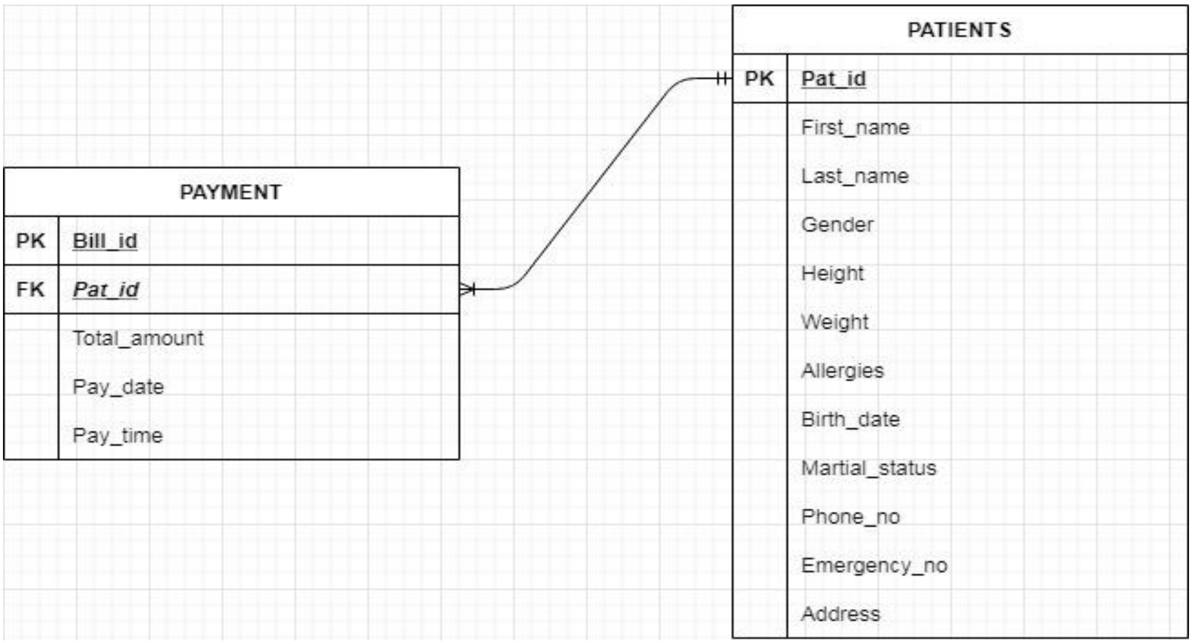
- One patient can undergo zero, one or many test.
- Each test report belongs to one patient only.

VIII. Relationship between Room and Patient:

- One room can have zero, one or many patients.
- One patient can be assigned one room only.

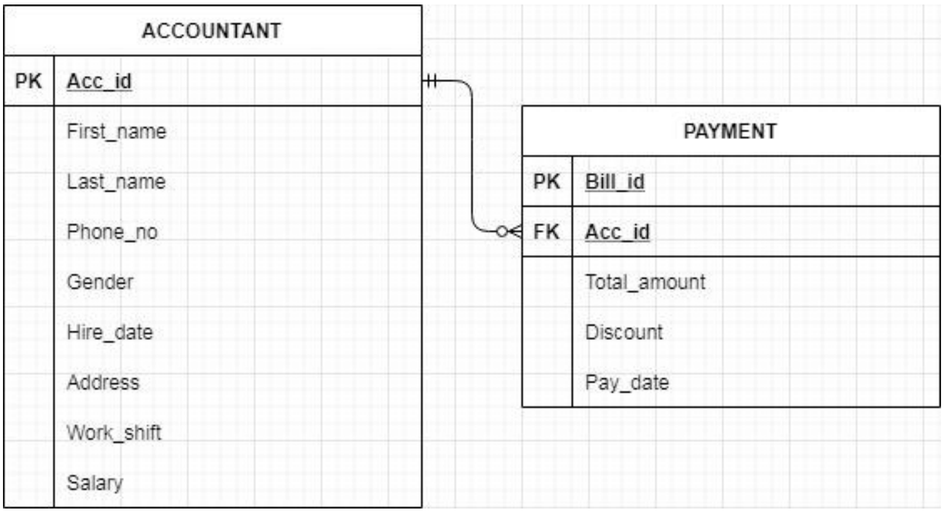


IX. Relationship between Bill and patient.



- One patient can have one or many bills.
- Each bill is paid by one patient.

X. Relationship between Payment and accountant:



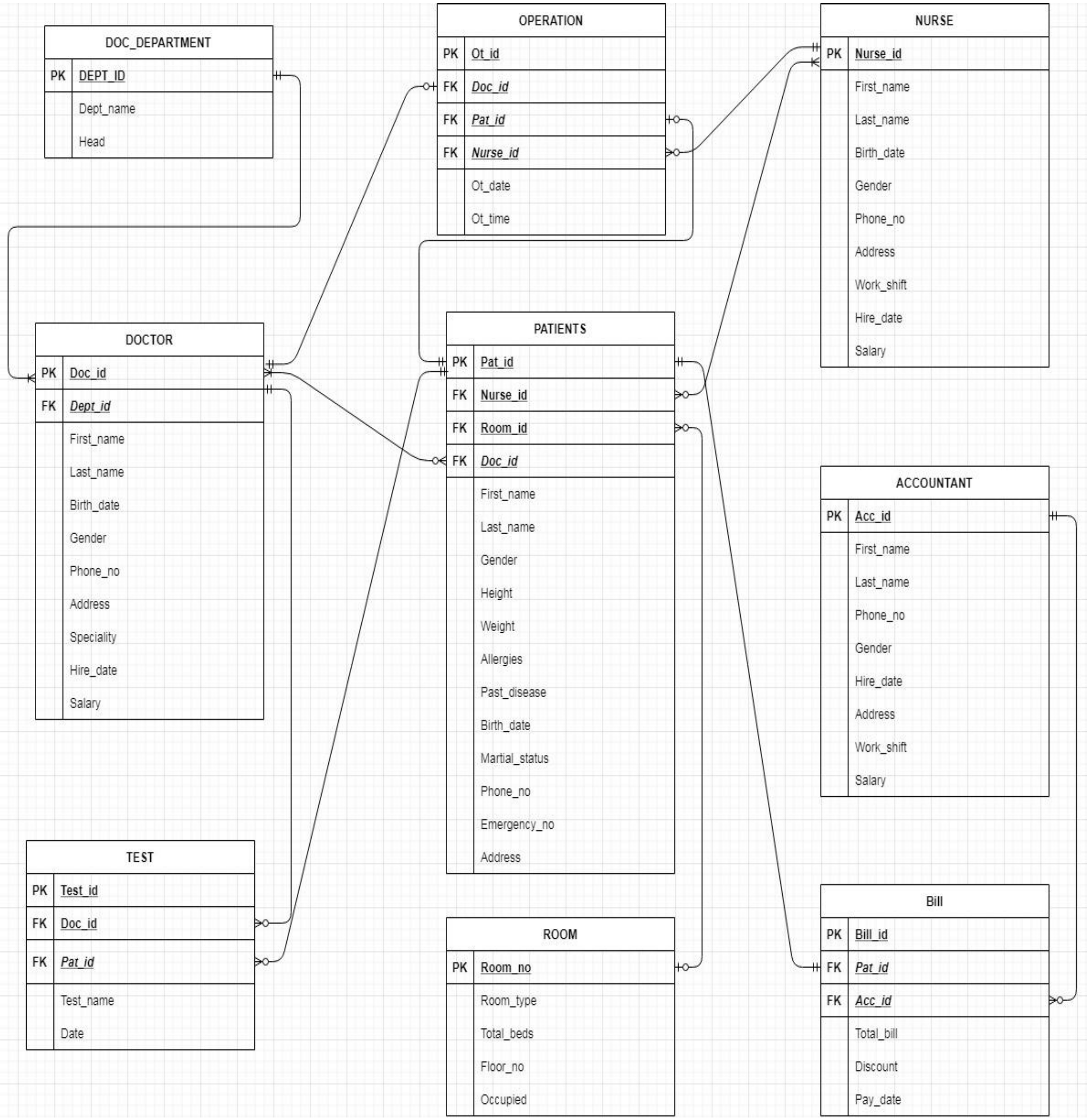
- One accountant can receive zero, one or many payments.
- One bill is received by one accountant only.

Relationship matrix

	Department	**Doctor**	**Patient**	**Nurse**	**Room**	**Operation**	**Test**	**Accountant**	**Payment**
Department		has							
Doctor	Belongs to		Examines			do	prescribe		
Patient		Examined by		Checked by	Admitted in		Takes		pays
Nurse			Checks on			Assist in			
Room			has			host			
Operation		Done by	Done on	Assisted by					
Test		Prescribed by	Taken by						
Accountant									receives
Payment			Done by					Received by	

Entity Relationship Diagram(ERD):

Before Normalization



Normalization:

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

It has mainly two goals: -

- ✓ First goal: eliminate redundant data.
- ✓ Second Goal: ensure data dependencies make sense.

Benefits of Normalization:

- Less storage space
- Quicker updates
- Less data inconsistency
- Clearer data relationships
- Easier to add data
- Flexible Structure

Bad database designs result in:

- Redundancy: inefficient storage.
- Anomalies: data inconsistency, difficulties in maintenance.

1NF, 2NF, 3NF are some of the early forms in the list that address this problem.

RULES OF NORMAL FORM:

❖ First normal form(1NF):

- No repeating groups
- No multi-valued columns
- A primary key has been defined
- All columns in the table are dependent on primary key

❖ Second normal form(2NF):

- Should already be in 1NF.
- No partial dependencies i.e. All non-key columns are fully dependent on the entire primary key.

❖ Third normal form(3NF):

- Should already be in 2NF.
- A non-key column cannot determine the value of another non-key column must depend directly on primary key

Normalization (Tables):

1. DEPARTMENT:

Dept_id	Dept_name
1	General
2	Surgeon

- **First Normal Form (1NF):**

- No repeating groups could occur for the given table because every cell of the Department table is different from each other.
- No multi-valued columns could appear to exist because every department has a single set of information to be stored.
- A primary key "**DEPT_ID**" has been defined.
- All columns in the table are dependent on primary key because each product holds a unique set of information.

Therefore, the given table is already in First Normal Form. So, now we could move further to check the table for 2NF.

- **Second Normal Form (2NF):**

- As we have concluded above that given table is already in 1NF, therefore it satisfies the first condition for 2NF.
- Also, no partial dependencies occur in the DEPARTMENT table because every set of non-key elements are fully dependent on the entire primary key.
- This means every Department name holds a unique set of information which does not co-relates with any other one.

- **Third Normal Form (3NF):**

- As we have concluded above that given table is already in 2NF, therefore it satisfies the first condition for 3NF.

- A non-key column cannot determine the value of another non-key because clearly each department has a unique department id. Hence, no transitive dependency occurs. This table is in 3NF.

2) DOCTOR: This table contain the sample data as follow and we will check whether it matches with the different criteria of three normal form of normalization.

Doc_id	First_name	Last_name	Gender	Birth_date	Phone_no	Address	Speciality	Hire_date	Salary	***Dept_id***
1	Sara	Davis	Male	3/06/1972	9803438765	Mumbai	Orthopaedist	5/06/2018	300000	1
2	Ross	Taylor	Male	6/03/1986	9666638765	Delhi	Surgeon	9/07/2017	200000	2
3	Jaspreet	Kaur	Female	17/10/1967	9781370121	Nabha	Heart Specialist	14/09/2015	400000	2
4	Manoj	Singla	Male	16/02/1984	9764646464	Delhi	Surgeon	15/01/2020	100000	2
5	Deepak	Sharma	Male	11/09/1967	9805558765	Delhi	Orthopaedist	5/11/2016	350000	1

- **First Normal Form (1NF):**

  - No repeating groups could occur for the given table because every cell of the Doctor table is different from each other.
  - No multi-valued columns could appear to exist because every doctor has a single set of information to be stored.
  - A primary key "**Doc_id**" has been defined.

Therefore, the given table is already in First Normal Form. So, now we could move further to check the table for 2NF.

- **Second Normal Form (2NF):**

  - As we have concluded above that given table is already in 1NF, therefore it satisfies the first condition for 2NF.

- Also, no partial dependencies occur in the DOCTER table because every set of non-key elements are fully dependent on the entire primary key.

- **Third Normal Form (3NF):**

  - As you see above, that given table is already in 2NF, therefore it matches the first condition for 3NF.
  - A non-key column cannot determine the value of another non-key because already the given table has one foreign key i.e. department id, which reduces its data redundancy up to the greatest extent.
  - Therefore, no Transitive Dependency. Hence, this table is in 3NF.

3) Patient: This table contain the sample data as follow and we will check whether it matches with the different criteria of three normal form of normalization.

Pat_id	First_name	Last_name	Gender	Height	Weight	Allergies	Birth_date	Martial_status	Phone_no	Emergency_contact_no	Address	*Room_no*
1	Rajiv	Sharma	Male	5.5	48	Latex	10/01/1988	Married	9803438765	9458966632	Sangrur	234
2	Neetu	Kaur	Female	5.8	58	Null	10/11/1968	Unmarried	9807898765	9455566632	Rajpura	372
3	Ravi	Kumar	Male	6.0	80	Dust Mites	15/05/2010	Married	9804568765	9888966632	Patiala	373
4	Rajnesh	Kumar	Male	5.11	47	Null	15/05/2004	Married	9866638765	9333966632	Chandigarh	375
5	Navpreet	Kaur	Female	5.6	50	Null	10/11/1978	Unmarried	9781370121	9666966632	Nabha	276

- **First Normal Form (1NF):**

  - No repeating groups could occur for the given table because every cell of the Patient table is different from each other.
  - No multi-valued columns could appear to exist because every patient has a single set of information to be stored.
  - A primary key "**Pat_id**" has been defined.

Therefore, the given table is already in First Normal Form. So, now we could move further to check the table for 2NF.

- **Second Normal Form (2NF):**

  - As we have concluded above that given table is already in 1NF, therefore it satisfies the first condition for 2NF.
  - Also, no partial dependencies occur in the PATIENT table because every set of non-key elements are fully dependent on the entire primary key.

- **Third Normal Form (3NF):**

  - As you see above, that given table is already in 2NF, therefore it matches the first condition for 3NF.
  - A non-key column cannot determine the value of another non-key because already the given table has one foreign key i.e. room_no, which reduces its data redundancy up to the greatest extent.
  - Therefore, no Transitive Dependency. Hence, this table is in 3NF.

4) Nurse: This table contain the sample data as follow and we will check whether it matches with the different criteria of three normal form of normalization.

Table is shown below:

Nurse_id	First_name	Last_name	Gender	Birth_date	Phone_no	Address	Work_shift	Hire_date	Salary
1	Savita	Kaur	Female	10/01/1977	9803438765	Delhi	Morning	5/06/2018	10,000
2	Raj	Singh	Male	10/01/1978	9666638765	Rajnagar	Evening	5/06/2016	20,000
3	Rasel	Patel	Female	10/01/1999	9999777346	Ropar	Night	5/06/2020	8000
4	Karim	Sharma	Male	10/01/1978	9764646464	Gurdaspur	Morning	5/06/2017	20000
5	Jhanvi	Sharma	Female	10/01/1977	9805558765	Nabha	Morning	5/06/2010	30000

- ## First Normal Form (1NF):

  - No repeating groups could occur for the given table because every row of the Nurse table is relatively different in nature.
  - No multi-valued columns exist because every column has a single set of information to be stored.
  - A primary key “**Nurse_id**” has been defined.

Therefore, the given table is already in First Normal Form. So, now we could move further to check the table for 2NF.

- ## Second Normal Form (2NF):

  - As we have seen above that given table is already in 1NF, therefore it matches the first condition for 2NF.
  - Also, no partial dependencies occur in the Nurse table because every non-key element is fully dependent on the entire primary key.

- **Third Normal Form (3NF):**

  - As we have seen above that given table is already in 2NF, therefore it satisfies the first condition for 3NF.
  - A non-key column cannot determine the value of another non-key attribute.

5) Operation: This table contain the sample data as follow and we will check whether it matches with the different criteria of three normal form of normalization.

<u>Ot_id</u>	Ot_name	Ot_date	*Doc_id*	*Pat_id*	*Nurse_id*
1	Dialysis	04-07-2021, 11-07-2021	3	2,3	5
2	Haemorrhoidectomy	05-07-2021	2	2	2
3	Joint Replacement	19-07-2021, 14-07-2021	3	4,5	3

- **First Normal Form (1NF):**

  - AS we can see that when one operation is done on two patients on two different dates the multivalued columns will exist. Therefore, to resolve this we will do as shown below.

<u>Ot_id</u>	Ot_name	Ot_date	*Doc_id*	**<u>Pat_id</u>**	*Nurse_id*
1	Dialysis	04-07-2021	3	2	5
2	Haemorrhoidectomy	05-07-2021	2	2	2
3	Joint Replacement	19-07-2021	3	4	3
1	Dialysis	11-07-2021	5	3	5
3	Joint Replacement	14-07-2021	3	5	3

- ## First Normal Form (1NF):

  - No repeating groups could occur for the given table because every row of the operation table is now relatively different in nature.
  - Multivalued columns which existed previously have been removed.
  - A composite primary key **Ot_id** and **Pat_id** has been identified.

- ## Second Normal Form (2NF):

  - Table is already in first normal form
  - Partial dependency exists in this table as **Ot_name** does not depend on the entire primary key as it only depends on the OT_id. Therefore, a new table will be created named Operation_name.

Ot_id	Ot_name
1	Dialysis
2	Haemorrhoidectomy
3	Joint Replacement

Ot_id	**Pat_id**	Ot_date	*Doc_id*	*Nurse_id*
1	2	04-07-2021	3	5
2	4	05-07-2021	2	2
3	2	19-07-2021	3	3
1	3	11-07-2021	5	5
3	5	14-07-2021	3	3

- ## Third Normal Form (3NF):

  - The given table is already in 2NF; therefore, it satisfies the first condition for 3NF.
  - No Transitive Dependency occurs in this table. Therefore, the table follows Third Normal Form.

6) TEST:

Test_id	Test_Name	Date	*Doc_id*	*Pat_id*
1	Blood	10/06/2021, 11/06/2021	3	4,1
2	Urine	08/05/2021	1	3
3	Ultrasound	18/05/2021	2	5
4	X-Ray	10/06/2021	3	4
5	Angiogram	26/06/2021	5	1

- **First Normal Form (1NF):**

  - Multivalued column exists in the above table as doctor can assign same test To different patients. Therefore, the table is not in the first normal form.

Test_id	Test_Name	Date	*Doc_id*	**Pat_id**
1	Blood	10/06/2021	3	4
1	Blood	11/06/2021	3	1
2	Urine	08/05/2021	1	3
3	Ultrasound	18/05/2021	2	5
4	X-Ray	10/06/2021	3	4
5	Angiogram	26/06/2021	5	1

- **First Normal Form (1NF):**

  - No repeating groups could occur for the given table because every row of the operation table is now relatively different in nature.
  - Multivalued columns which existed previously have been removed.
  - A composite primary key **Test_id** and **Pat_id** has been identified.

Therefore, the given table is already in First Normal Form. So, now we could move further to check the table for 2NF.

- **Second Normal Form (2NF):**

  - As we have seen above that given table is already in 1NF, therefore it satisfies the first condition for 2NF.
  - Partial dependency exists in the table as the Test_name does not depend on the entire primary key and it solely depends on the Test_id. Therefore, a new table named Test_name is created.

Test_id	**Pat_id**	*Doc_id*	Date
1	4	3	10/06/2021
1	1	3	11/06/2021
2	3	1	08/05/2021
3	5	2	18/05/2021
4	4	3	10/06/2021
5	1	5	26/06/2021

Test_id	Test_name
1	Blood
2	Urine
3	Ultrasound
4	X-Ray
5	Angiogram

- **Third Normal Form (3NF):**

  - The given table is already in 2NF; therefore, it satisfies the first condition for 3NF.
  - No Transitive Dependency occurs in this table. Therefore, the table follows Third Normal Form.

7) Accountant: This table contain the sample data as follow and we will check whether it matches with the different criteria of three normal form of normalization.

Acc_id	First_name	Last_name	Gender	Phone_no	Address	Work_shift	Hire_date	Salary
1	Surendra	Sharma	Male	9803438765	Delhi	Morning	5/06/2018	6000
2	Joey	Mike	Male	9666638765	Rajnagar	Evening	5/06/2016	7000
3	Sheldon	Cooper	Male	9999777346	Ropar	Night	5/06/2020	6000
4	Priyanka	Chopra	Female	9764646464	Gurdaspur	Morning	5/06/2017	6000

- **First Normal Form (1NF):**
  - No repeating groups could occur for the given table because every row of the Accountant table is relatively different in nature.
  - No multi-valued columns exist because every column has a single set of information to be stored.
  - A primary key "**Acc_id**" has been defined.

Therefore, the given table is already in First Normal Form. So, now we could move further to check the table for 2NF.

- **Second Normal Form (2NF):**
  - As we have seen above that given table is already in 1NF, therefore it matches the first condition for 2NF.
  - Also, no partial dependencies occur in the Accountant table because every non-key element is fully dependent on the entire primary key.

- **Third Normal Form (3NF):**
  - As we have seen above that given table is already in 2NF, therefore it satisfies the first condition for 3NF.
  - A non-key column cannot determine the value of another non-key attribute.

8) Appointment: This table contain the sample data as follow and we will check whether it matches with the different criteria of three normal form of normalization.

Appt_id	*Pat_id*	*Doc_id*	Start_time	End_time
132	1	1	18-06-2021 10:00	18-06-2021 11:00
265	2	2	18-06-2021 10:00	18-06-2021 11:00
365	1	1	18-06-2021 11:00	18-06-2021 12:30
468	4	4	18-06-2021 10:00	18-06-2021 11:00
598	4	3	19-06-2021 11:00	19-06-2021 12:00
769	1	5	20-06-2021 11:30	20-06-2021 12:00
862	5	2	20-06-2021 10:00	20-06-2021 11:00
622	2	5	20-06-2021 12:00	20-06-2021 12:30

- **First Normal Form (1NF):**

  - No repeating groups could occur for the given table because every row of the Payment table is relatively different in nature.
  - No multi-valued columns exist because every column has a single set of information to be stored.
  - A primary key "**Appt_id**" has been defined.

Therefore, the given table is already in First Normal Form. So, now we could move further to check the table for 2NF.

- **Second Normal Form (2NF):**

  - As we have seen above that given table is already in 1NF, therefore it matches the first condition for 2NF.
  - Also, no partial dependencies occur in the Appointment table because every non-key element is fully dependent on the entire primary key.

- **Third Normal Form (3NF):**

  - As we have seen above that given table is already in 2NF, therefore it satisfies the first condition for 3NF.
  - A non-key column cannot determine the value of another non-key attribute.

9) Payment: This table contain the sample data as follow and we will check whether it matches with the different criteria of three normal form of normalization.

Bill_id	*Pat_id*	*Acc_id*	Total_amount	Discount	Pay_date
111	2	3	10000	0.10	20-07-2021
222	1	3	30000	0.05	21-07-2021
333	3	2	8000	0.08	28-07-2021
444	5	1	30000	NILL	05-07-2021
555	4	4	20000	NILL	18-07-2021

- **First Normal Form (1NF):**

  - No repeating groups could occur for the given table because every row of the Payment table is relatively different in nature.
  - No multi-valued columns exist because every column has a single set of information to be stored.
  - A primary key “**Bill_id**” has been defined.

Therefore, the given table is already in First Normal Form. So, now we could move further to check the table for 2NF.

- **Second Normal Form (2NF):**

  - As we have seen above that given table is already in 1NF, therefore it matches the first condition for 2NF.
  - Also, no partial dependencies occur in the Payment table because every non-key element is fully dependent on the entire primary key.

- **Third Normal Form (3NF):**

  - As we have seen above that given table is already in 2NF, therefore it satisfies the first condition for 3NF.
  - A non-key column cannot determine the value of another non-key attribute.

10) Room: This table contain the sample data as follow and we will check whether it matches with the different criteria of three normal form of normalization.

Room_no	Room_type	Floor_no	Total_beds	Occupied
234	Vip	2	1	1
342	Normal	3	20	5
242	Vip	2	1	0
312	Normal	3	10	8
372	Normal	3	15	15

- **First Normal Form (1NF):**

  - No repeating groups could occur for the given table because every row of the Payment table is relatively different in nature.
  - No multi-valued columns exist because every column has a single set of information to be stored.
  - A primary key "**Room_no**" has been defined.

Therefore, the given table is already in First Normal Form. So, now we could move further to check the table for 2NF.

- **Second Normal Form (2NF):**

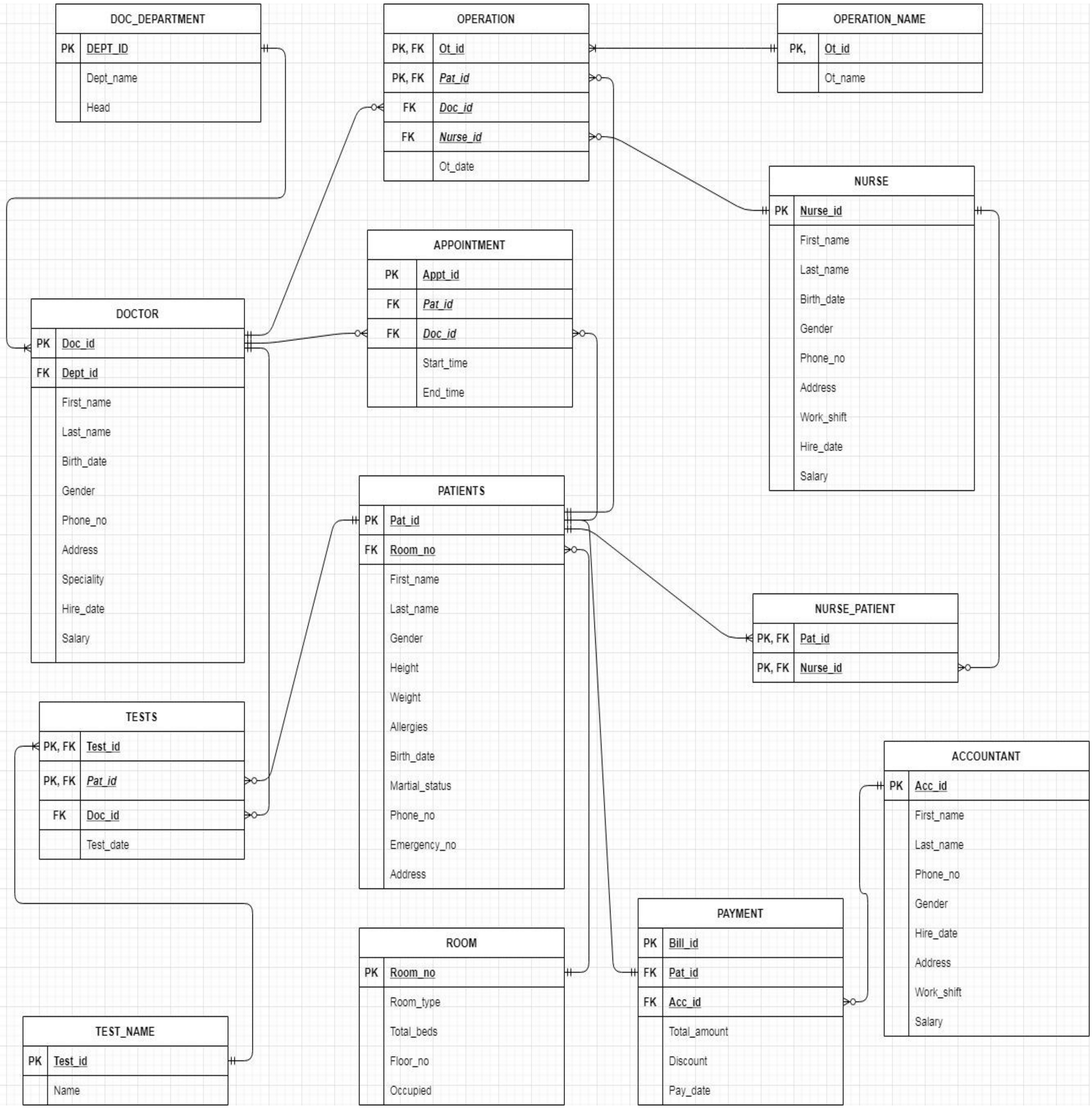
  - As we have seen above that given table is already in 1NF, therefore it matches the first condition for 2NF.
  - Also, no partial dependencies occur in the Room table because every non-key element is fully dependent on the entire primary key.

- **Third Normal Form (3NF):**

  - As we have seen above that given table is already in 2NF, therefore it satisfies the first condition for 3NF.
  - A non-key column cannot determine the value of another non-key attribute.

Entity Relationship Diagram(ERD):

After Normalization



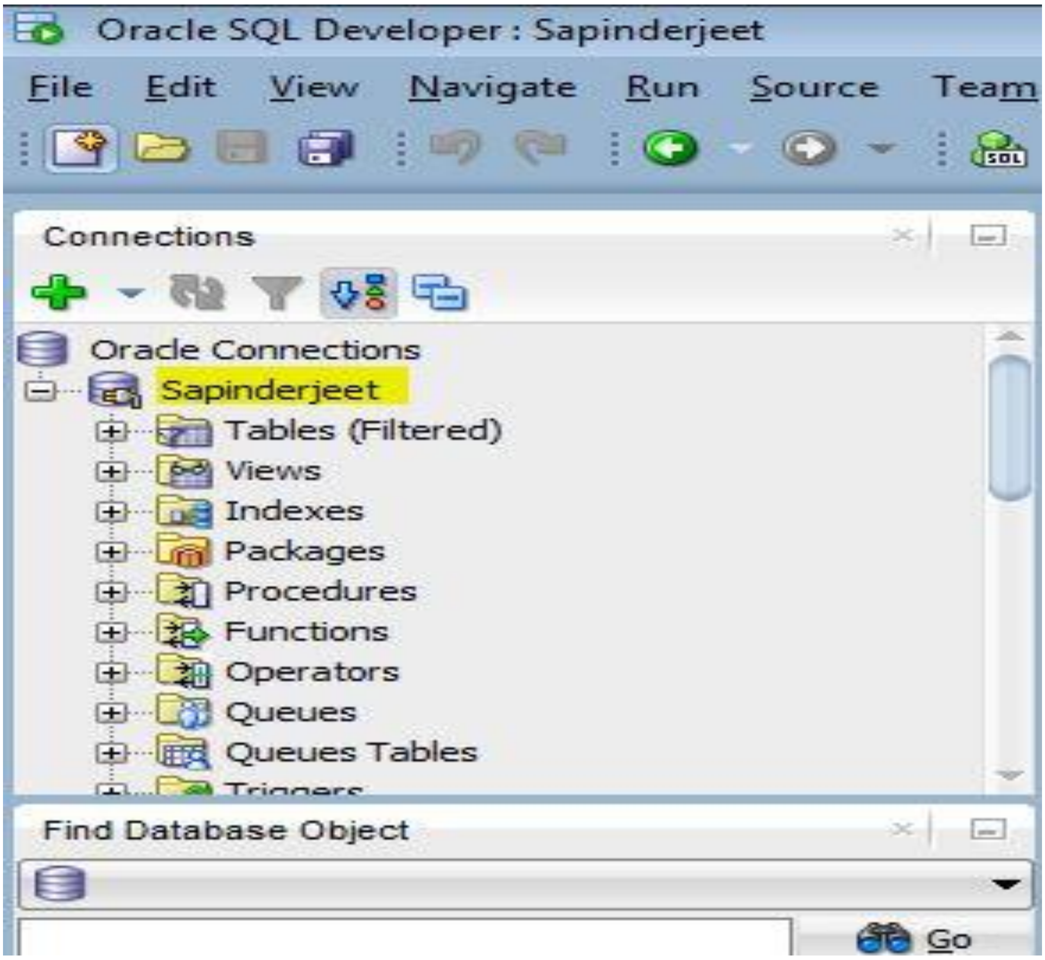
Before Normalization: 9 Tables

After Normalization: 13 Tables

New Tables Added: Nurse_patient, Appointment, Test_name, Operation_name

DATABASE QUERIES:

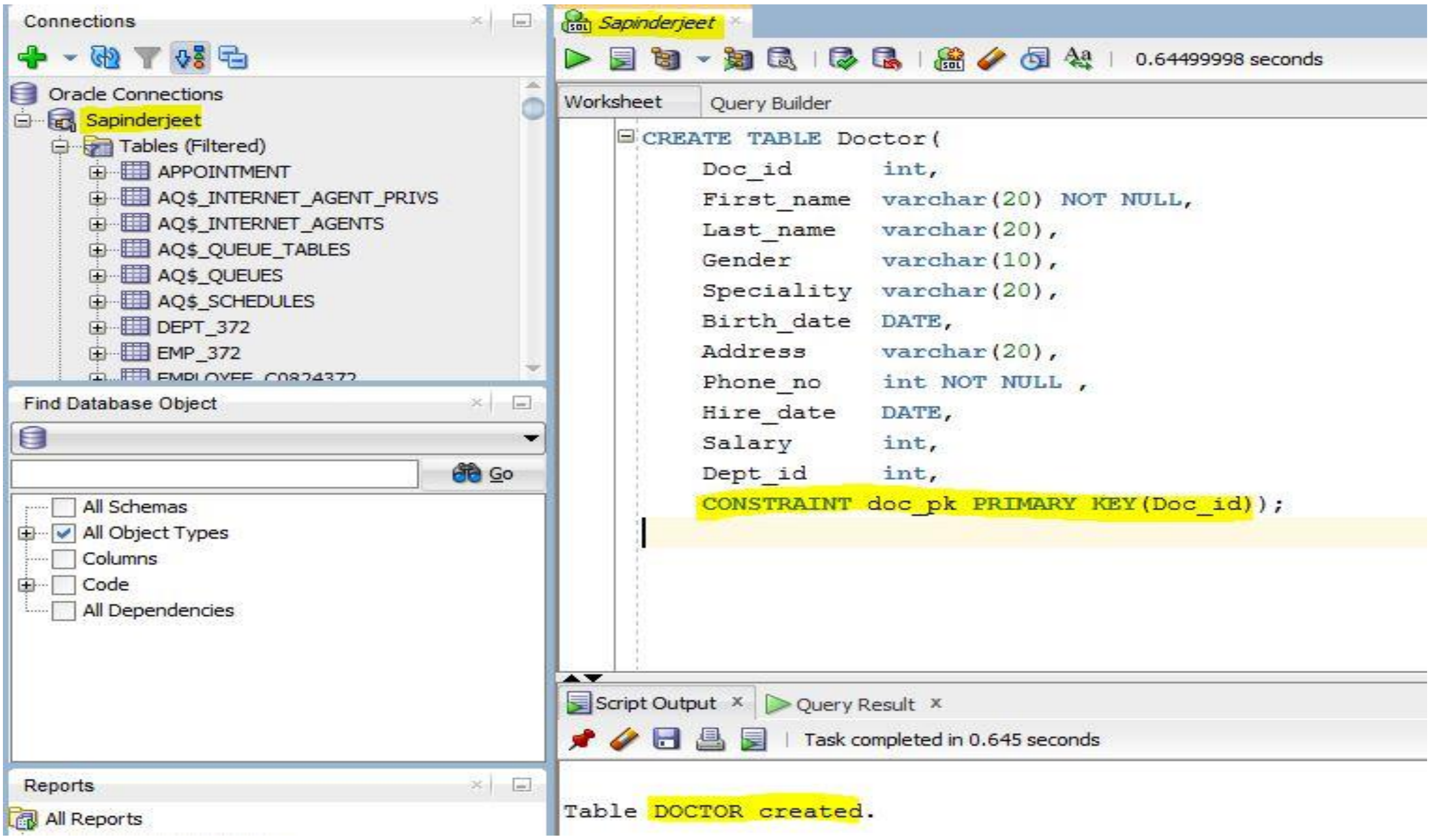
- For this project, the name of my database connection is **Sapinderjeet** as shown below:



- This project is implemented using oracle 18c and sql developer.
- All the table data will be implemented in the sql developer further different Queries will be implemented to retrieve data.

CREATING TABLES:

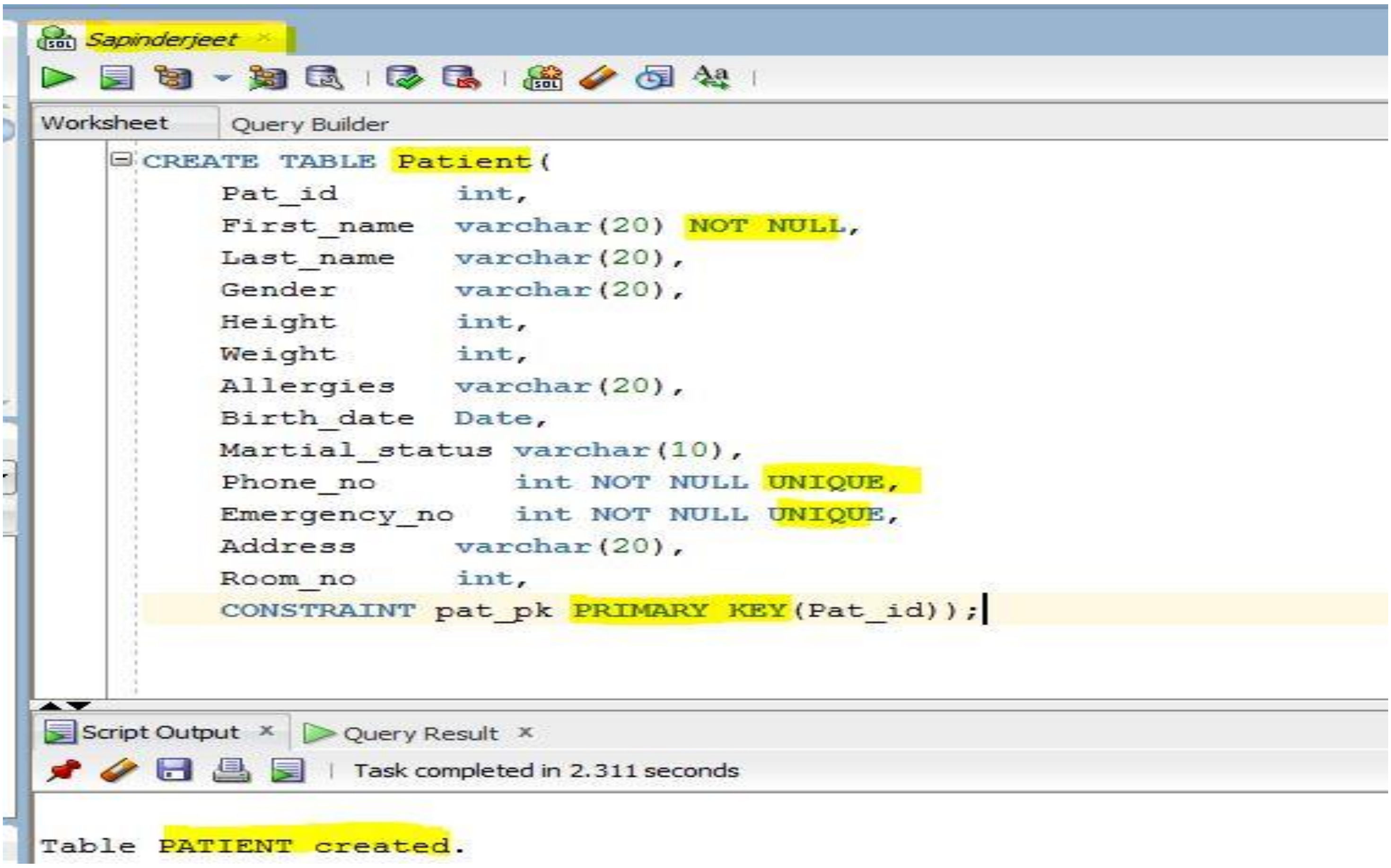
A. Doctor table:



```
CREATE TABLE Doctor(
    Doc_id      int,
    First_name  varchar(20) NOT NULL,
    Last_name   varchar(20),
    Gender      varchar(10),
    Speciality  varchar(20),
    Birth_date  DATE,
    Address     varchar(20),
    Phone_no    int NOT NULL ,
    Hire_date   DATE,
    Salary      int,
    Dept_id     int,
    CONSTRAINT doc_pk PRIMARY KEY(Doc_id);
```

```
Table DOCTOR created.
```

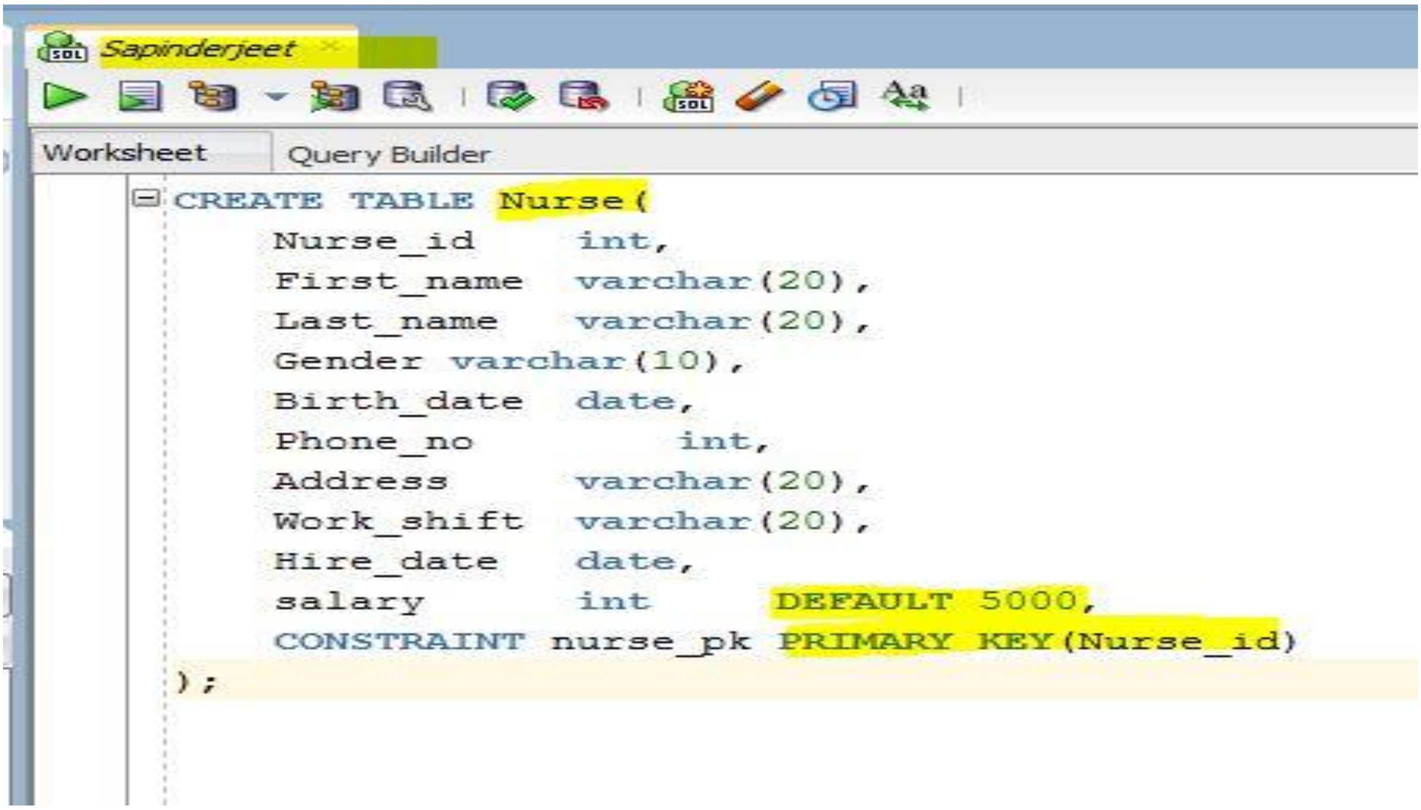
B. Patient Table



```
CREATE TABLE Patient(
    Pat_id          int,
    First_name      varchar(20) NOT NULL,
    Last_name       varchar(20),
    Gender          varchar(20),
    Height          int,
    Weight          int,
    Allergies       varchar(20),
    Birth_date      Date,
    Martial_status  varchar(10),
    Phone_no            int NOT NULL UNIQUE,
    Emergency_no        int NOT NULL UNIQUE,
    Address         varchar(20),
    Room_no         int,
    CONSTRAINT pat_pk PRIMARY KEY(Pat_id));
```

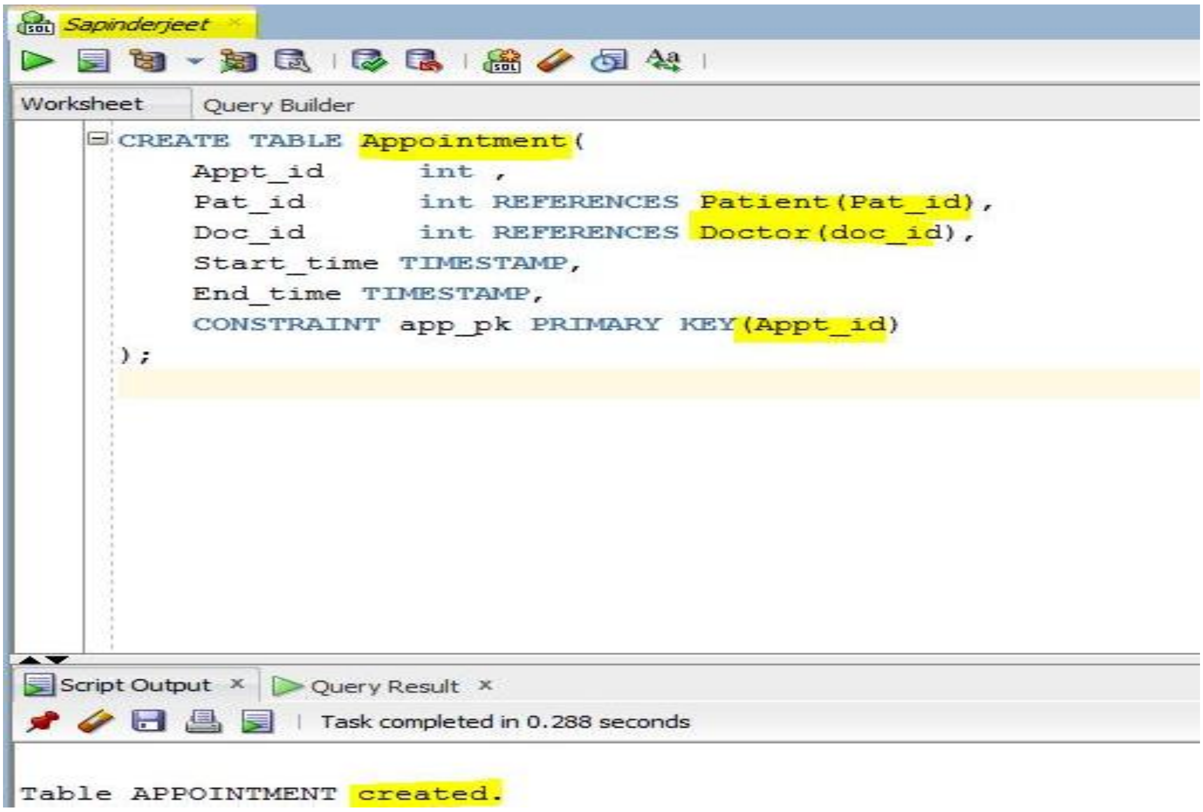
```
Table PATIENT created.
```

C. Nurse Table:



```
CREATE TABLE Nurse(
     Nurse_id     int,
     First_name   varchar(20),
     Last_name    varchar(20),
     Gender varchar(10),
     Birth_date   date,
     Phone_no          int,
     Address      varchar(20),
     Work_shift   varchar(20),
     Hire_date    date,
     salary       int       DEFAULT 5000,
     CONSTRAINT nurse_pk PRIMARY KEY(Nurse_id)
);
```

D. Appointment Table



```
CREATE TABLE Appointment(
     Appt_id      int ,
     Pat_id       int REFERENCES Patient(Pat_id),
     Doc_id       int REFERENCES Doctor(doc_id),
     Start_time TIMESTAMP,
     End_time TIMESTAMP,
     CONSTRAINT app_pk PRIMARY KEY(Appt_id)
);
```

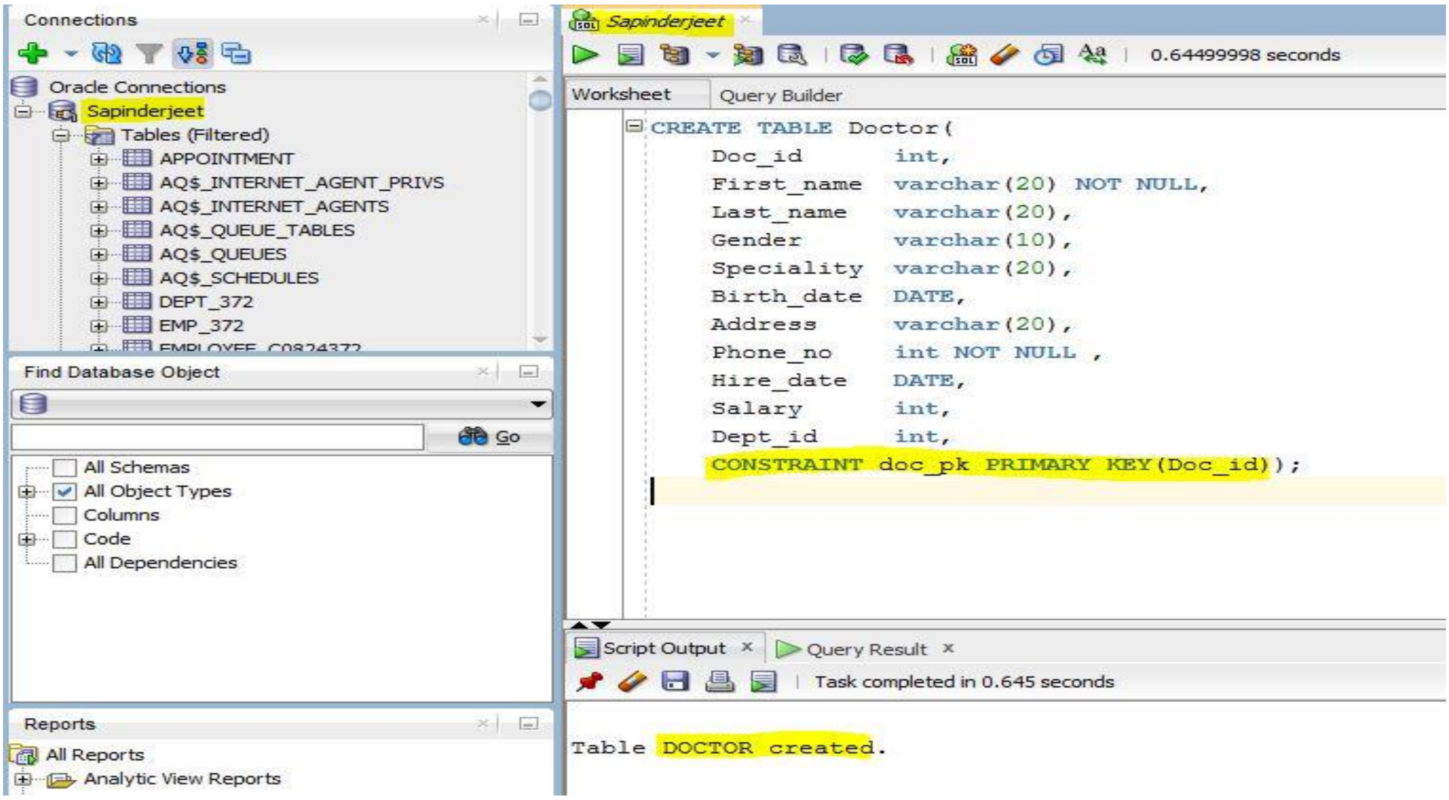
Task completed in 0.288 seconds

```
Table APPOINTMENT created.
```

Similarly, all the tables were created.

1. Constraints

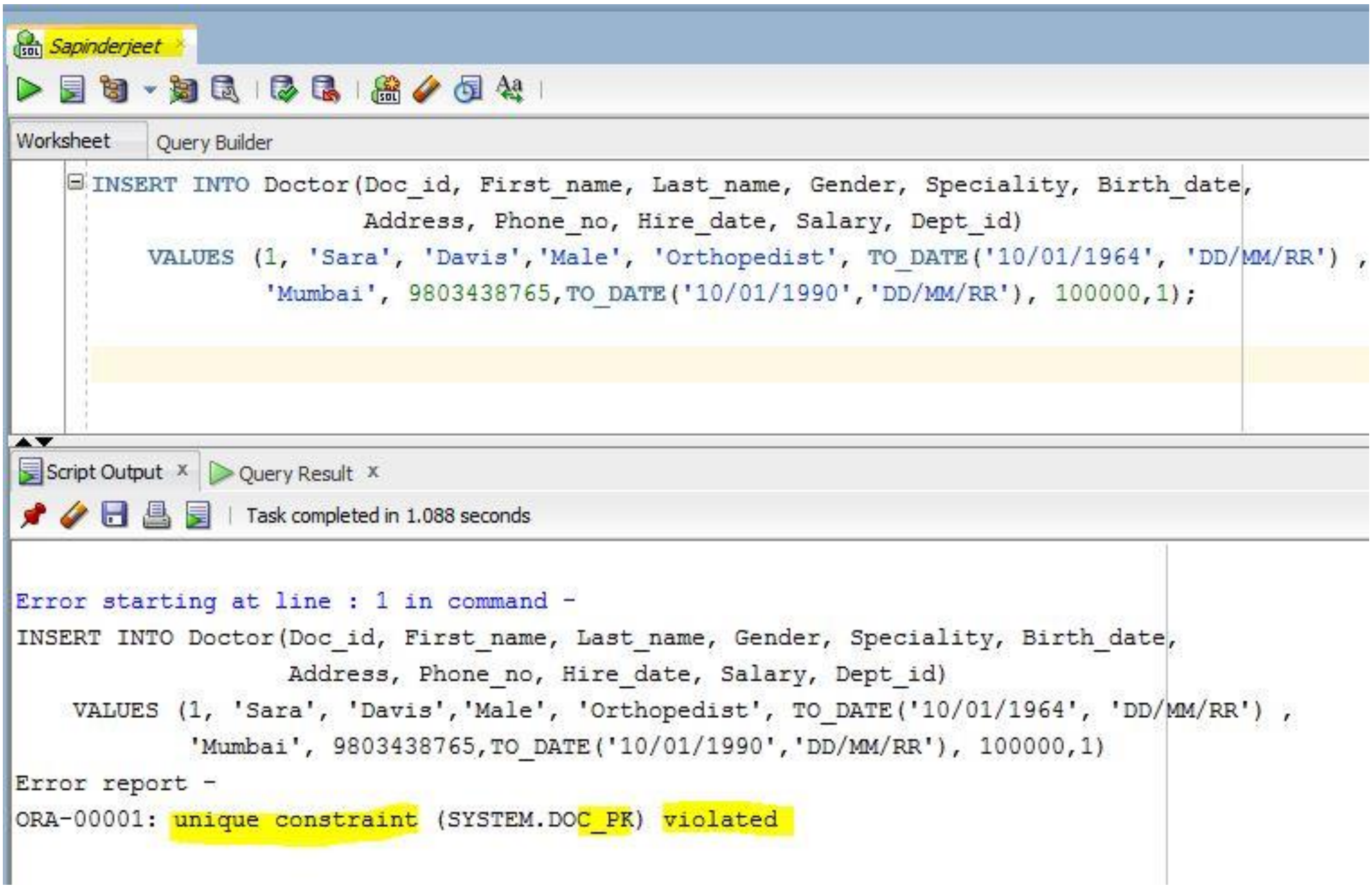
1.1) **Primary Key constraint**: Table Doctor



```
CREATE TABLE Doctor(
        Doc_id          int,
        First_name      varchar(20) NOT NULL,
        Last_name       varchar(20),
        Gender          varchar(10),
        Speciality      varchar(20),
        Birth_date      DATE,
        Address         varchar(20),
        Phone_no        int NOT NULL ,
        Hire_date       DATE,
        Salary          int,
        Dept_id         int,
        CONSTRAINT doc_pk PRIMARY KEY(Doc_id));
```

```
Table DOCTOR created.
```

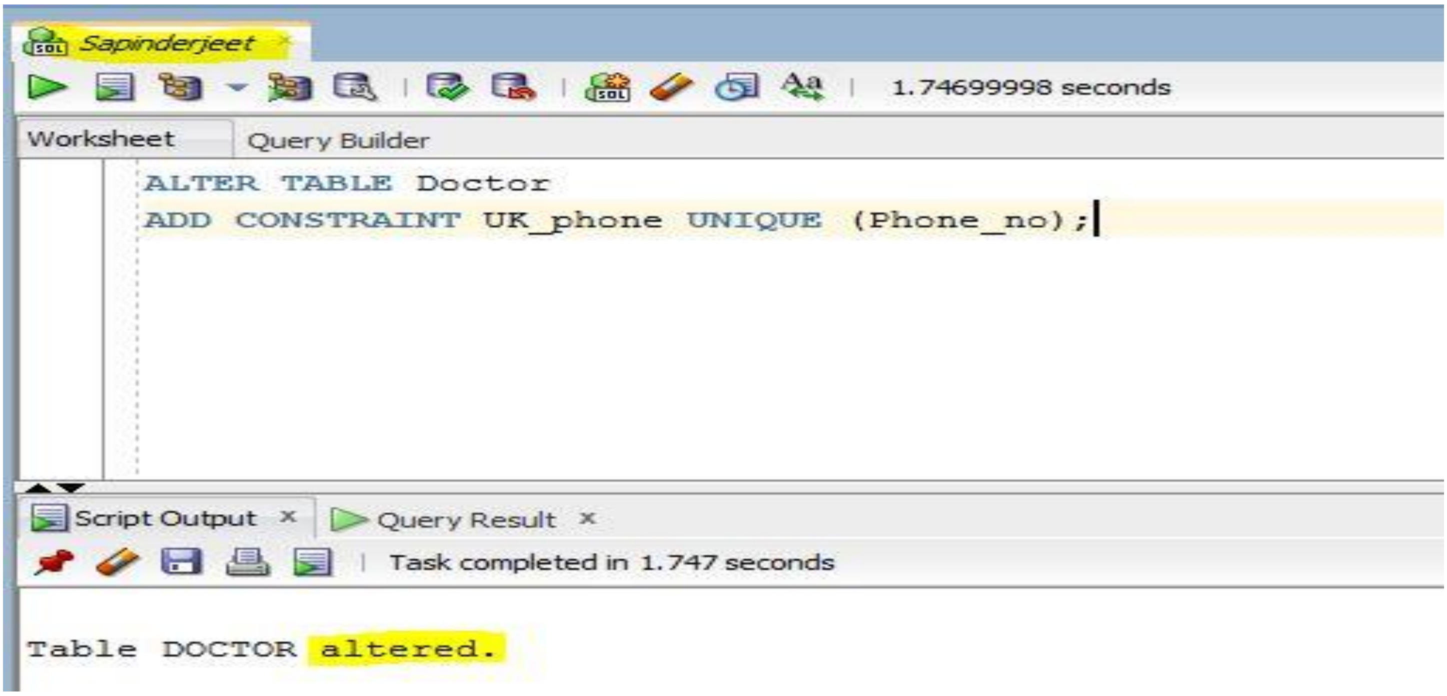
Checking: Here, sql gives error as primary key constraint is violated.



```
INSERT INTO Doctor(Doc_id, First_name, Last_name, Gender, Speciality, Birth_date,
                    Address, Phone_no, Hire_date, Salary, Dept_id)
    VALUES (1, 'Sara', 'Davis','Male', 'Orthopedist', TO_DATE('10/01/1964', 'DD/MM/RR') ,
            'Mumbai', 9803438765,TO_DATE('10/01/1990','DD/MM/RR'), 100000,1);
```

```
Error starting at line : 1 in command -
INSERT INTO Doctor(Doc_id, First_name, Last_name, Gender, Speciality, Birth_date,
                    Address, Phone_no, Hire_date, Salary, Dept_id)
    VALUES (1, 'Sara', 'Davis','Male', 'Orthopedist', TO_DATE('10/01/1964', 'DD/MM/RR') ,
            'Mumbai', 9803438765,TO_DATE('10/01/1990','DD/MM/RR'), 100000,1)
Error report -
ORA-00001: unique constraint (SYSTEM.DOC_PK) violated
```

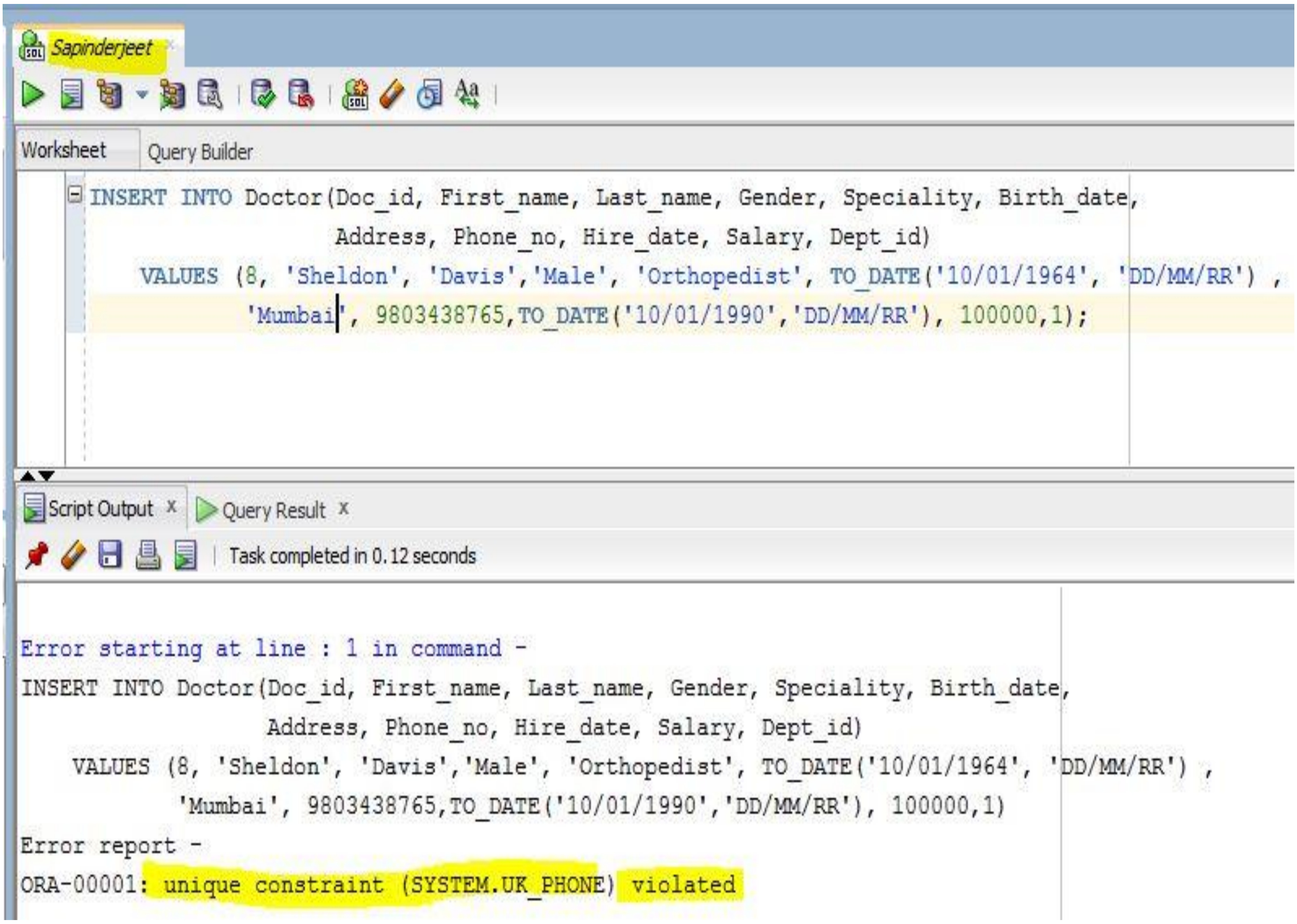
1.2) Unique Constraint: Phone_no should be unique.



```
ALTER TABLE Doctor
ADD CONSTRAINT UK_phone UNIQUE (Phone_no);
```

```
Table DOCTOR altered.
```

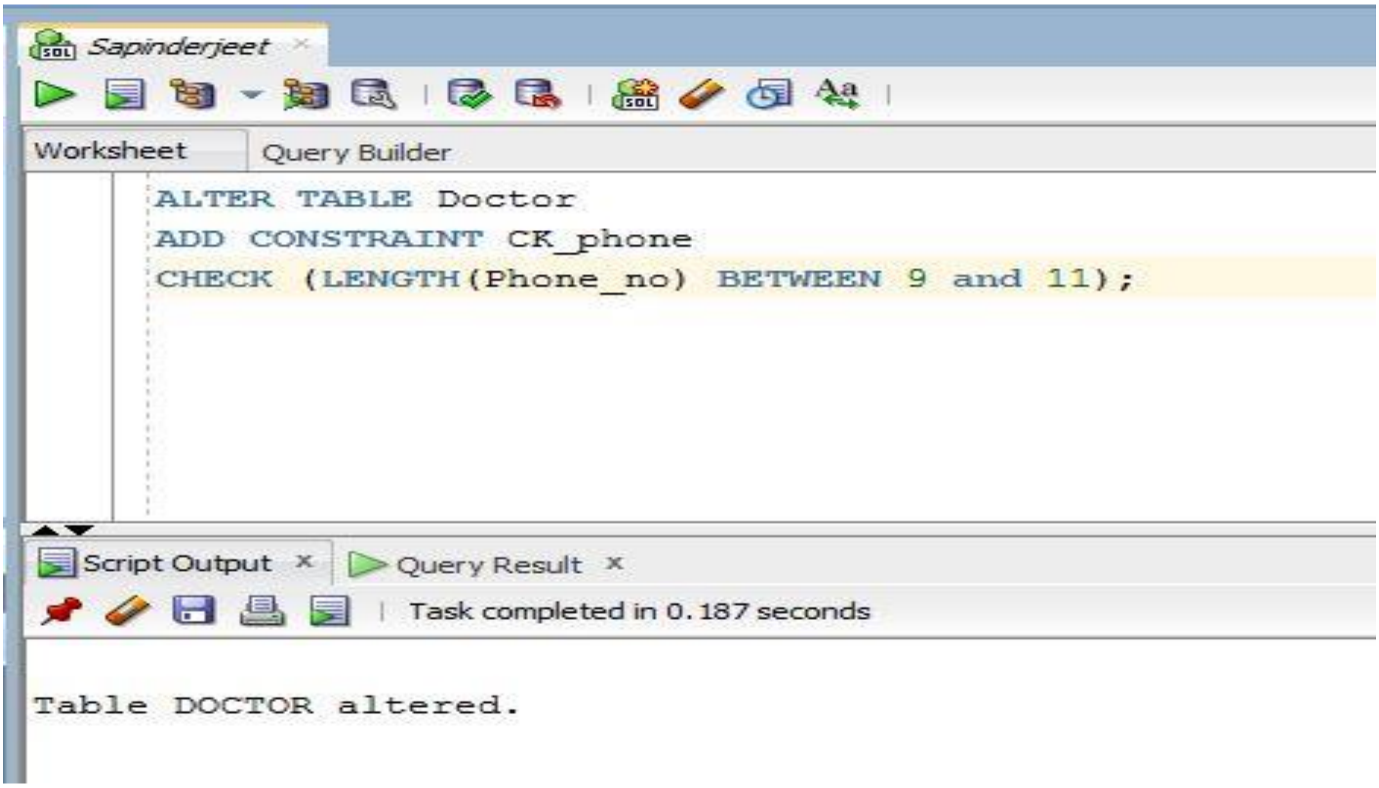
CHECKING: Here, the system will give error as phone number is not unique.



```
INSERT INTO Doctor(Doc_id, First_name, Last_name, Gender, Speciality, Birth_date,
                   Address, Phone_no, Hire_date, Salary, Dept_id)
    VALUES (8, 'Sheldon', 'Davis','Male', 'Orthopedist', TO_DATE('10/01/1964', 'DD/MM/RR') ,
            'Mumbai', 9803438765,TO_DATE('10/01/1990','DD/MM/RR'), 100000,1);
```

```
Error starting at line : 1 in command -
INSERT INTO Doctor(Doc_id, First_name, Last_name, Gender, Speciality, Birth_date,
                   Address, Phone_no, Hire_date, Salary, Dept_id)
    VALUES (8, 'Sheldon', 'Davis','Male', 'Orthopedist', TO_DATE('10/01/1964', 'DD/MM/RR') ,
            'Mumbai', 9803438765,TO_DATE('10/01/1990','DD/MM/RR'), 100000,1)
Error report -
ORA-00001: unique constraint (SYSTEM.UK_PHONE) violated
```

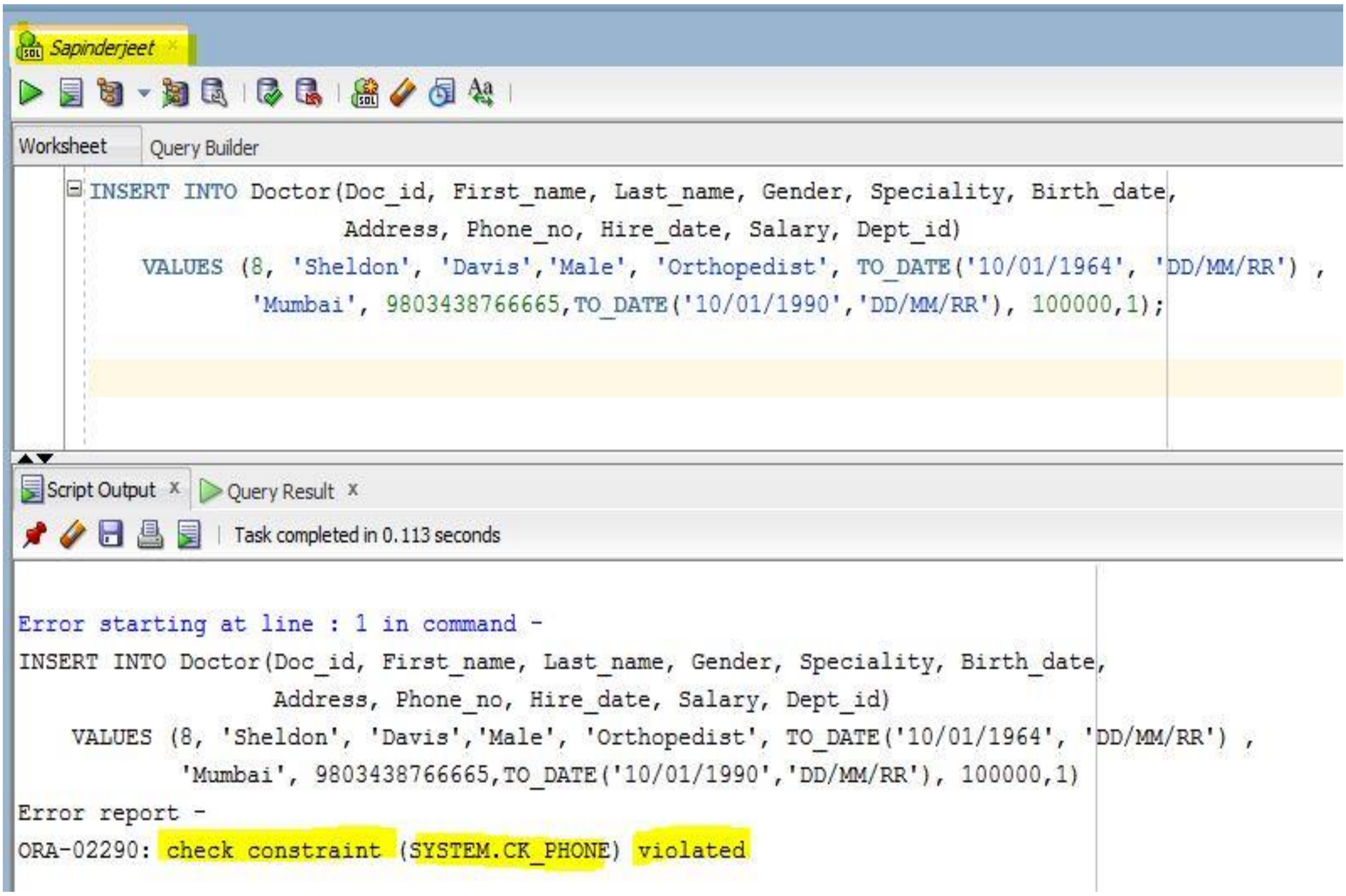
1.3) Check Constraint: Length of phone_no should be between 9 and 11.



```
ALTER TABLE Doctor
ADD CONSTRAINT CK_phone
CHECK (LENGTH(Phone_no) BETWEEN 9 and 11);
```

```
Table DOCTOR altered.
```

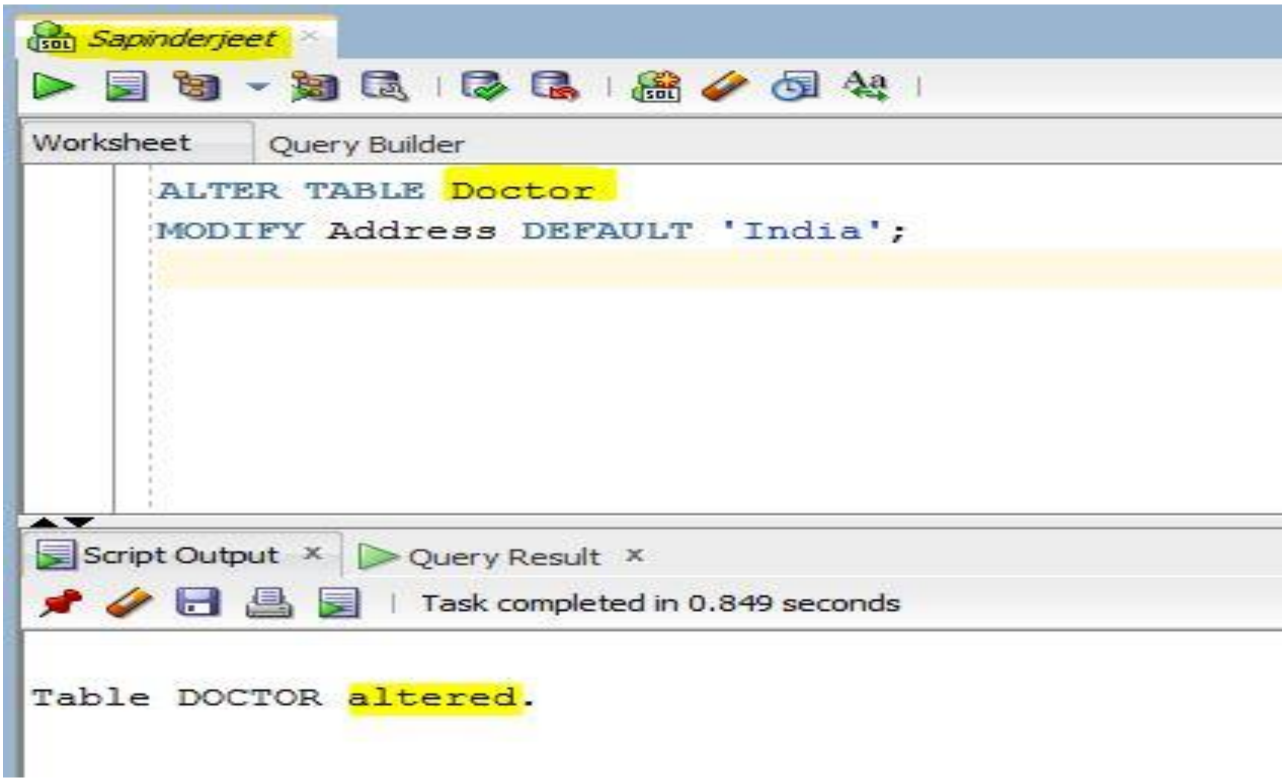
CHECKING: Here, the system will give error as the length of mobile no. is greater than 11.



```
INSERT INTO Doctor(Doc_id, First_name, Last_name, Gender, Speciality, Birth_date,
                   Address, Phone_no, Hire_date, Salary, Dept_id)
    VALUES (8, 'Sheldon', 'Davis','Male', 'Orthopedist', TO_DATE('10/01/1964', 'DD/MM/RR') ,
            'Mumbai', 9803438766665,TO_DATE('10/01/1990','DD/MM/RR'), 100000,1);
```

```
Error starting at line : 1 in command -
INSERT INTO Doctor(Doc_id, First_name, Last_name, Gender, Speciality, Birth_date,
                   Address, Phone_no, Hire_date, Salary, Dept_id)
    VALUES (8, 'Sheldon', 'Davis','Male', 'Orthopedist', TO_DATE('10/01/1964', 'DD/MM/RR') ,
            'Mumbai', 9803438766665,TO_DATE('10/01/1990','DD/MM/RR'), 100000,1)
Error report -
ORA-02290: check constraint (SYSTEM.CK_PHONE) violated
```

1.4) Default Constraint: By default, the address will be ‘India’

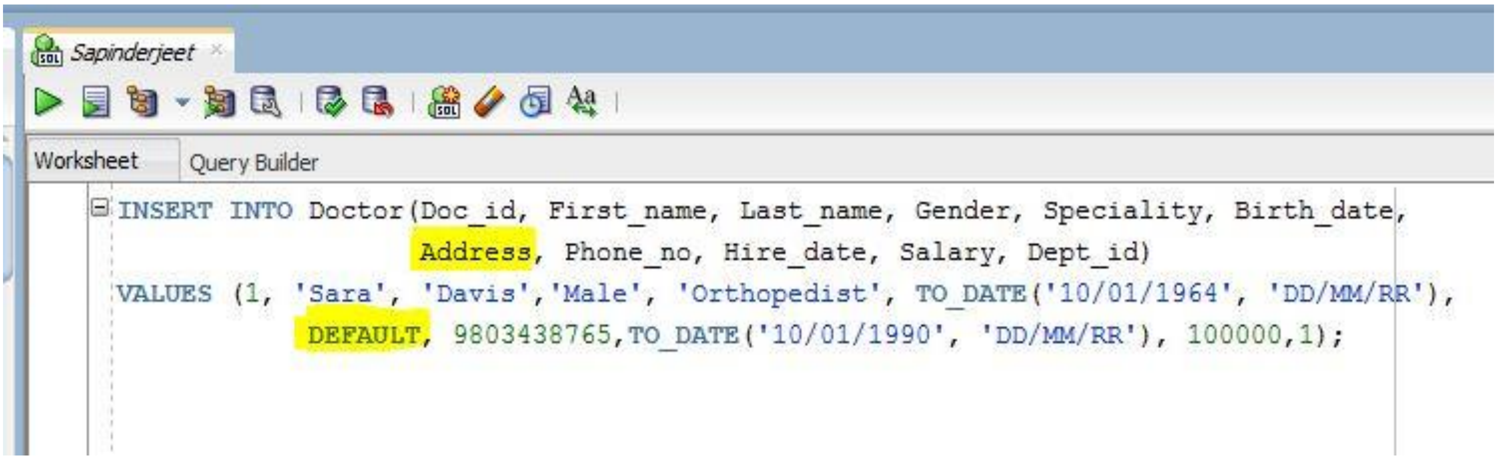


```
ALTER TABLE Doctor
MODIFY Address DEFAULT 'India';
```

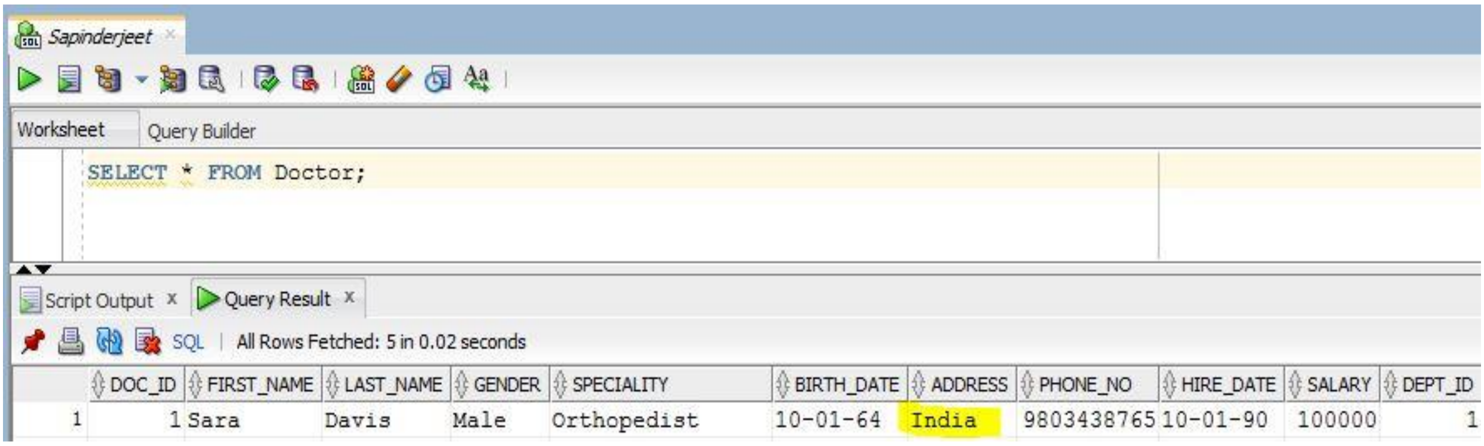
Task completed in 0.849 seconds

```
Table DOCTOR altered.
```

CHECKING:



```
INSERT INTO Doctor(Doc_id, First_name, Last_name, Gender, Speciality, Birth_date,
                 Address, Phone_no, Hire_date, Salary, Dept_id)
VALUES (1, 'Sara', 'Davis','Male', 'Orthopedist', TO_DATE('10/01/1964', 'DD/MM/RR'),
        DEFAULT, 9803438765,TO_DATE('10/01/1990', 'DD/MM/RR'), 100000,1);
```

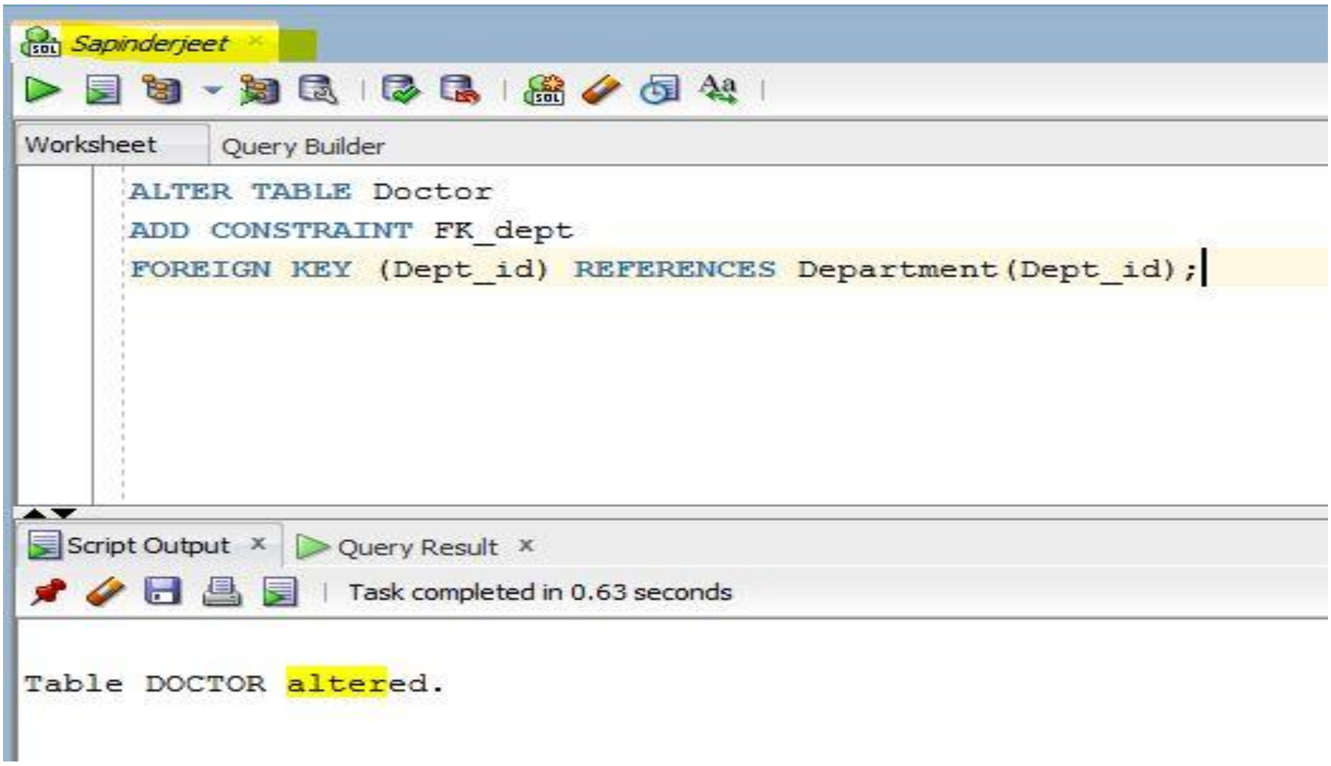


```
SELECT * FROM Doctor;
```

All Rows Fetched: 5 in 0.02 seconds

	DOC_ID	FIRST_NAME	LAST_NAME	GENDER	SPECIALITY	BIRTH_DATE	ADDRESS	PHONE_NO	HIRE_DATE	SALARY	DEPT_ID
1	1	Sara	Davis	Male	Orthopedist	10-01-64	India	9803438765	10-01-90	100000	1

1.5) Foreign Key Constraint:

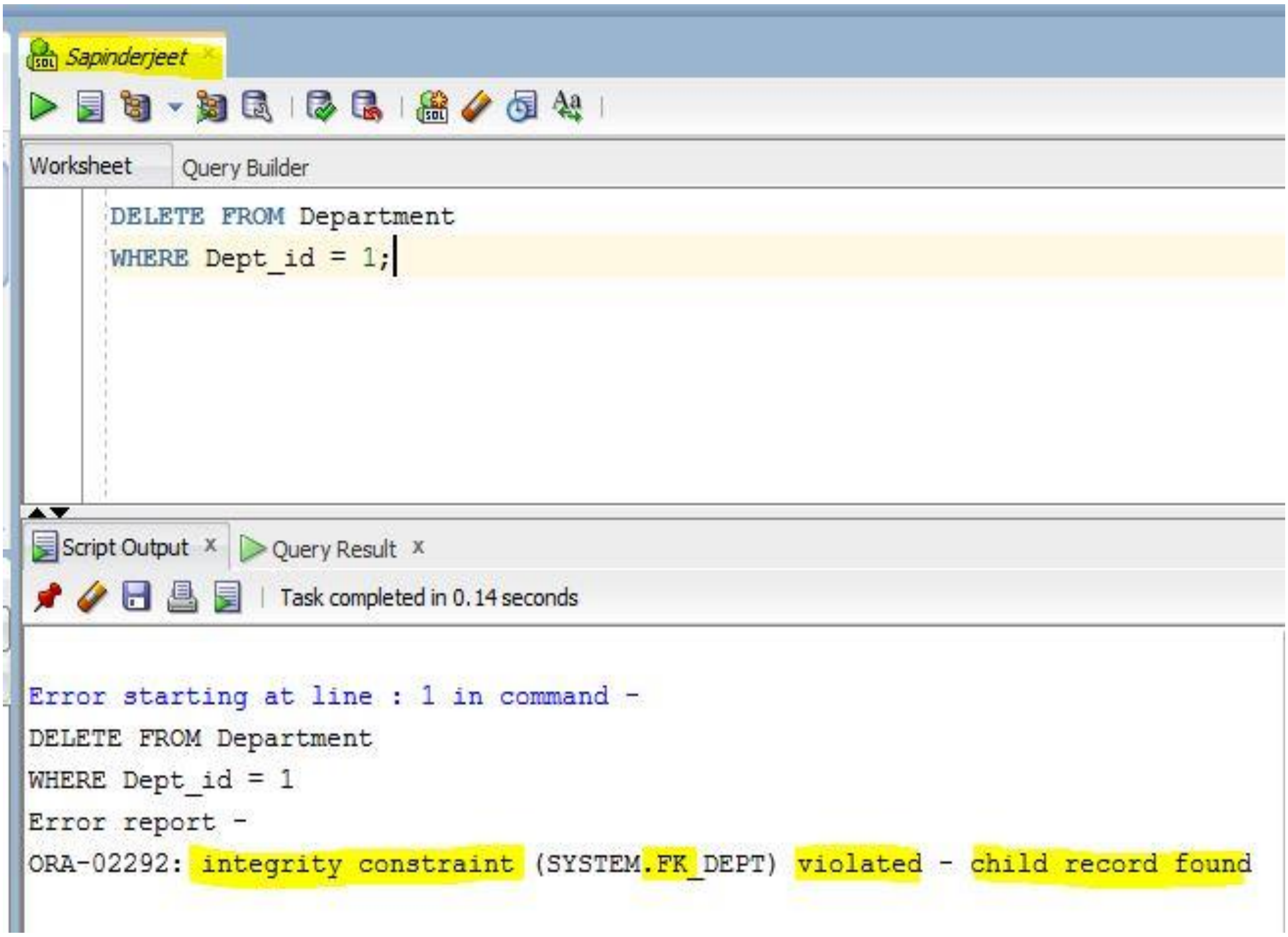


```
ALTER TABLE Doctor
ADD CONSTRAINT FK_dept
FOREIGN KEY (Dept_id) REFERENCES Department(Dept_id);
```

Task completed in 0.63 seconds

```
Table DOCTOR altered.
```

CHECKING: Here system will give error as foreign key constraint is violated.



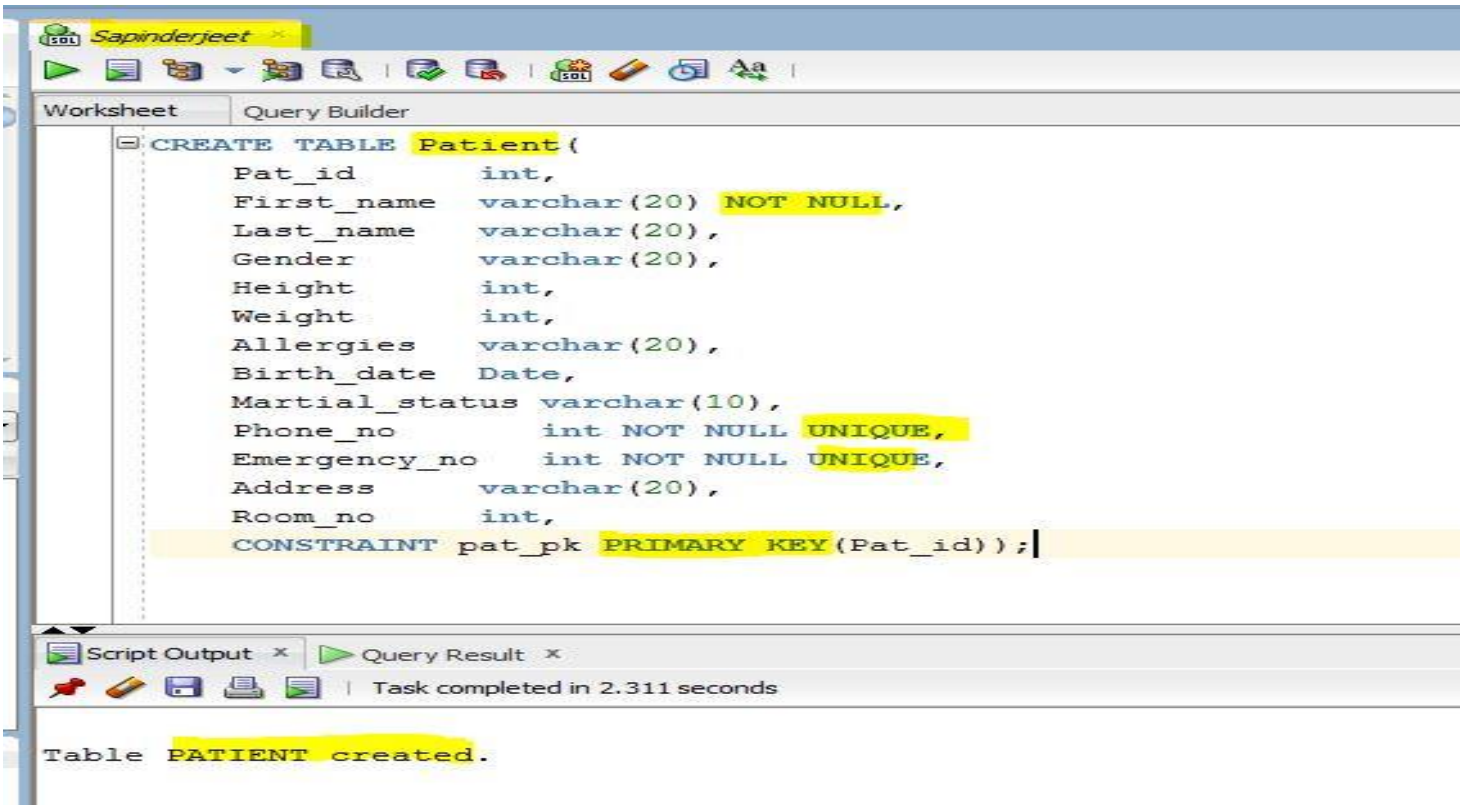
```
DELETE FROM Department
WHERE Dept_id = 1;
```

Task completed in 0.14 seconds

```
Error starting at line : 1 in command -
DELETE FROM Department
WHERE Dept_id = 1
Error report -
ORA-02292: integrity constraint (SYSTEM.FK_DEPT) violated - child record found
```

1.6 Constraints on other main tables:

a. Unique, Not Null, Primary Key constraint on **Patient Table.**

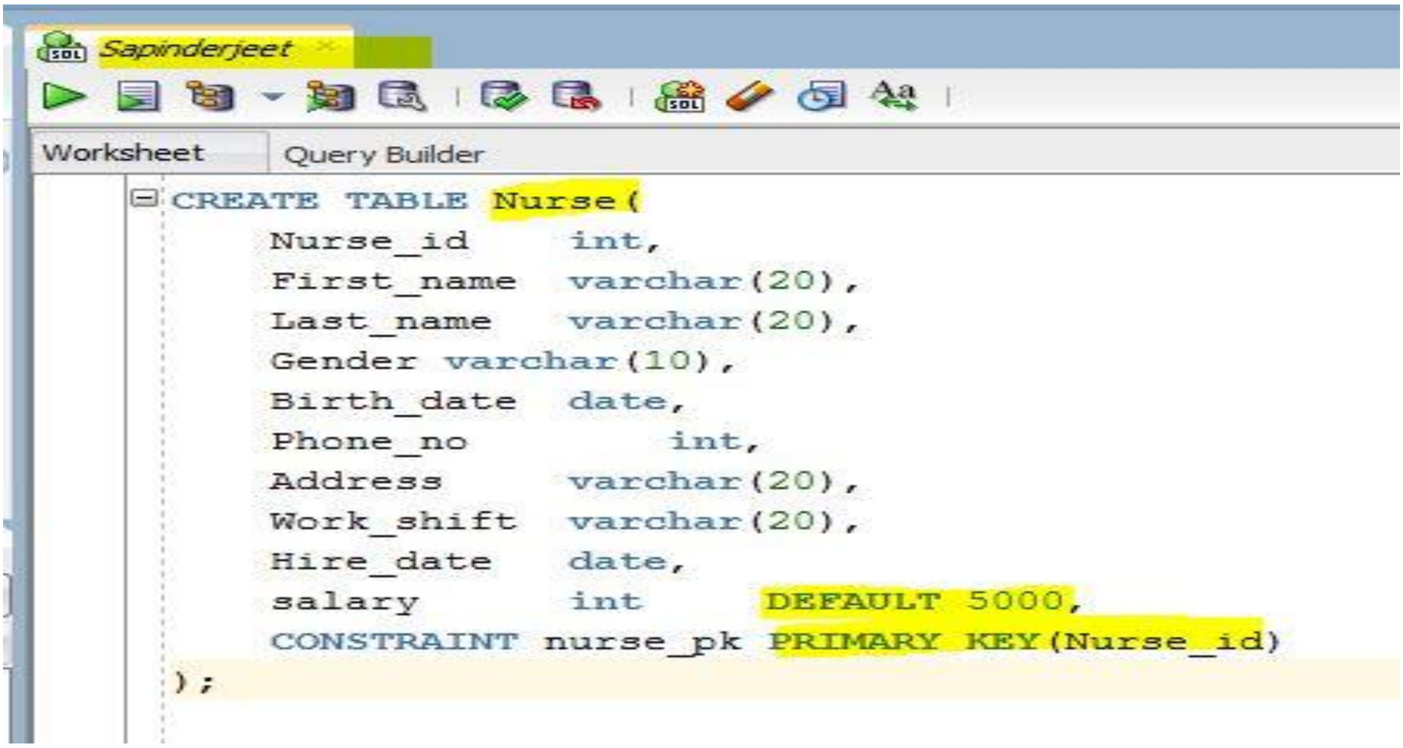


```
CREATE TABLE Patient(
        Pat_id          int,
        First_name      varchar(20) NOT NULL,
        Last_name       varchar(20),
        Gender          varchar(20),
        Height          int,
        Weight          int,
        Allergies       varchar(20),
        Birth_date      Date,
        Martial_status  varchar(10),
        Phone_no           int NOT NULL UNIQUE,
        Emergency_no       int NOT NULL UNIQUE,
        Address         varchar(20),
        Room_no         int,
        CONSTRAINT pat_pk PRIMARY KEY(Pat_id));
```

Task completed in 2.311 seconds

```
Table PATIENT created.
```

b. Default, Primary Key constraint on Nurse Table



```
CREATE TABLE Nurse(
        Nurse_id      int,
        First_name    varchar(20),
        Last_name     varchar(20),
        Gender  varchar(10),
        Birth_date    date,
        Phone_no              int,
        Address       varchar(20),
        Work_shift    varchar(20),
        Hire_date     date,
        salary        int       DEFAULT 5000,
        CONSTRAINT nurse_pk PRIMARY KEY(Nurse_id)
);
```

Similarly, all the other tables are created with constraints wherever necessary.

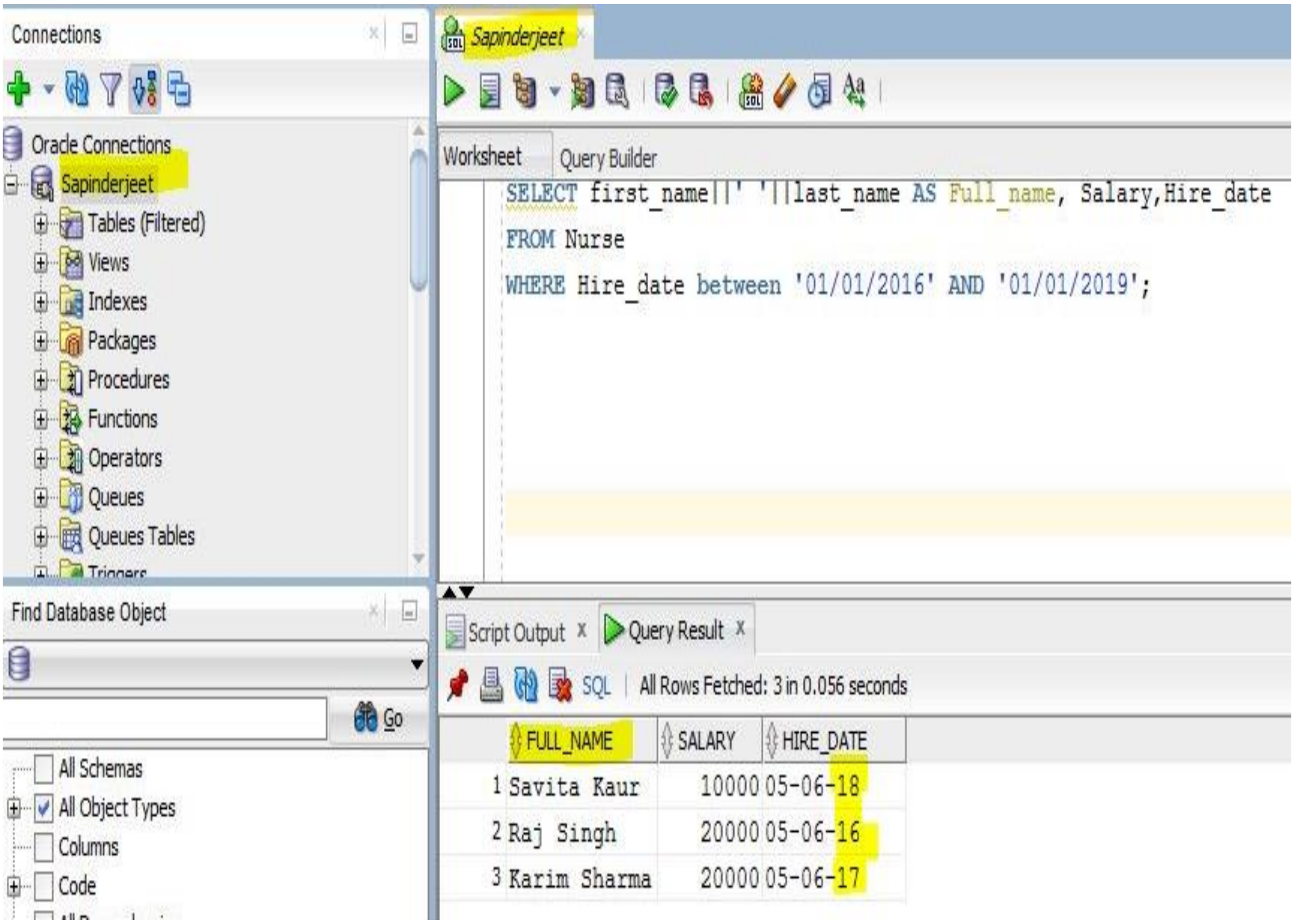
2. Single table queries

2.1) List all the nurses hired between 1 Jan 2016 and 1 Jan 2019.

SCRIPT:

```
SELECT first_name ||' '|| last_name AS Full_name, Salary, Hire_date
FROM Nurse
WHERE Hire_date between '01/01/2016' AND '01/01/2019';
```

Output:



```
SELECT first_name||' '||last_name AS Full_name, Salary,Hire_date
FROM Nurse
WHERE Hire_date between '01/01/2016' AND '01/01/2019';
```

All Rows Fetched: 3 in 0.056 seconds

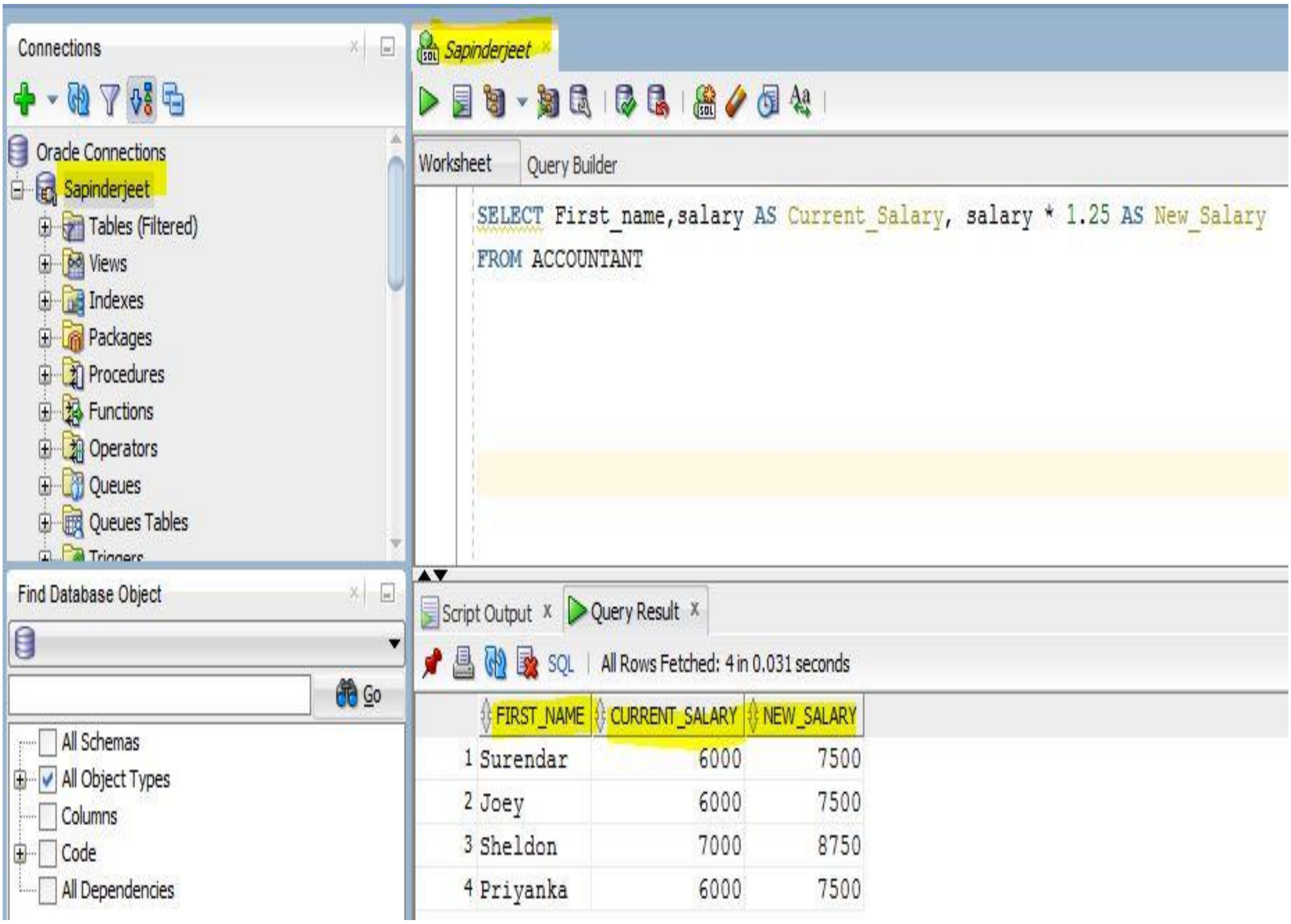
	FULL_NAME	SALARY	HIRE_DATE
1	Savita Kaur	10000	05-06-18
2	Raj Singh	20000	05-06-16
3	Karim Sharma	20000	05-06-17

2.2) Display salary of Accountants along with new salary (1.25) times the original salary.

SCRIPT:

```
SELECT first_name, Salary AS Current_Salary, salary*1.25 AS New_Salary
FROM Accountant
```

OUTPUT:



```
SELECT First_name,salary AS Current_Salary, salary * 1.25 AS New_Salary
FROM ACCOUNTANT
```

All Rows Fetched: 4 in 0.031 seconds

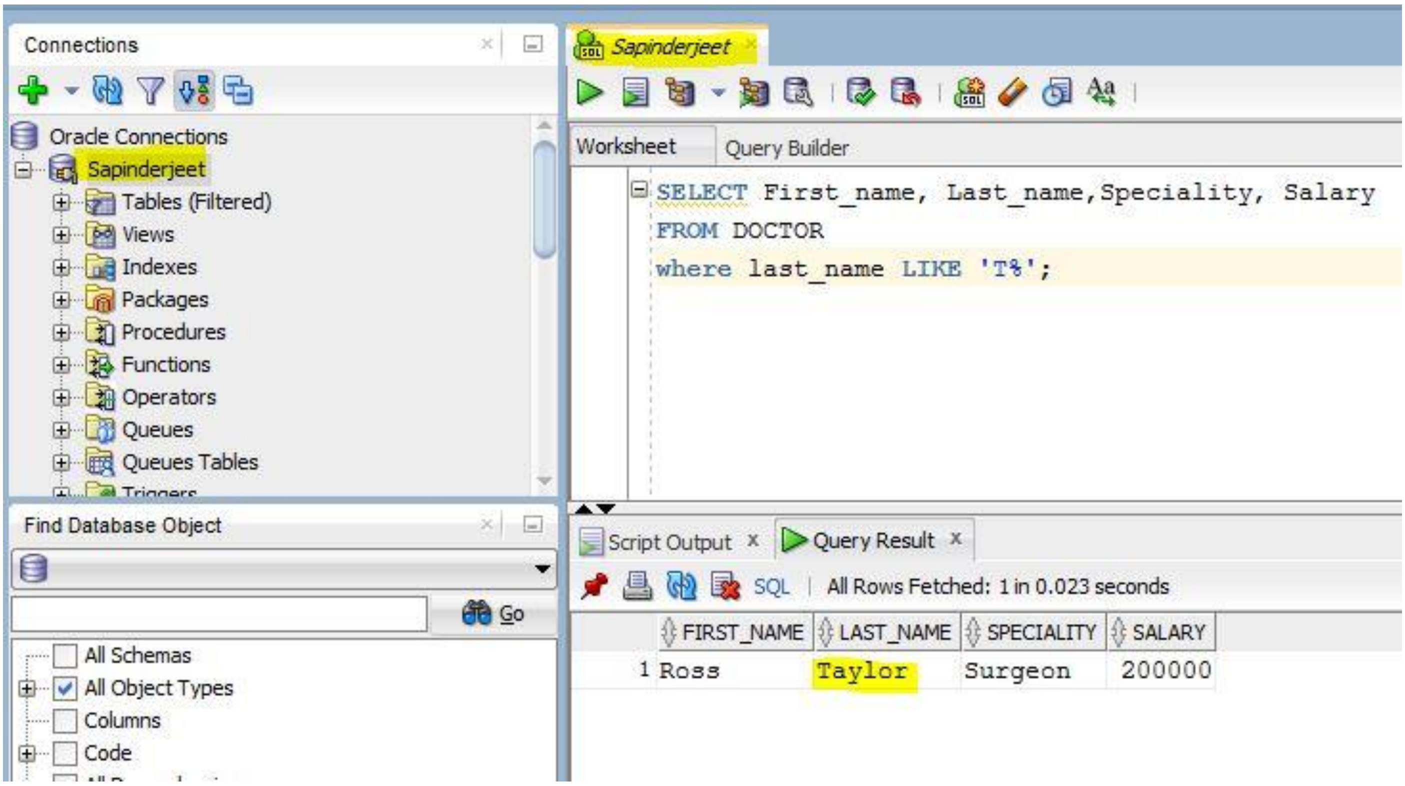
	FIRST_NAME	CURRENT_SALARY	NEW_SALARY
1	Surendar	6000	7500
2	Joey	6000	7500
3	Sheldon	7000	8750
4	Priyanka	6000	7500

2.3) Display the salary of the doctor whose last_name starts with ‘T’.

Script:

```
SELECT first_name, Last_name, Speciality, Salary
FROM Doctor
WHERE last_name LIKE ‘T%’
```

Output:



```
SELECT First_name, Last_name,Speciality, Salary
FROM DOCTOR
where last_name LIKE 'T%';
```

All Rows Fetched: 1 in 0.023 seconds

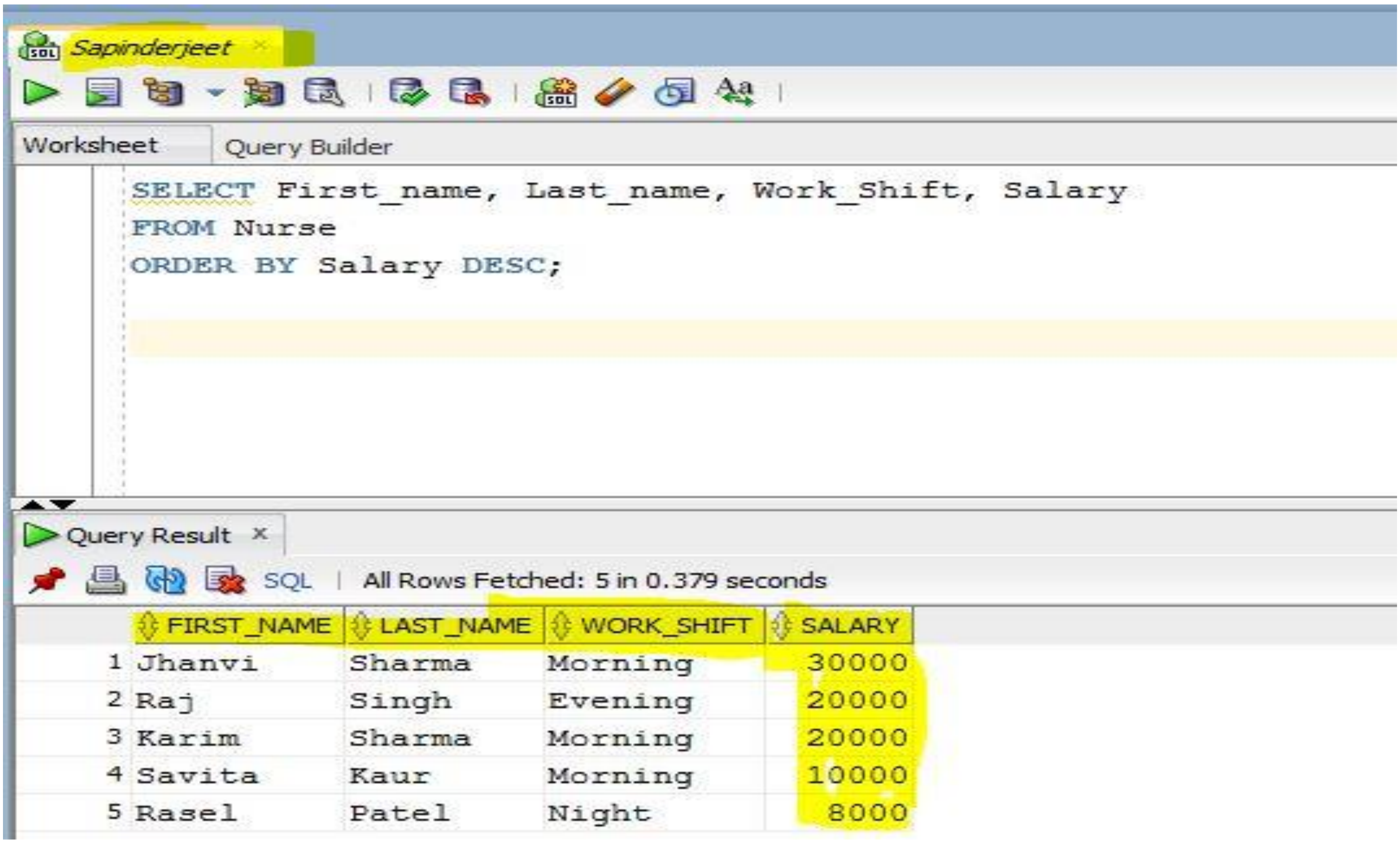
	FIRST_NAME	LAST_NAME	SPECIALITY	SALARY
1	Ross	Taylor	Surgeon	200000

2.4) Return the First_name, Last_name, Work_shift and salary from Nurses table. Order by salary descending.

SCRIPT:

```
SELECT First_name, Last_name, Work_shift, Salary
FROM Nurse
ORDER BY Salary DESC;
```

OUTPUT:



```
SELECT First_name, Last_name, Work_Shift, Salary
FROM Nurse
ORDER BY Salary DESC;
```

All Rows Fetched: 5 in 0.379 seconds

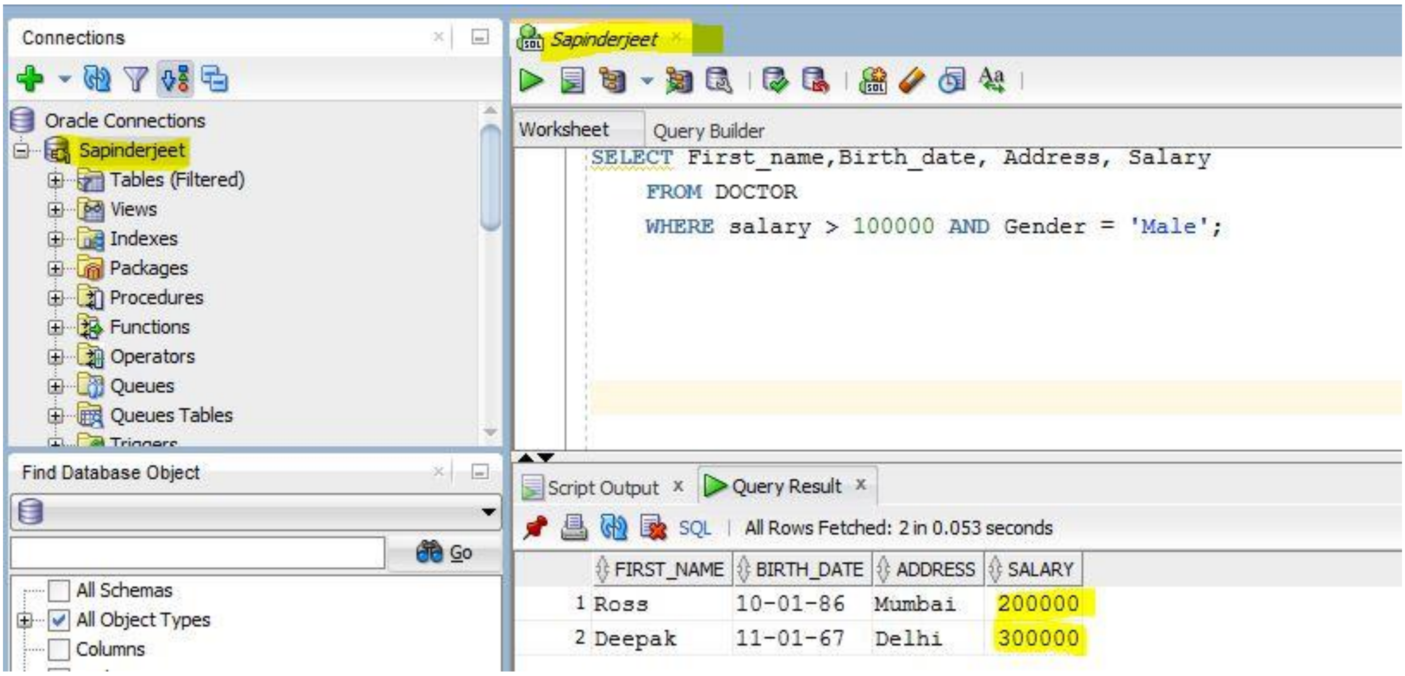
	FIRST_NAME	LAST_NAME	WORK_SHIFT	SALARY
1	Jhanvi	Sharma	Morning	30000
2	Raj	Singh	Evening	20000
3	Karim	Sharma	Morning	20000
4	Savita	Kaur	Morning	10000
5	Rasel	Patel	Night	8000

2.5) Find all the male doctors who earn more than 10,000.

SCRIPT:

```
SELECT First_name, Birth_date, Address, Salary
FROM Doctor
WHERE Salary > 100000 AND Gender = ‘Male’;
```

OUTPUT:



```
SELECT First_name,Birth_date, Address, Salary
    FROM DOCTOR
    WHERE salary > 100000 AND Gender = 'Male';
```

All Rows Fetched: 2 in 0.053 seconds

	FIRST_NAME	BIRTH_DATE	ADDRESS	SALARY
1	Ross	10-01-86	Mumbai	200000
2	Deepak	11-01-67	Delhi	300000

3. FUNCTIONS

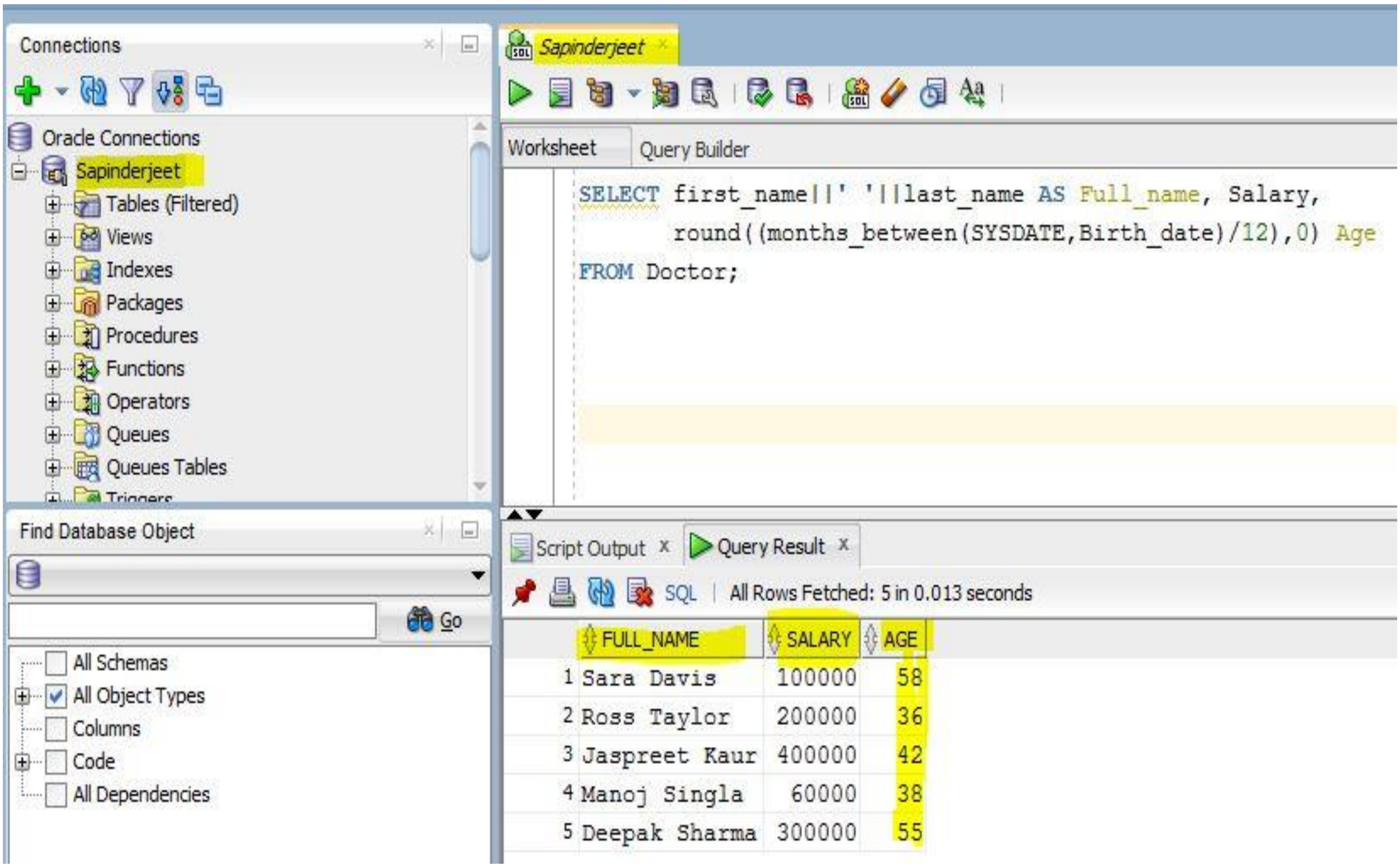
3.1) Find the age of all the doctors and show the full name in one columns as Full name with space.

Functions Used: Round, Months_between

SCRIPT:

```
SELECT first_name||' '||last_name AS Full_name, Salary,
round ((months_between (SYSDATE, Birth_date)/12),0) Age
FROM Doctor;
```

OUTPUT:



```
SELECT first_name||' '||last_name AS Full_name, Salary,
       round((months_between(SYSDATE,Birth_date)/12),0) Age
FROM Doctor;
```

All Rows Fetched: 5 in 0.013 seconds

	FULL_NAME	SALARY	AGE
1	Sara Davis	100000	58
2	Ross Taylor	200000	36
3	Jaspreet Kaur	400000	42
4	Manoj Singla	60000	38
5	Deepak Sharma	300000	55

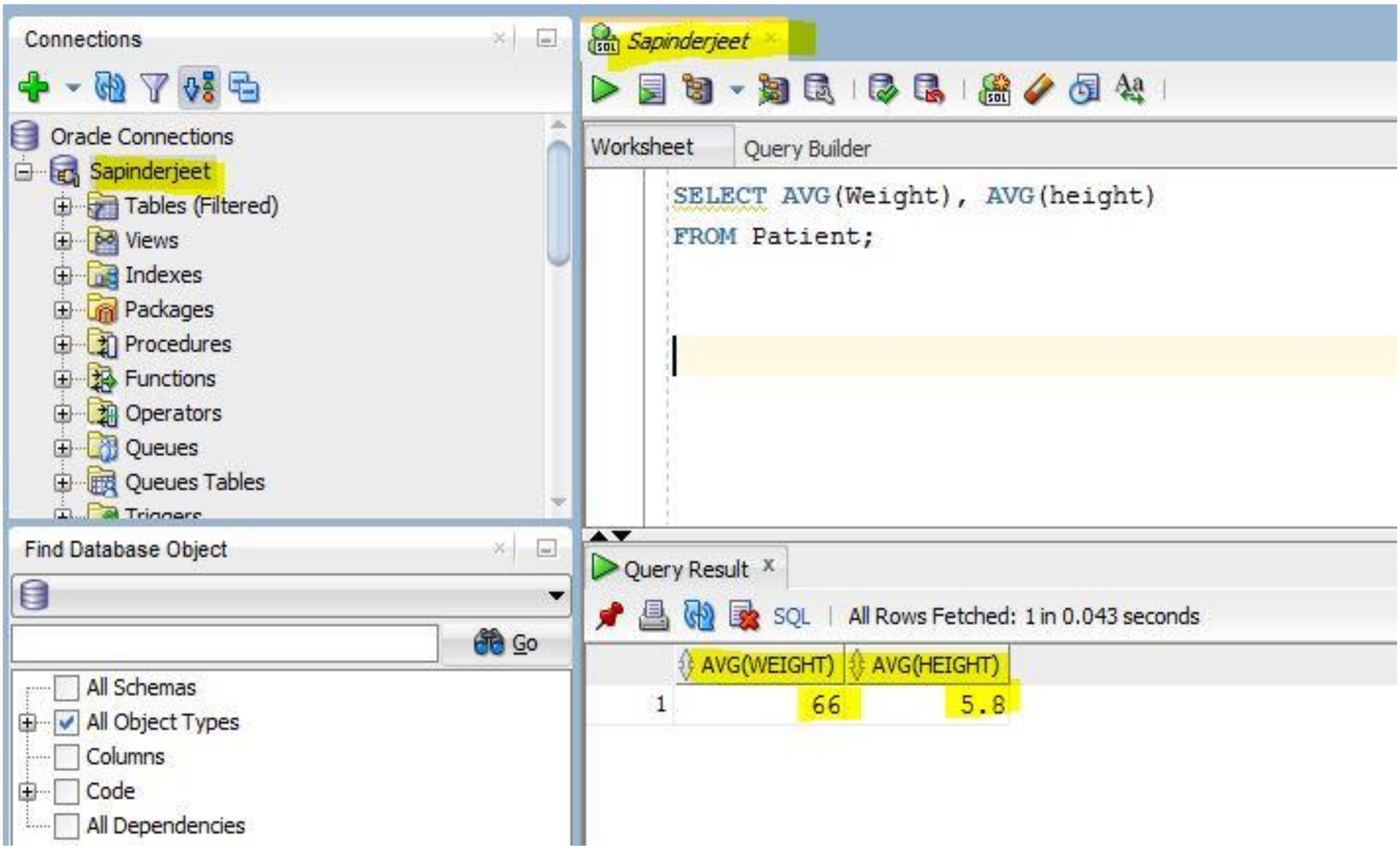
3.2) Find the average weight and height of all the patients.

Functions Used: Average(Avg)

Script:

```
SELECT AVG(Weight), AVG(height)
FROM Patient;
```

OUTPUT:



```
SELECT AVG(Weight), AVG(height)
FROM Patient;
```

All Rows Fetched: 1 in 0.043 seconds

```
   AVG(WEIGHT)  AVG(HEIGHT)
1           66          5.8
```

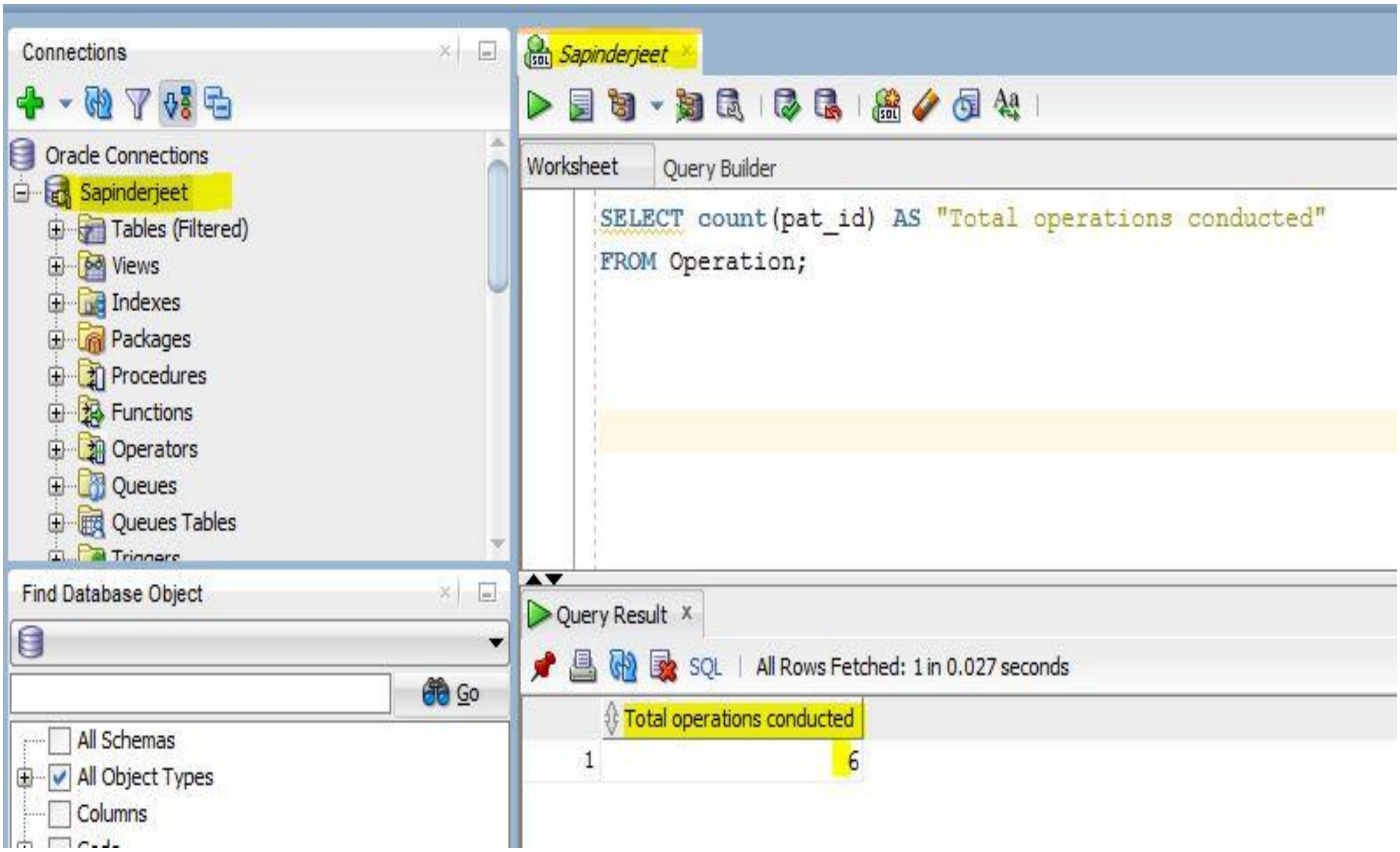
3.3) Find total number of operations conducted in the hospital.

Functions Used: Count

Script:

```
SELECT count(pat_id) AS "Total operations conducted"
FROM Operation;
```

OUTPUT:



```
SELECT count(pat_id) AS "Total operations conducted"
FROM Operation;
```

```
   Total operations conducted
1                           6
```

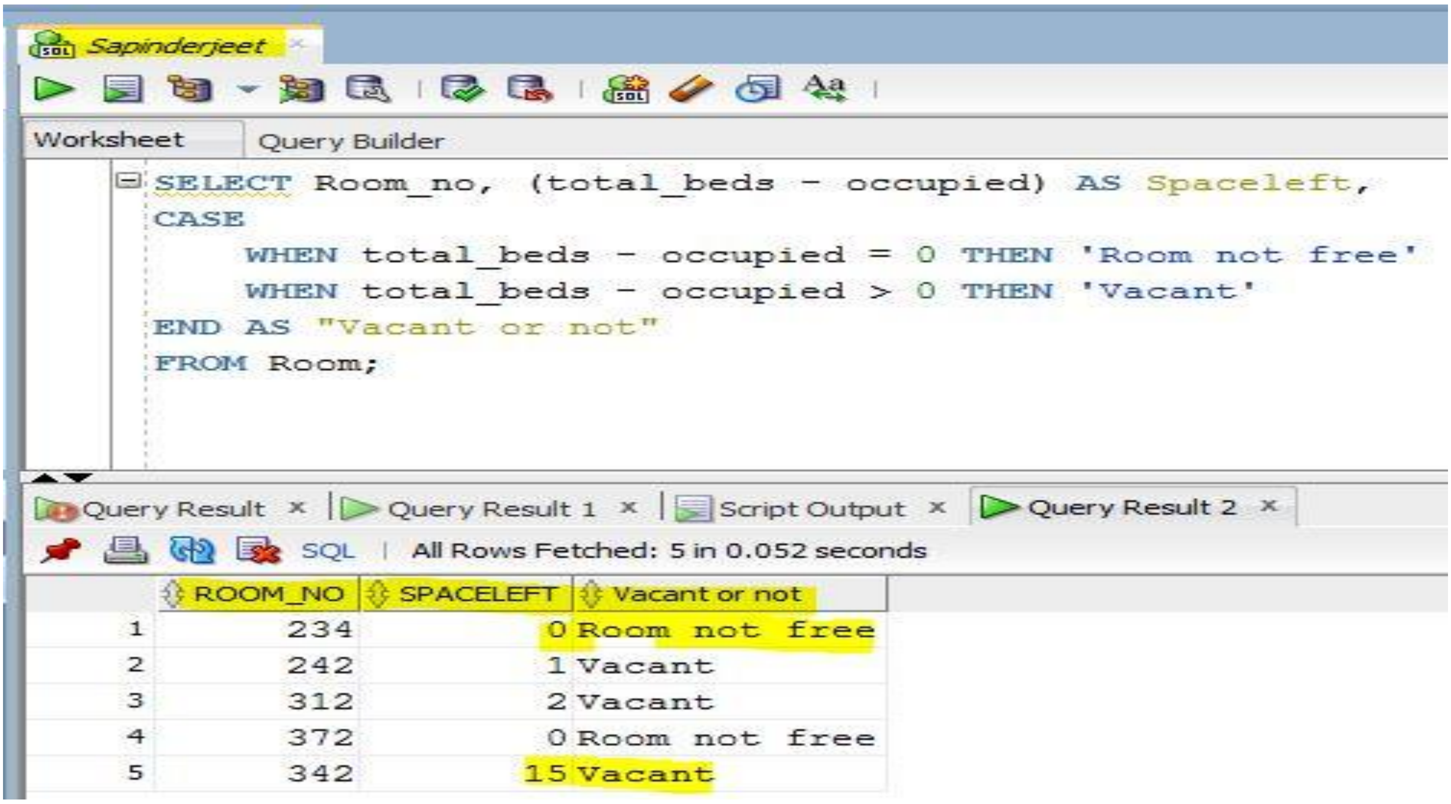
3.4) Find number of beds free in each room. Also if bed is not free show "Occupied" else show "Vacant".

Function Used: Select Case

Script:

```
SELECT Room_no, (total_beds - occupied) AS Spaceleft,
CASE
      WHEN total_beds - occupied = 0 THEN 'Room not free'
     WHEN total_beds - occupied > 0 THEN 'Vacant'
END AS "Vacant or not"
FROM Room;
```

Output:



```
SELECT Room_no, (total_beds - occupied) AS Spaceleft,
CASE
    WHEN total_beds - occupied = 0 THEN 'Room not free'
    WHEN total_beds - occupied > 0 THEN 'Vacant'
END AS "Vacant or not"
FROM Room;
```

All Rows Fetched: 5 in 0.052 seconds

	ROOM_NO	SPACELEFT	Vacant or not
1	234	0	Room not free
2	242	1	Vacant
3	312	2	Vacant
4	372	0	Room not free
5	342	15	Vacant

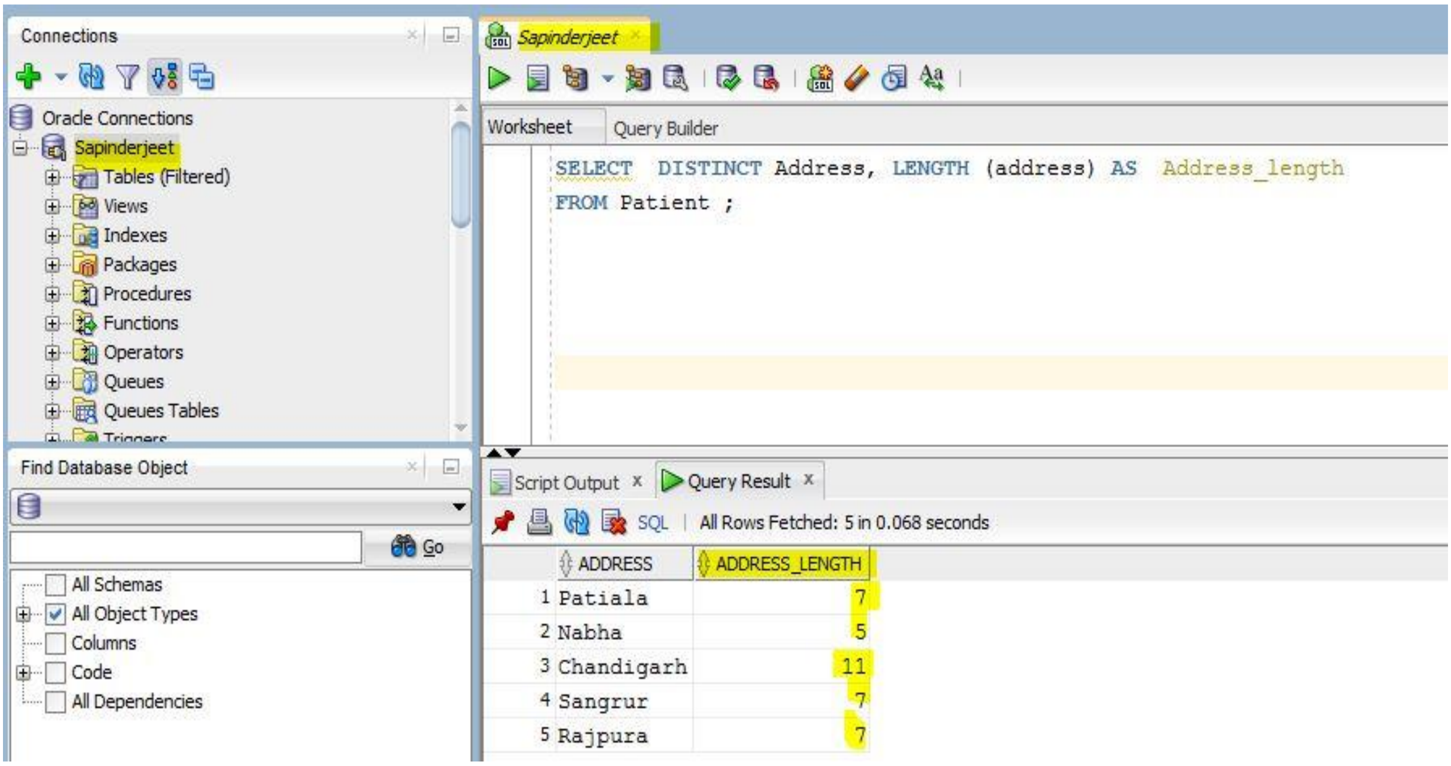
3.5) Display the address of all the patient along with the length of the place.

Functions Used: Length

Script:

```
SELECT DISTINCT Address, LENGTH (address) AS Address_length
FROM Patient;
```

OUTPUT:



```
SELECT DISTINCT Address, LENGTH (address) AS Address_length
FROM Patient ;
```

All Rows Fetched: 5 in 0.068 seconds

	ADDRESS	ADDRESS_LENGTH
1	Patiala	7
2	Nabha	5
3	Chandigarh	11
4	Sangrur	7
5	Rajpura	7

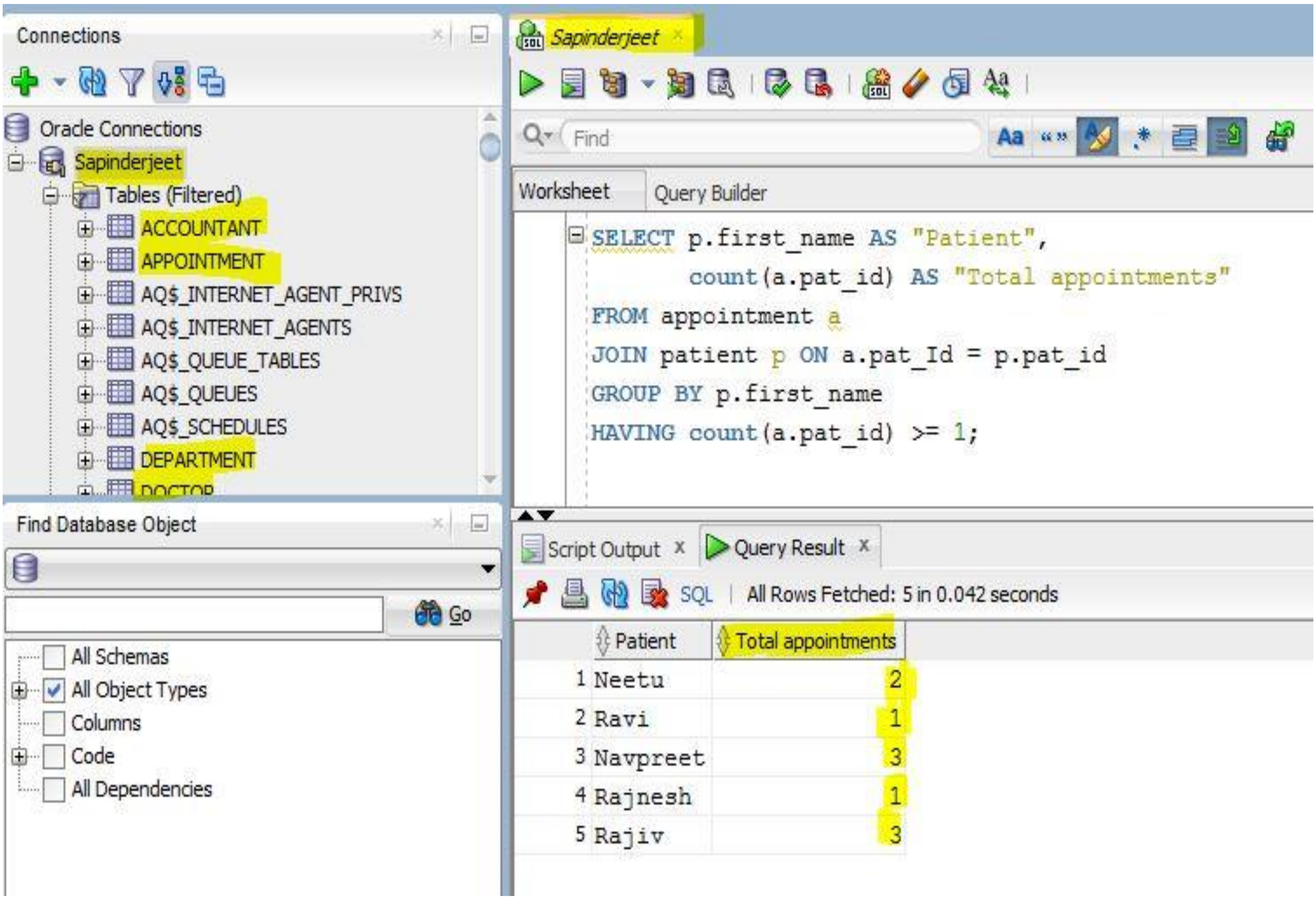
4. Group By

4.1) Write a query in SQL to find the name of the patients and the number of Appointments they have taken group by first_name.

Script:

```
SELECT p.first_name AS "Patient",
          count(a.pat_id) AS "Total appointments"
FROM appointment a
JOIN patient p ON a.pat_Id = p.pat_id
GROUP BY p.first_name
HAVING count(a.pat_id) >= 1;
```

OUTPUT:



```
SELECT p.first_name AS "Patient",
       count(a.pat_id) AS "Total appointments"
FROM appointment a
JOIN patient p ON a.pat_Id = p.pat_id
GROUP BY p.first_name
HAVING count(a.pat_id) >= 1;
```

All Rows Fetched: 5 in 0.042 seconds

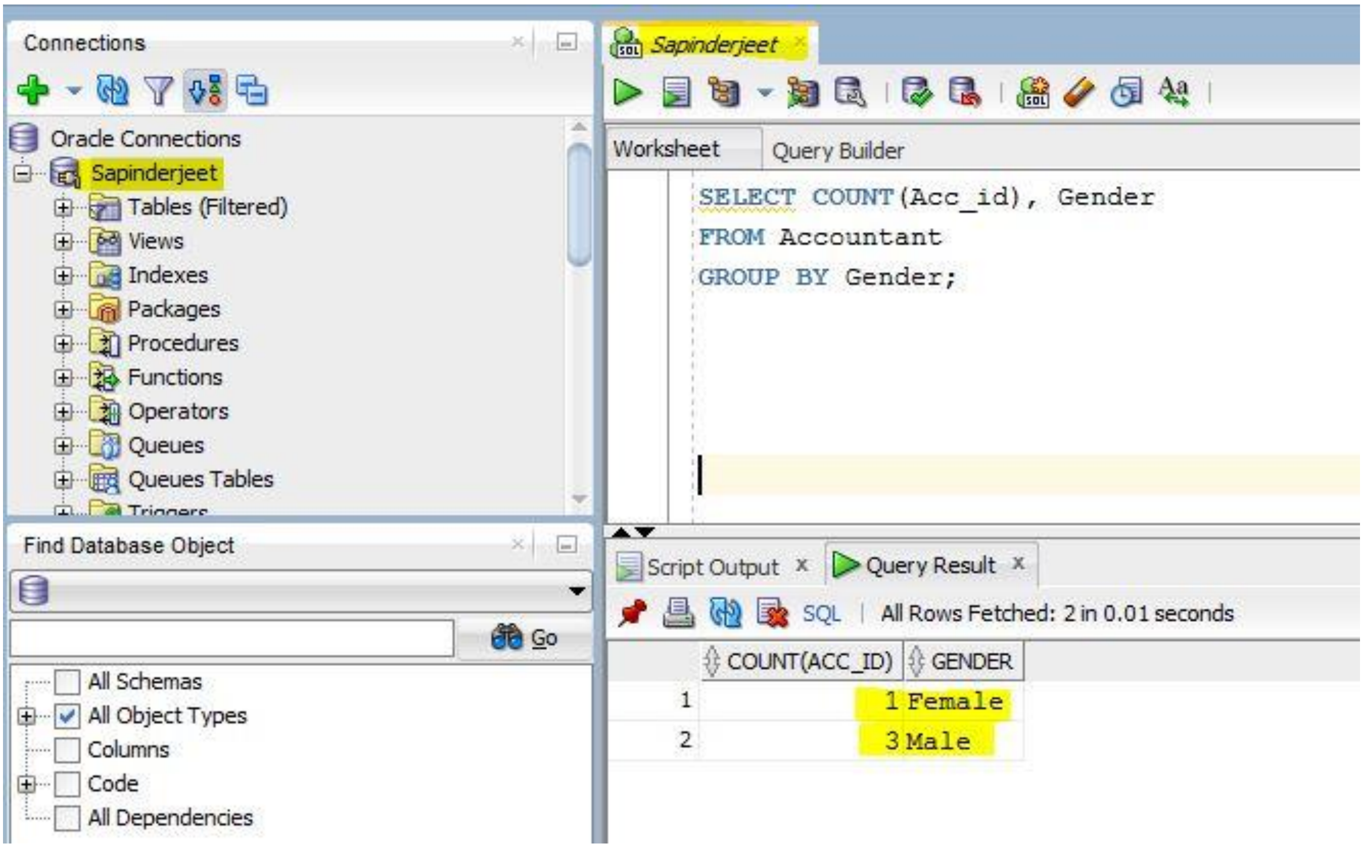
	Patient	Total appointments
1	Neetu	2
2	Ravi	1
3	Navpreet	3
4	Rajnesh	1
5	Rajiv	3

4.2) Use Group by to find total number of male and female accountant present in the hospital.

SCRIPT:

```
SELECT COUNT(Acc_id), Gender
FROM Accountant
GROUP BY Gender;
```

OUTPUT:



```
SELECT COUNT(Acc_id), Gender
FROM Accountant
GROUP BY Gender;
```

All Rows Fetched: 2 in 0.01 seconds

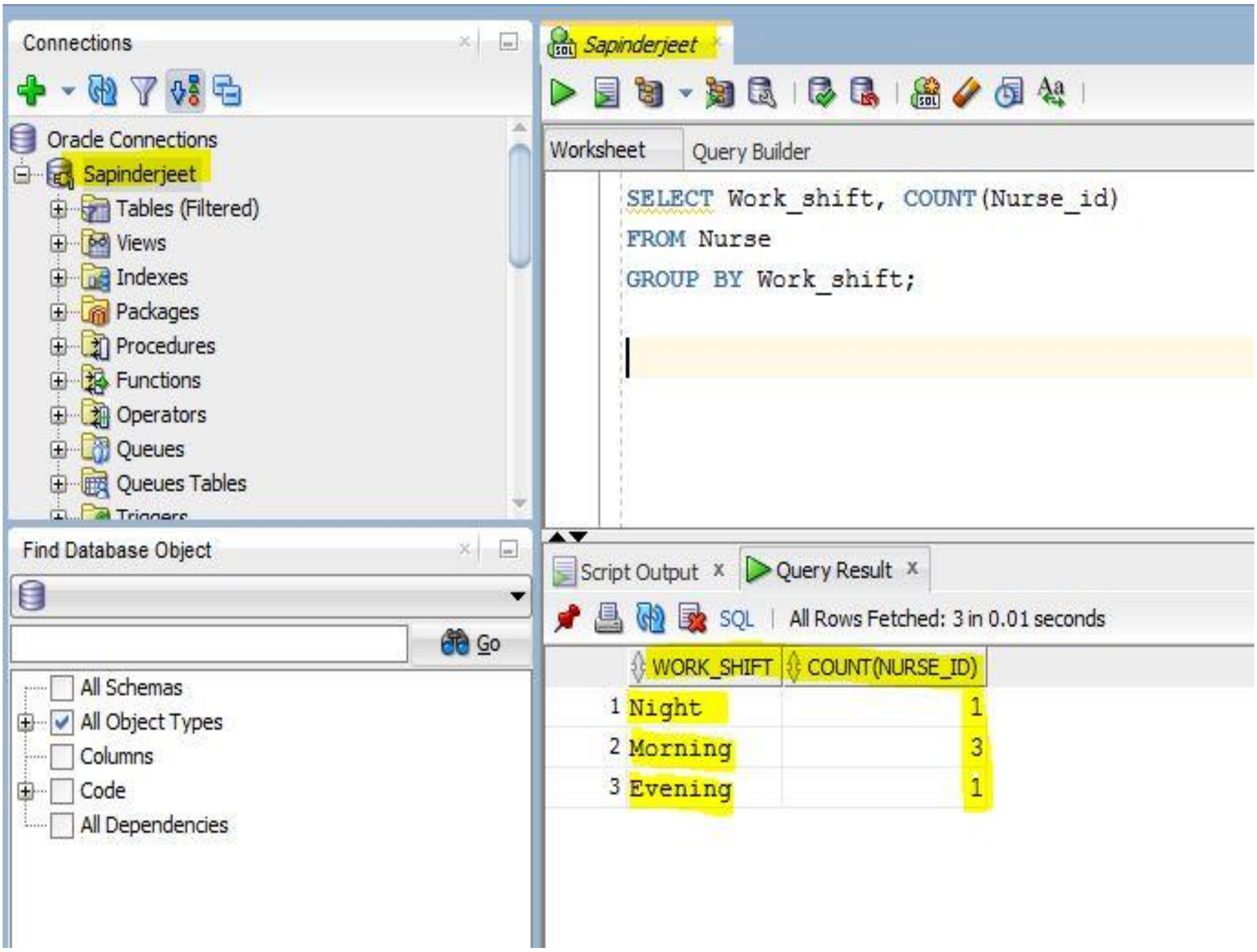
```
     COUNT(ACC_ID)  GENDER
1                1  Female
2                3  Male
```

4.3) Use Group by to find total number of nurses working in different shifts i.e. Morning, Evening, Night.

SCRIPT:

```
SELECT Work_shift, COUNT(Nurse_id)
FROM Nurse
GROUP BY Work_shift;
```

OUTPUT:



```
SELECT Work_shift, COUNT(Nurse_id)
FROM Nurse
GROUP BY Work_shift;
```

All Rows Fetched: 3 in 0.01 seconds

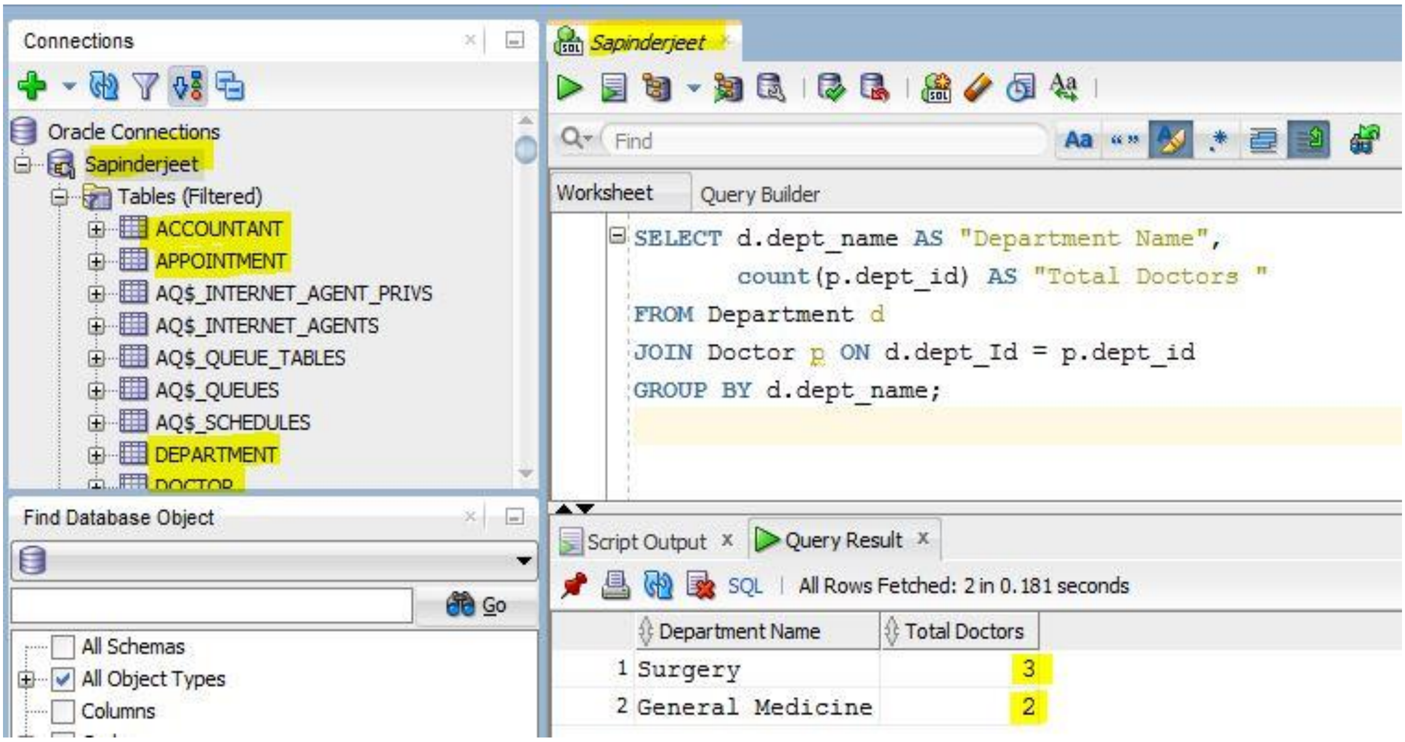
```
   WORK_SHIFT  COUNT(NURSE_ID)
1  Night                     1
2  Morning                   3
3  Evening                   1
```

4.4) Write a query in SQL to find the total doctors belonging to each department group by Dept_name.

SCRIPT:

```
SELECT  d.dept_name  AS  "Department Name",
             Count (p.dept_id)  AS  "Total Doctors "
FROM Department d
JOIN Doctor p ON d.dept_Id = p.dept_id
GROUP BY d.dept_name;
```

OUTPUT:



```
SELECT d.dept_name AS "Department Name",
              count(p.dept_id) AS "Total Doctors "
FROM Department d
JOIN Doctor p ON d.dept_Id = p.dept_id
GROUP BY d.dept_name;
```

All Rows Fetched: 2 in 0.181 seconds

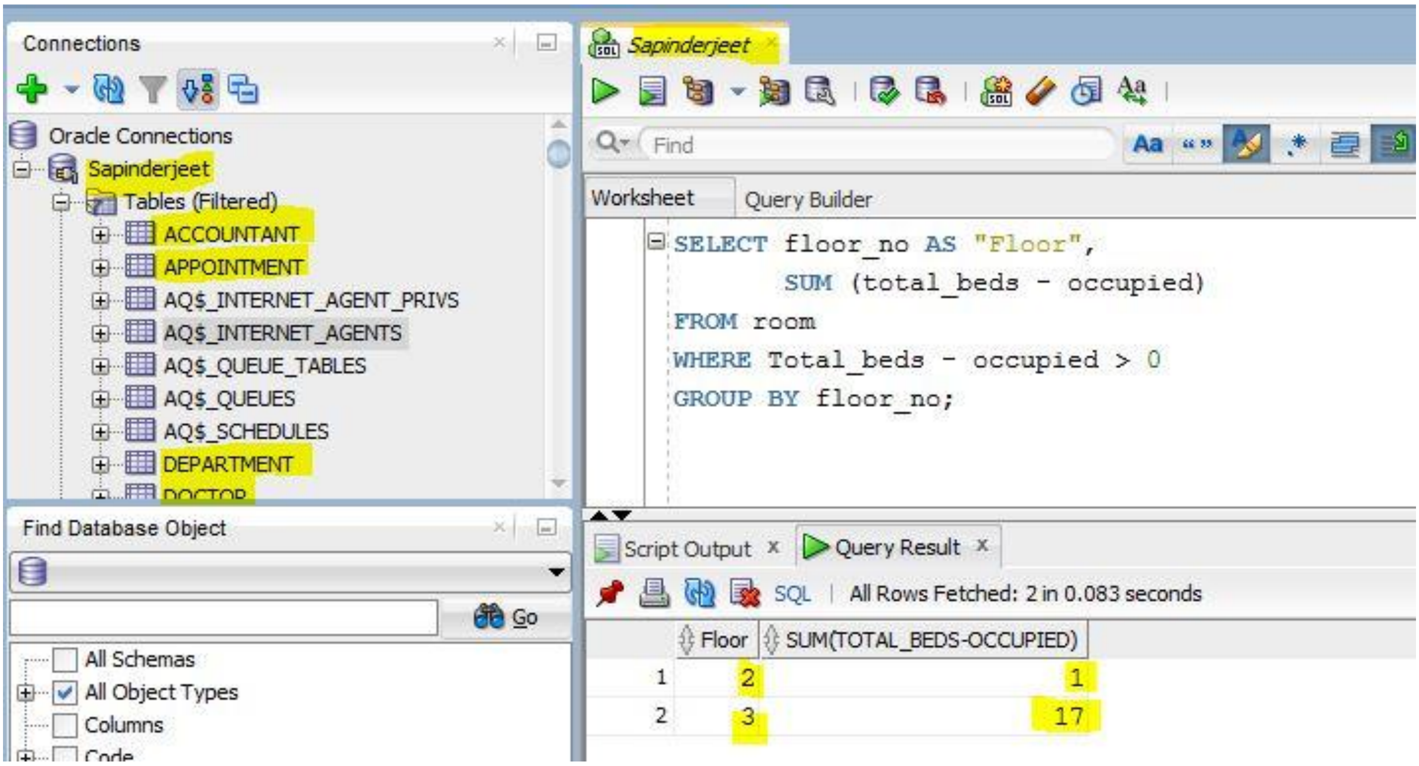
	Department Name	Total Doctors
1	Surgery	3
2	General Medicine	2

4.5) Write a query in SQL to find the total free beds on each floor group by Floor_no.

SCRIPT:

```
SELECT floor_no AS "Floor",
            SUM (total_beds - occupied)
FROM room
WHERE Total_beds - occupied > 0
GROUP BY floor_no;
```

OUTPUT:



```
SELECT floor_no AS "Floor",
            SUM (total_beds - occupied)
FROM room
WHERE Total_beds - occupied > 0
GROUP BY floor_no;
```

All Rows Fetched: 2 in 0.083 seconds

	Floor	SUM(TOTAL_BEDS-OCCUPIED)
1	2	1
2	3	17

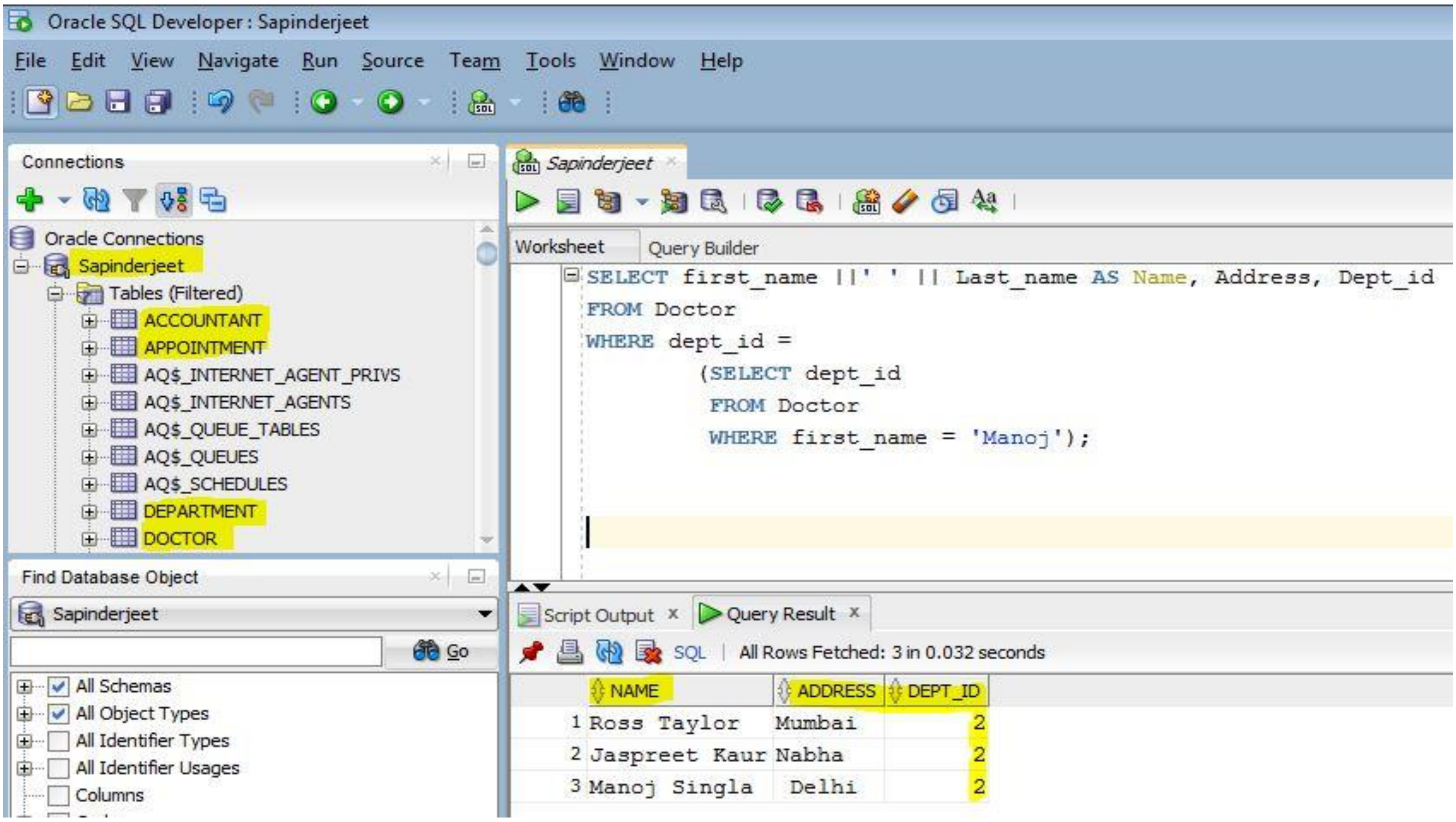
5. SUB QUERIES

5.1) Find the name and address of all doctors which works in the same department as the doctor "Manoj".

SCRIPT:

```
SELECT first_name ||' ' || Last_name AS Name, Address
FROM Doctor
WHERE dept_id =
            (SELECT dept_id
             FROM Doctor
             WHERE first_name = 'Manoj');
```

OUTPUT:



```
SELECT first_name ||' ' || Last_name AS Name, Address, Dept_id
FROM Doctor
WHERE dept_id =
            (SELECT dept_id
             FROM Doctor
             WHERE first_name = 'Manoj');
```

All Rows Fetched: 3 in 0.032 seconds

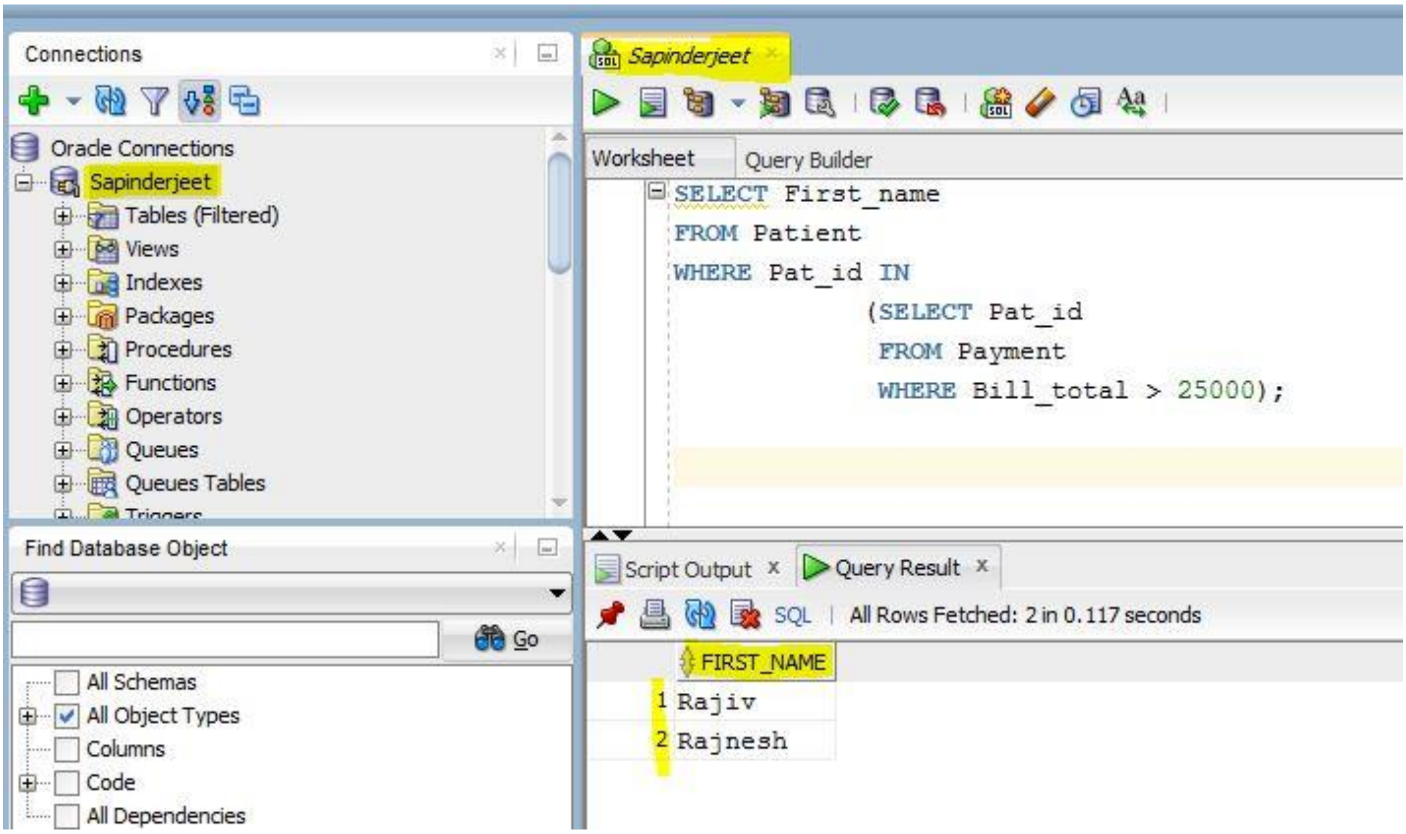
	NAME	ADDRESS	DEPT_ID
1	Ross Taylor	Mumbai	2
2	Jaspreet Kaur	Nabha	2
3	Manoj Singla	Delhi	2

5.2) List name all the patients who have total_bill greater than 25,000 using sub queries.

SCRIPT:

```
SELECT First_name
 FROM Patient
 WHERE Pat_id IN
                    (SELECT Pat_id
                     FROM Payment
                     WHERE Bill_total > 25000);
```

OUTPUT:



```
SELECT First_name
FROM Patient
WHERE Pat_id IN
              (SELECT Pat_id
               FROM Payment
               WHERE Bill_total > 25000);
```

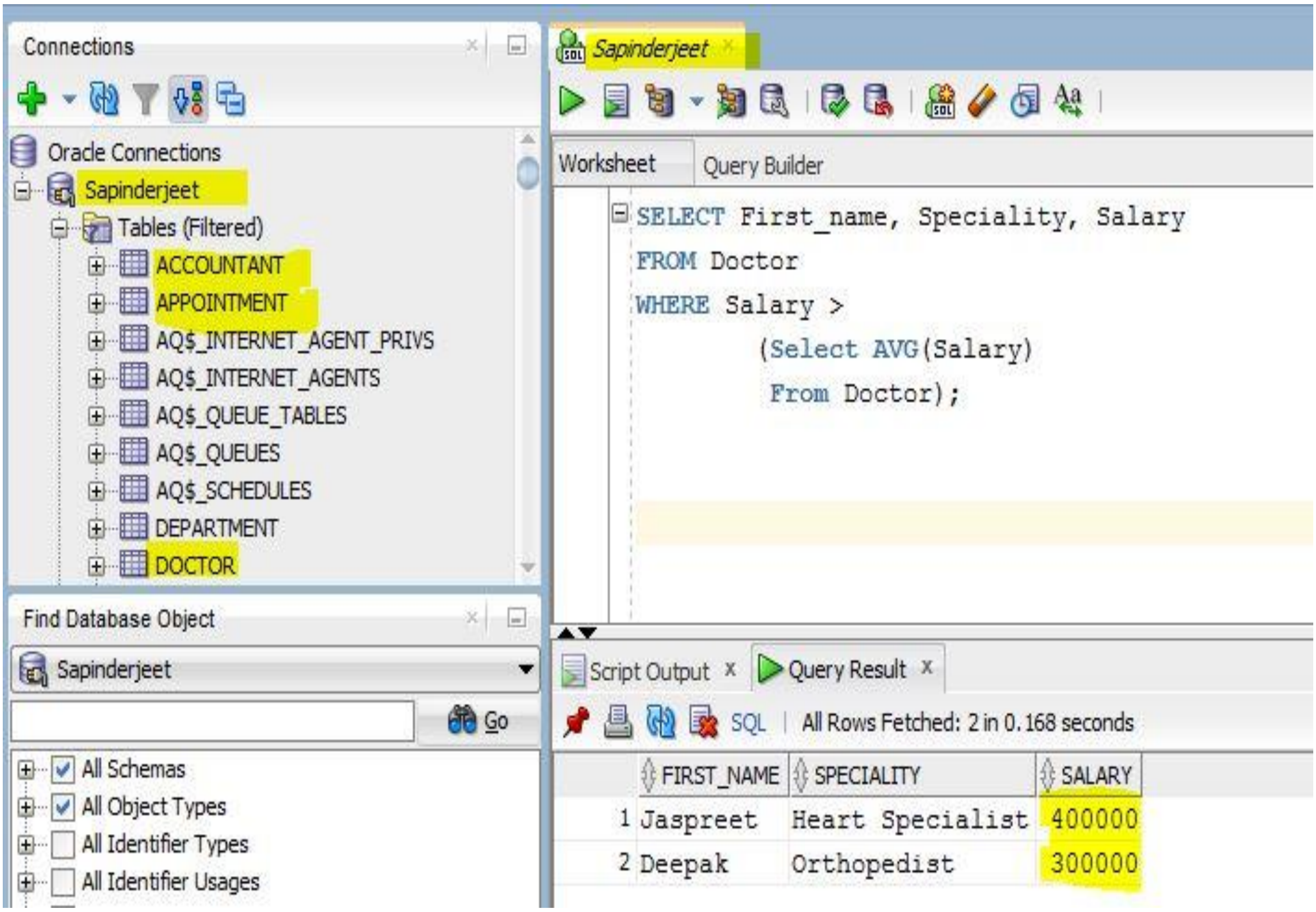
```
  FIRST_NAME
1 Rajiv
2 Rajnesh
```

5.3) Display all the doctors which have salary greater than the average salary.

SCRIPT:

```
SELECT First_name, Speciality, Salary
 FROM Doctor
 WHERE Salary >
                  (Select  AVG (Salary)
                   From Doctor);
```

OUTPUT:



```
SELECT First_name, Speciality, Salary
FROM Doctor
WHERE Salary >
        (Select AVG(Salary)
         From Doctor);
```

All Rows Fetched: 2 in 0.168 seconds

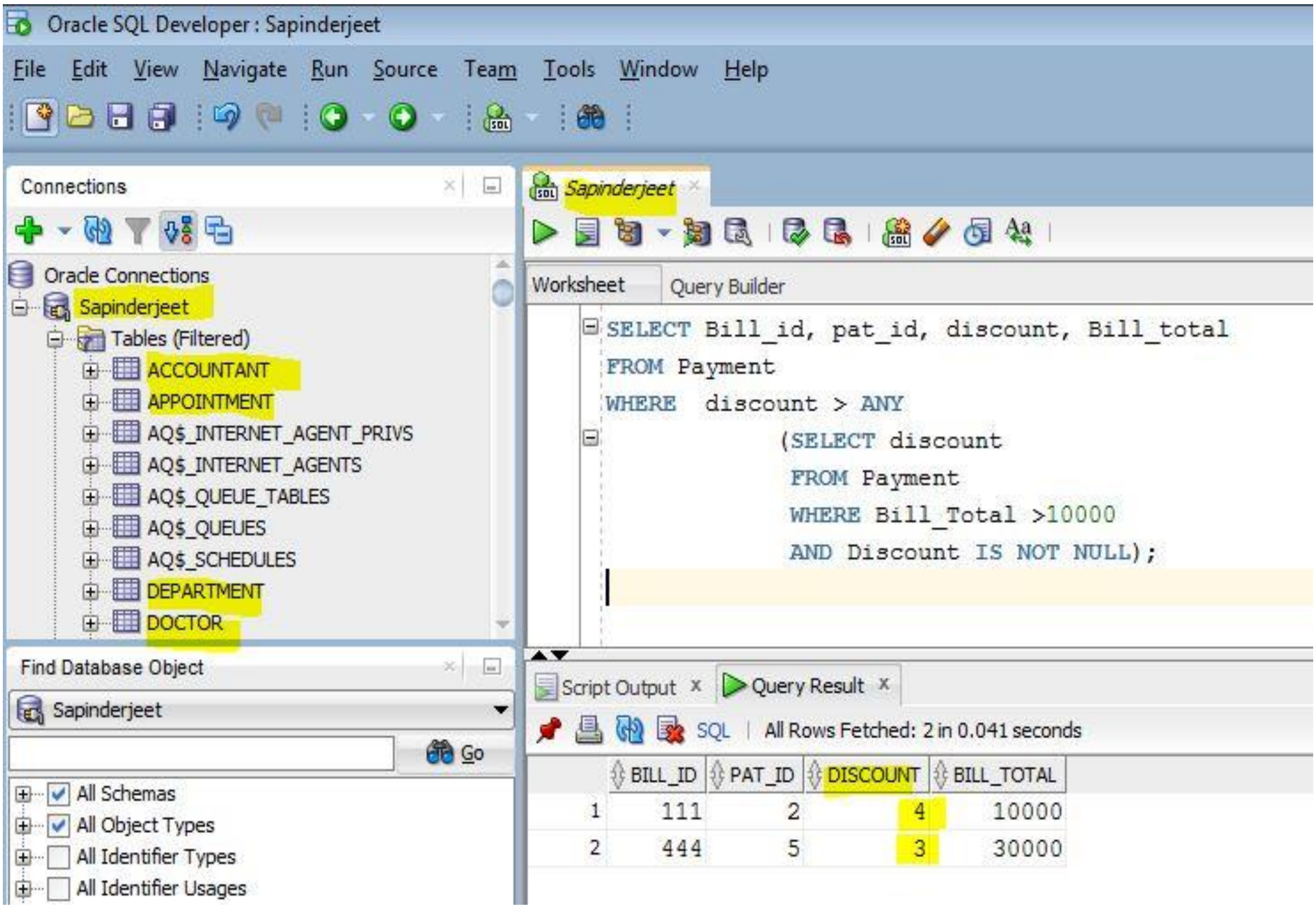
	FIRST_NAME	SPECIALITY	SALARY
1	Jaspreet	Heart Specialist	400000
2	Deepak	Orthopedist	300000

5.4) List all the payments where total_bill is greater than 10,000 and Discount is NOT NULL.

SCRIPT:

```
SELECT Bill_id, Pat_id, discount, Bill_total
 FROM Payment
 WHERE  discount > ANY
              (SELECT discount
               FROM Payment
               WHERE Bill_Total >10000
               AND Discount IS NOT NULL);
```

OUTPUT:



```
SELECT Bill_id, pat_id, discount, Bill_total
FROM Payment
WHERE  discount > ANY
             (SELECT discount
              FROM Payment
              WHERE Bill_Total >10000
              AND Discount IS NOT NULL);
```

All Rows Fetched: 2 in 0.041 seconds

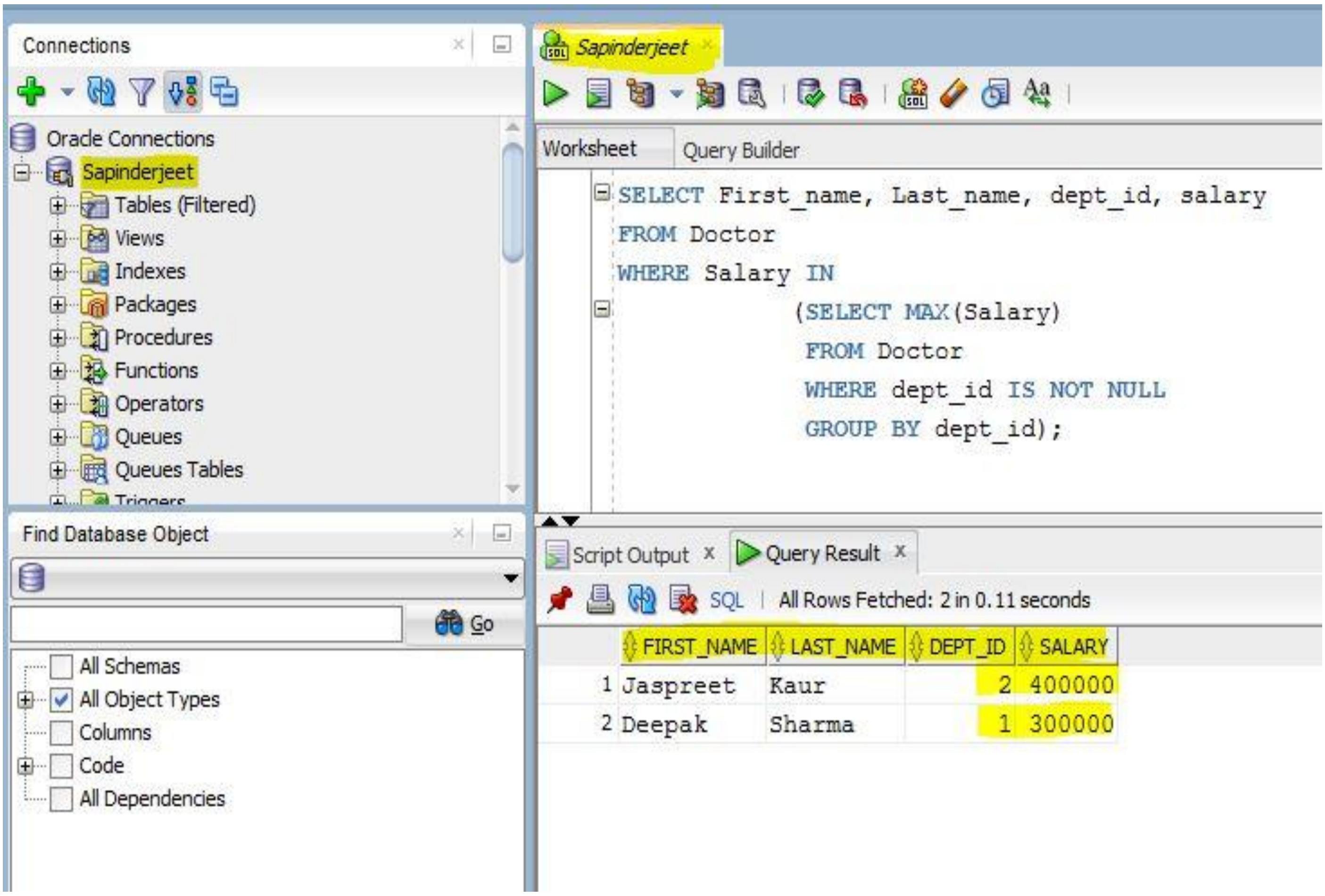
	BILL_ID	PAT_ID	DISCOUNT	BILL_TOTAL
1	111	2	4	10000
2	444	5	3	30000

5.4) Find which doctors has the highest salary department wise. Display their first name, last name, department id and salary using sub queries.

SCRIPT:

```
SELECT First_name, Last_name, dept_id, salary
FROM Doctor
WHERE Salary IN
                (SELECT MAX(Salary)
                FROM Doctor
                WHERE dept_id IS NOT NULL
                GROUP BY dept_id);
```

OUTPUT:



```
SELECT First_name, Last_name, dept_id, salary
FROM Doctor
WHERE Salary IN
                (SELECT MAX(Salary)
                 FROM Doctor
                 WHERE dept_id IS NOT NULL
                 GROUP BY dept_id);
```

	FIRST_NAME	LAST_NAME	DEPT_ID	SALARY
1	Jaspreet	Kaur	2	400000
2	Deepak	Sharma	1	300000

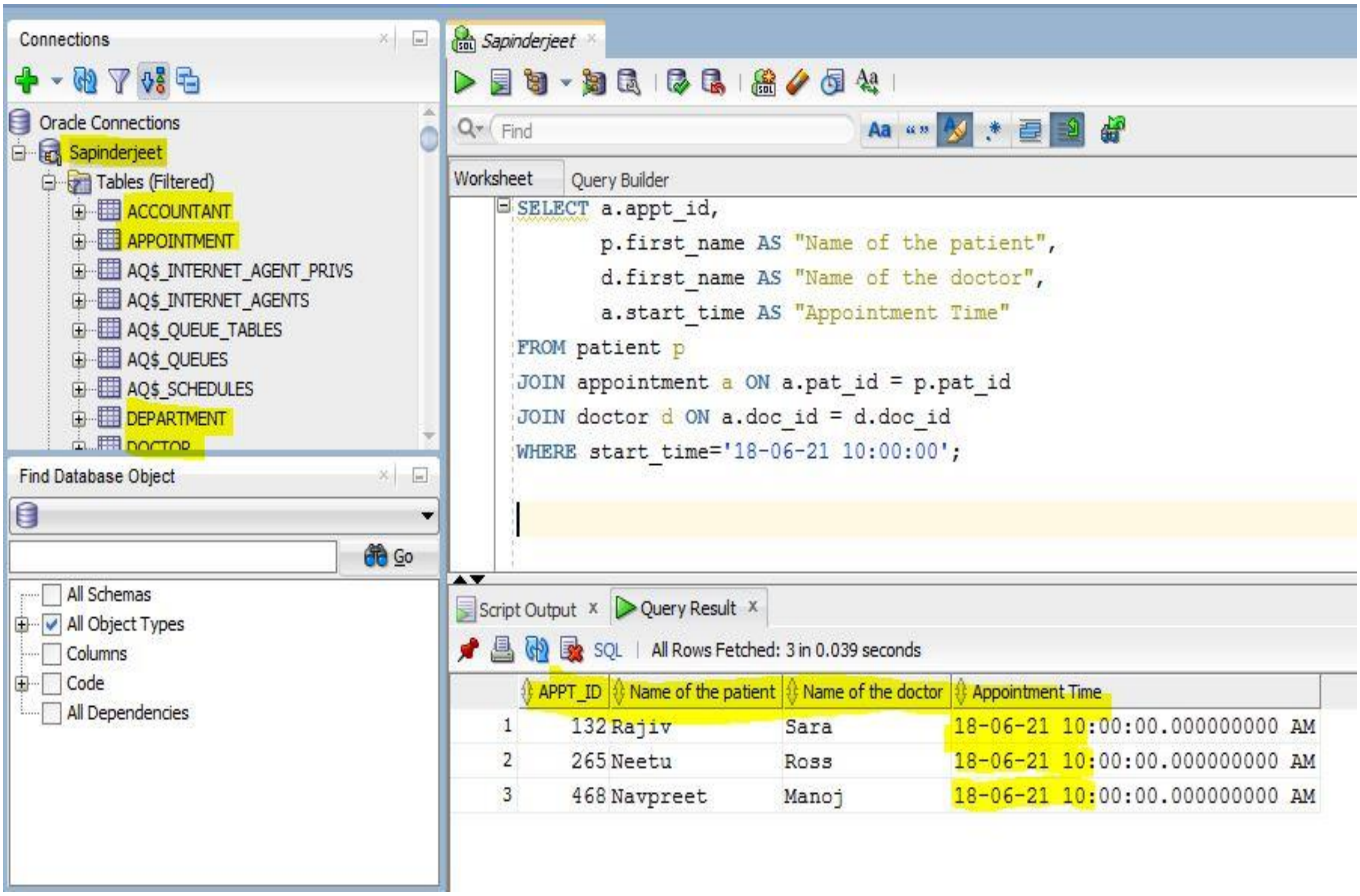
6. MULTIPLE TABLE QUERIES

6.1) Write a query in SQL to find the name of the patients who taken the appointment on the 18 June at 10 am, and also display their Doctor along with the Appointment id.

Script:

```
SELECT a.appt_id,
        p.first_name AS "Name of the patient",
        d.first_name AS "Name of the doctor",
        a.start_time AS "Appointment Time"
FROM patient p
JOIN appointment a ON a.pat_id = p.pat_id
JOIN doctor d ON a.doc_id = d.doc_id
WHERE start_time='18-06-21 10:00:00';
```

OUTPUT:



```
SELECT a.appt_id,
        p.first_name AS "Name of the patient",
        d.first_name AS "Name of the doctor",
        a.start_time AS "Appointment Time"
FROM patient p
JOIN appointment a ON a.pat_id = p.pat_id
JOIN doctor d ON a.doc_id = d.doc_id
WHERE start_time='18-06-21 10:00:00';
```

All Rows Fetched: 3 in 0.039 seconds

	APPT_ID	Name of the patient	Name of the doctor	Appointment Time
1	132	Rajiv	Sara	18-06-21 10:00:00.000000000 AM
2	265	Neetu	Ross	18-06-21 10:00:00.000000000 AM
3	468	Navpreet	Manoj	18-06-21 10:00:00.000000000 AM

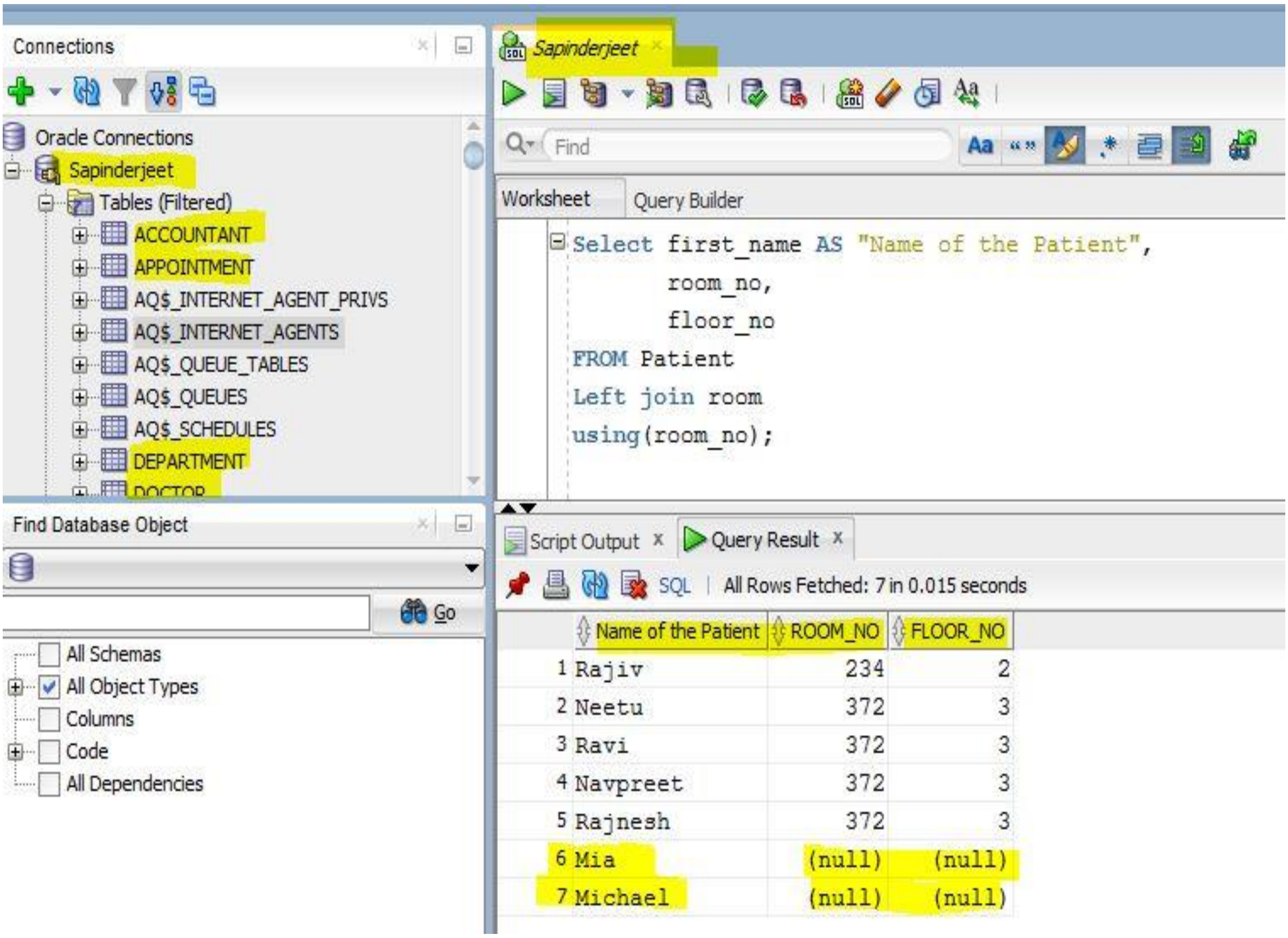
6.2) Find which patient is admitted in which room and which patients are not admitted.

Sol: Here left join will be used and those which will have null values will be the one that are not admitted in any room.

SCRIPT:

```
Select first_name AS "Name of the Patient",
        room_no,
        floor_no
FROM Patient
LEFT JOIN room USING (room_no);
```

OUTPUT:



```
Select first_name AS "Name of the Patient",
        room_no,
        floor_no
FROM Patient
Left join room
using(room_no);
```

All Rows Fetched: 7 in 0.015 seconds

	Name of the Patient	ROOM_NO	FLOOR_NO
1	Rajiv	234	2
2	Neetu	372	3
3	Ravi	372	3
4	Navpreet	372	3
5	Rajnesh	372	3
6	Mia	(null)	(null)
7	Michael	(null)	(null)

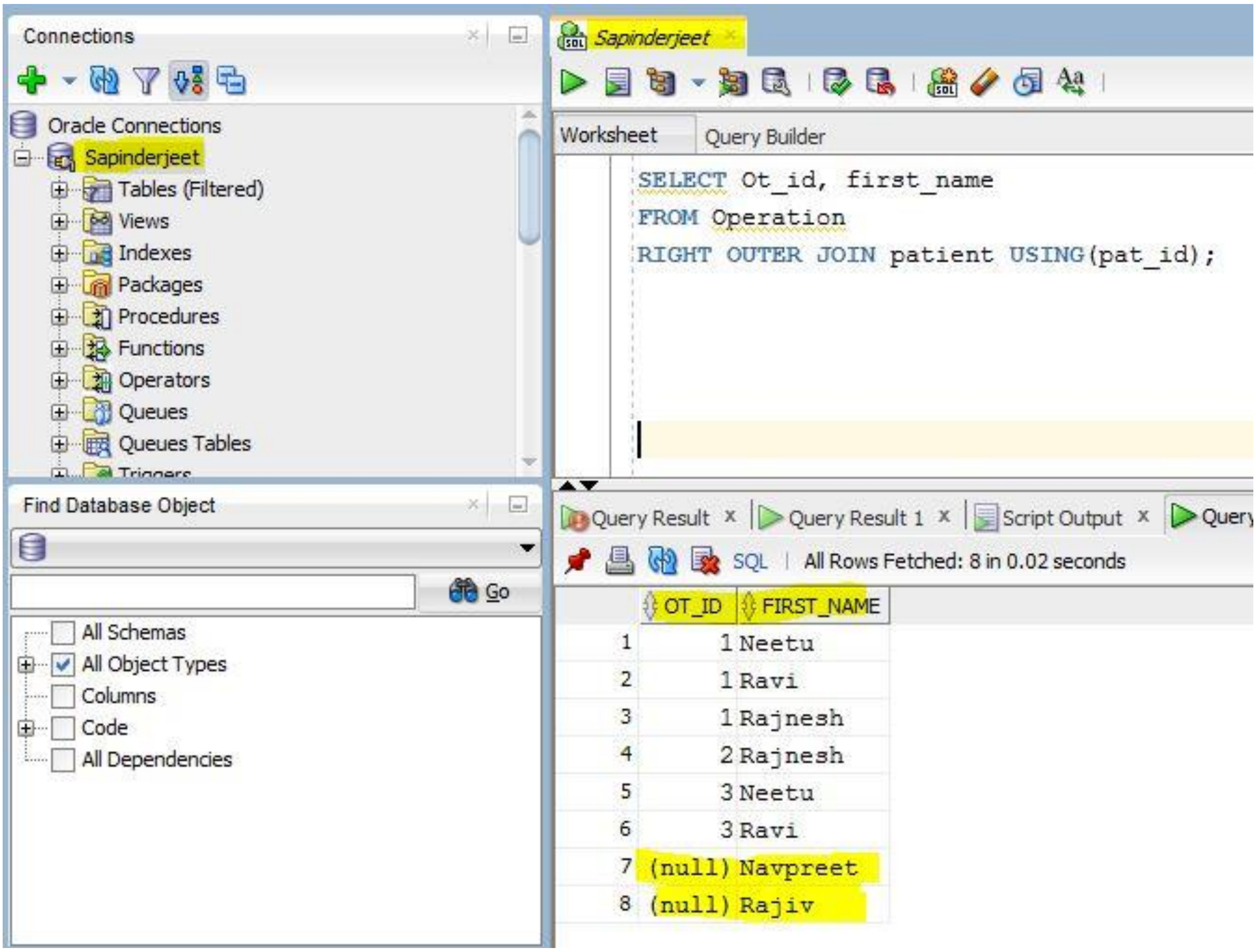
6.3) Display all the patients along with their operation id and also those patients who did not have any operation scheduled.

Sol: Here right join will be used and those which will have null values will be the one that do not have any operation scheduled.

SCRIPT:

```
SELECT Ot_id, first_name
FROM operation
RIGHT OUTER JOIN patient USING (pat_id);
```

OUTPUT:



```
SELECT Ot_id, first_name
FROM Operation
RIGHT OUTER JOIN patient USING(pat_id);
```

All Rows Fetched: 8 in 0.02 seconds

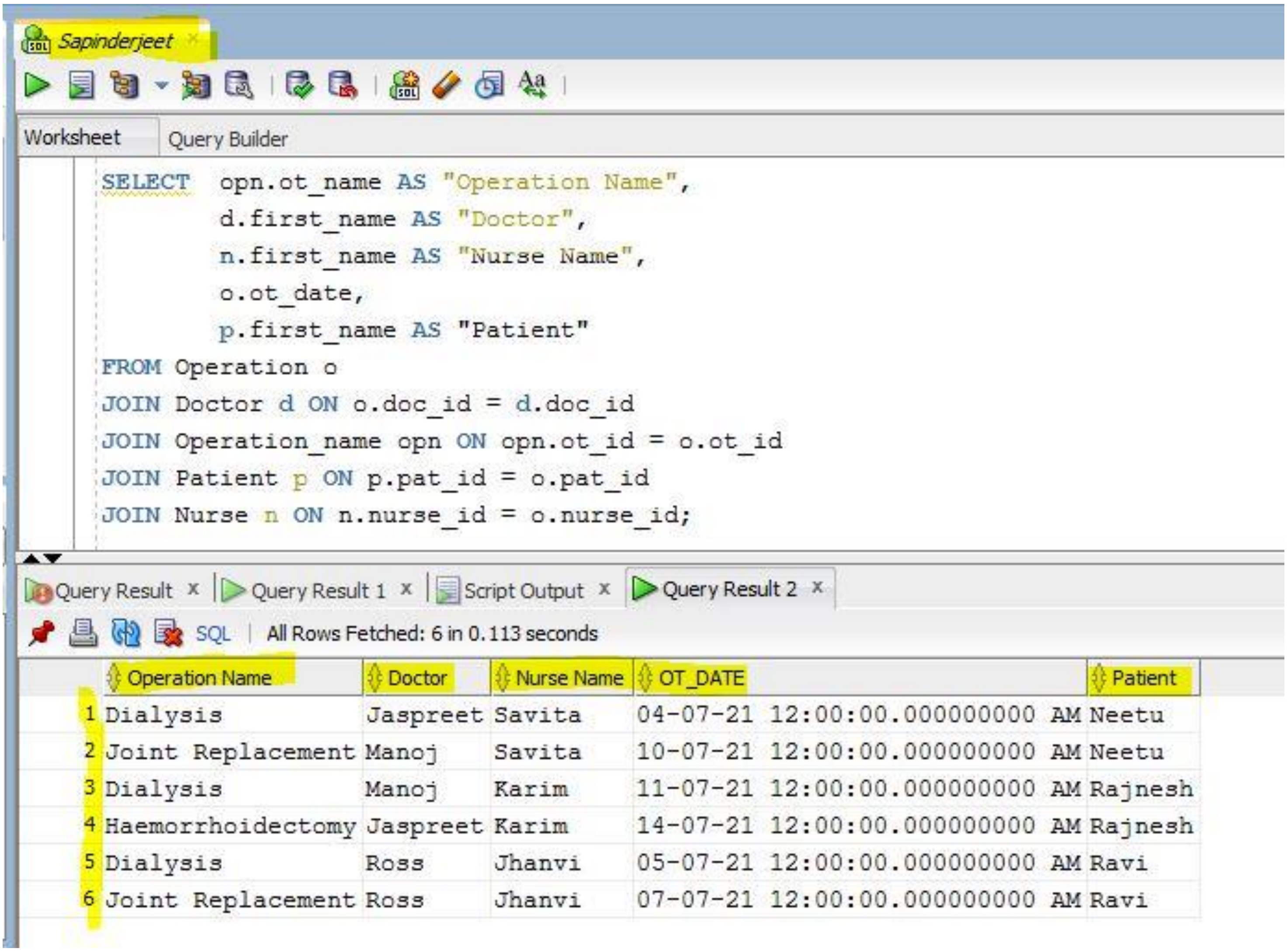
	OT_ID	FIRST_NAME
1	1	Neetu
2	1	Ravi
3	1	Rajnesh
4	2	Rajnesh
5	3	Neetu
6	3	Ravi
7	(null)	Navpreet
8	(null)	Rajiv

6.4) List all the operations conducted by the doctors along with the nurse assisting it with date and patient name.

SCRIPT:

```
SELECT  opn.ot_name AS "Operation Name",
        d.first_name AS "Doctor",
        n.first_name AS "Nurse Name",
        o.ot_date,
        p.first_name AS "Patient"
FROM Operation o
JOIN Doctor d ON o.doc_id = d.doc_id
JOIN Operation_name opn ON opn.ot_id = o.ot_id
JOIN Patient p ON p.pat_id = o.pat_id
JOIN Nurse n ON n.nurse_id = o.nurse_id;
```

OUTPUT:



```
SELECT  opn.ot_name AS "Operation Name",
        d.first_name AS "Doctor",
        n.first_name AS "Nurse Name",
        o.ot_date,
        p.first_name AS "Patient"
FROM Operation o
JOIN Doctor d ON o.doc_id = d.doc_id
JOIN Operation_name opn ON opn.ot_id = o.ot_id
JOIN Patient p ON p.pat_id = o.pat_id
JOIN Nurse n ON n.nurse_id = o.nurse_id;
```

All Rows Fetched: 6 in 0.113 seconds

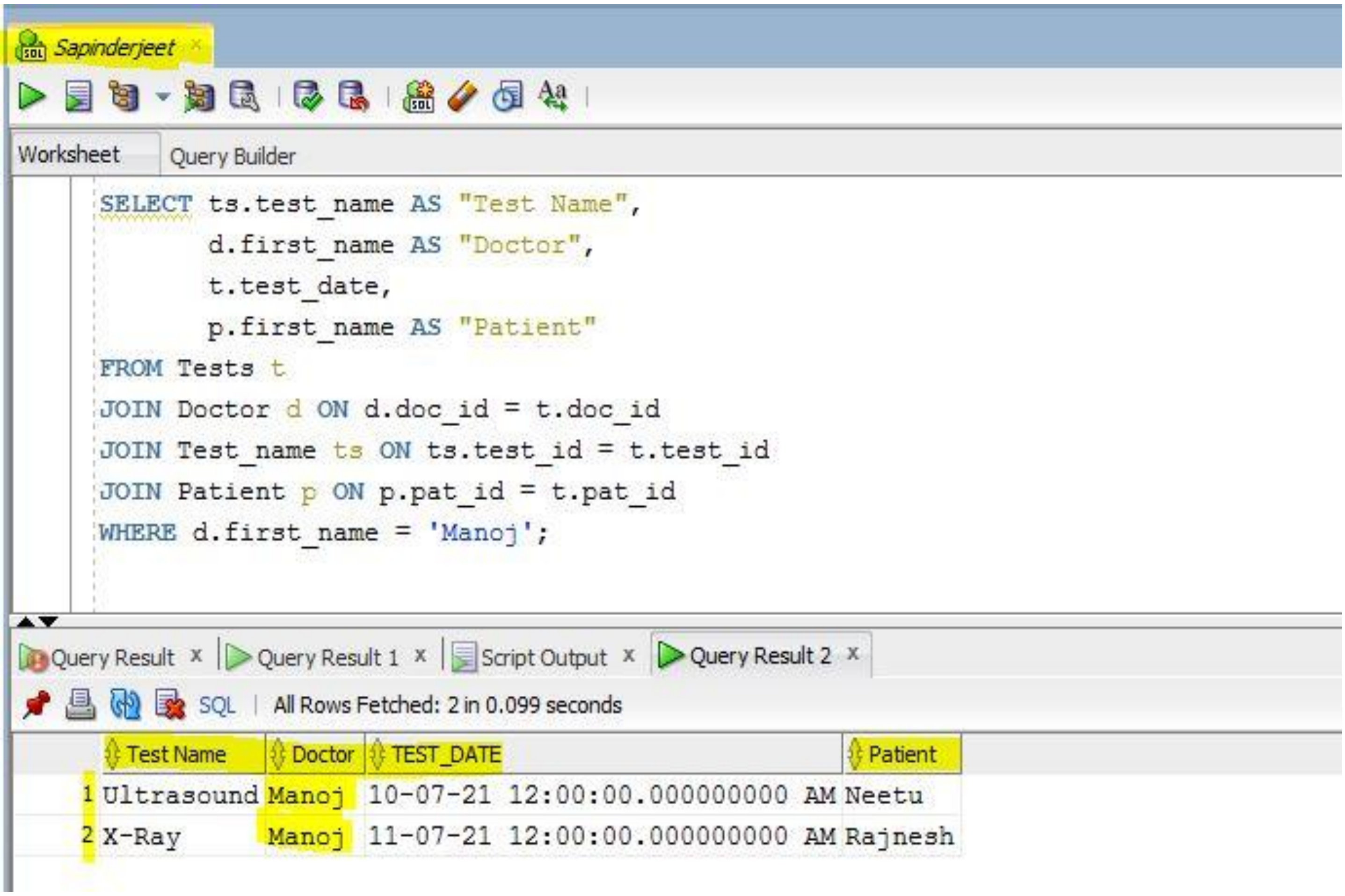
	Operation Name	Doctor	Nurse Name	OT_DATE	Patient
1	Dialysis	Jaspreet	Savita	04-07-21 12:00:00.000000000 AM	Neetu
2	Joint Replacement	Manoj	Savita	10-07-21 12:00:00.000000000 AM	Neetu
3	Dialysis	Manoj	Karim	11-07-21 12:00:00.000000000 AM	Rajnesh
4	Haemorrhoidectomy	Jaspreet	Karim	14-07-21 12:00:00.000000000 AM	Rajnesh
5	Dialysis	Ross	Jhanvi	05-07-21 12:00:00.000000000 AM	Ravi
6	Joint Replacement	Ross	Jhanvi	07-07-21 12:00:00.000000000 AM	Ravi

6.5) Find all the test which “Doctor Manoj” assign to patients.

SCRIPT:

```
SELECT  ts.test_name AS "Test Name",
        d.first_name AS "Doctor",
        t.test_date,
        p.first_name AS "Patient"
FROM Tests t
JOIN Doctor d ON d.doc_id = t.doc_id
JOIN Test_name ts ON ts.test_id = t.test_id
JOIN Patient p ON p.pat_id = t.pat_id
WHERE d.first_name = 'Manoj';
```

OUTPUT:



```
SELECT ts.test_name AS "Test Name",
       d.first_name AS "Doctor",
       t.test_date,
       p.first_name AS "Patient"
FROM Tests t
JOIN Doctor d ON d.doc_id = t.doc_id
JOIN Test_name ts ON ts.test_id = t.test_id
JOIN Patient p ON p.pat_id = t.pat_id
WHERE d.first_name = 'Manoj';
```

All Rows Fetched: 2 in 0.099 seconds

	Test Name	Doctor	TEST_DATE	Patient
1	Ultrasound	Manoj	10-07-21 12:00:00.000000000 AM	Neetu
2	X-Ray	Manoj	11-07-21 12:00:00.000000000 AM	Rajnesh