AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

PROJECT TITLE: DOCTOR APPOINTMENT MANAGEMENT SYSTEM

GROUP MEMBERS:

NAME	ID	CONTRIBUTION	TOP NAME
MD. EKRAMUL HASAN NIBIR	20-42740-1	25%	SCENARIO DESCRIPTION, TABLE CREATION, QUERY WRITING, ER DIAGRAM PL/SQL
MD. ABDUR RAHMAN FOYSAL	20-42742-1	25%	DATA INSERTION, RELATIONAL ALGEBRA, ACTIVITY DIAGRAM, PROJECT RPOPOSAL PL/SQL
MD.YOUSUF ALI SHADHIN	20-42783-1	25%	NORMALIZAION, SCHEMA DIAGRAM, UI DESIGN, USE CASE DIAGRAM DATABASE CONNECTION
TANJEMUL HOQUE TUHIN	19-41020-2	13%	DIAGRAM
PURAVI DEBNATH NITU	19-41371-3	12%	DIAGRAM

COURSE TITLE: ADVANCE DATABASE MANAGEMENT SYSTEM

SECTION: B

Contents

Introduction:

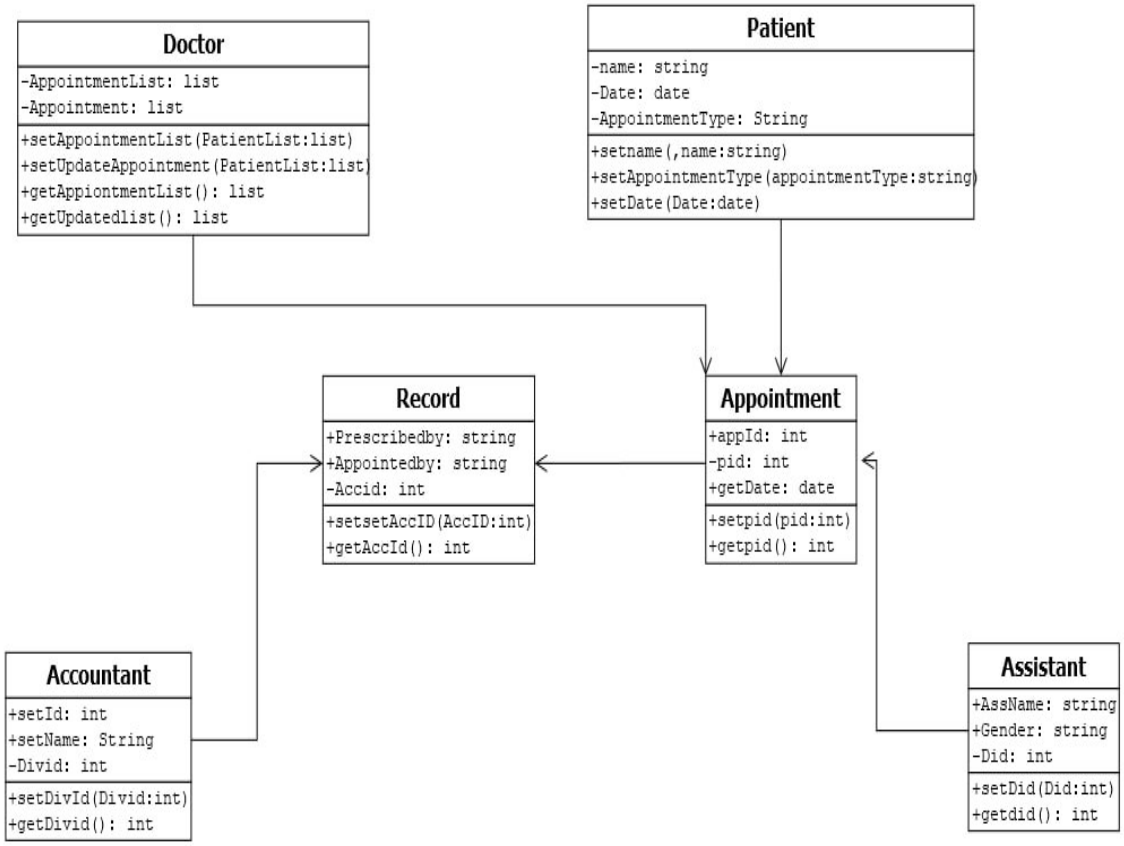
The management of healthcare systems requires efficient and accurate management of patient appointments, medical records, and healthcare providers. The database management system discussed in this case study focuses on the efficient handling of information related to a "Doctor Appointment Management System". The system allows patients to book appointments with doctors, while an assistant deals with managing the appointments. The system also keeps track of patient information, such as their blood group, phone numbers, and gender, and doctor information, such as their hire date, salary, and degrees. Additionally, the system maintains medical records, identifies patients prescribed by doctors, and manages payment to healthcare providers. The system's comprehensive approach ensures that all aspects of patient care and management are seamlessly integrated, resulting in a more efficient and effective healthcare system.

Project proposal:

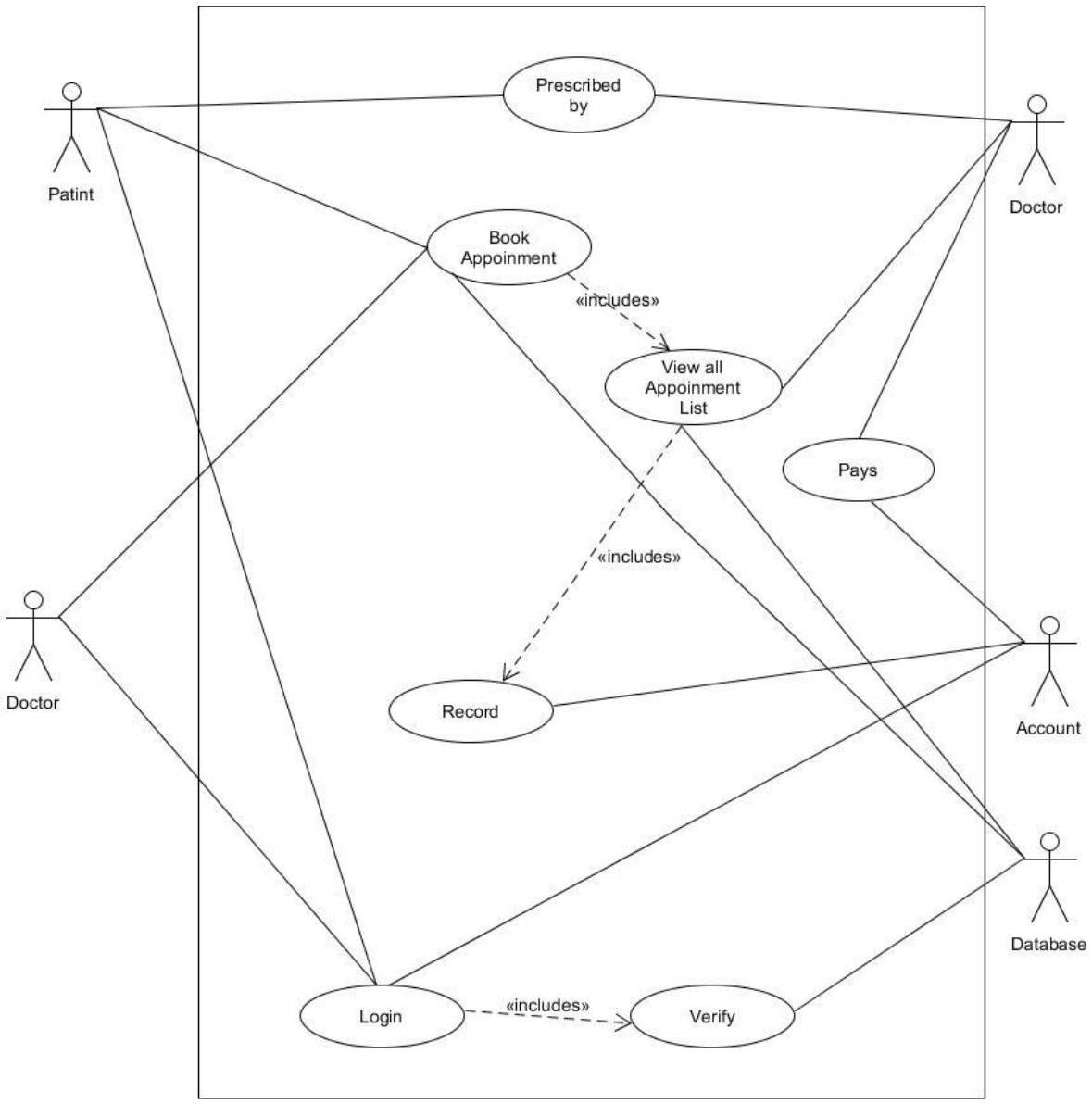
The proposed Doctor Appointment Management System aims to provide an efficient platform for patients to book appointments, doctors to manage appointments, and accountants to maintain financial records. The system includes modules for patient registration, appointment management, prescription and medical records, accountant management, and prioritizes security measures. The system will use MySQL, HTML, CSS and is scalable for future expansion.

Diagram:

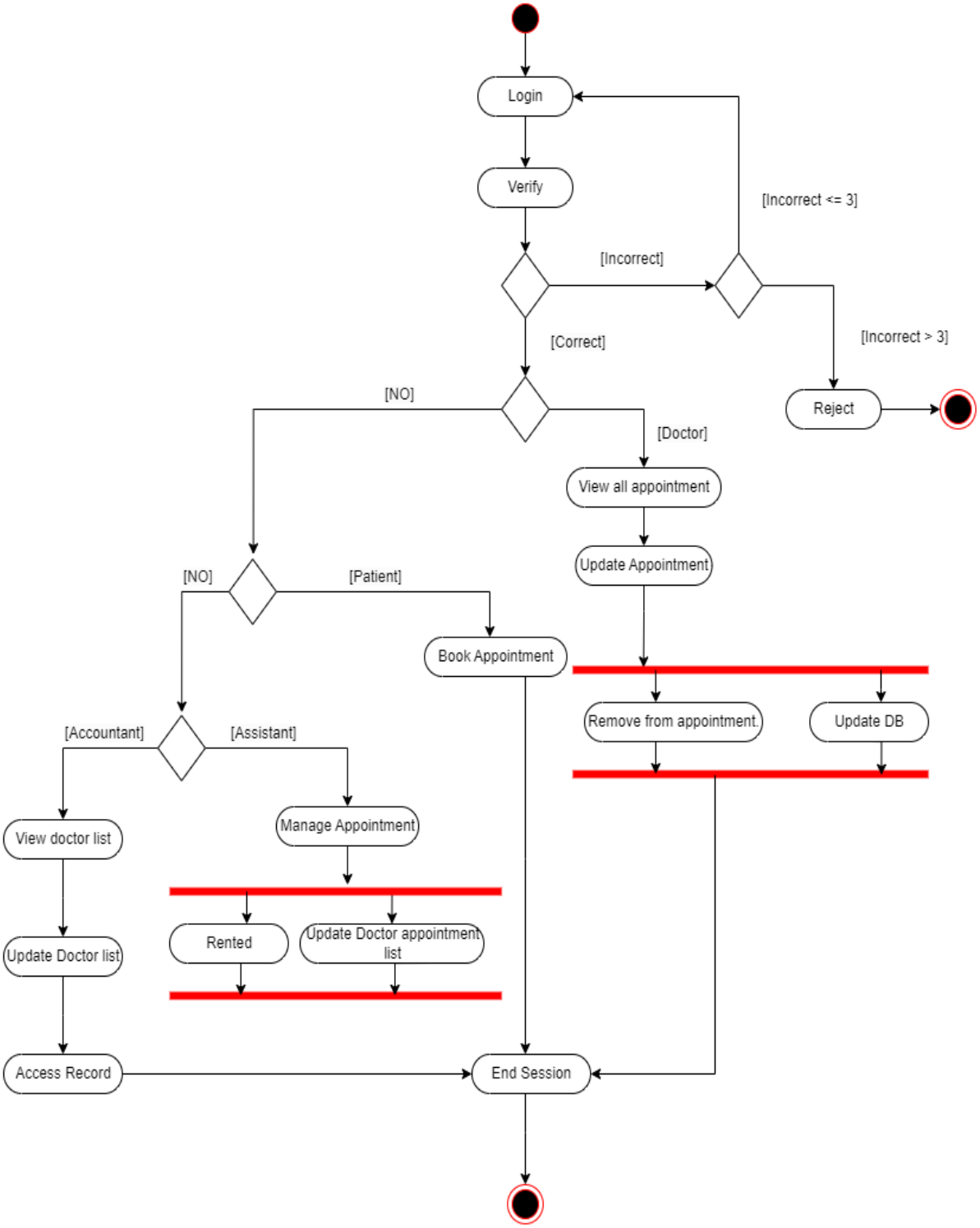
Class Diagram:



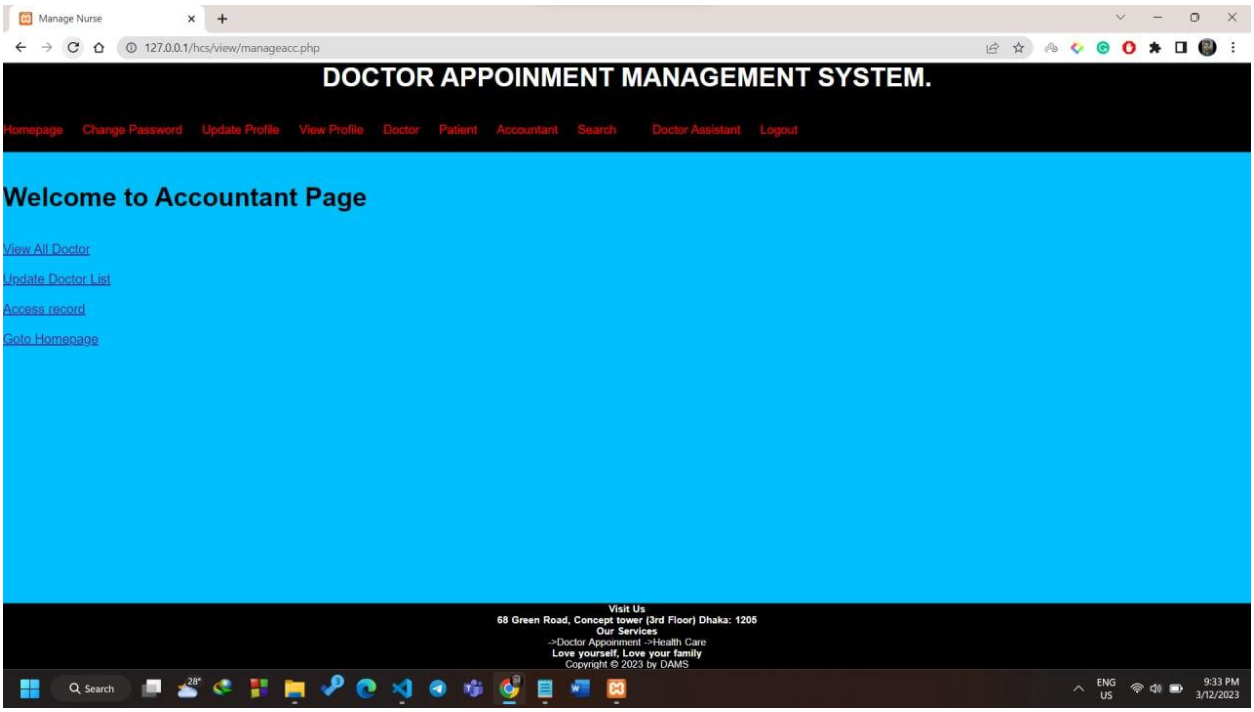
User Case Diagram:

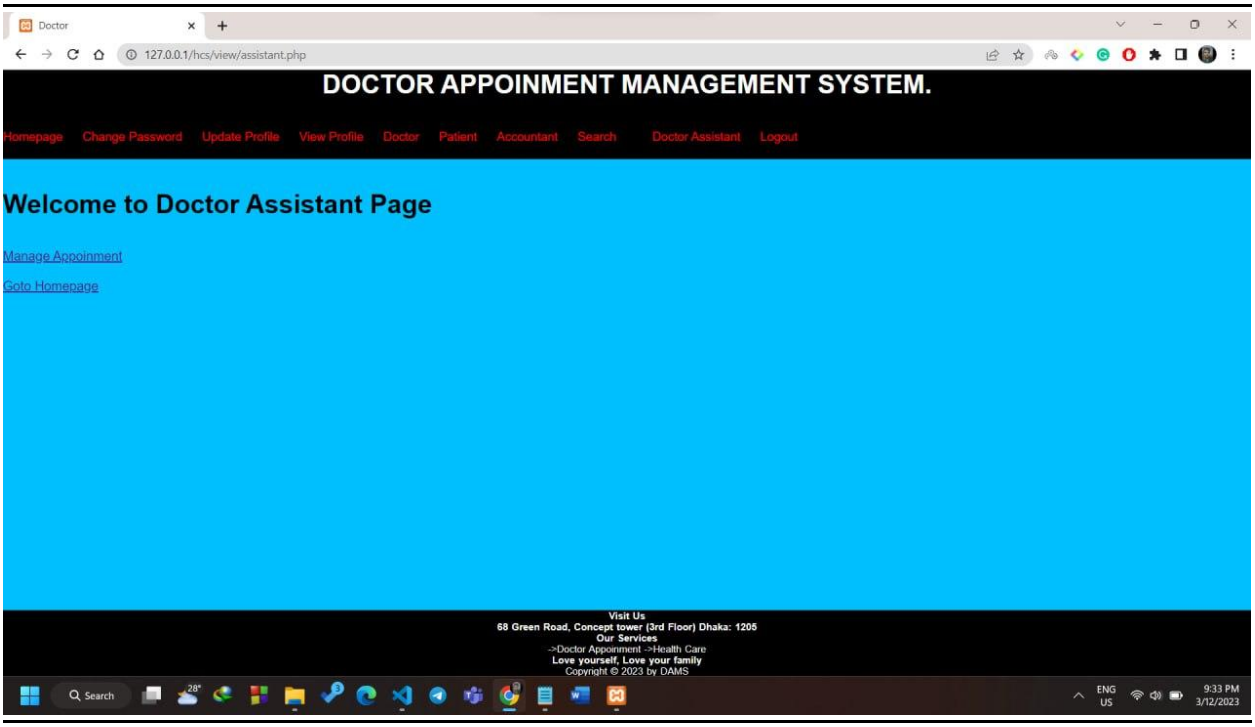


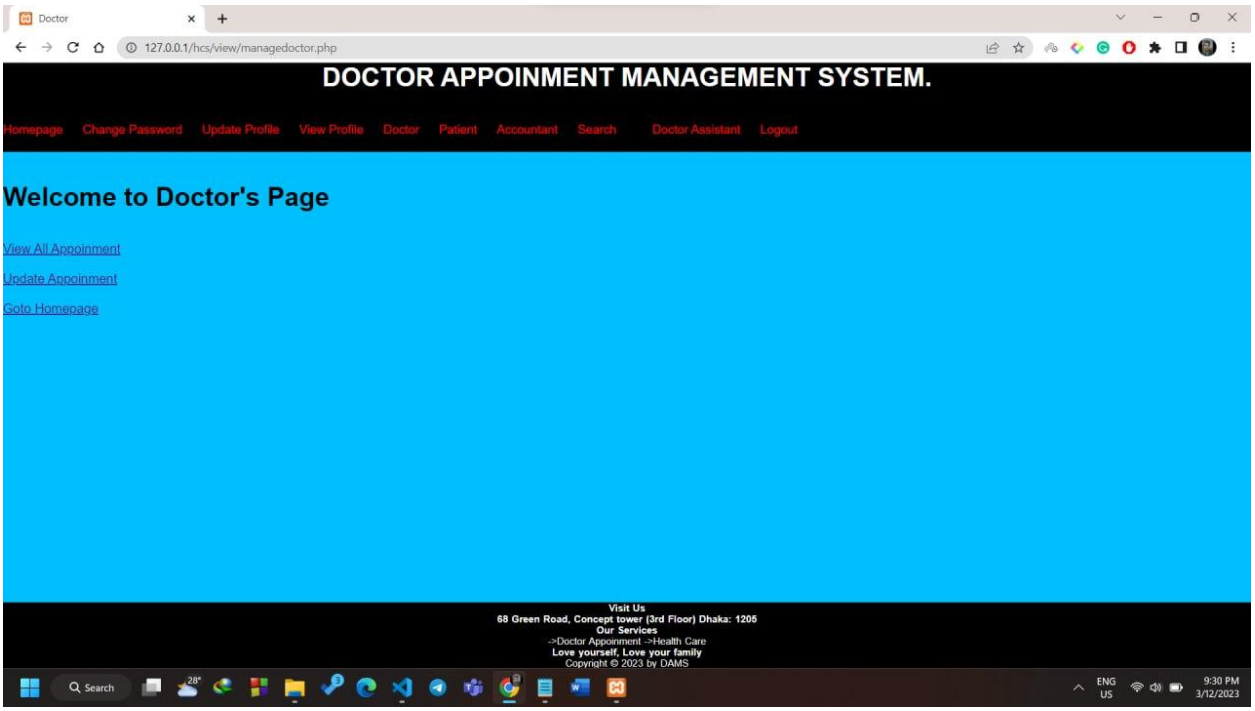
Activity Diagram:



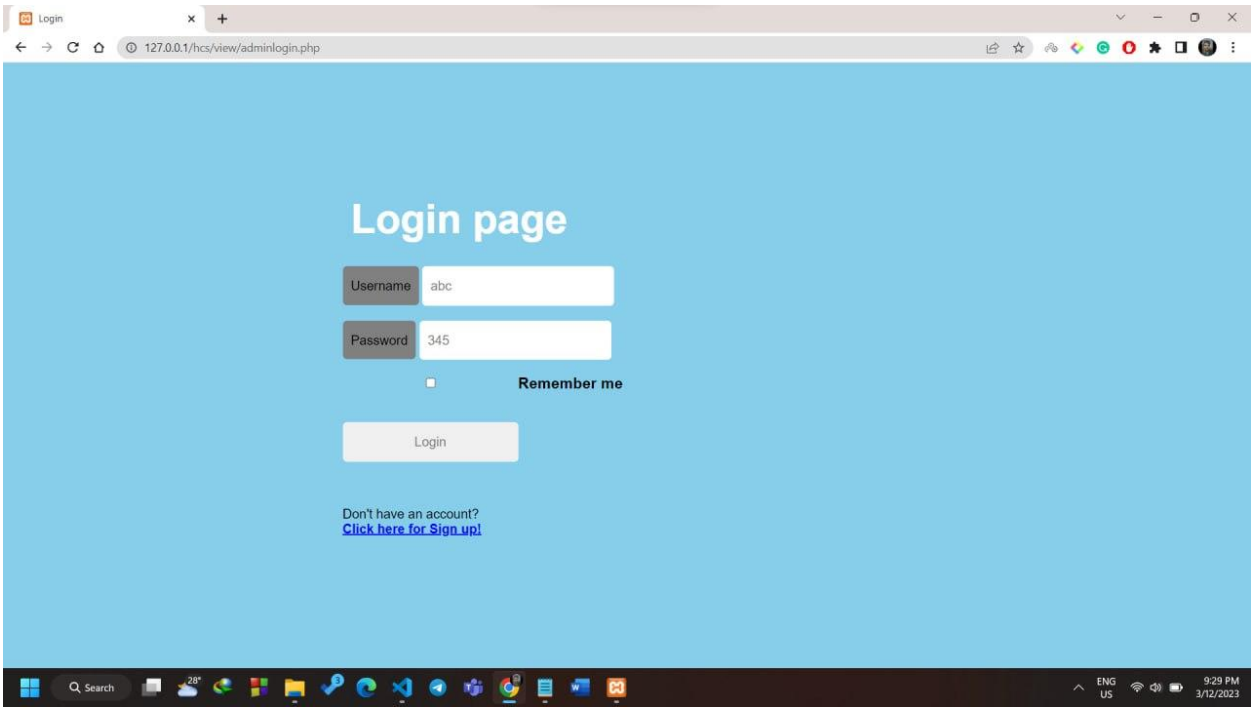
User Interface:






Doctor
127.0.0.1/hcs/view/managedoctor.php
DOCTOR APPOINMENT MANAGEMENT SYSTEM.
Homepage Change Password Update Profile View Profile Doctor Patient Accountant Search Doctor Assistant Logout
Welcome to Doctor's Page
View All Appoinment
Update Appoinment
Goto Homepage
Visit Us
68 Green Road, Concept tower (3rd Floor) Dhaka: 1205
Our Services
->Doctor Appoinment ->Health Care
Love yourself, Love your family
Copyright © 2023 by DAMS
Search
ENG US
9:30 PM 3/12/2023


Admin DashBoard
127.0.0.1/hcs/view/admindashboard.php
DOCTOR APPOINMENT MANAGEMENT SYSTEM.
Homepage Change Password Update Profile View Profile Doctor Patient Accountant Search Doctor Assistant Logout
Visit Us
68 Green Road, Concept tower (3rd Floor) Dhaka: 1205
Our Services
->Doctor Appoinment ->Health Care
Love yourself, Love your family
Copyright © 2023 by DAMS
Search
ENG US
9:29 PM 3/12/2023

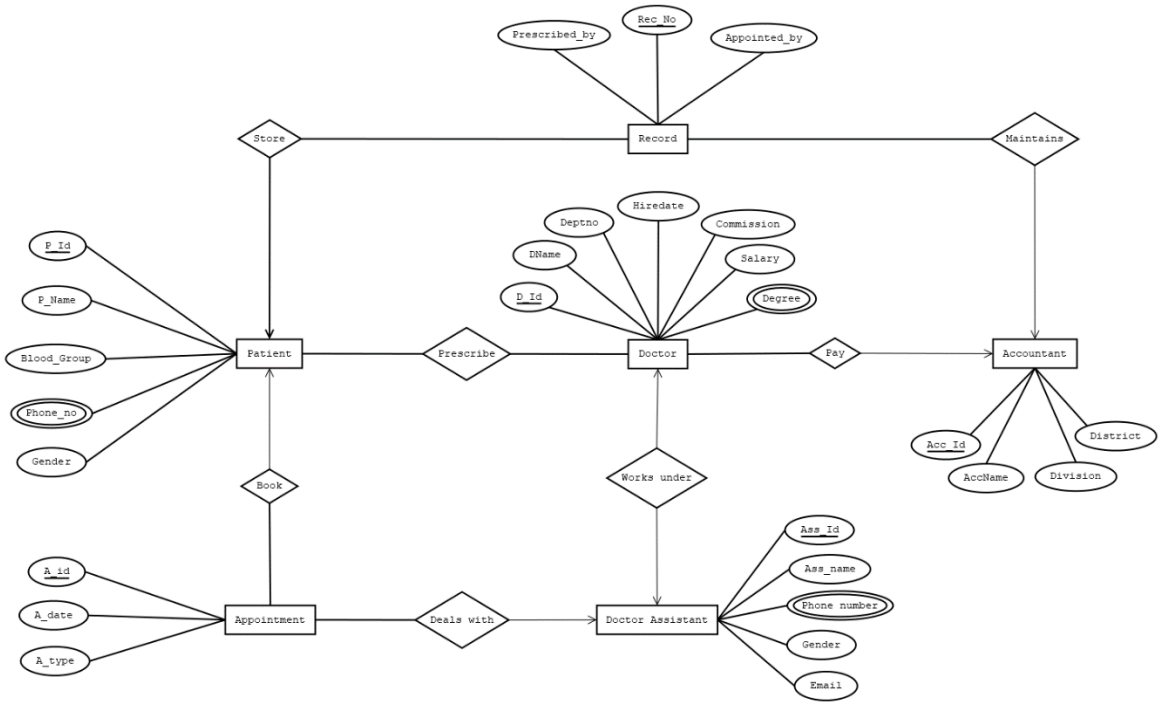


Scenario Description:

In this database management system, we can handle the information of a “DOCTOR APPOINTMENT MANAGEMENT SYSTEM”. In this system, Patient can book appointment. A patient can book many appointments. A patient is identified by P_id. A patient also has Pname,

Blood_Group, Phone_no, Gender. A patient can have multiple phone numbers. An appointment is identified by appointment_id. It also has appointment_date, appointment_type. An assistant deal with appointments. An assistant has Ass_id, Ass_name, gender, phone_number, email. An assistant works under a doctor. A doctor is identified by D_id, and have Dname, Hiredate, Salary, Degree, Department, Commission. A doctor can have multiple degrees. Patient prescribed by doctor. A patient can prescribe by many doctors. A doctor can prescribe many patients. An accountant pays multiple doctors. An accountant has unique Acc_id, and have Accname, Division, District. Accountant maintains records. One patient can have many records. A record is identified by Rec_no and have prescribed_by, Appointed_by.

ER diagram:



Normalization:

Prescribe- (D_id, DName, Hiredate, Degree, Salary, Deptno, Comm, P_id, PName, Blood_Group, Phone_no, Gender)

1NF: Degree and phone_no are multivalued attributes.

2NF: D_id, DName, Hiredate, Degree, Salary, Deptno, Comm

P_id, PName, Blood_Group, Phone_no, Gender

3NF: No, transitive dependency found

D_id, DName, Hiredate, Degree, Salary, Deptno, Comm

P_id, PName, Blood_Group, Phone_no,Gender

Tables from Prescribe,

1. D_id, DName, Hiredate, Salary, Deptno, Comm
2. D_id, Degree -> Composite pk
3. P_id, PName, Blood_Group, Gender
4. P_id,Phone_no -> Composite pk
5. DP_id, D_id, P_id

Book-(P_id, PName, Blood_Group, Phone_no, Gender, A_id, A_date, A_type)

1NF: phone_no are multivalued attributes.

2NF: P_id, PName, Blood_Group, Phone_no, Gender

A_id, A_date, A_type

3NF: No, transitive dependency found

P_id, PName, Blood_Group, Phone_no, Gender

A_id, A_date, A_type

Table from Book

6. P_id, PName, Blood_Group, Phone_no, Gender
7. P_id, Phone_no -> Composite pk
8. A_id, A_date, A_type, P_id

Deals with-(A_id, A_date, A_type, Ass_id, phone number, gender, email)

1NF: Phone_no is a multivalued attribute.

2NF: A_id, A_date, A_type

Ass_id, phone number, gender, email

3NF: No transitive dependency found

A_id, A_date, A_type

Ass_id, phone number, gender, email

Tables from Deals with,

9. A_id, A_date, A_type, Ass_id
10. Ass_id, phone number, gender, email

11. Ass_id, phone number -> Composite pk

Works Under – (Ass_id, phone number, gender, email, D_id, DName, Hiredate, Degree, Salary, Deptno, Comm)

1NF: Degree and phone_no are multivalued attributes.

2NF: Ass_id, phone number, gender, email

D_id, DName, Hiredate, Degree, Salary , Deptno, Comm

3NF: No transitive dependency found

Ass_id, phone number, gender, email

D_id, DName, Hiredate, Degree, Salary , Deptno, Comm

Tables from works under,

12. Ass_id, gender, email, D_id
13. Ass_id, phone number -> Composite
14. D_id, DName, Hiredate, Salary, deptno, Comm
15. D_id, Degree -> Composite pk

Pays- (Acc_id, Accname, Division, District, D_id, DName, Hiredate, Degree, Salary, Deptno, Comm)

1NF: Degree is a multivalued attribute.

2NF: Acc_id, Accname, Division, District,

D_id, DName, Hiredate, Degree, Salary, Deptno, Comm

3NF: Acc_id, Accname

Div_id, Division, District

D_id, DName, Hiredate, Degree, Salary, Deptno, Comm

Tables from Payment

16. Acc_id, Accname, Div_id
17. Div_id, Division, District,
18. D_id, DName, Hiredate, Salary, Deptno, Comm Acc_id
19. D_id, Degree ->composite pk

Maintain- (Acc_id, Accname, Division, District, Rec_no, Prescribed by, Appointed_by)

1NF: No multivalued attribute.

2NF: Acc_id, Accname, Division, District,

Rec_no, Appointed_by, Prescribed_by

3NF: Acc_id, Accname

Dis_id, Division, District,

Rec_no, Appointed_by, Prescribed_by

Tables from maintains

20. Acc_id, Accname, Dis_id
21. Dis_id, Division, District
22. Rec_no, Prescribed_by, Appointed_by, Acc_id

Store- (Rec_no, Pescribedby, Appointed_by, P_id, PName, Blood_Group, Phone_no, Gender)

1NF: Phone_no is a multivalued attribute

2NF: Rec_no, Appointed_by, Pescribedby

P_id, PName, Blood_Group, Phone_no, Gender

3NF: No transitive dependency found

Rec_no, Appointed_by, Pescribedby

P_id, PName, Blood_Group, Phone_no, Gender

Tables from Store

23. Rec_no, Prescribed_by, Appointed_by, P_id
24. P_id, PName, Blood_Group, Gender
25. P_id,Phone_no ->composite pk

1. ~~D_id, DName, Hiredate, Salary, dpetno, comm~~
2. D_id, Degree -> Composite pk
3. P_id, PName, Blood_Group, Gender
4. P_id,Phone_no -> Composite pk
5. DP_id, D_id, P_id
6. ~~P_id, PName, Blood_Group, Gender~~
7. ~~P_id, Phone_no -> Composite pk~~
8. A_id, A_date, A_type, P_id
9. A_id, A_date, A_type, Ass_id
10. ~~Ass_id, name, gender, email~~
11. Ass_id, phone number -> Composite pk
12. Ass_id, name, gender, email, D_id
13. ~~Ass_id, phone number -> Composite~~
14. ~~D_id, DName, Hiredate, Salary, depnto, comm~~
15. ~~D_id, Degree -> Composite pk~~
16. Acc_id, Accname, Div_id
17. Div_id, Division, District
18. D_id, DName, Hiredate, Salary, deptno, comm, Acc_id
19. ~~D_id, Degree ->composite pk~~
20. ~~Acc_id, Accname, Div_id~~
21. ~~Div_id, Division, District~~
22. Rec_no, Pescribedby, Appointed_by, Acc_id
23. Rec_no, Pescribedby, Appointed_by, P_id
24. ~~P_id, PName, Blood_Group, Gender~~
25. ~~P_id,Phone_no ->composite pk~~

Tables after normalization:

1. D_id, DName, Hiredate, Salary, Deptno, Comm, Acc_id --Doctor
2. D_id, Degree -> Composite pk –DoctorDegree
3. P_id, PName, Blood_Group, Gender –Patient
4. P_id,Phone_no -> Composite pk –patientContact
5. DP_id, D_id, P_id –DoctorPatient
6. A_id, A_date, A_type, P_id, Ass_id –Appointment
7. Ass_id, name, gender, email, D_id –Assistant
8. Ass_id, phone number -> Composite pk –AssitantContact
9. Acc_id, Accname, Div_id –Accountant
10. Div_id, Division, District –AccountantLoc
11. Rec_no, Pescribedby, Appointed_by, P_id, Acc_id –Record

Schema Diagram:

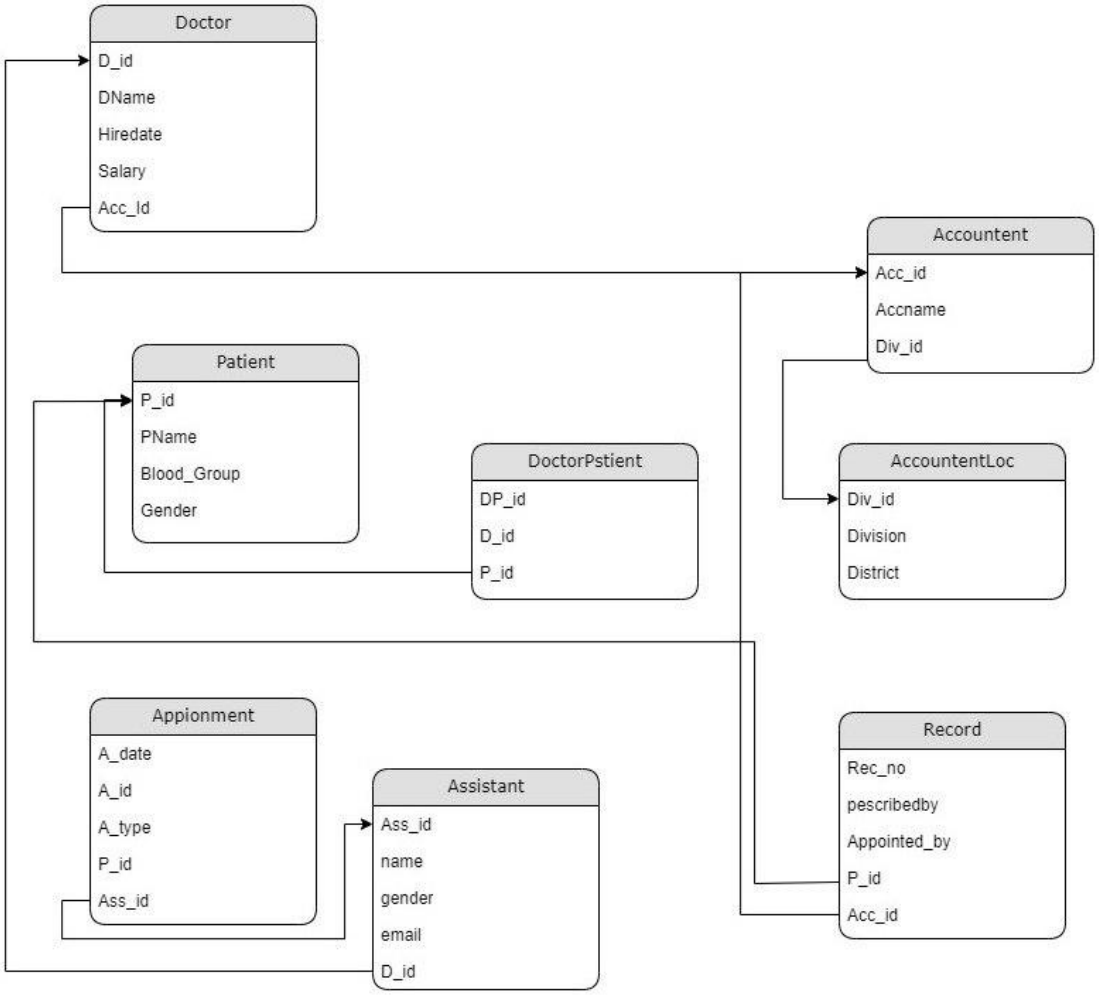
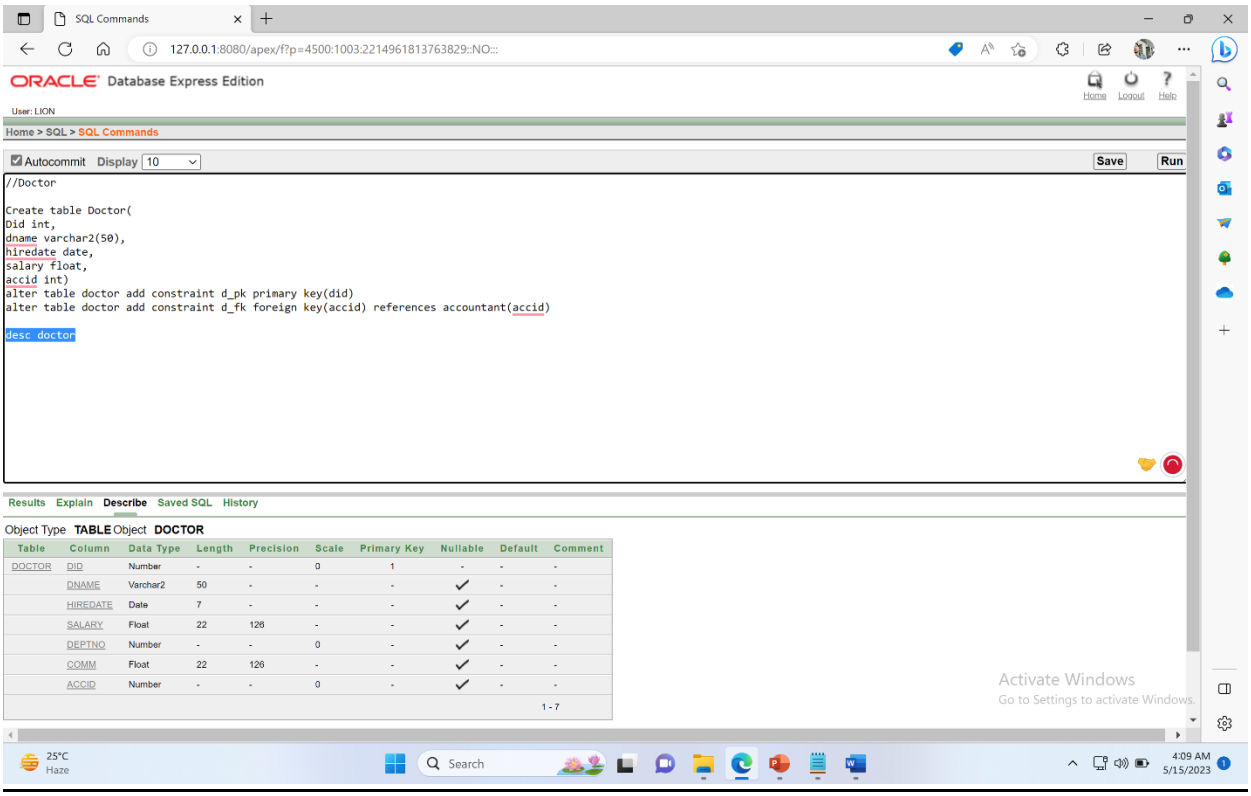


Table Creation:

Doctor Table:



Create table Doctor(

Did int,

dname varchar2(50),

hiredate date,

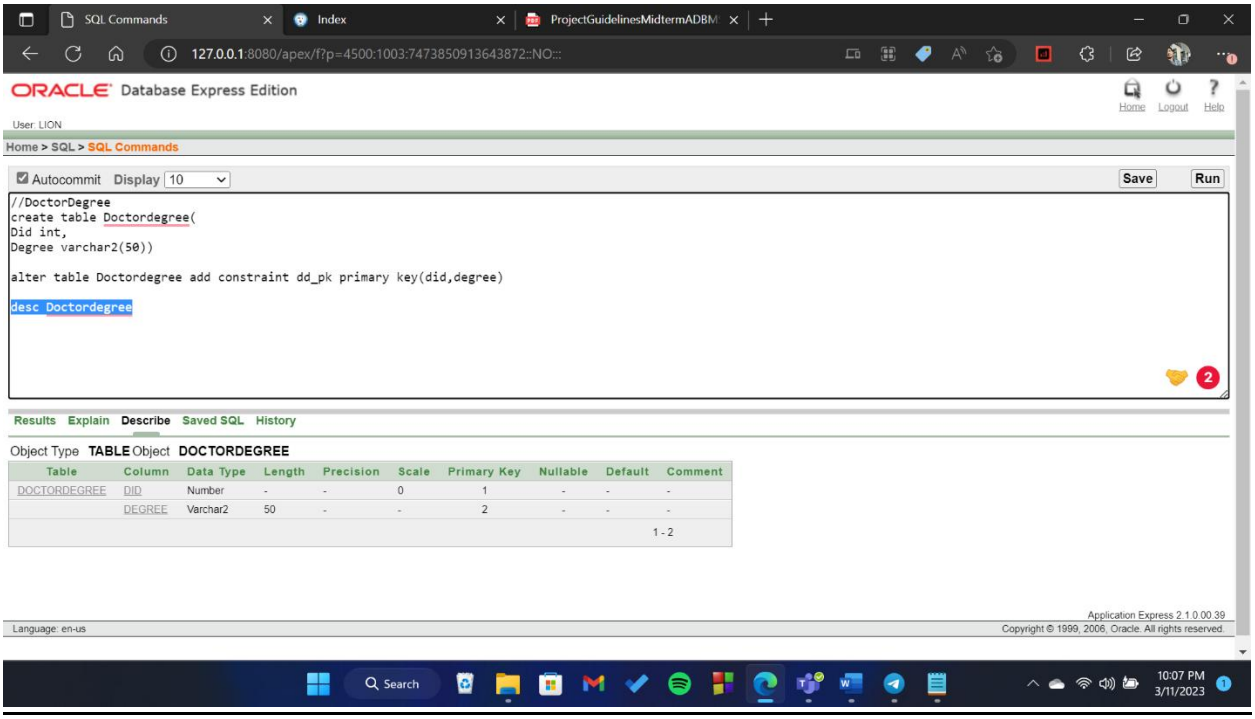
salary float,

accid int)

alter table doctor add constraint d_pk primary key(did)

alter table doctor add constraint d_fk foreign key(accid) references accountant(accid)

DoctorDegree Table:



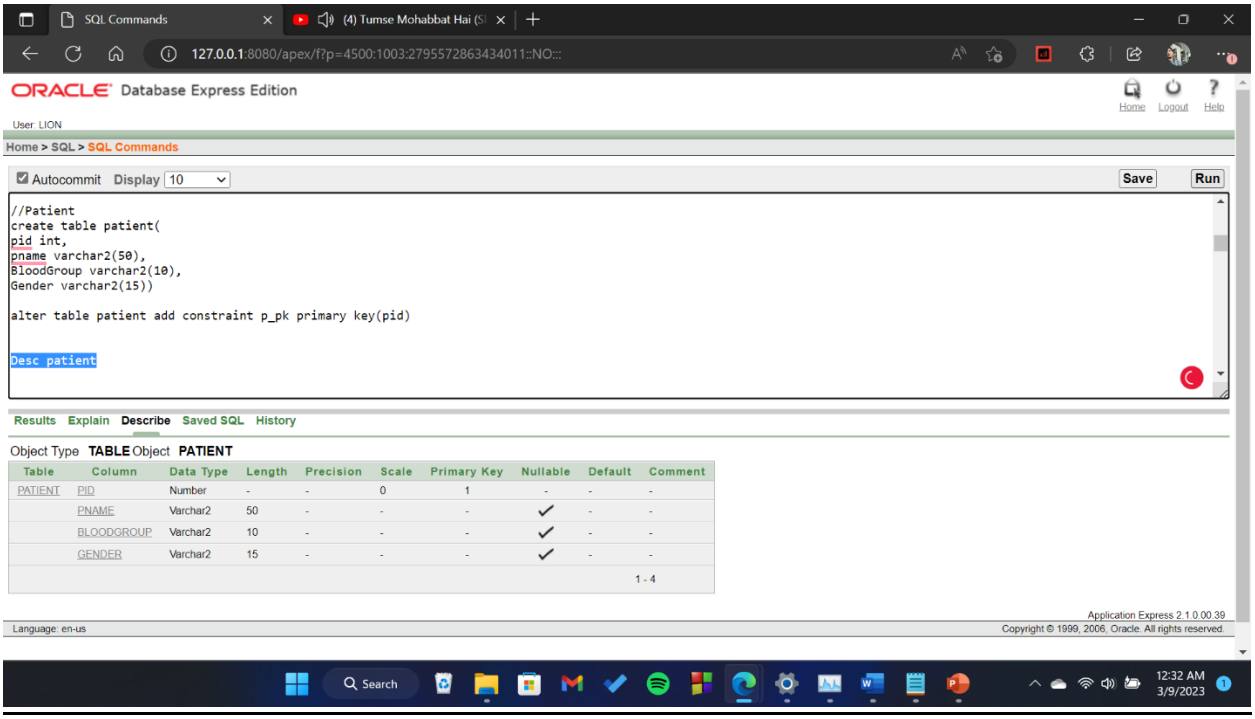
create table Doctordegree(

Did int,

Degree varchar2(50))

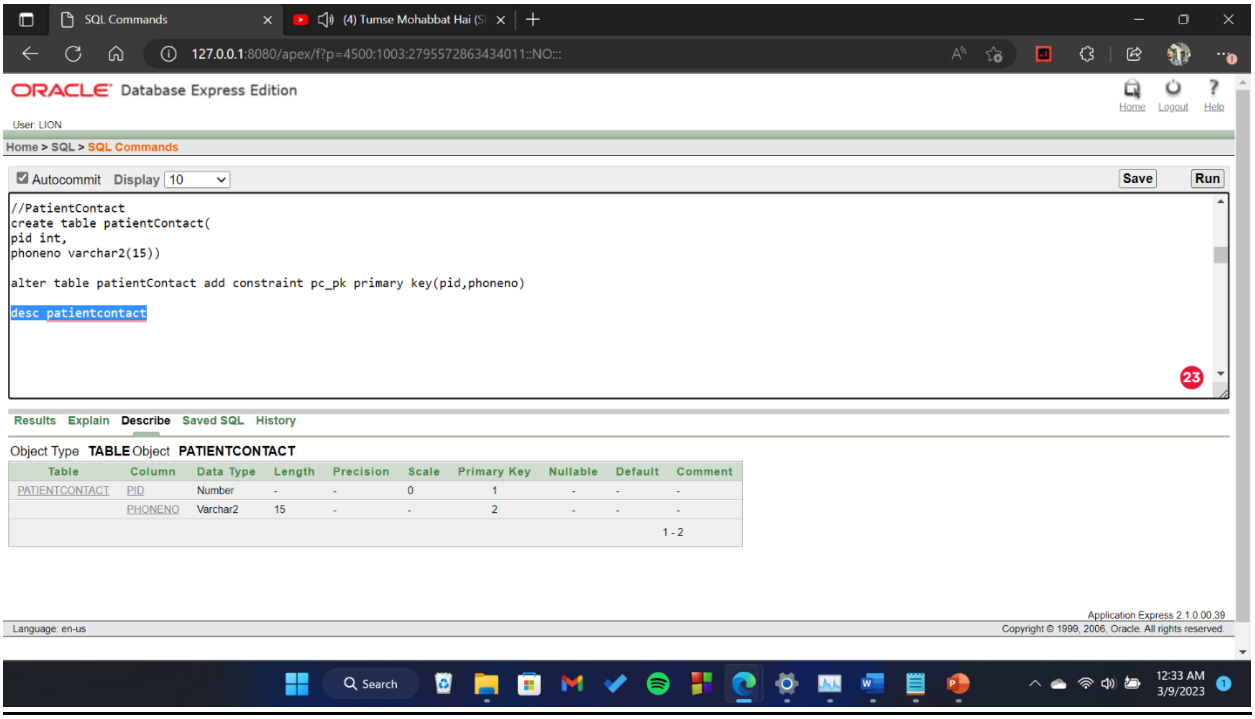
alter table Doctordegree add constraint dd_pk primary key(did,degree)

Patient Table:




```
create table patient(
pid int,
pname varchar2(50),
BloodGroup varchar2(10),
Gender varchar2(15))
alter table patient add constraint p_pk primary key(pid)
```

PatientContact:



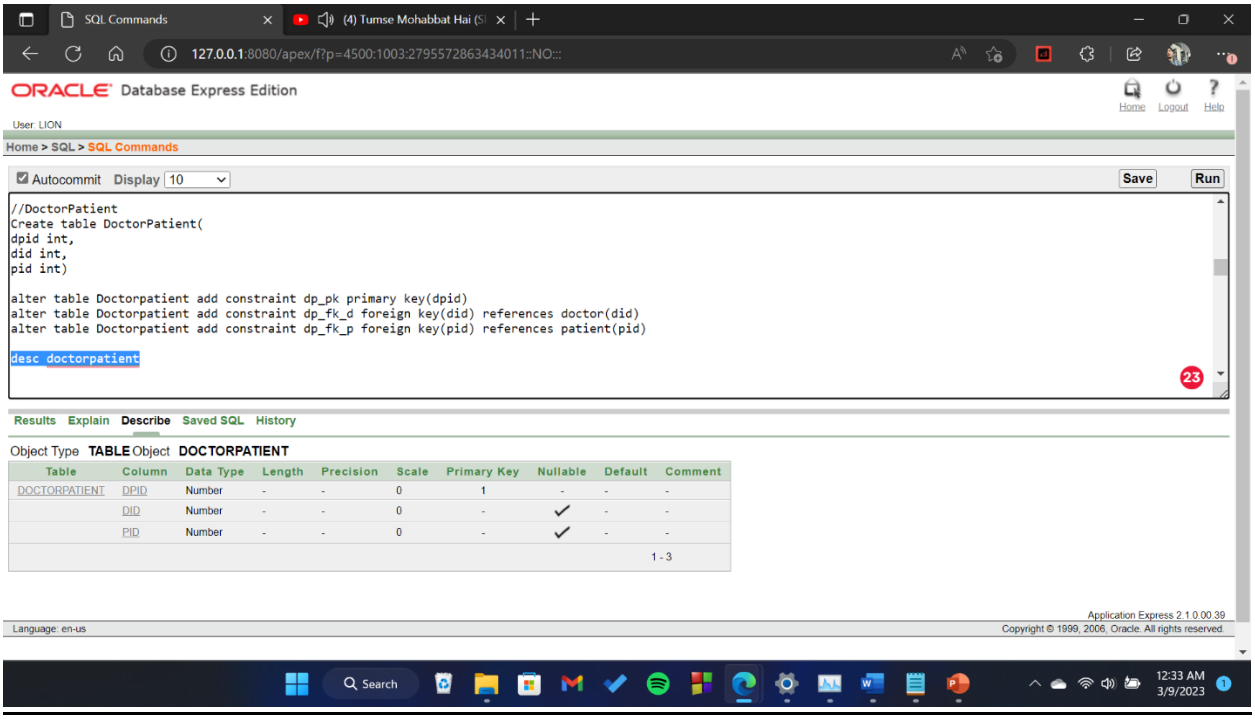
create table patientContact(

pid int,

phoneno varchar2(15))

alter table patientContact add constraint pc_pk primary key(pid,phoneno)

Doctor Pactient Table:



create table DoctorPatient(

dpid int,

did int,

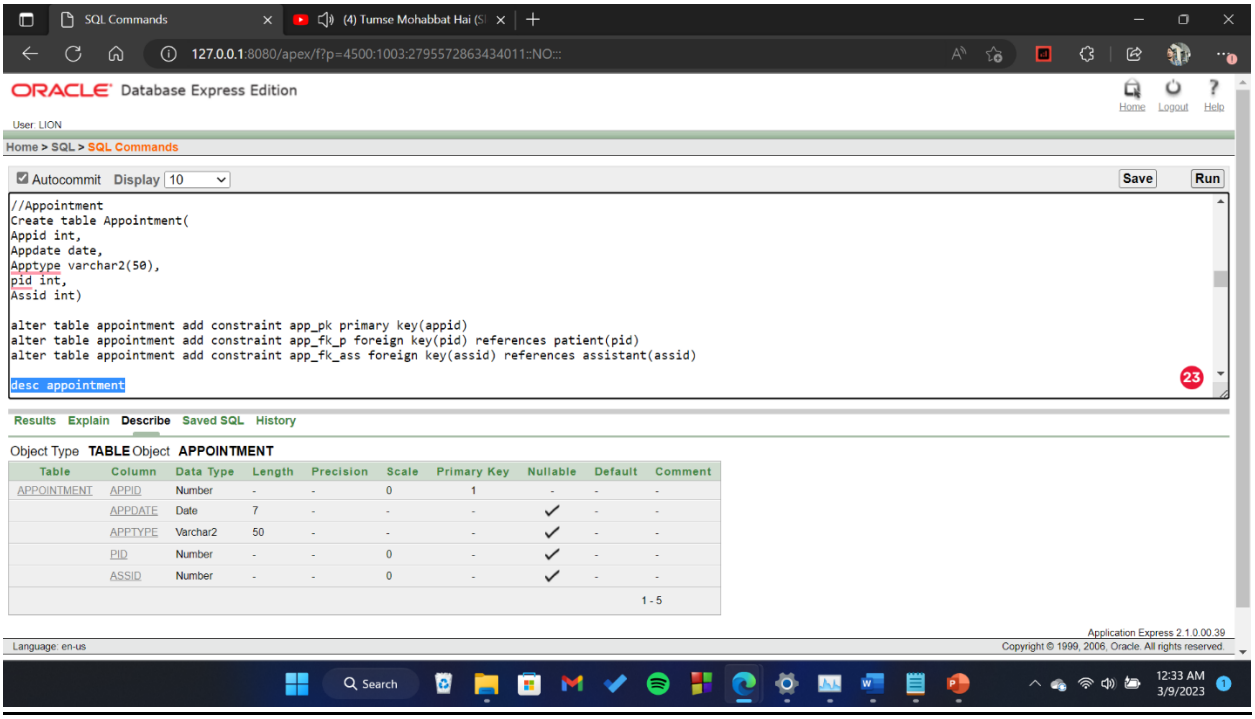
pid int)

alter table Doctorpatient add constraint dp_pk primary key(dpid)

alter table Doctorpatient add constraint dp_fk_d foreign key(did) references doctor(did)

alter table Doctorpatient add constraint dp_fk_p foreign key(pid) references patient(pid)

Appoinment Table:



Create table Appointment(

Appid int,

Appdate date,

Apptype varchar2(50),

pid int,

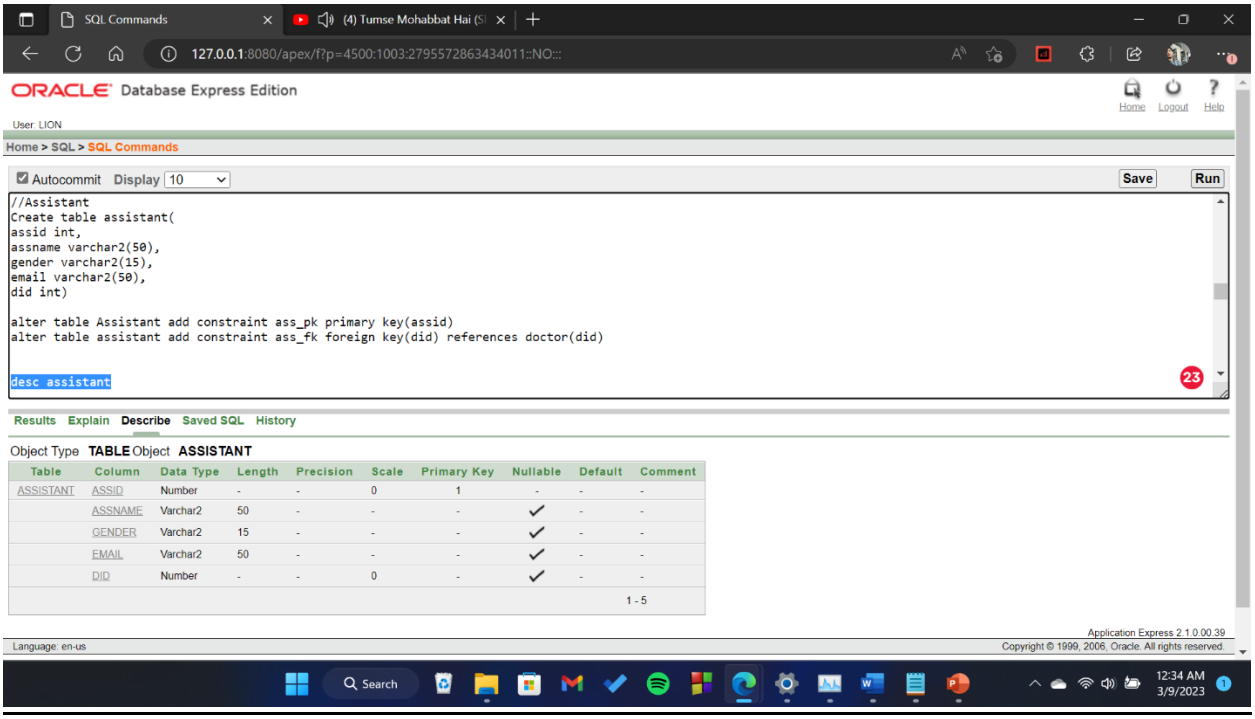
Assid int)

alter table appointment add constraint app_pk primary key(appid)

alter table appointment add constraint app_fk_p foreign key(pid) references patient(pid)

alter table appointment add constraint app_fk_ass foreign key(assid) references assistant(assid)

Assistant Table:



Create table assistant(

assid int,

assname varchar2(50),

gender varchar2(15),

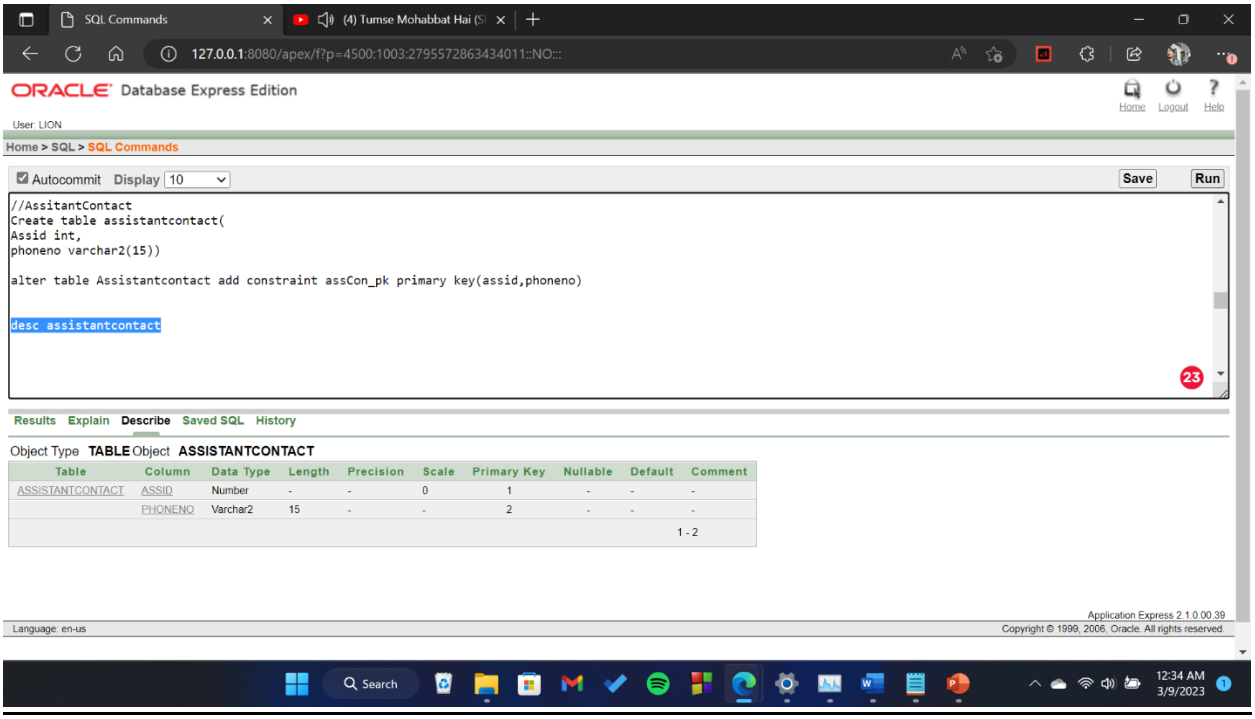
email varchar2(50),

did int)

alter table Assistant add constraint ass_pk primary key(assid)

alter table assistant add constraint ass_fk foreign key(did) references doctor(did)

AssitantContact Table:



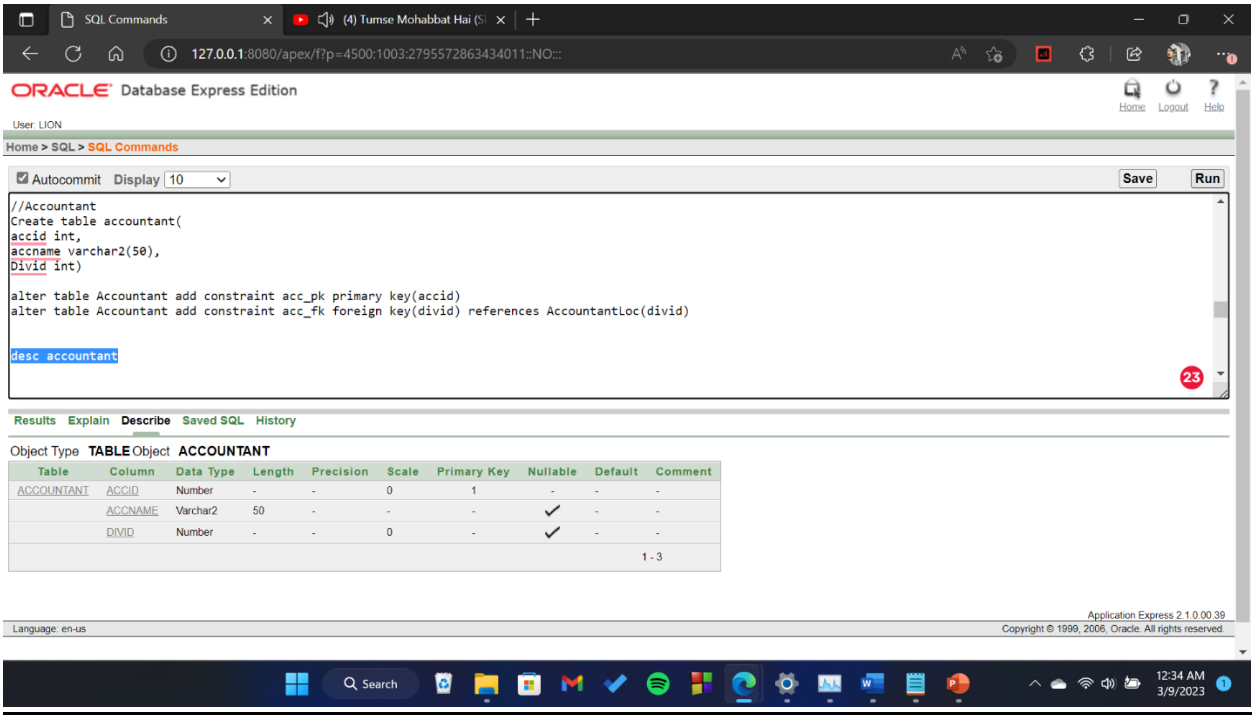
Create table assistantcontact(

Assid int,

phoneno varchar2(15))

alter table Assistantcontact add constraint assCon_pk primary key(assid,phoneno)

Accountant Table:



Create table accountant(

accid int,

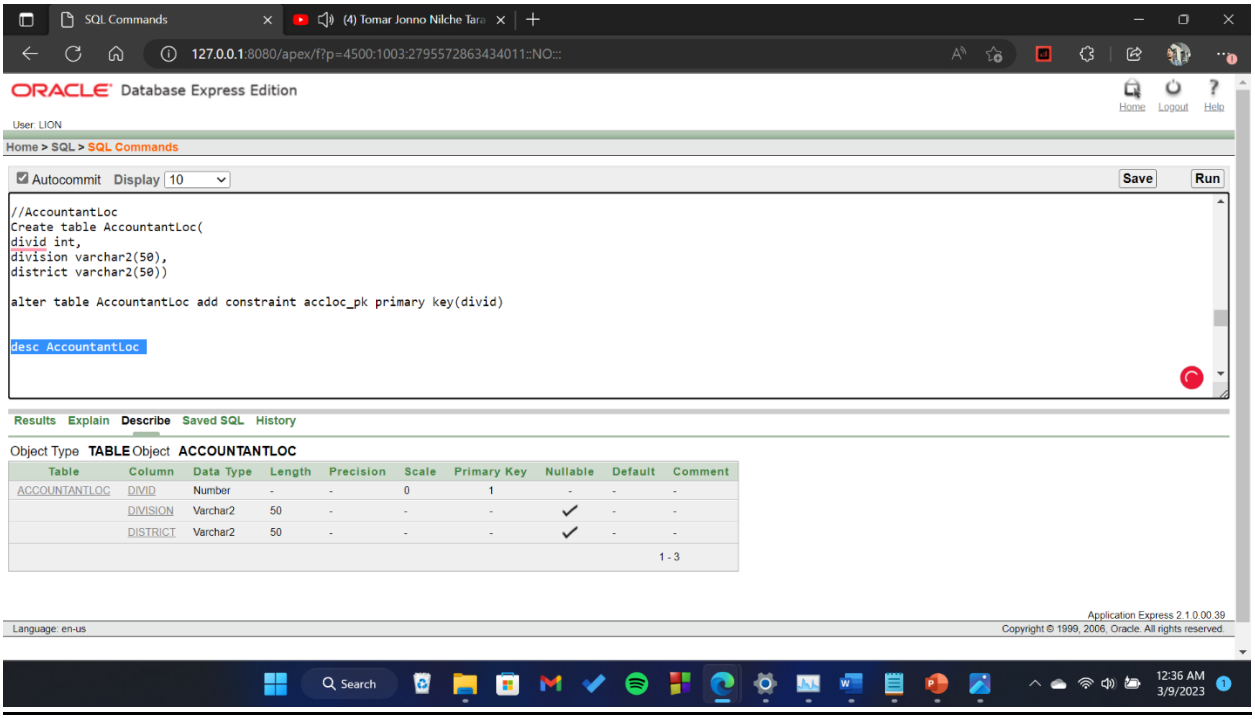
accname varchar2(50),

Divid int)

alter table Accountant add constraint acc_pk primary key(accid)

alter table Accountant add constraint acc_fk foreign key(divid) references AccountantLoc(divid)

AccountantLoc Table:



Create table AccountantLoc(

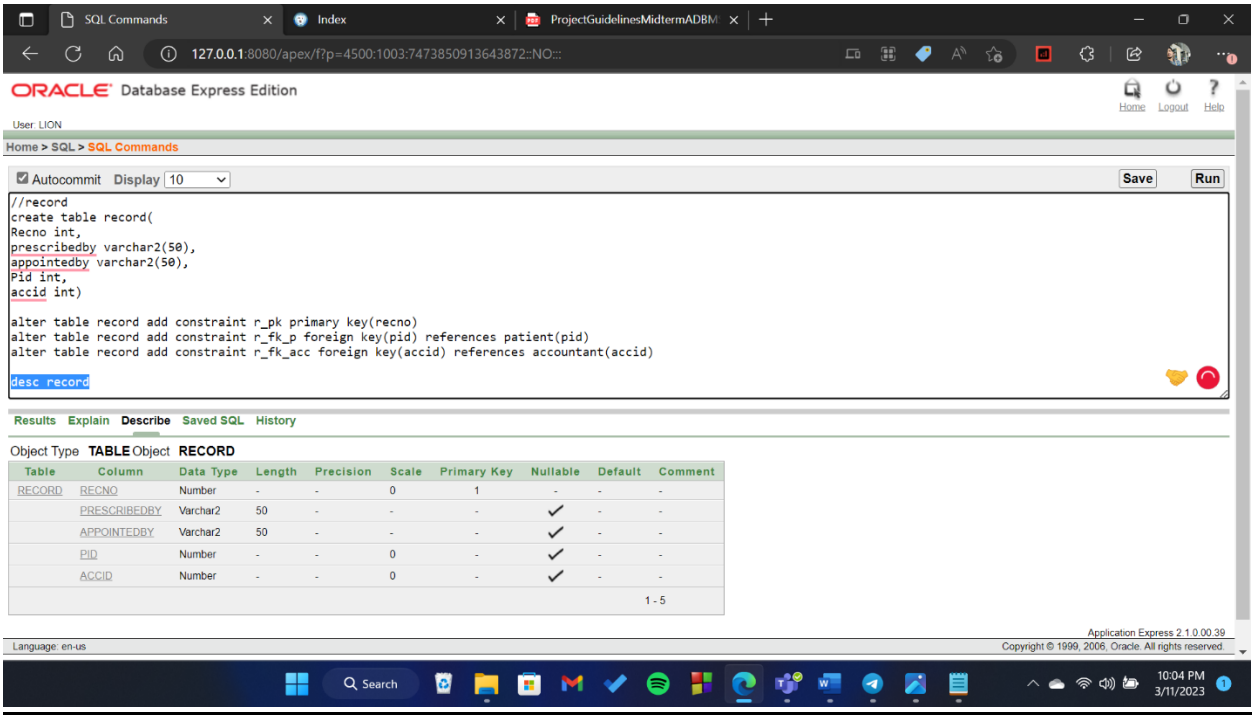
divid int,

division varchar2(50),

district varchar2(50))

alter table AccountantLoc add constraint accloc_pk primary key(divid)

Record Table:




```
create table record(
Recno int,
prescribedby varchar2(50),
appointedby varchar2(50),
Pid int,
accid int)

alter table record add constraint r_pk primary key(recno)
alter table record add constraint r_fk_p foreign key(pid) references patient(pid)
alter table record add constraint r_fk_acc foreign key(accid) references accountant(accid)
```

Queries For Sequences:

//Doctor(DOCSEQ_101)

```
create sequence docseq
increment by 1
start with 101
maxvalue 200
nocache
nocycle
```

//Patient(PATSEQ_5001)

```
create sequence patseq
increment by 1
start with 5001
maxvalue 6000
nocache
nocycle
```

//DoctorPatient(DPSEQ_6001)

```
create sequence DPseq
increment by 1
start with 6001
maxvalue 7000
nocache
nocycle
```

//Appointment(APPSEQ_7001)

```
create sequence APPseq
```

```
increment by 1
start with 7001
maxvalue 8000
nocache
nocycle

//Assistant(ASTSEQ_301)
create sequence ASTseq
increment by 1
start with 301
maxvalue 400
nocache
nocycle

//Accountant(ACCSEQ_1)
create sequence ACCseq
increment by 5
start with 5
maxvalue 100
nocache
nocycle

//AccountantLoc(ACCLSEQ_1201)
create sequence ACCLseq
increment by 1
start with 1201
maxvalue 1300
```

nocache

nocycle

//Record(RECSEQ_8001)

create sequence RECseq

increment by 1

start with 8001

maxvalue 9000

nocache

nocycle

Index Creation:

//doctor

Create index doctor_idx

on doctor(dname,salary)

//patient

Create index patient_idx

on patient(pname,bloodgroup)

//appointment

create index appointment_idx

on appointment(pid,appdate,apptype)

//assistant

Create index assinstant_idx

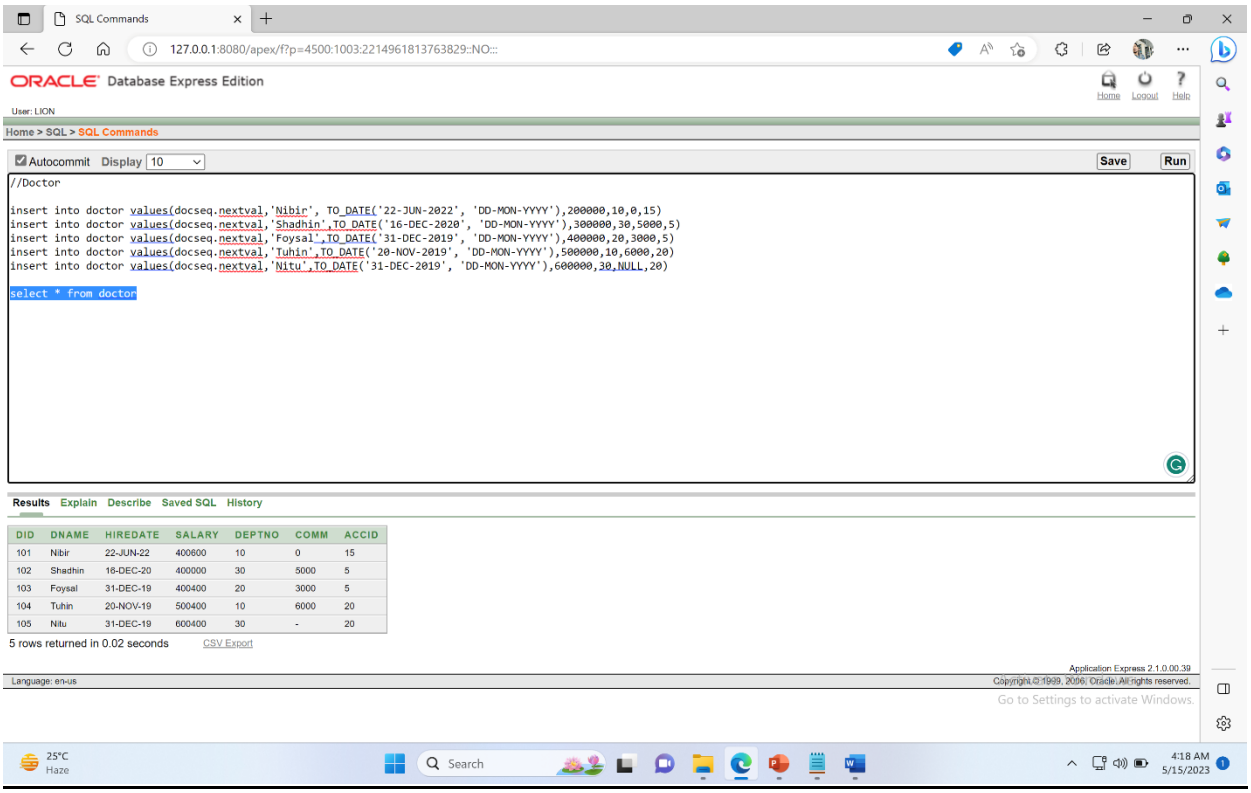
on assistant(assname,email)

//accountant

Create index accountant_idx

on accountant(accid,accname)

Data insertion:



//Doctor

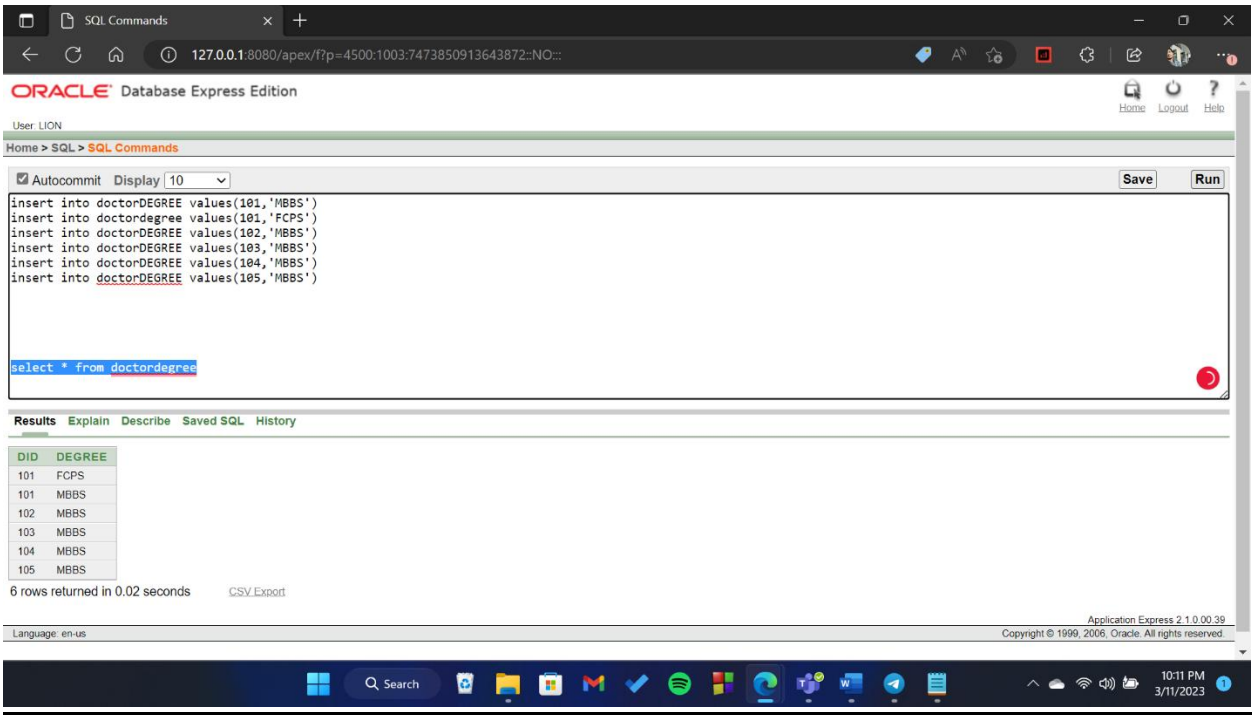
insert into doctor values(docseq.nextval,'Nibir', TO_DATE('22-JUN-2022', 'DD-MON-YYYY'),200000,10,0,15)

insert into doctor values(docseq.nextval,'Shadhin',TO_DATE('16-DEC-2020', 'DD-MON-YYYY'),300000,30,5000,5)

insert into doctor values(docseq.nextval,'Foysal',TO_DATE('31-DEC-2019', 'DD-MON-YYYY'),400000,20,3000,5)

insert into doctor values(docseq.nextval,'Tuhin',TO_DATE('20-NOV-2019', 'DD-MON-YYYY'),500000,10,6000,20)

insert into doctor values(docseq.nextval,'Nitu',TO_DATE('31-DEC-2019', 'DD-MON-YYYY'),600000,30,NULL,20)



//Doctordegree

insert into doctorDEGREE values(101,'MBBS')

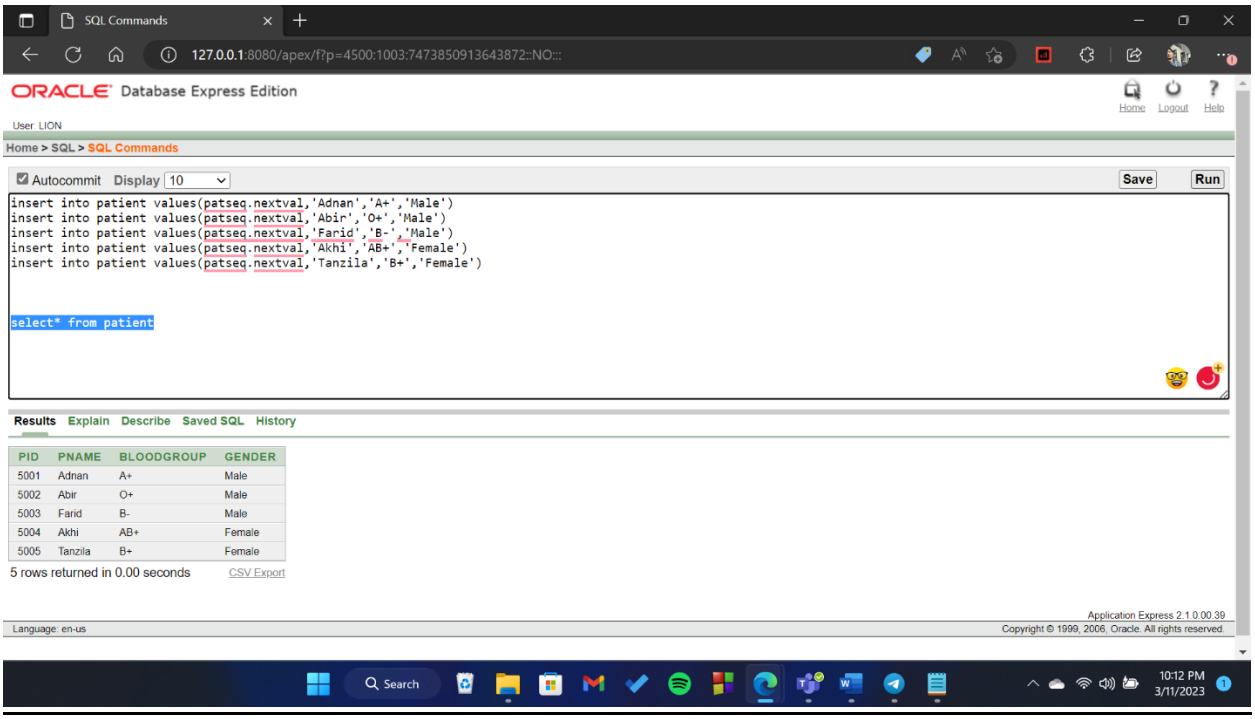
insert into doctordegree values(101,'FCPS')

insert into doctorDEGREE values(102,'MBBS')

insert into doctorDEGREE values(103,'MBBS')

insert into doctorDEGREE values(104,'MBBS')

insert into doctorDEGREE values(105,'MBBS')



//Patient

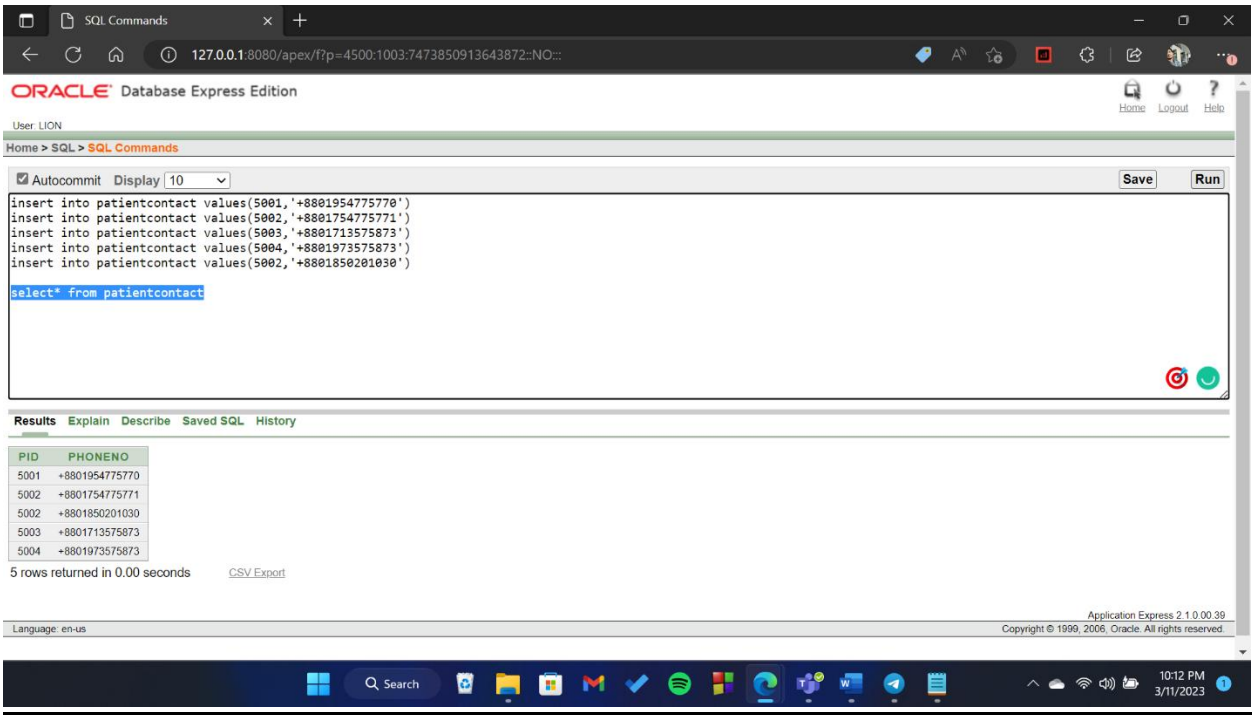
insert into patient values(patseq.nextval,'Adnan','A+','Male')

insert into patient values(patseq.nextval,'Abir','O+','Male')

insert into patient values(patseq.nextval,'Farid','B-','Male')

insert into patient values(patseq.nextval,'Akhi','AB+','Female')

insert into patient values(patseq.nextval,'Tanzila','B+','Female')



//PatientContact

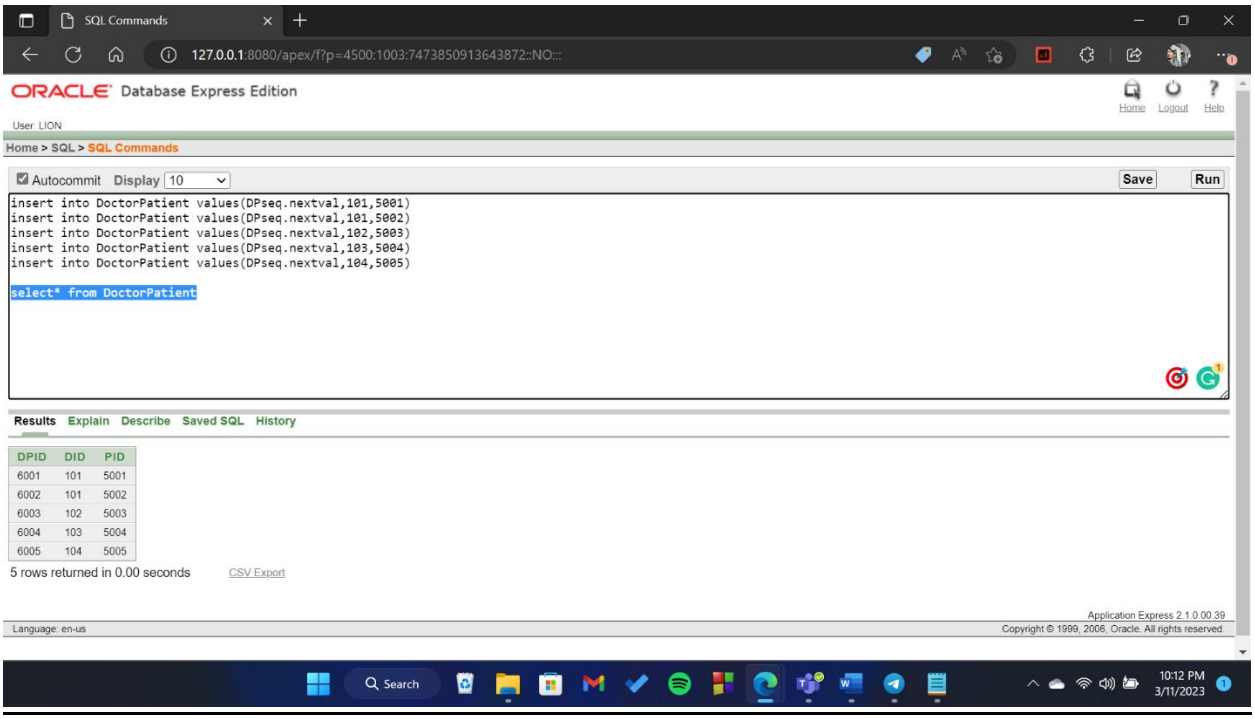
insert into patientcontact values(5001,'+8801954775770')

insert into patientcontact values(5002,'+8801754775771')

insert into patientcontact values(5003,'+8801713575873')

insert into patientcontact values(5004,'+8801973575873')

insert into patientcontact values(5002,'+8801850201030')



//DoctorPatient

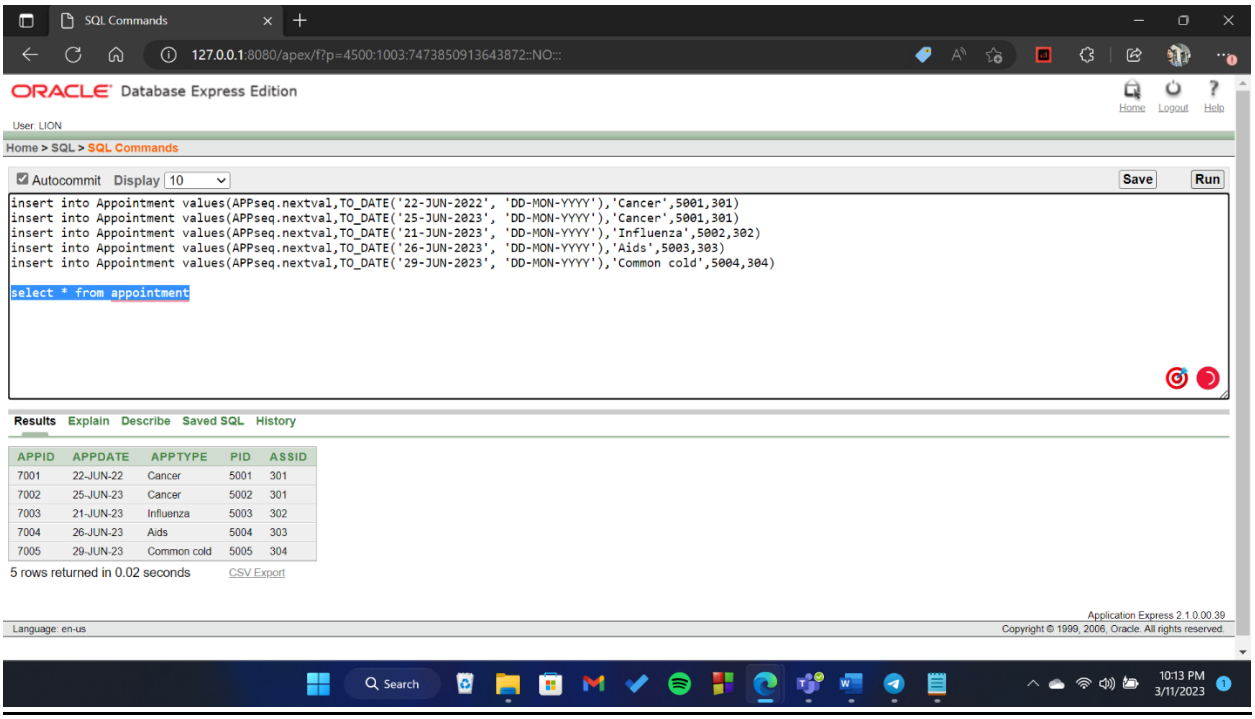
insert into DoctorPatient values(DPseq.nextval,101,5001)

insert into DoctorPatient values(DPseq.nextval,101,5002)

insert into DoctorPatient values(DPseq.nextval,102,5003)

insert into DoctorPatient values(DPseq.nextval,103,5004)

insert into DoctorPatient values(DPseq.nextval,104,5005)



// Appointment

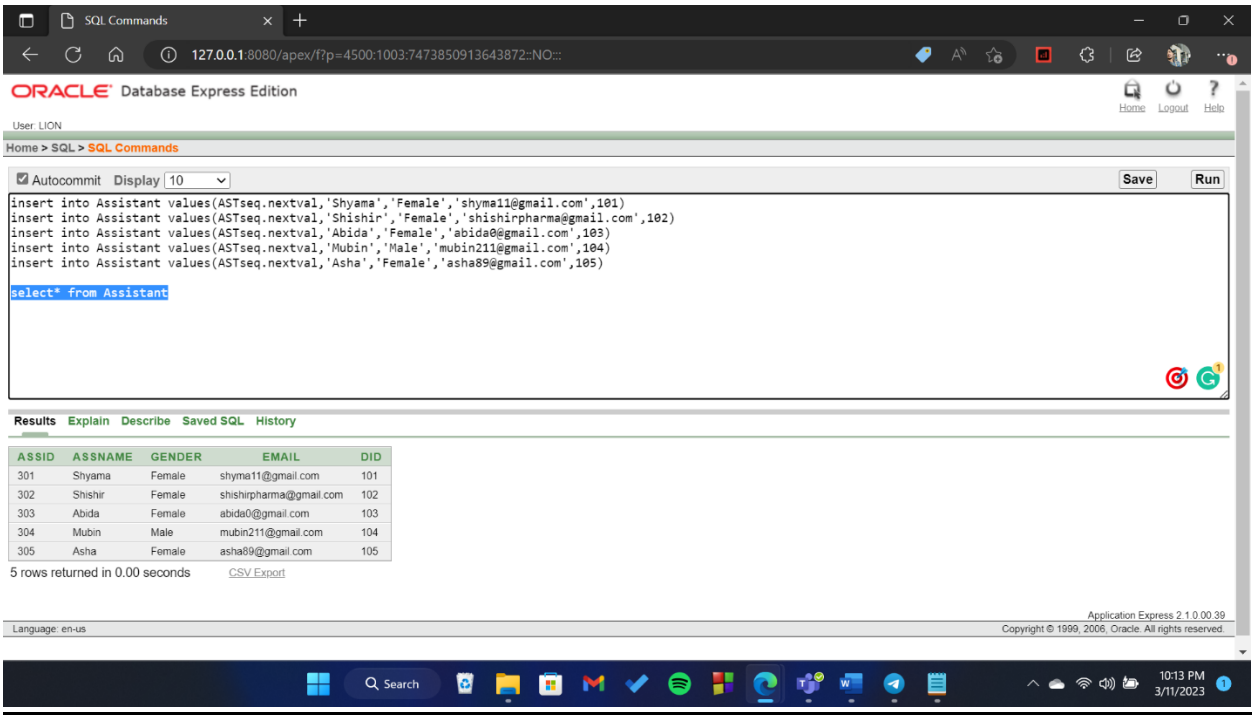
insert into Appointment values(APPseq.nextval,TO_DATE('22-JUN-2022', 'DD-MON-YYYY'),'Cancer',5001,301)

insert into Appointment values(APPseq.nextval,TO_DATE('25-JUN-2023', 'DD-MON-YYYY'),'Cancer',5001,301)

insert into Appointment values(APPseq.nextval,TO_DATE('21-JUN-2023', 'DD-MON-YYYY'),'Influenza',5002,302)

insert into Appointment values(APPseq.nextval,TO_DATE('26-JUN-2023', 'DD-MON-YYYY'),'Aids',5003,303)

insert into Appointment values(APPseq.nextval,TO_DATE('29-JUN-2023', 'DD-MON-YYYY'),'Common cold',5004,304)



//Assistant

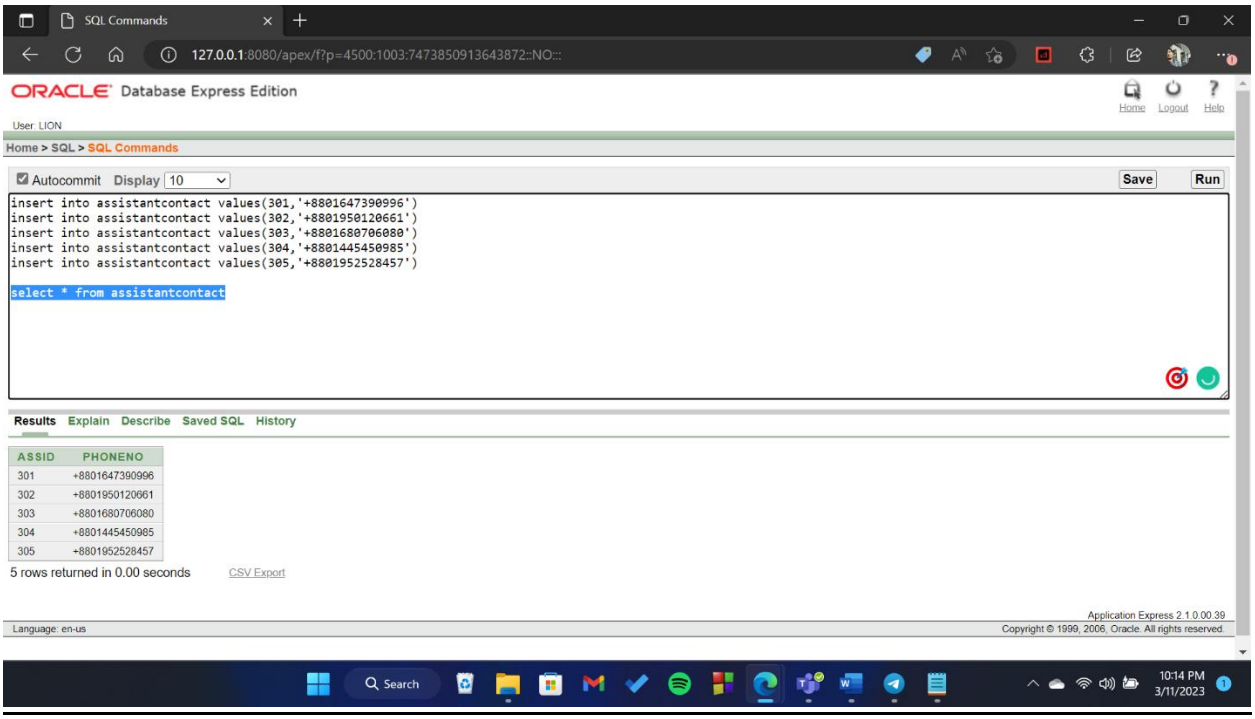
insert into Assistant values(ASTseq.nextval,'Shyama','Female','shyma11@gmail.com',101)

insert into Assistant values(ASTseq.nextval,'Shishir','Female','shishirpharma@gmail.com',102)

insert into Assistant values(ASTseq.nextval,'Abida','Female','abida0@gmail.com',103)

insert into Assistant values(ASTseq.nextval,'Mubin','Male','mubin211@gmail.com',104)

insert into Assistant values(ASTseq.nextval,'Asha','Female','asha89@gmail.com',105)



//AssistantContact

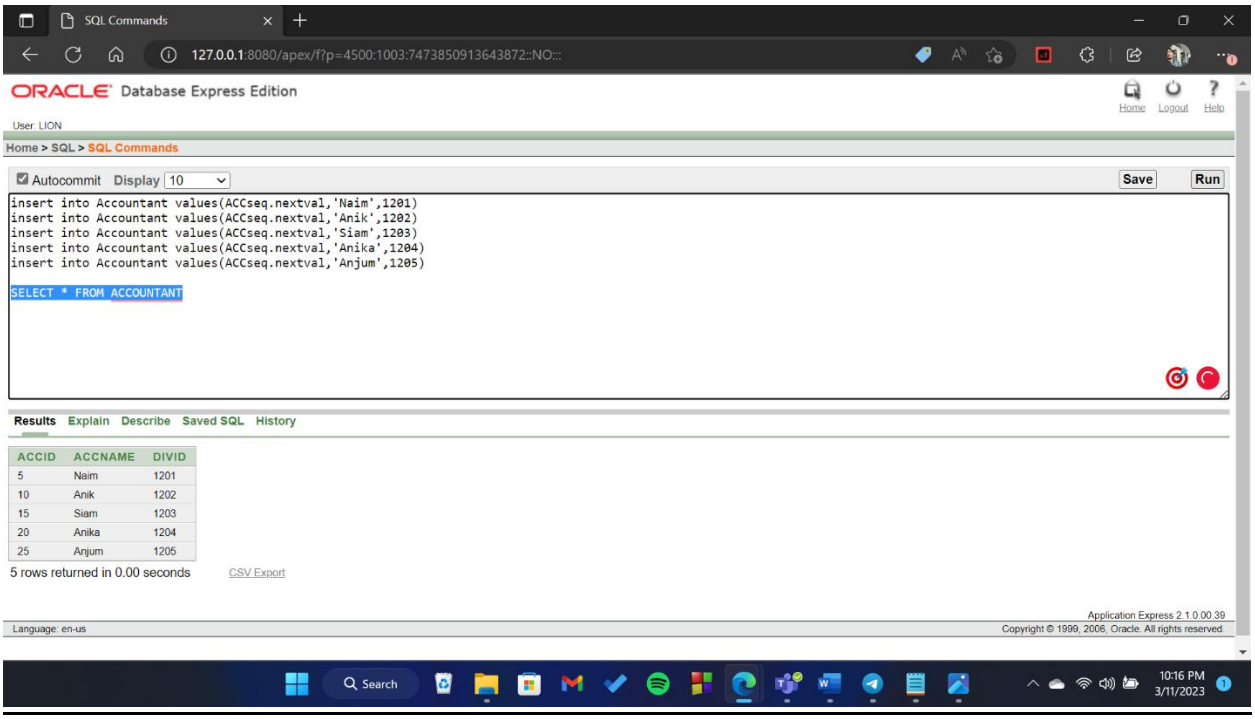
insert into assistantcontact values(301,'+8801647390996')

insert into assistantcontact values(302,'+8801950120661')

insert into assistantcontact values(303,'+8801680706080')

insert into assistantcontact values(304,'+8801445450985')

insert into assistantcontact values(305,'+8801952528457')



//Accountant

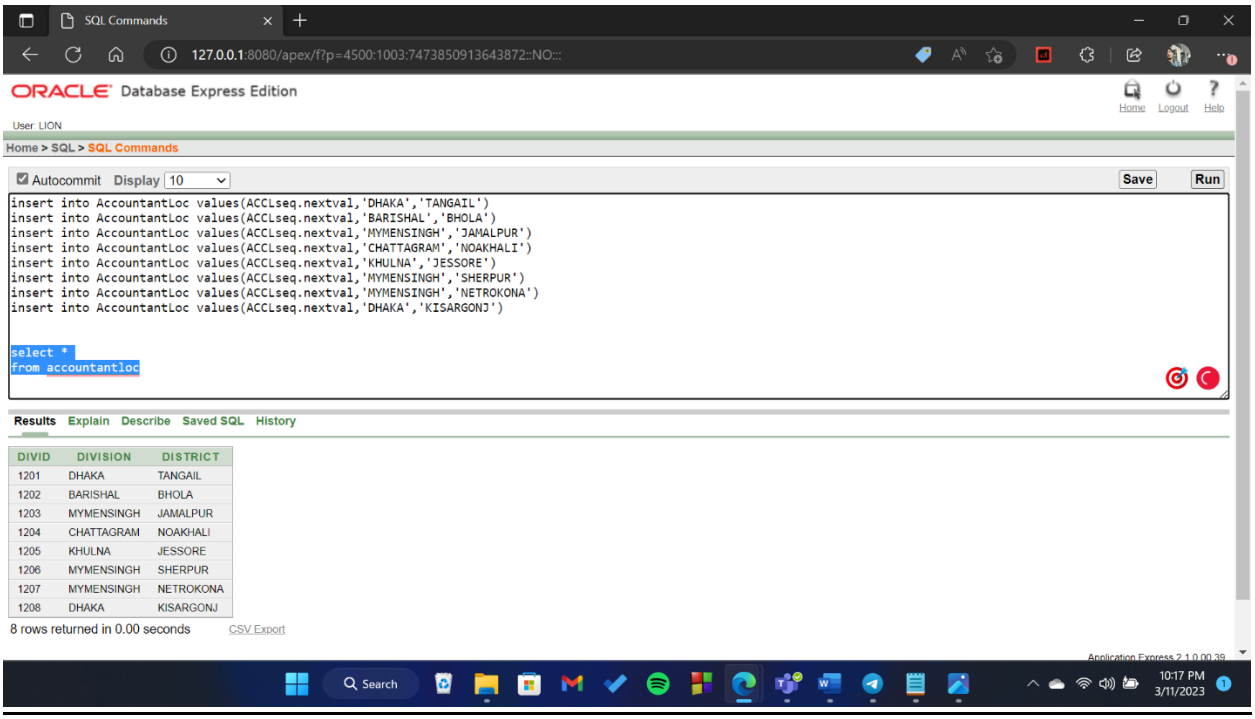
insert into Accountant values(ACCseq.nextval,'Naim',1201)

insert into Accountant values(ACCseq.nextval,'Anik',1202)

insert into Accountant values(ACCseq.nextval,'Siam',1203)

insert into Accountant values(ACCseq.nextval,'Anika',1204)

insert into Accountant values(ACCseq.nextval,'Anjum',1205)



//AccountantLoc

insert into AccountantLoc values(ACCLseq.nextval,'DHAKA','TANGAIL')

insert into AccountantLoc values(ACCLseq.nextval,'BARISHAL','BHOLA')

insert into AccountantLoc values(ACCLseq.nextval,'MYMENSINGH','JAMALPUR')

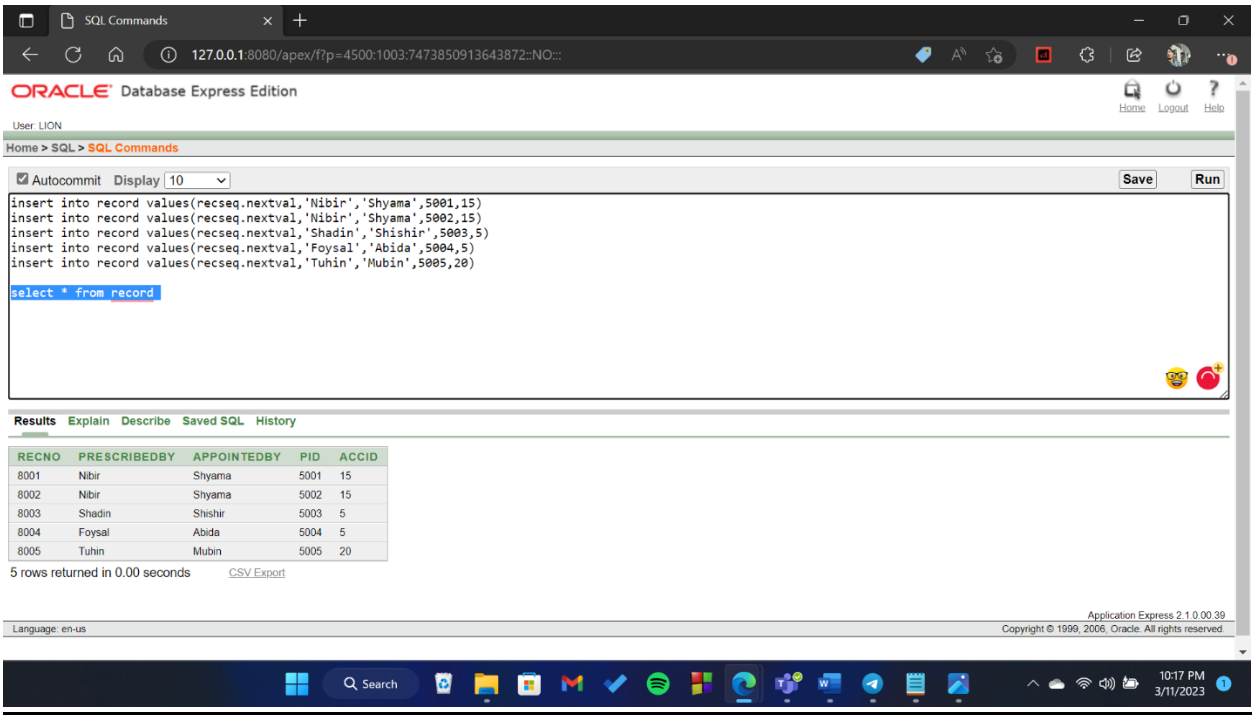
insert into AccountantLoc values(ACCLseq.nextval,'CHATTAGRAM','NOAKHALI')

insert into AccountantLoc values(ACCLseq.nextval,'KHULNA','JESSORE')

insert into AccountantLoc values(ACCLseq.nextval,'MYMENSINGH','SHERPUR')

insert into AccountantLoc values(ACCLseq.nextval,'MYMENSINGH','NETROKONA')

insert into AccountantLoc values(ACCLseq.nextval,'DHAKA','KISARGONJ')



//Record

insert into record values(recseq.nextval,'Nibir','Shyama',5001,15)

insert into record values(recseq.nextval,'Nibir','Shyama',5002,15)

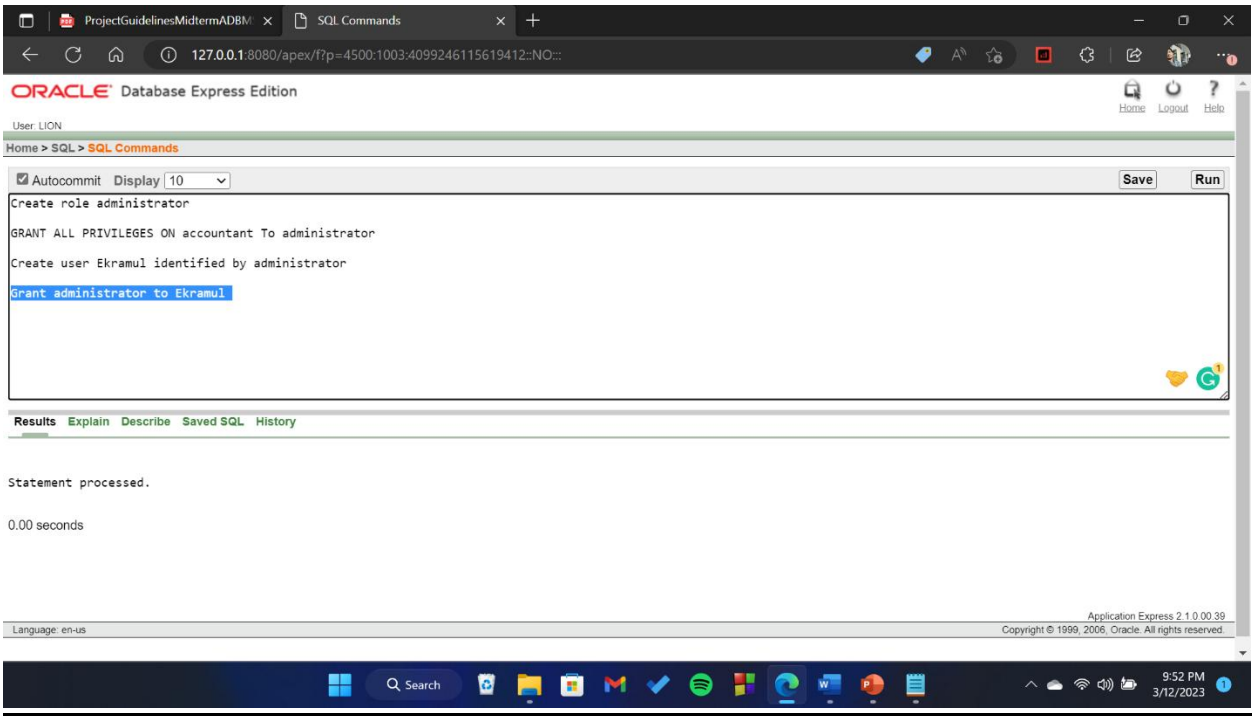
insert into record values(recseq.nextval,'Shadin','Shishir',5003,5)

insert into record values(recseq.nextval,'Foysal','Abida',5004,5)

insert into record values(recseq.nextval,'Tuhin','Mubin',5005,20)

Create user, role and assigning role:

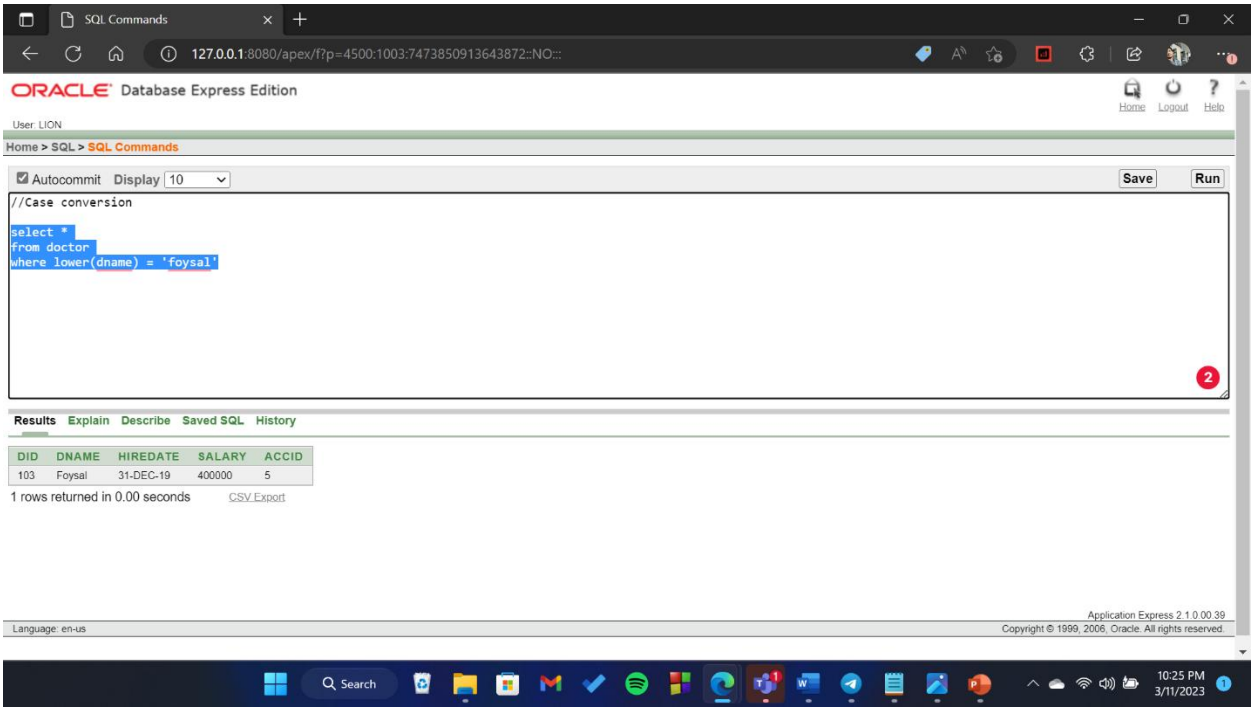
- Create a role administrator. The role administrator has access to all the object privileges. Ekramul (password: administrator) is the administrator.



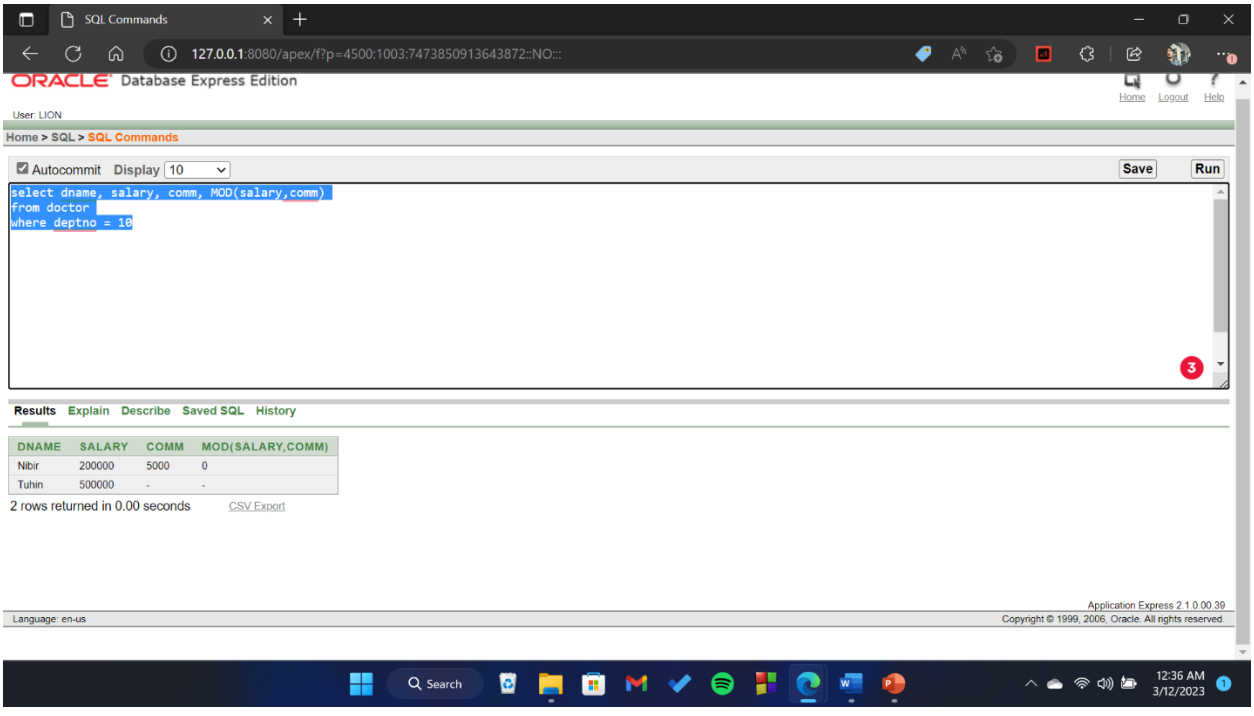
SQL Query Writing:

Single row function:

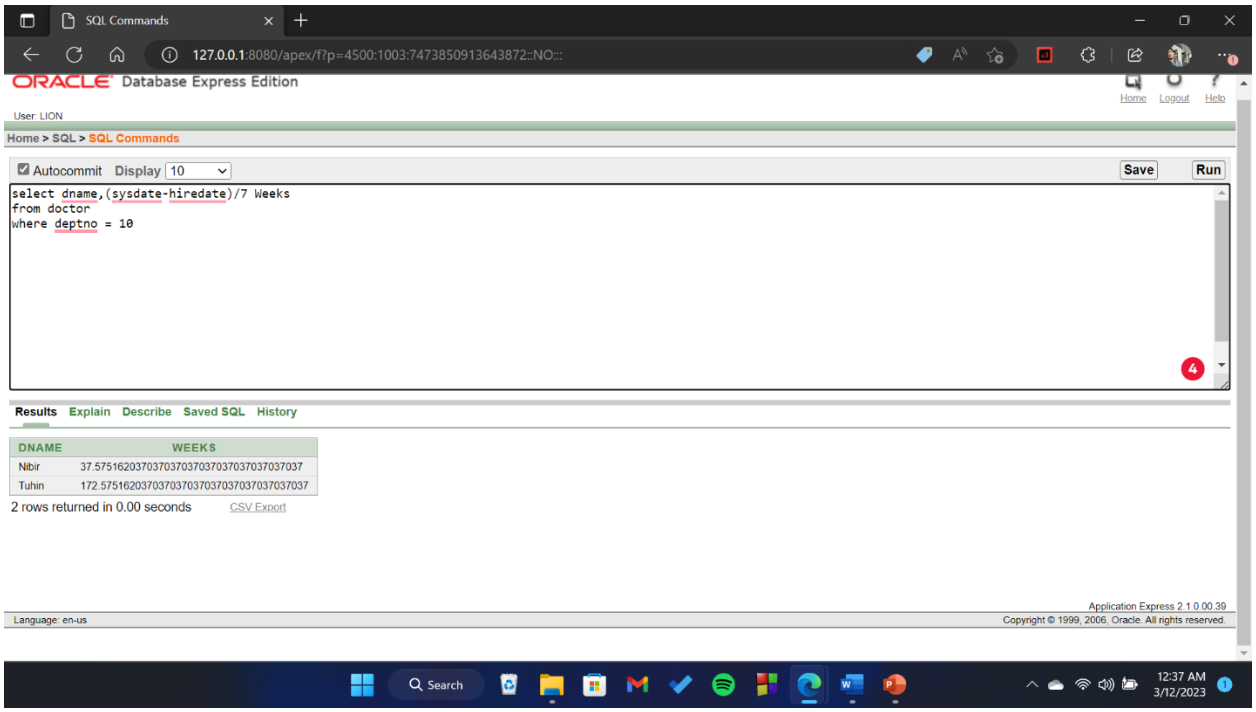
1. Write a query to display the information of the doctor named foysal.



2. Display the name, salary, comm and mod of comm and salary whose department no is 10

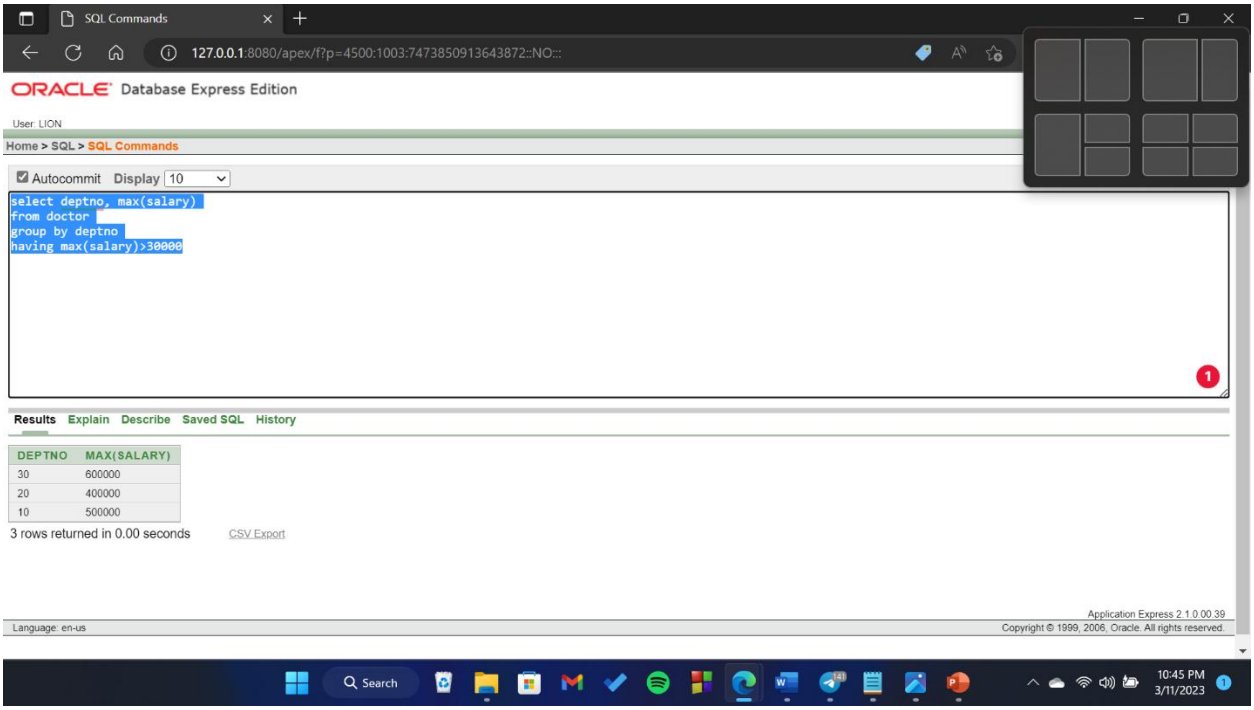


3. Display the doctor's name and the total number of weeks have they been attending whose department no is 10.

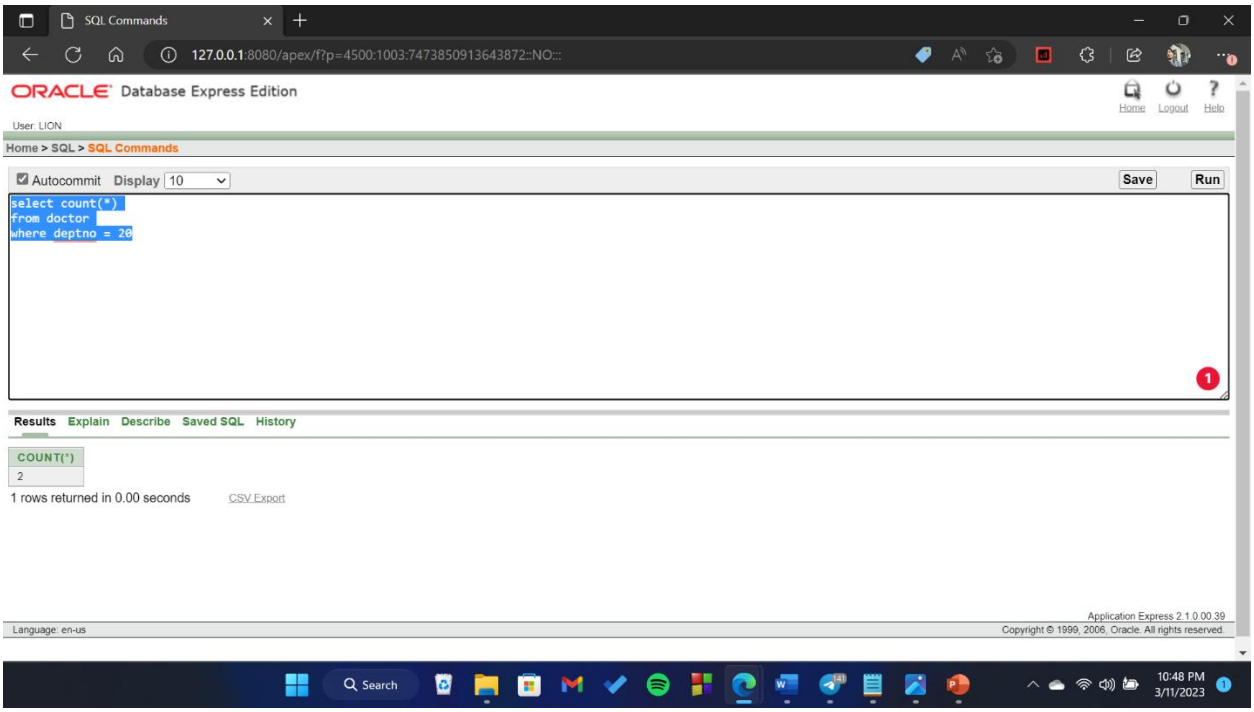


Group function:

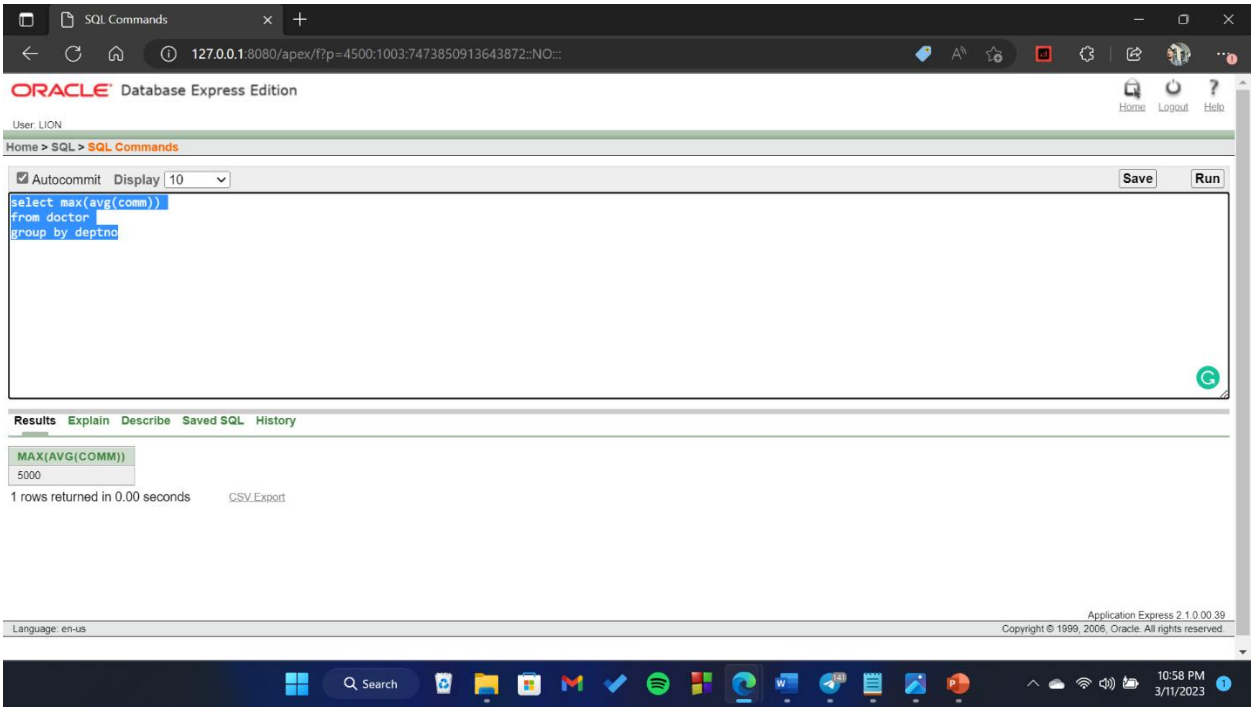
1. Display department wise Maximum salary, whose salary is higher than 300000.



2. Count the number of doctors who works under department 20.

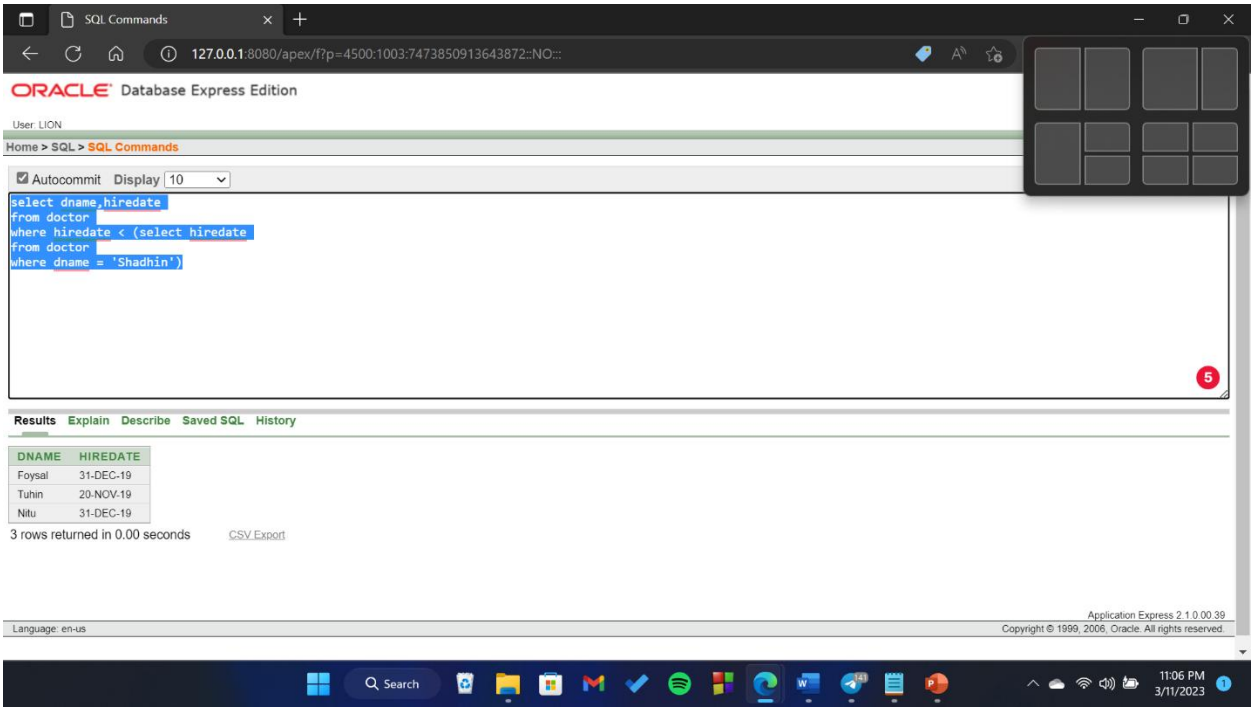


3. Display the maximum average salary

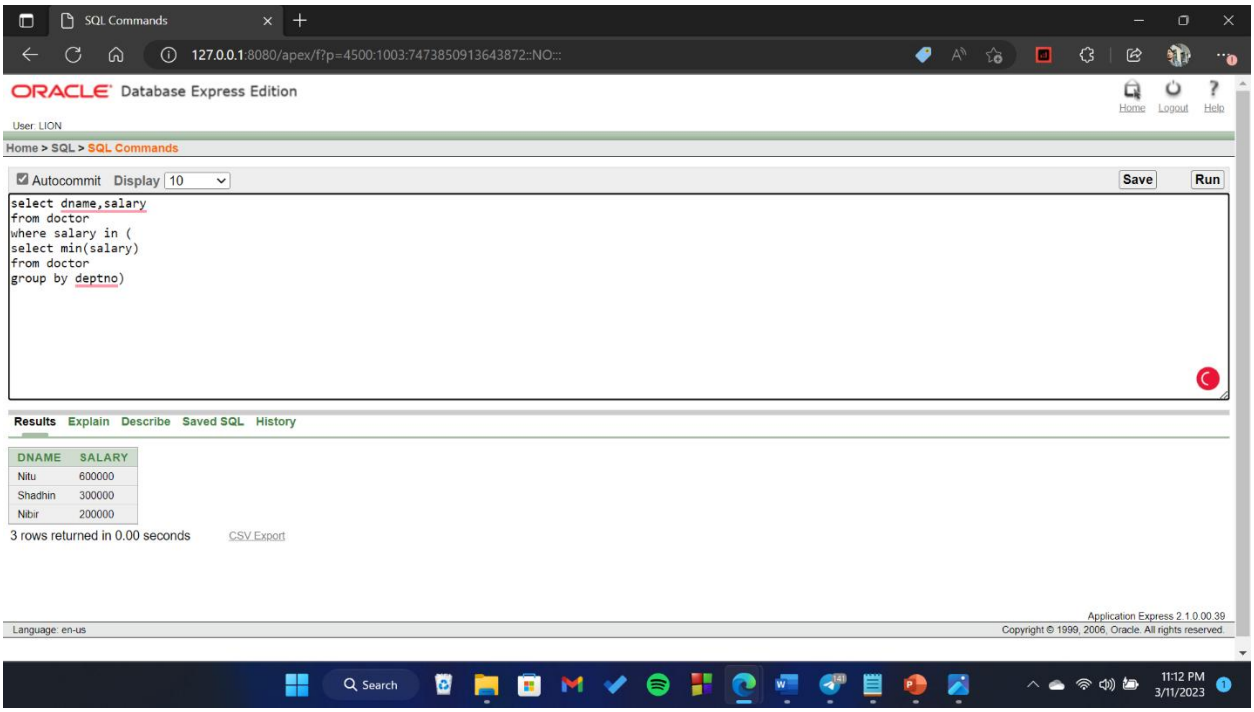


Subquery:

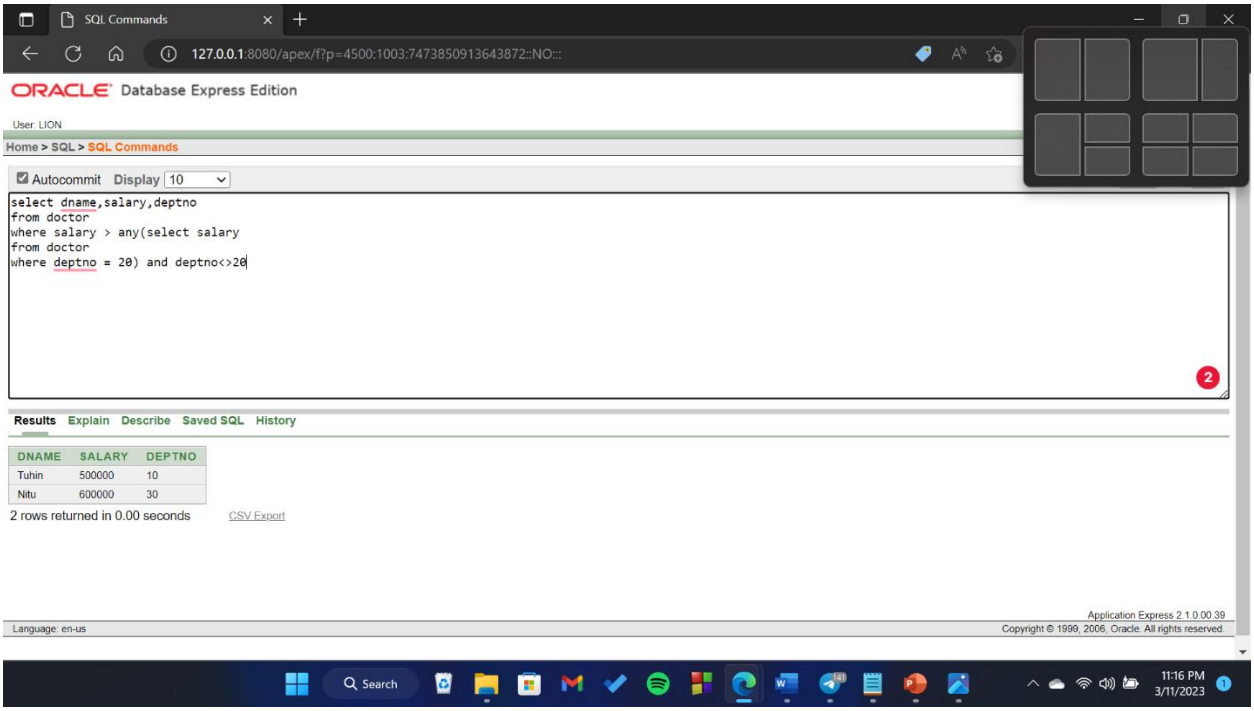
1. Display the doctor name and hiredate who joined before Shadin.



2. Display the doctor name who get the lowest salary department wise.

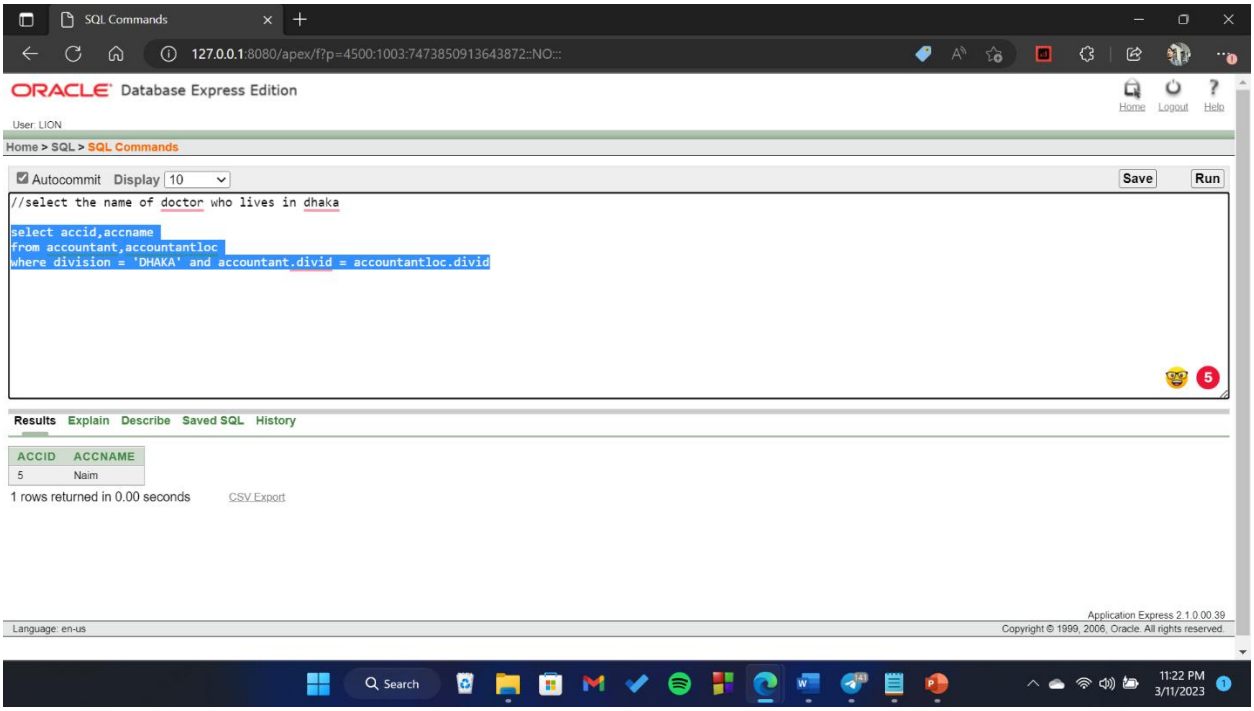


3. What is the list of doctors' department names, salaries, and department numbers, where the salary is higher than any doctor's salary in department 20, and the department number is not 20?

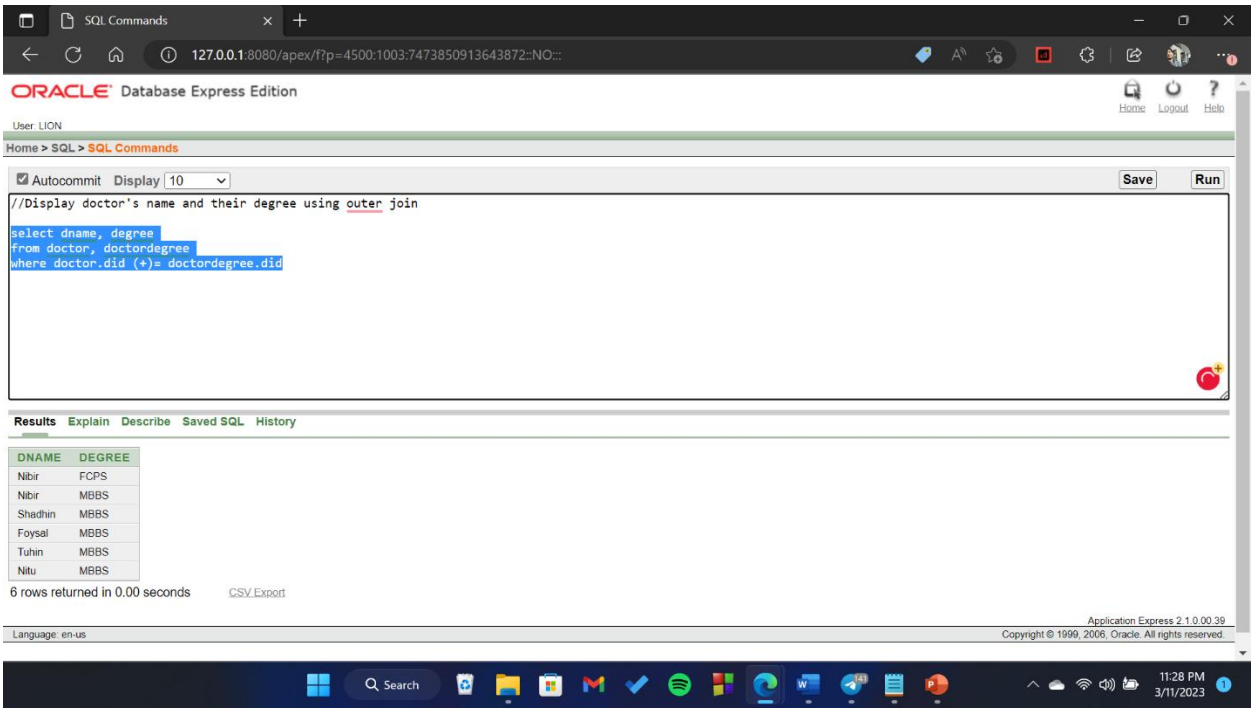


Joining:

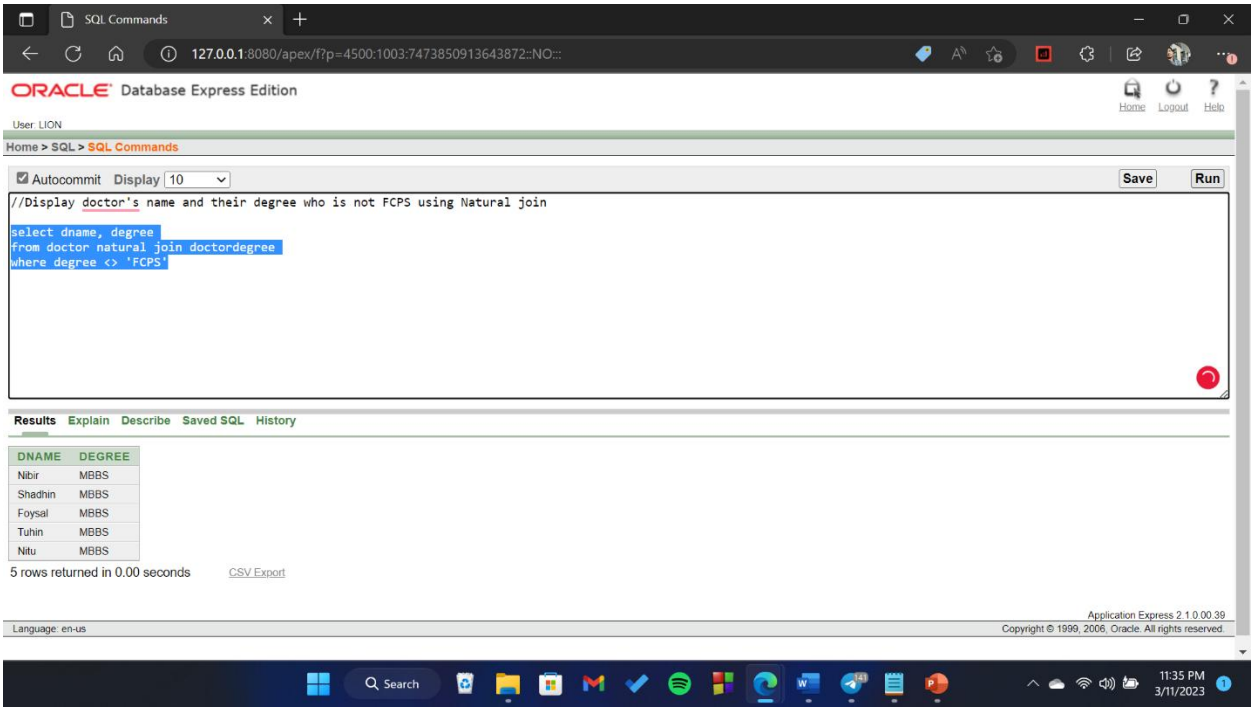
1. Display the name of doctor who lives in dhaka.



2. What is the list of doctors' names and degrees using outer join.

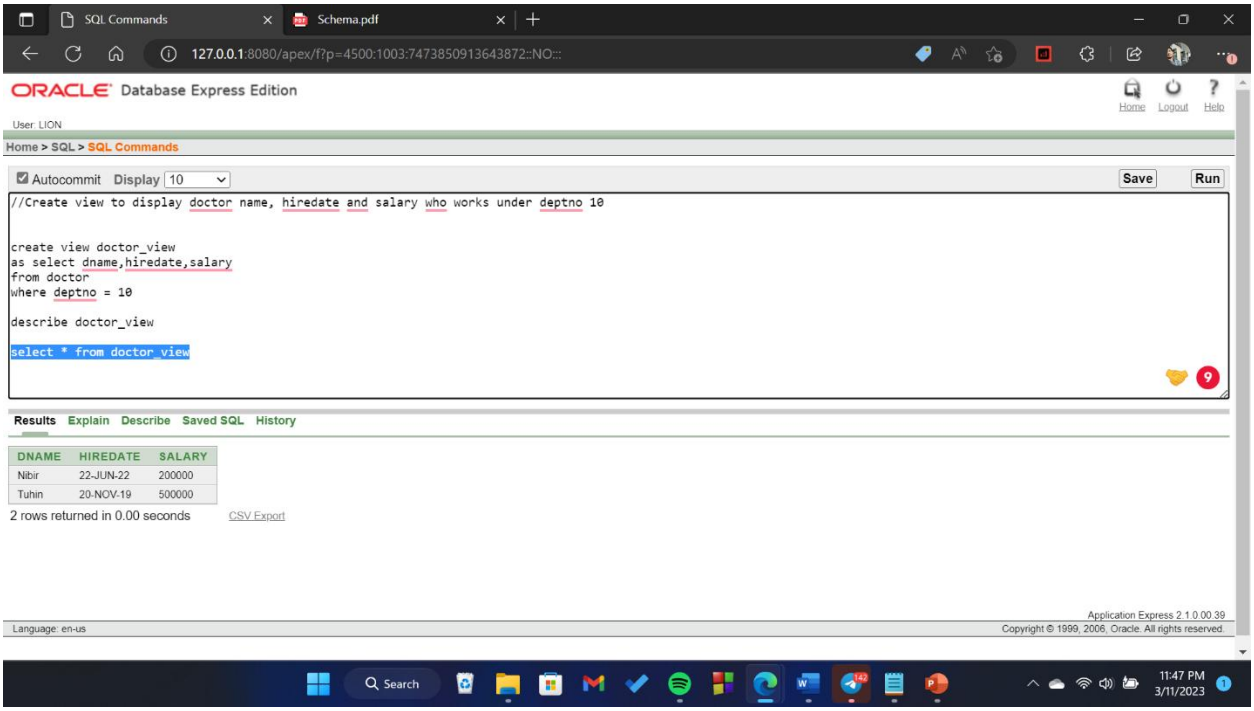


3. Display the Doctor's name and their degree who in not FCPS using Natural join

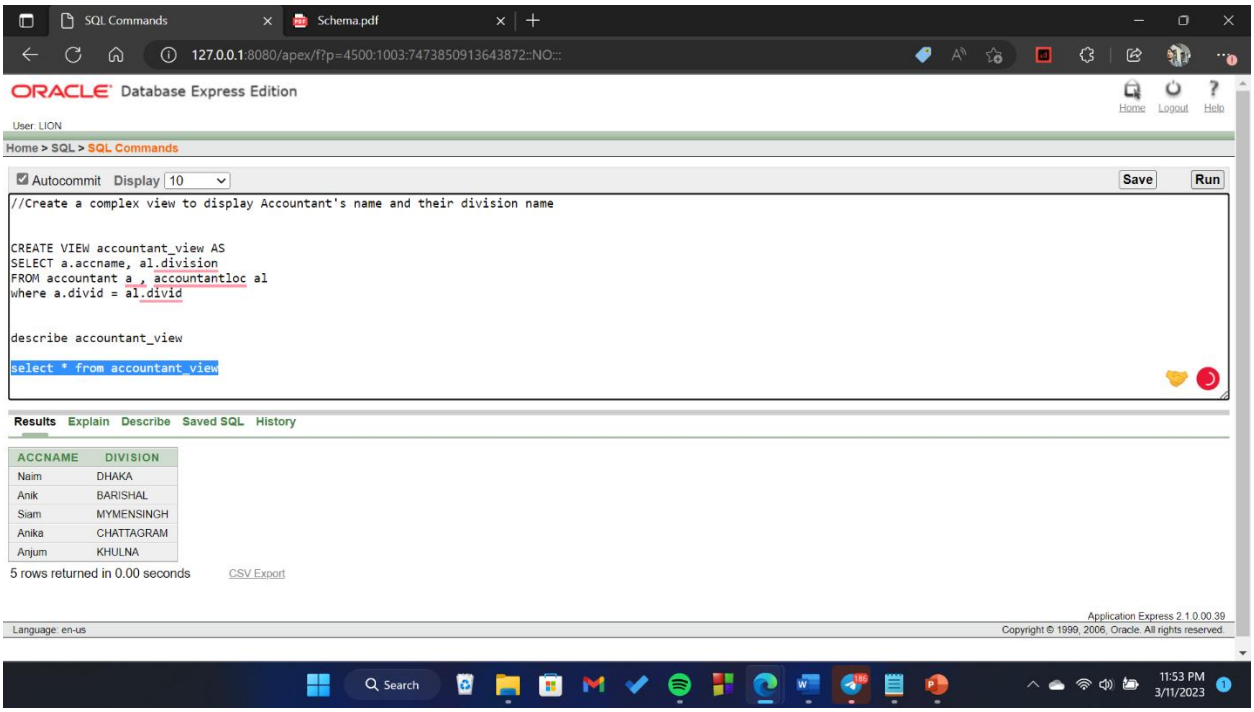


View:

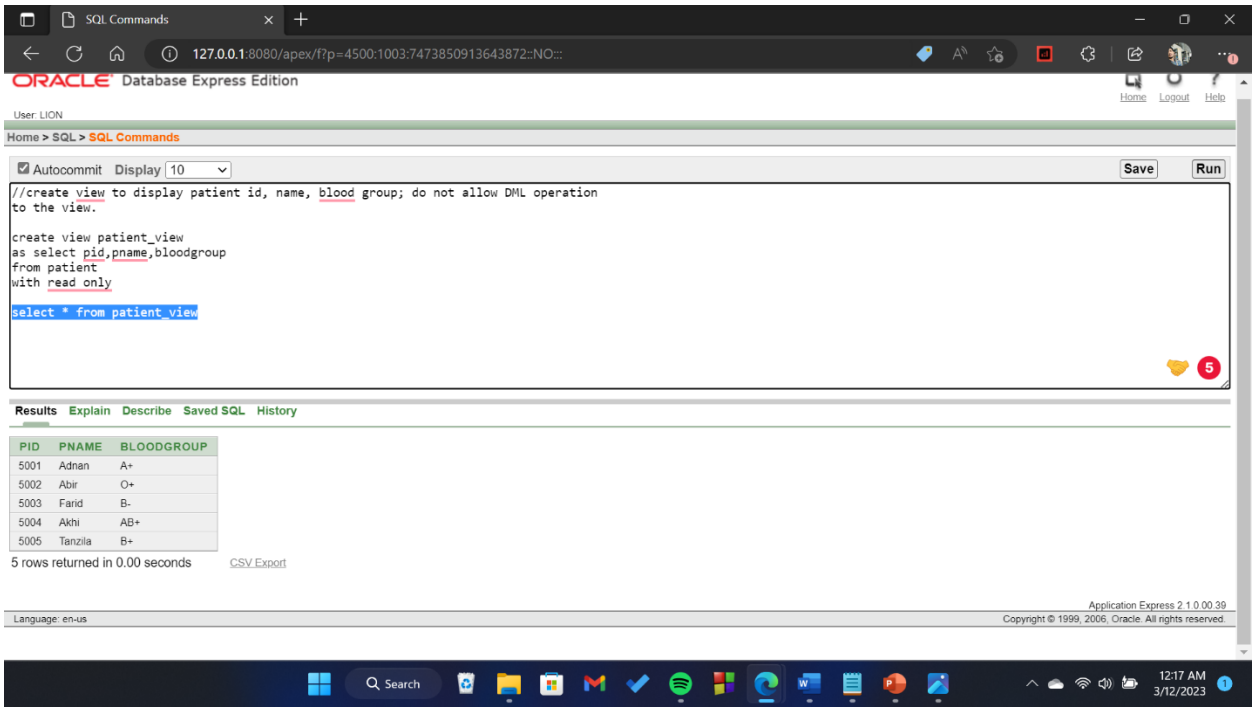
1. Create a view to display doctor name, hiredate and salary who works under deptno 10.



2. Create a complex view to display Accountant's name and their division accordingly.

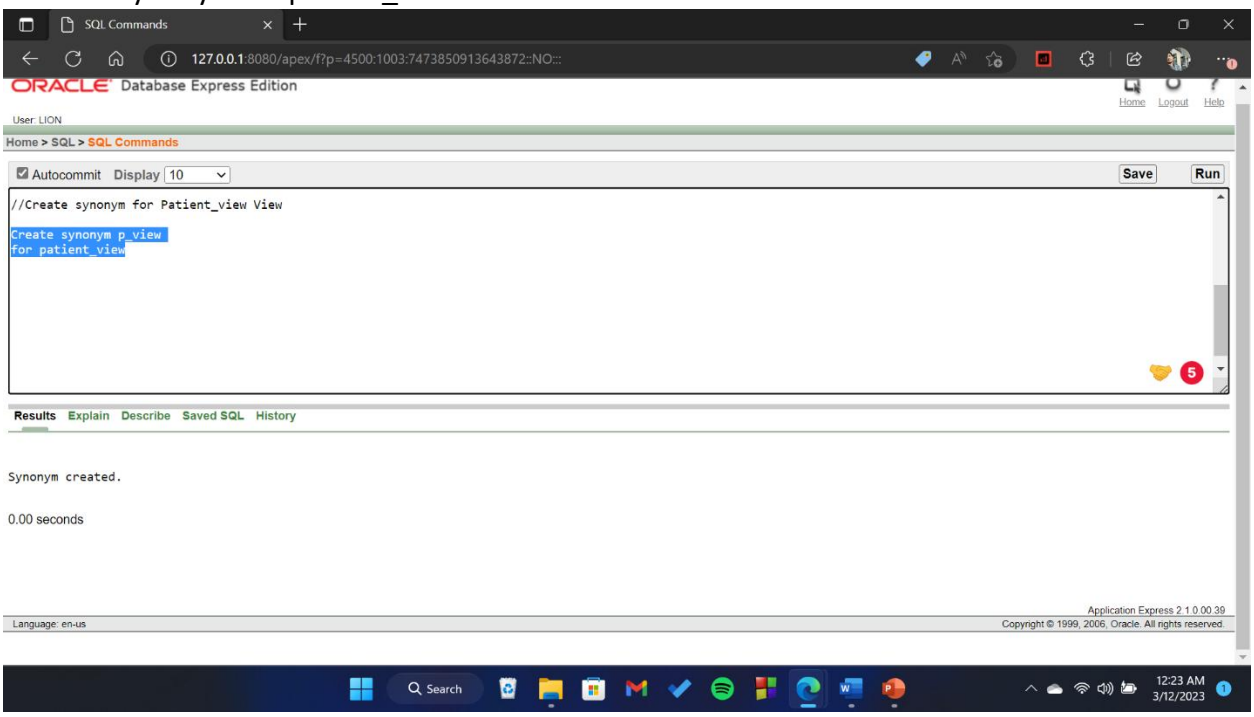


3. Create a view to display patient id, name and their blood group; Do not allow DML operation for the view.

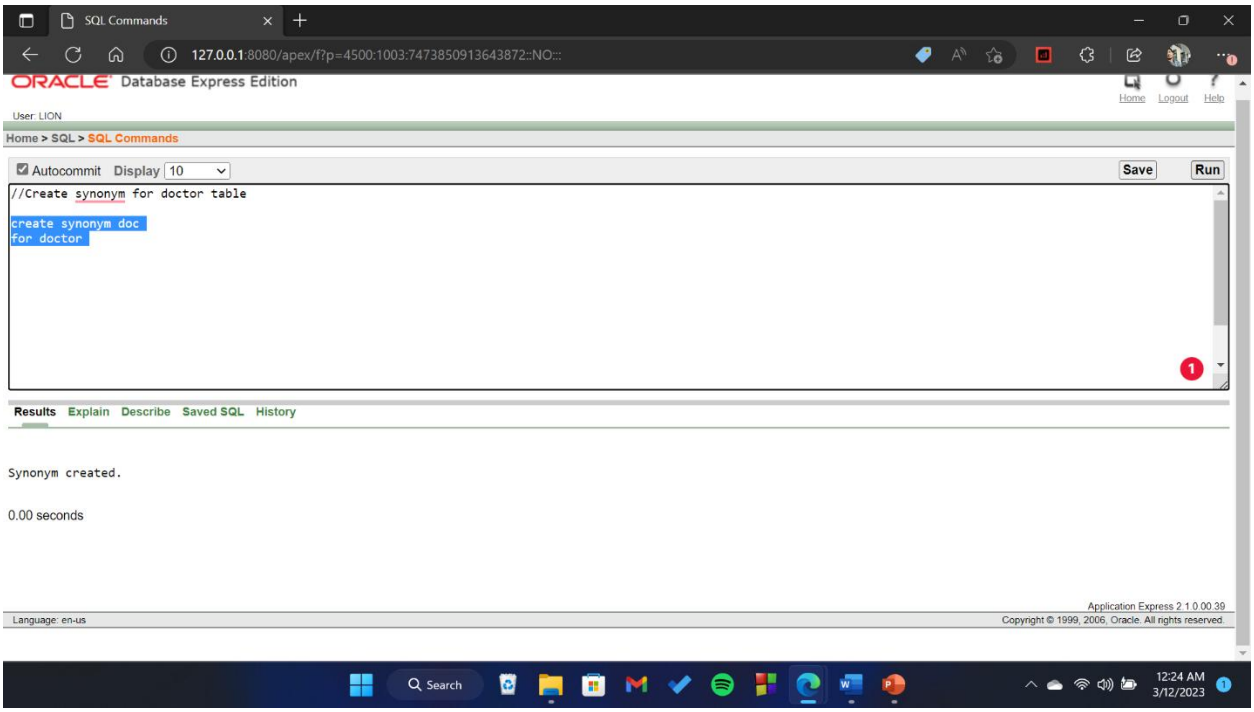


Synonym:

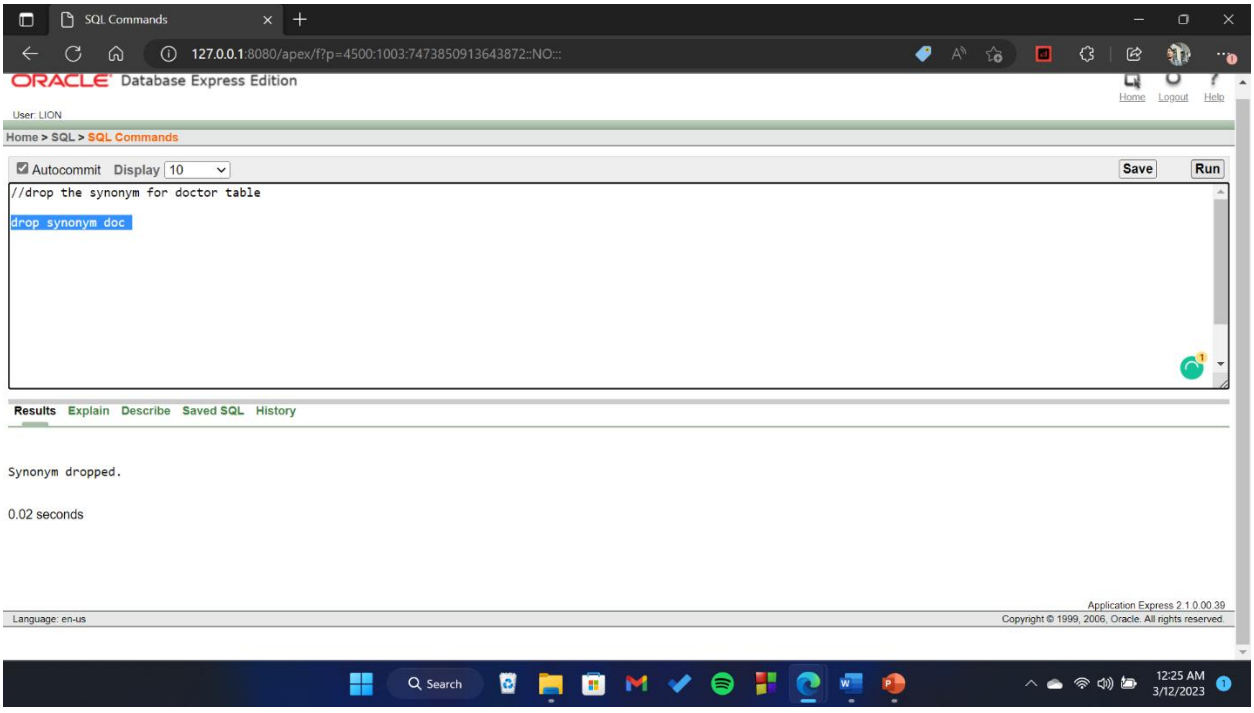
1. Create a synonym for patient_view view.



2. Create synonym for doctor table.



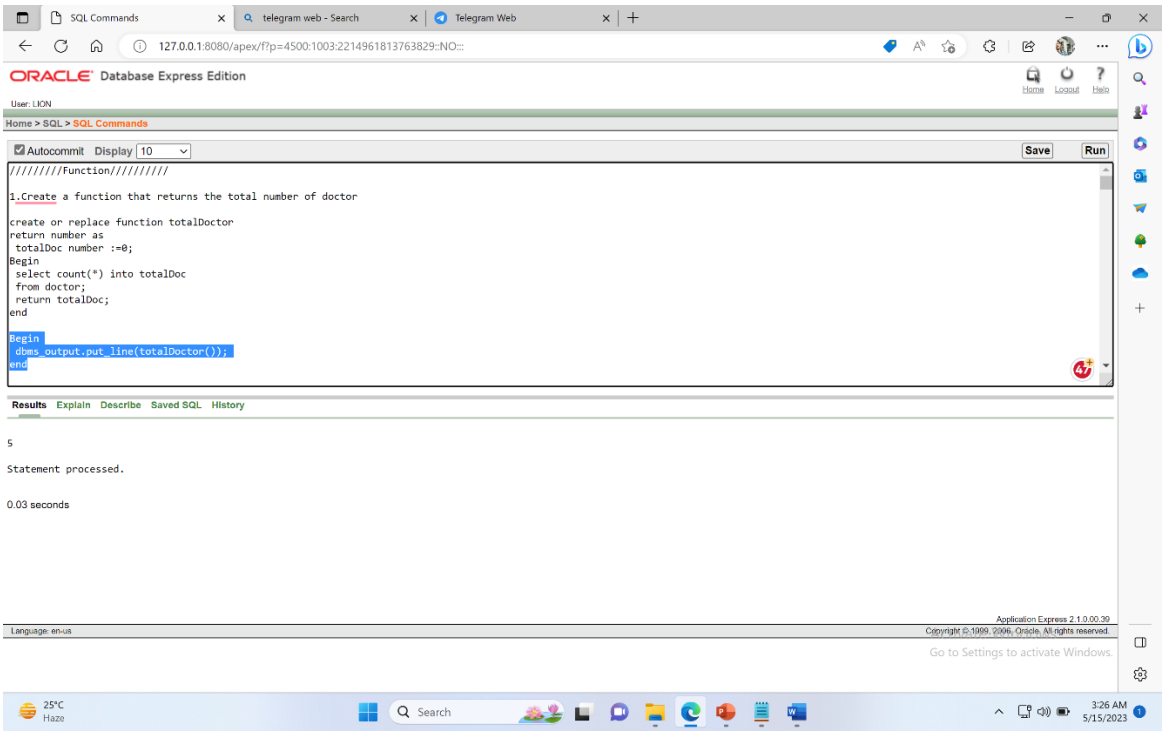
3. Delete the created synonym for doctor table.



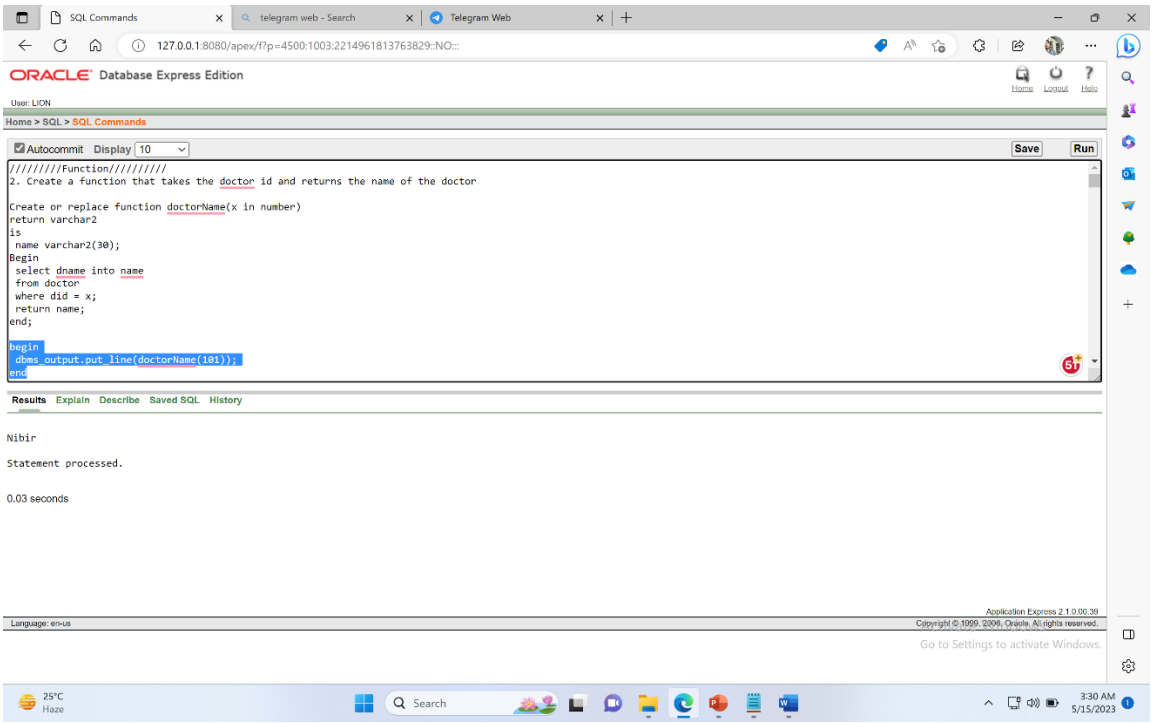
PL/SQL Query Writing:

Function:

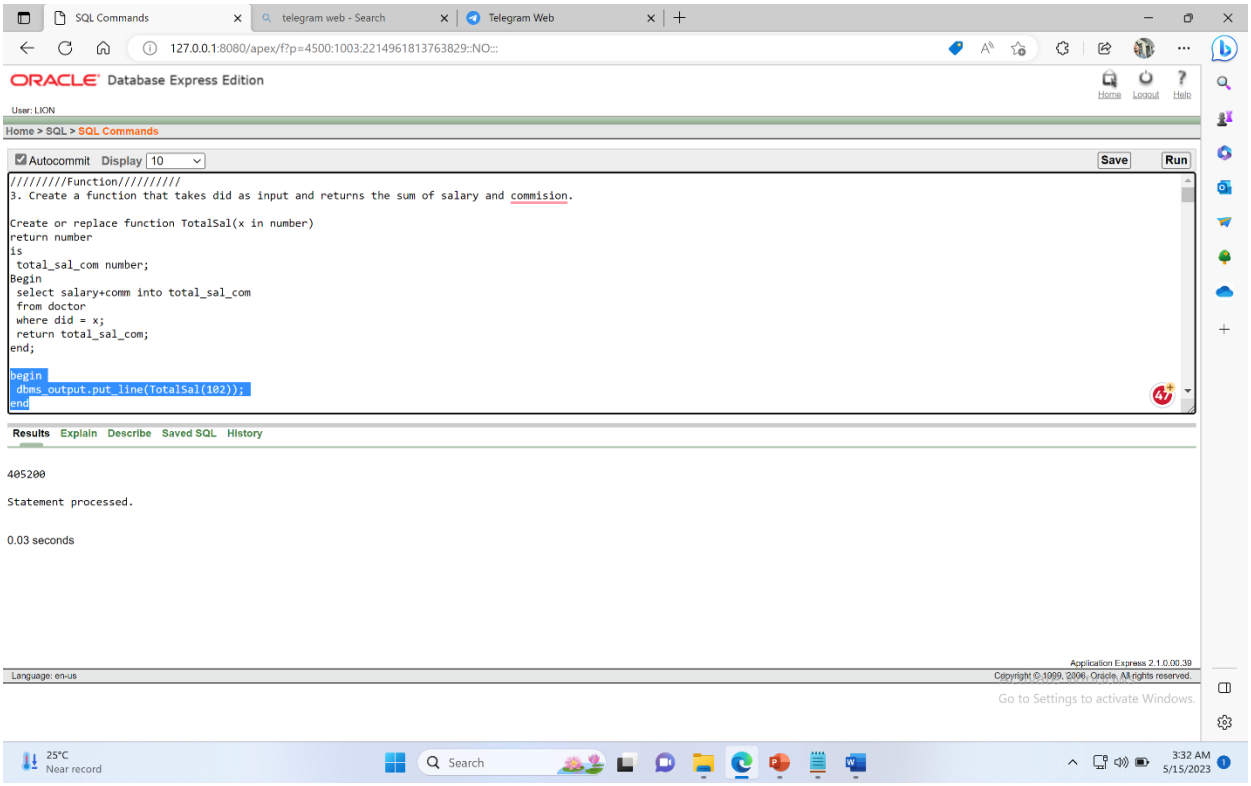
1.Create a function that returns the total number of doctor



2. Create a function that takes the doctor id and returns the name of the doctor

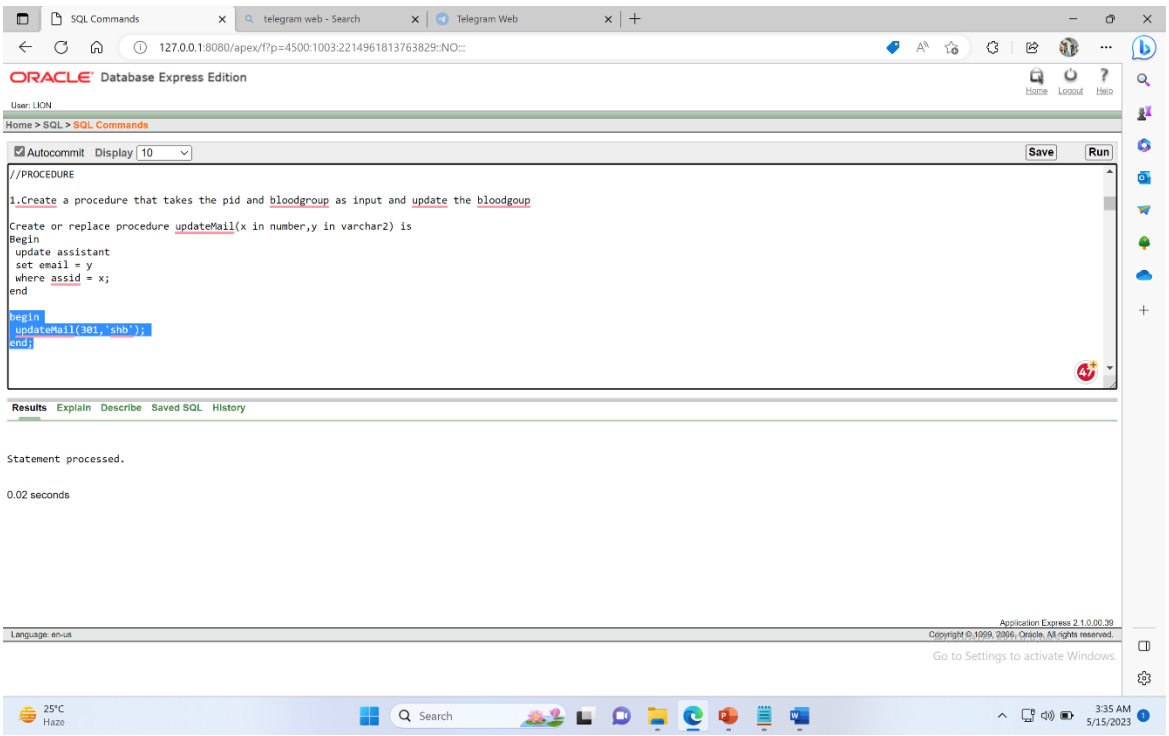


3. Create a function that takes did as input and returns the sum of salary and commision.

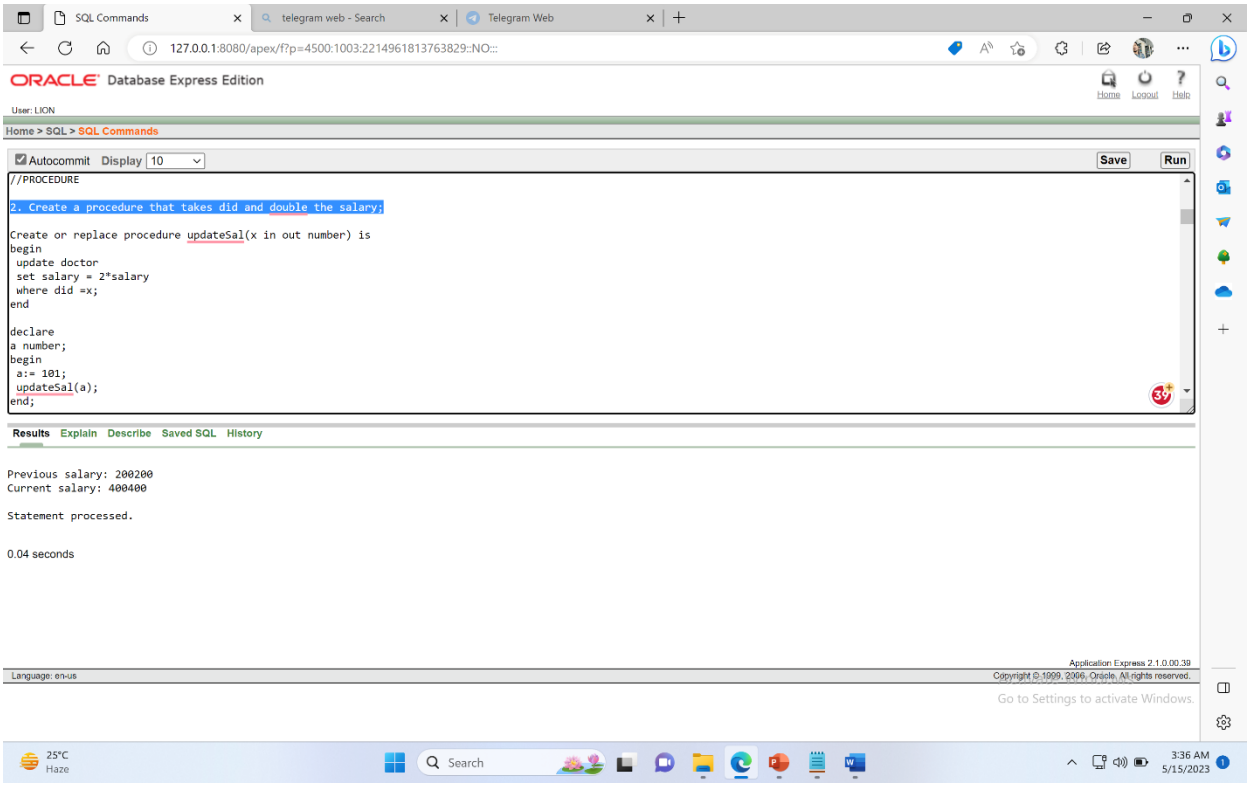


Procedure:

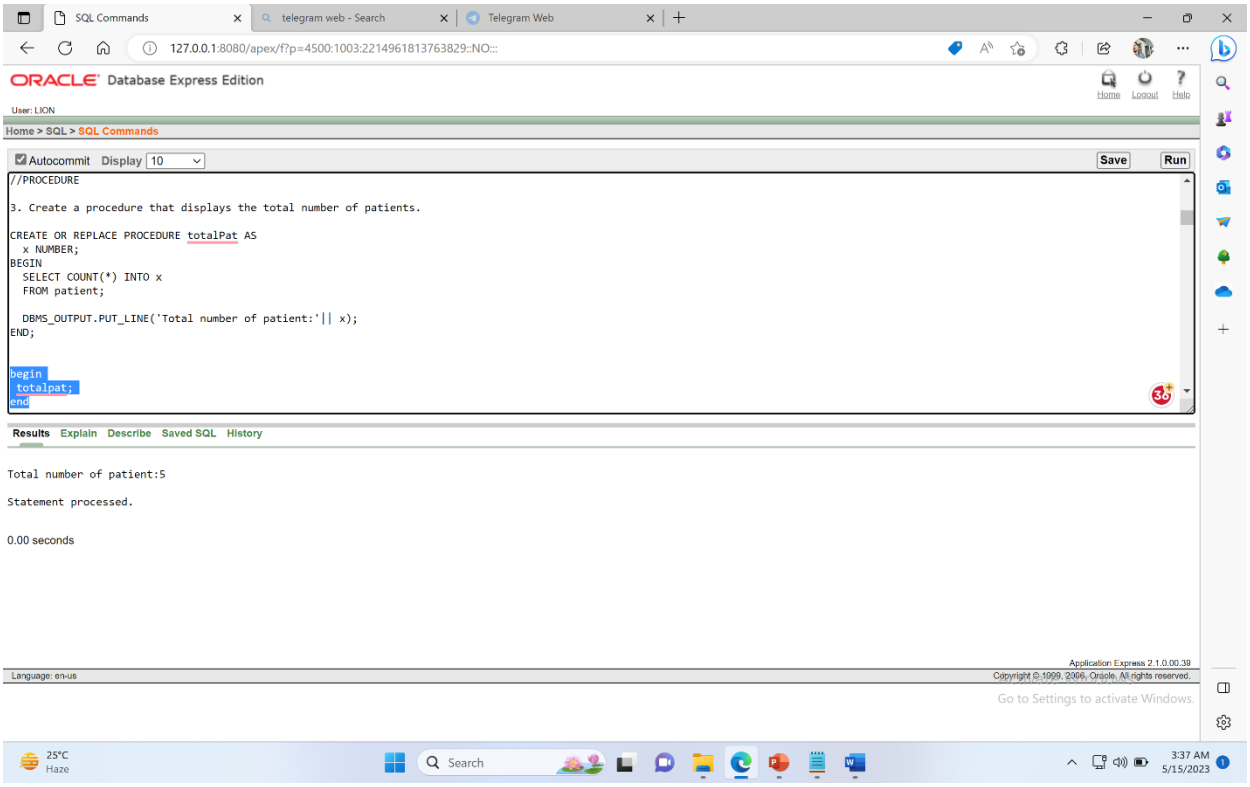
1.Create a procedure that takes the pid and bloodgroup as input and update the bloodgoup



2. Create a procedure that takes did and double the salary;

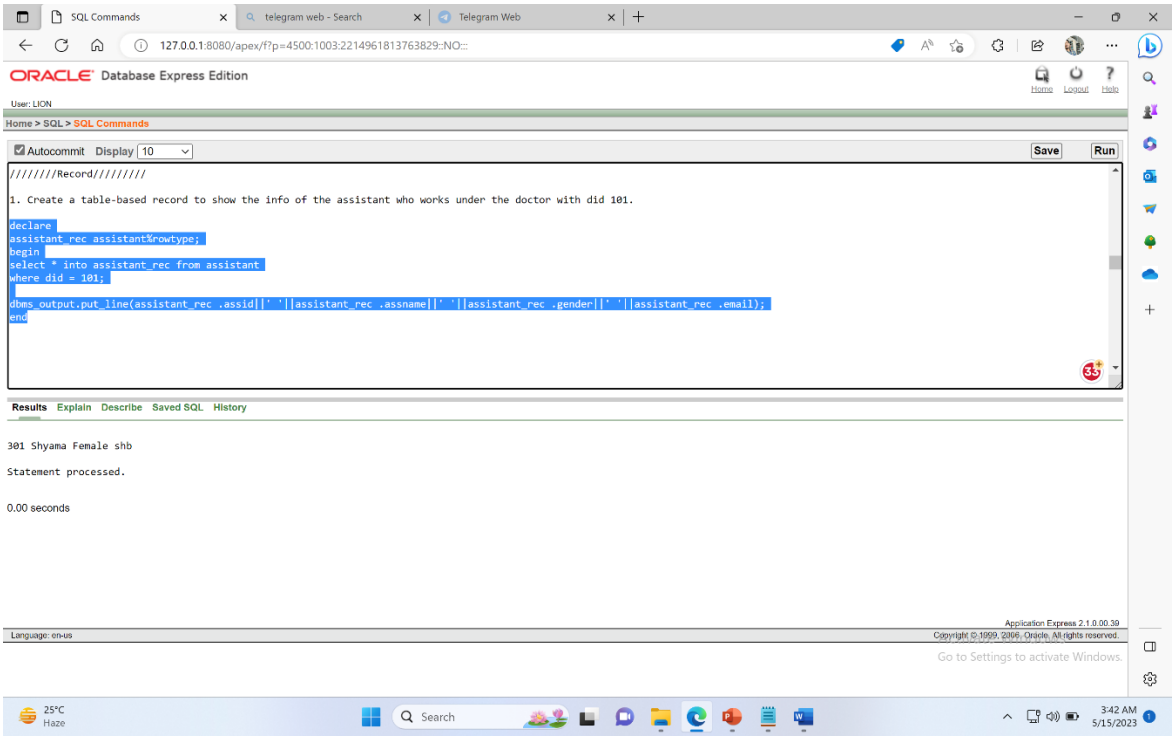


3. Create a procedure that displays the total number of patients.

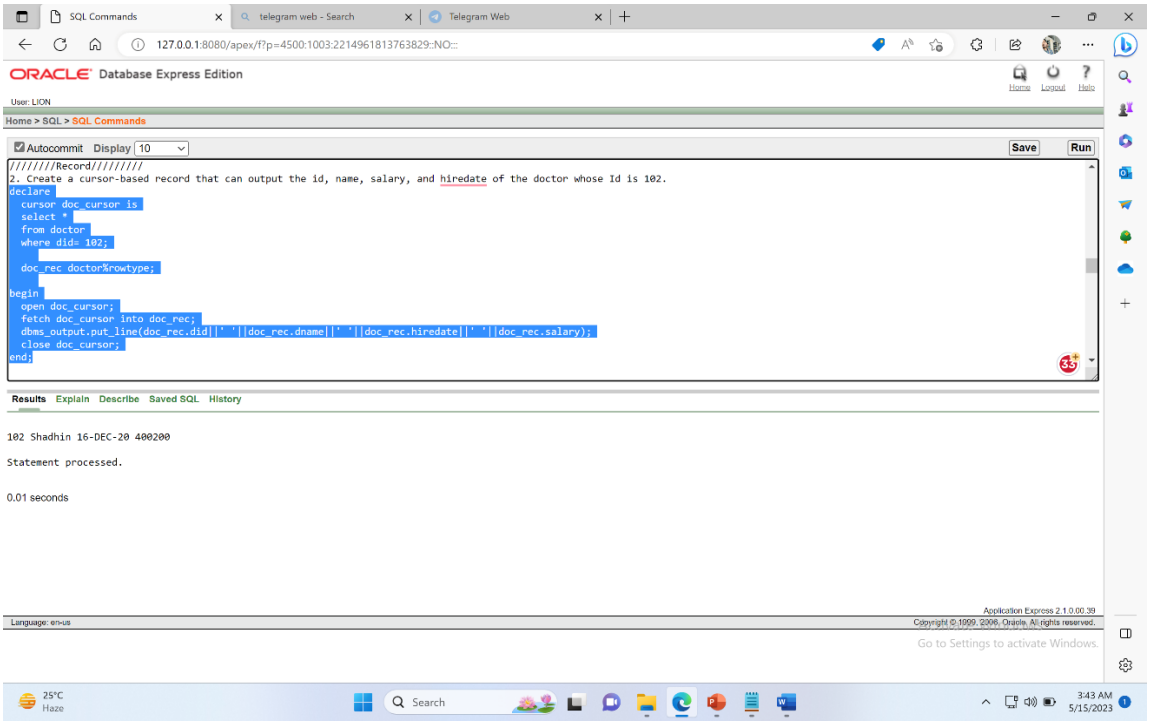


Record:

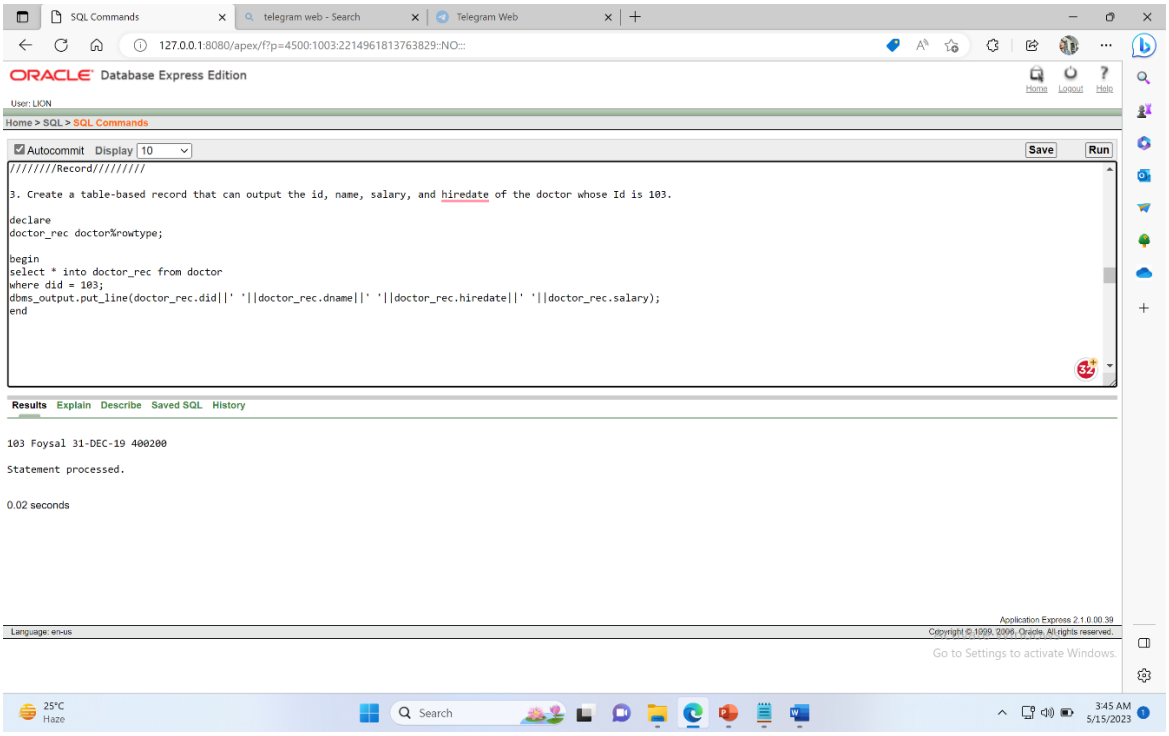
1. Create a table-based record to show the info of the assistant who works under the doctor with did 101.



2. Create a cursor-based record that can output the id, name, salary, and hiredate of the doctor whose Id is 102.

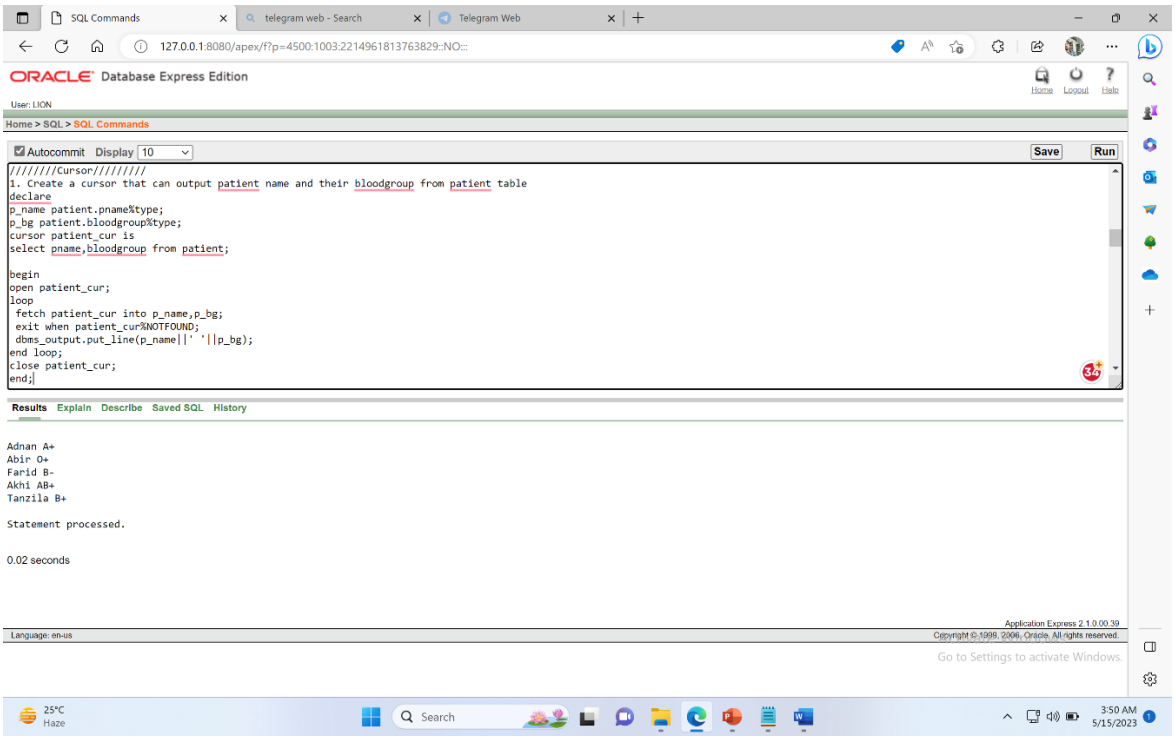


3. Create a table-based record that can output the id, name, salary, and hiredate of the doctor whose Id is 103.

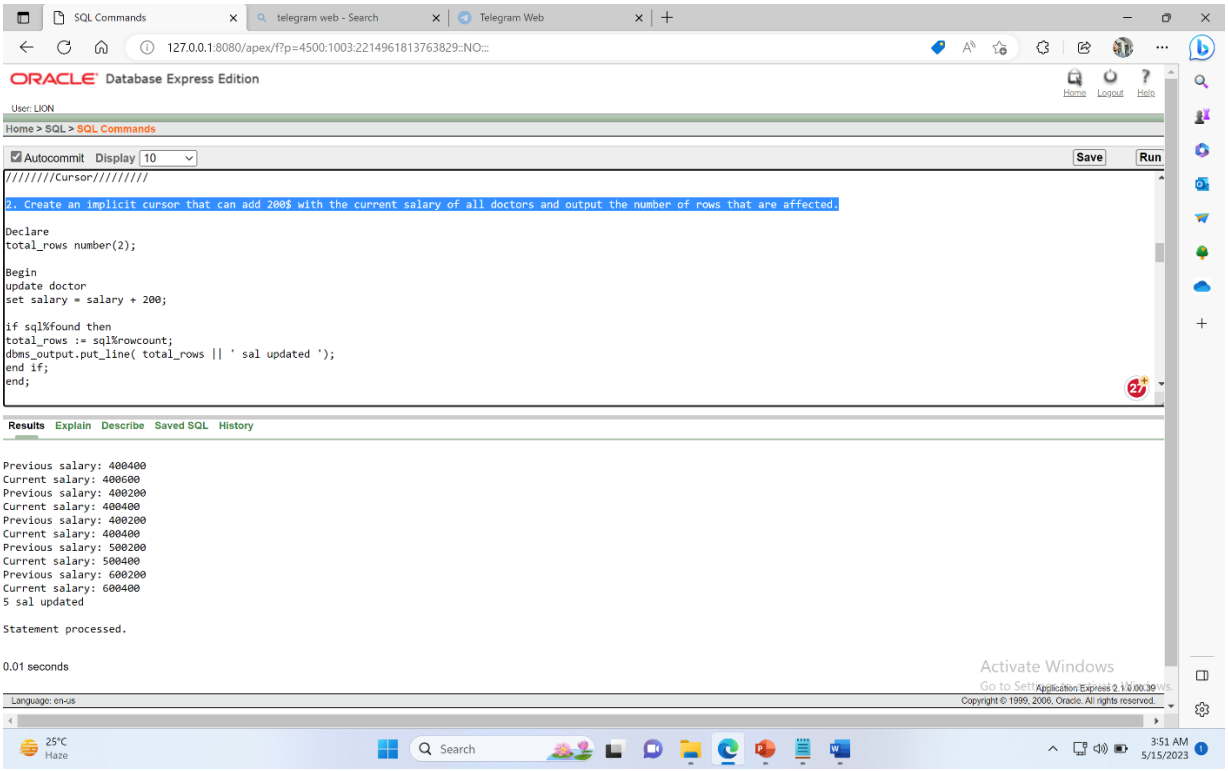


Cursor:

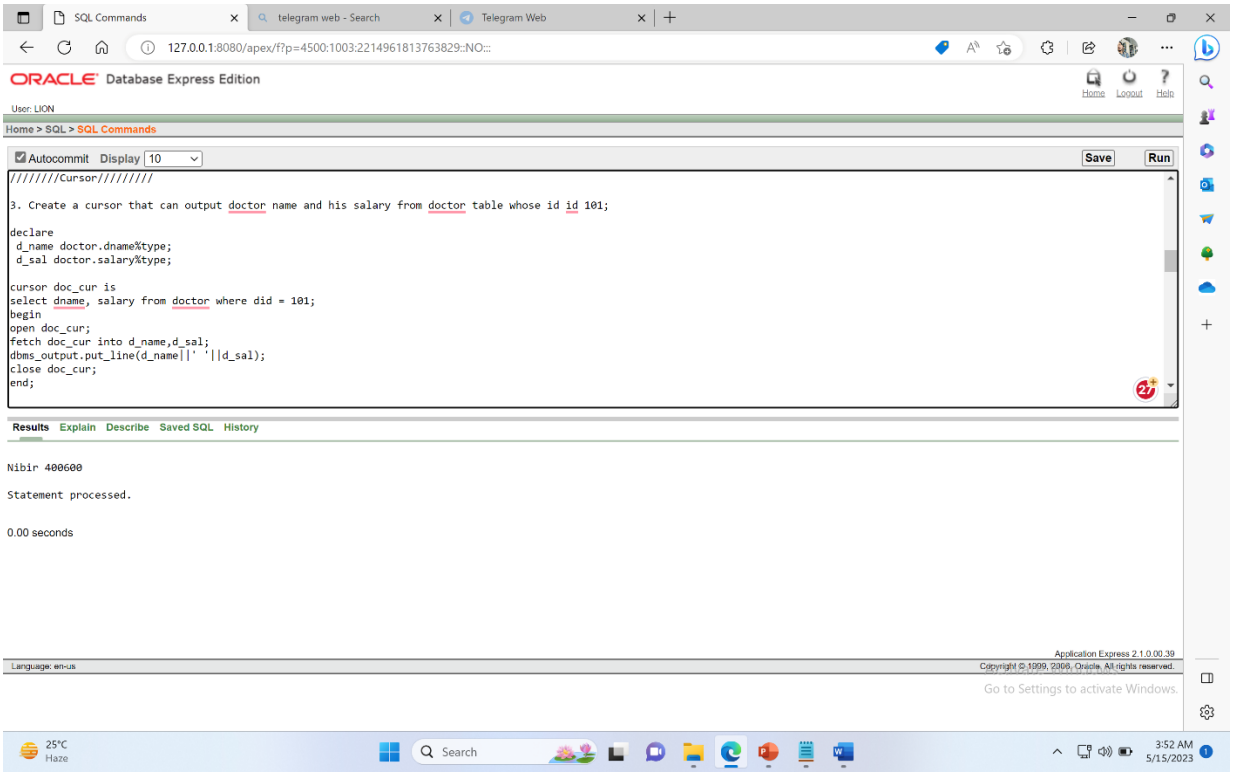
1. Create a cursor that can output patient name and their bloodgroup from patient table



2. Create an implicit cursor that can add 200$ with the current salary of all doctors and output the number of rows that are affected.

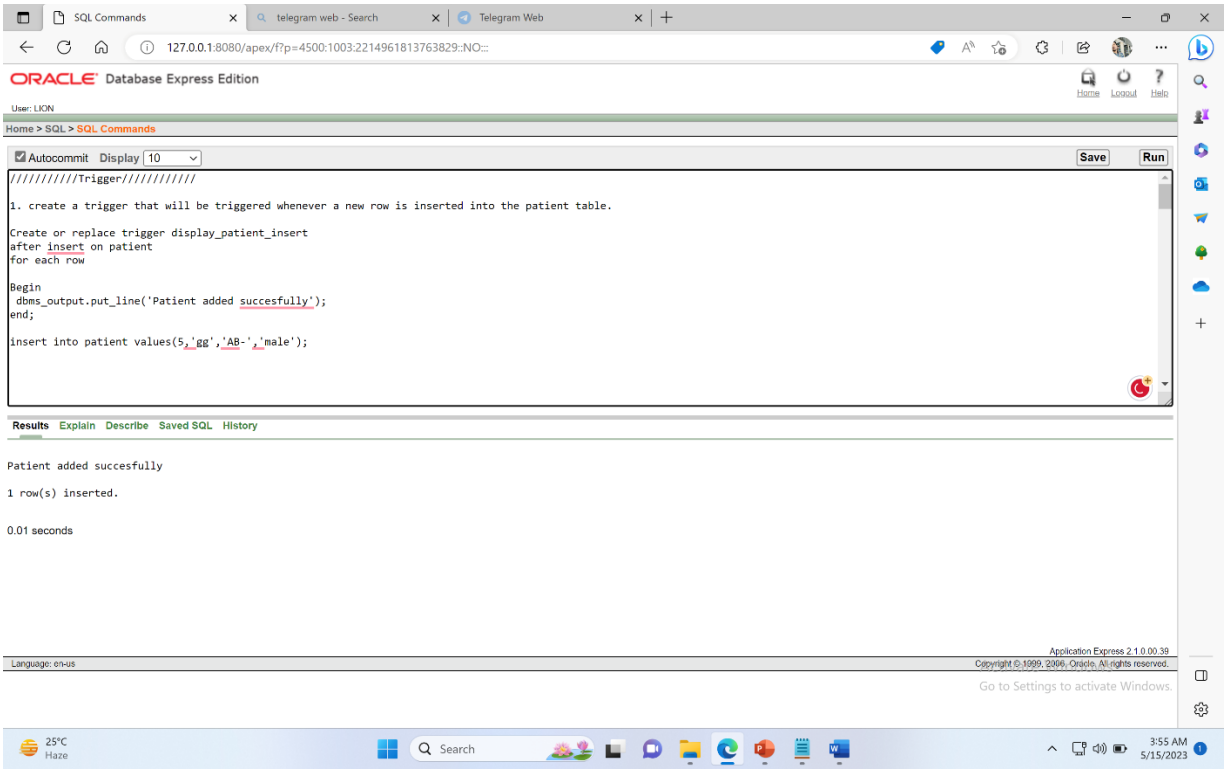


3. Create a cursor that can output doctor name and his salary from doctor table whose id 101.

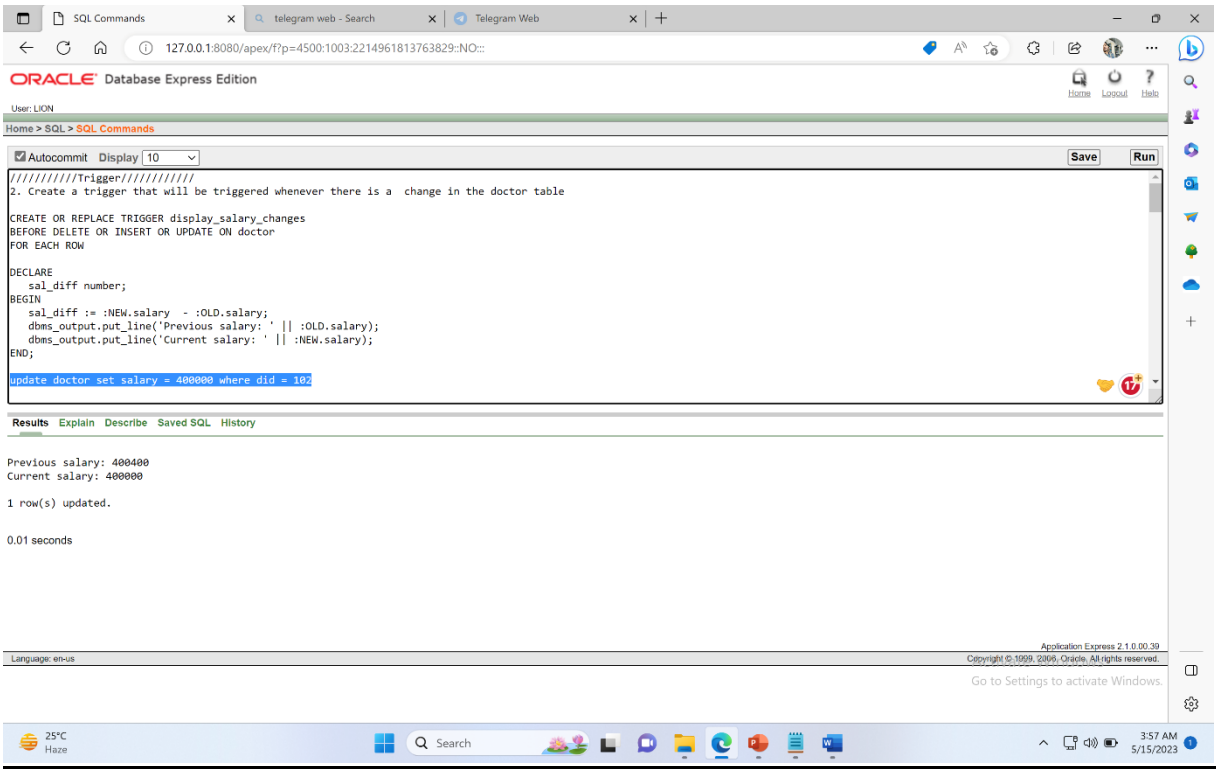


Trigger:

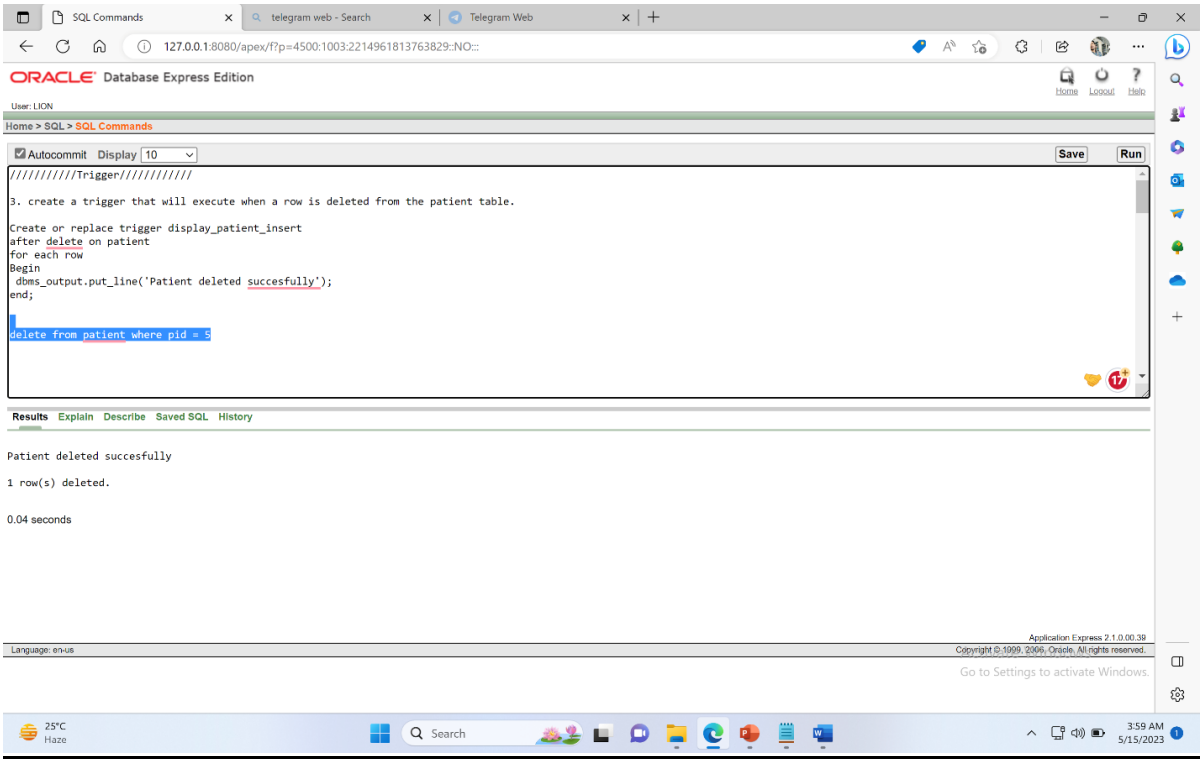
1. Create a trigger that will be triggered whenever a new row is inserted into the patient table.



2. Create a trigger that will be triggered whenever there is a change in the doctor table

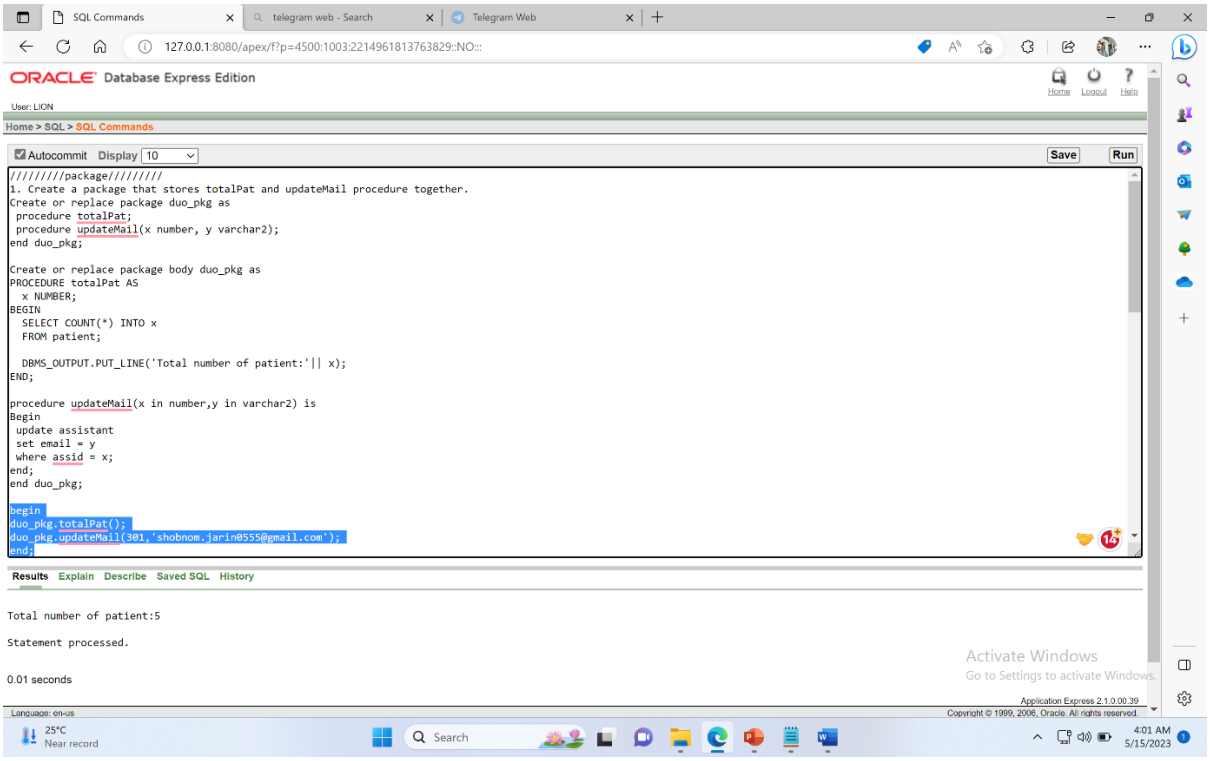


3. Create a trigger that will execute when a row is deleted from the patient table.

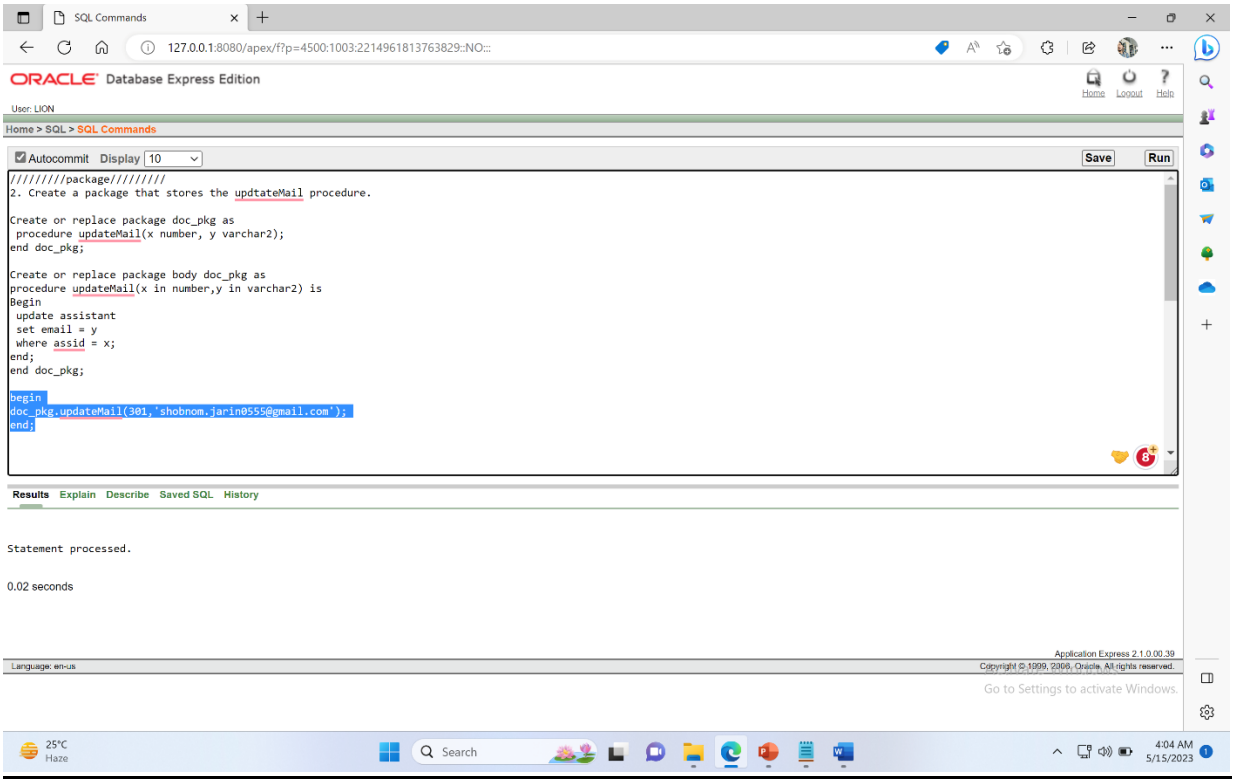


Package:

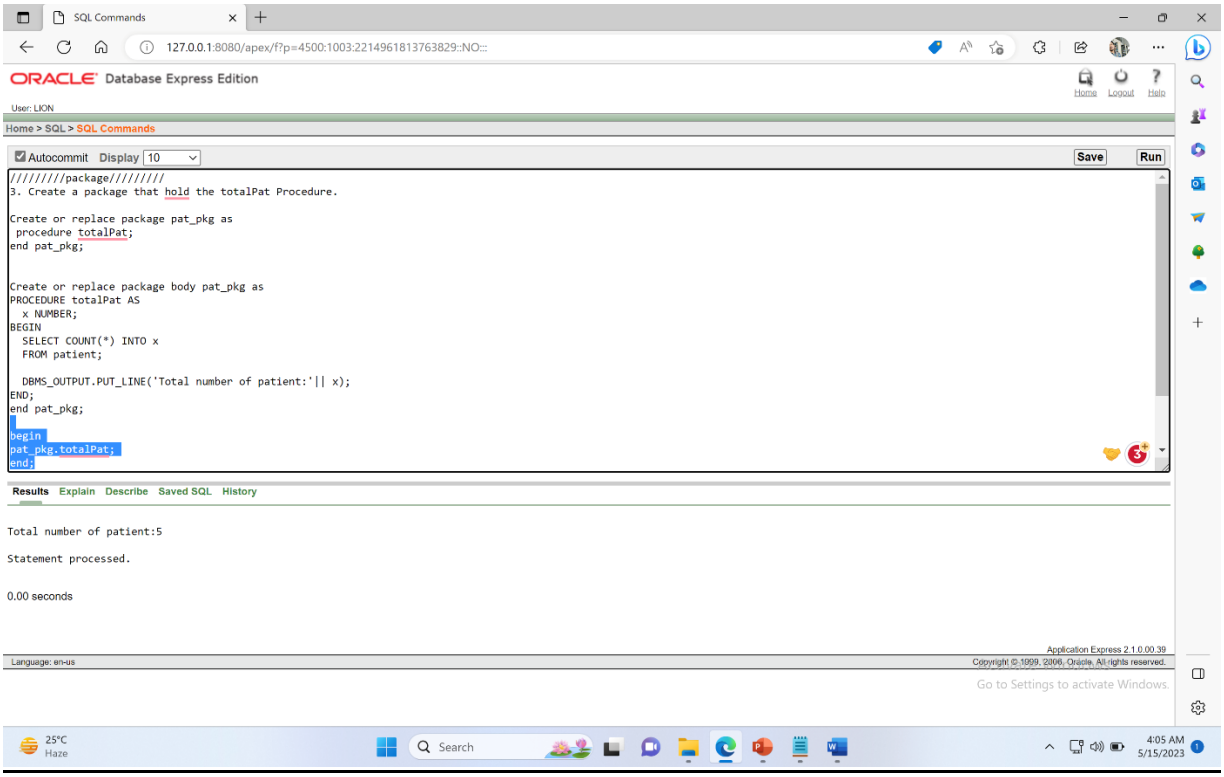
1. Create a package that stores totalPat and updateMail procedure together.



2. Create a package that stores the updtateMail procedure.



3. Create a package that hold the totalPat Procedure.



Relational Algebra:

1. Project Operation(π):
   ->To project the DoctorID and Name attributes from the Doctor table :

   πdid,dname(Doctor)

   ->To project only the Phone number attributes from the Patient Contact table:

   πPhoneno(PatientContact)

2. Selection Operation(σ):
   ->To select all the Acccountants who lives in Dhaka Division:

   σ Division='DHAKA'(Acccountant)

   ->To select all Patients name whose PID = 5002 :

   πName(σ PID=5002(Patient))

3. Cartesian Product Operation:
   ->To find all possible Appointments for each doctor :

   DoctorAppointment<-Doctor x Appointment

4. Union Operation(υ) :
   ->To combine all Appointments for June 21th,2023 and June 25th,2023 :

   Appointment_1<- σ Date='2023-06-21'(Appointment)
   Appointment_2<- σ Date='2023-06-25'(Appointment)

   Appointment_1 υ Appointment_2

5. Rename Operation (ρ) :
   ->To rename the ACCNAME attribute in the Accountant table to Accountant Name

   ρ Accountant Name/ACCNAME(Accountant)

Conclusion:

The Doctor Appointment Management System presented in this case study is an efficient and comprehensive system for managing patient appointments, medical records. The system also maintains medical records, identifies patients prescribed by doctors. The system provides a seamless approach to patient care and management, resulting in a more efficient and effective doctor appointment system.To improve this existing project, we can add several features, such as automated reminder notifications for patients about their appointments, online payment options for patients, and the ability for doctors to access medical records remotely.Finally, we can make the system scalable and adaptable to accommodate the growth of the healthcare system and future technological advancements.