



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 RPOOSAL 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

1	Introduction	4
2	Project Proposal.....	4
3	Diagram.....	5
3.1	Class Diagram	5
3.2	Use Case Diagram	6
3.3	Activity Diagram	7
4	Use Interface	8
5	Scenario Description	10
6	ER Diagram	11
7	Normalization	11
8	Schema Diagram	16
9	Table Creation	17
9.1	Screenshots of the created table	17
9.2	Queries for Sequences	29
9.3	Queries for Index Creation	31
10	Data Insertion	32
10.1	Screenshots of data insertion	32
10.2	Queries For data insertion	35
10.3	Screenshots of	43
10.3.1	Create Users	43
10.3.2	Assign Roles	42

10.3.3 Grant Privileges	42
11 SQL Query Writing	45
11.1 Single-row-function	45
11.2 Group Function	46
11.3 Subquery	47
11.4 Joining	48
11.5 View	49
11.6 Synonym	50
12 PL/SQL Query Writing	53
12.1 Function	53
12.2 Procedure	54
12.3 Record	56
12.4 Cursor	57
12.5 Trigger	59
12.6 Package	60
12 Relational Algebra	62
12.1 Project Operation.....	62
12.2 Selection Operation.....	62
12.3 Cartesian Product Operation	62
12.4 Union Operation	62
12.5 Rename Operation.....	62
13 Conclusion	663

Introduction:

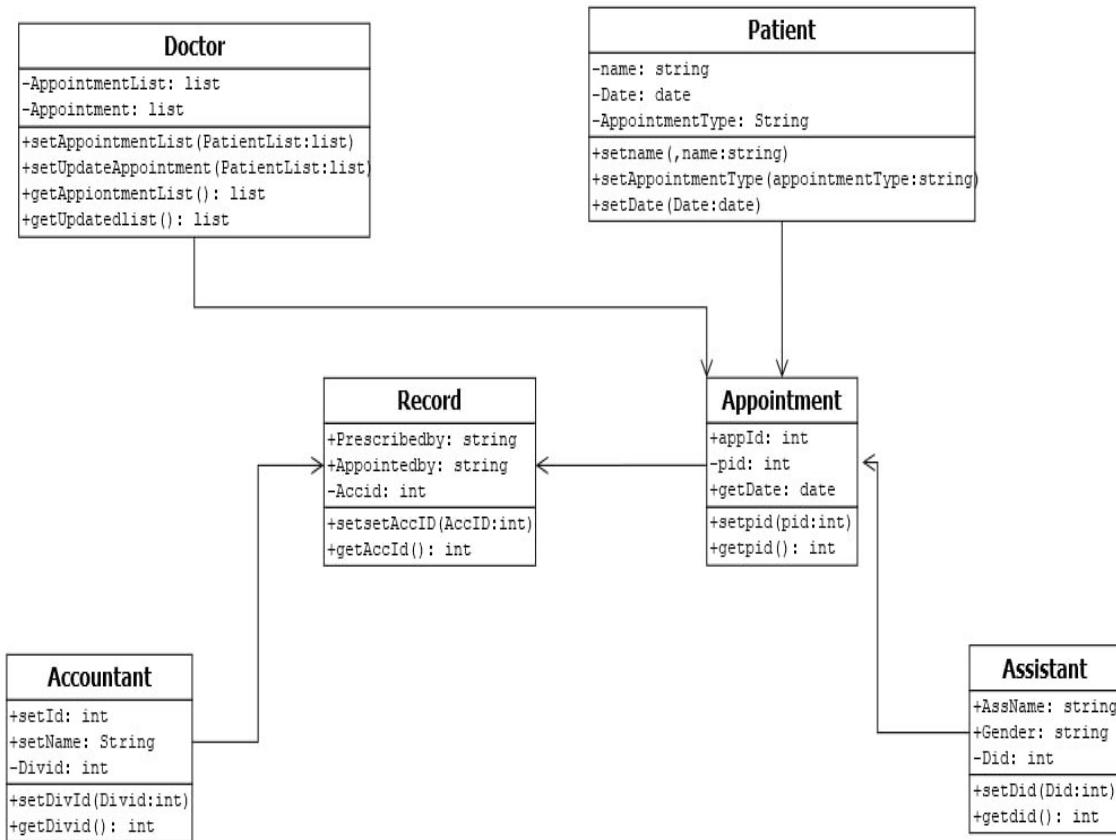
The management of healthcare systems requires efficient and accurate management of patient appointments, medical records, and healthcare providers. The database management system discussed in this case study focuses on the efficient handling of information related to a "Doctor Appointment Management 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:

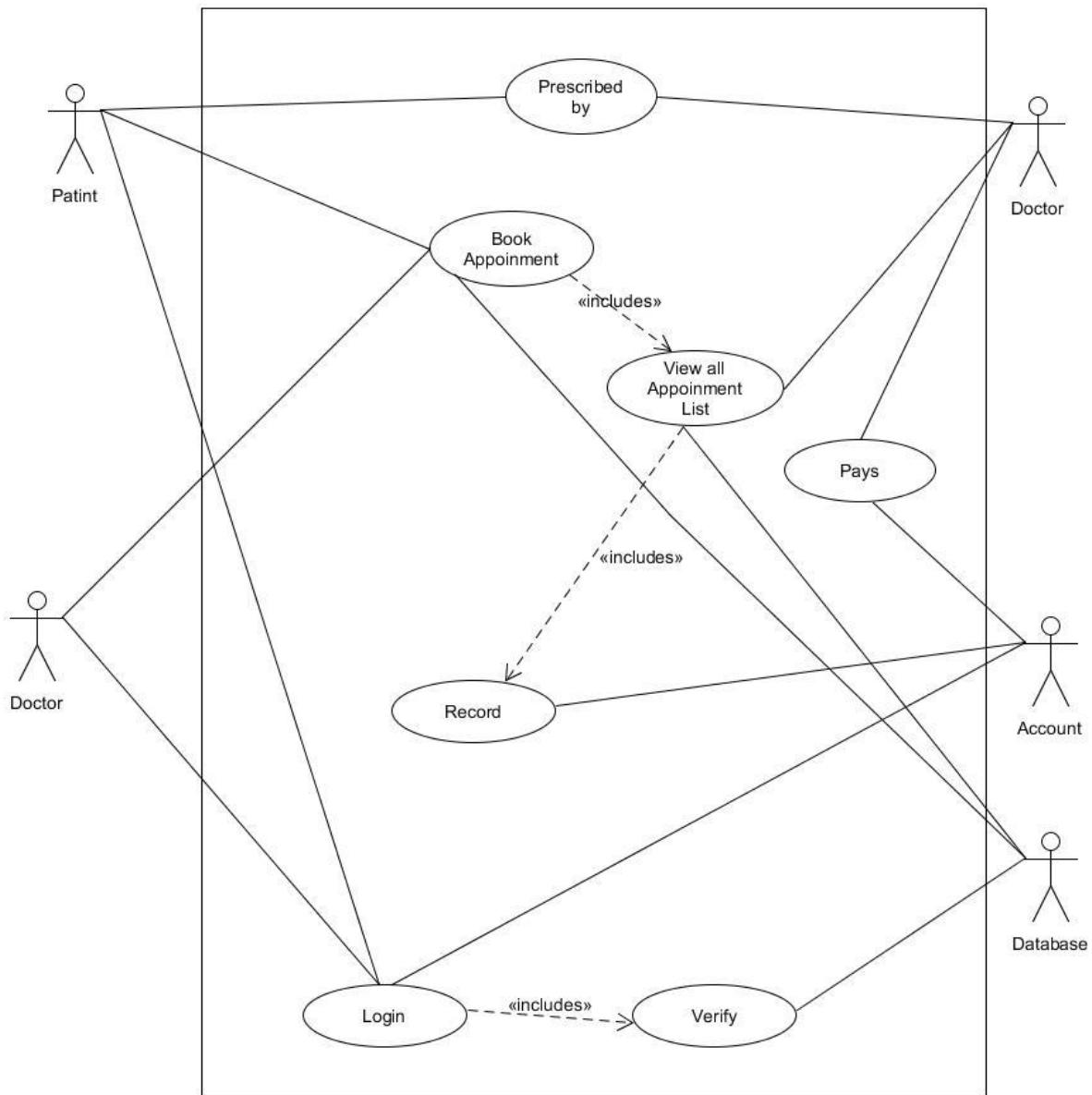
The proposed Doctor Appointment Management System aims to provide an efficient platform for patients to book appointments, doctors to manage appointments, and accountants to maintain financial records. The system includes modules for patient registration, appointment management, prescription and medical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

Diagram:

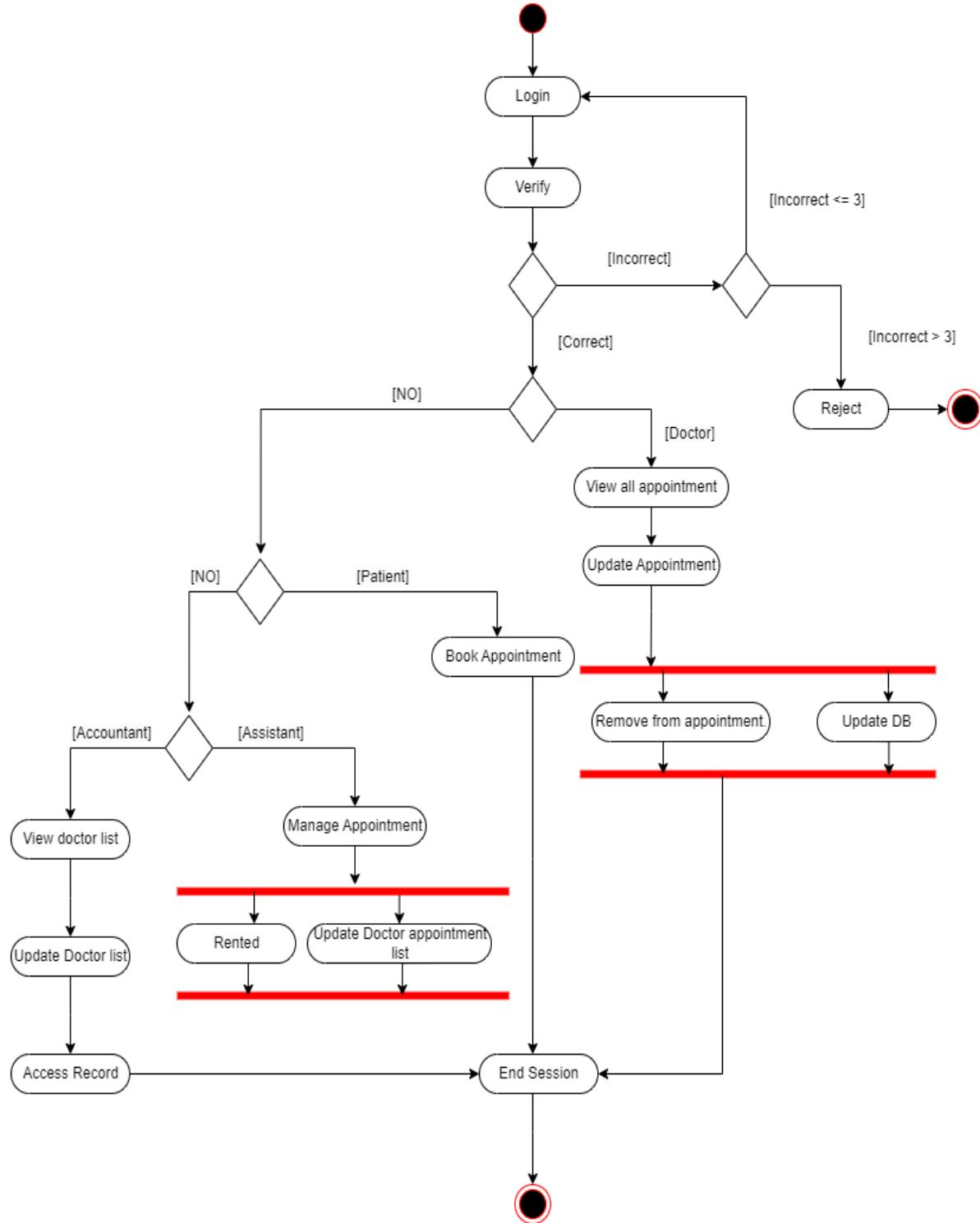
Class Diagram:



User Case Diagram:



Activity Diagram:



User Interface:

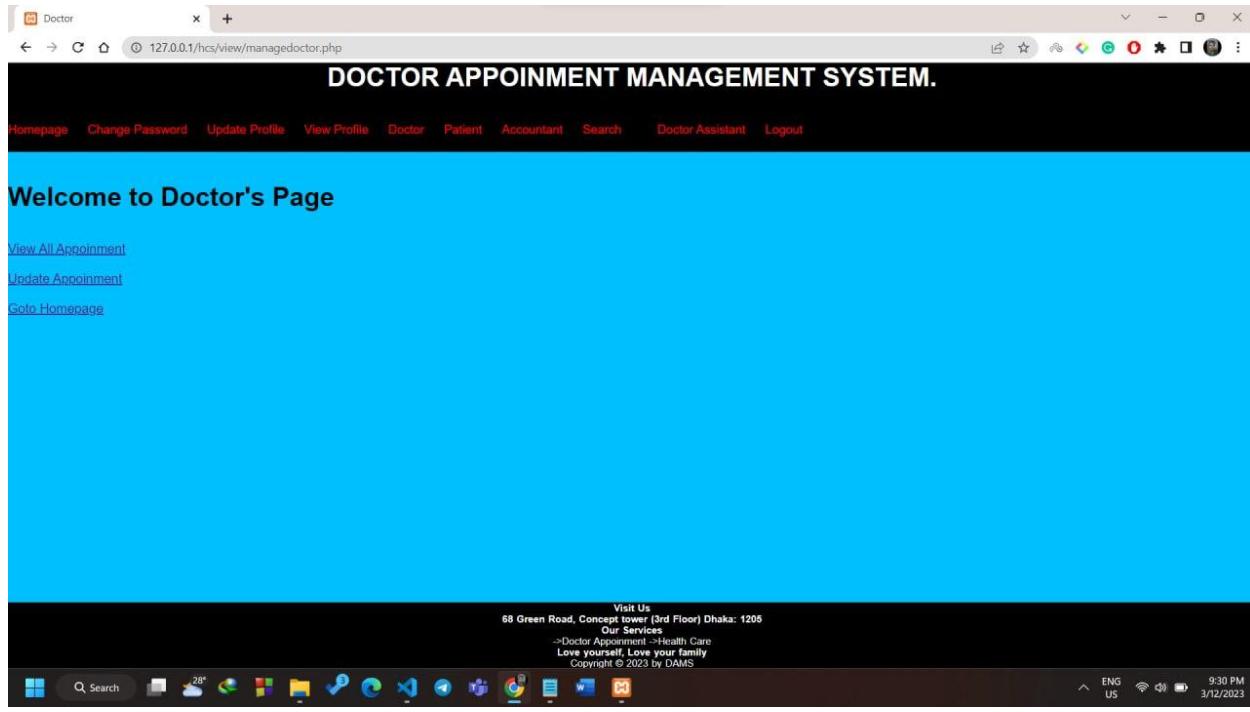
The image displays two screenshots of a web application titled "DOCTOR APPOINTMENT MANAGEMENT SYSTEM." Both screenshots are taken from a Windows operating system, showing the taskbar at the bottom.

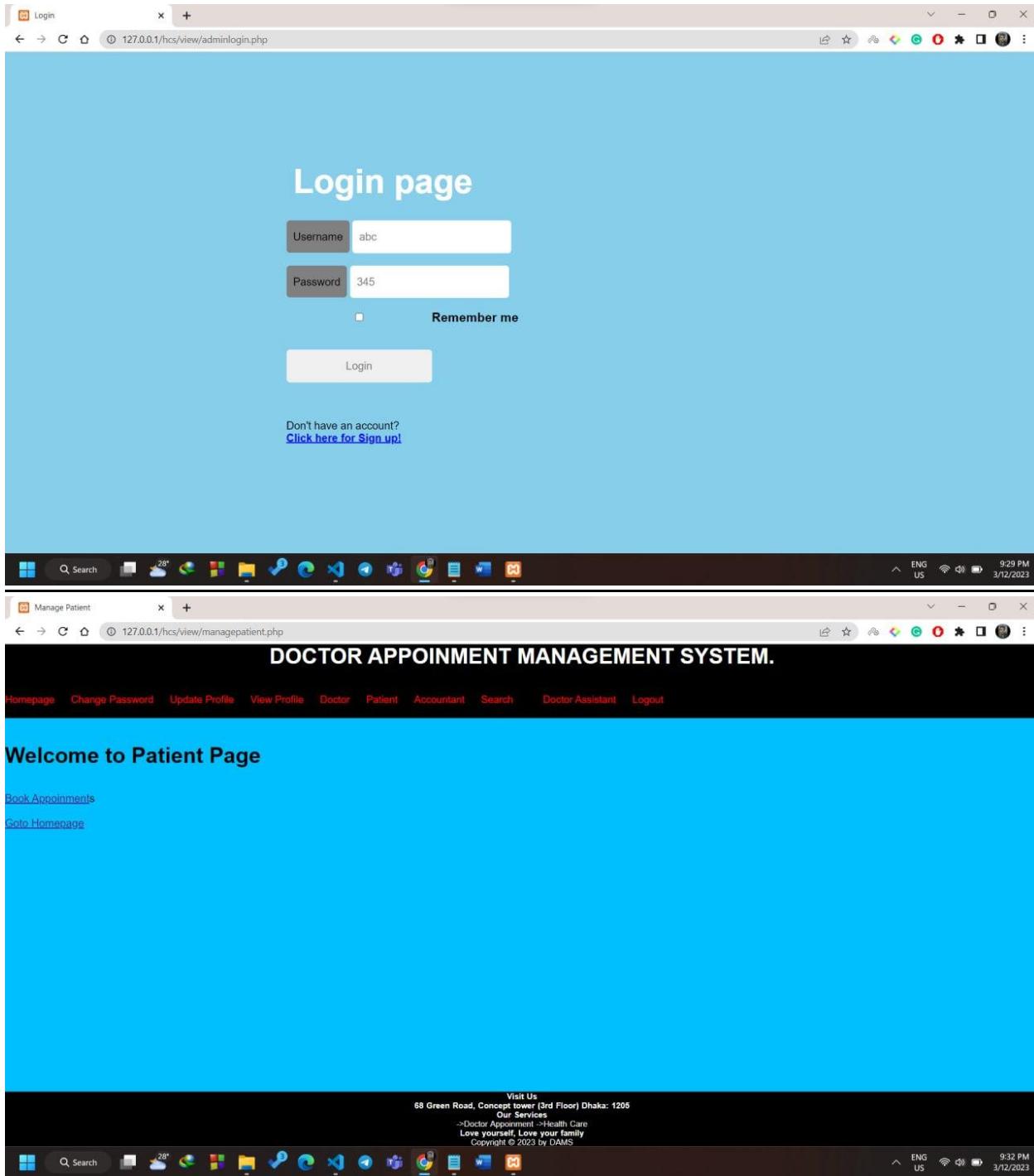
Screenshot 1: Accountant Page

- Page Title:** DOCTOR APPOINTMENT MANAGEMENT SYSTEM.
- Header:** Manage Nurse, 127.0.0.1/hcs/view/manageacc.php
- Navigation:** Homepage, Change Password, Update Profile, View Profile, Doctor, Patient, Accountant, Search, Doctor Assistant, Logout
- Content:** Welcome to Accountant Page
- Links:** View All Doctor, Update Doctor List, Access record, Goto Homepage
- Footer:** Visit Us, 68 Green Road, Concept tower (3rd Floor) Dhaka: 1205, Our Services, >Doctor Appointment, >Health Care, Love yourself, Love your family, Copyright © 2023 by DAMS
- Taskbar:** Shows various pinned icons like File Explorer, Microsoft Edge, and File Manager. The date and time are 3/12/2023, 9:33 PM. Language settings are ENG US.

Screenshot 2: Doctor Assistant Page

- Page Title:** DOCTOR APPOINTMENT MANAGEMENT SYSTEM.
- Header:** Doctor, 127.0.0.1/hcs/view/assistant.php
- Navigation:** Homepage, Change Password, Update Profile, View Profile, Doctor, Patient, Accountant, Search, Doctor Assistant, Logout
- Content:** Welcome to Doctor Assistant Page
- Links:** Manage Appointment, Goto Homepage
- Footer:** Visit Us, 68 Green Road, Concept tower (3rd Floor) Dhaka: 1205, Our Services, >Doctor Appointment, >Health Care, Love yourself, Love your family, Copyright © 2023 by DAMS
- Taskbar:** Shows various pinned icons like File Explorer, Microsoft Edge, and File Manager. The date and time are 3/12/2023, 9:33 PM. Language settings are ENG US.



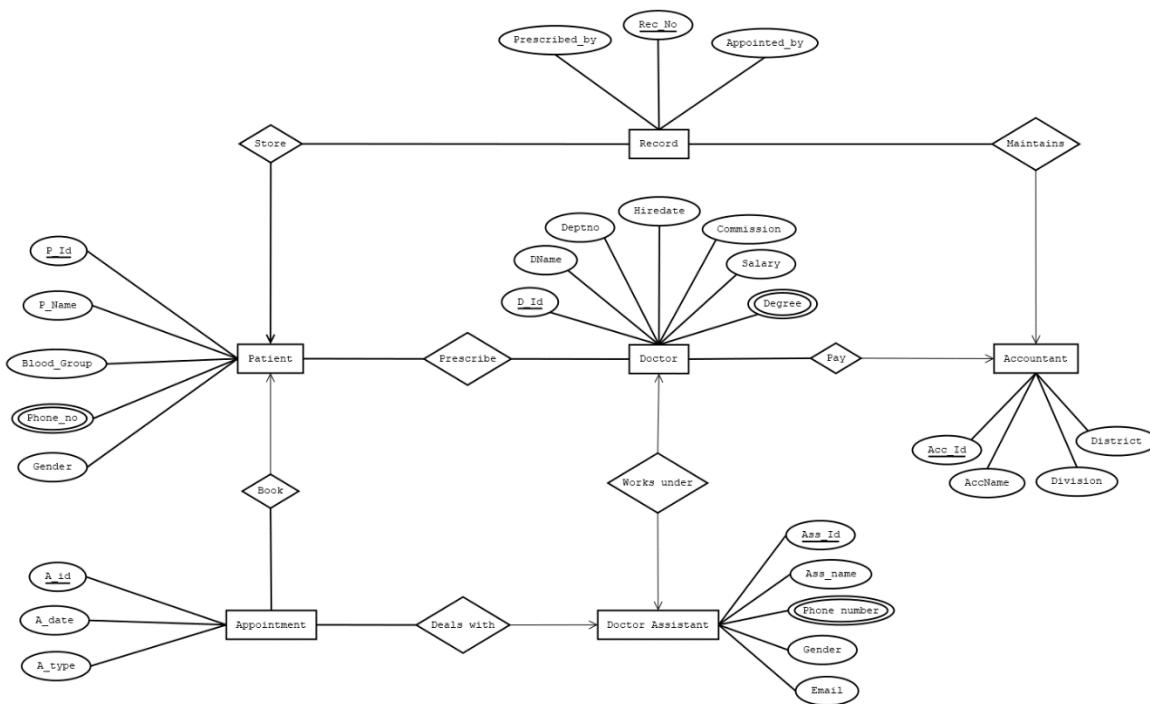


Scenario Description:

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**, Prescribedby, Appointed_by, **P_id**, PName, Blood_Group, Phone_no, Gender)

1NF: Phone_no is a multivalued attribute

2NF: **Rec_no**, Appointed_by, Prescribedby

P_id, PName, Blood_Group, Phone_no, Gender

3NF: No transitive dependency found

Rec_no, Appointed_by, Prescribedby

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, 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~~
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, deptno, 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, Prescribedby, Appointed_by, Acc_id~~
23. ~~Rec_no, Prescribedby, 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, Prescribedby, Appointed_by, P_id, Acc_id --Record~~

Schema Diagram:

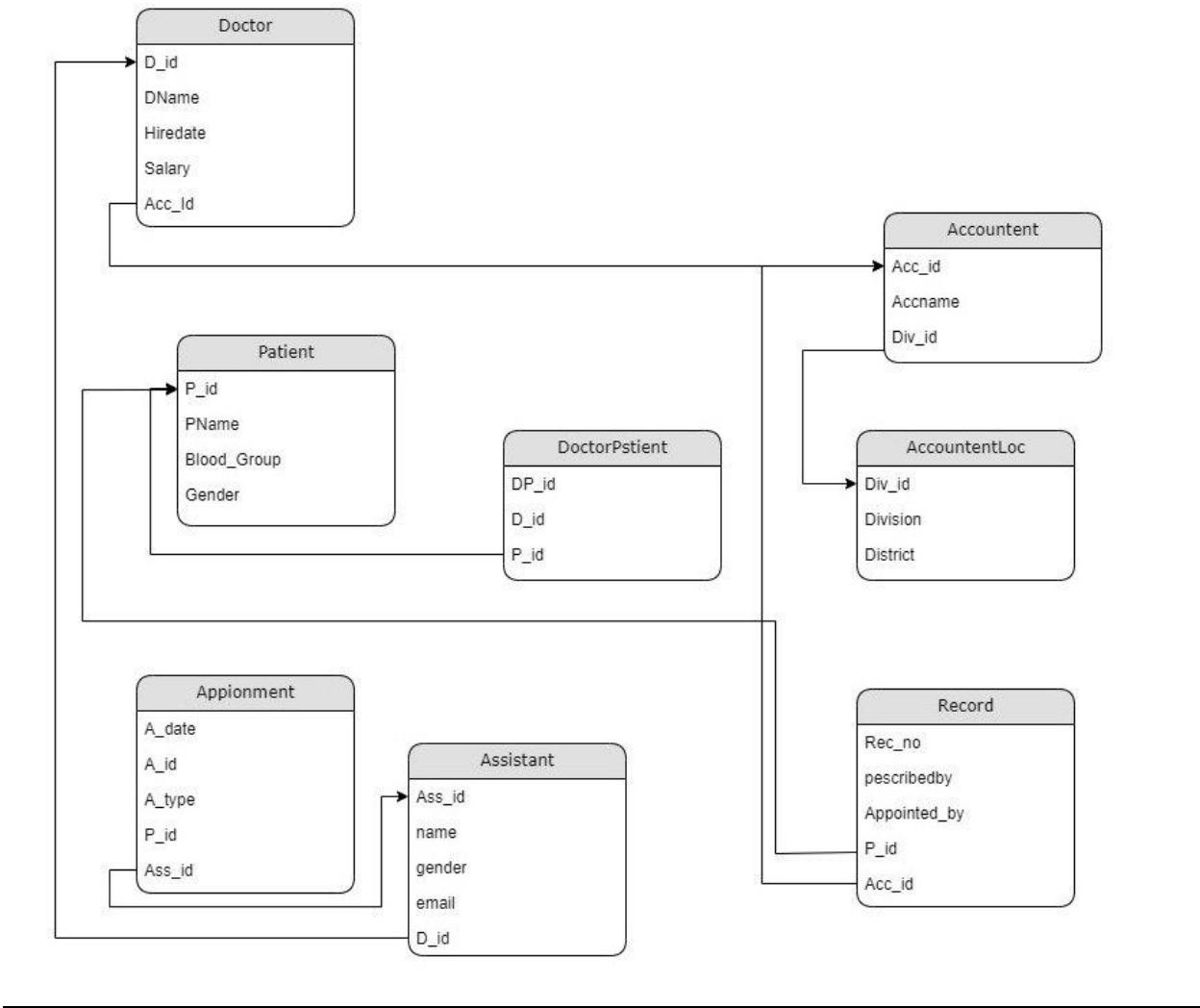


Table Creation:

Doctor Table:

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor window contains the following code:

```
//doctor
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)
desc doctor
```

Below the code, there is a table titled "Object Type TABLE Object DOCTOR" showing the column definitions:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DOCTOR	Did	Number	-	-	0	1	-	-	-
	DNAME	Varchar2	50	-	-	-	✓	-	-
	HIREDATE	Date	7	-	-	-	✓	-	-
	SALARY	Float	22	126	-	-	✓	-	-
	DEPTNO	Number	-	-	0	-	✓	-	-
	COMM	Float	22	126	-	-	✓	-	-
	ACCID	Number	-	-	0	-	✓	-	-

The status bar at the bottom right indicates "Activate Windows Go to Settings to activate Windows".

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:

The screenshot shows the Oracle Database Express Edition interface. In the top navigation bar, there are tabs for 'SQL Commands' and 'Index'. Below the navigation bar, the URL is 127.0.0.1:8080/apex/f?p=4500:1003:7473850913643872::NO:::.

In the main content area, there is a section titled 'ORACLE® Database Express Edition' with a sub-section 'User: LION'. Below this, the 'Home > SQL > SQL Commands' path is shown. A checkbox for 'Autocommit' is checked, and the 'Display' dropdown is set to 10. There are 'Save' and 'Run' buttons at the top right of the code editor.

The SQL code entered is:

```
//DoctorDegree
create table Doctordegree(
Did int,
Degree varchar2(50))

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

desc Doctordegree
```

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Describe' tab is selected. The object type is listed as 'TABLE Object DOCTORDEGREE'. A detailed table of columns is provided:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DOCTORDEGREE	Did	Number	-	0	1	-	-	-	-
DOCTORDEGREE	Degree	Varchar2	50	-	-	2	-	-	-

At the bottom of the page, it says 'Language: en-us' and 'Application Express 2.1.0 00.39 Copyright © 1999, 2006, Oracle. All rights reserved.'

create table Doctordegree(

Did int,

Degree varchar2(50))

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

Patient Table:

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered:

```
//Patient
create table patient(
pid int,
pname varchar2(50),
BloodGroup varchar2(10),
Gender varchar2(15))

alter table patient add constraint p_pk primary key(pid)

Desc patient
```

Below the code, the results of the 'DESCRIBE' command are shown:

Object Type	TABLE Object	PATIENT							
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PATIENT	PID	Number	-	-	0	1	-	-	-
	PNAME	Varchar2	50	-	-	-	✓	-	-
	BLOODGROUP	Varchar2	10	-	-	-	✓	-	-
	GENDER	Varchar2	15	-	-	-	✓	-	-

At the bottom of the interface, it says "Language: en-us" and "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved."

```
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:

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands section, the following SQL code is entered:

```
//PatientContact
create table patientContact(
pid int,
phoneno varchar2(15))

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

desc patientcontact
```

In the Object Type section, the details for the TABLE Object PATIENTCONTACT are displayed in a table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PATIENTCONTACT	PID	Number	-	-	0	1	-	-	-
	PHONENO	Varchar2	15	-	-	2	-	-	-

At the bottom, the system status bar shows "Language: en-us", "Application Express 2.1.0.00.39", "Copyright © 1999, 2006, Oracle. All rights reserved.", and the system tray includes icons for search, file explorer, task manager, mail, and other system utilities.

```
create table patientContact(
pid int,
phoneno varchar2(15))

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

Doctor Patient Table:

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered:

```
//DoctorPatient
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)

desc doctorpatient
```

The results section shows the table structure:

Object Type	TABLE Object	DOCTORPATIENT							
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DOCTORPATIENT	DPID	Number	-	-	0	1	-	-	-
	DID	Number	-	-	0	-	✓	-	-
	PID	Number	-	-	0	-	✓	-	-

At the bottom, the status bar indicates "Language: en-us" and "Copyright © 1999, 2006, Oracle. All rights reserved".

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)

Appointment Table:

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered:

```
//Appointment
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)

desc appointment;
```

The results section displays the table structure:

Object Type	TABLE Object	APPPOINTMENT							
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
APPOINTMENT	APPID	Number	-	-	0	1	-	-	-
	APPPDATE	Date	7	-	-	-	✓	-	-
	APPTYPE	VARCHAR2	50	-	-	-	✓	-	-
	PID	Number	-	-	0	-	✓	-	-
	ASSID	Number	-	-	0	-	✓	-	-

At the bottom, the status bar shows "Language: en-us" and "Copyright © 1999, 2006, Oracle. All rights reserved." The system tray indicates the date and time as "12:33 AM 3/9/2023".

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:

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered:

```
//Assistant
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)

desc assistant
```

The results tab shows the table structure:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ASSISTANT	ASSID	Number	-	-	0	1	-	-	-
	ASSNAME	VARCHAR2	50	-	-	-	✓	-	-
	GENDER	VARCHAR2	15	-	-	-	✓	-	-
	EMAIL	VARCHAR2	50	-	-	-	✓	-	-
	DID	Number	-	-	0	-	✓	-	-

Language: en-us Application Express 2.1.0.0.39
Copyright © 1999, 2006, Oracle. All rights reserved.

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:

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered:

```
//AssistantContact
Create table assistantcontact(
Assid int,
phoneno varchar2(15))

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

desc assistantcontact
```

The results section shows the table structure:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ASSISTANTCONTACT	ASSID	Number	-	-	0	1	-	-	-
	PHONENO	Varchar2	15	-	-	2	-	-	-

At the bottom, the status bar indicates "Language: en-us" and "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved."

Create table assistantcontact(

Assid int,

phoneno varchar2(15))

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

Accountant Table:

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following code:

```
//Accountant
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)

desc accountant
```

The results pane shows the object type as TABLE Object and the name as ACCOUNTANT. The table structure is as follows:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ACCOUNTANT	ACCID	Number	-	-	0	1	-	-	-
	ACCNAME	VARCHAR2	50	-	-	-	✓	-	-
	DIVID	Number	-	-	0	-	✓	-	-

At the bottom, it says Language: en-us and Application Express 2.1 0 0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

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:

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered:

```
//AccountantLoc
Create table AccountantLoc(
divid int,
division varchar2(50),
district varchar2(50))

alter table AccountantLoc add constraint accloc_pk primary key(divid)

desc AccountantLoc
```

Below the code, the results of the `desc AccountantLoc` command are displayed, showing the table structure:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ACCOUNTANTLOC	DIVID	Number	-	-	0	1	-	-	-
	DIVISION	VARCHAR2	50	-	-	-	✓	-	-
	DISTRICT	VARCHAR2	50	-	-	-	✓	-	-

At the bottom of the interface, the status bar shows "Language: en-us" and "Copyright © 1999, 2006, Oracle. All rights reserved." The system tray icons include a search bar, file, mail, taskbar, and system status.

Create table AccountantLoc(

divid int,

division varchar2(50),

district varchar2(50))

alter table AccountantLoc add constraint accloc_pk primary key(divid)

Record Table:

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands section, the following SQL code is entered:

```
//record
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)

desc record
```

In the Object Types section, the TABLE Object RECORD is selected. A table shows the columns and their properties:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RECORD	RECN0	Number	-	-	0	1	-	-	-
	PRESCRIBEDBY	Varchar2	50	-	-	-	✓	-	-
	APPOINTEDBY	Varchar2	50	-	-	-	✓	-	-
	PID	Number	-	-	0	-	✓	-	-
	ACCID	Number	-	-	0	-	✓	-	-

At the bottom, the status bar indicates "Language: en-us" and "Copyright © 1999, 2006, Oracle. All rights reserved." The system tray shows the date and time as "10:04 PM 3/11/2023".

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

necycle

//Assistant(SEQ_301)

create sequence ASTseq

increment by 1

start with 301

maxvalue 400

nocache

necycle

//Accountant(SEQ_1)

create sequence ACCseq

increment by 5

start with 5

maxvalue 100

nocache

necycle

//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 assistant_idx
```

```
on assistant(assname,email)
```

```
//accountant
```

```
Create index accountant_idx  
on accountant(accid,accname)
```

Data insertion:

The screenshot shows the Oracle Database Express Edition SQL Commands window. The code entered is:

```
//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,'Foyosal',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)  
select * from doctor
```

The results section displays the inserted data:

ID	DNAME	HIREDATE	SALARY	DEPTNO	COMM	ACCID
101	Nibir	22-JUN-22	400000	10	0	15
102	Shadhin	16-DEC-20	400000	30	5000	5
103	Foyosal	31-DEC-19	400400	20	3000	5
104	Tuhin	20-NOV-19	500400	10	6000	20
105	Nitu	31-DEC-19	600400	30	-	20

5 rows returned in 0.02 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.0.39
Copyright © 1999, 2006, Oracle. All rights reserved.
Go to Settings to activate Windows.

```
//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,'Foyosal',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)
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

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')

select * from doctordegrees

```

The results of the query are displayed in a table:

DID	DEGREE
101	FCPS
101	MBBS
102	MBBS
103	MBBS
104	MBBS
105	MBBS

6 rows returned in 0.02 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

//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')

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

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')

select* from patient;

```

The results show a table with the following data:

PID	PNAME	BLOODGROUP	GENDER
5001	Adnan	A+	Male
5002	Abir	O+	Male
5003	Farid	B-	Male
5004	Akhi	AB+	Female
5005	Tanzila	B+	Female

5 rows returned in 0.00 seconds [CSV Export](#)

Application Express 2.1 0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

//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')

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL in the address bar is 127.0.0.1:8080/apex/f?p=4500:1003:7473850913643872::NO:::. The page title is ORACLE® Database Express Edition. The user is LION.

The SQL code entered is:

```

 Autocommit   
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')
select* from patientcontact

```

The results show the inserted data:

PID	PHONENO
5001	+8801954775770
5002	+8801754775771
5002	+8801850201030
5003	+8801713575873
5004	+8801973575873

5 rows returned in 0.00 seconds [CSV Export](#)

Application Express 2.1 0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.
Language: en-us

//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')

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:7473850913643872::NO:::. The page title is ORACLE® Database Express Edition. The user is LION.

The SQL code entered is:

```
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)

select* from DoctorPatient
```

The results show a table with columns DPID, DID, and PID. The data is:

DPID	DID	PID
6001	101	5001
6002	101	5002
6003	102	5003
6004	103	5004
6005	104	5005

5 rows returned in 0.00 seconds. There is a CSV Export link.

At the bottom, it says Language: en-us, Application Express 2.1 0 00.39, Copyright © 1999, 2006, Oracle. All rights reserved.

//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)
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

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)

select * from appointment

```

The results section displays the inserted data:

APPID	APPPDATE	APPTYPE	PID	ASSID
7001	22-JUN-22	Cancer	5001	301
7002	25-JUN-23	Cancer	5002	301
7003	21-JUN-23	Influenza	5003	302
7004	26-JUN-23	Aids	5004	303
7005	29-JUN-23	Common cold	5005	304

5 rows returned in 0.02 seconds [CSV Export](#)

Application Express 2.1 0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

// 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)
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

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)

select* from Assistant

```

The results section displays the following table:

ASSID	ASSNAME	GENDER	EMAIL	DID
301	Shyama	Female	shyma11@gmail.com	101
302	Shishir	Female	shishirpharma@gmail.com	102
303	Abida	Female	abida0@gmail.com	103
304	Mubin	Male	mubin211@gmail.com	104
305	Asha	Female	asha89@gmail.com	105

5 rows returned in 0.00 seconds [CSV Export](#)

Application Express 2.1 0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

//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)

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following code:

```

 Autocommit Display 10
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')

select * from assistantcontact

```

The results section displays the following table:

ASSID	PHONENO
301	+8801647390996
302	+8801950120661
303	+8801680706080
304	+8801445450985
305	+8801952528457

5 rows returned in 0.00 seconds [CSV Export](#)

Application Express 2.1 0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

//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')

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

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)

SELECT * FROM ACCOUNTANT

```

The results section displays a table with the following data:

ACCID	ACCNAME	DIVID
5	Naim	1201
10	Anik	1202
15	Siam	1203
20	Anika	1204
25	Anjum	1205

5 rows returned in 0.00 seconds

CSV Export

Application Express 2.1 0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

//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)

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

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')

select *
from accountantloc

```

The results show an 8-row table:

DIVID	DIVISION	DISTRICT
1201	DHAKA	TANGAIL
1202	BARISHAL	BHOLA
1203	MYMENSINGH	JAMALPUR
1204	CHATTAGRAM	NOAKHALI
1205	KHULNA	JESSORE
1206	MYMENSINGH	SHERPUR
1207	MYMENSINGH	NETROKONA
1208	DHAKA	KISARGONJ

8 rows returned in 0.00 seconds [CSV Export](#)

//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')

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

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)

select * from record

```

The results section displays a table with the following data:

RECCNO	PRESCRIBEDBY	APPOINTEDBY	PID	ACCID
8001	Nibir	Shyama	5001	15
8002	Nibir	Shyama	5002	15
8003	Shadin	Shishir	5003	5
8004	Foysal	Abida	5004	5
8005	Tuhin	Mubin	5005	20

5 rows returned in 0.00 seconds [CSV Export](#)

Application Express 21.0.0.39
Copyright © 1999, 2006, Oracle. All rights reserved.

//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.

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following commands:

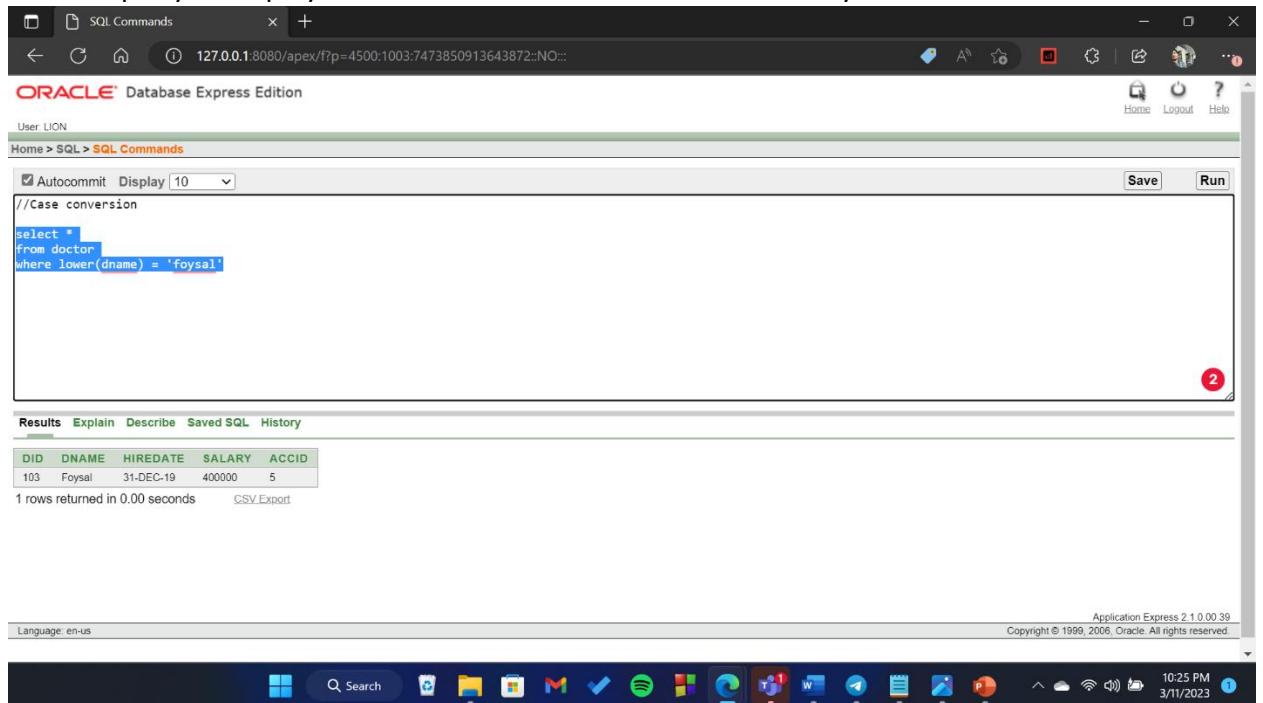
```
CREATE ROLE administrator
GRANT ALL PRIVILEGES ON accountant TO administrator
CREATE USER Ekramul IDENTIFIED BY administrator
GRANT administrator TO Ekramul
```

Below the SQL editor, the message "Statement processed." is displayed, along with execution time "0.00 seconds". At the bottom, the Application Express footer includes the version "Application Express 2.1 0 00.39", copyright information "Copyright © 1999, 2006, Oracle. All rights reserved.", and system status icons.

SQL Query Writing:

Single row function:

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



The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window displays the following query:

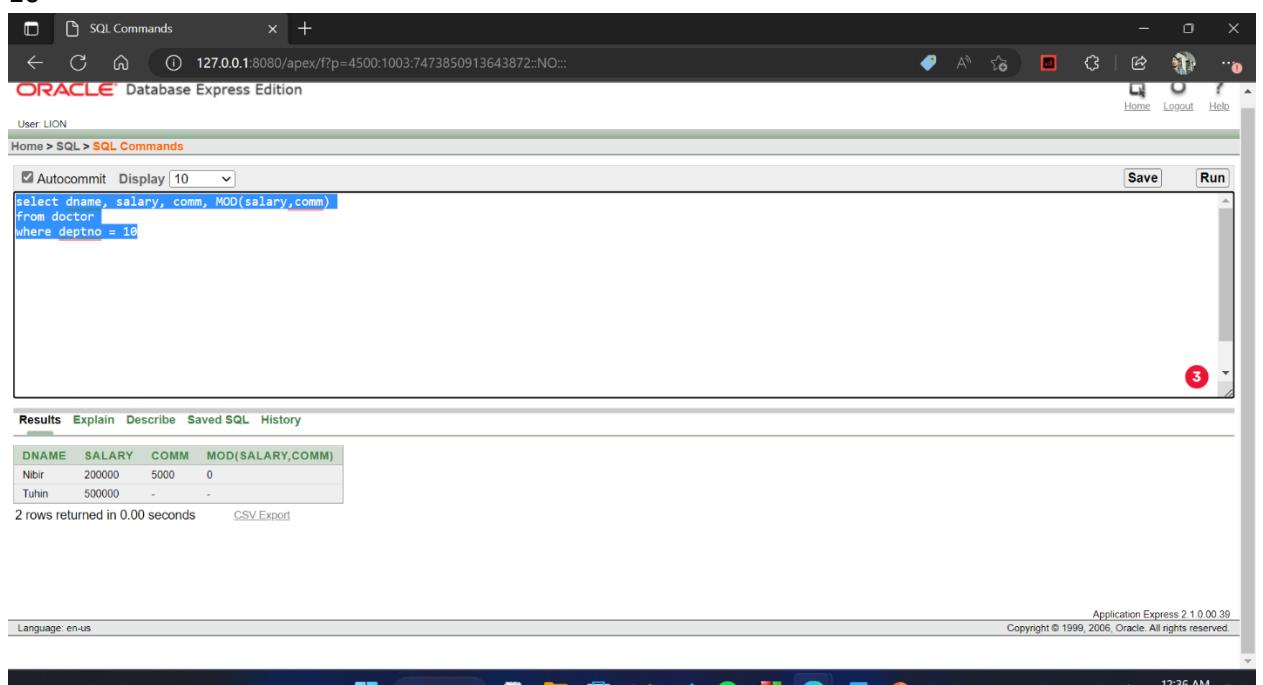
```
//Case conversion
select *
from doctor
where lower(dname) = 'foysal'
```

The results show one row:

DID	DNAME	HIREDATE	SALARY	ACCID
103	Foysal	31-DEC-19	400000	5

1 rows returned in 0.00 seconds

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



The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window displays the following query:

```
select dname, salary, comm, MOD(salary,comm)
from doctor
where deptno = 10
```

The results show two rows:

DNAME	SALARY	COMM	MOD(SALARY,COMM)
Nibir	200000	5000	0
Tuhin	500000	-	-

2 rows returned in 0.00 seconds

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.

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL in the browser is 127.0.0.1:8080/apex/f?p=4500:1003:7473850913643872::NO:::. The page title is SQL Commands. The user is LION. The navigation bar includes Home > SQL > SQL Commands. A toolbar has Autocommit checked and a Display dropdown set to 10. The SQL query entered is:

```
select deptno, max(salary)
from doctor
group by deptno
having max(salary)>30000
```

The results section shows a table with the following data:

DEPTNO	MAX(SALARY)
30	600000
20	400000
10	500000

3 rows returned in 0.00 seconds. There is a CSV Export link. The bottom status bar shows Language: en-us, Application Express 2.1.0.0.39, Copyright © 1999, 2006, Oracle. All rights reserved, and a system tray with icons for file, search, and system status.

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

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```
select count(*)
from doctor
where deptno = 20;
```

The results section displays:

COUNT(*)
2

1 rows returned in 0.00 seconds

At the bottom right of the interface, there is a red circle with the number '1' indicating a pending notification.

System tray icons are visible at the bottom of the screen, including a battery icon showing 10:48 PM and 3/11/2023.

3. Display the maximum average salary

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```
select max(avg(comm))
from doctor
group by deptno;
```

The results section displays:

MAX(AVG(COMM))
5000

1 rows returned in 0.00 seconds

At the bottom right of the interface, there is a green circle with the letter 'G' indicating a pending notification.

System tray icons are visible at the bottom of the screen, including a battery icon showing 10:58 PM and 3/11/2023.

Subquery:

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

The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window contains the following query:

```
select dname,hiredate
from doctor
where hiredate < (select hiredate
from doctor
where dname = 'Shadhin')
```

The results show three rows:

DNAME	HIREDATE
Foyal	31-DEC-19
Tuhin	20-NOV-19
Nitu	31-DEC-19

3 rows returned in 0.00 seconds

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved. 11:06 PM 3/11/2023

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

The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window contains the following query:

```
select dname,salary
from doctor
where salary in (
select min(salary)
from doctor
group by deptno)
```

The results show four rows:

DNAME	SALARY
Nitu	60000
Shadhin	300000
Nibir	200000

3 rows returned in 0.00 seconds

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved. 11:12 PM 3/11/2023

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?

The screenshot shows the Oracle Database Express Edition interface. The URL in the address bar is 127.0.0.1:8080/apex/f?p=4500:1003:7473850913643872::NO::.

The SQL command entered is:

```
select dname,salary,deptno
from doctor
where salary > any(select salary
from doctor
where deptno = 20) and deptno<>20;
```

The results table shows two rows:

DNAME	SALARY	DEPTNO
Tuhin	500000	10
Nitu	600000	30

2 rows returned in 0.00 seconds

CSV Export

Language: en-us Application Express 2.1 0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

Joining:

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

The screenshot shows the Oracle Database Express Edition interface. The URL in the address bar is 127.0.0.1:8080/apex/f?p=4500:1003:7473850913643872::NO::.

The SQL command entered is:

```
//select the name of doctor who lives in dhaka
select accid,accname
from accountant,accountantloc
where division = 'DHAKA' and accountant.divid = accountantloc.divid
```

The results table shows one row:

ACCID	ACCNAME
5	Naim

1 rows returned in 0.00 seconds

CSV Export

Language: en-us Application Express 2.1 0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

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

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```
//Display doctor's name and their degree using outer join
select dname, degree
from doctor, doctordegree
where doctor.did (+)= doctordegree.did;
```

The results table shows the following data:

DNAME	DEGREE
Nibir	FCPS
Nibir	MBBS
Shadhin	MBBS
Foysal	MBBS
Tuhin	MBBS
Nitu	MBBS

6 rows returned in 0.00 seconds

CSV Export

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

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

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```
//Display doctor's name and their degree who is not FCPS using Natural join
select dname, degree
from doctor natural join doctordegree
where degree <> 'FCPS';
```

The results table shows the following data:

DNAME	DEGREE
Nibir	MBBS
Shadhin	MBBS
Foysal	MBBS
Tuhin	MBBS
Nitu	MBBS

5 rows returned in 0.00 seconds

CSV Export

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

View:

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

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered and executed:

```
//Create view to display doctor name, hiredate and salary who works under deptno 10

create view doctor_view
as select dname, hiredate, salary
from doctor
where deptno = 10

describe doctor_view

select * from doctor_view
```

The results show two rows of data:

DNAME	HIREDATE	SALARY
Nibir	22-JUN-22	200000
Tuhin	20-NOV-19	500000

2 rows returned in 0.00 seconds

Application Express 2.1.0.0.39
Copyright © 1999, 2006, Oracle. All rights reserved.
Language: en-us

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

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered and executed:

```
//Create a complex view to display Accountant's name and their division name

CREATE VIEW accountant_view AS
SELECT a.accname, al.division
FROM accountant a , accountantloc al
where a.divid = al.divid

describe accountant_view

select * from accountant_view
```

The results show five rows of data:

ACCCNAME	DIVISION
Naim	DHAKA
Anik	BARISHAL
Siam	MYMENSINGH
Anika	CHATTAGRAM
Anjum	KHULNA

5 rows returned in 0.00 seconds

Application Express 2.1.0.0.39
Copyright © 1999, 2006, Oracle. All rights reserved.
Language: en-us

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

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```
//create view to display patient id, name, blood group; do not allow DML operation
to the view.

create view patient_view
as select pid,pname,bloodgroup
from patient
with read only

select * from patient.view
```

The results section displays a table with the following data:

PID	PNAME	BLOODGROUP
5001	Adnan	A+
5002	Abir	O+
5003	Farid	B-
5004	Akhi	AB+
5005	Tanzila	B+

5 rows returned in 0.00 seconds

CSV Export

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

Synonym:

1. Create a synonym for patient_view view.

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```
//Create synonym for Patient_view View
Create synonym p_view
for patient_view
```

The results section displays the message:

Synonym created.

0.00 seconds

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

2. Create synonym for doctor table.

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following code:

```
//Create synonym for doctor table
create synonym doc
for doctor;
```

The output window shows the message "Synonym created." and "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved." The taskbar at the bottom shows various application icons.

3. Delete the created synonym for doctor table.

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following code:

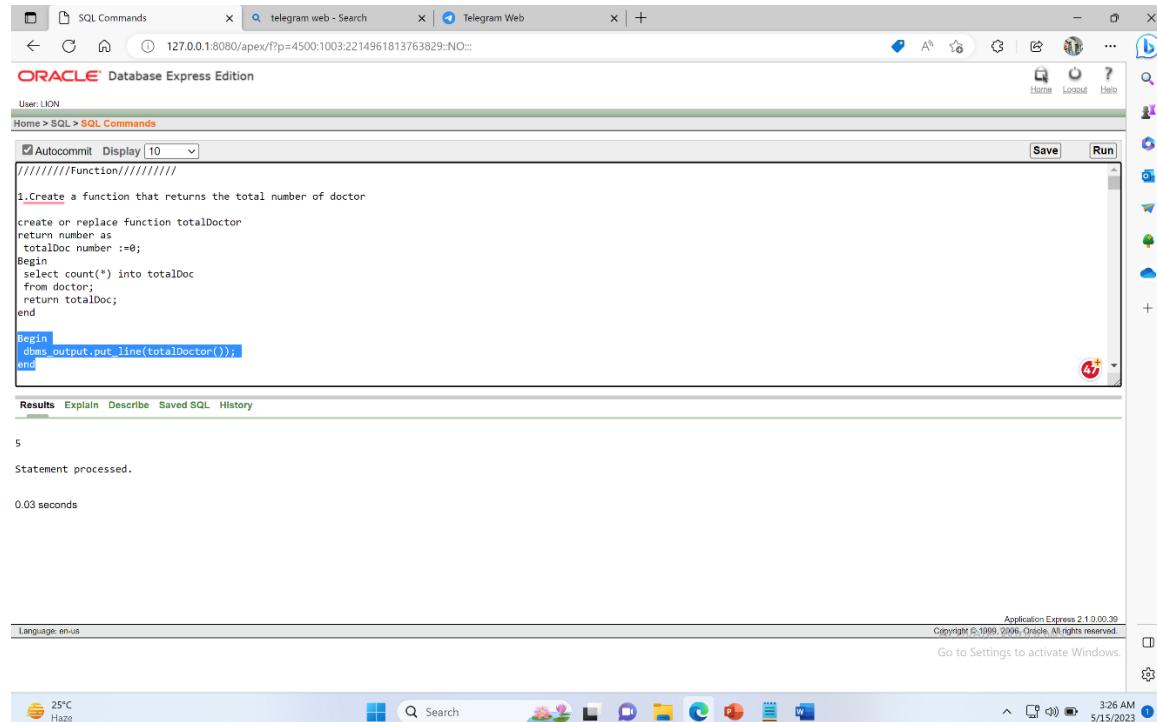
```
//drop the synonym for doctor table
drop synonym doc;
```

The output window shows the message "Synonym dropped." and "0.02 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved." The taskbar at the bottom shows various application icons.

PL/SQL Query Writing:

Function:

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



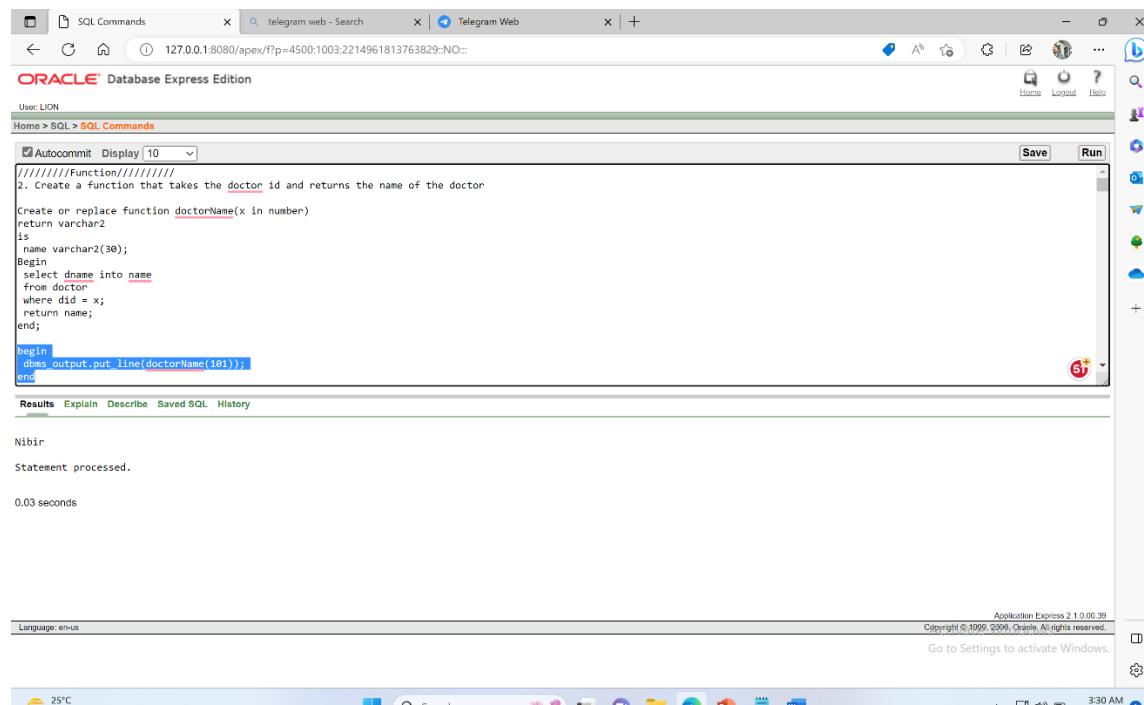
The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a function named `totalDoctor` is being created. The code defines a function that returns the count of doctors from the `doctor` table. The results show that the statement was processed successfully in 0.03 seconds.

```
1.Create a function that returns the total number of doctor
create or replace function totalDoctor
return number as
totalDoc number :=0;
Begin
select count(*) into totalDoc
from doctor;
return totalDoc;
end

Begin
dbms_output.put_line(totalDoctor());
end
```

Results:
5
Statement processed.
0.03 seconds

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



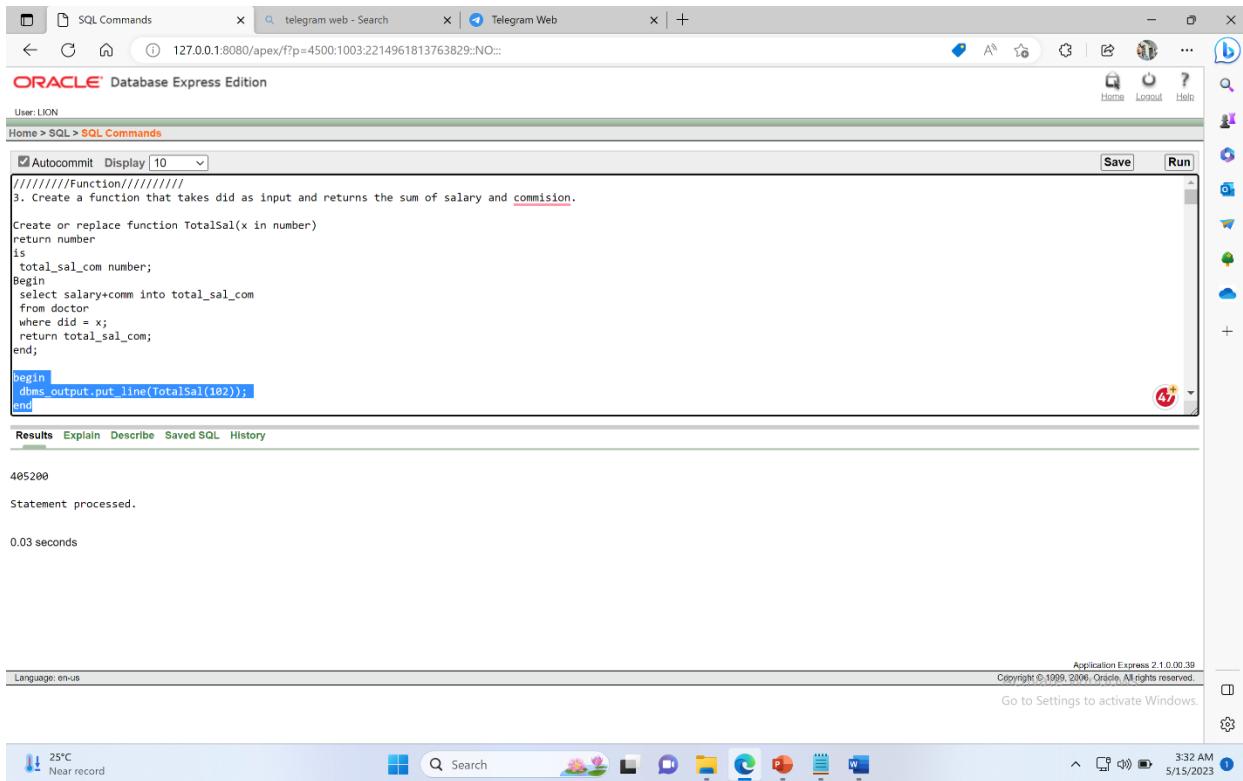
The screenshot shows the Oracle Database Express Edition interface. A function named `doctorName` is being created. This function takes a parameter `x` (representing doctor ID) and returns the doctor's name from the `doctor` table. The results show that the statement was processed successfully in 0.03 seconds.

```
2. Create a function that takes the doctor id and returns the name of the doctor
Create or replace function doctorName(x in number)
return varchar2;
is
name varchar2(30);
Begin
select dname into name
from doctor
where did = x;
return name;
end;

begin
dbms_output.put_line(doctorName(101));
end
```

Results:
Nibir
Statement processed.
0.03 seconds

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



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a function named `TotalSal` is being created. The code defines a function that takes a number `x` as input and returns the sum of salary and commission for a specific doctor ID `did`. The function uses a cursor to select the required data from the `doctor` table and then calculates the total salary and commission. The code ends with a `dbms_output.put_line` statement to display the result. The results pane shows the output of the function call `TotalSal(102)`, which returns `405200`.

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

Create or replace function TotalSal(x in number)
return number
is
total_sal_com number;
Begin
select salary+comm into total_sal_com
from doctor
where did = x;
return total_sal_com;
end;

begin
dbms_output.put_line(TotalSal(102));
end;
```

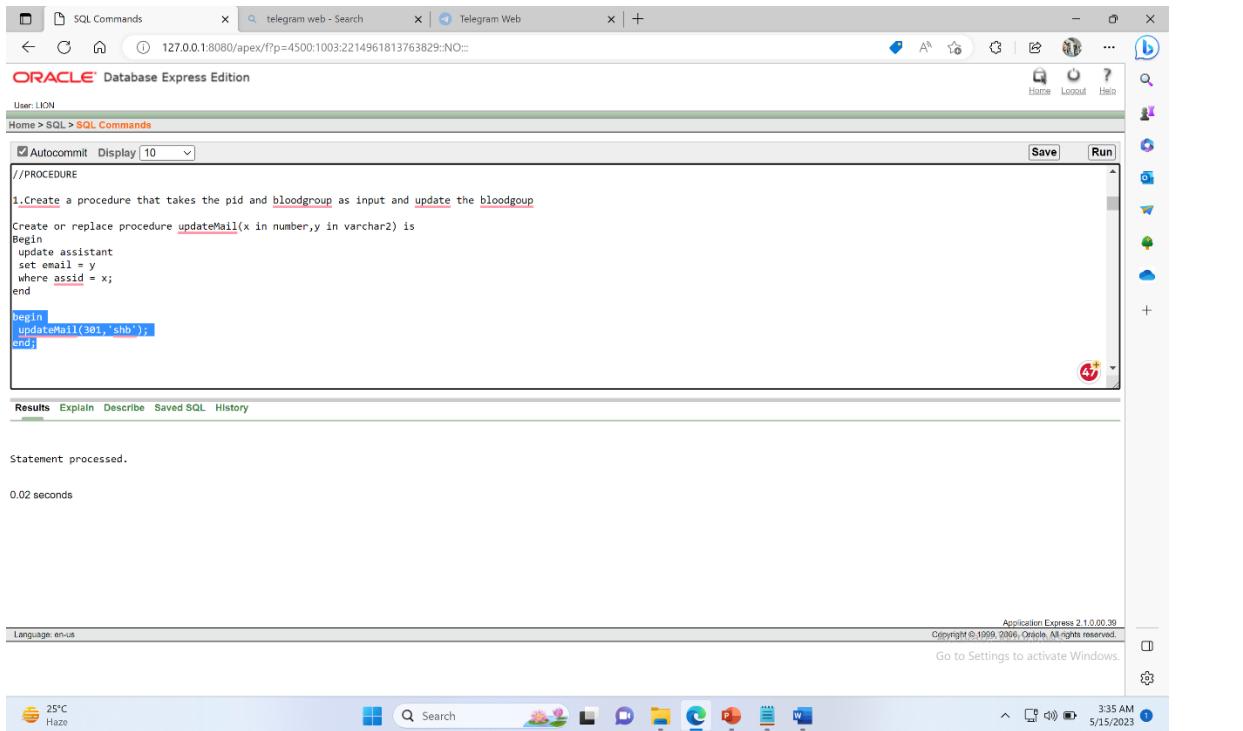
Results Explain Describe Saved SQL History

405200
Statement processed.
0.03 seconds

Language: en-us Application Express 2.1.0.0.39
Copyright © 1999, 2009, Oracle. All rights reserved.
Go to Settings to activate Windows.

Procedure:

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



The screenshot shows the Oracle Database Express Edition interface. A procedure named `updateMail` is being created. The code takes two inputs: `x` (number) and `y` (varchar2). It updates the `bloodgoup` field in the `assistant` table for the record where `assid` equals `x`. The procedure then calls itself with parameters `301` and `'shb'`. The results pane shows the output of the procedure call `updateMail(301, 'shb')`, which is `Statement processed.`

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

Create or replace procedure updateMail(x in number,y in varchar2) is
Begin
update assistant
set email = y
where assid = x;
end;

begin
updateMail(301,'shb');
end;
```

Results Explain Describe Saved SQL History

Statement processed.
0.02 seconds

Language: en-us Application Express 2.1.0.0.39
Copyright © 1999, 2009, Oracle. All rights reserved.
Go to Settings to activate Windows.

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

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a procedure named `updateSal` is created. The code doubles the salary for a given doctor ID (`did`). A test run shows a salary increase from 200200 to 400400.

```
//PROCEDURE
2. Create a procedure that takes did and double the salary;
Create or replace procedure updateSal(x in out number) is
begin
  update doctor
  set salary = 2*salary
  where did =>x;
end

declare
a number;
begin
a:= 101;
updateSal(a);
end;
```

Results:

```
Previous salary: 200200
Current salary: 400400
Statement processed.

0.04 seconds
```

Application Express 2.1.0.0.38
Language: en-us Copyright © 1999, 2006, Oracle. All rights reserved.
Go to Settings to activate Windows.

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

The screenshot shows the Oracle Database Express Edition interface. A procedure named `totalPat` is created to output the total number of patients. The code uses a cursor to count patients and then prints the result.

```
//PROCEDURE
3. Create a procedure that displays the total number of patients.
CREATE OR REPLACE PROCEDURE totalPat AS
x NUMBER;
BEGIN
SELECT COUNT(*) INTO x
FROM patient;

DBMS_OUTPUT.PUT_LINE('Total number of patient:'|| x);

END;

begin
totalpat;
end;
```

Results:

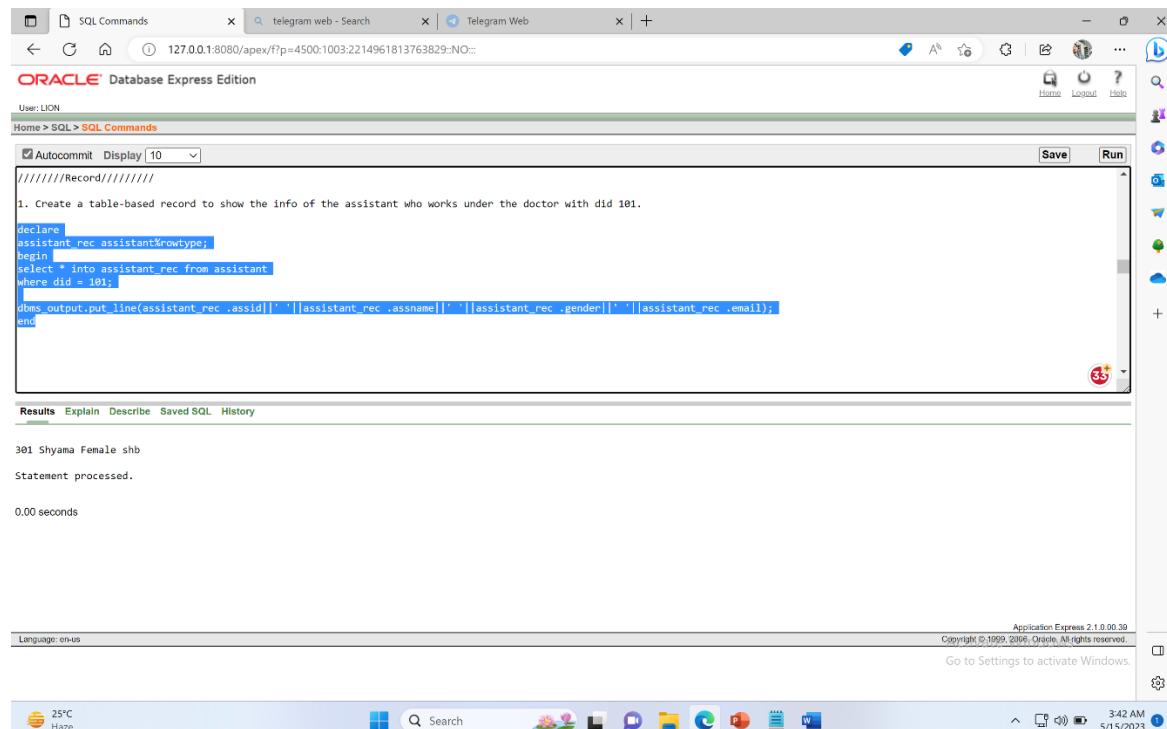
```
Total number of patient:5
Statement processed.

0.00 seconds
```

Application Express 2.1.0.0.38
Language: en-us Copyright © 1999, 2006, Oracle. All rights reserved.
Go to Settings to activate Windows.

Record:

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



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following PL/SQL code is run:

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

declare
  assistant_rec assistant%rowtype;
begin
  select * into assistant_rec from assistant
  where did = 101;
  dbms_output.put_line(assistant_rec .assid||'|'||assistant_rec .assname||'|'||assistant_rec .gender||'|'||assistant_rec .email);
end;
```

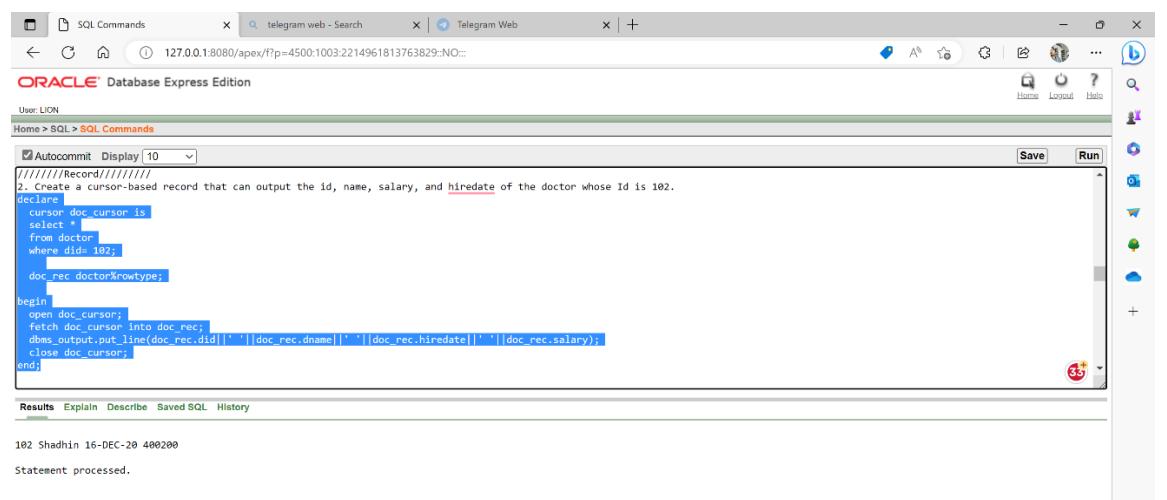
The results show a single row of data:

assid	assname	gender	email
301	Shyama	Female	shb

Statement processed. 0.00 seconds.

At the bottom, the status bar indicates Language: en-us and Application Express 2.1.0.0.39.

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



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following PL/SQL code is run:

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

declare
  cursor doc_cursor is
  select *
  from doctor
  where did= 102;
  doc_rec doctor%rowtype;
begin
  open doc_cursor;
  fetch doc_cursor into doc_rec;
  dbms_output.put_line(doc_rec.did||'|'||doc_rec.dname||'|'||doc_rec.hiredate||'|'||doc_rec.salary);
  close doc_cursor;
end;
```

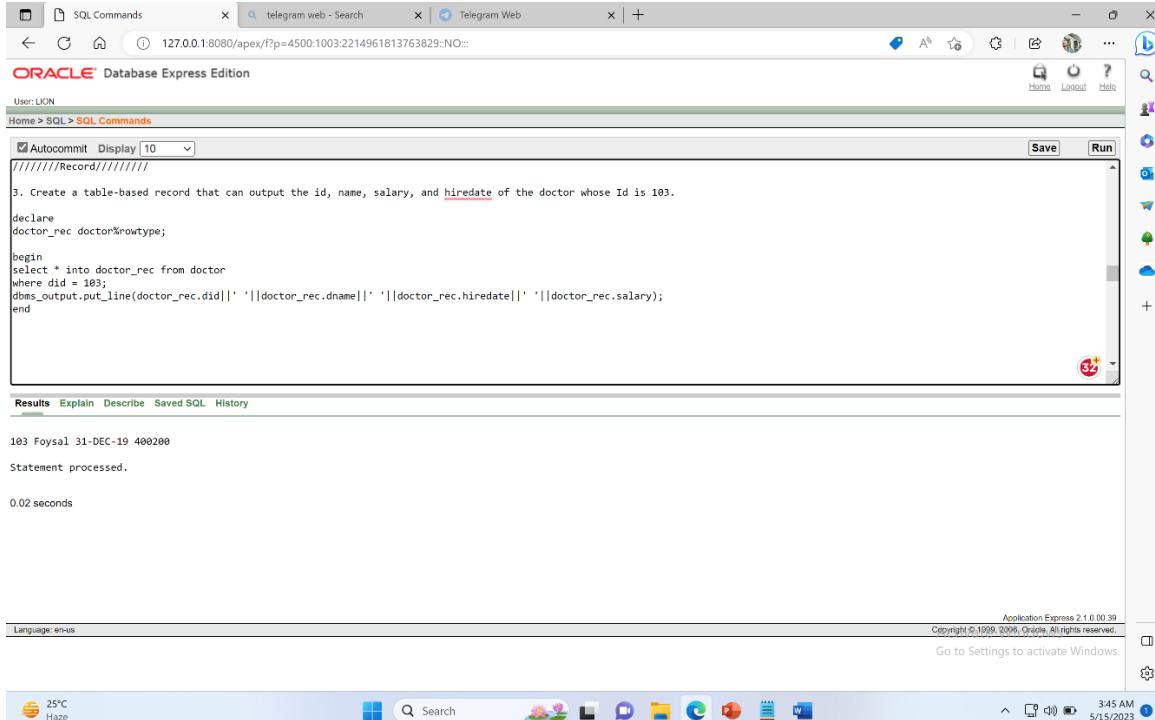
The results show a single row of data:

did	dname	hiredate	salary
102	Shadhin	16-DEC-20	400200

Statement processed. 0.01 seconds.

At the bottom, the status bar indicates Language: en-us and Application Express 2.1.0.0.39.

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



The screenshot shows the Oracle Database Express Edition interface within a web browser. The URL is 127.0.0.1:8080/apex/?p=4500:103:2214961813763829:NO. The page title is "SQL Commands". The code entered is:

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

declare
doctor_rec doctor%rowtype;

begin
select * into doctor_rec from doctor
where did = 103;
dbms_output.put_line(doctor_rec.did||' '||doctor_rec.dname||' '||doctor_rec.hiredate||' '||doctor_rec.salary);
end
```

The results show the output for doctor_id 103:

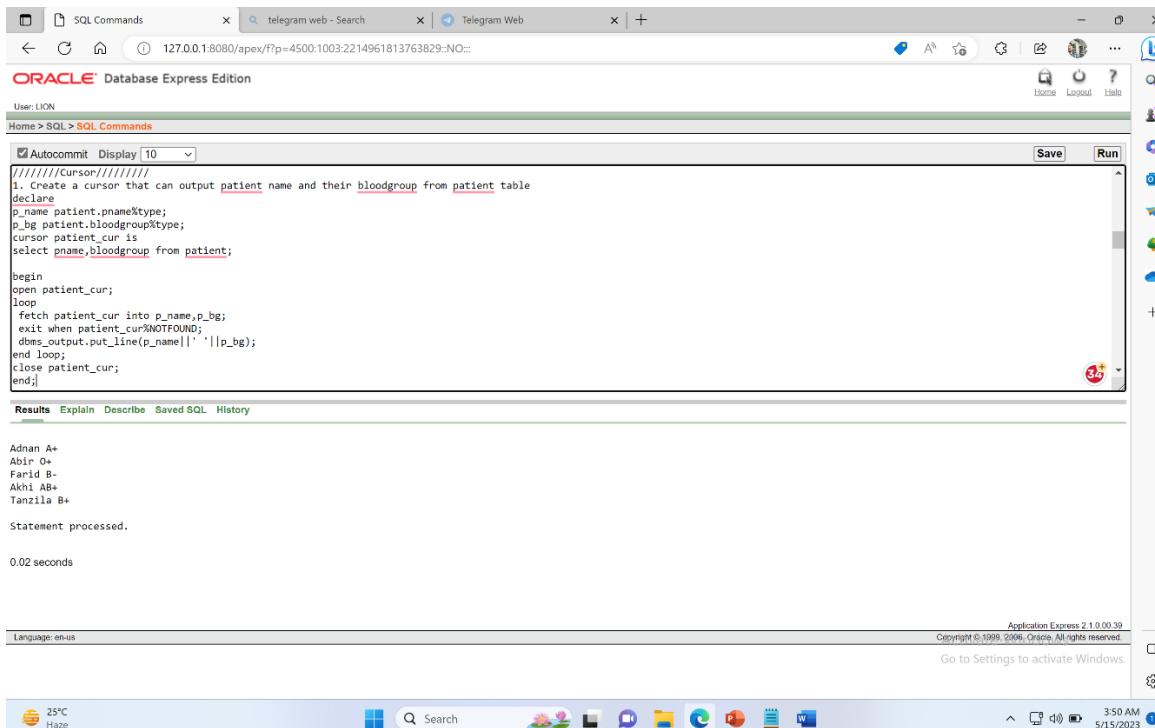
```
103 Foyosal 31-DEC-19 400200
Statement processed.

0.02 seconds
```

At the bottom right of the interface, there is a small icon with a red circle containing the number 32.

Cursor:

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



The screenshot shows the Oracle Database Express Edition interface within a web browser. The URL is 127.0.0.1:8080/apex/?p=4500:103:2214961813763829:NO. The page title is "SQL Commands". The code entered is:

```
///////cursor/////////
1. Create a cursor that can output patient name and their bloodgroup from patient table
declare
p_name patient.p_name%type;
p_bg patient.bloodgroup%type;
cursor patient_cur is
select pname,bloodgroup from patient;

begin
open patient_cur;
loop
fetch patient_cur into p_name,p_bg;
exit when patient_cur%NOTFOUND;
dbms_output.put_line(p_name||' '||p_bg);
end loop;
close patient_cur;
end;
```

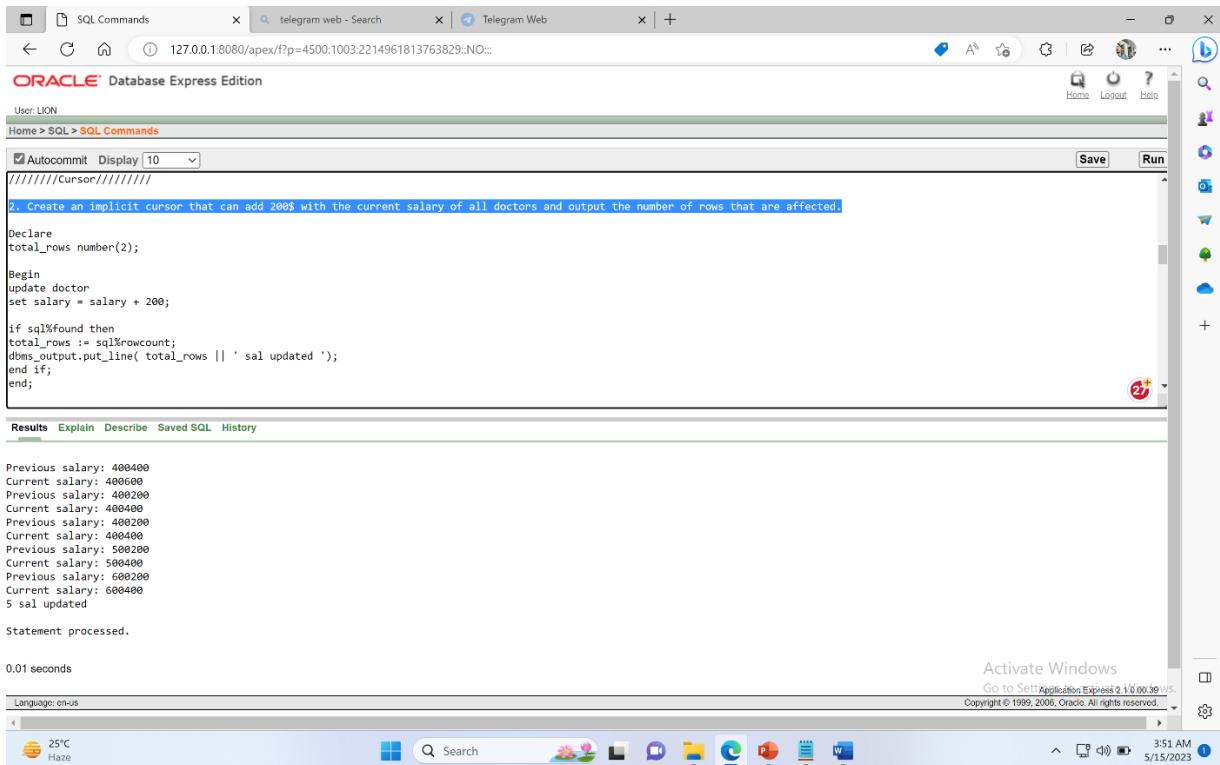
The results show the patient names and bloodgroups:

```
Adnan A+
Abir O-
Farid B-
Akhi AB+
Tanzila B+
Statement processed.

0.02 seconds
```

At the bottom right of the interface, there is a small icon with a red circle containing the number 32.

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.



```

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.

Declare
total_rows number(2);

Begin
update doctor
set salary = salary + 200;

if sql%found then
total_rows := sql%rowcount;
dbms_output.put_line( total_rows || ' sal updated ');
end if;
end;

```

Results

```

Previous salary: 400400
Current salary: 400600
Previous salary: 400200
Current salary: 400400
Previous salary: 400200
Current salary: 400400
Previous salary: 500200
Current salary: 500400
Previous salary: 600200
Current salary: 600400
5 sal updated

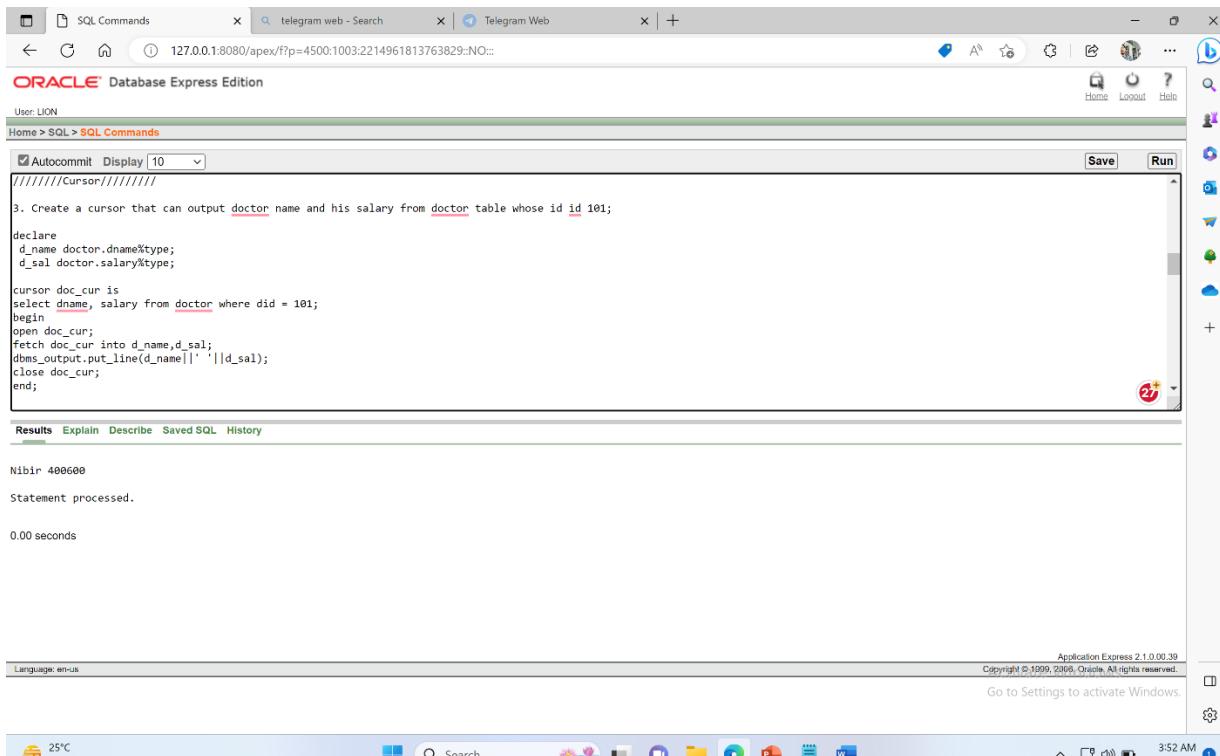
Statement processed.

0.01 seconds

```

Activate Windows
Go to Settings Application Express 2.1.0.0.0.39 vs.
Copyright © 1999, 2006, Oracle. All rights reserved.

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



```

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

declare
d_name doctor.dname%type;
d_sal doctor.salary%type;

cursor doc_cur is
select dname, salary from doctor where did = 101;
begin
open doc_cur;
fetch doc_cur into d_name,d_sal;
dbms_output.put_line(d_name||' '||d_sal);
close doc_cur;
end;

```

Results

```

Nibir 400600

Statement processed.

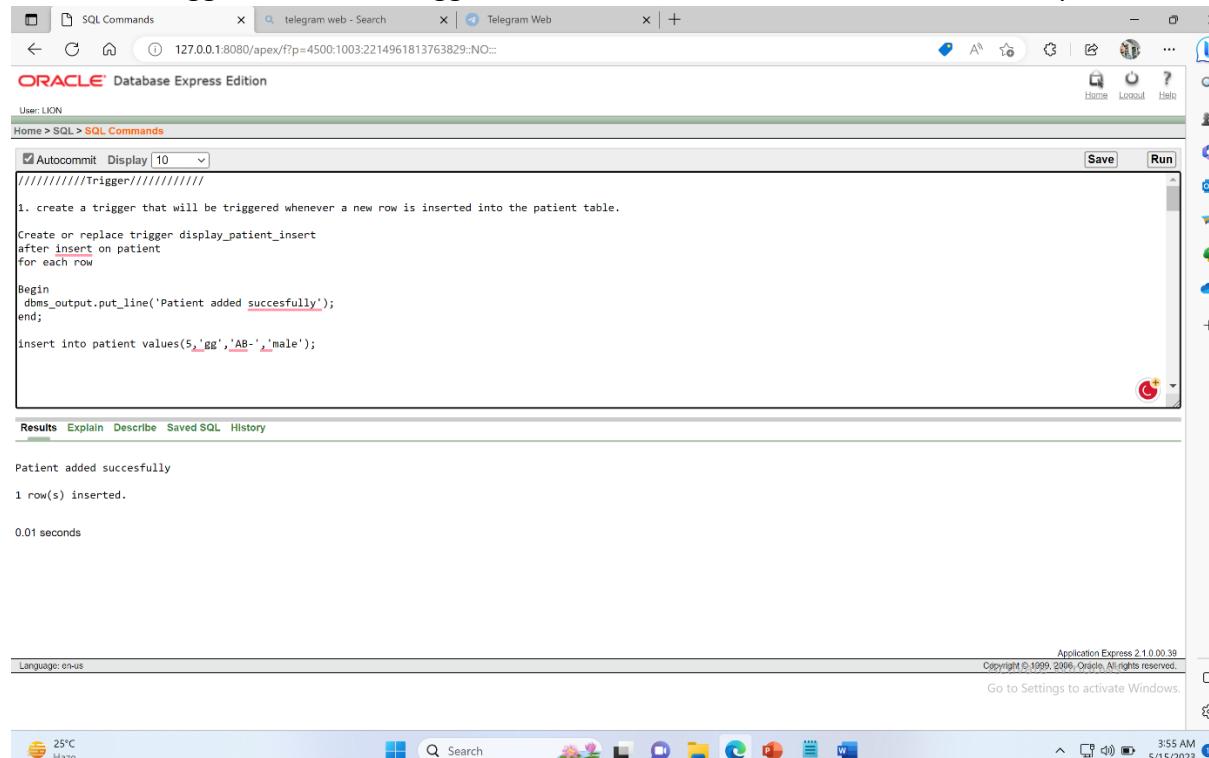
0.00 seconds

```

Activate Windows
Go to Settings to activate Windows.
Application Express 2.1.0.0.0.39
Copyright © 1999, 2006, Oracle. All rights reserved.

Trigger:

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



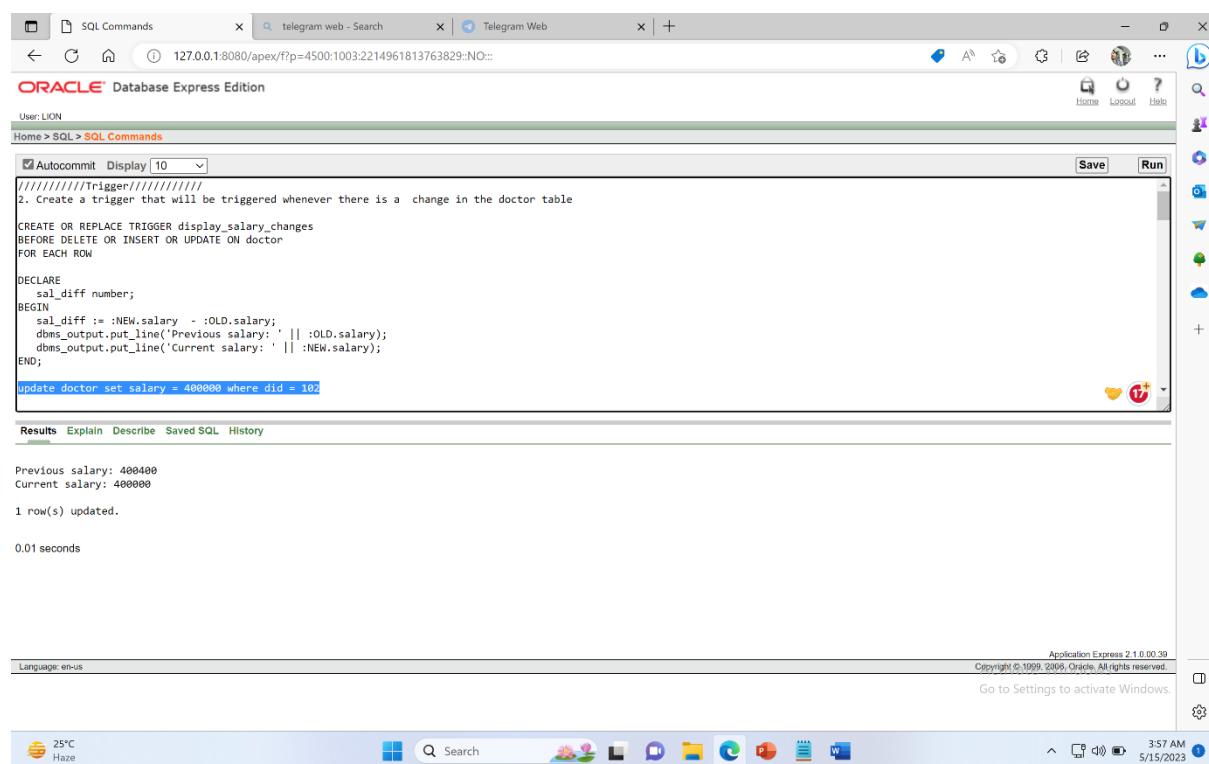
The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a trigger named 'display_patient_insert' is created. The code inserts a message into the DBMS_OUTPUT buffer and adds a new row to the 'patient' table. The results show a successful insertion of one row.

```
1. create a trigger that will be triggered whenever a new row is inserted into the patient table.  
Create or replace trigger display_patient_insert  
after insert on patient  
for each row  
Begin  
dbms_output.put_line('Patient added successfully');  
end;  
insert into patient values(5,'gg','AB-','male');
```

Patient added successfully
1 row(s) inserted.
0.01 seconds

Application Express 2.1.0.00.39
Copyright © 1999-2006, Oracle. All rights reserved.
Go to Settings to activate Windows.

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



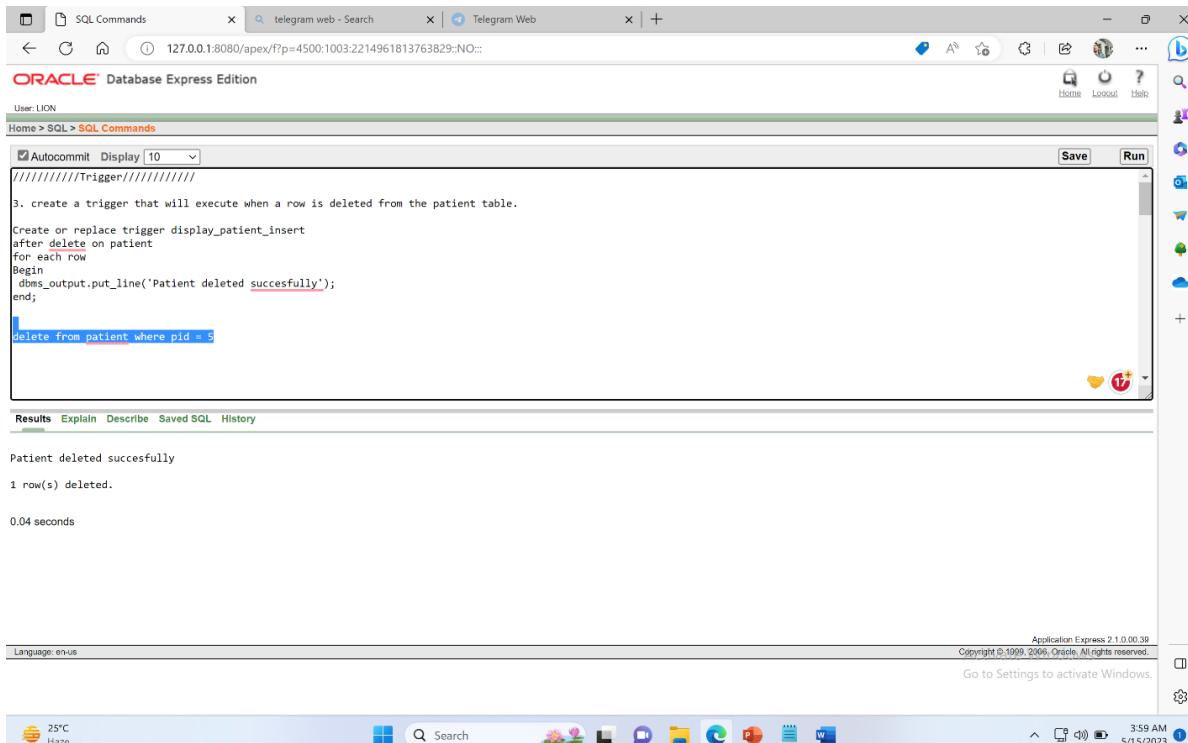
The screenshot shows the Oracle Database Express Edition interface. A trigger named 'display_salary_changes' is created to log salary changes in the 'doctor' table. It uses DBMS_OUTPUT to print the previous and current salaries for each updated row. The results show a salary update for doctor ID 10.

```
2. Create a trigger that will be triggered whenever there is a change in the doctor table  
CREATE OR REPLACE TRIGGER display_salary_changes  
BEFORE DELETE OR INSERT OR UPDATE ON doctor  
FOR EACH ROW  
DECLARE  
    sal_diff number;  
BEGIN  
    sal_diff := :NEW.salary - :OLD.salary;  
    dbms_output.put_line('Previous salary: ' || :OLD.salary);  
    dbms_output.put_line('Current salary: ' || :NEW.salary);  
END;  
update doctor set salary = 400000 where did = 10;
```

Previous salary: 400400
Current salary: 400000
1 row(s) updated.
0.01 seconds

Application Express 2.1.0.00.39
Copyright © 1999-2006, Oracle. All rights reserved.
Go to Settings to activate Windows.

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



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a trigger named 'display_patient_insert' is created. It triggers after a delete operation on the 'patient' table, and it outputs a message to the DBMS_OUTPUT when a row is deleted. The trigger body includes a delete statement for the 'patient' table where pid = 5. The results show that the row was deleted successfully, and the process took 0.04 seconds.

```


///////Trigger/////////
3. create a trigger that will execute when a row is deleted from the patient table.

Create or replace trigger display_patient_insert
after delete on patient
for each row
Begin
dbms_output.put_line('Patient deleted succesfully');
end;

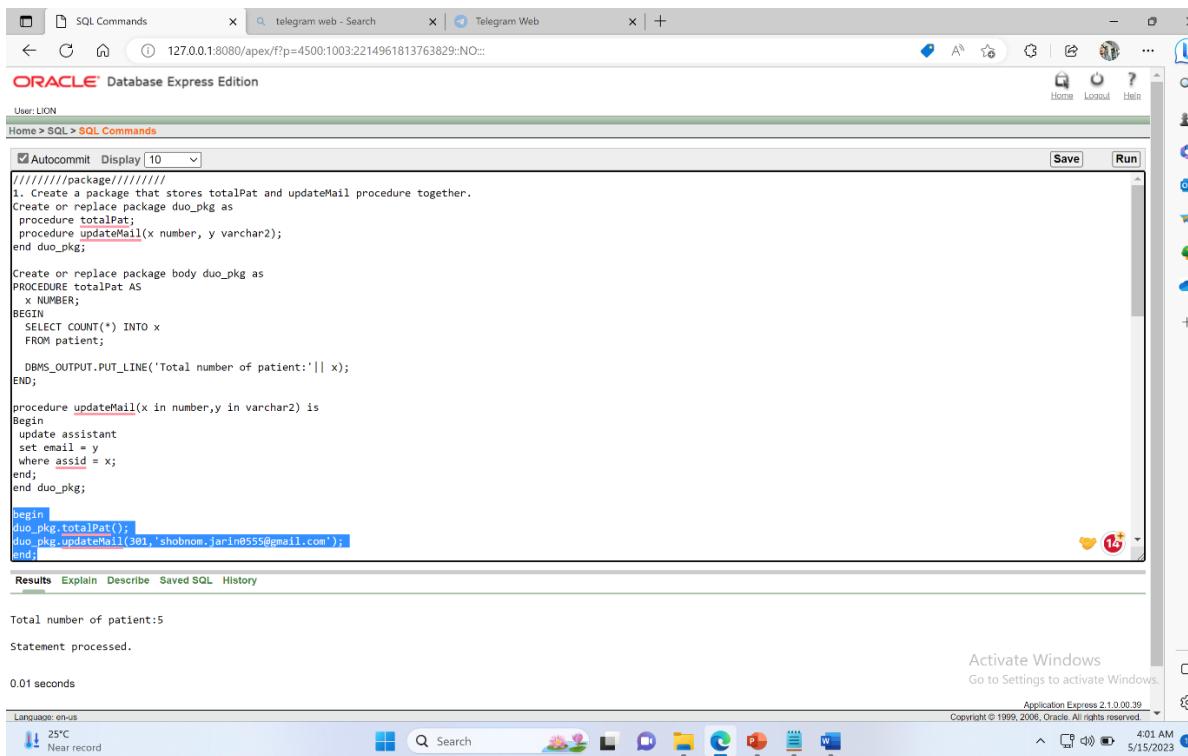
delete from patient where pid = 5


```

Patient deleted succesfully
1 row(s) deleted.
0.04 seconds

Package:

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



The screenshot shows the Oracle Database Express Edition interface. A package named 'duo_pkg' is created, containing two procedures: 'totalPat' and 'updateMail'. The 'totalPat' procedure counts the number of patients in the 'patient' table and outputs the result. The 'updateMail' procedure takes parameters 'x' (number) and 'y' (varchar2), sets the email to 'y', and updates the record in the 'patient' table where assid = x. The results show the total number of patients (5) and a statement processed message.

```


///////package/////////
1. Create a package that stores totalPat and updateMail procedure together.
Create or replace package duo_pkg as
procedure totalPat;
procedure updateMail(x number, y varchar2);
end duo_pkg;

Create or replace package body duo_pkg AS
PROCEDURE totalPat AS
 x NUMBER;
BEGIN
 SELECT COUNT(*) INTO x
 FROM patient;

 DBMS_OUTPUT.PUT_LINE('Total number of patient:'|| x);

END;

procedure updateMail(x in number,y in varchar2) is
Begin
 update assistant
 set email = y
 where assid = x;
end;
end duo_pkg;

begin
duo_pkg.totalPat();
duo_pkg.updateMail(301,'shobnom.jarin055@gmail.com');
end;


```

Total number of patient:5
Statement processed.
0.01 seconds

2. Create a package that stores the updateMail procedure.

The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window displays the following code:

```
///////package/////////
2. Create a package that stores the updateMail procedure.

Create or replace package doc_pkg as
procedure updateMail(x number, y varchar2);
end doc_pkg;

Create or replace package body doc_pkg as
procedure updateMail(x in number,y in varchar2) is
Begin
update assistant
set email = y
where assid = x;
end;
end doc_pkg;

begin
doc_pkg.updateMail(301,'shobnom.jarin055@gmail.com');
end;
```

The code creates a package `doc_pkg` with a single procedure `updateMail` that takes two parameters: `x` (number) and `y` (varchar2). The procedure updates a table named `assistant` by setting the `email` column to `y` where the `assid` column equals `x`. A `begin` block calls this procedure with `x=301` and `y='shobnom.jarin055@gmail.com'`.

Below the code, the results show:

Statement processed.
0.02 seconds

At the bottom right, the Windows taskbar shows the date as 5/15/2023 and the time as 4:04 AM.

3. Create a package that hold the totalPat Procedure.

The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window displays the following code:

```
///////package/////////
3. Create a package that hold the totalPat Procedure.

Create or replace package pat_pkg as
procedure totalPat;
end pat_pkg;

Create or replace package body pat_pkg as
PROCEDURE totalPat AS
x NUMBER;
BEGIN
SELECT COUNT(*) INTO x
FROM patient;
DBMS_OUTPUT.PUT_LINE('Total number of patient:'|| x);
END;
end pat_pkg;

begin
pat_pkg.totalPat;
end;
```

The code creates a package `pat_pkg` with a single procedure `totalPat`. This procedure selects the count of rows from the `patient` table into a variable `x`, then uses `DBMS_OUTPUT.PUT_LINE` to print the total number of patients. A `begin` block calls this procedure.

Below the code, the results show:

Total number of patient:5
Statement processed.
0.00 seconds

At the bottom right, the Windows taskbar shows the date as 5/15/2023 and the time as 4:05 AM.

Relational Algebra:

1. Project Operation(π):

->To project the DoctorID and Name attributes from the Doctor table :

$$\pi_{did, dname}(\text{Doctor})$$

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

$$\pi_{Phoneno}(\text{PatientContact})$$

2. Selection Operation(σ):

->To select all the Accountants who lives in Dhaka Division:

$$\sigma \text{Division} = \text{'DHAKA'}(\text{Accountant})$$

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

$$\pi_{Name}(\sigma \text{PID}=5002(\text{Patient}))$$

3. Cartesian Product Operation:

->To find all possible Appointments for each doctor :

$$\text{DoctorAppointment} <- \text{Doctor} \times \text{Appointment}$$

4. Union Operation(u) :

->To combine all Appointments for June 21th,2023 and June 25th,2023 :

$$\text{Appointment_1} <- \sigma \text{Date} = \text{'2023-06-21'}(\text{Appointment})$$
$$\text{Appointment_2} <- \sigma \text{Date} = \text{'2023-06-25'}(\text{Appointment})$$
$$\text{Appointment_1} \cup \text{Appointment_2}$$

5. Rename Operation (ρ) :

->To rename the ACCNAME attribute in the Accountant table to Accountant Name

$$\rho \text{ Accountant Name/ACCNAME}(\text{Accountant})$$

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.