AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

INTRODUCTION TO DATABASE

Fall 2023-2024
Section:E

TITLE

Doctor Appointment Management System

Supervised By
Rezwan Ahmad

Submitted By: DataCraft

Name	ID
NUSRAT JAHAN MIM	22-49020-3
NUR-E SARJIN KHAN	22-49754-3
MD. OWAFEEUZZAMAN PATWARY	22-49885-3

TABLE OF CONTENTS

TOPICS	Page no.
Title Page	1
Table of Content	2
1. Introduction	3
2. Case Study	4
3. ER Diagram	5
4. Normalization	6-7
5. Finalization	8
6. Table Creation	9-12
7. Data Insertion	13-15
8. Query Test	15-21
9. Database Connection	22-29
10. Conclusion	30

Introduction:

This is a database project on Doctor Appointment Management System that will store the data of the patients, receptionists, doctors, hospitals, and their branches. This system will help the medical sector to allow them to properly arrange and store their data for different hospitals. The database was initiated in Oracle and the Database connection was done with JAVA and connected it with ORACLE . The database is created with 8 tables consisting of different data.

Case Study:

Doctor Appointment Management System

In a Doctor Appointment Management System, a patient can schedule multiple doctors' appointments by calling the receptionist. Patients provide their name, age, gender, and phone number as a unique identifier. The patient pays the receptionist and to confirm payment, trx_id is stored. The Receptionist is identified by a unique ID, Name, and Phone Number. The doctors are distinguished by Name, ID, Phone Number. The system also stores the schedule of each doctor. Doctors can be specialized into cardiologist, pediatrician, neurologist. A doctor has atmost 3 receptionists and a receptionist may be assigned to atmost 2 doctors. Doctors may be affiliated with multiple Hospitals, each with a distinct licence_no, Name, email, and Phone Number. A hospital may have multiple doctors and branches. Branches have ID to be uniquely identified. The system also has the address, name, phone number, email of the Branches.

ER Diagram:

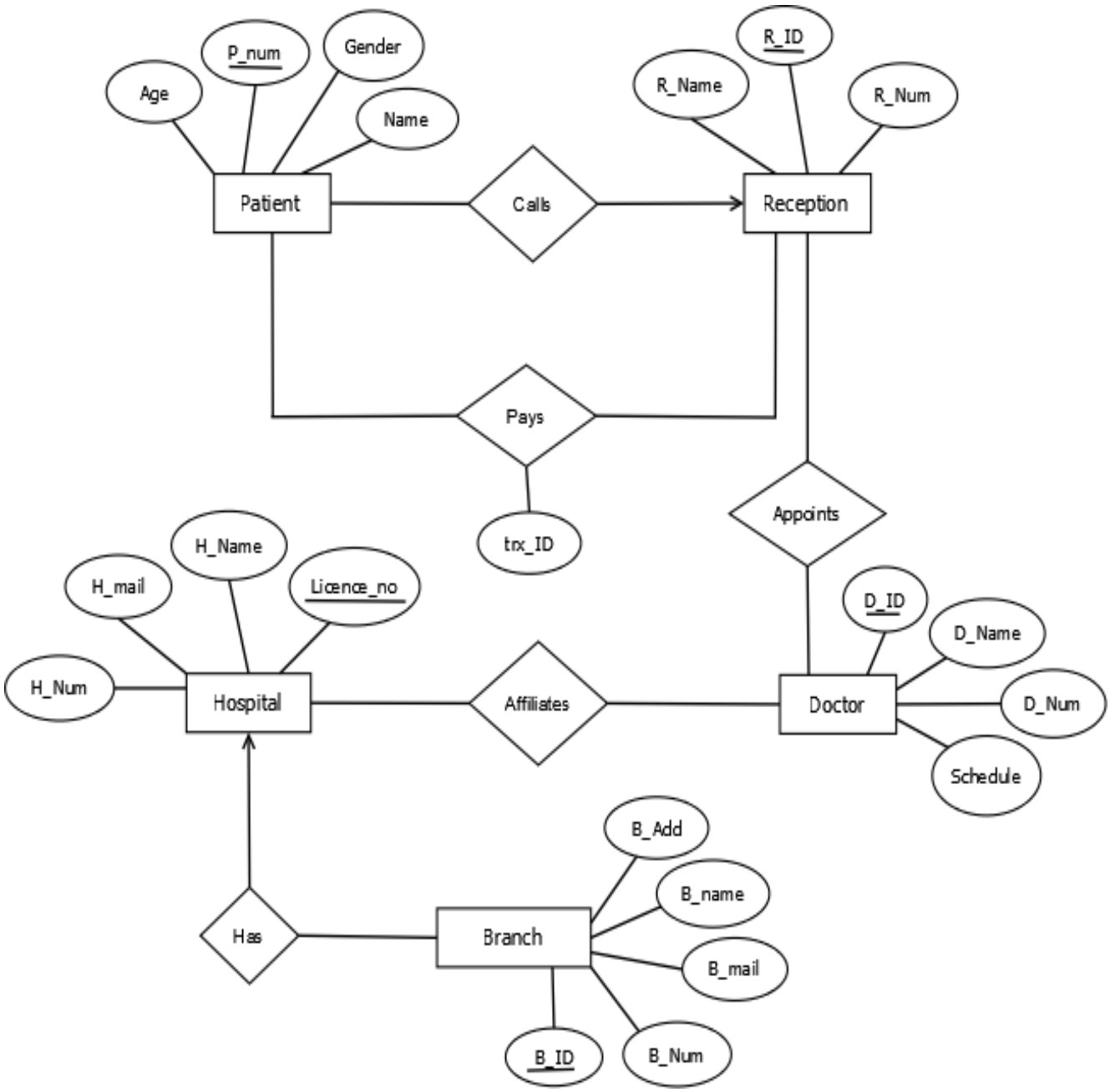

Fig 3: Doctor Appointment Management System

NORMALIZATION:

Call

UNF: name,age,gender,p_num,R_ID,R_name,R_num

1NF: p_num,name,age,gender, R_ID,R_name,R_num

2NF: (1)p_num(PK),gender,age,name,R_ID(FK)

(2)R_id(PK),R_name,R_num

3NF: same as 2NF

Pays

UNF: name,age,gender,p_num,R_ ID,R_name,R_num,trx_ ID

1NF: p_num,name,age,gender, R_ ID,R_name,R_num,trx_ ID

2NF: (1) trx_ ID (PK),p_num(FK),gender,age,name,R_ID(FK)

(2)R_ID(PK),R_name,R_num

3NF: same as 2NF

Appoints

UNF: R_ID,R_name,R_num,D_ID, D_name,D_num,schedule.

1NF: R_ID,R_name,R_num, D_ID, D_name,D_num,schedule

2NF: (1) R_ID(PK),R_name,R_num

(2) D_ID(PK), D_name,D_num,schedule.

(3)R_ID(PK), D_ID(FK)

3NF: same as 2NF

Affiliates :

UNF: D_ID, D_name,D_num,schedule,licence_no,H_name,H_mail,H_num

1NF: D_ID, D_name,D_num,schedule,licence_no,H_name,H_mail,H_num

2NF: (1) licence_no(PK),H_name,H_mail,H_num

(2) D_ID(PK), D_name,D_num,schedule.

(3)D_ID(PK),licence_no(FK)

3NF: same as 2NF

Has

UNF: licence_no,H_name,H_mail,H_num,B_ID,B_num,B_name,B_mail,B_add

1NF: licence_no,H_name,H_mail,H_num, B_ID,B_num,B_name,B_mail,B_add

2NF: (1) licence_no(PK),H_name,H_mail,H_num

(2) B_ID,B_num,B_name,B_mail,B_add,licence_no(FK)

3NF: same as 2NF

FINALIZATION

Columns	Tables
(1) p_num(PK),gender,age,name,R_id(FK)	[CALL]
(2) R_id(PK),R_name,R_num	[RECEPTIONIST]
(3) Trx_id(PK), p_num(FK),gender,age,name,R_id(FK)	[PAY]
(4) R_id(PK),R_name,R_num	x
(5) R_id(PK),R_name,R_num	x
(6) D_ID(PK), D_name,D_num,schedule.	[DOCTOR]
(7) R_id(PK), D_ID(FK)	[APPOINT]
(8) licence_no(PK),H_name,H_mail,H_num	[HOSPITAL]
(9) D_ID(PK), D_name,D_num,schedule.	x
(10) D_ID(PK),licence_no(FK)	[AFFILIATE]
(11) licence_no(PK),H_name,H_mail,H_num	x
(12) B_ID(PK),B_num,B_name,B_mail,B_add,licence_no(FK)	[HAS]

Final Tables:

(1) **CALL**: p_num(PK),gender,age,name,R_id(FK)

(2) **RECEPTIONIST**: R_id(PK),R_name,R_num

(3) **PAY:** Trx_id(PK), p_num(FK),gender,age,name,R_id(FK)

(4) **DOCTOR:** D_ID(PK), D_name,D_num,schedule.

(5) **APPOINT:** R_id(PK), D_ID(FK)

(6) **HOSPITAL:** licence_no(PK),H_name,H_mail,H_num

(7) **AFFILIATE:** D_ID(PK),licence_no(FK)

(8) **HAS:** B_ID(PK),B_num,B_name,B_mail,B_add,licence_no(FK)

Table Creation:

Call table:

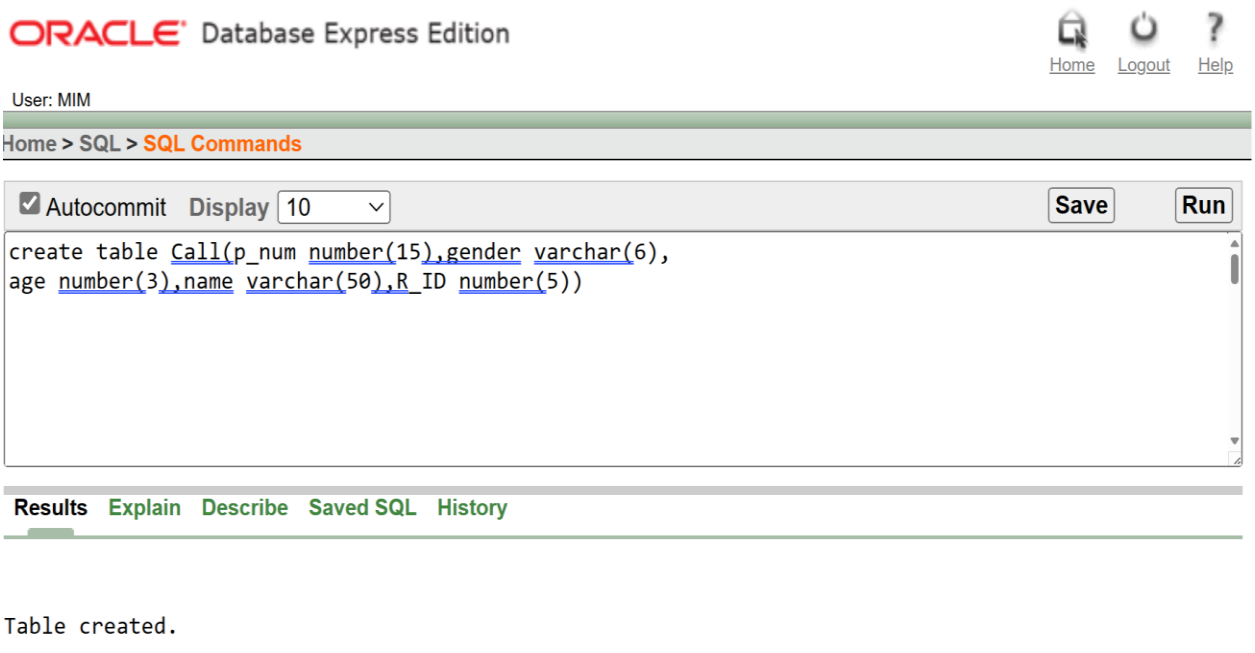

Fig 6.1: Call table creation

Receptionist table:

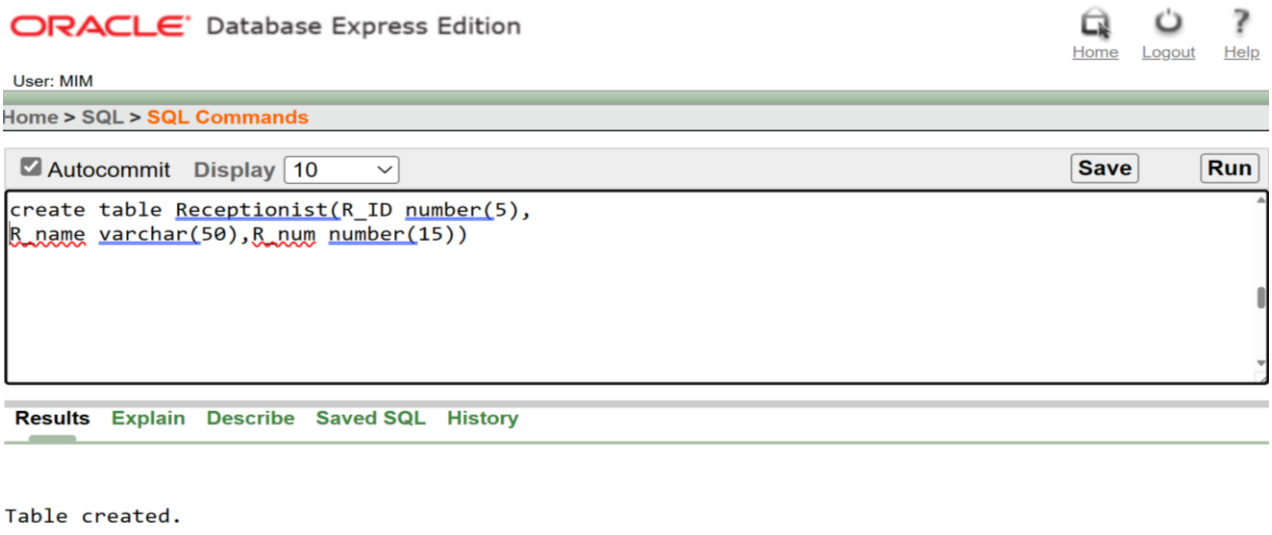

Fig 6.2: Receptionist table creation

Pay Table:

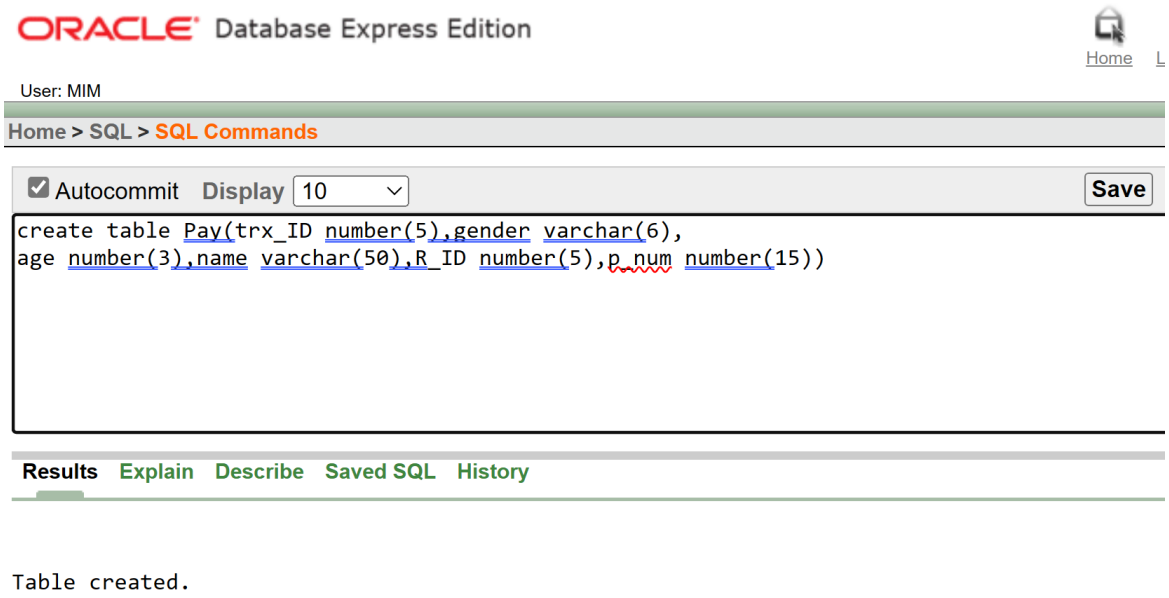

Fig 6.3: Pay table creation

Doctor Table:

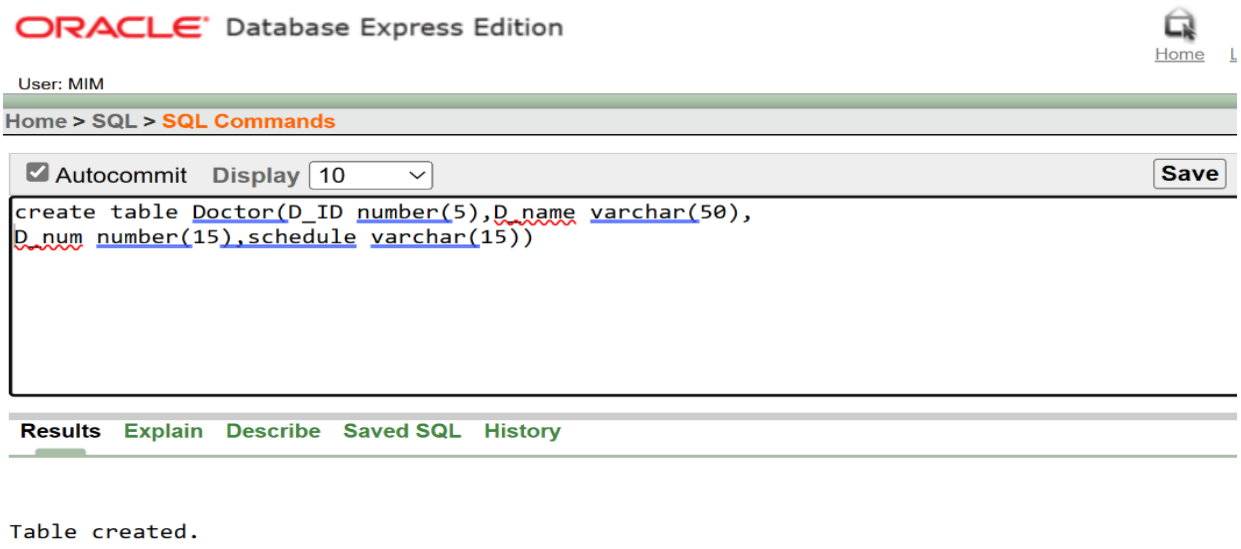

Fig 6.4: Doctor table creation

Appoint Table:

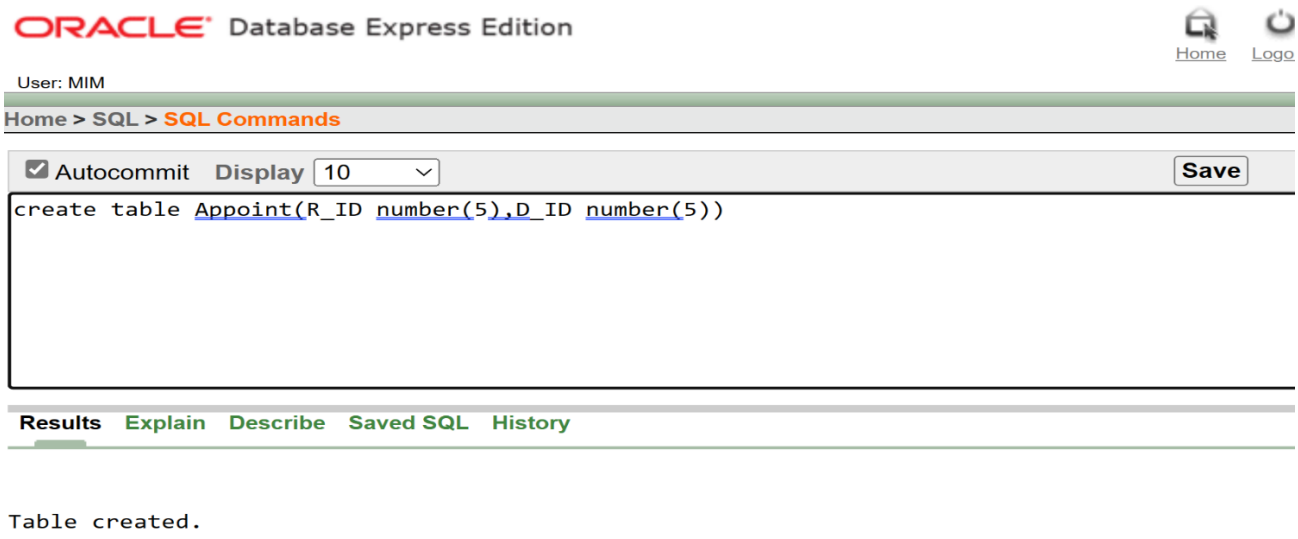

Fig 6.5: Appoint table creation

Hospital Table:

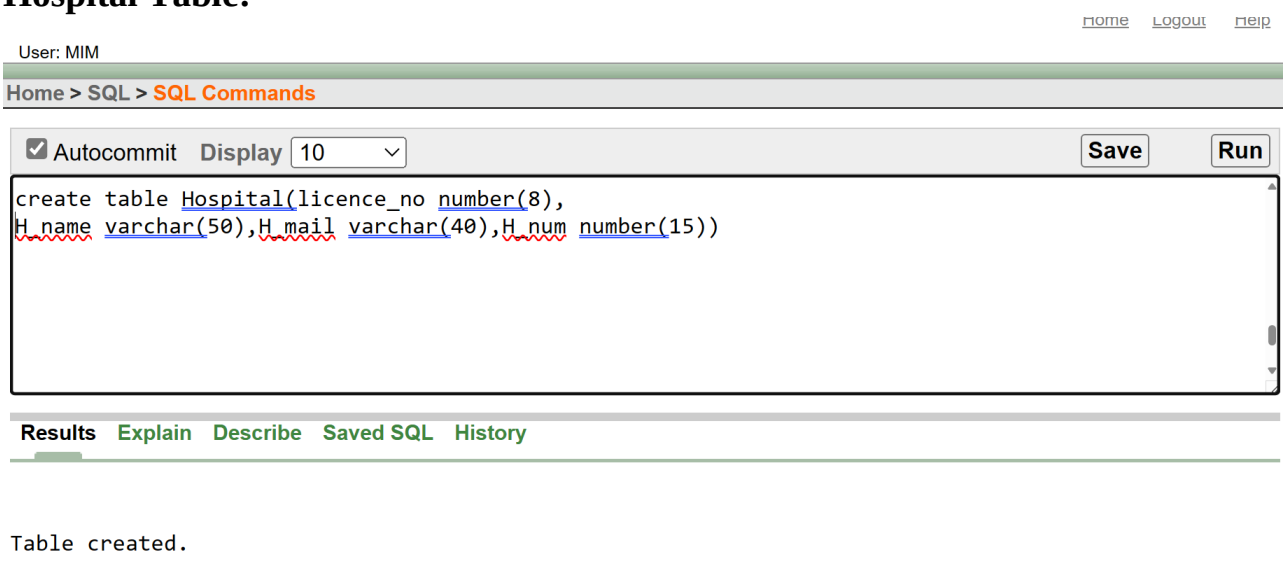

Fig 6.6: Hospital Table Creation

Affiliate Table:

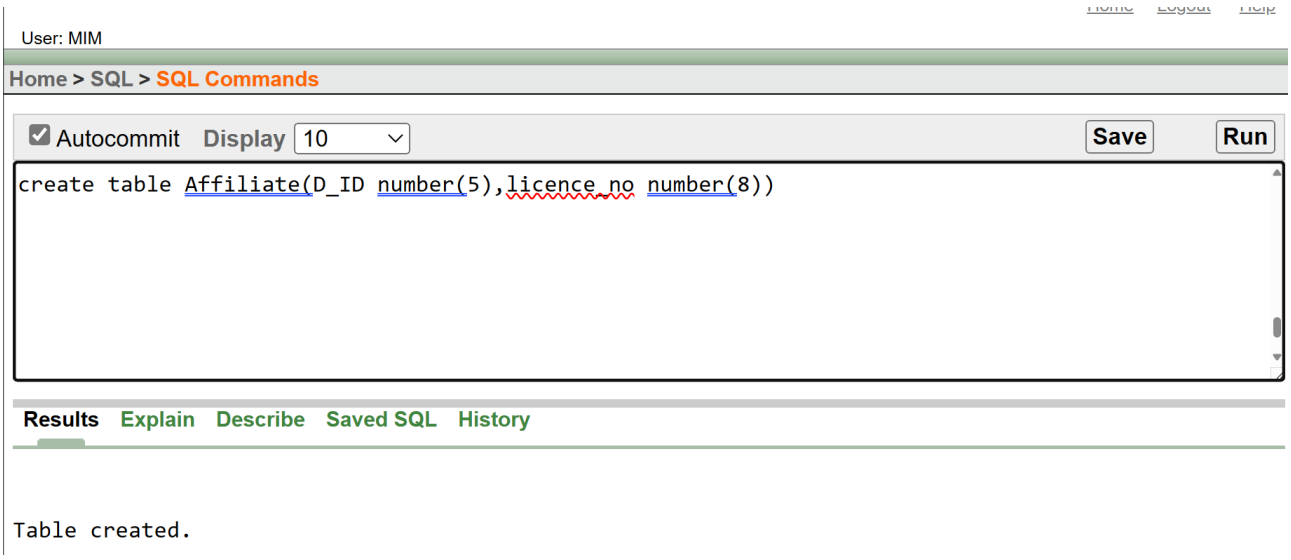

Fig 6.7: Affiliate Table Creation

Has Table:

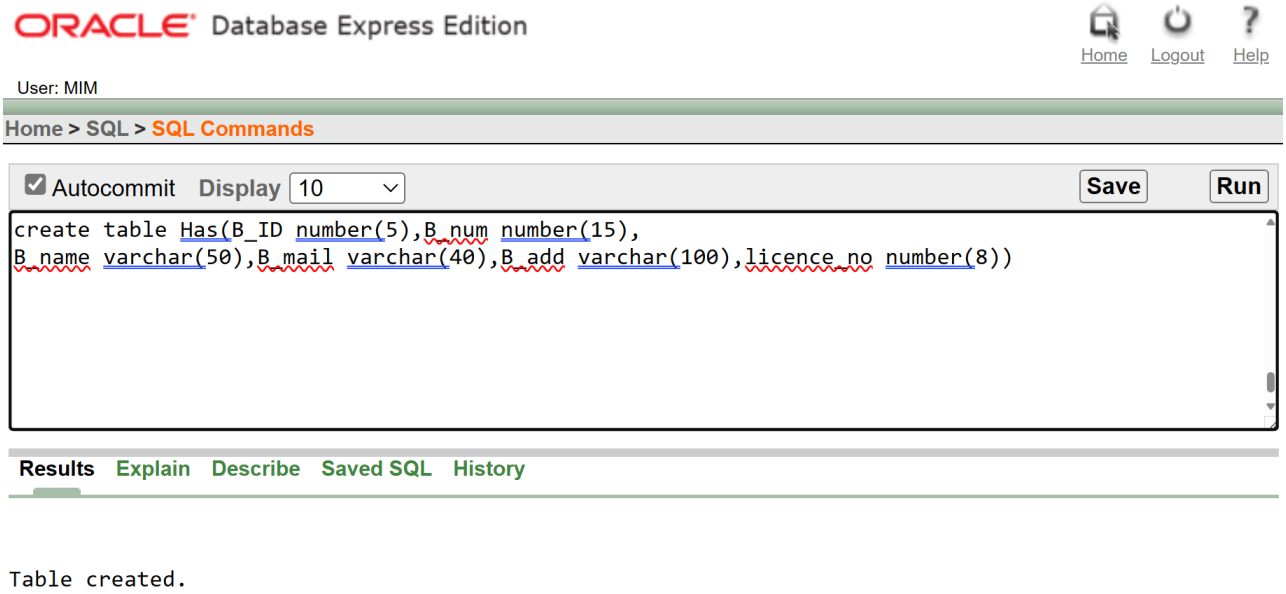

Fig 6.8: Has Table Creation

Value Insertion:

Affiliate Table:

D_ID	LICENCE_NO
101	53034
102	64045
103	75056
104	86067

4 rows returned in 0.00 seconds CSV Export

Fig 7.1: Value insertion in Affiliate Table

Appoint Table:

R_ID	D_ID
201	101
202	102
203	103
204	104

4 rows returned in 0.02 seconds CSV Export

Fig 7.2: Value insertion in Appoint Table

Call Table:

P_NUM	GENDER	AGE	NAME	R_ID
8801746897245	MALE	25	OWAFEEUZZAMAN PATWARY	201
8801846899246	FEMALE	21	NUR-E SARJINA KHAN	202
8801946897247	FEMALE	22	NUSRATJAHAN MIM	203
8801646897248	MALE	35	REZWAN AHMED	204

4 rows returned in 0.02 seconds CSV Export

Fig 7.3: Value insertion in Call Table

Doctor Table:

D_ID	D_NAME	D_NUM	SCHEDULE
102	Dr.Tofazzal Hossain	8801834257289	SUN-TUE
101	Dr.Zahidul Islam	8801913672845	SAT-THU
103	Dr.Munira Ferdousi	8801553859235	MON-THU
104	Dr.Umme Salma	8801957249107	MON-THU

Fig 7.4 :Value insertion in Doctor Table

Has Table:

B_ID	B_NUM	B_NAME	B_MAIL	B_ADD	LICENCE_NO
43035	96137878051	Popular Diagnostics Center Ltd-UTTARA	popular_udc@gmail.com	Uttara	53034
54046	96100096121	Ibn Sina Diagnostic Center-Uttara	ibn_sina@gmail.com	Uttara	64045
65057	96454047291	Uttara Crescent Hospital-2	u_crescent@gmail.com	Sector 7	75056
76068	92070738591	Labaid Diagnostics Uttara	labaid_dc@gmail.com	Sector 13	86067

4 rows returned in 0.02 seconds CSV Export

Fig 7.5: Value insertion in Has Table

Hospital Table:

LICENCE_NO	H_NAME	H_MAIL	H_NUM
64045	Ibn Sina Diagnostic Center	ibn_sina@gmail.com	9610009612
75056	Uttara Crescent Hospital	u_crescent@gmail.com	9645404729
53034	Popular Diagnostics Center Ltd	popular_dc@gmail.com	9613787805
86067	Labaid Diagnostics Uttara	labaid_dc@gmail.com	9207073859

4 rows returned in 0.00 seconds CSV Export

Fig 7.6: Value insertion in Hospital Table

Pay Table:

TRX_ID	GENDER	AGE	NAME	R_ID	P_NUM
56129	MALE	25	OWAFEEUZZAMAN PATWARY	201	8801746897245
78340	FEMALE	21	NUR-E SARJINA KHAN	202	8801846899246
89563	FEMALE	22	NUSRATJAHAN MIM	203	8801946897247
10781	MALE	35	REZWAN AHMED	204	8801646897248

4 rows returned in 0.00 seconds CSV Export

Fig 7.7: Value insertion in Pay Table

Receptionist Table:

R_ID	R_NAME	R_NUM
201	Mahfuzul Islam	8801437635923
202	Zihad Haque	8801737635927
203	Mahin Islam	8801937635977
204	Akbar Ali	8801883763592

4 rows returned in 0.00 seconds CSV Export

Fig 7.8: Value insertion in Receptionist Table

Query Test:

(1)Simple Query:

Show the age of patient named 'Rezwan Ahmed'

Autocommit Display 10

```
select age from call where name='REZWAN AHMED'
```

Results Explain Describe Saved SQL History

AGE
35

1 rows returned in 0.02 seconds CSV Export

Fig 8.1: Query Execution: Simple Query

(2)Single Row Function Query:

Find the B_name,B_add and length of the B_name which is in Uttara.

Autocommit Display 10

```
select b_name,b_add,length(b_name) from has where b_add='Uttara'
```

Results Explain Describe Saved SQL History

B_NAME	B_ADD	LENGTH(B_NAME)
Popular Diagnostics Center Ltd-UTTARA	Uttara	37
Ibn Sina Diagnostic Center-Uttara	Uttara	33

2 rows returned in 0.00 seconds CSV Export

Fig 8.2: Query Execution: Single Row Function Query

(3)Group Function Query:

Find the age of the oldest Patient.

Autocommit Display 10

```
select max(age) from call
```

Results Explain Describe Saved SQL History

MAX(AGE)
35

1 rows returned in 0.00 seconds CSV Export

Fig 8.3: Query Execution: Group Function Query

(4)Single Row Subquery:

Show the name of the patient who is younger than NUSRATJAHAN MIM

Autocommit Display 10 Save

```
select name, age from call where age<(select age from call where name='NUSRATJAHAN MIM')
```

Results Explain Describe Saved SQL History

NAME	AGE
NUR-E SARJINA KHAN	21

1 rows returned in 0.02 seconds CSV Export

Fig 8.4: Query Execution: Single Row Subquery

(5) Multiple Row Subquery:

Show the name of the doctors who do not have the same Schedule as Dr. Munira Ferdousi

Autocommit Display 10

```
select D_name from doctor where
schedule!=(select schedule from doctor where D_name='Dr.Munira Ferdousi')
```

Results Explain Describe Saved SQL History

D_NAME
Dr.Tofazzal Hossain
Dr.Zahidul Islam

2 rows returned in 0.02 seconds CSV Export

Fig 8.5: Query Execution: Multiple Row Subquery

(6.1)Joining- Equijoin:

Join appoint and receptionist table where R_ID is equal.

Autocommit Display 10

```
select receptionist.*,d_id from appoint,
receptionist where appoint.r_id= receptionist.r_id
```

Results Explain Describe Saved SQL History

R_ID	R_NAME	R_NUM	D_ID
201	Mahfuzul Islam	8801437635923	101
202	Zihad Haque	8801737635927	102
203	Mahin Islam	8801937635977	103
204	Akbar Ali	8801883763592	104

4 rows returned in 0.00 seconds CSV Export

Fig 8.6.1: Query Execution: Equijoin

(6.2) Joining- Self Join:

Show Payment Details of the patient who are younger than Rezwan Ahmed using selfjoin.

Autocommit Display 10 Save Run

```
select A.* from pay N, pay A where N.name='REZWAN AHMED'
and A.age< N.age
```

Results Explain Describe Saved SQL History

TRX_ID	GENDER	AGE	NAME	R_ID	P_NUM
56129	MALE	25	OWAFEEUZZAMAN PATWARY	201	8801746897245
78340	FEMALE	21	NUR-E SARJINA KHAN	202	8801846899246
89563	FEMALE	22	NUSRATJAHAN MIM	203	8801946897247

3 rows returned in 0.00 seconds CSV Export

Fig 8.6.2: Query Execution: Self-Join

(7.1) Simple View:

Create a view named Malepatient where the gender of the patient will be MALE.

Query: create view Malepatient as select name,gender from call where gender='MALE'

Autocommit Display 10

```
create view Malepatient as select name,gender from call where gender = 'MALE'
```

Results Explain Describe Saved SQL History

```
View created.
```

Fig 8.7.1.1: Simple view creation command

Results Explain Describe Saved SQL History

Object Type **VIEW** Object **MALEPATIENT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MALEPATIENT	NAME	Varchar2	50	-	-	-	✓	-	-
	GENDER	Varchar2	6	-	-	-	✓	-	-
									1 - 2

Fig 8.7.1.2: Description of the simple view

Results Explain Describe Saved SQL History

NAME	GENDER
OWAFEEUZZAMAN PATWARY	MALE
REZWAN AHMED	MALE

2 rows returned in 0.00 seconds CSV Export

Fig 8.7.1.3: Result of the simple view as a whole table

(7.2)Complex View:

Create a view of the Pay table which shows name and trx_id with Self-join

Query: CREATE VIEW TransectionInfo AS SELECT p1.name, p2.trx_id FROM Pay p1, Pay p2 where p1.p_num = p2.p_num

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
CREATE VIEW TransectionInfo AS SELECT p1.name, p2.trx_id
FROM Pay p1, Pay p2 where p1.p_num = p2.p_num
```

Results Explain Describe Saved SQL History

```
View created.
```

Fig 8.7.2.1: Complex view creation command

Results Explain Describe Saved SQL History

Object Type **VIEW** Object **TRANSECTIONINFO**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TRANSECTIONINFO	NAME	Varchar2	50	-	-	-	✓	-	-
	TRX_ID	Number	-	5	0	-	✓	-	-
									1 - 2

Fig 8.7.2.2: Description of the complex view

Results Explain Describe Saved SQL History

NAME	TRX_ID
OWAFEEUZZAMAN PATWARY	56129
NUR-E SARJINA KHAN	78340
NUSRATJAHAN MIM	89563
REZWAN AHMED	10781

4 rows returned in 0.00 seconds CSV Export

Fig 8.7.2.3: Result of the complex view as a whole table

Database Connection:

IDE used: Notepad++ ,version 8.5.6

Database Management System: Oracle 10g Express Edition

Java Version: 20.0.1

Java Connector: JDBC Oracle 14.jar

Affiliate Table database connection:

OracleConnJava.java

```java
import java.sql.*;

class OracleConnJava {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "browny");
            Statement stmt = con.createStatement();

            ResultSet rs = stmt.executeQuery("select * from Affiliate");

            while (rs.next()) {
                System.out.println(rs.getInt(1) + " " + rs.getInt(2));
            }

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

```
101 53034
102 64045
103 75056
104 86067
```

Appoint Table database connection:

OracleConnJava.java

```java

import java.sql.*;

class OracleConnJava {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "browny");
            Statement stmt = con.createStatement();

            ResultSet rs = stmt.executeQuery("select * from Appoint");

            while (rs.next()) {
                System.out.println(rs.getInt(1) + " " + rs.getInt(2));
            }

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}

```

Output:

```
201 101
202 102
203 103
204 104
```

Has Table database connection:

ConnJava.java

```java
import java.sql.*;

class OracleConnJava {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "browny");
            Statement stmt = con.createStatement();

            ResultSet rs = stmt.executeQuery("select * from Has");

            while (rs.next()) {
                System.out.println(rs.getInt(1) + "  " + rs.getLong(2) + "   " + rs.getString(3) + "    " + rs.getString(4) + "    " + rs.getString(5) + "    " + rs.getInt(6));
            }

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

```
43035  96137878051   Popular Diagnostics Center Ltd-UTTARA    popular_udc@gmail.com    Uttara    53034
54046  96100096121   Ibn Sina Diagnostic Center-Uttara    ibn_sina@gmail.com    Uttara    64045
65057  96454047291   Uttara Crescent Hospital-2    u_crescent@gmail.com    Sector 7    75056
76068  92070738591   Labaid Diagnostics Uttara    labaid_dc@gmail.com    Sector 13    86067
```

Hospital Table database connection:

*D:\Downloads\project\Project\Project\JavaGUI\New folder\OracleConnJava.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

OracleConnJava.java

```java
import java.sql.*;

class OracleConnJava {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "browny");
            Statement stmt = con.createStatement();

            ResultSet rs = stmt.executeQuery("select * from Hospital");

            while (rs.next()) {
                System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3) + " " + rs.getLong(4));
            }

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

```
53034 Popular Diagnostics Center Ltd popular_dc@gmail.com 9613787805
64045 Ibn Sina Diagnostic Center ibn_sina@gmail.com 9610009612
75056 Uttara Crescent Hospital u_crescent@gmail.com 9645404729
86067 Labaid Diagnostics Uttara labaid_dc@gmail.com 9207073859
```

Doctor Table database connection:

```
eConnJava.java
import java.sql.*;

class OracleConnJava {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "browny");
            Statement stmt = con.createStatement();

            ResultSet rs = stmt.executeQuery("select * from Doctor");

            while (rs.next()) {
                System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getLong(3) + " " + rs.getString(4));
            }

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

```
102 Dr.Tofazzal Hossain 8801834257289 SUN-TUE
101 Dr.Zahidul Islam 8801913672845 SAT-THU
103 Dr.Munira Ferdousi 8801553859235 MON-THU
104 Dr.Umme Salma 8801957249107 MON-THU
```

Pay Table database connection:

```
eConnJava.java

import java.sql.*;

class OracleConnJava {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "browny");
            Statement stmt = con.createStatement();

            ResultSet rs = stmt.executeQuery("select * from Pay");

            while (rs.next()) {
                System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getInt(3) + " " + rs.getString(4) + " " + rs.getInt(5) + " " + rs.getLong(6));
            }

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

```
56129 MALE 25 OWAFEEUZZAMAN PATWARY 201 8801746897245
78340 FEMALE 21 NUR-E SARJINA KHAN 202 8801846899246
89563 FEMALE 22 NUSRATJAHAN MIM 203 8801946897247
10781 MALE 35 REZWAN AHMED 204 8801646897248
```

Receptionist Table database connection:

```
cleConnJava.java
import java.sql.*;

class OracleConnJava {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "browny");
            Statement stmt = con.createStatement();

            ResultSet rs = stmt.executeQuery("select * from Receptionist");

            while (rs.next()) {
                System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getLong(3));
            }

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

```
201 Mahfuzul Islam 8801437635923
202 Zihad Haque 8801737635927
203 Mahin Islam 8801937635977
204 Akbar Ali 8801883763592
```

Call Table database connection:

ConnJava.java

```java
import java.sql.*;

class OracleConnJava {
    public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "browny");
            Statement stmt = con.createStatement();

            ResultSet rs = stmt.executeQuery("select * from Call");

            while (rs.next()) {
                System.out.println(rs.getLong(1) + " " + rs.getString(2) + " " + rs.getInt(3) + " " + rs.getString(4) + " " + rs.getInt(5));
            }

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

```
8801746897245 MALE 25 OWAFEEUZZAMAN PATWARY 201
8801846899246 FEMALE 21 NUR-E SARJINA KHAN 202
8801946897247 FEMALE 22 NUSRATJAHAN MIM 203
8801646897248 MALE 35 REZWAN AHMED 204
```

Conclusion:

The “Doctor Appointment Management System” can store various data about different Hospitals, Branches, Doctors, Receptionists and Patients and also constantly update data and information about them. The data can be properly organized in different tables and easily accessed. This can also be used throught Java applications.