



**Project Title: Birthdate Record Management
System**

**Course: Advanced Database Management
System**

Section: B

Submitted to

Juena Ahmed
Noshin

GROUP DETAILS

NAME	ID	Contribution
ADEEPTA SHUSHIL SHOVO	20-43092-1	25%
MD. YEASIN RAHAT	20-43097-1	25%
MD.ASIF AL MAMUN	20-43132-1	25%
A.S.M RIFAT	20-43767-2	25%

Contents

Topic Name	Page Number
Introduction	3
Project proposal	3-5
Diagram	5-7
User interface	8-10
Scenario Description	11-12
ER Diagram	13
Normalization	14-20
Schema Diagram	21
User Create	22
Table Crate	22-25
Data Insertion	26-31
Query Writing (SQL)	32-41
Query Writing (PL/SQL)	42-58
Relational Algebra	59
Conclusion	59-61

Introduction:

The Birth Record Management System is a project that is being considered with the goal of developing a thorough and effective system for managing birth records. Birth-related data will be stored, retrieved, and managed more easily with the help of the system, which will also ensure accuracy, security, and convenience for authorized users. By putting this system into place, we hope to increase data accuracy, expedite the birth registration procedure, and improve overall record management within the designated organization or jurisdiction.

Project Proposal:

Scope:

The following attributes will be present in the birth record management system:

Objectives:

- a. Efficient Data Management: Create a centralized database to securely store and maintain birth records, doing away with the need for paper forms and lowering the likelihood of data loss or errors.
- b. Simplified Birth Registration Process: Create a user-friendly interface for newborn registration that will make it simple for parents, medical staff, and authorized individuals to submit and confirm birth details.

- c. Secure Data Access and Authorization: Implement strong authentication and access control measures to guarantee that only people with the proper authorization can access, read, or extract birth records, safeguarding the confidentiality and security of sensitive data.
- d. Enhancing reporting and analytics tools will help with decision-making and resource allocation by generating statistical reports, demographic analyses, and trends connected to births.
- e. Integration with Existing Systems: To guarantee data consistency and interoperability, seamlessly interface the Birth Record Management System with other pertinent systems, such as hospital information systems or vital statistics databases.

**Feature
s:**

- a. Birth Registration: Create a user-friendly interface for registering newborns that collects necessary data, including the child's name, birthplace, parents' names, and other mandatory fields.
- b. Record management: Provide efficient search, retrieval, updating, and archiving of birth records for authorized staff, protecting data integrity and reducing duplication.
- c. Document submit: To ensure thorough record-keeping, permit users to submit supporting papers such birth certificates, medical records, and identification documents.
- d. Data validation: Use validation checks to guarantee that entered data is accurate and full, lowering errors and raising the standard of birth records as a whole.

- e. Reporting and Analytics: Create individualized reports and statistical evaluations using birth records to reveal insights into birth patterns, population expansion, and other pertinent data.
- f. Security and privacy: To protect sensitive data from unauthorized access or manipulation, use strong security measures, such as encryption, access limits, and audit trails.
- g. Integration: Establishing data interfaces and integration protocols will allow for easy data sharing with other systems while assuring data consistency and preventing duplication..

Class Diagram, Use Case Diagram, Activity Diagram:

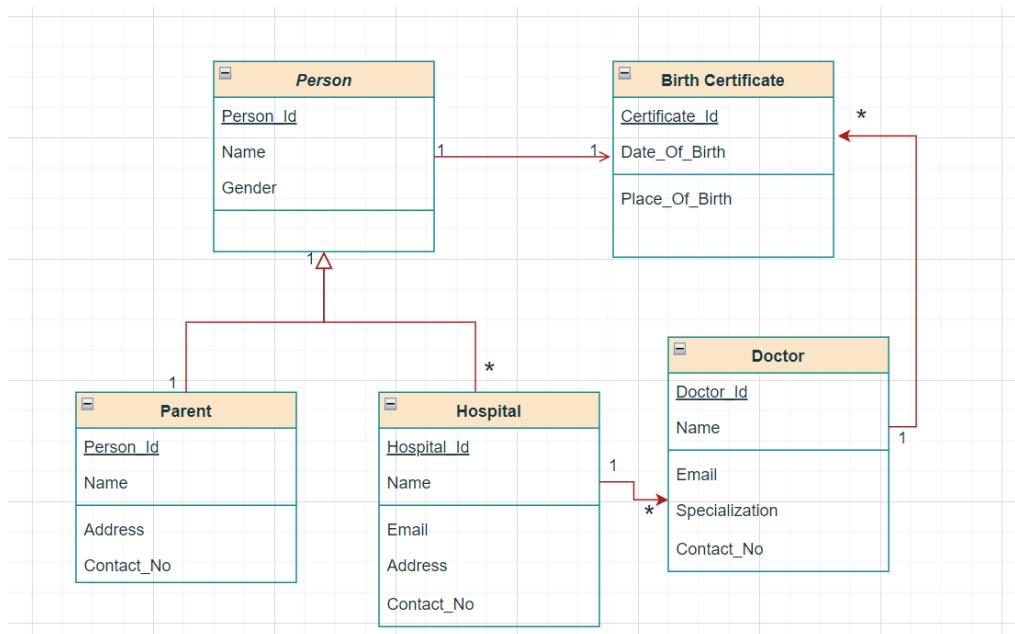


Figure 1: Class Diagram

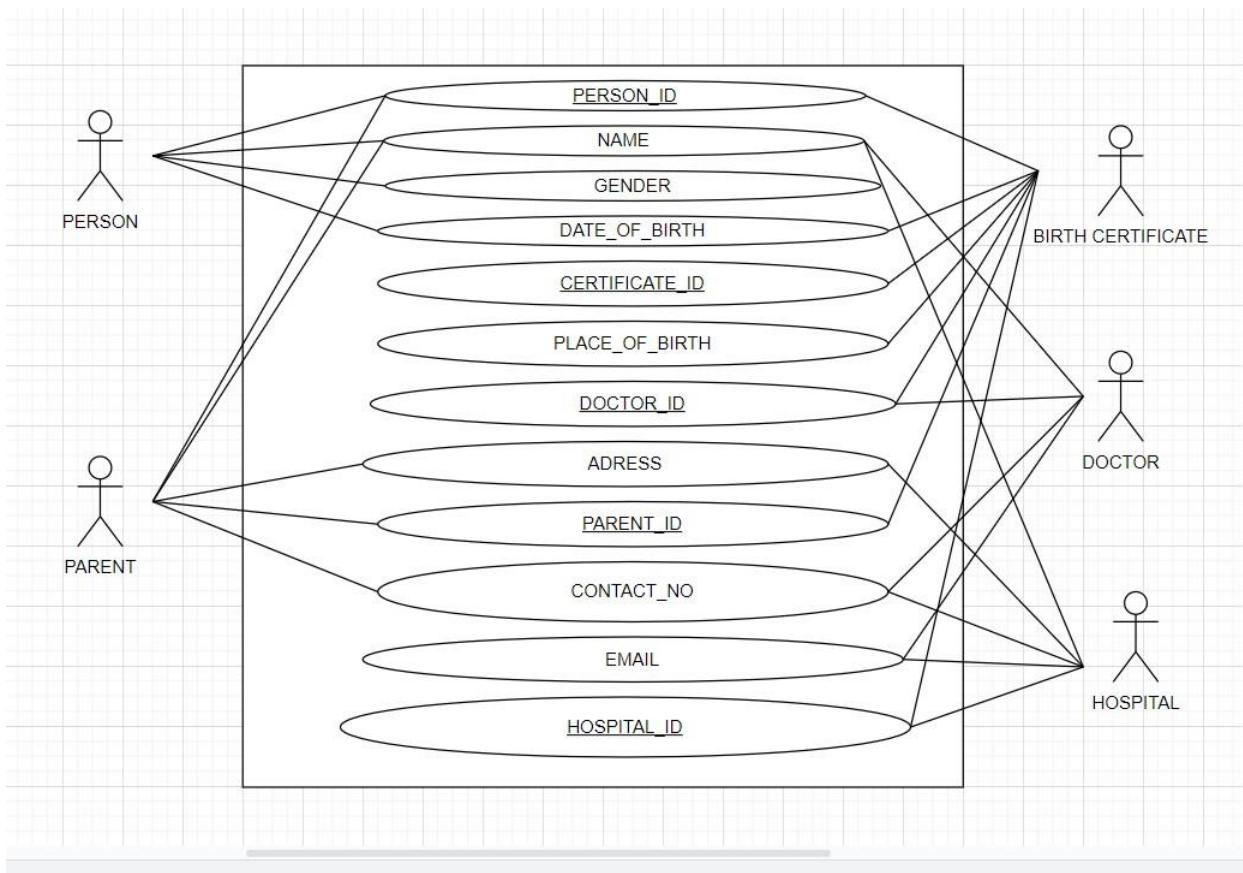


Figure 2: Use Case diagram

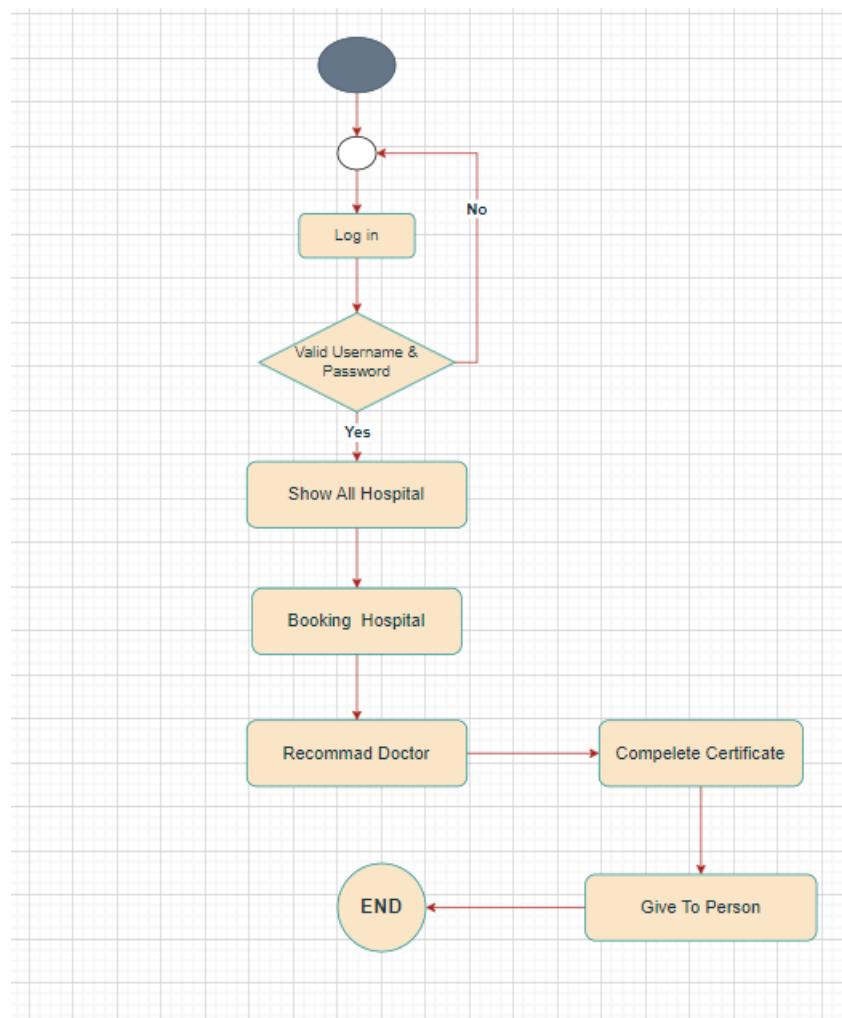


Figure 3: Activity Diagram

User Interface:

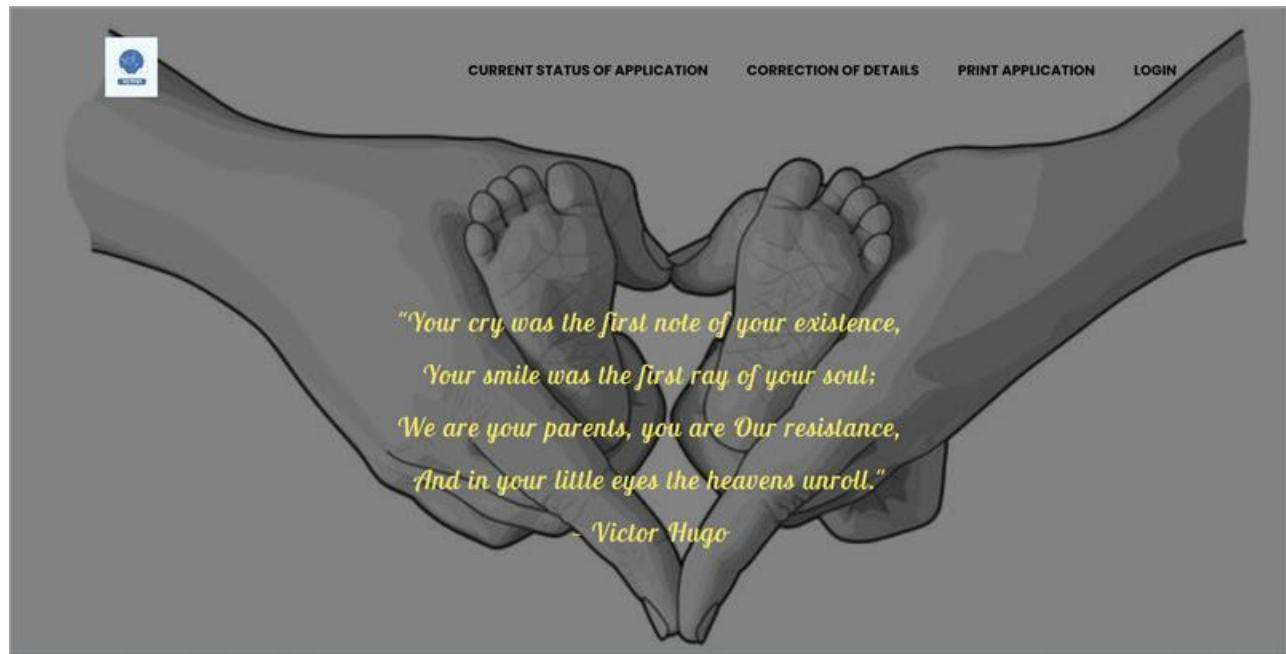
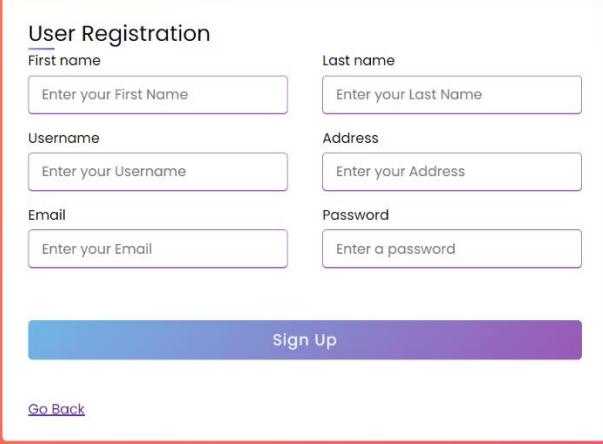
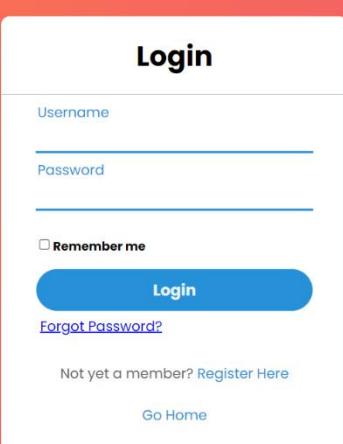


Fig: Home page



The registration page features a white header with the title "User Registration". Below the header are six input fields arranged in two columns of three. The first column contains "First name" (placeholder: "Enter your First Name") and "Username" (placeholder: "Enter your Username"). The second column contains "Last name" (placeholder: "Enter your Last Name") and "Address" (placeholder: "Enter your Address"). The third column contains "Email" (placeholder: "Enter your Email") and "Password" (placeholder: "Enter a password"). A large blue "Sign Up" button spans the width of the middle two columns. At the bottom left is a "Go Back" link.

Fig: Registration page



The login page has a white header with the word "Login" in bold black text. Below the header are two input fields: "Username" and "Password", each with a horizontal line underneath. There is also a "Remember me" checkbox. A large blue "Login" button is centered below the inputs. To the right of the button is a "Forgot Password?" link. At the bottom of the page, there is a "Not yet a member? Register Here" link and a "Go Home" link.

Fig: Log in page



[CURRENT STATUS OF APPLICATION](#) [CORRECTION OF DETAILS](#) [PRINT APPLICATION](#) [LOGIN](#)

Application Id

Date of Birth

Submit

Need Help ?
We are Always here for you!
Call our Hotline (10AM - 10PM).

Contact Us:
info@birth.com
+88 xxxxx xxxx

2000-2023 © Birth Record Management System POWERED BY CoinVerse

Fig: Application page

Scenario Description:

User Registration and Login:

Administrators and other designated hospital staff members would have their user accounts registered in the system. Using their login information, they could access the birth record management system.

Recording a Birth:

A hospital employee would go into the system and go to the "Birth Certificate" area whenever a new baby was born. They would start the record-creation process by entering pertinent information, such as the newborn's name, birthdate, gender, and any other necessary facts. The method would create a special identification number (certificate ID) for the birth certificate and link it to the infant.

Adding Parent Information:

The staff person would next input the parents' information, including their names, residences, phone numbers, and any other pertinent information. The system would save this data in the "Parent" table and give each parent a special ID (parent ID). Through foreign key links, the parents' person IDs would be connected to the birth certificate.

Doctor and Hospital Details:

The staff person would next note the information about the doctor or other healthcare provider who assisted with the delivery of the child. Name, area of expertise, contact information, and a special identification number (doctor ID) would all be included. The staff person would also input facts about the hospital or healthcare facility where the birth occurred, including its name, address, phone number, and other contact information, as well as a special identification code (hospital ID). Through foreign key restrictions, these tables would connect to the birth certificate table.

Accessing and Retrieving Records:

By using the search capabilities provided by the Birth Record Management System, authorized users will be able to locate birth records using a variety of parameters. For instance, hospital workers might look up data using the baby's name, birthdate, parents' names, or even the doctor's name. The system would do the searches and show the pertinent birth records and related information.

Updating and Modifying Records:

Authorized users can access the individual birth certificate and amend the appropriate data in the event that the birth record has to be changed or updated, such as to fix a misspelled name or update contact information. By updating the corresponding tables and upholding the linkages between entities, the system would guarantee data integrity.

Reporting and Analytics:

In order to provide statistical reports and analytics on birth records, the birth record management system would have reporting capabilities. Authorized users might produce reports on demographics, birth patterns, or any other pertinent information needed by the hospital management or governmental organizations.

ER Diagram:

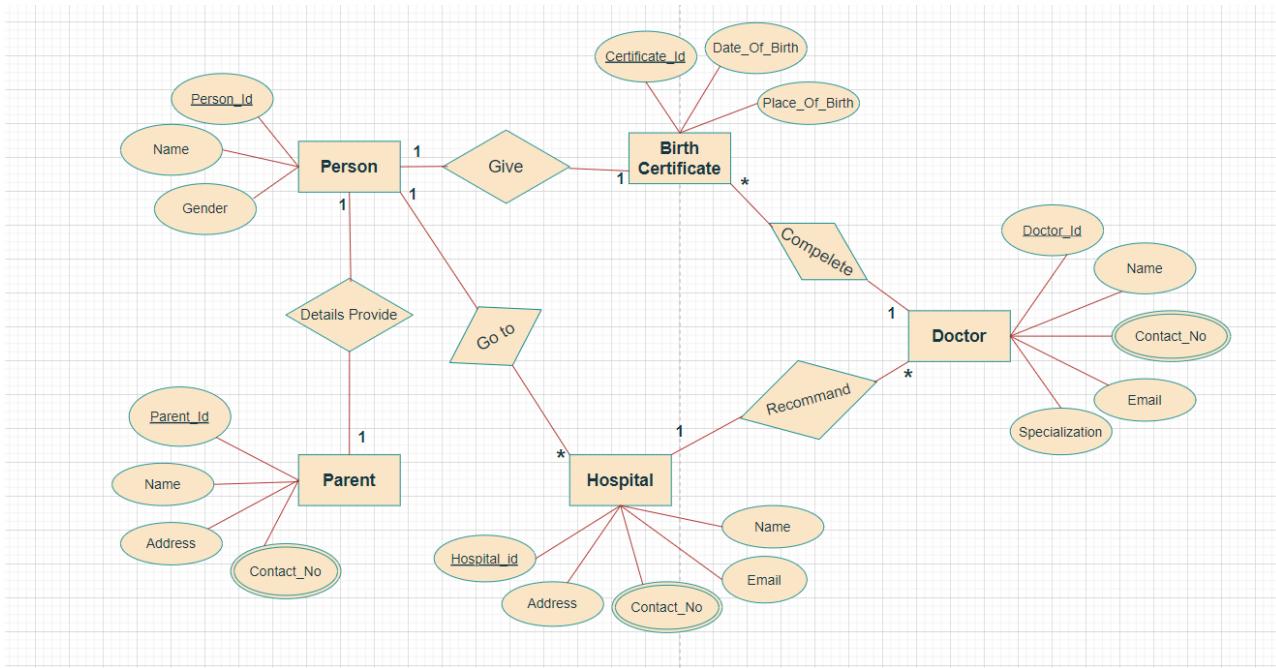


Figure 4: ER Diagram

Normalization:

Details Provide:

UNF

Details_Provide(Person_Id, Name, Gender, Parent_Id, Name, Address, Contact_No)

1NF

Contact No is a multi valued attribute.

1. Person_Id, Name, Gender, Parent_Id, Name, Address, Contact_No

2NF

1. Person_Id, Name, Gender

2. Parent_Id, Name, Address, Contact_No

3NF

There is no transitive dependency. Relation Already in 3NF.

1. Person_Id, Name, Gender

2. Parent_Id, Name, Address, Contact_No

Table Creation

1. Person_Id, Name, Gender, Parent_Id

2. Parent_Id, Name, Address, Contact_No

Go To:

UNF

Go_To(Person_Id, Name, Gender, Hospital_Id, Name, Email, Address, Contact_No)

1NF

Contact No is a multi valued attribute.

1. Person_Id, Name, Gender, Hospital_Id, Name, Email, Address, Contact_No

2NF

1. Person_Id, Name, Gender

2. Hospital_Id, Name, Email, Address, Contact_No

3NF

There is no transitive dependency. Relation Already in 3NF.

1. Person_Id, Name, Gender
2. Hospital_Id, Name, Email, Address, Contact_No

Table Creation

1. Person_Id, Name, Gender, Hospital_Id
2. Hospital_Id, Name, Email, Address, Contact_No

Recommend:

UNF

Recommend(Specialization, Doctor_Id, Name, Email, Contact_No, Hospital_Id, Name, Email, Address, Contact_No)

1NF

Contact No is a multi valued attribute.

1. Specialization, Doctor_Id, Name, Email, Contact_No, Hospital_Id, Name, Email, Address, Contact_No

2NF

1. Specialization, Doctor_Id, Name, Email, Contact_No

2. Hospital_Id, Name, Email, Address, Contact_No

3NF

There is no transitive dependency. Relation Already in 3NF.

1. Specialization, Doctor_Id, Name, Email, Contact_No

2. Hospital_Id, Name, Email, Address, Contact_No

Table Creation

1. Specialization, Doctor_Id, Name, Email, Contact_No

2. Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id

Complete:

UNF

Complete(Specialization, Doctor_Id, Name, Email, Contact_No, Certificate_Id, Date_Of_Birth, Place_Of_Birth)

1NF

Contact No is a multi valued attribute.

1. Specialization, Doctor_Id, Name, Email, Contact_No, Certificate_Id, Date_Of_Birth, Place_Of_Birth

2NF

1. Specialization, Doctor_Id, Name, Email, Contact_No

2. Certificate_Id, Date_Of_Birth, Place_Of_Birth

3NF

There is no transitive dependency. Relation Already in 3NF.

1. Specialization, Doctor_Id, Name, Email, Contact_No

2. Certificate_Id, Date_Of_Birth, Place_Of_Birth

Table Creation

1. Specialization, Doctor_Id, Name, Email, Contact_No, Certificate_Id
2. Certificate_Id, Date_Of_Birth, Place_Of_Birth

Give:

UNF

Give(Person_Id, Name, Gender, Certificate_Id, Date_Of_Birth, Place_Of_Birth)

1NF

There is no multi valued attribute. Relation Already in 1NF.

1. Person_Id, Name, Gender, Certificate_Id, Date_Of_Birth, Place_Of_Birth

2NF

1. Person_Id, Name, Gender

2. Certificate_Id, Date_Of_Birth, Place_Of_Birth

3NF

There is no transitive dependency. Relation Already in 3NF.

1. Person_Id, Name, Gender

2. Certificate_Id, Date_Of_Birth, Place_Of_Birth

Table Creation

1. Person_Id, Name, Gender

2. Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id

Temporary Tables:

1. Person_Id, Name, Gender, Parent_Id

2. Parent_Id, Name, Address, Contact_No

3. Person_Id, Name, Gender, Hospital_Id

4. Hospital_Id, Name, Email, Address, Contact_No

5. Specialization, Doctor_Id, Name, Email, Contact_No

6. Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id

7. Specialization, Doctor_Id, Name, Email, Contact_No, Certificate_Id

8. Certificate_Id, Date_Of_Birth, Place_Of_Birth

9. Person_Id, Name, Gender

10. Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id

Final Tables:

1. Person_Id, Name, Gender, Parent_Id, Hospital_Id
2. Parent_Id, Name, Address, Contact_No
3. Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id
4. Specialization, Doctor_Id, Name, Email, Contact_No, Certificate_Id
5. Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id

Schema Diagram:

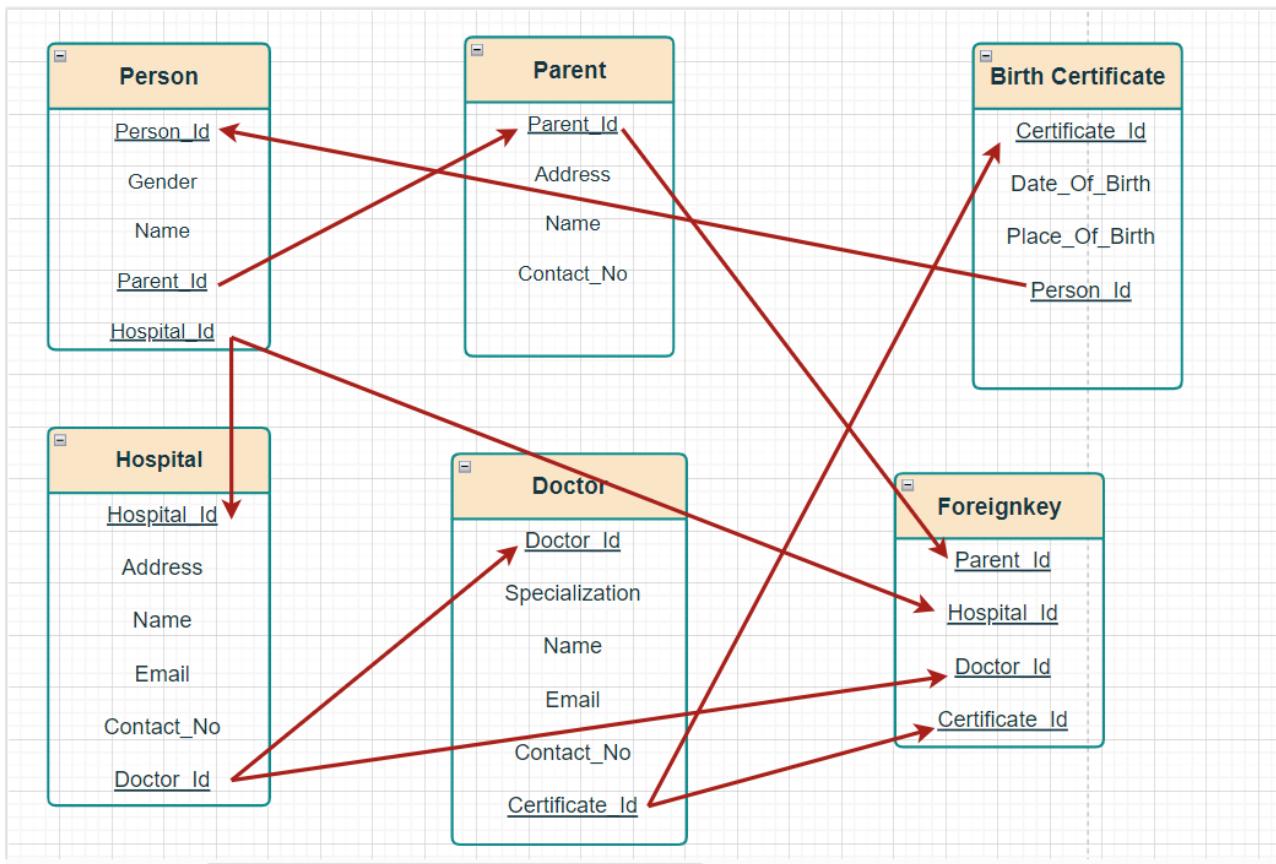
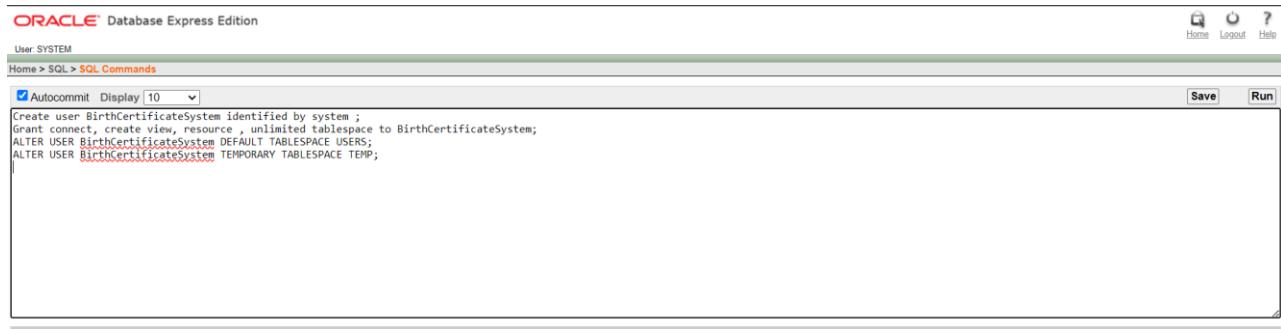


Figure 5: Schema Diagram

Table Creation:

Create User

```
Create user BirthCertificateSystem identified by system ;
Grant connect, create view, resource , unlimited tablespace to BirthCertificateSystem;
ALTER USER BirthCertificateSystem DEFAULT TABLESPACE USERS;
ALTER USER BirthCertificateSystem TEMPORARY TABLESPACE TEMP;
```



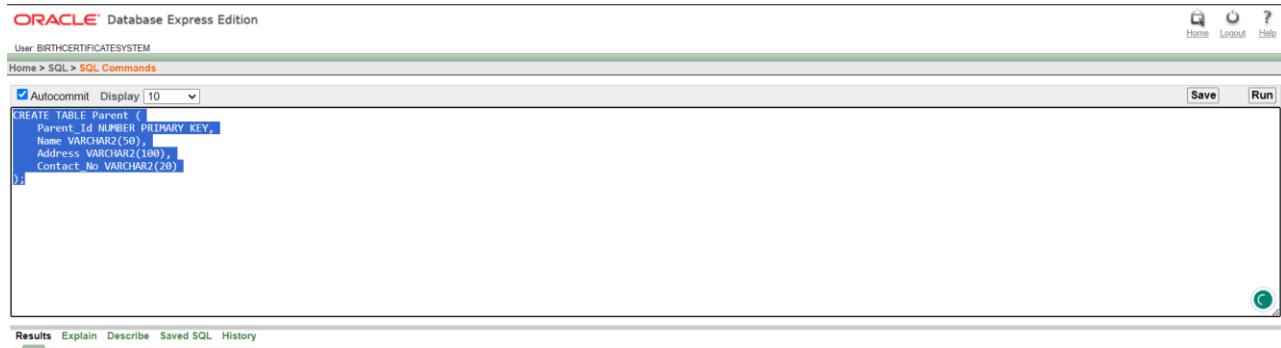
The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The menu bar has "User SYSTEM" and "Home > SQL > SQL Commands". The main area contains the following SQL code:

```
Create user BirthCertificateSystem identified by system ;
Grant connect, create view, resource , unlimited tablespace to BirthCertificateSystem;
ALTER USER BirthCertificateSystem DEFAULT TABLESPACE USERS;
ALTER USER BirthCertificateSystem TEMPORARY TABLESPACE TEMP;
```

There are "Save" and "Run" buttons at the top right.

Table Create

```
1. CREATE TABLE Parent (
    Parent_Id NUMBER PRIMARY KEY,
    Name VARCHAR2(50),
    Address VARCHAR2(100),
    Contact_No VARCHAR2(20)
);
```



The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The menu bar has "User BIRTHCERTIFICATESYSTEM" and "Home > SQL > SQL Commands". The main area contains the following SQL code:

```
CREATE TABLE Parent (
    Parent_Id NUMBER PRIMARY KEY,
    Name VARCHAR2(50),
    Address VARCHAR2(100),
    Contact_No VARCHAR2(20)
);
```

There are "Save" and "Run" buttons at the top right. A green circular progress indicator is visible in the bottom right corner.

Table created.

```

2. CREATE TABLE Person (
    Person_Id NUMBER PRIMARY KEY,
    Name VARCHAR2(50),
    Gender VARCHAR2(10),
    Parent_Id NUMBER,
    CONSTRAINT fk_Parent FOREIGN KEY (Parent_Id) REFERENCES Parent(Parent_Id)
);

```

```

ALTER TABLE Person
ADD Hospital_Id NUMBER;

```

```

ALTER TABLE Person
ADD CONSTRAINT fk_Hospital FOREIGN KEY (Hospital_Id) REFERENCES
Hospital(Hospital_Id);

```

The screenshot shows the Oracle Database Express Edition interface. The top navigation bar includes 'Home', 'Logout', and 'Help' buttons. The main area displays the SQL command for creating the PERSON table, followed by the results of the 'DESC Person' command, which shows the table structure with columns: PERSON_ID, NAME, GENDER, PARENT_ID, and HOSPITAL_ID.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PERSON	PERSON_ID	Number	-	-	-	1	-	-	-
	NAME	VARCHAR2	50	-	-	-	✓	-	-
	GENDER	VARCHAR2	10	-	-	-	✓	-	-
	PARENT_ID	Number	-	-	-	-	✓	-	-
	HOSPITAL_ID	Number	-	-	-	-	✓	-	-

```

3. CREATE TABLE BirthCertificate (
    Certificate_Id NUMBER PRIMARY KEY,
    Date_Of_Birth DATE,
    Place_Of_Birth VARCHAR2(100),

```

```

Person_Id NUMBER,
CONSTRAINT fk_Person FOREIGN KEY (Person_Id) REFERENCES Person(Person_Id)
);

```

The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The navigation bar includes "Home", "Logout", and "Help". The main menu bar has "User BIRTHCERTIFICATESYSTEM", "Home > SQL > SQL Commands". A toolbar below the menu has "Autocommit" checked, "Display 10", "Save", and "Run". The query window contains the SQL code for creating the BIRTHCERTIFICATE table. Below the query window is a results grid showing the table structure:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BIRTHCERTIFICATE	CERTIFICATE_ID	Number	-	-	-	1	-	-	-
	DATE_OF_BIRTH	Date	7	-	-	-	✓	-	-
	PLACE_OF_BIRTH	Varchar2	100	-	-	-	✓	-	-
	PERSON_ID	Number	-	-	-	-	✓	-	-

At the bottom of the results grid, it says "1 - 4".

```

4. CREATE TABLE Doctor (
Doctor_Id NUMBER PRIMARY KEY,
Name VARCHAR2(100),
Email VARCHAR2(50),
Contact_No VARCHAR2(20),
Specialization VARCHAR2(50),
Certificate_Id NUMBER,
CONSTRAINT fk_Certificate FOREIGN KEY (Certificate_Id) REFERENCES BirthCertificate
(Certificate_Id)
);

```

The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The navigation bar includes "Home", "Logout", and "Help". The main menu bar has "User BIRTHCERTIFICATESYSTEM", "Home > SQL > SQL Commands". A toolbar below the menu has "Autocommit" checked, "Display 10", "Save", and "Run". The query window contains the SQL code for creating the DOCTOR table. Below the query window is a results grid showing the table structure:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DOCTOR	DOCTOR_ID	Number	-	-	-	1	-	-	-
	NAME	Varchar2	100	-	-	-	✓	-	-
	EMAIL	Varchar2	50	-	-	-	✓	-	-
	CONTACT_NO	Varchar2	20	-	-	-	✓	-	-
	SPECIALIZATION	Varchar2	50	-	-	-	✓	-	-
	CERTIFICATE_ID	Number	-	-	-	-	✓	-	-

At the bottom of the results grid, it says "1 - 6".

```

5. CREATE TABLE Hospital (
    Hospital_Id NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
    Email VARCHAR2(50),
    Address VARCHAR2(200),
    Contact_No VARCHAR2(20),
    Doctor_Id NUMBER,
    CONSTRAINT fk_Doctor FOREIGN KEY (Doctor_Id) REFERENCES Doctor(Doctor_Id)
);

```

The screenshot shows the Oracle Database Express Edition interface. The title bar reads "ORACLE Database Express Edition". The menu bar includes "File", "Edit", "View", "Tools", "Help", and a "User" dropdown set to "BIRTHCERTIFICATESYSTEM". The main navigation bar has links for "Home", "SQL", and "SQL Commands". A toolbar below the navigation bar includes "Autocommit" (checked), "Display 10", "Save", and "Run". The main area is titled "DESC Hospital" and contains a large empty text box. Below this is a table titled "Object Type TABLE Object HOSPITAL" with columns: Table, Column, Data Type, Length, Precision, Scale, Primary Key, Nullable, Default, and Comment. The table data is as follows:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
HOSPITAL	HOSPITAL_ID	Number	-	-	-	1	✓	-	-
	NAME	VARCHAR2	100	-	-	-	✓	-	-
	EMAIL	VARCHAR2	50	-	-	-	✓	-	-
	ADDRESS	VARCHAR2	200	-	-	-	✓	-	-
	CONTACT_NO	VARCHAR2	20	-	-	-	✓	-	-
	DOCTOR_ID	Number	-	-	-	-	✓	-	-

At the bottom right of the table area, it says "1 - 6". In the bottom right corner of the entire window, it says "Application Express 21.0.0.0.39".

Data Insertion

For Table 1:

```
INSERT INTO Parent (Parent_Id, Name, Address, Contact_No)
VALUES (1, 'Adeepta', 'Nikunja-2', '555-1234');
```

```
INSERT INTO Parent (Parent_Id, Name, Address, Contact_No)
VALUES (2, 'Asif', 'Bashundhara', '555-5678');
```

```
INSERT INTO Parent (Parent_Id, Name, Address, Contact_No)
VALUES (3, 'Yeasin', 'Kuril', '555-7890');
```

```
INSERT INTO Parent (Parent_Id, Name, Address, Contact_No)
VALUES (4, 'Rifat', 'Uttara', '555-4567');
```

```
INSERT INTO Parent (Parent_Id, Name, Address, Contact_No)
VALUES (5, 'Rahat', 'Badda', '555-2222');
```

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

```

 AutoCommit   
INSERT INTO Parent (Parent_Id, Name, Address, Contact_No)
VALUES (1, 'Adepta', 'Nikunja-2', '555-1234');

INSERT INTO Parent (Parent_Id, Name, Address, Contact_No)
VALUES (2, 'Asif', 'Bashundhara', '555-5678');

INSERT INTO Parent (Parent_Id, Name, Address, Contact_No)
VALUES (3, 'Yeasin', 'Kuril', '555-7890');

INSERT INTO Parent (Parent_Id, Name, Address, Contact_No)
VALUES (4, 'Rifat', 'Uttara', '555-4567');

INSERT INTO Parent (Parent_Id, Name, Address, Contact_No)
VALUES (5, 'Rahat', 'Badda', '555-2222');

Select * From Parent

```

Results Explain Describe Saved SQL History

PARENT_ID	NAME	ADDRESS	CONTACT_NO
1	Adepta	Nikunja-2	555-1234
2	Asif	Bashundhara	555-5678
3	Yeasin	Kuril	555-7890
4	Rifat	Uttara	555-4567
5	Rahat	Badda	555-2222

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

For Table 2:

```
INSERT INTO Person (Person_Id, Name, Gender, Parent_Id, Hospital_Id)
VALUES (1, 'Adepta', 'Male', 1, 1);
```

```
INSERT INTO Person (Person_Id, Name, Gender, Parent_Id, Hospital_Id)
VALUES (2, 'Yeasin', 'Male', 2, 2);
```

```
INSERT INTO Person (Person_Id, Name, Gender, Parent_Id, Hospital_Id)
VALUES (3, 'Asif', 'Male', 3, 3);
```

```
INSERT INTO Person (Person_Id, Name, Gender, Parent_Id, Hospital_Id)
VALUES (4, 'Rifat', 'Male', 4, 4);
```

```
INSERT INTO Person (Person_Id, Name, Gender, Parent_Id, Hospital_Id)
VALUES (5, 'Rahat', 'Male', 5, 5);
```

The screenshot shows the MySQL Workbench interface. At the top, it says "WIKALLE Database express caption" and "User BIRTHCERTIFICATESYSTEM". There are links for "Home", "Logout", and "Help". Below that, it says "Home > SQL > SQL Commands". A dropdown menu shows "Autoconnect" and "Display 10". On the right, there are "Save" and "Run" buttons. The SQL pane contains the following code:

```

INSERT INTO Person (Person_Id, Name, Gender, Parent_Id, Hospital_Id)
VALUES (2, 'Yasin', 'Male', 2, 2);
INSERT INTO Person (Person_Id, Name, Gender, Parent_Id, Hospital_Id)
VALUES (3, 'Asif', 'Male', 3, 3);
INSERT INTO Person (Person_Id, Name, Gender, Parent_Id, Hospital_Id)
VALUES (4, 'Rifat', 'Male', 4, 4);
INSERT INTO Person (Person_Id, Name, Gender, Parent_Id, Hospital_Id)
VALUES (5, 'Rahat', 'Male', 5, 5);

Select * From Person;

```

The "Results" tab is selected, showing the following table:

PERSON_ID	NAME	GENDER	PARENT_ID	HOSPITAL_ID
1	Adeeqa	Male	1	1
2	Yasin	Male	2	2
3	Asif	Male	3	3
4	Rifat	Male	4	4
5	Rahat	Male	5	5

Below the table, it says "5 rows returned in 0.00 seconds" and has a "CSV Export" link.

For Table 3:

```
INSERT INTO BirthCertificate (Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id)
VALUES (1, TO_DATE('2000-01-15', 'YYYY-MM-DD'), 'City A', 1);
```

```
INSERT INTO BirthCertificate (Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id)
VALUES (2, TO_DATE('1998-05-02', 'YYYY-MM-DD'), 'City B', 2);
```

```
INSERT INTO BirthCertificate (Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id)
VALUES (3, TO_DATE('2005-11-10', 'YYYY-MM-DD'), 'City C', 3);
```

```
INSERT INTO BirthCertificate (Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id)
```

```

VALUES (4, TO_DATE('1993-03-25', 'YYYY-MM-DD'), 'City D', 4);

INSERT INTO BirthCertificate (Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id)
VALUES (5, TO_DATE('2001-09-18', 'YYYY-MM-DD'), 'City E', 5);

```

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

```

VALUES (2, TO_DATE('1998-05-02', 'YYYY-MM-DD'), 'City B', 2);
INSERT INTO BirthCertificate (Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id)
VALUES (3, TO_DATE('2005-11-10', 'YYYY-MM-DD'), 'City C', 3);
INSERT INTO BirthCertificate (Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id)
VALUES (4, TO_DATE('1993-03-25', 'YYYY-MM-DD'), 'City D', 4);
INSERT INTO BirthCertificate (Certificate_Id, Date_Of_Birth, Place_Of_Birth, Person_Id)
VALUES (5, TO_DATE('2001-09-18', 'YYYY-MM-DD'), 'City E', 5);

Select * From BirthCertificate;

```

The Results tab displays the following table:

CERTIFICATE_ID	DATE_OF_BIRTH	PLACE_OF_BIRTH	PERSON_ID
1	15-JAN-00	City A	1
2	02-MAY-98	City B	2
3	10-NOV-05	City C	3
4	25-MAR-93	City D	4
5	18-SEP-01	City E	5

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

For Table 4:

```

INSERT INTO Doctor (Doctor_Id, Name, Email, Contact_No, Specialization, Certificate_Id)
VALUES (1, 'Dr. Adeepa', 'drs@medclinic.com', '555-1234', 'Cardiology', 1);

```

```

INSERT INTO Doctor (Doctor_Id, Name, Email, Contact_No, Specialization, Certificate_Id)
VALUES (2, 'Dr. Asif', 'drd@citymed.com', '555-5678', 'Pediatrics', 2);

```

```

INSERT INTO Doctor (Doctor_Id, Name, Email, Contact_No, Specialization, Certificate_Id)
VALUES (3, 'Dr. Yeasin', 'drm@hosp.com', '555-7890', 'Internal Medicine', 3);

```

```

INSERT INTO Doctor (Doctor_Id, Name, Email, Contact_No, Specialization, Certificate_Id)
VALUES (4, 'Dr. Rahat', 'dre@gmail.com', '555-4567', 'Obstetrics', 4);

```

```
INSERT INTO Doctor (Doctor_Id, Name, Email, Contact_No, Specialization, Certificate_Id)
VALUES (5, 'Dr. Rifat', 'drw@healthcare.com', '555-2222', 'Orthopedics', 5);
```

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

```
INSERT INTO Doctor (Doctor_Id, Name, Email, Contact_No, Specialization, Certificate_Id)
VALUES (2, 'Dr. Asif', 'drd@citymed.com', '555-5678', 'Pediatrics', 2);
INSERT INTO Doctor (Doctor_Id, Name, Email, Contact_No, Specialization, Certificate_Id)
VALUES (3, 'Dr. Yeasin', 'drm@hosp.com', '555-7890', 'Internal Medicine', 3);
INSERT INTO Doctor (Doctor_Id, Name, Email, Contact_No, Specialization, Certificate_Id)
VALUES (4, 'Dr. Rahat', 'dre@gmail.com', '555-4567', 'Obstetrics', 4);
INSERT INTO Doctor (Doctor_Id, Name, Email, Contact_No, Specialization, Certificate_Id)
VALUES (5, 'Dr. Rifat', 'drw@healthcare.com', '555-2222', 'Orthopedics', 5);

Select * From Doctor
```

The results section displays the following table:

DOCTOR_ID	NAME	EMAIL	CONTACT_NO	SPECIALIZATION	CERTIFICATE_ID
1	Dr. Adepta	drs@medclinic.com	555-1234	Cardiology	1
2	Dr. Asif	drd@citymed.com	555-5678	Pediatrics	2
3	Dr. Yeasin	drm@hosp.com	555-7890	Internal Medicine	3
4	Dr. Rahat	dre@gmail.com	555-4567	Obstetrics	4
5	Dr. Rifat	drw@healthcare.com	555-2222	Orthopedics	5

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

For Table 5:

```
INSERT INTO Hospital (Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id)
VALUES (1, 'City General Hospital', 'info@cityhospital.com', 'Nikunja', '555-7890', 1);
```

```
INSERT INTO Hospital (Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id)
VALUES (2, 'Metropolitan Medical Center', 'contact@metroclinic.com', 'Kuril', '555-5678', 2);
```

```
INSERT INTO Hospital (Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id)
VALUES (3, 'Central Hospital', 'central@hosp.com', 'Uttara', '555-1234', 3);
```

```
INSERT INTO Hospital (Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id)
VALUES (4, 'Sunrise Health Clinic', 'sunriseclinic@email.com', 'Badda', '555-4567', 4);
```

```
INSERT INTO Hospital (Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id)
VALUES (5, 'Green Valley Hospital', 'gvh@healthcare.com', 'Bashundhara', '555-2222', 5);
```

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit

```
VALUES (2, 'Metropolitan Medical Center', 'contact@metroclinic.com', 'Kuril', '555-5678', 2);
INSERT INTO Hospital (Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id)
VALUES (3, 'Central Hospital', 'central@hosp.com', 'Uttara', '555-1234', 3);
INSERT INTO Hospital (Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id)
VALUES (4, 'Sunrise Health Clinic', 'sunriseclinic@email.com', 'Badda', '555-4567', 4);
INSERT INTO Hospital (Hospital_Id, Name, Email, Address, Contact_No, Doctor_Id)
VALUES (5, 'Green Valley Hospital', 'gvh@healthcare.com', 'Bashundhara', '555-2222', 5);

Select * From Hospital
```

Results Explain Describe Saved SQL History

HOSPITAL_ID	NAME	EMAIL	ADDRESS	CONTACT_NO	DOCTOR_ID
1	City General Hospital	info@cityhospital.com	Nikunja	555-7890	1
2	Metropolitan Medical Center	contact@metroclinic.com	Kuril	555-5678	2
3	Central Hospital	central@hosp.com	Uttara	555-1234	3
4	Sunrise Health Clinic	sunriseclinic@email.com	Badda	555-4567	4
5	Green Valley Hospital	gvh@healthcare.com	Bashundhara	555-2222	5

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

Query Writing:

SQL:

Single-row function:

1. Retrieve the length of each customer's name.

```
SELECT name, LENGTH(name) AS Length_Of_Name FROM Person;
```

The screenshot shows the Oracle Database Express Edition interface. The SQL command `SELECT name, LENGTH(name) AS Length_Of_Name FROM Person;` is entered in the SQL Commands window. The results are displayed in a table with columns 'NAME' and 'LENGTH_OF_NAME'. The data shows five rows: Adeeptha (7), Yeasin (6), Asif (4), Rifat (5), and Rahat (5). A CSV export link is available at the bottom of the results table.

NAME	LENGTH_OF_NAME
Adeeptha	7
Yeastin	6
Asif	4
Rifat	5
Rahat	5

2. Display the customer names in uppercase.

```
SELECT name, UPPER(name) AS Name_In_Uppercase FROM Person;
```

The screenshot shows the Oracle Database Express Edition interface. The SQL command `SELECT name, UPPER(name) AS Name_In_Uppercase FROM Person;` is entered in the SQL Commands window. The results are displayed in a table with columns 'NAME' and 'NAME_IN_UPPERCASE'. The data shows five rows: Adeeptha (ADEEPTA), Yeasin (YEASIN), Asif (ASIF), Rifat (RIFAT), and Rahat (RAHAT). A CSV export link is available at the bottom of the results table.

NAME	NAME_IN_UPPERCASE
Adeeptha	ADEEPTA
Yeastin	YEASIN
Asif	ASIF
Rifat	RIFAT
Rahat	RAHAT

3. Retrieve the email addresses of doctors who specialize in "Cardiology."

```
SELECT Email FROM Doctor WHERE Specialization = 'Cardiology';
```

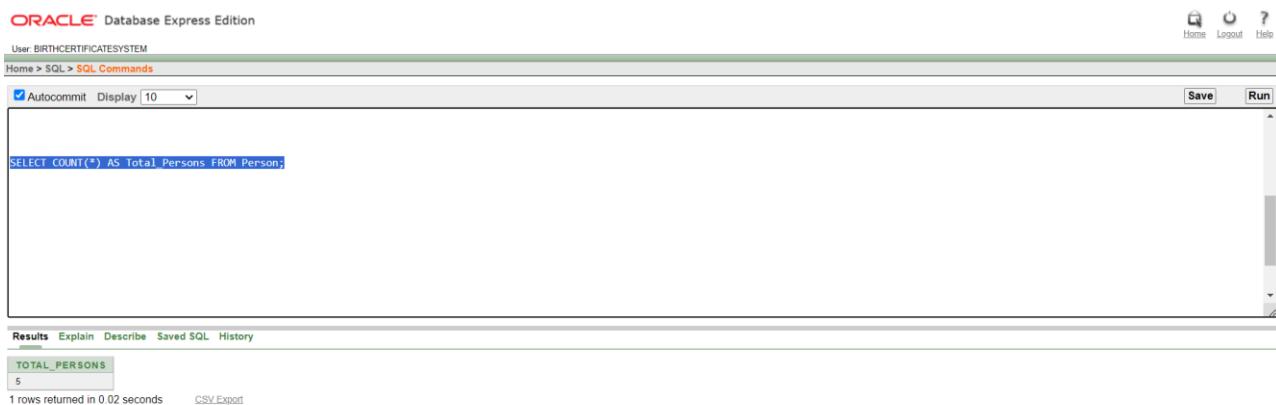


The screenshot shows the Oracle Database Express Edition interface. The SQL command `SELECT Email FROM Doctor WHERE Specialization = 'Cardiology';` is entered in the SQL Commands window. The results show one row: `EMAIL` with value `drs@medclinic.com`. The results table has a header row with columns: Results, Explain, Describe, Saved SQL, History. Below the table, it says `1 rows returned in 0.02 seconds` and provides a `CSV Export` link.

Group Function:

1. Calculate the total number of persons in the "Person" table.

```
SELECT COUNT(*) AS Total_Persons FROM Person;
```



The screenshot shows the Oracle Database Express Edition interface. The SQL command `SELECT COUNT(*) AS Total_Persons FROM Person;` is entered in the SQL Commands window. The results show one row: `TOTAL_PERSONS` with value `5`. The results table has a header row with columns: Results, Explain, Describe, Saved SQL, History. Below the table, it says `1 rows returned in 0.02 seconds` and provides a `CSV Export` link.

2. Calculate the total number of doctors who have a specialization in "Pediatrics" in the "Doctor" table.

```
SELECT COUNT(*) AS Total_Pediatrics_Doctors FROM Doctor WHERE Specialization = 'Pediatrics';
```

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit

```
SELECT COUNT(*) AS Total_Pediatrics_Doctors FROM Doctor WHERE Specialization = 'Pediatrics';
```

Results Explain Describe Saved SQL History

TOTAL_PEDIATRICS_DOCTORS
1

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

3. Find the average number of doctors per hospital in the "Hospital" table.

SELECT AVG(Doctor_Id) AS Average_Doctors_Per_Hospital FROM Hospital;

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit

```
SELECT AVG(Doctor_Id) AS Average_Doctors_Per_Hospital FROM Hospital;
```

Results Explain Describe Saved SQL History

AVERAGE_DOCTORS_PER_HOSPITAL
3

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

Sub Query:

1. Find the names of hospitals where at least one doctor specializes in "Cardiology."

SELECT Name FROM Hospital WHERE Hospital_Id IN (
 SELECT Hospital_Id FROM Doctor
 WHERE Specialization = 'Cardiology');

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit

```
SELECT Name FROM Hospital WHERE Hospital_Id IN (
SELECT Hospital_Id FROM Doctor
WHERE Specialization = 'Cardiology');
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

NAME
City General Hospital
Metropolitan Medical Center
Central Hospital
Sunrise Health Clinic
Green Valley Hospital

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

2. Find the names of doctors who have a specialization matching any of the doctors in Hospital with ID 1.

```
SELECT Name FROM Doctor WHERE Specialization IN (
SELECT Specialization FROM Hospital
WHERE Hospital_Id = 1);
```

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit

```
SELECT Name FROM Doctor WHERE Specialization IN (
SELECT Specialization FROM Hospital
WHERE Hospital_Id = 1);
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

NAME
Dr. Adeeba
Dr. Asif
Dr. Yesmin
Dr. Rahat
Dr. Rifat

5 rows returned in 0.01 seconds [CSV Export](#)

3. Find the names of parents who have more than one child

```
SELECT Name FROM Parent WHERE Parent_Id IN (
  SELECT Parent_Id FROM Person
  GROUP BY Parent_Id
  HAVING COUNT(*) > 1
);
```

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

```
SELECT Name FROM Parent WHERE Parent_Id IN (
  SELECT Parent_Id FROM Person
  GROUP BY Parent_Id
  HAVING COUNT(*) > 1
);
```

The results pane shows a single row:

NAME
Adeepsa

Below the results, it says "1 rows returned in 0.00 seconds".

Joinning

1. Retrieve the names of persons along with the hospital names where they were born, and the contact numbers of those hospitals.

```
SELECT P.Name AS Person_Name, H.Name AS Hospital_Name, H.Contact_No AS Hospital_Contact
FROM Person P
JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id
JOIN Hospital H ON P.Hospital_Id = H.Hospital_Id;
```

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT P.Name AS Person_Name, H.Name AS Hospital_Name, H.Contact_No AS Hospital_Contact
FROM Person P
JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id
JOIN Hospital H ON P.Hospital_Id = H.Hospital_Id;
```

Results Explain Describe Saved SQL History

PERSON_NAME	HOSPITAL_NAME	HOSPITAL_CONTACT
Adeeba	City General Hospital	555-7890
Yasin	Metropolitan Medical Center	555-5678
Asif	Central Hospital	555-1234
Rifat	Sunrise Health Clinic	555-4567
Rahat	Green Valley Hospital	555-2222

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

2. Retrieve the names of doctors along with their specialization and the hospital names where they work.

```
SELECT D.Name AS Doctor_Name, D.Specialization, H.Name AS Hospital_Name
FROM Doctor D
JOIN Hospital H ON D.Doctor_Id = H.Doctor_Id;
```

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT D.Name AS Doctor_Name, D.Specialization, H.Name AS Hospital_Name
FROM Doctor D
JOIN Hospital H ON D.Doctor_Id = H.Doctor_Id;
```

Results Explain Describe Saved SQL History

DOCTOR_NAME	SPECIALIZATION	HOSPITAL_NAME
Dr. Adeeba	Cardiology	City General Hospital
Dr. Asif	Pediatrics	Metropolitan Medical Center
Dr. Yasin	Internal Medicine	Central Hospital
Dr. Rahat	Obstetrics	Sunrise Health Clinic
Dr. Rifat	Orthopedics	Green Valley Hospital

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

3. Retrieve the names of persons along with the certificate IDs and their place of birth.

```
SELECT P.Name AS Person_Name, BC.Certificate_Id, BC.Place_Of_Birth
FROM Person P
JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id;
```

ORACLE® Database Express Edition

User: BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit

```
SELECT P.Name AS Person Name, BC.Certificate_Id, BC.Place_OF_Birth
FROM Person P
JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id;
```

Results Explain Describe Saved SQL History

PERSON_NAME	CERTIFICATE_ID	PLACE_OF_BIRTH
Adeeqa	1	City A
Yeesin	2	City B
Aisif	3	City C
Rifat	4	City D
Rahat	5	City E

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

View

1. Create a view that displays the names of Hospitals along with their Email and Doctor names.

```
CREATE VIEW HospitalInfo AS
SELECT H.Name AS Hospital_Name, H.Email, D.Name AS Doctor_Name
FROM Hospital H
JOIN Doctor D ON H.Doctor_Id = D.Doctor_Id;
```

Select * From HospitalInfo

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit

```
CREATE VIEW HospitalInfo AS
SELECT H.Name AS Hospital_Name, H.Email, D.Name AS Doctor_Name
FROM Hospital H
JOIN Doctor D ON H.Doctor_Id = D.Doctor_Id;
Select * From HospitalInfo;
```

Results Explain Describe Saved SQL History

HOSPITAL_NAME	EMAIL	DOCTOR_NAME
City General Hospital	info@cityhospital.com	Dr. Adepta
Metropolitan Medical Center	contact@metroclinic.com	Dr. Asif
Central Hospital	central@hosp.com	Dr. Yesin
Sunrise Health Clinic	sunriseclinic@email.com	Dr. Rahat
Green Valley Hospital	gvh@healthcare.com	Dr. Rifat

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

2. Create a view that provides information about persons including their gender, hospital names, and place of birth.

```
CREATE VIEW PersonDetails AS
SELECT P.Person_Id, P.Name AS Person_Name, P.Gender, H.Name AS Hospital_Name,
BC.Place_Of_Birth
FROM Person P
LEFT JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id
LEFT JOIN Hospital H ON P.Hospital_Id = H.Hospital_Id;
```

Select * From PersonDetails

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit

```
CREATE VIEW PersonDetails AS
SELECT P.Person_Id, P.Name AS Person_Name, P.Gender, H.Name AS Hospital_Name, BC.Place_Of_Birth
FROM Person P
LEFT JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id
LEFT JOIN Hospital H ON P.Hospital_Id = H.Hospital_Id;
Select * From PersonDetails
```

Results Explain Describe Saved SQL History

PERSON_ID	PERSON_NAME	GENDER	HOSPITAL_NAME	PLACE_OF_BIRTH
6	Shuvo	Male	City General Hospital	-
1	Adepta	Male	City General Hospital	City A
2	Yesin	Male	Metropolitan Medical Center	City B
3	Asif	Male	Central Hospital	City C
4	Rifat	Male	Sunrise Health Clinic	City D
5	Rahat	Male	Green Valley Hospital	City E

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

3. Create a view that combines information from the BirthCertificate table and the Person table to show names of persons along with their date of birth and place of birth.

```
CREATE VIEW PersonBirthInfo AS
SELECT P.Name AS Person_Name, BC.Date_Of_Birth, BC.Place_Of_Birth
FROM Person P
```

JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id;

Select * From PersonBirthInfo

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

```
CREATE VIEW PersonBirthInfo AS
SELECT P.Name AS Person_Name, BC.Date_Of_Birth, BC.Place_Of_Birth
FROM Person P
JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id;
Select * From PersonBirthInfo
```

Below the code, the results are displayed in a grid:

PERSON_NAME	DATE_OF_BIRTH	PLACE_OF_BIRTH
Adepta	15-JAN-00	City A
Yasin	02-MAY-98	City B
Asif	10-NOV-05	City C
Rifat	25-MAR-93	City D
Rahat	18-SEP-01	City E

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

Synonym:

1. Create a synonym named "ParentInfo" that refers to the "Parent" table.

CREATE OR REPLACE SYNONYM ParentInfo FOR Parent;

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

```
CREATE OR REPLACE SYNONYM ParentInfo FOR Parent;
```

Below the code, the results are displayed:

Synonym created.
0.00 seconds

2. Create a synonym named "PersonBirthInformation" that refers to a query combining information from the "Person" and "BirthCertificate" tables to show names of persons along with their date of birth and place of

birth.

```
CREATE OR REPLACE VIEW PersonBirthDetails AS
SELECT P.Name AS Person_Name, BC.Date_Of_Birth, BC.Place_Of_Birth
FROM Person P
JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id;
```

```
CREATE OR REPLACE SYNONYM PersonBirthInformation FOR PersonBirthDetails;
```

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

```
CREATE OR REPLACE VIEW PersonBirthDetails AS
SELECT P.Name AS Person_Name, BC.Date_Of_Birth, BC.Place_Of_Birth
FROM Person P
JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id;
CREATE OR REPLACE SYNONYM PersonBirthInformation FOR PersonBirthDetails;
```

Below the code, the results show:

```
Synonym created.
0.00 seconds
```

3. Create a synonym named "PersonAndCertificate" that refers to a query combining information from the "Person" and "BirthCertificate" tables to show names of persons along with their certificate IDs, if available.

```
CREATE OR REPLACE VIEW PersonCertificateInfo AS
SELECT P.Name AS Person_Name, BC.Certificate_Id
FROM Person P
LEFT JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id;
```

```
CREATE OR REPLACE SYNONYM PersonAndCertificate FOR PersonCertificateInfo;
```

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

```
CREATE OR REPLACE VIEW PersonCertificateInfo AS
SELECT P.Name AS Person_Name, BC.Certificate_Id
FROM Person P
LEFT JOIN BirthCertificate BC ON P.Person_Id = BC.Person_Id;
CREATE OR REPLACE SYNONYM PersonAndCertificate FOR PersonCertificateInfo;
```

Below the code, the results show:

```
Synonym created.
0.00 seconds
```

PL/SQL:

Function:

1. Create a PL/SQL function named "GetParentContact" that takes a parent ID as input and returns the contact number of the parent.

```
CREATE OR REPLACE FUNCTION GetParentContact(parentId NUMBER) RETURN VARCHAR2 AS
    parentContact VARCHAR2(20);
BEGIN
    SELECT Contact_No INTO parentContact
    FROM Parent
    WHERE Parent_Id = parentId;

    RETURN parentContact;
END;
```



The screenshot shows the Oracle Database Express Edition interface. The SQL Commands tab is selected. The code area contains the PL/SQL function definition. The status bar at the bottom indicates 'Function created.' and '0.05 seconds'.

```
ORACLE Database Express Edition
User BIRTHCERTIFICATESYSTEM
Home > SQL > SQL Commands
Autocommit Display 10 Save Run
CREATE OR REPLACE FUNCTION GetParentContact(parentId NUMBER) RETURN VARCHAR2 AS
    parentContact VARCHAR2(20);
BEGIN
    SELECT Contact_No INTO parentContact
    FROM Parent
    WHERE Parent_Id = parentId;
    RETURN parentContact;
END;

Function created.
0.05 seconds
```

2. Create a PL/SQL function named "GetHospitalEmail" that takes a hospital ID as input and returns the email address of the hospital.

```
CREATE OR REPLACE FUNCTION GetHospitalEmail(hospitalId NUMBER) RETURN VARCHAR2
AS
    hospitalEmail VARCHAR2(100);
BEGIN
    SELECT Email INTO hospitalEmail
    FROM Hospital
    WHERE Hospital_Id = hospitalId;

    RETURN hospitalEmail;
END;
```

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

```
CREATE OR REPLACE FUNCTION GetHospitalEmail(hospitalId NUMBER) RETURN VARCHAR2 AS
    hospitalEmail VARCHAR2(100);
BEGIN
    SELECT Email INTO hospitalEmail
    FROM Hospital
    WHERE Hospital_Id = hospitalId;
    RETURN hospitalEmail;
END;
```

Save Run

Results Explain Describe Saved SQL History

Function created.

0.00 seconds

3. Create a PL/SQL function named "GetDoctorCountBySpecialization" that takes a specialization as input and returns the count of doctors with that specialization.

```
CREATE OR REPLACE FUNCTION GetDoctorCountBySpecialization(inputSpecialization VARCHAR2)
RETURN NUMBER AS
doctorCount NUMBER;
BEGIN
    SELECT COUNT(*) INTO doctorCount
    FROM Doctor
    WHERE Specialization = inputSpecialization;

    RETURN doctorCount;
END;
/
```

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

```
CREATE OR REPLACE FUNCTION GetDoctorCountBySpecialization(inputSpecialization VARCHAR2) RETURN NUMBER AS
doctorCount NUMBER;
BEGIN
    SELECT COUNT(*) INTO doctorCount
    FROM Doctor
    WHERE Specialization = inputSpecialization;
    RETURN doctorCount;
END;
```

Save Run

Results Explain Describe Saved SQL History

Function created.

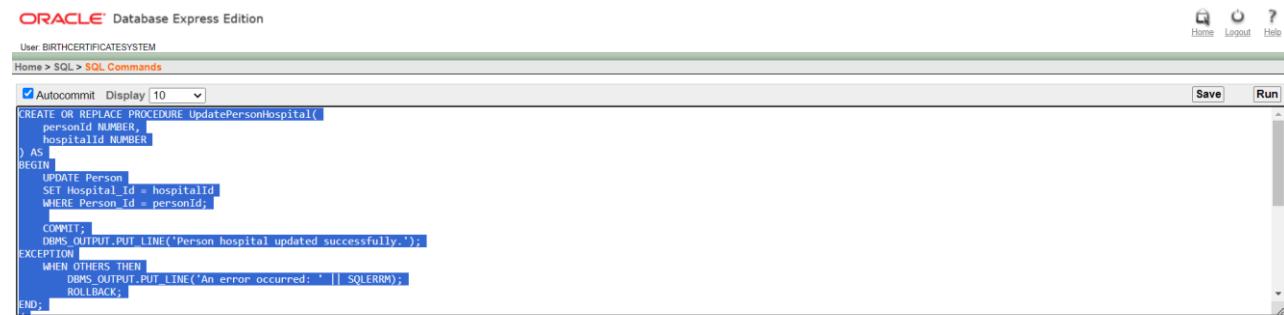
0.00 seconds

Procedure:

1. Create a PL/SQL procedure named "UpdatePersonHospital" that takes a person's ID and a hospital's ID as input, and updates the person's hospital association.

```
CREATE OR REPLACE PROCEDURE UpdatePersonHospital(
    personId NUMBER,
    hospitalId NUMBER
) AS
BEGIN
    UPDATE Person
    SET Hospital_Id = hospitalId
    WHERE Person_Id = personId;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Person hospital updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK;
END;
/
```



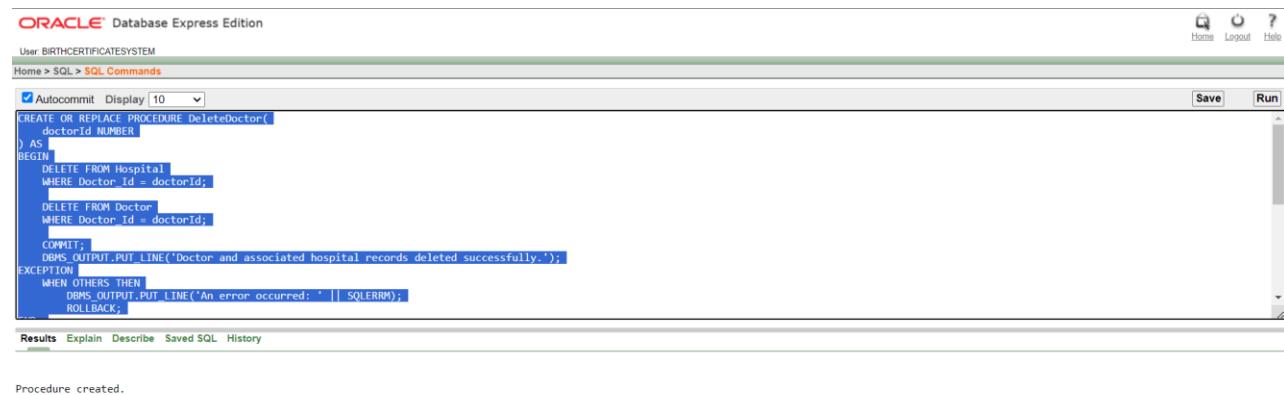
The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window displays the PL/SQL code for the 'UpdatePersonHospital' procedure. The code includes parameters for personId and hospitalId, a BEGIN block with an UPDATE statement, a COMMIT, a DBMS_OUTPUT.PUT_LINE for success, an EXCEPTION block with an error message, and a ROLLBACK. The code is highlighted in blue and black. The interface includes a toolbar with Home, Logout, and Help buttons, and a bottom navigation bar with Results, Explain, Describe, Saved SQL, and History tabs. The status bar at the bottom indicates 'Procedure created.' and '0.00 seconds'.

2. Create a PL/SQL procedure named "DeleteDoctor" that takes a doctor's ID as input, and deletes the doctor's record along with associated hospital records.

```
CREATE OR REPLACE PROCEDURE DeleteDoctor(
    doctorId NUMBER
) AS
BEGIN
    DELETE FROM Hospital
    WHERE Doctor_Id = doctorId;

    DELETE FROM Doctor
    WHERE Doctor_Id = doctorId;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Doctor and associated hospital records deleted successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK;
END;
/
```



The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window contains the PL/SQL code for the DeleteDoctor procedure. The code includes logic to delete records from the Hospital and Doctor tables based on a given doctor ID, a COMMIT statement, and an exception handling block for errors. The interface has a toolbar with icons for Home, Logout, and Help, and a menu bar with AutoCommit, Display, Save, and Run buttons. The results pane at the bottom shows the message "Procedure created." and a execution time of "0.00 seconds".

```
ORACLE Database Express Edition
User BIRTHCERTIFICATESYSTEM
Home > SQL > SQL Commands
CREATE OR REPLACE PROCEDURE DeleteDoctor(
    doctorId NUMBER
) AS
BEGIN
    DELETE FROM Hospital
    WHERE Doctor_Id = doctorId;

    DELETE FROM Doctor
    WHERE Doctor_Id = doctorId;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Doctor and associated hospital records deleted successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK;
END;
/
```

Procedure created.
0.00 seconds

3. Create a PL/SQL procedure named "UpdateDoctorEmail" that takes a doctor's ID and a new email address as input, and updates the doctor's email.

```

CREATE OR REPLACE PROCEDURE UpdateDoctorEmail(
    doctorId NUMBER,
    newEmail VARCHAR2
) AS
BEGIN
    UPDATE Doctor
    SET Email = newEmail
    WHERE Doctor_Id = doctorId;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Doctor email updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK;
END;
/

```

The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window contains the PL/SQL code provided above. The code creates a procedure named 'UpdateDoctorEmail' that takes two parameters: 'doctorId' (NUMBER) and 'newEmail' (VARCHAR2). It begins by updating the 'Doctor' table where 'Doctor_Id' matches 'doctorId'. After the update, it commits the transaction and prints a success message to the DBMS_OUTPUT. It also handles any errors by printing the SQL error message and rolling back the transaction. Finally, it ends the procedure definition. The status bar at the bottom indicates 'Procedure created.' and '0.00 seconds'.

```

ORACLE® Database Express Edition
User BIRTHCERTIFICATESYSTEM
Home > SQL > SQL Commands
CREATE OR REPLACE PROCEDURE UpdateDoctorEmail(
    doctorId NUMBER,
    newEmail VARCHAR2
) AS
BEGIN
    UPDATE Doctor
    SET Email = newEmail
    WHERE Doctor_Id = doctorId;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Doctor email updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK;
END;
/

```

Record:

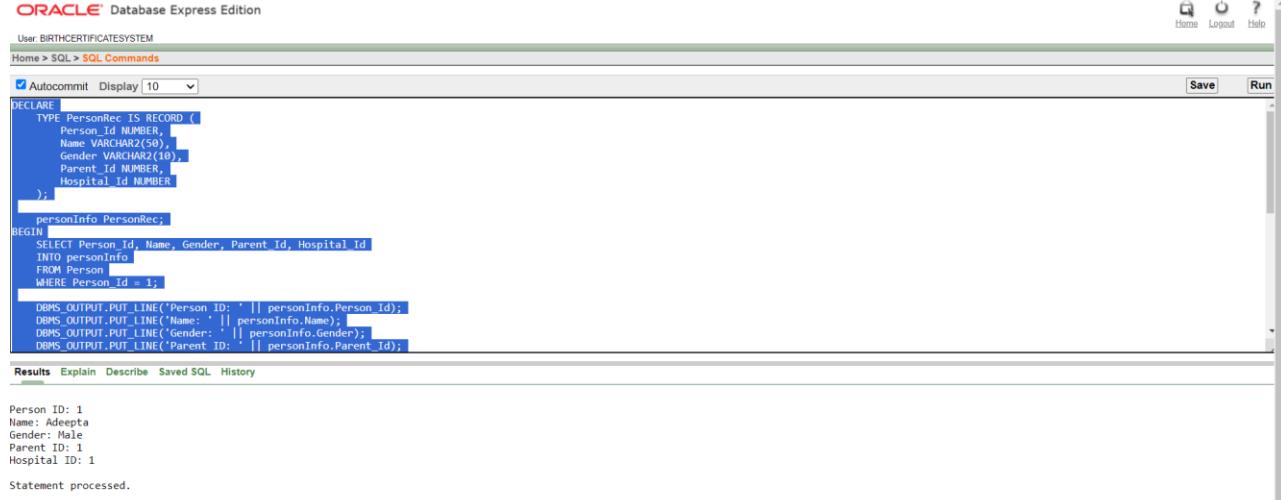
1.Create a PL/SQL record named "PersonInfo" that can hold the attributes of a person (Person_Id, Name, Gender, Parent_Id, Hospital_Id). Use this record to fetch and display information about a specific person.

DECLARE

```
TYPE PersonRec IS RECORD (
    Person_Id NUMBER,
    Name VARCHAR2(50),
    Gender VARCHAR2(10),
    Parent_Id NUMBER,
    Hospital_Id NUMBER
);
```

```
personInfo PersonRec;
BEGIN
    SELECT Person_Id, Name, Gender, Parent_Id, Hospital_Id
    INTO personInfo
    FROM Person
    WHERE Person_Id = 1;
```

```
DBMS_OUTPUT.PUT_LINE('Person ID: ' || personInfo.Person_Id);
DBMS_OUTPUT.PUT_LINE('Name: ' || personInfo.Name);
DBMS_OUTPUT.PUT_LINE('Gender: ' || personInfo.Gender);
DBMS_OUTPUT.PUT_LINE('Parent ID: ' || personInfo.Parent_Id);
DBMS_OUTPUT.PUT_LINE('Hospital ID: ' || personInfo.Hospital_Id);
END;
/
```



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

```
DECLARE
    TYPE PersonRec IS RECORD (
        Person_Id NUMBER,
        Name VARCHAR2(50),
        Gender VARCHAR2(10),
        Parent_Id NUMBER,
        Hospital_Id NUMBER
    );
    personInfo PersonRec;
BEGIN
    SELECT Person_Id, Name, Gender, Parent_Id, Hospital_Id
    INTO personInfo
    FROM Person
    WHERE Person_Id = 1;
    DBMS_OUTPUT.PUT_LINE('Person ID: ' || personInfo.Person_Id);
    DBMS_OUTPUT.PUT_LINE('Name: ' || personInfo.Name);
    DBMS_OUTPUT.PUT_LINE('Gender: ' || personInfo.Gender);
    DBMS_OUTPUT.PUT_LINE('Parent ID: ' || personInfo.Parent_Id);
    DBMS_OUTPUT.PUT_LINE('Hospital ID: ' || personInfo.Hospital_Id);
END;
/
```

The results pane below shows the output of the DBMS_OUTPUT.PUT_LINE statements:

```
Person ID: 1
Name: Adeeps
Gender: Male
Parent ID: 1
Hospital ID: 1
```

At the bottom, it says "Statement processed."

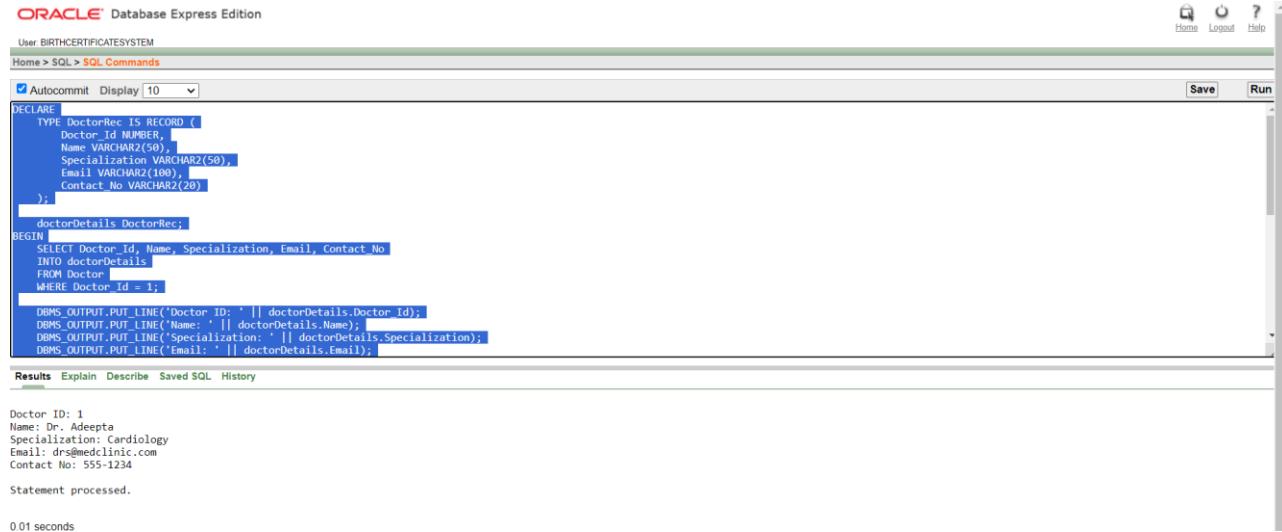
2. Create a PL/SQL record named "DoctorDetails" that can hold the attributes of a doctor (Doctor_Id, Name, Specialization, Email, Contact_No). Use this record to fetch and display information about a specific doctor.

DECLARE

```
TYPE DoctorRec IS RECORD (
    Doctor_Id NUMBER,
    Name VARCHAR2(50),
    Specialization VARCHAR2(50),
    Email VARCHAR2(100),
    Contact_No VARCHAR2(20)
);
```

```
doctorDetails DoctorRec;
BEGIN
    SELECT Doctor_Id, Name, Specialization, Email, Contact_No
    INTO doctorDetails
    FROM Doctor
    WHERE Doctor_Id = 1;
```

```
DBMS_OUTPUT.PUT_LINE('Doctor ID: ' || doctorDetails.Doctor_Id);
DBMS_OUTPUT.PUT_LINE('Name: ' || doctorDetails.Name);
DBMS_OUTPUT.PUT_LINE('Specialization: ' || doctorDetails.Specialization);
DBMS_OUTPUT.PUT_LINE('Email: ' || doctorDetails.Email);
DBMS_OUTPUT.PUT_LINE('Contact No: ' || doctorDetails.Contact_No);
END;
/
```



The screenshot shows the Oracle Database Express Edition interface. The SQL Commands tab is active. The code area contains the provided PL/SQL block. The results pane shows the output of the DBMS_OUTPUT.PUT_LINE statements:

```
Doctor ID: 1
Name: Dr. Adeeqta
Specialization: Cardiology
Email: drs@medclinic.com
Contact No: 555-1234
```

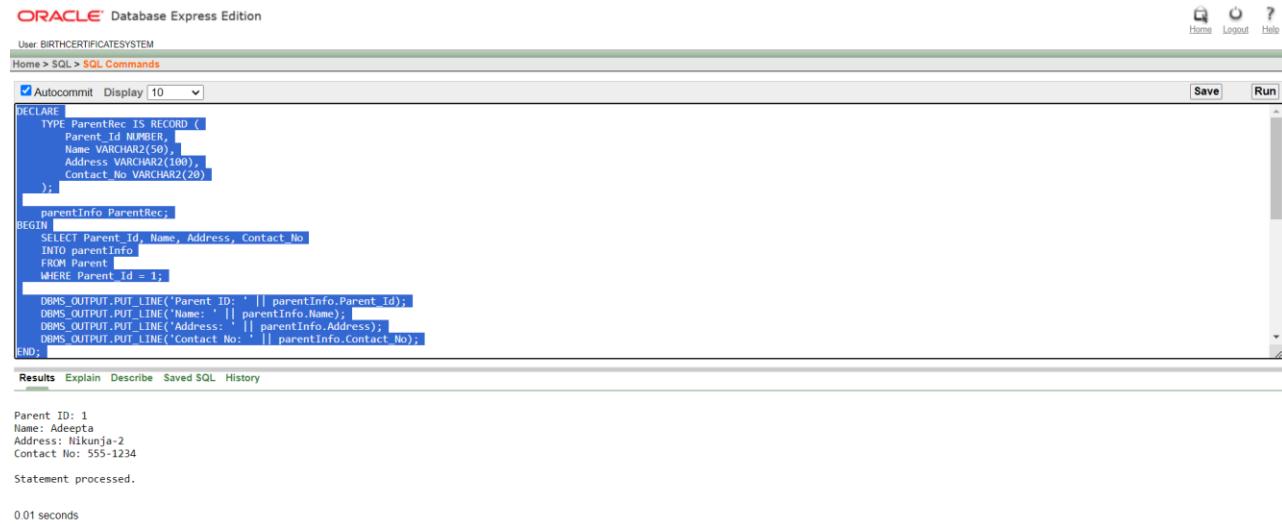
Below the results, it says "Statement processed." and "0.01 seconds".

3. Create a PL/SQL record named "ParentInfo" that can hold the attributes of a parent (Parent_Id, Name, Address, Contact_No). Use this record to fetch and display information about a specific parent.

```
DECLARE
  TYPE ParentRec IS RECORD (
    Parent_Id NUMBER,
    Name VARCHAR2(50),
    Address VARCHAR2(100),
    Contact_No VARCHAR2(20)
  );
```

```
  parentInfo ParentRec;
BEGIN
  SELECT Parent_Id, Name, Address, Contact_No
  INTO parentInfo
  FROM Parent
  WHERE Parent_Id = 1;

  DBMS_OUTPUT.PUT_LINE('Parent ID: ' || parentInfo.Parent_Id);
  DBMS_OUTPUT.PUT_LINE('Name: ' || parentInfo.Name);
  DBMS_OUTPUT.PUT_LINE('Address: ' || parentInfo.Address);
  DBMS_OUTPUT.PUT_LINE('Contact No: ' || parentInfo.Contact_No);
END;
/
```



The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window displays the PL/SQL code. The code defines a record type ParentRec with fields Parent_Id, Name, Address, and Contact_No. It then declares a variable parentInfo of type ParentRec and performs a SELECT statement to fetch a parent record where Parent_Id = 1. Finally, it uses DBMS_OUTPUT.PUT_LINE to print the values of the fetched record. The results pane at the bottom shows the output: Parent ID: 1, Name: Adepta, Address: Nikunja-2, and Contact No: 555-1234. The status bar indicates the statement was processed in 0.01 seconds.

```
ORACLE Database Express Edition
User: BIRTHCERTIFICATESYSTEM
Home > SQL > SQL Commands
AutoCommit: On | Display: 10 | Save | Run
DECLARE
  TYPE ParentRec IS RECORD (
    Parent_Id NUMBER,
    Name VARCHAR2(50),
    Address VARCHAR2(100),
    Contact_No VARCHAR2(20)
  );
  parentInfo ParentRec;
BEGIN
  SELECT Parent_Id, Name, Address, Contact_No
  INTO parentInfo
  FROM Parent
  WHERE Parent_Id = 1;
  DBMS_OUTPUT.PUT_LINE('Parent ID: ' || parentInfo.Parent_Id);
  DBMS_OUTPUT.PUT_LINE('Name: ' || parentInfo.Name);
  DBMS_OUTPUT.PUT_LINE('Address: ' || parentInfo.Address);
  DBMS_OUTPUT.PUT_LINE('Contact No: ' || parentInfo.Contact_No);
END;
/
Results Explain Describe Saved SQL History
Parent ID: 1
Name: Adepta
Address: Nikunja-2
Contact No: 555-1234
Statement processed.
0.01 seconds
```

Cursor:

1.Create a PL/SQL cursor named "MalePatientsCursor" that retrieves the names of male patients from the Person table.

```
DECLARE
  CURSOR MalePatientsCursor IS
    SELECT Name
      FROM Person
     WHERE Gender = 'Male';
```

```
  patientName VARCHAR2(50);
BEGIN
  OPEN MalePatientsCursor;
  LOOP
    FETCH MalePatientsCursor INTO patientName;
    EXIT WHEN MalePatientsCursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Male patient name: ' || patientName);
  END LOOP;
  CLOSE MalePatientsCursor;
END;
/
```

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

```
DECLARE
  CURSOR MalePatientsCursor IS
    SELECT Name
      FROM Person
     WHERE Gender = 'Male';
  patientName VARCHAR2(50);
BEGIN
  OPEN MalePatientsCursor;
  LOOP
    FETCH MalePatientsCursor INTO patientName;
    EXIT WHEN MalePatientsCursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Male patient name: ' || patientName);
  END LOOP;
  CLOSE MalePatientsCursor;
END;
/
```

Below the code, the results pane shows the output of the DBMS_OUTPUT.PUT_LINE statements:

```
Male patient name: Adeeba
Male patient name: Yesmin
Male patient name: Asif
Male patient name: Rifat
Male patient name: Rahat
Male patient name: Shuvo
```

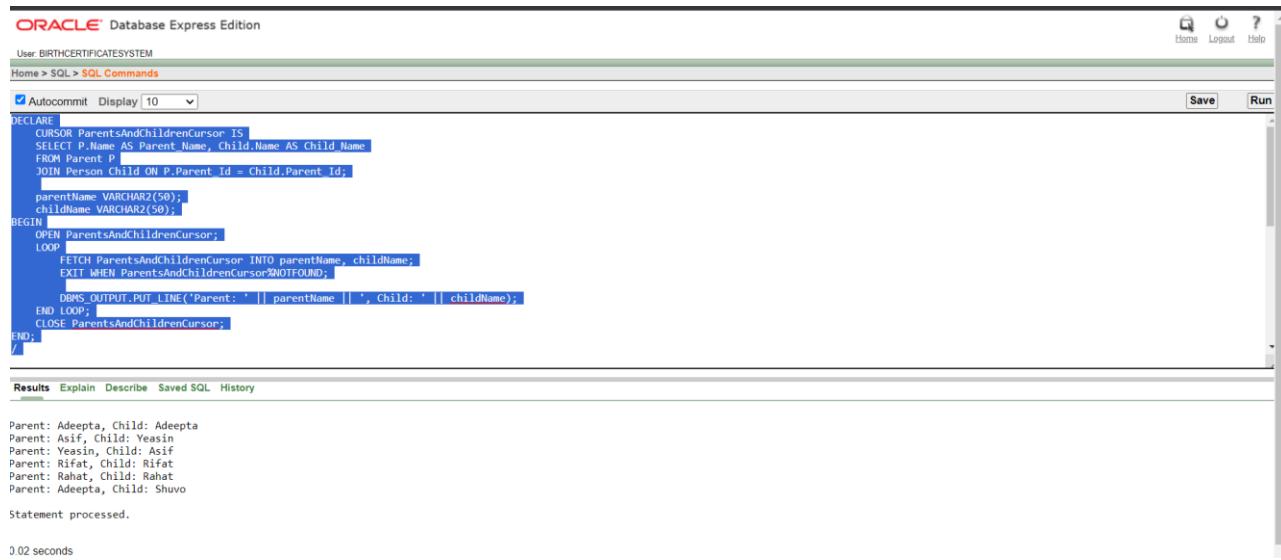
At the bottom, it says "Statement processed."

2. Create a PL/SQL cursor named "ParentsAndChildrenCursor" that retrieves the names of parents and their corresponding children's names from the Parent and Person tables.

```
DECLARE
  CURSOR ParentsAndChildrenCursor IS
    SELECT P.Name AS Parent_Name, Child.Name AS Child_Name
    FROM Parent P
    JOIN Person Child ON P.Parent_Id = Child.Parent_Id;

  parentName VARCHAR2(50);
  childName VARCHAR2(50);
BEGIN
  OPEN ParentsAndChildrenCursor;
  LOOP
    FETCH ParentsAndChildrenCursor INTO parentName, childName;
    EXIT WHEN ParentsAndChildrenCursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Parent: ' || parentName || ', Child: ' || childName);
  END LOOP;
  CLOSE ParentsAndChildrenCursor;
END;
/
```



The screenshot shows the Oracle Database Express Edition interface. The title bar reads "ORACLE Database Express Edition". The menu bar includes "File", "Edit", "View", "Tools", "Help", "Home", "Logout", and "Help". The main window is titled "Home > SQL > SQL Commands". The SQL editor contains the PL/SQL code provided above. The code is highlighted in blue. The "Autocommit" checkbox is checked, and the "Display" dropdown is set to 10. The "Run" button is visible at the top right. The results pane below shows the output of the DBMS_OUTPUT.PUT_LINE statements:

```
Parent: Adeeba, Child: Adeeba
Parent: Asif, Child: Yeasin
Parent: Asif, Child: Rifat
Parent: Rifat, Child: Rifat
Parent: Rahat, Child: Rahat
Parent: Adeeba, Child: Shuvo
```

At the bottom, it says "Statement processed." and "0.02 seconds".

3. Create a PL/SQL cursor named "ParentChildrenCountCursor" that retrieves the count of children for each parent from the Parent and Person tables. Display the parent's name along with the count of their children.

```
DECLARE
  CURSOR ParentChildrenCountCursor IS
    SELECT P.Name AS Parent_Name, COUNT(C.Person_Id) AS Children_Count
    FROM Parent P
    LEFT JOIN Person C ON P.Parent_Id = C.Parent_Id
    GROUP BY P.Name;

  parentName VARCHAR2(50);
  childrenCount NUMBER;

BEGIN
  OPEN ParentChildrenCountCursor;
  LOOP
    FETCH ParentChildrenCountCursor INTO parentName, childrenCount;
    EXIT WHEN ParentChildrenCountCursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Parent: ' || parentName || ', Children Count: ' || childrenCount);
  END LOOP;
  CLOSE ParentChildrenCountCursor;
END;
/
```



The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window displays the PL/SQL code for creating a cursor and printing parent names with their child counts. The results pane shows the output of the DBMS_OUTPUT.PUT_LINE statements for five parents: Adeepa, Rifat, Rahat, Yeasin, and Asif, each with a child count of 1.

```
ORACLE Database Express Edition
User BIRTHCERTIFICATESYSTEM
Home > SQL > SQL Commands
DECLARE
  CURSOR ParentChildrenCountCursor IS
    SELECT P.Name AS Parent_Name, COUNT(C.Person_Id) AS Children_Count
    FROM Parent P
    LEFT JOIN Person C ON P.Parent_Id = C.Parent_Id
    GROUP BY P.Name;

  parentName VARCHAR2(50);
  childrenCount NUMBER;

BEGIN
  OPEN ParentChildrenCountCursor;
  LOOP
    FETCH ParentChildrenCountCursor INTO parentName, childrenCount;
    EXIT WHEN ParentChildrenCountCursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Parent: ' || parentName || ', Children Count: ' || childrenCount);
  END LOOP;
  CLOSE ParentChildrenCountCursor;
END;
/
Results Explain Describe Saved SQL History
Parent: Adeepa, Children Count: 2
Parent: Rifat, Children Count: 1
Parent: Rahat, Children Count: 1
Parent: Yeasin, Children Count: 1
Parent: Asif, Children Count: 1
Statement processed.
0:00 commands
```

Trigger:

1. Create a PL/SQL trigger named "DeleteParentCascadeTrigger" that automatically deletes the associated children records in the Person table when a parent record is deleted from the Parent table.

```
CREATE OR REPLACE TRIGGER DeleteParentCascadeTrigger
AFTER DELETE ON Parent
FOR EACH ROW
BEGIN
    DELETE FROM Person
    WHERE Parent_Id = :old.Parent_Id;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Associated children records deleted due to parent deletion.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK;
END;
/
```

The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The menu bar includes "Home", "Logout", and "Help". The main window shows the SQL Commands tab. The SQL editor contains the PL/SQL code for the trigger. The code is as follows:

```
CREATE OR REPLACE TRIGGER DeleteParentCascadeTrigger
AFTER DELETE ON Parent
FOR EACH ROW
BEGIN
    DELETE FROM Person
    WHERE Parent_Id = :old.Parent_Id;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Associated children records deleted due to parent deletion.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK;
END;
/
```

Below the editor, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". The status bar at the bottom shows "Trigger created.", "0.03 seconds", and page numbers "Page | 53".

2. Create a trigger named ensure_unique_doctor_email that prevents the insertion of a new doctor into the Doctor table if their email already exists in the table.

```
CREATE OR REPLACE TRIGGER ensure_unique_doctor_email
BEFORE INSERT ON Doctor
FOR EACH ROW
DECLARE
    email_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO email_count
    FROM Doctor
    WHERE Email = :new.Email;

    IF email_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'A doctor with the same email already exists.');
    END IF;
END;
/
```



The screenshot shows the Oracle Database Express Edition interface. The title bar reads "ORACLE Database Express Edition". The menu bar includes "Home", "Logout", and "Help". The main window is titled "SQL Commands". The SQL editor contains the trigger creation script provided above. The "Run" button is visible at the top right of the editor. Below the editor, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History".

Trigger created.

0.03 seconds

Package:

1.Create a PL/SQL package named Certificate_Pkg with a procedure named Update_Certificate_Info that takes a Certificate_Id as input and updates the place of birth and date of birth in the BirthCertificate table for the corresponding person.

```
CREATE OR REPLACE PACKAGE Certificate_Pkg AS
    PROCEDURE Update_Certificate_Info(p_Certificate_Id IN BirthCertificate.Certificate_Id%TYPE);
END Certificate_Pkg;
/

CREATE OR REPLACE PACKAGE BODY Certificate_Pkg AS
    PROCEDURE Update_Certificate_Info(p_Certificate_Id IN BirthCertificate.Certificate_Id%TYPE) IS
        v_Person_Id BirthCertificate.Person_Id%TYPE;
    BEGIN
        SELECT Person_Id INTO v_Person_Id
        FROM BirthCertificate
        WHERE Certificate_Id = p_Certificate_Id;

        UPDATE BirthCertificate
        SET Date_Of_Birth = SYSDATE, Place_of_Birth = 'New Place'
        WHERE Person_Id = v_Person_Id;

        COMMIT;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            NULL;
    END Update_Certificate_Info;
END Certificate_Pkg;
/
```

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

Autocommit

```

CREATE OR REPLACE PACKAGE BODY Certificate_Pkg AS
  PROCEDURE Update_Certificate_Info(p_Certificate_Id IN BirthCertificate.Certificate_Id%TYPE) IS
    v_Person_Id BirthCertificate.Person_Id%TYPE;
  BEGIN
    SELECT Person_Id INTO v_Person_Id
    FROM BirthCertificate
    WHERE Certificate_Id = p_Certificate_Id;

    UPDATE BirthCertificate
    SET DateOfBirth = SYSDATE, PlaceOfBirth = 'New Place'
    WHERE Person_Id = v_Person_Id;

    COMMIT;
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        NULL;
    END Update_Certificate_Info;
  END Certificate_Pkg;
/

```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

Package Body created.

0.03 seconds

2. Create a PL/SQL package named Hospital_Pkg with a function named Get_Doctor_Count that takes a Hospital_Id as input and returns the number of doctors working at that hospital.

```

CREATE OR REPLACE PACKAGE Hospital_Pkg AS
  FUNCTION Get_Doctor_Count(p_Hospital_Id IN Hospital.Hospital_Id%TYPE) RETURN NUMBER;
END Hospital_Pkg;
/

```

```

CREATE OR REPLACE PACKAGE BODY Hospital_Pkg AS
  FUNCTION Get_Doctor_Count(p_Hospital_Id IN Hospital.Hospital_Id%TYPE) RETURN NUMBER IS
    v_Doctor_Count NUMBER;
  BEGIN
    SELECT COUNT(*) INTO v_Doctor_Count
    FROM Doctor
    WHERE Doctor_Id = p_Hospital_Id; -- Assuming Doctor_Id is the foreign key in the Hospital table

    RETURN v_Doctor_Count;
  END Get_Doctor_Count;
END Hospital_Pkg;
/

```

ORACLE Database Express Edition

User BIRTHCERTIFICATESYSTEM

Home > SQL > SQL Commands

```

 Autocommit   
CREATE OR REPLACE PACKAGE Hospital_Pkg AS
    FUNCTION Get_Doctor_Count(p_Hospital_Id IN Hospital.Hospital_Id%TYPE) RETURN NUMBER;
END Hospital_Pkg;
/

CREATE OR REPLACE PACKAGE BODY Hospital_Pkg AS
    FUNCTION Get_Doctor_Count(p_Hospital_Id IN Hospital.Hospital_Id%TYPE) RETURN NUMBER IS
        v_Doctor_Count NUMBER;
    BEGIN
        SELECT COUNT(*) INTO v_Doctor_Count
        FROM Doctor
        WHERE Doctor_Id = p_Hospital_Id; -- Assuming Doctor_Id is the foreign key in the Hospital table
        RETURN v_Doctor_Count;
    END Get_Doctor_Count;
END Hospital_Pkg;
/

```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

Package Body created.

0.08 seconds

3. Create a PL/SQL package named Person_Pkg with a procedure named Get_Person_Info that takes a Person_Id as input and returns the name, gender, parent name, and hospital name of the person. If the person is a doctor, also return their specialization.

```

CREATE OR REPLACE PACKAGE Person_Pkg AS
    PROCEDURE Get_Person_Info(
        p_Person_Id IN Person.Person_Id%TYPE,
        p_Name OUT Person.Name%TYPE,
        p_Gender OUT Person.Gender%TYPE,
        p_Parent_Name OUT Parent.Name%TYPE,
        p_Hospital_Name OUT Hospital.Name%TYPE,
        p_Specialization OUT Doctor.Specialization%TYPE
    );
END Person_Pkg;
/

```

```
CREATE OR REPLACE PACKAGE BODY Person_Pkg AS
```

```

PROCEDURE Get_Person_Info(
    p_Person_Id IN Person.Person_Id%TYPE,
    p_Name OUT Person.Name%TYPE,
    p_Gender OUT Person.Gender%TYPE,
    p_Parent_Name OUT Parent.Name%TYPE,
    p_Hospital_Name OUT Hospital.Name%TYPE,
    p_Specialization OUT Doctor.Specialization%TYPE
) IS
BEGIN
    SELECT Pe.Name, Pe.Gender, Pa.Name, Ho.Name, Do.Specialization
    INTO p_Name, p_Gender, p_Parent_Name, p_Hospital_Name, p_Specialization
    FROM Person Pe
    LEFT JOIN Parent Pa ON Pe.Parent_Id = Pa.Parent_Id
    LEFT JOIN Hospital Ho ON Pe.Hospital_Id = Ho.Hospital_Id
    LEFT JOIN Doctor Do ON Pe.Person_Id = Do.Doctor_Id
    WHERE Pe.Person_Id = p_Person_Id;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            p_Name := NULL;
            p_Gender := NULL;
            p_Parent_Name := NULL;
            p_Hospital_Name := NULL;
            p_Specialization := NULL;
END Get_Person_Info;
END Person_Pkg;
/

```

The screenshot shows the Oracle Database Express Edition interface. The title bar says "ORACLE Database Express Edition". The top menu has items like Home, Logout, Help, and a toolbar with icons for Home, Logout, and Help. The main window shows the SQL Commands page. The SQL editor contains the code for the Get_Person_Info procedure. The code is as follows:

```

CREATE OR REPLACE PACKAGE BODY Person_Pkg AS
    PROCEDURE Get_Person_Info(
        p_Person_Id IN Person.Person_Id%TYPE,
        p_Name OUT Person.Name%TYPE,
        p_Gender OUT Person.Gender%TYPE,
        p_Parent_Name OUT Parent.Name%TYPE,
        p_Hospital_Name OUT Hospital.Name%TYPE,
        p_Specialization OUT Doctor.Specialization%TYPE
    ) IS
    BEGIN
        SELECT Pe.Name, Pe.Gender, Pa.Name, Ho.Name, Do.Specialization
        INTO p_Name, p_Gender, p_Parent_Name, p_Hospital_Name, p_Specialization
        FROM Person Pe
        LEFT JOIN Parent Pa ON Pe.Parent_Id = Pa.Parent_Id
        LEFT JOIN Hospital Ho ON Pe.Hospital_Id = Ho.Hospital_Id
        LEFT JOIN Doctor Do ON Pe.Person_Id = Do.Doctor_Id
        WHERE Pe.Person_Id = p_Person_Id;

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                p_Name := NULL;
                p_Gender := NULL;
                p_Parent_Name := NULL;
                p_Hospital_Name := NULL;
                p_Specialization := NULL;
    END Get_Person_Info;
END Person_Pkg;
/

```

The code is highlighted in blue, indicating it is valid SQL. The interface includes a toolbar with "Autocommit" checked, a "Display" dropdown set to 10, and buttons for "Save" and "Run". Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History.

Package Body created.

0.02 seconds

Relation Algebra:

1. Find out his Person_Id whose User_name is Adepta
 $\Pi_{person_id}(\sigma_{User_name = "Adepta"} (Person))$
2. Find out the Person whose Person_address is Badda
 $\Pi_{Person_name}(\sigma_{Person_address = "Mirpur"} (Person))$
3. Find out the Person address whose Contact_number is "458856".
 $\Pi_{Person_address}(\sigma_{Contact_number = "458856"} (Person))$
4. Find out the Person name whose Person_address is Mirpur.
 $\Pi_{Person_name}(\sigma_{Person_address = "Mirpur"} (Person))$
5. Find out the Birth Certificate which name is Adepta.
 $\Pi_{Certificate}(\sigma_{Person_name = "Adepta"} (Birth\ Certificate)).$

Conclusion:

Findings:

- a. The "Person" table keeps track of details about specific people, such as their name, date of birth, and gender.
- b. The "Parent" table contains information on parents, such as their names, addresses, and phone numbers.
- c. Information regarding birth certificates, including their unique identity, associated person ID, date and place of birth, and parents' names, is kept in the "Birth Certificate" table.

- d. The "Doctor" table contains data on the doctors or healthcare providers who assisted in the delivery of the child, including their name, specialization, and contact details.
- e. "Hospital" table keeps details on the hospital or healthcare facility where the birth occurred, such as its name, address, and phone number.

Potential Future Work:

1. Improve Data Integrity: Add more constraints, including NOT NULL constraints or unique constraints, to protect the tables' data integrity.
2. Strengthen Relationship Management: Examine and improve table relationships while making sure that appropriate primary key and foreign key constraints are in place to preserve data consistency.
3. Increase Functionality: Take into account adding extra tables or features to offer broader functionality, such as keeping track of appointments, logging medical history, or handling financial data.
4. Apply data validation rules and tests to ensure that the provided data complies with stated requirements, such as accurate dates or acceptable email formats.
5. Improve User Interface: Make the user interface easier to use and intuitive so that data entering, searching, and reporting are all made possible.
6. Put Security Measures in Place: Boost Data Security with Appropriate Access Controls, Encryption, and Data Backup Techniques.
7. Performance Optimization: By making the proper indexes and optimizing SQL statements, analyses and improve query performance.

These suggested changes are meant to improve the project's usability, data integrity, functionality, and security. Consider variables including feasibility, time limits, and the particular requirements of the project when deciding which topics to priorities for the Final Term.

