**Child Adoption Database System**

**Rupinder Kaur**

**Mudiha Wazirali**

**Abstract**

The database project, ‘Child Adoption Database System’ starts with the registration of users (parent/ fund donor/ employee) and then navigates these users to their respective forms to perform operations such as entering personal details, searching, viewing, and then making decisions. For instance, the fund donor may choose from multiple available adoption centers to donate money. A prospective parent may select a child from any adoption center and file an application to adopt that child. After application review and approval by admin, parents may then view with security parameters a child’s health record. Likewise, a center can view prospective parents’ background check details to decide whether they are eligible to adopt or not.

There are three major constraints in this project:

* Admin must clear background check and health check on prospective parents before approving their application to adopt a child.
* Admin must clear background check on prospective employees before they are hired to

work.

* If a fund donor wants to make donation towards a selected center, then that center should have less than threshold funds to receive money.
* Donation money will be accepted only after center Admin clears background check for fund donor.

**Mission Statement**

The purpose of “Child Adoption Database System” is to manage a centralized system for child adoption that will facilitate future parents to adopt a child. It will also guide fund donors in making donations to the adoption centers. This system will help centers to keep up-to-date records of their employees, children, fund donors and the list of interested future parents.

**Mission Objectives**

To maintain (create, update, and delete) data on Registration.

To maintain (create, update, and delete) data on Parents.

To maintain (create, update, and delete) data on FundDonors.

To maintain (create, update, and delete) data on Employees.

To maintain (create, update, and delete) data on Children.

To maintain (create, update, and delete) data on Centers

To maintain (create, update, and delete) data on Health.

To maintain (create, update, and delete) data on Background Check.

To perform searches on Registration.

To perform searches on Parents.

To perform searches on FundDonors.

To perform searches on Employees.

To perform searches on Children.

To perform searches on Centers.

To perform searches on Health.

To perform searches on Background check.

To track the status of Children before they are adopted.

To track the status of Children after they are adopted.

To track the status of Parents wishing to adopt a child.

To track the status of Parents after they adopt a child.

To track the status of Fund Donors wishing to make donations.

To track the status of Fund Donors after they make donations.

To report on Registration.

To report on Parents.

To report on FundDonors.

To report on Employees.

To report on Children.

To report on Centers.

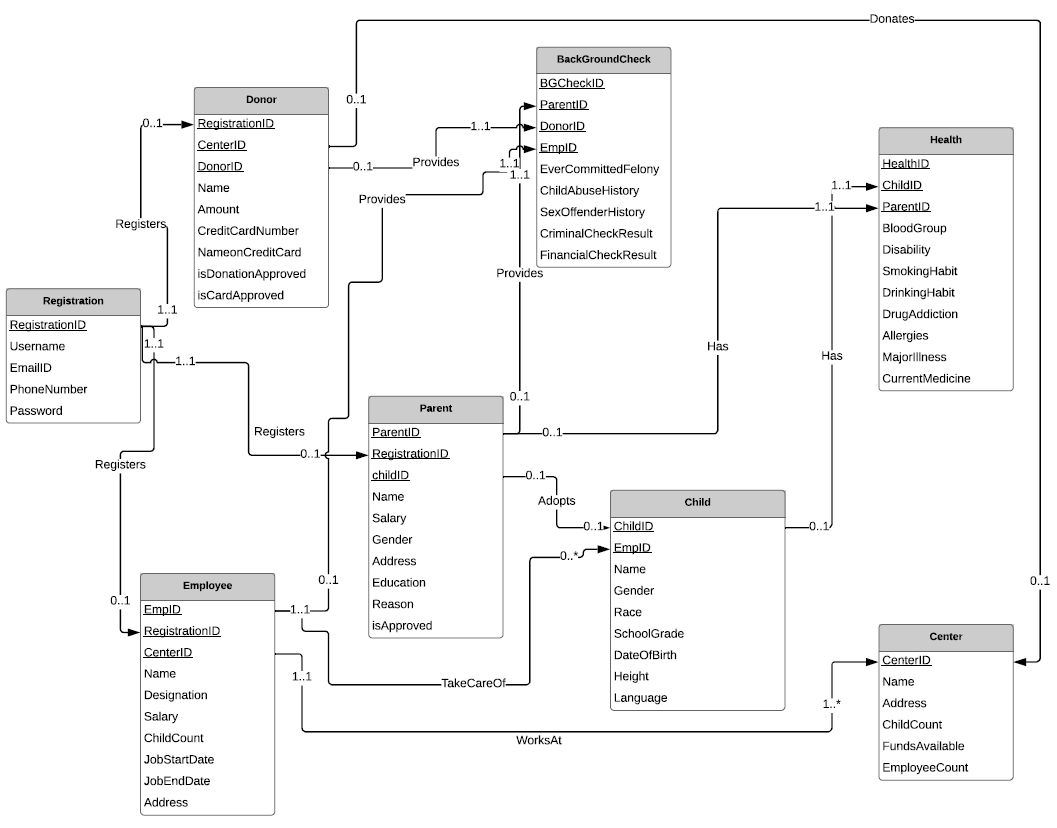
To report on Health.

To report on Background check.

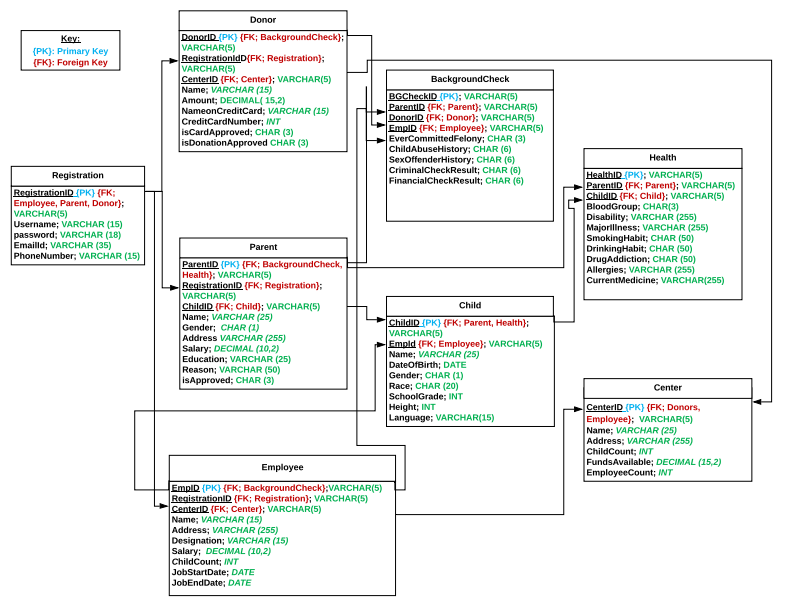
**Major User Views**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data | Access Type | Parent | Donor | ADMIN | Employee |
| Registration All  Registration | Maintain |  |  | X |  |
| Query |  |  | X |  |
| Report |  |  | X |  |
| Maintain | X | X | X | X |
| Query |  |  | X |  |
| Report | X | X | X | X |
| Parents All  Parents | Maintain |  |  | X |  |
| Query |  |  | X |  |
| Report |  |  | X |  |
| Maintain | X |  | X |  |
| Query |  |  | X |  |
| Report | X |  | X |  |
| FundDonors All  FundDonors | Maintain |  |  | X |  |
| Query |  |  | X |  |
| Report |  |  | X |  |
| Maintain |  | X | X |  |
| Query |  |  | X |  |
| Report |  | X | X |  |
| Employees All  Employees | Maintain |  |  | X |  |
| Query |  |  | X |  |
| Report |  |  | X |  |
| Maintain |  |  | X | X |
| Query |  |  | X |  |
| Report |  |  | X | X |
| Children All  Children | Maintain |  |  | X |  |
| Query | X |  | X | X |
| Report | X |  | X | X |
| Maintain |  |  | X |  |
| Query | X |  | X | X |
| Report | X |  | X | X |
| Centers All  Centers | Maintain |  |  | X |  |
| Query | X | X | X |  |
| Report | X | X | X |  |
| Maintain |  |  | X |  |
| Query | X | X | X | X |
| Report | X | X | X | X |
| Health All  Health | Maintain |  |  | X |  |
| Query |  |  | X |  |
| Report |  |  | X |  |
| Maintain |  |  | X |  |
| Query | X |  | X | X |
| Report | X |  | X | X |
| Back Ground Check All  Back Ground Check | Maintain |  |  | X |  |
| Query |  |  | X |  |
| Report |  |  | X |  |
| Maintain | X | X | X | X |
| Query | X | X | X | X |
| Report | X | X | X | X |

**ER Diagram**



**Relational Model**

****

**Attribute Dependencies/BCNF Justification**

**Registration Entity Type**:

**Prime attributes:** RegistrationID, Username, EmailID, PhoneNumber, Password

**Non-prime attributes:** Role

**Functional Dependencies:**

Primary key (RegistrationID) 🡪 Username, Role, Password, EmailId, PhoneNumber

Username 🡪 RegistrationID, EmailID, PhoneNumber, Password, Role

EmailID 🡪 RegistrationID, Username, PhoneNumber, Password, Role

PhoneNumber 🡪 RegistrationID, Username, EmailID, Password, Role

Password 🡪 RegistrationID, Username, EmailID, PhoneNumber, Role

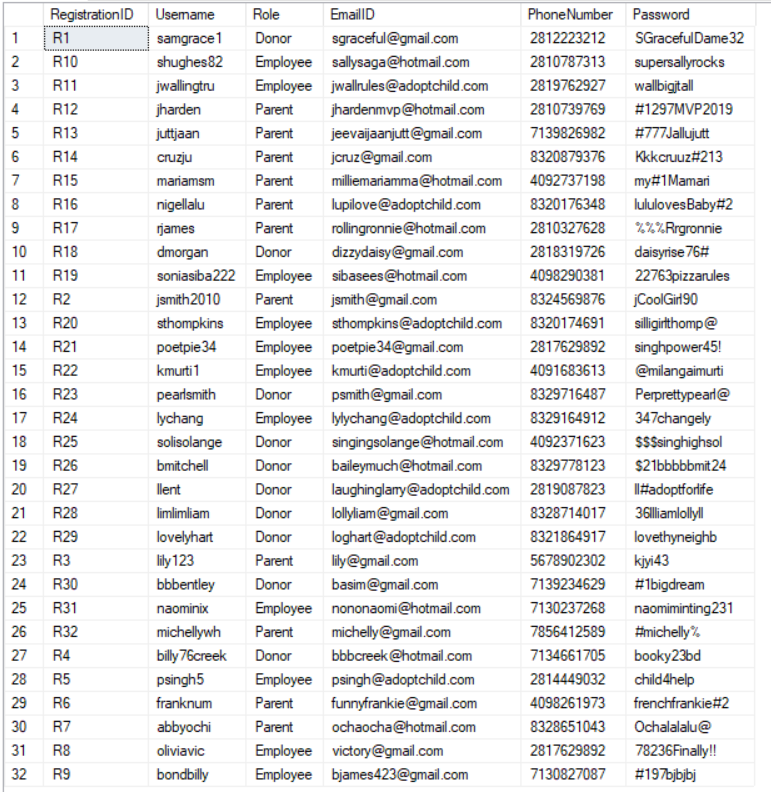
Registration is in 1NF because every cell (intersection of a column and a row) contains only one single value.

It is in 2NF because no partial dependencies exist; all non-prime attributes are fully functionally dependent upon RegistrationID, the primary key.

It is in 3NF because no non-prime attribute depends on other non-prime attributes, all non-prime attribute dependent upon RegistrationID, the primary key.

It is in BCNF because RegistraionID, the primary key, is not dependent on any of the other non-prime attributes.

SELECT \* FROM Registration;



**Donor Entity Type:**

**Prime attributes:** DonorID ,RegistrationID, CreditCardNumber, NameonCreditCard

**Non-prime attributes:** CenterID, Name, Amount, isCardApproved, isDonationApproved

**Functional dependencies:**

Primary key (DonorID) 🡪RegistrationID, CenterID, Name, Amount, NameonCreditCard, CreditCardNumber, isCardApproved, isDonationApproved

RegistrationID 🡪 DonorID ,CenterID, Name, Amount, NameonCreditCard, CreditCardNumber, isCardApproved, isDonationApproved

CreditCardNumber 🡪 DonorID , RegistrationID, CenterID, Name, Amount, NameonCreditCard, isCardApproved, isDonationApproved

NameonCreditCard 🡪 DonorID ,RegistrationID, CenterID, Name, Amount, CreditCardNumber, isCardApproved, isDonationApproved

Donor is in 1NF because every cell (intersection of a column and a row) contains only one single value.

It is in 2NF because no partial dependencies exist since RegistrationID and CreditCardNumber and NameonCreditCard are candidate keys; all non-prime attributes are fully functionally dependent upon RegistrationID, the primary key.

It is in 3NF because no non-prime attribute depends on other non-prime attributes, all non-prime attributes are dependent upon RegistrationID, the primary key.

It is in BCNF because DonorID, the primary key, is not dependent on any of the other non-prime attributes.

SELECT \* FROM Donor;



**Parent Entity Type:**

**Prime attributes:** ParentID , RegistrationID, Name

**Non-prime attributes:** ChildID, Gender, Address, Salary, Education, Reason, isApproved

**Functional dependencies:**

Primary key (ParentID) 🡪 RegistrationID, ChildID, Name, Gender, Address, Salary, Education, Reason, isApproved

RegistrationID 🡪 ParentID , ChildID, Name, Gender, Address, Salary, Education, Reason, isApproved

Name 🡪 ParentID , RegistrationID, ChildID, Gender, Address, Salary, Education, Reason, isApproved

Parent is in 1NF because every cell (intersection of a column and a row) contains only one single value.

It is in 2NF because no partial dependencies exist and all non-prime attributes are fully functionally dependent upon ParentID, the primary key.

It is in 3NF because no non-prime attribute depends on other non-prime attributes, all non-prime attributes are dependent upon ParentID, the primary key.

It is in BCNF because ParentID, the primary key, is not dependent on any of the other non-prime attributes.

SELECT \* FROM Parent;



**Employee Entity Type:**

Primary key (EmpID) 🡪 RegistrationID, CenterID, Name, Address, Designation, Salary, ChildCount, JobStartDate, JobEndDate

**Prime attributes:** EmpID , RegistrationID

**Non-prime attributes:** CenterID, Name, Address, Designation, Salary, ChildCount, JobStartDate, JobEndDate

**Functional dependencies:**

Primary key (EmpID) 🡪 RegistrationID, CenterID, Name, Address, Designation, Salary, ChildCount, JobStartDate, JobEndDate

RegistrationID 🡪 EmpID, CenterID, Name, Address, Designation, Salary, ChildCount, JobStartDate, JobEndDate

Employee is in 1NF because every cell (intersection of a column and a row) contains only one single value.

It is in 2NF because no partial dependencies exist since RegistrationID is candidate key; all non-prime attributes are fully functionally dependent upon EmpID, the primary key.

It is in 3NF because no non-prime attribute depends on other non-prime attributes, all non-prime attributes are dependent upon EmpID, the primary key.

It is in BCNF because EmpID, the primary key, is not dependent on any of the other non-prime attributes.

SELECT \* FROM Employee;



**BackgroundCheck Entity Type**:

**Prime attributes:** BGCheckID

**Non-prime attributes:** ParentID, DonorID, EmpID, EverComittedFelony, ChildAbuseHistory, SexOffenderHistory, CriminalCheckResult, FinancialCheckResult

**Functional dependencies:**

Primary key (BGCheckID) 🡪 ParentID, DonorID, EmpID, EverComittedFelony, ChildAbuseHistory, SexOffenderHistory, CriminalCheckResult, FinancialCheckResult

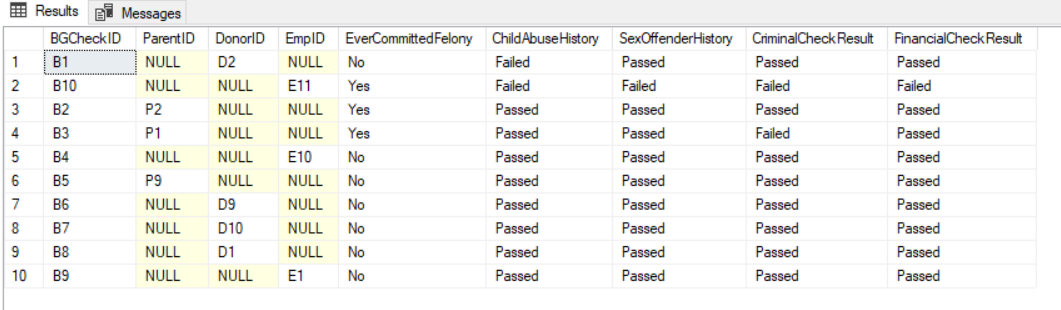
BackgroundCheck is in 1NF because every cell (intersection of a column and a row) contains only one single value.

It is in 2NF because no partial dependencies exist, and all non-prime attributes are fully functionally dependent upon BGCheckID, the primary key.

It is in 3NF because no non-prime attribute depends on other non-prime attributes, all non-prime attributes are dependent upon BGCheckID, the primary key.

It is in BCNF because BGCheckID, the primary key, is not dependent on any of the other non-prime attributes.

SELECT \* FROM BackgroundCheck;



**Child Entity Type:**

**Prime attributes:** ChildID

**Non-prime attributes:** EmpID, Name, DateofBirth, Gender, Race, SchoolGrade, Height, Language

**Functional dependencies:**

Primary key (ChildID) 🡪 EmpID, Name, DateofBirth, Gender, Race, SchoolGrade, Height, Language

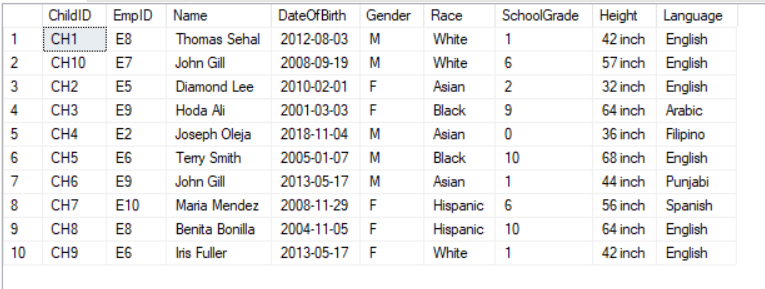
Child is in 1NF because every cell (intersection of a column and a row) contains only one single value.

It is in 2NF because no partial dependencies exist, and all non-prime attributes are fully functionally dependent upon ChildID, the primary key.

It is in 3NF because no non-prime attribute depends on other non-prime attributes, all non-prime attributes are dependent upon ChildID, the primary key.

It is in BCNF because ChildID, the primary key, is not dependent on any of the other non-prime attributes.

SELECT \* FROM Child;



**Health Entity Type:**

**Prime attributes:** HealthID

**Non-prime attributes:** ParentID, ChildID, BloodGroup, Disability, MajorIllness, SmokingHabit, DrinkingHabit, DrugAddiction, Allergies, CurrentMedicine

**Functional dependencies:**

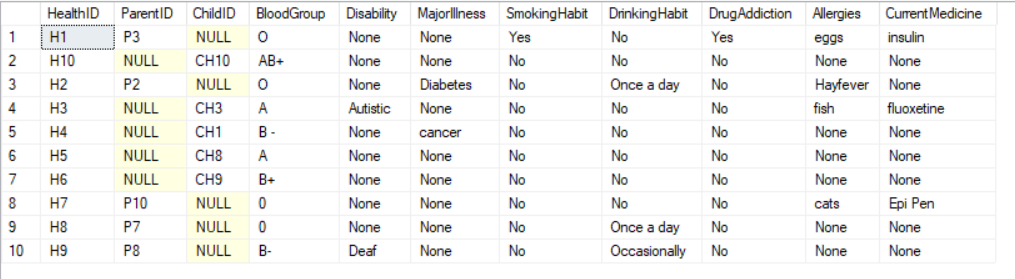
Primary key (HealthID) 🡪 ParentID, ChildID, BloodGroup, Disability, MajorIllness, SmokingHabit, DrinkingHabit, DrugAddiction, Allergies, CurrentMedicine

Health is in 1NF because every cell (intersection of a column and a row) contains only one single value.

It is in 2NF because no partial dependencies exist, and all non-prime attributes are fully functionally dependent upon HealthID, the primary key.

It is in 3NF because no non-prime attribute depends on other non-prime attributes, all non-prime attributes are dependent upon HealthID, the primary key.

It is in BCNF because HealthID, the primary key, is not dependent on any of the other non-prime attributes.

SELECT \* FROM Health;

**Center Entity Type:**

**Prime attributes:** CenterID , Name, Address

**Non-prime attributes:** ChildCount, FundsAvailable, EmployeeCount

**Functional dependencies:**

Primary key (CenterID) 🡪 Name, Address, ChildCount, FundsAvailable, EmployeeCount

Name 🡪 CenterID , Address, ChildCount, FundsAvailable, EmployeeCount

Address 🡪 CenterID, Name, ChildCount, FundsAvailable, EmployeeCount

Center is in 1NF because every cell (intersection of a column and a row) contains only one single value.

It is in 2NF because no partial dependencies; all non-prime attributes are fully functionally dependent upon CenterID, the primary key.

It is in 3NF because no non-prime attribute depends on other non-prime attributes, all non-prime attributes are dependent upon CenterID, the primary key.

It is in BCNF because CenterID, the primary key, is not dependent on any of the other non-prime attributes.

SELECT \* FROM Center;



**Use Cases & Their Realizations (SQL Queries)**

**Actor: Parent**

**Use Case Name: Register himself/herself**

**Steps:**

1. User clicks on “Parent Signup” button;
2. Prompt to enter: Username, Name, EmailID, PhoneNumber and Password.
3. Information is displayed; asks for submission;
4. Parent clicks on “Submit” button;
5. User’s details are saved in Registration;
6. User can Login to perform any other action;

**SQL Query:**

INSERT INTO Registration(RegistrationID,Username, Name, Role, EmailID, PhoneNumber, Password)

VALUES (‘R2’,'jsmith2010', 'Jenny Smith’, ‘Parent’, 'jsmith@gmail.com', ‘8324569876’, 'JCoolGirl90');

**Note: A registrationId = R2 will be generated for this parent.**

**Use Case Name: Update/Delete Registration**

**Steps:**

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the Registration information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in Registration;

**Update SQL Query**:

UPDATE Registration

SET Username = 'lily123', Name = 'Lily Singh', EmailID = ‘lily@gmail,com’, PhoneNumber = ‘5678902302’, Password = ‘kjyi43’

WHERE RegistrationID =’R3’;

**\*\*Note: Lily Singh wants to update her information.**

**Delete SQL Query:**

DELETE FROM Registration

WHERE RegistrationID= ‘R31’;

**\*\*Note: You will no longer see a tuple where RegistrationID = R31 on the Registration table.**

**Use Case Name: Enter details in Parents**

**Steps:**

1. User clicks on “Enter Parent Details” button;
2. Prompt to enter all the parent table details to enter.
3. Information is displayed; asks for submission;
4. Parent clicks on “Submit” button and ParentID is generated;
5. User’s details are saved in Parents.

**SQL Query:**

INSERT INTO Parent (ParentID, RegistrationID, Name, Gender, Address, Salary, Education, Reason)

VALUES (‘P1’,‘R4’, 'Jenny Smith', 'F', '2341 Winding Rd., Houston, Texas 77084’, '45000', 'Master’s Degree', 'Private Reasons');

**\*\*Note: A ParentID = P1 will be generated for this parent.**

**Use Case Name: Update/Delete Parent**

**Steps:**

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the Parent information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in Parent;

**Update SQL Query:**

UPDATE Parent

SET Salary = '66000', Education = 'Master’s Degree'

WHERE ParentID = ‘P2’;

\*\*Note: Lily Singh wants to update her salary and education level.

**Delete SQL Query:**

DELETE FROM Parent

WHERE ParentID = ‘P9’;

**\*\*Note: You will no longer see a tuple where Abby Ochoa, ParentID = P9 on the Parent table.**

**Use Case Name: Search and Select a child**

**Steps:**

1. User clicks on “Search Child” button;
2. Selection are made from different categories with specific parameters;
3. Basic details of all the children meeting these parameters are displayed;
4. User selects a child to view specific details;
5. User clicks on “Confirm” button;
6. His/her interest to adopt that child is submitted for approval.

**SQL Query:**

SELECT Name, DateOfBirth

FROM Child

WHERE gender = ‘F’

**\*\*Note: Here we are not only searching for an attribute group, female, from the Table Child but choosing and submitting a request for approval of a specific child, Diamond, chosen.**

SELECT DateOfBirth, Gender, Race, SchoolGrade, Height, Language

WHERE Name = ‘Diamond Lee’;

**Use Case Name: Enter Health information**

**Steps:**

1. User clicks on “Health Details” button;
2. Prompt to enter all the Health table information;
3. User clicks on “Submit” button;
4. Health details are saved;

**SQL Query:**

INSERT INTO Health (HealthID, ParentID, BloodGroup, Disability, MajorIllness, SmokingHabit, DrinkingHabit, DrugAddiction, Allergies, CurrentMedicine )

VALUES (‘H2’ , ‘P2’, ‘O’, ‘None’, ‘Diabetes’, ‘No’, ‘Once a day’, ‘No’, ‘Hayfever’, ‘None’);

**Use Case Name: Update/Delete Health**

**Steps:**

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the Health information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in Health;

**Update SQL Query:**

UPDATE Health

SET Disability = 'None', SmokingHabit = 'Yes', Allergies = ‘eggs’, CurrentMedicine = ‘insulin’

WHERE HealthID= ‘H1’;

**Delete SQL Query:**

DELETE FROM Health

WHERE HealthID= ‘H10’;

**\*\*Note: You will no longer see a tuple where HealthID = H10 in the Health table**.

**Use Case Name: Enter background check information**

**Steps:**

1. User Clicks on “BG Check” button;
2. Prompt to enter all the Health table information
3. User clicks on “Submit” button;
4. Background details are saved.

**SQL Query:**

INSERT INTO BackgroundCheck(BGCheckID, ParentID, EverCommittedFelony, ChildAbuseHistory, SexOffenderHistory, CriminalCheckResult, FinancialCheckResult)

VALUES (‘B3’ , ‘P1’,'Yes','Passed','Passed’, ‘Failed', ‘Passed’);

**Use Case Name: Update/Delete background check information**

**Steps:**

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the BackgroundCheck information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in BackgroundCheck;

**Update SQL Query:**

UPDATE BackgroundCheck

SET ChildAbuseHistory = ‘Failed', SexOffenderHistory = 'Passed'

WHERE BGCheckId = ‘B1’;

**Delete SQL Query:**

DELETE FROM BackgroundCheck

WHERE BGCheckId = ‘B10’;

**\*\*Note: You will no longer see a tuple BackgroundCheckId = B10 in the BackgroundCheck table.**

**Actor: Donor**

**Use Case Name: Register details**

**Steps:**

1. User clicks on “FundDonors Signup” button;
2. Prompt to enter all the Registration table information .
3. Information is displayed; asks for submission;
4. User clicks on “Submit” button and RegistrationID is created;
5. User’s details are saved in Registration;
6. User can Login to perform any other action;

**SQL Query:**

INSERT INTO Registration (RegistrationID, Username, Role, EmailID, PhoneNumber, Password)

VALUES (‘R1’,'samgrace1’, ‘Donor’, 'sgraceful@gmail.com', ‘2812223212’, 'SGracefulDame32');

**\*\*Note: A registrationId = R1 will be generated for this donor.**

**Use Case Name: Update/Delete Registration**

Steps:

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the Registration information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in Registration;

**Update SQL Query:**

UPDATE Registration

SET Username = 'billy76creek'

WHERE RegistrationID = ‘R4’;

**\*\*Note: Billy Creek donor candidate wants to update his username to “billy76creek”**

**Delete SQL Query:**

DELETE FROM Registration

WHERE RegistrationID = ‘R27’;

**\*\*Note: You will no longer see a tuple where RegistrationID = R27 in the Registration table.**

**Use Case Name: Select center to donate money**

**Steps:**

1. User clicks on “Center Details” button;
2. Information of all the centers is displayed;
3. User selects a center with specific parameters;
4. User is directed to selected center’s form to make donations or may opt to save CenterID to make donations later;

**SQL Query:**

SELECT Name, Address, ChildrenCount, FundsAvailable

FROM Center

WHERE FundsAvailable <= 300000;

**Use Case Name: Enter donations details**

**Steps**:

1. User Clicks on “Make Donation” button;
2. Prompt to enter all the Donor table information;
3. Information is displayed; ask for submission;
4. clicks on “Submit” button and DonorID is generated;
5. Information is saved in Donor.

**SQL Query**:

INSERT INTO Donor(DonorID, RegistrationID, CenterID, Name, Amount, NameonCreditCard)

VALUES (‘D1’, ‘R1’,’C2’, 'Samantha Grace', 12000, ‘Samantha Grace’);

**\*\*Note: Samantha has inputted $10,000 as her donation amount. Please note below she decides to change the amount in the Update Query.**

**Use Case Name: Update/Delete** Donor

**Steps**:

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the Donor information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in Donor;

**Update SQL Query:**

UPDATE Donor

SET amount = 12000

WHERE DonorID = ‘D1’;

**\*\*Note: Samantha Grace wants to instead donate $12,000 instead of $10,000**

**Delete SQL Query:**

DELETE FROM **Donor**

WHERE **DonorId = ‘D6’;**

**\*\*Note: You will no longer see a tuple where DonorID = D6 in the Donor table.**

**Use Case Name: Enter background check information**

**Steps:**

1. User Clicks on “BG Check” button;
2. Prompt to enter all the BackgroundCheck table information
3. User clicks on “Submit” button;
4. Background details are saved.

**SQL Query:**

INSERT INTO BackgroundCheck(BGCheckID, DonorID, EverCommittedFelony, ChildAbuseHistory,SexOffenderHistory, CriminalCheckResult, FinancialCheckResult)

VALUES (‘B8’,’D1’, 'No', 'Passed', 'Passed', 'Passed', 'Passed');

**Use Case Name: Update/Delete background check information**

**Steps:**

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the BackgroundCheck information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in BackgroundCheck;

**Update SQL Query:**

UPDATE BackgroundCheck

SET ChildAbuseHistory = 'Yes', SexOffenderHistory = 'Passed'

WHERE BGCheckID = ‘B1’;

**Delete SQL Query:**

DELETE FROM BackgroundCheck

WHERE BGCheckID = ‘D9’;

**Actor: Center Admin**

**Use Case Name: Enter Child details**

**Steps:**

1. User clicks on “Add new Child” Button;
2. Prompt to enter all the Child table information;
3. Information is displayed; asks for submission;
4. User clicks on “Confirm” button and ChildID is created;
5. Information is saved on Children.

**SQL Query:**

INSERT INTO Child (ChildID, EmpID, Name, DateOfBirth, Gender, Race, SchoolGrade, Height, Language )

VALUES (‘CH2’,’E5’, 'Diamond Lee','2010-02-01','F','Asian', 2, ‘32 inch’, 'English’);

**Use Case Name: Update/Delete Child details**

**Steps:**

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the Child information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in Child;

**Update SQL Query:**

UPDATE Child

SET SchoolGrade = 9, Language = 'Arabic'

WHERE ChildID = ‘CH3’;

**\*\*Note: Hoda Ali speaks Arabic and not English as primary language and is in 9th grade.**

**Delete SQL Query:**

DELETE FROM Child

WHERE ChildID = ‘CH9’;

**\*\*Note: You will no longer see a tuple where ChildID =CH9 in the Child table.**

**Use Case Name: Enter Child’s Health Details**

**Steps:**

1. User clicks on “Health Details” button;
2. Prompt to enter all the Health table information
3. User clicks on “Submit” button;
4. Health details is saved;

**SQL Query:**

INSERT INTO Health(HealthID, ChildID, BloodGroup, Disability, MajorIllness, SmokingHabit, DrinkingHabit, DrugAddiction, Allergies, CurrentMedicine )

VALUES (‘H3’,’CH3’, 'A', 'Autistic', 'None', 'No, 'No', 'No', 'fish', 'fluoxetine');

**Use Case Name: Update/Delete Health**

**Steps:**

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the Health information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in Health;

**Update SQL Query:**

UPDATE Health

SET MajorIllness = 'cancer'

WHERE HealthID = ‘H4’;

**\*\*Note: Admin updates MajorIllness in the Health table for Thomas Sehal when she gets news he has cancer.**

**Delete SQL Query:**

DELETE FROM Health

WHERE HealthID = ‘H5’;

**\*\*Note: You will no longer see a tuple where HealthID= H5 in the Health table.**

**Use Case Name: Enter Center Details**

**Steps:**

1. User clicks on “Add new Center” Button;
2. Prompt to enter all the Center table information;
3. Information is displayed; asks for submission;
4. User clicks on “Confirm” button and CenterID is created;
5. Information is saved on Center.

**SQL Query:**

INSERT INTO Center (CenterID, Name,Address)

VALUES (‘C3’, 'Northwest Location', '2983 Hwy 6, Houston, Texas 77081');

**Use Case Name: Update/Delete Center details**

**Steps:**

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the **Center** information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in **Center**;

**Update SQL Query:**

UPDATE Center

SET Adress = ‘2312 West Airport’, Sugarland, Texas 77478’

WHERE CenterID = ‘C4’;

**\*\*Note: The Town Center Location has moved, so the admin is updating the address.**

**Delete SQL Query:**

DELETE FROM Center

WHERE CenterID = ‘C6’;

**\*\* Due to funding issues, center C6 is no longer part of the adoption centers list.**

**Use Case Name: Assign Employee to Child**

Steps:

1. User clicks on “Search Employee” button;
2. User select employees with lowest case load to assign to child;
3. User enters EmpID to assign employee and prompts to confirm;
4. User clicks “confirm” button;
5. Employee is assigned to ChildID.

**SQL Query:**

UPDATE Child

SET EmpID =

(SELECT EmpID

FROM Employee

WHERE ChildCount =

(SELECT MIN (ChildCount)

FROM Employee))

WHERE ChildID = ‘CH2’;

**Use Case Name: Approve/Reject Parents application**

**Steps:**

1. User clicks on “View Parent Applications” button;
2. All submitted applications are displayed;
3. User selects application and clicks either “Accept” or “Reject” button;
4. Decision is saved with that application in is Approved field of Parents.

**SQL Query:**

SELECT \*

FROM Parent p, BackgroundCheck b, Health h

WHERE p.ParentID = b.ParentID AND b.ParentID = h.ParentID;

UPDATE Parent

SET isApproved = 'No’

WHERE ParentID = ‘P2’;

**Use Case Name: Approve/Reject FundDonors application**

**Steps:**

1. User clicks on “View Donors Applications” button;
2. All submitted applications are displayed;
3. User selects application and clicks either “Accept” or “Reject” button;
4. Decision is saved with that application in is DonationApproved field of Donor.

**SQL Query:**

SELECT \*

FROM Donor d, BackgroundCheck b

WHERE d.DonorIdD= b.DonorID;

UPDATE Donor

SET isCardApproved = ‘yes’, isDonationApproved = 'Yes'

WHERE DonorID = ‘D1’;

**Actor: Employee**

**Use Case Name: Enter Details**

Steps:

1. User clicks on “Employee detail” button;
2. Prompts to enter: Name, Address, Designation, JobStartDate, JobEndDate, ChildCount, and Salary.
3. Information is displayed; ask for submission;
4. User clicks on “Confirm” button: EmpID is created;
5. Information is saved in Employee

**SQL Query:**

INSERT INTO Employee (EmpID, RegistrationID, CenterID, Name, Address, Designation, ChildCount, JobStartDate, JobEndDate, Salary)

VALUES (‘E1’, ‘R21’, ‘E1’, 'Preeta Singh', ‘897 Connor St., Conroe, Texas 77320’, 'accountant’, 0, '2018-02-01', NULL, 60000)

**Use Case Name: Update/Delete Employee details**

**Steps:**

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the **Employee** information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in **Employee**;

**Update SQL Query:**

UPDATE Employee

SET Designation = 'Caretaker'

WHERE EmpID = ‘E5’;

**\*\* Note: Sally Hayes is updating her designation as a caretaker.**

**Delete SQL Query:**

DELETE FROM Employee

WHERE EmpID= ‘E4’;

**\*\*Note: Tuple where EmpID = E4 in Employee table will no longer be there.**

**Use Case Name: Enter background check information**

**Steps:**

1. User Clicks on “BG Check” button;
2. Prompt to enter all the Health table information
3. User clicks on “Submit” button;
4. Background details are saved.

**SQL Query:**

INSERT INTO BackgroundCheck(BGCheckID, EmpID, EverCommittedFelony, ChildAbuseHistory, SexOffenderHistory, CriminalCheckResult)

VALUES (‘B9’, ‘E1’,'No','Passed','Passed','Passed','Passed’)

**Use Case Name: Update/Delete background check information**

**Steps:**

1. User clicks on “Update/Delete” button;
2. Prompt to ask an option between update or delete.
3. If user clicks update, then Prompts to enter all the BackgroundCheck information for updating.
4. If User clicks delete button, then prompts to enter information for deletion.
5. User’s Information is updated or deleted in BackgroundCheck;

**Update SQL Query:**

UPDATE BackgroundCheck

SET EverCommittedFelony = 'Yes', ChildAbuseHistory = 'Failed’, SexOffenderHistory = ‘Failed’, CriminalCheckResult = “Failed”, FinancialCheckResult = ‘Failed’

WHERE BGCheckID= ‘B10’;

**\*\*Note: This prospective employee will not be hired because he did not pass several background check tests.**

**Delete SQL Query:**

DELETE FROM BackgroundCheck

WHERE BGCheckID = ‘B10’;

**\*\*Note: The tuple BGCheckID = B10 is no longer going to be on the BackgroundCheck table.**

**Test Plan of Successful Use Case Realizations & Their Records**

**All Entities’ Data 🡪 Please see the BCNF Verification section to see the full data tables successfully displayed along with their SQL SELECT \* queries.**

**Aggregate Use Cases:**

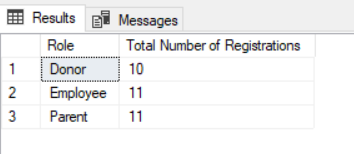
**Use Case Name: View count of All Registrations**

**Steps:**

1. User Clicks on “All Registrations” Button.
2. Total Number of Registrations by role count is displayed.

**SQL Query:**

SELECT Role, COUNT (RegistrationID ) AS "Total Number of Registrations" FROM Registration group by Role;



**Use Case Name: View count of All FundDonors**

**Steps:**

1. User Clicks on “All **FundDonors**” Button.
2. Total Number of **FundDonors** count is displayed.

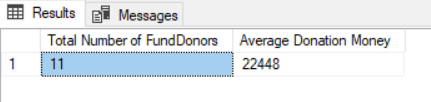
**SQL Query:**

SELECT COUNT (donorId) AS "Total Number of FundDonors",

AVG(amount) AS "Average Donation Money"

FROM FundsDonor;

SELECT COUNT (DonorID) AS "Total Number of FundDonors", AVG(Amount) AS "Average Donation Money" FROM Donor;



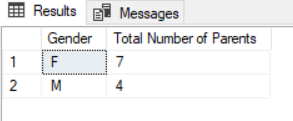
**Use Case Name: View count of Female and Male Parents**

**Steps:**

1. User Clicks on “Count Parents” Button.
2. Total Number of Parents count is displayed by gender.

**SQL Query:**

SELECT Gender,COUNT(ParentID) AS "Total Number of Parents" FROM Parent GROUP BY Gender;



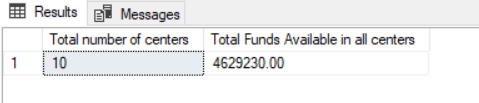
**Use Case Name: View Total Funds Available in all Centers**

**Steps:**

1. User Clicks on “View Total Funds” Button.
2. Total Amount of Funds is displayed.

**SQL Query:**

SELECT COUNT(CenterID) AS "Total number of centers", SUM(FundsAvailable) AS "Total Funds Available in all centers" FROM Center;



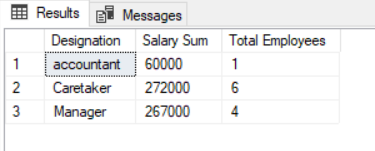
**Use Case Name: View count and sum of salaries of employees based on designation**

**Steps**:

1. User Clicks on “Employee Designation Details” Button.
2. Total Number of Employee working on specific designation and their salary sum will be displayed.
3. SQL Query:

**SQL Query:**

SELECT Designation, SUM(Salary) AS "Salary Sum", COUNT(EmpID) AS "Total Employees" FROM Employee GROUP BY Designation;



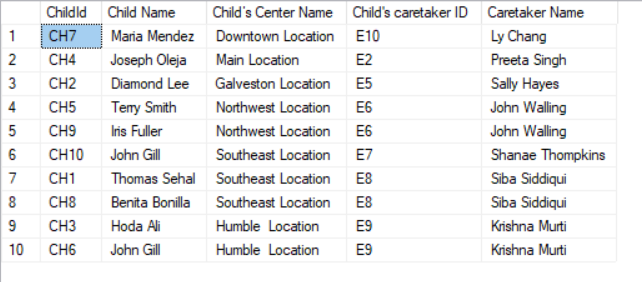
**Use Case Name: View Number of Children in Specific Center**

**Steps:**

1. User Clicks on “View Children” Button.
2. Total Number of Children Available in specific center with its centerId is displayed.

**SQL Query:**

SELECT ch.ChildId, ch.Name AS "Child Name", ce.Name AS "Child’s Center Name", e.EmpID AS "Child's caretaker ID",e.Name AS "Caretaker Name" FROM Child ch, Center ce,Employee e WHERE ch.EmpID = e.EmpID and e.CenterID = ce.CenterID;



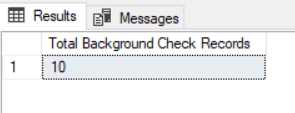
**Use Case Name: View count of All BackgroundCheck Records**

**Steps:**

1. User Clicks on “**Al**l **BackgroundCheck**” Button.
2. Total Number of **BackgroundCheck** count is displayed.

**SQL Query:**

SELECT COUNT(BGCheckID) AS "Total Background Check Records" FROM BackgroundCheck;



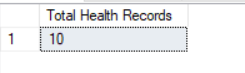
**Use Case Name: View count of All Health Records**

**Steps:**

1. User Clicks on **“All** **Health**” Button.
2. Total Number of **Health** count is displayed.

**SQL Query:**

SELECT COUNT(HealthID) AS "Total Health Records" FROM Health;

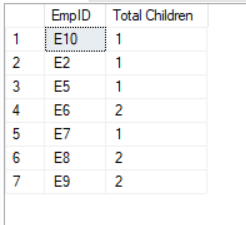


**Use Case Name: View average donation of All Employee Records**

**Steps:**

1. User Clicks on **“Employee Details”** Button.
2. Total Number of childrencount is displayed for employees.

SELECT EmpID, COUNT(ChildID) As "Total Children" FROM Child GROUP BY EmpID;



**Relationship Use Cases:**

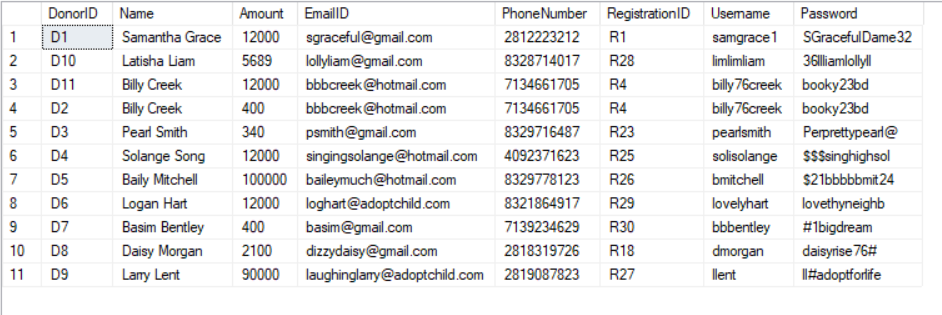
**Use Case Name: Get FundDonor’s Registration Detail**

Steps:

1. User Clicks on “View FundDonor’s Registration detail” Button.
2. Information from FundDonors and Registration table is displayed.

**SQL Query:**

SELECT d.DonorID,d.Name,d.Amount,r.EmailID,r.PhoneNumber,r.RegistrationID,r.Username,r.Passwod FROM Donor d, Registration r WHERE d.RegistrationID = r.RegistrationID;



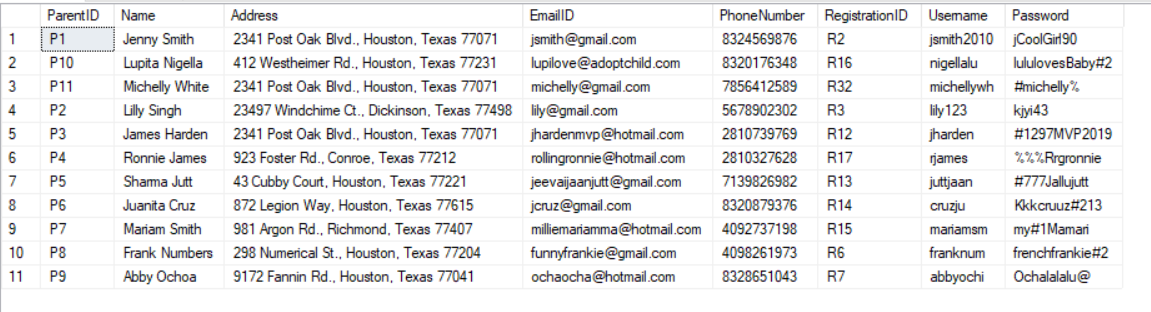
**Use Case Name: Get Parent’s Registration Detail**

**Steps:**

1. User Clicks on “View Parent’s Registration detail” Button.
2. Information from Parent and Registration table is displayed.

**SQL Query:**

SELECT p.ParentID,p.Name,p.Address, r.EmailID,r.PhoneNumber,r.RegistrationID,r.Username,r.Password FROM Parent p, Registration r WHERE p.RegistrationID = r.RegistrationID;



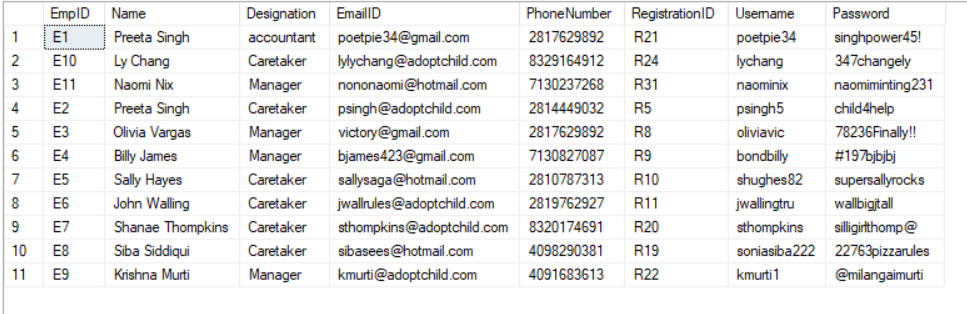
**Use Case Name: Get Employee’s Registration Detail**

**Steps:**

1. User Clicks on “View Employee’s Registration detail” Button.
2. Information from Employee and Registration table is displayed.

**SQL Query:**

SELECT e.EmpID,e.Name,e.Designation ,r.EmailID,r.PhoneNumber,r.RegistrationID,r.Username,r.Password FROM Employee e, Registration r WHERE e.RegistrationID = r.RegistrationID;



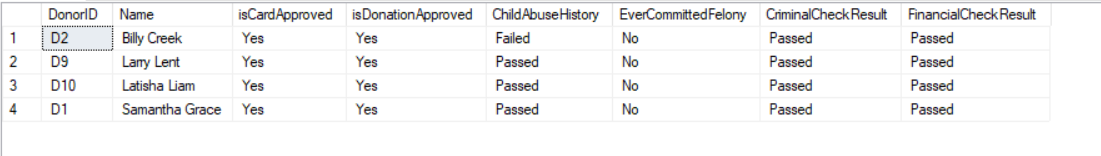
**Use Case Name: View FundDonor’s Background Check Detail**

**Steps:**

1. User Clicks on “View FundDonor’s Background check detail” Button.
2. Information from FundDonors and BackgroundCheck table is displayed.

**SQL Query:**

SELECT d.DonorID,d.Name,d.isCardApproved,d.isDonationApproved,b.ChildAbuseHistory,b.EverCommittedFelony,b.CriminalCheckResult,b.FinancialCheckResult FROM Donor d, BackgroundCheck b WHERE d.DonorID = b.DonorID;



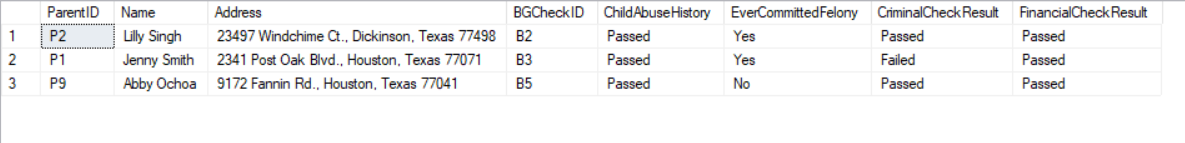
**Use Case Name: View Parent’s Background Check Detail**

**Steps:**

1. User Clicks on “View Parent’s Background check detail” Button.
2. Information from Parent and BackgroundCheck table is displayed.

**SQL Query:**

SELECT p.ParentID,p.Name,p.Address,b.BGCheckID,b.ChildAbuseHistory,b.EverCommittedFelony, b.CriminalCheckResult,b.FinancialCheckResult FROM Parent p, BackgroundCheck b WHERE p.ParentID = b.ParentID;



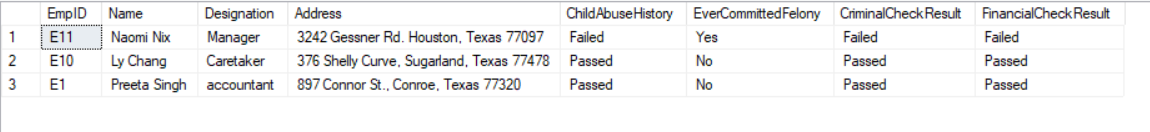
**Use Case Name: View Employee’s Background Check Detail**

**Steps:**

1. User Clicks on “View Employee’s Background check detail” Button.
2. Information from Employee and BackgroundCheck table is displayed.

**SQL Query:**

SELECT e.EmpID,e.Name,e.Designation,e.Address,b.ChildAbuseHistory,b.EverCommittedFelony,b.CriminalCheckResult,b.FinancialCheckResult FROM Employee e, BackgroundCheck b WHERE e.EmpID = b.EmpID;



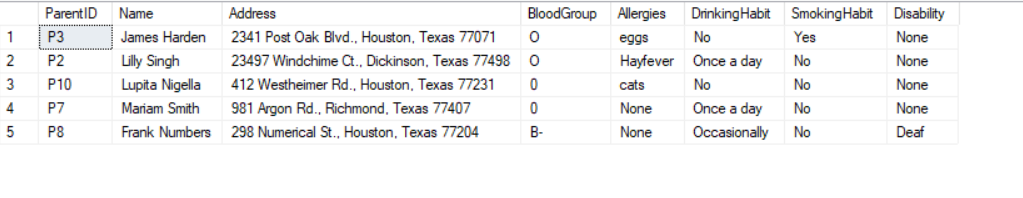
**Use Case Name: View Parent’s Health Detail**

**Steps:**

1. User Clicks on “View Parent’s Health detail” Button.
2. Information from Parent and Health table is displayed.

**SQL Query:**

SELECT p.ParentID,p.Name,p.Address,h.BloodGroup,h.Allergies,h.DrinkingHabit,h.SmokingHabit,h.Disability FROM Parent p, Health h WHERE p.ParentID = h.ParentID;



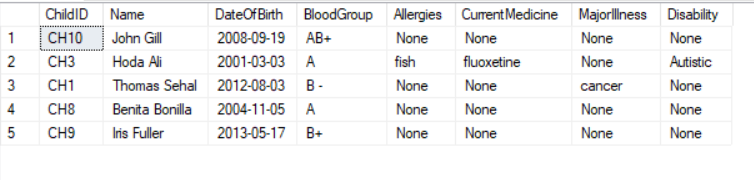
**Use Case Name: View Child’s Health Detail**

**Steps:**

1. User Clicks on “View Employee’s Health detail” Button.
2. Information from Child and Health table is displayed.

**SQL Query:**

SELECT ch.ChildID,ch.Name,ch.DateOfBirth ,h.BloodGroup,h.Allergies,h.CurrentMedicine, h.MajorIllness,h.Disability FROM Child ch, Health h WHERE ch.ChildID = h.ChildID;



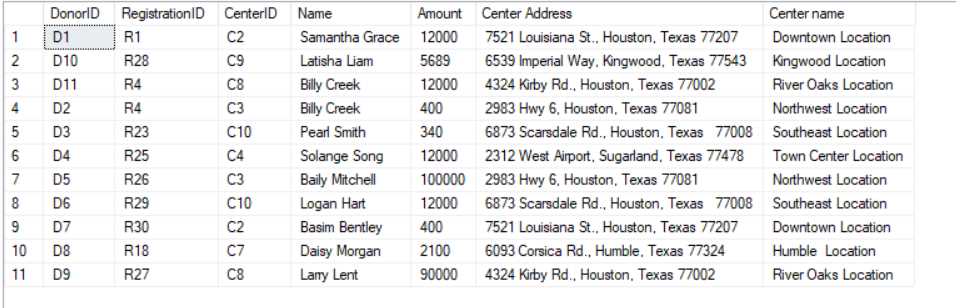
**Use Case Name: View FundsDonor’s Donation Detail with Center**

**Steps:**

1. User Clicks on “View FundDonor’s Donation Detail” Button.
2. Information from FundDonors and Center table is displayed.

**SQL Query:**

SELECT d.DonorID,d.RegistrationID,d.CenterID,d.Name,d.Amount,ce.Address AS "Center Address", ce.Name as "Center name" FROM Donor d, Center ce WHERE d.CenterID = ce.CenterID;



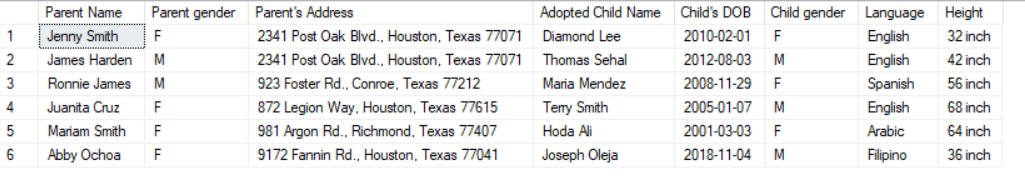
**Use Case Name: View Parent’s Adoption Detail**

Steps:

1. User Clicks on “View Parent’s Adoption detail” Button.
2. Information from Parent and Child table is displayed.

**SQL Query:**

SELECT p.Name AS "Parent Name",p.Gender as "Parent gender",p.Address as "Parent's Address",ch.Name AS "Adopted Child Name",ch.DateOfBirth AS "Child's DOB",ch.Gender AS "Child gender", ch.Language,ch.Height FROM Parent p, Child ch WHERE p.ChildID = ch.ChildID;



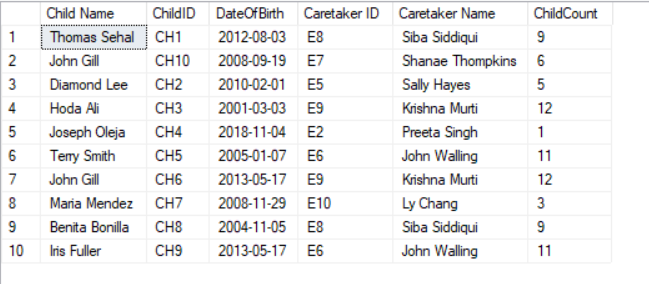
**Use Case Name: View Child’s Care Taker Detail**

**Steps:**

1. User Clicks on “View Child’s Care Taker Detail” Button.
2. Information from Employee and Child table is displayed.

**SQL Query:**

SELECT ch.Name AS "Child Name",ch.ChildID,ch.DateOfBirth,e.EmpID AS "Caretaker ID",e.Name AS "Caretaker Name",e.ChildCount FROM Child ch, Employee e WHERE ch.EmpID = e.EmpID;



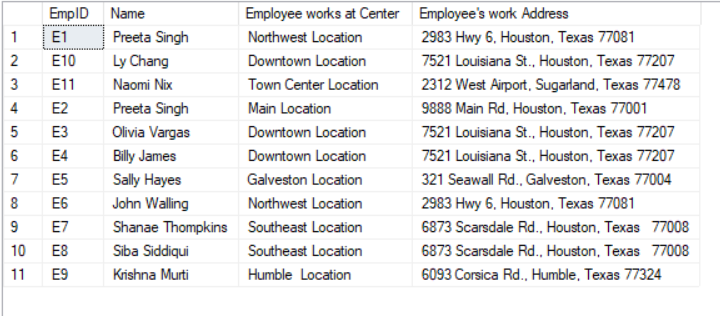
**Use Case Name: View Employee works at Which Center**

**Steps:**

1. User Clicks on “View Employee’s Center Information” Button.
2. Information from Employee and Center table is displayed.

**SQL Query:**

SELECT e.EmpID, e.Name, ce.Name AS "Employee works at Center",ce.Address AS "Employee's work Address" FROM Employee e, Center ce WHERE e.CenterID = ce.CenterID;



**Summary**

**Conclusion:**

The “Child Adoption Database System” has been implemented successfully in fulfilling the mission statement and mission objectives that were set forth. Use cases have been tested and run smoothly. Hence, this system has made it very easy for parents to adopt a child and for donors to make donations through a centralized system.

Due to the initial database creation’s success, we are now able to expand our horizons on future possibilities. Future development may include deeper relationship exploration between employees and their designation roles. Another possible avenue of reflective use of the database is promoting center resources efficacy based on funds and employee allocations. Furthermore, upcoming innovative steps may include connectivity for the people who want to celebrate special occasions with children at the adoption agency. They may book and organize the function(s) at the centers in conjunction with the assigned employees.

**References:**

**Course Textbook:**

Connolly, Thomas M., and Carolyn E. Beg. *Database Systems: a Practical Approach to Design, Implementation, and Management*. Pearson Education, 2015.

**PDatabase Platform:** SQL Server 2017