LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC6001NI Advanced Database System Development

40% Individual Coursework

Student Name: Rishabh Singh

London Met ID: 20048981

College ID: npo1cp4s210251

Assignment Due Date: 15th March 2023

Assignment Submission Date: 15th March 2023

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

Table of Figure

Table of table

1. Introduction

Database is a collection of organized data that is stored and accessed electronically which makes it easier to store large amounts of information both secure and efficiently. Now a days database management system is used by many organizations, businesses, research, and ecommerce. Along with that many businesses and organizations are moving to online voting methods for different purposes like choosing managers, employee of the month and many more.

In the given case study, we have a similar scenario where an organization is planning to establish a full fledge online voting system which will help them choose the employe of the month. After going through the requirements, we can know that an employee can vote any candidate from any department except themselves. The job history is also to be stored so we can find out the employee's current department as well as their previous and there are more features like saving voting details address etc. Based on the case study we are required to build an initial ERD, then move further to normalization which will be done till 3nf. Then developing a final ERD and making required assumptions and implementing it. We are also required to document the system and all the possible success and failed test cases.

The development of the system will be do ne using different tools and combining it together first we take Oracle SQL developer and create an er model then generate DDL script and run it in SQL developer and then implement the insert statements, Finally we move to ASP.NET webform where we display the table and dynamically edit update delete and insert in the tables also we combine different tables to get reasonable outputs which can be used later by the organization for their internal affairs we also delete and remove all data redundancy, anomalies and many to many relationships. After the completion of this the database will be fully functional and well tested.

2. Textual Analysis

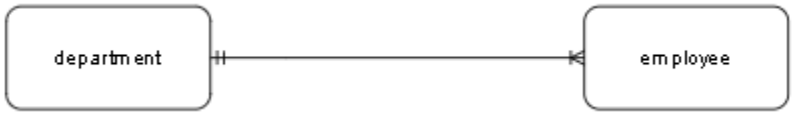


Figure 1: Employee department relation

Description: multiple employees can belong to one department, but each employee can belong to only one department.

Figure 2: Department role relation

Description: each department can have multiple roles, but each role can belong to only one department.

Figure 3: Role employee relation

Description: multiple employees can have the same role, but each employee can have only one role.

Figure 4: Job_History employee relation

Description: as each job history record belongs to only one employee, but each employee can have multiple job history records.

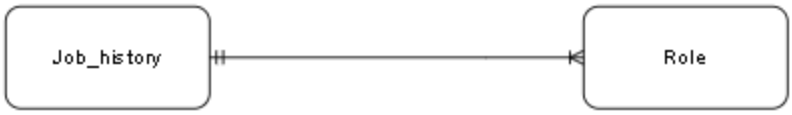


Figure 5: Job history role relation

Description: each job history record is associated with only one role.

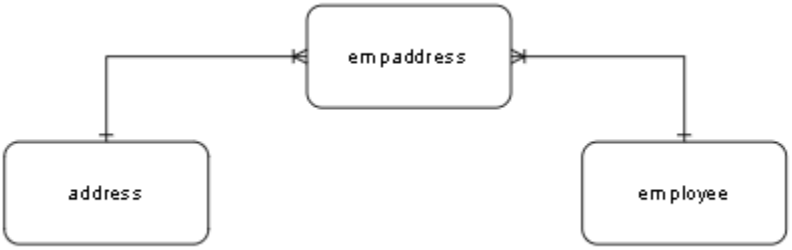


Figure 6: employee and address

Description: An employe can have multiple address and an address can be assigned to multiple employees which will be many too many to resolve it we add bridge table.

Figure 7: Job history department

Description: each job history record is associated with only one department.

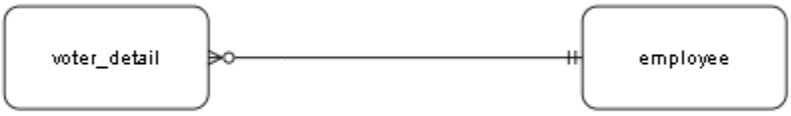


Figure 8: voter detail employee

Description: many-to-one, where each voter must be an employee, but not every employee needs to have a corresponding record in the Voter detail table.

3. ERD from Case Study

An Entity-Relationship Diagram (ERD) is a visual representation of data that shows the relationships between different entities. Entities are objects or concepts that have attributes, and relationships show how these entities are related to each other.

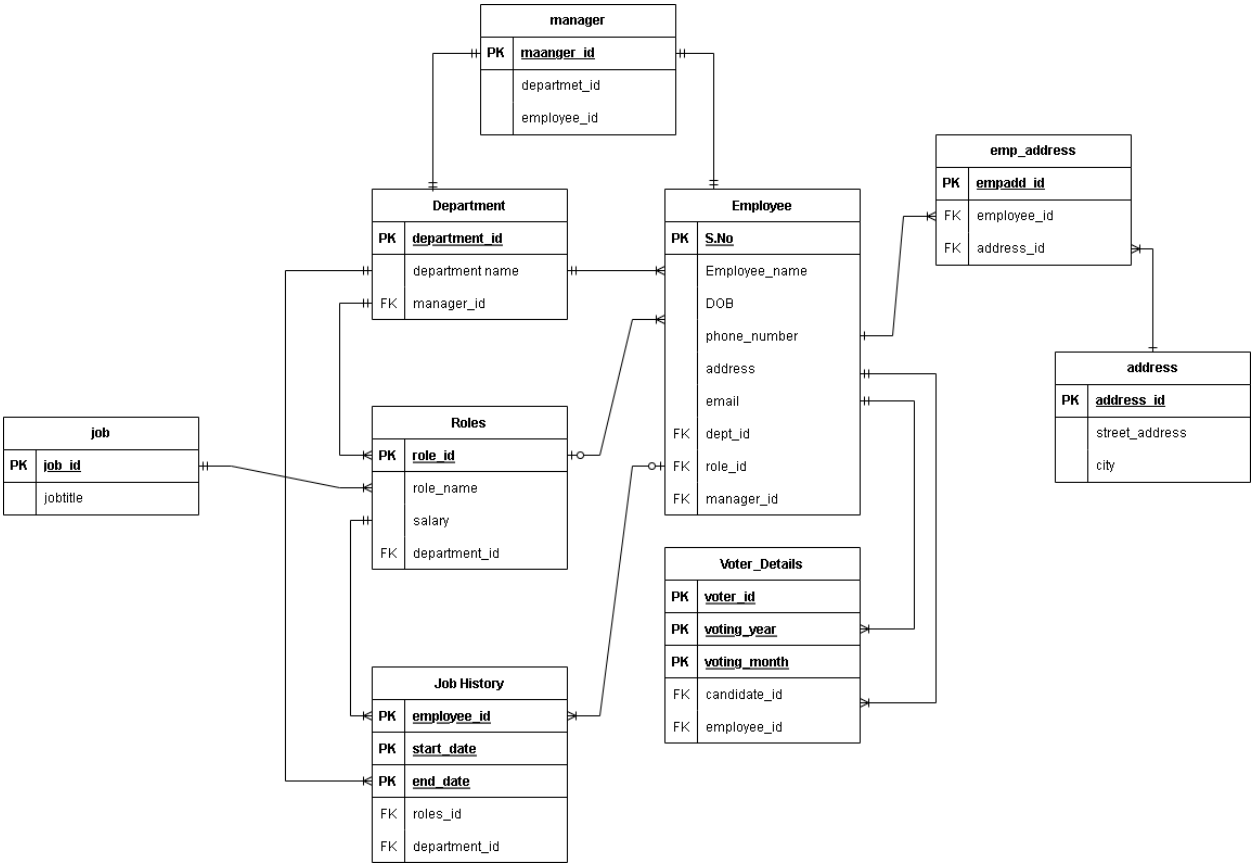


Figure 9: Initial ERD

4. Normalization

Normalization is a database design approach that avoids undesired characteristics such as Insertion, Update, and Deletion Anomalies by reducing data redundancy. Normalization rules break huge tables into smaller tables and use relationships to connect them. The goal of SQL normalization is to reduce duplicate data and ensure logical data storage. In this scenario we were asked to do normalization till 3NF.

4.1 Normalization of fig 1

S.N.	Employee Name	Date of Birth	Contact	Email Address	Address	Department
1	Erling Haland	1990-Aug-10	9876578768	Erling@gmail.com	New Baneshwor, Kathmandu, Nepal	Finance
2	Dejan Kulusevski	1988-Sep-20	9837476274	Dejan@gmail.com	Begnas Lake, Pokhara, Nepal	Human Resource
3	Lisandro Martinez	1989-Jan-12	9876565656	Lima@gmail.com	Itahari, Koshi, Nepal Sinamangal, Kathmandu, Nepal	Marketing
4	Raphael Varane	1990-Feb-14	9812373090	Raphael@gmail.com Dipileush@gmail.com	Sinamangal, Kathmandu, Nepal	Human Resource

Figure 10: Normalization fig 1

UNF

Employee: (S.No, emp_Name, dob, contact, {Email}, {address}, department)

S.No is taken as primary key and the curly braces are taken as repeating groups.

1NF

Employee: (S.No, emp_Name, dob, contact, dept_id, dept_name)

Emp-Email: (S.No*, Email)

Emp-address: (S.No*, address)

Here,

▪ Repeating groups were removed from Employee table to new tables.

▪ In Emp-Email table, S.No and Email were combined to form composite

primary key.

▪ A new attribute was added to Employee table called dept_Id.

▪ Department column was renamed as dept_Name.

2NF

▪ Since the Employee table doesn't consist of a composite key, it Is already in

2NF.

▪ Since the Emp-Email table doesn't consist of any non-key attribute, it Is already

in 2NF.

Since in the address table there exist a composite primary key and other table doesn't have any composite primary key so they are already in "2NF" but the address table has the possibility of partial dependency.

Checking partial dependency in address table

Address Id → Address

S.N. →Address Id , S.N. →

- After looking at the scenario the address attribute is partially dependent on the employee table peimary key to resolve this we created a new table address. Which contains address_id primary key and address atribute

Tables after 2NF:

Employee: (S.No, emp_Name, dob, contact, dept_Id, dept_Name)

Emp-Email: (S.No*, Email)

Emp-address: (S.No*, address_id*)

Address (Address_id, Address)

3NF

For Employee table:

S.No => emp_Name, dob, contact => X

S.No => dept_Id => dept_Name

▪ Transitive dependency is found in Employee table so it is broken down into new table named as Departments.

Employee (S.No, emp_Name, dob, contact, dept_Id*)

Departments (dept_Id, dept_Name)

Emp-Email: (S.No, Email)

Emp-address: (S.No, address)

Note:

▪ Emp-Email and Emp-address table are already in 3NF as they don't have any non-key attribute.

▪ Since, Transitive dependency is found in Employee table so it is broken down into new table named as Departments.

Final Tables:

Employee (S.No, emp_Name, dob, contact, dept_Id*)

Departments (dept_Id, dept_Name)

Emp-Email: (S.No, Email)

Emp-address: (S.No, address)

4.2 Normalization of fig 2

Voter ID	*Voter Name*	*Voting Year*	*Voting Month*	*Candidate ID*	*Candidate Name*	*Candidate Department*
1	Erling Haland	2019	January	*88*	*Nate Diaz*	*IT*
1	Erling Haland	*2019*	February	*132*	*Kamaru Usman*	*Finance*
1	Erling Haland	*2020*	August	*420*	*Dana White*	*Human Resource*
1	Erling Haland	2020	September	*7*	*khabib nurmagomedov*	*Finance*

Figure 11: Normalization figure 2

Assumptions:

Only one vote can be submitted by a person to one candidate each month.

UNF

Voter (voter_Id, voter_Name, {voting_year, {voting_month, candidate_Id, candidate_Name, candidate_department}}

▪ voter_Id is chosen as the primary key.

▪ voting_year is identified as a repeating group.

▪ voting_month, candidate_Id, candidate_Name, candidate_department are identified as a nested repeating group of the Voter table.

1NF: Removing repeating groups.

Voter (voter_id PK, voter_name)

Vote_Detail (voter_id FK, candidate_id FK, voting_year, voting_month)

Vote-detail (voter_Id*, voting_year, voting_month, candidate_Id, candidate_Name, candidate_department)

Note:

The primary key for the Voter table is voter_id.

The Vote_Detail table has a composite primary key consisting of voter_id, candidate_id, voting_year, and voting_month.

▪ A new attribute department_id was added to resolve the delete anomaly.

2NF: Removing partial dependencies.

For table Voter:

▪ Since the Voter table has no composite primary key, there is no partial dependency. Hence, the table is already in 2NF.

• For table Voter-year

▪ Since the Voter-year table does not contain any non-key attributes, the table is already in 2NF.

For the voter-detail table:

voter_Id => X

voting_year => X

voting_month => X

voter_Id*, voting_year* => X

voting_year*, voting_month => X

voter_Id*, voting_month => X

voter_Id*, voting_year*, voting_month => candidate_Id, candidate_Name, department_id, candidate_department

Since no partial dependency was found, the table voting-detail is in 2NF.

Tables after 2NF:

Voter (voter_Id, voter_name)

Voter-year (voter_Id*, voting_year)

Vote-detail (voter_Id*, voting_year*, voting_month, candidate_Id, candidate_Name,

department_Id, candidate_department)

3NF: Removing transitive dependencies.

For table Voter:

▪ Since there is only one non-key attribute, the table is already in 3NF.

For the Vote-detail table:

voter_Id*, voting_year*, voting_month => candidate_Id, department_Id

Candidate_Id => candidate_Name

Department_Id => candidate_department

▪ Since there are transitive dependencies, we remove them and create new tables.

Voter_detail (voter_Id*, voting_year*, voting_month, candidate_Id*, department_Id*)

Candidate (candidate_Id, candidate_Name)

Department (department_Id, candidate_department)

Note:

▪ Since transitive dependencies were found in table voting-detail, they were

removed and formed new tables which are Candidate and Department with

primary key candidate_Id and department_Id.

Final tables:

Voter (voter_id PK, voter_name)

Candidate (candidate_id PK, candidate_name, department_id FK)

Department (department_id PK, department_name)

Vote_Detail (voter_id FK, candidate_id FK, voting_year, voting_month, department_id FK)

5. Integration and Assumption

5.1. Integration

The normalization of Figure 1 and Figure 2 also with the entities of the ERD from the case study was taken into consideration and integrated together.

Address: (address_id PK, street_address, city, country)

Employee: (employee_id PK, employee_name, date_of_birth, contact, department_id FK, role_id FK, manager_id FK)

Manager_dept: (manager_id FK, department_id PK FK)

Department: (department_id PK, department_name)

Roles: (role_id PK, role_name, salary, department_id FK, JOB_ID FK)

Emp-email: (employee_id PK FK, email PK)

Employee-address: (employee_id PK FK, address_ID PK FK)

Job_History: (employee_id PK FK, start_date PK, end_date, role_id FK, dept_id FK)

Voter_Detail: (voter_id PK FK, voting_year PK, voting_month PK, candidate_id FK)
JOBS: (job_id PK, job_title, min_salary, max_salary)

5.1. Assumption:

- Employee can have multiple address and an address can be assigned to multiple employees to handle that bridge entity was created.
- Since employee can be both voter and candidate so voter_detail table references to employee_id for both voter and candidate.
- The department has manager and the manager is also the employee so to resolve the the data namolies a separate table manager_dept was created.
- In the scenario the department table is linked inside the employee table now if we change the department name there will be an update anomalies to prevent that we created department table.
- Also the same job can have different roles which may lead or chasm or fan trap to solve it role table references jobs.
- As this database is for employee of the month Only one vote can be submitted by an employee to one candidate each month to acheive it we have taken voter_id, year and month as composite primary key.
- Also end date in job_history is set to null so when the end date attrivute is null we can know that the employee is still working in that job.

6. Final ERD

An Entity-Relationship Diagram (ERD) is a visual representation of data that shows the relationships between different entities. Entities are objects or concepts that have attributes, and relationships show how these entities are related to each other. The relationships between these entities would be depicted in the ERD using lines and symbols. For example, we might show that each book is written by one or more authors using a "one-to-many" relationship symbol. Similarly, we might show that each customer can place many orders, but each order is associated with only one customer using another "one-to-many" relationship symbol.

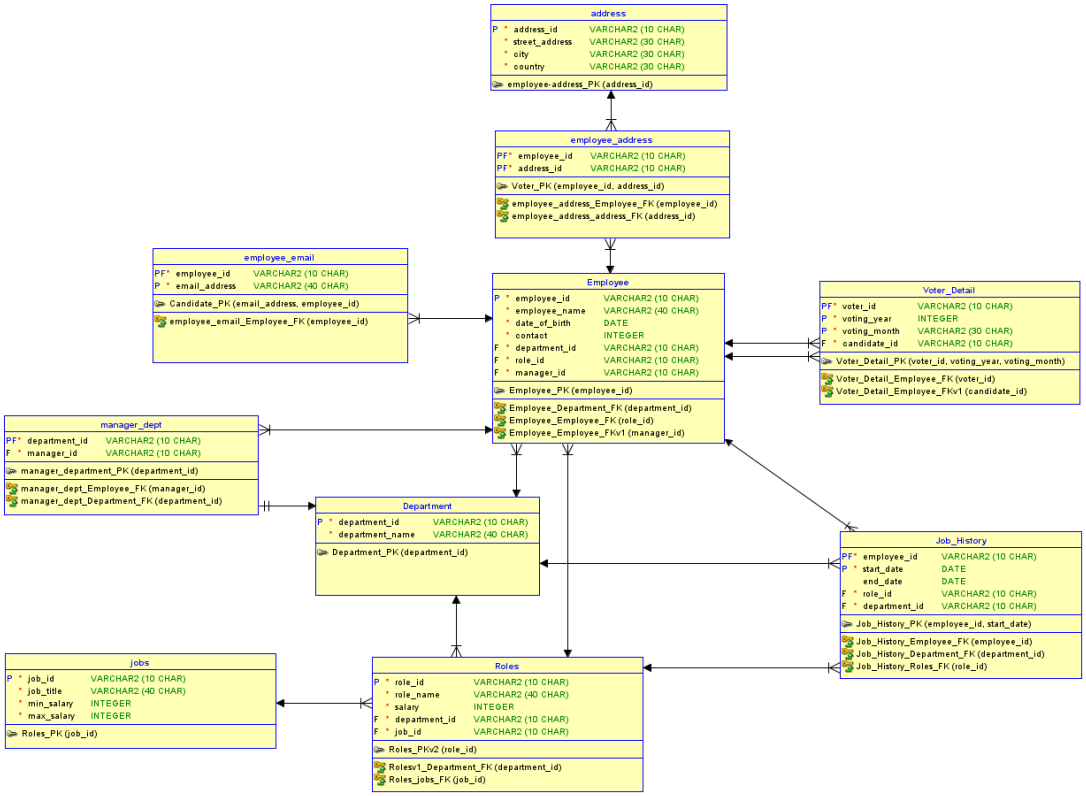


Figure 12: Final ERD

7. Data Dictionary

7.1 Data Dictionary for address table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Data
address_id	Varchar2	10	Primary Key Not null			To uniquely identify Each address	A001
street_address	Varchar2	30	Not null			To store the address name	Kumarigal
city	Varchar2	30	Not null			To store city name	Kathmandu
country	Varchar2	30	Not null			To store country name	Country

Table 1: Address

7.2 Data Dictionary for department table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Data
department_id	Varchar2	10	Primary Key Not null			To uniquely identify Each department	D001
department_name	Varchar2	40	Not null			To store the addresss name	Sales

Table 2: Department

7.3Data dictionary for jobs table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Data
job_id	Varchar2	10	Primary Key Not null			To uniquely identify Each job	1
job_title	Varchar2	40	Not null			To store the job title	Account Executive
min_salary	INTIGER		Not null			To store minimum salary	50000
max_salary	INTIGER		Not null			To store minimum salary	100000

Table 3: Jobs

7.4 Data dictionary for role table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Data
role_id	Varchar2	10	Primary Key			To uniquely identify Each role of the employee	1
role_name	Varchar2	40	Not null			It gives the title of the role of employee and stores it	Accountant
salary	Integer	10	Not null			It gives the job_id of the employee	50000
Department_id	Varchar2	10	Foreign Key			To store roles department	D001
Job_id	Varchar2	10	Foreign Key	job	job_id	To store job's role	2

Table 4: Role

7.5 Data dictionary for manager_dept table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Data
manager_id	Varchar2	10	Foreign Key	employee	emp_id	To store which employee is manager	2
department_id	Varchar2	10	Primary Key, Foreign Key	department	department_id	To store manager department allocation	D001

Table 5: Manager

7.6 Data Dictionary for employee table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Data
employee_id	Varchar2	10	Primary Key			To uniquely identify Each employee	1
Date_of_birth	DATE		Not null			To store employee date of birth	1990-02-10
contact	Varchar2	20	Not null			To store employee contact	9803542731
employee_name	Varchar2	40	Not null			To store employee name	Elizabeth Nguyen
department_id	Varchar2	10	Foreign Key	department	department_id	To store the employees allocated department	D001

role_id	Varchar2	10	Foreign Key	role	role_id	To store employees allocated role	R002
manager_id	Varchar2	10	Foreign Key	employee	emp_id	To store employees allocated manager	3

7.7 Data dictionary of employee_address table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Data
employee_id	Varchar2	10	Foreign Kry	employee	emp_id	To uniquely identify employee	1
address_id	Varchar2	10	Primary Key, Foreign Kry	address	address_id	To store employees address	A003

7.8 Data dictionary for email table

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Data
employee_id	Varchar2	10	Primary Key, Foreign Key	employee	emp_id	It uniquely identifies employee	1

Email_address	Varchar2	40	Primary Key			Stores the employee address	ram@gmail.com

7.9 Data dictionary for job-history

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Data
employee_id	Varchar2	10	Primary Key	employee	emp_id	To uniquely identify Each employee with their emp_id	1
start_date	DATE		Primary Key			To store start date of the employee job	2015-01-01
end_date	DATE		Null			To store end date of the employees job	2020-12-31
Role_id	Varchar2	10	Foreign Key	role	Role_id	To store the employees previous role	R001
Department_id	Varchar2	10	Foreign Key	role	department_id	To store the employees previous department	D001

7.10 Data dictionary for voter_detail

Column Name	Data Type	Size	Constraint	Reference Table	Reference Column	Description	Example Data
voter_id	Integer	10	Primary Key, Foreign Key	employee	employee_id	To uniquely identify the voter	1

voting_year	Varchar2	10	Primary Key			To store voting year	2022
voting_month	Varchar2	10	Primary Key			To store voting month	March
candidate_id	Integer		Foreign Key	employee	employee_id	To uniquely identify candidate	3

8. Script

The DDL script was generated using SQL data modeler below is the script that was generated.

```
  1. CREATE TABLE address (
  2.     address_id     VARCHAR2(10 CHAR) NOT NULL,
  3.     street_address VARCHAR2(30 CHAR) NOT NULL,
  4.     city           VARCHAR2(30 CHAR) NOT NULL,
  5.     country        VARCHAR2(30 CHAR) NOT NULL
  6. );
  7.
  8. ALTER TABLE address ADD CONSTRAINT "employee-address_PK" PRIMARY
KEY ( address_id );
  9.
 10. CREATE TABLE department (
 11.     department_id   VARCHAR2(10 CHAR) NOT NULL,
 12.     department_name VARCHAR2(40 CHAR) NOT NULL
 13. );
 14.
 15. ALTER TABLE department ADD CONSTRAINT department_pk PRIMARY KEY (
department_id );
 16.
 17. CREATE TABLE employee (
 18.     employee_id   VARCHAR2(10 CHAR) NOT NULL,
 19.     employee_name VARCHAR2(40 CHAR) NOT NULL,
 20.     date_of_birth DATE NOT NULL,
 21.     contact       INTEGER NOT NULL,
 22.     department_id VARCHAR2(10 CHAR) NOT NULL,
 23.     role_id       VARCHAR2(10 CHAR) NOT NULL,
 24.     manager_id    VARCHAR2(10 CHAR) NOT NULL
 25. );
 26.
 27. ALTER TABLE employee ADD CONSTRAINT employee_pk PRIMARY KEY (
employee_id );
 28.
 29. CREATE TABLE employee_address (
 30.     employee_id VARCHAR2(10 CHAR) NOT NULL,
 31.     address_id  VARCHAR2(10 CHAR) NOT NULL
 32. );
 33.
 34. ALTER TABLE employee_address ADD CONSTRAINT voter_pk PRIMARY KEY
( employee_id,
 35.
address_id );
 36.
```

```
 37. CREATE TABLE employee_email (
 38.     employee_id   VARCHAR2(10 CHAR) NOT NULL,
 39.     email_address VARCHAR2(40 CHAR) NOT NULL
 40. );
 41.
 42. ALTER TABLE employee_email ADD CONSTRAINT candidate_pk PRIMARY
KEY ( email_address,
 43.
employee_id );
 44.
 45. CREATE TABLE job_history (
 46.     employee_id   VARCHAR2(10 CHAR) NOT NULL,
 47.     start_date    DATE NOT NULL,
 48.     end_date      DATE,
 49.     role_id       VARCHAR2(10 CHAR) NOT NULL,
 50.     department_id VARCHAR2(10 CHAR) NOT NULL
 51. );
 52.
 53. ALTER TABLE job_history ADD CONSTRAINT job_history_pk PRIMARY KEY
( employee_id,
 54.
start_date );
 55.
 56. CREATE TABLE jobs (
 57.     job_id     VARCHAR2(10 CHAR) NOT NULL,
 58.     job_title  VARCHAR2(40 CHAR) NOT NULL,
 59.     min_salary INTEGER NOT NULL,
 60.     max_salary INTEGER NOT NULL
 61. );
 62.
 63. ALTER TABLE jobs ADD CONSTRAINT roles_pk PRIMARY KEY ( job_id );
 64.
 65. CREATE TABLE manager_dept (
 66.     department_id VARCHAR2(10 CHAR) NOT NULL,
 67.     manager_id    VARCHAR2(10 CHAR) NOT NULL
 68. );
 69.
 70. ALTER TABLE manager_dept ADD CONSTRAINT manager_department_pk
PRIMARY KEY ( department_id );
 71.
 72. CREATE TABLE roles (
 73.     role_id       VARCHAR2(10 CHAR) NOT NULL,
 74.     role_name     VARCHAR2(40 CHAR) NOT NULL,
 75.     salary        INTEGER NOT NULL,
 76.     department_id VARCHAR2(10 CHAR) NOT NULL,
 77.     job_id        VARCHAR2(10 CHAR) NOT NULL
```

```
 78. );
 79.
 80. ALTER TABLE roles ADD CONSTRAINT roles_pkv2 PRIMARY KEY ( role_id
);
 81.
 82. CREATE TABLE voter_detail (
 83.     voter_id     VARCHAR2(10 CHAR) NOT NULL,
 84.     voting_year  INTEGER NOT NULL,
 85.     voting_month VARCHAR2(30 CHAR) NOT NULL,
 86.     candidate_id VARCHAR2(10 CHAR) NOT NULL
 87. );
 88.
 89. ALTER TABLE voter_detail
 90.     ADD CONSTRAINT voter_detail_pk PRIMARY KEY ( voter_id,
 91.                                                  voting_year,
 92.                                                  voting_month );
 93.
 94. ALTER TABLE employee_address
 95.     ADD CONSTRAINT employee_address_address_fk FOREIGN KEY (
address_id )
 96.         REFERENCES address ( address_id );
 97.
 98. ALTER TABLE employee_address
 99.     ADD CONSTRAINT employee_address_employee_fk FOREIGN KEY (
employee_id )
100.         REFERENCES employee ( employee_id );
101.
102. ALTER TABLE employee
103.     ADD CONSTRAINT employee_department_fk FOREIGN KEY (
department_id )
104.         REFERENCES department ( department_id );
105.
106. ALTER TABLE employee_email
107.     ADD CONSTRAINT employee_email_employee_fk FOREIGN KEY (
employee_id )
108.         REFERENCES employee ( employee_id );
109.
110. ALTER TABLE employee
111.     ADD CONSTRAINT employee_employee_fk FOREIGN KEY ( role_id )
112.         REFERENCES roles ( role_id );
113.
114. ALTER TABLE employee
115.     ADD CONSTRAINT employee_employee_fkv1 FOREIGN KEY (
manager_id )
116.         REFERENCES employee ( employee_id );
117.
```

```
118. ALTER TABLE job_history
119.     ADD CONSTRAINT job_history_department_fk FOREIGN KEY (
department_id )
120.         REFERENCES department ( department_id );
121.
122. ALTER TABLE job_history
123.     ADD CONSTRAINT job_history_employee_fk FOREIGN KEY (
employee_id )
124.         REFERENCES employee ( employee_id );
125.
126. ALTER TABLE job_history
127.     ADD CONSTRAINT job_history_roles_fk FOREIGN KEY ( role_id )
128.         REFERENCES roles ( role_id );
129.
130. ALTER TABLE manager_dept
131.     ADD CONSTRAINT manager_dept_department_fk FOREIGN KEY (
department_id )
132.         REFERENCES department ( department_id );
133.
134. ALTER TABLE manager_dept
135.     ADD CONSTRAINT manager_dept_employee_fk FOREIGN KEY (
manager_id )
136.         REFERENCES employee ( employee_id );
137.
138. ALTER TABLE roles
139.     ADD CONSTRAINT roles_jobs_fk FOREIGN KEY ( job_id )
140.         REFERENCES jobs ( job_id );
141.
142. ALTER TABLE roles
143.     ADD CONSTRAINT rolesv1_department_fk FOREIGN KEY (
department_id )
144.         REFERENCES department ( department_id );
145.
146. ALTER TABLE voter_detail
147.     ADD CONSTRAINT voter_detail_employee_fk FOREIGN KEY (
voter_id )
148.         REFERENCES employee ( employee_id );
149.
150. ALTER TABLE voter_detail
151.     ADD CONSTRAINT voter_detail_employee_fkv1 FOREIGN KEY (
candidate_id )
152.         REFERENCES employee ( employee_id );
153.
```

After the generation of DDL script we then paste it in the SQL developer and run the script below is the image of the process.

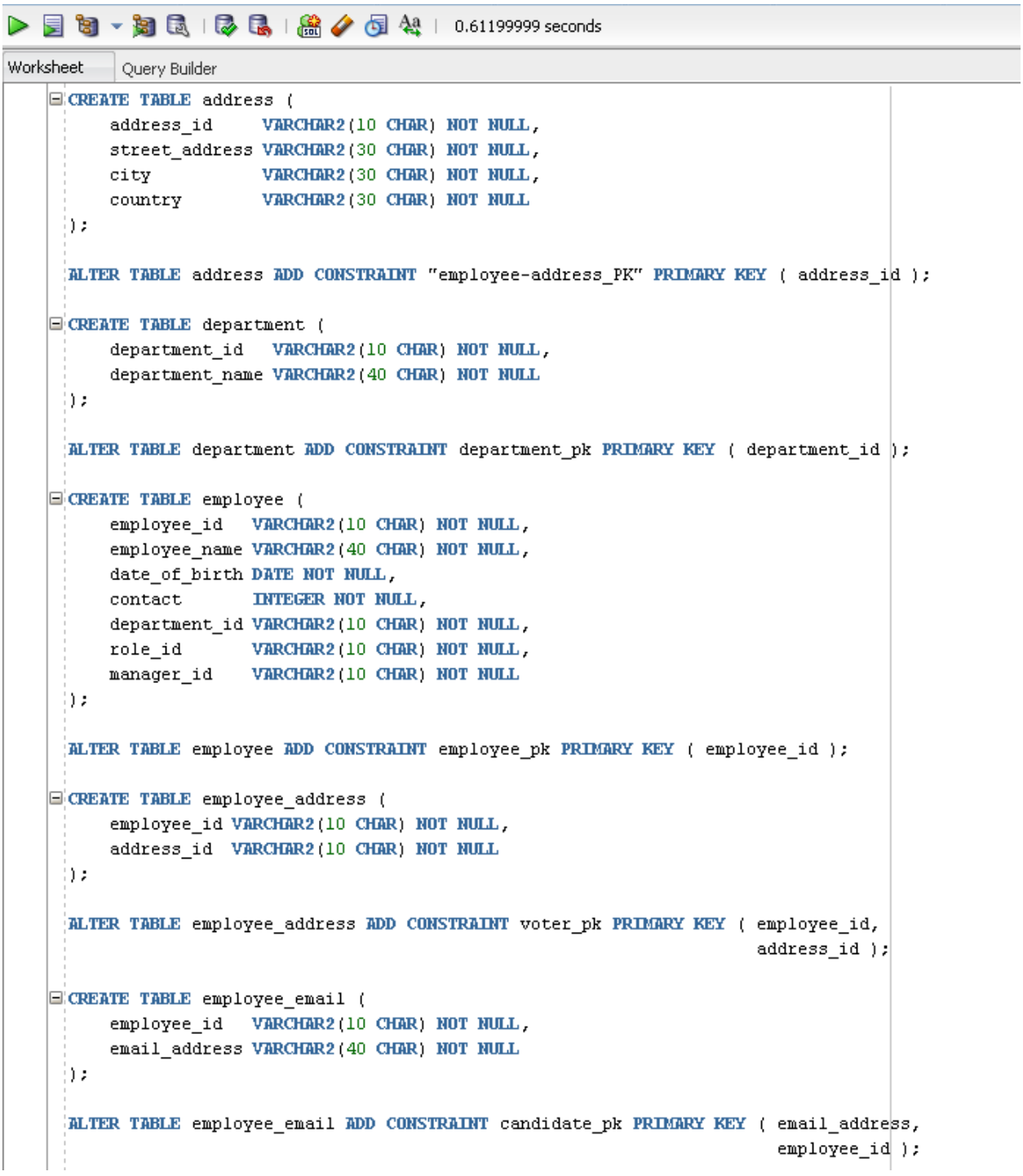


Figure 13: DDL script in SQL developer

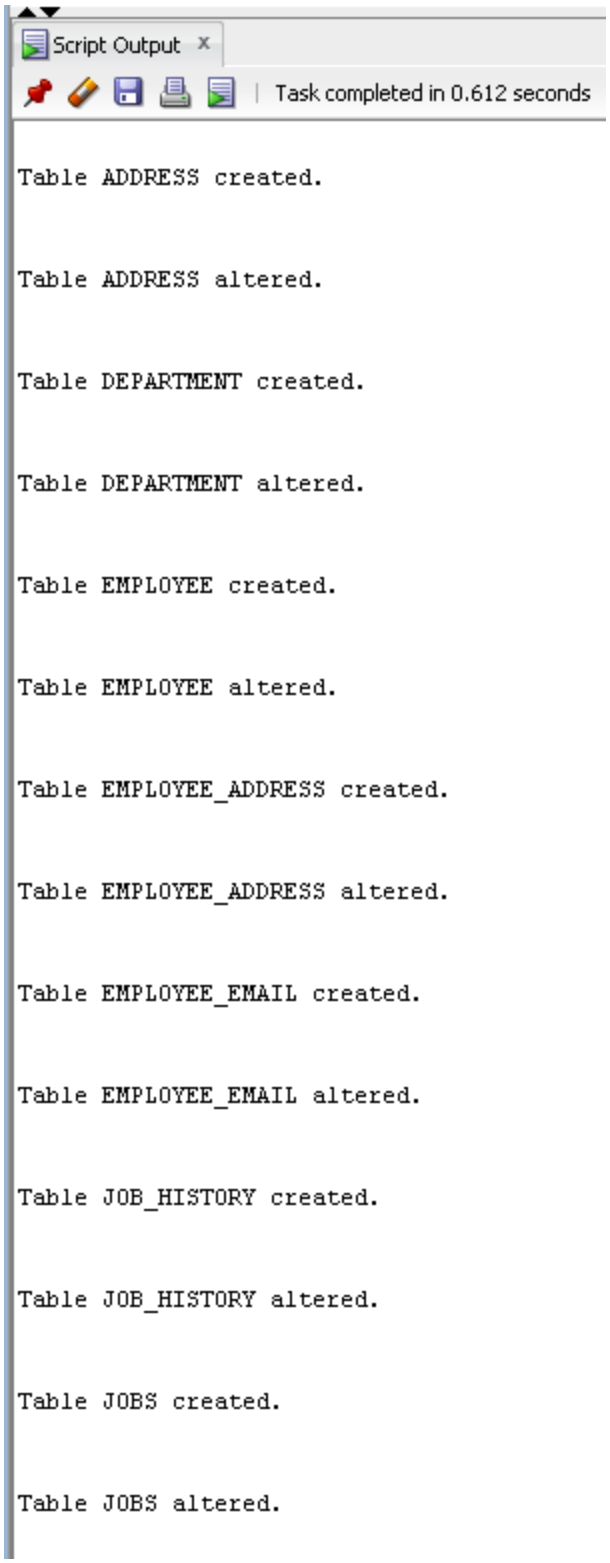


Figure 14: Table creation

```
Table JOBS altered.

Table MANAGER_DEPT created.

Table MANAGER_DEPT altered.

Table ROLES created.

Table ROLES altered.

Table VOTER_DETAIL created.

Table VOTER_DETAIL altered.

Table EMPLOYEE_ADDRESS altered.

Table EMPLOYEE_ADDRESS altered.

Table EMPLOYEE altered.

Table EMPLOYEE_EMAIL altered.

Table EMPLOYEE altered.

Table EMPLOYEE altered.
```

Figure 15: Table creation and alter.

```
Table EMPLOYEE_ADDRESS altered.

Table EMPLOYEE altered.

Table EMPLOYEE_EMAIL altered.

Table EMPLOYEE altered.

Table EMPLOYEE altered.

Table JOB_HISTORY altered.

Table JOB_HISTORY altered.

Table JOB_HISTORY altered.

Table MANAGER_DEPT altered.

Table MANAGER_DEPT altered.

Table ROLES altered.

Table ROLES altered.

Table VOTER_DETAIL altered.

Table VOTER_DETAIL altered.
```

Figure 16: Table altered.

9. Insert Statement

9.1. Address Table

```
INSERT ALL
INTO address (address_id, street_address, city, country) VALUES ('A001', 'New Baneshwor', 'Kathmandu', 'Nepal')
INTO address (address_id, street_address, city, country) VALUES ('A002', 'Begnas Lake', 'Pokhara', 'Nepal')
INTO address (address_id, street_address, city, country) VALUES ('A003', 'Koshi', 'Itahari', 'Nepal')
INTO address (address_id, street_address, city, country) VALUES ('A004', 'Sinamangal', 'Kathmandu', 'Nepal')
INTO address (address_id, street_address, city, country) VALUES ('A005', 'Naxal', 'Kathmandu', 'Nepal')
INTO address (address_id, street_address, city, country) VALUES ('A006', 'Jawalakhel', 'Lalitpur', 'Nepal')
INTO address (address_id, street_address, city, country) VALUES ('A007', 'Banglamukhi', 'Lalitpur', 'Nepal')
INTO address (address_id, street_address, city, country) VALUES ('A008', 'Basantapur', 'Kathmandu', 'Nepal')
INTO address (address_id, street_address, city, country) VALUES ('A009', 'Bhaisepati', 'Lalitpur', 'Nepal')
INTO address (address_id, street_address, city, country) VALUES ('A010', 'Basantapur', 'Kathmandu', 'Nepal')
SELECT * FROM DUAL;
COMMIT;
```

```
Script Output
Task completed in 0.054 seconds
10 rows inserted.


Commit complete.
```

Figure 17: Insert statement address

9.2. Department Table

```
INSERT ALL
INTO department (department_id, department_name) VALUES ('D001', 'Sales')
INTO department (department_id, department_name) VALUES ('D002', 'Information Technology')
INTO department (department_id, department_name) VALUES ('D003', 'Human Resources')
INTO department (department_id, department_name) VALUES ('D004', 'Legal')
INTO department (department_id, department_name) VALUES ('D005', 'Customer Service')
INTO department (department_id, department_name) VALUES ('D006', 'Operations')
INTO department (department_id, department_name) VALUES ('D007', 'Research and Development')
INTO department (department_id, department_name) VALUES ('D008', 'Public Relation')
INTO department (department_id, department_name) VALUES ('D009', 'Administration')
INTO department (department_id, department_name) VALUES ('D010', 'Quality Assurance')
SELECT * FROM DUAL;
COMMIT;
```

```
Script Output
Task completed in 0.048 seconds
10 rows inserted.


Commit complete.
```

Figure 18: Insert statement department

9.3. Jobs table

```
INSERT ALL
INTO jobs (job_id, job_title, min_salary, max_salary) VALUES ('1', 'Account Executive', 50000, 110000)
INTO jobs (job_id, job_title, min_salary, max_salary) VALUES ('2', 'Network Engineer', 40000, 220000)
INTO jobs (job_id, job_title, min_salary, max_salary) VALUES ('3', 'Manufacturing Engineer', 30000, 190000)
INTO jobs (job_id, job_title, min_salary, max_salary) VALUES ('4', 'Attorney', 100000, 200000)
INTO jobs (job_id, job_title, min_salary, max_salary) VALUES ('5', 'Representative', 15000, 90000)
INTO jobs (job_id, job_title, min_salary, max_salary) VALUES ('6', ' Operations Manager', 9000, 190000)
INTO jobs (job_id, job_title, min_salary, max_salary) VALUES ('7', 'Scientist', 10000, 50000)
INTO jobs (job_id, job_title, min_salary, max_salary) VALUES ('8', 'Specialist', 11000, 30000)
INTO jobs (job_id, job_title, min_salary, max_salary) VALUES ('9', 'Coordinator', 12000, 70000)
INTO jobs (job_id, job_title, min_salary, max_salary) VALUES ('10', 'Tester', 8000, 80000)
SELECT * FROM DUAL;
COMMIT;
```

Script Output | Task completed in 0.048 seconds

```
10 rows inserted.


Commit complete.
```

Figure 19: Insert statement jobs

9.4. Roles Table

```
INSERT ALL
INTO roles (role_id, role_name, salary, department_id, job_id) VALUES ('R001', 'Prospecting', 95000, 'D001', '2')
INTO roles (role_id, role_name, salary, department_id, job_id) VALUES ('R002', 'Network Administrator', 150000, 'D002', '4')
INTO roles (role_id, role_name, salary, department_id, job_id) VALUES ('R003', 'Quality Control', 110000, 'D003', '3')
INTO roles (role_id, role_name, salary, department_id, job_id) VALUES ('R004', 'Legal Counsel', 160000, 'D004', '5')
INTO roles (role_id, role_name, salary, department_id, job_id) VALUES ('R005', 'Issue Resolution', 90000, 'D005', '1')
INTO roles (role_id, role_name, salary, department_id, job_id) VALUES ('R006', 'Manage', 95000, 'D001', '2')
INTO roles (role_id, role_name, salary, department_id, job_id) VALUES ('R007', 'Innovate', 150000, 'D002', '4')
INTO roles (role_id, role_name, salary, department_id, job_id) VALUES ('R008', 'Communication', 110000, 'D003', '3')
INTO roles (role_id, role_name, salary, department_id, job_id) VALUES ('R009', 'Verify', 160000, 'D004', '5')
INTO roles (role_id, role_name, salary, department_id, job_id) VALUES ('R010', 'Repair', 90000, 'D005', '1')
SELECT * FROM DUAL;
COMMIT;
```

Script Output | Task completed in 0.049 seconds

```
10 rows inserted.


Commit complete.
```

Figure 20: Roles

9.5. Employee table

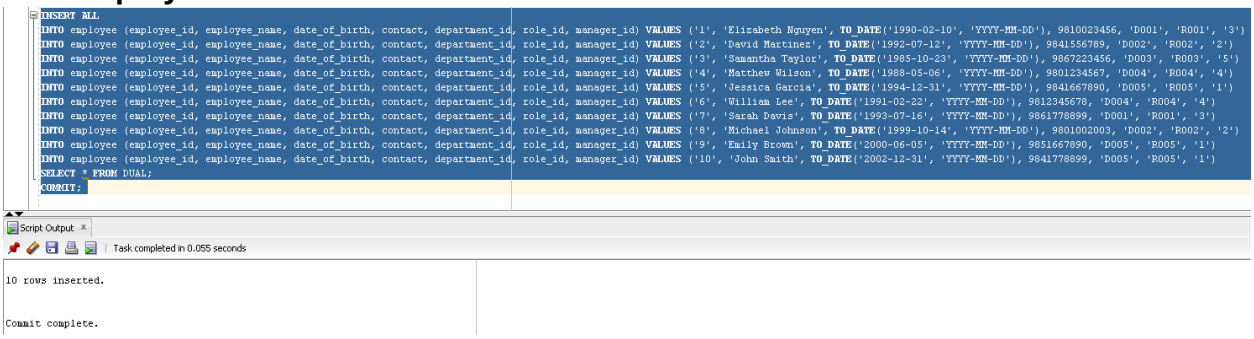


Figure 21: Insert statement employee

9.6. Job history table

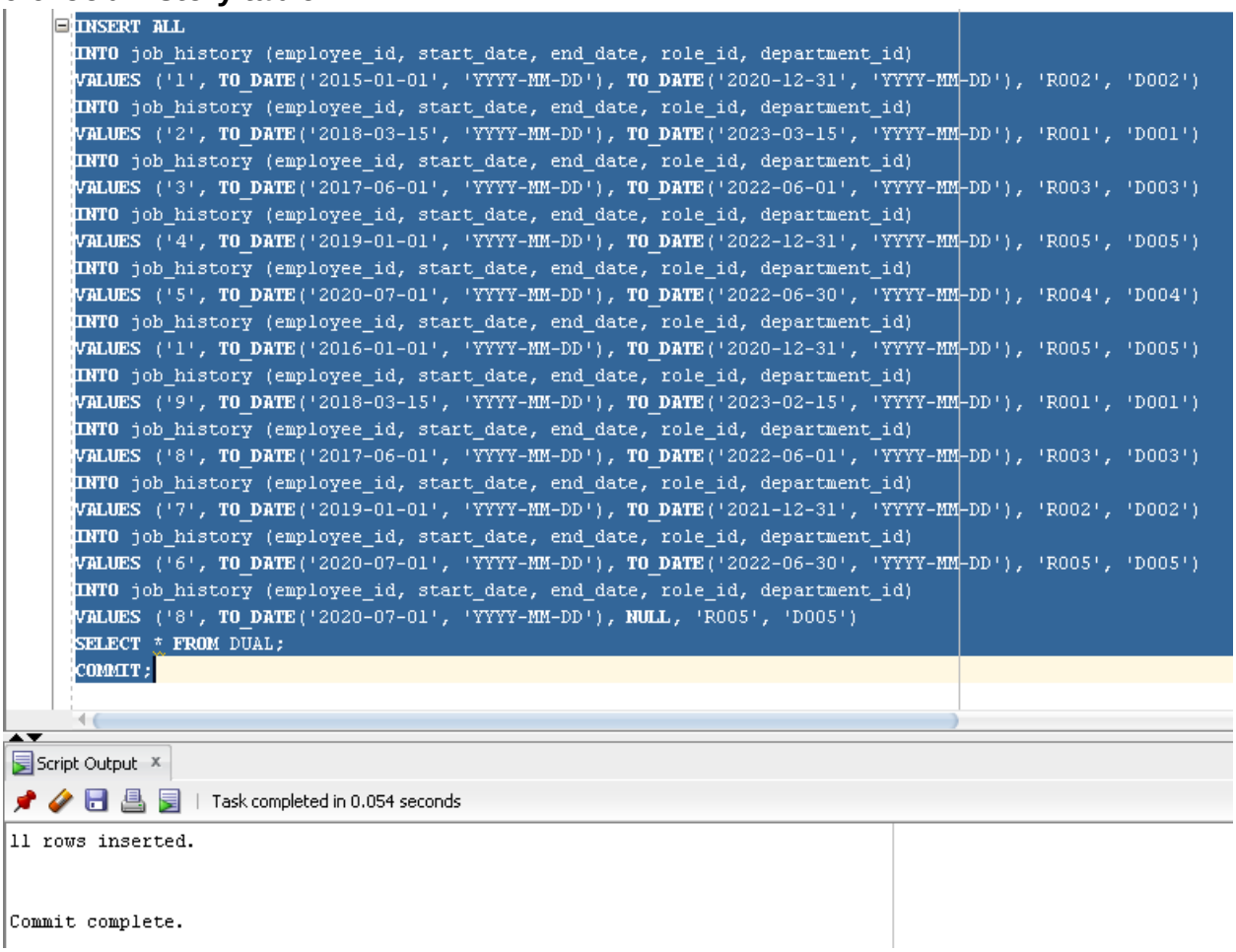


Figure 22: Job_History

9.7. Voter Detail table

```
INSERT ALL
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('1', 2022, 'March', '2')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('2', 2022, 'March', '3')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('3', 2021, 'February', '5')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('4', 2023, 'January', '1')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('5', 2023, 'April', '4')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('6', 2022,'January', '2')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('7', 2020, 'March', '3')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('8', 2021, 'August', '5')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('9', 2022, 'July', '1')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('9', 2023, 'June', '4')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('8', 2021, 'April', '4')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('7', 2020, 'May', '4')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('6', 2022, 'March', '4')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('3', 2022, 'March', '2')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('6', 2022, 'April', '3')
INTO voter_detail (voter_id, voting_year, voting_month, candidate_id)
VALUES ('8', 2021, 'February', '2')
SELECT * FROM DUAL;
COMMIT;
```

Script Output | Task completed in 0.047 seconds

```
16 rows inserted.


Commit complete.
```

Figure 23: Insert statement Voter detail.

9.8. Manager department table

```
INSERT ALL
INTO manager_dept (department_id, manager_id) VALUES ('D001', '3')
INTO manager_dept (department_id, manager_id) VALUES ('D002', '2')
INTO manager_dept (department_id, manager_id) VALUES ('D003', '5')
INTO manager_dept (department_id, manager_id) VALUES ('D004', '4')
INTO manager_dept (department_id, manager_id) VALUES ('D005', '1')
INTO manager_dept (department_id, manager_id) VALUES ('D006', '8')
INTO manager_dept (department_id, manager_id) VALUES ('D007', '7')
INTO manager_dept (department_id, manager_id) VALUES ('D008', '10')
INTO manager_dept (department_id, manager_id) VALUES ('D009', '9')
INTO manager_dept (department_id, manager_id) VALUES ('D010', '6')
SELECT * FROM DUAL;
COMMIT;
```

Script Output | Task completed in 0.047 seconds

```
10 rows inserted.


Commit complete.
```

Figure 24: Insert statement manager_id

9.9. Employee Email table

```
INSERT ALL
INTO employee_email (employee_id, email_address) VALUES ('1', 'elizabethnguyen@email.com')
INTO employee_email (employee_id, email_address) VALUES ('2', 'davidmartinez@email.com')
INTO employee_email (employee_id, email_address) VALUES ('3', 'samanthataylor@email.com')
INTO employee_email (employee_id, email_address) VALUES ('4', 'matthewwilson@email.com')
INTO employee_email (employee_id, email_address) VALUES ('5', 'jessicagarcia@email.com')
INTO employee_email (employee_id, email_address) VALUES ('6', ' williamlee@email.com')
INTO employee_email (employee_id, email_address) VALUES ('7', 'sarahdavis@email.com')
INTO employee_email (employee_id, email_address) VALUES ('8', 'michaeljohnson@email.com')
INTO employee_email (employee_id, email_address) VALUES ('9', 'emilybrown@email.com')
INTO employee_email (employee_id, email_address) VALUES ('10', ' johnsmith@email.com')
INTO employee_email (employee_id, email_address) VALUES ('1', ' eliz@email.com')
INTO employee_email (employee_id, email_address) VALUES ('6', ' lee@email.com')
SELECT * FROM DUAL;
COMMIT;
```

Script Output

Task completed in 0.062 seconds

```
12 rows inserted.


Commit complete.
```

Figure 25: employee_email

9.10 Employee Address

```
INSERT ALL
INTO employee_address (employee_id, address_id) VALUES ('1', 'A001')
INTO employee_address (employee_id, address_id) VALUES ('2', 'A002')
INTO employee_address (employee_id, address_id) VALUES ('3', 'A003')
INTO employee_address (employee_id, address_id) VALUES ('4', 'A004')
INTO employee_address (employee_id, address_id) VALUES ('5', 'A005')
INTO employee_address (employee_id, address_id) VALUES ('6', 'A006')
INTO employee_address (employee_id, address_id) VALUES ('7', 'A007')
INTO employee_address (employee_id, address_id) VALUES ('8', 'A008')
INTO employee_address (employee_id, address_id) VALUES ('9', 'A009')
INTO employee_address (employee_id, address_id) VALUES ('10', 'A010')
INTO employee_address (employee_id, address_id) VALUES ('3', 'A005')
INTO employee_address (employee_id, address_id) VALUES ('2', 'A009')
SELECT * FROM DUAL;
COMMIT;
```

Script Output

Task completed in 0.06 seconds

```
12 rows inserted.


Commit complete.
```

Figure 26: Insert statement employee address.

10 Select Statement

10.1. Employee Tbale

```
select * from employee;
```

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.005 seconds

	EMPLOYEE_ID	EMPLOYEE_NAME	DATE_OF_BIRTH	CONTACT	DEPARTMENT_ID	ROLE_ID	MANAGER_ID
1	1	Elizabeth Nguyen	10-FEB-90	9810023456	D001	R001	3
2	2	David Martinez	12-JUL-92	9841556789	D002	R002	2
3	3	Samantha Taylor	23-OCT-85	9867223456	D003	R003	5
4	4	Matthew Wilson	06-MAY-88	9801234567	D004	R004	4
5	5	Jessica Garcia	31-DEC-94	9841667890	D005	R005	1
6	6	William Lee	22-FEB-91	9812345678	D004	R004	4
7	7	Sarah Davis	16-JUL-93	9861778899	D001	R001	3
8	8	Michael Johnson	14-OCT-99	9801002003	D002	R002	2
9	9	Emily Brown	05-JUN-00	9851667890	D005	R005	1
10	10	John Smith	31-DEC-02	9841778899	D005	R005	1

10.2. Department table

```
select * from department;
```

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.004 seconds

	DEPARTMENT_ID	DEPARTMENT_NAME
1	D001	Sales
2	D002	Information Technology
3	D003	Human Resources
4	D004	Legal
5	D005	Customer Service
6	D006	Operations
7	D007	Research and Development
8	D008	Public Relation
9	D009	Administration
10	D010	Quality Assurance

10.3. Roles table

```
select * from roles;
```

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.003 seconds

	ROLE_ID	ROLE_NAME	SALARY	DEPARTMENT_ID	JOB_ID
1	R001	Prospecting	95000	D001	2
2	R002	Network Administrator	150000	D002	4
3	R003	Quality Control	110000	D003	3
4	R004	Legal Counsel	160000	D004	5
5	R005	Issue Resolution	90000	D005	1
6	R006	Manage	95000	D001	2
7	R007	Innovate	150000	D002	4
8	R008	Communication	110000	D003	3
9	R009	Verify	160000	D004	5
10	R010	Repair	90000	D005	1

10.4. Jobs table

```
select * from jobs;
```

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.002 seconds

	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	1	Account Executive	50000	110000
2	2	Network Engineer	40000	220000
3	3	Manufacturing Engineer	30000	190000
4	4	Attorney	100000	200000
5	5	Representative	15000	90000
6	6	Operations Manager	9000	190000
7	7	Scientist	10000	50000
8	8	Specialist	11000	30000
9	9	Coordinator	12000	70000
10	10	Tester	8000	80000

10.5. Job History table

```
select * from job_history;
```

Script Output | Query Result

SQL | All Rows Fetched: 11 in 0.002 seconds

	EMPLOYEE_ID	START_DATE	END_DATE	ROLE_ID	DEPARTMENT_ID
1	1	01-JAN-15	31-DEC-20	R002	D002
2	2	15-MAR-18	15-MAR-23	R001	D001
3	3	01-JUN-17	01-JUN-22	R003	D003
4	4	01-JAN-19	31-DEC-22	R005	D005
5	5	01-JUL-20	30-JUN-22	R004	D004
6	1	01-JAN-16	31-DEC-20	R005	D005
7	9	15-MAR-18	15-FEB-23	R001	D001
8	8	01-JUN-17	01-JUN-22	R003	D003
9	7	01-JAN-19	31-DEC-21	R002	D002
10	6	01-JUL-20	30-JUN-22	R005	D005
11	8	01-JUL-20	(null)	R005	D005

10.6. Voter Detail

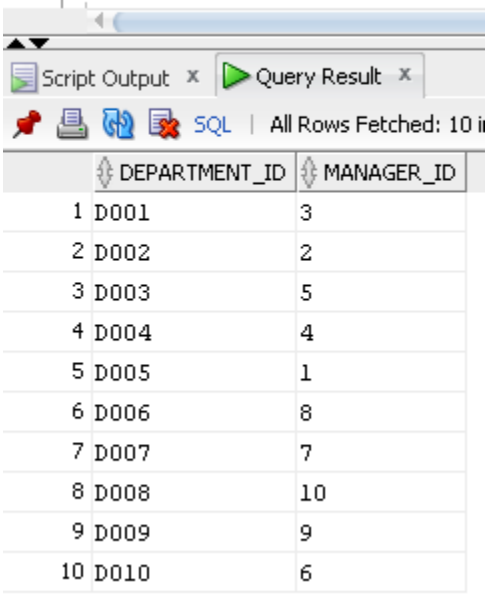
```
select * from voter_detail;
```

Script Output | Query Result | SQL | All Rows Fetched: 16 in 0.003 seconds

	VOTER_ID	VOTING_YEAR	VOTING_MONTH	CANDIDATE_ID
1	1	2022	March	2
2	2	2022	March	3
3	3	2021	February	5
4	4	2023	January	1
5	5	2023	April	4
6	6	2022	January	2
7	7	2020	March	3
8	8	2021	August	5
9	9	2022	July	1
10	9	2023	June	4
11	8	2021	April	4
12	7	2020	May	4
13	6	2022	March	4
14	3	2022	March	2
15	6	2022	April	3
16	8	2021	February	2

10.7. manager department

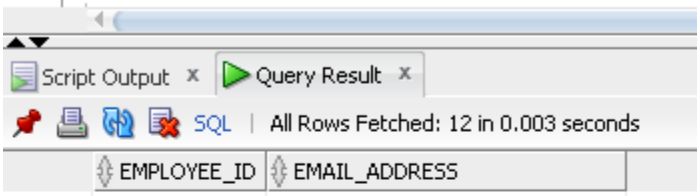
```
select * from manager_dept;
```



	DEPARTMENT_ID	MANAGER_ID
1	D001	3
2	D002	2
3	D003	5
4	D004	4
5	D005	1
6	D006	8
7	D007	7
8	D008	10
9	D009	9
10	D010	6

10.8. employee email

```
select * from employee_email;
```



	EMPLOYEE_ID	EMAIL_ADDRESS
1	1	eliz@email.com
2	10	johnsmith@email.com
3	6	lee@email.com
4	6	williamlee@email.com
5	2	davidmartinez@email.com
6	1	elizabethnguyen@email.com
7	9	emilybrown@email.com
8	5	jessicagarcia@email.com
9	4	matthewwilson@email.com
10	8	michaeljohnson@email.com
11	3	samanthataylor@email.com
12	7	sarahdavis@email.com

10.9. Address

```
select * from address;
```

Script Output | Query Result | SQL | All Rows Fetched: 10 in 0.002 seconds

	ADDRESS_ID	STREET_ADDRESS	CITY	COUNTRY
1	A001	New Baneshwor	Kathmandu	Nepal
2	A002	Begnas Lake	Pokhara	Nepal
3	A003	Koshi	Itahari	Nepal
4	A004	Sinamangal	Kathmandu	Nepal
5	A005	Naxal	Kathmandu	Nepal
6	A006	Jawalakhel	Lalitpur	Nepal
7	A007	Banglamukhi	Lalitpur	Nepal
8	A008	Basantapur	Kathmandu	Nepal
9	A009	Bhaisepati	Lalitpur	Nepal
10	A010	Basantapur	Kathmandu	Nepal

10.10. employee address

```
select * from employee_address;
```

Script Output | Query Result | SQL | All Rows Fetched: 12 in 0.003 seconds

	EMPLOYEE_ID	ADDRESS_ID
1	1	A001
2	10	A010
3	2	A002
4	2	A009
5	3	A003
6	3	A005
7	4	A004
8	5	A005
9	6	A006
10	7	A007
11	8	A008
12	9	A009

11 Forms

11.1. Dashboard

Dashboard is one of the most important pages for this online voting system as it is the landing page and users can navigate to other pages through this page and get some important information of the voting results and count of employees and department. It also displays graph with total votes.

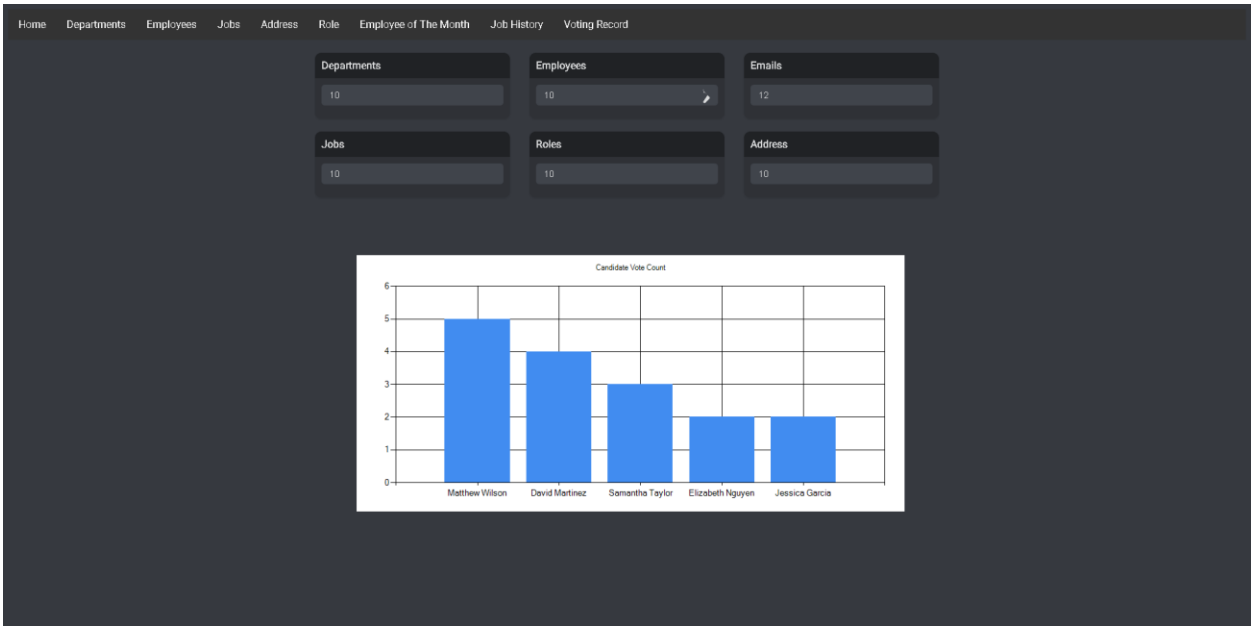


Figure 27: Dashboard

11.2. Complex form and Queries

The web forms are built using ASP.NET web application which uses a programming language C# which allows us to perform curd operations dynamically in a table The users can insert records in a table and save it also later edit and delete the records There is a total of 9 online forms that can be accessed by the users.

11.2.1. SQL Queries

This is the important part of the voting system we use different tables join them and display valuable results according to the requirements.

- **Voting record**

```
1. SELECT e1.employee_id , e1.employee_name, e1.date_of_birth,
2. e1.contact, vd.voting_year, vd.voting_month, e2.employee_name as
3. candidate_name, e2.department_id as candidate_department
4. FROM Employee e1
5. JOIN Voter_Detail vd ON e1.employee_id = vd.voter_id
6. JOIN Employee e2 ON vd.candidate_id = e2.employee_id
7. WHERE e1.employee_id = :employee
8.
```

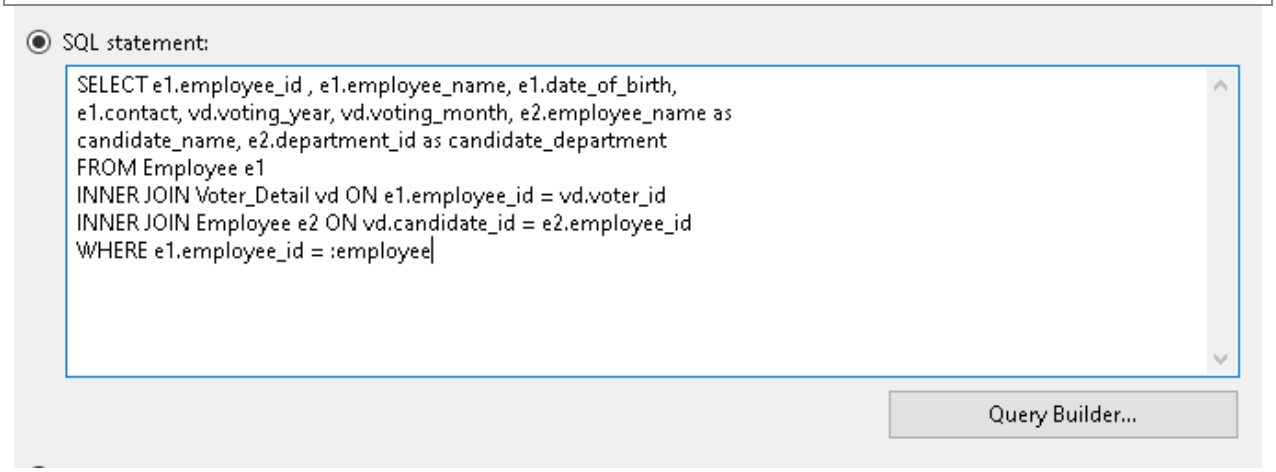


Figure 28: Voting Record

- **Job history**

```
 1. SELECT e.employee_id, e.employee_name, e.date_of_birth, e.contact,
 2.        jh.start_date AS job_start_date, jh.end_date AS
job_end_date,
 3.        d.department_name AS previous_department, r.role_name AS
previous_role
 4. FROM Employee e
 5. INNER JOIN Job_History jh ON e.employee_id = jh.employee_id
 6. INNER JOIN Department d ON jh.department_id = d.department_id
 7. INNER JOIN Roles r ON jh.role_id = r.role_id
 8. WHERE e.employee_id = :employee
 9. AND jh.end_date IS NOT NULL
10.
```

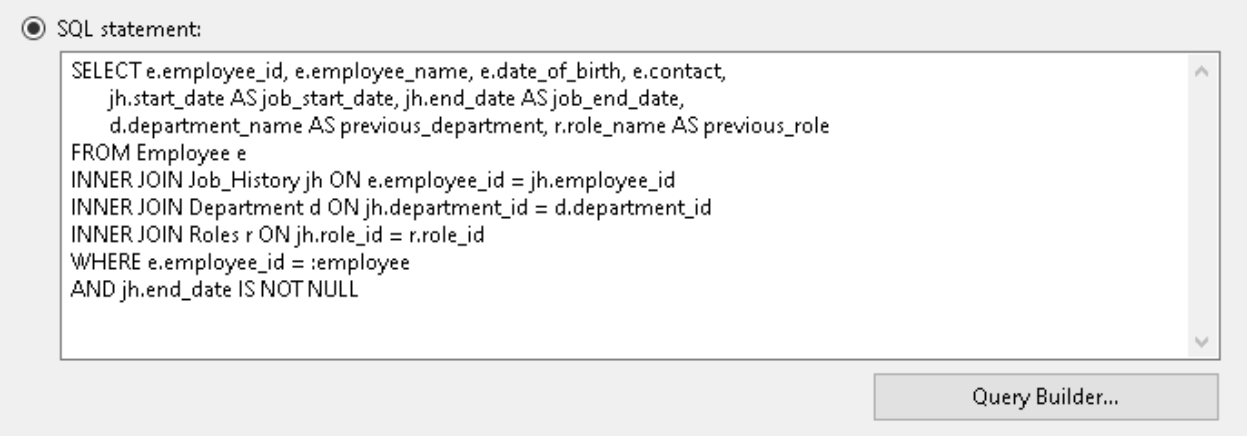


Figure 29: Job history

- **Top three candidate for specific month and year**

```
 1. SELECT v.candidate_id, e.employee_name, e.date_of_birth,
e.contact, v.vote_count
 2. FROM (
 3.   SELECT candidate_id, COUNT(*) AS vote_count
 4.   FROM voter_detail
 5.   WHERE voting_month = :month AND voting_year = :year
 6.   GROUP BY candidate_id
 7.   ORDER BY vote_count DESC
 8. ) v
 9. JOIN employee e ON e.employee_id = v.candidate_id
10. WHERE ROWNUM <= 3
11.
```

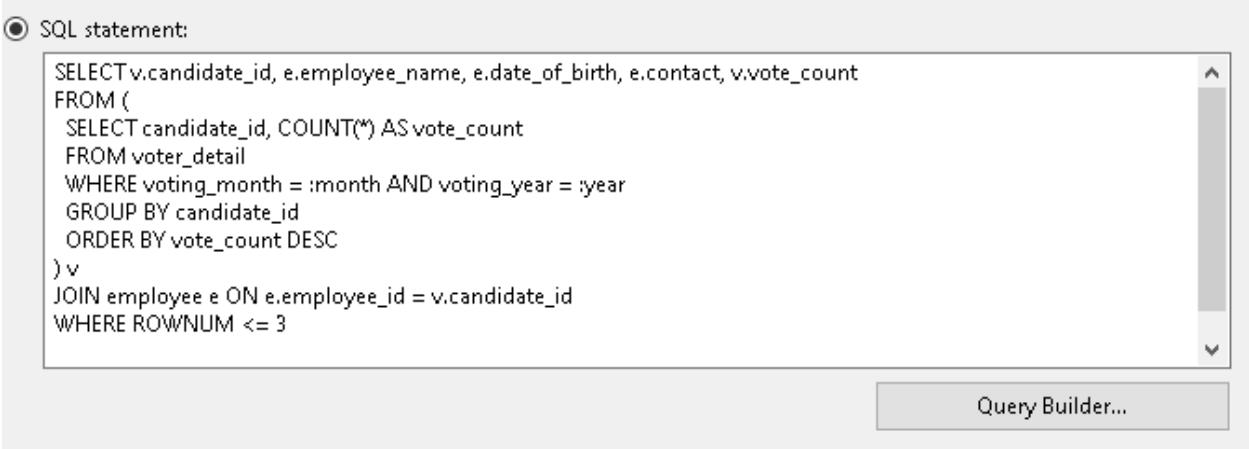


Figure 30: Top three candidate

11.2.2. Complex Forms

- **Employee of the month**

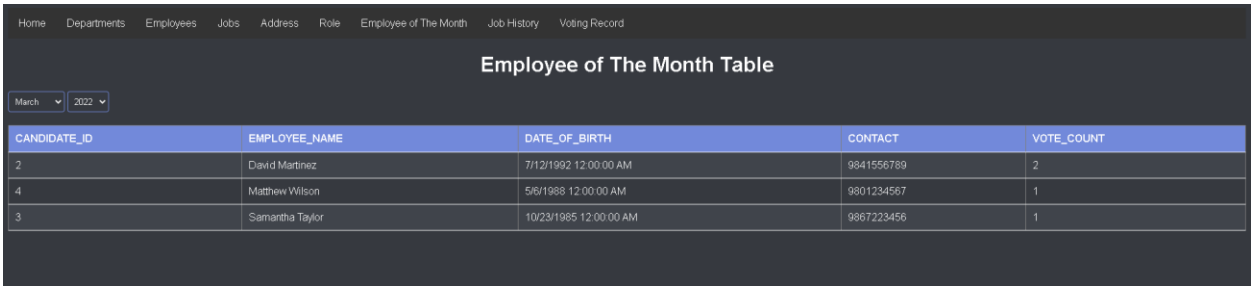


Figure 31: Complex form employee of the month

- **Job history table**

Figure 32: Complex form job history

- **Voting record table**

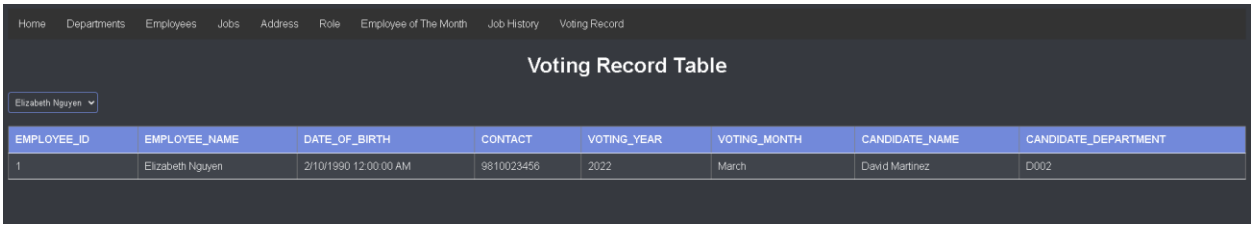


Figure 33: Complex form voting record

11.3. Simple Form

- **Department table**

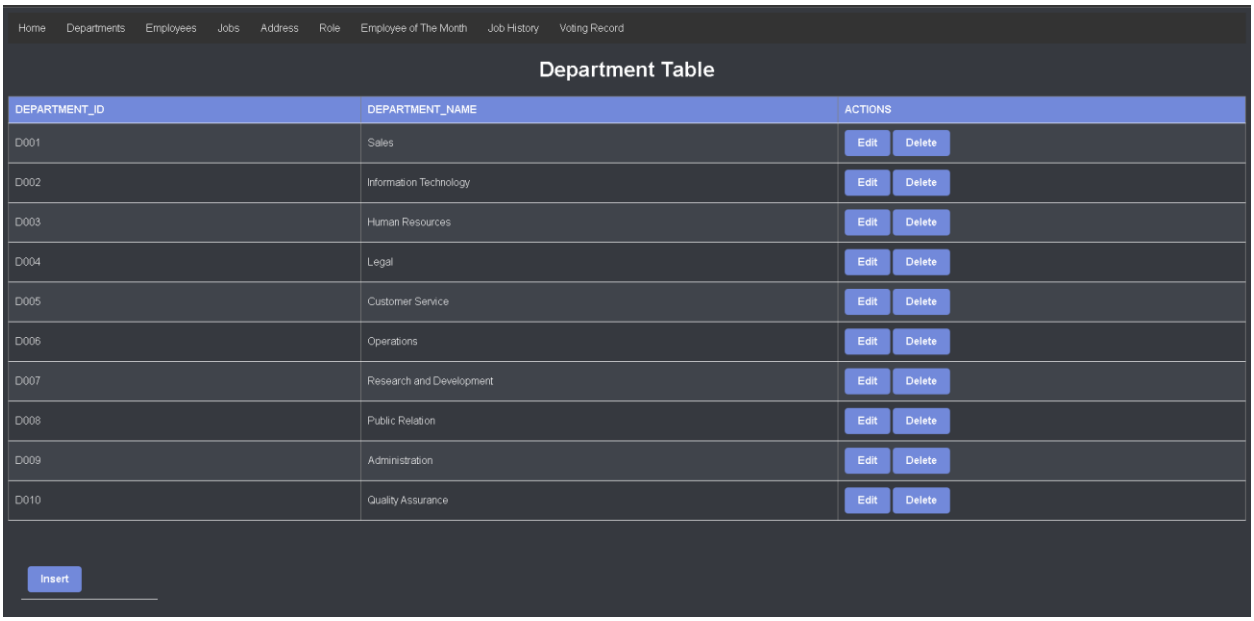


Figure 34: Simple department table

- **Address table**

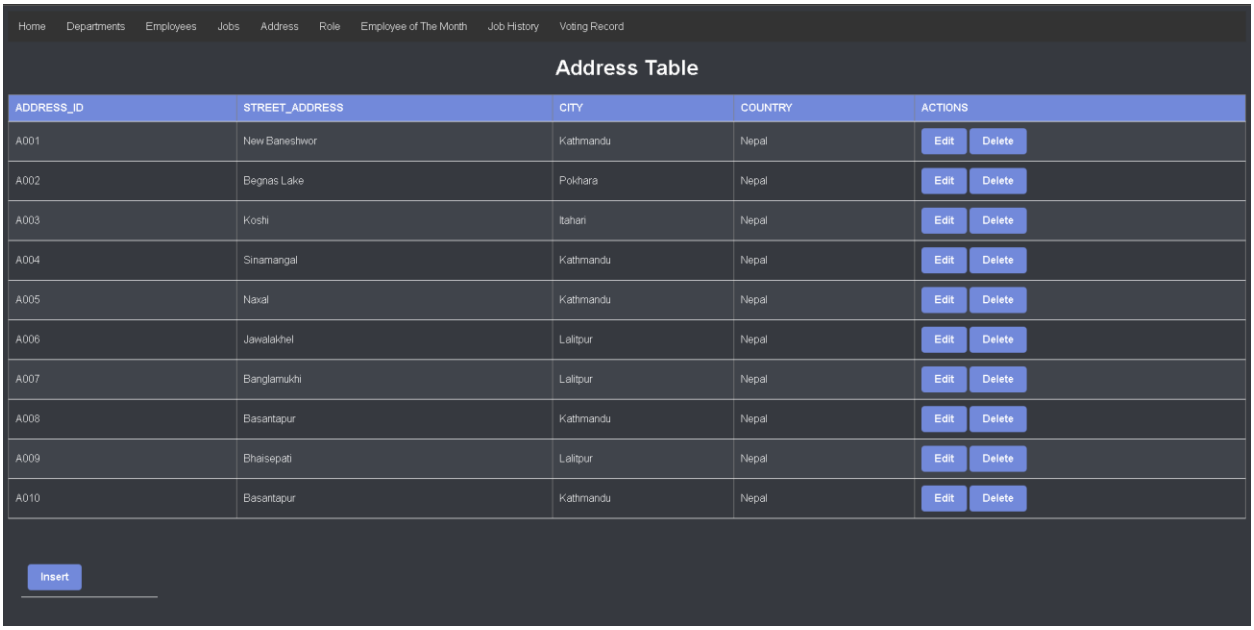


Figure 35: Simple form address table

- **Employee table**

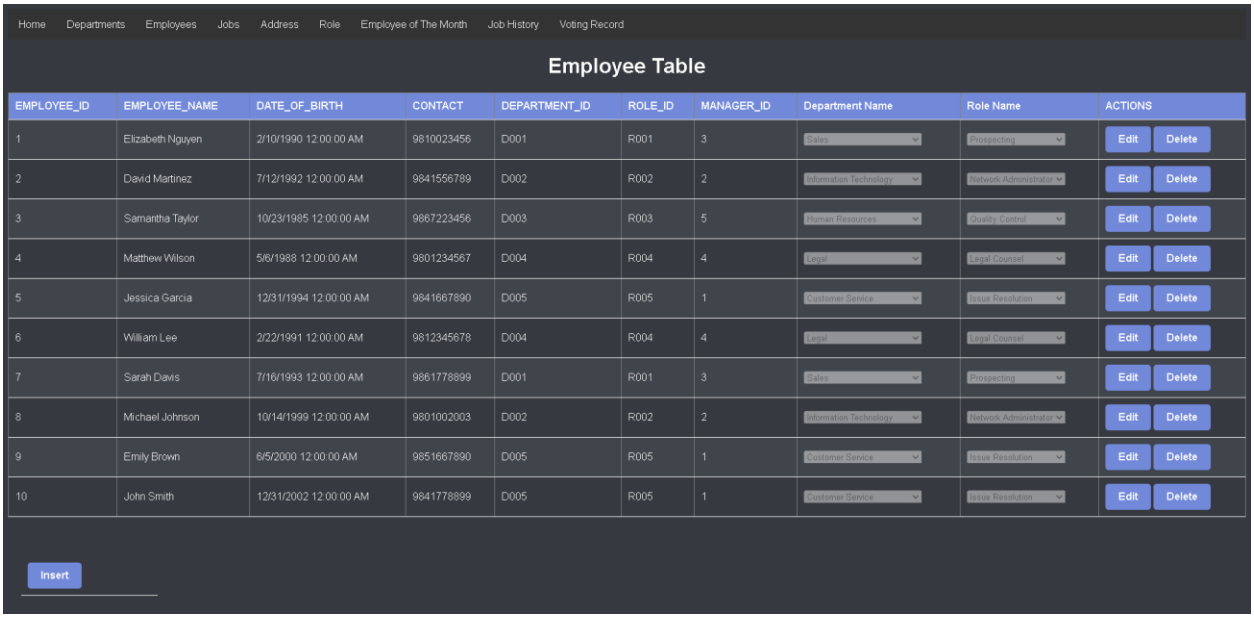


Figure 36: Simple form employee table

- **Jobs table**

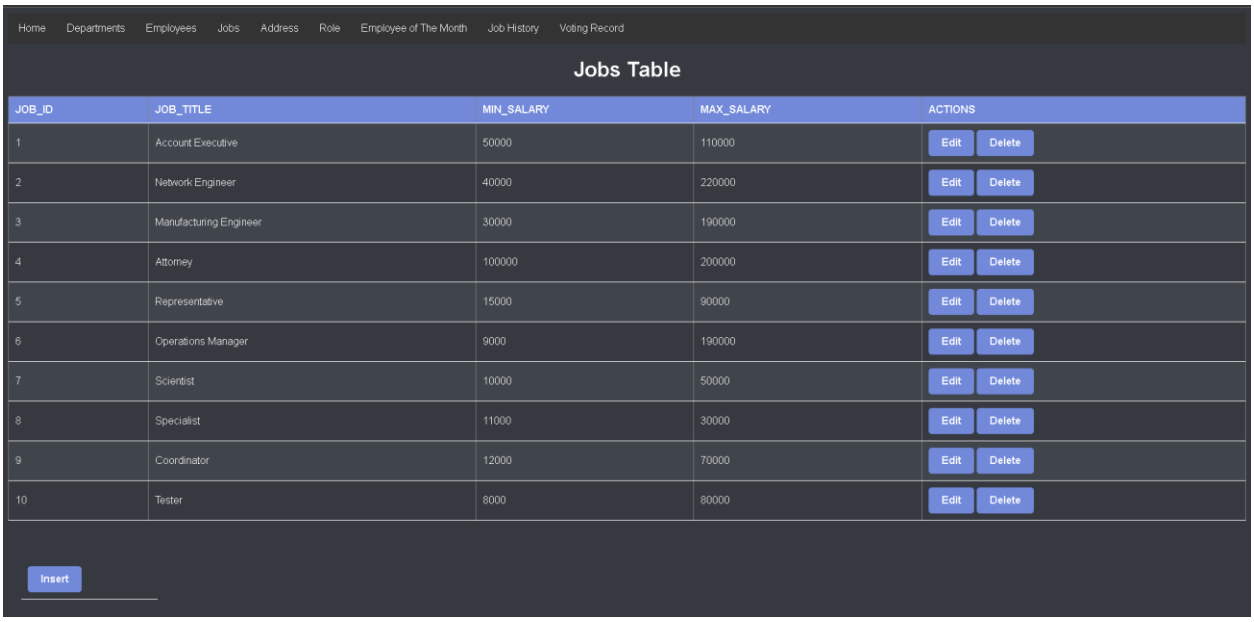


Figure 37: Jobs table

- **Role table**

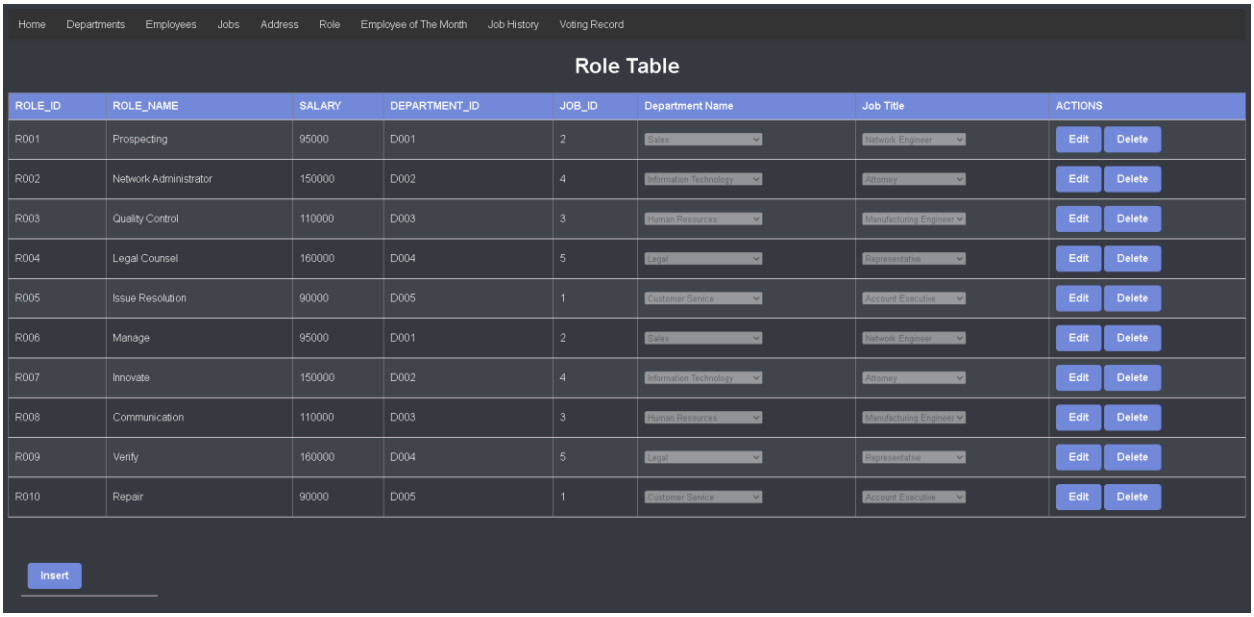


Figure 38: Role table

12. User Manual

The user manual is one of the most important parts for users trying to learn how to use the software or understand it or use if they encounter a problem before reaching out to customer service.

12.1 Introduction

The online voting system is designed for the users to view and get information about the organizations voting results where users can look at the voting results that is displayed in grid view or even bar chart. The users can do different crud operations on simple form on department, employee, address, role, and job tables. The software has 9 total forms each unique on its own. The user manual is divided in to three different sections. Navigation section, crud operations section and finally complex form.

12.1.1. Navigation bar

Navigation bar is one of the most important things of any software or web application which helps us to navigate thought different pages. The navigation bar for online voting system consists of 9 items the home button is highlighted because it is active nav link. Others are links for other pages. The nav bar is placed right at the top of the screen.

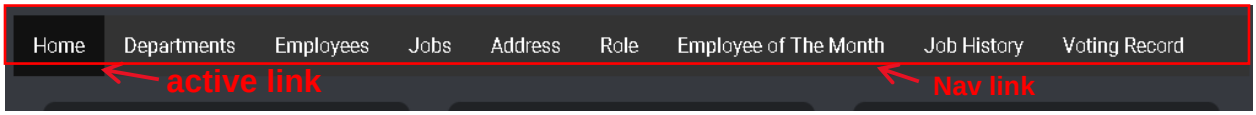


Figure 39: Nav

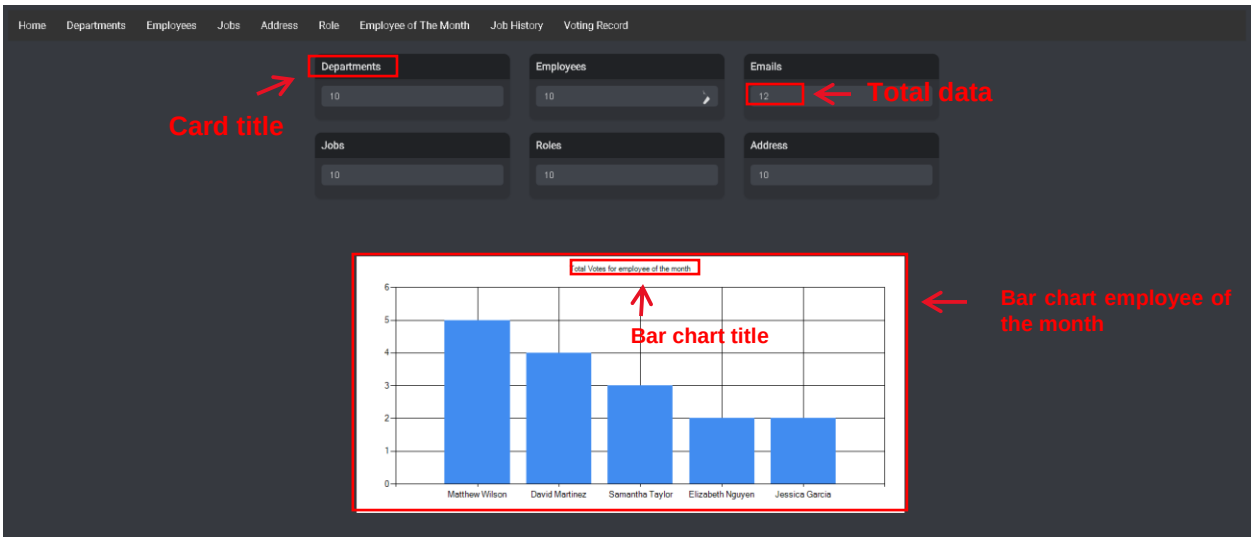


Figure 40: Dashboard

Now comes the dashboard page which has about 6 boxed which has its title and the total amount of data for that table in that database. There are 10 department, 10 employee, 12 emails, 10 jobs, 10 roles and 10 address.

12.1.2. Simple form
Employee Form

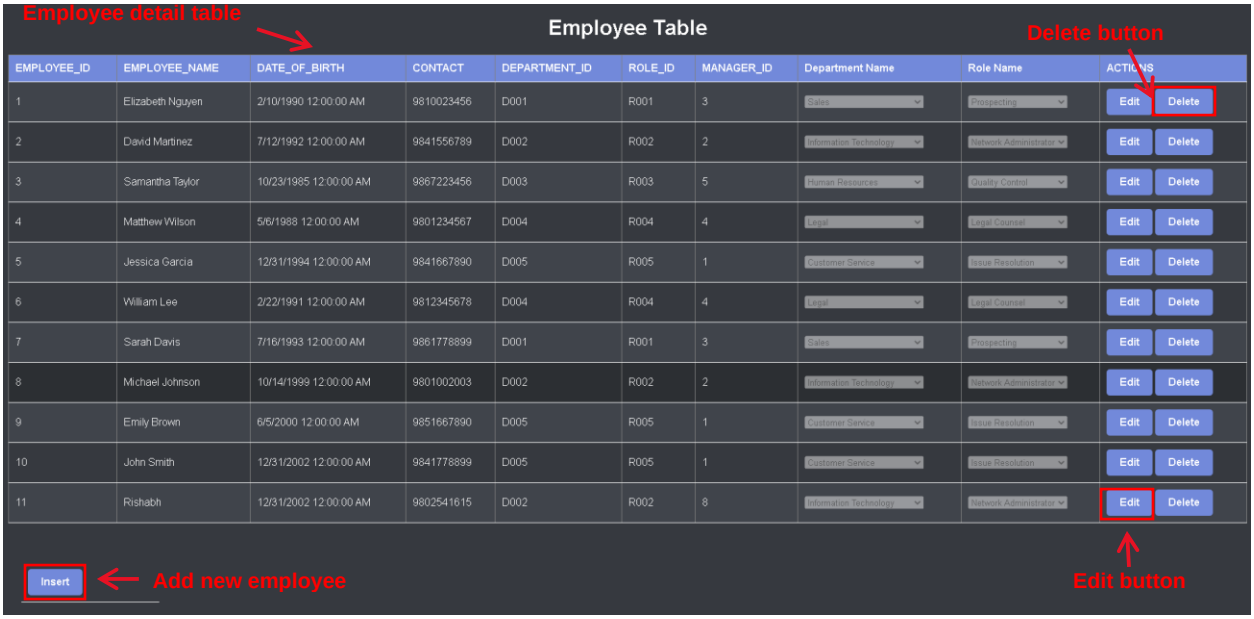


Figure 41: Employee

The simple form contains four main features create, update, delete, read also known as (CRUD). The delete button deletes the selected record the edit button edits the selected record when we enter value in the text field the insert button then opens a small form where we can enter new employee details.

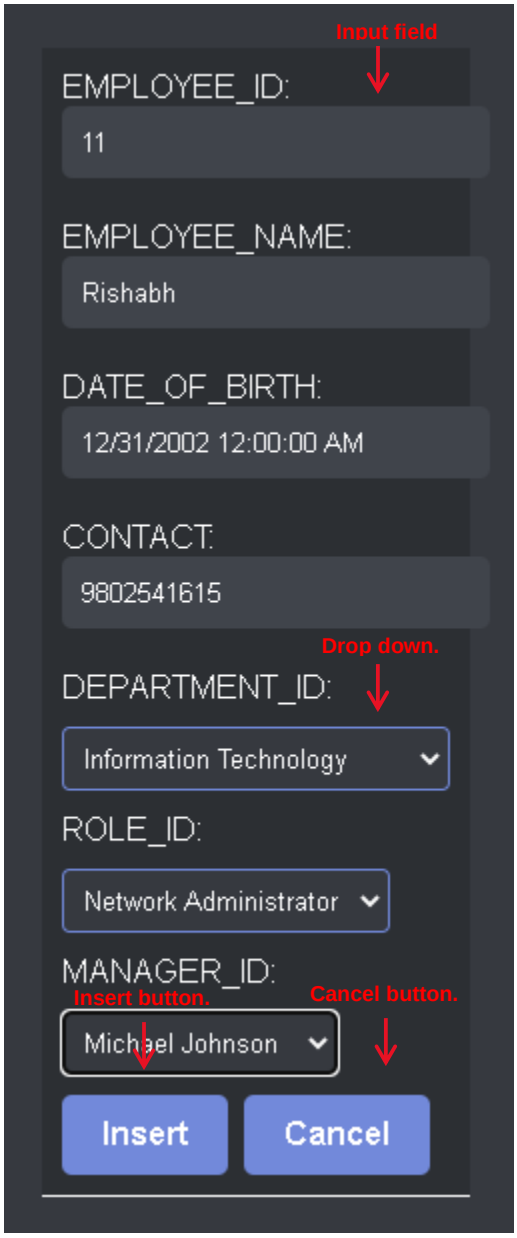


Figure 42: Employee form

Over here we are greeted with 3 drop down and two buttons the drop down is used to select the foreign key items as there could be conflict in primary key dropdown was used.

Then the insert button inserts the data on the form and the cancel will simply close the form.

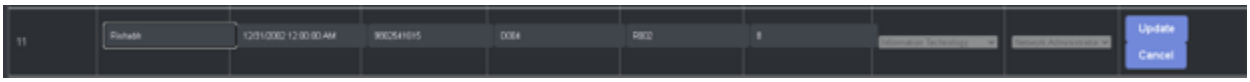


Figure 43: edit table

After the edit button is clicked we are shown the records in above format where we can edit any table then press the update button if not we can just press cancel button

Job Form

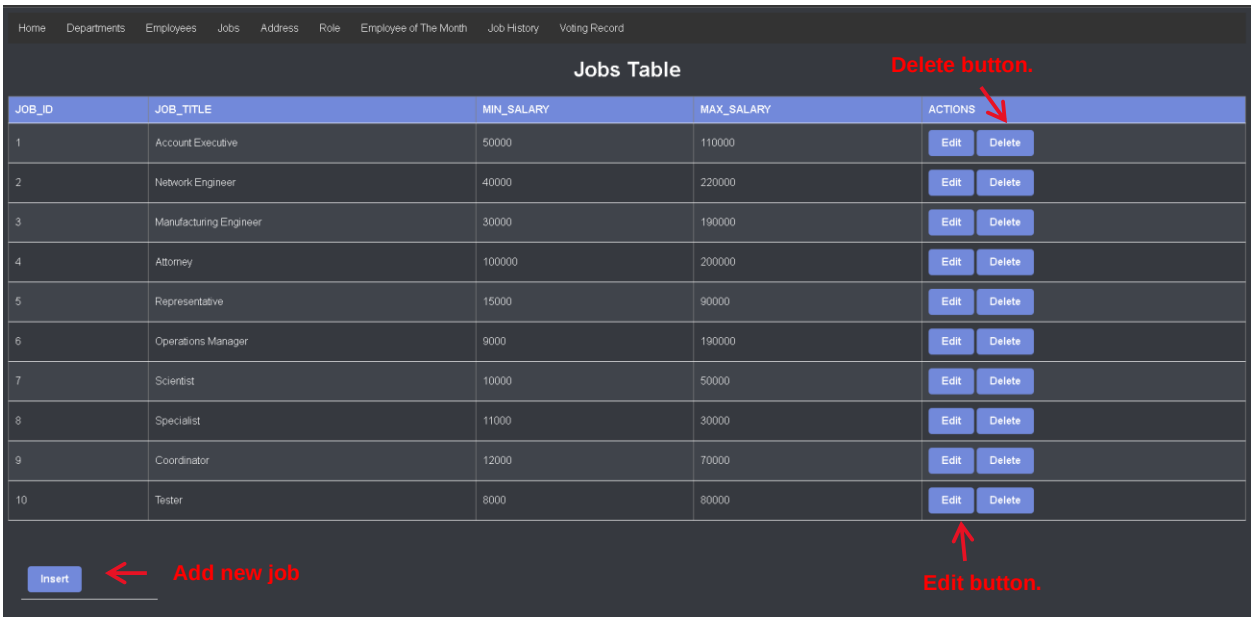


Figure 44: jobs table

The simple form contains four main features create, update, delete, read also known as (CRUD). The delete button deletes the selected record the edit button edits the selected record when we enter value in the text field the insert button then opens a small form where we can enter new employee details.

Other all functions are similar to the employee table

Department Form

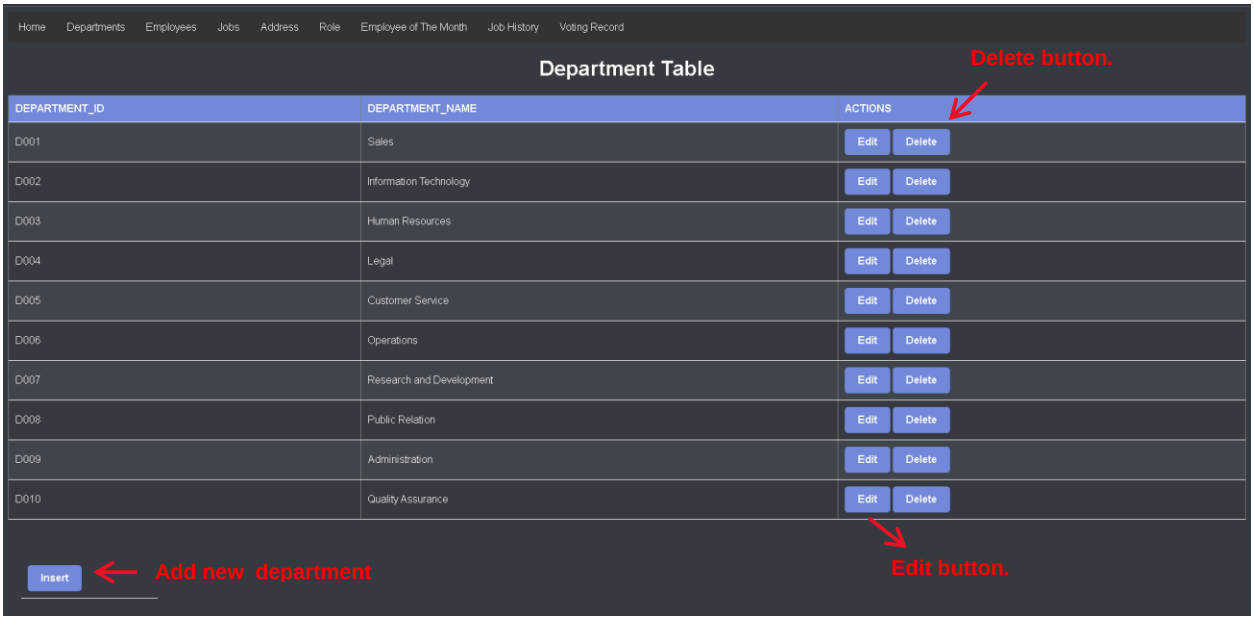


Figure 45: Department

Now all the functionality is same as of the employee form and it allows us to manipulate database in the same way.

Address Form

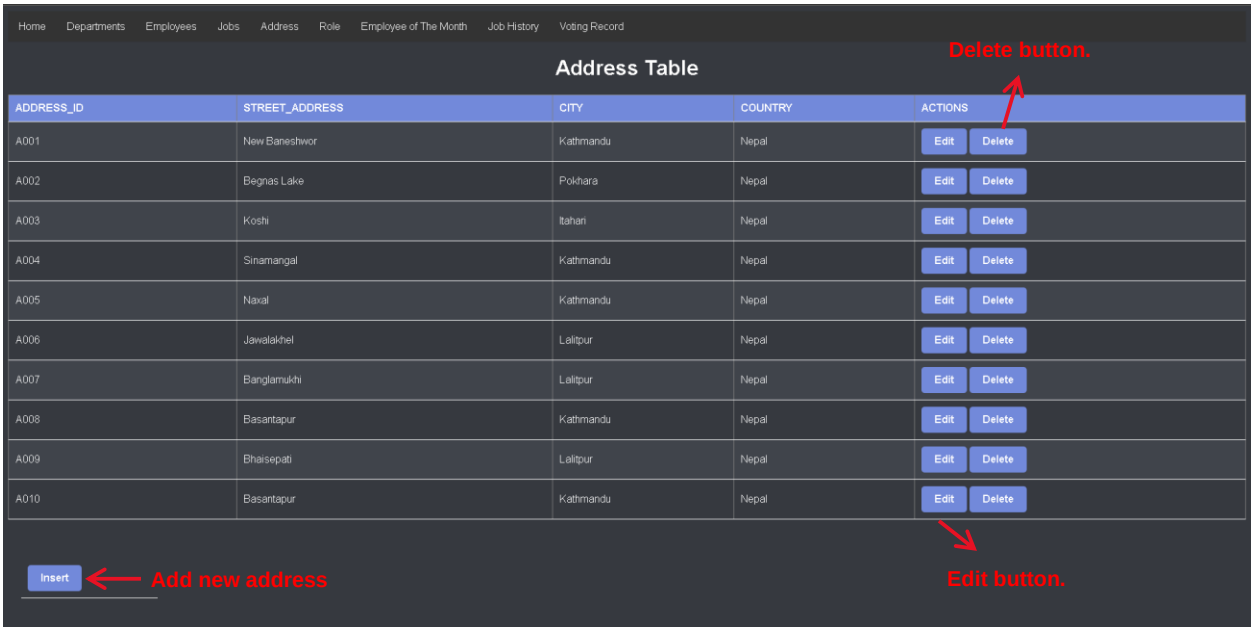


Figure 46: Address

Now all the functionality is same as of the employee form and it allows us to manipulate database in the same way.

Role form

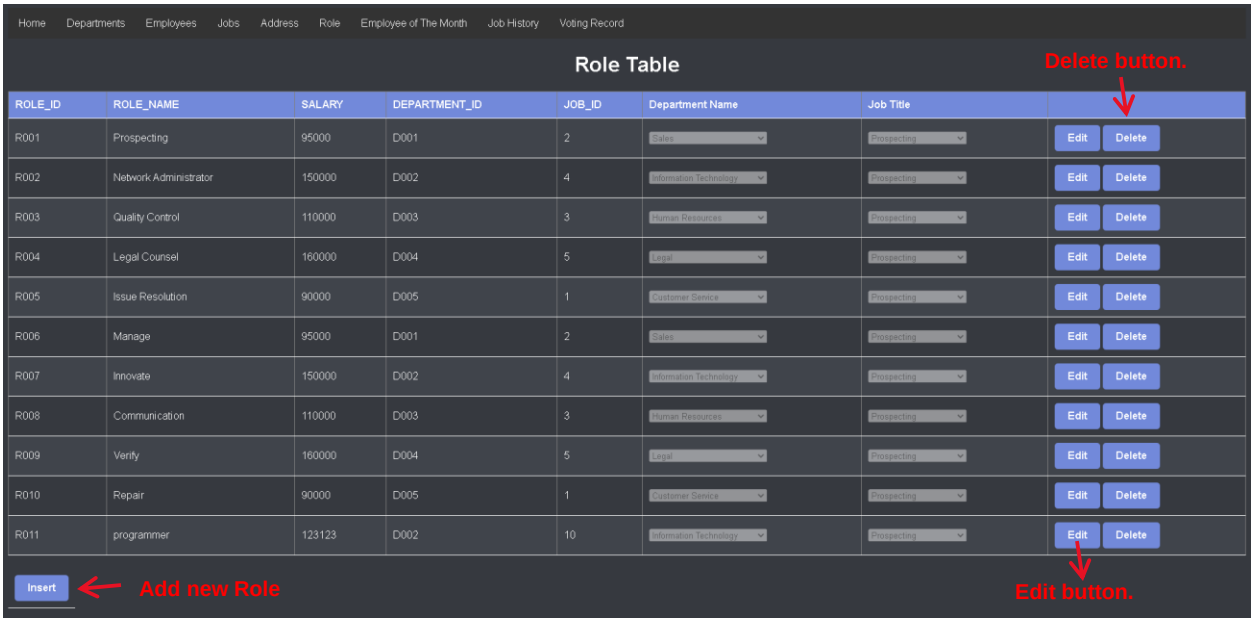


Figure 47: Role

Figure 48: Role

Now all the functionality is same as of the employee form and it allows us to manipulate database in the same way.

12.1.3. Complex Form
Employee of the month form

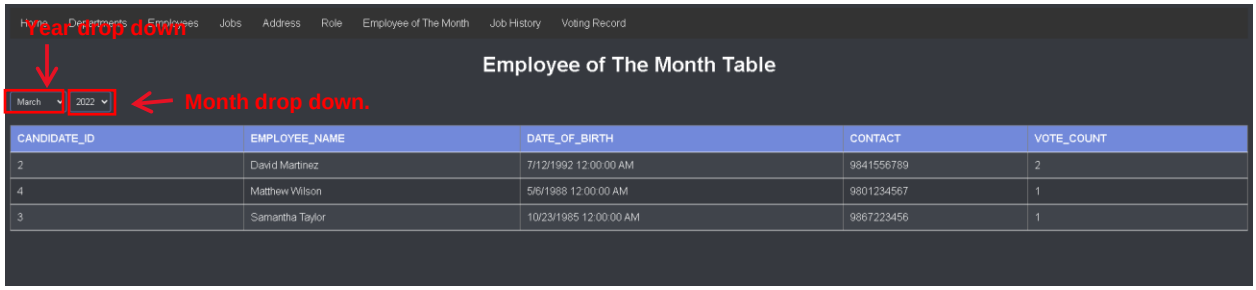


Figure 49: Employee of month

In the complex form the drop down allows us to select the desired month and year which will retrieve the total voting record of the specific month and year that we selected.

Voter Detail form

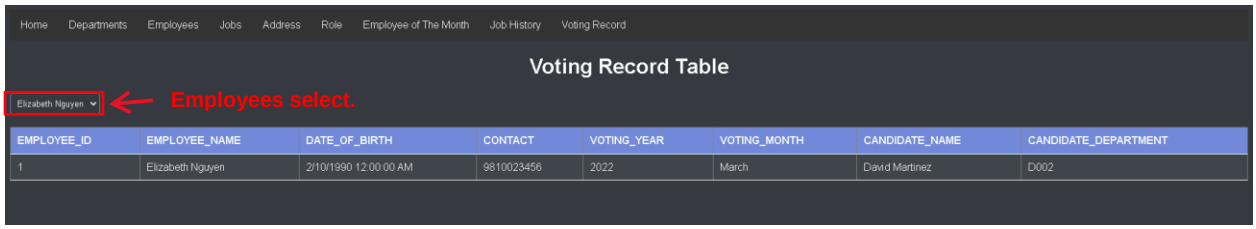


Figure 50: Voter detail

The complex form drop down for voting record allows us to select the voter which will retrieve the total voting record of the employee that we selected.

Job history form

Figure 51: Job history

The complex form drop down for job history allows us to select the history which will retrieve the total voting record of the employee that we selected.

13. Testing

13.1. Testing for address form

13.1.1. Insert operation.

Objective	To test insert operation of Address form
Action	Insert button was clicked. Then data was filled inside the text boxes and insert button was pressed again.
Expected Outcome	The data will be successfully inserted and shown in the gridview.
Actual Result	The data was displayed in the address grid view
Conclusion	Test Successful

Table 6: Insert operation address

Action:

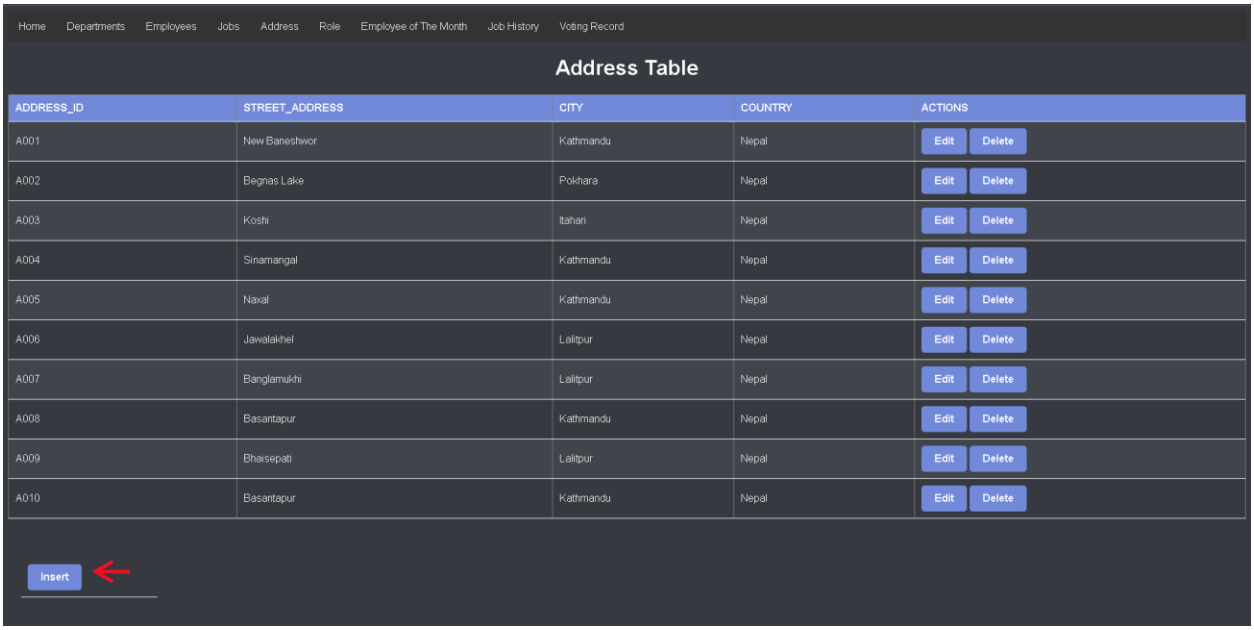


Figure 52: Test1

Action:

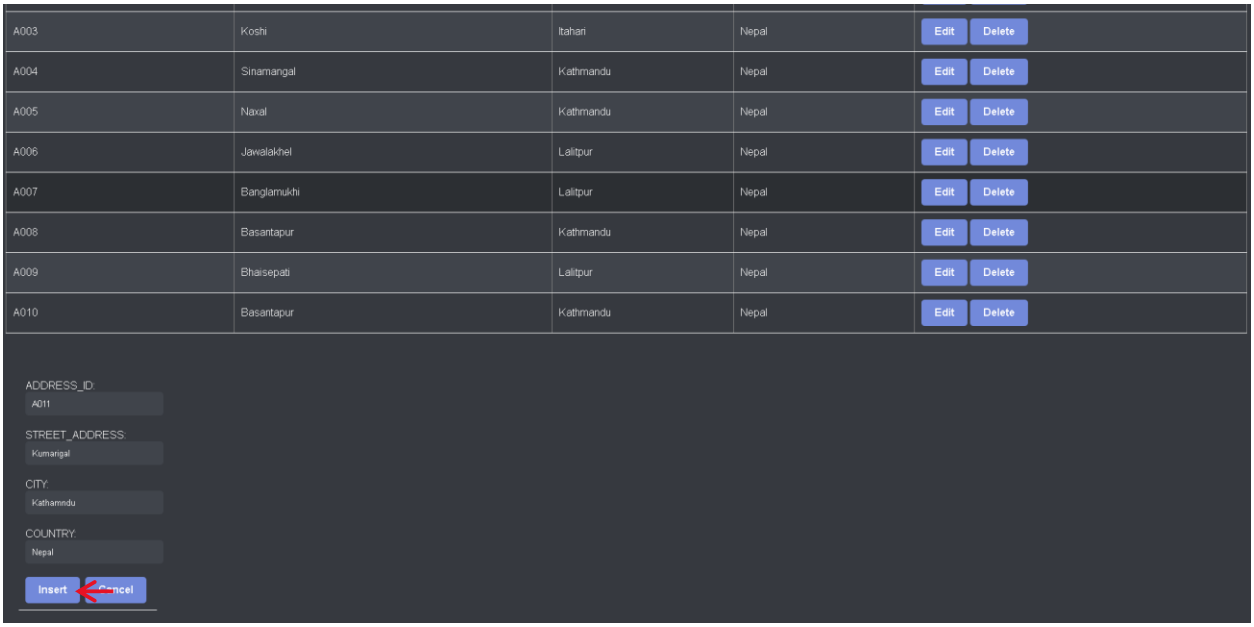


Figure 53: Test 2

Result:

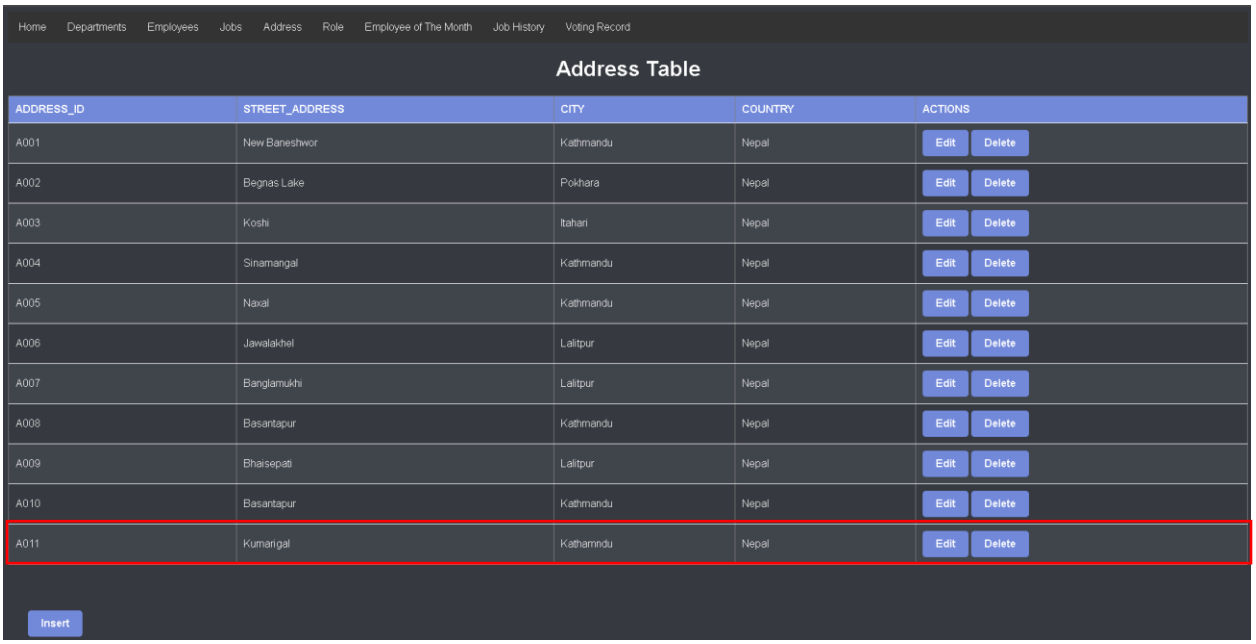


Figure 54: Test 3

13.1.2 Edit Operation

Objective	To test delete operation of address form
Action	Delete button was clicked inside grid view
Expected Outcome	The web page will refresh, and the data would be deleted
Actual Result	The data inside the address grid view was deleted
Conclusion	Test successful

Table 7: Edit operation address

Action:

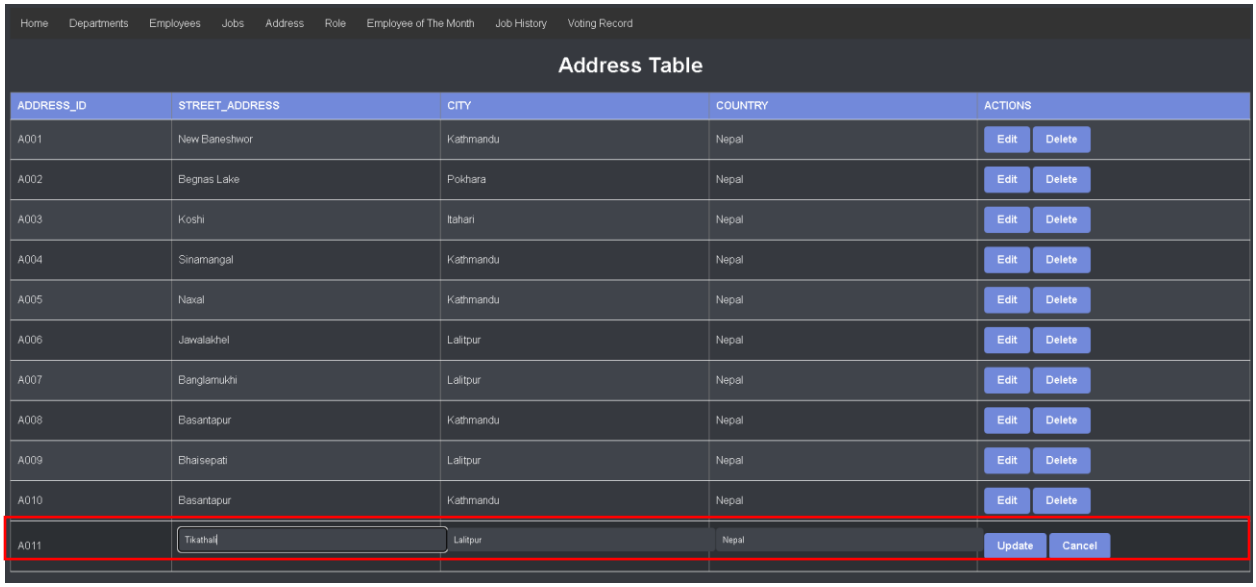


Figure 55: Test 4

Result:

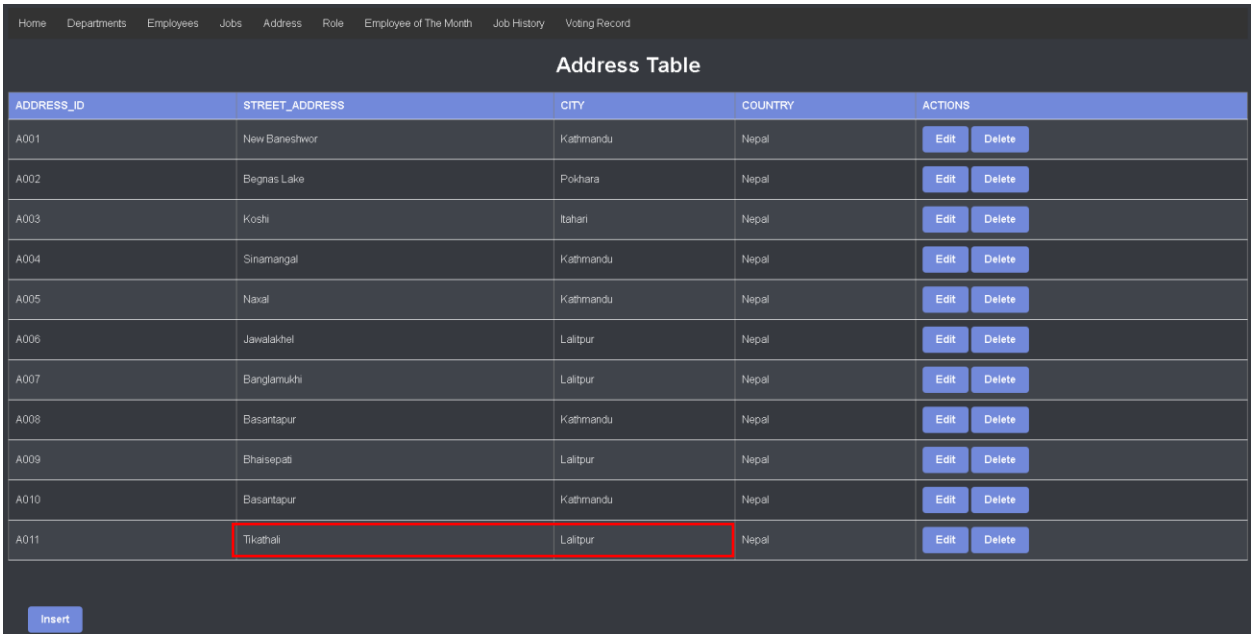


Figure 56: Test 5

13.1.3. Delete Operation

Objective	To test delete operation of address form
Action	Delete button was clicked inside the grid view.
Expected Outcome	The web page will refresh, and the data would be deleted.
Actual Result	The data inside the address Grid View was deleted.
Conclusion	Test Successful

Table 8: Delete operation address

Action:

Figure 57: Test 6 address

Result:

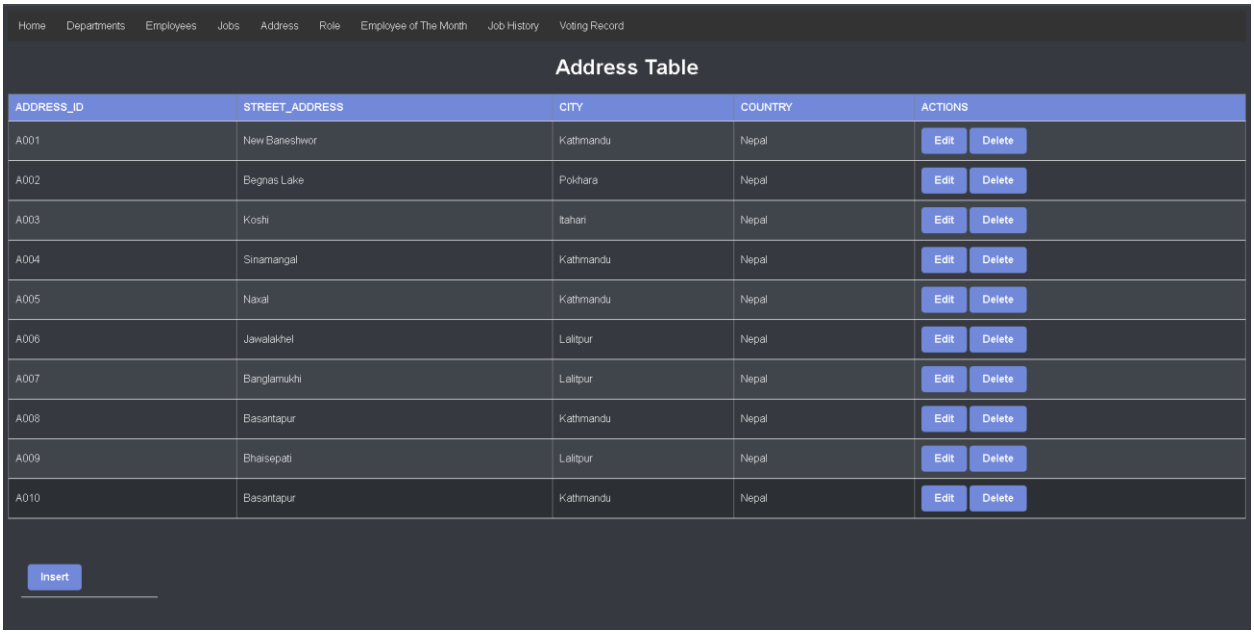


Figure 58: Test 7 address

13.2 Test case Employee

13.2.1 Insert operation.

Objective	To test insert operation of employee form
Action	Insert button was clicked. Then data was filled inside the text boxes and proper department and role was selected and insert button was pressed again.
Expected Outcome	The data will be successfully inserted and shown in the grid view.
Actual Result	The data was displayed in the employee grid view
Conclusion	Test Successful

Table 9: Insert operation employee

Action:

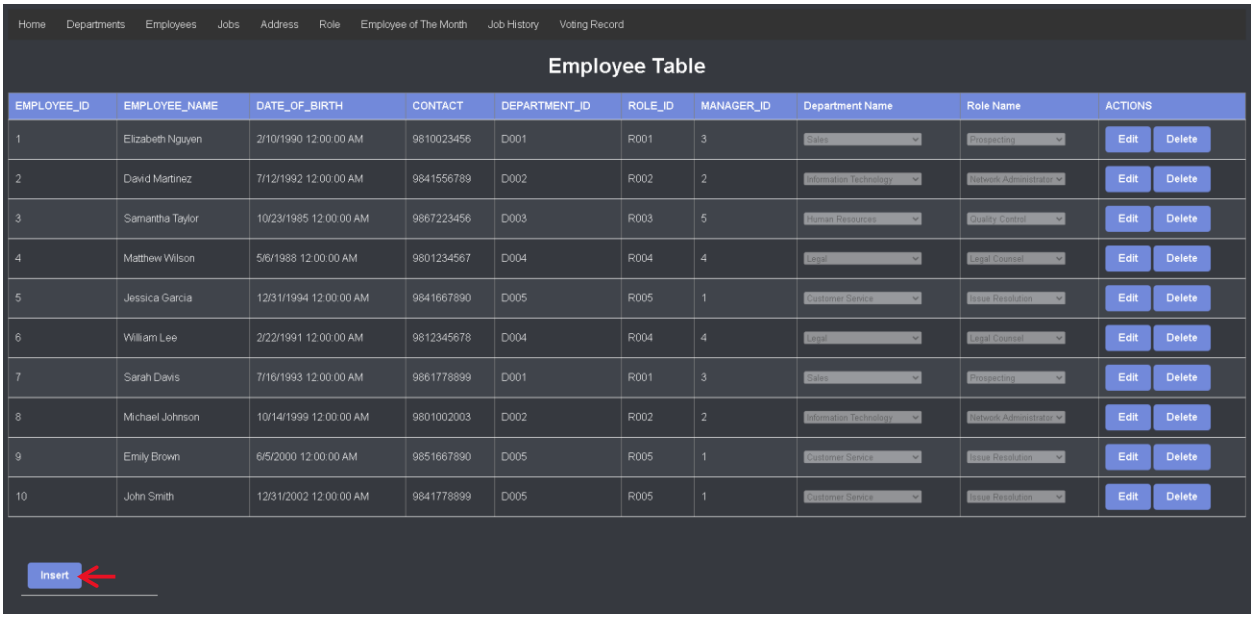


Figure 59: Test 8 employee

Action:

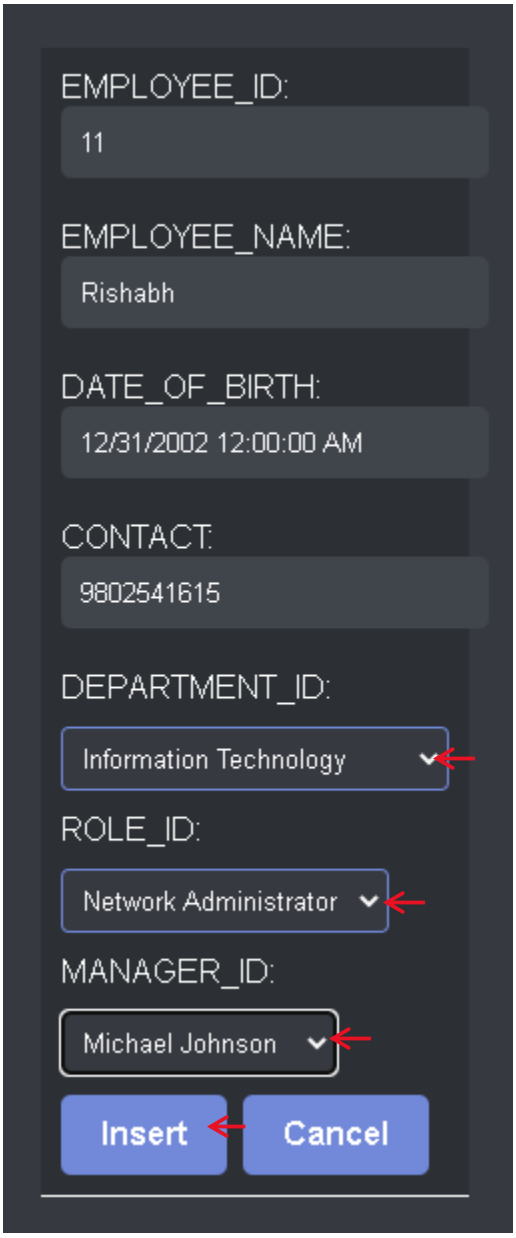


Figure 60: Test 9 employee form

Result:

Figure 61: Test 10 employee insert

13.2.2. Edit Operation

Objective	To test delete operation of employee form
Action	Delete button was clicked inside grid view
Expected Outcome	The web page will refresh, and the data would be deleted
Actual Result	The data inside the address grid view was deleted
Conclusion	Test successful

Table 10: Edit operation employee

Action:

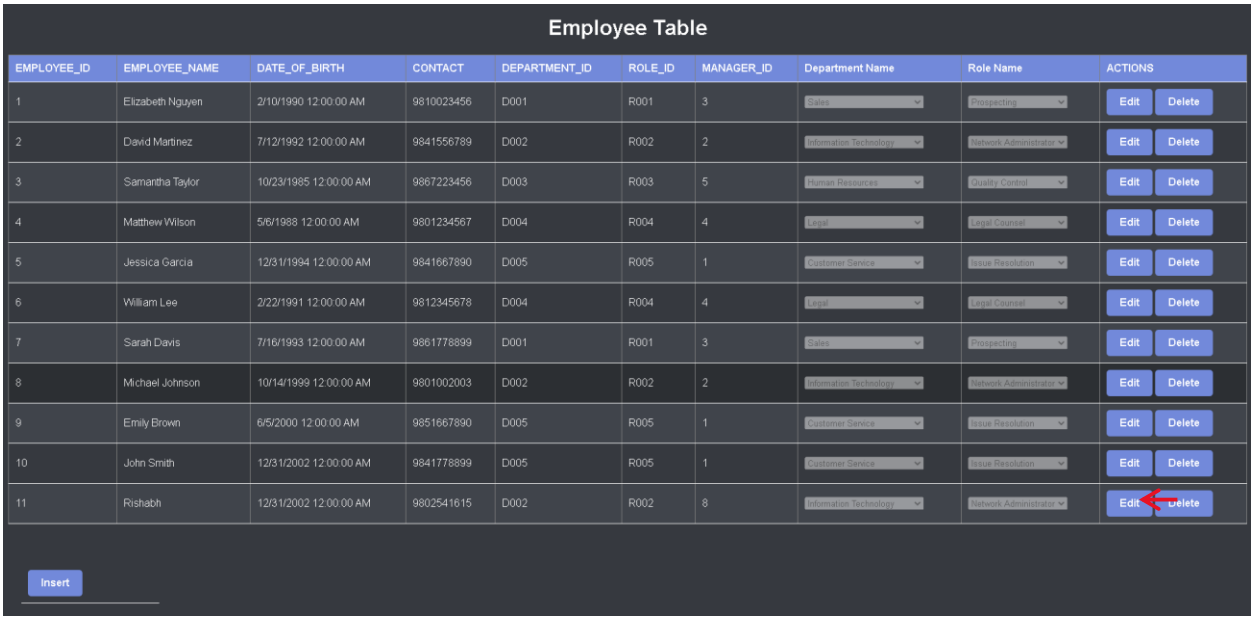


Figure 62: Test 11 employee form

Action:

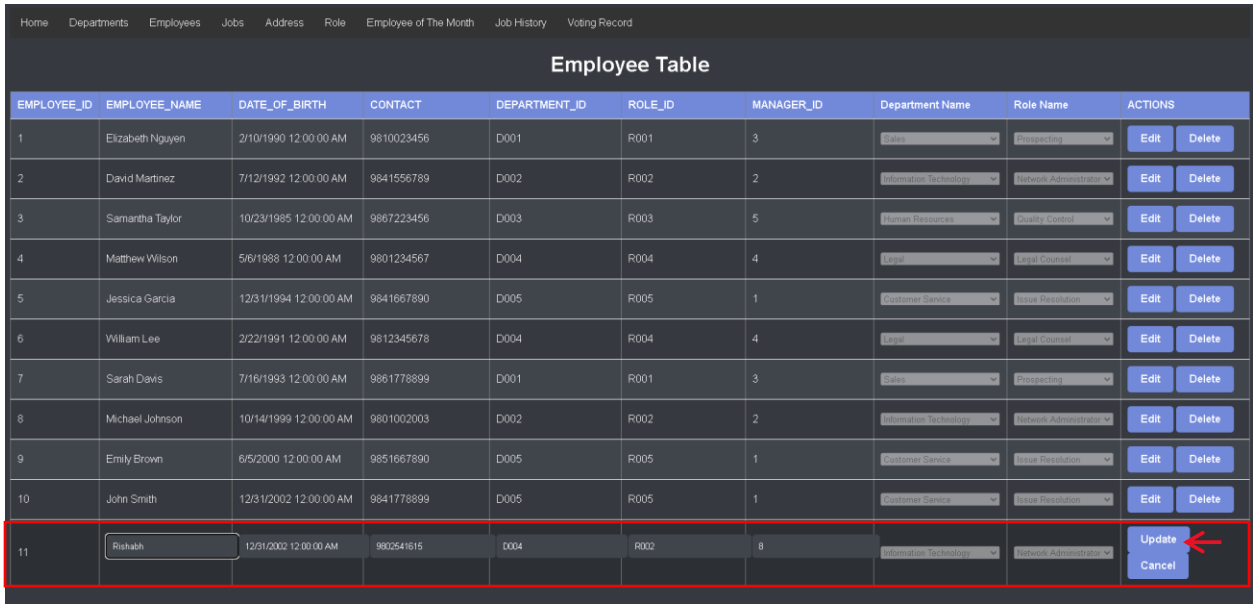


Figure 63: test 12 employee edit.

Result:

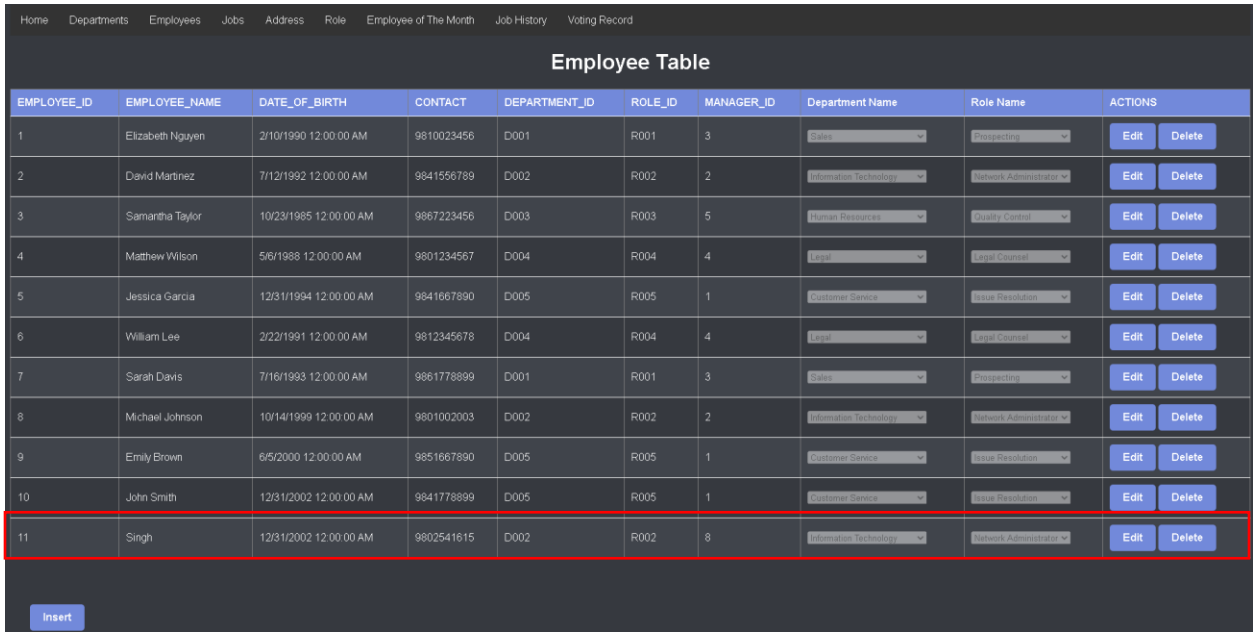


Figure 64: test13 employee updated.

13.2.3 Delete Operation

Objective	To test delete operation of employee form
Action	Delete button was clicked inside the grid view.
Expected Outcome	The web page will refresh, and the data would be deleted.
Actual Result	The data inside the address Grid View was deleted.
Conclusion	Test Successful

Table 11: delete operation employee.

Action:

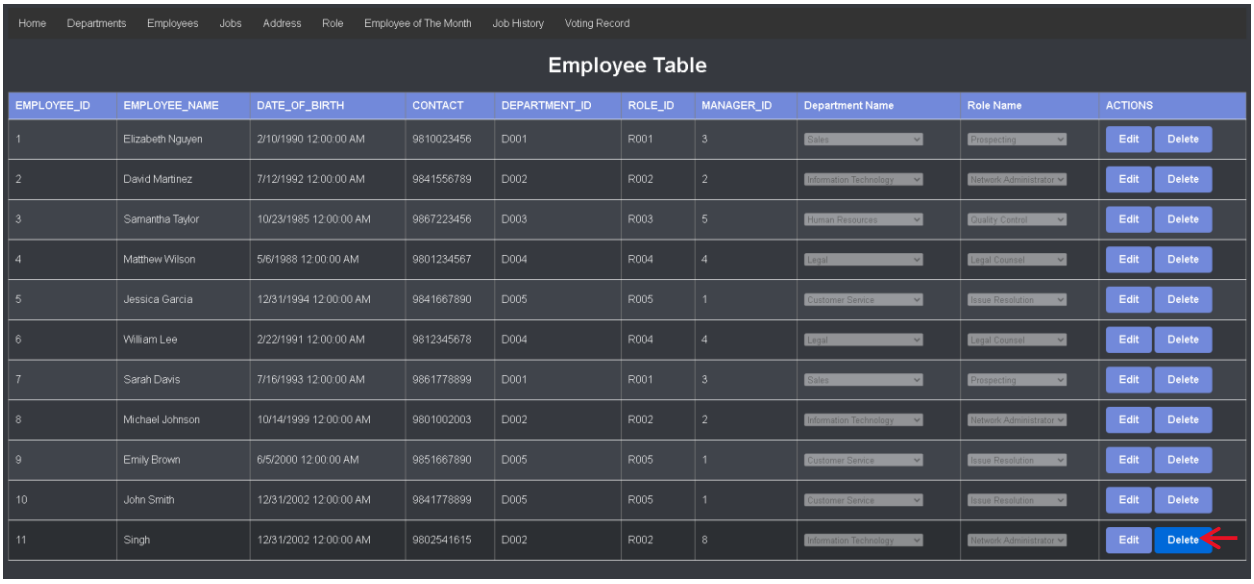


Figure 65: test 14 Employee delete

Result:

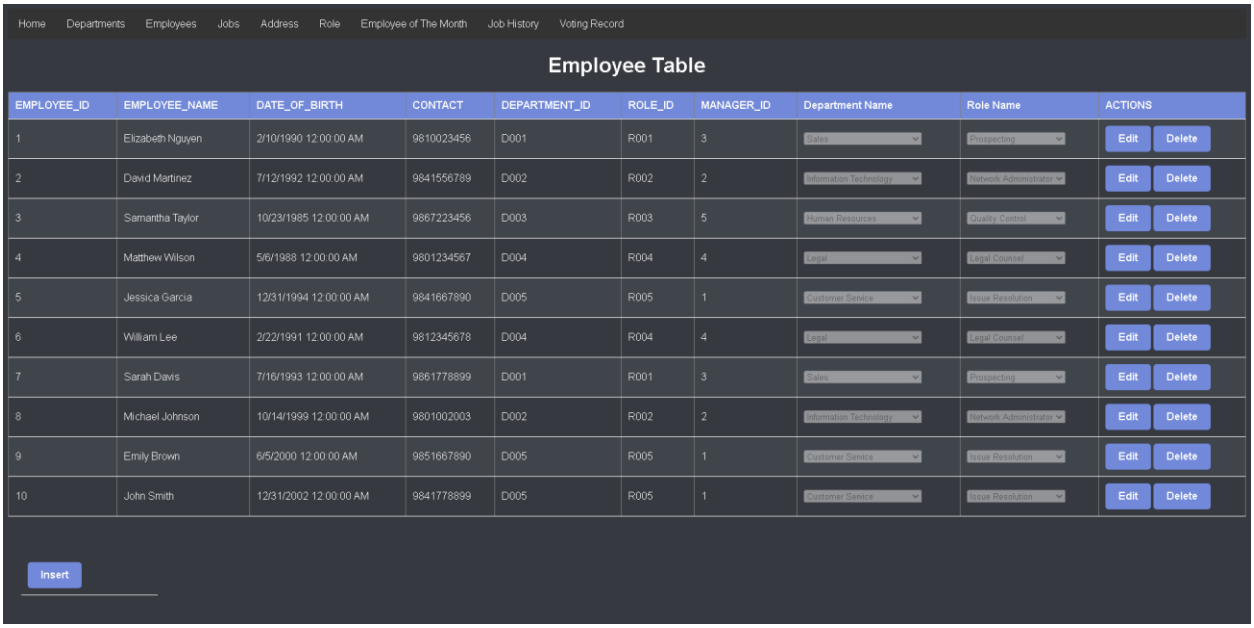


Figure 66: Test 15 employee deleted.

13.3 Job testing

13.3.1 Insert operation.

Objective	To test insert operation of Job form
Action	Insert button was clicked. Then data was filled inside the text boxes and insert button was pressed again.
Expected Outcome	The data will be successfully inserted and shown in the grid view.
Actual Result	The data was displayed in the address grid view
Conclusion	Test Successful

Table 12: insert job

Action:

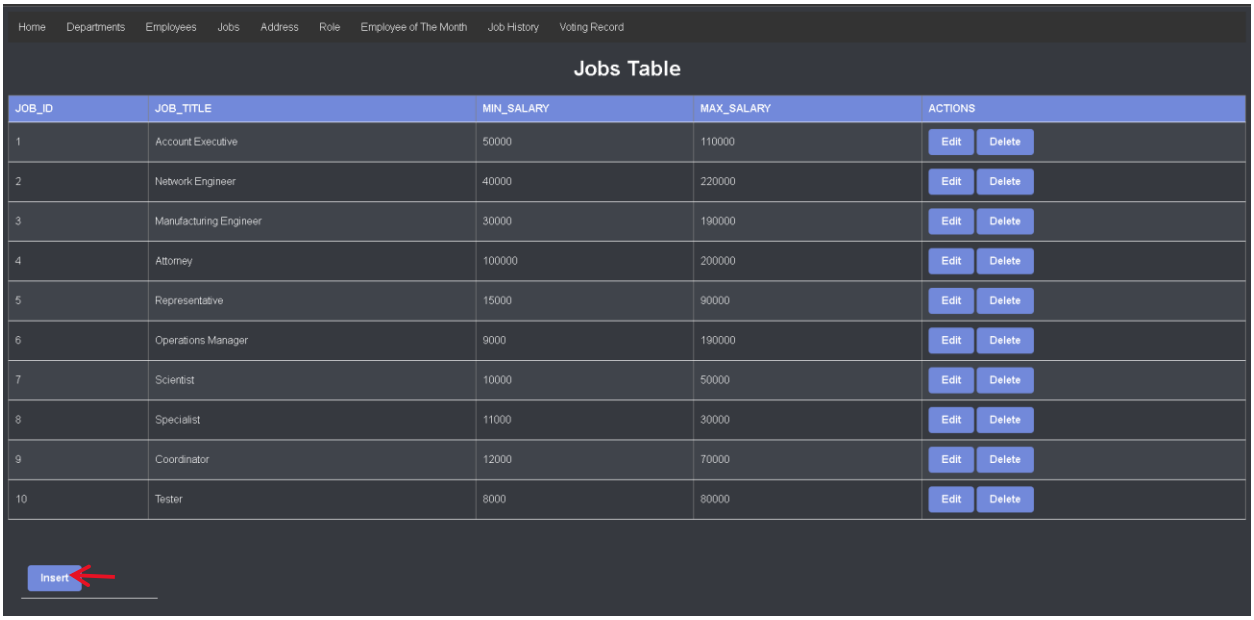


Figure 67: Test 15 job form

Action:

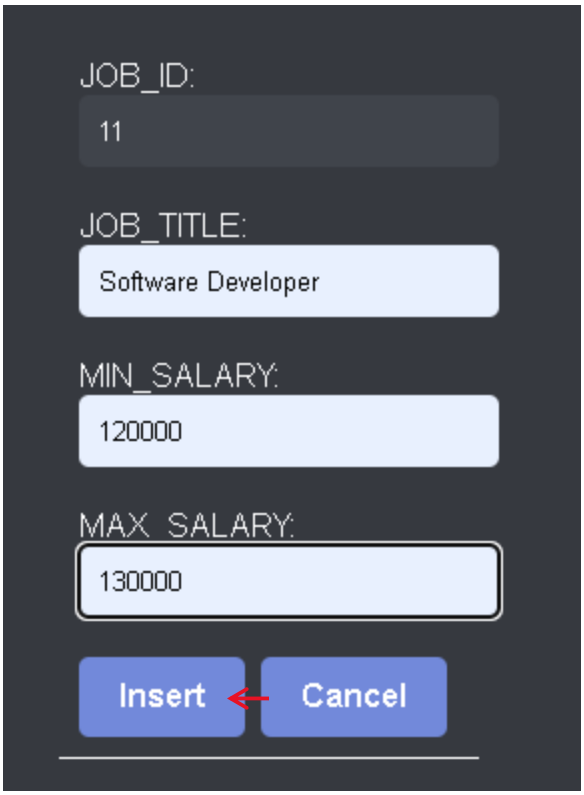


Figure 68: Test 16 job form

Result:

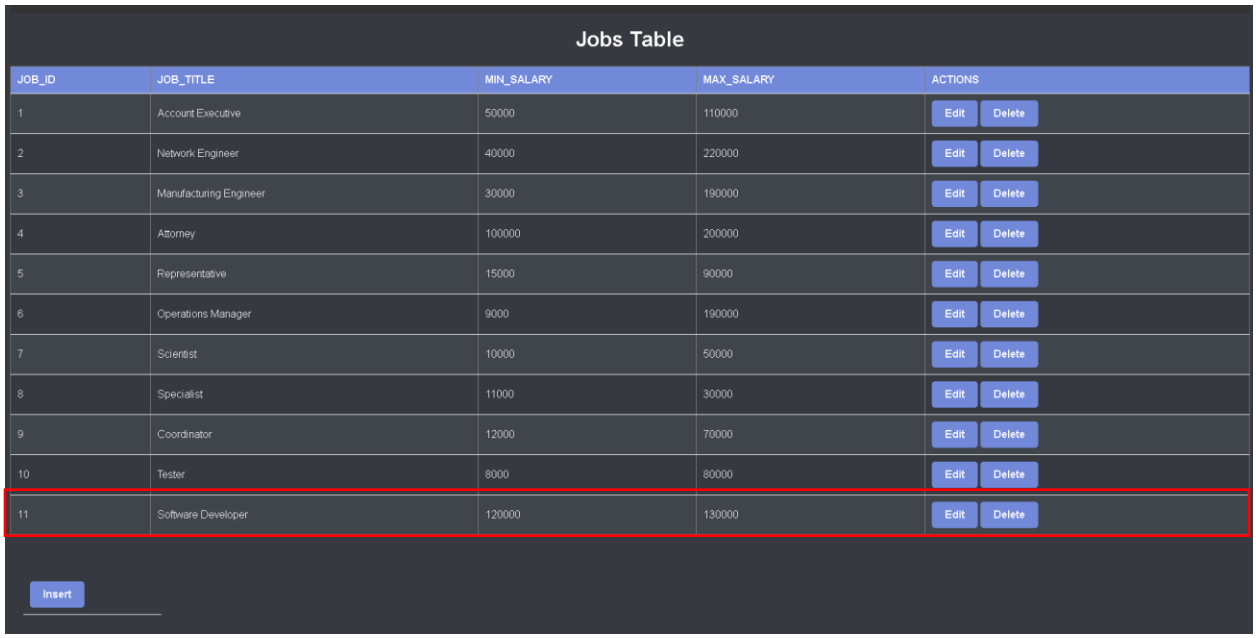


Figure 69: Test 16 job inserted.

13.3.2 Edit Operation

Objective	To test delete operation of Job form
Action	Delete button was clicked inside grid view
Expected Outcome	The web page will refresh, and the data would be deleted
Actual Result	The data inside the Job grid view was deleted
Conclusion	Test successful

Table 13: Edit job test

Action:

Figure 70: Test 17 jobs edit.

Action:

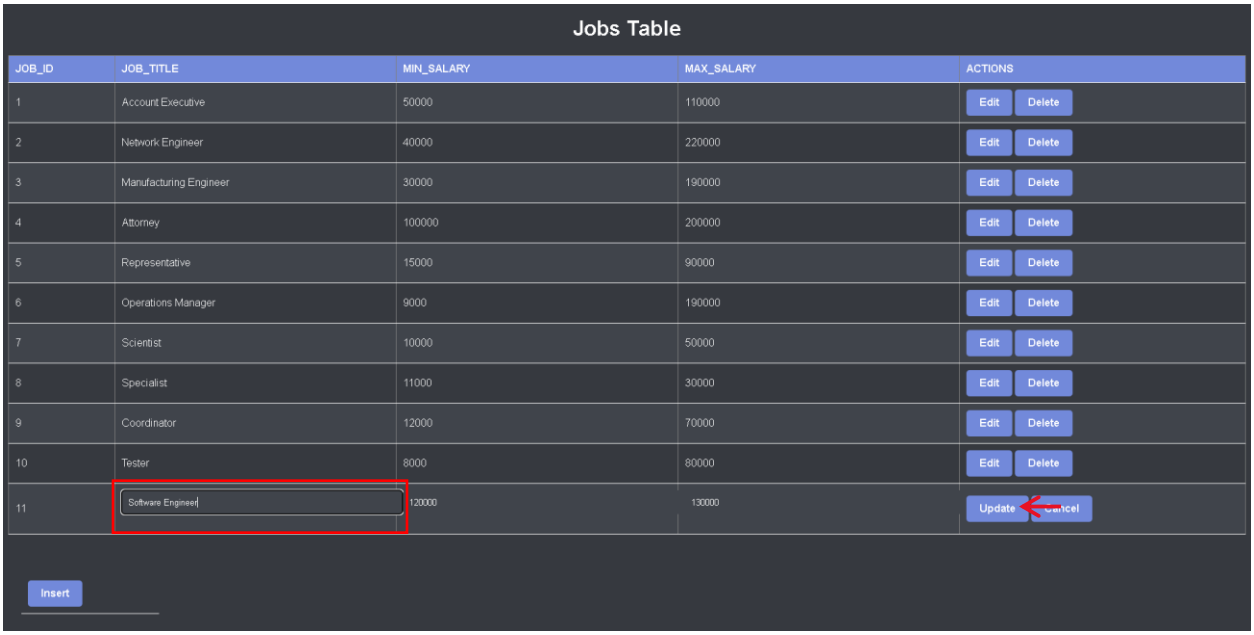


Figure 71: test 18 edit jobs

Result:

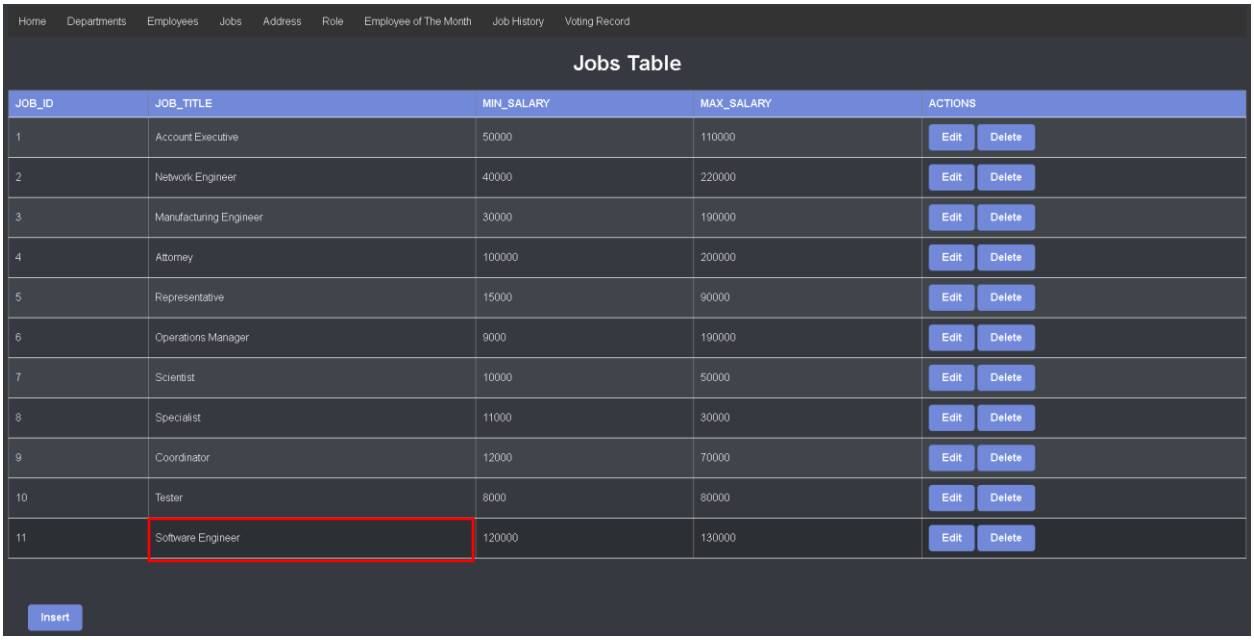


Figure 72: Test 19 updated

13.3.3 Delete Operation

Objective	To test delete operation of Job form
Action	Delete button was clicked inside the grid view.
Expected Outcome	The web page will refresh, and the data would be deleted.
Actual Result	The data inside the address Grid View was deleted.
Conclusion	Test Successful

Table 14: Delete operation

Action:

Figure 73: Test 20 delete action

Result:

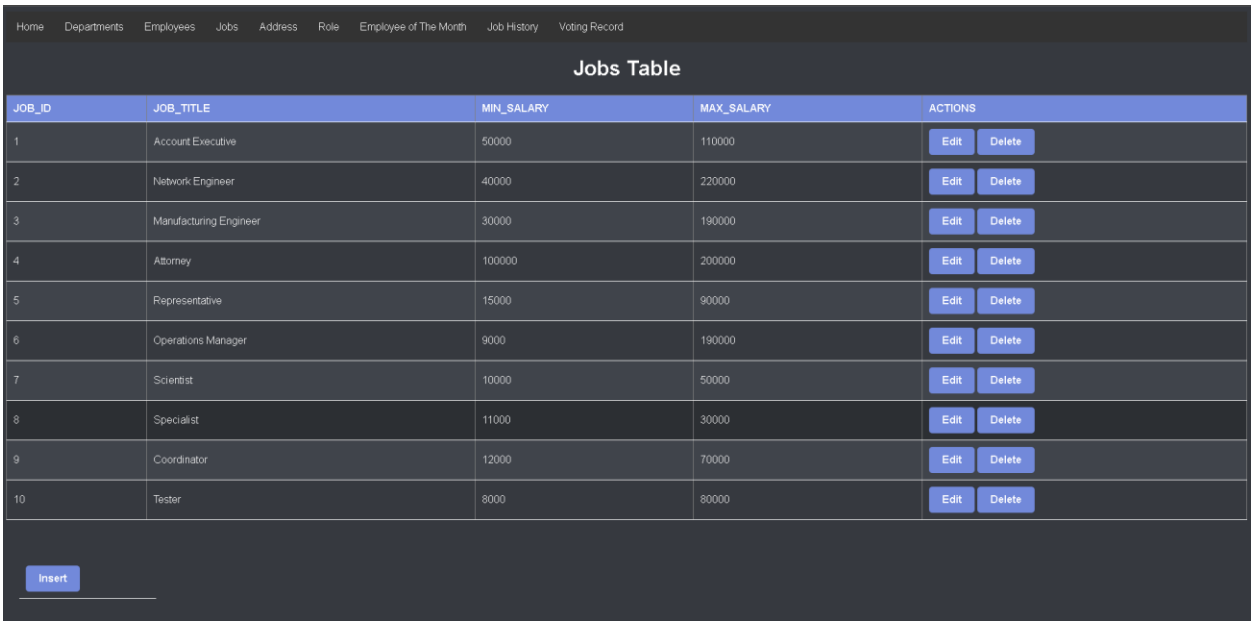


Figure 74: Test 21 deleted

13.4 Role table

13.4.1 Insert operation.

Objective	To test insert operation of Role form
Action	Insert button was clicked. Then data was filled inside the text boxes and insert button was pressed again.
Expected Outcome	The data will be successfully inserted and shown in the grid view.
Actual Result	The data was displayed in the Role grid view
Conclusion	Test Successful

Table 15: insert role

Action:

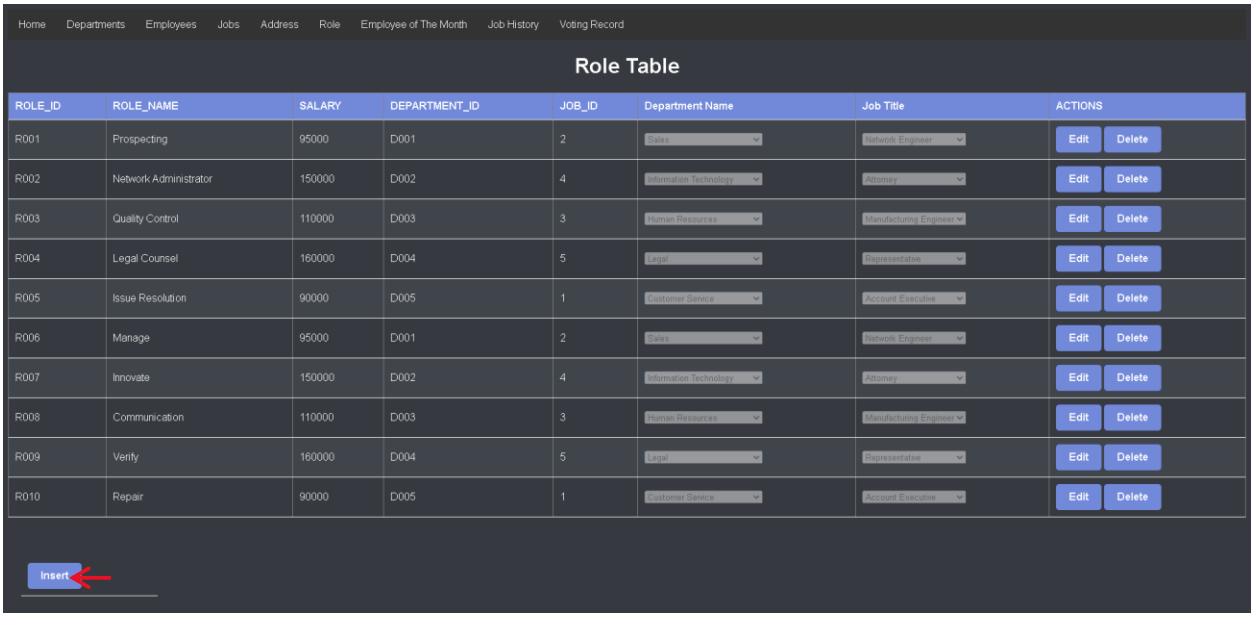


Figure 75: Test 22 insert role

Result:

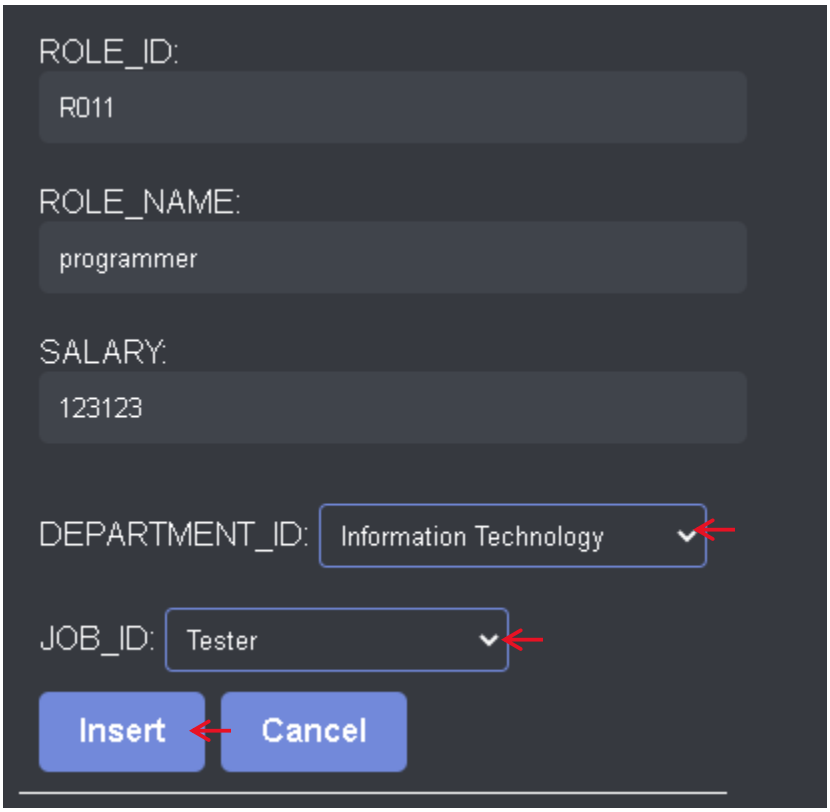


Figure 76: test 23 insert role form

Result:

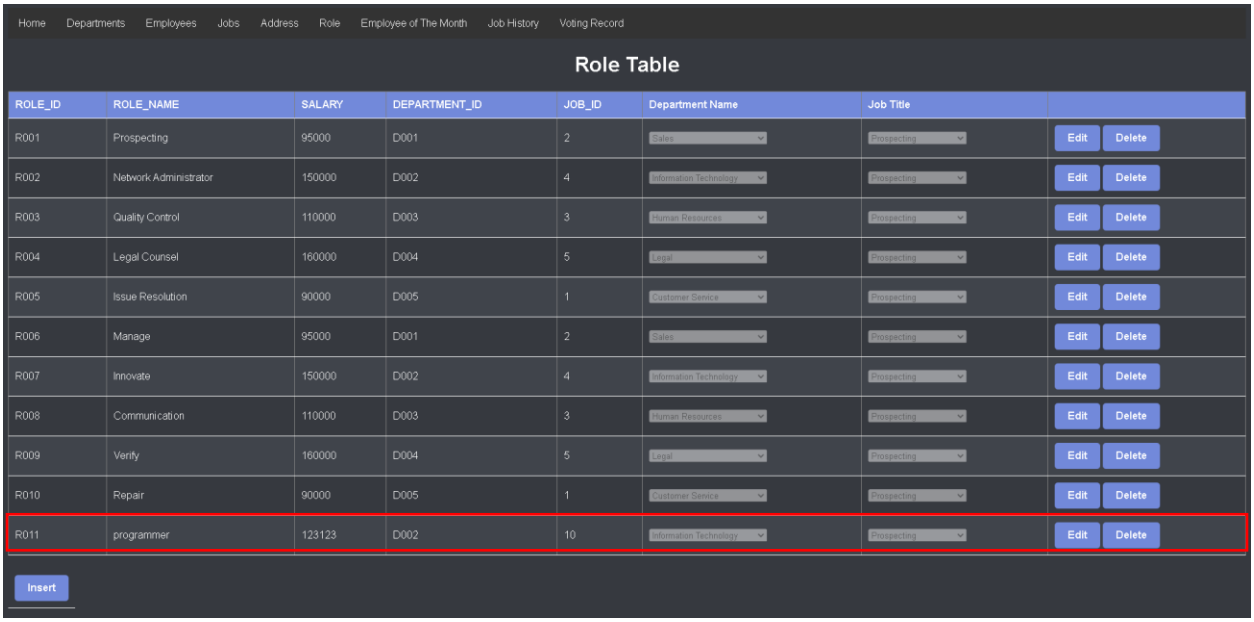


Figure 77: Test 24 inserted

13.4.1 Edit Operation

Objective	To test delete operation of Role form
Action	Delete button was clicked inside grid view
Expected Outcome	The web page will refresh, and the data would be deleted
Actual Result	The data inside the Role grid view was deleted
Conclusion	Test successful

Table 16: edit role

Action:

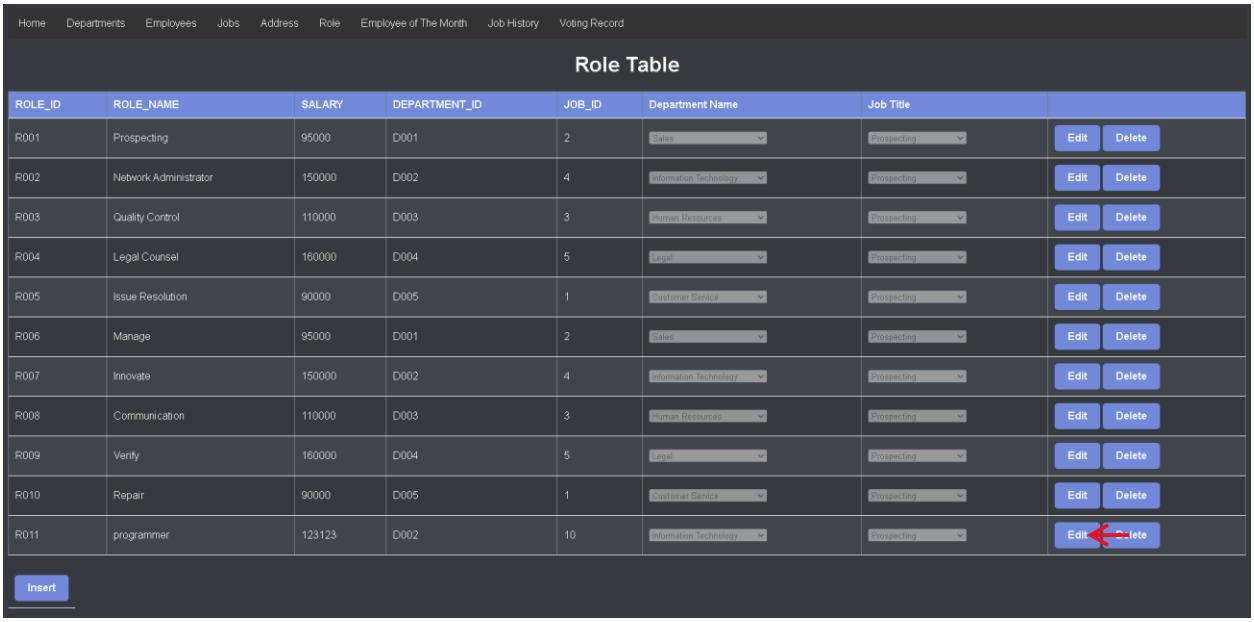


Figure 78 :Test 25 edit

Result:

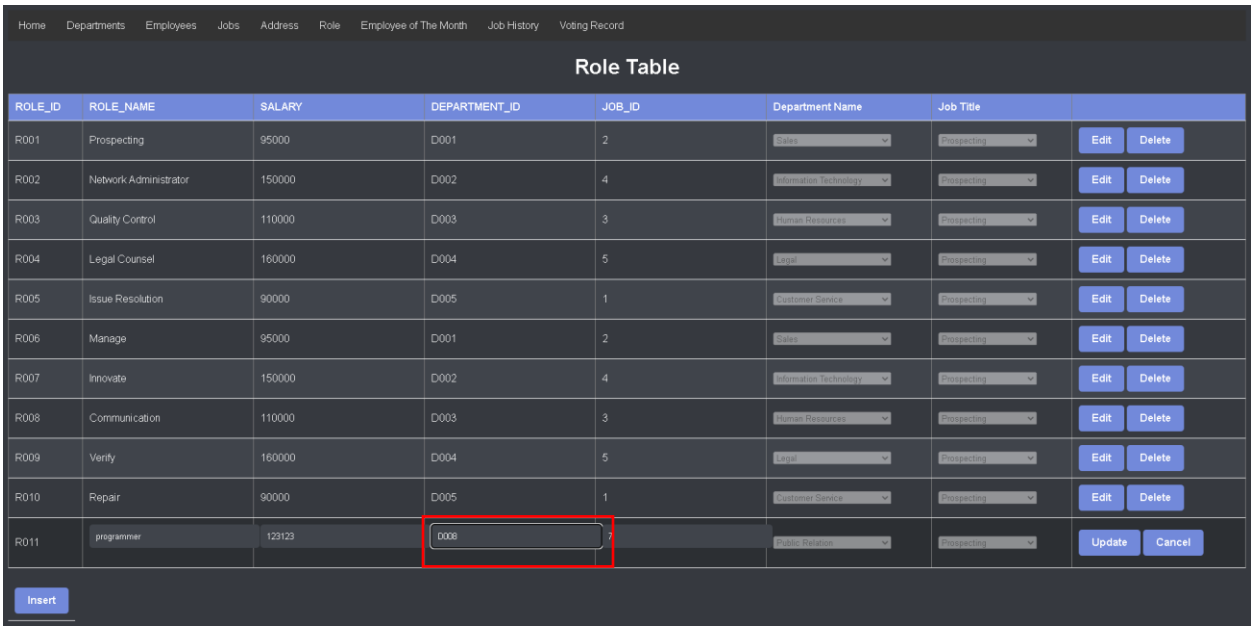


Figure 79: test 26 enter in field

Result:

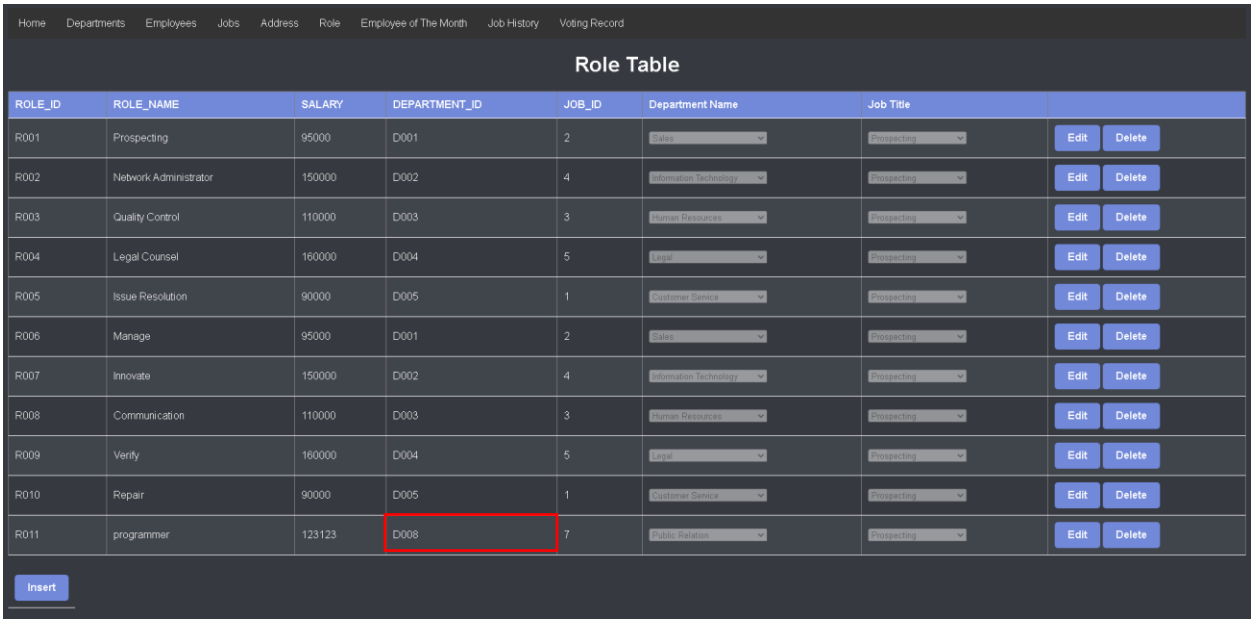


Figure 80: test 27 field updated.

13.4.3 Delete Operation

Objective	To test delete operation of Role form
Action	Delete button was clicked inside the grid view.
Expected Outcome	The web page will refresh, and the data would be deleted.
Actual Result	The data inside the Role Grid View was deleted.
Conclusion	Test Successful

Table 17: delete role

Action:

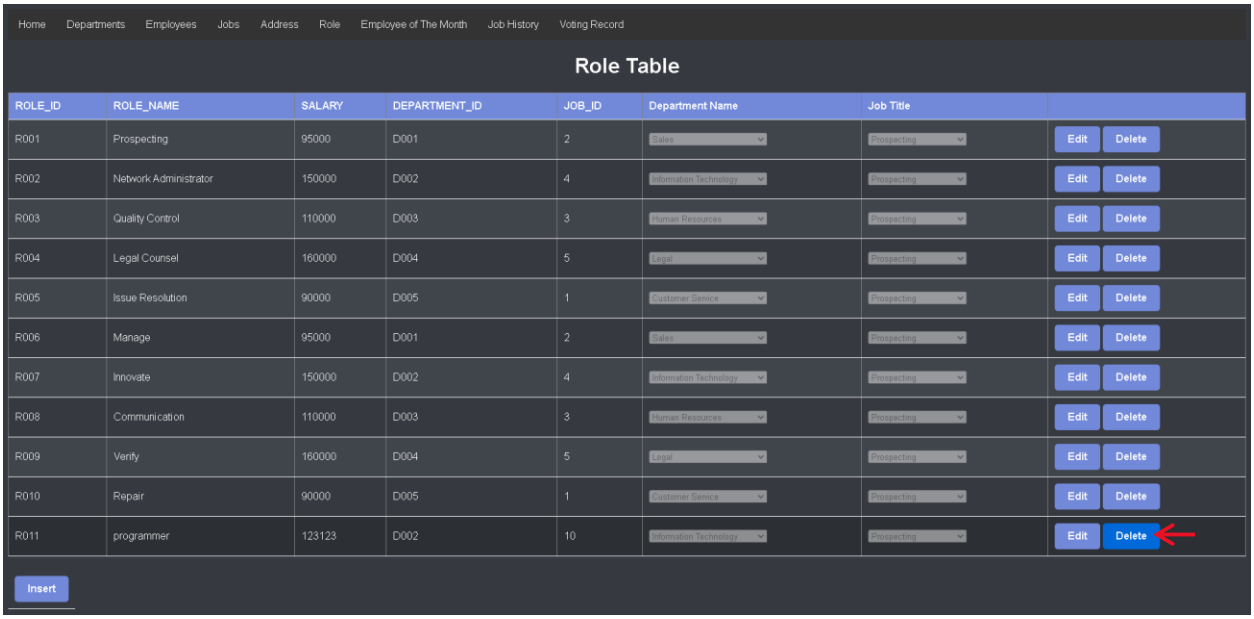


Figure 81: test 28 press delete.

Result:

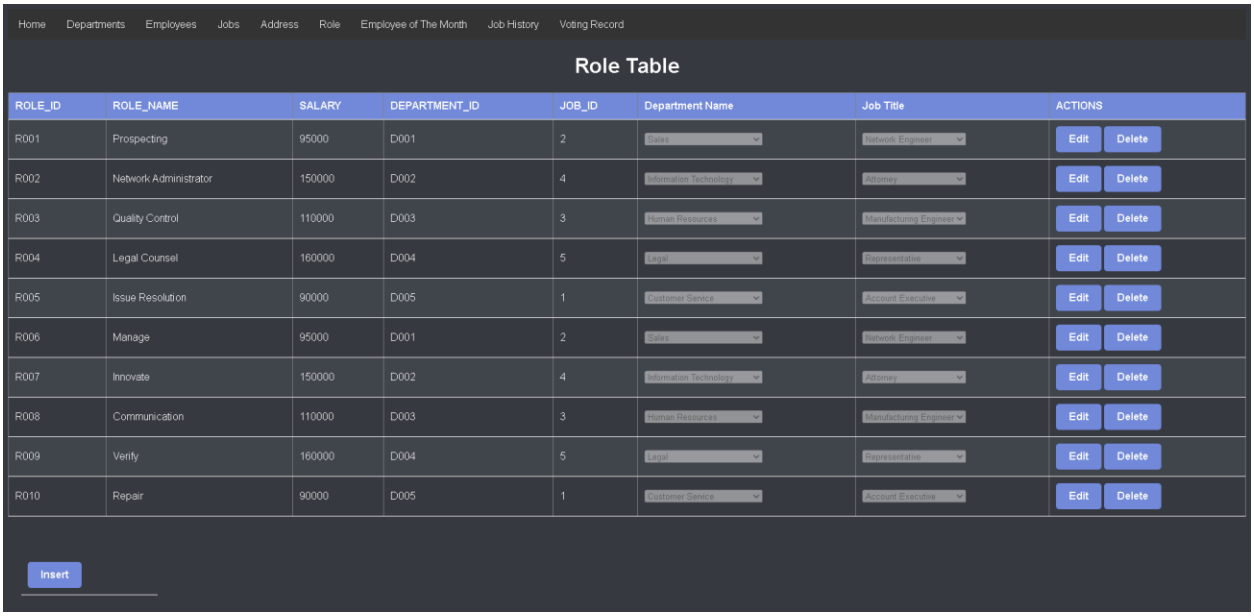


Figure 82: test 29 deleted

13.5. Department

13.5.1. Insert operation.

Objective	To test insert operation of Department form
Action	Insert button was clicked. Then data was filled inside the text boxes and insert button was pressed again.
Expected Outcome	The data will be successfully inserted and shown in the grid view.
Actual Result	The data was displayed in the Department grid view
Conclusion	Test Successful

Action

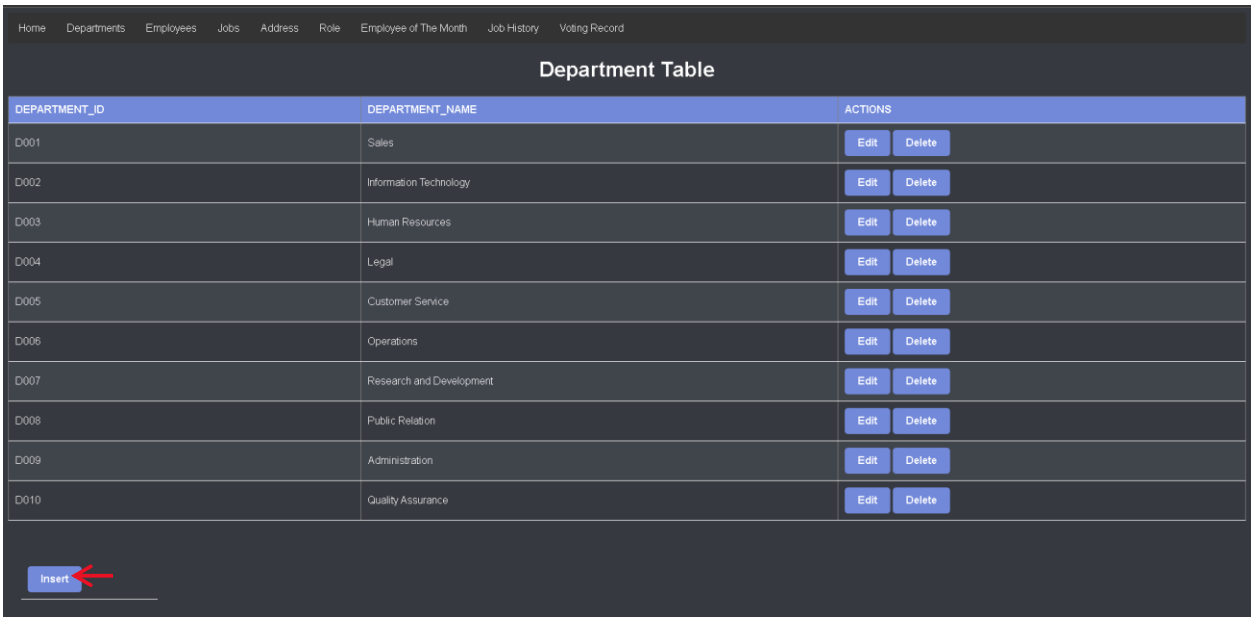


Figure 83: insert form

Action

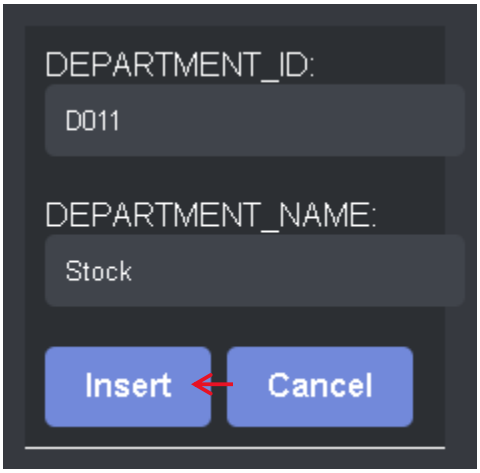


Figure 84: insert form.

Result

Figure 85: Inserted

13.5.2 Edit Operation

Objective	To test delete operation of address form
Action	Delete button was clicked inside grid view
Expected Outcome	The web page will refresh, and the data would be deleted
Actual Result	The data inside the Department grid view was deleted
Conclusion	Test successful

Action:

Figure 86 :Edit form.

Result:

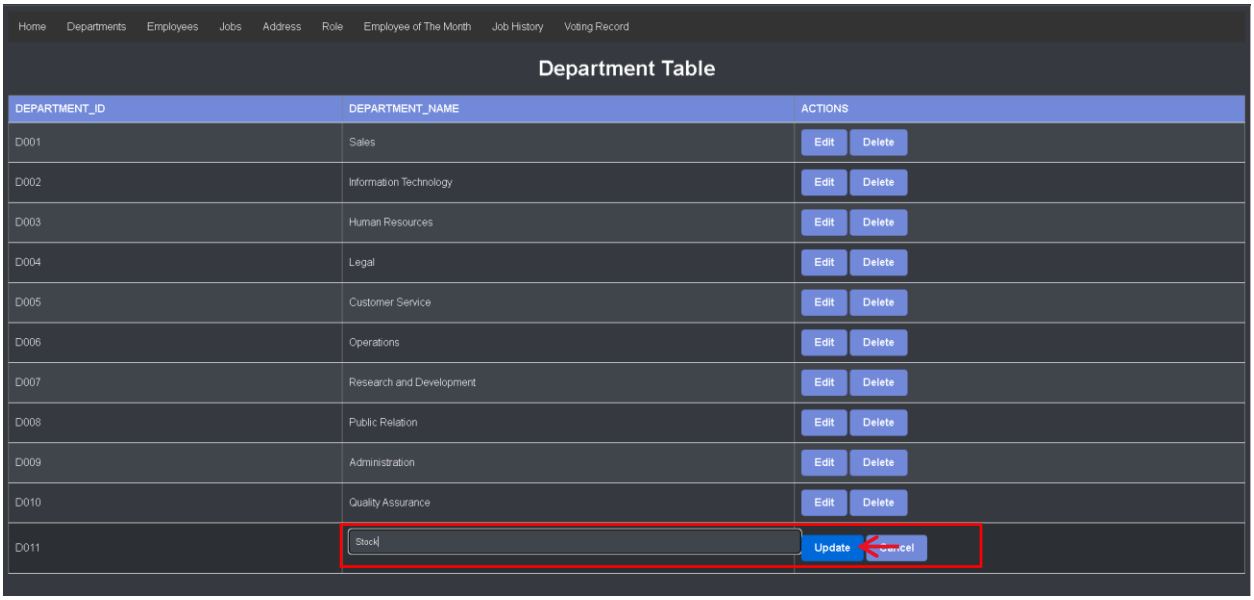


Figure 87: Form edited.

13.5.3 Delete Operation

Objective	To test delete operation of Department form
Action	Delete button was clicked inside the grid view.
Expected Outcome	The web page will refresh, and the data would be deleted.
Actual Result	The data inside the Department Grid View was deleted.
Conclusion	Test Successful

Action:

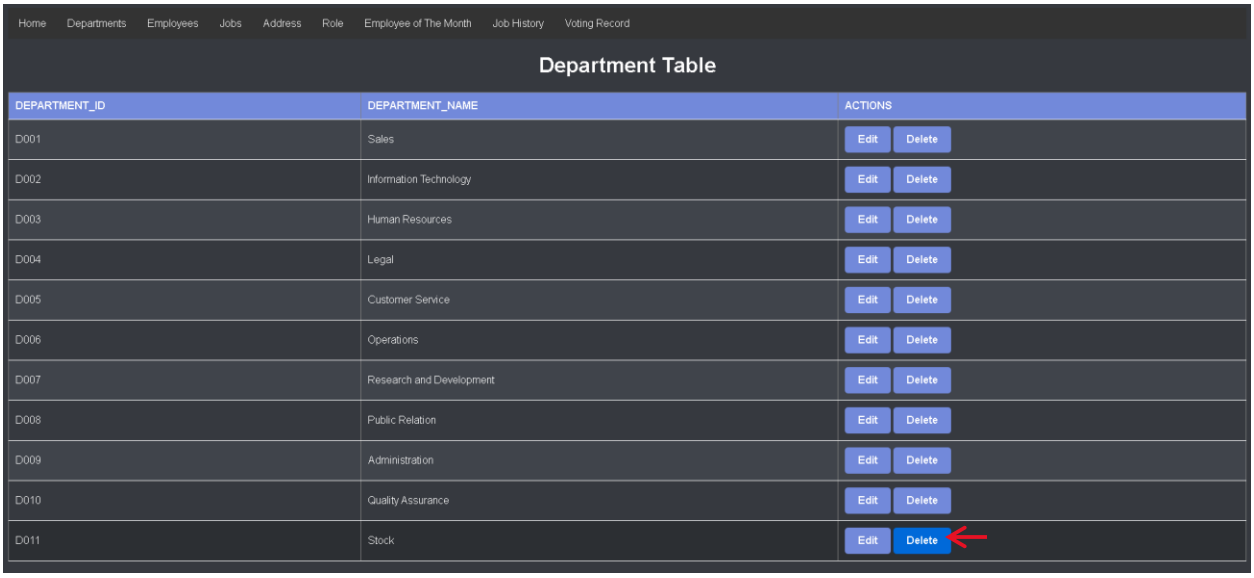


Figure 88: delete button pressed.

Result:

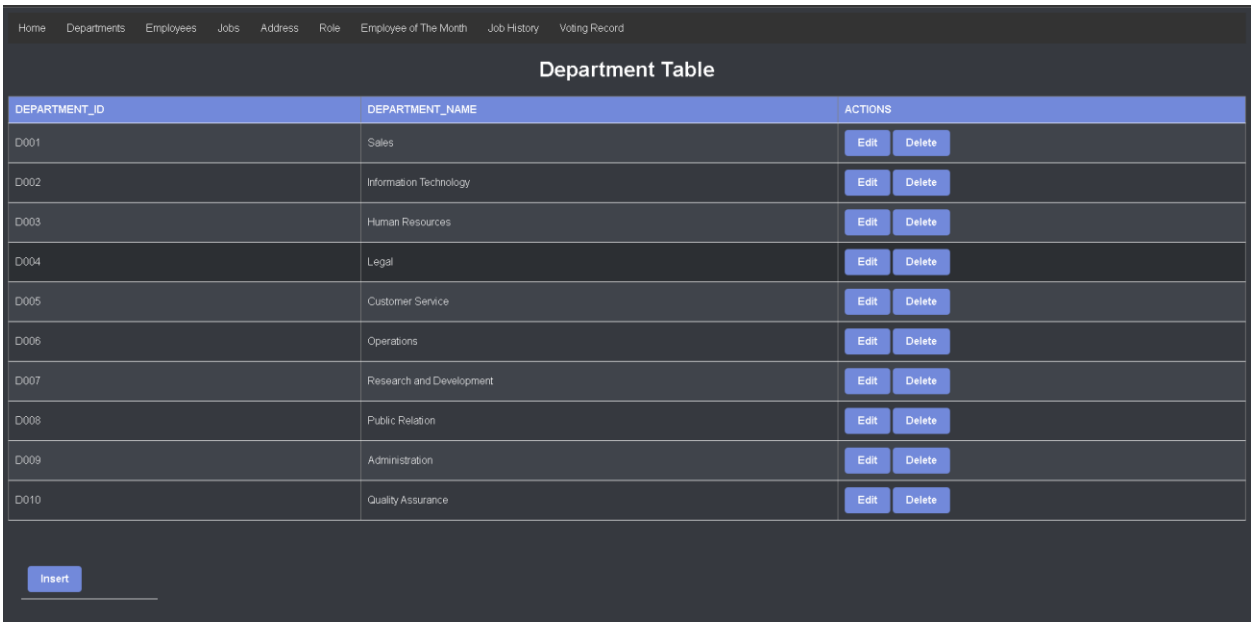


Figure 89: item deleted.

13.6 Employee of the month data filter

Objective	To test drop down menu.
Action	Selecting both year and month from drop down.
Expected Outcome	The web page will refresh, and new data will be displayed
Actual Result	The data inside the Employee of the month Grid View was changed.
Conclusion	Test Successful

Action:

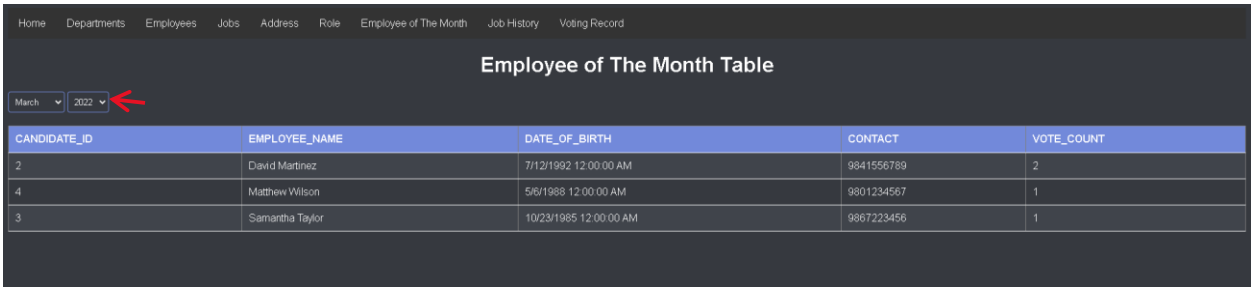


Figure 90: select drop down

Action:

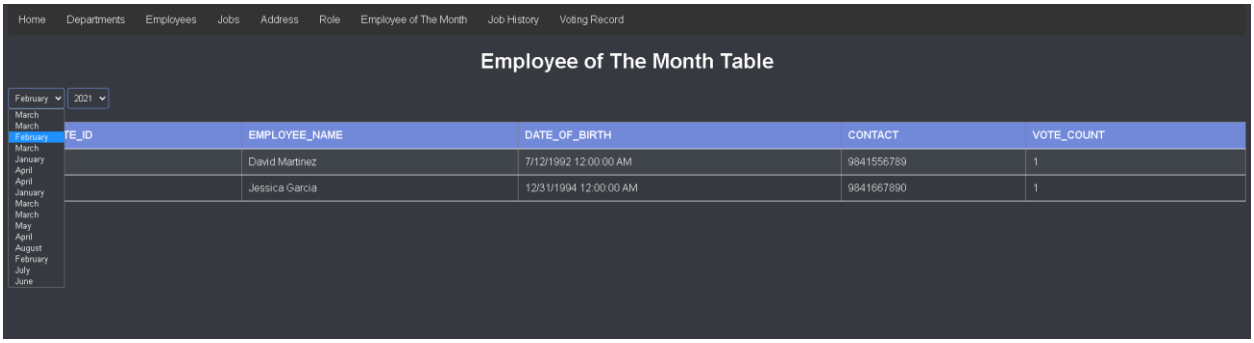


Figure 91: dropdown activated.

Result:

Figure 92: date filtered

13.7 Job History data filter

Objective	To test drop down menu.
Action	Selecting employee from drop down.
Expected Outcome	The web page will refresh and show record of selected employee.
Actual Result	The data inside the Job History Grid View was changed.
Conclusion	Test Successful

Action:

Figure 93: Select drop down employee.

Action:

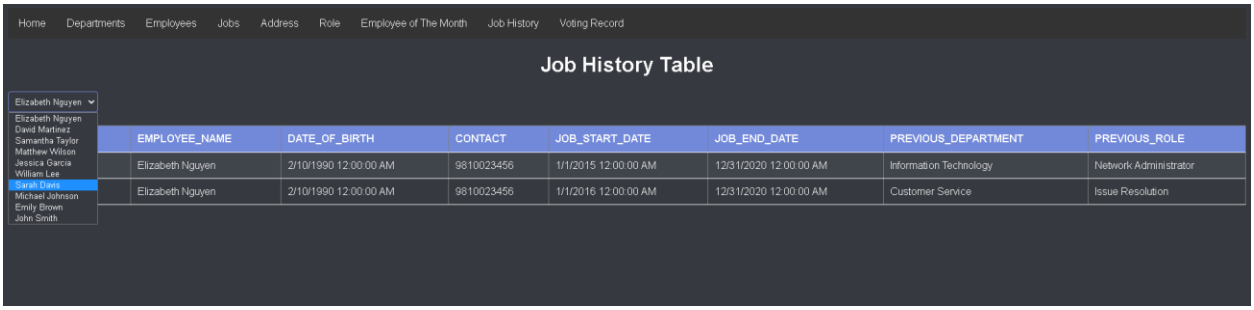


Figure 94: Select option.

Result:

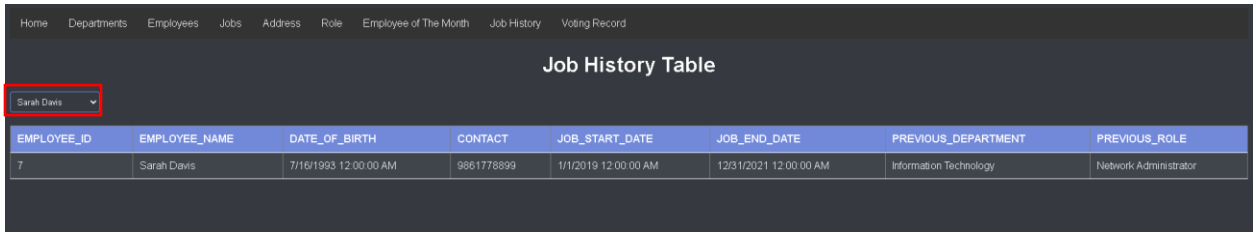


Figure 95: value filtered.

13.8 Voting record data filter

Objective	To test drop down menu.
Action	Selecting employee from drop down.
Expected Outcome	The web page will refresh, and employee record will be displayed.
Actual Result	The data inside voting record was changed.
Conclusion	Test Successful

Action:

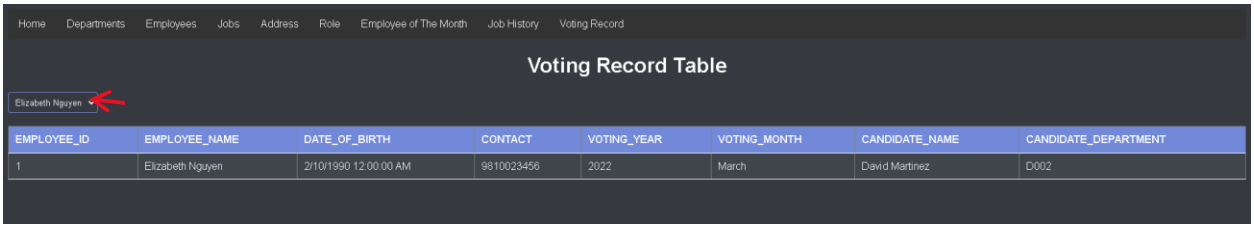


Figure 96: Select dropdown.

Action:

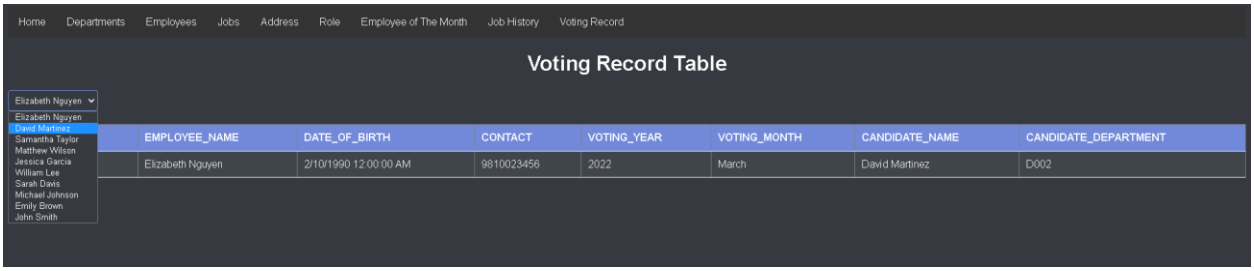


Figure 97: select item from list

Result:

Figure 98: item shown

14. Further Discussion

Upon building the online voting system for the organization the overall concept of planning to designing the database was done and learned a lot of things along the way. This application has benefited me with a lot of extensive information. The scenario had different parts from designing the database to executing to coding. The normalization was the steppingstone to this project upon completing normalization I got proper understanding about the scenario It was hard at times but upon completing the challenging task it has made me better at database management. The course work was a great opportunity to implement all our knowledge to solve the given scenario some of the tools got to learn are as follows.

- **Oracle data modeler**

I learned how to create ERD diagrams using the data modeler. This tool was used to design the final ERD and after that a DDL script was generated containing all the database tables that was present in the ERD.

- **Oracle sql developer**

Another important tool I used was Oracle sql developer it is an IDE which is offered by oracle Corporation. This helped in generation of database and inserting values. It saved me a lot of time it was mainly used to create table and insert values.

- **Visual studio 2019**

Visual studio code was used to create the whole webform where there was different option to design our form first, we connected the database with visual studio then the simple form, complex form and dashboard was generated accordingly. It made creation of web form a lot easier and productive.

- **Oracle Express 11g database**

I also learned to user oracle express it is the same database that many businesses rely on due to it being robust, cost-free, and easy to deploy. The oracle database was used by the whole project as a foundation.

- **Normalization and ERD**

Also learned how to normalize tables to 3nf in a systematic manner normalization is used to eliminate data redundancy and keep the data integrity. In this process inconsistent tables are separated into smaller ones. At first the database for the course work was redundant later we normalized it into proper tables till 3nf. After the normalization final ERD was made keeping the normalization in mind.

- **ASP.NET WEB APP (WEB FORM)**

It is a framework offered by Microsoft which was a part of the project. Web app was built using the .net framework for web application development.

15. Conclusion

In conclusion the entire course work has helped me to advance my skills in normalization, database management and creating web forms. I learned to use different tools while designing the database. I also aided me in improving my skills with dealing problems, fixing it, and creating a fully fledge application. Upon encountering many errors, a lot of research was done to pinpoint the problem and work on it not only it helped my debugging skills I also improved my research skills.

During the coursework I normalized an inconsistent database till 3nf where we eliminated data redundancy and abnormalities then we created ERD using data modeler and generated a DDL script. Proper tables were created using SQL developer and its proper values were inserted. After that a web form was established to perform CURD operations and proper complex forms. Upon completion all the coursework requirements were completed properly, and a fully functional system was established with about 9 active forms including dashboard. One of the problems I encountered at was SQL data source not working due to visual studio 2022 being installed I had to re install 2019 for it to work.

Upon completion of the system various test cases was designed to find any bugs or critical problems. The system is also well documented with proper manual and description. Overall, the database is well designed and is free of inconsistencies and redundancies.