

**CC6001NI Advanced Database System Development**

**40% Individual Coursework**

**Student Name:**

**London Met ID:**

**College ID:**

**Assignment Due Date:**

**Assignment Submission Date:**

**Word Count:**

*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.*

Contents

[1. Introduction 3](#_Toc127693361)

[2. Textual Analysis 3](#_Toc127693362)

[3. ERD from Case Study 3](#_Toc127693363)

[4. Normalization 4](#_Toc127693364)

[4.1 Normalization of fig 1 4](#_Toc127693365)

[4.2 Normalization of fig 2. 4](#_Toc127693366)

[5. Integration and Assumption 5](#_Toc127693367)

[6. Final ERD 6](#_Toc127693368)

[7. Data Dictionary 7](#_Toc127693369)

[8. Script 8](#_Toc127693370)

[9. Insert Statement 9](#_Toc127693371)

[10. Select Statement 10](#_Toc127693372)

[11. Forms 11](#_Toc127693373)

[11.1 Dashboard or Home Page 11](#_Toc127693374)

[11.2 Complex Form and Queries 12](#_Toc127693375)

[11.2.1 SQL Queries 12](#_Toc127693376)

[11.2.2 Complex Forms 12](#_Toc127693377)

[11.3 Simple Form 12](#_Toc127693378)

[12. User Manual 13](#_Toc127693379)

[13. Testing 14](#_Toc127693380)

[14. Further Discussion 16](#_Toc127693381)

[15. Conclusion 17](#_Toc127693382)

[References 18](#_Toc127693383)

**Abstract**

This individual coursework, Oracle SQL Developer Data Modeler and ASP.NET with C#, was presented in the eighth week of the fifth semester. The coursework comprises of evaluating, developing, and developing a web-based database application based on a specified business case study in this project, to establish online voting system to choose employee of the month. The application that will be created offers a variety of features and functionalities. We were required to deliver a software solution as well as documentation outlining the system's design and implementation for the project. The system will have several characteristics that will be created. For the organization ,a suitable web form should be created.

# 1. Introduction

Introduction to your Report.

To enable simple access to employees located all over the world, many companies are turning to Web-based applications. You have near-universal access once you're on the Internet; all you need is an Internet connection and a Web browser. Here, we are required to evaluate, develop, and develop a web-based database application based on a provided business case study in this project. We've been asked to provide a software solution as well as documentation documenting the system's design and implementation.

The goal of this project is to create a web-based database application using the C# programming language and an Oracle database. We've provided a scenario in which an organization is planning to establish online voting system to choose employee of the month. Employees are not allowed to vote themselves also one employee can work on one department only but allowed to vote an employee from any department. There is different role of employees in different department and everyone is eligible to vote. In Organization, each department have their own manager who are also an employee of that organization and organization keep the record of employee job history if any employee changes his/her department and job. Using Oracle SQL Developer Data Modeler and ASP.NET with C#, we create a system prototype. ERD was created using a database table data modeler software. SQL Developer was used to populate the table by inserting values and running the script. Visual studio 2022 community edition was used to create the web pages, which were written in the C# programming language. An appropriate web form has been developed for the organization service to the system.

# 2. Textual Analysis

Provide the textual analysis from the case study.

For example: -



Figure 1 Textual Analysis of Employee and Department

Description: One employee is assigned to either one department or none. One department can have at least one employee or many employee.

**Note**: Create textual analysis for every relationship identified from the Case study.

# 3. ERD from Case Study

Provide your initial ERD created from the case study

# 4. Normalization

## 4.1 Normalization of fig 1

Normalize fig1 with proper identification and representation of the repeating group, partial dependency and transitive dependency and **Keys as well.**  If any entities automatically transform from one normal form to another normal form, then do mention why that/those entities automatically transform.

**Note: - List all the final entities before moving to the next Normal form.**

**Normalization**

Figure 1

**UNF**

Here listing all the columns of the above table within the curly braces, and separating the repeating and non-repeating attributes with other curly braces we get,

Employee {sn, employee \_name, dob, contact, {email}, {address}, department\_id, department \_name}

**1NF**

In 1NF, the repeating and non-repeating attributes are separated into different tables with suitable labeling we get three tables as mentioned below,

Employee {sn, employee\_name, dob, contact, department\_id, department \_name}

Address {address\_id, sn\*, address}

Email {email, sn\*}

**2NF**

There aren't any partial functional dependencies in 'Employee' and 'Email' table. But Address' table has partial dependency which is being removed below.

Address {address\_ id, sn\*, address}

address > address id

so,

Address {address\_id, address}

Address\_Relation {address id, sn\*}

Since, there aren't any partial functional dependencies in 'Employee' and 'Email\* table, there won't be any changes within these tables.

Employee {sn, employee\_name, dob, contact, department\_ id, department\_name}

Email {email, sn\*}

**3NF**

Among all the above tables, the transitive dependency only exits in 'Employee' tables. Employee {sn, employee\_name , dob, contact, department\_ id, department\_name}

sn -> employee\_name

sn -> dob

sn -> contact

sn -> department id

department id -> department\_name

Here we can see, sn -> department\_id but department\_ id -> department name which defines transitive functional dependency which must be removed to achieve full functional dependency. Thus,

Employee {sn, employee \_name, dob, contact, department\_id\*}

Department {department\_id, department\_name}

Now the remaining tables are:

Address {address id, address}

Address\_Relation {address\_id\*, sn\*}

Email {email, sn\*}

## 4.2 Normalization of fig 2.

Follow the same process.

Figure 2

**UNF**

From the above table, for UNF, listing all the attributes within curly braces and then separating the repeating and non-repeating attributes with other curly braces we get,

Vote {voter\_id, voter name, {voting\_year, voting month, candidate\_id, candidate\_name, candidate\_department}}

**1NF**

For 1NF, all the repeating and non-repeating attributes are separated into different tables and give below as,

Voter {voter\_id, voter \_name}

Vote {voting\_year, voting\_month. voter\_id\*, candidate id, candidate\_name, candidate\_ department, department\_id}

**2NF**

For 2NF, we have to remove the partial functional dependency to achieve full functional dependency. Unfortunately, in the above tables there are not any partial functional dependency. Thus, all the tables will be same as 1NF, which are:

Voter {voter\_id, voter \_name}

Vote {voting\_year, voting\_month. voter\_id\*, candidate id, candidate\_name, candidate\_ department, department\_id}

**3NF**

To obtain 3NF, transitive functional dependency must be removed. The 'Vote' table seems to have transitive functional dependency as:

Voting\_year, voting\_month , voter\_id -> candidate\_id

Voting\_year, voting\_month , voter\_id -> candidate\_department\_id

Candidate\_id -> candidate\_name

Department\_id -> candidate\_department

Here, candidate\_ department\_id and candidate\_id depends on the composite key voting year, voting\_month, voter\_ id while candidate\_name depends on candidate\_ id and candidate \_department depends on department \_id. Removing these transitive dependencies, we get the following tables.

Voter {voter \_Id, voter \_name}

Candidate {candidate\_ id, candidate\_name}

Department {candidate\_department\_id, candidate \_department}

Vote {voting\_year, voting\_month, voter\_id\*, candidate\_ id\*, candidate \_department\_id\*}

# 5. Integration and Assumption

Combine all the entities derived from the case study, normalization of fig 1 and fig 2. Provide any assumptions that you have made for the integration of entities.

**Note: Be careful while integrating and make sure there should not exist many to many relationships and should not miss the relationship that exists in your initial ERD. At last, provide overall entities with attributes and proper key representation.**

# 6. Final ERD

Provide final ERD created from **SQL Data Modeler.**

**Note: Make sure your relationships line is not intersected in your Final ERD.**

**Use PROPER CARDINALITY {MENDATORY AND OPTIONALITY} AND ALSO USE appropriate Constraint for each column {UNIQUE CONSTRAINT, CHECK CONSTARINT, NOT NULL CONSTRAINT, DEFALUT CONSTRAINT ETC.}**

# 7. Data Dictionary

create a data dictionary for all entities following the given template

for eg: -

Table: **Employee**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Size** | **Constraint** | **Reference Table** | **Reference Column** | **Description** | **Example Data** |
| Emp\_id | Varchar | 10 | Primary Key |  |  | To uniquely identify Each employee | Emp1 |
| Emp\_name | Varchar | 50 | Not null |  |  | To store the employee’s name | Ram Sharma |
| Email | Varchar | 50 | Unique |  |  | To store employee’s email | ram@gmail.com |
| Dep\_id | Varchar | 10 | Foreign Key | Department | Dep\_id | To store the employee’s allocated department | D1 |

# 8. Script

Generate DDL script from Data Modeler and provide that script here. Just copy-paste the DDL script. And provide the screenshot of script execution in SQL Developer.

# 9. Insert Statement

Provide the screenshot for the execution of the insert statement through SQL developer for every entity.

For example:

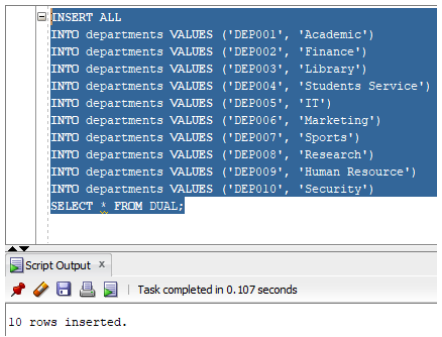


Figure Insertion into Department Table

**Note: - Do not forget to execute the commit statement after completing the insertion otherwise you will lose all the inserted data when you restart your Database.**

# 10. Select Statement

Provide the screenshot for the execution of select statements through SQL developer for every entity.

For eg:

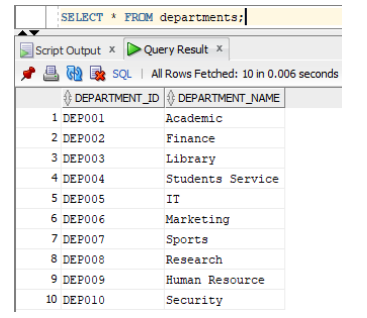


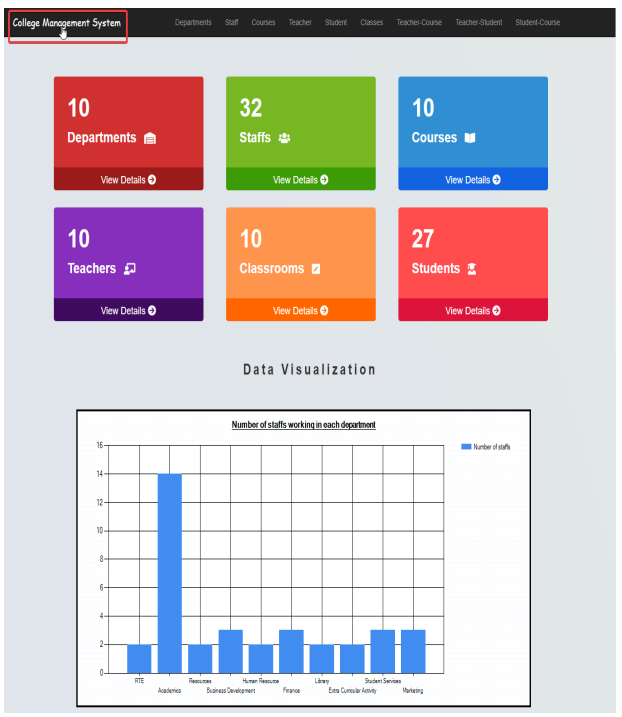
Figure Select statement for Department

# 11. Forms

## 11.1 Dashboard or Home Page

Provide a screenshot of your dashboard.

Eg:



## 11.2 Complex Form and Queries

### 11.2.1 SQL Queries

Provide all the sql queries for each complex forms.

### 11.2.2 Complex Forms

Provide a screenshot of all complex forms.

## 11.3 Simple Form

Provide a screenshot of all basic forms.

# 12. User Manual

Provide the user manual for all the forms with arrows and graphics to explain the process.   
**Note: - Use graphical representation rather than explaining in detail for better understanding.**

# 13. Testing

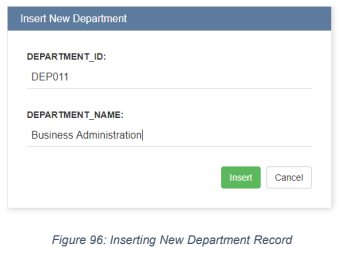
For each form implemented, list the individual tests that have been carried out together with their results.

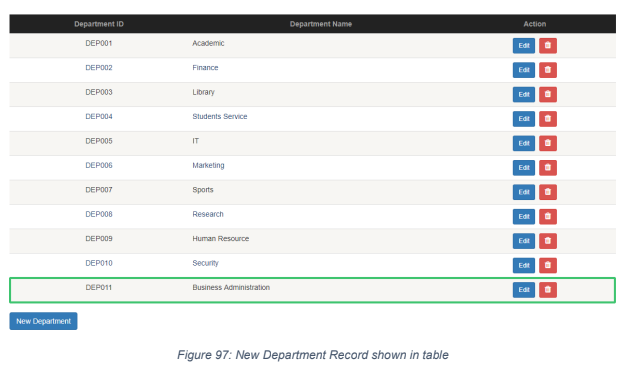
For basic forms, perform CRUD operation.   
For basic forms, perform data filter through the dropdown.

Proper test cases with Before, During and After Screenshots of data.

For example: -







Follow the same procedure for update and delete as well for every basic web form.

**Note: - There should be at least 2 failure test cases with correction measures.**

# 14. Further Discussion

Your discussion should summarise your experience in undertaking this coursework with the mention of 5 tools/techniques learned during coursework.

# 15. Conclusion

# References