

# Employee Performance Management System

Lam Nguyen – ID: 500838417

CPS510: Database Systems I

Section - 01

Instructor: Dr. A. Abhari

Ryerson University

## Table of Contents

<b><i>Introduction</i></b>	<b><i>3</i></b>
<b><i>Entity-Relationship Diagram</i></b>	<b><i>4</i></b>
<b><i>Database Schema with Normalization &amp; Functional Dependencies</i></b>	<b><i>5</i></b>
<b><i>Database Queries (SQL &amp; Relational Algebra)</i></b>	<b><i>7</i></b>
<b><i>UNIX Shell Implementation</i></b>	<b><i>9</i></b>
<b><i>GUI</i></b>	<b><i>11</i></b>
<b><i>Conclusion</i></b>	<b><i>15</i></b>

## Introduction

### Description

In the world today, employees are the most significant assets of an organization, so managing them properly helps to ensure successful performance appraisal and bring benefits to the system. However, difficulties in managing a large repository of employee data manually pose many challenges. To be more specific, it can be time-consuming, error-prone, and especially problematic for security systems. Thanks to employee performance management systems, handling and organizing data becomes much easier. Generally, it is a systematic approach to evaluate the performance of employees through which the organization can align their goals and objectives with available resources.

### Basic Function

An employee performance database management system has multiple beneficial uses for any company. From keeping track of all the employees within the company to storing evaluations of all employees, this system is necessary for a company to function efficiently. This DBMS has multiple functions including: listing/creating information about an employee, retrieving/updating the evaluation of an employee, finding which employees are in a certain department, displaying employees working under a certain manager, and many more.

To list information of an employee, searching for the employee\_id or fullname (as long as there are no two employees with the same name) is the easiest way. The information listed will be the employee's id, fullname, current position, assessment, etc. To add a new employee to the database, a new employee\_id needs to be created for that employee while adding the additional necessary information for the employee. To retrieve an employee's evaluation, search for the employee's id and display the result, and to update, search for the employee and replace the new evaluation with the old one. Finding employees working for a specific department requires searching for the department in the database and listing all employee ids associated with it. Similarly, the DBMS can search for a certain manager and display all employees working under them. A myriad amount of functions can be created for the employee database management system.

### Expectations

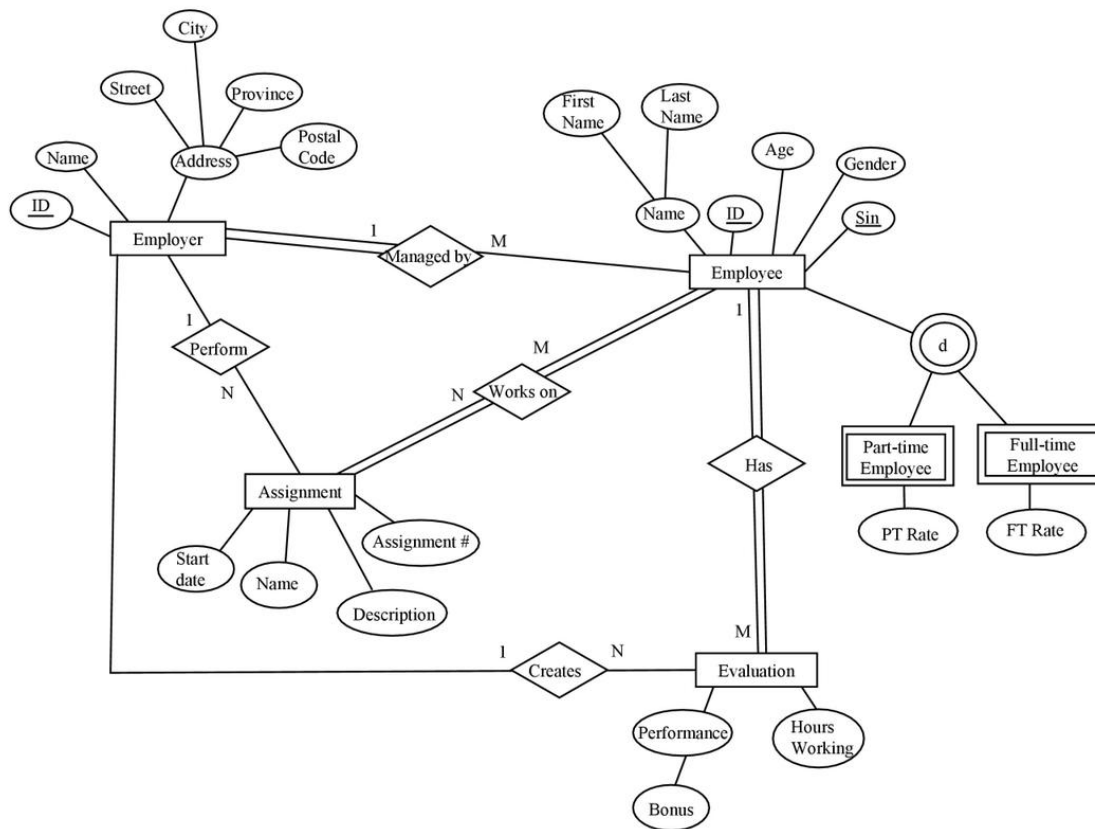
The information expected from the employee performance application could be segmented into three different database tables: employee's information, employee's assessment, employee's goals, and employer's signoff.

The employee's information table would store the employee's first and last name, identification number, position held, department, manager's name, and date of assessment. Of all the attributes, the employee's identification number is the primary key attribute because it is unique. As noted, storing an employee's information is the first crucial step to assessing their performance. Once the employee's information is recorded, the next step is to perform their assessment. The employee's assessment table would store the employee's identification number

along with assessment categories that they would be evaluated on, such as: communication, quality of work, integrity, initiative, and leadership. These criteria will be ranked in terms of “needs improvement, satisfactory, good, and excellent”. And finally, the employee’s signoff database table would contain the employee’s identification number, employee’s goals and achievements, and employer’s comments/reviews.

In this case, the database table relationship could be one-to-many. This means each employee could have many assessments and goals created, and this might be semi annually or quarterly. The primary key field in the information table, employee’s identification number, is designed to contain unique values. The foreign key field in the assessment and goals table is designed to allow multiple instances of the same value.

## Entity-Relationship Diagram



## Database Schema with Normalization & Functional Dependencies

### Employer Table

```
CREATE TABLE Employer(  
    EmployerID      NUMBER NOT NULL,  
    EmployerName    VARCHAR2(50) NOT NULL  
    StreetNo        NUMBER CHECK (StreetNo BETWEEN 1 AND 9999),  
    StreetName      VARCHAR2(25) NOT NULL,  
    City            VARCHAR2(25) NOT NULL,  
    Province        VARCHAR2(2) NOT NULL,  
    PostalCode      VARCHAR2(10) NOT NULL,  
    PRIMARY KEY     (EmployerID)  
);
```

- Functional dependencies for employer table:

EmployerID -> EmployerName, StreetNo, StreetName, City, Province, PostalCode, Address

This is not in 3NF. We first must make this 2NF into 3NF because as shown in the graph, there are non-candidate-key attribute that is transitively dependent on the candidate key. Now making it into 3NF, we have:

EmployerID -> EmployerName, PostalCode

PostalCode -> StreetNo, StreetName, City, Province

### Employee Table

```
CREATE TABLE Employee(  
    EmployeeID      NUMBER NOT NULL,  
    EmployeeFirstName VARCHAR2(25) NOT NULL,  
    EmployeeLastName VARCHAR2(25) NOT NULL,  
    Age            NUMBER CHECK (Age BETWEEN 18 AND 70),  
    Gender         VARCHAR2(25) NOT NULL,  
    SIN            NUMBER NOT NULL,  
    EmployeeType    VARCHAR2(25) NOT NULL,  
    Rate           DECIMAL CHECK (Rate >= 14.35),  
    EmployerID     NUMBER,  
    PRIMARY KEY     (EmployeeID, SIN),  
    FOREIGN KEY     (EmployerID) REFERENCES  
    Employer(EmployerID)  
);
```

- Functional dependencies for employee table:

EmployeeID, SIN -> EmployeeFirstName, EmployeeLastName

EmployeeID, SIN -> Age

EmployeeID, SIN -> Gender

EmployeeID, SIN -> EmployeeType  
EmployeeType -> Rate

This is not in 3NF. We must first make this into 2NF since there is partial dependency for the EmployeeID and sin. Now making it into 2NF, we have:

EmployeeID -> EmployeeFirstName, EmployeeLastName, Age, Gender, EmployeeType  
SIN -> EmployeeType, Rate

Since as we can tell in our graph there is a transitively dependency, now we have transform our 2NF to 3NF:

EmployeeID -> EmployeeFirstName, EmployeeLastName, Age, Gender  
SIN -> EmployeeType  
EmployeeType -> Rate

### Assignment Table

```
CREATE TABLE Assignment(  
    AssignmentNo          NUMBER NOT NULL,  
    AssignmentName        VARCHAR2(50) NOT NULL,  
    AssignmentDescription  VARCHAR2(255),  
    StartDate             DATE NOT NULL,  
    EmployerID            NUMBER,  
    EmployeeID            NUMBER,  
    SIN                   NUMBER,  
    PRIMARY KEY           (AssignmentNo, AssignmentName),  
    FOREIGN KEY           (EmployeeID, SIN) REFERENCES  
Employee,  
    FOREIGN KEY           (EmployerID) REFERENCES Employer  
);
```

- Functional dependencies for assignmnet table:

AssignmentNo, AssignmentName -> AssignmentDescription, StartDate

This is not in 2NF because there is no full dependency. Therefore, to transform it to 2NF, we get:

AssignmentNo -> StartDate  
AssignmentDescription -> AssignmentDescription

This is then in 3NF because there is no non-candidate-key attribute that is transitively dependent on any candidate key.

### Working PerformanceEvaluation

```

CREATE TABLE PerformanceEvaluation(
    WorkingHours          NUMBER CHECK (WorkingHours BETWEEN 1 AND
100),
    Assessment            CHAR NOT NULL,
    Bonus                 NUMBER,
    EmployerID            NUMBER,
    EmployeeID            NUMBER,
    SIN                   NUMBER,
    PRIMARY KEY            (Assessment),
    FOREIGN KEY            (EmployerID) REFERENCES Employer,
    FOREIGN KEY            (EmployeeID, SIN) REFERENCES Employee
);

```

- Functional dependencies for Working PerformanceEvaluation table:  
WorkingHours, Assessment -> Bonus

This is in 2NF because there is fully functional dependency. We cannot remove either working hours nor assessment. It is therefore also 3NF because there is no transitively dependency.

## Database Queries (SQL & Relational Algebra)

### SQL:

```

SELECT * FROM Employer
WHERE City = 'Toronto';

```

### Relational Algebra:

$$\sigma_{city='Toronto'}(Employer)$$

### SQL:

```

SELECT Gender, EmployeeType
FROM Employee
WHERE Gender = 'Male'
AND EmployeeType = 'Parttime';

```

### Relational Algebra:

$$\pi_{Gender, EmployeeType} \sigma_{gender='Male'}(Employee) \text{ and } \sigma_{EmployeeType='Parttime'}(Employee)$$

### SQL:

```

SELECT * FROM Assignment
WHERE AssignmentNo = 1;

```

**Relational Algebra:**

$\sigma_{AssignmentNo=1}(Assignment)$

**SQL:**

```
SELECT EmployeeID, EmployeeFirstName
FROM Employee
WHERE EmployeeFirstName <> 'Sarah';
```

**Relational Algebra:**

$\pi EmployeeID, EmployeeFirstName \sigma_{EmployeeFirstName \neq 'Sarah'}(Employee)$

**SQL:**

```
SELECT DISTINCT EmployeeID
FROM performanceevaluation
WHERE Assessment > 5
ORDER BY EmployeeID;
```

**Relational Algebra:**

$\pi EmployeeID (\sigma_{Assessment>5}(performanceevaluation))$

**SQL to create views:**

```
CREATE VIEW FemaleEmployees AS
SELECT Employee.EmployeeFirstName, Employee.EmployeeLastName,
Employee.Age, WorkingOn.AssignmentName
FROM Employee
FULL OUTER JOIN WorkingOn ON Employee.EmployeeID =
WorkingOn.EmployeeID
WHERE Gender = 'Female';
```

```
CREATE VIEW TopPerformmane AS
SELECT Employee.EmployeeFirstName, Employee.EmployeeLastName,
PerformanceEvaluation.WorkingHours, PerformanceEvaluation.Bonus
FROM PerformanceEvaluation
INNER JOIN Employee ON Employee.EmployeeID =
PerformanceEvaluation.EmployeeID
WHERE Assessment > 5;
```



# UNIX Shell Implementation

The screenshot shows a MobaXterm terminal window with a menu bar (Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, Help) and a toolbar. A 'Quick connect...' sidebar on the left shows a tree view of files under '/class-service/93nguye/'. The terminal pane displays the following output:

```
2) Create Tables
3) Populate Tables
4) Query Tables

X) Force/Stop/Kill Oracle DB

E) End/Exit
Choose:
1

SQL*Plus: Release 12.1.0.2.0 Production on Thu Oct 28 11:47:25 2021
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
drop_tables.sh: line 13: $'\r': command not found
```

The bottom status bar indicates 'UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>' and shows system information like '9°C Mostly cloudy' and '11:47 AM 2021-10-28'.

Figure 1: Drop Table

The screenshot shows a MobaXterm terminal window with the same interface as Figure 1. The terminal pane displays the following output:

```
2) Create Tables
3) Populate Tables
4) Query Tables

X) Force/Stop/Kill Oracle DB

E) End/Exit
Choose:
2

SQL*Plus: Release 12.1.0.2.0 Production on Thu Oct 28 11:47:41 2021
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL> 2 3 4 5 6 7 8 9 10
Table created.

SQL> SQL> 2 3 4 5 6 7 8 9 10 11 12 13
Table created.

SQL> SQL> 2 3 4 5 6 7 8 9 10 11 12
Table created.

SQL> SQL> 2 3 4 5 6 7 8 9 10 11
Table created.

SQL> SQL> 2 3 4 5 6 7 8 9
Table created.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
create_tables.sh: line 66: $'\r': command not found
```

The bottom status bar is identical to Figure 1, showing 'UNREGISTERED VERSION' and system information.

Figure 2: Create Table



## GUI

Web link: <https://apex.oracle.com/pls/apex/employeeedb/r/employee-performance-management-system/login?session=7076102400734>

Test Username: **tunglam030699@gmail.com**

Test Password: **tunglam030699**

For this project, GUI for employee performance management system was created using Oracle Application Express (APEX)

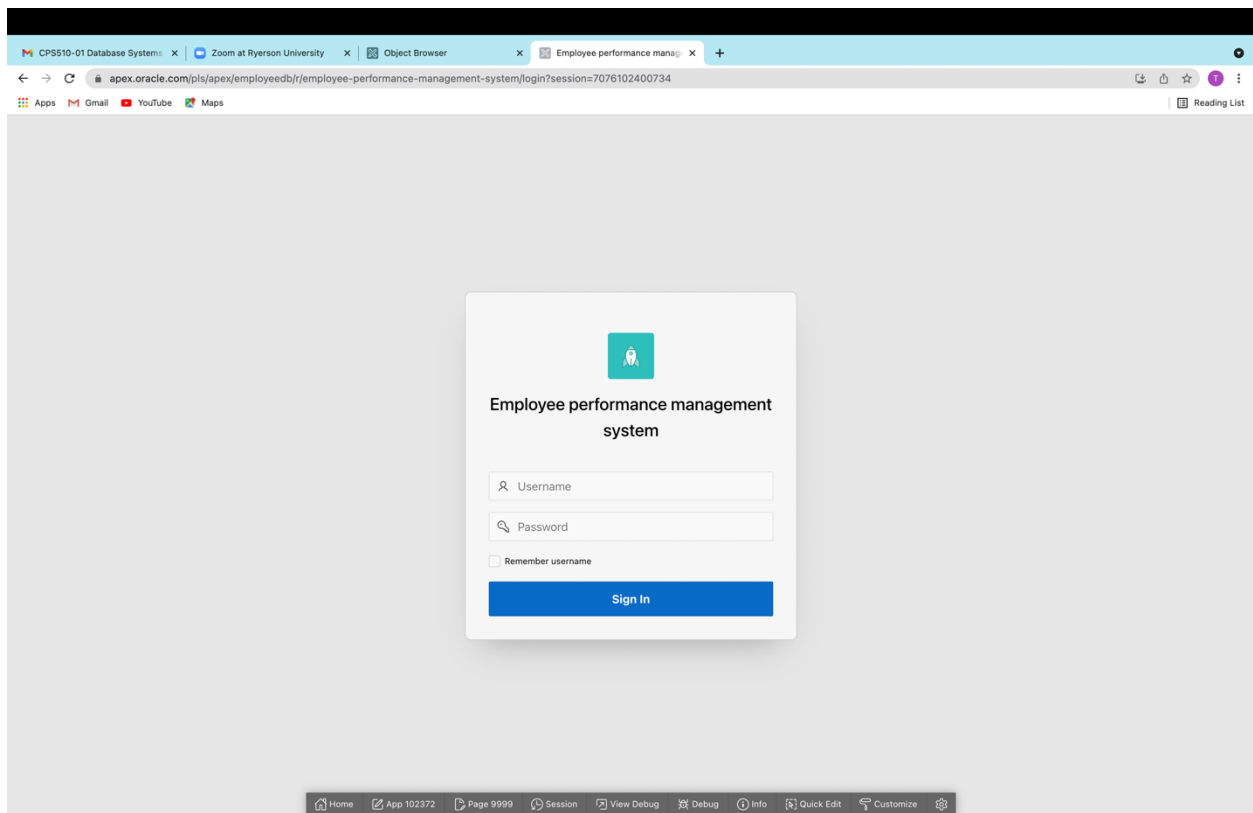


Figure 5: Login Page

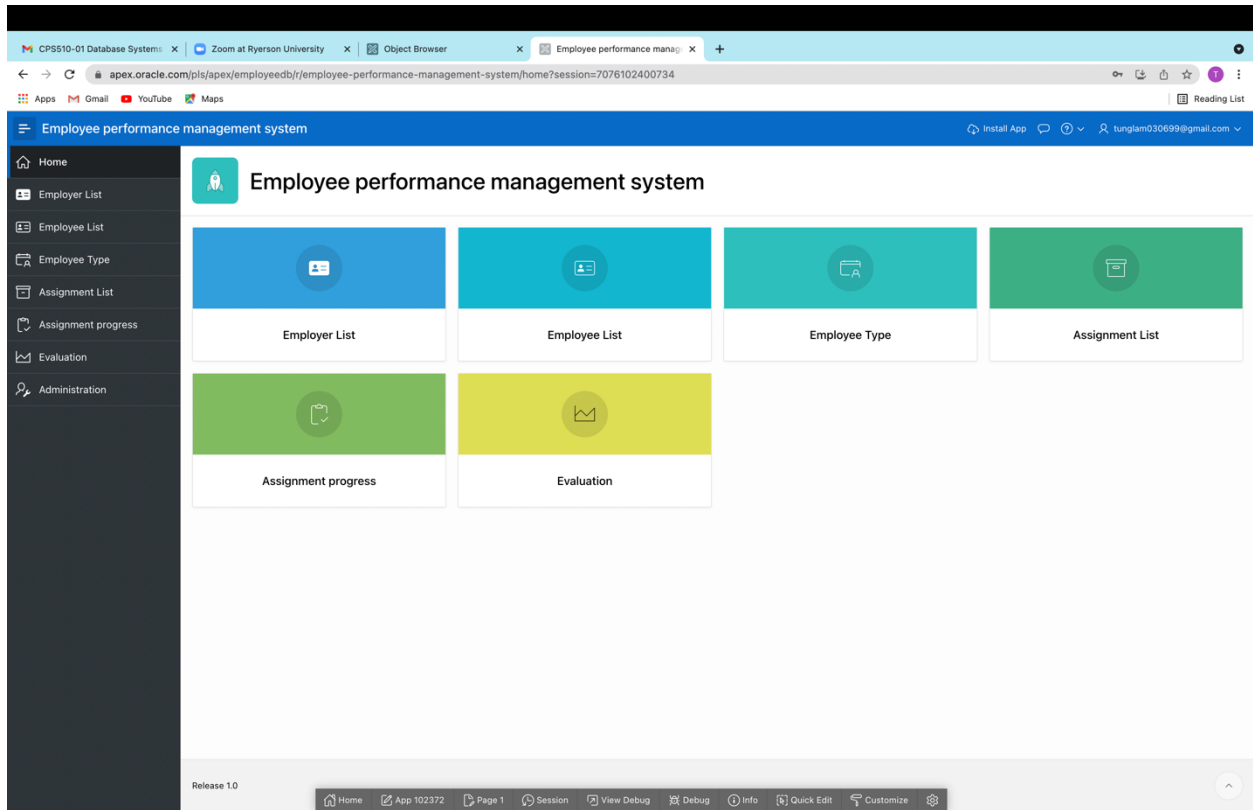


Figure 6: Dashboard

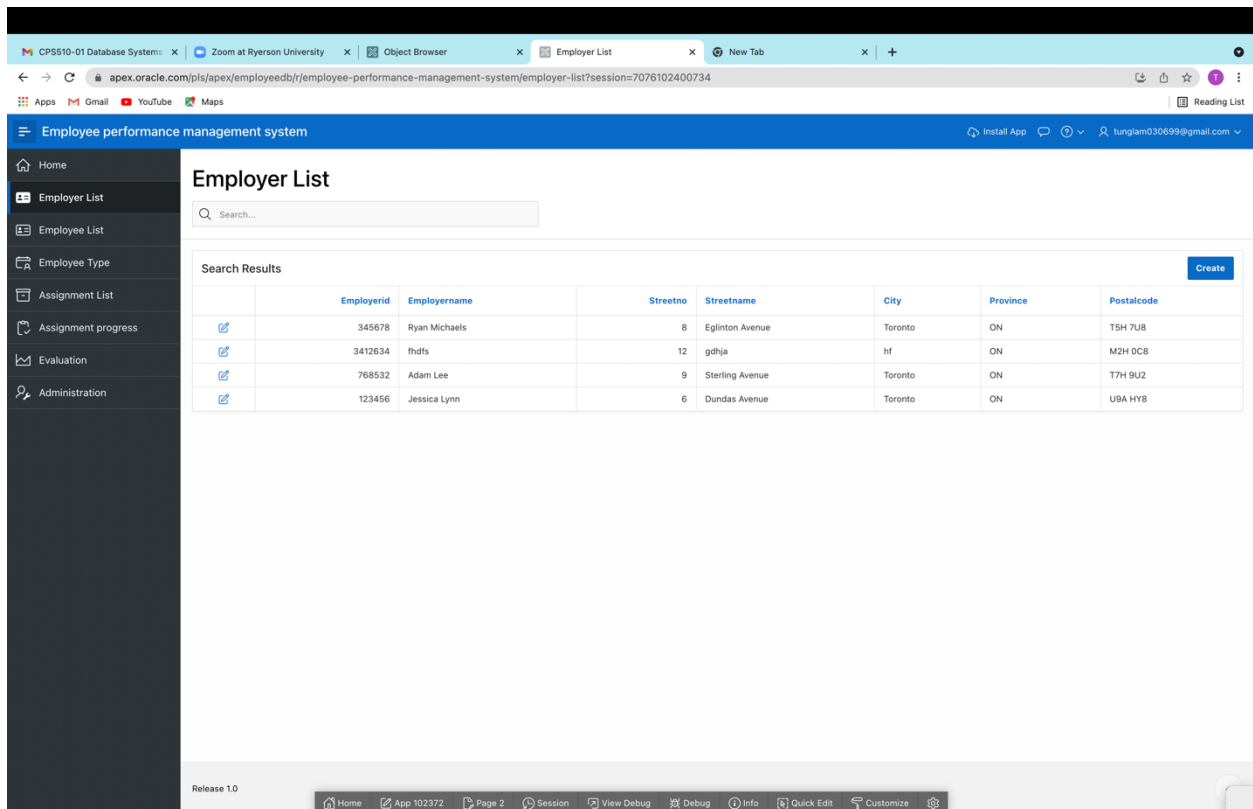


Figure 7: Employer List Page

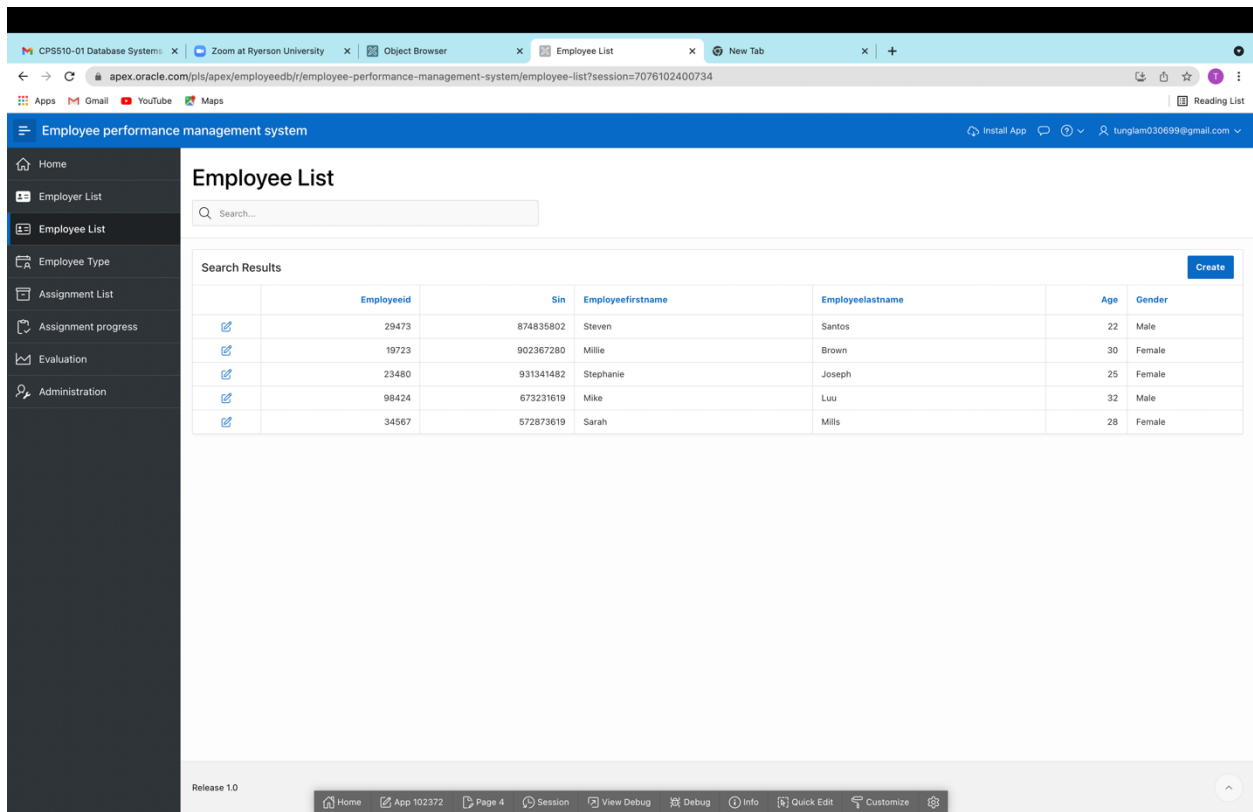


Figure 8: Employee List Page

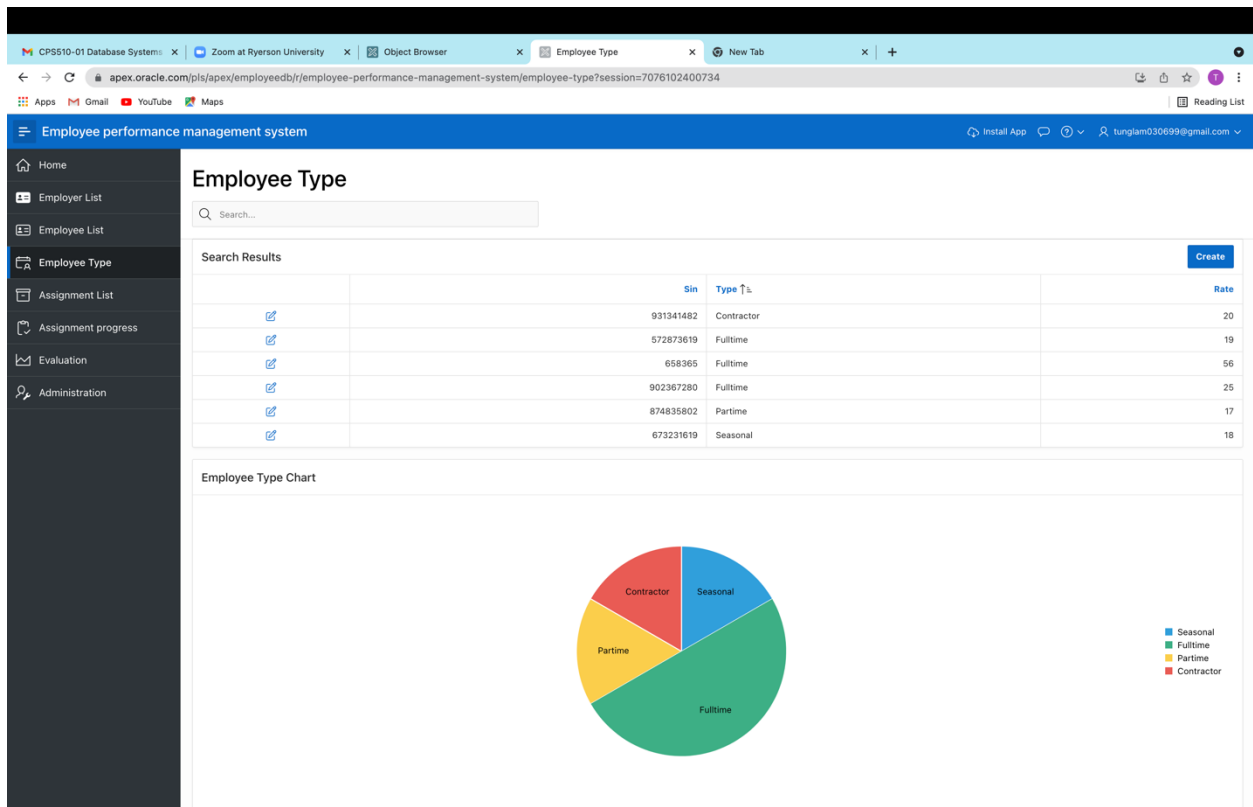


Figure 9: Employee Type Page

**Assignment List**

Search...

Search Results

	Assignmentno ↑	Assignmentname	Description
	1	Data pull	Pulling data into Excel
	2	Data analyze	Analyze data in PBI
	3	Data pipeline	Build a data pipeline in Azure
	4	Excel extraction	Use automation scripts to extract data in Excel
	5	SAP extraction	Pull data from the SAP systems
	6	Invoice pull	Pulling invoice data into Excel
	7	Invoice analyze	Analyze invoice data in PBI
	8	Invoice pipeline	Build an invoice data pipeline in Azure
	9	Invoice extraction	Use automation scripts to extract invoice data in Excel
	104	Invoice ssgfh	Segment invoice csgj

Release 1.0

Figure 10: Assignment List

**Assignment progress**

Search...

Search Results

	Assignmentno ↑	Startdate	Employeeid	Employerid	Progress
	1	11/9/2021	34567	345678	Finished
	2	10/20/2021	29473	123456	In Progress
	3	10/18/2021	19723	345678	In Progress
	4	11/4/2021	98424	768532	In Progress
	5	10/4/2021	23480	123456	Finished
	6	11/19/2021	34567	345678	In Progress
	7	9/23/2021	29473	123456	Finished
	8	11/1/2021	19723	345678	In Progress
	9	10/21/2021	98424	768532	In Progress
	10	11/15/2021	23480	345678	Finished

Release 1.0

Figure 11: Assignment Progress

## Conclusion

Working on this Employee Performance Management System has provided us with a solid foundation in all aspects of database design and implementation. Previously, we had not realized how important it was to represent data in a clear and concise way. With the theories we learned regarding entity-relationships diagrams, relational schema design, functional dependencies, normalization, etc., we were able to turn various pieces of data into a useful and accessible database.

On the technical side, we became accustomed to using SQL and the service provided by Oracle. Through this, we learned what it was like to create tables, drop tables, insert data and query information. Making a GUI using Oracle Application Express (APEX) also familiarized us with how a front-end interface connects and interacts with a back-end database.

This project also exercised our skills in teamwork, project management, and software development. In conclusion, it was a pleasure working on this database project since it truly allowed us to use the knowledge and skills acquired in this course.