|  |
| --- |
| [630-03 Database Management Systems & Design](https://stthomas.instructure.com/courses/57109) |
| Priyanka Namdev Patil |
| Name of the Project: Human Resources Management System |
| Module: Employee Management System |

****

“Human Resources Management System is a digital solution that combines several systems and processes to manage and optimize daily HR tasks and the overall HR goals of a given organization**”**

**Purpose:** The aim of this project is to create optimized and normalize skeleton of backend employee database. This backend database helps to create, manage, update and control data needed to run Human Resource department of any organization.

**Goal:** Employees are the backbone of any organization therefore employee data management plays a major role. The goal of the project is to design and create the database for the employee Management such as Demographics, payrates, personal information, organizational relationship, accruals, time management, compensation and performance etc. It serves as a backend database of OLAP and OLTP application for Human Resources department. This database helps to improve data efficiency, integrity, security and availability as compared to paper records. Role based access control can be implemented for employee data using this system i.e payroll should have access to only pay related information of employee. This database can be integrated with any front end and persistence layer to create efficient, data driven Human Resource Information System.

**Benefits:**

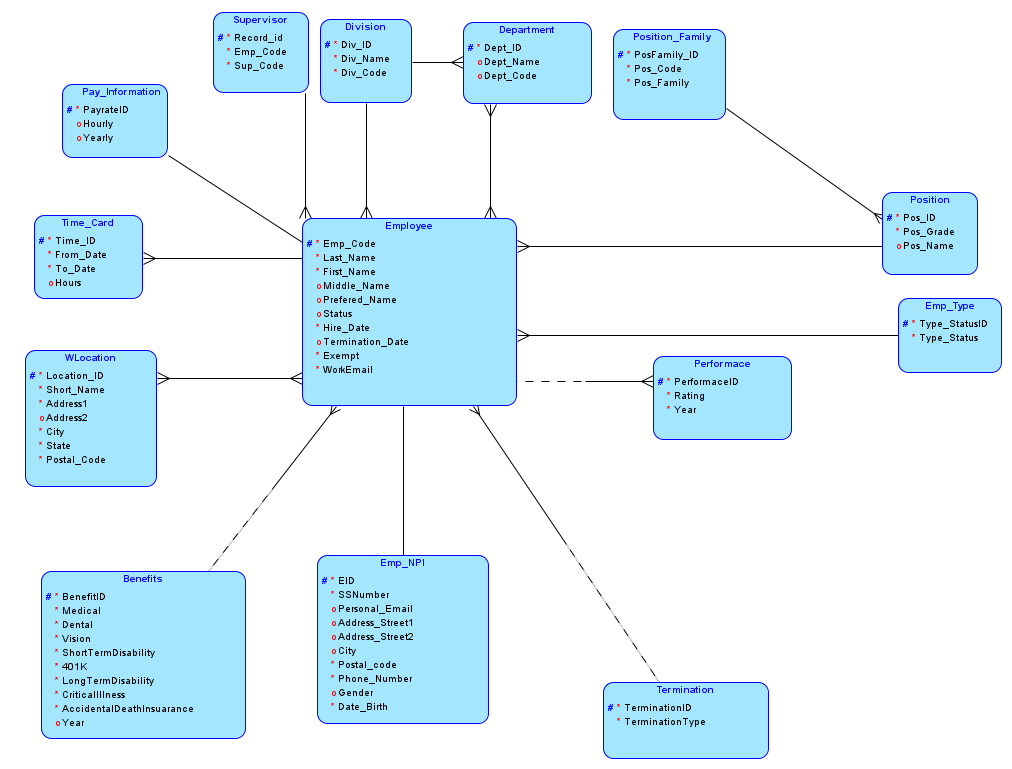
* More streamlined processes for employee data management
* Reduced data errors
* Employee tracking makes easy due to 24/7 data availability based on role
* Better and more accurate HR analytics
* Reduce overhead and wastages in HR operation

**Planning:**

We will create a logical model of HR system with the entities and attributes for different HR sub operations. We will create 1: 1, 1: N and N:M relationships between entities and define minimum and maximum cardinality between relationships. Will create constraints among the entities. We will create a relational view and then generate the DDL statements to create the skeleton of the database. We will add test data and assign and implement role-based security. At the end we will create the queries for different HR operations i.e., performance management, employee communication, compliance management etc.

**Development of HRIS:** After planning for the names for entities and relations used Data modeler to do forward engineering to create DDL statements. I started creating a logical diagram to generate entities and to create cardinalities among them.

**Logical Diagram:**



**Description:** Above logical diagram of HRIS system consist tables of 14 entities. All the entities cover major information of an employee within the organization.

Below are the attributes for all the entities.

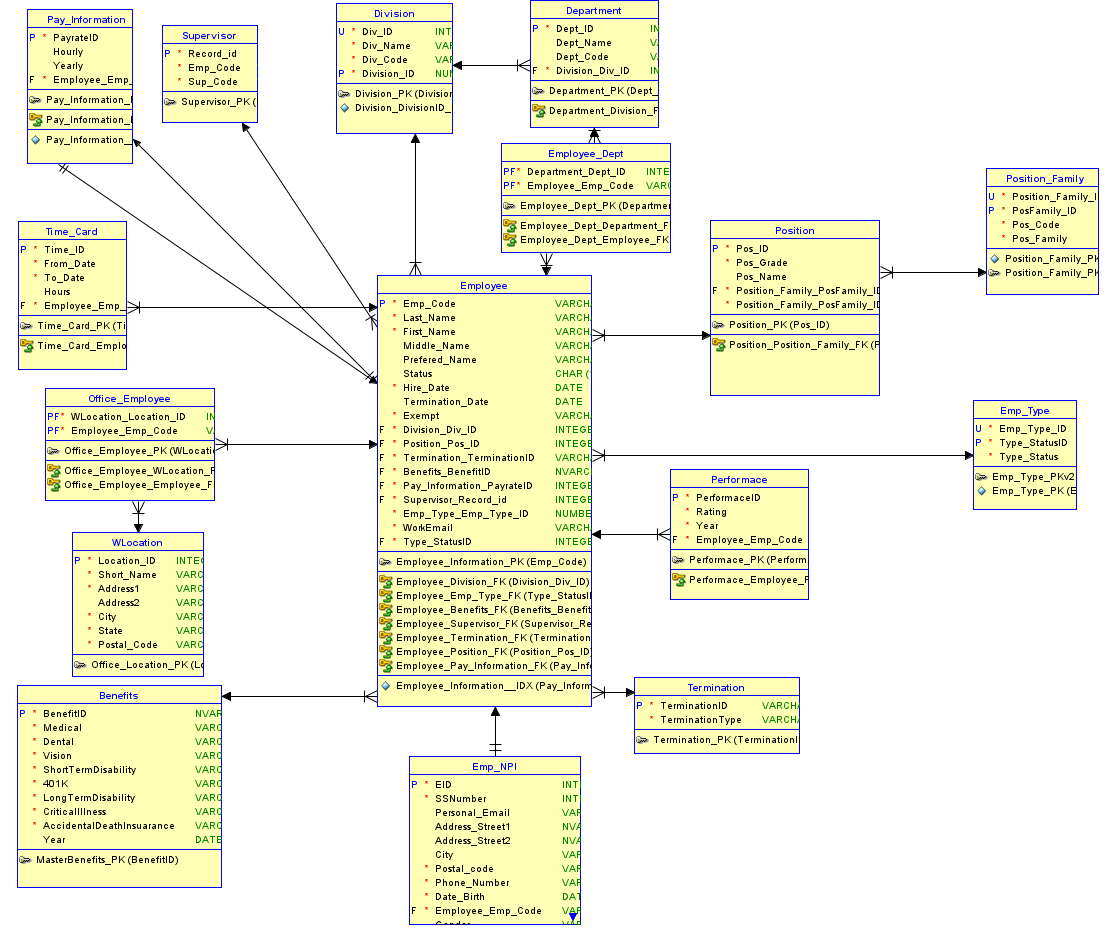
|  |  |  |
| --- | --- | --- |
| Sr No. | Entity | Attributes |
| 1 | Employee | Employee\_Code, Last\_Name, First\_Name, Middle\_Name, Preferred\_Name, Status, Hire\_Date, Termination\_Date, Exempt, WorkEmail |
| 2 | Emp\_NPI  (Employee Non-Public Information) | EID, SSNumber, Personal\_Email, Address\_Street1, Address Street2, City, Postal\_code, Phone\_Number, Gender, Date\_Birth |
| 3 | Office\_Location | Location\_ID, Short\_Name, Address1, Address2, City, State, Postal\_Code |
| 4 | Position\_Family | Position\_ID, Pos\_Code, Pos\_Family |
| 5 | Position | Position\_ID, Pos\_Grade, Pos\_Name |
| 6 | Emp\_Type | Type\_Status\_ID, Type\_Status |
| 7 | Division | DivisionID, Division\_Name, Division\_code |
| 8 | Department | Department\_ID, Department\_Name, Department\_Code |
| 9 | Supervisor | Record\_id, Emp\_Code, Sup\_Code |
| 10 | Pay\_Information | PayrateID, Hourly, Yearly |
| 11 | Time\_card | Time\_ID, From\_Date, To\_Date, Hours |
| 12 | Benefits | BenefitID, Medical, Dental, Vision, ShortTermDisability, 401K Later changed 401K name to F401K), LongTermDisability, CriticalIllness, AccidentalDeathInsuarance, Year ( |
| 13 | Performance | PerformanceID, Rating, Year |
| 14 | Termination | TerminationID, Termination Type |

In Employee & Benefit relation Benefits are optional because many organizations do not provide benefits until they complete30 days with the organization. Also, in Employee to Performance relation performance is optional because if employee has joined after September 30th, employees are not eligible for performance evaluation for the year

**Relations Between Entities:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No. | Entities | Relationship | Entity |
| 1 | Employee | **1:1** | EMP\_NPI, Pay\_Information |
| 2 | Employee | **1: N** | Position, Division, Emp\_Type, Performance, Termination, Timecard, Supervisor, Benefits |
| 3 | WLocation | **N:M** | Employee |
| 4 | Department | **N:M** | Employee |

**Relational Diagram:** Once the relationship has been established, we created a Relational table from logical View.



Relational diagram created two more tables i.e. **Associate tables**

|  |  |  |
| --- | --- | --- |
| 1 | Associate Tables | Office\_Employee & Employee Dept |

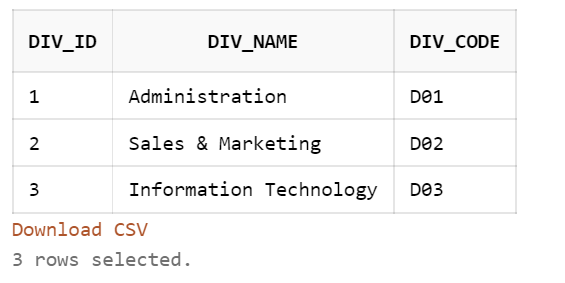
I generated DDL statements using forward engineering. 16 Tables, 3 Indexes and 35 alter Table statements were created. I also wrote some Triggers and Sequences for Div\_ID. Once this DDL statements are created, I verified them using Oracle Live SQL to check if queries were working.

Now it’s time to insert the data using Insert queries. I have created HRIS System for 250 employees in an organization from Jan 2019- till date which includes active and terminated employee’s data with the performance rating from 2019 till 2021. I created the mock data from [**https://www.mockaroo.com/**](https://www.mockaroo.com/)**,** andgenerated Insert query using [**https://tableconvert.com/excel-to-sql**](https://tableconvert.com/excel-to-sql)**.** For small tables which has less data I wrote Insert query for them.

Below are the tables for the system that were created using **Oracle Live SQL**.

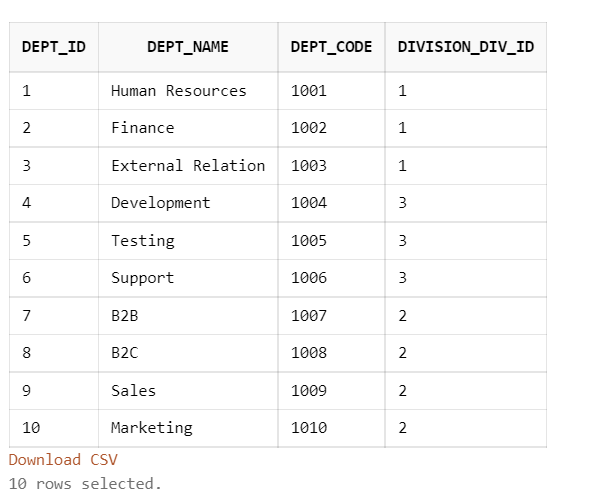
1. **Division Table**

*Select \* from Division;*



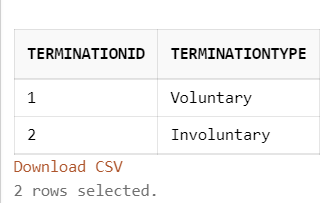
1. **Department Table**

*Select \* from Department;*



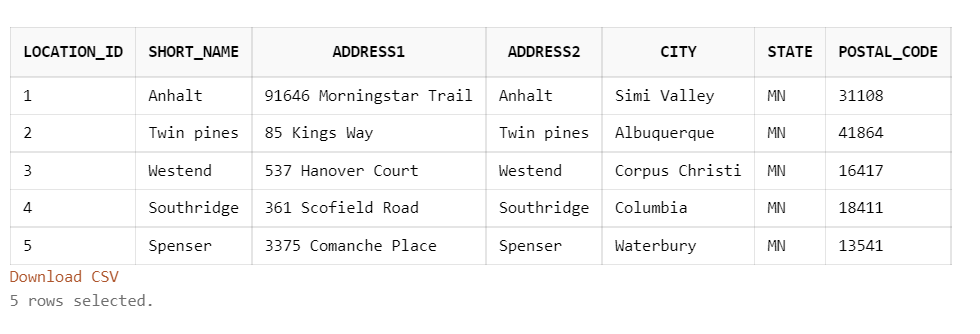
1. **Termination Table**

Select \* from Termination;



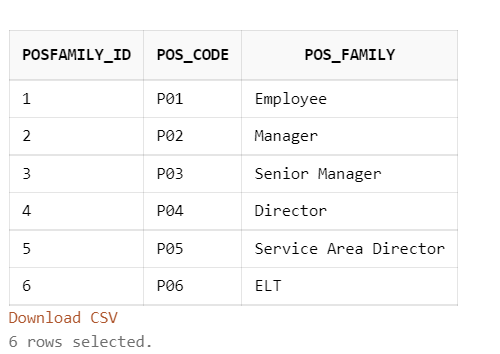
1. **WLocation**

***Select \* from WLocation;***

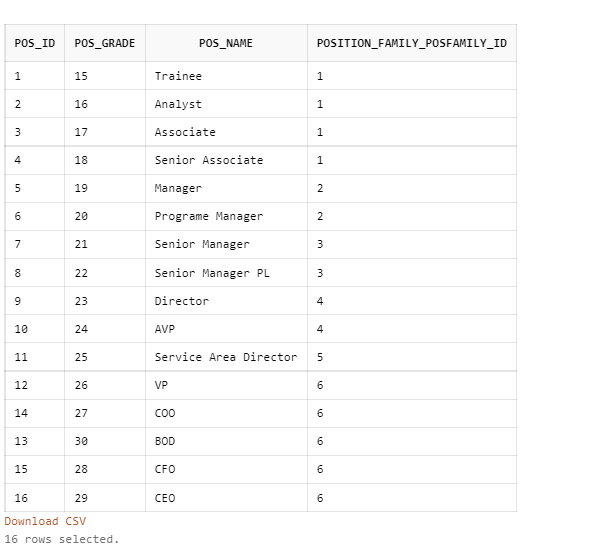


1. **Position Family Table:**

*Select \* from Position\_Family;*

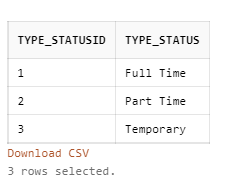


1. **Select \* from Position;**



1. **EMP\_Type Table**

*Select \* from EMP\_Type;*

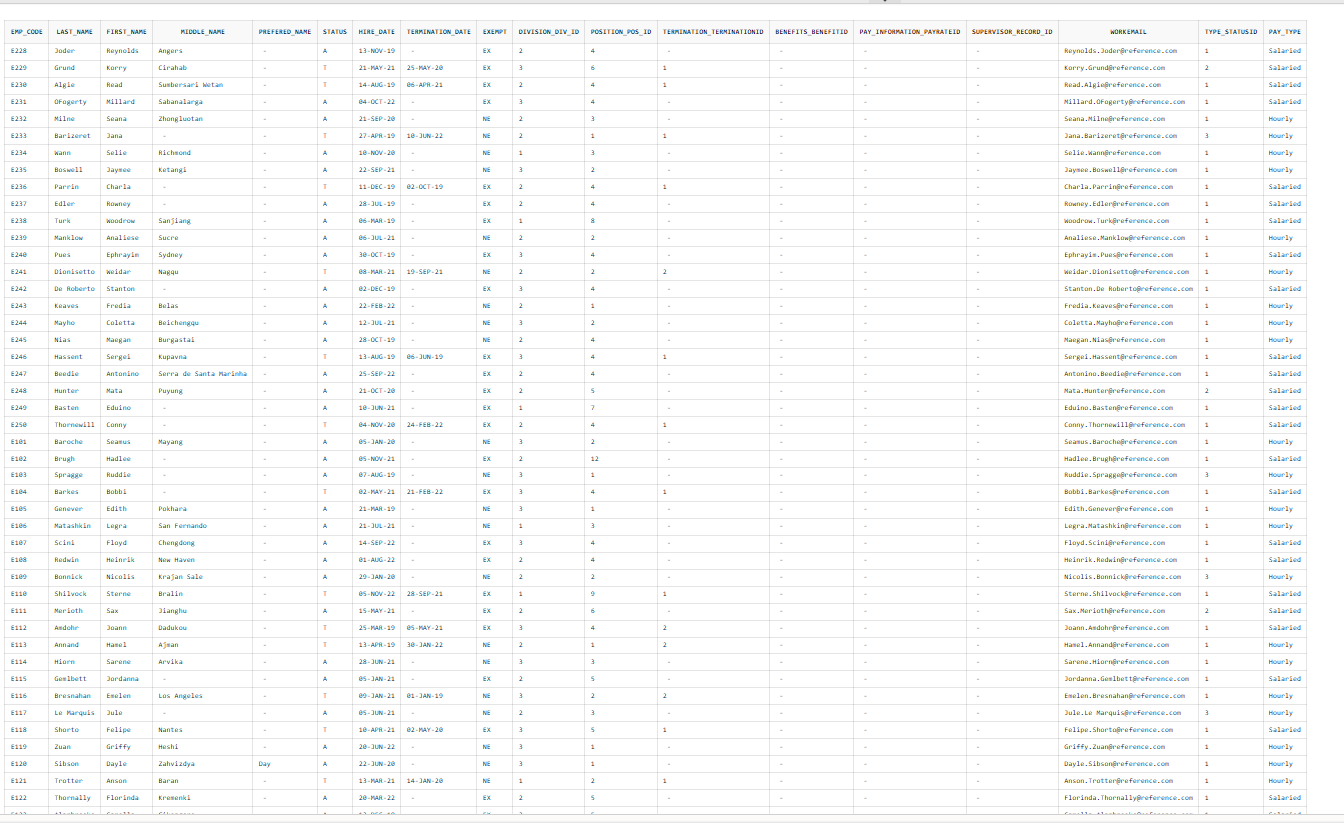


1. **Employee Table**

*Select Count (\*) From Employee;*

*Select \* from Employee*



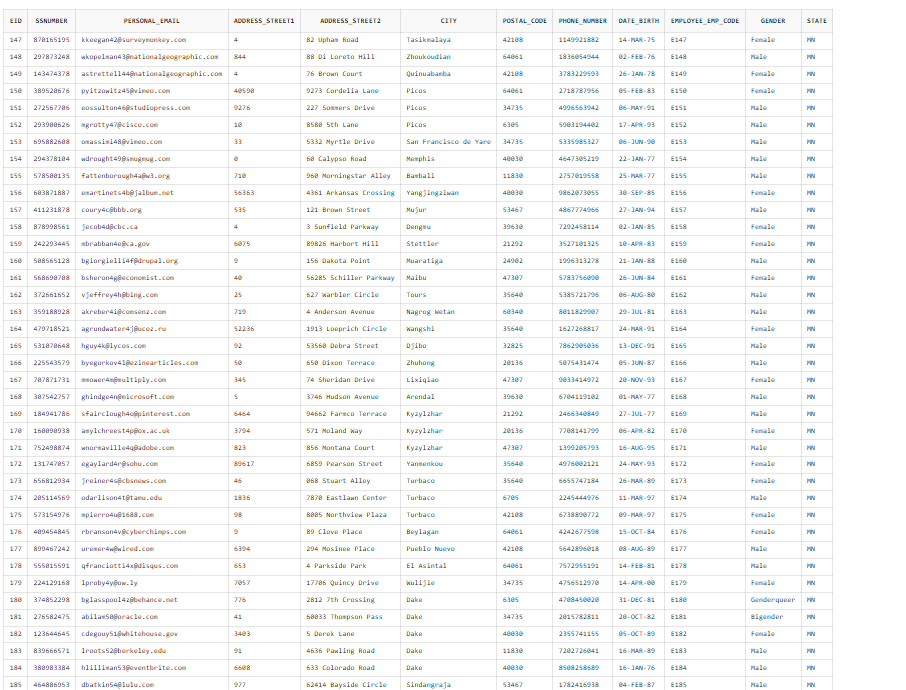


1. **Emp\_NPI Table**

*Select Count (\*) From emp\_npi;*

*Select \* From emp\_npi;*





1. ***Performance Table***

*Select Count (\*) From Performance;*

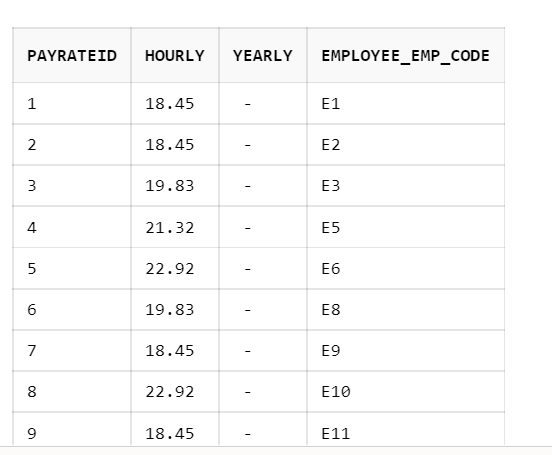
*Select \* From Performance;*



1. **Pay\_Information Table**

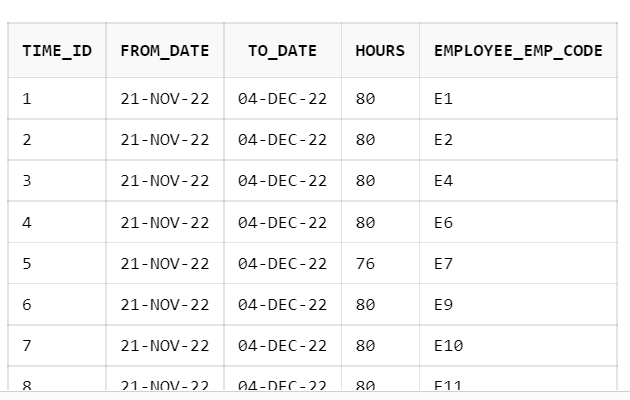
*Select Count (\*) from PAY\_INFORMATION;*

*Select Count (\*) from PAY\_INFORMATION;*



1. **Time\_Card Table**

*Select \* FROM Time\_Card;*



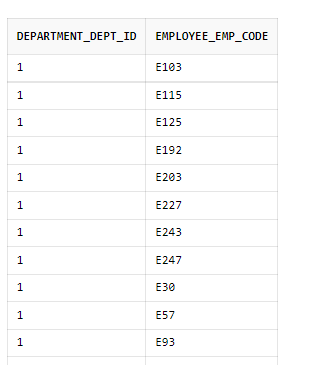
1. **Office\_Employee Table**

*select \* from office\_employee*



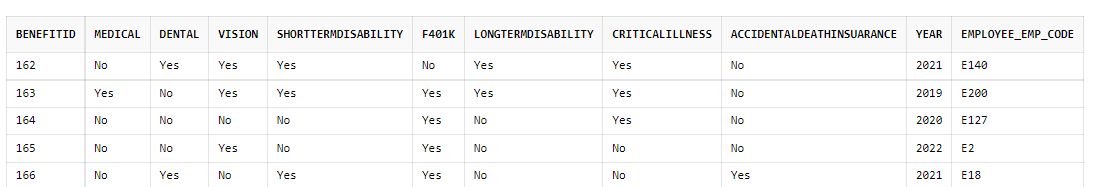
1. **Employee\_dept Table**

*select \* from employee\_dept*



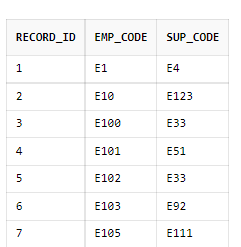
1. **Benefit Table**

*SELECT \* FROM BENEFITS*



1. **Supervisor Table**

*SELECT \* FROM Supervisor*

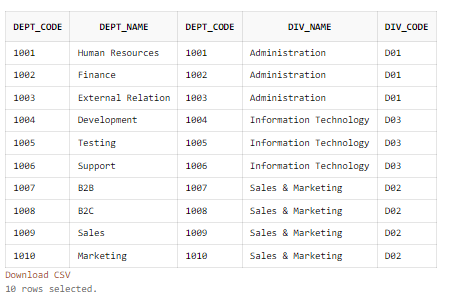


**Queries:**

To use the database for business purposes created Views as below:

1. To get all the division and departments within the organization: If you are accessing the labor allocation (i.e pay cost) for all the Division in the organization we can use this query:

*SELECT D.DEPT\_CODE, D.DEPT\_NAME, D.DEPT\_CODE, DI.DIV\_NAME,DI.DIV\_CODE FROM DEPARTMENT D JOIN DIVISION DI ON D.DIVISION\_DIV\_ID = DI.DIV\_ID;*



1. Salary of a particular Employee within a pay period: If any employee came to HR that their pay period salary was not calculated correct, we can use this query to cross check:

*SELECT E.EMP\_CODE, P.HOURLY, P.YEARLY, T.HOURS,T.FROM\_DATE, T.TO\_DATE,*

*CASE E.PAY\_TYPE*

*WHEN 'HOURLY' THEN TRUNC(P.HOURLY \* T.HOURS,2)*

*WHEN 'SALARIED' THEN TRUNC(((P.YEARLY/26)/80) \* T.HOURS,2)*

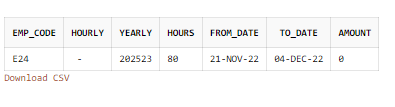
*ELSE 0*

*END AS AMOUNT*

*FROM EMPLOYEE E JOIN PAY\_INFORMATION P ON E.EMP\_CODE = P.EMPLOYEE\_EMP\_CODE*

*JOIN TIME\_CARD T ON E.EMP\_CODE=T.EMPLOYEE\_EMP\_CODE AND T.FROM\_DATE = TO\_DATE('11/21/2022','MM/DD/YYYY')*

*WHERE E.EMP\_CODE='E24';*



We can remove this statement and it will give us salary for all employees for that Pay period

1. Employees Detailed view with all the information like Employee Name, Status, Position, Termination date, Termination Type, Emp\_Type: In case of any reporting, we can use this query to get all the information of employees.

*CREATE VIEW EMPLOYEE\_DETAILS AS*

*SELECT E.EMP\_CODE,E.LAST\_NAME,E.FIRST\_NAME,E.STATUS,E.HIRE\_DATE,E.TERMINATION\_DATE,E.WORKEMAIL,E.EXEMPT,D.DIV\_NAME,D.DIV\_CODE,P.POS\_NAME,P.POS\_GRADE,PF.POS\_FAMILY,*

*ET.TYPE\_STATUS,E.PAY\_TYPE,PI.HOURLY,PI.YEARLY*

*FROM EMPLOYEE E*

*JOIN DIVISION D ON E.DIVISION\_DIV\_ID = D.DIV\_ID*

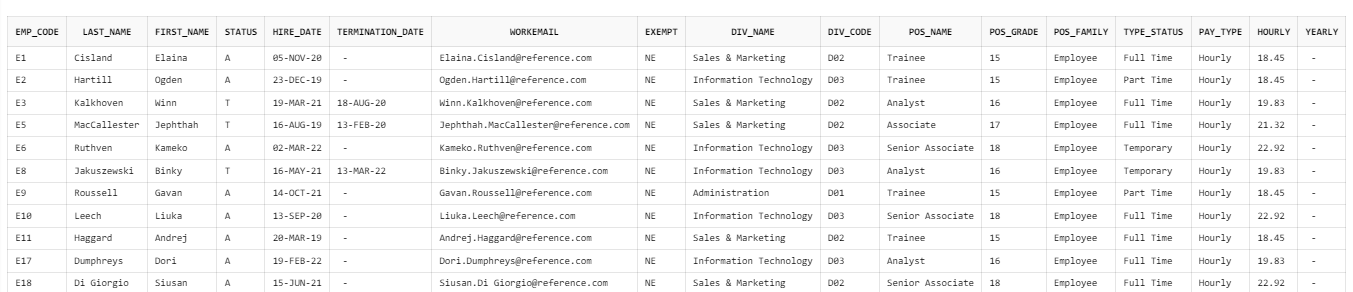
*JOIN POSITION P ON E.POSITION\_POS\_ID = P.POS\_ID*

*JOIN POSITION\_FAMILY PF ON PF.POSFAMILY\_ID=P.POSITION\_FAMILY\_POSFAMILY\_ID*

*JOIN EMP\_TYPE ET ON E.TYPE\_STATUSID = ET.TYPE\_STATUSID*

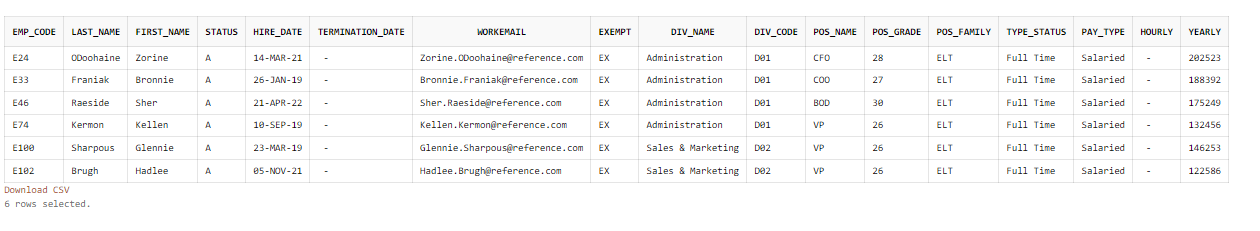
*JOIN PAY\_INFORMATION PI ON E.EMP\_CODE = PI.EMPLOYEE\_EMP\_CODE;*

*Select \* from EMPLOYEE\_DETAILS*



1. Details of all ELT (Executive Leadership Team):

*SELECT \* FROM EMPLOYEE\_DETAILS WHERE POS\_FAMILY='ELT';*



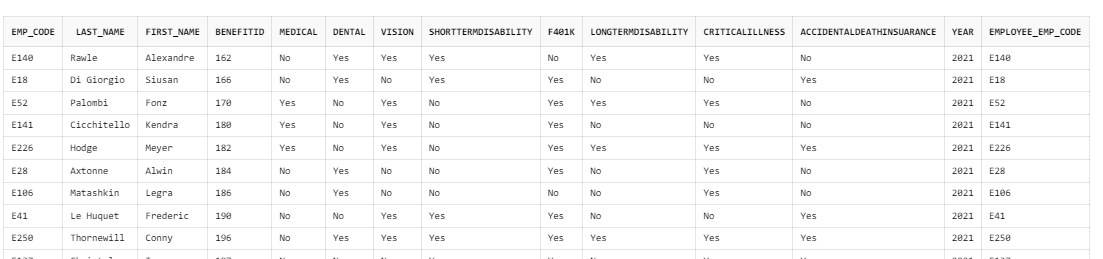
1. Benefits specific for year:

*SELECT E.EMP\_CODE,E.LAST\_NAME,E.FIRST\_NAME,B.\* FROM EMPLOYEE E JOIN BENEFITS B ON E.EMP\_CODE = B.EMPLOYEE\_EMP\_CODE AND B.YEAR=2021;*

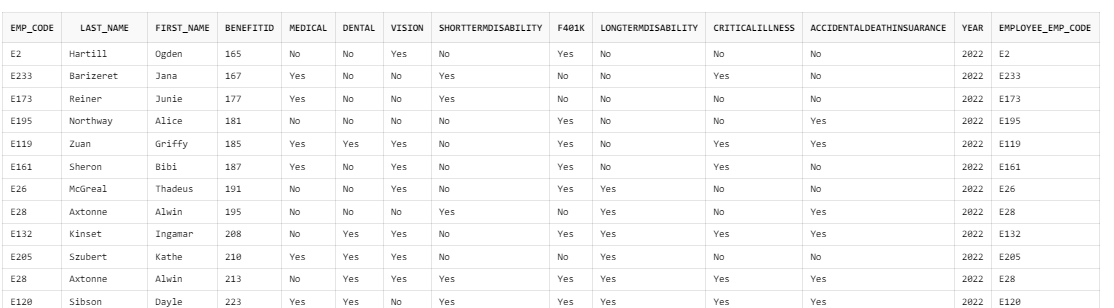
*SELECT E.EMP\_CODE,E.LAST\_NAME,E.FIRST\_NAME,B.\* FROM EMPLOYEE E JOIN BENEFITS B ON E.EMP\_CODE = B.EMPLOYEE\_EMP\_CODE AND B.YEAR=2022;*

*SELECT E.EMP\_CODE,E.LAST\_NAME,E.FIRST\_NAME,B.\* FROM EMPLOYEE E JOIN BENEFITS B ON E.EMP\_CODE = B.EMPLOYEE\_EMP\_CODE AND B.YEAR=2020;*

Year 2021



Year 2022



Year 2020



1. HR Only view Emp\_NPI Details:

This information is nonpublic and has access only to HR. This is used to complete employee verifications, sending mails regarding 401K annual mails. Etc.

*CREATE VIEW EMPLOYEE\_NPI\_DETAILS AS*

*SELECT E.EMP\_CODE,E.LAST\_NAME,E.FIRST\_NAME,E.STATUS,E.HIRE\_DATE,E.TERMINATION\_DATE,E.WORKEMAIL,E.EXEMPT,D.DIV\_NAME,D.DIV\_CODE,P.POS\_NAME,P.POS\_GRADE,PF.POS\_FAMILY,*

*ET.TYPE\_STATUS,E.PAY\_TYPE,PI.HOURLY,PI.YEARLY,ENPI.SSNUMBER,ENPI.PERSONAL\_EMAIL,*

*ENPI.ADDRESS\_STREET1,ENPI.ADDRESS\_STREET2,ENPI.CITY,ENPI.POSTAL\_CODE,ENPI.STATE,ENPI.PHONE\_NUMBER,ENPI.DATE\_BIRTH,ENPI.GENDER*

*FROM EMPLOYEE E*

*JOIN DIVISION D ON E.DIVISION\_DIV\_ID = D.DIV\_ID*

*JOIN POSITION P ON E.POSITION\_POS\_ID = P.POS\_ID*

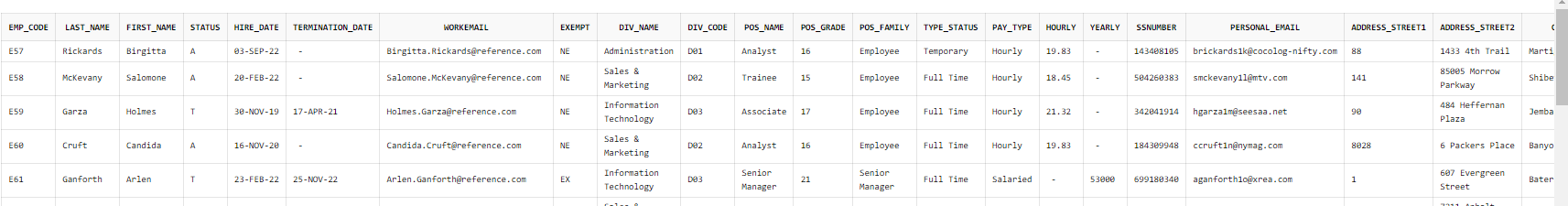
*JOIN POSITION\_FAMILY PF ON PF.POSFAMILY\_ID=P.POSITION\_FAMILY\_POSFAMILY\_ID*

*JOIN EMP\_TYPE ET ON E.TYPE\_STATUSID = ET.TYPE\_STATUSID*

*JOIN PAY\_INFORMATION PI ON E.EMP\_CODE = PI.EMPLOYEE\_EMP\_CODE*

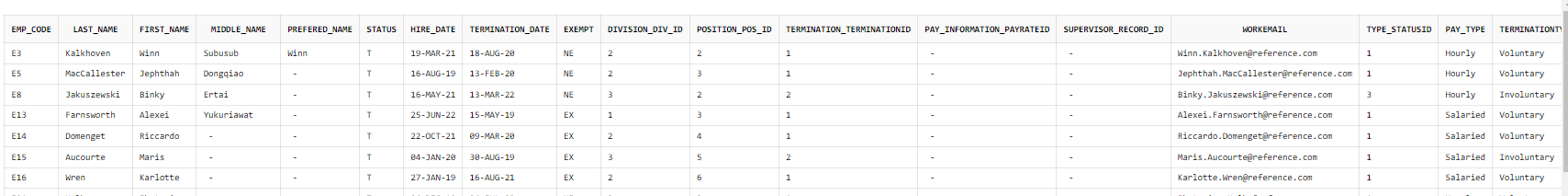
*JOIN EMP\_NPI ENPI ON E.EMP\_CODE=ENPI.EMPLOYEE\_EMP\_CODE;*

*SELECT \* FROM EMPLOYEE\_NPI\_DETAILS*



1. Termination Details from above view:

*SELECT E.\* , T.TERMINATIONTYPE FROM EMPLOYEE E JOIN TERMINATION T ON E.TERMINATION\_TERMINATIONID = T.TERMINATIONID WHERE E.STATUS='T';*



1. View for Non Performer (Rating 1) : In case of any Reduction in Force we may need to find which employees we can let go.

*SELECT E.EMP\_CODE, E.LAST\_NAME,E.FIRST\_NAME,P.RATING,P.YEAR, E.STATUS,E.HIRE\_DATE,E.WORKEMAIL,E.EXEMPT,D.DIV\_NAME,D.DIV\_CODE,P.POS\_NAME,P.POS\_GRADE,PF.POS\_FAMILY,*

*ET.TYPE\_STATUS,E.PAY\_TYPE,PI.HOURLY,PI.YEARLY*

*FROM EMPLOYEE E*

*JOIN DIVISION D ON E.DIVISION\_DIV\_ID = D.DIV\_ID*

*JOIN POSITION P ON E.POSITION\_POS\_ID = P.POS\_ID*

*JOIN POSITION\_FAMILY PF ON PF.POSFAMILY\_ID=P.POSITION\_FAMILY\_POSFAMILY\_ID*

*JOIN EMP\_TYPE ET ON E.TYPE\_STATUSID = ET.TYPE\_STATUSID*

*JOIN PAY\_INFORMATION PI ON E.EMP\_CODE = PI.EMPLOYEE\_EMP\_CODE*

*JOIN PERFORMANCE P ON E.EMP\_CODE=P.EMPLOYEE\_EMP\_CODE AND P.RATING = 1*

*WHERE E.STATUS='A';*

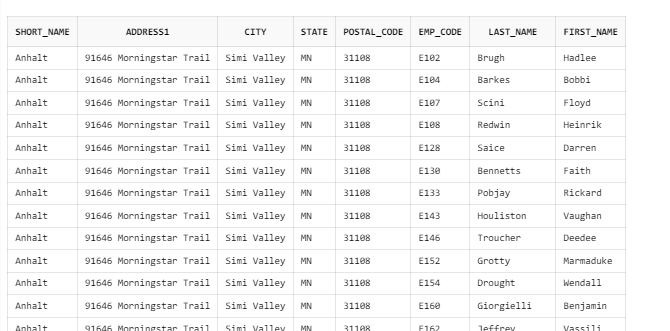
\*\*We can also run this report for getting the ratings for high performers so we can in batch edit the appraisals.



1. Employee office location Information:

*SELECT w.SHORT\_NAME,w.ADDRESS1, w.CITY, w.STATE, w.POSTAL\_CODE, E.emp\_code, E.Last\_Name, E.First\_Name from wlocation w join office\_employee oe on w.LOCATION\_ID = oe.WLOCATION\_LOCATION\_ID*

*join Employee e on oe.EMPLOYEE\_EMP\_CODE = e.Emp\_Code order by w.LOCATION\_ID;*



**Topics From Class:**

**Normalization:** I have used normalization principles to create the entities for data integrity and prevent anomalies to reduce redundancies.

**Logical and relational Diagram:** I have created logical diagram and relational diagram using data modeler and then used forward engineering to create DDL statements.

**SQL DML and DDL:** DDL statements were created for managing table structures, including CREATE TABLE, ALTER TABLE, DROP TABLE, and TRUNCATE TABLE. Using these statements, I have built the table structure for the HRIS. Then completed DML statements for managing data INSERT, UPDATE, DELETE.

**Views:** I have created different views to promote data encapsulation as well as easy access based on role for example, I have created employee\_details view that can be viewed by all managers, leadership team. And created one more view employee\_NPI details that contains non public information of an employee. Only HR personnel and audit team should have access to this data.

**Sequnce / Triggers:** I have created sequence to generate incremental number for surrogate key, also have created before insert trigger. This trigger takes next value from sequence and assigned to surrogate key value for record being inserted.Example: Division table

**Observation:** After creating DDL statements from relational diagram we added one attribute, and updated name of 401K attribute to F401K in benefits table. Also, I observed that it is not a great idea to make start a attribute name with number (401K) it was giving error, so we updated the attribute name. While inserting bulk data in the tables Employee and Employee\_NPI that has before insert trigger were high amount of time and eventually failing so, I removed trigger on Employee and Employee\_NPI tables.

**Conclusion:** HRIS helps to create modules and add more information where we can have whole lifecycle of an employee with regards to HR. Like we can have modules like ATS – where an applicant can apply for the position, once selected convert into an employee and stay with the organization till end of his journey in organization. We can add Leaves, FMLA, Employee Changes etc. The system has all the information regarding an employee which can be used for multiple reasons like – Reporting, Planning, Budgeting, maintains, manages, and processes detailed employee information and human resources-related policies and procedures, Performance Evaluation, Employee Verification Payroll etc.

During our class I learned regarding DBMS, logical models, relational models, SQL, DDL/DML statement and PL/SQL construct. This learning helped to ideate, design and build working RDBMS model of HRIS system. I also learned database normalization, integrity constrains and cardinalities during class. This has enabled me to apply all sort of optimization to build robust backend database for an HRIS system that can scale to serve HR applications. Project to build HRIS backend database helps to brainstorm and apply all individual concepts I learned during class. It gave hands-on opportunity to learn the tools. I learnt to create the update of bulk data insert upload, that can be used to data migration. Also learnt how database system can help to solve real life problems by better organization and centralized control.

If time permitted, I would have loved to write the store procedures and functions to create utilities for CRUD operations. And developed the payroll part of the system where we can know more about paystubs, taxes, leaves etc.