Final Project Report
Payroll Management System

PAYDAY

December 6, 2020

Redghy Jean
Dani Major
Juana Tavera

Table of Contents

Introduction

This database will serve as a payroll system for a company and its employees. The database that we are creating oversees all that has to do with the way a business is paying workers and documenting business charges. This payroll system generally caters towards the employers of general organizations that look for a productive method to pay and monitor worker money related records. The customer base may comprise businesses of all sizes that need an advantageous payroll system of their own. The design of this framework allows it to be seamlessly carried out whichever sort of business our customers may require.

Our motivation for creating this product is that our team is composed of young adults eager to begin our careers in the professional workforce. Whether we are working for a business or beginning our own, we wanted to create a tool that could be valuable to all persons involved.

This database is intended for businesses that might want to monitor their money related records without the dissatisfaction of having to interminably look for one specific snippet of data. It will keep their information composed and can be utilized for discovering records on an employee easily and expeditiously. One of the significant parts of our product incorporates the portion of hours for explicit projects, teams, and departments. This element permits businesses to effortlessly track and determine which department and project budgets to pay employees out of, allowing for proper budgeting and allocation of funds. This element will likewise be favorable for employees of all levels, by monitoring their hours for each pay period, and their hourly compensations.

System Features

Employee Pay Tracking
Tax Deductions
Employee Base Salary
Pay & Benefits
Employee Personal Information
Performance Reviews

Requirement Analysis

User Requirements (Features our users want in this payroll management system - PAYDAY)

1. The system will allow the user to find the company and department an employee is under.
2. The system will allow the user to view the performance of an employee.
3. The system will allow the user to view an employee's pay stub.
4. The system will allow the user to view the total payments that an employee received.
5. The system will allow the user to view the total deductions that an employee received.
6. The system will allow the user to view an employee's state and federal tax.
7. The system will allow the user to view if an employee chooses to receive benefits from the company.
8. The system will allow the user to view an employee's base salary and bonuses.
9. The system will allow the user to view if an employee has ever taken a leave or if they will take a leave.
10. The system will allow the user to view how many absences and tardies an employee has had.

Constraints

1. A Company can have many departments and many employees.
2. A Department can have many employees but can only be a part of one department.
3. An employee has to authenticate themselves with their Employee ID to view their performance.
4. An employee will only have one benefit package.
5. An employee will only receive one bonus every year.
6. An employee will only receive at most one payout.
7. An employee can have up to 20 absences and 30 tardies in one year.

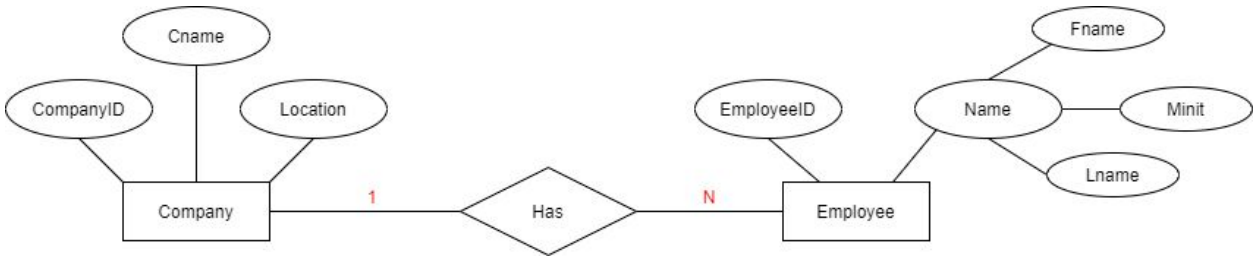
Data Requirements

1. Company data - CompanyID, Cname, Location; Company has a relationship with Department and Employee.
2. Department data - DepartmentID, Dname; Department has a relationship with Company and Employee.
3. Employee data - EmployeeID, Fname, Minit, Lname; Employee has a relationship with Company, Department, Performance, and Payout.
4. Performance data - PerformanceID, SpeedTests; Performance has a relationship with Employee.

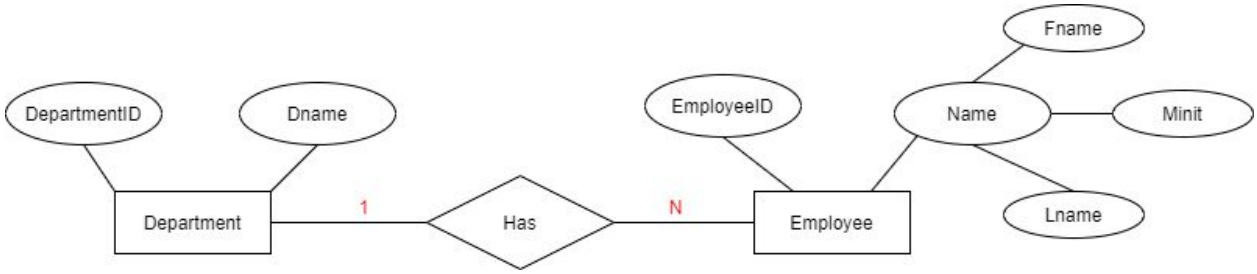
5. Payout data - PaystubID; Payout has a relationship with Employee, Payments, and Deductions.
6. Payments data - Ptotal; Payments has a relationship with Payout, Salary, and Leave.
7. Deductions data - Dtotal; Deductions has a relationship with Payout, Tax, Attendance, and Benefits.
8. Salary data - SalaryID, baseSalary, Bonus; Salary has a relationship with Payments.
9. Leave data - leaveID, leavePay, fromDate, toDate; Leave has a relationship with Payments.
10. Tax data - TaxID, StateTax, FederalTax; Tax has a relationship with Deductions.
11. Attendance data - AttendanceID, Absences, Tardiness; Attendance has a relationship with Deductions.
12. Benefits - BenefitsID, Retirement, Insurance; Benefits has a relationship with Deductions.

Functional Requirements

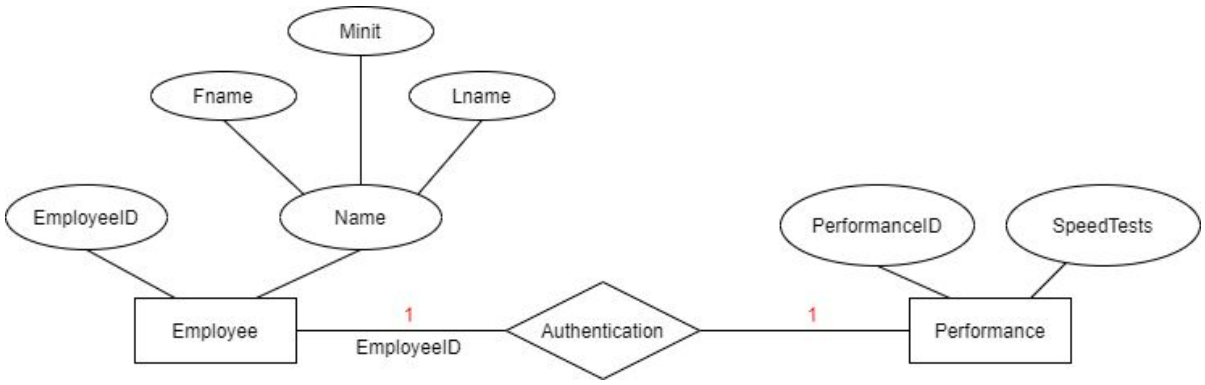
1. Find the name of the Company an Employee works for with Employee Company_ID and Company Cname.



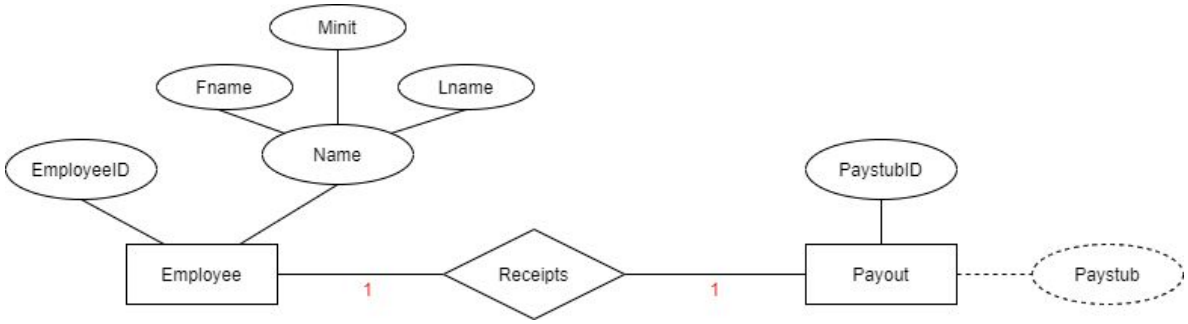
2. Find the name of the Department an Employee is a part of with Employee EmployeeID and Department Dname. Display the name of the Company they work for as well.



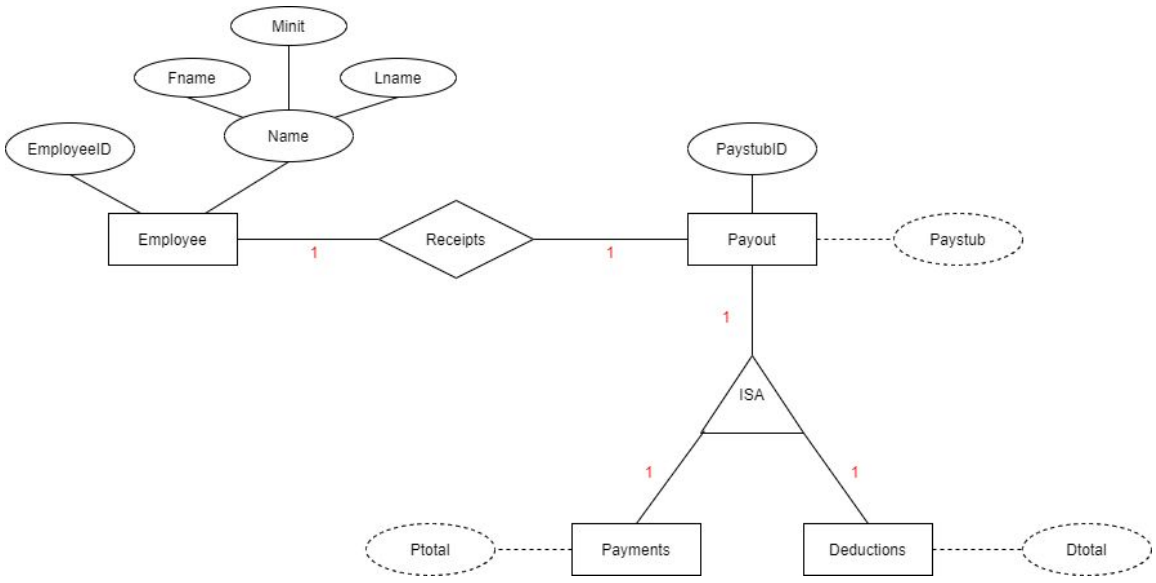
3. Find the speed test of an employee with an Employee EmployeeID and Performance SpeedTests. Order by the fastest time and companyID. Display the name of the company as well.



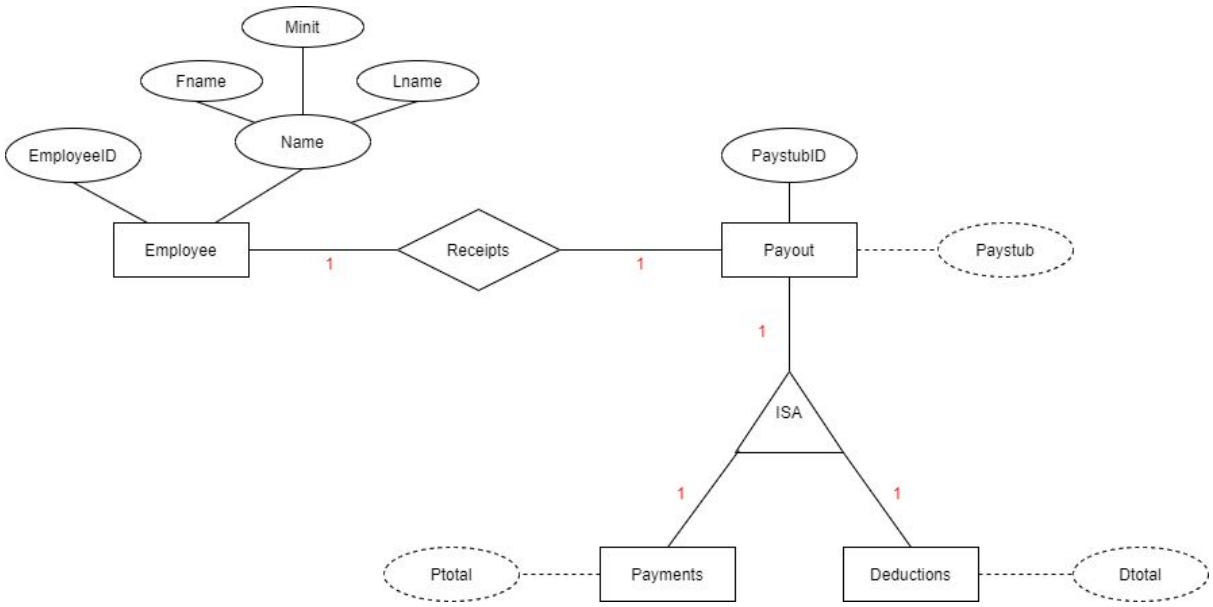
4. Find an employee's payout with Employee EmployeeID, Payout PaystubID, and Payout Paystub. Display the first name and last name of the employee and the Company's name.



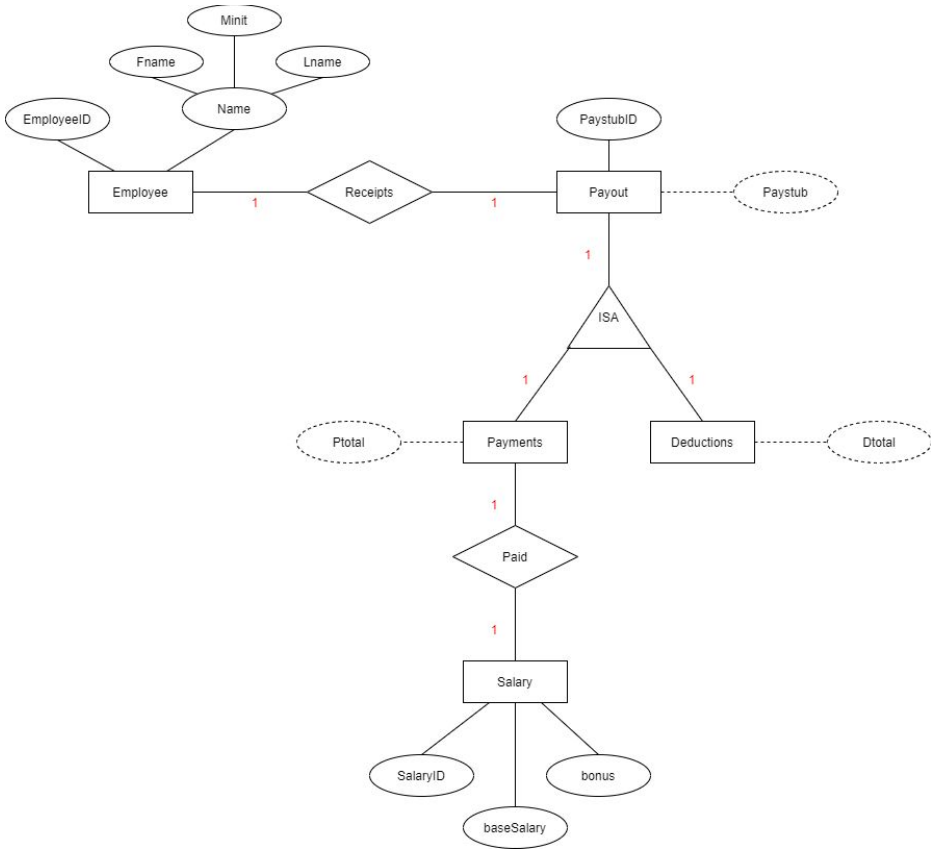
5. Find an employee's payments received with Payments Ptotal and Employee EmployeeID. Display the BaseSalary, Bonus, and LeavePay. Order by BaseSalary.



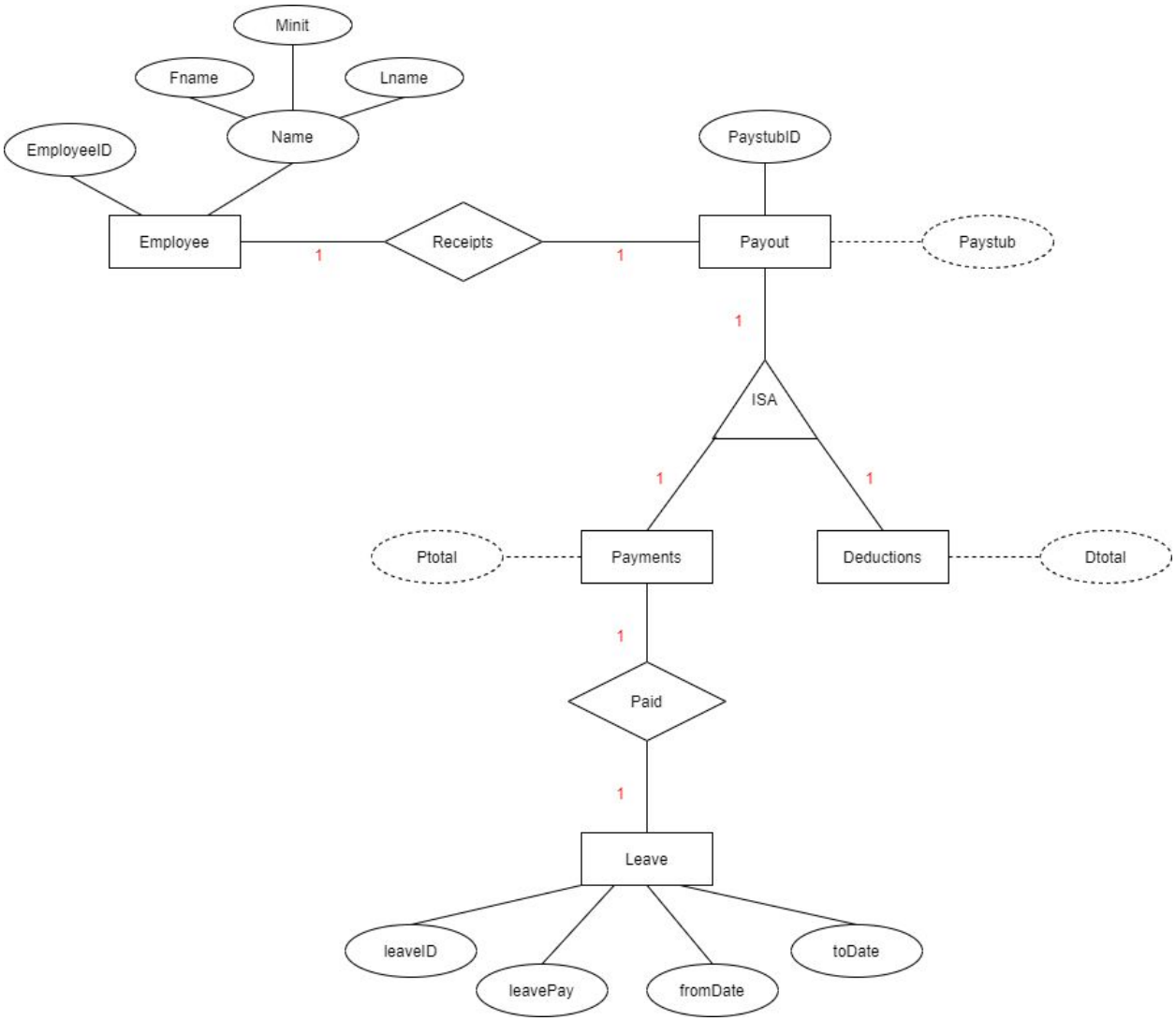
6. Find an employee's deductions received with Deductions Dtotal and Employee EmployeeID. Display Tax deductions and Benefits deductions and Absences.



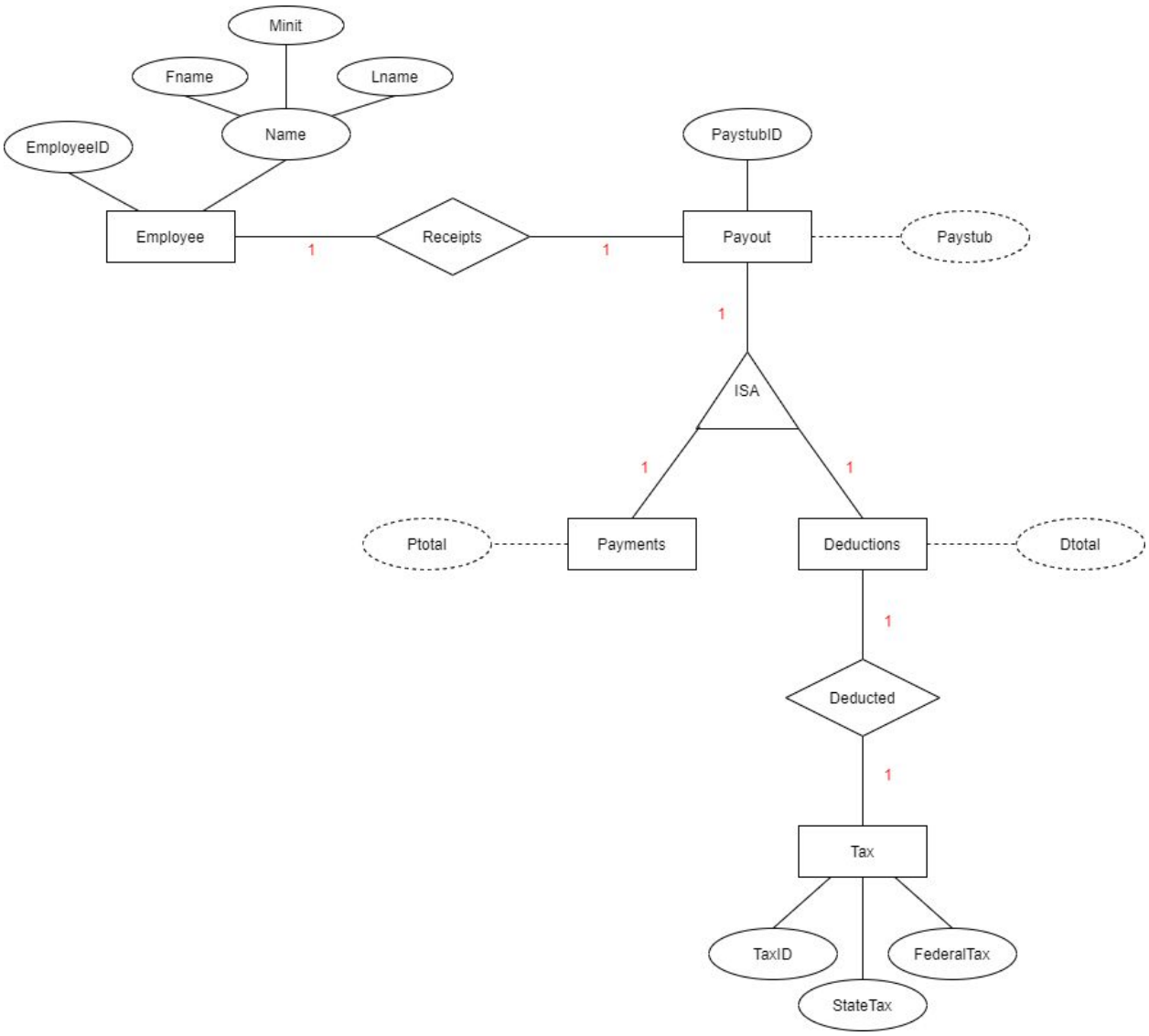
7. Find all salaries that are greater than 50000. Order by company name and department name.



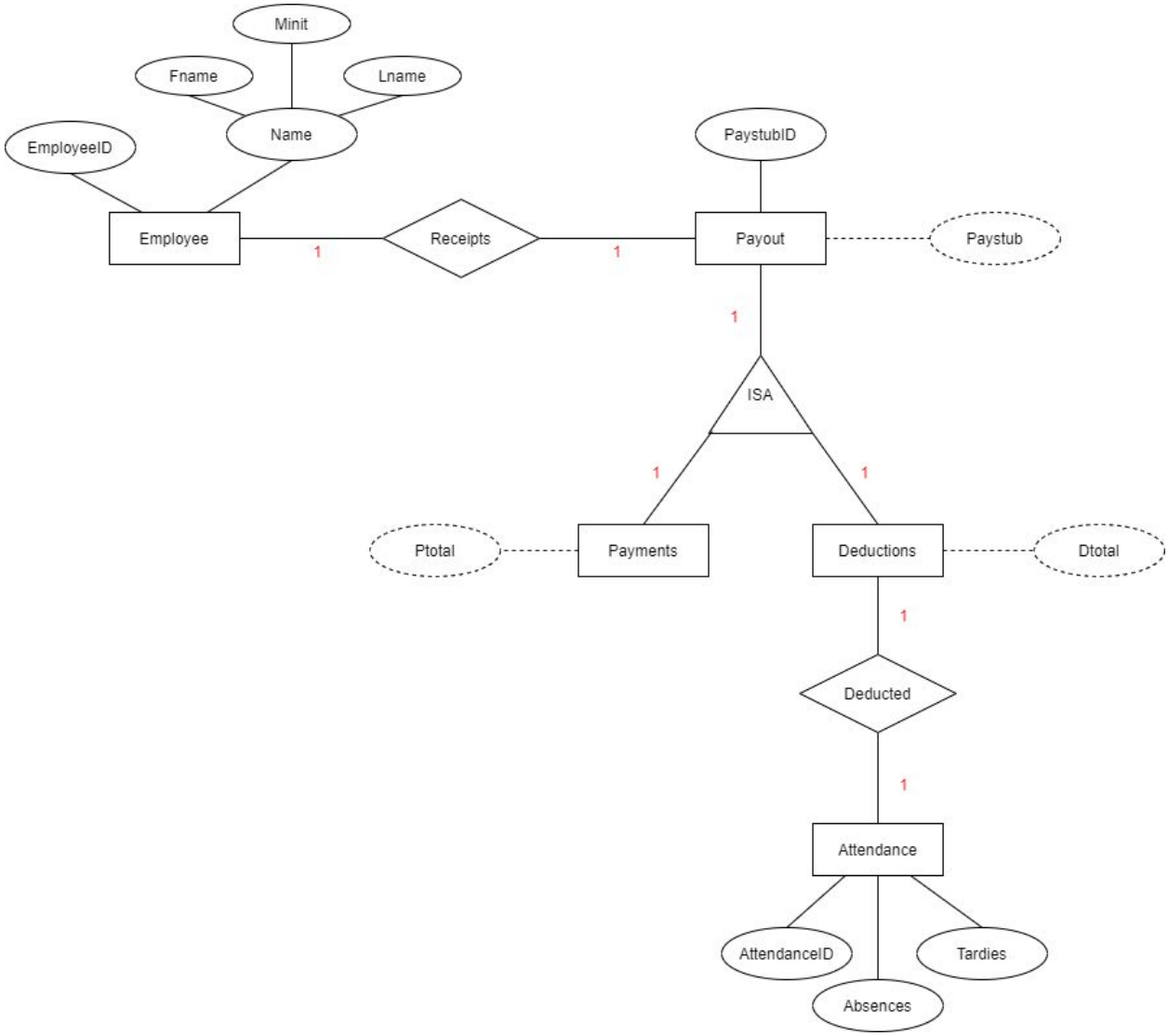
8. Find all the employees that had a leave fromDate and toDate in 2020. Order by Company name.



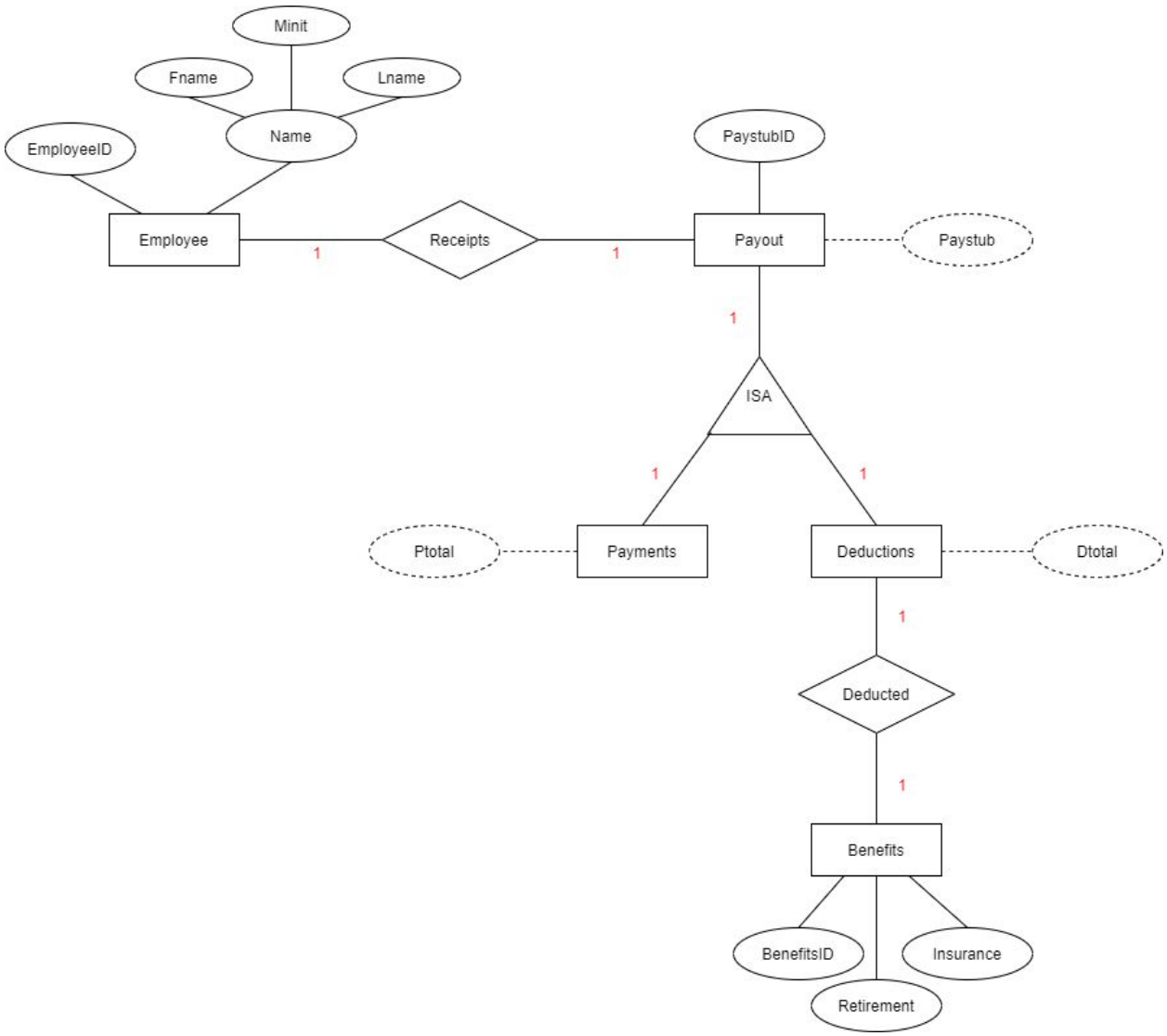
9. Find how much an employee's payout goes to taxes with Tax StateTax, FederalTax and Employee EmployeeID. Order by Company name.



10. Find which employees have greater than 3 absences and tardies. Order by company name and department name.



11. Find out how much an employee's payout goes to benefits with BenefitsID, Benefits Retirement, Benefits Insurance and EmployeeID. Order by Company name.



Conceptual Design

Entities and Attributes

Entity 1: Company

Entity

- The entity named Company houses all the departments and employees who are enrolled in using the payroll system. The Company entity has two attributes, all of which are mandatory for each company to provide in order to use this service

Attributes

- CompanyID: This attribute will provide a unique way for companies to identify themselves within the system.
- Cname: This attribute refers to the specific name of the company using the payroll system.
- Location: This attribute refers to the location of the companies who are using the payroll system.

Relationships

1. The entity Company has a relationship with the entity Department. This relationship represents that the company has many departments.
2. The entity Company has a relationship with the entity Employee. This relationship represents that the company has many employees.

Primary Key

- It is identified by the CompanyID attribute because it will have a unique value unlike Cname and Location.

Entity 2: Department

Entity

- The entity named Department houses all the employees who are a part of the company and their respective department. The Department entity has two attributes, all of which are mandatory for each company to provide in order to use this service.

Attributes

- DepartmentID: This attribute will provide a unique way for companies to identify their department within the system.

- Dname: This attribute refers to the specific name of the department within a company.

Relationships

1. The entity Department has a relationship with the entity Company. The relationship represents that a department can be a part of only one company.
2. The entity Department has a relationship with the entity Employee. The relationship represents that a department can have many, at least one, employee(s) who are a part of it.

Primary Key

- It is identified by the DepartmentID attribute because it will have unique attributes unlike Dname.

Entity 3: Employee

Entity

- The entity named Employee takes on the role of the most intractable entity. This entity is determined by Company and Employee and stores the personal information of each company's specific employees.

Attributes

- EmployeeID: This attribute refers to the identification number that each company uses to identify their employees.
- Fname: This attribute refers to an employee's first name.
- Minit: This attribute refers to the middle initial of an employee's name if applicable.
- Lname: This attribute refers to an employee's last name.

Relationship

1. The entity Employee has a relationship with the entity Company. An employee can only be a part of one company while a company may have many employees.
2. The entity Employee has a relationship with the entity Department. An employee can only be a part of one department within the specific company while a department may have many employees.
3. The entity Employee has a relationship with the entity Performance. An employee can view their performance via the authentication of their assigned EmployeeID in order to upkeep system maintenance. The employee must be currently employed and verified in the system to be accounted for in the system maintenance. This is purely for system performance in order to keep outputting the most current and accurate results possible.

4. The entity Employee has a relationship with the entity Payout via the receipts relationship. This relationship exists in order to pay the employees who work for their respective company.

Primary Key

- It is identified by the EmployeeID attribute as it is the most unique way to identify the employee's data.

Entity 4: Performance

Entity

- The entity named Performance exists to exhibit system performance and upkeep maintenance via speed tests and cleaning. This entity will hold an updated payroll roster in order to keep up to date with who is and who isn't currently active within the payroll system.

Attributes

- PerformanceID: This attribute provides a unique way to identify an employee's performance.
- SpeedTests: This attribute takes care of upkeep and maintenance to keep the system working as new.

Relationship

1. The entity Performance has a relationship with the entity Employee. This relationship exists via the authentication relationship. It is in there to verify if an employee is still hired at the company. Also they will be able to see how they are performing

Primary Key

- It is identified by the PerformanceID attribute because it is unique.

Entity 5: Payout

Entity

- The entity named Payout takes on the role of paying the employees based on the data managed in entities Payments and Deductions. This entity houses the paystub and roster for the employee and the companies who use the payroll system.

Attributes

- PaystubID: This attribute provides a unique way to identify an employee's pay stub.
- Paystub: This attribute houses the employee's weekly pay stub and salary.

Relationship

1. Payout entity has a relationship with the entity Employee via the receipts relationship. This relationship exists in order to connect employees with information towards their pay in respect to the company's payroll.
2. The Payout entity also has a relationship with the entities Payments and Deductions via the ISA relationship. The ISA relationship classifies the entities in an entity set into subclasses; therefore, the Payments and Deductions entities determine the Payout of an employee.

Primary Key

- It is identified by PaystubID because it is unique.

Entity 6: Payments

Entity

- The entity named Payments is part of the entity hierarchy with Payout and Deductions. This entity houses the gross pay of an employee. It contains only one attribute Ptotal.

Attributes

- Ptotal: This attribute is a derived attribute. It is calculated through the baseSalary, Bonus, and leavePay.

Relationship

1. The entity Payments has a relationship with the entity Payout. This relationship is an ISA relationship because they are in an entity hierarchy. It exists to view how the gross pay of an employee before the deductions are made.
2. The entity Payments also has a relationship with entities Salary and Leave via the paid relationship. This relationship exists between them because the two entities, Salary and Leave, contribute the derived attribute Ptotal.

Primary Key

- It is identified by the attributes Ptotal and PaystubID because Ptotal on its own is not unique enough.

Entity 7: Deductions

Entity

- The entity named Deductions is part of the entity hierarchy with Payout and Payments. This entity houses how much pay is deducted from the employee. It only has one attribute Dtotal.

Attributes

- Dtotal: This attribute is a derived attribute which is calculated through deductions such as StateTax, FederalTax, Absences, Retirement and Insurance.

Relationship

1. The entity Deductions has a relationship with the entity Payout. This relationship is an ISA relationship because they are in an entity hierarchy. It exists to view how much pay is deducted from an employee's pay stub.
2. The entity Deductions has a relationship with entities Tax, Attendance, and Benefits via deducted relationship. This relationship exists between them because the three entities, Tax, Attendance, and Benefits, contribute to the derived attribute Dtotal.

Primary Key

- It is identified by the attributes Dtotal and PayoutID because Dtotal will not have unique enough values.

Entity 8: Salary

Entity

- The entity named Salary houses the base salary for all employees within a company. It is an important entity that is mandatory for each company to complete when hiring new employees.

Attributes

- SalaryID: This attribute provides a unique way for a company to identify their employee's salary and bonus.
- baseSalary: This attribute refers to the standard amount of pay determined by the company that the employee works for.
- Bonus: This attribute can be left null and is up to the discretion of the department in which the employee is under to fill out.

Relationship

1. The entity Salary has a relationship with entity Payments via the paid relationship. This relationship exists because the entity Salary and its attributes contribute to the derived attribute Ptotal.

Primary Key

- It is identified by the SalaryID because it is unique.

Entity 9: Leave

Entity

- The entity named Leave contains all the employees who have had a leave within the company. It is not mandatory for all employees within a company to have this entity filled out.

Attributes

- leaveID: This attribute is a unique way to identify the occurrence of a leave an employee within the company.
- leavePay: This attribute contains the amount an employee will get paid when they are on leave from the company.
- fromDate: This attribute contains the start date of the leave of an employee.
- toDate: This attribute contains the end data of the leave of an employee.

Relationship

1. The entity Leave has a relationship with entity Payments via the paid relationship. This relationship exists because the entity Leave and its attribute, leavePay, contribute to the derived attribute Ptotal.

Primary Key

- It is identified by the leaveID because it is unique.

Entity 10: Tax

Entity

- The entity named tax serves to regulate taxes invoked by the state and federal rates. This entity will have a relation with Deductions which has a relationship with Payout. It collects tax money and stores the amount that should be refunded given that the employee files their taxes.

Attributes

- TaxID: This attribute is a unique way for a company to identify the employee's taxes.
- StateTax: This attribute attests to the amount of taxes invoked by state governments to be paid.
- FederalTax: This attribute attests to the amount of taxes invoked by the federal government to be paid.

Relationship

1. The entity Tax has a relationship with Deductions via the deducted relationship. This relationship exists because the entity Tax contributes to the derived attribute Dtotal.

Primary Key

- It is identified by the TaxID because the StateTax attribute on its own is not unique enough. StateTax is however, more unique than the federal tax because it fluctuates from location to location unlike federal tax which is more a national standard.

Entity 11: Attendance

Entity

- The entity named Attendance houses the attendance of the employees. It contains only the attributes when the employee misses work or is late.

Attributes

- AttendanceID: This attribute provides a unique way for a company to identify an employee's absences and tardies.
- Absences: This attribute contains the amount of times a specific employee has not been to work.
- Tardies: This attribute contains the amount of times a specific employee has been late to work.

Relationship

1. The entity Attendance has a relationship with Deductions via the deducted relationship. This relationship exists because the entity Attendance and its attribute, Absences, contributes to the derived attribute Dtotal.

Primary Key

- It is identified by the AttendanceID because it is unique.

Entity 12: Benefits

Entity

- The entity named Benefits houses the benefits the employee receives from the company they work such as insurance and retirement benefits.

Attributes

- BenefitsID: This attribute provide a unique way for company to identify an employee's benefits.

- Retirement: This attribute contains how much of an employee's pay stub goes to their retirement plan i.e. 401k.
- Insurance: This attribute contains how much of an employee's pay stub goes to their insurance i.e. healthcare, vision, etc.

Relationship

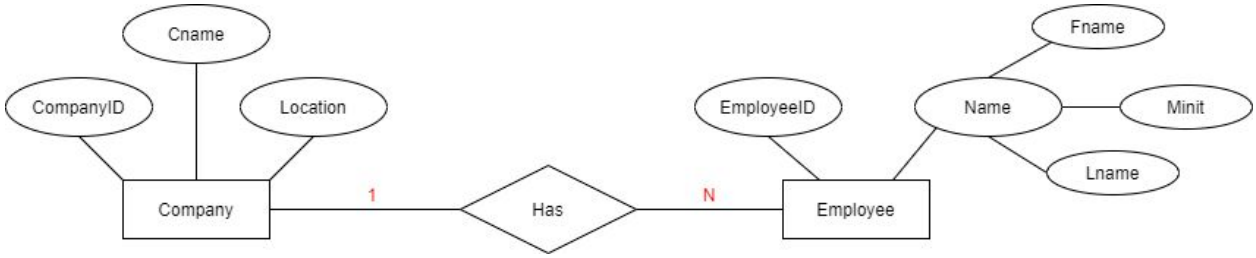
1. The entity Benefits has a relationship with Deductions via the deducted relationship. This relationship exists because the entity Benefits contributes to the derived attribute Dtotal.

Primary Key

- It is identified by the attribute BenefitsID because it is unique.

Relationships

Relationship 1: Has



Relation

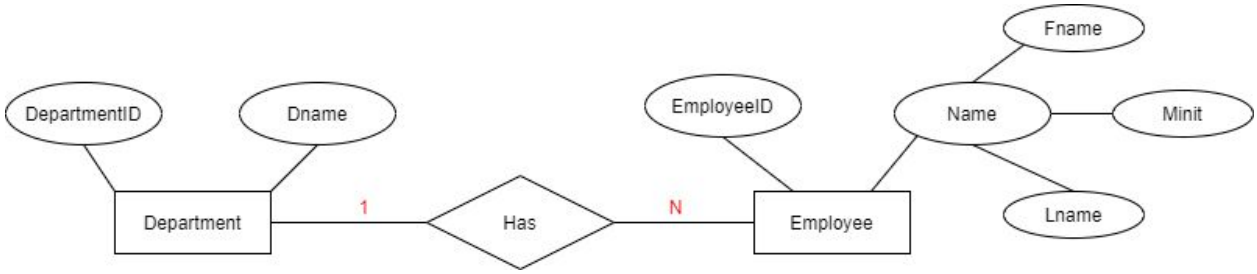
- Shows the relation between entities Company and Employee.

Attributes

- CompanyID (Foreign Key in Employee from Company): This key directly pushes specific information from a specific company onto the user.

Cardinality

- The cardinality of Has relationship between Company and Employee is (1: N).



Relation

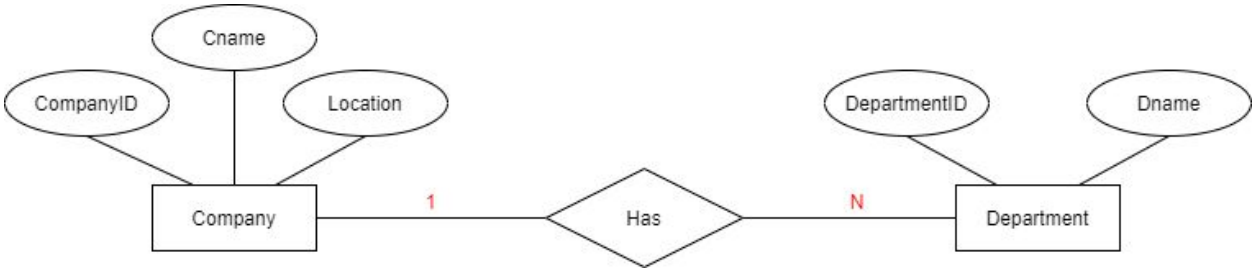
- Shows the relations between entities Department and Employee.

Attributes

- DepartmentID (Foreign Key in Employee from Department): This key directly pushes specific information from a specific department onto the user.

Cardinality

- The cardinality of Has relationship between Department and Employee is (1: N).



Relation

- Shows the relation between entities Company and Department.

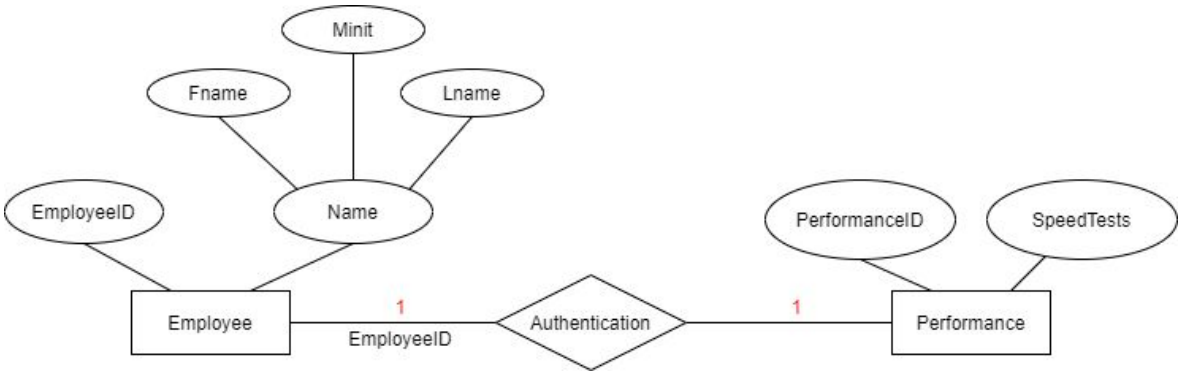
Attributes

- CompanyID (Foreign Key in Department from Company): This key directly pushes specific information from a specific company onto the user.

Cardinality

- The cardinality of Has relationship between Company and Employee is (1: N).

Relationship 2: Authentication



Relation

- Shows the relations between entities Employee and Performance.

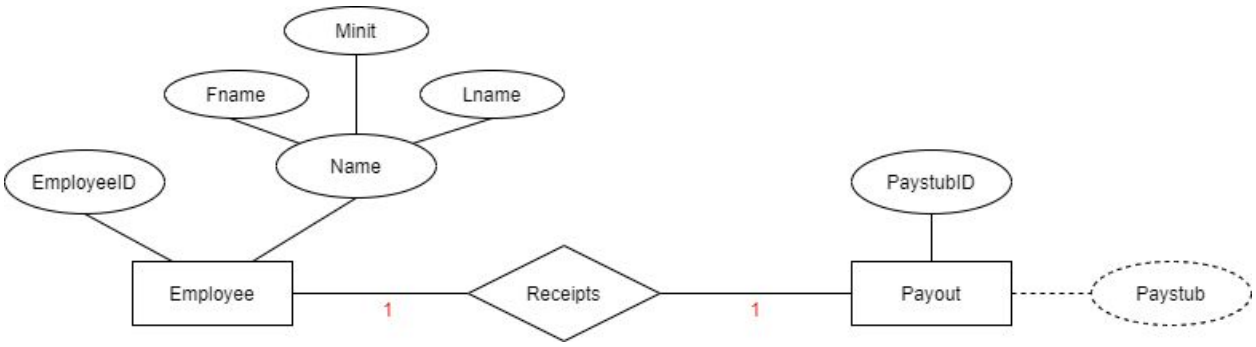
Attributes

- EID (Foreign Key in Performance from Employee): This key directly pushes the Employee information to entity Performance, with this foreign key users will be able to see if a specific employee is currently working there.

Cardinality

- The cardinality of Authentication relationship between Employee and Performance is (1: 1).

Relationship 3: Receipts



Relation

- Shows the relationship between entities Employee and Payout

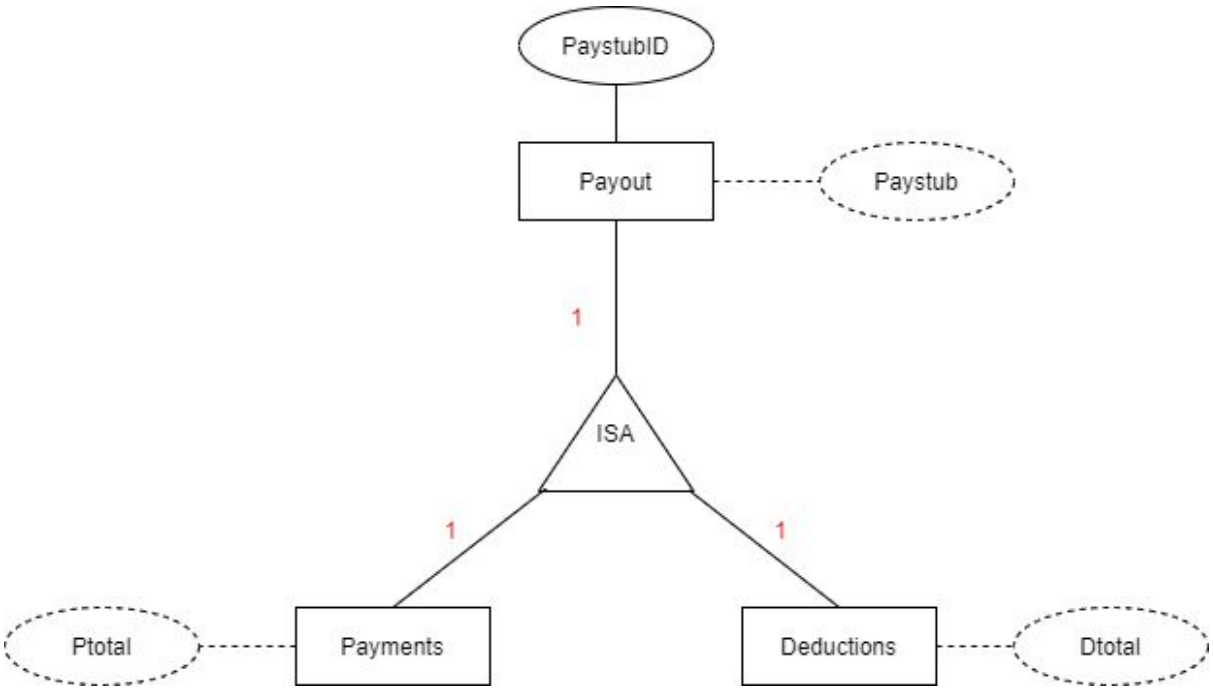
Attributes

- EID (Foreign Key in Payout from Employee): This key directly pushes the Employee information to entity Payout, with this foreign key users will be able to see how much a specific employee gets on their pay stub.

Cardinality

- The cardinality of Receipts relationship between Employee and Payout is (1: 1).

Relationship 4: ISA



Relation

- Shows the relation between entities Payout, Payments, and Deductions.

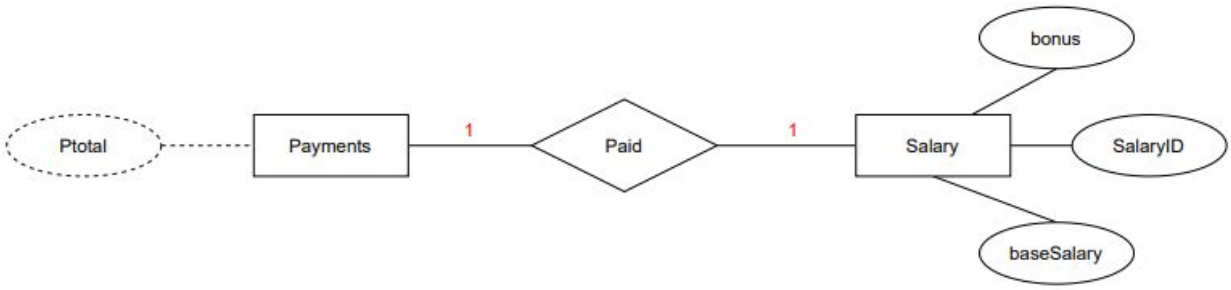
Attributes

- EID (Foreign Key in Payout from Employee): This key directly pushes the Employee information to entity Payout, with this foreign key users will be able to see how much a specific employee gets on their pay stub.
- Derived attributes: Paystub, Ptotal, and Dtotal.
- Paystub derived from Ptotal - Dtotal.
- Ptotal derived from baseSalary + Bonus (if applicable) + leavePay.
- Dtotal derived from StateTax + FederalTax + Retirement (if applicable) + Insurance (if applicable).

Cardinality

- The cardinality of ISA relationship between Payout, Payments, and Deductions is (1: 1).

Relationship 5: Paid



Relation

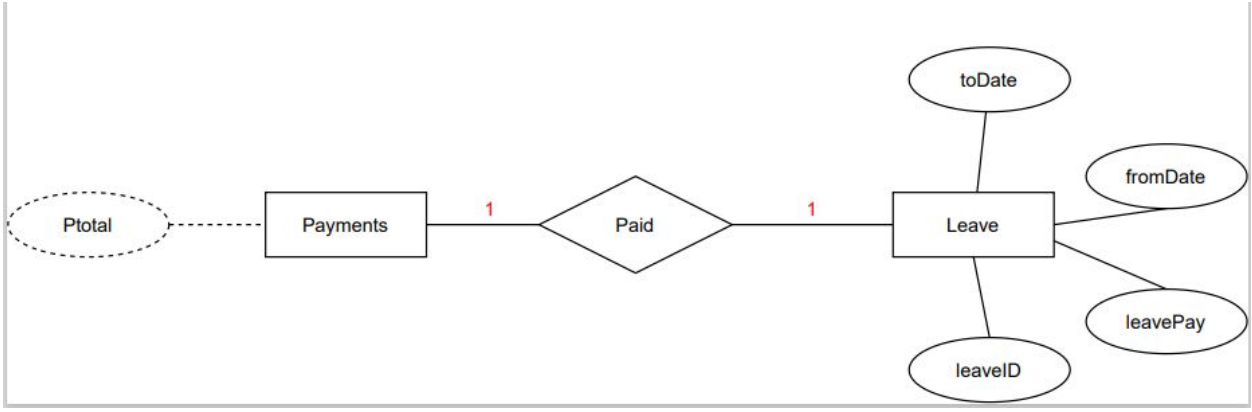
- Shows the relation between entities Payments and Salary.

Attributes

- EID (Foreign Key in Salary from Employee): This key directly pushes the Employee information to entity Salary, with this foreign key users will be able to see how much a specific employee gets on their pay stub.
- PID (Foreign Key in Salary from Payout): This key directly pushes the Payout information to entity Salary, with this foreign key users will be able to see what paystub it is.
- Ptotal derived from baseSalary + Bonus (if applicable) + leavePay.

Cardinality

- The cardinality of Paid relationship between Payments and Salary is (1: 1).



Relation

- Shows relation between entities Payments and Leave

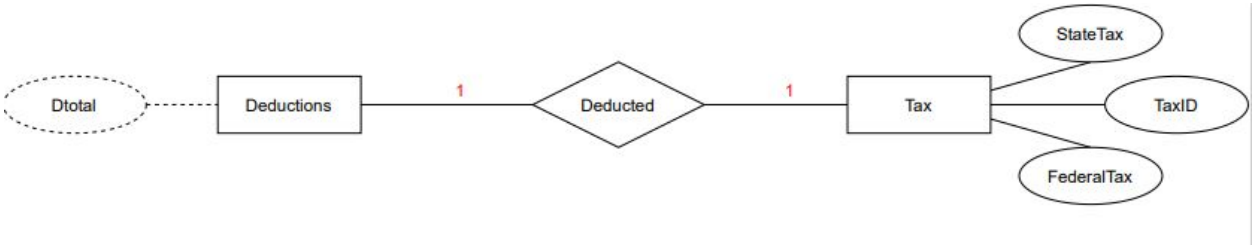
Attributes

- EID (Foreign Key in Leave from Employee): This key directly pushes the Employee information to entity Payout, with this foreign key users will be able to see how much a specific employee gets on their pay stub.
- PID (Foreign Key in Leave from Payout): This key directly pushes the Payout information to entity Leave, with this foreign key users will be able to see what paystub it is.
- Ptotal derived from baseSalary + Bonus (if applicable) + leavePay.

Cardinality

- The cardinality of Paid relationship between Payments and Leave is (1: 1).

Relationship 6: Deducted



Relation

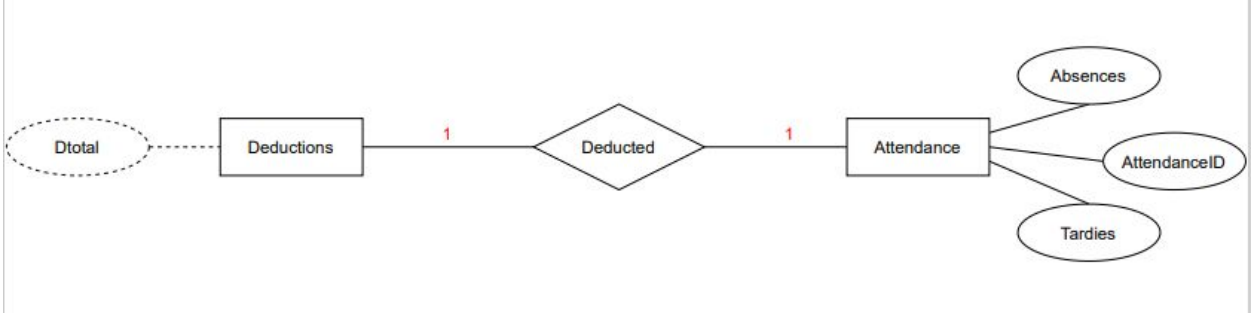
- Shows the relation between entities Deductions and Tax.

Attributes

- EID (Foreign Key in Tax from Employee): This key directly pushes the Employee information to entity Payout, with this foreign key users will be able to see how much a specific employee gets on their pay stub.
- PID (Foreign Key in Tax from Payout): This key directly pushes the Payout information to entity Tax, with this foreign key users will be able to see what paystub it is.
- Depending on the location of the employees the tax deduction can determine which state tax to deduct from that employee's paycheck.
- Dtotal derived from StateTax + FederalTax + Retirement (if applicable) + Insurance (if applicable).

Cardinality

- The cardinality of Deducted relationship between Deductions and Tax is (1: 1).



Relation

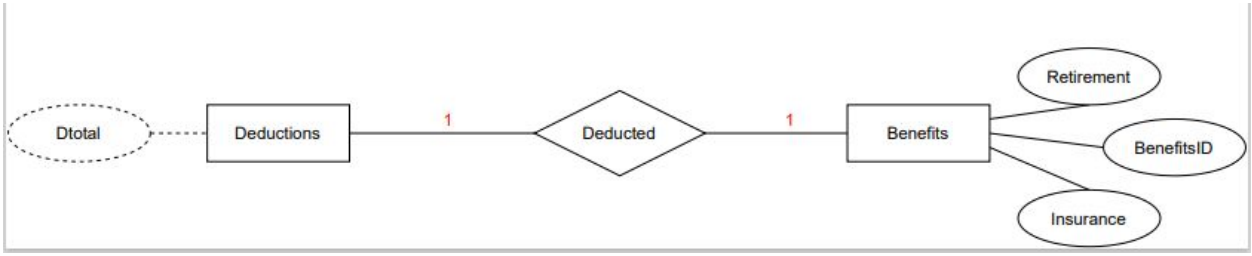
- Shows the relations between entities Deductions and Attendance.

Attributes

- EID (Foreign Key in Attendance from Employee): This key directly pushes the Employee information to entity Attendance, with this foreign key users will be able to see how much a specific employee gets on their pay stub.
- PID (Foreign Key in Attendance from Payout): This key directly pushes the Payout information to entity Attendance, with this foreign key users will be able to see what paystub it is.
- Dtotal derived from StateTax + FederalTax + Retirement (if applicable) + Insurance (if applicable).

Cardinality

- The cardinality of Deducted relationship between Deductions and Attendance is (1: 1).



Relation

- Shows the relations between entities Deductions and Benefits.

Attributes

- EID (Foreign Key in Benefits from Employee): This key directly pushes the Employee information to entity Benefits, with this foreign key users will be able to see how much a specific employee gets on their pay stub.
- PID (Foreign Key in Benefits from Payout): This key directly pushes the Payout information to entity Benefits, with this foreign key users will be able to see what paystub it is.

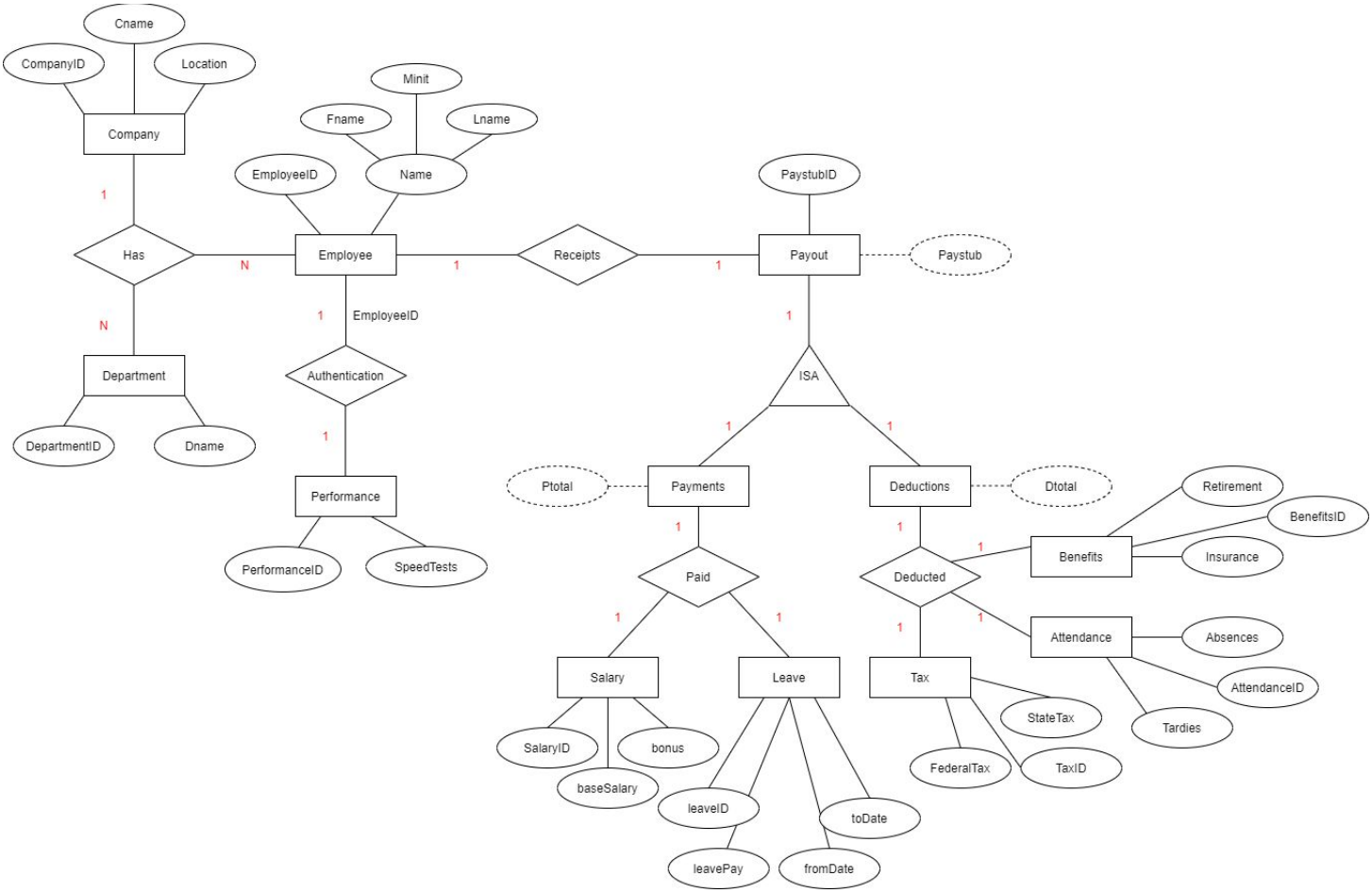
- Dtotal derived from StateTax + FederalTax + Retirement (if applicable) + Insurance (if applicable).

Cardinality

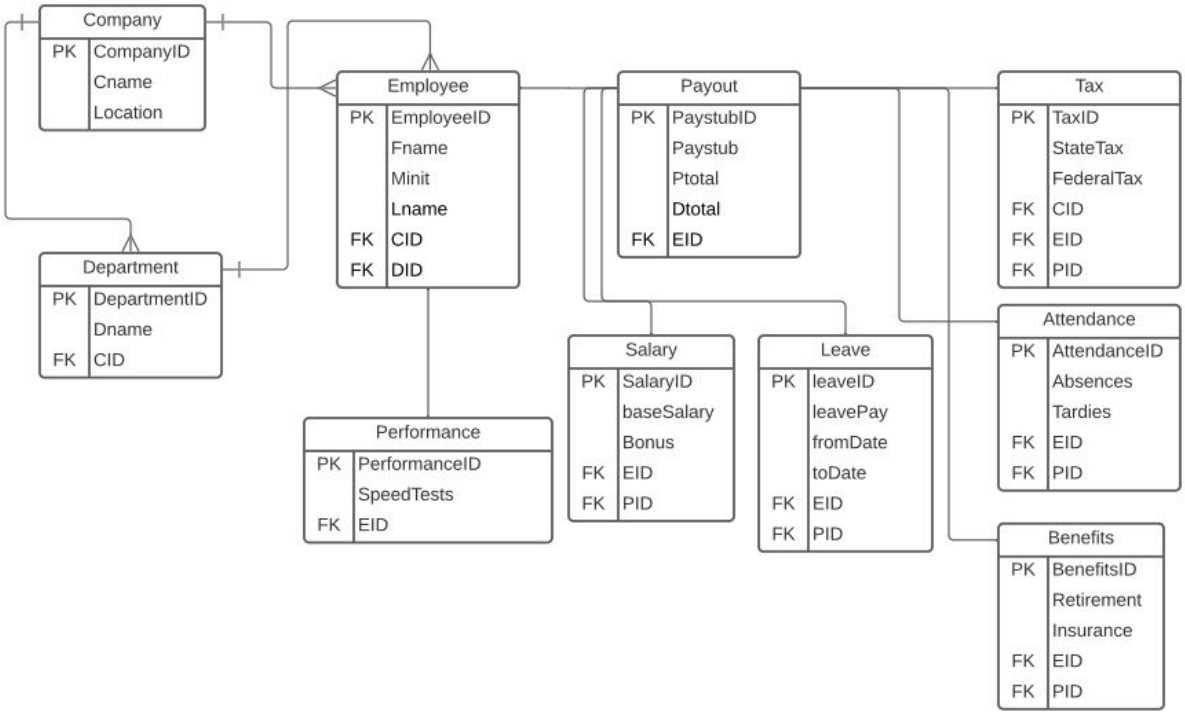
- The cardinality of Deducted relationship between Deductions and Benefits is (1: 1).

Mapping

ER Diagram with Cardinalities



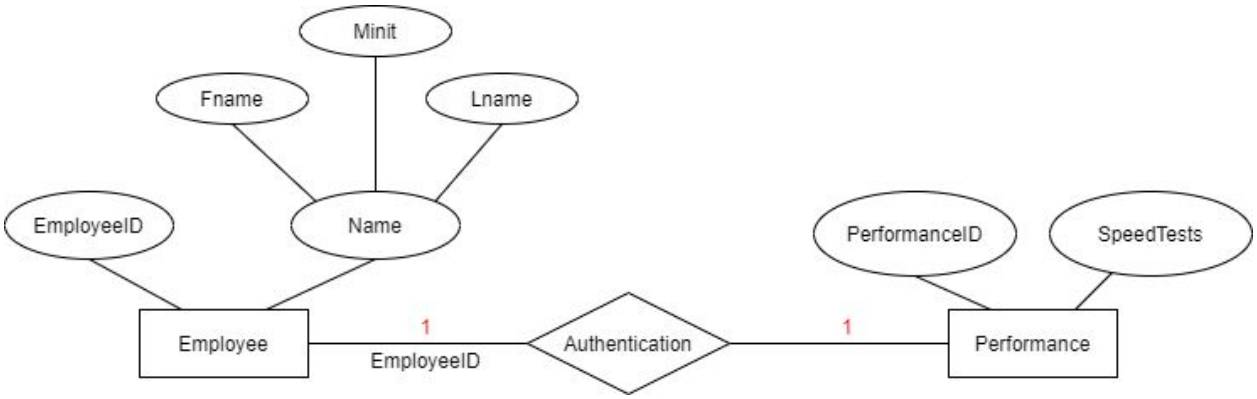
Relational Model



First we began with mapping regular entity types. All of the entities in our ER model are regular, therefore, we inserted all of the attributes in each entity to their respective entities. For example, Employee (EmployeeID, Fname, Minit, Lname). Then we also chose a primary key for each table. Afterwards, following the ER-to-Relational Mapping Algorithm, we mapped the Binary 1:1 Relation Types. In our ER Diagram most of our relationships are 1:1. Afterwards we mapped the 1:N relationships.

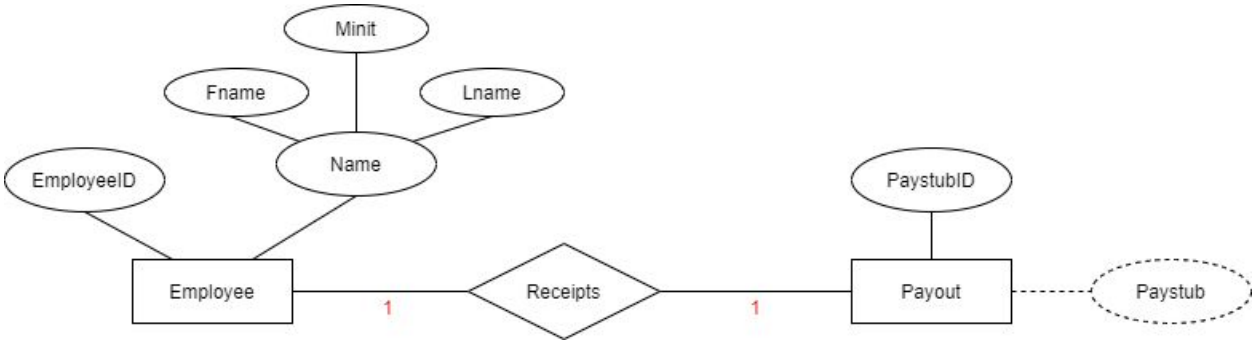
Mapping 1:1 Relationships

Employee - Authentication - Performance



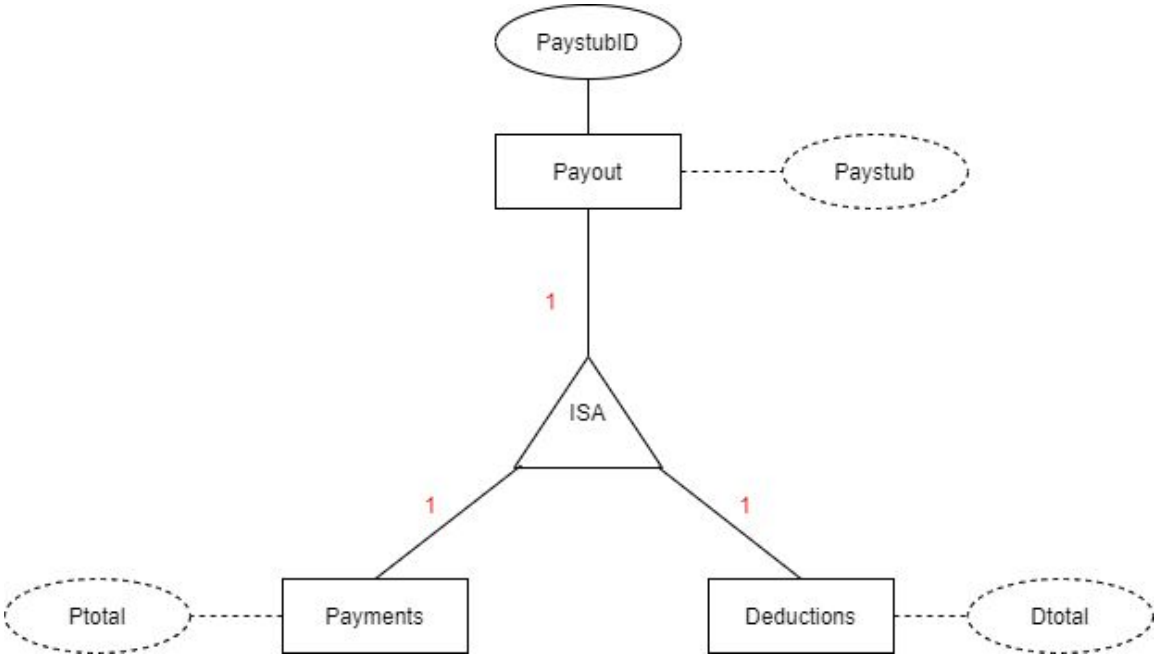
Here we used the foreign key approach. We chose the relation Performance to include a foreign key, EID (EmployeeID), which is the primary key of Employee. Therefore: Performance(PerformanceID, SpeedTests, FK: EID)

Employee - Receipts - Payout



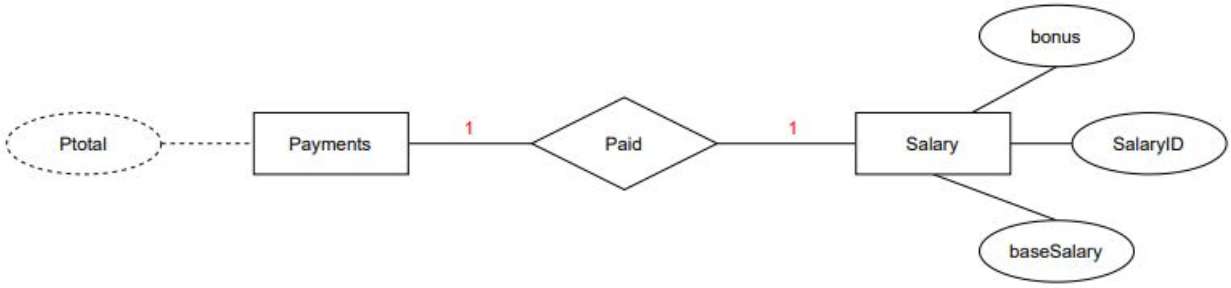
Here we used the foreign key approach. We chose the relation Payout to include foreign key, EID (EmployeeID), which is the primary key of Employee. Therefore: Payout(PaystubID, Paystub, FK:EID).

Payout - ISA - Payments & Deductions

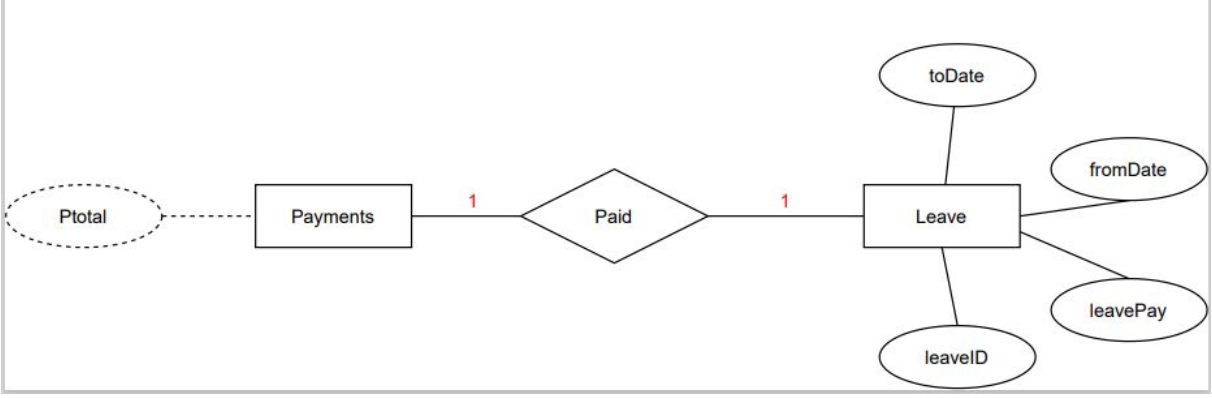


Here we also used the foreign key approach because it is a 1:1 relationship. However, it is also more complicated because it is also an ISA relationship. Therefore the foreign key in this relationship is EmployeeID: Paystub. → Payout(PaystubID, Ptotal, Dtotal, FK: EID)

Payments - Paid - Salary & Leave

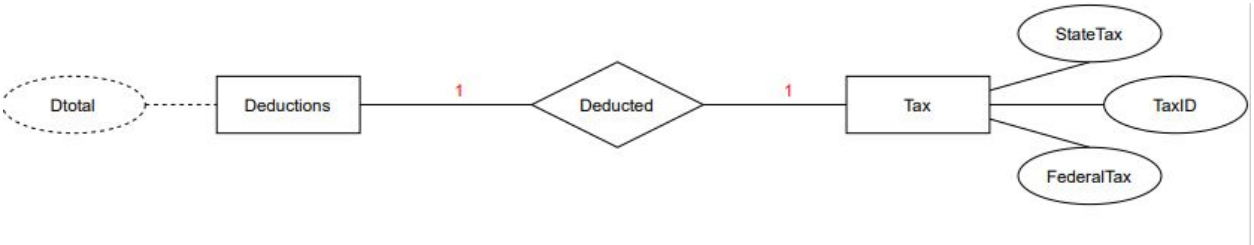


Here we also used the foreign key approach. The primary key of Payout, because Payments' attribute is in Payout relation, is PaystubID. The foreign keys are: EmployeeID and PaystubID Therefore: Salary(SalaryID, baseSalary, Bonus, FK: EID, FK:PID)

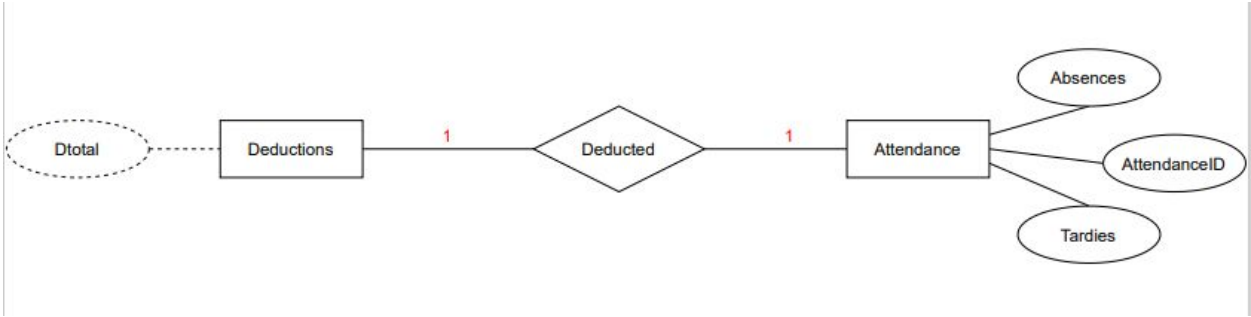


Here we also used the foreign key approach. The primary key of Payout, because Payments' attribute is in Payout relation, is PaystubID. The foreign keys are: EmployeeID and PaystubID. Therefore: Leave(leaveID, leavePay, fromDate, toDate, FK: EID, FK: PID)

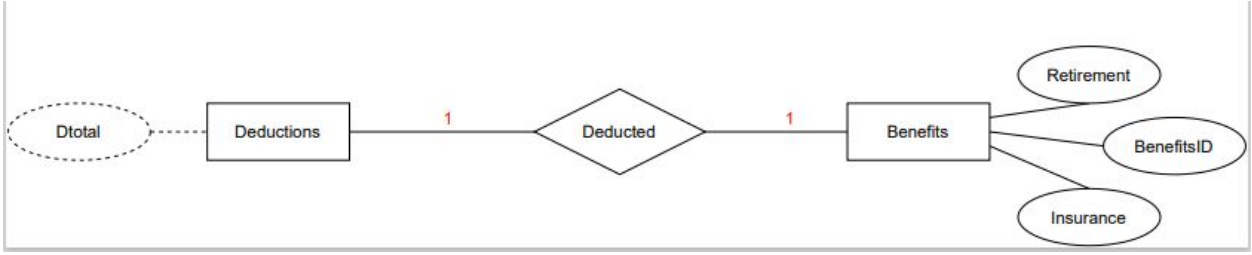
Deductions - Deducted - Tax, Attendance, & Benefits



Here we used the foreign key approach. The primary key of Payout, because Deductions' attribute is in Payout relation, is PaystubID. The foreign keys are: CompanyID, EmployeeID, and PayoutID. CompanyID will be able to determine the StateTax because it is based on location. Therefore: Tax(TaxID, StateTax, FederalTax, FK: CID, FK: EID, FK: PID)



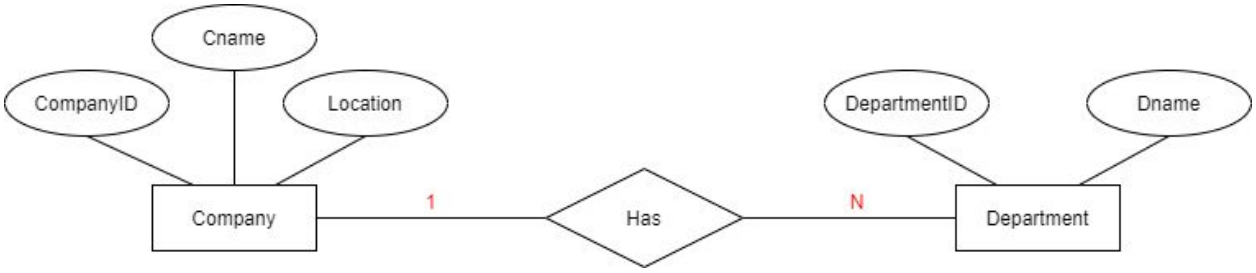
Here we used the foreign key approach. The primary key of Payout, because Deductions' attribute is in Payout relation, is PaystubID. The foreign keys are: EmployeeID and PaystubID. Therefore: Attendance (AttendanceID, Absences, Tardies, FK: EID, FK: PID)



Here we used the foreign key approach. The primary key of Payout, because Deductions' attribute is in Payout relation, is PaystubID. The foreign keys are: EmployeeID and PaystubID. Therefore: Benefits(BenefitsID, Retirement, Insurance, FK: EID, FK: PID)

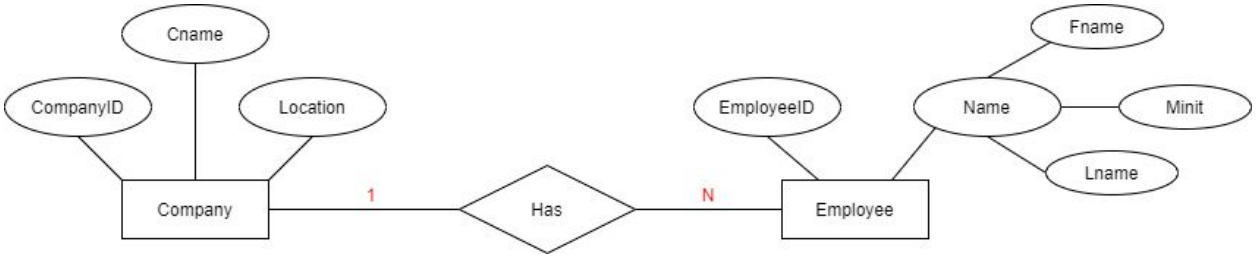
Mapping 1:N Relationships

Company - Has - Department



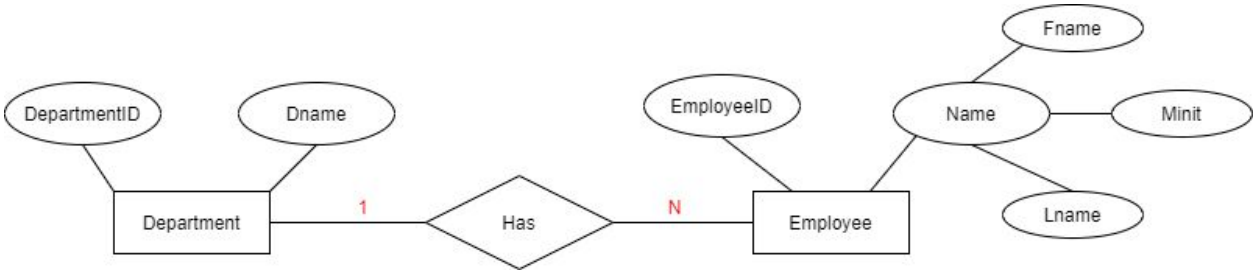
Here we are mapping a 1:N relationship with the Department relation being the N-side. Therefore, we have to include the primary key of Company - CompanyID as the foreign key in the Department relation and include any simple attribute of Has relationship (does not have any). Therefore: Department(DepartmentID, Dname, FK: CID)

Company - Has - Employee



Here we are mapping a 1:N relationship with the Employee relation being the N-side. Therefore, we have to include the primary key of Company - CompanyID as the foreign key in the Employee relation and include any simple attribute of Has relationship (does not have any). Therefore: Employee(EmployeeID, Fname, Minit, Lname, FK: CID)

Department - Has - Employee



Here we are mapping a 1:N relationship with the Employee relation being the N-side. Therefore, we have to include the primary key of Department - DepartmentID as the foreign key in the Employee relation and include any simple attribute of Has relationship (does not have any). Therefore: Employee(EmployeeID, Fname, Minit, Lname, FK: CID, FK: DID)

Normalization

Company (<u>CompanyID</u>, Cname, Location)

Company *C*(*CIDCnL*) - Functional Dependencies (FDs) {*CID* → *Cn*, *CID* → *L*}

(a) List all candidate keys for Company.

- *CIDCnL* - all attributes are always a superkey
- *CIDL* - can get attribute *Cn* through *CID* → *Cn*
- *CIDCn* - can get attribute *L* through *CID* → *L*
- *CID* - can get attribute *Cn* through *CID* → *Cn*, and then get attribute *L* through *CID* → *L*
- Therefore the candidate key is: *CID*.

(b) Is Company in 3NF?

- *CID* → *Cn*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? Yes it is a candidate key.
  (c) Is the right side a prime attribute? No.
- *CID* → *L*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? Yes it is a candidate key.
  (c) Is the right side a prime attribute? No.

(c) Therefore Company relation is in 3NF

Department (<u>DepartmentID</u>, Dname, FK: CID)

Department *D*(*DIDDnCID*) - FDs {*DID* → *Dn*, *CID* → *DID*}

(a) List all candidate keys for Department.

- *DIDDnCID* - all attributes are always a superkey
- *DIDCID* - can get attribute *Dn* through *DID* → *Dn*
- *DnCID* - can get attribute *DID* through *CID* → *DID*
- Therefore the candidate keys are: *DIDCID* and *DnCID*.

(b) Is Department in 3NF?

- *DIDCID* → *Dn*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key - *DnCID*.
- *CID* → *DID*
  (a) Is the right side a member of the left side? No.

(b) Is the left side a super key? No.
(c) Is the right side a prime attribute? Yes it is part of the candidate keys - *DIDCID*.

(c) Therefore Department is in 3NF.

Employee (<u>EmployeeID</u>, Fname, Minit, Lname, FK: CID, FK: DID)
Employee *E*(*EIDFnMLnCIDDID*) - FDs {*EID* → *Fn*, *EID* → *M*, *EID* → *Ln*, *FnLn* → *EID*, *CIDDID* → *EID*}

(a) List all candidate keys for Employee.
- *EIDFnMLnCIDDID* - all attributes are always a superkey
- *EIDMLnCIDDID* - can get attribute *Fn* through *EID* → *Fn*
- *EIDFnLnCIDDID* - can get attribute *M* through *EID* → *M*
- *EIDFnMCIDDID* - can get attribute *Ln* through *EID* → *Ln*
- *FnMLnCIDDID* - can get attribute *EID* through *FnLn* → *EID*
- *FnMLnCIDDID* - can get attribute *EID* through *CIDDID* → *EID*
- *EIDCIDDID* - can get attribute *Fn* through *EID* → *Fn*, then can get attribute *M* through *EID* → *M*, and then can get attribute *Ln* through *EID* → *Ln*
- Candidate keys for Employee are: *EIDCIDDID* and *FnMLnCIDDID*.

(b) Is Employee in 3NF?
- *EID* → *Fn*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *FnMLnCIDDID*.
- *EID* → *M*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *FnMLnCIDDID*.
- *EID* → *Ln*
  (a) Is the right side a member of the left side? No
  (b) Is the left side a super key? No
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *FnMLnCIDDID*
- *FnLn* → *EID*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.

(c) Is the right side a prime attribute? Yes it is part of the candidate key: *EIDCIDDID*.

- *CIDDID → EID*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *EIDCIDDID*.

(c) Therefore Employee is in 3NF.

Performance (<u>PerformanceID</u>, SpeedTests, FK: EID)

Performance *Pe*(*PIDStEID*) - FDs {*PID → St*, *EID → PID*}

(a) List all the candidate keys for Performance.
- *PIDStEID* - all attributes are always a super key
- *PIDEID* - can get attribute *St* through *PID → St*
- *StEID* - can get attribute *PID* through *EID → PID*
- The candidate keys for Performance are: *PIDEID*, *StEID*.

(b) Is Performance in 3NF?
- *PID → St*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *StEID*.
- *EID → PID*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *PIDEID*.

(c) Therefore Performance is in 3NF.

Payout (<u>PaystubID</u>, Paystub, Ptotal, Dtotal, FK: EID)

Payout *Pa*(*PaEIDPsPtDtEID*) - FDs {*PaEID → Pt, PaEID → Dt, PtDt → Ps}*

(a) List all the candidate keys for Payout.
- *PaEIDPsPtDtEID*- all attributes are always a super key
- *PaEIDPsDtEID* - can get attribute *Pt* through *PaEID → Pt*
- *PaEIDPsPtEID* - can get attribute *Dt* through *PaEID → Dt*
- *PaEIDPtDtEID*- can get attribute *Ps* through *PtDt → Ps*

- *PaEIDPsEID*- can get attribute *Pt* through *PaEID* → *Pt*, and then get attribute *Dt* through *PaEID* → *Dt*
- The candidate keys for Payout are: *PaEIDPtDtEID*, *PaEIDPsEID*.

(b) Is Payout in 3NF?
- *PaEID* → *Pt*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *PaEIDPtDtEID*.
- *PaEID*→ *Dt*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? Yes it is a candidate key.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *PaEIDPtDtEID*.
- *PtDt* → *Ps*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *PaEIDPsEID*.

(c) Therefore Payout is in 3NF.

Salary (<u>SalaryID</u>, baseSalary, Bonus, FK: EID, FK: PID)

Salary *S*(*SIDBsBoEIDPID*) - FDs {*SID* → *Bs*, *SID* → *Bo*, *EIDPID* → *SID*}

(a) List all candidate keys for Salary.
- *SIDBsBoEIDPID* - all attributes are always a super key
- *SIDBoEIDPID* - can get attribute *Bs* through *SID* → *Bs*
- *SIDBsEIDPID*- can get attribute *Bo* through *SID* → *Bo*
- *BsBoEIDPID* - can get attribute *SID* through *EIDPID* → *SID*
- *SIDEIDPID* - can get attribute *Bs* through *SID* → *Bs*, and then get attribute *Bo* through *SID* → *Bo*
- The candidate key for Salary is *BsBoEIDPID*, *SIDEIDPID*.

(b) Is Salary in 3NF?
- *SID* → *Bs*
  (a) Is the right side a member of the left side? No
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *BsBoEIDPID*.

- *SID* → *Bo*
    (a) Is the right side a member of the left side? No
    (b) Is the left side a super key? No.
    (c) Is the right side a prime attribute? Yes it is part of the candidate key: *BsBoEIDPID*.
- *EIDPID* → *SID*
    (a) Is the right side a member of the left side? No
    (b) Is the left side a super key? No.
    (c) Is the right side a prime attribute? Yes it is part of the candidate key: *SIDEIDPID*.

(c) Therefore Salary is in 3NF.

Leave (<u>leaveID</u>, leavePay, fromDate, toDate, FK: EID, FK: PID)

Leave *L*(*LIDLpFdTdEIDPID*) - FDs {*LID* → *Lp*, *LID* → *Fd*, *LID* → *Td*, *EIDPID* → *LID*}

(a) List all candidate keys for Leave.
- *LIDLpFdTdEIDPID* - all attributes are always a super key
- *LIDFdTdEIDPID* - can get attribute *Lp* through *LID* → *Lp*
- *LIDLpTdEIDPID* - can get attribute *Fd* through *LID* → *Fd*
- *LIDLpFdEIDPID* - can get attribute *Td* through *LID* → *Td*
- *LpFdTdEIDPID* - can get attribute *LID* through *EIDPID* → *LID*
- *LIDEIDPID* - can get attribute *Lp* through *LID* → *Lp*, then get attribute *Fd* through *LID* → *Fd*, and then get attribute *Td* through *LID* → *Td*
- The candidate keys in Leave are: *LpFdTdEIDPID*, *LIDEIDPID*.

(b) Is Leave in 3NF?
- *LID* → *Lp*
    (a) Is the right side a member of the left side? No.
    (b) Is the left side a super key? No.
    (c) Is the right side a prime attribute? Yes it is part of the candidate key: *LpFdTdEIDPID*.
- *LID* → *Fd*
    (a) Is the right side a member of the left side? No.
    (b) Is the left side a super key? No.
    (c) Is the right side a prime attribute? Yes it is part of the candidate key: *LpFdTdEIDPID*.
- *LID* → *Td*
    (a) Is the right side a member of the left side? No.
    (b) Is the left side a super key? No.

(c) Is the right side a prime attribute? Yes it is part of the candidate key: *LpFdTdEIDPID*.

- *EIDPID → LID*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *LIDEIDPID*.

(c) Therefore Leave is in 3NF

Tax (<u>TaxID</u>, StateTax, FederalTax, FK: CID, FK: EID, FK: PID)

Tax *T*(*TIDSxFxCIDEIDPID*) - FDs {*TID→ Sx*, *TID→ Fx*, *CID → Sx*, *EIDPID → TID*} (Found online that state tax changes can have an affect on federal tax)

(a) List all the candidate keys for Tax.
- *TIDSxFxCIDEIDPID*- all attributes are always a super key
- *TIDFxCIDEIDPID* - can get attribute *Sx* through *TID→ Sx*
- *TIDSxCIDEIDPID* - can get attribute *Fx* through *TID→ Fx*
- *TIDFxCIDEIDPID* - can get attribute *Sx* through *CID → Sx*
- *SxFxCIDEIDPID* - can get attribute *TID* through *EIDPID → TID*
- *TIDCIDEIDPID* - can get attribute *Sx* through *TID→ Sx*, and then get attribute *Fx* through *TID→ Fx*
- The candidate keys for Tax are: *SxFxCIDEIDPID*, *TIDFxCIDEIDPID*, *TIDCIDEIDPID*.

(b) Is Tax 3NF?
- *TID→ Sx*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *SxFxCIDEIDPID*.
- *TID→ Fx*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *SxFxCIDEIDPID*, *TIDFxCIDEIDPID*.
- *CID → Sx*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is part of the candidate key: *SxFxCIDEIDPID*.

- *EIDPID → TID*
    (a) Is the right side a member of the left side? No.
    (b) Is the left side a super key? No.
    (c) Is the right side a prime attribute? Yes it is part of the candidate key: *TIDCIDEIDPID*, *TIDFxCIDEIDPID*.

(c) Therefore Tax is in 3NF.

Attendance (<u>AttendanceID</u>, Absences, Tardies, FK: EID, FK: PID)

Attendance *A*(*AIDAbTaEIDPID*) - FDs {*AID → Ab*, *AID → Ta*, *EIDPID → AID*}

(a) List all the candidate keys for Attendance.
- *AIDAbTaEIDPID*- all attributes will always be a super key
- *AIDTaEIDPID* - can get attribute *Ab* through *AID → Ab*
- *AIDAbEIDPID* - can get attribute *Ta* through *AID → Ta*
- *AIDEIDPID* - can get attribute *Ab* through *AID → Ab*, and then get attribute *Ta* through *AID → Ta*
- *AbTaEIDPID* - can get attribute *AID*through *EIDPID → EIDPID*
- The candidate keys for Attendance are: *AIDEIDPID*, *AbTaEIDPID*.

(b) Is Attendance in 3NF?
- *AID → Ab*
    (a) Is the right side a member of the left side? No.
    (b) Is the left side a super key? No.
    (c) Is the right side a prime attribute? Yes it is part of the candidate key: *AbTaEIDPID*.
- *AID→ Ta*
    (a) Is the right side a member of the left side? No.
    (b) Is the left side a super key? No.
    (c) Is the right side a prime attribute? Yes it is part of the candidate key: *AbTaEIDPID*.
- *EIDPID → AID*
    (a) Is the right side a member of the left side? No.
    (b) Is the left side a super key? No.
    (c) Is the right side a prime attribute? Yes it is part of the candidate key: *AIDEIDPID*.

(c) Therefore Attendance is in 3NF.

Benefits (<u>BenefitsID</u>, Retirement, Insurance, FK: EID, FK: PID)

Benefits *B*(*BIDRIEIDPID*) - FDs {*BID* → *R*, *BID* → *I*, *EIDPID* → *BID*}

(a) List all the candidate keys for Benefits.

- *BIDRIEIDPID* - all attributes will always be a super key
- *BIDIEIDPID* - can get attribute *R* through *BID*→ *R*
- *BIDREIDPID* - can get attribute *I* through *BID*→ *I*
- *RIEIDPID* - can get attribute *BID* through *EIDPID* → *BID*
- *BIDEIDPID* - can get attribute *R* through *BID*→ *R*, and then get attribute *I* through *BID*→ *I*
- The candidate key for Benefits is: *RIEIDPID*, *BIDEIDPID*.

(b) Is Benefits in 3NF?

- *BID* → *R*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is a part of the candidate key: *RIEIDPID*.
- *BID*→ *I*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is a part of the candidate key: *RIEIDPID*.
- *EIDPID* → *BID*
  (a) Is the right side a member of the left side? No.
  (b) Is the left side a super key? No.
  (c) Is the right side a prime attribute? Yes it is a part of the candidate key: *BIDEIDPID*.

(c) Therefore Benefits is in 3NF.

Data Dictionary

Table	Attribute	Data Type	Primary Key	Foreign Key	Constraints
company	Company_ID	INT	Yes		
company	Cname	VARCHAR(25)			
company	Location	VARCHAR(25)			
department	Department_ID	INT	Yes		
department	Dname	VARCHAR(25)			
department	CID	INT		company(CompanyID)	
employee	EmployeeID	INT	Yes		6 digits, unsigned
employee	Fname	VARCHAR(25)			
employee	Minit	CHAR(1)			
employee	Lname	VARCHAR(25)			
employee	Company_ID	INT		company(Company_ID)	
employee	DID	INT		department(Department_ID)	
performance	PerformanceID	INT	Yes		
performance	SpeedTests	TIME			hh:mm:ss[.nnnnnn]
performance	Employee_ID	INT		employee(EmployeeID)	6 digits, unsigned
payout	PaystubID	INT	Yes		3 digits, unsigned
payout	EID	INT		employee(EmployeeID)	6 digits, unsigned

salary	Salary_ID	INT	Yes		
salary	baseSalary	INT			In thousands (k)
salary	Bonus	INT			
salary	EmpID	INT		employee(EmployeeID)	6 digits, unsigned
salary	PID	INT		payout(PayoutID)	3 digit, unsigned
leavep	leaveID	INT	Yes		5 digits, unsigned
leavep	leavePay	INT			
leavep	fromDate	DATE			Format YYYY-MM-DD
leavep	toDate	DATE			Format YYYY-MM-DD
leavep	Em_ID	INT		employee(EmployeeID)	6 digits, unsigned
leavep	Ps_ID	INT		payout(PayoutID	3 digits, unsigned
tax	Tax_ID	INT	Yes		
tax	StateTax	DECIMAL			(5,2)
tax	FederalTax	DECIMAL			(5,2)
tax	Co_ID	INT		company(Company_ID)	
tax	E_ID	INT		employee(EmployeeID)	6 digits, unsigned
tax	P_ID	INT		payout(PayoutID)	3 digits, unsigned
attendance	Attendance_ID	INT	Yes		
attendance	Absences	INT			
attendance	Tardies	INT			

attendance	EmID	INT		employee(EmployeeID)	6 digits, unsigned
attendance	PaID	INT		payout(PayoutID)	3 digits, unsigned
benefits	Benefits_ID	INT	Yes		
benefits	Retirement	INT			
benefits	Insurance	INT			
benefits	EmployID	INT		employee(EmployeeID)	6 digits, unsigned
benefits	Pay_ID	INT		payout(PayoutID)	3 digits, unsigned
payments (view)	EmpID	INT			6 digits, unsigned (from employee)
payments (view)	Fname	VARCHAR(25)			From employee
payments (view)	Lname	VARCHAR(25)			From employee
payments (view)	payments_Total	INT			Derived attribute
paystub (view)	EmployeeID	INT			6 digits, unsigned (from employee)
paystub (view)	Fname	VARCHAR(25)			From employee
paystub (view)	Lname	VARCHAR(25)			From employee
paystub (view)	payments_Total	INT			Derived attribute (from payments (view))
paystub (view)	tax_deduction_total	INT			Derived attribute(from tax)
paystub (view)	benefits_deduction_total	INT			Derived attribute(from benefits)
paystub (view)	paystub	INT			Derived attribute
paystub (view)	Absences	INT			From attendance

paystub (view)	Tardies	INT			From attendance
payout_view (view)	PaystubID	INT			3 digits, unsigned (from payout)
payout_view (view)	EID	INT			6 digits, unsigned (from payout)
payout_view (view)	payments_Total	INT			Derived attribute (from payments (view))
payout_view (view)	deductions_Total	INT			Derived attribute
payout_view (view)	paystub	INT			Derived attribute (from paystub (view))

Implementation

Functional Requirements

1. Find the name of the Company an Employee works for with Employee Company_ID and Company Cname.

Company_ID	Fname	Lname	Cname	Location
1	Kim	Boucha	Companee	Atlanta
1	Hanson	Deck	Companee	Atlanta
1	Victor	Eeyus	Companee	Atlanta
1	Lisa	Newcar	Companee	Atlanta
1	Jason	Response	Companee	Atlanta
2	John	Apple	WorkIt	New York
2	Jesse	Gonzalez	WorkIt	New York
2	April	Ludgate	WorkIt	New York
2	John	Smith	WorkIt	New York
2	Jane	Thompson	WorkIt	New York
5	Scooby	Doo	PrimeFinacial	Seattle
5	Junie	Jones	PrimeFinacial	Seattle
5	Daniel	Miller	PrimeFinacial	Seattle
5	Arya	Stark	PrimeFinacial	Seattle
5	Alice	Wonder	PrimeFinacial	Seattle

2. Find the name of the Department an Employee is a part of with Employee EmployeeID and Department Dname. Display the name of the Company they work for as well.

EmployeeID	Fname	Lname	Dname	Cname
500005	Alice	Wonder	Information Security	PrimeFinacial
200003	April	Ludgate	Reception	WorkIt
500003	Arya	Stark	Diversity/Inclusion	PrimeFinacial
500001	Daniel	Miller	Marketing	PrimeFinacial
100001	Hanson	Deck	App Development	Companee
200002	Jane	Thompson	Management	WorkIt
100005	Jason	Response	DevOps	Companee
200005	Jesse	Gonzalez	Human Resources	WorkIt
200004	John	Apple	Management	WorkIt
200001	John	Smith	Human Resources	WorkIt
500002	Junie	Jones	Marketing	PrimeFinacial
100003	Kim	Boucha	Engineering	Companee
100002	Lisa	Newcar	Engineering	Companee
500004	Scooby	Doo	Diversity/Inclusion	PrimeFinacial
100004	Victor	Eeyus	Engineering	Companee

3. Find the speed test of an employee with an Employee EmployeeID and Performance SpeedTests. Order by the fastest time and companyID. Display the name of the company as well.

SpeedTests	Company_ID	Fname	Lname	Cname
NULL	1	Lisa	Newcar	Companee
NULL	1	Victor	Eeyus	Companee
NULL	1	Jason	Response	Companee
NULL	2	John	Smith	WorkIt
NULL	2	Jane	Thompson	WorkIt
NULL	2	April	Ludgate	WorkIt
NULL	2	John	Apple	WorkIt
NULL	2	Jesse	Gonzalez	WorkIt
NULL	5	Daniel	Miller	PrimeFinacial
NULL	5	Arya	Stark	PrimeFinacial
00:43:10	1	Kim	Boucha	Companee
00:45:30	5	Junie	Jones	PrimeFinacial
00:58:40	1	Hanson	Deck	Companee
01:00:00	1	Hanson	Deck	Companee
01:01:10	5	Alice	Wonder	PrimeFinacial
01:06:10	5	Scooby	Doo	PrimeFinacial

4. Find an employee's payout with Employee EmployeeID, Payout PaystubID, and Payout Paystub. Display the first name and last name of the employee and the Company's name.

PaystubID	Cname	Fname	Lname	paystub
101	Companee	Hanson	Deck	53400.00
102	Companee	Lisa	Newcar	45040.00
103	Companee	Kim	Boucha	62560.00
104	Companee	Victor	Eeyus	57130.00
105	Companee	Jason	Response	70420.00
201	WorkIt	John	Smith	31390.00
202	WorkIt	Jane	Thompson	60852.00
203	WorkIt	April	Ludgate	32190.00
204	WorkIt	John	Apple	50544.00
205	WorkIt	Jesse	Gonzalez	35120.00
501	PrimeFinacial	Daniel	Miller	61800.00
502	PrimeFinacial	Junie	Jones	11432.00
503	PrimeFinacial	Arya	Stark	42570.00
504	PrimeFinacial	Scooby	Doo	45420.00
505	PrimeFinacial	Alice	Wonder	66740.00

5. Find an employee's payments received with Payments Ptotal and Employee EmployeeID. Display the BaseSalary, Bonus, and LeavePay. Order by BaseSalary.

Company_ID	Fname	Lname	BaseSalary	Bonus	leavePay	payments_Total
1	Lisa	Newcar	40000	4000	10000	54000
2	John	Smith	40000	500	1000	41500
2	April	Ludgate	40000	500	1000	41500
2	Jesse	Gonzalez	40000	0	2000	42000
1	Hanson	Deck	50000	5000	10000	65000
1	Victor	Eeyus	55000	5500	10000	70500
5	Arya	Stark	55000	1000	2000	58000
5	Scooby	Doo	55000	1000	2000	58000
1	Kim	Boucha	60000	6000	10000	76000
2	John	Apple	65000	900	2000	67900
1	Jason	Response	70000	7000	10000	87000
2	Jane	Thompson	70000	1200	2000	73200
5	Daniel	Miller	72000	2000	1000	75000
5	Alice	Wonder	80000	1500	2000	83500
5	Junie	Jones	130000	3300	1000	134300

6. Find an employee's deductions received with Deductions Dtotal and Employee EmployeeID. Display Tax deductions and Benefits deductions.

Company_ID	Fname	Lname	tax_deduction_total	benefits_deduction_total	deductions_Total
2	John	Smith	5810.00	4300	10110.00
2	April	Ludgate	5810.00	3500	9310.00
2	Jesse	Gonzalez	5880.00	1000	6880.00
1	Lisa	Newcar	7560.00	1400	8960.00
1	Hanson	Deck	9100.00	2500	11600.00
5	Arya	Stark	9280.00	6150	15430.00
5	Scooby	Doo	9280.00	3300	12580.00
2	John	Apple	9506.00	7850	17356.00
1	Victor	Eeyus	9870.00	3500	13370.00
2	Jane	Thompson	10248.00	2100	12348.00
1	Kim	Boucha	10640.00	2800	13440.00
5	Daniel	Miller	12000.00	1200	13200.00
1	Jason	Response	12180.00	4400	16580.00
5	Alice	Wonder	13360.00	3400	16760.00
5	Junie	Jones	21488.00	3100	24588.00

7. Find all salaries that are greater than 50000. Order by company name and department name.

Cname	Dname	Fname	Lname	BaseSalary
Companee	DevOps	Jason	Response	70000
Companee	Engineering	Kim	Boucha	60000
Companee	Engineering	Victor	Eeyus	55000
PrimeFinacial	Diversity/Inclusion	Arya	Stark	55000
PrimeFinacial	Diversity/Inclusion	Scooby	Doo	55000
PrimeFinacial	Information Security	Alice	Wonder	80000
PrimeFinacial	Marketing	Daniel	Miller	72000
PrimeFinacial	Marketing	Junie	Jones	130000
WorkIt	Management	Jane	Thompson	70000
WorkIt	Management	John	Apple	65000

8. Find all the employees that had a leave fromDate and toDate in 2020. Order by Company name.

Cname	Fname	Lname	fromDate	toDate
Companee	Lisa	Newcar	2020-11-25	2020-12-01
Companee	Victor	Eeyus	2020-10-25	2020-11-01
Companee	Hanson	Deck	2020-11-28	2020-12-13
Companee	Kim	Boucha	2020-09-25	2020-10-10
Companee	Jason	Response	2020-06-25	2020-07-08
PrimeFinacial	Alice	Wonder	2020-12-13	2020-12-27
WorkIt	April	Ludgate	2020-08-19	2020-09-01
WorkIt	John	Apple	2020-01-13	2020-01-26
WorkIt	John	Smith	2020-02-19	2020-03-01
WorkIt	Jane	Thompson	2020-04-13	2020-04-26
WorkIt	Jesse	Gonzalez	2020-10-16	2020-10-28

9. Find how much an employee's payout goes to taxes with Tax StateTax, FederalTax and Employee EmployeeID. Order by Company name.

Cname	Fname	Lname	payments_Total	tax_deduction_total
Companee	Hanson	Deck	65000	9100.00
Companee	Lisa	Newcar	54000	7560.00
Companee	Kim	Boucha	76000	10640.00
Companee	Victor	Eeyus	70500	9870.00
Companee	Jason	Response	87000	12180.00
PrimeFinacial	Daniel	Miller	75000	12000.00
PrimeFinacial	Junie	Jones	134300	21488.00
PrimeFinacial	Arya	Stark	58000	9280.00
PrimeFinacial	Scooby	Doo	58000	9280.00
PrimeFinacial	Alice	Wonder	83500	13360.00
WorkIt	John	Smith	41500	5810.00
WorkIt	Jane	Thompson	73200	10248.00
WorkIt	April	Ludgate	41500	5810.00
WorkIt	John	Apple	67900	9506.00
WorkIt	Jesse	Gonzalez	42000	5880.00

10. Find which employees have greater than 3 absences and tardies. Order by company name and department name.

Cname	Dname	Fname	Lname	Absences	Tardies
Companee	DevOps	Jason	Response	6	5
PrimeFinacial	Diversity/Inclusion	Arya	Stark	5	4

11. Find out how much an employee's payout goes to benefits with BenefitsID, Benefits Retirement, Benefits Insurance and EmployeeID. Order by Company name.

Cname	Fname	Lname	payments_Total	benefits_deduction_total
Companee	Hanson	Deck	65000	2500
Companee	Lisa	Newcar	54000	1400
Companee	Kim	Boucha	76000	2800
Companee	Victor	Eeyus	70500	3500
Companee	Jason	Response	87000	4400
PrimeFinacial	Daniel	Miller	75000	1200
PrimeFinacial	Junie	Jones	134300	3100
PrimeFinacial	Arya	Stark	58000	6150
PrimeFinacial	Scooby	Doo	58000	3300
PrimeFinacial	Alice	Wonder	83500	3400
WorkIt	John	Smith	41500	4300
WorkIt	Jane	Thompson	73200	2100
WorkIt	April	Ludgate	41500	3500
WorkIt	John	Apple	67900	7850
WorkIt	Jesse	Gonzalez	42000	1000

Summary

Summary of Work

Overall building this database was difficult but manageable. We found that nothing was set in stone until we completed the implementation phase of this project. We constantly had to update and in some cases completely scrap what we had already made. Here are some challenges that we encountered:

1. Setting up the table/relational model in mySQL Workbench.
    - The main problem that arose when adding the tables was through the payout table. If you reference our relational model you can see that we wanted to include the following columns in our payout table: PK:PaystubID, Ptotal, Dtotal, Paystub, and the FK: EID. However, since most of those attributes were derived attributes those are not recommended to be in a regular table but a view. A view in SQL is a virtual table based on the result-set of a SQL statement.
2. Normalization
    - This problem also ties into our first challenge as well. At first we did not intend to have all of those IDs as primary keys in our table. For example, we wanted to have in the tax table: StateTax, and FK: EmployeeID together as our primary key but upon further research found that it was not possible. Also when we tried to do them alone, only StateTax, we found the console throwing us errors because it contained duplicate and since it is supposed to be a primary key (unique) that is not allowed. Therefore we decided to add the mostly auto-incremented IDs in our tables.
3. Attendance's attribute (Absences) in Deductions
    - At first we intended to also have Absences contribute to the derived attribute Dtotal (deductions_Total) but after giving it more thought we decided against it. For it to actually make sense we would have to add more attributes to the employee table such as hourly rate and hours worked. We still kept it as part of the deducted relationship because we do eventually want to fix it.
4. Time-Conflicts
    - Usually whenever one team member was free the others were not, so it was difficult to find a time where all of us could meet. COVID-19, also made it difficult since we were not able to meet up in person to talk through the challenges that we encountered.

Contributions

- Redghy Jean: Project Topic, ER Model, Mapping, Conceptual Design (entities and attributes), Relationships (relations and attributes), Normalization, Data Dictionary, Implementation (create), Summary, PowerPoint Presentation

- Dani Major: Introduction, ER Model with Cardinality Ratios, Conceptual Design (cardinalities), Normalization, Data Dictionary, Implementation (inserts), Summary, PowerPoint Presentation
- Juana Tavera: Requirement Analysis (user requirements, constraints, data requirements, and functional requirements), Mapping, Normalization, Data Dictionary, Implementation (views and queries), Summary, Demo Video, PowerPoint Presentation

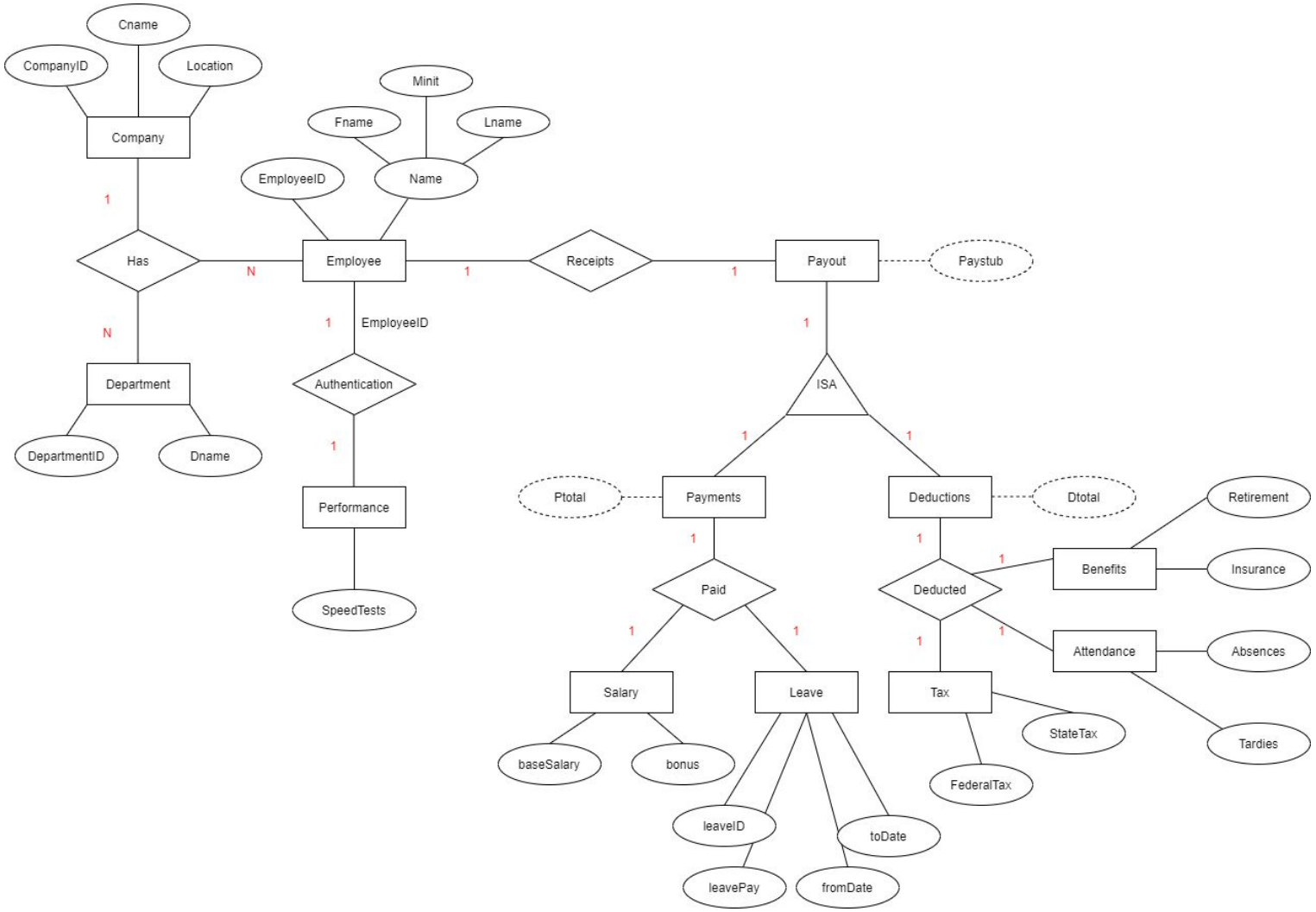
Access Link:

Demo Video (Functional Requirements Only):
https://drive.google.com/file/d/1ebIBd_OG7SKfoMjXY6Muc75lmOGYk2Rt/view?usp=sharing

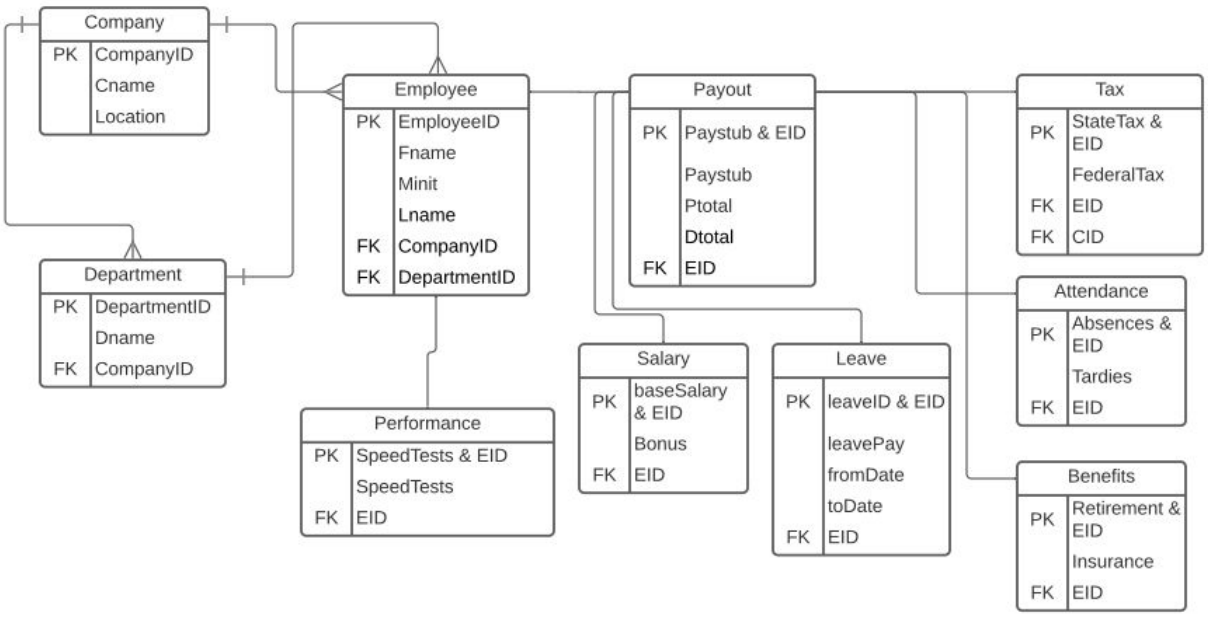
Demo Video (Tables, Views, and Functional Requirements):
https://drive.google.com/file/d/1NziLcseoj6x9dQCec9ElMQ_ZzvgfzJRF/view?usp=sharing

Appendix

ER Diagram with Cardinalities



Relational Model



SQL Code

Functional Requirements

/*Functional Requirement 1 - Find the name of the Company an employee works for with Employee EmployeeID, Employee CID, and Company Cname*/

```
SELECT
        EMP.Company_ID,
   EMP.Fname,
   EMP.Lname,
   COM.Cname,
   COM.Location
FROM payroll.employee AS EMP
        LEFT JOIN payroll.company AS COM
   ON EMP.Company_ID =  COM.Company_ID
ORDER BY
        EMP.Company_ID,
   EMP.Lname;
```

```
/*Functional Requirement 2 - Find the name of the Department an Employee is a part of with
EmployeeID and Department Dname. Display the name of the Company they work for as well*/

SELECT
        EMP.EmployeeID,
   EMP.Fname,
   EMP.Lname,
   DEPT.Dname,
   COM.Cname
FROM payroll.employee AS EMP
        LEFT JOIN payroll.department AS DEPT
   ON EMP.DID = DEPT.Department_ID
   LEFT JOIN payroll.company AS COM
   ON EMP.Company_ID = COM.Company_ID
ORDER BY
        EMP.Fname,
   EMP.Lname;

/*Functional Requirement 3- Find the Speed Tests of an employee with an EmployeeID and
Performance SpeedTests. Order by the fastest time and company ID. Display the name of the
company as well.*/

SELECT
        PER.SpeedTests,
   COM.Company_ID,
   EMP.Fname,
   EMP.Lname,
   COM.Cname
FROM payroll.employee AS EMP
        LEFT JOIN payroll.performance AS PER
   ON EMP.EmployeeID = PER.Employee_ID
   LEFT JOIN payroll.company AS COM
   ON EMP.Company_ID = COM.Company_ID
ORDER BY
        PER.SpeedTests,
   COM.Company_ID;

/*Functional Requirement 4 - Find an employee's payout with Employee EmployeeID, Payout
PaystubID, and Payout Paystub. Display the first name and last name of the employee and the
Company's name.*/
```

```
SELECT
        PAY.PaystubID,
   COM.Cname,
   EMP.Fname,
   EMP.Lname,
   PTUB.paystub
FROM payroll.employee AS EMP
        LEFT JOIN payroll.payout AS PAY
   ON EMP.EmployeeID = PAY.EID
   LEFT JOIN payroll.paystub AS PTUB
   ON EMP.EmployeeID = PTUB.EmployeeID
   LEFT JOIN payroll.company AS COM
   ON EMP.Company_ID = COM.Company_ID
ORDER BY
        PAY.PaystubID

/*Functional Requirement 5 - Find an employee's payments received with Payments Ptotal and
Employee EmployeeID. Display the BaseSalary, Bonus, and LeavePay*/

SELECT
        COM.Company_ID,
        EMP.Fname,
   EMP.Lname,
   SAL.BaseSalary,
   SAL.Bonus,
   LEA.leavePay,
   (IFNULL(SAL.BaseSalary,0) + IFNULL(SAL.Bonus,0) + IFNULL(LEA.leavePay,0)) as
payments_Total
FROM payroll.employee AS EMP
        LEFT JOIN payroll.salary AS SAL
   ON EMP.EmployeeID = SAL.EmpID
   LEFT JOIN payroll.leavep AS LEA
   ON EMP.EmployeeID = LEA.Em_ID
   LEFT JOIN payroll.company AS COM
   ON EMP.Company_ID = COM.Company_ID
GROUP BY
        EMP.Fname,
   EMP.Lname
ORDER BY
```

```
        SAL.BaseSalary;

/*Functional Requirement 6 - Find an employee's deductions received with Deductions Dtotal
and Employee EmployeeID. Display Tax deductions and Benefits deductions and Absences.*/

SELECT
        COM.Company_ID,
   EMP.Fname,
   EMP.Lname,
   PAY.tax_deduction_total,
   PAY.benefits_deduction_total,
   PAY.deductions_Total
FROM payroll.employee AS EMP
        LEFT JOIN payroll.company AS COM
   ON EMP.Company_ID = COM.Company_ID
   LEFT JOIN(
                SELECT
                        PAY.EmployeeID,
        PAY.tax_deduction_total,
        PAY.benefits_deduction_total,
        SUM(PAY.tax_deduction_total + PAY.benefits_deduction_total) AS deductions_Total
                FROM payroll.paystub AS PAY
      GROUP BY PAY.EmployeeID
   )AS PAY
   ON EMP.EmployeeID = PAY.EmployeeID
GROUP BY
        EMP.Fname,
   EMP.Lname
ORDER BY
        PAY.tax_deduction_total;

/*Functional Requirement 7 - Find all salaries that are greater than 50000. Order by company
name and department name*/

SELECT
        COM.Cname,
   DEPT.Dname,
        EMP.Fname,
   EMP.Lname,
   SAL.BaseSalary
```

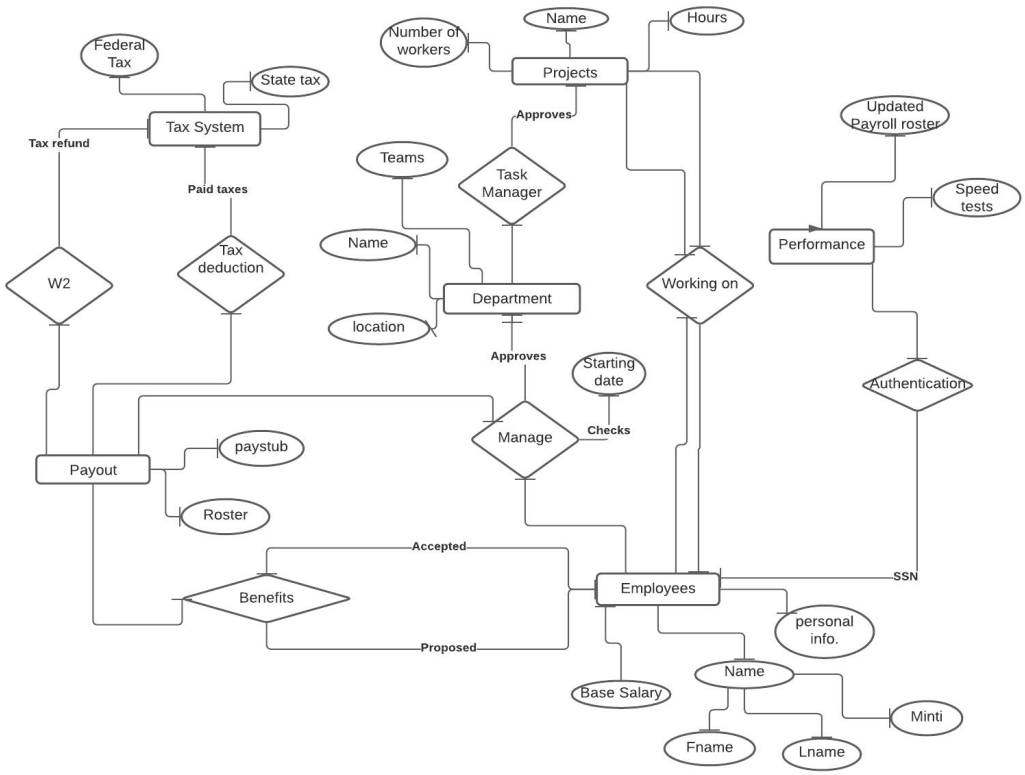
```
FROM payroll.employee AS EMP
        LEFT JOIN payroll.company AS COM
    ON EMP.Company_ID = COM.Company_ID
    LEFT JOIN payroll.department AS DEPT
    ON EMP.DID = DEPT.Department_ID
    LEFT JOIN payroll.salary AS SAL
    ON EMP.EmployeeID = SAL.EmpID
WHERE SAL.BaseSalary > 50000
ORDER BY
        COM.Cname,
        DEPT.Dname;

/*Functional Requirement 8 - Find all the employees that had a leave fromDate and toDate in
2020. Order by Company name*/
SELECT
        COM.Cname,
        EMP.Fname,
    EMP.Lname,
    LEA.fromDate,
    LEA.toDate
FROM payroll.employee AS EMP
        LEFT JOIN payroll.leaveP AS LEA
        ON EMP.EmployeeID = LEA.Em_ID
    LEFT JOIN payroll.company AS COM
    ON EMP.Company_ID = COM.Company_ID
WHERE YEAR(LEA.fromDate) = 2020 AND YEAR(LEA.toDate) = 2020
ORDER BY
        COM.Cname

/*Functional Requirement 9 - Find how much an employee's payout goes to taxes with Tax
StateTax, FederalTax and Employee EmployeeID. Order by Company name*/
SELECT
        COM.Cname,
    EMP.Fname,
    EMP.Lname,
    TAX.payments_Total,
    TAX.tax_deduction_total
FROM payroll.employee AS EMP
        LEFT JOIN (
                SELECT
```

```
                PAY.EmpID,
        PAY.payments_Total,
                (PAY.payments_Total * SUM(TAX.StateTax + TAX.FederalTax)) AS
tax_deduction_total
            FROM payroll.payments AS PAY
                LEFT JOIN payroll.tax as TAX
                ON PAY.EmpID = TAX.E_ID
            GROUP BY
                PAY.payments_Total,
                PAY.EmpID
        ) AS TAX
    ON EMP.EmployeeID = TAX.EmpID
    LEFT JOIN payroll.company AS COM
    ON EMP.Company_ID = COM.Company_ID
GROUP BY
        EMP.Fname,
    EMP.Lname
ORDER BY
        COM.Cname;

/* Functional Requirement 10 - Find which employees have greater than 3 absences and tardies.
Order by company name and department name.*/

SELECT
        COM.Cname,
    DEPT.Dname,
    EMP.Fname,
    EMP.Lname,
    ATT.Absences,
    ATT.Tardies
FROM payroll.employee AS EMP
        LEFT JOIN payroll.company AS COM
    ON EMP.Company_ID = COM.Company_ID
    LEFT JOIN payroll.department AS DEPT
    ON EMP.DID = DEPT.Department_ID
    LEFT JOIN payroll.attendance AS ATT
    ON EMP.EmployeeID = ATT.EmID
WHERE ATT.Absences > 3 AND ATT.Tardies > 3
ORDER BY
        COM.Cname,
```

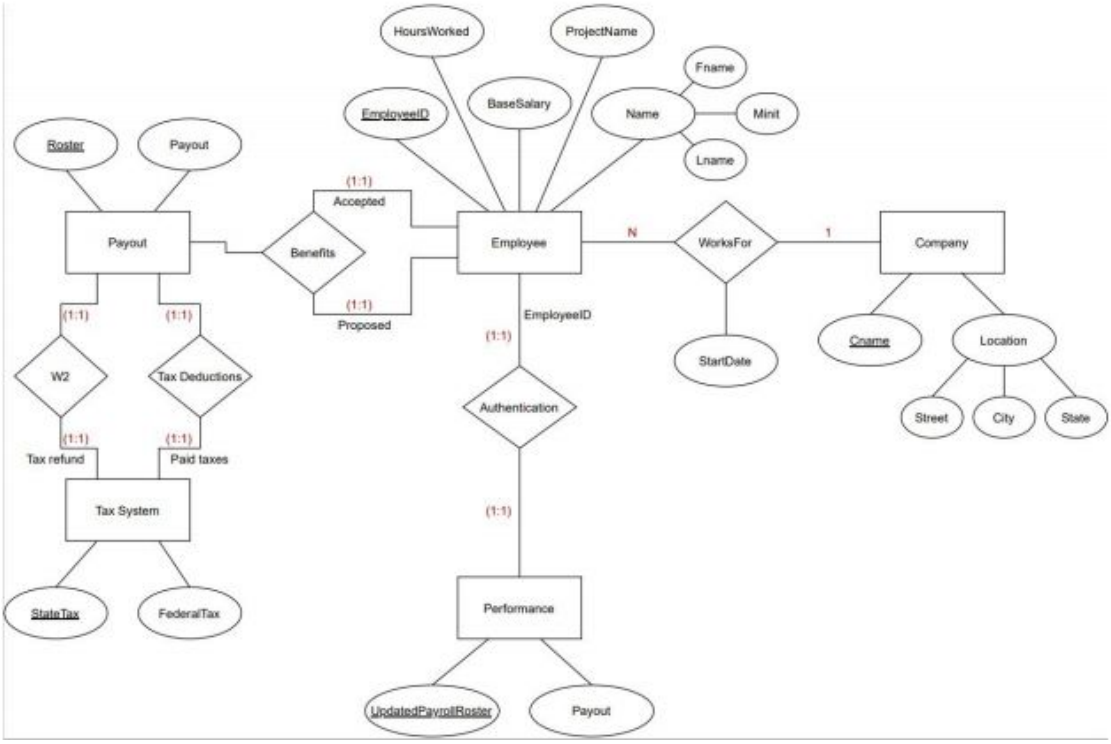
```
    DEPT.Dname;

/*Functional Requirement 11 - Find out how much an employee's payout goes to benefits with
BenefitsID, Benefits Retirement, Benefits Insurance and EmployeeID. Order by Company
name.*/

SELECT
        COM.Cname,
        EMP.Fname,
    EMP.Lname,
    PAY.payments_Total,
    BEN.benefits_deduction_total
FROM payroll.employee AS EMP
        LEFT JOIN payroll.company AS COM
    ON EMP.Company_ID = COM.Company_ID
    LEFT JOIN payroll.payments AS PAY
    ON EMP.EmployeeID = PAY.EmpID
        LEFT JOIN (
                SELECT
                        BEN.EmployID,
                        SUM(BEN.Retirement + BEN.Insurance) AS benefits_deduction_total
                FROM payroll.benefits AS BEN
                GROUP BY BEN.EmployID
    ) AS BEN
    ON EMP.EmployeeID = BEN.EmployID
GROUP BY
        EMP.Fname,
    EMP.Lname
ORDER BY
        COM.Cname;
```

Changelog

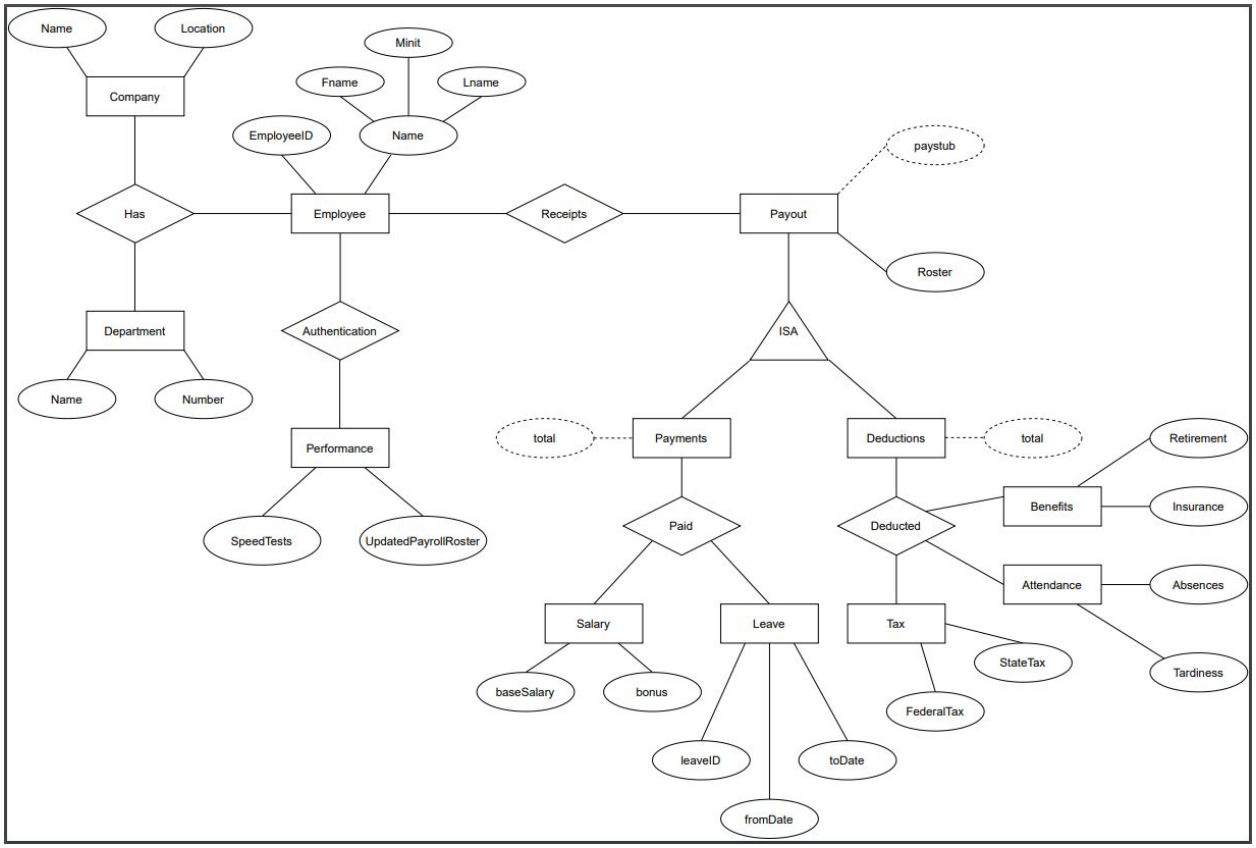
On 10/4/2020 - Project Design Report, Feedback: Our project was too similar to the example taught in class. Here is the original ER Model:



On 10/22/2020 - Our updated ER diagram does not meet the requirements. Must be at least 12 entities. Here is the ER Model:



On 10/28/2020 - Our mostly new ER Model was approved by both Miss Chuyan and Professor Li. Here is the ER Model:



On 11/12/2020 - Our relational model for the above diagram did not have the needed foreign keys.

Relational Model

Company: (Cname, Location)
Department: (Dname, Number, FK: Cname)
Employee: (EmployeeID, Fname, Minit, Lname, FK: Cname, FK: Dname)
Performance: (UpdatedPayrollRoster, SpeedTests, FK: EmployeeID)
Payout: (Roster, paystub)
Payments: (total, FK: Roster, FK: baseSalary, FK: leaveID)
Salary: (baseSalary, bonus)
Leave: (leaveID, fromDate, toDate)
Deductions: (total, FK: Roster, FK: StateTax, FK: Retirement, FK: Retirement)
Tax: (StateTax, FederalTax)
Benefits: (Retirement, Insurance)
Attendance: (Absences, Tardiness)

On 11/18/2020 - Updated ER Model and Relational Model once again due to Normalization challenges. Ended up being our final updates. They are listed in the beginning of Mapping and Appendix.

On 11/25/2020 - Encountered challenges with setting up tables in mySQL Workbench led to the Views. More details on Summary and Conclusion page.