**Norbert Shema**

**Database Design**

**Project Name: Payroll Management System *(PMS)***

Table of Contents

[Change Log 3](#_Toc83041319)

[Project Selection and Description 4](#_Toc83041320)

[Use Cases and Basic Data Needs 4](#_Toc83041321)

[Structural Database Business Rules 4](#_Toc83041322)

[Conceptual Entity Relationship Diagram (C-ERD) 5](#_Toc83041323)

[Full Database Physical ERD (P-ERD) 5](#_Toc83041324)

[SQL for Table Build 5](#_Toc83041325)

[SQL for Data Insert 6](#_Toc83041326)

[Reusable, Transaction Oriented Stored Procedure 6](#_Toc83041327)

[History Table & Trigger 6](#_Toc83041328)

[Business Query 7](#_Toc83041329)

[Summary and Reflection 7](#_Toc83041330)

# Change Log

*2024-09-16: Iteration 1*

1. *Created initial project document with inclusion of project description and use cases*

*2024-10-08: Iteration 2*

1. *Made some changes on why the project topic is interesting to me.*
2. *Updated the use case 3 and add new entities.*
3. *Completed iteration 2, Entity Identification and Structural Business Rules*

*2024-10-25: Iteration 3*

1. *Moved the summary and reflection to the end of the document and removed the attributes and the constraints form the entities.*
2. *Updated the Structural Business Rules for each two paired entities.*
3. *Complete iteration 4. Modified the project description and use cases as needed, modified the entities and structural business rules as needed, added one more use case, Created the conceptual entity relationship diagram.*

*2024-11-11-24: Iteration 4*

1. *Added a new entity for use case 1&2.*
2. *Updated and added two new business rules and updated the C-ERD*
3. *Completed the iteration 4 by adding the list of table fields and physical entity relationship diagram.*

*2024-12-03: Iteration 5*

1. *Updated the final Physical ERD*
2. *Updated figure 7 and changed the names to avoid attributes that has the same names as key words in SQL*
3. *Completed iteration 5 by creating the SQL scripts to build tables and constraints (PKs and FKs)*
4. *Created SQL scripts to insert a small amount of data into each table*
5. *Completed summary and reflection.*

*2024-12-14: Iteration 6*

1. *Updated the structural business rules, your conceptual ERD, and physical ERD.*
2. *Created a payroll history table and update both the create table and the insert table scripts*
3. *Created a store procedure and a trigger to insert data in the history table*
4. *Create a select business query*
5. *Updated and finalized the summary and reflection.*

# 

# Project Selection and Description*.*

**Overview**

The Payroll Management System database is intended to automate and simplify an organization's payroll administration procedures. With the help of this system, employee payments will be managed effectively and efficiently, guaranteeing accuracy in wage distributions, tax deductions, and regulatory compliance.

**Who will use the database**

The departments of Human Resources (HR) and Payroll are responsible for maintaining employee information, including wages, deductions, bonuses, and other details related to finances.

1. Accounting departments: To monitor costs associated with payroll taxes and salaries for the company.
2. Management Systems: For producing reports on payroll, taxes, and financial planning.
3. Employees and employers: Indirectly, by viewing their payroll history through self-service portals and pay stubs.

**What kind of data it will contain**

Important payroll-related data will be kept in the database, including:

1. Employee ID, name, position, department, pay level, financial account details, and tax status are all included in the employee data.
2. Payroll information includes net salary, tax withholdings, overtime, bonuses, deductions, and allowances.
3. Participation and Working Hours: Information directly related to payroll that comes from timesheets, attendance logs, and leave records.
4. Details on taxes and benefits: information regarding the rates of federal and state taxes, benefits (such as health insurance and retirement programs), and other expenditures.
5. Payroll Periods: Dates of payroll run-off, pay cycles (monthly, bi-weekly, etc.), and related reports.
6. Payroll history includes past invoices, modifications, and audit trails.

**How you envision it will be used, and most importantly**

1. Process Payroll: Determine employee wages automatically based on the number of hours worked, the salary level, bonuses, and deductions.
2. Create salary Slips: Give employees thorough explanations of their total salary, deductions, and net compensation.
3. Verify Adherence: Comply with tax laws by computing and deducting the relevant taxes and deductions.
4. Give a report: Provide reporting on payroll costs, tax obligations, and overall payroll performance for HR, management, and accounting.
5. Examine and Monitor Payroll Modifications: Keep track of previous payroll information and any alterations, such as adjustments or compensation changes.

**Why I Am Interested in It**

I'm interested in building this payroll management system because payroll is an essential yet necessary operation for all types of businesses. This issue interests me because, as a business major and software development major, it will allow me to work and interconnect in both fields, working with databases and on business-related projects. Additionally, if I can grasp both the business and IT aspects, this would be a great start toward my goal of owning my own business in the future.

# Use Cases and Basic Data Needs

**Use Case 1: Employee Payroll Calculation Summary**

An employee's compensation is determined by the payroll system considering their position, level, attendance, and deductions.

**Steps to take:**

1. HR keeps track of the worker's hours worked and attendance for the duration of the pay period.
2. The employee's hourly rate and salary grade are obtained from the database by the payroll department.
3. Bonuses, overtime pay, and other allowances are applied by the system.
4. From the employee's gross salary, the system computes insurance, tax, and other deductions.
5. Deductions from gross compensation are subtracted to determine the net salary.
6. An employee's pay slip is prepared, and the final payment is entered into the database.

Employees will have to sign up for the first time and include their information such as

A screenshot of a document

Description automatically generated

Figure 1

The above shows a sample of a self-created table with fields needed to generate the payroll in the database.

**Use Case 2: Employee Adding or Changing Personal Information.**

HR modifies a workers financial and personal data, including bank account and tax information.

Steps:

1. The employee provides updated personal or financial information (e.g., a new bank account).
2. HR gets access to the worker's system profile.
3. HR enters or updates the worker's tax information, bank account information, and other personal data.
4. For use in processing payroll in the future, the revised data is stored in the database.

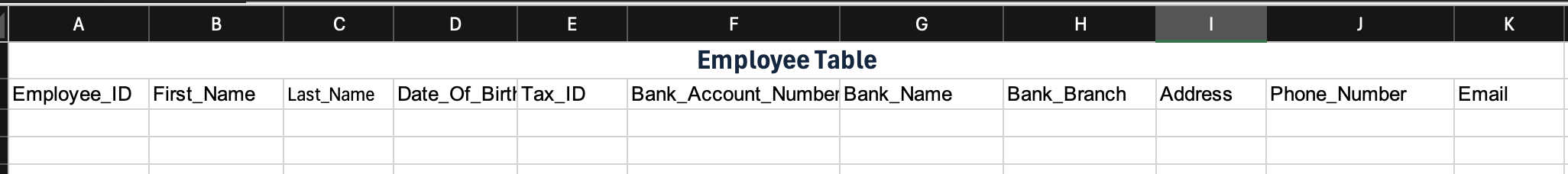


Figure 2

The figure 2 show a self-created table with significant information fields needed for the adding and updating the employee information.

**Use Case 3: Benefits and insurance management**

This use case presents a feature that is directly integrated with the payroll system to manage employee benefits, including health insurance.

**Steps:**

1. Enrollment in Benefits: During an open enrollment period, workers select from a variety of benefit plans (such as health insurance, retirement plans, transportation, and food allowances).
2. Assignment of Benefit Plans: HR adds the chosen benefits, along with any employee contributions and corporate charges, to the employee's profile.
3. Contributions to Benefits Each Month: Every payroll cycle, the system performs calculations.

The entities needed to perform this use case are shown in figure 3

A screenshot of a document

Description automatically generated

Figure 3

**Use Case 4: Employee Leave and Attendance Management**

This use case facilitates the tracking of leave requests, approvals, and staff attendance. By considering attendance records and leave type (such as paid or unpaid leave), the payroll system keeps this data to precisely determine compensation. This will help in compensation calculation of any kind of time and hours worked.

**Steps to take:**

1. Submission of a Leave Request: The employees will use the system to submit a leave request, indicating the type of leave (paid, unpaid, medical). They will also add any necessary information such as the start and end date for the requested leave.
2. Logging Attendance Records: Check-in and check-out timings are recorded in the system along with daily attendance. If applicable, overtime is noted and recorded separately from regular hours.

In figures 4 and 5 we can see the entities needed to perform the use case 4.

A screenshot of a phone

Description automatically generated

Figure 4

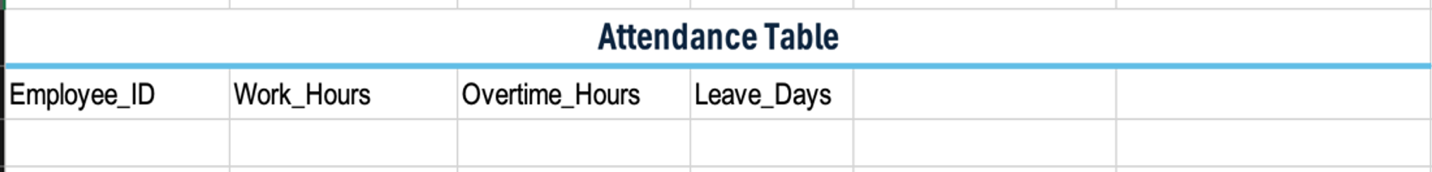


Figure 5

**Use case 5:** Tracking payroll change history for audit and future reference.

This use case ensures that all changes made to payroll records are documented for auditing purposes and future references. It maintains a comprehensive log of updates, allowing for transparency and compliance with organizational and regulatory requirements.

**Steps to Take:**

1. Recording Payroll Changes: All modifications to payroll, such as salary adjustments, tax updates, or bonuses, are recorded in the system. Each entry includes the old and new values, the reason for the change, and the date and time of the update.
2. Generating Change History Reports: Authorized users can generate reports summarizing payroll changes over a specific period. These reports include details like change type, timestamps, and the user responsible for the modifications, ensuring transparency and accountability.

# Payroll Management System Entity identification and Structural Database Business Rules

The first step will be the hiring an employee and getting all the information needed about the employee and setting up their accounts for tracking their hours work and all the payment details as well as their personal information. We will specifically record the employee identification, attendance information, payroll information, benefit plans and the employee benefits information. These vital fields that need to remain null include Employee ID, First and Last Name, Gross Pay, Net Pay, Tax Deductions, and Bank Account Number.

***Defining the entities:***

**For use case 1&2:** The following entities.

* Employee
* Attendance
* Payroll
* Pay period

**For use case 3:** These are the entities.

* Benefit plan
* Employee Benefits

**For use case 4**: These are the entities

* Leave request
* Employee
* Attendance

**For use case 5:** These are the entities

* Payroll
* Payroll history

***Relationships/ Structural business rules***

1. **Employee and Attendance:**

* An Employee may have many Attendance records.
* An Attendance record is for one Employee.

1. **Employee and Payroll:**

* An Employee can have more than one Payroll record.
* A Payroll record is for one Employee.

1. **Employee and Benefits:**

* An Employee may have many Benefits.
* A Benefit can be had by many Employees.

1. **Payroll and Attendance:**

* A Payroll record includes multiple/many Attendance records.
* An Attendance record is associated with one Payroll.

1. **Payroll and Pay Period:**

* A payroll is associated with one pay period
* A pay period may be associated many payrolls.

1. **Attendance and pay period:**

* An attendance is associated with one pay period.
* A pay period may be associated with may attendances.

1. **Leave Request and Employee:**

* An Employee may have multiple Leave Requests.
* A Leave Request is associated with only one Employee.

1. **Payroll and Payroll history**

* Each **Payroll History** record must reference exactly one **Payroll** record
* A payroll may have many payroll history entries in association with.

# Conceptual Entity Relationship Diagram (C-ERD)

A diagram of a company employee

Description automatically generated

Figure 6

The above figure 6 shows a C-ERD based on the list of entities and structural business rules

# Full Database Physical ERD (P-ERD)

After the entities, relationships, and rules formalized, a listing of attributes has been drafted, as

preparation for normalization. The listing below also includes the primary or foreign keys listed.

A table of documents with text

Description automatically generated with medium confidence

Figure 7

The figure 7 above shows a list of needed fields in each table for the related entities. The fields also include the primary key, any required foreign keys and a basic set of important additional fields.

**Physical ERD**

A diagram of a work flow

Description automatically generated

Figure 8

The figure 8 above shows the physical ERD, with attributes added

# SQL for Table Build*.*

*DROP TABLE Payroll\_History*

*DROP TABLE Leave\_Request;*

*DROP TABLE Employee\_Benefits\_XREF;*

*DROP TABLE Benefit\_Plan;*

*DROP TABLE Payroll;*

*DROP TABLE Pay\_Period;*

*DROP TABLE Attendance;*

*DROP TABLE Employee;*

*-- Create Employee Table*

*CREATE TABLE Employee (*

*Employee\_ID INT IDENTITY(1,1) PRIMARY KEY*

*,First\_Name VARCHAR(50)*

*,Last\_Name VARCHAR(50)*

*,Department VARCHAR(100)*

*,Hire\_Date DATE*

*);*

*-- Create Attendance Table*

*CREATE TABLE Attendance (*

*Attendance\_ID INT IDENTITY(1,1) PRIMARY KEY*

*,Employee\_ID INT REFERENCES Employee (Employee\_ID)*

*,TimeDate DATE*

*,CurrentStatus VARCHAR(20)*

*,Hours\_Worked DECIMAL(5,2)*

*);*

*-- Create Pay\_Period Table*

*CREATE TABLE Pay\_Period (*

*Pay\_Period\_ID INT IDENTITY(1,1) PRIMARY KEY*

*,StartDate DATE*

*,End\_Date DATE*

*,Period\_Name VARCHAR(50)*

*,CurrentStatus VARCHAR(20)*

*);*

*-- Create Payroll Table*

*CREATE TABLE Payroll (*

*Payroll\_ID INT IDENTITY(1,1) PRIMARY KEY*

*,Employee\_ID INT REFERENCES Employee (Employee\_ID)*

*,Pay\_Period\_ID INT REFERENCES Pay\_Period(Pay\_Period\_ID)*

*,Total\_Earnings DECIMAL(10,2)*

*,Total\_Deductions DECIMAL(10,2)*

*);*

*-- Create Benefit\_Plan Table*

*CREATE TABLE Benefit\_Plan (*

*Benefit\_ID INT IDENTITY(1,1) PRIMARY KEY*

*,Benefit\_Name VARCHAR(100)*

*,Coverage\_Type VARCHAR(50)*

*,Employer\_Covered DECIMAL(5,2)*

*,Employee\_Covered DECIMAL(5,2)*

*,Employee\_ID INT REFERENCES Employee (Employee\_ID)*

*,Enrollment\_Date DATE*

*,CurrentStatus VARCHAR(20)*

*);*

*-- Create Employee\_Benefits\_XREF*

*CREATE TABLE Employee\_Benefits\_XREF (*

*EB\_ID INT IDENTITY(1,1) PRIMARY KEY*

*,Employee\_ID INT REFERENCES Employee (Employee\_ID)*

*,Benefit\_ID INT REFERENCES Benefit\_Plan(Benefit\_ID)*

*,Enrollment\_Date DATE*

*,CurrentStatus VARCHAR(20)*

*);*

*-- Create Leave\_Request Table*

*CREATE TABLE Leave\_Request (*

*Leave\_Request\_ID INT IDENTITY(1,1) PRIMARY KEY*

*,Employee\_ID INT REFERENCES Employee (Employee\_ID)*

*,StartDate DATE*

*,End\_Date DATE*

*,CurrentStatus VARCHAR(20)*

*);*

*--create Payroll\_History Table*

*CREATE TABLE Payroll\_History (*

*History\_ID INT IDENTITY(1,1) PRIMARY KEY*

*,Payroll\_ID INT NOT NULL*

*,Employee\_ID INT NOT NULL*

*,Pay\_Period\_ID INT NOT NULL*

*,Old\_Earnings DECIMAL(10,2)*

*,New\_Earnings DECIMAL(10,2)*

*,Old\_Deductions DECIMAL(10,2)*

*,New\_Deductions DECIMAL(10,2)*

*,Change\_Date DATETIME DEFAULT GETDATE()*

*,Changed\_By VARCHAR(100) -- Username or system ID of the person updating*

*);*

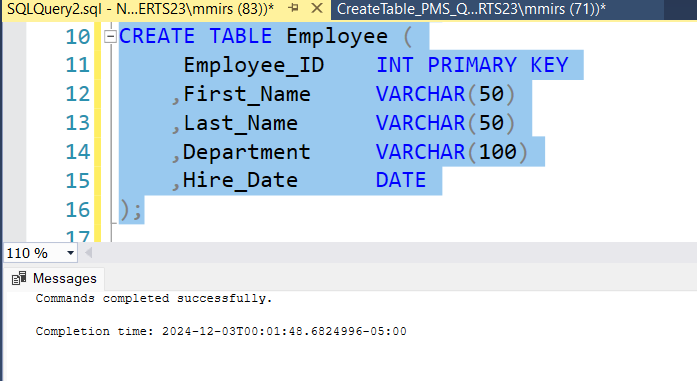


Figure 9

The figure 9 shows the creation of the employee table in the database.

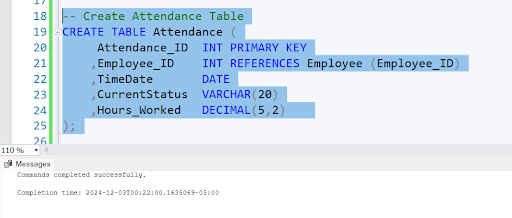


Figure 10

The figure 9 shows the creation of the attendance table in the database.

***A full script was provided above for the creation of each table need in the database***

# SQL for Data Insert

*-- Insert into Employee*

*INSERT INTO Employee (First\_Name, Last\_Name, Department, Hire\_Date)*

*VALUES*

*('Norbert', 'Shema', 'Engineering', '2020-01-15'),*

*('Laura', 'Ishimwe', 'HR', '2019-11-25');*

*-- Insert into Attendance*

*INSERT INTO Attendance (Employee\_ID, TimeDate, CurrentStatus, Hours\_Worked)*

*VALUES*

*(1, '2024-12-01', 'Present', 8.0),*

*(2, '2024-12-01', 'Sick Leave', 0.0);*

*-- Insert into Pay\_Period*

*INSERT INTO Pay\_Period (StartDate, End\_Date, Period\_Name, CurrentStatus)*

*VALUES*

*('2024-11-01', '2024-11-15', 'Pay Period 1', 'Closed'),*

*('2024-11-16', '2024-11-30', 'Pay Period 2', 'Closed');*

*-- Insert into Payroll*

*INSERT INTO Payroll (Employee\_ID, Pay\_Period\_ID, Total\_Earnings, Total\_Deductions)*

*VALUES*

*(1, 1, 3000.00, 250.00),*

*(2, 2, 2800.00, 200.00);*

*-- Insert into Benefit\_Plan*

*INSERT INTO Benefit\_Plan (Benefit\_Name, Coverage\_Type, Employer\_Covered, Employee\_Covered, Employee\_ID, Enrollment\_Date, CurrentStatus)*

*VALUES*

*('Health Insurance', 'Full Coverage', 500.00, 100.00, 1, '2024-01-01', 'Active'),*

*('Dental Insurance', 'Partial Coverage', 200.00, 50.00, 2, '2024-03-15', 'Active');*

*-- Insert into Employee\_Benefits\_XREF*

*INSERT INTO Employee\_Benefits\_XREF (Employee\_ID, Benefit\_ID, Enrollment\_Date, CurrentStatus)*

*VALUES*

*(1, 1, '2024-01-01', 'Active'),*

*(2, 2, '2024-03-15', 'Active');*

*-- Insert into Leave\_Request*

*INSERT INTO Leave\_Request (Employee\_ID, StartDate, End\_Date, CurrentStatus)*

*VALUES*

*(1, '2020-01-15', '2024-12-10', 'Approved'),*

*(2, '2019-11-25', '2024-12-05', 'Pending');*

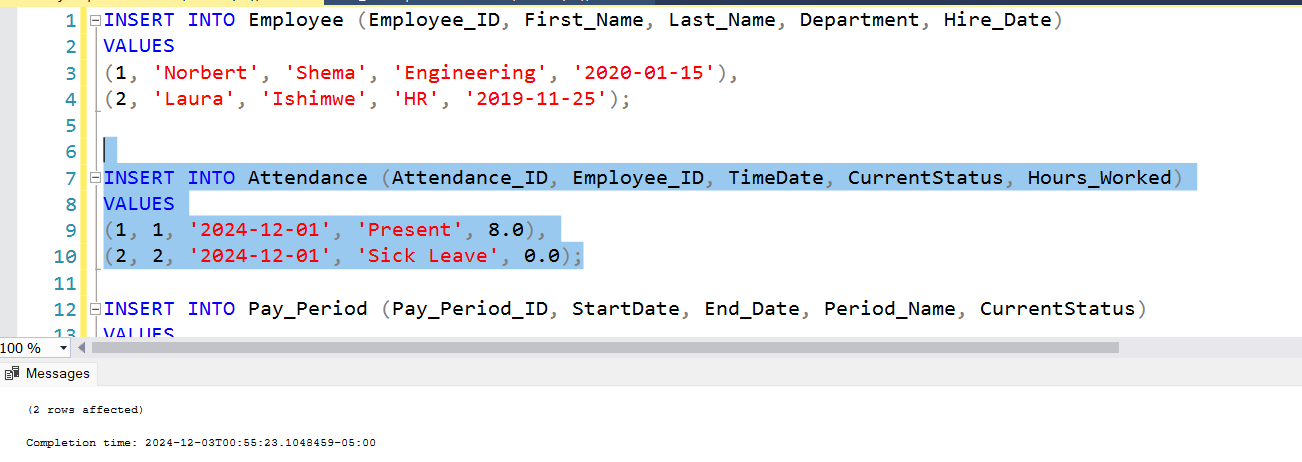


Figure 11

In the figure above, shows data being inserted in one of the table, attendance table, in the database.

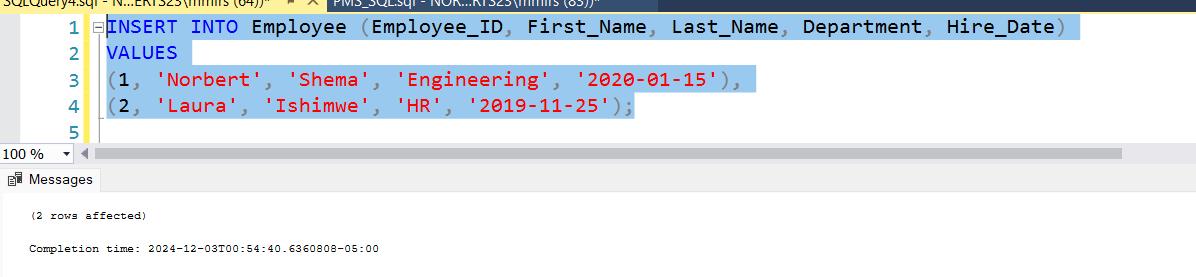


Figure 12

In the figure above, shows data being inserted in one of the table, employee table, in the database.

***A full insert script was provided above for the insertion of two sample data in each table created in this database for the purpose of functionality.***

# Reusable, Transaction Oriented Stored Procedure

A screenshot of a computer

Description automatically generated

Figure 13

Figure 13 shows a Created and reusable stored procedure that completes the steps of transactions necessary to add data to a database in the employee table.

A screenshot of a computer

Description automatically generated

Figure 14

In figure 14, we can see the execution of a reusable stored procedure for inserting data in the employee table.

**A full SQL script file that includes the SQL for the creation of the stored procedure was included with the final project documentation***.*

# History Table & Trigger

A screenshot of a computer

Description automatically generated

Figure 15

Figure 15 shows the Create a history table query to track changes to a value(s) within the payroll table.

A screenshot of a computer

Description automatically generated

Figure 16

Figure 16 shows a developed trigger that automates the inserting of data into the payroll history table.

A screenshot of a computer

Description automatically generated

Figure 17

Figure 17 shows a successful execution of the history table build and the inserting of data into the history table through a trigger.

**A full SQL script file that contains the creation of the trigger was included with the final project documentation.**

# Business Query

*Business Query question:*

Determine which employees have worked more than 20 hours in the current pay period and have active benefits.

*Why This Query is Important:*

Employee Productivity: Identifies employees who are working a significant number of hours.

Active Benefits: Ensures employees with active benefits are working enough to qualify for those benefits.

A screenshot of a computer

Description automatically generated

Figure 18

The figure 18 shows the business query used to provide the necessary data for the DB being executed.

Summary and Reflection

***Iteration 1***

The assignment was to develop and design a database for a payroll management system (PMS) to automate and streamline payroll procedures. This included specifying the types of data the database would hold, identifying the users, and explaining how the system would be use in addition to establishing the goal of the system. I was able to identify some important cases, including payroll computation for employees, personal information updates, payroll report generation, and tax filing. To guarantee that the database would be constructed to satisfy all functional requirements, significant database fields required to support these activities were also identified.

My ability to critically analyze the elements and procedures necessary for a productive payroll administration system has improved because of this project. Understanding how various departments—HR, payroll, and accounting—would interact with the system was one of the work's major achievements. This assisted me in determining the database fields and configuration required for smooth departmental integration and cooperation.

The security of sensitive data, including employee bank account information, salary information, and tax information, is one area of concern. It will be essential to implement sufficient security measures like access control and encryption, and I'd seek advice on the best ways to protect personal information.

I am excited to know how the passwords and how in case of an issue it can be solve if someone was paid the wrong amount and how to keep track that we have trusted data and UpToDate.

***Iteration 2***

The goal of this iteration 2 was to define the Payroll Management System (PMS) database's structure. Throughout this iteration I got an opportunity to think about different entities that are included in this project such as Employee, Attendance, Payroll, Benefit Plan, Employee Benefits and their attributes. I was also able to check and tell the relationships between tables. I was able to make constraint to establish the relationships in the data and give the data trust and minimizing the errors in entering and changing the data.

Considering Use Cases for Payroll Processing, through this iteration I was able to make sure the structure enables payroll computations, benefits administration, employee information changes, and attendance monitoring, through use cases. Some issues I face while in this iteration was the entities choosing and the attributes because I could argue that some attributes could also be entities and some business rules can be made for them. I was however able to solve it by just picking the entities needed for this PMS project and assigned them the corresponding attributes.

However, I still have concerns on the security of the data and how the access control will work to have data protection and privacy and referential integrity. As this project still have my very interested, I am looking forward to seeing how the database will function.

***Iteration 3***

After the completion of the iteration 3, I was able to visualize the diagram for the declared entities and the made structural business rules. I can argue that I can make more business rules, and some could be altered to mean something different but for the sake of the understand and visualizing the relationships I have chosen to revolve around employees because it’s the purpose of making a payroll system. I remain with questions like does all entities have to have a relationship with others or can we have two entities that are related but can be much of a relation with other entities? Overall, this iteration was a success I look forward to testing it out in further iterations.

***Iteration 4***

Throughout this iteration I was able to go further and visualize the physical diagram of the database and filled out different entities required for the set entities and relationships. I was able to incorporate specific data types, lengths, and cross-referencing to effectively manage payroll, attendance, benefits, and leave requests. For instance, an efficient data integrity and referential accuracy across all functional areas in the many to many relationships between the employees and the benefits. I remain with some concerns about how this design might be improved even more to accommodate situations in which workers have distinct pay periods or benefit plans tailored to their positions or levels of seniority? I remain eager to explorer more about the design and the queries for its functional requirement.

***Iteration 5***

After the completion of iteration 5, I am now in a more confident level with this project. I can now visual the database by creating all the needed table, setting the constraints and coming up with the data sample to be inserted in each table. I was also able to successfully run the scripts.

The completed files *(PMS\_Create Script and PMS\_Insert script)* for creating and inserting the data in the database was supplied as a file with the project but can also be viewed above in the document for easy referencing.

**Final Iteration 6**

Throughout the final iteration, I was able to be an explorer and successfully create a stored procedure for one of the database's tables. I developed a trigger and executed it for one of the database tables in case of data changes. A business query was provided and executed to demonstrate what the database could return. From prior iterations, I updated the insert and create queries to include the identity on the primary keys. In addition, a new table in the database necessitated an update to the structural business rules, conceptual ERD, and physical ERD.

What made this project successful was that I was able to start building the PMS database from scratch and be able to visually view the structural layout of the database without worrying about the SQL code. I was able to finish all the iterations, from establishing business rules to completing physical diagrams, to creating tables and moving on to procedural SQL. However, I believe there are numerous options and paths we may pursue in this database, and much more can be investigated. I am confident in the final look of the project.

*All the scripts mentioned above that was used in this project are provided in a zip file (PMS\_Scripts) that is attached to this submission.*