Milestone 2 “Description”

Deadline for Milestone 2 is on 24/11/2025 at 11:59

1. Important Guidelines:
   - **The Milestone description is subject to minor changes that would be announced on CMS. Do check the CMS periodically.**
   - All alpha-numeric attributes should be defined with type varchar(50).
   - All IDs in the project are identity(1,1).
   - All numeric values should be defined with type int or decimal.
   - Phone numbers should be defined as char(11).
   - Semester should be defined as char(3) such as W25 or S24.
   - You must follow the exact column order, data types, and column names as outlined in the provided schema. Failing to follow this guideline will result in a grade loss.
   - You must follow the names of the required functions, stored procedures, and views. **Not matching the names will result in a grade loss.**
   - You must follow the order of the needed columns in case a table is required to be returned from a function or fetched from a view.

- You must follow the order and data types of the parameters required for a stored procedure or function with a specific order and certain data type. Failing to follow this guideline will result in grade loss of that function/stored procedure/view.
- You should do the appropriate join if you are required to implement a function/view that fetches data from different tables in your database.
- You should insert the data in the appropriate tables in your database if the required stored procedure inserts data in multiple tables in your database.
- Always think **about all consequences** and required actions to be done in every requirement.
- You may use helper functions/procedures if needed.

1.1 Schema Constraints:

- The rating in performance should be from 1 to 5.
- The status of any document can either be valid or expired.
- The status of any leave can either be approved, rejected or pending with the default value set to **pending**.
- The type of deductions can either be unpaid, missing_hours, missing_days.

- The status of any deduction can either be pending or finalized, with the default value set to **pending**.
- The status of attendance can either be absent or attended, with the default value set to **Absent**.
- The type of contract for an employee can either be full_time or part_time.
- The employment_status can either be active, onleave, notice_period or resigned.
- The type of medical leave can either be sick or maternity.

1.2 General Schema Guidelines:

- An employee’s salary is calculated as salary = base_salary (based on role) + (%year_of_experience/100)× years_of_experience × base_salary
- The role HR Representative has a name format representing the department they are assigned to such as HR_Representative_MET or HR_Representative_IET.
- If an employee has more than one role, the overtime factor and base salary of the higher rank will be used.
- Departments can either be (MET,IET….etc), HR department, or Medical department.
- The total required hours per day for full_time employees is 8 hours.
- In table Employee_Replace_Employee Emp_1 is replaced by Emp_2.

1.3 Payroll and Attendance Guidelines:

- Finalizing the deductions, payrolls and attendance of the current month occurs when requested.
- Deductions are finalized when they are reflected in the payroll.
- Bonuses are added to the payroll in case of overtime.
- Rate per hour = (employee_salary/22 days)/8 hours
- Overtime amount = rate per hour × ([overtime factor (based on role) × extra hours in attendance]/100).

1.4 Leaves Guidelines:

- Employees who are part-time are not eligible for annual, unpaid, and maternity leaves.
- For leaves requiring Dean’s approval, in case the Dean is onleave, the vice-Dean approves/rejects instead.
- Dean and vice-Dean cannot be onleave at the same time.
- When the Dean or Vice Dean submits an annual/unpaid leave request, it must be approved/rejected by the President and HR representative.

- When an HR employee submits an unpaid leave request, it must be approved/rejected by the President and HR Manager.
- The last person to approve/reject leaves is the HR employee and accordingly adjusts the final status of the leave.
- If any of the employees in the approval hierarchy reject the leave, the final status of the leave should be rejected.
- Compensation leaves are approved by HR employees if the employee applying for the leave spent at least 8 hours during his/her day off. The compensation has to be requested within the same month and should have a valid reason.
- Accidental leaves duration is only 1 day per leave.
- An employee can only have **one** approved unpaid leave per year, with a maximum duration of 30 days.
- You are not required to handle leave requests submitted by the President, Vice President, or HR Manager.

- **For leaves approval hierarchy and conditions refer to the below table and milestone 1 description.**

Role	**Belongs to Department**	**Rank**
President	-	1
Vice President	-	2
Dean	MET, IET, …. etc	3
HR Manager	HR Department	3
Vice Dean	MET, IET, …. etc	4
HR Representative	HR Department	4
Lecturer	MET, IET, …. etc	5
Teaching Assistant (TA)	MET, IET, …. etc	6
Medical Doctor	Medical Department	-

- **Any compilation/syntax error in any of the requirements will result in a grade loss.**

2. Requirements:

You are required to implement the following:

2.1 Basic Structure of the Database:

a) Write an SQL query to create database called "University_HR_ManagementSystem_Team_No"

b) Put the queries that create all the tables of your database with their definition inside this procedure.

   i) Type: stored procedure
   ii) Name: createAllTables
   iii) Input:Nothing
   iv) Output:Nothing

c) Drop all tables that your database has inside this procedure.

   i) Type: stored procedure
   ii) Name: dropAllTables
   iii) Input:Nothing
   iv) Output:Nothing

d) Drop all implemented stored procedures (except this one), functions and views that you implemented in this milestone.

   i) Type: stored procedure
   ii) Name: dropAllProceduresFunctionsViews
   iii) Input:Nothing
   iv) Output:Nothing

e) Clear all records in all tables existing in your database.

   i) Type: stored procedure
   ii) Name: clearAllTables
   iii) Input:Nothing
   iv) Output:Nothing

2.2 Basic Data Retrieval:

a) Fetch details for all employees.
   - i) Type: view
   - ii) Name: allEmployeeProfiles
   - iii) Input:Nothing
   - iv) Output: Table (includes employee_ID, first_name, last_name, gender, email, address, years_of_experience, official_day_off,type_of_contract,employment_status, annual_balance, accidental_balance)

b) Fetch the number of employees per department.
   - i) Type: view
   - ii) Name: NoEmployeeDept
   - iii) Input:Nothing
   - iv) Output: Table

c) Fetch details for the performance of all employees in all Winter semesters.
   - i) Type: view
   - ii) Name: allPerformance
   - iii) Input:Nothing
   - iv) Output: Table

d) Fetch details of all rejected medical leaves.
   - i) Type: view
   - ii) Name: allRejectedMedicals
   - iii) Input:Nothing
   - iv) Output: Table

e) Fetch the attendance records for all employees for yesterday.

   i) Type: view
   ii) Name: allEmployeeAttendance
   iii) Input:Nothing
   iv) Output: Table

2.3 As an admin I should be able to:

a) Update the status of all expired documents depending on the expiry date.

   i) Name: Update_Status_Doc
   ii) Type: Stored Procedure
   iii) Input: Nothing
   iv) Output : Nothing

b) Remove deductions of resigned employees based on their employment_status.

   i) Name: Remove_Deductions
   ii) Type: Stored Procedure
   iii) Input: Nothing
   iv) Output : Nothing

c) Update the employee's employment_status daily based on whether the employee is on leave or active. **You may use 2.5.f.**

   i) Name: Update_Employment_Status
   ii) Type: Stored Procedure
   iii) Input: Employee_ID int
   iv) Output : Nothing

d) Create a lookup table named Holiday that has a unique automated **holiday_id** type int, **name** type varchar(50), **from_date** type date and **to_date** type date . A lookup table in SQL is a small, reference table used to store a fixed set of values that other tables can reference.

i) Name: Create_Holiday

ii) Type: Stored Procedure

iii) Input: Nothing

iv) Output : Nothing

e) Add a new official holiday.

i) Name: Add_Holiday

ii) Type: Stored Procedure

iii) Input: holiday_name varchar(50), from_date, to_date

iv) Output: Nothing

f) Initiate attendance records for the current day for all employees.

i) Name: Intitiate_Attendance

ii) Type: Stored Procedure

iii) Input: Nothing

iv) Output : Nothing

g) Update the attendance record for the current day for a certain employee along with the status.

i) Name: Update_Attendance

ii) Type: Stored Procedure

iii) Input: Employee_id int, check-in time, check-out time.

iv) Output : Nothing

h) Remove attendance records for all employees during official holidays.

  i) Name: Remove_Holiday
  ii) Type: Stored Procedure
  iii) Input: Nothing
  iv) Output : Nothing

i) Remove unattended dayoff for a certain employee from the attendance records of the current month.

  i) Name: Remove_DayOff
  ii) Type: Stored Procedure
  iii) Input: Employee_id int
  iv) Output: Nothing

j) Remove approved leaves for a certain employee from the attendance records.

  i) Name: Remove_Approved_Leaves
  ii) Type: Stored Procedure
  iii) Input: Employee_id int
  iv) Output: Nothing

k) Replace another employee.

  i) Name: Replace_employee
  ii) Type: Stored Procedure
  iii) Input: Emp1_ID int, Emp2_ID int, from_date date, to_date date
  iv) Output: Nothing

2.4 As an HR Employee, I should be able to:

a) login using my Id and password.

i) Name: HRLoginValidation

ii) Type: Function

iii) Input: employee_ID int, password varchar(50)

iv) Output: Success bit

b) Approve/reject annual/accidental leaves based on the employee’s balance.

i) Name: HR_approval_an_acc

ii) Type: Stored Procedure

iii) Input: request_ID int, HR_ID int

iv) Output: Nothing

c) Approve/reject unpaid leaves.

i) Name: HR_approval_unpaid

ii) Type: Stored Procedure

iii) Input: request_ID int, HR_ID int

iv) Output: Nothing

d) Approve/reject compensation leaves.

i) Name: HR_approval_comp

ii) Type: Stored Procedure

iii) Input: request_ID int, HR_ID int

iv) Output: Nothing

e) Add deduction due to missing hours. While adding the deduction, reference the attendance_id of the first record of the month that has less than 8 hours.

i) Name: Deduction_hours

ii) Type: Stored Procedure

iii) Input: employee_ID int

iv) Output: Nothing

f) Add deduction due to missing days.

i) Name: Deduction_days

ii) Type: Stored Procedure

iii) Input: employee_ID int

iv) Output: Nothing

g) Add deduction due to unpaid leave. If an unpaid leave spans two months, two separate deductions should be created, one for each month, based on the approved unpaid leave days that occurred within that month.The deduction rate per day will be equivalent to the rate applied for a missing day.

i) Name: Deduction_unpaid

ii) Type: Stored Procedure

iii) Input: employee_ID int

iv) Output: Nothing

h) Calculate bonus amount based on overtime (**will be used in 2.4.i**).

i) Name: Bonus_amount

ii) Type: Function

iii) Input: employee_ID int

iv) Output: Bonus value

i) Generate the monthly payroll.
  - i) Name: Add_Payroll
  - ii) Type: Stored Procedure
  - iii) Input: employee_ID int, from date, to date
  - iv) Output: Nothing

2.5 As an Employee I should be able to:

a) login using my Id and password.
  - i) Name: EmployeeLoginValidation
  - ii) Type: Function
  - iii) Input: employee_ID int, password varchar(50)
  - iv) Output: Success bit

b) Retrieve my performance for a certain semester.
  - i) Name: MyPerformance
  - ii) Type: Table Valued Function
  - iii) Input: employee_ID int, semester char(3)
  - iv) Output: Table

c) Retrieve attendance records for the current month, excluding my unattended official_day_off.
  - i) Name: MyAttendance
  - ii) Type: Table Valued Function
  - iii) Input: employee_ID int
  - iv) Output: Table

d) Retrieve last month’s payroll details.
  - i) Name: Last_month_payroll
  - ii) Type: Table Valued Function
  - iii) Input: employee_ID int
  - iv) Output: Table

e) Fetch all deductions caused by attendance issues in a certain period.

i) Name: Deductions_Attendance

ii) Type: Table Valued Function

iii) Input: employee_ID int, month int

iv) Output: Table

f) Verify whether the employee will be on leave during the specified period without referencing the attendance table. If the employee has a leave request with a **pending** status, treat it as **approved** for verification purposes, without modifying its actual status.

i) Name: Is_On_Leave

ii) Type: Function

iii) Input: employee_ID int, from date, to date

iv) Output: Success bit

g) Apply for an annual leave. Populate the approval table accordingly with the corresponding employees for the leaves' approval based on the hierarchy.

i) Name: Submit_annual

ii) Type: Stored Procedure

iii) Input: employee_ID int, replacement_emp int, start_date date, end_date date

iv) Output: Nothing

h) Retrieve the status of all my submitted annual and accidental leaves during the current month.

i) Name: Status_leaves

ii) Type: Table Valued Function

iii) Input: employee_ID int

iv) Output: Table includes (request_ID, date_of_request, status)

i) As a Dean/Vice-dean/President I can approve/reject annual leaves. In case the person of replacement isn’t on leave and works in the same department, the leave gets approved.

  i) Name: Upperboard_approve_annual

  ii) Type: Stored Procedure

  iii) Input: request_ID int, Upperboard_ID int, replacement_ID int

  iv) Output: Nothing

j) Apply for an accidental leave. Populate the approval table accordingly with the corresponding employees for the leaves’ approval based on the hierarchy.

  i) Name: Submit_accidental

  ii) Type: Stored Procedure

  iii) Input: employee_ID int, start_date date, end_date date

  iv) Output: Nothing

k) Apply for a medical leave. Populate the approval table accordingly with the corresponding employees for the leaves’ approval based on the hierarchy.

  i) Name: Submit_medical

  ii) Type: Stored Procedure

  iii) Input: employee_ID int, start_date date, end_date date, type varchar(50), insurance_status bit, disability_details varchar(50), document_description varchar(50), file_name varchar(50)

  iv) Output: Nothing

l) Apply for unpaid leave. Populate the approval table accordingly with the corresponding employees for the leaves’ approval based on the hierarchy.

  i) Name: Submit_unpaid

  ii) Type: Stored Procedure

  iii) Input: employee_ID int, start_date date, end_date date, document_description varchar(50), file_name varchar(50)

  iv) Output: Nothing

m) As a Dean/Vice-dean/President I can approve/reject unpaid leaves. In case a memo document is submitted with a valid reason, the leave gets approved. Deductions are not reflected in this query.

i) Name: Upperboard_approve_unpaids

ii) Type: Stored Procedure

iii) Input: request_ID int, Upperboard_ID int

iv) Output: Nothing

n) Apply for a compensation leave. Populate the approval table accordingly with the corresponding employees for the leaves' approval based on the hierarchy.

i) Name: Submit_compensation

ii) Type: Stored Procedure

iii) Input: employee_ID int, compensation_date date, reason varchar(50), date_of_original_workday date, replacement_emp int

iv) Output: Nothing

o) As a Dean, I can evaluate employees within the same department.

i) Name: Dean_andHR_Evaluation

ii) Type: Stored Procedure

iii) Input: employee_ID int, rating int, comment varchar(50), semester char(3)

iv) Output: Nothing