TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CNTT & TT

BÁO CÁO MÔN THỰC HÀNH CƠ SỞ DỮ LIỆU

ĐỀ TÀI
XÂY DỰNG CƠ SỞ DỮ LIỆU QUẢN LÝ NGUỒN NHÂN LỰC VÀ TRẢ LƯƠNG

Giảng viên hướng dẫn: TS. Đỗ Bá Lâm

Sinh viên thực hiện
Lê Đình Tuyên 20194715
Nguyễn Trọng Tuệ 20194710
Nguyễn Anh Tuấn 20184220
Nguyễn Hữu Tuấn 20184221

MỤC LỤC……..……1

PHẦN I: XÂY DỰNG BÀI TOÁN

1. ***Xây dựng bài toán***

1.1. *Bài toán*

Vấn đề tính lương trong nhân sự là một vấn đề nhạy cảm, cần sự kín kẽ, bảo mật cũng như chuẩn xác. Thật dễ dàng nếu công ty của bạn có quy mô vừa và nhỏ, tuy nhiên đối với những doanh nghiệp có quy mô nhân sự lên đến hàng trăm, hàng nghìn nhân viên thì rủi ro sai số rất dễ xảy ra, và nghiêm trọng hơn nữa là lộ thông tin lương nhân viên, gây ra những mâu thuẫn không đáng có.

Từ bài toán được nêu trên, ý tưởng xây dựng CSDL quản lý nguồn nhân lực và trả lương ra đời.

1.2. *Mục đích*

Hệ CSDL quản lý nguồn nhân lực và trả lương cung cấp một thiết kế hệ thống thông tin quản lý nguồn nhân lực và trả lương theo tháng một cách thuận tiện, công ty có thể sử dụng để giảm thiểu rủi ro có thể xảy ra trong quá trình trả lương, tổng kết lương trong tháng và làm cho quá trình này hiệu quả, nhanh chóng, thuận tiện, chính xác.

1.3. *Đối tượng sử dụng CSDL*

CSDL được xây dựng phục vụ 2 nhóm đối tượng chính: nhóm đối tượng người quản lý CSDL (lãnh đạo, quản lý) và nhóm đối tượng người dùng CSDL là nhân viên công ty.

Đối với nhóm đối tượng người quản lý CSDL sẽ có quyền chỉnh sửa, cập nhật, truy vấn thông tin lương thưởng, khoản cắt giảm, thông tin cũng như trạng thái của nhân viên cùng các thông tin liên quan hàng tháng trong hệ quản trị CSDL. Người quản lý CSDL sẽ sử dụng tài khoản với đầy đủ quyền thao tác trên hệ CSDL được xây dựng.

Đối với nhóm đối tượng người dùng bao gồm nhân viên sẽ được cấp tài khoản với tên đăng nhập và mật khẩu riêng để truy cập vào hệ CSDL với quyền truy vấn các thông tin liên quan đến lương thưởng, các phụ khoản khấu trừ phát sinh, thông tin và trạng thái của nhân viên cùng các thông tin có liên quan khác

đến vấn đề quản lý lương của mình hàng tháng. Nhóm đối tượng này không có quyền thực hiện các thao tác chỉnh sửa, cập nhật đối với dữ liệu. Các khúc mắc liên quan đến nhân lực và lương cần được đề đạt lên lãnh đạo cấp trên để xử lý, sau đó được phê duyệt và chuyển tiếp cho người quản lý hệ CSDL chỉnh sửa và cập nhật.

2. ***Yêu cầu chức năng đối với hệ thống CSDL***

2.1. *Quản lý nguồn nhân lực của công ty*

CSDL chứa các thông tin cơ bản của toàn bộ nhân lực trong công ty bao gồm tên, ngày sinh, địa chỉ, thông tin liên lạc, chức vụ, trạng thái làm việc, phòng ban làm việc của nhân viên. Người dùng CSDL bao gồm nhân viên công ty có thể kiểm tra được thông tin cá nhân của nhân viên để phục vụ vấn đề quản lý lương thưởng cũng như các vấn đề phát sinh như việc chỉnh sửa thông tin cá nhân trong trường hợp có sai sót xảy ra, cũng như quản lý những thay đổi, chỉnh sửa đã được thực hiện. Việc nắm rõ nguồn nhân lực hiện có của công ty thông qua sự trợ giúp của hệ CSDL giúp công ty có thể đưa ra những định hướng, kế hoạch làm việc, tuyển nhân lực, cắt giảm nhân lực, điều chỉnh nhân lực,… một cách hợp lý và tối ưu.

2.2. *Quản lý lương của nhân viên trong công ty*

Việc quản lý lương thưởng của nhân viên trong công ty trở nên dễ dàng, nhanh chóng và chính xác hơn với sự trợ giúp của hệ thống CSDL quản lý nguồn nhân lực và trả lương đã được xây dựng. Ngoài mức lương cơ bản của nhân viên, các khoản tiền được tăng thêm cho từng vị trí nhân viên như làm thêm giờ, khen thưởng, trợ cấp; cũng như các khoản khấu trừ, cắt giảm do nghỉ phép, vi phạm, vấn đề ngân sách công ty, phí đầu tư, nâng cấp, sử dụng cơ sở vật chất, phí phúc lợi xã hội, ... đều được lưu trữ vào trong hệ CSDL theo từng tháng. Từ đây, việc tính toán lương thưởng nhân viên theo cá nhân, một nhóm, một phòng ban hoặc thậm chí là toàn bộ nhân viên công ty được thực hiện một cách chính xác và thuận tiện, từ đó đưa ra các chiến lược, kế hoạch quản lý quỹ lương cũng như nguồn nhân lực hợp lý, giúp công ty phát triển.

PHẦN II: THIẾT KẾ CƠ SỞ DỮ LIỆU

1. ***Xây dựng dữ liệu đầu vào CSDL***

Với bài toán quản lý nguồn nhân lực và trả lương được xây dựng, cùng các yêu cầu và mục đích được đặt ra, CSDL yêu cầu các dữ liệu đầu vào với thuộc tính, định dạng như sau:

Thông tin nhân viên gồm có:

- Tên nhân viên: tối đa 50 ký tự tiếng Việt, có dấu.
- ID nhân viên: số nguyên, tối đa 11 ký tự.
- ID chức vụ: số nguyên, tối đa 11 ký tự.
- Tên chức vụ: tối đa 50 ký tự tiếng Việt, có dấu.
- ID phòng ban: số nguyên, tối đa 3 ký tự.
- Tên phòng ban: tối đa 50 ký tự tiếng Việt, có dấu.
- Tuổi: 18 – 50 tuổi.
- Giới tính: Nam/ Nữ..
- Địa chỉ: tối đa 100 ký tự tiếng Việt, có dấu.
- Số điện thoại: tối đa 20 ký tự tiếng Việt, có dấu
- Email: tối đa 40 ký tự
- Quốc tịch: tối đa 20 ký tự tiếng Việt, có dấu
- Ngày vào/ ngày rời công ty: định dạng ngày tháng năm yyyy-mm-dd.
- Mức lương cơ sở: DECIMAL(10,2), tất cả các nhân viên trong công ty đều có chung mức lương cơ sở 3000000 VND.
- Hệ số lương: DECIMAL(4,2) (hệ số lương theo cá nhân/ chức vụ).
- Trạng thái nhân viên: đánh ID theo số nguyên, tên trạng thái là ký tự tiếng Việt tối đa 50 ký tự.

Thông tin lương theo tháng của nhân viên gồm có:

- Lương cơ bản (lương cứng) một tháng: DECIMAL(10,2) (VND), được tính bằng (mức lương cơ sở * hệ số lương) .

- Các khoản tăng trong tháng (làm thêm giờ, khen thưởng, trợ cấp), các khoản giảm trong tháng (khấu trừ, nghỉ phép): DECIMAL(10,2) (VND). Trong đó:
    - Làm thêm giờ (overtime): (mức trả thêm theo giờ * số giờ làm thêm).
    - Khen thưởng (incentive): tùy theo cá nhân, năng suất làm việc, khối lượng công việc hoàn thành, các khen thưởng khác, …
    - Trợ cấp (allowance): mức trợ cấp theo cá nhân, chức vụ.
    - Khấu trừ (deduction): khấu trừ do vi phạm, do cắt giảm ngân sách theo tình hình tài chính công ty, do các khoản phí phúc lợi xã hội, …
    - Nghỉ phép: trừ theo hệ số số ngày không nghỉ phép của cá nhân trên tổng số ngày công ty hoạt động trong tháng.
- Số ngày nghỉ, giờ làm thêm trong tháng: số nguyên.
- Ngày làm thêm/ ngày nghỉ: ngày định dạng yyyy-mm-dd.
- Tổng lương trong tháng: DECIMAL(10,2) (VND), tính bằng lương cơ bản, cộng các khoản tăng, trừ đi khoản giảm, ta được lương trong tháng đó của nhân viên

2. ***Các thao tác, chức năng chính đối với CSDL***

2.1. *Đối với người quản trị CSDL*

- Truy vấn, hiệu chỉnh, sửa đổi, cập nhật thông tin cá nhân (họ tên, tuổi, số điện thoại, trạng thái làm việc, …), các thông tin này là các dữ liệu được đưa vào CSDL với định dạng đã được nêu ở mục 1 của phần II bài báo cáo.
- Truy vấn, hiệu chỉnh, sửa đổi, cập nhật thông tin lương thưởng (lương cơ bản, tăng thưởng , khấu trừ) cùng các thông tin có liên quan như ngày nghỉ, số giờ làm thêm, … của bất cứ cá nhân nào trong phạm vi công ty. Các thông tin này là các dữ liệu được đưa vào CSDL với định dạng đã được nêu ở mục 1 của phần II bài báo cáo.
- Tính toán tổng lương của từng nhân viên, một nhóm nhân viên, một phòng ban hoặc toàn bộ nhân viên công ty trong tháng.

2.2. *Đối với người dùng CSDL*

- Truy vấn thông tin cá nhân (họ tên, tuổi, số điện thoại, trạng thái làm việc, …), các thông tin này là các dữ liệu được đưa vào CSDL với định dạng đã được nêu ở mục 1 của phần II bài báo cáo.
- Truy vấn lương thưởng (lương cơ bản, tăng thưởng, khấu hao, khấu trừ) cùng các thông tin có liên quan như ngày nghỉ, số giờ làm thêm, … của người dùng đó ứng với tài khoản định danh đã được cung cấp. Các thông tin này là các dữ liệu được đưa vào CSDL với định dạng đã được nêu ở mục 1 của phần II bài báo cáo.
- Tính toán tổng lương của nhân viên đó trong tháng tương ứng với tài khoản định danh đã được cung cấp.

3. ***Các bảng trong cơ sở dữ liệu***

➢ **Quy ước**: khóa chính của bảng được **in đậm**, khóa ngoài của bảng tham chiếu đến khóa chính của bảng khác được *in nghiên và gạch chân*.

1. employee (**employee_id**, employee_name, age, gender, address, phone, email, nationality, join_date, leave_date, basic_salary, coefficients_salary, *position_id*, employee_status_id)

2. employee_status (***employee_status_id***, employee_status_name)

3. position (**position_id**, position_name, department_id)

4. department (**department_id**, department_name)

5. overtime (**overtime_id**, *employee_id*, overtime_date, hour_start, hour_end, total_hours, overtime_value)

6. incentive (**incentive_id**, *employee_id*, incentive_value)

7. position_allowance (***position_id***, ***allowance_id***)

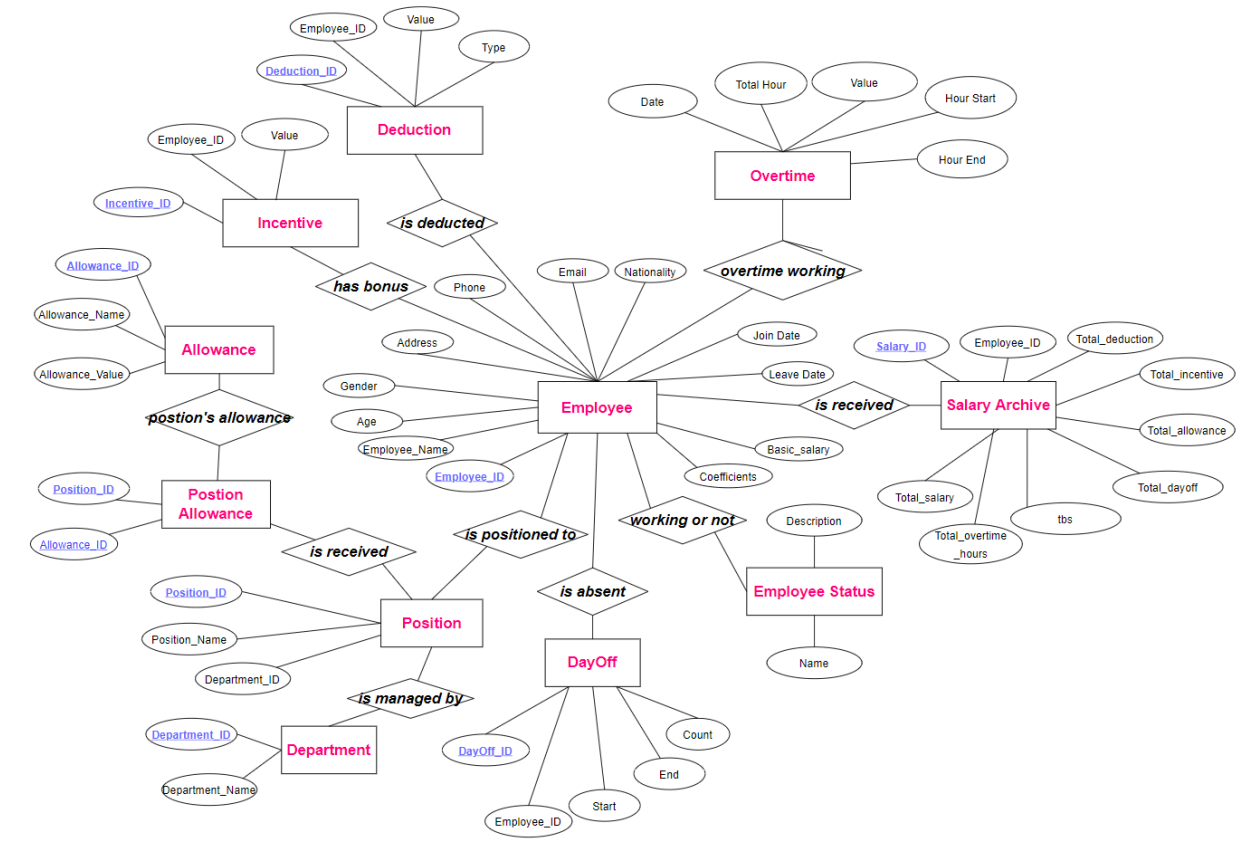
8. allowance (**allowance_id**, allowance_name, allowance_value)

9. day_off (**dayoff_id**, *employee_id*, start_dayoff_date, end_dayoff_date, count_dayoff)

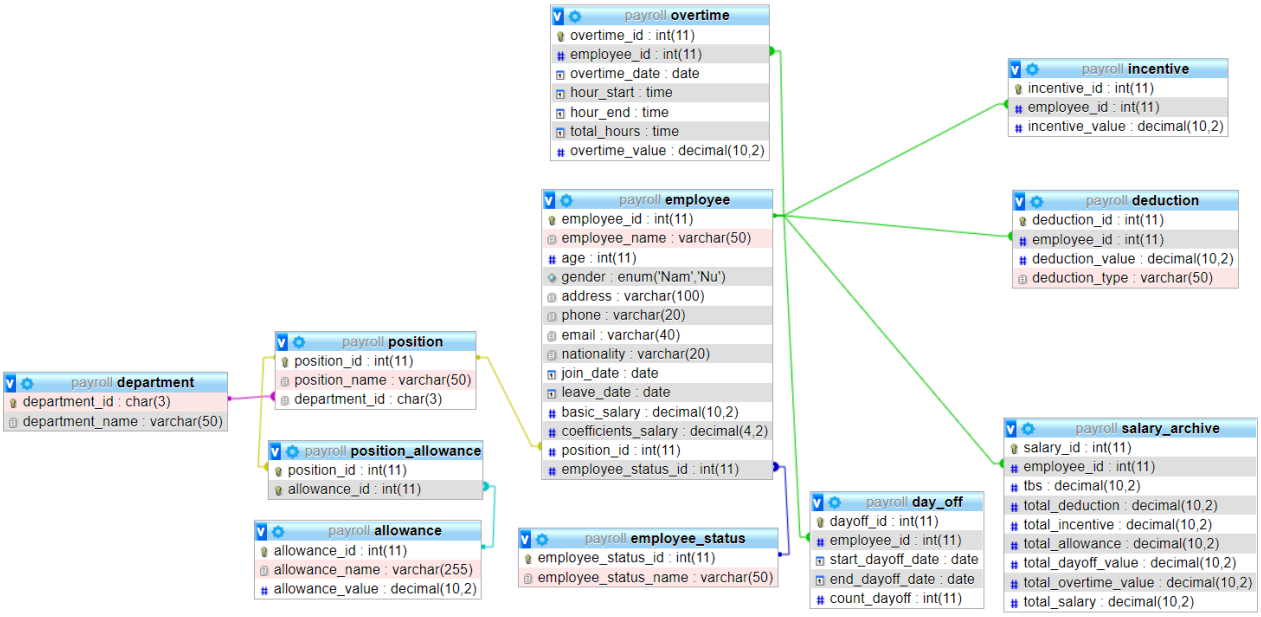
10. deduction (**deduction_id**, *employee_id*, deduction_value, deduction_type)

11. salary_archive (**salary_id**, *employee_id*, tbs, total_deduction, total_incentive, total_allowance, total_dayoff_value, total_overtime_value, total_salary)

4. *Sơ đồ thực thể liên kết ERD*



5. ***Lược đồ quan hệ***



PHẦN III: PHÂN CÔNG & TỔ CHỨC THỰC HIỆN

Các công việc chính gồm:

- Lên kế hoạch làm việc nhóm, lên ý tưởng, xây dựng bài toán, thiết kế sơ khai CSDL và mục đích CSDL hướng tới: cả 4 thành viên trong nhóm cùng tham gia thực hiện.
- Thiết kế các sơ đồ phục vụ quá trình xây dựng CSDL: Nguyễn Trọng Tuệ, Lê Đình Tuyên.
- Thiết kế file thuyết trình PowerPoint: Lê Đình Tuyên.
- Viết báo cáo bài tập lớn: Nguyễn Trọng Tuệ.
- Tạo bảng, thiết kế dữ liệu, nhập và hiệu chỉnh dữ liệu: Nguyễn Trọng Tuệ (1 bảng), Lê Đình Tuyên (2 bảng), Nguyễn Anh Tuấn, Nguyễn Hữu Tuấn (4 bảng).
- Thảo luận các vấn đề liên quan đến CSDL (xung đột có thể xảy ra giữa các bảng, các trường dữ liệu; dữ liệu mẫu xa rời, trái với thực tế; hiệu chỉnh số lượng bản ghi, …) cũng như xây dựng các câu truy vấn CSDL: các thành viên cùng tham gia các buổi họp nhóm để cùng đóng góp, xây dựng.

PHẦN IV: TRUY VẤN CSDL

Các câu truy vấn được chọn để trình bày được đánh dấu bằng màu đỏ

LÊ ĐÌNH TUYÊN 20194715

Câu 1: **Đưa ra thông tin những nhân viên có địa chỉ ở Ninh Bình**

```
SELECT * FROM employee WHERE address = 'Ninh Bình';
```

Câu 2: **Đưa ra thông tin những nhân viên có tên bắt đầu bằng 'N'**

```
SELECT * FROM employee WHERE employee_name LIKE 'N%';
```

Câu 3: **Đưa ra những nhân viên ở phòng kỹ thuật - CNTT**

```
SELECT * FROM employee, position, department WHERE
employee.position_id = position.position_id AND position.department_id =
department.department_id AND department.department_name = 'Phòng kỹ
thuật - CNTT';
```

Câu 4: **Đưa ra số lượng nhân viên Nữ đang nghỉ thai sản**

```
SELECT COUNT(employee.employee_id) AS So_Luong FROM employee,
employee_status WHERE employee.employee_status_id =
employee_status.employee_status_id AND
employee_status.employee_status_name = 'Nghỉ thai sản' AND gender =
'Nu' AND leave_date IS NULL;
```

Câu 5: **Đưa ra tỉnh thành có số lượng nhân viên nhiều nhất**

```
SELECT address FROM employee GROUP BY address HAVING
COUNT(employee_id) >= ALL(SELECT COUNT(employee_id) FROM
employee GROUP BY address);
```

Câu 6: **Tạo View với mã nhân viên, tên nhân viên, chức vụ, phòng ban, bảng lương với tổng lương giảm dần**

```
CREATE VIEW view1 AS SELECT employee.employee_id AS MSNV,
employee.employee_name AS Ten, position.position_name AS Chuc_vu,
department.department_name AS Phong_ban, salary_archive.total_salary
AS Tong_luong

FROM employee, position, department, salary_archive

WHERE employee.employee_id = salary_archive.employee_id AND
employee.position_id = position.position_id AND position.department_id =
department.department_id ORDER BY total_salary DESC;
```

Câu 7: **Đưa ra nhân viên làm việc lâu năm nhất**

```
DELIMITER $$

CREATE FUNCTION getTimeEmployeeWork(join_date DATE, leave_date
DATE)

RETURNS INT

DETERMINISTIC

BEGIN

  DECLARE time_work INT;

  IF leave_date IS NULL THEN SET time_work = YEAR(NOW()) -
YEAR(join_date);

  ELSE SET time_work = YEAR(leave_date) - YEAR(join_date);

  END IF;

RETURN time_work;

END $$

DELIMITER ;
```

```
SELECT * FROM employee WHERE getTimeEmployeeWork(join_date,
leave_date) = (SELECT MAX(getTimeEmployeeWork(join_date,
leave_date)) FROM employee);
```

Câu 8: **Lưu lại log sau khi chỉnh sửa bảng employee**

```
CREATE TABLE employee_log (
  id INT NOT NULL AUTO_INCREMENT,
  content_changed VARCHAR(1000),
  username_changed VARCHAR(30),
  at_time DATETIME,
  PRIMARY KEY (id)
);

DELIMITER $$
CREATE TRIGGER employee_log_update AFTER UPDATE ON employee
FOR EACH ROW
BEGIN
  SET @log = CONCAT('employee_id: ', OLD.employee_id, ', ');
  IF NEW.employee_id != OLD.employee_id THEN SET @log =
CONCAT(@log, 'id: ', OLD.employee_id, '->', NEW.employee_id, '; '); END IF;
  IF NEW.employee_name != OLD.employee_name THEN SET @log =
CONCAT(@log, 'name: ', OLD.employee_name, '->', NEW.employee_name, '; ');
END IF;
  IF NEW.age != OLD.age THEN SET @log = CONCAT(@log, 'age: ', OLD.age,
'->', NEW.age, '; '); END IF;
  IF NEW.gender != OLD.gender THEN SET @log = CONCAT(@log, 'gender: ',
OLD.gender, '->', NEW.gender, '; '); END IF;
  IF NEW.address != OLD.address THEN SET @log = CONCAT(@log, 'address:
', OLD.address, '->', NEW.address, '; '); END IF;
```

```
  IF NEW.phone != OLD.phone THEN SET @log = CONCAT(@log, 'phone: ', OLD.gender, '->', NEW.gender, '; '); END IF;

  IF NEW.email != OLD.email THEN SET @log = CONCAT(@log, 'email: ', OLD.email, '->', NEW.email, '; '); END IF;

  IF NEW.nationality != OLD.nationality THEN SET @log = CONCAT(@log, 'nationality: ', OLD.nationality, '->', NEW.nationality, '; '); END IF;

  IF NEW.join_date != OLD.join_date THEN SET @log = CONCAT(@log, 'nationality: ', OLD.join_date, '->', NEW.join_date, '; '); END IF;

  IF NEW.leave_date != OLD.leave_date THEN SET @log = CONCAT(@log, 'nationality: ', OLD.leave_date, '->', NEW.leave_date, '; '); END IF;

  IF NEW.basic_salary != OLD.basic_salary THEN SET @log = CONCAT(@log, 'basic_salary: ', OLD.basic_salary, '->', NEW.basic_salary, '; '); END IF;

  IF NEW.coefficients_salary != OLD.coefficients_salary THEN SET @log = CONCAT(@log, 'coefficients: ', OLD.coefficients_salary, '->', NEW.coefficients_salary, '; '); END IF;

  IF NEW.position_id != OLD.position_id THEN SET @log = CONCAT(@log, 'position_id: ', OLD.position_id, '->', NEW.position_id, '; '); END IF;

  IF NEW.employee_status_id != OLD.employee_status_id THEN SET @log = CONCAT(@log, 'status_id: ', OLD.employee_status_id, '->', NEW.employee_status_id, '; '); END IF;

  INSERT INTO employee_log (content_changed, username_changed, at_time) VALUES (@log, CURRENT_USER(), NOW());

END $$

DELIMITER ;
```

Câu 9: **Cập nhật bảng lương cho các nhân viên sử dụng procedure**

```
DELIMITER $$

CREATE FUNCTION cv(x INT)

RETURNS INT

DETERMINISTIC

BEGIN
```

```
  DECLARE result INT;
  IF x IS NULL THEN
    SET result = 0;
  ELSE
    SET result = x;
  END IF;
  RETURN result;
END $$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE updateSalary()
BEGIN
  DELETE FROM salary_archive;
  INSERT INTO salary_archive (employee_id, tbs, total_deduction,
total_incentive, total_allowance, total_dayoff_value, total_overtime_value,
total_salary)
  SELECT table5.employee_id, table5.tbs, Total_Deduction,
Total_Incentive_Value, cv(Total_Allowance), Total_Dayoff_Value,
Total_Overtime_Value,  (tbs + cv(Total_Allowance) + Total_Incentive_Value +
Total_Overtime_Value - Total_Deduction - Total_Dayoff_Value) AS Total_Salary
FROM
  (SELECT table1.employee_id, table1.position_id,
table1.basic_salary*table1.coefficients_salary AS tbs, Total_Deduction,
Total_Incentive_Value, Total_Dayoff_Value, Total_Overtime_Value FROM
  (SELECT employee.employee_id, position_id, basic_salary, coefficients_salary,
cv(SUM(deduction_value)) AS Total_Deduction FROM employee left join
deduction on employee.employee_id = deduction.employee_id WHERE
leave_date IS NULL OR (MONTH(leave_date) = MONTH(NOW()) AND
YEAR(leave_date) = YEAR(NOW())) GROUP BY employee.employee_id)
table1,
```

```
  (SELECT employee.employee_id, cv(SUM(overtime_value)) AS
Total_Overtime_Value FROM employee left join overtime on
employee.employee_id = overtime.employee_id WHERE leave_date IS NULL OR
(MONTH(leave_date) = MONTH(NOW()) AND YEAR(leave_date) =
YEAR(NOW())) GROUP BY employee.employee_id) table2,

  (SELECT employee.employee_id, cv(SUM(count_dayoff))*100000 AS
Total_Dayoff_Value FROM employee left join day_off on employee.employee_id
= day_off.employee_id WHERE leave_date IS NULL OR (MONTH(leave_date) =
MONTH(NOW()) AND YEAR(leave_date) = YEAR(NOW())) GROUP BY
employee.employee_id) table3,

  (SELECT employee.employee_id, cv(SUM(incentive_value)) AS
Total_Incentive_Value FROM employee left join incentive on
employee.employee_id = incentive.employee_id WHERE leave_date IS NULL
OR (MONTH(leave_date) = MONTH(NOW()) AND YEAR(leave_date) =
YEAR(NOW())) GROUP BY employee.employee_id) table4

  WHERE table1.employee_id = table2.employee_id AND table1.employee_id =
table3.employee_id AND table1.employee_id = table4.employee_id) table5 left
join

  (SELECT position_id, SUM(allowance_value) AS Total_Allowance FROM
position_allowance, allowance WHERE position_allowance.allowance_id =
allowance.allowance_id GROUP BY position_id) table6

  on table5.position_id = table6.position_id;

END $$

DELIMITER ;


CALL updateSalary();
```

Câu 10: **Cập nhật, backup bảng lương và xóa các dữ liệu của tháng trước để cập nhật dữ liệu mới cho tháng sau**

```
    DELIMITER $$

    CREATE DEFINER=`root`@`localhost` EVENT `BackupAndDelete`

    ON SCHEDULE EVERY 1 MONTH ON COMPLETION PRESERVE
    ENABLE

    DO
```

```
BEGIN
  CALL updateSalary();
  SET @sql_text = CONCAT("SELECT * FROM salary_archive INTO
OUTFILE '/var/lib/mysql-files/" , DATE_FORMAT(NOW(),
'%Y%m%d_%H%i%s') , "_salary_archive.csv"', " FIELDS TERMINATED
BY ',' OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY
'\n"');
  PREPARE s1 FROM @sql_text;
  EXECUTE s1;
  DEALLOCATE PREPARE s1;
  DELETE FROM day_off;
  DELETE FROM deduction;
  DELETE FROM incentive;
  DELETE FROM overtime;
  DELETE FROM salary_archive;
END $$
DELIMITER ;
```

NGUYỄN TRỌNG TUỆ 20194710

Câu 1: **Đưa ra trạng thái làm việc của một nhân viên bất kì**

```
SELECT employee_id, employee_name, employee_status_name FROM
employee INNER JOIN employee_status USING(employee_status_id)
WHERE employee_id = 250;
```

Câu 2: **Tính tổng khoản khấu trừ do nhân viên công ty làm hỏng đồ đạc, thiết bị**

```
SELECT SUM(deduction_value) FROM deduction WHERE deduction_type
= 'Làm hỏng đồ đạc, thiết bị';
```

Câu 3: **Đưa ra các nhân viên có nghỉ phép trong khoảng từ ngày 12 đến ngày 14 tháng 1 năm 2022**

```
SELECT employee_id, employee_name, start_dayoff_date, end_dayoff_date
FROM employee INNER JOIN day_off USING(employee_id) WHERE
start_dayoff_date >= '2022-01-12' AND end_dayoff_date <= '2022-01-14';
```

Câu 4: **Tính tổng khoản phí nhận được từ việc làm thêm của từng nhân viên làm việc tại phòng Quản trị chất lượng**

```
SELECT employee_id, employee_name, SUM(overtime_value) FROM
employee INNER JOIN overtime USING(employee_id) INNER JOIN
position USING(position_id) INNER JOIN department
USING(department_id) WHERE department_name = 'Phòng quản trị chất
lượng' GROUP BY employee_id ORDER BY employee_id ASC;
```

Câu 5: **Đưa ra thông tin phòng ban có nhiều nhân viên làm việc nhất**

```
SELECT department_id, department_name, COUNT(department_id) AS
number_of_employees FROM employee INNER JOIN position USING
(position_id) INNER JOIN department USING(department_id) GROUP BY
department_id HAVING COUNT(department_id) >= ALL (SELECT
COUNT(department_id) FROM employee INNER JOIN position USING
(position_id) INNER JOIN department USING(department_id) GROUP BY
department_id);
```

Câu 6: **Tính tổng số nhân viên làm thêm giờ trong tháng sử dụng VIEW**

CREATE VIEW OvertimeID AS SELECT DISTINCT employee_id FROM overtime;

SELECT COUNT(employee_id) AS NumberOfEmployeesWorkingOvertime from OvertimeID;

Câu 7: Quản lý các cập nhật liên quan đến phòng ban (sử dụng TRIGGER)

create table Department_Logs(id int AUTO_INCREMENT NOT NULL PRIMARY KEY, id_before char(3) DEFAULT NULL, id_after char(3) DEFAULT NULL,name_before varchar(50) DEFAULT NULL, name_after varchar(50) DEFAULT NULL, change_at DATETIME DEFAULT NULL);

DELIMITER //

CREATE TRIGGER before_department_update BEFORE UPDATE ON department FOR EACH ROW BEGIN

INSERT INTO Department_Logs(id_before, id_after, name_before, name_after, change_at)

VALUES

(OLD.department_id, NEW.department_id, OLD.department_name, NEW.department_name, NOW());

END//

DELIMITER ;

UPDATE department SET department_name = 'Phòng quản trị nhân lực' where department_id = 'P01';

UPDATE department SET department_id = 'T01' where department_id = 'P01';

UPDATE department SET department_id = 'P01' where department_id = 'T01';

Select * from department;

select * from Department_Logs order by change_at desc;

Câu 8: **Đưa ra tổng số nhân viên đang theo trạng thái của công ty (sử dụng PROCEDURE)**

```
DELIMITER $$

CREATE PROCEDURE GetNumberOfEmployees(IN
p_employee_status_name varchar(50)) BEGIN DECLARE totalEmployees
INT DEFAULT 0; select count(*) into totalEmployees from employee
where employee_status_id in (select employee_status_id from
employee_status where employee_status_name =
p_employee_status_name); select totalEmployees; END$$

DELIMITER ;

CALL GetNumberOfEmployees('Đang làm việc');
```

Câu 9: **Tính tổng lương của 1 nhân viên bất kỳ trong tháng sử dụng PROCEDURE và không sử dụng bảng salary_archive**

```
DELIMITER $$

CREATE PROCEDURE GetTotalSalaryOfAnEmployee (

IN p_employee_id int, OUT p_total_salary DECIMAL(10,2))

BEGIN

DECLARE overtime_sum DECIMAL(10,2) DEFAULT 0;

DECLARE dayoff_sum DECIMAL(10,2) DEFAULT 0;

DECLARE basic_salary_sum DECIMAL(10,2) DEFAULT 0;

DECLARE incentive_sum DECIMAL(10,2) DEFAULT 0;

DECLARE allowance_sum DECIMAL(10,2) DEFAULT 0;

 DECLARE deduction_sum DECIMAL(10,2) DEFAULT 0;

 select (basic_salary * coefficients_salary) into basic_salary_sum from
employee  where employee_id = p_employee_id;

IF (select sum(allowance_value) from employee inner join position
using(position_id) inner join position_allowance using(position_id) inner
```

```
join allowance using(allowance_id) where employee_id = p_employee_id) >
0 THEN select sum(allowance_value) into allowance_sum from employee
inner join position using(position_id) inner join position_allowance
using(position_id) inner join allowance using(allowance_id) where
employee_id = p_employee_id; END IF;

IF (select SUM(incentive_value) from incentive where employee_id =
p_employee_id) > 0 THEN select sum(incentive_value) into incentive_sum
from incentive  where employee_id = p_employee_id;  END IF;

IF (select SUM(overtime_value) from overtime where employee_id =
p_employee_id) >0 THEN select sum(overtime_value) into overtime_sum
from employee inner join overtime using(employee_id) where employee_id
= p_employee_id; END IF;

IF (select sum(count_dayoff) from day_off where employee_id =
p_employee_id ) > 0 THEN select (sum(count_dayoff)) into dayoff_sum
from employee inner join day_off using(employee_id) where employee_id =
p_employee_id; END IF;

IF (select SUM(deduction_value) from deduction where employee_id =
p_employee_id) > 0 THEN select sum(deduction_value) into
deduction_sum from deduction  where employee_id =
p_employee_id;   END IF;

 SET p_total_salary = ((basic_salary_sum + overtime_sum + incentive_sum
+ allowance_sum) - (dayoff_sum * 100000 + deduction_sum));

END$$

DELIMITER ;

CALL  GetTotalSalaryOfAnEmployee(2498, @total);

select @total;
```

Câu 10: **Nhân viên nghỉ phép từ 3 ngày trở lên trong tháng sẽ nhận được nhắc nhở từ công ty và tăng mức cảnh cáo, kiểm tra xem nhân viên đó có bị đặt vào trạng thái cảnh cáo trong tháng này hay không để đưa phương án xử lý thích hợp (sử dụng FUNCTION và VIEW)**

- Các mức cảnh cáo:

- Trên 6 ngày: MỨC 3
- 5 - 6 ngày: MỨC 2
- 3 - 4 ngày: MỨC 1
- Dưới 3 ngày: MỨC 0

```
DELIMITER $$
CREATE FUNCTION HavingDayOff( dayoff INT )
RETURNS VARCHAR(12)
DETERMINISTIC
BEGIN
DECLARE aLevel VARCHAR(12);
IF dayoff > 6 THEN SET aLevel = 'MỨC 3';
ELSEIF (dayoff <= 6 AND dayoff > 4) THEN SET aLevel = 'MỨC 2';
ELSEIF (dayoff <= 4 AND dayoff > 2) THEN SET aLevel = 'MỨC 1';
ELSE SET aLevel = 'MỨC 0'; END IF; RETURN (aLevel); END$$
DELIMITER ;
CREATE VIEW SUMDayOff AS SELECT employee.employee_id as
employee_id, employee_name, sum(count_dayoff) as total_dayoff from
employee LEFT JOIN day_off ON employee.employee_id =
day_off.employee_id group by employee_id;
SELECT employee_id, employee_name, total_dayoff,
HavingDayOff(total_dayoff) as AlertLevel FROM SUMDayOff Order by
total_dayoff desc, HavingDayOff(total_dayoff) desc;
SELECT employee_id, employee_name, total_dayoff,
HavingDayOff(total_dayoff) as AlertLevel FROM SUMDayOff WHERE
employee_id = 1549;
```

NGUYỄN ANH TUẤN 20184220

Câu 1: **Số nhân viên đang còn là nhân viên của công ty (bao gồm nhân viên đang làm việc, nghỉ phép)**

```
SELECT count(E1.employee_id) FROM employee E1
INNER JOIN employee_status E2
ON E1.employee_status_id = E2.employee_status_id
WHERE E2.employee_status_name NOT IN('Về hưu', 'Thôi việc');
```

Câu2: **đưa ra danh sách bao gồm tên nhân viên, mã số của nhân viên và số giờ tăng ca của nhân viên đó trong ngày 20-10-2022 theo thứ tự giảm dần của số giờ tăng ca và tăng dần của mã số sinh viên**

```
SELECT E.employee_name, E.employee_id, O.total_hours
FROM overtime O
INNER JOIN employee E
ON O.employee_id = E.employee_id
WHERE overtime_date = '2022-01-20'
ORDER BY total_hours DESC, employee_id ASC;
```

Câu 3: **đưa ra danh sách nhân viên không làm tăng ca kể từ ngày 1-1-2022**

```
SELECT count(employee_id)
FROM employee
WHERE employee_id NOT IN(
```

```
SELECT employee_id FROM overtime
WHERE overtime_date >= '2022-01-20');
```

Câu 4: **Danh sách những nhân viên phá hoại tài sản của công ty không dưới 2 lần**

```
SELECT employee_id, count(deduction_id) AS SL
FROM deduction
WHERE deduction_type = 'Làm hỏng đồ đạc, thiết bị'
GROUP BY employee_id HAVING SL >= 2;
```

Câu 5: **Sử dụng VIEW in ra danh sách số lượng nhân viên xin nghỉ phép với mỗi loại lí do nghỉ khác nhau;**

```
CREATE VIEW DS_NGHI AS
SELECT count(e1.employee_id) AS SL_NGHIPHEP,
e2.employee_status_name
FROM employee e1 INNER JOIN employee_status e2
ON e1.employee_status_id = e2.employee_status_id
WHERE e2.employee_status_name LIKE 'Nghỉ%'
GROUP BY e2.employee_status_name;

SELECT * FROM DS_NGHI;
```

Câu 6: **Viết 1 store cho phép xóa tất cả nhân viên đã nghỉ hưu**

```
DROP PROCEDURE IF EXISTS Delete_Employee_With_Status;
DELIMITER $$
CREATE PROCEDURE  Delete_Employee_With_Status(IN
in_emp_statusid INT UNSIGNED)
```

```
BEGIN

DELETE FROM employee WHERE employee_status_id =
in_emp_statusid;

END$$

DELIMITER ;

CALL  Delete_Employee_With_Status(5);
```

Câu 7: **Tạo trigger Không cho phép người dùng thêm bất kỳ nhân viên nào vào phòng marketing nữa, khi thêm thì hiện ra thông báo "Department marketing cannot add more employees"**

```
DROP TRIGGER IF EXISTS Trg_notaddmarketingemployee;

DELIMITER $$

CREATE TRIGGER Trg_notaddmarketingemployee

BEFORE INSERT ON `employee`

FOR EACH ROW

BEGIN

DECLARE emp_ID TINYINT;

SELECT p.position_id INTO emp_ID

FROM position p INNER JOIN department d

ON p.department_id = d.department_id

WHERE d.department_name = 'Phòng Marketing' AND p.position_name =
'Nhân viên';

IF (NEW.position_id = emp_ID) THEN

SIGNAL SQLSTATE '12345'
```

```
SET MESSAGE_TEXT = 'Department marketing cannot add more
employees';

END IF;

END$$

DELIMITER ;


insert into employee() values (5000,'anh tuan', 22,'Nam', 'Thanh Hoa',
'094493778','anhtuan@gmail.com','Viet Nam', '2020-01-01', NULL,
1000000.00, 3,8, 1);
```

Câu 8: **Tạo trigger không cho phép người dùng xóa nhân viên đang trong trạng thái đang làm việc còn lại các nhân viên với trạng thái khác thì sẽ cho phép xóa và sẽ xóa tất cả các thông tin liên quan tới nhân viên đó**

```
DELIMITER $$

CREATE TRIGGER trigger_delete_account

BEFORE DELETE ON `employee`

FOR EACH ROW

BEGIN

DECLARE v_trangthai VARCHAR(50);

SET v_trangthai = 1;

IF (OLD.employee_id = v_trangthai) THEN

SIGNAL SQLSTATE '12345'

SET MESSAGE_TEXT = 'Nhân viên đang làm việc, không thể xóa!!';

END IF;

END $$
```

```
DELIMITER ;


DELETE FROM employee WHERE employee_id = '1';
```

Câu 9: **Viết 1 store cho phép người dùng nhập vào 1 chuỗi và trả về nhân viên có tên chứa chuỗi của người dùng nhập vào hoặc trả về phòng ban có name chứa chuỗi của người dùng nhập vào**

```
DROP PROCEDURE IF EXISTS sp_getempnameordepname;

DELIMITER $$
CREATE PROCEDURE sp_getempnameordepname
( IN var_String VARCHAR(50), IN flag TINYINT )
BEGIN
IF flag = 1 THEN
	SELECT e.employee_name FROM employee e
	WHERE e.employee_name LIKE CONCAT("%",var_String,"%");
ELSE
	SELECT d.department_name FROM department d
	WHERE d.department_name LIKE CONCAT("%",var_String,"%");
END IF;
END$$
DELIMITER ;

Call sp_getempnameordepname('An', 1);
```

Câu 10: **Viết 1 store cho phép người dùng nhập vào thông tin id, name, address, phone trong store sẽ tự động gán: email sẽ tự động thêm đuôi @gmail.com sau name đã loại bỏ khoảng trắng**

```
DROP PROCEDURE IF EXISTS sp_insert_employee;
DELIMITER $$
CREATE PROCEDURE sp_insert_employee
( IN var_employee_id INT,
 IN var_employee_name VARCHAR(50),
 IN var_address VARCHAR(100),
 IN var_phone VARCHAR(20)
```

```
)
BEGIN
DECLARE v_email VARCHAR(50) DEFAULT
CONCAT(REPLACE(LOWER(var_employee_name),' ', ''), '@gmail.com');
DECLARE v_age INT UNSIGNED DEFAULT 20;
DECLARE v_gender VARCHAR(20) DEFAULT 'Nam';
DECLARE v_nationality VARCHAR(30) DEFAULT 'Việt Nam';
DECLARE v_join_date  DATE DEFAULT now();
DECLARE v_leave_date  DATE DEFAULT NULL;
DECLARE v_salary_basic  DECIMAL(10,2) DEFAULT 20000000.00;
DECLARE v_position_id  INT DEFAULT 48;
DECLARE v_employee_status_id  INT DEFAULT 1;

INSERT INTO `employee`(`employee_id`, `employee_name`, `age`,
`gender`, `address`,
`phone`, `email`, `nationality`, `join_date`, `leave_date`,
`basic_salary`, `position_id`, `employee_status_id`)
VALUES (var_employee_id, var_employee_name, v_age, v_gender,
var_address,
 var_phone, v_email, v_nationality, v_join_date, v_leave_date,
 v_salary_basic, v_position_id, v_employee_status_id);
END$$
DELIMITER ;

DELETE FROM employee WHERE employee_id = 3000;
Call sp_insert_employee('3000','Bao Ngoc', 'Thanh Hoa', '0123423423');
SELECT * FROM employee ORDER BY employee_id DESC;
```

NGUYỄN HỮU TUẤN 20184221

Câu 1: **Top 10 người có lương cao nhất**

```
SELECT * FROM `salary_archive` ORDER BY `total_salary` DESC
LIMIT 10;
```

Câu 2: **Đưa ra thông tin người đc thưởng 2 lần trở lên**

```
SELECT e.*,COUNT(i.employee_id) as TotalIncentive

FROM employee e,incentive i

WHERE e.employee_id = i.employee_id

GROUP BY i.employee_id

HAVING COUNT(i.employee_id) >= 2;
```

Câu 3: **Đưa ra danh sách những người đã nghỉ hưu hoặc thôi việc**

```
SELECT e.employee_name, e.gender,e.join_date,e.leave_date,
es.employee_status_name

FROM employee e

INNER JOIN employee_status es ON

e.employee_status_id = es.employee_status_id

WHERE es.employee_status_name = 'Về hưu' OR
es.employee_status_name = 'Thôi việc';
```

Câu 4: **Đưa ra số lượng trưởng phòng, phó trưởng phòng,thư ký, nhân viên**

```
SELECT  p.position_name, COUNT(p.position_name) AS Total

FROM employee e,position p

WHERE e.position_id = p.position_id
```

```
GROUP BY p.position_name;
```

Câu 5: **Phòng có số lượng nhân viên làm việc đông nhất(gồm trưởng phòng, phó trưởng phòng, thư ký,nhân viên)**

```
SELECT d.department_name,COUNT(p.department_id) as Total

FROM employee e,position p,department d

WHERE e.position_id = p.position_id

AND p.department_id = d.department_id

GROUP BY p.department_id

ORDER BY COUNT(p.department_id) DESC

LIMIT 1 ;
```

Câu 6: **Đưa ra 3 tỉnh có nhiều nhân viên nhất và vẫn đang làm việc**

```
SELECT e.address,COUNT(e.address) as Total

FROM employee e

INNER JOIN employee_status es

ON e.employee_status_id = es.employee_status_id

WHERE es.employee_status_name = 'Đang làm việc' OR
es.employee_status_name = 'Nghỉ thai sản' OR es.employee_status_name =
'Nghỉ ốm'

GROUP BY e.address

ORDER BY COUNT(e.address) DESC

LIMIT 3;
```

Câu 7: **Nhân viên làm ở phòng Phòng kỹ thuật – CNTT và được thưởng nhiều nhất**

```
CREATE VIEW TotalIncentive
AS SELECT e1.employee_id,e1.employee_name,SUM(i.incentive_value) as Total
FROM employee e1,incentive i
WHERE e1.employee_id = i.employee_id
AND e1.employee_id IN (
    SELECT e2.employee_id
    FROM employee e2,position p,department d
    WHERE e2.position_id = p.position_id
    AND p.department_id = d.department_id
    AND d.department_name = 'Phòng kỹ thuật - CNTT'
)
GROUP BY i.employee_id
ORDER BY Total DESC;


SELECT *
FROM TotalIncentive
WHERE Total = (
    SELECT MAX(Total)
    FROM TotalIncentive);
```

Câu 8: **Update lương mỗi khi thêm tiền thưởng**

```
SELECT * FROM salary_archive WHERE employee_id = 3;


DELIMITER $$
 CREATE TRIGGER after_incentive_insert
 AFTER INSERT ON incentive
 FOR EACH ROW
BEGIN
 UPDATE salary_archive
SET
 total_incentive = total_incentive + NEW.incentive_value,
 total_salary = total_salary + NEW.incentive_value
WHERE
        employee_id = NEW.employee_id;
END$$
 DELIMITER ;

INSERT INTO incentive (employee_id, incentive_value) VALUES (3,
100000);


SELECT * FROM salary_archive WHERE employee_id = 3;
```

Câu 9: **Update lương mỗi khi thêm khấu trừ**

```
SELECT * FROM salary_archive WHERE employee_id = 3;
```

```
DELIMITER $$
 CREATE TRIGGER after_deduction_insert
 AFTER INSERT ON deduction
 FOR EACH ROW
BEGIN
 UPDATE salary_archive
SET
 total_deduction = total_deduction + NEW.deduction_value,
 total_salary = total_salary - NEW.deduction_value
WHERE
        employee_id = NEW.employee_id;
END$$
 DELIMITER ;

INSERT INTO deduction (employee_id, deduction_value, deduction_type)
VALUES (3, 100000, 'Làm hỏng đồ đạc');

SELECT * FROM salary_archive WHERE employee_id = 3;
```

Câu 10: **Thưởng cho 3 phòng ban có trung bình lương cao nhất (sử dụng function và view)**

```
DELIMITER $$
CREATE FUNCTION BonusSalary(credit INT)
RETURNS VARCHAR(20)
```

```
DETERMINISTIC

BEGIN

DECLARE bonus DECIMAL(10,2);

IF credit = 1 THEN SET bonus = 5000000;

ELSEIF (credit = 2) THEN SET bonus = 4000000;

ELSEIF (credit = 3) THEN SET bonus = 3000000;

END IF;

RETURN (bonus);

END$$

DELIMITER ;



CREATE VIEW AverageSalary

AS SELECT d.department_name,AVG(s.total_salary) as average_salary,
ROW_NUMBER() OVER(ORDER BY AVG(s.total_salary) DESC) AS
Num

FROM salary_archive s,employee e,position p,department d

WHERE s.employee_id = e.employee_id

AND e.position_id = p.position_id

AND p.department_id = d.department_id

GROUP BY d.department_name

ORDER BY average_salary DESC

LIMIT 3;
```

```
SELECT department_name,average_salary, BonusSalary(Num) AS Bonus

FROM AverageSalary;
```