A. Implement logging for changes in table “employees” using triggers:
1. Create table “log” with columns: ID (primay key, integer, autoincrement),TS (DATETIME), USR(VARCHAR(63)), EV(VARCHAR(15)),MSG(VARCHAR(255))

```
CREATE TABLE log (
LOG_ID INTEGER UNSIGNED AUTO_INCREMENT PRIMARY KEY,
DT DATETIME NOT NULL,
USR VARCHAR(63),
EV VARCHAR(15),
MSG VARCHAR(255)
);
```

2. Create trigger that add to the “log” table current time (use function NOW()), the current user (USER()), the text “add” and the employee full name when new employee is added

```
DELIMITER $$
CREATE TRIGGER emp_ai_log
AFTER INSERT
ON employee
FOR EACH ROW
BEGIN
      INSERT log (TS,USR,EV,MSG)
      VALUES (NOW(),USER(),"add",CONCAT(NEW.FIRST_NAME,' ',NEW.LAST_NAME));
END $$
DELIMITER ;
```

3. Same but when “employee” table has updated row

```
DELIMITER $$
CREATE TRIGGER emp_au_log
AFTER UPDATE
ON employee
FOR EACH ROW
BEGIN
      INSERT log (TS,USR,EV,MSG) VALUES
      (NOW(),USER(),"update",CONCAT(NEW.FIRST_NAME,' ',NEW.LAST_NAME));
END $$
DELIMITER ;
```

4. Same but when some employee is deleted

```
DELIMITER $$
CREATE TRIGGER emp_bd_log
BEFORE DELETE
ON employee
FOR EACH ROW
BEGIN
      INSERT log (TS,USR,EV,MSG) VALUES
      (NOW(),USER(),"delete",CONCAT(OLD.FIRST_NAME,' ',OLD.LAST_NAME));
END $$
DELIMITER ;
```

5. Modify, add and delete some rows in “employee” table and observe the content of “log” table

B. Implement table with current salary property over the whole company (min, max, average, count, last updated), updated by triggers every time when employees are added, removed or changed;

1. Create table status with columns: PK (Primary key, Integer, default 0), SALARY_MIN, SALARY_MAX, SALARY_AVG (FLOAT), EMP_COUNT (INTEGER UNSIGNED, DEFAULT 0, NOT NULL), LAST_UPD (DATATIME)

```
CREATE TABLE status (
  PK INTEGER UNSIGNED DEFAULT 0 PRIMARY KEY,
  SALARY_AVG FLOAT,
  SALARY_MIN FLOAT,
  SALARY_MAX FLOAT,
  EMP_COUNT INTEGER UNSIGNED DEFAULT 0 NOT NULL,
  LAST_UPD DATETIME
);
```

2. Create query that replaces the single row in “status” with actual values. Use REPLACE … SELECT syntax:

```
REPLACE status (PK,SALARY_AVG,SALARY_MIN,SALARY_MAX,EMP_COUNT,LAST_UPD)
SELECT 0,avg(SALARY),max(SALARY),min(SALARY),COUNT(1),NOW() FROM employee;
```

3. Create triggers in employee table **after** delete that update the content of table “status” using the query from the previous point

```
DELIMITER $$
CREATE TRIGGER emp_ad_status
AFTER DELETE
ON employee
FOR EACH ROW
BEGIN
  REPLACE status (PK,SALARY_AVG,SALARY_MIN,SALARY_MAX,EMP_COUNT,LAST_UPD)
  SELECT 0,avg(SALARY),max(SALARY),min(SALARY),COUNT(1),NOW() FROM employee;
END $$
DELIMITER ;
```

4. Enhance the triggers from points A.2. and A.3 to perform the status table

```
DROP TRIGGER emp_ai_log;
DELIMITER $$
CREATE TRIGGER emp_ai_log
AFTER INSERT
ON employee
FOR EACH ROW
BEGIN
      INSERT log (TS,USR,EV,MSG)
      VALUES (NOW(),USER(),"add",CONCAT(NEW.FIRST_NAME,' ',NEW.LAST_NAME));
      REPLACE status (PK,SALARY_AVG,SALARY_MIN,SALARY_MAX,EMP_COUNT,LAST_UPD)
      SELECT 0,avg(SALARY),max(SALARY),min(SALARY),COUNT(1),NOW() FROM employee;
END $$
DELIMITER ;

DROP TRIGGER emp_au_log;
DELIMITER $$
CREATE TRIGGER emp_au_log
AFTER UPDATE
ON employee
FOR EACH ROW
BEGIN
      INSERT log (TS,USR,EV,MSG) VALUES
      (NOW(),USER(),"update",CONCAT(NEW.FIRST_NAME,' ',NEW.LAST_NAME));
      REPLACE status (PK,SALARY_AVG,SALARY_MIN,SALARY_MAX,EMP_COUNT,LAST_UPD)
      SELECT 0,avg(SALARY),max(SALARY),min(SALARY),COUNT(1),NOW() FROM employee;
END $$
DELIMITER ;
```

5. Modify the “employee” table and observe that changes in “status” table