Question 1

Create an INSURANCE table (insuranceNO, empID, credit_limit, remark) where each employee has only one insurance contract. When employee's information is updated, insurance information must also be updated.

1.1 Write an SQL command to create INSURANCE data table with additional cascade update key constraint (FK) on updates. Note: Define the data type of "remark" as varchar(100 + [4th and 5th digit of your student ID])

Example: If your student ID is 603 "57" 25521, the data type of "remark" is varchar(157).

```
CREATE TABLE IF NOT EXISTS INSURANCE (
        insuranceNo CHAR(10) NOT NULL,
    empID CHAR(10) NOT NULL,
    credit_limit INT (10) NOT NULL,
    remark VARCHAR (117) NOT NULL,
    PRIMARY KEY (INsuranceNo),
    FOREIGN KEY (empID) REFERENCES EMPLOYEE (empID) ON UPDATE cascade
)
```

1.2 Write an update command on the EMPLOYEE table to test this key constraint.

```
UPDATE EMPLOYEE
SET empID = '1010010001'
WHERE empID = 1000010001;
```

Question 2

2.1 Find all customer names who got the discounts greater than "0 + [5th and 6th digit of your student ID]" on "2016-04-12".

Example: If your student ID is 6031 "12" 5521, find all customer names who got the discounts greater than 12

```
SELECT cusName FROM CUSTOMER WHERE cusID IN
(SELECT DISTINCT cusID FROM INVOICE_ITEM it
        INNER JOIN INVOICE i ON it.INvoiceID = i.INvoiceID
    INNER JOIN CUSTOMER c ON i.cusID = c.cusID
    WHERE DATEDIFF(INvoiceDate,'2016-04-12') = 0
    AND discount > 76);
```

2.2 Write a select command to show the cusID, invoiceID, and invoiceDate of all sale transactions these are brought by employee's name = "Charlie Hirono" or "John Moore".

```
SELECT empID, empName, INvoiceID FROM INVOICE i
    INNER JOIN EMPLOYEE e ON e.empID = i.empID
    WHERE empName = "CHARlie HirONo" or empName = "John Moore ";
```

Question 3

3.1 Create a trigger named updated_product that keeps logging information after the cost of product is updated. The details , pid, old cost and new cost of the update are stored in the remark column, and system_date column keeps timestamp (SYSDATE()) of the action.

```
DELIMITER $$
CREATE TRIGGER updated_product
AFTER UPDATE ON PRODUCT
FOR EACH ROW
BEGIN
        IF (old.cost != new.cost) THEN
                INSERT INTO SYSTEM_LOG (remark,system_date)
                VALUES (
                        CONCAT(
                "pid : ", new.pid,
                " old cost : ", old.cost,
                " new cost : ", new.cost
            ),
                        SYSDATE()
                );
        END IF;
END;
$$
DELIMITER ;
```

3.2 Update product's cost from "10400" to 10000 + [6th and 7th number of your student ID]*100 of product id = "1110801234321".

Example: If your student ID is 60310 "25" 521, update product's cost from "10400" to "12500"

```
UPDATE PRODUCT
SET cost = "10620"
WHERE pID = "1110801234321";
```

Question 4

Create a stored procedure named calculate_retire which calculates the retirement payment (400% of the salary) AND retired date (when age = 60 + [8th number of your student ID]).

Example: If your student ID is 6031025 “5” 21, retired age is 65

Expected Result :	Name	Age	retired_date	Salary	Retirement_payment
	XXX YYY	50	14-4-2016	40,000.00	160,000.00

Hint : DATE_ADD(SYSDATE(), INTERVAL 4 YEAR) function add a specified time interval to date. TIMESTAMPDIFF(YEAR, '2010-01-10' , SYSDATE()) returns a value after subtracting a datetime expression from another.

```
DELIMITER $$
CREATE PROCEDURE calculate_retire()
BEGIN
SELECT
    empName as "Name",
    TIMESTAMPDIFF(YEAR, birthdate, SYSDATE()) as Age,
    DATE_FORMAT(DATE_ADD(birthdate, INTERVAL 60 YEAR), '%d-%m-%Y') as retired_date,
    format(salary,2) as Salary,
    format(4 * salary, 2) as Retirement_payment
FROM EMPLOYEE;
END $$
DELIMITER;
```