4(b) Cursor to list the customer id, name, and address from the CUSTOMERS table where age>25 and determine the number of affected rows.

```
CREATE TABLE Customers(id INTEGER, name varchar(100), age INTEGER, address varchar(100), salary INTEGER);

insert into Customers VALUES(1, 'Ramesh', 32, 'Ahmedabad', 2500);

insert into Customers VALUES(2, 'Khilan', 25, 'Delhi',2000);

insert into Customers VALUES(3, 'Kaushik', 23,'Kota', 2500);

insert into Customers VALUES(4, 'Chaitali', 25,'Mumbai', 7000);

insert into Customers VALUES(5, 'Hardik', 27, 'Bhopal', 9000);

insert into Customers VALUES(6, 'Komal', 22, 'MP', 5000);


DELIMITER $$

CREATE  PROCEDURE  Average_age_Employee( INOUT info varchar(4000), INOUT count_no INTEGER )

BEGIN

        DECLARE  finished  INTEGER DEFAULT 0;

        DECLARE emp_id int ;

        DECLARE emp_name  varchar(50) DEFAULT '';

        DECLARE emp_addr  varchar(50) DEFAULT '';

         -- declare cursor for employee detail

        DEClARE  curdetail

                CURSOR FOR

                SELECT id, name, address FROM Customers where age>25;

        -- declare NOT FOUND handler

        DECLARE  CONTINUE  HANDLER FOR NOT FOUND SET finished = 1;

        OPEN curdetail;

        getdetail: LOOP

                FETCH curdetail INTO emp_id, emp_name, emp_addr;
```

```
            IF finished = 1 THEN
                    LEAVE getdetail;
            END IF;
            -- build detail list
            SET info = CONCAT(emp_id, ',', emp_name,  ',', emp_addr,';' ,info);
            SET count_no = count_no +1;
            END LOOP getdetail;
        CLOSE curdetail;
END$$
DELIMITER ;
SET  @employeedetail  = '' ;
SET @count_no = 0;
CALL Average_age_Employee (@employeedetail, @count_no );
SELECT  @employeedetail, @count_no;
```

```
@employeedetail @count_no
5,Hardik,Bhopal;1,Ramesh,Ahmedabad;     2
```

5 Create a MYSQL trigger for Product table that activates for BEFORE UPDATE event (Updating the price of product) to log the old price of a product in separate table named PriceLogs (Product Code, Price, Updated_at).

```
CREATE TABLE PriceLogs (
  productCode VARCHAR(15),
  price DECIMAL(10,2),
  updated_at TIMESTAMP NOT NULL
                    DEFAULT CURRENT_TIMESTAMP
          ON UPDATE CURRENT_TIMESTAMP);

DELIMITER $$

CREATE TRIGGER before_products_update
  BEFORE UPDATE ON products
  FOR EACH ROW
BEGIN
   IF OLD.Price <> NEW.Price THEN
      INSERT INTO PriceLogs(product_code,price)
      VALUES(old.productCode,old.Price);
   END IF;
```

```
    END$$
    DELIMITER ;
```