Homework 5

IVAN MARTINOVIC

Homework Assigned: Feb. 25, 2021
Homework Due: March 4, 2021
Submission: Submit via CANVAS. This work is to be completed INDIVIDUALLY. No help besides the textbook should be taken. *Copying any answers or part of answers from other sources (including your colleagues) will earn you a grade of zero.* All homework is to be typed, no handwritten diagrams or pictures of diagrams are accepted. Handwritten submissions will result in a 20% grade penalty.
Maximum Points: 100 pts

Please make sure to write your name inside the document, at the top.

You are given a database schema consisting of three relations, whose schemas are given below. You are expected to develop your own test data sets and to perform thorough testing on these data sets using Oracle. Also test your SQL statements out on the sample data that we will provide.

Product(model, manufacturer, type)
PC(model, speed, ram, hd, rd, price)
Laptop(model, speed, ram, hd, screen, price)

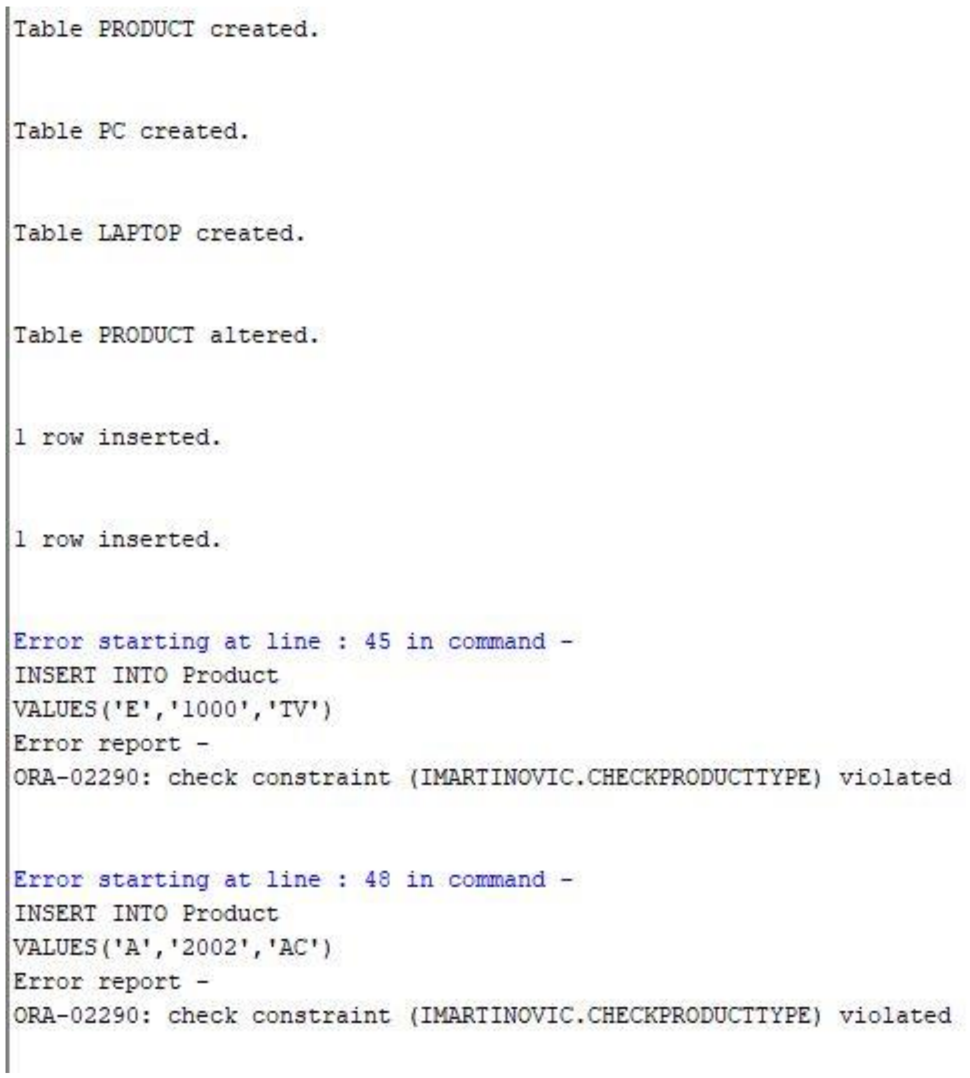
Problem 1. Constraints Specification in SQL [25 pts]

Assume no primary keys, nor foreign key constraints have been defined on this schema yet.
Write SQL DDL statements to add the constraints below whenever possible. Do not use triggers!
Please REVIEW the supplementary Integrity Constraints Slides Attached to this homework to learn about “Check” constraints.
Demonstrate that your solution works. If you cannot enforce the constraints through DDL statements explain why.

1. Add a constraint to the Product table to enforce that the type of the product must either be ’PC’ or ’Laptop’.

```
DROP TABLE PC;

DROP TABLE Laptop;

DROP TABLE Product;

CREATE TABLE Product(
        manufacturer CHAR(10),
      model CHAR(10),
        type CHAR(10)
);

CREATE TABLE PC(
        model CHAR(10),
	speed INT,
	ram INT,
	hd INT,
	rd CHAR(10),
	price INT
);


CREATE TABLE Laptop(
	model CHAR(10),
	speed INT,
	ram INT,
	hd INT,
	screen number(4,2),
	price INT

);

ALTER TABLE Product
ADD CONSTRAINT checkProductType CHECK (type = 'PC' OR type = 'Laptop');

/* TEST CASES */
/* These should pass */
INSERT INTO Product
VALUES('E','1000','PC');

INSERT INTO Product
VALUES('A','2002','Laptop');

/* These should fail */
INSERT INTO Product
VALUES('E','1000','TV');

INSERT INTO Product
VALUES('A','2002','AC');
```

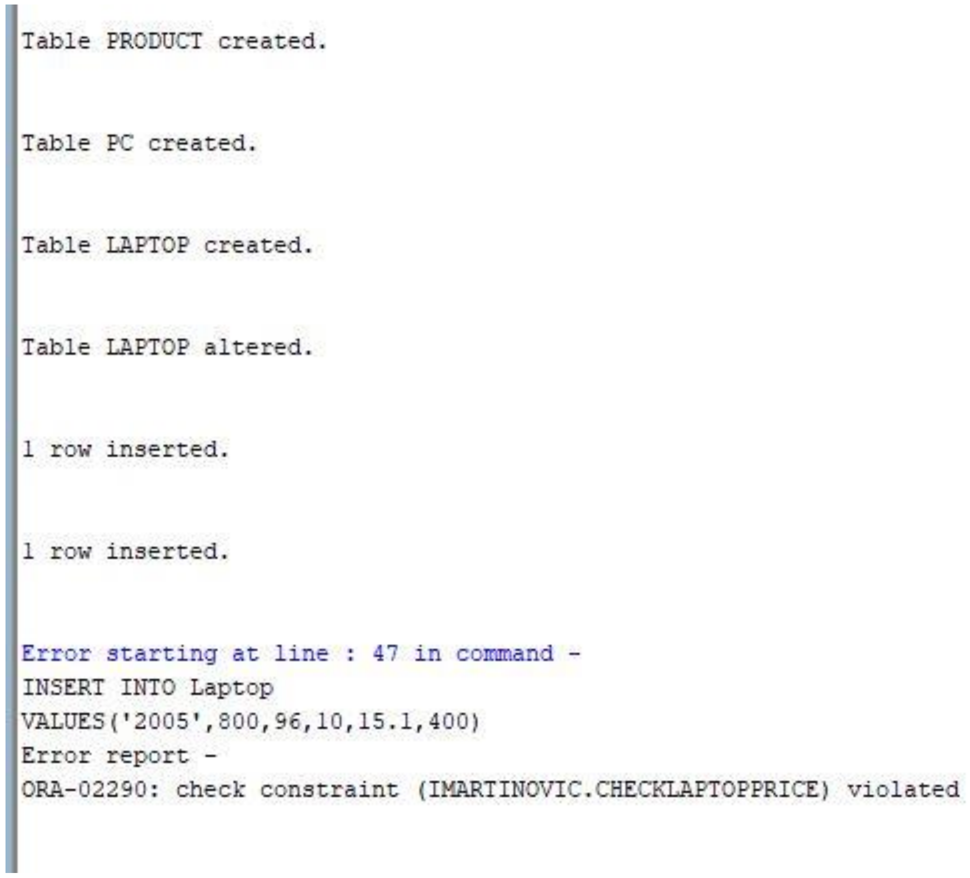
The Result of Running the SQL Script:


```
Table PRODUCT created.


Table PC created.


Table LAPTOP created.


Table PRODUCT altered.


1 row inserted.


1 row inserted.


Error starting at line : 45 in command -
INSERT INTO Product
VALUES('E','1000','TV')
Error report -
ORA-02290: check constraint (IMARTINOVIC.CHECKPRODUCTTYPE) violated


Error starting at line : 48 in command -
INSERT INTO Product
VALUES('A','2002','AC')
Error report -
ORA-02290: check constraint (IMARTINOVIC.CHECKPRODUCTTYPE) violated
```

2. Add an additional constraint that the price of any Laptop must be at least 500.

```
DROP TABLE PC;

DROP TABLE Laptop;

DROP TABLE Product;

CREATE TABLE Product(
        manufacturer CHAR(10),
    model CHAR(10),
        type CHAR(10)
);

CREATE TABLE PC(
        model CHAR(10),
            speed INT,
            ram INT,
            hd INT,
            rd CHAR(10),
            price INT
);


CREATE TABLE Laptop(
            model CHAR(10),
            speed INT,
            ram INT,
            hd INT,
            screen number(4,2),
            price INT

);

ALTER TABLE Laptop
ADD CONSTRAINT checkLaptopPrice CHECK (price >= 500);


/* TEST CASES */
/* These should pass */
INSERT INTO Laptop
VALUES('2002',700,64,5,12.1,1000);

INSERT INTO Laptop
VALUES('2004',850,64,10,15.1,500);


/* These should fail */
INSERT INTO Laptop
VALUES('2005',800,96,10,15.1,400);
```

The Result of Running the SQL Script:



```
Table PRODUCT created.


Table PC created.


Table LAPTOP created.


Table LAPTOP altered.


1 row inserted.


1 row inserted.


Error starting at line : 47 in command -
INSERT INTO Laptop
VALUES('2005',800,96,10,15.1,400)
Error report -
ORA-02290: check constraint (IMARTINOVIC.CHECKLAPTOPPRICE) violated
```

3. Add the constraint that a laptop with a larger model number must also have a higher price than one with a lower model number.

   We could attempt to solve the problem using a CHECK Constraint which uses a nested SELECT statement which ORACLE does not allow.

   It would look something like:

   ALTER TABLE Laptop
   ADD CONSTRAINT checkLaptopPriceModel CHECK (price > (SELECT MAX(price) FROM Laptop WHERE model < referenceModel));

   NOTE: Using such an approach doesn't allow you to specify the exact referenceModel the models should be smaller than inside the nested select statement

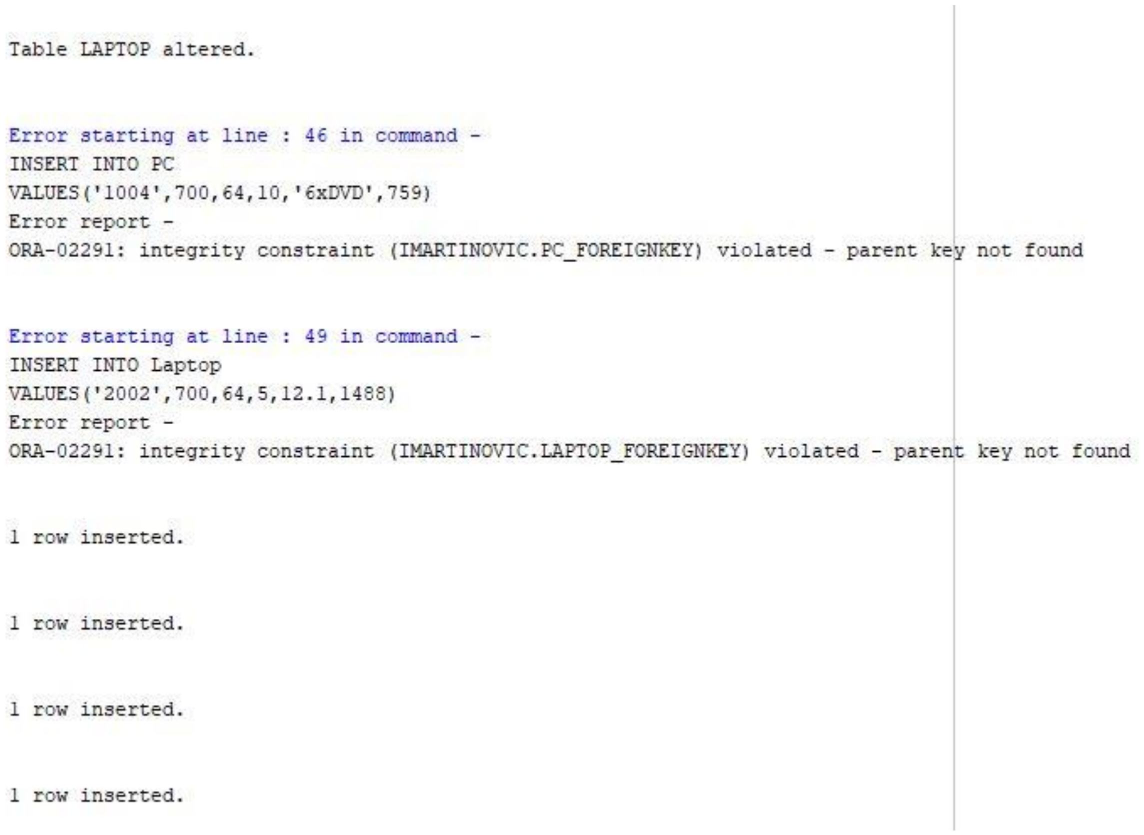
   On the other hand we could easily enforce the constraint using triggers.

4. Add the constraint that any PC and Laptop corresponds to a model number that also exists in the Product table.

```
DROP TABLE PC;

DROP TABLE Laptop;

DROP TABLE Product;

CREATE TABLE Product(
            manufacturer CHAR(10),
      model CHAR(10),
            type CHAR(10)
);

CREATE TABLE PC(
            model CHAR(10),
            speed INT,
            ram INT,
            hd INT,
            rd CHAR(10),
            price INT
);


CREATE TABLE Laptop(
            model CHAR(10),
            speed INT,
            ram INT,
            hd INT,
            screen number(4,2),
            price INT

);

ALTER TABLE Product
ADD CONSTRAINT Product_PrimaryKey PRIMARY KEY (model);

ALTER TABLE PC
ADD CONSTRAINT PC_ForeignKey FOREIGN KEY (model) REFERENCES PRODUCT (model);

ALTER TABLE Laptop
ADD CONSTRAINT Laptop_ForeignKey FOREIGN KEY (model) REFERENCES PRODUCT (model);



/* TEST CASES */
/* These should fail */
INSERT INTO PC
VALUES('1004',700,64,10,'6xDVD',759);
```

```
INSERT INTO Laptop
VALUES('2002',700,64,5,12.1,1488);


/* These should pass */

INSERT INTO Product
VALUES('C','1004','PC');

INSERT INTO Product
VALUES('A','2002','Laptop');

INSERT INTO PC
VALUES('1004',700,64,10,'6xDVD',759);

INSERT INTO Laptop
VALUES('2002',700,64,5,12.1,1488);
```

The Result of Running the SQL Script:



```
Table LAPTOP altered.


Error starting at line : 46 in command -
INSERT INTO PC
VALUES('1004',700,64,10,'6xDVD',759)
Error report -
ORA-02291: integrity constraint (IMARTINOVIC.PC_FOREIGNKEY) violated - parent key not found


Error starting at line : 49 in command -
INSERT INTO Laptop
VALUES('2002',700,64,5,12.1,1488)
Error report -
ORA-02291: integrity constraint (IMARTINOVIC.LAPTOP_FOREIGNKEY) violated - parent key not found


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

5. Add the constraint that in our product database table we only maintain products from at most 5 different manufacturers (so to assure that the quality we offer is high).

To add such constraint we need to write a nested SQL querry inside the CHECK statement which is not supported by Oracle. The CHECK statement would look like the following:

```
ALTER TABLE Product
ADD CONSTRAINT manCountCheck CHECK (
(SELECT COUNT (Manufacturer) FROM (SELECT DISTINCT Manufacturer FROM Product)) <= 5);
```

Problem 2. Trigger Specification in SQL [50 pts]

Consider the relational schema below with primary key constraints as specified. However, the designer forgot to define any foreign keys ☺

Product(model PRIMARY KEY, manufacturer, type)
PC(model PRIMARY KEY, speed, ram, hd, rd, price)
Laptop(model PRIMARY KEY, speed, ram, hd, screen, price)

Now determine how many triggers you need, what events you need to monitor, and lastly what actions are the most meaningful to take for each of the examples below. Test out your triggers, once designed, on the data set that we provide.
Turn in your script illustrating the success of your tests.

1. Write one or more triggers to enforce overlap constraints, namely, to specify that when inserting a new laptop, the model number should not also appear in the PC table, and vice versa.

```
DROP TABLE PC;

DROP TABLE Laptop;

DROP TABLE Product;

CREATE TABLE Product(
        manufacturer CHAR(10),
    model CHAR(10) PRIMARY KEY,
        type CHAR(10)
);

CREATE TABLE PC(
        model CHAR(10) PRIMARY KEY,
        speed INT,
        ram INT,
        hd INT,
        rd CHAR(10),
        price INT
);


CREATE TABLE Laptop(
        model CHAR(10) PRIMARY KEY,
        speed INT,
        ram INT,
        hd INT,
        screen number(4,2),
        price INT

);


        CREATE OR REPLACE TRIGGER overlapTriggerPC
           BEFORE INSERT OR UPDATE OF model
           ON PC
           FOR EACH ROW

           DECLARE
           countM NUMBER;
            BEGIN
              SELECT COUNT (model) INTO countM FROM Laptop WHERE model = :new.model;
              IF(countM = 1 ) THEN
                 raise_application_error(-20001, 'Foreign Key Trigger');
              END IF;
            END;
        /

        CREATE OR REPLACE TRIGGER overlapTriggerLaptop
           BEFORE INSERT OR UPDATE OF model
           ON Laptop
           FOR EACH ROW

           DECLARE
           countM NUMBER;
            BEGIN
              SELECT COUNT (model) INTO countM FROM PC WHERE model = :new.model;
              IF(countM = 1 ) THEN
                 raise_application_error(-20001, 'Foreign Key Trigger');
              END IF;
            END;
        /

        /* TEST CASES  */
        INSERT INTO PC
```

```
/* These should pass */
VALUES('1000',700,64,10,'48xCD',799);

INSERT INTO Laptop
VALUES('2002',700,64,5,12.1,1488);

/* These should fail */
INSERT INTO PC
VALUES('2002',700,64,10,'48xCD',799);

INSERT INTO Laptop
VALUES('1000',700,64,5,12.1,1488);
```

2. Write one or more triggers to specify that for any tuple in the PC table the hard disk of the PC is at least 100 times greater than its RAM. (Note that the hard disk is in GB, while RAM is in MB).

```
DROP TABLE PC;

DROP TABLE Laptop;

DROP TABLE Product;

CREATE TABLE Product(
        manufacturer CHAR(10),
    model CHAR(10) PRIMARY KEY,
        type CHAR(10)
);

CREATE TABLE PC(
        model CHAR(10) PRIMARY KEY,
        speed INT,
        ram INT,
        hd INT,
        rd CHAR(10),
        price INT
);


CREATE TABLE Laptop(
        model CHAR(10) PRIMARY KEY,
        speed INT,
        ram INT,
        hd INT,
        screen number(4,2),
        price INT

);
```

```
CREATE OR REPLACE TRIGGER ramToDiskRatioTrigger
   BEFORE INSERT OR UPDATE
   ON PC
   FOR EACH ROW

    BEGIN
      IF(:new.hd < 0.1 * :new.RAM) THEN
         raise_application_error(-20001, 'HardDisk To RAM Trigger');
      END IF;
    END;
/

/* TEST CASES */
/* These should PASS */
INSERT INTO PC
VALUES('1000',700, 100, 100,'48xCD',799);

INSERT INTO PC
VALUES('1001',700,100,10,'48xCD',799);

UPDATE PC SET hd = 100, ram = 10 WHERE model = 1000;

/* These should FAIL */

INSERT INTO PC
VALUES('1002',700, 100, 9,'48xCD',799);

UPDATE PC SET hd = 9, ram = 100 WHERE model = 1000;
```

3. Write the needed triggers to enforce that whenever the prices of a Product model are being modified, that then there is a “log tuple” inserted into a special relational table (call it Product-Monitoring) that indicates the model number of the modified product listing, the type of the product (pc, printer, etc.), the old price, and the new price, the time of the modification. Note that a command such as “to char(sysdate,’dd-mm-yyyy: hh24:mi’)” could be used to produce a date value. Also, remember that to first create the product-Monitoring table with the appropriate attributes.

```
DROP TABLE PC;

DROP TABLE Laptop;

DROP TABLE Product;

DROP TABLE Monitor;

CREATE TABLE Product(
            manufacturer CHAR(10),
       model CHAR(10) PRIMARY KEY,
            type CHAR(10)
);

CREATE TABLE PC(
            model CHAR(10) PRIMARY KEY,
            speed INT,
            ram INT,
            hd INT,
            rd CHAR(10),
            price INT
);


CREATE TABLE Laptop(
            model CHAR(10) PRIMARY KEY,
            speed INT,
            ram INT,
            hd INT,
            screen number(4,2),
            price INT

);

CREATE TABLE Monitor (
   model CHAR(10),
   Type CHAR(10),
   oldPrice INT,
   newPrice INT,
   logDate CHAR(20),
   PRIMARY KEY (model, logDate)
);

CREATE OR REPLACE TRIGGER priceChangeLogPCTrigger
   BEFORE UPDATE OF price
   ON PC
   FOR EACH ROW

   DECLARE

    BEGIN
      INSERT INTO Monitor(model, type, oldPrice, newPrice, logDate) VALUES (:new.model, 'PC',
:old.price, :new.price, TO_CHAR(sysdate, 'dd-mm-yyyy: hh24:mi'));
    END;
/

CREATE OR REPLACE TRIGGER priceChangeLogLaptopTrigger
   BEFORE UPDATE OF price
   ON Laptop
   FOR EACH ROW

   DECLARE

    BEGIN
       INSERT INTO Monitor(model, type, oldPrice, newPrice, logDate) VALUES (:new.model, 'Laptop',
:old.price, :new.price, TO_CHAR(sysdate, 'dd-mm-yyyy: hh24:mi:ss'));
    END;
/

INSERT INTO PC
VALUES('1004',700,64,10,'6xDVD',759);

INSERT INTO Laptop
VALUES('2002',700,64,5,12.1,1488);

/*these changes shouldn't appear*/
UPDATE PC SET hd = 9 WHERE model = 1004;
UPDATE Laptop SET ram = 9 WHERE model = 2002;

/* these changes should appear*/
UPDATE PC SET price = 9 WHERE model = 1004;
UPDATE Laptop SET price = 9  WHERE model = 2002;

SELECT COUNT(model) FROM Monitor; /* Should return 2 */
```

4. Write one or more triggers to enforce the constraint that at all times the Product table is consistent with the other two tables. This is an extension of the foreign key constraint semantics. So now assume here that you did not have access in your DBMS to any direct support for foreign keys. That is, if in the Product table, a product row is specified as being of PC type then its model number also appears in the corresponding PC table. Similarly, if a product is of type laptop, then its model number must also appear in the laptop table. If the type VALUE is "NULL", then it should appear in none of the other tables. Or, vice-versa, check that any tuple that is being inserted into the Laptop or the PC table, either already exists in the Product table, or if not then you also will add it into the Product table as part of the current update.

```
DROP TABLE PC;

DROP TABLE Laptop;

DROP TABLE Product;

CREATE TABLE Product(
            manufacturer CHAR(10),
      model CHAR(10) PRIMARY KEY,
            type CHAR(10)
);

CREATE TABLE PC(
            model CHAR(10) PRIMARY KEY,
            speed INT,
            ram INT,
            hd INT,
            rd CHAR(10),
            price INT
);


CREATE TABLE Laptop(
            model CHAR(10) PRIMARY KEY,
            speed INT,
            ram INT,
            hd INT,
            screen number(4,2),
            price INT

);

CREATE OR REPLACE TRIGGER foreignKeyProductUpdate
   BEFORE UPDATE OF type
   ON Product
   FOR EACH ROW

   DECLARE
   countM NUMBER;
    BEGIN
    IF (:old.type <> :new.type) THEN
        IF (:old.type = 'PC' ) THEN
           DELETE FROM PC WHERE model = :old.model;
       END IF;
       IF (:old.type = 'Laptop' ) THEN
           DELETE FROM PC WHERE model = :old.model;
       END IF;
    END IF;
    END;
/
```

```
CREATE OR REPLACE TRIGGER foreignKeyProductDelete
   BEFORE DELETE
   ON Product
   FOR EACH ROW

   DECLARE
   countM NUMBER;
    BEGIN
       IF (:old.type = 'PC' ) THEN
          DELETE FROM PC WHERE model = :old.model;
      END IF;
      IF (:old.type = 'Laptop' ) THEN
          DELETE FROM Laptop WHERE model = :old.model;
      END IF;
    END;
/


CREATE OR REPLACE TRIGGER foreignKeyPC
   BEFORE INSERT OR UPDATE OF model
   ON PC
   FOR EACH ROW

   DECLARE
   countM NUMBER;
   BEGIN
      SELECT COUNT (model) INTO countM FROM Product WHERE model = :new.model AND type = 'PC';
      IF(countM = 0) THEN
         raise_application_error(-20001, 'Foreign Key Trigger: No such model exists in Product Table');
      END IF;

    END;
/

CREATE OR REPLACE TRIGGER foreignKeyLaptop
   BEFORE INSERT OR UPDATE OF model
   ON Laptop
   FOR EACH ROW

   DECLARE
   countM NUMBER;
   BEGIN
      SELECT COUNT (model) INTO countM FROM Product WHERE model = :new.model AND type =
'Laptop';
      IF(countM = 0) THEN
         raise_application_error(-20001, 'Foreign Key Trigger: No such model exists in Product Table');
      END IF;

   END;
/
/* Inserts in the PC or Laptop Table without a corresponding insert in the Product Table is illegal */
INSERT INTO PC
VALUES('1004',700,64,10,'6xDVD',759);

INSERT INTO Laptop
VALUES('2002',700,64,5,12.1,1488);
/* If a model of type PC exists, we shouldn't be allowed to add a laptop with the same model and vice-
versa*/
INSERT INTO Product
VALUES('A','2002','Laptop');

INSERT INTO Product
VALUES('C','1004','PC');

INSERT INTO PC
VALUES('2002',700,64,10,'6xDVD',759);

INSERT INTO Laptop
VALUES('1004',700,64,5,12.1,1488);

/* Inserts first in the Product Table then in either of the two other Tables Should Work */
INSERT INTO PC
VALUES('1004',700,64,10,'6xDVD',759);

INSERT INTO Laptop
VALUES('2002',700,64,5,12.1,1488);

/* Changing the type inside the Product Table should delete the corresponding model from the PC or
Laptop Tables (only if the OLD and NEW types differ */
UPDATE Product SET type = 'PC' WHERE model = 1004; /* Should do nothing */
SELECT COUNT(model) FROM PC WHERE model = 1004;   /* Should return 1*/
UPDATE Product SET type = 'Laptop' WHERE model = 1004; /* Should erase entry from PC table */
SELECT COUNT(model) FROM PC WHERE model = 1004;   /* Should return 0*/
```

/* Removing a record from the Product Table should also remove the records from the two other tables */
DELETE FROM Product WHERE model = 2002;
SELECT COUNT(model) FROM Laptop WHERE model = 2002;   /* Should return 0*/

/* Note: The NULL Case doesn't have to be tested, since an insert in the PC or Laptop tables would be disallowed if the type is NULL*/
/* If we attempt to alter the type of an entry to NULL, it will automatically delete the records from PC or Laptop tables*/

Problem 3: View Specification in SQL [25 pts]

Consider a database schema consisting of three relations, whose schemas are given below (the key fields are underlined). Please load data into your tables and test your answers on Oracle.

Product(model, manufacturer, type)

PC(model, speed, ram, hd, rd, price)

Laptop(model, speed, ram, hd, screen, price)

Consider the following view PCPriceList defined as:

CREATE VIEW PCPriceList AS Select model, price from PC

1. Use the above view to find the PC with the cheapest price. Show the SQL query.

   Can you delete from this view directly? Discuss. Show how!

DROP TABLE PC;

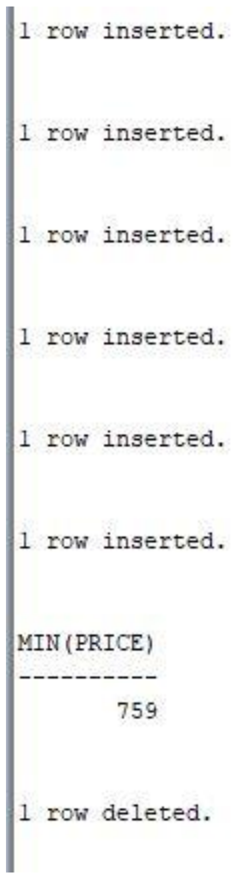
DROP TABLE Laptop;

DROP TABLE Product;

DROP VIEW PCPriceList;

```
CREATE TABLE Product(
        manufacturer CHAR(10),
    model CHAR(10) PRIMARY KEY,
        type CHAR(10)
);

CREATE TABLE PC(
        model CHAR(10) PRIMARY KEY,
        speed INT,
        ram INT,
        hd INT,
        rd CHAR(10),
        price INT,
    FOREIGN KEY (model) REFERENCES Product(model)
);


CREATE TABLE Laptop(
        model CHAR(10) PRIMARY KEY,
        speed INT,
        ram INT,
        hd INT,
        screen number(4,2),
        price INT,
    FOREIGN KEY (model) REFERENCES Product(model)
);

CREATE VIEW PCPriceList  AS Select model, price from PC;
```

```
INSERT INTO Product

VALUES('E','1000','PC');

INSERT INTO PC

VALUES('1000',700,64,10,'48xCD',799);


INSERT INTO Product

VALUES('B','1002','PC');

INSERT INTO PC

VALUES('1002',1500,128,60,'12xDVD',2499);


INSERT INTO Product

VALUES('B','1003','PC');

INSERT INTO PC

VALUES('1003',700,64,10,'8xDVD',899);


INSERT INTO Product

VALUES('C','1004','PC');

INSERT INTO PC

VALUES('1004',700,64,10,'6xDVD',759);


/* SELECE THE MINIMUM PRICE */

SELECT MIN(price) FROM PCPriceList;  /* should return 759 */

/* DELETING IS POSSIBLE (the view references only one Table)*/

DELETE FROM PCPriceList WHERE model = '1004';
```


```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


MIN(PRICE)
----------
       759


1 row deleted.
```

2. Can you perform an insert such as: INSERT INTO PCPriceList(model) VALUES (2005)? When yes and when no? Discuss.

DROP TABLE PC;

DROP TABLE Laptop;

DROP TABLE Product;

DROP VIEW PCPriceList;

CREATE TABLE Product(

    manufacturer CHAR(10),

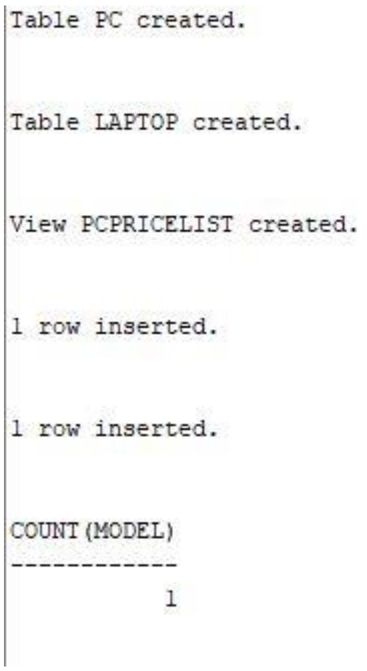
```
    model CHAR(10) PRIMARY KEY,
        type CHAR(10)
);

CREATE TABLE PC(
        model CHAR(10) PRIMARY KEY,
        speed INT,
        ram INT,
        hd INT,
        rd CHAR(10),
        price INT,
    FOREIGN KEY (model) REFERENCES Product(model)
);



CREATE TABLE Laptop(
        model CHAR(10) PRIMARY KEY,
        speed INT,
        ram INT,
        hd INT,
        screen number(4,2),
        price INT,
    FOREIGN KEY (model) REFERENCES Product(model)
);

CREATE VIEW PCPriceList  AS Select model, price from PC;

/* INSERTING IS ALSO POSSIBLE SINCE Primary Key is included (needs to be first inserted in the Product
Table)*/
```

INSERT INTO Product(model) VALUES (2005);

INSERT INTO PCPriceList(model) VALUES (2005);

/* Should return 1 */

SELECT COUNT(model) FROM PC WHERE model = '2005';



```
Table PC created.


Table LAPTOP created.


View PCPRICELIST created.


1 row inserted.


1 row inserted.


COUNT(MODEL)
------------
           1
```

3. What about an insert such as: INSERT INTO PCPriceList (price) VALUES (1700)? Discuss. Show what happens.

DROP TABLE PC;

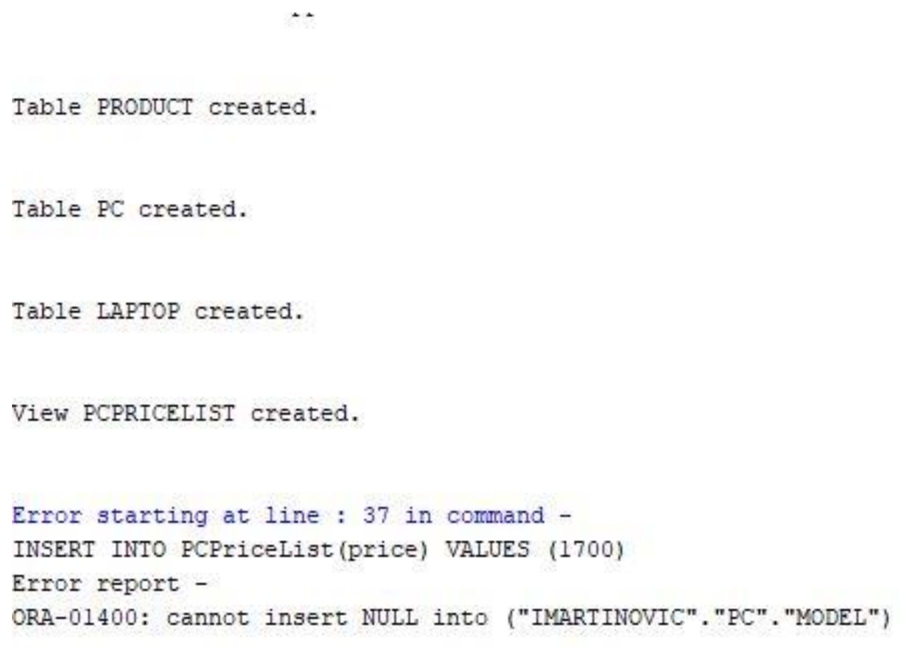
DROP TABLE Laptop;

DROP TABLE Product;

DROP VIEW PCPriceList;

```
CREATE TABLE Product(
            manufacturer CHAR(10),
      model CHAR(10) PRIMARY KEY,
            type CHAR(10)
);

CREATE TABLE PC(
            model CHAR(10) PRIMARY KEY,
            speed INT,
            ram INT,
            hd INT,
            rd CHAR(10),
            price INT,
      FOREIGN KEY (model) REFERENCES Product(model)
);



CREATE TABLE Laptop(
            model CHAR(10) PRIMARY KEY,
            speed INT,
            ram INT,
            hd INT,
            screen number(4,2),
            price INT,
      FOREIGN KEY (model) REFERENCES Product(model)
);
CREATE VIEW PCPriceList  AS Select model, price from PC;
/* INSERTING IS NOT POSSIBLE SINCE Primary Key is NOT included (i.e. trying to INSERT NULL as Primary
Key */
INSERT INTO PCPriceList(price) VALUES (1700);
```

<u>RESULT OF INSERT STATEMENT:</u>



```
Table PRODUCT created.


Table PC created.


Table LAPTOP created.


View PCPRICELIST created.


Error starting at line : 37 in command -
INSERT INTO PCPriceList(price) VALUES (1700)
Error report -
ORA-01400: cannot insert NULL into ("IMARTINOVIC"."PC"."MODEL")
```

---

4. Now using SQL DDL, define a second view extendedPC(manufacturer, model, speed, ram, hd, rd, price, type). This view will give every PC made by each manufacturer. Can you delete from this extendedPC view? Discuss. Show what happens.

```
DROP TABLE PC;

DROP TABLE Laptop;

DROP TABLE Product;

DROP VIEW PCPriceList;

DROP VIEW extendedPC;

CREATE TABLE Product(

            manufacturer CHAR(10),

      model CHAR(10) PRIMARY KEY,

            type CHAR(10)

);

CREATE TABLE PC(
            model CHAR(10) PRIMARY KEY,

            speed INT,

            ram INT,

            hd INT,

            rd CHAR(10),

            price INT,

      FOREIGN KEY (model) REFERENCES Product(model)

);

CREATE TABLE Laptop(

            model CHAR(10) PRIMARY KEY,

            speed INT,

            ram INT,

            hd INT,

            screen number(4,2),

            price INT,

      FOREIGN KEY (model) REFERENCES Product(model)

);

CREATE VIEW PCPriceList  AS Select model, price from PC;

CREATE VIEW extendedPC AS

SELECT Pr.manufacturer, PC.model, PC.speed, PC.RAM, PC.hd, PC.price, Pr.type

FROM Product Pr, PC

WHERE Pr.model = PC.model;


INSERT INTO Product

VALUES('E','1000','PC');
```

INSERT INTO PC

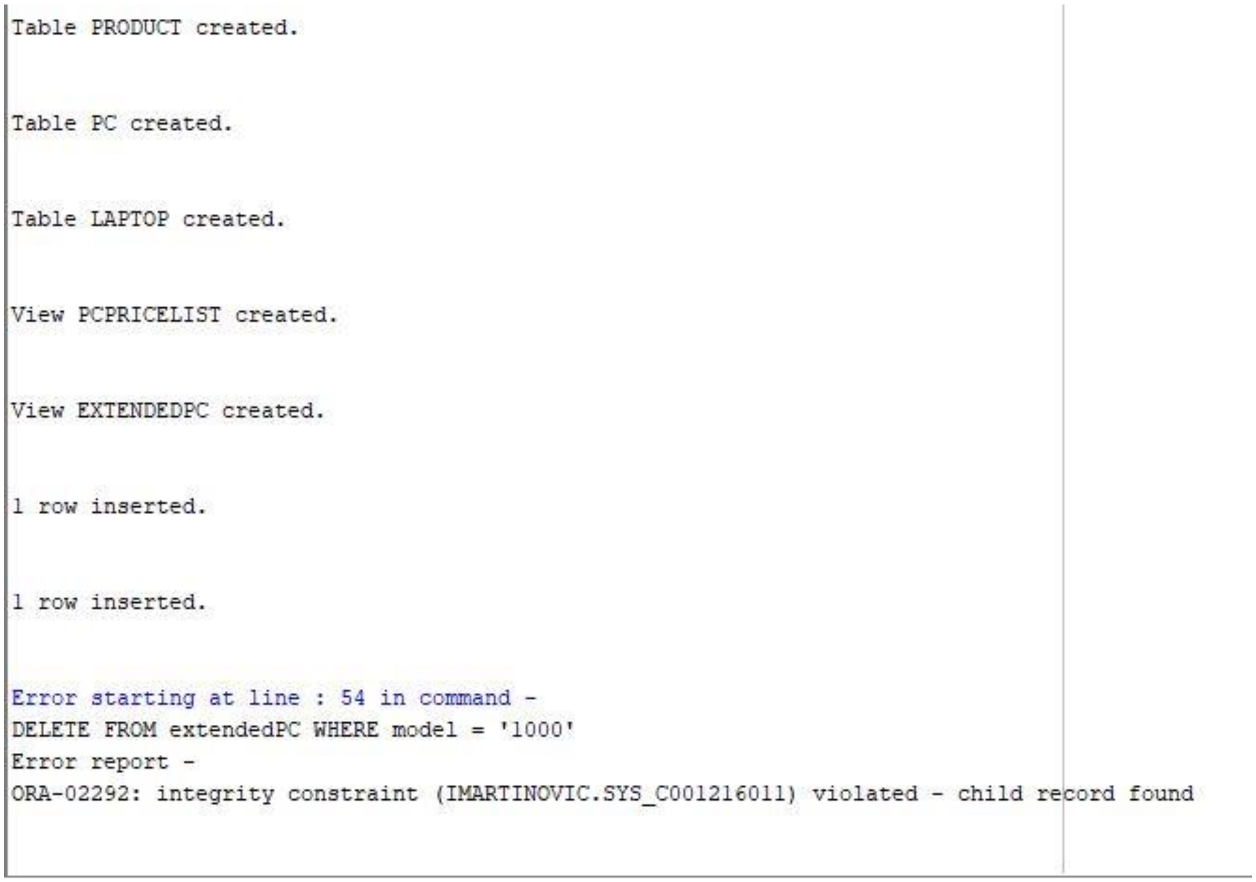
VALUES('1000',700,64,10,'48xCD',799);

/* DELETE doesn’t work because of the Foreign Key Constraint. Oracle will throw a integrity constraint violation: Child record found (this persists even if we define the extendedPC to select Pr.model)

NOTE: If there are no foreign keys, only the entry in the PC table will be deleted !!!

*/

DELETE FROM extendedPC WHERE model = '1000';



```
Table PRODUCT created.


Table PC created.


Table LAPTOP created.


View PCPRICELIST created.


View EXTENDEDPC created.


1 row inserted.


1 row inserted.


Error starting at line : 54 in command -
DELETE FROM extendedPC WHERE model = '1000'
Error report -
ORA-02292: integrity constraint (IMARTINOVIC.SYS_C001216011) violated - child record found
```

Deliverables:

Students should submit a .pdf file containing their appropriately numbered SQL statements and other responses to questions. The file MUST be a .pdf file (other file types need to be converted to .pdf for submission).

Submission:

Submit via Canvas. Late submissions will not be accepted.

Submission notes:

1) Include your name on the sheet
2) Be sure that your submission is in .pdf format.
3) Handwritten solutions or pictures of handwritten solutions will not be accepted.