Database Lab Take Home Assignment - 5

Q1. Construct a table with the following details given below:
Inventory (PID, PNAME, QUANTITY, PRICE)
Product (ProdID, PNAME, QUANTITY, STATUS, LOG)
Purchase (ProductID, BNAME, PNAME, QUANTITY, PRICE, DATE)
PID: Primary key for the table Inventory.
ProdID: Primary key for the table Product.
ProductID, BNAME: Primary keys for the table Purchase.

Database Creation

```
CREATE DATABASE 214161008_05;
```

```
mysql> CREATE DATABASE 214161008_05;
Query OK, 1 row affected (0.26 sec)
```

Select Database

```
USE 214161008_05;
```

```
mysql> USE 214161008_05;
Database changed
mysql>
```

Inventory table creation

```
CREATE TABLE INVENTORY
(
   PRODUCT_ID INTEGER,
   PRODUCT_NAME VARCHAR(20),
   QUANTITY INTEGER,
   PRICE FLOAT,
   PRIMARY KEY (PRODUCT_ID)
);
```

```
mysql> CREATE TABLE INVENTORY
    -> (
    ->     PRODUCT_ID INTEGER,
    ->     PRODUCT_NAME VARCHAR(20),
    ->     QUANTITY INTEGER,
    ->     PRICE FLOAT,
    ->     PRIMARY KEY (PRODUCT_ID)
    -> );
Query OK, 0 rows affected (2.05 sec)

mysql>
```

Product Table creation

```
CREATE TABLE PRODUCT
(
   PRODUCT_ID INTEGER,
   PRODUCT_NAME VARCHAR(20),
   QUANTITY INTEGER,
   STATUS VARCHAR(20),
   PRODUCT_LOG DATETIME,
   PRIMARY KEY (PRODUCT_ID)
);
```

```
mysql> CREATE TABLE PRODUCT
    -> (
    ->     PRODUCT_ID INTEGER,
    ->     PRODUCT_NAME VARCHAR(20),
    ->     QUANTITY INTEGER,
    ->     STATUS VARCHAR(20),
    ->     PRODUCT_LOG DATETIME,
    ->     PRIMARY KEY (PRODUCT_ID)
    -> );
Query OK, 0 rows affected (0.83 sec)

mysql>
```

Purchase Table Creation

```
CREATE TABLE PURCHASE
(
   PRODUCT_ID INTEGER,
   BRAND_NAME VARCHAR(20),
   PRODUCT_NAME VARCHAR(20),
   QUANTITY INTEGER,
```

```
   PRICE FLOAT,
   PURCHASE_DATE DATE,
   PRIMARY KEY (PRODUCT_ID, BRAND_NAME)
);
```

```
mysql> CREATE TABLE PURCHASE
    -> (
    ->     PRODUCT_ID INTEGER,
    ->     BRAND_NAME VARCHAR(20),
    ->     PRODUCT_NAME VARCHAR(20),
    ->     QUANTITY INTEGER,
    ->     PRICE FLOAT,
    ->     PURCHASE_DATE DATE,
    ->     PRIMARY KEY (PRODUCT_ID, BRAND_NAME)
    -> );

Query OK, 0 rows affected (1.90 sec)

mysql>
mysql>
```

I. Write a before insert and before delete trigger on Inventory table to log the products in the Product.

Before insert trigger

```
DELIMITER //
CREATE TRIGGER NEW_INTO_INVENTORY BEFORE INSERT ON INVENTORY
FOR EACH ROW
BEGIN
   IF (SELECT count(*) FROM PRODUCT WHERE PRODUCT_ID = NEW.PRODUCT_ID) = 0 THEN
       INSERT INTO PRODUCT VALUES (NEW.PRODUCT_ID, NEW.PRODUCT_NAME,
NEW.QUANTITY, 'AVAILABLE', NOW());
   ELSE
       UPDATE PRODUCT
       SET QUANTITY = NEW.QUANTITY, STATUS = 'AVAILABLE', PRODUCT_LOG = NOW()
       WHERE PRODUCT_ID = NEW.PRODUCT_ID;
   END IF;
END;
//
DELIMITER ;
```

```
mysql> DELIMITER //
mysql> CREATE TRIGGER NEW_INTO_INVENTORY BEFORE INSERT ON INVENTORY
    -> FOR EACH ROW
    -> BEGIN
    ->     IF (SELECT count(*) FROM PRODUCT WHERE PRODUCT_ID = NEW.PRODUCT_ID) = 0 THEN
    ->         INSERT INTO PRODUCT VALUES (NEW.PRODUCT_ID, NEW.PRODUCT_NAME, NEW.QUANTITY, 'AVAILABLE', NOW());
    ->     ELSE
    ->         UPDATE PRODUCT
    ->         SET QUANTITY = NEW.QUANTITY, STATUS = 'AVAILABLE', PRODUCT_LOG = NOW()
    ->         WHERE PRODUCT_ID = NEW.PRODUCT_ID;
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.21 sec)

mysql> DELIMITER ;
mysql>
```

Inserting Data into table

```sql
INSERT INTO INVENTORY VALUES (1, 'SMARTPHONE', 3, 750);
INSERT INTO INVENTORY VALUES (3, 'SPEAKER', 2, 220);
INSERT INTO INVENTORY VALUES (2, 'SMARTWATCH', 4, 540);
INSERT INTO INVENTORY VALUES (4, 'MIC', 8, 100);
INSERT INTO INVENTORY VALUES (5, 'TABLETS', 6, 680);
INSERT INTO INVENTORY VALUES (6, 'KEYBOARD', 10, 140);
INSERT INTO INVENTORY VALUES (7, 'LAPTOP', 12, 1025);
```

```
mysql> INSERT INTO INVENTORY VALUES (1, 'SMARTPHONE', 3, 750);
Query OK, 1 row affected (0.60 sec)

mysql> INSERT INTO INVENTORY VALUES (3, 'SPEAKER', 2, 220);
Query OK, 1 row affected (0.19 sec)

mysql> INSERT INTO INVENTORY VALUES (2, 'SMARTWATCH', 4, 540);
Query OK, 1 row affected (0.18 sec)

mysql> INSERT INTO INVENTORY VALUES (4, 'MIC', 8, 100);
Query OK, 1 row affected (0.21 sec)

mysql> INSERT INTO INVENTORY VALUES (5, 'TABLETS', 6, 680);
Query OK, 1 row affected (0.27 sec)

mysql> INSERT INTO INVENTORY VALUES (6, 'KEYBOARD', 10, 140);
Query OK, 1 row affected (0.17 sec)

mysql> INSERT INTO INVENTORY VALUES (7, 'LAPTOP', 12, 1025);
Query OK, 1 row affected (0.22 sec)

mysql>
```

Updated table data in inventory

```
SELECT * FROM INVENTORY;
```

```
mysql> SELECT * FROM INVENTORY;
+------------+--------------+----------+-------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | PRICE |
+------------+--------------+----------+-------+
|          1 | SMARTPHONE   |        3 |   750 |
|          2 | SMARTWATCH   |        4 |   540 |
|          3 | SPEAKER      |        2 |   220 |
|          4 | MIC          |        8 |   100 |
|          5 | TABLETS      |        6 |   680 |
|          6 | KEYBOARD     |       10 |   140 |
|          7 | LAPTOP       |       12 |  1025 |
+------------+--------------+----------+-------+
7 rows in set (0.01 sec)

mysql>
```

Updated table data in Product

```
SELECT * FROM PRODUCT;
```

```
mysql> SELECT * FROM PRODUCT;
+------------+--------------+----------+-----------+---------------------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | STATUS    | PRODUCT_LOG         |
+------------+--------------+----------+-----------+---------------------+
|          1 | SMARTPHONE   |        3 | AVAILABLE | 2021-11-18 16:49:28 |
|          2 | SMARTWATCH   |        4 | AVAILABLE | 2021-11-18 16:49:29 |
|          3 | SPEAKER      |        2 | AVAILABLE | 2021-11-18 16:49:29 |
|          4 | MIC          |        8 | AVAILABLE | 2021-11-18 16:49:29 |
|          5 | TABLETS      |        6 | AVAILABLE | 2021-11-18 16:49:29 |
|          6 | KEYBOARD     |       10 | AVAILABLE | 2021-11-18 16:49:29 |
|          7 | LAPTOP       |       12 | AVAILABLE | 2021-11-18 16:49:30 |
+------------+--------------+----------+-----------+---------------------+
7 rows in set (0.00 sec)

mysql>
```

Before delete Trigger

```
DELIMITER //
CREATE TRIGGER REMOVE_FROM_INVENTORY BEFORE DELETE ON INVENTORY
FOR EACH ROW
BEGIN
   UPDATE PRODUCT
   SET QUANTITY = 0, STATUS = 'NOT AVAILABLE', PRODUCT_LOG = NOW()
   WHERE PRODUCT_ID = OLD.PRODUCT_ID;
END;
//
DELIMITER ;
```

```
mysql> DELIMITER //
mysql> CREATE TRIGGER REMOVE_FROM_INVENTORY BEFORE DELETE ON INVENTORY
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE PRODUCT
    ->     SET QUANTITY = 0, STATUS = 'NOT AVAILABLE', PRODUCT_LOG = NOW()
    ->     WHERE PRODUCT_ID = OLD.PRODUCT_ID;
    -> END;
    -> //
Query OK, 0 rows affected (0.24 sec)

mysql> DELIMITER ;
mysql>
```

Delete product having product_id = 1

```
DELETE FROM INVENTORY WHERE PRODUCT_ID = 1 ;
```

```
mysql> DELETE FROM INVENTORY WHERE PRODUCT_ID = 1 ;
Query OK, 1 row affected (0.19 sec)

mysql>
```

Updated table data in inventory

```
SELECT * FROM INVENTORY;
```

```
mysql> SELECT * FROM INVENTORY;
+------------+--------------+----------+-------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | PRICE |
+------------+--------------+----------+-------+
|          2 | SMARTWATCH   |        4 |   540 |
|          3 | SPEAKER      |        2 |   220 |
|          4 | MIC          |        8 |   100 |
|          5 | TABLETS      |        6 |   680 |
|          6 | KEYBOARD     |       10 |   140 |
|          7 | LAPTOP       |       12 |  1025 |
+------------+--------------+----------+-------+
6 rows in set (0.00 sec)
```

Updated table data in Product

```
SELECT * FROM PRODUCT;
```

```
mysql> SELECT * FROM PRODUCT;
+------------+--------------+----------+--------------+---------------------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | STATUS       | PRODUCT_LOG         |
+------------+--------------+----------+--------------+---------------------+
|          1 | SMARTPHONE   |        0 | NOT AVAILABLE | 2021-11-18 16:54:58 |
|          2 | SMARTWATCH   |        4 | AVAILABLE    | 2021-11-18 16:49:29 |
|          3 | SPEAKER      |        2 | AVAILABLE    | 2021-11-18 16:49:29 |
|          4 | MIC          |        8 | AVAILABLE    | 2021-11-18 16:49:29 |
|          5 | TABLETS      |        6 | AVAILABLE    | 2021-11-18 16:49:29 |
|          6 | KEYBOARD     |       10 | AVAILABLE    | 2021-11-18 16:49:29 |
|          7 | LAPTOP       |       12 | AVAILABLE    | 2021-11-18 16:49:30 |
+------------+--------------+----------+--------------+---------------------+
7 rows in set (0.00 sec)

mysql>
```

II. Write a before update trigger on Inventory table to check that Quantity and Price should not be less than ‘0’.

Before Update trigger on Inventory

```
DELIMITER //
CREATE TRIGGER BEFORE_UPDATING_INVENTORY BEFORE UPDATE ON INVENTORY
FOR EACH ROW
BEGIN
   IF NEW.QUANTITY IS NOT NULL THEN
       IF NEW.QUANTITY < 0 THEN
```

```
            SIGNAL SQLSTATE "25000"
            SET MESSAGE_TEXT = "QUANTITY can't be less than 0";
        END IF;
    END IF;

    IF NEW.PRICE IS NOT NULL THEN
        IF NEW.PRICE < 0 THEN
            SIGNAL SQLSTATE "25000"
            SET MESSAGE_TEXT = "PRICE can't be less than 0";
        END IF;
    END IF;
END;
//
DELIMITER ;
```

```
mysql> DELIMITER //
mysql> CREATE TRIGGER BEFORE_UPDATING_INVENTORY BEFORE UPDATE ON INVENTORY
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.QUANTITY IS NOT NULL THEN
    ->         IF NEW.QUANTITY < 0 THEN
    ->             SIGNAL SQLSTATE "25000"
    ->             SET MESSAGE_TEXT = "QUANTITY can't be less than 0";
    ->         END IF;
    ->     END IF;
    ->
    ->     IF NEW.PRICE IS NOT NULL THEN
    ->         IF NEW.PRICE < 0 THEN
    ->             SIGNAL SQLSTATE "25000"
    ->             SET MESSAGE_TEXT = "PRICE can't be less than 0";
    ->         END IF;
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.34 sec)

mysql> DELIMITER ;
```

Update query for the inventory (failed hence price can’t be to less than zero)

```
UPDATE INVENTORY
SET PRICE = -20
WHERE PRODUCT_ID = 5;
```

```
mysql> UPDATE INVENTORY
    -> SET PRICE = -20
    -> WHERE PRODUCT_ID = 5;
ERROR 1644 (25000): PRICE can't be less than 0
mysql>
```

Update query for the inventory(failed hence quantity can’t be to less than zero)

```
UPDATE INVENTORY
SET QUANTITY = -1
WHERE PRODUCT_ID = 3;
```

```
mysql> UPDATE INVENTORY
    -> SET QUANTITY = -1
    -> WHERE PRODUCT_ID = 3;
ERROR 1644 (25000): QUANTITY can't be less than 0
mysql>
```

III. Write a after delete trigger on Inventory table to remove the products having Quantity less than ‘0’.

(Delete all the data for fresh start)

```
DELETE FROM INVENTORY;
```

```
mysql> DELETE FROM INVENTORY;
Query OK, 7 rows affected (0.19 sec)
```

Insert into inventory table

```
INSERT INTO INVENTORY VALUES (1, 'SMARTPHONE', 5, 750);
INSERT INTO INVENTORY VALUES (3, 'SPEAKER', 2, 220);
INSERT INTO INVENTORY VALUES (2, 'SMARTWATCH', 4, 540);
INSERT INTO INVENTORY VALUES (4, 'MIC', 8, 100);
INSERT INTO INVENTORY VALUES (5, 'TABLETS', 6, 680);
INSERT INTO INVENTORY VALUES (6, 'KEYBOARD', 10, 140);
INSERT INTO INVENTORY VALUES (7, 'LAPTOP', 12, 1025);
```

```
mysql> INSERT INTO INVENTORY VALUES (1, 'SMARTPHONE', 5, 750);
Query OK, 1 row affected (0.18 sec)

mysql> INSERT INTO INVENTORY VALUES (3, 'SPEAKER', 2, 220);
Query OK, 1 row affected (0.10 sec)

mysql> INSERT INTO INVENTORY VALUES (2, 'SMARTWATCH', 4, 540);
Query OK, 1 row affected (0.18 sec)

mysql> INSERT INTO INVENTORY VALUES (4, 'MIC', 8, 100);
Query OK, 1 row affected (0.17 sec)

mysql> INSERT INTO INVENTORY VALUES (5, 'TABLETS', 6, 680);
Query OK, 1 row affected (0.17 sec)

mysql> INSERT INTO INVENTORY VALUES (6, 'KEYBOARD', 10, 140);
Query OK, 1 row affected (0.12 sec)

mysql> INSERT INTO INVENTORY VALUES (7, 'LAPTOP', 12, 1025);
Query OK, 1 row affected (0.16 sec)
```

Updated inventory table

```
SELECT * FROM INVENTORY;
```

```
mysql> SELECT * FROM INVENTORY;
+------------+--------------+----------+-------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | PRICE |
+------------+--------------+----------+-------+
|          1 | SMARTPHONE   |        5 |   750 |
|          2 | SMARTWATCH   |        4 |   540 |
|          3 | SPEAKER      |        2 |   220 |
|          4 | MIC          |        8 |   100 |
|          5 | TABLETS      |        6 |   680 |
|          6 | KEYBOARD     |       10 |   140 |
|          7 | LAPTOP       |       12 |  1025 |
+------------+--------------+----------+-------+
7 rows in set (0.00 sec)
```

Update product table

```
SELECT * FROM PRODUCT;
```

```
mysql> SELECT * FROM PRODUCT;
+------------+--------------+----------+-----------+---------------------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | STATUS    | PRODUCT_LOG         |
+------------+--------------+----------+-----------+---------------------+
|          1 | SMARTPHONE   |        5 | AVAILABLE | 2021-11-18 17:23:43 |
|          2 | SMARTWATCH   |        4 | AVAILABLE | 2021-11-18 17:23:43 |
|          3 | SPEAKER      |        2 | AVAILABLE | 2021-11-18 17:23:43 |
|          4 | MIC          |        8 | AVAILABLE | 2021-11-18 17:23:43 |
|          5 | TABLETS      |        6 | AVAILABLE | 2021-11-18 17:23:43 |
|          6 | KEYBOARD     |       10 | AVAILABLE | 2021-11-18 17:23:44 |
|          7 | LAPTOP       |       12 | AVAILABLE | 2021-11-18 17:23:45 |
+------------+--------------+----------+-----------+---------------------+
7 rows in set (0.00 sec)

mysql>
```

After delete trigger on inventory

```
DELIMITER //
CREATE TRIGGER AFTER_DELETING_INVENTORY AFTER DELETE ON INVENTORY
FOR EACH ROW
BEGIN
   UPDATE PRODUCT
   SET QUANTITY = 0, STATUS = 'NOT AVAILABLE', PRODUCT_LOG = NOW()
   WHERE PRODUCT_ID = OLD.PRODUCT_ID;

   DELETE FROM PRODUCT WHERE QUANTITY < 0 ;
END;
//
DELIMITER ;
```

```
mysql> DELIMITER //
mysql> CREATE TRIGGER AFTER_DELETING_INVENTORY AFTER DELETE ON INVENTORY
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE PRODUCT
    ->     SET QUANTITY = 0, STATUS = 'NOT AVAILABLE', PRODUCT_LOG = NOW()
    ->     WHERE PRODUCT_ID = OLD.PRODUCT_ID;
    ->
    ->     DELETE FROM PRODUCT WHERE QUANTITY < 0 ;
    -> END;
    -> //
Query OK, 0 rows affected (0.19 sec)

mysql> DELIMITER ;
mysql>
```

Delete command on inventory

```
DELETE FROM INVENTORY WHERE PRODUCT_ID = 3;
```

```
mysql> DELETE FROM INVENTORY WHERE PRODUCT_ID = 3;
Query OK, 1 row affected (0.15 sec)

mysql>
```

(updated inventory product_id=3 gone)

```
SELECT * FROM INVENTORY;
```

```
mysql> SELECT * FROM INVENTORY;
+------------+--------------+----------+-------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | PRICE |
+------------+--------------+----------+-------+
|          1 | SMARTPHONE   |        5 |   750 |
|          2 | SMARTWATCH   |        4 |   540 |
|          4 | MIC          |        8 |   100 |
|          5 | TABLETS      |        6 |   680 |
|          6 | KEYBOARD     |       10 |   140 |
|          7 | LAPTOP       |       12 |  1025 |
+------------+--------------+----------+-------+
6 rows in set (0.00 sec)
```

(updated product product_id=3 updated quantity as 0 and not available)

```
SELECT * FROM PRODUCT;
```

```
mysql> SELECT * FROM PRODUCT;
+------------+--------------+----------+---------------+---------------------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | STATUS        | PRODUCT_LOG         |
+------------+--------------+----------+---------------+---------------------+
|          1 | SMARTPHONE   |        5 | AVAILABLE     | 2021-11-18 17:23:43 |
|          2 | SMARTWATCH   |        4 | AVAILABLE     | 2021-11-18 17:23:43 |
|          3 | SPEAKER      |        0 | NOT AVAILABLE | 2021-11-18 17:30:48 |
|          4 | MIC          |        8 | AVAILABLE     | 2021-11-18 17:23:43 |
|          5 | TABLETS      |        6 | AVAILABLE     | 2021-11-18 17:23:43 |
|          6 | KEYBOARD     |       10 | AVAILABLE     | 2021-11-18 17:23:44 |
|          7 | LAPTOP       |       12 | AVAILABLE     | 2021-11-18 17:23:45 |
+------------+--------------+----------+---------------+---------------------+
7 rows in set (0.00 sec)

mysql>
```

IV. Write a before insert trigger on Purchase table to check whether the product selected by the user is available or not for purchase. If available,enter an entry else show error.

Before insert trigger on purchase

```
DELIMITER //
CREATE TRIGGER INSERT_TO_PURCHASE BEFORE INSERT ON PURCHASE
FOR EACH ROW
BEGIN
   DECLARE CURRENT_STATUS VARCHAR(20);

   SELECT STATUS INTO CURRENT_STATUS
   FROM PRODUCT
   WHERE PRODUCT_ID = NEW.PRODUCT_ID;

   IF CURRENT_STATUS <> 'AVAILABLE' THEN
       SIGNAL SQLSTATE "25000"
       SET MESSAGE_TEXT = "PRODUCT isn't available for purchase.";
   END IF;

END;
//
DELIMITER ;
```

```
mysql> DELIMITER //
mysql> CREATE TRIGGER INSERT_TO_PURCHASE BEFORE INSERT ON PURCHASE
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE CURRENT_STATUS VARCHAR(20);
    ->
    ->     SELECT STATUS INTO CURRENT_STATUS
    ->     FROM PRODUCT
    ->     WHERE PRODUCT_ID = NEW.PRODUCT_ID;
    ->
    ->     IF CURRENT_STATUS <> 'AVAILABLE' THEN
    ->         SIGNAL SQLSTATE "25000"
    ->         SET MESSAGE_TEXT = "PRODUCT isn't available for purchase.";
    ->     END IF;
    ->
    -> END;
    -> //
Query OK, 0 rows affected (0.21 sec)

mysql> DELIMITER ;
mysql>
```

Insert a new data to purchase(since the product_id = 3 isn’t ‘available’ so throw the error message)

```
INSERT INTO PURCHASE VALUES (3, 'LOGITECH', 'SPEAKER', 5, 220, '2021-05-15');
```

```
mysql> INSERT INTO PURCHASE VALUES (3, 'LOGITECH', 'SPEAKER', 5, 220, '2021-05-15');
ERROR 1644 (25000): PRODUCT isn't available for purchase.
mysql>
```

Insert a new data to purchase(since the product_id = 4 is ‘available’ so insertion successful)

```
INSERT INTO PURCHASE VALUES (4, 'BOYA', 'MIC', 3, 200, '2021-08-02');
```

```
mysql> INSERT INTO PURCHASE VALUES (4, 'BOYA', 'MIC', 3, 200, '2021-08-02');
Query OK, 1 row affected (0.23 sec)

mysql>
```

Update purchase table with product_id = 4

```
SELECT * FROM PURCHASE;
```

```
mysql> SELECT * FROM PURCHASE;
+------------+------------+--------------+----------+-------+---------------+
| PRODUCT_ID | BRAND_NAME | PRODUCT_NAME | QUANTITY | PRICE | PURCHASE_DATE |
+------------+------------+--------------+----------+-------+---------------+
|          4 | BOYA       | MIC          |        3 |   200 | 2021-08-02    |
+------------+------------+--------------+----------+-------+---------------+
1 row in set (0.00 sec)
```

Quantity is to updated in the upcoming question's query

```
SELECT * FROM PRODUCT;
```

```
mysql> SELECT * FROM PRODUCT;
+------------+--------------+----------+---------------+---------------------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | STATUS        | PRODUCT_LOG         |
+------------+--------------+----------+---------------+---------------------+
|          1 | SMARTPHONE   |        5 | AVAILABLE     | 2021-11-18 17:23:43 |
|          2 | SMARTWATCH   |        4 | AVAILABLE     | 2021-11-18 17:23:43 |
|          3 | SPEAKER      |        0 | NOT AVAILABLE | 2021-11-18 17:30:48 |
|          4 | MIC          |        8 | AVAILABLE     | 2021-11-18 17:23:43 |
|          5 | TABLETS      |        6 | AVAILABLE     | 2021-11-18 17:23:43 |
|          6 | KEYBOARD     |       10 | AVAILABLE     | 2021-11-18 17:23:44 |
|          7 | LAPTOP       |       12 | AVAILABLE     | 2021-11-18 17:23:45 |
+------------+--------------+----------+---------------+---------------------+
7 rows in set (0.00 sec)

mysql>
```

V. Extending IV, also check for the quantity of the product. If available, enter an entry else show error.

Before insert trigger on purchase

```
DELIMITER //
CREATE TRIGGER INSERT_IN_PURCHASE_QUANTITY BEFORE INSERT ON PURCHASE
FOR EACH ROW
BEGIN
   DECLARE CURRENT_STATUS VARCHAR(20);
   DECLARE CURRENT_QUANTITY INTEGER;

   SELECT STATUS INTO CURRENT_STATUS
   FROM PRODUCT
   WHERE PRODUCT_ID = NEW.PRODUCT_ID;

   IF CURRENT_STATUS = 'AVAILABLE' THEN

       SELECT QUANTITY INTO CURRENT_QUANTITY
```

```
        FROM PRODUCT
        WHERE PRODUCT_ID = NEW.PRODUCT_ID;

        IF CURRENT_QUANTITY < NEW.QUANTITY THEN
            SIGNAL SQLSTATE "25000"
            SET MESSAGE_TEXT = "Sufficient Products not available.";
        END IF;

    ELSEIF CURRENT_STATUS <> 'AVAILABLE' THEN
        SIGNAL SQLSTATE "25000"
        SET MESSAGE_TEXT = "PRODUCT isn't available for purchase.";

    END IF;
END;
//
DELIMITER ;
```

```
mysql> DELIMITER //
mysql> CREATE TRIGGER INSERT_IN_PURCHASE_QUANTITY BEFORE INSERT ON PURCHASE
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE CURRENT_STATUS VARCHAR(20);
    ->     DECLARE CURRENT_QUANTITY INTEGER;
    ->
    ->     SELECT STATUS INTO CURRENT_STATUS
    ->     FROM PRODUCT
    ->     WHERE PRODUCT_ID = NEW.PRODUCT_ID;
    ->
    ->     IF CURRENT_STATUS = 'AVAILABLE' THEN
    ->
    ->         SELECT QUANTITY INTO CURRENT_QUANTITY
    ->         FROM PRODUCT
    ->         WHERE PRODUCT_ID = NEW.PRODUCT_ID;
    ->
    ->         IF CURRENT_QUANTITY < NEW.QUANTITY THEN
    ->             SIGNAL SQLSTATE "25000"
    ->             SET MESSAGE_TEXT = "Sufficient Products not available.";
    ->         END IF;
    ->
    ->     ELSEIF CURRENT_STATUS <> 'AVAILABLE' THEN
    ->         SIGNAL SQLSTATE "25000"
    ->         SET MESSAGE_TEXT = "PRODUCT isn't available for purchase.";
    ->
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.25 sec)

mysql> DELIMITER ;
mysql>
```

Insert a new data to purchase(since the product_id = 5 is available but isn't having the required quantity so throw the error message)

```
INSERT INTO PURCHASE VALUES (5, 'APPLE', 'TABLET', 8, 200, '2021-01-12');
```

```
mysql> INSERT INTO PURCHASE VALUES (5, 'APPLE', 'TABLET', 8, 200, '2021-01-12');
ERROR 1644 (25000): Sufficient Products not available.
mysql>
```

Insert a new data to purchase(since the product_id = 5 is 'available' and purchasing quantity is less than available so successfully inserted)

```
INSERT INTO PURCHASE VALUES (5, 'APPLE', 'TABLET', 3, 200, '2021-01-12');
```

```
mysql> INSERT INTO PURCHASE VALUES (5, 'APPLE', 'TABLET', 3, 200, '2021-01-12');
Query OK, 1 row affected (0.16 sec)

mysql>
```

Updated data on purchase table

```
SELECT * FROM PURCHASE;
```

```
mysql> SELECT * FROM PURCHASE;
+------------+------------+--------------+----------+-------+---------------+
| PRODUCT_ID | BRAND_NAME | PRODUCT_NAME | QUANTITY | PRICE | PURCHASE_DATE |
+------------+------------+--------------+----------+-------+---------------+
|          4 | BOYA       | MIC          |        3 |   200 | 2021-08-02    |
|          5 | APPLE      | TABLET       |        3 |   200 | 2021-01-12    |
+------------+------------+--------------+----------+-------+---------------+
2 rows in set (0.00 sec)

mysql>
```

VI. Write a after insert trigger on Purchase table to update the values at Inventory table.

Clean the entire purchase table since we have to update the quantities as well

```
DELETE FROM PURCHASE;
```

```
mysql> DELETE FROM PURCHASE;
Query OK, 2 rows affected (0.12 sec)

mysql>
mysql>
```

After insert trigger on purchase table

```
DELIMITER //
CREATE TRIGGER TRACK_QUANTITY AFTER INSERT ON PURCHASE
FOR EACH ROW
BEGIN
   DECLARE CURRENT_STATUS VARCHAR(20);
   DECLARE CURRENT_QUANTITY INTEGER;
   DECLARE UPDATED_QUANTITY INTEGER ;

   SELECT STATUS INTO CURRENT_STATUS
   FROM PRODUCT
   WHERE PRODUCT_ID = NEW.PRODUCT_ID;

   IF CURRENT_STATUS = 'AVAILABLE' THEN

       SELECT QUANTITY INTO CURRENT_QUANTITY
       FROM PRODUCT
       WHERE PRODUCT_ID = NEW.PRODUCT_ID;

       IF CURRENT_QUANTITY < NEW.QUANTITY THEN
           SIGNAL SQLSTATE "25000"
           SET MESSAGE_TEXT = "Sufficient Products not available.";

       ELSE
           SET UPDATED_QUANTITY = CURRENT_QUANTITY - NEW.QUANTITY;

           UPDATE INVENTORY
           SET QUANTITY = UPDATED_QUANTITY
           WHERE PRODUCT_ID = NEW.PRODUCT_ID;

           UPDATE PRODUCT
           SET QUANTITY = UPDATED_QUANTITY
           WHERE PRODUCT_ID = NEW.PRODUCT_ID;
       END IF;

   ELSEIF CURRENT_STATUS <> 'AVAILABLE' THEN
```

```
        SIGNAL SQLSTATE "25000"
        SET MESSAGE_TEXT = "PRODUCT isn't available for purchase.";

    END IF;
END;
//
DELIMITER ;
```

```
mysql> DELIMITER //
mysql> CREATE TRIGGER TRACK_QUANTITY AFTER INSERT ON PURCHASE
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE CURRENT_STATUS VARCHAR(20);
    ->     DECLARE CURRENT_QUANTITY INTEGER;
    ->     DECLARE UPDATED_QUANTITY INTEGER ;
    ->
    ->     SELECT STATUS INTO CURRENT_STATUS
    ->     FROM PRODUCT
    ->     WHERE PRODUCT_ID = NEW.PRODUCT_ID;
    ->
    ->     IF CURRENT_STATUS = 'AVAILABLE' THEN
    ->
    ->         SELECT QUANTITY INTO CURRENT_QUANTITY
    ->         FROM PRODUCT
    ->         WHERE PRODUCT_ID = NEW.PRODUCT_ID;
    ->
    ->         IF CURRENT_QUANTITY < NEW.QUANTITY THEN
    ->             SIGNAL SQLSTATE "25000"
    ->             SET MESSAGE_TEXT = "Sufficient Products not available.";
    ->
    ->         ELSE
    ->             SET UPDATED_QUANTITY = CURRENT_QUANTITY - NEW.QUANTITY;
    ->
    ->             UPDATE INVENTORY
    ->             SET QUANTITY = UPDATED_QUANTITY
    ->             WHERE PRODUCT_ID = NEW.PRODUCT_ID;
    ->
    ->             UPDATE PRODUCT
    ->             SET QUANTITY = UPDATED_QUANTITY
    ->             WHERE PRODUCT_ID = NEW.PRODUCT_ID;
    ->         END IF;
    ->
    ->     ELSEIF CURRENT_STATUS <> 'AVAILABLE' THEN
    ->         SIGNAL SQLSTATE "25000"
    ->         SET MESSAGE_TEXT = "PRODUCT isn't available for purchase.";
    ->
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.30 sec)

mysql> DELIMITER ;
mysql>
```

Before requesting for any purchase the current data at product and purchase table

```
SELECT * FROM PRODUCT;
SELECT * FROM PURCHASE;
```

```
mysql> SELECT * FROM PRODUCT;
+------------+--------------+----------+---------------+---------------------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | STATUS        | PRODUCT_LOG         |
+------------+--------------+----------+---------------+---------------------+
|          1 | SMARTPHONE   |        5 | AVAILABLE     | 2021-11-18 17:23:43 |
|          2 | SMARTWATCH   |        4 | AVAILABLE     | 2021-11-18 17:23:43 |
|          3 | SPEAKER      |        0 | NOT AVAILABLE | 2021-11-18 17:30:48 |
|          4 | MIC          |        8 | AVAILABLE     | 2021-11-18 17:23:43 |
|          5 | TABLETS      |        6 | AVAILABLE     | 2021-11-18 17:23:43 |
|          6 | KEYBOARD     |       10 | AVAILABLE     | 2021-11-18 17:23:44 |
|          7 | LAPTOP       |       12 | AVAILABLE     | 2021-11-18 17:23:45 |
+------------+--------------+----------+---------------+---------------------+
7 rows in set (0.00 sec)

mysql> SELECT * FROM PURCHASE;
Empty set (0.00 sec)

mysql>
```

Since the product_id = 5 is available and required quantity is less than available so query successful

```
INSERT INTO PURCHASE VALUES (5, 'APPLE', 'TABLET', 3, 200, '2021-01-12');
```

```
mysql> INSERT INTO PURCHASE VALUES (5, 'APPLE', 'TABLET', 3, 200, '2021-01-12');
Query OK, 1 row affected (0.18 sec)

mysql>
```

The updated quantity for prduct_id = 5 is reduced by 3 from 6, since 3 units are purchased

```
SELECT * FROM INVENTORY;
```

```
mysql> SELECT * FROM INVENTORY;
+------------+--------------+----------+-------+
| PRODUCT_ID | PRODUCT_NAME | QUANTITY | PRICE |
+------------+--------------+----------+-------+
|          1 | SMARTPHONE   |        5 |   750 |
|          2 | SMARTWATCH   |        4 |   540 |
|          3 | SPEAKER      |        2 |   220 |
|          4 | MIC          |        8 |   100 |
|          5 | TABLETS      |        3 |   680 |
|          6 | KEYBOARD     |       10 |   140 |
|          7 | LAPTOP       |       12 |  1025 |
+------------+--------------+----------+-------+
7 rows in set (0.00 sec)

mysql>
```

Updated purchase table

```
SELECT * FROM PURCHASE;
```

```
mysql> SELECT * FROM PURCHASE;
+------------+------------+--------------+----------+-------+---------------+
| PRODUCT_ID | BRAND_NAME | PRODUCT_NAME | QUANTITY | PRICE | PURCHASE_DATE |
+------------+------------+--------------+----------+-------+---------------+
|          5 | APPLE      | TABLET       |        3 |   200 | 2021-01-12    |
+------------+------------+--------------+----------+-------+---------------+
1 row in set (0.00 sec)

mysql>
```

Q2. Construct a table with the following details given below:

Employee (EID, DeptID, ENAME, PNUMBER, SALARY, DESIGNATION)
Department (DID, DeptNAME, NUM_OF_EMPLOYEES, HEAD_NAME, MANAGERS, WORKERS)
Post (DESIGNATION, NUM_OF_EMPLOYEES, TOTAL_AMOUNT)
EID, DeptID: Primary keys for the table Employee.
DID: Primary key for the table Department.
DESIGNATION: Primary keys for the table Post.

Department Table Creation

```
CREATE TABLE DEPARTMENT
(
   DEPARTMENT_ID INTEGER,
   DEPARTMENT_NAME VARCHAR(20),
   NUM_OF_EMPLOYEES INTEGER,
   HEAD_NAME VARCHAR(20),
   MANAGERS INTEGER,
   WORKERS INTEGER,
   PRIMARY KEY (DEPARTMENT_ID)
);
```

```
mysql> CREATE TABLE DEPARTMENT
    -> (
    ->     DEPARTMENT_ID INTEGER,
    ->     DEPARTMENT_NAME VARCHAR(20),
    ->     NUM_OF_EMPLOYEES INTEGER,
    ->     HEAD_NAME VARCHAR(20),
    ->     MANAGERS INTEGER,
    ->     WORKERS INTEGER,
    ->     PRIMARY KEY (DEPARTMENT_ID)
    -> );
Query OK, 0 rows affected (1.02 sec)
```

POST Table Creation

```
CREATE TABLE POST
(
   DESIGNATION VARCHAR(20),
```

```
   NUM_OF_EMPLOYEES INTEGER,
   TOTAL_AMOUNT FLOAT,
   PRIMARY KEY (DESIGNATION)
);
```

```
mysql> CREATE TABLE POST
    -> (
    ->     DESIGNATION VARCHAR(20),
    ->     NUM_OF_EMPLOYEES INTEGER,
    ->     TOTAL_AMOUNT FLOAT,
    ->     PRIMARY KEY (DESIGNATION)
    -> );
Query OK, 0 rows affected (2.26 sec)
```

EMPLOYEE Table Creation

```
CREATE TABLE EMPLOYEE
(
   EMPLOYEE_ID INTEGER,
   DEPARTMENT_ID INTEGER,
   EMPLOYEE_NAME VARCHAR(20),
   POST_NUMBER INTEGER,
   SALARY FLOAT,
   DESIGNATION VARCHAR(20),
   PRIMARY KEY (EMPLOYEE_ID, DEPARTMENT_ID),
   FOREIGN KEY (DESIGNATION) REFERENCES POST(DESIGNATION),
   FOREIGN KEY (DEPARTMENT_ID) REFERENCES DEPARTMENT(DEPARTMENT_ID)
);
```

```
mysql> CREATE TABLE EMPLOYEE
    -> (
    ->     EMPLOYEE_ID INTEGER,
    ->     DEPARTMENT_ID INTEGER,
    ->     EMPLOYEE_NAME VARCHAR(20),
    ->     POST_NUMBER INTEGER,
    ->     SALARY FLOAT,
    ->     DESIGNATION VARCHAR(20),
    ->     PRIMARY KEY (EMPLOYEE_ID, DEPARTMENT_ID),
    ->     FOREIGN KEY (DESIGNATION) REFERENCES POST(DESIGNATION),
    ->     FOREIGN KEY (DEPARTMENT_ID) REFERENCES DEPARTMENT(DEPARTMENT_ID)
    -> );
Query OK, 0 rows affected (1.10 sec)
```

I. Write a before insert and before delete trigger on the Employee table to insert/update records in Department and Post table.

```
DELIMITER //
CREATE TRIGGER INSERTION_UPDATE_DEPT_POST BEFORE INSERT ON EMPLOYEE
FOR EACH ROW
BEGIN
   INSERT INTO DEPARTMENT(DEPARTMENT_ID) VALUES (NEW.DEPARTMENT_ID);
   INSERT INTO POST(DESIGNATION) VALUES (NEW.DESIGNATION);
END;
//
DELIMITER ;
```

```
mysql> DELIMITER //
mysql> CREATE TRIGGER INSERTION_UPDATE_DEPT_POST BEFORE INSERT ON EMPLOYEE
    -> FOR EACH ROW
    -> BEGIN
    ->    INSERT INTO DEPARTMENT(DEPARTMENT_ID) VALUES (NEW.DEPARTMENT_ID);
    ->    INSERT INTO POST(DESIGNATION) VALUES (NEW.DESIGNATION);
    -> END;
    -> //
Query OK, 0 rows affected (0.23 sec)

mysql> DELIMITER ;
mysql>
```

II. Write a before insert trigger on the Employee table to input custom value for column having NULL value.

III. Extending II, also write to allow DeptID to be an existing one.

IV. Write a before delete trigger on the Employee table to update values in the Department and Post table.

V. Write a before update trigger on the Department table to restrict changes on the NUM_OF_EMPLOYEES column.

VI. Write a before update trigger on the Department table to change value on NUM_OF_EMPLOYEES column only when there is a change in MANAGERS or WORKERS columns and update on Post table.

VII. Write a before insert and before update trigger on the Department table to use HEAD_NAME with the Employee having Designation as HEAD.

VIII. Write a before update trigger on Post table to restrict changes onNUM_OF_EMPLOYEES column