Integrantes:

Acosta Baez Bryan Alan

Gonzalez Munguía Antonio

1. **Se creo la base de datos GAMESTOP con el siguiente commando:**

CREATE DATABASE GAMESTOP;

2. **Se crearon las siguientes tablas:**

```
CREATE TABLE PRODUCTS(
ID_PRODUCT INT NOT NULL AUTO_INCREMENT,
NAME  VARCHAR(100) NOT NULL,
BRAND VARCHAR(100),
DESCRIPTION VARCHAR(100),
PRICE DOUBLE NOT NULL,
STOCK INT,
PRIMARY KEY(ID_PRODUCT)
);
```

```
CREATE TABLE ORDERS(
ID_ORDER INT NOT NULL AUTO_INCREMENT,
DATE_ORDER TIMESTAMP,
TOTAL DOUBLE NOT NULL,
ID_DESCRIPTION INT,
PRIMARY KEY(ID_ORDER),
FOREIGN KEY(ID_DESCRIPTION) REFERENCES ORDERS_DESCRIPTION(ID_DESCRIPTION)
);
```

```
CREATE TABLE ORDERS_DESCRIPTION(
ID_ORDER_DESCRIPTION INT NOT NULL,
ID_PRODUCT INT,
UNITS INT NOT NULL,
PRIMARY KEY(ID_ORDER_DESCRIPTION,ID_PRODUCT),
FOREIGN KEY(ID_PRODUCT) REFERENCES PRODUCTS(ID_PRODUCT)
);
```

```
CREATE TABLE PRODUCTS_AUDIT(
AUDIT_DATE TIMESTAMP,
AUDIT_USER VARCHAR(40) NOT NULL,
AUDIT_ACTION ENUM('update','delete','insert'),
ID_PRODUCT INT(11),
PRICE DOUBLE
);
```

3. **Se creo un trigger para controlar el stock de los productos**

```
--trigger para actualizar tabla productos en el campo stock cuando se hacen ordenes--
DELIMITER //
DROP TRIGGER IF EXISTS TG_STOCKUPDATE_AI;
CREATE TRIGGER TG_STOCKUPDATE_AI
AFTER INSERT ON ORDERS_DESCRIPTION
FOR EACH ROW
BEGIN
DECLARE NEW_UNITS INT DEFAULT 0;
DECLARE ID_PRO INT DEFAULT 0;
DECLARE FINALIZADO INT DEFAULT 0;
DECLARE UNIT INT DEFAULT 0;
DECLARE CUR CURSOR FOR
SELECT ID_PRODUCT, STOCK FROM PRODUCTS WHERE ID_PRODUCT = NEW.ID_PRODUCT
FOR UPDATE;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET FINALIZADO =1;
OPEN CUR;
REPEAT
FETCH CUR INTO ID_PRO, UNIT;
IF UNIT > 0 THEN SET NEW_UNITS= UNIT-NEW.UNITS;
ELSE SET NEW_UNITS = 1;
END IF;
UPDATE PRODUCTS SET STOCK = NEW_UNITS WHERE ID_PRODUCT = ID_PRO;
UNTIL FINALIZADO = 1 END REPEAT;
CLOSE CUR;
END
//
DELIMITER ;
```

4. **Se creo el trigger para modificar los precios**

```
DROP TRIGGER IF EXISTS TG_PRICEUPDATE_AI;
DELIMITER //
--Empezamos a crear el trigger
CREATE TRIGGER TG_PRICEUPDATE_AI
AFTER INSERT ON ORDERS
FOR EACH ROW
BEGIN
--Declaramos las variables
DECLARE UNIT INT;
DECLARE ID_PRO INT;
declare FINALIZADO INT DEFAULT 0;
DECLARE MENOR INT;
DECLARE PRI DOUBLE (5,2);
DECLARE ID_MENOR INT DEFAULT (SELECT @ID_MENOR:= ID_PRODUCT FROM ORDERS_DESCRIPTION
WHERE ID_ORDER_DESCRIPTION = NEW.ID_DESCRIPTION AND UNITS =(SELECT MIN(UNITS) FROM ORDERS_DESCRIPTION
    WHERE ID_ORDER_DESCRIPTION = NEW.ID_DESCRIPTION));

DECLARE CUR CURSOR FOR
SELECT ID_PRODUCT, UNITS
FROM orders_description
WHERE ID_ORDER_DESCRIPTION = NEW.ID_DESCRIPTION AND ID_PRODUCT != ID_MENOR
FOR UPDATE;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET FINALIZADO = 1;
```

```
--Restamos el 1% al producto menos vendido
SET MENOR =(SELECT MIN(UNITS) FROM ORDERS_DESCRIPTION WHERE ID_ORDER_DESCRIPTION =
    NEW.ID_DESCRIPTION AND ID_PRODUCT = ID_MENOR);
SET PRI = (SELECT PRICE FROM PRODUCTS WHERE ID_PRODUCT = ID_MENOR);
SET PRI = (PRI - (PRI * (MENOR*0.01)));

--Si el precio es menor a 1, dejamos 1 por defecto
IF PRI > 1 THEN UPDATE PRODUCTS SET PRICE = PRI WHERE ID_PRODUCT = ID_MENOR;
ELSE UPDATE PRODUCTS SET PRICE = 1 WHERE ID_PRODUCT = ID_MENOR;
END IF;

--Abrimos cursor para subir el precio de los productos
OPEN CUR;
REPEAT
FETCH CUR INTO ID_PRO, UNIT;
SET PRI = (SELECT PRICE FROM PRODUCTS WHERE ID_PRODUCT = ID_PRO);
SET PRI = (PRI+(PRI*(UNIT*(0.01))));
UPDATE PRODUCTS SET PRICE = PRI WHERE ID_PRODUCT = ID_PRO;
UNTIL FINALIZADO = 1 END REPEAT;
CLOSE CUR;

END
//
DELIMITER ;
```

5. **Tabla de productos con datos**

```
+------------+------------------------+----------+-----------------+-------+-------+
| ID_PRODUCT | NAME                   | BRAND    | DESCRIPTION     | PRICE | STOCK |
+------------+------------------------+----------+-----------------+-------+-------+
|          1 | PERSONA 5              | ATLUS    | RPG             | 60.49 |    90 |
|          2 | BLOODBORNE             | SONY     | ACTION RPG      | 15.47 |    95 |
|          3 | GRAND THEFT AUTO V     | ROCKSTAR | ACTION          | 10.39 |    98 |
|          4 | RESIDENT EVIL 2 REMAKE | CAPCOM   | SURVIVAL HORROR | 59.39 |    99 |
|          5 | FIFA 19                | EA       | SPORTS          | 29.99 |   100 |
+------------+------------------------+----------+-----------------+-------+-------+
5 rows in set (0.31 sec)
```

6. **Insercción de compras a la tabla order_descrption**

```
mysql> select * from orders_description;
+-----------------------+------------+-------+
| ID_ORDER_DESCRIPTION | ID_PRODUCT | UNITS |
+-----------------------+------------+-------+
|                     1 |          1 |    10 |
|                     1 |          2 |     5 |
|                     5 |          3 |     2 |
|                     5 |          4 |     1 |
|                    10 |          4 |     4 |
|                    10 |          5 |     1 |
+-----------------------+------------+-------+
6 rows in set (0.00 sec)
```

7. **Insercción de la compra hecha en la table orders, donde se activa el trigger**

```
mysql> insert into orders values (null,now(),267.55,10);
Query OK, 1 row affected (1.18 sec)

mysql> select * from products;
+------------+------------------------+----------+-----------------+-------+-------+
| ID_PRODUCT | NAME                   | BRAND    | DESCRIPTION     | PRICE | STOCK |
+------------+------------------------+----------+-----------------+-------+-------+
|          1 | PERSONA 5              | ATLUS    | RPG             | 60.49 |    90 |
|          2 | BLOODBORNE             | SONY     | ACTION RPG      | 15.47 |    95 |
|          3 | GRAND THEFT AUTO V     | ROCKSTAR | ACTION          | 10.39 |    98 |
|          4 | RESIDENT EVIL 2 REMAKE | CAPCOM   | SURVIVAL HORROR | 64.24 |    95 |
|          5 | FIFA 19                | EA       | SPORTS          | 29.69 |    99 |
+------------+------------------------+----------+-----------------+-------+-------+
5 rows in set (0.00 sec)
```