National University
Of Computer and Emerging Sciences

Lab Task # 10

Name – Muhammad Taha

Roll NO – 23P-0559

SECTION – BCS(4A)

Subject – Database Systems - LAB

Class Task:

Try to create a similar example where:

- **You add a new product to a product table.**
- **Add an inventory entry and set a savepoint.**
- **Deduct from inventory on order placement.**
- **Rollback if the inventory quantity goes negative, ensuring inventory consistency.**

```
-- 23P-0559 Muhammad Taha BCS(4A)

-- 1. Create tables
CREATE TABLE Product (
   ProductID INT PRIMARY KEY,
   ProductName VARCHAR(100)
);

CREATE TABLE Inventory (
   ProductID INT,
   Quantity INT,
   FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);

-- 2. Add a new product
INSERT INTO Product (ProductID, ProductName) VALUES (1, 'Wireless Mouse');

-- 3. Add inventory entry
INSERT INTO Inventory (ProductID, Quantity) VALUES (1, 10);

-- 4. Set a savepoint
SAVEPOINT InventorySave;

-- 5. Simulate order placement (deduct quantity)
UPDATE Inventory
SET Quantity = Quantity - 12
WHERE ProductID = 1;

-- 6. Check and rollback if quantity is negative
-- (Normally you would do this in a stored procedure or transaction block)

-- Let's manually check
SELECT Quantity FROM Inventory WHERE ProductID = 1;

-- Assume we fetched Quantity and found it < 0

-- Rollback to savepoint if negative
ROLLBACK TO InventorySave;

-- (Optionally) commit if all good
COMMIT;
```

```
-- 23P-0559 Muhammad Taha BCS(4A)

-- 1. Create tables
CREATE TABLE Product (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100)
);
```

Script Output | Query Result

Task completed in 0.047 seconds

```
Table PRODUCT created.

Table INVENTORY created.

1 row inserted.

1 row inserted.

Savepoint created.
```

Task 1:

Create a new table named book_inventory with columns for book_id, book_name, quantity, and price.
Insert three different book records with initial quantities.
Without committing the transaction, reduce the quantity of one book and create a savepoint named
quantity_update.

```
-- 23P-0559 Muhammad Taha BCS(4A)

-- 1. Create the book_inventory table
CREATE TABLE book_inventory (
   book_id INT PRIMARY KEY,
   book_name VARCHAR(100),
   quantity INT,
   price DECIMAL(10, 2)
);

-- 2. Insert three different book records separately
INSERT INTO book_inventory (book_id, book_name, quantity, price)
VALUES (1, 'The Great Gatsby', 15, 9.99);
```

INSERT INTO book_inventory (book_id, book_name, quantity, price)
VALUES (2, '1984', 10, 12.50);

INSERT INTO book_inventory (book_id, book_name, quantity, price)
VALUES (3, 'To Kill a Mockingbird', 20, 8.75);

-- (No COMMIT yet!)

-- 3. Reduce the quantity of one book
UPDATE book_inventory
SET quantity = quantity - 3
WHERE book_id = 2; -- Reducing quantity of '1984'

-- 4. Create a savepoint after the update
SAVEPOINT quantity_update;

Worksheet | Query Builder

```sql
-- 23P-0559 Muhammad Taha BCS(4A)

-- 1. Create the book_inventory table
CREATE TABLE book_inventory (
    book_id INT PRIMARY KEY,
    book_name VARCHAR(100),
    quantity INT,
    price DECIMAL(10, 2)
);

-- 2. Insert three different book records separately
INSERT INTO book_inventory (book_id, book_name, quantity, price)
VALUES (1, 'The Great Gatsby', 15, 9.99);
```

Script Output | Query Result

Task completed in 0.058 seconds

```
1 row inserted.


1 row updated.


Savepoint created.
```

Task 2:

In the staff table, add a new staff member with an initial salary. Increase their salary by 12% and create a savepoint named salary_boost. Further increase the salary by 8%. Roll back the transaction to the salary_boost savepoint to undo the second increase.

```
-- 23P-0559 Muhammad Taha BCS(4A)

-- 1. Create the staff table
CREATE TABLE staff (
    staff_id INT PRIMARY KEY,
    staff_name VARCHAR(100),
    salary DECIMAL(10, 2)
);

-- 2. Add a new staff member with an initial salary
INSERT INTO staff (staff_id, staff_name, salary)
VALUES (1, 'Alice Johnson', 5000.00);

-- (No COMMIT yet!)

-- 3. Increase their salary by 12%
UPDATE staff
SET salary = salary * 1.12
WHERE staff_id = 1;

-- 4. Create a savepoint after the 12% increase
SAVEPOINT salary_boost;

-- 5. Further increase salary by 8%
UPDATE staff
SET salary = salary * 1.08
WHERE staff_id = 1;

-- 6. Roll back to the salary_boost savepoint (undo the 8% increase)
ROLLBACK TO salary_boost;

-- (Optionally) COMMIT to finalize the 12% increase
COMMIT;
```

Worksheet | Query Builder

```sql
-- 23P-0559 Muhammad Taha BCS(4A)

-- 1. Create the staff table
CREATE TABLE staff (
    staff_id INT PRIMARY KEY,
    staff_name VARCHAR(100),
    salary DECIMAL(10, 2)
);
```

Script Output | Query Result

Task completed in 0.048 seconds

```
1 row updated.


Savepoint created.


1 row updated.


Rollback complete.


Commit complete.
```

Task 3:

Use the vendors and supplies tables.
Insert a new vendor into the vendors table.
Then, insert a supply record for the vendor in the supplies table.
Use transaction control to ensure that both the vendor and supply records are inserted only if both
statements succeed; otherwise, roll back the changes.

```
-- 23P-0559 Muhammad Taha BCS(4A)

-- 1. Create vendors table

CREATE TABLE vendors (
   vendor_id INT PRIMARY KEY,
   vendor_name VARCHAR(100)
);

-- 2. Create supplies table
CREATE TABLE supplies (
   supply_id INT PRIMARY KEY,
   vendor_id INT,
   supply_name VARCHAR(100),
   FOREIGN KEY (vendor_id) REFERENCES vendors(vendor_id)
);
-- Step 1: Insert into vendors
INSERT INTO vendors (vendor_id, vendor_name)
VALUES (2, 'Quality Stationery');

-- Step 2: Insert into supplies
INSERT INTO supplies (supply_id, vendor_id, supply_name)
VALUES (200, 2, 'Notebooks');

-- Step 3: Commit if both inserts succeed
COMMIT;
```

```
-- 23P-0559 Muhammad Taha BCS(4A)

-- 1. Create vendors table

CREATE TABLE vendors (
    vendor_id INT PRIMARY KEY,
    vendor_name VARCHAR(100)
);

-- 2. Create supplies table (if not already created
CREATE TABLE supplies (
```

Script Output | Query Result

Task completed in 0.048 seconds

```
*Cause:    Usually a PL/SQL compilation error.
*Action:

1 row inserted.


1 row inserted.


Commit complete.
```

Task 4:

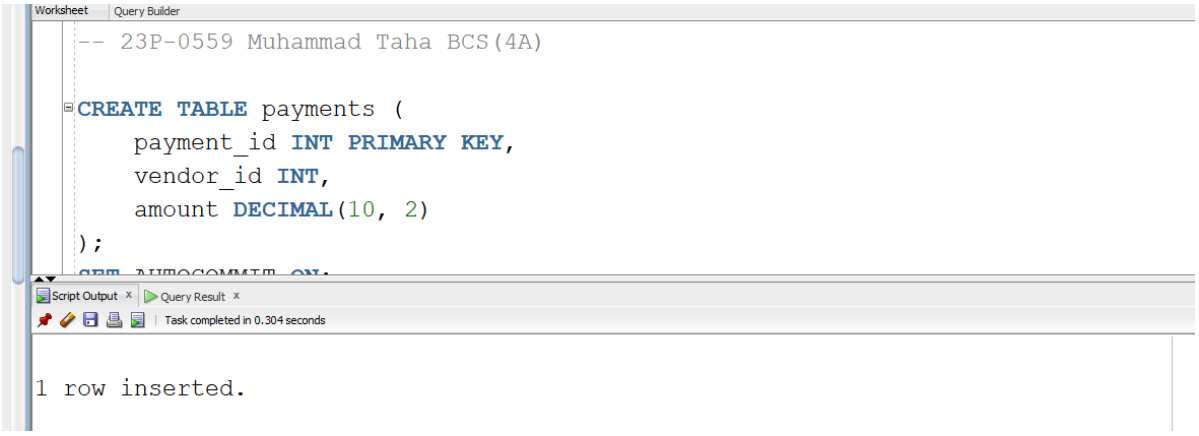
Enable AUTOCOMMIT mode in your SQL environment.
Insert a row in the payments table with payment_id, vendor_id, and amount.
After the insertion, verify if the row has been committed automatically.
Disable AUTOCOMMIT afterward.

```
-- 23P-0559 Muhammad Taha BCS(4A)

CREATE TABLE payments (
   payment_id INT PRIMARY KEY,
   vendor_id INT,
   amount DECIMAL(10, 2)
);
SET AUTOCOMMIT ON;

-- Insert into payments table
INSERT INTO payments (payment_id, vendor_id, amount)
VALUES (101, 5, 250.00);

-- Check if the row has been inserted
SELECT * FROM payments WHERE payment_id = 101;
```



Task 5:

Using the account_transactions table, simulate a transaction where multiple withdrawals and deposits are made on an account.
Set multiple savepoints after each withdrawal or deposit operation.
Rollback to a specific savepoint to undo one of the deposits.

```
-- PL/SQL block to insert transactions
BEGIN
   -- Withdraw 100 from the account
   INSERT INTO account_transactions (transaction_id, account_id, transaction_type, amount,
transaction_date)
   VALUES (transaction_seq.NEXTVAL, 1, 'withdrawal', 100, SYSDATE);
   SAVEPOINT withdrawal_1;

   -- Deposit 200 into the account
   INSERT INTO account_transactions (transaction_id, account_id, transaction_type, amount,
transaction_date)
   VALUES (transaction_seq.NEXTVAL, 1, 'deposit', 200, SYSDATE);
   SAVEPOINT deposit_1;

   -- Withdraw 50 from the account
   INSERT INTO account_transactions (transaction_id, account_id, transaction_type, amount,
transaction_date)
   VALUES (transaction_seq.NEXTVAL, 1, 'withdrawal', 50, SYSDATE);
   SAVEPOINT withdrawal_2;

   -- Deposit 300 into the account
   INSERT INTO account_transactions (transaction_id, account_id, transaction_type, amount,
transaction_date)
   VALUES (transaction_seq.NEXTVAL, 1, 'deposit', 300, SYSDATE);
   SAVEPOINT deposit_2;

   -- Rollback to the deposit_1 savepoint (undo the second deposit)
   ROLLBACK TO SAVEPOINT deposit_1;
```

```
    -- Commit the changes (those up to savepoint deposit_1)
    COMMIT;
END;
```

Worksheet | Query Builder

```
-- 23P-0559 Muhammad Taha BCS(4A)

CREATE TABLE account_transactions (
    transaction_id INT PRIMARY KEY,
    account_id INT,
    transaction_type VARCHAR(20),
    amount DECIMAL(10, 2),
    transaction_date TIMESTAMP
);

CREATE SEQUENCE transaction_seq
```

Script Output | Query Result

Task completed in 0.096 seconds

```
PL/SQL procedure successfully completed.
```