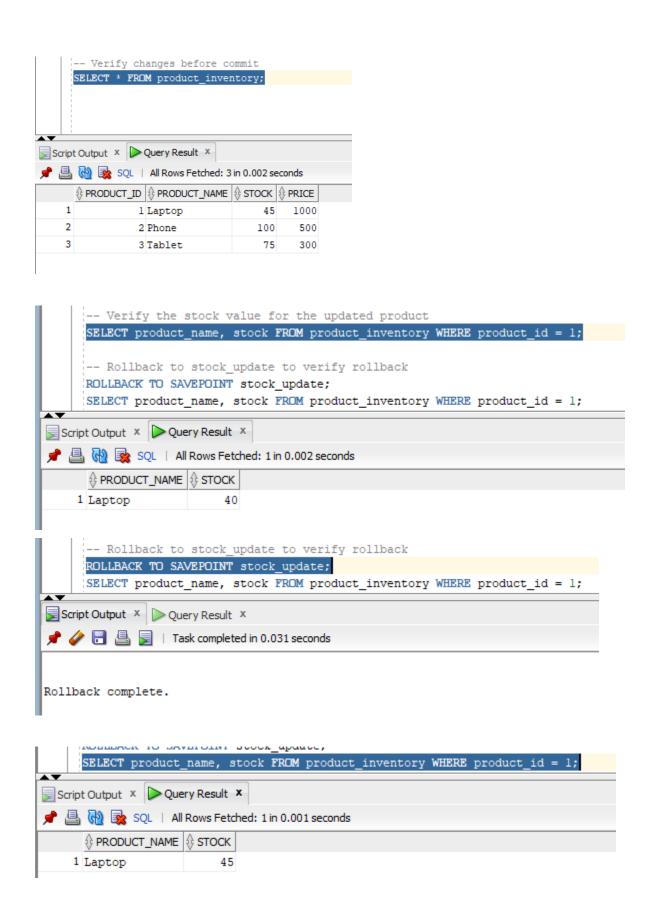
Lab tasks

TASK 1

```
Worksheet Query Builder
   --1. Create a new table named product_inventory with columns for product_id, product_name, stock,
    --and price. Insert three different product records with initial stock values. Without committing the
    --transaction, reduce the stock of one of the products and create a savepoint named stock update.
   CREATE TABLE product_inventory (
       product_id NUMBER PRIMARY KEY,
        product_name VARCHAR2(50),
        stock NUMBER,
        price NUMBER
    );
    INSERT INTO product_inventory (product_id, product_name, stock, price) VALUES (1, 'Laptop', 50, 1000);
    INSERT INTO product_inventory (product_id, product_name, stock, price) VALUES (2, 'Phone', 100, 500);
    INSERT INTO product_inventory (product_id, product_name, stock, price) VALUES (3, 'Tablet', 75, 300);
    UPDATE product inventory SET stock = stock - 5 WHERE product id = 1;
    SAVEPOINT stock_update;
    SELECT * FROM product_inventory;
    SELECT product_name, stock FROM product_inventory WHERE product_id = 1;
    ROLLBACK TO SAVEPOINT stock update;
    SELECT product_name, stock FROM product_inventory WHERE product_id = 1;
```

```
Create the product_inventory table
    CREATE TABLE product inventory (
         product_id NUMBER PRIMARY KEY,
          product_name VARCHAR2(50),
          stock NUMBER,
          price NUMBER
      -- Insert three product records
     INSERT INTO product_inventory (product_id, product_name, stock, price) VALUES (1, 'Laptop', 50, 1000);
      INSERT INTO product_inventory (product_id, product_name, stock, price) VALUES (2, 'Phone', 100, 500);
     INSERT INTO product_inventory (product_id, product_name, stock, price) VALUES (3, 'Tablet', 75, 300);
      -- Reduce stock and set a savepoint
     UPDATE product_inventory SET stock = stock - 5 WHERE product_id = 1;
     SAVEPOINT stock_update;
      -- Verify changes before commit
     SELECT * FROM product_inventory;
Script Output X Query Result X
📌 🧽 🔚 볼 📕 | Task completed in 0.032 seconds
Table PRODUCT_INVENTORY created.
```

```
-- Insert three product records
      INSERT INTO product_inventory (product_id, product_name, stock, price) VALUES (1, 'Laptop', 50, 1000);
      INSERT INTO product_inventory (product_id, product_name, stock, price) VALUES (2, 'Phone', 100, 500);
      INSERT INTO product_inventory (product_id, product_name, stock, price) VALUES (3, 'Tablet', 75, 300);
      -- Reduce stock and set a savepoint
      UPDATE product_inventory SET stock = stock - 5 WHERE product_id = 1;
     SAVEPOINT stock_update;
      -- Verify changes before commit
      SELECT * FROM product_inventory;
Script Output X De Query Result X
 🎤 🥔 🖥 🖺 🔋 | Task completed in 0.032 seconds
1 row inserted.
1 row inserted.
1 row inserted.
       -- Reduce stock and set a savepoint
       UPDATE product_inventory SET stock = stock - 5 WHERE product_id = 1;
       SAVEPOINT stock_update;
       -- Verify changes before commit
       SELECT * FROM product inventory;
Script Output X Duery Result X
 📌 🤌 🔡 🖺 🔋 | Task completed in 0.032 seconds
1 row updated.
    SAVEPOINT stock_update;
     -- Verify changes before commit
    SELECT * FROM product_inventory;
Script Output X Query Result X
📌 🥢 🔡 볼 🔋 | Task completed in 0.032 seconds
Savepoint created.
```



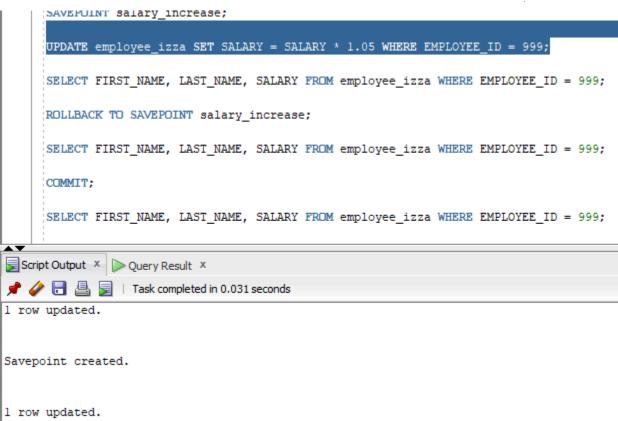
```
-----a savepoint named salary_increase, and then further increase it by another 5%. Rollback to the
     --===salary_increase savepoint.
    CREATE TABLE employee_izza AS SELECT * FROM HR.EMPLOYEES;
    SELECT * FROM employee izza;
    INSERT INTO employee_izza (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, HIRE_DATE, JOB_ID, SALARY)
    VALUES (999, 'John', 'Doe', 'JOHN.DOE@COMPANY.COM', SYSDATE, 'IT_PROG', 5000);
    SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY FROM employee mehak WHERE EMPLOYEE_ID = 999;
    SET TRANSACTION NAME 'salary_update';
    UPDATE employee_izza
    SET SALARY = SALARY * 1.1
    WHERE EMPLOYEE_ID = 999;
    SAVEPOINT salary_increase;
    UPDATE employee_izza
    SET SALARY = SALARY * 1.05
    WHERE EMPLOYEE ID = 999;
    SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
    ROLLBACK TO SAVEPOINT salary increase;
Script Output X Duery Result X
📌 🧳 🖥 🚇 🕎 | Task completed in 0.115 seconds
*ACCION:
```

Table EMPLOYEE_IZZA created.

		La	La	ı		
	COLUMN_NAME	⊕ DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EMPLOYEE_ID	NUMBER(6,0)	Yes	(null)	1	(null)
2	FIRST_NAME	VARCHAR2 (20 BYTE)	Yes	(null)	2	(null)
3	LAST_NAME	VARCHAR2 (25 BYTE)	No	(null)	3	(null)
4	EMAIL	VARCHAR2 (25 BYTE)	No	(null)	4	(null)
5	PHONE_NUMBER	VARCHAR2 (20 BYTE)	Yes	(null)	5	(null)
6	HIRE_DATE	DATE	No	(null)	6	(null)
7	JOB_ID	VARCHAR2(10 BYTE)	No	(null)	7	(null)
8	SALARY	NUMBER(8,2)	Yes	(null)	8	(null)
9	COMMISSION_PCT	NUMBER (2,2)	Yes	(null)	9	(null)
10	MANAGER_ID	NUMBER(6,0)	Yes	(null)	10	(null)
11	DEPARTMENT_ID	NUMBER (4,0)	Yes	(null)	11	(null)

```
SELECT * FROM employee izza;
       INSERT INTO employee_izza (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, HIRE_DATE, JOB_ID, SALARY)
       VALUES (999, 'John', 'Doe', 'JOHN.DOE@COMPANY.COM', SYSDATE, 'IT_PROG', 5000);
       SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY FROM employee mehak WHERE EMPLOYEE_ID = 999;
       SET TRANSACTION NAME 'salary_update';
       UPDATE employee_izza
       SET SALARY = SALARY * 1.1
       WHERE EMPLOYEE ID = 999;
       SAVEPOINT salary_increase;
       UPDATE employee_izza
       SET SALARY = SALARY * 1.05
       WHERE EMPLOYEE_ID = 999;
       SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
       ROLLBACK TO SAVEPOINT salary increase:
 Script Output × Query Result ×
 📌 🖺 🙀 🗽 SQL | Fetched 50 rows in 0.017 seconds
       $\\psi$ EMPLOYEE_ID $\\psi$ FIRST_NAME $\\psi$ LAST_NAME $\\psi$ LAST_NAME $\\psi$ EMAIL $$\\psi$ PHONE_NUMBER $$\\psi$ HIRE_DATE $$\\psi$ JOB_ID $$\\psi$ SALARY $\\phi$ COMMISSION_PCT $$\\psi$ MANAGER_ID $$\\phi$ DEPARTMENT_ID $$\\psi$
                 100 Steven King
                                              SKING 515.123.4567
                                                                             17-JUN-03 AD PRES
                                                                                                        24000
                                                                                                                        (null)
                                                                                                                                      (null)
                                            NKOCHHAR 515.123.4568
                 101 Neena Kochhar
                                                                            21-SEP-05 AD_VP
                                                                                                        17000
                                                                                                                         (null)
                 102 Lex
                                  De Haan LDEHAAN 515.123.4569 13-JAN-01 AD_VP
                                                                                                        17000
      3
                                                                                                                         (null)
                                                                                                                                         100
                                                                                                                                                           90
                 103 Alexander Hunold
                                              AHUNOLD 590.423.4567
                                                                             03-JAN-06 IT_PROG
                                                                                                         9000
                                                                                                                         (null)
                                                                                                                                         102
                                                                                                                                                           60
                                              BERNST 590.423.4568
                                                                             21-MAY-07 IT_PROG
                                                                                                         6000
                                                                                                                                                           60
                 104 Bruce
                                Ernst
                                                                                                                         (null)
                                                                                                                                         103
     6
                 105 David Austin
                                              DAUSTIN 590.423.4569
                                                                             25-JUN-05 IT_PROG
                                                                                                         4800
                                                                                                                         (null)
                                                                                                                                         103
                                                                                                                                                           60
                 106 Valli
                                 Pataballa VPATABAL 590.423.4560
                                                                             05-FEB-06 IT_PROG
                                                                                                         4800
                                                                                                                         (null)
                                                                                                                                         103
                                                                                                                                                          60
                                                                             07-FEB-07 IT_PROG
     8
                 107 Diana
                                  Lorentz
                                              DLORENTZ 590.423.5567
                                                                                                         4200
                                                                                                                         (null)
                                                                                                                                         103
                                                                                                                                                          60
                             Greenberg NGREENBE 515.124.4569
     9
                                                                             17-AUG-02 FI_MGR
                 108 Nancy
                                                                                                                         (null)
     INSERT INTO employee_izza (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, HIRE_DATE, JOB_ID, SALARY) VALUES (999, 'John', 'Doe', 'JOHN.DOE&COMPANY.COM', SYSDATE, 'II_PROG', 5000);
     SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY FROM employee mehak WHERE EMPLOYEE_ID = 999;
     SET TRANSACTION NAME 'salary_update';
    UPDATE employee_izza
SET SALARY = SALARY * 1.1
     WHERE EMPLOYEE ID = 999:
     SAVEPOINT salary_increase;
    UPDATE employee_izza
SET SALARY = SALARY * 1.05
WHERE EMPLOYEE_ID = 999;
     SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
    ROLLBACK TO SAVEPOINT salary_increase;
     SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
     COMMIT:
Script Output × Duery Result ×
📌 🥢 🔡 遏 🔋 | Task completed in 0.034 seconds
 Cause:
*Action:
Table EMPLOYEE_IZZA created.
1 row inserted.
```

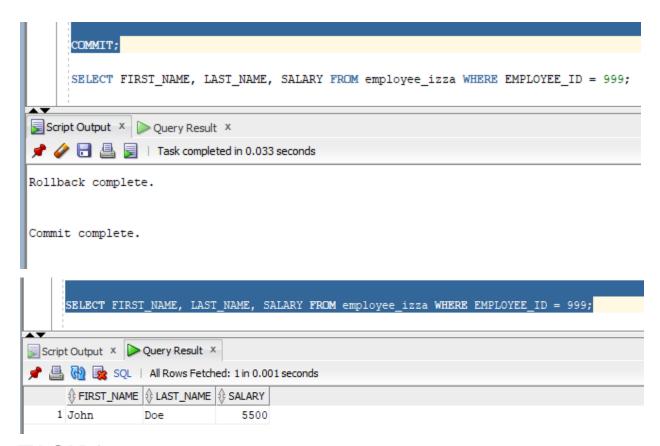
```
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
     SET TRANSACTION NAME 'salary_update';
     UPDATE employee_izza
     SET SALARY = SALARY * 1.1
     WHERE EMPLOYEE ID = 999;
     SAVEPOINT salary_increase;
     UPDATE employee_izza
     SET SALARY = SALARY * 1.05
     WHERE EMPLOYEE ID = 999;
     SELECT FIRST NAME, LAST NAME, SALARY FROM employee izza WHERE EMPLOYEE ID = 999;
     ROLLBACK TO SAVEPOINT salary_increase;
     SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
Script Output × Query Result ×
📌 🚇 🙀 🔯 SQL | All Rows Fetched: 1 in 0.003 seconds
  1 999 John
                           Doe
     COMMIT;
      SET TRANSACTION NAME 'salary update';
     UPDATE employee_izza
     SET SALARY = SALARY * 1.1
      WHERE EMPLOYEE ID = 999;
     SAVEPOINT salary increase;
     UPDATE employee_izza
     SET SALARY = SALARY * 1.05
     WHERE EMPLOYEE ID = 999;
     SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
     ROLLBACK TO SAVEPOINT salary_increase;
     SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
      COMMIT;
Script Output X Query Result X
📌 🤌 🔡 🖺 🔋 | Task completed in 0.031 seconds
*Cause:
          self-evident
*Action: commit (or rollback) transaction, and re-execute
Commit complete.
Transaction NAME succeeded.
```



```
SELECT FIRST NAME, LAST NAME, SALARY FROM employee izza WHERE EMPLOYEE ID = 999;
     ROLLBACK TO SAVEPOINT salary_increase;
     SELECT FIRST NAME, LAST NAME, SALARY FROM employee izza WHERE EMPLOYEE ID = 999;
     COMMIT;
     SELECT FIRST NAME, LAST NAME, SALARY FROM employee izza WHERE EMPLOYEE ID = 999;
Script Output X Query Result X
📌 🚇 🙌 💁 SQL | All Rows Fetched: 1 in 0.001 seconds

⊕ FIRST_NAME | ⊕ LAST_NAME | ⊕ SALARY

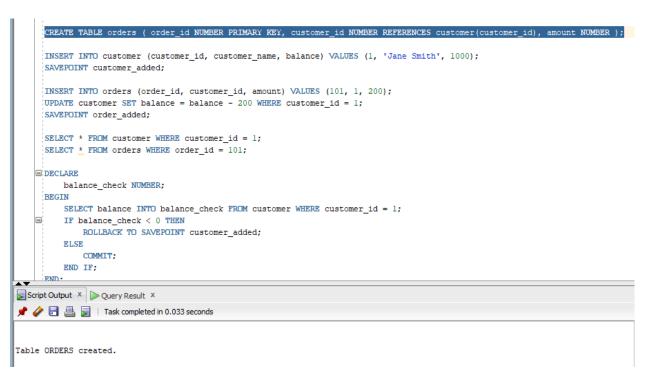
    1 John
                                5775
                 Doe
      ROLLBACK TO SAVEPOINT salary_increase;
     SELECT FIRST NAME, LAST NAME, SALARY FROM employee izza WHERE EMPLOYEE ID = 999;
     COMMIT:
      SELECT FIRST NAME, LAST NAME, SALARY FROM employee izza WHERE EMPLOYEE ID = 999;
Script Output X Deguery Result X
📌 🧼 🔡 💂 📘 | Task completed in 0.033 seconds
1 row updated.
Rollback complete.
      SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
      COMMIT;
      SELECT FIRST NAME, LAST NAME, SALARY FROM employee izza WHERE EMPLOYEE ID = 999;
Script Output X Query Result X
📌 📇 🙀 🕵 SQL | All Rows Fetched: 1 in 0.001 seconds
      1 John
                  Doe
                                  5500
```



```
======3. Use the customer and orders tables. Insert a new customer into the customer table. Then, insert an
         ======order for this customer in the orders table. Use a transaction control to ensure that both the
   -=======customer and order are inserted only if both statements are successful; otherwise, roll back the
  --===changes.
 CREATE TABLE customer (customer_id NUMBER PRIMARY KEY, customer_name VARCHAR2(50), balance NUMBER);
 CREATE TABLE orders ( order id NUMBER PRIMARY KEY, customer id NUMBER REFERENCES customer (customer id), amount NUMBER);
 INSERT INTO customer (customer_id, customer_name, balance) VALUES (1, 'Jane Smith', 1000);
 SAVEPOINT customer added;
 INSERT INTO orders (order_id, customer_id, amount) VALUES (101, 1, 200);
 UPDATE customer SET balance = balance - 200 WHERE customer_id = 1;
 SAVEPOINT order_added;
 SELECT * FROM customer WHERE customer_id = 1;
 SELECT * FROM orders WHERE order_id = 101;
□ DECLARE balance_check NUMBER;
     SELECT balance INTO balance_check FROM customer WHERE customer_id = 1;
     IF balance_check < 0 THEN
         ROLLBACK TO SAVEPOINT customer added;
     RLSE
        COMMIT;
     END IF;
 END:
  SELECT * FROM customer;
  SELECT * FROM orders;
```

```
🖻 -------3. Use the customer and orders tables. Insert a new customer into the customer table. Then, insert an
        ========order for this customer in the orders table. Use a transaction control to ensure that both the
     -----customer and order are inserted only if both statements are successful; otherwise, roll back the
      --===changes.
   CREATE TABLE customer (
         customer_id NUMBER PRIMARY KEY,
         customer_name VARCHAR2(50),
         balance NUMBER
   CREATE TABLE orders (
        order_id NUMBER PRIMARY KEY,
         customer_id NUMBER REFERENCES customer(customer_id),
    );
     INSERT INTO customer (customer_id, customer_name, balance) VALUES (1, 'Jane Smith', 1000);
     SAVEPOINT customer_added;
     INSERT INTO orders (order_id, customer_id, amount) VALUES (101, 1, 200);
     UPDATE customer SET balance = balance - 200 WHERE customer_id = 1;
     SAVEPOINT order_added;
     SELECT * FROM customer WHERE customer_id = 1;
     SELECT * FROM orders WHERE order_id = 101;
   DECLARE
   halance check NUMBER.
Script Output X Query Result X
📌 🤌 🖥 🖺 🔋 | Task completed in 0.032 seconds
```

Table CUSTOMER created.

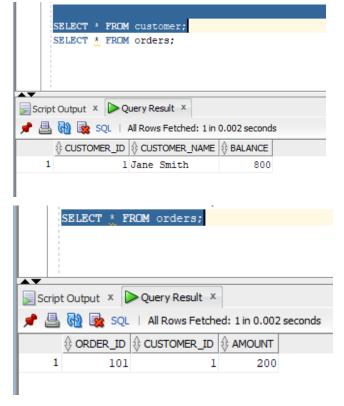


```
INSERT INTO customer (customer_id, customer_name, balance) VALUES (1, 'Jane Smith', 1000);
      SAVEPOINT customer added;
     INSERT INTO orders (order_id, customer_id, amount) VALUES (101, 1, 200);
     UPDATE customer SET balance = balance - 200 WHERE customer_id = 1;
     SAVEPOINT order added;
     SELECT * FROM customer WHERE customer_id = 1;
     SELECT * FROM orders WHERE order id = 101;
    □ DECLARE
        balance check NUMBER;
     BEGIN
        SELECT balance INTO balance_check FROM customer WHERE customer_id = 1;
        IF balance check < 0 THEN
             ROLLBACK TO SAVEPOINT customer_added;
         ELSE
             COMMIT;
         END IF;
    END.
Script Output X Deguery Result X
📌 🧽 🔡 🖺 🔋 | Task completed in 0.033 seconds
l row inserted.
      SAVEPOINT customer_added;
     INSERT INTO orders (order_id, customer_id, amount) VALUES (101, 1, 200);
     UPDATE customer SET balance = balance - 200 WHERE customer id = 1;
     SAVEPOINT order added;
     SELECT * FROM customer WHERE customer_id = 1;
     SELECT * FROM orders WHERE order_id = 101;
    ■ DECLARE
       balance_check NUMBER;
     BEGIN
        SELECT balance INTO balance_check FROM customer WHERE customer_id = 1;
        IF balance check < 0 THEN
            ROLLBACK TO SAVEPOINT customer_added;
         ELSE
            COMMIT;
         END IF;
Script Output X Deguery Result X
🎤 🥔 🔒 📕 | Task completed in 0.031 seconds
Savepoint created.
```

```
INSERT INTO orders (order_id, customer_id, amount) VALUES (101, 1, 200);
      UPDATE customer SET balance = balance - 200 WHERE customer id = 1;
      SAVEPOINT order added;
      SELECT * FROM customer WHERE customer_id = 1;
      SELECT * FROM orders WHERE order_id = 101;
    ■ DECLARE
         balance_check NUMBER;
      BEGIN
         SELECT balance INTO balance check FROM customer WHERE customer id = 1;
        IF balance_check < 0 THEN</pre>
              ROLLBACK TO SAVEPOINT customer_added;
          ELSE
              COMMIT;
         END IF:
      END;
      SELECT * FROM customer;
      SELECT * FROM orders;
Script Output X Deguery Result X
📌 🥜 🔡 🖺 🔋 | Task completed in 0.031 seconds
1 row inserted.
     UPDATE customer SET balance = balance - 200 WHERE customer_id = 1;
    SAVEPOINT order_added;
    SELECT * FROM customer WHERE customer id = 1;
    SELECT * FROM orders WHERE order_id = 101;
   ■ DECLARE
       balance_check NUMBER;
       SELECT balance INTO balance_check FROM customer WHERE customer_id = 1;
      IF balance check < 0 THEN
          ROLLBACK TO SAVEPOINT customer added;
       ELSE
       END IF;
    END:
    SELECT * FROM customer;
    SELECT * FROM orders;
Script Output X Query Result X
🎤 🥜 🖥 🖺 🔋 | Task completed in 0.033 seconds
1 row updated.
```

```
SAVEPOINT order_added;
     SELECT * FROM customer WHERE customer_id = 1;
     SELECT * FROM orders WHERE order_id = 101;
   ■ DECLARE
         balance check NUMBER;
        SELECT balance INTO balance_check FROM customer WHERE customer_id = 1;
        IF balance_check < 0 THEN
            ROLLBACK TO SAVEPOINT customer added;
        ELSE
            COMMIT:
        END IF;
     SELECT * FROM customer;
     SELECT * FROM orders;
Script Output X Duery Result X
📌 🥢 🔡 💂 🔋 | Task completed in 0.03 seconds
Savepoint created.
      SAVEPUINI OFGET_added;
      SELECT * FROM customer WHERE customer id = 1;
     SELECT * FROM orders WHERE order_id = 101;
    ■ DECLARE
        balance_check NUMBER;
     BEGIN
        SELECT balance INTO balance_check FROM customer WHERE customer_id = 1;
        IF balance_check < 0 THEN</pre>
             ROLLBACK TO SAVEPOINT customer_added;
         ELSE
            COMMIT;
        END IF;
     END;
     SELECT * FROM customer;
     SELECT * FROM orders;
Script Output X Query Result X
📌 🚇 🙀 🗽 SQL | All Rows Fetched: 1 in 0.003 seconds
    1
              1 Jane Smith
```

```
SELECT * FROM orders WHERE order_id = 101;
    ■ DECLARE
         balance check NUMBER;
     BEGIN
         SELECT balance INTO balance check FROM customer WHERE customer id = 1;
        IF balance_check < 0 THEN
            ROLLBACK TO SAVEPOINT customer_added;
         ELSE
             COMMIT;
        END IF;
     END;
     SELECT * FROM customer;
     SELECT * FROM orders;
Script Output × Query Result ×
📌 🖺 🙀 🗽 SQL | All Rows Fetched: 1 in 0.002 seconds
      1
            101
                                  200
   ☐ DECLARE balance check NUMBER;
     BEGIN
        SELECT balance INTO balance_check FROM customer WHERE customer_id = 1;
        IF balance_check < 0 THEN</pre>
            ROLLBACK TO SAVEPOINT customer added;
         ELSE
             COMMIT;
         END IF;
     END;
     SELECT * FROM customer;
     SELECT * FROM orders;
Script Output X Decry Result X
📌 🧼 🔡 📕 | Task completed in 0.13 seconds
*Action:
PL/SQL procedure successfully completed.
```



```
CREATE TABLE sales (sales_id NUMBER PRIMARY KEY, customer_id NUMBER REFERENCES customer(customer_id), amount NUMBER);

SET AUTOCOMMIT ON;

INSERT INTO sales (sales_id, customer_id, amount) VALUES (1, 1, 150);

SELECT * FROM sales;

SET AUTOCOMMIT OFF;

SELECT * FROM sales WHERE sales_id = 1;

Script Output * Query Result *

Query Result *

Query Result *
```

Table SALES created.

```
SET AUTOCOMMIT ON;
     INSERT INTO sales (sales_id, customer_id, amount) VALUES (1, 1, 150);
     SELECT * FROM sales;
     SET AUTOCOMMIT OFF;
     SELECT * FROM sales WHERE sales_id = 1;
 Script Output X DQuery Result X
 📌 🤌 🔡 🖺 🔋 | Task completed in 0.048 seconds
1 row inserted.
Commit complete.
     SELECT * FROM sales;
     SET AUTOCOMMIT OFF;
     SELECT * FROM sales WHERE sales id = 1;
Script Output × Query Result ×
📌 🖺 🙀 🗽 SQL | All Rows Fetched: 1 in 0.003 seconds
     1
                                 150
      SET AUTOCOMMIT OFF;
      SELECT * FROM sales WHERE sales id = 1;
Script Output × Query Result ×
📌 📇 🙌 🗽 SQL | All Rows Fetched: 1 in 0.001 seconds
      1
                                   150
```

```
========5. Using the transactions table, simulate a transaction where multiple debits and credits are made on ========an account. Set multiple savepoints after each debit or credit operation, and then rollback to a ========specific savepoint to undo one of the operations.
      REATE TABLE transactions (transaction_id NUMBER FRIMARY KEY, account_id NUMBER NOT NULL, transaction_type VARCHAR2(10) CHECK (transaction_type IN ('debit', 'credit')), amount NUMBER NOT NULL transaction date DATE DEFAULT SYSDATE
     START WITH 1
INCREMENT BY 1;
     INSERT INTO transactions (transaction_id, account_id, transaction_type, amount)
VALUES (transactions_seq.NEXTVAL, 1, 'debit', 100);
SAVEPOINT transaction_1;
     INSERT INTO transactions (transaction_id, account_id, transaction_type, amount)
VALUES (transaction_seq.NEXTVAL, 1, 'credit', 200);
SAVEPOINT transaction_2;
     INSERT INTO transactions (transaction_id, account_id, transaction_type, amount) VALUES (transactions_seq.NEXTVAL, 1, 'debit', 50);
     SAVEPOINT transaction 3;
     SELECT * FROM transactions;
    SELECT * FROM transactions;
Script Output X Query Result X
| Task completed in 0.084 seconds
Table TRANSACTIONS created.
             CREATE SEQUENCE transactions_seq START WITH 1 INCREMENT BY 1;
            INSERT INTO transactions (transaction_id, account_id, transaction_type, amount)
            VALUES (transactions_seq.NEXTVAL, 1, 'debit', 100);
            SAVEPOINT transaction 1;
            INSERT INTO transactions (transaction_id, account_id, transaction_type, amount)
            VALUES (transactions_seq.NEXTVAL, 1, 'credit', 200);
            SAVEPOINT transaction 2;
            INSERT INTO transactions (transaction_id, account_id, transaction_type, amount)
            VALUES (transactions_seq.NEXTVAL, 1, 'debit', 50);
            SAVEPOINT transaction_3;
            SELECT * FROM transactions;
            ROLLBACK TO SAVEPOINT transaction 2;
             SELECT * FROM transactions;
   Script Output X Query Result X
   📌 🤌 🔡 🖺 🔋 | Task completed in 0.046 seconds
   Table Inangactions created.
```

Sequence TRANSACTIONS SEQ created.

```
INSERT INTO transactions (transaction_id, account_id, transaction_type, amount) VALUES (transactions_seq.NEXTVAL, 1, 'debit', 100);
       SAVEPOINT transaction 1;
       INSERT INTO transactions (transaction_id, account_id, transaction_type, amount) VALUES (transactions_seq.NEXTVAL, 1, 'credit', 200);
       SAVEPOINT transaction_2;
       INSERT INTO transactions (transaction_id, account_id, transaction_type, amount) VALUES (transactions_seq.NEXTVAL, 1, 'debit', 50);
       SAVEPOINT transaction 3;
       SELECT * FROM transactions;
       ROLLBACK TO SAVEPOINT transaction_2;
       SELECT * FROM transactions;
 Script Output × Query Result ×
 📌 🧼 🖥 🚇 📘 | Task completed in 0.032 seconds
 l row inserted.
 Savepoint created.
      INSERT INTO transactions (transaction_id, account_id, transaction_type, amount) VALUES (transactions_seq.NEXTVAL, 1, 'credit', 200);
      INSERT INTO transactions (transaction_id, account_id, transaction_type, amount) VALUES (transactions_seq.NEXTVAL, 1, 'debit', 50);
      SAVEPOINT transaction_3;
      SELECT * FROM transactions;
      ROLLBACK TO SAVEPOINT transaction 2;
      SELECT * FROM transactions;
Script Output X Query Result X
 📌 🤌 🔡 볼 🔋 | Task completed in 0.032 seconds
1 row inserted.
Savepoint created.
      INSERT INTO transactions (transaction_id, account_id, transaction_type, amount) VALUES (transactions_seq.NEXTVAL, 1, 'debit', 50);
      SAVEPOINT transaction 3;
      SELECT * FROM transactions;
      ROLLBACK TO SAVEPOINT transaction_2;
      SELECT * FROM transactions;
Script Output X Query Result X
📌 🥢 🔒 遏 | Task completed in 0.032 seconds
1 row inserted.
Savepoint created.
```

