Lab tasks

TASK 1

Worksheet | Query Builder

```
--============================================================Task1 :=================================
--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 | Query Result

Task completed in 0.032 seconds

```
Table PRODUCT_INVENTORY created.
```

```sql
-- 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 | Query Result

Task completed in 0.032 seconds

```
1 row inserted.


1 row inserted.


1 row inserted.
```

```sql
-- 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 | Query Result

Task completed in 0.032 seconds

```
1 row updated.
```

```sql
SAVEPOINT stock_update;

-- Verify changes before commit
SELECT * FROM product_inventory;
```

Script Output | Query Result

Task completed in 0.032 seconds

```
Savepoint created.
```

```sql
-- Verify changes before commit
SELECT * FROM product_inventory;
```

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.002 seconds

	PRODUCT_ID	PRODUCT_NAME	STOCK	PRICE
1	1	Laptop	45	1000
2	2	Phone	100	500
3	3	Tablet	75	300

```sql
-- Verify the stock value for the updated product
SELECT product_name, stock FROM product_inventory WHERE product_id = 1;

-- Rollback to stock_update to verify rollback
ROLLBACK TO SAVEPOINT stock_update;
SELECT product_name, stock FROM product_inventory WHERE product_id = 1;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.002 seconds

	PRODUCT_NAME	STOCK
1	Laptop	40

```sql
-- Rollback to stock_update to verify rollback
ROLLBACK TO SAVEPOINT stock_update;
SELECT product_name, stock FROM product_inventory WHERE product_id = 1;
```

Script Output | Query Result

Task completed in 0.031 seconds

```
Rollback complete.
```

```sql
SELECT product_name, stock FROM product_inventory WHERE product_id = 1;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.001 seconds

	PRODUCT_NAME	STOCK
1	Laptop	45

TASK 2

```
--==============2 In the employee table, add a new employee with a salary. Then, increase their salary by 10%, set
--===============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 × | Query Result ×

Task completed in 0.115 seconds

```
*Action:

Table EMPLOYEE_IZZA created.
```

	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

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	(null)	(null)	90
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	(null)	100	90
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	(null)	100	90
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	(null)	102	60
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	(null)	103	60
6	105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	4800	(null)	103	60
7	106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06	IT_PROG	4800	(null)	103	60
8	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200	(null)	103	60
9	108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-02	FI_MGR	12008	(null)	101	100

```
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;

SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;

COMMIT;
```

Script Output × | Query Result ×

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;

COMMIT;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.003 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
1	999	John	Doe	5000

```
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 | Query Result

Task completed in 0.031 seconds

```
*Cause:    self-evident
*Action:   commit (or rollback) transaction, and re-execute

Commit complete.


Transaction NAME succeeded.
```

```
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;

SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
```

Script Output | Query Result

Task completed in 0.031 seconds

```
Transaction NAME succeeded.


1 row updated.


Savepoint created.
```

```
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;

SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
```

Script Output | Query Result

Task completed in 0.031 seconds

```
1 row updated.


Savepoint created.


1 row updated.
```

```
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 | Query Result

SQL | All Rows Fetched: 1 in 0.001 seconds

	FIRST_NAME	LAST_NAME	SALARY
1	John	Doe	5775

```
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 | Query Result

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 | Query Result

SQL | All Rows Fetched: 1 in 0.001 seconds

	FIRST_NAME	LAST_NAME	SALARY
1	John	Doe	5500

```
COMMIT;

SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
```

Script Output | Query Result

Task completed in 0.033 seconds

```
Rollback complete.


Commit complete.
```

```
SELECT FIRST_NAME, LAST_NAME, SALARY FROM employee_izza WHERE EMPLOYEE_ID = 999;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.001 seconds

	FIRST_NAME	LAST_NAME	SALARY
1	John	Doe	5500

TASK 3

```
--================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;
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;
```

```
--================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;
```

Script Output | Query Result

Task completed in 0.032 seconds

```
Table CUSTOMER created.
```

```
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;
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 | Query Result

Task completed in 0.033 seconds

```
Table ORDERS 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 | Query Result

Task completed in 0.033 seconds

```
1 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;
END;
```

Script Output | Query Result

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
        ROLLBACK TO SAVEPOINT customer_added;
    ELSE
        COMMIT;
    END IF;
END;

SELECT * FROM customer;
SELECT * FROM orders;
```

Script Output | Query Result

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;
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

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;
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

Task completed in 0.03 seconds

```
Savepoint created.
```

```
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;

SELECT * FROM customer;
SELECT * FROM orders;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.003 seconds

	CUSTOMER_ID	CUSTOMER_NAME	BALANCE
1	1	Jane Smith	800

```
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

	ORDER_ID	CUSTOMER_ID	AMOUNT
1	101	1	200

```
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

Task completed in 0.13 seconds

```
*Action:

PL/SQL procedure successfully completed.
```

```
SELECT * FROM customer;
SELECT * FROM orders;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.002 seconds

	CUSTOMER_ID	CUSTOMER_NAME	BALANCE
1	1	Jane Smith	800

```
SELECT * FROM orders;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.002 seconds

	ORDER_ID	CUSTOMER_ID	AMOUNT
1	101	1	200

TASK 4

```
--========================================4. Enable AUTOCOMMIT mode in your SQL environment. Insert a row in the sales table with
--========================================sales_id, customer_id, and amount. After the insertion, check if the row has been committed
--========================================automatically. Disable AUTOCOMMIT afterward.

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;
```

```
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

Task completed in 0.033 seconds

```
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 | Query Result

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

	SALES_ID	CUSTOMER_ID	AMOUNT
1	1	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

	SALES_ID	CUSTOMER_ID	AMOUNT
1	1	1	150

TASK 5

```
--============================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.

CREATE TABLE transactions (transaction_id NUMBER PRIMARY 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
);

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 | Query Result
Task completed in 0.084 seconds
02000. 00000 -  "missing %s keyword"

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 | Query Result
Task completed in 0.046 seconds

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

```
1 row inserted.


Savepoint created.
```

```
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

```
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 | Query Result | Task completed in 0.032 seconds

```
1 row inserted.


Savepoint created.
```

```
SELECT * FROM transactions;

ROLLBACK TO SAVEPOINT transaction_2;

SELECT * FROM transactions;
```

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.003 seconds

	TRANSACTION_ID	ACCOUNT_ID	TRANSACTION_TYPE	AMOUNT	TRANSACTION_DATE
1	1	1	debit	100	20-NOV-24
2	2	1	credit	200	20-NOV-24
3	3	1	debit	50	20-NOV-24

```
ROLLBACK TO SAVEPOINT transaction_2;

SELECT * FROM transactions;
```

Script Output | Query Result

Task completed in 0.033 seconds

```
Rollback complete.
```

```
SELECT * FROM transactions;
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.001 seconds

	TRANSACTION_ID	ACCOUNT_ID	TRANSACTION_TYPE	AMOUNT	TRANSACTION_DATE
1	1	1	debit	100	20-NOV-24
2	2	1	credit	200	20-NOV-24