23K-6005

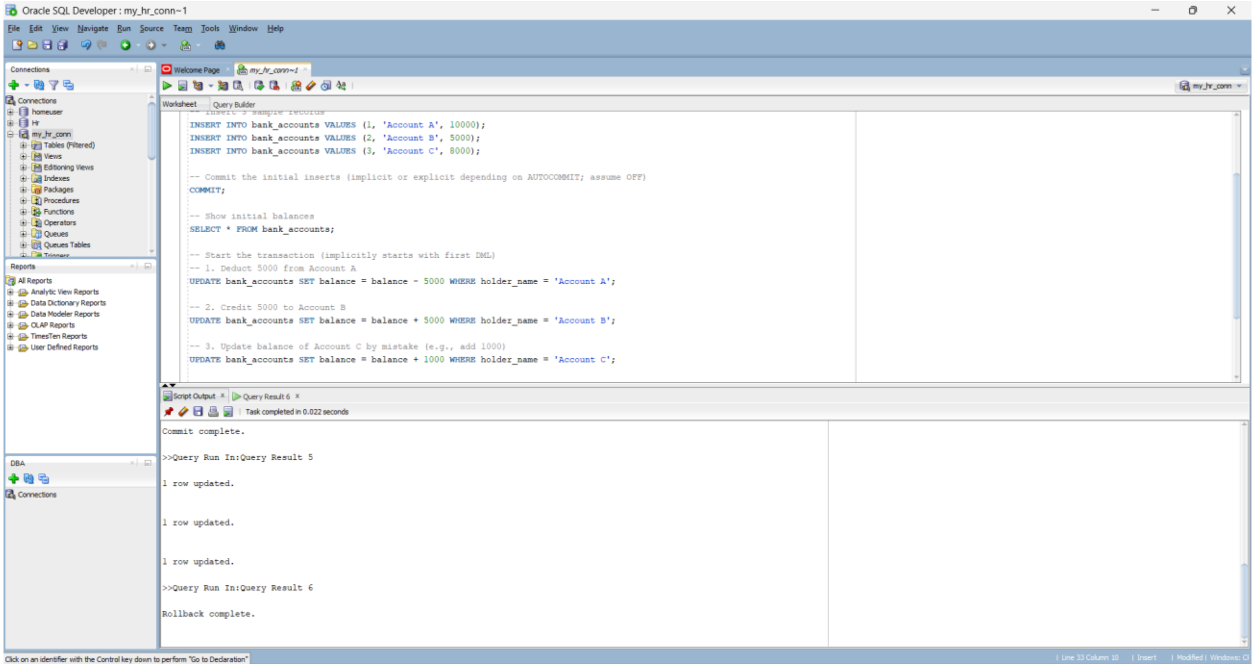
HARIS AHMED

BSAI-5A

DATABASE LAB 10:

LAB TASKS

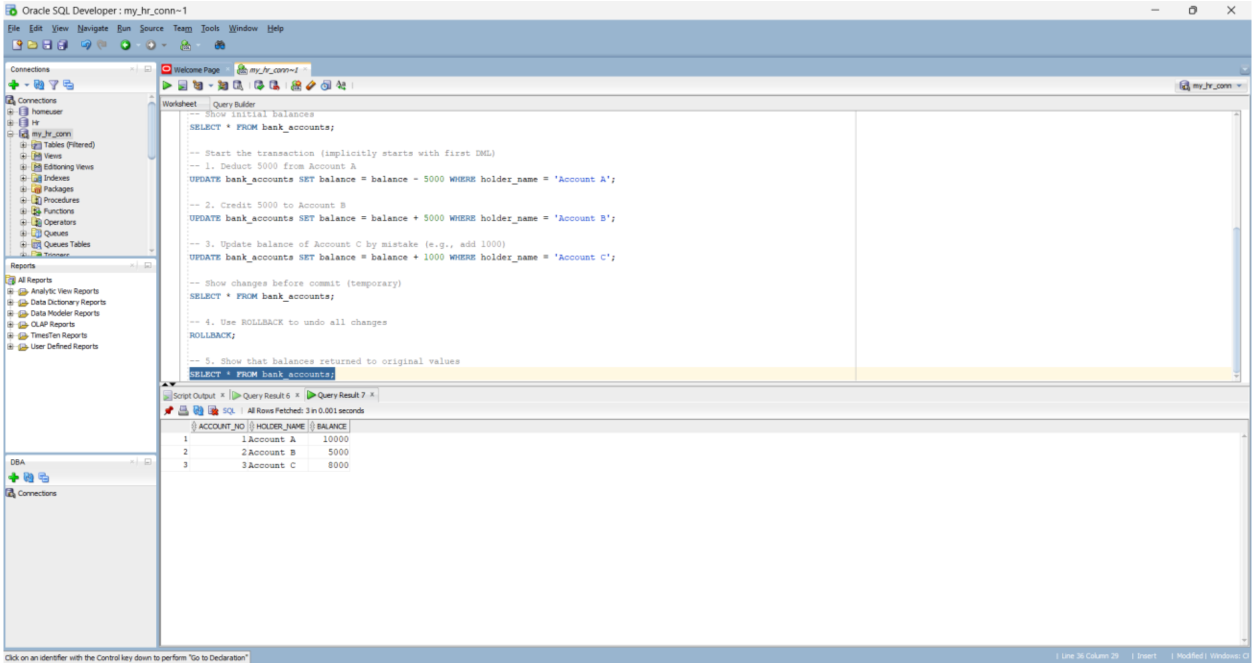
1a)



```sql
-- Insert 3 sample records
INSERT INTO bank_accounts VALUES (1, 'Account A', 10000);
INSERT INTO bank_accounts VALUES (2, 'Account B', 5000);
INSERT INTO bank_accounts VALUES (3, 'Account C', 8000);

-- Commit the initial inserts (implicit or explicit depending on AUTOCOMMIT; assume OFF)
COMMIT;

-- Show initial balances
SELECT * FROM bank_accounts;

-- Start the transaction (implicitly starts with first DML)
-- 1. Deduct 5000 from Account A
UPDATE bank_accounts SET balance = balance - 5000 WHERE holder_name = 'Account A';

-- 2. Credit 5000 to Account B
UPDATE bank_accounts SET balance = balance + 5000 WHERE holder_name = 'Account B';

-- 3. Update balance of Account C by mistake (e.g., add 1000)
UPDATE bank_accounts SET balance = balance + 1000 WHERE holder_name = 'Account C';
```

```
Commit complete.

>>Query Run In:Query Result 5

1 row updated.


1 row updated.


1 row updated.

>>Query Run In:Query Result 6

Rollback complete.
```

1b)

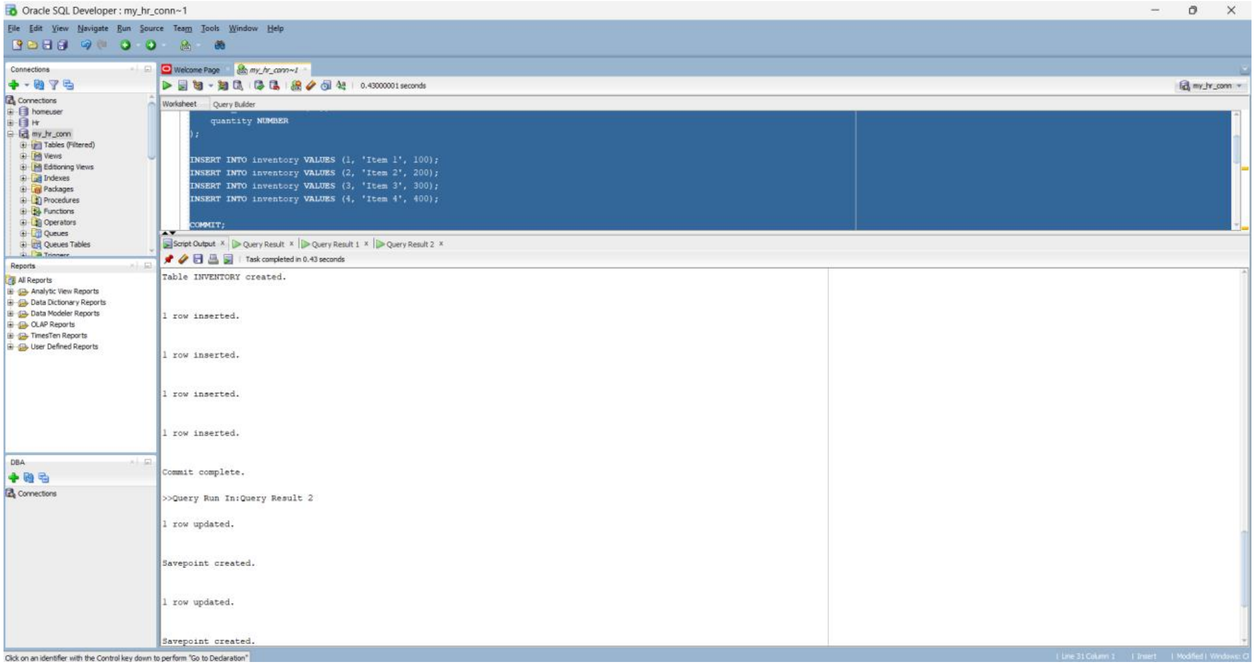


```
-- Show initial balances
SELECT * FROM bank_accounts;

-- Start the transaction (implicitly starts with first DML)
-- 1. Deduct 5000 from Account A
UPDATE bank_accounts SET balance = balance - 5000 WHERE holder_name = 'Account A';

-- 2. Credit 5000 to Account B
UPDATE bank_accounts SET balance = balance + 5000 WHERE holder_name = 'Account B';

-- 3. Update balance of Account C by mistake (e.g., add 1000)
UPDATE bank_accounts SET balance = balance + 1000 WHERE holder_name = 'Account C';

-- Show changes before commit (temporary)
SELECT * FROM bank_accounts;

-- 4. Use ROLLBACK to undo all changes
ROLLBACK;

-- 5. Show that balances returned to original values
SELECT * FROM bank_accounts;
```

All Rows Fetched: 3 in 0.001 seconds

	ACCOUNT_NO	HOLDER_NAME	BALANCE
1	1	Account A	10000
2	2	Account B	5000
3	3	Account C	8000

2a)

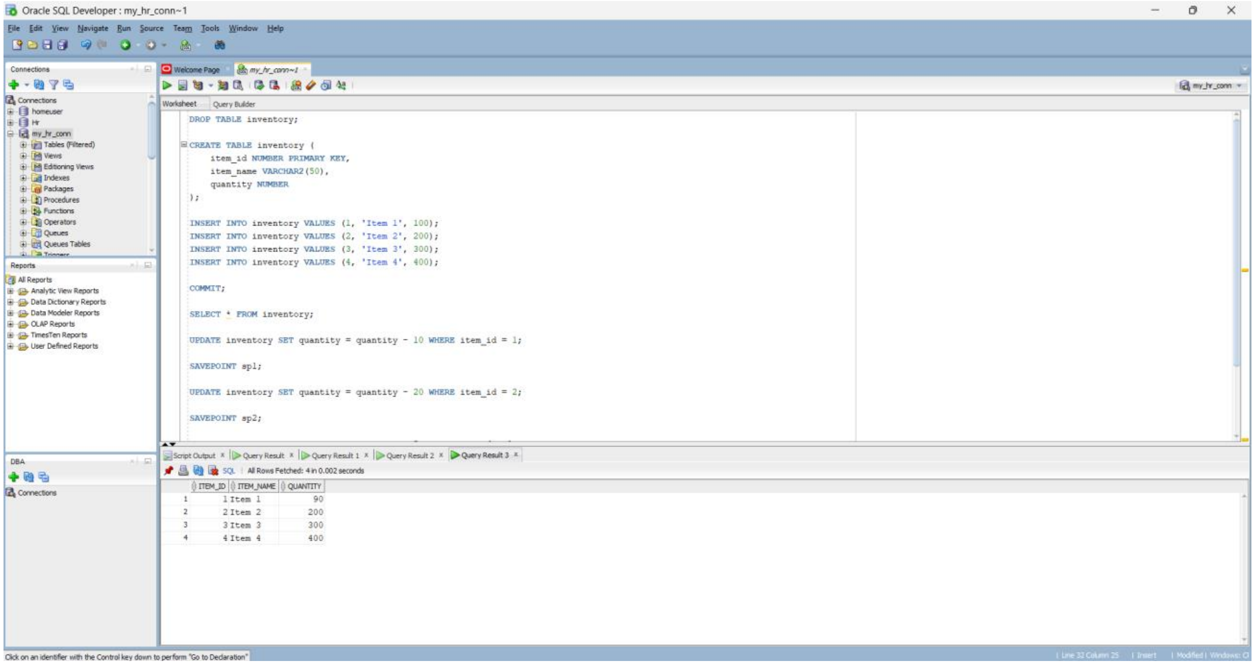


```
    quantity NUMBER
);

INSERT INTO inventory VALUES (1, 'Item 1', 100);
INSERT INTO inventory VALUES (2, 'Item 2', 200);
INSERT INTO inventory VALUES (3, 'Item 3', 300);
INSERT INTO inventory VALUES (4, 'Item 4', 400);

COMMIT;
```

Task completed in 0.43 seconds

```
Table INVENTORY created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


Commit complete.

>>Query Run In:Query Result 2

1 row updated.


Savepoint created.


1 row updated.


Savepoint created.
```

2b)

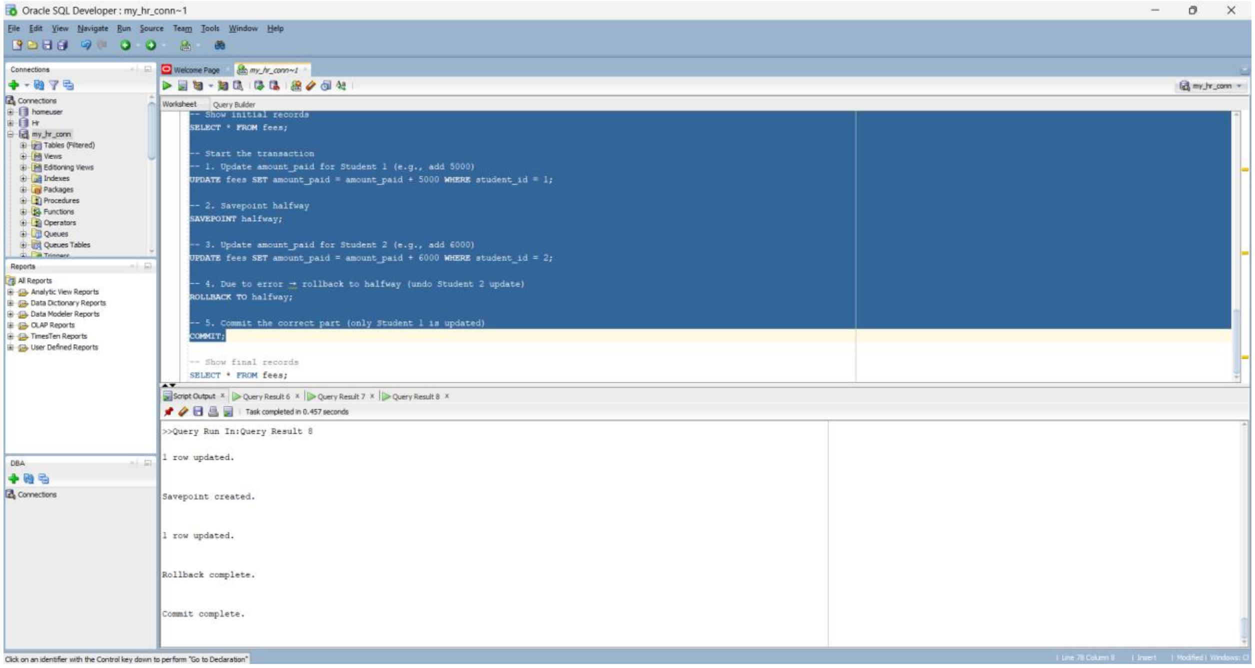


```sql
DROP TABLE inventory;

CREATE TABLE inventory (
    item_id NUMBER PRIMARY KEY,
    item_name VARCHAR2(50),
    quantity NUMBER
);

INSERT INTO inventory VALUES (1, 'Item 1', 100);
INSERT INTO inventory VALUES (2, 'Item 2', 200);
INSERT INTO inventory VALUES (3, 'Item 3', 300);
INSERT INTO inventory VALUES (4, 'Item 4', 400);

COMMIT;

SELECT * FROM inventory;

UPDATE inventory SET quantity = quantity - 10 WHERE item_id = 1;

SAVEPOINT sp1;

UPDATE inventory SET quantity = quantity - 20 WHERE item_id = 2;

SAVEPOINT sp2;
```

All Rows Fetched: 4 in 0.002 seconds

	ITEM_ID	ITEM_NAME	QUANTITY
1	1	Item 1	90
2	2	Item 2	200
3	3	Item 3	300
4	4	Item 4	400

3a)

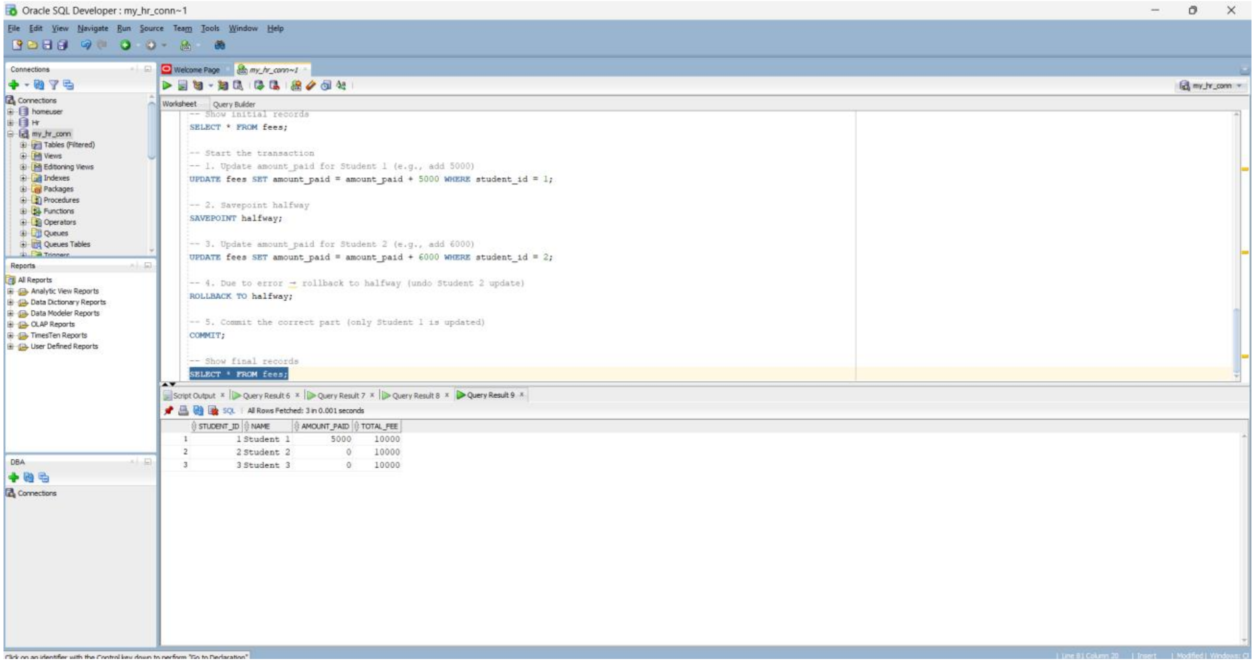


```sql
-- Show initial records
SELECT * FROM fees;

-- Start the transaction
-- 1. Update amount_paid for Student 1 (e.g., add 5000)
UPDATE fees SET amount_paid = amount_paid + 5000 WHERE student_id = 1;

-- 2. Savepoint halfway
SAVEPOINT halfway;

-- 3. Update amount_paid for Student 2 (e.g., add 6000)
UPDATE fees SET amount_paid = amount_paid + 6000 WHERE student_id = 2;

-- 4. Due to error → rollback to halfway (undo Student 2 update)
ROLLBACK TO halfway;

-- 5. Commit the correct part (only Student 1 is updated)
COMMIT;

-- Show final records
SELECT * FROM fees;
```

Task completed in 0.457 seconds

```
>>Query Run In:Query Result 8

1 row updated.


Savepoint created.


1 row updated.


Rollback complete.


Commit complete.
```

3b)

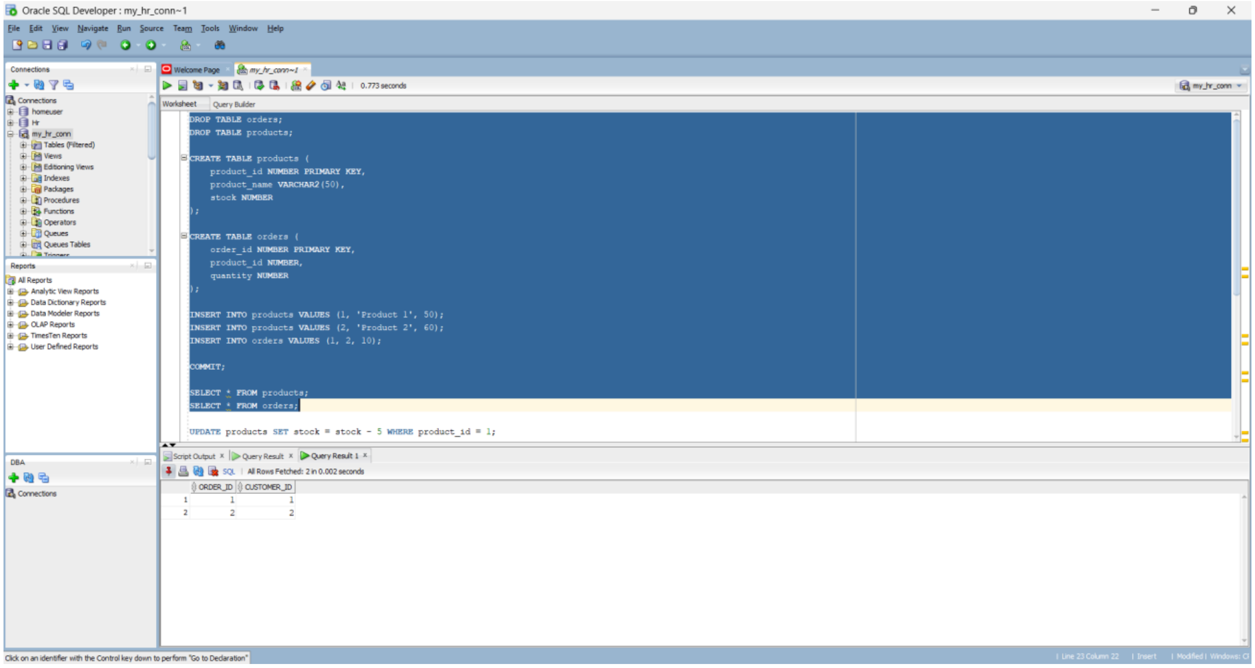


```sql
-- Show initial records
SELECT * FROM fees;

-- Start the transaction
-- 1. Update amount_paid for Student 1 (e.g., add 5000)
UPDATE fees SET amount_paid = amount_paid + 5000 WHERE student_id = 1;

-- 2. Savepoint halfway
SAVEPOINT halfway;

-- 3. Update amount_paid for Student 2 (e.g., add 6000)
UPDATE fees SET amount_paid = amount_paid + 6000 WHERE student_id = 2;

-- 4. Due to error → rollback to halfway (undo Student 2 update)
ROLLBACK TO halfway;

-- 5. Commit the correct part (only Student 1 is updated)
COMMIT;

-- Show final records
SELECT * FROM fees;
```

All Rows Fetched: 3 in 0.001 seconds

	STUDENT_ID	NAME	AMOUNT_PAID	TOTAL_FEE
1	1	Student 1	5000	10000
2	2	Student 2	0	10000
3	3	Student 3	0	10000

4a)

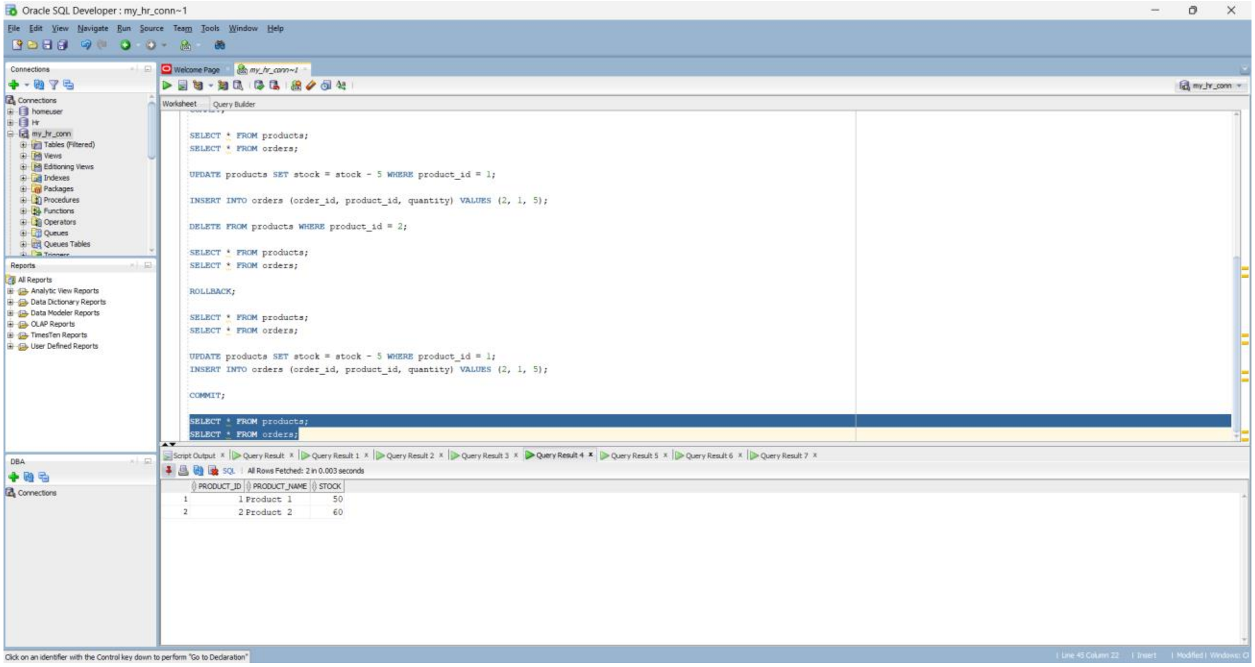


```sql
DROP TABLE orders;
DROP TABLE products;

CREATE TABLE products (
    product_id NUMBER PRIMARY KEY,
    product_name VARCHAR2(50),
    stock NUMBER
);

CREATE TABLE orders (
    order_id NUMBER PRIMARY KEY,
    product_id NUMBER,
    quantity NUMBER
);

INSERT INTO products VALUES (1, 'Product 1', 50);
INSERT INTO products VALUES (2, 'Product 2', 60);
INSERT INTO orders VALUES (1, 2, 10);

COMMIT;

SELECT * FROM products;
SELECT * FROM orders;

UPDATE products SET stock = stock - 5 WHERE product_id = 1;
```

All Rows Fetched: 2 in 0.002 seconds

	ORDER_ID	CUSTOMER_ID
1	1	1
2	2	2

4b)



Oracle SQL Developer : my_hr_conn~1

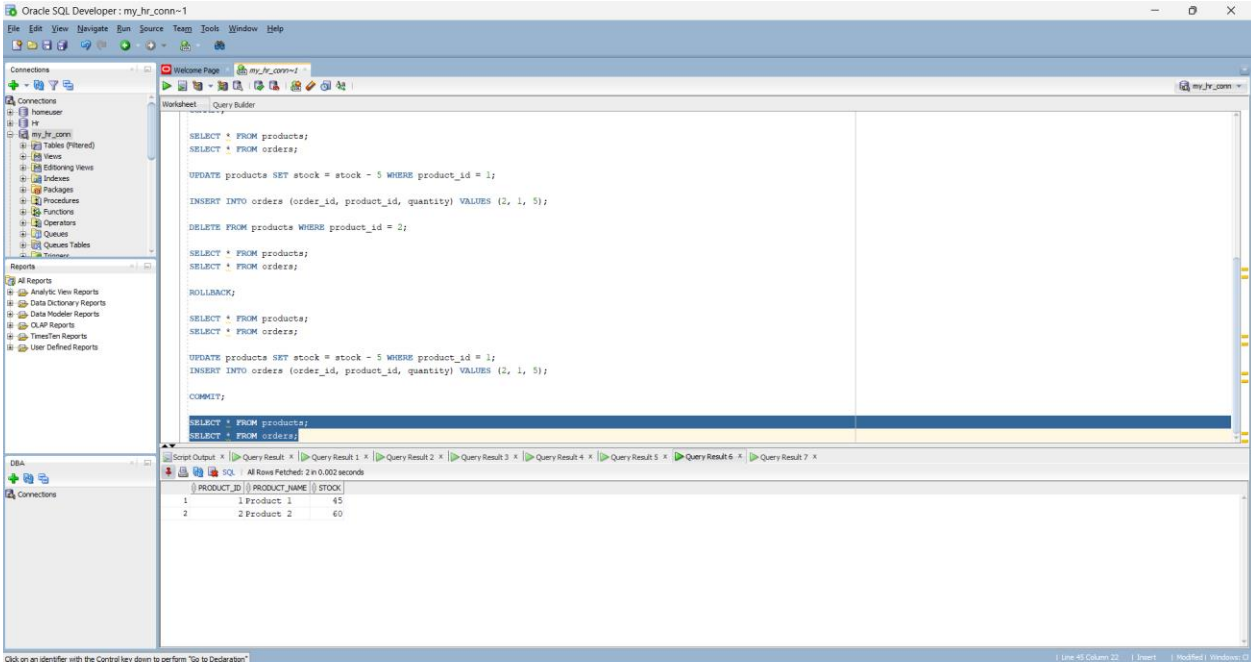
```
SELECT * FROM products;
SELECT * FROM orders;

UPDATE products SET stock = stock - 5 WHERE product_id = 1;

INSERT INTO orders (order_id, product_id, quantity) VALUES (2, 1, 5);

DELETE FROM products WHERE product_id = 2;

SELECT * FROM products;
SELECT * FROM orders;

ROLLBACK;

SELECT * FROM products;
SELECT * FROM orders;

UPDATE products SET stock = stock - 5 WHERE product_id = 1;
INSERT INTO orders (order_id, product_id, quantity) VALUES (2, 1, 5);

COMMIT;

SELECT * FROM products;
SELECT * FROM orders;
```

All Rows Fetched: 2 in 0.003 seconds

	PRODUCT_ID	PRODUCT_NAME	STOCK
1	1	Product 1	50
2	2	Product 2	60

Click on an identifier with the Control key down to perform "Go to Declaration"

4c)



Oracle SQL Developer : my_hr_conn~1

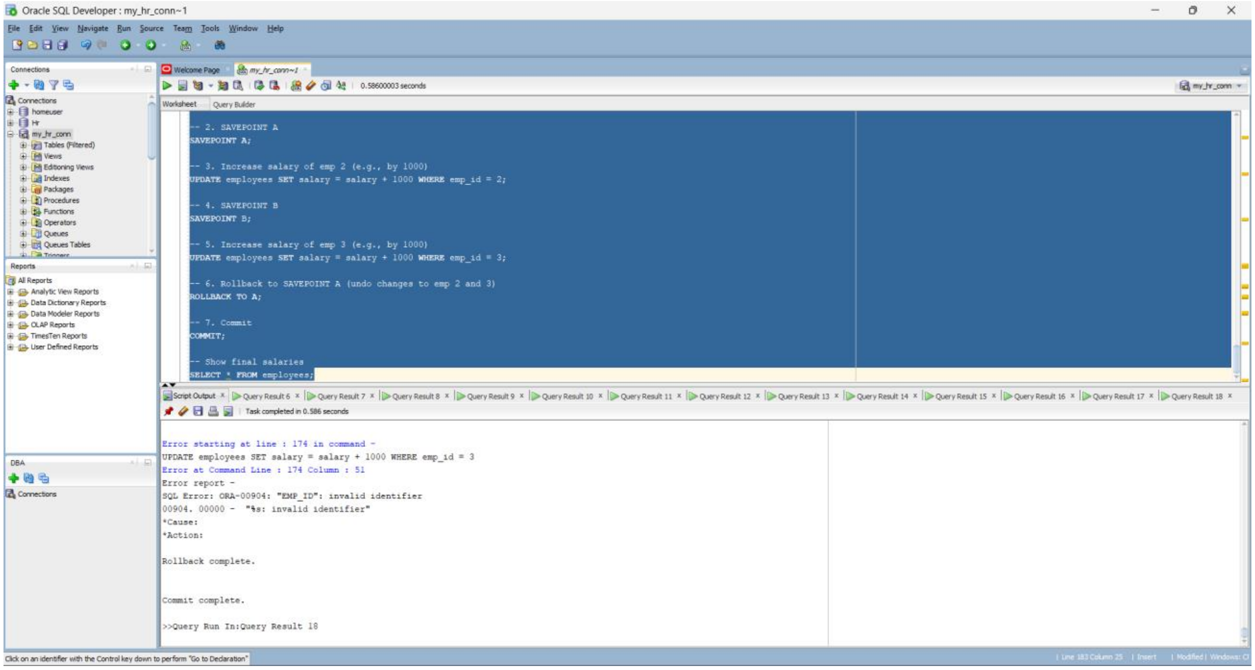
```
SELECT * FROM products;
SELECT * FROM orders;

UPDATE products SET stock = stock - 5 WHERE product_id = 1;

INSERT INTO orders (order_id, product_id, quantity) VALUES (2, 1, 5);

DELETE FROM products WHERE product_id = 2;

SELECT * FROM products;
SELECT * FROM orders;

ROLLBACK;

SELECT * FROM products;
SELECT * FROM orders;

UPDATE products SET stock = stock - 5 WHERE product_id = 1;
INSERT INTO orders (order_id, product_id, quantity) VALUES (2, 1, 5);

COMMIT;

SELECT * FROM products;
SELECT * FROM orders;
```

All Rows Fetched: 2 in 0.002 seconds

	PRODUCT_ID	PRODUCT_NAME	STOCK
1	1	Product 1	45
2	2	Product 2	60

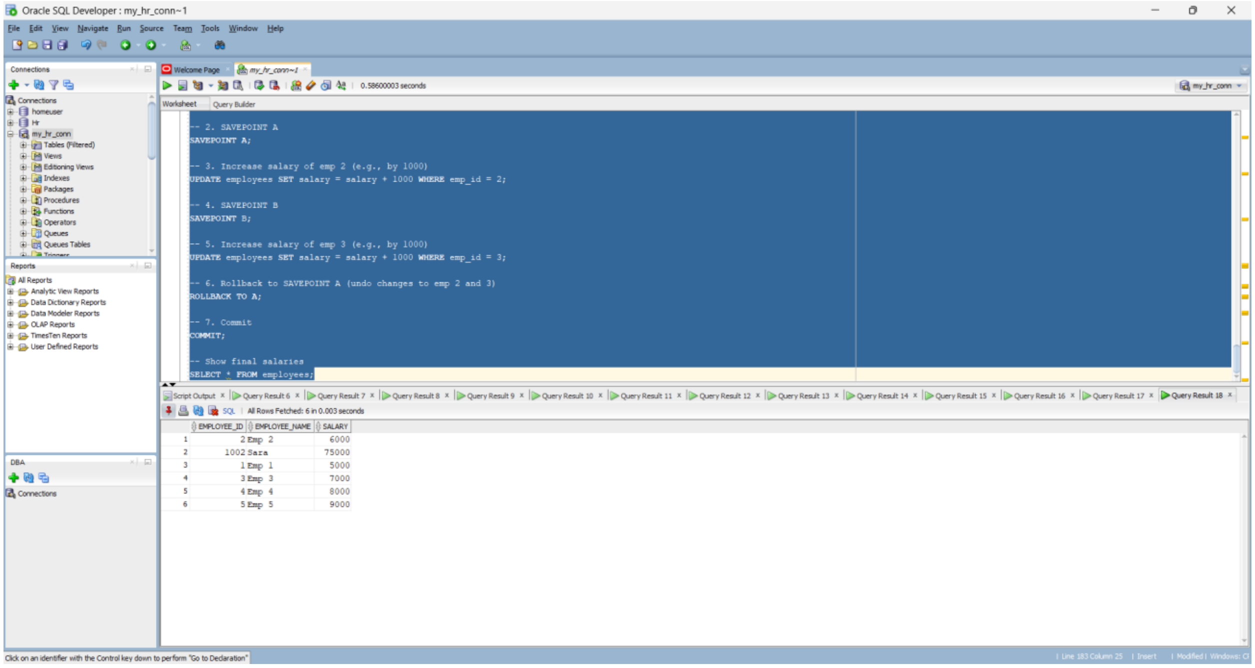
Click on an identifier with the Control key down to perform "Go to Declaration"

5a and b)



```
-- 2. SAVEPOINT A
SAVEPOINT A;

-- 3. Increase salary of emp 2 (e.g., by 1000)
UPDATE employees SET salary = salary + 1000 WHERE emp_id = 2;

-- 4. SAVEPOINT B
SAVEPOINT B;

-- 5. Increase salary of emp 3 (e.g., by 1000)
UPDATE employees SET salary = salary + 1000 WHERE emp_id = 3;

-- 6. Rollback to SAVEPOINT A (undo changes to emp 2 and 3)
ROLLBACK TO A;

-- 7. Commit
COMMIT;

-- Show final salaries
SELECT * FROM employees;
```

```
Error starting at line : 174 in command -
UPDATE employees SET salary = salary + 1000 WHERE emp_id = 3
Error at Command Line : 174 Column : 51
Error report -
SQL Error: ORA-00904: "EMP_ID": invalid identifier
00904. 00000 -  "%s: invalid identifier"
*Cause:
*Action:

Rollback complete.


Commit complete.

>>Query Run In:Query Result 18
```


Oracle SQL Developer : my_hr_conn~1
File Edit View Navigate Run Source Team Tools Window Help
Welcome Page
my_hr_conn~1
0.58600003 seconds
Worksheet
Query Builder
Query Result 18
All Rows Fetched: 6 in 0.003 seconds

```
-- 2. SAVEPOINT A
SAVEPOINT A;

-- 3. Increase salary of emp 2 (e.g., by 1000)
UPDATE employees SET salary = salary + 1000 WHERE emp_id = 2;

-- 4. SAVEPOINT B
SAVEPOINT B;

-- 5. Increase salary of emp 3 (e.g., by 1000)
UPDATE employees SET salary = salary + 1000 WHERE emp_id = 3;

-- 6. Rollback to SAVEPOINT A (undo changes to emp 2 and 3)
ROLLBACK TO A;

-- 7. Commit
COMMIT;

-- Show final salaries
SELECT * FROM employees;
```

	EMPLOYEE_ID	EMPLOYEE_NAME	SALARY
1	2	Emp 2	6000
2	1002	Sara	75000
3	1	Emp 1	5000
4	3	Emp 3	7000
5	4	Emp 4	8000
6	5	Emp 5	9000