Name: Urooba Gohar

Roll No: 22P-9216

Secion: BSCS-6A

Database Systems Labtask 10

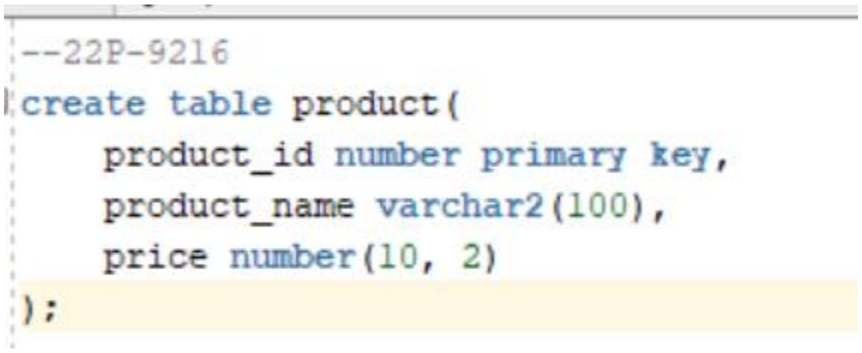
Class Task:

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.

Answer:

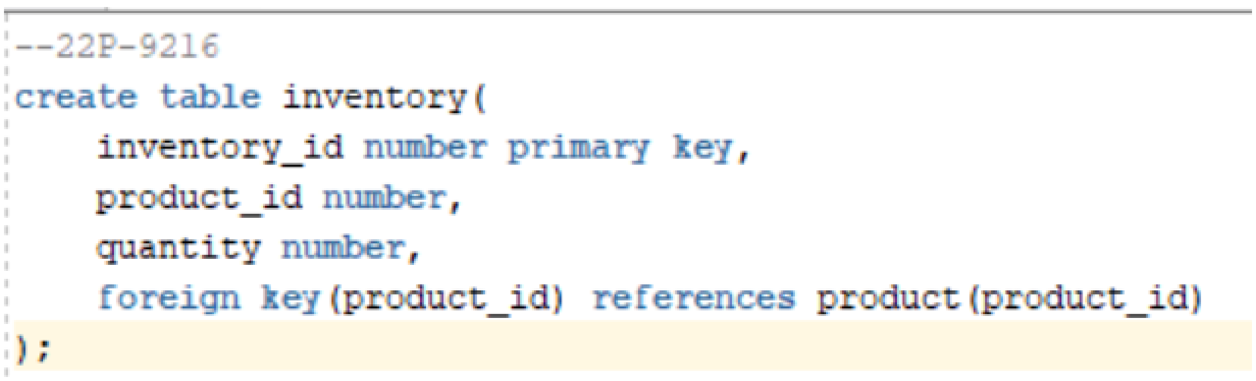
First we create product table:



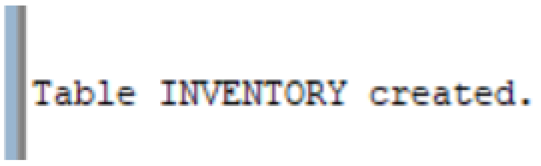
```
--22P-9216
create table product(
    product_id number primary key,
    product_name varchar2(100),
    price number(10, 2)
);
```

```
Table PRODUCT created.
```

Now create inventory table:

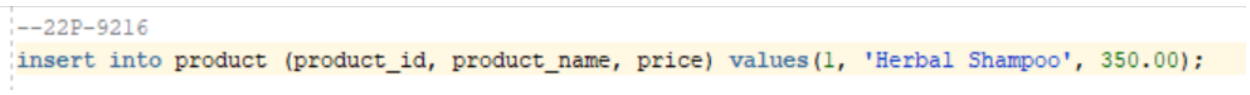


```
--22P-9216
create table inventory(
    inventory_id number primary key,
    product_id number,
    quantity number,
    foreign key(product_id) references product(product_id)
);
```



```
Table INVENTORY created.
```

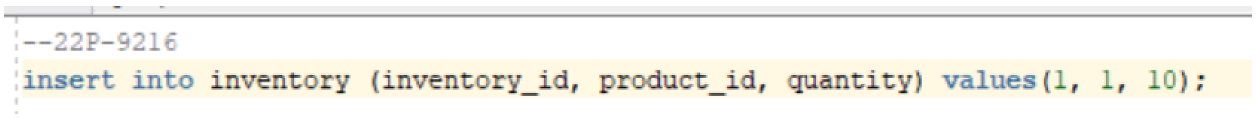
Now we insert a product:



```
--22P-9216
insert into product (product_id, product_name, price) values(1, 'Herbal Shampoo', 350.00);
```

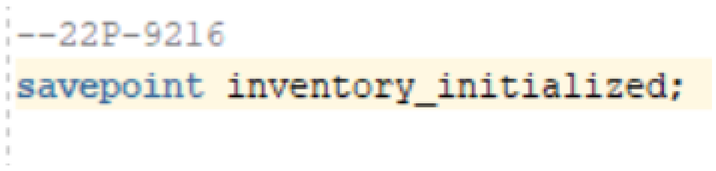
```
1 row inserted.
```

Insert initial inventory and set savepoint:

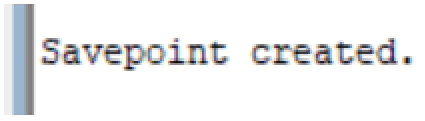


```
--22P-9216
insert into inventory (inventory_id, product_id, quantity) values(1, 1, 10);
```

```
1 row inserted.
```

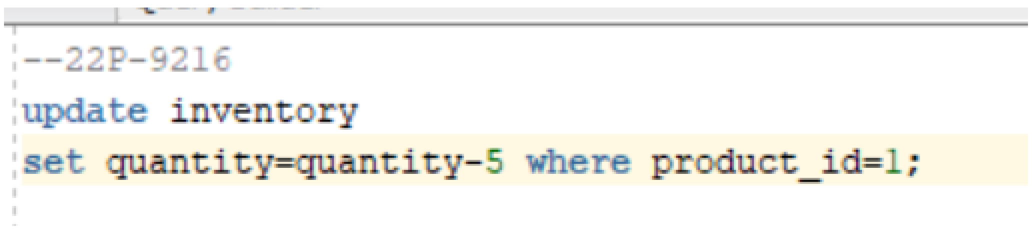


```
--22P-9216
savepoint inventory_initialized;
```

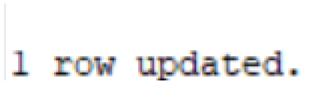


```
Savepoint created.
```

Simulate order:

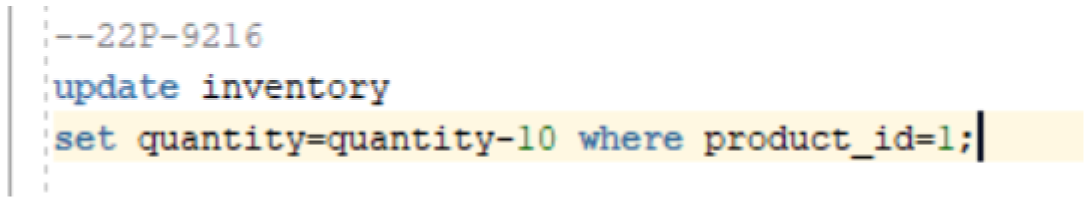


```
--22P-9216
update inventory
set quantity=quantity-5 where product_id=1;
```

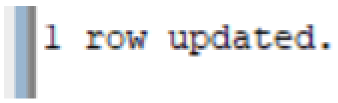


```
1 row updated.
```

Simulate another order that would cause negative stock:

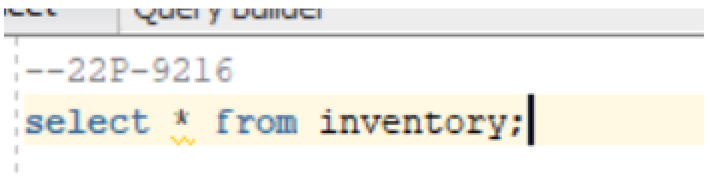


```
--22P-9216
update inventory
set quantity=quantity-10 where product_id=1;
```

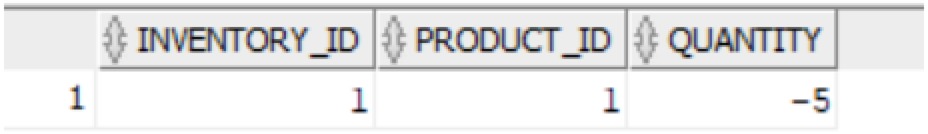


```
1 row updated.
```

Check quantity:

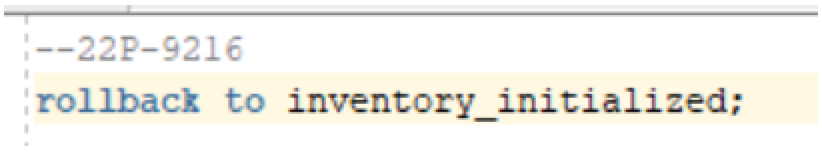


```
--22P-9216
select * from inventory;
```

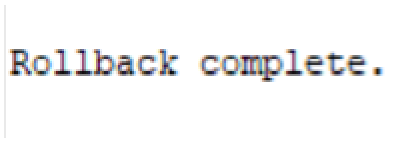


	INVENTORY_ID	PRODUCT_ID	QUANTITY
1	1	1	-5

If negative, rollback to safe point:

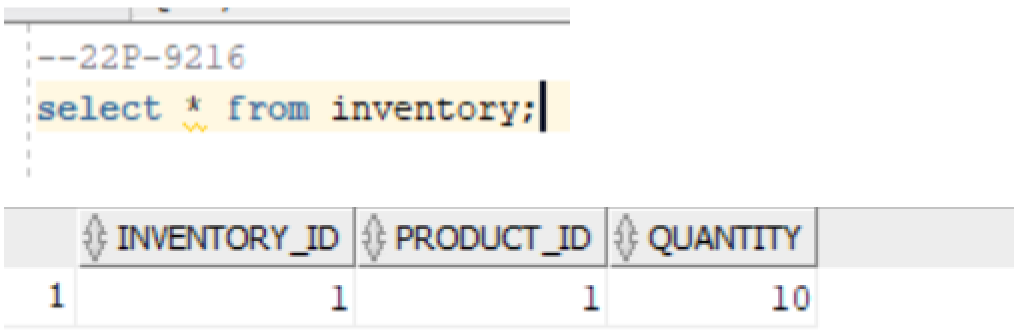


```
--22P-9216
rollback to inventory_initialized;
```



```
Rollback complete.
```

Verify rollback:



```
--22P-9216
select * from inventory;
```

	INVENTORY_ID	PRODUCT_ID	QUANTITY
1	1	1	10

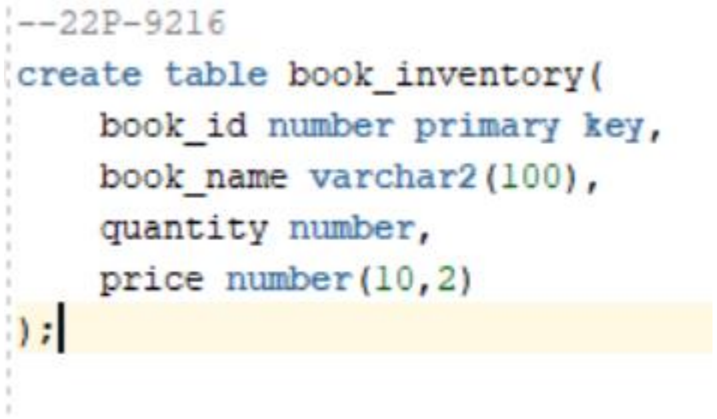
____

Task 1:

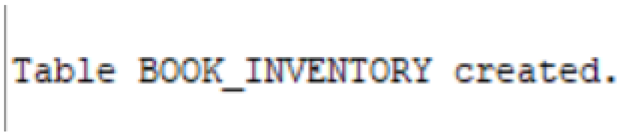
Create a new table named book_inventory with columns for book_id, book_name, quantity, and price. Insert three different book records with initial quantities. Without committing the transaction, reduce the quantity of one book and create a savepoint named quantity_update.

Answer:

Create the book_inventory table:

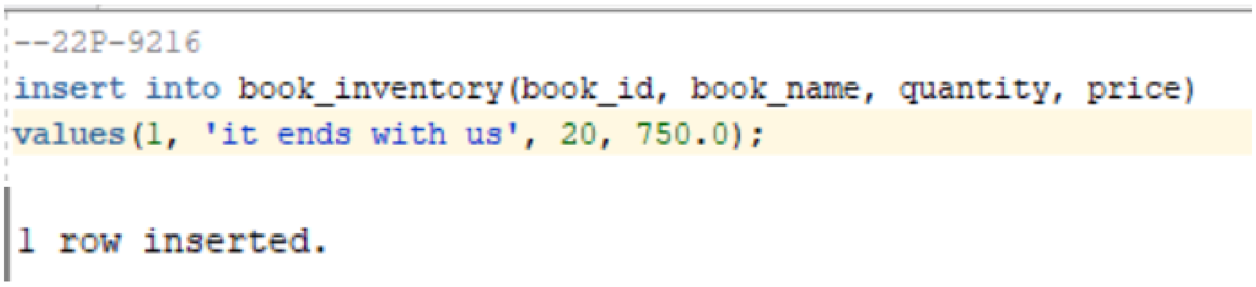


```
--22P-9216
create table book_inventory(
    book_id number primary key,
    book_name varchar2(100),
    quantity number,
    price number(10,2)
);
```



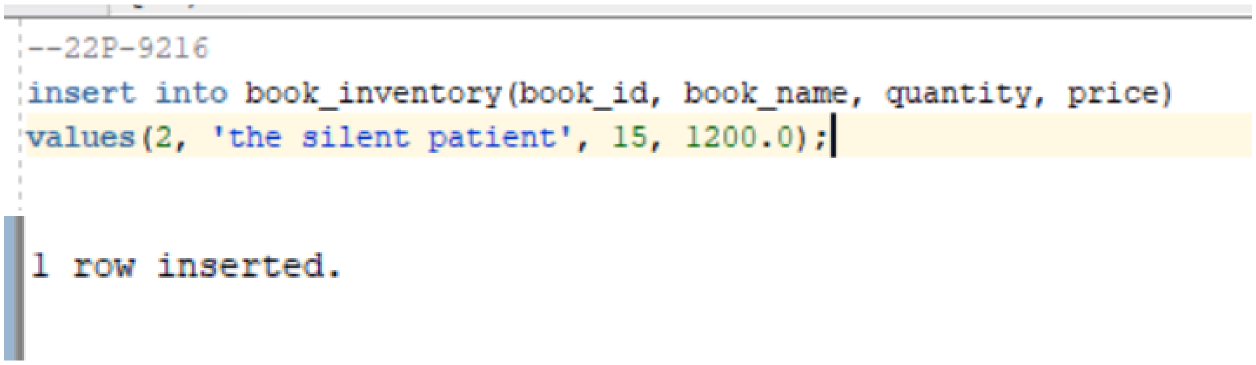
```
Table BOOK_INVENTORY created.
```

Insert three books with initial quantities:



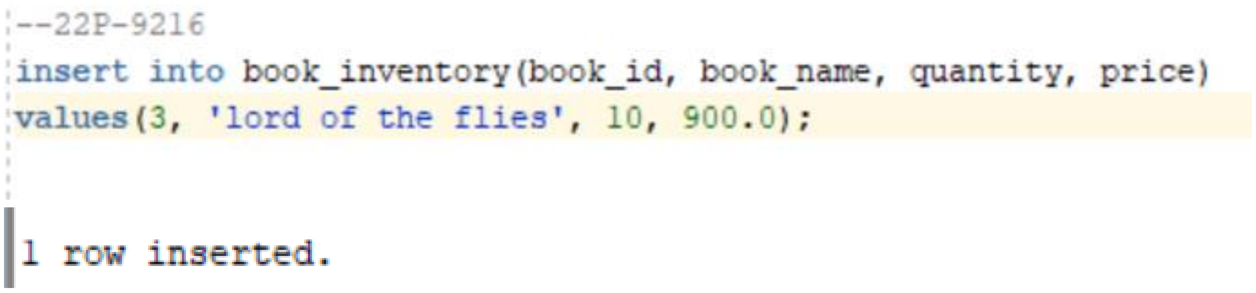
```
--22P-9216
insert into book_inventory(book_id, book_name, quantity, price)
values(1, 'it ends with us', 20, 750.0);
```

```
1 row inserted.
```



```
--22P-9216
insert into book_inventory(book_id, book_name, quantity, price)
values(2, 'the silent patient', 15, 1200.0);
```

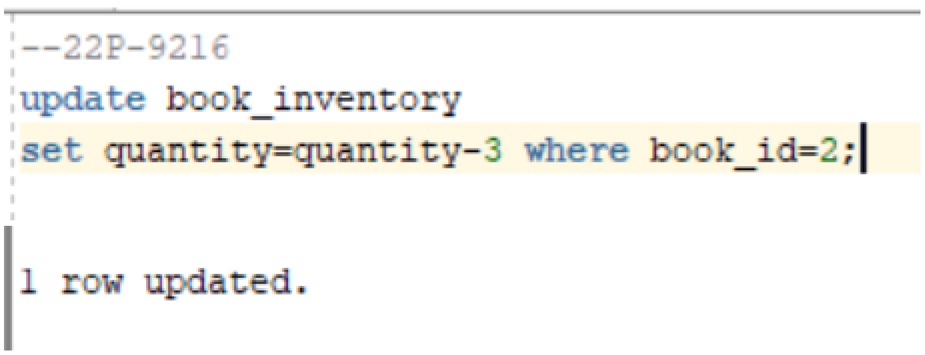
```
1 row inserted.
```



```
--22P-9216
insert into book_inventory(book_id, book_name, quantity, price)
values(3, 'lord of the flies', 10, 900.0);
```

```
1 row inserted.
```

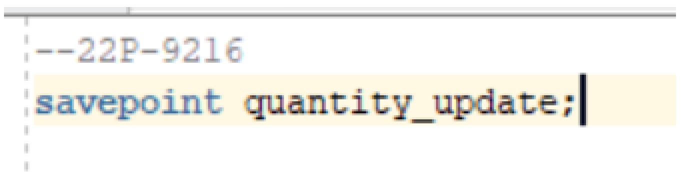
Reduce the quantity of book 2:



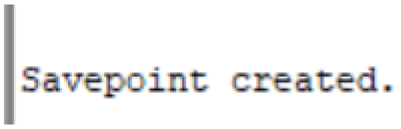
```
--22P-9216
update book_inventory
set quantity=quantity-3 where book_id=2;
```

```
1 row updated.
```

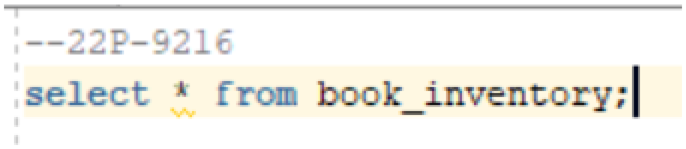
Create a savepoint named quantity_update:

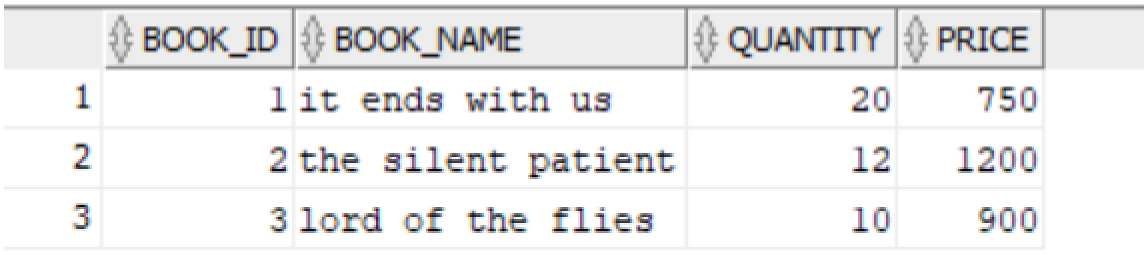


```
--22P-9216
savepoint quantity_update;
```


```
Savepoint created.
```

Verify the change before committing:


```
--22P-9216
select * from book_inventory;
```



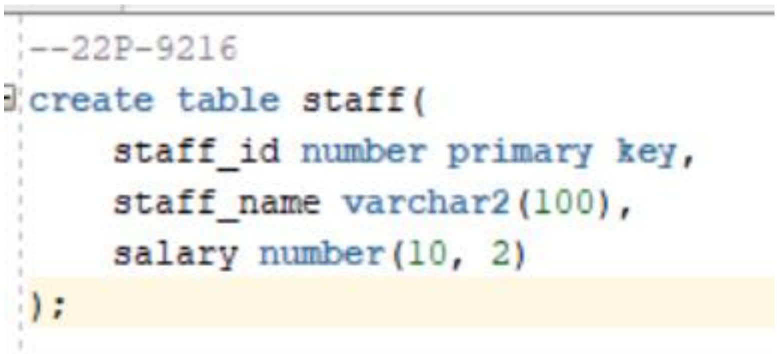
	BOOK_ID	BOOK_NAME	QUANTITY	PRICE
1	1	it ends with us	20	750
2	2	the silent patient	12	1200
3	3	lord of the flies	10	900

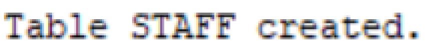
Task 2:

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.

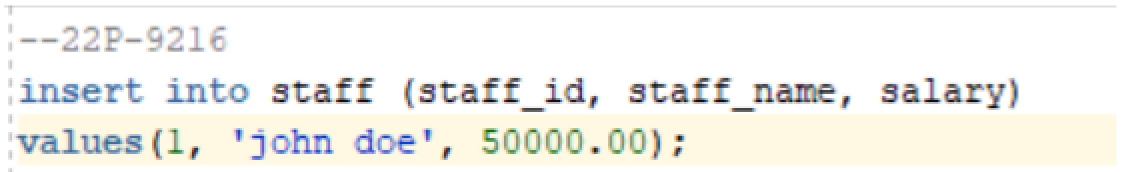
Answer:

Create the staff table:

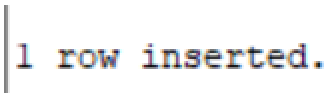

```
--22P-9216
create table staff(
    staff_id number primary key,
    staff_name varchar2(100),
    salary number(10, 2)
);
```


```
Table STAFF created.
```

Insert a new staff member with initial salary:

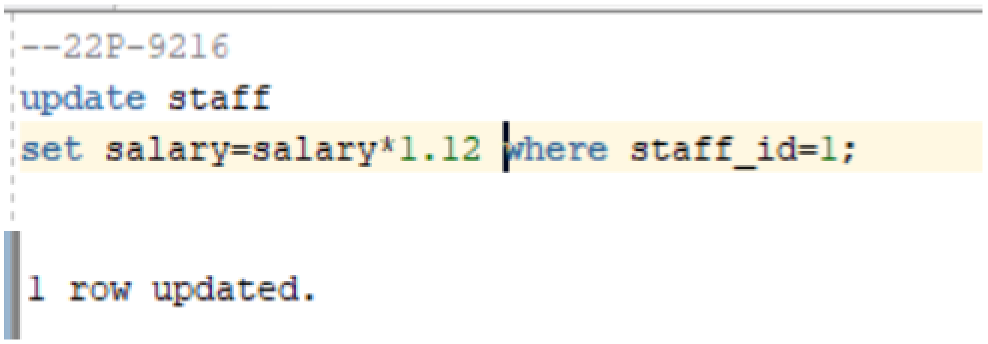


```
--22P-9216
insert into staff (staff_id, staff_name, salary)
values(1, 'john doe', 50000.00);
```



```
1 row inserted.
```

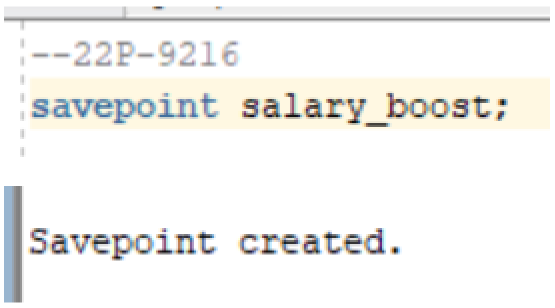
Increase the salary by 12%:



```
--22P-9216
update staff
set salary=salary*1.12 where staff_id=1;
```

```
1 row updated.
```

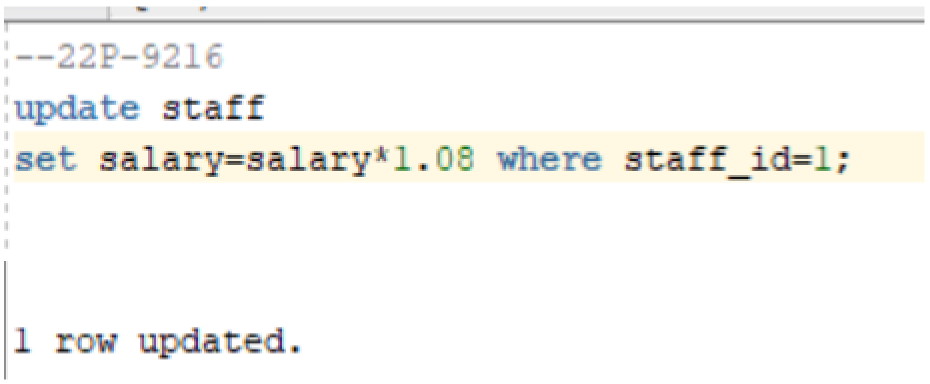
Create a savepoint after the 12% increase:



```
--22P-9216
savepoint salary_boost;
```

```
Savepoint created.
```

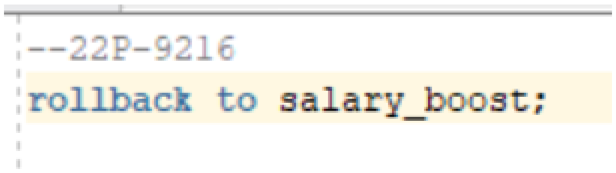
Further increase the salary by 8%:



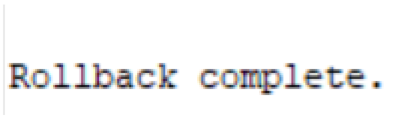
```
--22P-9216
update staff
set salary=salary*1.08 where staff_id=1;
```

```
1 row updated.
```

Rollback to the savepoint to undo the 8% increase:

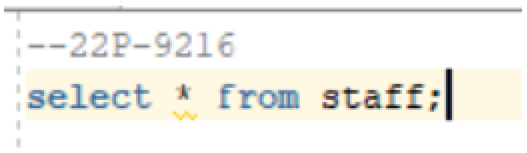


```
--22P-9216
rollback to salary_boost;
```

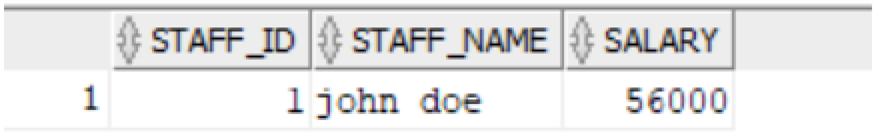


```
Rollback complete.
```

Verify the result:



```
--22P-9216
select * from staff;
```



	STAFF_ID	STAFF_NAME	SALARY
1	1	john doe	56000

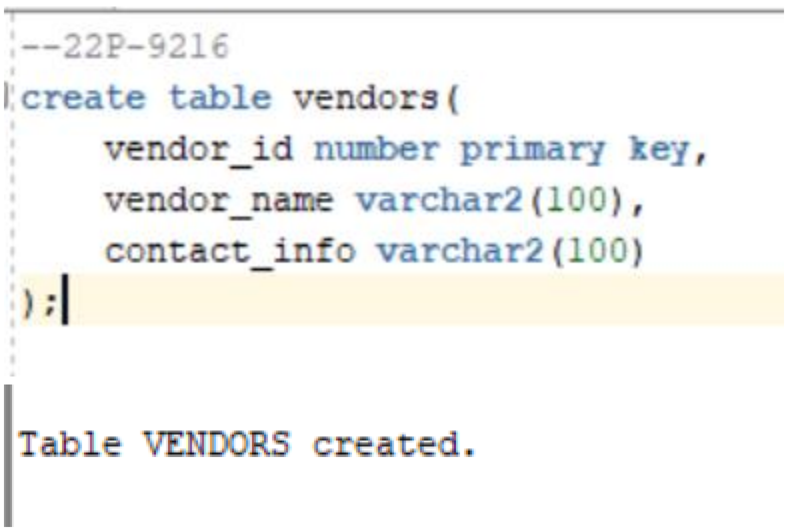
____

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.

Answer:

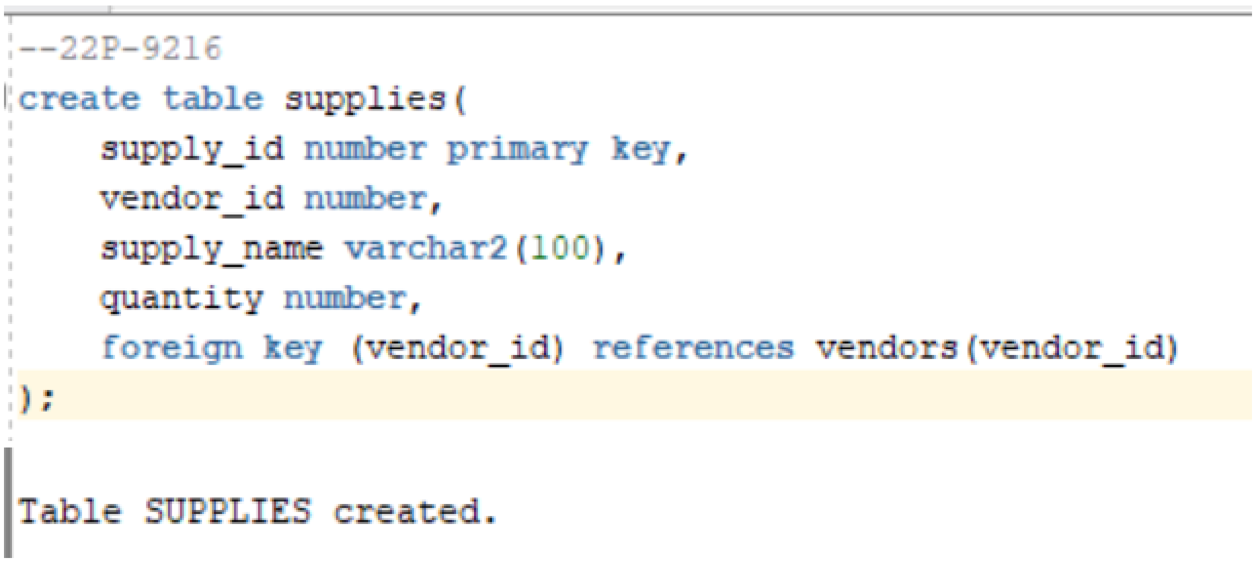
Create the vendors table:



```
--22P-9216
create table vendors(
    vendor_id number primary key,
    vendor_name varchar2(100),
    contact_info varchar2(100)
);
```

```
Table VENDORS created.
```

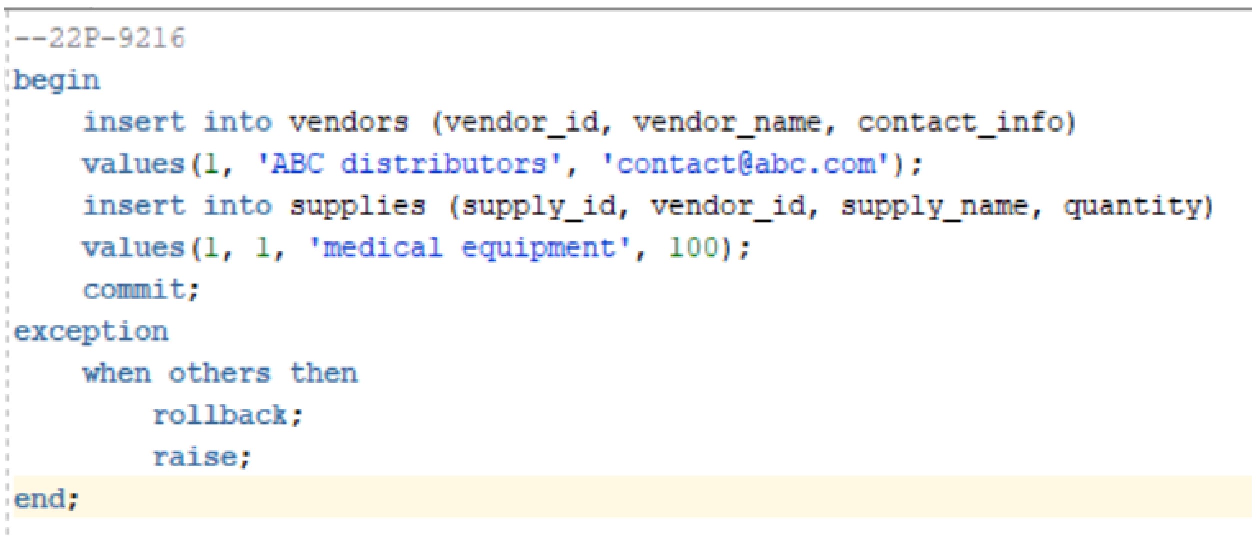
Create the supplies table:



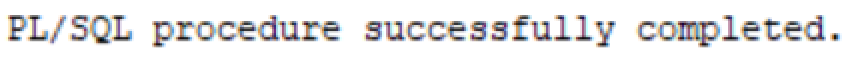
```
--22P-9216
create table supplies(
    supply_id number primary key,
    vendor_id number,
    supply_name varchar2(100),
    quantity number,
    foreign key (vendor_id) references vendors(vendor_id)
);
```

```
Table SUPPLIES created.
```

Begin the transaction by inserting a new vendor, a new supply record for this vendor, and committing the transaction if both insertions succeed:

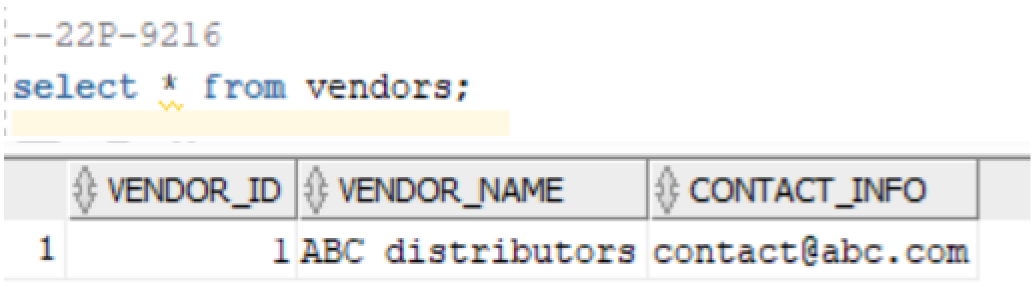


```
--22P-9216
begin
    insert into vendors (vendor_id, vendor_name, contact_info)
    values(1, 'ABC distributors', 'contact@abc.com');
    insert into supplies (supply_id, vendor_id, supply_name, quantity)
    values(1, 1, 'medical equipment', 100);
    commit;
exception
    when others then
        rollback;
        raise;
end;
```



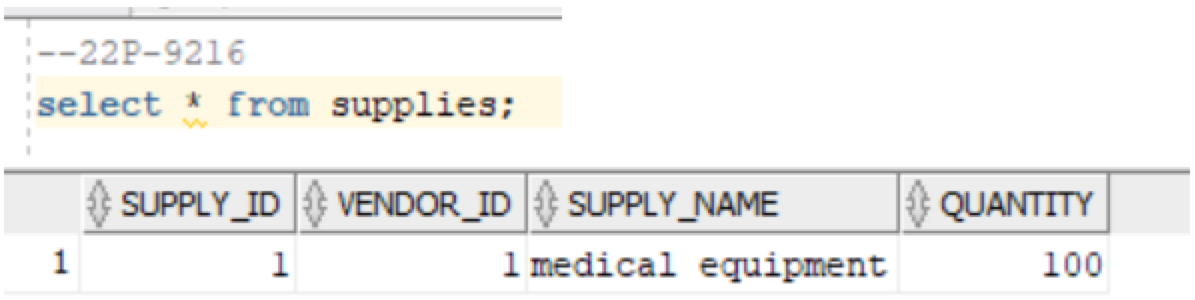
```
PL/SQL procedure successfully completed.
```

Verify the changes:



```
--22P-9216
select * from vendors;
```

	VENDOR_ID	VENDOR_NAME	CONTACT_INFO
1	1	ABC distributors	contact@abc.com



```
--22P-9216
select * from supplies;
```

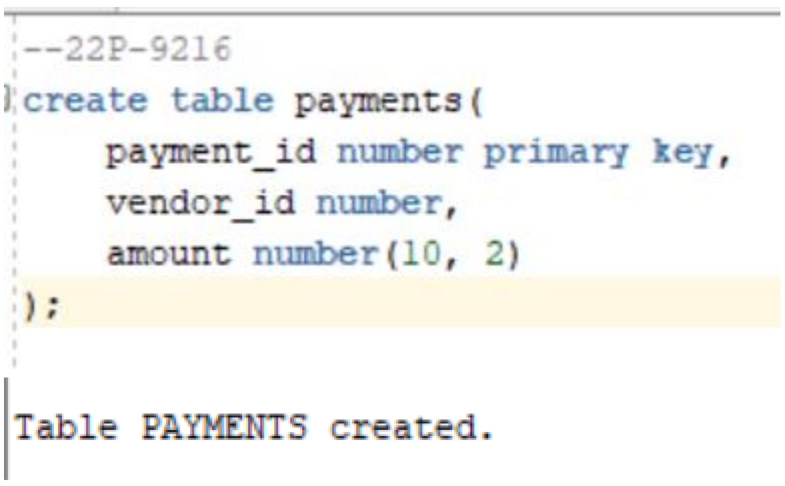
	SUPPLY_ID	VENDOR_ID	SUPPLY_NAME	QUANTITY
1	1	1	medical equipment	100

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.

Answer:

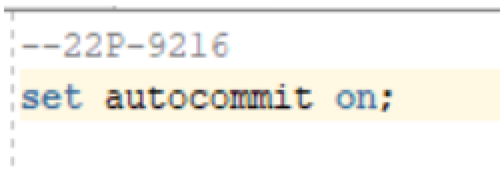
Create the payments table:



```
--22P-9216
create table payments(
    payment_id number primary key,
    vendor_id number,
    amount number(10, 2)
);
```

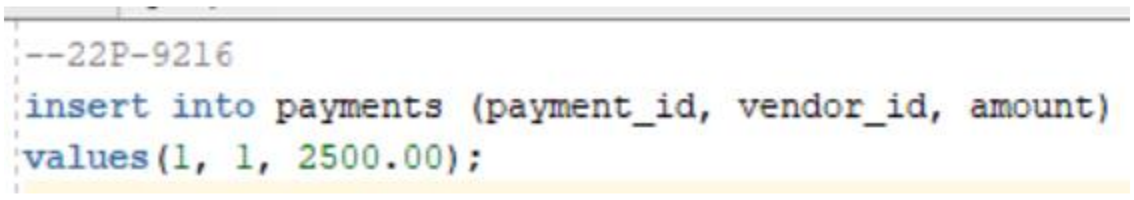
```
Table PAYMENTS created.
```

Enable autocommit mode:

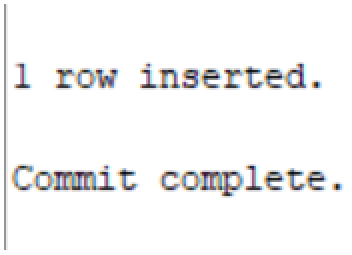


```
--22P-9216
set autocommit on;
```

Insert a payment record:



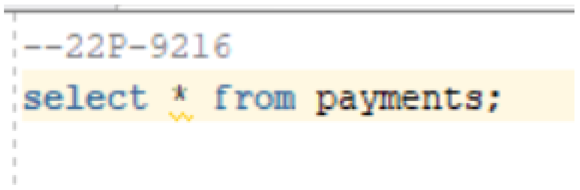
```
--22P-9216
insert into payments (payment_id, vendor_id, amount)
values(1, 1, 2500.00);
```



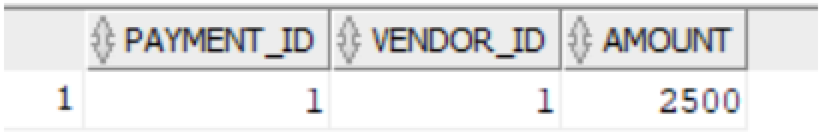
```
1 row inserted.

Commit complete.
```

Verify the insertion:



```
--22P-9216
select * from payments;
```



	PAYMENT_ID	VENDOR_ID	AMOUNT
1	1	1	2500

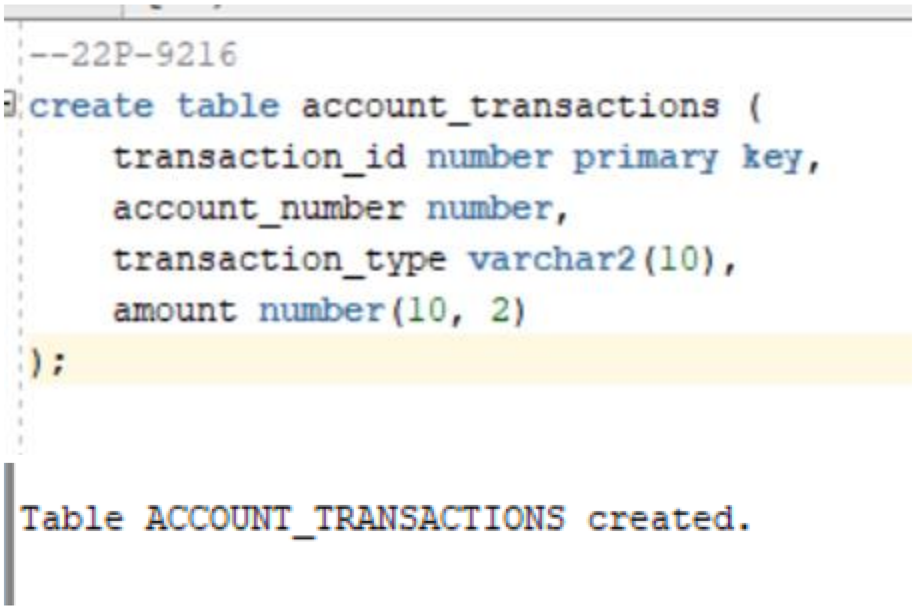
____

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..

Answer:

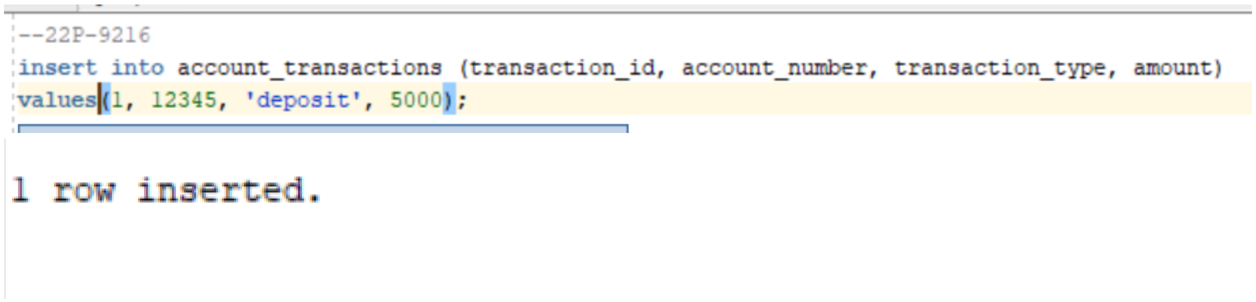
Create the account_transcations table:



```
--22P-9216
create table account_transactions (
    transaction_id number primary key,
    account_number number,
    transaction_type varchar2(10),
    amount number(10, 2)
);
```

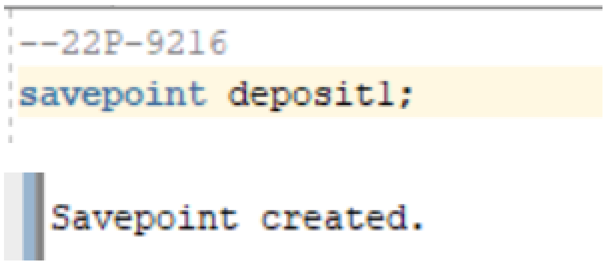
```
Table ACCOUNT_TRANSACTIONS created.
```

Begin a transaction and insert sample operations:



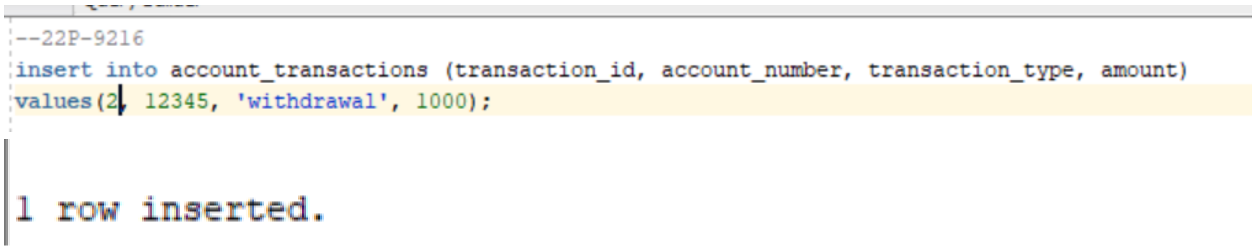
```
--22P-9216
insert into account_transactions (transaction_id, account_number, transaction_type, amount)
values(1, 12345, 'deposit', 5000);
```

```
1 row inserted.
```



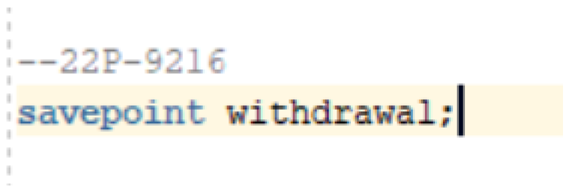
```
--22P-9216
savepoint deposit1;
```

```
Savepoint created.
```

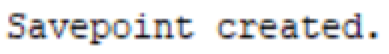


```
--22P-9216
insert into account_transactions (transaction_id, account_number, transaction_type, amount)
values(2, 12345, 'withdrawal', 1000);
```

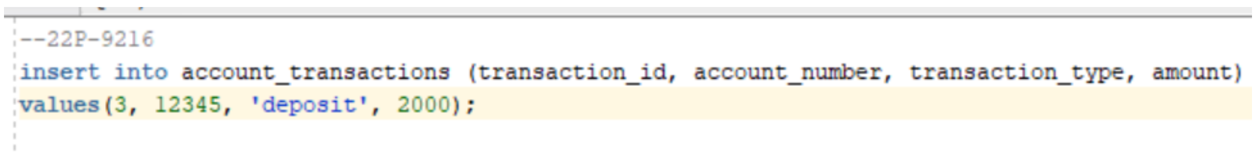
```
1 row inserted.
```



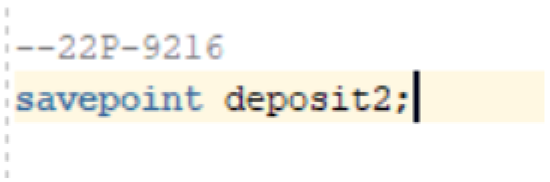
```
--22P-9216
savepoint withdrawal;
```



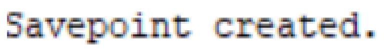
```
Savepoint created.
```



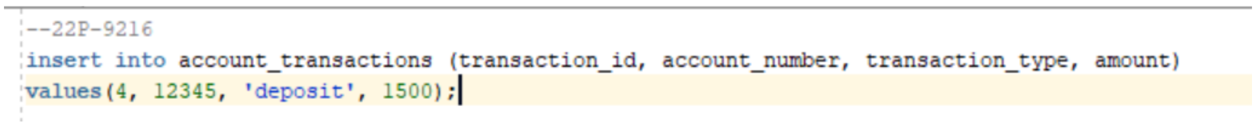
```
--22P-9216
insert into account_transactions (transaction_id, account_number, transaction_type, amount)
values(3, 12345, 'deposit', 2000);
```



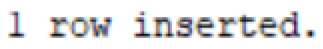
```
--22P-9216
savepoint deposit2;
```



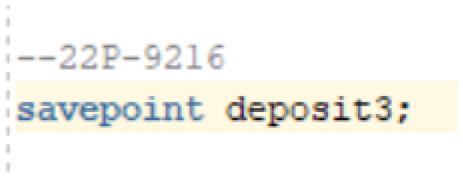
```
Savepoint created.
```



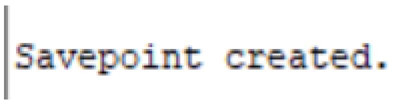
```
--22P-9216
insert into account_transactions (transaction_id, account_number, transaction_type, amount)
values(4, 12345, 'deposit', 1500);
```



```
1 row inserted.
```

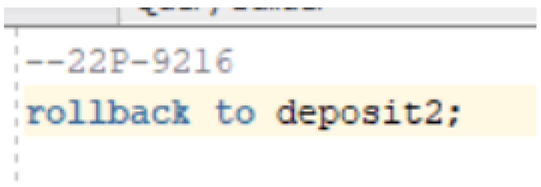


```
--22P-9216
savepoint deposit3;
```

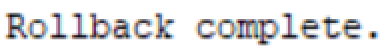


```
Savepoint created.
```

Rollback to deposit2:

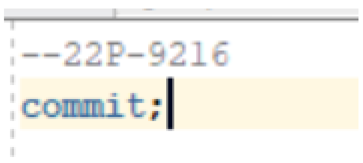


```
--22P-9216
rollback to deposit2;
```

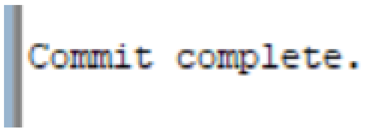


```
Rollback complete.
```

Commit the transaction:

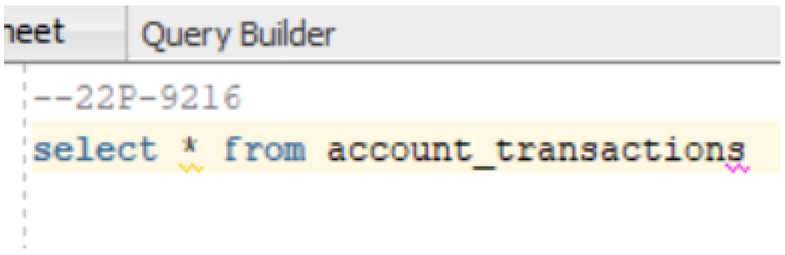


```
--22P-9216
commit;
```

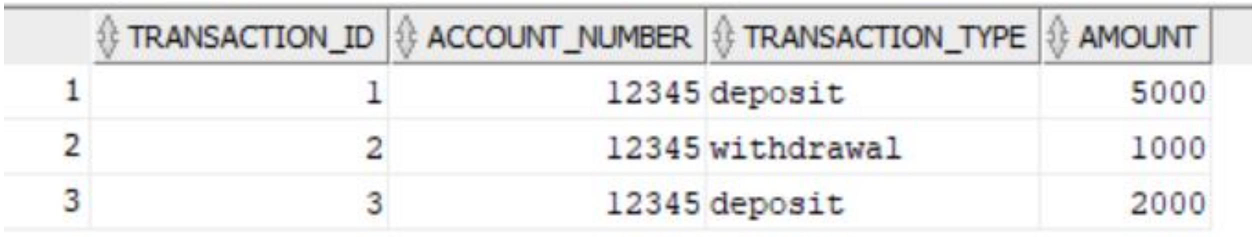


```
Commit complete.
```

Verify changes:



```
--22P-9216
select * from account_transactions
```



	TRANSACTION_ID	ACCOUNT_NUMBER	TRANSACTION_TYPE	AMOUNT
1	1	12345	deposit	5000
2	2	12345	withdrawal	1000
3	3	12345	deposit	2000

____