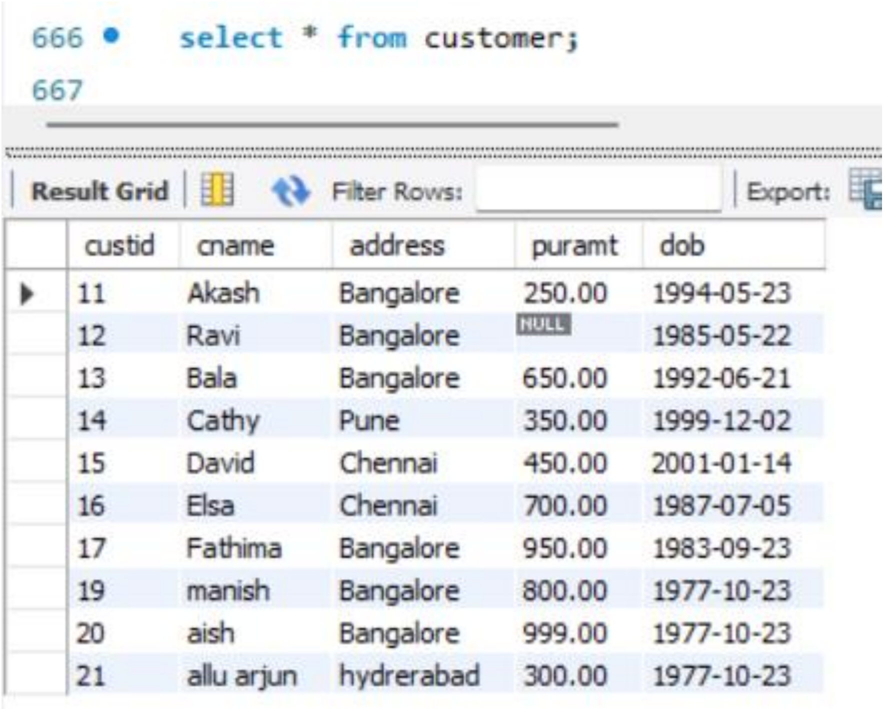
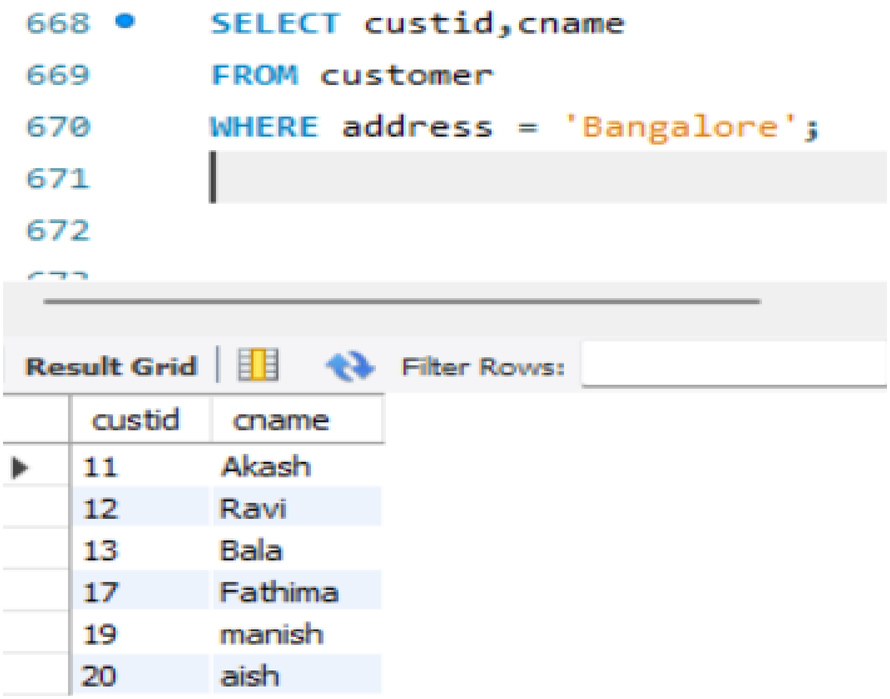
DQL ASSIGNMENT

Assignment 1: Write a SELECT query to retrieve all columns from a 'customers' table, and modify it to return only the customer name and email address for customers in a specific city.



```sql
select * from customer;
```

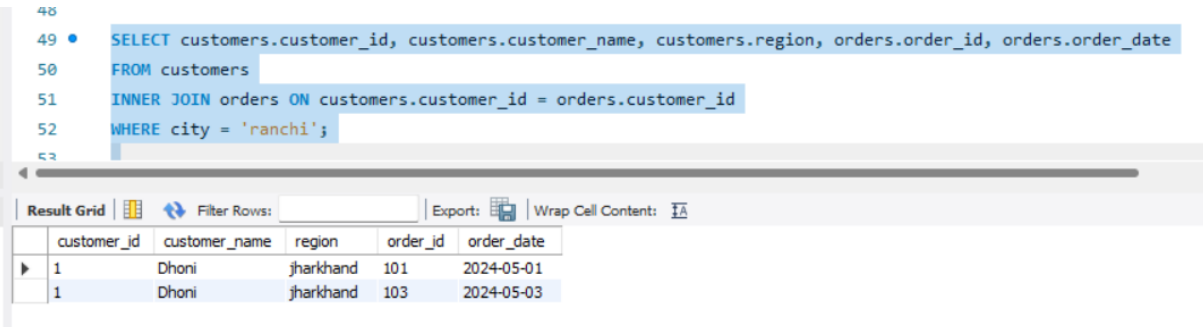
	custid	cname	address	puramt	dob
▶	11	Akash	Bangalore	250.00	1994-05-23
	12	Ravi	Bangalore	NULL	1985-05-22
	13	Bala	Bangalore	650.00	1992-06-21
	14	Cathy	Pune	350.00	1999-12-02
	15	David	Chennai	450.00	2001-01-14
	16	Elsa	Chennai	700.00	1987-07-05
	17	Fathima	Bangalore	950.00	1983-09-23
	19	manish	Bangalore	800.00	1977-10-23
	20	aish	Bangalore	999.00	1977-10-23
	21	allu arjun	hydrerabad	300.00	1977-10-23



```sql
SELECT custid,cname
FROM customer
WHERE address = 'Bangalore';
```

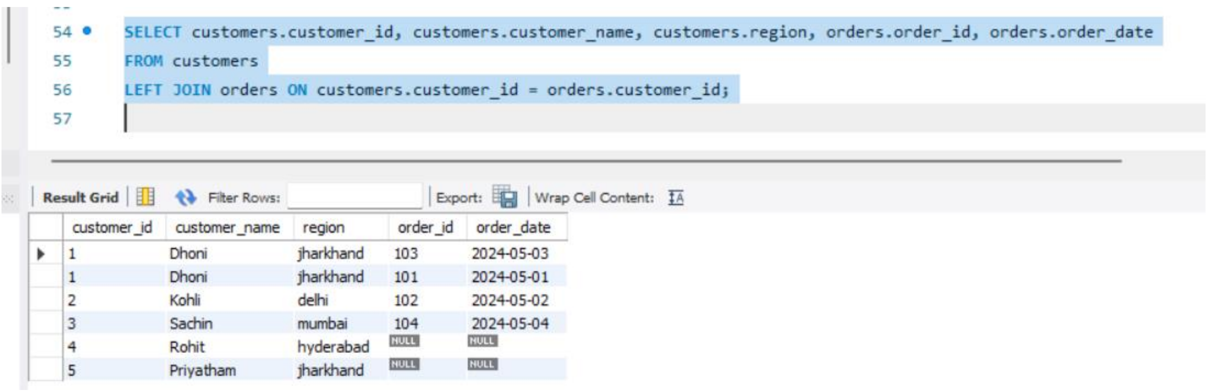
	custid	cname
▶	11	Akash
	12	Ravi
	13	Bala
	17	Fathima
	19	manish
	20	aish

Assignment 2: Craft a query using an INNER JOIN to combine 'orders' and 'customers' tables for customers in a specified region, and a LEFT JOIN to display all customers including those without orders.



```
SELECT customers.customer_id, customers.customer_name, customers.region, orders.order_id, orders.order_date
FROM customers
INNER JOIN orders ON customers.customer_id = orders.customer_id
WHERE city = 'ranchi';
```

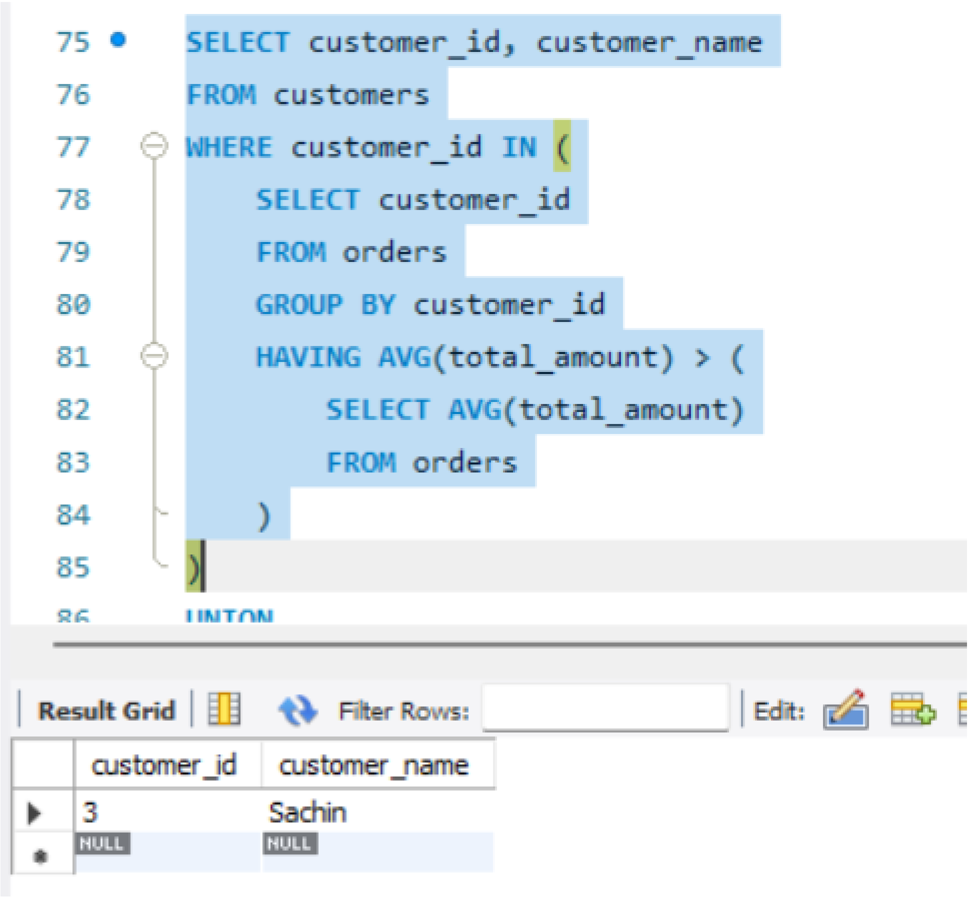
customer_id	customer_name	region	order_id	order_date
1	Dhoni	jharkhand	101	2024-05-01
1	Dhoni	jharkhand	103	2024-05-03



```
SELECT customers.customer_id, customers.customer_name, customers.region, orders.order_id, orders.order_date
FROM customers
LEFT JOIN orders ON customers.customer_id = orders.customer_id;
```

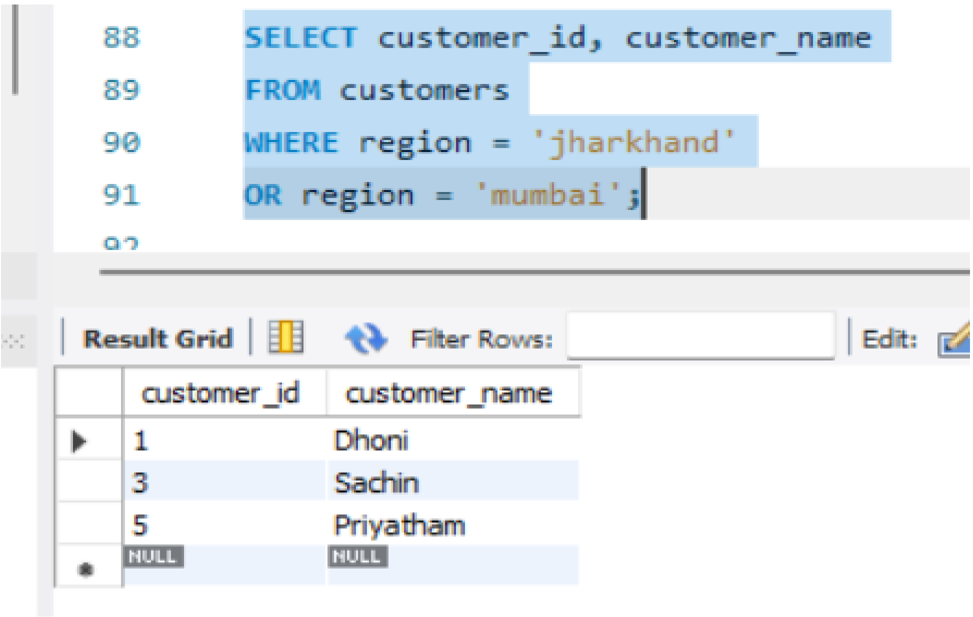
customer_id	customer_name	region	order_id	order_date
1	Dhoni	jharkhand	103	2024-05-03
1	Dhoni	jharkhand	101	2024-05-01
2	Kohli	delhi	102	2024-05-02
3	Sachin	mumbai	104	2024-05-04
4	Rohit	hyderabad	NULL	NULL
5	Priyatham	jharkhand	NULL	NULL

Assignment 3: Utilize a subquery to find customers who have placed orders above the average order value, and write a UNION query to combine two SELECT statements with the same number of columns.



```
SELECT customer_id, customer_name
FROM customers
WHERE customer_id IN (
    SELECT customer_id
    FROM orders
    GROUP BY customer_id
    HAVING AVG(total_amount) > (
        SELECT AVG(total_amount)
        FROM orders
    )
)
UNION
```

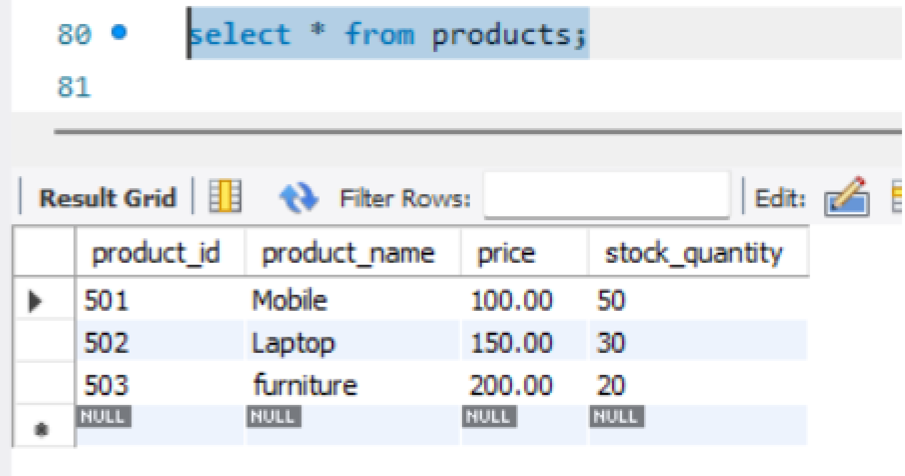
customer_id	customer_name
3	Sachin
NULL	NULL



```
SELECT customer_id, customer_name
FROM customers
WHERE region = 'jharkhand'
OR region = 'mumbai';
```

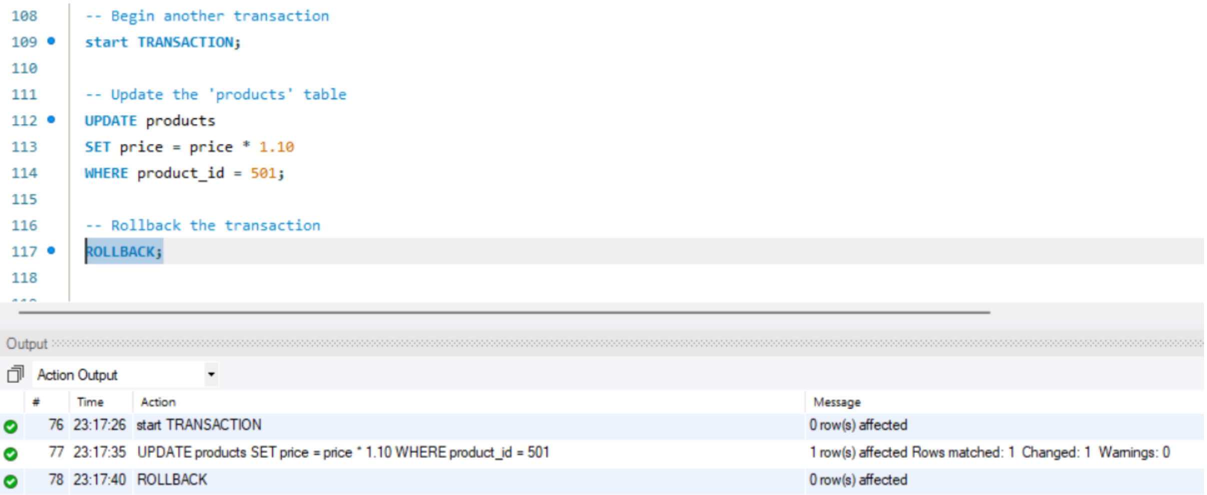
customer_id	customer_name
1	Dhoni
3	Sachin
5	Priyatham
NULL	NULL

Assignment 4: Compose SQL statements to BEGIN a transaction, INSERT a new record into the 'orders' table, COMMIT the transaction, then UPDATE the 'products' table, and ROLLBACK the transaction.



```sql
select * from products;
```

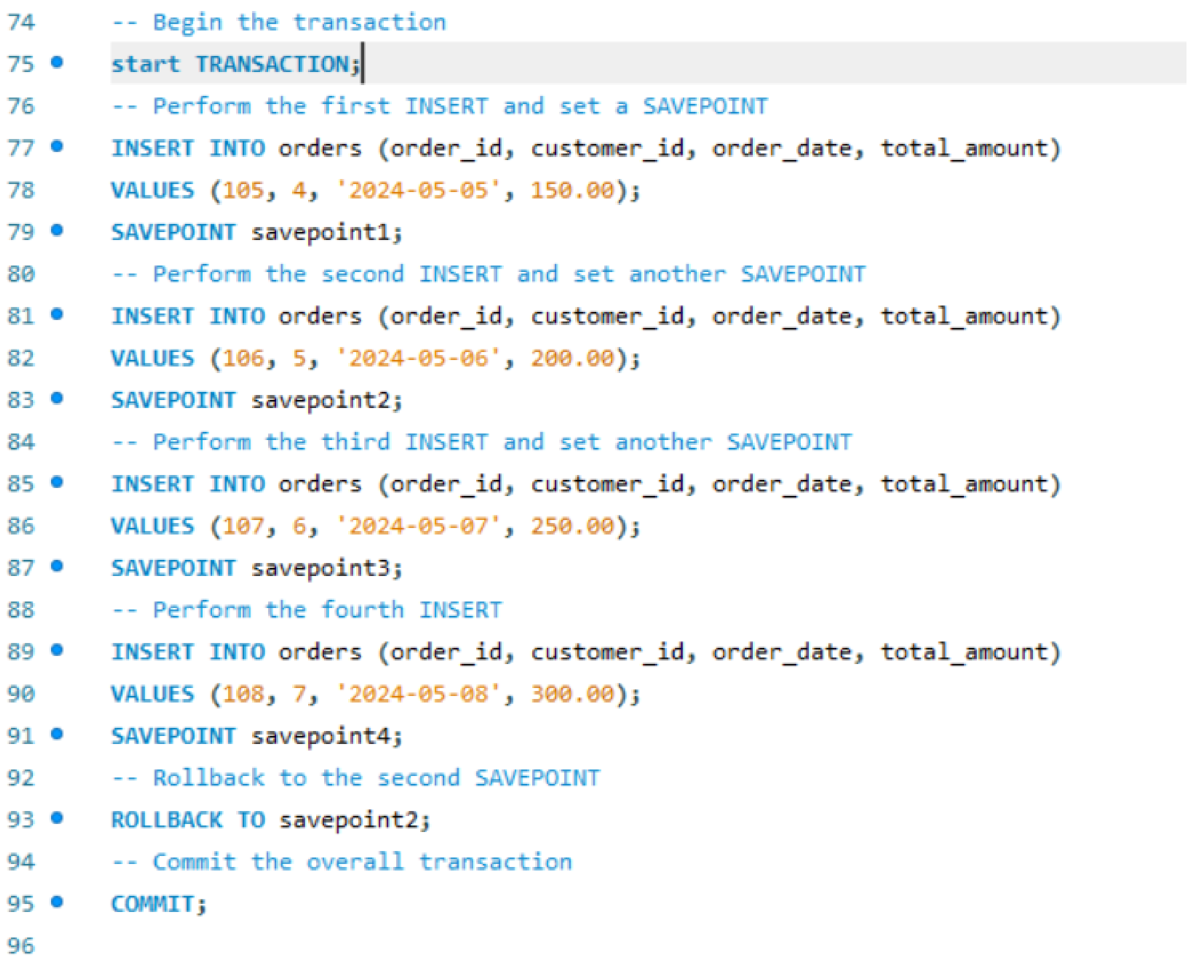
	product_id	product_name	price	stock_quantity
▶	501	Mobile	100.00	50
	502	Laptop	150.00	30
	503	furniture	200.00	20
	NULL	NULL	NULL	NULL



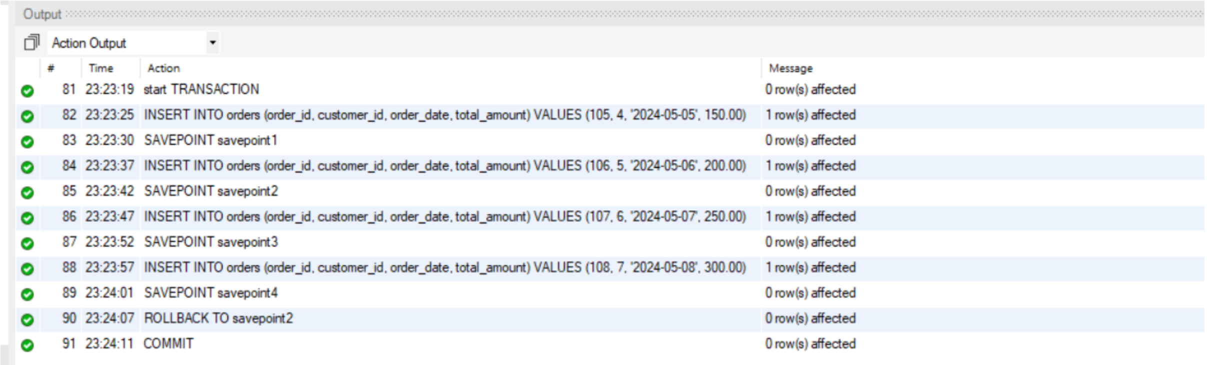
```sql
-- Begin another transaction
start TRANSACTION;

-- Update the 'products' table
UPDATE products
SET price = price * 1.10
WHERE product_id = 501;

-- Rollback the transaction
ROLLBACK;
```

	#	Time	Action	Message
✓	76	23:17:26	start TRANSACTION	0 row(s) affected
✓	77	23:17:35	UPDATE products SET price = price * 1.10 WHERE product_id = 501	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✓	78	23:17:40	ROLLBACK	0 row(s) affected

Assignment 5: Begin a transaction, perform a series of INSERTs into 'orders', setting a SAVEPOINT after each, rollback to the second SAVEPOINT, and COMMIT the overall transaction.



```sql
-- Begin the transaction
start TRANSACTION;
-- Perform the first INSERT and set a SAVEPOINT
INSERT INTO orders (order_id, customer_id, order_date, total_amount)
VALUES (105, 4, '2024-05-05', 150.00);
SAVEPOINT savepoint1;
-- Perform the second INSERT and set another SAVEPOINT
INSERT INTO orders (order_id, customer_id, order_date, total_amount)
VALUES (106, 5, '2024-05-06', 200.00);
SAVEPOINT savepoint2;
-- Perform the third INSERT and set another SAVEPOINT
INSERT INTO orders (order_id, customer_id, order_date, total_amount)
VALUES (107, 6, '2024-05-07', 250.00);
SAVEPOINT savepoint3;
-- Perform the fourth INSERT
INSERT INTO orders (order_id, customer_id, order_date, total_amount)
VALUES (108, 7, '2024-05-08', 300.00);
SAVEPOINT savepoint4;
-- Rollback to the second SAVEPOINT
ROLLBACK TO savepoint2;
-- Commit the overall transaction
COMMIT;
```



#	Time	Action	Message
81	23:23:19	start TRANSACTION	0 row(s) affected
82	23:23:25	INSERT INTO orders (order_id, customer_id, order_date, total_amount) VALUES (105, 4, '2024-05-05', 150.00)	1 row(s) affected
83	23:23:30	SAVEPOINT savepoint1	0 row(s) affected
84	23:23:37	INSERT INTO orders (order_id, customer_id, order_date, total_amount) VALUES (106, 5, '2024-05-06', 200.00)	1 row(s) affected
85	23:23:42	SAVEPOINT savepoint2	0 row(s) affected
86	23:23:47	INSERT INTO orders (order_id, customer_id, order_date, total_amount) VALUES (107, 6, '2024-05-07', 250.00)	1 row(s) affected
87	23:23:52	SAVEPOINT savepoint3	0 row(s) affected
88	23:23:57	INSERT INTO orders (order_id, customer_id, order_date, total_amount) VALUES (108, 7, '2024-05-08', 300.00)	1 row(s) affected
89	23:24:01	SAVEPOINT savepoint4	0 row(s) affected
90	23:24:07	ROLLBACK TO savepoint2	0 row(s) affected
91	23:24:11	COMMIT	0 row(s) affected

Assignment 6: Draft a brief report on the use of transaction logs for data recovery and create a hypothetical scenario where a transaction log is instrumental in data recovery after an unexpected shutdown.

Report on the Use of Transaction Logs for Data Recovery

Introduction

Transaction logs are a critical component of database management systems (DBMS). They record all transactions and modifications made to the database, providing a detailed history of changes. This report highlights the importance of transaction logs for data recovery and presents a hypothetical scenario where they are instrumental in recovering data after an unexpected shutdown.

Importance of Transaction Logs

1. **Data Integrity and Consistency**: Transaction logs ensure data integrity and consistency by keeping a record of every transaction. In the event of a system failure, the logs can be used to restore the database to a consistent state.
2. **Point-in-Time Recovery**: Transaction logs enable point-in-time recovery, allowing administrators to restore the database to a specific moment before a failure occurred. This is crucial for minimizing data loss and maintaining business continuity.
3. **Crash Recovery**: After an unexpected shutdown or crash, transaction logs help recover uncommitted transactions. The DBMS can use the logs to roll back incomplete transactions and redo committed transactions that were not yet written to the database.
4. **Auditing and Compliance**: Transaction logs provide a trail of all database operations, which is essential for auditing and ensuring compliance with regulatory requirements.

Hypothetical Scenario

Scenario: A Financial Services Company Facing an Unexpected Shutdown

Company Profile: A financial services company manages a database containing critical information such as customer accounts, transactions, and balances. The database is crucial for daily operations, including processing transactions and generating financial reports.

Incident: Unexpected System Shutdown

At 3:00 PM on a busy business day, the company's database server unexpectedly shuts down due to a hardware failure. The last full backup was taken at 2:00 PM, and numerous transactions were processed between 2:00 PM and 3:00 PM.

Role of Transaction Logs in Recovery

1. **Initial Assessment**:
    - The database administrator (DBA) identifies the hardware failure and initiates the repair process. Once the hardware issue is resolved, the DBA begins the data recovery process.
2. **Restoring from Backup**:

- The DBA restores the database from the last full backup taken at 2:00 PM. However, this backup does not include the transactions processed between 2:00 PM and 3:00 PM.

3. **Applying Transaction Logs**:
    - The DBA accesses the transaction logs, which contain a record of all transactions from 2:00 PM to the moment of the shutdown.
    - The DBMS uses the transaction logs to apply all committed transactions to the restored database. This process includes:
        - **Redoing**: Applying all changes from committed transactions that were not yet written to the database.
        - **Undoing**: Rolling back any incomplete transactions to ensure data consistency.
4. **Recovery Completion**:
    - By 3:30 PM, the DBA successfully recovers the database to its state at 3:00 PM, just before the shutdown. All customer transactions processed between 2:00 PM and 3:00 PM are intact, and the database is consistent.
5. **Verification**:
    - The DBA performs verification checks to ensure all data is accurate and the system is fully operational. Customers and employees can resume normal activities without any data loss.

Conclusion

Transaction logs play a vital role in data recovery, providing a reliable method to restore databases to a consistent state after unexpected failures. In the hypothetical scenario, transaction logs were instrumental in recovering the financial services company's database, ensuring minimal data loss and maintaining business continuity. This highlights the importance of implementing robust transaction logging and regular backups in any critical database system.