Assignment-1

Question-1 Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.?

**ACID Properties**:

The ACID properties are a set of four properties that guarantee the reliability of transactions in a database:

Atomicity: This property ensures that a transaction is all or nothing. Either all the operations within the transaction are completed successfully, or none of them are. If any part of the transaction fails, the entire transaction is rolled back.

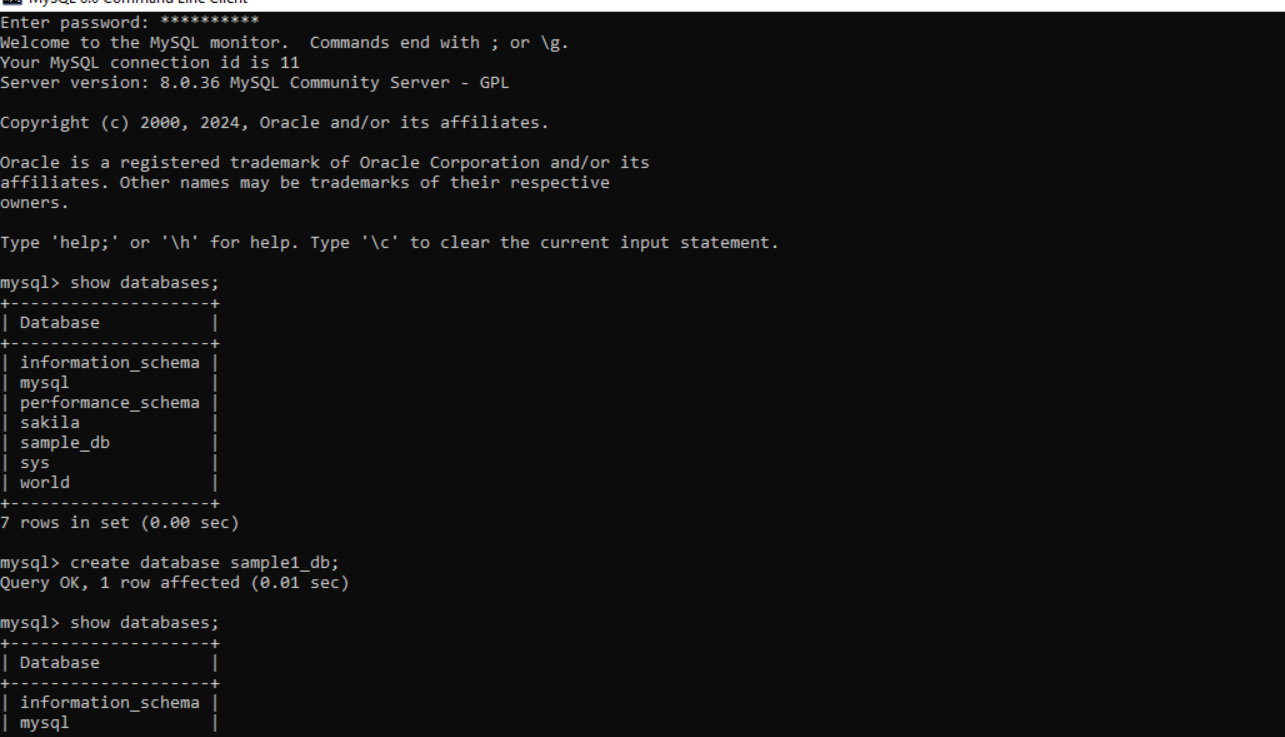
Consistency: This property ensures that the database remains in a consistent state before and after the transaction. All integrity constraints, such as foreign key constraints, must be maintained. In other words, the database should move from one consistent state to another consistent state after the transaction.

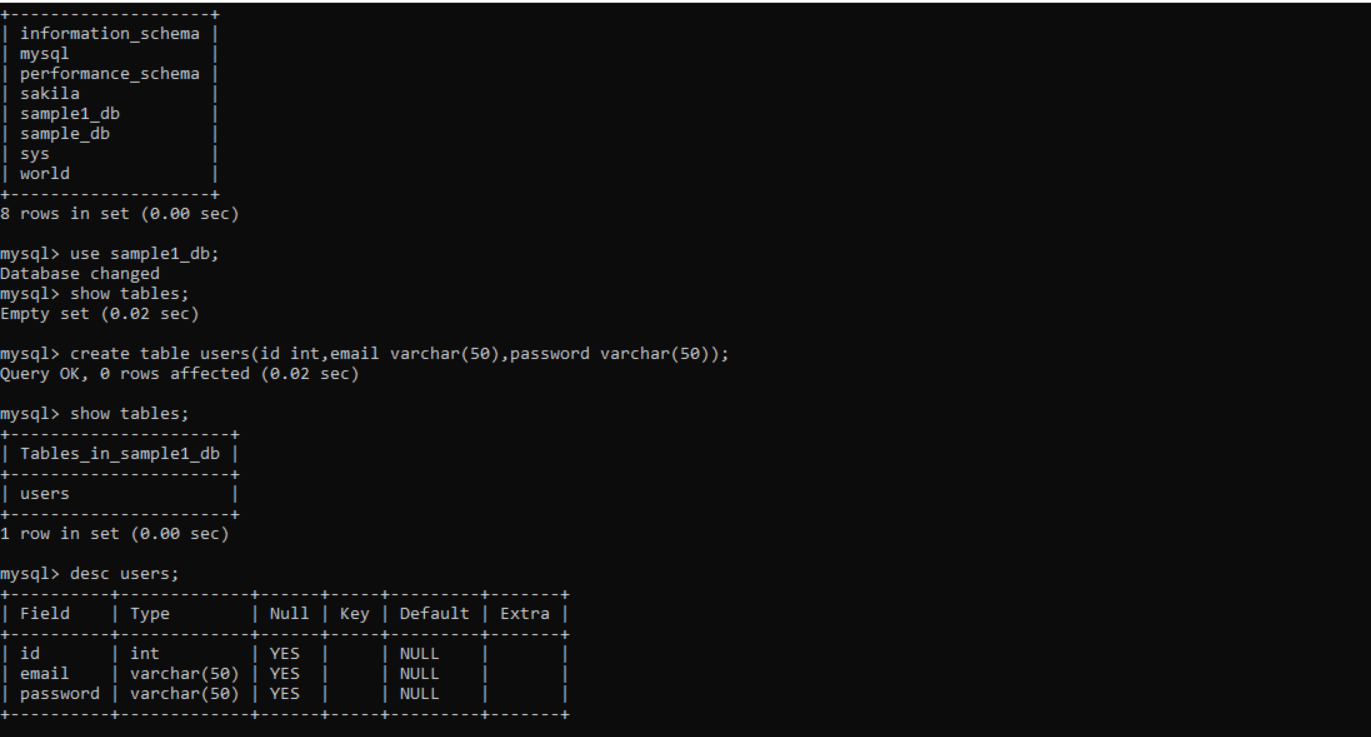
Isolation: This property ensures that the execution of transactions concurrently produces the same result as if they were executed serially. Transactions should be isolated from each other, so that the operations of one transaction are not visible to other transactions until the first transaction is complete.

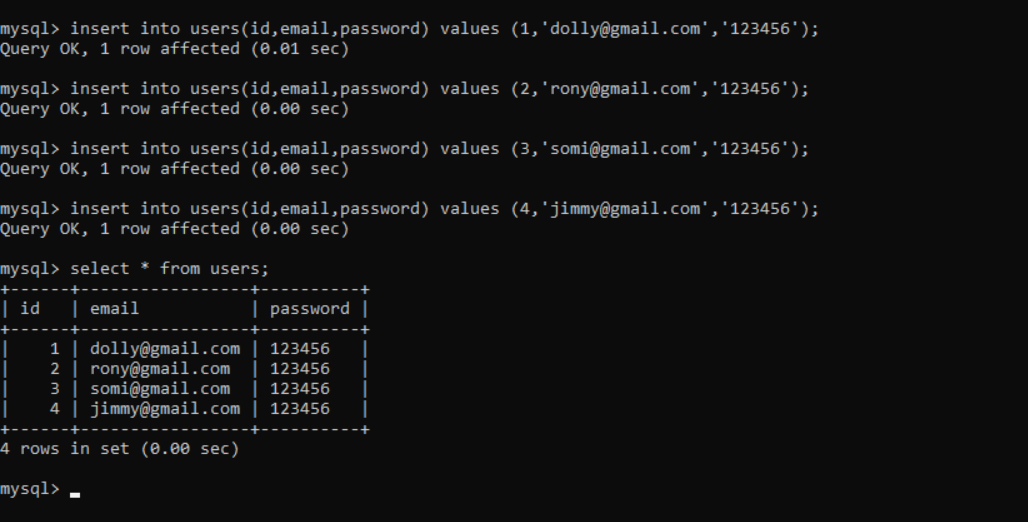
Durability: This property guarantees that once a transaction is committed, its changes are permanent and will not be lost, even in the event of a system failure.

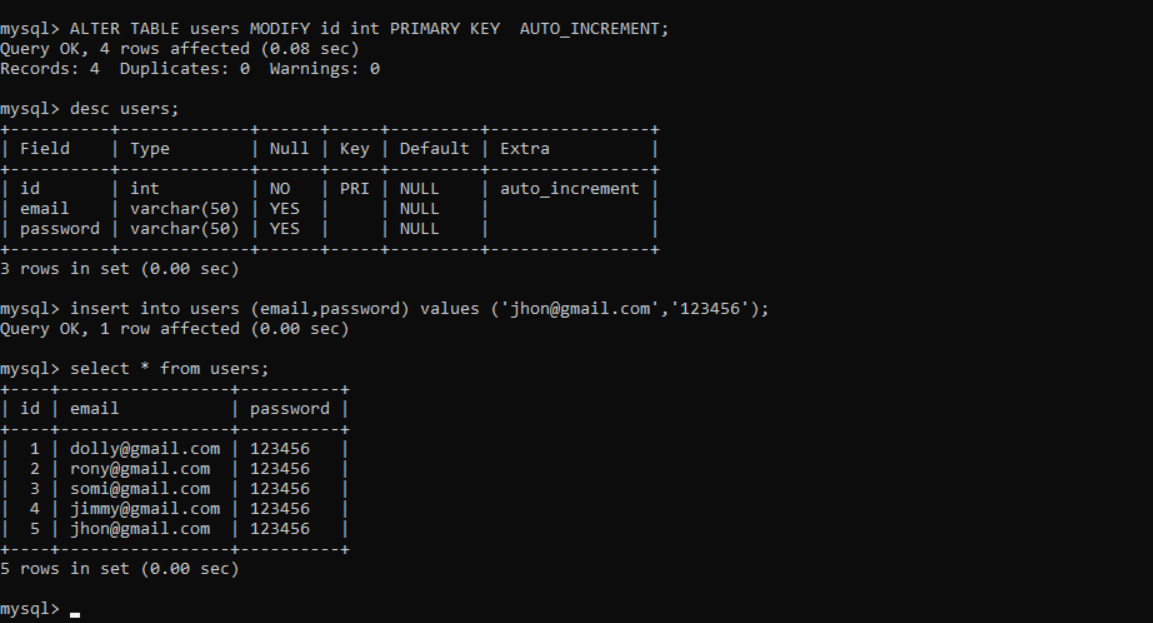
Assignment-4

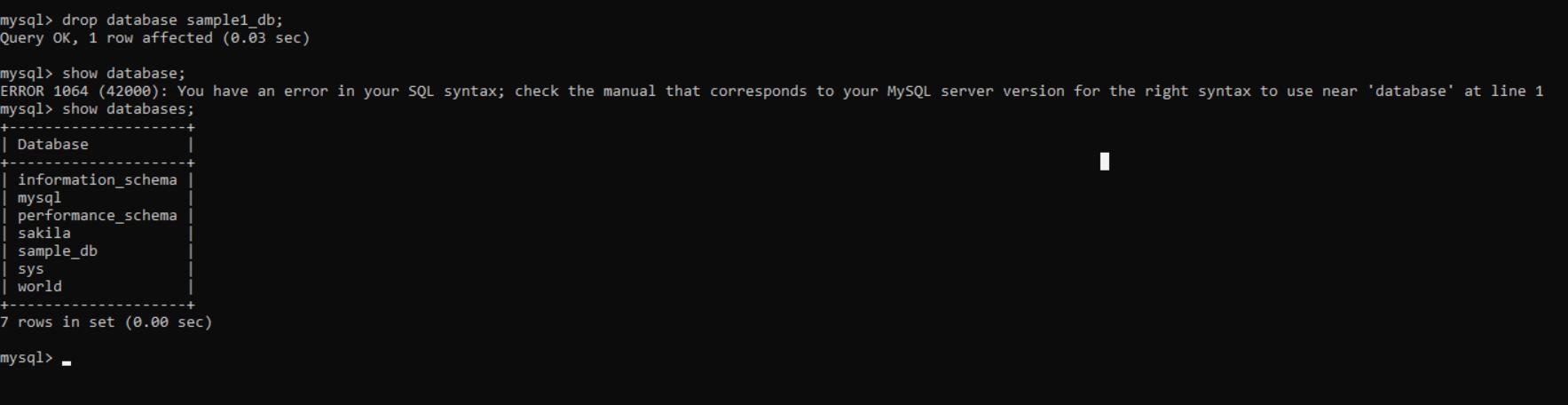
Question-4 Write SQL statements to CREATE a new database and tables that reflect the library schema you designed earlier. Use ALTER statements to modify the table structures and DROP statements to remove a redundant table.?





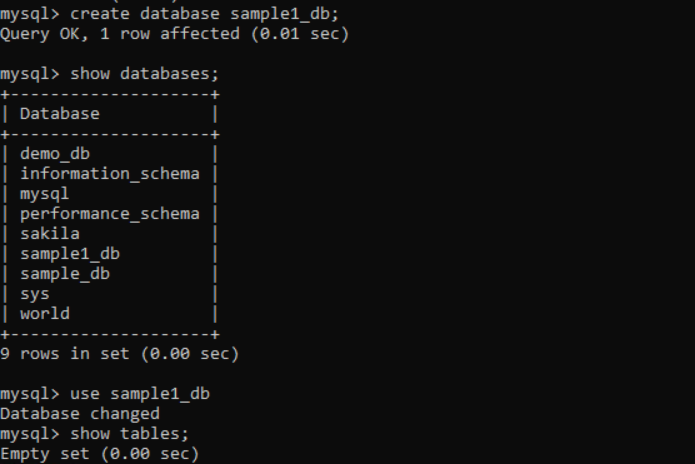


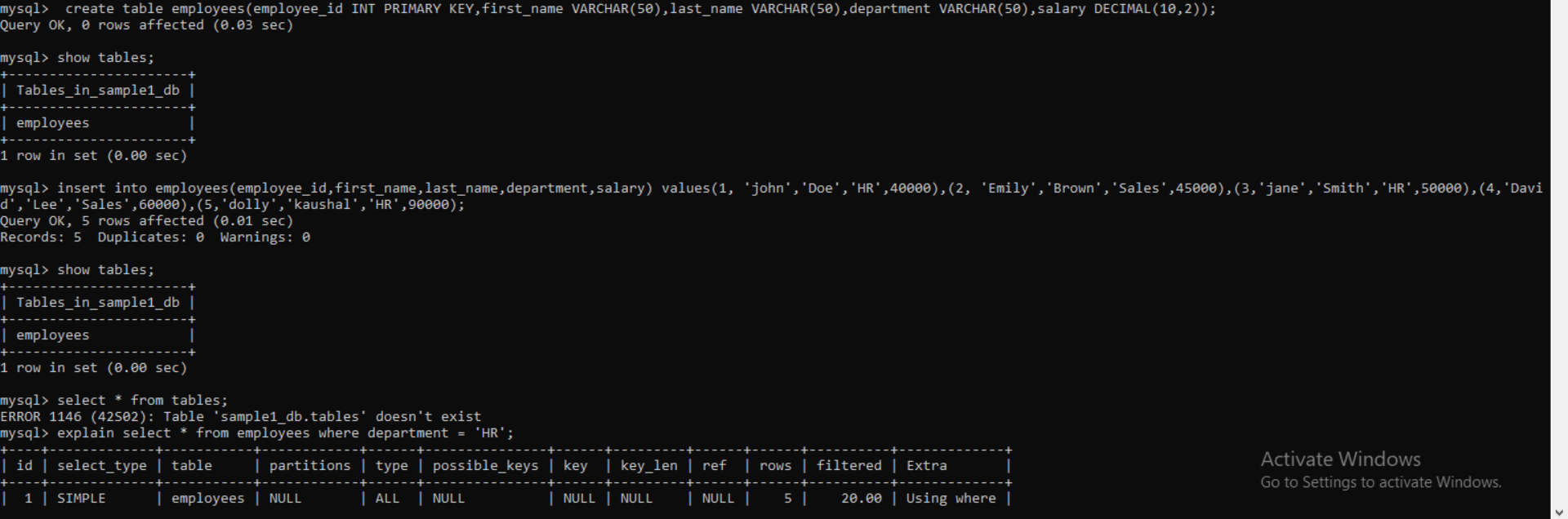


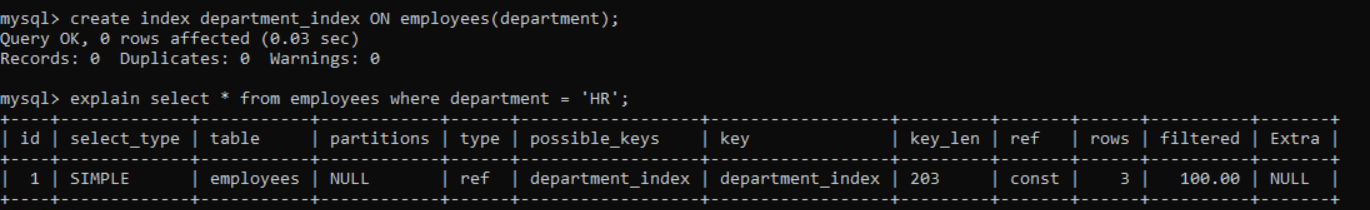


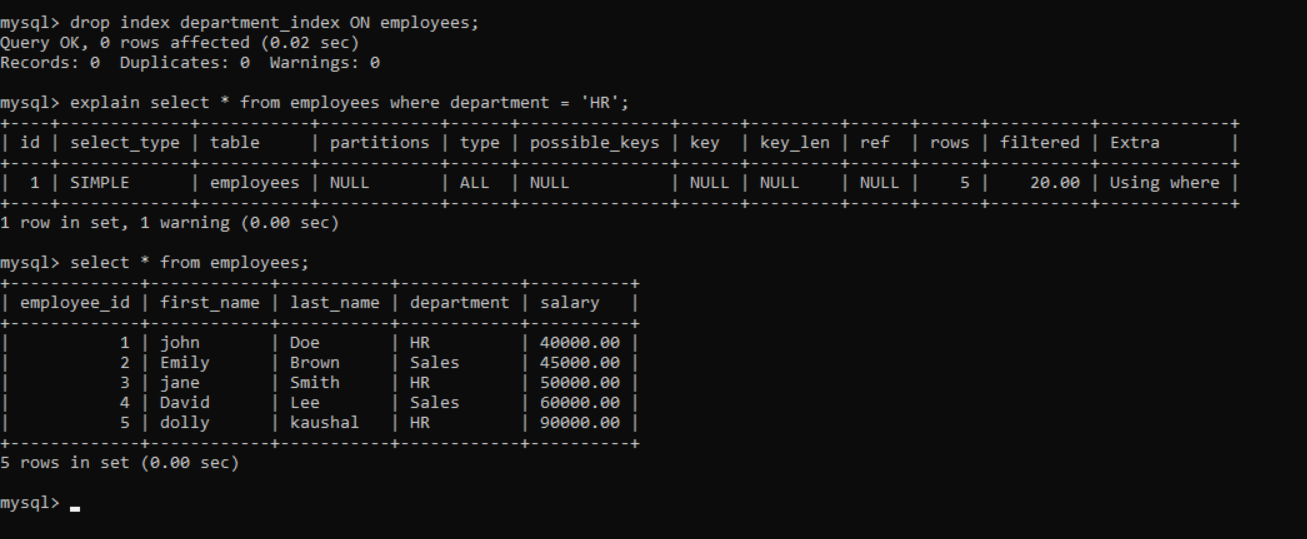
Assignment-2

Question-2 Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyze the impact on query execution.?

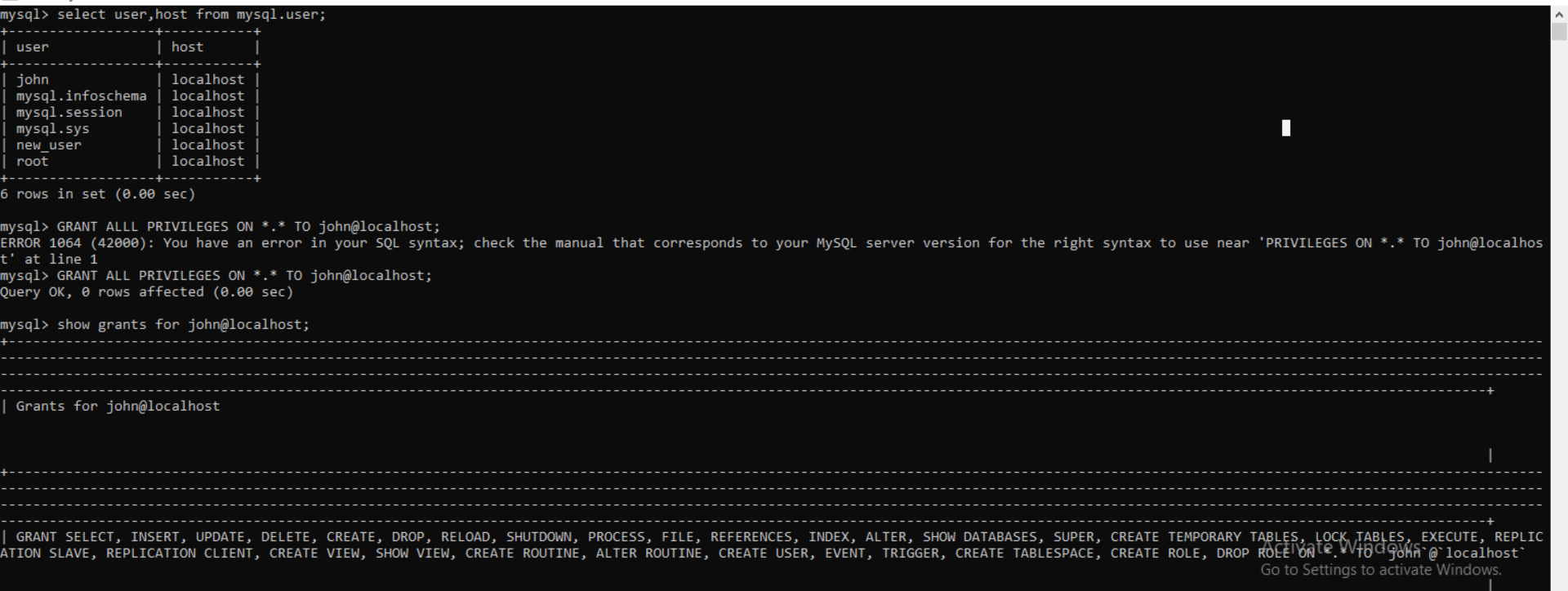








Question-6 Create a new database user with specific privileges using the CREATE USER and GRANT commands. Then, write a script to REVOKE certain privileges and DROP the user.

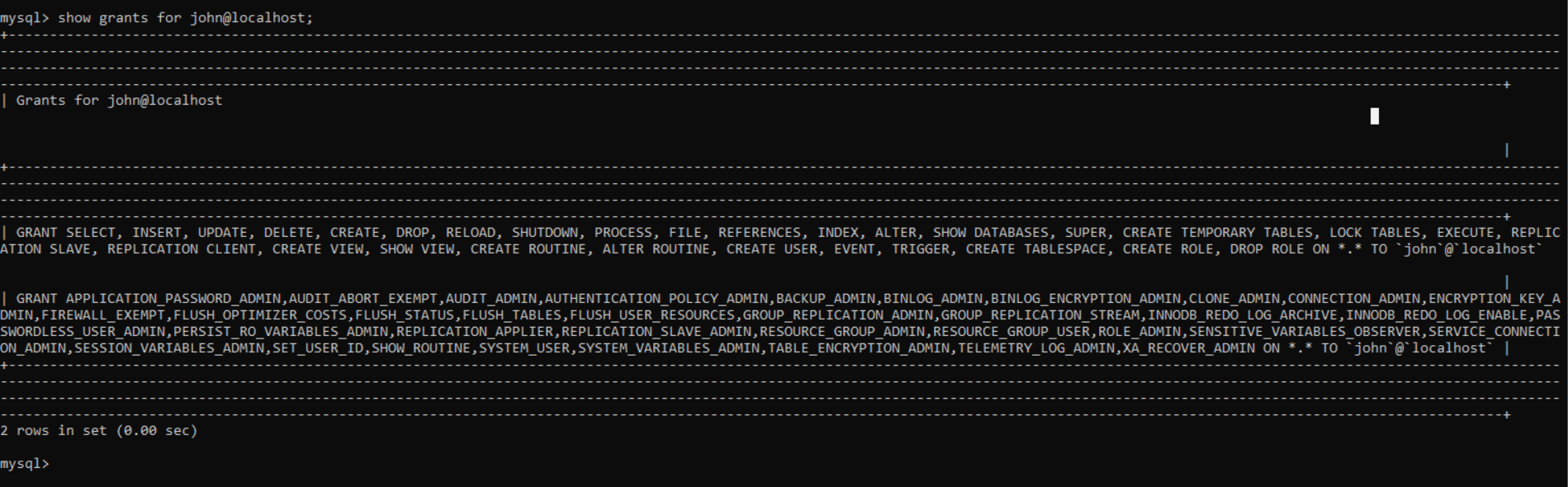


CREATE USER is used to create a new user.

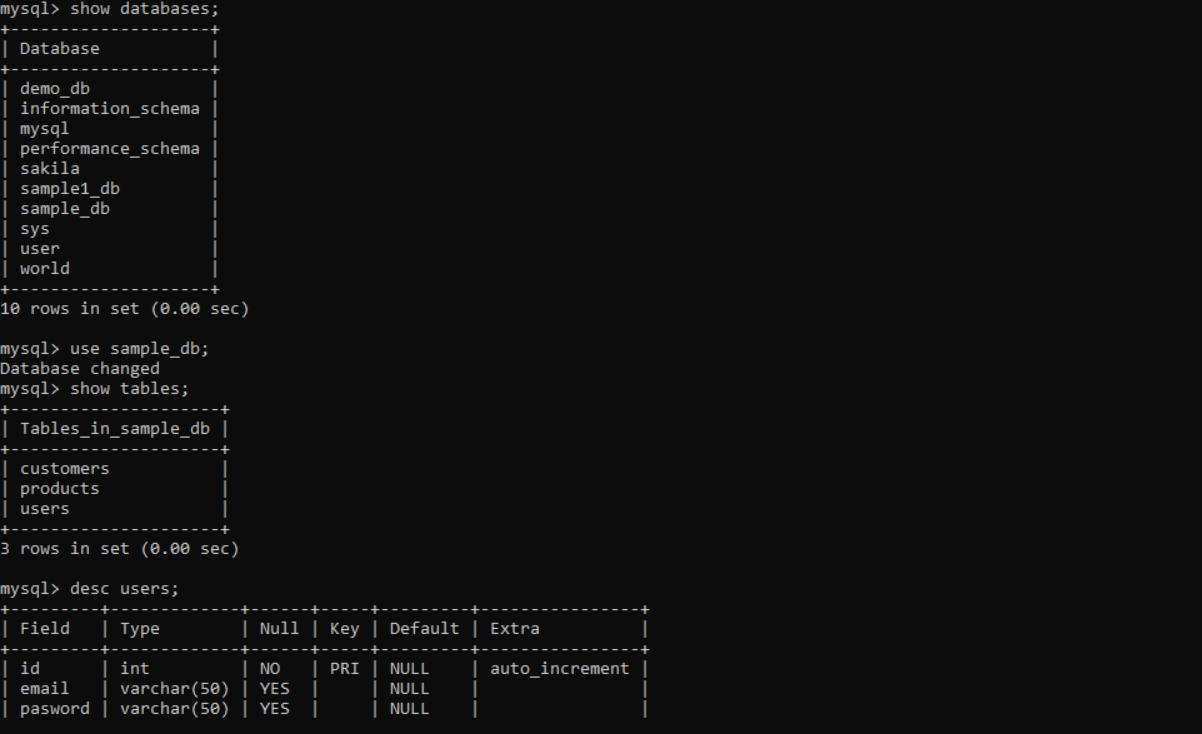
GRANT is used to grant specific privileges to the user.

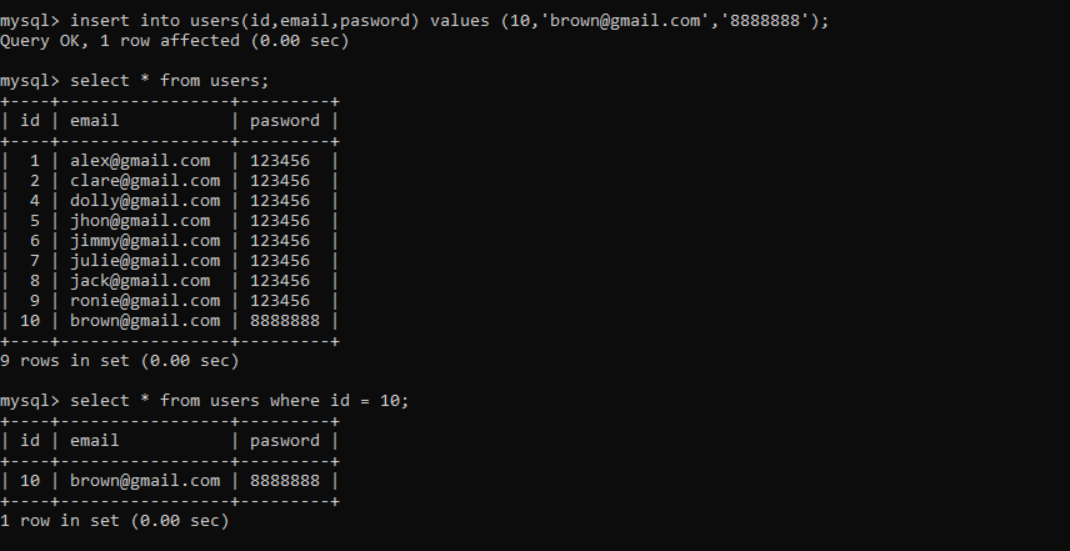
REVOKE is used to revoke certain privileges from the user.

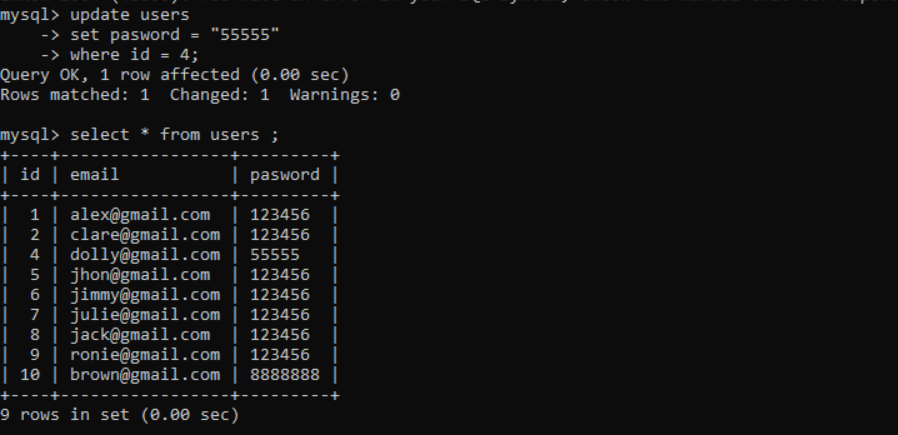
DROP USER is used to drop the user.



Question-7 Prepare a series of SQL statements to INSERT new records into the library tables, UPDATE existing records with new information, and DELETE records based on specific criteria. Include BULK INSERT operations to load data from an external source."









BULK INSERT from an external source:

Suppose you have a CSV file named books\_data.csv with columns title, author, published\_year, genre, copies\_available, you can bulk insert the data using:

**-- Bulk insert data from CSV into books table**

LOAD DATA INFILE '/path/to/books\_data.csv'

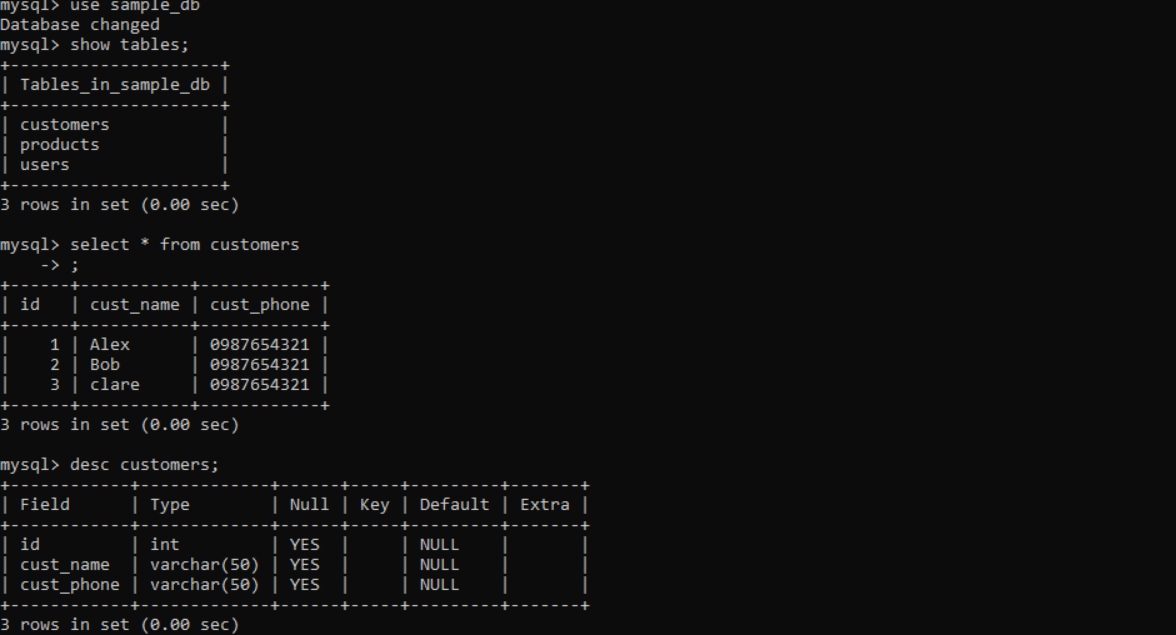
INTO TABLE books

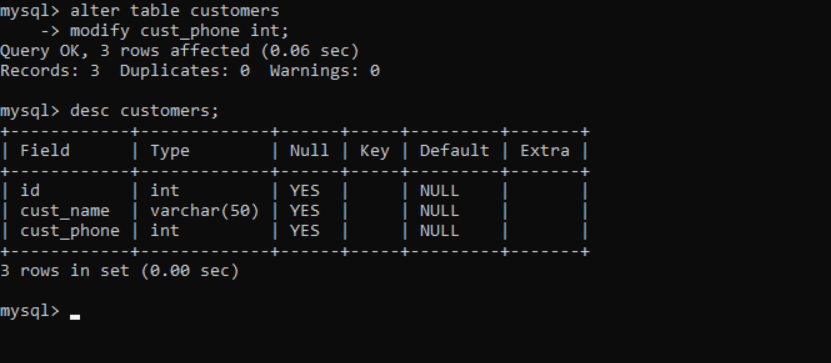
FIELDS TERMINATED BY ',' ENCLOSED BY '"'

LINES TERMINATED BY '\n'

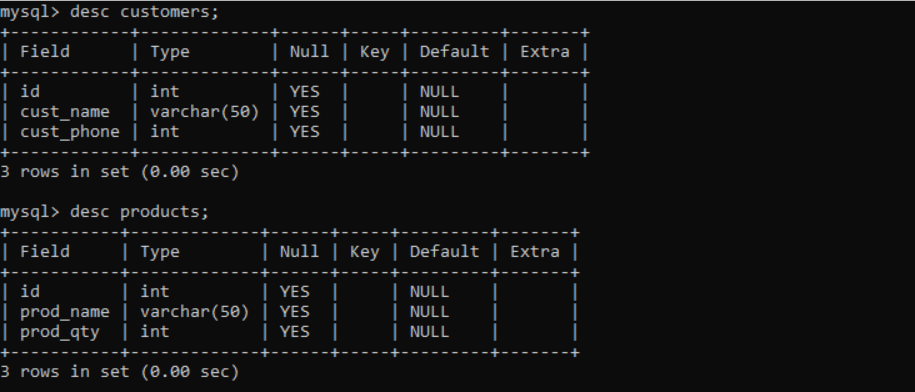
IGNORE 1 ROWS; -- Ignore header row if exists

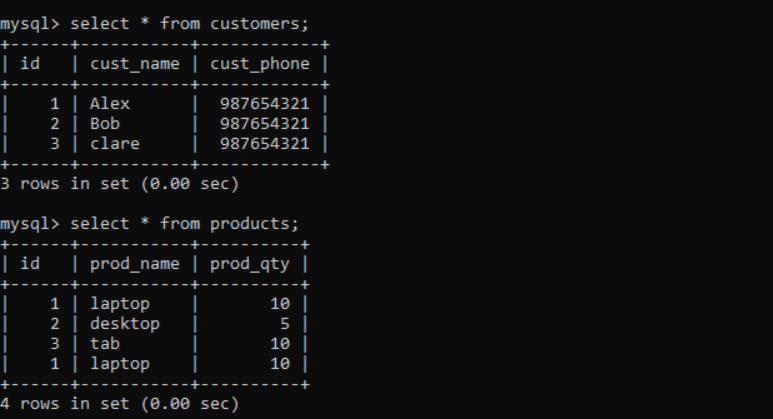
Question-1 Write a SELECT query to retrieve all columns from a 'customers' table, and modify it to return only the customer name and email address for customers in a specific city.

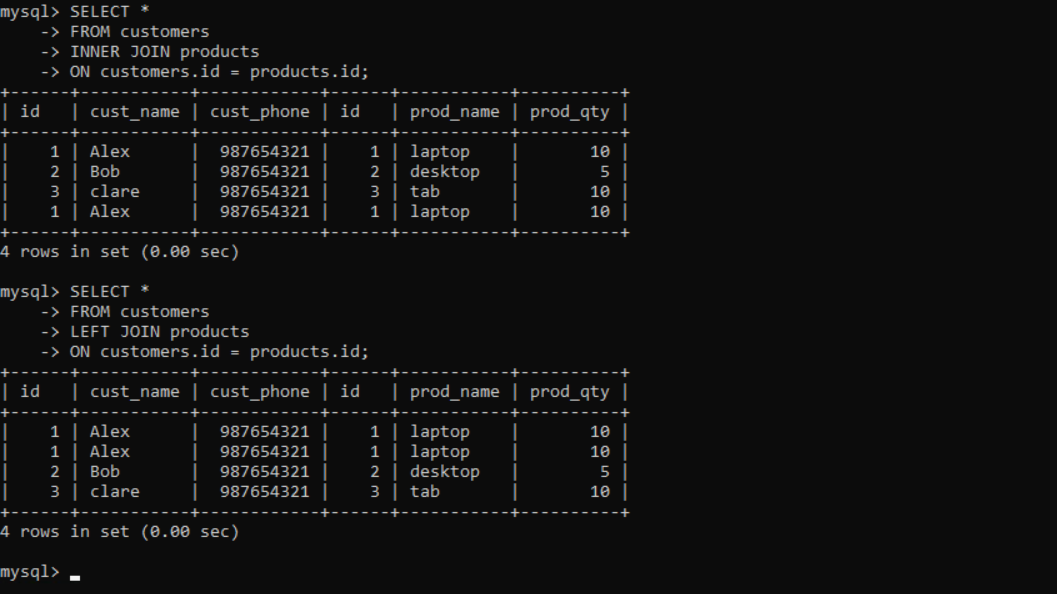




Question-2Craft a query using an INNER JOIN to combine 'orders' and 'customers' tables for customers in a specified region, and a LEFT JOIN to display all customers including those without orders.







Question-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.

**----Query1---**

SELECT customer\_id, customer\_name

FROM customers

WHERE customer\_id IN (

SELECT customer\_id

FROM orders

GROUP BY customer\_id

HAVING AVG(order\_value) > (

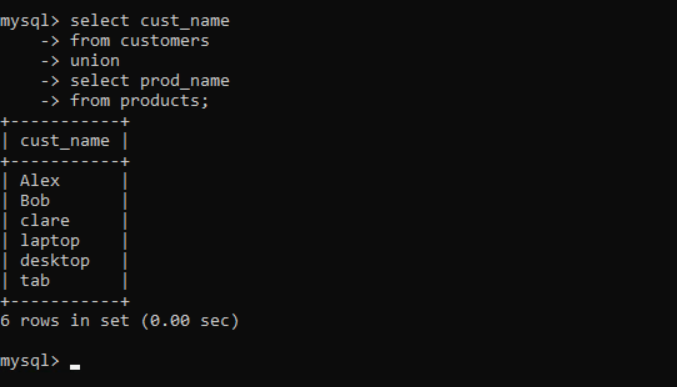
SELECT AVG(order\_value)

FROM orders

)

);

**----Query2---**

****

Question-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.

**-- Begin transaction**

BEGIN;

**-- Insert a new record into the 'orders' table**

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_value)

VALUES (1001, 101, '2024-05-17', 150.00);

**-- Commit the transaction**

COMMIT;

**-- Update the 'products' table**

UPDATE products

SET stock\_quantity = stock\_quantity - 1

WHERE product\_id = 2001;

**-- Rollback the transaction**

ROLLBACK;

**Explanation:**

BEGIN;: Starts a new transaction.

INSERT INTO orders ...: Inserts a new record into the 'orders' table.

COMMIT;: Commits the transaction, saving the changes permanently.

UPDATE products ...: Updates the 'products' table, reducing the stock quantity for a specific product.

ROLLBACK;: Rolls back the transaction, undoing any changes made since the beginning of the transaction.

Question-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.

**-- Begin transaction**

START TRANSACTION;

**-- Insert into orders with savepoint 1**

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_value)

VALUES (1001, 101, '2024-05-17', 150.00);

SAVEPOINT savepoint1;

**-- Insert into orders with savepoint 2**

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_value)

VALUES (1002, 102, '2024-05-18', 200.00);

SAVEPOINT savepoint2;

**-- Insert into orders with savepoint 3**

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_value)

VALUES (1003, 103, '2024-05-19', 175.00);

**-- Rollback to savepoint 2**

ROLLBACK TO savepoint2;

**-- Commit the overall transaction**

COMMIT;

**Explanation:**

START TRANSACTION;: Begins the transaction.

SAVEPOINT savepoint1;, SAVEPOINT savepoint2;: Sets savepoints after each INSERT operation.

ROLLBACK TO savepoint2;: Rolls back to the second savepoint, undoing changes made after that point.

COMMIT;: Commits the overall transaction, saving the changes made before the rollback.

Question-6Draft 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:** **The Use of Transaction Logs for Data Recovery**

**Introduction:**

Transaction logs play a crucial role in ensuring data integrity and facilitating recovery in the event of unexpected system failures, crashes, or other disasters. These logs maintain a record of all database modifications, allowing for the reconstruction of data to a consistent state before the failure occurred.

**Importance of Transaction Logs:**

**Transaction logs provide several benefits for data recovery:**

**Redundancy:** They serve as a redundant copy of the database changes, ensuring that no modifications are lost.

**Recovery Point:** Transaction logs provide a recovery point from which database administrators can restore data to a consistent state.

**Durability:** Even in the event of a system crash, transaction logs remain intact, safeguarding against data loss.

**How Transaction Logs Work:**

Transaction logs record every transaction that modifies the database. Each log entry contains information about the operation performed, such as INSERT, UPDATE, DELETE, along with the affected data. These logs are typically stored separately from the database itself, often on different disks or locations, to mitigate the risk of simultaneous failure.

**Scenario:**

Suppose a company maintains a database of customer orders for an e-commerce platform. During a routine update, the server experiences an unexpected shutdown due to a power outage.

---------------------------The End------------------

Hypothetical Scenario:

Before the shutdown, the database was in a consistent state, with all transactions either committed or pending.

When the server reboots, the database management system (DBMS) detects the incomplete transactions and consults the transaction logs.

Using the transaction logs, the DBMS identifies the last completed transaction and rolls back any incomplete transactions to maintain consistency.

The DBMS then applies the logged transactions to restore the database to its state just before the unexpected shutdown.

As a result, no data is lost, and the system is restored to its previous operational state without any inconsistencies or corruption.

Conclusion:

Transaction logs are essential for ensuring data integrity and facilitating rapid recovery in the event of unexpected system failures. By maintaining a record of all database modifications, transaction logs provide a reliable mechanism for restoring data to a consistent state, minimizing downtime, and preserving business continuity.