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.

ELECT \* FROM customers;

SELECT customer\_name, email\_address

FROM customers

WHERE city = 'Specific City';

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.

Inner join

SELECT c.customer\_name, o.order\_date, o.total\_amount

FROM customers c

INNER JOIN orders o ON c.customer\_id = o.customer\_id

WHERE c.region = 'Specified Region';

Left join:

SELECT c.customer\_name, o.order\_date, o.total\_amount

FROM customers c

LEFT JOIN orders o ON c.customer\_id = o.customer\_id

WHERE c.region = 'Specified Region';

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 c.customer\_name, o.total\_amount

FROM customers c

INNER JOIN orders o ON c.customer\_id = o.customer\_id

WHERE o.total\_amount > (

SELECT AVG(total\_amount)

FROM orders

);

Using union:

SELECT c.customer\_name, o.total\_amount

FROM customers c

INNER JOIN orders o ON c.customer\_id = o.customer\_id

WHERE o.total\_amount > 1000

UNION

SELECT c.customer\_name, o.total\_amount

FROM customers c

INNER JOIN orders o ON c.customer\_id = o.customer\_id

WHERE o.order\_date > '2020-01-01';

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.

 BEGIN TRANSACTION;

INSERT INTO orders (customer\_id, order\_date, total\_amount)

VALUES (1, '2022-01-01', 500.00);

COMMIT;

UPDATE products

SET price = price \* 1.10

WHERE product\_id = 1;

ROLLBACK;

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.

BEGIN TRANSACTION;

INSERT INTO orders (customer\_id, order\_date, total\_amount)

VALUES (1, '2022-01-01', 500.00);

SAVEPOINT sp1;

INSERT INTO orders (customer\_id, order\_date, total\_amount)

VALUES (2, '2022-01-05', 300.00);

SAVEPOINT sp2;

INSERT INTO orders (customer\_id, order\_date, total\_amount)

VALUES (3, '2022-01-10', 200.00);

ROLLBACK TO SAVEPOINT sp2;

COMMIT;

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: The Role of Transaction Logs in Data Recovery**

Transaction logs are a critical component of database management systems, playing a vital role in ensuring data integrity and facilitating data recovery in the event of unexpected shutdowns or system failures. A transaction log is a record of all changes made to the database, including inserts, updates, and deletes, which allows the database to be restored to a consistent state in case of a failure.

The transaction log serves as a safety net, enabling database administrators to recover data and maintain business continuity in the face of unexpected events. By analyzing the transaction log, administrators can identify the last consistent state of the database before the failure, and then roll back the database to that point. This ensures that all transactions are accounted for, and the database is restored to a consistent state.

The benefits of transaction logs in data recovery are numerous:

* **Data Integrity**: Transaction logs ensure that data is consistent and accurate, even in the event of a failure.
* **Rapid Recovery**: With a transaction log, administrators can quickly recover data and restore the database to a consistent state.
* **Business Continuity**: By minimizing downtime and ensuring rapid recovery, transaction logs help maintain business continuity and reduce the impact of unexpected events.

**Hypothetical Scenario:**

**Company:** Online Banking Platform, "SecureBank"

**Scenario:** SecureBank's database server experiences an unexpected shutdown due to a power outage, resulting in the loss of all transactions processed during the last hour. The database is left in an inconsistent state, with some transactions partially processed and others missing.

**Transaction Log to the Rescue:**

Upon restarting the server, the database administrator analyzes the transaction log to identify the last consistent state of the database before the shutdown. The log reveals that 500 transactions were processed during the last hour, including deposits, withdrawals, and transfers.

Using the transaction log, the administrator rolls back the database to the last consistent state, ensuring that all transactions are accounted for. The administrator then reapplies the transactions that were not committed, using the log to recreate the exact sequence of events.

**Outcome:**

Thanks to the transaction log, SecureBank is able to recover all transactions and restore the database to a consistent state. The online banking platform is back online within 30 minutes, with all customer accounts accurately reflected. The rapid recovery minimizes the impact on customers and ensures business continuity.

In this scenario, the transaction log plays a crucial role in data recovery, enabling SecureBank to restore its database to a consistent state and maintain customer trust. The transaction log's ability to facilitate rapid recovery and ensure data integrity makes it an essential component of any database management system.