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.

Ans:

- ***Query to return all columns from customer table:***

```
mysql> SELECT * FROM customers;
+------------+--------------+---------------------+------------+--------------+-------------+-------+---------+------------+---------------------+
| CustomerID | Name         | Email               | Phone      | Address      | City        | State | Country | PostalCode | created_at          |
+------------+--------------+---------------------+------------+--------------+-------------+-------+---------+------------+---------------------+
|          1 | John Doe     | johndoe@example.com | 1234567890 | 123 Main St  | New York    | NY    | USA     | 10001      | 2025-01-20 08:42:19 |
|          2 | Jane Smith   | janesmith@example.com | 0987654321 | 456 Elm St   | Los Angeles | CA    | USA     | 90001      | 2025-01-20 08:42:19 |
|          3 | Alice Johnson | alicej@example.com  | 5556667777 | 789 Oak St   | New York    | NY    | USA     | 10002      | 2025-01-20 08:42:19 |
|          4 | Bob Brown    | bobb@example.com    | 3334445555 | 321 Pine St  | Chicago     | IL    | USA     | 60601      | 2025-01-20 08:42:19 |
+------------+--------------+---------------------+------------+--------------+-------------+-------+---------+------------+---------------------+
4 rows in set (0.00 sec)
```

- ***Query to return only customer's name and email for specific city:***

```
mysql> SELECT Name, EMail From customers where city ='Chicago';
+-----------+-----------------+
| Name      | EMail           |
+-----------+-----------------+
| Bob Brown | bobb@example.com |
+-----------+-----------------+
1 row in set (0.00 sec)

mysql> SELECT Name, EMail From customers where city like '%n%';
+---------------+----------------------+
| Name          | EMail                |
+---------------+----------------------+
| John Doe      | johndoe@example.com  |
| Jane Smith    | janesmith@example.com |
| Alice Johnson | alicej@example.com   |
+---------------+----------------------+
3 rows in set (0.00 sec)
```

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.

Ans:

- **INNER JOIN:**

```
mysql> select * from orders;
+---------+------------+------------+-------------+-----------+
| OrderID | CustomerID | OrderDate  | TotalAmount | Status    |
+---------+------------+------------+-------------+-----------+
|       1 |          1 | 2025-01-15 |      250.00 | Completed |
|       2 |          2 | 2025-01-16 |      150.00 | Pending   |
|       3 |          1 | 2025-01-17 |      300.00 | Cancelled |
|       4 |          4 | 2025-01-18 |      500.00 | Completed |
+---------+------------+------------+-------------+-----------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM customers INNER JOIN orders ON customers.CustomerID = orders.CustomerID WHERE customers.City = 'New York';
+------------+----------+--------------------+------------+-------------+----------+-------+---------+------------+---------------------+---------+------------+------------+-------------+-----------+
| CustomerID | Name     | Email              | Phone      | Address     | City     | State | Country | PostalCode | created_at          | OrderID | CustomerID | OrderDate  | TotalAmount | Status    |
+------------+----------+--------------------+------------+-------------+----------+-------+---------+------------+---------------------+---------+------------+------------+-------------+-----------+
|          1 | John Doe | johndoe@example.com | 1234567890 | 123 Main St | New York | NY    | USA     | 10001      | 2025-01-20 08:42:19 |       1 |          1 | 2025-01-15 |      250.00 | Completed |
|          1 | John Doe | johndoe@example.com | 1234567890 | 123 Main St | New York | NY    | USA     | 10001      | 2025-01-20 08:42:19 |       3 |          1 | 2025-01-17 |      300.00 | Cancelled |
+------------+----------+--------------------+------------+-------------+----------+-------+---------+------------+---------------------+---------+------------+------------+-------------+-----------+
2 rows in set (0.01 sec)

mysql>  SELECT * FROM customers INNER JOIN orders ON customers.CustomerID = orders.CustomerID WHERE customers.City = 'Chicago';
+------------+----------+-----------------+------------+-------------+---------+-------+---------+------------+---------------------+---------+------------+------------+-------------+-----------+
| CustomerID | Name     | Email           | Phone      | Address     | City    | State | Country | PostalCode | created_at          | OrderID | CustomerID | OrderDate  | TotalAmount | Status    |
+------------+----------+-----------------+------------+-------------+---------+-------+---------+------------+---------------------+---------+------------+------------+-------------+-----------+
|          4 | Bob Brown | bobb@example.com | 3334445555 | 321 Pine St | Chicago | IL    | USA     | 60601      | 2025-01-20 08:42:19 |       4 |          4 | 2025-01-18 |      500.00 | Completed |
+------------+----------+-----------------+------------+-------------+---------+-------+---------+------------+---------------------+---------+------------+------------+-------------+-----------+
1 row in set (0.00 sec)

mysql>
```

- **Left Join:**

```
mysql> SELECT
    ->     customers.Name AS CustomerName,
    ->     customers.Email AS CustomerEmail,
    ->     orders.OrderID,
    ->     orders.OrderDate,
    ->     orders.TotalAmount,
    ->     orders.Status
    -> FROM
    ->     customers
    -> LEFT JOIN
    ->     orders
    -> ON
    ->     customers.CustomerID = orders.CustomerID;
+---------------+---------------------+---------+------------+-------------+-----------+
| CustomerName  | CustomerEmail       | OrderID | OrderDate  | TotalAmount | Status    |
+---------------+---------------------+---------+------------+-------------+-----------+
| John Doe      | johndoe@example.com |       1 | 2025-01-15 |      250.00 | Completed |
| John Doe      | johndoe@example.com |       3 | 2025-01-17 |      300.00 | Cancelled |
| Jane Smith    | janesmith@example.com |       2 | 2025-01-16 |      150.00 | Pending   |
| Alice Johnson | alicej@example.com  |    NULL | NULL       |        NULL | NULL      |
| Bob Brown     | bobb@example.com    |       4 | 2025-01-18 |      500.00 | Completed |
+---------------+---------------------+---------+------------+-------------+-----------+
5 rows in set (0.04 sec)

mysql>
```

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.

Ans:

- **Union:**

```
mysql> SELECT
    ->     customers.Name AS Name,
    ->     customers.Email AS Email,
    ->     'Last 30 Days Orders' AS OrderCategory
    -> FROM
    ->     customers
    -> INNER JOIN
    ->     orders
    -> ON
    ->     customers.CustomerID = orders.CustomerID
    -> WHERE
    ->     orders.OrderDate >= CURDATE() - INTERVAL 30 DAY
    ->
    -> UNION
    ->
    -> SELECT
    ->     customers.Name AS Name,
    ->     customers.Email AS Email,
    ->     'No Orders' AS OrderCategory
    -> FROM
    ->     customers
    -> LEFT JOIN
    ->     orders
    -> ON
    ->     customers.CustomerID = orders.CustomerID
    -> WHERE
    ->     orders.OrderID IS NULL;
+---------------+----------------------+---------------------+
| Name          | Email                | OrderCategory       |
+---------------+----------------------+---------------------+
| John Doe      | johndoe@example.com  | Last 30 Days Orders |
| Jane Smith    | janesmith@example.com | Last 30 Days Orders |
| Bob Brown     | bobb@example.com     | Last 30 Days Orders |
| Emily Davis   | emilyd@example.com   | Last 30 Days Orders |
| Michael Lee   | michaell@example.com | Last 30 Days Orders |
| Alice Johnson | alicej@example.com   | No Orders           |
| Sarah Green   | sarahg@example.com   | No Orders           |
| David King    | davidk@example.com   | No Orders           |
+---------------+----------------------+---------------------+
8 rows in set (0.13 sec)

mysql>
```

- **SUBQUERY:**

```
mysql> SELECT
    ->     customers.Name AS CustomerName,
    ->     customers.Email AS CustomerEmail,
    ->     orders.TotalAmount AS OrderAmount
    -> FROM
    ->     orders
    -> INNER JOIN
    ->     customers
    -> ON
    ->     customers.CustomerID = orders.CustomerID
    -> WHERE
    ->     orders.TotalAmount > (SELECT AVG(TotalAmount) FROM orders);
+--------------+----------------------+-------------+
| CustomerName | CustomerEmail        | OrderAmount |
+--------------+----------------------+-------------+
| Bob Brown    | bobb@example.com     |      500.00 |
| Michael Lee  | michaell@example.com |      400.00 |
| Bob Brown    | bobb@example.com     |      600.00 |
+--------------+----------------------+-------------+
3 rows in set (0.16 sec)

mysql>
```

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.

Ans:

- ***Commit:***

```
mysql> Select * from orders;
+---------+------------+------------+-------------+-----------+
| OrderID | CustomerID | OrderDate  | TotalAmount | Status    |
+---------+------------+------------+-------------+-----------+
|       1 |          1 | 2025-01-15 |      250.00 | Completed |
|       2 |          2 | 2025-01-16 |      150.00 | Pending   |
|       3 |          1 | 2025-01-17 |      300.00 | Cancelled |
|       4 |          4 | 2025-01-18 |      500.00 | Completed |
|      13 |          8 | 2025-01-19 |      200.00 | Completed |
|      14 |          9 | 2025-01-19 |      400.00 | Pending   |
|      15 |          4 | 2025-01-20 |      100.00 | Completed |
|      16 |          4 | 2025-01-21 |      600.00 | Cancelled |
+---------+------------+------------+-------------+-----------+
8 rows in set (0.00 sec)

mysql>
mysql> INSERT INTO orders (CustomerID, OrderDate, TotalAmount,status)
    -> VALUES (1, CURDATE(), 500.00,"Pending");
Query OK, 1 row affected (0.04 sec)

mysql>
mysql>
mysql> COMMIT;
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> Select * from orders;
+---------+------------+------------+-------------+-----------+
| OrderID | CustomerID | OrderDate  | TotalAmount | Status    |
+---------+------------+------------+-------------+-----------+
|       1 |          1 | 2025-01-15 |      250.00 | Completed |
|       2 |          2 | 2025-01-16 |      150.00 | Pending   |
|       3 |          1 | 2025-01-17 |      300.00 | Cancelled |
|       4 |          4 | 2025-01-18 |      500.00 | Completed |
|      13 |          8 | 2025-01-19 |      200.00 | Completed |
|      14 |          9 | 2025-01-19 |      400.00 | Pending   |
|      15 |          4 | 2025-01-20 |      100.00 | Completed |
|      16 |          4 | 2025-01-21 |      600.00 | Cancelled |
|      17 |          1 | 2025-01-20 |      500.00 | Pending   |
+---------+------------+------------+-------------+-----------+
9 rows in set (0.00 sec)
```

- ***Rollback:***

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> UPDATE orders
    -> SET TotalAmount = 999.00
    -> WHERE OrderID = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> Select * from orders;
+---------+------------+------------+-------------+-----------+
| OrderID | CustomerID | OrderDate  | TotalAmount | Status    |
+---------+------------+------------+-------------+-----------+
|       1 |          1 | 2025-01-15 |      999.00 | Completed |
|       2 |          2 | 2025-01-16 |      150.00 | Pending   |
|       3 |          1 | 2025-01-17 |      300.00 | Cancelled |
|       4 |          4 | 2025-01-18 |      500.00 | Completed |
|      13 |          8 | 2025-01-19 |      200.00 | Completed |
|      14 |          9 | 2025-01-19 |      400.00 | Pending   |
|      15 |          4 | 2025-01-20 |      100.00 | Completed |
|      16 |          4 | 2025-01-21 |      600.00 | Cancelled |
|      17 |          1 | 2025-01-20 |      550.00 | Pending   |
+---------+------------+------------+-------------+-----------+
9 rows in set (0.00 sec)

mysql>
mysql> ROLLBACK;
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> Select * from orders;
+---------+------------+------------+-------------+-----------+
| OrderID | CustomerID | OrderDate  | TotalAmount | Status    |
+---------+------------+------------+-------------+-----------+
|       1 |          1 | 2025-01-15 |      250.00 | Completed |
|       2 |          2 | 2025-01-16 |      150.00 | Pending   |
|       3 |          1 | 2025-01-17 |      300.00 | Cancelled |
|       4 |          4 | 2025-01-18 |      500.00 | Completed |
|      13 |          8 | 2025-01-19 |      200.00 | Completed |
|      14 |          9 | 2025-01-19 |      400.00 | Pending   |
|      15 |          4 | 2025-01-20 |      100.00 | Completed |
|      16 |          4 | 2025-01-21 |      600.00 | Cancelled |
|      17 |          1 | 2025-01-20 |      550.00 | Pending   |
+---------+------------+------------+-------------+-----------+
9 rows in set (0.00 sec)
```

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.

Ans:

- **Preforming INSERT operations After starting transaction create SAVEPOINTS:**

```
START TRANSACTION;

INSERT INTO orders (CustomerID, OrderDate, TotalAmount, Status)
VALUES (10, '2025-01-20', 2150.00, 'Completed');
INSERT INTO orders (CustomerID, OrderDate, TotalAmount, Status)
VALUES (1, '2025-01-20', 550.00, 'Pending');

SAVEPOINT Monday;

INSERT INTO orders (CustomerID, OrderDate, TotalAmount, Status)
VALUES (11, '2025-01-21', 3200.00, 'Pending');
INSERT INTO orders (CustomerID, OrderDate, TotalAmount, Status)
VALUES (10, '2025-01-20', 550.00, 'Completed');

SAVEPOINT Tuesday;

INSERT INTO orders (CustomerID, OrderDate, TotalAmount, Status)
VALUES (8, '2025-01-22', 1520.00, 'Completed');
INSERT INTO orders (CustomerID, OrderDate, TotalAmount, Status)
VALUES (8, '2025-01-22', 120.00, 'Completed');
SAVEPOINT Wednesday;
```

- **Table current status:**

```
mysql> select * from  orders;
+---------+------------+------------+-------------+-----------+
| OrderID | CustomerID | OrderDate  | TotalAmount | Status    |
+---------+------------+------------+-------------+-----------+
|       1 |          1 | 2025-01-15 |      250.00 | Completed |
|       2 |          2 | 2025-01-16 |      150.00 | Pending   |
|       3 |          1 | 2025-01-17 |      300.00 | Cancelled |
|       4 |          4 | 2025-01-18 |      500.00 | Completed |
|      13 |          8 | 2025-01-19 |      200.00 | Completed |
|      14 |          9 | 2025-01-19 |      400.00 | Pending   |
|      15 |          4 | 2025-01-20 |      100.00 | Completed |
|      16 |          4 | 2025-01-21 |      600.00 | Cancelled |
|      17 |          1 | 2025-01-20 |      550.00 | Pending   |
|      18 |         10 | 2025-01-20 |     2150.00 | Completed |
|      19 |          1 | 2025-01-20 |      550.00 | Pending   |
|      20 |         11 | 2025-01-21 |     3200.00 | Pending   |
|      21 |         10 | 2025-01-20 |      550.00 | Completed |
|      22 |          8 | 2025-01-22 |     1520.00 | Completed |
|      23 |          8 | 2025-01-22 |      120.00 | Completed |
+---------+------------+------------+-------------+-----------+
15 rows in set (0.00 sec)
```

- ***Rolling back to 'TUESDAY SAVEPOINT' and getting Table Status:***

```
mysql> ROLLBACK TO SAVEPOINT Tuesday;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from  orders;
+---------+------------+------------+-------------+-----------+
| OrderID | CustomerID | OrderDate  | TotalAmount | Status    |
+---------+------------+------------+-------------+-----------+
|       1 |          1 | 2025-01-15 |      250.00 | Completed |
|       2 |          2 | 2025-01-16 |      150.00 | Pending   |
|       3 |          1 | 2025-01-17 |      300.00 | Cancelled |
|       4 |          4 | 2025-01-18 |      500.00 | Completed |
|      13 |          8 | 2025-01-19 |      200.00 | Completed |
|      14 |          9 | 2025-01-19 |      400.00 | Pending   |
|      15 |          4 | 2025-01-20 |      100.00 | Completed |
|      16 |          4 | 2025-01-21 |      600.00 | Cancelled |
|      17 |          1 | 2025-01-20 |      550.00 | Pending   |
|      18 |         10 | 2025-01-20 |     2150.00 | Completed |
|      19 |          1 | 2025-01-20 |      550.00 | Pending   |
|      20 |         11 | 2025-01-21 |     3200.00 | Pending   |
|      21 |         10 | 2025-01-20 |      550.00 | Completed |
+---------+------------+------------+-------------+-----------+
13 rows in set (0.00 sec)

mysql>
```

- ***Finally Committing changes:***

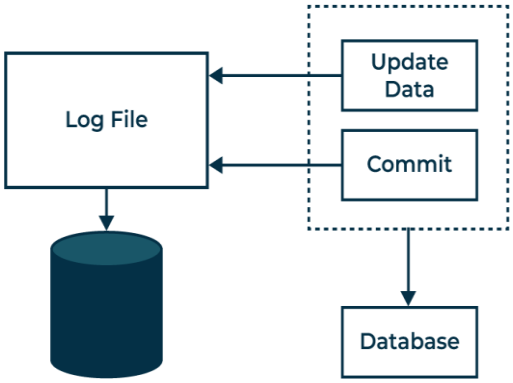
```
mysql> Commit;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from  orders;
+---------+------------+------------+-------------+-----------+
| OrderID | CustomerID | OrderDate  | TotalAmount | Status    |
+---------+------------+------------+-------------+-----------+
|       1 |          1 | 2025-01-15 |      250.00 | Completed |
|       2 |          2 | 2025-01-16 |      150.00 | Pending   |
|       3 |          1 | 2025-01-17 |      300.00 | Cancelled |
|       4 |          4 | 2025-01-18 |      500.00 | Completed |
|      13 |          8 | 2025-01-19 |      200.00 | Completed |
|      14 |          9 | 2025-01-19 |      400.00 | Pending   |
|      15 |          4 | 2025-01-20 |      100.00 | Completed |
|      16 |          4 | 2025-01-21 |      600.00 | Cancelled |
|      17 |          1 | 2025-01-20 |      550.00 | Pending   |
|      18 |         10 | 2025-01-20 |     2150.00 | Completed |
|      19 |          1 | 2025-01-20 |      550.00 | Pending   |
|      20 |         11 | 2025-01-21 |     3200.00 | Pending   |
|      21 |         10 | 2025-01-20 |      550.00 | Completed |
+---------+------------+------------+-------------+-----------+
13 rows in set (0.00 sec)
```

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.

Ans:

- **Transactions Logs:**
  - It records all **database transactions with changes made previously and metadata**.
  - When System failure occur, these are helpful in **data recovery**.
  - This **will maintain ACID properties** as transactions are intact even a burst of system.



- **Transaction logs in DBMS:**
  - Consists the type of operation performed (e.g., **INSERT, UPDATE, DELETE**)
  - **Data affected by transactions**.
  - It also has **data before and after any transaction**.
- **Role & working:**
  - It can **recover uncommitted** data.
  - It can have **point in time recovery**.
  - **All audits of transactions are stored**.
  - Very useful in crash recovery as using it we can **rollback or roll forward** transactions.
- **Logs In MYSQL:**
  - **In MYSQL logs generally found in binary format**
  - **Below image shows current binary state:**

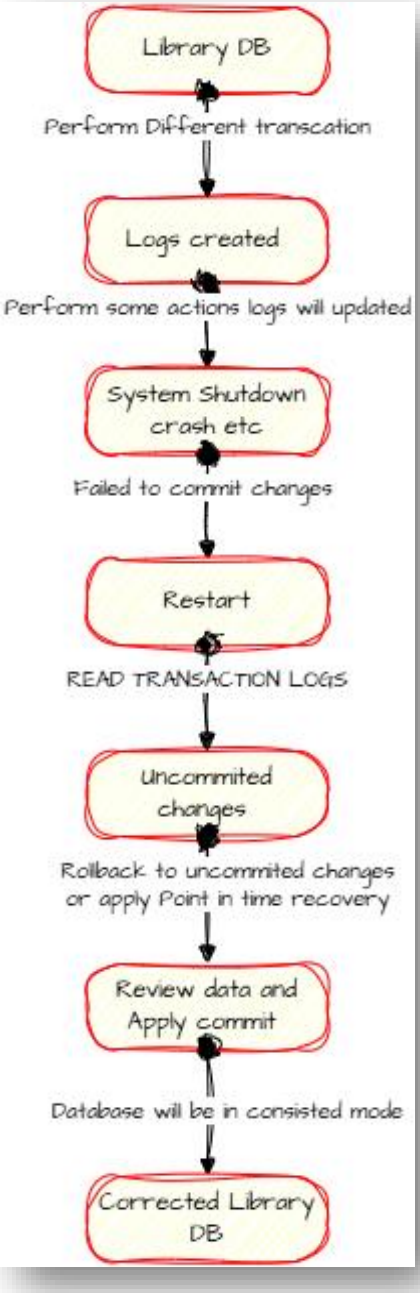
```
mysql> Show master status
    -> ;
+--------------------------+----------+--------------+------------------+-------------------+
| File                     | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+--------------------------+----------+--------------+------------------+-------------------+
| DESKTOP-RILG1IE-bin.000008 |  2118729 |              |                  |                   |
+--------------------------+----------+--------------+------------------+-------------------+
1 row in set (0.01 sec)
```

  - **Image: getting older binary logs**

```
mysql> show binary logs;
+--------------------------+-----------+-----------+
| Log_name                 | File_size | Encrypted |
+--------------------------+-----------+-----------+
| DESKTOP-RILG1IE-bin.000001 |       180 | No        |
| DESKTOP-RILG1IE-bin.000002 |   2098370 | No        |
| DESKTOP-RILG1IE-bin.000003 |       157 | No        |
| DESKTOP-RILG1IE-bin.000004 |       157 | No        |
| DESKTOP-RILG1IE-bin.000005 |       157 | No        |
| DESKTOP-RILG1IE-bin.000006 |       157 | No        |
| DESKTOP-RILG1IE-bin.000007 |       157 | No        |
| DESKTOP-RILG1IE-bin.000008 |   2118729 | No        |
+--------------------------+-----------+-----------+
8 rows in set (0.06 sec)
```

Scenario for Library Management application:

In Library management we transact things like adding, issuing, returning books also there might be Fine and membership concept. Now, imagine while Imposing a Fine on returning late of book to a customer the database servers are stopped unexpectedly.



- **Impact:**
  - **Data Loss for book, fine status and customer Data as well.**
  - **A business Loss to client as well as users.**

- **How Transaction Logs will help:**
  - Let's imagine that system failed at 12pm.
  - Transactions table will have data at POINT OF TIME. So Last Transaction can be found in Logs before 12pm.
  - So now when System restarts, we can find these logs and can be recover and rollback to last committed or uncommitted data.
  - Modern DB's automatically rollback last uncommitted data using this transaction logs.

  After complete recovery process DB will be restored to consistent state, and if process before failure is committed it will return to that else If the transactions were uncommitted, they will be rolled back, ensuring that no half-processed data remains. The library database will be consistent and accurate, reflecting only completed transactions.

- **Summary of process:**
  - Some transactions are made and recorded to transaction logs.
  - System Failure occurs.
  - Last transaction is already in logs.
  - System started.
  - System or developer found the last change there In log and rollback to the last saved or uncommitted data.
  - End.

Ref:

- https://media.geeksforgeeks.org/wp-content/uploads/20231110122147/Log-Based-recovery-in-DBMS.png
- **Draw.io for creating diagrams.**