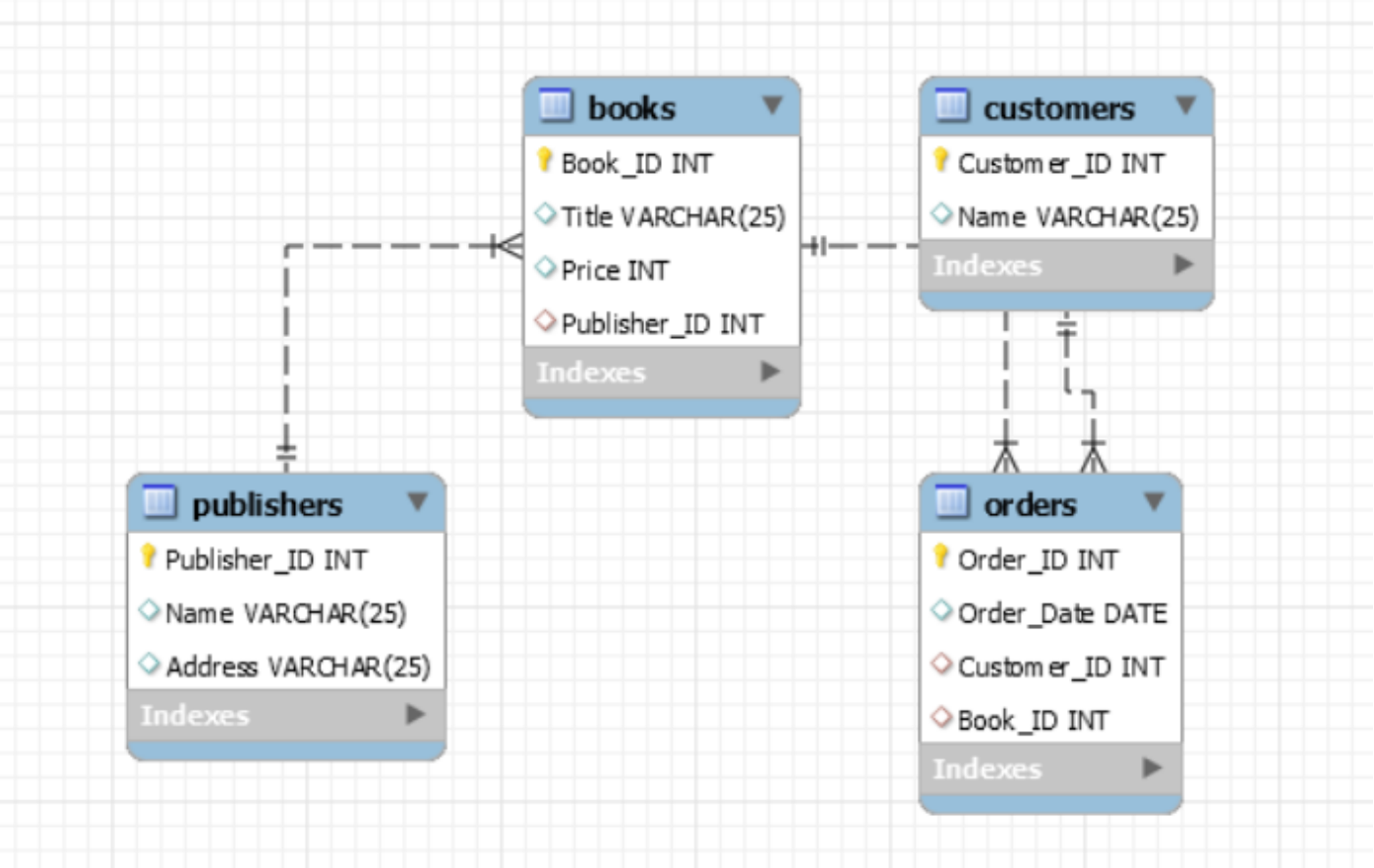
Assignment 1:

Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form.




```
use wipro_project;
CREATE TABLE Publishers (
  Publisher_ID INT PRIMARY KEY,
  Name VARCHAR(25),
  Address VARCHAR(25)
);

INSERT INTO Publishers VALUES
(1, 'Scribner', '123 road'),
(2, 'JB Lippincott', '432 road');
```

```
CREATE TABLE Books (
  Book_ID INT PRIMARY KEY,
  Title varchar(25),
  Price int,
  Publisher_ID int,
  FOREIGN KEY (Publisher_ID) REFERENCES Publishers(Publisher_ID)
);

INSERT INTO  Books values(1, 'The Great', 100, 1),
(2, 'To Kill', 200, 2),
(3, 'Hello sir', 100, 1);

CREATE TABLE Customers (
  Customer_ID INT PRIMARY KEY,
  Name VARCHAR(25)
);

INSERT INTO Customers VALUES
(1, 'Amrut'),
(2, 'Amit');

CREATE TABLE Orders (
  Order_ID INT PRIMARY KEY,
  Order_Date DATE,
  Customer_ID INT,
  Book_ID INT,
  FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID),
  FOREIGN KEY (Book_ID) REFERENCES Books(Book_ID)
);
```

```
INSERT INTO Orders VALUES
(1, '2024-05-01', 1, 1),
(2, '2024-05-02', 2, 2),
(3, '2024-05-03', 1, 3);
```

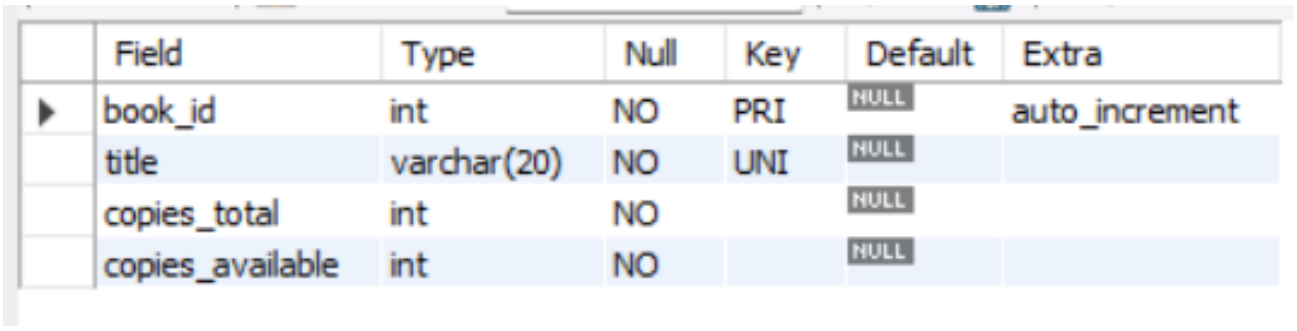
Assignment 2:

Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and CHECK. Include primary and foreign keys to establish relationships between tables.

```
CREATE TABLE Book (
    book_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(20) NOT NULL UNIQUE,
    copies_total INT NOT NULL CHECK (copies_total > 0),
    copies_available INT NOT NULL CHECK (copies_available >= 0)
);
```

```
CREATE TABLE Borrowing (
    borrowing_id INT PRIMARY KEY AUTO_INCREMENT,
    book_id INT NOT NULL,
    borrower_id INT NOT NULL,
    borrow_date DATE NOT NULL,
    return_date DATE NOT NULL,
    FOREIGN KEY (book_id) REFERENCES Book(book_id)
);
```

Book table:



Field	Type	Null	Key	Default	Extra
book_id	int	NO	PRI	NULL	auto_increment
title	varchar(20)	NO	UNI	NULL	
copies_total	int	NO		NULL	
copies_available	int	NO		NULL	

Borrow table:

	Field	Type	Null	Key	Default	Extra
▶	borrowing_id	int	NO	PRI	NULL	auto_increment
	book_id	int	NO	MUL	NULL	
	borrower_id	int	NO		NULL	
	borrow_date	date	NO		NULL	
	return_date	date	NO		NULL	

Assignment 3:

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.

Atomicity:

All operations within a transaction are successfully completed or none are ensuring indivisibility.

Consistency:

Ensure that the database remains in a valid state before and after the transaction maintaining data integrity.

Insolation:

Ensure that transactions operate independently of each other, preventing transactions from interfacing with each other intermediate states to maintain data consistency.

Durability:

Durability ensures that once a transaction is committed, its changes are permanently saved and will not be lost, even in the event of system failures, ensuring data persistence.

ACCOUNT TABLE

```
CREATE TABLE Bank (
   id INT PRIMARY KEY,
   balance DECIMAL(10, 2)
);
--Inserting values
INSERT INTO Bank VALUES (1, 2345),(2, 6543),(3, 1234),(4, 7654);
```

```
--Start transaction
START TRANSACTION;
-- Set the isolation level to READ COMMITTED
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
--The above query does not execute because TRANSACTION is in start mode.
--use ROLLBACK to stop transaction


--Lock the row for update
UPDATE accounts SET balance = 1000 WHERE id = 1;
```

Assignment 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.

```
CREATE DATABASE library;


USE library;


CREATE TABLE records (
     id INT PRIMARY KEY AUTO_INCREMENT,
   title VARCHAR(25) NOT NULL,
   author_name  VARCHAR(25) NOT NULL
);


ALTER TABLE records ADD COLUMN  price int;


DESC records;
```

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	title	varchar(25)	NO		NULL	
	author_name	varchar(25)	NO		NULL	

```
--Drop table.

DROP TABLE records;
```

Assignment 5:

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.

```
CREATE TABLE records (
   id INT PRIMARY KEY AUTO_INCREMENT,
   title VARCHAR(25) NOT NULL,
   author_name VARCHAR(25) NOT NULL
);
```

```
CREATE INDEX index_name ON records(title);
```

```
SELECT * FROM records WHERE title="Java";
```

```
DROP INDEX index_name ON records;
```

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up search.

Assignment 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 a new user with password
CREATE USER 'user'@'localhost' IDENTIFIED BY 'Basu@123';

-- Grant privileges to the new user on wipro_project
GRANT SELECT, INSERT, UPDATE ON wipro_project.* TO 'user'@'localhost';

-- Apply the changes
FLUSH PRIVILEGES;

-- Revoke UPDATE privilege from 'user' on wipro_project
REVOKE UPDATE ON wipro_project.* FROM 'user'@'localhost';

-- Apply the changes
FLUSH PRIVILEGES;

-- Drop the user 'user'
DROP USER 'user'@'localhost';

-- Apply the changes
FLUSH PRIVILEGES;
```

Assignment 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.

```
CREATE TABLE Book (
    book_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(20) NOT NULL UNIQUE,
    copies_total INT NOT NULL CHECK (copies_total > 0),
    copies_available INT NOT NULL CHECK (copies_available >= 0)
);


CREATE TABLE Borrowing (
    borrowing_id INT PRIMARY KEY AUTO_INCREMENT,
    book_id INT NOT NULL,
    borrower_id INT NOT NULL,
    borrow_date DATE NOT NULL,
    return_date DATE NOT NULL,
    FOREIGN KEY (book_id) REFERENCES Book(book_id)
);
```

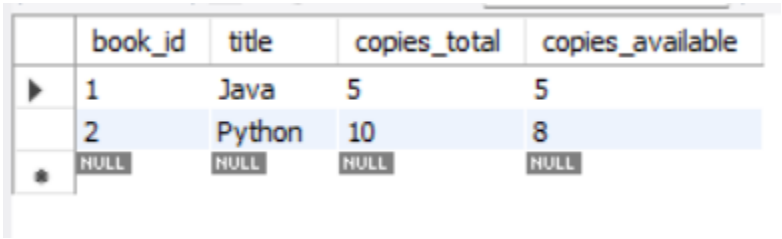
INSERT INTO Book VALUES (1,'Java', 5, 5),(2,'Python',10,9);

book_id	title	copies_total	copies_available
1	Java	5	5
2	Python	10	9
NULL	NULL	NULL	NULL

INSERT INTO Borrowing VALUES (1, 2, 20,'2024-05-15', '2024-06-15'),(2,2,21,'2024-05-15','2024-05-20');

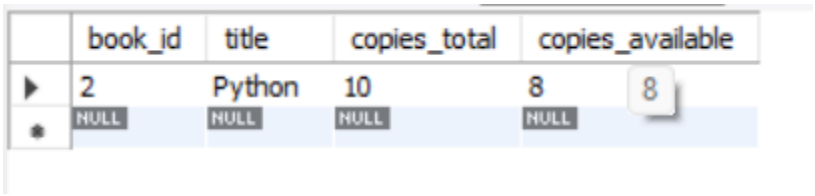
borrowing_id	book_id	borrower_id	borrow_date	return_date
1	2	20	2024-05-15	2024-06-15
2	2	21	2024-05-15	2024-05-20
NULL	NULL	NULL	NULL	NULL

UPDATE Book SET copies_available = 8 WHERE book_id = 2;



book_id	title	copies_total	copies_available
1	Java	5	5
2	Python	10	8
NULL	NULL	NULL	NULL

DELETE FROM Book WHERE title=’java’;



book_id	title	copies_total	copies_available
2	Python	10	8
NULL	NULL	NULL	NULL

LOAD DATA INFILE 'D:/Wipro/Borrowing.csv' INTO TABLE Borrowing

FIELDS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\r\n'

IGNORE 1 ROWS;ss

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.

```
CREATE TABLE customer(name varchar(20),
                      age int,
                     Phone varchar(10),
                     Email varchar(25),
                     city varchar(20)
                    );
```

	Field	Type	Null	Key	Default	Extra
▶	name	varchar(20)	YES		NULL	
	age	int	YES		NULL	
	Phone	varchar(10)	YES		NULL	
	Email	varchar(25)	YES		NULL	
	city	varchar(20)	YES		NULL	

```
INSERT INTO customer VALUES('Basanagoud',22,'7265896148','basu@gmail.com','Belagavi'),
                    ('Amrut',22,'9632689456','amrut@gmail.com','Bengalur'),
                    ('Amit',23,'7562486359','amit@gmail.com','Belagavi'),
                    ('Shrinivas',22,'8632569745','shri@gmail.com','Bailhongal'),
                    ('Santosh',21,'9456358965','santosh@gmail.com','Bengalur');
```

	name	age	Phone	Email	city
▶	Basanagoud	22	7265896148	basu@gmail.com	Belagavi
	Amrut	22	9632689456	amrut@gmail.com	Bengalur
	Amit	23	7562486359	amit@gmail.com	Belagavi
	Shrinivas	22	8632569745	shri@gmail.com	Bailhongal
	Santosh	21	9456358965	santosh@gmail.com	Bengalur

```
SELECT name, email FROM customer WHERE city = 'Bengalur';
```

	name	email
▶	Amrut	amrut@gmail.com
	Santosh	santosh@gmail.com

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.

```
CREATE TABLE customers(C_ID int PRIMARY KEY,
                Name VARCHAR(20),
                Address VARCHAR(20)
);


INSERT INTO customers VALUES(1,'Basanagoud','Bengaluru'),
                (2,'Amrut','Bengaluru'),
                (3,'Amit','Belagavi'),
                (4,'Shrinivas','Bailhongal'),
                (5,'Santosh','Bengaluru');
```

name	age	Phone	Email	city
Basanagoud	22	7265896148	basu@gmail.com	Belagavi
Amrut	22	9632689456	amrut@gmail.com	Bengalur
Amit	23	7562486359	amit@gmail.com	Belagavi
Shrinivas	22	8632569745	shri@gmail.com	Bailhongal
Santosh	21	9456358965	santosh@gmail.com	Bengalur

```
CREATE TABLE orders(O_ID int PRIMARY KEY,
                C_ID INT,
                Product_count int,
                FOREIGN KEY(C_ID) REFERENCES customers(C_ID)
);
INSERT INTO orders VALUES(1,1,4), (2,3,1), (3,2,1),
                (4,1,3),
                (5,2,2),
                (6,4,1);
```

	O_ID	C_ID	Product_count
▶	1	1	4
	2	3	1
	3	2	1
	4	1	3
	5	2	2
	6	4	1
*	NULL	NULL	NULL

```
SELECT c.C_ID,
    c.Name,
    c.Address
    FROM customers c
        LEFT JOIN orders o ON c.C_ID=o.C_ID;
```

	C_ID	Name	Address
▶	1	Basanagoud	Bengaluru
	1	Basanagoud	Bengaluru
	2	Amrut	Bengaluru
	2	Amrut	Bengaluru
	3	Amit	Belagavi
	4	Shrinivas	Bailhongal
	5	Santosh	Bengaluru

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 * FROM customers;

C_ID	Name	Address	value
1	Basanagoud	Bengaluru	NULL
2	Amrut	Bengaluru	NULL
3	Amit	Belagavi	NULL
4	Shrinivas	Bailhongal	NULL
5	Santosh	Bengaluru	NULL
NULL	NULL	NULL	NULL

SELECT * FROM orders;

O_ID	C_ID	Product_count
1	1	4
2	3	1
3	2	1
4	1	3
5	2	2
6	4	1
NULL	NULL	NULL

SELECT C_ID, O_ID

FROM orders

WHERE Product_count > (SELECT AVG(Product_count) FROM orders);

C_ID	O_ID
1	1
1	4
NULL	NULL

SELECT C_ID, Name FROM customers

UNION

SELECT O_ID, Product_count FROM orders;

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;

INSERT INTO orders VALUES(7,2,3);

O_ID	C_ID	Product_count
1	1	4
2	3	1
3	2	1
4	1	3
5	2	2
6	4	1
7	2	3
NULL	NULL	NULL

commit;

SELECT * FROM Products;

ProductID	ProductName	Price
1	Phone	15000
2	Laptop	50000
3	TV	20000
4	Byke	100000
NULL	NULL	NULL

ProductID	ProductName	Price
1	Phone	15000
2	Laptop	50000
3	TV	20000
4	Byke	100000
NULL	NULL	NULL

UPDATE products SET price = 120000 WHERE ProductName='Byke';

ProductID	ProductName	Price
1	Phone	15000
2	Laptop	50000
3	TV	20000
4	Byke	120000
NULL	NULL	NULL

ROLLBACK;

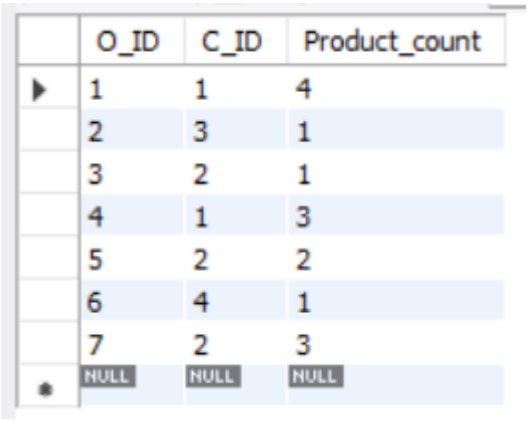
ProductID	ProductName	Price
1	Phone	15000
2	Laptop	50000
3	TV	20000
4	Byke	100000
NULL	NULL	NULL

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.

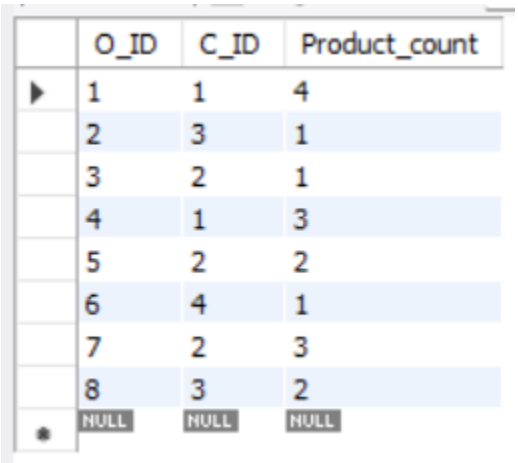
BEGIN;

select*from Products;



O_ID	C_ID	Product_count
1	1	4
2	3	1
3	2	1
4	1	3
5	2	2
6	4	1
7	2	3
NULL	NULL	NULL

INSERT INTO orders VALUES(8,3,2);



O_ID	C_ID	Product_count
1	1	4
2	3	1
3	2	1
4	1	3
5	2	2
6	4	1
7	2	3
8	3	2
NULL	NULL	NULL

SAVEPOINT savepoint;

ROLLBACK TO SAVEPOINT savepoint;

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.

Transaction log:

- A transaction log record all transactions executed in the database it stores information about each transaction.
- Recovery models determine how transactions are logged and cleared from the log.
- Recovery models: Simple recovery model, Full recovery model

Hypothetical scenario:

- All committed transactions are stored in a transaction log.
- Uncommitted transactions are rolled back during recovery.
- SQL server automatically replies the transaction log to restore the database to the consistent state.
- Transaction log helps recover data up to the point of the last committed transaction before the shutdown.
- No data loss occurs because the transaction log captures all changes.