**Assignment 1: Analyse 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.**

**Business Scenario:** Online Retail Store

A company operates an online retail store that sells various products to customers. The store manages inventory, processes orders, and tracks customer information. Customers can browse products, add them to their cart, and place orders for delivery.

Now, let's break down the entities, relationships, attributes, and cardinality for this scenario:

**Entities:**

1. Customer

2. Product

3. Order

4. Order Item

5. Address

**Relationships:**

1. Customer places Order (1-to-many)

2. Customer has Address(1-to-1)

3. Order contains Order Item (1-to-many)

4. Order has Shipping Address (1-to-1)

5. Order is for Product (many-to-many)

**Attributes:**

1. Customer: Customer\_ID (Primary Key), Name, Email, Phone

2. Product: Product\_ID (Primary Key), Name, Description, Price, Quantity

3. Order: Order\_ID (Primary Key), OrderDate, TotalAmount

4. Order Item: OrderItem\_ID (Primary Key), Quantity, Subtotal

5. Address: Address\_ID (Primary Key), Street, City, State, PINCode

**Cardinality:**

1. Customer places Order: One customer can place many orders, but each order is placed by only one customer. (1-to-many)

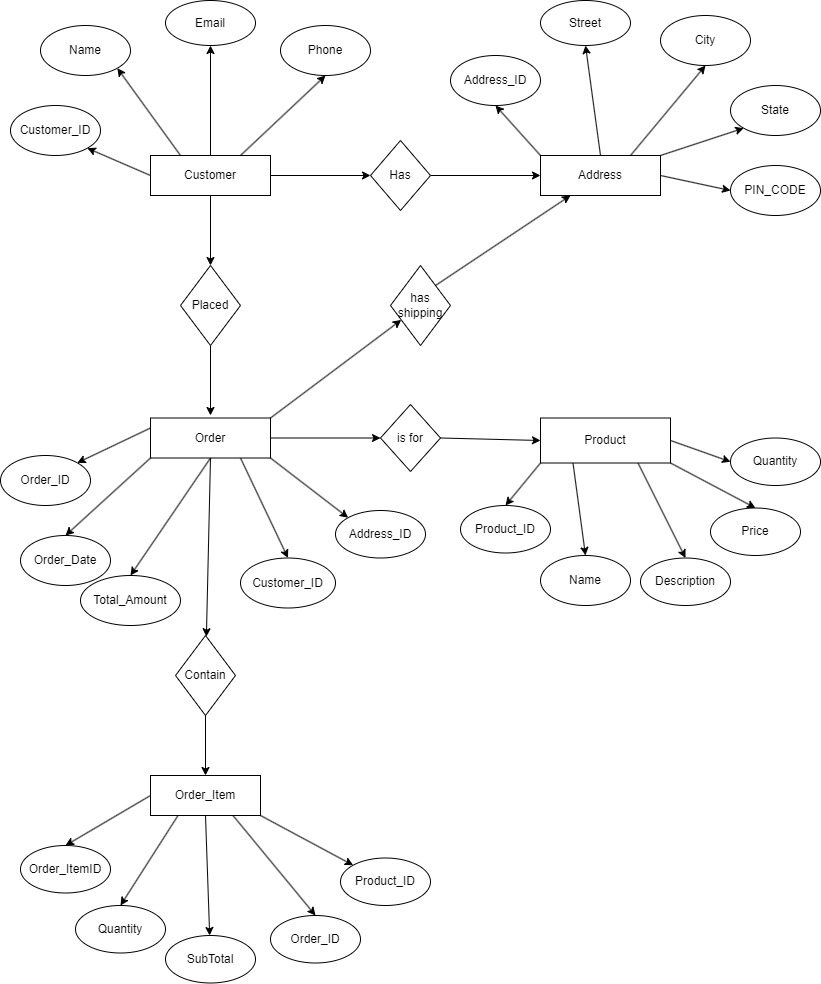
2. Customer has Address: Each customer has one address, and each address can be associated with only one customer. (1-to-1)

3. Order contains Order Item: Each order can contain multiple order items, but each order item belongs to only one order. (1-to-many)

4. Order has Shipping Address: Each order has one shipping address, and each address can be associated with only one order. (1-to-1)

5. Order is for Product: Each order can include multiple products, and each product can be in multiple orders. (many-to-many)

Now, we can create an ER diagram reflecting these entities, relationships, attributes, and cardinality.



**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.**

The ACID properties are the four key characteristics that ensure reliability and consistency in database transactions:

1. Atomicity: This means that a transaction is indivisible or all-or-nothing. Either all the operations within a transaction are successfully completed and committed, or if any operation fails, the transaction is rolled back to its original state.

2. Consistency: After a transaction is completed, the database remains in a consistent state. This ensures that the data meets all the integrity constraints, such as foreign key constraints or data type constraints.

3. Isolation: Transactions operate independently of each other. Even if multiple transactions are running simultaneously, the outcome should be the same as if they were executed serially, without interference from other transactions.

4. Durability: Once a transaction is committed, its changes are permanent and will survive system failures. The changes are stored permanently and cannot be undone.

Now, regarding the SQL statements to simulate a transaction with locking and demonstrate different isolation levels:

Suppose we have a simple banking scenario where we transfer money between two accounts. Let's start with a basic SQL transaction:

BEGIN TRANSACTION;

UPDATE accounts SET balance = balance - 100 WHERE account\_id = 123;

UPDATE accounts SET balance = balance + 100 WHERE account\_id = 456;

COMMIT;

This transaction transfers 100 from account 123 to account 456.

Demonstrating Different Isolation Levels:

We can set different isolation levels to control the visibility of data changes made by other transactions.

For example:

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

This allows a transaction to read data that has been modified by other transactions but not yet committed.

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

This allows a transaction to read only committed data. It prevents dirty reads.

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

This ensures that a transaction sees a consistent snapshot of the database throughout its duration. It prevents non-repeatable reads.

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

This is the strictest isolation level. It ensures that transactions are executed serially, one after another, to prevent any concurrency issues.

Each isolation level provides a different level of concurrency control and consistency, allowing developers to choose the appropriate level based on their application's requirements for data integrity and performance.