Task 4.

Given A Sample Unnormalised Dataset (Provided By The Instructor), Normalise It Up To 3NF.
Clearly Explain Each Step And The Rationale Behind Your Decisions.

Understanding Normalization

Normalization is the process of organizing data in a database to reduce redundancy and improve integrity. The key stages are:

- 1NF (First Normal Form): Ensure atomicity (no repeating groups).
- 2NF (Second Normal Form): Ensure no partial dependencies (every non-key column depends on the entire primary key).
- 3NF (Third Normal Form): Ensure no transitive dependencies (non-key columns depend only on the primary key).

Here is our Unnormalized dataset:

Order_id	Customer_name	Customer_Address	Product_id	Product_name	Quantity	Total_price
101	Gift Ebere	13 Ekpo Abasi	P001	Laptop	1	2000
102	John Doe	15 Eta Agbor	P002	Charger	3	600
103	Gift Ebere	13 Ekpo Abasi	P003	Monitor	2	7000
104	Asuquo Ekpo	45 Goldie	P004	Tablet	4	4000

Probelms with this dataset:
1. Data reduncdancy (repetition of information)
   - Customer name and address repeats if a customer places multiple orders
   - Product name repeats for same product ID.
   - Solution: Separate both customer and product to different tables.

2. Data Inconsistency
   - If Asuquo changes his address, we must update every row where he appears.
   - If someone mispells "Tablet" in one row, it breaks consistency.

     Solution: Store Customer details in one place and link them using a unique Customer ID.

3. Insertion Anomalies (Difficult to Add New Data)
   - Suppose a new product is added but hasn't been ordered yet.
     - Problem: We cannot insert the product without an order.

Solution: Create a separate Product table.

4. Deletion Anomalies (Accidental Data Loss)
   - If we delete Order 102, we also lose information about the charger product.

Solution : Store Products separately so deleting an order doesn't remove product data.

Converting To 1NF:

- Normalize customer table:- Customers should not be repeated
- Normalize product table:- Products should not be embeded into Orders

Customers Table:

Customer_Id	Customer_Name	Customer_Address
1	Gift Ebere	13 Ekpo Abasi
2	John Doe	15 Eta Agbor
3	Asuquo Ekpo	45 Goldie

Products Table:

Product_Id	Product_Name	Unit_Price
P001	Laptop	2000
P002	Charger	200
P003	Monitor	4500
P004	Tablet	1000

Order Table:

Order_Id	Customer_Id
101	1
102	2
103	1
104	3

Order_details Table:

Order_Id	Product_Id	Quantity	Total_Price
101	P001	1	2000
102	P002	3	600
103	P003	2	7000
104	P004	4	4000

Converting to 2NF:

- Must be in 1NF
- Remove partail dependency where non-key column depends only on part of a composite key.

Since Customer_Name and Customer_Address depend only on Customer_ID, they are already in a separate table.

The Product_Name and Unit_Price depend only on Product_ID, so they are also separate.

Total_price column of the Order_details Table depends on the Product_id (derived from unit_price * quantity).

We have to remove the Total_price to meet the rule of 2NF.

Order_details Table after 2NF:

Order_Id	Product_Id	Quantity
101	P001	1
102	P002	3
103	P003	2
104	P004	4

Reason for Change:

- Total_Price is derived from Unit_Price * Quantity, so it shouldn't be stored. Instead, it should be calculated when needed

Converting to 3NF:

- Must be in 2NF.
- Remove transitive dependencies (a column should depend only on the primary key, not on another non-key column).

Indentifying Transitive Dependencies

- The Orders table currently links customers, but Customer_Address depends on Customer_ID, not Order_ID.
- The Products table contains Product_Name, which depends on Product_ID.
- The Order_Details table contains Quantity, which is fine since it's directly related to the order.

From 2NF tables, our structured table properly adher to 3NF rules.