Objective: The objective of this lab is to practice designing databases in a fundamentally sound manner by ensuring that our tables are normalized to decrease redundancy and to increase efficiency.

Instructions:
Given the following database table, normalize the table up to the 3rd normal form. Make sure you show your steps (1st normal form, 2nd normal form), please do not just show me the resultant 3rd normal form tables. For each step you are required to show what the table(s) structure will look like (without the data) and provide a brief one paragraph description.
Finally, briefly comment (one paragraph) on whether this is a good database design at the very end. Would this database benefit from further normalization? Research BCNF, 4th Normal Form, and 5th Normal Form to be able to answer this question.

Note: Part of the difficulty here is determining what the primary key(s) will be!

Department	Product Code	Aisle Number	Price	Unit of Measure
Produce	4081	1	0.35	lb
Produce	4027	1	0.90	lb
Produce	4108	1	1.99	lb
Butcher	331100	5	1.50	lb
Butcher	331105	5	2.40	lb
Butcher	332110	5	5.00	lb
Freezer	411100	6	1.00	ea
Freezer	521101	6	1.00	ea
Freezer	866503	6	5.00	ea
Freezer	866504	6	5.00	ea

Normalization: A database design technique to store the data efficiently and logically by reducing redundancy and eliminating any undesirable characteristics.

Step 1:

FIRST NORMAL FORM or 1NF

For a table to be in the First Normal Form, we must make sure that all the columns have single valued attributes/columns and the values stored in each column are of the same domain. Also, we all the column names in the table should be unique. For the first normal form, the order in which the data is stored doesn't matter. So, the given table can be the same for this. We can consider *Product Code* as the primary key, as it can uniquely identify each record in the table.

Hence, the table would look like the same –

Product Code	Department	Aisle Number	Price	Unit of Measure

Step 2:

SECOND NORMAL FORM or 2NF

The table is said to be obeying the 2nd normal form if it complies with the rules of 1st normal form and it shouldn't have partial dependency. Any attribute in the table is partial dependent on a composite primary key if it depends only on a portion of the primary key, not on the whole primary key. In the case of our simple table, it is obvious that we will not be able to make it into 2NF, as our primary key is not composite. A table with a simple primary key is automatically in second normal form.

So, our table in 2NF would be the same as 1NF

Step 3:

THIRD NORMAL FORM or 3NF

Rule: It must be in 2NF and it must not have any transitive dependencies. Another lesser known rule is that you cannot have any non-key attributes having FDs between each other. Not really a common occurrence at this stage but it is worth noting.

In this table, by looking at it, aisle number is functionally dependent on the department and vice versa. We need to cut that out.

Also, each the unit of measure and the department have a functional dependency. Everything in the produce department for example is measured in pounds (lb), everything in the butcher department is measured in pounds (lb), and everything in the freezer is measured by item (ea). This is a trick, see I know the answer to this exercise is already online, so I changed the unit of measure column so that it has a functional dependency on department (the version online does not have this functional dependency).

So how do we proceed, we could:

Product(productCode, price)

Department(deptName, aisleNum, unitOfMeasure)

However, the database schema above causes us to lose a relationship despite the fact that it is in 3NF. We can no longer link the product to the department (there is no relationship).

So what we could do is:

Product(productCode, price, deptName)

Department(deptName, aisleNum, unitOfMeasure)

Note, in the product table, the department name is a foreign key. Therefore, there is a link (relationship) between Product and Department.

At this point, our little example cannot be further decomposed to obtain higher types of normalization. Actually, it has already attained higher levels of normalization. However, it can achieve BCNF or Boyce-Codd Normal form if we can partition table 1 into two tables (*product code-price, product code-unit of measure*), where each column depends on another column, the other column is unique: satisfying the BCNF rules.

Traditionally, Boyce-Codd normal form is the gold standard of table design. In spite of the fact that the fourth and fifth normal forms remove additional redundancies, they are uncommon and of little practical concern. There is generally a need to make separate efforts to move into subsequent levels of normalizing data in complex databases.

additional redundancies, they are uncommon and of little practical concern. There is generally a need to make separate efforts to move into subsequent levels of normalizing data in complex databases.
