Design a database based on the given description. Your output should be a set of named tables with mock data inserted (you only need a few rows of mock data). Column names and data types must also be included.

Your database design must conform to 3NF. Beneath each table write a short paragraph explaining why the table conforms to 3NF. Make sure you note the primary key(s) and the foreign key(s) if any.

A small company is expanding into new offices spread across three buildings. To keep track of office furniture, computers, and printers, the company would like a database to keep track of everything. Each piece of office furniture, computer, or printer is given an identification number. Each item (which is either a piece of furniture, a computer, or a printer) is placed in a room of one of three buildings. The building manager is responsible for the items in their building. The building manager is identified by an employee ID, first name, last name, and DOB.

Research insertion, deletion, and update anomalies as they relate to database design. Normalized databases should not suffer from any of the aforementioned anomalies. Based on the database design you constructed in Part 2 – Q1, provide examples of how your normalized database design does not suffer from insertion, deletion, and update anomalies (one example for each anomaly is sufficient).

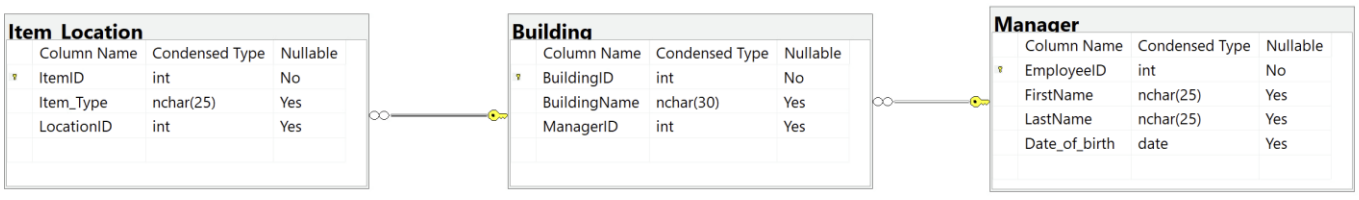
Consider the following relational instance:

CarSales(car#, date_sold, salesperson#, commission%, discount_amount)

Assume that a car may be sold my multiple salespeople thus the primary key is [car#, salesperson#]. Some additional dependencies to consider are:
date_sold → discount_amount
salesperson# → commission%
Based on the given primary key, is this relational instance in 1NF, 2NF, or 3NF? Why or why not? How would you successively normalize it completely (up to 3NF)?

Database Design:

Assumptions:
All the items are stored in a single room in each building.
Tracking items on building level rather than room level.
Database Model:



Normal Forms:

A table is said to be in 1NF when if each row is unique and no column contains multiple values.
A table is in 2NF when it is in 1NF and there are no partial functional dependencies.
When a table is in 2NF and if there are no transitive dependencies, then it is said to be in 3NF.

Manager Table:

	EmployeeID	FirstName	LastName	Date_of_birth
1	500	Scott	Eby	1990-02-01
2	501	Jake	Borlund	1983-07-08
3	502	Siri	Kantipudi	1996-06-10
4	503	Catherine	Kelly	1993-06-12

Primary Key: EmployeeID
There is only one primary key attribute. There is no chance for partial dependencies. Here, the columns FirstName, LastName and Date_of_birth depend on the EmployeeID directly. There are no dependencies among themselves implying absence of transitive dependencies. Hence, based on the above-mentioned definitions, the table is in 3NF.

Building Table:

	BuildingID	BuildingName	ManagerID
1	101	Einstein	500
2	102	Newton	503
3	103	Hawking	502

Primary Key: BuildingID
Foreign Key: ManagerID
ManagerID of Building Table references EmployeeID of Manager Table.

There is a single attribute as primary key leaving no place for partial dependencies. The attribute BuildingName completely depends upon the primary key. So, there are no transitive dependencies too. Hence this table is in 3NF.

Item_Location Table:

	ItemID	Item_Type	LocationID
1	11001	Printer	101
2	11431	Furniture	103
3	12345	Furniture	101
4	12636	Furniture	101
5	13598	Computer	103
6	14108	Computer	102
7	14418	Furniture	102
8	17108	Printer	103

Primary Key: ItemID
Foreign Key: LocationID
LocationID of Item_location table references BuildingID of Building Table.
Here also, there is a single attribute ItemID as primary key, so there are no partial dependencies. Item_type depends upon ItemID, so there are no transitive dependencies. Referring to the above-mentioned definitions of normal forms, the table is said to be in 3NF.

Insertion Anomaly:

If a new manager joins, a new record is entered in Manager table. No changes have to be made in other tables. When he/she is assigned to a new building, corresponding new record can be entered. If the tables are not normalized and manager details existed in building table, new manager entry would not have been possible.

Deletion Anomaly:

If item 11001 is discarded, we can remove the complete record without losing information about the building 101 where it was located. If the tables are not normalized and building 101 details such as name are located in the item table itself, then removing the item would have resulted in removing 101 building information too.

Update Anomaly:

If building 101's name is being changed from 'Einstein' to ' Galileo', it is to be just changed in the Building table record. The referenced number 101 in the Item_Storage table does not change which appeared in 3 records. If the tables were not normalized and the building details existed in the Item_storage table directly, name change should have been required in those 3 records.

The primary keys for the given table are car# and salesperson# combined.
Assuming a record in the table may look like
12 W 34 || 2022-01-01 || 456 || 20 || 1100

The given relational instance is in 1NF as it does not contain multiple values in any attribute and all rows are unique.
It is not in 2NF as commission% depends upon salesperson# alone leading to partial dependency.
It is not in 3NF as discount_amount depends upon date_sold leading to transitive dependency.
Referring to the above-mentioned definitions of normal forms, the table can be said to be in 1NF.

It can be converted into 3NF as follows:
Table 1 :
Car# || Salesperson# || Date_Sold
Primary key : composite key of Car# and Salesperson#
Table 2:
Salesperson# || commission%
Primary key : Salesperson#
Table 3:
Date_Sold || discount_amount
Primary key: Date_Sold

References:
https://www.w3schools.com/sql/func_mysql_substring_index.asp
https://www.softwaretestinghelp.com/mysql-stored-procedure/
https://www.w3resource.com/mysql/string-functions/mysql-regexp-function.php
https://www.javatpoint.com/mysql-not-regexp-operator
https://www.geeksforgeeks.org/mysql-regular-expressions-regexp/
https://dataschool.com/how-to-teach-people-sql/how-regex-works-in-sql/

Attribution:

*These assignments were completed by **Ravi Chandan Pandi,** and they represent his original work completed for academic purposes during his studies and self-learning purposes.*

Please note that the documents shared here are intended for educational and informational purposes only. Any unauthorized use or reproduction is strictly prohibited. If you have any questions or would like to reach out to Ravi, you can contact him on LinkedIn. [https://www.linkedin.com/in/ravichandan/].
