Question 1

Ensure 1nF:

- empName turns into {firstName, middleName, lastName}
- location turns into {strNumber, strName, cityName, provName, postalCode}. Now we have:

Employee
empID
deptID
firstName
middleName
lastName
job
salary

--

Project
projID
title
budget
funds

--

Assigned
empID
projID
role

--

Department
deptID
deptName
strNumber

Department
strName
cityName
provName
postalCode

Ensure 2nF

- Employee Dependencies:
  - {empID} -> {firstName, middleName, lastName, jobs, salary}
  - {empID, deptID} -> {firstName, middleName, lastName, job, salary}
  - The first dependency vialotes 2nf, therefore we will need to break this table into the tables below.
- Employee Dependency after Splitting:
  - {empID} -> {firstName, middleName, lastName, jobs, salary}

Employee
empID
firstName
middleName
lastName
job
salary

- Employee Departments after Splitting:
  - {empID, deptID} -> {empID, deptID} *Trivial*

Employee Departments
empID
deptID

- Project Dependencies
  - {projID} -> {title, budjet, funds}
  - These functional dependencies do not violate 2nf, the table stays the same.

Project
projID
title
budget
funds

- Assigned Dependencies
    - {empID, projID, role} -> {empID, projID, role} *Trivial*
    - These functional dependencies do not violate 2nf, the table stays the same.

Assigned
empID
projID
role

- Department Dependencies
    - {deptID} -> {deptName}
    - {deptID, strNumber, strName, cityName, provName, postalCode} -> {deptName}
    - {postalCode} -> {cityName, provName}
    - {cityName, provName } -> {postalCode}
    - {strNumber, strName, cityName, provName} -> {postalCode}
    - Assuming duplicate deptName are possible
    - Functional dependencies 1 and 2 violate 2nf. This requires splitting into two tables.
- Department After splitting:
    - {deptID} -> {deptName}

Department
deptID
deptName

- Department Location:
    - {deptID, strName, strNumber, postalCode} -> {deptID, strName, strNumber, postalCode}

Department Location
deptID
strNumber
strName
postalCode

- PostalCode:
    - {postalCode} -> {cityName, provName}

PostalCode
postalCode
cityName
provName

- Final Result after 2nf: **Employee**| -----|----- empID| firstName| middleName| lastName| job| salary|

Employee Departments

empID

deptID

Project

projID

title

budget

funds

Assigned

empID

projID

role

Department

deptID

deptName

Department Location

deptID

strNumber

strName

postalCode

PostalCode

postalCode

cityName

provName

Ensure 3nF

- Employee Dependencies:
  - {empID} -> {empID, firstName, middleName, lastName, jobs, salary}
  - No transitive dependencies therefore no 3nF violation.
- Employee Departments:
  - {empID, deptID} -> {empID, deptID} *Trivial*
  - No transitive dependencies therefore no 3nF violation.

- Project Dependencies
  - {projID} -> {projID, title, budjet, funds}
  - No transitive dependencies therefore no 3nF violation.
- Assigned Dependencies
  - {empID, projID, role} -> {empID, projID, role} *Trivial*
  - No transitive dependencies therefore no 3nF violation.
- Department Dependencies
  - {deptID} -> {deptName}
  - No transitive dependencies therefore no 3nF violation.
- Department Locations
  - {deptID, strName, strNumber, postalCode} -> {deptID, strName, strNumber, postalCode}
  - No transitive dependencies therefore no 3nF violation.
- Postal Code:
  - {postalCode} -> {cityName, provName}
  - No transitive dependencies therefore no 3nF violation.

Ensure 3.5nF

Since all tables only have 1 functional dependency, it is already in BCNF after ensuring 3nF.