**Example 1 — Course Enrollment (large, realistic)**

**Relation R**(StudentID, StudentName, CourseID, CourseName, InstructorID, InstructorName, DeptID, DeptName, Semester, Grade)

**Given FDs**

1. StudentID → StudentName
2. CourseID → CourseName, InstructorID, DeptID
3. InstructorID → InstructorName, DeptID
4. DeptID → DeptName
5. (StudentID, CourseID) → Semester, Grade

**Step A — 1NF**  
All attributes atomic → 1NF satisfied.

**Step B — Find candidate key(s)**  
Check (StudentID, CourseID)+:

* From (5): get Semester, Grade.
* From (CourseID): get CourseName, InstructorID, DeptID.
* From (StudentID): get StudentName.
* From (InstructorID): get InstructorName and DeptID.
* From (DeptID): get DeptName.  
  So (StudentID, CourseID) determines all attributes → **primary key = (StudentID, CourseID)**. No smaller key.

**Step C — 2NF (remove partial dependencies)**  
Non-key attributes depending only on part of the composite key:

* StudentName depends on StudentID (partial) → move to STUDENT.
* Everything derived from CourseID depends on part CourseID → move to COURSE.

Decompose:

* STUDENT(StudentID, StudentName) — key: StudentID
* COURSE(CourseID, CourseName, InstructorID, InstructorName, DeptID, DeptName) — key: CourseID
* ENROLL(StudentID, CourseID, Semester, Grade) — key: (StudentID, CourseID)

**Step D — 3NF (remove transitive dependencies)**  
Look at COURSE: InstructorID → DeptID and DeptID → DeptName. DeptID in COURSE may be transitively determined via Instructor. Better to create INSTRUCTOR and DEPARTMENT tables:

* INSTRUCTOR(InstructorID, InstructorName, DeptID) — key: InstructorID
* DEPARTMENT(DeptID, DeptName) — key: DeptID
* Adjust COURSE to: COURSE(CourseID, CourseName, InstructorID) — CourseID -> CourseName, InstructorID

**Step E — Check BCNF**  
Examine each relation:

* STUDENT: StudentID → StudentName; StudentID is key → BCNF.
* DEPARTMENT: DeptID → DeptName; DeptID key → BCNF.
* INSTRUCTOR: InstructorID → InstructorName, DeptID; InstructorID key → BCNF.
* COURSE: CourseID → CourseName, InstructorID; CourseID key → BCNF.
* ENROLL: (StudentID,CourseID) → Semester,Grade; key is composite → BCNF.

**Final BCNF decomposition**

* STUDENT(StudentID, StudentName)
* DEPARTMENT(DeptID, DeptName)
* INSTRUCTOR(InstructorID, InstructorName, DeptID)
* COURSE(CourseID, CourseName, InstructorID)
* ENROLL(StudentID, CourseID, Semester, Grade)

**Comments to students:** All FDs are preserved and no BCNF violations remain. This is a typical multi-step normalization: remove partials → remove transitives → ensure BCNF.

**Example 2 — Orders, Products and Suppliers (business example)**

**Relation R**(OrderID, OrderDate, CustomerID, CustomerName, CustomerAddress, ProductID, ProductName, SupplierID, SupplierName, SupplierAddress, Qty)

**FDs**

1. OrderID → OrderDate, CustomerID
2. CustomerID → CustomerName, CustomerAddress
3. ProductID → ProductName, SupplierID
4. SupplierID → SupplierName, SupplierAddress
5. (OrderID, ProductID) → Qty (each order can have multiple products; composite key)

**Step A — 1NF:** atomic attributes — OK.

**Step B — Candidate key**  
(OrderID, ProductID) determines qty and via OrderID → OrderDate, CustomerID and via ProductID → product info. So primary key = (OrderID, ProductID).

**Step C — 2NF (remove partial deps)**  
Attributes depending on OrderID alone (OrderDate, CustomerID) and those depending on ProductID alone (ProductName, SupplierID) should be separated.

**Decompose**

* ORDER\_HEADER(OrderID, OrderDate, CustomerID) — key: OrderID
* ORDER\_LINE(OrderID, ProductID, Qty) — key: (OrderID,ProductID)
* CUSTOMER(CustomerID, CustomerName, CustomerAddress)
* PRODUCT(ProductID, ProductName, SupplierID)
* SUPPLIER(SupplierID, SupplierName, SupplierAddress)

**Step D — 3NF/BCNF checks**

* CUSTOMER: CustomerID key → BCNF.
* SUPPLIER: SupplierID key → BCNF.
* PRODUCT: ProductID → SupplierID and ProductID is key for PRODUCT → BCNF.
* ORDER\_HEADER: OrderID → OrderDate, CustomerID and OrderID key → BCNF.
* ORDER\_LINE: (OrderID,ProductID) is key → BCNF.

**Note on unit price:** If business needs the price at time of order, store PriceAtOrder in ORDER\_LINE (it will be functionally dependent on (OrderID,ProductID) so BCNF still holds). If you store UnitPrice only in PRODUCT, you capture current price not historical.

**Final BCNF set**

* CUSTOMER(CustomerID, CustomerName, CustomerAddress)
* SUPPLIER(SupplierID, SupplierName, SupplierAddress)
* PRODUCT(ProductID, ProductName, SupplierID)
* ORDER\_HEADER(OrderID, OrderDate, CustomerID)
* ORDER\_LINE(OrderID, ProductID, Qty [, PriceAtOrder])

**Example 3 — Classic BCNF vs 3NF counterexample (shows tradeoff)**

**Relation R(A, B, C)**  
**FDs:**

1. AB → C
2. C → A

**Step A — 1NF:** OK.

**Step B — Candidate keys**

* Compute (A,B)+: from AB → C we get C ⇒ AB+ = {A,B,C} → so AB is a candidate key.
* Is any single attribute a key? C+ gives A (by C→A); with A alone you cannot get B, so C is not key. A alone doesn't give B or C. B alone no. So only key is AB.

**Step C — 2NF/3NF**

* No partial dependencies on AB since AB is the key and the only nontrivial FD with part of key is none.
* But C → A is a transitive-ish dependency w.r.t AB? More importantly, C → A has LHS C which is **not** a superkey; that violates BCNF.

**Step D — BCNF decomposition**  
Because C → A has left side not a superkey, decompose R on C → A:

* R1(C, A) with FD C → A (C key)
* R2(B, C) (projection)

Now check FDs:

* C → A is preserved in R1.
* But AB → C — is it preserved? AB is not contained within a single decomposed relation (R1 has A,C; R2 has B,C). Neither contains AB together, so you **cannot** check AB → C locally — the FD is not preserved by this BCNF decomposition.

**Conclusion / Teaching point:**

* After BCNF decomposition you have relations in BCNF but **dependency preservation** can be lost.
* To enforce AB → C you’d need to use constraints that span multiple tables (e.g., application logic, triggers, or check via joins), or stay at 3NF where dependency preservation is possible.

**If we kept 3NF instead:**  
3NF decomposition that preserves dependencies could be:

* R1(A,B,C) (original) — but this is not BCNF.
* Or decompose into R1(A,B,C) (with AB→C) and R2(C,A) — but this is messy. The main point: 3NF can sometimes preserve all FDs while BCNF cannot.

**Extra practice problems (for students to try)**

I. **Library**: R(BookID, Title, AuthorID, AuthorName, PublisherID, PublisherName, ShelfNo)  
FDs:

* BookID → Title, AuthorID, PublisherID, ShelfNo
* AuthorID → AuthorName
* PublisherID → PublisherName  
  Normalize to BCNF.

II. **University Projects**: R(StudentID, StudentName, ProjectID, ProjectTitle, SupervisorID, SupervisorName, SupervisorOffice)  
FDs:

* StudentID → StudentName
* ProjectID → ProjectTitle, SupervisorID
* SupervisorID → SupervisorName, SupervisorOffice
* (StudentID, ProjectID) is key for R (student may work on multiple projects)  
  Normalize to BCNF.

III. **Hospital Visits**: R(VisitID, VisitDate, PatientID, PatientName, DoctorID, DoctorName, DeptID, DeptName, Diagnosis)  
FDs:

* VisitID → VisitDate, PatientID, DoctorID, Diagnosis
* PatientID → PatientName
* DoctorID → DoctorName, DeptID
* DeptID → DeptName  
  Normalize to BCNF.

IV. **Movie Streaming**: R(UserID, UserName, MovieID, MovieTitle, GenreID, GenreName, Rating)  
FDs:

* UserID → UserName
* MovieID → MovieTitle, GenreID
* GenreID → GenreName
* (UserID, MovieID) → Rating  
  Normalize to BCNF.

V. **Supply Chain** (trickier): R(ProductID, WarehouseID, StockQty, WarehouseManagerID, ManagerName)  
FDs:

* (ProductID, WarehouseID) → StockQty, WarehouseManagerID
* WarehouseManagerID → ManagerName  
  Normalize to BCNF.

# Short answers / hints for practice

I. Library → Decompose into BOOK(BookID, Title, AuthorID, PublisherID, ShelfNo), AUTHOR(AuthorID, AuthorName), PUBLISHER(PublisherID, PublisherName). All should be BCNF if AuthorID → AuthorName and PublisherID → PublisherName are in separate relations.

II. University Projects → STUDENT(StudentID, StudentName), PROJECT(ProjectID, ProjectTitle, SupervisorID), SUPERVISOR(SupervisorID, SupervisorName, SupervisorOffice), STUDENT\_PROJECT(StudentID, ProjectID).

III. Hospital Visits → VISIT(VisitID, VisitDate, PatientID, DoctorID, Diagnosis), PATIENT(PatientID, PatientName), DOCTOR(DoctorID, DoctorName, DeptID), DEPT(DeptID, DeptName).

IV. Movie Streaming → USER(UserID, UserName), MOVIE(MovieID, MovieTitle, GenreID), GENRE(GenreID, GenreName), RATING(UserID,MovieID,Rating).

V. Supply Chain → Decompose: WAREHOUSE(WarehouseID, WarehouseManagerID), MANAGER(WarehouseManagerID, ManagerName), INVENTORY(ProductID, WarehouseID, StockQty). Check BCNF: if WarehouseID→WarehouseManagerID then WAREHOUSE ok; INVENTORY's key is (ProductID,WarehouseID) and manager info moved out.