**Functional Dependencies, Canonical cover & Normalization**

* **User** (userName, firstName, lastName, email, contactNumber, password)

**Candidate keys**:

userName, email.

userName ⊆ User and userName → User

email ⊆ User. email → User

We defined userName as primary key since email can be changed by a user.

**Functional Dependencies**:

F = {userName → firstName, userName → lastName, userName → email, userName → contactNumber, userName → password}

**Canonical Cover**:

* Step 1: Making right hand side (RHS) a single attribute.

RHS is already a single attribute in all FDs.

* Step 2: Having Left Hand Side in simple form.

userName → firstName (No left redundancy)

userName → lastName (No left redundancy)

userName → email (No left redundancy)

userName → contactNumber (No left redundancy)

userName → password (No left redundancy)

* Step 3: Remove redundant FDs.
* For: userName → firstName

Let G = F – {userName → firstName}

G = {userName → lastName, userName → email, userName → contactNumber, userName → password}

userName+G = {lastName, email, contactNumber, password}

Since firstName ∉ userName+G so userName → firstName is not redundant.

* For: userName → lastName

Let G = F – {userName → lastName}

G = {userName → firstName, userName → email, userName → contactNumber, userName → password}

userName+G = {firstName, email, contactNumber, password}

Since lastName ∉ userName+G so userName → lastName is not redundant.

* For: userName → email

Let G = F – {userName → email}

G = {userName → firstName, userName → lastName, userName → contactNumber, userName → password}

userName+G = {firstName, lastName, contactNumber, password}

Since email ∉ userName+G so userName → email is not redundant.

* For: userName → contactNumber

Let G = F – {userName → contactNumber}

G = {userName → firstName, userName → lastName, userName → email, userName → password}

userName+G = {firstName, lastName, email, password}

Since contactNumber ∉ userName+G so userName → contactNumber is not redundant.

* For: userName → password

Let G = F – {userName → password}

G = {userName → firstName, userName → lastName, userName → email, userName → contactNumber}

userName+G = {firstName, lastName, email, contactNumber}

Since password ∉ userName+G so userName → password is not redundant.

Since there is no redundant Functional Dependencies in F so F is a Canonical Cover of itself.

**Normalization**

The primary key of User is “userName”.

F = {userName → firstName, userName → lastName, userName → email, userName → contactNumber, userName → password}

Since RHS of all FDs in F is the primary key so User is in BCNF.

**Summary**

Primary Key: userName

Functional Dependencies: {userName → firstName, userName → lastName, userName → email, userName → contactNumber, userName → password}

Canonical Cover: {userName → firstName, userName → lastName, userName → email, userName → contactNumber, userName → password}

Normalization: BCNF

* **Employer** (userName, employerName, accStatus, balance)

**Candidate key**:

userName

userName ⊆ Employer and userName → Employer

**Functional Dependencies**:

F = {userName → employerName, userName → accStatus, userName → balance}

**Canonical Cover:**

* Step 1: Making RHS single attribute.

All the FDs have single attribute in the RHS.

* Step 2: Having LHS in simple form.

{userName → employerName} (No left redundancy)

{userName → accStatus} (No left redundancy)

{userName → balance} (No left redundancy)

* Step 3: Removing redundant FDs
* For: userName → employerName

Let G = F – {userName → employerName}

G = {userName → accStatus, userName → balance}

userName+G = {accStatus, balance}

Since employerName ∉ userName+G so userName → employer is not redundant.

* For: userName → accStatus

Let G = F – {userName → accStatus}

G = {userName → employerName, userName → balance}

userName+G = {employerName, balance}

Since accStatus ∉ userName+G so userName → accStatus is not redundant.

* For: userName → balance

Let G = F – {userName → balance}

G = {userName → employerName, userName → accStatus}

userName+G = {employerName, accStatus}

Since balance ∉ userName+G so userName → balance is not redundant.

There are no redundant functional dependencies in F. So F is a canonical cover of itself.

**Normalization**

Primary Key: userName

F = {userName → employerName, userName → accStatus, userName → balance}

Since RHS of all FD’s in F is the primary key so Employer is in BCNF.

**Summary**

Primary Key: userName

Functional Dependencies: {userName → employerName, userName → accStatus, userName → balance}

Canonical Cover: {userName → employerName, userName → accStatus, userName → balance}

Normalization: BCNF

* **Applicant** (userName, accStatus)

**Candidate key**:

userName

userName ⊆ Applicant and userName → Applicant

**Functional Dependencies**:

F = {userName → accStatus}

**Canonical Cover:**

* Step 1: Making RHS single attribute.

All the FDs have single attribute in the RHS.

* Step 2: Having LHS in simple form.

{userName → accStatus} (No left redundancy)

* Step 3: Removing redundant FDs
* For: userName → accStatus

Let G = F – {userName → accStatus}

G = {}

Since accStatus ∉ userName+G so userName → accStatus is not redundant.

There are no redundant functional dependencies in F. So, F is a canonical cover of itself.

**Normalization**

Primary Key: userName

F = {userName → accStatus}

Since RHS of all FD’s in F is the primary key so Applicant is in BCNF.

**Summary**

Primary Key: userName

Functional Dependencies: {userName → accStatus}

Canonical Cover: {userName → accStatus}

Normalization: BCNF

* **Admin** (userName)

**Candidate Key:** userName.

userName ⊆ Admin and userName → Admin

**Functional Dependencies:** F = {}

**Canonical Cover:** F is a canonical cover of itself.

**Normalization:** BCNF

**Summary**

Primary Key: userName

Functional Dependencies: {}

Canonical Cover: {}

Normalization: BCNF

* **EmployerCategory** (category, monthlyCharge)

**Candidate Key:** category.

category ⊆ EmployerCategory and category → EmployerCategory

**Functional Dependencies**:

F = {category → monthlyCharge}

**Canonical Cover:**

There are no redundant functional dependencies in F. So F is a canonical cover of itself.

**Normalization:**

Primary Key: category

F = {category → monthlyCharge}

Since RHS of all FD’s in F is the primary key so EmployerCategory is in BCNF.

**Summary**

Primary Key: category

Functional Dependencies: {category → monthlyCharge}

Canonical Cover: {category → monthlyCharge}

Normalization: BCNF

* **ApplicantCategory** (category, monthlyCharge)

**Candidate Key:** category.

category ⊆ ApplicantCategory and category → ApplicantCategory

**Functional Dependencies**:

F = {category → monthlyCharge}

**Canonical Cover:**

There are no redundant functional dependencies in F. So F is a canonical cover of itself.

**Normalization:**

Primary Key: category

F = {category → monthlyCharge}

Since RHS of all FD’s in F is the primary key so ApplicantCategory is in BCNF.

**Summary**

Primary Key: category

Functional Dependencies: {category → monthlyCharge}

Canonical Cover: {category → monthlyCharge}

Normalization: BCNF

* **Job** (jobID, employerUserName, title, datePosted, description, category, jobStatus, empNeeded)

**Candidate key**:

jobID

jobID ⊆ Job and jobID → Job

**Functional Dependencies**:

F = {jobID → employerUserName, jobID → title, jobID → datePosted, jobID → description, jobID → category, jobID → jobStatus, jobID → empNeeded}

**Canonical Cover:**

* Step 1: Making RHS single attribute.

All the FDs have single attribute in the RHS.

* Step 2: Having LHS in simple form.

{jobID → employerUserName} (No left redundancy)

{jobID → title} (No left redundancy)

jobID → datePosted} (No left redundancy)

{jobID → description} (No left redundancy)

{jobID → category} (No left redundancy)

{jobID → jobStatus} (No left redundancy)

{jobID → empNeeded} (No left redundancy)

* Step 3: Removing redundant FDs
* For: jobID → employerUserName

Let G = F – {jobID → employerUserName}

G = {jobID → title, jobID → datePosted, jobID → description, jobID → category, jobID → jobStatus, jobID → empNeeded}

jobID+G = {title, datePosted, description, category, jobStatus, empNeeded}

Since employerUserName∉ jobID+G so jobID → employerUserNameis not redundant.

* For: jobID → title

Let G = F – {jobID → title}

G = {jobID → employerUserName, jobID → datePosted, jobID → description, jobID → category, jobID → jobStatus, jobID → empNeeded}

jobID+G = {employerUserName, datePosted, description, category, jobStatus, empNeeded}

Since title∉ jobID+G so jobID → titleis not redundant.

* For: jobID → datePosted

Let G = F – {jobID → datePosted}

G = {jobID → employerUserName, jobID → title, jobID → description, jobID → category, jobID → jobStatus, jobID → empNeeded}

jobID+G = {employerUserName, title, description, category, jobStatus, empNeeded}

Since datePosted ∉ jobID+G so jobID → datePosted is not redundant.

* For: jobID → description

Let G = F – {jobID → description}

G = {jobID → employerUserName, jobID → title, jobID → datePosted, jobID → category, jobID → jobStatus, jobID → empNeeded}

jobID+G = {employerUserName, title, datePosted, category, jobStatus, empNeeded}

Since description ∉ jobID+G so jobID → description is not redundant.

* For: jobID → category

Let G = F – {jobID → category}

G = {jobID → employerUserName, jobID → title, jobID → datePosted, jobID → description, jobID → jobStatus, jobID → empNeeded}

jobID+G = {employerUserName, title, datePosted, description, jobStatus, empNeeded}

Since category ∉ jobID+G so jobID → category is not redundant.

* For: jobID → jobStatus

Let G = F – {jobID → jobStatus}

G = {jobID → employerUserName, jobID → title, jobID → datePosted, jobID → description, jobID → category, jobID → empNeeded}

jobID+G = {employerUserName, title, datePosted, description, category, empNeeded}

Since jobStatus ∉ jobID+G so jobID → jobStatus is not redundant.

* For: jobID → empNeeded

Let G = F – {jobID → empNeeded}

G = {jobID → employerUserName, jobID → title, jobID → datePosted, jobID → description, jobID → category, jobID → jobStatus}

jobID+G = {employerUserName, title, datePosted, description, category, jobStatus}

Since empNeeded ∉ jobID+G so jobID → empNeeded is not redundant.

There are no redundant functional dependencies in F. So F is a canonical cover of itself.

**Normalization**

Primary Key: jobID

F = {jobID → employerUserName, jobID → title, jobID → datePosted, jobID → description, jobID → category, jobID → jobStatus, jobID → empNeeded}

Since RHS of all FD’s in F is the primary key so Job is in BCNF.

**Summary**

Primary Key: jobID

Functional Dependencies: {jobID → employerUserName, jobID → title, jobID → datePosted, jobID → description, jobID → category, jobID → jobStatus, jobID → empNeeded}

Canonical Cover: {jobID → employerUserName, jobID → title, jobID → datePosted, jobID → description, jobID → category, jobID → jobStatus, jobID → empNeeded}

Normalization: BCNF

* **DefaultMOP** (cardNumber, auto\_manual)

**Candidate key**:

cardNumber

cardNumber ⊆ DefaultMOPand cardNumber → DefaultMOP

**Functional Dependencies**:

F = {cardNumber → auto\_manual}

**Canonical Cover:**

There are no redundant functional dependencies in F. So, F is a canonical cover of itself.

**Normalization**

Primary Key: cardNumber

F = {cardNumber → auto\_manual}

Since RHS of all FD’s in F is the primary key so DefaultMOP is in BCNF.

**Summary**

Primary Key: cardNumber

Functional Dependencies: {cardNumber → auto\_manual}

Canonical Cover: {cardNumber → auto\_manual}

Normalization: BCNF

* **CreditCardInfo** (CCNumber, expireDate, userUserName, CCBNumber)

**Candidate key**:

{CCNumber, expireDate}

{CCNumber, expireDate} ⊆ CreditCardInfo and {CCNumber, expireDate} → CreditCardInfo

**Functional Dependencies**:

F = {CCNumber, expireDate → userUserName, CCNumber, expireDate → CCBNumber}

**Canonical Cover:**

* Step 1: Making RHS single attribute.

All the FDs have single attribute in the RHS.

* Step 2: Having LHS in simple form.

CCNumber+ = CCNumber.

expireDate+ = expireDate.

CCNumber, expireDate → userUserName (No left redundancy)

CCNumber, expireDate → CCBNumber (No left redundancy)

* Step 3: Removing redundant FDs
* For: CCNumber, expireDate → userUserName

Let G = F – {CCNumber, expireDate → userUserName}

G = {CCNumber, expireDate → CCBNumber}

{CCNumber, expireDate}+G = {CCBNumber}

Since userUserName ∉ {CCNumber, expireDate}+G so CCNumber, expireDate → userUserName is not redundant.

* For: CCNumber, expireDate → CCBNumber

Let G = F – {CCNumber, expireDate → CCBNumber}

G = {CCNumber, expireDate → userUserName}

{CCNumber, expireDate}+G = {userUserName}

Since CCBNumber ∉ {CCNumber, expireDate}+G so CCNumber, expireDate → CCBNumber is not redundant.

There are no redundant functional dependencies in F. So, F is a canonical cover of itself.

**Normalization**

Primary Key: CCNumber, expireDate

F = {CCNumber, expireDate → userUserName, CCNumber, expireDate → CCBNumber}

Since RHS of all FD’s in F is the primary key so CreditCardInfo is in BCNF.

**Summary**

Primary Key: CCNumber, expireDate

Functional Dependencies: {CCNumber, expireDate → userUserName, CCNumber, expireDate → CCBNumber}

Canonical Cover: {CCNumber, expireDate → userUserName, CCNumber, expireDate → CCBNumber}

Normalization: BCNF

* **PADInfo** (accountNumber, userUserName, instituteNumber)

**Candidate key**:

accountNumber

accountNumber ⊆ PADInfo and accountNumber → PADInfo

**Functional Dependencies**:

F = {accountNumber → userUserName, accountNumber → instituteNumber}

**Canonical Cover:**

* Step 1: Making RHS single attribute.

All the FDs have single attribute in the RHS.

* Step 2: Having LHS in simple form.

{accountNumber → userUserName} (No left redundancy)

{accountNumber → instituteNumber} (No left redundancy)

* Step 3: Removing redundant FDs
* For: accountNumber → userUserName

Let G = F – {accountNumber → userUserName}

G = {accountNumber → instituteNumber}

accountNumber+G = {instituteNumber}

Since userUserName ∉ accountNumber+G so accountNumber → userUserName is not redundant.

* For: accountNumber → instituteNumber

Let G = F – {accountNumber → instituteNumber}

G = {accountNumber → userUserName}

accountNumber+G = {userUserName}

Since instituteNumber ∉ accountNumber+G so accountNumber → instituteNumber is not redundant.

There are no redundant FD. So, F is a canonical cover of itself.

**Normalization**

Primary Key: accountNumber

F = {accountNumber → userUserName, accountNumber → instituteNumber}

Since RHS of all FD’s in F is the primary key so PADInfo is in BCNF.

**Summary**

Primary Key: accountNumber

Functional Dependencies: {accountNumber → userUserName, accountNumber → instituteNumber}

Canonical Cover: {accountNumber → userUserName, accountNumber → instituteNumber}

Normalization: BCNF

* **PADBranch** (PADAccountNumber, branchNumber)

**Candidate key**:

PADAccountNumber

PADAccountNumber ⊆ PADBranch and PADAccountNumber → PADBranch

**Functional Dependencies**:

F = {PADAccountNumber → branchNumber}

**Canonical Cover:**

There are no redundant FD. So, F is a canonical cover of itself.

**Normalization**

Primary Key: PADAccountNumber

F = {PADAccountNumber → branchNumber}

Since RHS of all FD’s in F is the primary key so PADBranch is in BCNF.

**Summary**

Primary Key: PADAccountNumber

Functional Dependencies: {PADAccountNumber → branchNumber}

Canonical Cover: {PADAccountNumber → branchNumber}

Normalization: BCNF

* **ApplicantBalane** (applicantUserName, balance)

**Candidate key**:

applicantUserName

applicantUserName ⊆ ApplicantBalane and applicantUserName → ApplicantBalane

**Functional Dependencies**:

F = {applicantUserName → balance}

**Canonical Cover:**

There are no redundant FD. So, F is a canonical cover of itself.

**Normalization**

Primary Key: PADAccountNumber

F = {PADAccountNumber → branchNumber}

Since RHS of all FD’s in F is the primary key so PADBranch is in BCNF.

**Summary**

Primary Key: PADAccountNumber

Functional Dependencies: {PADAccountNumber → branchNumber}

Canonical Cover: {PADAccountNumber → branchNumber}

Normalization: BCNF

* **Payment** (cardNumber, paymentDate, amount)

**Candidate key**:

{cardNumber, paymentDate}

{cardNumber, paymentDate} ⊆ Payment and {cardNumber, paymentDate} → Payment

**Functional Dependencies**:

F = {cardNumber, paymentDate → amount}

**Canonical Cover:**

* Step 1: Making RHS single attribute.

All the FDs have single attribute in the RHS.

* Step 2: Having LHS in simple form.

cardNumber+ = cardNumber.

paymentDate+ = paymentDate.

cardNumber, paymentDate → amount (No left redundancy)

* Step 3: Removing redundant FDs

There are no redundant functional dependencies in F. So, F is a canonical cover of itself.

**Normalization**

Primary Key: cardNumber, paymentDate

F = {cardNumber, paymentDate → amount}

Since RHS of all FD’s in F is the primary key so Payment is in BCNF.

**Summary**

Primary Key: cardNumber, paymentDate

Functional Dependencies: {cardNumber, paymentDate → amount}

Canonical Cover: {cardNumber, paymentDate → amount}

Normalization: BCNF

* **SelectEmployerCategory** (employerUserName, startDate, charge, employerCategory)

**Candidate key**:

{employerUserName, startDate}

{employerUserName, startDate} ⊆ SelectEmployerCategory and {employerUserName, startDate} → SelectEmployerCategory

**Functional Dependencies**:

F = {employerUserName, startDate → charge, employerUserName, startDate → employerCategory}

**Canonical Cover:**

* Step 1: Making RHS single attribute.

All the FDs have single attribute in the RHS.

* Step 2: Having LHS in simple form.

employerUserName+ = employerUserName.

startDate + = startDate.

{employerUserName, startDate} → charge (No left redundancy)

{employerUserName, startDate} → employerCategory (No left redundancy)

* Step 3: Removing redundant FDs
* For: {employerUserName, startDate} → charge

Let G = F – {employerUserName, startDate → charge}

G = {employerUserName, startDate → employerCategory}

{employerUserName, startDate}+G = {employerCategory}

Since charge ∉ {employerUserName, startDate}+G so employerUserName, startDate → charge is not redundant.

* For: {employerUserName, startDate} → employerCategory

Let G = F – {employerUserName, startDate → employerCategory}

G = {employerUserName, startDate → charge}

{employerUserName, startDate}+G = {charge}

Since employerCategory ∉ {employerUserName, startDate}+G so employerUserName, startDate → employerCategory is not redundant.

There are no redundant functional dependencies in F. So, F is a canonical cover of itself.

**Normalization**

Primary Key: employerUserName, startDate

F = {employerUserName, startDate → charge, employerUserName, startDate → employerCategory}

Since RHS of all FD’s in F is the primary key so SelectEmployerCategory is in BCNF.

**Summary**

Primary Key: employerUserName, startDate

Functional Dependencies: {employerUserName, startDate → charge, employerUserName, startDate → employerCategory}

Canonical Cover: {employerUserName, startDate → charge, employerUserName, startDate → employerCategory}

Normalization: BCNF

* **SelectApplicantCategory** (applicantUserName, startDate, charge, applicantCategory)

**Candidate key**:

{applicantUserName, startDate}

{applicantUserName, startDate} ⊆ SelectApplicantCategory and {applicantUserName, startDate} → SelectApplicantCategory

**Functional Dependencies**:

F = {applicantUserName, startDate → charge, applicantUserName, startDate → applicantCategory}

**Canonical Cover:**

* Step 1: Making RHS single attribute.

All the FDs have single attribute in the RHS.

* Step 2: Having LHS in simple form.

employerUserName+ = employerUserName.

startDate + = startDate.

{applicantUserName, startDate} → charge (No left redundancy)

{applicantUserName, startDate} → applicantCategory (No left redundancy)

* Step 3: Removing redundant FDs
* For: {applicantUserName, startDate} → charge

Let G = F – {applicantUserName, startDate → charge}

G = {applicantUserName, startDate → applicantCategory}

{applicantUserName, startDate}+G = {applicantCategory}

Since charge ∉ {applicantUserName, startDate}+G so applicantUserName, startDate → charge is not redundant.

* For: {applicantUserName, startDate} → applicantCategory

Let G = F – {applicantUserName, startDate → applicantCategory}

G = {applicantUserName, startDate → charge}

{applicantUserName, startDate}+G = {charge}

Since applicantCategory ∉ {applicantUserName, startDate}+G so applicantUserName, startDate → applicantCategory is not redundant.

There are no redundant functional dependencies in F. So, F is a canonical cover of itself.

**Normalization**

Primary Key: applicantUserName, startDate

F = {applicantUserName, startDate → charge, applicantUserName, startDate → applicantCategory}

Since RHS of all FD’s in F is the primary key so SelectApplicantCategory is in BCNF.

**Summary**

Primary Key: applicantUserName, startDate

Functional Dependencies: {applicantUserName, startDate → charge, applicantUserName, startDate → applicantCategory}

Canonical Cover: {applicantUserName, startDate → charge, applicantUserName, startDate → applicantCategory}

Normalization: BCNF

* **Apply** (applicantUserName, jobID, applicationStatus, applicationDate)

**Candidate key**:

{applicantUserName, jobID}

{applicantUserName, jobID} ⊆ Apply and {applicantUserName, jobID} → Apply

**Functional Dependencies**:

F = {applicantUserName, jobID → applicationStatus, applicantUserName, jobID → applicationDate}

**Canonical Cover:**

* Step 1: Making RHS single attribute.

All the FDs have single attribute in the RHS.

* Step 2: Having LHS in simple form.

applicantUserName + = applicantUserName.

jobID+ = jobID.

{applicantUserName, jobID} → applicationStatus (No left redundancy)

{applicantUserName, jobID} → applicationDate (No left redundancy)

* Step 3: Removing redundant FDs
* For: {applicantUserName, jobID} → applicationStatus

Let G = F – {applicantUserName, jobID → applicationStatus}

G = {applicantUserName, jobID → applicationDate}

{applicantUserName, jobID}+G = {applicationDate}

Since applicationStatus ∉ {applicantUserName, jobID}+G so applicantUserName, jobID → applicationStatus is not redundant.

* For: {applicantUserName, jobID} → applicationDate

Let G = F – {applicantUserName, jobID → applicationDate}

G = {applicantUserName, jobID → applicationStatus}

{applicantUserName, jobID}+G = {applicationStatus}

Since applicationDate ∉ {applicantUserName, jobID}+G so applicantUserName, jobID → applicationDate is not redundant.

There are no redundant functional dependencies in F. So, F is a canonical cover of itself.

**Normalization**

Primary Key: applicantUserName, jobID

F = {applicantUserName, jobID → applicationStatus, applicantUserName, jobID → applicationDate}

Since RHS of all FD’s in F is the primary key so Apply is in BCNF.

**Summary**

Primary Key: applicantUserName, jobID

Functional Dependencies: {applicantUserName, jobID → applicationStatus, applicantUserName, jobID → applicationDate}

Canonical Cover: {applicantUserName, jobID → applicationStatus, applicantUserName, jobID → applicationDate}

Normalization: BCNF