Functional Dependency with Examples-

- A **Functional dependency** is a relationship between attributes. Suppose that given the value of one attribute, we can obtain the value of another attribute.
- For example, if we know the value of customer account number, we can obtain customer address, balance etc. By this, we say that customer address and balance is functionally dependent on customer account number.
- In **general terms**, attribute Y(customer address and balance) is functionally dependent on the attribute X(customer account number), if the value of X determines the value of Y.
- Functional dependency is **represented** by an **arrow sign (→)** that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side. Attributes on the left side are called **Determinants.**

Let us consider a relation

Item Price(P)	Number of Items(N)	Total Price=P*N
100	10	1000
200	10	2000
300	10	3000
400	10	4000

Total Price = Item Price * Number of Items In this case, Total Price is functionally dependent on Item Price and Number Of Items. Functional Dependencies are written using the following notation:
(Item price, Number of Items) → Total Price

Consider another relation R(S#,STATUS,CITY,P#,QTY)

S#	STATUS	CITY	P#	QTY
S001	10	Ujjain	P001	105
S001	10	Ujjain	P002	130
S001	10	Ujjain	P003	125
S001	10	Ujjain	P004	200
S002	30	Surat	P001	115
S002	30	Surat	P002	120
S003	40	Rohtak	P001	300
S004	10	Ujjain	P004	400
S004	10	Ujjain	P005	400

R relation hold following Functional Dependencies :
S# → CITY
S# → STATUS
CITY → STATUS
(S#,P#) → QTY
Figure 1 shows the functional dependency diagram for the relation R.

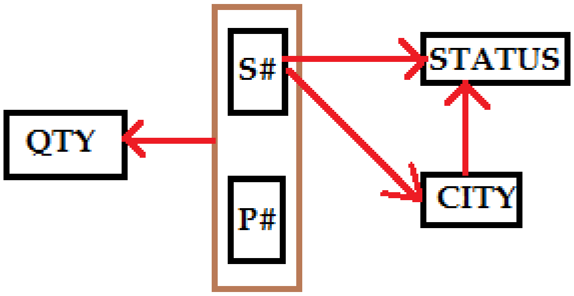

FIGURE 1

Properties of Functional Dependency -

- The Functional Dependency will deal with one to one relationship among the attributes, some times it may be one to many or many to one relationships
- Functional Dependency must be defined on schema, not on instance of the schema. For example,

  ab → cd
  a → b
  ab → c etc.

- A Functional Dependency must be a <u>non trivial/semi non trivial or completely non trivial</u>. It does not be trivial. A trivial Functional Dependency is the one where RHS is a subset of LHS. Example of Trivial Functional Dependency - abc → bc
- A Functional Dependency is non trivial if atleast one of the RHS attributes are not part of the LHS attributes. A Functional Dependency is completely non trivial if none of the RHS attributes are part of the LHS attributes. Example of Non Trivial - abc → cde Example of Complete Non Trivial - abc → def.

Types of Functional Dependency with Example -

Various Types of Functional Dependencies are -

- **Single Valued Functional Dependency**
- **Fully Functional Dependency**
- **Partial Functional Dependency**
- **Transitive Functional Dependency**
- **Trivial Functional Dependency**
- **Non Trivial Functional Dependency**
  - **Complete Non Trivial Functional Dependency**
  - **Semi Non Trivial Functional Dependency**

Single Valued Functional Dependency -

Database is a collection of related information in which one information depends on another information. The information is either single-valued or multi-valued. For example, the name of the person or his.her date of birth are single valued facts. But the qualification of a person is a multivalued facts. A simple example of single value functional dependency is when A is the primary key of an entity (eg. SID) and B is some single valued attribute of the entity (eg. Sname). Then, A → B must always hold.

CID	SID	Sname
C1	S1	A
C1	S2	A
C2	S1	A
C3	S1	A

SID → Sname			**Sname → SID**	**X**
S1	A		A	S1
S1	A		A	S2
S1	A			

For every SID, there should be unique name (X → Y) **Definition :** Let R be the relational schema and X, Y be the set of attributes over R. t1, t2 be the any tuples of R. X → Y exists in relation R only if t1.X = t2.X then t1.Y = t2.Y If condition fails - then dependency is not there.

Fully Functional Dependency -

A functional dependency $P \rightarrow Q$ is full functional dependency if removal of any attribute A from P means that the dependency does not hold any more. or In a relation R, an attribute Q is said to be fully functional dependent on attribute P, if it is functionally dependent on P and not functionally dependent on any proper subset of P. The dependency $P \rightarrow Q$ is left reduced, their being no extraneous attributes in the left hand side of the dependency. If $AD \rightarrow C$, is fully functional dependency, then we cannot remove A or D. i.e. C is fully functional dependent on AD. If we are able to remove A or D, then it is not full functional dependency. Another Example, Consider the following Company Relational Schema,

{SSN, PNUMBER} → HOURS is a full FD since neither SSN → HOURS nor PNUMBER → HOURS hold

{SSN, PNUMBER} → ENAME is not a full FD (it is called a partial dependency) since SSN → ENAME also holds

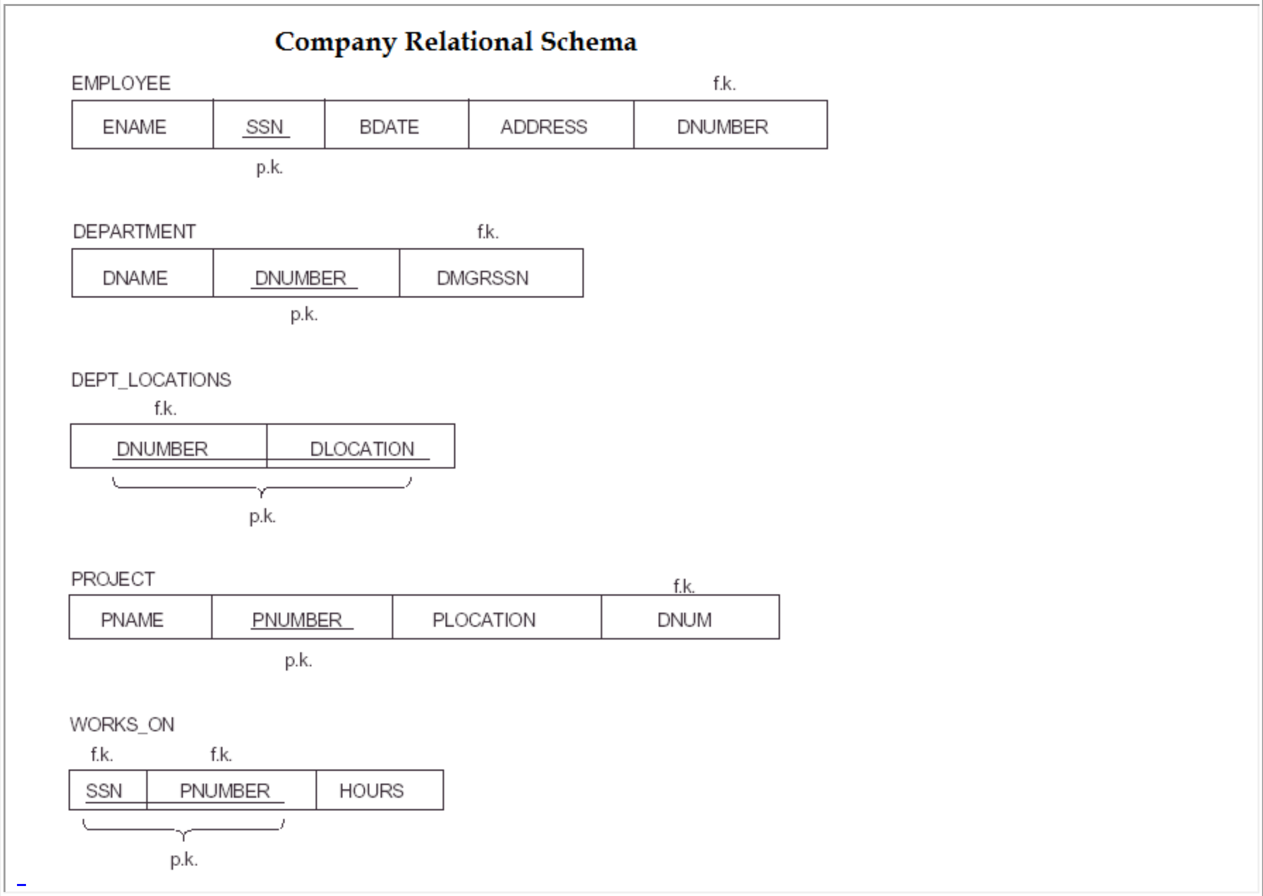

FIG 1

Partial Functional Dependency -

A Functional Dependency in which one or more non key attributes are functionally depending on a part of the primary key is called partial functional dependency. or where the determinant consists of key attributes, but not the entire primary key, and the determined consist of non-key attributes.
For example, Consider a Relation R(A,B,C,D,E) having
FD : AB → CDE where PK is AB.

Then, { A → C; A → D; A → E; B → C; B → D; B → E }
all are Partial Dependencies.

Transitive Dependency -

Given a relation R(A,B,C) then dependency like A-->B, B-->C is a transitive dependency, since A-->C is implied .

In the above Fig 1,
SSN --> DMGRSSN is a transitive FD
{since SSN --> DNUMBER and DNUMBER --> DMGRSSN hold}

SSN --> ENAME is non-transitive FD since there is no set of attributes X
where SSN --> X and X --> ENAME.

Trivial and Non Trivial Functional Dependency examples -

Trivial Functional Dependency -
Some functional dependencies are said to be trivial because they are satisfied by all relations. Functional dependency of the form A-->B is trivial if $B \subseteq A$. or A trivial Functional Dependency is the one where RHS is a subset of LHS.
Example, A-->A is satisfied by all relations involving attribute A.
SSN-->SSN
PNUMBER-->PNUMBER
SSN PNUMBER -->PNUMBER
SSN PNUMBER --> SSN PNUMBER

Non Trivial Functional Dependency -
Non Trivial Functional Dependency can be categorized into -

- Complete Non Trivial Functional Dependency
- Semi Non Trivial Functional Dependency

Complete Non Trivial Functional Dependency - A Functional Dependency is completely non trivial if none of the RHS attributes are part of the LHS attributes.
Example, SSN --> Ename,
PNUMBER --> PNAME
PNUMBER--> BDATE X
Semi Non Trivial Functional Dependencies - A Functional Dependency is semi non trivial if atleast one of the RHS attributes are not part of the LHS attributes. {TRIVIAL + NONTRIVIAL}
Question 1 :

A	B	C
1	1	1
1	2	1
2	1	2
2	2	3

Identify Non Trivial Functional Dependency ?
Solution :

S.NO	Dependencies	**Non Trivial FD ?**
1	A→B	×
2	A→C	×
3	A→BC	×
4	B→A	×
5	B→C	×
6	B→AC	×
7	C→A	√
8	C→B	×
9	C→AB	×
10	AB→C	√
11	BC→A	√
12	AC→B	×

A→B is not a non trivial FD because, for 2, it has two outputs. i.e 2→2 and 2→3.
for AB→C, 11→1, 12→1, 21→2, 22→3, so Non trivial.
Question 2:
R(A B C D)
AB {Candidate Key}
A→C
B→D .
Where is the redundancy exist ?

Solution: (A C) and (B D) is suffering from redundancy.

Question 3:

Consider a relation with schema R(A,B,C,D) and FDs {AB -> C, C -> D, D -> A}.

a. What are some of the nontrivial FDs that can be inferred from the given FDs?

Some examples:

C -> ACD
D -> AD
AB -> ABCD
AC -> ACD
BC -> ABCD
BD -> ABCD
CD -> ACD
ABC -> ABCD
ABD -> ABCD
BCD -> ABCD

Inference Rules For Functional Dependencies -

Let S be the set of functional dependencies that are specified on relation schema R. Numerous other dependencies can be inferred or deduced from the functional dependencies in S. **Example :** Let S = {A → B, B → C} We can infer the following functional dependency from S: A → C

Armstrong's Inference Rules -

Let A, B and C and D be arbitrary subsets of the set of attributes of the giver relation R, and let AB be the union of A and B. Then,⇒→

- **Reflexivity :** If B is subset of A, then A → B
- **Augmentation :** If A → B, then AC → BC
- **Transitivity :** If A → B and B → C, then A → C.
- **Projectivity or Decomposition Rule :** If A → BC, Then A → B and A → C

  Proof :
  Step 1 : A → BC (GIVEN)
  Step 2 : BC → B (Using Rule 1, since B ⊆ BC)
  Step 3 : A → B (Using Rule 3, on step 1 and step 2)

- **Union or Additive Rule :** If A→B, and A→C Then A→BC.

  Proof :
  Step 1 : A → B (GIVEN)
  Step 2 : A → C (given)
  Step 3 : A → AB (using Rule 2 on step 1, since AA=A)
  Step 4 : AB → BC (using rule 2 on step 2)
  Step 5 : A → BC (using rule 3 on step 3 and step 4)

- **Pseudo Transitive Rule :** If A → B, DB → C, then DA → C

  Proof :
  Step 1 : A → B (Given)
  Step 2 : DB → C (Given)
  Step 3 : DA → DB (Rule 2 on step 1)
  Step 4 : DA → C (Rule 3 on step 3 and step 2)

- **These are not commutative as well as associative. i.e. if X → Y then Y → X x (not possible)**
- **Composition Rule :** If A → B, and C → D, then AC → BD.
- **Self Determination Rule :** A → A is a self determination rule.

Question 1:
Prove or disprove the following inference rules for functional dependencies.
Note: Read "⇒" as implies
a. $\{X \rightarrow Y, Z \rightarrow W\} \Rightarrow XZ \rightarrow YW$??
b. $\{X \rightarrow Y, XY \rightarrow Z\} \Rightarrow X \rightarrow Z$
c. $\{XY \rightarrow Z, Y \rightarrow W\} \Rightarrow XW \rightarrow Z$
Solution :
Method : Use Armstrong's Axioms or Attribute closure to prove or disprove.
a. $\{X \rightarrow Y, Z \rightarrow W\} \Rightarrow XZ \rightarrow YW$??
$XZ \rightarrow XZ$
$XZ \rightarrow XW$ (Z -> W)
$XZ \rightarrow W$ (decomposition rule)

$XZ \rightarrow XZ$
$XZ \rightarrow YZ$ (X -> Y)
$XZ \rightarrow Y$ (decomposition rule)

$\Rightarrow XZ \rightarrow YW$ (union rule)
Hence True.
b. $\{X \rightarrow Y, XY \rightarrow Z\} \Rightarrow X \rightarrow Z$??

$XY \rightarrow Z$
$XX \rightarrow Z$ (pseudotransitivity rule as $X \rightarrow Y$)
$\Rightarrow X \rightarrow Z$
Hence True.

c. $\{XY \rightarrow Z, Y \rightarrow W\} \Rightarrow XW \rightarrow Z$??
$W \rightarrow W$
$X \rightarrow X$
$Y \rightarrow YW$
$Z \rightarrow Z$
$WX \rightarrow WX$
$WY \rightarrow WY$
$WZ \rightarrow WZ$
$XY \rightarrow WXYZ$
$XZ \rightarrow XZ$
$YZ \rightarrow WYZ$

Therefore $WX \rightarrow Z$ is not true
You can also find the attribute closure for WX and show that closure set does not contain Z.

Question 2:
Consider a relational scheme R with attributes A,B,C,D,F and the FDs
$A \rightarrow BC$
$B \rightarrow E$
$CD \rightarrow EF$
Prove that functional dependency $AD \rightarrow F$ holds in R.
Step 1 : $A \rightarrow BC$ (Given)
Step 2 : $A \rightarrow C$ (Decomposition Rule applied on step 1)
Step 3 : $AD \rightarrow CD$ (Augmentation Rule applied on step 2)
Step 4 : $CD \rightarrow EF$ (Given)
Step 5 : $AD \rightarrow EF$ (transivity Rule applied on step 3 and 4)
Step 6 : $AD \rightarrow F$ (Decomposition Rule applied on step 5)

Closure of Functional Dependency Example and Applications of Closure –

Closure of a set (X^+) is the set of attributes functionally determined by X. Let S be the set of functional dependencies on relation R. Let X is set of attributes that appear on left hand side of some FD in S and we want to determine the set of all attributes that are dependent on X. Thus for each such set of attribute X, we determine the set X^+ of attributes that are functionally determined by X based on S, X^+ is called closure of X under S.

Algorithm to find the X^+ or Algorithm to find Closure of Functional Dependency is :

$X^+ = X$;
repeat
oldX^+ := X^+
for each FD $Y \rightarrow Z$ in S do
if Y is subset of X^+ then
$X^+ := X^+ \cup Z$;
until (X^+ =old X^+) /* If X^+ did not change then leave loop*/

Example:

Suppose we are given relation R with attributes A,B,C,D and FDs
$A \rightarrow BC$
$B \rightarrow CD$
To find the Closure of all the attributes :
Let us find closure of A firstly.

$$A^+ = A$$
$$\downarrow$$
$$A(BC) \quad \{As\ A \to BC\}$$
$$\downarrow$$
$$AB(CD)C \quad \{As\ B \to CD\}$$
$$\Rightarrow ABCDC = ABCD$$
$$So,\ A^+ = ABCD$$

(B^+) = {BCD}
(C^+) = {C}
(D^+) = {D}
(AB^+) = {ABCD}
(AC^+) = {ABCD}
(AD^+) = {ABCD}
(BC^+) = {BCD}
(BD^+) = {BCD}
(CD^+) = {CD}
(ABC^+) = {ABCD}
(ABD^+) = {ABCD}
(ACD^+) = {ABCD}
(BCD^+) = {BCD}
($ABCD^+$) = {ABCDEF}

Question : Compute the closure of following set F of functional dependencies for relation schema R = {A, B, C, D, E}.

A → BC
CD → E
B → D
E → A

Solution :

Attribute closure of all the attributes :

$$E^+ = E \downarrow E(A) \downarrow EA(\underline{B}C) \downarrow EAB(D)C$$

Since all the attributes are covered. So, stop the algorithm:

$\therefore\ E^+ = \{ABCDE\}$

similarly, for $B^+ = B \downarrow B(D)$

Since no new attributes are desired, So stopping the algorithm.

$\therefore\ B^+ = (BD)$

$(A)^+ = \{ABCDE\}$
$(C)^+ = \{C\}$
$(D)^+ = \{D\}$
$(AB)^+ = \{ABCDE\}$
$(AC)^+ = \{ABCDE\}$
$(AD)^+ = \{ABCDE\}$
$(AE)^+ = \{ABCDE\}$
$(BC)^+ = \{ABCDE\}$
$(BD)^+ = \{BD\}$
$(BE)^+ = \{ABCDE\}$
$(CD)^+ = \{ABCDE\}$
$(CE)^+ = \{ABCDE\}$
$(DE)^+ = \{ABCDE\}$
$(ABC)^+ = \{ABCDE\}$
$(ABD)^+ = \{ABCDE\}$
$(ABE)^+ = \{ABCDE\}$
$(ACD)^+ = \{ABCDE\}$
$(ACE)^+ = \{ABCDE\}$
$(ADE)^+ = \{ABCDE\}$
$(BCD)^+ = \{ABCDE\}$
$(BDE)^+ = \{ABCDE\}$
$(CDE)^+ = \{ABCDE\}$
$(ABCD)^+ = \{ABCDE\}$
$(ABCE)^+ = \{ABCDE\}$
$(ABDE)^+ = \{ABCDE\}$
$(ACDE)^+ = \{ABCDE\}$
$(BCDE)^+ = \{ABCDE\}$

Applications of Closure set of attributes –

It is used to identify the additional Functional Dependencies.
It is used to identify keys (Candidate Keys and Super Keys).
It is used to identify the Prime and Non Prime Attributes.
It is used to identify equivalence of Functional Dependency.
It is used to identify the irreducible set of Functional Dependencies or Canonical Cover of Functional Dependency.

Prime and Non Prime Attributes in DBMS with Example -

- **Prime Attributes -** Attribute set that belongs to any candidate key are called Prime Attributes. (union of all the candidate key attribute) {CK1 ∪ CK2 ∪ CK3 ∪ ......} If Prime attribute determined by other attribute set, then more than one candidate key is possible. For example, If A is Candidate Key, and X→A, then, X is also Candidate Key .
- **Non Prime Attribute -** Attribute set does not belongs to any candidate key are called Non Prime Attributes.

Some Questions Based on Prime Attributes and Non Prime Attributes -

Question 1 :
Given a relation R(ABCDEF) having FDs
{AB→C, C→D, D→E, F→B, E→F}
Identify the prime attributes and non prime attributes.
Solution :
$(AB)^+$: {ABCDEF} ⇒ Super Key
$(A)^+$: {A} ⇒ Not Super Key
$(B)^+$: {B} ⇒ Not Super Key
Prime Attributes : {A,B}
(AB) → Candidate Key
↓ (as F → B)
$(AF)^+$: {AFBCDE}
$(A)^+$: {A} ⇒ Not Super key
$(F)^+$: {FB} ⇒ Not Super Key
(AF) → Candidate Key
↓
$(AE)^+$: {AEFBCD}
$(A)^+$: {A} ⇒ Not Super key
$(E)^+$: {EFB} ⇒ Not Super key
(AE) → Candidate Key
↓
$(AD)^+$: {ADEFBC}
$(A)^+$: {A} ⇒ Not Super key
$(D)^+$: {DEFB} ⇒ Not Super key

$(AD) \rightarrow$ Candidate Key

$\downarrow$

$(AC)^+$: {ACDEFB}

$(A)^+$: {A} $\Rightarrow$ Not Super Key

$(C)^+$: {DCEFB} $\Rightarrow$ Not Super Key

$\Rightarrow$ Candidate Keys {AB, AF, AE, AD, AC}

$\Rightarrow$ Prime Attributes {A,B,C,D,E,F}

$\Rightarrow$ Non Prime Attributes {}

Question 2:

Given a relation R(ABCDEF) having FDs

$\{AB \rightarrow C, C \rightarrow DE, E \rightarrow F, C \rightarrow B\}$

Identify the prime attributes and non prime attributes.

Solution :

$(AB)^+$: {A B C D E F}

$(A)^+$: {A}

$(B)^+$: {B}

(AB) $\Rightarrow$ (AC), $(AC)^+$: {ABCDEF}

$(C)^+$: {DECBF}

$\Rightarrow$ Candidate Keys {AB, AC}

$\Rightarrow$ Prime Attributes {A,B,C}

$\Rightarrow$ Non Prime Attributes {D,E,F}

Question 3:

Given a relation R(ABCDEFGHIJ) having FDs

$\{AB \rightarrow C, A \rightarrow DE, B \rightarrow F, F \rightarrow GH, D \rightarrow IJ\}$

Identify the prime attributes and non prime attributes.

Solution :

$(AB)^+$: {ABCDEFGHIJ}

$(A)^+$: {DEIJA}

$(B)^+$: {FGHB}

$\Rightarrow$ Candidate Keys {AB}

$\Rightarrow$ Prime Attributes {A,B}

$\Rightarrow$ Non Prime Attributes {C,D,E,F,G,H,I,J}

Question 4:

Given a relation R(ABDLPT) having FDs

$\{B \rightarrow PT, A \rightarrow D, T \rightarrow L\}$

Identify the prime attributes and non prime attributes.

Solution :

$(AB)^+$: {ABPTDL}

$(A)^+$: {DA}

$(B)^+$: {BPTL}

$\Rightarrow$ Candidate Keys {AB}

⇒ Prime Attributes {A,B}
⇒ Non Prime Attributes {D,L,P,T}

Question 5:
Given a relation R(ABCDEFGH) having FDs
{$E \rightarrow G$, $AB \rightarrow C$, $AC \rightarrow B$, $AD \rightarrow E$, $B \rightarrow D$, $BC \rightarrow A$}
Identify the prime attributes and non prime attributes.
Solution :
$(ABFH)^+$: {ABCDEFGH}
$(A)^+$: {A}
$(B)^+$: {BD}
$(F)^+$: {F}
$(H)^+$: {H}
(AB) : {ABCDEG}
(AF) : {AF}
(AH) : {AH}
(BF) : {BFD}
(BH) : {BHD}
(FH) : {FH}
(ABF) : {ABFCDEG}
(ABH) : {ABHCDEG}
(AFH) : {AFH}
(BFH) : {BDFH}
None of the proper sets of {ABFH} will determine all the attributes.
So {ABFH} ⇒ minimal super key or candidate key

(A B FH)
↓
(A (AC) FH) ⇒ (ACFH)

(A BFH)
↓
((BC) BFH) ⇒ (BCFH)
⇒ Candidate Keys {ABFH, ACFH, BCFH}
⇒ Prime Attributes {A,B,C,F,H}
⇒ Non Prime Attributes {D,E,G}

Question 6 :
Given a relation R(ABCDE) having FDs
{$A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$}
Identify the prime attributes and non prime attributes.
Solution :
$(A)^+$: {ABCDE} ⇒ (Candidate Key)
$(E)^+$: {ABCDE} ⇒ (Candidate Key)

⇒ Candidate Keys {A,E}
⇒ Prime Attributes {A,E}
⇒ Non Prime Attributes {B,C,D}

Question 7 :
Consider a relation scheme R(ABCDEH) on which the following Functional Dependencies hold:
{A → B, BC → D, E → C, D → A}.
What are the candidate keys of R and identify the prime attributes and non prime attributes.
Solution :
$(BEH)^+$: {BEHCDA} ⇒ Super Key
$(B)^+$: {B}
$(E)^+$: {EC}
$(H)^+$: {H}
$(BE)^+$: {BECDA}
$(BH)^+$: {BHD}
$(EH)^+$: {EHC}

None of the proper sets of {BEH} will determine all the attributes.
So, {BEH} ⇒ minimal super key or candidate key.

(B EH)
↑
(A EH)
↑
(D EH)
⇒ Candidate Keys {BEH, AEH, DEH}
⇒ Prime Attributes {A,B,D,E,H}
⇒ Non Prime Attributes {C}

Equivalence of Sets of Functional Dependencies Example -

1. Let F & G are two functional dependency sets. These two sets F & G are equivalent if $E^+ = F^+$. Equivalence means that every functional dependency in F can be inferred from G, and every functional dependency in G an be inferred from F. →
2. F and G are equal only if
   i. F covers G- means that all functional dependency of G are logically numbers of functional dependency set F⇒F⊇G.
   ii. G covers F-means that all functional dependency of F are logically members of functional dependency set G⇒G⊇F

F covers G	True	True	False	False
G covers F	True	False	True	False
Result	F=G	F⊃G	G⊃F	No Comparison

Question 1 :
Consider the two sets F and G with their FDs as below :

F :
A → C
AC → D
E → AD
E → H

G:
A → CD
E → AH

Check whether two sets are equivalent or not.
Solution :
Step 1 : Take Set F and Check G is covered from F or not.

$(A)^+ = \{ACD\}$
$(E)^+ = \{EADHC\}$
Hence, both A → CD and E → AH are covered.

⇒ G is derived from F. Hence G is covered by F.
⇒ F ⊇ G . ....(1)

Step 2 : Take Set G and Check F is covered from G or not.

$(A)^+ = \{ACD\}$
$(AC)^+ = \{ACD\}$
$(E)^+ = \{EAHCD\}$
Hence F = {A → C, AC → D, E → AD, E → H} is covered.

⇒ F is derived from G. Hence F is covered from G.
⇒ G ⊇ F. ....(2)

From (1) and (2), F and G are equivalent.
Question 2 :
Consider the two sets P and Q with their FDs as below :

P :
A → B
AB → C

Q :
A → BC
D → AE

D $\rightarrow$ ACE

Check whether two sets are equivalent or not.

Solution :

Step 1 : Take Set P and Check Q is covered from P or not.

$(A)^+ = \{ABC\}$

$(D)^+ = \{DACEB\}$

Hence, both A $\rightarrow$ BC and D $\rightarrow$ AE are covered.

$\Rightarrow$ Q is derived from P. Hence Q is covered by P.

$\Rightarrow P \supseteq Q$ . ....(1)

Step 2 : Take Set Q and Check P is covered from Q or not.

$(A)^+ = \{ABC\}$

$(AB)^+ = \{ABC\}$

$(D)^+ = \{DAEBC\}$

Hence P = {A $\rightarrow$ B, AB $\rightarrow$ C, D $\rightarrow$ ACE} is covered.

$\Rightarrow$ P is derived from Q. Hence P is covered by Q.

$\Rightarrow Q \supseteq P$. ....(2)

From (1) and (2), P and Q are equivalent.

Functional Dependency Set Closure (F+)

Functional Dependency Set Closure of F is the set of all functional dependencies that are determined by it.

Example of Functional Dependency Set Closure :

Consider a relation R(ABC) having following functional dependencies : F = { A $\rightarrow$ B, B $\rightarrow$ C } To find the Functional Dependency Set closure of F^+ :

$(\Phi)^+ = \{\Phi\}$

$\Rightarrow \Phi \rightarrow \Phi$

$\Rightarrow$ 1 FD

$(A)^+ = \{ABC\}$

$\Rightarrow$ A $\rightarrow$ Φ, A $\rightarrow$ A, A $\rightarrow$ B, A $\rightarrow$ C,
A $\rightarrow$ BC, A $\rightarrow$ AB, A $\rightarrow$ AC, A $\rightarrow$ ABC

$\Rightarrow$ 8 FDs = $(2)^3$

... where 3 is number of attributes in closure

$(B)^+ = \{BC\}$

$\Rightarrow$ B $\rightarrow$ Φ, B $\rightarrow$ B, B $\rightarrow$ C, B $\rightarrow$ BC

$\Rightarrow$ 4 FDs = $(2)^2$

$(C)^+ = \{C\}$

$\Rightarrow$ C $\rightarrow$ Φ, C $\rightarrow$ C

$\Rightarrow$ 2 FDs = $(2)^1$

$(AB)^+ = \{ABC\}$

$\Rightarrow$ AB $\rightarrow$ Φ, AB $\rightarrow$ A, AB $\rightarrow$ B, AB $\rightarrow$ C,
AB $\rightarrow$ AB, AB $\rightarrow$ BC, AB $\rightarrow$ AC, AB $\rightarrow$ ABC

$\Rightarrow$ 8 FDs = $(2)^3$

$(BC)^+$ = {BC}

$\Rightarrow$ BC $\rightarrow \Phi$, BC $\rightarrow$ B, BC $\rightarrow$ C, BC $\rightarrow$ BC

$\Rightarrow$ 4 FDs = $(2)^2$

$(AC)^+$ = {ABC}

$\Rightarrow$ AC $\rightarrow \Phi$, AC $\rightarrow$ A, AC $\rightarrow$ C, AC $\rightarrow$ C,
AC $\rightarrow$ AC, AC $\rightarrow$ AB, AC $\rightarrow$ BC, AC $\rightarrow$ ABC

$\Rightarrow$ 8 FDs = $(2)^3$

$(ABC)^+$ = {ABC}

$\Rightarrow$ ABC $\rightarrow \Phi$, ABC $\rightarrow$ A, ABC $\rightarrow$ B, ABC $\rightarrow$ C,
ABC $\rightarrow$ BC, ABC $\rightarrow$ AB, ABC $\rightarrow$ AC, ABC $\rightarrow$ ABC

$\Rightarrow$ 8 FDs = $(2)^3$

So, the Functional Dependency Set Closure of $(F)^+$ will be :
F^+ = {
$\Phi \rightarrow \Phi$, A $\rightarrow \Phi$, A $\rightarrow$ A, A $\rightarrow$ B, A $\rightarrow$ C, A $\rightarrow$ BC, A $\rightarrow$ AB, A $\rightarrow$ AC, A $\rightarrow$ ABC,
B $\rightarrow \Phi$, B $\rightarrow$ B, B $\rightarrow$ C, B $\rightarrow$ BC, C $\rightarrow \Phi$, C $\rightarrow$ C, AB $\rightarrow \Phi$, AB $\rightarrow$ A, AB $\rightarrow$ B,
AB $\rightarrow$ C, AB $\rightarrow$ AB, AB $\rightarrow$ BC, AB $\rightarrow$ AC, AB $\rightarrow$ ABC, BC $\rightarrow \Phi$, BC $\rightarrow$ B,
BC $\rightarrow$ C, BC $\rightarrow$ BC, AC $\rightarrow \Phi$, AC $\rightarrow$ A, AC $\rightarrow$ C, AC $\rightarrow$ C, AC $\rightarrow$ AC, AC $\rightarrow$ AB,
AC $\rightarrow$ BC, AC $\rightarrow$ ABC, ABC $\rightarrow \Phi$, ABC $\rightarrow$ A, ABC $\rightarrow$ B, ABC $\rightarrow$ C, ABC $\rightarrow$ BC,
ABC $\rightarrow$ AB, ABC $\rightarrow$ AC, ABC $\rightarrow$ ABC
}

The Total FDs will be :
1 + 8 + 4 + 2 + 8 + 4 + 8 + 8 = 43 FDs
Consider another relation R(AB) having following functional dependencies : F = { A $\rightarrow$ B, B $\rightarrow$ A }

To find the Functional Dependency Set closure of F^+ :
$(\Phi)^+$ = {Φ} $\Rightarrow$ 1
$(A)^+$ = {AB} $\Rightarrow$ 4 = $(2)^2$
$(B)^+$ = {AB} $\Rightarrow$ 4 = $(2)^2$
$(AB)^+$ = {AB} $\Rightarrow$ 4 = $(2)^2$
Total = 13

How to Find Candidate Key using Functional Dependencies -

To identify Candidate Key, Let R be the relational schema, and X be the set of attributes over R. X^+ determine all the attributes of R, and therefore X is said to be superkey of R. If there are no superflous attributes in the Super key, then it will be Candidate Key. **In short, a minimal Super Key is a Candidate Key.**

Example/Question 1 :

Let R(ABCDE) is a relational schema with following functional dependencies.

AB $\rightarrow$ C

DE $\rightarrow$ B

CD $\rightarrow$ E

Step 1: Identify the SuperKeys -

ACD, ABD, ADE, ABDE, ACDB, ACDE, ACDBE.

Step 2: Find minimal super key -

Neglecting the last four keys as they can be trimmed down, so, checking the first three keys (ACD, ABD and ADE)

For SuperKey : ACD

$(A)^+ = \{A\}$ - {Not determine all attributes of R}

$(C)^+ = \{C\}$ - {Not determine all attributes of R}

$(D)^+ = \{D\}$ - {Not determine all attributes of R}

For SuperKey : ABD

$(A)^+ = \{A\}$ - {Not determine all attributes of R}

$(B)^+ = \{B\}$ - {Not determine all attributes of R}

$(D)^+ = \{D\}$ - {Not determine all attributes of R}

For SuperKey : ADE

$(A)^+ = \{A\}$ - {Not determine all attributes of R}

$(D)^+ = \{D\}$ - {Not determine all attributes of R}

$(E)^+ = \{E\}$ - {Not determine all attributes of R}

Hence none of proper sets of SuperKeys is not able to determine all attributes of R, So ACD, ABD, ADE all are minimal superkeys or candidate keys.

Example/Question 2 :

Let R(ABCDE) is a relational schema with following functional dependencies -

AB $\rightarrow$ C

C $\rightarrow$ D

B $\rightarrow$ EA

Find Out the Candidate Key ?

Step 1: Identify the super key

(AB^+) : {ABCDE} $\Rightarrow$ Superkey

(C^+) : {CD} $\Rightarrow$ Not a Superkey

(B^+) : {BEACD} $\Rightarrow$ Superkey

So, Super Keys will be B, AB, BC, BD, BE, BAC, BAD, BAE, BCD, BCE, BDE, BACD, BACE, BCDE, ABDE, ABCDE

Step 2: Find minimal super key -
Taking the first one key, as all other keys can be trimmed down -
(B^+) : {EABCD} {determine all the attributes of R}

Since B is a minimal SuperKey ⇒ B is a Candidate Key.
So, the Candidate Key of R is - B.

How to Find Super Key From Functional Dependencies

Let R be the relational schema, and X be the set of attributes over R. If X^+ determine all the attributes of R, then X is said to be superkey of R. **To Identify Super keys, we need to follow some steps -**

- Compute Closure for the attributes of combination of attributes on the LHS of Functional Dependency i.e. Determinants.
- If any closure includes all the attributes, then that can be declared as a key for the table.

To Identify Super Key -
For Example :
Let R(ABCDE) is a relational schema with following functional dependencies.
$AB \rightarrow C$
$DE \rightarrow B$
$CD \rightarrow E$
Step 1: Identify the closure of LHS of FD -
$(AB)^+ = ABC$
$(DE)^+ = DEB$
$(CD)^+ = CDE = CDEB$ {as $DE \rightarrow B$}
No SuperKey Found in step 1.
Step 2: If no super Key found from step 1, then follow step 2 to find a new key by applying augment rule.
Apply Augment rule until all attributes are mentioned in the closure result. So, choosing $(CD)^+$ as it contains more attributes than others one i.e. CDEB,

$(ACD)^+ = ABCDE$ {By augment Rule}
Hence $(ACD)^+$ determines all the attributes of R. So
ACD is a SuperKey.

As ACD is a super key, we make the combination of remaining attributes with ACD. So, superkeys are -
ACDB,
ACDE,
ACDBE.
Step 3: Follow step 3 to Identify more superkeys from new SuperKey (ACD) by applying Pseudo Transitive rule -

Check the other Functional Dependencies in which the LHS is a subset of new super key, and that on its RHS contains some other attribute of new superkey.

There is only one i.e. AB → C, Remove the attributes of the RHS of these from new superkey.

Doing so gives you a key, that are certainly superkeys, but not necessarily irreducible ones:
A(AB)D = ABD : SuperKey

Other Super Keys will be
ADBE,
ABDC, (Already Found)
ABDCE. (Already Found)

Repeat the procedure again for the new superkey(ABD) till we get all superkeys.
Step 3 Continued... (To find SuperKey from ABD)
For ABD, We have a functional dependency again i.e. {DE → B}. So,
A(DE)D = ADE : SuperKey

Other Super keys will be -
ADEB, (Already Found)
ADEC, (Already Found)
ADEBC. (Already Found)
Step 3 Continued... (To find SuperKey from ABD)
For ADE, ⇒ CD → E, So, AD(CD) = ACD : SuperKey that we already have
So all the superkeys for R will be - ACD, ACDB, ACDE, ACDBE ABD, ABDE, ADE.

Membership Test for Functional Dependency or Identification of additional Functional Dependencies by Closure Set -

What is the Membership Test for Functional Dependency ??

Let F be the Functional Dependency set. $X \rightarrow Y$ be any non trivial functional dependency. $X \rightarrow Y$ implied in Functional Dependency Set only if X^+ determines Y.

Method :

- To determine, whether a particular Functional Dependency can be derived from the original set or not, compute the closure from the original functional dependencies. If the closure contains the RHS of the original functional dependency or the determined attributes, then it will become additional functional dependency. For example, $AB \rightarrow DE$ is a functional dependency. Compute the closure of AB=AB+. If AB^+ includes 'DE', then $AB \rightarrow DE$ is true and will become additional functional dependency.
- Repeatedly check all additional functional dependencies to find out new functional dependency set.

Question :

Given the following FDs

$A \rightarrow BC$
$CD \rightarrow E$
$E \rightarrow C$
$D \rightarrow AEH$
$ABH \rightarrow BD$
$DH \rightarrow BC$

a) Find out, whether $BCD \rightarrow H$ or not from the following Functional Dependencies?

Solution :

Compute the closure of BCD = BCD^+

BCD^+ = BCD
= ABCDEH {as $D \rightarrow AEH$}

Hence $BCD \rightarrow H$ is true as the closure contains H.

b) Find out, whether $AED \rightarrow C$ or not from the above given Functional Dependencies?

Solution:

Compute the closure of AED.

ADE^+ = ADE
= ABCDE {as $A \rightarrow BC$}

Hence $AED \rightarrow C$ is true as the closure contains C

Question 2:

Following dependencies are given:

$AB \rightarrow CD$
$AF \rightarrow D$
$DE \rightarrow F$

$C \rightarrow G$
$F \rightarrow E$
$G \rightarrow A$

Which of the following is false ?

(a) $(CF)^+ = \{ACDEFG\}$
(b) $(BG)^+ = \{ABCDG\}$
(c) $(AF)^+ = \{ACDEFG\}$
(d) $(AB)^+ = \{ABCDG\}$

Solution : option (c) is false . as $(AF)^+$ will give {AFED} but not {ACDEFG}

Question 2:

Consider a relational scheme R with attributes A,B,C,D,F and the FDs

$A \rightarrow BC$
$B \rightarrow E$
$CD \rightarrow EF$

Prove that functional dependency $AD \rightarrow F$ holds in R.

Solution :

Compute the closure of AD.

$(AD)^+ = ADBCEF$

Hence $AD \rightarrow E$ is true as the closure contains E.

Minimal Sets of Functional Dependencies or Irreducible Set of Functional Dependencies or Removal of Extraneous Attributes or Canonical Cover -

Some **Terms** Before Making a Move to the Minimal Sets of Functional Dependencies /Canonical Cover -

- **Non Redundant Functional Dependency -** A set F of FDs is non redundant, if there is no proper subset F'of F such that F≡F'.
- **Redundant Functional Dependency -** If a set F' which is a proper subset of F exists, then F is redundant.
- **Extraneous Attributes -** are the unnecessary/extra attributes appearing in individual functional dependencies either on LHS or on RHS.

Our intention is to **remove the Non redundant Functional Dependencies** and **also the removal of extraneous attributes** without changing the closure of F before making a move to the normalization process. The **procedure** to find the minimal sets of functional dependencies is as follows :

- **Step 1 : Apply the Union Simplification Inference Rule.**
- **Step 2 : Removal of Redundant Functional Dependency -** To check whether a functional dependency is redundant or not , first hide that functional from set and then find closure attributes those are at left of that functional dependency without using reflexivity rule , if closure contains same attributes for whom we are finding closure then functional dependency is redundant, remove this functional dependency from the set.
- **Step 3 : Removal of Extraneous Attributes -** Look for the FDs having more than one attribute on the LHS. Remove an attribute from the FD and take the closure of remaining attributes from the other FDs. If the removed attribute exists in the closure, then it is a redundant attribute/extraneous attribute and remove it from that FD. Repeat this process for all the remaining Attributes.
- **Step 4:** If step 3 = Success(i.e. found an extraneous attribute), then apply step 2 again followed by step 4. If step 3 = fails (i.e no extraneous attributes found) then go to step 4.
- **Step 5 : Apply the union rule to get the minimal sets of functional dependencies.**

Questions on Minimal Cover or Irreducible Sets or Canonical Cover -

Question 1 :Find the minimal cover or irreducible set from the following functional dependencies :

AB → CD

BC → D

Solution :

Step 1 : Union Simplification :

AB → C ..(1)

AB → D ..(2)

BC → D ..(3)

Step 2 : Removal of Redundant set of FDs -

(i) Find the closure of LHS of (1), i.e. Compute (AB)+ from (2) and (3) by hiding (1). If we get RHS of (1) in the closure,i.e. 'C', then (1) is redundant, otherwise non redundant.

(AB)+ = {ABD}

as (AB)+ will not determine C, So AB → C is non redundant.

(ii) Similarly for (2), Compute (AB)+ from (1) and (3) by hiding (2).

(AB)+ = {ABCD}

as (AB)+ will determine D, therefore, AB → D is redundant.

Now, Remaining FDs = {AB→C, BC→D}

(iii) For (3), Compute (BC)+ from (1) by hiding (3).

(BC)+ = {BC}

as (BC)+ will not determine D, so, BC → D is also non redundant.

Step 3 : LHS Simplification or Removal of Extraneous Attributes -

FDs remained after step 2

AB→C .. (3)

BC→D .. (4)

<u>AB → C</u>

(i) Remove A from AB → C, and find B+ from (4). If the attribute A exists in the closure, then A is said to be redundant.

B+ = {B} ⇒ A is Non Redundant

(ii) Remove B from AB → C and find A+ from (4).

A+ = {A} ⇒ B is Non Redundant.

<u>BC → D</u>

(i) Remove B from BC → D, and find C+ from (3).

C+ = {C} ⇒ B is Non Redundant

(ii) Remove C from BC → D and find B+ from (3).

B+ = {B} ⇒ C is Non Redundant

Apply step 2 again if we find any redundant attribute, otherwise skip

So Finally, Irreducible set will be

AB→C

BC→D

The another Example will give you more clarification about solving minimal cover questions.

Question 2 :

Find the minimal cover or irreducible set from the following functional dependencies :

ABCD → E ..(1)

E → D ..(2)

AC → D ..(3)

A → B ..(4)

Solution :

step 1 : No need for union simplification as FDs are already simplified.

Step 2 : Removal of redundant set of FDs -

(i) For (1), Compute (ABCD)+ from (2) (3) and (4) by hiding (1).

(ABCD)+ = {ABCD}

⇒ ABCD → E is non redundant.

(ii) For (2), Compute (E)+ from (1) (3) and (4) by hiding (2).

(E)+ = {E}

⇒ E → D is non redundant.

(iii) For (3), Compute (AC)+ from (1) (2) and (4) by hiding (3).

(AC)+ = {ACB}

⇒ AC → D is non redundant.

(iv) For (4), Compute (A)+ from (1) (2) and (3) by hiding (4).

(A)+ = {A}

⇒ A → B is non redundant.

Step 3 : LHS Simplification or Removal of Extraneous Attributes -

FDs remained after step 2 is same as in step 1.

<u>ABCD → E</u>

(i) Remove A from ABCD → E, and find BCD+ from (2) (3) and (4). If attribute A exists in the closure, then A is said to be redundant.

(BCD)+ = {BCD} ⇒ A is Non Redundant

(ii) Remove B from ABCD → E and find ACD+ from (2) (3) and (4).

(ACD)+ = {ACDB} ⇒ B is Redundant.

⇒ FD will now reduced to ACD → E

(iii) Remove C from ACD → E, and find AD+ from (2) (3) and (4).

(AD)+ = {ADB} ⇒ C is Non Redundant.

(iv) Remove D from ACD → E, and find AC+ from (2) (3) and (4).

(AC)+ = {ACDB} ⇒ D is Redundant.

⇒ FD will now reduced to AC → E

<u>AC → D</u>

(i) Remove A from AC → D and find C+ from (1) (2) and (4).

(C)+ = {C} ⇒ A is non redundant.

(ii) Remove C from AC → D and find A+ from (1) (2) and (4).

(A)+ = {AB} ⇒ C is non redundant.

FDs remained after step 3 are :

AC → E ..(5)

E → D ..(6)

AC → D ..(7)

A → B ..(8)

Step 4 : Applying step 2 again to find redundant FDs -

(i) For (5), Compute (AC)+ from (6) (7) and (8) by hiding (5).

(AC)+ = {ACDB}

⇒ AC → E is non redundant.

(ii) For (6), Compute (E)+ from (5) (7) and (8) by hiding (6).

(E)+ = {E}

⇒ E → D is non redundant.

(iii) For (7), Compute (AC)+ from (5) (6) and (8) by hiding (7).

(AC)+ = {ACEDB}

⇒ AC → D is redundant. So, excluded.

(iv) For (8), Compute (A)+ from (5) and (6) by hiding (8).

(A)+ = {AB}

⇒ A → B is non redundant.

So Finally, Irreducible set will be

AC → E

E → D

A → B

Questions on Super Keys and Candidate Keys using Closure

Identify Super Keys and Candidate keys :

Question 1 :

Let R(ABCDE) is a relational schema, where

$(AB)^+ = ABCDE$

$(A)^+ = ABCDE$

Is AB: Candidate Key or Not??

Solution :

AB : Not a Candidate Key, AB is only : Super Key

Question 2 :

Let R(ABCDE) is a relational Schema having FDs

$\{AB \rightarrow C, C \rightarrow D, B \rightarrow E\}$

Find out the Candidate Key ?

Solution :

$(AB^+) : \{ABCDE\}$ ⇒ super key

$(A^+) : \{A\}$ ×

$(B^+) : \{EB\}$ ×

∴ AB : minimal superkey ⇒ Candidate Key. No subset of its attributes is a key.

Question 3 :

Let R(ABCDE) is a relational schema having FDs

$\{AB \rightarrow C , C \rightarrow D, B \rightarrow EA\}$

Find Out the Candidate Key ?

Solution :

$(AB^+) : \{ABCDE\}$ ⇒Superkey

$(A^+) : \{A\}$

$(B^+) : \{EABCD\}$ ⇒ Superkey

⇒ B is Candidate Key.

Question 4 :

Let R(ABCDE) is a relational schema having FDs

$\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$

Find out the Candidate Key ?

Solution :

$(AE^+) :\{ABCDE\}$ ⇒ SuperKey

$(A^+) : \{ABCD\}$

$(E^+) : \{E\}$

AE : Candidate Key. No subset of its attributes is a key.

Question 5 :

Let R(ABCDEF) is a relational schema having FDs

$\{A \rightarrow BCDEF, BC \rightarrow ADEF, B \rightarrow C, D \rightarrow E\}$

Find out the Candidate Key ?

Solution:

$(A)^+$: ABCDEF ⇒ (SuperKey)

$(BC)^+ : \{BCADEF\}$ ⇒ (SuperKey)

$(B)^+$: {BCADEF}
$(C)^+$: {C}
{A, B} ⇐ Candidate Key. No subset of its attributes is a key.

Question 6:

Given the following set F of functional dependencies for relation schema R = {A, B, C, D, E}.

{A -> BC, CD -> E, B -> D, E -> A}

List the candidate keys for R.

Solution :

$(A)^+$: ABCDE ⇒ (SuperKey)
$(E)^+$: EABCD ⇒ (SuperKey)
$(CD)^+$: CDEAB ⇒ (SuperKey)
$(CB)^+$: CBDEA ⇒ (SuperKey)

Any combination of attributes that includes those is a superkey.
From above , the minimal super keys are ⇒ A, E, CD and BC.
Hence, the candidate keys are A, E, CD, BC.

Question 7:

Consider a relation R(A,B,C,D,E) with the following dependencies:
{AB-> C, CD -> E, DE -> B}
Is AB a candidate key of this relation? If not, is ABD? Explain your answer.
No. The closure of AB does not give you all of the attributes of the relation.
For ABD,
(ABD)+ = ABDCE ⇒ Super Key
(A) = {A}
(B) = {B}
(D) = {D}
⇒ ABD is a candidate key. No subset of its attributes is a key.

Question 8 :

Consider a relation with schema R(A,B,C,D) and FDs {AB -> C, C -> D, D -> A}. What are all candidate keys of R?
$(AB)^+$: ABCD ⇒ (SuperKey)
$(A)^+$: A ⇒ (Not able to determine all the attributes)
$(B)^+$: B ⇒ (not able to determine all the attributes)
$(DB)^+$: ABCD ⇒ (SuperKey)
$(CB)^+$: ABCD ⇒ (SuperKey)
$(D)^+$: DA ⇒ (Not able to determine all the attributes)
$(C)^+$: CDA ⇒ (Not able to determine all the attributes)

⇒ By calculating an attribute closure we can see the candidate keys are: AB, BC, and BD.