Database: A database is a collection of related data which represents some aspect of the real world.
A (das). database system is designed to be built and populated with data for a certain task.

Database Management System (DBMS)
It is a software for storing and retrieving users data while considering appropiate security measures.

- It consist of group of programs which can manipulate the database.
- DBMS accepts the request for data from an application and instructs the Operating System to provide specific data.
- In large system, DBMS helps user and other 3rd-party software to store and retrieve data.

Database System are introduced to solve following problems.

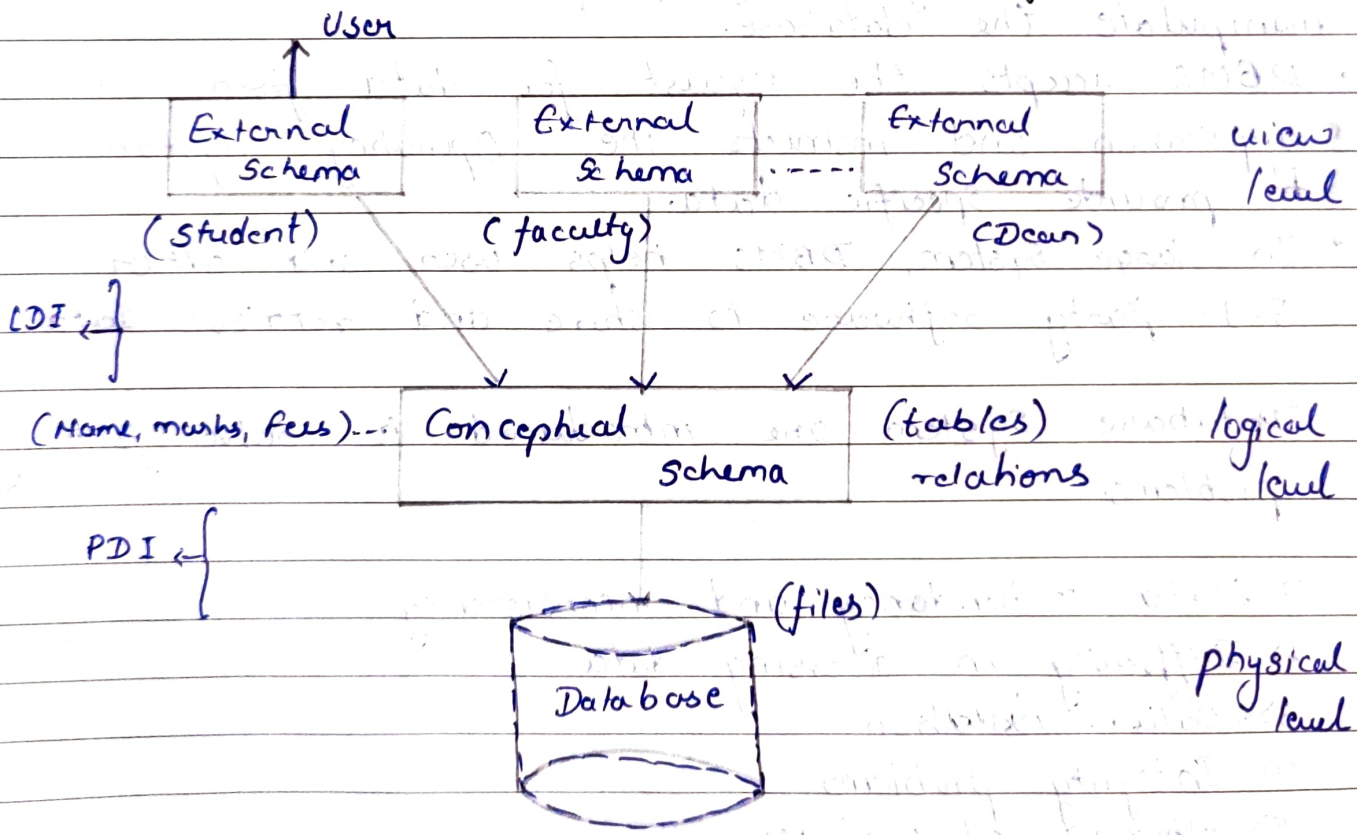
1) Data redundancy and inconsistency
2) Difficulty in accessing data
3) Data isolation
4) Integrity problems
5) Atomicity of updates
6) Concurrent acces by multiple users
7) Security problems

* Tier 2 v/s Tier 3 Architecture

All web applications are Tier3 Architecture. and offline booking in railway considerd as Tier2 architecture.

* Schema = Logical Representation of data.
In RDMS schema refers to tables.

* Three schema Architecture (Three level of abstraction)



- Inactual data stored in the form of files in databases.

The main objective of 3 level of architecture is to enable multiple multiple users to access the same data with personalized view.

① External Schema: External schema is also known as view schema.
- Each view schema describes the database part that a particular user / group is interested and hides the remaining database from the user group.
- View schema describes the end user interaction with the database system.

② Conceptual level: The Conceptual Schema describes the design of a database at the Conceptual level. Conceptual level is also known as logical level.
- The Conceptual schema describes the (Shape) Structure of database.
- It also describes what data to be stored and relationship among those data.
- Internal details (such as Implementation of data structure is hidden).
- programmer and database administrators work at this level.

③ physical level: The internal (physical) level has internal schema which describes the physical storage structure of the database.
- Internal schema also known as physical schema.
- It uses physical data model.
- It is used to define how data will be stored in a block.

Three schema Architechure Seperates the user's views from the physical structure of the database. This seperation is desirable for the following reason.

1) Different user need different view of the same data.
2) The approach in which a particular user needs to see the data may change over time.
3) The users of the databases should not worry about physical Implementation and internal working of the database.
4) All users should able to access the same data according to their requirements.
5) Database Algo. should be able to change the conceptual structure of the database without affecting the users.
6) Internal structure of database should be unaffected by changes to physical aspects of storage.

* Logical data Independence
changes made in Conceptual schema doesn't reflect in view level.

- Logical data Independence is ability to modify logical schema without causing application program's to be rewritten.

- Logical data independence means if we add some new columns or remove some columns from the table then the user view and programs should not change.
- Logical data independence is more difficult to achieve than physical data independence, since application program are heavily dependent on the logical structure of the data that they can access.

* physical Data Independence

- Physical data independence is ability to modify the physical schema without causing application program to be rewritten.
- Modification at the physical levels are occasionally necessary to improve the performance.
- It means we change physical storage level without affecting the conceptual or external view of data.
- The new changes are absorbed by mapping techniques.

* Keys

It is used to uniquely identify any record or row of data from the table.
It is used to establish and identify relationship between tables.

For Eg Student ID is used as keys in the student table because it is unique for each students.

* Types of keys

① Primary key: The key which is most suitable from <candidate key> set are known as primary key. It is the first key used to identify one and only one instance of entity uniquely.

② Candidate key: A candidate is an set of attributes which uniquely identify as a key. Except primary key remaining attributes are considered a candidate keys.
- The candidate key's are some strong as primary keys.

③ Super key: A super keys is a combination of all possible attributes which can uniquely identify two tuples in a table.
- Super set of candidate keys are super keys i.e, candidate keys + some additional keys.

Eg (Stu-Id, Name) ⇒ Super keys

④ Foreign keys = Foreign keys are the column of the table used to point the primary key of another table.

RollNo	Name	address
1	A	Delhi
2	B	Mumbai
3	A	chd
⋮		

Base table

Course id	CourseN	RollNo
C_1	DBMS	1
C_2	Networks	2

Referencing table

In above table RollNo act as foreign keys. Multiple foreign keys also can exist.

⑤ Alternate keys = one key is chosen as the primary key from these candidate keys, and remaining candidate keys if it exist termed as alternate keys.

ie, $C.k = P.k + A.k$

- The key other than the primary key from the set of candidate keys are known as alternate keys.

⑥ Composit keys = Whenever primary key consist of more than one attributes. It is known as Composit keys. This keys is also known as Concatened keys.

Eg In employee relations, we assume that an employee may assigned to multiple roles. and may work on multiple projects. so primary keys will be Composit of Emp Id, Emp role, Proj ID.

* ER Relationship

- ER diagram or Entity Relationship diagram is a conceptual model that gives the graphical representation of the logical structure of database.
- An ER diagram is mainly composed of following three components.

① Entity set 
② Attributes 
③ Relationship set 

① Entity set = An entity set are the same type of entities.

① Strong Entity set = A strong entity set is an entity set that contains sufficient attributes to uniquely identify its entities.

2) In other words, primary key exist for strong entity set.

② Weak Entity set = A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities.

2) In other words, primary key doesn't Exist for weak Entity set.

3) But It contain partial key called discriminator.

4) Discriminator can identify group of entities from the entity set.

5) Discriminator is represented by underlying with a dashed line.

* Attributes: Attributes are the descriphive property which are owned by each Enhity of an enhity set.

* Types of atmibutes:-

① Simple attributes: Simple atmibutes are those atmibutes which can not be divided further i.e, Age.

② Composit atmibutes: Composit atmibutes are those atmibutes which are composed of many other simple atmibutes. i.e, Name, address.

③ Single valued atmibutes: A single valued atmibute can have only a single value.
for example a person can have only one DOB, age etc.
But it can be further simple, or composit atmibutes i.e, DOB is Composit atmibute and age is simple atmibutes.

④ Multi-valued Atmibutes: Multi-valued atmibutes are those atmibutes which can take more than for a given enhity from enhity set.
for example Mobile No. address

⑤ Stored Atmibutes: Stored atmibutes are the atmibutes that are physically stored in the database Represented by Normal eclipse.

⑥ Derived atmibutes: Derived atmibutes are those which can be derived from other atmibutes i.e, Age derived from DOB.

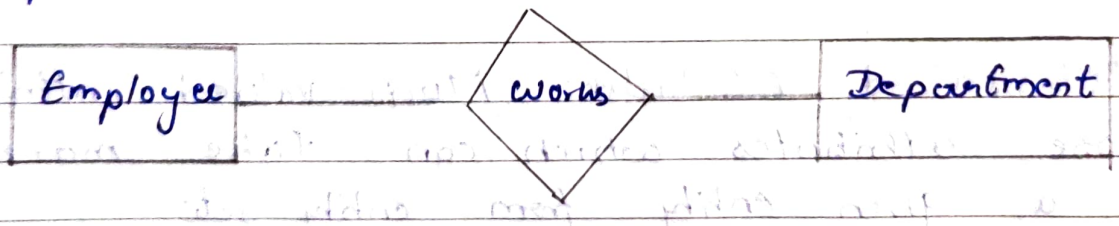
⑦ Key Attributes: Key attributes are unique attributes which can uniquely identify an entity from the entity set.
for example Roll No. is an key attribute.
It is denoted by underline.

⑧ Required Attributes: These attributes are mandatory attributes which can't leaved as NULL.

⑨ Complex Attributes: Those attributes that are Complex as they made from Composit attributes as well Multivalued attributes.
Eg Address is an example of complex attributes.

* ° of Relationship / Types of Relationship / Cardinality

Cardinality Constraint defines the maximum number of relationship instances in which entity can participate.



① One to One Relationship: An entity in set A can be associated with at most one entity in set B.
Or,
Entity in set B can be associated with atmost one entity in set B. A.

Date: YOUVA

fk fk

Eid	Ename	age
E_1	A	20
E_2	B	25
E_3	C	28
E_4	A	24
E_5	B	25
⋮		

EID	DID
E_1	D_1
E_2	D_2
E_3	D_3

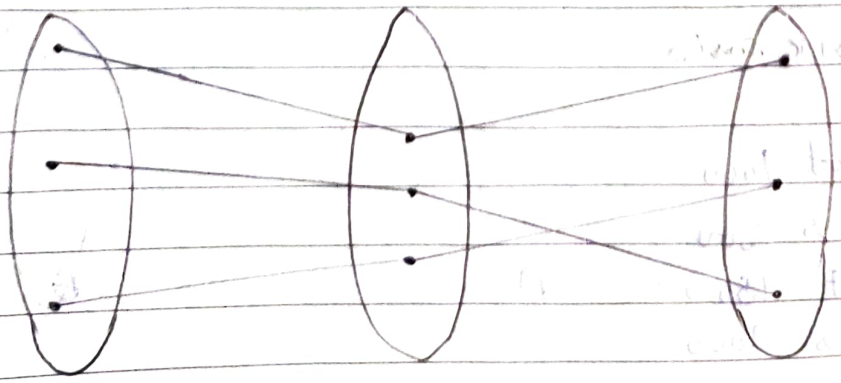
works.

DID	Dname	Loc
D_1	IT	Bangl
D_2	HR	Delhi
D_3	HR	Goa

Primary key can be either EID or DID.

Eid	Ename	age	DID
E_1	A	20	D_1
E_2	B	25	D_2
E_3	C	28	D_3
E_4	A	24	-
E_5	B	25	-

} Combined new table from 1-1 relationship



(2) One - to Many Relationship

An entity in set A can be associated with (n) any number of entities in set B.
or,
Entity in set B associated with atmost one entity in set A.

P.K

Id	Name	city
C_1	A	Goa
C_2	B	Delhi
C_3	C	Kota
C_4	A	Mumbai

F.K F.K

ID	O_no	Date
C_1	O_1	
C_2	O_2	
C_2	O_3	
C_2	O_4	

P.K

O_no	Item Name	Cost
O_1	Bucket	1000
O_2	Shoes	2000
O_3	Shirt	1500
O_4	Jeans	2000

↓ (My contain Duplicate)

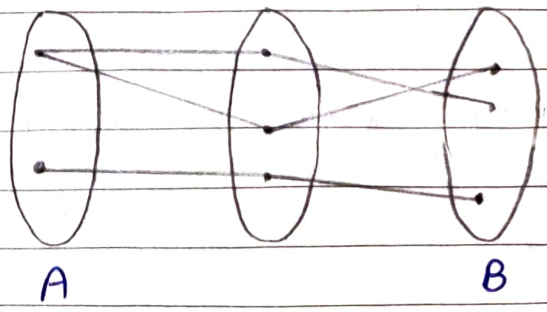
Relation table also contain attribute known as descriptive attributes.

(Unique attributes)

Primary Key : O_No.

Table-2 & Table-3 can merged and forming Single table.

Id	O_No	Item Name	Cost
C_1	O_1	Bucket	1000
C_1	O_2	shoes	2000
C_2	O_3	Shirt	1500
C_2	O_4	Jeans	2000

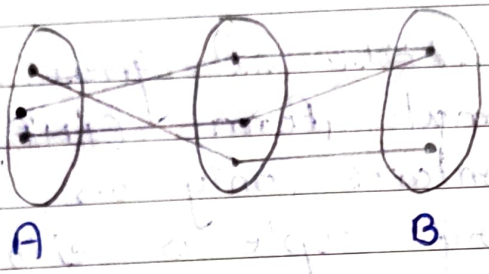


③ Many-to-one Relationship

A entity in set A associated with at most one entity in set B.

Or,

entity in set B associated with n number entity in set A.



④ Many to Many Relationship (M-N)

A entity in set A associated with N number of entity in set B.

Or,

An entity in set B associated with N number of entity in set A.

Roll No	Name	Age
1	A	16
2	B	17
3	A	16
4	C	17
5	D	15

Roll No	C-id
1	C_1
2	C_2
1	C_4
2	C_1
3	C_3

C-id	Name	credit
C_1	Maths	4
C_2	Physics	4
C_3	Chemistry	4
C_4	Hindi	4

primary key doesn't exist in relational table
↳ Reduction of tables.

Composit key = Roll No cid

* Normalization

It is a technique to remove or reduce the redundancy from a table.

* Types of Normal forms

① First Normal form (1NF): A given relation is called in first Normal form (1NF) if each cell of the table contains only an atomic value i.e, attribute of every tuple is either single valued or null value.

Or,

Table should not contain any Multivalued attribute.

- Converting Table In 1st Normal form

Roll No	Name	Course
1	Sai	C/C++
2	Harsh	Java
3	others	C/DBMS

→ Multivalued attributes

Primary key = RollNo

② Method - I

RollNo	Name	Course
1	Sai	C
1	Sai	C++
2	Harsh	Java
3	others	C
3	others	DBMS

primary key = RollNocourse

ⓑ Method - II

RollNo	Name	Course I	Course II
1	Sai	C	C++
2	Harsh	Java	Null
3	others	C	DBMS

Primary key = RollNo

ⓒ Method - III

RollNo	Name
1	Sai
2	Harsh
3	others

Base Table

Primary key = RollNo

RollNo	course
1	C
1	C++
2	Java
3	C
3	DBMS

Derived Table

Foreign key: RollNo.

primary key: RollNo course

* Rule for 1st Normal form

1) Each attribute should contain atomic values
2) A column should contain value from the same domain.
3) Each column should have unique name.
4) No ordering of Rows and Columns.
5) No duplicate rows.

* closure of Attribute set

The set of all those attributes which can be functionally determined from an attribute set.

* Functional Dependency

A functional dependency $A \rightarrow B$ holds, if two tuples having same value of attribute A also have same value for attribute B.

Determinant ← $X \rightarrow Y$ → Dependend attribute

(X determines Y)
(Y is determined by X)

• Types of Functional Dependency

ⓐ Trivial Functional Dependencies:
A functional dependency $X \rightarrow Y$ is said to be trival if and only if $X \subseteq Y$.
and L.H.S of & R.H.S $\neq \phi$.

ⓑ Non-Trivial Functional Dependencies:
A functional dependency from $X \rightarrow Y$ is said to be Non trival if there (at least) exist one attribute in R.H.S of a functional dependency that is not a part of LHS then it called as non-trival functional dependency.
and L.H.S & R.H.S $= \phi$

• Properties

① Reflexiwity: If Y is a subset of X: $X \rightarrow Y$.
② Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$.
③ Transitive: If $X \rightarrow Y$, $Y \rightarrow Z$ then $X \rightarrow Z$.
④ Union: If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

⑤ Decomposition: If $X \to YZ$ then $X \to Y$ and $X \to Z$.
⑥ Pseudo transitivity: If $X \to Y$ and $WY \to Z$ then $WX \to Z$
⑦ Composition: If $X \to Y$ and $Z \to W$ then $XZ \to YW$

Qus: Use closure Method to determine candidate key?
Ans:

① Relation: R(ABCD)

FD: Functional Dependency: $\{A \to B, B \to C, C \to D\}$

$A^+ = BCDA$
$B^+ = CDB$
$CA^+ = DC$
$D^+ = D$
$(AB)^+ = ABCD$ → Not Candidate key (It's Super key)

Candidate key: {A}
Prime attribute: {A}
N-Prime attribute: {B, C, D}

S.K ⊆ C.K

② Relation: R(ABCD)

FD: $\{A \to B, B \to C, C \to D, D \to A\}$

$A^+ = ABCD$
$B^+ = BCDA$
$C^+ = CDAB$
$D^+ = DABC$

Candidate key: {A, B, C, D}
Prime Attribute: Which is used in Relation function. ie, {A, B, C, D}
Non-Prime Attribute: {ø}

③ Relation: R(ABCDE): E must required to from Candidate by checking right side of FD set.

FD: $\{A \to B, BC \to D, E \to C, D \to A\}$

$AE^+ = AEBCD$ ✓
$BE^+ = BECDA$ ✓
$CE^+ = CE$ X
$DE^+ = DECAB$ ✓

C.K = {(A) E}
{(D) E}
{(B) E}

Step1: First find single Candidate key
Step2: and check A present in right in set. If Yes, replace it and so...

Prime Attributes = {ABDE}

(2) Second Normal Form (2NF):

A given relation is said to be in (2NF) if and only if:

- Relation already exist in 1NF.
- No partial dependency exist in the relation.

ie, $A \to B$ is called a partial dependency only if
A is subset of Candidate key.
B is non-prime attribute.

Partial Dependency: Proper subset of Ck → non prime attributes

Qus-1 ① $R(A, B, C, D, E, F)$.
$F.D = \{A \to B, B \to C, C \to D, D - E\}$

Ans.

$AB\not{C}\not{D}\not{E}F^+ = \{ABCDEF\}$

AF^+ = SUPER KEY
IT MAY OR MAY NOT CANDIDATE KEY
To check, it find proper closer subset.
Ck Never can't be a super key.

Proper Subset of A^+ = ABCDE [Not Sk]
Proper Subset of F^+ = F [Not Sk]

⇒ Ck: AF

Proper subset of Ck: {A, F}
Prime attributes PK: {A, F}
None-Prime attributes: {B, C, D, E}

If prime attributes present right side in functional dependency then there must exist another candidate key.

Since, In above Qustion

proper subset of Ck: ⟶ : Non-prime attributes
∴ Relation is Not in 2 NF.

② R (A, B, C, D)
FD = { AB → CD, C ⟶ A, D → B }

$AB \not\subset \emptyset^+ = ABCD$

SK: $AB^+ = ABCD$ Proper Subset of $A^+ = A$ [Not Sk]
Proper Subset of $B^+ = B$ [Not Sk]

⇒ (AB) Must be Candidate key.

CB DB

Proper Subset of $C^+ = CA$ [Not Sk]
Proper Subset of $B^+ = B$ [Not Sk]

⇒ CB Must be Candidate key.

proper Subset of $D^+ = DB$ [NSk]
proper Subset of $B^+ = B$ [NSk]

⇒ DB Must be Candidate key.

Ck: { AB, CD, DB }

proper subset of Candidate key: { A, B, C, D, }
prime Attributs: { A, B, C, D }
N-proime Attributes: { ∅ }

Since, there are doesn't exist N-prime attributes.
∴, Relation is in 2nd Normal form.

③ $R(A, B, C, D)$

$FD: \{A \to B, B \to C, C \to D\}$

$AB \not\subset D^+ = A\ B\ C\ D$

Sk: A^+ proper subset of A^+ = AB [Not Sk]
~ Closure

Candidiate key = {A}

proper subset of ck = {ϕ}

prime Attributes = ϕ

N-prime Attributes = {A, B, C, D}

Since, proper subset of ck is ϕ

∴ No partial functional dependency.

⇒ Relation is in 2ND Normal Form

Note: If Ck contain only single attribute then, Relation must be in 2ND Normal Form

③ Third Normal Form (3NF): A Given relation is in 3NF if and only if

① Relation already exist in 2NF.
② No transitive dependency exists for non-prime attributes.

$A \to B$ is called in transitive dependency if and only if A is not super key and B is non-prime attributes

Transitive dependency occur when,

$A \to B$ & $B \to C$

$\Rightarrow$, $A \to C$ where A & C are NPA (Non-prime attributes)

and also,

A table is in 3NF if and only if for each of its non-trival functional dependency at least one of the following condition holds.

① LHS is Super key or CK.
② R.H.S is prime attribute.

Qus:1 R(A, B, C, D)
FD: (A → B, B → C, C → D)

Sk: $A^+ = \{A\,B\,C\,D\}$
Sk: $A^+ = \{A\,B\,C\,D\}$
CK: A
P.A = A
N.P.A = B, C, D

∴, Since, there exist Non Prime attribute on R.H.S
∴, Not in 3rd Normal Form. i.e, $B \to C$, $C \to D$
NPA → NPA

Ques2: $R(A,B,C,D,E,F)$?
$FD: \{AB \to CDEF, BD \to F\}$ (PA → NPA)

Ans. $ABDE^+ = \{ABCDEF\}$

Sk: $AB^+ = \{ABCDEF\}$
$A^+ = \{A\}$ Not Sk
$B^+ = \{B\}$ Not Sk

Ck: $\{AB\}$
P.A = $\{A, B\}$
NPA = $\{C, D, E, F\}$

If L.H.S is Prime attribute, then no need to check R.H.S.

Ques3: $R(A, B, C, D, E)$
$FD: \{A \to B, B \to C, C \to D, D \to A\}$

Ans.

$ABDE^+ = \{ABCDE\}$
Sk = E
Ck = $\{E\}$ proper ⊆ of Ck: $\{\emptyset\}$
P.A = $\{E\}$
N.P.A = $\{A, B, C, D\}$

Ques3: $R(A, B, C, D)$
$FD: \{AB \to CD, D \to A\}$

Ans.

$ABD^+ = \{ABCD\}$
Sk = AB $A^+ = \{A\}$, $B^+ = \{B\}$ [Not Sk]
Ck = AB
Ck: $\{AB, DB\}$
PA = $\{A, B, D\}$
NPA = $\{C\}$

④ BCNF (Boyce-Codd Normal Form)

The Relation exist in 3NF.
For each non-trival functional dependency $A' \rightarrow B'$, A is a Super key of a relation.

Qus1: R(A, B, C)
FD: $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

Ans1:

Sk: $A^+ = \{A\,B\,C\}$
Sk: $A = \{ABC\}$

Ck: $\{A, C\}$ } → By checking right side in FD.
Ck: $\{A, C, B\}$
PA = $\{A, B, C\}$
N.P.A = $\{\phi\}$

It must be in 2nd as well as 3rd Normal Form.
As, there is no PD, TD.

BCNF form also exist.

Qus2: R (A, B, C, D, E) → $BC \rightarrow A$, $BC \rightarrow C$, $BC \rightarrow E$
FD: $\{A \rightarrow BCDE,\ BC \rightarrow ACE,\ D \rightarrow E\}$
$A \rightarrow B$, $A \rightarrow D$, $A \rightarrow C$, $A \rightarrow E$

Ans2:

Sk: $A^+ = \{A\,B\,C\,D\,E\}$
Sk: $A = \{A\,B\,C\,D\,E\}$
Ck: $\{A, BC, AC, BA\}$

Since, BC, ~~AC, BA~~ are Not Sk Then, it must be Ck.
AC, BA are Not C.K. i.e, proper of these key is A which already Super key.

PA = $\{A, B, C\}$
N.P.A = $\{D, E\}$

BCNF: Doesn't Exist (D Not S.K)
3NF: Doesn't Exist ($D \rightarrow E$, ie, NPA → NPA)
2NF: Exist

Ques3: R(A, B, C, D, E)
FD: {AB → CDE, D → A}

Ans 5

~~A B C D E~~ $^+$ = {A B C D E}
SK: AB^+ = {A B C D E}

A^+ = A {Not S.k}
B^+ = B {Not S.k}

CK: {A, B} CK: {AB, DB}

D^+ = D A (N.Sk)
B^+ = B (Not Sk)

P.A = {A, B, D}
N.P.A = {C, D, E}

BCNF: ✓ ✗
3rd: ✓ ✓
2nd: ✓ ✓

Highest 3rd NF Exist.

BCNF: ✓ ✗
3rd: ✓ ✓
2nd: ✓ ✓

Highest 3rd NF Exist.

Note: Third Normal form always ensures dependency Preserving decomposition but not in BCNF.

* Decomposition of a Relation

The process of breaking up or dividing a single relation into two or more sub relation is called decomposition of relation.

- Properties of Decomposition

① Lossless Decomposition: Lossless decomposition ensures
a) No information is lost from the original relation during decomposition.
b) When the sub relations are joined back, then same relation must be obtained that was decomposed

② Dependency Preservation: Dependency Preservation ensures
a) None of functional dependencies that on the orginal realation are lost.
b) The sub relation still hold or satisfy the functional dependencies of the orginal relation.

* Types of Decomposition.

① lossless join Decomposition:
Consider a relation R which is decomposed into sub relation R_1, R_2
This decomposition is called lossless join decomposition when the join of the sub relations results in the same relation R that was decomposed.
- For lossless join decomposition, we always have $R_1 \bowtie R_2 \bowtie R_3 \ldots \bowtie R_n = R$ where $\bowtie$ is a natural join operator.

② Lossy Join Decomposition: This decomposition is called lossy join decomposition when the join of Subrelation does not result in the same relation R that was decomposed.

Example of lossy Join decomposition

①

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

→ R_1 (AB)

→ R_2 (BC)

A	B
1	2
2	2
3	3

B	C
2	1
2	2
3	2

Joining Table using Natural Join

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

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

(Spurious Tuples) → Extra rows after rejoining i.e why it's called lossy join.

* Rules of Decomposition

ⓐ Common attribute should be Ck or Sk of either R_1 or R_2 or both.

1st N.form	2nd N. form	3rd N. form	BCNF
① No multivalued attribute. ② Contain only single valued.	① In 1st N.F + ② No partial dependency ③ only contain Full dependency	① In 2nd N.F + ② No transitive dependency. ③ No non-prime should determine non-prime.	① In 3rd N.F + ② L.H.S must be C.k or S.k.

4th N. form	5th N. form
① In BCNF + ② No multivalued dependency. X →→ Y Ram → 3 Phone No 3 Email Id	① In 4th form + ② Lossless decomposition.

Qus: For the following functional Dependecies find the correct minimal cover?

FD: {A → B, C → B, D → ABC, AC → D}

Ans:

Step 1: {A → B, C → B, D → A, D → B, D → C, AC → D}

Step 2: Hide each functional dependency one by one and check that RHS obtained from that another particular dependency or Not if avilable then by call that dependency.

{A⁺ : A, C⁺ : C, D⁺ : DABC, D⁺ : ACB, D⁺ : DAB

MR: {A → B, C → B, D → A, Don't repeat it, D → C, AC → D}

(Must Required)

Page No.
Date
YOUVA

Step3: Try to obtain atomic attribute at LHS.

$AC \rightarrow D$

A^+: A { if closure of A attribute contains C then we will remove C from F.D. }

C^+: CB { same applicable for C }

Final Ans: $\{A \rightarrow B, C \rightarrow B, D \rightarrow AC, AC \rightarrow D\}$

Qus2: R(ABCDEF), check Highest Normal Form?
FD: { $AB \rightarrow C$, $C \rightarrow DE$, $E \rightarrow F$, $F \rightarrow A$ }

Ans2:

Sk: $ABCDEF^+ = \{ABCDEF\}$

Sk: $AB^+ = \{ABCDEF\}$ A^+: {A} [Not S.k]
B^+: {B} [Not S.k]

Sk: $FB^+ = \{ABCDEF\}$ F^+: {FA} [N.S.k]
B^+: {B} [N.S.k]

Sk: $EB^+ = \{ABCDEF\}$ E^+: [N.S.k]
B^+: [N.S.k]

Sk: $DB^+ = \{\phi, DB\}$

Sk: $CB^+ = \{ABCDEF\}$ C^+: {CDEFA} [N.S.k]

Ck: { AB, FB, EB, CB }
P.A = { A, B, C, E, F }
N.P.A = { D }

BCNF:	✓	×	×	×
3rd:	✓	×	✓	✓
2nd:	✓	×	✓	✓

L.H.S is proper subset && R.H.S Non prime then, table not in 2NF.

Ques 3: How to find out Normal form of a relation?

Ans3:

R(ABCDEF)

Ck = { AB, FB, EB, CB }

P.A = { A, B, C, E, F }

Non-P.A = { D }

FD: { AB → C, C → D, C → E, E → F, F → A }

② 2nd	F & F = F 'FD'	T & T = T 'PD'	T & F = F 'f.D'	T & F = F 'f.D'	T & F = F 'f.D'

Partial Dependency exist ⇒ 2nd N.F doesn't exist.

- Converting Table into 2nd Normal form

R (ABCDEF)

R_1 (ABCEF)

R_1 { AB → C, C → E, E → F, F → A }

Ck: { AB, FB, EB, CB }

PA = { A, B, C, E, F }

N.P.A = { D }

T ‖ T = T 'fD'	F ‖ T = T 'fD'	F ‖ T = F 'fD'	F ‖ T = T 'f.D'

R_1 Exists in 3rd Normal form.

R_2 (CD)

R_2 { C → D }

Ct = CD

C is Candidate ke

CK: { C }

PA = { C }

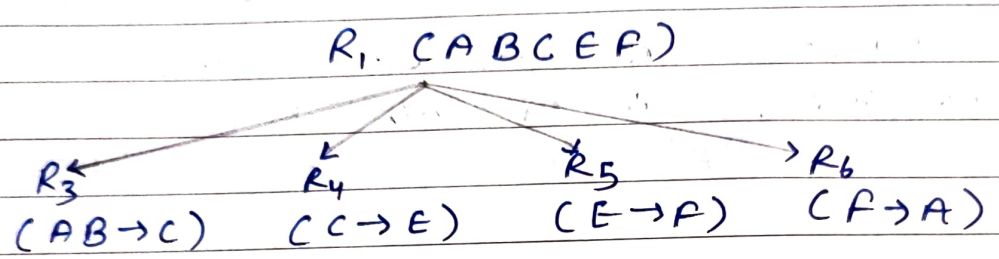
N.P.A = { A, B, D, E, F }

T ‖ F = T 'f.D'

R_1:	$\{AB \to C$,	$C \to F$,	$E \to F$,	$F \to A\}$
BCNF:	check	L.H.S is	Candidate	key
	Tmue ✓	false ×	false ×	false ×

R_2:	$\{C \to D\}$
	Tmue ✓

R_2 is in BCNF



Ck: {AB} Ck: {C} Ck: {E} Ck: {F}

Now, R_3, R_4, R_5 & R_6 is in BCNF.

Ques 4: Which of the following statement is true?

Ans 4:

(a) A relation is in 3rd Normal form is always in BCNF: false

(b) A relation is in 2nd Normal form then, it is not in 1st N.F : false.

(c) A relation is in BCNF then it is in 2NF: True

(d) A relation is in 2NF then it contain partial dependency: false.

Ques 5: R(A B C D E F G H) Find No. no. of C.k ?

$F = \{CH \to G, A \to BC, B \to CFH, E \to A, F \to EG\}$

Ans 5:

Sk: $ABCDEFGH^+ = \{ABCDEFGH\}$

Sk: $AD^+ = \{ABCDEFGH\}$ $A^+ = [N.S.k]$

$D^+ = [N.S.k]$

Sk: $ED^+ = \{EDABCFHEG\}$ E^+: [M·S·k]

$E^+ = \{EABCFHG\}$

Sk: $FD^+ = \{EGFDABCH\}$

Sk: $BD^+ = \{CFHBDEGA\}$

Total No. of C·k = 4

Ques6: Check which of the following Schema is in 3rd N·F but not in BCNF?

Ans 6:

Schema 1: Registration (rollno, Courses)

FD: {rollno → Courses)

C·k = {rollno} Since, L·H·S is Ck ∴ BCNF.

P·A = {rollno}

N·P·A = {courses}

Schema 2: Registration (rollno, courseid, email)

FD: {rollno.courseid → email}

email → rollno.

C·k = {rollno courseid} Since, email (L·H·S is non C·k) ∴ not in BCNF

P·A = {rollno, courseid} also, L·H·S is C·k or R·H·S is P·A ∴, 3rd N·F

N·P·A = {email}

Schema 3: Registration (rollno, courseid, marks, grade)

FD: {rollno courseid → marks, grade, marks → grade}

C·k = {rollno courseid} Since, marks is non C·k ∴ NBCNF

P·A = {rollno, courseid} also, L·H·S is non·C·k and

N·P·A = {marks, grade} R·H·S is N·P·A ∴, Not in 2nd.

also, ⊆ of C·k doesn't determine NPA.

Schema 4: Registration(rollno, courseid, credits)
FD: { rollno courseid → credit, courseid → credit}

C.k = { rollno courseid }
P.A = { rollno, courseid}
N.P.A = {credits}

Since, courseid is Non.C.k ∴, Not in BCNF.
also, also L.H.S contain Non-C.k and
R.H.S contain N.P.A ∴, Not in 3rd N.F
also, proper subset of L.H.S i.e, courseid determine
Non. Prime attribute (N.P.A) ∴, Not in 2nd N.F
∴ The relation is 1st N.F.

- Equivalence of functional dependency:

Qus: X = { A → B, B → C)
Y = { A → B, B → C, A → C}

Ans 1:

Two different Relation are said to be Equivalence relation iff (if and only if) X Covers Y and Y Covers X.

① X Covers Y {X ⊇ Y}

A → ABC {Yes}
B → BC {Yes}
A → ABC {Yes}

Since X ⊇ Y && Y ⊇ X
∴, X ≡ Y

② Y Covers X {Y ⊇ X}

A → ABC {Yes}
B → BC {Yes}

Ques 2: $X = \{AB \to CD, B \to C, C \to D\}$
$Y = \{AB \to C, AB \to D, C \to D\}$

Ans 2:

① $X \supseteq Y$: X Covers Y

$AB \to CDAB$ {Yes}
$AB \to CDAB$ {Yes}
$C \to CDAB$ {Yes}

② $Y \supseteq X$: Y Covers X

$AB \to CDAB$ {Yes}
$B \to \emptyset$ {No}
$C \to D$ {Yes}

Since, $X \supseteq Y$ && $Y \not\supseteq X$
$\therefore X \neq Y$.

Ques 3: Let $R(ABCD)$ with FD: $\{A \to B, B \to C, C \to D, D \to B\}$
R is decomposed into $R_1(AB)$, $R_2(BC)$, $R_3(BD)$?

Ans 3:

- Dependency Preserving Decomposition

$R_1(AB)$	$R_2(BC)$	$R_3(BD)$
$A \to B$ ✓	$B \to C$ ✓	$B \to D$ ✓
$B \to A$ X	$C \to B$ ✓	$D \to B$ ✓

Since, FD: $\{A \to B, B \to C, C \to D, D \to B\}$ all dependency occur in table {may direct or may be indirect}
$\therefore$ FD Preserved.

Qus 4: Let R(ABCD) with functional dependencies {AB → CD, D → A} decompose into R_1(AD) & R_2(BCD)?

Ans 4:

R_1(AD)		R_2(BCD)	
A → D	×	B → CD	×
D → A	✓	C → BD	×
		D → BC	×
		BC → D	×
		CD → B	×
		BD → C	×✓

Since, AB → AB in Table ∴, functional dependencies doesn't Exist.

Qus 5: (Itend)- Identify Highest Normal form?

Ans 5:

R(A,B,C,D,E)

FD: { AB → CDE
D → BE }

P.A = { A, B, D }

Ck: { AB, AD }

N.P.A = { C, E }

BCNF: ✓ ×
3rd: ✓ ×
2nd: ✓ ×

FD is in 1st Normal form.

Qus 6: Identify Highest Normal form?

Ans 6:

R(ABCD)

FD: { AB → C, ABD → C, ABC → D, AC → D }

Ck: { AB }

P.A = { A, B }

N.P.A = { C, D }

BCNF: ✓ × × ×
3rd: ✓ × × × FD is in 2nd NF.
2nd: ✓ ✓ ✓ ✓

Page No.
Date: YOUVA

Ques-7. Identify Highest Normal form?

Ans-7.

R(A B C D)

FD: { A → BC D, BC → A D, D → B }

Ck: { A, BC, CD }

PA = { A, B, C, D }

N.P.A = { ∅ }

BCNF: ✓ ✓ X

3rd: ✓ ✓ ✓ It is in 3rd Normal F.

Ques8: Let R(A, B, C, D, E, F) with Functional dependencies FD: { A → B, B → C, C → D, D → A } decompose into R_1(ABC) & R_2(CDE) check dependencies preserved or Not?

Ans 8:

R_1(ABC)	R_2(CDE)
A^+ = ABCD A → BC	C^+ = ABCD C → BD
B^+ = BCDA B → CDA	D^+ = ABCD D → BC
C^+ = DABC C → AB	E^+ = EABCD ∅
AB^+ = ABCD AB → C	CD^+ = CDAB
BC^+ = BCDA BC → A	DE^+ = DEABC DE → C
CA^+ = CADB CA → B	EC^+ = ECDAB EC → D
F_1: A → BC, B → CA, C → AB	F_2: C → D, D → B

Since, all Functional dependency can made from F_1 & F_2 ∴, dependency preserved.

Ques9: Check for dependency preserving decomposition?

Ans-9:

$R(ABCDE)$

$FD: \{A \to BCD,\ B \to AE,\ BC \to AED,\ D \to E,\ C \to DE\}$

$R_1(A,B)$	$R_2(B,C)$	$R_3(C,D,E)$
$A^+ = ABCDE$ $A \to B$	$B^+ = ABCDE$ $B \to C$	$C^+ = CDE$ $C \to DE$
$B^+ = ABCDE$ $B \to A$	$C^+ = CDE$	$D^+ = DE$ $D \to E$
		$E^+ = E$
		$CD^+ = CDE$ $CD \to E$
$F_1: \{A \to B, B \to A\}$	$F_2: \{B \to C\}$	$DE^+ = DE$
		$EC^+ = ECD$ $EC \to D$
		$F_3: \{C \to DE, D \to E\}$

$F_1 \cup F_2 \cup F_3 = \{A \to B, B \to A, B \to C, C \to DE, D \to E\}$

$A^+ = ABCDE$ (True)

$B^+ = BACDE$ (True)

$BC^+ = BCADE$ (True)

$D = DE$ (True)

$C = CDE$ (True)

Since, G covers F & F covers G

∴, Functional dependency Exist.

Ques10: Check For lossless decomposition or Lossy decomposition?

Ans10:

$R(A, B, C, D, E)$

1	1	2	1	3
2	2	2	1	3
3	1	6	3	6
4	2	8	5	7
5	3	9	5	7

① $R_1(ABC)$ $R_2(DE)$

$R_1 \cup R_2 = ABCDE$

But $R_1 \cap R_2 = \phi$ ∴ Lossy

② $D(ABC)$, $C(CD)$

$R_1 \cap R_2 = C$ But $R_1 \cup R_2 = ABCD$ ∴, Lossy

③ $R_1(ABC)$, $R_2(CDE)$

$R_1 \cup R_2 = ABCDE$

$R_1 \cap R_2 = C$

and insertion Must be candidate key for either R_1 or R_2.

i.e, from C we can drive AB.

$C \to ABX$ $C \to DE$

$C \to A$ ✗ $C \to D$ ✓

$C \to B$ ✗ $C \to E$ ✓

⇒ C is candidate key for R_2

∴, Lossless decomposition.

④ $R_1(ABC)$, $R_2(ABDE)$

$R_1 \cup R_2 = ABCDE$

$R_1 \cap R_2 = AB$

i.e, $AB \to C$ ✓ $AB \to DE$ ✓

$AB \to D$ ✓

$AB \to E$ ✓

⇒ AB is Candidate key for R_2

∴, Lossless decomposition.

⑤ $R_1(AB)$, $R_2(BCDE)$

$R_1 \cup R_2 = ABCDE$

$R_1 \cap R_2 = B$

i.e, $B \to A$ ✗ $B \to CDE$ ✗

$B \to C$ ✗

$B \to D$ ✗

$B \to E$ ✗

⇒ B is non-Candidate key.

∴, Lossy decomposition.

(B) $R_1(AB)$, $R_2(CD)$, $R_3(DE)$

$R_2 \cap R_3 = D$

$D \to C$ ✗ $\quad D \to E$ ✓

$\Rightarrow D$ is Candidate key for R_3.

But $R_1 \cap R_3 = \phi$

$\Rightarrow \therefore$ Lossy decomposition

Ques II: $R(ABCD)$: check for Lossy or Lossless decomposition?
FD: $\{A \to B, B \to C, C \to A\}$
$R_1(ABC)$ $\quad R_2(CD)$

Ans II:
Step 0: $R_1 \cap R_2 = C$
Step 1: check for Candidate key (inner intersection part)
Step 2: $C^+ = CAB$. [C.k if Exist then it must be Lossless]
Since, C is Candidate $\therefore$ Lossless decomposition.

- **Canonical Cover / Minimal Cover / Irreducable Set of FDs.**

For functional Dependency $f \cdot f'$ is called Canonical cover if f' don't have
(1) Extraneous attribute / redundant attribute.
(2) redundant FD.

Steps for Converting in Canonical form

Step 1: Splitting rule so that in every FD right hand side has single attribute.
Step 2: Remove extraneous attribute.
Step 3: Remove Redundent FD.

Ques 1: Check for Canonical Cover?

Ans 1: FD: {AB → C, C → AB, B → C, ABC → AC, A → C, AC → B}

Step 1:

MFD: {AB → C, C → A, C → B, B → C, ABC → A, ABC → C, A → C, AC → B}

Step 2:

MFD: {B → C, C → A, C → B, B → C, only Non-Trivial D. A → C, A → B}

$C^+ = BC$ $C^+ = ACB$ $B^+ = B$ (I R.C)

$A^+ = ABC$ $A^+ = CA$

↓

Redundant cover

Step 3: {C → A, B → C, A → B}

Ques 2: Check for Lossless or Lossy decomposition?

R(A B C D E F)

F: {AB → C, C → D, D → EF, F → A, D → B}

D: { R_1(ABC) R_2(CDE) R_3(EF)}

Ans 2:

$R_1 \cup R_2$ = ABCDE

$R_1 \cap R_2$ = C $C^+ = CDEFAB$

Since, C is Candidate key for Both R_2 & R_2.

∴, $R_1 \bowtie R_2$ is lossless decomposition.

R_1(ABC) R_2(CDE)

R_{12}(ABCDE) R_3(EF)

$R_{12} \cup R_3$ = ABCDEF

$R_{12} \cap R_3$ = E $E^+ = E$

Since, E is non-Candidate key for Both R_{12} & R_3.

∴, Overall Functional Dependency is lossy decomposition?

- Converting a relation from 1st NF to 2nd NF:

Ques 1: $R(A, B, C, D)$
$F = \{A \to B, B \to C\}$

Ans 1:
Sk: $ABCD^+ = \{ABCD\}$
Sk: $AD^+ = \{ABCD\}$ $A^+ = \{A, B\}$ [N.S.K]
$D^+ = \{D\}$ [N.S.K]

⇒ Ck: $\{AD\}$
P.A = $\{A, D\}$
N.P.A = $\{B, C\}$
1st N.F: ✓ ✓
2nd N.F: ✗ ✓
$A \to B$

$A^+ = ABC$ {Decomposition of Table}

$R_1: \{A, B, C\}$ $R2: \{A, D\}$

$A^+: \{ABC\}$ $A \to BC$ $A^+: \{ABC\}$
$B^+: \{BC\}$ $B \to C$ $D^+: \{D\}$
$C^+: \{C\}$
$AB^+: \{ABC\}$ $AB \to C$ $F_2 = \{\phi\}$ ∴, it is BCNF.
$BC^+: \{BC\}$
$CA^+: \{ACB\}$ $AC \to B$

$f_1 = \{A \to BC, B \to C\}$
C.K = A
It is in 2nd Normal form.

Qus 2: $R(A, B, C, D)$
$FD: \{A \to B, C \to D\}$

Ans 2:

Sk: $ABCD^+ = \{ABCD\}$
Sk: $AC^+ = \{ABCD\}$ $A^+ = \{AB\}$ [N.S.K]
$D^+ = \{D\}$ [N.S.K]

Ck: $\{AC\}$
P.A = $\{A, C\}$
N.P.A = $\{B, D\}$

1st: ✓ ✓
2nd: X X

$A^+ = \{A, B\}$	$C^+ = \{C, D\}$	$R_3 = \{A, C\}$
$R_1 = \{A, B\}$	$R_2 = \{C, D\}$	
Sk: A	Sk: C	
$A^+ = \{AB\}$ $A \to B$	$C^+ = \{CD\}$ $C \to D$	$A^+ = \{AB\}$
$B^+ = \{B\}$	$D^+ = \{D\}$	$C^+ = \{C, D\}$
$F_1 = \{A \to B\}$	$F_2 = \{C \to D\}$	$F_3 = \{\}$

Since, F_1, F_2, and F_3 contain L.H.S as Superkey
∴, given relation is in BCNF.

- Converting a relation from 2nd NF to 3rd NF:

Qus3: $R(A, B, C, D)$
$FD: \{A \to B, B \to C, C \to D\}$

Ans 3:
Sk: $ABCD^+ = \{ABCD\}$
Sk: $A = \{ABCD\}$

$Ck = \{A\}$
$P.A = \{A\}$
$N.P.A = \{B, C, D\}$

1st :	✓	✓	✓
2nd :	✓	✓	✓
3rd :	✓	X	X

$A^+ : ABC$
$R_1 : \{AB\}$
Sk: A
$A^+ : \{ABCD\}$ $A \to B$
$B^+ : \{BCD\}$

$B \to C$
$B^+ : BCD$
$R_2 : \{B, C, D\}$
Sk: B
$B^+ : \{BCD\}$ $B \to CD$
$C^+ : \{CD\}$ $C \to D$
$D^+ : \{D\}$
$BC^+ : \{BCAB\}$ $BC \to D$
$CD^+ : \{CD\}$
$BD^+ : \{BCD\}$ $BD \to C$

$C \to D$
$C^+ = CD$
$R_3 : \{C, D\}$
Sk: C
$C^+ = \{CD\}$ $C \to D$
$D^+ = \{D\}$

$F_1 = \{A \to B\}$
Since F_1 LHS Contain Sk ∴, BCNF.

$F_2 = \{B \to CD, C \to D\}$

$F_3 = \{C \to D\}$
BCNF.

For $F_2 = \{B \to CD, C \to D\}$
Sk: $BCD^+ = \{BCD\}$
Sk: $B^+ = \{BCD\}$
$Ck = \{B\}$ $P.A = \{B\}$ $N.P.A = \{C, D\}$
Since, $C \to D$, $N.P.A \to N.P.A$ ∴, R_2 Not in 3rd NF

$R_2 : \{BCD\}$
$FD : \{B \to CD, C \to D\}$

1st : ✓ ✓
2nd : ✓ ✓
3rd : ✓ ✗

$C \to D$

$C^+ = \{CD\}$

$R_{11} = \{CD\}$
Sk : C

$C^+ = \{CD\}$ $C \to D$
$D^+ = \{D\}$

$F_{11} = \{C \to D\}$

$R_{12} = \{CB\}$
Sk : B

$C^+ = \{CD\}$ ℓ.
$B^+ = \{BCD\}$ $B \to C$

$F_{12} = \{B \to C\}$

Since, F_{11} & F_{12} contain L.H.S as Super key
∴, F_{11} & F_{12} are in BCNF.

$R_{12}\{(B,C)\}$, $R_3\{(C,D)\}$, $R_1\{(A,B)\}$

Also, Thise decomposition is Lossless decompusition Since, all $F_1 \cup F_2 \cup F_3 \cup F_{11} \cup F_{12}$ makes F itself directly or may be indirectly.

* Relational Algebra

Relational Algebra is a procedural query language which takes an relation as an input and generates a relation as an output.

- **Basic Operators**

① σ (Selection): Select rows based on given condition.

Rollno	Name	Age
1	A	20
2	B	21
3	A	19

Query: Retrive the name of student whose Rollno = 2?
Ans:

π ('name') (σ ('Roll No = 2') (Student)) → (2, B, 21)

Projection works on columns.
Selection always works on rows (tuples).

② π (projection): Projects / select some columns.

Query: Retrive the rollno. from stable (Student)?
Ans:

$\pi_{\text{Roll no.}}$ (Student) → 1, 2, 3

Query: Retrive roll, name from student?
Ans: $\pi_{\text{name, rollno}}$ (Student) π_{name} (Student)
O/p → A, B

③ X (Cross product): Cross product of relations returns, $m*n$ rows where m and n are number of rows in R_1 and R_2 respectively. colums will be $(m+n)$.

Eg:

R_1

A	B	C
1	2	3
2	1	4

R_2

C	D	E
3	4	5
2	1	2

$R_1 \times R_2$:

A	B	C	C	D	E
1	2	3	3	4	5
1	2	3	2	1	2
2	1	4	3	4	5
2	1	4	2	1	2

At Least one (Colwise) Column should be common in Cross product.

④ - (Set - Difference): $R_1 - R_2$ Returns those tuples which are in R_1 but not in R_2.

Max no. of Rows = m

Min no. of rows = $m - n$

Eg: Find Name of student But not Employee?

(Student)

RollNo.	Name
1	A
2	B
3	C

(Employee)

EmpNo	Name
7	E
1	A

$(\lambda_{Name}(St) - \lambda_{Name}(Employee))$

⇓

Name
B
C

Student - Employee =

R..	Name
2	B
3	C

(5) ∪ (union): Returns Those tuples which are either in R_1 & R_2.

(1) No. of Colums must be same in no.
(2) Domain of Every Column should be Same.
(3) Column Name of Union table is always left side of Table.
(4) Max no. of Rows = m + n
min no. of Rows = max(m, n).

Ques: Find Name of student and who is student as well Employee?

Ans:

Rollno	Name
1	A
2	B
3	C

(Student)

Emp No	Name
7	E
1	A

(Employee)

$A \cup B$ = (Student) ∪ (Employee)

Roll No	Name
1	A
2	B
3	C
7	E

Π_{Name} (Student) ∪ Π_{Name} (Employee)

Name
A
B
C
E

⑥ Division Method (1): Division Operator A/B will return {Derived Operator} those tuples in A which are associated with Every tuples in B.

Note:- Attribute of B should be proper subset of A
The attribute of A/B will Attribute of A - Attribute of B

Qus: Retrive Sid of students who enrolled in every/all courses.

Ans: Division Method is used in case of every/all.

Sid	Cid
S_1	C_1
S_2	C_{21}
S_1	C_2
S_3	C_2

Enrolled 'E'

Cid
C_1
C_2

Course 'C'

π_{Sid} (Enrolled) X π_{Cid} (Course)

Sid	Cid
S_1	C_1
S_1	C_2
S_2	C_1
S_2	C_2
S_3	C_1
S_3	C_2

Sid
S_1
S_2
S_3

X

Cid
C_1
C_2

$\pi_{Sid}(Enrolled) \times \pi_{Cid}(Course) - \pi_{Sid,Cid}(Enrolled)$

Sid	Cid
S_1	C_1
S_1	C_2
S_2	C_1
S_2	C_2
S_3	C_1
S_3	C_2

$-$

Sid	Cid
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

$=$

SId	Cd
S_2	C_2
S_3	C_1

$\pi_{Sid}(Enrolled) - \pi_{Sid}\left((\pi_{Sid}(Enrolled) \times \pi_{Cid}(Courses)) - (\pi_{Sid,Cid}(E))\right)$

Sid
S_1
S_2
S_3

$-$

Sid
S_2
S_3

$=$

Sid
S_1

(7) ρ (Rename): Renaming a Relation to another relation.

- Joins: A join is an SQL / Relational Algebra operation performed to establish a connection between two or more database table based on matching columns, thereby creating a relationship between the tables.

  Joins = Cross product + Some condition.

(a) Natural join: In Natural join, equality conditions on common attributes hold and duplicate attributes are removed by default.

Note:- Natural join $\overset{\sim}{=}$ Crossproduct if two relation have no common attributes.

also, attributes (P.k) having different values are removed by default.

Page No.:
Date:
YOUVA

Ques: Find the Emp Names who is working in a Department

Ans:

Emp

E-No	EName	Address
1	Ram	Delhi
2	Varun	chd.
3	Ravi	chd.
4	Amrit	Delhi

Dept

DepNo	Name	Eno
D_1	HR	1
D_2	IT	2
D_3	MRKT	4

Select E_Name from Emp, Dept where, Emp.Eno = Dept.Eno

E-No	E-Name	Address	DepNo	Name	Eno.
1	Ram	Delhi	D_1	HR	1
1	Ram	Delhi	D_2	IT	2
1	Ram	Delhi	D_3	MRKT	4
2	Varun	chd.	D_1	HR	1
2	Varun	chd.	D_2	IT	2
2	Varun	chd.	D_3	MRKT	4
3	Ravi	chd.	D_1	HR	1
3	Ravi	chd.	D_2	IT	2
3	Ravi	chd.	D_3	MRKT	4
4	amit	Delhi	D_1	HR	1
4	Amrit	Delhi	D_2	IT	2
4	Amrit	Delhi	D_3	MRKT	4

Emp. Dept.

E-No	E-Name	DepNo	Eno
1	Ram	D_1	1
2	Varun	D_2	2
4	Amrit	D_3	4

By Default, rows are removed whose Eno. is not Same

(b) Self Join: A self join in which a table is joined with itself (which is also called unary relationship) especially when a table has a Foreign key which refrence its own primary key.

Qus: find student id who is enrolled in at least two courses?

Ans:

Study

S_id	C_id	Sine
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

Select T_1.sid from Study as T_1, Study as T_2
where T_1.sid = T_2.sid ---- and
T_1.cid <> T_2.cid ----
Not Equal to = <> ----

S_id	C_id	S_id	C_id
S_1	C_1	S_1	C_1
S_1	C_1	S_2	C_2
S_1	C_1	S_1	C_2
S_2	C_2	S_1	C_1
S_2	C_2	S_2	C_2
S_2	C_2	S_1	C_2
S_1	C_2	S_1	C_1
S_1	C_2	S_2	C_2
S_1	C_2	S_1	C_2

⇒

S_id	C_id	S_id	Cid
S_1	C_1	S_1	C_2
S_1	C_2	S_1	C_1

⇒ S_1

Page No.
Date: YOUVA

① Equi Join: It is a special case of conditional join when only equality conditions are applied between attributes.

Qus: Find the Emp name who worked in a Department having location same as their address?

Ans:

Emp

E-No	E-Name	Address
1	Ram	Delhi
2	Varun	chd.
3	Ravi	chd.
4	Amit	Delhi

Dept

Dept No.	Location	E-no
D_1	Delhi	1
D_2	Pune	2
D_3	Patna	4

Select E-Name from Emp, Dept where.
Equi join Condition → Emp. E-no = Dept. E-no and Emp. Address = Dept. location;

The difference between Natural join and Equi join that In case of Equijoin we compare any two attributes. (with minimum requirement Equal to operation between any two (common attribute.)

E-No	E-Name	Address	DeptNo.	Location	E-no
1	Ram	Delhi	D_1	Delhi	1
2	Varun	chd.	D_1	Delhi	1
3	Ravi	chd.	D_1	Delhi	1
4	Amit	Delhi	D_1	Delhi	1

⇒

E-Name
Ram

(d) $\bowtie_c$ (Conditional Join): Selection from two or more tables based on some condition.
(Cross product followed by selection).

(e) ⟕ (Left outer Join): It Gives result on the basis of matching (Natural Join) rows which are in left table but not in right Table.

(Left outer Join) = Natural Join + Some extra attributes.

(A)

Ques:

Emp

Emp-No	Emp-Name	Dept-No
E_1	Varun	D_1
E_2	Amrit	D_2
E_3	Ravi	D_1
E_4	Nitin	-

Dept

Dept-No	D-name	LOC
D_1	IT	Delhi
D_2	HR	Hyd.
D_3	Finance	Pune

Select Emp-no, e-name, d-name, LOC
from Emp Left Outer Join Dept
on (Emp. Dept-No = Dept. Dept-No);

Emp-No	E-Name	D-name	LOC
E_1	Varun	IT	Delhi
E_2	Amrit	HR	Hyd.
E_3	Ravi	IT	Delhi
E_4	Nitin	-	-

Consider two Relation R_1 & R_2.
When we apply Join on two relation R_1 & R_2, Some tuples of R_1 do not appear in the result set which does not satisfy the join condition. The attributes of R_1 which do not satify the join condition will have values as Null for attribute of R_2.

(f) ⟖ (Right outer Join):
It gives the matching rows and which are in right table but not in Left table.

Right outer join gives all tuples of S (R_2) in the result set. The tuples of R_2 which does not satisfy the join condition will have values as Null for attribute of R_1.

Ans:

Emp

Emp-No	E-Name	Dept-No
E_1	Varun	D_1
E_2	Amrit	D_2
E_3	Ravi	D_3

Dept

Dept-No	D-Name	LOC
D_1	IT	Delhi
D_2	HR	Hyd.
D_3	Finance	Pune
D_4	Testing	Noida

Select Emp-No, E-Name, D-Name, LOC
from Emp Right Outer Join Dept
on (Emp.Dept_No = Dept.Dept_no);

Emp-No	E-Name	D-Name (Dept-No)	LOC
E_1	Varun	D_1	Delhi
E_2	Amrit	D_2	Hyd.
E_3	Ravi	D_3	Pune
-	-	D_4	Noida

(3) FOJ (Full outer Join): When applying join on two relation R_1 and R_2, some tuples of R_1 or R_2 do not appear in result set which does not satisfy the join condition.
But full order outer join gives all the values of R_1 and all attributes of R_2 in result set.
The tuples of R_2 which do not satisfy the join condition will values Null values for attributes of R_1 & vice-versa.
Full outer Join = LOJ ∪ ROJ

• Introduction to Structured Query Language

• **SQL:** SQL is Standard Language for storing and manipulating and retrieving data in databases.

• **DDL:** DDL is short name of Data definition language, which deals with databases schemas descriptions of how data should reside in the database.

1) **CREATE:** To create the databases and its objects like (table, index, views, store procedure, functions and triggers).

Syntax:→

```
CREATE TABLE <TABLE NAME>
(
    Colum1  datatype;
    Colum 2 datatype;
    Colum3  datatype;
      :
      :
);
```

2) **ALTER:** ALTER/ changes the structure of existing database.

Syntax:→

```
ALTER TABLE <TABLE NAME>
ADD Column_name datatype;

ALTER TABLE <TABLE NAME>
DROP Column_name;

ALTER TABLE <TABLE NAME>
MODIFY <column_name> varchar(<new_len>);
```

ALTER TABLE <TABLE NAME>
RENAME <old_Column_NAME> to <new_column_NAME);

ALTER TABLE <TABLE NAME>
ADD PRIMARY KEY (column_NAME);

3) DROP: delete columns from database.

4) TRUNCATE: Remove all records from a table, including all spaces allocated for the record are removed.

5) RENAME: Rename an column Name.

- DML: DML is short name of Data manipulation language which deals with data manipulation and include most common SQL commands such SELECT, INSERT, UPDATE, DELETE etc. and it is used store, modify, retrieve, delete and update data in database.

1) SELECT: retrieve data from a database.
2) INSERT: Insert data into Single table.
3) UPDATE: Updates Existing data within a table.
4) DELETE: Delete all records from a databases table.
5) MERGE: UPSERT Operation (insert or update).

Syntax:→ UPDATE table_name
SET Column 1 = value 1,
Colum 2 = value 2,
WHERE Condition;

Delete	Drop	Truncate
1) Delete command used to delete rows.	2) Drop / Delete's all Rows, colums and attribute.	1) Truncate delete all rows in Single run.
2) Roll back possible.		
DELETE FROM <T.N> WHERE <Condition>;	2) Roll back Not possible.	2) R-B Not possible
	DROP <table Name>;	Truncate table <tname>;

- Contraints in SQL:→ Constraints in MYSQL is used to specify the rule that allows or restricts what values/ data will be stored in the table.
They provide the suitable method to ensure data accuracy and integrity inside the table.

1) Column level constraints: These constraints are applied only to the single columns that limits the type of particular column.
2) Table level constraints: These constraints are applied to entire table that limits the type of data of the whole table.

1) Unique: This Constraint ensure that all values inserted into the colun will be Unique.

```
CREATE TABLE ShirtBrands
(
    Id INTEGER,
    BrandName Varchar(40) Unique,
    Size Varchar(30),
);
```

2) Not Null: This constraint specifies that the column cannot have NULL or empty values.

```
CREATE TABLE Student
(
    Id INTEGER,
    LastName TEXT NOT NULL,
    FirstName TEXT NOT NULL,
    City VARCHAR (35);
);
```

3) Primary key: This constraint is used to specify identify each record in a table uniquely. If the columns contains primary key constraints, then it cannot be NULL or empty.

```
CREATE TABLE Persons
(
    Id INT NOT NULL PRIMARY KEY,
    Name varchar (45) Not NULL,
    Age int,
    City Varchar (25),
);
```

4) Check: It controls the value in a particular column, It ensures that the inserted value in the column must be satisfied with the give condition.

```
CREATE TABLE Persons(
    ID INT NOT NULL;
    Name Varchar (15) Not NULL;
    Age int Check (Age >= (18));
```

5) DEFAULT: This constraint is used to set the default value for the particular column where we have not specified any value.

```
CREATE TABLE Persons(
    Id int NOT NULL;
    Name varchar(15) NOT NULL,
    Age int,
    CITY varchar(25) DEFAULT 'kota');
```

6) AUTO_INCREMENT: This Constraint automatically generates a unique number whenever we insert a new record into the table.
Generally we use this Constraint as primary key.

```
CREATE TABLE Animals(
    Id int NOT NULL; Auto_INCREMENT,
    Name char(30) NOT NULL,
    Primary key (id));
```

7) ENUM: The ENUM datatype in MySQL is a string object. It allows us to limit the values chosen from a list of permitted values of the column specification at the time of table creation.

```
CREATE TABLE Shirts(
    Id int Primary key AUTO_INCREMENT,
    name varchar(35);
    size ENUM('small', 'medium', 'large', 'x-large'));
```

8) INDEX: This constraints allows us to create and retrieve values from table very quickly and easily.

```
CREATE TABLE Shirts (
    Id int PRIMARY KEY AUTO_INCREMENT,
    name Varchar(35);
    Size ENUM ('Small', 'Medium', 'large', 'X-large');
```

Step1: The following illustration creates a table named shirts that contain 5 columns, id, name and size.
Step2: We Need to insert values into shirts table.
step3: Now, execute this statement for creating index.

```
CREATE INDEX <idx_name> ON shirts (<col_name>);
```

Step4: We can us the query below to retrieve the data using index column.

```
SELECT * FROM shirts USE ~~index~~ INDEX (idx_name);
 USE
```

(9) FOREGIN KEY: This constraints is used to links two table togethen. It is also known as refrencing key. A foregin key columns matches primary key fields of another table.

```
CREATE TABLE Persons(
    Person_ID int NOT NULL PRIMARY key,
    Name Varchar(45) NOT NULL,
    Age int,
    City varchar(25));
```

```
CREATE TABLE Orders (
    Order_ID int NOT NULL PRIMARY KEY,
    Order_NUM int NOT NULL,
    Person_ID int,
    FOREIGN KEY (Person_ID) REFERENCES Persons (Person_ID));
```

Person_ID	Name	Age	city
1	Stephen	15	Florida
2	Joseph	35	California
3	Peter	40	Alaska

Table orders Contain the following data

Order_ID	Order_Num	Person_ID
1	5544	1
2	3266	2
3	1290	3
4	7890	1

Ques1: Write a SQL Query to display maximum salary from emp table?

Ques2: Write a SQL Query to display Employee name who is taking maximum salary?

E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	40000
5	Varun	IT	50000

Ans1:

```
SELECT MAX (SALARY)
  FROM Emp;
```

Ans2:

```
SELECT E_NAME
  FROM Emp
  WHERE Salary = ( SELECT MAX (Salary)
                   FROM Emp);
```

Ques3: Write a SQL Query to display second highest Salary from Emp table?

Ans3:

```
SELECT MAX (Salary)
  FROM Emp
  WHERE Salary <> (SELECT MAX (Salary)
                   FROM Emp);
```

Ques4: Write a SQL Query to display Employee who is taking second highest salary?

Ans4:

```
SELECT E_NAME
  FROM Emp
  WHERE Salary = (SELECT MAX (Salary)
                  FROM Emp
                  WHERE Salary <> (SELECT MAX (Salary)...
                                   FROM Emp);
                  WHERE
```

Ques5: Write a query to Display all dept names with no. of Employess working in that?

Ans5:

```
SELECT dept, count(*),
  FROM Emp
  GROUP BY dept;
```

→ we only use aggreate function with group by command.

Ques6: Write a query to display all the dept names where no. of Employees are less than 2?

Ans6:

WHERE Caus can't used with Group by we want to use Having. for Conditions.

```
SELECT Dept
  FROM emp
  GROUP BY Dept
  HAVING Count(*) < 2;
```

Ques7: Write a query to display Employee name working in dept where no. of employees are less than 2?

Ans7:

```
SELECT E-NAME
  FROM Emp
  WHERE Dept = ( SELECT Dept
      (In)  FROM Emp
            GROUP BY Dept
            HAVING count(*) < 2);
```

Ques8: Write a query to display highest salary department wise and name of Emp who is taking that Salary?

Ans 8:

```
SELECT E_name
  FROM Emp
  WHERE Salary In ( SELECT MAX(Salary)
                    FROM Emp
                    GROUP BY DEPt);
```

Ques 9: Delhi Details of Emp whose address is either Delhi or chd or Pune?

Ans-9-

Eid	Ename	address
1	Ravi	chd.
2	Varun	Delhi
3	Nitin	Pune
4	Robin	Banglore
5	Ammy	chd.

Eid	Pid	Pname	Location
1	P_1	IOT	Banglore
5	P_2	BIG Data	Delhi
3	P_3	Retail	Mumbai
4	P_4	Android	Hydrabad

```
Select *
  FROM Emp
  WHERE Salary --
  WHERE Address IN ('Delhi', 'chd', 'Pune');
```

Ques-10: Find the name of Emp who are working on a project?

Ans-10: Select Ename

```
        Select Eid
          FROM project);
```

Ans: 10
```
SELECT ENAME
FROM Emp                    Distinct
WHERE Eid In ( SELECT (Eid)
                FROM project);
```

Corelated SubQuery

Qus: 11 Find the detail of Emp who is working on at least one project?

Ans: 11

Exists / Not Exists are worked on corelated query.

```
SELECT * from Emp
WHERE Eid exists ( Select Eid
                   from project
                   WHERE Emp.Eid = project.Eid);
```

In Exists outer Query will run n times.
for that inner Query will also run n times.

- Aggregate functions

① MIN(): Returns smallest value on the selected columns.

Syntax:
```
SELECT MIN (column_name)
FROM table_name
WHERE Condition;
```

② MAX(): Returns Largest value of the selected columns.

Syntax:
```
SELECT MAX (column_name)
FROM table_name
WHERE Condition;
```

③ AVG(): The AVG() function returns the average value of the numeric column.

Syntax:
```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

④ Count(): The Count() function returns the number of rows that matches a specified criterion.

Syntax:
```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

Example:
```
SELECT COUNT(ProductID)
FROM products;
```

⑤ Sum(): The sum() function returns the total sum of a numeric column.

Syntax:
```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

Example:
```
SELECT SUM(Quantity)
FROM OrderDetails.
```

- Corelated Subquery (Synchronized Query)

1) It is a subquery that uses values from outer query.
2) Top to down approach.

Ques1: Find all employess details who work in a department?

Ans1:

SEL Emp

Eid	Name	address
1	A	Delhi
2	B	Pune
5	A	chd.
4	B	Delhi
3	C	Pune
6	D	Mumbai
7	E	Hyd.

Dept

D.id	name	Eid
D_1	HR	1
D_2	IT	2
D_3	MRkT	3
D_4	Testing	4

```
Select *
FROM Employee
WHERE exists ( Select *
               FROM Dept
               WHERE emp.eid = dept.eid);
```

Execute No. (1)

Execute No. 2

Page No.
Date: YOUVA

Ques: Question Related with Nested Subquery & Correlated Subquery & Joins?
Find Detail of all Employee who work in any Dept?

Emp.

E_id	Name
1	A
2	B
3	C
5	D
	E

Dept.

dept_no.	name	E_id
D_1	IT	1
D_2	HR	2
D_3	MRKT.	3

Nested Subquery	Correlated Subquery	Joins
→ Bottom Up	→ Top down approach	→ Cross product + Condition

- Nested

```
SELECT * ~~Emp~~
  FROM Emp
  WHERE E.id IN ( SELECT E_id
                  FROM Dept);
```

Output:

1	A
2	B
3	C

- Correlated Query

```
SELECT *
  FROM Emp
  WHERE Exists (SELECT E_id
                FROM Dept
                WHERE Emp.E_id = ~~Emp~~ Dept.Eid;
```

- Joins query

SELECT */ ~~> also attributes keyword can be used.
FROM Emp, dept
WHERE Emp.Eid = Dept.Eid;

Qus1: Find Nth highest Salary using SQL?

Ans1:

E_1

Id	Salary
1	10000
2	20000
3	20000
4	30000
5	40000
6	50000

Id	Salary	E_2 Round1 (10k)	Round2 (20k)
1	10000	>10k ✗	>20k ✗
2	20000	>10k ✓	>20k ✗
3	20000	>10k ✗	>20k ✗
4	30000	>10k ✓	>20k ✓
5	40000	>10k ✓	>20k ✓
6	50000	>10k ✓	>20k ✓
		4	3

Select Id, Salary
from Emp (E1) → temp / Refrencing Name.
WHERE N-1 = (Select count (Distinct Salary)
FROM Emp E_2
WHERE E_2.Salary > E_1.Salary);

Where N = nth highest Salary.

- Importants keywords:

① SELECT: The select command is used to select data from a database.
Syntax: SELECT Column1, column2
FROM table_name;
Here Colum 1, column are field names of the table you want to select data, for selecting all field use *

2) WHERE: WHERE clause is used to filter records.

Syntax:
```
SELECT column1, colum2....
FROM table.name
WHERE condition;
```

3) AND, OR, NOT: The where clause can be combined with AND, OR, Not Operators.
The AND and OR operator are used to filter records based on more than one condition.

Syntax:
```
SELECT *
  FROM Table_name
  WHERE Condition1 AND Condition2 AND
                                Condition3
```

4) ORDER BY: The order by keyword is used to sort the result set in ascending and descending order.
Order by by default works on ascending order.
To sort the records in descending order, we use DESC keyword.

Syntax:
```
SELECT colum1, colum2
  FROM Table.name
  ~~WHERE~~
  ORDER BY colum1, column2 ASC/DESC;
```

5) INSERT INTO: Insert into is used to insert new records in a table.
Imp. point: Order of values is the same order as the columns in the tables.

Syntax:
```
INSERT INTO table.name (column1, column2.....)
  VALUES (value1, value2, value3.....);
```

6) NULL Value: It is not possible to set for NULL values with comparsion operator such as =, < or <>. We have to use IS NULL and IS NOT NULL Operator.

Syntax:

```
SELECT column_name
FROM table_name
WHERE Column_name is NULL;
```

7) UPDATE: The UPDATE statement is used to modify the existing records in a table.

Syntax:

```
UPDATE table_name
SET Column1 = value1, Colum2 = value2,...
WHERE Condition;
```

Example:

```
UPDATE Customers
SET ContactName = 'Alfred', City = 'Frankfurt'
WHERE CustomerID = 1;
```

8) DELETE: DELETE statement is used to delete the existing record in a table (row).

Syntax:

```
DELETE FROM table_name
WHERE Condition;
DELETE FROM table_name;
```

In 2nd Syntax all rows are deleted.

Example:

```
DELETE FROM Customers
WHERE Name = 'Alfred';
```

⑨ SELECT TOP: SELECT Top clause is used to specify the number of records to return.

Syntax1:

```
SELECT Column_name
FROM table_name
WHERE Condition
LIMIT Number;
```

Syntax2:

```
SELECT column name(s)
FROM table.name
ORDER BY Column-names(s)
FETCH FIRST number ROWS Only;
```

Syntax3:

```
SELECT Column-name(s)
FROM table_name
WHERE ROWNUM <= number;
```

Syntax4:

```
SELECT TOP number column_name(s)
FROM table_name
WHERE Condition;
```

Example1:

```
SELECT TOP 3 *
FROM Customers;
```

Example2:

```
SELECT *
FROM Customers
LIMIT 3;
```

Example3:

```
SELECT *
FROM Customers
FETCH FIRST 3 ROWS Only;
```

10) **LIKE:** The LIKE Operator is used in a WHERE clause to search for a specified pattern in a column.

a) % : represents zero, one, multiple characters.
b) _ : represents one, single characters.

Syntax:

```
SELECT Column1, column2
    FROM table_name
    WHERE Column LIKE pattern;
```

A) WHERE Customer Name LIKE a%
⇒ Find any value that start with 'a'.

B) WHERE Customer Name LIKE %a
⇒ Find any value that end with 'a'.

C) WHERE Customer Name LIKE %or%
⇒ Find any value that have or in any position.

D) WHERE CustomerName LIKE _r%
⇒ Find any value that have r in second pos.

E) WHERE Customer Name LIke a_%
⇒ Find any value's that start with a and rest are atleast 2 characters in length.

F) WHERE CustomerName LIKE a__%
⇒ Finds any value's that start with a and have atleast 3 charachrs in length.

G) WHERE Customer Name LIKE a%o
⇒ Finds any value that start with a and end with o.

11) IN: The IN operator allow you to specify multiple values in Where clause.
IN operator is a shorthand for multiple OR conditions.

Syntax:

```
SELECT column_name(s)
    FROM table_name
    WHERE column_name IN (value1, value2, ...);
```

Syntax2:

```
SELECT column_name(s)
    FROM table_name
    WHERE column_name IN (Sub Query);
```

Example1:

```
SELECT *
    FROM Customers
    WHERE COUNTRY IN ('Germany', 'France');
```

Example2:

```
SELECT *
    FROM Customers
    WHERE COUNTRY IN (SELECT Country FROM
                      Suppliers);
```

12) BETWEEN: The BETWEEN Operator selects the value within a given range. The values can be number, text or date.

The BETWEEN Operator is inclusive i.e, Begin and end values are included.

Syntax:

```
SELECT column_name(s)
    FROM table_name
    WHERE Column_name BETWEEN value1 AND
                              value2;
```

Example1:

```
SELECT * FROM products
    WHERE price BETWEEN 10 AND 20;
```

13) Joins: A join clause is used to combine rows from two or more tables, based on a related column between them.

① INNER JOIN: SELECT ROWS that have values FROM both tables.

Syntax:

```
SELECT Column.name (s)
    FROM TABLE1 ~~INNER Join table2~~
    INNER JOIN table2
    ON table1.column.name = table2.name;
```

Example:

```
SELECT *
  FROM Orders
  INNER JOIN Customers
  ON orders.Cust_Id = Customers.Cust_Id;
```

② LEFT (OUTER) JOIN: All Records from table1 (left Table) and matching record from right table.
The result is 0 from the right side if there is no match.

Syntax:

```
SELECT Column_name(s)
    FROM TABLE1
    LEFT JOIN TABLE2
    ON table1.column_name = table2.column.name;
```

Example:

```
SELECT Customers.CustName, Orders.OrderId
    FROM Customers
    LEFT JOIN Orders
    ON Customers.CustId = Orders.CustId
```

③ RIGHT (OUTER) JOIN: All records from tabe2 and matching records from left table. The result is 0 from Left side if there is no match.

Syntax:

```
SELECT column_name(s)
    FROM table1
    RIGHT JOIN table2
    ON table1.column name = table2.column_name;
```

④ FULL (OUTER) JOIN: The full order join keyword returns all records when there is a match in left (table1) or right (table2) table records.

Syntax:

```
SELECT Column_name(s)
    FROM table1
    FULL OUTER JOIN table2
    ON table1.name = table2.name
    WHERE Condition;
```

14) UNION: The union Operator is used to combine the result-set of two or more SELECT statements.

1) Every Select statement with union must have same number of columns.
2) The columns must also have similar datatypes.
3) The columns in every select statement must also be in same order.

The union operator selects only distinct values by default. To allow Duplicate value use Union all.

Syntax:

```
SELECT column_name
    FROM table1
    UNION
SELECT column_name
    FROM table2;
```

15) GROUP BY: The group by statements groups rows that have same values into summary rows (like find the no. of customers in each country. The GroupBy statement is often used with aggreate functions.

Syntax:

```
SELECT Column_name(s)
  FROM Table_name
  WHERE Condition
  GROUP BY Column_names(s)
  ORDER BY Column_names(s);
```

Example:

```
SELECT COUNT (Customer Id), Country
  FROM Customers
  GROUP BY Country
  ORDER BY COUNT(Customer ID) DESC;
```

16) HAVING: The having clause was added to SQL because the WHERE cannot be used used with aggregate functions.

WHERE have more priority than having.

Syntax:

```
SELECT Column_name(s)
  FROM Table_name
  WHERE Condition
  GROUP BY Column_name(s)
  HAVING Condition
  ORDER BY column_name(s);
```

Example:

```
SELECT COUNT (customerID), Country
  FROM Customers
  GROUP BY Country
  HAVING COUNT (customerID) > 5;
```

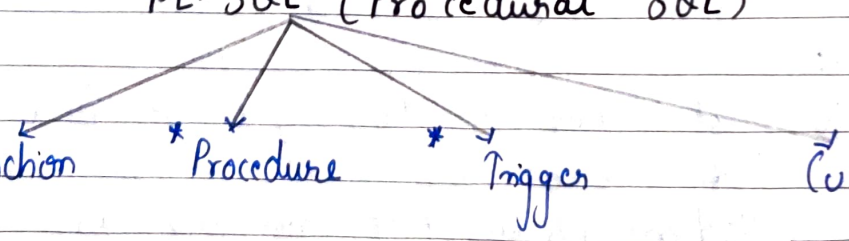
Page No.
Date: YOUVA

Ques1: You need to display last name of employees who have A as second character in there name?
a) select last_name from emp where last_name like _A%
b) Select last_name from emp where last_name = *A%
c) Select last_name from emp where last_name like %A%
d) Select last_name from emp where last_name like *A

Ques2: A command to remove relation from SQL?
a) Delete table <tablename>
b) Drop table <tablename>
c) Erase table <tablename>
d) ALTER table <tablename>

Ques3: In the following, schema R is R(a,b)
Q1: Select * from R
Q2: (Select * from R) union (Select * from R)
Q3: Select distinct * from R
A) Q_1 Q_2 Q_3 produce same result.
B) only Q_1, Q_2 produce same result.
C) Only Q_2, Q_3 produce same result.
d) Q_1, Q_2, Q_3 produce different result.

PL-SQL (Procedural SQL)



PL-SQL defines by What to do
How to do
SQL defines by What to do.

declaration	a, int
Executable code	begin end
Exception Handling	(Errors)