1. Consider the sailor - Boats - Reserve DB described as s(sid, sname, rating, age), b(bid, bname, color), r(Bsid, bid, date) Write each of the following queries in SQL and Relational algebra.

a) Find the colours of boats reserved by Alber.

```
SELECT DISTINCT B.color
From Sailors S
Join Reserves R on s.sid = R.sid
Join Boats B on R.bid = B.bid
Where S.sname = 'Alber';
```

π - color (σ - sname = 'Alber' (S)
$\bowtie$ S.sid = Rsid
$\bowtie$ R.bid = B.bid)

b. Find all sailor ids of sailors who have a rating of at least 8 or reserved boat 103.

```
SELECT DISTINCT sid
From sailors
Where rating >= 8
union
select Distinct sid
From Reserves
where bid = 103;
```

π sid (σ - rating >= 8 (S))
$\cup$
π_2 sid (σ - bid = 103(R))

c. Find the names of sailor who have not reserved a boat whose name contains the string "storm" Order the name in ascending order.

```
select distinct S.sname
from sailors S
where S.sid Not in (
    (select R.sid
     from Reserves R
     Join Boats B on R.bid = B.bid
     where B.bname like '%storm%'
    )
Order by S.sname ASC;
```

StormBoats $\leftarrow$ σ - bname like '%storm%' (B)
StormRes $\leftarrow$ StormBoats $\bowtie$ B.bid = R.bid R
StormSailors $\leftarrow$ π - sid (stormRes)

AllSailors $\leftarrow$ π - sid, sname(S)
NonstormSailors $\leftarrow$ AllSailors - (stormSailors $\bowtie$ sid)
Result $\leftarrow$ π - sname (NonstormSailors)

d. Find the sailors ids of sailors with age over 20 who have not reserved a boat whose name include the string "thrender".

```
Select distinct S.sid
from Sailor S
where S.age > 20
and S.sid not in (
        select R.sid
        from Reserve R
        Join Boats B on R.bid = B.bid
        where B.bname like '%thunder%' );
```

Oldersailors ← σ_age >20(S)

ThunderBoat ← σ_bname LIKE '%thunder%' (B)

ThunderRes ← ThunderBoat ⋈ B.bid = R.bid R

ThunderSailors ← π_sid (ThunderRes)

OlderNonThunders ← π_sid (oldersailors) − ThunderSailors.

(2) Define SELECT operation in Relational algebra.

The select operation (denoted by σ (sigma)) is used to select a subset of the tuple from a relation based on a selection condition.

* SE acts as a filter in ~~the~~ database queries, as it keeps only those tuple that satisfy the qualifying condition.

For example:

(a) select the employee tuple whose department number is 4:

$$\sigma_{DNO=4}(EMPLOYEE)$$

(b) select the employee tuple whose salary is greater than ₹ 30,000.

$$\sigma_{salary>30,000}(EMPLOYEE)$$

* The selection condition is a Boolean (conditional) expression specified on the attribute of relation R.

* Tuples that make the condition true are selected and those make the condition false are filtered out.

* select operation produces a relation S that has the same schema as R.

* It is commutative in nature, hence a cascade of select operation may be applied.

* The number of tuple in the result of a SELECT is less than (or equal to) the number of tuples in the input relation R.

(3) Discuss about trigger.

A trigger is a procedure that runs automatically when a certain event occurs in the DBMS. In many cases it is convinient to specify the type of action to be taken when certain events occur and when certain conditions are satisfied.

CREATE TRIGGER statement is used to implement such actions in SQL.

General form:

```
Create Trigger <name>
Before | After |<events>
For each row | For each statement
When (<condition>)
<action>
```

A trigger has three components.

i) Event :- Trigger ~~happens~~ ~~when~~ is activated when an event happens. (Insert, update, Delet)

Whether it is before the event or after the event.

ii) Condition (optional): activate if a mentioned condition is true.

iii) Action: These ~~are~~ is actions performed by Trigger.

Example: If the employee salary increased by more than 10%. Then increment the rank field by 1.

```
Create trigger Empsal
Before update of salary on Employee
for each row
Begin
    If (:newsalary > (:old.salary*1.1)) Then
        :new.rank:= :old.rank+1;
    END IF;
END;
```

where : → is an assignment operator

(11) Define UNION operation in Relational algebra?

Union (U) is a binary operation denoted by U. The result of R U S, is a relation that includes all the tuples that are either in R or in S or in both R and S, and Duplicate tuples are eliminated, providing distinct tuple.

The two operand Relations R and S must be "type compatible"

i.e., they must have same number of attributes or have same or compatible domains

Example:

To retrieve the social security numbers of all employees who either work in department 5 ~~(result)~~ or directly supervise and employee who works in department 5.

DEP5_EMPS ← $\sigma_{DNO=5}$(EMPLOYEE)

RESULT1 ← π_{SSN}(DEP5_EMPS)

RESULT2 (SSN) ← $\pi_{SUPERSSN}$(DEP5_EMPS)

RESULT ← RESULT1 U RESULT2

Result1

ssn
123456789
333445555
666884444
453453453

Result2

SSN
333445555
888665555

Result

SSN
123456789
333445555
666884444
453453453
888665555

* R1(A1, A2, ..., An) and R2(B1, B2, ..., Bn) are type compatible if
  * they have the same number of attributes
  * The domains of corresponding attributes are type compatible
    i.e., dom(Ai) = dom(Bi) for i=1, 2, ..., n

(5.) What is the use of group by clause?

The group by clause groups rows that have the same values in specified columns into summary rows.

Its most often used with aggregate functions, like:

count() - number of items
sum() - total of values
avg() - average
min()/max() - minimum/maximum

Syntax:

```
Select column1, AGG_Func(column2)
from table
Group by column1;
```

* Every column in the select must either be a part of the Groupby, or used in an aggregate function.
* You can use having to filter groups after aggregation.

Example:

```
select department, Avg(salary) as avg-salary
from employees
```

Result:-

two columns of attribute department and avg-salary, where the average of each salary are the tuples under each department.

(6.) List the aggregate functions supported by SQL? // Explain about Aggregate operators in sql with example.

Aggregate functions are used to summarize information from multiple tuples into a single-tuple summary. A number of built-in aggregate functions exist:

count, sum, max, min and avg.

The count functions returns the number of tuples or values as specified in a query.

The functions sum, max, min and avg can be applied to a set or multiset of numeric values and return, respectively the sum, maximum value, minimum value and average (mean) of those values. These functions can be used in the select clause or in a Having clause.

Example:

i> Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary.

```
select sum (salary), max (salary), min (salary),
       avg (salary)
from employee;
```

Ii) Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
select sum (salary), max (salary), min (salary),
       avg (salary)
from (Employee join Department on Dno = Dnumber)
where Dname = 'Research';
```

7. Discuss the basic form of SQL query using group and having clause?

having provides a condition on the summary information regarding the group of tuples associated with each value of the grouping attributes. Only the groups that satisfy the condition are retrieved in the result of the query.

Example:

For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.

```
Select Pnumber, Pname, count (*)
from Project, works_on
where Pnumber = Pno
Group by Pnumber, Pname
Having count (*) > 2;
```

Pnumber and Pname from project table/relation is selected and by the condition Pnumber = Pno (Project) (Works_On)

A new ~~of~~ result and table consisting of Pnumber and pname, which has number of employee count more than 2, is displayed.

8. Define CROSS PRODUCT operation in Relational algebra.

This operation is used to combine tuples from two relations in a combinatorial fashion.

It is denoted by $R(A1, A2, \ldots, An) \times S(B1, B2, \ldots, Bm)$

* Result obtained is a relation Q with degree n+m attributes:

$Q(A1, A2, \ldots, An, B1, B2, \ldots, Bm)$, in that order.

* The resulting relation state has one tuple for each combination of tuples - one from R and one from S.

Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then R X S will have $n_R * n_S$ tuples.

* The two operands do not have to be "type compatible"

* Cross product is not a meaningful opeation. But it can be made meaningful when followed by other opeations.

Example for non-meaningful opeation:-

Female_emps ← $\sigma_{sex='F'}$ (Employee)

Empnames ← $\pi_{Fname, Lname, ssn}$ (Female_emps)

Emp_dependents ← Empnames × Dependent.

⇒ Here Emp_dependents will contain every combination

* To make it meaningful – in the above given situation, to keep only combinations where the Dependent is related to the Employee, we add a select opeation as follows:

Female_emps ← $\sigma_{sex='F'}$ (Employee)

Empnames ← $\pi_{Fname, Lname, ssn}$ (Female_emps)

Emp_dependents ← Empnames × Dependent

Actual_deps ← $\sigma_{ssn=Essn}$ (Emp_dependents)

result ← $\pi_{Fname, Lname, dependent\_name}$ (Actual_deps)

9. Discuss correlated Nested Queries:

Some queries require that existing values in the database be fetched and then used in a comparison condition. Such queries can be conveniently formulated by using nested queries, which are complete select-from-where blocks within the where clause of another query. The other query is called the outer query.

Whenever a condition in where clause of a nested query references some attribute of a relation declared in the outer query, the two queries are said to be correlated.

Example:

```
select E.Fname, E.Lname
From Employee As E
where E.ssn IN (select Essn
    From Dependent as D
    where E.Fname = D.Dependent_name
    And E.sex = D.sex);
```

The nested query is evaluated once for each tuple (or combination of tuples) in the ~~oth~~ outer query. We can think of query in above example as follows: For each EMPLOYEE tuple, evaluate the nested query, which retrieves the Essn values for all Dependent tuples with the same sex and name as that Employee tuple; if the ssn value of the Employee tuple is in the result of the nested query, then select that Employee tuple.

10. What are Views? How are they created give an example? What are the problems associated with view manipulation?

A view in SQL terminology is a single table that is derived from other tables. other tables can be base tables or previously defined views.

* A view does not necessarily exist in physical form. It is considered to be a virtual table, in contrast to base tables, whose tuples are always physically stored in the database.

In SQL, the command to specify a view is Create view. The view is given a (virtual) table name (or view name), a list of attribute names, and a query to specify the contents of the view.

Example:

```
Create view work_on1
as select Fname, Lname, Pname, Hours
From Employee, Project, Work_on
Where Ssn = Essn and Pno = Pnumber.
```

In this case, Work_on1 inherits the names of the view attributes from the defining (base) tables Employee, Project and Work_on.

Problems associated with view manipulation:

* A view is supposed to be always up-to-date; if we modify the tuples in the base tables on which the view is defined, the view must automatically reflect these changes. Hence, the view is not realized or materialized at the time of view definition but rather at the time when we specify a query on the view. It is the responsibility of the DBMS and not the user to make sure that the view is kept up-to-date.

* One strategy, called query modification, involves modifying or transforming the view query into a query on the underlying base tables.

The disadvantage of this approach is that it is inefficient for views defined via complex queries that are time-consuming to execute, especially if multiple queries are going to be applied to the same view within a short period of time.

* If the view is not queried for a certain period of time, the system may then automatically remove the physical table (materialized view) and recompute it from scratch when future queries reference the view.

11. Write and explain a query to find the names of sailor who have reserved a red boat?

SQL Query:

```
Select distinct S.sname
From sailors S
Join Reserves R on S.sid = R.sid
Join Boat B on R.bid = B.bid
Where B.color = 'red';
```

* the sailor table is assigned to an alias S, this table contains details like sid, sname, rating, and age.
* Join Reserves joins the sailor and Reserves table using the sid field. It connects each sailor to their reservations.
* Then the resultant is joind with Boats table using bid. Now we know which sailor reserved which specific boat.
* And then these are filtered to include only reservations for boats that are red.

From the result, we select only the names of the sailors (sname), that even if a sailor has reserved a red boat more than once, their name appears only once.

Example:

S.Sname	B.colour
Alice	red
John	red
John	red

Final result => S.Sname
Alice
John.

12. Explain set operations of ~~Relation~~ Relational Algebra with example.

SQL has directly incorporated some of the set operations from mathematical set theory which are also part of relational algebra.

* set union (union)
* set difference (Except) and
* set Intersection (Intersection.

The relations resulting from these set operations are sets of tuples; that is duplicate tuples are eliminated from the result. These set operations apply only to union-compatible relations, so we must make sure that the two relations on which we apply the operation have the same attributes and that the attributes appear in the same order in both relations.

i) Union:- $\cup$:- Results of $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R & S, and gives distinct elements/details

Ex: $Dep5\_emps \leftarrow \sigma_{dno=5}(Employee)$

$Result1 \leftarrow \pi_{ssn}(Dep5\_emps)$

$Result2(ssN) \leftarrow \pi_{superssn}(Dep5\_emps)$

$Result \leftarrow Result1 \cup Result2$

=> To retrieve the social security number of all employees who either work in department 5 or directly supervise an employee who works in department 5.

ii) Intersection ($\cap$) :- Result of the operation $R \cap S$ is a relation that includes all tuples that are in both R and S.

Example:

π_country, city (Customers)

$\cap$

π_country, city (Branches)

Project the country & city columns from both the Customers and Branches relations. Then returns the common country and city pairs.

iii) Difference (—) :- Result of R-S, is a relation that includes all tuples that are in R but not in S.

Example:-

π_EmployeeID, name (E)

—

π_contractorID, name (C)

Project name and EmployeeId from Employees relation and name and contractorId from contractor relation and returns, EmployeeId, who are not in contractors relation.

13) Explain about the selection, Union, Projection, Rename, division and Cartesian product operations in relational algebra.

i) Selection:

select operation (σ) is used to select a subset of the tuples from a relation based on a selection condition.

Example:

* select the Employee tuples whose department number is 4:

$\sigma_{Dno=4}$ (Employee)

ii) Union ($\cup$) :

The result of $R \cup S$ is a relation that includes all tuples that are either in R or in S or in both R and S

π_EmployeeID, name (F)

$\cup$

π_EmployeeID, name (P)

Employees who are working fulltime and Parttime.

iii) Projection: (π)

<attribute list> is the desired list of attributes from relation R. → The project operation removes any duplicate tuples as the result of the project operation must be a set of tuples.

Example:

π_name, Department (Employee)

Select the Name and Department column from the Employee relation

iv) Rename:- (ρ)

In some case, we may want to rename the attribute of a relation or the relation name or both. In such case, rename (ρ) is used in the queries.

Example:

to rename the EmployeeID to EmpID and Name to EmpName.

ρ (EmpId, EmpName, Department, salary) (Employee)

v) Division: ($\div$)

$R(z) \div S(z)$, where X subset z. let $Y = z - X$ (and hence $z = X \cup Y$); that is let Y be the set of attributes of R that are not attributes of S.

The result of Division is a relation $T(Y)$, that include a tuple t if tuple t_R appear in R with $t_R[Y] = t$, and with

$t_R[X] = t_S$ for every tuple t_S in S.

Ex:

π-sid $(R) \div B$

This gives sid values from R that are associated with every bid of B.

Cartesian product:

This operation is used to combine tuples from two relations in a combinational fashion.

$R(A1, A2, \ldots, An) \times S(B1, B2, \ldots, Bm)$

Example:

Employees x Department.

→ Every pair of one employee with one department.

14. Write and Explain a Query for finding the name of sailor who have reserved a Red or a Green Boat

Query:

```
Select distinct s.sname
from sailor s
Join Reserves r on s.sid = r.sid
Join Boats b ON r.bid = b.bid
where b.color IN ('Red', 'Green');
```

* Sailors table assigning to alias S. Then sailor and Reserve table is joined to find which sailor reserved which boat.

* The resultant is joined with Boat table, to get the details of the reserved boats, the it is filter to only those boat that are either Red or Green.

And then distinct snames are chosen and thus shown as result.

(15) Write and explain a query for finding the colors of Boats reserved by 'Lubber'.

- Query:

```
SELECT Distinct b.color
From Sailors S
Join Reserves r ON s.sid = r.sid
 Join boats b on r.bid = b.bid
 Where s.sname = 'Lubber';
```

* Sailors table is assigned to its alias S.
* Then the sailors table is joined with Reserves table to find out which sailor reserved what boat.
* The above resultant is joined with the Boats table to access details of the boats that were reserved.
* Then it is filtered to only the sailor named 'Lubber'.
* Then return with the distinct colors of all boats that Lubber had reserved.

(16) Discuss Normalization. Define First Normal Form. List Different Normal Forms.

Normalization is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies.

It is a multi step process that puts data into tabular form by removing duplicated data from the relation tables.

It can be considered as a "filtering" or "purification" process to make the design have successively better quality.

First normal form:

It is defined to disallow multivalued attributes, composite attributes, and their combinations.

* It states that the domain of an attribute must include only atomic (simple, invisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute.

for example:

Emp_proj

		Project	
Ssn	Ename	Pnumbu	Hours

This is nested relation, which is not allowed in first normal form. And can be resolved by

Creating seperate relation called project with pnumber and hours, along with project name.

Different
There ~~approach~~ Normal forms are.

i) First normal form → comprik / complex attributes
ii) Second normal form → full functional dependency
iii) Third normal form → transitive dependency
iv) Boyce-Codd Normal form - nontrivial functional dependency
v) ~~multivalued and~~
Fourth Normal form - Multivalued Functional dependency
vi) Fift Normal form - Join Dependencies.

(14) Explain with the help of example of the following relational algebra operation. Join, Difference, Union, Select.

Join:

The sequence of cartesian product followed by select is used quite commonly to identify and select related tuples from two relations.

The operation Join combines this sequence into a single operation.

General form of Join operation on two relations R (A1, A2, A3, ..., An) and S (B1, B2, B3, ..., Bm) is:

$R \bowtie_{\langle join\ condition \rangle} S$

Example:- To get the manager's name from each department, we need to combine each Department tuple with the Employee tuple whose ssn value matches with Mgrssn value in department tuple.

i.e.,

Dept_Mgr ← Department $\bowtie_{Mgrssn=ssn}$ Employee.

18. Discuss the E.R to Relational mapping algorithm with example for each step.

Step 1:- Mapping of Regular Entity type.

* Create a relation (table) for each regular entity.
* Include all simple attributes.
* Choose a primary key (if it's composite, use all parts)

Example: For Employee, include attributes like ssn, Name, Address etc. and make ssn the primary key.

Step 2:- Map weak Entity types

* Creat a relation for each weak entity
* Include all attributes plus the primary key of the owners as a foriegn key.
* Use the owner's key + weak entity's partial key as the composite primary key.

Ex: Dependent includes, Dep_name, Bdate, Essn (foriegn key)
Primary key = {Essn, Dependent_name}.

Step 3:- Map Binary 1:1 Relationships.

* Foriegn key Approach - Add the primary key of one entity to the other as a foriegn key.

* Merged Relation - Merge both entities into one if both have total participation.

* Cross-reference table:- Create a separate relation. (Less common for 1:1)

Example:- Manages → Add Mgr_ssn to department and Mgr_start_date.

Step 6: Map Binary 1:N Relationships.

* Add the primary keys of the "1" side as a foreign key to the "N" side.

* Include any attributes of the relationship

* Example: Works_for → Add Dnumber as Dno to Employee.

Step 5: Map Binary M:N relations.

* Create a new relation, include the primary keys of both entities as foreign keys.

* Include any attributes of the relationship.

* The composite primary key is both foreign keys.

Example: Works_on → has Essn, Pno and Hours.

Step 6:- Map Multivalued Attributes:

* Create a seperate relation for each multivalued attribute.

* Include the attribute and the primary key to the owning entity.

* Composity primary key = (owning entity's PK + multivalued attribute)

Ex: Locations → Creat Dept_locations with Dnumber and Dlocation.

Step 7: Map N-ary Relationship Type.

* For relationships involving more than 2 entities.

→ Create a new relation, include the primary keys of all involved entities as foreign keys.

* Add any relationship attributes.

Example: Supply → includes Sname, Partno, proj-name.

(M) Illustrate Insert, delete, update, alter, grant, revoke and drop commands in SQL.

Insert:- The insert operation provides a list of attribute values for a new tuple t that is to be inserted into a relation R.

This violate any of the four types of constraints:- Domain constraints, key constraints, Entity integrity, Referential integrity.

Ex: Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4>

Result:- This insertion violate the entity integrity constraint (NULL for the primary key ssn), so it is rejected.

→ (Correct insertion)
Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4>

Delete:- Delete operation is used to delete any tuple t from the relation R.

To specify deletion, a condition on the attributes of the relation select the tuple (or tuples) to be deleted.

Example:

Delete the WORKS_ON tuple with Essn = '999887777' and Pno = 10.

This delete exactly one tuple, whereas

Delete the works_on tuple with Essn = '999887777'

will delete all the tuples with the Essn, no matter the value of projects pnumber in it.

iii) update:- (modify) used to change the values of one or more attributes in a tuple (or tuples) of some relation R. It is necessary to specify a condition on the attributes of the relation to select the tuple to be modified.

Example:

Update the salary of the Employee tuple with ssn = '999887777' to 28000.

Here, the tuple of Employee relation, where the attribute salary corresponding to the ssn given, will updated to the value 28000.

iv) Alter:-

The definition of a base table or of other named schema elements can be changed by using the Alter command. For base tables, the possible alter table actions include adding or dropping a column (attribute), changing a column definition, or adding or dropping table constraint.

Ex:-

Alter table company.Employee add column Job Varchar(12)

=> Another attribute Job is added to the relation Employee.

v) grant:-

The grant command is used in SQL to give permission to a user, role or application so that they can do specific actions on database objects like tables, views or stored procedures.

structure:-

Grant <privilege>
on <object>
To <user/role>
[WITH GRANT OPTION];

<privilege> - actions like select, insert, update, delete etc.
<object> - the table or other object the permission apply to
<user/role> - who you're granting permission to

vi) Revoke:-

Revoke command is used in SQL to remove permission that were previously granted to a user, role or group.

Revoke <privilege>
on <object>
From <user/role>

<privilege> - permissions you want to take away
<object> - Table or other object
<user/role> - user or role that currently has the privilege

vii) drop:- drop command can be used to drop named schema elements, such as tables, domains, or constraints.

If a whole schema is no longer needed:-

Drop schema company cascade / Restrict

Cascade - remove all the elements in it too

Restrict - only when the schema is empty.

To drop a table:-

Drop table Dependent cascade.

To drop a constraint:-

If one attribute already has a constraint,

Drop constraint pk_student

20. Explain informal design guidelines for relational schema design.

We have four informal design guidelines for relation schemas.

i) Guideline 1:- Making sure that the semantics of the attributes is clear in the schema.

Informally, each tuple in a relation should represent one entity or relationship instance.

* Attributes of different entities should not be mixed in the same relation.
* If it is required, it should be foreign keys used to refer to other entities.

ii) Guideline 2:- Reducing the redundant information in tuples.

Designing a schema that does not suffer from the insertion, deletion and update anomalies.

Ex:- Emp-proj (Empn#, Projn#, Ename, Pname, No_hours)

i) changing a single name of projectno PJ will result in changing all the employees working on that project

ii) Cannot insert a project unless an employee is assigned to it, and cannot insert an employee unless he/she is assigned to a project.

iii) When one project is deleted, it will result in deletion of all the employees working on that project.

iii) Guideline 3: Reducing the NULL values in tuples.

Relations should be designed such that their tuples will have as few NULL values as possible.

Attributes that are NULL frequently could be placed in seperate relations

NULL values is seen when, attribute not applicable for a tuple. or when the value is unknown or if it is not available.

iv) Guideline 4: Disallowing the possibility of generating spurious tuples.

The relations should be designed to satisfy the lossless join condition. i.e., No spurious tuples should be generated by doing a natural-join of any relations.

That means, meaningful results are to be generated, instead of unwanted, space consuming, redundant values.

21. Discuss the Equijoin and Natural Join with suitable example.

Equijoin:-

The most common use of join involve join conditions with equality comparisons only. Such a join, where the only comparison operator used is '=', is called an equijoin.

Example:

```
Select student.name, student.id, record.class, record.city
From student, record
Join record
on student.city = record.city;
```

Natural join:- *

The two join attributes, or each pair of corresponding join attributes, have the same name in both relations.

To apply a natural joint on the Dnumber attributes of Department and Dept_locations,

Dept_locs ← Department * Dept_locations.

* Only attribute with the same name is Dnumber
* An implicit join condition is created based on this attribute:-

Department.Dnumber = Dept_locations.Dnumber

22. What is Normalization? What are the conditions that are required for a relation to be in 1NF, 2NF. (16)

Second normal form:- (2NF) is based on the concept of full functional dependency.

A functional dependency $X \rightarrow Y$ is a full functional dependency if removal of any attribute A from X means that the dependency does not hold any more; that is, for any attribute $A \in X$, $(X - \{A\})$ does not functionally determine Y.

* the test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key.

Definition:- A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R.

Example:

~~ssn, pnumber → Hours~~ #

Hours is fully dependent of the ssn and Pnumber, as hours of working is assigned particularly for the ssn and the project the employee is working on.

(23) What is meant by functional dependencies?

A functional Dependency is a constraint between two sets of attribute from the database.

Given a relation R, a set of attribute X in R is said to functionally determine another attribute Y, also in R, if and only if each X value is associated with atmost one Y value.

Consider the relation schema Emp-proj:

Emp-proj.

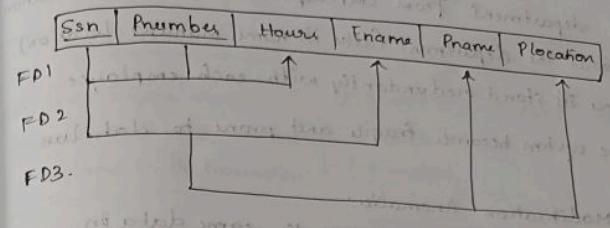
Ssn	Pnumber	Hours	Ename	Pname	Plocation

function dependencies in these are:

* ssn → Ename
* Pnumber → {Pname, Plocation}
* {ssn, Pnumber} → Hours

* The value of an employee's social security number (ssn) uniquely determines the employee name (Ename)
* The value of a project's number (Pnumber) uniquely determines the project name (Pname) and location (Plocation).
* A combination of ssn and Pnumber values uniquely determine the number of hours the employee currently work on the project.

Emp-proj



(24) Explain the types of anomalies with example.

Anomalies are classified into:

(i) Insertion anomalies
(ii) deletion anomalies
(iii) modification anomalies.

(i) Insertion anomalies:-

Difficulty in inserting data because some other data is also required.

Example: In an Emp-Dept (which stores both employee and department info), if we have to add a new department, that doesn't yet have any employee, it is an insertion anomaly because, the primary key-ssn in employee table cannot be NULL, Violating entity integrity.

(ii) Deletion Anomaly:

Losing useful data because of another piece of data was deleted.

Example:- If we delete the last employee working in a department from EMP-dept, we lose all information about the department (like its name, manager, location) since it's stored redundantly with each employee.

* The system becomes fragile and prone to data loss.

iii) Modification Anomalies:

Need to change the same data in multiple places; Failure to do so leads to inconsistencies.

Example: If the manager of a department changes, we have to update that manager's ID in every row of the Emp-dept table for employees in that department. If it fails, we have inconsistent data.

which leads to errors and more time consumption

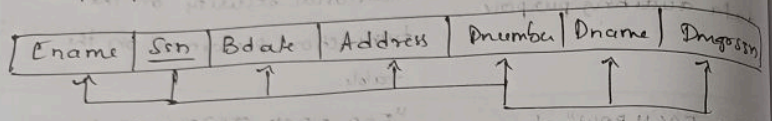
Q5 (a) Define trigger, and explain its three parts. Compare row level and statement level triggers. (3)

Row level	Statement level
Row level trigger executes once for each and every row in the transaction	Statement level triggers execute only once for each single transaction.
Specifically used for data auditing purpose	Used for enforcing all additional security on the transactions performed on the table.
"FOR EACH ROW" clause is present in Create trigger command	"FOR EACH STATEMENT" clause is ommitted in Create trigger command.
Ex: If 1500 rows are to be inserted into a table, the row level trigger would execute 1500 times	Ex: If 1500 rows are to be inserted into a table, the statement level trigger would execute only once.

26. Demonstrate transitive dependency? Give an example.

A functional dependency $X \to Y$ in a relation schema R is a transitive dependency if there exist a set of attribute z that are neither a primary nor a subset of any key of R (candidate key) and both $X \to Z$ and $Y \to Z$ holds.

Emp_dept:-



SSN → Dmgrssn is a transitive functional dependency since ssn → Dnumber and Dnumber → Dmgrssn hold. Dnumber is neither a key itself nor a subset of the key of EMP_Dept.

* ~~Explain schema CPE Cycle tho~~

* SSN → Ename is non-transitive since there is no set of attributes X where SSN → X and X → Ename

27. Explain BCNF. What are the steps to be followed to convert a relation in 3NF to BCNF?

Boyce-Codd Normal Form, was proposed as a simple form of 3NF. Every relation in BCNF is also in 3NF; however, a relation in 3NF is not necessarily in BCNF

⇒ A relation schema R is in BCNF if whenever a nontrivial functional dependency $X \to A$ holds in R, then X is a superkey of R.

Only if $X \to A$ holds in a relation schema R with X not being a superkey and A being a prime attribute will R be in 3NF but not in BCNF

Example:-

Consider a relation

Course (CourseID, Instructor, Room)

~~Here~~, functional dependencies hold:-

CourseId → Instructor

Room → Instructor

Here, courseId is the primary key, so the relation is in 3NF, but not in BCNF

Because Roo → Instructor but Room is not a superkey ~~(primary key)~~ but it determines another attribute (Instructor), which violates BCNF.

To make this a BCNF, we decompose course relation into two relations.

i) Room_Instructor (Room, Instructor) which handles the dependency Room → Instructor.

ii) Course_info : (CourseID, Room) which keeps the link between courseID and Room.

⇒ Room_Instructor

Room	Instructor
101	Alice
102	Bob

Course_Info

CourseID	Room
CS101	101
CS102	102

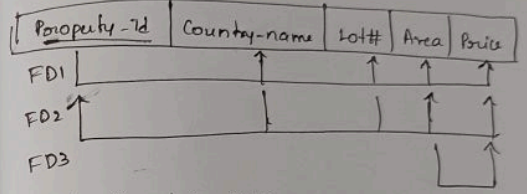
This has * No Redundant data
* No anomalies
* Both relations are in BCNF.

28) What is normalization? What are the conditions that are required for a relation to be in 1NF, 2NF & 3NF. explain with examples (6)

3NF:-

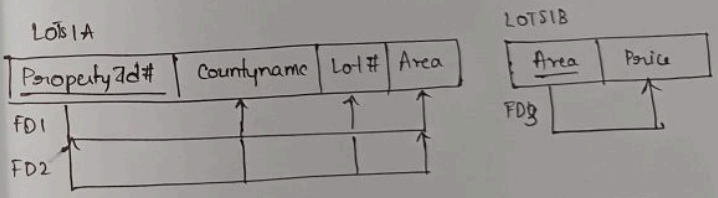
A relation schema R is in third normal form (3NF), if, whenever a nontrivial functional dependency $X \rightarrow A$ holds in R, either (a) X is superkey of R, or (b) A is a prime attribute of R.

consider a relation LOTS1



This is not in 3NF, because Area is not a superkey and Price is not a prime attribute in LOTS1

To normalize it into 3NF, we decompose it into the relations schemas LOTS1A and LOTS1B



* Now both LOTS1A and LOTS1B are in 3NF.