SQL

Prepared by
Chintan R. Chatterjee
Assistant Professor
Computer Engineering Department
Faculty of Technology
Dharmsinh Desai University

Introduction to SQL

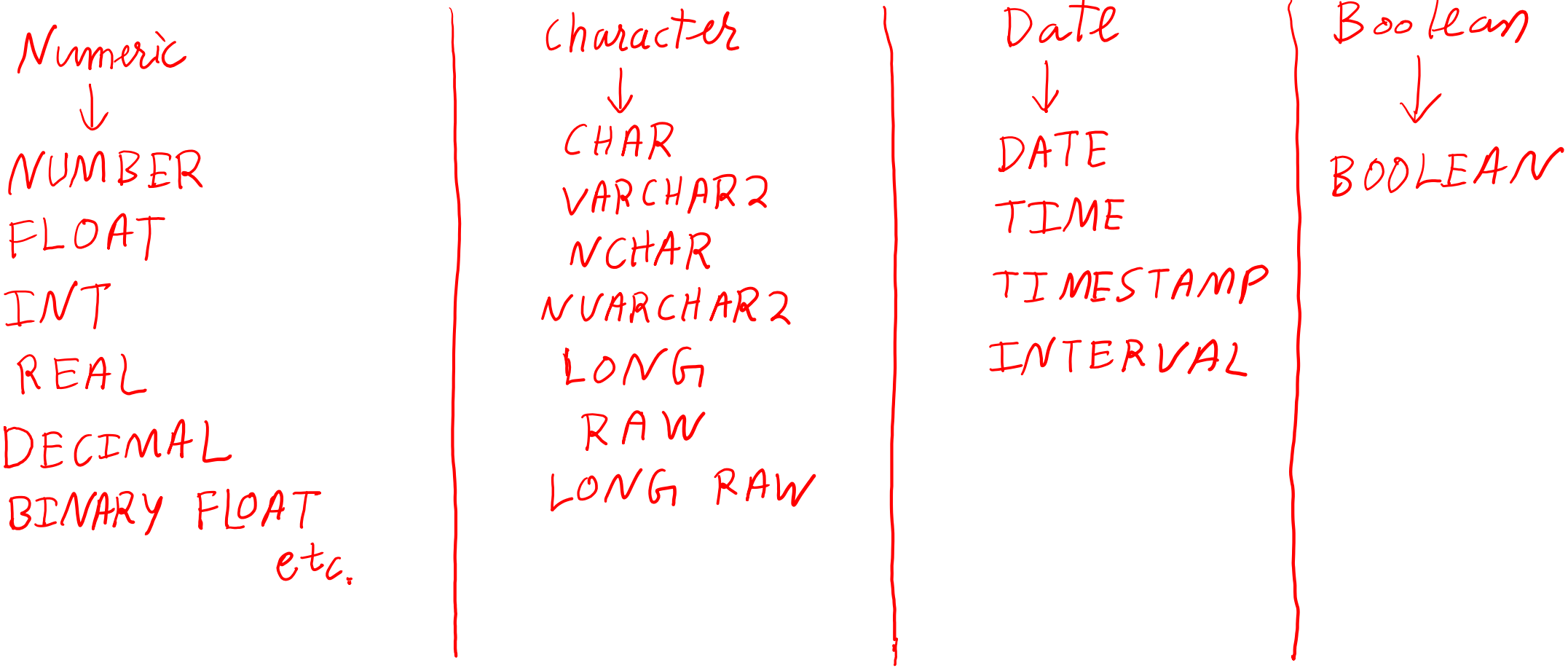
- E.F. Codd: A Relational Model of data for large shared data banks.
- RAYMOND and DONALD: SEQUEL
  - A Structured English QUEry Language
  - It is based on Relational Algebra and Tuple Relational Calculus.
  - First implemented on System ‘R’
  - Oracle, Microsoft SQL Server, MS Access, MySQL, etc.
  - Portable
  - Basics and extensions

Create Table

- CREATE: Used to create a table in database
- Syntax: CREATE TABLE <tablename>(<col1> datatype(width),
  <col2> datatype(width), ...);
- Ex: CREATE TABLE Customer(Name varchar2(30), Cust_id Number,
  Address varchar2(50));

DATA TYPES

- We have numerous data types in SQL.



Constraints in SQL

- We have 7 constraints in SQL

1. NOT NULL
2. UNIQUE
3. Primary Key
4. Foreign Key
5. CHECK
6. DEFAULT
7. REF Constraints

Constraints in SQL

- Constraints can be defined at two levels

a) Column level

b) Table level

➢NOT NULL Constraint should be defined at column level

➢Composite Key should be defined at table level

Constraints in SQL

- Our Requirement is:

Dept: dept_id(P.K.)
dept_name(UNIQUE)
location

Emp: emp_id(P.K.), emp_name(NOT NULL)
job, dept_id(REF dept.id of Dept)
mgr, Salary (>= 5000),
Comm(If not given default 100)
email,
Phone

UNIQUE

```
CREATE TABLE Dept(dept_id NUMBER PRIMARY KEY,
                  dept_name varchar2(30),
                   location varchar2(100),
                   UNIQUE(dept_name));

CREATE TABLE Emp(emp_id NUMBER PRIMARY KEY,
                 emp_name varchar2(30) NOT NULL,
                 dept_id NUMBER REFERENCES Dept(dept_id),
                 job varhchar2(30), mgr varchar2(30),
                 salary NUMBER CHECK(Salary >= 5000),
                 comm NUMBER DEFAULT 100,
                 email varchar2(30), phone varchar2(13),
                 UNIQUE(email,phone));
```

INSERT

- Syntax 1: INSERT INTO <tablename> Values(attr_value1, attr_value2,……);
- Ex: INSERT INTO Dept Values(1, 'CE', 'Nadiad');
- Attr. values should in in the order of attr. list
- Number of attr. values should be equal to the number of attr. in the attr. list

Dept :

dept_id	dept_name	location
1	CE	Nadiad

INSERT

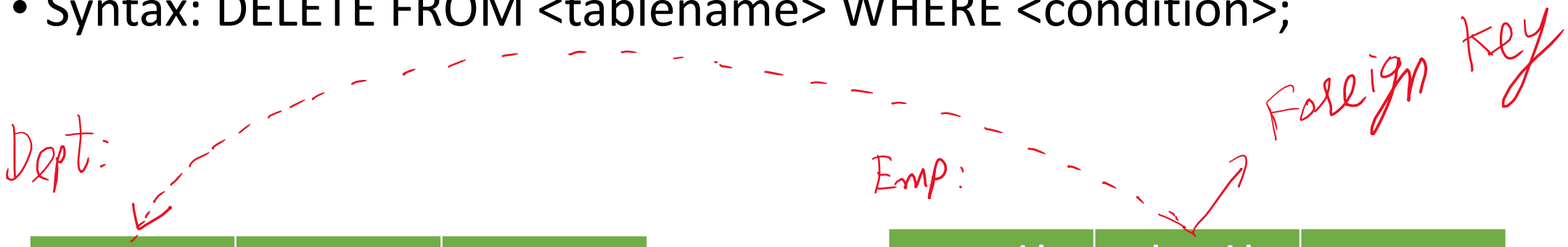
- Syntax 2: INSERT INTO <tablename>(attr1, attr2,attr3,……)
  Values(attr_value1, attr_value2,……..);
- Ex: INSERT INTO Dept(dept_id, dept_name, location)
  Values(2, 'IT', 'Ahmedabad');
- Ex: INSERT INTO Dept(dept_id, dept_name)
  Values(3, 'EC');

Dept :

dept_id	dept_name	location
1	CE	Nadiad
2	IT	Ahmedabad
3	EC	NULL

DELETE

- Deletes the data from the existing table
- Syntax: DELETE FROM <tablename> WHERE <condition>;



dept_id	dept_name	location
1	CE	Nadiad
2	IT	Ahmedabad
3	EC	Bhavnagar

emp_id	dept_id	emp_name
1	3	Abhay
2	2	Kunal
3	1	Chintan

DELETE

- Examples:
- DELETE FROM Emp WHERE emp_id = 1;

Emp:

emp_id	dept_id	emp_name
2	2	Kunal
3	1	Chintan

- DELETE * FROM Emp;

Emp:

emp_id	dept_id	emp_name

DELETE

- DELETE FROM Dept WHERE Dept_id = 1;

Referential Integrity gets Constraint violeted

Dept:

dept_id	dept_name	location
2	IT	Ahmedabad
3	EC	Bhavnagar

Emp:

emp_id	dept_id	emp_name
1	3	Abhay
2	2	Kunal
3	1	Chintan

TRUNCATE

- The TRUNCATE command will make the table empty i.e. all the table data will be deleted but the structure and database object is still alive and the table can be reused normally.
- Syntax: TRUNCATE TABLE <tablename>;
- Ex: TRUNCATE TABLE Emp;

  TRUNCATE TABLE Dept;

TRUNCATE vs DELETE

- DELETE command can be rollbacked. After deleting the data also, if you use ROLLBACK and SAVEPOINTS TCL commands, deleted data can be restored into the table as it is. But it is not possible with TRUNCATE.
- Whenever you are using DELETE command, you can specify conditions in WHERE clause for deleting so and so records. But, if you use TRUNCATE command, you cannot specify conditions and all the records will be deleted.

DROP

- Deletes both data and table
- Syntax: DROP TABLE <tablename>;
- Ex: DROP TABLE Emp;

  DROP TABLE Dept;

UPDATE

- Used for updating the data
- Syntax: UPDATE <tablename> SET <attr> = <value>
  WHERE <condition>;
- Ex: UPDATE Dept SET dept_name = ‘AI’ WHERE location = ‘Bhvanagar’;

Dept :

dept_id	dept_name	location
2	IT	Ahmedabad
3	AI	Bhavnagar

UPDATE

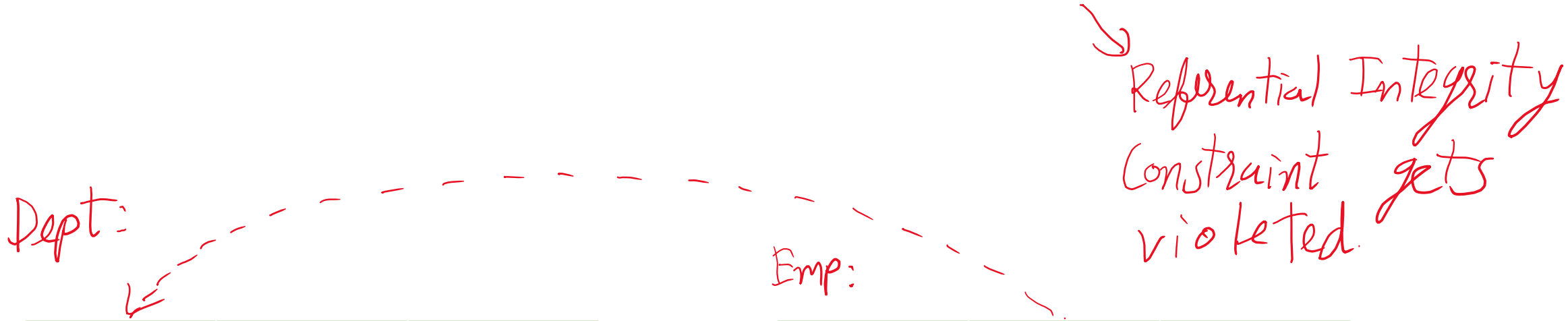
- UPDATE Emp SET dept_id = 2 WHERE emp_id = 3;

Emp :

emp_id	dept_id	emp_name
1	3	Abhay
2	2	Kunal
3	2	Chintan

UPDATE

- UPDATE Dept SET dept_id = 4 WHERE dept_name = 'AI';

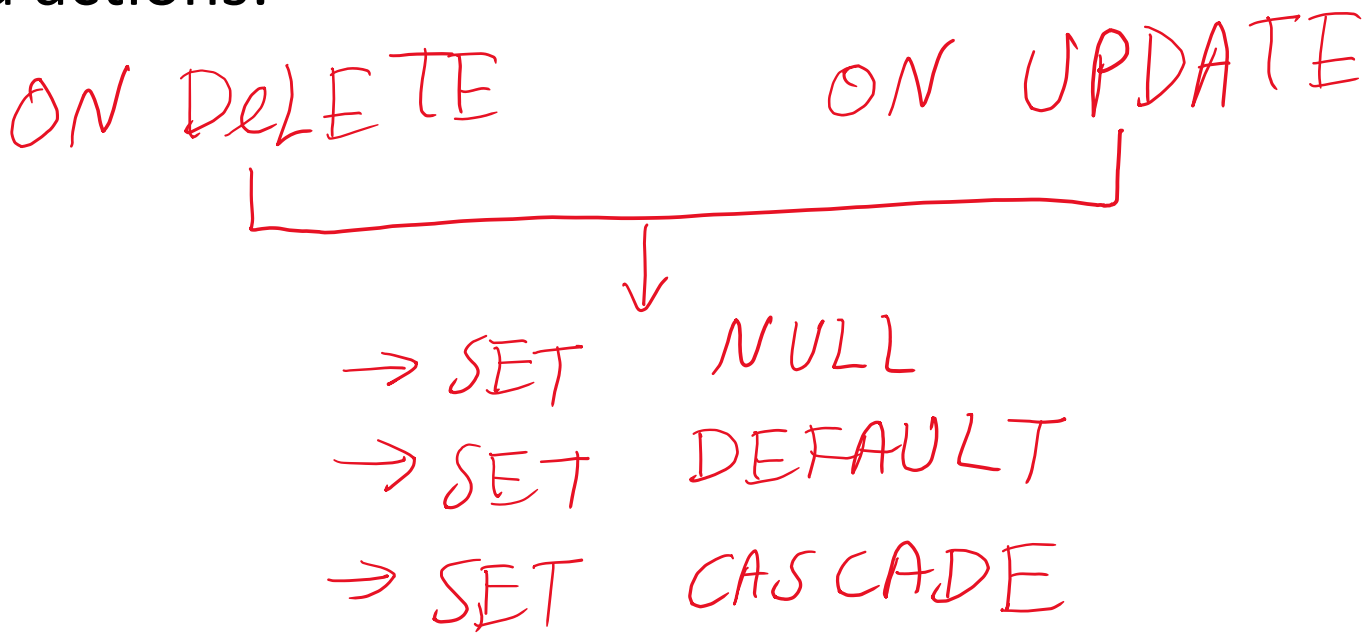


dept_id	dept_name	location
2	IT	Ahmedabad
4	AI	Bhavnagar

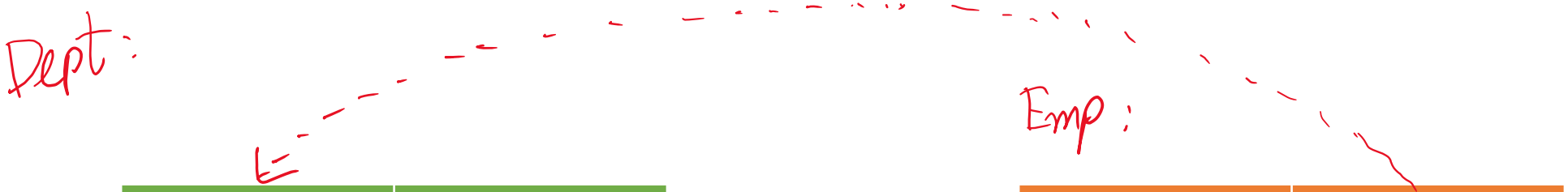
emp_id	dept_id	emp_name
1	3	Abhay
2	2	Kunal
3	2	Chintan

Referential Triggred Actions

- Allows us to further describe the relationship between the referential column and the object it references by attaching a ‘referential triggred action’ to the foreign key.
- Three triggred actions:



Referential Triggred Actions



dept_id	dept_name
1	CE
2	IT
3	EC
4	CH
5	IC
6	ME

emp_id	dept_id
1	3
2	2
3	4
4	6
5	5
6	1

```
CREATE TABLE Emp( emp_id NUMBER, dept_id NUMBER REFERENCES Dept(dept_id));
```

Referential Triggred Actions

- SET NULL
- dept_id NUMBER REFERENCES Dept(dept_id) ON DELETE SET NULL ON UPDATE SET NULL
- Examples:
- DELETE FROM Dept WHERE dept_id = 5;
- UPDATE Dept SET dept_id = 7 WHERE dept_id = 6;

Dept:

dept_id	dept_name
1	CE
2	IT
3	EC
4	CH
7	ME

Emp:

emp_id	dept_id
1	3
2	2
3	4
4	NULL
5	NULL
6	1

Referential Triggred Actions

- SET DEFAULT
- dept_id NUMBER DEFAULT 1 REFERENCES Dept(dept_id) ON DELETE SET DEFAULT ON UPDATE SET DEFAULT
- Examples:
- DELETE FROM Dept WHERE dept_id = 2;
- UPDATE Dept SET dept_id = 8 WHERE dept_id = 3;

Dept:

dept_id	dept_name
1	CE
8	EC
4	CH
7	ME

Emp:

emp_id	dept_id
1	1
2	1
3	4
4	NULL
5	NULL
6	1

Referential Triggred Actions

- SET CASCADE
- dept_id NUMBER REFERENCES Dept(dept_id) ON DELETE SET CASCADE ON UPDATE SET CASCADE
- Examples:
- DELETE FROM Dept WHERE dept_id = 4;
- UPDATE Dept SET dept_id = 9 WHERE dept_id = 1;

Dept:

dept_id	dept_name
9	CE
8	EC
7	ME

Emp:

emp_id	dept_id
1	9
2	9
4	NULL
5	NULL
6	9

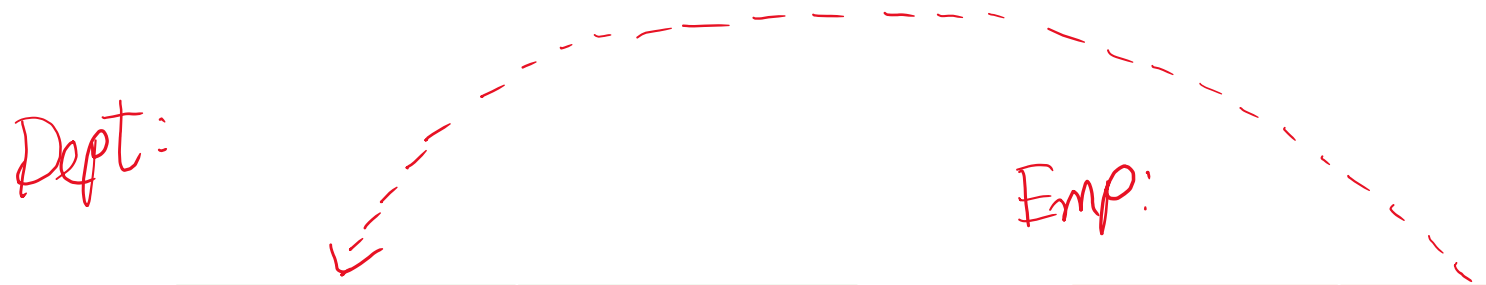
ALTER

- It is used to add, delete or modify columns in an existing table.
- Also used to add and drop various constraints on an existing table.
- Ex: CREATE TABLE Emp(emp_id NUMBER, emp_name char(10) UNIQUE,
  salary NUMBER CHECK(Salary >=5000)
  phone varchar2(13), pincode varchar2(6));
- ALTER TABLE Emp ADD PRIMARY KEY(emp_id);
- ALTER TABLE Emp DROP UNIQUE(emp_name);
- ALTER TABLE Emp MODIFY CHECK(salary >= 10000);
- ALTER TABLE Emp ADD Addr varcvhar2(30);
- ALTER TABLE Emp MODIFY phone charchar2(20);
- ALTER TABLE Emp DROP COLUMN pincode;

SELECT

- Used to retrieve data from database
- Syntax: SELECT <attr_list> FROM <table_list> WHERE <conditions>;
- attr_list: List of attributes whose values are to be retrieved by the query
- table_list: List of tables from which we retrieve the data
- condition: Conditional [Boolean] expressions that identifies the rows retrieved by the query

SELECT



dept_id	dept_name
1	CE
2	IT
3	EC
4	CH
5	IC

emp_id	dept	emp_name	Job
1	5	Abhay	a
2	2	Kunal	b
3	1	Chintan	a
4	2	Fatema	a
5	4	Prapti	b
6	3	Zarana	c

SELECT

- Q: Retrieve the names of the employees who works for department 1

➢ SELECT emp_name FROM Emp WHERE dept = 1;

- Q: Retrieve the names, emp_id of the employees who works for IT dept.

➢ SELECT emp_id, emp_name FROM Emp, Dept

WHERE dept = dept_id AND dept_name = 'IT';

- Retrieve the jobs of all the employees

➢SELECT job FROM Emp; → (a, b, a, a, b, c)

➢SELECT DISTINCT job FROM Emp; → {a, b, c}

VIEW

Dept :

dept_id	dept_name
1	CE
2	IT
3	EC
4	CH
5	IC

Student :

stud_id	dept	stud_name
1	2	Abhay
2	1	Kunal
3	3	Chintan
4	2	Fatema
5	4	Prapti
6	2	Zarana
7	2	Mohit

VIEW

- CREATE TABLE Stu_IT AS (SELECT stud_name, stud_id
  FROM Student, Dept
  WHERE dept = dept_id AND dept_name = ‘IT’);

Stu_IT :

stud_id	stud_name
1	Abhay
4	Fatema
6	Zarana
7	Mohit

VIEW

- It is a virtual table based on the result set of a SQL Statement.
- Syntax: CREATE VIEW <viewname> AS(SELECT <col_names>
  FROM <table_names>
  WHERE <conditions>);
- CREATE VIEW CEStudents AS (SELECT stud_name, stud_id
  FROM Student, Dept
  WHERE dept = dept_id AND dept_name = ‘CE’);

- SELECT stud_name FROM CEStudents WHERE stud_id = 2;

Aliasing

- SQL Aliases are used to temporarily rename a table (or) a column in a table.

Dept:

Dept_id	Dept_name
1	Accounting
2	Sales
3	Research
4	Finance

Emp:

Emp_id	dept_id	Emp_name	Sup_id	Sal
1	1	Abhay	7	50000
2	3	Kunal	4	55000
3	1	Chintan	7	59000
4	4	Fatema	7	50000
5	2	Prapti	4	35000
6	1	Zarana	7	90000
7	1	Mohit	NULL	30000

Aliasing

- Q: For each employee retrieve employee's id, name, salary and the name of his/her immediate supervisor.

➢ SELECT E.Emp_id AS "Employee_id", E.Sal AS "Salary",

E.Emp_name AS "Employee_name",

S.Emp_name AS "Supervisor_name"

FROM Emp AS E, Emp AS S

WHERE E.Sup_id = S.Emp_id;

Aliasing

- Q: Retrieve the names of the employees and their corresponding name of the department.

➢ SELECT E.Emp_name AS “Employee_name”,

D.Dept_name AS “Department_name”

FROM Emp E, Dept D

WHERE E.dept_id = D.Dept_id;

Pattern Matching

- LIKE: It is used to search for a specific pattern in a column
- We have two wild cards here

i. % , represents any sequence of 0 or more characters.

ii. _ , used to replace a single character.

Student :

name	%marks	rank	email
Chintan	78.8%	1234	abc_def@ghi.com
Yash	78.8%	1235	abc.def
Raj	90%	66	abc_xyz@ghi.com

Pattern Matching

- Q: Retrieve all the students whose name starts with ‘R’

➢ SELECT * FROM Student WHERE name LIKE ‘R%’;

- Q: Display the names of the students who secured 4 digit rank

➢ SELECT name FROM Student WHERE rank LIKE ‘_ _ _ _’;

❖ SELECT name FROM Student WHERE marks LIKE ‘90\%’ escape ‘\’ AND
email LIKE ‘abc_%@%.%’ escape ‘\’;

- Q: Retrieve all the students whose name does not starts with ‘R’

➢ SELECT * FROM Student WHERE name NOT LIKE ‘R%’;

Set Operations

- UNION
- INTERSECT
- EXCEPT

} These corresponds to mathematical set theory operations {∪, ∩, −}

- UNION ALL
- INTERSECT ALL
- EXCEPT ALL

Set Operations

Enroll :

studId	courseNAME	Grade
1	Maths	A
2	Physics	A
3	Chemistry	C
2	Maths	B
2	Biology	B
1	Chemistry	A
3	Physics	D
2	Chemistry	A
2	History	A

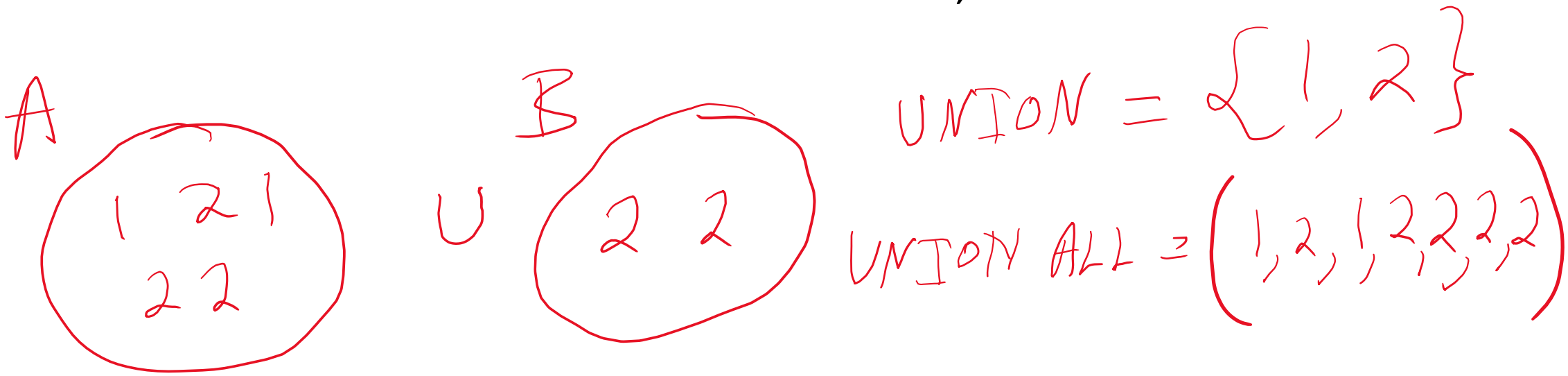
Set Operations

- Retrieve the list of studID's who get either grade 'A' or grade 'B'

- SELECT studID FROM Enroll WHERE Grade = 'A'

  UNION

  SELECT studID FROM Enroll WHERE Grade = 'B';



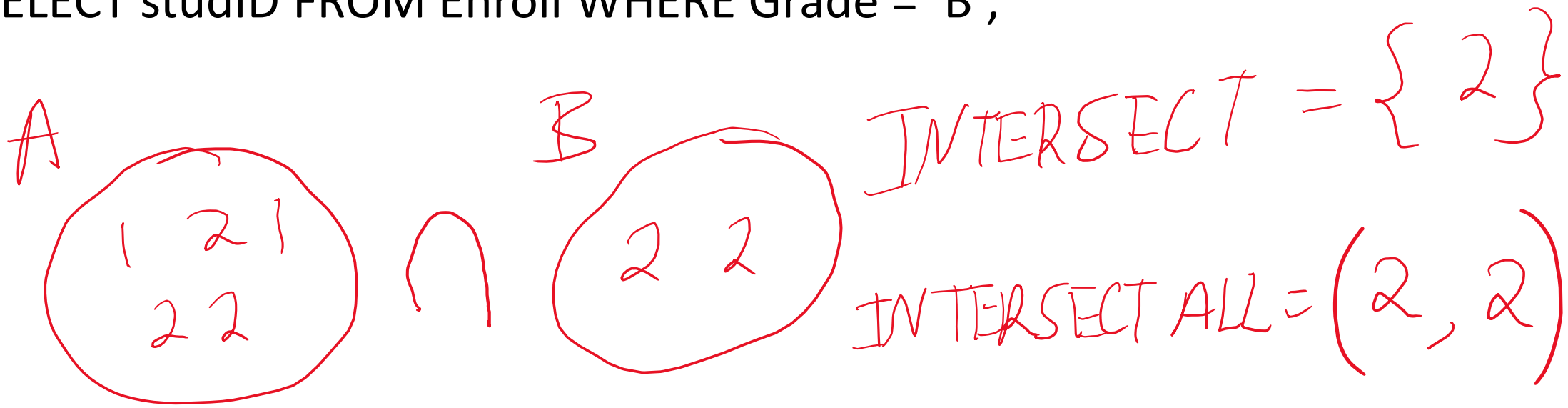
Set Operations

- Retrieve the list of studID's who get both grade 'A' and grade 'B'

- SELECT studID FROM Enroll WHERE Grade = 'A'

  INTERSECT

  SELECT studID FROM Enroll WHERE Grade = 'B';



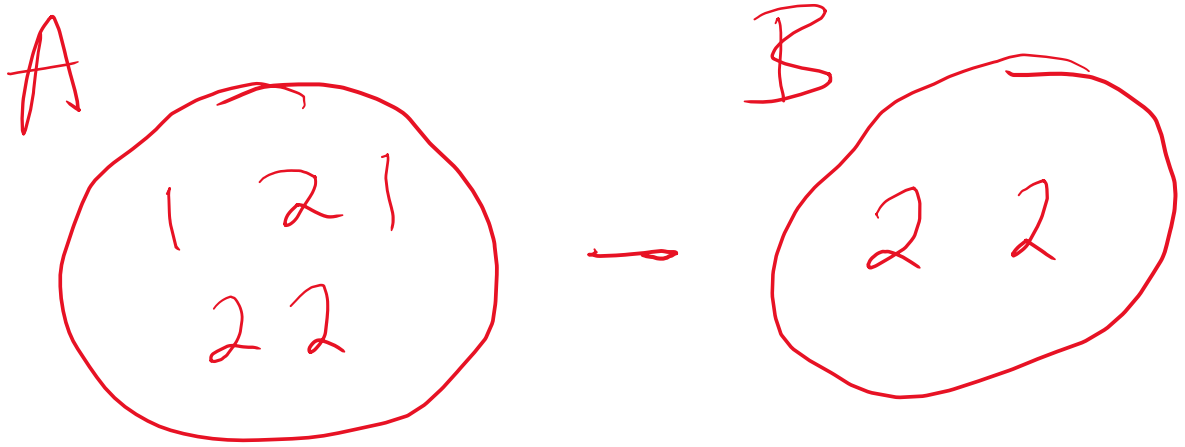
Set Operations

- Retrieve the list of studID's who get grade 'A' but not grade 'B'

- SELECT studID FROM Enroll WHERE Grade = 'A'

  EXCEPT

  SELECT studID FROM Enroll WHERE Grade = 'B';



EXCEPT = {1}

EXCEPT ALL = (1,1,2)

Arithmetic Operators

- +, -, *, /
- SQL Allows the use of arithmetic operators in queries on numeric domains.

❖Increase the salary of the employee by 10 percent and display the salary and employee name.

➢SELECT Emp_name, Sal*1.1 FROM Emp;

- CONCATENATE
- For string data type, the concatenate operator ‘||’ can be used in a query to append two string values.

➢SELECT FNAME || LNAME AS “FULL_NAME” FROM Emp;

BETWEEN

- SELECT Column_name(s) FROM Table_name WHERE
  Column_name BETWEEN Val_1 AND Val_2;

- It is a comparision operator on numeric domain.

❖ Retrieve all the employees whore salary is between 30,000 and 50,000

➢ SELECT * FROM Emp WHERE salary BETWEEN 30000 AND 50000;

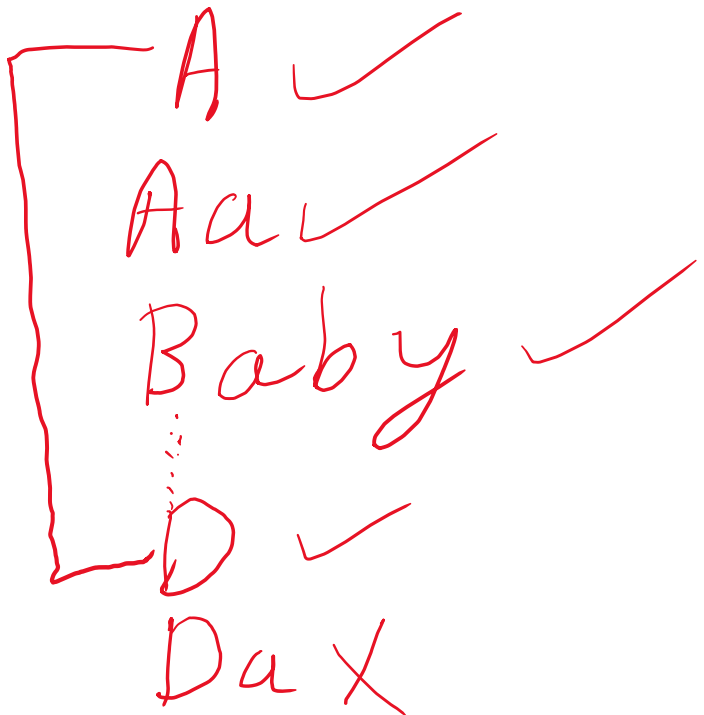
❖ Retrieve all the employees whore salary is not between 30,000 and 50,000

➢ SELECT * FROM Emp WHERE salary NOT BETWEEN 30000 AND 50000;

BETWEEN

- It is also a comparison operator on text and dates

❖ SELECT * FROM Emp WHERE emp_name BETWEEN ‘A’ AND ‘D’;



BETWEEN

- SELECT * FROM Emp WHERE hire-date BETWEEN
  '01-JAN-2010' AND '31-DEC-2010';

BETWEEN → Including Boundaries Also

NOT BETWEEN → Excluding Boundaries

ORDER BY

- ORDER BY is used to order/sort the result of the SQL query in ascending (or) descending order.
- Syntax: SELECT <column_list> FROM <table_list> ORDER BY <column_1> ASC/DESC, <column_2> ASC/DESC……;

R:

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

ORDER BY

- SELECT A,B,C FROM R ORDER BY A,B,C;

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

ORDER BY

- SELECT A,B,C FROM R ORDER BY A DESC,B,C;

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

ORDER BY

- SELECT A,B,C FROM R ORDER BY A DESC,B,C DESC;

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

SCHEMA for every example in the latter slides

- Employee(FNAME, LNAME, SSN, Bdate, Address, Sex, Salary, Superssn, Dno)
- Department(Dname, Dnumber, Mgrssn, MgrStartDate)
- Dept_Locations(Dnumber, Dlocation)
- Project(Pname, Pnumber, Plocation, Dnum)
- WORKS_ON(essn, Pno, Hours)
- Dependent(essn, Dependent_name, sex, bdate, Relationship)

ORDER BY Example

- Retrieve the list of employee names, their department & project names they are working on, ordered by department name and with in each department, ordered alphabetically by lastname, firstname.

➢ SELECT d.Dname, e.LNAME, e.FNAME, p.Pname

FROM Employee e, WORKS_ON w, Project p, Department d

WHERE e.Dno = d.Dnumber AND e.SSN = w.essn AND p.Pnumber = w.Pno

ORDER BY d.Dname, e.LNAME, e.FNAME;

Dealing with NULL values in various contexts

- Arithmetic expressions (+, - , *, /)

A
5
4
NULL
1

A
6
5
NULL
2

Dealing with NULL values in various contexts

- Logical expressions

A	B	A AND B	A OR B
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F
T	UK	UK	T
F	UK	F	UK
UK	UK	UK	UK

$(1 < NULL)$ AND $(1 < 2)$

$\Rightarrow$ UK AND T

$\Rightarrow$ UK (Unknown)

NOT (T) = F

NOT (F) = T

NOT (UK) = UK

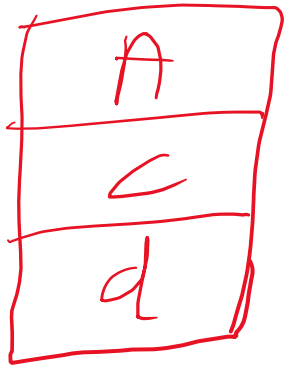
Dealing with NULL values in various contexts

- SELECT A FROM R WHERE B = NULL;

o/p: No rows

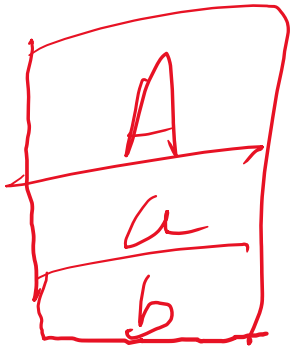
- SELECT A FROM R WHERE B IS NULL;

o/p:



- SELECT A FROM R WHERE B IS NOT NULL;

o/p:



R :

A	B
a	1
b	2
c	NULL
d	NULL

Dealing with NULL values in various contexts

- SELECT B FROM R;

(1, 2, N, N)

- SELECT DISTINCT B FROM R;

(1, 2, N)

R:

A	B
a	1
b	2
c	NULL
d	NULL

- (1, 2, N, N) UNION (1, 3, N, N) = {1, 2, 3, N}
- (1, 2, N, N) INTERSECT (1, 3, N, N) = {1, N}

IN Operator

- The IN Operator allows you to specify multiple values in a WHERE clause.
- Syntax: SELECT column_name(s) FROM Table_name

  WHERE Column_name IN(val1, val2,….);
- Ex: Retrieve the Fnames of employees who works for the departments 2,3,4,5.

➢ SELECT Fnames FROM Employee WHERE Dno IN(2,3,4,5);

- Ex: Find Fname, Address of the employees who work for the department located in ‘Newyork’

➢ SELECT Fname, Address FROM Employee

WHERE Dno IN(SELECT Dnumber FROM Dept_Locations

WHERE Dlocation = ‘Newyork’);

IN Operator Example-1

- Ex: Retrieve the Fnames, Lnames of all the managers.

➢ SELECT Fname, Lname FROM Employee e

WHERE SSN IN(SELECT Mgrssn FROM Department d);

- Ex: Retrieve the Fname of each employee who has a dependent with the same Fname and same sex as the employee.

➢ SELECT e.Fname FROM Employee e

WHERE e.SSN IN(SELECT d.essn FROM Dependent d

WHERE e.Fname = d.Dependent_name AND

e.sex = d.sex);

IN Operator Example-2

- Find out all the employee id’s who worked for some (or) any project on which employee 10 has worked for same numbers of hours.

➢SELECT DISTINCT essn FROM WORKS_ON

WHERE (Pno, Hours) IN(SELECT Pno, Hours FROM WORKS_ON

WHERE essn = 10);

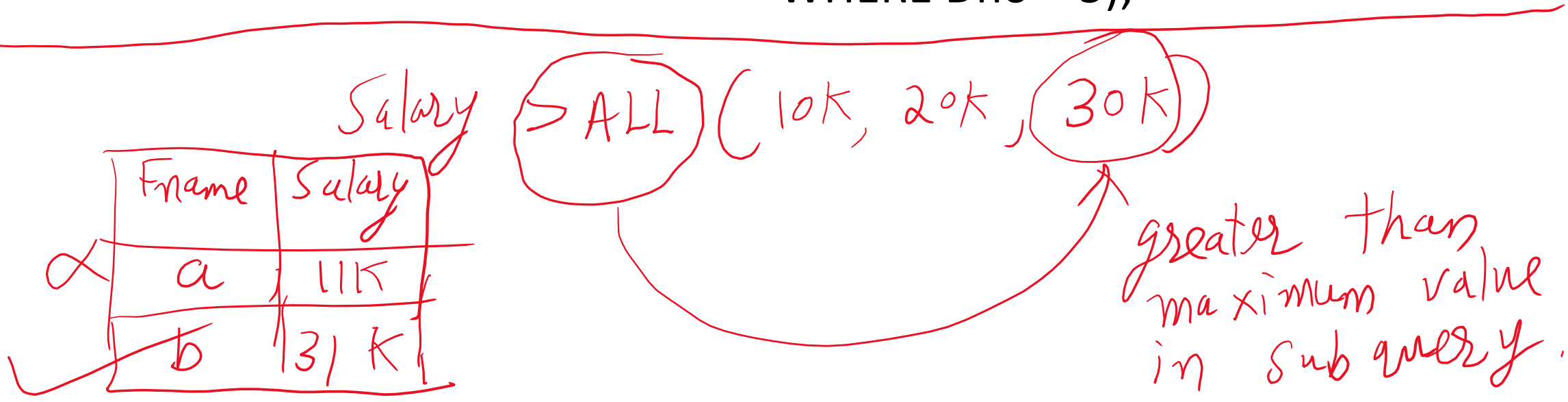
ANY/SOME, ALL(>, <, >=, <=, <>, =)

- Find out Fnames of all the employees whore salary is greater than all employees in department No: 5.

➢ SELECT FNAME FROM Employee

WHERE Salary > ALL(SELECT Salary FROM Employee

WHERE Dno = 5);



ANY/SOME, ALL(>, <, >=, <=, <>, =)

- SELECT FNAME FROM Employee

  WHERE Salary > ANY(SELECT Salary FROM Employee

  WHERE Dno = 5);

Salary > ANY (10K, 20K, 30K)

greater than minimum value in subquery.

	Fname	Salary
✓	a	11K
✓	b	20K
✓	c	21K
X	d	9K

$= ANY \equiv = SOME \equiv IN$

$<> ANY \equiv <> SOME \equiv NOT\ IN$

EXISTS

- The SQL EXISTS Condition is used in combination with a subquery and is considered to be met, if the subquery returns at least 1 row.
- Ex1: Retrieve the Fnames of the employees who have dependents with some Fname, sex as that of the employee.

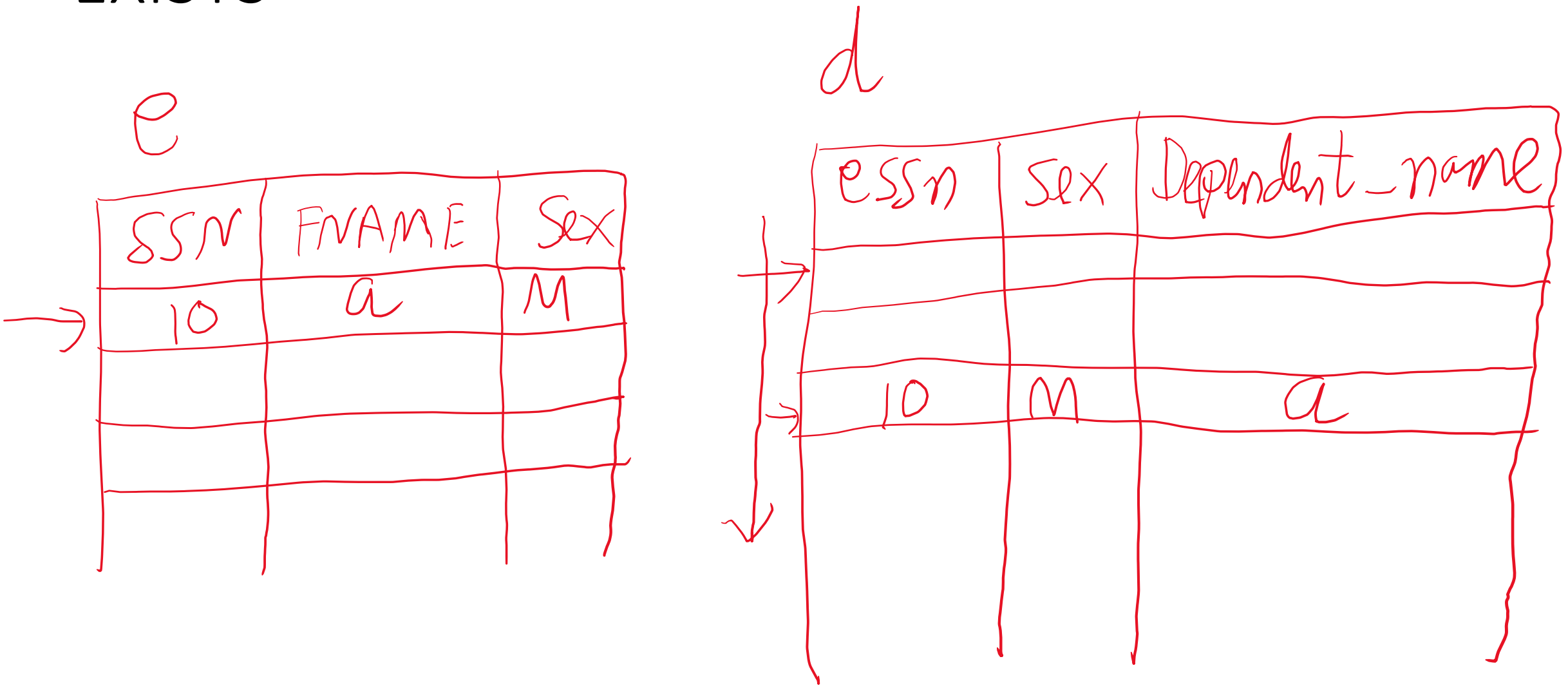
➢ SELECT e.FNAME FROM Employee e

WHERE EXISTS(SELECT * FROM Dependent d

WHERE e.SSN = d.essn AND e.Sex = d.sex AND

e.FNAME = d.Dependent_name);

EXISTS



NOT EXISTS

- Ex2: Retrieve the Fnames of the employees who have no dependents.

➢ SELECT e.FNAME FROM Employee e

WHERE NOT EXISTS(SELECT * FROM Dependent d

WHERE e.SSN = d.essn);

EXISTS Example

- List the Fnames of managers who have at least one dependent.

➢ SELECT e.FNAME FROM Employee
WHERE EXISTS(SELECT * FROM Dependent d WHERE e.SSN = d.essn)
AND EXISTS(SELECT * FROM Department Dept
WHERE e.SSN = Dept.Mgrssn);

NOT EXISTS Example

- Retrieve the Fname of each employee who works on all the projects controlled by department number 5.

➢ SELECT FNAME FROM Employee e WHERE

NOT EXISTS((SELECT Pnumber FROM Project WHERE Dnum = 5)

EXCEPT

(SELECT Pno FROM WORKS_ON w WHERE e.SSN = w.essn));

Non-Corelated Subqueries

- Find name of the student who has maximum marks in any subject.

➢ SELECT Name, Marks FROM Student
WHERE Marks = (SELECT MAX(Marks) FROM Student);

o/p:

Name	Marks
Chintan	100

Student :

Name	Subject	Marks
Chintan	CN	100
Kunal	DBMS	90
Chintan	DBMS	80
Prapti	CN	70
Kunal	CN	70
Prapti	DBMS	60

Non-Corelated Subqueries

- Print Marks of all students in every subject except the students that are suspended.

➢ SELECT * FROM Student

WHERE Name NOT IN(SELECT Name

FROM Suspended);

Suspended :

Name	Reason
Kunal	Discipline
Rishit	Attendance

Student :

Name	Subject	Marks
Chintan	CN	100
Kunal	DBMS	90
Chintan	DBMS	80
Prapti	CN	70
Kunal	CN	70
Prapti	DBMS	60

Non-Corelated Subqueries

- Find Second Highest Salary.

➢ SELECT MAX(Salary) AS Second_max_Sal FROM Employee
WHERE Salary < (SELECT MAX(Salary) FROM Employee);

o/p :

Second_max_Sal
55000

Employee :

Name	Dept	Salary
Chintan	Engineering	60000
Kunal	Sales	40000
Shivam	Sales	45000
Prapti	Engineering	55000
Meet	Marketing	50000
Mohit	Marketing	48000

Non-Corelated Subqueries

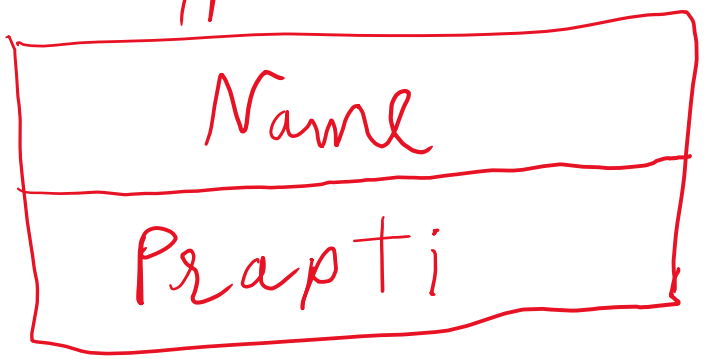
- Find name(s) of the employee(s) having second highest salary.

➢ SELECT Name FROM Employee WHERE

Salary = (SELECT MAX(Salary) AS Second_max_Sal FROM Employee

WHERE Salary < (SELECT MAX(Salary) FROM Employee));

o/p:



Employee:

Name	Dept	Salary
Chintan	Engineering	60000
Kunal	Sales	40000
Shivam	Sales	45000
Prapti	Engineering	55000
Meet	Marketing	50000
Mohit	Marketing	48000

Non-Corelated Subqueries

- Find details of the employee with N-th highest salary. (Let N=5)

➢ SELECT * FROM Employee WHERE

Salary = (SELECT MIN(Salary) FROM Employee WHERE

Salary IN(SELECT DISTINCT TOP 5

Salary FROM Employee

ORDER BY Salary DESC));

Employee:

Name	Dept	Salary
Chintan	Engineering	60000
Kunal	Sales	40000
Shivam	Sales	45000
Prapti	Engineering	55000
Meet	Marketing	50000
Mohit	Marketing	48000

Corelated Subqueries

- Subquery uses value of the outer query.
- Subquery is evaluated for each row of the outer query.

Employee :

Name	Dept_id	Salary
Chintan	2	50000
Kunal	2	30000
Shivam	1	40000
Prapti	3	80000
Meet	1	30000
Raj	1	20000
Mohit	3	70000

Corelated Subqueries

- Example: Find Name and Dept_id of every employee whose salary is above average in the department.

➢ SELECT E1.Name, E1.Dept_id FROM Employee E1 WHERE

Salary > (SELECT AVG(Salary) FROM Employee E2

WHERE E1.Dept_id = E2.Dept_id);

o/p:

Name	Dept_id
Chintan	2
Shivam	1
Prapti	3

Corelated Subqueries

Employee :

Name	Dept_id	Salary
Chintan	2	50000
Kunal	2	30000
Shivam	1	40000
Prapti	3	80000
Meet	1	30000
Raj	1	20000
Mohit	3	70000

Department :

Dept_id	Name
1	Engineering
2	Sales
3	Marketing
4	HR
5	Graphics

Corelated Subqueries

- Find names of the departments that do not have any employee.

➢ SELECT d.Name FROM Department d WHERE

NOT EXISTS (SELECT e.Dept_id FROM Employee e

WHERE e.Dept_id = d.Dept_id);

- Print Employee Names, their salaries and their Department average salaries.

➢ SELECT Names, Salary, (SELECT AVG(Salary) FROM Employee e1

WHERE e1.Dept_id = e2.Dept_id) AS Average

FROM Employee e2;

Aggregate Functions

- Aggregate Functions are functions that take a collection(a set or multiset) of values as input and return a single value.
- Average : AVG ; Minimum : MIN ; Maximum : MAX ; Total : SUM ; Count : COUNT
- SELECT AVG(Salary) FROM Emp;
- AVG and SUM have to be strictly applied on numeric domains. Applying these functions on any other domain is meaningless.
- AVG, MIN, MAX and SUM can be applied to only one attribute.
- COUNT can be applied to any number of attributes.
- SELECT AVG(DISTINCT Salary) FROM Emp;
- We can apply arithmetic operators while using aggregate functions.
- SELECT AVG(marks_subject1 + marks_subject2) FROM Student;
- We cannot specify aggregate functions within WHERE clause.

Dealing with NULL values in Aggregate functions

- All aggregate functions except COUNT(*) ignore NULL values in their input collection.
- The count of an empty collection is defined to be Zero(0) and all other aggregate operations return a value of NULL when applied on an empty set.

Ex:

A	B
1	NULL
2	NULL
NULL	NULL
3	NULL

COUNT(*) = 4

COUNT(A) = 3

COUNT(B) = 0

SUM(A) = 6

AVG(A) = 2

↳ $\frac{1+2+3}{3}$

MAX(A) = 3

MIN(A) = 1

SUM(B) = NULL

AVG(B) = NULL

MAX(B) = NULL

MIN(B) = NULL

GROUP BY

Employee:

Emp_id	Dno	Salary
1	1	10000
2	2	20000
3	3	30000
4	1	40000
5	2	50000
6	3	60000

- Find the average salary in each department.

➢ SELECT Dno, AVG(Salary) FROM Employee GROUP BY Dno;

o/p:

1	25000
2	35000
3	45000

- All attributes used in the GROUP BY clause need to appear in the SELECT clause. Any attribute that is not present in the GROUP BY clause must appear only inside the aggregate function in the SELECT clause.

HAVING clause

- List the department numbers with average salary greater than 20,000.

➢ SELECT Dno FROM EMPLOYEE GROUP BY Dno HAVING AVG(Salary) > 20000

- According to SQL Standards, the SQL query can contain a HAVING clause only if it has a GROUP BY clause. Different practical implementation may contradict this statement.

Order of execution of an SQL Query

GROUP BY Example

- For each department that has more than 5 employees retrieve the department name and the number of employees who are making more than 40,000.

➢ SELECT Dname, COUNT(*) FROM Deparmtent, Employee

WHERE Dnumber = Dno AND Salary > 40000 AND

Dno IN(SELECT Dno FROM Employee GROUP BY Dno HAVING

COUNT(*) > 5)

GROUP BY Dname;

JOINS

- NORMAL JOIN:

➢ Specify a condition on which you are going to join the tables.

➢ SELECT * FROM (R JOIN S ON A=C);

R:

A	B
1	a
2	b
3	c
4	d
5	e
6	f

S:

C	D
1	g
2	h
3	i
7	j
8	k
9	l

o/p:

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i

JOINS

- NATURAL JOIN:
  - It is the same as a normal join, the only difference is we are not going to specify any condition. We are going to apply natural join only under the condition that two attributes are the same or they have the same name in the tables. Otherwise, we temporarily rename the attribute and apply the natural join.
  - SELECT * FROM (R NATURAL JOIN S);

o/p:

A	B	D
1	a	g
2	b	h
3	c	i

R:

A	B
1	a
2	b
3	c
4	d
5	e
6	f

S:

A	D
1	g
2	h
3	i
7	j
8	k
9	l

JOINS

- LEFT OUTER JOIN:

➢SELECT * FROM (R LEFT OUTER JOIN S ON A=C);

R:

A	B
1	a
2	b
3	c
4	d
5	e
6	f

S:

C	D
1	g
2	h
3	i
7	j
8	k
9	l

o/p:

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i
4	d	NULL	NULL
5	e	NULL	NULL
6	f	NULL	NULL

JOINS

- RIGHT OUTER JOIN:

➢SELECT * FROM (R RIGHT OUTER JOIN S ON A=C);

R:

A	B
1	a
2	b
3	c
4	d
5	e
6	f

S:

C	D
1	g
2	h
3	i
7	j
8	k
9	l

o/p:

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i
NULL	NULL	7	j
NULL	NULL	8	k
NULL	NULL	9	l

JOINS

- FULL OUTER JOIN:

➢SELECT * FROM (R FULL OUTER JOIN S ON A=C);

o/p:

A	B	C	D
1	a	1	g
2	b	2	h
3	c	3	i
4	d	NULL	NULL
5	e	NULL	NULL
6	f	NULL	NULL
NULL	NULL	7	j
NULL	NULL	8	k
NULL	NULL	9	l

R:

A	B
1	a
2	b
3	c
4	d
5	e
6	f

S:

C	D
1	g
2	h
3	i
7	j
8	k
9	l

JOINS

- NATURAL LEFT OUTER JOIN:

➢ SELECT * FROM (R NATURAL LEFT OUTER JOIN S);

o/p:

A	B	D
1	a	g
2	b	h
3	c	i
4	d	NULL
5	e	NULL
6	f	NULL

R:

A	B
1	a
2	b
3	c
4	d
5	e
6	f

S:

A	D
1	g
2	h
3	i
7	j
8	k
9	l

JOINS

- NATURAL RIGHT OUTER JOIN:

➢ SELECT * FROM (R NATURAL RIGHT OUTER JOIN S);

R:

A	B
1	a
2	b
3	c
4	d
5	e
6	f

S:

A	D
1	g
2	h
3	i
7	j
8	k
9	l

o/p:

A	B	D
1	a	g
2	b	h
3	c	i
7	NULL	j
8	NULL	k
9	NULL	l

JOINS

- NATURAL FULL OUTER JOIN:

➢ SELECT * FROM (R NATURAL FULL OUTER JOIN S);

o/p:

A	B	D
1	a	g
2	b	h
3	c	i
4	d	NULL
5	e	NULL
6	f	NULL
7	NULL	j
8	NULL	k
9	NULL	l

R:

A	B
1	a
2	b
3	c
4	d
5	e
6	f

S:

A	D
1	g
2	h
3	i
7	j
8	k
9	l

JOINS Example

- Q1:

➢ SELECT FNAME, LNAME, Address FROM (Employee JOIN Department ON Dno = Dnumber) WHERE Dname = ‘Research’;

➢ From this query, we are going to get information of the employees who are working for the Research department.

JOINS Example

- Q2:

➢ SELECT FNAME, LNAME, Address FROM

(Employee NATURAL JOIN Department AS Dept(Dname, Dno, Mgrssn, MgrStartDate)) WHERE Dname = ‘Research’;

➢ Here, we have used natural join so that no condition is needed for doing join operation. So, for doing natural join, we have renamed the attribute Dnumber of Department table to Dno. From this query, we are going to get information of the employees who are working for the Research department.

JOINS Example

- Q3:

➢ SELECT e.LNAME AS Emp_name , s.LNAME AS Sup_Name FROM

(Employee AS e LEFT OUTER JOIN Employee AS s ON e.Superssn = s.SSN);

➢ In this query, every employee will be joined with his/her corresponding supervisor and by applying LEFT OUTER JOIN, the employees that are not having any corresponding supervisor will also be listed in the output.

GATE 2012 question on SQL

which of the following statements are TRUE about an SQL query? P:An SQL query can contain a HAVING clause even if it does not have a GROUP BY clause Q:An SQL query can contain a HAVING clause even if it has a GROUP BY clause R: All attributes used in the GROUP BY clause must appear in the SELECT clause S: Not all attributes used in the GROUP BY clause need to appear in the SELECT clause

A P and R

B P and S

✓ Q and R

D Q and S

GATE 2012 question on SQL

- **Explanation:**
- The SQL GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause. The attributes used in the GROUP BY clause must be present in the SELECT statement.
- The HAVING Clause enables you to specify conditions that filter which group results appear in the results. The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause. So, we cannot use the HAVING clause without the GROUP BY clause.

GATE 1998 question on SQL

Suppose we have a database consisting of the following three relations.
• FREQUENTS (student, parlor) giving the parlors each student visits.
• SERVES (parlor, ice cream) indicating what kind of ice-creams each parlor serves.
• LIKES (student, ice cream) indicating what ice-creams each student likes.
(Assume that each student likes at least one ice cream and frequents at least one parlor)
Express the following in SQL:
Print the students that frequent at least one parlor that serves some ice cream that they like.

Explanation: Answer:

```
SELECT DISTINCT FREQUENTS.student FROM
   FREQUENTS, SERVES, LIKES
   WHERE
       FREQUENTS.parlor=SERVES.parlor
       AND
       SERVES.ice-cream=LIKES.ice-cream
       AND
       FREQUENTS.student=LIKES.student;
```

Which of the following is/are correct?

A An SQL query automatically eliminates duplicates

B An SQL query will not work if there are no indexes on the relations

C SQL permits attribute names to be repeated in the same relation

✓ None of the above

Database-Management-System SQL Gate-1999

Explanation:

→ SQL won't remove duplicates like relational algebra projection, we have to remove it explicitly by distinct.

→ If there are no indexes on the relation SQL, then also it works.

→ SQL does not permit 2 attributes to have same name in a relation.

Given relations r(w, x) and s(y, z), the result of

```
select distinct w,x
from r, s
```

is guaranteed to be same as r, provided

- ✓ r has no duplicates and s is non-empty
- B r and s have no duplicates
- C s has no duplicates and r is non-empty
- D r and s have the same number of tuples

Database-Management-System SQL Gate-2000

Explanation:

r has no duplicate, if r can have duplicates it can be remove in the final state. s in non-empty if s is empty then r*s becomes empty.

GATE 2003 question on SQL

Consider the following SQL query

```
select distinct a1, a2,........., an
from r1, r2,........, rm
where P
```

For an arbitrary predicate P, this query is equivalent to which of the following relational algebra expressions ?

✓ $\prod_{a_1,a_2,\ldots a_n} \sigma_p(r_1 \times r_2 \times \ldots \times r_m)$

B $\prod_{a_1,a_2,\ldots a_n} \sigma_p(r_1 \bowtie r_2 \bowtie \ldots \bowtie r_m)$

C $\prod_{a_1,a_2,\ldots a_n} \sigma_p(r_1 \cup r_2 \cup \ldots \cup r_m)$

D $\prod_{a_1,a_2,\ldots a_n} \sigma_p(r_1 \cap r_2 \cap \ldots \cap r_m)$

Explanation:

If we want to get distinct elements then we need to perform cross product in between the relations r_1, r_2, …. r_m.

GATE 2011 question on SQL

Database table by name Loan_Records is given below.

Borrower	Bank_Manager	Loan_Amount
Ramesh	Sunderajan	10000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	7000.00

What is the output of the following SQL query?

```
SELECT Count(*)
FROM  ( ( SELECT Borrower, Bank_Manager
          FROM Loan_Records) AS S
          NATURAL JOIN ( SELECT Bank_Manager, Loan_Amount
                         FROM Loan_Records) AS T );
```

A 3

B 9

✓ 5

D 6

S

Borrower	Bank – Manager
Ramesh	Sunderajan
Suresh	Ramgopal
Mahesh	Sunderjan

T

Bank – Manager	Loan - Amount
Sunderajan	10000.00
Ramgopal	5000.00
Sunderjan	7000.00

After executing the given query, the output would be

Borrower	Bank – Manager	Load – Amount
Ramesh	Sunderajan	10000.00
Ramesh	Sunderajan	7000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	10000.00
Mahesh	Sunderajan	7000.00

GATE 2012 question on SQL

Consider the above tables A, B and C. How many tuples does the result of the following SQL query contains?

A

Id	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

B

Id	Name	Age
15	Shreya	24
25	Hari	40
98	Rohit	20
99	Rohit	11

C

Id	Name	Area
10	2200	02
99	2100	01

```
SELECT A.id
FROM   A
WHERE  A.age > ALL (SELECT B.age
                    FROM   B
                    WHERE  B. name = "arun")
```

A 4

✓ 3

C 0

D 1

GATE 2012 question on SQL

```
SELECT A. Id
FROM A
WHERE A. Age  > ALL (SELECT B. Age
                     FROM B
                     WHERE B. Name  =  'Arun') ;
```

① ②

First query (2) will be executed and 0 (no) rows will be selected because in relation B there is no Name 'Arun'.

The outer query (1) results the follow and that will be the result of entire query now. (Because inner query returns 0 rows).

Id
12
15
99

Given the following schema:

```
employees(emp-id, first-name, last-name, hire-date, dept-id, salary)
departments(dept-id, dept-name, manager-id, location-id)
```

You want to display the last names and hire dates of all latest hires in their respective departments in the location ID 1700. You issue the following query:

```
SQL> SELECT last-name, hire-date
     FROM employees
     WHERE (dept-id, hire-date) IN ( SELECT dept-id, MAX(hire-date)
                                     FROM employees JOIN departments USING(dept-id)
                                     WHERE location-id = 1700
                                     GROUP BY dept-id);
What is the outcome?
```

A It executes but does not give the correct result.

✓ It executes and gives the correct result.

C It generates an error because of pairwise comparison.

D It generates an error because the GROUP BY clause cannot be used with table joins in a subquery.

Database-Management-System SQL GATE 2014(Set-01)

[redacted] Explanation:

The given SQL query will display the last names and hire-dates of all latest hires in their respective departments in the location ID 1700. So, correct option is (B).

SQL allows tuples in relations, and correspondingly defines the multiplicity of tuples in the result of joins. Which one of the following queries always gives the same answer as the nested query shown below:

```
select * from R where a in (select S.a from S)
```

A select R.* from R,S where R.a=S.a

B select distinct R.* from R,S where R.a=S.a

✓ select R.* from R,(select distinct a from S) as S1 where R.a=S1.a

D select R.* from R,S where R.a=S.a and is unique R

Database-Management-System SQL Gate 2014 Set -02

Explanation:

Multiplicity of duplicate tuples will be distributed when there is a match between R.a and S.a; and for that match, S.a's value is repeated in each cases except the third case. So, the output of query given in the question matches with the output of (C).

The relation book (title, price) contains the titles and prices of different books. Assuming that no two books have the same price, what does the following SQL query list?

```
select title
from book as B
where (select count(*)
   from book as T
   where T.price > B.price) < 5
```

A Titles of the four most expensive books

B Title of the fifth most inexpensive book

C Title of the fifth most expensive book

✓ Titles of the five most expensive books

Database-Management-System SQL Gate-2005

Explanation:

Which results titles of the five most expensive books.

The where clause of outer query will be true for 5 most expensive books.

Consider the set of relations shown below and the SQL query that follows.

```
Students: (Roll_number, Name, Date_of_birth)
  Courses: (Course number, Course_name, Instructor)
  Grades: (Roll_number, Course_number, Grade)
```

```
select distinct Name
        from Students, Courses, Grades
        where Students. Roll_number = Grades.Roll_number
              and Courses.Instructor = Korth
              and Courses.Course_number = Grades.Course_number
              and Grades.grade = A
```

Which of the following sets is computed by the above query?

A Names of students who have got an A grade in all courses taught by Korth

B Names of students who have got an A grade in all courses

✓ Names of students who have got an A grade in at least one of the courses taught by Korth

D in none of the above

Database-Management-System SQL Gate-2003

Explanation:

The query results a names of students who got an A grade in at least one of the courses taught by korth.

The employee information in a company is stored in the relation

Employee (name, sex, salary, deptName)

Consider the following SQL query

```
Select deptName
        From Employee
        where sex = 'M'
        Group by deptName
        Having avg (salary) >
            (select avg (salary) from Employee)
```

It returns the names of the department in which

A. the average salary is more than the average salary in the company

B. the average salary of male employees is more than the average salary of all male employees in the company

C. the average salary of male employees is more than the average salary of employees in the same department

✓ the average salary of male employees is more than the average salary in the company

Database-Management-System SQL Gate-2004

Explanation:

Group by (avg(salary) > (select avg (salary) from employee))

This results the employees who having the salary more than the average salary.

Sex = M

Selects the Male employees whose salary is more than the average salary in the company.

GATE 2006 question on SQL

Consider the relation "enrolled(student, course)" in which (student, course) is the primary key, and the relation "paid(student, amount)" where student is the primary key. Assume no null values and no foreign keys or integrity constraints. Given the following four queries:

```
Query1: select student from enrolled where
        student in (select student from paid)
Query2: select student from paid where
        student in (select student from enrolled)
Query3: select E.student from enrolled E, paid P
         where E.student = P.student
Query4:  select student from paid where exists
        (select * from enrolled where enrolled.student
         = paid.student)
```

Which one of the following statements is correct?

A. All queries return identical row sets for any database.

✓ Query2 and Query4 return identical row sets for all databases but there exist databases for which Query1 and Query2 return different row sets.

C. There exist databases for which Query3 returns strictly fewer rows than Query2.

D. There exist databases for which Query4 will encounter an integrity violation at runtime.

GATE 2006 question on SQL

Consider Table examples as:

Table enrolled		Table Paid	
Student	Course	Student	Amount
abcd	d1	abcd	2000
PQRS	d1	PQRS	1000
KLMN	d1	UVWX	1000
abcd	d2		

Query1 : Output

abcd

abcd

PQRS

Query 2 : Output

abcd

PQRS

Query 3 : Output

abcd

PQRS

Query 4 : Output

abcd

PQRS

Query 2 & Query 4 gives same results but Query 1 & Query 3 gives different results.

SELECT operation in SQL is equivalent to

A the selection operation in relational algebra

B the selection operation in relational algebra, except that SELECT in SQL retains duplicates

C the projection operation in relational algebra

✓ the projection operation in relational algebra, except that SELECT in SQL retains duplicates

Database-Management-System SQL GATE 2015 (Set-01)

Explanation:

SELECT operation in SQL perform vertical partitioning which is performed by projection operation in relational calculus but SQL is multi sets; hence (D).

Consider the following relation

```
Cinema (theater, address, capacity)
```

Which of the following options will be needed at the end of the SQL query

```
SELECT P1. address
FROM Cinema P1
```

Such that it always finds the addresses of theaters with maximum capacity?

✓ WHERE P1.capacity >= All (select P2.capacity from Cinema P2)

B WHERE P1.capacity >= Any (select P2.capacity from Cinema P2)

C WHERE P1.capacity > All (select max(P2.capacity) from Cinema P2)

D WHERE P1.capacity > Any (select max(P2.capacity) from Cinema P2)

Database-Management-System SQL GATE 2015(Set-03)

Explanation:

Inner query collects capacities of all the theatres and in outer query we are filtering the tuples with the condition “capacity >= All”.
So the theatres which are having maximum capacity will satisfy the condition.

GATE 1997 question on SQL

Let R(a, b, c) and S(d, e, f) be two relations in which d is the foreign key of S that refers to the primary key of R. Consider the following four operations R and S

(a) Insert into R (b) Insert into S

(c) Delete from R (d) Delete from S

Which of the following is true about the referential integrity constraint above?

A None of (a), (b), (c) or (d) can cause its violation

B All of (a), (b), (c) and (d) can cause its violation

C Both (a) and (d) can cause its violation

✓ Both (b) and (c) can cause its violation

GATE 1997 question on SQL

Let take example:

R:

a	b	c
S_1	—	—
S_2	—	—
S_3	—	—

S:

d	e	f
S_1	—	—
S_2	—	—

Here 'd' is the foreign key of S and let 'a' is the primary key of R.

(A) Insertion into R: will cause no violation.

(B) Insertion into S: may cause violation because there may not be entry of the tuple in relation R. Example entry of $\langle S_4, \_\_, \_\_\rangle$ is not allowed.

(C) Delete from R: may cause violation. For example, deletion of tuple $\langle S_2, \_\_, \_\_\rangle$ will cause violation as there is entry of S_2 in the foreign key table.

(D) Delete from S: will cause no violation as it does not result inconsistency.

GATE 2004 question on SQL

Consider two tables in a relational database with columns and rows as follows:

Table: Student

ROLL_NO	NAME	DEPT_ID
1	ABC	1
2	DEF	1
3	GHI	2
4	JKL	3

Table: Department

DEPT_ID	DEPT_NAME
1	A
2	B
3	C

Roll_no is the primary key of the Student table, Dept_id is the primary key of the Department table and Student.Dept_id is a foreign key from Department.Dept_id
What will happen if we try to execute the following two SQL statements?

1. update Student set Dept_id = Null where Roll_on = 1

1. update Department set Dept_id = Null where Dept_id = 1

A	Both (i) and (ii) will fail
B	(i) will fail but (ii) will succeed
✔	(i) will succeed but (ii) will fail
D	Both (i) and (ii) will succeed

GATE 2004 question on SQL

- **Explanation:**
- Here in (i), when we update in Student table, Dept_id = Null, then it will not cause any problem to the referenced table.
- But in (ii) if we set in Department table, Dept_id = Null, then it will produce inconsistency because in the Student table we will still have the tuples containing the Dept_id = 1.

A table T1 in a relational database has the following rows and columns:

roll no.	marks
1	10
2	20
3	30
4	Null

The following sequence of SQL statements was successfully executed on table T1.

```
Update T1 set marks = marks + 5
Select avg(marks) from T1
```

What is the output of the select statement?

A 18.75

B 20

✓ 25

D NULL

Database-Management-System SQL Gate 2004-IT

Explanation:

Update on null values gives null. Now, avg function ignores null values. So, here avg will be (15+25+35)/3 = 25

A relational database contains two tables student and department in which student table has columns roll_no, name and dept_id and department table has columns dept_id and dept_name. The following insert statements were executed successfully to populate the empty tables:

```
Insert into department values (1, 'Mathematics')
Insert into department values (2, 'Physics')
Insert into student values (1, 'Navin', 1)
Insert into student values (2, 'Mukesh', 2)
Insert into student values (3, 'Gita', 1)
```

How many rows and columns will be retrieved by the following SQL statement?

```
Select * from student, department
```

A 0 row and 4 columns

B 3 rows and 4 columns

C 3 rows and 5 columns

✓ 6 rows and 5 columns

Database-Management-System SQL Gate 2004-IT

[redacted] Explanation:

Simply, cartesian product of two tables will result

rows = 3 * 2 = 6

Columns = 3 + 2 = 5

In an inventory management system implemented at a trading corporation, there are several tables designed to hold all the information. Amongst these, the following two tables hold information on which items are supplied by which suppliers, and which warehouse keeps which items along with the stock-level of these items. Supply = (supplierid, itemcode) Inventory = (itemcode, warehouse, stocklevel) For a specific information required by the management, following SQL query has been written

```
Select distinct STMP.supplierid
From Supply as STMP
Where not unique (Select ITMP.supplierid
                  From Inventory, Supply as ITMP
                  Where STMP.supplierid = ITMP.supplierid
                  And ITMP.itemcode = Inventory.itemcode
                  And Inventory.warehouse = 'Nagpur');
```

For the warehouse at Nagpur, this query will find all suppliers who

A do not supply any item

B supply exactly one item

C supply one or more items

✓ supply two or more items

Database-Management-System SQl Gate 2005-IT

Explanation:

Here (not unique) in nested query ensures that only for those suppliers it return True which supplies more than 1 item in which case supplier id in inner query will be repeated for that supplier. Hence, the answer is (D) which supply two or more items.

The following table has two attributes A and C where A is the primary key and C is the foreign key referencing A with on-delete cascade.

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

The set of all tuples that must be additionally deleted to preserve referential integrity when the tuple (2,4) is deleted is:

A (3,4) and (6,4)

B (5,2) and (7,2)

✓ (5,2), (7,2) and (9,5)

D (3,4), (4,3) and (6,4)

Database-Management-System Referential-Integrity Gate-2005

Explanation:

If (2,4) is deleted and that results foreign key of value with 2 also deleted i.e., (5,2), (7,2) are also delete. And again deleting Primary keys (5), (7) means delete the foreign key value (5,7) that results (9,5) also deleted.
Totally (5,2), (7,2), (9,5) are also deleted.

Consider a database with three relation instances shown below. The primary keys for the Drivers and Cars relation are did and cid respectively and the records are stored in ascending order of these primary keys as given in the tables. No indexing is available

D: Drivers relation

did	dname	rating	age
22	Kartikeyen	7	25
29	Salman	1	33
31	Boris	8	55
32	Amoldt	8	25
58	Schumacher	10	35
64	Sachin	7	35
71	Senna	10	16
74	Sachin	9	35
85	Rahul	3	25
95	Ralph	3	53

R: Reserves relation

did	cid	day
22	101	10/10/06
22	102	10/10/06
22	103	08/10/06
22	104	07/10/06
31	102	10/11/06
31	103	06/11/06
31	104	12/11/06
64	101	05/09/06
64	102	08/09/06
74	103	08/09/06

C: cars relation

cid	cname	colour
101	Renault	blue
102	Renault	red
103	Ferrari	green
104	Jaguar	red

What is the output of the following SQL query?

```
select D.dname
from Drivers D
where D.did in (
                select R.did
                from Cars C, Reserves R
                where R.cid = C.cid and C.colour = 'red'
                intersect
                select R.did
                from Cars C, Reserves R
                where R.cid = C.cid and C.colour = 'green'
                )
```

✓ Karthikeyan, Boris

B Sachin, Salman

C Karthikeyan, Boris, Sachin

D Schumacher, Senna

Database-Management-System SQL Gate 2006-IT

Question 114 Explanation:

For colour = "Red"

did = {22, 22, 31, 31, 64}

For colour = "Green"

did = {22, 31, 74}

Intersection of Red and Green will be = {22, 31}, which is Karthikeyan and Boris.

Student (school-id, sch-roll-no, sname, saddress) School (school-id, sch-name, sch-address, sch-phone) Enrolment(school-id sch-roll-no, erollno, examname) ExamResult(erollno, examname, marks) What does the following SQL query output?

```
SELECT  sch-name, COUNT (*)
FROM    School C, Enrolment E, ExamResult R
WHERE   E.school-id = C.school-id
AND
E.examname = R.examname AND E.erollno = R.erollno
AND
R.marks = 100 AND S.school-id IN (SELECT school-id
                                  FROM student
                                  GROUP BY school-id
                                  HAVING COUNT (*) > 200)
GROUP By school-id
/* Add code here. Remove these lines if not writing code */
```

A. for each school with more than 200 students appearing in exams, the name of the school and the number of 100s scored by its students

B. for each school with more than 200 students in it, the name of the school and the number of 100s scored by its students

C. for each school with more than 200 students in it, the name of the school and the number of its students scoring 100 in at least one exam

✓ nothing; the query has a syntax error

Database-Management-System SQL Gate 2008-IT

Explanation:

If select clause consist of aggregate and non-aggregate columns, all non-aggregate columns in the select clause must appear in Group By clause. But in this Group By clause consists school-id instaed of school-name.

A relational schema for a train reservation database is given below. Passenger (pid, pname, age) Reservation (pid, class, tid)

```
Table: Passenger
pid    pname    age
-----------------
 0     Sachin    65
 1     Rahul     66
 2     Sourav    67
 3     Anil      69

Table : Reservation
pid  class  tid
---------------
 0     AC    8200
 1     AC    8201
 2     SC    8201
 5     AC    8203
 1     SC    8204
 3     AC    8202
```

What pids are returned by the following SQL query for the above instance of the tables?

```
SLECT pid
FROM Reservation ,
WHERE class ‘AC’ AND
    EXISTS (SELECT *
        FROM Passenger
        WHERE age > 65 AND
        Passenger. pid = Reservation.pid)
```

A 1, 0

B 1, 2

✓ 1, 3

D 1, 5

Database-Management-System SQL 2010

[redacted] Explanation:

Passenger:

Pid	Pname	Age
0	Sachin	65
1	Rahul	66
2	Sourav	67
3	Anil	69

Age > 65

Pid	Class	tid
0	AC	8200
1	AC	8201
2	SC	8201
5	AC	8203
1	SC	8204
3	AC	8202

Class = 'AC'

— 1, 3 Pids are returned

Given the following statements:

```
S1: A foreign key declaration can always
    be replaced by an equivalent check
    assertion in SQL.
S2: Given the table R(a,b,c) where a and
    b together form the primary key, the
    following is a valid table definition.
    CREATE TABLE S (
        a INTEGER,
        d INTEGER,
        e INTEGER,
        PRIMARY KEY (d),
        FOREIGN KEY (a) references R)
```

Which one of the following statements is CORRECT?

A S1 is TRUE and S2 is FALSE.

B Both S1 and S2 are TRUE.

C S1 is FALSE and S2 is TRUE.

✓ Both S1 and S2 are FALSE.

Database-Management-System SQL GATE 2014(Set-01)

[redacted] Explanation:

S1: False

Using a check constraint, we can have the same effect as foreign key while adding elements to the child table. But while deleting elements from the parent table the referential integrity constraint is no longer valid. So, a check constraint cannot replace a foreign key.

S2: False:

Foreign key in one table should be defined as a primary key in other table. In above table definition, table S has a foreign key that refers to field 'a' of R. The field 'a' in table S is part of the primary key and part of the key cannot be declared as a foreign key.

Student

Roll. No	Student_Name
1	Raj
2	Rohit
3	Raj

Performance

Roll. No	Course	Marks
1	Math	80
1	English	70
2	Math	75
3	English	80
2	Physics	65
3	Math	80

Consider the following relations:

```
SELECT S. Student_Name, sum(P.Marks)
      FROM Student S, Performance P
      WHERE S.Roll_No = P.Roll_No
      GROUP BY S.Student_Name
```

The number of rows that will be returned by the SQL query is _____

✓ 2

B 3

C 4

D 5

Database-Management-System SQL GATE 2015 (Set-01)

Explanation:

Output table is

Raj	310
Rohit	140

GATE 2017 question on SQL

Consider the following database table named top_scorer.

top_scorer

player	country	goals
Klose	Germany	16
Ronaldo	Brazil	15
G Muller	Germany	14
Fontaine	France	13
Pele	Brazil	12
Klinsmann	Germany	11
Kocsis	Hungary	11
Batistuta	Argentina	10
Cubillas	Peru	10
Lato	Poland	10
Lineker	England	10
T Muller	Germany	10
Rahn	Germany	10

```
SELECT ta.player FROM top_scorer AS ta
WHERE ta.goals > ALL (SELECT tb.goals
                      FROM top_scorer AS tb
                      WHERE tb.country = 'Spain')
AND ta.goals > ANY (SELECT tc.goals
                      FROM top_scorer AS tc
                      WHERE tc.country = 'Germany')
```

The number of tuples returned by the above SQL query is _____.

- ✔ 7
- B 8
- C 9
- D 10

GATE 2017 question on SQL

```
SELECT ta.player FROM top_scorer As ta
WHERE
ta.goals > ALL (SELECT tb.goals FROM
                top_scorer As tb
                WHERE tb.country = 'Spain')                    (1)
AND ta.goals > ANY (SELECT tc.goals FROM
      top_scorer As tc
      WHERE tc.country = 'Germany');                           (2)
```

In the given database table top_scorer no players are there from 'Spain'.

So, the query (1) results 0 and ALL (empty) is always TRUE.

The query (2) selects the goals of the players those who are belongs to 'Germany'.

So, it results in ANY (16, 14, 11, 10).

So, the outer most query results the player names from top_scorer, who have more goals.

Since, the minimum goal by the 'Germany' player is 10, it returns the following 7 rows.

Player
Klose
Ronaldo
G.Muller
Fontaine
Pele
Klinsman
Kocsis

GATE 2017 question on SQL

Consider the following tables T1 and T2.

T1

P	Q
2	2
3	8
7	3
5	8
6	9
8	5
9	8

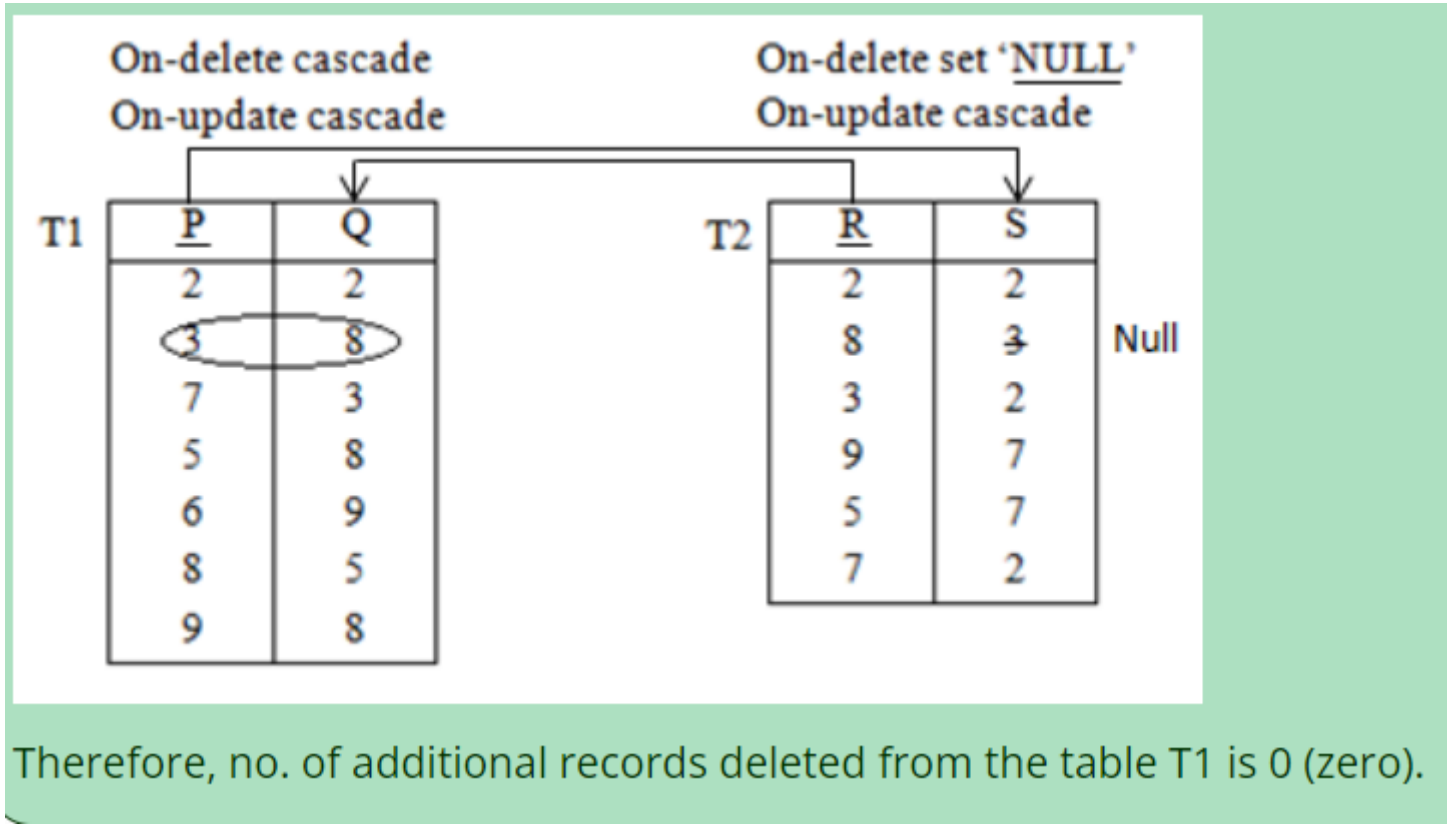
T2

R	S
2	2
8	3
3	2
9	7
5	7
7	2

In table T1, **P** is the primary key and **Q** is the foreign key referencing **R** in table T2 with on-delete cascade and on-update cascade. In table T2, **R** is the primary key and **S** is the foreign key referencing **P** in table T1 with on-delete set NULL and on-update cascade. In order to delete record ⟨3,8⟩ from table T1, the number of additional records that need to be deleted from table T1 is _____.

- ✓ 0
- B 1
- C 2
- D 3

GATE 2017 question on SQL



GATE 2017 question on SQL

Consider a database that has the relation schema EMP (EmpId, EmpName, and DeptName). An instance of the schema EMP and a SQL query on it are given below.

EMP		
EmpId	EmpName	DeptName
1	XYA	AA
2	XYB	AA
3	XYC	AA
4	XYD	AA
5	XYE	AB
6	XYF	AB
7	XYG	AB
8	XYH	AC
9	XYI	AC
10	XYJ	AC
11	XYK	AD
12	XYL	AD
13	XYM	AE

```
SELECT AVG(EC. Num)
FROM EC
WHERE (DeptName, Num)IN
      (SELECT DeptName, COUNT(EmpId) AS
                        EC(DeptName, Num)
       FROM EMP
       GROUP BY DeptName)
```

The output of executing the SQL query is _____.

- ✓ 2.6
- B 2.7
- C 2.8
- D 2.9

GATE 2017 question on SQL

The given query is

The given query is

```
SELECT AVG(EC. Num)FROM EC
    WHERE (DeptName, Num)
IN
(SELECT DeptName, COUNT(EmpId)as
          EC(DeptName, Name)
 FROM EMP GROUP BY DeptName;
```

Outer Query

Inner Query

→ We start evaluating from the inner query.

The inner query forms DeptName wise groups and counts the DeptName wise EmpIds.

→ In inner query DeptName, Count(EmpId) is the alias name DeptName, Num.

So, the output of the inner query is,

EC	
DeptName	Num
AA	4
AB	3
AC	3
AD	2
AE	1

The outer query will find the

Avg(Num) = (4+3+3+2+1)/5 = 2.6

Consider the following two tables and four queries in SQL.

Book (isbn, bname), Stock (isbn, copies)

```
Query 1:    SELECT B.isbn, S.copies
            FROM Book B INNER JOIN Stock S
            ON B.isbn = S.isbn;

Query 2:    SELECT B.isbn, S.copies
            FROM Book B LEFT OUTER JOIN Stock S
            ON B.isbn = S.isbn;

Query 3:    SELECT B.isbn, S.copies
            FROM Book B RIGHT OUTER JOIN Stock S
            ON B.isbn = S.isbn;

Query 4:    SELECT B.isbn, S.copies
            FROM Book B FULL OUTER JOIN Stock S
            ON B.isbn = S.isbn;
```

Which one of the queries above is certain to have an output that is a superset of the outputs of the other three queries?

A Query 1

B Query 2

C Query 3

✓ Query 4

Database-Management-System SQL Gate 2018

GATE 2020 question on SQL

Consider a relational database containing the following schemas.

Catalogue

sno	pno	cost
S1	P1	150
S1	P2	50
S1	P3	100
S2	P4	200
S2	P5	250
S3	P1	250
S3	P2	150
S3	P5	300
S3	P4	250

Suppliers

sno	sname	location
S1	M/s Royal furniture	Delhi
S2	M/s Balaji furniture	Bangalore
S3	M/s Premium furniture	Chennai

Parts

pno	pname	part_spec
P1	Table	Wood
P2	Chair	Wood
P3	Table	Steel
P4	Almirah	Steel
P5	Almirah	Wood

The primary key of each table is indicated by underlying the constituent fields.

```
SELECT s.sno, s.sname
FROM Suppliers s, Catalogue c
WHERE s.sno = c.sno AND
Cost > (SELECT AVG (cost)
FROM Catalogue
WHERE pno = 'P4'
GROUP BY pno);
```

The number of rows returned by the above SQL query is

GATE 2020 question on SQL

The inner query "select avg(cost) from catalogue where pno='P4' group by pno;" returns:

```
 AVG(COST)
 ------------
   225
```

The outer query "select s.sno,s.sname from suppliers s, catalogue c where s.sno=c.sno" returns:

```
SNO   SNAME
----------------------------------------
S1  M/s Royal furniture
S1 M/s Royal furniture
S1  M/s Royal furniture
S2      M/s Balaji furniture
S2      M/s Balaji furniture
S3      M/s Premium furniture
S3      M/s Premium furniture
S3      M/s Premium furniture
S3      M/s Premium furniture
```

So, the final result of the query is:

```
SN      SNAME
----------------------------------------
S2      M/s Balaji furniture
S3      M/s Premium furniture
S3      M/s Premium furniture
S3      M/s Premium furniture
```

Therefore, 4 rows will be returned by the query.

GATE 2019 question on SQL

A relational database contains two tables Student and Performance as shown below:

Student

Roll_no.	Student_name
1	Amit
2	Priya
3	Vinit
4	Rohan
5	Smita

Performance

Roll_no.	Subject_code	Marks
1	A	86
1	B	95
1	C	90
2	A	89
2	C	92
3	C	80

The primary key of the Student table is Roll_no. For the Performance table, the columns Roll_no. and Subject_code together from the primary key. Consider the SQL query given below:

```
SELECT S.Student_name, sum (P.Marks)
FROM Student S, Performance P
WHERE P.Marks > 84
GROUP BY S.Student_name;
```

The number of rows returned by the above SQL query is ____.

A 0

B 9

C 7

✓ 5

GATE 2019 question on SQL

(Executed under Oracle Express Edition)

```
SQL> SELECT S.Student_name,sum(P.Marks)
2 FROM Student S,Performance P
3 WHERE P.Marks>84
4 GROUP BY S.Student_name;
```

Student	sum(P.Marks)
Rohan	452
Vinit	452
Priya	452
Amit	452
Smita	452

GATE 2021 question on SQL

The relation scheme given below is used to store information about the employees of a company, where empId is the key and deptId indicates the department to which the employee is assigned. Each employee is assigned to exactly one department.

emp(empId, name, gender, salary, deptId)

Consider the following SQL query:

```
select deptId, count (*)
from emp
where gender = "female" and salary > (select avg(salary) from emp)
group by deptId;
```

The above query gives, for each department in the company, the number of female employees whose salary is greater than the average salary of

A employees in the department

B female employees in the company

✓ employees in the company

D female employees in the department

Database-Management-System Nested-Query

GATE 2021 question on SQL

The given SQL query is:

```
Select deptId, count(*)
from emp
where gender = "female" and
salary > (Select avg(salary) from emp)
group by deptId
```

Inner query

- The inner query will return the average salary of the emp table.
- Hence, the given query will return the deptId wise count of female employees whose salary is greater than the average salary of the emp table.

Thank You