Note: To print schema use describe table_name
To select database → use database_name
To show → show database_name, show table_name

SQL (Structured Query Language)

1- SQL is domain-specific language means unlike general purpose it can be used for relational database only.

2- SQL is declarative language. Unlike procedural language we dont have to tell 'how to do'. we have to tell 'what to do' like in CSS.
PL/SQL is procedural SQL which is not declarative.

3- i) DDL (Data Definition Language): For defining database schema
eg: create, drop, alter, truncate, comments, rename

ii) DML (Data Manipulation Language): Manipulating of data present in the database
eg: insert, update, delete, lock

iii) TCL (Transaction Control Language): Deal with transaction within the database
eg: commit, rollback, savepoint, set transaction.

iv) DQL (Data Query Language): Performing queries on data within schema objects.

v) DCL (Data Control Language): Dealing with rights, permissions of database.

3- i) grant: to grant privileges of database
ii) revoke: to revoke privileges of database.

4- Operators: Like, between, (In, Not in), (Exists/Not Exists), Conditional)

5- Clauses (distinct, order by, group by, from, having)
6- aggregate functions - MAX, MIN, SUM, COUNT.

7- Joins Σ Nested Queries
8- PL & SQL (Triggers, function, cursor, pointers).

alter

1- Add column
2- remove column
3- modify datatype
4- modify datatype length
5- add constraints
6- remove constraints
7- rename column/table

alter table student
add address varchar(30),
add id int not null;

alter table student
drop column address;

alter table student → change datatype
modify address varchar(50)
or vice → alter

alter table student
~~rename id to s_no;~~
rename column id to s_no;

alter table student
add primary key(s_no);

Update

update emp
set salary = salary * 2

delete (DML)

delete records/rows of table

delete from student; → to delete all

delete from student
where id = 2;

drop (DDL)

drop table student;

truncate

1) ~~truncate & delete~~
truncate & delete both delete rows. Unlike delete, we cant delete specific records using where. It just deletes all rows.
2) truncate is faster as it does not create log like delete so using rollback rows deleted cannot be retrieved.

Constraint

- unique
- not null
- primary key
- check → check (age > 18)
- foreign key → for referencial integrity
- default → salary int default 10000

alter table student
add constraint mycustname unique (email) check (age > 18)

alter table student
modify status bool not null default false

Normalization

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

Row level: Whole rows including the id is duplicated.
Sol'n → Set id as prim key

Column level duplicacy:

Sid	Sname	Cid	Cname	Salary
1	Ram	C_1	DBMS	30k
2	Ravi	C_2	JAVA	50k
3	Nitin	C_1	DBMS	30k
4	Amrit	C_1	DBMS	30k

Problems with these:
1- Insertion Anomaly
2- Deletion Anomaly
3- Updation Anomaly
Anomaly means problem occuring in special occasion.

Insertion: This anomaly occurs when certain attributes cannot be inserted to table without presence of other attributes.

If we wanted to insert new course C10 then we also need student details as S.id is primary so can't be null.

2- Deletion: If we delete record of student with Sid 2 then maybe ~~~~ information of course can be ~~deleted~~ deleted if the this student is the only one taking that course.

3- Updation: If we increase salary of course C_1 from 30k to 40k then all ~~~~ records with course C_1 will be updated & ~~there~~ there may be 1000s of records thus ~~~~ affecting efficiency.

One simple sol" is that we divide the tables into

Sid	Sname

Crd	Cname	Salary

First Normal form

→ Table should not contain any multivalued attribute

R_no	Name	Course
1	Sai	C/C++
2	Harsh	Java

→ Multivalued so not 1NF

R_no	Name	Course
1	Sai	C
1	Sai	C++
2	Harsh	Java

Primary key → (Roll_no, Course) composite primary key

R_no	Name	Course1	Course2
1	Sai	C	C++
2	Harsh	Java	NULL

Prim key → (R_no)

for a column having multi-value seperate it and ~~~~ with crack duplicated another table with prim key so that all other records are not

3rd way (Base + Reference table)

Roll_no	Name
1	Sai
2	Harsh

Roll_no	Course
1	C
1	C++
2	Java

2nd Normal form

1) Table must be 1st normal form
2) All the non-prime attributes should be fully functional dependent on candidate key.

Customer_ID	Store_ID	Location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

Primary key → ~~Candidate~~ (Customer_ID, Store_ID)

2 candidate keys ie (Customer_ID, Store_ID)

One non-prime attribute 'Location' is dependent upon Store_ID but not on customer_ID and for 2nd normal form, it should ~~~~ dependent on all candidate keys.

Customer_ID	Store_ID
1	1
1	3
2	1
3	2
4	3

Prim key = (Customer_ID, Store_ID)

Store_ID	Location
1	Delhi
2	Bangalore
3	Mumbai

Prim key = Store_ID

Student ID Location is dependent on all ~~the~~ candidate keys in 2nd table so 2NF
In 1st table, no non-prime attributes so already 2NF

2) There should be no partial dependency in the relation

Third Normal form (3NF)

1) Table must be 2NF
2) There ~~is~~ should be no transitive dependency

Transitive dependency means $A \to B$, $B \to C$
and ~~B is non-prime~~ B & C are non-prime

If B is prime then ~~it~~ its not transitive

Rollno	State	City
1	Punjab	Mohali
2	Haryana	Ambala
3	Punjab	Mohali
4	Haryana	Ambala
5	Bihar	Patna

Rollno → State & State → City
(State → City: non-prime)
Transitive dependency (indirect through non-prime)

Rollno	State
1	Punjab
2	Haryana
3	Punjab
4	Haryana
5	Bihar

State	City
Punjab	Mohali
Haryana	Ambala
Bihar	Patna

BCNF (Boyce Codd Normal Form)

Rollno	Name	Voter id	age
1	Ravi	K0123	20
2	Varun	M034	21
3	Ravi	K786	23
4	Rahul	D286	21

1) Must be 3NF
2) LHS of each FD ~~should~~ should be CK or super key

CK = {Rollno, Voter-ID}
FD (Functional Dependency):
- Rollno → name
- Rollno → Voterid
- Voterid → age
- Voterid → Rollno

Relation which is 3NF but not BCNF

CK = {(A,B), (A,C)}
FD = {A,B → C, C → B}
None are non-prime so 3NF but
C here is not the super key as it alone can't identify record.

Lossless and Lossy Decomposition

R

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

Decompose R_1

A	B
1	2
2	2
3	3

R_2

B	C
2	1
2	2
3	2

Find value of C if value of A=1

A & C are in diff table so we need to join

select R_2.C from R_2 Natural Join R_1 where R_1.A = 1;

Natural join: 1) Cross product of table. Total no of rows in cross-product = m×n where m is no of rows in R_1 & n is no. of rows in R_2.

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

2) Selecting those records in which common attribute/cols have same value

Here B is common column.

R'

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

→ these extra tuple aka spurious tables

Now after composing back to R' we get more tuples and is not same as original table R.

so $R \rightarrow R_1 \bowtie R_2$ is lossy decomposition as we are not getting original table back

This decomposition is lossy in terms of inconsistency

Note: To prevent this inconsistency, while decomposition, common attribute should be CK or SK of either R_1 or R_2 or both

So while decomposing R, we can use A as common attribute as A is candidate key

Condition for Lossless decomposition

1) $R_1(AB)$, $R_2(AC)$

1) $R_1 \cup R_2 = R$
   $AB \cup AC$
   $ABC \equiv ABC$

2) $R_1 \cap R_2 \neq \phi$ (means one common attribute must be there).

3) R_1 CK or R_2 CK or both

This is also the condn of 5NF.

SQL Queries and Subqueries (Part-1)

Q1- Write a SQL Query to display maximum salary from emp table

Emp

E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20,000
3	Ravi	HR	30,000
4	Nitin	MRKT	40,000
5	Varun	IT	50,000

Ans select max(salary) from emp; → 50,000

Q2- Write a SQL query to display employee name who is taking maximum salary.

select E_name from Emp → outer query
where salary =
(select max(salary) from emp) → inner query

First inner query will be executed then its result will be used.

→
select E_name from Emp
where salary = 50,000
→ Varun

Q3- Write a SQL query to display second highest salary from Emp table?

Means if we exclude max salary then we need to find max from rest.

naming this query as Qi

select max(salary) from Emp
where salary <> (select max(salary) from Emp)
↓ ↓
50,000

So first inner query will be executed then salary table is generated containing all except 50k
10k then aggregate fun is executed
20k so we get 40k.
30k
40k

2- Q. select emp name having 2nd max salary

select E_name from Emp
where salary = (qi)

Part - 3

Q- Write a query to display all the dept names along with no. of emps working in that?

select dept, count(*) from emp
group by dept

HR	2
MRKT	2
IT	1

Here we dont have to find no. of emps for one particular dept so we cant use where. We have to find for all dept so we need to use group by which will group all similar dept to one

Part - 4

Q Write a query to display all dept names where no. of emps are less than 2.

It seems that where clause is to be used but where for grouped up records if we have to display specific ones, we have to use 'having' clause instead of 'where'.

qi = select dept from Emp group by dept having count(*) < 2;

2) tell emp name whose dep having no. of employees less than 2.

select E_name from Emp
where dept In (qi)

In is used bcz there can me multiple departments

name In (a,b,c) is equall to name = a or name = b or name = c.

Part - 5

Q- Display highest salary dept wise & name of emp taking that salary.
First of all we need to find highest salary dept wise

qi = select max(salary) from emp group by dept.

Select E_name, salary from emp
where salary In (qi);

Part-6

Q- Detail of emp whose address is either Delhi or Chd or Pune

Emp

Eid	Ename	Address
1	Ravi	Chd
2	Varun	Delhi
3	Nitin	Pune
4	Robby	Bangalore
5	Ammy	Chd

Project

Eid	Pid	Pname	Location
1	P_1	IOT	Bangalore
5	P_2	Big Data	Delhi
3	P_3	Retail	Mumbai
4	P_4	Android	Hyderabad

select * from Emp where Address In ('Delhi', 'Chd', 'Pune')

Its reverse is NOT IN

Part-7

Find the name of emps who are working on project

First lets find the Eids who are working on project

q1= select Eid from Project

- select Ename from Emp where Eid In (q1);

Also it can be done by

select Ename from emp, project where ~~emp.Eid~~ emp.E_id = project.E_id;

Part-8 (Exists/Not Exists)

Q- Find the detail of Emp who is working on atleast one project?

Exist/Not exist is for correlated nested query instead of normal nested query.

In normal nested query, first inner query is executed & its output is used in outer query.

In correlated nested query, each row of output is checked with inner query output which is true/false for exists/not exists.

Unlike normal nested, inner query is executed multiple times depending upon number of rows in tables of outer query whereas in normal, inner query is executed only 1 time.

select * from Emp where Eid exists (select Eid from Project where Emp.Eid = Project.Eid)

exists will return true if inner query produces non-empty output else false.

eg -
for 1st row of Emp
inner query → 1
1 is returned so exist, return true.
2nd row
nothing returned so exist return false
as 2 not equal to any Eid in proj
for 3rd, 4th & 5th, 3, 4 & 5 returns.

In correlated nested query, as inner query is executed again & again with each row of outer query. so table used in outer query can be used in inner query.

In normal nested its not possible unless included in from clause.