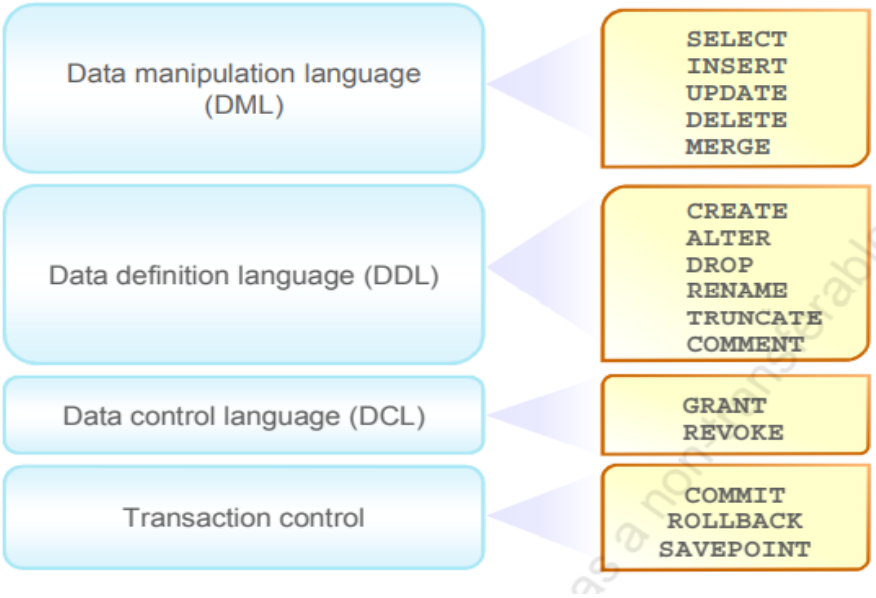
SQL Commands



DDL Commands: deals with schema or the structure of the table.

1. CREATE command
    a. Used to create tables and databases.
    b. CREATE TABLE TABLE_NAME (add attributes and its type).

```
CREATE TABLE student(
roll_no int,
sname varchar(20),
total_marks int
);
```

2. ALTER command
    a. Used to modify the schema of the table.
    b. ALTER TABLE TABLE_NAME [add || rename || modify || drop] column_name

```
alter table student add age int;

alter table student modify sname varchar(30);

alter table student rename column sname to student_name;

alter table student drop column age;
```

3. DROP command
    a. Used to delete or drop the schema.

b. DROP TABLE table_name

4. TRUNCATE command
    a. Used to delete records of the table, but no structure.
    b. TRUNCATE TABLE table_name;

```
DROP table student;
TRUNCATE table student;
```

DML Commands: deals with records of the table by inserting, updating and deleting.

1. INSERT INTO
    a. Used to insert values in the table.
    b. INSERT INTO table_name VALUES (v1, v2, v3)
    c. INSERT INTO table_name (c1,c2,c3) VALUES (v1,v2,v3)
2. UPDATE
    a. Used to modify or update the records based on conditions.
    b. UPDATE TABLE student set column_name where conditions.
3. DELETE
    a. Used to delete all records or desired records based on conditions
    b. DELETE * FROM table_name WHERE conditions

```
INSERT INTO student VALUES (1, 'Harshit', 450);
INSERT INTO student (roll_no, sname, total_marks) VALUES (1, 'Harshit',
450)
UPDATE TABLE student set total_marks = 400 WHERE roll_no = 1;
DELETE * FROM student WHERE roll_no = 1;
```

Table 2: employees

```
create table employees(
eid int,
ename varchar(30),
job_id int,
salary int,
commission int,
dept_id int);
```

```
—-Inserting records


insert into employees values(4,'Shivam',2,60000,2,3);
insert into employees values(3,'Thakur',1,75000,3,3);
insert into employees values(2,'Anuj',2,70000,1,2);
insert into employees values(1,'Harshit',1,80000,2,3);
```

SELECT

- Used to display all records from the table.
- Used to display all records for desired columns.

```
SELECT * FROM employees;
SELECT eid,ename,salary from employees;
```

SELECT with DISTINCT

- Used to remove duplicate values for columns

```
SELECT DISTINCT job_id from employees;
SELECT DISTINCT job_id,dept_id from employees;
```

SELECT with Arithmetic expressions

- If you need to modify the way in which data is displayed.
- If you want to perform calculations
- Operators can be used to perform expressions

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

```
select * from employees;
select eid,salary,salary+500 from employees;
select eid,salary,commission,0.01*commission*salary from employees;
```

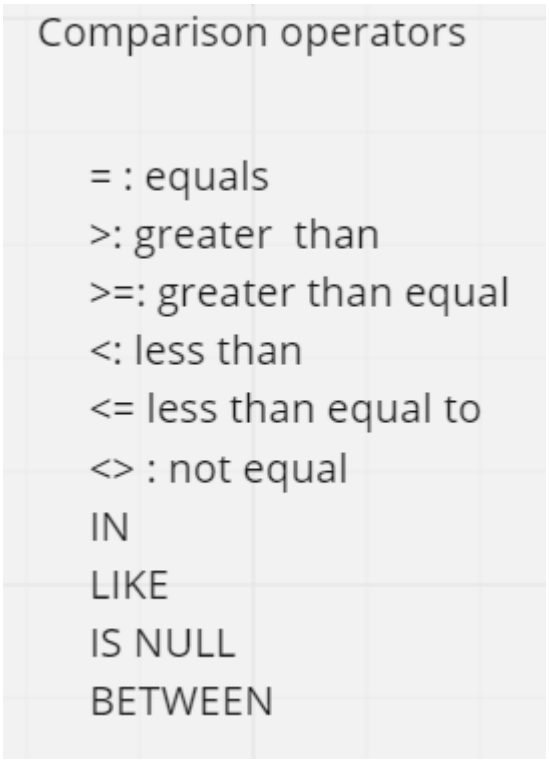
Column alias

- Used to rename columns to meaningful names while querying the result.
- as is optional
- Use double quotes if the name has space.
- If there is no space, we can write it as is.
- We can give name to arithmetic expressions.

```
select ename as employee_name from employees;
select ename  employee_name from employees;
select salary+10 as "New salary" from employees;
```

Operators

- Used in implementing conditions in where clause
- Comparison and logical operators

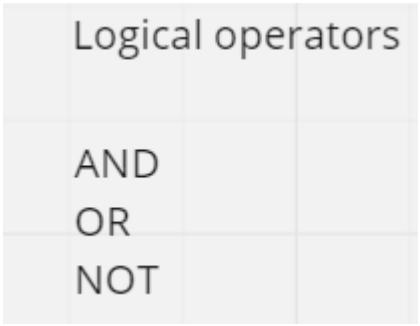



```
Comparison operators


-- =
select * from employees where eid=1;
```

```
select * from employees where ename='Thakur';

-- > >= < <=
select * from employees where salary>60000;
select * from employees where salary<=70000;

-- <>
select * from employees where eid<>2;

-- IN
select * from employees where salary in (60000,70000);

-- like
select * from employees where ename like '____';
select * from employees where ename like 'T%';

-- Between and
select * from employees where salary between 60000 and 75000;

-- is NULL
select * from employees where dept_id is NULL;
```

Logical operators

- Used to provide multiple conditions in query




```
select * from employees where salary>60000 and dept_id=3;
select * from employees where salary>60000 or dept_id=3
select * from employees where NOT salary>70000;
```

Character functions

Function	Result
LOWER(SQL Course)	sql course
UPPER(SQL Course)	SQL COURSE
INITCAP(SQL Course)	Sql Course

Function	Result
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6

```
select * from dual;

select lower('HELLO') from dual;

select upper('hello') from dual;

select initcap('hello world') from dual;


select concat('hello','world') from dual;


select substr('helloworld',1,5) from dual;
select substr('helloworld',-5,5) from dual;
 select length('hello') from dual;

select instr('hello','e') from dual;
```

Number functions

```
 create table demo(
 eid int,
 ename varchar(20),
 profit number(6,4));

 insert into demo values(1,'anuj',54.463);
 insert into demo values(2,'amit',26.421);
 insert into demo values(3,'shibu',18.466);
 insert into demo values(4,'thakur',21.872);
```

```
select * from demo;
```

Function	Result
ROUND(45.926, 2)	45.93
TRUNC(45.926, 2)	45.92
CEIL (2.83)	3
FLOOR (2.83)	2
MOD (1600, 300)	100

```
select round(profit,2) from demo;
select trunc(profit,1) from demo;
select floor(profit) from demo;
select mod(profit,2) from demo;
```

NVL: NULL functions

```
select * from employees;

select eid,ename,NVL(commission,0)+2 from employees;
```

Table 3:

```
create table employee(
eid int,
ename varchar(20),
job_id int,
salary int);


insert into employee values(1,'Harshit',1,2000);
insert into employee values(2,'Anuj',2,2000);
insert into employee values(3,'Shivam',1,3000);
insert into employee values(4,'Sukrit',3,5000);
insert into employee values(5,'Thakur',3,8000);
```

Case expression

-- Question 1
/* if job_id = 1 increase salary by 300
   if job_id = 2 increase salary by 200
   if job_id = 3 increase salary by 100
*/

```
select job_id,salary,(case
                        job_id when 1 then salary+300
                               when 2 then salary+200
                               when 3 then salary+100
                               else salary
                        end)"Modified salary" from employee;
```

-- Question 2

-- sal<1000 => less paid
-- 1000<=sal<=3000 =>low salary
-- 3001<=sal<=5000 => medium
-- 5001<=sal<=8000 =>high

-- else
-- overpaid

```
select job_id,salary,(case
                        when salary<1000 then 'less paid'
                               when salary>=1000 and salary<=3000 then 'low'
                               when salary>=3001 and salary<=5000 then 'medium'
                               when salary>=5001 and salary<=8000 then 'high'
                               else 'over paid'
                       end)"Modified salary" from employee;
```

Aggregate functions

```
--sum()

select sum(salary) as "sum of salary" from employee;

-- max()
select max(salary) as "max of salary" from employee;
```

```sql
-- min()
select min(salary) as "min of salary" from employee;

-- count()
select count(salary) as "count of salary" from employee;


--AVG()
select AVG(salary) as "Average of salary" from employee;
```

Order by

```sql
-- Order by clause single column

select * from employee order by salary asc;
select * from employee order by job_id desc;

-- Order by clause multiple column
select * from employee order by job_id,salary;
select * from employee order by job_id desc,salary asc
```

Fetch and offset

```sql
select * from employee fetch first 2 rows only

select * from employee offset 2 rows
fetch first 2 rows only;

select * from employee offset 3 rows;
```

Group by

```sql
create table student(
sid int,
sname varchar(20),
branch varchar(20),
Ctc int);

insert into student values(1,'Harshit','CSE',800000);
insert into student values(2,'Anuj','ME',600000);
```

```
insert into student values(3,'Tiwari','CSE',900000);
insert into student values(4,'Shivam','ME',500000);
insert into student values(5,'Thakur','ME',400000);
```

```
select * from student;

-- Write SQL query to count number of students in each department

select branch,count(*) from student group by branch;


-- write SQL query to cal avg package in each department

select branch,avg(ctc) from student group by branch;
```

Having clause

```
-- display branch having 1 student

select branch,count(*) from student group by branch having count(*)=1;


-- display avg ctc for CSE and ME

select branch,avg(ctc) from student group by branch having branch in
('CSE','ME');

-- display avg ctc for CSE and ME avg ctc > 500000

select branch,avg(ctc) from student where branch in ('CSE','ME') group
by branch having avg(ctc)>500000;
```

Joins

```
create table student(
sid int,
sname varchar(20),
branch_id int);

create table branch(
branch_id int,
bname varchar(20));
```

```
insert into student values(1,'Harshit',1);
insert into student values(2,'Shivam',2);
insert into student values(3,'Anuj',2);
insert into student values(4,'Thakur',2);
insert into student values(5,'Sukrit',3);
insert into student values(6,'Amit',4);
insert into student values(7,'Aditya',1);

insert into branch values(1,'CSE');
insert into branch values(2,'ME');
insert into branch values(3,'EE');
insert into branch values(4,'ECE');


select * from student;

select * from branch;
```

Cross Join

```
select * from student stu,branch br;
select stu.sname, br.branch_id from student stu,branch br;
select stu.sname, br.branch_id from student stu cross join branch br;
```

Inner Join

```
select * from student cross join branch where
student.branch_id=branch.branch_id;

select * from student s inner join branch b on s.branch_id=b.branch_id;
```

Left outer Join

```
select * from student s left outer join branch b on
s.branch_id=b.branch_id;
```

Right outer Join

```
select * from student s right outer join branch b on
s.branch_id=b.branch_id;
```

Full outer Join

```
select * from student s full outer join branch b on
s.branch_id=b.branch_id;
```