Computer Language

Foundation

database

```
-- drop existing database
drop database MyDB;

-- create new database
create database MyDB;

-- use the database for further operations
use MyDB;
```

DDL (Data Definition Language)

create table

```
create table department (
    id integer primary key,
    name varchar(20)
);

create table employee (
    id integer identity(1, 1),
    name varchar(20),
    departmentId int,
    managerId int,
    primary key (id),
    foreign key (departmentId) references department (id) on delete cascade
);

create table addresses (
    id integer primary key identity(1, 1),
    city varchar(20),
    state varchar(20),
    country varchar(20),
    zipcode varchar(6)
);

create table car (
```

```
    id int primary key identity(1, 1),
    model varchar(20),
    company varchar(20),
    deleted int default 0
);
```

drop table

```
drop table employee;
```

alter table

```
-- add new column named company
alter table employee add company varchar(20);

-- add new column named salary
alter table employee add salary int;
```

truncate table

```
-- remove all the rows inside the table
-- resets the rowid (hidden column used to maintain the identity)
truncate table employee;
```

rename table

```
exec sp_rename employee, newEmployee;
```

DML (Data Manipulation Language)

insert values

```

-- department
insert into department (id, name) values (10, 'Dev');
insert into department (id, name) values (20, 'QA');
insert into department (id, name) values (30, 'DevOps');
insert into department (id, name) values (40, 'Admin');

-- this will allow you to insert multiple records at the same time
insert into department (id, name) values
    (10, 'Dev'), (20, 'QA'), (30, 'DevOps'), (40, 'Admin');

-- employees
insert into employee (name, departmentId, managerId, salary) values ('Amit',
10, 3, 20);
insert into employee (name, departmentId, managerId, salary) values
```

```
('Harshal', 20, 0, 25);
insert into employee (name, departmentId, managerId, salary) values
('Shrikant', 10, 0, 55);
insert into employee (name, departmentId, managerId, salary) values
('Chhaya', 20, 2, 23);
insert into employee (name, departmentId, managerId, salary) values
('Akash', 10, 1, 40);
insert into employee (name, departmentId, managerId, salary) values
('Harprit', 10, 1, 27);

-- this will not execute as the department 80 does not exist
-- insert into employee (name, departmentId, managerId, salary) values
('test', 80, 1, 27);



-- car
insert into car (model, company) values ('carens', 'kia');
insert into car (model, company) values ('i20', 'hyundai');
```

update records

```
-- employee
update employee set managerId = 0 where id = 4;

-- car
-- simulate the delete operation on car
update car set deleted = 1 where id = 1;
```

delete records

```
-- delete all employees
delete from employee;
```

DQL (Data Query Language)

simple query

```
-- get all departments
select id, name from department;

-- get all employees
select id, name, departmentId, managerId from employee;

-- get all cars
select id, model, company, deleted from car;
```

filtering data rows

```
-- get all the active (not deleted) car
select id, model, company, deleted from car where deleted = 0;
```

using wildcards

```
-- % used to specify any character any times
-- A% -> Amit, Akash
-- Am% -> Amit

-- _ used to specify a single character
-- b_d -> bid, bad

-- find employees having name starting with A
select id, name, managerId from employee where name like 'A%';

-- find employees having sh in their names
select id, name, managerId from employee where name like '%sh%';

-- find employees having name starting with A OR having sh in their names
select id, name, managerId from employee where name like 'A%' or name like
'%sh%';

-- find employees having name starting with A AND having sh in their names
select id, name, managerId from employee where name like 'A%' and name like
'%sh%';
```

alias column

```
-- rename name column as EmployeeName in the result set
select id as EmployeeId, name as EmployeeName, managerId from employee;
```

alias table

```
-- create an alias for employee as e
select e.id, e.name, e.managerId from employee e;
```

aggregate functions

```
-- get the count of all records
select count(*) as EmployeeCount from employee;

-- get the sum of
select sum(salary) as TotalSalary from employee;

-- get minimum
```

```
select min(salary) as LowestSalary from employee;

-- get maximum
select max(salary) as HighestSalary from employee;
```

order by

```
-- get all employees sorted by their salaries in ascending order
select id, name, departmentId, salary from employee order by salary;
select id, name, departmentId, salary from employee order by salary ASC;

-- get all employees sorted by their salaries in ascending order
select id, name, departmentId, salary from employee order by salary DESC;
```

between

```
-- select employees from department between 10 20
select id, name, departmentId, salary from employee where departmentId
between 10 and 20;
```

in

```
-- select employees from department 10, 20, 30, 40
select id, name, departmentId, salary from employee where departmentId in
(10, 20, 30, 40);
```

group by

```
-- without filtering
select count(*) as EmployeeCount, departmentId from employee group by
departmentId;
select sum(salary) as TotalSalary, departmentId from employee group by
departmentId;
select min(salary) as LowestSalary, departmentId from employee group by
departmentId;
select max(salary) as HighestSalary, departmentId from employee group by
departmentId;


-- with filtering
select count(*) as EmployeeCount, departmentId from employee group by
departmentId having departmentId = 10;
select sum(salary) as TotalSalary, departmentId from employee group by
departmentId having departmentId = 10;
select min(salary) as LowestSalary, departmentId from employee group by
departmentId having departmentId = 10;
```

```
select max(salary) as HighestSalary, departmentId from employee group by
departmentId having departmentId = 10;
```

Joins

setup

```
create table department (
    id integer primary key,
    name varchar(20)
);

create table employee (
    id integer identity(1, 1),
    name varchar(20),
    departmentId int,
    managerId int,
    salary int,
    primary key (id)
);

insert into department (id, name) values
    (10, 'Dev'), (20, 'QA'), (30, 'DevOps'), (40, 'Admin');

insert into employee (name, departmentId, managerId, salary) values ('Amit',
10, 3, 20), ('Harshal', 20, 0, 25), ('Shrikant', 10, 0, 55), ('Chhaya', 20,
2, 23), ('Akash', 10, 1, 40), ('Harprit', 10, 1, 27), ('Suresh', 80, 1, 25),
('Suresh', 70, 1, 29);
```

inner join

- getting the common records from both the table

```
-- get employee information along with  the department name
select e.id, e.name as EmployeeName, e.departmentId, e.salary, d.name as
DepartmentName from employee e
inner join department d on e.departmentId = d.id;
```

left outer join

- common records + all records from left table;

```
select e.id, e.name as EmployeeName, e.departmentId, e.salary, d.name as
DepartmentName from employee e
left outer join department d on e.departmentId = d.id;
```

right outer join

- common records + all records from right table;

```
select e.id, e.name as EmployeeName, e.departmentId, e.salary, d.name as
DepartmentName from employee e
right outer join department d on e.departmentId = d.id;
```

full outer join

```
select e.id, e.name as EmployeeName, e.departmentId, e.salary, d.name as
DepartmentName from employee e
full outer join department d on e.departmentId = d.id;
```

cross join

```
select e.id, e.name as EmployeeName, e.departmentId, e.salary, d.name as
DepartmentName from employee e
cross join department d;
```

self join

```
select e.id, e.name as EmployeeName, e.departmentId, e.salary, e1.name as
ManagerName from employee e
left outer join employee e1 on e.managerId = e1.id
```

union

- combination of first and second query
- requirement: both the queries must produce similar projection

```
select id, name from employee
union select id, name from department
```

constraints

check

```
create table users (
    id integer primary key identity(1, 1),
    name varchar(20),
    age int check (age >= 18)
);

-- insert records

-- inserted
insert into users (name, age) values ('user1', 20);
```

```
-- error while inserting this record as age < 20
insert into users (name, age) values ('user2', 10);
```

not null

```
create table users (
    id integer primary key identity(1, 1),
    name varchar(20),
    email varchar(20) unique,
    password varchar(20) not null
);
```

unique

```
create table users (
    id integer primary key identity(1, 1),
    name varchar(20),
    email varchar(20) unique
);
```

Views

- create a temporary table
- can not pass a parameter to a view

```
-- create a view
create view employee_with_manager as select e.id, e.name as EmployeeName,
e.departmentId, e.salary, e1.name as ManagerName from employee e left outer
join employee e1 on e.managerId = e1.id

-- get data from view
select * from employee_with_manager;

-- create view to get employee details
create view employee_details as select * from employee;
select * from employee_details;
```

Stored Procedure

without parameters

- faster than executing the queries

```
-- create procedure
create procedure sp_employee_details
```

```
as
begin
    select id, name, departmentId, salary from employee;
end

-- call procedure
exec sp_employee_details
```

with parameters

```
-- create procedure
create procedure sp_employee_details_with_id
    @id int = null
as
begin
    select id, name, departmentId, salary from employee
    where id = @id;
end

-- call procedure
exec sp_employee_details_with_id @id = 1
```