SQL Queries with Explanation

Schema Creation:

```
-- show databases;
create database testdb;
use testdb;
CREATE TABLE Student
(
sid INT NOT NULL,
name VARCHAR(255) NOT NULL,
address VARCHAR(255) NOT NULL,
phone VARCHAR(255) NOT NULL,
email VARCHAR(255) NOT NULL,
PRIMARY KEY (sid)
);

CREATE TABLE Course
(
cid INT NOT NULL,
name VARCHAR(255) NOT NULL,
price NUMERIC(2) NOT NULL,
description VARCHAR(255) NOT NULL,
PRIMARY KEY (cid)
);

CREATE TABLE Enrollment
(
Amount NUMERIC(2) NOT NULL,
date DATE NOT NULL,
eid INT NOT NULL,
sid INT NOT NULL,
cid INT NOT NULL,
PRIMARY KEY (eid),
FOREIGN KEY (sid) REFERENCES Student(sid),
FOREIGN KEY (cid) REFERENCES Course(cid)
);
```

CRUD Operations:

```
show databases;
create database IF NOT EXISTS mindsprint;
use mindsprint;
create table employee (id INT primary key,name varchar(100), department
varchar(100),
salary decimal(10,2));
-- Describe Table
describe employee;

-- Alter Table to add new Column
alter table employee
add column position varchar(100);
-- Insert single record using all fields
insert into employee values (1,'alex','I.T',45678,'Jr. Developer');
-- Insert record by using field name
insert into employee (id,position,department,salary,name)
values (2, 'Sr.Developer','I.T.',87690,'John');
-- If you skip any field it will take null
insert into employee (id,position,salary,name)
values (3, 'Sr.Developer',87690,'Devid');
-- If you wnat to insert multiple values
insert into employee (id,position,department,salary,name)
values
(4, 'Designer','I.T.',2500,'bob'),
(5, 'Manager','Sales',90000,'Catherine'),
(6, 'Manager','SAP',10000,'Jack');
-- To verify
select * from employee;

-- update data
update employee set salary=25000 where id=4;

-- delete data
delete from employee where id=3;

-- truncate data (means it recreate the table)
truncate table employee;

-- delete the entire table
```

```
drop table employee;

-- Let's cretae table using extras
create table employees(id INT primary key auto_increment,
name varchar(100), position varchar(100),
department varchar(100),salary decimal(10,2));

-- describe again
describe employees;
-- insert
insert into employees (name,position,department,salary)
values
('Alex','Jr.Developer','IT',45000),
('Bob','Sr.Developer','IT',90000),
('Catherine','DBA','IT',67000),
('Devid','Financial Advisor','Finance',45000),
('Jack','Sr. Manager','Sales',55000);
-- verify
select * from employees;
-- single column retrival
select name from employees;
select name,position from employees;
-- Select UseCases
select 1+1 as 'result';
select concat('John',' ','Doe') as fullname;
select now() today;
select upper('Hello World') as 'Upper Case';

-- Where clause
-- retrive data of employees whose salary is more than 50000
-- retrive data of employees whose salary is equal to 100000
-- retrive data of employees whose id is 4
-- retrive data of employee whose name is alex
-- retrive data of employees whose salary is between 30000 to 70000
-- retrive employees list from department IT
```

Joins and Transaction Examples

```
-- PK and FK Constraints
create database sample;
use sample;
```

```
-- create Department table
create table departments (id INT primary key, name varchar(100) not null);
-- Insert Some Sample Records
INSERT INTO departments (id, name)
VALUES
(1, 'Sales'),
(2, 'R&D'),
(3, 'Marketing'),
(4, 'Finance'),
(5, 'Human Resources');
-- Verify the inserted Data
select * from departments;
-- Create Employee Table which makes many to one relationship between
department and employees
-- one department can have many employees
create table employees
(id int primary key,
name varchar(100) not null,
position varchar(100) not null,
salary decimal(10,2),
department_id INT,
foreign key (department_id)
references Departments(id)
);
describe employees;
INSERT INTO employees
(id, name, position, salary, department_id)
VALUES
(1, 'John Doe', 'Manager', 75000.00, 1),
(2, 'Jane Smith', 'Developer', 65000.00, 2),
(3, 'Emily Johnson', 'Designer', 60000.00, 3),
(4, 'Michael Brown', 'Analyst', 70000.00, 4),
(5, 'Linda Green', 'Manager', 75000.00, 1),
(6, 'James White', 'Developer', 65000.00, 2),
(7, 'William Black', 'Developer', NULL, 2),
(8, 'Mary Blue', 'HR', 50000.00, 5);
select * from employees;
-- Inner join to take the common details from both tables
select e.id,e.name,e.position Designation ,e.salary,d.name 'Department Name'
from employees e
```

```sql
on e.department_id = d.id;

-- lets insert one record in employee table without department_id
insert into employees (id,name,position,salary,department_id)
values (9,'Test User','Testing',45672,NULL);

-- Left join to take the common details as well as values from left table
select e.id,e.name,e.position Designation ,e.salary,d.name 'Department Name'
from employees e
left join departments d
on e.department_id = d.id;

-- let's insert one record in department
insert into departments values (6,'IT');

-- Right join to take the common details as well as values from Right table
select e.id,e.name,e.position Designation ,e.salary,d.name 'Department Name'
from employees e
left join departments d
on e.department_id = d.id;

-- execute Full Outer Join
select e.id,e.name,e.position Designation ,e.salary,d.name 'Department Name'
from employees e
left join departments d
on e.department_id = d.id
union
select e.id,e.name,e.position Designation ,e.salary,d.name 'Department Name'
from employees e
right join departments d
on e.department_id = d.id;

-- CROSS Join
select e.id EmployeeID, e.name EmployeeName, d.id DepartmentId, d.name
DepartmentName
from employees e
cross join departments d;


select count(*) "No of Employees", department_id
```

```sql
group by department_id;

-- Find our Total employees based on Department Name
select dep.name "Department Name", count(*) "total employees"
from employees emp
join departments dep
on emp.department_id=dep.id
group by dep.name;

-- Total sum of salary based on department
select dep.name "Department Name", sum(emp.salary) "total salary"
from employees emp
join departments dep
on emp.department_id=dep.id
group by dep.name;

-- Having Clause
select dep.name "Department Name", sum(emp.salary) 'Total Salary'
from employees emp
join departments dep
on emp.department_id=dep.id
group by dep.name
having `Total Salary`>100000;

-- List the department which is having more than 2 employees
select dep.name "Department Name", count(*) "total employees"
from employees emp
join departments dep
on emp.department_id=dep.id
group by dep.name
having `total employees`>2;

-- RollUp
select dep.name "Department Name", sum(emp.salary) "total salary"
from employees emp
join departments dep
on emp.department_id=dep.id
group by (dep.name) with rollup;

-- ***********TRANSACTION EXAMPLE *****************
```

```
account_id VARCHAR(10) PRIMARY KEY,
account_name VARCHAR(100),
balance DECIMAL(10, 2)
);

INSERT INTO Accounts (account_id, account_name, balance) VALUES
('A001', 'Alice', 1000.00),
('A002', 'Bob', 1500.00),
('A003', 'Charlie', 2000.00);

select * from Accounts;

Begin;
update Accounts set balance= balance+100 where account_id='A001';
update Accounts set balance= balance-100 where account_id='A002';
commit;
select * from Accounts;

Begin;
update Accounts set balance= balance+100 where account_id='A001';
update Accounts set balance= balance-100 where account_id='A002';
rollback;
-- **** Save Point****
Begin;
update Accounts set balance= balance+100 where account_id='A001';
savepoint sp1;
update Accounts set balance= balance-100 where account_id='A002';

rollback to sp1;
commit;
select * from Accounts;

DELIMITER // -- change the termination symbol
CREATE procedure getAllAccounts() -- create the procedure
begin
select * from Accounts;
end // -- last line of procedute which ends with //
DELIMITER ; -- changes the termination symbol to ;

call getAllAccounts();
```