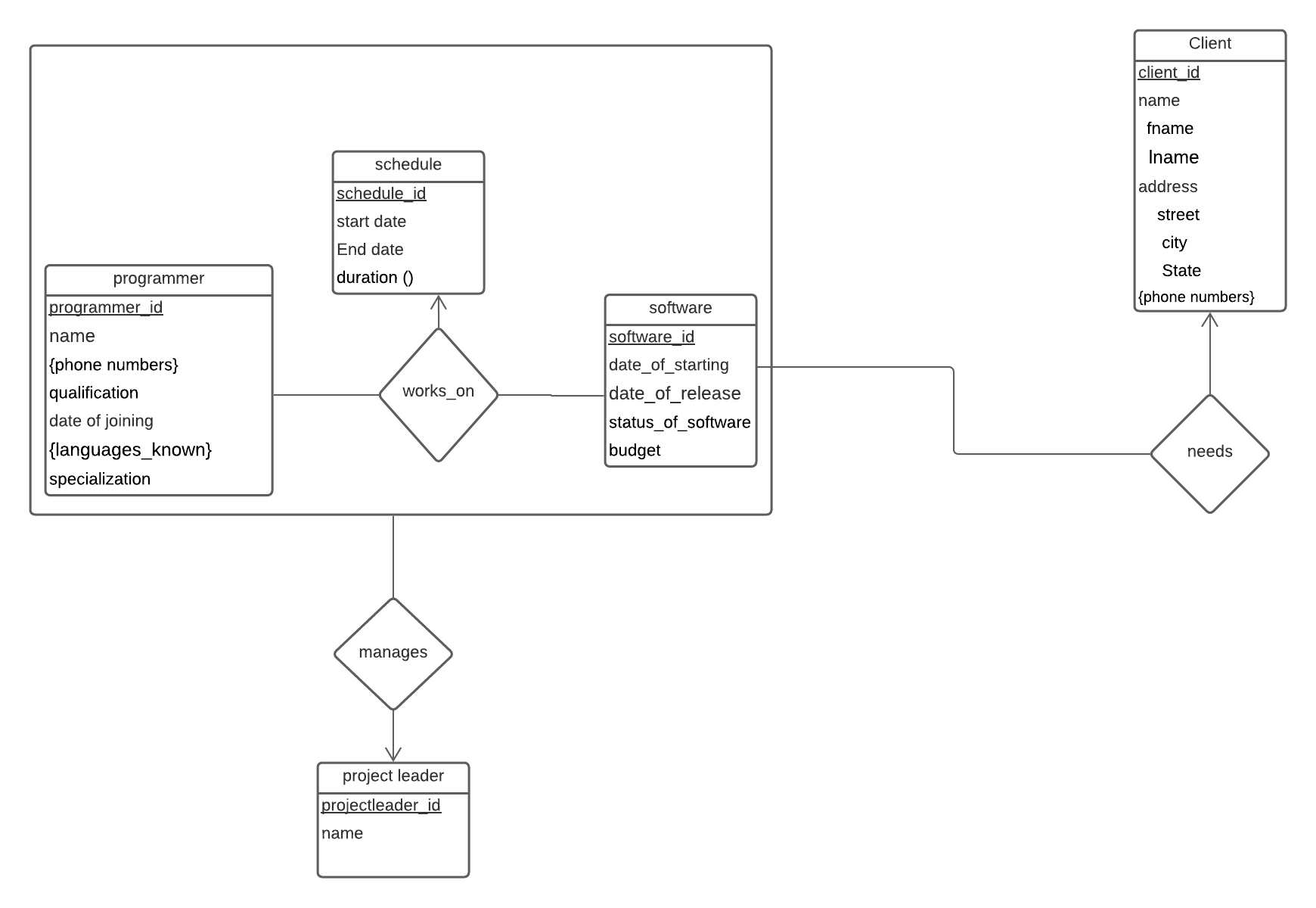
**E-HUB**

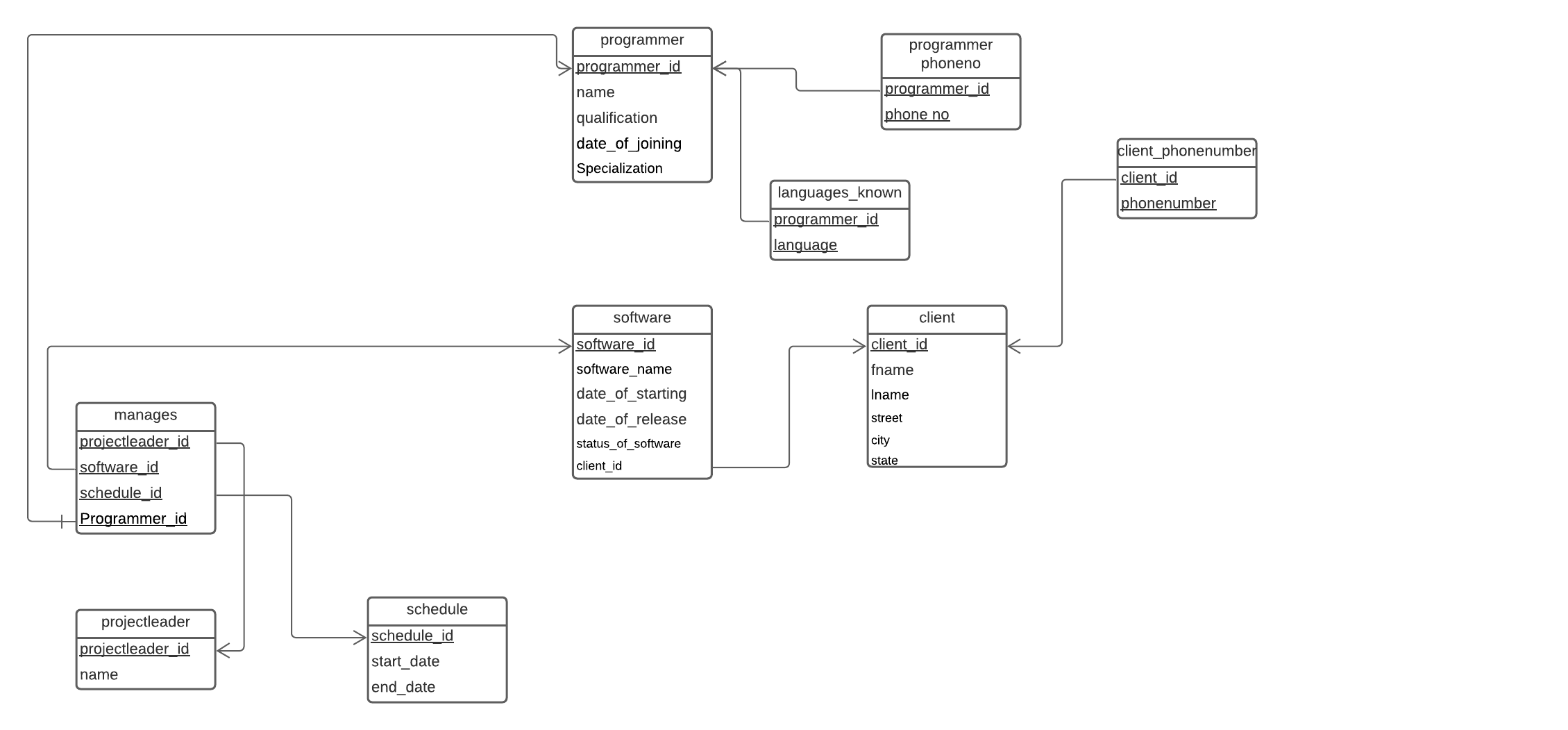
**Abstract:**

E-hub is a software company that provides various types of software solutions to clients across India. It has recruited various programmers for software development. Each programmer is identified by the id, date of joining, experience, qualification, specialization, programming\_languages\_known. Specialization signifies the programming language for which he is most specialized in. The company may develop more than one software for one client. Each client is identified by his id, name, address, phone numbers. The details of software developed for the client such as date\_of\_commencement, date\_of\_release, status\_of\_software, etc. are also maintained. Each software may be developed by more than one programmer with one project leader. The software's working by a programmer will have schedule\_id, start\_date, end\_date.

**Er Diagram:**



**Schema Diagram:**



**Relational schema:**

We have 5 entities namely -Client,Programmer,Software,Project leader,Schedule

Composite attributes – name,address(client entity)

Multivalued attributes -phone\_numbers,languages\_known(programmer entity),

phone\_numbers(client entity)

Derived attributes – duration(schedule entity)

The strong entity set reduces to schema with the same attributes.

The composite attributes get simplified by creating separate attributes.

Programmer(programmer\_id, name , phone\_numbers, qualification,specialization date\_of\_joining)

Languages\_known(programmer\_id,language)

Programmer\_phoneno(programmer\_id,phone\_no)

Client(client\_id,fname,lname,street,city,state)

Client\_phonenumbers(client\_id,phonenumber)

Software(software\_id,date\_of\_starting,date\_of\_release,status\_of\_software,client\_id)

Project\_leader(projectleader\_id,name)

Manages(projectleader\_id,software\_id,schedule\_id,programmer\_id)

Schedule(schedule\_id,start\_date,end\_date)

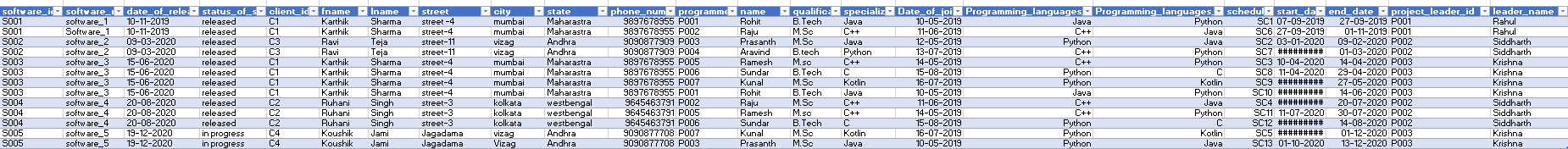
**DBMS PROJECT REVIEW-02**

**1NF**:

● A relation is said to be in 1NF if and only if all the attributes are having atomic domains(cannot be decomposed into smaller pieces).

* Address is a composite attribute which contains street, city, state. So, we will split address into street,city,state.
* Programming\_languages\_known is multivalued attribute. So, we will create two separate columns as programming\_languages\_known1 and programming\_languages\_known2
* Client name is composite attribute contains fname, lname. So, we will split client name into fname and lname.

After 1nf:



**Functional Dependencies:**

Software\_id -> software name, date\_of\_release, status\_of\_software, projectleader\_id, name, client\_id, fname, lname, street, city, state

ClientID -> fname, lname,street,city,state

Programmer\_id -> name, specialization, qualification, date\_of\_joining, programming\_languages\_known1, programming\_languages\_known2

Software\_id,programmer\_id -> schedule\_id, start\_date, end\_date

Schedule\_id -> start\_date, end\_date

Projectleader\_id -> name

Candidate key is software\_idprogrammer\_id

**Second Normal Form(2NF):**

● A relation is said to be in 2NF if and only if it is in 1NF.

● There should not be any partial dependency present in the relation.

*P.D = proper subset of CK -----> non-prime attributes*

In 1nf table the partial dependencies are software\_id -> software\_name, date\_of\_release, status\_of\_software, projectleader\_id, name, client\_id, fname, lname, street, city

Programmer\_id -> name, specialization, qualification, date\_of\_joining, programming\_languages\_known1, programming\_languages\_known2

In 1nf table the full dependency is software\_id, programmer\_id -> schedule\_id, start\_date, end\_date

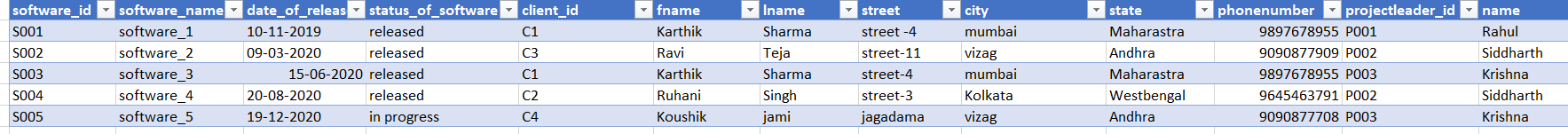
**After Decomposing tables in 2nf the tables are**:

**SoftwareClient(2NF):**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Software\_id | software name | Date\_of\_release | Status\_of\_software | Projectleader\_id | name | Client\_id | fname | lname | street | city | State |

In this table transitive dependencies are present. They are projectleader\_id ->name

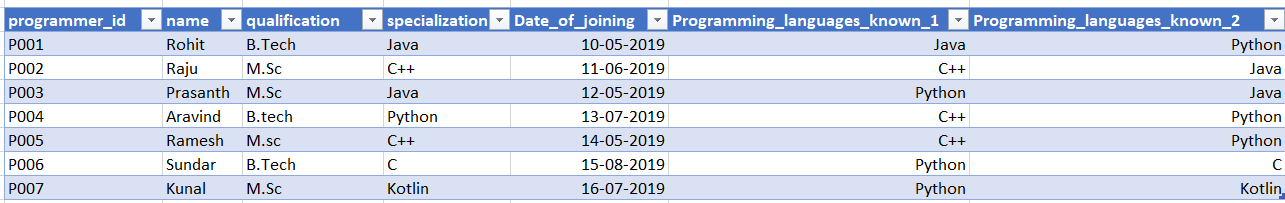
Client\_id->fname,lname,street,city,state



**Programmer(3NF):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Programmer\_id | name | specialization | qualification | Date\_of\_joining | Programming\_languages\_known1 | Programming\_languages\_known2 |

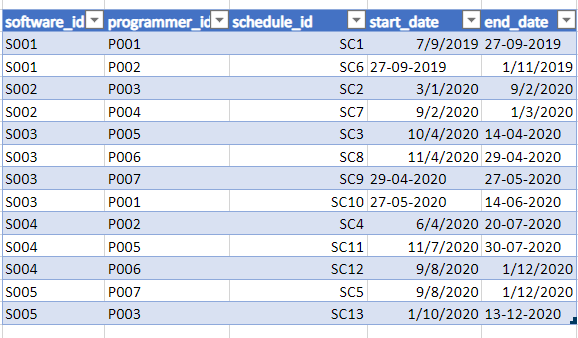
In this table there are no transitive dependencies so, this table is already in 3nf.



**SoftwareProjectSchedule(2NF):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software\_id | Programmer\_id | Schedule\_id | Start\_date | End\_date |

In this table transitive dependency is schedule\_id -> start\_date, End\_Date



**Third Normal Form(3NF):**

● A relation is said to be in 3NF if and only if it is 2NF.

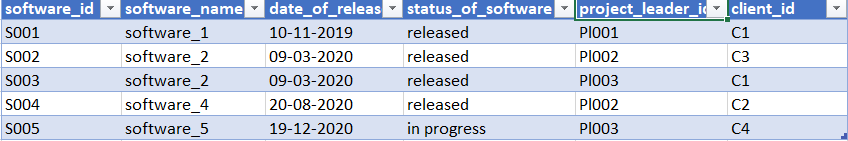
● There should not be any transitive dependency for non prime attributes in the relation.

T.D = non-prime attribute ---> non-prime attribute

**After decomposing the tables in 3nf the tables are**:

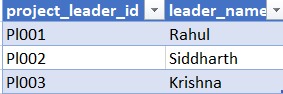
**Software(3NF):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Software\_id | Software name | Date\_of\_release | Status\_of\_software | Projectleader\_id | Client\_id |



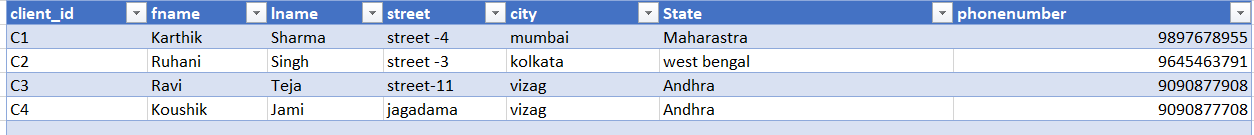
**Projectleader(3NF):**

|  |  |
| --- | --- |
| Projectleader\_id | Leader\_name |



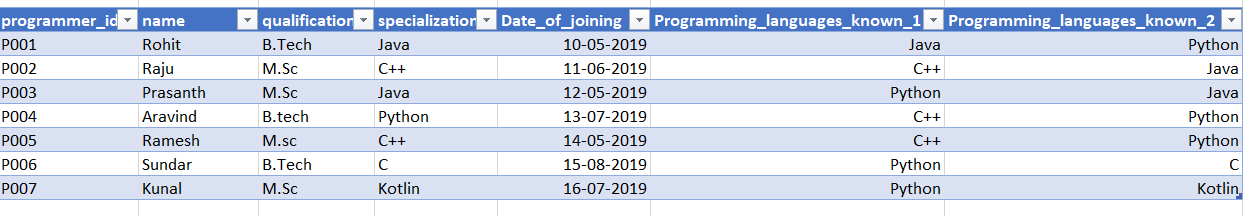
**Client(3NF):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Client\_id | fname | lname | street | city | state |



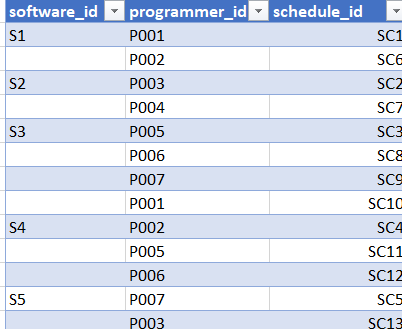
**Programmer(3NF):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Programmer\_id | Name | specialization | qualification | Date\_of\_joining | Programming\_languages\_known |



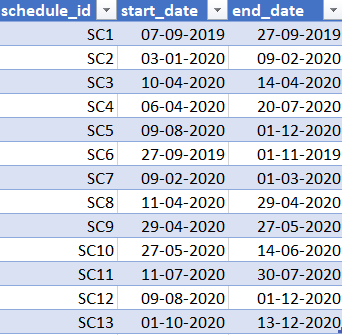
**SoftwareSchedule(3NF):**

|  |  |  |
| --- | --- | --- |
| Software\_id | Programmer\_id | Schedule\_id |



**Schedule(3NF):**

|  |  |  |
| --- | --- | --- |
| Schedule\_id | Start\_date | End\_date |



**Differences between the relations obtained from the ER Model and after Normalization :**

* Before applying normalization, the tables were defined and realised based on the ER diagram. As a result, there was a possibility for data redundancy and inconsistency. In order to avoid that, we applied normalization on the complete table and hence, the number of tables in the final model is considerably reduced.
* After normalization, we reduced to 6 tables as opposed from the tables obtained via the ER Model - eliminating redundancy by dropping the Recognized relation

**DDL statements (WITH CONSTRAINTS SET WHEREVER NECESSARY) based on this set of normalized relations.**

**Programmer:**

create table Programmer

( programmer\_id varchar(5),

name varchar(20) not null,

qualification varchar(20),

specialization varchar(20),

date\_of\_joining date,

programming\_languages\_known1 varchar(10),

programming\_languages\_known2 varchar(10),

primary key (programmer\_id)

);

**Client:**

create table client

( client\_id varchar(5),

fname varchar(15) not null,

lname varchar(15),

street varchar(15),

city varchar(15),

state varchar(15),

phonenumber numeric(10),

primary key(client\_id)

);

**Projectleader:**

create table projectleader

( projectleader\_id varchar(5),

name varchar(20),

primary key(projectleader\_id)

);

**Schedule:**

create table schedule (

schedule\_id varchar(5),

start\_date date,

end\_date date,

primary key(schedule\_id));

**Software:**

create table software

(software\_id varchar(5),

software\_name varchar(15),

Date\_of\_release date,

status\_of\_software varchar(10),

projectleader\_id varchar(5) ,

client\_id varchar(5),

primary key(software\_id),

foreign key(projectleader\_id) references projectleader(projectleader\_id) on delete set null,

foreign key(client\_id) references client(client\_id) on delete set null

);

**Softwareschedule :**

create table softwareschedule

(software\_id varchar(5),

programmer\_id varchar(5),

schedule\_id varchar(5),

primary key(software\_id,programmer\_id),

foreign key(software\_id) references software(software\_id)on delete cascade,

foreign key(programmer\_id) references programmer(programmer\_id) on delete cascade,

foreign key(schedule\_id) references schedule(schedule\_id) on delete set null

);

**DML STATEMENTS:**

insert into programmer values

('P001','Rohit','B.Tech' ,'Java','10/5/2019','Java','Python'),

('P002','Raju','M.Sc','C++','11/6/2019','C++','Java'),

('P003','Prasanth','M.Sc','Java','12/5/2019','Python','Java'),

('P004','Aravind','B.tech','Python','13/7/2019','C++','Python'),

('P005','Ramesh','M.sc','C++','14/5/2019','C++','Python'),

('P006','Sundar','B.Tech','C','15/8/2019','Python','C'),

('P007','Kunal','M.Sc', 'Kotlin','16/7/2019','Python','Kotlin');

insert into client

values

('C1','Karthik','Sharma','street -4','mumbai','Maharastra',9897678955),

('C2','Ruhani', 'Singh','street -3','kolkata','west bengal',9645463791),

('C3','Ravi','Teja','street-11','vizag','Andhra',9090877908),

('C4','Koushik','Jami','jagadama','vizag','Andhra',9090877708);

insert into projectleader values

('Pl001','Rahul'),

('Pl002','Siddharth'),

('Pl003', 'Krishna');

insert into software values

('S001','software\_1','10/11/2019','released','Pl001','C1'),

('S002','software\_2','9/3/2020','released','Pl002','C3'),

('S003','software\_3','15/6/2020','released','Pl003','C1'),

('S004','software\_4','20/8/2020','released','Pl002','C2'),

('S005','software\_5','19/12/2020','inprogress','Pl003','C4');

insert into Schedule values

('SC1','7/9/2019','27/9/2019'),

('SC2','3/1/2020','9/2/2020'),

('SC3','10/4/2020','14/4/2020'),

('SC4','6/4/2020','20/7/2020'),

('SC5','9/8/2020','1/12/2020'),

('SC6','27/09/2019','01/11/2019'),

('SC7','09/02/2020','01/03/2020'),

('SC8','11/04/2020','29/04/2020'),

('SC9','29/04/2020','27/05/2020'),

('SC10','27/05/2020','14/06/2020'),

('SC11','11/07/2020','30/07/2020'),

('SC12','30/07/2020','14/08/2020'),

('SC13','01/10/2020','13/12/2020');

insert into softwareschedule values

('S001','P001','SC1'),

('S001','P002','SC6'),

('S002','P003','SC2'),

('S002','P004','SC7'),

('S003','P005','SC3'),

('S003','P006','SC8'),

('S003','P007','SC9'),

('S003','P001','SC10'),

('S004','P002','SC4'),

('S004','P005','SC11'),

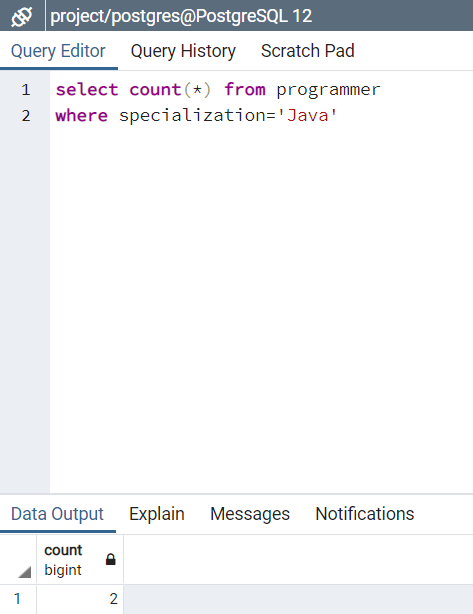
('S004','P006','SC12'),

('S005','P007','SC5'),

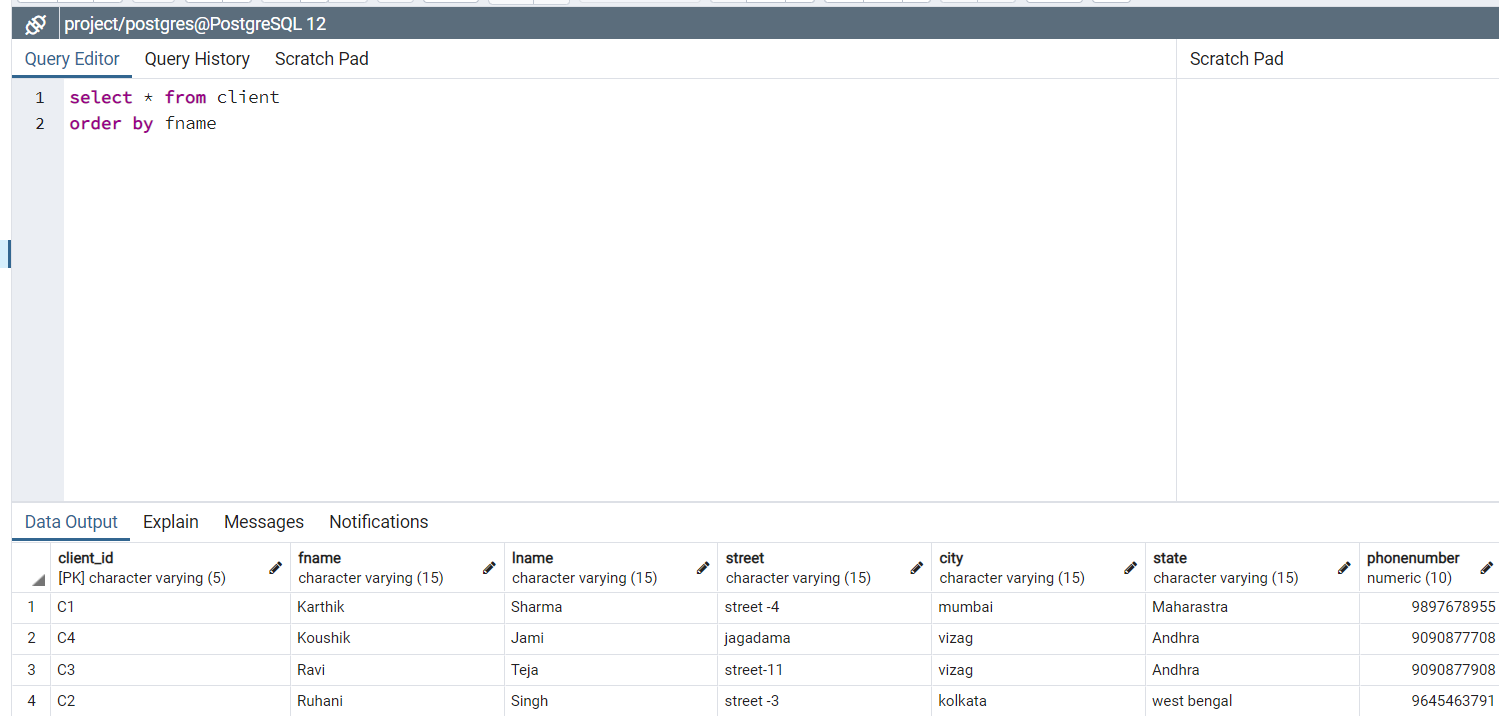
('S005','P003','SC13');

**Queries:**

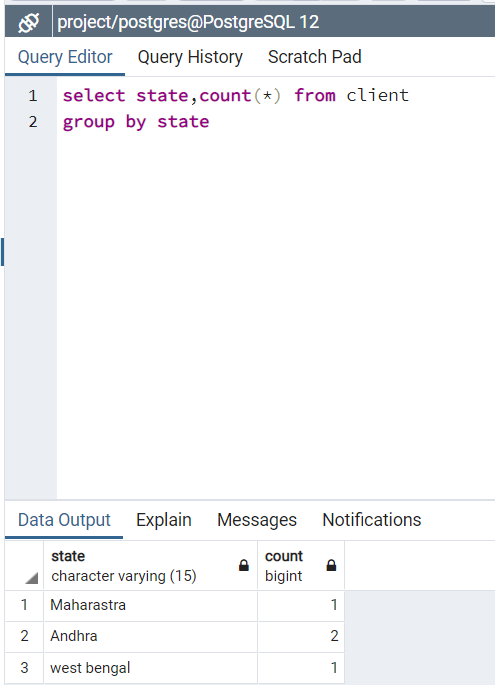
1) count the programmers who are specialized in Java.



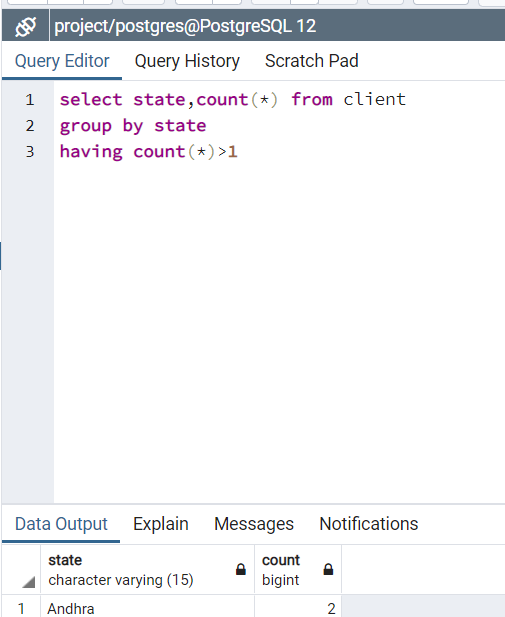
2) list the clients in alphabetically order according to fname.



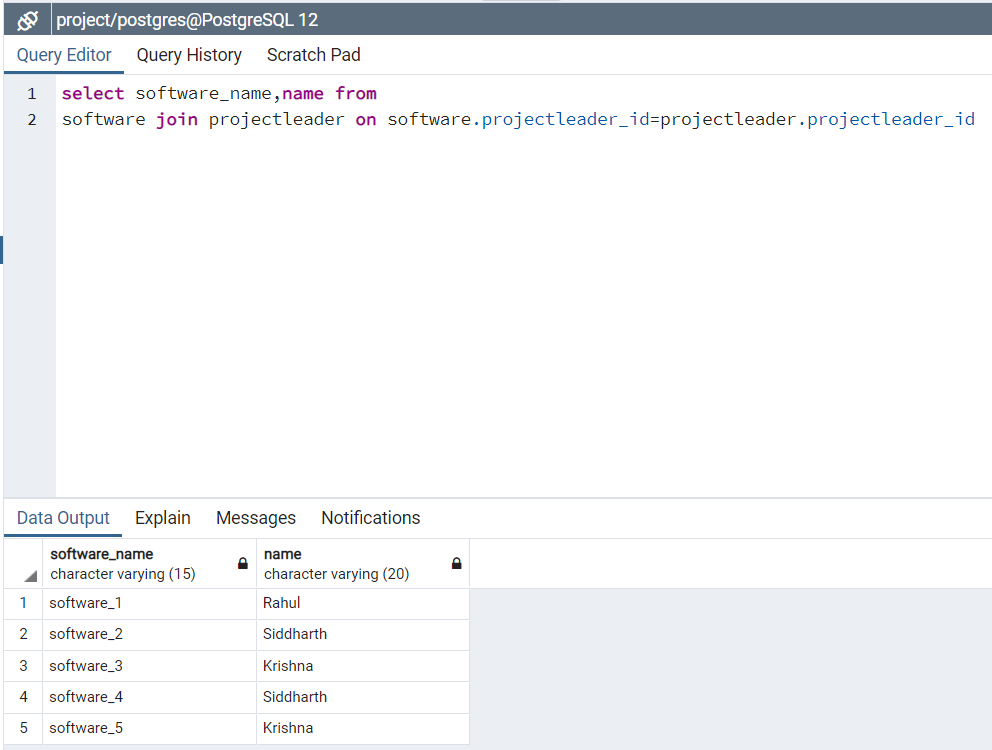
3) count number of clients of each state



4) list the states having more than one client.



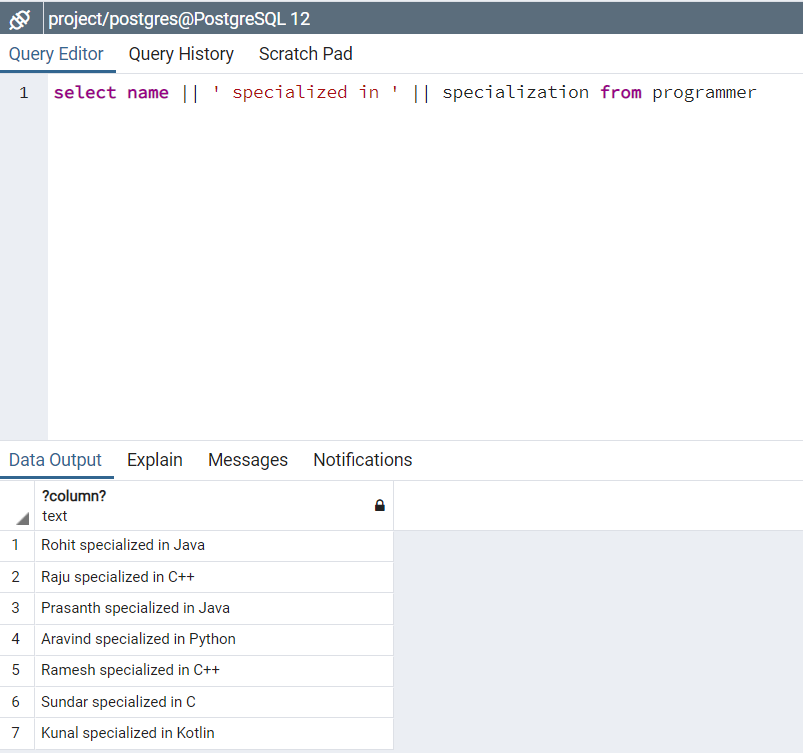
5) list all software's with their project leaders.



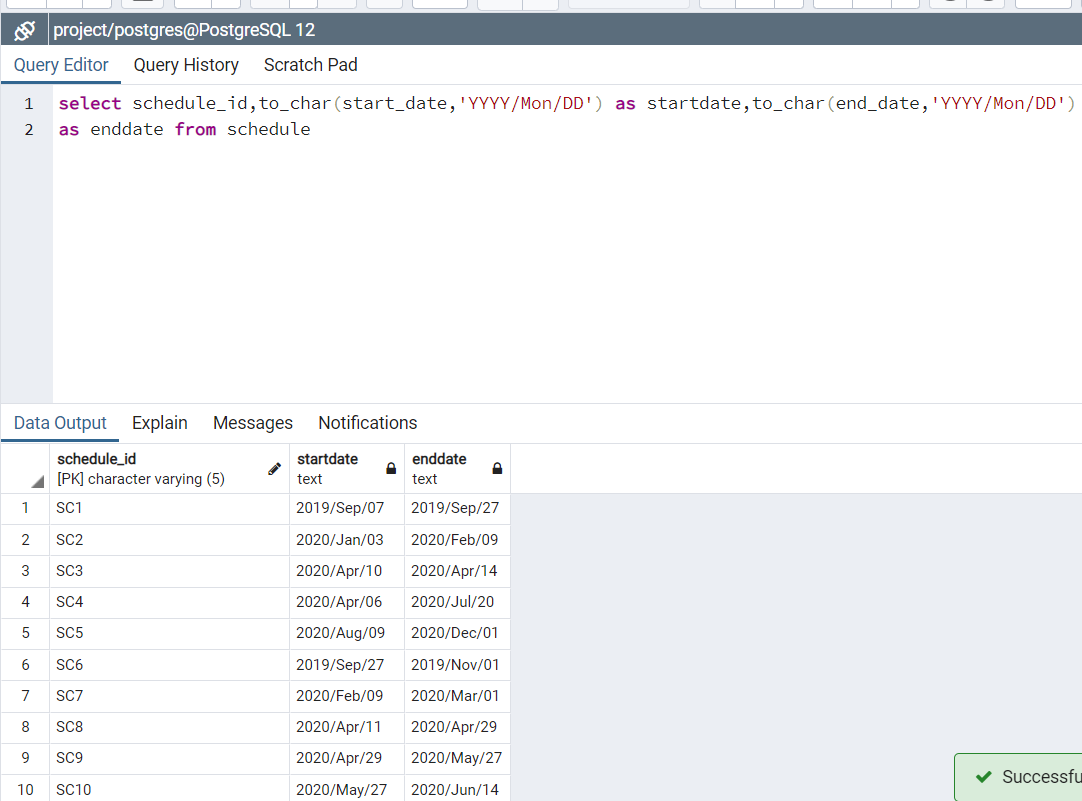
6) list client whose fname starts with K and lname starts with J



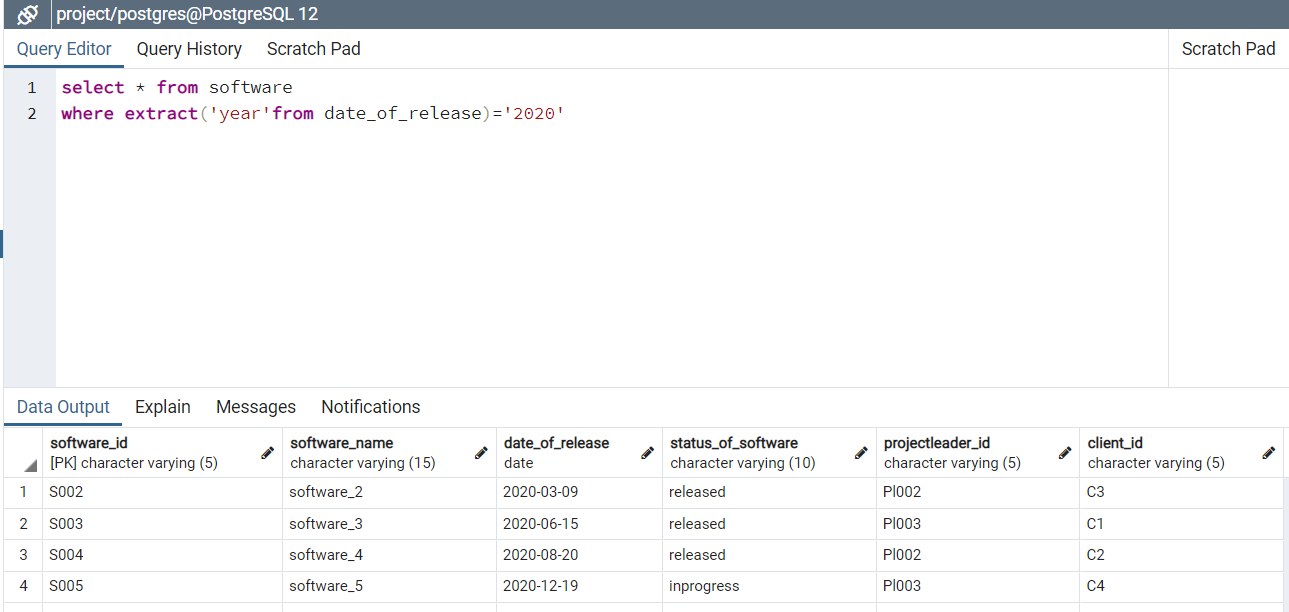
7) concat programmer name with specialization



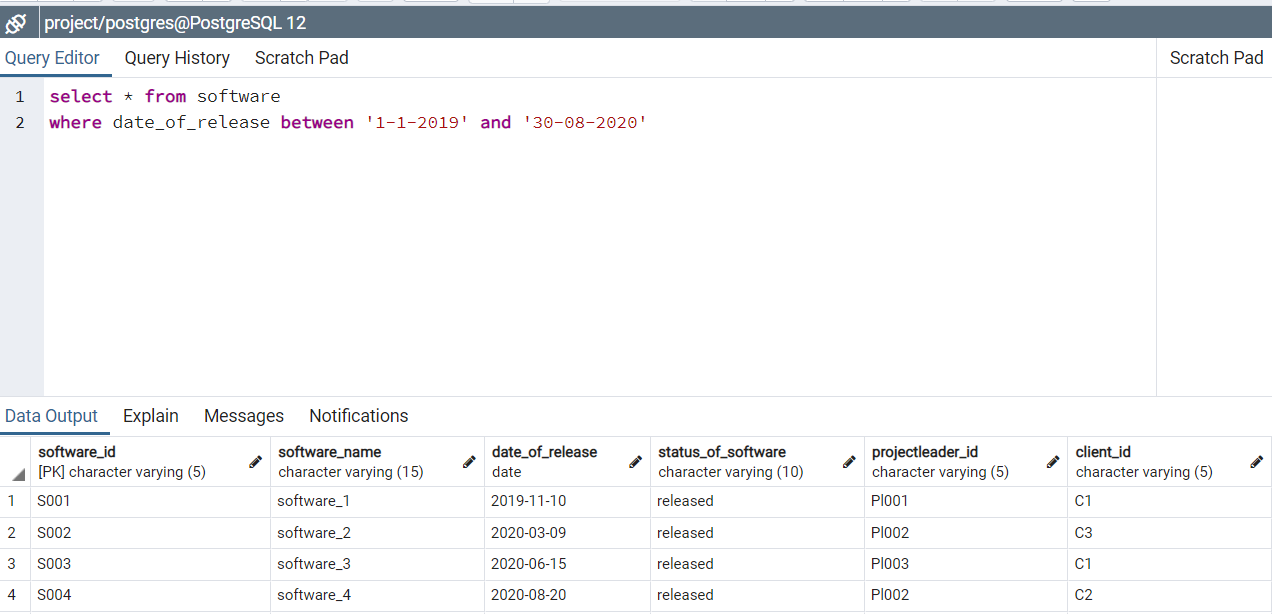
8) Format the start date of schedule in YYYY/Mon/DD as well as end date in YYYY/Mon/DD.



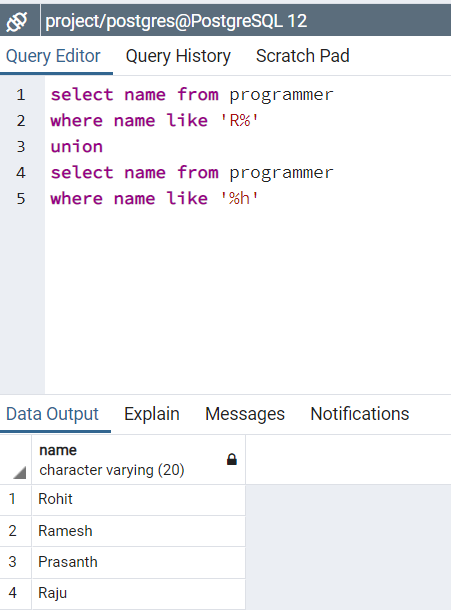
9) list all the software's which are released in year 2020

.

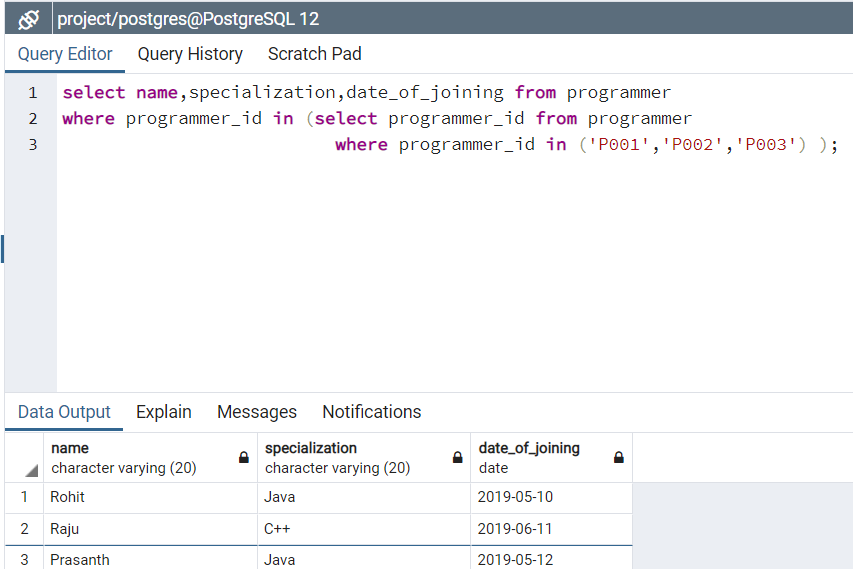
10) list the software's between 1-1-2019 and 30-06-2020



11) find programmer name whose name starts with ‘R’ or ending with ‘h’ or both



12) list programmer name, specialization, date\_of\_joining whose programmer id is ‘P001’,’P002’,’P003’ .





**T.Raviteja (18354)**

**S.M.Mustaq Ahammad(18350)**

**Jami Koushik (18363)**

**P.V.Dinesh(18341)**

**R.Manoj(18344)**