



--- Students --

1. Get students with a particular skill/Interest:

**def get\_skill(skill,Roll\_No=-1):**

=> SELECT data.student\_id, data.full\_name, data.year\_of\_study, data.branch, contact.telegram\_handle, contact.github, contact.email\_address

FROM

(

SELECT s.student\_id, s.full\_name, s.year\_of\_study, s.branch

FROM students s, interests i

WHERE i.student\_id = s.student\_id and i.interest= {**Enter Text**}

) data

LEFT JOIN

contact\_details contact

ON data.student\_id = contact.student\_id ;

1. Upload a particular document:

**def upload\_doc(Roll\_No,course\_no,name\_of\_content,type\_of\_doc,link):**

=> INSERT INTO contents VALUES

(null,{**Time**},{**Roll NO**},

{**Name of content**},{**Doc Type**},{**Link**});

1. Show document related to a particular course:

**def find\_by\_course(course):**

=> SELECT \*

FROM contents

Where course\_no = {**Text**};

1. View admins of all the clubs:

**def view\_admins():**

=> SELECT a.student\_id, a.name, a.club\_id, a.club\_name, c.telegram\_handle, c.github, c.email\_address

From admins a

LEFT JOIN

contact\_details c

ON a.student\_id = c.student\_id ;

1. Update your contact details.

**def update\_contact(Roll\_No,tele,github):**

=> UPDATE contact\_details

SET telegram\_handle = {**Enter Text**}, email\_address = {**Enter Text**}, github = {**Enter Text**}

WHERE student\_id = {**Enter Text**};

Show the upcoming events of the clubs which you are a part of.

=> Select E.club\_id , E.photos\_link , E.date, E.event\_updates, E.event\_name , E.description

From events E , members M

Where M.student\_id = 'Text' and M.club\_id = E.club\_id and E.date >= 'Current Date and Time'

//can be skipped. We have **Queries - 4**

Show the events of the club for a particular date.

=> Select \*

From events E

Where E.club\_id = 'Text' and Date(E.date) = 'Text'

**// Queries - 4** will work

1. Search for a particular person by his/her roll no.

**def search\_by\_roll(Roll\_No):**

//We can do it by using names as well but in that case, we’ll have to make the whole data(his roll no and name) available for each user in the starting only

=> SELECT student\_id, year\_of\_study, branch

FROM students

WHERE student\_id = {**Enter Text**};

*View contact details of a admins*

*=> SELECT a.student\_id, a.name, a.club\_id, a.club\_name, c.telegram\_handle, c.github, c.email\_address*

*From contact\_details c*

*RIGHT JOIN admins a*

*ON a.student\_id = c.student\_id ;* Similar to 4

1. View all interest of a person

**def profile\_data(Roll\_No):**

=> SELECT interest

FROM INTEREST

WHERE student\_id = {**Enter Text**};

1. View all the clubs of a given person

**def profile\_data(Roll\_No):**

=> SELECT \*

FROM members

WHERE student\_id = {**Enter Text**};

1. Verify the credentials of a user:

**def verify\_cerdential(Roll\_No, password):**

=> SELECT COUNT(\*)

FROM students

WHERE student\_id = {**Enter Text**} and password = {**Enter Text**};"

1. Change Password of a particular user:

**def change\_pass(Roll\_No, password):**

// to be done after forgot password

=> UPDATE students

SET

password = {**Enter Text**}

WHERE student\_id = {**Enter Text**};

--Interests--

1. Number of students for each interest.

**def interest\_popularity():**

select interests.interest,count(students.student\_id)

from students,interests where students.student\_id = interests.student\_id

group by interests.interest;

1. Number of students with different interests in different branches.

**def branch\_interest():**

=> Select students.branch,interests.interest,count(students.student\_id)

from students,interests where students.student\_id = interests.student\_id

group by students.branch,interests.interest;

Students who are studying in a given year and branch and have a specific interest.

=> select students.student\_id,students.full\_name from students,interests

where students.year\_of\_study = **{Year Input}** and

students.branch = **{branch input}** and

students.student\_id = interests.student\_id and interests.interest = **{Interest Input}**;

//Can be skipped

Students who have not yet filled their interests.

select students.student\_id,students.full\_name from

students left join interests on students.student\_id=interests.student\_id

where interests.int\_id is null;

//Can be skipped

1. Suggest clubs based on interests of a student //bonus!

**def suggest\_club(Roll\_No):**

**HardCoded. No Query**

1. Suggest courses to interests //bonus

**def suggest\_courses(Roll\_No):**

**HardCoded. No Query**

// these need to be hardcoded into the backend

-- FMS --

1. Resolve requests/ complaints from FMS end

**def resolve\_FMS(request\_id,isRequest = False):**

=>UPDATE FMS\_{}

SET

Status = “Yes”

WHERE request\_id = %s;

1. Cancel FMS request

**def cancel\_FMS(req\_Id):**

=> DELETE FROM FMS\_Request

WHERE request\_Id = {**Enter Text**};

1. Know status about your FMS complaint/request.

**def status\_FMS(email):**

=>SELECT request\_ID,Phone\_Number,Location\_ID,\

Room,Description,Status

FROM FMS\_Request

WHERE Email = %s;

SELECT complaint\_ID,Phone\_Number,Location\_ID,

Room,Description,Status

FROM FMS\_complaint

WHERE Email = %s;

1. Make a new FMS request.

**def new\_FMS\_req\_comp(Name,Email,Phone,Location\_ID,Room,Description,isRequest = False):**

=> INSERT INTO FMS\_Request VALUES

(NULL,{**NAME**},{**EMAIL**},{**Phone**},{**Location Id**},{**Room No**},{**Description**},”No”);

1. Make a new FMS complaint.

**def new\_FMS\_req\_comp(Name,Email,Phone,Location\_ID,Room,Description,isRequest = False):**

=> INSERT INTO FMS\_Complaint VALUES (NULL,{**NAME**},{**EMAIL**},{**Phone**},{**Location Id**},{**Room No**},{**Description**},”No”);

**Queries:-**

1. Number of students for each club:-

**def popular\_club():**

select count(M.student\_id), C.club\_name

from members M,clubs\_and\_societies C

where M.club\_id = C.club\_id

group by M.club\_id

order by count(M.student\_id) DESC;

1. Select students with a particular tag:

**def search\_Tag(tag):**

select student\_id

from members M

where M.club\_tag = "Text";

1. Add a entry into the calendar:

**def insert\_to\_calender(Roll\_No,DateTime,Title,Description,Venue):**

insert into calender values(

NULL,{Roll\_no},{Date Time},{Title},{Description},{Room});

Sample Example- (NULL,"2018275","2020-01-31 23:23:45","Dance Competition","Random Desc-1","C-101");

1. Give all the events of a particular student from his calender on a particular date:

**def get\_events(Roll\_No,date):**

SELECT title, descriptions, venue

FROM calender C

WHERE C.student\_id = %s AND Date(C.datex) = %s;

Name,Roll\_no of students who are enrolled in more than 1 clubs:

select M.student\_id, count(M.student\_id)

from members M

where (select count(\*) from members M1 where M1.student\_id = M.student\_id) >= 2

group by M.student\_id;

1. Show all the uploaded contents of a particular student:-

**def show\_uploads(Roll\_No):**

Drop view if exist contentx;

CREATE VIEW contentx as

SELECT student\_id,created\_at,name\_of\_content,type\_of\_doc,link\_of\_doc

FROM contents;

select \*

from contentx C

where C.student\_id = "2018275";

// I have added a view here to show the concept of view in our model, although it is not doing something very important.

1. Lists of clubs a particular student have subscribed to:-

**def show\_Subscribed(Roll\_No):**

select club\_name

from Subscribers S

where student\_id = “TEXT”;

1. Reviews given to the files uploaded by a particular student:-

**def reviews\_to\_me(Roll\_No):**

select R.rev\_id,R.student\_id as Reviewer,R.rating,R.comments,C.name\_of\_content,C.student\_id as Uploader

from contents C, reviews R

where C.student\_id = "Text" and R.rev\_id = C.con\_id;

1. Average ratings of all the contents which at least have 1 rating given to it:-

**def avg\_rating\_all():**

select avg(R.rating),R.rev\_id,C.name\_of\_content,C.course\_no,C.student\_id as Uploader

from reviews R, contents C

where rev\_id = con\_id

group by R.rev\_id

order by avg(R.rating) DESC;

1. Average rating of the contents uploaded by a particular student:-

**def avg\_rating\_my(Roll\_No):**

select avg(R.rating),R.rev\_id,C.name\_of\_content,C.course\_no,C.student\_id as Uploader

from reviews R, contents C

where rev\_id = con\_id and C.student\_id = "Text"

group by R.rev\_id

order by avg(R.rating) DESC;

1. Average rating query with variance also:-

select variance(R.rating),avg(R.rating),R.rev\_id,C.name\_of\_content,C.student\_id as Uploader

from reviews R, contents C

where rev\_id = con\_id and C.student\_id = "2018279"

group by R.rev\_id;