# DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

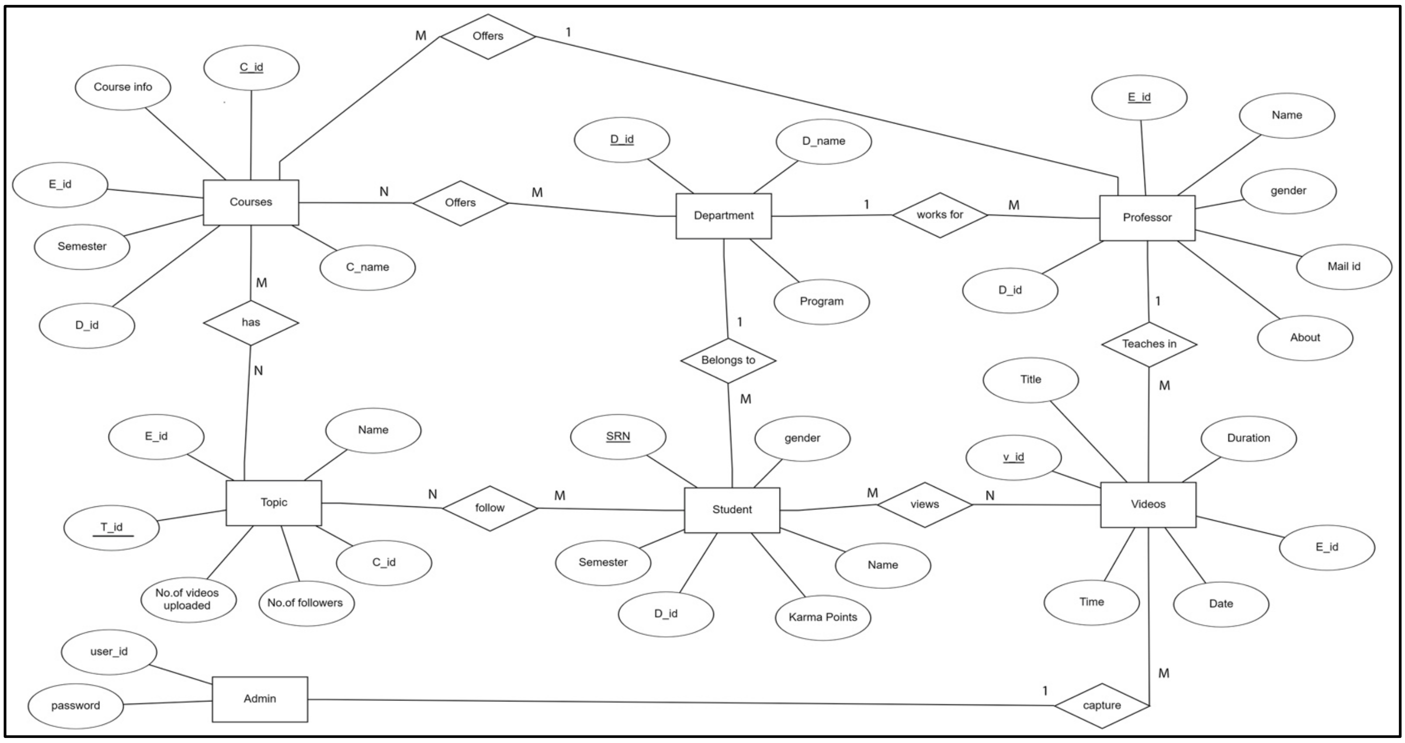
| **SRN :** **PES1201801050** | **Name** : Swarup Banik | **Evaluation date and time:** |
|---|---|---|

## IMPARTUS DATABASE SYSTEM

### Problem statement:

Impartus Database System maintains database of PES University's Impartus website where **Student** ( *identified by* SRN, Name, D_id, Karma_Points, gender and Semester ) can view **Videos** ( *identified by* v_id , Duration, Title, Date, Time and E_id ) uploaded by the **Professor**'s(*identified by* E_id , Name, Mail_id and D_id )on respective **Topic**'s*( identified by* T_id, TopicName,E_id and C_id ) depending upon which **Courses**( *identified by* C_name,C_id, D_id and E_id ) and **Department**( *identified by* D_id,D_name and Program) they belong to and controlled by an **Admin**( *identified by* user_id and password ) .

### ER Diagram :



## Database Schema :

DEPARTMENT

| | | |
|---|---|---|
| D_id | D_name | Program |

STUDENT

| | | | | |
|---|---|---|---|---|
| SRN | Name | Semester | Karma_points | D_id |

VIDEOS

| | | | | | |
|---|---|---|---|---|---|
| V_id | Title | Duration | Time | Date | E_id |

PROFESSOR

| | | | | | |
|---|---|---|---|---|---|
| E_id | Name | Mail_id | About | C_id | D_id |

COURSES

| | | | | | |
|---|---|---|---|---|---|
| C_id | C_name | Anchor_teacher | Semester | C_info | D_id |

TOPIC

| | | | | | |
|---|---|---|---|---|---|
| T_id | Name | E_id | No_of_videos | No_of_followers | C_id |

ADMIN

| | |
|---|---|
| User_id | Password |

## Functional Dependencies:

1. *Department :-*
   - D_id -> {D_id,D_name,Program}
   - D_name ->{D_id,D_name,Program}

2. *Student :-*
   - SRN -> {SRN,Name,Semester,Karma Points,D_id}
   - {SRN,Name} -> {SRN,Name,Semester,Karma Points,D_id}

3. *Courses :-*
   - C_id ->{C_id,C_name,Semester,Course_info,D_id,E_id}
   - C_name -> {C_id,C_name,Semester,Course_info,D_id,E_id}

4. *Professor :-*
   - E_id -> {E_id, Name,Mail id,About,D_id}
   - Mail_id -> {E_id,Name,Mail id,About, D_id}

5. *Videos :-*
   - V_id -> {V_id,Duration,Date,Time,Title,E_id}
   - Title -> {V_id,Duration,Date,Time,Title,E_id}
   - {V_id,E_id} -> {V_id,Duration,Date,Time,Title,E_id}

6. *Topic :-*
   - T_id -> {T_id,E_id,No.of videos uploaded,No.of followers,Name,C_id}
   - Name -> {T_id,E_id,No.of followers,Name,C_id}

7. *Admin :-*
   - User_id -> {user_id,Password}

## Candidate keys:

*1) Department :-*

Candidate Key :- { D_id,D_name }

Primary Key :- { D_id }

As **D_id** and **D_name** both can determine all the attributes as shown by Functional dependency. So, both of them are candidate keys and suitable for making primary key.

But I have made **D_id** as the Primary key. (D_id is the unique number given to each department.)

*2) Student :-*

Candidate Key :- { SRN }

Primary Key :- { SRN }

As **SRN** of a student is an unique idntification number given by PES University to identify each student.

*3) Courses :-*

Candidate Key :- { C_id,C_name }

Primary Key :- { C_id }

As **C_id** and **C_name** both can determine all the attributes as shown by Functional dependency. So, both of them are candidate keys and suitable for making primary key.

But I have made **C_id** as the Primary key. (C_id is the unique number given to each Course.)

*4) Professor :-*

Candidate Key :- { E_id,Mail_id }

Primary Key :- { E_id }

As **E_id** and **Mail_id** both can determine all the attributes as shown by Functional dependency. So, both of them are candidate keys and suitable for making primary key.

But I have made **E_id** as the Primary key. (E_id is the unique number given to each professor.)

*5) Videos :-*

Candidate Key :- { V_id, Title }

Primary Key :- { V_id }

As **V_id** and **Title** both can determine all the attributes as shown by Functional dependency. So, both of them are candidate keys and suitable for making primary key.

But I have made **V_id** as the Primary key because it distinguishes each tuple from the other.

*6) Topic :-*

Candidate Key :- { T_id, Name }

Primary Key :- { T_id }

As **T_id** and **Name** both can determine all the attributes as shown by Functional dependency. So, both of them are candidate keys and suitable for making primary key.

But I have made **T_id** as the Primary key. (T_id is the unique number given to each topic .)

*7) Admin :-*

Candidate Key :- { User_id }

Primary Key :- { User_id }

As **User_id** is an unique idntification to identify the admin.

## Normalization:

If we see the database Schema here the schema is already in **3-NF**.(Normal Form)

- For **1-NF**: Every cell should have atomic values. (It should not have multi-values).
- For **2-NF**: Not Even one Functional Dependency should be partial.
- For **3-NF**: Not Even Single Functional Dependency should have Transitive Functional Dependency

Here for us to violate 1-NF we can add a multivalued attribute to any of the entity and convert it into 1-NF by splitting the entity into two tables as Base Table and Referencing Table where the Referencing Table will be have the multivalued attribute with primary key as foreign key.

Now to violate 2-NF we require single functional dependency i.e. some attributes depends on non-primary key or a part of primary key.

## Lossless Join Property:

If you have relation R and on decomposition into two relations R1, R2 and the natural join of R1 and R2 can get back relation R again. This implies that there has been no loss of data/reductant rows.

➔ ***To check for lossless join decomposition using FD set, following conditions must hold:***

1. Union of Attributes of R1 and R2 must be equal to attribute of R. Each attribute of R must be either in R1 or in R2.
```
Att(R1) U Att(R2) = Att(R)
```
2. Intersection of Attributes of R1 and R2 must not be NULL.
```
Att(R1) ∩ Att(R2) ≠ Φ
```

3. Common attribute must be a key for at least one relation (R1 or R2)

```
Att(R1) ∩ Att(R2) -> Att(R1) or Att(R1) ∩ Att(R2) -> Att(R2)
```

So for *example* we take **Student** table:-

1.) R1(SRN,Semester,Name,gender) and R2( Name , Karma_points , D_id ) will not be a lossless join because if you combine these they obviously won't give us all the columns so there is loss of information.
2.) R1(SRN,Semester,Name,gender) and R2(SRN,Karma_points,D_id) Will be a lossless join because there is an attribute which is in common and unique so there wont be any loss of information.

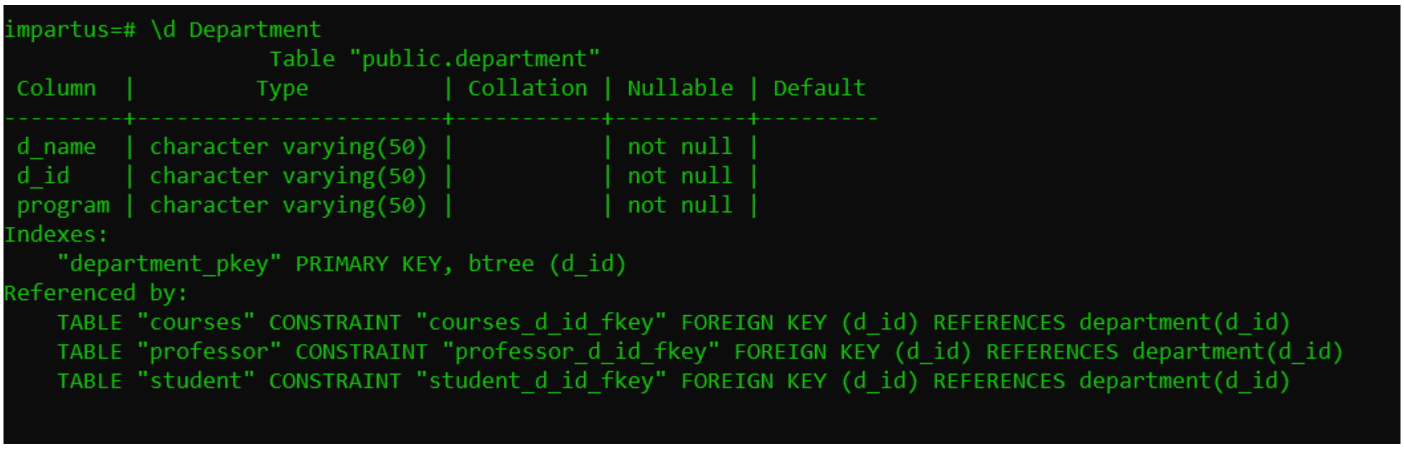
➔ ***Conclusion***

So here by checking over database schema we came to a conclusion that none of the tables that we have in the current relational schema can be decomposed into more two tables such that it's a lossless decomposition. This is happening because we already have in suitable form.

## <u>DDL: TABLE SCRIPTS WITH THEIR INTEGRITY CONSTRAINTS</u>

➔ **Table Department**

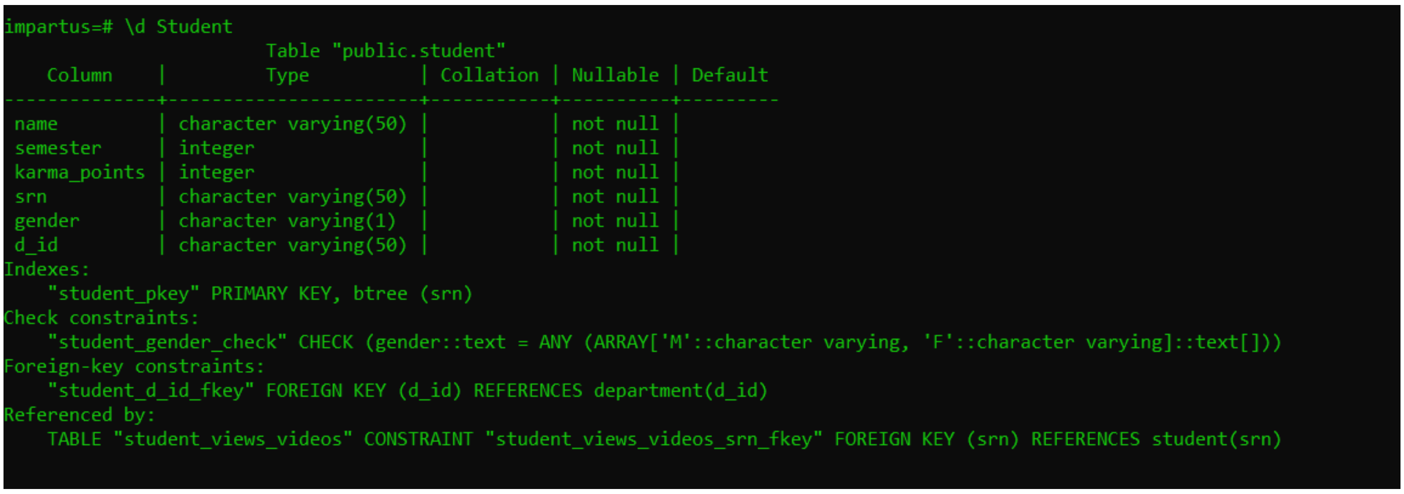
```
    CREATE TABLE Department
(
 D_name VARCHAR(50) NOT NULL UNIQUE,
 D_id VARCHAR(50) NOT NULL UNIQUE,
 Program VARCHAR(50) NOT NULL,
 PRIMARY KEY (D_id)
);
```



```
impartus=# \d Department
                    Table "public.department"
 Column  |         Type          | Collation | Nullable | Default
---------+-----------------------+-----------+----------+---------
 d_name  | character varying(50) |           | not null |
 d_id    | character varying(50) |           | not null |
 program | character varying(50) |           | not null |
Indexes:
    "department_pkey" PRIMARY KEY, btree (d_id)
Referenced by:
    TABLE "courses" CONSTRAINT "courses_d_id_fkey" FOREIGN KEY (d_id) REFERENCES department(d_id)
    TABLE "professor" CONSTRAINT "professor_d_id_fkey" FOREIGN KEY (d_id) REFERENCES department(d_id)
    TABLE "student" CONSTRAINT "student_d_id_fkey" FOREIGN KEY (d_id) REFERENCES department(d_id)
```

## ➔ Table Student

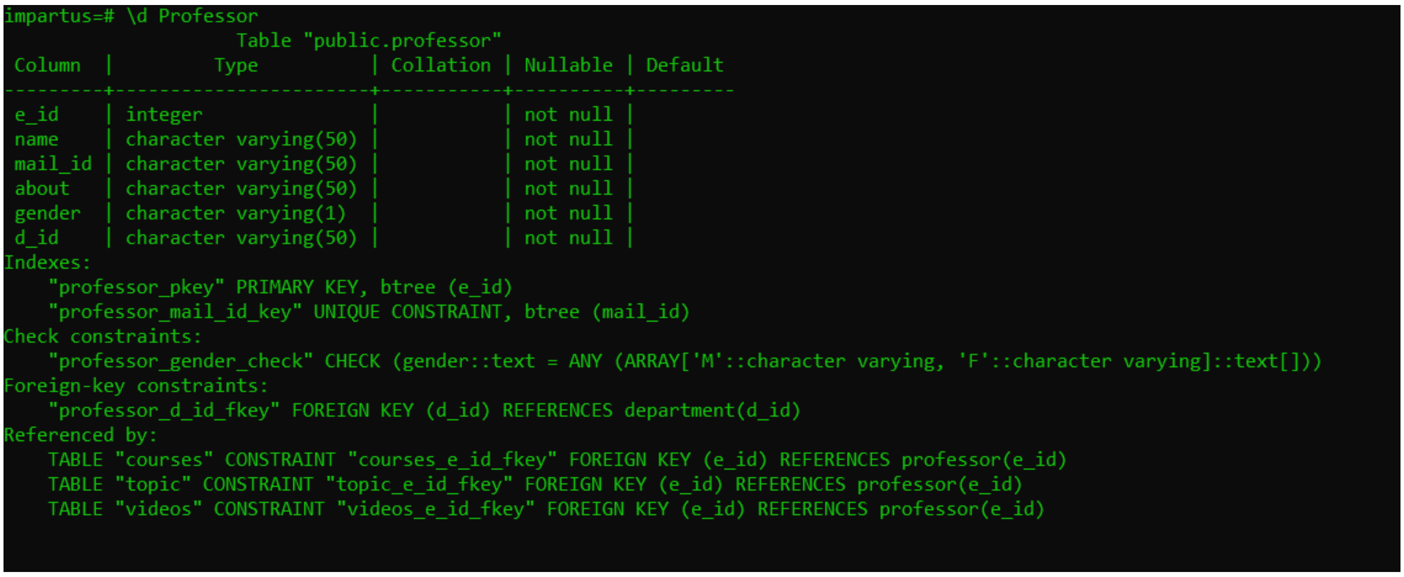
```
CREATE TABLE Student
(
 Name VARCHAR(50) NOT NULL,
 Semester INT NOT NULL,
 Karma_Points INT NOT NULL,
 SRN VARCHAR(50) NOT NULL UNIQUE,
 gender VARCHAR(1) NOT NULL,
 D_id VARCHAR(50) NOT NULL,
 PRIMARY KEY (SRN),
 FOREIGN KEY (D_id) REFERENCES Department(D_id),
 CHECK(gender IN('M','F'))
);
```



```
impartus=# \d Student
                      Table "public.student"
    Column    |         Type          | Collation | Nullable | Default
--------------+-----------------------+-----------+----------+---------
 name         | character varying(50) |           | not null |
 semester     | integer               |           | not null |
 karma_points | integer               |           | not null |
 srn          | character varying(50) |           | not null |
 gender       | character varying(1)  |           | not null |
 d_id         | character varying(50) |           | not null |
Indexes:
    "student_pkey" PRIMARY KEY, btree (srn)
Check constraints:
    "student_gender_check" CHECK (gender::text = ANY (ARRAY['M'::character varying, 'F'::character varying]::text[]))
Foreign-key constraints:
    "student_d_id_fkey" FOREIGN KEY (d_id) REFERENCES department(d_id)
Referenced by:
    TABLE "student_views_videos" CONSTRAINT "student_views_videos_srn_fkey" FOREIGN KEY (srn) REFERENCES student(srn)
```

## ➔ Table Professor

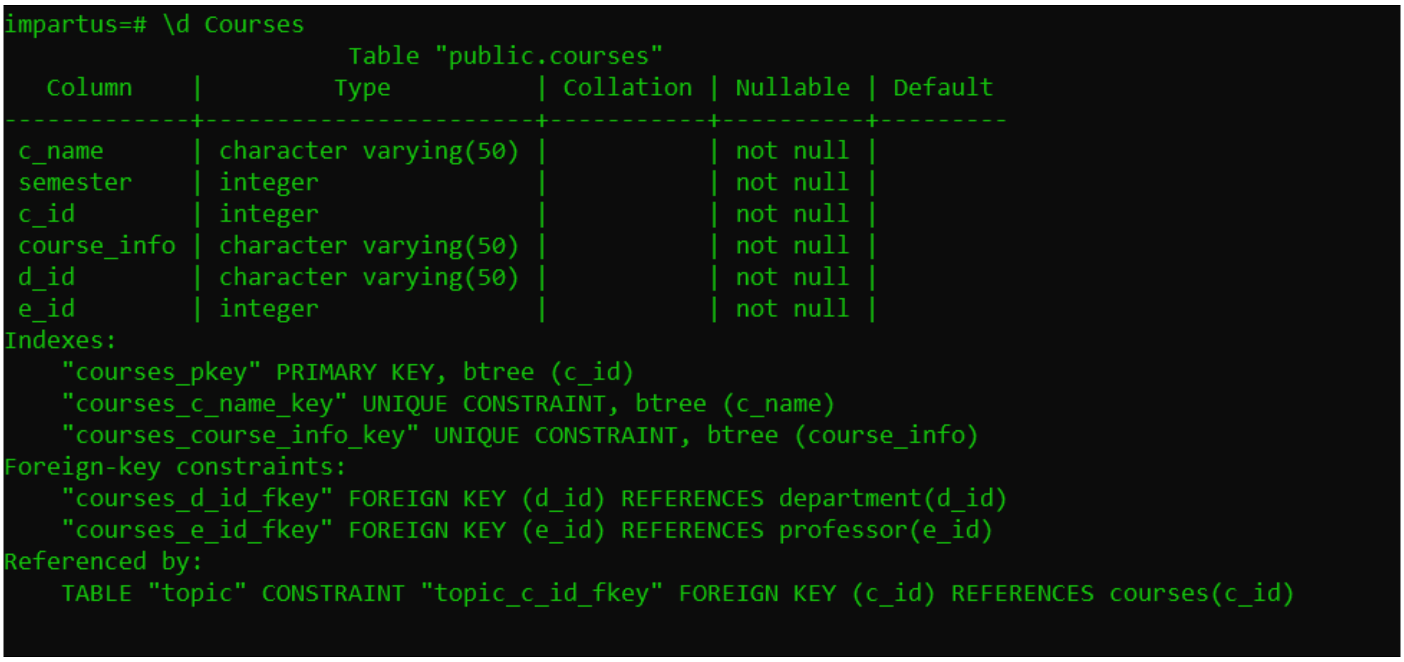
```
        CREATE TABLE Professor
 (
  E_id INT NOT NULL UNIQUE,
  Name VARCHAR(50) NOT NULL,
  Mail_id VARCHAR(50) NOT NULL UNIQUE,
  About VARCHAR(50) NOT NULL,
  gender VARCHAR(1) NOT NULL,
  D_id VARCHAR(50) NOT NULL,
  PRIMARY KEY (E_id),
  FOREIGN KEY (D_id) REFERENCES Department(D_id),
  CHECK(gender IN('M','F'))
);
```



```
impartus=# \d Professor
                    Table "public.professor"
 Column  |         Type          | Collation | Nullable | Default
---------+-----------------------+-----------+----------+---------
 e_id    | integer               |           | not null |
 name    | character varying(50) |           | not null |
 mail_id | character varying(50) |           | not null |
 about   | character varying(50) |           | not null |
 gender  | character varying(1)  |           | not null |
 d_id    | character varying(50) |           | not null |
Indexes:
    "professor_pkey" PRIMARY KEY, btree (e_id)
    "professor_mail_id_key" UNIQUE CONSTRAINT, btree (mail_id)
Check constraints:
    "professor_gender_check" CHECK (gender::text = ANY (ARRAY['M'::character varying, 'F'::character varying]::text[]))
Foreign-key constraints:
    "professor_d_id_fkey" FOREIGN KEY (d_id) REFERENCES department(d_id)
Referenced by:
    TABLE "courses" CONSTRAINT "courses_e_id_fkey" FOREIGN KEY (e_id) REFERENCES professor(e_id)
    TABLE "topic" CONSTRAINT "topic_e_id_fkey" FOREIGN KEY (e_id) REFERENCES professor(e_id)
    TABLE "videos" CONSTRAINT "videos_e_id_fkey" FOREIGN KEY (e_id) REFERENCES professor(e_id)
```

## ➔ Table Courses

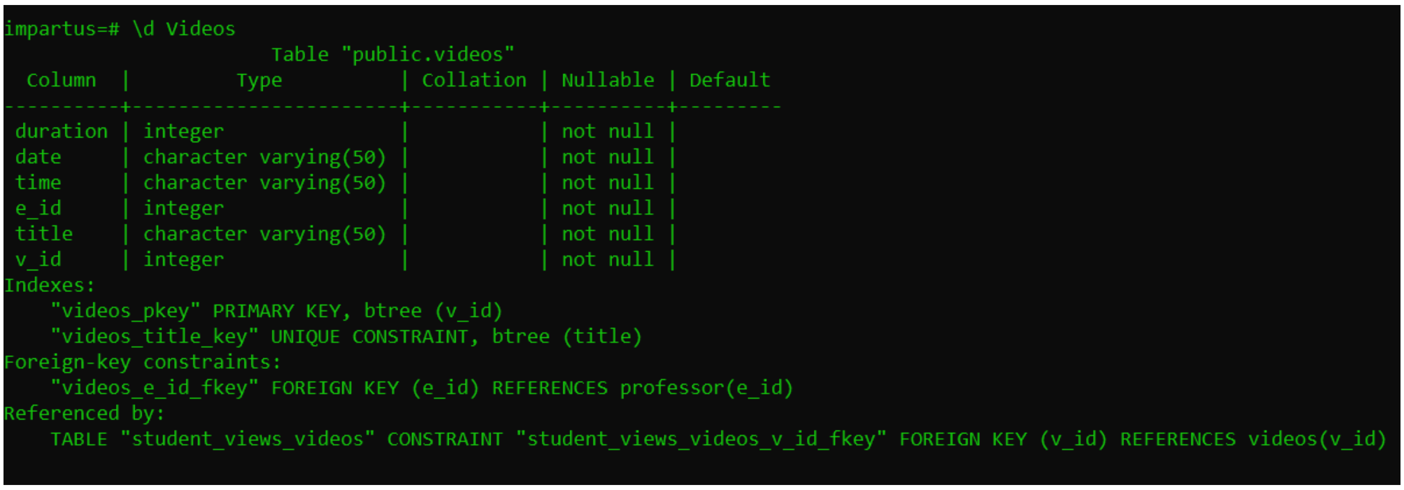
```
      CREATE TABLE Courses
(
  C_name VARCHAR(50) NOT NULL UNIQUE,
  Semester INT NOT NULL,
  C_id INT NOT NULL UNIQUE,
  Course_info VARCHAR(50) NOT NULL UNIQUE,
  D_id VARCHAR(50) NOT NULL,
  E_id INT NOT NULL,
  PRIMARY KEY (C_id),
  FOREIGN KEY (D_id) REFERENCES Department(D_id),
  FOREIGN KEY (E_id) REFERENCES Professor(E_id)
);
```



```
impartus=# \d Courses
                        Table "public.courses"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 c_name      | character varying(50) |           | not null |
 semester    | integer               |           | not null |
 c_id        | integer               |           | not null |
 course_info | character varying(50) |           | not null |
 d_id        | character varying(50) |           | not null |
 e_id        | integer               |           | not null |
Indexes:
    "courses_pkey" PRIMARY KEY, btree (c_id)
    "courses_c_name_key" UNIQUE CONSTRAINT, btree (c_name)
    "courses_course_info_key" UNIQUE CONSTRAINT, btree (course_info)
Foreign-key constraints:
    "courses_d_id_fkey" FOREIGN KEY (d_id) REFERENCES department(d_id)
    "courses_e_id_fkey" FOREIGN KEY (e_id) REFERENCES professor(e_id)
Referenced by:
    TABLE "topic" CONSTRAINT "topic_c_id_fkey" FOREIGN KEY (c_id) REFERENCES courses(c_id)
```

## ➔ Table videos

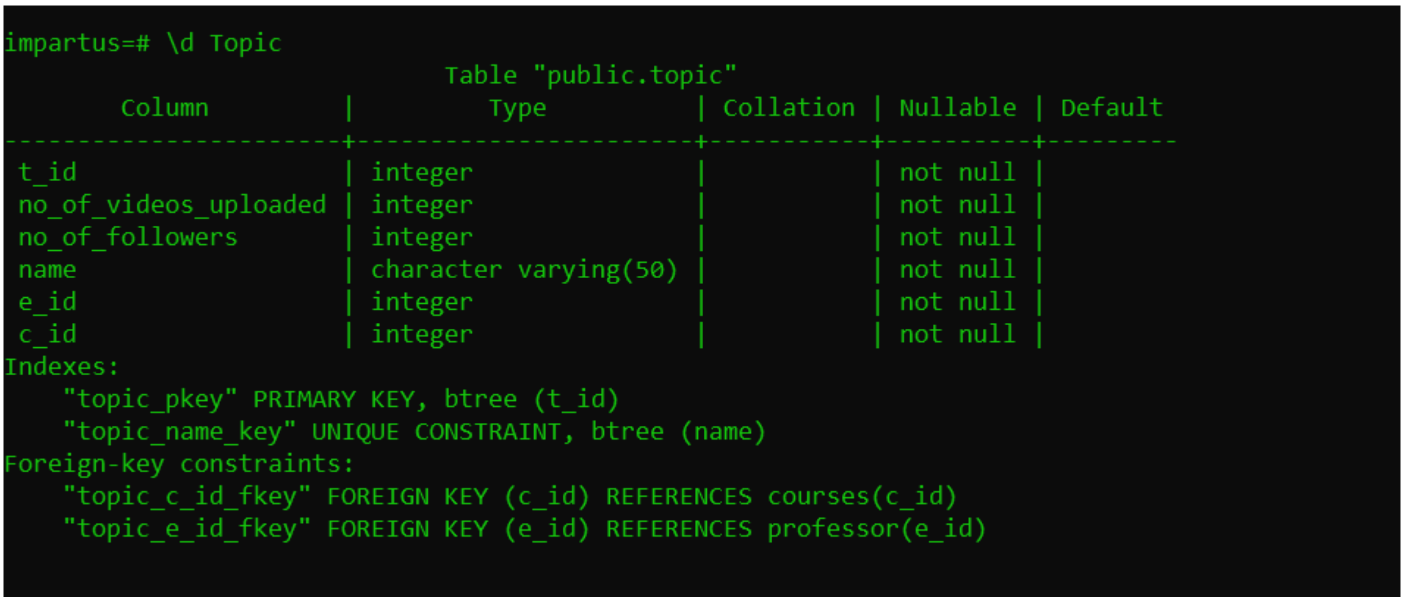
```
      CREATE TABLE Videos
(
 Duration INT NOT NULL,
 Date VARCHAR(50) NOT NULL,
 Time VARCHAR(50) NOT NULL,
 E_id INT NOT NULL,
 Title VARCHAR(50) NOT NULL UNIQUE,
 v_id INT NOT NULL UNIQUE,
 PRIMARY KEY (v_id),
 FOREIGN KEY (E_id) REFERENCES Professor(E_id)
);
```



```
impartus=# \d Videos
                       Table "public.videos"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 duration | integer               |           | not null |
 date     | character varying(50) |           | not null |
 time     | character varying(50) |           | not null |
 e_id     | integer               |           | not null |
 title    | character varying(50) |           | not null |
 v_id     | integer               |           | not null |
Indexes:
    "videos_pkey" PRIMARY KEY, btree (v_id)
    "videos_title_key" UNIQUE CONSTRAINT, btree (title)
Foreign-key constraints:
    "videos_e_id_fkey" FOREIGN KEY (e_id) REFERENCES professor(e_id)
Referenced by:
    TABLE "student_views_videos" CONSTRAINT "student_views_videos_v_id_fkey" FOREIGN KEY (v_id) REFERENCES videos(v_id)
```

## ➔ Table Topic

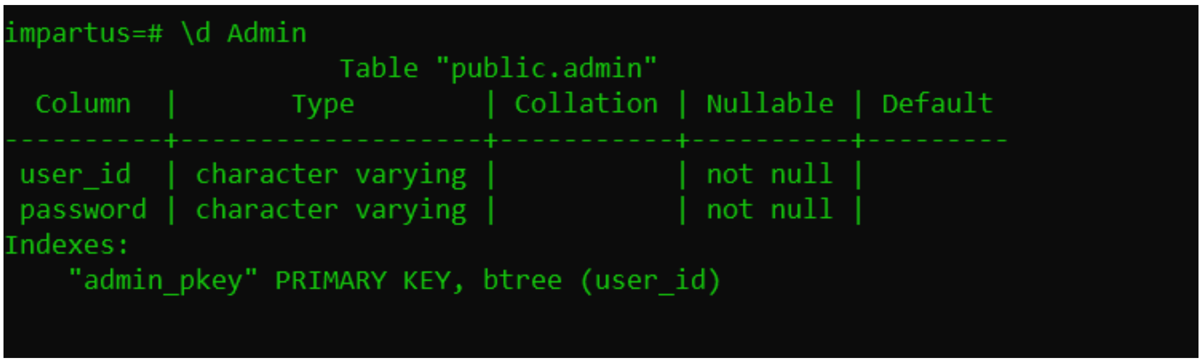
```
      CREATE TABLE Topic
(
  T_id INT NOT NULL UNIQUE,
  No_of_videos_uploaded INT NOT NULL,
  No_of_followers INT NOT NULL,
  Name VARCHAR(50) NOT NULL UNIQUE,
  E_id INT NOT NULL,
  C_id INT NOT NULL,
  PRIMARY KEY (T_id),
  FOREIGN KEY (E_id) REFERENCES Professor(E_id),
  FOREIGN KEY (C_id) REFERENCES Courses(C_id)
);
```



```
impartus=# \d Topic
                              Table "public.topic"
       Column         |         Type          | Collation | Nullable | Default
----------------------+-----------------------+-----------+----------+---------
 t_id                 | integer               |           | not null |
 no_of_videos_uploaded | integer              |           | not null |
 no_of_followers      | integer               |           | not null |
 name                 | character varying(50) |           | not null |
 e_id                 | integer               |           | not null |
 c_id                 | integer               |           | not null |
Indexes:
    "topic_pkey" PRIMARY KEY, btree (t_id)
    "topic_name_key" UNIQUE CONSTRAINT, btree (name)
Foreign-key constraints:
    "topic_c_id_fkey" FOREIGN KEY (c_id) REFERENCES courses(c_id)
    "topic_e_id_fkey" FOREIGN KEY (e_id) REFERENCES professor(e_id)
```

## ➔ Table Admin

```
      CREATE TABLE Admin
(
  user_id VARCHAR NOT NULL UNIQUE,
  password VARCHAR NOT NULL,
  PRIMARY KEY (user_id)
);
```


```
impartus=# \d Admin
                     Table "public.admin"
  Column  |       Type        | Collation | Nullable | Default
----------+-------------------+-----------+----------+---------
 user_id  | character varying |           | not null |
 password | character varying |           | not null |
Indexes:
    "admin_pkey" PRIMARY KEY, btree (user_id)
```

- **Adding sample <u>INSERT statements(</u>**20 each**<u>)</u> as well, that would be used for demo.**

**Table Department**

INSERT INTO Department VALUES ('CSE', '0001', 'BTECH');
INSERT INTO Department VALUES ('ECE', '0002', 'BTECH');
INSERT INTO Department VALUES ('MECH', '0003', 'BTECH');
INSERT INTO Department VALUES ('EEE', '0004', 'BTECH');
INSERT INTO Department VALUES ('BT', '0005', 'BTECH');
INSERT INTO Department VALUES ('CIVIL', '0006', 'BTECH');
INSERT INTO Department VALUES ('DESIGN', '0007', 'BDESIGN');
INSERT INTO Department VALUES ('ARCH', '0008', 'BDESIGN');
INSERT INTO Department VALUES ('BBA18', '0009', 'BBA');
INSERT INTO Department VALUES ('MBA18', '0010', 'MBA');
INSERT INTO Department VALUES ('MTECH18', '0011', 'MTECH');
INSERT INTO Department VALUES ('CHEM', '0012', 'BTECH');
INSERT INTO Department VALUES ('ENV', '0013', 'BTECH');
INSERT INTO Department VALUES ('THERM', '0014', 'BTECH');
INSERT INTO Department VALUES ('AEROSPC', '0015', 'BTECH');
INSERT INTO Department VALUES ('AERONAUT', '0016', 'BTECH');
INSERT INTO Department VALUES ('AGGRICUL', '0017', 'BTECH');
INSERT INTO Department VALUES ('PETROLE', '0018', 'BTECH');
INSERT INTO Department VALUES ('IT', '0019', 'BTECH');
INSERT INTO Department VALUES ('TEXTIL', '0020', 'BTECH');

**Table student**

```
INSERT INTO Student VALUES ('TEJA',4,2000,'PES1201800408','M','0001');
INSERT INTO Student VALUES ('SAMSON',2,170,'PESECE2018','M','0002');
INSERT INTO Student VALUES ('SHARAN',4,800,'PESMECH2018','M','0003');
INSERT INTO Student VALUES ('THARUN',7,290,'PESEEE2018','M','0004');
INSERT INTO Student VALUES ('YAMINI',5,3000,'PESBT2018','F','0005');
INSERT INTO Student VALUES ('TEJAS',4,700,'PESCVL2018','M','0006');
INSERT INTO Student VALUES ('AJAZ',7,1000,'PESDSN2018','M','0007');
INSERT INTO Student VALUES ('YUTHIKA',8,2400,'PESARC2018','F','0008');
INSERT INTO Student VALUES ('ARPIT',6,10,'PESBBA2018','M','0009');
INSERT INTO Student VALUES ('AAYUSH',4,20,'PESMBA2018','M','0010');
INSERT INTO Student VALUES ('CHARAN',2,1700,'PESMT2018','M','0011');
INSERT INTO Student VALUES ('YUKTA',5,200,'PESCHM2018','F','0012');
INSERT INTO Student VALUES ('CELIN',3,299,'PESENV2018','F','0013');
INSERT INTO Student VALUES ('VEDANT',4,2992,'PESTH2018','M','0014');
INSERT INTO Student VALUES ('JIMMY',4,3000,'PESASP2018','M','0015');
INSERT INTO Student VALUES ('SANDRA',4,2000,'PESARN2018','F','0016');
INSERT INTO Student VALUES ('TANYA',4,1000,'PESAG2018','F','0017');
INSERT INTO Student VALUES ('SHRUTIKA',5,2900,'PESPT2018','F','0018');
INSERT INTO Student VALUES ('PRIYA',3,1000,'PESIT2018','F','0019');
INSERT INTO Student VALUES ('DHEERAJ',1,2300,'PESTXT2018','M','0020');
INSERT INTO Student VALUES ('SAKSHI',6,20,'PESCSE2018','F','0001');
```

**Table Professor**

```
INSERT INTO Professor VALUES (11,'DASH','dash@gmail.com','Ex google
employee','M','0001');
INSERT INTO Professor VALUES (22,'DARSHANA','darshana@yahoo.com','Ex
apple employee','F','0002');
INSERT INTO Professor VALUES (33,'MONICA','monica@yahoo.com','Ex amazon
employee','F','0003');
INSERT INTO Professor VALUES (44,'SHARMILA','dsa@hotmail.com','Ex apple
employee','F','0004');
INSERT INTO Professor VALUES (55,'PUNAM','punam@yahoo.com','Ex google
employee','F','0002');
INSERT INTO Professor VALUES (66,'AJOY','AJOY@yahoo.com','HOD','M','0005');
INSERT INTO Professor VALUES (90,'PIUL','piul@yahoo.com','Ex IITB
employee','F','0006');
INSERT INTO Professor VALUES (77,'KAMAL','kamal@yahoo.com','Ex Reddsun
employee','M','0007');
```

```
INSERT INTO Professor VALUES (88,'DARSHAN','darshan@yahoo.com','Ex
flipkart employee','M','0008');
INSERT INTO Professor VALUES (99,'DARSH','darsh@gmail.com','Ex HP
employee','M','0009');
INSERT INTO Professor VALUES (12,'ANAA','ana@yahoo.com','International
Player','F','0010');
INSERT INTO Professor VALUES (21,'ASTHA','asthan@hotmail.com','First Job
','F','0011');
INSERT INTO Professor VALUES (13,'SHANTI','shanti@yahoo.com','Best Teacher
Award 2010','F','0012');
INSERT INTO Professor VALUES (31,'DISHA','disha@yahoo.com','PES
alumini','F','0013');
INSERT INTO Professor VALUES (131,'ARKA','arka@yahoo.com','Orkut
Employee','M','0014');
INSERT INTO Professor VALUES (14,'AYESHA','ayesha@yahoo.com','Best
Employee award','F','0015');
INSERT INTO Professor VALUES (41,'STUTI','stuti@yahoo.com','Ex stanford
professor','F','0016');
INSERT INTO Professor VALUES (15,'DSHAN','dshan@yahoo.com','Ex ICICI
employee','M','0017');
INSERT INTO Professor VALUES (51,'PAWAN','pawan@yahoo.com','Ex WIPRO
employee','M','0018');
INSERT INTO Professor VALUES (61,'SHANA','shana@yahoo.com','Ex apple
employee','F','0019');
INSERT INTO Professor VALUES (16,'JYOTHI','jyothi@outlook.com','Ex IISC
professor','F','0020');
```

**Table Courses**

```
INSERT INTO Courses VALUES ('DBMS', 1, 111,'Data Base','0001',11);
INSERT INTO Courses VALUES ('MPCA', 2, 222,'Computer
Architecture','0002',22);
INSERT INTO Courses VALUES ('GMT', 3,333, 'Mass Transfer','0003',33);
INSERT INTO Courses VALUES ('LA', 4,444, 'Linear Algebra','0004',44);
INSERT INTO Courses VALUES ('WT', 5,555, 'Web technology','0005',55);
INSERT INTO Courses VALUES ('TOC', 6,666, 'Theory of computation','0006',66);
INSERT INTO Courses VALUES ('IOT', 7, 777,'Internet of things','0007',77);
INSERT INTO Courses VALUES ('DDCO', 8, 888,'Design of computer','0008',88);
INSERT INTO Courses VALUES ('ST', 1, 999,'Structural','0009',99);
INSERT INTO Courses VALUES ('ET', 2, 121,'Economic','0010',12);
INSERT INTO Courses VALUES ('PHY', 3, 131,'Physics','0011',13);
INSERT INTO Courses VALUES ('DS', 4, 141,'Data structure','0012',21);
INSERT INTO Courses VALUES ('DAA', 5, 151,'Algorithm','0013',31);
INSERT INTO Courses VALUES ('PYT', 6, 161,'Python','0014',14);
INSERT INTO Courses VALUES ('JAV', 7, 171,'Java','0015',15);
INSERT INTO Courses VALUES ('C', 8, 181,'C CLass','0016',41);
INSERT INTO Courses VALUES ('BD', 1, 191,'BIG DATA','0017',51);
INSERT INTO Courses VALUES ('CC', 3, 201,'Cloud Computing','0018',16);
INSERT INTO Courses VALUES ('LP', 5, 221,'Processor','0019',61);
INSERT INTO Courses VALUES ('MTH1', 6, 231,'Mathematics','0020',22);
```

**Table videos**

```
INSERT INTO Videos VALUES(35,'2020-01-21','13:45:14',11,'VIDEO13',1);
INSERT INTO Videos VALUES(60,'2020-02-22','13:35:14',12,'VIDEO14',2);
INSERT INTO Videos VALUES(65,'2020-05-21','13:45:14',13,'VIDEO15',3);
INSERT INTO Videos VALUES(15,'2020-06-21','23:45:14',14,'VIDEO16',4);
INSERT INTO Videos VALUES(45,'2010-02-21','11:45:14',15,'VIDEO17',5);
INSERT INTO Videos VALUES(75,'2020-09-21','13:45:14',16,'VIDEO18',6);
INSERT INTO Videos VALUES(66,'2020-04-21','13:05:14',22,'VIDEO19',7);
INSERT INTO Videos VALUES(33,'2020-08-11','13:45:14',33,'VIDEO20',8);
INSERT INTO Videos VALUES(45,'2020-04-21','13:45:34',44,'VIDEO1',11);
INSERT INTO Videos VALUES(30,'2020-04-22','13:35:34',55,'VIDEO2',12);
INSERT INTO Videos VALUES(55,'2020-03-21','13:45:24',66,'VIDEO3',13);
INSERT INTO Videos VALUES(25,'2020-07-21','23:45:34',77,'VIDEO4',14);
INSERT INTO Videos VALUES(55,'2010-04-21','11:45:34',88,'VIDEO5',15);
INSERT INTO Videos VALUES(35,'2020-12-21','13:45:34',99,'VIDEO6',16);
INSERT INTO Videos VALUES(16,'2020-11-21','13:05:34',90,'VIDEO7',17);
INSERT INTO Videos VALUES(43,'2020-04-11','13:45:44',31,'VIDEO8',18);
INSERT INTO Videos VALUES(50,'2020-05-21','13:45:34',131,'VIDEO9',19);
INSERT INTO Videos VALUES(25,'2019-04-21','01:45:34',41,'VIDEO10',20);
INSERT INTO Videos VALUES(45,'2020-04-21','13:45:34',51,'VIDEO11',10);
INSERT INTO Videos VALUES(15,'2020-04-29','15:45:34',61,'VIDEO12',9);
```

**Table Topic**

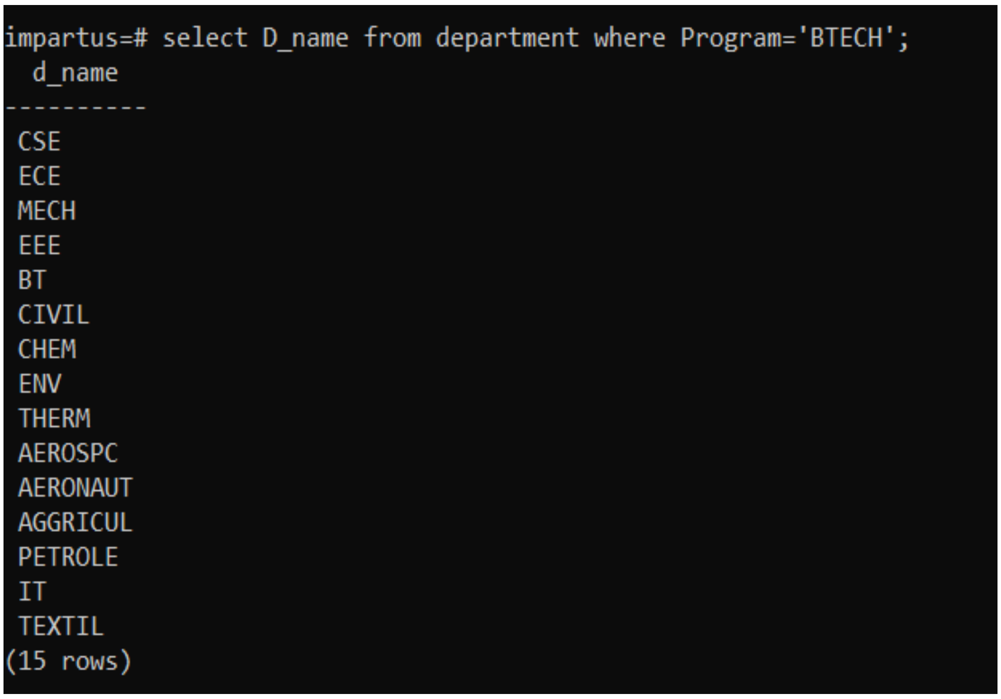
```
INSERT INTO Topic VALUES(1,5,1280,'TOPIC1',11,111);
INSERT INTO Topic VALUES(2,6,2,'TOPIC2',22,222);
INSERT INTO Topic VALUES(3,7,156,'TOPIC3',33,333);
INSERT INTO Topic VALUES(4,5,1290,'TOPIC4',44,444);
INSERT INTO Topic VALUES(5,5,1200,'TOPIC5',55,555);
INSERT INTO Topic VALUES(6,3,1000,'TOPIC6',66,666);
INSERT INTO Topic VALUES(7,6,780,'TOPIC7',77,777);
INSERT INTO Topic VALUES(8,5,190,'TOPIC8',88,888);
INSERT INTO Topic VALUES(9,99,5680,'TOPIC9',99,999);
INSERT INTO Topic VALUES(10,4,120,'TOPIC10',12,121);
INSERT INTO Topic VALUES(11,2,12,'TOPIC11',21,131);
```

**Table Admin**

```
INSERT INTO Admin VALUES ('6999','password');
```

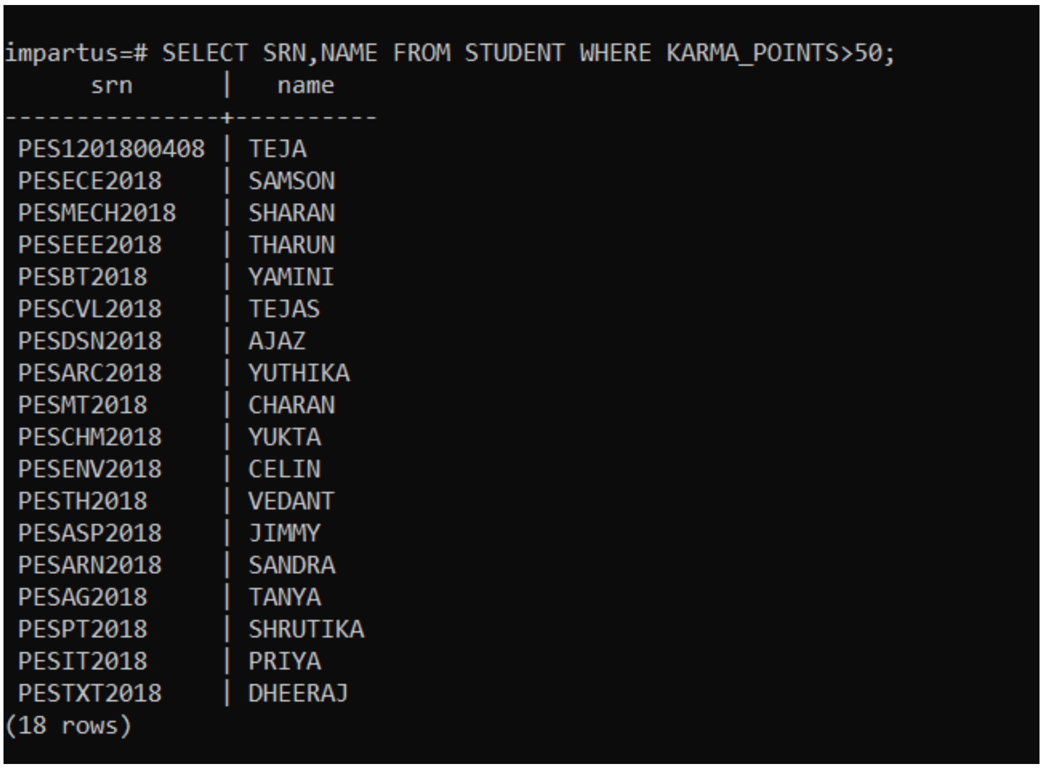
## SQL Queries:

*1.) Select all the Departments of BTECH.*



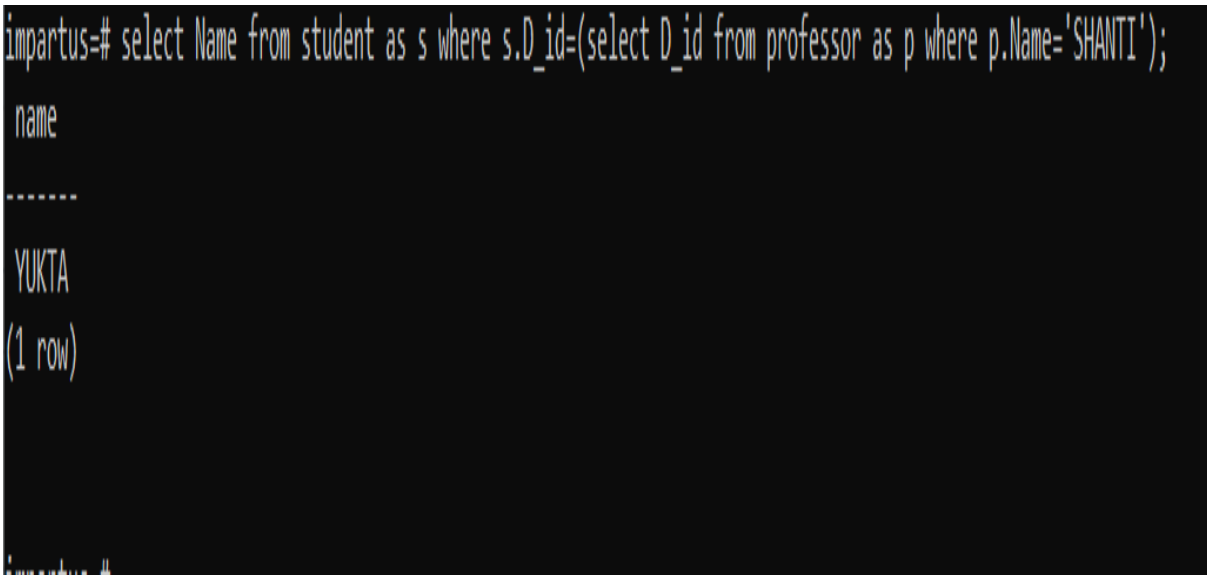
```
impartus=# select D_name from department where Program='BTECH';
  d_name
----------
 CSE
 ECE
 MECH
 EEE
 BT
 CIVIL
 CHEM
 ENV
 THERM
 AEROSPC
 AERONAUT
 AGGRICUL
 PETROLE
 IT
 TEXTIL
(15 rows)
```

*2.) List of all the students having karma points greater than 50.*



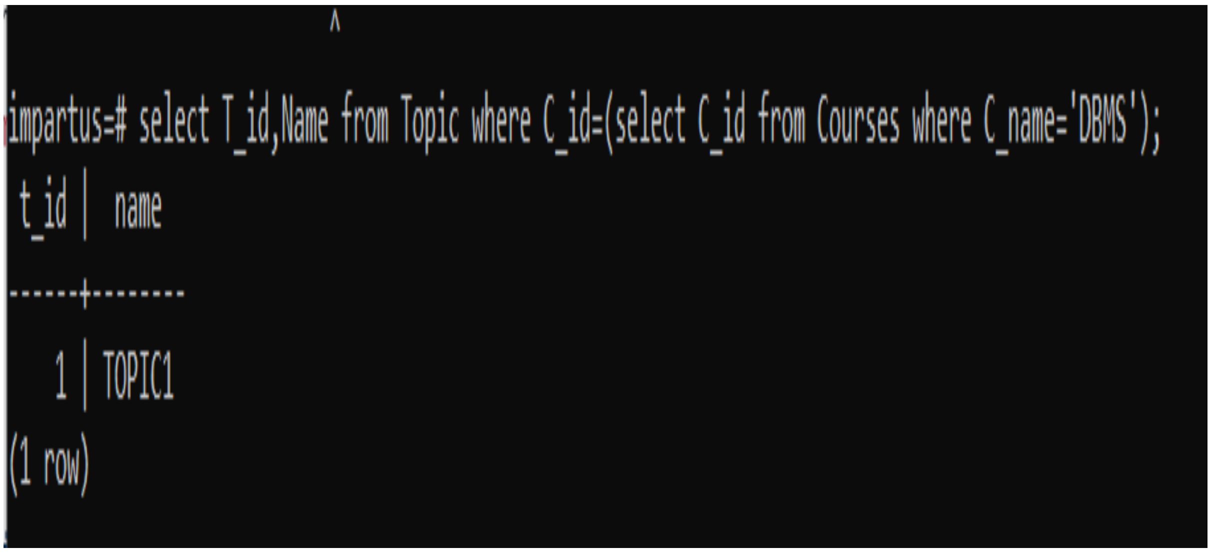
```
impartus=# SELECT SRN,NAME FROM STUDENT WHERE KARMA_POINTS>50;
      srn      |   name
---------------+----------
 PES1201800408 | TEJA
 PESECE2018    | SAMSON
 PESMECH2018   | SHARAN
 PESEEE2018    | THARUN
 PESBT2018     | YAMINI
 PESCVL2018    | TEJAS
 PESDSN2018    | AJAZ
 PESARC2018    | YUTHIKA
 PESMT2018     | CHARAN
 PESCHM2018    | YUKTA
 PESENV2018    | CELIN
 PESTH2018     | VEDANT
 PESASP2018    | JIMMY
 PESARN2018    | SANDRA
 PESAG2018     | TANYA
 PESPT2018     | SHRUTIKA
 PESIT2018     | PRIYA
 PESTXT2018    | DHEERAJ
(18 rows)
```

*3.) List all the students who are in the same department as professor SHANTI .*
**(**Used Aliases AS S)



```
impartus=# select Name from student as s where s.D_id=(select D_id from professor as p where p.Name='SHANTI');
 name
-------
 YUKTA
(1 row)
```

*4.) List all the topics of the course with course name='DBMS'.*
*(Nested Queries)*

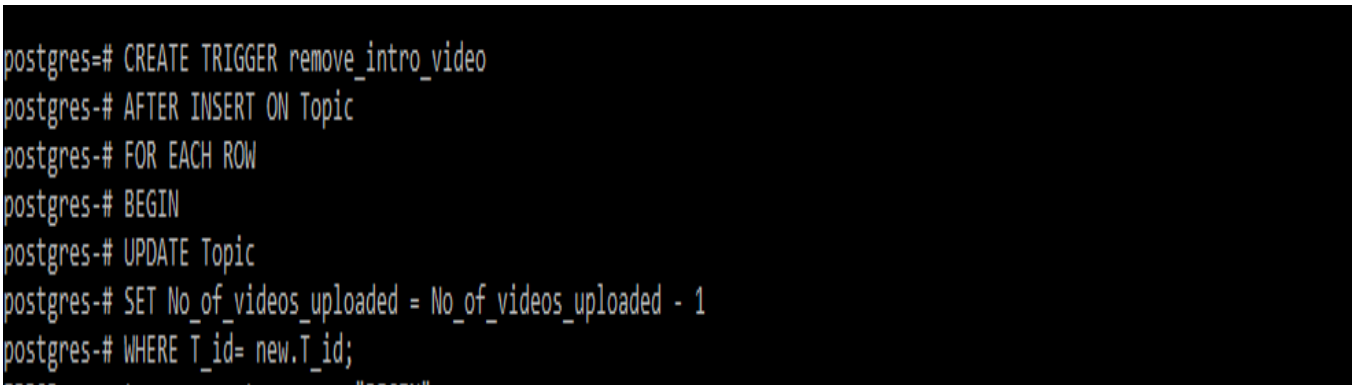


```
                        ^
impartus=# select T_id,Name from Topic where C_id=(select C_id from Courses where C_name='DBMS');
 t_id | name
------+-------
    1 | TOPIC1
(1 row)
```

## **Triggers:**

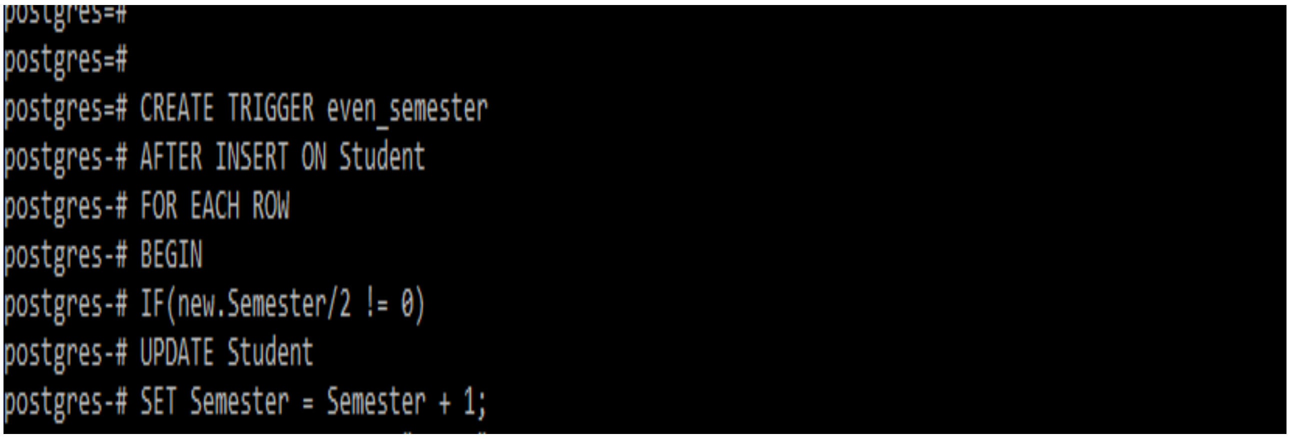
There are 2 triggers used in this database to update the table Topic and Student whenever the event takes place.

*1) This trigger is used to correct the number of videos uploaded as there is an error in the values entered .*
*In this attribute the value entered includes one common introduction of impartus video which was removed from the data before entering in the database as it wasn't related to the topic but number of video wasn't decreased by one. So AFTER insert the values in database it gets triggered and gives us the actual no.of videos uploaded in the topic.*



```
postgres=# CREATE TRIGGER remove_intro_video
postgres-# AFTER INSERT ON Topic
postgres-# FOR EACH ROW
postgres-# BEGIN
postgres-# UPDATE Topic
postgres-# SET No_of_videos_uploaded = No_of_videos_uploaded - 1
postgres-# WHERE T_id= new.T_id;
```

*2) This trigger is used to correct the semester detail in Students table.*
*Some Student's semester value was put according to the previous semester and this is the time of the year when even semester goes on. So whenever the value is inserted after that a trigger is initiated which check whether the semester is odd or even.*
*If odd then the correct detail is updated ..*



```
postgres=#
postgres=#
postgres=# CREATE TRIGGER even_semester
postgres-# AFTER INSERT ON Student
postgres-# FOR EACH ROW
postgres-# BEGIN
postgres-# IF(new.Semester/2 != 0)
postgres-# UPDATE Student
postgres-# SET Semester = Semester + 1;
```

(THANK YOU)