

# **NORTHEASTERN UNIVERSITY, BOSTON**



## **DAMG6210 Data Mgt and Database Design**

### **University Recommendation System**

[https://github.com/Muskansri1/University\\_ Recommendation\\_System](https://github.com/Muskansri1/University_ Recommendation_System)

**SUBMITTED BY:-**

Muskan Srivastava  
Sameer Nimse

002794929  
002752914

## **ABOUT THE PROJECT:**

The opportunity to study in the US has been a dream for many international students. More than 1 million foreign students apply to graduate school for higher studies in the US. Multiple factors such as the tuition fees, university ranking, courses available, return on investment, etc. play a huge role in the final selection of a university,

The abundance of information and lack of credible data on the internet often leaves ambitious students in a dilemma. They are often left confused about the major they want to get in or the professor they are interested in, or even their chances of getting into the university.

In this project, we aim to build a university recommendation system that would act as a one-stop platform for students to aid them in their university shortlisting process.

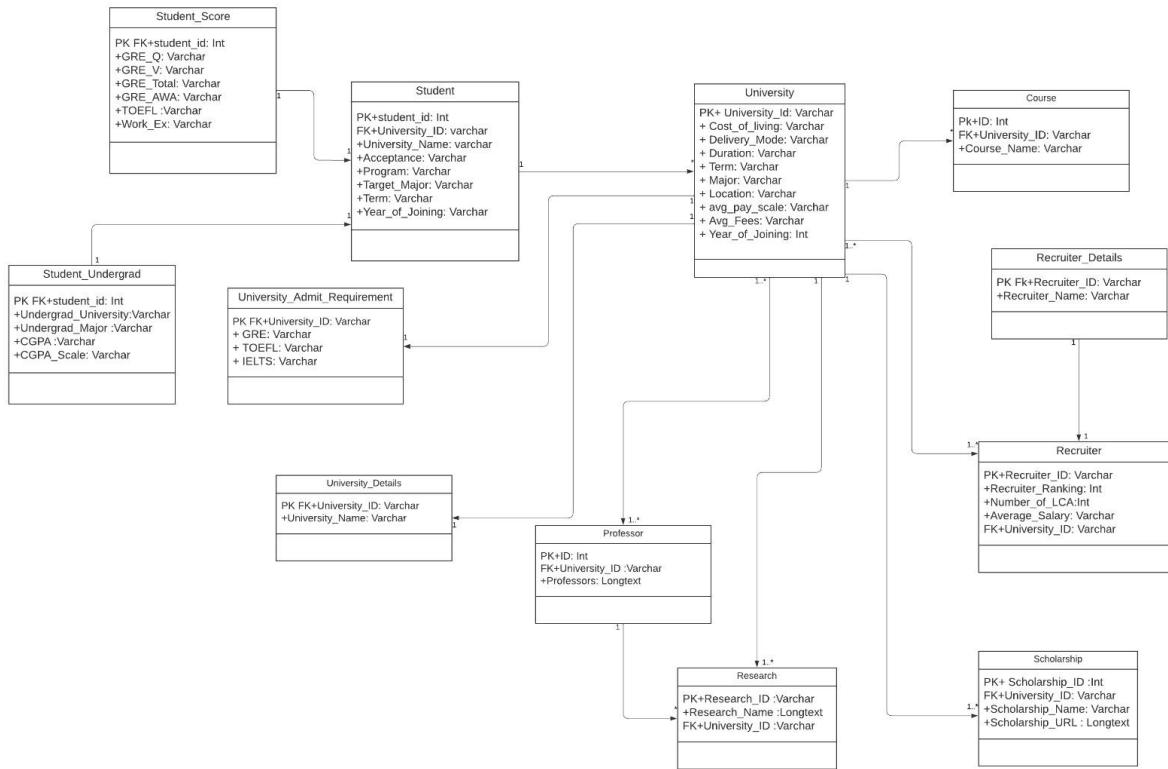
The application/website would feature data from data sources like Yocket, USnews, Admits.fyi, Edulix, timeshighereducation etc. We would scrap the mentioned data from the university websites. The data collected would be cleaned and fed into the database which would be standardized so that the potential students could do their research. The students could have an overview of the entire course curriculum, professors who would be taking a particular course at the university, and university ranking.

We also aim to include a prediction system in the application to let the potential students predict their chances of getting into the university based on their GRE, TOEFL, Undergrad GPA, work experience, etc.

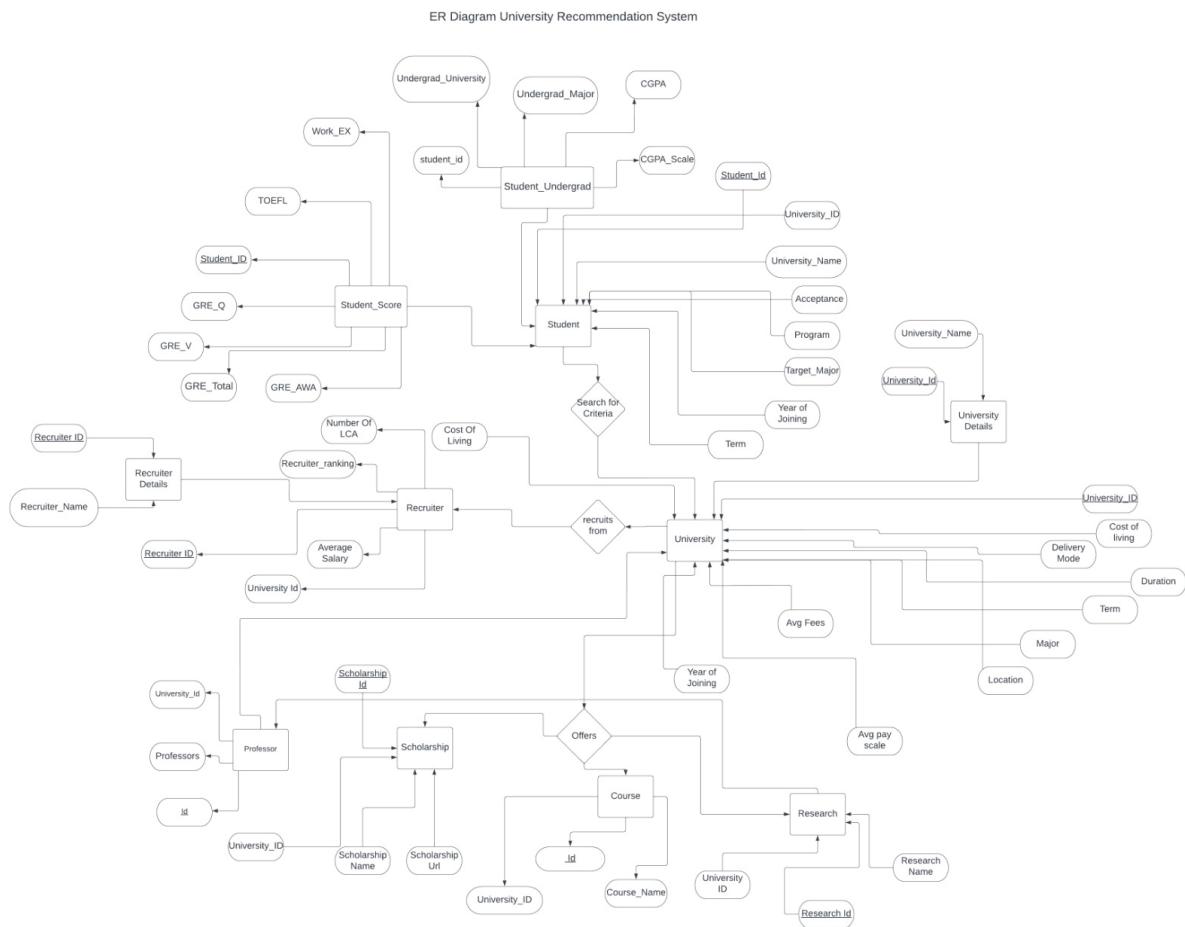


## CLASS DIAGRAM:

CLASS DIAGRAM UNIVERSITY RECOMMENDATION SYSTEM



## ER DIAGRAM



## **DATASET SELECTION:**

The following kinds of data have been sourced from the internet via various scrapping methods:

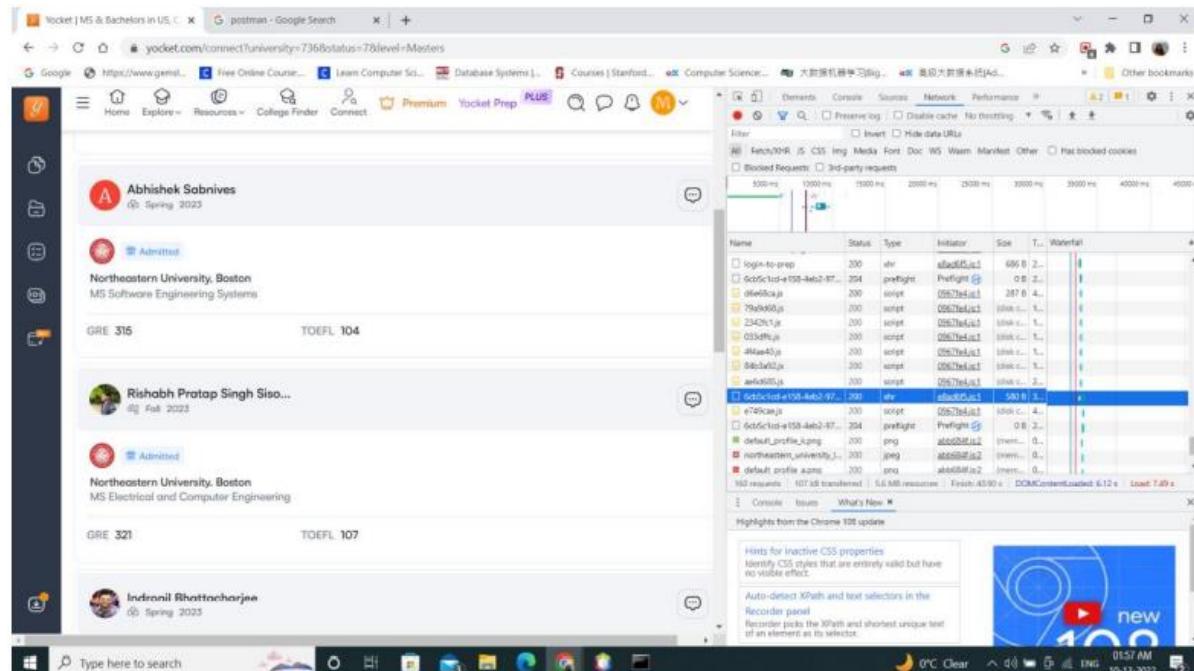
1. University Information:
  - a. University Name
  - b. Expected (Avg) Pay Scale post-graduation
  - c. Delivery Mode for the course: Online/On-Campus
  - d. Duration of the course
  - e. Term: Fall/Spring/Summer/Winter
  - f. Year of Joining
  - g. Major
  - h. Location of the University
  - i. Cost of Living
  - j. Average Fees of the University
  - k. Expected GRE Score of University for Admission
  - l. Expected TOEFL Score of the University for Admission
  - m. Expected IELTS Score of the University for Admission
2. Student Information: This information would provide us with information of the past admitted students so that we can build an information system to predict results.
  - a. Target University Name
  - b. Acceptance Status
  - c. Program Applied to : MS/Mtech/..
  - d. Target Major
  - e. Term applied in
  - f. Year of Joining
  - g. Following scores of the applicant
    - i. GRE\_Q
    - ii. GRE\_V
    - iii. GRE\_Total
    - iv. GRE\_AWA
    - v. TOEFL Scores
  - h. Following information about the undergrad colleges/course
    - i. Undergrad University
    - ii. Undergrad Major
    - iii. CGPA
    - iv. CGPA Scale Evaluation
  - i. Work Experience
3. Scholarship Available in a University:
  - a. Scholarship Name
  - b. Scholarship URL
  - c. University Name which provides the Scholarship

4. Research Opportunities available in the university:
  - a. Research Name
  - b. University Name
5. Recruitment Details: The purpose of this information is to have an estimate of the ROI a student would potentially get post the completion of the course. This table would feature details of the recruiters, their rankings, average package offered to the students and the university it hires maximum from
  - a. Recruiter Name
  - b. Recruiter Ranking
  - c. Number of LCA
  - d. Average Salary offered
  - e. University the recruiter hires maximum from
6. Professor teaching in the University:
  - a. University Name
  - b. Professor Name
7. Courses available in the University:
  - a. Course Available
  - b. University Name

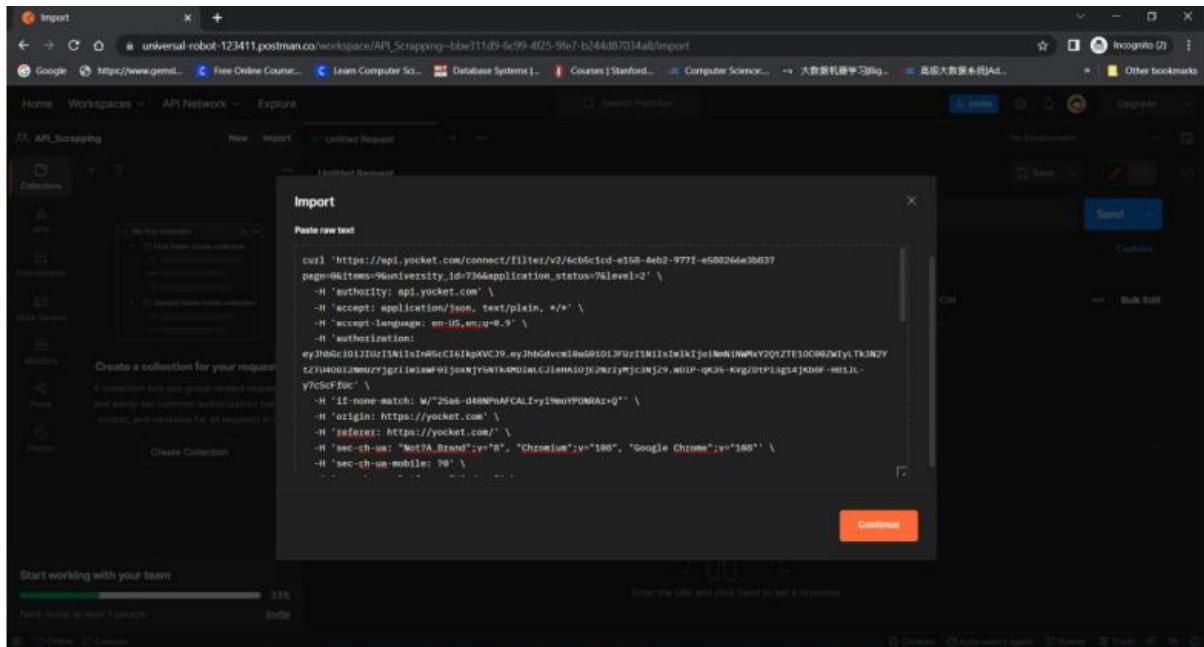
### The methods used to scrape the data:

All the data scrapped and fetched are real-time and dynamic. We have also imported data from Kaggle which was scrapped from Edulix.com as its original source.

#### 1. Scrapping through APIs:



Finding the xhr type files for the API scrapping to import in Postman and send a get request.



Importing the curl bash and fetching the results from Postman in JSON Format.

The screenshot shows the Postman interface after the curl command has been imported and executed. The URL in the address bar is `https://api.yocket.com/connect/filter/v2/6cb5c1cd-e158-4eb2-977f-e588266e3b83`. The main view displays a GET request with parameters 'page' set to 0 and 'items' set to 0. The response body is shown in JSON format, listing two items. The first item is a file named 'Profile Matching your selection'. The second item is a user profile for 'Krishna Shulpalkar'.

```

1 {
2   "state": true,
3   "message": "Profile Matching your selection",
4   "data": [
5     {
6       "id": "2818853",
7       "uid": "5e43d234-faf3-4ebc-adbf-a992baa9a9",
8       "first_name": "Krishna",
9       "last_name": "Shulpalkar",
10      "name": "Krishna Shulpalkar",
11      "username": "Krishna_yb88",
12      "team": "spring",
13      "role_id": 1,
14      "year": 2023,
15      "level": 2
16    }
17  ]
18}

```

## Converting from JSON to CSV.

Save your result: convertcsv .csv Download Result EOL: CRLF											
<b>id</b>	<b>uuid</b>	<b>first_name</b>	<b>last_name</b>	<b>name</b>	<b>username</b>	<b>term</b>	<b>role_id</b>	<b>year</b>	<b>level</b>	<b>created_at</b>	<b>profile_picture_url</b>
1010853	b543d234-faf14e8c-acbf-4a92ba5aa8a9	Krishna	Dhulipala	Krishna Dhulipala	Krishna_Yy6k0	spring	1	2023	2	2022-06-22T17:07:39.867Z	https://static.stupidsid.com/images/us
651670	f2cbf9aa-c277-400a-8733-7decabb2aa2d7	Abhishek	Sabnives	Abhishek Sabnives	AbhishekSabnives	spring	1	2023	2	2021-01-02T06:23:55.000Z	https://static.stupidsid.com/images/us
404962	b66b2038-e8b8-401f-80c3-de19a02720bf	Rishabh	Sisodia	Rishabh Pratap Singh Sisodia	rishabhsisodia	fall	1	2023	2	2018-07-21T18:25:54.000Z	https://lh3.googleusercontent.com/a-/A
551321	46b45290-b2ef-43fe-8ca9-7b36a09a011b	Indronil	Bhattacharjee	Indronil Bhattacharjee	Indronil2409	spring	1	2023	2	2018-11-11T22:05:42.000Z	https://lh3.googleusercontent.com/a-
995581	42ec0d194-95c2-4f5a-8d93-c743a5008793	Sharad	Raina	Sharad Raina	Sharad_QqQFY	spring	7	2023	2	2022-06-11T10:30:55.060Z	https://static.stupidsid.com/images/us
795949	6f7664ef-116e-4fa9-8c23-6b6e09497246	Rohith	Ravindran	Rohith Ravindran	Rohith_ro6qX	fall	1	2022	2	2021-11-12T09:27:40.656Z	https://static.stupidsid.com/images/us
759807	93504fdc-51d3-4228-9faf-	Harshini	Niranjan	Harshini Niranjan	Harshini_gYpzk	fall	1	2022	2	2021-09-13T20:43:26.886Z	https://static.stupidsid.com/images/us

## Result:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	<b>id</b>	<b>uuid</b>	<b>first_name</b>	<b>last_name</b>	<b>name</b>	<b>username</b>	<b>term</b>	<b>role_id</b>	<b>year</b>	<b>level</b>	<b>created_at</b>	<b>profile_picture_url</b>	<b>updated_at</b>	<b>university</b>	<b>university</b>	<b>university</b>	<b>university</b>	<b>university</b>	<b>university</b>	<b>university</b>	<b>university</b>	<b>university</b>
2	1010853	b543d234-Krishna	Dhulipala	Krishna Dhulipala	Krishna_Yy6k0	spring	1		2023	2	2022-06-2	https://sta	2022-12-1	1010853	76268	7	736	2022-12-1	Northeast-northeast	TRUE	northeast Software I	M
3	651670	f2cbf9aa-Abhishek	Sabnives	Abhishek Sabnives	AbhishekSabnives	spring	1		2023	2	2021-01-02	https://lh3	2022-12-0	651670	76268	7	736	2022-12-0	Northeast-northeast	TRUE	northeast Software I	M
4	404962	b66b2038-Indronil	Sisodia	Indronil Prishabhsisodia	Indronil2409	fall	1		2023	2	2018-07-2	https://lh3	2022-12-0	404962	136	7	736	2022-12-0	Northeast-northeast	TRUE	northeast Electrical I	M
5	551321	46b45290-Indronil	Bhattacharjee	Indronil Bhattacharjee	Indronil2409	spring	1		2023	2	2019-11-1	https://lh3	2022-12-0	551321	58448	7	736	2022-12-0	Northeast-northeast	TRUE	northeast Data Anal	M
6	995581	42ec0d194-Sharad	Raina	Sharad Rai Sharad	Sharad_QqQFY	spring	7		2023	2	2022-06-1	https://sta	2022-12-0	995581	49261	7	736	2022-12-0	Northeast-northeast	TRUE	northeast Health Inf	M
7	795949	6f7664ef-Rohith	Ravindran	Rohith Ras Rohith_rav	Rohith_ro6qX	fall	1		2022	2	2021-11-1	https://sta	2022-12-0	795949	76268	7	736	2022-12-0	Northeast-northeast	TRUE	northeast Software I	M
8	759807	93504fdc-Harshini	Niranjan	Harshini N Harshini	Harshini_gYpzk	fall	1		2022	2	2021-09-1	https://sta	2022-12-0	759807	138	7	736	2022-12-0	Northeast-northeast	TRUE	northeast Mechanic	M

## 2. Web Scrapping via Python:

```
In [ ]: import pandas as pd
```

```
In [ ]: df = pd.read_html("https://en.wikipedia.org/wiki/List_of_engineering_schools#United_States")
```

```
In [ ]: df[6]
```

	State (city)[13]	University	School	Master's/doctoralprograms?
0	Alaska (Anchorage)	University of Alaska Anchorage	School of Engineering	Yes
1	Alaska (Fairbanks)	University of Alaska Fairbanks	College of Engineering and Mines	Yes
2	Alabama (Huntsville)	Alabama A&M University	College of Engineering, Technology, and Physic...	Yes
3	Alabama (Auburn)	Auburn University	Samuel Ginn College of Engineering	Yes
4	Alabama (Tuskegee)	Tuskegee University	College of Engineering	Yes
...	...	...	...	...
367	Wisconsin (Madison)	University of Wisconsin–Madison	College of Engineering	Yes
368	Wisconsin (Milwaukee)	University of Wisconsin–Milwaukee	College of Engineering and Applied Science	Yes
369	Wisconsin (Platteville)	University of Wisconsin–Platteville	NaN	NaN
370	Wisconsin (Menomonie)	University of Wisconsin–Stout	College of Science, Technology, Engineering an...	NaN
371	Wyoming (Laramie)	University of Wyoming	College of Engineering and Applied Science	Yes

372 rows × 4 columns

```
In [ ]: df[6].to_excel('US_University.xlsx')
```

```
In [2]: import pandas as pd
```

```
In [3]: df = pd.read_html("https://www.myvisajobs.com/Reports/2022-H1B-Visa-Sponsor.aspx")
```

```
In [10]: df[3]
```

	0	1	2	3
0	Rank	H1B Visa Sponsor	Number of LCA *	Average Salary
1	1	Cognizant Technology Solutions	12681	\$92,766
2	2	Amazon	11486	\$126,163
3	3	Tata Consultancy Services	9822	\$93,484
4	4	Google	9421	\$156,793
5	5	Microsoft	7329	\$146,569
6	6	Facebook	6090	\$175,750
7	7	Ernst & Young	5797	\$118,653
8	8	Infosys	5649	\$92,689
9	9	Apple	4239	\$168,416
10	10	Hcl America	4113	\$99,754
11	11	Accenture	3830	\$140,893
12	12	IBM	3800	\$126,521

### 3. Scapping through tools like Parsehub:

Parsehub is free web scraping tool to turn any site into a spreadsheet or API:

### 4. Downloading the dataset from Kaggle:

<https://www.kaggle.com/datasets/nitishabharathi/university-recommendation>

#### Context

For an aspiring graduate student, choosing which universities to apply to is a conundrum. Often, the students wonder if their profile is good enough for a certain university. This issue can be addressed by building a recommendation system based on various classification algorithms.

#### Content

Original.csv contains profiles of students with admits/rejects to 45 different universities in USA.

score.csv contains score conversion for GRE old to GRE new score mechanism.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
major	researchE	industryEx	toeflScore	program	departmer	greV	greQ	greA	termAndY	ugCollege	cpga	cpgaScale	univName	admit	
Systems ar	0	18	112	MS	Instrumen	160	167	4.5	Fall - 2015	Dharmasir	8.5	10	Worcester	1	
Manufactu	0	0		MS		0			Fall - 2013		0	0	Worcester	1	
(MIS / MSI	0	66	94	MS	Computer	146	157	3	Fall - 2015	IET DAVV	78.28	100	Worcester	1	
	0	0				0					0	0	Worcester	1	
MIS	0	0	81	MS	computer	420	770	2.5	Fall - 2011	Pune Univ	57	100	Worcester	1	
MIS	0	0	273	MS	CE	410	1010	600	Fall - 2006	Thadomal	52	100	Worcester	1	
MIS	0	0	104	MS	Computer	150	161	4.5	Fall - 2015	University	62.2	100	Worcester	1	
MIS-mana	0	0	95	MS	IT	147	156	3	Fall - 2012	MU	52	100	Worcester	1	
MIS	0	0	101	MS	I.T	490	740	3	Fall - 2011	MU	64	100	Worcester	1	
Computer	0	0	107	MS	Informatic	550	780	4.5	Fall - 2011	K J Somaiy	71.4	100	Worcester	1	
MIS	0	0	94	MS	IT	420	680	3.5	Fall - 2012	RAIT	57.3	100	Worcester	1	
Computer	0	0	99	MS	Electronics	490	780	3.5	Fall - 2011	RAIT	65	100	Worcester	1	
Computer	0	0	99	MS	Electronics	490	780	3.5	Fall - 2011	Ramrao Ad	65	100	Worcester	1	
Computer	0	0	99	MS	Electronics	490	780	3.5	Fall - 2011	Ramrao Ad	65	100	Worcester	1	
MIS	0	0	91	MS	Computer	146	161	3	Fall - 2016	University	54.84	100	Worcester	1	
(MIS / MSI	0	39	101	MS	CS	149	159	3	Fall - 2015	AITR RGpv	76.5	100	Worcester	1	
Mechanica	0	0	96	MS	Mechanic	620	690	3	Spring - 20	University	70.5	100	Worcester	1	
MIS	0	0	24	98	MS	Electroni	143	159	3	Fall - 2015	MU	55	100	Worcester	1
Computer	0	0	111	MS	Computer	156	159	3.5	Fall - 2014	RGpv	72.38	100	Worcester	1	
Robotics	0	0	112	MS	Mechatror	154	161	3.5	Fall - 2014	SASTRA	7.8	10	Worcester	1	

## **CREATING THE TABLES:**

### **1. Pre-Normalization:**

The tables created before Normalization are as follows:

#### **1. UNIVERSITY TABLE:**

```
CREATE TABLE university( University_ID VARCHAR(255), University_Name  
VARCHAR(255), avg_pay_scale INT, Delivery_Mode VARCHAR(255), Duration  
VARCHAR(255), Term VARCHAR(255), Year_of_Joining INT, Major  
VARCHAR(255), Location VARCHAR(255), Cost_of_Living INT, GRE INT, TOEFL  
INT, IELTS FLOAT, PRIMARY KEY (University_ID) );
```

```
In [14]: cursor.execute("describe university")  
for x in cursor:  
    print(x)  
  
('University_ID', 'varchar(255)', 'NO', 'PRI', None, '')  
('University_Name', 'varchar(255)', 'YES', '', None, '')  
('avg_pay_scale', 'varchar(255)', 'YES', '', None, '')  
('Delivery_Mode', 'varchar(255)', 'YES', '', None, '')  
('Duration', 'varchar(255)', 'YES', '', None, '')  
('Term', 'varchar(255)', 'YES', '', None, '')  
('Year_of_Joining', 'int', 'YES', '', None, '')  
('Major', 'varchar(255)', 'YES', '', None, '')  
('Location', 'varchar(255)', 'YES', '', None, '')  
('Cost_of_Living', 'varchar(255)', 'YES', '', None, '')  
('Avg_Fees', 'varchar(255)', 'YES', '', None, '')  
('GRE', 'varchar(255)', 'YES', '', None, '')  
('toefl', 'varchar(255)', 'YES', '', None, '')  
('ielts', 'varchar(255)', 'YES', '', None, '')  
  
In [19]: for i,row in University_df.iterrows():  
    cursor.execute("INSERT INTO university values (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)", (int(row['University_ID']),row['University_Name'],row['avg_pay_scale'],row['Delivery_Mode'],row['Duration'],row['Term'],row['Year_of_Joining'],row['Major'],row['Location'],row['Cost_of_Living'],row['Avg_Fees'],row['GRE'],row['toefl'],row['ielts']))  
    uni_recomm.commit()  
  
cursor.execute("SELECT * from university")  
records=cursor.fetchall()  
  
print(records)  
uni_recomm.commit()  
  
((1, 'Northeastern University', '100000', 'On-Campus', '2', 'Fall', 2022, 'MSIS', 'Boston', '9600', '54000', '310', '100',  
'7.5'), ('10', 'Cleveland State University', '36100', 'On-Campus', '1.8', 'Fall', 2022, 'MSIS', 'Cleveland', '24693', '21702',  
'295', '78', '6'), ('11', 'Cornell University', '113700', 'On-Campus', '2', 'Fall', 2022, 'MSCS', 'Ithaca', '19726', '29500',  
'205', '100', '7'), ('12', 'Dalhousie University', '120240', 'On Campus', '2', 'Fall', 2022, 'MCS', 'Canada', '16776', '16960')
```

#### **2. SCHOLARSHIP TABLE:**

```
CREATE TABLE Scholarship ( Scholarship_ID INT, Scholarship_Name  
Varchar(255), Scholarship_URL Varchar (255), PRIMARY KEY (Scholarship_ID)  
);
```

```

: for i, row in Scholarship_df.iterrows():
    cursor.execute("INSERT INTO Scholarship values (%s,%s,%s)", (int(row['Scholarship_ID']),row['scholarship_name'],row['scholarship_desc']))
    uni_recomm.commit()

cursor.execute("SELECT * from Scholarship")
records=cursor.fetchall()

print(records)
uni_recomm.commit()

```

((1, 'Cornaro Scholarship', 'https://www.kappagammapi.org/cornaro-scholarship/'), (2, 'NCAA Postgraduate Scholarship Program', 'http://www.ncaa.org/ncaa-postgraduate-scholarship-program'), (3, 'Eiffel Excellence Scholarship Program', 'https://www.campusfrance.org/en/eiffel-scholarship-program-of-excellence'), (4, 'Fulbright Foreign Student Program', 'https://us.fulbrightonline.org/'), (5, 'Clarendon Scholarships at University of Oxford', 'http://www.ox.ac.uk/clarendon/'), (6, 'Gates Cambridge Scholarships', 'https://www.gatescambridge.org/'), (7, 'Kate Neal Kinley Memorial Fellowship', 'http://faa.illinois.edu/alumni-friends/kate-neal-kinley-memorial-fellowship'), (8, 'Forte Fellows Program', 'http://www.fortefoundation.org/site/PageServer?pagename=mba-fellows#VPyCecZVVE8'), (9, 'Richard A. Freund International Scholarship', 'https://asq.org/about-asq/asq-awards/freundschaer'), (10, 'Olivia James Traveling Fellowship', 'https://www.archaeological.org/grant/james-traveling-fellowship/'), (11, 'Helen M. Woodruff Fellowship', 'https://www.archaeological.org/grant/woodruff-fellowship/'), (12, 'Anna C. and Oliver C. Colburn Fellowship', 'https://www.archaeological.org/grant/colburn-fellowships/'), (13, 'Harriet and Leon Pomerance Fellowship', 'https://www.archaeological.org/grant/pomerance-fellowship/'), (14, 'Kate B. and Hall J. Peterson Fellowships', 'https://www.americanantiquarian.org/short-termfellowship'), (15, 'The Legacy Fellowship', 'https://www.americanantiquarian.org/short-termfellowship'), (16, 'American Antiquarian Society-American Society for 18th-Century Studies Fellowship', 'https://www.americanantiquarian.org/short-termfellowship'), (17, 'Lapidés Fellowship in Pre-1865 Juvenile Literature and Ephemera', 'https://www.americanantiquarian.org/short-termfellowship'), (18, 'Lou Hochberg Thesis and Dissertation Awards', 'http://www.orgonelab.org/hochberg.htm'), (19, 'The John and Mary Stetson Fellowship', 'https://www.orgonelab.org/stetson.htm'), (20, 'The John and Mary Stetson Fellowship', 'https://www.orgonelab.org/stetson.htm'))

### **3. RESEARCH TABLE:**

```

CREATE TABLE Research( Research_ID Varchar(255), Research_Name
VARCHAR(255), University_ID Varchar(255), PRIMARY KEY (Research_ID),
FOREIGN KEY (University_ID) REFERENCES university(University_ID) );

```

```

for i, row in Research_df.iterrows():
    cursor.execute("INSERT INTO research values (%s,%s,%s)", (int(row['Research_ID']),row['Research_Name'],row['University_ID']))
    uni_recomm.commit()

cursor.execute("SELECT * from research")
records=cursor.fetchall()

print(records)
uni_recomm.commit()

```

### **4. COURSE TABLE:**

```

CREATE TABLE Course ( ID INT, University_ID VARCHAR(255),
University_Name VARCHAR(255), Course_Name VARCHAR(255), PRIMARY
KEY (ID), FOREIGN KEY (University_ID) REFERENCES
university(University_ID) );

```

```

for i, row in Course_df.iterrows():
    cursor.execute("INSERT INTO course values (%s,%s,%s)", (int(row['ID']),row['University_ID'],row['Course_Name']))
    uni_recomm.commit()

cursor.execute("SELECT * from course")
records=cursor.fetchall()

print(records)
uni_recomm.commit()

```

((1, '1', 'Northeastern University', '3-D Animation'), (2, '1', 'Northeastern University', 'Accounting'), (3, '1', 'Northeastern University', 'Accounting and Business Administration (MS/MBA)'), (4, '1', 'Northeastern University', 'Accounting and Financial Decision Making'), (5, '1', 'Northeastern University', 'Adult-Gerontology Acute Care Nursing'), (6, '1', 'Northeastern University', 'Adult-Gerontology Acute Care Nursing (CAGS)'), (7, '1', 'Northeastern University', 'Adult-Gerontology Primary Care Nursing (CAGS)'), (8, '1', 'Northeastern University', 'Advanced and Intelligent Manufacturing'), (9, '1', 'Northeastern University', 'Agile Project Management'), (10, '1', 'Northeastern University', 'Analytics'), (11, '1', 'Northeastern University', 'Applied Analytics'), (12, '1', 'Northeastern University', 'master-of-sports-leadership-boston-267'), (13, '1', 'Northeastern University', 'master-of-sports-leadership-online-268'), (14, '1', 'Northeastern University', 'master-of-sports-leadership-charlotte-14376'), (15, '1', 'Northeastern University', 'graduate-certificate-in-supply-chain-engineering-management-14430'), (16, '1', 'Northeastern University', 'graduate-certificate-in-supply-chain-management-boston-14306'), (17, '1', 'Northeastern University', 'graduate-certificate-in-supply-chain-management-online-14307'), (18, '1', 'Northeastern University', 'graduate-certificate-in-sustainability-and-business-18881'), (19, '1', 'Northeastern University', 'graduate-certificate-in-sustainability-and-climate-change-policy-18771'), (20, '1', 'Northeastern University', 'graduate-certificate-in-sustainability-engineering-19153'), (21, '1', 'Northeastern University', 'master-of-science-in-sustainable-building-systems-5284'), (22, '1', 'Northeastern University', 'graduate-certificate-in-sustainable-energy-systems-14426'), (23, '1', 'Northeastern University', 'master-of-design-in-sustainable-urban-environments-one-year-program-5232'), (24, '1', 'Northeastern University', 'master-of-design-in-sustainable-urban-environments-two-year-program-5233'), (25, '1', 'Northeastern University', 'graduate-certificate-in-technology-systems-management-14362'), (26, '1', 'Northeastern University', 'master-of-science-in-telecommunication-networks-17108'), (27, '1', 'Northeastern University', 'transitional-doctor-of-physical-therapy-online-243'), (28, '1', 'Northeastern University', 'graduate-certificate-in-sustainability-and-business-18881'))

### **5. PROFESSOR TABLE:**

```

CREATE TABLE Professor ( ID INT, University_ID VARCHAR(255),
Professor_Names VARCHAR(255), PRIMARY KEY (ID), FOREIGN KEY
(University_ID) REFERENCES university(University_ID) );

```

```

for i,row in Professor_df.iterrows():
    cursor.execute("INSERT INTO Professor values (%s,%s,%s)", (int(row['ID']),row['University_ID'],row['Professors']))
uni_recomm.commit()

cursor.execute("SELECT * from Professor")
records=cursor.fetchall()

print(records)
uni_recomm.commit()

```

## 6. STUDENT TABLE:

```

CREATE TABLE Student ( student_id INT, University_Name varchar(255),
University_ID varchar(255), Acceptance VARCHAR(255), Program
VARCHAR(255), Target_Major VARCHAR(255), Term VARCHAR(255),
Year_of_Joining INT, GRE_Q INT, GRE_V INT, GRE_Total INT, GRE_AWA INT,
TOEFL INT, Undergrad_University VARCHAR(255), Undergrad_Major
VARCHAR(255), CGPA Float, CGPA_Scale Float, Work_Ex INT, PRIMARY KEY
(student_id), FOREIGN KEY (University_ID) REFERENCES University
(University_ID) );

```

```

for i,row in Student_df.iterrows():
    cursor.execute("INSERT INTO student values (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)", (int(row['Student_ID']),row['University_ID'],row['Acceptance'],row['Program'],row['Target_Major'],row['Term'],row['Year_of_Joining'],row['GRE_Q'],row['GRE_V'],row['GRE_Total'],row['GRE_AWA'],row['TOEFL'],row['Undergrad_University'],row['Undergrad_Major'],row['CGPA'],row['CGPA_Scale'],row['Work_Ex']))
uni_recomm.commit()

cursor.execute("SELECT * from student")
records=cursor.fetchall()

print(records)
uni_recomm.commit()

```

((1, 'Northeastern University', '1', 'Admit', 'MS', 'Computer Science', 'Fall ', '2012', '157', '142', '299', '3.5', '97', 'University of Mumbai', 'Computer Engineering', '64.37', '100', '0'), (2, 'Northeastern University', '1', 'Admit', 'MS', 'Industrial Engineering', 'Fall ', '2011', 'None', 'None', '0', 'None', '0', '0', '0', '0'), (3, 'Northeastern University', '1', 'Admit', 'MS', 'Industrial Engineering', 'Fall ', '2011', '770', '460', '1230', '3', '105', 'fr.c.r.c.e.(bandra)', 'production engineering', '75.36', '100', '0'), (4, 'Northeastern University', '1', 'Admit', 'MS', 'Computer Science', 'Fall ', '2013', '159', '152', '311', '3', '94', 'None', 'CS', '3.3', '10', '0'), (5, 'Northeastern University', '1', 'Admit', 'MS', 'Management Information System', 'Fall ', '2014', '158', '144', '302', '3', '100', 'Apeejay College of Engineering', 'ECE', '66.4', '100', '0'), (6, 'Northeastern University', '1', 'Admit', 'MS', 'Computer Science', 'Fall ', '2013', '162', '155', '317', '4', 'None', 'RNSIT', 'CS', '72', '100', '0'), (7, 'Northeastern University', '1', 'Admit', 'MS', 'Computer Science', 'Fall ', '2013', '160', '154', '314', '3.5', '108', 'Sri Bhagawan Mahaveer Jain College of Engineering', 'Information Science', '73', '100', '0'), (8, 'Northeastern University', '1', 'Admit', 'MS', 'Comp Science/Comp Networks/Comp Engg', 'Fall ', '2012', '760', '650', '1410', '3.5', '104', 'Sardar Patel College of Engineering', 'Computer Engineering', '75.9', '100', '0'), (9, 'Northeastern University', '1', 'Admit', 'MS', 'Computer Engineering / Computer Networking / Computer Science', 'Fall ', '2015', '164', '152', '316', 'None', '97', 'None', '0', '8.52', '10', '56'), (10, 'Northeastern University', '1', 'Admit', 'MS', 'MIS', 'Fall ', '2011', '760', '680', '1440', '4', '111', 'Fr CRIT Mumbai University', 'Mechanical Engineering', '54.6', '100', '0'), (11, 'Northeastern University', '1', 'Admit', 'MS', 'Information Security', 'Fall ', '2009', '740', '54.0', '1280', '5', '113', 'VTU', 'CS', '67', '100', '0'), (12, 'Northeastern University', '1', 'Admit', 'MS', '(MIS / MSIM / MIS / MSIT)', 'Fall ', '2015', '157', '146', '303', '3', '94', 'IET DAVV', 'Computer Engineering', '78.28', '100', '66'), (13, 'Northeastern University', '1', 'Admit', 'MS', 'Electronics and Communication', 'Fall ', '2012', '780', '590', '1370', '3.5')

## 7. RECRUITER TABLE:

```

CREATE TABLE Recruiter( Recruiter_ID Varchar(255), Recruiter_Name
Varchar(255), Recruiter_Ranking INT, Number_of_LCA INT, Average_Salary
INT, Max_Hiring_from_University Varchar(255), University_ID Varchar(255),
PRIMARY KEY (Recruiter_ID), FOREIGN KEY (University_ID) REFERENCES
university(University_ID) );

```

```

for i, row in Recruiter_df.iterrows():
    cursor.execute("INSERT INTO recruiter values (%s,%s,%s,%s,%s,%s)", (int(row['Recruiter_ID']),row['Recruiter_Name'],row['Recommender'],row['Company'],row['Industry'],row['JobTitle']))
    uni_recomm.commit()

cursor.execute("SELECT * from recruiter")
records=cursor.fetchall()

print(records)
uni_recomm.commit()

```

((1, 'Cognizant Technology Solutions', 1, 12681, '\$92,766', 'Texas A&M University', '32'), ('10', 'Hcl America', 10, 4113, '\$99,754', 'San Jose State University', '25'), ('11', 'Accenture', 11, 3830, '\$140,893', 'New York University', '20'), ('12', 'IBM', 12, 3800, '\$126,521', 'Carnegie Mellon University', '59'), ('13', 'Deloitte Consulting', 13, 3756, '\$119,341', 'University of Illinois Urbana Champaign', '41'), ('14', 'Intel', 14, 3367, '\$127,627', 'Stanford University', '26'), ('15', 'Capgemini', 15, 3325, '\$102,410', 'Northeastern University', '1'), ('16', 'Wal-Mart Associates', 16, 3218, '\$133,510', 'University of California Los Angeles', '35'), ('17', 'Jpmorgan Chase', 17, 2987, '\$133,103', 'University of Phoenix\x00', '60'), ('18', 'Wipro', 18, 2858, '\$80,871', 'University of Texas, Austin', '50'), ('19', 'Tekorg', 19, 2755, '\$80,529', 'University of Illinois Chicago', '62'), ('2', 'Amazon', 2, 11486, '\$126,163', 'Ohio State University', '65'), ('20', 'CompuNet Software Group', 20, 2231, '\$108,016', 'Georgia Institute of Technology', '14'), ('21', 'Qualcomm Technologies', 21, 2167, '\$152,754', 'Stanford University', '26'), ('22', 'Salesforce.com', 22, 2164, '\$141,276', 'University of Illinois, Urbana Champaign', '41'), ('23', 'Tech Mahindra', 23, 2107, '\$94,298', 'Rochester Institute of Technology', '23'), ('24', 'Cisco Systems', 24, 2013, '\$145,750', 'University of Southern California', '48'), ('25', 'Goldman Sachs &', 25, 1579, '\$134,689', 'Harvard University', '15'), ('3', 'Tata Consultancy Services', 3, 9822, '\$93,484', 'Indiana University Bloomington', '17'), ('4', 'Google', 4, 9421, '\$156,793', 'Stanford University', '26'), ('5', 'Microsoft', 5, 7329, '\$146,569', 'University of Washington', '65'), ('6', 'Facebook', 6, 6090, '\$175,750', 'University of California, Berkeley', '58'), ('7', 'Ernst & Young', 7, 5797, '\$118,653', 'University of Southern California', '48'), ('8', 'Infosys', 8, 5649, '\$92,689', 'University of Texas Dallas', '51'), ('9', 'Apple', 9, 4239, '\$168,416',

As seen below: the tables had multi-valued rows and there was a transitive dependency in the tables.

**Result Grid** | Filter Rows: Export: Wrap Cell Content:

Professors	Course_Name	University_Name
Rebecca Alber, Marvin C. Akin, Walter Allen, Melissa Sach... Urban and Regional Planning		University of California, San Diego
Rebecca Alber, Marvin C. Akin, Walter Allen, Melissa Sach... Urban and Regional Planning - Institut d'Etudes de Paris		University of California, San Diego
Rebecca Alber, Marvin C. Akin, Walter Allen, Melissa Sach... Urban Planning		University of California, San Diego
Rebecca Alber, Marvin C. Akin, Walter Allen, Melissa Sach... Urban Planning Department		University of California, San Diego
Rebecca Alber, Marvin C. Akin, Walter Allen, Melissa Sach... World Arts and Cultures/Dance Department		University of California, San Diego
Abadi Daniel J., Abbasi Hosseini, Abed Eyad, Abera Nicole Taylor,... Applied Thanatology (Graduate Certificate)		University of Michigan, Ann Arbor
Abadi Daniel J., Abbasi Hosseini, Abed Eyad, Abera Nicole Taylor,... Biochemistry (MS, PhD, MD/PhD)		University of Michigan, Ann Arbor
Abadi Daniel J., Abbasi Hosseini, Abed Eyad, Abera Nicole Taylor,... Biomedical Entrepreneurship (Graduate Certificate)		University of Michigan, Ann Arbor
Abadi Daniel J., Abbasi Hosseini, Abed Eyad, Abera Nicole Taylor,... Biomedical Sciences (Dental School)		University of Michigan, Ann Arbor
Abadi Daniel J., Abbasi Hosseini, Abed Eyad, Abera Nicole Taylor,... Cellular and Molecular Biomedical Science (MS)		University of Michigan, Ann Arbor

student_id	Target_University	Target_Major	Year_of_Joining	GRE_Total	TOEFL	University_GRE_Cutoff	University_TOEFL_cutoff	accept
22	Northeastern University	Computer Science	2016	323	116	310	100	Admit
147	Northeastern University	MIS	2016	307	91	310	100	Admit
255	Northeastern University	MIS	2016	306	100	310	100	Admit
377	Northeastern University	Industrial Engineering	2016	321	111	310	100	Admit
405	Northeastern University	Computer Science	2016	321	101	310	100	Admit
446	Northeastern University	Computer Science	2016	315	98	310	100	Admit
585	Northeastern University	Engineering Management	2016	319	108	310	100	Admit
654	Northeastern University	Computer Science	2016	314	110	310	100	Admit
724	Northeastern University	Engineering Management	2016	310	106	310	100	Admit
937	Northeastern University	Computer Engineering	2016	323	None	310	100	Admit
954	Northeastern University	Computer Science	2016	314	112	310	100	Admit

Therefore, we followed the rules of Normalization as follows:

1. **First normal form (1NF)**
  - Each table has a primary key: minimal set of attributes which can uniquely identify a record
  - The values in each column of a table are atomic (No multi-value attributes allowed).
  - There are no repeating groups: two columns do not store similar information in the same table
2. **Second normal form (2NF)**
  - All requirements for 1st NF must be met.
  - No partial dependencies.

- No calculated data
- 3. Third normal form (3NF)**
- All requirements for 2nd NF must be met.
  - Eliminate fields that do not directly depend on the primary key; that is no transitive dependencies

## CREATE STATEMENTS:

**1. University\_Details:**

```
CREATE TABLE University_Details
(University_ID varchar(255),
University_Name varchar(255),
PRIMARY KEY (University_ID),
FOREIGN KEY (University_ID) REFERENCES university(University_ID)
);
```

**2. Student Table:**

```
CREATE TABLE Student
(student_ID int,
University_ID Varchar(255),
Acceptance varchar(255),
Program varchar(255),
Target_Major Varchar(255),
Term varchar(255),
Year_of_Joining varchar(255),
PRIMARY KEY (student_ID),
FOREIGN KEY (University_ID) REFERENCES university(University_ID)
);
```

**3. Student\_Score:**

```
CREATE TABLE Student_Score
(student_ID int,
GRE_Q varchar(255),
GRE_V varchar(255),
GRE_Total varchar(255),
GRE_AWA varchar(255),
TOEFL varchar(255),
Work_Ex varchar(255),
PRIMARY KEY (student_ID),
FOREIGN KEY (student_ID) REFERENCES student(student_ID)
);
```

**4. Recruiter\_Details:**

```
CREATE TABLE Recruiter_Details
(Recruiter_ID varchar(255),
Recruiter_Name varchar(255),
PRIMARY KEY (Recruiter_ID),
FOREIGN KEY (Recruiter_ID) REFERENCES recruiter(Recruiter_ID)
);
```

**5. Scholarship:**

```
CREATE TABLE Scholarship
```

```

(Scholarship_ID int,
Scholarship_Name varchar(255),
Scholarship_URL longtext,
University_ID varchar(255),
PRIMARY KEY (Scholarship_ID),
FOREIGN KEY (University_ID) REFERENCES university(University_ID)
);

```

**6. Student\_Undergrad:**

```

CREATE TABLE Student_Undergrad
(student_ID int,
Undergrad_University Varchar(255),
Undergrad_Major varchar(255),
CGPA varchar(255),
CGPA_Scale Varchar(255),
PRIMARY KEY (student_ID),
FOREIGN KEY (student_ID) REFERENCES student(student_ID)
);

```

**7. University\_Admit\_Requirement:**

```

CREATE TABLE University_Admit_Requirement
(University_ID VARCHAR(255),
GRE Varchar(255),
toefl varchar(255),
ielts varchar(255),
PRIMARY KEY (University_ID),
FOREIGN KEY (University_ID) REFERENCES university(University_ID)
);

```

The rest of the create statements are same as before. Please find the table description below:

**8. Course**

	Field	Type	Null	Key	Default	Extra
▶	ID	int	NO	PRI	NULL	
	University_ID	varchar(255)	YES	MUL	NULL	
	Course_Name	varchar(255)	YES		NULL	

**9. Professor:**

	Field	Type	Null	Key	Default	Extra
▶	ID	int	NO	PRI	NULL	
	University_ID	varchar(255)	YES	MUL	NULL	
	Professors	longtext	YES		NULL	

**10. Recruiter:**

	Field	Type	Null	Key	Default	Extra
▶	Recruiter_ID	varchar(255)	NO	PRI	NULL	
	Recruiter_Ranking	int	YES		NULL	
	Number_of_LCA	int	YES		NULL	
	Average_Salary	varchar(255)	YES		NULL	
	University_ID	varchar(255)	YES	MUL	NULL	

## 11. Research

	Field	Type	Null	Key	Default	Extra
▶	Research_ID	varchar(255)	NO	PRI	NULL	
	Research_Name	longtext	YES		NULL	
	University_ID	varchar(255)	YES	MUL	NULL	

## 12. University

	Field	Type	Null	Key	Default	Extra
▶	University_ID	varchar(255)	NO	PRI	NULL	
	avg_pay_scale	varchar(255)	YES		NULL	
	Duration	varchar(255)	YES		NULL	
	Term	varchar(255)	YES		NULL	
	Year_of_Joining	int	YES		NULL	
	Major	varchar(255)	YES		NULL	
	Avg_Fees	varchar(255)	YES		NULL	

After Inserting in the new table:

```
1      select * from recruiter;
```



The screenshot shows a MySQL Workbench result grid titled "Result Grid". The grid displays 20 rows of data from a table named "recruiter". The columns are labeled: Recruiter\_ID, Recruiter\_Ranking, Number\_of\_LCA, Average\_Salary, and University\_ID. The data is as follows:

	Recruiter_ID	Recruiter_Ranking	Number_of_LCA	Average_Salary	University_ID
▶	1	1	12681	\$92,766	32
	10	10	4113	\$99,754	25
	11	11	3830	\$140,893	20
	12	12	3800	\$126,521	59
	13	13	3756	\$119,341	41
	14	14	3367	\$127,627	26
	15	15	3325	\$102,410	1
	16	16	3218	\$133,510	35
	17	17	2987	\$133,103	60
	18	18	2858	\$80,871	50
	19	19	2755	\$80,529	62
	2	2	11486	\$126,163	65
	20	20	2231	\$108,016	14

```
1 select * from research;
```

	Research_ID	Research_Name	University_ID
▶	1	Environmental Economics and Policy	15
	10	School of Life Sciences Undergraduate Researc...	6
	11	Sustainability Undergraduate Research Experie...	6
	12	Watts College of Public Service and Community...	6
	13	Ira A. Fulton Schools of Engineering Summer R...	6
	14	Web & Visualization Developer (Media Cloud Pro...	1
	15	Research in Programming Languages and Machi...	1
	16	Multiple research options on experimental multi...	1
	17	Imaging microbiome interactions: Microfluidics	1
	18	Microscopy	1
	19	Molecular Biology	1
	2	Investigating the Mold Resiliency of Buildings in...	15
	20	and Automation	1

1 select \* from student;

	student_ID	University_ID	Acceptance	Program	Target_Major	Term	Year_of_Joining
▶	1	1	Admit	MS	Computer Science	Fall	2012
	2	1	Admit	MS	Industrial Engineering	Fall	2011
	3	1	Admit	MS	Industrial Engineering	Fall	2011
	4	1	Admit	MS	Computer Science	Fall	2013
	5	1	Admit	MS	Management Information System	Fall	2014
	6	1	Admit	MS	Computer Science	Fall	2013
	7	1	Admit	MS	Computer Science	Fall	2013
	10	1	Admit	MS	MIS	Fall	2011
	11	1	Admit	MS	Information Security	Fall	2009
	13	1	Admit	MS	Electronics and Communication	Fall	2012
	15	1	Admit	MS	Information Security	Fall	2013
	16	1	Admit	MS	Computer Science	Fall	2012
	17	1	Admit	MS	Computer Science	Fall	2015

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a query editor window containing the SQL command:

```
1   select * from university_admit_requirement;
```

Below the query editor is a results grid titled "Result Grid". The grid displays the following data:

	University_ID	GRE	toefl	ielts
▶	1	310	100	7.5
	10	295	78	6
	11	295	100	7
	12	300	92	7
	13	Waived	88	7
	14	Waived	79	6.5
	15	Waived	80	6.5
	16	309	90	6.5
	17	Waived	100	7
	18	316	79	6.5
	19	310	79	6.5
	2	300	100	7.5
	20	321	100	7

## DATA CLEANING EXAMPLES:

- Standardizing the data

```
In [64]: cursor.execute("update student_undergrad set CGPA = CGPA/10 where CGPA_Scale = 100")
records=cursor.fetchall()

print(records)
uni_recomm.commit()

()
```

```
In [65]: cursor.execute("update student_undergrad set CGPA_Scale = 10 where CGPA_Scale = 100")
records=cursor.fetchall()

print(records)
uni_recomm.commit()

()
```

- Altering the dataset to suit the table

```
In [167]: cursor.execute("ALTER Table University drop column University_Name")
for x in cursor:
    print(x)
```

```
In [168]: cursor.execute("select * from university")
for x in cursor:
    print(x)
```

```
cursor.execute("ALTER Table recruiter drop column Recruiter_Name")
for x in cursor:
    print(x)
```

```
cursor.execute("select * from recruiter")
for x in cursor:
    print(x)
```

```
cursor.execute("ALTER Table Student drop column University_Name")
for x in cursor:
    print(x)
```

```
cursor.execute("select * from recruiter")
for x in cursor:
    print(x)
```

```
In [217]: cursor.execute("delete from student where Target_Major like '%/%'")
```

```
Out[217]: 467
```

```
In [220]: cursor.execute("SELECT * from student")
records=cursor.fetchall()

print(records)
uni_recomm.commit()
```

## USECASES w.r.t New Tables:

1. Display the list of all the courses offered by a university along with the list of all the professors present in the university.

**Description:** This query returns a list of professors along with their university ID and the courses offered by that university

**Actor:** Student

**Precondition:** The student must select a professor

**Steps:**

**Actor Action:** The student views the professor teaching a particular course from a university

**System Response:** The system displays a list of all the professors from a university along with the professors

**Postcondition:** A list is generated of all the professors at a university along with their courses

### SQL Query:

```
select P.Professors, C.Course_Name, UD.University_Name  
from course C, professor P, university_details UD  
where P.University_ID = C.University_ID and P.university_ID = UD.University_ID  
and UD.University_ID = C.University_ID
```

```
1  select P.Professors, C.Course_Name, UD.University_Name  
2  from course C, professor P, university_details UD  
3  where P.University_ID = C.University_ID and P.university_ID = UD.University_ID  
4  and UD.University_ID = C.University_ID
```

Professors	Course_Name	University_Name
Mehdi Abedi	3-D Animation	Northeastern University
Emad Aboelela	3-D Animation	Northeastern University
Gregory D. Abowd	3-D Animation	Northeastern University
Kal Bugrara	3-D Animation	Northeastern University
Vishal Chawla	3-D Animation	Northeastern University
Nick Brown	3-D Animation	Northeastern University
Daniel Peters	3-D Animation	Northeastern University
Mehdi Abedi	Accounting	Northeastern University
Emad Aboelela	Accounting	Northeastern University
Gregory D. Abowd	Accounting	Northeastern University
Kal Bugrara	Accounting	Northeastern University

2. Search for a distinct research opportunity (for example – Machine Learning) available in a particular university along with the list of all the professors of that university.

**Description:** This query returns a list of ongoing research in a university under a professor

**Actor:** Student

**Precondition:** The student must select research opportunities

**Steps:**

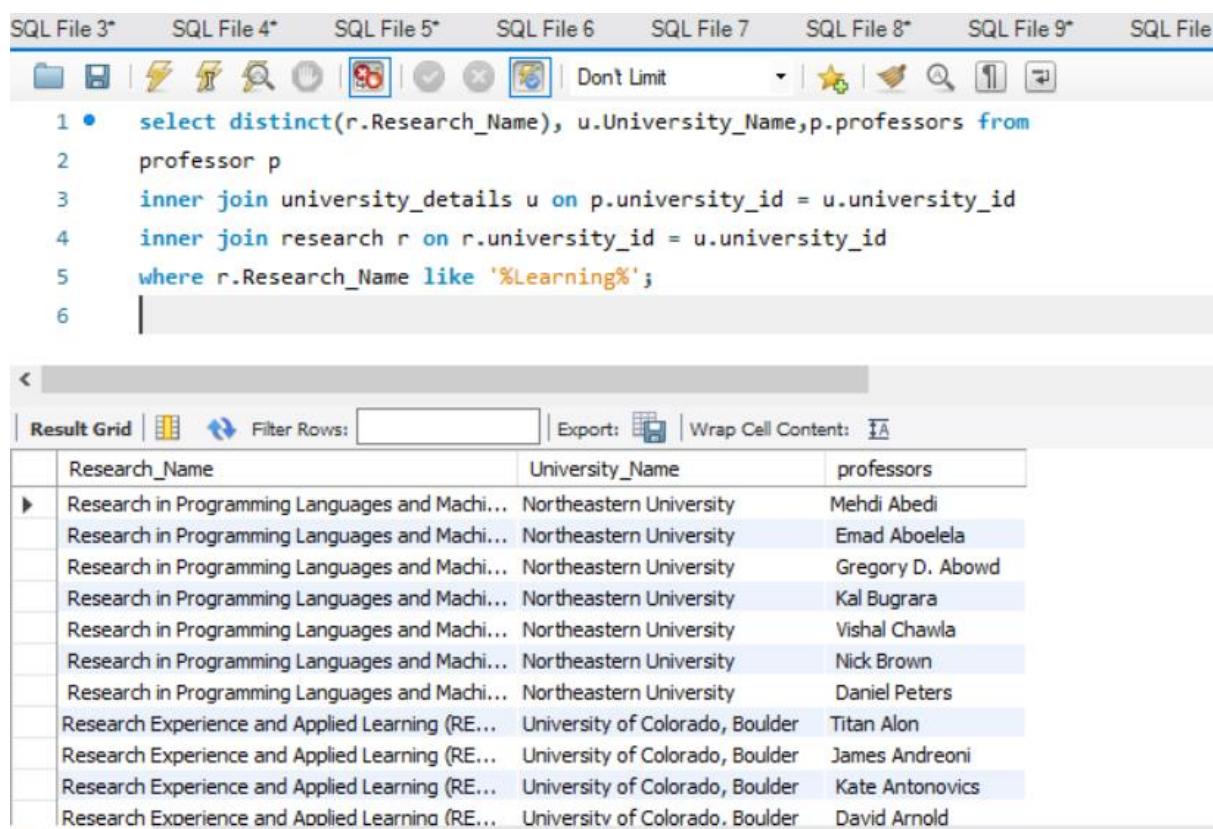
**Actor Action:** The student views the ongoing research from a university

**System Response:** The system displays all the ongoing research from a university under the professor

**Postcondition:** The system generates a list of ongoing research

**SQL Query:**

```
select distinct(r.Research_Name), u.University_Name,p.professors from
professor p
inner join university_details u on p.university_id = u.university_id
inner join research r on r.university_id = u.university_id
where r.Research_Name like '%Learning%';
```



The screenshot shows a SQL query editor interface with the following details:

- Toolbar: SQL File 3\*, SQL File 4\*, SQL File 5\*, SQL File 6, SQL File 7, SQL File 8\*, SQL File 9\*, SQL File.
- Query Editor:

```
1 • select distinct(r.Research_Name), u.University_Name,p.professors from
2 professor p
3 inner join university_details u on p.university_id = u.university_id
4 inner join research r on r.university_id = u.university_id
5 where r.Research_Name like '%Learning%';
6 |
```
- Result Grid:

	Research_Name	University_Name	professors
▶	Research in Programming Languages and Machi...	Northeastern University	Mehdi Abedi
	Research in Programming Languages and Machi...	Northeastern University	Emad Aboelela
	Research in Programming Languages and Machi...	Northeastern University	Gregory D. Abowd
	Research in Programming Languages and Machi...	Northeastern University	Kal Bugrara
	Research in Programming Languages and Machi...	Northeastern University	Vishal Chawla
	Research in Programming Languages and Machi...	Northeastern University	Nick Brown
	Research in Programming Languages and Machi...	Northeastern University	Daniel Peters
	Research Experience and Applied Learning (RE...	University of Colorado, Boulder	Titan Alon
	Research Experience and Applied Learning (RE...	University of Colorado, Boulder	James Andreoni
	Research Experience and Applied Learning (RE...	University of Colorado, Boulder	Kate Antonovics
	Research Experience and Applied Learning (RE...	University of Colorado, Boulder	David Arnold

**3. Display the recruiter ranking along with the average salary and the university from which it hires the maximum number of students**

**Description:** Display a list of recruiters along with the average salary that they offer and the maximum number of students they hire from a particular university

**Actor:** Student

**Precondition:** The student must select a list of top recruiters

**Steps:**

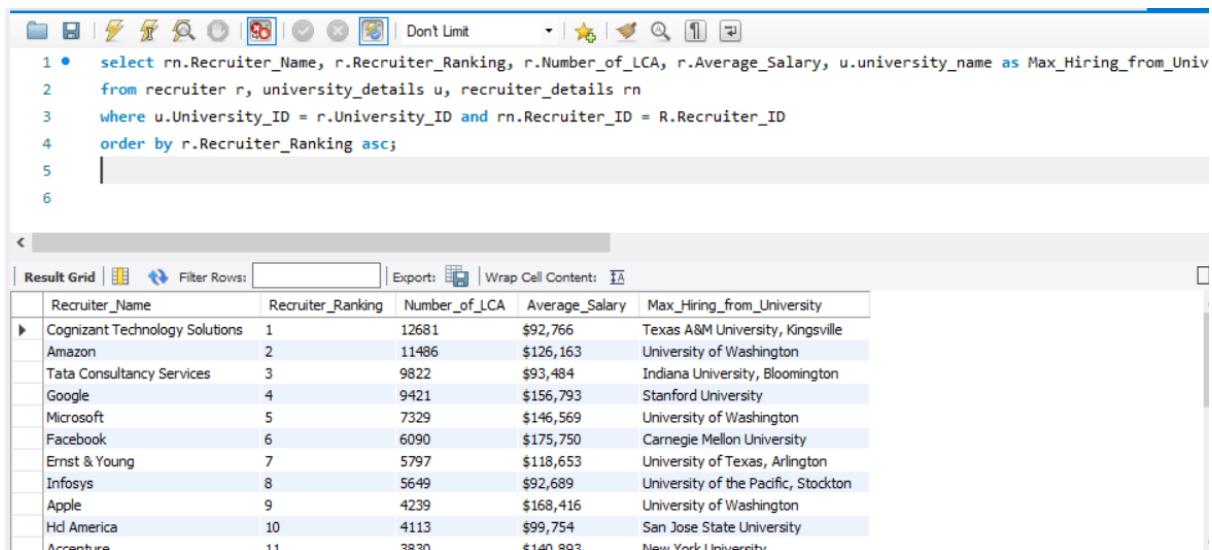
**Actor Action:** The student views the recruiters along with the average package

**System Response:** The system displays a list of all the recruiters and the university from which they hire the maximum number of students

**Postcondition:** The system generates a list of recruiters

**SQL Query:**

```
select rn.Recruiter_Name, r.Recruiter_Ranking, r.Number_of_LCA,
r.Average_Salary, u.university_name as Max_Hiring_from_University
from recruiter r, university_details u, recruiter_details rn
where u.University_ID = r.University_ID and rn.Recruiter_ID = R.Recruiter_ID
order by r.Recruiter_Ranking asc;
```



The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a text editor window containing the SQL query. The query selects recruiter information and university names where the university ID matches the recruiter ID and the recruiter ranking is ascending. The result grid below shows 11 rows of data, each with columns: Recruiter\_Name, Recruiter\_Ranking, Number\_of\_LCA, Average\_Salary, and Max\_Hiring\_from\_University. The data includes entries for Cognizant Technology Solutions, Amazon, Tata Consultancy Services, Google, Microsoft, Facebook, Ernst & Young, Infosys, Apple, Hd America, and Arrestris.

Recruiter_Name	Recruiter_Ranking	Number_of_LCA	Average_Salary	Max_Hiring_from_University
Cognizant Technology Solutions	1	12681	\$92,766	Texas A&M University, Kingsville
Amazon	2	11486	\$126,163	University of Washington
Tata Consultancy Services	3	9822	\$93,484	Indiana University, Bloomington
Google	4	9421	\$156,793	Stanford University
Microsoft	5	7329	\$146,569	University of Washington
Facebook	6	6090	\$175,750	Carnegie Mellon University
Ernst & Young	7	5797	\$118,653	University of Texas, Arlington
Infosys	8	5649	\$92,689	University of the Pacific, Stockton
Apple	9	4239	\$168,416	University of Washington
Hd America	10	4113	\$99,754	San Jose State University
Arrestris	11	3220	\$140,802	New York University

**4. Display the list of all the professors that teach under a particular university.**

**Description:** The student views the professors from a particular university

**Actor:** Student

**Precondition:** The student must select a university

**Steps:**

**Actor Action:** The student views the professor from a particular university

**System Response:** The system generates a list of professors from a university

**Postcondition:** The system generates all the professors

**SQL Query:**

```
select p.Professors, ud.university_name
from professor p inner join university_details ud on p.University_ID =
ud.University_ID
where University_Name = "Northeastern University"
```

The screenshot shows a MySQL Workbench window. At the top, there's a toolbar with various icons. Below the toolbar, an SQL editor window contains the following code:

```

1 • select p.Professors, ud.university_name
2   from professor p inner join university_details ud on p.University_ID = ud.University_ID
3   where University_Name = "Northeastern University"
4

```

Below the SQL editor is a result grid titled "Result Grid". It has two columns: "Professors" and "university\_name". The data is as follows:

Professors	university_name
Mehdi Abedi	Northeastern University
Emad Aboelela	Northeastern University
Gregory D. Abowd	Northeastern University
Kal Bugrara	Northeastern University
Vishal Chawla	Northeastern University
Nick Brown	Northeastern University
Daniel Peters	Northeastern University

## 5. Display the conditions on which a student is admitted to a university along with their graduate major, and average pay scale after graduation

**Description:** Displays the entire academics of a student based on admit

**Actor:** Student

**Precondition:** The student must select a student ID

**Steps:**

**Actor Action:** The student selects a student ID and views the entire academic

**System Response:** The system generates a list of all the admitted students

**Postcondition:** A list is generated with all the amidst

### SQL Query:

```

Select s.student_ID, s.acceptance, ss.GRE_Q,
ss.GRE_V,ss.GRE_AWA,ss.TOEFL, su.Undergrad_University,
su.Undergrad_Major, u.university_name, un.avg_pay_scale, r.university_id
from Student s, University_Details u, university un, recruiter r, student_score
ss, student_undergrad su
where s.university_id = u.university_id
and u.university_id = r.university_id
and un.University_ID = u.University_ID
and ss.student_ID = s.student_ID
and su.student_ID = s.student_ID
and s.acceptance = 'Admit';

```

The screenshot shows a SQL query editor interface with multiple tabs at the top labeled SQL File 3\*, SQL File 4\*, SQL File 5\*, SQL File 6, SQL File 7, SQL File 8\*, SQL File 9\*, SQL File 10\*, SQL File 11\*, SQL File 12\*, and SQL File 13\*. Below the tabs is a toolbar with various icons. A code editor window contains the following SQL query:

```

1 • Select s.student_ID, s.acceptance, ss.GRE_Q, ss.GRE_V, ss.GRE_AWA, ss.TOEFL, su.Undergrad_University, su.Undergrad_Major, u.university_name
2   from Student s, University_Details u, university un, recruiter r, student_score ss, student_undergrad su
3   where s.university_id = u.university_id
4   and u.university_id = r.university_id
5   and un.University_ID = u.University_ID
6   and ss.student_ID = s.student_ID
7   and su.student_ID = s.student_ID
8   and s.acceptance = 'Admit';

```

Below the query is a result grid table with the following data:

student_ID	acceptance	GRE_Q	GRE_V	GRE_AWA	TOEFL	Undergrad_University	Undergrad_Major	university_name	avg_J
1	Admit	157	142	3.5	97	University of Mumbai	Computer Engineering	Northeastern University	10000
2	Admit	None	None	None	None	None	0	Northeastern University	10000
3	Admit	770	460	3	105	fr.c.r.c.e.(bandra)	production engineering	Northeastern University	10000
4	Admit	159	152	3	94	None	CS	Northeastern University	10000
5	Admit	158	144	3	100	Aapeejay College of Engineering	ECE	Northeastern University	10000
6	Admit	162	155	4	None	RNSIT	CS	Northeastern University	10000
7	Admit	160	154	3.5	108	Sri Bhagawan Mahaveer Jain College of Enginee...	Information Science	Northeastern University	10000
10	Admit	760	680	4	111	Fr CRIT Mumbai University	Mechanical Engineering	Northeastern University	10000
11	Admit	740	540	5	113	VTU	CS	Northeastern University	10000
13	Admit	780	590	3.5	114	Manipal Institute of Technology	ECE	Northeastern University	10000

At the bottom left, it says "Result 47". On the right side of the result grid, there are several icons: Result Grid (selected), Form Editor, Field Types, and Read Only.

## 6. Display the list of a specific course offered by all the universities.

**Description:** This query returns a value of all the universities that provide a particular course

**Actor:** Student

**Precondition:** The student must select a course

**Steps:**

**Actor Action:** The student selects a particular course that he wishes to be enrolled for

**System Response:** The system displays a list of all the universities that provide that particular course

**Postcondition:** A list of all the universities that offer analytics courses is displayed

### SQL Query:

```

select c.course_name, u.university_name
from course c inner join university_details u on c.University_ID = u.University_ID
where c.course_name like '%Analytics%';

```

The screenshot shows a SQL database interface with a toolbar at the top and a code editor below it. The code editor contains the following SQL query:

```

1 •  select c.course_name, u.university_name
2   from course c inner join university_details u on c.University_ID = u.University_ID
3   where c.course_name like '%Analytics%';

```

Below the code editor is a result grid table with two columns: "course\_name" and "university\_name". The data is as follows:

course_name	university_name
Analytics	Northeastern University
Applied Analytics	Northeastern University
graduate-certificate-in-urban-analytics-14299	Northeastern University
Masters of Data Analytics	Northeastern University
Big Data Analytics (MS) (SDSU)	Ohio State University
Big Data Analytics (MS) (SDSU GC)	Ohio State University
Management - Business Analytics MS	University of California, San Diego
Business Analytics	University of Maryland Baltimore
Civic Analytics	University of Maryland Baltimore
Biomedical Data Analytics	Arizona State University Tempe
Analytics	Arizona State University Tempe

## 7. Display the data about previous admits where the year of joining >= 2015

**Description:** To view a list of all the previous admits to a university after 2015

**Actor:** The student

**Precondition:** The student must select a year

**Steps:**

**Actor Action:** The student selects the year and displays the admits

**System Response:** The system displays a list of all the previous admits

**Postcondition:** The list is displayed with all the previous admits

### SQL Query:

```

select s.student_id, ud.university_name as
Target_University,s.Target_Major,s.Year_of_Joining,ss.GRE_Total,ss.TOEFL
,ua.GRE as University_GRE_Cutoff,ua.TOEFL as
University_TOEFL_cutoff,s.acceptance
from student s, university u, university_details ud, student_score ss,
university_admit_requirement ua
where u.University_ID = s.University_ID and ud.University_ID =
u.University_ID
and ss.student_ID = s.student_ID and ua.University_ID = ud.University_ID
and s.Year_of_Joining>2015;

```

The screenshot shows a SQL editor interface with multiple tabs at the top labeled SQL File 3\*, SQL File 4\*, SQL File 5\*, SQL File 6, SQL File 7, SQL File 8\*, SQL File 9\*, SQL File 10\*, SQL File 11\*, SQL File 12\*, and SQL File. Below the tabs is a toolbar with icons for file operations like Open, Save, and Print, along with a 'Dont Limit' button. The main area contains a SQL query:

```

1 •   select s.student_id, ud.university_name as Target_University,s.Target_Major,s.Year_of_Joining,ss.GRE_Total,ss.TOEFL,ua.GRE_a
2   from student s, university u, university_details ud, student_score ss, university_admit_requirement ua
3   where u.University_ID = s.University_ID and ud.University_ID = u.University_ID
4   and ss.student_ID = s.student_ID and ua.University_ID = ud.University_ID
5   and s.Year_of_Joining>2015;
6

```

Below the query is a result grid titled 'Result Grid' with columns: student\_id, Target\_University, Target\_Major, Year\_of\_Joining, GRE\_Total, TOEFL, University\_GRE\_Cutoff, University\_TOEFL\_cutoff, and acceptance. The data shows 49 rows of student admissions information.

	student_id	Target_University	Target_Major	Year_of_Joining	GRE_Total	TOEFL	University_GRE_Cutoff	University_TOEFL_cutoff	acceptance
▶	22	Northeastern University	Computer Science	2016	323	116	310	100	Admit
	147	Northeastern University	MIS	2016	307	91	310	100	Admit
	255	Northeastern University	MIS	2016	306	100	310	100	Admit
	377	Northeastern University	Industrial Engineering	2016	321	111	310	100	Admit
	405	Northeastern University	Computer Science	2016	321	101	310	100	Admit
	446	Northeastern University	Computer Science	2016	315	98	310	100	Admit
	585	Northeastern University	Engineering Management	2016	319	108	310	100	Admit
	654	Northeastern University	Computer Science	2016	314	110	310	100	Admit
	724	Northeastern University	Engineering Management	2016	310	106	310	100	Admit
	937	Northeastern University	Computer Engineering	2016	323	None	310	100	Admit
	954	Northeastern University	Computer Science	2016	314	112	310	100	Admit
Result 49									

## 8. Display the list of courses choices based on specifications a student can select from post getting the admit.

**Description:** To view a list of all the possible choices student has

**Actor:** Student

**Precondition:** The student must select a course name for acceptance

**Steps:**

**Actor Action:** The student must select a course name

**System Response:** A list is displayed with all the possible admits a student has from a university

**Postcondition:** A list is displayed with all possible universities

### SQL Query:

```
select s.student_id, ud.university_name,s.Acceptance,c.Course_Name from
student s join course c on c.University_ID = s.University_ID join
university_details ud on ud.university_id = c.University_ID and s.Acceptance
= 'Admit' and c.Course_Name like '%Engineering%';
```

The screenshot shows a database interface with a query editor at the top and a result grid below. The query is:

```

1 • select s.student_id, ud.university_name,s.Acceptance,c.Course_Name
2   from student s join course c on c.University_ID = s.University_ID join university_details ud on ud.university_id = c.University_ID
3   and s.Acceptance = 'Admit' and c.Course_Name like '%Engineering%';

```

The result grid displays the following data:

student_id	university_name	Acceptance	Course_Name
1	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...
2	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...
3	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...
4	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...
5	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...
6	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...
7	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...
10	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...
11	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...
13	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...
15	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineering...

## 9. Display the list of desired research opportunities available in a university based on the course selection of a student.

**Description:** To view the list of all the research in a particular course

**Actor:** Student

**Precondition:** The student must select a research domain say Machine Learning

**Steps:**

**Actor Action:** The student must select a domain for research

**System Response:** The system displays a list of all the research in a particular course

**Postcondition:** A list is displayed with all the research going on in a course

**SQL Query:**

```

select r.research_name, c.course_name from research r inner join course c
where r.University_ID = c.University_ID and r.Research_Name like
'%Machine Learning%' and course_name in (select Course_Name from
course where course_name like '%Analytics%');

```

The screenshot shows a SQL query editor interface with multiple tabs at the top labeled SQL File 3\*, SQL File 4\*, SQL File 5\*, SQL File 6, SQL File 7, SQL File 8\*, SQL File 9\*, SQL File 10\*, SQL File 11\*, SQL File 12\*, and SQL File 1. The SQL File 1 tab is active. Below the tabs is a toolbar with various icons. The main area contains a SQL query:

```

1 • select r.research_name, c.course_name
2   from research r inner join course c
3     where r.University_ID = c.University_ID
4     and r.Research_Name
5       like '%Machine Learning%'
6     and course_name in (select Course_Name from course where course_name like '%Analytics%');

```

Below the query is a result grid titled "Result Grid". It has two columns: "research\_name" and "course\_name". The data in the grid is as follows:

research_name	course_name
Research in Programming Languages and Machi...	Analytics
Research in Programming Languages and Machi...	Applied Analytics
Research in Programming Languages and Machi...	graduate-certificate-in-urban-analytics-14299
Research in Programming Languages and Machi...	Masters of Data Analytics

## 10. Display the list of research opportunities available in an university based on the location of the university.

**Description:** To display a list of research going in a particular location in which the university exists

**Actor:** Student

**Precondition:** The student must select a location

**Steps:**

**Actor Action:** The student selects a location to look for research

**System Response:** The system displays a list of universities in that location where research is going on

**Postcondition:** A list is displayed with the research going in a particular location in which the university exists

**SQL Query:**

```

select r.research_name, ud.university_name
from research r, university_details ud, university u
where r.university_id = ud.university_id
and u.Location = "Boston";

```

SQL File 3\* SQL File 4\* SQL File 5\* SQL File 6 SQL File 7 SQL File 8\* SQL File 9\* SQ

1 • select r.research\_name, ud.university\_name  
2 from research r, university\_details ud, university u  
3 where r.university\_id = ud.university\_id  
4 and u.Location = "Boston";  
5  
6 |

Result Grid | Filter Rows: Export: Wrap Cell Content:

	research_name	university_name
▶	New College Undergraduate Inquiry and Resea...	Arizona State University Tempe
	Fulton Undergraduate Research Initiative (FURI)	Arizona State University Tempe
	Climate Dynamics	Harvard University
	Modeling of Atmospheric Chemistry	Harvard University
	Aging and Climate Change in China	Harvard University
	Office of Research Advancement	University of Texas, Arlington
	Office for the Protection of Research Subjects	University of Texas, Arlington
	Institutional Animal Care and Use Committee	University of Texas, Arlington
	Department of Contracts and Grants	University of Texas, Arlington
	Department of Animal Resources	University of Texas, Arlington
	Targeted detection of allergies in Quinoa Flour ...	University of Maryland College...

## VIEWS w.r.t New Tables:

1. Display the list of all the courses offered by a university along with the list of all the professors present in the university.

**Description:** This query returns a list of professors along with their university ID and the courses offered by that university

**Actor:** Student

**Precondition:** The student must select a professor

**Steps:**

**Actor Action:** The student views the professor teaching a particular course from a university

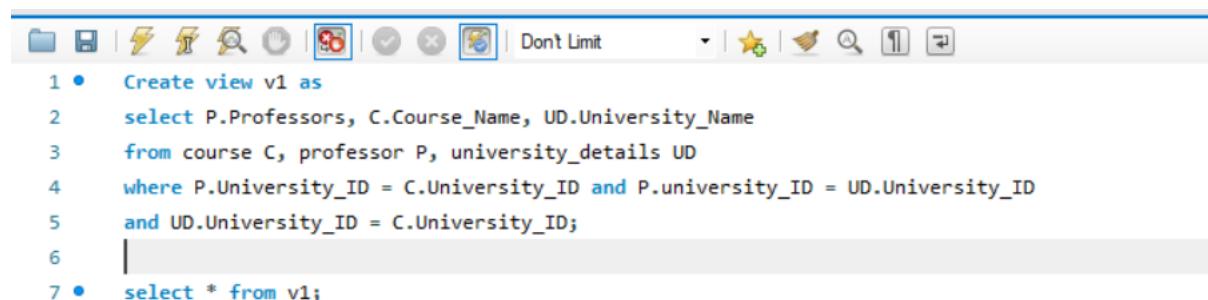
**System Response:** The system displays a list of all the professors from a university along with the professors

**Postcondition:** A list is generated of all the professors at a university along with their courses

## SQL Query:

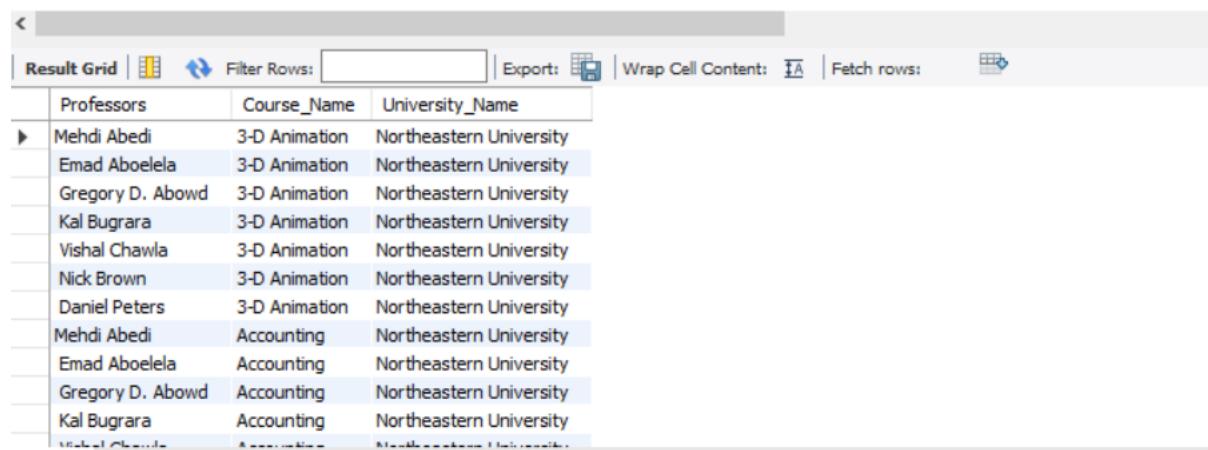
Create view v1 as

```
select P.Professors, C.Course_Name, UD.University_Name  
from course C, professor P, university_details UD  
where P.University_ID = C.University_ID and P.university_ID = UD.University_ID  
and UD.University_ID = C.University_ID
```



The screenshot shows a SQL query editor interface. At the top, there's a toolbar with various icons. Below the toolbar, the SQL code for creating a view is displayed in a step-by-step manner, with each line numbered from 1 to 7. Lines 1 through 6 show the creation of the view and its definition, while line 7 shows the execution of the query. The code is as follows:

```
1 • Create view v1 as  
2     select P.Professors, C.Course_Name, UD.University_Name  
3         from course C, professor P, university_details UD  
4     where P.University_ID = C.University_ID and P.university_ID = UD.University_ID  
5         and UD.University_ID = C.University_ID;  
6  
7 • select * from v1;
```



The screenshot shows a result grid from a database query. The grid has three columns: 'Professors', 'Course\_Name', and 'University\_Name'. The data is as follows:

	Professors	Course_Name	University_Name
▶	Mehdi Abedi	3-D Animation	Northeastern University
	Emad Aboelela	3-D Animation	Northeastern University
	Gregory D. Abowd	3-D Animation	Northeastern University
	Kal Bugrara	3-D Animation	Northeastern University
	Vishal Chawla	3-D Animation	Northeastern University
	Nick Brown	3-D Animation	Northeastern University
	Daniel Peters	3-D Animation	Northeastern University
	Mehdi Abedi	Accounting	Northeastern University
	Emad Aboelela	Accounting	Northeastern University
	Gregory D. Abowd	Accounting	Northeastern University
	Kal Bugrara	Accounting	Northeastern University
	Vishal Chawla	Accounting	Northeastern University

- 2. Search for a distinct research opportunity (for example – Machine Learning) available in a particular university along with the list of all the professors of that university.**

**Description:** This query returns a list of ongoing research in a university under a professor

**Actor:** Student

**Precondition:** The student must select research opportunities

**Steps:**

**Actor Action:** The student views the ongoing research from a university

**System Response:** The system displays all the ongoing research from a university under the professor

**Postcondition:** The system generates a list of ongoing research

#### **SQL Query:**

Create view v2 as

```
select distinct(r.Research_Name), u.University_Name,p.professors from
professor p
inner join university_details u on p.university_id = u.university_id
inner join research r on r.university_id = u.university_id
where r.Research_Name like '%Learning%';
```

The screenshot shows a SQL editor interface with multiple tabs at the top labeled SQL File 4\*, SQL File 5\*, SQL File 6, SQL File 7, SQL File 8\*, SQL File 9\*, SQL File 10\*, SQL File 11\*, and SQL File 12\*. Below the tabs is a toolbar with various icons for file operations. The main area contains the following SQL code:

```
1 • Create view v2 as
2   select distinct(r.Research_Name), u.University_Name,p.professors from
3     professor p
4     inner join university_details u on p.university_id = u.university_id
5     inner join research r on r.university_id = u.university_id
6     where r.Research_Name like '%Learning%';
7
8 • select * from v2;
```

Below the code, there is a "Result Grid" section with a table showing the results of the query. The table has three columns: Research\_Name, University\_Name, and professors. The data is as follows:

Research_Name	University_Name	professors
Research in Programming Languages and Machi...	Northeastern University	Mehdi Abedi
Research in Programming Languages and Machi...	Northeastern University	Emad Aboelela
Research in Programming Languages and Machi...	Northeastern University	Gregory D. Abowd
Research in Programming Languages and Machi...	Northeastern University	Kal Bugrara
Research in Programming Languages and Machi...	Northeastern University	Vishal Chawla
Research in Programming Languages and Machi...	Northeastern University	Nick Brown
Research in Programming Languages and Machi...	Northeastern University	Daniel Peters
Research Experience and Applied Learning (RE...	University of Colorado, Boulder	Titan Alon
Research Experience and Applied Learning (RE...	University of Colorado, Boulder	James Andreoni
Research Experience and Applied Learning (RE...	University of Colorado, Boulder	Kate Antonovics
Research Experience and Applied Learning (RE...	University of Colorado, Boulder	David Arnold

- 3. Display the recruiter ranking along with the average salary and the university from which it hires the maximum number of students**

**Description:** Display a list of recruiters along with the average salary that they offer and the maximum number of students they hire from a particular university

**Actor:** Student

**Precondition:** The student must select a list of top recruiters

**Steps:**

**Actor Action:** The student views the recruiters along with the average package

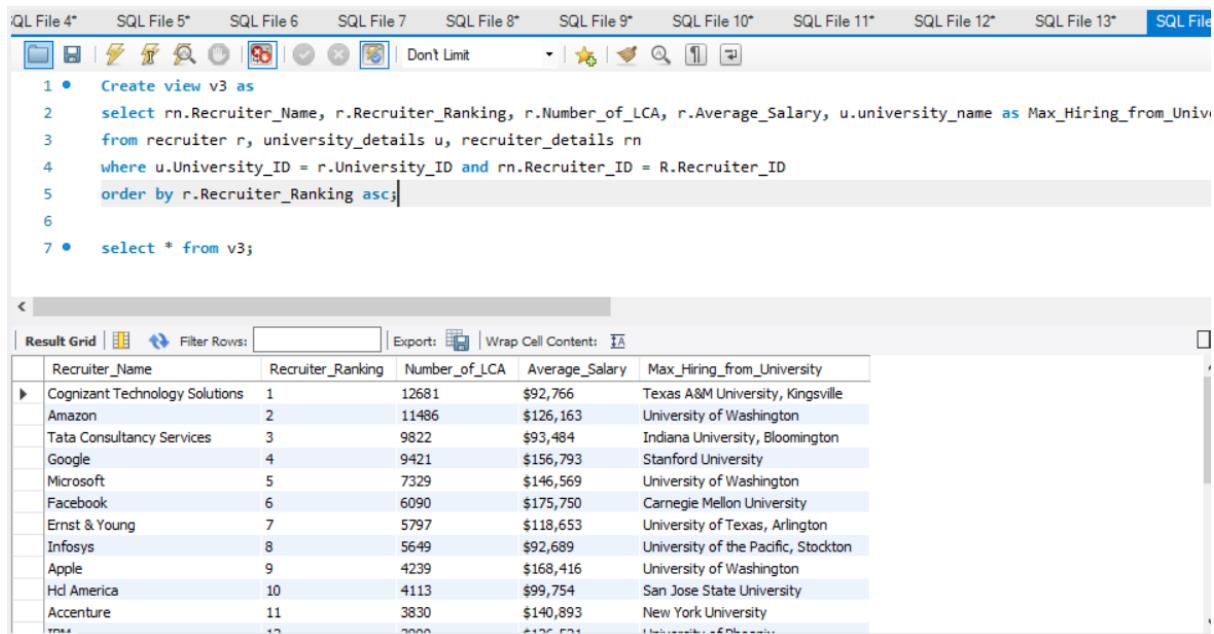
**System Response:** The system displays a list of all the recruiters and the university from which they hire the maximum number of students

**Postcondition:** The system generates a list of recruiters

### SQL Query:

Create view v3 as

```
select rn.Recruiter_Name, r.Recruiter_Ranking, r.Number_of_LCA,
r.Average_Salary, u.university_name as Max_Hiring_from_University
from recruiter r, university_details u, recruiter_details rn
where u.University_ID = r.University_ID and rn.Recruiter_ID = R.Recruiter_ID
order by r.Recruiter_Ranking asc;
```



The screenshot shows the SQL Server Management Studio interface. At the top, there is a toolbar with various icons. Below the toolbar, a menu bar has 'SQL File 4\*' through 'SQL File 13\*' and 'SQL File' selected. In the main area, there is a code editor window containing the following SQL script:

```
1 • Create view v3 as
2     select rn.Recruiter_Name, r.Recruiter_Ranking, r.Number_of_LCA,
3            r.Average_Salary, u.university_name as Max_Hiring_from_University
4            from recruiter r, university_details u, recruiter_details rn
5            where u.University_ID = r.University_ID and rn.Recruiter_ID = R.Recruiter_ID
6            order by r.Recruiter_Ranking asc;
7 • select * from v3;
```

Below the code editor is a result grid window titled 'Result Grid'. It contains a table with the following data:

Recruiter_Name	Recruiter_Ranking	Number_of_LCA	Average_Salary	Max_Hiring_from_University
Cognizant Technology Solutions	1	12681	\$92,766	Texas A&M University, Kingsville
Amazon	2	11486	\$126,163	University of Washington
Tata Consultancy Services	3	9822	\$93,484	Indiana University, Bloomington
Google	4	9421	\$156,793	Stanford University
Microsoft	5	7329	\$146,569	University of Washington
Facebook	6	6090	\$175,750	Carnegie Mellon University
Ernst & Young	7	5797	\$118,653	University of Texas, Arlington
Infosys	8	5649	\$92,689	University of the Pacific, Stockton
Apple	9	4239	\$168,416	University of Washington
Hd America	10	4113	\$99,754	San Jose State University
Accenture	11	3830	\$140,893	New York University
max	12	2000	\$100,000	University of California, Berkeley

#### 4. Display the list of all the professors that teach under a particular university.

**Description:** The student views the professors from a particular university

**Actor:** Student

**Precondition:** The student must select a university

**Steps:**

**Actor Action:** The student views the professor from a particular university

**System Response:** The system generates a list of professors from a university

**Postcondition:** The system generates all the professors

**SQL Query:**

Create view v4 as

```
select p.Professors, ud.university_name  
from professor p inner join university_details ud on p.University_ID =  
ud.University_ID  
where University_Name = "Northeastern University"
```

The screenshot shows a SQL query editor interface. At the top, there are tabs for 'SQL File 4\*' through 'SQL File 12\*'. Below the tabs is a toolbar with various icons. The main area contains a numbered list of SQL statements. Statements 1 through 6 are part of creating a view named v4, which selects professors and their university names from a professor table joined with a university\_details table where the university name is 'Northeastern University'. Statement 7 shows the execution of the view with a select statement. Below the code, there is a 'Result Grid' section with a table showing the results. The table has two columns: 'Professors' and 'university\_name'. The data consists of seven rows, each containing a professor's name and the 'Northeastern University'.

Professors	university_name
Mehdi Abedi	Northeastern University
Emad Aboelela	Northeastern University
Gregory D. Abowd	Northeastern University
Kai Bugrara	Northeastern University
Vishal Chawla	Northeastern University
Nick Brown	Northeastern University
Daniel Peters	Northeastern University

**5. Display the conditions on which a student is admitted to a university along with their graduate major, and average pay scale after graduation**

**Description:** Displays the entire academics of a student based on admit

**Actor:** Student

**Precondition:** The student must select a student ID

**Steps:**

**Actor Action:** The student selects a student ID and views the entire academic

**System Response:** The system generates a list of all the admitted students

**Postcondition:** A list is generated with all the amidst

**SQL Query:**

Create view v5 as

```

Select s.student_ID, s.acceptance, ss.GRE_Q,
ss.GRE_V, ss.GRE_AWA, ss.TOEFL, su.Undergrad_University,
su.Undergrad_Major, u.university_name, un.avg_pay_scale, r.university_id
from Student s, University_Details u, university un, recruiter r, student_score
ss, student_undergrad su
where s.university_id = u.university_id
and u.university_id = r.university_id
and un.University_ID = u.University_ID
and ss.student_ID = s.student_ID
and su.student_ID = s.student_ID
and s.acceptance = 'Admit';

```

The screenshot shows a SQL Server Management Studio window with multiple tabs at the top. The active tab is 'SQL File 14\*' which contains the following SQL code:

```

1 • Create view v5 as
2 Select s.student_ID, s.acceptance, ss.GRE_Q, ss.GRE_V, ss.GRE_AWA, ss.TOEFL, su.Undergrad_University, su.Undergrad_Major, u.university_name, un.avg_pay_scale, r.university_id
3 from Student s, University_Details u, university un, recruiter r, student_score ss, student_undergrad su
4 where s.university_id = u.university_id
5 and u.university_id = r.university_id
6 and un.University_ID = u.University_ID
7 and ss.student_ID = s.student_ID
8 and su.student_ID = s.student_ID
9 and s.acceptance = 'Admit';
10
11 • select * from v5;

```

Below the code, the 'Result Grid' shows the output of the query. The table has the following columns: student\_ID, acceptance, GRE\_Q, GRE\_V, GRE\_AWA, TOEFL, Undergrad\_University, Undergrad\_Major, university\_name, avg\_. The data is as follows:

student_ID	acceptance	GRE_Q	GRE_V	GRE_AWA	TOEFL	Undergrad_University	Undergrad_Major	university_name	avg_
1	Admit	157	142	3.5	97	University of Mumbai	Computer Engineering	Northeastern University	10000
2	Admit	None	None	None	None	fr.c.r.c.e.(bandra)	production engineering	Northeastern University	10000
3	Admit	770	460	3	105		CS	Northeastern University	10000
4	Admit	159	152	3	94	None	ECE	Northeastern University	10000
5	Admit	158	144	3	100	Apeejay College of Engineering	CS	Northeastern University	10000
6	Admit	162	155	4	None	RNSIT	Information Science	Northeastern University	10000
7	Admit	160	154	3.5	108	Sri Bhagawan Mahaveer Jain College of Enginee...		Northeastern University	10000

## 6. Display the list of a specific course offered by all the universities.

**Description:** This query returns a value of all the universities that provide a particular course

**Actor:** Student

**Precondition:** The student must select a course

**Steps:**

**Actor Action:** The student selects a particular course that he wishes to be enrolled for

**System Response:** The system displays a list of all the universities that provide that particular course

**Postcondition:** A list of all the universities that offer analytics courses is displayed

### SQL Query:

Create view v6 as

select c.course\_name, u.university\_name

```
from course c inner join university_details u on c.University_ID = u.University_ID  
where c.course_name like '%Analytics%';
```

The screenshot shows a SQL interface with a toolbar at the top and a code editor below it. The code editor contains the following SQL query:

```
1 • Create view v6 as  
2   select c.course_name, u.university_name  
3   from course c inner join university_details u on c.University_ID = u.University_ID  
4   where c.course_name like '%Analytics%'  
5  
6 • select * from v6;
```

Below the code editor is a result grid titled "Result Grid". It has two columns: "course\_name" and "university\_name". The data is as follows:

course_name	university_name
Analytics	Northeastern University
Applied Analytics	Northeastern University
graduate-certificate-in-urban-analytics-14299	Northeastern University
Masters of Data Analytics	Northeastern University
Big Data Analytics (MS) (SDSU)	Ohio State University
Big Data Analytics (MS) (SDSU GC)	Ohio State University
Management - Business Analytics MS	University of California, San Diego
Business Analytics	University of Maryland Baltimore

## 7. Display the data about previous admits where the year of joining >= 2015

**Description:** To view a list of all the previous admits to a university after 2015

**Actor:** The student

**Precondition:** The student must select a year

**Steps:**

**Actor Action:** The student selects the year and displays the admits

**System Response:** The system displays a list of all the previous admits

**Postcondition:** The list is displayed with all the previous admits

### SQL Query:

Create view v7 as

```
select s.student_id, ud.university_name as  
Target_University,s.Target_Major,s.Year_of_Joining,ss.GRE_Total,ss.TOEFL  
,ua.GRE as University_GRE_Cutoff,ua.TOEFL as  
University_TOEFL_cutoff,s.acceptance  
from student s, university u, university_details ud, student_score ss,  
university_admit_requirement ua  
where u.University_ID = s.University_ID and ud.University_ID =  
u.University_ID  
and ss.student_ID = s.student_ID and ua.University_ID = ud.University_ID  
and s.Year_of_Joining>2015;
```

The screenshot shows a MySQL Workbench interface. At the top, there are tabs for various SQL files. Below the tabs is a toolbar with icons for file operations, search, and preview. The main area contains a SQL query:

```

1 • Create view v7 as
2   select s.student_id, ud.university_name as Target_University,s.Target_Major,s.Year_of_Joining,ss.GRE_Total,ss.TOEFL,u
3   from student s, university u, university_details ud, student_score ss, university_admit_requirement ua
4   where u.University_ID = s.University_ID and ud.University_ID = u.University_ID
5   and ss.student_ID = s.student_ID and ua.University_ID = ud.University_ID
6   and s.Year_of_Joining>2015;
7
8 • select * from v7;

```

Below the query is a result grid titled "Result Grid". It has columns for student\_id, Target\_University, Target\_Major, Year\_of\_Joining, GRE\_Total, TOEFL, University\_GRE\_Cutoff, University\_TOEFL\_cutoff, and acceptance. The data is as follows:

student_id	Target_University	Target_Major	Year_of_Joining	GRE_Total	TOEFL	University_GRE_Cutoff	University_TOEFL_cutoff	acceptance
22	Northeastern University	Computer Science	2016	323	116	310	100	Admit
147	Northeastern University	MIS	2016	307	91	310	100	Admit
255	Northeastern University	MIS	2016	306	100	310	100	Admit
377	Northeastern University	Industrial Engineering	2016	321	111	310	100	Admit
405	Northeastern University	Computer Science	2016	321	101	310	100	Admit
446	Northeastern University	Computer Science	2016	315	98	310	100	Admit
585	Northeastern University	Engineering Management	2016	319	108	310	100	Admit
654	Northeastern University	Computer Science	2016	314	110	310	100	Admit

## 8. Display the list of courses choices based on specifications a student can select from post getting the admit.

**Description:** To view a list of all the possible choices student has

**Actor:** Student

**Precondition:** The student must select a course name for acceptance

**Steps:**

**Actor Action:** The student must select a course name

**System Response:** A list is displayed with all the possible admits a student has from a university

**Postcondition:** A list is displayed with all possible universities

### SQL Query:

Create view v8 as

```

select s.student_id, ud.university_name,s.Acceptance,c.Course_Name from
student s join course c on c.University_ID = s.University_ID join
university_details ud on ud.university_id = c.University_ID and s.Acceptance
= 'Admit' and c.Course_Name like '%Engineering%';

```

```

1 Create view v8 as
2 select s.student_id, ud.university_name,s.Acceptance,c.Course_Name from student s join course c on c.University_ID = s.Unive
3
4 • select * from v8;

```

student_id	university_name	Acceptance	Course_Name
1	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineerin...
2	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineerin...
3	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineerin...
4	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineerin...
5	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineerin...
6	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineerin...
7	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineerin...
10	Northeastern University	Admit	graduate-certificate-in-supply-chain-engineerin...

**9. Display the list of desired research opportunities available in a university based on the course selection of a student.**

**Description:** To view the list of all the research in a particular course

**Actor:** Student

**Precondition:** The student must select a research domain say Machine Learning

**Steps:**

**Actor Action:** The student must select a domain for research

**System Response:** The system displays a list of all the research in a particular course

**Postcondition:** A list is displayed with all the research going on in a course

**SQL Query:**

Create view v9 as

```

select r.research_name, c.course_name from research r inner join course c
where r.University_ID = c.University_ID and r.Research_Name like
'%Machine Learning%' and course_name in (select Course_Name from
course where course_name like '%Analytics%');

```

The screenshot shows a database interface with a query editor at the top and a result grid below. The query editor contains the following SQL code:

```

1 Create view v9 as
2 select r.research_name, c.course_name from research r inner join course c where r.University_ID = c.University_ID and r.Research_Name like
3
4 • select * from v9;

```

The result grid displays the following data:

research_name	course_name
Research in Programming Languages and Machi...	Analytics
Research in Programming Languages and Machi...	Applied Analytics
Research in Programming Languages and Machi...	graduate-certificate-in-urban-analytics-14299
Research in Programming Languages and Machi...	Masters of Data Analytics

On the right side of the interface, there is a vertical toolbar with icons for Result Grid, Form Editor, Field Types, and Query Stats. The 'Result Grid' icon is highlighted.

#### **10. Display the list of research opportunities available in an university based on the location of the university.**

**Description:** To display a list of research going in a particular location in which the university exists

**Actor:** Student

**Precondition:** The student must select a location

**Steps:**

**Actor Action:** The student selects a location to look for research

**System Response:** The system displays a list of universities in that location where research is going on

**Postcondition:** A list is displayed with the research going in a particular location in which the university exists

**SQL Query:**

Create view v10 as

```

select r.research_name, ud.university_name
from research r, university_details ud, university u
where r.university_id = ud.university_id
and u.Location = "Boston";

```

SQL File 37\* SQL File 38\* SQL File 40\* SQL File 41 SQL File 42\* SQL File 43\* SQL File 44\* SQL File 45\* SQL File 46\*

Limit to 1000 rows

```
1 • Create view v10 as
2   select r.research_name, ud.university_name
3   from research r, university_details ud, university u
4   where r.university_id = ud.university_id
5   and u.Location = "Boston";
6
7 • select * from v10;
```

Result Grid | Filter Rows:  Export: Wrap Cell Content:

research_name	university_name
New College Undergraduate Inquiry and Resea...	Arizona State University Tempe
Fulton Undergraduate Research Initiative (FURI)	Arizona State University Tempe
Climate Dynamics	Harvard University
Modeling of Atmospheric Chemistry	Harvard University
Aging and Climate Change in China	Harvard University
Office of Research Advancement	University of Texas, Arlington
Office for the Protection of Research Subjects	University of Texas, Arlington
Institutional Animal Care and Use Committee	University of Texas, Arlington
Department of Contracts and Grants	University of Texas, Arlington
Department of Animal Resources	University of Texas, Arlington
Targeted detection of allergies in Quinoa Flour ...	University of Maryland College...
Investigating Regulation of Pro-Inflammatory G...	University of Maryland College...
Virtual Reality Technology In Seconf Language ...	University of Maryland College...
Research and Fact-Checking for Book Chapter	Harvard University

## SAMPLE OUTPUT FROM ALL THE TABLES:

### 1. Course:

```
1  select * from course;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

ID	University_ID	Course_Name
6	1	Adult-Gerontology Acute Care Nursing (CAGS)
7	1	Adult-Gerontology Primary Care Nursing (CAGS)
8	1	Advanced and Intelligent Manufacturing
9	1	Agile Project Management
10	1	Analytics
11	1	Applied Analytics
12	1	master-of-sports-leadership-boston-267
13	1	master-of-sports-leadership-online-268
14	1	master-of-sports-leadership-charlotte-14376
15	1	graduate-certificate-in-supply-chain-engineerin...
16	1	graduate-certificate-in-supply-chain-manageme...
17	1	graduate-certificate-in-supply-chain-manageme...
18	1	graduate-certificate-in-sustainability-and-busin...
19	1	graduate-certificate-in-sustainability-and-climat...

### 2. Professor:

```
1  select * from professor;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

ID	University_ID	Professors
1	58	Pieter Abbeel
2	58	Rediet Abebe
3	58	Ahmed Alaa
4	58	Elad Alon
5	58	Venkat Anantharam
6	58	Gopala Krishna Anumanchipalli
7	58	Mekhail Anwar
8	58	Murat Arcak Ana Claudia Arias
9	58	David Attwood
10	58	Babak Ayazifar
11	36	Titan Alon
12	36	James Andreoni
13	36	Kate Antonovics
14	36	David Arnold

### 3. Recruiter

```
SQL File 37* SQL File 38* SQL File 40* SQL File 41 SQL File 42* SQL
1   select * from recruiter;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

	Recruiter_ID	Recruiter_Ranking	Number_of_LCA	Average_Salary	University_ID
▶	1	1	12681	\$92,766	32
	10	10	4113	\$99,754	25
	11	11	3830	\$140,893	20
	12	12	3800	\$126,521	59
	13	13	3756	\$119,341	41
	14	14	3367	\$127,627	26
	15	15	3325	\$102,410	1
	16	16	3218	\$133,510	35
	17	17	2987	\$133,103	60
	18	18	2858	\$80,871	50
	19	19	2755	\$80,529	62
	2	2	11486	\$126,163	65
	20	20	2231	\$108,016	14
	21	21	2167	\$152,754	26
	22	22	2164	\$141,276	41
	23	23	2107	\$94,298	23
	24	24	2142	\$115,750	42

### 4. Recruiter\_Details

```
SQL File 37* SQL File 38* SQL File 40* SQL File 41 SQL File 42* SQL
1   select * from recruiter_details;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

	Recruiter_ID	Recruiter_Name
▶	1	Cognizant Technology Solutions
	10	Hd America
	11	Accenture
	12	IBM
	13	Deloitte Consulting
	14	Intel
	15	Capgemini
	16	Wal-Mart Associates
	17	Jpmorgan Chase
	18	Wipro
	19	Tekorg
	2	Amazon
	20	Compunnel Software Group
	21	Qualcomm Technologies
	22	Salesforce.Com
	23	Tech Mahindra
	24	Others

## 5. Research

The screenshot shows a MySQL Workbench interface with the following details:

- Toolbar icons: folder, table, lightning bolt, magnifying glass, search, refresh, etc.
- Text input field: "Limit to 1000 rows".
- Query editor: "1 select \* from research;"
- Result Grid: A table with columns "Research\_ID", "Research\_Name", and "University\_ID".
- Data rows (approximate values):

Research_ID	Research_Name	University_ID
1	Environmental Economics and Policy	15
10	School of Life Sciences Undergraduate Researc...	6
11	Sustainability Undergraduate Research Experie...	6
12	Watts College of Public Service and Community...	6
13	Ira A. Fulton Schools of Engineering Summer R...	6
14	Web & Visualization Developer (Media Cloud Pro...	1
15	Research in Programming Languages and Machi...	1
16	Multiple research options on experimental multi...	1
17	Imaging microbiome interactions: Microfluidics	1
18	Microscopy	1
19	Molecular Biology	1
2	Investigating the Mold Resiliency of Buildings in...	15
20	and Automation	1
21	Data Analysis and Visualization of Hurricane Ris...	1
22	Undergraduate Research Assistant – Metal 3D ...	1
23	Translational research opportunities in leukemo...	1

## 6. Scholarship

The screenshot shows a MySQL Workbench interface with the following details:

- Toolbar icons: folder, table, lightning bolt, magnifying glass, search, refresh, etc.
- Text input field: "Limit to 1000 rows".
- Query editor: "1 select \* from scholarship;"
- Result Grid: A table with columns "Scholarship\_ID", "Scholarship\_Name", "Scholarship\_URL", and "University\_ID".
- Data rows (approximate values):

Scholarship_ID	Scholarship_Name	Scholarship_URL	University_ID
1	John B. Ervin Scholars Program	https://admissions.wustl.edu/cost-aid/scholarsh...	65
2	Stamps Scholarship	http://stamps.ezitsolutions.net/portfolios/unive...	34
3	Mork Family Scholarship	https://ahf.usc.edu/meritscholars/merit-scholar...	47
4	Stamps Scholarship	http://www.stampsfoundation.org/2012/11/20/...	42
5	Stamps President's Scholars Program	http://stampsps.gatech.edu/	14
6	Trustee Scholarship	http://www.bu.edu/admissions/apply/tuition-ai...	3
7	Torch Scholars Program	http://www.northeastern.edu/torch/	1
8	Coronat Scholars	https://financialaid.syr.edu/scholarships/su/?re...	4
9	Banneker/Key Scholars Program	http://www.bannekerkey.umd.edu/stampsbk.html	41
11	Eminence Fellows Scholarship	http://undergrad.osu.edu/cost-and-aid/merit-b...	64
12	Stamps Scholarship	https://admissions.illinois.edu/Invest/scholarshi...	40
13	National Scholars Program	https://www.clemson.edu/academics/programs...	9
14	Civic & Service Engagement Scholar...	http://oglethorpe.edu/schwk17/about-our-scho...	60
15	Forty Acres Scholarship	https://www.texasexes.org/scholarships/forty-...	49
16	Eugene McDermott Scholars Program	http://www.utdallas.edu/mcdermott/	50
*	NULL	NULL	NULL

## 7. Student

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the SQL command:

```
1 select * from student;
```

The result grid displays 20 rows of data from the student table, with columns: student\_ID, University\_ID, Acceptance, Program, Target\_Major, Term, and Year\_of\_Joining. The data includes various student records with different majors like Computer Science, Industrial Engineering, MIS, and Electronics and Communication, across different years of joining (2009 to 2014) and terms (Fall).

	student_ID	University_ID	Acceptance	Program	Target_Major	Term	Year_of_Joining
▶	1	1	Admit	MS	Computer Science	Fall	2012
2	1	Admit	MS	Industrial Engineering	Fall	2011	
3	1	Admit	MS	Industrial Engineering	Fall	2011	
4	1	Admit	MS	Computer Science	Fall	2013	
5	1	Admit	MS	Management Information System	Fall	2014	
6	1	Admit	MS	Computer Science	Fall	2013	
7	1	Admit	MS	Computer Science	Fall	2013	
10	1	Admit	MS	MIS	Fall	2011	
11	1	Admit	MS	Information Security	Fall	2009	
13	1	Admit	MS	Electronics and Communication	Fall	2012	
15	1	Admit	MS	Information Security	Fall	2013	
16	1	Admit	MS	Computer Science	Fall	2012	
17	1	Admit	MS	Computer Science	Fall	2015	
18	1	Admit	MS	Computer Science	Fall	2011	
19	1	Admit	MS	Computer Science	Fall	2013	
20	1	Admit	MS	Computer Science	Fall	2014	

## 8. Student\_Score

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the SQL command:

```
1 select * from student_score;
```

The result grid displays 20 rows of data from the student\_score table, with columns: student\_ID, GRE\_Q, GRE\_V, GRE\_Total, GRE\_AWA, TOEFL, and Work\_Ex. The data includes GRE scores ranging from 148 to 780 and TOEFL scores ranging from 94 to 113.

	student_ID	GRE_Q	GRE_V	GRE_Total	GRE_AWA	TOEFL	Work_Ex
▶	1	157	142	299	3.5	97	0
2	None	None	0	None	None	0	
3	770	460	1230	3	105	0	
4	159	152	311	3	94	0	
5	158	144	302	3	100	0	
6	162	155	317	4	None	0	
7	160	154	314	3.5	108	0	
10	760	680	1440	4	111	0	
11	740	540	1280	5	113	0	
13	780	590	1370	3.5	114	0	
15	158	148	306	3	94	0	
16	800	610	1410	3.5	110	0	
17	164	155	319	3	111	36	
18	760	630	1390	3.5	103	0	
19	164	157	321	3	108	0	
20	161	163	324	3.5	109	0	

## 9. Student\_Undergrad

```
1   select * from student_undergrad;
```

	student_ID	Undergrad_University	Undergrad_Major	CGPA	CGPA_Scale
29	MU	IT	5.7	10	
30	Jaypee Institute of Information Technology	Electronics and Teleco...	8.8	10	
31	Jai Narain Vyas University Jodhpur	Electrical Engineering	6.7	10	
32	University of Mumbai	Computer	6.4	10	
33	Anna University	Information Technology	7.12	10	
34	JNTU	ece	7.1	10	

## 10. University

```
1   select * from university;
```

	University_ID	avg_pay_scale	Duration	Term	Year_of_Joining	Major	Avg_Fees	Location	Cost_of_Living
1	100000	2	Fall	2022	MSIS	54000	Boston	9600	
10	36100	1.8	Fall	2022	MSIS	21702	Cleveland	24693	
11	113700	2	Fall	2022	MSCS	29500	Ithaca	19726	
12	24249	2	Fall	2022	MSCS	16860	Canada	16776	
13	56200	2	Fall	2022	MSCS	22110	Virginia	59782	
14	94000	2	Fall	2022	MSCS	18225	Atlanta	55672	
15	128900	1.8	Fall	2022	MSCS	89952	Boston	83463	
16	68647	2	Fall	2022	MSCS	26336	Chicago	12600	
17	136000	1.8	Fall	2023	MSCS	24375	Indiana	38680	
18	60156	2	Fall	2022	MSCS	23790	Iowa	35004	
19	45700	2	Fall	2023	MSCS	14385	Kansas	15046	
2	90000	2	Fall	2022	MSCS	60000	Texas	7800	
20	83000	2	Fall	2022	MSCS	35982	New York	40000	
21	71128	2	Fall	2022	MCE	29145	New York	83250	
22	30000	2	Fall	2022	MSDA	28155	Pennsylv...	30000	
23	56197	1	Fall	2023	MSBA	54176	New York	72128	
24	"	"	Fall	2022	MCE	48272	"	22500	

## 11. University Admit Requirement

```
1 select * from university_admit_requirement;
```

Result Grid | Filter Rows: | Edit: | Export/Imp

	University_ID	GRE	toefl	ielts
▶	1	310	100	7.5
	10	295	78	6
	11	295	100	7
	12	300	92	7
	13	Waived	88	7
	14	Waived	79	6.5
	15	Waived	80	6.5
	16	309	90	6.5
	17	Waived	100	7
	18	316	79	6.5
	19	310	79	6.5
	2	300	100	7.5
	20	321	100	7
	21	316	90	7
	22	314	80	6.5
	23	307	88	6.5
...	...	...	...	...

## 12. University Details

```
1 select * from university_details;
```

Result Grid | Filter Rows: | Edit: | Export/Imp

	University_ID	University_Name
▶	1	Northeastern University
	10	Cleveland State University
	11	Cornell University
	12	Dalhousie University
	13	George Mason University
	14	Georgia Institute of Technology
	15	Harvard University
	16	Illinois Institute of Technology, Chicago
	17	Indiana University, Bloomington
	18	Iowa State University
	19	Kansas State University
	2	University of Texas, San Antonio
	20	New York University
	21	NYU Tandon School of Engineering
	22	Pennsylvania State University
	23	Rochester Institute of Technology
...	...	...