

### **Business Problem Statement:**

Voyage a Student Consultancy app. It provides consultancy services for students to apply for Universities across many regions like US(United States), UK, Europe, Australia, and Canada. A student needs to go to the University's website and apply for every single university, to simplify this procedure, we partner with universities across the globe to provide a one-stop solution where a student can submit and track all their university applications in an easier and more efficient way. The Universities appoint their representative on our platform to give their decision on applications based on student credentials. We also organize career fairs, providing a platform for universities to promote and highlight their USPs.

### **Problem Statement:**

In the app every user must create an account and indicate their personal details and every user should also be uniquely identifiable.

Each app user will be given 3 options: 'University Admin,' 'Student,' and 'Employee.'

A 'University admin' must provide both SSN and Name, he represents the University and will have the authorization to decide on students' application statuses. Each University has a unique ID, Name, and address.

Universities offer admission to their programs. Each program has a unique ID, Name, and student requirements. Different universities might offer similar programs.

An 'Employee' of the consultancy must provide SSN, Name, and address. The consultancy has a total of four departments namely 'Finance', "Sales", 'Marketing', and 'IT'. The universities can collaborate with the consultancy's marketing department to organize Fairs to promote their Programs. The "Sales" department is responsible for bringing new universities on board.

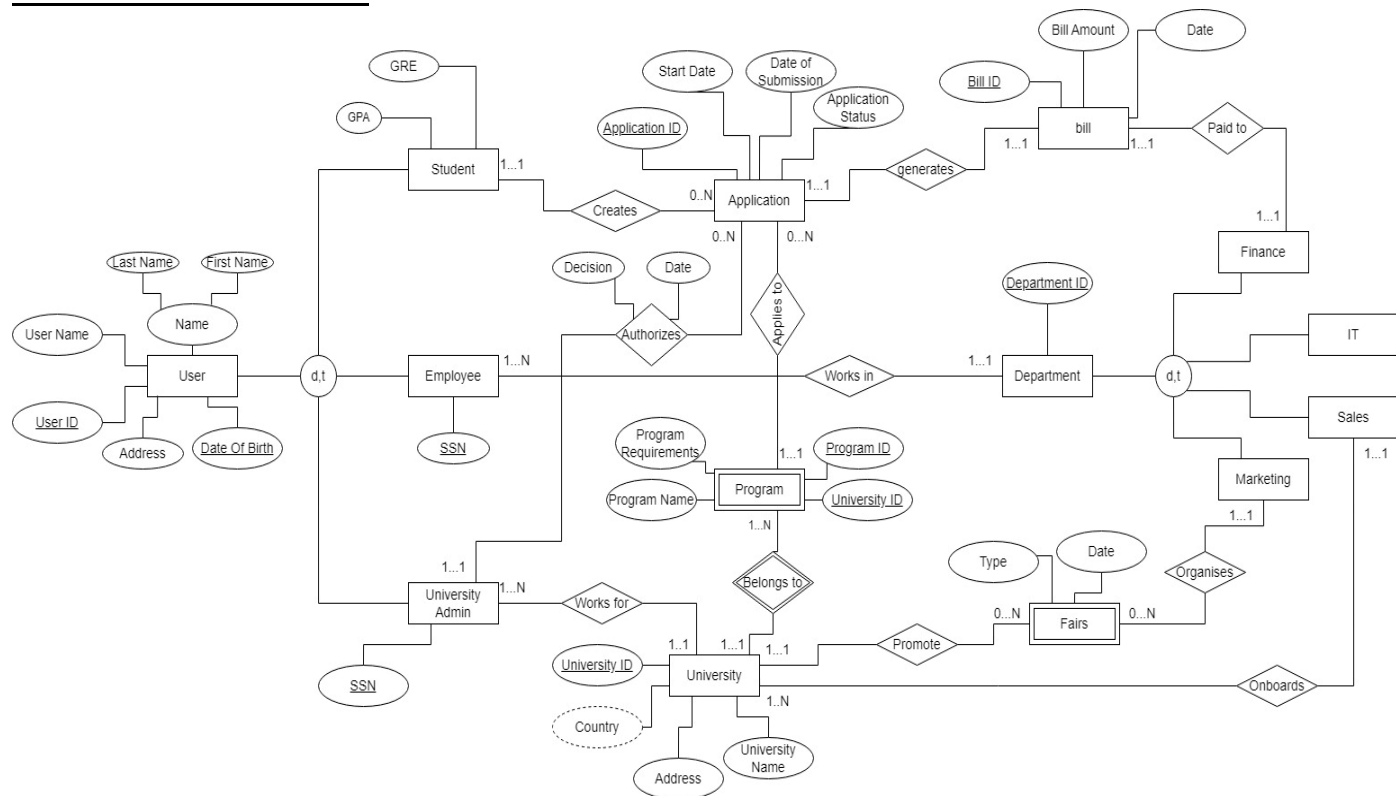
A 'Student' needs to provide his Name, Date of Birth, Past Education details, and Address.

Each 'Application' is given a unique Application ID, start date, Date of submitted Application, and Application status. Each Application generates a bill with a unique 'bill ID' paid by the student to the finance department.

## Constraints and assumptions in the model:

- Each university provides admission for at least one of its programs.
- Different universities might offer similar programs.
- Each program of a university has its own 'University admin' hence a university can have more than one 'University admin'.
- An employee can only work in one of the departments.
- A student can create multiple applications and apply for multiple programs.
- A student can submit only one application for each program.
- A single program can receive multiple applications.
- Each application generates only one bill.

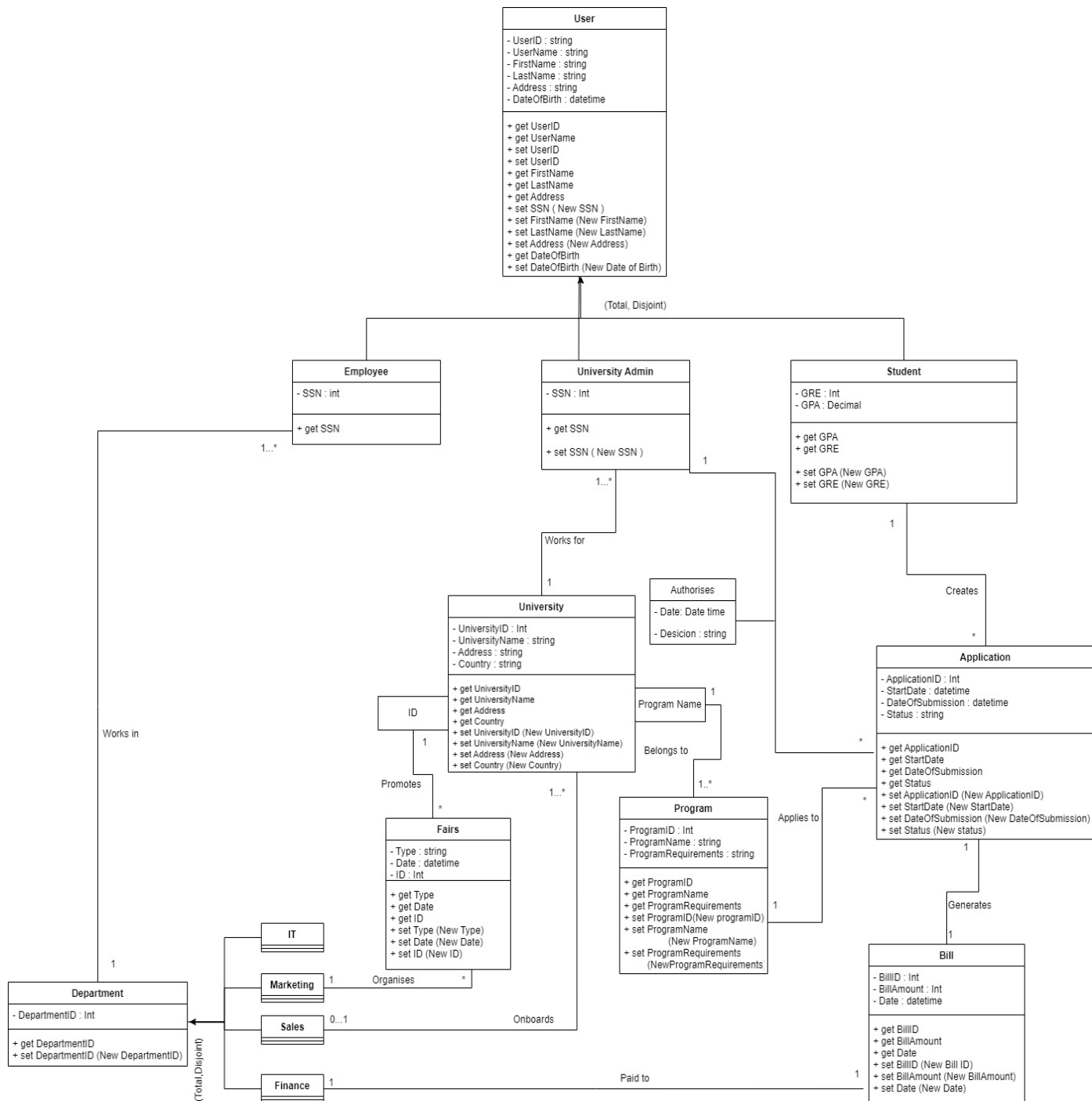
## EER MODEL DIAGRAM:



**Referential data:** User, student, employee, university admin, university, department

**Transactional data:** Bill, application

## UML CLASS DIAGRAM:



## **MAPPING TO A RELATIONAL MODEL:**

### **User (UserID, UserName, Address, First\_Name, Last\_Name, Date\_Of\_Birth )**

Primary Key: UserID

### **User\_Employee (UserID, SSN, DepartmentID)**

Primary Key: UserID; Foreign Key: UserID (NOT NULL), DepartmentID (NOT NULL)

Since UserID is both primary key and foreign key, 'NOT NULL' condition is given.

The relationship between 'User\_Employee' and 'Department' is '1:N', hence 'NOT NULL' condition is given for 'DepartmentID'.

### **User\_Student (UserID, GPA, GRE)**

Primary Key: UserID; Foreign Key: UserID (NOT NULL)

Since UserID is both primary key and foreign key, 'NOT NULL' condition is given.

### **User\_UniversityAdmin (UserID, SSN, UniversityID)**

Primary Key: UserID; Foreign Key: UserID (NOT NULL), UniversityID (NOT NULL)

Since UserID is both primary key and foreign key, 'NOT NULL' condition is given.

The relationship between 'User\_UniversityAdmin' and 'University' is '1:N', hence 'NOT NULL' condition is given for 'UniversityID'.

### **University (UniversityID, UniveristyName, Address, Country, Sales\_DepartmentID)**

Primary Key: UniversityID; Foreign Key: Sales\_departmentID (CAN BE NULL)

The relationship between 'University' and 'Sales\_Department' is '0 : N', hence 'CAN BE NULL' condition is given for 'Sales\_DepartmentID'.

### **Program (ProgramID, ProgramName, UniveristyID)**

Primary Key: (ProgramID, UniversityID); Foreign Key: UniversityID (NOT NULL)

Since 'Program' is a weak entity, 'UniversityID' is part of primary key and also a foreign key, 'NOT NULL' condition is given.

### **Application (ApplicationID, StartDate, DateOfSubmission, ApplicationStatus, Authorize\_Decision, Authorize\_Date, Student\_UserID, ProgramID, UniveristyID, UniversityAdmin\_UserID)**

Primary Key: ApplicationID; Foreign Key: Student\_UserID (NOT NULL), ProgramID (NOT NULL), UniversityID (NOT NULL), UniversityAdmin\_UserID (NOT NULL)

The relationship between 'User\_student' and 'application' is '1:N', hence 'NOT NULL' condition is given for 'Student\_UserID'.

The relationship between 'program' and 'application' is '1:N', hence 'NOT NULL' condition is given for 'ProgramID', 'UniversityID'.

The relationship between 'UniversityAdmin' and 'application' is '1:N', hence 'NOT NULL' condition is given for 'UniversityAdmin\_UserID'.

### **Bill (Bill\_ID, BillAmount, Date, *ApplicationID*, *Finance\_DepartmentID*)**

Primary Key: Bill\_ID; Foreign Key: *ApplicationID* (NOT NULL), *Finance\_DepartmentID* (NOT NULL)

The relationship between 'Bill' and 'application' is '1:1', hence 'NOT NULL' condition is given for 'ApplicationID'.

The relationship between 'Bill' and 'Finance\_Department' is '1:1', hence 'NOT NULL' condition is given for 'Finance\_DepartmentID'.

### **Fairs (FairID, Type, Date, UniversityID, *Marketing\_DepartmentID*)**

Primary Key: (FairID, UniversityID) Foreign Key: UniversityID (NOT NULL), Marketing\_DepartmentID (NOT NULL)

Since 'Fairs' is a weak entity, 'UniversityID' is part of primary key and also a foreign key, 'NOT NULL' condition is given.

The relationship between 'Fairs' and 'Marketing\_Department' is '1:N', hence 'NOT NULL' condition is given for 'Marketing\_DepartmentID'.

### **Department (Department\_ID, Department\_name)**

Primary Key: Department\_ID

## ANALYTICS USING MYSQL:

### 1) Analytic purpose:

**calculate and compare how much revenue the firm generated each year.**

Query to calculate total revenue per year in descending order.

```
SELECT YEAR(BILL_DATE) AS YEAR, SUM(BILL_AMOUNT) AS TOTAL_REVENUE
FROM BILL
GROUP BY YEAR(BILL_DATE)
ORDER BY TOTAL_REVENUE DESC;
```

	year	total_revenue
▶	2021	58944
	2023	50832
	2022	44979

### 2) Analytic purpose:

**which is the most popular program? And help the students to find out how much competition is there for their desired program.**

the query is written to calculate the total number of applications all programs have in descending order, all universities combined.

```
WITH TEMP AS (SELECT PROGRAM_ID, COUNT(APPLICATION_ID) AS TOTAL_APPLICATIONS
FROM APPLICATION
GROUP BY PROGRAM_ID)
SELECT DISTINCT T.PROGRAM_ID, P.PROGRAM_NAME, T.TOTAL_APPLICATIONS
FROM TEMP T, PROGRAM P
WHERE T.PROGRAM_ID = P.PROGRAM_ID
ORDER BY T.TOTAL_APPLICATIONS DESC;
```

	program_id	program_name	total_applications
▶	101	CS	635
	104	MBA	607
	102	IS	578
	103	IE	569

**3) Analytic purpose:**

**finding out which country is the most popular for education to help students decide.**

The query is written to find the total number of applications by country in descending order.

```
WITH TEMP AS (SELECT UNIVERSITY_ID, COUNT(APPLICATION_ID) AS TOTAL_APPLICATIONS
               FROM APPLICATION
               GROUP BY UNIVERSITY_ID)
SELECT DISTINCT U.COUNTRY, SUM(T.TOTAL_APPLICATIONS) AS TOTAL
FROM TEMP T, UNIVERSITY U
WHERE T.UNIVERSITY_ID = U.UNIVERSITY_ID
GROUP BY U.COUNTRY
ORDER BY TOTAL DESC;
```

	country	total
►	United States	1177
	Canada	983
	United Kingdom	229

**4) Analytic purpose:**

**finding out the most popular university.**

The query is to calculate the total number of applications to all universities in descending order.

```
WITH TEMP AS (SELECT UNIVERSITY_ID, COUNT(APPLICATION_ID) AS TOTAL_APPLICATIONS
               FROM APPLICATION
               GROUP BY UNIVERSITY_ID)
SELECT DISTINCT T.UNIVERSITY_ID, U.UNIVERSITY_NAME, T.TOTAL_APPLICATIONS
FROM TEMP T, UNIVERSITY U
WHERE T.UNIVERSITY_ID = U.UNIVERSITY_ID
ORDER BY T.TOTAL_APPLICATIONS DESC;
```

	university_id	university_name	total_applications
►	1	University of Windsor	275
	8	Fresno City College	261
	3	Thompson Rivers University	253
	9	University of Wisconsin - Platteville	236
	2	Assumption University	233
	7	University of Connecticut Health Center	230
	10	University of Portsmouth	229
	5	University of Miami	228
	4	Carleton University	222
	6	Thunderbird School of Global Management	222

5) Analytic purpose:

which university is the most difficult to get accepted into?

Query to calculate the acceptance percentage for each university in descending order.

```
WITH TEMP_ADMITTED AS (SELECT UNIVERSITY_ID,
                           COUNT(AUTHORIZE_DECISION) AS TOTAL_ADMITTED
                        FROM APPLICATION
                        WHERE AUTHORIZE_DECISION = "admitted"
                        GROUP BY UNIVERSITY_ID),
TEMP_REJECTED AS (SELECT UNIVERSITY_ID,
                           COUNT(AUTHORIZE_DECISION) AS TOTAL_REJECTED
                    FROM APPLICATION
                    WHERE AUTHORIZE_DECISION = "rejected"
                    GROUP BY UNIVERSITY_ID)
SELECT U.UNIVERSITY_ID, U.UNIVERSITY_NAME,
       (TA.TOTAL_ADMITTED/(TR.TOTAL_REJECTED + TA.TOTAL_ADMITTED))*100 AS PERCENTAGE
FROM UNIVERSITY U, TEMP_ADMITTED TA, TEMP_REJECTED TR
WHERE U.UNIVERSITY_ID = TA.UNIVERSITY_ID
      AND U.UNIVERSITY_ID = TR.UNIVERSITY_ID
ORDER BY PERCENTAGE;
```

	university_id	university_name	percentage
▶	6	Thunderbird School of Global Management	46.8468
	5	University of Miami	47.8070
	3	Thompson Rivers University	49.0119
	8	Fresno City College	49.0421
	9	University of Wisconsin - Platteville	50.0000
	4	Carleton University	50.8696
	7	University of Connecticut Health Center	50.9091
	1	University of Windsor	51.5284
	10	University of Portsmouth	57.5107
	2	Assumption University	

6) Analytic purpose:

To how many universities have each student applied to on average? This will help the students to figure out how many universities they should apply to.

Query to calculate the average number of applications submitted by each student each year.

```
WITH TEMP AS (SELECT STUDENT_USER_ID,
                       COUNT(APPLICATION_ID) AS TOTAL,
                       YEAR(START_DATE) AS YEAR
               FROM APPLICATION
               GROUP BY STUDENT_USER_ID, YEAR(START_DATE))
SELECT YEAR, AVG(TOTAL) AS AVERAGE
FROM TEMP
GROUP BY YEAR;
```

	year	average
▶	2023	2.8700
	2022	2.7871
	2021	3.8627



7) Analytic purpose:

which programs are the most difficult to get an acceptance? Help the students to figure out their chances of getting admission.

acceptance percentage for each program

```
WITH TEMP_ADMITTED AS (SELECT PROGRAM_ID,
                             COUNT(AUTHORIZE_DECISION) AS TOTAL_ADMITTED
                        FROM APPLICATION
                        WHERE AUTHORIZE_DECISION = "admitted"
                        GROUP BY PROGRAM_ID),
TEMP_REJECTED AS (SELECT PROGRAM_ID,
                             COUNT(AUTHORIZE_DECISION) AS TOTAL_REJECTED
                   FROM APPLICATION
                   WHERE AUTHORIZE_DECISION = "rejected"
                   GROUP BY PROGRAM_ID)
SELECT DISTINCT P.PROGRAM_ID, P.PROGRAM_NAME,
               (TA.TOTAL_ADMITTED/(TR.TOTAL_REJECTED + TA.TOTAL_ADMITTED))*100 AS PERCENTAGE
FROM PROGRAM P, TEMP_ADMITTED TA, TEMP_REJECTED TR
WHERE P.PROGRAM_ID = TA.PROGRAM_ID
      AND P.PROGRAM_ID = TR.PROGRAM_ID
ORDER BY PERCENTAGE;
```

	program_id	program_name	percentage
▶	102	IS	47.9239
	103	IE	50.9666
	104	MBA	51.2356
	101	CS	51.3386

8) Analytic purpose:

Find out which department has the highest number of employees.

the query to calculate the total number of employees in each department in descending order.

```
SELECT D.DEPARTMENT_NAME, UE.DEPARTMENT_ID, COUNT(UE.USER_ID)
FROM USER_EMPLOYEE UE, DEPARTMENT D
WHERE UE.DEPARTMENT_ID = D.DEPARTMENT_ID
GROUP BY UE.DEPARTMENT_ID;
```

	department_name	department_id	count(ue.user_id)
▶	sales	4	4
	finance	1	4
	IT	3	1
	marketing	2	1

9) Analytic purpose:

which university organized the most fairs to promote themselves?

the total number of fairs organized by each university in descending order.

```
WITH TEMP AS (SELECT UNIVERSITY_ID, COUNT(*) AS TOTAL
                FROM FAIRS
                GROUP BY UNIVERSITY_ID)
SELECT T.UNIVERSITY_ID, U.UNIVERSITY_NAME, T.TOTAL
FROM TEMP T, UNIVERSITY U
WHERE T.UNIVERSITY_ID = U.UNIVERSITY_ID
ORDER BY T.TOTAL DESC;
```

	university_id	university_name	total
▶	2	Assumption University	7
	8	Fresno City College	4
	1	University of Windsor	3
	3	Thompson Rivers University	3
	4	Carleton University	3
	6	Thunderbird School of Global Management	3
	5	University of Miami	2
	7	University of Connecticut Health Center	2
	10	University of Portsmouth	2
	9	University of Wisconsin - Platteville	1

10) Analytic purpose:

helping the student find out the most popular programs.

The query is written to find the top 3 programs and their universities with the highest number of applications.

```
WITH TEMP AS (SELECT U.UNIVERSITY_NAME, P.PROGRAM_NAME,
                    COUNT(A.APPLICATION_ID) AS TOTAL_APPLICATIONS
                FROM APPLICATION A, PROGRAM P, UNIVERSITY U
                WHERE A.UNIVERSITY_ID = U.UNIVERSITY_ID
                      AND A.UNIVERSITY_ID = P.UNIVERSITY_ID
                      AND P.PROGRAM_ID = A.PROGRAM_ID
                GROUP BY A.UNIVERSITY_ID, A.PROGRAM_ID)
SELECT T1.UNIVERSITY_NAME, T1.PROGRAM_NAME, T1.TOTAL_APPLICATIONS
FROM TEMP T1
WHERE 3 > (SELECT COUNT(*)
            FROM TEMP T2
            WHERE T1.TOTAL_APPLICATIONS < T2.TOTAL_APPLICATIONS)
ORDER BY T1.TOTAL_APPLICATIONS DESC;
```

	university_name	program_name	total_applications
▶	University of Windsor	CS	77
	Thompson Rivers University	MBA	77
	University of Windsor	IE	73

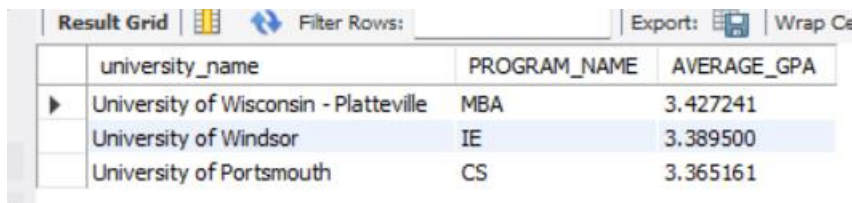
**11) Analytic purpose:**

**How much GPA does a student need to get an admit to a program?**

The query is written to the average GPA of admitted students.

```
CREATE VIEW AVG_GPA AS
SELECT U.UNIVERSITY_NAME, P.PROGRAM_NAME, AVG(US.GPA) AS AVERAGE_GPA
FROM APPLICATION A, UNIVERSITY U, USER_STUDENT US, PROGRAM P
WHERE A.UNIVERSITY_ID = U.UNIVERSITY_ID AND A.STUDENT_USER_ID = US.USER_ID
      AND A.PROGRAM_ID = P.PROGRAM_ID AND A.UNIVERSITY_ID = P.UNIVERSITY_ID
      AND A.AUTHORIZE_DECISION = "ADMITTED"
GROUP BY A.UNIVERSITY_ID, A.PROGRAM_ID;
```

```
SELECT AG.UNIVERSITY_NAME, AG.PROGRAM_NAME, AG.AVERAGE_GPA
FROM AVG_GPA AG
WHERE 3 > (SELECT COUNT(*)
           FROM AVG_GPA AG2
           WHERE AG.AVERAGE_GPA < AG2.AVERAGE_GPA)
ORDER BY AG.AVERAGE_GPA DESC;
```



	university_name	PROGRAM_NAME	AVERAGE_GPA
▶	University of Wisconsin - Platteville	MBA	3.427241
	University of Windsor	IE	3.389500
	University of Portsmouth	CS	3.365161

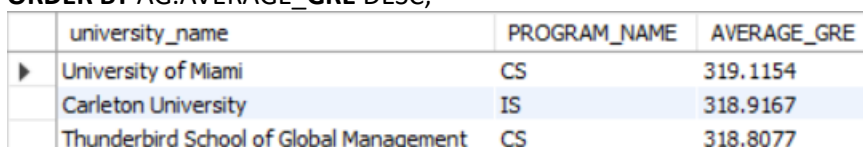
**12) Analytic purpose:**

**How much GRE score does a student need to get an admit to a program?**

The query is written to create a view to get the average GRE score of admitted students for each program. Using the view retrieve the top 3 programs with the highest average GRE.

```
CREATE VIEW AVG_GRE AS
SELECT U.UNIVERSITY_NAME, P.PROGRAM_NAME, AVG(US.GRE) AS AVERAGE_GRE
FROM APPLICATION A, UNIVERSITY U, USER_STUDENT US, PROGRAM P
WHERE A.UNIVERSITY_ID = U.UNIVERSITY_ID AND A.STUDENT_USER_ID = US.USER_ID
      AND A.PROGRAM_ID = P.PROGRAM_ID AND A.UNIVERSITY_ID = P.UNIVERSITY_ID
      AND A.authorize_decision = "admitted"
GROUP BY A.UNIVERSITY_ID, A.PROGRAM_ID;
```

```
SELECT AG.UNIVERSITY_NAME, AG.PROGRAM_NAME, AG.AVERAGE_GRE
FROM AVG_GRE AG
WHERE 3 > (SELECT COUNT(*)
           FROM AVG_GRE AG2
           WHERE AG.AVERAGE_GRE < AG2.AVERAGE_GRE)
ORDER BY AG.AVERAGE_GRE DESC;
```



	university_name	PROGRAM_NAME	AVERAGE_GRE
▶	University of Miami	CS	319.1154
	Carleton University	IS	318.9167
	Thunderbird School of Global Management	CS	318.8077

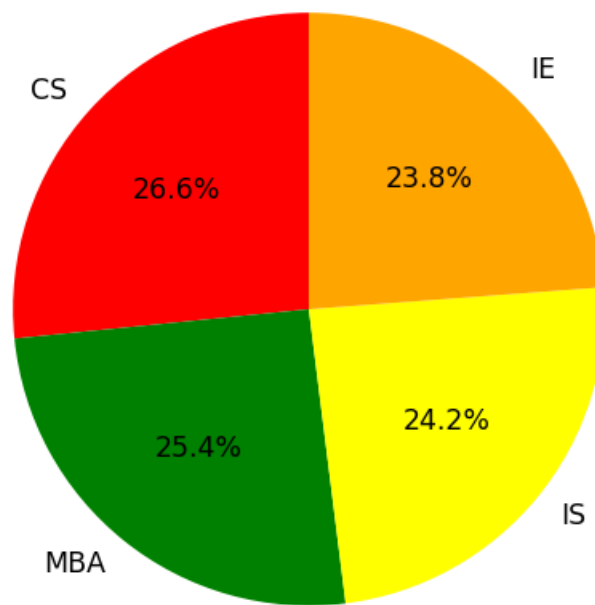
### ACCESSING THE MYSQL DATABASE USING PYTHON:

By Connecting the MYSQL database to Python following plots were generated

**1) Analytic purpose:**

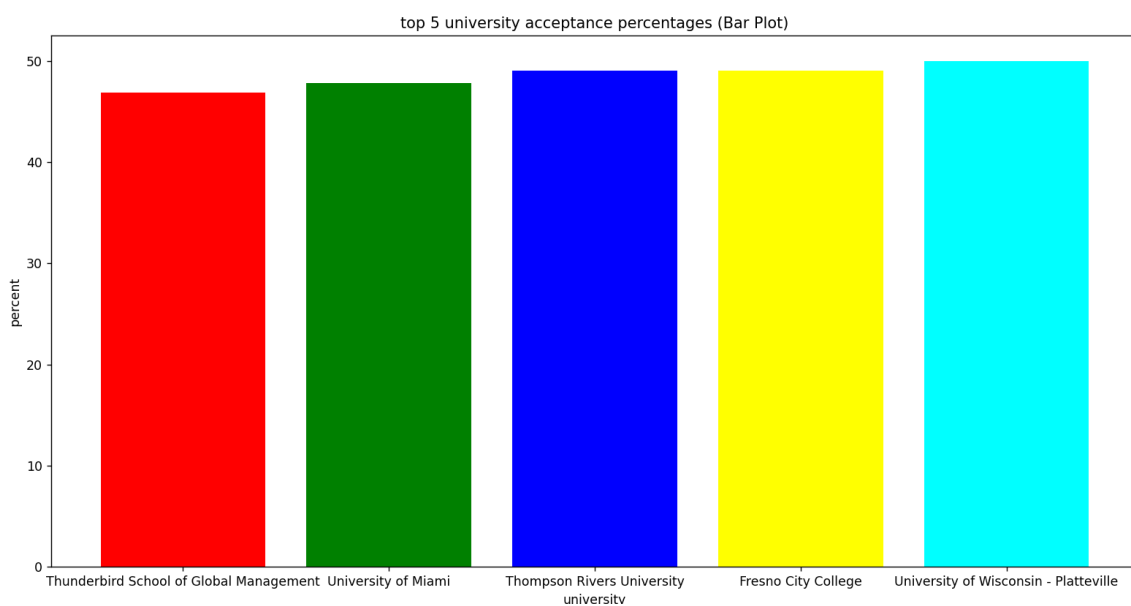
**Created a pie chart to show which program is most popular.**

**Total APPLICATIONS FOR EACH PROGRAM (Pie Chart)**



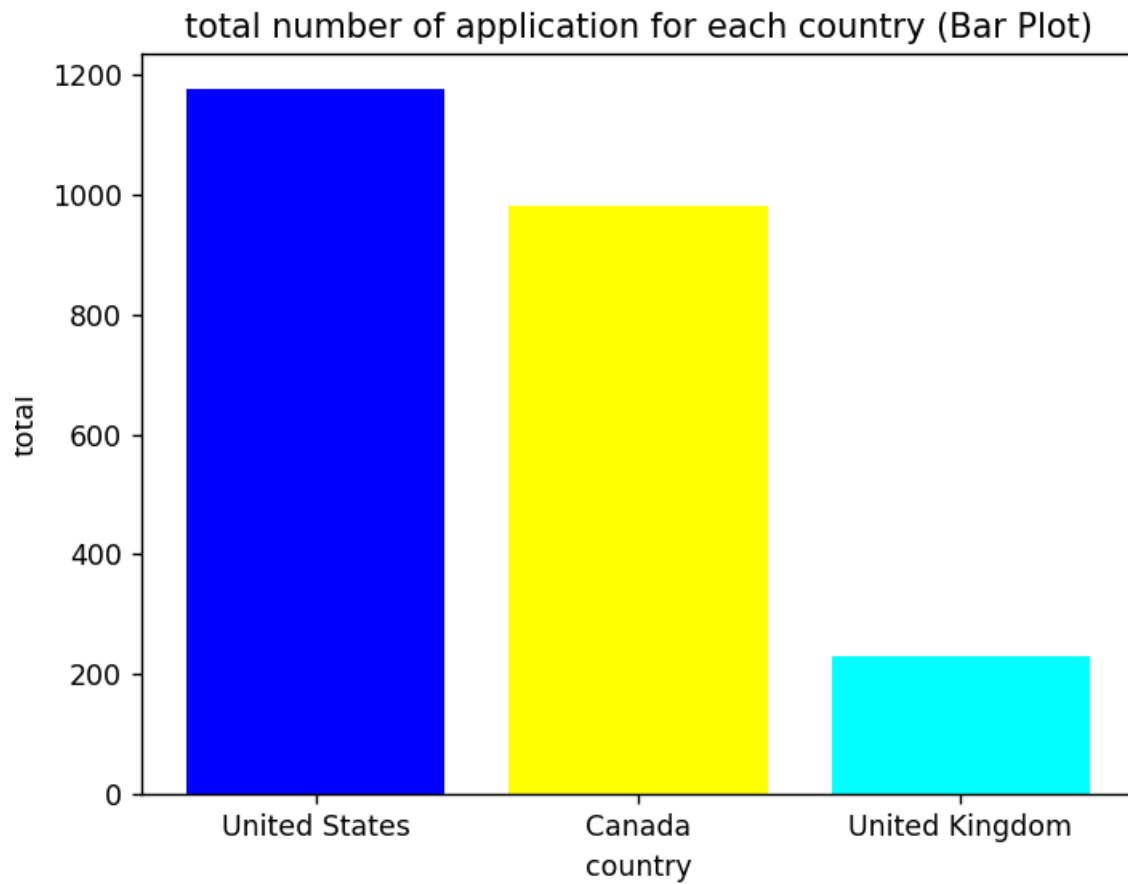
**2) Analytic purpose:**

**Created a bar plot to show the top 5 universities which are most difficult to get admission into (lowest acceptance rate).**



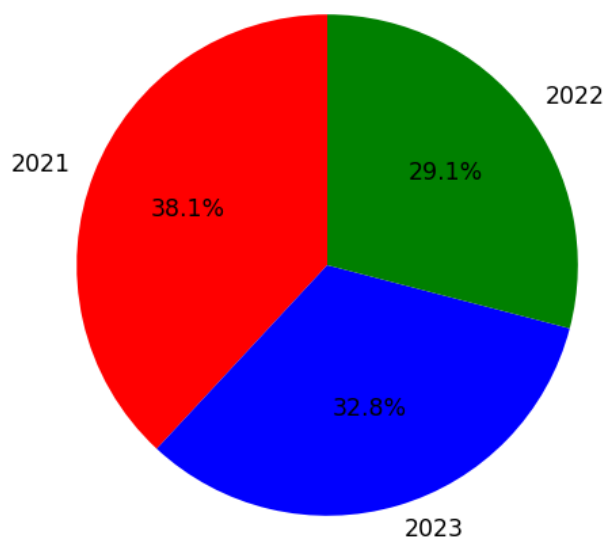
3) Analytic purpose:

finding out which country is the most popular for education to help students decide.



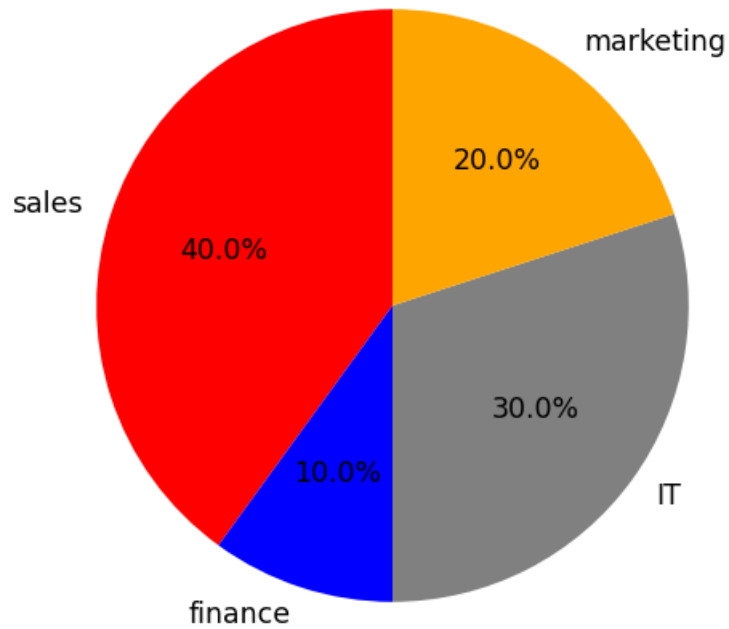
4) Analytic purpose: compare how much revenue the firm generated each year.

Total Revenue by Year (Pie Chart)



5) Analytic purpose: compare how employees are there in each department

Total EMPLOYEES FOR EACH DEPARMENT (Pie Chart)



### NOSQL implementation:

Application, bill, program, university, user\_employee, user\_student tables were created in MONGODB compass(GUI) and the following queries were run

#### 1) Analytic purpose:

**calculate and compare how much revenue the firm generated each year.**

Query to calculate total revenue per year in descending order.

```
db.bill.aggregate([
  {$project:{year:{$year:{$dateFromString:{dateString: "$BILL_date"}}},bill_amount:1}},
  {$group: {_id:"$year", totalamount: {$sum:"$bill_amount"}}},
  {$sort:{totalamount: -1}},
  {$project:{_id: 0, year: "$_id", totalamount:1}}
])
```

```
> db.bill.aggregate([{$project:{year:{$year:{$dateFromString:{dateString: "$BILL_date"}}},bill_amount:1}},
  {$group: {_id:"$year", totalamount: {$sum:"$bill_amount"}}},
  {$sort:{totalamount: -1}},
  {$project:{_id: 0, year: "$_id", totalamount:1}}])
< {
  totalamount: 58944,
  year: 2021
}
{
  totalamount: 50832,
  year: 2023
}
{
  totalamount: 44979,
  year: 2022
}
```

#### 2) Analytic purpose:

**Find out which department has the highest number of employees.**

the query to calculate the total number of employees in each department in descending order.

```
db.user_employee.aggregate([
  {$group: {_id: "$department_id", numberofemployees: {$sum:1}}},
  {$project:{_id:0,department_id:"$_id",numberofemployees:1}},
  {$sort:{numberofemployees: -1}}
])
```

```

> db.user_employee.aggregate([{$group: {_id: "$department_id", numberofemployees: {$sum:1}}},
{$project: {_id:0, department_id:"$ _id", numberofemployees:1}},
{$sort:{numberofemployees: -1}}])
< {
  numberofemployees: 4,
  department_id: 4
}
{
  numberofemployees: 4,
  department_id: 1
}
{
  numberofemployees: 1,
  department_id: 3
}
{
  numberofemployees: 1,
  department_id: 2
}

```

### 3) Analytic purpose:

#### finding out the most popular university.

The query is to calculate the total number of applications to all universities in descending order.

```

db.applications.aggregate([
{$group: {_id: "$university_id", numberofapplications: {$sum:1}}},
{$project :{_id: 0, university_id:"$ _id", numberofapplications:1}},
{$sort:{numberofapplications: -1}},
{$limit: 3}
])

```

```

> db.applications.aggregate([{$group: {_id: "$university_id", numberofapplications: {$sum:1}}},
{$project :{_id: 0, university_id:"$ _id", numberofapplications:1}},
{$sort:{numberofapplications: -1}},
{$limit: 3}])
< {
  numberofapplications: 275,
  university_id: 1
}
{
  numberofapplications: 261,
  university_id: 8
}
{
  numberofapplications: 253,
  university_id: 3
}

```



#### 4) Analytic purpose:

**finding out which country is the most popular for education to help students decide.**

The query is written to find the total number of applications by country in descending order.

```
db.applications.aggregate([
  {$lookup: {from: "university", localField: "university_id", foreignField: "university_id", as:
"university"}},
  {$unwind: "$university"},
  {$group: {_id: "$university.country", NumberofApplications: { $sum: 1 }}},
  {$project: {Country: "$_id", NumberofApplications: 1, _id: 0}},
  {$sort:{NumberofApplications: -1}}
])
```

```
> db.applications.aggregate([
  {$lookup: {from: "university", localField: "university_id", foreignField: "university_id", as: "university"}},
  {$unwind: "$university"},
  {$group: {_id: "$university.country", NumberofApplications: { $sum: 1 }}},
  {$project: {Country: "$_id", NumberofApplications: 1, _id: 0}},
  {$sort:{NumberofApplications: -1}}
])
< {
  NumberofApplications: 1177,
  Country: 'United States'
}
{
  NumberofApplications: 983,
  Country: 'Canada'
}
{
  NumberofApplications: 229,
  Country: 'United Kingdom'
}
```

#### 5) How much GPA does a student need to get an admit to a program?

The query is written to get the average GPA of admitted students and top 3 are displayed.

```
db.applications.aggregate([
  {$match:{"authorize_decision":"admitted"}},
  {$lookup:{from:"user_student",
    localField:"student_user_id",
    foreignField:"user_id",
    as: "user_student"}},
  {$unwind:"$user_student"},
  {$lookup: {from: "university",
    localField: "university_id",
    foreignField: "university_id",
    as: "university"}},
  {$unwind: "$university"},
  {$group:{_id: "$university.university_name", averageGPA:{ $avg:"$user_student.GPA"}},
  {$project: {university:"$_id", averageGPA: 1, _id:0}},
  {$sort:{averageGPA: -1}},
  {$limit:3}}])
```

```

> db.applications.aggregate([
  {$match:{"authorize_decision":"admitted"}},
  {$lookup:{from:"user_student", localField:"student_user_id", foreignField:"user_id", as: "user_student"}},
  {$unwind:"$user_student"},
  {$lookup: {from: "university", localField: "university_id", foreignField: "university_id", as: "university"}},
  {$unwind: "$university"},
  {$group:{_id: "$university.university_name", averageGPA:{$avg:"$user_student.GPA"}}},
  {$project: {university:"$_id", averageGPA: 1, _id:0}},
  {$sort:{averageGPA: -1}},
  {$limit:3}])
< {
  averageGPA: 3.304576271186441,
  university: 'University of Wisconsin - Platteville'
}
{
  averageGPA: 3.27093220338983,
  university: 'University of Portsmouth'
}
{
  averageGPA: 3.2702142857142857,
  university: 'University of Windsor'
}

```