/\*

**STUDENT DATABASE**

**Business Requirements:**

The main idea of this project is to construct a student database for a University, to manage the data related to the student application process.

I have assumed the following requirements:

1. The University must be able to maintain the degrees information (Degree names like Data Analytics, and Degree level such as Bachelor/Master/PhD/Post Doc etc.) that it offers to the prospective students

2. For each degree offered, the University must be able to maintain information on acceptable pre-requisite degrees for admission into the offered degree programs.

3. The University must be able to maintain the fee details (both In-state, Out-state fee) for each degree it offers. For instance, in most of the Universities, the fees for MBA degree is more than other Master programs. So the University must be able to maintain different fees for different programs.

4. The University offers subsidized fee (in-state fee) for some International students, whose country of citizenship is an under developed country. The University will obtain the list of various countries and their per-capita details annually, and classify the countries into 4 groups - LOW, LOWER MIDDLE, UPPER MIDDLE and HIGH Incomes. Students whose citizenship is from "LOW" per-capita countries, are offered In-State fee, by default. The University obtains the countries information from the following website:

[**http://www.iawp.org/joiniawp/countrylist.htm**](http://www.iawp.org/joiniawp/countrylist.htm)

(A program in R will be built to parse the data from this website, and load the countries information into one of the tables)

5. The database must store prospective applicant details (such as his first name, middle initial, last name, address, city, state, country (where he currently lives), and his country of citizenship. The country of citizenship will help the University to decide whether the student should be offered In-state fee or not.

6. Each prospective applicant must be able to apply to more than one programs (if desired).

7. Each prospective student must be able to track his application status.

9. Each prospective student must be able to withdraw his application, and the student details must be deleted from the system.

10. The University should be able to store whether the degree is available online or class or both.

Here are the list of **11 tables** I will be building (as per physical data model), to support Student application system business requirements given above:

1. **APPLICATION\_STATUS\_CODES**

This table stores various application status codes, such as "Application Created", "Application Submitted" etc.

1. **EDUCATION\_LEVELS**

This table stores all the education levels, such as "High School", "Bachelors" etc.

1. **INCOME\_CODES**

This table stores all the Income codes, such as LOW, HIGH, LOWER MIDDLE etc.

1. **COUNTRY\_INCOME\_CLASSIFICATION**

This table stores all the countries and their Income levels. This table will be used to determine if a student belongs to LOW Income country

1. **IN\_STATE\_FEE\_CODES**

This table will store all the fee codes of in-state fee. The codes will be referenced by other tables to represent in-state fee

1. **OUT\_STATE\_FEE\_CODES**

This table will store all the fee codes of out-state fee. The codes will be referenced by other tables to represent out-state fee

1. **DEGREE\_LEVEL**

This table stores the degree levels, such as BACHELOR, MASTER, PhD. etc.

1. **DEGREES\_OFFERED**

This table stores the degrees offered by the University.

1. **DEGREE\_PRE\_REQUISITES**

This table stores the acceptable education levels for admission into each degree offered.

1. **STUDENT\_PROFILE**

This table stores all the student details

1. **STUDENT\_APPLICATION**

This table stores all the applications information, when a student applies to various degree programs.

**NOTE on relationships:**

1. At the logical level, there is a Many-To-Many relationship between Education\_Codes and Degrees\_Offered. A student can obtain admission into a degree program with multiple levels of education, and the same education level can be referenced by more than one degree program. Hence the Many-To-Many relationship between Education\_Codes and Degrees\_Offered entities. At the physical level, this relationship is broken into One-To-Many relationships, since Many-To-Many relationships cannot be represented at physical level, without an associate table. In this example, the Many-To-Many relationship between Education\_Codes and Degrees\_Offered is broken down into 3 tables (at physical level): Degrees\_Offered, Degree\_Pre\_Requisites and Education\_Levels. The Degrees\_Offered table will contain degrees list, Education\_Levels will contain all the education levels, and the associate table - Degree\_Pre\_Requisites, will map the acceptable education levels to various degree programs at the University. For instance, the University can accept students with Bachelor degree (of 3 years) and Bachelor’s Degree (of 4 years) into its MBA program, while the Masters in Data Analytics program needs Bachelor’s degree (of 4 years). All such information is represented by three tables: Degrees\_Offered, Degree\_Pre\_Requisites and Education\_Levels

2. Among all other entities in the logical and physical models, we have One-To-Many relationships. Please refer to the attached "student Database Design.pdf" document for logical and physical database design. The document can be found at

[**https://github.com/msekhar12/MSDA\_PROJECT\_1.git**](https://github.com/msekhar12/MSDA_PROJECT_1.git)

All the reporting queries will be listed at the end of this document, as the last section. REPROTING QUERIES.

Between STUDENT\_PROFILE and STUDENT\_APPLICATION Tables, we have One-To-Many relationship, with ON DELETE CASCADE rule for foreign key. Hence if a student profile is deleted from STUDENT\_PROFILE Table, then it will have a cascading effect on STUDENT\_APPLICATION Table, and all the relevant rows from STUDENT\_APPLICATION will also be deleted. Except for this relationship, all other relationships are ON DELETE RESTRICT.

\*/

/\* The DDL - Data Definition Language statements follow \*/

/\*Create **Table: 1** **-** **APPLICATION\_STATUS\_CODES Table**\*/

**DROP TABLE IF EXISTS APPLICATION\_STATUS\_CODES CASCADE;**

**CREATE TABLE APPLICATION\_STATUS\_CODES(STATUS\_ID INTEGER NOT NULL PRIMARY KEY,STATUS\_DESCRIPTION CHAR(60) NOT NULL);**

/\*

The application status table, will maintain various application statuses. The current system will maintain 5 application statuses:

1. APPLICATION CREATED

2. APPLICATION SUBMITTED

3. WAITING FOR PENDING DOCUMENTS

4. APPLICATION UNDER REVIEW

5. DECISION AVAILABLE

The University can add any new application statuses, if needed, by inserting data into this table.

\*/

**INSERT INTO APPLICATION\_STATUS\_CODES VALUES(1,'APPLICATION CREATED');**

**INSERT INTO APPLICATION\_STATUS\_CODES VALUES(2,'APPLICATION SUBMITTED');**

**INSERT INTO APPLICATION\_STATUS\_CODES VALUES(3,'WAITING FOR PENDING DOCUMENTS');**

**INSERT INTO APPLICATION\_STATUS\_CODES VALUES(4,'APPLICATION UNDER REVIEW');**

**INSERT INTO APPLICATION\_STATUS\_CODES VALUES(5,'DECISION AVAILABLE');**

/\*

To display all the application status codes use the following query

\*/

**SELECT \* FROM APPLICATION\_STATUS\_CODES;**

/\* CREATE **Table: 2 - EDUCATION\_LEVELS** \*/

/\*The EDUCATION\_LEVELS table will maintain all the education levels, that any typical applicant would have. This table will also be useful to maintain the pre-requisite education needed for various degree programs the University offers.

\*/

**DROP TABLE IF EXISTS EDUCATION\_LEVELS CASCADE;**

**CREATE TABLE EDUCATION\_LEVELS(EDUCATION\_ID INTEGER NOT NULL PRIMARY KEY,EDUCATION\_DESCRIPTION VARCHAR(250) NOT NULL);**

/\* Insert education levels information into EDUCATION\_LEVELS Table\*/

**INSERT INTO EDUCATION\_LEVELS VALUES(1,'NO HIGH SCHOOL');**

**INSERT INTO EDUCATION\_LEVELS VALUES(2,'HIGH SCHOOL');**

**INSERT INTO EDUCATION\_LEVELS VALUES(3,'10+2 OR INTERMEDIATE OR PRE-UNIVERSITY');**

**INSERT INTO EDUCATION\_LEVELS VALUES(4,'BACHELORS DEGREE - 3 YEARS');**

**INSERT INTO EDUCATION\_LEVELS VALUES(5,'BACHELORS DEGREE - 4 YEARS');**

**INSERT INTO EDUCATION\_LEVELS VALUES(6,'MASTERS DEGREE');**

**INSERT INTO EDUCATION\_LEVELS VALUES(7,'Ph. D');**

**INSERT INTO EDUCATION\_LEVELS VALUES(8,'Post Doctorate');**

/\* To display all the Education levels, use this query\*/

**SELECT \* FROM EDUCATION\_LEVELS;**

/\*CREATE **Table: 3 – INCOME\_CODES**\*/

/\*The INCOME\_CODES Table will maintain various income levels available. The University can add any new Income levels, if needed in future\*/

**DROP TABLE IF EXISTS INCOME\_CODES CASCADE;**

**CREATE TABLE INCOME\_CODES(INCOME\_ID INTEGER NOT NULL PRIMARY KEY,**

**INCOME\_CLASS CHAR(60) NOT NULL);**

/\*Insert data into INCOME\_CODES\*/

**INSERT INTO INCOME\_CODES VALUES(1,'HIGH INCOME');**

**INSERT INTO INCOME\_CODES VALUES(2,'UPPER MIDDLE');**

**INSERT INTO INCOME\_CODES VALUES(3,'LOWER MIDDLE');**

**INSERT INTO INCOME\_CODES VALUES(4,'LOW');**

/\*To get the current data from INCOME\_CODES Table, use this query\*/

**SELECT \* FROM INCOME\_CODES;**

/\* CREATE **Table: 4 - COUNTRY\_INCOME\_CLASSIFICATION** \*/

/\*The COUNTRY\_INCOME\_CLASSIFICATION Table will maintain information about all the countries in the world, and the country classification

- LOW or HIGH or UPPER MIDDLE or LOWER MIDDLE, depending on the per-capita income. This table will be loaded periodically by parsing the data from the following website:

[**http://www.iawp.org/joiniawp/countrylist.htm**](http://www.iawp.org/joiniawp/countrylist.htm)

The following R-Program Code can scrape the data present at the above website, and create a file (CSV) that can be readily loaded into the table **COUNTRY\_INCOME\_CLASSIFICATION**

The following R-CODE Must be executed from R-Studio, and NOT from the PostgreSql Editor. The R following R-Code can be downloaded from my github folder (in the file name **Parse\_Income\_levels.R**):

[**https://github.com/msekhar12/MSDA\_PROJECT\_1.git**](https://github.com/msekhar12/MSDA_PROJECT_1.git)

**#Install the XML Package, if needed, using the following command**

**#install.packages("XML")**

**#This function will parse the data from the website: "http://www.iawp.org/joiniawp/countrylist.htm"**

**#and creates a file which can be loaded into a postgres table.**

**library("XML")**

**u <- "http://www.iawp.org/joiniawp/countrylist.htm"**

**tbls <- readHTMLTable(u,which=1,colClasses=c("integer","character","character","character","character"))**

**#We are interested in the following rows and columns only**

**tbls[6:213,c(3,4,7)]**

**#Cleanse the data**

**# tbls$class <- ifelse(grepl("Lower middle",tbls[,7]),'Lower Middle',tbls[,7])**

**# tbls$class <- ifelse(grepl("High income",tbls[,7]),'High Income',tbls$class)**

**# tbls$class <- ifelse(grepl("Upper middle",tbls[,7]),'Upper Middle',tbls$class)**

**# tbls$class <- ifelse(grepl("Low income",tbls[,7]),'Low',tbls$class)**

**tbls$class <- ifelse(grepl("Lower middle",tbls[,7]),3,tbls[,7])**

**tbls$class <- ifelse(grepl("High income",tbls[,7]),1,tbls$class)**

**tbls$class <- ifelse(grepl("Upper middle",tbls[,7]),2,tbls$class)**

**tbls$class <- ifelse(grepl("Low income",tbls[,7]),4,tbls$class)**

**tbls[6:213,c(3,4,7,10)]**

**#Move the desired columns to another data frame**

**tbls\_mod <- tbls[6:213,c(3,4,10)]**

**names(tbls\_mod) <- c("Country", "Country\_Code","Income Classification")**

**#Clean any unnecessary commas in col1**

**tbls\_mod[,1] <- gsub("\\, The","",tbls\_mod[,1])**

**tbls\_mod[,1] <- gsub("\\,","",tbls\_mod[,1])**

**write.csv(tbls\_mod,file = "income\_levels.csv", quote=F,row.names=F)**

\*/

**DROP TABLE IF EXISTS COUNTRY\_INCOME\_CLASSIFICATION CASCADE;**

**CREATE TABLE COUNTRY\_INCOME\_CLASSIFICATION(COUNTRY\_ID SERIAL PRIMARY KEY, COUNTRY\_CODE CHAR(10) NOT NULL,**

**COUNTRY\_NAME VARCHAR(254) NOT NULL, INCOME\_ID INTEGER NOT NULL,**

**CONSTRAINT INCOME\_ID\_FK FOREIGN KEY(INCOME\_ID) REFERENCES INCOME\_CODES(INCOME\_ID) ON DELETE RESTRICT**

**);**

/\*Use the following COPY Command to LOAD the CSV File into the COUNTRY\_INCOME\_CLASSIFICATION. The input CSV file will be created by an R Program, by parsing the data from the website:

[**http://www.iawp.org/joiniawp/countrylist.htm**](http://www.iawp.org/joiniawp/countrylist.htm)

**NOTE:** The R-Code is given in the previous comment above. Or you can also get the R-Code used from the file **Parse\_Income\_levels.R** in the following Git location

[**https://github.com/msekhar12/MSDA\_PROJECT\_1.git**](https://github.com/msekhar12/MSDA_PROJECT_1.git)

NOTE that the generated CSV File must be stored at the location to which the LOAD Job has access to. Also the permissions on the CSV file must be changed to at least to READ. In Unix, you may use the command: ***chmod 666 <file\_name> or chmod 777 <file\_name>***. In Windows, the file permissions can be changed, by right clicking on the file, and sharing the file with everyone or to specific Users who would load the CSV file to the **COUNTRY\_INCOME\_CLASSIFICATION** Table.

\*/

**COPY COUNTRY\_INCOME\_CLASSIFICATION (COUNTRY\_NAME,COUNTRY\_CODE, INCOME\_ID)**

**FROM 'C:\Users\Sekhar\Documents\CUNY\607 Assignments\income\_levels.CSV' DELIMITER ',' CSV HEADER;**

/\*The following query will display all the countries and their income classifications\*/

**SELECT A.COUNTRY\_ID, A.COUNTRY\_CODE, A.COUNTRY\_NAME, A.INCOME\_ID, B.INCOME\_CLASS**

**FROM**

**COUNTRY\_INCOME\_CLASSIFICATION A, INCOME\_CODES B WHERE A.INCOME\_ID = B.INCOME\_ID;**

/\*CREATE **TABLE: 5** - **IN\_STATE\_FEE\_CODES** \*/

/\*The IN\_STATE\_FEE\_CODES will maintain the in-state fee information\*/

**DROP TABLE IF EXISTS IN\_STATE\_FEE\_CODES CASCADE;**

**CREATE TABLE IN\_STATE\_FEE\_CODES**

**(**

**FEE\_ID INTEGER NOT NULL PRIMARY KEY,**

**CREDIT\_HOUR\_FEE DECIMAL(30,2) NOT NULL**

**);**

/\*Insert some sample data\*/

**INSERT INTO IN\_STATE\_FEE\_CODES VALUES(1,450.00);**

**INSERT INTO IN\_STATE\_FEE\_CODES VALUES(2,650.00);**

**INSERT INTO IN\_STATE\_FEE\_CODES VALUES(3,350.00);**

/\*TO SELECT all the data from IN\_STATE\_FEE\_CODES table, use this query:\*/

**SELECT \* FROM IN\_STATE\_FEE\_CODES;**

/\*CREATE **TABLE: 6** - **OUT\_STATE\_FEE\_CODES** \*/

/\*The OUT\_STATE\_FEE\_CODES will maintain the Out-state fee information\*/

**DROP TABLE IF EXISTS OUT\_STATE\_FEE\_CODES CASCADE;**

**CREATE TABLE OUT\_STATE\_FEE\_CODES**

**(**

**FEE\_ID INTEGER NOT NULL PRIMARY KEY,**

**CREDIT\_HOUR\_FEE DECIMAL(30,2) NOT NULL**

**);**

/\*Insert some sample data\*/

**INSERT INTO OUT\_STATE\_FEE\_CODES VALUES(1,850.00);**

**INSERT INTO OUT\_STATE\_FEE\_CODES VALUES(2,1050.00);**

**INSERT INTO OUT\_STATE\_FEE\_CODES VALUES(3,750.00);**

/\*TO SELECT all the data from OUT\_STATE\_FEE\_CODES table, use this query:\*/

**SELECT \* FROM OUT\_STATE\_FEE\_CODES;**

/\*CREATE **TABLE: 7** – **DEGREE\_LEVEL** \*/

/\*

The DEGREE\_LEVEL Table will maintain all the available degree levels that the University could potentially offer.

Even though the University does NOT necessarily offer all the degree programs at all the levels currently, the University must still be able to maintain the possible degree levels, it could offer in future, using this table.

\*/

**DROP TABLE IF EXISTS DEGREE\_LEVEL CASCADE;**

**CREATE TABLE DEGREE\_LEVEL(LEVEL INTEGER NOT NULL PRIMARY KEY,LEVEL\_NAME CHAR(50) NOT NULL);**

/\*Insert some sample data\*/

**INSERT INTO DEGREE\_LEVEL VALUES(1,'BACHELOR DEGREE');**

**INSERT INTO DEGREE\_LEVEL VALUES(2,'MASTERS DEGREE');**

**INSERT INTO DEGREE\_LEVEL VALUES(3,'Ph.D');**

**INSERT INTO DEGREE\_LEVEL VALUES(4,'Post Doc.');**

/\*To SELECT all the degree levels available use the following query\*/

**SELECT \* FROM DEGREE\_LEVEL;**

/\* CREATE **TABLE: 8** - **DEGREES\_OFFERED** \*/

/\*The **DEGREES\_OFFERED** Table will maintain all the degrees the University currently offers\*/

**DROP TABLE IF EXISTS DEGREES\_OFFERED CASCADE;**

**CREATE TABLE DEGREES\_OFFERED**

**(**

**DEGREE\_ID INTEGER NOT NULL PRIMARY KEY,**

**CREDIT\_HOURS INTEGER NOT NULL,**

**IN\_STATE\_FEE\_ID INTEGER NOT NULL REFERENCES IN\_STATE\_FEE\_CODES(FEE\_ID),**

**OUT\_STATE\_FEE\_ID INTEGER NOT NULL REFERENCES OUT\_STATE\_FEE\_CODES(FEE\_ID),**

**AVAILABLE\_ONLINE CHAR(1) NOT NULL CONSTRAINT CHK\_1 CHECK(AVAILABLE\_ONLINE IN ('Y','N')),**

**AVAILABLE\_CLASS\_ROOM CHAR(1) NOT NULL CONSTRAINT CHK\_2 CHECK(AVAILABLE\_CLASS\_ROOM IN ('Y','N')),**

**DEGREE\_NAME CHAR(100) NOT NULL,**

**DEGREE\_LEVEL INTEGER NOT NULL REFERENCES DEGREE\_LEVEL(LEVEL)**

**);**

/\*Insert some data into the degrees offered table\*/

**INSERT INTO DEGREES\_OFFERED VALUES**

**(**

**1,60,1,1,'Y','Y','Business Administration',1);**

**INSERT INTO DEGREES\_OFFERED VALUES**

**(**

**2,40,2,2,'Y','Y','Business Administration',2);**

**INSERT INTO DEGREES\_OFFERED VALUES**

**(**

**3,36,3,3,'Y','N','Data Analytics',2);**

/\* To select the degrees offered, use this query.\*/

**SELECT \* FROM DEGREES\_OFFERED;**

/\*The above query will not display meaningful information, since everything is displayed in codes. To see some meaningful information

use the following query. This query will list the degrees offered, levels (Masters/Bachelors etc., if the degree available online or offline or both and finally the in-state and out-state fee per credit hour)\*/

**SELECT A.DEGREE\_NAME, B.LEVEL\_NAME, A.AVAILABLE\_ONLINE,**

**A.AVAILABLE\_CLASS\_ROOM, C.CREDIT\_HOUR\_FEE IN\_STATE\_FEE\_PER\_CREDIT\_HR,D.CREDIT\_HOUR\_FEE OUT\_STATE\_FEE\_PER\_CREDIT\_HR**

**FROM**

**DEGREES\_OFFERED A, DEGREE\_LEVEL B, IN\_STATE\_FEE\_CODES C, OUT\_STATE\_FEE\_CODES D**

**WHERE A.DEGREE\_LEVEL = B.LEVEL AND A.IN\_STATE\_FEE\_ID = C.FEE\_ID AND A.OUT\_STATE\_FEE\_ID = D.FEE\_ID;**

/\*CREATE **Table: 9 - DEGREE\_PRE\_REQUISITES** \*/

/\*The DEGREE\_PRE\_REQUISITES Table will associate the pre-requisite degrees acceptable for admission into various degree programs at the University\*/

**DROP TABLE IF EXISTS DEGREE\_PRE\_REQUISITES CASCADE;**

**CREATE TABLE DEGREE\_PRE\_REQUISITES**

**(**

**DEGREE\_ID INTEGER NOT NULL,**

**EDUCATION\_ID INTEGER NOT NULL,**

**CONSTRAINT DEGREE\_PRE\_REQ\_PK PRIMARY KEY(DEGREE\_ID,EDUCATION\_ID),**

**CONSTRAINT DEGREE\_PRE\_REQ\_FK1 FOREIGN KEY(DEGREE\_ID) REFERENCES DEGREES\_OFFERED,**

**CONSTRAINT DEGREE\_PRE\_REQ\_FK2 FOREIGN KEY(EDUCATION\_ID) REFERENCES EDUCATION\_LEVELS**

**);**

/\*Insert some sample data into DEGREE\_PRE\_REQUISITES Table\*/

INSERT INTO DEGREE\_PRE\_REQUISITES

VALUES

(

1,3

);

INSERT INTO DEGREE\_PRE\_REQUISITES

VALUES

(

2,5

);

INSERT INTO DEGREE\_PRE\_REQUISITES

VALUES

(

2,4

);

INSERT INTO DEGREE\_PRE\_REQUISITES

VALUES

(

3,5

);

/\*To determine all the pre-requisite degrees needed for all the offered degree programs at the University, Use the following query\*/

**SELECT A.DEGREE\_NAME, D.LEVEL\_NAME,B.EDUCATION\_DESCRIPTION ENROLLMENT\_PRE\_REQ FROM DEGREES\_OFFERED A,**

**EDUCATION\_LEVELS B, DEGREE\_PRE\_REQUISITES C, DEGREE\_LEVEL D**

**WHERE A.DEGREE\_ID = C.DEGREE\_ID AND B.EDUCATION\_ID = C.EDUCATION\_ID AND D.LEVEL = A.DEGREE\_LEVEL;**

/\*CREATE **Table 10** - **STUDENT\_PROFILE** \*/

/\*The STUDENT\_PROFILE Table will maintain information about the students, who create a student profile. \*/

**DROP TABLE IF EXISTS STUDENT\_PROFILE CASCADE;**

**CREATE TABLE STUDENT\_PROFILE**

**(**

**STUDENT\_ID SERIAL NOT NULL PRIMARY KEY,**

**FIRST\_NAME CHAR(60) NOT NULL,**

**MIDDLE\_INITIAL CHAR(1) NULL,**

**LAST\_NAME CHAR(60) NOT NULL,**

**ADDRESS\_LINE\_1 CHAR(120) NOT NULL,**

**ADDRESS\_LINE\_2 CHAR(120) NULL,**

**CITY CHAR(60) NOT NULL,**

**STATE CHAR(60) NULL,**

**COUNTRY CHAR(100) NOT NULL,**

**ZIP CHAR(10) NOT NULL,**

**EMAIL CHAR(120) NOT NULL,**

**PHONE CHAR(60) NULL,**

**GENDER CHAR(1) NULL,**

**HIGHEST\_EDUCATION INTEGER NOT NULL,**

**COUNTRY\_OF\_CITIZENSHIP INTEGER NOT NULL,**

**CREATE\_TIMESTAMP TIMESTAMP NOT NULL,**

**UPDATE\_TIMESPAMT TIMESTAMP NOT NULL,**

**CONSTRAINT STUDENT\_FK\_1 FOREIGN KEY(HIGHEST\_EDUCATION) REFERENCES EDUCATION\_LEVELS ON DELETE RESTRICT,**

**CONSTRAINT STUDENT\_FK\_2 FOREIGN KEY(COUNTRY\_OF\_CITIZENSHIP) REFERENCES COUNTRY\_INCOME\_CLASSIFICATION ON DELETE RESTRICT**

**);**

**INSERT INTO STUDENT\_PROFILE VALUES**

**(**

**1,'SAMUEL',NULL,'ALEX','34 SOMERSET ROAD',NULL,'OMAHA','NE', 'USA','61229', 'XYZ@DUMMY.COM',NULL,'M',3,99,current\_timestamp,current\_timestamp);**

**INSERT INTO STUDENT\_PROFILE VALUES**

**(**

**2,'NIKITA',NULL,'YURI','122 ROSEWOOD DR',NULL,'HOLLYWOOD','CA', 'USA','91224', 'ABC@DUMMY.COM',NULL,'M',5,154,current\_timestamp,current\_timestamp);**

**INSERT INTO STUDENT\_PROFILE VALUES**

**(**

**3, 'AMIT',NULL,'SAXENA','129 WATERFRONT PL',NULL,'JERSEY CITY','NJ', 'USA','08123', '123@DUMMY.COM',NULL,'M',5,87,current\_timestamp,current\_timestamp);**

**INSERT INTO STUDENT\_PROFILE VALUES**

**(**

**4, 'JEFF',NULL,'SHIAU','XEN AV',NULL,'SHANGHAI','EA', 'CHINA','XRRR', '123@DUMMY.COM',NULL,'M',3,41,current\_timestamp,current\_timestamp);**

**INSERT INTO STUDENT\_PROFILE VALUES**

**(**

**5, 'DIANA',NULL,'YATES','EMERALD AVE',NULL,'ROCHESTER','MI', 'USA','48091', '123@DUMMY.COM',NULL,'F',6,204,current\_timestamp,current\_timestamp);**

/\*Select data from STUDENT\_PROFILE\*/

**SELECT \* FROM STUDENT\_PROFILE;**

/\*CREATE **Table: 11 - STUDENT\_APPLICATION** \*/

/\*This table will map the students and the degrees to which they have applied to. This is an associate table\*/

**DROP TABLE IF EXISTS STUDENT\_APPLICATION CASCADE;**

**CREATE TABLE STUDENT\_APPLICATION**

**(**

**STUDENT\_ID INTEGER NOT NULL,**

**DEGREE\_ID INTEGER NOT NULL,**

**STATUS\_ID INTEGER NOT NULL,**

**CREATE\_TIMESTAMP TIMESTAMP NOT NULL,**

**UPDATE\_TIMESTAMP TIMESTAMP NOT NULL,**

**CONSTRAINT STUDENT\_APP\_PK PRIMARY KEY(STUDENT\_ID,DEGREE\_ID),**

**CONSTRAINT STUDENT\_APP\_FK\_1 FOREIGN KEY(DEGREE\_ID) REFERENCES DEGREES\_OFFERED ON DELETE RESTRICT,**

**CONSTRAINT STUDENT\_APP\_FK\_2 FOREIGN KEY(STATUS\_ID) REFERENCES APPLICATION\_STATUS\_CODES ON DELETE RESTRICT,**

**CONSTRAINT STUDENT\_APP\_FK\_3 FOREIGN KEY(STUDENT\_ID) REFERENCES STUDENT\_PROFILE ON DELETE CASCADE,**

**CONSTRAINT STUDENT\_APP\_FK\_4 FOREIGN KEY(DEGREE\_ID) REFERENCES DEGREES\_OFFERED ON DELETE RESTRICT**

**);**

/\*Insert some sample data\*/

**INSERT INTO STUDENT\_APPLICATION**

**VALUES**

**(**

**1,1,1,CURRENT\_TIMESTAMP,CURRENT\_TIMESTAMP**

**);**

**INSERT INTO STUDENT\_APPLICATION**

**VALUES**

**(**

**2,3,2,CURRENT\_TIMESTAMP,CURRENT\_TIMESTAMP**

**);**

**INSERT INTO STUDENT\_APPLICATION**

**VALUES**

**(**

**2,2,1,CURRENT\_TIMESTAMP,CURRENT\_TIMESTAMP**

**);**

**INSERT INTO STUDENT\_APPLICATION**

**VALUES**

**(**

**3,2,4,CURRENT\_TIMESTAMP,CURRENT\_TIMESTAMP**

**);**

**INSERT INTO STUDENT\_APPLICATION**

**VALUES**

**(**

**4,1,5,CURRENT\_TIMESTAMP,CURRENT\_TIMESTAMP**

**);**

**/\*REPROTING QUERIES\*/**

/\*

**QUERY\_1:**

To find students who qualify for subsidized fee, use the following query. The column "qualifies\_for\_subsidy" shows if the student is eligible for in\_state fee or not.

In this query's output, you can observe that one student qualifies for subsidized fee, since his country of citizenship is Kenya, and Kenya is a LOW income country.

This query uses the following 3 tables:

**STUDENT\_PROFILE**

**COUNTRY\_INCOME\_CLASSIFICATION**

**INCOME\_CODES**

\*/

**SELECT A.STUDENT\_ID, A.FIRST\_NAME, A.LAST\_NAME,**

**CASE WHEN C.INCOME\_CLASS = 'LOW' THEN 'YES' ELSE 'NO'**

**END AS QUALIFIES\_FOR\_SUBSIDY, C.INCOME\_CLASS,**

**B.COUNTRY\_NAME AS COUNTRY\_OF\_CITIZENSHIP, A.COUNTRY AS CURRENT\_COUNTRY**

**FROM STUDENT\_PROFILE A, COUNTRY\_INCOME\_CLASSIFICATION B, INCOME\_CODES C**

**WHERE A.COUNTRY\_OF\_CITIZENSHIP = B.COUNTRY\_ID AND**

**C.INCOME\_ID = B.INCOME\_ID**

**ORDER BY 1;**

/\*

**QUERY\_2:**

Using the **Query-1**, we can create another report, to get all the students who applied for various programs, the program names, the program level, the required credit hours, the credit hour fee, total program fee, and the student's country of citizenship.

In this report you can find that 2 people (out of 5 people) have applied for the Business Administration degree at Bachelors level. One applicant is from Kenya, and the other from China. Since Kenya is a LOW income country, the applicant from Kenya is eligible for

in-state-fee, and hence his fee is very lesser than the fee of Chinese applicant.

This query uses the following tables 9 tables:

**STUDENT\_APPLICATION**

**APPLICATION\_STATUS\_CODES**

**DEGREES\_OFFERED**

**DEGREE\_LEVEL**

**IN\_STATE\_FEE\_CODES**

**OUT\_STATE\_FEE\_CODES**

**STUDENT\_PROFILE**

**COUNTRY\_INCOME\_CLASSIFICATION**

**INCOME\_CODES**

\*/

**SELECT A.STUDENT\_ID, A.FIRST\_NAME, A.LAST\_NAME, A.QUALIFIES\_FOR\_SUBSIDY,**

**D.DEGREE\_NAME AS APPLIED\_TO, E.LEVEL\_NAME AS APPLIED\_DEGREE\_LEVEL, C.STATUS\_DESCRIPTION APPLICATION\_STATUS,**

**D.CREDIT\_HOURS AS REQUIRED\_CREDIT\_HOURS,**

**CASE WHEN A.QUALIFIES\_FOR\_SUBSIDY = 'YES' THEN F.CREDIT\_HOUR\_FEE**

**ELSE G.CREDIT\_HOUR\_FEE END AS PROGRAM\_FEE\_PER\_CREDIT\_HOUR,**

**CASE WHEN A.QUALIFIES\_FOR\_SUBSIDY = 'YES' THEN F.CREDIT\_HOUR\_FEE \* D.CREDIT\_HOURS**

**ELSE G.CREDIT\_HOUR\_FEE \* D.CREDIT\_HOURS END AS TOTAL\_PROGRAM\_FEE,**

**A.COUNTRY\_OF\_CITIZENSHIP**

**FROM**

**(**

**SELECT A.STUDENT\_ID, A.FIRST\_NAME, A.LAST\_NAME,**

**CASE WHEN C.INCOME\_CLASS = 'LOW' THEN 'YES' ELSE 'NO'**

**END AS QUALIFIES\_FOR\_SUBSIDY, C.INCOME\_CLASS,**

**B.COUNTRY\_NAME AS COUNTRY\_OF\_CITIZENSHIP, A.COUNTRY AS CURRENT\_COUNTRY**

**FROM STUDENT\_PROFILE A, COUNTRY\_INCOME\_CLASSIFICATION B, INCOME\_CODES C**

**WHERE A.COUNTRY\_OF\_CITIZENSHIP = B.COUNTRY\_ID AND**

**C.INCOME\_ID = B.INCOME\_ID**

**) AS A,**

**STUDENT\_APPLICATION B, APPLICATION\_STATUS\_CODES C, DEGREES\_OFFERED D, DEGREE\_LEVEL E,**

**IN\_STATE\_FEE\_CODES F, OUT\_STATE\_FEE\_CODES G**

**WHERE**

**A.STUDENT\_ID = B.STUDENT\_ID AND**

**B.STATUS\_ID = C.STATUS\_ID AND**

**B.DEGREE\_ID = D.DEGREE\_ID AND**

**D.DEGREE\_LEVEL = E.LEVEL AND**

**D.IN\_STATE\_FEE\_ID = F.FEE\_ID AND**

**D.OUT\_STATE\_FEE\_ID = G.FEE\_ID**

**ORDER BY 1;**

/\*

**QUERY\_3:**

This query pulls all the students who have created a student profile, along with the details of whether the student has started working on any application or just has created the profile.

This query uses a LEFT OUTER JOIN, and it returns NULL values for students who have created a student profile, but did not start application for any program. Such NULL values are handled by

COALESCE function, to display a meaningful message like "--- NOT APPLIED YET ---”

This query uses the following 5 tables:

**STUDENT\_APPLICATION**

**DEGREES\_OFFERED**

**APPLICATION\_STATUS\_CODES**

**STUDENT\_PROFILE**

**DEGREE\_LEVEL**

\*/

**SELECT A.STUDENT\_ID, A.FIRST\_NAME, A.LAST\_NAME,**

**COALESCE(B.STATUS\_DESCRIPTION,'--- NOT APPLIED YET ---') APPLICATION\_STATUS,**

**COALESCE(B.APPLIED\_TO,'---NOT YET APPLIED YET ---') AS APPLIED\_TO,**

**COALESCE(B.DEGREE\_LEVEL,'--- NOT YET APPLIED YET ---') DEGREE\_LEVEL**

**FROM STUDENT\_PROFILE A LEFT OUTER JOIN**

**(SELECT D.STUDENT\_ID, D.FIRST\_NAME, D.LAST\_NAME, B.DEGREE\_NAME AS APPLIED\_TO, E.LEVEL\_NAME AS DEGREE\_LEVEL, C.STATUS\_DESCRIPTION**

**FROM**

**STUDENT\_APPLICATION A, DEGREES\_OFFERED B, APPLICATION\_STATUS\_CODES C, STUDENT\_PROFILE D, DEGREE\_LEVEL E**

**WHERE A.STUDENT\_ID = D.STUDENT\_ID AND**

**A.DEGREE\_ID = B.DEGREE\_ID AND A.STATUS\_ID = C.STATUS\_ID AND E.LEVEL = B.DEGREE\_LEVEL) B ON A.STUDENT\_ID = B.STUDENT\_ID**

**ORDER BY 1;**

/\*

**QUERY\_4:**

To find degrees offered, degree level and enrollment PRE-requisites

This query uses the following 4 tables:

**DEGREES\_OFFERED**

**EDUCATION\_LEVELS**

**DEGREE\_PRE\_REQUISITES**

**DEGREE\_LEVEL**

\*/

**SELECT A.DEGREE\_NAME, D.LEVEL\_NAME,B.EDUCATION\_DESCRIPTION ENROLLMENT\_PRE\_REQ FROM DEGREES\_OFFERED A,**

**EDUCATION\_LEVELS B, DEGREE\_PRE\_REQUISITES C, DEGREE\_LEVEL D**

**WHERE A.DEGREE\_ID = C.DEGREE\_ID AND B.EDUCATION\_ID = C.EDUCATION\_ID AND D.LEVEL = A.DEGREE\_LEVEL;**

/\*

**QUERY\_5:**

This query will list the degrees offered, levels (Masters/Bachelors etc., if the degree available online or offline or both and finally the in-state and out-state fee per credit hour)

This query uses the following 4 tables:

**DEGREES\_OFFERED**

**DEGREE\_LEVEL**

**IN\_STATE\_FEE\_CODES**

**OUT\_STATE\_FEE\_CODES**

\*/

**SELECT A.DEGREE\_NAME, B.LEVEL\_NAME, A.AVAILABLE\_ONLINE,**

**A.AVAILABLE\_CLASS\_ROOM, C.CREDIT\_HOUR\_FEE IN\_STATE\_FEE\_PER\_CREDIT\_HR,D.CREDIT\_HOUR\_FEE OUT\_STATE\_FEE\_PER\_CREDIT\_HR**

**FROM**

**DEGREES\_OFFERED A, DEGREE\_LEVEL B, IN\_STATE\_FEE\_CODES C, OUT\_STATE\_FEE\_CODES D**

**WHERE A.DEGREE\_LEVEL = B.LEVEL AND A.IN\_STATE\_FEE\_ID = C.FEE\_ID AND A.OUT\_STATE\_FEE\_ID = D.FEE\_ID;**

**/\* DEMONSTRATION OF ON DELETE CASCADE FUNCTIONALITY \*/**

/\*Between the tables STUDENT\_PROFILE and STUDENT\_APPLICATION, we have ON DELETE CASCADE Relationship. Hence if we delete any student record from STUDENT\_PROFILE, then all the dependent records are deleted from STUDENT\_APPLICATION.\*/

/\*To demonstrate the CASCADE DELETES, use this query to find students who are currently applied to at least one program\*/

**SELECT D.STUDENT\_ID, D.FIRST\_NAME, D.LAST\_NAME, B.DEGREE\_NAME AS APPLIED\_TO, E.LEVEL\_NAME AS DEGREE\_LEVEL, C.STATUS\_DESCRIPTION**

**FROM**

**STUDENT\_APPLICATION A, DEGREES\_OFFERED B, APPLICATION\_STATUS\_CODES C, STUDENT\_PROFILE D, DEGREE\_LEVEL E**

**WHERE A.STUDENT\_ID = D.STUDENT\_ID AND**

**A.DEGREE\_ID = B.DEGREE\_ID AND A.STATUS\_ID = C.STATUS\_ID AND E.LEVEL = B.DEGREE\_LEVEL**

**ORDER BY 1;**

/\*

The query above displays that we currently have 4 students applied to at least one program, and out of these 4 students, one student NIKITA (student ID: 2) has applied for 2 programs (Masters in Business Administration and Masters in Data Analytics). If this student withdraws the application then all the dependent rows in STUDENT APPLICATION will be deleted. Since this student has applied for 2 degree programs, 2 records must be automatically deleted from STUDENT\_APPLICATION table.

\*/

/\*

This query displays 2 records from the table STUDENT\_APPLICATION belonging to Student ID 2

\*/

**SELECT \* FROM STUDENT\_APPLICATION**

**WHERE STUDENT\_ID = 2;**

/\*

Let us delete the data related to Student ID 2 from STUDENT\_PROFILE

\*/

**DELETE FROM STUDENT\_PROFILE WHERE STUDENT\_ID = 2;**

/\*

If the following query is executed again, we get NO rows, since the data related to STUDENT ID 2 was deleted from STUDENT\_PROFILE, and since the table STUDENT\_APPLICATION is dependent on STUDENT\_PROFILE, we had a cascaded delete from STUDENT\_APPLICATION

\*/

**SELECT \* FROM STUDENT\_APPLICATION**

**WHERE STUDENT\_ID = 2;**

/\*

**FUTURE REQUIREMENTS**

\*/

/\*

1. The current database must be enhanced to store Students Login credentials (User\_ID, Encrypted password and security questions)
2. The current database must be enhanced to store if the application needs GRE/GMAT/TOEFL/SAT Scores, the minimum acceptable scores, and what was the average score in the last academic year.
3. The current database must be enhanced to store the scholarship availability, their application deadlines.

\*/