Randy Van

N01989686

DESCRIPTION:

This database was constructed for University admissions. The admissions office for universities would use this database to search and filter incoming high students based on various categories.

**Design and Implementation:**

A close up of text on a white background

Description automatically generated

**FinancialAidPackage:**(FinancialAidPackageId,Grants,Scholarships,Loans)

PK: FinancialAidPackageID (INT)

Alternate Key: Grants(VARCHAR),Scholarships (VARCHAR), Loans(VARCHAR)

The FinancialAidPackage table gives information about if a student receives grants, loans or scholarships. The table has a primary key of FinancialPackageID and alternate keys of Grants, Scholarships and Loans. Since the table is uniquely identified by the primary key it satisfies the conditions of 1st normal form. Also, since there are no partial dependencies and a single primary key with no composite keys, the table satisfies 2nd normal form. In addition, 3rd normal form is satisfied because there aren’t any transitive dependencies on the primary key. After achieving 3rd normal form the table is normalized.

**FinancialAid:**(FinancialAidID,FinancialAidPackageID,Amount)

PK: FinancialAidID(INT)

FK: FinancialAidPackageId(INT) FinancialAidPackage.FinancialAidPackageID(INT)

Alternate Key: Amount(INT)

The FinancialAid table has a primary key of FinancialAidID, a foreign key of FinancialAidID that links to the FinancialAidPackageID that links to the FinancialAid.FinancialAidID package table with an alternalte key of the Amount awarded to the student. Since the table is uniquely identified by the primary key it satisfies the conditions of 1st normal form. Also, since there are no partial dependencies and a single primary key with no composite keys, the table satisfies 2nd normal form. In addition, 3rd normal form is satisfied because there aren’t any transitive dependencies on the primary key. After achieving 3rd normal form the table is normalized.

**HighSchool:**(HighSchoolId,Name,Country,States,City,County)

PK: SchoolID(INT)

Alternate Keys:Country(VARCHAR),Name(VARCHAR),State(VARCHAR),City(VARCHAR),

County(VARCHAR)

The High school table gives information on what high school a student attended and geographical information about the high school. The HighSchool table has a primary key of HighSchoolID with alternate keys of Country, name, states, city and county. Since the table is uniquely identified by the primary key it satisfies the conditions of 1st normal form. Also, since there are no partial dependencies and a single primary key with no composite keys, the table satisfies 2nd normal form. In addition, 3rd normal form is satisfied because there aren’t any transitive dependencies on the primary key. After achieving 3rd normal form the table is normalized.

**Student:**(Studen ID,LastName,FirstName,HighSchool,Age,Gender,Ethnicity,HighSchoolID, WeightedGPA,UnweightedGPA,ClassRank,FinancialAid D,SATScore)

PK: Student ID(INT)

FK: HighSchoolID HighSchool.HighschoolID

FinancialAidID FinancialAid.FinancialAidID

Alternate Keys: StudentName(VARCHAR),Age(INT),Gender(VARCHAR), Ethnicity(VARCHAR),WeightedGPA(DECIMAL),UnweightedGPA(DECIMAL), ClassRank(INT), SATScore(INT)

The Student table describe information about the personal identity of the student and some of their academic scores. The student table has a primary key of StudentID, has a foreign key of HIghschoolID that linkes to the Highschool.HighschoolID table and a FinancialAidID that linkes to the FinancialAid.FinancialAidID table Also the table has an alternate keys of StudentName, Age, Gender, Ethnicity, WeightedGPA, UnweightedGPA, ClassRank and SATScore. Since the table is uniquely identified by the primary key it satisfies the conditions of 1st normal form. Also, since there are no partial dependencies and a single primary key with no composite keys, the table satisfies 2nd normal form. In addition, 3rd normal form is satisfied because there aren’t any transitive dependencies on the primary key. After achieving 3rd normal form the table is normalized.

**Department:**(DepartmentID,DepartmentName)

PK: DepartmentID(INT)

Alternate Keys: DepartmentName(VARCHAR)

The Department table lists the DepartmentName and the department name that the student has applied to. Department has a primary key of DepartmentID and a foreign key of DepartmentName. Since the table is uniquely identified by the primary key it satisfies the conditions of 1st normal form. Also, since there are no partial dependencies and a single primary key with no composite keys, the table satisfies 2nd normal form. In addition, 3rd normal form is satisfied because there aren’t any transitive dependencies on the primary key. After achieving 3rd normal form the table is normalized.

**Applications**:(ApplicationID,ApplicationDate,ApplicationStatus,StudentID)

PK: ApplicationID(INT)

FK: StudentID(INT) Student.StudentID(INT)

Alternate Keys: Application Date(VARCHAR), Application Status(VARCHAR)

The Applications table has records of the student’s ApplicationID, ApplicationDate, ApplicationStatus and StudentID. The primary key is the ApplicationID, has alternate keys of ApplicatioDate and ApplicationStatus with a foreign key that links to the Student.StudentID table. Since the table is uniquely identified by the primary key it satisfies the conditions of 1st normal form. Also, since there are no partial dependencies and a single primary key with no composite keys, the table satisfies 2nd normal form. In addition, 3rd normal form is satisfied because there aren’t any transitive dependencies on the primary key. After achieving 3rd normal form the table is normalized.

**Department Application**:(DepartmentApplicationID,ApplicationID,DepartmentID)

PK: DepartmentApplicationID(INT)

FK: ApplicationID(INT) Applications.ApplicationID(INT)

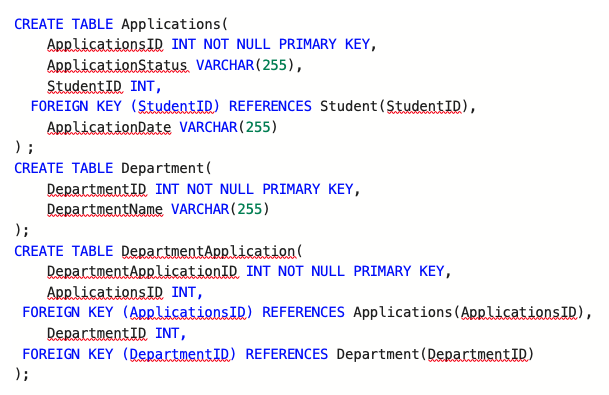
DepartmentID(INT) Department.DepartmentID(INT)

The DepartmentApplication table is an intermediary table between Departments and Application. To avoid the troubles of a many to many relationship and to achieve normalization the DepartmentApplication table was created with a primary key of DepartmentApplicationID, a foreign key of ApplicationID that links to the Applications.ApplicationsID table and a DepartmentID that links to a Department.DepartmentID table. Since the table is uniquely identified by the primary key it satisfies the conditions of 1st normal form. Also, since there are no partial dependencies and a single primary key with no composite keys, the table satisfies 2nd normal form. In addition, 3rd normal form is satisfied because there aren’t any transitive dependencies on the primary key. After achieving 3rd normal form the table is normalized.

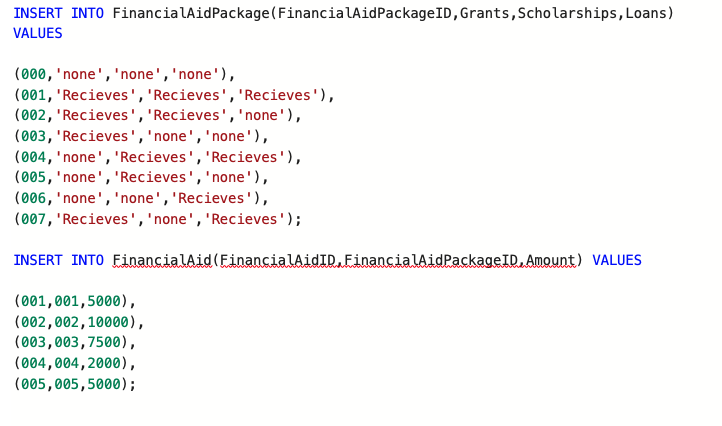
Database Construction:

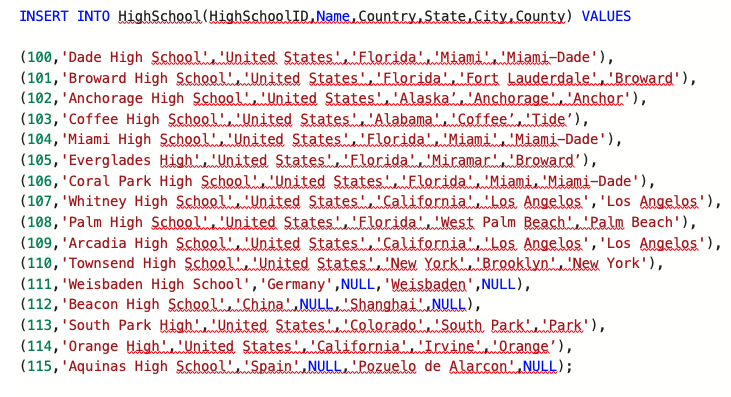
A screenshot of a cell phone

Description automatically generated

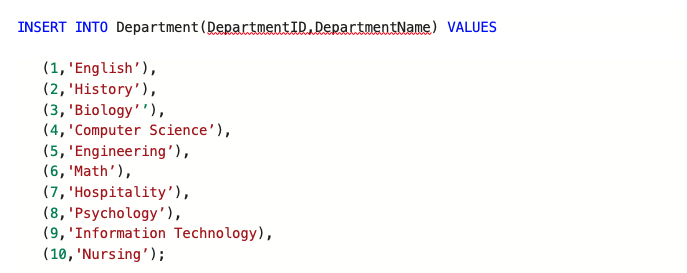


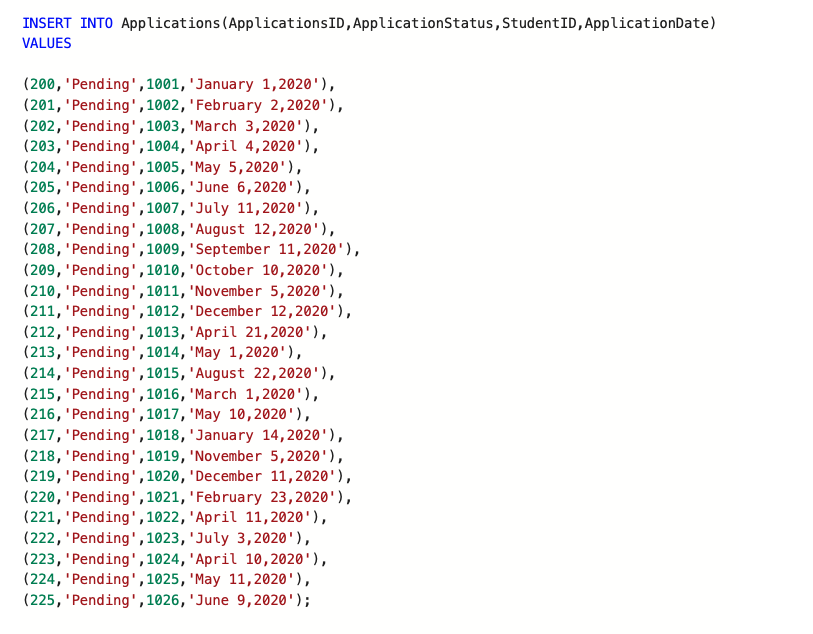
**INSERT DATA**











INSERT INTO DepartmentApplication(DepartmentApplicationID,ApplicationsID,DepartmentID) VALUES

(2000,200,1),

(2001,201,2),

(2002,202,3),

(2003,203,4),

(2004,204,5),

(2005,205,6),

(2006,206,7),

(2007,207,8),

(2008,208,9),

(2009,209,10),

(2010,210,1),

(2011,211,2),

(2012,212,3),

(2013,213,5),

(2014,214,5),

(2015,215,6),

(2016,216,7),

(2017,217,8),

(2018,218,9),

(2019,219,10),

(2020,220,1),

(2021,221,2),

(2022,222,3),

(2023,223,4),

(2024,224,5),

(2025,225,6);

**SELECT \* Statements for all tables**

**SELECT \* FROM FinancialPackage;**

**A picture containing table, sitting, black, pizza

Description automatically generated**

**SELECT \* FROM Financialaid;**

**A screenshot of a computer

Description automatically generated**

**SELECT \* FROM HighSchool;**

**A green and black text

Description automatically generated**

**SELECT \* FROM Student;**

**A picture containing sitting, table, green, holding

Description automatically generated**

**SELECT \* FROM Department;**

**A screenshot of a cell phone

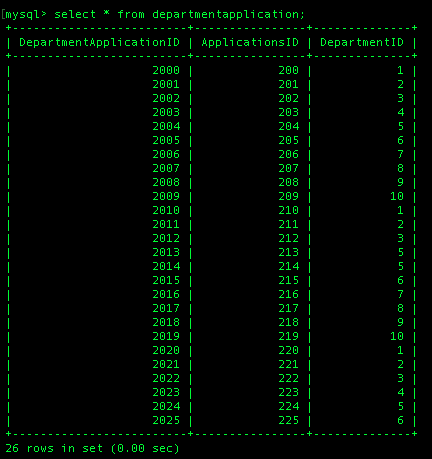
Description automatically generated**

**SELECT \* FROM Applications;**

**A picture containing computer

Description automatically generated**

**SELECT \* FROM DepartmentApplication;**

****

**QUERIES**

**Multi-Table Query:**

SELECT s.LastName,s.FirstName,h.Country,a.ApplicationDate

FROM Student s

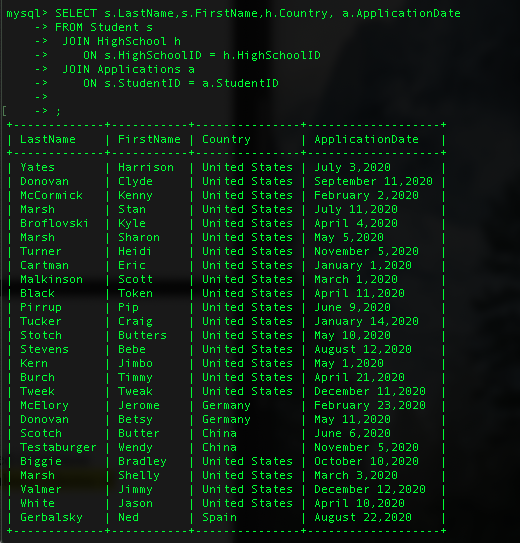
JOIN HighSchool h

ON s.HighSchoolID=h.HighSchoolID

JOIN Applications a

ON s.StudentID=a.StudentID

This query is a three way join that takes the names of all students, the country where they are from and their application date.



SELECT Lastname,Firstname,Ethnicity, Grants, Scholarships

FROM student s join FinancialAid FA

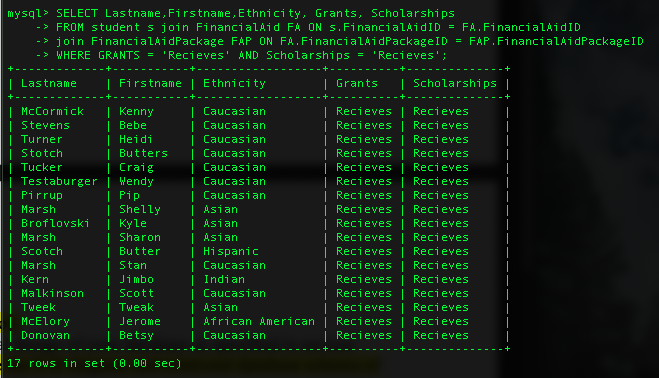
ON s.FinancialAidID = FA.FinancialAidID

JOIN FinancialAidPackage FAP

ON FA.FinancialAidPackageID = FAP.FinancialAidPackageID

WHERE Grants = 'Recieves' AND Scholarships = 'Recieves';

Takes the name and ethnicity of students who receive both grants and scholarships.



**AGGREGATE FUNCTION QUERIES**

SELECT city, MAX(unweightedGPA) as gpa

FROM Highschool h

JOIN student s

ON h.highschoolID = s.highschoolID

GROUP BY city

List cities and groups the max or top GPA in that city.

A screenshot of a cell phone

Description automatically generated

SELECT country, AVG(SATscore) as sat

FROM Highschool h

JOIN student s

ON h.highschoolID = s.highschoolID

GROUP BY country;

Takes the average SAT scores of students and groups them by country.

A screenshot of a cell phone

Description automatically generated

**NULL SEARCH CONDITION**

SELECT lastname,firstname,FinancialAidID

FROM Students WHERE FinancialAidID is NULL;

Searches students who don’t have a FinancialAidID and don’t receive financial aid.

A close up of a computer

Description automatically generated

**SUBQUERY STATEMENT**

SELECT lastname,firstname,weightedGPA FROM student as s

WHERE weightedGPA > (SELECT AVG(weightedGPA) FROM student);

This query gets the names of the students who have a greater weightedGPA than the overall average weightedGPA of all students. Average weightedGPA of all students was 3.846.

A picture containing black, holding, table, standing

Description automatically generated

**HAVING CLAUSE**

SELECT lastname,firstname,amount

FROM financialaid fa join student s

ON fa.financialaidID = s.financialaidID

HAVING amount > 5000;

This query gives the names of students who receive financial aid of more than $5000..

A picture containing sitting, black, table, pizza

Description automatically generated