Shane Ẽire Byrne

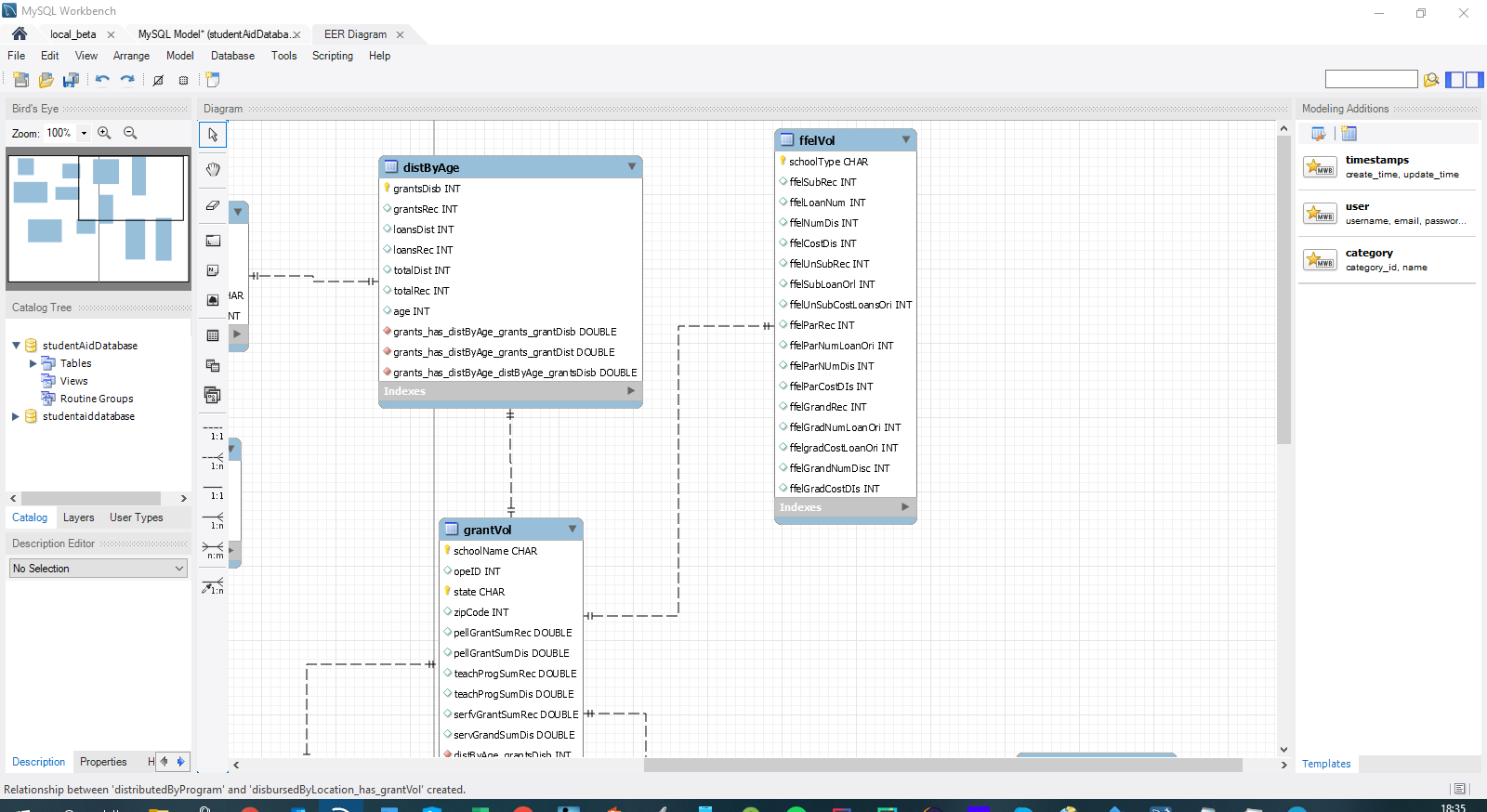
DAT390

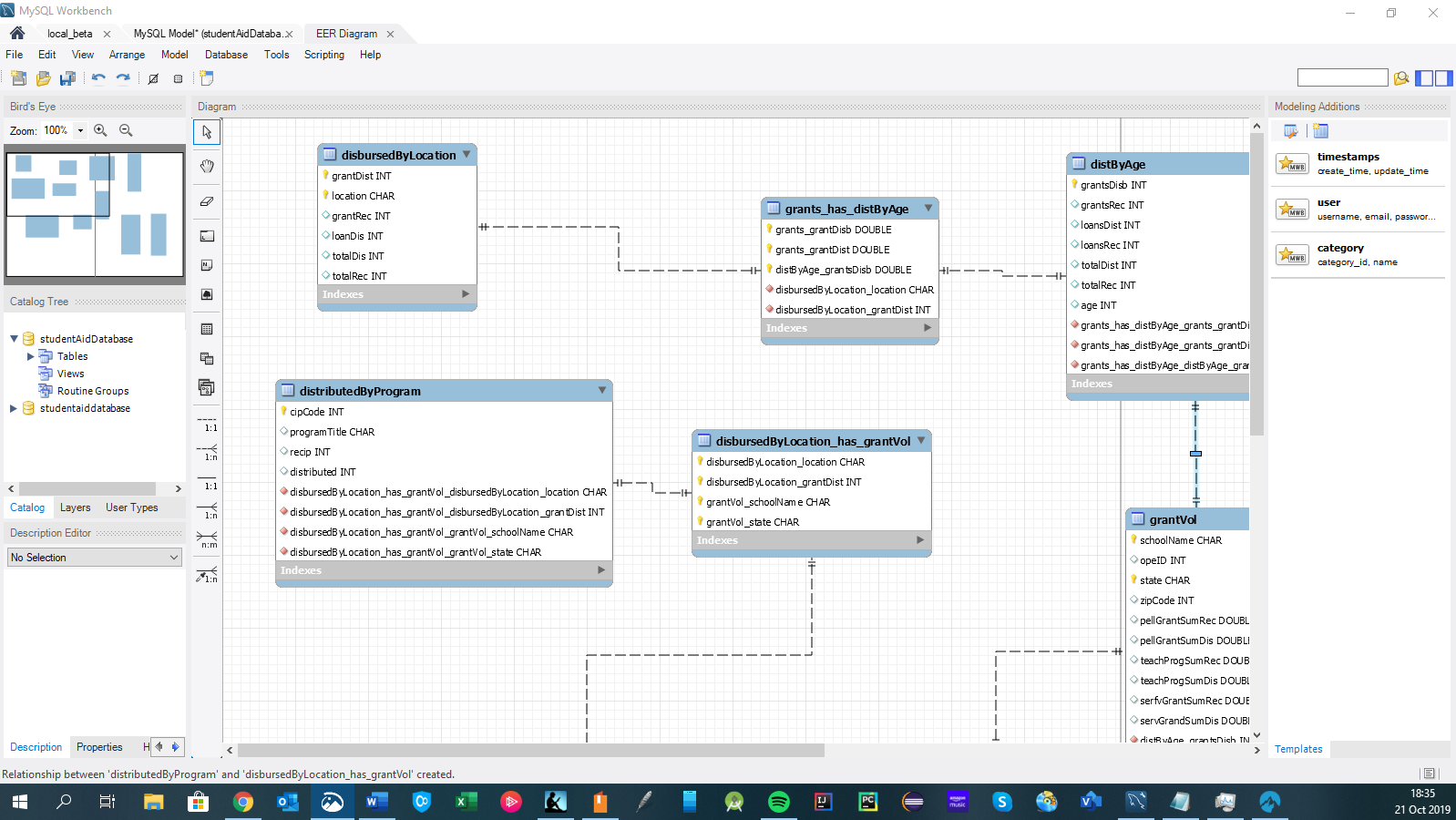
U.S. Student Aid Data Summary

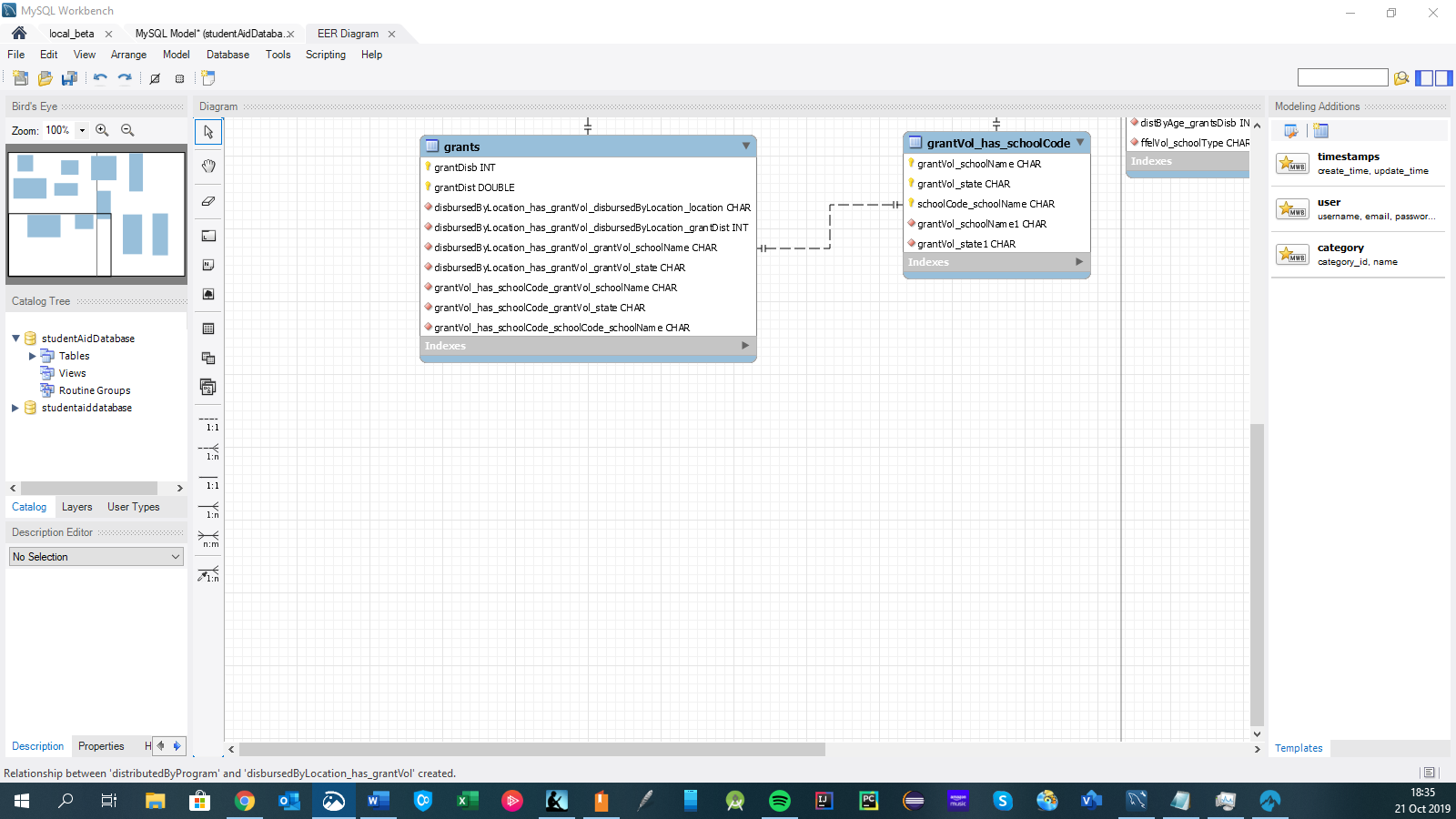
**Schema**

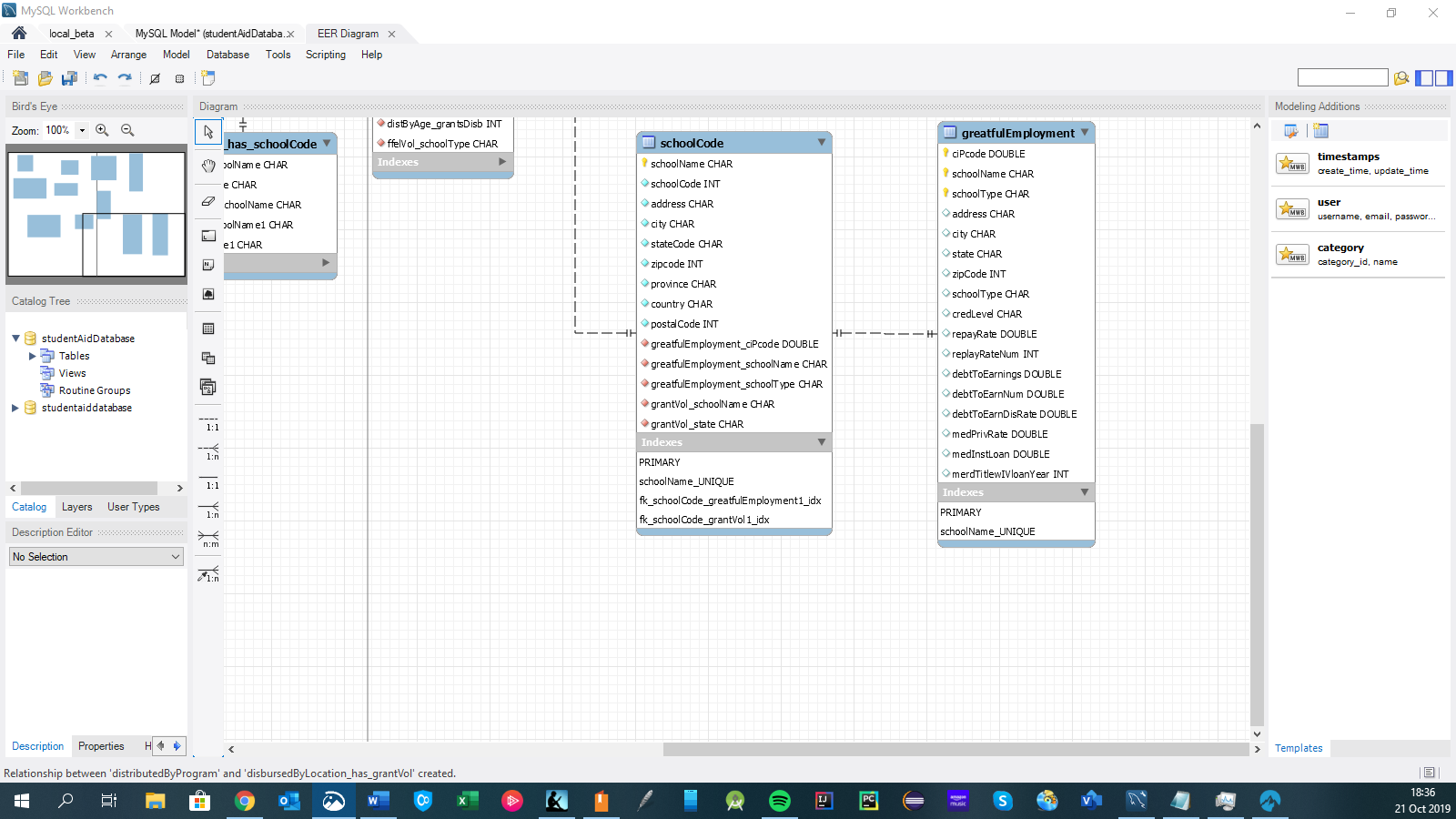
In a nutshell, a Schema being a representation of a plan or theory in the form of an outline or model. The underline schema developed for the U.S. Student Aid Data project being one that tied certain factors presented in the given spreadsheets together. The establishment of primary keys (this being a unique set of data values that can be cross referenced between spreadsheets within this context) and the establishment of foreign keys (these being columns that proved a line between data across two tables or spreadsheets within the context of this).

School name in and of itself functions as a primary key between tables. This in turn connects several of the spreadsheets alone as some of them do contain the schools name within its database. School code is utilized as a primary key to connect some of the tables as well as the use of opeID as some tables utilized the same opeID sequence to identify schools by a decimal number.









Initially I believed this was the best course for a schema for this database but did run into some difficulty in the MySQL language which in turn had issues with the data being streamed into the tables through the import methods MySQL provides. Most issues stemmed from a conversion issue in which the MySQL program executing the code would perceive the data being imported as the incorrect data type (exempli gratia an all to common error being the difference between a float and a double). Some of this issue was corrected with the conversion of data from its numerical values to its String equivalent; the issue presented with this in the long term being no calculations can be made to String data types as it is not seen as a numerical set of data. The perceived “fix” for this short coming being the conversion of a String data type to its numerical equivalent. In most programming languages this can be done by use of a conversion key word (exempli grata  the CAST() keyword which casts from one Data type (generally the initial data type of the given data) to another data type (the assigned data type one instructs the data to be cast to).

Like most languages I have written in, my main strategy was to comprise the code in its simplest form. The simplification of commands allows them to be easier to understand by nature. Part of this is accomplished by already tying primary keys (such as school name) together into corresponding tables. This approach of joining tables eliminates the need for code that would have to call and join each table anytime data would need to be achieved since the combination of data is already available within the established sub-table.

Best practices of MySQL used being to assign items variable names (exempli grata, table names can be set as variables of only a few characters, this establishes a visual reference for each value being used in a query allowing for less characters to be used in a query and making code less monotonous and easier for an individual to follow.

To switch between schemas, it is easier to keep them both on the same server as one can use the ALTER TABLE keywords. This changes the reference of one schema to another.

In order to allow the data to be streamed from the spreadsheets easier, the files were converted from Excel to csv format. This stripped away the formatting options available within an Excel spreadsheet and leaves just the core data itself. The conversion of the files in turn makes them easier to work with and load into tables.

In terms of the overall project overall, my main goal is cleaner MySQL script as the script I generated for this was trouble some in its own regard. While it was readable, I was not completely successful in getting all the bugs out and still have issues with importing data from tables, most of these issues through my own research are conversion issues between the data contained within the columns of each spreadsheet and the anticipated data within the code. A possible fix for this would be something akin to the generic data type available in Java, further research into MySQL would help with this and would help to famuliarise myself with the language a lot more. Practice with imploring genetics into MySQL columns could potentially allow the easy passage of data from spreadsheets as the code itself would look at the physical data type of what is being presented to it.

/\*DROP DATABASE IF EXISTS studentaiddatabase;\*/

CREATE DATABASE IF NOT EXISTS studentAidDatabase CHARACTER SET utf8mb4;

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema studentAidDatabase

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `studentAidDatabase` DEFAULT CHARACTER SET utf8mb4 ;

USE `studentAidDatabase` ;

-- -----------------------------------------------------

-- Table `studentAidDatabase`.`schoolCode`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `studentAidDatabase`.`schoolCode` (

`schoolName` CHAR NOT NULL,

`schoolCode` INT NOT NULL,

`address` CHAR NOT NULL,

`city` CHAR NOT NULL,

`stateCode` CHAR NOT NULL,

`zipcode` INT NOT NULL,

`province` CHAR NOT NULL,

`country` CHAR NOT NULL,

`postalCode` INT ,

PRIMARY KEY (`schoolName`),

UNIQUE INDEX `schoolName\_UNIQUE` (`schoolName` ASC) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `studentAidDatabase`.`disbursedByLocation`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `studentAidDatabase`.`disbursedByLocation` (

`grantDist` INT NOT NULL,

`location` CHAR NOT NULL,

`grantRec` INT NULL,

`loanDis` INT NULL,

`totalDis` INT NULL,

`totalRec` INT NULL,

PRIMARY KEY (`location`, `grantDist`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `studentAidDatabase`.`greatfulEmployment`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `studentAidDatabase`.`greatfulEmployment` (

`ciPcode` DOUBLE NOT NULL,

`schoolName` CHAR NOT NULL,

`address` CHAR NULL,

`city` CHAR NULL,

`state` CHAR NULL,

`zipCode` INT NULL,

`schoolType` CHAR NOT NULL,

`credLevel` CHAR NULL,

`repayRate` DOUBLE NULL,

`replayRateNum` INT NULL,

`debtToEarnings` DOUBLE NULL,

`debtToEarnNum` DOUBLE NULL,

`debtToEarnDisRate` DOUBLE NULL,

`medPrivRate` DOUBLE NULL,

`medInstLoan` DOUBLE NULL,

`merdTitlewIVloanYear` INT NULL,

PRIMARY KEY (`ciPcode`, `schoolName`, `schoolType`),

UNIQUE INDEX `schoolName\_UNIQUE` (`schoolName` ASC) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `studentAidDatabase`.`ffelVol`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `studentAidDatabase`.`ffelVol` (

`schoolType` CHAR NOT NULL,

`ffelSubRec` INT NULL,

`ffelLoanNum` INT NULL,

`ffelNumDis` INT NULL,

`ffelCostDis` INT NULL,

`ffelUnSubRec` INT NULL,

`ffelSubLoanOrl` INT NULL,

`ffelUnSubCostLoansOri` INT NULL,

`ffelParRec` INT NULL,

`ffelParNumLoanOri` INT NULL,

`ffelParNUmDis` INT NULL,

`ffelParCostDIs` INT NULL,

`ffelGrandRec` INT NULL,

`ffelGradNumLoanOri` INT NULL,

`ffelgradCostLoanOri` INT NULL,

`ffelGrandNumDisc` INT NULL,

`ffelGradCostDIs` INT NULL,

PRIMARY KEY (`schoolType`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `studentAidDatabase`.`distByAge`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `studentAidDatabase`.`distByAge` (

`grantsDisb` INT NOT NULL,

`grantsRec` INT NULL,

`loansDist` INT NULL,

`loansRec` INT NULL,

`totalDist` INT NULL,

`totalRec` INT NULL,

`age` INT NULL,

PRIMARY KEY (`grantsDisb`))

ENGINE = InnoDB

COMMENT = ' ';

-- -----------------------------------------------------

-- Table `studentAidDatabase`.`grantVol`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `studentAidDatabase`.`grantVol` (

`schoolName` CHAR NOT NULL,

`opeID` INT NULL,

`state` CHAR NOT NULL,

`zipCode` INT NULL,

`pellGrantSumRec` DOUBLE NULL,

`pellGrantSumDis` DOUBLE NULL,

`teachProgSumRec` DOUBLE NULL,

`teachProgSumDis` DOUBLE NULL,

`serfvGrantSumRec` DOUBLE NULL,

`servGrandSumDis` DOUBLE NULL,

PRIMARY KEY (`schoolName`, `state`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `studentAidDatabase`.`distributedByProgram`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `studentAidDatabase`.`distributedByProgram` (

`cipCode` DOUBLE NOT NULL,

`programTitle` CHAR NULL,

`recip` INT NULL,

`distributed` INT NULL,

PRIMARY KEY (`cipCode`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `studentAidDatabase`.`distByAge`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `studentAidDatabase`.`distByAge` (

`grantsDisb` INT NOT NULL,

`grantsRec` INT NULL,

`loansDist` INT NULL,

`loansRec` INT NULL,

`totalDist` INT NULL,

`totalRec` INT NULL,

`age` INT NULL,

`grants\_has\_distByAge\_grants\_grantDisb` DOUBLE NOT NULL,

`grants\_has\_distByAge\_grants\_grantDist` DOUBLE NOT NULL,

`grants\_has\_distByAge\_distByAge\_grantsDisb` DOUBLE NOT NULL,

PRIMARY KEY (`grantsDisb`),

INDEX `fk\_distByAge\_grants\_has\_distByAge1\_idx` (`grants\_has\_distByAge\_grants\_grantDisb` ASC, `grants\_has\_distByAge\_grants\_grantDist` ASC, `grants\_has\_distByAge\_distByAge\_grantsDisb` ASC) ,

CONSTRAINT `fk\_distByAge\_grants\_has\_distByAge1`

FOREIGN KEY (`grants\_has\_distByAge\_grants\_grantDisb` , `grants\_has\_distByAge\_grants\_grantDist` , `grants\_has\_distByAge\_distByAge\_grantsDisb`)

REFERENCES `studentAidDatabase`.`grants\_has\_distByAge` (`grants\_grantDisb` , `grants\_grantDist` , `distByAge\_grantsDisb`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

COMMENT = ' ';

-- -----------------------------------------------------

-- Table `studentAidDatabase`.`grants`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `studentAidDatabase`.`grants` (

`grantDisb` INT NOT NULL,

`grantDist` DOUBLE NOT NULL,

PRIMARY KEY (`grantDisb`, `grantDist`))

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

SHOW VARIABLES LIKE 'securefilepriv';

LOAD DATA INFILE 'AY18Disbursements-by-Program.csv'

INTO TABLE `studentAidDatabase`.`distributedByProgram`

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE "grants-ay18-19-q3.xls" INTO TABLE `studentAidDatabase`.`grantVol`

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE "GE2011InformationalRates102512\_0.XLS" INTO TABLE `studentAidDatabase`.`gainfulEmployment`

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE "FL\_Dashboard\_AY2009\_2010\_Q4.xls" INTO TABLE `studentAidDatabase`.`ffelVol`

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE "dl-dashboard-ay2018-2019-q3.xls" INTO TABLE `studentAidDatabase`.`dlVolschool`

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE "AY18Disbursements-by-Program.xls" INTO TABLE `studentAidDatabase`.`distributedByProgram`

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE "AY18Disbursements-by-Location.xls" INTO TABLE `studentAidDatabase`.`disbursedByLocation`

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE "AY18Disbursements-by-Age.xls" INTO TABLE `studentAidDatabase`.`distByAge`

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE "1920FedSchoolCodeList.xlsx" INTO TABLE `studentAidDatabase`.`schoolCode`

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

/\*Week 3 inquiries\*/

SELECT \* FROM `studentAidDatabase`.`schoolCode\_has\_greatfulEmployment` AS t1

CROSS JOIN `studentAidDatabase`.`greatfulEmployment\_has\_ffelVol` AS t2

WHERE fk\_greatfulEmployment\_has\_grantVol\_greatfulEmployment1\_idx > 1000;

SELECT \* FROM `studentAidDatabase`.`grantVol\_has\_schoolCode` AS t3;

SELECT `studentAidDatabase`.`distByAge` AS t4;

SELECT AVG(t4.`loansDist`);

