**CPSC 50900 Database Systems Project**

All your efforts this semester will focus on this project to be done BY YOURSELF for which you will develop, document, implement, secure, and program with a database. You will post the **artifacts of work** (XML files, model diagrams, code) as a **GitHub repository**, and you will add to it steadily throughout the semester. You will document all your efforts in a single **Word document** that you'll write throughout the semester, sharing it with me periodically to review and provide comment.  
  
**To summarize: Create a report using Word, and store all of your writing there. For technical documents, post those to a GitHub account and include the link in your report at the end of your proposal.**   
  
The report must have a title page, and the total is worth 10 points for clarity and organization.  
Organizing your files in GitHub by type with clear and unique file names is worth another 10 points.

Your project will consist of the following sections:

**Initial Proposal**

In your report document:

You will describe the data you aim to store. What is your application or business? What data will be storing? What is the value of this data? Where will the data come from? Who will use this data?

Rubric: Your response to each of these five questions will be graded out of 3 points.

* 3 points: clear, complete descriptions that convey the importance and meaning of your data
* 2 points: mostly clear descriptions, although some additional data would have helped in some sections – If it sounds like you are writing without saying anything (“fluff”), then you are losing a point.
* 1 point: necessary details are lacking in many of your responses.

Oscars are the most respectable and highly valued awards in the film fraternity and in the cinema lover community. I will be storing the information of all the people who won or got nominated for the Oscars in all categories. usually, there are five to 6 nominations and only one of them wins.

As a movie lover i watch and appreciate a lot of movies and i like many other movie lovers want to know about the technicians and artists involved. Some of them are exceptional and needs to be appreciated and awarded for their service in the film and for the society. The list of people nominated for the Oscar assures best performance and people look for their names in a movie or get interested in a movie solely because of the involvements of those guys.

The people who gets an Oscar or gets nominated for the Oscars are the people who are at their best in their respective fields and they have every right to be known and discovered by many others for the exceptional service they provide. People also will have an idea on whom to checkout for not to be disappointed.

The data is found on the official websites ofoscars and in various other websites created by film enthusiasts. The information is widely spread, and it takes time to collect all the information from the year of its formation.

The people who are film enthusiasts and who are generally into watching movies can use this data to explore the work of great artists. This data helps them to connect the people with the art undiscovered by them and the art finds the people for which it is intended to.

I want to create an application which curates the movies for the people to watch the best content and suggests the viewers the best. This allows the subscribers to watch the art which they are seeking and they become dependent on the application to curate the list of movies for them.

Files are located at:

Your report might look similar, but please don’t copy what I wrote.

Total points possible: 15

**Relational Database Design Process**

Identify at least 5 entity sets and their attributes (at least 3 attributes per entity), and describe these entities in your **report**. This will look a lot like the description component of the Aquarium assignment. Then illustrate that you understand normalization and how to use it to reduce uncontrolled redundancy in your database design by creating a single ERD diagram using either UML or Crow’s Foot notation and upload that drawing to **GitHub**. The ERD must be a **physical** model (includes data types and key information).

Rubric: Your work will be graded as follows:

* 4 points for describing each of 5 entities and their attributes in your report
* 10 points for the physical model in GitHub

Total points possible: 25

Here the functional dependencies are actor id->actor name, film id->film name, movie id->movie name ,and the determinants include actor id, actress id, movie id, Oscar held year, film id.

Different entity sets are Actor, Award winning movies, Oscar held, films Nominated, .

Actor-filmcast : one-to-many

Film\_cast-Film : one-to-one

Fillms-Films nominated : : one-to-many

Oscar held-Films nominate : one-to-many

Foreign keys are actor id, actress id,Diagram

Description automatically generated

**Data Sources**

Create examples (**three** records per entity) of all of your data in a parseable format such as XML, JSON, or delimited like comma separated values and submit the files to **GitHub**. In your **report**, explain the process of how your software stores each entity and attribute, what is the source of the attributes, and what data is default.  You don’t have to describe every attribute, but I want to know at least two details about each entity, and how that data enters the system.  
  
My example would include something like “As the business owner I would manually input my employee data (entity) which includes name, phone number, and address (specific details and attributes) into the database. The database updates the employee ID automatically (a default condition that defines the remaining attribute of my entity). However, I use an arglebargle software tool (and if I see arglebargle in your report, you are losing a lot of points!) to read product information (another entity) and format the part number, description, cost, and type (all the attributes of the product entity) into an XML data structure. All of these attributes come from the supplier. ” Notice, I was specific with these attributes and determined which were default, which were created, and where they all came from.

Rubric: Your work will be graded as follows:

* 5 points per populated file that has attributes which **match your design.** If you are missing one or two, then you lose a couple points. If you miss a lot, you may lose half or all the points.
* 5 points per entity: you described the contents of the data files in detail, including referencing their origin and explaining how they were structured.

Total points possible: 50

*Description of the datafiles:*

* *Details: The data collected for the database file contains all of nomination on the Academy Awards Oscar. The information includes all the data regarding the Oscar nominations and details of the awardee.*
* Origin: The data was collect from the (Kaggle.com). The data was initially in the form of the csv file later it was filtered into Json and xml. The raw data is processed and removed some unwanted columns.
* Structure : The files are structured into fields ,subfields , categories and later divided according to the year of nomination

Additional files to be included: I would like to include the field named “previously nominated “and “no of Oscars “ in the datafiles.

* I will sort out the data into various tables which will be easy to access and edit data.
* I will filter all fields and replace winner column with no of awards section
* The data was manually separated in the excel and separated according to their respective fields.

**Data Definition Language Scripts**

OLD INSTRUCTIONS:

First, use Vertabello to generate a script of SQL commands that build the database and its table structures. Or, you can write scripts or build Excel spreadsheets that take your data files and generate scripts of SQL insert statements from them. Use the MySQL *source* command to run the various scripts needed to build and populate the database in MySQL. Include the source code and / or Excel spreadsheets you use to manipulate and populate the data. Make sure all your tables have at least three records in them and that you've linked the tables through their foreign keys.

UPDATED INSTRUCTIONS:

To simplify the instructions, first you need a script that creates the table structures. Then, you need the ‘load’ command to import the data you saved to an XML or CSV file. You might have also used JSON or another format, but you’ll have to research that method yourself.

Assign columns as a **Primary key** or **Foreign key** (where needed) so that if you modify a row in one table, the changes update in the other tables. Obviously, do not assign columns as keys if they are not shared across tables.

In both of the following sections, I offer you CHOICES (by number)– not steps to complete. Pick your favorite ONE, and execute that choice.

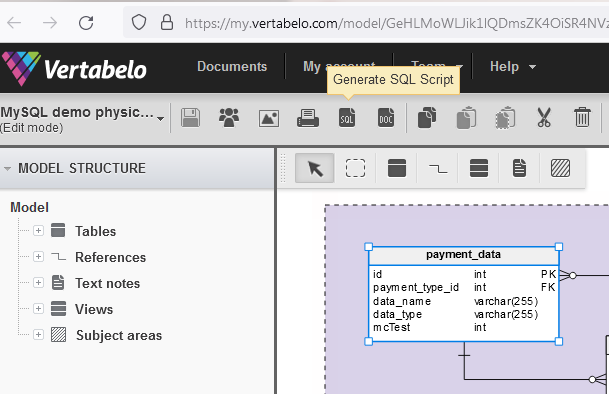
TO CREATE THE TABLE:

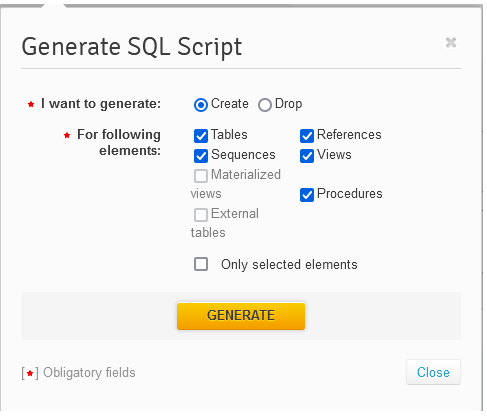
1. Manually write a script with CREATE and VALUE statements. You can either write this at the prompt (can be tedious if you make errors), copy/paste a script into the prompt, or save your script as a .sql file and use ‘SOURCE’ to load your script like you did when you loaded the nation database for the SQL#2 assignment.

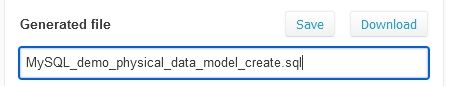
You may feel this option is easier/faster than using Vertebelo. However, if you drew your model in Vertebelo, then the script is just a 4 step point-and-click process.

2. Generate the script using a tool like Vertebelo. Here is the procedure:

a) Create the tables/database:



b) Using the SQL icon above the drawing area, choose to Generate SQL Script: 

c) Click on ‘Generate’ to export the SQL file:  


d) download, and copy the .sql file to your favorite directory where you administrate MariaDB. This is where you use the ‘SOURCE’ command to load the sql file like you did with the nation db example.

TO IMPORT DATA INTO MariaDB:

1. Use “LOAD DATA LOCAL INFILE ‘filename’ “ to load a delimited Excel file (usually a comma separated value, CSV).

<https://mariadb.com/kb/en/load-data-infile/>

You’ll notice the ‘set’ at the end gives you the option to manipulate data during the load. In fact, nearly all of those commands are optional since they are shown within square brackets, [ ]. Thus, this step is very simple! You may also redefine the delimiter character if you are not using a CSV.

2. If you are using XML files to store data, then see this example:

<https://mariadb.com/kb/en/load-xml/>

Again, a single command will load the contents of your file.

Rubric: Your work will be graded as follows:

* Database and table creation statements (manual or Vertebelo) saved as an sql script file. Upload to GitHub. 8 points
* The SQL commands for populating the tables **and** what they do (explain each of the commands and options that you used in the script) documented in your report: 8 points
* Screenshots of your successful attempts to populate each table with at least three records: 4 points

Total points possible: 20

the scripts contain records of various tables.

The actor tables insert statements, and all the remaining tables are created and populated using the insert statements.

The scripts contain insert and create statements which are used in creating and inserting records into the tables created in the database.

-- tables

-- Table: Actor

CREATE TABLE Actor (

Actor\_ID int NOT NULL,

Actor\_fname varchar(15) NOT NULL,

Actor\_lname varchar(10) NOT NULL,

Actor\_age int NOT NULL,

Actor\_Gender varchar(5) NOT NULL,

CONSTRAINT Actor\_pk PRIMARY KEY (Actor\_ID)

);

-- Table: Film

CREATE TABLE Film (

Film\_ID int NOT NULL,

Release\_year int NOT NULL,

CONSTRAINT Film\_pk PRIMARY KEY (Film\_ID)

);

-- Table: Film\_cast

CREATE TABLE Film\_cast (

Filmcast\_ID int NOT NULL,

Actor\_ID int NOT NULL,

Film\_ID int NOT NULL,

CONSTRAINT Film\_cast\_pk PRIMARY KEY (Filmcast\_ID)

);

-- Table: Films\_nominated

CREATE TABLE Films\_nominated (

nom\_id int NOT NULL,

Oh\_year int NOT NULL,

Film\_ID int NOT NULL,

CONSTRAINT Films\_nominated\_pk PRIMARY KEY (nom\_id)

);

-- Table: Oscar\_held

CREATE TABLE Oscar\_held (

Oh\_year int NOT NULL,

Awards\_given int NOT NULL,

People\_attended int NOT NULL,

Oh\_location varchar(20) NOT NULL,

CONSTRAINT Oscar\_held\_pk PRIMARY KEY (Oh\_year)

);

-- foreign keys

-- Reference: Film\_cast\_Actor (table: Film\_cast)

ALTER TABLE Film\_cast ADD CONSTRAINT Film\_cast\_Actor FOREIGN KEY Film\_cast\_Actor (Actor\_ID)

REFERENCES Actor (Actor\_ID);

-- Reference: Film\_cast\_Film (table: Film\_cast)

ALTER TABLE Film\_cast ADD CONSTRAINT Film\_cast\_Film FOREIGN KEY Film\_cast\_Film (Film\_ID)

REFERENCES Film (Film\_ID);

-- Reference: Films\_nominated\_Film (table: Films\_nominated)

ALTER TABLE Films\_nominated ADD CONSTRAINT Films\_nominated\_Film FOREIGN KEY Films\_nominated\_Film (Film\_ID)

REFERENCES Film (Film\_ID);

-- Reference: Films\_nominated\_Oscar\_held (table: Films\_nominated)

ALTER TABLE Films\_nominated ADD CONSTRAINT Films\_nominated\_Oscar\_held FOREIGN KEY Films\_nominated\_Oscar\_held (Oh\_year)

REFERENCES Oscar\_held (Oh\_year);

insert into Actor values(125,'Mahesh','Babu',45,'M');

insert into Actor values(126,'Leonardo','DeCaprio',56,'M');

insert into Actor values(127,'Tony','Stark',50,'M');

insert into Actor values(100,'heli','margarate',39,'F');

insert into Actor values(101,'Kate','Winslet',30,'F');

insert into Actor values(102,'molly','james',25,'F');

insert into Film values(1,'titanic',2000);

insert into Film values(2,'return of the apes',2005);

insert into Film values(3,'beast',2002);

insert into Film values(4,'The Jungle Book',2010);

insert into film\_cast values(120,126,1);

insert into film\_cast values(122,102,3);

insert into film\_cast values(123,100,2);

insert into film\_cast values(124,125,4);

insert into Oscar\_held values(2001,30,5000,'sydney');

insert into Oscar\_held values(2004,30,4500,'Hollywood');

insert into Oscar\_held values(2006,30,6000,'paris');

insert into Films\_nominated values(12,2004,1);

insert into Films\_nominated values(13,2006,3);

insert into Films\_nominated values(14,2001,2);

Text

Description automatically generated

**Data Manipulation Language Scripts**

Write the SQL commands for twelve queries. Two queries should be insert statements, two should update statements, one should be a delete statement, one should be a simple select statement that selects a subset of the rows and columns from one table, two should be a select statements that select data from a joining of two tables, two should use summary functions to generate statistics about the data, one should be a multi-table query, and one should be another query of your choice. Show the queries and screenshots of the results in your Word document **report**, and save your queries in a commented sql script to GitHub. Points for the SQL statements will be scored in the scripts on GitHub, points for the screenshots will be scored in your report.

Rubric: Your work will be graded as follows:

* 1 point each for the two insert statements
* 1 point each for the two update statements
* 1 point for the delete statement
* 1 point for the simple select statement
* 2 points each for the 2 join statements
* 2 points each for the two that use summary statements
* 2 points for the multi-table query
* 2 points for the query of your choice.
* 12 points for showing the query and a screenshot of the corresponding result set back-to-back for each of these queries in your Word document.

Total points possible: 30

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text, chat or text message

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

**Indexes**

Improve the performance of your design by adding indexes to various tables. Show the SQL needed to add the indexes. Explain why you chose the ones you added. Explain how you would demonstrate the impact the indexes had on the performance of various queries. (what tests could you run? Remember the slow query log?) . You can put all of this information in your **report.**

Rubric: Your work will be graded as follows:

* 6 points for clearly defining at least three indexes and explaining why you chose them.
* 3 points for showing the sql needed to generate the indexes
* 3 points for explaining how you would demonstrate the performance improvement afforded by the indexes.

Total points possible: 12

I’ve chosen my three indexes on actor first name,Film id and Oscar held year since I think that these will be the most searched in the database.

Everyone is interested in knowing the actor name who has done execellent acting hence placing an index on it will be much efficient to search them. Likewise the various Films attached to their Film id helps in faster fetching. Oscar held helps more efficient searches on the diversely held Oscar ceremonies.

**Text

Description automatically generated**

**Views**

Add two views to your database to provide easy access to combinations of data from multiple tables. Document the work and screenshots in your **report**.

Rubric: Your work will be graded as follows:

* 2 points for including the SQL for generating the two views in your Word document
* 2 points for including screenshots for the data contained in each view in your **Word document**
* 2 points for explaining why each view is a valuable addition to your database
* 2 points for explaining who might benefit most from having access to each view.

Total points possible: 8

create view Oscar\_movies as

select f.film\_name, a.actor\_fname, f.Oscar\_held\_oh\_year

from films f, Actor a, Oscar\_held o, film\_cast fc

where f.movie\_id=fc.film\_id and a.act\_id=fc.Actor\_act\_id and at.Actress\_id=fc.Actress\_Actress\_id;

create view [Oscar\_movies\_premiere] as

select awm.film\_name, fn.premiere\_yr

from Film awm, Films\_nominated fn

where awm.movie\_id=fn.movie\_id;

I’ve created views on award winning movie’s cast and its premiere dates

These views are useful for viewing the films and their premiere dates. While the 1st view is for viewing the various films that are awarded and their cast.

**Triggers**

Add a trigger to a table so that data will be updated when a certain event occurs. Document your triggers and screenshots in your **report**.

Rubric: Your work will be graded as follows:

* 2 points for including the SQL for the trigger in your Word document
* 2 points for clearly explaining the purpose of the trigger
* 2 points for a screenshot and explanation that shows the trigger in action.

Total points possible: 6

delimiter //

create trigger del\_rec after delete

on Film

for each row

select old.Film\_name+" deleted!" as output;

delimiter ;

this trigger is for informing whether the selected record is deleted or not.

**Transactions**

Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior? Document all of this in your **report**.

Rubric: Your work will be graded as follows:

* 3 points for clearly explaining the importance of transactions to ensuring ACID behavior
* 3 points for including a screenshot and accompanying explanation of a MySQL transaction.

Total points possible: 6

Transactions are very important in operating or accessing the database because by default a lock will be placed on the working table whenever we start a transaction. This helps in maintain the acid properties which are

* Atomicity – there are no sequential changes. All are done in a single step
* Consistency – the data doesn’t changes before and after a transaction.
* Isolation – every transactions occurring is isolated. Hence no interruptions will take place.
* Durability – after the transaction the data is durable.

Text

Description automatically generated

In this transaction, we are updating the a film record in the film table.

**Security**

Identify the different kinds of users who will use your database. Write GRANT statements to define the privileges for these different kinds of users. Document all of this in your **report.**

Rubric: Your work will be graded as follows:

* 6 points for clearly identifying and describing the various kinds of users who will use the databases and identifying and justifying what privileges each should have.
* 4 points for writing GRANT statements that assign privileges to these different kinds of users.
* 4 points for demonstrating with screenshots that your GRANT statements do distinguish among different kinds of users in regard to what they can do with the database.

Total points possible: 14

The various users are

Oscar administrator, who manages the whole oscarInfo database and sees to the proper working of it.

Grant all on OscarInfo.\* to ‘admin@oscar’@’localhost’;

Reader, who proofreads the total database and points to any data errors.

Grant select on OscarInfo.\* to ‘reader@oscar’@’localhost’;

Security manager, who sees to the security of the all databases.

Grant all on \*.\* to ‘security@oscar’@’localhost’;

Text

Description automatically generated

**Locking**

Explain the purpose of locking tables and show how to do that to prevent inconsistencies that may arise in your data when concurrent transactions take place. Document all of this in your **report.**

Rubric: Your work will be graded as follows:

* 3 points for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.
* 3 points for providing a screenshot and accompanying explanation of locking tables.

Total points possible: 6

Suppose there are two delete actions that are being executed on a particular table, but the table contains only one record. Hence, in this case an error occurs which is there will be no more records to delete after the other delete action executes.

Text

Description automatically generated

Here the actor table and film\_cast table are locked and open for only reading. Hence whenever we try to insert into these tables it shows an error

**Backup**

How you will back up your database. What commands will you issue? How frequently will the commands run? How can they be automated? Where will the backups be stored? Document all of this in your **report.**

Rubric: Your work will be graded as follows:

* 12 points for clearly explaining and justifying your database backup strategy, including the frequency with which you will back up the database, how you will automate backups, where you will store them, and how you will secure them. You will earn three points for addressing each factor (frequency, location, automation, and security)
* 3 points for providing a screenshot of the command you would issue to back up the database.

Total points possible: 15

I’ve would back up my database once for every week and to a location different than where the actual database resides. Because if the disk containing the actual database crashes, this will wont crash.

I would automate the back up process by writing a .bat script which is then scheduled to run for every 7 days and I would secure the database using the encryption algorithms.

A picture containing text

Description automatically generated

**Python Programming**

Write a Python program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document **report**, and also post the program to your **GitHub** repository.  
  
The report is just a SELECT \* command that displays the contents of each table. If you have many records, you might want to look into the LIMIT option to restrict your output. Since the previous parts of this assignment specify 3 records, your output should be anywhere from 3 to 10 records per table.

Rubric: Your work will be graded as follows:

* 12 points for writing a Python script (and including its code in the Word doc) that will pull data from a database and store it to a text file and present it to the screen. Your code (the .py file) must have comments in it that explain how it works. You will be awarded 3 points for successfully connecting to the database, 3 points for successfully querying it, and 4 points for presenting the data to the screen and to a file. Internal comments count for 2 points.
* 2 points for posting the code to GitHub
* 4 points for showing a screenshot of your running the script and showing the results it produces on the screen.

Total points possible: 18

**--**

import mysql.connector  
from mysql.connector import errorcode  
from mysql.connector.errors import DataError  
  
*# Method to get the records from the database to store in a file and to present it.*def WriteRows(conn,Query,file):  
 cur = conn.cursor()  
 cur.execute(Query)  
 q = Query.split(' ')  
 file.write(q[3]+" Table:\n")  
 print(q[3]+" Table:")  
 for rec in cur.fetchall():  
 line=""  
 for r in rec:  
 line+=str(r)+","  
 file.write(line+"\n")  
 print(line)  
 file.write("\n")  
 print("\n")  
  
try:  
 *# Connecting to Database* conn = mysql.connector.connect(  
 user = "root",  
 password = "maria@123",  
 host = "localhost",  
 database = "OscarInfo"  
 )  
 print("database successfully connected!")  
  
 *# Opening a file to write* file = open('Dataset.txt','w')  
 print("File successfully opened!")  
  
 *# Querying the Tables* WriteRows(conn,"select \* from Actor",file)  
 WriteRows(conn,"select \* from Film",file)  
 WriteRows(conn,"select \* from Film\_cast",file)  
 WriteRows(conn,"select \* from Films\_nominated",file)  
 WriteRows(conn,"select \* from Oscar\_held",file)  
 print("Dataset created into the file successfully!")  
   
except mysql.connector.Error as err:  
 print("Error connecting to database")  
 exit()  
except IOError:  
 print("File not found or path incorrect")  
 exit()  
finally:  
 conn.close()

**output:**

**A picture containing background pattern

Description automatically generated**

**BONUS POINTS:**

If you complete all of the other sections, then you may try this section for bonus points. Feel free to do the PHP or the “Suggested Future Work”, or both!  
 **PHP Programming**

Build an HTML form that enables the user to specify criteria to search by. Use PHP to show the results of the query on a resulting web page. Make sure you include protections against an SQL injection attack. Include your HTML and PHP code in your Word document, and also post the files to your GitHub repository.

Rubric: Your work will be graded as follows:

* 4 points for writing an HTML form the user will use to enter search criteria
* 8 points for a PHP script that uses the search criteria and returns results
* 4 points for an HTML page that shows the results
* 4 points for explaining what SQL injection might be run on your website and explaining how you prevented it.
* 4 points for providing screen shots of your PHP website in action.
* 2 points for posting your code to GitHub

Total points possible: 26

**Suggested Future Work**

Describe the limitations of your current database and explain how you or someone else could improve the design to address these shortcomings. Also describe how you might take advantage of leverage cloud services to increase the performance and availability of your database. Finally, explain the advantages and disadvantages of storing your data in a NoSQL format instead.  
  
This requires researching what NoSQL is, what makes it different from MySQL, explaining how the 2 systems are different, and providing APA citations of your sources. You don’t need to create a separate bibliography or references section, but do provide the formatted source at the bottom of the document.

Tip: Purdue has an automatic citation generator page that is incredibly useful for this function.

Rubric: Your work will be graded as follows:

* 3 points for clearly describing the limitations of your database design
* 3 points for explaining how you would address these shortcomings
* 3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.
* 3 points for explaining the advantages and disadvantages of storing your data in a document-based NoSQL format instead.
* 2 points for the source references.

Total points possible: 14

Grand Total:

245 + 40 bonus