**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.

We've put in place a system that uses data on the prices of agricultural commodities in markets across India to provide some useful insights into some areas. As a result of this system's functions, users will be able to gain valuable insights that will affect their decisions throughout the entire process of growing crops to selling them.

1. Price patterns for various commodities throughout these many geographies and markets, so that the user can infer trends and use speculative projections from these trends when deciding whether to cultivate specific crops, sell them in certain markets, and so on.

2. Optimal locations for the user to sell the commodity they've grown based on price trends and distance-based costs, so the user can make the most money from the product they've developed. As a result of the system's features, users will be able to gain valuable information that will affect their actions throughout the full process of growing crops to selling them.

Files are located at:  
[www.github/mattsRobots](http://www.github/mattsRobots) (not a real address)

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

**1.1. Table STATE**

**1.1.1. Columns**

| **Column name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| State\_ID | int | PK |  |
| State\_name | varchar(25) |  |  |
| State\_zone | varchar(25) |  |  |

**1.2. Table District**

**1.2.1. Columns**

| **Column name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| District\_ID | int | PK |  |
| Distrrict\_name | varchar(25) |  |  |
| District\_headquarters | varchar(25) |  |  |
| STATE\_State\_ID | int |  |  |

**1.3. Table Market**

**1.3.1. Columns**

| **Column name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| Market\_ID | int | PK |  |
| Market\_name | varchar(25) |  |  |
| Market\_location | varchar(25) |  |  |
| District\_District\_ID | int |  |  |

**1.4. Table Commodity**

**1.4.1. Columns**

| **Column name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| Commodity\_ID | int | PK |  |
| Commodity\_name | varchar(25) |  |  |
| Commodity\_Variety | varchar(25) |  |  |

**1.5. Table Prices**

**1.5.1. Columns**

| **Column name** | **Type** | **Properties** | **Description** |
| --- | --- | --- | --- |
| Price\_ID | int | PK |  |
| Date | date |  |  |
| Price\_min | int |  |  |
| Price\_max | int |  |  |
| Market\_Market\_ID | int |  |  |
| Commodity\_Commodity\_ID | int |  |  |

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.

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.

From data.gov.in, we obtained information about agricultural market locations, prices, and so forth. We coupled that with geographic data from https://www.kaggle.com/sirpunch/indian-census-data-with-geospatial-indexing matching to the longitude and latitude of these various districts.

• The data that we acquired was not prepared, and several files had to be collected and processed for usage, thus we used the links given above to use readymade data

Some xml files were converted to csv format.

> Because the data was in the form of a universal relation, we had to process it for simplicity of use and to adhere to acceptable schema design guidelines.

> We processed over 30 files as a result of the preceding phase in order to convert information into the format we required.

> There were also some additional flaws, such as small errors in the names of 'Districts' in different files. As a result, we were able to discover many common and most typical scenarios while processing and incorporated processes for auto rectification during the pre-processing period

* Size of the raw dataset (before clean-up): 201MB
* Size of the cleaned dataset: 100 MB

.

Total points possible: 50

**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.

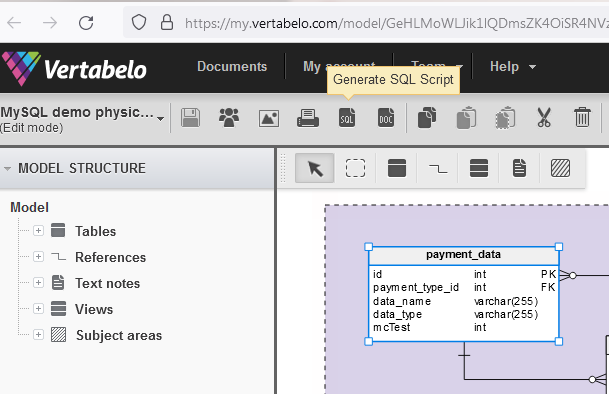
TO CREATE THE TABLE:

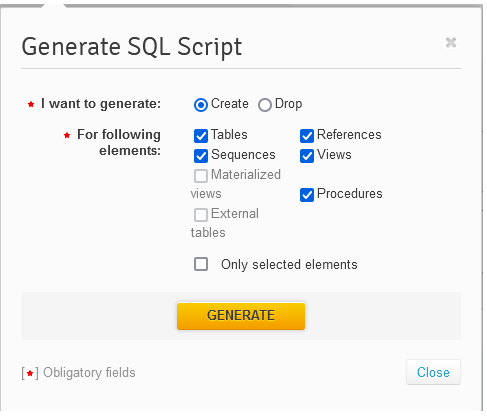
1. Manually write a script with CREATE and VALUE statements. You can either write this at the prompt (can be tedious if you make errors), copy/paste a script into the prompt, or save your script as a .sql file and use ‘SOURCE’ to load your script like you did when you loaded the nation database for the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

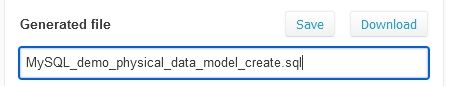
You may feel this option is easier/faster than using Vertebelo. However, if you drew your model in Vertebelo, then the script is just a 4 step point-and-click process.

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

**#Populating tables**

insert into Commodity values(10,'paddy','grains');

insert into Commodity values(11,'millets','grains');

insert into Commodity values(12,'poultry','meat');

insert into state values(101,'Telangana','south');

insert into state values(102,'Delhi','north');

insert into state values(103,'Tamil Nadu','south');

insert into District values(121,'Ranga Reddy','ranga reddy',101);

insert into District values(122,'vadai pakam','konnol',103);

insert into District values(123,'New Delhi','delhi',102);

insert into Market values(1,'Akash market','akash nagar',121);

insert into Market values(2,'Fruits market','Amberpet',121);

insert into Market values(3,'paini market','akainavi',122);

insert into Market values(4,'reddy market','bn reddy',123);

insert into prices values(50,'12/17/2021',30,50,1,10);

insert into prices values(51,'12/17/2021',35,55,2,10);

insert into prices values(52,'12/17/2021',220,250,4,12);

insert into prices values(53,'12/17/2021',80,100,3,11);

insert command is used to insert the data into a table.

Using source command to create tables.

Text

Description automatically generated

Using source command to populate the table created.

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

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

The indexes that I’ve chosen are on market id, state id and price id. I’ve chosen them since price is most important attribute for a commodity and different makets have different prices. hence to better fetch them I’ve created those indexes.

**create index ndx\_mID**

**on Market(market\_id);**

**create index ndx\_sID**

**on state(state\_id);**

**create index ndx\_pID**

**on prices(price\_id);**

by using the index, the records in that table are fetched faster than without using the index

**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 market\_prices as

select m.market\_name, c.commodity\_name, p.price\_min, p.price\_max

from market m, commodity c, prices p

where p.market\_id=m.market\_id and p.commodity\_id=c.commodity\_id;

Text

Description automatically generated with medium confidence

create view grain\_commodities as

select commodity\_id, commodity\_name

from commodity

where commodity\_variety='grains';

Text

Description automatically generated with low confidence

I think that the 1st view can help in understanding the various market’s commodity prices hence I created it while the 2nd view lists all the commodities under the grains variety which can be useful in calculating the grain amount.

An agricultural government officer might benefit the most from 1st view while for the 2nd view, the farmer benefits the most.

**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 no\_negative\_cID before insert on commodity

for each row

begin

if new.commodity\_id<=0 then

signal sqlstate '45000' set message\_text = 'commodity id cannot be less than or equal to zero!';

end if;

end$

delimiter ;

this trigger tells the user that it is invalid for commodity id to have negative values whenever he tries to insert a commodity record with id <=0.

Text

Description automatically generated

**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

The transactions are a very important aspect in maintaining the acid properties of the database. This is because the transactions operate in a way which follows the acid behaviour.

Text

Description automatically generated

**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

My users are db administrator who oversees the whole database. Next is the db analysist, who manages the database and updates it periodically. Lastly a technician who sees to the security of the database.

grant all on \*.\* to 'admin'@'localhost';

grant all on AgriManage.\* to 'analysist'@'localhost';

grant select,update,delete,insert on AgriManage.\* to 'security'@'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

Locking tables is necessary during operations on it since it prevents any inconsistencies that arises during the operation. E.g.: if you two different persons try to update the same record simultaneously in a table, the data will be inconsistent.

Text

Description automatically generated

Here the commodity table is locked and no insertions, updations or deletions can take place. Only the select stmt works.

**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 would like to back up the database every 10 days since by 10 days of time huge amount of data will be collected and the location ill choose is cloud since even if the current disk crashes ill still have the database copy. I would automate the database back up process by creating a .bat script and scheduling it using windows scheduler. The encryptions algorithms will keep it secure.

Text

Description automatically generated

Graphical user interface, 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, Error  
  
table = ['Commodity', 'State', 'District', 'Market', 'Prices']  
*# Trying to connect to database*try:  
 con = mysql.connector.connect(host='localhost', user='root', password='maria@123', database='AgriManage')  
 rec\_file = open("records.txt", "w")  
 print("Database and file successfully connected")  
 cur = con.cursor()  
  
 *# fetching the records* for ta in table:  
 cur.execute("select \* from " + ta)  
 print("Records of "+ta + "\n")  
 rec\_file.write("Records of "+ta + "\n")  
 for rec in cur.fetchall():  
 msge = ""  
 for r in rec:  
 msge += str(r) + ", "  
 msge += "\n"  
  
 *# writing the records to the txt rec\_file* rec\_file.write(msge)  
 print(msge)  
 rec\_file.write("\n")  
 print("\n")  
  
except Error as err:  
 print("error has taken place")  
 print(err)  
 exit()  
finally:  
  
 *# finally closing the connection* con.close()  
 rec\_file.close()

**OUTPUT**

**Background pattern

Description automatically generated with low confidence**

**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