**Employee Management System**

**CPSC 5090** Database systems 004

Deepsai Vemulapalli - [deepsaivemulapalli@lewisu.edu](mailto:deepsaivemulapalli@lewisu.edu)

Saiprakash Reddy Appidi - [saiprakashreddyapp@lewisu.edu](mailto:saiprakashreddyapp@lewisu.edu)

Manisai Bejjanki - [manisaibejjanki@lewisu.edu](mailto:manisaibejjanki@lewisu.edu)

Lewis University

Raymond Klump

GitHub: https://github.com/DeepsaiVemulapalli/Project\_description\_database-system-project.git

**All your efforts this semester will focus on a single project for which you will develop, document,** **implement, secure, and program with a database. You will post the artifacts of work 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. This too you should post as part of your GitHub repository.**

**Your project will consist of the following.**

**a. Title Page**

**The title page shows the title of your project and the number and name of this class, lists the names and email addresses of you or your team, and shows your GitHub repository's address, which you've shared with me.**

**Rubric: The title page is included and shows the necessary content. 5 points**

**Total points possible: 5**

**b. Initial Proposal**

**You will describe the data you aim to store. What data will be storing? Why are you interested in this data? Why is it important? Where will the data come from? Who will use this data? What kind of application do you plan to build with it?**

The whole data of Employee Management system is present in four different files. Employees personal information like first\_name, last\_name, phone\_number, hire\_date, email, salary, department\_id, job\_id, commission\_pct,manager\_id are stored in the Employees table. Employees has an attribute called manager\_id self-referencing foreign key to this table. This data can be used to get all the details of an employee if employee\_id which is primary key is known for a particular person. Data for this table was present in csv format.

Next table or data we have in Employee Management system is Departments. It has attributes like department\_id, department\_name, manager\_id and location\_id. Department\_id should be the primary key and department\_name should not be null. Employees and Job\_history have a foreign key to this table. Data of Department is present in json format.

Jobs table holds the different names of job roles within the company. Employees table has a foreign key to this table. It has following attributes: job\_id, job\_title, min\_salary and max\_salary. Data is inserted into this table from json format in which data is initially stored. Job\_history table holds the history of jobs that employees have held in the past. All the above three tables have a foreign key to this table.

**Rubric: Your response to each of these six 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**
* **1 point: necessary details are lacking in many of your responses.**

**You will also earn 2 additional points for coming up with a descriptive title for your project.**

**Total points possible: 20**

**c. Data Sources**

**Gather your data in text files. The text files may be csv, tab-delimited, xml, json, or some other custom format. Not all the files need be of the same type. Identify what each file contains by indicating where it came from, explaining in detail how it structured, and describing how you will reorganize the data into a relational database. Post your data files to your GitHub repository, and provide samples of the data in your Word doc.**

Employee database is a critical piece of information, and it is being managed in silos by multiple teams within the company. The reason it’s important is that it’s a key point of contact between your business and your employees. By collecting all this information from different teams in one place, it provides better insight and can help us to make better decisions in the business. One can make sure that they have employees with the right skills that help increase the revenue. An employee database is essential for organizations that are looking to streamline their human resources (HR) processes, as well as their business processes. This is because a well-designed, organized employee database will not only help you run your business smoothly, but it will also help you improve the performance and productivity as well.

In today's technologically advanced world, businesses need to have a better idea of their employees to retain the best and improve the weakest. The employee database may be a powerful tool that can assist you in meeting these goals. Whether one is interested in keeping a complete list of each employee or need a quick and easy way to keep up with the team's performance, the employee database can help.

**Some of the benefits of an employee database are:**

**Better Employee Performance:** The employee database is a simple yet effective tool that can be used to measure and manage the performance of employees. With all of the personal information stored, it can be easy to find information about your employees, their strengths, and weaknesses. The information can then be used to help your employees improve their performance.

**Better Retention:** Another benefit of the employee database is that it can help you keep your employee's contact information. Even if your employees move or leave the company, the contact information can still be kept in the database. If you're interested in making sure that your company knows who is coming and going, the employee database is a perfect solution.

**Accessible Data:** One more advantage to the employee database is that it's accessible, which means you can access it from anywhere. That way, if you need to check up on your employee's work status or performance, you don't have to run to their desk, which can save a lot of time.

**Rubric: Your work will be graded as follows:**

* **5 points: you gathered multiple data files that contain the data that will populate your databases. If you do not use multiple data files, you will not receive credit.**
* **5 points: you described the contents of the data files in detail, including referencing their origin and explaining how they were structured.**
* **3 points: you identify which fields you plan to include in your database, including their data types and any constraints you expect to impose on the data or steps you'll have to take to clean up the data.**
* **2 points: you post the data files to your GitHub account and make it possible for me to see them.**

**Total points possible: 15**

**d. Alternative Ways to Store the Data**

**We will study alternatives to storing data in a relational database. Some of the alternatives come from several decades ago, including the hierarchical and network models. Some are newer options, such as NoSQL databases that use JSON or some other encoding. Describe in detail how to store the data using two alternatives to relational databases. Be sure to describe how you would implement the alternatives and the advantages and disadvantages of each.**

Alternative ways to store the data

Alternative - 1

An employee database isn't the only way to store employee information. If you don't want to create an employee database, you can also use a spreadsheet to store your employee information. A spreadsheet can also store employee information.

Advantages: Cost: Using a spreadsheet can be inexpensive, which is another advantage to using it. You don't need to spend money on adding data entry software to your computer or buying an employee database. Usability: The data is more easily accessible to everyone, which means that anyone can be able to view it. If you're looking to keep track of your employee's daily activities, it's easier to use a spreadsheet than an employee database.

Disadvantages: A spreadsheet isn't a good place to keep large amounts of data, such as a spreadsheet containing a list of your employees. The spreadsheet can also have issues managing change in information. A spreadsheet also can be difficult to maintain if you have a lot of data to store. You need to have a certain amount of skill to keep the data in a spreadsheet up to date.

Alternative - 2

Network model: This type of structure can be good if you're interested in tracking employee data throughout your company. If this data is spread out across different parts of the company, it could be difficult to keep an eye on it. This model is good for companies with numerous offices or campuses, since this data would be distributed among the various locations.

Advantages: The main advantage of the network model is the fact that it's a centralized structure. This means that there would be a central point of data, and any information related to that data would also be centralized. This makes it easy to access the information, and make sure that employees are following a certain strategy. Simple Setup, no setup cost, ease of use and easy to change.

Disadvantages: More Complex: Requires a lot of coordination Network model is more difficult to implement in a start-up.

**Rubric: Your work will be graded as follows**

* **5 points for clearly describing how your data could be stored using one alternative to relational databases and what the advantages and disadvantages of that approach would be.**
* **5 points for clearly describing how your data could be stored using another alternative to relational databases and what the advantages and disadvantages of that approach would be.**

**Total points possible: 10**

**e. Relational Database Design Process**

**Consider the list of fields you identified in part c. Identify functional dependencies that exist among them. For each functional dependency, identify the determinants and the fields they determine. This becomes the basis for identifying your entity sets, which then become your tables. Give each entity set or table you identify in this way a unique and clear name, making sure that the names you use are singular nouns. Then list the relationships that exist among the various entity sets. For each relationship, identify its connectivity (one-to-one, one-to-many, many-to-many) and participation (optional or mandatory). Finally, make sure that none of the attributes you've assigned to each entity set are multi-valued. If they are, take the steps needed to break them down.**

In Employee table, job\_id is a foreign key which references and takes the value from job\_id in Jobs table. Another attribute manager\_id also references the employee\_id of the employee table itself. But manager\_id will be matching with the manager\_id of the particular Department based on Department table. Department\_id of the employees table is also a foreign key which is referenced from the department\_id of the Department table. Employee\_id of the Employee table is the primary key which is unique for each person and non-nullable value.

In Job\_history table, employee\_id is the foreign key which references the employee\_id in the Employees table. Along with that, job\_id and department\_id references the job\_id and department\_id of the Job and Department tables respectively.

In Department table, department\_id is the primary key, It has one foreign key called manager\_id which references employee\_id of Employee table.

Jobs table is the table which has no foreign keys. But it has primary key called job\_id which can be used in other tables for getting job\_id.

Many-to-One relationship is a more complicated relationship where there is more than one way for a set of instances of one entity to be related with one instance of another entity. For example, a user can have many phone numbers, so a Many-to-One relationship is a one-to-many relationship from user to phone number. Employees has many to one relationship with Department table. In this, department\_id in Employees table are maintaining many to one relationship with department\_id in Department table. Employee table also maintains many to one relationship with Jobs table. That relationship is formed by the job\_id present in both the tables.

**Rubric: Your work will be graded as follows:**

* **8 points for identifying all the functional dependencies, including determinants and the columns whose values they determine.**
* **2 points for naming the entity sets that make up your data with clear, easy-to-understand names.**
* **6 points for identify the relationships among the entity sets and identifying connectivity and participation for each.**
* **2 points for breaking down multi-valued attributes.**

**Total points possible: 18**

**f. Relational Database Design**

**This is where you will complete your database design. For each of the entity sets you identified in the preceding section, analyze them to make sure they pass 2nd, 3rd, 4th, and Boyce-Codd Normal Form. If they do not, introduce additional entity sets or key changes to make sure that they do. Then, add foreign keys to connect entity sets that are related. For many-to-many relationships, introduce bridge entity sets to convert them into two one-to-many relationships. Also, consider whether you should introduce surrogate keys to create a more efficient primary key for some of your entity sets. Finally, diagram your design in Vertabello. Make sure your ER diagram correctly shows all entity sets, their primary and foreign keys, the data types for each attribute, and the connectivity and participation characteristics of each entity set. Your final Vertabello design should be something you could actually implement in a relational database management system.**

1st NF:

Employee Job history is not in 1st normal form at the time of initial data modelling

"employee\_id";"start\_date";"end\_date";"job\_id";"department\_id"

102;"2001-01-13";"2006-07-24";"IT\_PROG";60

101;"1997-09-21";"2001-10-27";"AC\_ACCOUNT,AC\_MGR ";110

201;"2004-02-17";"2007-12-19";"MK\_REP";20

114;"2006-03-24";"2007-12-31";"ST\_CLERK";50

122;"2007-01-01";"2007-12-31";"ST\_CLERK";50

200;"1995-09-17";"2001-06-17";"AD\_ASST,AC\_ACCOUNT";90

176;"2006-03-24";"2006-12-31";"SA\_REP,SA\_MAN";80

after converting to 1st normal form

"employee\_id";"start\_date";"end\_date";"job\_id";"department\_id"

102;"2001-01-13";"2006-07-24";"IT\_PROG";60

101;"1997-09-21";"2001-10-27";"AC\_ACCOUNT";110

101;"2001-10-28";"2005-03-15";"AC\_MGR";110

201;"2004-02-17";"2007-12-19";"MK\_REP";20

114;"2006-03-24";"2007-12-31";"ST\_CLERK";50

122;"2007-01-01";"2007-12-31";"ST\_CLERK";50

200;"1995-09-17";"2001-06-17";"AD\_ASST";90

176;"2006-03-24";"2006-12-31";"SA\_REP";80

176;"2007-01-01";"2007-12-31";"SA\_MAN";80

200;"2002-07-01";"2006-12-31";"AC\_ACCOUNT";90

2nd NF:

JOB\_HISTORY(2nd normal form)

below data is in 1st normal form

"employee\_id";"start\_date";"end\_date";"job\_id";"department\_id"

102;"2001-01-13";"2006-07-24";"IT\_PROG";60

101;"1997-09-21";"2001-10-27";"AC\_ACCOUNT";110

101;"2001-10-28";"2005-03-15";"AC\_MGR";110

201;"2004-02-17";"2007-12-19";"MK\_REP";20

114;"2006-03-24";"2007-12-31";"ST\_CLERK";50

122;"2007-01-01";"2007-12-31";"ST\_CLERK";50

200;"1995-09-17";"2001-06-17";"AD\_ASST";90

176;"2006-03-24";"2006-12-31";"SA\_REP";80

176;"2007-01-01";"2007-12-31";"SA\_MAN";80

200;"2002-07-01";"2006-12-31";"AC\_ACCOUNT";90

if we assume employee\_ID and Job\_id are primitive attributes and start date and end date are not primitive attributes then

we have to split above data into

EMPLOYEE

"employee\_id";"start\_date";"end\_date"

102;"2001-01-13";"2006-07-24"

101;"1997-09-21";"2001-10-27"

101;"2001-10-28";"2005-03-15"

201;"2004-02-17";"2007-12-19"

114;"2006-03-24";"2007-12-31"

122;"2007-01-01";"2007-12-31"

200;"1995-09-17";"2001-06-17"

176;"2006-03-24";"2006-12-31"

176;"2007-01-01";"2007-12-31"

200;"2002-07-01";"2006-12-31"

EMPLOYEE\_JOB

"employee\_id";"job\_id"

102;"IT\_PROG"

101;"AC\_ACCOUNT"

101;"AC\_MGR"

201;"MK\_REP"

114;"ST\_CLERK"

122;"ST\_CLERK"

200;"AD\_ASST"

176;"SA\_REP"

176;"SA\_MAN"

200;"AC\_ACCOUNT"

3rd NF:

"employee\_id";"first\_name";"last\_name";"email";"phone\_number";"hire\_date";"job\_id";"salary";"commission\_pct";"manager\_id";"department\_id"

100;"Steven";"King";"SKING";"515.123.4567";"2003-06-17";"AD\_PRES";24000;0;0;90

101;"Neena";"Kochhar";"NKOCHHAR";"515.123.4568";"2005-09-21";"AD\_VP";17000;0;100;90

102;"Lex";"De Haan";"LDEHAAN";"515.123.4569";"2001-01-13";"AD\_VP";17000;0;100;90

103;"Alexander";"Hunold";"AHUNOLD";"590.423.4567";"2006-01-03";"IT\_PROG";9000;0;102;60

104;"Bruce";"Ernst";"BERNST";"590.423.4568";"2007-05-21";"IT\_PROG";6000;0;103;60

105;"David";"Austin";"DAUSTIN";"590.423.4569";"2005-06-25";"IT\_PROG";4800;0;103;60

106;"Valli";"Pataballa";"VPATABAL";"590.423.4560";"2006-02-05";"IT\_PROG";4800;0;103;60

107;"Diana";"Lorentz";"DLORENTZ";"590.423.5567";"2007-02-07";"IT\_PROG";4200;0;103;60

108;"Nancy";"Greenberg";"NGREENBE";"515.124.4569";"2002-08-17";"FI\_MGR";12008;0;101;100

109;"Daniel";"Faviet";"DFAVIET";"515.124.4169";"2002-08-16";"FI\_ACCOUNT";9000;0;108;100

110;"John";"Chen";"JCHEN";"515.124.4269";"2005-09-28";"FI\_ACCOUNT";8200;0;108;100

111;"Ismael";"Sciarra";"ISCIARRA";"515.124.4369";"2005-09-30";"FI\_ACCOUNT";7700;0;108;100

112;"Jose Manuel";"Urman";"JMURMAN";"515.124.4469";"2006-03-07";"FI\_ACCOUNT";7800;0;108;100

113;"Luis";"Popp";"LPOPP";"515.124.4567";"2007-12-07";"FI\_ACCOUNT";6900;0;108;100

114;"Den";"Raphaely";"DRAPHEAL";"515.127.4561";"2002-12-07";"PU\_MAN";11000;0;100;30

115;"Alexander";"Khoo";"AKHOO";"515.127.4562";"2003-05-18";"PU\_CLERK";3100;0;114;30

None of the employee non primitive attributes are having transitive dependency on primitive attribute employee\_id.

Below is the snapshot of physical database model form Vertabello:

Diagram

Description automatically generated

**Rubric: Your work will be graded as follows:**

* **4 points for the normalization analysis of your entity sets.**
* **3 points for introducing bridge entity sets.**
* **3 points for choosing foreign keys and perhaps more efficient surrogate keys**
* **10 points for correctly depicting your physical database model in Vertabello**

**Total points possible: 20**

**g. Data Definition Language Scripts**

**Use Vertabello to generate a script of SQL commands that build the database and its table structures. 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.**

-- Table: DEPARTMENTS

CREATE TABLE DEPARTMENTS (

Dept\_ID int NOT NULL,

Department\_Name varchar(30) NOT NULL,

Manager\_ID int NOT NULL,

Location\_ID int NOT NULL,

CONSTRAINT DEPARTMENTS\_pk PRIMARY KEY (Dept\_ID)

);

-- Table: EMPLOYEES

CREATE TABLE EMPLOYEES (

Employee\_ID int NOT NULL,

First\_Name varchar(30) NOT NULL,

Last\_Name varchar(30) NOT NULL,

Email varchar(30) NOT NULL,

Phone\_Number varchar(15) NOT NULL,

Hire\_Date date NOT NULL,

Salary double(10,2) NOT NULL,

Comm double(10,2) NOT NULL,

Job\_ID varchar(30) NOT NULL,

Dept\_ID int NOT NULL,

Manager\_ID int NOT NULL,

CONSTRAINT EMPLOYEES\_pk PRIMARY KEY (Employee\_ID)

);

-- Table: JOBS

CREATE TABLE JOBS (

Job\_ID varchar(30) NOT NULL,

Job\_Title varchar(50) NOT NULL,

Min\_Salary int NOT NULL,

Max\_Salalry int NOT NULL,

CONSTRAINT JOBS\_pk PRIMARY KEY (Job\_ID)

);

-- Table: JOB\_HISTORY

CREATE TABLE JOB\_HISTORY (

Employe\_ID int NOT NULL,

Dept\_ID int NOT NULL,

Job\_ID varchar(30) NOT NULL,

Start\_Date date NOT NULL,

End\_Date date NOT NULL,

);

-- foreign keys

-- Reference: EMPLOYEES\_DEPARTMENTS (table: EMPLOYEES)

ALTER TABLE EMPLOYEES ADD CONSTRAINT EMPLOYEES\_DEPARTMENTS FOREIGN KEY EMPLOYEES\_DEPARTMENTS (Dept\_ID)

REFERENCES DEPARTMENTS (Dept\_ID);

-- Reference: EMPLOYEES\_EMPLOYEES (table: EMPLOYEES)

ALTER TABLE EMPLOYEES ADD CONSTRAINT EMPLOYEES\_EMPLOYES

FOREIGN KEY (Manager\_ID)

REFERENCES EMPLOYEES (Employee\_ID);

-- Reference: EMPLOYEES\_JOBS (table: EMPLOYES)

ALTER TABLE EMPLOYEES ADD CONSTRAINT EMPLOYEES\_JOBS FOREIGN KEY EMPLOYEES\_JOBS (Job\_ID)

REFERENCES JOBS (Job\_ID);

-- Reference: JOB\_HISTORY\_EMPLOYEES (table: JOB\_HISTORY)

ALTER TABLE JOB\_HISTORY ADD CONSTRAINT JOB\_HISTORY\_EMPLOYEES FOREIGN KEY JOB\_HISTORY\_EMPLOYEES (Employee\_ID)

REFERENCES EMPLOYEES (Employee\_ID);

Graphical user interface, application, Word

Description automatically generated

**Rubric: Your work will be grades as follows:**

* **Database and table creation statements from Vertabello saved as a sql script file: 3 points**
* **Scripts you write or Excel spreadsheets you create to generate SQL commands for populating the tables, uploaded to GitHub: 8 points**
* **Descriptions of the scripts and Excel spreadsheets you wrote along with code excerpts included in the Word document: 5 points**
* **Screenshots of your successful attempts to use the MySQL *source* command to populate each table with at least three records: 4 points**

**Total points possible: 20**

**h. 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, and save your queries in a commented sql script to GitHub.**

Insert Statements:

Insert statement used o insert the data into tables

Graphical user interface, table

Description automatically generated with medium confidence

Update Statements:

We can update the table data using update statement with primary key value in a table

Graphical user interface, text, application, Teams

Description automatically generated

Graphical user interface, application

Description automatically generated

After updating data:

Graphical user interface, application

Description automatically generated Delete Statements:

Graphical user interface, text, application

Description automatically generated

Select statement:

Select statement used to fetch all rows from table

Graphical user interface, application

Description automatically generated

Join Statements:

We can use joins to retrieve the data from more than one table/same table.

Joined emp table with jobs

Graphical user interface, application

Description automatically generated with medium confidence

Joined emp table with Dept table to know the department details of that employees

Graphical user interface, application

Description automatically generated

Summary Statements:

Below query gives number employees count

Graphical user interface, text, application

Description automatically generated

Get Highest Salary using below query from employee table

Graphical user interface, text, application

Description automatically generated

Multi-table query:

Graphical user interface, application

Description automatically generated

Sub-queries:

Displaying max salary of emp details

Graphical user interface, application, Word

Description automatically generated

**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**

**i. 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.

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

**j. Views**

Add two views to your database to provide easy access to combinations of data from multiple tables.

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

**k. Triggers**

Add a trigger to a table so that data will be updated when a certain event occurs

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

**l. Transactions**

Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior?

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

**m. 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.

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

**n. 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.

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

**o. 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?

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 and for including a portion of the resulting file.

Total points possible: 15

**p. 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, and also post the program to your GitHub repository.

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 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

**q. 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

**r. 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.

Rubric: Your work will be graded as follows:

* 3 points for clearly describing the limitations of your databases
* 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.

Total points possible: 12

**s. Activity Log**

As an appendix, the team will keep a daily diary or log of their activity. What did you or your team study in this class each day? What did you learn? What did you accomplish or build or design? You don't have to enter something every day, but there should be at least three entries each week. Since we have eight weeks, that means you should make 3 posts to the Activity Log each week, for a total of at least 24 posts. Each post will be worth 1 point.

If you are working as part of a team, make sure you clearly identify which team member worked on which tasks. The Activity Log should help me figure out how each team member contributed to the project. If I cannot discern who worked on what aspects of the project from the activity log, no points will be awarded for it.

**Activity Log**

|  |  |  |
| --- | --- | --- |
| Name | Week | Activity |
| Deepsai Vemulapalli | Week 1 | Formed a group and arranged a meeting with team to discuss on the database system project. |
| Week 2 | Prepared initial Proposal |
| Week 3 | Working on vertibola |
| Week 4 | SQL statements & inserting values |
| Manisai Bejjanki | Week 1 | Searching resources like internet, journal, csv files to find the datasets to work on. |
| Week 2 | Gathering all data sources |
| Week 3 | Worked on Quires related, and entity related |
| Week 4 | Table structure and insert values |
| Saiprakash Reddy Appidi | Week 1 | Collecting required data to start the project and planning to work on project as a group. |
| Week 2 | Searched for the alternatives and its advantages and disadvantages. |
| Week 3 | Created queries except the insert queries |
| Week 4 | Insert sequel table creation |

Total points possible: 24

**Periodic progress checks**

To make sure you and your team are progressing, you will submit your paper periodically for me to review and comment on. Each submission is worth 20 points. I will not be grading the quality of each section, but rather whether you've made progress on the required pieces by the deadline. If any part of what is required on a particular day is missing or seems to have gotten very little attention, you will earn 0 points out of 20 for the progress check. You must demonstrate that you are making progress on completing each part by the deadlines listed below.

|  |  |
| --- | --- |
| Deadline | Sections for which you must demonstrate significant progress |
| November 6 at 11:59pm | a. Title page  b. Initial proposal  c. Data sources  d. Alternative ways to store the data  s. Activity Log – at least six entries covering the first two weeks |
| November 20 at 11:59pm | e. Relational database design process  f. Relational database design  g. Data definition language scripts  h. Data manipulation language scripts  s. Activity Log – at least six entries covering the past two weeks |
| December 4 at 11:59pm | i. Indexes  j. Views  k. Triggers  l. Transactions  m. Security  n. Locking  o. Backup  r. Suggested future work  s. Activity Log – at least six entries covering the past two weeks |

I will review your progress update and provide comments. Please then use those comments to improve your work. In that way, you can guarantee that your final product will be A-quality work.

**Final Report and Submission**

Your final report is due Friday, December 17, at 6pm. Your github site must be complete at that time as well, containing all the files that comprise your project, and you must have shared access to the repository with me.

The overall quality of your paper in terms of the clarity and correctness of your writing will count for ten points. How well you organize your GitHub repository to present your work in an easy-to-understand manner will count for another ten points.

**Final grade**

In total, then, this is how points will be awarded:

5 Title Page

20 Initial Proposal

15 Data Sources

10 Alternative Ways to Store the Data

18 Relational Database Design Process

20 Relational Database Design

20 Data Definition Language Scripts

30 Data Manipulation Language Scripts

12 Indexes

8 Views

6 Triggers

6 Transactions

14 Security

6 Locking

15 Backup

18 Python Programming

26 PHP Programming

12 Future Work

10 Overall quality of paper

10 Organization of GitHub Site

20 Progress Update 1 due November 6 at 11:59pm

Covers parts a, b, c, d, and s.

If any part is missing or shows very little progress, 0 points

will be awarded.

20 Progress Update 2 due November 20 at 11:59pm

Covers parts e, f, g, h, and s

If any part is missing or shows very little progress, 0 points

will be awarded.

20 Progress Update 3 due December 4 at 11:59pm

Covers parts i, j, k, l, m, n, o, r, and s

If any part is missing or shows very little progress, 0 points

will be awarded.

24 Activity Log (must clearly indicate which team members worked

on which tasks, or 0 points will be awarded)

If you add up all the scores, you find that the number of points you can earn this course is 365.

**Teamwork**

You may work in teams of up to 3 people. If you do work with others, make sure you divide the work so that everyone has an opportunity to learn and practice skills. Make sure your activity log names who did what for each entry. If your activity log fails to identify how individuals of your team contributed, you will earn 0 points out of 24 for the Activity Log.

**Pace yourself**

What you submit at the end of the semester is what will determine the bulk of your final grade. The periodic check-ins are important milestones, and I will provide you feedback at each of them to help you improve your work and refine it into a product that is worthy of a great score at the end. If you don't follow my advice, or if you don't work judiciously to accomplish the parts that are listed for each milestone, please do not expect to receive a good grade.

**Template**

[Here's a template](https://www.dropbox.com/s/npcb3zuk9alpanl/project_template.docx?dl=0) you should use to structure your report. Just fill in the various sections as the course proceeds.