# A. Title Page

Lewis University  
CPSC 50900: Database Systems   
Spring 2025 Term Project

Military Personnel Flight: Retentions Contract Database

Sri Komma, srirkomma@lewisu.edu

Travis Lester, travisllester@lewisu.edu

Work products stored in the Github repository <https://github.com/TravisLLester/MPF-Retentions-Database.git>

Table of Contents

[A. Title Page 1](#_heading=h.gjdgxs)

[Schedule of Milestones 1](#_heading=h.30j0zll)

[B. Initial Proposal 2](#_heading=h.1fob9te)

[C. Data Sources 2](#_heading=h.3znysh7)

[D. Alternative Ways to Store the Data 3](#_heading=h.tyjcwt)

[E. Conceptual and Logical Models 3](#_heading=h.3dy6vkm)

[F. Physical Model 4](#_heading=h.4d34og8)

[G. Populate the database with data 5](#_heading=h.2s8eyo1)

[H. Data Manipulation Language (DML) Scripts 5](#_heading=h.17dp8vu)

[I. Indexes 6](#_heading=h.3rdcrjn)

[J. Views 6](#_heading=h.26in1rg)

[K. Stored Programs (Stored Procedures, Stored Functions, Triggers) 6](#_heading=h.lnxbz9)

[L. Transactions 7](#_heading=h.35nkun2)

[M. Database Security 7](#_heading=h.1ksv4uv)

[N. Locking and Concurrent Access 7](#_heading=h.44sinio)

[O. Backing Up Your Database 8](#_heading=h.2jxsxqh)

[P. Programming 8](#_heading=h.z337ya)

[Q. Suggested Future Work 8](#_heading=h.3j2qqm3)

[R. Activity Log 9](#_heading=h.1y810tw)

# Schedule of Milestones

Here is a schedule that shows when each milestone is due and what sections comprise it.

| Deadline | Sections for which you must demonstrate significant progress |
| --- | --- |
| February 4 at 11:59pm | a. Title page  b. Initial proposal  c. Data sources  d. Alternative ways to store the data  r. Activity Log – at least six entries covering the first two weeks |
| February 18  at 11:59pm | e. Conceptual and logical models  f. Physical model  g. Populate the database with data  r. Activity Log – at least six entries covering the past two weeks |
| March 4 at 11:59pm | h. Data manipulation language (DML) scripts  i. Indexes  j. Views  l. Transactions  m. Security  r. Activity Log – at least six entries covering the past two weeks |

The remaining sections – Triggers, Locking and Concurrency, Backup, and Programming, will be turned in with the final report, which is due March 16 at 11:59pm.

# B. Initial Proposal

*Description: 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?*

*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.*

*As you consider various ideas for your project, keep in mind that your database is going to have to store data for at least 8 different types of things. Each of these different “types of things” will become a table in the database you design and build. So, the idea can’t be so narrow that you can’t identify at least eight different types of things in it that you’d store data about.*

*Total points possible: 20*

**What data will be storing?**

The data we will be storing is crucial for managing the re-enlistment and contract extension process for every Air Force member stationed at Kirtland Air Force Base who wishes to extend or renew their contract. Currently, the process relies heavily on email communications and a hierarchical folder system, leading to significant challenges in tracking the status of contracts. These documents often get lost or remain untouched for extended periods, which causes delays in processing. As a result, contracts can go past due, leading to instances where Airmen inadvertently fall into "no-pay" status, which can have serious financial consequences.

Our new data management system will address these issues by streamlining contract processing and providing real-time reporting capabilities. The system will be integrated with a front-end Microsoft Access interface, with the back end stored in a MySQL database. This setup will allow for efficient tracking and reporting on various aspects of the contract lifecycle, including upcoming deadlines, contracts under review, past due contracts, and more.

**We will store data related to:**

Team Members: Information on team members responsible for processing contracts, including their assignments to specific contracts. This will allow me to provide statistical data on how many contracts each member completes.

Contract Types: We will distinguish between contract renewals and extensions to ensure clarity in the contract processing stages. This differentiation will also provide valuable statistical data for leadership, particularly in tracking Kirtland Air Force Base's retention rates. Currently, we can track how many individuals separate from the military each year at Kirtland, but by incorporating data on how many Airmen choose to stay and extend their contracts, we can provide leaders with a comprehensive view of retention trends. This insight will help inform decision-making and support strategies aimed at improving retention at the base.

Contract Statuses: Tracking the progress of contracts through various stages, such as pending, drafted, awaiting signatures, sent to members, finalization, and confirmation.

Airmen's Information: Details on the Airmen requesting contract renewals or extensions, including their squadron affiliation and their commanding officer's name, which is required for final approval on the contract. It will also link to First sergeants’ information which is crucial if member goes past due on their contract.

Squadron Details: Information about the squadrons (or units) to which Airmen belong, including their commander, first sergeant, contact information, and CSS point of contact. This data is essential for follow-up and tracking, especially when Airmen are unreachable or when contracts are overdue.

Major Commands (MAJCOMs): Tracking which squadrons fall under each Major Command at Kirtland Air Force Base. This data will provide leadership with statistical insights on contract status across different units. We currently have 6 different Major Commanders at Kirtland.

First Sergeants: Critical data on first sergeants who are the primary point of contact when an airman falls into "no-pay" status. They coordinate with finance to resolve payment issues manually if necessary.

The system will ensure timely processing, help prevent delays, and provide leadership with valuable data on contract statuses, including metrics on completion, drafts, and more. This centralized approach will ensure that no contract is lost in the shuffle and will enable us to provide critical support to Airmen while ensuring they remain under contract without disruption.

**Why are you interested in this data?**

We are deeply invested in this data because the timely processing of contracts directly impacts the careers and financial stability of Airmen. When a contract is delayed due to inefficiencies, disorganization, or lack of oversight, it places an airman’s career in jeopardy. These delays can lead to severe consequences, including missed assignments, lost Permanent Change of Station (PCS) opportunities, and even "no-pay" status—where a service member is left without a paycheck.

Since being stationed at Kirtland Air Force Base, Travis has personally witnessed over 100 cases of Airmen going unpaid, more than 50 Airmen losing assignments, and countless instances of frustration and stress due to delayed contract processing. These failures not only affect individual service members but also undermine unit readiness and morale.

By implementing a real-time tracking system, we aim to eliminate these issues by providing complete visibility into contract statuses. This system will ensure contracts are processed efficiently, prevent Airmen from falling into financial hardship, and give leadership the data they need to monitor and improve retention efforts. It’s time to move beyond outdated, manual processes and bring an organized, data-driven approach to contract management at Kirtland Air Force Base.

**Why is it important?**

Ensuring the efficient and timely processing of contracts is critical for both the individual Airmen and the overall operational readiness of Kirtland Air Force Base. When contracts are delayed or mishandled, Airmen can fall into "no-pay" status, lose assignment opportunities, or even face disruptions in their military careers. These setbacks create unnecessary financial and emotional stress for service members, impacting their morale, focus, and overall well-being.

Beyond the individual level, ineffective contract management also affects the Air Force's mission readiness. If Airmen are unable to transition smoothly between assignments or extend their service as planned, it can lead to staffing shortages, decreased unit effectiveness, and administrative inefficiencies. Leadership also lacks crucial data to assess retention trends, making it difficult to address manpower challenges proactively.

By implementing a real-time tracking system, we will provide complete visibility into the contract process, ensuring accountability, efficiency, and timely execution. This will not only safeguard Airmen’s careers but also enhance unit effectiveness and provide leadership with accurate data for strategic decision-making. In short, improving contract management is not just about paperwork, it’s about taking care of our people, strengthening operational readiness, and ensuring the Air Force retains its highly trained personnel.

**Where will the data come from?**

Due to the Privacy Act of 1978, most of the data in our system will be anonymized or simulated to protect personal information. However, we will leverage publicly available Air Force data, including contract types, Major Commands, Squadrons, First Sergeants, Commanders, and Commander’s Support Staff.

For internal tracking and assignment purposes, the team will utilize an Alpha Roster that contains all data, which is provided weekly to Travis. This roster contains essential personnel information needed to manage contract processing effectively. However, to maintain compliance with privacy regulations, we will omit personally identifiable information (such as real names, addresses, and email addresses) when storing or displaying data in our system.

By structuring the system in this way, we can ensure data security while still enabling effective tracking, reporting, and decision-making for contract management at Kirtland Air Force Base. When Travis implements this on the Air Force Secure Network, he will then add all the private information for full operational use.

Who will use this data?

This data will be used by several key personnel involved in contract management and personnel tracking at Kirtland Air Force Base, including:

Retention Team Members – They will use the system to track contract extensions and reenlistments, ensuring timely processing and preventing delays that could lead to pay interruptions or assignment losses.

Assignment Team Members – Responsible for managing Airmen’s career progress, assignments, and contract updates, ensuring smooth transitions between duty stations and career stages.

Commanders and First Sergeants – They will have access to contract status updates to monitor Airmen under their command. If a contract is delayed or an airman is at risk of falling into “no-pay” status, they can intervene to expedite the process.

Commander’s Support Staff (CSS) – They will assist in tracking contract deadlines and ensuring required paperwork is completed promptly. If a contract is overdue, they will coordinate with leadership to resolve issues.

Finance Personnel – If an airman enters "no-pay" status due to a contract processing delay, finance teams will work closely with First Sergeants and Commanders to manually process pay adjustments.

Base Leadership & Analysts – The system will generate reports on contract processing efficiency, retention statistics, and potential problem areas. This data will help leadership identify trends, allocate resources, and improve retention strategies at Kirtland Air Force Base.

By centralizing and streamlining contract tracking, this system will ensure greater efficiency, accountability, and transparency in managing Airmen’s career extensions and reenlistments.

**What kind of application do you plan to build with it?**

After completing this course project, Travis plans to develop a comprehensive contract management system with a user-friendly Microsoft Access front end and a robust MySQL back end. The course project will lay the foundation for the back-end database structure, ensuring a scalable and efficient system.

This system is specifically designed for the retention team at Kirtland Air Force Base to streamline contract processing, enhance efficiency, and minimize errors. In hopes of creating this, maybe other bases can adopt and utilize as well.

Key Features and functionality:

User-Friendly Interface – The Microsoft Access front end will provide an intuitive and accessible interface for team members to manage contracts without needing advanced database knowledge

Contract Management – Users will be able to create new contracts, automatically storing them in the MySQL database, ensuring that all data is structured and easily retrievable.

Real-Time Contract Tracking – The system will allow users to search for specific Airmen, view their contract details, and update contract statuses (e.g., Pending, Drafted, Awaiting Signatures, Sent to Member, Finalized, or Confirmed).

Automated Reporting & Queries – The application will include predefined queries to generate reports with a single click. Users will be able to pull real-time data on:

* Upcoming contract expirations
* Past-due contracts
* Retention statistics
* Processing timelines and bottlenecks
* Breakdowns by Major Command (MAJCOM), squadron, and unit

Role-Based Access –

Retention team members will have full editing rights to manage contracts.

Commanders, First Sergeants, and Finance personnel will have read-only access, allowing them to check contract statuses in real time.

This reduces the hundreds of unnecessary emails previously sent to the organization’s inbox, improving response times and efficiency.

Long-Term Benefits:

By digitizing and centralizing contract tracking, this system will eliminate lost documents, reduce processing delays, and prevent Airmen from falling into "no-pay" status. It will also improve communication across squadrons, finance, and leadership while providing valuable retention data to base leadership.

# C. Data Sources

*Description: 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 is 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.*

*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*

Air Force Personnel Center (AFPC) -<https://www.afpc.af.mil/>

Kirtland AFB Home website - [Kirtland Air Force Base > Home](https://www.kirtland.af.mil/)

Air Education and Training command website - [Air Education and Training Command > Home](https://www.aetc.af.mil/)

Personnel Alpha Roster pulled from Air Force Network PC

* Alpha Roster
* Wings
* Groups
* Squadrons
* MAJCOMs
* Commanders
* First Sergeants

# D. Alternative Ways to Store the Data

*Description: 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.*

*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*

Currently, contract data at Kirtland Air Force Base is stored using a secure folder-based hierarchical system. While this method provides a structured approach, it lacks efficiency and transparency, leading to delays, miscommunication, and unnecessary workload.

Current Process Breakdown:

Received Folder – When contract requests arrive via a worksheet, they are manually placed into this folder.

Drafted Folder – Once a contract is drafted, it is moved here. At this stage, the document awaits approval from a Non-Commissioned Officer (NCO), who ensures accuracy.

Sent to Member Folder – After NCO approval, the contract is emailed to the Airman, and the document is moved to this folder.

Ready for Processing Folder – When the signed contract is returned, it is placed here, awaiting final processing in the Air Force system.

Review Folder – Three days after processing, the contract is checked for errors. If successful, it moves forward; if not, it stays here until corrections are made.

Completed Folder – Once verified, the contract is placed in this final folder, confirming that it has been successfully updated in the system and that the Airman’s pay is secure.

Challenges with the Current Hierarchical System:

Lack of Access Control – Only certain individuals have access to the secure drive, leading to hundreds of emails requesting contract statuses.

Limited Visibility on Contract Ownership – There is no way to track who is working on a contract at any given time.

Human Error – Contracts can easily be misplaced if a team member forgets to move the file to the correct folder, leading to lost or delayed contracts.

Inefficient Tracking – The current system lacks real-time monitoring, making it difficult to see contract status at a glance.

Redundant Workload – Team members must manually check, verify, and move files, adding unnecessary steps and increasing the risk of processing errors.

Advantages:

* Fast access to hierarchical data: Since the structure is predefined, retrieving data along a specific path is efficient, however, maintaining it is difficult. You forget to do one step by dragging the folder to the next folder, causing issues and misplacement.
* Data Integrity: Relationships between records are enforced by the tree structure, minimizing redundant data.
* Security & Access Control: Data access can be managed at different levels, restricting access based on hierarchy levels.

Disadvantages:

* Limited Flexibility: Hierarchical models struggle with dynamic relationships that may change over time.
* Redundant Data Storage: Since relationships are strictly parent-child, certain contract details may need to be duplicated across different nodes in order to create the many-to-many or many-to-one relationships.
* Complex Maintenance: Adding new contract states or restructuring the hierarchy requires significant database modification. This causes delays and people not wanting to change the current structure.

A second approach is to use a NoSQL document store based on JSON files, similar to MongoDB. MongoDB stores data as JSON objects, allowing for flexible and scalable data structures. Each contract is stored as a JSON document containing all relevant details in a single, self-contained entry.

For the data we are tracking, each contract would follow a structure like this:

{

“Contract\_ID”: “001”,

“Member”: “John Doe”,

“rank”: “SSgt”,

“Unit”: “377th Air Base Wing”,

“email”: “johndoe@us.af.mil”

}

A record like this would be created for every individual with a contract. The JSON files can then be imported into the database and processed line by line.

Advantages:

* High Flexibility: The schema-less design allows contracts to evolve without rigid structural constraints.
* Efficient Querying: NoSQL databases support indexing and flexible queries, enabling faster searches.
* Scalability: JSON-based databases are optimized for horizontal scaling, making them well-suited for distributed data management.
* Real-Time Updates: Changes to contracts can be reflected instantly across the system.

Disadvantages:

* Complex Query Processing: Unlike SQL, NoSQL databases may require additional application-side logic for complex joins.
* Data Consistency Challenges: Since NoSQL follows an eventual consistency model, real-time synchronization across distributed nodes may require extra effort.
* Higher Storage Requirements: Because each JSON document contains all relevant contract details, some data redundancy may occur.

This approach is particularly useful for environments where scalability and flexibility are prioritized over strict relational constraints.

# E. Conceptual and Logical Models

*Description: First, come up with a conceptual model. The conceptual model identifies the entity sets and the relationships among them. For each relationship, identify the connectivity and the participation (optional or mandatory).*

*Now that you know the entity sets, the next step is to develop the logical model by adding attributes. For each entity set, identify the attributes that describe the entity set. This may include references to other entity sets that are involved in relationships. Then, identify the functional dependencies that exist among them. For each functional dependency, identify the determinants and the fields they determine, like this:*

*determinant, or, determinants 🡪 attributes, they, determine*

*This becomes the basis for identifying your entity sets, which will become your tables when we move to the physical model in the next section. The attributes listed on the left of the arrows are candidates to become your primary key attributes. Attributes that are references to other entity sets are candidates to become the foreign keys.*

*For entity sets that have multi-attribute determinants, replace them with surrogate keys. This makes it easier to identify each entity in the set and to define foreign keys.*

*Then apply normalization to make sure that your design satisfies First, Second, and Third Normal forms. For 1st Normal Form, make sure that all attributes are indivisible. This may require adding an entity set that lists values that appear in comma-separated lists as individual entities. For 2nd Normal Form, make sure there are no partial dependencies (this won’t be a problem if all your entity sets have single-attribute determinants). Finally, make sure all your entity sets are in 3rd Normal Form. This means that you have to split transitive dependencies into separate entity sets and add relationships between the original entity set and the new ones.*

*Finally, draw the logical model as an ERD. At this point, your design will have entity sets, their relationships, and their attributes. M:N relationships are acceptable at this point, as we’ll remove them in the physical model.*

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

* *5 points for identifying all entity sets*
* *5 points for writing each relationship between entity sets as two sentences and correctly identifying their connectivity and participation.*
* *5 points for adding attributes to entity sets and writing the functional dependencies correctly. Replace multi-attribute determinants with surrogate keys.*
* *4 points for performing the normalization steps. Make sure your design is in 3rd Normal Form.*
* *5 points for drawing the ERD for the logical model. At this point, the ERD will show entity sets, relationships, attributes, and primary identifiers. The design may include M:N relationships at this point. We’ll get rid of those in the physical model.*

*Total points possible: 24*

ENTER YOUR RELATIONAL DATABASE DESIGN DESCRIPTION HERE. INCLUDE A PICTURE OF YOUR ERD.

# F. Physical Model

*Description: This is where you will complete your database design. Add data types, including size constraints, uniqueness constraints, and auto-incrementing for all attributes. Replace many-to-many relationships with two one-to-many relationships using bridge entity sets. Add additional entity sets that you think could be helpful for storing the acceptable values of particular attributes. (For example, if you were storing student data, valid student statuses might include Good Standing, Graduated, On Probation, Expelled. Put those in a table and create a relationship back to the student table). Draw the ERD for the physical model.*

*Using the final ERD, write the SQL DDL statements needed to create the database, its tables, and the relationships among them. Run these statements in MySQL to build your database. Provide screen shots that show the database you built in MySQL, including its tables and descriptions of some of the tables. To show a list of databases and a list of the tables in a particular database, use the show command. To see a description for a table, use the describe command.*

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

* *3 points for introducing bridge entity sets (if necessary)*
* *3 points for adding data types and other constraints on the data.*
* *3 points for introducing other entity sets and their relationships that help enforce what values can be assigned to particular attributes (if necessary)*
* *5 points for drawing the ERD for the physical model. If you used Vertabelo, the resulting ERD must be free of errors and warnings*
* *6 points for generating the SQL scripts that build the database and then running the script in mysql. Demonstrate that the script built the database and its tables with screenshots that show that you ran the show and describe commands.*

*You will be penalized 4 points if your database doesn’t have at least 8 appropriately defined tables.*

*Total points possible: 20*

DESCRIBE THE STEPS YOU TOOK TO COMPLETE THE PHYSICAL MODEL. THEN SHOW THE ERD FOR THE PHYSICAL MODEL. THEN SHOW THE SQL COMMANDS THAT BUILD THE DATABASE. (FOR THIS, YOU MAY REFER TO A PARTICULAR FILE IN YOUR GITHUB REPOSITORY. MAKE SURE YOU INVITE ME AS A COLLABORATOR ON YOUR REPOSITORY SO THAT I CAN ACCESS THE SCRIPT.) FINALLY, SHOW SCREEN SHOTS THAT PROVE THAT YOU BUILT THE DATABASE AND ITS STRUCTURES IN MYSQL.

# G. Populate the database with data

*Description: You built the database in section F, and it now exists in mysql. Now populate it with your data. Take your original data source or sources and generate insert statements from them. Store the insert statements in a text file, and then use the mysql source command to run these insert statements to populate the various table structures. Generating the necessary insert statements may require writing Python scripts or manipulating Excel databases to convert the data from your original data sources.*

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

* *Explain step-by-step and very clearly how you created the required SQL statements from your initial data. Write it as a set of instructions. 5 points*
* *Show the file of insert statements that you ran in MySQL. You may do this either by including the listing in this report or by identifying the file in your GitHub that contains the insert statements. Make sure I have access to your GitHub repository. 4 points*
* *Show screenshots of the data in your MySQL database. To do this, run select statements for each table and show screen shots of what is displayed: 5 points*

*Total points possible: 14*

*import json*

*data = { # Create a Python dictionary*

*"name": "My Data",*

*"value": None, # Use None directly*

*"count": 123,*

*"is\_active": True,*

*"items": [1, 2, 3],*

*"nested\_object": {*

*"a": "hello",*

*"b": False*

*}*

*}*

*json\_data\_string = json.dumps(data) # Convert the dictionary to a JSON string*

*print(json\_data\_string) # For debugging*

*try:*

*data\_loaded = json.loads(json\_data\_string) # Parse the JSON string*

*print(data\_loaded["name"])*

*print(data\_loaded["value"])*

*# ... print other data ...*

*except json.JSONDecodeError as e:*

*print(f"JSON Decode Error: {e}")*

*# Handle the error*

The code provided demonstrates how to load JSON data into a Python dictionary, which can then be used to generate SQL INSERT statements for populating the database.

# H. Data Manipulation Language (DML) Scripts

*Description: 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.*

*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.*
* *6 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: 24*

ENTER DML WORK HERE

# I. Indexes

*Description: 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:*

* *3 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*
* *2 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

*Total points possible: 8*

ENTER YOUR INDEX WORK HERE

# J. Views

*Description: 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:*

* *3 points for including the SQL for generating the two views in your Word document*
* *3 points for including screenshots for the data contained in each view in your Word document*
* *3 points for explaining why each view is a valuable addition to your database*

*Total points possible: 9*

ENTER YOUR WORK WITH VIEWS HERE

# K. Stored Programs (Stored Procedures, Stored Functions, Triggers)

*Description: Add a stored procedure, stored function or trigger to a table and demonstrate using it.*

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

* *3 points for including the SQL for the stored program (procedure, function, or trigger in your Word document*
* *3 points for clearly explaining the purpose of the stored program*
* *3 points for a screenshot and explanation that shows the stored program in action.*

*Total points possible: 9*

ENTER YOUR WORK WITH STORED PROGRAMS HERE

# L. Transactions

*Description: 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:*

* *5 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: 8*

ENTER YOUR WORK WITH TRANSACTIONS HERE

# M. Database Security

*Description: 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:*

* *4 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: 12*

ENTER YOUR WORK WITH DATABASE SECURITY HERE

# N. Locking and Concurrent Access

*Description: 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: 5*

ENTER YOUR WORK WITH LOCKING AND CONCURRENT ACCESS HERE

# O. Backing Up Your Database

*Description: 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:*

* *6 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)*
* *2 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: 8*

ENTER YOUR WORK ON DATABASE BACKUPS HERE

# P. Programming

*Description: Write a Python, Java, or PHP 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:*

* *10 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*
* *6 points for showing a screenshot of your running the script and showing the results it produces on the screen.*

*Total points possible: 18*

ENTER YOUR PYTHON, PHP, or JAVA DATABASE PROGRAMMING WORK HERE

# Q. Suggested Future Work

*Description: 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*

ENTER YOUR SUGGESTED FUTURE WORK IDEAS HERE

# R. Activity Log

*Description: As an appendix, the team will keep a frequently updated 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.*

*Total points possible: 24*

**WEEK 1**

Entry 1: During Week 1, we started by reviewing foundational concepts related to databases. Although both Sri and I have prior experience with MySQL, this week provided an important refresher. We particularly focused on different database models, and we appreciated revisiting the Hierarchical Storage Model. It had been over a year since we last worked with traversal methods like post-order, in-order, and pre-order, so this review helped reinforce my understanding of these concepts. By revisiting these foundational principles, I gained a clearer perspective on their applications within real-world database systems.

Entry 2: At the end of Week 1, Sri and I discussed forming a team for our project. We brainstormed several potential database ideas and, after thorough discussion, decided to develop a contract database management system. Our goal is to design a solution that tracks reenlistment contracts and extensions for the Air Force. We identified several key challenges our system would address: ensuring contracts are submitted and processed efficiently, managing the high volume of contracts, and improving the workflow and transparency in the review process. Although we considered an alternative project for Amazon’s database, we felt the contract management system was a more impactful choice that could solve immediate organizational challenges.

Entry 3: In the beginning of Week 2, Sri and I held a Google Meet session to map out the structure of our database. We began drafting tables and defining the essential data required for our project. Sri led the research, pulling data from public Air Force websites, while I focused on table design, relationship definitions, and establishing keys within Vertabelo. We also collaborated on completing Section B of our project, analyzing the benefits and feasibility of our database management system. Our joint effort and clear planning helped lay a solid foundation for the development of our final project. I felt confident in our approach and look forward to seeing it evolve.

WEEK 2

Entry 4: On Tuesday of Week 2, the class covered various database management systems, including the hierarchical model. We were familiar with the concept but didn’t realize that hierarchical systems were categorized in this way. The examples presented in class helped us understand how hierarchical structures influence key decisions in our own project, particularly around foreign keys and primary keys. Following the class, Sri and I discussed some challenges our team was facing with our database design. Sri reached out to better understand the issues at hand that Travis was facing, allowing us to refine our approach to the rough draft. We then started drafting a plan for which tables would be necessary to implement the system.

Entry 5: Over the weekend of Week 2, Sri and I met again via Google Meet to finalize Section B of the project proposal. After both of us had taken notes individually, we merged our ideas and worked together to create a polished version of the proposal. We also discussed the project title, striving to come up with a unique and professional name. Sri contributed by researching potential alternatives, while I worked on finalizing Section D of the alternative solutions. We also discussed Section C, where Sri pinged me with questions for clarification. With both of us working collaboratively, we created a comprehensive and well-rounded submission.

Entry 6: On Sunday, February 2, Sri and Travis reconvened via Google Meet to complete Section D: Alternative Solutions for the project. With that finalized, they shifted focus to Section C: Data Files. Sri conducted research and identified websites offering public datasets, while Travis gathered documents from his secure database on the Air Force network, exporting the data into a .csv file. To comply with privacy regulations, he anonymized individual names, ensuring only publicly available information remained unchanged. As the discussion progressed, Sri introduced Python code examples for importing JSON files, providing a foundation for future data integration. They also revisited their Vertabelo database schema, further refining the structure as they prepared for the next project submission.For Travis, he has experience with mySQL, but with much smaller database complexities. The complexity of databases with so many different foreign keys is rough, but coming along.