# A. Medical Device Startup Company Operations Database Application

Lewis University  
CPSC 50900: Database Systems   
Spring 2024 Term Project

Medical device startup company operations database application

Sharath Kumar Reddy Sashani, SharathKumarReddyS@lewisu.edu

Bindu Boddu, [binduboddu@lewisu.edu](mailto:binduboddu@lewisu.edu)

Work products stored in the Github repository

https://github.com/sharath561/Medical-device-database-system.git

Table of Contents

[A. Title Page 1](#_Toc157667443)

[Schedule of Milestones 1](#_Toc157667444)

[B. Initial Proposal 2](#_Toc157667445)

[C. Data Sources 2](#_Toc157667446)

[D. Alternative Ways to Store the Data 3](#_Toc157667447)

[E. Conceptual and Logical Models 3](#_Toc157667448)

[F. Physical Model 4](#_Toc157667449)

[G. Populate the database with data 5](#_Toc157667450)

[H. Data Manipulation Language (DML) Scripts 5](#_Toc157667451)

[I. Indexes 6](#_Toc157667452)

[J. Views 6](#_Toc157667453)

[K. Triggers 6](#_Toc157667454)

[L. Transactions 7](#_Toc157667455)

[M. Database Security 7](#_Toc157667456)

[N. Locking and Concurrent Access 7](#_Toc157667457)

[O. Backing Up Your Database 8](#_Toc157667458)

[P. Programming 8](#_Toc157667459)

[Q. Suggested Future Work 8](#_Toc157667460)

[R. Activity Log 9](#_Toc157667461)

# 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 |
| January 29 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 12  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 |
| February 26 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.

# 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.*

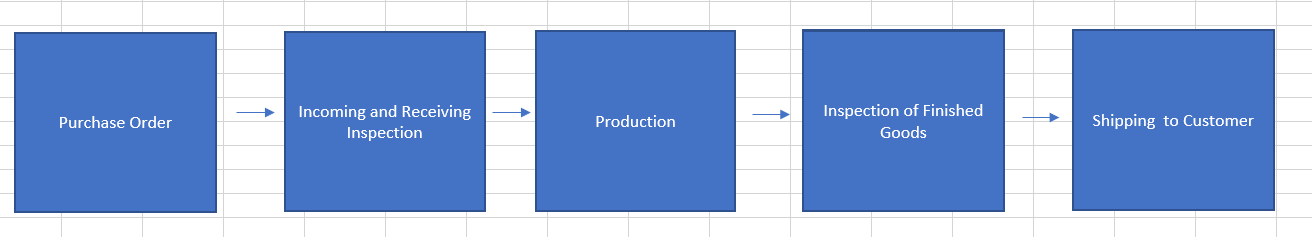
*As you consider various ideas for your project, keep in mind that your database is going to have to store data for at least 8 different types of things. Each of these different “types of things” will become a table in the database you design and build. So, the idea can’t be so narrow that you can’t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

*Total points possible: 20*

You will describe the data you aim to store. What data will be storing?

The purpose of this project is to implement a database system for a medical startup company, which will store the purchase order, vendor list , inventory list , and the analyzer part list , the cartridge part list, the quality list Etc. We are taking a hypothetical medical startup company which will create an analyzer and a cartridge. This cartridge will take the sample of the patient and will be inserted into analyzer and then this analyzer will process the cartridge and let the nurse know if the patient has covid or not. So, this project will focus mostly on the operations, Finance and the quality aspects of the company, the research and development and HR part is out of the scope for this project as all the R&D has been done and the final product which needs to be commercialized has been forwarded it to finance, quality and operations. The workflow of this project will start from a person who will place a purchase order, based on the lead time of the inventory. Once the purchase order has been placed and the item has been received it will go through the qualities and inspection to make sure the parts are conforming as per the quality and the FDA regulations. Parallelly we will also be tracking the work order of the amount of analyzer and cartridge that needs to be built. Once we have received the number of analyzer and cartridge to be built, that information will be sent to the operations and operations is going to start pulling the parts from the inventory. This database needs to be smart enough that the moment any new inventory comes in and the part out for making the devices is pulled, it needs to constantly update the values. Once all the analyzer and cartridge has been built and has been shipped the work order needs to be closed and the process needs to be repeated all over again.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purchase Orders | Quality | Production | Inventory Management | Shipping |
| Purchase Orders | Incoming/Receiving | Work Orders In/Out | Inventory Control Cycle/R&D/Scrap | Shipping |
|  | Finished Goods |  |  |  |
|  | MRB |  |  |  |



Why are you interested in this data?

We are trying to implement a whole database system for the medical startup company from scratch to final deliverable and push it to the production level. We will get the understanding of both database systems as well as the business operation side of running a company.

Why is it important?

It is important that we learn these technical and non-technical skills so that in the future when I'm employed, I can confidently say that the skills I acquired through this course and given any roles and responsibilities thrown at me, we can confidently finish the task or project. As we are also working in a team, we will learn a valuable skill of collaboration.

Where will the data come from?

As this is a hypothetical startup company so we will be creating the data from the scratch and also with the help of Google we can also look through some of the data which is available for other medical company or we can write a python script where it will automatically generate the data.

Who will use this data?

As we are implementing the databases system for the company, any person belonging to the following department like finance, operations and quality and also who is involved with any of the steps like placing a purchase order, performing the quality or inspection on the parts, performing the quality on the finished goods, creating the work orders, Interacting to the vendors to get a quote, the operation people who is going to look into the inventory and realize how much inventory we have on hand and if we are running low on inventory then they can approach the concerned person to place the order for those particular parts.

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

To create a database system where all the records of the concern departments like (finance, operations and quality) of a company will be stored. There will be a front end of database system where people can either query the database and update the values or insert the value into the database. This application is work in progress as we are working through the semester, this part will be updated accordingly.

# 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*

Identifying what each file contains by indicating where it came from,

Currently we have total number of 6 excel files. so, the files are purchase order, production workbook, quality, inventory control, shipping. So, all these five workbooks will be connected to the final workbook called Inventory management system and these will be connected via formula called get pivot table and anytime we will be updating any one of these CSV files and then if we refresh the final Inventory management system file all the values will be refreshed and updated.

explaining in detail how it is structured, and

So, each of these five CSV files will serve a purpose of either quality, shipping production purchase order and inventory control and all of the values will be corresponding to the functions of each operation. And as all of these values are either numeric or floating or character.

Describing how you will reorganize the data into a relational database.

We can create different tables based on different CSV data and all the columns can be used as an attribute and the corresponding data type can be extracted from the column name, also by creating the primary key, and foreign key we can build the entity-relationship model logically, and further implement physically. This is still work in progress and will be updated as we progress through this project.

Post your data files to your GitHub repository, and provide samples of the data in your Word doc.

Link is already provided.

Purchase order

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Name** | **Vendor Name** | **PO #** | **PO Issue Date** | **Item #** | **Quantity Ordered** | **Price per part** | **Total Price of Parts** | **Total PO Price** |
| 1/29/2024 | xyz | (Approved) Abcam | PO-0001 | 1/29/2024 | AS-00001 | 1 | $1.00 | $1.00 | $1.00 |

Quality

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Name** | **Item #** | **Quantity Passed Inspection** | **Quantity Issued to MRB** |
| 1/29/2024 | xyz | AS-00006 | 100 | 0 |

Incoming Inventory Control

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Receipt Date** | **Purchase Order Number** | **Vendor Name** | **Vendor Part Number** | **Description** | **Item #** | **Lot Number** | **Expiration Date** | **Total Item Quantity Received** | **Receiving Status** |
| xyz | 1/30/2024 | PO-0001 | Abcam | x-007 | This is sample | AS-00006 | 123456 | 1/1/2030 | 100 | Ready for QC |

Production

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Open Work Order Date** | **Closed Work Order Date** | **Name** | **Work Order #** | **Description** | **Item #** | **Quantity Being Built** | **Status** |
| 1/1/2024 | 1/30/2024 | First Name | WO-0001 | Packaged 20X Box Cartridge TBI | AS-00082 | 100 | WIP |

Shipping

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Name** | **Customer** | **Sales Order #** | **Packing List #** | **Item #** | **Quantity Shipped** |
| 1/30/2024 | First Name | University Medical Center of El Paso (AKA UMC of El Paso) | | 123456 | AS-00045 | 100 |

Inventory Management System

Analyzer Part List

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Vendor Part #** | **Item #** | **Description** | **Vendor** | **Location** | **Unit of Measure** | **Available Quantity** | **Total Inventory** | **Quantity Waiting Inspection** | **Quantity On Receiving Hold** | **Quantity on Production Hold** | **Quantity On Order** | **MRB Quantity** | **RMA Quantity** | **Scrapped Quantity** | **R&D Issued Quantity** |
| VP-1111112 | AS-00006 | Wire Harness, PCB to POGO |  | 2 | each | 18 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| VP-1111113 | AS-00016 | Analog Front End - AFE, Analyzer |  | 2 | each | 23 | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| VP-1111114 | AS-00018 | Test Card, PCB |  | 2 | each | 15 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Cartridge Part List

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Item #** | **Description** | **Total Inventory** | **Quantity Waiting Inspection** | **Quantity On Receiving Hold** | **Quantity on Production Quality Hold** | **Quantity On Order** | **MRB Quantity** | **RMA Quantity** | **Scrapped Quantity** | **R&D Issued Quantity** |
| AS-00069 | Anti-GFAP, EDC activated | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AS-00074 | Panel Subassembly, PMOS Cartridge | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AS-00078 | Midcard Fluidics, PMOS Cartridge | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Part Quantity

|  |  |  |
| --- | --- | --- |
| **Material Description** | **Part Number** | **Qty Per Build** |
| Wire Harness, PCB to POGO | AS-00006 | 1 |
| PCBA, AFE | AS-00016 | 1 |
| PCB, ETC | AS-00018 | 2 |
| PCB-A, Processor Board | AS-00019 | 1 |

# 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*

Alternative 1: NoSQL Database with JSON Encoding

NoSQL databases means not only SQL, so we can implement the relational database in NoSQL model particularly employing JSON encoding. To implement this, we can use MongoDB software. Data is typically stored in a document-oriented manner. Each record will be represented as a JSON. Any insert, update and delete of the record can be done to JSON.

Implementation:

* In the case of an Inventory Management System (IMS), the relational model likely consists of tables like Products, Orders, Customers, and Suppliers, each with their respective attributes and relationships.
* Each inventory item can be represented as a document, containing attributes such as item ID, description, quantity in stock, price, supplier details, etc. These attributes can be stored as key-value pairs within the document.

Advantages:

* NoSQL databases don't enforce a rigid schema like relational databases making it easier to implement the database.
* Since the data is being generated exponentially every day and with different type of data like audio, video, images we need a solution better that RDBMS which can handle these file format. NoSQL databases are designed for distributed environments, allowing for seamless scaling as data grows.
* With simpler data models and horizontal scalability, NoSQL databases can offer better performance for my application.
* Can be extremely fast and flexible

Disadvantages:

* There is no standard access language and all applications are custom built.
* NoSQL databases often sacrifice ACID (Atomicity, Consistency, Isolation, Durability) properties for scalability and performance. This can lead to eventual consistency and data integrity challenges. As we are storing FDA regulated data it becomes mandatory to have consistency and data integrity.
* Developers accustomed to relational databases may face a learning curve when transitioning to NoSQL databases and JSON-based data modeling.

Alternative 2: Hierarchical Database Model

The hierarchical database model organizes data in a tree-like structure with parent-child relationships. Each record contains information about its parent record, forming a hierarchy. Retrieving data typically involves navigating the tree structure from parent to child or vice versa. Access to data is often indexed for efficient retrieval. Since we have all the excel connected where Inventory Management System serving as a Parents and all the other files serving as a child, this way we can implement this model.

Implementation:

* The IMS data can be organized hierarchically, with a root node representing the inventory management system and child nodes representing different categories of inventory items, such as products, components, vendors, etc.
* Each inventory item is linked to its parent node, representing its category or group.

Advantages:

* Conceptually simple to implement
* Data integrity is ensured
* Offer efficient storage and retrieval is faster using indexed access.

Disadvantages:

* Rigid in structure, making it challenging to accommodate changes in data organization or relationships.
* It may struggle to scale with growing data and evolving application requirements, especially compared to modern NoSQL or relational databases.

# 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*

Conceptual Model:

* Entity Sets:
* Purchase Order
* Quality
* Receiving Status
* Vendor List
* Incoming Inventory Control
* Production
* Shipping
* Analyzer Part List
* Cartridge Part List
* Part
* Employee

Attributes and Functional Dependencies:

Purchase order table

PO # --> Date, Vendor ID,Item #,PO Issue Date,Quantity Ordered,Price per part,Total Price of Parts,Total PO Price

Quality table

Quality\_ID--> Date,Emp\_ID,Item #,Quantity Passed Inspection,Quantity Issued to Material Review Board

Receiving Status table

Receiving Status ID--> Receiving Status,Description

Vendor List table

Vendor ID --> Vendor Name, Vendor Part Number,Description, Item #

Incoming Inventory Control table

Incoming ID--> Emp\_ID,Receipt Date,Purchase Order Number,Vendor ID,Lot Number,Expiration Date,Total Item Quantity Received, Receiving Status ID

Production table

Work Order #--> Open Work Order Date,Closed Work Order Date,Name,Description,Item #,Quantity Being Built,Status

Shipping table

shipping ID --> Date,First Name, Last Name,Customer ,Sales Order #, Packing List #,Item #,Quantity Shipped

Analyzer Part List table

Item #--> Vendor Part #,Description,Vendor Location,Unit of Measure,Available Quantity,Total Inventory,Quantity Waiting Inspection,Quantity On Receiving Hold, Quantity on Production Hold,Quantity On Order,MRB Quantity, RMA Quantity,Scrapped Quantity, R&D Issued Quantity

Cartridge Part List table

Item #-->Description,Total Inventory,Quantity Waiting Inspection,Quantity On Receiving Hold,Quantity on Production Quality Hold,Quantity On Order,MRB Quantity,RMA Quantity,Scrapped Quantity,R&D Issued Quantity

Part table

part\_id --> Material description,Item #

Employee Table

Emp\_ID --> First Name, Last Name

Normalization:

1NF -- Every value is atomic(indivisible),Ensuring all attributes are atomic and no repeating groups exist.

Split all the names into first name and last name

2NF -- No Partial dependenncies,Eliminating partial dependencies to achieve Second Normal Form (2NF).

When a non-determinant depends on only part of the primary key

3NF -- Split transitive dependencies into separate entity sets and add relationships between the original entity set and the new ones.

Eliminating transitive dependencies to achieve Third Normal Form (3NF).

Relationships:

Purchase Order (PO) - Vendor List

Connectivity: One-to-Many (One PO can have one Vendor, but a Vendor can have multiple POs)

Participation: Mandatory on both sides (Each PO must have a corresponding Vendor, and each Vendor must have at least one associated PO)

Quality - Employee

Connectivity: Many-to-One (Many quality checks can be performed by one employee)

Participation: Mandatory on the Quality side, optional on the Employee side (Each quality check must be associated with an employee, but not all employees may be involved in quality checks)

Incoming Inventory Control (Incoming) - Receiving Status

Connectivity: One-to-One (Each incoming inventory control has one receiving status, and each receiving status is associated with one incoming inventory control)

Participation: Mandatory on the Incoming side, mandatory on the Receiving Status side (Each incoming inventory control must have a receiving status, and each receiving status must be associated with an incoming inventory control)

Incoming Inventory Control (Incoming) - Vendor List

Connectivity: Many-to-One (Many incoming inventory controls can be associated with one vendor)

Participation: Mandatory on the Incoming side, mandatory on the Vendor List side (Each incoming inventory control must have a vendor, and each vendor must have at least one associated incoming inventory control)

Production - Employee

Connectivity: Many-to-One (Many production records can be associated with one employee)

Participation: Mandatory on the Production side, optional on the Employee side (Each production record must be associated with an employee, but not all employees may be involved in production)

Shipping - Employee

Connectivity: Many-to-One (Many shipping records can be associated with one employee)

Participation: Mandatory on the Shipping side, optional on the Employee side (Each shipping record must be associated with an employee, but not all employees may be involved in shipping)

Analyzer Part List - Vendor List

Connectivity: Many-to-One (Many parts in the analyzer part list can be associated with one vendor)

Participation: Mandatory on the Analyzer Part List side, mandatory on the Vendor List side (Each part in the analyzer part list must have a vendor, and each vendor must have at least one associated part)

Cartridge Part List - Vendor List

Connectivity: Many-to-One (Many parts in the cartridge part list can be associated with one vendor)

Participation: Mandatory on the Cartridge Part List side, mandatory on the Vendor List side (Each part in the cartridge part list must have a vendor, and each vendor must have at least one associated part)

Analyzer Part List - Part

Connectivity: One-to-One (Each part in the analyzer part list is associated with one part entity)

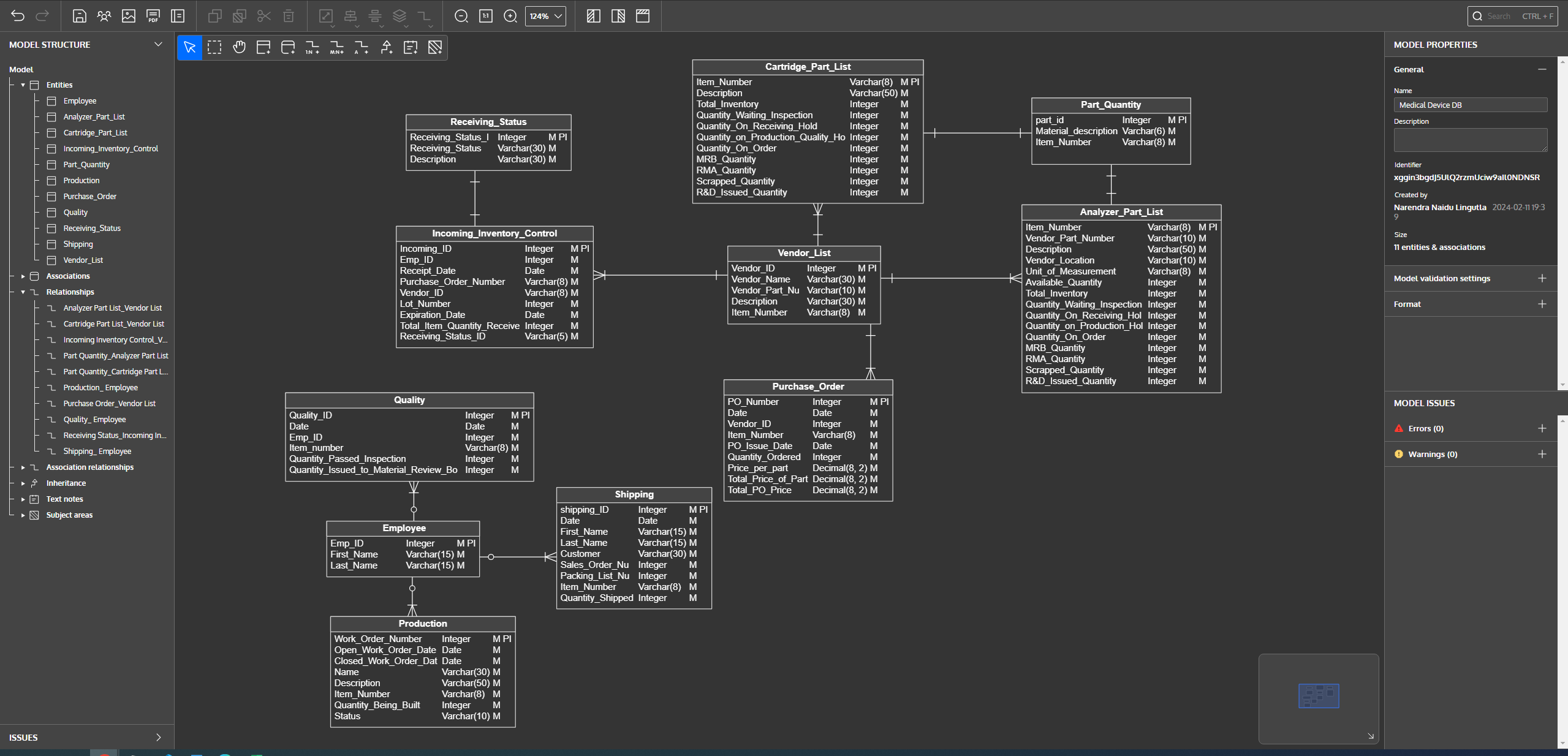
Participation: Mandatory on the Analyzer Part List side, masndatory on the Part side (Each part in the analyzer part list must have corresponding details in the part entity, and each part entity must be associated with a part in the analyzer part list)

Cartridge Part List - Part

Connectivity: One-to-One (Each part in the cartridge part list is associated with one part entity)

Participation: Mandatory on the Cartridge Part List side, mandatory on the Part side (Each part in the cartridge part list must have corresponding details in the part entity, and each part entity must be associated with a part in the cartridge part list)

ERD for Logical Model:



# 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.*

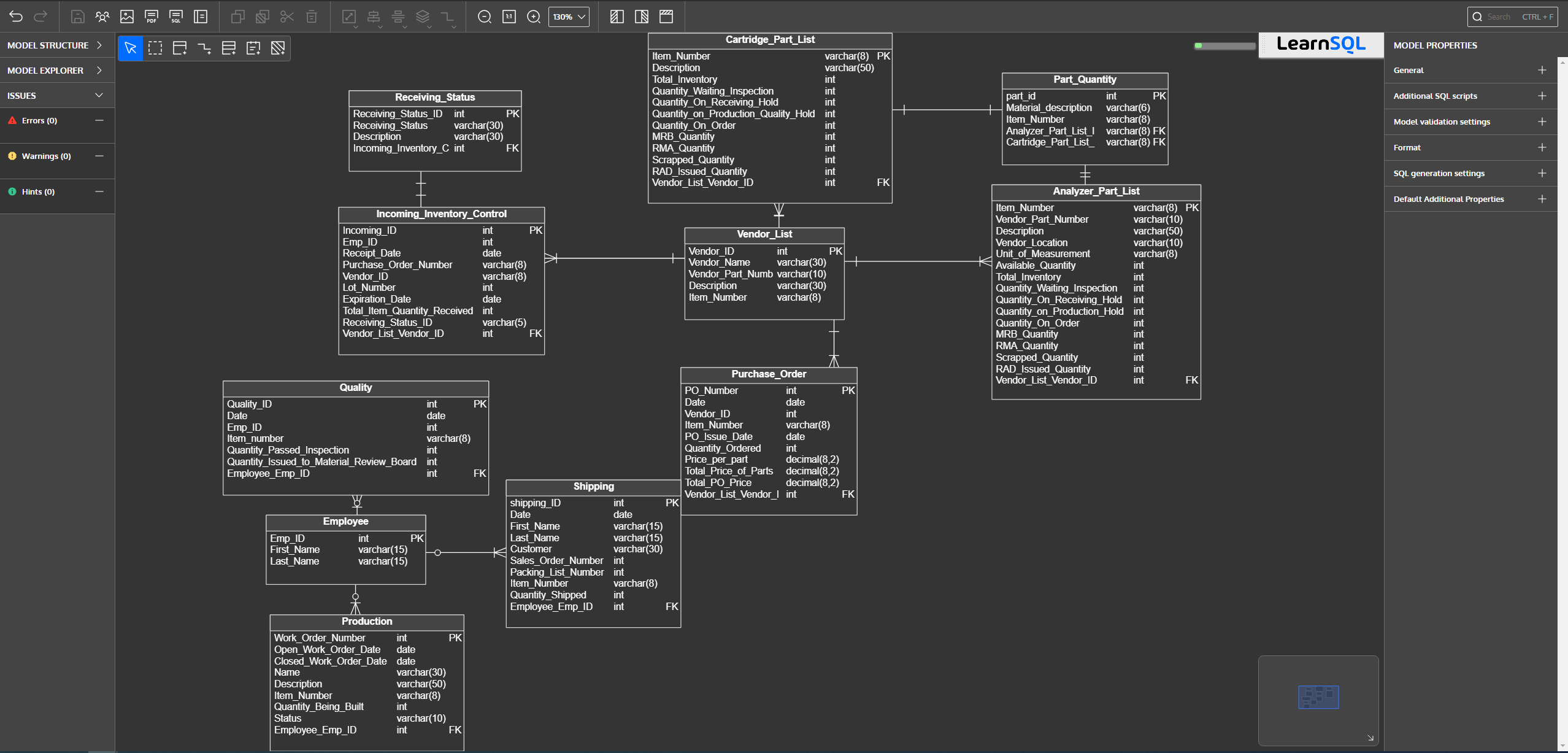
*You will be penalized 4 points if your database doesn’t have at least 8 appropriately defined tables.*

*Total points possible: 20*

Describe the steps you took to complete the physical model:

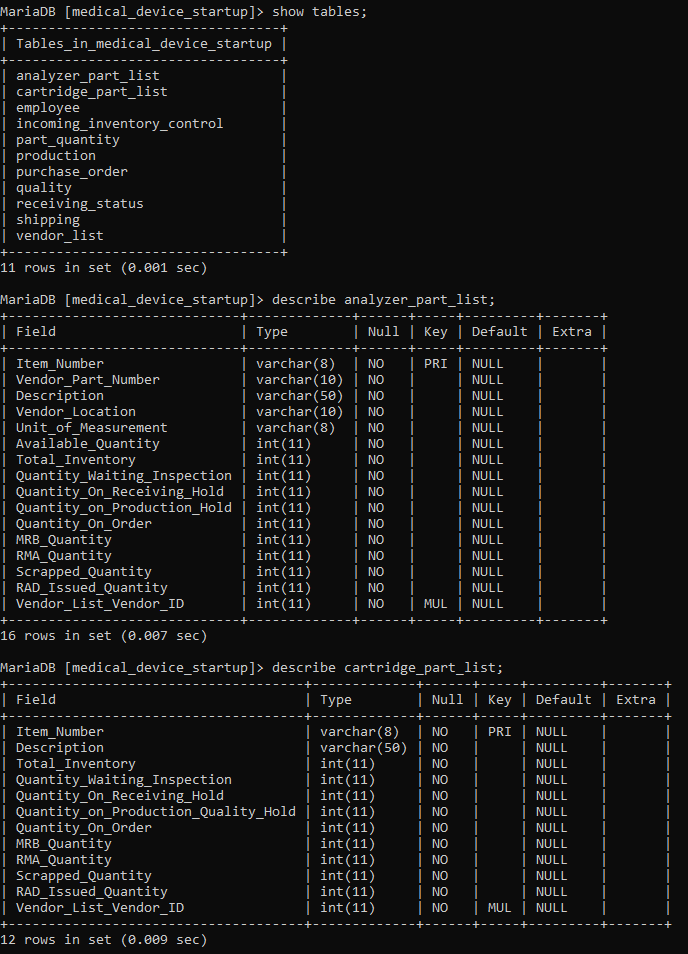
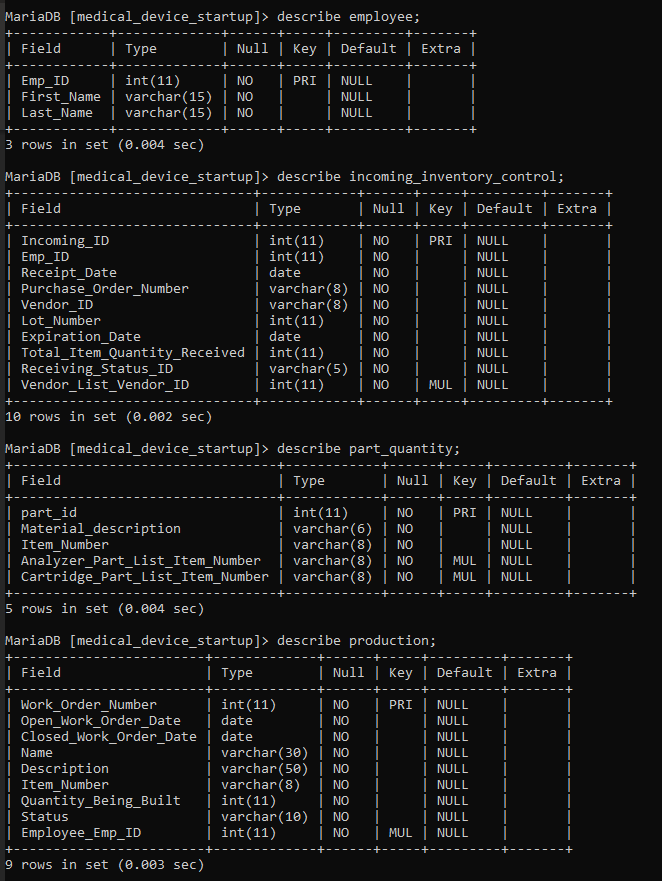
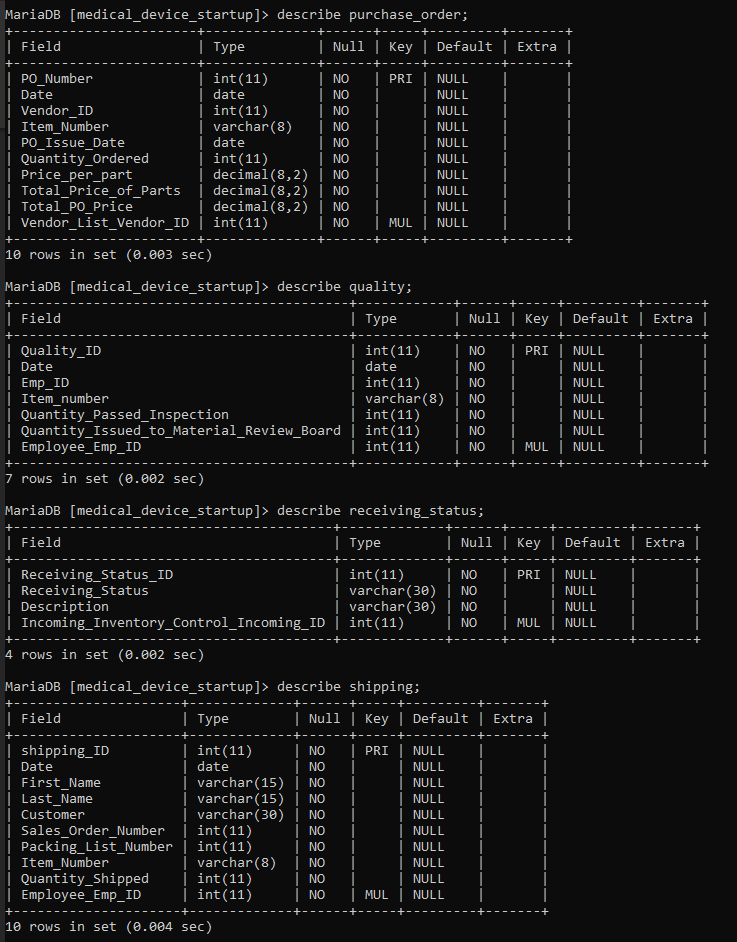
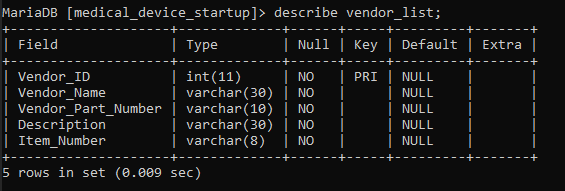
* After saving the logical model we can press Ctrl+G to generate physical model.
* We name our physical model and then select the target database engine, Here we are selecting MySql 8.x
* click generate physical model.
* A new tab will open with the workbook physical model.

ERD for Physical Model:



Show the sql commands that build the database.

* mysql -h localhost -u root -p
* create database medical\_device\_startup;
* use medical\_device\_startup;
* source Medical\_Device\_DB\_Physical\_Export\_create.sql
* show tables;
* describe analyzer\_part\_list;
* describe cartridge\_part\_list;
* describe employee;
* describe incoming\_inventory\_control;
* describe part\_quantity;
* describe production;
* describe purchase\_order;
* describe quality;
* describe receiving\_status;
* describe shipping;
* describe vendor\_list;

Screen shots that prove that we 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*

Explain step-by-step and very clearly how you created the required SQL statements from your initial data.

* First updated the excel sheet with the pseudo data.
* Describe all the table and got the column name and then created the insert statements manually.

Show the file of insert statements that you ran in MySQL.

Insert.sql in github

Show screenshots of the data in your MySQL database.

-- Insert

INSERT INTO Employee VALUES ('1001','Brosina','Hoffman');

INSERT INTO Quality VALUES ('1001','2024-12-02','1001','PN-0001','100','0','1001');

INSERT INTO Vendor\_List VALUES ('1001','Abcad','VP-0001','Hex Nuts','PN-0001');

INSERT INTO Shipping VALUES ('1001','2024-12-02','Mary','Kay','NMSU','SO-0001','PA-0001','PN-0001','1000','1001');

INSERT INTO Incoming\_Inventory\_Control VALUES ('1001','1001','2024-12-02','PO-0001','VI-00001','33671','2030-12-02','0445','001','1001');

INSERT INTO Analyzer\_Part\_List VALUES ('PN-00200','VP-111141','Shoulder Screw, #4-40, 1/8” x 1-1/2” Shoulder, 18-8 SS','USA','Each',4542,7120,0,0,0,0,0,0,0,0,'1001');

INSERT INTO Cartridge\_Part\_List VALUES ('PN-00500','Top Shroud, PMOS Cartridge','71250','0','0','0','0','0','0','0','0','1001');

INSERT INTO Part\_Quantity VALUES ('PN-00001','Bumper','AS-00006','PN-00200','PN-00500');

INSERT INTO Production VALUES ('WO-0001','2024-12-02','02/30/2024','New Order','From New Company','PN-00012','100','Building','1001');

INSERT INTO Purchase\_Order VALUES ('PO-0001','2024-12-02','VI-0001','PN-0001','02/01/2024','454','10.00','4540.00','4540.00','1001');

INSERT INTO Receiving\_Status VALUES ('1001','WIP','Work In Progress', '1001');

-- End of file.

SELECT \* FROM analyzer\_part\_list;

SELECT \* FROM cartridge\_part\_list;

SELECT \* FROM employee;

SELECT \* FROM incoming\_inventory\_control;

SELECT \* FROM part\_quantity;

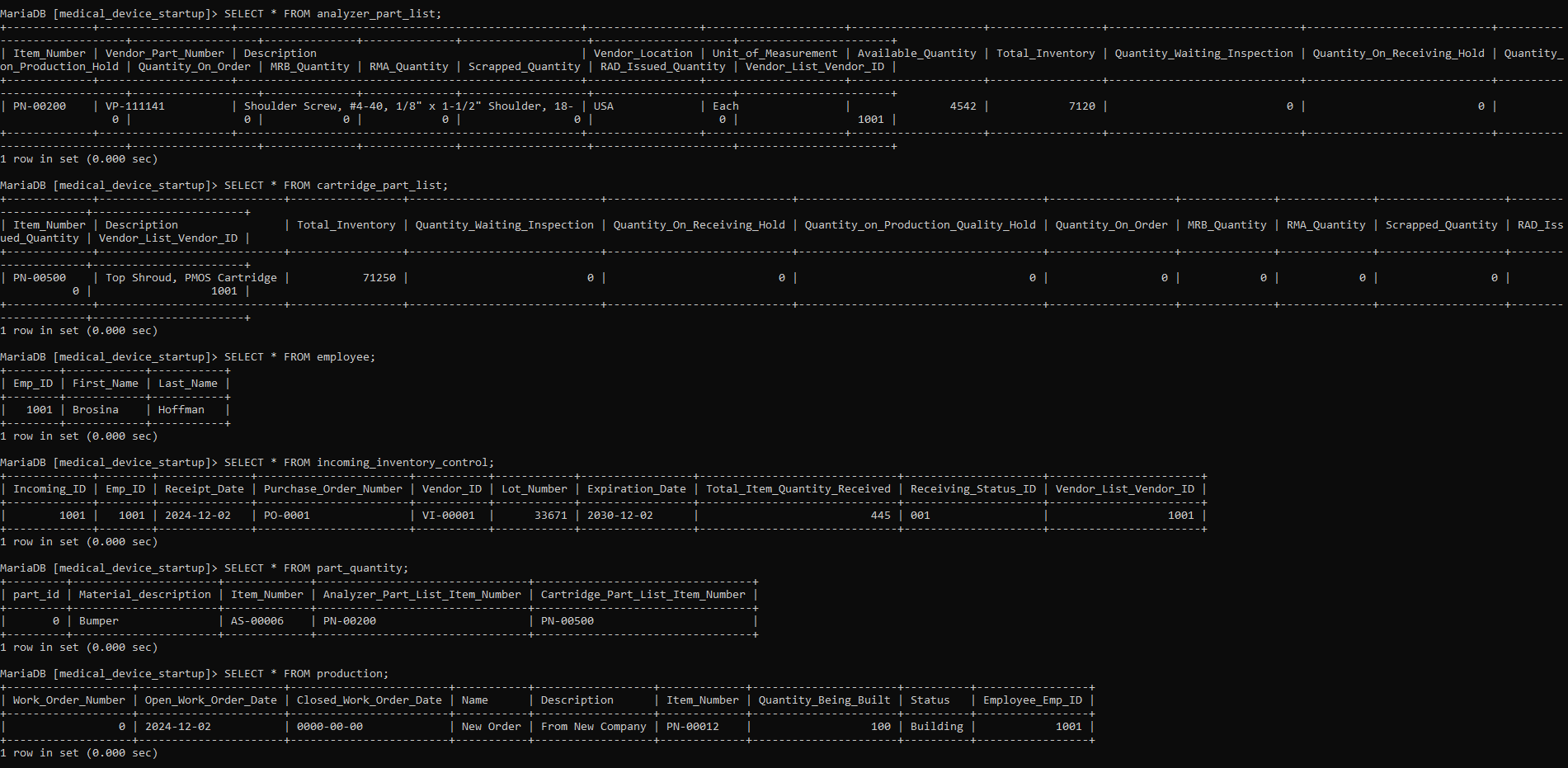
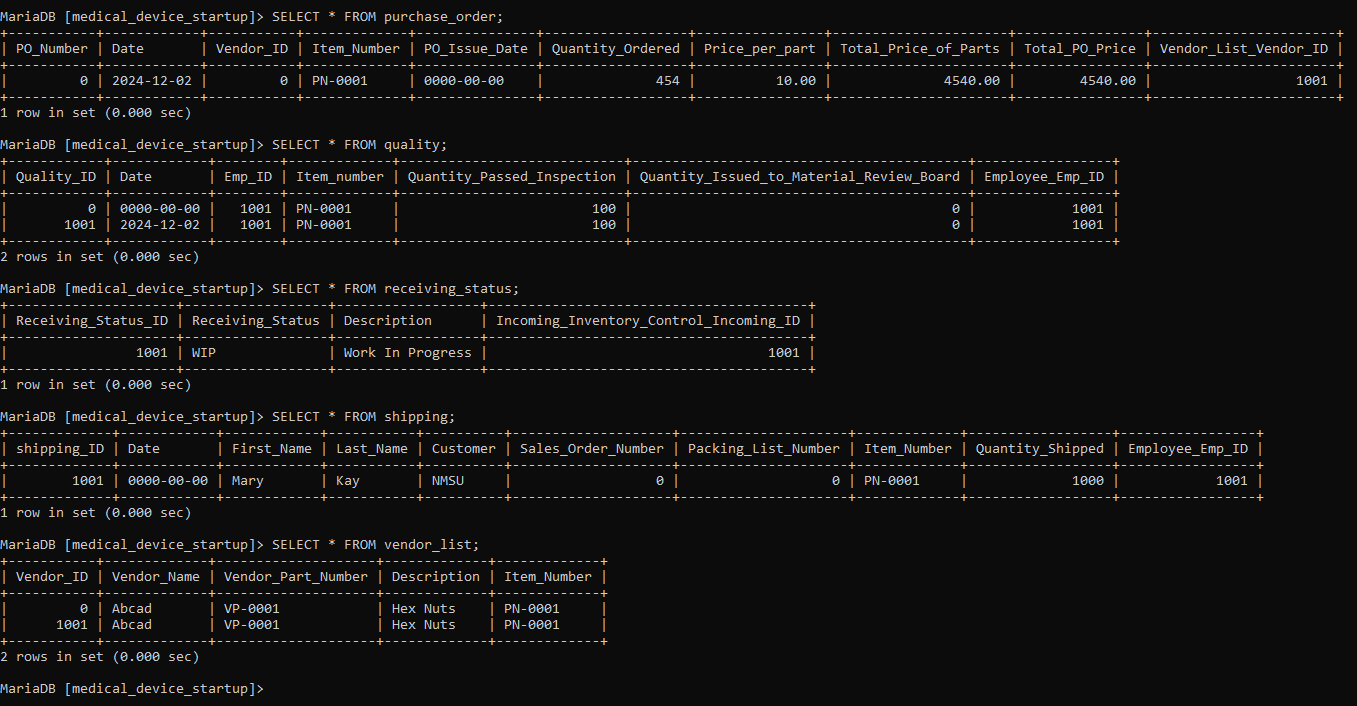
SELECT \* FROM production;

SELECT \* FROM purchase\_order;

SELECT \* FROM quality;

SELECT \* FROM receiving\_status;

SELECT \* FROM shipping;

SELECT \* FROM vendor\_list;  

# 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,*

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

***https://github.com/sharath561/Medical-device-database-system/blob/main/H-Data%20Manipulation%20Language%20Script.sql***

--1 point each for the two insert statements

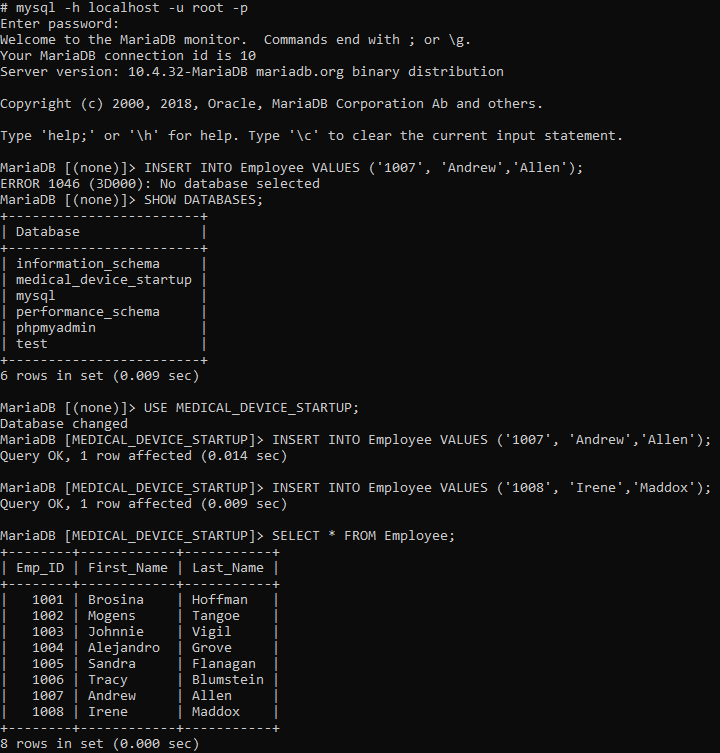
-- Insert statement 1

INSERT INTO Employee VALUES ('1007', 'Andrew','Allen');

-- Insert statement 2

INSERT INTO Employee VALUES ('1008', 'Irene','Maddox');

SELECT \* FROM Employee;

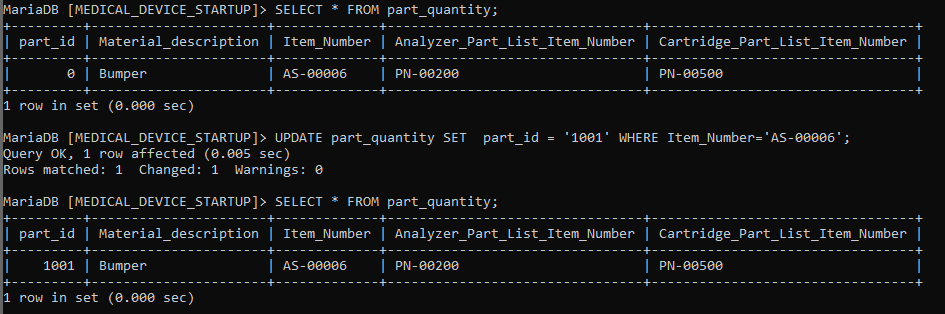


--1 point each for the two update statements

-- Update statement 1

SELECT \* FROM part\_quantity;

UPDATE part\_quantity SET part\_id = '1001' WHERE Item\_Number='AS-00006';

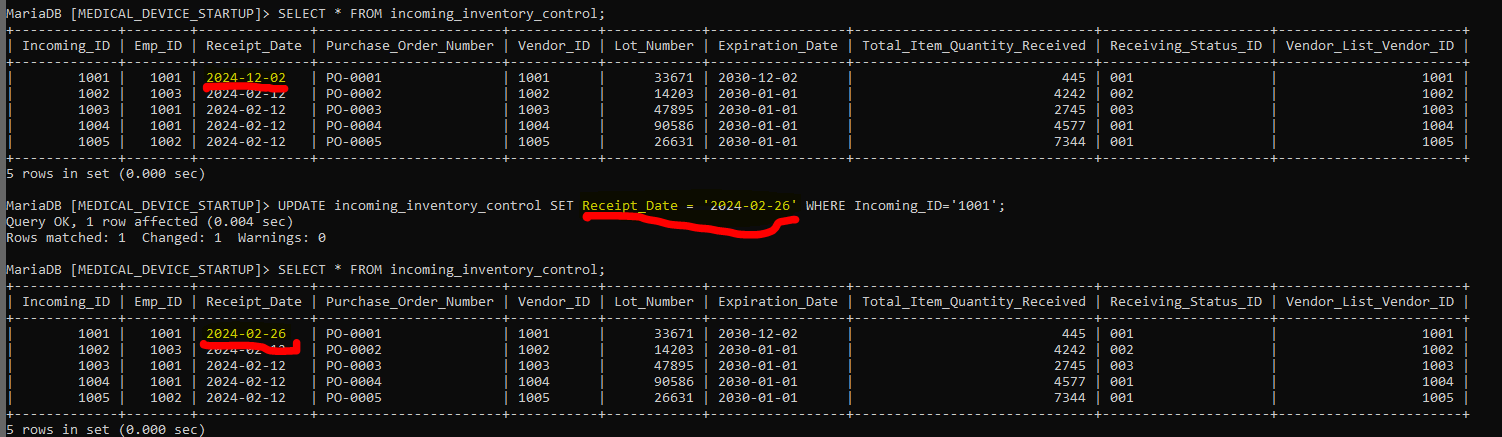
SELECT \* FROM part\_quantity; 

-- Update statement 2

SELECT \* FROM incoming\_inventory\_control;

UPDATE incoming\_inventory\_control SET Receipt\_Date = '2024-02-26' WHERE Incoming\_ID='1001';

SELECT \* FROM incoming\_inventory\_control;



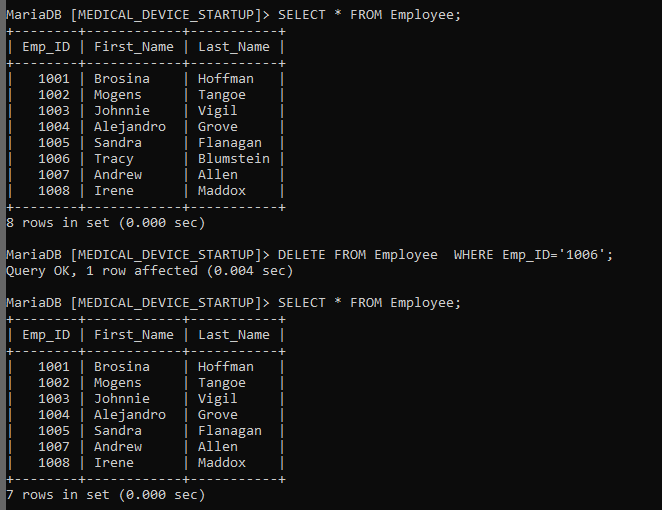
--1 point for the delete statement

-- Delete statement

SELECT \* FROM Employee;

DELETE FROM Employee WHERE Emp\_ID='1006';

SELECT \* FROM Employee;



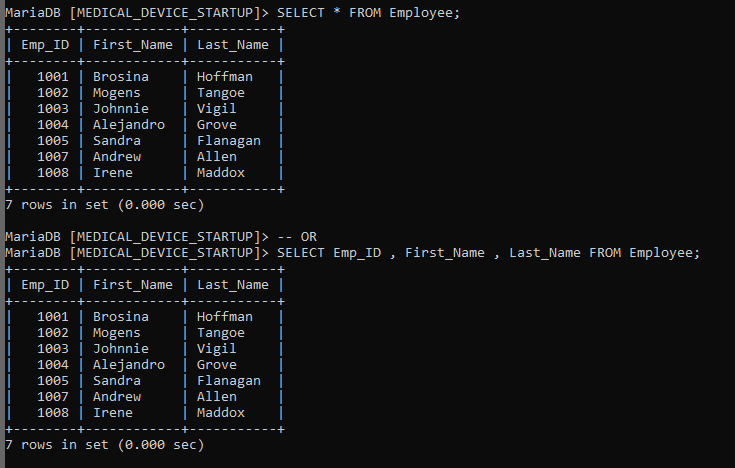
--1 point for the simple select statement

-- Simple select statement

SELECT \* FROM Employee;

-- OR

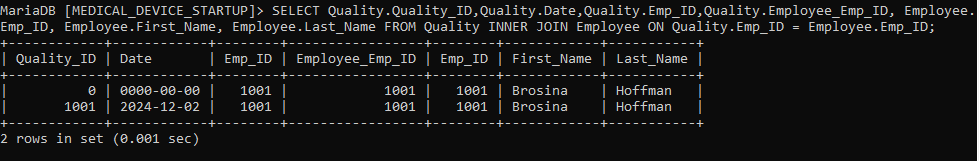
SELECT Emp\_ID , First\_Name , Last\_Name FROM Employee;



--2 points each for the 2 join statements

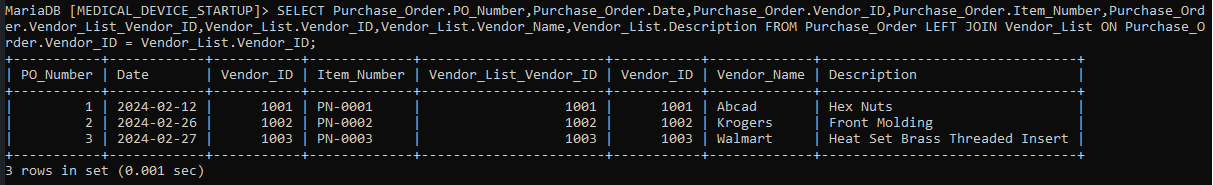
-- Select statement with inner join

SELECT Quality.Quality\_ID,Quality.Date,Quality.Emp\_ID,Quality.Employee\_Emp\_ID, Employee.Emp\_ID, Employee.First\_Name, Employee.Last\_Name FROM Quality INNER JOIN Employee ON Quality.Emp\_ID = Employee.Emp\_ID;



-- Select statement with left join

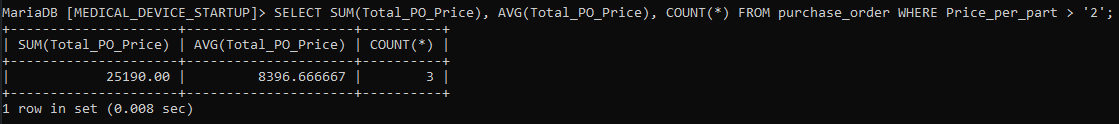
SELECT Purchase\_Order.PO\_Number,Purchase\_Order.Date,Purchase\_Order.Vendor\_ID,Purchase\_Order.Item\_Number,Purchase\_Order.Vendor\_List\_Vendor\_ID,Vendor\_List.Vendor\_ID,Vendor\_List.Vendor\_Name,Vendor\_List.Description FROM Purchase\_Order LEFT JOIN Vendor\_List ON Purchase\_Order.Vendor\_ID = Vendor\_List.Vendor\_ID;



--2 points each for the two that use summary statements

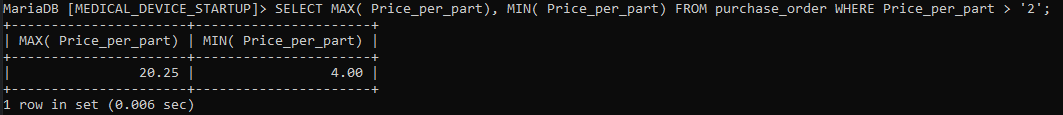
-- Summary function 1

SELECT SUM(Total\_PO\_Price), AVG(Total\_PO\_Price), COUNT(\*) FROM purchase\_order WHERE Price\_per\_part > '2';



-- Summary function 2

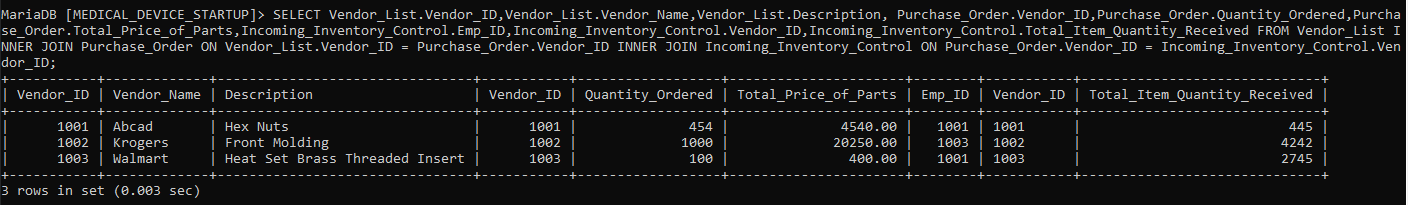
SELECT MAX( Price\_per\_part), MIN( Price\_per\_part) FROM purchase\_order WHERE Price\_per\_part > '2';



--2 points for the multi-table query

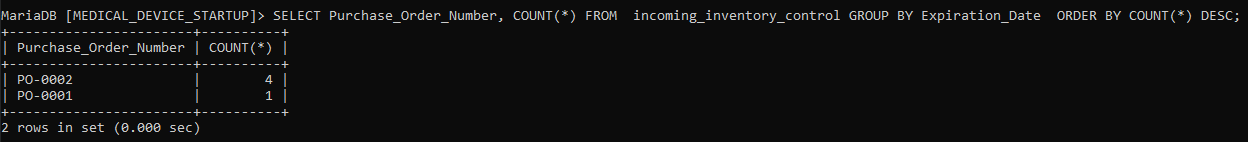
-- Multi-table query

SELECT Vendor\_List.Vendor\_ID,Vendor\_List.Vendor\_Name,Vendor\_List.Description, Purchase\_Order.Vendor\_ID,Purchase\_Order.Quantity\_Ordered,Purchase\_Order.Total\_Price\_of\_Parts,Incoming\_Inventory\_Control.Emp\_ID,Incoming\_Inventory\_Control.Vendor\_ID,Incoming\_Inventory\_Control.Total\_Item\_Quantity\_Received FROM Vendor\_List INNER JOIN Purchase\_Order ON Vendor\_List.Vendor\_ID = Purchase\_Order.Vendor\_ID INNER JOIN Incoming\_Inventory\_Control ON Purchase\_Order.Vendor\_ID = Incoming\_Inventory\_Control.Vendor\_ID;



--2 points for the query of your choice.

-- Your chosen query

SELECT Purchase\_Order\_Number, COUNT(\*) FROM incoming\_inventory\_control GROUP BY Expiration\_Date ORDER BY COUNT(\*) DESC; 

* *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.*

*The SQL Statements are neatly written in the .sql file and been uploaded to Git. Just for the sake of convenience I have combined all the statement in one line.*

*Total points possible: 24*

# 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*

* *3 points for clearly defining at least three indexes and explaining why you chose them.*

EXPLAIN SELECT Vendor\_List.Vendor\_ID,Vendor\_List.Vendor\_Name,Vendor\_List.Description, Purchase\_Order.Vendor\_ID,Purchase\_Order.Quantity\_Ordered,Purchase\_Order.Total\_Price\_of\_Parts,Incoming\_Inventory\_Control.Emp\_ID,Incoming\_Inventory\_Control.Vendor\_ID,Incoming\_Inventory\_Control.Total\_Item\_Quantity\_Received FROM Vendor\_List INNER JOIN Purchase\_Order ON Vendor\_List.Vendor\_ID = Purchase\_Order.Vendor\_ID INNER JOIN Incoming\_Inventory\_Control ON Purchase\_Order.Vendor\_ID = Incoming\_Inventory\_Control.Vendor\_ID;

SHOW INDEXES FROM Vendor\_List;

SHOW INDEXES FROM Purchase\_Order;

SHOW INDEXES FROM Incoming\_Inventory\_Control;

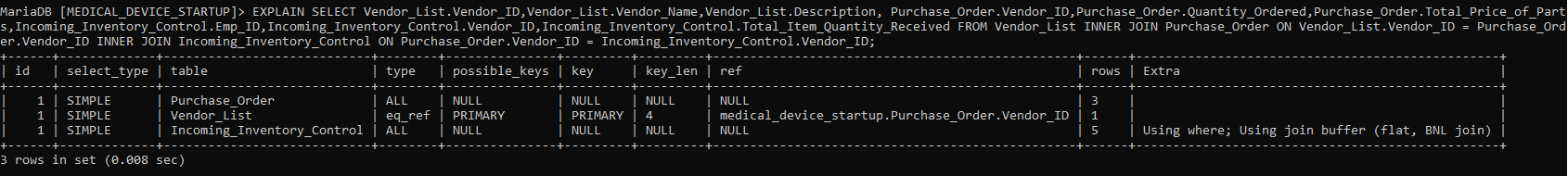
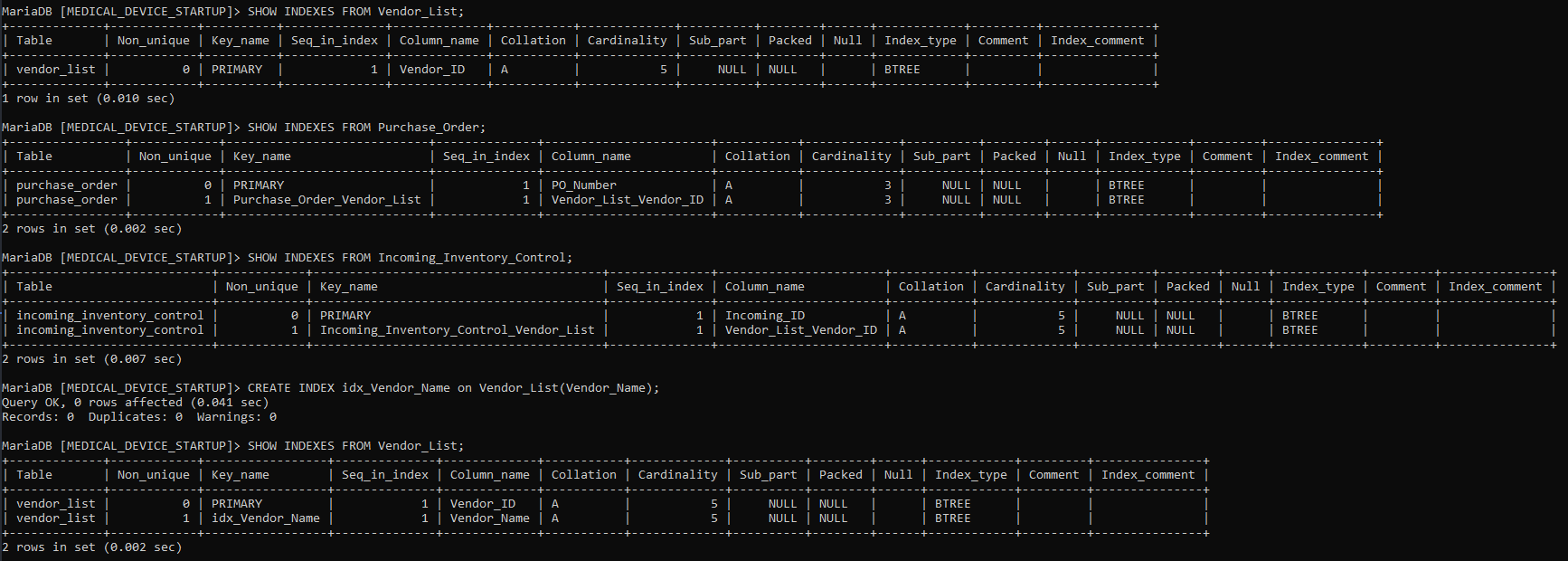
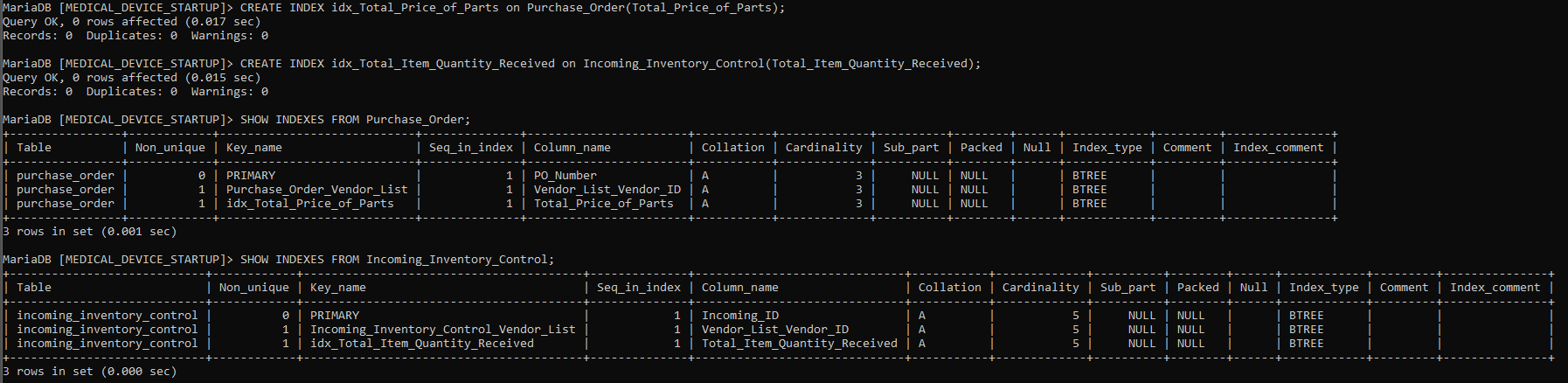
CREATE INDEX idx\_Vendor\_Name on Vendor\_List(Vendor\_Name);

CREATE INDEX idx\_Total\_Price\_of\_Parts on Purchase\_Order(Total\_Price\_of\_Parts);

CREATE INDEX idx\_Total\_Item\_Quantity\_Received on Incoming\_Inventory\_Control(Total\_Item\_Quantity\_Received);

SHOW INDEXES FROM Purchase\_Order;

SHOW INDEXES FROM Incoming\_Inventory\_Control;

**Indexes Definition and Explanation:**

a. Index on Vendor\_Name in the Vendor\_List table:

Reason: We often query vendor information by their names, such as when displaying vendor details or searching for specific vendors.

b. Index on Total\_Price\_of\_Parts in the Purchase\_Order table:

Reason: Orders are frequently analyzed based on their total price, especially for financial reporting and budgeting purposes.

c. Index on Total\_Item\_Quantity\_Received in the Incoming\_Inventory\_Control table:

Reason: Monitoring inventory levels and tracking received quantities are common tasks, so having an index on this column can improve the performance of related queries.

**Explanation:**

The chosen indexes target columns frequently used in queries, aiming to speed up data retrieval operations. Indexes on vendor name, order total price, and received item quantities address common querying patterns within the respective tables.

By indexing these columns, we reduce the time required to locate and retrieve relevant data, improving overall database performance.

* *3 points for showing the sql needed to generate the indexes*

SHOW INDEXES FROM Vendor\_List;

SHOW INDEXES FROM Purchase\_Order;

SHOW INDEXES FROM Incoming\_Inventory\_Control;

CREATE INDEX idx\_Vendor\_Name on Vendor\_List(Vendor\_Name);

CREATE INDEX idx\_Total\_Price\_of\_Parts on Purchase\_Order(Total\_Price\_of\_Parts);

CREATE INDEX idx\_Total\_Item\_Quantity\_Received on Incoming\_Inventory\_Control(Total\_Item\_Quantity\_Received);

SHOW INDEXES FROM Purchase\_Order;

SHOW INDEXES FROM Incoming\_Inventory\_Control;

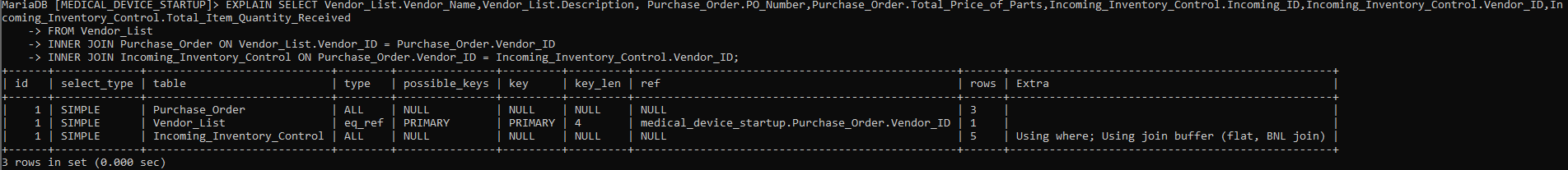
* *2 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

EXPLAIN SELECT Vendor\_List.Vendor\_Name,Vendor\_List.Description, Purchase\_Order.PO\_Number,Purchase\_Order.Total\_Price\_of\_Parts,Incoming\_Inventory\_Control.Incoming\_ID,Incoming\_Inventory\_Control.Vendor\_ID,Incoming\_Inventory\_Control.Total\_Item\_Quantity\_Received

FROM Vendor\_List

INNER JOIN Purchase\_Order ON Vendor\_List.Vendor\_ID = Purchase\_Order.Vendor\_ID

INNER JOIN Incoming\_Inventory\_Control ON Purchase\_Order.Vendor\_ID = Incoming\_Inventory\_Control.Vendor\_ID;



By Comparing the Total rows scanned before and after indexing. Here my tables have miniscule data so there is not significant improvement in the performance. Indexing need to be done on column which is frequently queried.

# 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:*

* *2 points for including the SQL for generating the two views in your Word document*

-- View 1: Combined data from Vendor\_List and Purchase\_Order tables

CREATE VIEW Vendor\_Purchase\_View AS

SELECT v.Vendor\_ID, v.Vendor\_Name, v.Description, p.PO\_Number, p.Quantity\_Ordered, p.Price\_per\_part

FROM Vendor\_List v

INNER JOIN Purchase\_Order p ON v.Vendor\_ID = p.Vendor\_ID;

-- View 2: Combined data from Incoming\_Inventory\_Control and Receiving\_Status

CREATE VIEW Incoming\_Receiving\_View AS

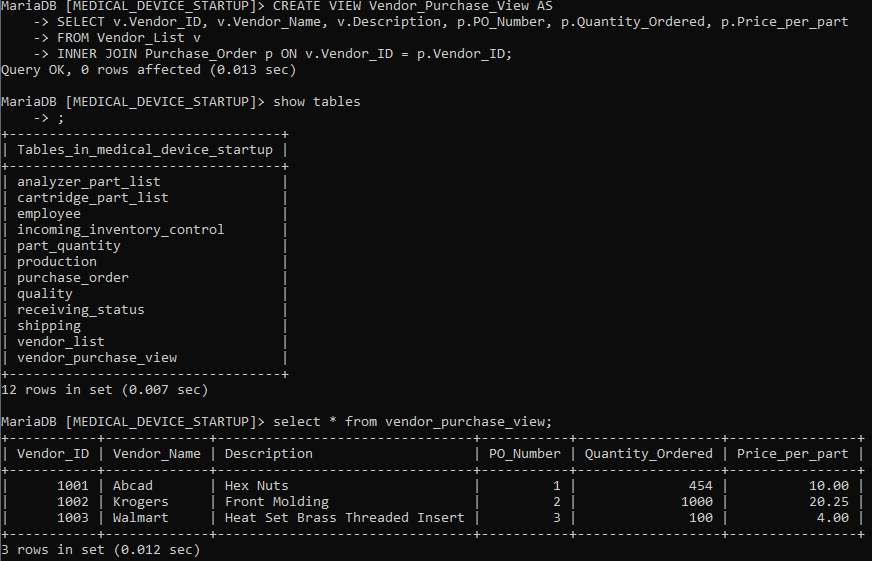
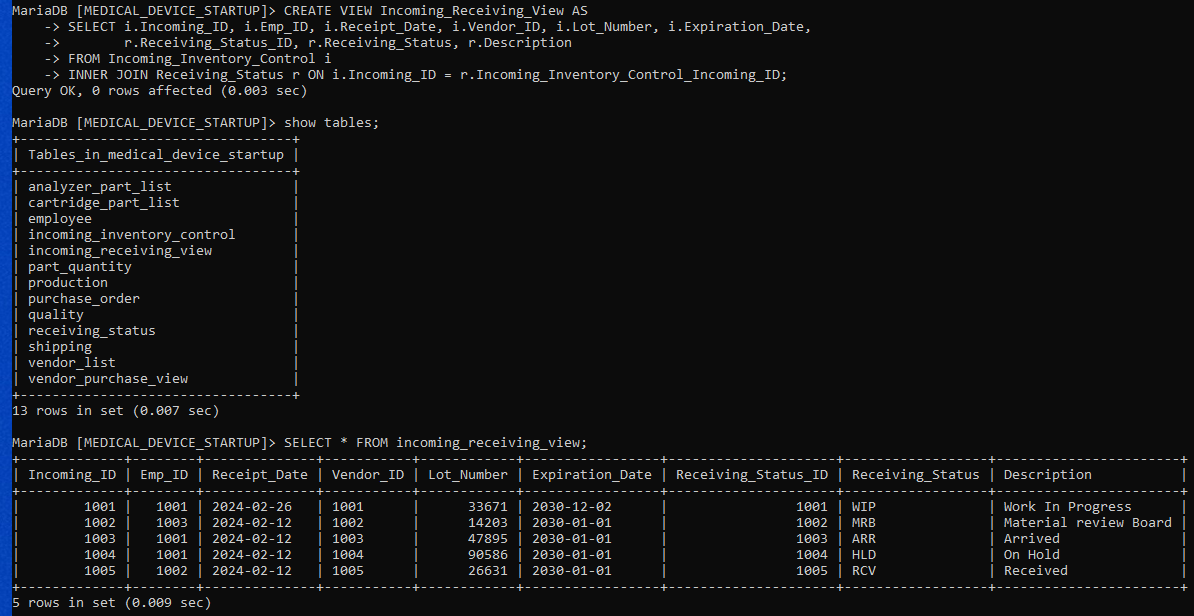
SELECT i.Incoming\_ID, i.Emp\_ID, i.Receipt\_Date, i.Vendor\_ID, i.Lot\_Number, i.Expiration\_Date,

       r.Receiving\_Status\_ID, r.Receiving\_Status, r.Description

FROM Incoming\_Inventory\_Control i

INNER JOIN Receiving\_Status r ON i.Incoming\_ID = r.Incoming\_Inventory\_Control\_Incoming\_ID;

* *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*

**Explanation of Views:**

**Vendor\_Purchase\_View:** This view provides a consolidated view of vendor information and corresponding purchase orders. It simplifies querying data related to vendor purchases and allows for easy analysis of vendor performance and purchase history.

**Inventory\_Control\_View:** This view integrates information from the Incoming\_Inventory\_Control and Receiving\_Status tables, providing a comprehensive overview of incoming inventory and its receiving status. By merging data from these tables, end users gain insights into the arrival of inventory, its handling status, and any pertinent remarks associated with its receipt. The view streamlines the process of monitoring inventory reception and managing stock quantities efficiently. With this consolidated view, users can promptly evaluate the status of incoming items and make well-informed decisions regarding inventory management strategies.

By creating these views, I enhance the accessibility and usability of database by providing end users with pre-defined combinations of data from multiple tables. These views streamline data retrieval processes and enable users to obtain valuable insights without the complexity of writing intricate SQL queries every time.

*Total points possible: 6*

# 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:*

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

*Total points possible: 6*

• The SQL for the stored trigger in your Word document

USE medical\_device\_startup;

SELECT \* FROM incoming\_inventory\_control;

SET @Total\_Item\_Quantity\_Received\_Whole = (SELECT SUM(Total\_Item\_Quantity\_Received) FROM Incoming\_Inventory\_Control);

SELECT @Total\_Item\_Quantity\_Received\_Whole;

DELIMITER //

CREATE TRIGGER Total\_Quantity BEFORE INSERT ON Incoming\_Inventory\_Control FOR EACH ROW

SET @Total\_Item\_Quantity\_Received\_Whole =  @Total\_Item\_Quantity\_Received\_Whole + NEW.Total\_Item\_Quantity\_Received //

DELIMITER ;

SELECT SUM(Total\_Item\_Quantity\_Received) FROM Incoming\_Inventory\_Control;

INSERT INTO Incoming\_Inventory\_Control VALUES('1006','1002','2024-03-12','PO-0005','1005','26631','2030-01-01','7344','001','1005');

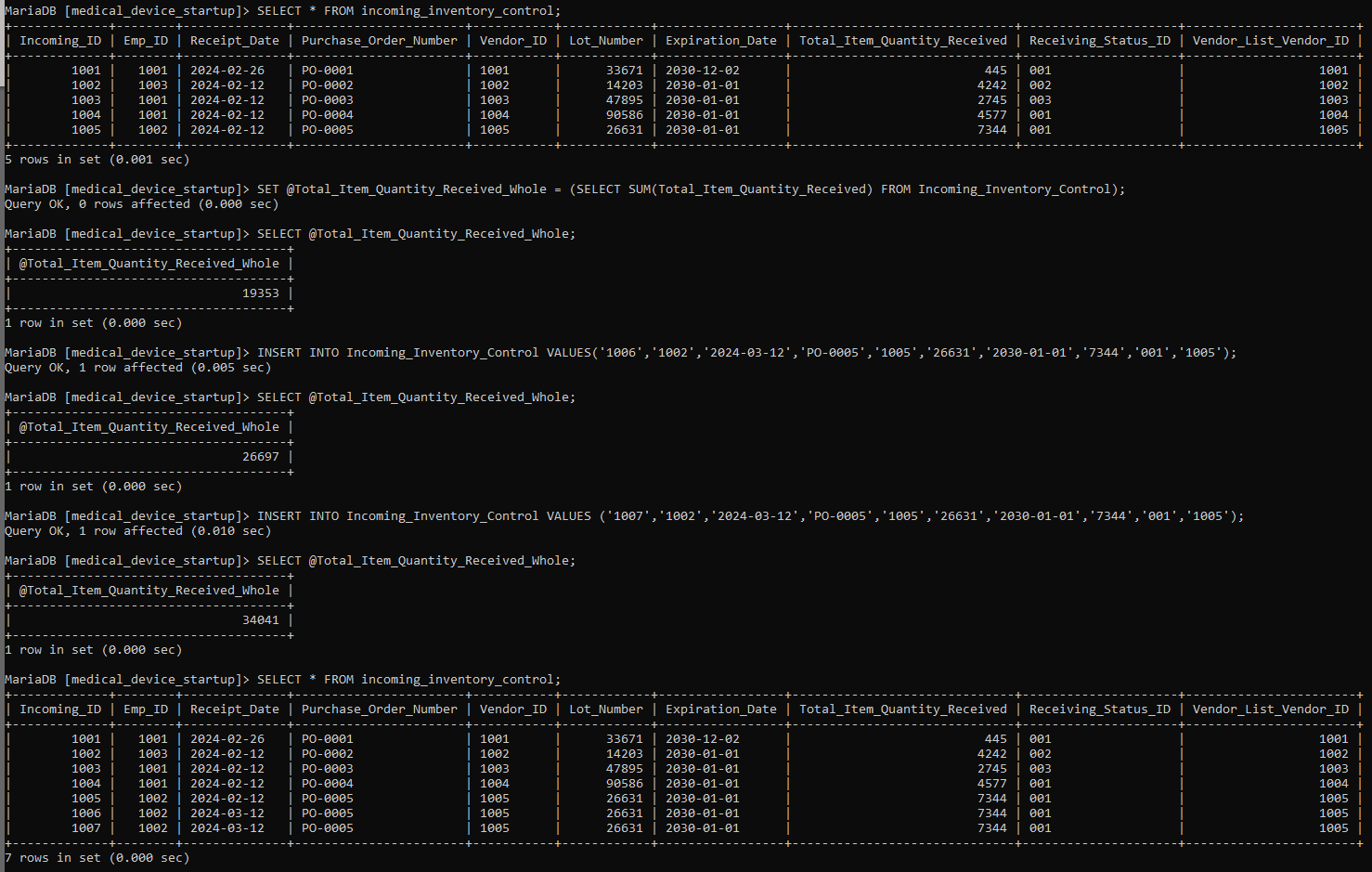
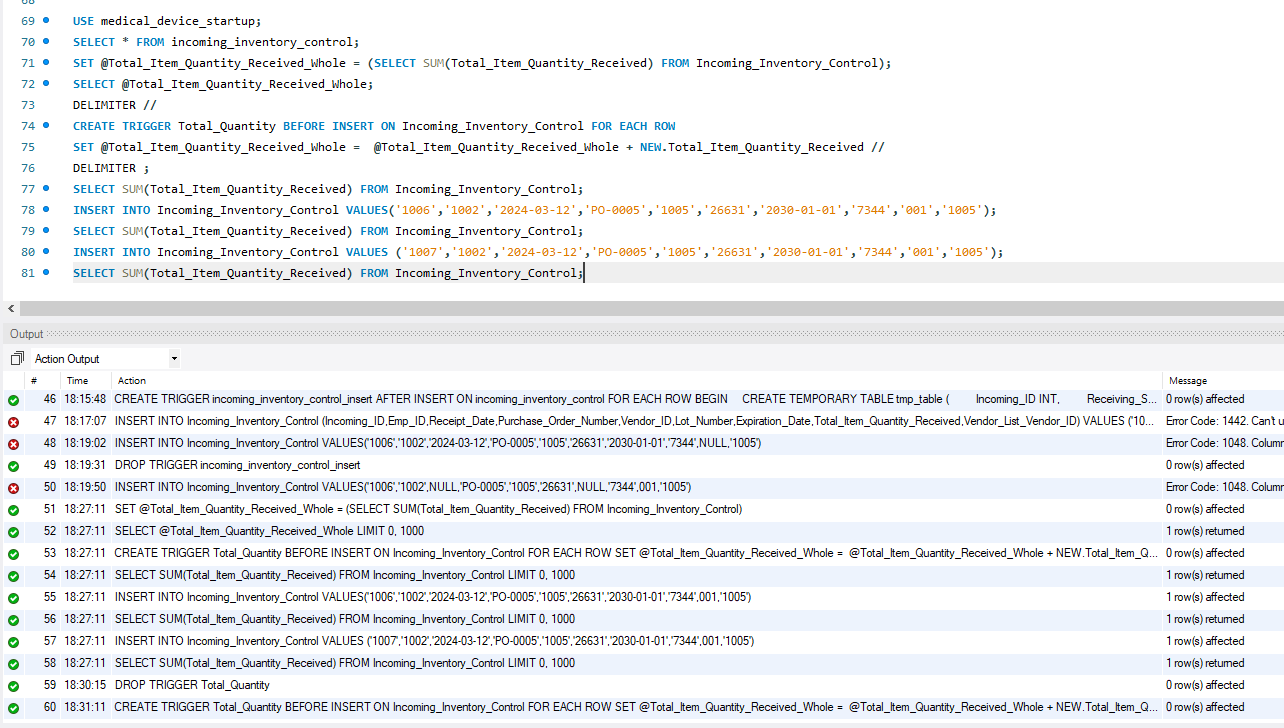
SELECT SUM(Total\_Item\_Quantity\_Received) FROM Incoming\_Inventory\_Control;

INSERT INTO Incoming\_Inventory\_Control VALUES ('1007','1002','2024-03-12','PO-0005','1005','26631','2030-01-01','7344','001','1005');

SELECT SUM(Total\_Item\_Quantity\_Received) FROM Incoming\_Inventory\_Control;

• Explaining the purpose of the stored program

The purpose of this trigger is to update the total quantity available in stock in the Inventory table automatically whenever a new row is inserted into the table. It ensures that the inventory levels are accurately reflected in real-time based on the row inserted.

• Screenshot and explanation that shows the stored program in action.  

In the screenshot, we can see that when a new row has been inserted into the table. After the insertion, the trigger Total\_Quantity is activated, which then updates the quantity in stock in the Inventory table based on the Total\_Item\_Quantity\_Received. This demonstrates how the trigger automates the process of updating inventory levels based on incoming orders.

# 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:*

* *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*

* Explain the importance of transactions to ensuring ACID behavior

Transactions group together multiple statements so that they are either all done, or

none are done and are essential in databases to ensure ACID behavior, which stands for Atomicity, Consistency, Isolation, and Durability.

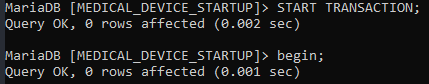
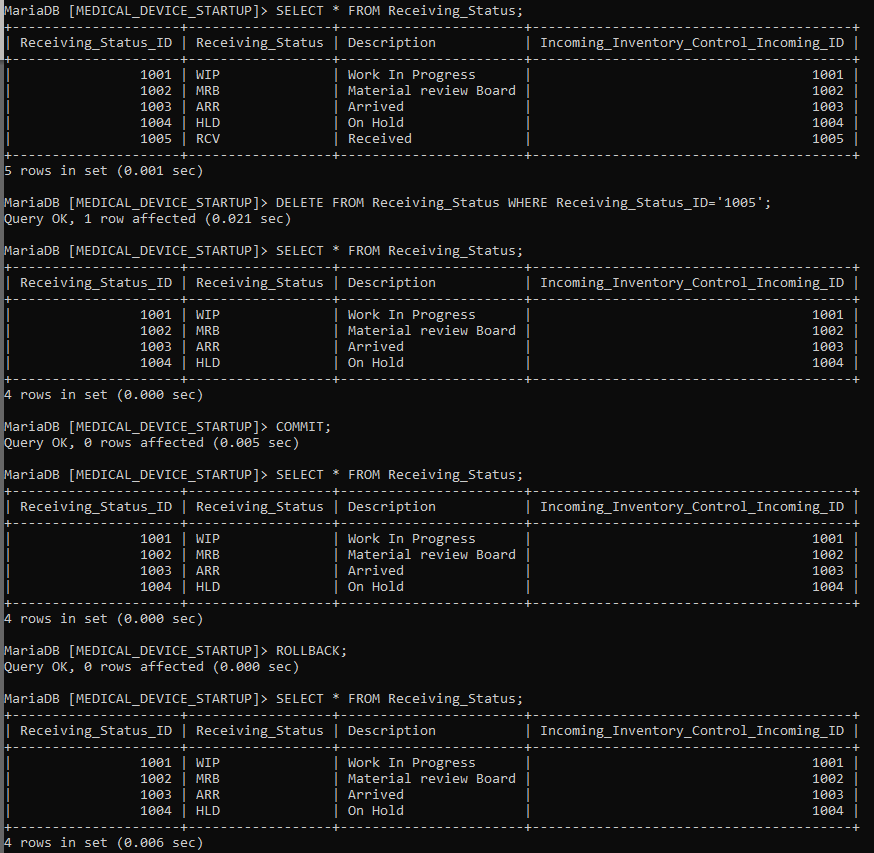
**Atomicity:** Transactions ensure that all operations within the transaction are completed successfully, or none of them are. This prevents partial updates that could leave the database in an inconsistent state.

**Consistency:** Transactions maintain the integrity of the database by enforcing rules and constraints, ensuring that it moves from one valid state to another.

**Isolation:** Transactions run independently of each other, preventing interference between concurrent transactions. This ensures that each transaction sees a consistent view of the database.

**Durability:** Once a transaction is committed, its changes are permanently stored in the database, even in the event of a system failure.

**Screenshot and Explanation:** Here's a screenshot demonstrating a MySQL transaction:

In this transaction, we begin by starting a transaction with the START TRANSACTION OR BEGIN; statement. Then, we perform several SQL operations, such as deleting data. Finally, we commit the transaction using COMMIT, which makes all changes permanent. If an error occurs or if we need to cancel the changes, we can use ROLLBACK to revert the database to its state before the transaction started.

Transactions are crucial for maintaining data integrity and ensuring that database operations are reliable and consistent, which is essential for the smooth functioning of applications and systems relying on the database.

# 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*

* 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.

In our database system, we have identified three types of users and their corresponding privileges:

**Administrators:** These users have full control over the database system. They should have privileges to create, modify, and delete databases, tables, and users. They should also be able to execute any SQL command.

**Data Analysts:** These users primarily work with data analysis and reporting. They require read access to all tables and databases but should not have privileges to modify or delete data.

**Application Users:** These users interact with the database through applications. They require specific privileges based on the operations the application performs, such as INSERT, UPDATE, DELETE, and SELECT.

* 4 points for writing GRANT statements that assign privileges to these different kinds of users.

--Security

CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin';

CREATE USER 'analyst'@'localhost' IDENTIFIED BY 'analyst';

CREATE USER 'app\_user'@'localhost' IDENTIFIED BY 'app';

SELECT USER, AUTHENTICATION\_STRING, HOST FROM MYSQL.USER;

GRANT ALL PRIVILEGES ON \*.\* TO 'admin'@'localhost' WITH GRANT OPTION;

SHOW GRANTS FOR 'admin'@'localhost';

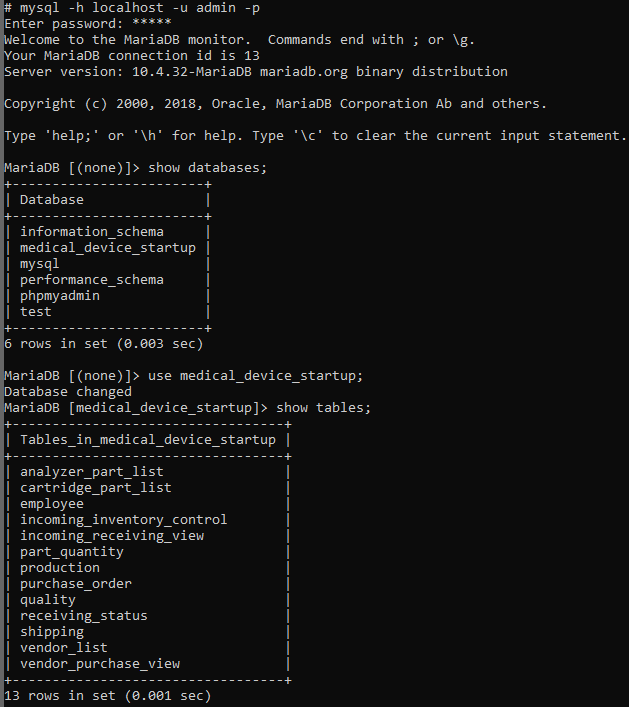
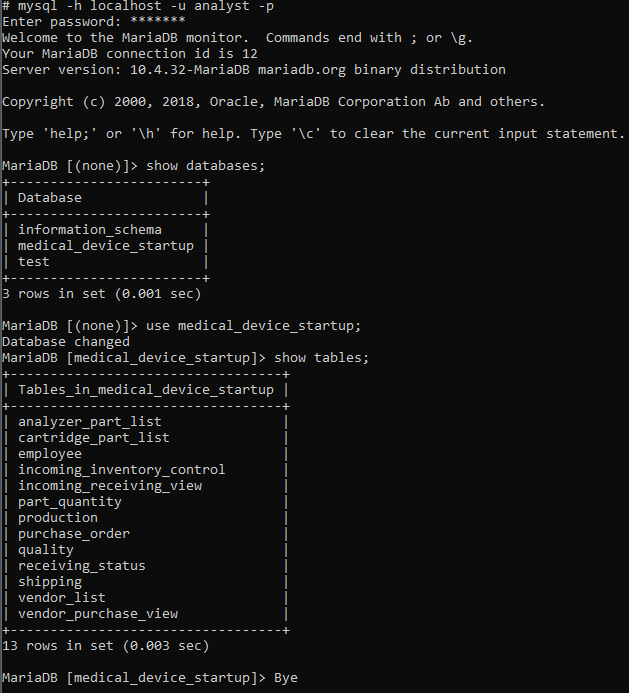
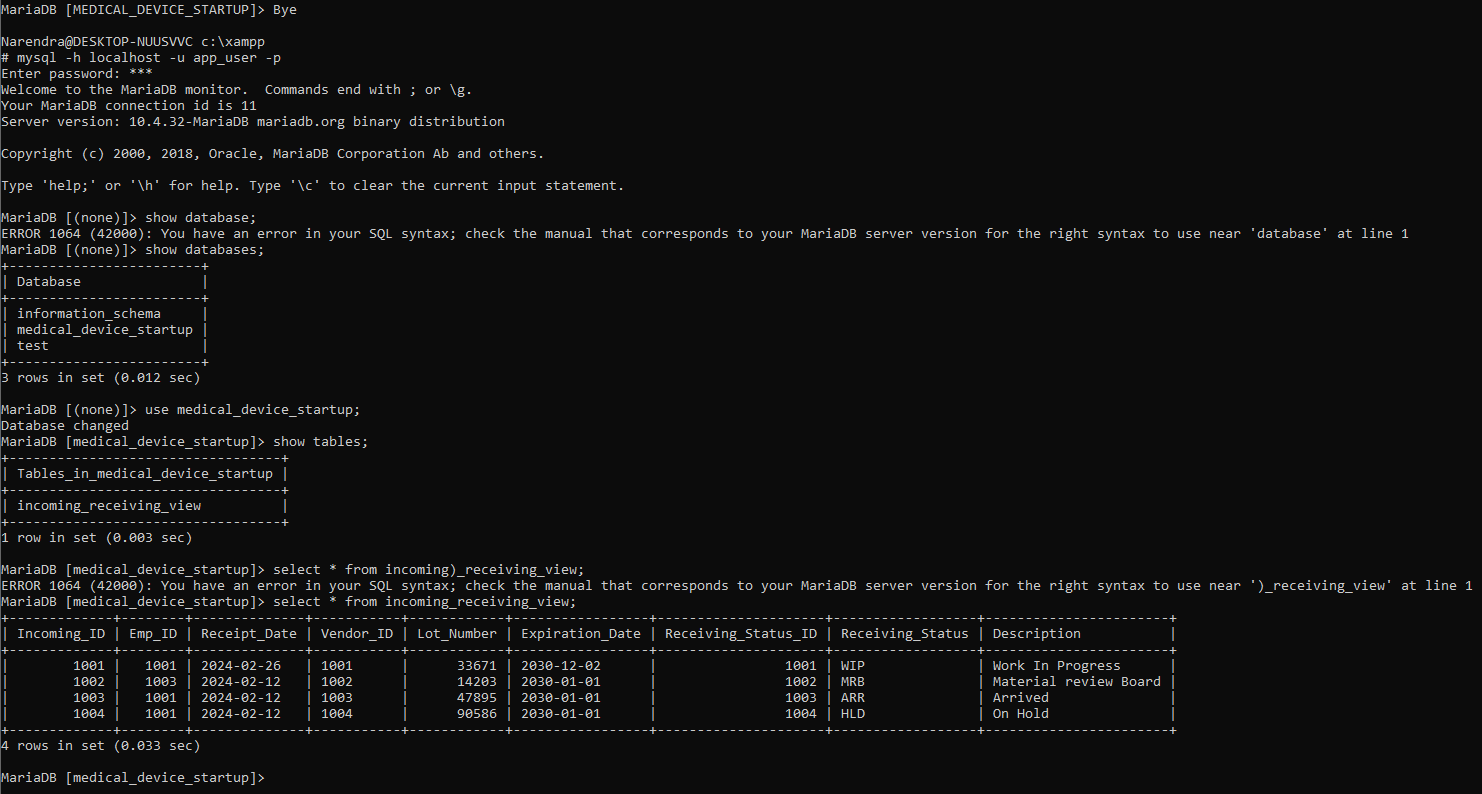
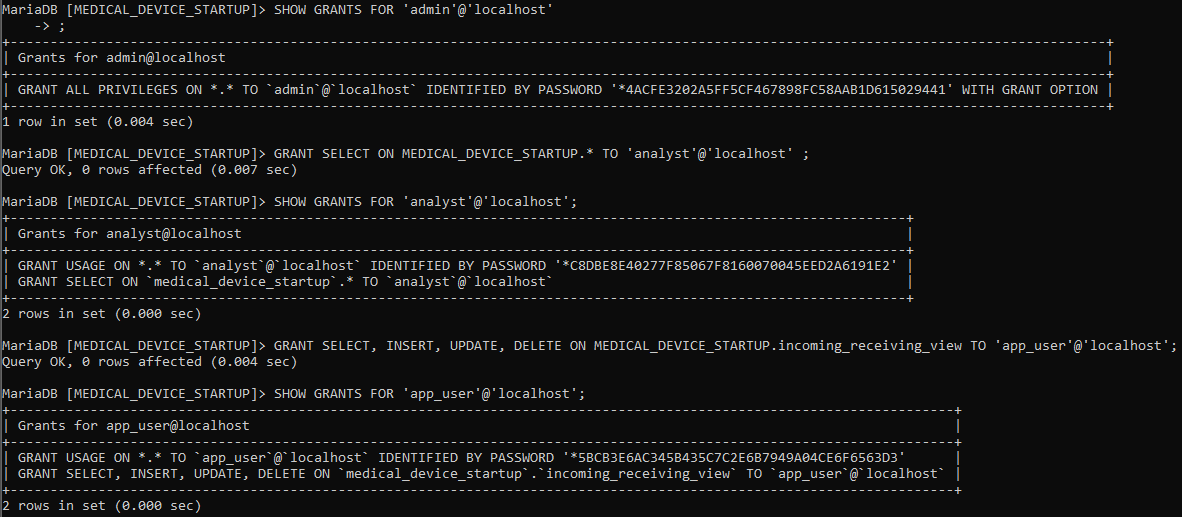
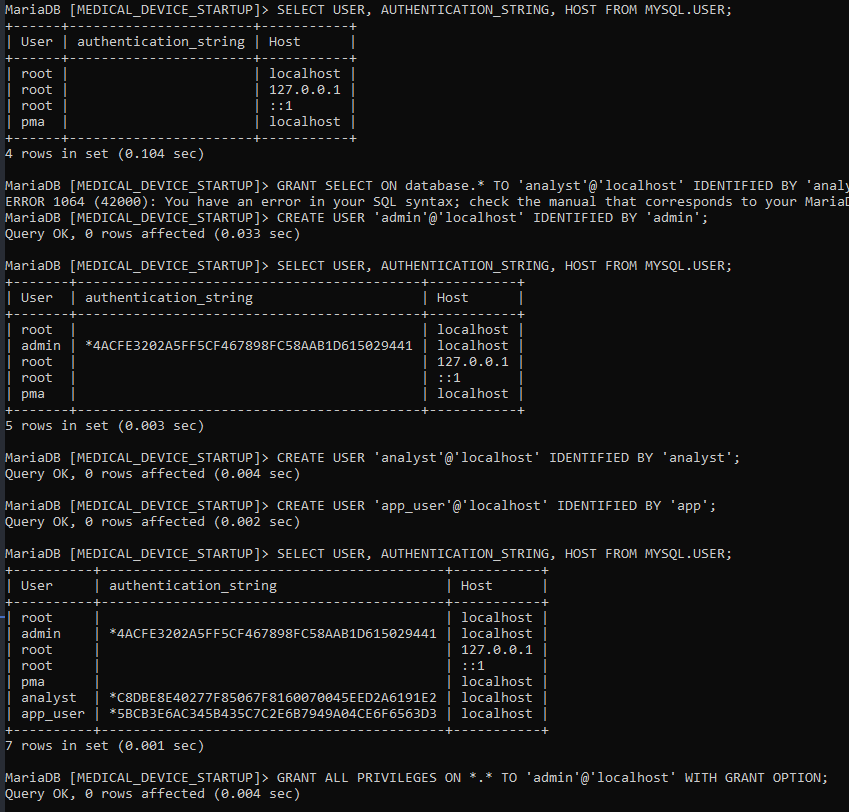
GRANT SELECT ON MEDICAL\_DEVICE\_STARTUP.\* TO 'analyst'@'localhost' ;

SHOW GRANTS FOR 'analyst'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE ON MEDICAL\_DEVICE\_STARTUP.incoming\_receiving\_view TO 'app\_user'@'localhost';

SHOW GRANTS FOR 'app\_user'@'localhost';

* 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.



# 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.*
* *2 points for providing a screenshot and accompanying explanation of locking tables.*

*Total points possible: 5*

ENTER YOUR WORK WITH LOCKING AND CONCURRENT ACCESS HERE

* for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.

Locking tables is crucial to maintain data consistency, especially in scenarios where concurrent transactions take place. Considering a scenario where multiple users simultaneously attempt to update the same data in a table. Without proper locking mechanisms, inconsistencies can occur, leading to data corruption or incorrect results.

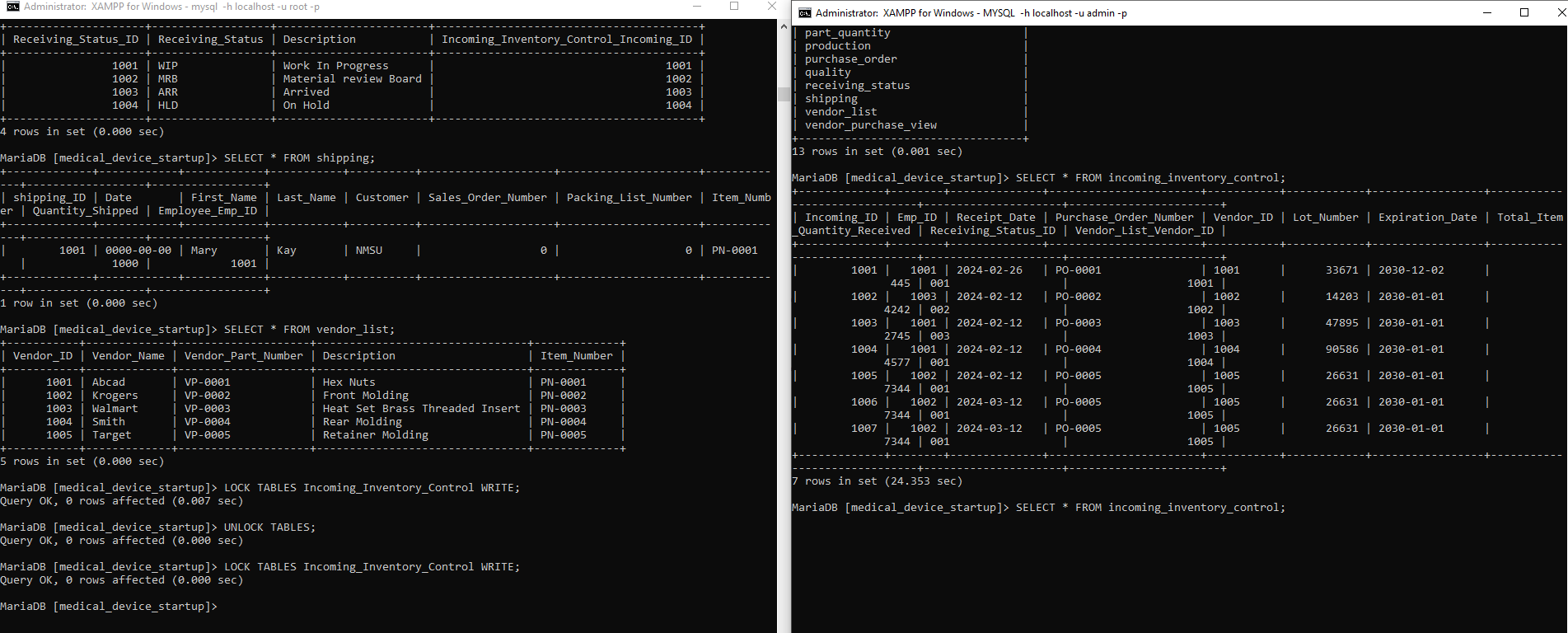
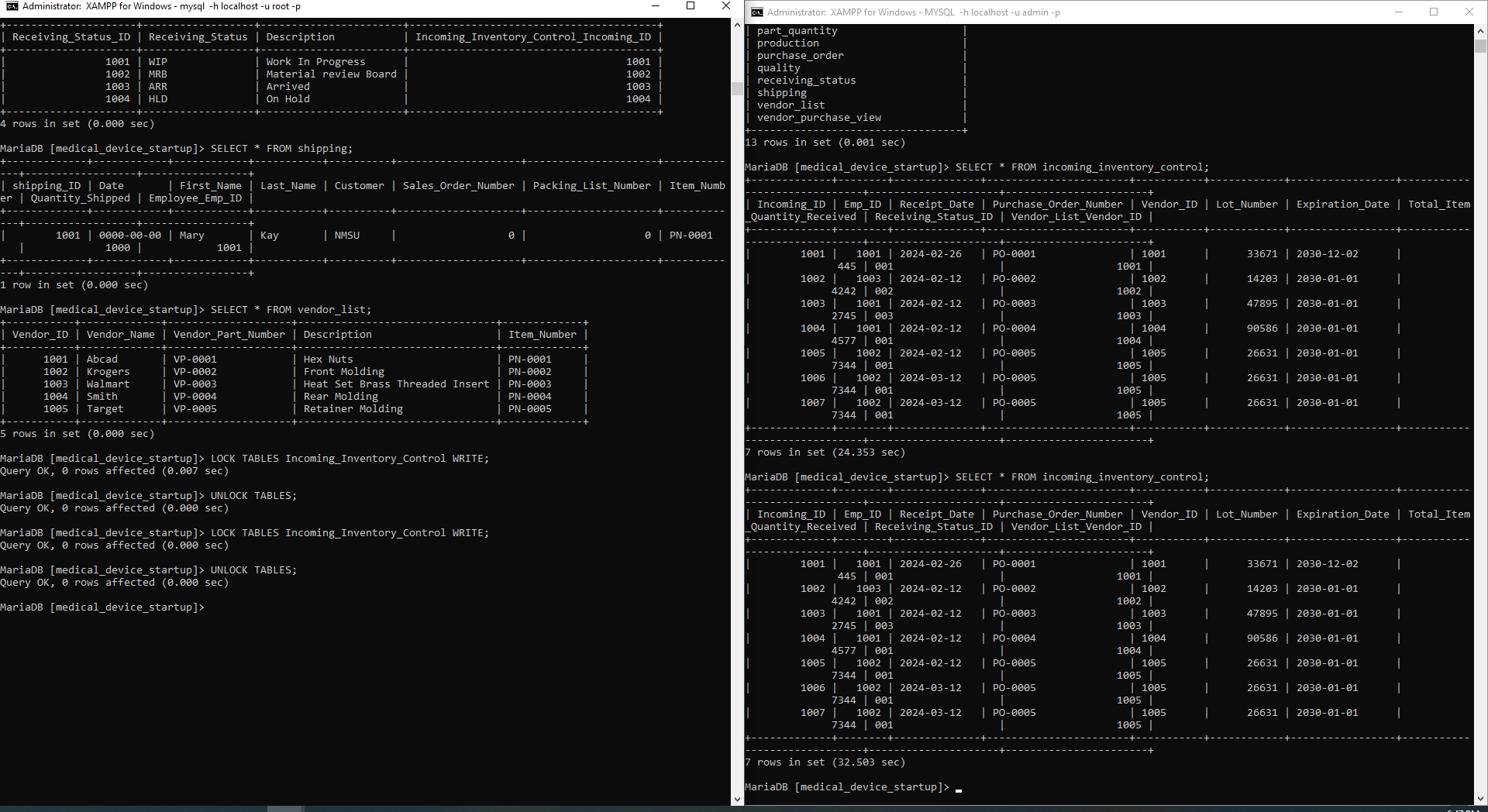
Let's consider a scenario where two users attempt to update the same inventory item's quantity at the same time. Without table locking, both users might retrieve the current quantity, perform their updates, and then save the updated quantity. However, if one user's update process completes before the other, it could overwrite the changes made by the other user, resulting in data inconsistency.

* for providing a screenshot and accompanying explanation of locking tables.

To prevent such inconsistencies, we can use table-level locking mechanisms in MySQL. For instance, we can use the LOCK TABLES statement to explicitly lock tables during critical operations. Here's an example:

LOCK TABLES Inventory WRITE;

In the screenshot below, we can observe that the Inventory table has been locked with a write lock by a root user. This prevents other users admin from displaying the table until the lock is released.

By employing table locking, we ensure that only one user can make changes to the table at a time, preventing inconsistencies and maintaining data integrity in scenarios with concurrent transactions.

# 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*

* 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)

Database backup is crucial for ensuring data integrity, security, and disaster recovery. Here's my strategy for backing up the database:

**Frequency of Backups:** Daily backups ensure minimal data loss, while incremental backups capture frequent changes without putting excessive load on the system.

1. Perform daily backups to ensure minimal data loss in case of failure.
2. Additionally, take incremental backups throughout the day for frequent changes.

**Automation of Backups:** Automating backups with cron jobs reduces manual effort and ensures consistency and reliability in the backup process.

1. Using cron jobs or scheduling tools like AutoMySQLBackup, phpMyBackupPro and SQLBackupAndFTP to automate the backup process.
2. Schedule full backups daily and incremental backups hourly or as per the frequency of changes or as per the business requirements.

**Backup Storage Location:** Storing backups in a separate location ensures data availability in case of disasters like hardware failures, accidents, or cyberattacks.

1. Storing backups in a separate physical or cloud storage location like S3 Glacier Deep Archive to prevent data loss due to hardware failures or disasters.
2. Utilize redundant storage solutions to ensure high availability and reliability of backups.

**Security of Backups:** Encryption and access controls safeguard backup files from unauthorized access, ensuring data confidentiality and integrity.

1. Encrypting backup files to prevent unauthorized access to sensitive data.
2. Implement access controls and authentication mechanisms to restrict access to backup files.

* 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.

Command to Issue for Backup:

mysqldump -u root -p medical\_device\_startup > medical\_device\_startup\_backup.sql

This command creates a backup of the medical\_device\_startup and saves it to a file named medical\_device\_startup\_backup.sql.

Here's a portion of the resulting backup file:

-- MariaDB dump 10.19  Distrib 10.4.32-MariaDB, for Win64 (AMD64)

--

-- Host: localhost    Database: medical\_device\_startup

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

-- Server version 10.4.32-MariaDB

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `analyzer\_part\_list`

--

DROP TABLE IF EXISTS `analyzer\_part\_list`;

/\*!40101 SET @saved\_cs\_client     = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `analyzer\_part\_list` (

  `Item\_Number` varchar(8) NOT NULL,

  `Vendor\_Part\_Number` varchar(10) NOT NULL,

  `Description` varchar(50) NOT NULL,

  `Vendor\_Location` varchar(10) NOT NULL,

  `Unit\_of\_Measurement` varchar(8) NOT NULL,

  `Available\_Quantity` int(11) NOT NULL,

  `Total\_Inventory` int(11) NOT NULL,

  `Quantity\_Waiting\_Inspection` int(11) NOT NULL,

  `Quantity\_On\_Receiving\_Hold` int(11) NOT NULL,

  `Quantity\_on\_Production\_Hold` int(11) NOT NULL,

  `Quantity\_On\_Order` int(11) NOT NULL,

  `MRB\_Quantity` int(11) NOT NULL,

  `RMA\_Quantity` int(11) NOT NULL,

  `Scrapped\_Quantity` int(11) NOT NULL,

  `RAD\_Issued\_Quantity` int(11) NOT NULL,

  `Vendor\_List\_Vendor\_ID` int(11) NOT NULL,

  PRIMARY KEY (`Item\_Number`),

  KEY `Analyzer\_Part\_List\_Vendor\_List` (`Vendor\_List\_Vendor\_ID`),

  CONSTRAINT `Analyzer\_Part\_List\_Vendor\_List` FOREIGN KEY (`Vendor\_List\_Vendor\_ID`) REFERENCES `vendor\_list` (`Vendor\_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `analyzer\_part\_list`

--

LOCK TABLES `analyzer\_part\_list` WRITE;

/\*!40000 ALTER TABLE `analyzer\_part\_list` DISABLE KEYS \*/;

INSERT INTO `analyzer\_part\_list` VALUES ('PN-00200','VP-0001','Shoulder Screw, #4-40, 1/8\" x 1-1/2\" Shoulder, 18-','USA','Each',4542,7120,0,0,0,0,0,0,0,0,1001),('PN-00201','VP-0002','Locknut, #4-40 Thread, Nylon Insert, 18-8 SS','USA','Each',2144,5650,0,0,0,0,0,0,0,0,1002),('PN-00202','VP-0003','Cartridge Interface, Nest Flag','USA','Each',4257,4680,0,0,0,0,0,0,0,0,1003),('PN-00203','VP-0004','External Retaining Ring, 12mm','USA','Each',4556,6300,0,0,0,0,0,0,0,0,1004),('PN-00204','VP-0005','BHCS, M3 x 0.5mm, 5mm Long, 316 SS','USA','Each',5675,3600,0,0,0,0,0,0,0,0,1005);

/\*!40000 ALTER TABLE `analyzer\_part\_list` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `cartridge\_part\_list`

--

DROP TABLE IF EXISTS `cartridge\_part\_list`;

/\*!40101 SET @saved\_cs\_client     = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `cartridge\_part\_list` (

  `Item\_Number` varchar(8) NOT NULL,

  `Description` varchar(50) NOT NULL,

  `Total\_Inventory` int(11) NOT NULL,

  `Quantity\_Waiting\_Inspection` int(11) NOT NULL,

  `Quantity\_On\_Receiving\_Hold` int(11) NOT NULL,

  `Quantity\_on\_Production\_Quality\_Hold` int(11) NOT NULL,

  `Quantity\_On\_Order` int(11) NOT NULL,

  `MRB\_Quantity` int(11) NOT NULL,

  `RMA\_Quantity` int(11) NOT NULL,

  `Scrapped\_Quantity` int(11) NOT NULL,

  `RAD\_Issued\_Quantity` int(11) NOT NULL,

  `Vendor\_List\_Vendor\_ID` int(11) NOT NULL,

  PRIMARY KEY (`Item\_Number`),

  KEY `Cartridge\_Part\_List\_Vendor\_List` (`Vendor\_List\_Vendor\_ID`),

  CONSTRAINT `Cartridge\_Part\_List\_Vendor\_List` FOREIGN KEY (`Vendor\_List\_Vendor\_ID`) REFERENCES `vendor\_list` (`Vendor\_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

# P. Programming

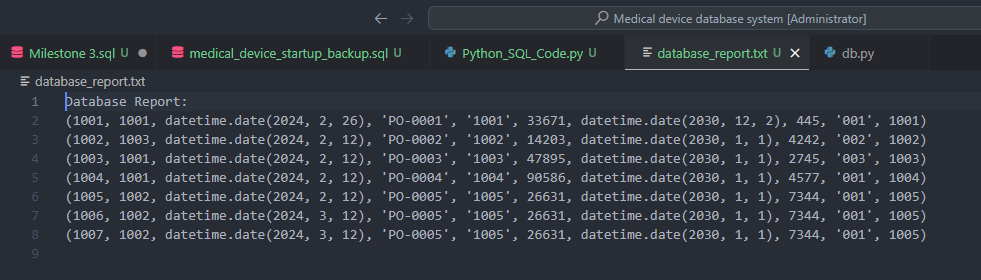
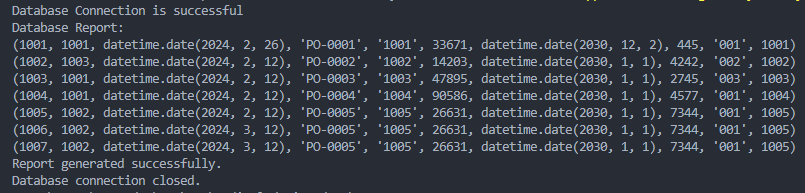
*Description: Write a Python program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document, and also post the program to your GitHub repository.*

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

* *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*

* *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.****Python\_SQL\_Code.py and database\_report.txt files pushed in git***
* import mysql.connector
* # Connect to the MySQL database
* try:
* connection = mysql.connector.connect(
* *host*="localhost", *user*="root", *password*="", *database*="medical\_device\_startup"
* )
* cursor = connection.cursor()
* if connection.is\_connected():
* print("Database Connection is successful")
* else:
* print("Not connected")
* # Execute the SQL query
* query = "SELECT \* FROM incoming\_inventory\_control"
* cursor.execute(query)
* # Fetch all rows from the result set
* rows = cursor.fetchall()
* # Print the data to the screen
* print("Database Report:")
* for row in rows:
* print(row)
* # Write the data to a text file
* with open("database\_report.txt", "w") as file:
* file.write("Database Report:\n")
* for row in rows:
* file.write(str(row) + "\n")
* print("Report generated successfully.")
* except mysql.connector.Error as error:
* print("Error:", error)
* finally:
* # Close the database connection
* if connection.is\_connected():
* cursor.close()
* connection.close()
* print("Database connection closed.")



# 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*

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

**Limited scalability:** The current database may struggle to handle large volumes of data and concurrent user requests.

**Single point of failure:** In case of hardware failures or disasters, there's a risk of data loss and downtime.

**Limited availability:** Access to the database may be restricted to local networks, limiting accessibility.

* *3 points for explaining how you would address these shortcomings*

Implementing database sharding or partitioning to distribute data across multiple servers for improved scalability.

Setting up database replication and backups to ensure data redundancy and minimize the risk of data loss.

Implementing load balancing and failover mechanisms to enhance database availability and minimize downtime.

* *3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.*

Migrating the database to a cloud service provider like Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure.

**Advantages of cloud migration include:**

**Scalability:** Cloud platforms offer scalable resources, allowing the database to handle increasing workloads.

**High availability:** Cloud services typically provide built-in redundancy and failover mechanisms, ensuring HA of the database.

**Disaster recovery:** Cloud providers offer backup and recovery services, making it easier to recover data in case of disasters.

**Cost-effectiveness:** Cloud services offer pay-as-you-go pricing models, allowing for cost optimization based on usage.

* *3 points for explaining the advantages and disadvantages of storing your data in a document-based NoSQL format instead.*

**Advantages:**

**Flexible schema:** NoSQL databases like MongoDB allow for flexible data models, making it easier to accommodate evolving data requirements.

**Scalability:** NoSQL databases are designed for horizontal scalability, allowing for seamless scaling as data grows.

**Better performance for certain use cases:** NoSQL databases can offer better performance for certain types of queries, especially those involving large volumes of data.

**Disadvantages:**

**Lack of ACID transactions:** NoSQL databases often sacrifice ACID properties for scalability and performance, leading to eventual consistency and data integrity challenges.

**Learning curve:** Developers accustomed to relational databases may face challenges in adapting to NoSQL data modeling and querying techniques.

**Limited support for complex queries:** NoSQL databases may not support complex queries and joins as efficiently as relational databases, limiting their suitability for certain use cases.

# 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 - 1/15-1/19

* Sharath – I installed XAMPP software for mySql and ERWIN to create ERD model
* Sharath – Researched on the project I should be working on and gathered all the required files.
* Bindu – I have setup git and other software. Went through the files which are shared by my teammates and set it up in the system. I have setup VS code

Week 2 - 1/22-1/26

* Sharath – Uploaded all the data files to GitHub
* Sharath - started working on making conceptual model for the data I already have.
* Bindu - -I have gone through the documentation step by step and I created the sample tables and sample data

Week 3 - 1/29-2/02

* Sharath - Installed Xampp and wrote create and insert statement in MySQL and took screeshot
* Bindu - I have extracted a dataset from an online source and imported it into an Excel
* Sharath - Tried doing the model in ERWIN but found Vertabelo is way easier.

Week 4 - 2/05-2/09

* Sharath - Finished the word documentations for this week milestone.
* Bindu - I have researched and gone through the data is managing with a relational data model and gathered the reference to understand much better
* Sharath – Created both the models in Vertabelo and pushed all the files into the GitHub

Week 5 - 2/12-2/16

* Sharath - Worked on learning DDL and DML by practicing on XAMPP
* Sharath - Looked at some of the tutorial on google to learn more about SQL. Went through your recording to understand the material and able to perform the milestone.
* Sharath - Finished part H of the word documentations for next week milestone.

Week 6 - 2/19-2/23

* Sharath - Finished doing indexes, views, transaction and security part of this milestone
* Sharath - Finished all the work by myself and learned a great deal on how the databases work.
* Sharath - Finished the word documentations for this week milestone and uploaded on Git and finished submission.

Week 7 - 2/26-3/01

* Sharath - Learning and performing the stored program on my databases was very challenging. I require more work to get good grasp of it. I believe I can look at how other people are implementing the stored procedure in industry and learn from them.
* Sharath - I was able to implement the backup concept for one of my friend where the deadline was fast approaching and he was not able to connect to the database using python. What I did was take a backup of his database, pushing the sql to git and pulled on my computer. Restore the database and was able to connect to database using python on my computer. This way we were able to implement the project on my laptop. After troubleshooting his computer, we found that python was not correctly installed.
* Sharath - Did google search and learn about the cloud sql and their advantages and disadvantages

Week 8 - 3/04-3/08

* Sharath - Completed the whole report by myself and implemented the feedback you have given to me.
* Sharath - Experimenting the python code to connect to database and performing DDL and DML
* Sharath - Installed mysql workbench to get the skill of using a GUI software for SQL.