# A. Title Page

Lewis University  
CPSC 50900: Database Systems   
Spring 2024 Term Project

**Integrative Database Solution for SME Resource Management**

**Report Prepared by**

* Leelasrinivasraju Sarikonda leelasrinivasrajus@lewisu.edu
* Mohan Saty Surapa Raju Nadimpalli mohansatyasurapara@lewisu.edu

Work products stored in the Github repository: [https://github.com/leelasrinivasrajusarikonda/CPSC50900DatabaseSystems](https://github.com/leelasrinivasrajusarikonda/CPSC-50900-Database-Systems)

Table of Contents

[A. Title Page 1](#_Toc152230519)

[B. Initial Proposal 2](#_Toc139688952)

[C. Data Sources 3](#_Toc102302373)

[D. Alternative Ways to Store the Data 3](#_Toc253987590)

[E. Relational Database Design Process 4](#_Toc47888474)

[F. Relational Database Design 4](#_Toc1794510899)

[G. Data Definition Language (DDL) Scripts 5](#_Toc9532520)

[H. Data Manipulation Language Scripts 5](#_Toc2014337146)

[I. Indexes 6](#_Toc364229772)

[J. Views 6](#_Toc423795366)

[K. Triggers 6](#_Toc937775206)

[L. Transactions 7](#_Toc1336454402)

[M. Database Security 7](#_Toc1164301272)

[N. Locking and Concurrent Access 7](#_Toc1578907519)

[O. Backing Up Your Database 7](#_Toc1968323262)

[P. Programming 8](#_Toc723961217)

[Q. Suggested Future Work 8](#_Toc1323642763)

[R. Activity Log 9](#_Toc159721872)

# 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. Relational database design process  f. Relational database design  g. Data definition language scripts  h. Data manipulation language scripts  r. Activity Log – at least six entries covering the past two weeks |
| February 26 at 11:59pm | i. Indexes  j. Views  l. Transactions  m. Security  n. Locking  o. Backup  r. Activity Log – at least six entries covering the past two weeks |

# 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 cant be so narrow that you cant identify at least eight different types of things in it that youd store data about.*

*Total points possible: 20*

**Proposal**

Our project is designed to serve as a comprehensive data repository tailored for small to mediumsized enterprises (SMEs), encompassing a wide range of information crucial to inventory management. This includes detailed product information, supplier contacts, sales transactions, customer profiles, order histories, and employee data. Such a diverse collection of data ensures a holistic approach to inventory control and overall business operations, addressing various factors essential for effective management.

The significance of this project lies in its focus on inventory management, a key area for SMEs. By providing detailed insights into stock levels, supplier relationships, and customer purchasing behaviors, our project directly contributes to enhancing the efficiency and profitability of retail and wholesale businesses. The importance of this data cannot be overstated; its the lifeline of an SMEs operational success, playing a crucial role in ensuring business processes are both effective and efficient.

Central to the effectiveness of this initiative is the principle of effective inventory management. Striking the right balance of stock is critical; overstocking can lead to unnecessary costs and product obsolescence, while insufficient stock can result in lost sales and customer dissatisfaction. Our project aims to empower businesses with a robust database they need to store their data and make informed decisions, minimize waste, and enhance customer satisfaction, all of which are key to maintaining a competitive edge.

To ensure the datas relevance and practical applicability, we are sourcing it from simulated business operations that closely mirror realworld scenarios. The inclusion of product catalogs, supplier data, sales records, and customer transaction histories offer a comprehensive view of business operations from multiple perspectives, enhancing the databases utility and scope.

The primary beneficiaries of this database are SME owners, inventory managers, and sales personnel. They will utilize this database for routine data management and leverage it for strategic planning and decisionmaking. Beyond daily operations, the database is also a valuable resource for business analysts and software developers, who can use it to enhance system features and functionalities.

Our vision is to develop a fullyfledged database system for inventory management, incorporating features like stock level monitoring, order processing, sales tracking, and report generation. This system is not just a database; its a tool for gaining insights into inventory trends, promoting efficient stock management, and improving overall business operations.

To effectively meet these diverse needs, our system will include but not limited to tables such as Products, Suppliers, Sales, Customers, Categories, and Employees. Each table is designed with a specific purpose in mind, collectively creating a robust and comprehensive inventory management system database. This thoughtful design ensures that all aspects of inventory management are covered, providing SMEs with a powerful database to manage their operations more effectively.

# C. Data Sources

*Description:* *Gather your data in text files. The text files may be csv, tabdelimited, 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 youll 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*

**Data Source Overview**

In our database project for the Inventory Management System, weve collected various data files, predominantly in CSV format, to populate our database effectively. These files cover different aspects of inventory management and are structured to facilitate integration into a relational database.

1. **Categories.csv**: Contains categories of products with fields like Category ID, Name, and Description. Its pivotal for classifying inventory items and will be linked to products in the database.
2. **Customers.csv:** Details customer profiles including Customer ID, Name, and Contact Information. This data will be connected to sales transactions to analyze customer purchasing behaviors.
3. **Employees.csv**: Lists employee information such as Employee ID, Name, Role, and Contact Details, essential for managing user roles and access within the system.
4. **Products.csv**: A comprehensive list of inventory items with details like Product ID, Name, Supplier ID, Category ID, Price, and Stock. This file is central to the database, interlinking with sales, categories, and suppliers.
5. **Sales.csv** and **Sales\_item.csv**: These files record sales transactions. **Sales.csv** includes overall sale data, while Sales\_item.csv details individual items sold, connecting inventory to customer purchases.
6. **Suppliers.csv**: Contains supplier information crucial for inventory procurement, including Supplier ID, Name, and Contact Details.
7. **Supply\_orders.csv**: Tracks inventory orders with details like Order ID, Date, Supplier ID, Product ID, and Quantity, vital for stock management.

Each file is structured with appropriate data types and constraints for integration into the database, ensuring data integrity and efficient management.

# 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 Database Storage methods**

In our exploration of alternative database storage methods for the Inventory Management System database, we came across two distinct models: the Hierarchical Database Model and the NoSQL DocumentOriented Database, each offering unique advantages and facing specific challenges.

The Hierarchical Database Model, one of the oldest forms of database structures, operates on a simple, treelike hierarchy. In this model, data is organized into levels, with a single parent node linked to various child nodes. For our Inventory Management System database, this could mean categorizing Products as a primary segment and branching it into subsegments like Categories, Suppliers, and Sales. Implementation involves establishing these parentchild relationships, which simplifies data navigation and ensures data integrity due to the clear hierarchical structure. However, this models’ major limitations lie in its inflexibility and inability to efficiently handle complex hierarchies or multiple relational links, only supporting onetomany relationships.

Conversely, the NoSQL DocumentOriented Database, particularly ones utilizing JSONlike formats, presents a more modern and flexible approach. In this model, data entities such as Products or Suppliers are stored as individual documents in a format similar to JSON, each containing keyvalue pairs. These documents are then grouped into collections for better organization. The primary advantage of this system lies in its flexibility; there’s no predefined schema, allowing for easy adjustments and additions. Additionally, its highly scalable, making it wellsuited for handling large datasets and high user traffic. Despite these benefits, the NoSQL model introduces complexities in aggregating data from various collections and may lead to consistency issues, as it typically ensures eventual consistency rather than immediate data accuracy.

Through these alternative models, our project explores different dimensions of database management. While the Hierarchical model provides a structured and easytounderstand framework, it falls short in handling more complex, relational data scenarios. In contrast, the NoSQL approach, with its adaptable and scalable nature, may introduce challenges in data aggregation and realtime consistency. Each models’ adoption would depend on the specific needs and scale of the database application, highlighting the importance of aligning the database structure with the project requirements.

# E. Relational Database Design Process

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

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

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

*Total points possible: 18*

**Relational Database Design Process**

In database creation, identifying functional dependencies and determinants is crucial for ensuring data integrity and optimizing query performance. This section outlines the functional dependencies identified in the project's database tables, naming conventions for entity sets, and the relationships between these entities.

**Functional Dependencies and Determinants**

Functional dependencies are fundamental to relational database design, ensuring that the database accurately reflects the realworld entities it represents. A determinant is an attribute or a set of attributes on which other attributes depend. Below are the determinants and dependent attributes identified for each table:

1. Products Table

* Determinant: Product ID
* Dependent Attributes: name, description, category\_id, supplier\_id, price

1. Categories Table

* Determinant: Category ID
* Dependent Attributes: name, description

1. Suppliers Table

* Determinant: Supplier ID
* Dependent Attributes: name, phone\_number, email, address, account\_number

1. Customers Table

* Determinant: Customer ID
* Dependent Attributes: name, phone\_number, email, address, city, postal\_code, country

1. Employees Table

* Determinant: Employee ID
* Dependent Attributes: name, email, phone, address

1. Sales Table

* Determinant: Sale ID
* Dependent Attributes: employee\_id, customer\_id, amount, date

1. Sales\_item Table

* Composite Determinant: Sale ID, Product ID
* Dependent Attributes: amount, quantity

1. Supply\_orders Table

* Determinant: Order ID
* Dependent Attributes: supplier\_id, product\_id, quantity, total\_cost, date

**Naming Entity Sets**

For clarity and ease of reference within the database schema, the following entity sets have been named:

1. products
2. categories
3. suppliers
4. customers
5. employees
6. sales
7. sales\_item
8. supply\_orders

**Identifying Relationships Between Entities**

The relationships between entities play a crucial role in database structure and function. The identified relationships are as follows:

1. **products - categories**: One-to-Many. Mandatory for Product, Optional for Category.
2. **sales - customers**: Many-to-One. Mandatory for Sale, Optional for Customer.
3. **sales - employees**: Many-to-One. Mandatory for Sale, Optional for Employee.
4. **sales\_item - products**: Many-to-One. Mandatory for sales\_item, Optional for Product.
5. **sales\_item - sales**: Many-to-One. Mandatory for sales\_item, Optional for Sale.
6. **supply\_orders - suppliers**: Many-to-One. Mandatory for supply\_orders, Optional for Supplier.
7. **supply\_orders - products**: Many-to-One. Mandatory for supply\_orders, Optional for Product.

# F. Relational Database Design

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

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

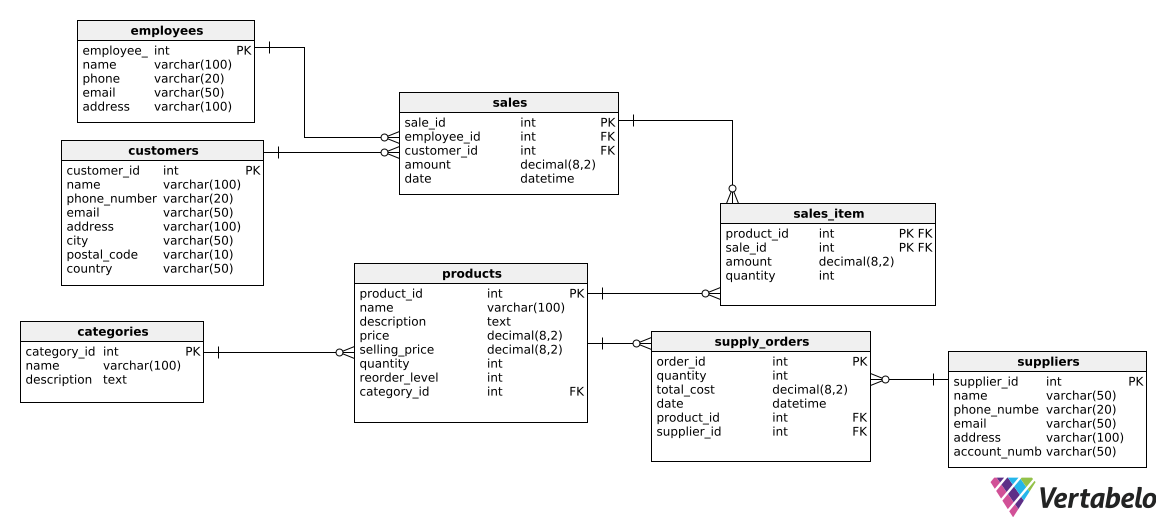
* *4 points for the normalization analysis of your entity sets.*
* *3 points for introducing bridge entity sets.*
* *3 points for choosing foreign keys and perhaps more efficient surrogate keys*
* *10 points for correctly depicting your physical database model in Vertabello*

*You will be penalized 4 points if your database doesnt have at least 8 appropriately defined tables.*

*Total points possible: 20*

**Database Design – Entity Relational Model**

The provided Entity-Relationship Diagram (ERD) offers a comprehensive visual representation of the database structure. It depicts the interconnections between various entities such as employees, customers, products, sales, sales items, supply orders, categories, and suppliers. Each entity is clearly defined with its attributes, primary keys (PK), and foreign keys (FK), highlighting the relationships ranging from one-to-many to many-to-one. This diagram serves as a blueprint for understanding the database schema, showing how different tables relate to one another and laying the groundwork for the implementation of a relational database that efficiently manages the store's operations and data flow.

****

# G. Data Definition Language (DDL) Scripts

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

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

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

*Total points possible: 20*

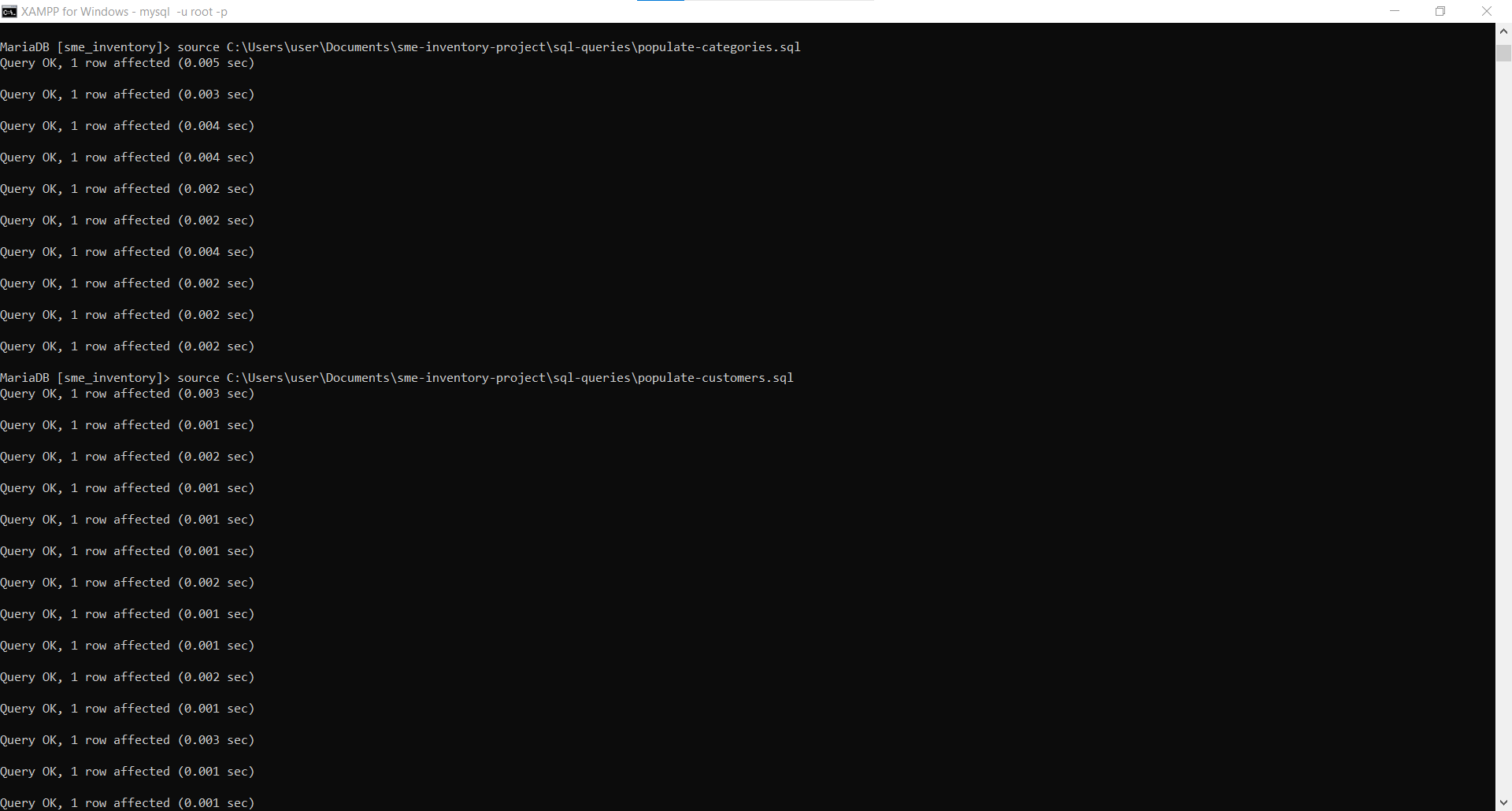
**Database Definition and Creation**

The process of initializing a database begins with the creation of its structure, for which SQL script, is generated by database design tool Vertabelo. These scripts contain the necessary SQL statements to create tables and define their relationships, ensuring that the database schema is accurately implemented.

Following the establishment of the database structure, the next critical step involves populating these tables with data. To facilitate this, a Python script has been crafted to automate the conversion of previously discussed data files in CSV format into SQL INSERT statements. The excerpt of code provided below showcases a function named generate\_insert\_statements, which reads the contents of CSV files within a specified directory, dynamically constructs the corresponding INSERT statements for each row of data, and then compiles these statements into a .sql file. This automated approach streamlines the process of transferring bulk data into the newly created database tables, ensuring efficiency and accuracy in initializing the database with the required data.



The screenshot below displays the output from a MySQL command-line interface, where the source command has been used to execute SQL scripts that populate database tables with records. Each "Query OK" message confirms the successful insertion of a row into the respective tables within the sme\_inventory database. Although the screenshot shows only a portion of the output, but this suggests that the tables have been populated. This indicates a successful data import operation, a critical step in initializing the database with starting data sets for use in the inventory system.



# H. Data Manipulation Language 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 multitable 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.*

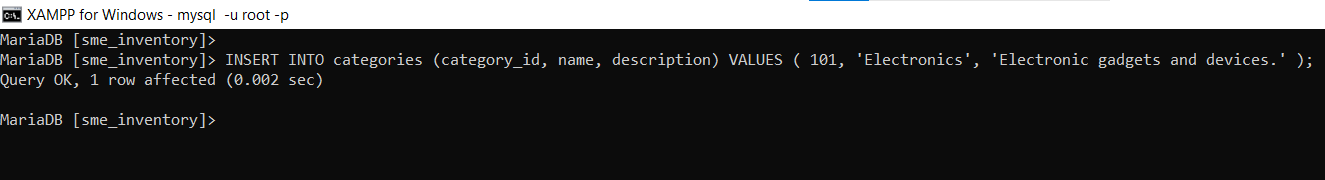
*Rubric: Your work will be graded as follows:*

* *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 multitable query*
* *2 points for the query of your choice.*
* *6 points for showing the query and a screenshot of the corresponding result set backtoback for each of these queries in your Word document.*

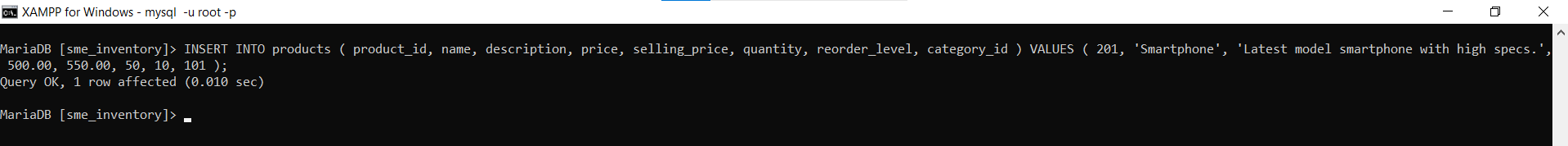
*Total points possible: 24*

1. **Insert Statements**

* **Insert a New Category**

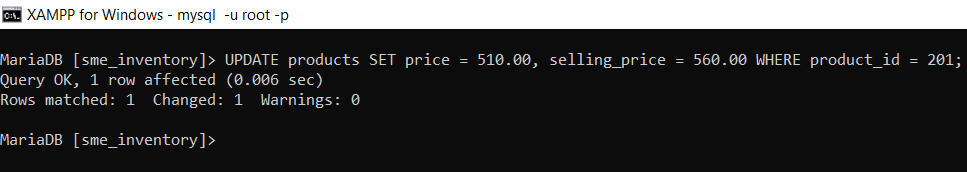
****

* **Insert a New Product**

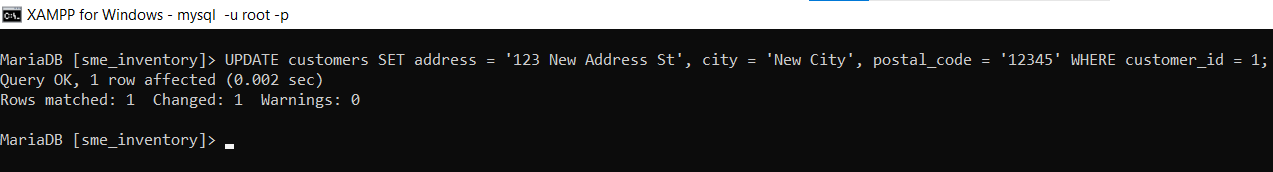
****

1. **Update Statements**

* **Update Product Price and Selling Price**

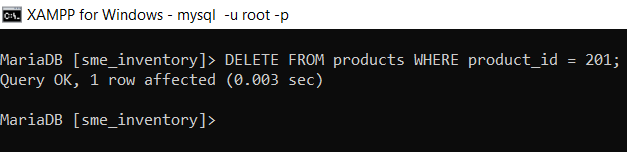
****

* **Update Customer Address**

****

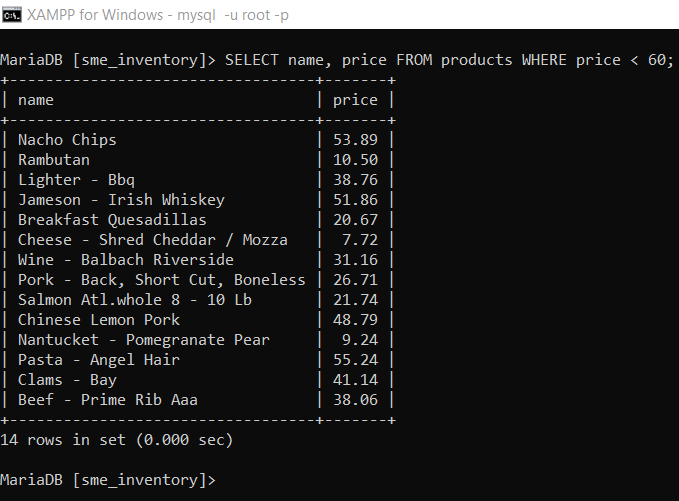
1. **Delete Statement**

* **Delete a Product**

****

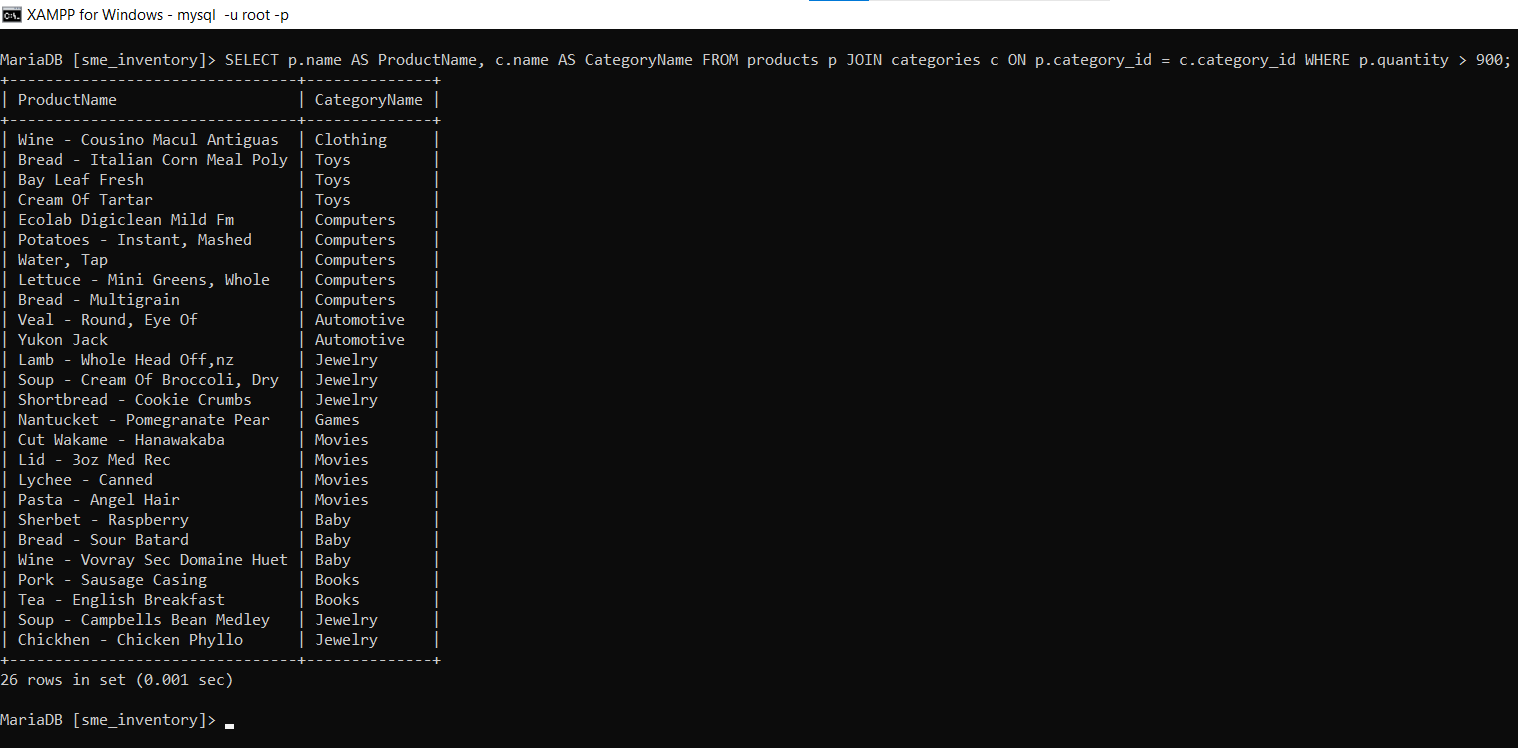
1. **Simple Select Statement**

* **Select Product Names and Prices**

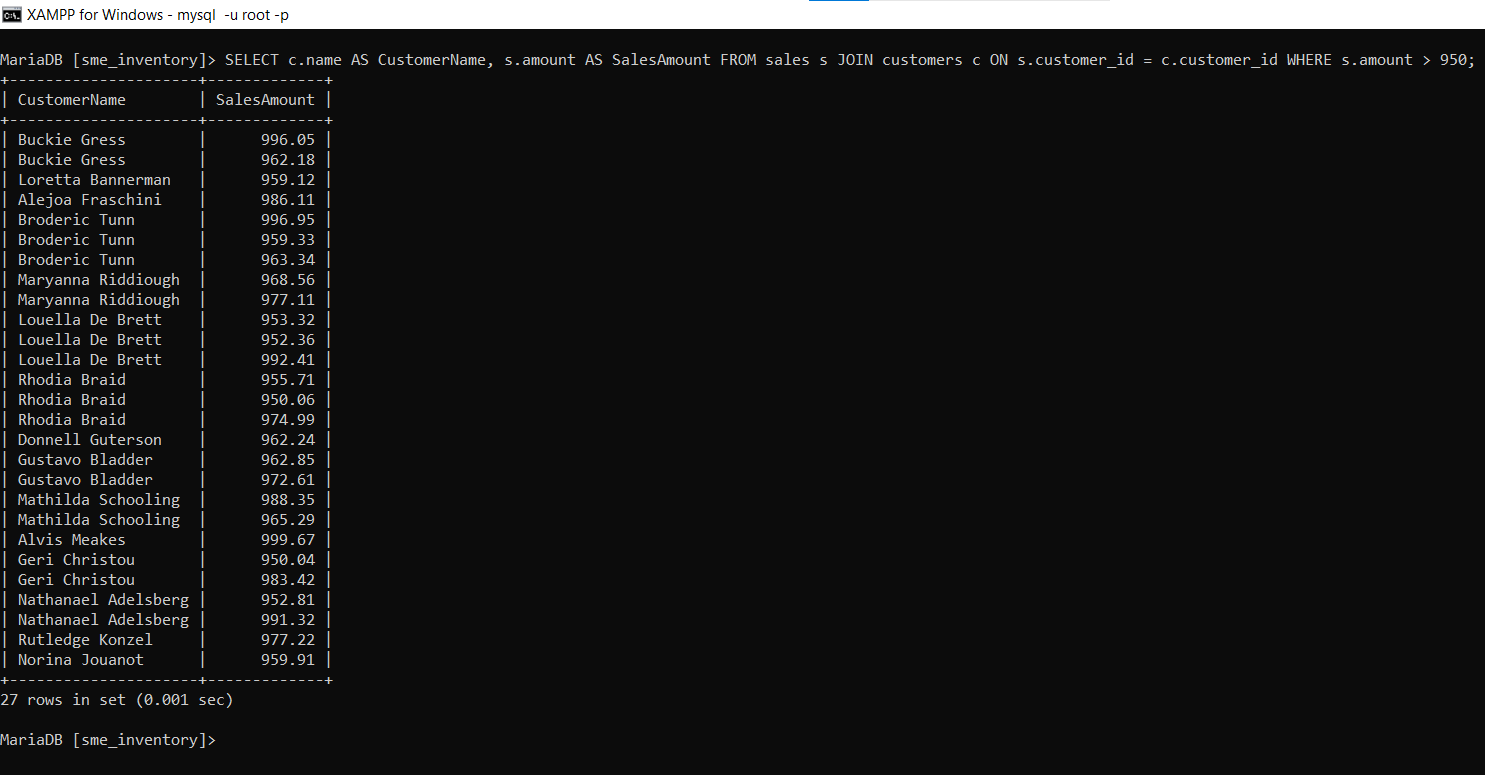
****

1. **Join Statements**

* **Product Names with Their Categories**

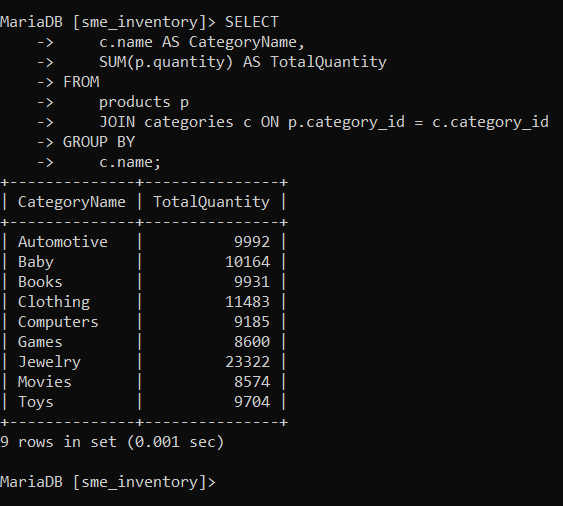
****

* **Sales Amounts and Customer Names for each sale order**

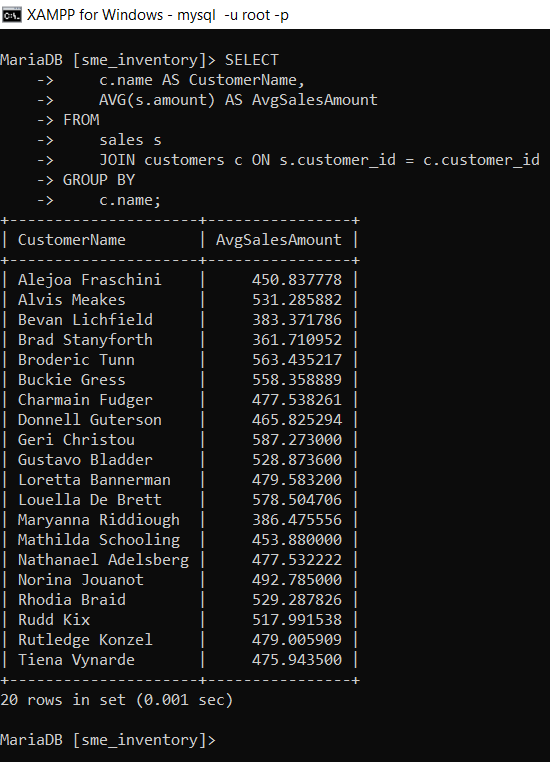
****

1. **Summary Functions**

* **Total Quantity of Products Grouped by Categories**

****

* **Average sales order amount for customers**

****

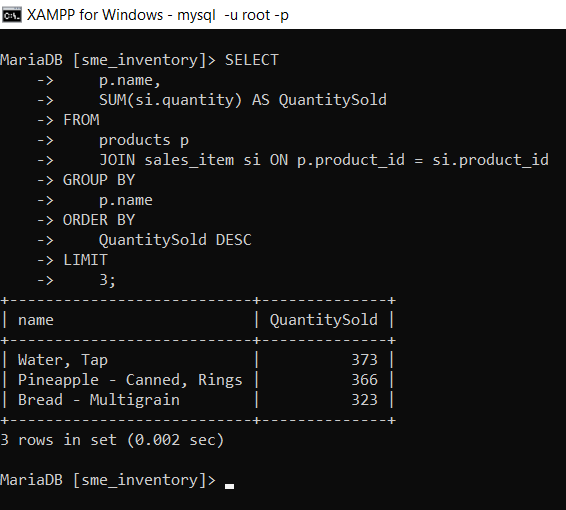
1. **Multi-table Query**

* **Products and Their Sales Quantities**

****

1. **Query of Choice**

* **Top 3 Best-Selling Products**

****

# 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:*

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

*Total points possible: 6*

ENTER YOUR WORK WITH VIEWS HERE

# K. Triggers

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

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

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

*Total points possible: 6*

ENTER YOUR WORK WITH TRIGGERS 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:*

* *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*

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.*
* *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

# 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 documentbased 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 dont 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*

**Activity Logs**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Contributor** | **Activity** | **Description** |
| Jan 14, 2024 | Leelasrinivasraju Sarikonda and Mohan Nadimpalli | Project kickoff | Researched on various database systems and their applications |
| Jan 16, 2024 | Mohan Nadimpalli | Initial Proposal Drafting | Drafted the initial proposal, focusing on project objectives and significance. |
| Jan 18, 2024 | Leelasrinivasraju Sarikonda | Research on Data Sources | Researched potential data sources for the inventory management system. |
| Jan 21, 2024 | Mohan Nadimpalli | Data Files Organization | Organized data files into CSV format for database integration. |
| Jan 24, 2024 | Leelasrinivasraju Sarikonda | Alternative Database Models Exploration | Explored and documented NoSQL and Hierarchical models as alternatives to relational databases. |
| Jan 28, 2024 | Leelasrinivasraju Sarikonda | GitHub Setup | Created the GitHub repository for the project and prepare the README document. |
| Jan 31, 2024 | Leelasrinivasraju Sarikonda | Relational Database Design Process | Identified functional dependencies among the fields provided, designated determinants, and outlined entity sets. Initiated naming conventions for entity sets. |
| Feb 2, 2024 | Mohan Nadimpalli | Relational Database Design Process (Cont.) | Established relationships among entity sets, identifying connectivity and participation for each. |
| Feb 4, 2024 | Leelasrinivasraju Sarikonda | Relational Database Design | Analyzed entity sets for adherence to Normalization. Began ER diagram design in Vertabelo. |
| Feb 7, 2024 | Mohan Nadimpalli | CSV to SQL Python Script | Developed Python Script to convert CSV data files to SQL INSERT queries. |
| Feb 9, 2024 | Leelasrinivasraju Sarikonda | Data Definition Language (DDL) Scripts | Used Vertabelo to generate DDL scripts for database creation. Evaluating the script ensuring foreign keys were linked. Prepared documentation and screenshots for report inclusion. |
| Feb 11, 2024 | Mohan Nadimpalli | Data Manipulation Language Scripts | Wrote SQL commands for various operations including insert, update, delete, and select queries. Prepared documentation and screenshots for report inclusion. |