**PROJECT TITLE**

**SOFT BUDDY**

NAME:

GITHUB URL:

CURRENT DATE:04-Nov-2022

**Part-1 Design**

**1) Identify the specific products that you sell related to the theme above, and write a short description of your business that includes what you sell, the size of your organization, general locations, and annual sales goals.**

Soft Buddy is a toy manufacturing and retail sales company based out of the USA. We are famous for making fine-quality soft toys like teddy bears, pillows, cuddling teddies, stuffed animals, etc. Soft Buddy is a small-scale organization with an employee base of 20–30 and an annual revenue of $5 million.

The company's head office and warehouse are located in California, and manufacturing units are located in other countries like Vietnam. We import raw materials like fabrics, plastic, thread, and other things from India and China.

**2) Compare your business to a real one by researching the annual sales and number of employees that the real business has.**

**\* Write the name of the similar business.**

**A)** Plan toys

**\* Write its sales for 2021, and current number of employees.**

**A)** Annual sales: $21 M

Current no of employees: 82

**\* Also provide the URL of where you found the information.**

**A)** <https://www.zoominfo.com/c/plantoys-inc/95224880>

**3) Write your description that explains your Client Server design and what service each tier provides. Even if your network does not use many tiers, identify 5 tier functions and describe how each would be useful.**

A)Client-server architecture is a computing model where the server hosts, sends, and manages all the resources and tasks required by the client. In this architecture, there are one or more clients connected to a central server.

Our organization uses a 3-tier client-server architecture.

**Presentation tier:** This is the first layer, which provides the user interface. The user interface is a graphical user interface. It provides information related to browsing, sales, purchasing, etc. It attaches to tiers in the network by computing results and sending them to the browser.

**Logic tier:** It is also known as the data access tier or middle tier. It controls the application's functions by performing processing. The components built into this layer exist on the server.

**Data tier:** This is the data management layer, which consists of a data base or server. In this layer, data is stored and retrieved. The main responsibility of this layer is to keep data consistent and independent.

**The functions of the 5-tier architecture:**

Presentation: The end user sees the application in this format.

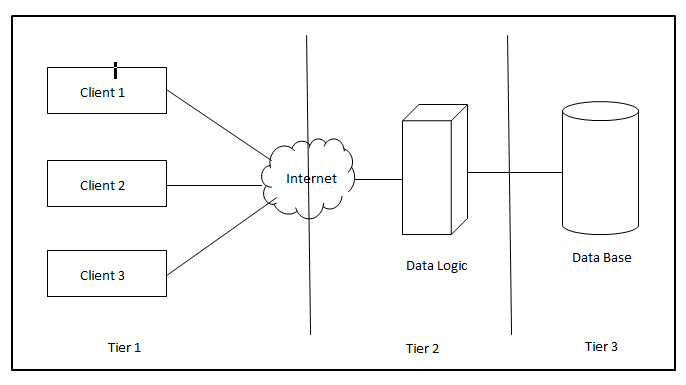
User interface: This is where user interactions are interpreted.

Business logic: The data validation, modification, security, processing, database lookups, etc. takes place here. Depending on the level of security required by the action, such responsibilities may be split between the UI side and data access.

Data Access: In this section, you abstract the database operations you must carry out, such as insert, remove, etc. Between your program and databases, it functions as a kind of interface.

Data Storage: Database servers are appropriate here for storing data. The Data Access layer allows users to telepathically add, delete, and alter data on any system.

**\* Also draw a simple diagram to illustrate your network design**

****

**4) The business will need to store many different types of data. That means elements of your business related to building and selling your products, not computer data types. Think about 5 important types of data that are related to your business. What do you need to track? List those types in your report.**

**Product data:** Product data will be used to track a product's cost, availability, and sale dependent on the sale of another product, as well as to provide a discount for that product. I might use the product inventory to assess my business.

**Sales data:** The data from the sales cycle across all devices and channels gives retailers a variety of information into how each line of business functions when combined with the information from customer and product information. This knowledge opens the door for alternative strategies to pairing the ideal clients with the ideal goods.

**Logistics data:** Information on delivery partners, warehouse specifications, and storage capacity will all be tracked by logistics data. This data will be used to check the capacity of the warehouse, the status of the shipments, and returns.

**Order data:** Employees use order data to determine which orders are being delivered efficiently and to determine the interests and needs of the customer. He can verify shipment times, identify any gaps in the supply chain, and develop new storage by monitoring how goods reach and depart warehouses using logistics data.

**Employee data:** This includes the employee's full name, contact information, date of birth, marital status, employment data, family information, emergency contact information, and health insurance information so that we can keep track of all employees.

**5) What kind of tool did we mention in class that performs data translations from one system to another?**

The tool is the CastIron. With the help of CastIron Cloud integration products, you can quickly combine cloud and on-premise applications, cut integration costs, and increase productivity in cloud and software as a service (SaaS) models. In place of custom coding, on-demand tooling, or conventional middleware, they offer a graphical configuration technique to assist you in rapidly and easily integrating applications.

**\* Name FIVE current examples of that tool, and provide the URL where you found the information.**

**MuleSoftAnypoint Platform:** Anypoint Platform is a unified, extremely effective hybrid integration platform that connects apps, data, and devices through APIs to construct an application network.

**Knowledge Vault:** A cloud-based analytics, auditing, reporting, and management platform called Knowledge Vault is for businesses looking for a quick and economical method to secure and improve their Office 365 environments and well-known cloud storage systems.

**Cloud Elements:** The best platform for SaaS app developers and the digital enterprise is called Cloud Elements. Make integration a competitive advantage for you**.**

**Bedrock Data:**  BedrockData offers solutions to continually synchronize, clean, and integrate various cloud systems in real-time.

**Veeqo:** It interfaces with Amazon, eBay, Magent, Shopify, and WooCommerce to sync inventory and print shipping labels. Beautiful online inventory and order management software.

<https://www.saashub.com/ibm-cast-iron-cloud-integration-alternatives>

**\* Name three common data formats that computer systems use to transfer product data between systems.**

**.xml –XML file:** Data descriptions are produced by means of XML (Extensible Markup Language). The XML standard provides a versatile way for businesses to build information formats and share structured data electronically over both private and public networks.

**.tsv – Tab Separated value:**  It is a text format which store data in a table structure where each record in the table is stored as one line of the text file. The field values are separated by tab characters.The files are stored with .tsv extension.

**.json file:**It is a standard text based format for representing data based on JavaScript object syntax. It is used to transfer data in web applications.

**Part-2**

**\*Data Entities, Attributes, and Files.**

**1) List 5 entities and provide a short description of each to identify what they are.**

**A)** Entity is an real-world object that represents data in Relational Database Management System(RDBMS). The characteristics of an entity is described by its attributes. The data stored under these attributes are termed as records. The 5 entities that we use for our organization are :

**Stores:**  Stores entity is used to collect all the information about stores that are registered for my business which helps in managing orders received from the stores and sends orders to the stores. This also helps in expanding my business to open new stores in more localities. It contains the attributes like store id which acts as a primary key for the stores entity, store name, contact number and location.

**Product:** The information about all the products that are manufactured are stored under this entity. By using this data i can check all my product availability, cost and estimate sales. Every product has a unique id which acts a primary key for this entity and it is represented as productID. It also contains other attributes like product name, Quantity, price and Description.

**Orders:** when an order is placed by an stores owner an orderid is created which acts as an primary key for this entity. The orderid is received by the employee in depository to check the product availability and ships the order to the stores by using the same orderid. Order entity also have other attributes like date and time of order creation, product id(FK) , product quantity(FK) from the products entity, Storeid (FK) and StoreLoc from Stores entity.

**Logistics:** Logistics data is stored to keep track of all information about the deliveries of the orders placed. It stores information about our delivery partners, repository location , repository id from where the order is shipped storeid(FK) from stores entity to deliver the order. The deliveryID acts as a primary key for this entity.

**Employee:** The details about employees working in our organization are stored under this entity. Each employee is assigned with unique id called EmployeeID which acts as a primary key for this entity. It stores all the information about employees under the attributes like Employee Name , contact number, DOB, DOJ, Gender.

**2) Create 5 simple tables describing your entities, their attributes, and the data type of each attribute.**

**A) Stores Entity Table:**

|  |  |
| --- | --- |
| **Stores** | |
| S\_ID(PK) | Int(10) |
| S\_ Name | varchar(20) |
| Contact Number | varchar(13) |
| S\_Loc | varchar(100) |

**Products Entity Table:**

|  |  |
| --- | --- |
| **Products** | |
| P\_ ID(PK) | Int |
| P\_Name | varchar(20) |
| Price | Float(7,2) |
| P\_Quantity | Int(6) |
| Description | varchar(255) |

**Orders Entity Table:**

|  |  |
| --- | --- |
| **orders** | |
| O­\_ID(PK) | Int |
| O\_DateTime | datetime |
| P\_ ID(Fk) | Int |
| Quantity | Int |
| Price(FK) | Float(7,2) |
| O\_Total | Float(7,2) |
| S\_ID(FK) | Int |
| S\_Loc(fK) | Varchar(100) |

**Logistics Entity Table:**

|  |  |
| --- | --- |
| **Logistics** | |
| dlvry\_ID(PK) | Int |
| Repository\_Id | Int |
| Repository\_Loc | varchar(100) |
| O\_ID(FK) | Int |
| S\_ID(FK) | Int |
| S\_Loc(FK) | varchar(100) |

**Employee Entity Table:**

|  |  |
| --- | --- |
| **Employee** | |
| Emp\_ID(PK) | Int |
| Emp \_First\_Name | varchar(20) |
| Emp\_Last\_Name | varchar(20) |
| Gender | Varchar(1) |
| Contact\_Num | Varchar(13) |
| Role | varchar(50) |

**3) Create .CSV files for your entities and attributes. Each entity should have its own file. That means you will end up with 5 CSV files. Each file should also have 3 records that are specific examples of that entity. Save these files to your github.**

**A)** I have created five **.CSV** files for five entities with three records each. I have uploaded all my .CSV files to my github. The link to the github is provided in the title page.

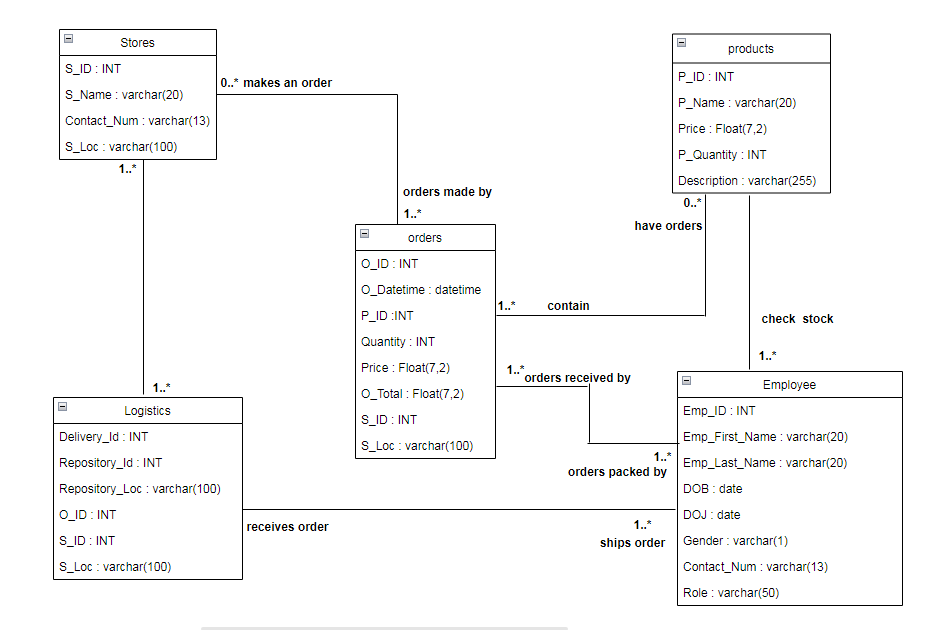
**4) Repeat step 3, but format the files in XML. That means you will have 5 files with 3 records each that populate at least 3 attributes per record using XML tags. You can define the tags any way you choose, but you must use the XML style. Save these files to your github.**

**A)** I have created five **.XML** files for 5 entities with three records each. I have uploaded all my .XML files to my github. The link to the github is provided in the title page.

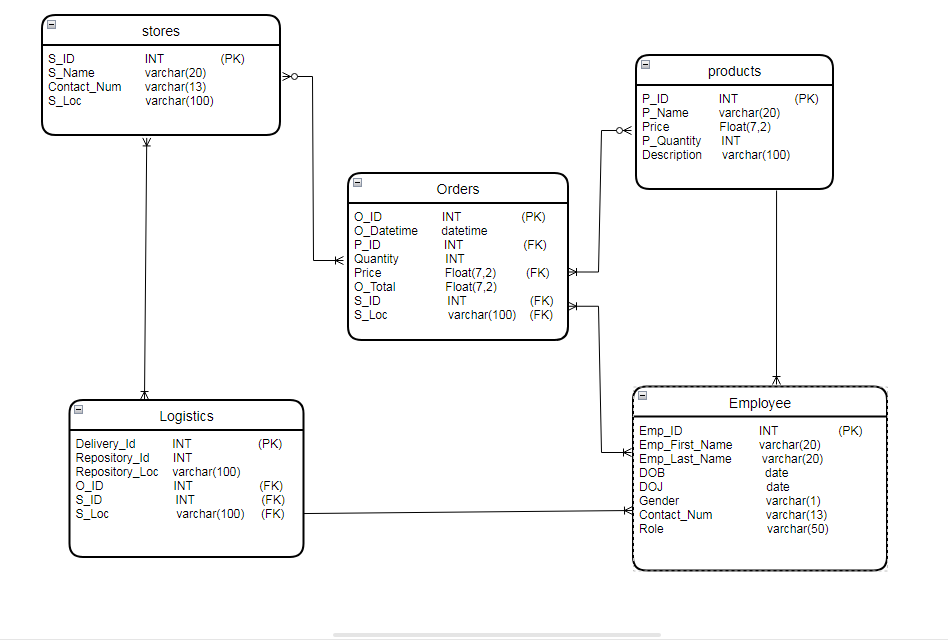
**Part-3**

**ENTITY RELATIONSHIP DIAGRAM (ERD)**

**1) Create an ERD using UML**

****

**2)Create a second ERD using Crow’s Foot notation.**

****

**3) Finally, write a description of the relationships. Using the time/sales/store/product example from Week 2, you might write:   
  
Sales depends on 1 and only 1 time entity, 1 or zero store (online sale) , and 1 or many products.**

* Stores makes an zero or many orders, receives order from one or more Logistics.
* Orders placed from one or more stores, received by one or more employees, and contains one or more products.
* Employee receives one or more orders, checks one or more products, ships through one or more logistics.
* Products can have zero or more orders, products availability are checked by one or more employees.
* Logistics ships one or more orders to stores, receives one or more shipments from the employees working in the repository.

**Part - 4(a)**

**1. Create the SQL script for creating your data tables. You should have 1 table per entity for a total of 5 tables. You can either write this in one script or split your work into 5 small scripts. Save and submit script(s) to your github.**

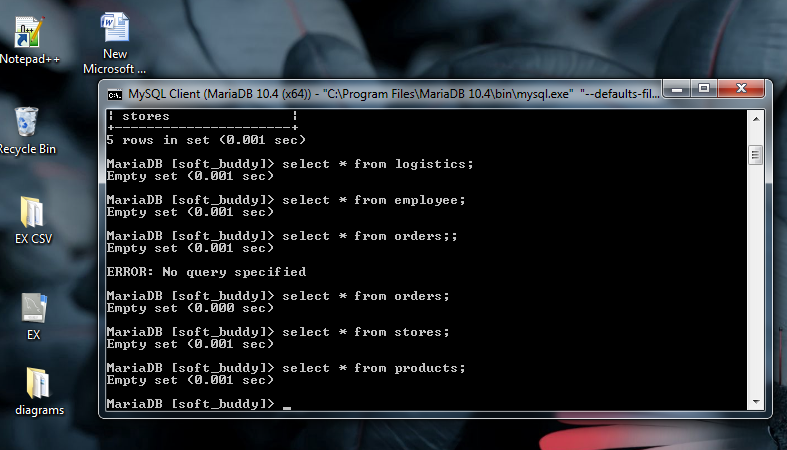
I had created the SQL script for creating data tables and uploaded my “.sql” files to github.

**2. Create the SQL script to load your data files into your tables. Again, you may split up your script. Save and submit your work to your github.**

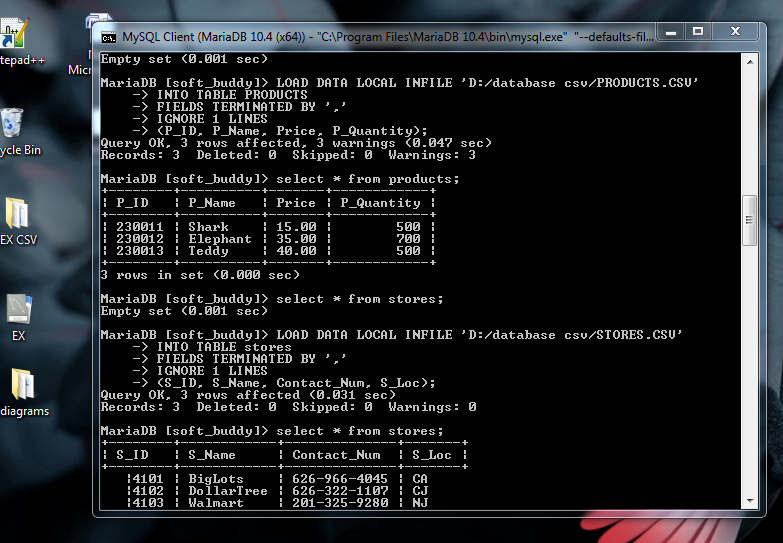
I had created the SQL script to load data files in to tables and uploaded my “.sql “ files to github.

**3. Take screen captures of your data tables that show they are empty. I only need one screen, so if you can only fit two or three tables to a screen, that’s fine. Then take a screen capture of the load command after it successfully executes. Then finally take a screen capture of the same tables populated with data.**

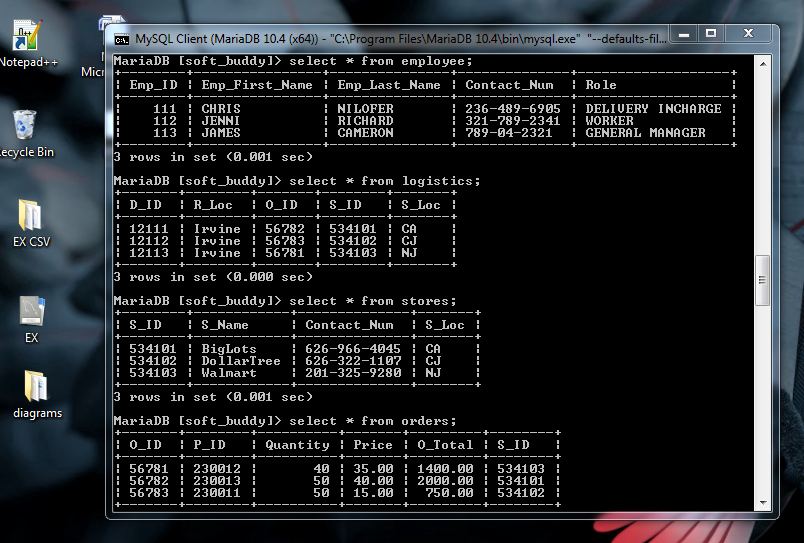
* **EMPTY TABLES:**

****

* **LOADING DATA FILES:**

****

* **POPULATED TABLES:**

****

**Part- 4(b)**

**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 queries should be a select statements that select data from a joining of two tables, three queries should use summary functions to generate statistics about the data, and one query should be a multi-table query. Show the queries and screenshots of the results in your Word document report, and save your queries in a commented sql script to GitHub.**

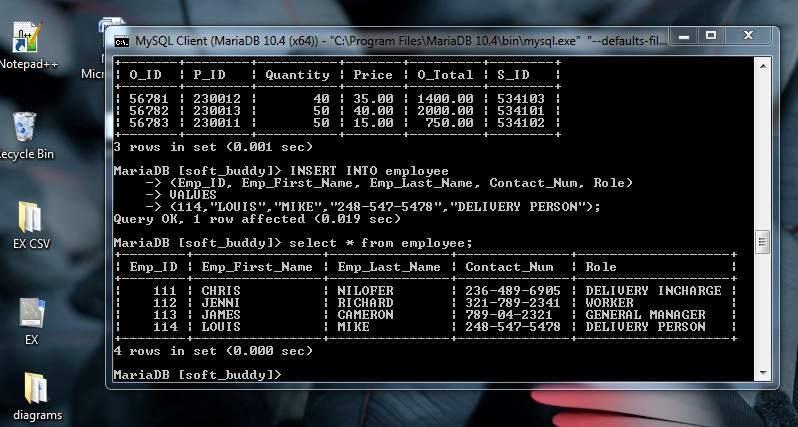
* **Insert Command:**

INSERT INTO employee

(Emp\_ID, Emp\_First\_Name, Emp\_Last\_Name, Contact\_Num, Role)

VALUES

(114,"LOUIS","MIKE","248-547-5478","DELIVERY PERSON");



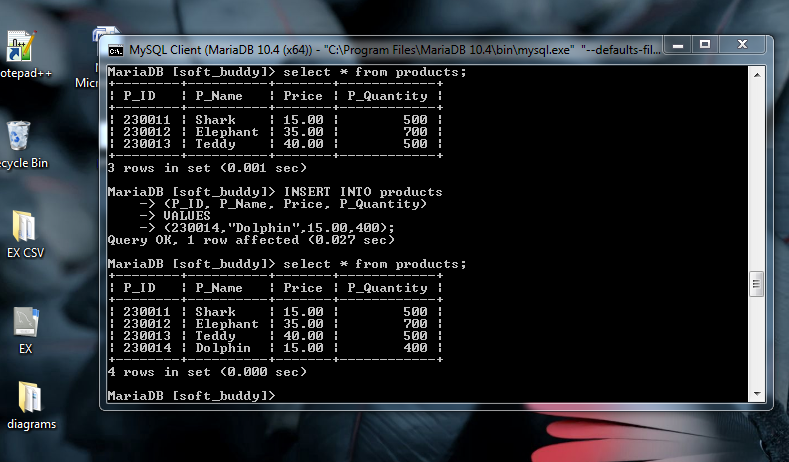
* **Insert Command:**

INSERT INTO products

(P\_ID, P\_Name, Price, P\_Quantity)

VALUES

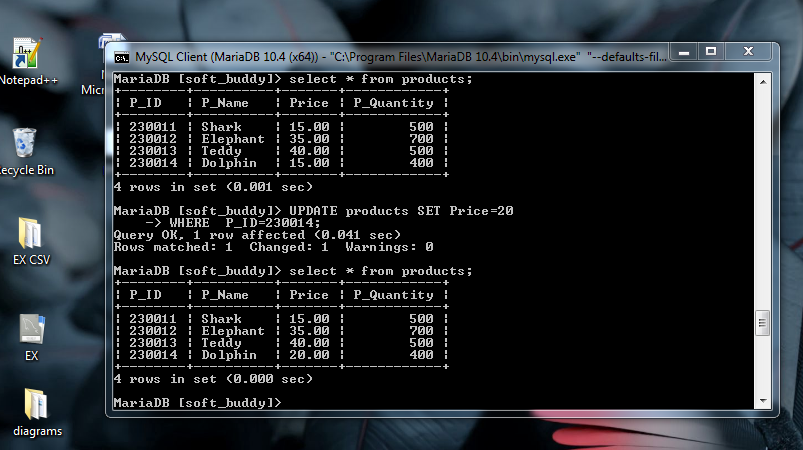
(230014,"Dolphin",15.00,400);



* **Update Command:**

UPDATE products SET Price=20

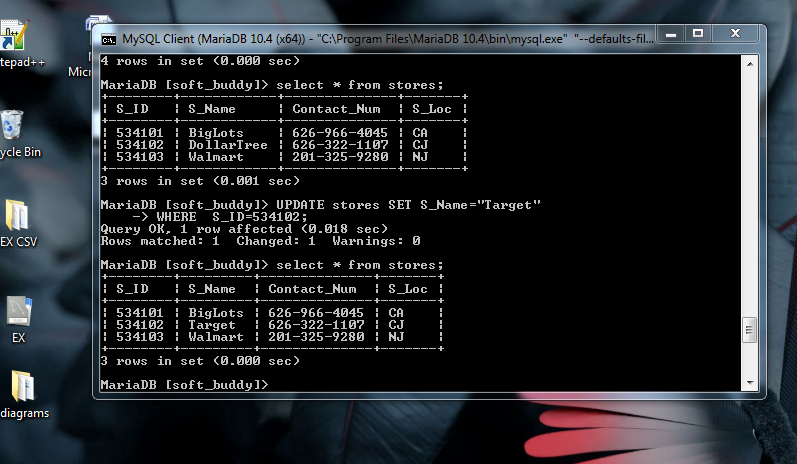
WHERE P\_ID=230014;



* **Update Command:**

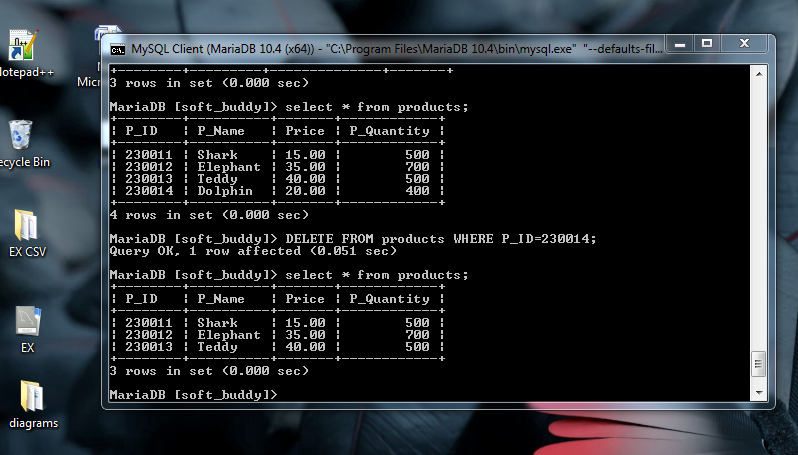
UPDATE stores SET S\_Name="Target"

WHERE S\_ID=534102;



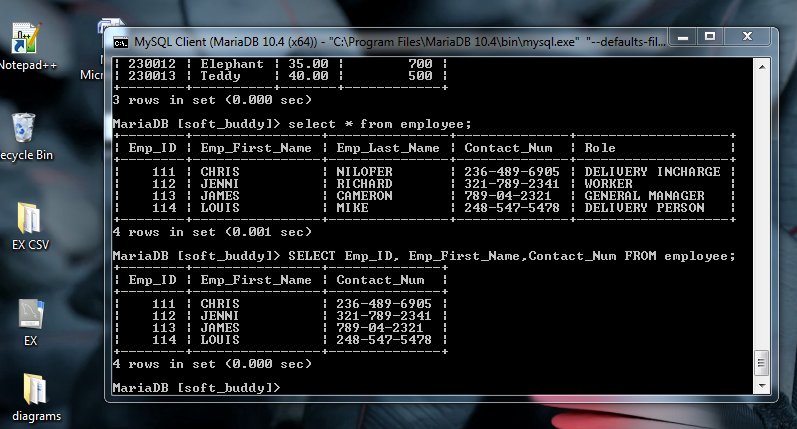
* **Delete Command :**

DELETE FROM products WHERE P\_ID=230014;



* **Select Command:**

SELECT Emp\_ID, Emp\_First\_Name,Contact\_Num FROM employee;

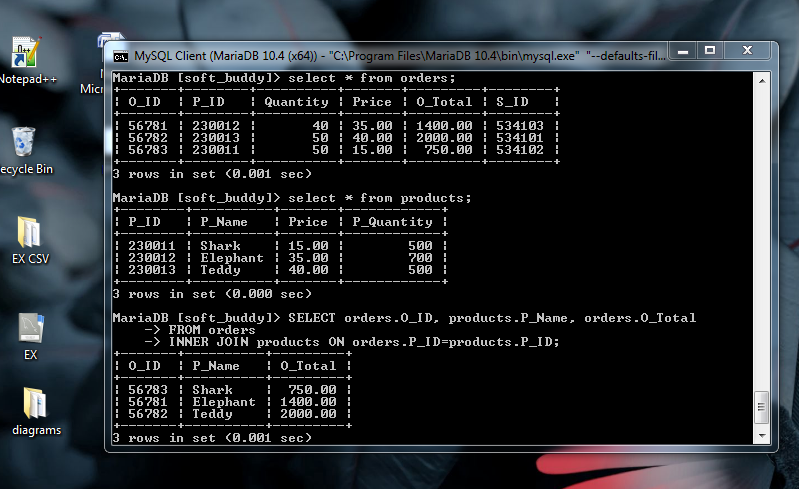


* **Inner join Command:**

SELECT orders.O\_ID, products.P\_Name, orders.O\_Total

FROM orders;

INNER JOIN products ON orders.P\_ID=products.P\_ID;

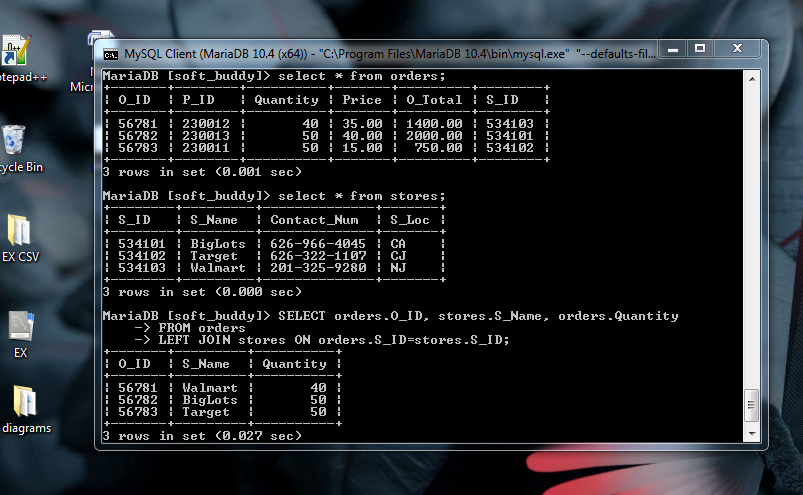


* **Left join Command:**

SELECT orders.O\_ID, stores.S\_Name, orders.Quantity

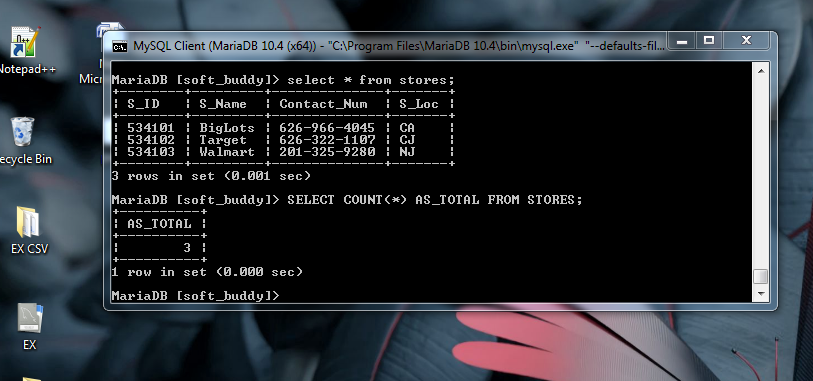
FROM orders

LEFT JOIN stores ON orders.S\_ID=stores.S\_ID;



* **Count Function :**

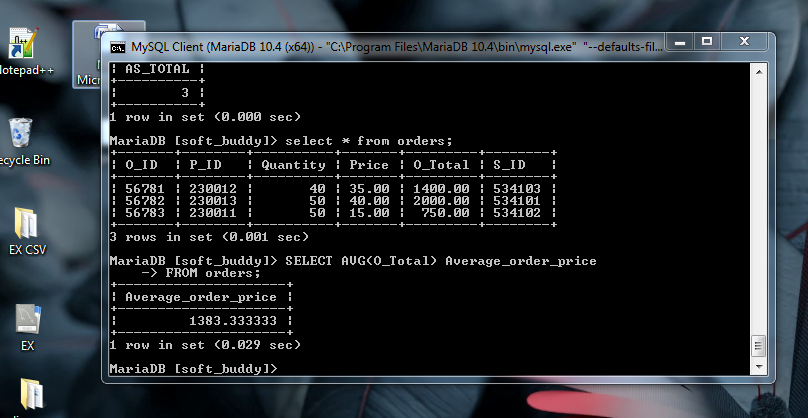
SELECT COUNT(\*) AS\_TOTAL FROM STORES;



* **Average Function:**

SELECT AVG(O\_Total) Average\_order\_price

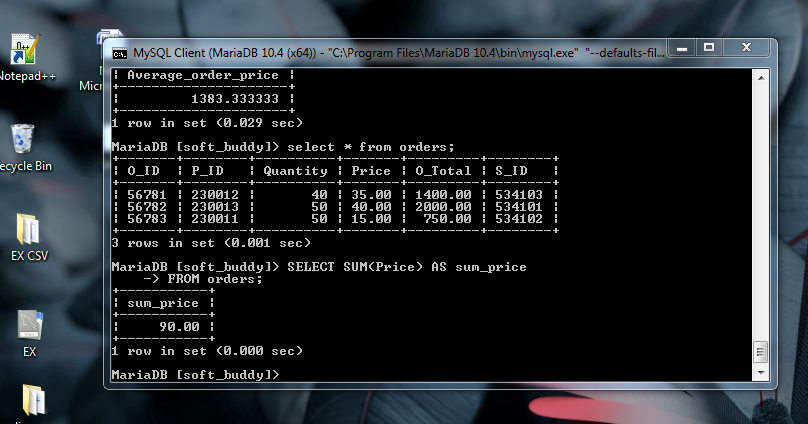
FROM orders;



* **Sum Function:**

SELECT SUM(Price) AS sum\_price

FROM orders;

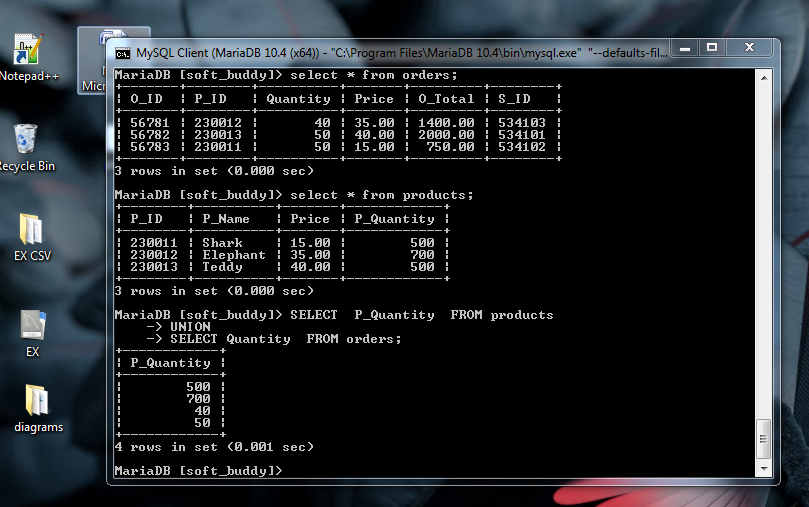


* **Multitable Query:**

SELECT P\_Quantity FROM products

UNION

SELECT Quantity FROM orders;



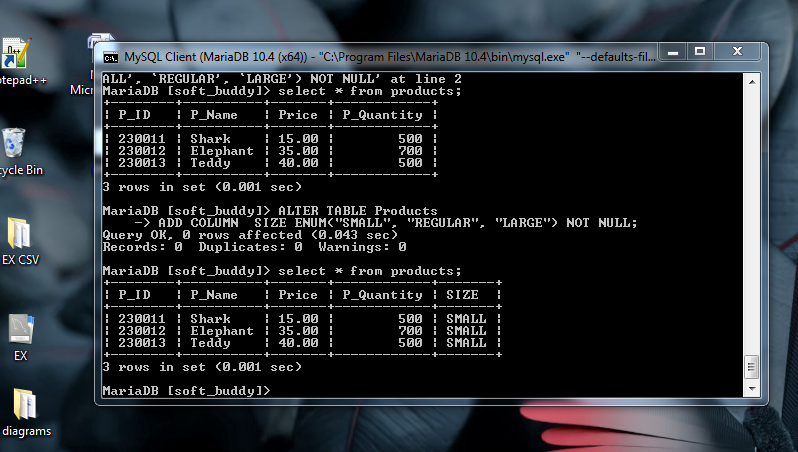
**1. Add a new column of type ‘ENUM’. Document in your report what the purpose of this column is, and what the choices represent. Add a constraint that prevents null values. Take screen captures of your table before and after the column is created.**

**ENUM:** An ENUM is a string object with a value chosen from a list of permitted values that are enumerated explicitly in the column specification at table creation time. It is short for enumeration, which means that each column may have one of the specified possible values. Reduces the likelihood that errors will be introduced into the code by making it easier to read. Here ,I added a column of name “Size” with enum datatype to choose the size of the product from the choices listed like “small, Regular, Large”.

**Adding size column to products table with ENUM datatype:**

ALTER TABLE Products

ADD COLUMN SIZE ENUM(“SMALL”, “REGULAR”, “LARGE”) NOT NULL;



**2. Insert a new record of data into the same table that you added the new column. This requires two steps:**

**a) show that the system throws an error if you try to leave the required ‘enum’ value blank.**

**b) modify your insert to include a valid entry so that the data works properly.**

**a) Inserting values without size column value throws error:**

INSERT INTO products

(P\_ID, P\_Name, Price, P\_Quantity,SIZE)

VALUES

(230014,"Dolphin",15.00,400,” “);

**b) Modifying insert command so that data works properly:**

INSERT INTO products

(P\_ID, P\_Name, Price, P\_Quantity,SIZE)

VALUES

(230014,"Dolphin",15.00,400,”LARGE”);

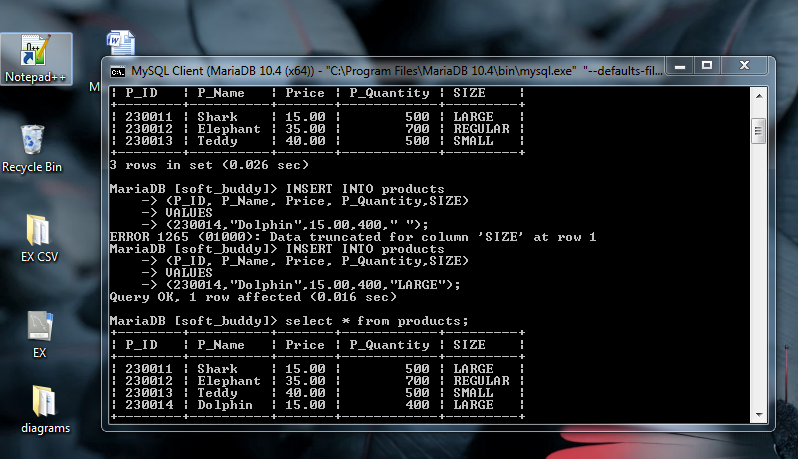
(OR)

(230014,"Dolphin",15.00,400, 3);

Each enumeration value has an index:

•The elements listed in the column specification are assigned index numbers, beginning with 1.

•The index value of the empty string error value is 0.



**3.**  **In your report, explain what changes you can make to each column of the record you just added in #2. Then, modify that data to show those changes. Include a screen capture that shows the ‘after’ result. The ‘before’ state is already documented from step 2.**

I have made changes to the previous records in the Size column for different products by using update command.

**UPDATING THE SIZE COLUMN OF PREVIOUS RECORDS:**

* UPDATE Products

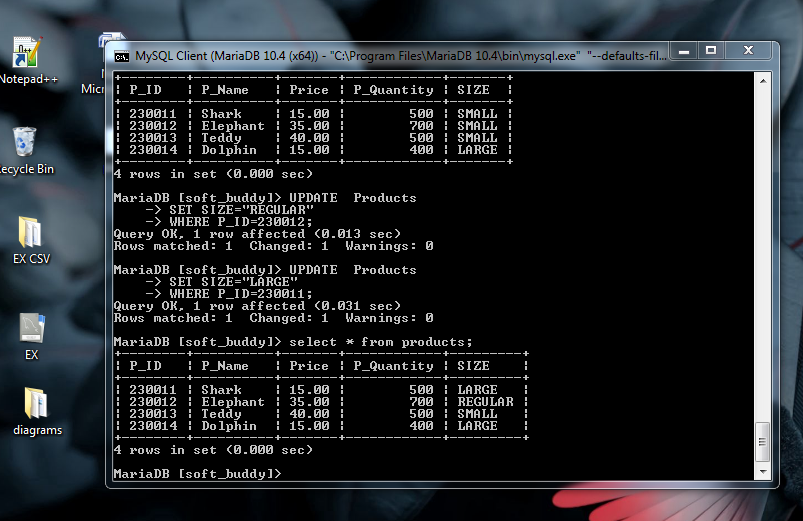
SET SIZE=”REGULAR”

WHERE P\_ID=230012;

* UPDATE Products

SET SIZE=”LARGE”

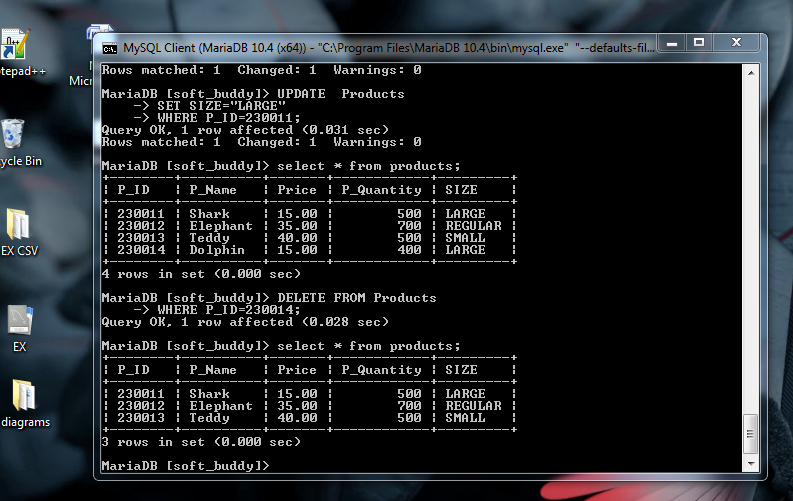
WHERE P\_ID=230011;

****

**4.** **Delete the new record. Use the ‘where’ clause to specifically select just the new record. Show a screen capture of the command and the ‘after’ result.**

DELETE FROM Products

WHERE P\_ID=230014;



**5. Demonstrate the use of the ‘AUTO\_INCREMENT’ constraint by adding a record to a table with this feature. If you do not have this feature already enabled, then alter a table to include a new column. Show a screen capture of the table before you add the new record, and another screen capture after you add the new record to demonstrate the auto increment.**

**AUTO\_INCREMENT:** Auto-increment is a concept in SQL which automatically generates the unique number in the field when the new row is entered into the table. This feature is generally used for the Primary Key field, where we need to create a unique value for every record. If we use the Auto-Increment concept in an integer column and insert the value of the first tuple as 1, then the value of the second tuple is automatically generated as 2.

**Altering Employee table to modify ID column with auto\_increment constraint:**

ALTER TABLE EMPLOYEE

MODIFY Emp\_ID INT NOT NULL AUTO\_INCREMENT;

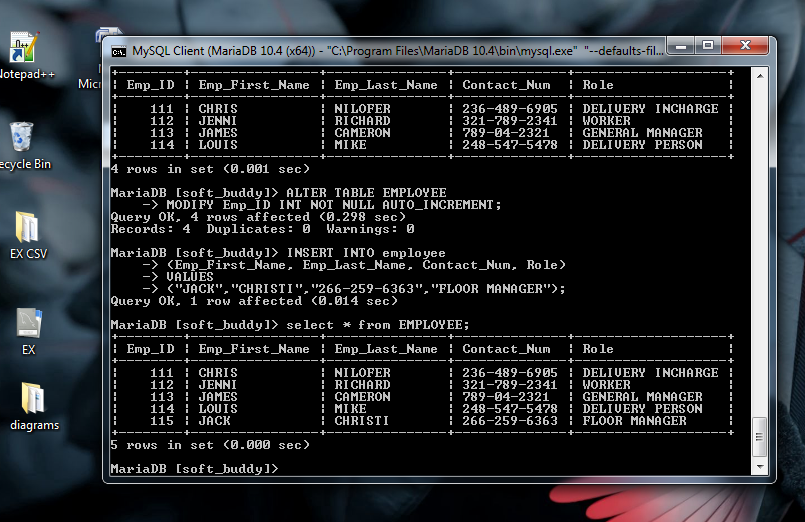
**Inserting new record after altering table:**

INSERT INTO employee

(Emp\_First\_Name, Emp\_Last\_Name, Contact\_Num, Role)

VALUES

("JACK","CHRISTI","266-259-6363","FLOOR MANAGER");



**6. Repeat step 5, except demonstrate the use of the ‘DEFAULT’ constraint by adding a record to a table with this feature. You may need to add a column with this feature if you don’t already have it. Prove that adding a new record of data without this value during entry will still get the default value after your ‘insert’ command executes.**

**DEFAULT:** The **DEFAULT Constraint** is used to fill a column with a default and fixed value. The value will be added to all new records when no other value is provided.

**Altering logistics table to add a default constraint:**

ALTER TABLE logistics

ALTER R\_Loc SET DEFAULT “Irvine”;

**Inserting new record:**

INSERT INTO logistics

(D\_ID,O\_ID, S\_ID, S\_Loc)

VALUES

(12114,56781,534103,"NJ"),

(12115,56782,534102,"CJ");

