****

**Amateras Business Database**

**COURSE NUMBER: CPSC\_50900\_005**

**COURSE TITLE: DATABASE SYSTEMS**

**STUDENT NAME: MANIKANTA NAKKA**

**STUDENT ID: L30063676**

**INSTRUCTOR’S NAME: Matthew CLAVELLI**

**INDEX**

* 1. **Introduction**
  2. **Relational Database Design [ Description of entities and their attributes ]**
  3. **Data Sources**
  4. **Structures of the tables related to each entity**
  5. **Data Definition Language Scripts**
  6. **Data Manipulation Language Scripts**
  7. **Indexes**
  8. **Views**
  9. **Triggers**
  10. **Transaction**
  11. **Security**
  12. **Locking**
  13. **Backup**
  14. **Python programming**
  15. **Bonus work**

**1.16 References**

**1.1 Introduction**

**I have a small-scale business named Amateras which deals with selling Laptops. My business deals with selling laptops of various brands such as Dell, Lenovo, Asus, and Sony.  
 I opted to create a database which keeps track of several data related entities, such as Laptop Basic Details, Customer data, Employee data, Sales Data, and also Laptop system specifications data.   
This data will help me in keeping a track of laptops models I have in my inventory, and also keep track of the amount of units I’ve sold over the period of time. I need to keep track of my employees and keep track of their shift hours and the role they have to fulfill while they are in my store.   
Customer data will help me keep track of the laptop sold to a customer. This includes laptop id, customer name, date on which the product was sold and also the cost for which the customer has bought it. This sales data will help me in claiming any warranty related to the laptop, and also it can be used when the customer wants to return the product because of dissatisfaction.   
And since I need to store all details of a laptop product, sorted by the product ID. This data will contain key specifications about the laptop. A customer wants to know technical specifications about the product he wants to buy, this data will come in handy to provide better customer experience while negotiating the product.**

**I will manually enter the data into the database using an open source database tool called phpMyAdmin. This tool is written is PHP and offers various features for managing the database such as creating, importing, exporting and displaying the data stored in my database.**

**The data such as Laptop details will be used by customers to browse all the laptop modules which the store has to offer.  
While the employees and sales data will help the business to keep track of their employees and sales.  
Customer details data will be helpful in maintaining a customer service relationship, which can be useful while claiming a warranty or while returning the product.**

**This data is very critical to my business as it manages the above points which I have mentioned.**

**My Database files are located at:  
https://github.com/ssj4869/amateras\_business\_database**

**1.2 Relational Database Design**

**Description of Entities and their Attributes:**

**Entity [ 1 ] - Laptops:**

**Description: This Entity will store basic information about a laptop product. This Entity will help in maintaining another entity called Laptop\_data which consists of specifications about the laptop**

**Attributes:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Attribute** | **Data Type** | **Description** |
| **1** | **model\_id**  **(PRIMARY KEY)** | **VARCHAR** | **This will uniquely identify the laptop models. The data type is a combination of both numbers and characters: Ex: A12B22** |
| **2** | **model\_name** | **CHAR** | **This attribute will store the Laptop Name** |
| **3** | **model\_company** | **CHAR** | **This attribute will save the company name of laptop, for example: Dell, Asus, Lenovo, Apple** |
| **4** | **model\_cost** | **Float** | **This attribute will save the cost of the laptop product. I have used float because the cents value in price also counts. Example: 599.59. The currency used here is USD** |

**Entity [ 2 ] - laptop\_data**

**Description: This Entity stores detailed information about laptop specifications. Such as the storage size, Ram size, the type of operating system, the processor used.**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Attribute** | **Data Type** | **Description** |
| **1** | **model\_id**  **(Foreign Key)** | **VARCHAR** | **This attribute is liked with Entity Laptop, acts as foreign key. It is a primary key in Laptop entity** |
| **2** | **model\_processor** | **VARCHAR** | **This attribute includes both processor model id and full name of the processor**  **Example: Intel Core i7 11th gen 1133** |
| **3** | **model\_ram\_size** | **INT** | **Specifies the SIZE of ram ( in Gigabytes ) installed on the laptop; EX: 16** |
| **4** | **model\_HDD\_size** | **VARCHAR** | **specifies the SIZE of HDD in Gigabytes or Terrabytes:  example 1 -: 512GB example 2 -: 2TB** |
| **5** | **model\_operating\_system** | **VARCHAR** | **This specifies the type of operating system installed on the laptop, which also includes version number of the operating system Example (1): Windows 11 – HOME Edition Example (2): Kali Linux – Version 10** |
| **6** | **model\_screen\_size** | **FLOAT** | **specifies the screen size of the laptop. I used float because some of the laptop screen sizes also includes the dot value. All the size mentioned here are measured in inches Example 1: 14.4**  **Example 2: 13.0** |

**ENTITY [ 3 ] – customer\_data  
Description: This entity stores the basic customer details such as the name, the model\_id which they have bought, the cost of the laptop, purchase date and mode of payment.  
  
Attributes:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Attribute** | **Data Type** | **Description** |
| **1** | **model\_id** | **VARCHAR** | **Linked with Laptop\_data attribute, contains the laptop model id** |
| **2** | **model\_cost** | **Float** | **stores the cost of laptop in USD. Ex : 599.65** |
| **3** | **purchase\_date** | **DATE** | **This stores the date on which the customer has purchased the product** |
| **4** | **mode\_of\_payment** | **CHAR** | **Stores the mode of payment, example cash (or) card. If the payment is done in cash then while returning the product the payment will be made back to the customer in cash. Or else he will get back his money on his card.** |
| **5** | **customer\_name** | **CHAR** | **this stores the formal name of the customer who has purchased our product. Example: Jhon ( or ) Mike Hussey** |

**Entity [ 4 ] – Sales\_data:  
This attribute contains basic sales data of the products which I’ve sold including model id of the product, cost of the product and on which date it has been sold.**

**Attributes:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Attribute** | **Data Type** | **Description** |
| **1** | **model\_id** | **VARCHAR** | **Linked with Laptop\_data attribute, contains the laptop model id** |
| **2** | **model\_name** | **VARCHAR** | **contains the laptop name. such as ASUS ROG GEN 5, Microsoft Surface PRO 2** |
| **3** | **sale\_date** | **date** | **contains the date on which the product was sold** |
| **4** | **sale\_cost** | **float** | **stores the cost of laptop in USD** |

**Entity [ 5 ] – employee\_data:  
This attribute contains the basic information about the supporting staff of our company. Includes name, the role in our store, a unique id assigned for him. I use this data to keep track of the employees working in our store. I manually store this data in the database**

**Attributes:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Attribute** | **Data Type** | **Description** |
| **1** | **emp\_id**  **( PRIMARY KEY )** | **INT** | **We use this emp id to assign a unique ID to each of our employee working in our store.** |
| **2** | **emp\_name** | **CHAR** | **This attribute stores the name of employee** |
| **3** | **emp\_role** | **CHAR** | **contains the designation of the employee – example: Sales Executive or Load Manager or Store Manager** |
| **4** | **joining\_date** | **date** | **I use this for storing on which date the employee has joined our store** |

**Entity [ 6 ] – employee\_cost\_details**

**I use this data to store information about the employee’s working hours, How many hours he works in our store per week and also how much is their billing per week. This Entity is liked with employee\_data entity.**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Attribute** | **Data Type** | **Description** |
| **1** | **emp\_id** | **INT** | **Acts as foreign key, is linked with employee\_data entity** |
| **2** | **emp\_work\_hours** | **VARCHAR** | **This stores the employee’s shift hours in simple format such as 7 AM – 12PM** |
| **3** | **emp\_hours\_per\_week** | **INT** | **This stores how many hours the employee should work per week. This will be used in calculation of his billing per week** |
| **4** | **emp\_payperhour** | **Float** | **stores how much the employee will be earning per hour** |

**1.3 Data Sources:**

**Since I run this business, I will enter my employee data ( An entity ), which contains basic details about an employee such as his Name, his role in our business, his contact number, the date on which he/she has joined our business.  
Every employee is assigned a unique ID, this will help in differentiating all other employees.  
I could’ve used SQL’s AUTO\_INCREMENT feature here, but that would be plain and boring.   
So I created a 4 digit employee ID randomly which contains numeric value.**

**To keep track of my employee’s billing details I created another entity called employee\_billing\_details**

**I will use this entity to store billing and work hours of an employee.**

**Attribute emp\_id is a foreign key linked with employee entity.**

**Emp\_work\_hours will be used for storing employee work timings**

**>> Example: 11 AM – 5 PM**

**Emp\_hours\_per\_week will store how many hours the employee will work in a single week.  
>> Example: 45**

**Emp\_payperhour is used for storing how much money my employee will earn per week ( in dollars )  
>> Example: 16**

**I use a software tool called phpMyAdmin, which is an open source database tool which helps in managing the database. This tool provides command line terminal as well as GUI for managing the data.  
I use this tool to read product\_information.csv ( An entity ) file. This file is sent to me by the dealer from where I get my stock from.  
This file contains all details regarding a laptop such as**

**- Model\_ID – This is a primary key which uniquely identify each laptop. The data type of this key is VARCHAR**

**- Model\_name – Name of the Laptop. Datatype: char**

**- model\_company – the company which manufactured the laptop. Datatype: char  
  
- model\_cost – the cost of laptop in dollars. Datatype: float  
  
- model\_processor – the full name of processor including the model number, equipped on the   
motherboard of the laptop. datatype: varchar.  
  
- model\_ram\_size – this specifies the ram size installed on the laptop. Datatype: INT**

**-model\_hdd\_size – specifies the size of storage in Gigabytes or Terabytes. Datatype: varchar**

**-model\_operating\_system – specifies the operating system installed on the laptop, including the version number. Datatype: varchar**

**-model\_screen\_size – this attribute specifies the display size of the laptop, in inches. Datatype: float**

**I created two entities called “Laptops” And “Laptop\_data”. The laptops entity is related to product\_information entity. Model\_id is a primary key attribute which connects Laptop, product\_information and laptop\_data entities.**

**I use these two entities to store all the laptop information sent to me by dealer.  
The laptop entity stores very brief details about the product such as the Laptop ID, model name, model company and also the cost of the product.  
I use SQL SELECT command to pick all these attributes from the product\_information table which I have created.  
  
SQL QUERY: SELECT model\_id, model\_name,model\_company,model\_cost FROM product\_information**

**By doing so, I will store the result data set in a file called laptops.csv.**

**I will then go to enter this data using phpmyadmin export feature to load all the result set data into the laptops table.**

**My next step will be to import other key attributes from product\_information file to load them into laptop\_data entity**

**This laptop\_data entity will store key specifications of the product such as laptop id, processor, ram size, storage size, operating system installed on the laptop and screen size.**

**Here in this entity, Laptop ID acts as a foreign key, which is linked to Laptop entity.**

**I use the following query to fetch key specifications of the product from product\_information file:  
SELECT model\_id,model\_processor,model\_ram\_size,model\_hdd\_size,model\_operating\_system,model\_screen\_size FROM product\_information;**

**I will store the result set of this query into laptop\_data table by using export and import feature provided by phpmyadmin.**

* **Next, I have created another entity called sales\_data.   
  I will use this entity to keep track of the products I have sold. I store the data manually in this entity which helps me in calculating the total sale cost of the products I’ve sold.**
* **This data consists of model\_id, which is a foreign key linked with Laptop entity, model\_name, sale\_date which stores the date on which the product was sold and sale\_cost of the product.**
* **I use sale\_cost.csv file to add data separated by delimiter comma.  
  Example: LEN1A2, Lenovo Note Book Pro, 2021-01-11,659.99**

**The last entity in my database is customer\_data**

**I use this entity to keep track of the customers who have purchased my product.**

**I use this data when a customer would want to claim warranty of the product he has purchased, or while returning the product in case of any damage while delivery.**

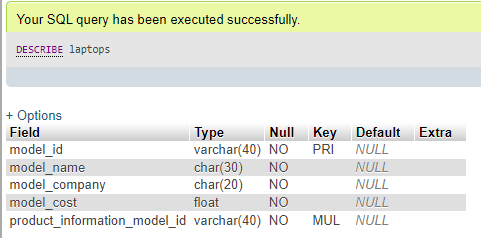
**Model\_id is a foreign key attribute linked with Laptops entity.**

**Model\_cost is used to store the cost of the product while buying.**

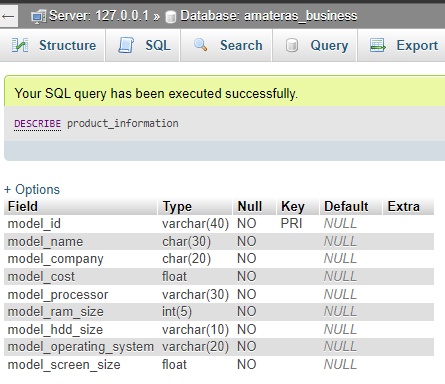
**Mode\_of\_payment is an entity which will store the how the customer has purchased the product? Is it by cash or card?  
this entity is important while refunding the money to the customer in case he decides to return the product. If the customer has paid the money through a card, the amount will be credited to his card, or else we will hand him the cash. This is important to balance the cash and credit economy of the store.**

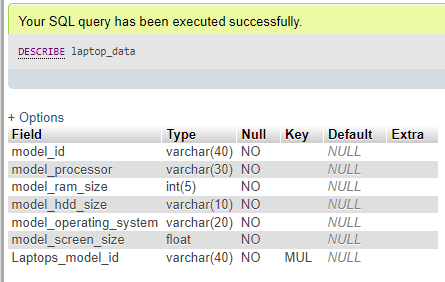
**Customer\_name will store the name of the customer who purchased our product and purchase\_date will store the date on which the customer has bought the product.**

**1.4 Structure of the tables related to each entity:**

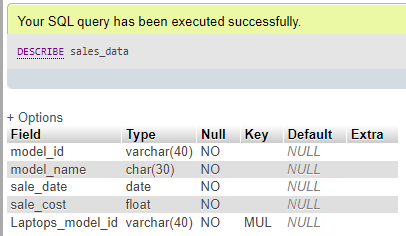
**>> Entity name: Laptops  
**

**>> Entity name: product\_information**

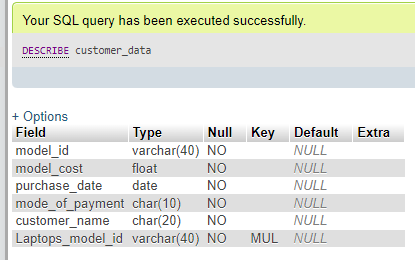
****

**>> Entity name: laptop\_data  
  
**

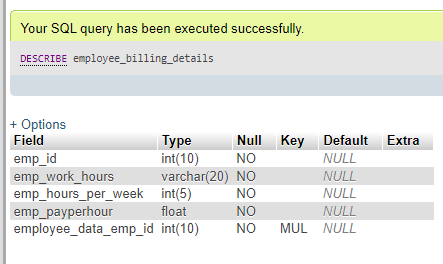
**>> Entity name: sales\_data**

****

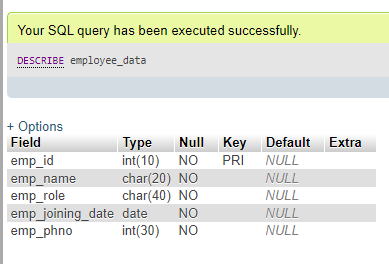
**>> Entity name: customer\_data**

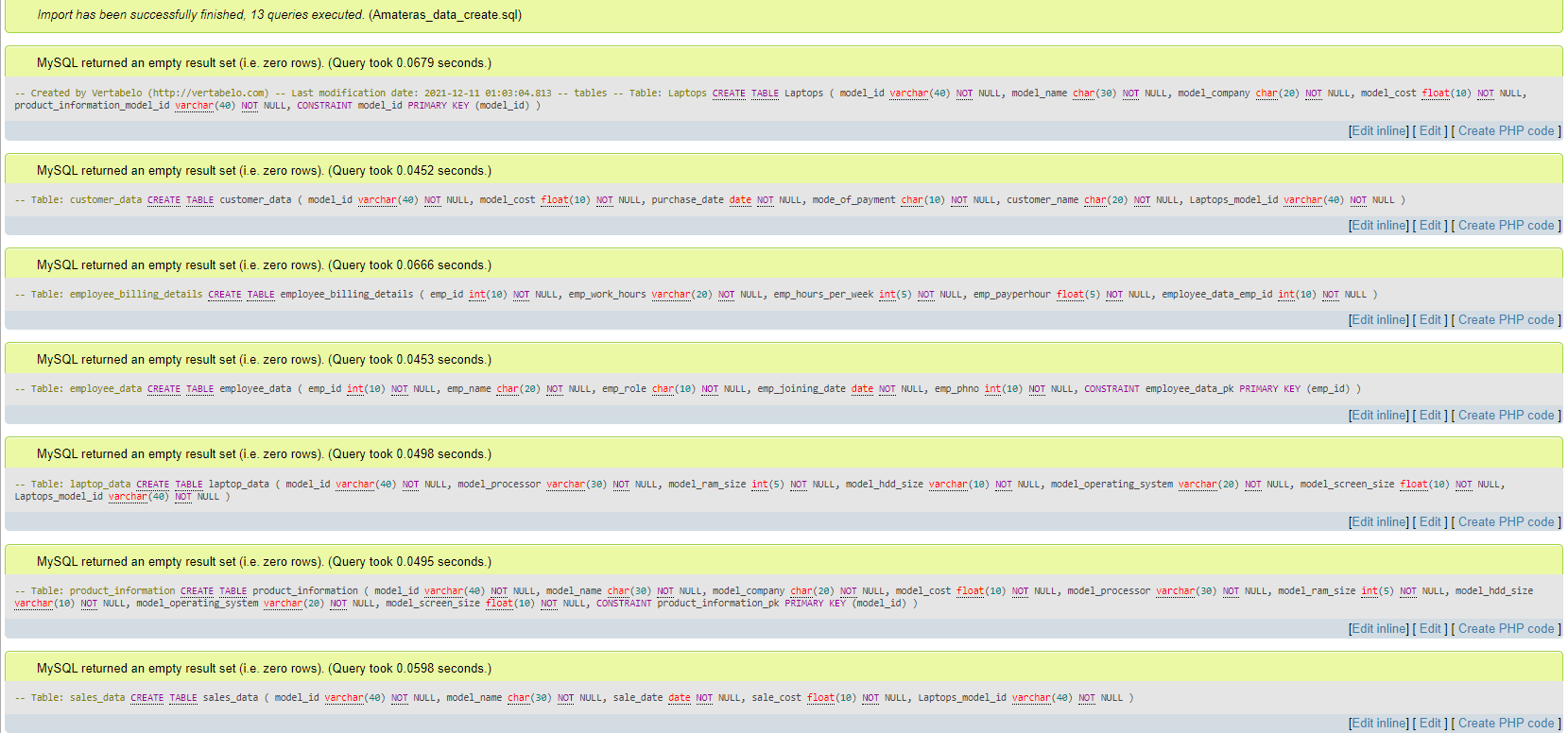
****

**>> Entity name: employee\_billing\_details**

****

**>> Entity name: employee\_data**

****

**1.5 Data Definition Language Scripts:**

**SQL Table creation using .sql script generated by vertebelo**

**1.5.1 SQL Commands Used for populating tables:**

**To add a value to a table, we use the SQL command called INSERT**

**The INSERT INTO statement is used to insert new records in a table.**

**INSERT INTO Syntax**

**It is possible to write the INSERT INTO statement in two ways:**

**1. Specify both the column names and the values to be inserted:**

**2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:**

**INSERT INTO table\_name**

**VALUES (value1, value2, value3, ...);**

**1.5.2**

**Table Name: product\_information  
SQL COMMANDS:**

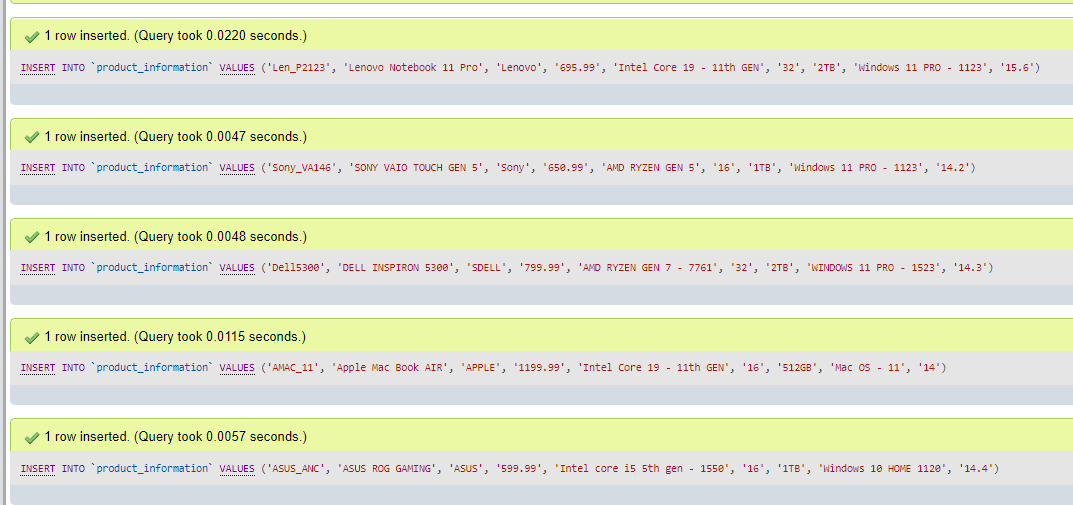
**INSERT INTO `product\_information` VALUES ('Len\_P2123', 'Lenovo Notebook 11 Pro', 'Lenovo', '695.99', 'Intel Core 19 - 11th GEN', '32', '2TB', 'Windows 11 PRO - 1123', '15.6')**

**INSERT INTO `product\_information` VALUES ('Sony\_VA146', 'SONY VAIO TOUCH GEN 5', 'Sony', '650.99', 'AMD RYZEN GEN 5', '16', '1TB', 'Windows 11 PRO - 1123', '14.2')**

**INSERT INTO `product\_information` VALUES ('Dell5300', 'DELL INSPIRON 5300', 'SDELL', '799.99', 'AMD RYZEN GEN 7 - 7761', '32', '2TB', 'WINDOWS 11 PRO - 1523', '14.3')**

**INSERT INTO `product\_information` VALUES ('AMAC\_11', 'Apple Mac Book AIR', 'APPLE', '1199.99', 'Intel Core 19 - 11th GEN', '16', '512GB', 'Mac OS - 11', '14')**

**INSERT INTO `product\_information` VALUES ('ASUS\_ANC', 'ASUS ROG GAMING', 'ASUS', '599.99', 'Intel core i5 5th gen - 1550', '16', '1TB', 'Windows 10 HOME 1120', '14.4')**

****

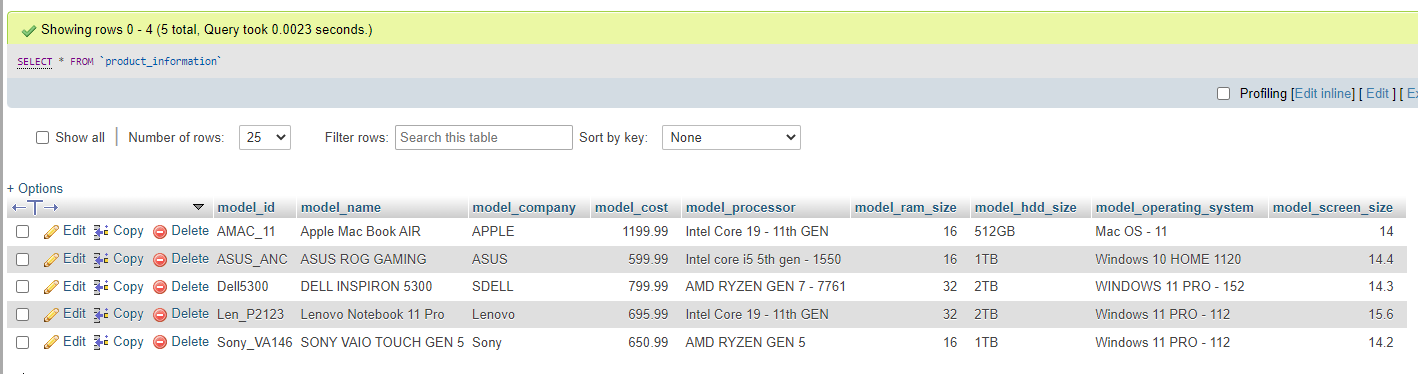
**The product\_information table has 9 columns. All the 9 values are required to insert a new record into the table.**

**Output of the above query will result in 5 rows created in the table.**

**To Display the Table, we use the sql command SELECT**

**SELECT \* FROM product\_information**

**Output:**

****

**Table name: laptops  
SQL COMMANDS:**

**INSERT INTO `laptops` VALUES ('ASUS\_ANC', 'ASUS ROG GAMING', 'ASUS', '599.99', 'ASUS\_ANC')**

**INSERT INTO `laptops` VALUES ('AMAC\_11', 'Apple Mac Book AIR', 'APPLE', '1199.99', 'AMAC\_11')**

**INSERT INTO `laptops` VALUES ('Dell5300', 'DELL INSPIRON 5300', 'SDELL', '799.99', 'Dell5300')**

**INSERT INTO `laptops` VALUES ('Len\_P2123', 'Lenovo Notebook 11 Pro', 'Lenovo', '695.99', 'Len\_P2123')**

**INSERT INTO `laptops` VALUES ('Sony\_VA146', 'SONY VAIO TOUCH GEN 5', 'Sony', '650.99', 'Sony\_VA146')**

**The laptops table has 5 columns, while the 5th column stores the foreign key of product\_information table**

****

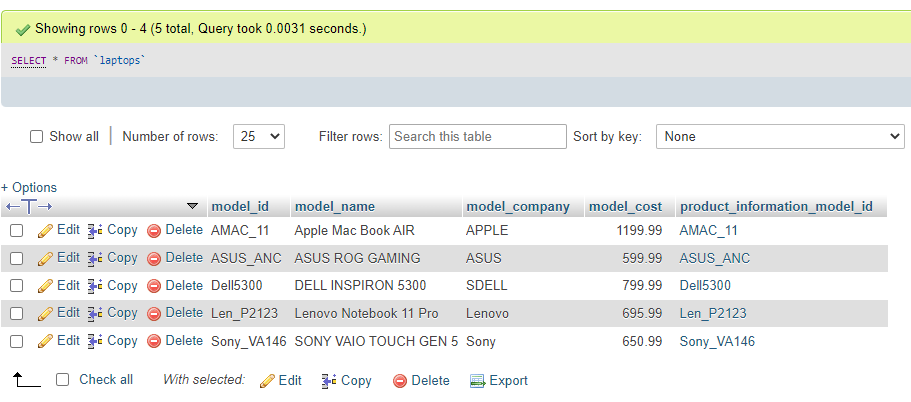
**We use INSERT command to create records in the table.**

**The output of the above query will result in 5 rows created.**

**To display the output, we use SELECT command.**

**SELECT \* FROM laptops**

**Output:**

****

**Table name: laptop\_data**

**SQL COMMANDS:**

**INSERT INTO `laptop\_data` VALUES ('AMAC\_11', 'Intel Core 19 - 11th GEN', '16', '512GB', 'Mac OS - 11', '14', 'AMAC\_11')**

**INSERT INTO `laptop\_data` VALUES ('ASUS\_ANC', 'Intel core i5 5th gen - 1550', '16', '1TB', 'Windows 10 HOME 1120', '14.4', 'ASUS\_ANC')**

**INSERT INTO `laptop\_data` VALUES ('Dell5300', 'AMD RYZEN GEN 7 - 7761', '32', '2TB', 'WINDOWS 11 PRO - 152', '14.3', 'Dell5300')**

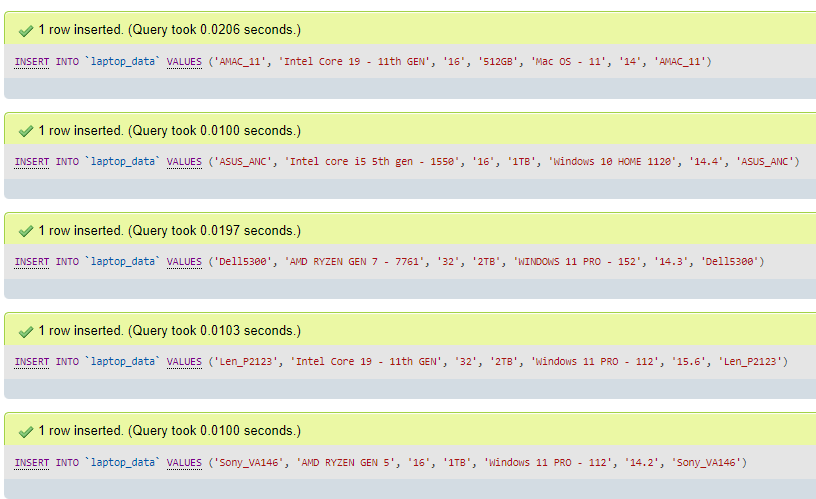
**INSERT INTO `laptop\_data` VALUES ('Len\_P2123', 'Intel Core 19 - 11th GEN', '32', '2TB', 'Windows 11 PRO - 112', '15.6', 'Len\_P2123')**

**INSERT INTO `laptop\_data` VALUES ('Sony\_VA146', 'AMD RYZEN GEN 5', '16', '1TB', 'Windows 11 PRO - 112', '14.2', 'Sony\_VA146')**

**The last column in the table contains primary key from Laptops Table**

**We use INSERT command to create records in the table.**

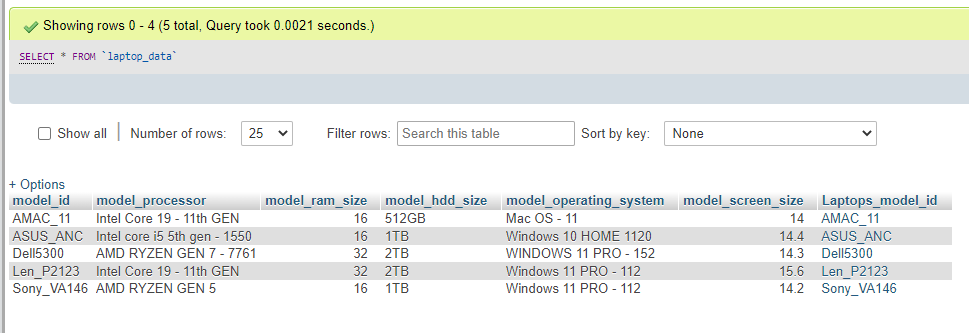
**The output of the above query will result in 5 rows created.**

****

**To display the output, we use SELECT command.**

**SELECT \* FROM laptop\_data**

**Output:**

****

**Table name: sales\_data**

**SQL STATEMENTS:**

**INSERT INTO `sales\_data` VALUES ('Len\_P2123', 'Lenovo Notebook 11 Pro', '2021-12-08', '695.99', 'Len\_P2123')**

**INSERT INTO `sales\_data` VALUES ('ASUS\_ANC', 'ASUS ROG GAMING', '2021-12-01', '599.99', 'ASUS\_ANC')**

**INSERT INTO `sales\_data` VALUES ('Sony\_VA146', 'SONY VAIO TOUCH GEN 5', '2021-11-09', '650.99', 'Sony\_VA146')**

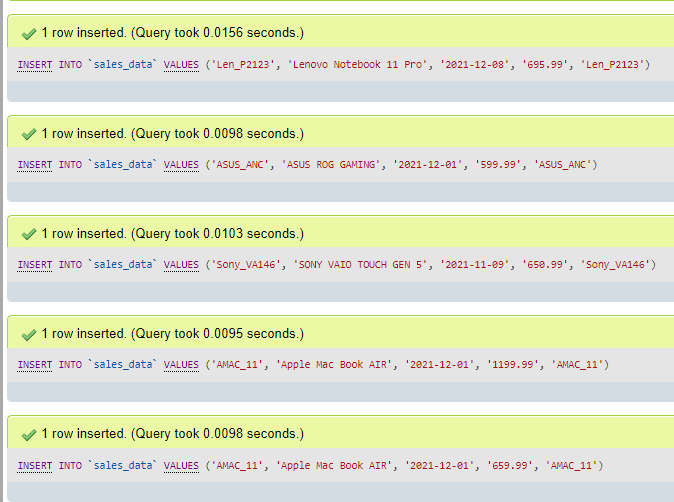
**INSERT INTO `sales\_data` VALUES ('AMAC\_11', 'Apple Mac Book AIR', '2021-12-01', '1199.99', 'AMAC\_11')**

**INSERT INTO `sales\_data` VALUES ('AMAC\_11', 'Apple Mac Book AIR', '2021-12-01', '659.99', 'AMAC\_11')**

**We use INSERT command to create records in the table.**

**The last column of the table contains primary key from the table Laptop.**

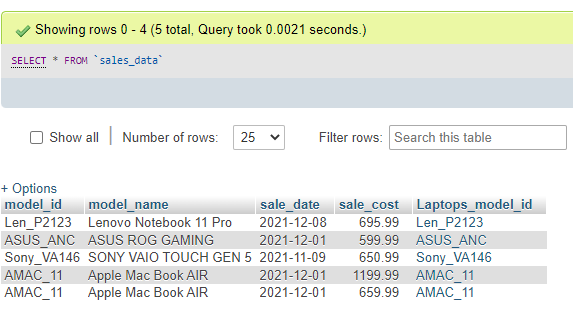
**The output of the above query will result in 5 rows created.**

****

**To display the output, we use SELECT command.**

**SELECT \* FROM sales\_data**

**OUTPUT:**

****

**Table Name: employee\_data**

**SQL STATEMENTS:**

**INSERT INTO `employee\_data` VALUES ('1003', 'Gabriel Jesus', 'Cashier', '2021-02-11', '9874563219')**

**INSERT INTO `employee\_data` VALUES ('1320', 'Robert Lewandowski', 'Customer service', '2021-01-01', '1549874563')**

**INSERT INTO `employee\_data` VALUES ('1325', 'Sarah Joseph', 'Sales Executive', '2021-05-11', '9784632150')**

**INSERT INTO `employee\_data` VALUES ('1413', 'Daniel James', 'Database manager', '2021-06-12', '6589741230')**

**INSERT INTO `employee\_data` VALUES ('1987', 'Williamson kane', 'Store Manager', '2020-11-11', '8456321029')**

****

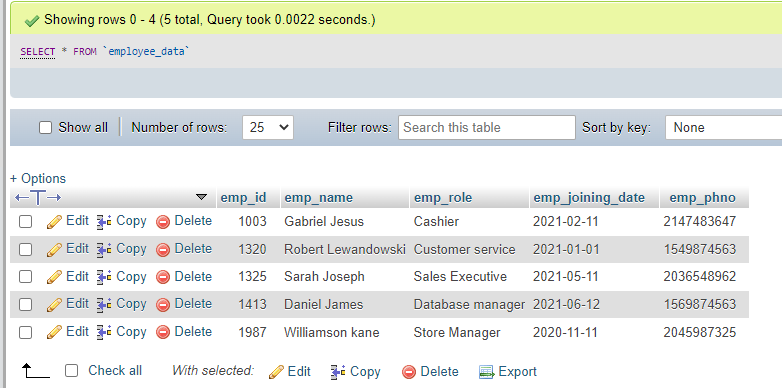
**We use INSERT command to create records in the table.**

**The output of the above query will result in 5 rows created.**

**To display the output, we use SELECT command.**

**SELECT \* FROM employee\_data**

**Output:**

****

**Table Name: employee\_billing\_details**

**SQL STATEMENTS:**

**INSERT INTO `employee\_billing\_details` VALUES ('1003', '11AM - 5PM', '40', '15', '1003')**

**INSERT INTO `employee\_billing\_details` VALUES ('1320', '10AM - 6PM', '42', '17', '1320')**

**INSERT INTO `employee\_billing\_details` VALUES ('1325', '11AM - 5PM', '40', '17', '1325')**

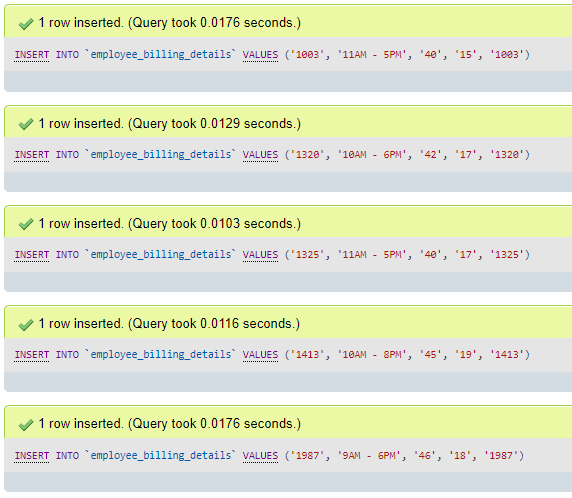
**INSERT INTO `employee\_billing\_details` VALUES ('1413', '10AM - 8PM', '45', '19', '1413')**

**INSERT INTO `employee\_billing\_details` VALUES ('1987', '9AM - 6PM', '46', '18', '1987')**

**We use INSERT command to create records in the table.**

**The output of the above query will result in 5 rows created.**

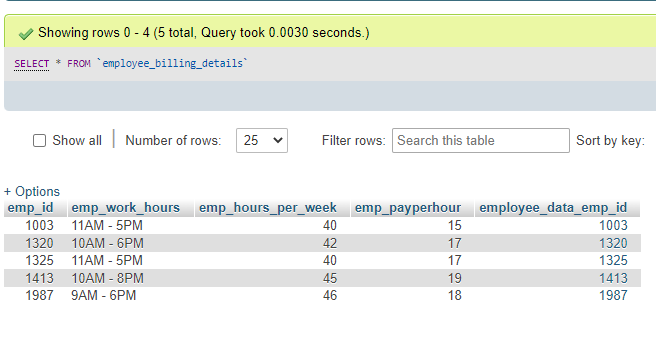
**The last column of the table will contain primary key linked with employee\_details table, and here in this table, it is emp\_id**

****

**To display the output, we use SELECT command.**

**SELECT \* FROM employee\_billing\_details**

**OUTPUT:**

****

**TABLE NAME: customer\_data**

**SQL STATEMENTS:**

**INSERT INTO `customer\_data` VALUES ('Len\_P2123', '695.99', '2021-12-02', 'CASH', 'John Michael', 'Len\_P2123')**

**INSERT INTO `customer\_data` VALUES ('Sony\_VA146', '650.99', '2021-12-01', 'CARD', 'Robert Joseph', 'Sony\_VA146')**

**INSERT INTO `customer\_data` VALUES ('ASUS\_ANC', '599.99', '2021-12-09', 'CARD', 'Williams Maria', 'ASUS\_ANC')**

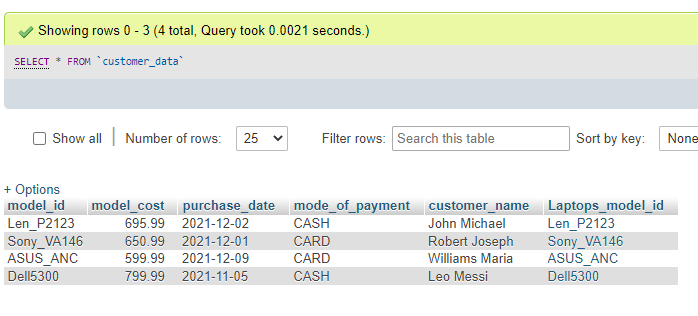
**INSERT INTO `customer\_data` VALUES ('Dell5300', '799.99', '2021-11-05', 'CASH', 'Leo Messi', 'Dell5300')**

****

**We use INSERT command to create records in the table.**

**The output of the above query will result in 4 rows created.**

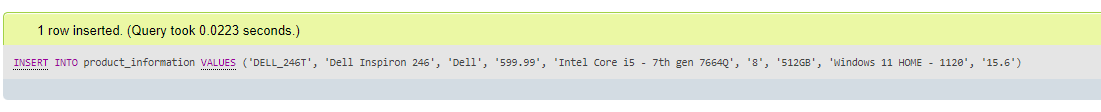
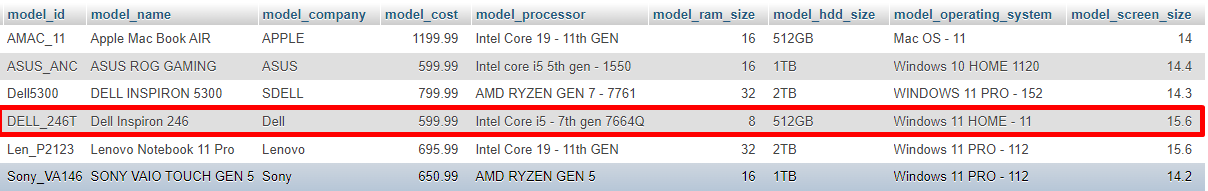
**OUTPUT:**

****

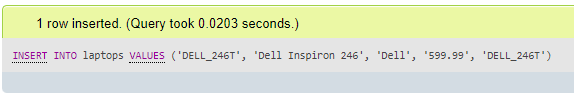
**1.6 DATA Manipulation Language Scripts:**

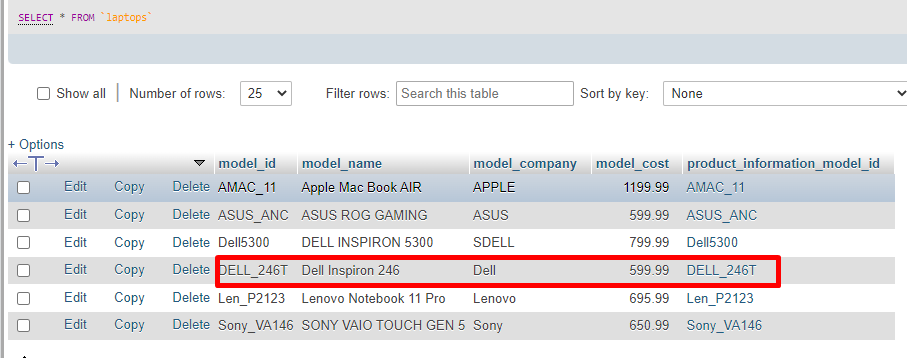
1. **INSERT Statement –**

**INSERT INTO product\_information VALUES**

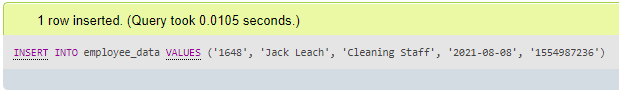
**('DELL\_246T', 'Dell Inspiron 246', 'Dell', '599.99', 'Intel Core i5 - 7th gen 7664Q', '8', '512GB', 'Windows 11 HOME - 1120', '15.6');**

**INSERT #2**

**INSERT INTO laptops VALUES ('DELL\_246T', 'Dell Inspiron 246', 'Dell', '599.99', 'DELL\_246T');**

****

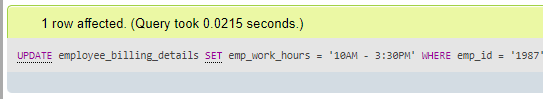
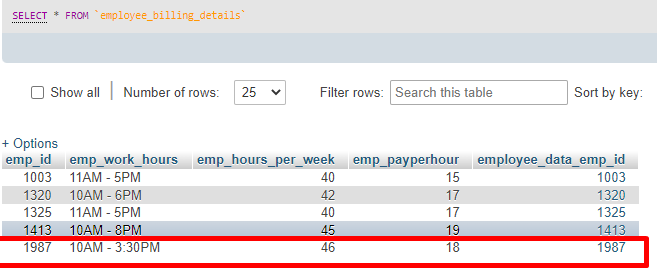
**INSERT #3 INSERT INTO employee\_data VALUES ('1648', 'Jack Leach', 'Cleaning Staff', '2021-08-08', '9746321598');**

****

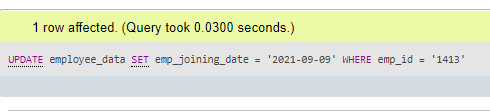
****

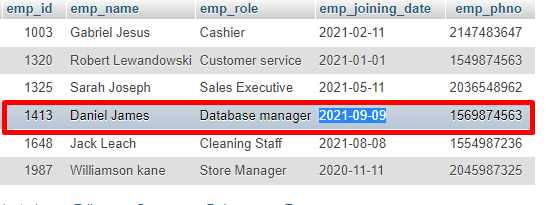
1. **UPDATE Statements:**

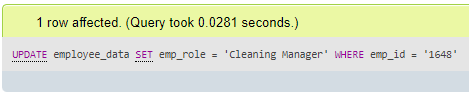
**UPDATE employee\_billing\_details SET emp\_work\_hours = '10AM - 3:30PM' WHERE emp\_id = '1987'**

****

**#2   
UPDATE employee\_data SET emp\_joining\_date = '2021-09-09' WHERE emp\_id = '1413';**

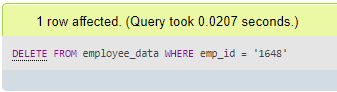
****

****

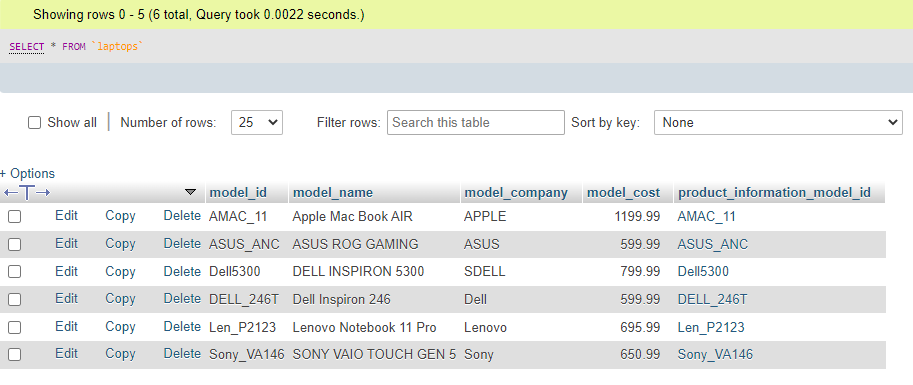
**#3  
UPDATE employee\_data SET emp\_role = 'Cleaning Manager' WHERE emp\_id = '1648';**

1. **SQL DELETE STATEMENT**

**DELETE FROM employee\_data WHERE emp\_id = '1648';**

****

1. **SQL Simple SELECT Statement**

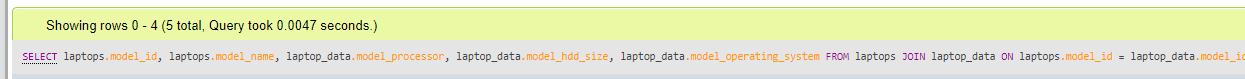
**SELECT \* FROM laptops**

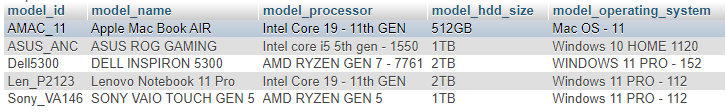
1. **SQL Join Statements**

**#1**

**SELECT laptops.model\_id, laptops.model\_name, laptop\_data.model\_processor, laptop\_data.model\_hdd\_size, laptop\_data.model\_operating\_system FROM laptops**

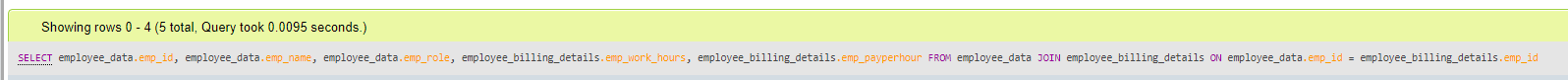
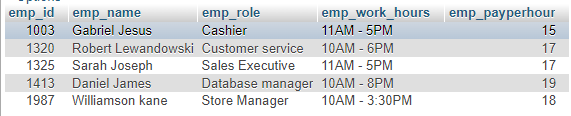
**JOIN laptop\_data ON laptops.model\_id = laptop\_data.model\_id**

****

****

**# 2**

**SELECT employee\_data.emp\_id, employee\_data.emp\_name, employee\_data.emp\_role, employee\_billing\_details.emp\_work\_hours, employee\_billing\_details.emp\_payperhour FROM employee\_data JOIN employee\_billing\_details ON employee\_data.emp\_id = employee\_billing\_details.emp\_id;**

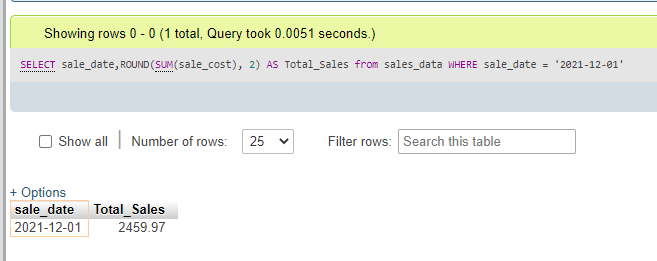
****

1. **SUMMARY Statements**

**I USE THIS QUERY TO Know how much sales I’ve earned on 2021-12-01**

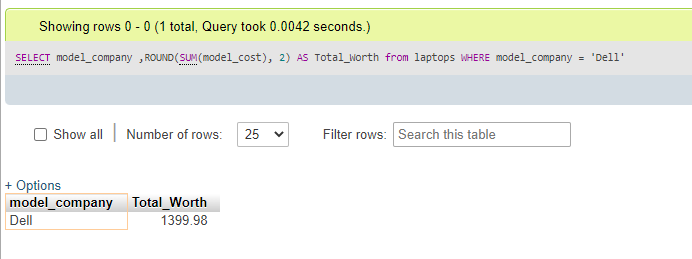
**#1 SQL**

**SELECT sale\_date,ROUND(SUM(sale\_cost), 2) AS Total\_Sales from sales\_data WHERE sale\_date = '2021-12-01';**

****

**#2 SQL   
I use this query to check the worth of the dell laptops which I have in my inventory.**

**SELECT model\_company ,ROUND(SUM(model\_cost), 2) AS Total\_Worth from laptops WHERE model\_company = 'Dell';**

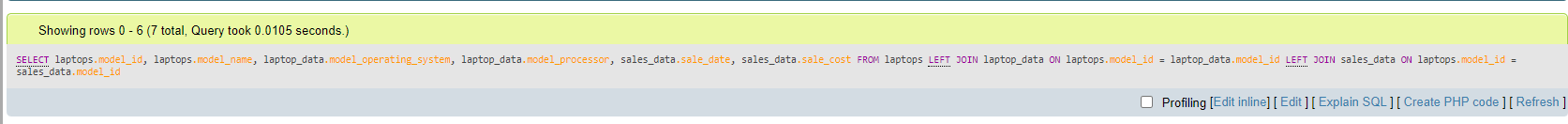
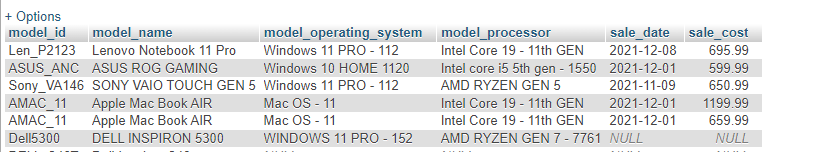
****

1. **Multi-table Query**

**SELECT laptops.model\_id, laptops.model\_name, laptop\_data.model\_operating\_system, laptop\_data.model\_processor, sales\_data.sale\_date, sales\_data.sale\_cost FROM laptops**

**LEFT JOIN laptop\_data ON laptops.model\_id = laptop\_data.model\_id**

**LEFT JOIN sales\_data ON laptops.model\_id = sales\_data.model\_id**

****

**I have joined 3 tables [ laptops, laptop\_data and sales\_data ] to get this query output.  
I have used LEFT JOIN to join two tables using the primary key model\_id which is shared among 3 of the above mentioned tables.**

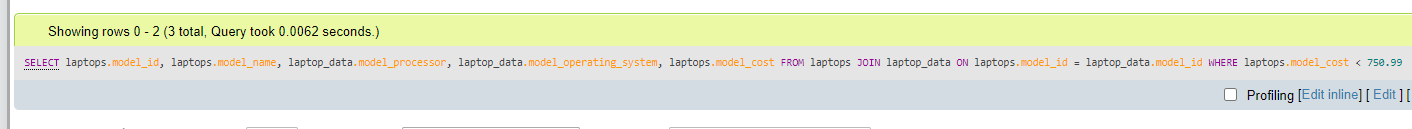
**In the above output, sale\_date and sale\_cost shows NULL becase the product has not been sold yet, thus the default value of that data will be NULL!**

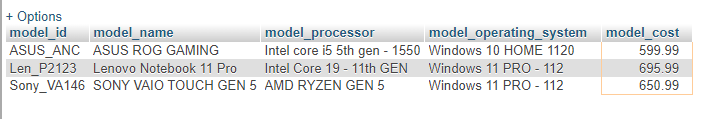
1. **Query of My Choice:**

**SELECT laptops.model\_id, laptops.model\_name, laptop\_data.model\_processor, laptop\_data.model\_operating\_system, laptops.model\_cost FROM laptops**

**JOIN laptop\_data ON laptops.model\_id = laptop\_data.model\_id**

**WHERE laptops.model\_cost < 750.99;**

****

****

**In the above query, I wanted to select all laptops which cost below 750.99**

**1.7 Indexes**

**The CREATE INDEX statement is used to create indexes in tables.**

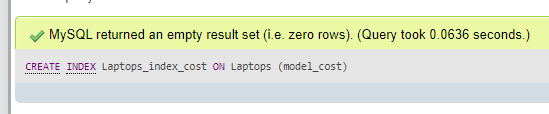
**Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.  
  
Index #1 –**

**Name: Laptops\_index\_cost**

**SQL: CREATE INDEX Laptops\_index\_cost ON Laptops (model\_cost);**

**Reason: I have created this index on laptop cost because every customer has a set themselves a fixed budget to buy a product. So the one of the first question a customer will ask to the sales person is   
“I’m looking to buy a good laptop which costs below 700. Do you have any suggestions?”**

**So Indexing on the model\_cost will help me to fetch my data very quickly.**

****

**Index #2 –**

**Name: Laptops\_index\_company**

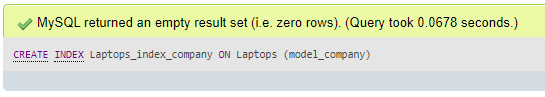
**SQL: CREATE INDEX Laptops\_index\_company ON Laptops (model\_company);**

**Reason: I have created this index on laptop company because every brand has it’s own value. And the customers while buying a product will prefer to buy only a particular product manufactured by Xyz Company. Example Dell (or) Apple (or) Lenovo**

**Let’s say, a customer queries to buy a Lenovo laptop only.. then indexing on model\_company will help in fetching all Lenovo laptops.**

**Thus, creating an index on model\_company will help me to fetch xzy laptop company within no time.**

**Indexing will help to fetch the data very fast!**

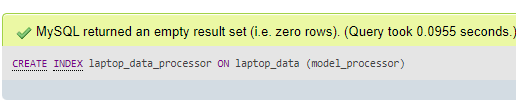
****

**Index #3 -**

**Name: laptop\_data\_processor**

**CREATE INDEX laptop\_data\_processor ON laptop\_data (model\_processor);**

**Reason: I have created this index on model\_processor because every customer wants his laptop to be fast enough to get his task done. Few of the customers have very specific request of the laptop with a processor of their choice..   
Let’s say, “I want to buy a Laptop which has the latest intel core i9 GEN”  
Thus, adding index on this attribute will help in fetching the specific processor laptops real quick.!**

****

**Indexing Conclusion:**

**From Index #1, #2 and #3, the performance of the fetching data can be improved a lot.**

**The reason I chose to index 3 specific columns from different tables is because,**

**Let’s say a customer comes up with a very specific query such as**

**“I’m looking to buy a Dell Laptop which costs below 750, and I want a model which has the intel i7 11th generation processor”**

**Here I have indexed 3 attributes “Model company”, “Model Cost” and “Model Processor”**

**So when such query arises, the database can fetch the result set within no time by using indexes.. And according to my research, a basic user of laptop who uses it for his education and to browse some digital media will not know much about technical specifications about the product. He/She will only know the popular brands such as Apple or Dell, and the processor he wants.**

**And Yeah, every customer will have to look at his budget while investing on something big item…**

**So, All the 3 indexes I have defined will help in fetching the data from database very quickly.**

**1.8 Views**

**In SQL, a view is a virtual table based on the result-set of an SQL statement.**

**A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.**

**View #1**

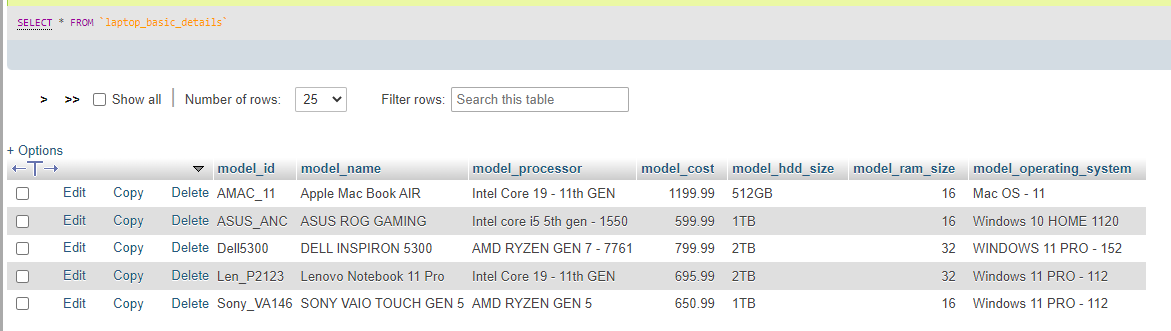
**Name: laptop\_basic\_details**

**SQL for creating view:**

**CREATE VIEW laptop\_basic\_details AS**

**SELECT laptops.model\_id, laptops.model\_name, laptop\_data.model\_processor, laptops.model\_cost, laptop\_data.model\_hdd\_size, laptop\_data.model\_ram\_size,laptop\_data.model\_operating\_system FROM laptops**

**JOIN laptop\_data ON laptops.model\_id = laptop\_data.model\_id**

**Output:  
SELECT \* FROM laptop\_basic\_details;**

**I have created this view because it will help in providing basic details of the product and few selected key specifications.**

**There are few users who only want to know the cost of the product, the manufacturer of It, the storage size and the size of storage the product provides.  
So out of a bunch of specifications of a laptop, there are very few and specific key specifications a customer would always want to check upon before buying a product.**

**This view will benefit the customer because of the inclusion of main key specifications of the product in one single view.**

**View #2**

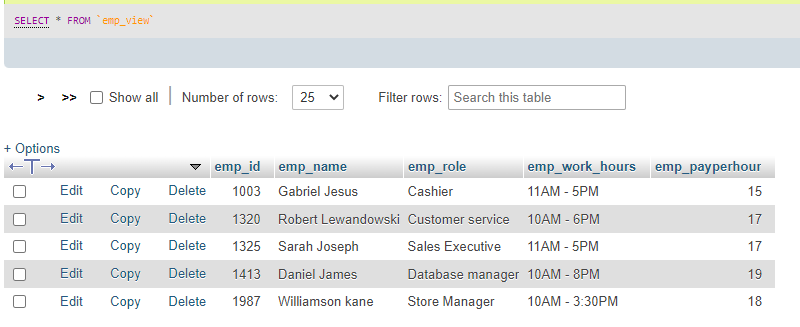
**Name: emp\_view**

**SQL for creating this view:**

**CREATE VIEW emp\_view AS**

**SELECT employee\_data.emp\_id, employee\_data.emp\_name, employee\_data.emp\_role, employee\_billing\_details.emp\_work\_hours, employee\_billing\_details.emp\_payperhour FROM employee\_data JOIN employee\_billing\_details ON employee\_data.emp\_id = employee\_billing\_details.emp\_id;**

**Output:  
Select \* FROM emp\_view**

****

**I have created this view for my own personal use. This view will join data from two tables emp\_data and emp\_billing\_details.**

**This view will fetch two important columns from emp\_billing\_details: emp\_work\_hours and emp\_payperhour**

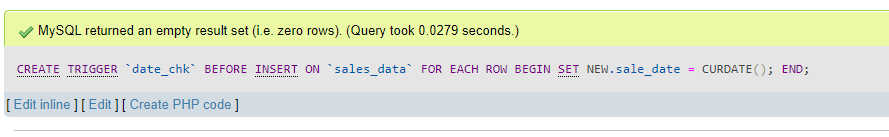
**With this view, I can know which important details such as an employee’s ID, his name, the role he plays in our business, his working hours and how much he will earn per hour.**

**Keeping these key attributes in one view will help me to know all these specific details in one particular table.**

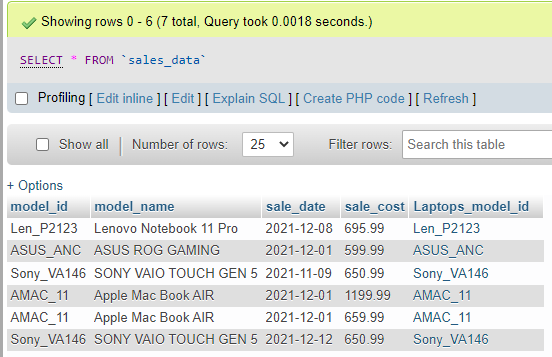
**This will help me in calculating his weekly payout. I will use a scripting language to remove String values from emp\_work\_hours and that will result in number of hours he will work per day. I can keep track of an employee with his name and his billing per hour and I can calculate his weekly payout with this view.**

**1.9 TRIGGERS  
Trigger statement:**

**CREATE TRIGGER `date\_chk` BEFORE INSERT ON `sales\_data` FOR EACH ROW BEGIN SET NEW.sale\_date = CURDATE(); END;**

****

**INSERT SQL:  
INSERT INTO sales\_data VALUES ('Sony\_VA146', 'SONY VAIO TOUCH GEN 5', '', '650.99', 'Sony\_VA146');**

****

**I have created a trigger for sale\_date attribute. This trigger will fire up when a record is inserted into the table ‘sales\_data’. This trigger will always call for current date and assign that date value to the record.**

**This will help me in not typing the current date every time, as I will always be inserting records at the end of business day, the function CURDATE() from MySQL will help in automatically storing current date in the sale\_date column.**

**In my Above INSERT statement, I have not mentioned any date.. I just left the column empty. That column will be managed by date\_chk trigger, which will set current date to the column which has been inserted into sales\_data table.**

**1.10 TRANSACTION**

**A transaction is a sequential group of database manipulation operations, which is performed as if it were one single work unit. In other words, a transaction will never be complete unless each individual operation within the group is successful. If any operation within the transaction fails, the entire transaction will fail.**

**Practically, you will club many SQL queries into a group and you will execute all of them together as a part of a transaction.**

**Properties of Transactions**

**Transactions have the following four standard properties, usually referred to by the acronym ACID −**

**Atomicity − This ensures that all operations within the work unit are completed successfully; otherwise, the transaction is aborted at the point of failure and previous operations are rolled back to their former state.**

**Consistency − This ensures that the database properly changes states upon a successfully committed transaction.**

**Isolation − This enables transactions to operate independently on and transparent to each other.**

**Durability − This ensures that the result or effect of a committed transaction persists in case of a system failure.**

**In MySQL, the transactions begin with the statement BEGIN WORK (or) START TRANSACTION and end with either a COMMIT or a ROLLBACK statement. The SQL commands between the beginning and ending statements form the bulk of the transaction.**

**SQL STATEMENTS WITH OUTPUT:**

**START TRANSACTION;**

**UPDATE sales\_data SET sale\_date = "2011-11-11"**

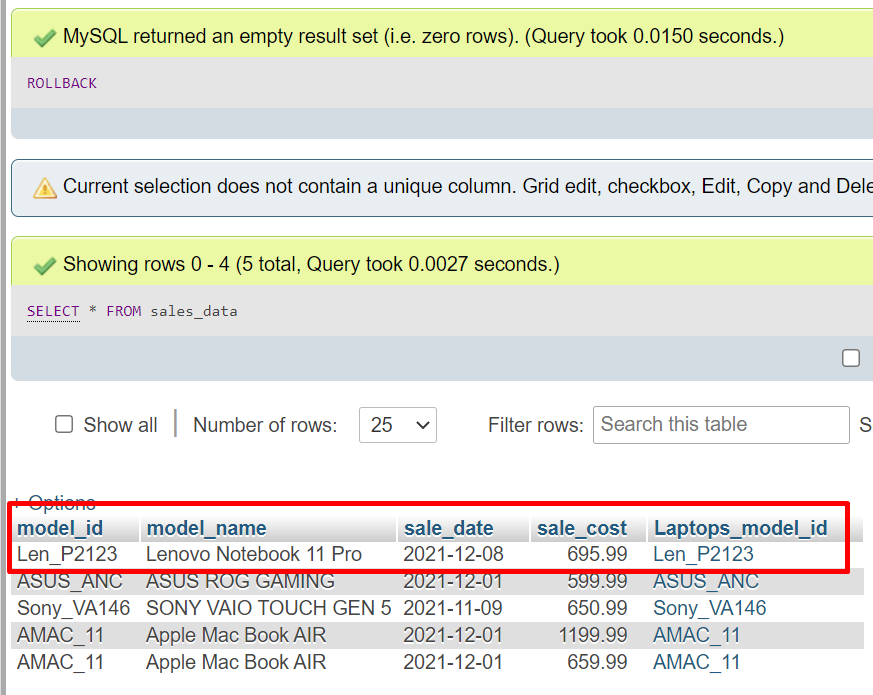
**WHERE model\_id = "Len\_P2123";**

**SELECT \* FROM sales\_data;**

**This part of query has been executed as a transaction. However we have not used COMMIT or ROLLBACK.   
COMMIT will make the transaction permanent, while ROLLBACK will restore the database to the previous state before UPDATE Operation.  
Here I will show the output After Performing INSERT Statement**

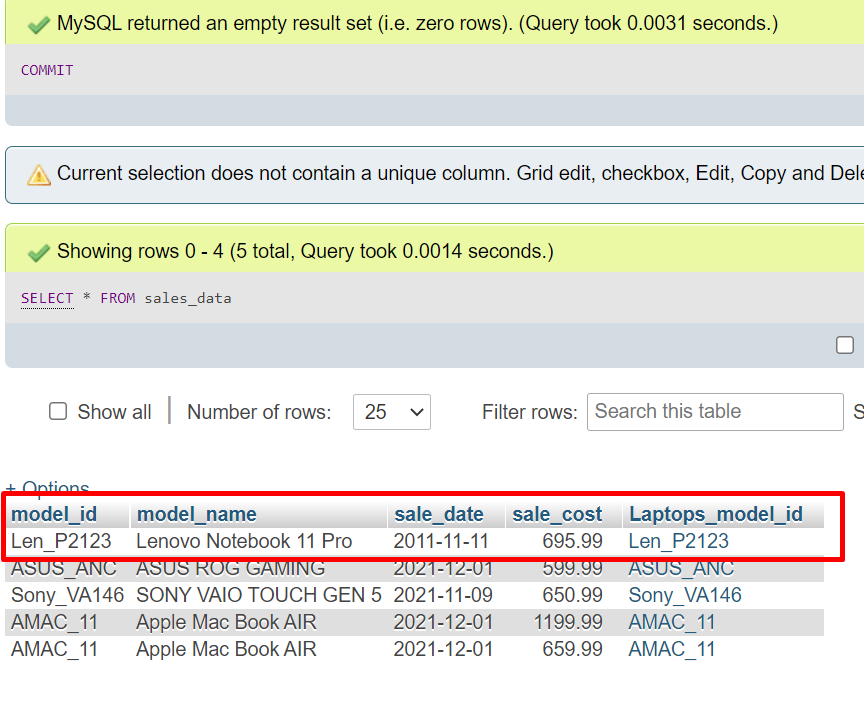
**However, this is only temporary as I have not performed COMMIT operation. COMMIT operation is needed to make permanent changes to a transaction.  
Now let’s say I do not like this update to happen and I want to go back to the previous value of sale\_date.. Here, I will use ROLLBACK to go back to previous state of the database.**

**SQL STATEMENT:  
 ROLLBACK;  
SELECT \* FROM sales\_data;**

****

**After using ROLLBACK; command, you can see in this output, the previous value of sale\_date has been restored… This is because ROLLBACK command will restore the state of database PRIOR to COMMIT operation.**

**Now let’s say I want these changes to be permanent.. In that case, I’ll use COMMIT statement to make these changes permanent.**

**  
SQL STATEMENT:   
COMMIT;  
SELECT \* FROM sales\_data**

**From the above output, we can see the sale\_date value has been updated and permanent changes have been made after using the SQL statement COMMIT.   
  
Conclusion:  
MYSQL transactions work as a single unit. It’s either all in or nothing. Execute everything or nothing.   
Don’t like the change and want to go back before previous committed state? USE ROLLBACK  
Want to make the changes permanent? Use SQL statement COMMIT**

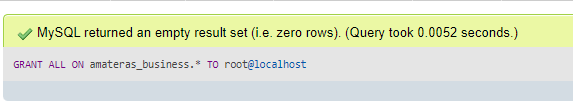
**1.11 Security**

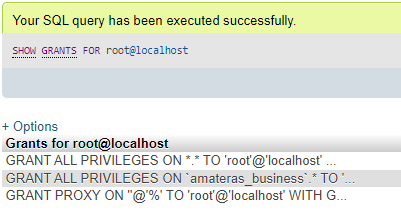
**Various users who are going to use my database:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **User** | **Permissions** |
| **1** | **root@localhost** | **ALL** |
| **2** | **accountant@localhost** | **INSERT, SELECT ON sales\_data table** |
| **3** | **customer@localhost** | **SELECT ON laptop\_data** |
| **4** | **vendor@localhost** | **SELECT,INSERT,UPDATE On product\_information** |

1. **Root  
   username: root@localhost  
     
   The root user is the owner of the business and the database. He has all privileges on all the tables. He can INSERT, UPDATE, DELETE, DROP, SELECT, CREATE, ALTER, TRUNCATE.  
   He can do whatever he wants with the database.   
   They can:  
   - Update all the tables using update command  
   - Create new data and tables  
   - Delete any data  
   - DROP an entire database or any table they want  
   - Can empty the whole data of a table.**

**SQL STATEMENTS: SHOW GRANTS for root@localhost;**

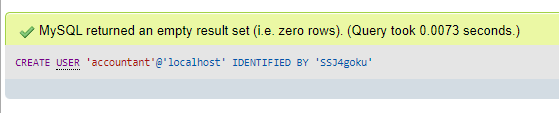
****

****

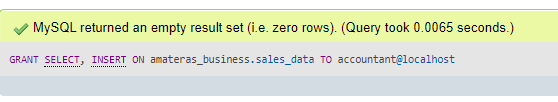
1. **Accountant**

**This user is responsible for managing sales data of our business. He role is to add new records to sales\_data table and to view the sales data to calculate total sales amount on any given particular day or overall of all sales data.**

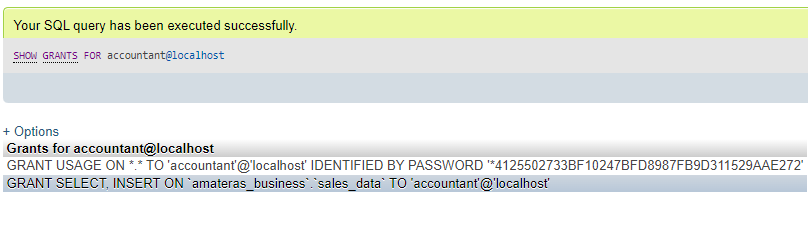
**>> First create a new user for accountant with password**

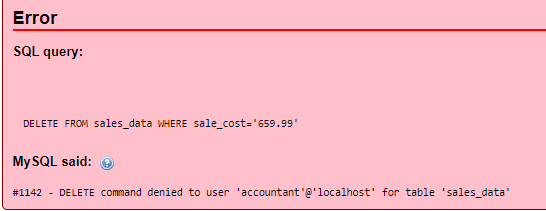
****

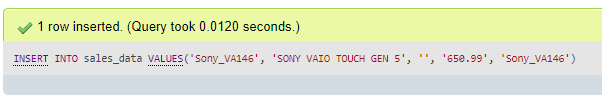
**>> Assign Grants to the user  
SQL STATEMENT: GRANT SELECT, INSERT ON amateras\_business.sales\_data TO accountant@localhost**

****

**SQL Statement: SHOW GRANTS FOR accountant@localhost**

**Let us try to delete a record in sales\_data table. The user accountant does not have permission to delete a record.  
SQL STATEMENT: DELETE FROM sales\_data WHERE sale\_cost = ‘659.99’;**

****

**LET us try to insert a record in sales\_data table.**

**We can see the response “1 row inserted”, which means INSERT statement, is working for the user Accountant.**

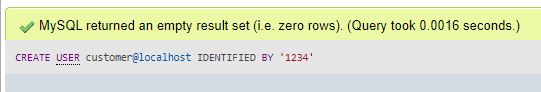
1. **Customer**

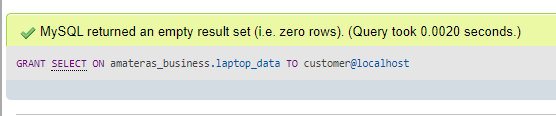
**This user is like any general customer, who can only view the laptops available.**

**The permission assigned to this user is SELECT.**

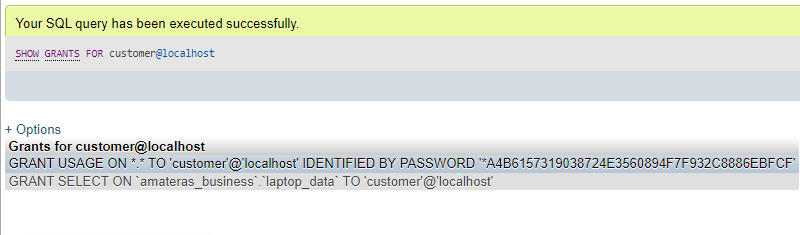
**He/She can only query the database table “laptop\_data” using SELECT command.**

**>> Create user:  
SQL: CREATE USER customer@localhost IDENTIFIED BY ‘1234’;**

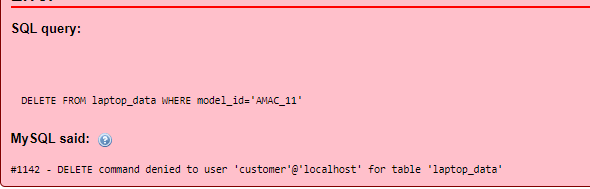
****

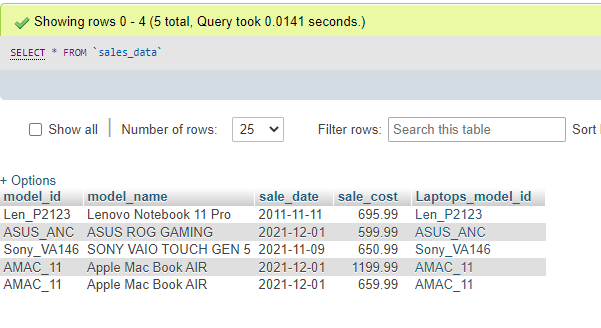
****

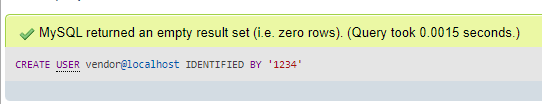
**>> GRANT SELECT permission to the user.  
SQL STATEMENT:  
GRANT SELECT ON amateras\_business.laptop\_data TO customer@localhost**

**View GRANTS for the user:  
SQL STATEMENT: SHOW GRANTS FOR customer@localhost**

**I tried to DELETE a record in laptop\_data. But since the user did only had the permission to use SELECT statement, Any other Commands will not work!**

**SQL STATEMENT: DELETE FROM laptop\_data WHERE model\_id=’AMAC\_11’;**

**However, the command SELECT will work for the customer user.**

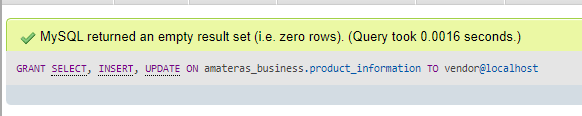
1. **Vendor**

**The vendor is the user, who is basically a supplier of products for our business. They have access to our database for the product\_information table.**

**SQL STATEMENT: CREATE USER vendor@localhost IDENTIFIED BY ‘1234’;**

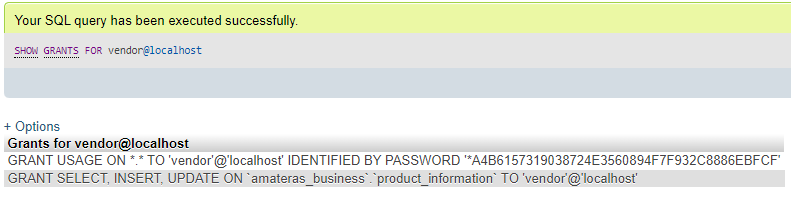
**SELECT, INSERT and UPDATE permissions are assigned to the vendor user.**

**The vendor can view the data, insert any new product into the table and also make changes to the data by using UPDATE command.**

**SQL STATEMENT:**

**GRANT SELECT, INSERT, UPDATE ON amateras\_business.product\_information TO vendor@localhost**

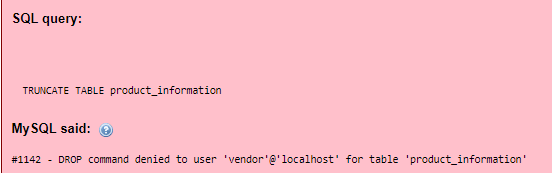
**>> View Permissions assigned to the user by the following command  
SHOW GRANTS FOR vendor@localhost;**

****

**Any other commands except SELECT, UPDATE and INSERT will not work for the user vendor.**

**Let us try to use TRUNCATE command for the vendor user.**

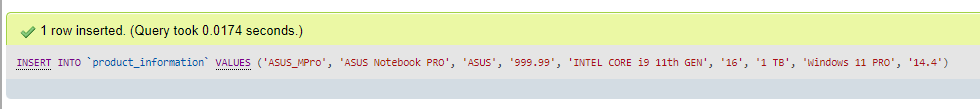
**SQL COMMAND:   
TRUNCATE TABLE product\_information**

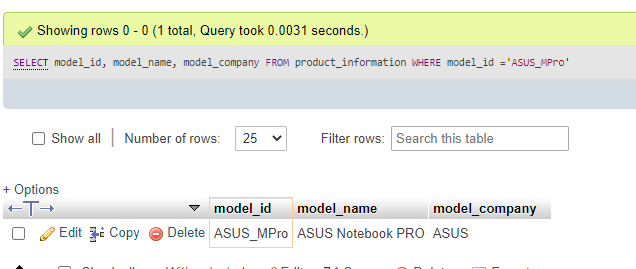
****

**As expected, the permission is denied for the user.**

**Let’s try to insert a new record.**

**SQL COMMAND:**

**INSERT INTO `product\_information` VALUES ('ASUS\_MPro', 'ASUS Notebook PRO', 'ASUS', '999.99', 'INTEL CORE i9 11th GEN', '16', '1 TB', 'Windows 11 PRO', '14.4') **

****

**We can see the INSERT and SELECT Statement is working perfectly for the user vendor. However other commands such as DROP and ALTER or any other commands will not work!**

**1.12 Locking**

**A lock is a mechanism associated with a table used to restrict the unauthorized access of the data in a table. MySQL allows a client session to acquire a table lock explicitly to cooperate with other sessions to access the table's data. MySQL also allows table locking to prevent it from unauthorized modification into the same table during a specific period.**

**A session in MySQL can acquire or release locks on the table only for itself. Therefore, one session cannot acquire or release table locks for other sessions. It is to note that we must have a TABLE LOCK and SELECT privileges for table locking.**

**Table Locking in MySQL is mainly used to solve concurrency problems. It will be used while running a transaction, i.e., first read a value from a table (database) and then write it into the table (database).**

**MySQL provides two types of locks onto the table, which are:**

**READ LOCK: This lock allows a user to only read the data from a table.**

**WRITE LOCK: This lock allows a user to do both reading and writing into a table.**

**MYSQL Lock Syntax:**

**LOCK TABLES table\_name [READ | WRITE];**

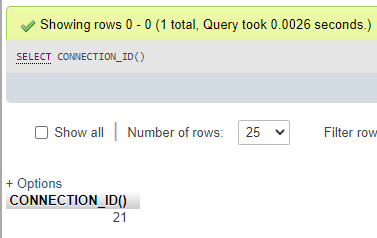
**We can also lock more than one table in MySQL by using a list of comma-separated table's names with lock types:**

**LOCK TABLES tab\_name1 [READ | WRITE],**

**tab\_name2 [READ | WRITE],.....;**

**We will first connect to the database and use the CONNECTION\_ID() function that gives the current connection id in the first session as follows:**

**SQL: SELECT CONNECTION\_ID();**

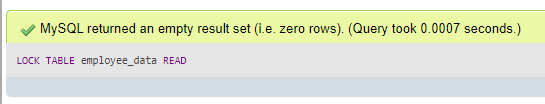
****

**So in first session, we are going to work with connection\_id 21.**

**We are going to lock employee\_data table on READ Mode.**

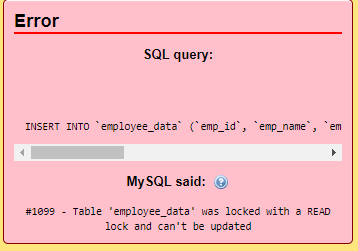
**My SQL Lock statement for READ:**

**LOCK TABLE employee\_data READ**

****

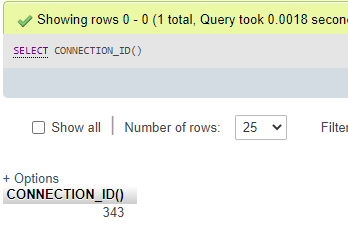
**Let’s try to insert a record in this table.**

**SQL Statement:  
INSERT INTO `employee\_data` VALUES ('1069', 'Michael David', 'Sales Executive', '2021-02-11', '1203256478');**

****

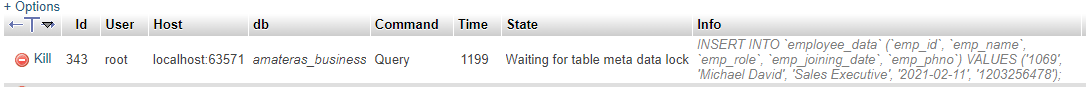
**Since the table is locked in READ-ONLY mode, write operation is not permitted until the lock is released or removed from the table.**

**Now let’s keep this connection alive and open a new connection.**

****

**INSERT DATA:**

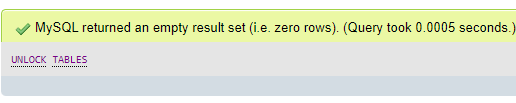
**SQL Statement:  
INSERT INTO `employee\_data` VALUES ('1069', 'Michael David', 'Sales Executive', '2021-02-11', '1203256478');**

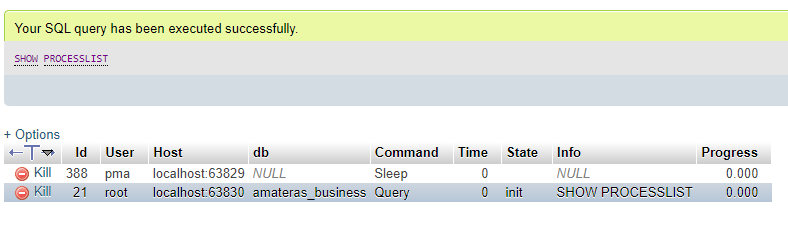
**The MySQL will return the following output.**

**And that’s how MySQL READ lock will prevent other connections from using the table when the read lock is applied to a table.**

**WRITE lock will allow Both READ and WRITE operations on a table. Similarly, when this lock is applied, it will not let other mysql connections to execute write query until the lock is released!!**

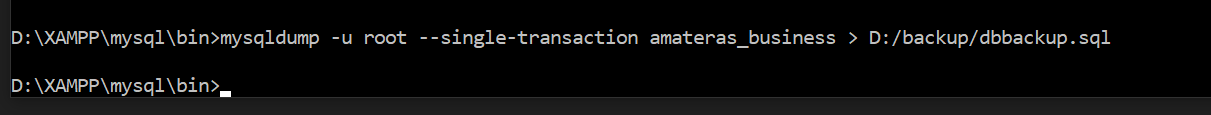
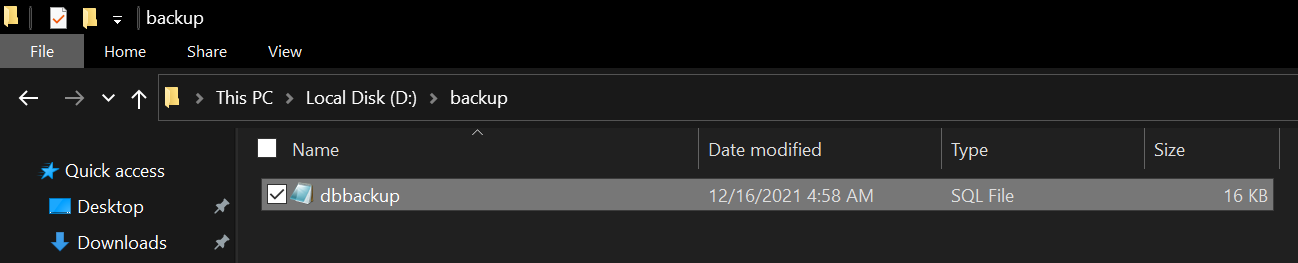
**Remove LOCK (or) UN-Lock table.**

****

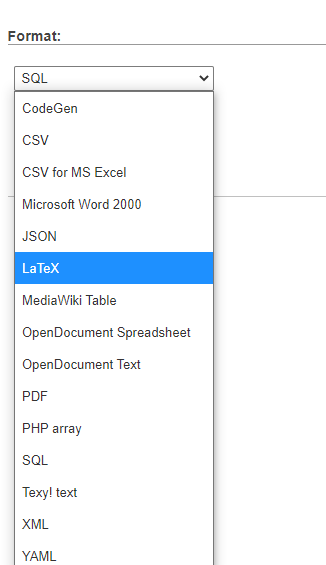
**After removing the locks on table, when we use the command to show processes list, we should see the pending query has been executed. This type of locking mechanism ensures to prevent inconsistencies in a database**

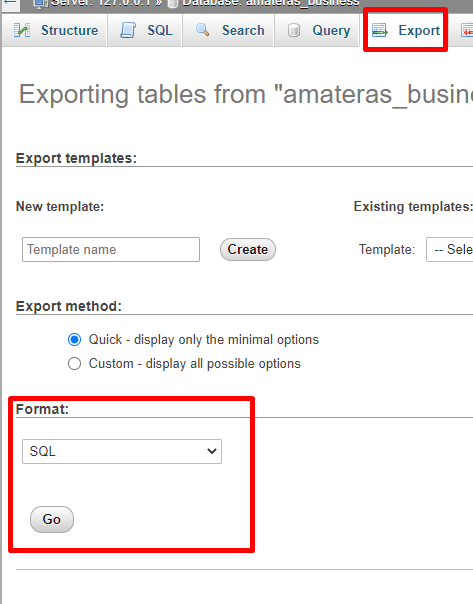
**1.13 Backup**

**Command for backing up database:**

**the command mysqldump is a built-in mysql backup feature, which will dump the selected database in .sql script format.**

**In the above command, I have selected “D:/backup” folder to save my backup. This will save the .sql script backup locally on my machine. Below is the screenshot for a saved backup file generated by mysqldump command.**

**PhpMyAdmin provides Backup option in a more simple way.  
Click export and select the format in which you want the data to be dumped.  
The below are the following types of formats available to dumb the database.**

****

**Automation of backup using Windows task scheduler:**

**First create a script called backup.bat and save it in any location.**

**Paste the following code:**

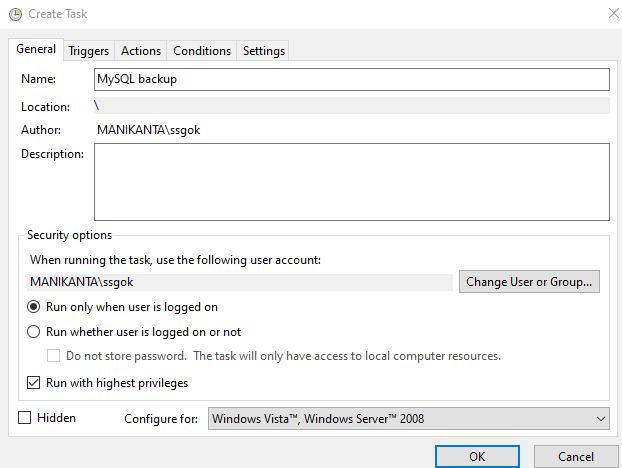
**@echo off**

**set TIMESTAMP=%DATE:~10,4%%DATE:~4,2%%DATE:~7,2%**

**"D:\xampp\mysql\bin\mysqldump.exe" -uroot -hlocalhost amateras\_business> D:\backup\test.%TIMESTAMP%.sql**

**Pause**

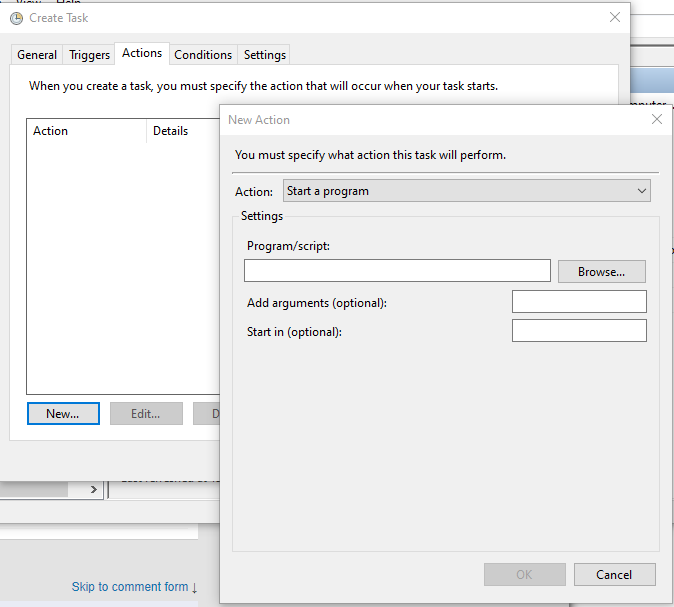
**Open windows task scheduler by going to run and type taskschd.msc**

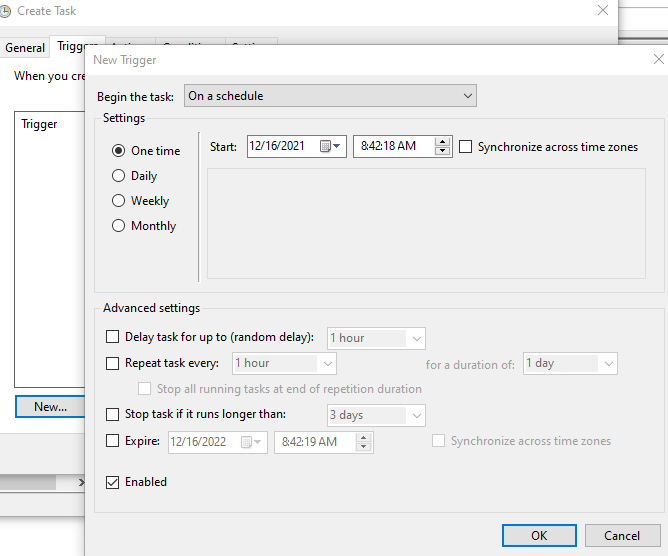
****

**Create a task with any name**

**Click actions and select new**

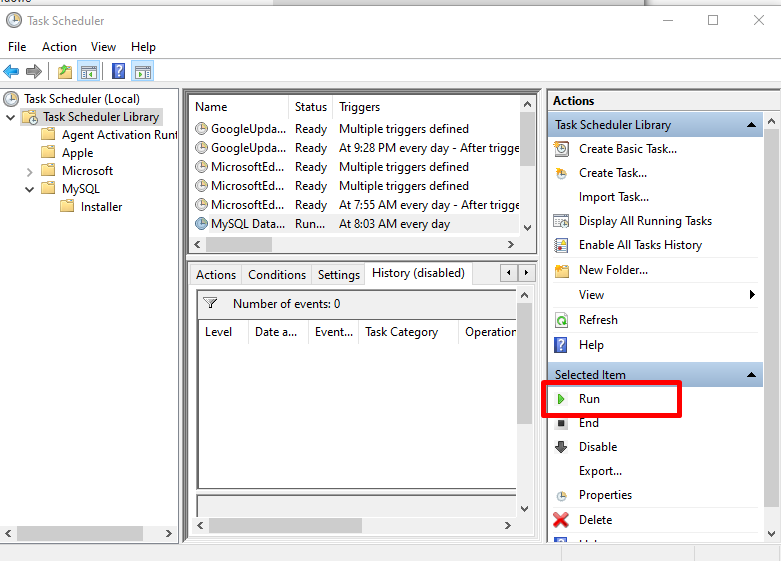
**Browse the script location and click ok.  
Click Trigger to add a time and select the frequency of backup [ one time | daily | weekly | monthly ]**

****

****

**Click OK to add frequency**

**If you want to execute the task in any given time or manually, then click run and it should store the backup file in the path which is mentioned in the script.**

****

**Database backup encryption and decryption using openssl**

**MySQL file encryption & decryption command line:**

**encryption:**

**openssl enc -aes-256-cbc -in /path/example.sql -out /path/example.file**

**decryption:**

**openssl enc -aes-256-cbc -d -in /path/example.file > /path/example.sql**

**1.14 Python programming**

**import mysql.connector**

**import csv**

**#### Connect to mysql database**

**#### requires host, username,password and the target database**

**mydb = mysql.connector.connect(**

**host="localhost",**

**user="root",**

**password="",**

**database="amateras\_business"**

**)**

**## Menu driven program to print data from database**

**print("Welcome to amateras\_business database")**

**print("1. Show product\_information table\n2. Show laptop Table")**

**print("3. Show laptop\_data table\n4. Show employee\_data table")**

**print("5. Show employee\_billing\_data table\n6. Show sales\_data table")**

**print("7. Exit")**

**ch = int(input("Enter your choice: ")) ## stores user input as integer**

**print("\n")**

**## table names which are present in my database**

**## i stored them in array format to write only one block of code**

**## this will help in using only one block of code for multiple tables**

**tables = ['', 'product\_information', 'laptops', 'laptop\_data', 'employee\_data', 'employee\_billing\_details', 'sales\_data']**

**table\_name = tables[ch]; #This variable will hold the value of the table name which the user has selected**

**print("Showing table: "+table\_name)**

**print("-"\*50)**

**mycursor = mydb.cursor() ## connect to cursor object to execute sql statements**

**while(ch in range(1,7)):**

**res = mycursor.execute("SELECT \* FROM "+tables[ch]) ## writing only one SQL query for multiple tables by using array**

**myresult = mycursor.fetchall()**

**for x in myresult:**

**print(x) ## prints result to screen**

**print("-"\*50)**

**print("\n")**

**file = open(table\_name+'.txt', 'w') ## open the file to write data**

**with file as f:**

**f.writelines([(str(i)+'\n') for i in myresult]) ## this will write all the mysql table data to file**

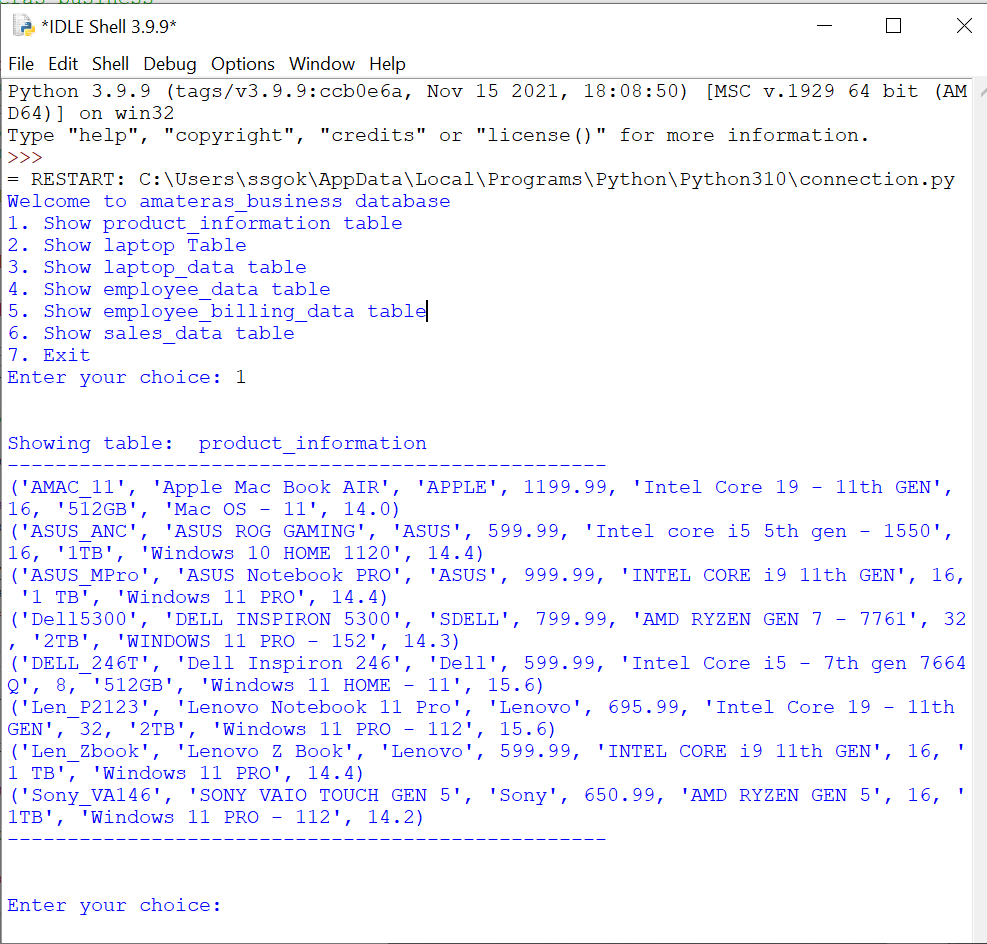
**ch = int(input("Enter your choice: ")) ##writing this here will help to generate the menu again using while condition**

**if(ch > 6):**

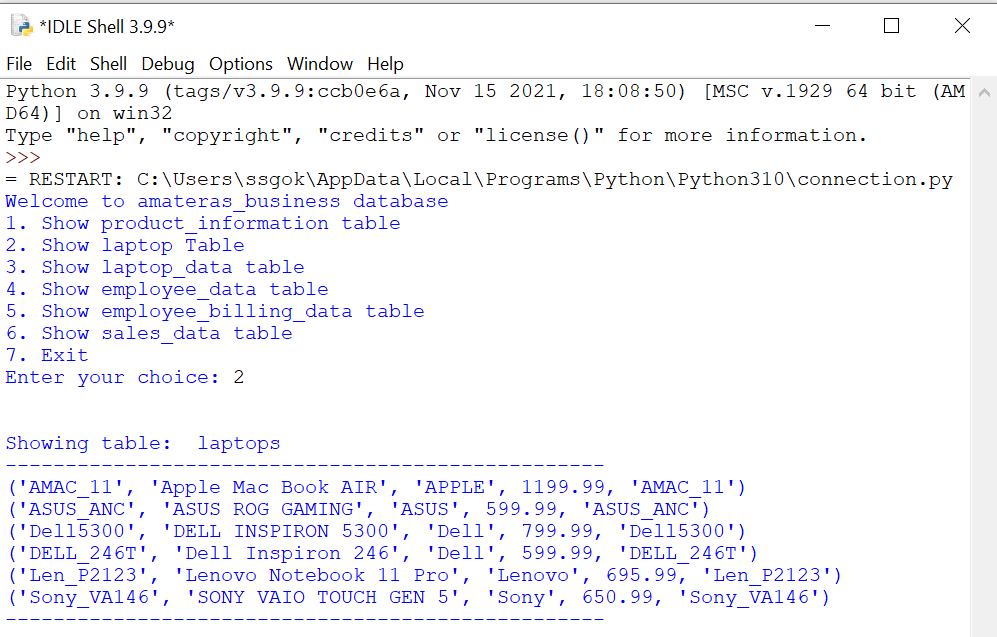
**print("Exiting the program.. have a good day");**

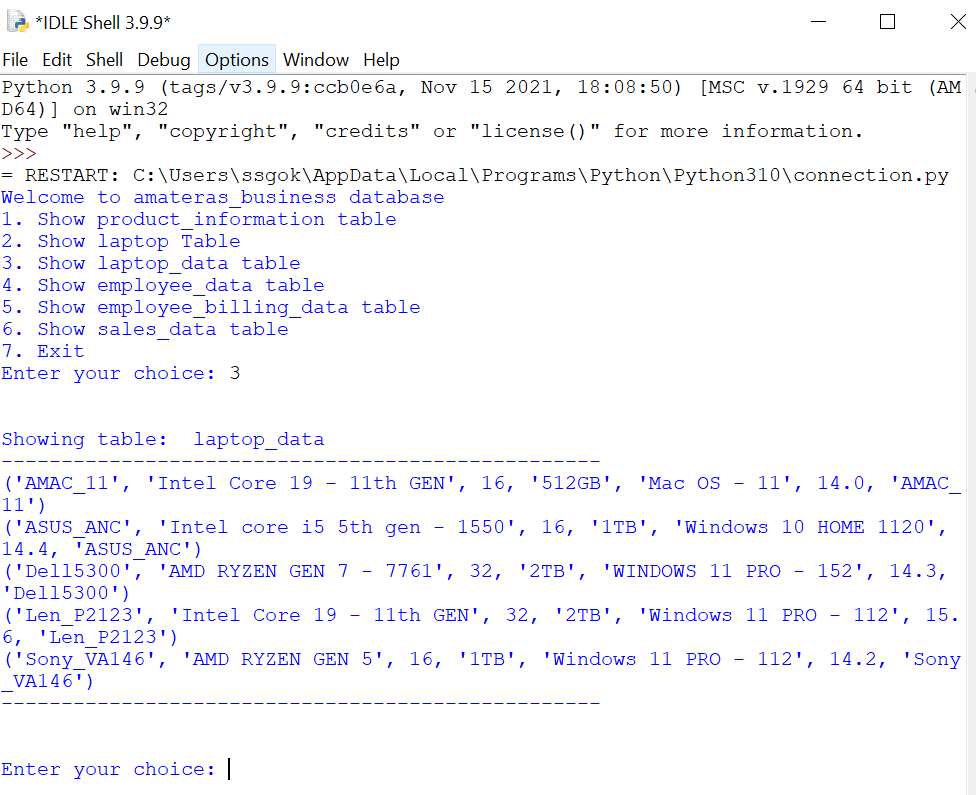
**file.close() ## close the file pointer  
mydb.close() ## close mysql connection**

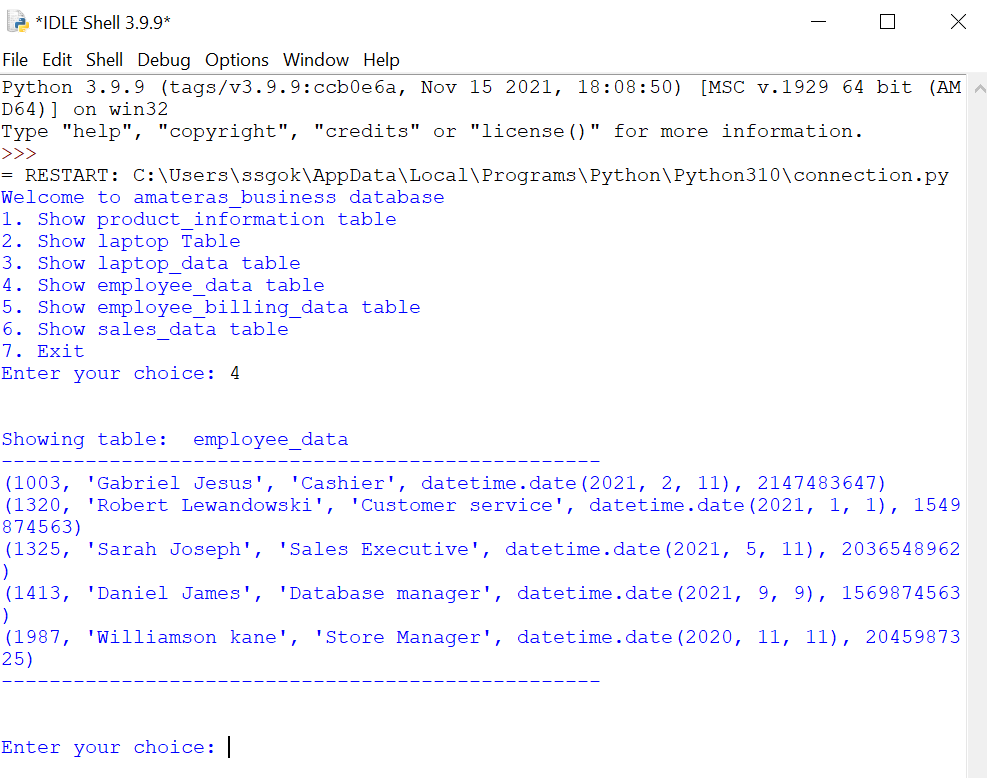
**Output screenshots:**

**For table “product\_information”**

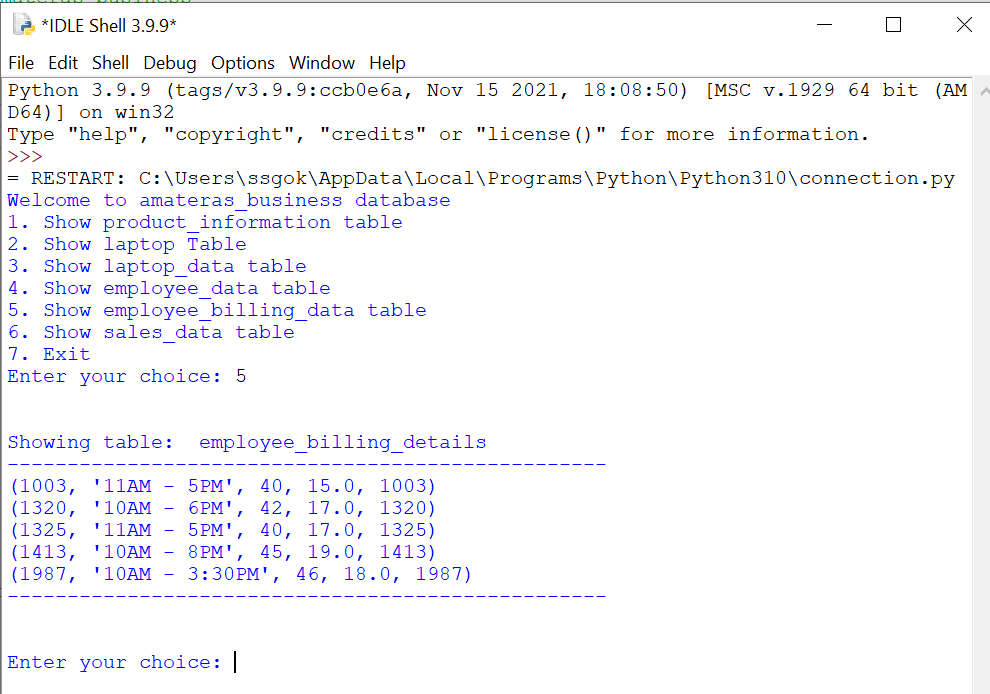
**For table laptops:**

****

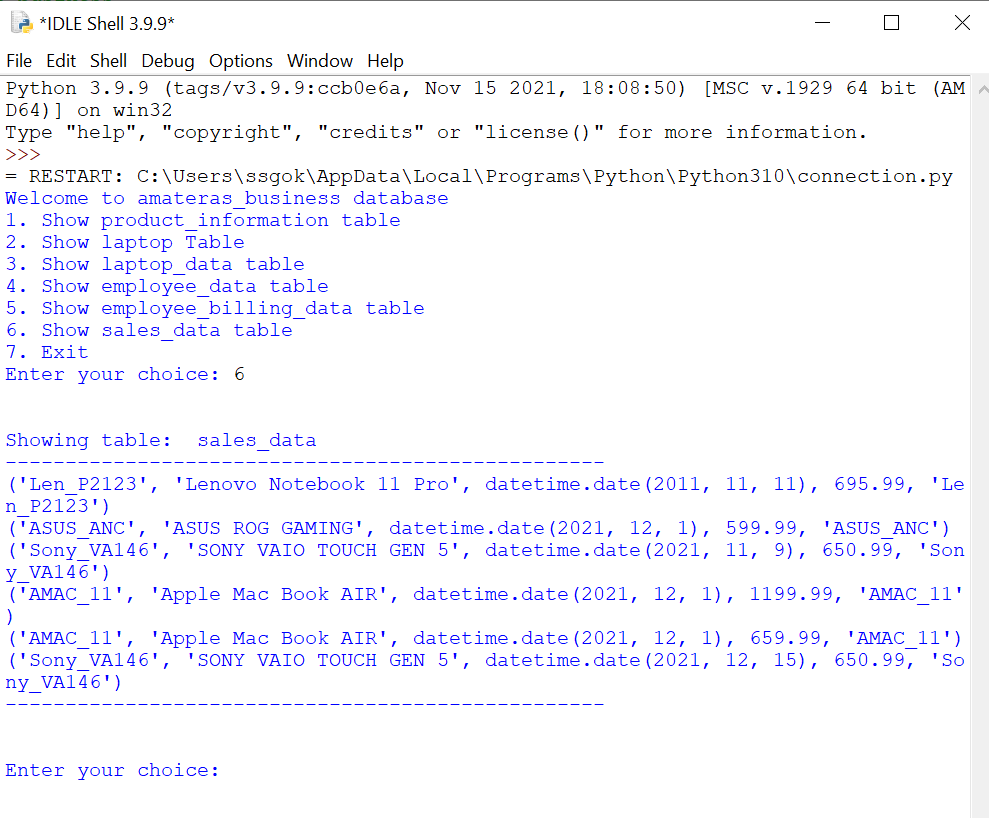
**For table laptop\_data**

**For employee\_data table:**

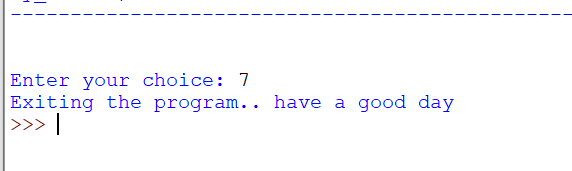
**For employee\_billing\_data table:**

****

**For sales\_data table:**

****

**Exit condition:**

****

**This python program works by connecting to amateras\_business database, the name which I have given for my business database.**

**I will be logging in as the user root on localhost server.**

**This program works by displaying options [ 1 – 7 ]. Each option will result in displaying a table associated with that number while #7 works as exit from program.**

**Bonus Work:**

**PHP SCRIPT**

**PHP has an advantage when it comes to writing small scale scripts.**

**In a .PHP file, we can write small blocks of CSS, HTML and also javascript code**

**Since my database search project is a very small scale code, I have written all piece of code in one single file .PHP  
So according to rubric, the task of creating a .html file for HTML form, PHP script and output .html file are all stored in one single .PHP file**

**There’s a software called “XAMPP” which can run PHP, Apache and MySQL.**

**Download link: https://www.apachefriends.org/download.html**

**The software can be downloaded from above URL.  
Tutorial for XAMPP installation:** [**https://www.javatpoint.com/installation-process-of-xampp**](https://www.javatpoint.com/installation-process-of-xampp)

**Paste the code in a file and name the extension as “.PHP”  
example: myscript.php  
For windows:  
place this script file in the folder: XAMPP/htdocs/  
I’m not sure about linux environment but im pretty sure we will find the same path after installing XAMPP on linux machine.**

**Open XAMPP control panel and start Apache and mysql services.  
Open web browser and type “localhost/[folder\_name]/file\_name.php”**

**Here, the [folder\_name] is optional if you have placed the .php file inside htdocs folder**

**Let’s say the path to the php script is :  
XAMPP/htdocs/DBMS/myscript.php  
Then the browser url should be: localhost/DBMS/myscript.php**

**If the path to php script is  
XAMPP/htdocs/myscript.php  
Then the browser url should be: localhost/myscript.php**

**Note: The .php script file should be placed in /htdocs folder or any sub folder of /htdocs like I have mentioned in my above example.  
or else the php code will not work.**

**CODE:**

**<?php**

**$result = '';**

**$q='';**

**?>**

**<html>**

**<head>**

**<script type="text/javascript">**

**function update() {**

**var table = document.getElementById('table').value;**

**if(table == 'product\_information')**

**data = '<br><h3>Search criteria for '+table+' is name of the Laptop. Ex: Lenovo (or) Asus</h3>';**

**else**

**data = '<br><h3>Search criteria for '+table+' is employee name</h3>';**

**document.getElementById("data").innerHTML=data;**

**}**

**</script>**

**<title>Amateras Business Database</title>**

**<style>**

**#mytable {**

**font-family: Arial, Helvetica, sans-serif;**

**border-collapse: collapse;**

**width: 100%;**

**}**

**#mytable td, #mytable th {**

**border: 1px solid #ddd;**

**padding: 8px;**

**}**

**#mytable tr:nth-child(even){background-color: #f2f2f2;}**

**#mytable tr:hover {background-color: #ddd;}**

**#mytable th {**

**padding-top: 12px;**

**padding-bottom: 12px;**

**text-align: left;**

**background-color: #04AA6D;**

**color: white;**

**}**

**</style>**

**</head>**

**<body>**

**<h1><center>Amateras Business Database</center></h1>**

**<hr>**

**<br>**

**<form action = "dbms.php" method="POST">**

**Select a table to perform search operation:**

**<select id="table" onclick="update()"name='table\_'>**

**<option value=''>Select</option>**

**<option value='product\_information'>Product Information</option>**

**<option value='employee\_data'>Employee Data</option>**

**</select>**

**<div id="data"><br></div>**

**<br><br>**

**Enter Search string&nbsp;<input type="text" size="40" name="query" value="<?=$q;?>">&nbsp;<input type="submit" value="Search" name="search">**

**<br><br><?=$result;?>**

**</form>**

**</body>**

**</html>**

**<?php**

**if(isset($\_POST['search'])) {**

**$search = $\_POST['query'];**

**$q = $search;**

**$table = $\_POST['table\_'];**

**$conn = mysqli\_connect("localhost", "root", "", "amateras\_business");**

**$field = ($table == "product\_information") ? "model\_name" : "emp\_name";**

**$SQL = "SELECT \* FROM $table WHERE $field LIKE '%$search%'";**

**$res = mysqli\_query($conn, $SQL);**

**$fields = array(**

**'product\_information' => array('model\_id', 'model\_name', 'model\_company', 'model\_cost', 'model\_processor','model\_ram\_size', 'model\_hdd\_size', 'model\_operating\_system', 'model\_screen\_size'),**

**'employee\_data' => array('emp\_id', 'emp\_name', 'emp\_role', 'emp\_joining\_date', 'emp\_phno')**

**);**

**?>**

**<Table id="mytable">**

**<tr>**

**<?php**

**foreach($fields[$table] as $key=>$name) {**

**?>**

**<th><?=$name;?></th>**

**<?php**

**}?>**

**</tr>**

**<?php**

**while($row = mysqli\_fetch\_array($res)) {**

**//foreach($row as $key =>$value)**

**?><tr>**

**<?php**

**foreach($fields[$table] as $key=>$name) {**

**?><td><?=$row[$name]?></td><?php**

**}**

**?>**

**</tr>**

**<?php**

**}**

**}**

**?>**

**Working of code including output screenshots:  
This script will search two tables in my database**

1. **Product\_information**
2. **Employee\_data**

**The search criteria for the table “product\_information” is “model\_name” row in the table..**

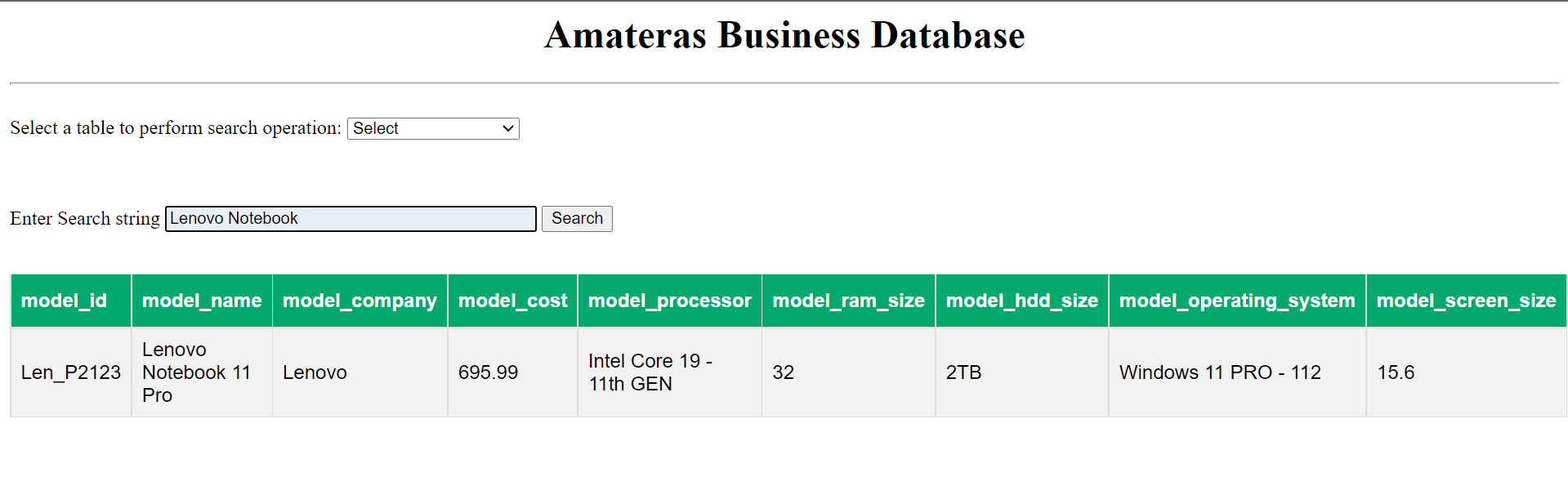
**Which means to get results, we need to search for a specific laptop name in the search box.**

**The search criteria for table “employee\_data” is emp\_name.**

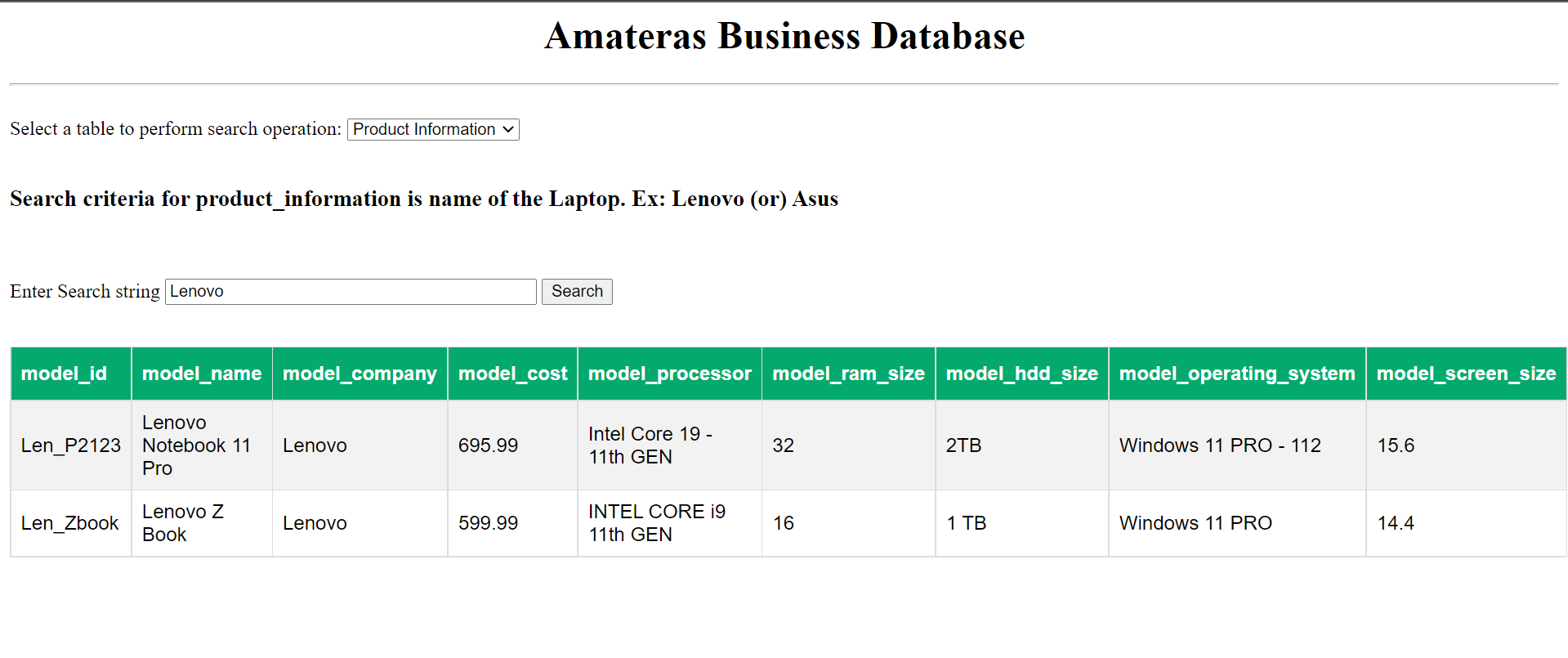
**To get results, we need to search for an employee using his name in the search box.**

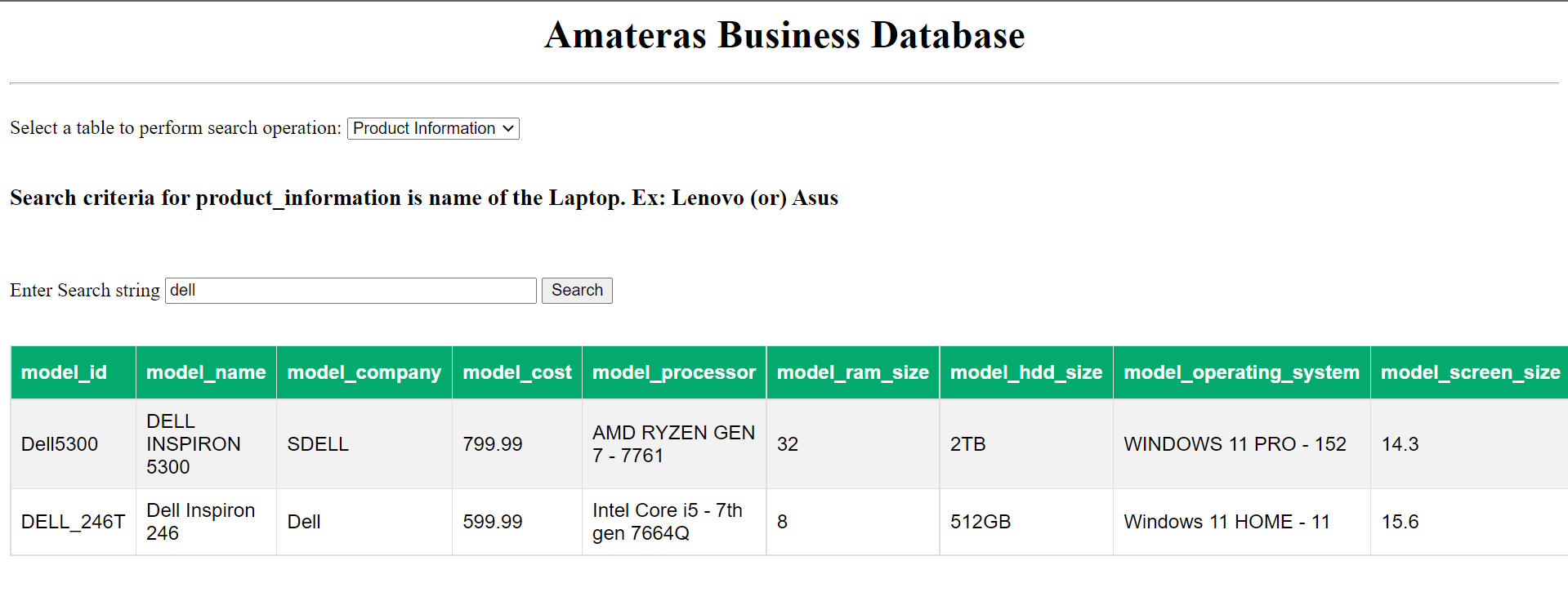
**Output screenshots:**

**#1 Input: table = product\_information, query = Lenovo Notebook**

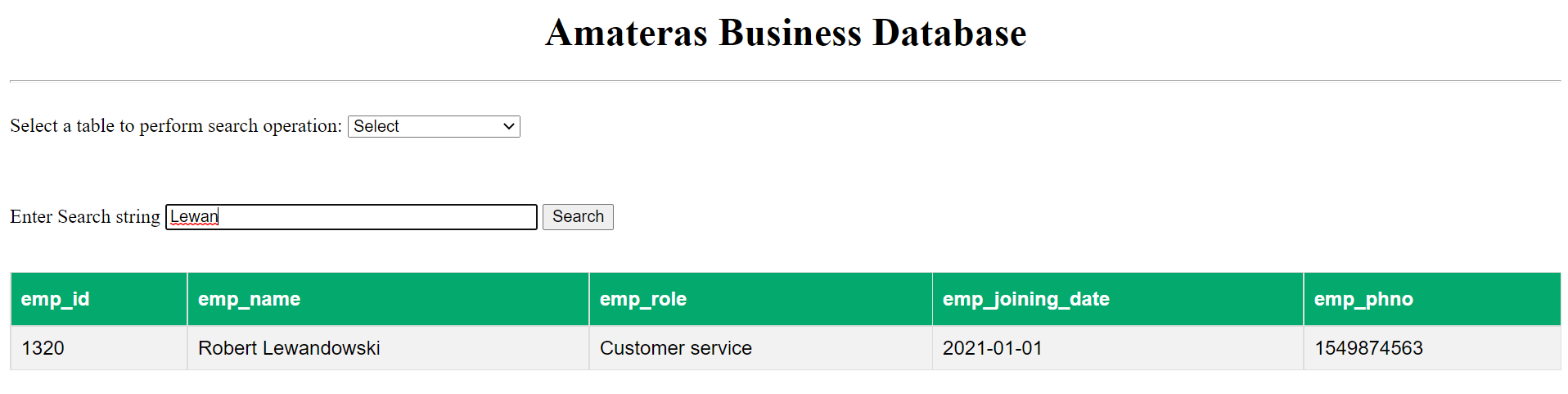
****

**#2 input Table: product\_information, query = Lenovo**

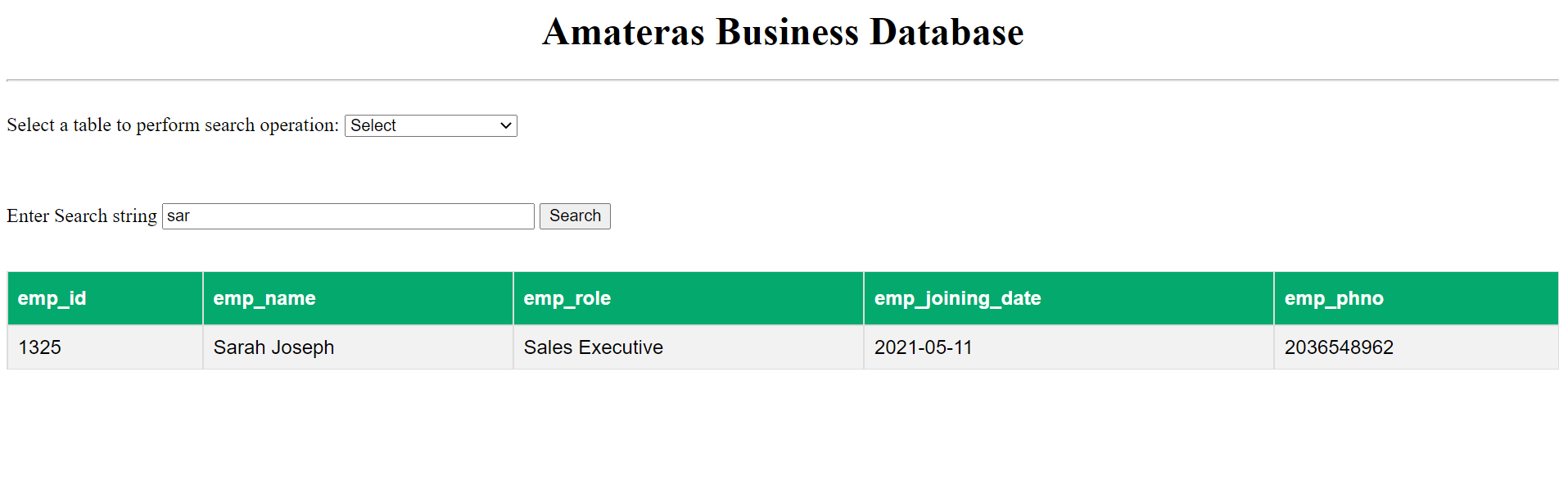
****

****

**#4 input table = Employee Details, query = lewan**

****

**#5 input table = employee details, query = sar**

****

**SQL Injection attack**

**A**[**SQL injection**](https://owasp.org/www-community/attacks/SQL_Injection)**attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.**

**INJECTION PREVENTION IN MY SCRIPT:  
In my script, I have made two php variables to perform the search criteria**

**My SQL:**

**$SQL = "SELECT \* FROM $table WHERE $field LIKE '%$search%'";**

**Here, the $table and $field are variables which will get values from the website. Here, the attacker cannot modify them because these values have to be given from a selection based HTML form..**

**Thus, the attacker cannot input his code or modify the values.**

**This type of protection is called FORM-VALIDATION in MYSQL.**

**Suggested future work:**

**My present database limitations would be proper encryption of database, constant monitoring of database using a database admin physically.**

**I would overcome these issues by moving my database to a cloud provider such as amazon AWS.**

**They provide better data encryption, Tools and a virtual database admin to keep and manage my data securely.**

**Amazon is known world wide for their various services like e-commerce and IT development.**

**Moving my database from local to cloud:  
Amazon AWS provides tools to move our database from our server to Amazon hosted web servers.**

**https://aws.amazon.com/getting-started/hands-on/move-to-managed/migrate-my-sql-to-amazon-rds/**

**Moving to cloud has a lot of benefits**

**Advantages of using a Cloud database include:**

**- Fast automated recovery from failures**

**- Either built-in to a larger package with nothing to configure, or comes with a straightforward GUI-based configuration**

**- Managed backups, restores, and automated scheduling**

**- Does not require any operating system knowledge**

**- Automated on-the-go scaling with the ability to simply define the scaling rules or manually adjust**

**- Potentially lower cost**

**- Device independence**

**- Better performance**

**- Scalability**

**- Automatic failover/high availability**

**- Widely accessible**

**-Minimal outlay investment; you can always buy your own server later.**

**- You can provision a few servers in the cloud immediately, don't have to wait a few days for shipping, don't have to spend time setting up the environment**

**- Excellent choice for those with space considerations**

**- May be able to pay little or nothing for unused time**

**Some of the disadvantages include:**

**Security and privacy issues**

**Requires a constant Internet connection (bandwidth costs!)**

**Loss of control over resources: e.g., can you delete the data?**

**Data ownership: Who owns the data? Where is it? Who will accept the risk for compromised data?**

**Data is tied to the provider**

**Requires staff that has specialized skill set**

**No proven track record yet**

**Advantages of moving my data to NoSQL database**

**NoSQL databases offer many benefits over relational databases. NoSQL databases have flexible data models, scale horizontally, have incredibly fast queries, and are easy for developers to work with.**

**Flexible data models**

**NoSQL databases typically have very flexible schemas. A flexible schema allows you to easily make changes to your database as requirements change. You can iterate quickly and continuously integrate new application features to provide value to your users faster.**

**Horizontal scaling**

**Most SQL databases require you to scale-up vertically (migrate to a larger, more expensive server) when you exceed the capacity requirements of your current server. Conversely, most NoSQL databases allow you to scale-out horizontally, meaning you can add cheaper, commodity servers whenever you need to.**

**Fast queries**

**Queries in NoSQL databases can be faster than SQL databases. Why? Data in SQL databases is typically normalized, so queries for a single object or entity require you to join data from multiple tables. As your tables grow in size, the joins can become expensive. However, data in NoSQL databases is typically stored in a way that is optimized for queries. The rule of thumb when you use MongoDB is Data is that is accessed together should be stored together. Queries typically do not require joins, so the queries are very fast.**

**Easy for developers**

**Some NoSQL databases like MongoDB map their data structures to those of popular programming languages. This mapping allows developers to store their data in the same way that they use it in their application code. While it may seem like a trivial advantage, this mapping can allow developers to write less code, leading to faster development time and fewer bugs.**

**References:**

1. [**https://www.w3schools.com/sql/default.asp**](https://www.w3schools.com/sql/default.asp)
2. [**https://dev.mysql.com/doc/refman/8.0/en/**](https://dev.mysql.com/doc/refman/8.0/en/)
3. [**https://aws.amazon.com/getting-started/hands-on/move-to-managed/migrate-my-sql-to-amazon-rds/**](https://aws.amazon.com/getting-started/hands-on/move-to-managed/migrate-my-sql-to-amazon-rds/)
4. [**https://www.databasejournal.com/features/mssql/should-you-move-your-mysql-database-to-the-cloud.html**](https://www.databasejournal.com/features/mssql/should-you-move-your-mysql-database-to-the-cloud.html)
5. [**https://www.phptutorial.net/php-tutorial/php-form/**](https://www.phptutorial.net/php-tutorial/php-form/)
6. [**https://www.tutorialspoint.com/python/python\_database\_access.htm**](https://www.tutorialspoint.com/python/python_database_access.htm)
7. [**https://dev.mysql.com/doc/refman/5.7/en/trigger-syntax.html**](https://dev.mysql.com/doc/refman/5.7/en/trigger-syntax.html)
8. [**https://dev.mysql.com/doc/refman/8.0/en/views.html**](https://dev.mysql.com/doc/refman/8.0/en/views.html)