

# **LAPTOP ORDERING SYSTEM**

TEAM Members : Aswath Senthil Kumar and Amrit Gupta

Group Name : GuptaASenthil KumarA

PROFESSOR : Kathleen Durant

**Database Management Project**  
**Fall 2022**

# README

Our project has been built using python and MySQL. The dependencies for our project can be installed using the command “pip install -r requirements.txt”. Please note our project also requires MySQL Server to be running and our database dump should be imported from our self-contained dump file(db\_dump.sql).

Our project can be run by executing the command “python project.py”. The program will prompt the user for the database username and password.

On successful connection to the database the program will display options to select either “customer options” or “employee options”. Upon selecting an option the program will display the appropriate menu with options to either login to an existing account or create a new account.

A new customer account can be created by following the prompts. To create a new employee account, an auth code ("busybee") needs to be entered.

Post account creation the customer or employee will be able to login and perform their respective operations.

Operations provided to a customer are -

1. Edit profile
  - a. Update mobile number
  - b. Update password
2. Delete account
3. Place an order for a laptop
  - a. Filter laptops based on RAM, SSD, Screen Size, and Operating System
4. Check order status
5. Cancel an existing order
6. Check payment status

Operations provided to an employee are -

1. Edit Profile
  - a. Update mobile number
  - b. Update password
2. Approve a customer payment
3. Visualize laptop model sales
4. Visualize brand sales

Project video link (please download the video and play for better resolution) - [https://drive.google.com/file/d/1-kgANPB4j22VMD3ccw5BSbJ-MVmioMBj/view?usp=share\\_link](https://drive.google.com/file/d/1-kgANPB4j22VMD3ccw5BSbJ-MVmioMBj/view?usp=share_link)

# **Technical Specifications**

## **Software Requirements**

- Python 3
- MySQL Server
- MySQL Workbench
- tabulate
- pymysql
- seaborn
- matplotlib
- numpy
- pandas

## **Hardware Requirements**

- RAM – 4 GB
- Storage – 10 GB
- CPU – 2

## Procedures

The below list gives the list of Procedures we used in our database application.

1. **createCustomer** - Procedure to create a new customer which will also insert a record into the address table for the associated customer.
2. **createEmployee** - Procedure to create a new employee
3. **updateCustomerPhone** - Procedure to update the customer Phone number
4. **updateCustomerPassword** - Procedure to update the customer Password
5. **updateEmployeePhone** - Procedure to update the employee Phone number
6. **updateEmployeePassword** - Procedure to update the customer Password
7. **insertOrder** - Procedure to insert into the order table which will also insert into the places table for the specified order.
8. **cancelOrder** - Procedure to cancel an order which will check all the necessary conditions and then cancel the order.
9. **readLaptopModels** - Procedure to list all the laptop models for a user
10. **deleteCustomer** - Procedure to delete a customer. This procedure holds the logic and takes care of deletion from all the other tables as well wherever the customer is involved.
11. **listOrdersCustomer** - Procedure to list the orders for a customer
12. **insertPayment** - Procedure to insert the Payment record into the Payment table for a specified order.
13. **updatePaymentStatus** - Procedure to update the Payment status for a customer by an employee. This will also update the order table
14. **getListCustomers** - Procedure to get the list of customers associated to an employee
15. **getOrdersListCust** - Procedure to get the list of orders for a customer.
16. **returnLaptopModelCount** - Procedure to get the laptop models and the total number of orders for that model.
17. **returnBrandCount** - Procedure to return the brand and the total number of orders for that brand.
18. **returnCustomerEmails** - Procedure to return all customer email IDs
19. **checkCustomerPassword** - Procedure to verify a customer Password
20. **returnEmployeeEmails** - Procedure to return all employee email IDs
21. **checkEmployeePassword** - Procedure to verify an employee Password

## Functions

The below list gives the list of Functions we used in our database application.

- 1.) **returnOrderStatus** - Function to return the order status for a laptop order
- 2.) **returnPaymentStatus** - Function to return the Payment status for a laptop order
- 3.) **getCustomerID** - Function to get the customerID for the given email address.
- 4.) **getEmployeeID** - Function to get the employeeID for the given email address.

## Triggers

The below list gives the list of Triggers we used in our database application.

- 1.) **deletePlaces** - Trigger to delete an entry from the places table after the associated entry is deleted from the Payment table.
- 2.) **deleteOrder** - Trigger to delete an entry from the order table after the associated entry is deleted from the places table.
- 3.) **deleteCustomerAddress** - Trigger to delete an entry from the customer table after the associated entry is deleted from the address table.
- 4.) **addLapBrand** - Trigger to update the numOfProducts variable whenever a laptop order is placed from the specified brand.
- 5.) **addLapModel** - Trigger to update the numOfLaptops variable whenever a laptop order is placed from the specified laptop model.

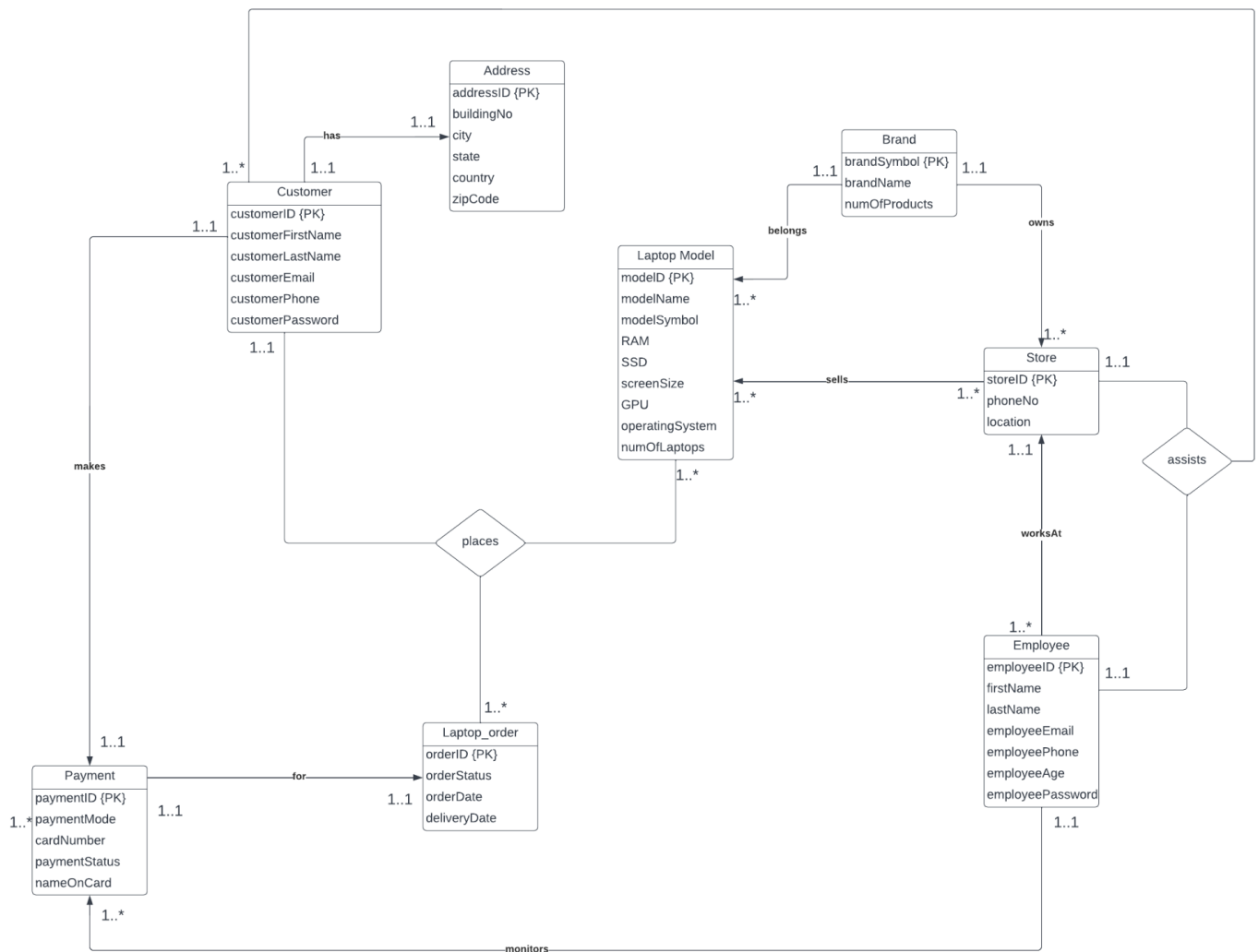
We have also used **Transactions** in the following procedures:

- 1.) **cancelOrder**
- 2.) **createCustomer**
- 3.) **createEmployee**
- 4.) **insertOrder**
- 5.) **insertPayment**

We have used **Exit Handlers** for exception handling in the following Procedures:

- 1.) createCustomer
- 2.) createEmployee
- 3.) cancelOrder
- 4.) insertOrder
- 5.) insertPayment

This is the final UML representation for our Application

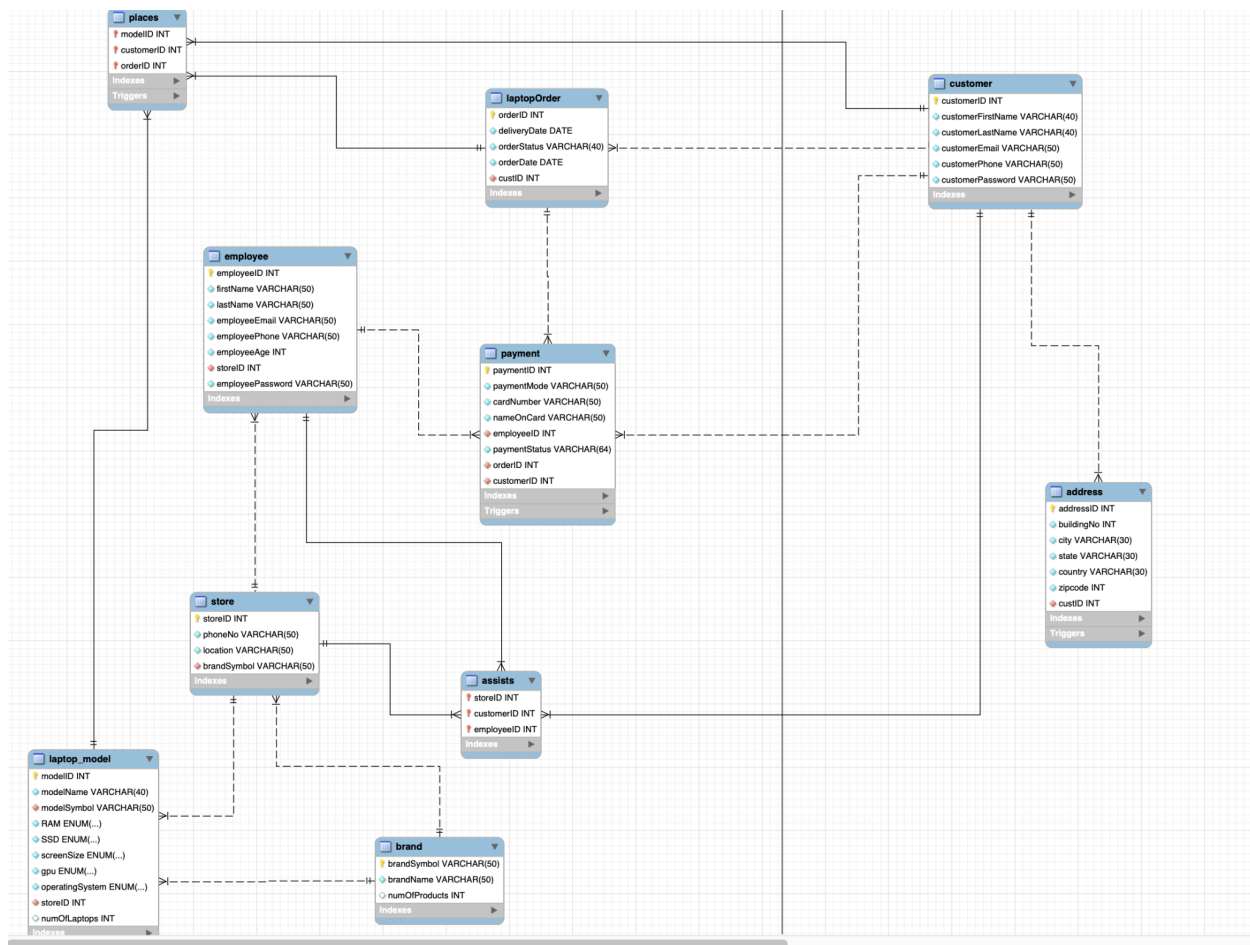


We removed a few redundant relationships from our initial design and created our final design such that it denotes only the necessary relationships.

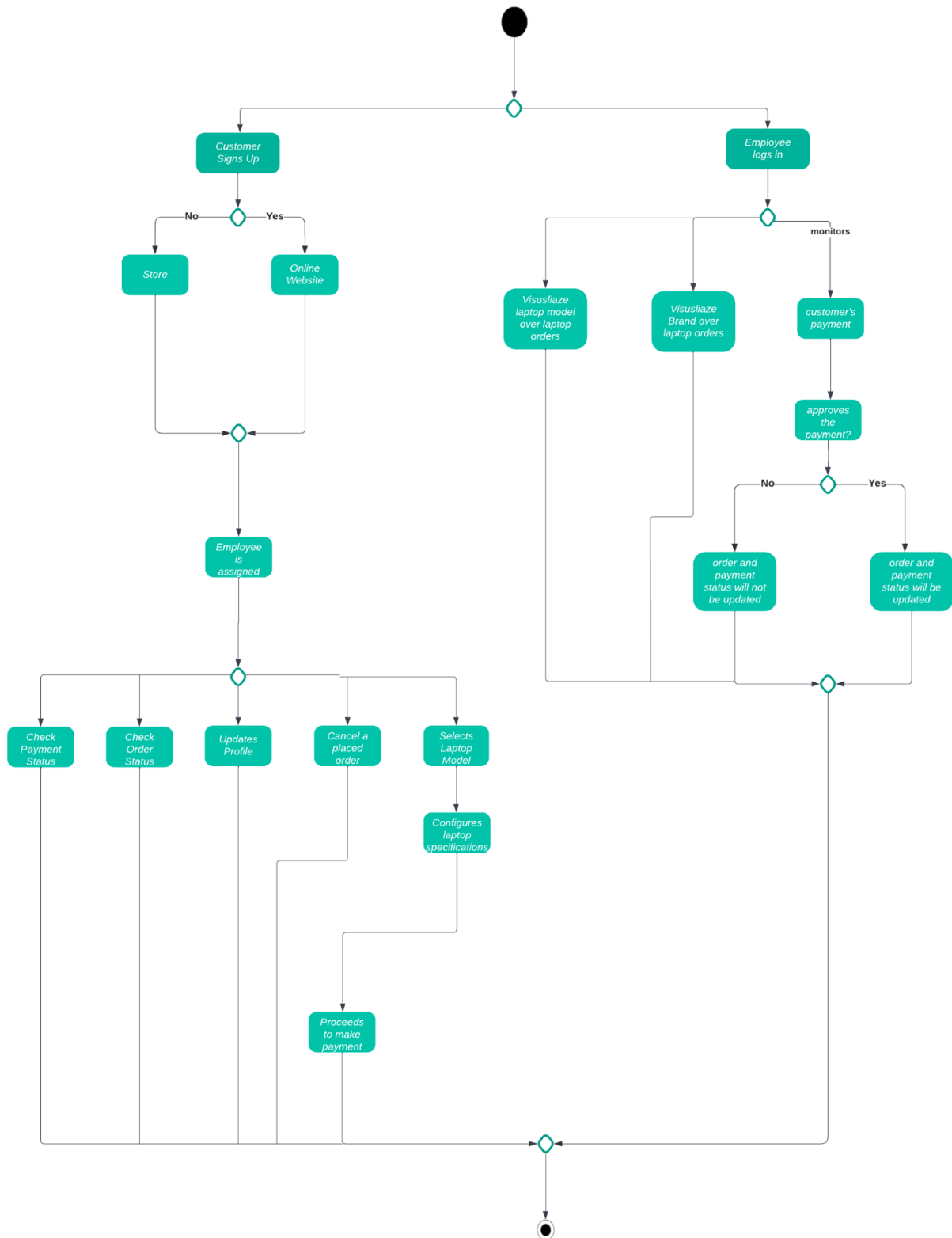
We had initially included a Supplier entity which we were not able to add to our design because of limited scope for the entity and the entity seemed redundant with the Brand entity so we removed it in our final design.

We also had receipt and review entities that we initially had as a part of our design. We had these entities mainly for visualizations. But since we had enough parameters to have good visualizations, we continued visualization with our existing parameters.

## Logical Design



The Flow Chart of our Database System is below:

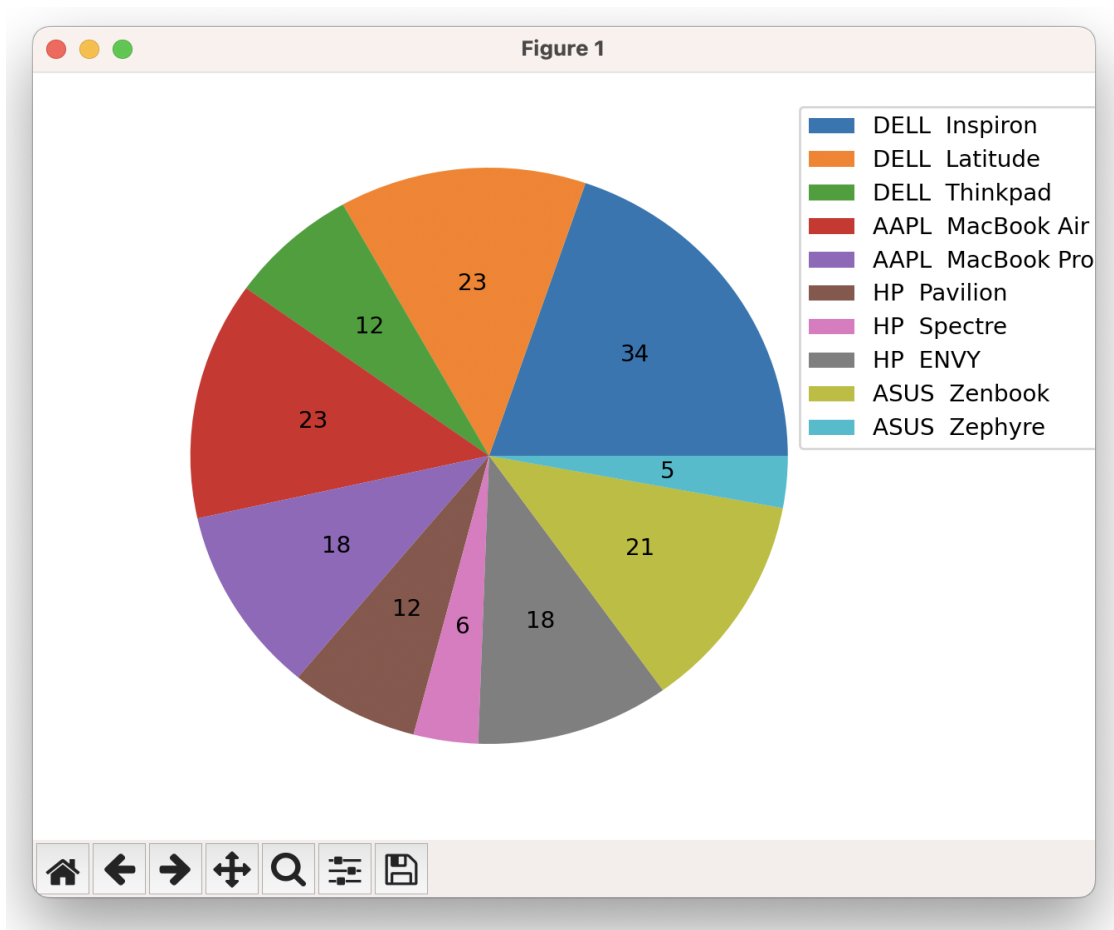




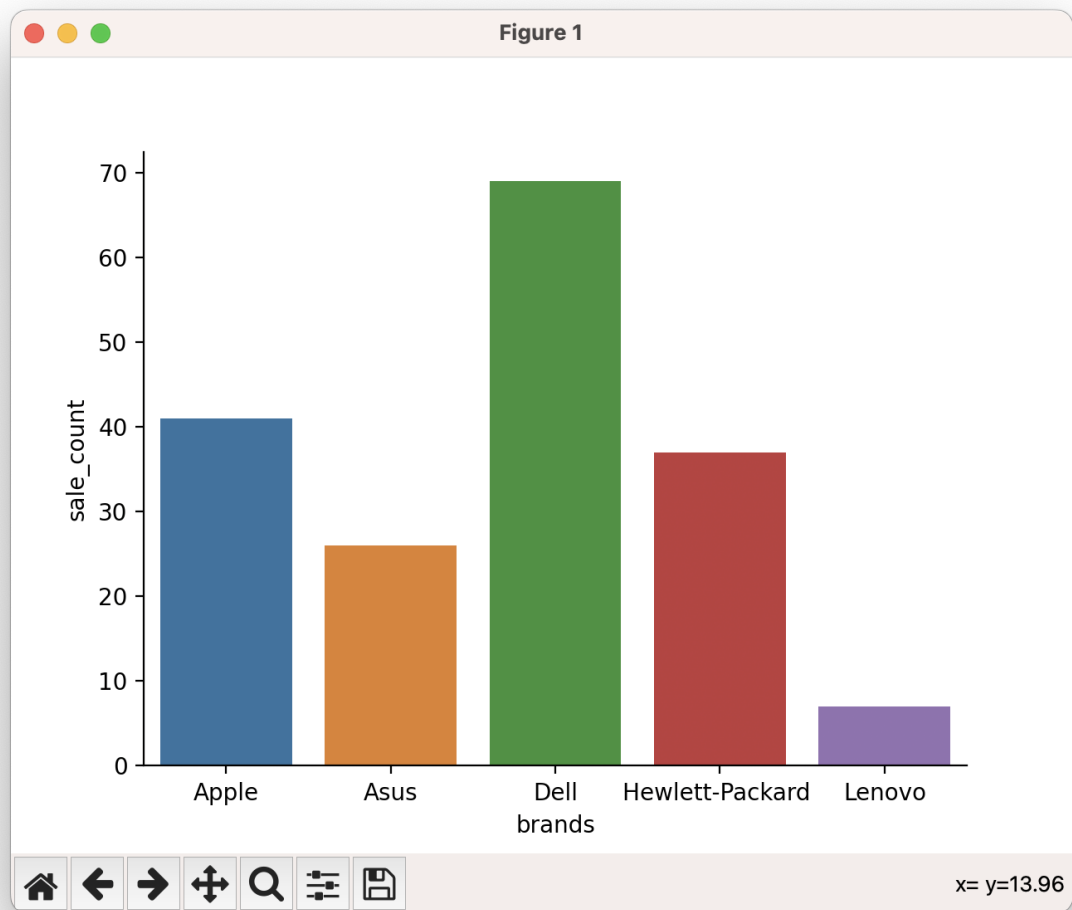
- 1) The Customer checks our online website store to see the list of laptop models currently available based on his filter criteria
- 2) The customer will configure the specifications for the laptop based on his/her requirements.
- 3) Now the customer will select a laptop model, update his profile, cancel an order, check order status, check payment status.
- 4) The customer will proceed to place the order once they are done with selecting the model and configuring the specifications
- 5) If the customer confirms the order, they need to make a payment.

## Screenshots of our program

Visualization of laptop model sales



## Visualization of brand sales



## Customer Menu

```
dbms_project — python project.py — 100x20
Please select one of the below options to continue
Enter 1 for customer options
Enter 2 for employee options
1
Enter 1 to login to existing customer account
Enter 2 to create new customer account
1

Customer Login Selected
Enter your email: sam@gmail.com
Enter your password:
Login Successful

Enter 1 to edit your profile
Enter 2 to delete your account
Enter 3 to place an order for a laptop
Enter 4 to check order status
Enter 5 to cancel an order
Enter 6 to check payment status
```

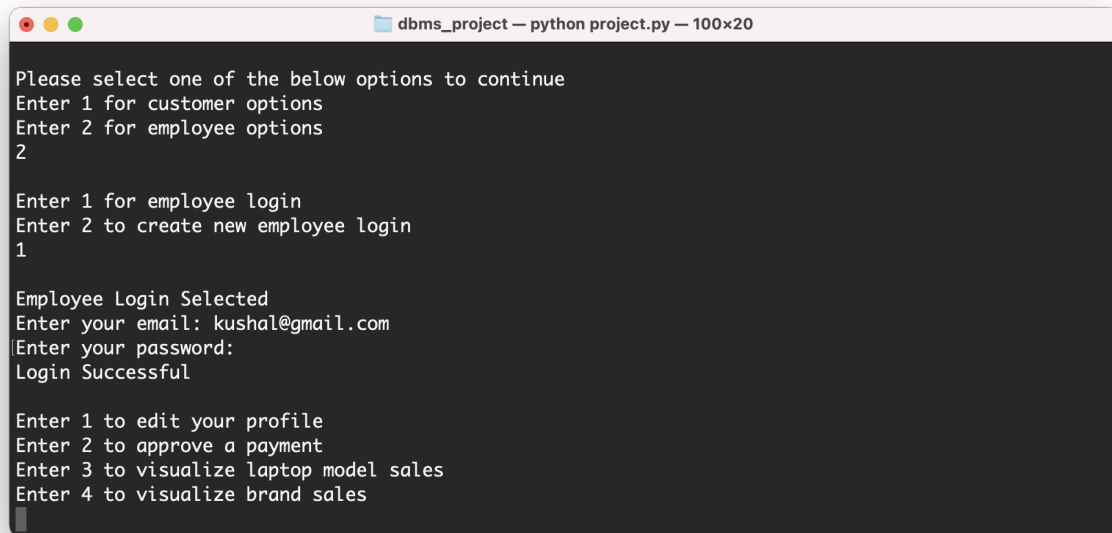
## Available Laptops

```
dbms_project — python project.py — 100x20
Available screen size options are 13, 14, 16
Enter the screen size (Leave blank for skip):
Available operating system options are Windows, macOS
Enter the operating system (Leave blank for skip):

LIST OF AVAILABLE LAPTOPS:
MODEL ID  MODEL NAME  BRAND  RAM  SSD  SCREEN SIZE  GPU  OS
-----
1  Inspiron    DELL    8    512    14    8    Windows
2  Latitude    DELL    16   1024   16    8    Windows
3  Thinkpad    DELL    16    512    13    8    Windows
4  MacBook Air  AAPL    32   1024   14    8    macOS
5  MacBook Pro  AAPL    16    512    16    8    macOS
6  Pavilion    HP       8    512    13    8    Windows
7  Spectre     HP      32   1024   14    8    Windows
8  ENVY        HP      16    512    16    8    Windows
9  Zenbook     ASUS    32   1024   16    8    Windows
10 Zephyre     ASUS    16   1024   16    8    Windows

Enter the model ID of the laptop to be purchased:
```

## Employee Menu



```
dbms_project — python project.py — 100x20

Please select one of the below options to continue
Enter 1 for customer options
Enter 2 for employee options
2

Enter 1 for employee login
Enter 2 to create new employee login
1

Employee Login Selected
Enter your email: kushal@gmail.com
Enter your password:
Login Successful

Enter 1 to edit your profile
Enter 2 to approve a payment
Enter 3 to visualize laptop model sales
Enter 4 to visualize brand sales
```

## Bonus

We have used complex queries in the **insertPayment** - There is a **JOIN of 4 different tables** in order to fetch the employeeID and insert into the Payment table.

We have Cascading delete in the procedure **deleteCustomer**: In this procedure a customer will be associated with many tables like address, laptopOrder, payment, places, assists and also the customer table. Therefore we have used triggers which removes the corresponding customer details in all the above mentioned rows and finally deletes from the customer table.

We have attempted another bonus problem by adding **visualizations** to our projects.

The employee is provided with the feature to visualize data about laptop model sales and laptop brand sales.

We have used a **transaction** inorder to create a new customer. By doing so it will also add an entry to the address table for that particular customer. It will also add an entry into the assists table which denotes which employee will be assisting that customer during his/her purchase of the laptop. We have used another transaction inorder to update two tables at the same time which are our order table and the payment table which is updated by the employee.

## **Future Work**

The database may be used to support web applications of a multi-brand laptop store.

The functionality may be enhanced by adding support to store customer reviews of the laptops.

We may also add support for voucher/discount codes that customers may use.

We can also support additional functionality for employees by adding suppliers and allowing employees interacting with suppliers to place orders for laptop parts and accessories. Another enhancement for employee functionality would be to add more visualizations enabling an employee to gain insight into customer shopping patterns.