Name: Krishna Karthik Reddy Jonnala

NYU ID: kj2056

Course Section Number: CSCI-GA.2433-001

**Final Report**

**Total in points** (100 points total): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

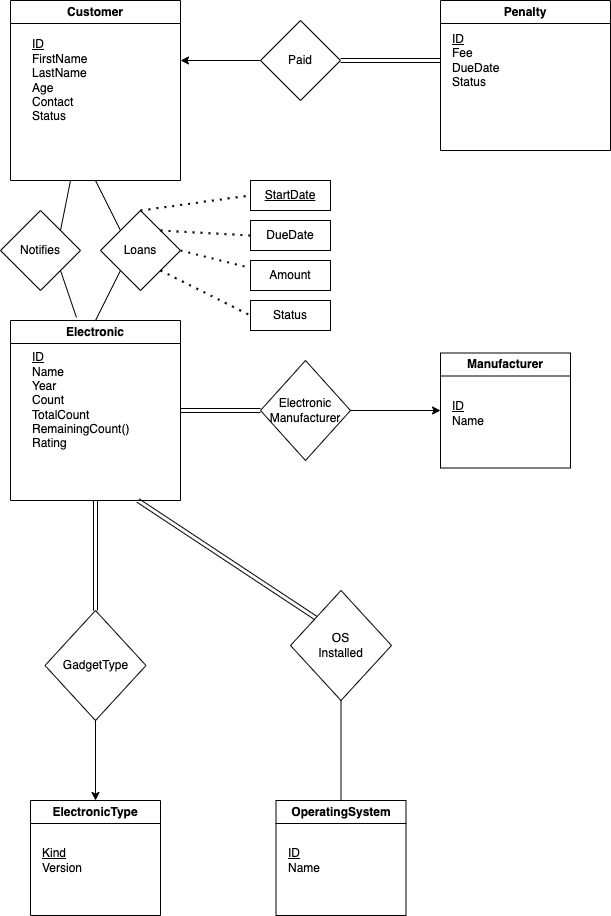
**Professor’s Comments:**

**Business Requirements**

I chose a simpler business instead of an Insurance example as I am not well aware of terms mentioned involving Insurance. I chose a simple Electronics rental business as a case study for the project.

A **Customer** is any individual that has an account stored in the business’s database system. Each **Customer** has a unique ID, First Name, Last Name, Age (indicates account age not customer age, i.e., active since), Contact and Status (Good and Bad) which indicates the trust level of the Customer based on their previous transactions with the rental company. Customers can loan Electronics with software (**OS**) (for this project, let's consider onlyLaptop, Phone and Consoles for rental) and each has **Name** (Ex: MacBook Air M1), **Manufacturer** (Ex: Apple, Dell etc), **Year** in which the item was released, **Count** number of such items, **Cost** to rent the equipment for a month, **Rating**. An Electronic device can have multiple OS (dual boot etc).

The loaner laptops data is stored as **Loan**, which has start and due date of loan, deposit amount, status whether the laptop is returned within due date otherwise the Customer will be Penalized (Penalty). If Customer doesn’t return an item within the due date Customer’s status will be updated to Bad until the Penalty fee is paid. If a particular laptop is unavailable, the Customers can request to be notified (Notify) once the item is available.



**Entity Sets**

* Entity Set **Customer** has a primary key as unique ID with attributes FirstName, LastName, Age (account active since), Contact and Status
* Entity Set **Electronic** has a primary key as a unique ID (Ex: Hardware ID) with attributes Name, release Year, TotalCount, Cost per day, Rating. RemainingCount() is a derived attribute.
* Entity Set **Manufacturer** indicates the manufacturer of the corresponding Electronic gadget. It has a primary key as a unique identifier ID with attributes Name.
* Entity set **ElectronicType** indicates the type(Laptop, Phone or console) of the gadget, it has a single attribute which is primary key Kind (Laptop, Phone or Console).
* Entity set **OperatingSystem** indicates the OS installed on the electronic device and it has primary key as unique identifier ID and other attribute Name.
* If the Customer does not return the loaned device by due date he will be penalized. This is saved in entity set **Penalty** with primary key as unique ID with attributes Fee, DueDate and Status.

**Relations**

* Relation Loans: When a Customer **Loans** an Electronic device. This is a many-to-many relation. Relationship attribute: StartDate with other attributes DueDate, Amount and Status.
* Relation Notifies: When a Customer opts to be notified if a certain Electronic device is in stock again to be loaned. This is a many-to-many relation.
* Relation Paid: When Customer finishes paying their fee due to a penalty. Each Penalty is linked to exactly one Customer and a Customer can have multiple penalties. Therefore this is a one-to-many relationship.
* Relation OSInstalled: An Electronic device should have at least one OS installed. There is no restriction on whether if every OS should have an electronic device in the catalog.
* Relation ElectronicManufacturer: Similar to OSInstalled, an electronic device should have at least one manufacturer but every manufacturer need not have a device in the catalog. Therefore this is a many-to-one relationship.
* Finally, relation GadgetType indicates the binary relation between device and the type of device.

**Logical Schema for ER model**

Entities and Relations

* Customer (ID, FirstName, LastName, Age, Contact, Status)
* Penalty(ID, CustomerID, Fee, DueDate, Status) with foreign key CustomerID referring Customer(ID). Penalty data for corresponding deleted Customer can be removed. Cascade on delete.
* Electronic(ID, Name, Year, TotaCount, Rating, ManufacturerID, Kind) with foreign key ManufacturerID, Kind referencing to Manufacturer(ID) and ElectronicType(Kind)
* OperatingSystem(ID, Name)
* ElectronicType(Kind, Version)
* Manufacturer(ID, Name)
* Loans(CustomerID, ElectronicID, StartDate, DueDate, Amount, Status) with foreign keys (CustomerID, ElectronicID) referring to Customer(ID), Electronic(ID). Relation value can be deleted once the Customer/Electronic item is deleted. Cascade on Delete.
* Notifies(CustomerID, ElectronicID) foreign key CustomerID, ElectronicID referring to Customer(ID) and Electronic(ID)
* OSInstalled(ElectronicID, OperatingSystemID) foreign key ElectronicID, OperatingSystemID referring to Electronic(ID) and OperatingSystem(ID)

**Implementation**

Made a few name changes like ID to CustomerID, Status to Customer Status in order to avoid using SQL default terms.

Code available at: <https://github.com/krishnakarthiknyu/databasecourseproject>

Tools used:

* Local MySQL server on Ubuntu
* Azure Data Studio as IDE for writing SQL scripts. It helped in running sql commands and queries similar to python scripts in Jupyternotebook.
* Python to convert the dataset into desired format (need numpy and pandas)
* Azure Database for MySQL flexible server

Data processing and cleaning up:

* For now, I am considering only Laptops as part of the project
* Dataset used: <https://www.kaggle.com/code/danielbethell/laptop-prices-prediction/data>
* It has various laptop information and price details. I’ve used python to clean up the data and only have the required information.
* The cleaned up data is formatted into desired format and saved as manufacturers.csv, electronics.csv, osinstalled.csv
* The above mentioned python code can be viewed here: [code](https://github.com/krishnakarthiknyu/databasecourseproject/blob/main/laptopdataprocessing.py)

**Database definitions and creation of tables**

SQL script: [DefinitionsAndInitiation.sql](https://github.com/krishnakarthiknyu/databasecourseproject/blob/main/DefinitionsAndInitiation.sql)

Database with the name *‘Rentaldb’* is created.

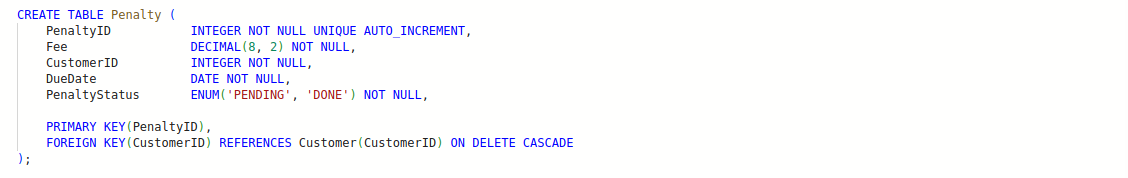


Tables are created for each of the Entities mentioned in previous parts

*Customer*



*Penalty*

**

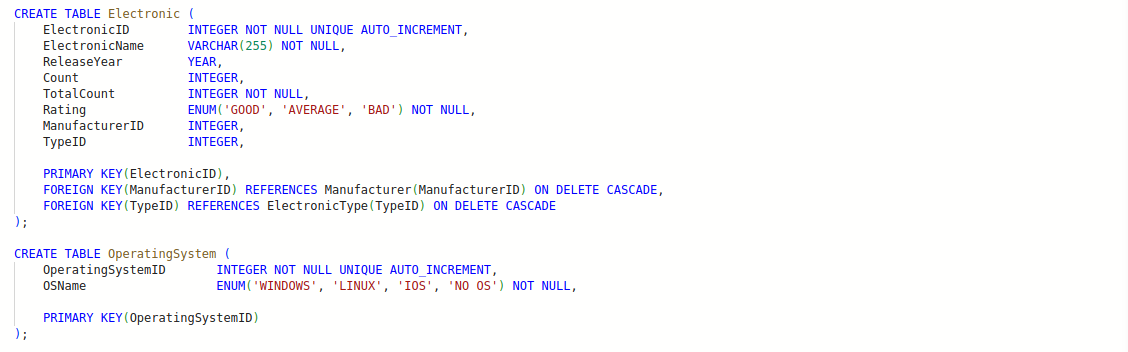
*Manufacturer*

**

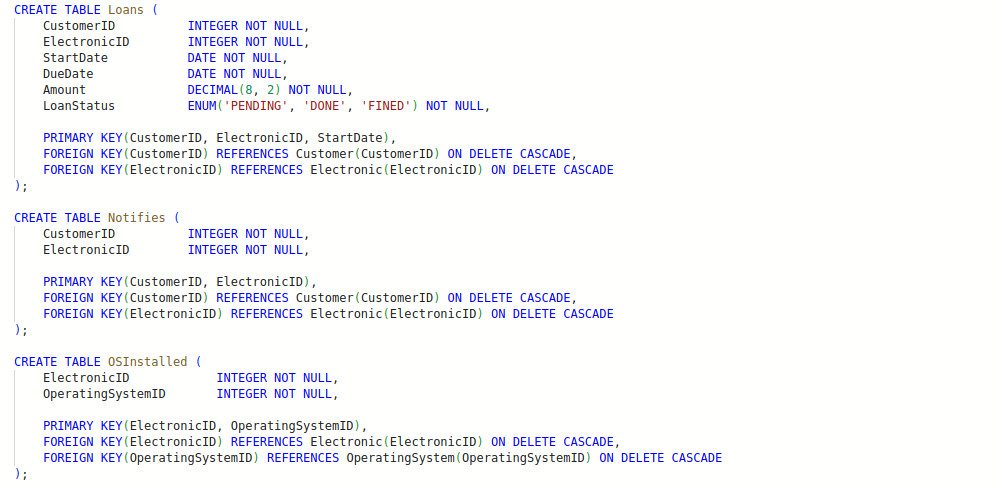
*ElectronicType*

**

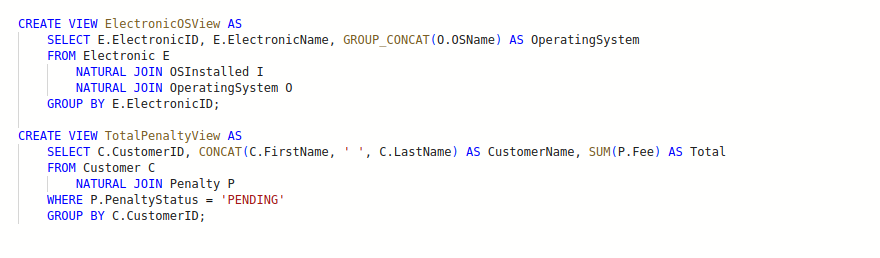
*Electronic and Operating System*

**

Tables are created for relations:  *Loans, Notifies, OSInstalled*



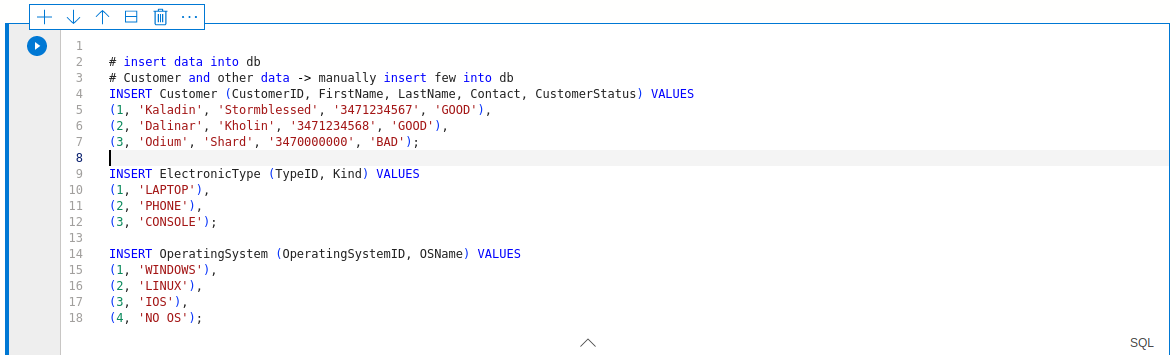
Couple of views are created for better viewing of the data: *ElectronicOSView and TotalPenaltyView*



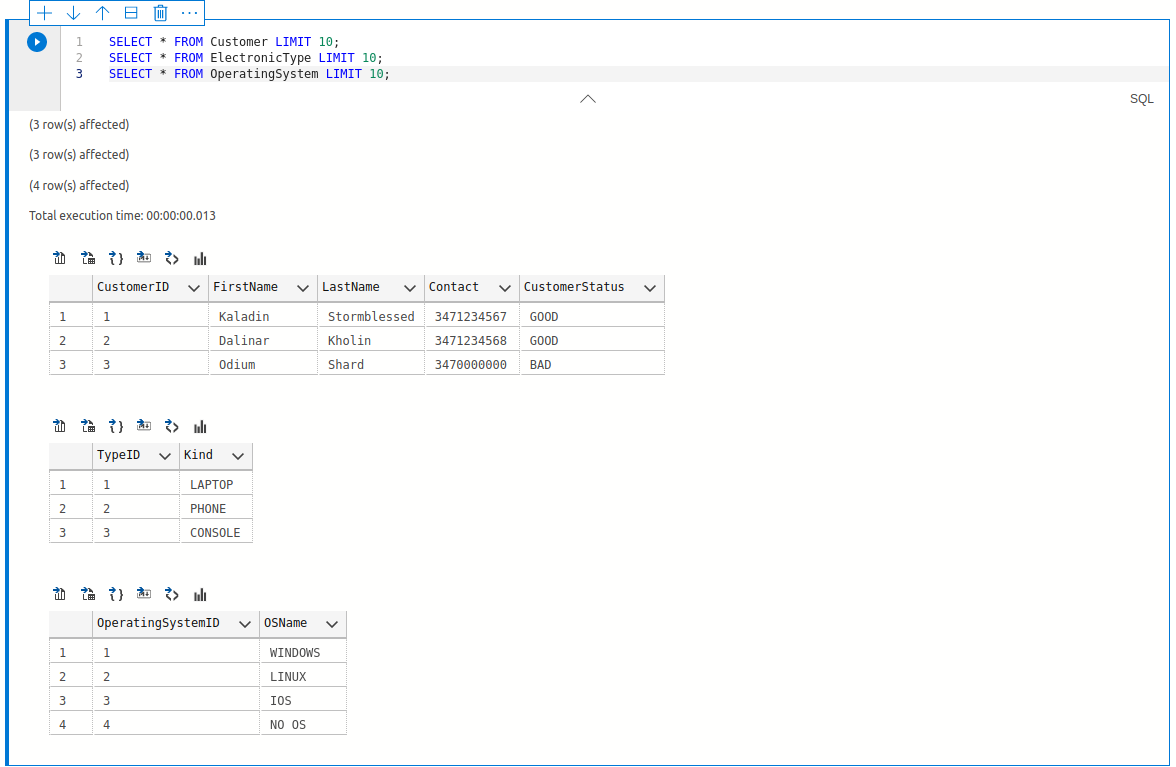
**Inserting some test data into the database**

SQL Script: [InsertData.sql](https://github.com/krishnakarthiknyu/databasecourseproject/blob/main/InsertData.sql)

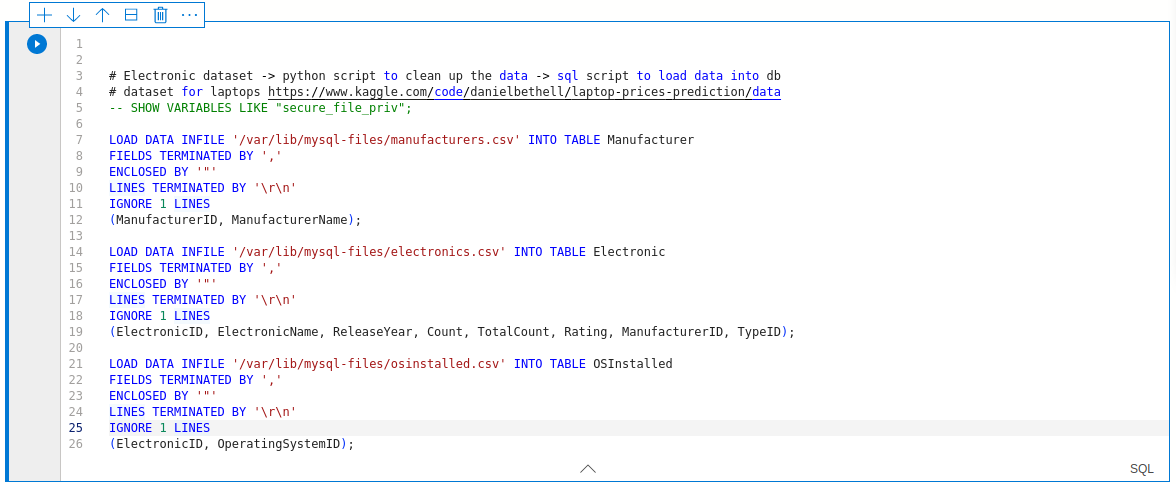
Here I inserted few rows of data into *Customer, ElectronicType, OperatingSystem* tables

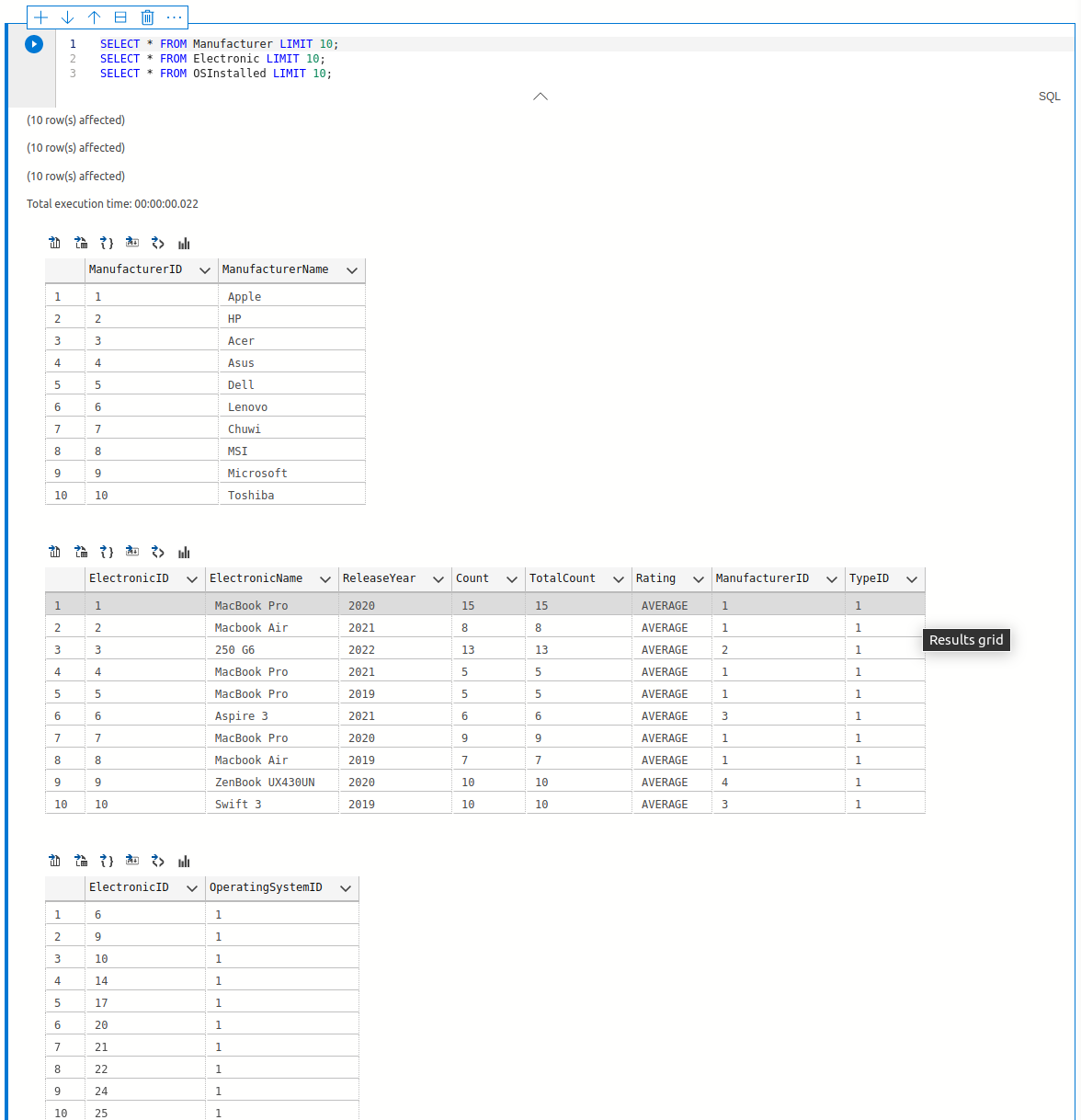


Querying the data to check

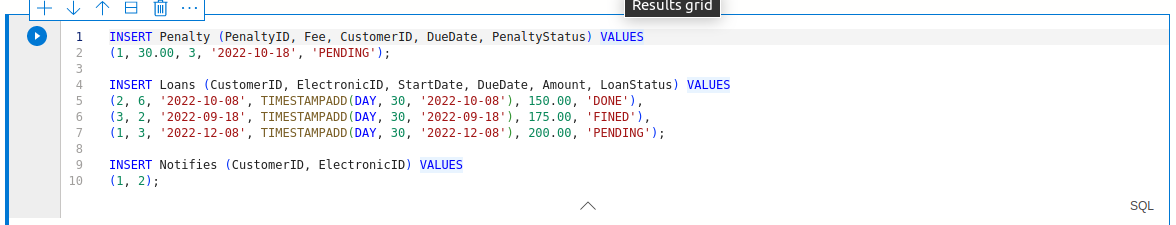


In order to upload the data from ‘manufacturers.csv’, ‘electronics.csv’ and ‘osinstalled.csv’ i used LOAD INTO command in SQL but it gave few issues as SQL only allow files inside sequre\_file\_priv directory which for my case is: ‘/var/lib/mysql-files/manufacturers.csv’

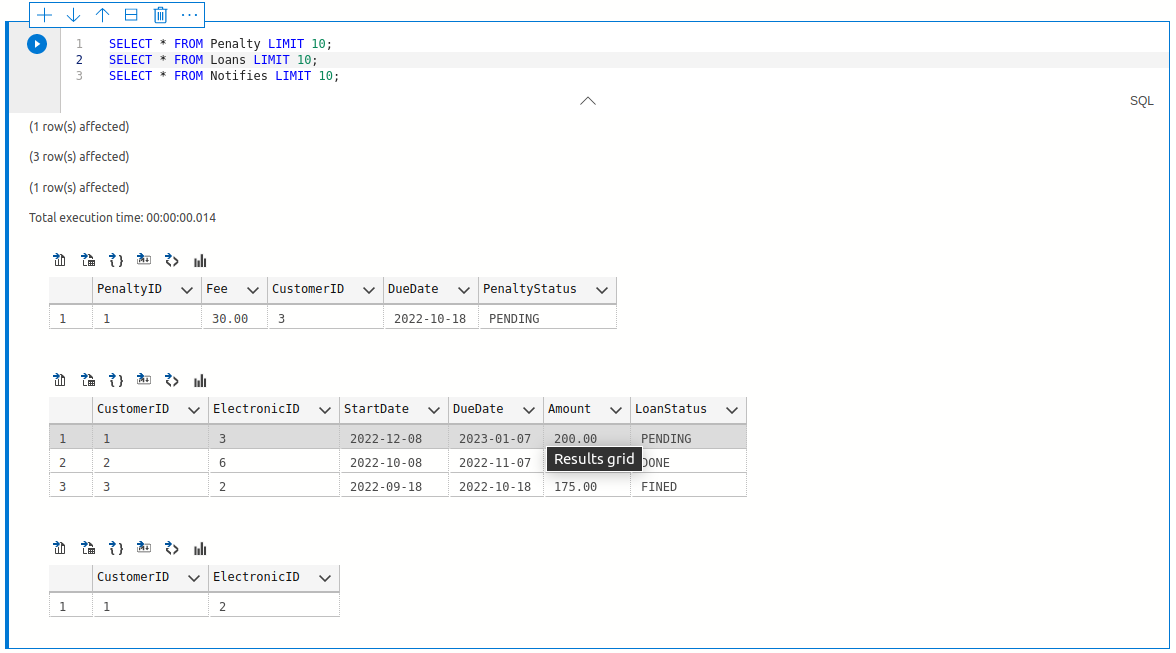


Queried the data to verify.

Inserted relations data



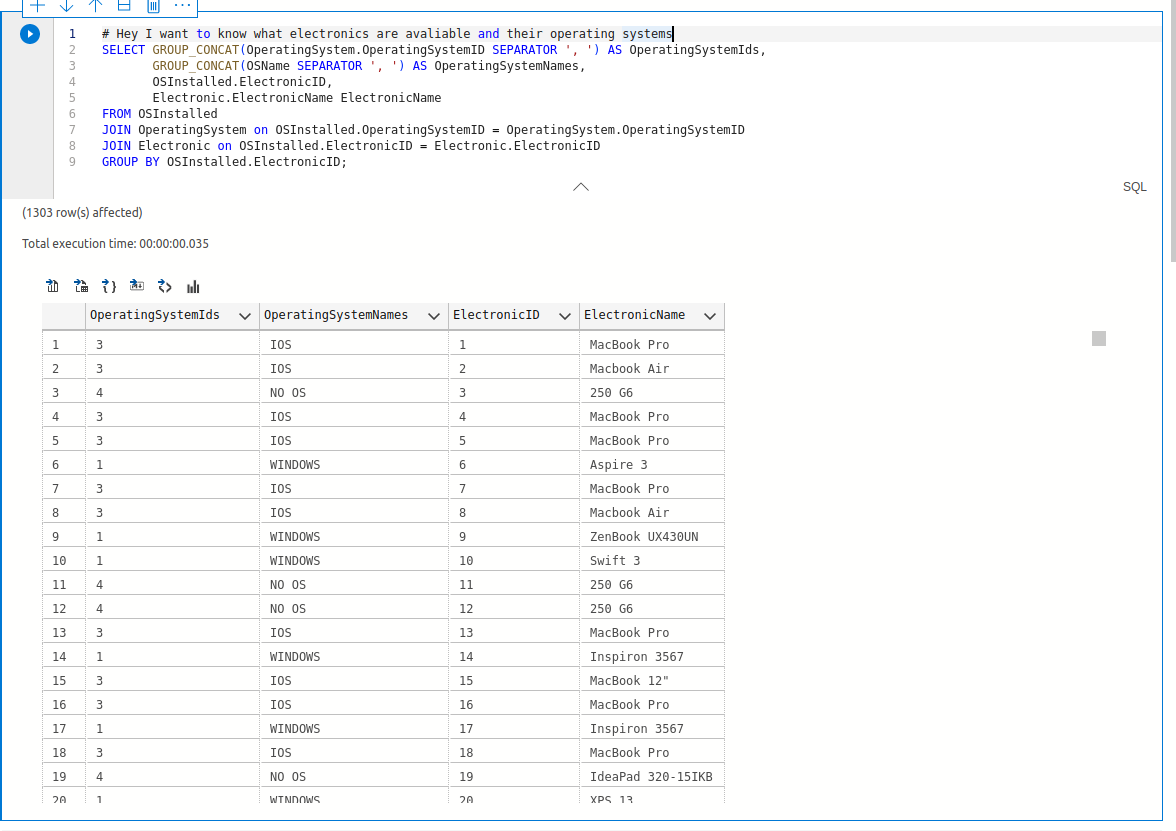
Query



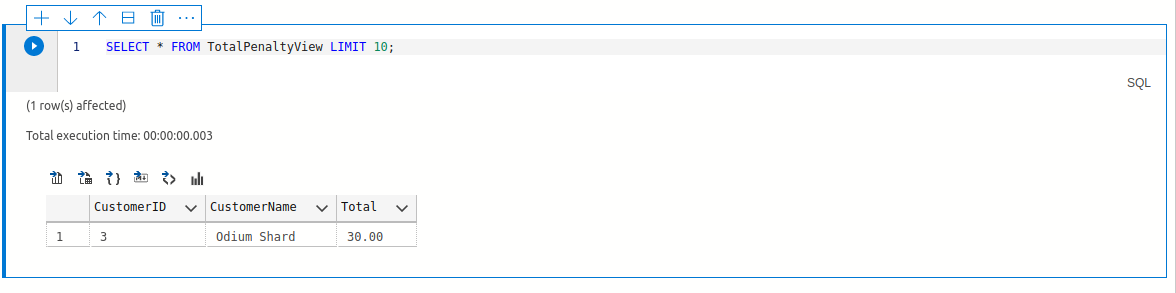
**Queries and Updates essential for the business use cases**

SQL Script: [Procedures\_and\_examples.sql](https://github.com/krishnakarthiknyu/databasecourseproject/blob/main/procedures_and_examples.sql)

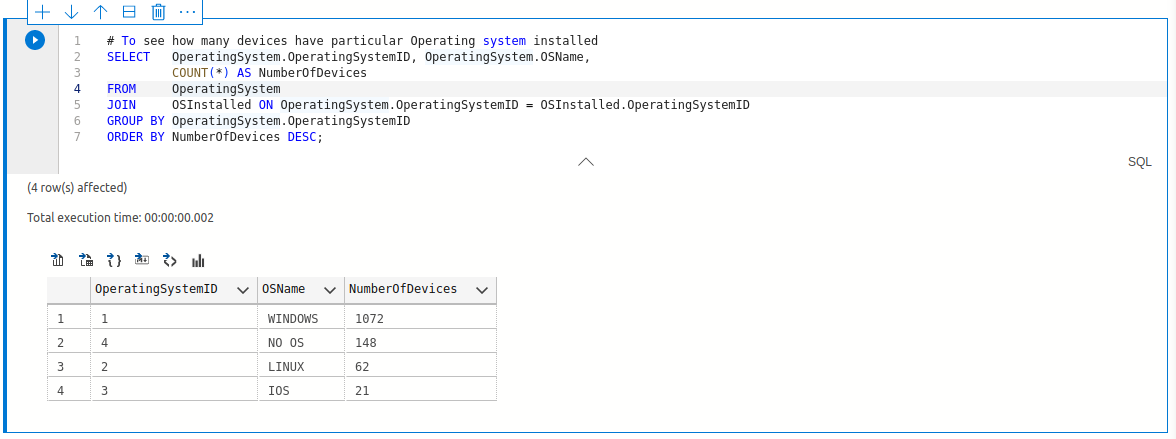
* If we want to know what electronics are available and their operating system details



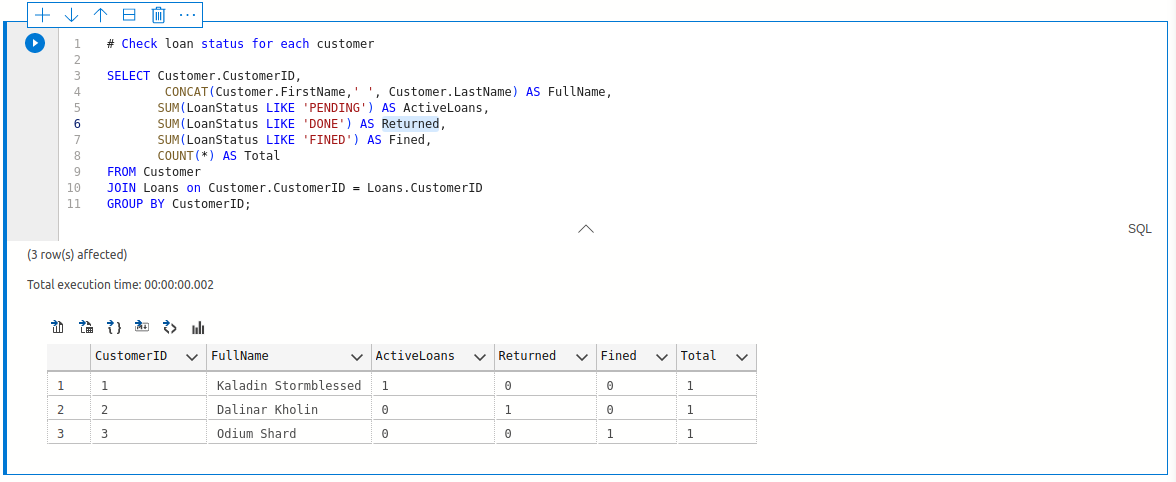
* If we want to know total penalty due by a customer



* To view total number of devices which have a particular OS installed (A device can have multiple OS installed, dual boot etc)



* To check all customers active loans and pending loan details

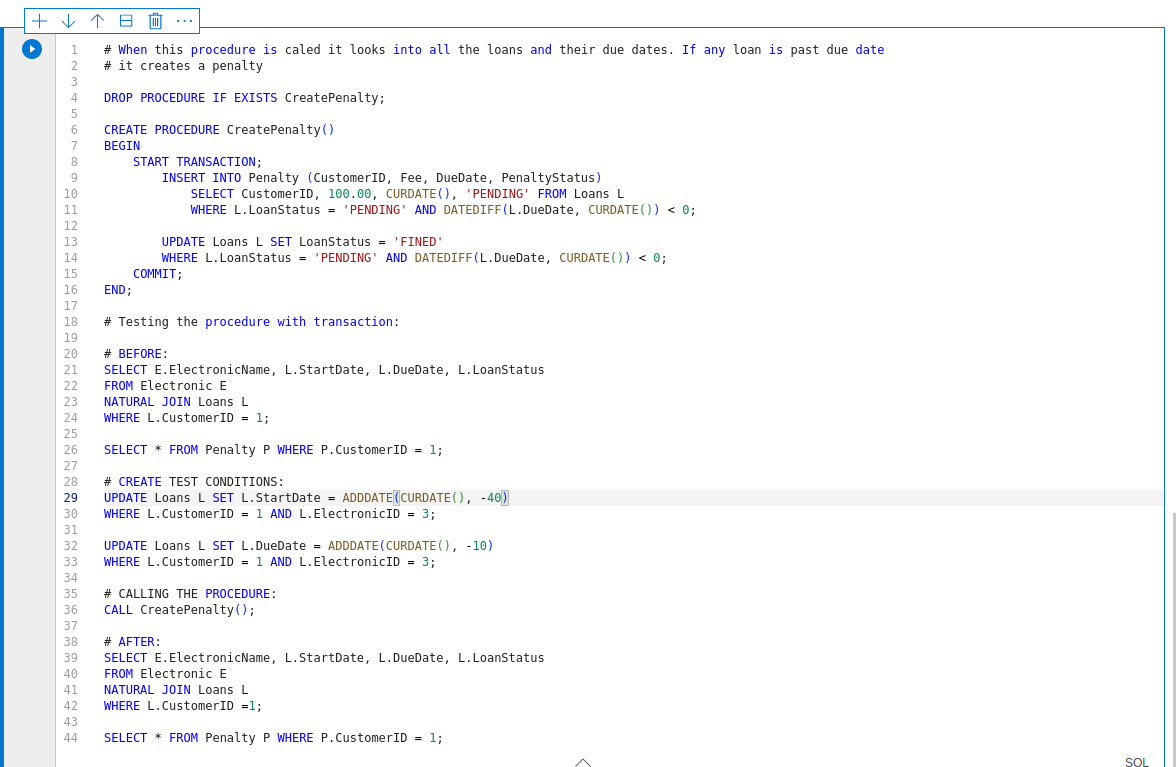


**Other Procedures and Events**

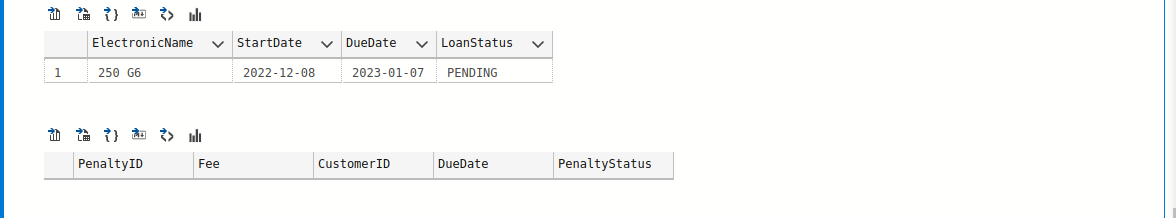
* Procedure for loaning a device to a customer.



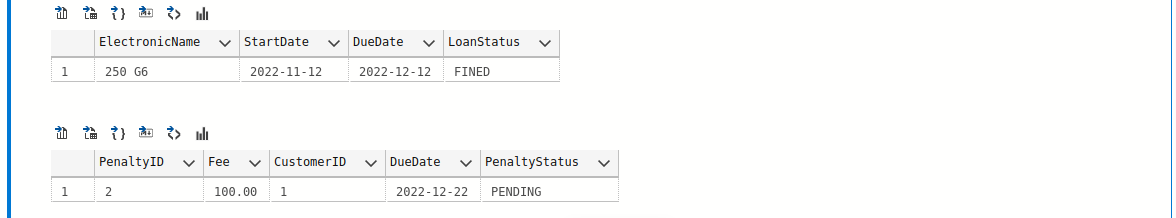
* Procedure to penalize customers if they missed the due date.



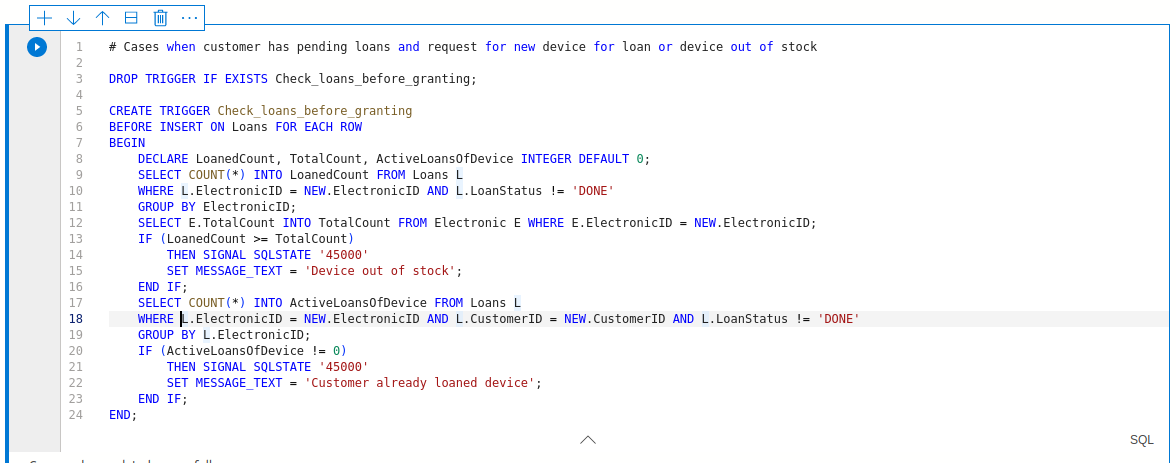
Before penalty:



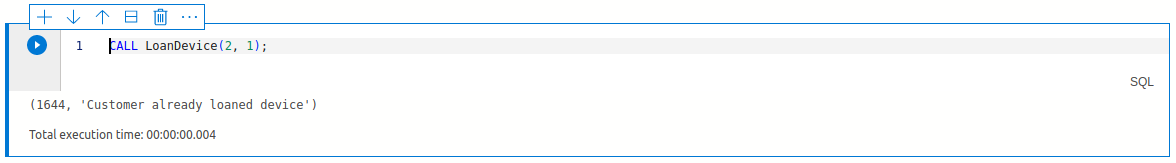
After penalty:



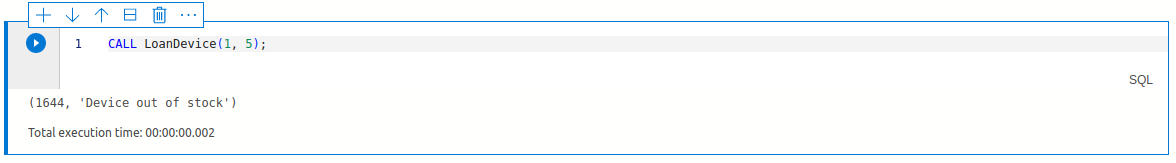
* If a customer has a pending loan and request for new loan - Deny



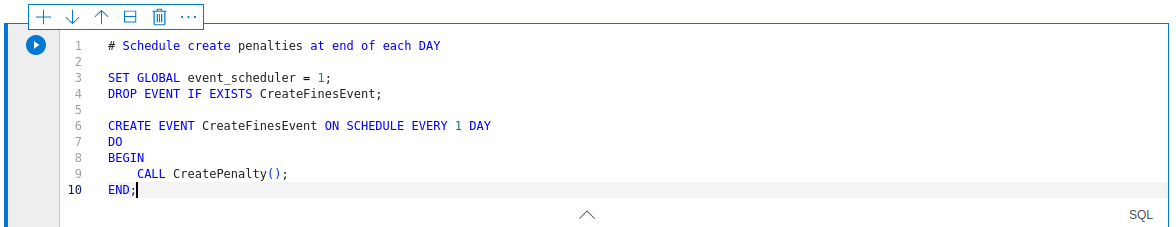
Test



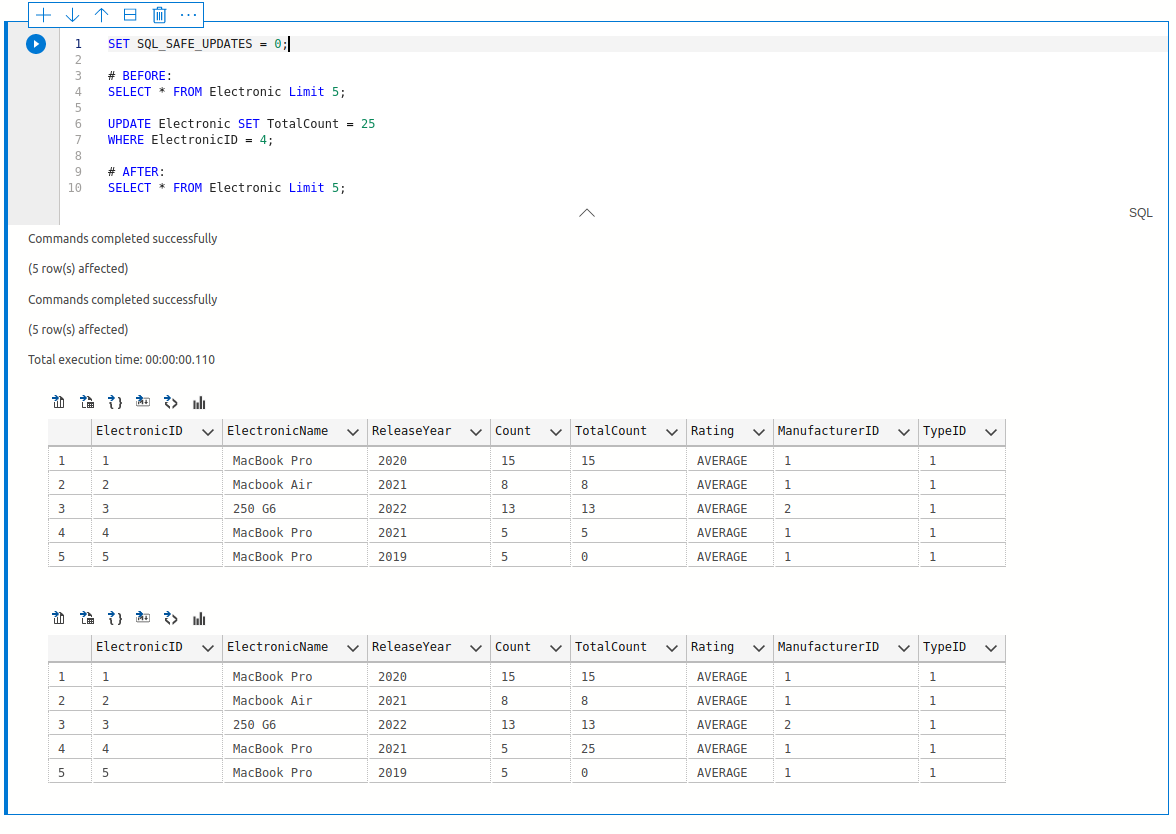
* Respond device out of stock if the electronic item is all loaned out



* At EOD check for all late due loans and create penalties for each of them

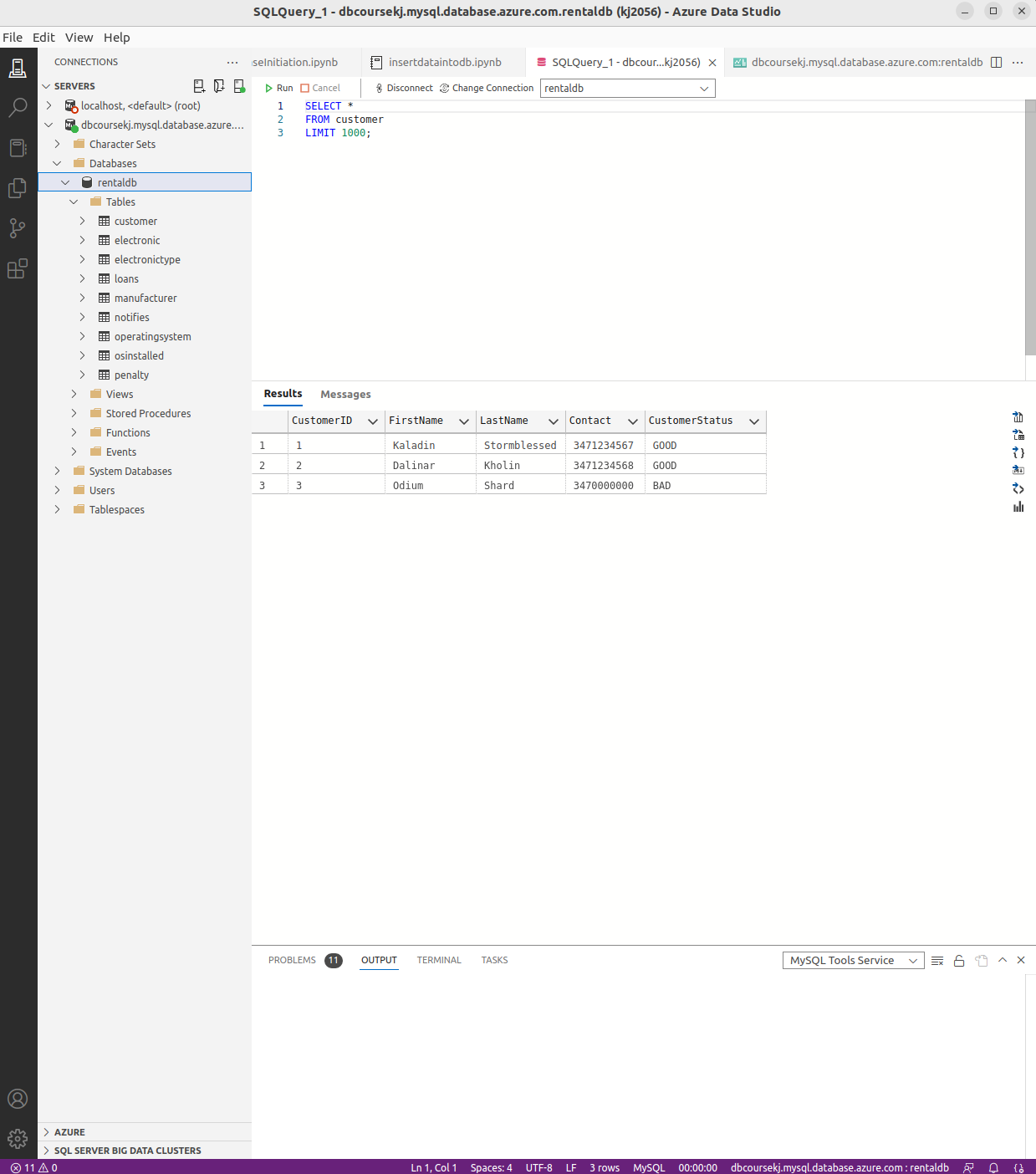


* Some updates on tables



**Update on implementation progress**

All these are done in local db and later planning to do it on Azure MySQL server. For now, I was able to insert some of the data into the cloud.



**Improvements and Future work**

* For current implementation, the laptop data processing and cleaning up is done in external python script. Ideally this can be done as a service function, where when a data dump is received, it automatically processes the data.
* More details about the electronic devices like RAM, CPU etc can be maintained in future also devices other than laptops (like mobiles, consoles etc) should be added.
* Predictions for right pricing of devices can be made in future based on popularity and depreciation from its release year.
* Simple secure login and UI could be useful to view.
* When a customer subscribes to be notified when an out of stock device is back in stock, the business should be able to notify the customer through his contact details.

**How to run this locally**

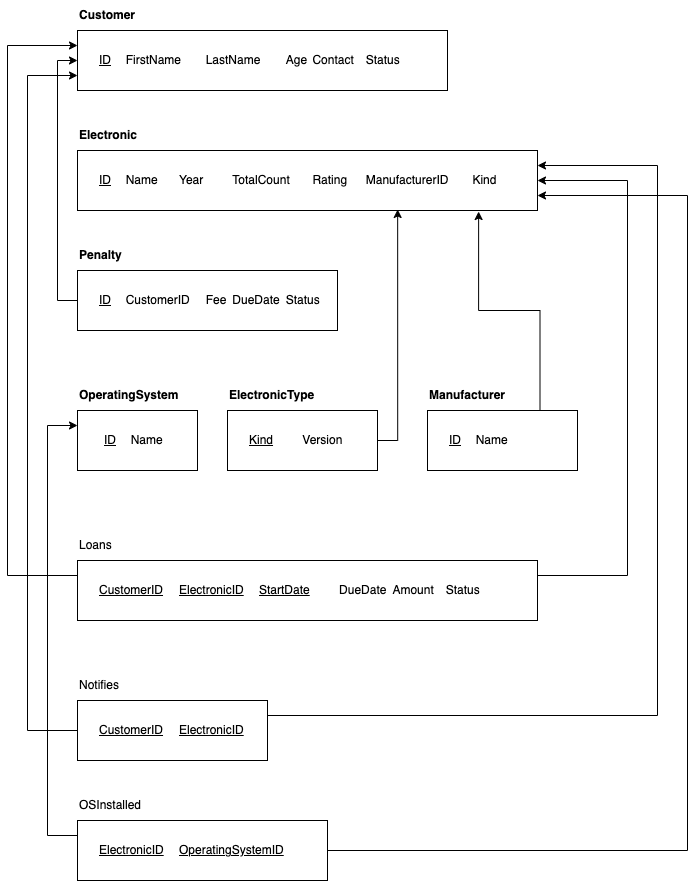
1. For the laptop pricing data, we need Python3, numpy and pandas. Or you can find the required CSV files in codebase ‘data/’ directory.
   1. RUN `python laptopdataprocessing.py`
   2. to generate the required data in csv format.
2. MySQL server running
3. Run `DatabaseInitiation.sql` SQL script that will create the database and the tables
4. Run `InsertData.sql` script to insert data into the tables. Before running the script make sure to
5. Find out the secure file directory by `SHOW VARIABLES LIKE “secure\_file\_priv”`
6. Move the csv files created in step 1 to the above directory
7. Or we can skip a, b if we use sql management studio and upload the CSV files.

5. Run `procedures\_and\_examples.sql` to run some example queries and updates.

**References and Tools**

* Dataset: <https://www.kaggle.com/code/danielbethell/laptop-prices-prediction/data>
* Cleaning up dataset: <https://www.kaggle.com/code/danielbethell/laptop-prices-prediction/notebook>
* Inspired from [library catalog system](https://github.com/seby-sbirna/Database-Design-Modeling-and-Programming-of-National-Library-Catalog-System)

Azure Data [Studio](https://azure.microsoft.com/en-us/products/data-studio/#overview)



**Normalization and extensions**

At least 3NF is required for all to not suffer from update anomalies in the database.

* Customer (ID, FirstName, LastName, Age, Contact, Status)

All attributes are atomic and it has a single attribute as primary key, hence at least 2NF. As there are no transitive dependency for non-primary attributes and 2NF, therefore it satisfies 3NF

* Penalty(ID, CustomerID, Fee, DueDate, Status)

All attributes are atomic and it has a single attribute as primary key, hence at least 2NF.

Non-primary key attributes have no transitive dependency so it satisfies 3NF

* Electronic(ID, Name, Year, TotaCount, Rating, ManufacturerID, Kind)

All attributes are atomic and it has a single attribute as primary key, hence at least 2NF.

Non-primary key attributes have no transitive dependency so it satisfies 3NF

* OperatingSystem(ID, Name)

All attributes are atomic and it has a single attribute as primary key, hence at least 2NF.

Non-primary key attributes have no transitive dependency so it satisfies 3NF

* ElectronicType(Kind, Version)

All attributes are atomic and it has a single attribute as primary key, hence at least 2NF.

Non-primary key attributes have no transitive dependency so it satisfies 3NF

* Manufacturer(ID, Name)

All attributes are atomic and it has a single attribute as primary key, hence at least 2NF.

Non-primary key attributes have no transitive dependency so it satisfies 3NF

* Loans(CustomerID, ElectronicID, StartDate, DueDate, Amount, Status)

All attributes are atomic and non-primary key attributes are not partially dependent on the candidate key.

Non-primary key attributes have no transitive dependency so it satisfies 3NF

* Notifies(CustomerID, ElectronicID)

All attributes are atomic and no non-primary keys so at least 3NF.

* OSInstalled(ElectronicID, OperatingSystemID)

Similar to Notifiedz, this is at least 3NF.