

Name: Krishna Karthik Reddy Jonnala

NYU ID: kj2056

Course Section Number: CSCI-GA.2433-001

### Final Report

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

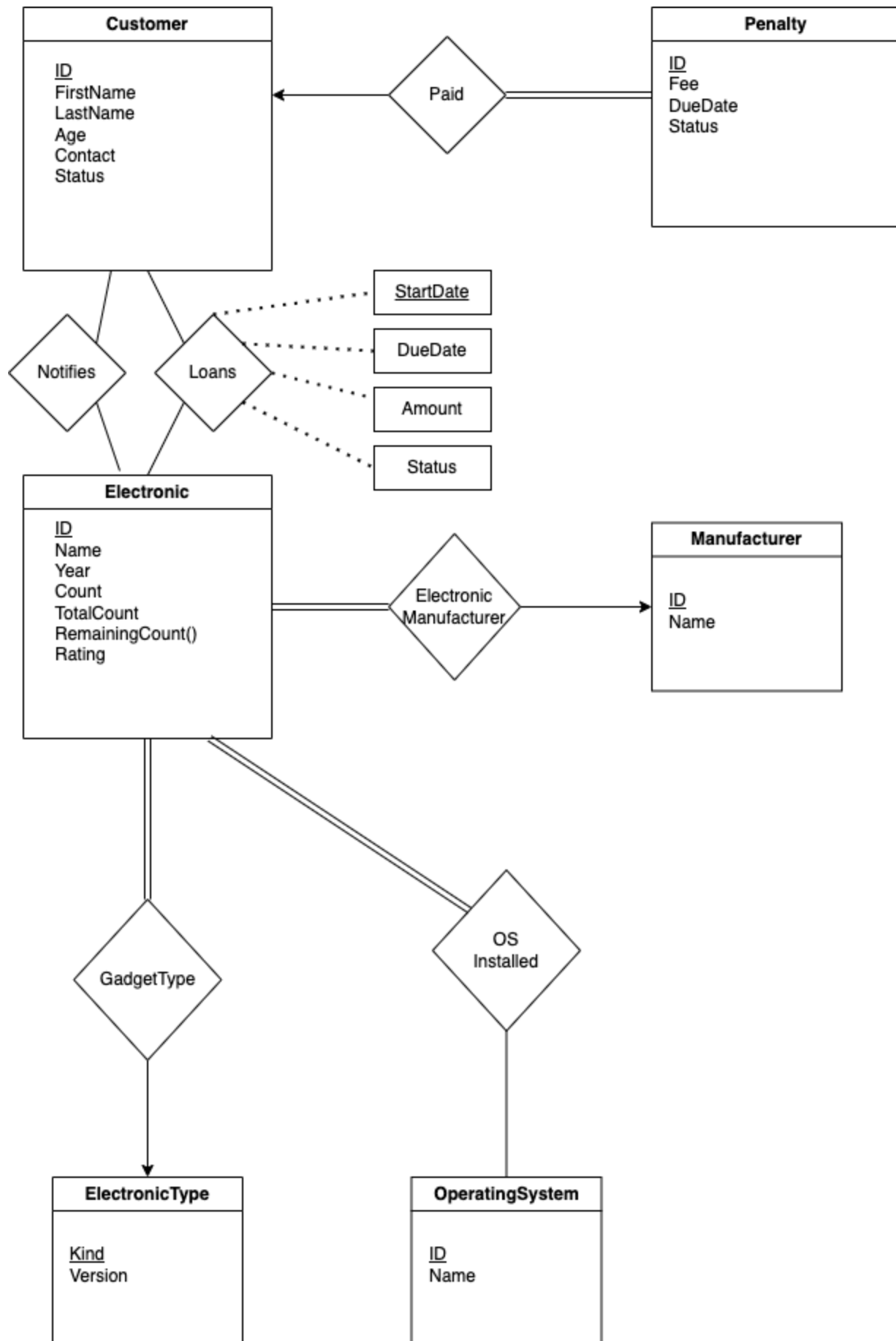
**Professor's Comments:**

#### 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 only Laptop, 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.

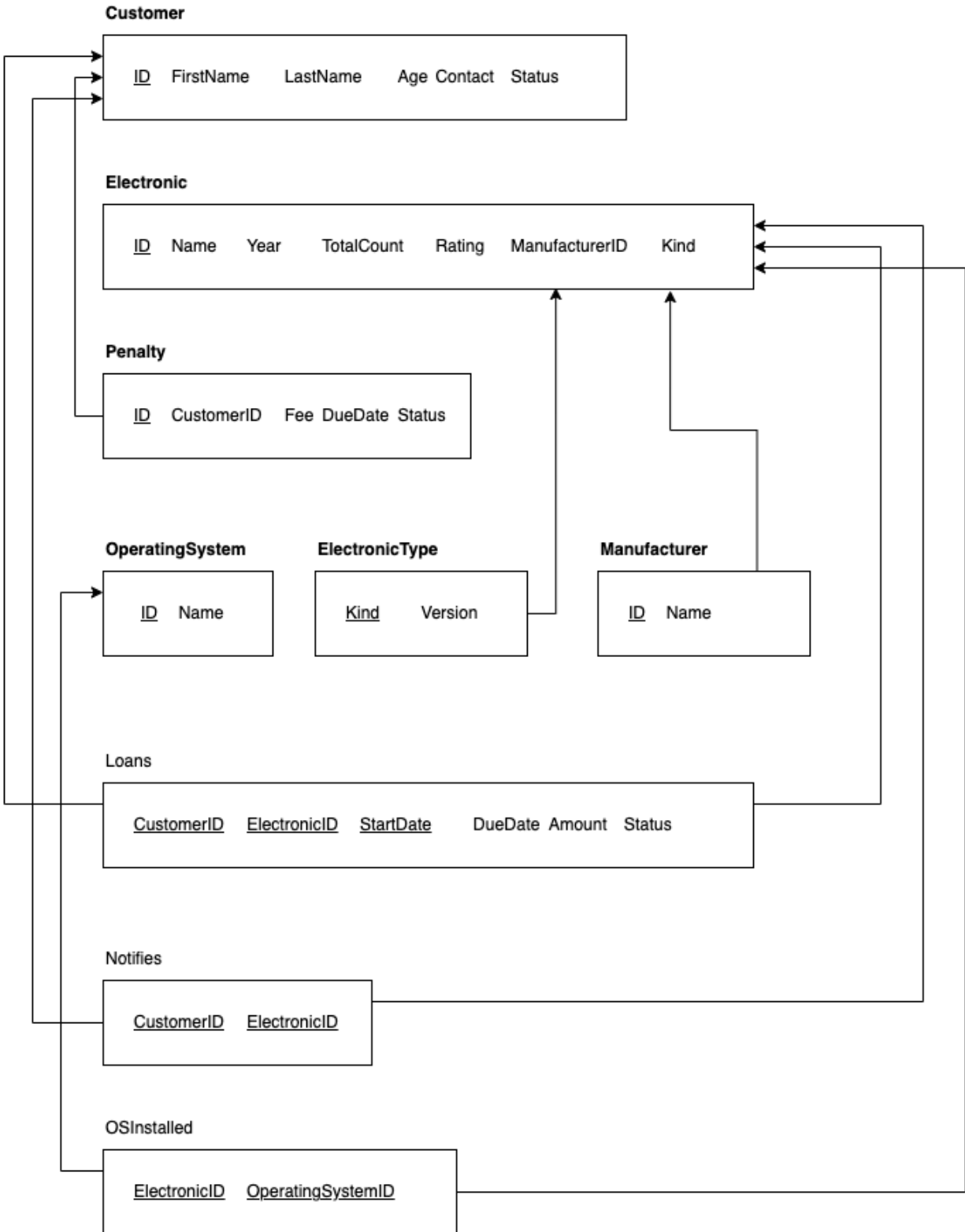
## Relations

- 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)



## **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, TotalCount, 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 Notifies, this is at least 3NF.

## **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](#)

## Database definitions and creation of tables

SQL script: [DefinitionsAndInitiation.sql](#)

Database with the name 'Rentaldb' is created.

```
Run Cancel | Disconnect Change Connection Select Database
1 # Creating database
2
3 DROP DATABASE IF EXISTS Rentaldb;
4 CREATE DATABASE Rentaldb;
5
6 USE Rentaldb;
7
```

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

### Customer

```
CREATE TABLE Customer (
  CustomerID INTEGER NOT NULL UNIQUE AUTO_INCREMENT,
  FirstName VARCHAR(255) NOT NULL,
  LastName VARCHAR(255) NOT NULL,
  Contact VARCHAR(255) NOT NULL,
  CustomerStatus ENUM('GOOD', 'BAD') NOT NULL,
  PRIMARY KEY(CustomerID)
);
```

### Penalty

```
CREATE TABLE Penalty (
  PenaltyID INTEGER NOT NULL UNIQUE AUTO_INCREMENT,
  Fee DECIMAL(8, 2) NOT NULL,
  CustomerID INTEGER NOT NULL,
  DueDate DATE NOT NULL,
  PenaltyStatus ENUM('PENDING', 'DONE') NOT NULL,
  PRIMARY KEY(PenaltyID),
  FOREIGN KEY(CustomerID) REFERENCES Customer(CustomerID) ON DELETE CASCADE
);
```

### Manufacturer

```
CREATE TABLE Manufacturer (
  ManufacturerID INTEGER NOT NULL UNIQUE AUTO_INCREMENT,
  ManufacturerName VARCHAR(255) NOT NULL,
  PRIMARY KEY(ManufacturerID)
);
```

### ElectronicType

```
CREATE TABLE ElectronicType (
  TypeID INTEGER NOT NULL UNIQUE AUTO_INCREMENT,
  Kind ENUM('LAPTOP', 'PHONE', 'CONSOLE') NOT NULL,
  PRIMARY KEY(TypeID)
);
```



## Electronic and Operating System

```
CREATE TABLE Electronic (
    ElectronicID      INTEGER NOT NULL UNIQUE AUTO_INCREMENT,
    ElectronicName    VARCHAR(255) NOT NULL,
    ReleaseYear       YEAR,
    Count             INTEGER,
    TotalCount        INTEGER NOT NULL,
    Rating            ENUM('GOOD', 'AVERAGE', 'BAD') NOT NULL,
    ManufacturerID    INTEGER,
    TypeID            INTEGER,

    PRIMARY KEY(ElectronicID),
    FOREIGN KEY(ManufacturerID) REFERENCES Manufacturer(ManufacturerID) ON DELETE CASCADE,
    FOREIGN KEY(TypeID) REFERENCES ElectronicType(TypeID) ON DELETE CASCADE
);

CREATE TABLE OperatingSystem (
    OperatingSystemID INTEGER NOT NULL UNIQUE AUTO_INCREMENT,
    OSName            ENUM('WINDOWS', 'LINUX', 'IOS', 'NO OS') NOT NULL,

    PRIMARY KEY(OperatingSystemID)
);
```

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

```
CREATE TABLE Loans (
    CustomerID        INTEGER NOT NULL,
    ElectronicID       INTEGER NOT NULL,
    StartDate         DATE NOT NULL,
    DueDate           DATE NOT NULL,
    Amount            DECIMAL(8, 2) NOT NULL,
    LoanStatus        ENUM('PENDING', 'DONE', 'FINED') NOT NULL,

    PRIMARY KEY(CustomerID, ElectronicID, StartDate),
    FOREIGN KEY(CustomerID) REFERENCES Customer(CustomerID) ON DELETE CASCADE,
    FOREIGN KEY(ElectronicID) REFERENCES Electronic(ElectronicID) ON DELETE CASCADE
);

CREATE TABLE Notifies (
    CustomerID        INTEGER NOT NULL,
    ElectronicID       INTEGER NOT NULL,

    PRIMARY KEY(CustomerID, ElectronicID),
    FOREIGN KEY(CustomerID) REFERENCES Customer(CustomerID) ON DELETE CASCADE,
    FOREIGN KEY(ElectronicID) REFERENCES Electronic(ElectronicID) ON DELETE CASCADE
);

CREATE TABLE OSInstalled (
    ElectronicID       INTEGER NOT NULL,
    OperatingSystemID  INTEGER NOT NULL,

    PRIMARY KEY(ElectronicID, OperatingSystemID),
    FOREIGN KEY(ElectronicID) REFERENCES Electronic(ElectronicID) ON DELETE CASCADE,
    FOREIGN KEY(OperatingSystemID) REFERENCES OperatingSystem(OperatingSystemID) ON DELETE CASCADE
);
```

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

```
CREATE VIEW ElectronicOSView AS
SELECT E.ElectronicID, E.ElectronicName, GROUP_CONCAT(O.OSName) AS OperatingSystem
FROM Electronic E
    NATURAL JOIN OSInstalled I
    NATURAL JOIN OperatingSystem O
GROUP BY E.ElectronicID;

CREATE VIEW TotalPenaltyView AS
SELECT C.CustomerID, CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName, SUM(P.Fee) AS Total
FROM Customer C
    NATURAL JOIN Penalty P
WHERE P.PenaltyStatus = 'PENDING'
GROUP BY C.CustomerID;
```

## Inserting some test data into the database

SQL Script: [InsertData.sql](#)

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

```
+ ↓ ↑ ☐ ☐ ...  
1  
2 # insert data into db  
3 # Customer and other data -> manually insert few into db  
4 INSERT Customer (CustomerID, FirstName, LastName, Contact, CustomerStatus) VALUES  
5 (1, 'Kaladin', 'Stormblessed', '3471234567', 'GOOD'),  
6 (2, 'Dalinar', 'Kholin', '3471234568', 'GOOD'),  
7 (3, 'Odium', 'Shard', '3470000000', 'BAD');  
8  
9 INSERT ElectronicType (TypeID, Kind) VALUES  
10 (1, 'LAPTOP'),  
11 (2, 'PHONE'),  
12 (3, 'CONSOLE');  
13  
14 INSERT OperatingSystem (OperatingSystemID, OSName) VALUES  
15 (1, 'WINDOWS'),  
16 (2, 'LINUX'),  
17 (3, 'IOS'),  
18 (4, 'NO OS');
```

SQL

## Querying the data to check

```
+ ↓ ↑ ☐ ☐ ...  
1 SELECT * FROM Customer LIMIT 10;  
2 SELECT * FROM ElectronicType LIMIT 10;  
3 SELECT * FROM OperatingSystem LIMIT 10;
```

(3 row(s) affected)

(3 row(s) affected)

(4 row(s) affected)

Total execution time: 00:00:00.013

☐ ☐ ☐ ☐ ☐ ☐ ☐

	CustomerID	FirstName	LastName	Contact	CustomerStatus
1	1	Kaladin	Stormblessed	3471234567	GOOD
2	2	Dalinar	Kholin	3471234568	GOOD
3	3	Odium	Shard	3470000000	BAD

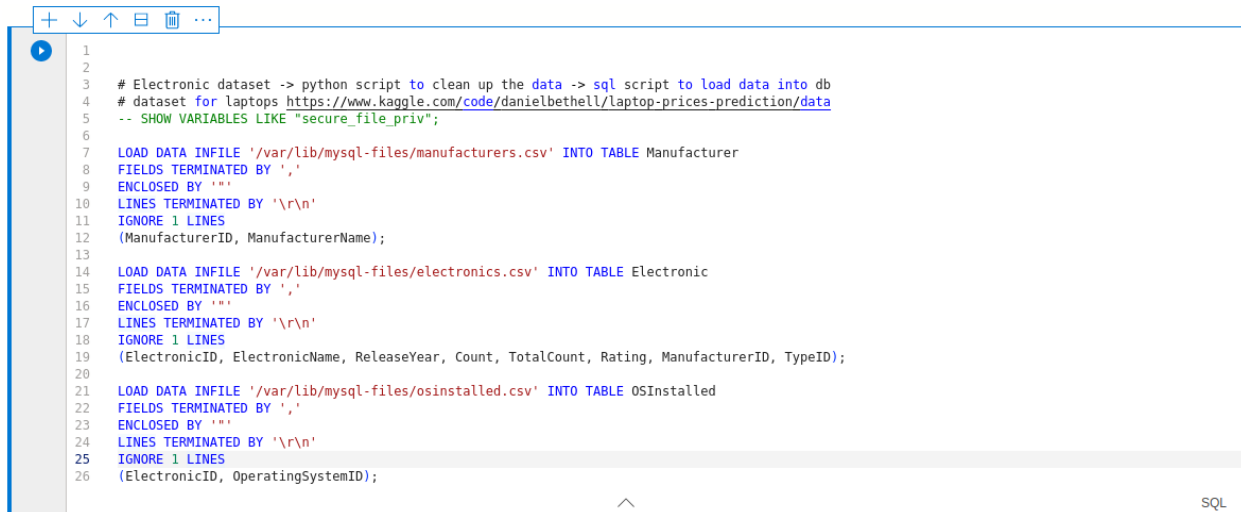
☐ ☐ ☐ ☐ ☐ ☐ ☐

	TypeID	Kind
1	1	LAPTOP
2	2	PHONE
3	3	CONSOLE

☐ ☐ ☐ ☐ ☐ ☐ ☐

	OperatingSystemID	OSName
1	1	WINDOWS
2	2	LINUX
3	3	IOS
4	4	NO OS

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 secure\_file\_priv directory which for my case is: '/var/lib/mysql-files/manufacturers.csv'



```
1
2
3 # Electronic dataset -> python script to clean up the data -> sql script to load data into db
4 # dataset for laptops https://www.kaggle.com/code/danielbethell/laptop-prices-prediction/data
5 -- SHOW VARIABLES LIKE "secure_file_priv";
6
7 LOAD DATA INFILE '/var/lib/mysql-files/manufacturers.csv' INTO TABLE Manufacturer
8 FIELDS TERMINATED BY ','
9 ENCLOSED BY ''
10 LINES TERMINATED BY '\r\n'
11 IGNORE 1 LINES
12 (ManufacturerID, ManufacturerName);
13
14 LOAD DATA INFILE '/var/lib/mysql-files/electronics.csv' INTO TABLE Electronic
15 FIELDS TERMINATED BY ','
16 ENCLOSED BY ''
17 LINES TERMINATED BY '\r\n'
18 IGNORE 1 LINES
19 (ElectronicID, ElectronicName, ReleaseYear, Count, TotalCount, Rating, ManufacturerID, TypeID);
20
21 LOAD DATA INFILE '/var/lib/mysql-files/osinstalled.csv' INTO TABLE OSInstalled
22 FIELDS TERMINATED BY ','
23 ENCLOSED BY ''
24 LINES TERMINATED BY '\r\n'
25 IGNORE 1 LINES
26 (ElectronicID, OperatingSystemID);
```

SQL

Queried the data to verify.

+ ↓ ↑ ☰ ...

```

1 SELECT * FROM Manufacturer LIMIT 10;
2 SELECT * FROM Electronic LIMIT 10;
3 SELECT * FROM OSInstalled LIMIT 10;
```

SQL

(10 row(s) affected)

(10 row(s) affected)

(10 row(s) affected)

Total execution time: 00:00:00.022

🔍 📄 🔍 ⚙️ ↺ 📊

	ManufacturerID ▾	ManufacturerName ▾
1	1	Apple
2	2	HP
3	3	Acer
4	4	Asus
5	5	Dell
6	6	Lenovo
7	7	Chuwi
8	8	MSI
9	9	Microsoft
10	10	Toshiba

🔍 📄 🔍 ⚙️ ↺ 📊

	ElectronicID ▾	ElectronicName ▾	ReleaseYear ▾	Count ▾	TotalCount ▾	Rating ▾	ManufacturerID ▾	TypeID ▾
1	1	MacBook Pro	2020	15	15	AVERAGE	1	1
2	2	Macbook Air	2021	8	8	AVERAGE	1	1
3	3	250 G6	2022	13	13	AVERAGE	2	1
4	4	MacBook Pro	2021	5	5	AVERAGE	1	1
5	5	MacBook Pro	2019	5	5	AVERAGE	1	1
6	6	Aspire 3	2021	6	6	AVERAGE	3	1
7	7	MacBook Pro	2020	9	9	AVERAGE	1	1
8	8	Macbook Air	2019	7	7	AVERAGE	1	1
9	9	ZenBook UX430UN	2020	10	10	AVERAGE	4	1
10	10	Swift 3	2019	10	10	AVERAGE	3	1

Results grid

🔍 📄 🔍 ⚙️ ↺ 📊

	ElectronicID ▾	OperatingSystemID ▾
1	6	1
2	9	1
3	10	1
4	14	1
5	17	1
6	20	1
7	21	1
8	22	1
9	24	1
10	25	1

Inserted relations data

+ ↓ ↑ ☰ ☹ ...

▶

1   INSERT   Penalty   (PenaltyID, Fee, CustomerID, DueDate, PenaltyStatus)   VALUES

2   (1, 30.00, 3, '2022-10-18', 'PENDING');

3

4   INSERT   Loans   (CustomerID, ElectronicID, StartDate, DueDate, Amount, LoanStatus)   VALUES

5   (2, 6, '2022-10-08', TIMESTAMPADD(DAY, 30, '2022-10-08'), 150.00, 'DONE');

6   (3, 2, '2022-09-18', TIMESTAMPADD(DAY, 30, '2022-09-18'), 175.00, 'FINED');

7   (1, 3, '2022-12-08', TIMESTAMPADD(DAY, 30, '2022-12-08'), 200.00, 'PENDING');

8

9   INSERT   Notices   (CustomerID, ElectronicID)   VALUES

10   (1, 2);

SQL

Query

+ ↓ ↑ ☰ ☹ ...

▶

1   SELECT \* FROM Penalty LIMIT 10;

2   SELECT \* FROM Loans LIMIT 10;

3   SELECT \* FROM Notices LIMIT 10;

SQL

(1 row(s) affected)

(3 row(s) affected)

(1 row(s) affected)

Total execution time: 00:00:00.014

☰ ☹ ☹ ☹ ☹ ☹

	PenaltyID	Fee	CustomerID	DueDate	PenaltyStatus
1	1	30.00	3	2022-10-18	PENDING

☰ ☹ ☹ ☹ ☹ ☹

	CustomerID	ElectronicID	StartDate	DueDate	Amount	LoanStatus
1	1	3	2022-12-08	2023-01-07	200.00	PENDING
2	2	6	2022-10-08	2022-11-07	DONE	
3	3	2	2022-09-18	2022-10-18	175.00	FINED

☰ ☹ ☹ ☹ ☹ ☹

	CustomerID	ElectronicID
1	1	2

## Queries and Updates essential for the business use cases

SQL Script: [Procedures\\_and\\_examples.sql](#)

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

The screenshot shows a SQL query execution interface. The query is as follows:

```
1 # Hey I want to know what electronics are available and their operating systems
2 SELECT GROUP_CONCAT(OperatingSystem.OperatingSystemID SEPARATOR ', ') AS OperatingSystemIds,
3        GROUP_CONCAT(OSName SEPARATOR ', ') AS OperatingSystemNames,
4        OSInstalled.ElectronicID,
5        Electronic.ElectronicName ElectronicName
6 FROM OSInstalled
7 JOIN OperatingSystem on OSInstalled.OperatingSystemID = OperatingSystem.OperatingSystemID
8 JOIN Electronic on OSInstalled.ElectronicID = Electronic.ElectronicID
9 GROUP BY OSInstalled.ElectronicID;
```

(1303 row(s) affected)

Total execution time: 00:00:00.035

	OperatingSystemIds	OperatingSystemNames	ElectronicID	ElectronicName
1	3	IOS	1	MacBook Pro
2	3	IOS	2	Macbook Air
3	4	NO OS	3	250 G6
4	3	IOS	4	MacBook Pro
5	3	IOS	5	MacBook Pro
6	1	WINDOWS	6	Aspire 3
7	3	IOS	7	MacBook Pro
8	3	IOS	8	Macbook Air
9	1	WINDOWS	9	ZenBook UX430UN
10	1	WINDOWS	10	Swift 3
11	4	NO OS	11	250 G6
12	4	NO OS	12	250 G6
13	3	IOS	13	MacBook Pro
14	1	WINDOWS	14	Inspiron 3567
15	3	IOS	15	MacBook 12"
16	3	IOS	16	MacBook Pro
17	1	WINDOWS	17	Inspiron 3567
18	3	IOS	18	MacBook Pro
19	4	NO OS	19	IdeaPad 320-15IKB
20	1	WINDOWS	20	XPS 13

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

The screenshot shows a SQL query execution interface. The query is as follows:

```
1 SELECT * FROM TotalPenaltyView LIMIT 10;
```

(1 row(s) affected)

Total execution time: 00:00:00.003

	CustomerID	CustomerName	Total
1	3	Odium Shard	30.00

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

SQL

```

1  # To see how many devices have particular Operating system installed
2  SELECT OperatingSystem.OperatingSystemID, OperatingSystem.OSName,
3         COUNT(*) AS NumberOfDevices
4  FROM   OperatingSystem
5  JOIN   OSInstalled ON OperatingSystem.OperatingSystemID = OSInstalled.OperatingSystemID
6  GROUP BY OperatingSystem.OperatingSystemID
7  ORDER BY NumberOfDevices DESC;

```

(4 row(s) affected)

Total execution time: 00:00:00.002

	OperatingSystemID	OSName	NumberOfDevices
1	1	WINDOWS	1072
2	4	NO OS	148
3	2	LINUX	62
4	3	IOS	21

- To check all customers active loans and pending loan details

SQL

```

1  # Check loan status for each customer
2
3  SELECT Customer.CustomerID,
4         CONCAT(Customer.FirstName, ' ', Customer.LastName) AS FullName,
5         SUM(LoanStatus LIKE 'PENDING') AS ActiveLoans,
6         SUM(LoanStatus LIKE 'DONE') AS Returned,
7         SUM(LoanStatus LIKE 'FINED') AS Fined,
8         COUNT(*) AS Total
9  FROM   Customer
10 JOIN   Loans ON Customer.CustomerID = Loans.CustomerID
11 GROUP BY CustomerID;

```

(3 row(s) affected)

Total execution time: 00:00:00.002

	CustomerID	FullName	ActiveLoans	Returned	Fined	Total
1	1	Kaladin Stormblessed	1	0	0	1
2	2	Dalinar Kholin	0	1	0	1
3	3	Odium Shard	0	0	1	1

## Other Procedures and Events

- Procedure for loaning a device to a customer.

```
1 DROP PROCEDURE IF EXISTS LoanDevice;
2
3 CREATE PROCEDURE LoanDevice(IN vCustomerID INTEGER, IN vElectronicID INTEGER)
4 BEGIN
5     INSERT Loans(CustomerID, ElectronicID, StartDate, DueDate, Amount, LoanStatus)
6     VALUES (vCustomerID, vElectronicID, NOW(), ADDDATE(CURDATE(), INTERVAL 30 DAY), 150.00, 'PENDING');
7 END ;
8
9 # Testing the procedure:
10
11 # BEFORE:
12 SELECT E.ElectronicName, L.StartDate, L.DueDate, L.LoanStatus
13 FROM Electronic E
14 NATURAL JOIN Loans L
15 WHERE L.CustomerID = 2;
16
17 # CALLING THE PROCEDURE:
18 CALL LoanDevice(2, 1);
19
20 # AFTER:
21 SELECT E.ElectronicName, L.StartDate, L.DueDate, L.LoanStatus
22 FROM Electronic E
23 NATURAL JOIN Loans L
24 WHERE L.CustomerID = 2;
25
```

Commands completed successfully

Commands completed successfully

(1 row(s) affected)

Commands completed successfully

(2 row(s) affected)

Total execution time: 00:00:00.380

	ElectronicName	StartDate	DueDate	LoanStatus
1	Aspire 3	2022-10-08	2022-11-07	DONE

	ElectronicName	StartDate	DueDate	LoanStatus
1	MacBook Pro	2022-12-22	2023-01-21	PENDING
2	Aspire 3	2022-10-08	2022-11-07	DONE

SQL



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

```

1  # When this procedure is called it looks into all the loans and their due dates. If any loan is past due date
2  # it creates a penalty
3
4  DROP PROCEDURE IF EXISTS CreatePenalty;
5
6  CREATE PROCEDURE CreatePenalty()
7  BEGIN
8      START TRANSACTION;
9      INSERT INTO Penalty (CustomerID, Fee, DueDate, PenaltyStatus)
10         SELECT CustomerID, 100.00, CURDATE(), 'PENDING' FROM Loans L
11         WHERE L.LoanStatus = 'PENDING' AND DATEDIFF(L.DueDate, CURDATE()) < 0;
12
13         UPDATE Loans L SET LoanStatus = 'FINED'
14         WHERE L.LoanStatus = 'PENDING' AND DATEDIFF(L.DueDate, CURDATE()) < 0;
15     COMMIT;
16 END;
17
18 # Testing the procedure with transaction:
19
20 # BEFORE:
21 SELECT E.ElectronicName, L.StartDate, L.DueDate, L.LoanStatus
22 FROM Electronic E
23 NATURAL JOIN Loans L
24 WHERE L.CustomerID = 1;
25
26 SELECT * FROM Penalty P WHERE P.CustomerID = 1;
27
28 # CREATE TEST CONDITIONS:
29 UPDATE Loans L SET L.StartDate = ADDDATE(CURDATE(), -40)
30 WHERE L.CustomerID = 1 AND L.ElectronicID = 3;
31
32 UPDATE Loans L SET L.DueDate = ADDDATE(CURDATE(), -10)
33 WHERE L.CustomerID = 1 AND L.ElectronicID = 3;
34
35 # CALLING THE PROCEDURE:
36 CALL CreatePenalty();
37
38 # AFTER:
39 SELECT E.ElectronicName, L.StartDate, L.DueDate, L.LoanStatus
40 FROM Electronic E
41 NATURAL JOIN Loans L
42 WHERE L.CustomerID = 1;
43
44 SELECT * FROM Penalty P WHERE P.CustomerID = 1;

```

Before penalty:



	ElectronicName	StartDate	DueDate	LoanStatus
1	250 G6	2022-12-08	2023-01-07	PENDING



	PenaltyID	Fee	CustomerID	DueDate	PenaltyStatus
--	-----------	-----	------------	---------	---------------

After penalty:



	ElectronicName	StartDate	DueDate	LoanStatus
1	250 G6	2022-11-12	2022-12-12	FINED



	PenaltyID	Fee	CustomerID	DueDate	PenaltyStatus
1	2	100.00	1	2022-12-22	PENDING

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

```

1  # Cases when customer has pending loans and request for new device for loan or device out of stock
2
3  DROP TRIGGER IF EXISTS Check_loans_before_granting;
4
5  CREATE TRIGGER Check_loans_before_granting
6  BEFORE INSERT ON Loans FOR EACH ROW
7  BEGIN
8      DECLARE LoanedCount, TotalCount, ActiveLoansOfDevice INTEGER DEFAULT 0;
9      SELECT COUNT(*) INTO LoanedCount FROM Loans L
10     WHERE L.ElectronicID = NEW.ElectronicID AND L.LoanStatus != 'DONE';
11     GROUP BY ElectronicID;
12     SELECT E.TotalCount INTO TotalCount FROM Electronic E WHERE E.ElectronicID = NEW.ElectronicID;
13     IF (LoanedCount >= TotalCount)
14     THEN SIGNAL SQLSTATE '45000'
15     SET MESSAGE_TEXT = 'Device out of stock';
16     END IF;
17     SELECT COUNT(*) INTO ActiveLoansOfDevice FROM Loans L
18     WHERE L.ElectronicID = NEW.ElectronicID AND L.CustomerID = NEW.CustomerID AND L.LoanStatus != 'DONE';
19     GROUP BY L.ElectronicID;
20     IF (ActiveLoansOfDevice != 0)
21     THEN SIGNAL SQLSTATE '45000'
22     SET MESSAGE_TEXT = 'Customer already loaned device';
23     END IF;
24 END;

```

## Test

```

1  CALL LoanDevice(2, 1);

```

(1644, 'Customer already loaned device')

Total execution time: 00:00:00.004

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

```

1  CALL LoanDevice(1, 5);

```

(1644, 'Device out of stock')

Total execution time: 00:00:00.002

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

```

1  # Schedule create penalties at end of each DAY
2
3  SET GLOBAL event_scheduler = 1;
4  DROP EVENT IF EXISTS CreateFinesEvent;
5
6  CREATE EVENT CreateFinesEvent ON SCHEDULE EVERY 1 DAY
7  DO
8  BEGIN
9      CALL CreatePenalty();
10 END;

```

- Some updates on tables

+

↓

↑

☐

☒

...

▶

1

2

3

4

5

6

7

8

9

10

```

SET SQL_SAFE_UPDATES = 0;

# BEFORE:
SELECT * FROM Electronic Limit 5;

UPDATE Electronic SET TotalCount = 25
WHERE ElectronicID = 4;

# AFTER:
SELECT * FROM Electronic Limit 5;

```

SQL

Commands completed successfully

(5 row(s) affected)

Commands completed successfully

(5 row(s) affected)

Total execution time: 00:00:00.110

📄

🔍

🔗

📄

🔗

📄

	ElectronicID	ElectronicName	ReleaseYear	Count	TotalCount	Rating	ManufacturerID	TypeID
1	1	MacBook Pro	2020	15	15	AVERAGE	1	1
2	2	Macbook Air	2021	8	8	AVERAGE	1	1
3	3	250 G6	2022	13	13	AVERAGE	2	1
4	4	MacBook Pro	2021	5	5	AVERAGE	1	1
5	5	MacBook Pro	2019	5	0	AVERAGE	1	1

📄

🔍

🔗

📄

🔗

📄

	ElectronicID	ElectronicName	ReleaseYear	Count	TotalCount	Rating	ManufacturerID	TypeID
1	1	MacBook Pro	2020	15	15	AVERAGE	1	1
2	2	Macbook Air	2021	8	8	AVERAGE	1	1
3	3	250 G6	2022	13	13	AVERAGE	2	1
4	4	MacBook Pro	2021	5	25	AVERAGE	1	1
5	5	MacBook Pro	2019	5	0	AVERAGE	1	1

## 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.

The screenshot displays the Azure Data Studio interface. The title bar indicates the active query is 'SQLQuery\_1 - dbcoursekj.mysql.database.azure.com.rentaldb (kj2056)'. The left sidebar shows the 'CONNECTIONS' pane with 'SERVERS' expanded, listing 'localhost, <default> (root)' and 'dbcoursekj.mysql.database.azure.com'. Under 'Databases', 'rentaldb' is selected, showing a list of tables including 'customer', 'electronic', 'electronicstype', 'loans', 'manufacturer', 'notifies', 'operatingsystem', 'osinstalled', 'penalty', 'Views', 'Stored Procedures', 'Functions', 'Events', 'System Databases', 'Users', and 'Tablespaces'.

The central editor pane shows a SQL query:

```
1 SELECT *
2 FROM customer
3 LIMIT 1000;
```

The 'Results' pane at the bottom displays the output of the query as a table with 6 columns: CustomerID, FirstName, LastName, Contact, and CustomerStatus. The results show 3 rows of data.

	CustomerID	FirstName	LastName	Contact	CustomerStatus
1	1	Kaladin	Stormblessed	3471234567	GOOD
2	2	Dalinar	Kholin	3471234568	GOOD
3	3	Odium	Shard	3470000000	BAD

The bottom status bar shows 'Ln 1, Col 1 Spaces: 4 UTF-8 LF 3 rows MySQL 00:00:00 dbcoursekj.mysql.database.azure.com : rentaldb'.

## 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.
  - a. RUN ``python laptopdataprocessing.py``
  - b. to generate the required data in csv format.
2. MySQL server running
3. Run ``Databaselnitiation.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
  - a. Find out the secure file directory by ``SHOW VARIABLES LIKE "secure_file_priv"``
  - b. Move the csv files created in step 1 to the above directory
  - c. 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](#)

Azure Data [Studio](#)