**D**

**Group 1**

**CIS 9340 UTA 28747: Principles of Database Management Systems Fall 2022**

**Electronics Rental Company**

Sierra Levy - sierra.levy@baruchmail.cuny.edu

Johnny Huang - johnny.huang1@baruchmail.cuny.edu

Erica Kane - erica.kane@baruchmail.cuny.edu

Letty Uy - manilette.uy@baruchmail.cuny.edu

**Table of Contents**

[**Executive Summary 1**](#_heading=h.20lohvjhlwov)

[**Business Scenario 2**](#_heading=h.pupt7j7i13l)

[Information to be tracked 3](#_heading=h.nfa8i6wa4od5)

[Initial List of Entities Identified 3](#_heading=h.85m32yp7h40)

[**ER Model using UML Notation 5**](#_heading=h.gqylbkposbym)

[Relationship Sentences 5](#_heading=h.stmlg3xb667b)

[**Converting the ERD to a Relational Model 6**](#_heading=h.w74swuaaafc1)

[**Normalization 6**](#_heading=h.bitdsi4rm5kv)

[**Creating Tables with SQL 16**](#_heading=h.2nuxedd4orj1)

[**Scenarios 31**](#_heading=h.wn0zv2plibgj)

# 

# 

# 

# Executive Summary

The following is a database management systems project that attempts to replicate a real-world business setting and the business’ attempt to create a database system. As a group, we brainstormed potential businesses, ultimately deciding on an electronics rental company to base our project on. With a proposal written and approved, we entered the systems analysis phase. In this phase, we gathered the requirements of the users of the electronics rental company. We created an entity-relationship diagram (ERD) using UML notation to capture said requirements. With that completed came the logical and physical modeling phase. In this phase, we converted the ERD to a relational data model (RDM), and put the RDM through the normalization process to eliminate redundancy and inconsistent dependencies. Following this is the database implementation phase where we used SQL to implement the normalized set of relations to create a working database that should represent the needs of our electronics rental company.

# **Business Scenario**

When the pandemic hit and most people were forced to work from home, the demand for electronics rose. Many students and at-home workers required better remote set-ups, but the cost to buy new quality tech equipment was high and many people found they couldn’t afford new equipment. That’s when Electronic Rental Company came to the rescue and created an electronic rental system that allowed customers to rent products when needed. Customers rent equipment and return when they are done at a fraction of the price. The company started out small, but as interest and inventory grew, keeping up with demand became harder. Workers struggled with keeping track of products, receiving returns in a timely manner, and other issues that made it difficult to keep the business moving smoothly. The owner of the company suggested a new database management system to better organize and track inventory in hopes of growing a more successful business.

The main problems faced were keeping track of inventory, following up with customers, returning shipments to the correct warehouses, and keeping track of rental/lease lengths. To remedy these issues, some initial information that would help solve the problem would be to track customer information, rental order number, rental date, return date, stock info (stock IDs), inventory, tracking numbers, shipment details, condition of product for replacement, subscription type (corporate, individual, or student), purchase orders (to record and track merchandise purchases made from various vendors), and lease length.

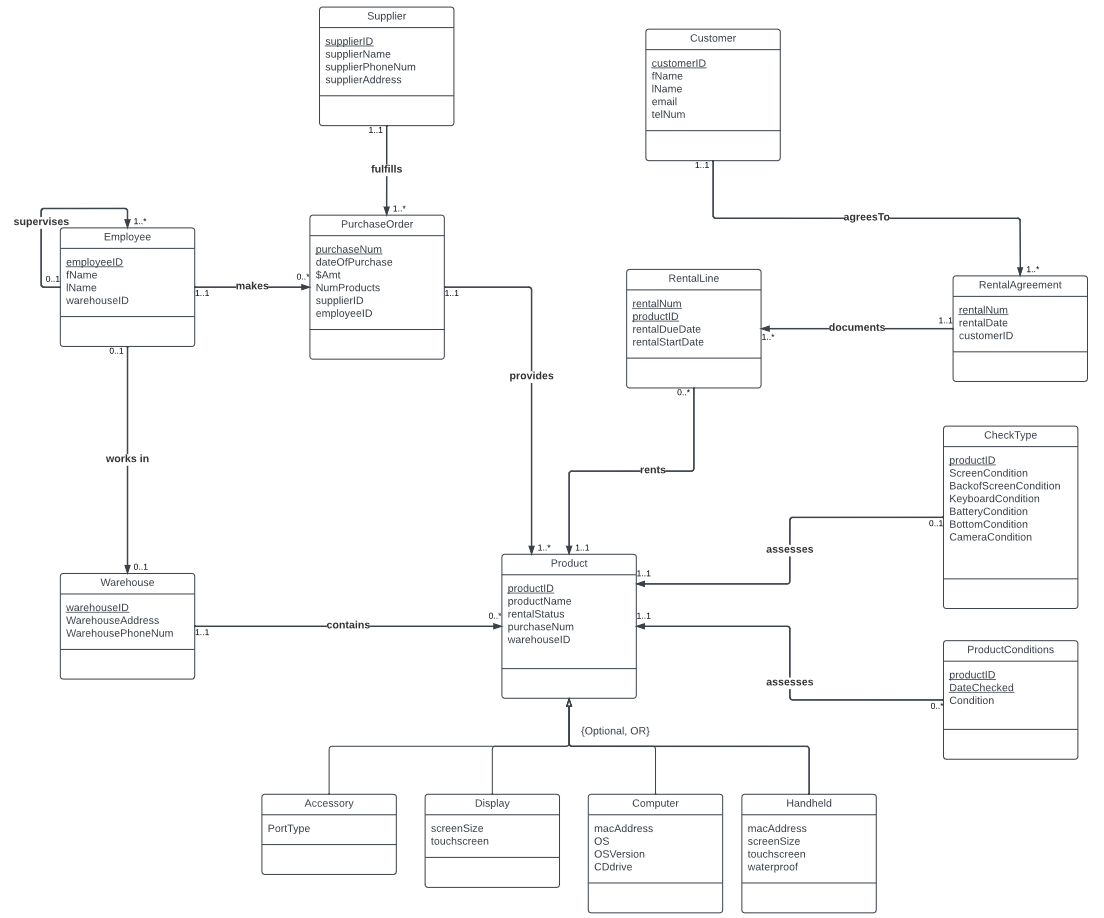
## **Information to be tracked**

The main problems faced were keeping track of inventory, following up with customers, returning shipments to the correct warehouses, and keeping track of rental/lease lengths. To remedy these issues, some initial information that would help solve the problem would be to track customer information, rental order number, rental date, return date, stock info (stock IDs), inventory, tracking numbers, shipment details, condition of product for replacement, subscription type (corporate, individual, or student), purchase orders (to record and track merchandise purchases made from various vendors), and lease length

## **Initial List of Entities Identified**

* Employees
* Customers
* Equipments/Inventory
* Rental Information
* Product Shipment Information

# **ER Model using UML Notation**



## **Relationship Sentences**

1. One **supplier** fulfills many **purchase orders**.
2. One **purchase order** is fulfilled by one **supplier**.
3. One **supervisor** supervises many **employees**.
4. One **employee** is supervised by one **supervisor**.
5. One **employee** makes many **purchase orders**.
6. One **purchase order** is made by one **employee**.
7. One **purchase order** provides many **products**.
8. One **product** is provided by one **purchase order**.
9. One **employee** works in one **warehouse**.
10. One **warehouse** is worked by one **employee**.
11. One **customer** agrees to many **rental agreements**.
12. One **rental agreement** is agreed to by one **customer**.
13. One **rental agreement** documents many **rental lines**.
14. One **rental line** is documented by one **rental agreement**.
15. One **rental line** rents one **product**.
16. One **product** is rented to many **rental lines**.
17. One **check type** assesses one **product**.
18. One **product** is assessed by one **check type**.
19. One **product** condition assesses one **product**.
20. One **product** is assessed by many **product conditions**.

# **Converting the ERD to a Relational Model**

* Employee (employeeID, fName, lName, warehouseID)
* Supervisor (employeeID, fName, lName, warehouseID, supervisorID)
* Supplier (supplierID, supplierName, supplierPhoneNum, supplierAddress)
* PurchaseOrders (purchaseNum, dateOfPurchase, dollarAmount, NumProducts, supplierID, employeeID)
* Warehouse (warehouseID, warehousePhone, WarehouseAddress)
* Product (productID, productName, rentalStatus, purchaseNum, warehouseID)
* Display (productID, screenSize, touchScreen)
* Computer (productID, macAddress, OS, OSVersion, CDdrive)
* Handheld (productID, macAddress, screenSize, touchScreen, waterProof)
* Accessory (productID, portType)
* RentalLine (rentalNum, productID, rentalDueDate, rentalStartDate)
* RentalAgreement (rentalNum, rentalDate, customerID, productID)
* Customer (customerID, cusfName, cuslName, email, telNum)
* CheckType (productID, screenCondition, backOfScreenCondition, keyboardCondition, batteryCondition, bottomCondition, cameraCondition)
* ProductConditions (productID, dateChecked, condition)

# **Normalization**

**Employee Relation**

Employee (employeeID, fName, lName, warehouseID)

**Step 1: identify keys**

* employeeID

**Step 2: functional dependencies**

* FD1: employeeID → fName, lName, warehouseID

**Step 3: do we have keys?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, in BCNF

**Final Relations:**

* R1 (employeeID, fName, lName, warehouseID)

**Supervisor Relation**

Supervisor (employeeID, fName, lName, warehouseID, supervisorID)

**Step 1: identify keys**

* employeeID, supervisorID

**Step 2: functional dependencies**

* employeeID → supervisorID, fName, lName, warehouseID

**Step 3: do we have keys?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, in BCNF

**Final Relations:**

* R1 (employeeID, supervisorID, fName, lName, warehouseID)

**Supplier Relation**

Supplier (supplierID, supplierName, supplierPhoneNum, supplierAddress)

**Step 1: identify keys**

* supplierID

**Step 2: functional dependencies**

* FD1: supplierID → supplierName, supplierPhoneNum, supplierAddress
* FD2: supplierName → supplierPhoneNum, supplierAddress

**Step 3: do we have keys?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* Yes, FD2: supplierName → supplierPhoneNum, supplierAddress
* R1 (supplierID, supplierName)
* R2 (supplierName, supplierPhoneNum, supplierAddress)

**Step 6: do we have non-key attribute determine part of key?**

* No, in BCNF

**Final Relations:**

* R1 (supplierID, supplierName)
* R2 (supplierName, supplierPhoneNum, supplierAddress)

**PurchaseOrders Relation**

PurchaseOrders (purchaseNum, dateOfPurchase, dollarAmount, NumProducts, supplierID, employeeID)

**Step 1: identify keys**

* purchaseNum

**Step 2: functional dependencies**

* FD1: purchaseNum → dateOfPurchase, dollarAmount, NumProducts, supplierID, employeeID

**Step 3: do we have keys?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, in BCNF

**Final Relations:**

* R1 (purchaseNum, dateOfPurchase, dollarAmount, NumProducts, supplierID, employeeID)

**Warehouse Relation**

Warehouse (warehouseID, WarehouseAddress, WarehousePhoneNum)

**Step 1: identify keys**

* warehouseID

**Step 2: functional dependencies**

* warehouseID → WarehouseAddress, WarehousePhoneNum

**Step 3: do we have keys?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, in BCNF

**Final Relations:**

* R1 (warehouseID, WarehouseAddress, WarehousePhoneNum)

**Product Relation**

Product (productID, productName, rentalStatus, purchaseNum, warehouseID)

**Step 1: identify keys**

* productID

**Step 2: functional dependencies**

* FD1: productID → productName, rentalStatus, purchaseNum, warehouseID

**Step 3: do we have keys?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No transitive dependencies in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, in BCNF

**Final Relations:**

* R1 (productID, productName, rentalStatus, purchaseNum, warehouseID)

**Display Relation**

Display (productID, screenSize, touchScreen)

**Step 1: identify keys**

* productID

**Step 2: functional dependencies**

* productID → screenSize, touchScreen

**Step 3: do we have keys?**

* Yes, it’s in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, it’s in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, it’s in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, it’s in BCNF

**Final Relations:**

* R1 (productID, screenSize, touchScreen)

**Computer Relation**

Computer (productID, macAddress, OS, OSVersion, CDdrive)

**Step 1: identify keys**

* productID

**Step 2: functional dependencies**

* FD1: productID → macAddress, OS, OSVersion, CDdrive)

**Step 3: do we have keys?**

* Yes, it’s in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, it’s in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, it’s in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, it’s in BCNF

**Final Relations:**

* R1 (productID, macAddress, OS, OSVersion, CDdrive)

**Handheld Relation**

Handheld (productID, macAddress, screenSize, touchScreen, waterProof)

**Step 1: identify keys**

* productID

**Step 2: functional dependencies**

* productID → macAddress, screenSize, touchScreen, waterProof

**Step 3: do we have keys?**

* Yes, it’s in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, it’s in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, it’s in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, it’s in BCNF

**Final Relations:**

* R1 (productID, macAddress, screenSize, touchScreen, waterProof)

**Accessory Relation**

Accessory (productID, portType)

**Step 1: identify keys**

* productID

**Step 2: functional dependencies**

* productID → portType

**Step 3: do we have keys?**

* Yes, it’s in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, it’s in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, it’s in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, it’s in BCNF

**Final Relations:**

* R1 (productID, portType)

**RentalLine Relation**

RentalLine (rentalNum, productID, rentalDueDate, rentalStartDate)

**Step 1: identify keys**

* rentalNum
* productID

**Step 2: functional dependencies**

* FD1: rentalNum, productID → rentalDueDate, rentalStartDate
* FD2: rentalNum → rentalDueDate, rentalStartDate

**Step 3: do we have key(s)?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* Yes, FD2: rentalNum → rentalDueDate, rentalStartDate
* R1 (rentalNum, rentalDueDate, rentalStartDate)
* R2 (rentalNum, productID)

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No

**Step 6: do we have non-key attribute determine part of key?**

* No, in BCNF

**Final Relations**:

* R1 (rentalNum, rentalDueDate, rentalStartDate)
* R2 (rentalNum, productID)

**RentalAgreement Relation**

RentalAgreement (rentalNum, rentalDate, customerID, productID)

**Step 1: identify keys**

* rentalNum

**Step 2: functional dependencies**

* rentalNum → rentalDate, customerID, productID

**Step 3: do we have keys?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, it’s in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, it’s in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, it’s in BCNF

**Final Relations:**

* R1: (rentalNum, rentalDate, customerID, productID)

**Customer Relation**

Customer (customerID, cusfName, cuslName, email, telNum)

**Step 1: identify keys**

* customerID

**Step 2: functional dependencies**

* customerID → cusfName, cuslName, email, telNum

**Step 3: do we have keys?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, it’s in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, it’s in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, it’s in BCNF

**Final Relations:**

* R1 (customerID, cusfName, cuslName, email, telNum)

**CheckType Relation**

CheckType (productID, screenCondition, backOfScreenCondition, keyboardCondition, batteryCondition, bottomCondition, cameraCondition)

**Step 1: identify keys**

* productID

**Step 2: functional dependencies**

* productID → screenCondition, backOfScreenCondition, keyboardCondition, batteryCondition, bottomCondition, cameraCondition)

**Step 3: do we have keys?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, it’s in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, it’s in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, it’s in BCNF

**Final Relations:**

* R1 (productID, screenCondition, backOfScreenCondition, keyboardCondition, batteryCondition, bottomCondition, cameraCondition)

**ProductConditions Relation**

ProductConditions (productID, dateChecked, condition)

**Step 1: identify keys**

* productID

**Step 2: functional dependencies**

* productID → dateChecked, condition

**Step 3: do we have keys?**

* Yes, in 1NF

**Step 4: do we have partial key dependencies? (part of key determines non-key attributes)**

* No, it’s in 2NF

**Step 5: do we have transitive dependencies? (Non-Key attribute(s) determines non-key attribute(s))**

* No, it’s in 3NF

**Step 6: do we have non-key attribute determine part of key?**

* No, it’s in BCNF

**Final Relations:**

* R1 (productID, dateChecked, condition)

# **Creating Tables with SQL**

CREATE TABLE warehouse (

warehouseID NUMBER NOT NULL,

warehouseStreet varchar(250),

warehouseCity varchar(250),

warehouseState varchar(2),

warehouseZip varchar(5),

warehousePhoneNum varchar(50)

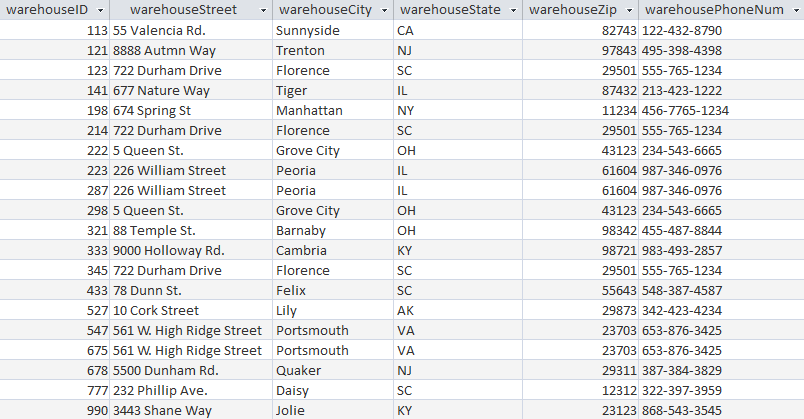
);

ALTER TABLE warehouse

ADD PRIMARY KEY (warehouseID);

INSERT INTO warehouse (warehouseID, warehouseAddress, warehousePhoneNum)

VALUES (214, ‘722 Durham Drive’, ‘ Florence’, ‘SC’, ‘ 29501’, ‘555-765-1234’) ;



CREATE TABLE employee (

employeeID NUMBER NOT NULL,

fName varchar(250),

lName varchar(250),

warehouseID NUMBER

);

ALTER TABLE employee

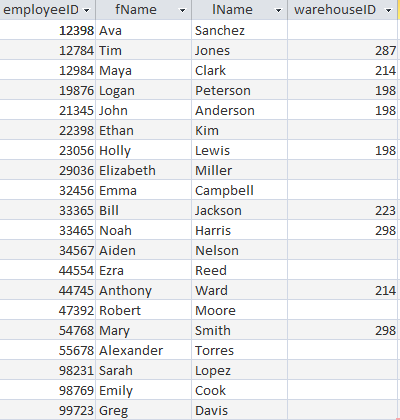
ADD PRIMARY KEY (employeeID);

ALTER TABLE employee

ADD FOREIGN KEY (warehouseID) REFERENCES warehouse(warehouseID);

INSERT INTO employee (employeeID, fName, lName, warehouseID)

VALUES (54768, ‘Mary’, ‘Smith’, 298) ;



CREATE TABLE supervisor (

employeeID NUMBER NOT NULL,

supervisorID NUMBER NOT NULL,

fName varchar(250),

lName varchar(250),

warehouseID NUMBER

);

ALTER TABLE supervisor

ADD PRIMARY KEY (employeeID);

ALTER TABLE supervisor

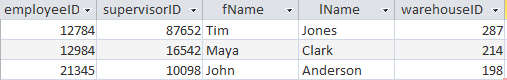
ADD FOREIGN KEY (warehouseID) REFERENCES warehouse(warehouseID);

ALTER TABLE supervisor

ADD FOREIGN KEY (employeeID) REFERENCES employee(employeeID);

INSERT INTO supervisor (employeeID, supervisor ID, fName, lName, warehouseID)

VALUES (21345, 10098, ‘John’, ‘Anderson’, 198) ;



CREATE TABLE customer (

customerID NUMBER NOT NULL,

cusFName varchar(250),

cusLName varchar(250),

email varchar(250),

telNum varchar(50)

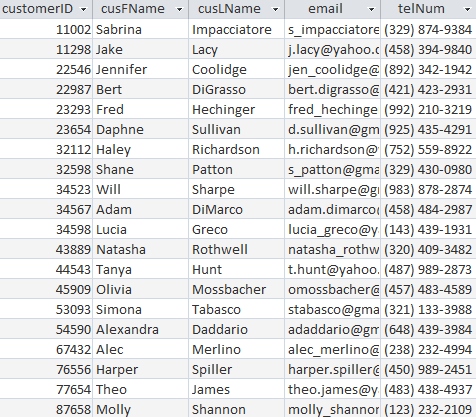
);

ALTER TABLE customer

ADD PRIMARY KEY (customerID);

INSERT INTO customer (customerID, cusFName, cusLName, email, telNum)

VALUES (22987, ‘Bert’, ‘DiGrasso’, ‘bert.digrasso@gmail.com’, ‘421-423-2931’) ;



CREATE TABLE supplierInfo (

supplierName varchar(250) NOT NULL,

supplierPhoneNum varchar(50),

supplierStreet varchar(250),

supplierCity varchar(250),

supplierState varchar(2),

supplierZip varchar(5)

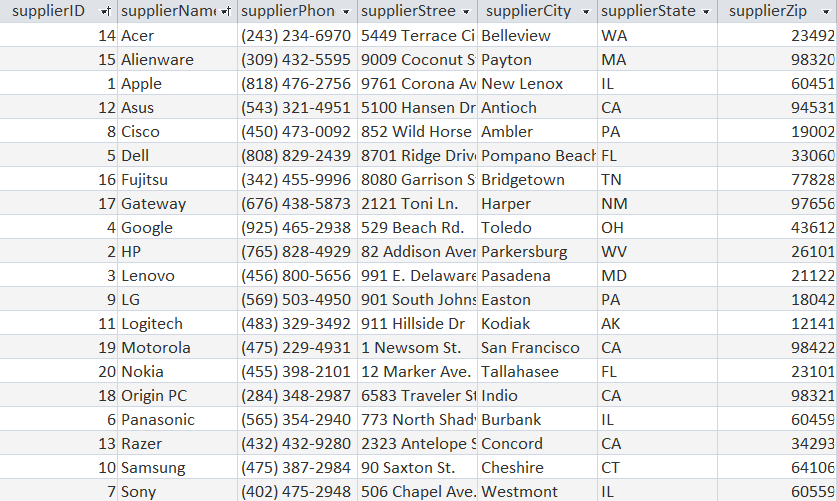
);

ALTER TABLE supplierInfo

ADD PRIMARY KEY (supplierName);

INSERT INTO supplierInfo (supplierName, supplierPhoneNum, supplierAddress)

VALUES (‘Apple’, ‘(818) 476-2756’, ‘9761 Corona Ave.’, ‘New Lenox’, ‘IL’, ‘60451’) ;



CREATE TABLE supplier (

supplierID NUMBER NOT NULL,

supplierName varchar(250)

);

ALTER TABLE supplier

ADD PRIMARY KEY (supplierID) ;

ALTER TABLE supplier

ADD FOREIGN KEY (supplierName) REFERENCES supplierInfo(supplierName) ;

INSERT INTO supplier (supplierID, supplierName)

VALUES (1, ‘Apple’) ;



CREATE TABLE purchaseOrders (

purchaseNum NUMBER NOT NULL,

dateOfPurchase DATE,

dollarAmount NUMBER,

numProducts NUMBER,

supplierID NUMBER,

employeeID NUMBER

);

ALTER TABLE purchaseOrders

ADD PRIMARY KEY (purchaseNum);

ALTER TABLE purchaseOrders

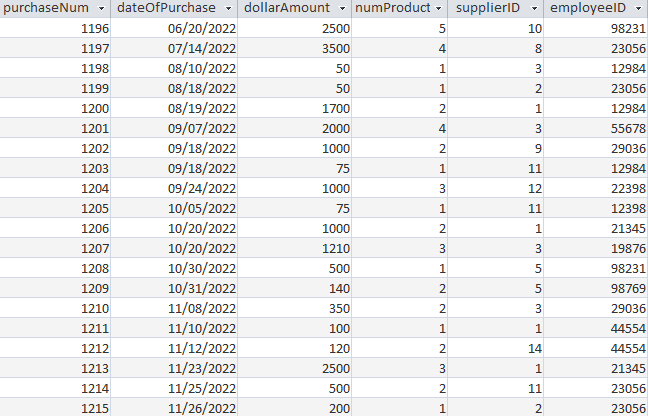
ADD FOREIGN KEY (employeeID) REFERENCES employee(employeeID);

ALTER TABLE purchaseOrders

ADD FOREIGN KEY (supplierID) REFERENCES supplier(supplierID);

INSERT INTO purchaseOrders (purchaseNum, dateOfPurchase, $amt, numProducts, supplierID, employeeID)

VALUES (1213, “11/23/2022”, 2500, 3, 1, 21345) ;



CREATE TABLE product (

productID NUMBER NOT NULL,

productName varchar(250),

rentalStatus varchar(250),

purchaseNum NUMBER,

warehouseID NUMBER

);

ALTER TABLE product

ADD PRIMARY KEY (productID);

ALTER TABLE product

ADD FOREIGN KEY (warehouseID) REFERENCES warehouse(warehouseID);

ALTER TABLE product

ADD FOREIGN KEY (purchaseNum) REFERENCES purchaseOrders(purchaseNum);

INSERT INTO product (productID, productName, rentalStatus, purchaseNum, warehouseID)

VALUES (111, ‘MacBook Pro’, ‘Rented’, 1190, 123) ;



CREATE TABLE display (

productID NUMBER NOT NULL,

screenSize varchar(250),

touchScreen varchar(250)

);

ALTER TABLE display

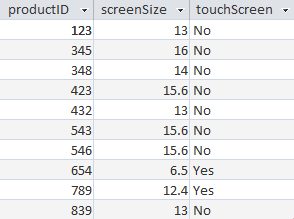
ADD PRIMARY KEY (productID);

ALTER TABLE display

ADD FOREIGN KEY (productID) REFERENCES product(productID) ;

INSERT INTO display (productID, screenSize, touchScreen)

VALUES (839, 13, ‘No’) ;



CREATE TABLE computer (

productID NUMBER NOT NULL,

macAddress varchar(250),

OS varchar(250),

OSVersion varchar(250),

CDdrive varchar(250)

);

ALTER TABLE computer

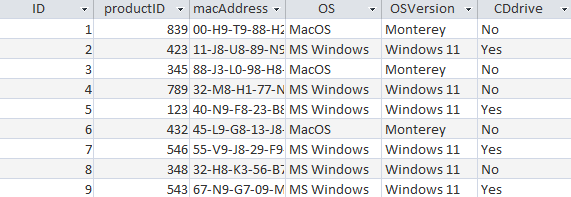
ADD PRIMARY KEY (productID);

ALTER TABLE computer

ADD FOREIGN KEY (productID) REFERENCES product(productID) ;

INSERT INTO computer (productID, macAddress, OS, OSVersion, CDdrive)

VALUES (839, ‘00-H9-T9-88-H2-89’, ‘MacOS’, ‘Monterey’, ‘No’) ;



CREATE TABLE handheld (

productID NUMBER NOT NULL,

macAddress varchar(250),

screenSize varchar(250),

touchScreen varchar(250),

waterProof varchar(250)

);

ALTER TABLE handheld

ADD PRIMARY KEY (productID);

ALTER TABLE handheld

ADD FOREIGN KEY (productID) REFERENCES product(productID) ;

INSERT INTO handheld (productID, macAddress, screenSize, touchScreen, waterProof)

VALUES (654, ‘88-Y3-E6-45-E1-J9-89’, 6.5, ‘Yes’, ‘Yes’) ;



CREATE TABLE accessory (

productID NUMBER NOT NULL,

portType varchar(250)

);

ALTER TABLE accessory

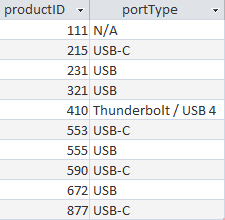
ADD PRIMARY KEY (productID);

ALTER TABLE accessory

ADD FOREIGN KEY (productID) REFERENCES product(productID) ;

INSERT INTO accessory (productID, portType)

VALUES (672, ‘USB’) ;



CREATE TABLE checkType (

productID NUMBER NOT NULL,

screenCondition varchar(250),

backOfScreenCondition varchar(250),

keyboardCondition varchar(250),

batteryCondition varchar(250),

bottomCondition varchar(250),

cameraCondition varchar(250)

);

ALTER TABLE checkType

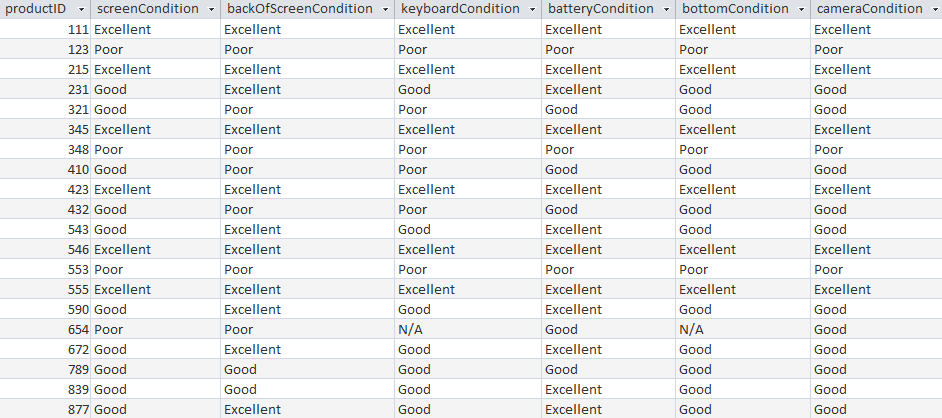
ADD PRIMARY KEY (productID);

ALTER TABLE checkType

ADD FOREIGN KEY (productID) REFERENCES product(productID) ;

INSERT INTO checkType (productID, screenCondition, backOfScreenCondition, keyboardCondition, batteryCondition, bottomCondition, cameraCondition)

VALUES (839, ‘Good’, ‘Good’, ‘Good’, ‘Excellent’, ‘Good’, ‘Good’) ;



CREATE TABLE productConditions (

productID NUMBER NOT NULL,

dateChecked DATE,

condition varchar(250)

);

ALTER TABLE productConditions

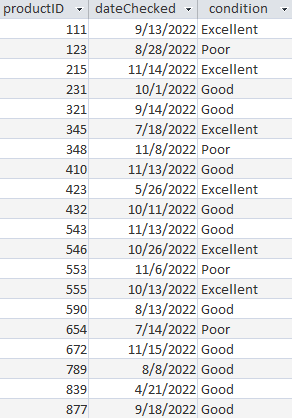
ADD PRIMARY KEY (productID);

ALTER TABLE productConditions

ADD FOREIGN KEY (productID) REFERENCES product(productID) ;

INSERT INTO productConditions (productID, dateChecked, condition)

VALUES (839, ‘4/21/2022’, ‘Good’);



CREATE TABLE rentalAgreement (

rentalNum NUMBER NOT NULL,

rentalDate DATE,

customerID NUMBER,

productID NUMBER

);

ALTER TABLE rentalAgreement

ADD PRIMARY KEY (rentalNum);

ALTER TABLE rentalAgreement

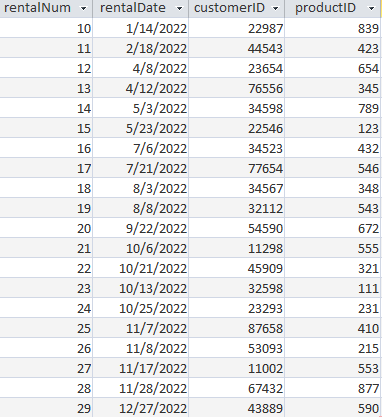
ADD FOREIGN KEY (productID) REFERENCES product(productID);

ALTER TABLE rentalAgreement

ADD FOREIGN KEY (customerID) REFERENCES customer(customerID);

INSERT INTO rentalAgreement (rentalNum, rentalDate, customerID, productID)

VALUES (10, ‘1/14/2022’, 22987, 839) ;



CREATE TABLE rentalLine (

rentalNum NUMBER NOT NULL,

rentalDueDate DATE,

rentalStartDate DATE,

rentalReturnDate DATE

);

ALTER TABLE rentalLine

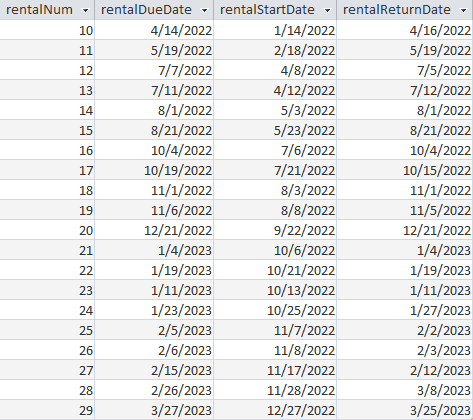
ADD PRIMARY KEY (rentalNum);

ALTER TABLE rentalLine

ADD FOREIGN KEY (rentalNum) REFERENCES rentalAgreement(rentalNum);

INSERT INTO rentalLine (rentalNum, rentalDueDate,rentalStartDate, rentalReturnDate)

VALUES (10, “4/14/2022”, “1/14/2022”, “04/16/2022”) ;



CREATE TABLE rentalLineProduct (

rentalNum NUMBER NOT NULL,

productID NUMBER NOT NULL

);

ALTER TABLE rentalLineProduct

ADD PRIMARY KEY (rentalNum, productID);

ALTER TABLE rentalLineProduct

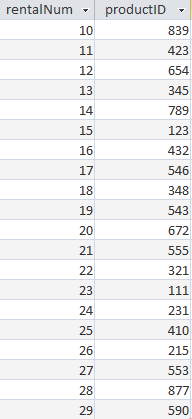
ADD FOREIGN KEY (productID) REFERENCES product(productID);

ALTER TABLE rentalLine

ADD FOREIGN KEY (rentalNum) REFERENCES rentalAgreement(rentalNum);

INSERT INTO rentalLineProduct (rentalNum, productID)

VALUES (10,839);



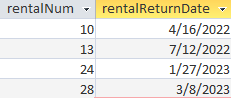
# **Scenarios**

1. Write a query for rental return dates that are past the due date.

SELECT rentalNum, rentalReturnDate

FROM rentalLine

WHERE rentalReturnDate > rentalDueDate ;



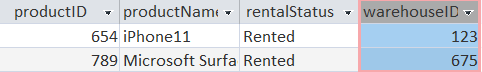
1. Write a query to display the product ID, product name, rental status, and warehouseID for any items that have touchscreen capability.

SELECT p.productID, p.productName, p.rentalStatus, p.warehouseID

FROM product p, display d

WHERE p.productID = d.productID

AND touchScreen = 'Yes' ;

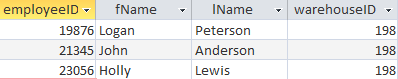


1. Write a query to display all employees, including supervisors, for the warehouse whose warehouseID is 198.

SELECT \*

FROM employee

WHERE warehouseID = 198 ;



1. Write a query to display the total number of purchases handled by employee ID 12984.

SELECT employeeID, COUNT(purchaseNum) AS TotalPurchaseOrders

FROM purchaseOrders

WHERE employeeID = 12984

GROUP BY employeeID ;



1. Write a query to display the purchase number, date of purchase, dollar amount, number of products for orders made with Logitech.

SELECT purchaseNum, dateOfPurchase, dollarAmount, numProducts

FROM purchaseOrders po, supplier s

WHERE po.supplierID = s.supplierID AND supplierName = "Logitech" ;

