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

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.

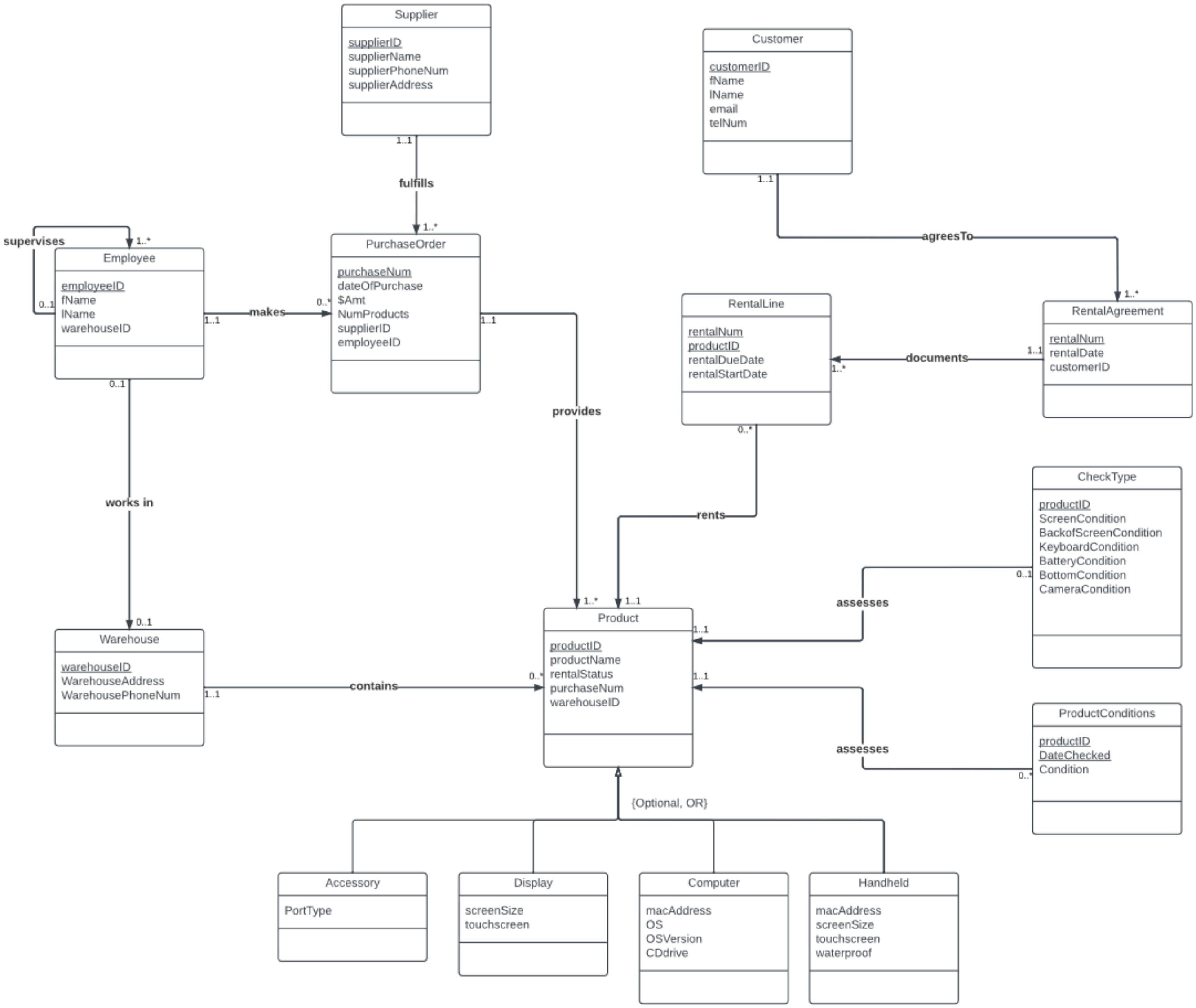
Information to be tracked

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?

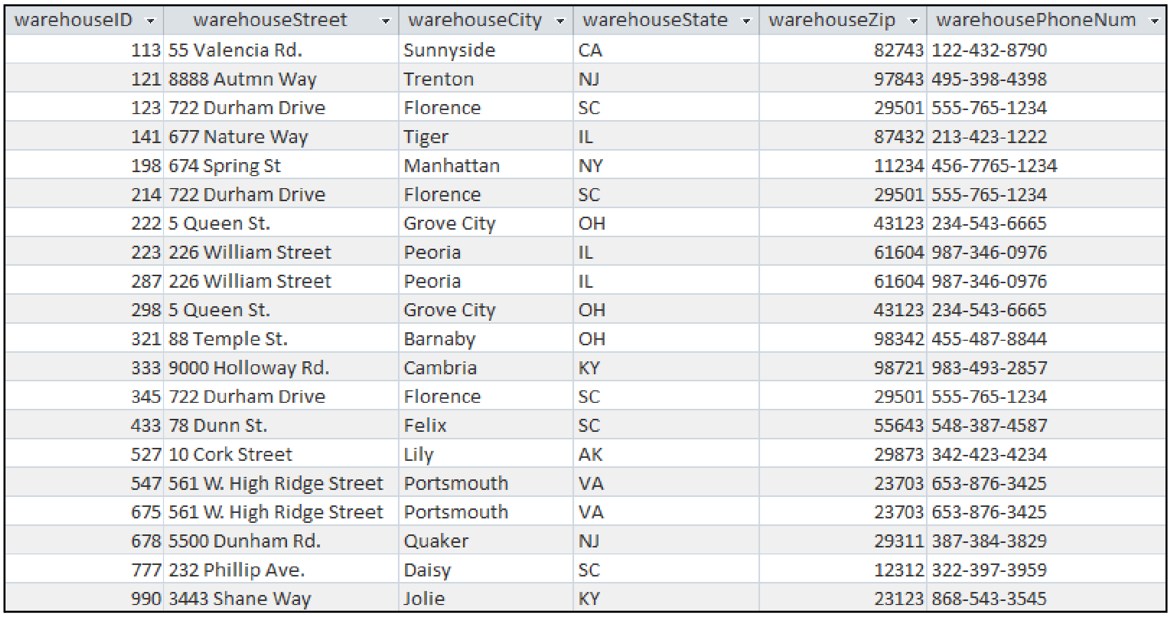
- No, it's in BCNF

Final Relations:

- R1 (productID, dateChecked, condition)

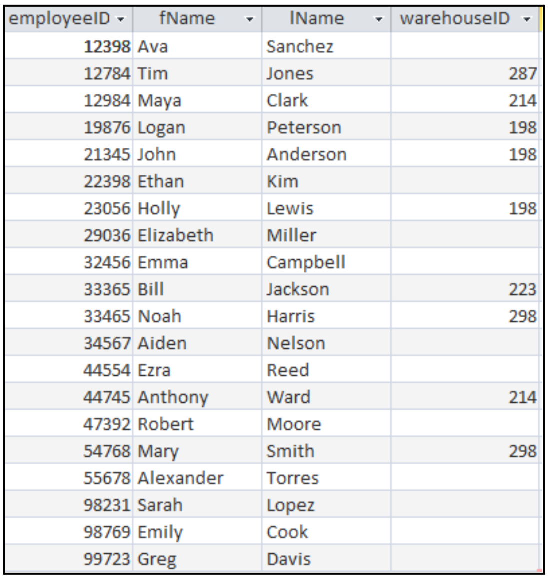
Creating Tables with SQL

```
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’) ;
```



warehouseID	warehouseStreet	warehouseCity	warehouseState	warehouseZip	warehousePhoneNum
113	55 Valencia Rd.	Sunnyside	CA	82743	122-432-8790
121	8888 Autmn Way	Trenton	NJ	97843	495-398-4398
123	722 Durham Drive	Florence	SC	29501	555-765-1234
141	677 Nature Way	Tiger	IL	87432	213-423-1222
198	674 Spring St	Manhattan	NY	11234	456-7765-1234
214	722 Durham Drive	Florence	SC	29501	555-765-1234
222	5 Queen St.	Grove City	OH	43123	234-543-6665
223	226 William Street	Peoria	IL	61604	987-346-0976
287	226 William Street	Peoria	IL	61604	987-346-0976
298	5 Queen St.	Grove City	OH	43123	234-543-6665
321	88 Temple St.	Barnaby	OH	98342	455-487-8844
333	9000 Holloway Rd.	Cambria	KY	98721	983-493-2857
345	722 Durham Drive	Florence	SC	29501	555-765-1234
433	78 Dunn St.	Felix	SC	55643	548-387-4587
527	10 Cork Street	Lily	AK	29873	342-423-4234
547	561 W. High Ridge Street	Portsmouth	VA	23703	653-876-3425
675	561 W. High Ridge Street	Portsmouth	VA	23703	653-876-3425
678	5500 Dunham Rd.	Quaker	NJ	29311	387-384-3829
777	232 Phillip Ave.	Daisy	SC	12312	322-397-3959
990	3443 Shane Way	Jolie	KY	23123	868-543-3545

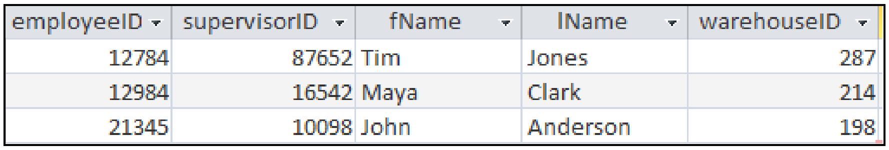
```
CREATE TABLE employee (
employeeID NUMBER NOT NULL,
fName varchar(250),
lName varchar(250),
warehouseID NUMBER
);

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) ;
```



employeeID	fName	lName	warehouseID
12398	Ava	Sanchez	
12784	Tim	Jones	287
12984	Maya	Clark	214
19876	Logan	Peterson	198
21345	John	Anderson	198
22398	Ethan	Kim	
23056	Holly	Lewis	198
29036	Elizabeth	Miller	
32456	Emma	Campbell	
33365	Bill	Jackson	223
33465	Noah	Harris	298
34567	Aiden	Nelson	
44554	Ezra	Reed	
44745	Anthony	Ward	214
47392	Robert	Moore	
54768	Mary	Smith	298
55678	Alexander	Torres	
98231	Sarah	Lopez	
98769	Emily	Cook	
99723	Greg	Davis	

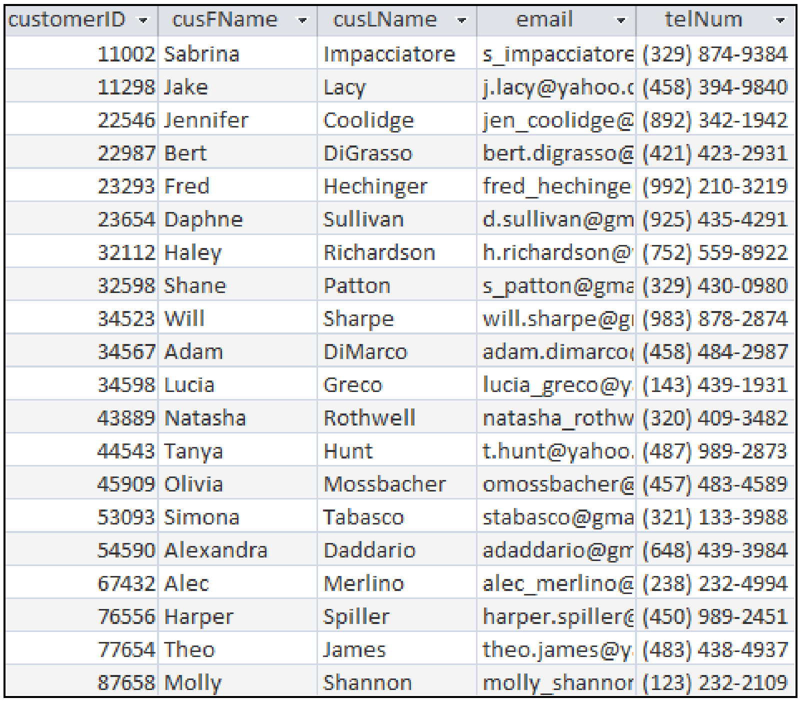
```
CREATE TABLE supervisor (
employeeID NUMBER NOT NULL,
supervisorID NUMBER NOT NULL,
fName varchar(250),
lName varchar(250),
warehouseID NUMBER
);

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) ;
```



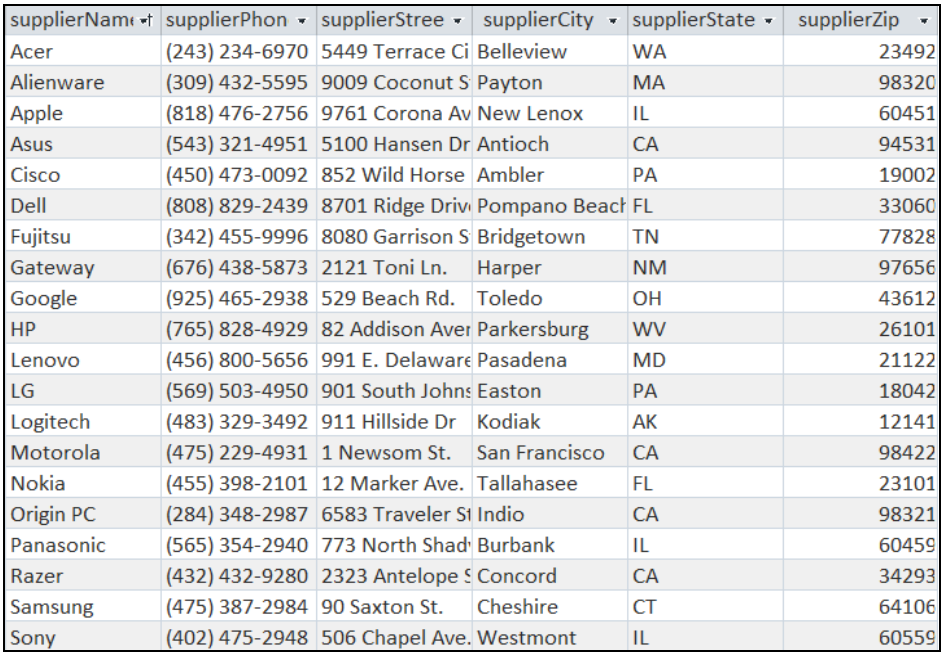
employeeID	supervisorID	fName	lName	warehouseID
12784	87652	Tim	Jones	287
12984	16542	Maya	Clark	214
21345	10098	John	Anderson	198

```
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’) ;
```



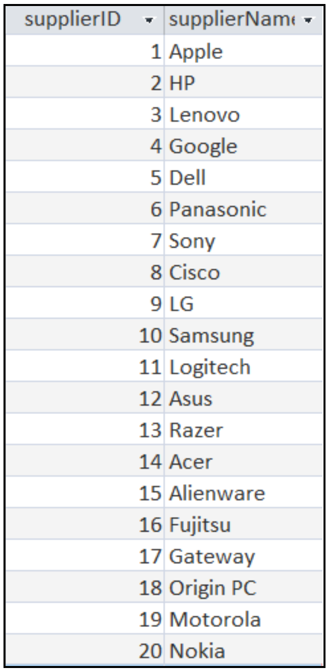
customerID	cusFName	cusLName	email	telNum
11002	Sabrina	Impacciatore	s_impacciatore	(329) 874-9384
11298	Jake	Lacy	j.lacy@yahoo.c	(458) 394-9840
22546	Jennifer	Coolidge	jen_coolidge@	(892) 342-1942
22987	Bert	DiGrasso	bert.digrasso@	(421) 423-2931
23293	Fred	Hechinger	fred_hechinge	(992) 210-3219
23654	Daphne	Sullivan	d.sullivan@gm	(925) 435-4291
32112	Haley	Richardson	h.richardson@	(752) 559-8922
32598	Shane	Patton	s_patton@gma	(329) 430-0980
34523	Will	Sharpe	will.sharpe@gı	(983) 878-2874
34567	Adam	DiMarco	adam.dimarco	(458) 484-2987
34598	Lucia	Greco	lucia_greco@y	(143) 439-1931
43889	Natasha	Rothwell	natasha_rothw	(320) 409-3482
44543	Tanya	Hunt	t.hunt@yahoo.	(487) 989-2873
45909	Olivia	Mossbacher	omossbacher@	(457) 483-4589
53093	Simona	Tabasco	stabasco@gma	(321) 133-3988
54590	Alexandra	Daddario	adaddario@gm	(648) 439-3984
67432	Alec	Merlino	alec_merlino@	(238) 232-4994
76556	Harper	Spiller	harper.spiller@	(450) 989-2451
77654	Theo	James	theo.james@y	(483) 438-4937
87658	Molly	Shannon	molly_shannor	(123) 232-2109

```
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’) ;
```



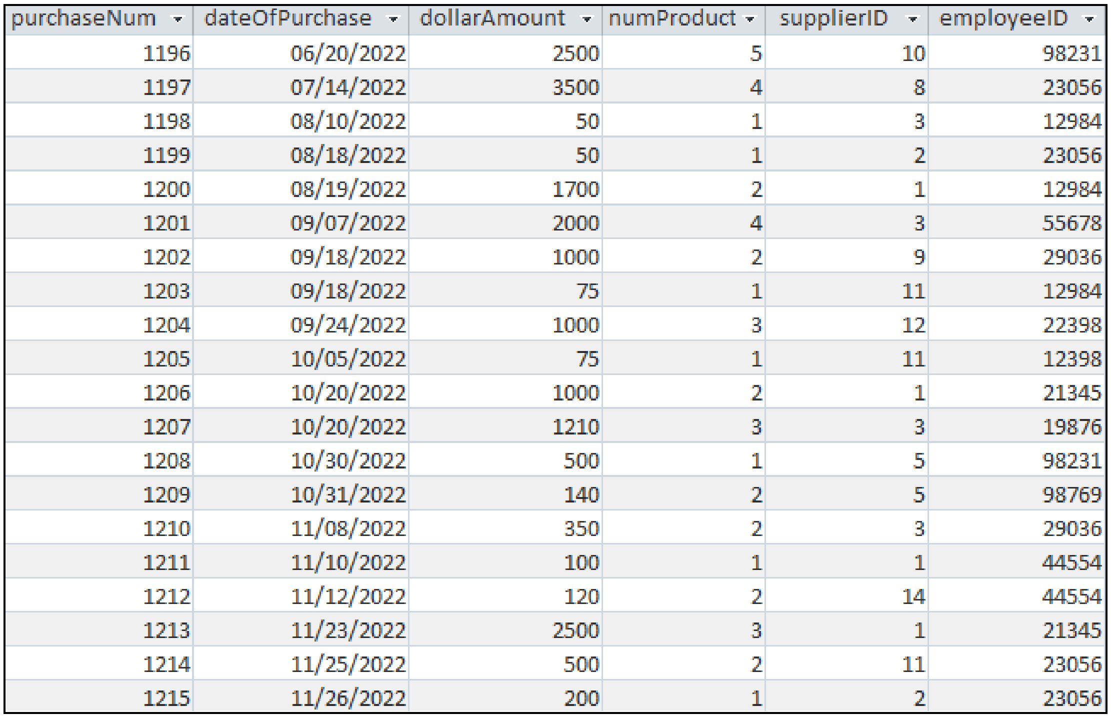
supplierNam	supplierPhon	supplierStree	supplierCity	supplierState	supplierZip
Acer	(243) 234-6970	5449 Terrace Ci	Belleview	WA	23492
Alienware	(309) 432-5595	9009 Coconut S	Payton	MA	98320
Apple	(818) 476-2756	9761 Corona Av	New Lenox	IL	60451
Asus	(543) 321-4951	5100 Hansen Dr	Antioch	CA	94531
Cisco	(450) 473-0092	852 Wild Horse	Ambler	PA	19002
Dell	(808) 829-2439	8701 Ridge Driv	Pompano Beacl	FL	33060
Fujitsu	(342) 455-9996	8080 Garrison S	Bridgetown	TN	77828
Gateway	(676) 438-5873	2121 Toni Ln.	Harper	NM	97656
Google	(925) 465-2938	529 Beach Rd.	Toledo	OH	43612
HP	(765) 828-4929	82 Addison Aver	Parkersburg	WV	26101
Lenovo	(456) 800-5656	991 E. Delaware	Pasadena	MD	21122
LG	(569) 503-4950	901 South Johns	Easton	PA	18042
Logitech	(483) 329-3492	911 Hillside Dr	Kodiak	AK	12141
Motorola	(475) 229-4931	1 Newsom St.	San Francisco	CA	98422
Nokia	(455) 398-2101	12 Marker Ave.	Tallahasee	FL	23101
Origin PC	(284) 348-2987	6583 Traveler St	Indio	CA	98321
Panasonic	(565) 354-2940	773 North Shad	Burbank	IL	60459
Razer	(432) 432-9280	2323 Antelope S	Concord	CA	34293
Samsung	(475) 387-2984	90 Saxton St.	Cheshire	CT	64106
Sony	(402) 475-2948	506 Chapel Ave.	Westmont	IL	60559

```
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’) ;
```



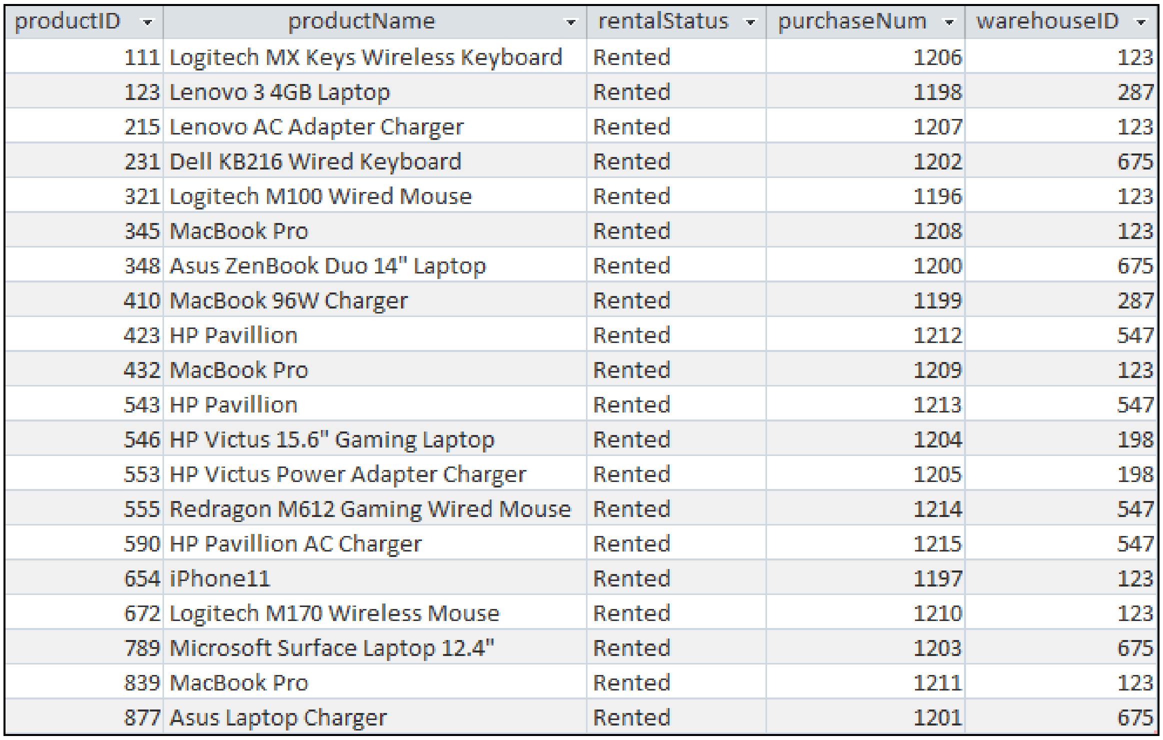
supplierID	supplierNam
1	Apple
2	HP
3	Lenovo
4	Google
5	Dell
6	Panasonic
7	Sony
8	Cisco
9	LG
10	Samsung
11	Logitech
12	Asus
13	Razer
14	Acer
15	Alienware
16	Fujitsu
17	Gateway
18	Origin PC
19	Motorola
20	Nokia

```
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) ;
```



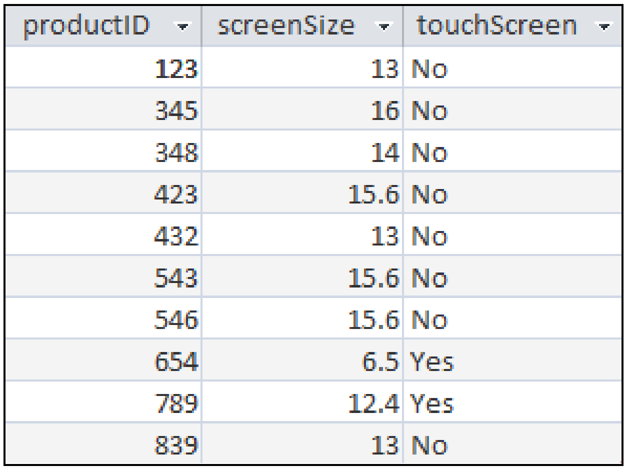
purchaseNum	dateOfPurchase	dollarAmount	numProduct	supplierID	employeeID
1196	06/20/2022	2500	5	10	98231
1197	07/14/2022	3500	4	8	23056
1198	08/10/2022	50	1	3	12984
1199	08/18/2022	50	1	2	23056
1200	08/19/2022	1700	2	1	12984
1201	09/07/2022	2000	4	3	55678
1202	09/18/2022	1000	2	9	29036
1203	09/18/2022	75	1	11	12984
1204	09/24/2022	1000	3	12	22398
1205	10/05/2022	75	1	11	12398
1206	10/20/2022	1000	2	1	21345
1207	10/20/2022	1210	3	3	19876
1208	10/30/2022	500	1	5	98231
1209	10/31/2022	140	2	5	98769
1210	11/08/2022	350	2	3	29036
1211	11/10/2022	100	1	1	44554
1212	11/12/2022	120	2	14	44554
1213	11/23/2022	2500	3	1	21345
1214	11/25/2022	500	2	11	23056
1215	11/26/2022	200	1	2	23056

```
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) ;
```



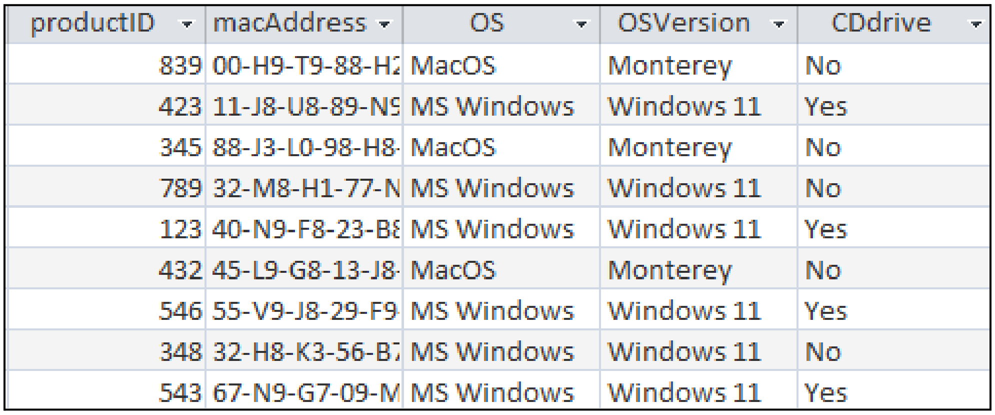
productID	productName	rentalStatus	purchaseNum	warehouseID
111	Logitech MX Keys Wireless Keyboard	Rented	1206	123
123	Lenovo 3 4GB Laptop	Rented	1198	287
215	Lenovo AC Adapter Charger	Rented	1207	123
231	Dell KB216 Wired Keyboard	Rented	1202	675
321	Logitech M100 Wired Mouse	Rented	1196	123
345	MacBook Pro	Rented	1208	123
348	Asus ZenBook Duo 14" Laptop	Rented	1200	675
410	MacBook 96W Charger	Rented	1199	287
423	HP Pavillion	Rented	1212	547
432	MacBook Pro	Rented	1209	123
543	HP Pavillion	Rented	1213	547
546	HP Victus 15.6" Gaming Laptop	Rented	1204	198
553	HP Victus Power Adapter Charger	Rented	1205	198
555	Redragon M612 Gaming Wired Mouse	Rented	1214	547
590	HP Pavillion AC Charger	Rented	1215	547
654	iPhone11	Rented	1197	123
672	Logitech M170 Wireless Mouse	Rented	1210	123
789	Microsoft Surface Laptop 12.4"	Rented	1203	675
839	MacBook Pro	Rented	1211	123
877	Asus Laptop Charger	Rented	1201	675

```
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’) ;
```



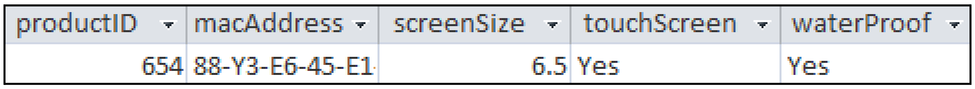
productID	screenSize	touchScreen
123	13	No
345	16	No
348	14	No
423	15.6	No
432	13	No
543	15.6	No
546	15.6	No
654	6.5	Yes
789	12.4	Yes
839	13	No

```
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’) ;
```



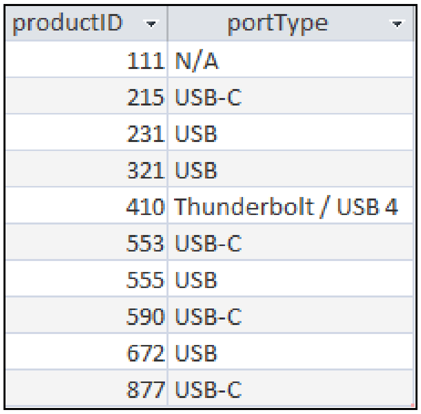
productID	macAddress	OS	OSVersion	CDdrive
839	00-H9-T9-88-H2	MacOS	Monterey	No
423	11-J8-U8-89-N9	MS Windows	Windows 11	Yes
345	88-J3-L0-98-H8-	MacOS	Monterey	No
789	32-M8-H1-77-N	MS Windows	Windows 11	No
123	40-N9-F8-23-B8	MS Windows	Windows 11	Yes
432	45-L9-G8-13-J8-	MacOS	Monterey	No
546	55-V9-J8-29-F9-	MS Windows	Windows 11	Yes
348	32-H8-K3-56-B7	MS Windows	Windows 11	No
543	67-N9-G7-09-M	MS Windows	Windows 11	Yes

```
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’) ;
```



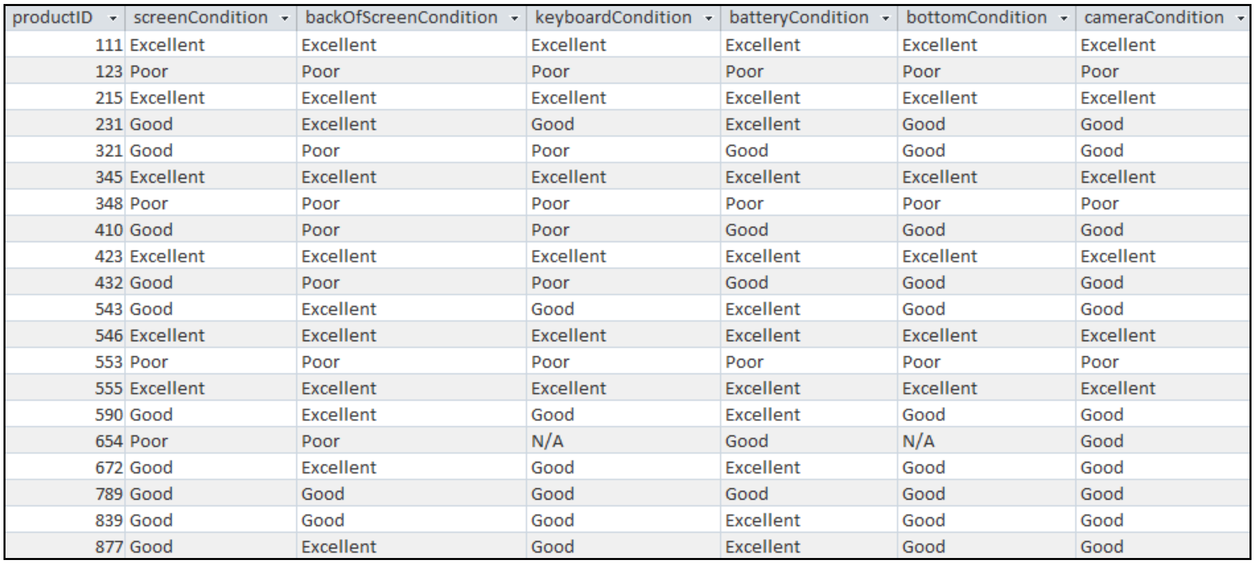
productID	macAddress	screenSize	touchScreen	waterProof
654	88-Y3-E6-45-E1	6.5	Yes	Yes

```
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’) ;
```



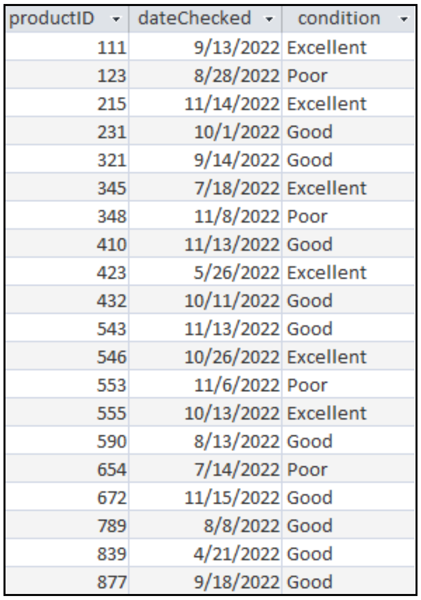
productID	portType
111	N/A
215	USB-C
231	USB
321	USB
410	Thunderbolt / USB 4
553	USB-C
555	USB
590	USB-C
672	USB
877	USB-C

```
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’) ;
```



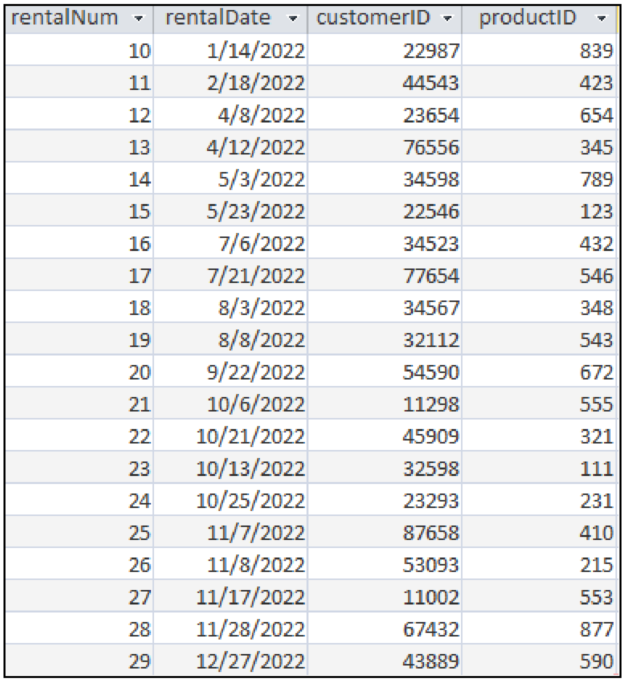
productID	screenCondition	backOfScreenCondition	keyboardCondition	batteryCondition	bottomCondition	cameraCondition
111	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
123	Poor	Poor	Poor	Poor	Poor	Poor
215	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
231	Good	Excellent	Good	Excellent	Good	Good
321	Good	Poor	Poor	Good	Good	Good
345	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
348	Poor	Poor	Poor	Poor	Poor	Poor
410	Good	Poor	Poor	Good	Good	Good
423	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
432	Good	Poor	Poor	Good	Good	Good
543	Good	Excellent	Good	Excellent	Good	Good
546	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
553	Poor	Poor	Poor	Poor	Poor	Poor
555	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
590	Good	Excellent	Good	Excellent	Good	Good
654	Poor	Poor	N/A	Good	N/A	Good
672	Good	Excellent	Good	Excellent	Good	Good
789	Good	Good	Good	Good	Good	Good
839	Good	Good	Good	Excellent	Good	Good
877	Good	Excellent	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’);
```



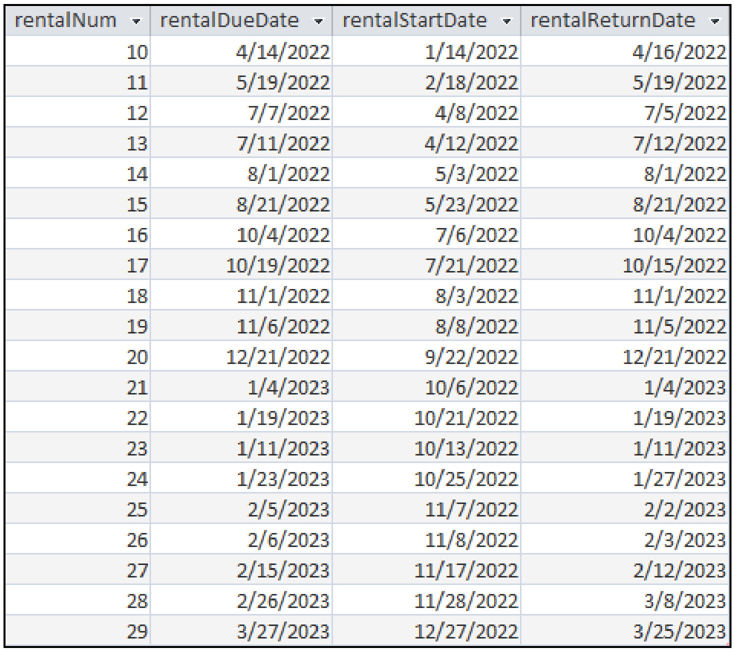
productID	dateChecked	condition
111	9/13/2022	Excellent
123	8/28/2022	Poor
215	11/14/2022	Excellent
231	10/1/2022	Good
321	9/14/2022	Good
345	7/18/2022	Excellent
348	11/8/2022	Poor
410	11/13/2022	Good
423	5/26/2022	Excellent
432	10/11/2022	Good
543	11/13/2022	Good
546	10/26/2022	Excellent
553	11/6/2022	Poor
555	10/13/2022	Excellent
590	8/13/2022	Good
654	7/14/2022	Poor
672	11/15/2022	Good
789	8/8/2022	Good
839	4/21/2022	Good
877	9/18/2022	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) ;
```



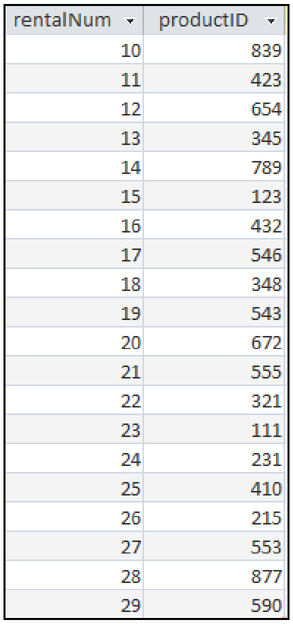
rentalNum	rentalDate	customerID	productID
10	1/14/2022	22987	839
11	2/18/2022	44543	423
12	4/8/2022	23654	654
13	4/12/2022	76556	345
14	5/3/2022	34598	789
15	5/23/2022	22546	123
16	7/6/2022	34523	432
17	7/21/2022	77654	546
18	8/3/2022	34567	348
19	8/8/2022	32112	543
20	9/22/2022	54590	672
21	10/6/2022	11298	555
22	10/21/2022	45909	321
23	10/13/2022	32598	111
24	10/25/2022	23293	231
25	11/7/2022	87658	410
26	11/8/2022	53093	215
27	11/17/2022	11002	553
28	11/28/2022	67432	877
29	12/27/2022	43889	590

```
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”) ;
```



rentalNum	rentalDueDate	rentalStartDate	rentalReturnDate
10	4/14/2022	1/14/2022	4/16/2022
11	5/19/2022	2/18/2022	5/19/2022
12	7/7/2022	4/8/2022	7/5/2022
13	7/11/2022	4/12/2022	7/12/2022
14	8/1/2022	5/3/2022	8/1/2022
15	8/21/2022	5/23/2022	8/21/2022
16	10/4/2022	7/6/2022	10/4/2022
17	10/19/2022	7/21/2022	10/15/2022
18	11/1/2022	8/3/2022	11/1/2022
19	11/6/2022	8/8/2022	11/5/2022
20	12/21/2022	9/22/2022	12/21/2022
21	1/4/2023	10/6/2022	1/4/2023
22	1/19/2023	10/21/2022	1/19/2023
23	1/11/2023	10/13/2022	1/11/2023
24	1/23/2023	10/25/2022	1/27/2023
25	2/5/2023	11/7/2022	2/2/2023
26	2/6/2023	11/8/2022	2/3/2023
27	2/15/2023	11/17/2022	2/12/2023
28	2/26/2023	11/28/2022	3/8/2023
29	3/27/2023	12/27/2022	3/25/2023

```
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);
```

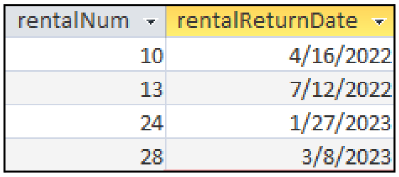


rentalNum	productID
10	839
11	423
12	654
13	345
14	789
15	123
16	432
17	546
18	348
19	543
20	672
21	555
22	321
23	111
24	231
25	410
26	215
27	553
28	877
29	590

Scenarios

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

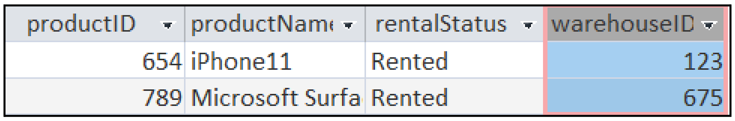
```
SELECT rentalNum, rentalReturnDate
FROM rentalLine
WHERE rentalReturnDate > rentalDueDate ;
```



rentalNum	rentalReturnDate
10	4/16/2022
13	7/12/2022
24	1/27/2023
28	3/8/2023

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

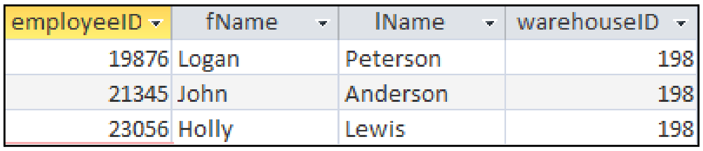
```
SELECT p.productID, p.productName, p.rentalStatus, p.warehouseID
FROM product p, display d
WHERE p.productID = d.productID
AND touchScreen = 'Yes' ;
```



productID	productNam	rentalStatus	warehouseID
654	iPhone11	Rented	123
789	Microsoft Surfa	Rented	675

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

```
SELECT *
FROM employee
WHERE warehouseID = 198 ;
```

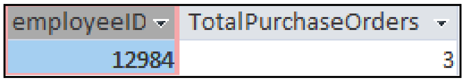


employeeID	fName	lName	warehouseID
19876	Logan	Peterson	198
21345	John	Anderson	198
23056	Holly	Lewis	198

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

```
SELECT employeeID, COUNT(purchaseNum) AS TotalPurchaseOrders
FROM purchaseOrders
```

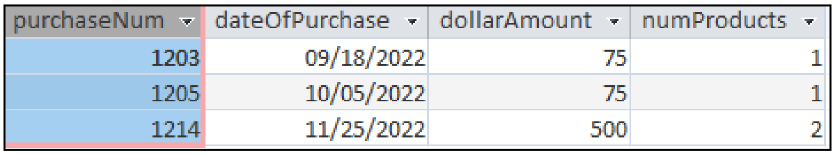
```
WHERE employeeID = 12984
GROUP BY employeeID ;
```



employeeID	TotalPurchaseOrders
12984	3

5. 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" ;
```



purchaseNum	dateOfPurchase	dollarAmount	numProducts
1203	09/18/2022	75	1
1205	10/05/2022	75	1
1214	11/25/2022	500	2