

# Project Title: Camera Equipment Rental

## Previous Reports

### Assignment 1

The camera rental business has been around for 50+ years. First being plagued with paper slips and notes for a single store, now it's typically managed with databases that can manage thousands of rentals for an entire country. Customers have access to a wide array of DSLR, SLR, point and shoots and all types of other equipment that would be impossible to get otherwise. This ease of access is due to the Camera Equipment Rental DBMS.

The database will track all the inventory owned by the company, the customers they are rented to, and the payment system. A customer can search for and rent an item from the inventory of cameras and accessories. They can view the unique camera or accessory specifications for each item. When they complete the rental an invoice is billed to the customer through their billing account with a cost and due date. Each rental is separated by a unique rental ID and is connected to its invoice and customer through InvoiceID and CustomerID. However, if the item is out, the website will show the customer when the item will be returned by checking the item's current rental record and the due date associated with the item.

The project will use 5 main tables "Customers", "Inventory", "Rentals", "Cameras", and "Accessories". For the customer database, only the vital information from each customer is stored, CustomerID, BillingID, CustomerName and CustomerMailingAddress. The primary key is each customer's unique identification number CustomerID. For the inventory database, the items ModelID, Availability, UseCount, Condition, PurchaseID, and RentalCost. The primary key is that each individual store has its own unique identification number InventoryID. For the Camera database, the items specific to sensor resolution, FStop, VideoResolution, LensType, BatteryType, and ProductName are kept. The primary key is each camera's ModelID. For the accessories database, the AccessoryType is kept. The primary key is the ModelID.

Examples of some queries and update operations include:

1. (Query) List all cameras & accessories for a specific brand
2. (Query) List all current & previous rentals of a customer

3. (Update) Adding new equipment to the inventory list with ProductID=23, AcquisitionDate =2023-09-29, Available=1 (Available), and Condition=New, RentalCostIndex = 45 (\$45 per day)
4. (Update) Change the condition of an inventory item from NEW to Excellent Condition
5. (Query) List an invoice for rented equipment

Examples of some relations:

1. Each CAMERA record is related to an INVENTORY record.
2. Each ACCESSORIES record is related to an INVENTORY record.
3. Each RENTAL record is related to one CUSTOMER record and one INVENTORY record.
4. Each INVOICE record is related to one RENTAL record and one BILLING record.

Examples of some of the tables:

Customers (4XXXX)

CustomerID (PK)	BillingID(FK)	Name	Mailing Address
40001	50001	Peter Parker	21 John Street

Rentals (1XXXX)

RentalID (PK)	CustomerID (FK)	InvoiceID (FK)	ItemID (FK)	RentalDate	DueDate
10001	40001	20001	30001	2023-09-15	2023-09-30

Inventory (3XXXX)

ItemID (PK)	ProductID (FK)	Acquisition Date	Available	Condition	RentalCostIndex
30001	90001	2020-08-16	0	5	40
30002	95001	2023-04-12	1	4	20

Cameras (90XXX)

ProductID (PK)	Brand	Name	LensType	Aperture	Megapixels	Resolution	Description
90001	Sony	A7 III	35mm prime	2.5	48MP	8k	Loremipsum

Accessories (95XXX)

ProductID (PK)	Brand	Name	AccessoryType	Weight	Description
95001	FujiFilm	Shoe mount flash EF-X500	Flash	50g	Loremipsum

Billing Accounts

BillingID (PK)	CustomerID (FK)	Balance	Billing Address
50001	40001	-\$600	21 John Street

Invoices

---

InvoiceID (PK)	RentalID (FK)	BillingID (FK)	Cost	DueDate
20001	10001	50001	\$600	2023-09-15

### Assignment 3

#### Create the Customer table

```
CREATE TABLE Customer(
    CustomerID NUMBER PRIMARY KEY,
    CustomerName VARCHAR2(128),
    MailingAddress VARCHAR2(512),
    PhoneNumber CHAR(10),
    BillingAddress VARCHAR2(512),
    Email VARCHAR2(128)
);
```

#### Create the Equipment table

```
CREATE TABLE Equipment(
    EquipmentID NUMBER PRIMARY KEY,
    Brand VARCHAR2(10),
    Description VARCHAR2(40),
    Name VARCHAR2(10)
);
```

#### Create the Camera table

```
CREATE TABLE Camera(
    EquipmentID NUMBER PRIMARY KEY,
    MegaPixel NUMBER,
    Resolution NUMBER,
    LensType VARCHAR2(10),
    Aperture VARCHAR2(10),
    FOREIGN KEY (EquipmentID) REFERENCES Equipment(EquipmentID)
);
```

**Create the Accessory table**

```
CREATE TABLE Accessory(  
    EquipmentID  NUMBER PRIMARY KEY,  
    Weight NUMBER(*,1),  
    Color VARCHAR2(10),  
    AccessoryType VARCHAR2(10),  
    FOREIGN KEY (EquipmentID) REFERENCES Equipment(EquipmentID)  
);
```

**Create the Inventory table**

```
CREATE TABLE Inventory(  
    ItemID  NUMBER PRIMARY KEY,  
    RentalCost  NUMBER(*,2),  
    AcquisitionDate  DATE,  
    Condition VARCHAR2(20),  
    Availability NUMBER(1),  
    EquipmentID NUMBER,  
    FOREIGN KEY (EquipmentID) REFERENCES Equipment(EquipmentID)  
);
```

**Create the Rental table**

```
CREATE TABLE Rental(  
    RentalID  NUMBER PRIMARY KEY,  
    RentalDate  DATE,  
    DueDate  DATE,  
    ItemID NUMBER,  
    FOREIGN KEY (ItemID) REFERENCES Inventory(ItemID),  
    CustomerID NUMBER,  
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),  
    CompleteFlag NUMBER(1)  
);
```

**Create the Invoices table**

```
CREATE TABLE Invoices(  
    InvoiceID NUMBER PRIMARY KEY,  
    PaymentStatus NUMBER(1),  
    DueDate DATE,  
    InvoiceCost NUMBER(*,2),  
    RentalID NUMBER,  
    FOREIGN KEY (RentalID) REFERENCES Rental(RentalID)  
);
```

**Create the Payment table**

```
CREATE TABLE Payment(  
    PaymentID NUMBER PRIMARY KEY,  
    PaymentDate DATE,  
    PaymentAmount NUMBER(*,2),  
    CustomerID NUMBER,  
    InvoiceID NUMBER,  
    FOREIGN KEY (InvoiceID) REFERENCES Invoices(InvoiceID),  
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)  
);
```

**Insert dummy data into Equipment, Inventory, Camera and Accessory table**

```
INSERT INTO equipment VALUES (1,'Brand','TEST','CamerA');  
INSERT INTO camera values (1,16,1960,'prime','f/2.8');  
INSERT INTO inventory values (1, 40,TO_DATE('14-SEP-2000', 'DD-MON-YYYY'),  
'Good', 1,1);  
  
INSERT INTO equipment VALUES (2, 'BRand','TEst','accesoriy');  
INSERT INTO accessory VALUES (2, 14, 'blue','flasj');  
INSERT INTO inventory VALUES (2, 40,TO_DATE('14-SEP-1999', 'DD-MON-YYYY'),  
'Good', 1,2);
```

**Insert dummy data to the customer table**

```
INSERT INTO customer VALUES  
(1,'Eldoss','HOUSE',9119119911,'HOUSE','eldoss@gmail.com');
```

### Insert a new rental record and its invoice

```
INSERT INTO rental VALUES(1, SYSDATE,  
TO_DATE('14-OCT-2023','DD-MON-YYYY'),1,1,1);  
INSERT INTO invoices  
VALUES(1,1,TO_DATE('14-OCT-2023','DD-MON-YYYY'),2900,1);
```

### Insert a payment record

```
INSERT INTO payment  
VALUES(1,TO_DATE('1-OCT-2023','DD-MON-YYYY'),2900,1,1);
```

### Test SELECT statements

```
SELECT * FROM equipment INNER JOIN camera ON  
equipment.equipmentid=camera.equipmentid;  
SELECT * FROM equipment INNER JOIN accessory ON  
equipment.equipmentid=accessory.equipmentid;  
SELECT * FROM customer;  
SELECT * FROM rental WHERE rental.CustomerID = 1;  
SELECT * FROM rental INNER JOIN invoices ON rental.RentalID = invoices.RentalID  
WHERE rental.CustomerID = 1;  
SELECT * FROM payment WHERE payment.CustomerID = 1;
```

## Assignment 4

### Assignment 4 Pt 1.

The query below returns a list of all equipment made by the brand 'Sony'.

```
SELECT * FROM equipment WHERE Brand = 'Sony';
```

Result:

EQUIPMENTID	BRAND	DESCRIPTION	NAME
1	Sony	Alpha A7 III	Mirrorless
6	Sony	Alpha 7R IV	Mirrorless
11	Sony	Rechargeable battery	NP-FZ100 Rechargeable Battery

The query below returns a list of all inventory items ordered by the date it was acquired from oldest to newest.

```
SELECT * FROM inventory ORDER BY ACQUISITIONDATE ASC;
```

Result:

ITEMID	RENTALCOST	ACQUISIT	CONDITION	AVAILABILITY	EQUIPMENTID
8	20	15-07-12	Excellent	1	8
14	10	17-06-05	Fair	1	14
2	35	17-07-21	Good	1	2
7	22	17-10-15	Fair	1	7
9	30	18-03-05	Good	1	9
3	18	18-03-05	Fair	1	3
4	28	18-05-10	Excellent	1	4
13	40	19-09-14	Good	1	13
10	25	19-11-20	Fair	1	10
6	15	20-06-18	Excellent	1	6
11	50	20-08-22	Excellent	1	11
1	50	20-09-14	Excellent	1	1
12	15	20-12-30	Good	1	12
5	42	21-01-02	Good	1	5

The query below returns the sum of items under a specific condition.

```
SELECT condition, COUNT(ITEMID) FROM inventory GROUP BY condition;
```

Result:

CONDITION	COUNT(ITEMID)
Excellent	5
Fair	4
Good	5

The query below returns all the cameras that have a lenstype of 'Full Frame' and are more than 12 megapixels.

```
SELECT * FROM CAMERA WHERE MEGAPIXEL > 12 AND LENSTYPE = 'Full Frame';
```

Result:

EQUIPMENTID	MEGAPIXEL	RESOLUTION	LENSTYPE	APERTURE
1	24	2020	Full Frame	f/2.8
2	30	2016	Full Frame	f/4.0
4	36	2017	Full Frame	f/2.8
6	61	2019	Full Frame	f/1.4

The query below returns all accessories with the type Lens.

```
SELECT * FROM accessory WHERE ACCESSORYTYPE='Lens';
```

Result:



EQUIPMENTID	WEIGHT	COLOR	ACCESSORYT
9	50	Black	Lens
10	24	Black	Lens
12	16	Black	Lens

The query below returns all payments ordered in descending order.

```
SELECT * FROM Payment ORDER BY paymentamount DESC;
```

Result:

PAYMENTID	PAYMENTID	PAYMENTAMOUNT	CUSTOMERID	INVOICEID
1	23-09-30	120	1	2
5	23-10-12	90	2	1
4	23-10-10	75	3	2
3	23-10-05	60	2	3
2	23-09-30	40	1	1

The query below returns the total amount each customer has paid.

```
SELECT CUSTOMERID, SUM(PAYMENTAMOUNT) FROM Payment GROUP BY CUSTOMERID;
```

Result:

CUSTOMERID	SUM(PAYMENTAMOUNT)
1	160
2	150
3	75

The query below returns a list of all items that are past their due date and not completed/returned

```
SELECT * FROM Rental WHERE CompleteFlag=0 and DueDate<SYSDATE;
```

Result:

RENTALID	RENTALDA	DUEDATE	ITEMID	CUSTOMERID	COMPLETEFLAG
3	03-12-23	21-12-25	1	1	0

The query below returns a list of all the rentals and how much money has been paid towards the invoice.

```
SELECT RentalID,SUM(InvoiceCost) FROM Invoices WHERE PAYMENTSTATUS = 1 GROUP BY RentalID;
```

Result:

RENTALID	SUM(INVOICECOST)
1	40
2	5

#### Assignment 4 Pt 2.

```
SELECT * FROM equipment INNER JOIN camera ON
equipment.equipmentid=camera.equipmentid;
```

	EQUIPMENTID	BRAND	DESCRIPTION	NAME	EQUIPMENTID_1	MEGAPIXEL	RESOLUTION	LENTYPE	APERTURE
1	1	Sony	Alpha A7 III	Mirrorless	1	24	2020	Full Frame	f/2.8
2	2	Canon	EOS 5D Mark IV	DSLR	2	30	2016	Full Frame	f/4.0
3	3	Canon	EOS Rebel T7i	DSLR	3	24	2017	APS-C	f/3.5-5.6
4	4	Nikon	D850	DSLR	4	36	2017	Full Frame	f/2.8
5	5	Fujifilm	X-T4	Mirrorless	5	26	2020	APS-C	f/2.8
6	6	Sony	Alpha 7R IV	Mirrorless	6	61	2019	Full Frame	f/1.4
7	7	Panasonic	Lumix GH5	Mirrorless	7	20	2017	M4/3	f/2.8

```
SELECT * FROM equipment INNER JOIN accessory ON
equipment.equipmentid=accessory.equipmentid;
```

	EQUIPMENTID	BRAND	DESCRIPTION	NAME	EQUIPMENTID_1	WEIGHT	COLOR	ACCESSORYTYPE
1	8	Canon	High-performance flash	Speedlite 430EX	8	14	Black	Flash
2	9	Nikon	Sharp lens	AF-S NIKKOR 50mm f/1.8G	9	50	Black	Lens
3	10	Canon	Pro-grade zoom lens	EF 24-70mm f/2.8L II USM	10	24	Black	Lens
4	11	Sony	Rechargeable battery	NP-FZ100 Rechargeable Battery	11	1	Black	Battery
5	12	Fujifilm	Zoom lens for Fujifilm X-series	XF 16-55mm f/2.8 R LM WR	12	16	Black	Lens
6	13	Canon	Rechargeable battery	LP-E6N Rechargeable Battery	13	1	Black	Battery
7	14	Manfrotto	Sturdy aluminum tripod	MT190XPRO3 Aluminum Tripod	14	0	Blue	Tripod

```
SELECT * FROM rental INNER JOIN invoices ON rental.RentalID =
invoices.RentalID WHERE rental.CustomerID = 1;
```

RENTALID	RENTALDATE	DUEDATE	ITEMID	CUSTOMERID	COMPLETEFLAG	INVOICEID	PAYMENTSTATUS	DUEDATE_1	INVOICECOST	RENTALID_1
1	1 03-12-23	23-12-25	1	1	0	1		1 23-10-05	40	1
2	1 03-12-23	23-12-25	1	1	0	2		0 21-02-03	15	1
3	2 03-01-22	24-06-02	1	1	1	3		1 09-01-02	5	2

The view below returns a list of all equipment made by the brand 'Sony'.

```
CREATE VIEW sony_Cameras AS (SELECT * FROM equipment WHERE Brand = 'Sony');
SELECT * FROM sony_Cameras;
```

	EQUIPMENTID	BRAND	DESCRIPTION	NAME
1	1	Sony	Alpha A7 III	Mirrorless
2	6	Sony	Alpha 7R IV	Mirrorless
3	11	Sony	Rechargeable battery	NP-FZ100 Rechargeable Battery

The view below returns all rentals that have not been paid for how much is due

```
CREATE VIEW unpaid_rentals AS (SELECT REN.*, INV.InvoiceCost,
INV.PaymentStatus FROM Rental REN INNER JOIN invoices INV ON REN.RentalID =
INV.RentalID WHERE INV.PaymentStatus=0 and INV.DueDate<SYSDATE);
SELECT * FROM unpaid_rentals;
```

RENTALID	RENTALDATE	DUEDATE	ITEMID	CUSTOMERID	COMPLETEFLAG	INVOICECOST	PAYMENTSTATUS
1	1 03-12-23	23-12-25	1	1	0	15	0

The view below returns all the information about cameras from the combined data in the equipment and camera table.

```
CREATE VIEW camera_info AS (SELECT equipment.brand, equipment.description,
equipment.name, camera.megapixel, camera.resolution, camera.lenstype,
camera.aperture FROM equipment INNER JOIN camera ON
equipment.equipmentid=camera.equipmentid);
SELECT * FROM camera_info;
```

	BRAND	DESCRIPTION	NAME	MEGAPIXEL	RESOLUTION	LENTYPE	APERTURE
1	Sony	Alpha A7 III	Mirrorless	24	2020	Full Frame	f/2.8
2	Canon	EOS 5D Mark IV	DSLR	30	2016	Full Frame	f/4.0
3	Canon	EOS Rebel T7i	DSLR	24	2017	APS-C	f/3.5-5.6
4	Nikon	D850	DSLR	36	2017	Full Frame	f/2.8
5	Fujifilm	X-T4	Mirrorless	26	2020	APS-C	f/2.8
6	Sony	Alpha 7R IV	Mirrorless	61	2019	Full Frame	f/1.4
7	Panasonic	Lumix GH5	Mirrorless	20	2017	M4/3	f/2.8

## Assignment 5

### Unix Implementation

menu.sh

```

| Oracle All Inclusive Tool
|
| Main Menu - Select Desired Operation(s):
|
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>
|
|-----
|----
| M) View Manual
|
| 1) Drop Tables
| 2) Create Tables
| 3) Populate Tables
| 4) Query Tables
|
| X) Force/Stop/Kill Oracle DB
|
| E) End/Exit
| Choose:

```

drop\_tables.sh

```

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
View dropped.

SQL>
View dropped.

SQL>
View dropped.

SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

```

## create\_tables.sh

```

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> 2 3 4 5 6 7 8
Table created.

SQL> 2 3 4 5 6
Table created.

SQL> 2 3 4 5 6 7 8
Table created.

SQL> 2 3 4 5 6 7
Table created.

SQL> 2 3 4 5 6 7 8 9
Table created.

SQL> 2 3 4 5 6 7 8 9 10
Table created.

SQL> 2 3 4 5 6 7 8
Table created.

SQL> 2 3 4 5 6 7 8 9
Table created.

SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

```

## populate\_tables.sh

```
SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

queries.sh

SQL>

INVOICEID	PAYMENTSTATUS	DUE DATE	INVOICECOST	RENTALID
1		1 05-OCT-23	40	1
2		0 03-FEB-21	15	1
3		1 02-JAN-09	5	2

SQL>

RENTALID	SUM(INVOICECOST)
1	40
2	5

SQL>

PAYMENTID	PAYMENTDA	PAYMENTAMOUNT	CUSTOMERID	INVOICEID
2	30-SEP-23	40	1	1
1	30-SEP-23	120	1	2
3	05-OCT-23	60	2	3
4	10-OCT-23	75	3	2
5	12-OCT-23	90	2	1

SQL>

PAYMENTID	PAYMENTDA	PAYMENTAMOUNT	CUSTOMERID	INVOICEID
1	30-SEP-23	120	1	2
5	12-OCT-23	90	2	1
4	10-OCT-23	75	3	2
3	05-OCT-23	60	2	3
2	30-SEP-23	40	1	1

## Queries

The query below gets all the cameras that are not lens type APS-C

```
SELECT equipmentid, description , name
FROM equipment
WHERE NOT EXISTS ( Select * FROM camera
                    WHERE LensType = 'APS-C' AND
                    equipment.equipmentid = camera.equipmentid );
```

Result:

	EQUIPMENTID	DESCRIPTION	NAME
1	8	High-performance flash	Speedlite 430EX
2	12	Zoom lens for Fujifilm X-series	XF 16-55mm f/2.8 R LM WR
3	10	Pro-grade zoom lens	EF 24-70mm f/2.8L II USM
4	7	Lumix GH5	Mirrorless
5	4	D850	DSLR
6	14	Sturdy aluminum tripod	MT190XPRO3 Aluminum Tripod
7	9	Sharp lens	AF-S NIKKOR 50mm f/1.8G
8	13	Rechargeable battery	LP-E6N Rechargeable Battery
9	1	Alpha A7 III	Mirrorless
10	2	EOS 5D Mark IV	DSLR
11	11	Rechargeable battery	NP-FZ100 Rechargeable Battery
12	6	Alpha 7R IV	Mirrorless

The query list all cameras that are either Sony or Nikon

```
SELECT Brand, Description, Name, megapixel, resolution, Lenstype, aperture
FROM equipment INNER JOIN camera ON equipment.equipmentid=camera.equipmentid
WHERE brand='Sony' UNION
SELECT Brand, Description, Name, megapixel, resolution, Lenstype, aperture
FROM equipment INNER JOIN camera ON equipment.equipmentid=camera.equipmentid
WHERE brand='Nikon';
```

Result:

	BRAND	DESCRIPTION	NAME	MEGAPIXEL	RESOLUTION	LENSTYPE	APERTURE
1	Nikon	D850	DSLR	36	2017	Full Frame	f/2.8
2	Sony	Alpha 7R IV	Mirrorless	61	2019	Full Frame	f/1.4
3	Sony	Alpha A7 III	Mirrorless	24	2020	Full Frame	f/2.8



Gets invoice information and minus all the invoices that have paymentstatus 1

```
SELECT customer.customername, rental.itemID, invoices.invoiceid,  
rental.rentaldate, rental.duedate, invoices.paymentstatus  
FROM rental  
INNER JOIN invoices ON rental.RentalID = invoices.RentalID  
INNER JOIN customer ON rental.CustomerID = customer.CustomerID MINUS  
SELECT customer.customername, rental.itemID, invoices.invoiceid,  
rental.rentaldate, rental.duedate, invoices.paymentstatus  
FROM rental  
INNER JOIN invoices ON rental.RentalID = invoices.RentalID  
INNER JOIN customer ON rental.CustomerID = customer.CustomerID WHERE  
invoices.paymentstatus = 1;
```

Result:

	CUSTOMERNAME	ITEMID	INVOICEID	RENTALDATE	DUE DATE	PAYMENTSTATUS
1	Sug	1	2	03-12-23	23-12-25	0
2	cat	5	3	03-12-23	21-12-25	0

Gets all the customers and minus the customer with same billing and mailing address

```
SELECT * FROM CUSTOMER MINUS( SELECT * FROM CUSTOMER WHERE billingaddress =  
mailingaddress);
```

Result:

	CUSTOMERID	CUSTOMERNAME	MAILINGADDRESS	PHONENUMBER	BILLINGADDRESS	EMAIL
1	2	dog	511 Italy	11341	451 band	bohnhn@gmail.com
2	3	cat	21 arabia	3451	11 Texas	Dougggy@hotmail.com
3	6	Michael	456 Oak Avenue	5559876543	789 Maple Drive	michael@email.com
4	7	Emily	789 Pine Lane	5555678901	101 Cherry Street	emily@email.com

The query below list all brands that are rented and the sum of their total rental cost.

```
SELECT brand, SUM(rentalcost) FROM (  
Select * FROM equipment INNER JOIN inventory ON equipment.equipmentid =  
inventory.itemid  
INNER JOIN rental ON rental.itemid = inventory.itemid  
) GROUP BY brand;
```

Result:

	BRAND	SUM(RENTALCOST)
1	Sony	50
2	Nikon	28
3	Fujifilm	42

The query below lists all brands that have been rented and the average of their invoices where the average is greater than 10

```
SELECT brand, AVG(InvoiceCost) FROM (
    Select * FROM equipment
    INNER JOIN inventory
    ON equipment.equipmentid = inventory.itemid
    INNER JOIN rental
    ON rental.itemid = inventory.itemid
    INNER JOIN invoices
    ON rental.RentalID = invoices.RentalID
) GROUP BY brand
HAVING AVG(InvoiceCost) > 10;
```

Result:

	BRAND	AVG(INVOICECOST)
1	Sony	16.25
2	Canon	47.5

## Assignment 6

### Customer

CustomerID → {CustomerName, MailingAddress, PhoneNumber, BillingAddress, Email}

### Payment

PaymentID → {PaymentDate, PaymentAmount, CustomerID, InvoiceID}

### Rental

RentalID → {RentalDate, DueDate, ItemID, CustomerID, CompleteFlag}

### Invoices

InvoiceID → {PaymentStatus, DueDate, InvoiceCost, RentalID}

### Inventory

ItemID → {RentalCost, AcquisitionDate, Condition, Availability, EquipmentID}

### Equipment

EquipmentID  $\rightarrow$  {Brand,Description,Name}

#### **Camera**

EquipmentID  $\rightarrow$  {MegaPixel,Resolution,LensType,Aperture}

#### **Accessory**

EquipmentID  $\rightarrow$  {Weight,Color,AccessoryType}

#### **customer\_makes\_payment (1:M)**

PaymentID  $\rightarrow$  CustomerID

#### **customer\_makes\_rental (1:M)**

RentalID  $\rightarrow$  CustomerID

#### **payment\_for\_invoices (1:1)**

PaymentID  $\rightarrow$  InvoiceID

InvoiceID  $\rightarrow$  PaymentID

#### **invoices\_has\_rental (1:1)**

InvoiceID  $\rightarrow$  RentalID

RentalID  $\rightarrow$  InvoiceID

#### **rental\_has\_inventory (1:1)**

RentalID  $\rightarrow$  ItemID

ItemID  $\rightarrow$  RentalID

#### **equipment\_contains\_inventory (1:M)**

ItemID  $\rightarrow$  EquipmentID

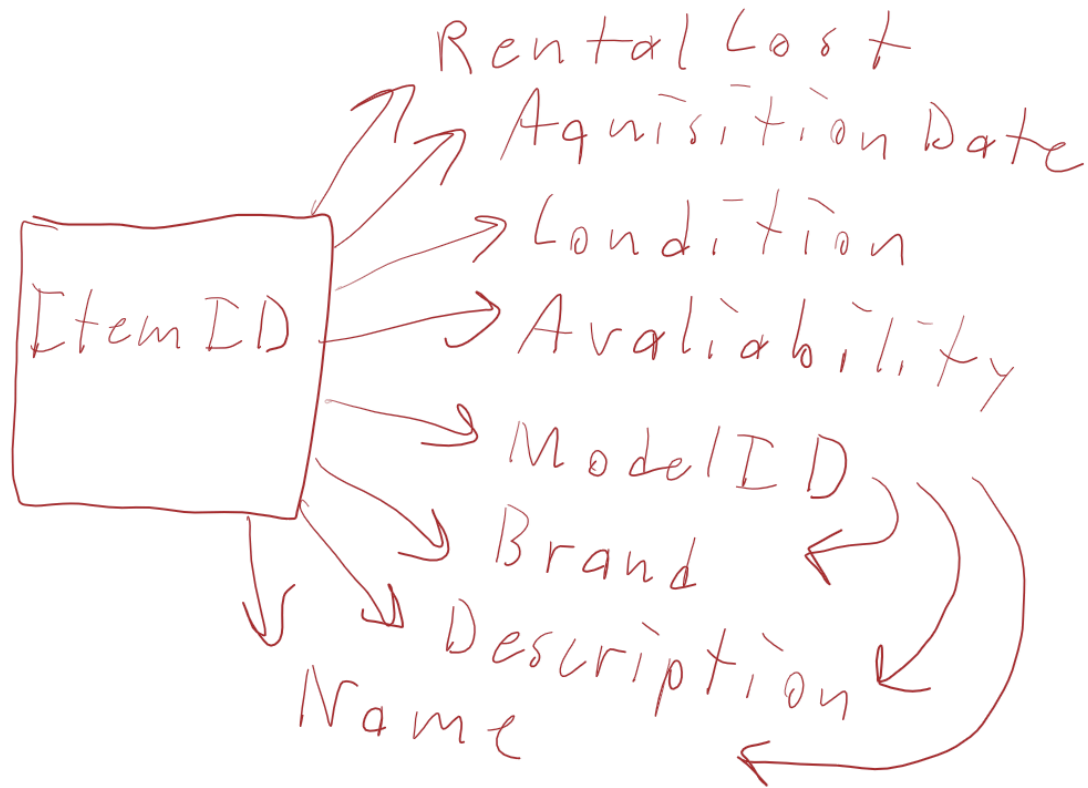
### **Assignment 7**

#### **Transitive FD**

Schema: inventory(ItemID, RentalCost, AcquisitionDate, Condition, Availability, ModelID, Brand, Description, Name)

FD:

1. ItemID  $\rightarrow$  RentalCost, AcquisitionDate, Condition, Availability, ModelID, Brand, Description, Name
2. ModelID  $\rightarrow$  Brand, Description, Name



Brand, Description and Name are transitively dependent on ModelID, making this table not 3NF.

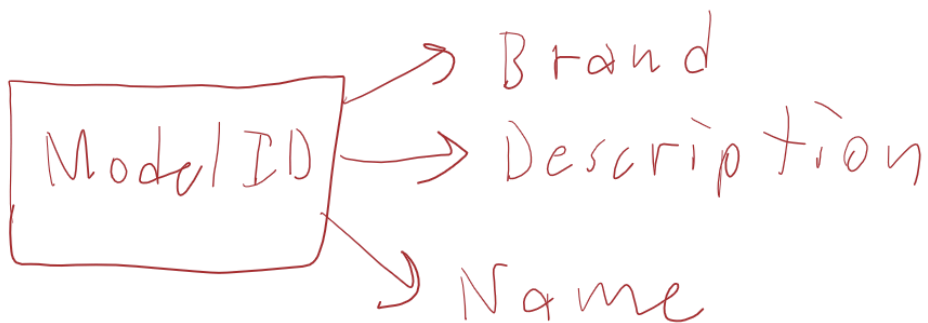
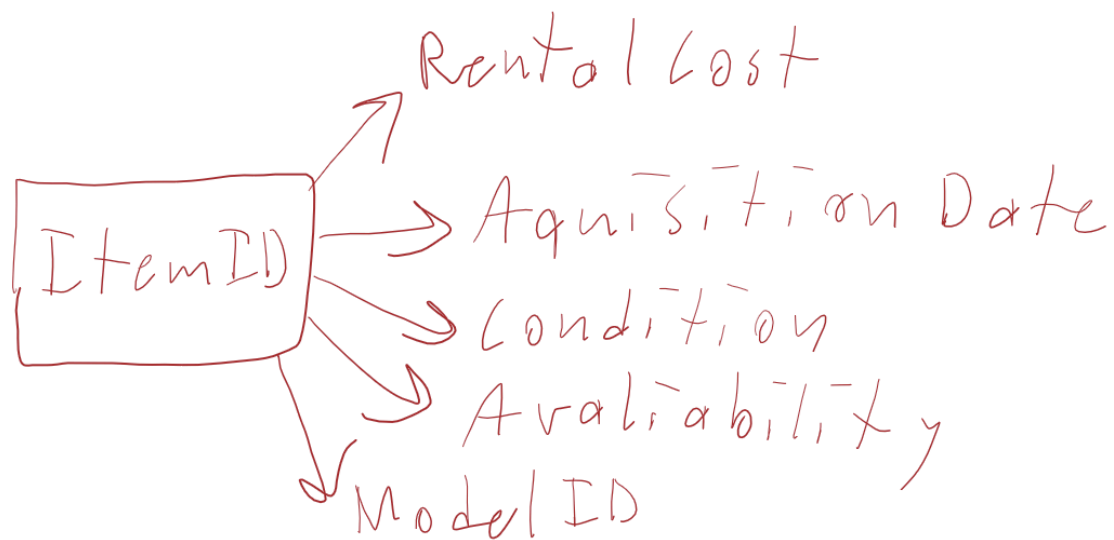
#### Transform to 3NF

Schema: Inventory(ItemID, RentalCost, AcquisitionDate, Condition, Availability, ModelID)  
Equipment(ModelID, Brand, Description, Name)

FD:

Inventory: ItemID → RentalCost, ModelID

Equipment: ModelID → Brand, Description, Name

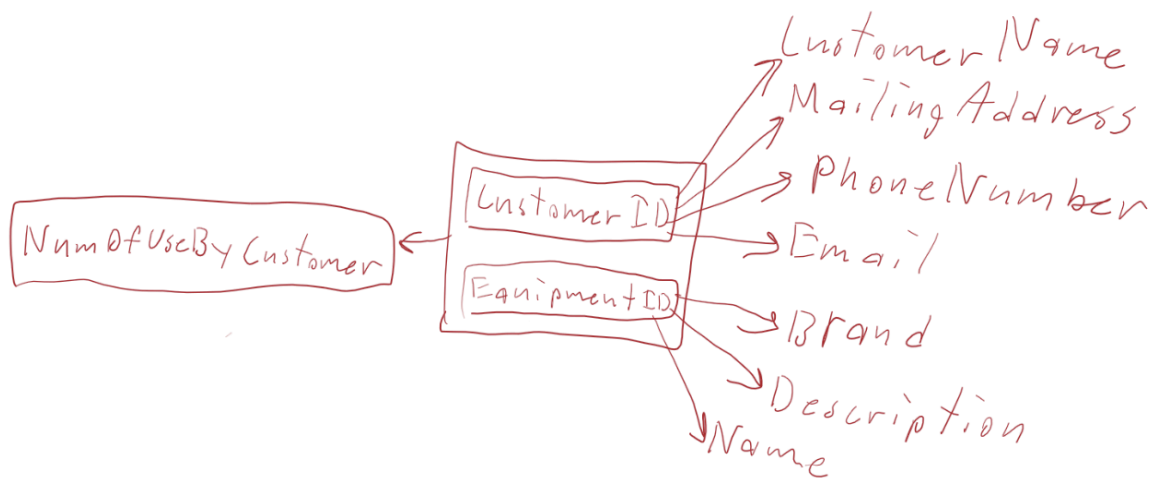


**Compound Primary Key (Partial Dependency) to 2NF:**

Inventory(CustomerID, EquipmentID, NumOfUseByCustomer, CustomerName, MailingAddress, PhoneNumber, Email, Brand, Description, Name)

FD:

- CustomerID  $\rightarrow$  CustomerName, MailingAddress, PhoneNumber, Email
- EquipmentID  $\rightarrow$  Brand, Description, Name
- CustomerID, EquipmentID  $\rightarrow$  NumOfUseByCustomer

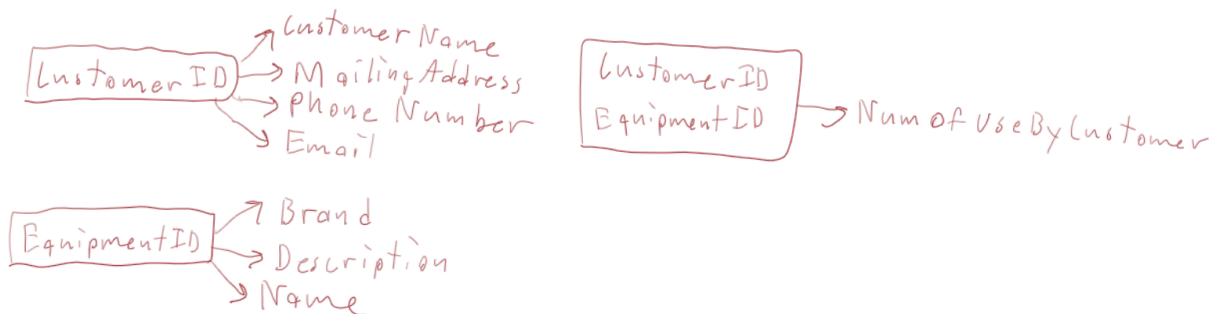


Schema for 2NF Table:

Customer(CustomerID, CustomerName, MailingAddress, PhoneNumber, Email)

Equipment(EquipmentID, Brand, Description, Name)

ReoccurringRentals(CustomerID, EquipmentID, NumOfUseByCustomer)



After normalization, all our tables are in 3NF.

### Customer

CustomerID → {CustomerName, MailingAddress, PhoneNumber, BillingAddress, Email}

### Payment

PaymentID → {PaymentDate, PaymentAmount, CustomerID, InvoiceID}

### Rental

RentalID → {RentalDate, DueDate, ItemID, CustomerID, CompleteFlag}

#### **Invoices**

InvoiceID → {PaymentStatus, DueDate, InvoiceCost, RentalID}

#### **Inventory**

ItemID → {RentalCost, AcquisitionDate, Condition, Availability, EquipmentID}

#### **Equipment**

EquipmentID → {Brand, Description, Name}

#### **Camera**

EquipmentID → {MegaPixel, Resolution, LensType, Aperture}

#### **Accessory**

EquipmentID → {Weight, Color, AccessoryType}

### **Assignment 8**

#### **Customers Table:**

CustomerID → {CustomerName, MailingAddress, PhoneNumber, BillingAddress, Email}

#### **Inventory Table:**

ItemID → {RentalCost, AcquisitionDate, Condition, Availability, EquipmentID}

#### **Equipment Table:**

EquipmentID → {Brand, Description, Name}

#### **Camera Table:**

EquipmentID → {MegaPixel, Resolution, LensType, Aperture}

#### **Accessory Table:**

EquipmentID → {Weight, Color, AccessoryType}

#### **Rentals Table:**

{RentalID} → {ItemID, CustomerID, RentalDate, DueDate, CompleteFlag}

#### **Invoices Table:**

{InvoiceID} → {RentalID, PaymentStatus, DueDate, InvoiceCost}

#### **Payments Table:**

{PaymentID} → {PaymentDate, PaymentAmount, CustomerID, InvoiceID}

All of our tables are already in BCNF.

### Step 1: Determine functional dependencies

The addition of transitive and partial dependencies will be added onto Customers as seen below.

$R(\text{Cus\_ID}, \text{Cus\_Name}, \text{Mail\_Addr}, \text{Phone\_No}, \text{Bill\_Addr}, \text{Bill\_ID}, \text{Bill\_Bal}, \text{Cus\_Rank})$

$\text{Cus\_ID} \rightarrow \{\text{Cus\_Name}, \text{Mail\_Addr}, \text{Phone\_No}, \text{Bill\_Addr}, \text{Bill\_ID}, \text{Bill\_Bal}, \text{Cus\_Rank}\}$

$\text{Phone\_No}, \text{Bill\_Addr} \rightarrow \{\text{Bill\_ID}\}$

$\text{Phone\_No}, \text{Bill\_ID} \rightarrow \{\text{Bill\_Bal}\}$

$\text{Bill\_ID} \rightarrow \{\text{Bill\_Bal}\}$

$\text{Bill\_Bal} \rightarrow \{\text{Cus\_Rank}\}$

$\text{Cus\_Rank} \rightarrow \{\text{Cus\_Name}\}$

### Step 2a: Break RHS and find redundancies

FD:

$\text{Cus\_ID} \rightarrow \text{Cus\_Name}$

$\text{Cus\_ID} \rightarrow \text{Mail\_Addr}$

$\text{Cus\_ID} \rightarrow \text{Phone\_No}$

$\text{Cus\_ID} \rightarrow \text{Bill\_Addr}$

$\text{Phone\_No}, \text{Bill\_Addr} \rightarrow \text{Bill\_ID}$

$\text{Phone\_No}, \text{Bill\_ID} \rightarrow \text{Bill\_Bal}$

$\text{Bill\_ID} \rightarrow \text{Bill\_Bal}$



$\text{Bill\_Bal} \rightarrow \text{Cus\_Rank}$

$\text{Cus\_Rank} \rightarrow \text{Cus\_Name}$

FD = {C  $\rightarrow$  N  
C  $\rightarrow$  P  
C  $\rightarrow$  A  
C  $\rightarrow$  M  
C  $\rightarrow$  B  
C  $\rightarrow$  L  
C  $\rightarrow$  R  
PA  $\rightarrow$  B  
PB  $\rightarrow$  L  
B  $\rightarrow$  L  
L  $\rightarrow$  R  
R  $\rightarrow$  N}

C  $\rightarrow$  N: C+= {C,P,A,M,B,L,R,N} We get N, so redundant

C  $\rightarrow$  P: C+= {C,N,A,M,B,L,R} We do not get P, so not redundant

C  $\rightarrow$  A: C+= {C,N,P,M,B,L,R} We do not get A, so not redundant

C  $\rightarrow$  M: C+= {C,N,P,A,B,L,R} We do not get M, so not redundant

C  $\rightarrow$  B: C+= {N,P,A,M,L,R,B,L} We get B, so redundant

C  $\rightarrow$  L: C+= {N,P,A,M,B,R,L} We get L, so redundant

C  $\rightarrow$  R: C+= {N,P,A,M,B,R,L} We get R, so redundant

PA  $\rightarrow$  B: PA+= {P,A} We do not get B, so not redundant

$PB \rightarrow L$ :  $PB \rightarrow L$  We get L, so redundant.

$B \rightarrow L$ :  $B \rightarrow L$  We do not get L, so not redundant

$L \rightarrow R$ :  $L \rightarrow R$  We do not get R, so not redundant

$R \rightarrow N$ :  $R \rightarrow N$  We do not get N, so not redundant

**So, after removing redundancies:**

FD = {  
   $C \rightarrow P$   
   $C \rightarrow A$   
   $C \rightarrow M$   
   $PA \rightarrow B$   
   $B \rightarrow L$   
   $L \rightarrow R$   
   $R \rightarrow N$ }

**Step 2b: Find and remove partial dependencies (or minimizing LHS)**

$PA \rightarrow B$ :  $PA \rightarrow B$ ,  $A \rightarrow B$  so this FD is fully dependent.

$PB \rightarrow L$ :  $PB \rightarrow L$ ,  $B \rightarrow L$ , so it is partially dependent. But it was removed in the step 2a.

**Step 3: Find keys**

FD = {  
   $C \rightarrow P$   
   $C \rightarrow A$   
   $C \rightarrow M$   
   $PA \rightarrow B$   
   $B \rightarrow L$   
   $L \rightarrow R$   
   $R \rightarrow N$ }

So {C} is the only element in the key.

$C \rightarrow \{C, P, A, M, B, L, R, N\}$

**Step 4: Make tables**

We have FD : {  
   $C \rightarrow P$   
   $C \rightarrow A$   
   $C \rightarrow M$   
   $PA \rightarrow B$   
   $B \rightarrow L$

$$\begin{aligned} L &\rightarrow R \\ R &\rightarrow N \end{aligned}$$

We make a relation for each Functional dependency.

We combine  $C \rightarrow P$ ,  $C \rightarrow A$ ,  $C \rightarrow M$  to  $C \rightarrow \{P,A,M\}$

$R1(C,P,A,M)$  with FD  $C \rightarrow \{P,A,M\}$   
 $R2(P,A,B)$  with FD  $PA \rightarrow \{B\}$  We combined  $C \rightarrow P$  and  $C \rightarrow A$   
 $R3(B,L)$  with FD  $B \rightarrow \{L\}$   
 $R4(L,R)$  with FD  $L \rightarrow \{R\}$   
 $R5(R,N)$  with FD  $R \rightarrow \{N\}$

## Assignment 9

## Assignment 10 (Queries RA)

### Simple Queries:

SELECT \* FROM equipment WHERE Brand = 'Sony';

$\sigma_{Brand='Sony'}(equipment)$

SELECT \* FROM inventory ORDER BY ACQUISITIONDATE ASC;

inventory

No Equivalent Relational model for ascending order

SELECT condition, COUNT(ITEMID) FROM inventory GROUP BY condition;

condition  $F_{count(ITEMID)}(inventory)$

SELECT \* FROM CAMERA WHERE MEGAPIXEL > 12 AND LENSTYPE = 'Full Frame';

$\sigma_{MEGAPIXEL>12 \ \& \ LENSTYPE='Full \ Frame'}(Camera)$

SELECT \* FROM accessory WHERE ACCESSORYTYPE='Lens';

$\sigma_{ACCESSORYTYPE='Lens'}(accessory)$

SELECT \* FROM Payment ORDER BY paymentamount DESC;

Payement

No Equivalent Relational model for ascending order

SELECT CUSTOMERID, SUM(PAYMENTAMOUNT) FROM Payment GROUP BY CUSTOMERID;

CUSTOMERID  $F_{SUM(PAYMENTAMOUNT)}(Payement)$

The query below returns a list of all items that are past their due date and not completed/returned

```
SELECT * FROM Rental WHERE CompleteFlag=0 and DueDate<SYSDATE;
```

$\sigma_{CompleteFlag=0 \ \& \ DueDate<SYSDATE}(Rental)$

The query below returns a list of all the rentals and how much money has been paid towards the invoice.

```
SELECT RentalID,SUM(InvoiceCost) FROM Invoices WHERE PAYMENTSTATUS = 1 GROUP BY RentalID;
```

$RENTALID \bowtie_{SUM(NVOICECOST)}(\sigma_{PAYMENTSTATUS=1}(Invoices))$

### Advance Queries:

```
SELECT * FROM equipment INNER JOIN camera ON
equipment.equipmentid=camera.equipmentid;
```

$equipment \bowtie camera$

```
SELECT * FROM equipment INNER JOIN accessory ON
equipment.equipmentid=accessory.equipmentid;
```

$equipment \bowtie accessory$

```
SELECT * FROM rental INNER JOIN invoices ON rental.RentalID =
invoices.RentalID WHERE rental.CustomerID = 1;
```

$\sigma_{CustomerID=1}(rental) \bowtie invoices$

### LAB 5 Queries:

The query below gets all the cameras that are not lens type APS-C

```
SELECT equipmentid, description , name
FROM equipment
WHERE NOT EXISTS ( Select * FROM camera
                    WHERE LensType = 'APS-C' AND
                    equipment.equipmentid = camera.equipmentid );
```

$APS \leftarrow \sigma_{LensType = 'APS-C'}(Equipment \bowtie camera)$

$Result \leftarrow (Equipment) - APS$

$\Pi_{equipmentid, description , name}(Result)$

The query list all cameras that are either Sony or Nikon

```

SELECT Brand, Description, Name, megapixel, resolution, Lenstype, aperture
FROM equipment INNER JOIN camera ON equipment.equipmentid=camera.equipmentid
WHERE brand='Sony' UNION
SELECT Brand, Description, Name, megapixel, resolution, Lenstype, aperture
FROM equipment INNER JOIN camera ON equipment.equipmentid=camera.equipmentid
WHERE brand='Nikon';

```

```

 $\Pi_{\text{Brand, Description, Name, megapixel, resolution, Lenstype, aperture}}(\sigma_{\text{brand} = \text{'Sony'}}(\text{Equipment} \bowtie \text{camera})) \cup$ 
 $\Pi_{\text{Brand, Description, Name, megapixel, resolution, Lenstype, aperture}}(\sigma_{\text{brand} = \text{'Nikon'}}(\text{Equipment} \bowtie \text{camera}))$ 

```

**Gets invoice information and minus all the invoices that have paymentstatus 1**

```

SELECT customer.customername, rental.itemID, invoices.invoiceid,
rental.rentaldate, rental.duedate, invoices.paymentstatus
FROM rental
INNER JOIN invoices ON rental.RentalID = invoices.RentalID
INNER JOIN customer ON rental.CustomerID = customer.CustomerID MINUS
SELECT customer.customername, rental.itemID, invoices.invoiceid,
rental.rentaldate, rental.duedate, invoices.paymentstatus
FROM rental
INNER JOIN invoices ON rental.RentalID = invoices.RentalID
INNER JOIN customer ON rental.CustomerID = customer.CustomerID WHERE
invoices.paymentstatus = 1;

```

```

 $(\Pi_{\text{customer.customername, rental.itemID, invoices.invoiceid, rental.rentaldate, rental.duedate, invoices.paymentstatus}}((\text{rental} \bowtie \text{invoices}) \bowtie \text{customer}))) - (\Pi_{\text{customer.customername, rental.itemID, invoices.invoiceid, rental.rentaldate, rental.duedate, invoices.paymentstatus}}(\sigma_{\text{invoices.paymentstatus} = 1}((\text{rental} \bowtie \text{invoices}) \bowtie \text{customer}))))$ 

```

**Gets all the customers and minus the customer with same billing and mailing address**

```

SELECT * FROM CUSTOMER MINUS( SELECT * FROM CUSTOMER WHERE billingaddress = mailingaddress);
CUSTOMER-( $\sigma_{\text{billingaddress} = \text{mailingaddress}}(\text{customer})$ )

```

The query below list all brands that are rented and the sum of their total rental cost.

```
SELECT brand, SUM(rentalcost) FROM (
Select * FROM equipment INNER JOIN inventory ON equipment.equipmentid =
inventory.itemid
INNER JOIN rental ON rental.itemid = inventory.itemid
) GROUP BY brand;
```

```
brand F SUM(rentalcost) (equipment equipment.equipmentid = inventory.itemid (inventory rental))
```

The query below lists all brands that have been rented and the average of their invoices where the average is greater than 10

```
SELECT brand, AVG(InvoiceCost) FROM (
    Select * FROM equipment
    INNER JOIN inventory
    ON equipment.equipmentid = inventory.itemid
    INNER JOIN rental
    ON rental.itemid = inventory.itemid
    INNER JOIN invoices
    ON rental.RentalID = invoices.RentalID
) GROUP BY brand
HAVING AVG(InvoiceCost) > 10;
```

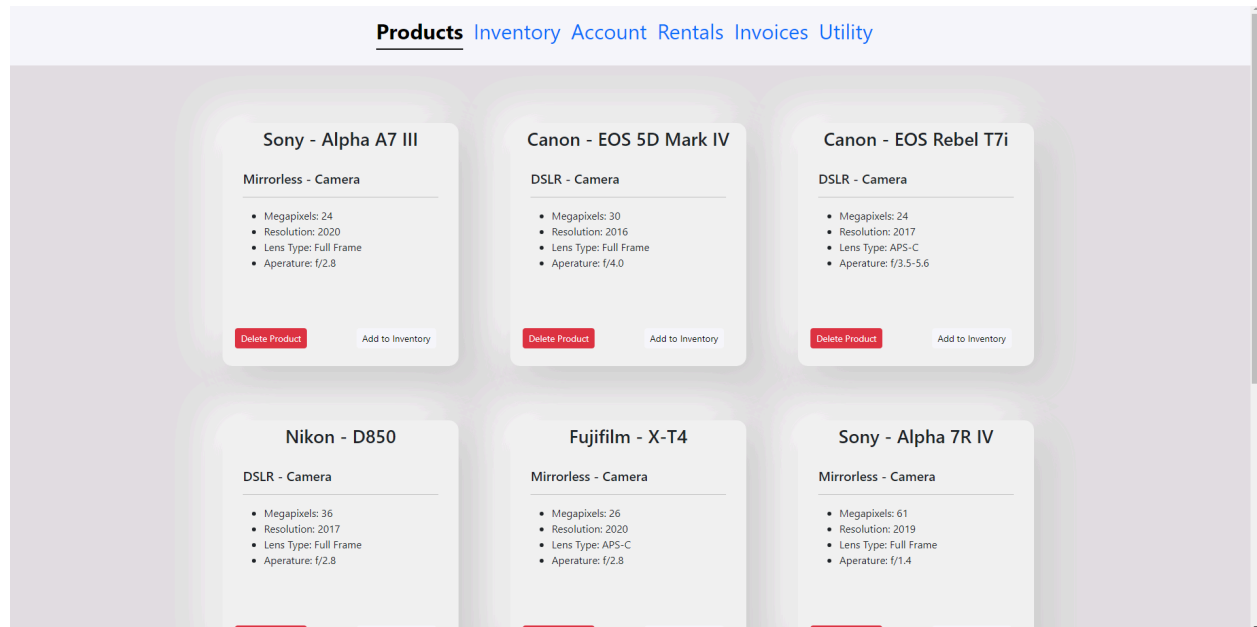
```
 $\sigma_{AVG(InvoiceCost) > 10} (brand F_{AVG(Invoicecost)} (equipment_{equipment.equipmentid} = inventory.itemid (inventory_{(rental_{invoices}))}))$ 
```

## Final UI

### Instructions to run the project:

To run the project, you need to install a node and install the zip for Oracle Instant Client and add it to the root file of the projectT to enable thick mode. Then you need to create a .env file with the login information for the OracleDB and the connection string. Then run npm install, to install all the packages required to run the project. Once that is complete, run npm run-script run and the project should be open in your browser at <http://localhost:3000/>.

Main UI Page for each Table:



Create a new Camera & Equipment:

The screenshot shows a modal form titled "Add Camera to Product Catalogue" with a close button (X). The form contains the following fields:

- Brand
- Name
- Description
- MegaPixels
- Resolution
- Lens Type
- Aperture

At the bottom of the form are two buttons: "Close" and "Create".

Create a new Accessory & Equipment:

Add Accessory to Product Catalogue

×

Brand

Name

Description

Weight (grams)

Color

Accessory Type

Close

Create

Delete a Camera/Accessory:

Delete

×

Are you sure you want to delete the following?

test - 123

Close

Delete

- Meg
- Res
- Len
- Ape

Delete Pro

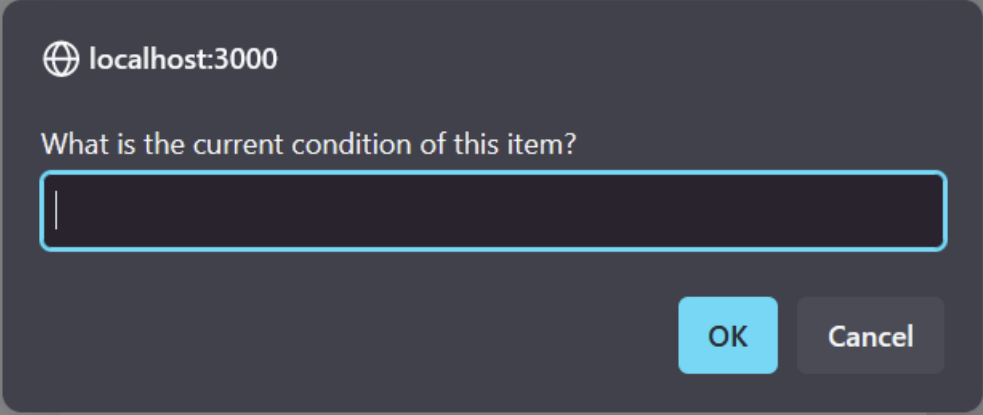
Admin Page to Create/Drop/Populate Tables:



[Drop Tables](#) [Create Tables](#) [Populate Tables](#)

Result:

Edit condition of an inventory item:



The image shows a dark-themed modal dialog box with a title bar containing a globe icon and the text "localhost:3000". The main content area contains the question "What is the current condition of this item?" followed by a large, empty text input field with a light blue border. At the bottom right of the dialog are two buttons: "OK" in a light blue box and "Cancel" in a dark grey box.