1 Stored Procedures

1.1 Stored Procedure 1

```
DELIMITER $$
CREATE PROCEDURE `prReservationRange`
      (IN customernum INTEGER, IN firstdate DATE, IN seconddate DATE,
        OUT reservationquant INTEGER, OUT itemcount INTEGER)
  BEGIN
    DECLARE switch DATE;
    IF firstdate > seconddate THEN
      SET switch = seconddate;
      SET seconddate = firstdate;
      SET firstdate = switch;
    END IF;
    SELECT COUNT(RESERVATION_NUM) INTO reservationquant FROM Reservation
      WHERE Customer_Num = customernum AND
      Reservation_Date BETWEEN firstdate AND seconddate;
    SELECT SUM(QUANTITY) INTO itemcount FROM Reservation AS r
      INNER JOIN Reservation_Item AS i ON (r.Reservation_Num = i.Reservation_Num)
        WHERE CUSTOMER_NUM = customernum AND
        Reservation_Date BETWEEN firstdate AND seconddate;
  END$$
DELIMITER ;
```

1.1.1 example 1

```
CALL `prReservationRange`(1, '2017-03-09', '2017-01-05', @numberofreservations, @numOfItems);
SELECT @numberofreservations, @numOfItems;
```

#	@numberofreservatior	@numOfItems
1	3	7

1.1.2 example 2

```
CALL `prReservationRange`(1, '2017-05-09', '2017-01-05', @numberofreservations, @numOfItems);
SELECT @numberofreservations, @numOfItems;
```

#	@numberofreservatior	@numOfItems
1	3	7

1.2 Stored Procedure 2

```
DELIMITER $$
CREATE PROCEDURE `prRiskAssessment`
      (IN customernum INTEGER, OUT rentalrisk VARCHAR(11))
  BEGIN
    DECLARE reports integer;

    SELECT COUNT(PROBLEM_DESCRIPTION) INTO reports FROM Rental_History
      WHERE Customer_Num = customernum AND
    #We are assuming that there exist entries in rental_history that are NOT problem rentals,
    # and that only those with entries in the problem_description would be problem rentals.
    # If that is not the case, the following line can be removed:
      Problem_Description IS NOT NULL AND
      Rental_Date BETWEEN '2016-01-01' AND '2016-12-31';
    IF reports > 5 THEN
      SET rentalrisk = 'High-Risk';
    ELSEIF reports > 2 AND reports < 6 THEN
```

```
      SET rentalrisk = 'Medium-Risk';
    ELSE
      SET rentalrisk = 'Low-Risk';
    END IF;
  END$$
DELIMITER ;
```

1.2.1 example 1

```
CALL `prRiskAssessment`('3',@risklevel);
SELECT @risklevel;
```

#	@risklevel
1	Medium-Risk

1.2.2 example 1

```
CALL `prRiskAssessment`('1',@risklevel);
SELECT @risklevel;
```

#	@riskleve
1	Low-Risk

1.3 Stored Procedure 3

```
DELIMITER $$
CREATE PROCEDURE isReorderNecessary () #(IN rowID int)
BEGIN #We assume that Reorder_qty is a trigger quantity and that equality triggers a reorder.
  UPDATE Equipment_Type SET Reorder_Necessary = (Inventory_count <= Reorder_qty);
  #To update a specific row add the following:
  #WHERE Equipment_type_code = rowID AND Inventory_count < Reorder_qty;
END $$
DELIMITER ;
```

1.3.1 example 1

```
CALL `isReorderNecessary`();
SELECT * FROM Equipment_Type;
```

#	Equipment_type_code	Equipment_type_description	Equipment_type_rental_charge	Damage_deposit	Inventory_count	Reorder_qty	Reorder_Necessary
1	1	Canoe	35.00	400.00	1	0	0
2	2	Sea Kayak Solo	40.00	600.00	3	2	0
3	3	W.W.Kayak	30.00	400.00	1	0	0
4	4	Sit-On-Top Kayak	30.00	400.00	2	2	1
5	5	Paddle Raft	30.00	400.00	0	1	1
6	6	Oar Raft	60.00	400.00	2	1	0
7	7	Duckie	35.00	400.00	1	0	0
8	8	Sea Kayak Tandem	60.00	600.00	1	0	0

1.3.2 example 2

```
UPDATE Equipment_Type SET Reorder_qty = 2
WHERE Equipment_type_code = 1;

CALL `isReorderNecessary`();
SELECT * FROM Equipment_Type;
```

#	Equipment_type_code	Equipment_type_description	Equipment_type_rental_charge	Damage_deposit	Inventory_count	Reorder_qty	Reorder_Necessary
1	1	Canoe	35.00	400.00	1	2	1
2	2	Sea Kayak Solo	40.00	600.00	3	2	0
3	3	W.W.Kayak	30.00	400.00	1	0	0
4	4	Sit-On-Top Kayak	30.00	400.00	2	2	1
5	5	Paddle Raft	30.00	400.00	0	1	1
6	6	Oar Raft	60.00	400.00	2	1	0
7	7	Duckie	35.00	400.00	1	0	0
8	8	Sea Kayak Tandem	60.00	600.00	1	0	0

2 Full Code Listing

```
DROP DATABASE TetonWhitewater_2;

CREATE DATABASE TetonWhitewater_2;

USE TetonWhitewater_2;

#use backticks    `    around table names that are collisions
######### TABLE DEFINITIONS ##########

CREATE TABLE Customer
(
  Customer_num                INT NOT NULL UNIQUE AUTO_INCREMENT,
  C_name                      VARCHAR(35) NOT NULL,
  C_street                    VARCHAR(40) NOT NULL,
  C_city                      VARCHAR(60) NOT NULL,
  C_state                     VARCHAR(2) NOT NULL,
  C_zip                       INT NOT NULL,
  C_telephone                 VARCHAR(20) NOT NULL,
  C_email                     VARCHAR(40),
  PRIMARY KEY (Customer_num)
);

CREATE TABLE Rental_History
(
  Customer_num                INT NOT NULL,
  Rental_date                 DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  Problem_description         TEXT,
  Amount_owed                 DECIMAL(10,2),
  PRIMARY KEY (Customer_num, Rental_date)
);

CREATE TABLE Reservation
(
  Reservation_num             INT NOT NULL UNIQUE AUTO_INCREMENT,
  Reservation_date            DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  Reservation_pickup_date     DATE NOT NULL,
  Reservation_return_date     DATE NOT NULL,
  Reservation_deposit_amount  DECIMAL(10,2) NOT NULL,
  Customer_num                INT NOT NULL, #FOREIGN KEY
  PRIMARY KEY (Reservation_num)
);

CREATE TABLE Reservation_Item (
  Reservation_num             INT NOT NULL,
  Equipment_type_code         INT NOT NULL,
  Quantity                    INT NOT NULL DEFAULT 0, #Probably do not need the default, but I
    ↪ put it anyway
  PRIMARY KEY(Reservation_num, Equipment_type_code) #Not sure how to handle keys for composite
    ↪ entities
);

CREATE TABLE Rental_Contract
(
  Contract_num                INT NOT NULL UNIQUE AUTO_INCREMENT,
  Pickup_date                 DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  Scheduled_return_date       DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  Actual_return_date          DATETIME,
  Retained_deposit            DECIMAL(10,2),
  Customer_num                INT NOT NULL, #Foreign Key
  PRIMARY KEY (Contract_num)
);

CREATE TABLE Rental_Item (
  Contract_num                INT NOT NULL,
  Equipment_ID                INT NOT NULL,
```

```
  Rental_item_charge           DECIMAL(10,2) NOT NULL,
  Rental_item_deposit_amount   DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (Contract_num, Equipment_ID)
);

CREATE TABLE Equipment
(
  Equipment_ID                 INT UNIQUE AUTO_INCREMENT, #Does this need to be UNIQUE?
  Equipment_Status             VARCHAR(10) NOT NULL,
  #possible make equipment_status enum(NEW,USED,MINOR_DAMAGE,MAJOR_DAMAGE,UNUSABLE)?
  Equipment_rental_count       INT NOT NULL,
  Original_cost                INT NOT NULL,#Do we need this to be NOT NULL #J# I Don't think so
  Equipment_type_code          INT NOT NULL, #Foreign Key
  PRIMARY KEY(Equipment_ID)
);

CREATE TABLE Equipment_Type
(
  Equipment_type_code          INT UNIQUE AUTO_INCREMENT,
  Equipment_type_description   TEXT NOT NULL,#Do we need this to be NOT NULL
  Equipment_type_rental_charge DECIMAL(10,2) NOT NULL,
  Damage_deposit               DECIMAL(10,2) NOT NULL,
  Inventory_count              INT NOT NULL,
  Reorder_qty                  INT UNSIGNED NOT NULL,
  PRIMARY KEY(Equipment_type_code)
);

CREATE TABLE Equipped_With
(
  Equipment_type_code          INT NOT NULL,
  Accessory_code               INT NOT NULL,
  Quantity                     INT NOT NULL,
  PRIMARY KEY(Equipment_type_code,Accessory_code)
);

CREATE TABLE Accessory
(
  Accessory_code               INT UNIQUE AUTO_INCREMENT,
  Accessory_description        TEXT NOT NULL,#Do we need this to be NOT NULL #J#This is the name
    ↪ of the object
  Replacement_cost             DECIMAL(9,2) NOT NULL,
  Accessory_Inventory_Count    INT NOT NULL,
  Accessory_reorder_qty        INT UNSIGNED NOT NULL,
  PRIMARY KEY(Accessory_code)
);

CREATE TABLE `Order` #How do we handle this where it wants to interpret Order as an SQL command?
(
  Order_num                    INT AUTO_INCREMENT,
  Order_date                   DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  Order_date_received          DATETIME,#Null if it has not been received yet
  Supplier_num                 INT NOT NULL, #Foreign Key
  PRIMARY KEY(Order_num)
);

CREATE TABLE Line_Item_Equipment
(
  Ord_num                     INT NOT NULL,
  Equipment_type_code         INT NOT NULL,
  E_order_item_qty            INT UNSIGNED NOT NULL,
  E_order_item_price          DECIMAL(10,2) NOT NULL,
  PRIMARY KEY(Ord_num,Equipment_type_code)
);

CREATE TABLE Line_Item_Accessory
(
  Ord_num                     INT NOT NULL,
  Accessory_code              INT NOT NULL,
```

```
  A_order_item_qty            INT UNSIGNED NOT NULL,
  A_order_item_price          DECIMAL(10,2) NOT NULL,
  PRIMARY KEY(Ord_Num,Accessory_code)
);

CREATE TABLE Supplier
(
  Supplier_num                INT AUTO_INCREMENT,
  S_name                      VARCHAR(60) NOT NULL,
  S_addr                      VARCHAR(255) NOT NULL,
  S_city                      VARCHAR(60) NOT NULL,
  S_state                     VARCHAR(2), #Do we need to specify the length? Why not string? Or
    ↪ since we don't know if it's a 2-letter
  S_zip                       INT NOT NULL,
  S_phone                     VARCHAR(20),
  S_fax                       VARCHAR(20),
  S_contact                   VARCHAR(60) NOT NULL,
  S_email                     VARCHAR(255),
  #Are all of these require or can we make some of them null? #J# I believe most are not required
    ↪ .
  PRIMARY KEY(Supplier_num)
);


############## FOREIGN KEYS ##############

ALTER TABLE Reservation
ADD FOREIGN KEY (CUSTOMER_NUM)
  REFERENCES Customer (CUSTOMER_NUM)
  ON UPDATE CASCADE;


ALTER TABLE Rental_Contract
ADD CONSTRAINT FOREIGN KEY (CUSTOMER_NUM) REFERENCES Customer (CUSTOMER_NUM)
ON UPDATE CASCADE;

ALTER TABLE Equipment
ADD CONSTRAINT FOREIGN KEY (EQUIPMENT_TYPE_CODE)
  REFERENCES Equipment_Type (EQUIPMENT_TYPE_CODE)
  ON UPDATE CASCADE;

ALTER TABLE `Order`
ADD CONSTRAINT FOREIGN KEY (SUPPLIER_NUM)
  REFERENCES Supplier (SUPPLIER_NUM)
  On UPDATE CASCADE;

##From my understanding, we use   On Update Cascade  in all the tables
## where the foreign keys originate from, to cascade to the tables with foreign keys
-- ALTER TABLE Customer
-- ADD(ON UPDATE CASCADE);

-- ALTER TABLE Equipment_Type
-- ADD CONSTRAINT ON UPDATE CASCADE;

-- ALTER TABLE Supplier
-- ADD(ON UPDATE CASCADE);

######### SAMPLE DATA ##########
INSERT INTO Customer (C_name, C_street, C_city, C_state, C_zip, C_telephone, C_email) VALUES
('Jonathon Boden', '7786 New Saddle Drive', 'Ottumwa', 'IA', 52501,'202-555-0119', '
    ↪ alloneword@gmail.com'),
('Sampson Walleye', '1 High Noon Avenue', 'Albany', 'NY', 12203, '(283) 843-9772', '
    ↪ falldownfred@yahoo.com'),
('Walter Smith', '8785 Windfall St.','Whitehall', 'PA', 18052, '(271) 844-9365', '
    ↪ frozenkittyfritters@aol.com'),
('Allonar Blake', '1 N. Cactus Ave.','Green Bay', 'WI', 54302, '(663) 646-4717', '
    ↪ fallenlondon@failbetter.com'),
('Jimmy Dean', '1 High Noon Avenue', 'Albany', 'NY', 12203, '(722) 279-7386', 'meboy@gmail.com');
```

```
INSERT INTO Rental_History (Customer_num, Rental_date, Problem_description, Amount_owed) VALUES
(3,'2016-02-01','Damaged',35.00),
(3,'2016-02-07','Damaged',35.00),
(2,'2017-03-14',NULL,210.00),
(4,'2017-03-29',NULL,360.00),
(3,'2016-04-05','Damaged',35.00);

INSERT INTO Reservation (Reservation_date, Reservation_pickup_date, Reservation_return_date,
    ↪ Reservation_deposit_amount, Customer_num) VALUES
('2017-01-05', '2017-01-28', '2017-02-02', 500.00, 1),
('2017-02-07', '2017-02-09', '2017-02-13', 275.00, 1),
('2017-02-17', '2017-03-14','2017-03-14', 400.00, 2),
('2017-03-07', '2017-03-29','2017-04-10', 350.00, 3),
('2017-03-09', '2017-04-05', '2017-04-06', 425.00, 4),
('2017-03-09', '2017-04-05', '2017-04-06', 425.00, 1),
('2017-03-08', '2017-04-06', '2017-04-07', 425.00, 5);

INSERT INTO Reservation_Item (Reservation_num, Equipment_type_code, Quantity) VALUES
(1,1,1),
(2,2,1),
(3,1,1),
(4,4,1),
(5,3,1),
(1,2,5);

INSERT INTO Rental_Contract (Pickup_date, Scheduled_return_date, Actual_return_date,
    ↪ Retained_deposit, Customer_num) VALUES
('2017-02-01T13:10:00', '2017-02-02T13:09:59', '2017-02-02T9:11:00', 0, 1),
('2017-02-09T8:01:00', '2017-02-13T8:00:59', null, null, 1), #purposely included null values as
    ↪ overdue
('2017-03-14T15:25:00','2017-03-14T15:24:59', '2017-03-15T8:10:00', 200.00, 2),
('2017-03-29T16:45:00','2017-04-10T16:44:59', '2017-04-10T16:45:01', 175.00, 3),
('2017-04-05T9:12:00', '2017-04-06T9:11:59', '2017-04-06T9:11:59', 50, 4),
('2017-05-05T9:12:00', '2017-05-06T9:11:59', '2017-05-06T9:11:59', 100, 1),
('2017-05-05T9:12:00', '2017-05-06T9:11:59', '2017-05-06T9:11:59', 100, 4);


INSERT INTO Rental_Item (Contract_num, Equipment_ID, Rental_item_charge,
    ↪ Rental_item_deposit_amount) VALUES
(1,1,35.00,400.00),
(2,2,210.00,40.00),
(3,1,35.00,400.00),
(4,4,360.00,400.00),
(5,3,35.00,400.00),
(6,1,35.00,400.00),
(7,7,35.00,400.00);

INSERT INTO Equipment_Type (Equipment_type_description, Equipment_type_rental_charge,
    ↪ Damage_deposit, Inventory_count, Reorder_qty) VALUES
('Canoe', 35.00, 400.00, 1, 0),
('Sea Kayak Solo', 40.00, 600.00, 3, 2),
('W.W.Kayak', 30.00, 400.00, 1, 0),
('Sit-On-Top Kayak', 30.00, 400.00, 2, 2),
('Paddle Raft', 30.00, 400.00, 0, 1),
('Oar Raft', 60.00, 400.00, 2, 1),
('Duckie', 35.00, 400.00, 1, 0),
('Sea Kayak Tandem', 60.00, 600.00, 1, 0);

INSERT INTO Equipment (Equipment_status, Equipment_rental_count, Original_cost,
    ↪ Equipment_type_code) VALUES
('In Stock', 3, 800.00, 1),
('rented', 7, 1200.00, 2),
('IN STOCK', 1, 800.00, 1),
('In stock', 15, 1200, 3), #purposely included no decimals on price
('sold', 15, 1200, 7),
('Damaged', 3, 1200, 4),
('Rented', 3, 1200, 2);
```

```
INSERT INTO Equipped_With (Equipment_type_code, Accessory_code, Quantity) VALUES
(1,1,2),
(1,2,1),
(2,1,1),
(2,2,1),
(3,2,1);
#this table coould do with a lot more entries

INSERT INTO Accessory (Accessory_description, Replacement_cost, Accessory_inventory_count,
    ↪ Accessory_reorder_qty) VALUES
('Lifejacket', 35.00, 40, 20),
('Paddle', 40.00, 60, 30),
('Roof Pad', 30.00, 4, 2),
('Tie down', 3.00, 40, 25),
('Bail Bucket', 5.00, 40, 10),
('Helmet', 6.00, 60, 30);

INSERT INTO Supplier (S_name,S_addr,S_city,S_State, S_zip,S_phone,S_fax,S_contact,S_email) VALUES
("Dave's Kayak Factory","4586 Sage St.","Kansas City","NB",96587,"(923) 546-9874","(813) 825-1254
    ↪ ","Dave Davidson","dave@supermail.net"),
("The Whitewater Emporium","1235 Smith St.","Nantucket","TN",64852,"(325) 625-6454","(642)
    ↪ 363-4141","Will Williamson","Will@gmail.com"),
("Fat Pigeon Products","879 SE. Cherry Hill Dr.","Holbrook","NY",11741,"(271) 844-9365","(271)
    ↪ 844-9365x63","Kama Jozafat","jokama@fatpidgeon.com"),
("Alpha Moose Kayak","9025 University St.","Emporia","KS",64852,"(663) 646-4717",NULL,"Surya
    ↪ Torsten","Torsten@amooseyak.com"), #Contact number left null
("Turtle Shoe Rafts","70 Glenholme Drive","Bozeman","MT",96587,"(457) 213-9438","(457) 825-1254",
    ↪ "Carlos Genadi","Carlos@TurtleStew.net");

INSERT INTO `Order` (Order_date,Order_date_received,Supplier_num) VALUES
('2014-12-05','2014-12-13',3),
('2015-05-10','2015-05-20',3),
('2016-09-07','2016-09-08',1),
('2017-12-12',NULL,4),
('2017-11-18',NULL,2);

INSERT INTO Line_Item_Equipment (Ord_num, Equipment_type_code, E_order_item_qty,
    ↪ E_order_item_price) VALUES
(1,2,5,4000),
(2,2,3,3600),
(3,5,1,1200),
(4,5,1,1200),
(5,1,2,1600);

INSERT INTO Line_Item_Accessory (Ord_num, Accessory_code, A_order_item_qty, A_order_item_price)
    ↪ VALUES
(1,2,10,400),
(2,1,20,700),
(3,4,15,45),
(1,3,5,150),
(2,6,8,54);

Select * From Accessory;
Select * From Customer;
Select * From Equipment;
Select * From Equipment_Type;
Select * From Equipped_With;
Select * From Line_Item_Accessory;
Select * From Line_Item_Equipment;
Select * From `Order`;
Select * From Rental_Contract;
Select * From Rental_History;
Select * From Rental_Item;
Select * From Reservation;
Select * From Reservation_Item;
Select * From Supplier;
```

```
#1  List the equipment ID, due date, and status for all equipment that is currently rented.
SELECT eq.Equipment_ID, rc.Scheduled_return_date AS 'Due Date/Time', eq.Equipment_Status as '
    ↪ Status'
  From Rental_Item AS ri INNER JOIN Equipment AS eq
    ON (eq.Equipment_ID = ri.Equipment_ID)
  INNER JOIN Rental_Contract AS rc
    ON (ri.Contract_num = rc.Contract_num)
  WHERE equipment_status LIKE 'rented';


#2  List all equipment types whose description ends with "kayak."
SELECT Equipment_type_Code, Equipment_type_Description AS 'Description'
  FROM Equipment_Type
    WHERE Equipment_type_Description LIKE '%kayak';

#3  List the customer name, equipment type, equipment id, and scheduled return date for all
    ↪ equipment rented (picked up) on February 9, 2017.
SELECT C.C_Name, et.Equipment_Type_Description, eq.Equipment_ID, rc.Scheduled_Return_Date
  FROM Rental_Contract AS rc INNER JOIN Customer AS C
    ON C.Customer_Num = rc.Customer_Num
  INNER JOIN Rental_Item AS ri
    ON (rc.Contract_Num = ri.Contract_Num)
  INNER JOIN Equipment AS eq
    ON (ri.Equipment_ID = eq.Equipment_ID)
  INNER JOIN Equipment_Type AS et
    ON (eq.equipment_type_code = et.Equipment_type_code)
  WHERE rc.Pickup_date LIKE '2017-02-09%';

#4  List all rental incident reports (customer id, customer name, date, problem description,
    ↪ amount owed) associated with a specified customer.
SELECT *
  FROM Rental_History
    WHERE Customer_num = 1; ############## This number used as sample for 'specified customer',
    ↪ this will eventually be a stored procedure

#5  a) List each equipment type and the total number of rentals from that category.
Select et.Equipment_type_description, SUM(eq.Equipment_rental_count) AS 'Total Rental Count'
  From Equipment_Type AS et INNER JOIN Equipment AS eq
    ON (et.Equipment_Type_Code = eq.Equipment_Type_Code)
  GROUP BY et.EQUIPMENT_TYPE_CODE;

#5 b) List the type of equipment and total number of rentals for the equipment type that was
    ↪ rented most often.

SELECT et.Equipment_type_description AS DESCRIPTION, SUM(eq.Equipment_rental_count) AS `Most Used
    ↪ `
  From Equipment_Type AS et INNER JOIN Equipment AS eq
    ON (et.Equipment_Type_Code = eq.Equipment_Type_Code)
  GROUP BY et.Equipment_Type_Code
  HAVING `Most Used` >= ALL (SELECT SUM(eq.Equipment_rental_count)
                From Equipment_Type AS et INNER JOIN Equipment AS eq
                ON (et.Equipment_Type_Code = eq.Equipment_Type_Code)
                GROUP BY et.Equipment_Type_Code);


#6 List all equipment types and number of suppliers for those equipment types
#   that were supplied by multiple suppliers.
SELECT Equipment_Type.Equipment_Type_Description AS 'Equipment Type',
    COUNT(DISTINCT `Order`.Supplier_num) AS 'Number of Different Suppliers'
    FROM Equipment_Type INNER JOIN Line_Item_Equipment
      ON (Equipment_Type.Equipment_type_code = Line_Item_Equipment.Equipment_type_code)
    INNER JOIN `Order`
      ON (Line_Item_Equipment.Ord_num = `Order`.Order_num)
    GROUP BY Equipment_Type.Equipment_Type_Code
      HAVING COUNT(DISTINCT `Order`.Supplier_Num) > 1;

#7 Given an equipment type (e.g., Canoe), indicate the description and the number of items that
```

```
#    are available to be rented (in stock).

SELECT Equipment_Type.Equipment_type_description AS 'EQUIPMENT TYPE',
      COUNT(Equipment.Equipment_Status) AS 'Number In Stock'
  FROM Equipment_Type INNER JOIN Equipment ON (Equipment_Type.Equipment_Type_Code = Equipment.
    ↪ Equipment_Type_Code)
      WHERE Equipment.Equipment_Status LIKE 'In%'
      GROUP BY Equipment_Type.Equipment_Type_Description
        HAVING Equipment_Type.Equipment_Type_Description LIKE 'CANOE'; #THIS IS ADDED BECAUSE OF
    ↪ THE 'GIVEN A TYPE', EVENTUALLY THIS WILL BE A VARIABLE PASSED TO A STORED PROCEDURE.

#8 List the equipment id, equipment type, and status of all rentals that are overdue.
SELECT et.Equipment_type_code, et.Equipment_type_description, e.Equipment_Status FROM
    ↪ Rental_Contract AS rc
INNER JOIN Rental_Item AS ri ON (rc.Contract_num = ri.Contract_num)
INNER JOIN Equipment AS e ON (ri.Equipment_ID = e.Equipment_ID)
INNER JOIN Equipment_Type AS et ON (et.Equipment_type_code = e.Equipment_type_code)
WHERE rc.Scheduled_return_date < CURRENT_DATE AND rc.Actual_return_date IS NULL;
#Note, that this search looks for rentals that are overdue, not just those labeled as "Overdue".


#9 List the total amount owed for all incident reports in the rental history table
SELECT SUM(Amount_owed) FROM Rental_History;

# 10 Count the number of reservations that are scheduled to be picked up during the weekend of
    ↪ January 28 & 29, 2017
SELECT COUNT(Reservation_num) FROM Reservation
WHERE Reservation_pickup_date = '2017-1-28' OR Reservation_pickup_date = '2017-1-29';

# 11 Show the equipment type id, equipment type description, and average days rented for all
    ↪ types of equipment on a type-by-type basis
SELECT et.Equipment_type_code, et.Equipment_type_description, ROUND(AVG(DATEDIFF(rc.
    ↪ Actual_return_date,rc.Pickup_date)),1) as 'Average Days Rented'
FROM Rental_Contract AS rc
INNER JOIN Rental_Item AS ri ON (rc.Contract_num = ri.Contract_num)
INNER JOIN Equipment AS e ON (ri.Equipment_ID = e.Equipment_ID)
INNER JOIN Equipment_Type AS et ON (et.Equipment_type_code = e.Equipment_type_code)
GROUP BY et.Equipment_type_code;
#Using Actual_return_date instead of Scheduled return date ignores any items not yet returned,
#but provides a more accurate estimate of days rented, as opposed to days planned rented.

#12 List supplier number and supplier name for all suppliers for which there are no current
    ↪ orders. Sort the list in ascending order by supplier name.
SELECT s.Supplier_num, s.S_name FROM Supplier AS s
LEFT JOIN `Order` as o ON (o.Supplier_num = s.Supplier_num)
WHERE o.Order_num IS NULL
ORDER BY s.S_name ASC;
#This is assuming that the phrase 'No Current Orders' implies there is no record of an order
    ↪ being made


#13 - (Needs GROUP BY and HAVING) "Write a query that provides the number of times a deposit was
    ↪ withheld, the total amount withheld, and the associated customer number for each customer
    ↪ that has had a deposit withheld more than once."
SELECT Customer_Num, COUNT(Retained_Deposit)  AS 'Number of Times Withheld', SUM(Retained_deposit
    ↪ ) AS 'Total of Withholdings'
  FROM Rental_Contract
    WHERE Retained_Deposit > 0
  GROUP BY Customer_Num
    HAVING COUNT(Retained_Deposit) > 1;

#14 - (Join AT LEAST three tables) "Write a query that lists each type of equipment that has been
    ↪  ordered with the name of the suppliers whom we have ordered that product from in the past
    ↪ .
SELECT DISTINCT Equipment_Type.Equipment_Type_Description AS 'Equipment Type', Supplier.S_name AS
    ↪  'Suppliers'
  FROM Equipment_Type INNER JOIN Line_Item_Equipment USING (Equipment_Type_Code)
            INNER JOIN `Order` ON (Line_Item_Equipment.Ord_num = `Order`.Order_Num)
```

```
            INNER JOIN Supplier USING (Supplier_num)
      ORDER BY Equipment_Type.Equipment_Type_Code;

#15 - (Outer Join) "Write a query that lists the customer number and reservation number for all
    ↪ reservations that do not have a matching rental on the declared pickup date, implying that
    ↪  they were not picked up as planned"
SELECT Reservation.Reservation_num AS 'Reservation #',
    Reservation.Customer_num AS 'Made By Customer #',
    Reservation.Reservation_Pickup_Date AS 'Was Not Picked Up On'
       FROM Rental_Contract INNER JOIN
        (SELECT DATE(Pickup.Pickup_Date) AS 'DATEONLY', Pickup.Contract_Num AS 'ID'
          FROM Rental_Contract AS Pickup) Pickup
            ON Rental_Contract.Contract_Num = Pickup.ID
      RIGHT JOIN Reservation
          ON (Rental_Contract.Customer_Num = Reservation.Customer_num
            AND Reservation.Reservation_Pickup_Date = Pickup.Dateonly)
    WHERE Dateonly IS NULL
    ORDER BY Reservation.Customer_num;

#STORED PROCEDURES
#1 Write a stored procedure that takes a customer number as well as two dates as input
# parameters and returns two numbers as output parameters: 1) number of reservations
# made by that customer between those two dates (both inclusive) and 2) total number of
# items on those reservations.
# your procedure should first check the two date values received from the user to determine
# which one is greater than the other one and decide on which one should be considered
# the start date of the period.
# Also, write the code to test the procedure



DELIMITER $$
CREATE PROCEDURE `prReservationRange`
      (IN customernum INTEGER, IN firstdate DATE, IN seconddate DATE,
        OUT reservationquant INTEGER, OUT itemcount INTEGER)
  BEGIN
    DECLARE switch DATE;
    IF firstdate > seconddate THEN
      SET switch = seconddate;
      SET seconddate = firstdate;
      SET firstdate = switch;
    END IF;
    SELECT COUNT(RESERVATION_NUM) INTO reservationquant FROM Reservation
      WHERE Customer_Num = customernum AND
      Reservation_Date BETWEEN firstdate AND seconddate;
    SELECT SUM(QUANTITY) INTO itemcount FROM Reservation AS r
      INNER JOIN Reservation_Item AS i ON (r.Reservation_Num = i.Reservation_Num)
        WHERE CUSTOMER_NUM = customernum AND
        Reservation_Date BETWEEN firstdate AND seconddate;
  END$$
DELIMITER ;

CALL `prReservationRange`(1, '2017-03-09', '2017-01-05', @numberofreservations, @numOfItems);
SELECT @numberofreservations, @numOfItems;

CALL `prReservationRange`(1, '2017-05-09', '2017-01-05', @numberofreservations, @numOfItems);
SELECT @numberofreservations, @numOfItems;

#2 Write a stored procedure that takes a customer number as the input parameter and returns a
# message (as the output parameter) that indicates whether the customer is  High-risk ,
    ↪  Mediumrisk
# or  Low-risk . Also, write the code to test the procedure.

DELIMITER $$
CREATE PROCEDURE `prRiskAssessment`
      (IN customernum INTEGER, OUT rentalrisk VARCHAR(11))
  BEGIN
    DECLARE reports integer;
```

```
    SELECT COUNT(PROBLEM_DESCRIPTION) INTO reports FROM Rental_History
      WHERE Customer_Num = customernum AND
    #We are assuming that there exist entries in rental_history that are NOT problem rentals,
    # and that only those with entries in the problem_description would be problem rentals.
    # If that is not the case, the following line can be removed:
      Problem_Description IS NOT NULL AND
      Rental_Date BETWEEN '2016-01-01' AND '2016-12-31';
    IF reports > 5 THEN
      SET rentalrisk = 'High-Risk';
    ELSEIF reports > 2 AND reports < 6 THEN
      SET rentalrisk = 'Medium-Risk';
    ELSE
      SET rentalrisk = 'Low-Risk';
    END IF;
  END$$
DELIMITER ;

CALL `prRiskAssessment`('3',@risklevel);
SELECT @risklevel;

CALL `prRiskAssessment`('1',@risklevel);
SELECT @risklevel;
#3 Modify the Equipment_Type table to include a new Boolean attribute called
# Reorder_necessary that will be set to true when an item needs to be reordered. The
# attribute should default to false.

# Since the default value may not be valid for all the data in your table, write a stored
# procedure to set the value of the new Reorder_necessary attribute to its correct value.
# If the Inventory_count is less than or equal to the Reorder_qty, then set the
# Reorder_necessary value to true.
# Otherwise, set the Reorder_necessary value to false.
# Execute the procedure to reset the value.

ALTER TABLE Equipment_Type
ADD COLUMN Reorder_Necessary bool default false; #Because it is a boolean, values are 0 or 1

DELIMITER $$
CREATE PROCEDURE isReorderNecessary () #(IN rowID int)
BEGIN #We assume that Reorder_qty is a trigger quantity and that equality triggers a reorder.
  UPDATE Equipment_Type SET Reorder_Necessary = (Inventory_count <= Reorder_qty);
  #To update a specific row add the following:
  #WHERE Equipment_type_code = rowID AND Inventory_count < Reorder_qty;
END $$
DELIMITER ;

CALL `isReorderNecessary`();
SELECT * FROM Equipment_Type;

UPDATE Equipment_Type SET Reorder_qty = 2
WHERE Equipment_type_code = 1;

CALL `isReorderNecessary`();
SELECT * FROM Equipment_Type;
```