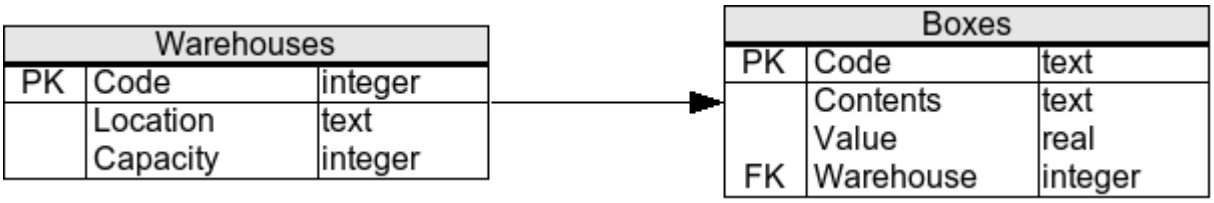
DBI202: LAB 5

Exercise 1:



1. Select all warehouses.
2. Select all boxes with a value larger than $150.
3. Select all distinct contents in all the boxes.
4. Select the average value of all the boxes.
5. Select the warehouse code and the average value of the boxes in each warehouse.
6. Same as previous exercise, but select only those warehouses where the average value of the boxes is greater than 150.
7. Select the code of each box, along with the name of the city the box is located in.
8. Select the warehouse codes, along with the number of boxes in each warehouse. Optionally, take into account that some warehouses are empty (i.e., the box count should show up as zero, instead of omitting the warehouse from the result).
9. Select the codes of all warehouses that are saturated (a warehouse is saturated if the number of boxes in it is larger than the warehouse's capacity).
10. Select the codes of all the boxes located in Chicago.
11. Create a new warehouse in New York with a capacity for 3 boxes.
12. Create a new box, with code "H5RT", containing "Papers" with a value of $200, and located in warehouse 2.
13. Reduce the value of all boxes by 15%.
14. Apply a 20% value reduction to boxes with a value larger than the average value of all the boxes.
15. Remove all boxes with a value lower than $100.

Solution Exercise 1

```
CREATE TABLE Warehouses (
   Code INTEGER PRIMARY KEY NOT NULL,
   Location TEXT NOT NULL ,
   Capacity INTEGER NOT NULL
 );

 CREATE TABLE Boxes (
   Code TEXT PRIMARY KEY NOT NULL,
   Contents TEXT NOT NULL ,
   Value REAL NOT NULL ,
   Warehouse INTEGER NOT NULL,
```

```
   CONSTRAINT fk_Warehouses_Code FOREIGN KEY (Warehouse) REFERENCES
Warehouses(Code)
 );
```

```
INSERT INTO Warehouses(Code,Location,Capacity) VALUES(1,'Chicago',3);
 INSERT INTO Warehouses(Code,Location,Capacity) VALUES(2,'Chicago',4);
 INSERT INTO Warehouses(Code,Location,Capacity) VALUES(3,'New York',7);
 INSERT INTO Warehouses(Code,Location,Capacity) VALUES(4,'Los Angeles',2);
 INSERT INTO Warehouses(Code,Location,Capacity) VALUES(5,'San
Francisco',8);

 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('0MN7','Rocks',180,3);
 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('4H8P','Rocks',250,1);
 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('4RT3','Scissors',190,4);
 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('7G3H','Rocks',200,1);
 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('8JN6','Papers',75,1);
 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('8Y6U','Papers',50,3);
 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('9J6F','Papers',175,2);
 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('LL08','Rocks',140,4);
 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('P0H6','Scissors',125,1);
 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('P2T6','Scissors',150,2);
 INSERT INTO Boxes(Code,Contents,Value,Warehouse)
VALUES('TU55','Papers',90,5);
```

1. Select all warehouses.

```
SELECT * FROM Warehouses;
```

2. Select all boxes with a value larger than $150.

```
SELECT * FROM Boxes
WHERE Value > 150;
```

3. Select all distinct contents in all the boxes.

```
SELECT DISTINCT Contents
FROM Boxes;
```

4. Select the average value of all the boxes.

```
SELECT AVG(Value)
FROM Boxes;
```

5. Select the warehouse code and the average value of the boxes in each warehouse.

```
SELECT Warehouse, AVG(Value)
FROM Boxes
GROUP BY Warehouse;
```

6. Same as previous exercise, but select only those warehouses where the average value of the boxes is greater than 150.

```
SELECT Warehouse, AVG(Value)
FROM Boxes
GROUP BY Warehouse
HAVING AVG(Value) > 150;
```

7. Select the code of each box, along with the name of the city the box is located in.

```
SELECT Boxes.Code, Location
FROM Warehouses INNER JOIN Boxes
ON Warehouses.Code = Boxes.Warehouse;
```

8. Select the warehouse codes, along with the number of boxes in each warehouse. Optionally, take into account that some warehouses are empty (i.e., the box count should show up as zero, instead of omitting the warehouse from the result).

```
/* Not taking into account empty warehouses */
SELECT Warehouse, COUNT(*)
FROM Boxes
GROUP BY Warehouse;
/* Taking into account empty warehouses */
SELECT Warehouses.Code, COUNT(Boxes.Code)
FROM Warehouses LEFT JOIN Boxes
```

```
ON Warehouses.Code = Boxes.Warehouse
GROUP BY Warehouses.Code;
```

9. Select the codes of all warehouses that are saturated (a warehouse is saturated if the number of boxes in it is larger than the warehouse's capacity).

```
SELECT Code
FROM Warehouses
WHERE Capacity <
  (SELECT COUNT(*)
   FROM Boxes
   WHERE Warehouse = Warehouses.Code);
/* Alternate method not involving nested statements */
SELECT Warehouses.Code
FROM Warehouses JOIN Boxes ON Warehouses.Code = Boxes.Warehouse
GROUP BY Warehouses.code, Warehouses.Capacity
HAVING Count(Boxes.code) > Warehouses.Capacity
```

10. Select the codes of all the boxes located in Chicago.

```
/* Without subqueries */
SELECT Boxes.Code
FROM Warehouses RIGHT JOIN Boxes
ON Warehouses.Code = Boxes.Warehouse
WHERE Location = 'Chicago';
/* With a subquery */
SELECT Code
FROM Boxes
WHERE Warehouse IN
   (SELECT Code
    FROM Warehouses
    WHERE Location = 'Chicago');
```

11. Create a new warehouse in New York with a capacity for 3 boxes.

```
INSERT INTO Warehouses(Location,Capacity)VALUES ('New York',3);
```

12. Create a new box, with code "H5RT", containing "Papers" with a value of $200, and located in warehouse 2.

```
INSERT INTO Boxes VALUES('H5RT','Papers',200,2);
```

13. Reduce the value of all boxes by 15%.

```
UPDATE Boxes SET Value = Value * 0.85;
```

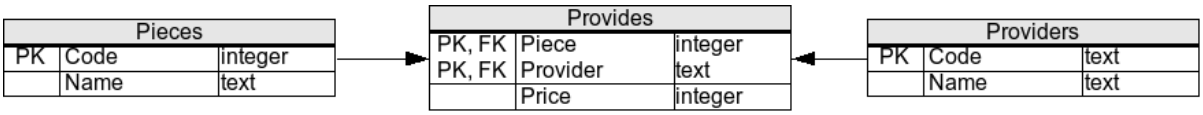
14. Apply a 20% value reduction to boxes with a value larger than the average value of all the boxes.

```
UPDATE Boxes SET Value = Value * 0.80
WHERE Value > (SELECT AVG(Value) FROM (SELECT * FROM Boxes) AS X);
```

15. Remove all boxes with a value lower than $100.

```
DELETE FROM Boxes WHERE Value < 100;
```

Exercise 2:



1. Select the name of all the pieces.
2. Select all the providers' data.
3. Obtain the average price of each piece (show only the piece code and the average price).
4. Obtain the names of all providers who supply piece 1.
5. Select the name of pieces provided by provider with code "HAL".
6. For each piece, find the most expensive offering of that piece and include the piece name, provider name, and price (note that there could be two providers who supply the same piece at the most expensive price).
7. Add an entry to the database to indicate that "Skellington Supplies" (code "TNBC") will provide sprockets (code "1") for 7 cents each.
8. Increase all prices by one cent.
9. Update the database to reflect that "Susan Calvin Corp." (code "RBT") will not supply bolts (code 4).

10. Update the database to reflect that "Susan Calvin Corp." (code "RBT") will not supply any pieces (the provider should still remain in the database).

```
CREATE TABLE Pieces (
   Code INTEGER PRIMARY KEY NOT NULL,
   Name TEXT NOT NULL
 );

 CREATE TABLE Providers (
  Code TEXT PRIMARY KEY NOT NULL,
  Name TEXT NOT NULL
 );

 CREATE TABLE Provides (
   Piece INTEGER
     CONSTRAINT fk_Pieces_Code REFERENCES Pieces(Code),
   Provider TEXT
     CONSTRAINT fk_Providers_Code REFERENCES Providers(Code),
   Price INTEGER NOT NULL,
   PRIMARY KEY(Piece, Provider)
 );
```

```
INSERT INTO Providers(Code, Name) VALUES('HAL','Clarke Enterprises');
 INSERT INTO Providers(Code, Name) VALUES('RBT','Susan Calvin Corp.');
 INSERT INTO Providers(Code, Name) VALUES('TNBC','Skellington Supplies');

 INSERT INTO Pieces(Code, Name) VALUES(1,'Sprocket');
 INSERT INTO Pieces(Code, Name) VALUES(2,'Screw');
 INSERT INTO Pieces(Code, Name) VALUES(3,'Nut');
 INSERT INTO Pieces(Code, Name) VALUES(4,'Bolt');

 INSERT INTO Provides(Piece, Provider, Price) VALUES(1,'HAL',10);
 INSERT INTO Provides(Piece, Provider, Price) VALUES(1,'RBT',15);
 INSERT INTO Provides(Piece, Provider, Price) VALUES(2,'HAL',20);
 INSERT INTO Provides(Piece, Provider, Price) VALUES(2,'RBT',15);
 INSERT INTO Provides(Piece, Provider, Price) VALUES(2,'TNBC',14);
 INSERT INTO Provides(Piece, Provider, Price) VALUES(3,'RBT',50);
 INSERT INTO Provides(Piece, Provider, Price) VALUES(3,'TNBC',45);
 INSERT INTO Provides(Piece, Provider, Price) VALUES(4,'HAL',5);
 INSERT INTO Provides(Piece, Provider, Price) VALUES(4,'RBT',7);
```

1. Select the name of all the pieces.

```
SELECT Name FROM Pieces;
```

2. Select all the providers' data.

```sql
SELECT * FROM Providers;
```

3. Obtain the average price of each piece (show only the piece code and the average price).

```sql
SELECT Piece, AVG(Price)
FROM Provides
GROUP BY Piece;
```

4. Obtain the names of all providers who supply piece 1.

```sql
/* Without subquery */
SELECT Providers.Name
FROM Providers INNER JOIN Provides
ON Providers.Code = Provides.Provider
AND Provides.Piece = 1;
/* With subquery */
SELECT Name
FROM Providers
WHERE Code IN
(SELECT Provider FROM Provides WHERE Piece = 1);
```

5. Select the name of pieces provided by provider with code "HAL".

```sql
/* Without subquery */
SELECT Pieces.Name
FROM Pieces INNER JOIN Provides
ON Pieces.Code = Provides.Piece
AND Provides.Provider = 'HAL';
 /* With IN subquery */
SELECT Name
FROM Pieces
WHERE Code IN
(SELECT Piece FROM Provides WHERE Provider = 'HAL');
/* With EXISTS subquery */
SELECT Name
FROM Pieces
WHERE EXISTS
(
SELECT * FROM Provides
WHERE Provider = 'HAL'
AND Piece = Pieces.Code);
```

6. For each piece, find the most expensive offering of that piece and include the piece name, provider name, and price (note that there could be two providers who supply the same piece at the most expensive price).

```
SELECT Pieces.Name, Providers.Name, Price
FROM Pieces INNER JOIN Provides ON Pieces.Code = Piece
               INNER JOIN Providers ON Providers.Code = Provider
WHERE Price =
(
SELECT MAX(Price) FROM Provides
WHERE Piece = Pieces.Code
);
```

7. Add an entry to the database to indicate that "Skellington Supplies" (code "TNBC") will provide sprockets (code "1") for 7 cents each.

```
INSERT INTO Provides VALUES (1, 'TNBC', 7);
```

8. Increase all prices by one cent.

```
UPDATE Provides SET Price = Price + 1;
```

9. Update the database to reflect that "Susan Calvin Corp." (code "RBT") will not supply bolts (code 4).

```
DELETE FROM Provides
WHERE Provider = 'RBT'
AND Piece = 4;
```

10. Update the database to reflect that "Susan Calvin Corp." (code "RBT") will not supply any pieces (the provider should still remain in the database).

```
DELETE FROM Provides
WHERE Provider = 'RBT';
```

Exercise 3: