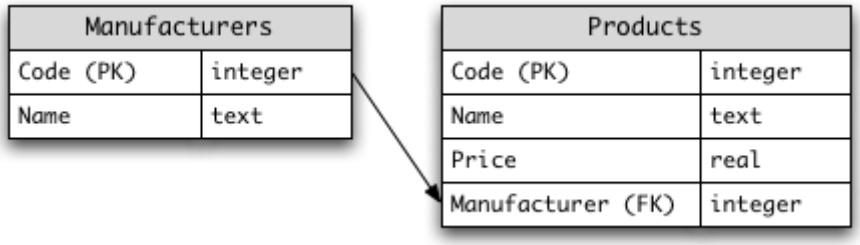
Task 1



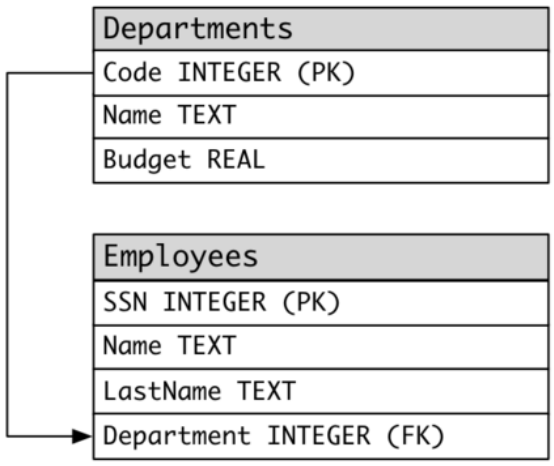

```
Create tables
CREATE TABLE IF NOT EXISTS Manufacturers (
  Code INTEGER,
  Name TEXT NOT NULL,
  PRIMARY KEY (Code)
);

CREATE TABLE IF NOT EXISTS Products (
  Code INTEGER,
  Name TEXT NOT NULL,
  Price INTEGER NOT NULL,
  Manufacturer INTEGER NOT NULL,
  PRIMARY KEY (Code)
);

INSERT INTO Manufacturers(Code,Name) VALUES(1,'Sony');
INSERT INTO Manufacturers(Code,Name) VALUES(2,'Creative Labs');
INSERT INTO Manufacturers(Code,Name) VALUES(3,'Hewlett-Packard');
INSERT INTO Manufacturers(Code,Name) VALUES(4,'Iomega');
INSERT INTO Manufacturers(Code,Name) VALUES(5,'Fujitsu');
INSERT INTO Manufacturers(Code,Name) VALUES(6,'Winchester');

INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(1,'Hard drive',240,5);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(2,'Memory',120,6);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(3,'ZIP drive',150,4);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(4,'Floppy disk',5,6);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(5,'Monitor',240,1);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(6,'DVD drive',180,2);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(7,'CD drive',90,2);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(8,'Printer',270,3);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(9,'Toner cartridge',66,3);
INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(10,'DVD burner',180,2);
```

Questions

```
-- 1.1 Select the names of all the products in the store.
-- 1.2 Select the names and the prices of all the products in the store.
-- 1.3 Select the name of the products with a price less than or equal to $200.
-- 1.4 Select all the products with a price between $60 and $120.
-- 1.5 Select the name and price in cents (i.e., the price must be multiplied
by 100).
-- 1.6 Compute the average price of all the products.
-- 1.7 Compute the average price of all products with manufacturer code equal
to 2.
```

```
-- 1.8 Compute the number of products with a price larger than or equal to
$180.
-- 1.9 Select the name and price of all products with a price larger than or
equal to $180, and sort first by price (in descending order), and then by name
(in ascending order).
-- 1.10 Select all the data from the products, including all the data for each
product's manufacturer.
-- 1.11 Select the product name, price, and manufacturer name of all the
products.
-- 1.12 Select the average price of each manufacturer's products, showing only
the manufacturer's code.
-- 1.13 Select the average price of each manufacturer's products, showing the
manufacturer's name.
-- 1.14 Select the names of manufacturer whose products have an average price
larger than or equal to $150.
-- 1.15 Select the name and price of the cheapest product.
-- 1.16 Select the name of each manufacturer along with the name and price of
its most expensive product.
-- 1.17 Add a new product: Loudspeakers, $70, manufacturer 2.
-- 1.18 Update the name of product 8 to "Laser Printer".
-- 1.19 Apply a 10% discount to all products.
-- 1.20 Apply a 10% discount to all products with a price larger than or equal
to $120.
```

Task 2



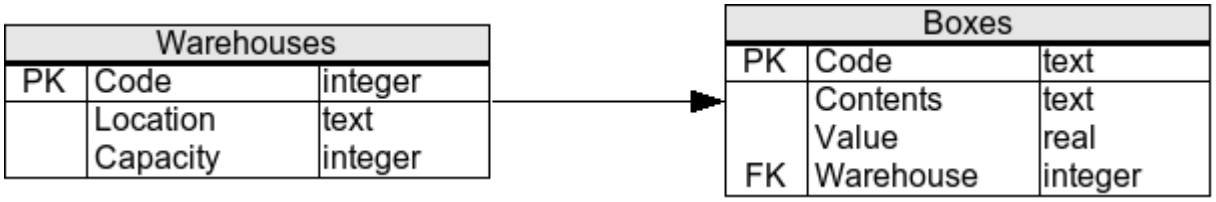

```
CREATE TABLE IF NOT EXISTS Departments (
  Code INTEGER,
  Name TEXT NOT NULL,
  Budget decimal NOT NULL,
  PRIMARY KEY (Code)
);

CREATE TABLE IF NOT EXISTS Employees (
  SSN INTEGER,
  Name varchar(255) NOT NULL ,
  LastName varchar(255) NOT NULL ,
  Department INTEGER NOT NULL ,
  PRIMARY KEY (SSN)
);


INSERT INTO Departments(Code,Name,Budget) VALUES(14,'IT',65000);
INSERT INTO Departments(Code,Name,Budget) VALUES(37,'Accounting',15000);
INSERT INTO Departments(Code,Name,Budget) VALUES(59,'Human Resources',240000);
INSERT INTO Departments(Code,Name,Budget) VALUES(77,'Research',55000);

INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('123234877','Michael','Rogers',14);
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('152934485','Anand','Manikutty',14);
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('222364883','Carol','Smith',37);
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('326587417','Joe','Stevens',37);
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('332154719','Mary-Anne','Foster',14);
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('332569843','George','ODonnell',77);
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('546523478','John','Doe',59);
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('631231482','David','Smith',77);
```

```
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('654873219','Zacary','Efron',59);
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('745685214','Eric','Goldsmith',59);
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('845657245','Elizabeth','Doe',14);
INSERT INTO Employees(SSN,Name,LastName,Department)
VALUES('845657246','Kumar','Swamy',14);
```

Questions

```
-- 2.1 Select the last name of all employees.
-- 2.2 Select the last name of all employees, without duplicates.
-- 2.3 Select all the data of employees whose last name is "Smith".
-- 2.4 Select all the data of employees whose last name is "Smith" or "Doe".
-- 2.5 Select all the data of employees that work in department 14.
-- 2.6 Select all the data of employees that work in department 37 or department 77.
-- 2.7 Select all the data of employees whose last name begins with an "S".
-- 2.8 Select the sum of all the departments' budgets.
-- 2.9 Select the number of employees in each department (you only need to show the
department code and the number of employees).
-- 2.10 Select all the data of employees, including each employee's department's data.
-- 2.11 Select the name and last name of each employee, along with the name and budget
of the employee's department.
-- 2.12 Select the name and last name of employees working for departments with a
budget greater than $60,000.
-- 2.13 Select the departments with a budget larger than the average budget of all the
departments.
-- 2.14 Select the names of departments with more than two employees.
-- 2.15 Select the name and last name of employees working for the two departments
with lowest budget.
-- 2.16  Add a new department called "Quality Assurance", with a budget of $40,000 and
departmental code 11.
-- And Add an employee called "Mary Moore" in that department, with SSN 847-21-9811.
-- 2.17 Reduce the budget of all departments by 10%.
-- 2.18 Reassign all employees from the Research department (code 77) to the IT
department (code 14).
-- 2.19 Delete from the table all employees in the IT department (code 14).
-- 2.20 Delete from the table all employees who work in departments with a budget
greater than or equal to $60,000.
-- 2.21 Delete from the table all employees.
```

Task 3



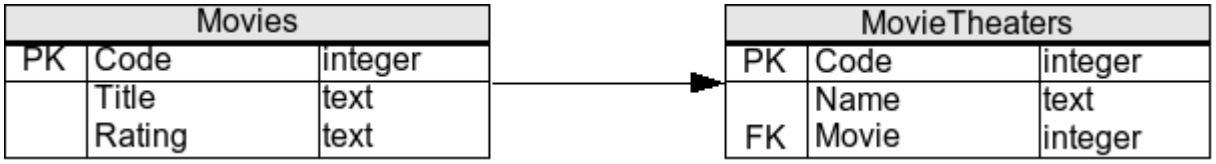

```
CREATE TABLE Warehouses (
   Code INTEGER NOT NULL,
   Location VARCHAR(255) NOT NULL ,
   Capacity INTEGER NOT NULL,
   PRIMARY KEY (Code)
 );
CREATE TABLE Boxes (
    Code CHAR(4) NOT NULL,
    Contents VARCHAR(255) NOT NULL ,
    Value REAL NOT NULL ,
    Warehouse INTEGER NOT NULL,
    PRIMARY KEY (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);
```

Questions

```
--3.1 Select all warehouses.
--3.2 Select all boxes with a value larger than $150.
--3.3 Select all distinct contents in all the boxes.
--3.4 Select the average value of all the boxes.
--3.5 Select the warehouse code and the average value of the boxes in each warehouse.
--3.6 Same as previous exercise, but select only those warehouses where the average
value of the boxes is greater than 150.
--3.7 Select the code of each box, along with the name of the city the box is located
in.
--3.8 Select the warehouse codes, along with the number of boxes in each warehouse.
--3.9 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).
```

```
--3.10 Select the codes of all the boxes located in Chicago.
--3.11 Create a new warehouse in New York with a capacity for 3 boxes.
--3.12 Create a new box, with code "H5RT", containing "Papers" with a value of $200,
and located in warehouse 2.
--3.13 Reduce the value of all boxes by 15%.
-- 3.14 Delete all records of boxes from saturated warehouses (no boxes were harmed).
--3.15 Remove all boxes with a value lower than $100.
-- 3.16 Add Index for column "Warehouse" in table "boxes"
    -- !!!NOTE!!!: index should NOT be used on small tables in practice
-- 3.17 Print all the existing indexes
    -- !!!NOTE!!!: index should NOT be used on small tables in practice
-- 3.18 Remove (drop) the index you added just
    -- !!!NOTE!!!: index should NOT be used on small tables in practice
```

Task 4

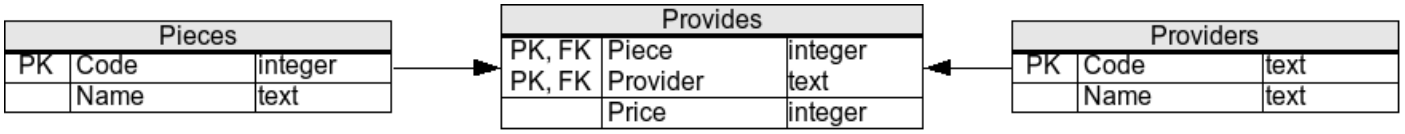



```
CREATE TABLE IF NOT EXISTS Movies (
  Code INTEGER,
  Title VARCHAR(255) NOT NULL,
  Rating VARCHAR(255),
  PRIMARY KEY (Code)
);

CREATE TABLE IF NOT EXISTS MovieTheaters (
  Code INTEGER,
  Name VARCHAR(255) NOT NULL,
  Movie INTEGER,
  PRIMARY KEY (Code)
  );

INSERT INTO Movies(Code,Title,Rating) VALUES(1,'Citizen Kane','PG');
INSERT INTO Movies(Code,Title,Rating) VALUES(2,'Singin'' in the Rain','G');
INSERT INTO Movies(Code,Title,Rating) VALUES(3,'The Wizard of Oz','G');
INSERT INTO Movies(Code,Title,Rating) VALUES(4,'The Quiet Man',NULL);
INSERT INTO Movies(Code,Title,Rating) VALUES(5,'North by Northwest',NULL);
INSERT INTO Movies(Code,Title,Rating) VALUES(6,'The Last Tango in Paris','NC-17');
INSERT INTO Movies(Code,Title,Rating) VALUES(7,'Some Like it Hot','PG-13');
INSERT INTO Movies(Code,Title,Rating) VALUES(8,'A Night at the Opera',NULL);

INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(1,'Odeon',5);
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(2,'Imperial',1);
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(3,'Majestic',NULL);
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(4,'Royale',6);
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(5,'Paraiso',3);
INSERT INTO MovieTheaters(Code,Name,Movie) VALUES(6,'Nickelodeon',NULL);
```

Questions

```
-- 4.1 Select the title of all movies.
-- 4.2 Show all the distinct ratings in the database.
-- 4.3 Show all unrated movies.
-- 4.4 Select all movie theaters that are not currently showing a movie.
-- 4.5 Select all data from all movie theaters and, additionally, the data from the
movie that is being shown in the theater (if one is being shown).
-- 4.6 Select all data from all movies and, if that movie is being shown in a theater,
show the data from the theater.
-- 4.7 Show the titles of movies not currently being shown in any theaters.
-- 4.8 Add the unrated movie "One, Two, Three".
-- 4.9 Set the rating of all unrated movies to "G".
-- 4.10 Remove movie theaters projecting movies rated "NC-17".
```

Task 5



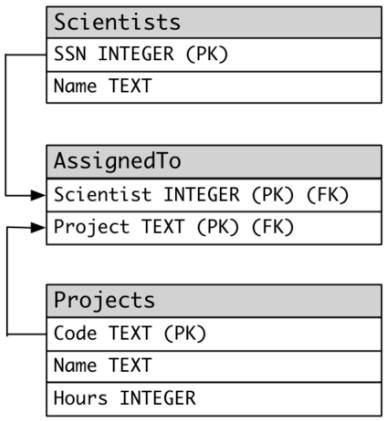

```
CREATE TABLE Pieces (
 Code INTEGER NOT NULL,
 Name TEXT NOT NULL,
 PRIMARY KEY (Code)
 );
CREATE TABLE Providers (
 Code VARCHAR(40) NOT NULL,
 Name TEXT NOT NULL,
PRIMARY KEY (Code)
 );
CREATE TABLE Provides (
 Piece INTEGER,
 Provider VARCHAR(40),
 Price INTEGER NOT NULL,
 PRIMARY KEY(Piece, Provider)
 );


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);
```

Questions

```
-- 5.1 Select the name of all the pieces.
-- 5.2  Select all the providers' data.
-- 5.3 Obtain the average price of each piece (show only the piece code and the
average price).
-- 5.4  Obtain the names of all providers who supply piece 1.
-- 5.5 Select the name of pieces provided by provider with code "HAL".
-- 5.6 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) Note: Insert a new record in the
```

table to see if your query can be scaled up to answer the query if there are two providers who supply the same piece at the most expensive price.
-- 5.7 Add an entry to the database to indicate that "Skellington Supplies" (code "TNBC") will provide sprockets (code "1") for 7 cents each.
-- 5.8 Increase all prices by one cent.
-- 5.9 Update the database to reflect that "Susan Calvin Corp." (code "RBT") will not supply bolts (code 4).
-- 5.10 Update the database to reflect that "Susan Calvin Corp." (code "RBT") will not supply any pieces (the provider should still remain in the database).

Task 6



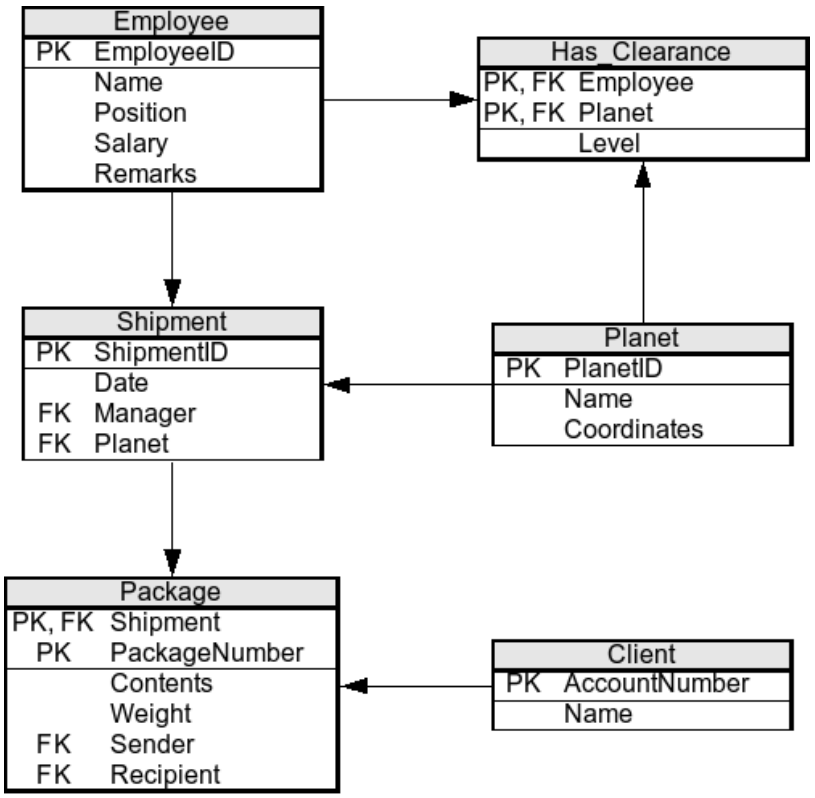

```
CREATE TABLE Scientists (
  SSN int,
  Name Char(30) not null,
  Primary Key (SSN)
);

CREATE TABLE Projects (
  Code Char(4),
  Name Char(50) not null,
  Primary Key (Code)
);

CREATE TABLE AssignedTo (
  Scientist int not null,
  Project char(4) not null,
  Hours int,
  Primary Key (Scientist, Project)
);

INSERT INTO Scientists(SSN,Name)
VALUES(123234877,'Michael Rogers'),
(152934485,'Anand Manikutty'),
(222364883, 'Carol Smith'),
(326587417,'Joe Stevens'),
(332154719,'Mary-Anne Foster'),
(332569843,'George ODonnell'),
(546523478,'John Doe'),
(631231482,'David Smith'),
(654873219,'Zacary Efron'),
(745685214,'Eric Goldsmith'),
(845657245,'Elizabeth Doe'),
(845657246,'Kumar Swamy');

INSERT INTO Projects ( Code,Name)
VALUES ('AeH1','Winds: Studying Bernoullis Principle'),
('AeH2','Aerodynamics and Bridge Design'),
('AeH3','Aerodynamics and Gas Mileage'),
('AeH4','Aerodynamics and Ice Hockey'),
```

```
('AeH5','Aerodynamics of a Football'),
('AeH6','Aerodynamics of Air Hockey'),
('Ast1','A Matter of Time'),
('Ast2','A Puzzling Parallax'),
('Ast3','Build Your Own Telescope'),
('Bte1','Juicy: Extracting Apple Juice with Pectinase'),
('Bte2','A Magnetic Primer Designer'),
('Bte3','Bacterial Transformation Efficiency'),
('Che1','A Silver-Cleaning Battery'),
('Che2','A Soluble Separation Solution');

INSERT INTO AssignedTo ( Scientist, Project, Hours)
VALUES (123234877,'AeH1', 156),
(152934485,'AeH3',189),
(222364883,'Ast3', 256),
(326587417,'Ast3', 789),
(332154719,'Bte1', 98),
(546523478,'Che1',89),
(631231482,'Ast3',112),
(654873219,'Che1', 299),
(745685214,'AeH3', 6546),
(845657245,'Ast1', 321),
(845657246,'Ast2', 9684),
(332569843,'AeH4', 321);
```

Questions

```
-- 6.1 List all the scientists' names, their projects' names,
    -- and the total hours worked on each project,
    -- in alphabetical order of project name, then scientist name.
-- 6.2 Now list the project names and total hours worked on each, from most to least
total hours.
-- 6.3 Select the project names which are not assigned yet
```

Task 7




```
CREATE TABLE Employee (
  EmployeeID INTEGER,
  Name VARCHAR(255) NOT NULL,
  Position VARCHAR(255) NOT NULL,
  Salary REAL NOT NULL,
  Remarks VARCHAR(255),
  PRIMARY KEY (EmployeeID)
);

CREATE TABLE Planet (
  PlanetID INTEGER,
  Name VARCHAR(255) NOT NULL,
  Coordinates REAL NOT NULL,
  PRIMARY KEY (PlanetID)
);

CREATE TABLE Shipment (
  ShipmentID INTEGER,
  Date DATE,
  Manager INTEGER NOT NULL,
  Planet INTEGER NOT NULL,
  PRIMARY KEY (ShipmentID)
);

CREATE TABLE Has_Clearance (
  Employee INTEGER NOT NULL,
  Planet INTEGER NOT NULL,
  Level INTEGER NOT NULL,
```

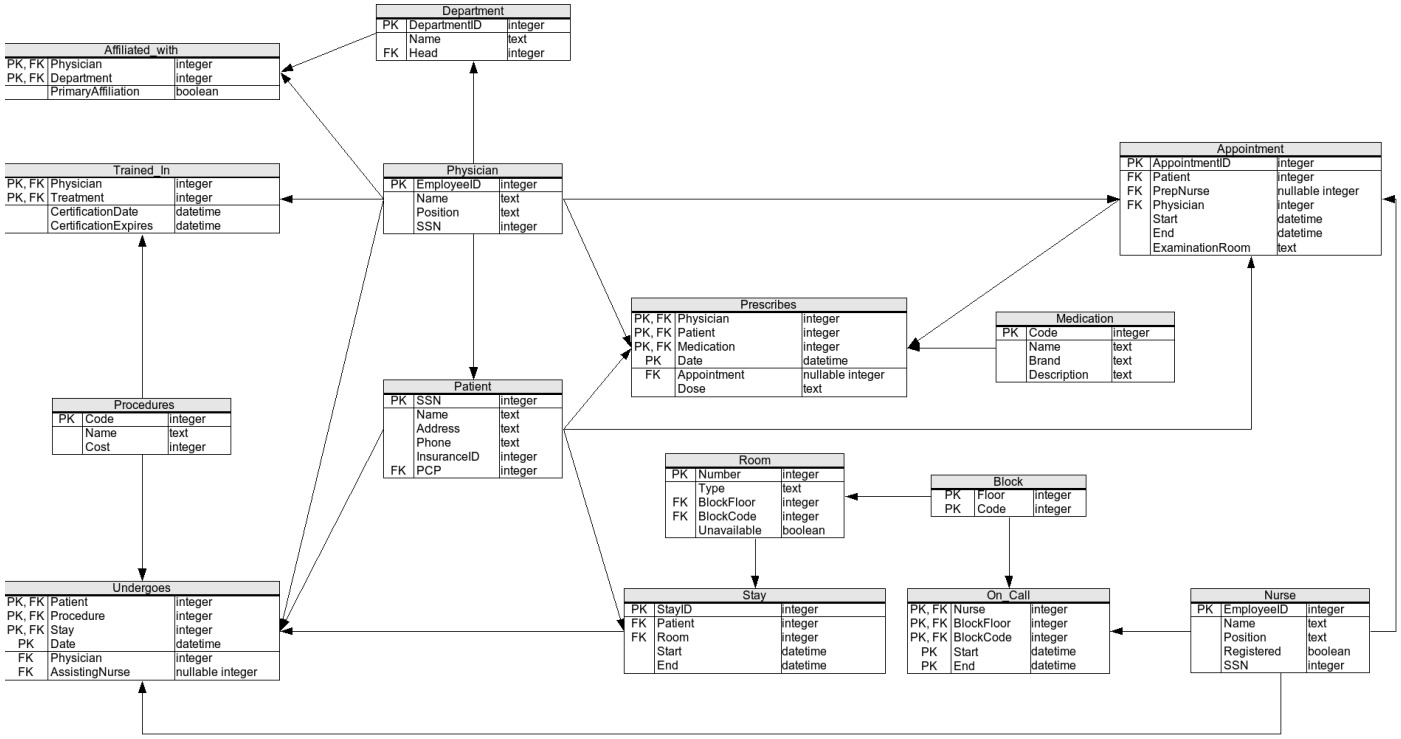
```
  PRIMARY KEY(Employee, Planet)
);

CREATE TABLE Client (
  AccountNumber INTEGER,
  Name VARCHAR(255) NOT NULL,
  PRIMARY KEY (AccountNumber)
);

CREATE TABLE Package (
  Shipment INTEGER NOT NULL,
  PackageNumber INTEGER NOT NULL,
  Contents VARCHAR(255) NOT NULL,
  Weight REAL NOT NULL,
  Sender INTEGER NOT NULL,
  Recipient INTEGER NOT NULL,
  PRIMARY KEY(Shipment, PackageNumber)
  );


INSERT INTO Client VALUES(1, 'Zapp Brannigan');
INSERT INTO Client VALUES(2, "Al Gore's Head");
INSERT INTO Client VALUES(3, 'Barbados Slim');
INSERT INTO Client VALUES(4, 'Ogden Wernstrom');
INSERT INTO Client VALUES(5, 'Leo Wong');
INSERT INTO Client VALUES(6, 'Lrrr');
INSERT INTO Client VALUES(7, 'John Zoidberg');
INSERT INTO Client VALUES(8, 'John Zoidfarb');
INSERT INTO Client VALUES(9, 'Morbo');
INSERT INTO Client VALUES(10, 'Judge John Whitey');
INSERT INTO Client VALUES(11, 'Calculon');
INSERT INTO Employee VALUES(1, 'Phillip J. Fry', 'Delivery boy', 7500.0, 'Not to be
confused with the Philip J. Fry from Hovering Squid World 97a');
INSERT INTO Employee VALUES(2, 'Turanga Leela', 'Captain', 10000.0, NULL);
INSERT INTO Employee VALUES(3, 'Bender Bending Rodriguez', 'Robot', 7500.0, NULL);
INSERT INTO Employee VALUES(4, 'Hubert J. Farnsworth', 'CEO', 20000.0, NULL);
INSERT INTO Employee VALUES(5, 'John A. Zoidberg', 'Physician', 25.0, NULL);
INSERT INTO Employee VALUES(6, 'Amy Wong', 'Intern', 5000.0, NULL);
INSERT INTO Employee VALUES(7, 'Hermes Conrad', 'Bureaucrat', 10000.0, NULL);
INSERT INTO Employee VALUES(8, 'Scruffy Scruffington', 'Janitor', 5000.0, NULL);
INSERT INTO Planet VALUES(1, 'Omicron Persei 8', 89475345.3545);
INSERT INTO Planet VALUES(2, 'Decapod X', 65498463216.3466);
INSERT INTO Planet VALUES(3, 'Mars', 32435021.65468);
INSERT INTO Planet VALUES(4, 'Omega III', 98432121.5464);
INSERT INTO Planet VALUES(5, 'Tarantulon VI', 849842198.354654);
INSERT INTO Planet VALUES(6, 'Cannibalon', 654321987.21654);
INSERT INTO Planet VALUES(7, 'DogDoo VII', 65498721354.688);
INSERT INTO Planet VALUES(8, 'Nintenduu 64', 6543219894.1654);
INSERT INTO Planet VALUES(9, 'Amazonia', 65432135979.6547);
INSERT INTO Has_Clearance VALUES(1, 1, 2);
INSERT INTO Has_Clearance VALUES(1, 2, 3);
INSERT INTO Has_Clearance VALUES(2, 3, 2);
INSERT INTO Has_Clearance VALUES(2, 4, 4);
INSERT INTO Has_Clearance VALUES(3, 5, 2);
INSERT INTO Has_Clearance VALUES(3, 6, 4);
INSERT INTO Has_Clearance VALUES(4, 7, 1);
INSERT INTO Shipment VALUES(1, '3004/05/11', 1, 2);
```

```
INSERT INTO Shipment VALUES(2, '3004/05/11', 1, 2);
INSERT INTO Shipment VALUES(3, NULL, 2, 3);
INSERT INTO Shipment VALUES(4, NULL, 2, 4);
INSERT INTO Shipment VALUES(5, NULL, 7, 5);
INSERT INTO Package VALUES(1, 1, 'Undeclared', 1.5, 1, 2);
INSERT INTO Package VALUES(2, 1, 'Undeclared', 10.0, 2, 3);
INSERT INTO Package VALUES(2, 2, 'A bucket of krill', 2.0, 8, 7);
INSERT INTO Package VALUES(3, 1, 'Undeclared', 15.0, 3, 4);
INSERT INTO Package VALUES(3, 2, 'Undeclared', 3.0, 5, 1);
INSERT INTO Package VALUES(3, 3, 'Undeclared', 7.0, 2, 3);
INSERT INTO Package VALUES(4, 1, 'Undeclared', 5.0, 4, 5);
INSERT INTO Package VALUES(4, 2, 'Undeclared', 27.0, 1, 2);
INSERT INTO Package VALUES(5, 1, 'Undeclared', 100.0, 5, 1);
```

Questions

```
-- 7.1 Who received a 1.5kg package?
    -- The result is "Al Gore's Head".

-- 7.2 What is the total weight of all the packages that he sent?


-- Optional questions

-- 7.3 Retrieve the names of employees who have clearance level 4 or higher.

-- 7.4 Rank the employees based on their salary in descending order, showing their
names and positions alongside their rank. (Hint: Window Function)

-- 7.5 Create a CTE to calculate the total weight of packages sent to each planet,
then join it with the Planet table to display the planet names and their corresponding
total weights.

-- 7.6 Retrieve the names of employees who have managed the shipment of packages to
planets that no other employee has shipped to.

-- 7.7 Retrieve the names of planets along with the number of shipments made to each
planet, but exclude planets where the total weight of packages sent is less than 20
units.
```

Task 8




```
CREATE TABLE Physician (
EmployeeID INTEGER NOT NULL,
Name TEXT NOT NULL,
Position TEXT NOT NULL,
SSN INTEGER NOT NULL,
PRIMARY KEY (EmployeeID)
);

CREATE TABLE Department (
DepartmentID INTEGER NOT NULL,
Name TEXT NOT NULL,
Head INTEGER NOT NULL,
PRIMARY KEY (DepartmentID)
);

CREATE TABLE Affiliated_With (
Physician INTEGER NOT NULL,
Department INTEGER NOT NULL,
PrimaryAffiliation BOOLEAN NOT NULL,
PRIMARY KEY(Physician, Department)
);

CREATE TABLE Procedures (
Code INTEGER NOT NULL,
Name TEXT NOT NULL,
Cost REAL NOT NULL,
PRIMARY KEY (Code)
);

CREATE TABLE Trained_In (
Physician INTEGER NOT NULL,
Treatment INTEGER NOT NULL,
CertificationDate DATETIME NOT NULL,
```

```
CertificationExpires DATETIME NOT NULL,
PRIMARY KEY(Physician, Treatment)
);

CREATE TABLE Patient (
SSN INTEGER NOT NULL,
Name TEXT NOT NULL,
Address TEXT NOT NULL,
Phone TEXT NOT NULL,
InsuranceID INTEGER NOT NULL,
PCP INTEGER NOT NULL,
PRIMARY KEY (SSN)
);

CREATE TABLE Nurse (
EmployeeID INTEGER NOT NULL,
Name TEXT NOT NULL,
Position TEXT NOT NULL,
Registered BOOLEAN NOT NULL,
SSN INTEGER NOT NULL,
PRIMARY KEY (EmployeeID)
);

CREATE TABLE Appointment (
AppointmentID INTEGER NOT NULL,
Patient INTEGER NOT NULL,
PrepNurse INTEGER,
Physician INTEGER NOT NULL,
Start DATETIME NOT NULL,
End DATETIME NOT NULL,
ExaminationRoom TEXT NOT NULL,
PRIMARY KEY (AppointmentID)
);

CREATE TABLE Medication (
Code INTEGER NOT NULL,
Name TEXT NOT NULL,
Brand TEXT NOT NULL,
Description TEXT NOT NULL,
PRIMARY KEY(Code)
);

CREATE TABLE Prescribes (
Physician INTEGER NOT NULL,
Patient INTEGER NOT NULL,
Medication INTEGER NOT NULL,
Date DATETIME NOT NULL,
Appointment INTEGER,
Dose TEXT NOT NULL,
PRIMARY KEY(Physician, Patient, Medication, Date)
);

CREATE TABLE Block (
Floor INTEGER NOT NULL,
Code INTEGER NOT NULL,
PRIMARY KEY(Floor, Code)
);
```

```
CREATE TABLE Room (
Number INTEGER NOT NULL,
Type TEXT NOT NULL,
BlockFloor INTEGER NOT NULL,
BlockCode INTEGER NOT NULL,
Unavailable BOOLEAN NOT NULL,
PRIMARY KEY(Number)
);

CREATE TABLE On_Call (
Nurse INTEGER NOT NULL,
BlockFloor INTEGER NOT NULL,
BlockCode INTEGER NOT NULL,
Start DATETIME NOT NULL,
End DATETIME NOT NULL,
PRIMARY KEY(Nurse, BlockFloor, BlockCode, Start, End)
);

CREATE TABLE Stay (
StayID INTEGER NOT NULL,
Patient INTEGER NOT NULL,
Room INTEGER NOT NULL,
Start DATETIME NOT NULL,
End DATETIME NOT NULL,
PRIMARY KEY(StayID)
);

CREATE TABLE Undergoes (
Patient INTEGER NOT NULL,
Procedure INTEGER NOT NULL,
Stay INTEGER NOT NULL,
Date DATETIME NOT NULL,
Physician INTEGER NOT NULL,
AssistingNurse INTEGER,
PRIMARY KEY(Patient, Procedure, Stay, Date)
);


INSERT INTO Physician VALUES(1,'John Dorian','Staff Internist',111111111);
INSERT INTO Physician VALUES(2,'Elliot Reid','Attending Physician',222222222);
INSERT INTO Physician VALUES(3,'Christopher Turk','Surgical Attending
Physician',333333333);
INSERT INTO Physician VALUES(4,'Percival Cox','Senior Attending Physician',444444444);
INSERT INTO Physician VALUES(5,'Bob Kelso','Head Chief of Medicine',555555555);
INSERT INTO Physician VALUES(6,'Todd Quinlan','Surgical Attending
Physician',666666666);
INSERT INTO Physician VALUES(7,'John Wen','Surgical Attending Physician',777777777);
INSERT INTO Physician VALUES(8,'Keith Dudemeister','MD Resident',888888888);
INSERT INTO Physician VALUES(9,'Molly Clock','Attending Psychiatrist',999999999);

INSERT INTO Department VALUES(1,'General Medicine',4);
INSERT INTO Department VALUES(2,'Surgery',7);
INSERT INTO Department VALUES(3,'Psychiatry',9);

INSERT INTO Affiliated_With VALUES(1,1,1);
INSERT INTO Affiliated_With VALUES(2,1,1);
```

```
INSERT INTO Affiliated_With VALUES(3,1,0);
INSERT INTO Affiliated_With VALUES(3,2,1);
INSERT INTO Affiliated_With VALUES(4,1,1);
INSERT INTO Affiliated_With VALUES(5,1,1);
INSERT INTO Affiliated_With VALUES(6,2,1);
INSERT INTO Affiliated_With VALUES(7,1,0);
INSERT INTO Affiliated_With VALUES(7,2,1);
INSERT INTO Affiliated_With VALUES(8,1,1);
INSERT INTO Affiliated_With VALUES(9,3,1);

INSERT INTO Procedures VALUES(1,'Reverse Rhinopodoplasty',1500.0);
INSERT INTO Procedures VALUES(2,'Obtuse Pyloric Recombobulation',3750.0);
INSERT INTO Procedures VALUES(3,'Folded Demiophtalmectomy',4500.0);
INSERT INTO Procedures VALUES(4,'Complete Walletectomy',10000.0);
INSERT INTO Procedures VALUES(5,'Obfuscated Dermogastrotomy',4899.0);
INSERT INTO Procedures VALUES(6,'Reversible Pancreomyoplasty',5600.0);
INSERT INTO Procedures VALUES(7,'Follicular Demiectomy',25.0);

INSERT INTO Patient VALUES(100000001,'John Smith','42 Foobar
Lane','555-0256',68476213,1);
INSERT INTO Patient VALUES(100000002,'Grace Ritchie','37 Snafu
Drive','555-0512',36546321,2);
INSERT INTO Patient VALUES(100000003,'Random J. Patient','101 Omgbbq
Street','555-1204',65465421,2);
INSERT INTO Patient VALUES(100000004,'Dennis Doe','1100 Foobaz
Avenue','555-2048',68421879,3);

INSERT INTO Nurse VALUES(101,'Carla Espinosa','Head Nurse',1,111111110);
INSERT INTO Nurse VALUES(102,'Laverne Roberts','Nurse',1,222222220);
INSERT INTO Nurse VALUES(103,'Paul Flowers','Nurse',0,333333330);

INSERT INTO Appointment VALUES(13216584,100000001,101,1,'2008-04-24 10:00','2008-04-24
11:00','A');
INSERT INTO Appointment VALUES(26548913,100000002,101,2,'2008-04-24 10:00','2008-04-24
11:00','B');
INSERT INTO Appointment VALUES(36549879,100000001,102,1,'2008-04-25 10:00','2008-04-25
11:00','A');
INSERT INTO Appointment VALUES(46846589,100000004,103,4,'2008-04-25 10:00','2008-04-25
11:00','B');
INSERT INTO Appointment VALUES(59871321,100000004,NULL,4,'2008-04-26
10:00','2008-04-26 11:00','C');
INSERT INTO Appointment VALUES(69879231,100000003,103,2,'2008-04-26 11:00','2008-04-26
12:00','C');
INSERT INTO Appointment VALUES(76983231,100000001,NULL,3,'2008-04-26
12:00','2008-04-26 13:00','C');
INSERT INTO Appointment VALUES(86213939,100000004,102,9,'2008-04-27 10:00','2008-04-21
11:00','A');
INSERT INTO Appointment VALUES(93216548,100000002,101,2,'2008-04-27 10:00','2008-04-27
11:00','B');

INSERT INTO Medication VALUES(1,'Procrastin-X','X','N/A');
INSERT INTO Medication VALUES(2,'Thesisin','Foo Labs','N/A');
INSERT INTO Medication VALUES(3,'Awakin','Bar Laboratories','N/A');
INSERT INTO Medication VALUES(4,'Crescavitin','Baz Industries','N/A');
INSERT INTO Medication VALUES(5,'Melioraurin','Snafu Pharmaceuticals','N/A');

INSERT INTO Prescribes VALUES(1,100000001,1,'2008-04-24 10:47',13216584,'5');
```

```
INSERT INTO Prescribes VALUES(9,100000004,2,'2008-04-27 10:53',86213939,'10');
INSERT INTO Prescribes VALUES(9,100000004,2,'2008-04-30 16:53',NULL,'5');

INSERT INTO Block VALUES(1,1);
INSERT INTO Block VALUES(1,2);
INSERT INTO Block VALUES(1,3);
INSERT INTO Block VALUES(2,1);
INSERT INTO Block VALUES(2,2);
INSERT INTO Block VALUES(2,3);
INSERT INTO Block VALUES(3,1);
INSERT INTO Block VALUES(3,2);
INSERT INTO Block VALUES(3,3);
INSERT INTO Block VALUES(4,1);
INSERT INTO Block VALUES(4,2);
INSERT INTO Block VALUES(4,3);

INSERT INTO Room VALUES(101,'Single',1,1,0);
INSERT INTO Room VALUES(102,'Single',1,1,0);
INSERT INTO Room VALUES(103,'Single',1,1,0);
INSERT INTO Room VALUES(111,'Single',1,2,0);
INSERT INTO Room VALUES(112,'Single',1,2,1);
INSERT INTO Room VALUES(113,'Single',1,2,0);
INSERT INTO Room VALUES(121,'Single',1,3,0);
INSERT INTO Room VALUES(122,'Single',1,3,0);
INSERT INTO Room VALUES(123,'Single',1,3,0);
INSERT INTO Room VALUES(201,'Single',2,1,1);
INSERT INTO Room VALUES(202,'Single',2,1,0);
INSERT INTO Room VALUES(203,'Single',2,1,0);
INSERT INTO Room VALUES(211,'Single',2,2,0);
INSERT INTO Room VALUES(212,'Single',2,2,0);
INSERT INTO Room VALUES(213,'Single',2,2,1);
INSERT INTO Room VALUES(221,'Single',2,3,0);
INSERT INTO Room VALUES(222,'Single',2,3,0);
INSERT INTO Room VALUES(223,'Single',2,3,0);
INSERT INTO Room VALUES(301,'Single',3,1,0);
INSERT INTO Room VALUES(302,'Single',3,1,1);
INSERT INTO Room VALUES(303,'Single',3,1,0);
INSERT INTO Room VALUES(311,'Single',3,2,0);
INSERT INTO Room VALUES(312,'Single',3,2,0);
INSERT INTO Room VALUES(313,'Single',3,2,0);
INSERT INTO Room VALUES(321,'Single',3,3,1);
INSERT INTO Room VALUES(322,'Single',3,3,0);
INSERT INTO Room VALUES(323,'Single',3,3,0);
INSERT INTO Room VALUES(401,'Single',4,1,0);
INSERT INTO Room VALUES(402,'Single',4,1,1);
INSERT INTO Room VALUES(403,'Single',4,1,0);
INSERT INTO Room VALUES(411,'Single',4,2,0);
INSERT INTO Room VALUES(412,'Single',4,2,0);
INSERT INTO Room VALUES(413,'Single',4,2,0);
INSERT INTO Room VALUES(421,'Single',4,3,1);
INSERT INTO Room VALUES(422,'Single',4,3,0);
INSERT INTO Room VALUES(423,'Single',4,3,0);

INSERT INTO On_Call VALUES(101,1,1,'2008-11-04 11:00','2008-11-04 19:00');
INSERT INTO On_Call VALUES(101,1,2,'2008-11-04 11:00','2008-11-04 19:00');
INSERT INTO On_Call VALUES(102,1,3,'2008-11-04 11:00','2008-11-04 19:00');
INSERT INTO On_Call VALUES(103,1,1,'2008-11-04 19:00','2008-11-05 03:00');
```

```
INSERT INTO On_Call VALUES(103,1,2,'2008-11-04 19:00','2008-11-05 03:00');
INSERT INTO On_Call VALUES(103,1,3,'2008-11-04 19:00','2008-11-05 03:00');

INSERT INTO Stay VALUES(3215,100000001,111,'2008-05-01','2008-05-04');
INSERT INTO Stay VALUES(3216,100000003,123,'2008-05-03','2008-05-14');
INSERT INTO Stay VALUES(3217,100000004,112,'2008-05-02','2008-05-03');

INSERT INTO Undergoes VALUES(100000001,6,3215,'2008-05-02',3,101);
INSERT INTO Undergoes VALUES(100000001,2,3215,'2008-05-03',7,101);
INSERT INTO Undergoes VALUES(100000004,1,3217,'2008-05-07',3,102);
INSERT INTO Undergoes VALUES(100000004,5,3217,'2008-05-09',6,NULL);
INSERT INTO Undergoes VALUES(100000001,7,3217,'2008-05-10',7,101);
INSERT INTO Undergoes VALUES(100000004,4,3217,'2008-05-13',3,103);

INSERT INTO Trained_In VALUES(3,1,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(3,2,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(3,5,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(3,6,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(3,7,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(6,2,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(6,5,'2007-01-01','2007-12-31');
INSERT INTO Trained_In VALUES(6,6,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(7,1,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(7,2,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(7,3,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(7,4,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(7,5,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(7,6,'2008-01-01','2008-12-31');
INSERT INTO Trained_In VALUES(7,7,'2008-01-01','2008-12-31');
```

Questions

-- 8.1 Obtain the names of all physicians that have performed a medical procedure they have never been certified to perform.

-- 8.2 Same as the previous query, but include the following information in the results: Physician name, name of procedure, date when the procedure was carried out, name of the patient the procedure was carried out on.

-- 8.3 Obtain the names of all physicians that have performed a medical procedure that they are certified to perform, but such that the procedure was done at a date (Undergoes.Date) after the physician's certification expired (Trained_In.CertificationExpires).

-- 8.4 Same as the previous query, but include the following information in the results: Physician name, name of procedure, date when the procedure was carried out, name of the patient the procedure was carried out on, and date when the certification expired.

-- 8.5 Obtain the information for appointments where a patient met with a physician other than his/her primary care physician. Show the following information: Patient name, physician name, nurse name (if any), start and end time of appointment, examination room, and the name of the patient's primary care physician.

-- 8.6 The Patient field in Undergoes is redundant, since we can obtain it from the Stay table. There are no constraints in force to prevent inconsistencies between these two tables. More specifically, the Undergoes table may include a row where the patient ID does not match the one we would obtain from the Stay table through the Undergoes.Stay foreign key. Select all rows from Undergoes that exhibit this inconsistency.

-- 8.7 Obtain the names of all the nurses who have ever been on call for room 123.

```
-- 8.8 The hospital has several examination rooms where appointments take place.
Obtain the number of appointments that have taken place in each examination room.
-- 8.9 Obtain the names of all patients and their primary care physician, such that
the following are true:
    -- The patient has been prescribed some medication by his/her primary care
physician.
    -- The patient has undergone a procedure with a cost larger that $5,000
    -- The patient has had at least two appointments where the nurse who prepared the
appointment was a registered nurse.
    -- The patient's primary care physician is not the head of any department.
```

Task 9

Download the csv file from this link and import the file. If using DB Browser SQLite, try to do it without any tutorials. In case you need any help, follow this video.

Questions

-- 9.1 Give the package name and how many times they're downloaded. Order by the 2nd column descending.
-- 9.2 Give the package ranking (based on how many times it was downloaded) during 9AM to 11AM
-- 9.3 How many records (downloads) are from China ("CN") or Japan("JP") or Singapore ("SG")?
-- 9.4 Print the countries whose downloads are more than the downloads from China ("CN")
-- 9.5 Print the average length of the package name of all the UNIQUE packages
-- 9.6 Get the package whose download count ranks 2nd (print package name and its download count).
-- 9.7 Print the name of the package whose download count is bigger than 1000.
-- 9.8 The field "r_os" is the operating system of the users.
    --   Here we would like to know what main system we have (ignore version number), the relevant counts, and the proportion (in percentage).

Task 10

```
CREATE TABLE PEOPLE (id INTEGER, name CHAR);

INSERT INTO PEOPLE VALUES(1, "A");
INSERT INTO PEOPLE VALUES(2, "B");
INSERT INTO PEOPLE VALUES(3, "C");
INSERT INTO PEOPLE VALUES(4, "D");

CREATE TABLE ADDRESS (id INTEGER, address CHAR, updatedate date);

INSERT INTO ADDRESS VALUES(1, "address-1-1", "2016-01-01");
INSERT INTO ADDRESS VALUES(1, "address-1-2", "2016-09-02");
INSERT INTO ADDRESS VALUES(2, "address-2-1", "2015-11-01");
INSERT INTO ADDRESS VALUES(3, "address-3-1", "2016-12-01");
INSERT INTO ADDRESS VALUES(3, "address-3-2", "2014-09-11");
INSERT INTO ADDRESS VALUES(3, "address-3-3", "2015-01-01");
INSERT INTO ADDRESS VALUES(4, "address-4-1", "2010-05-21");
INSERT INTO ADDRESS VALUES(4, "address-4-2", "2012-02-11");
INSERT INTO ADDRESS VALUES(4, "address-4-3", "2015-04-27");
INSERT INTO ADDRESS VALUES(4, "address-4-4", "2014-01-01");
```

Questions

```
-- 10.1 Join table PEOPLE and ADDRESS, but keep only one address information for each
person (we don't mind which record we take for each person).
    -- i.e., the joined table should have the same number of rows as table PEOPLE
-- 10.2 Join table PEOPLE and ADDRESS, but ONLY keep the LATEST address information
for each person.
    -- i.e., the joined table should have the same number of rows as table PEOPLE
```

Questions and solutions

Task 1

```sql
-- 1.1 Select the names of all the products in the store.
SELECT Name
FROM Products;

-- 1.2 Select the names and the prices of all the products in the store.
SELECT Name, Price
FROM Products;

-- 1.3 Select the name of the products with a price less than or equal to $200.
SELECT Name
FROM Products
WHERE Price <= 200;

-- 1.4 Select all the products with a price between $60 and $120.
SELECT *
FROM Products
WHERE Price BETWEEN 60 AND 120;

SELECT *
FROM Products
WHERE Price >= 60 AND Price <=120;

-- 1.5 Select the name and price in cents (i.e., the price must be multiplied by 100).
SELECT Name, Price*100 AS price_in_cents
FROM Products;

-- 1.6 Compute the average price of all the products.
SELECT AVG(Price) AS average_price
FROM Products;

SELECT sum(Price)/count(Price) AS average_price
FROM Products;

-- 1.7 Compute the average price of all products with manufacturer code equal to 2.
SELECT AVG(Price) AS average_price
FROM Products
WHERE Manufacturer = 2;

-- 1.8 Compute the number of products with a price larger than or equal to $180.
SELECT COUNT(*) AS number_of products
FROM Products
WHERE Price>=180;

-- 1.9 Select the name and price of all products with a price larger than or equal to
$180, and sort first by price (in descending order), and then by name (in ascending
order).
SELECT Name, Price
FROM Products
WHERE Price>=180
ORDER BY Price DESC, Name;
```

```sql
-- 1.10 Select all the data from the products, including all the data for each
product's manufacturer.
SELECT a.*, b.*
FROM Products a
JOIN Manufacturers b
ON a.Manufacturer = b.Code;

SELECT a.*, b.*
FROM Products a, Manufacturers b
WHERE a.Manufacturer = b.Code;

-- 1.11 Select the product name, price, and manufacturer name of all the products.
SELECT Products.Name, Price, Manufacturers.Name
FROM Products
JOIN Manufacturers
ON Products.Manufacturer = Manufacturers.Code;

-- 1.12 Select the average price of each manufacturer's products, showing only the
manufacturer's code.
SELECT AVG(Price) AS average_price, Manufacturer
FROM Products
GROUP BY Manufacturer;

-- 1.13 Select the average price of each manufacturer's products, showing the
manufacturer's name.
SELECT AVG(a.Price) AS average_price, b.name
FROM Products a
JOIN Manufacturers b
ON a.Manufacturer = b.Code
GROUP BY b.Name;

-- 1.14 Select the names of manufacturer whose products have an average price larger
than or equal to $150.
SELECT AVG(a.Price) AS average_price, b.Name
FROM Manufacturers b
JOIN Products a
ON b.Code = a.Manufacturer
GROUP BY b.Name
HAVING AVG(a.Price)>=150;

SELECT AVG(Price) AS average_price, Manufacturers.Name
FROM Products, Manufacturers
WHERE Products.Manufacturer = Manufacturers.Code
GROUP BY Manufacturers.Name
HAVING AVG(Price) >= 150;


-- 1.15 Select the name and price of the cheapest product.
SELECT Name, Price
FROM Products
WHERE Price = (
SELECT MIN(Price)
FROM Products
);

SELECT Name, Price
FROM Products
```

```
ORDER BY Price
LIMIT 1;

-- 1.16 Select the name of each manufacturer along with the name and price of its most
expensive product.
-- using subquery in FROM
SELECT max_price.name, max_price, p.name
FROM (
SELECT m.code, m.name, MAX(p.price) max_price
FROM manufacturers m
JOIN products p ON m.code=p.manufacturer
GROUP BY m.code, m.name
ORDER BY m.name) max_price
JOIN products p ON max_price.code=p.manufacturer
WHERE max_price.max_price=p.price;

-- using subquery in WHERE
SELECT *
FROM manufacturers m
JOIN products p ON m.code=p.manufacturer
WHERE price IN (SELECT MAX(p.price) max_price
FROM manufacturers m
JOIN products p ON m.code=p.manufacturer
GROUP BY m.code, m.name
ORDER BY m.name);

-- using CTE
WITH max_price AS (
SELECT m.code, m.name, MAX(p.price) max_price
FROM manufacturers m
JOIN products p ON m.code=p.manufacturer
GROUP BY m.code, m.name
ORDER BY m.name
)

SELECT m.name, max_price, p.name
FROM max_price m
JOIN products p ON m.code=p.manufacturer
WHERE m.max_price=p.price

-- using VIEW
CREATE VIEW max_price AS
SELECT m.code, m.name, MAX(p.price) max_price
FROM manufacturers m
JOIN products p ON m.code=p.manufacturer
GROUP BY m.code, m.name
ORDER BY m.name;

SELECT m.name, max_price, p.name
FROM max_price m
JOIN products p ON m.code=p.manufacturer
WHERE m.max_price=p.price




-- 1.17 Add a new product: Loudspeakers, $70, manufacturer 2.
```

```sql
INSERT INTO Products
VALUES (11, 'Loudspeakers', 70, 2);

-- 1.18 Update the name of product 8 to "Laser Printer".
UPDATE Products
SET Name = 'Laser Printer'
WHERE code=8;

-- 1.19 Apply a 10% discount to all products.
UPDATE Products
SET Price=price*0.9;

-- 1.20 Apply a 10% discount to all products with a price larger than or equal to
$120.
UPDATE Products
SET Price = price * 0.9
WHERE Price >= 120;
```

Task 2

```
-- 2.1 Select the last name of all employees.
SELECT LastName
FROM Employees;


-- 2.2 Select the last name of all employees, without duplicates.
SELECT DISTINCT LastName
FROM Employees;


-- 2.3 Select all the data of employees whose last name is "Smith".
SELECT *
FROM Employees
WHERE LastName = 'Smith';


-- 2.4 Select all the data of employees whose last name is "Smith" or "Doe".
SELECT *
FROM Employees
WHERE LastName IN ('Smith', 'Doe');

SELECT *
FROM Employees
WHERE LastName = 'Smith' OR LastName = 'Doe';


-- 2.5 Select all the data of employees that work in department 14.
SELECT *
FROM Employees
WHERE Department = 14;


-- 2.6 Select all the data of employees that work in department 37 or department 77.
SELECT *
FROM Employees
WHERE Department = 37 OR Department = 77;

SELECT *
FROM Employees
WHERE Department IN (37, 77);


-- 2.7 Select all the data of employees whose last name begins with an "S".
SELECT *
FROM Employees
WHERE LastName LIKE 'S%';


-- 2.8 Select the sum of all the departments' budgets (hint: use UNION).
SELECT 'Total Budget' AS Name, SUM(Budget) AS Budget
FROM Departments
UNION
SELECT Name,SUM(Budget)sum
from Departments
```

```
group by Name
order by Budget;

-- 2.9 Select the number of employees in each department (you only need to show the
department code and the number of employees).
SELECT Department, COUNT(*) AS number_of_employees
FROM Employees
GROUP BY Department;


-- 2.10 Select all the data of employees, including each employee's department's data.
SELECT E.*, D.*
FROM Employees E
JOIN Departments D
ON E.Department = D.Code
ORDER BY E.Department;

SELECT SSN, E.Name AS EmpName, LastName, D.Name AS DepName, Department, Code, Budget
FROM Employees E
JOIN Departments D
ON E.Department = D.Code
ORDER BY E.Department;

-- 2.11 Select the name and last name of each employee, along with the name and budget
of the employee's department.
SELECT E.Name, E.LastName, D.Name AS DepName, D.Budget
FROM Employees AS E
JOIN Departments AS D
ON E.Department = D.Code
ORDER BY DepName;


-- 2.12 Select the name and last name of employees working for departments with a
budget greater than $60,000.
/* Without subquery */
SELECT Employees.Name, LastName
FROM Employees
JOIN Departments
ON Employees.Department = Departments.Code
AND Departments.Budget > 60000;

/* With subquery */
SELECT Name, LastName
FROM Employees
WHERE Department IN
  (SELECT Code
  FROM Departments
  WHERE Budget > 60000);

/* With CTE */
WITH filtered_deps AS (
SELECT *
FROM Departments
WHERE Budget > 60000
)

SELECT E.Name, E.LastName
```

```
FROM Employees AS E
JOIN filtered_deps AS F
ON E.Department=F.Code;

-- 2.13 Select the departments with a budget larger than the average budget of all the
departments.
Budget of all the Departments.
SELECT *
FROM Departments
WHERE Budget > (
SELECT AVG(Budget) FROM Departments
);


-- 2.14 Select the names of departments with more than two employees.
/* With subquery */
SELECT Name
FROM Departments
WHERE Code IN
  (SELECT Department
   FROM Employees
   GROUP BY Department
   HAVING COUNT(*) > 2
  );

/* With JOIN */
SELECT Departments.Name
FROM Employees
JOIN Departments
ON Department = Code
GROUP BY Departments.Name
HAVING COUNT(*) > 2;


-- 2.15 Select the name and last name of employees working for the two departments
with lowest budget.
/* With CTE */
WITH lowest_budget AS (SELECT *
        FROM Departments
        ORDER BY Budget
        LIMIT 2
)

SELECT E.Name, E.LastName
FROM Employees AS E
JOIN lowest_budget AS L
ON E.Department=L.Code

/* With subquery */
SELECT Name, LastName
FROM Employees
WHERE Department IN (
SELECT lowest_budget.Code
FROM (SELECT *
        FROM Departments
        ORDER BY Budget
        LIMIT 2) AS lowest_budget
```

```
ORDER BY lowest_budget.Budget
);


-- 2.16  Add a new department called "Quality Assurance", with a budget of $40,000 and
departmental code 11. and Add an employee called "Mary Moore" in that department, with
SSN 847-21-9811.
INSERT INTO Departments values(11, 'Quality Assurance', 40000);
INSERT INTO Employees values(847219811, 'Mary', 'Moore', 11);

-- 2.17 Reduce the budget of all departments by 10%.
UPDATE Departments
SET Budget = 0.9 * Budget;


-- 2.18 Reassign all employees from the Research department (code 77) to the IT
department (code 14).
UPDATE Employees
SET Department = 14
WHERE Department = 77;


-- 2.19 Delete from the table all employees in the IT department (code 14).
DELETE FROM Employees
WHERE Department = 14;


-- 2.20 Delete from the table all employees who work in departments with a budget
greater than or equal to $60,000.
DELETE FROM Employees
WHERE Department IN (
SELECT Code
FROM Departments
WHERE Budget>=60000
);


-- 2.21 Delete from the table all employees.
DELETE
FROM Employees;
```

Task 3

```
--3.1 Select all warehouses.
SELECT *
FROM Warehouses;


--3.2 Select all boxes with a value larger than $150.
SELECT *
FROM Boxes
WHERE Value>150;


--3.3 Select all distinct contents in all the boxes.
SELECT DISTINCT Contents
FROM Boxes;


--3.4 Select the average value of all the boxes.
SELECT ROUND(AVG(Value), 2) AS AvgValue
FROM Boxes;


--3.5 Select the warehouse code and the average value of the boxes in each warehouse.
SELECT Warehouse, ROUND(AVG(Value), 2) AS AvgValue
FROM Boxes
GROUP BY Warehouse;


--3.6 Same as previous exercise, but select only those warehouses where the average
value of the boxes is greater than 150.
SELECT warehouse , AVG(value) AS “Average value”
FROM Boxes
GROUP BY warehouse
HAVING AVG(value)>150


--3.7 Select the code of each box, along with the name of the city the box is located
in.
SELECT B.Code, W.Location
FROM Boxes AS B
JOIN Warehouses AS W
ON B.Warehouse=W.Code
ORDER BY W.Code;

--3.8 Select the warehouse codes, along with the number of boxes in each warehouse.
SELECT Warehouse, COUNT(*) AS number_of_boxes
FROM Boxes
GROUP BY Warehouse;

--3.9 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 B.Warehouse, W.Capacity, COUNT(B.Code) AS number_of_boxes
FROM Warehouses AS W
JOIN Boxes AS B
ON W.Code=B.Warehouse
```

```
GROUP BY W.Code
HAVING W.Capacity < number_of_boxes
ORDER BY W.Code;

--3.10 Select the codes of all the boxes located in Chicago.
SELECT B.Code, W.Location
FROM Warehouses AS W
JOIN Boxes AS B
ON W.Code=B.Warehouse
WHERE W.Location = 'Chicago';

--3.11 Create a new warehouse in New York with a capacity for 3 boxes.
INSERT INTO Warehouses
VALUES(6, 'New York', 3);

--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', 'Paper', 200, 2);

--3.13 Reduce the value of all boxes by 15%.
UPDATE Boxes
SET Value = Value * 0.85;

--3.14 Remove all boxes with a value lower than $100.
DELETE FROM Boxes
WHERE Value < 100;

-- 3.15 Remove all boxes from saturated warehouses.
-- not for MySQL
DELETE
FROM Boxes
WHERE Warehouse IN
      (SELECT Code
      FROM Warehouses
      WHERE Capacity <
            (SELECT COUNT(*)
            FROM Boxes
            WHERE Warehouse = Warehouses.Code
            )
      );

– works on MySQL
CREATE TEMPORARY TABLE all_info AS
SELECT w.*, b.code as box_code, b.contents, b.value AS box_value,
CASE
      WHEN COUNT(w.code)>capacity THEN 'Yes'
      ELSE 'No'
END AS saturation
FROM warehouses w
JOIN boxes b ON w.code=b.warehouse
GROUP BY warehouse;

UPDATE all_info
SET box_code ='', contents='', box_value=0
WHERE saturation='Yes';
```

```
SELECT *
FROM all_info;

-- with CTE for MySQL
WITH overcapacity AS (
SELECT w.code -- as warehouse_code, capacity, COUNT(b.code) AS number_boxes
FROM warehouses w
JOIN boxes b ON w.code=b.warehouse
GROUP BY w.code, capacity
HAVING COUNT(b.code) > capacity)

DELETE b
FROM boxes b
JOIN overcapacity o ON b.warehouse=o.code;



-- for SQLite
-- 3.15 Remove all boxes from saturated warehouses.
WITH overcapacity AS (
SELECT Warehouses.Code, Warehouses.Capacity,
CASE
WHEN COUNT(Boxes.Code)< Warehouses.Capacity THEN 'No'
ELSE 'Yes'
END AS 'saturation'
FROM Warehouses
JOIN Boxes ON Boxes.Warehouse = Warehouses.Code
GROUP BY Warehouses.Code, Warehouses.Capacity
HAVING COUNT(Boxes.Code) > Warehouses.Capacity)

DELETE
FROM Boxes
WHERE warehouse = (SELECT code FROM overcapacity);


-- 3.16 Add Index for column "Warehouse" in table "boxes"
    -- !!!NOTE!!!: index should NOT be used on small tables in practice
CREATE INDEX INDEX_WAREHOUSE
ON Boxes (Warehouse);

-- 3.17 Print all the existing indexes
    -- !!!NOTE!!!: index should NOT be used on small tables in practice
SELECT *
FROM SQLITE_MASTER
WHERE type = "index";

-- 3.18 Remove (drop) the index you added just
    -- !!!NOTE!!!: index should NOT be used on small tables in practice
DROP INDEX INDEX_WAREHOUSE;
```

DAY 8

-- 8.1 Obtain the names of all physicians that have performed a medical procedure they have never been certified to perform

```
Option 1 (MySQL)
SELECT DISTINCT P.Name, Pr.Code, Pr.Name as Procedure_Name, T.CertificationDate, U.Date as
Procedure_Date
```

```
FROM Physician P, Procedure Pr
LEFT JOIN Trained_In T ON T.Physician = P.EmployeeID AND Pr.Code = T.Treatment
LEFT JOIN Undergoes U ON U.Physician = P.EmployeeID AND U.Procedure=PR.Code
WHERE T.CertificationDate IS NULL AND U.Date is NOT NULL

Option 2 (MySQL)
SELECT p.EmployeeID, p.name, u.Procedure , pr.name, t.CertificationDate
FROM Physician p
JOIN Undergoes u ON p.EmployeeID=u.Physician
LEFT JOIN Trained_In t ON t.treatment=u.Procedure AND p.EmployeeID=t.Physician
JOIN Procedure pr ON pr.code=u.Procedure
WHERE t.CertificationDate IS NULL;



-- 8.3 Obtain the names of all physicians that have performed a medical procedure that
they are certified to perform,
-- but such that the procedure was done at a date (Undergoes.Date) after the
physician's certification expired
-- (Trained_In.CertificationExpires).
(MySQL)
SELECT p.name, a.name, certificationexpires, date
FROM physician p
JOIN trained_in t ON p.employeeid=t.physician
JOIN undergoes u ON p.employeeid=u.physician
JOIN procedures a ON a.code=t.treatment
WHERE date>certificationexpires;


-- 8.4 Same as the previous query, but include the following information in the
results: Physician name, name of procedure,
-- date when the procedure was carried out, name of the patient the procedure was
carried out on, and date when the
-- certification expired.

(MySQL)
SELECT p.name, a.name, certificationexpires, c.name patient_name, date
FROM physician p
JOIN trained_in t ON p.employeeid=t.physician
JOIN undergoes u ON p.employeeid=u.physician
JOIN patient c ON u.patient=c.SSN
JOIN procedures a ON a.code=t.treatment
WHERE date>certificationexpires;


DAY 10
using CTE
WITH full_list AS
(SELECT *
FROM PEOPLE
JOIN ADDRESS ON PEOPLE.ID = ADDRESS.ID)

SELECT
DISTINCT name,
FIRST_VALUE (updatedate) OVER (PARTITION BY name ORDER BY updatedate DESC) AS latest_date,
FIRST_VALUE(address) OVER (PARTITION BY name ORDER BY updatedate DESC) AS latest_address
FROM full_list
```

not using CTE

```
select distinct p.id, name,
first_value(address) over (partition by name order by updatedate desc
                    ) latest_address,
first_value(updatedate) over (partition by name --order by updatedate desc
                    ) latest_date
from people p
join address a on p.id=a.id
order by name
```