```
/*
   Databases Tutorial 1: SQL
   Sample Solution - 2018-11-09
*/

-----------------------------------------------------
/* Question 1: Basic
Given the following departments table:

CREATE TABLE Departments(
    department_id INT NOT NULL,
    department_name CHAR(50) NOT NULL,
    CONSTRAINT departments_pk PRIMARY KEY (department_id)
);

Create an SQL table called employees that stores employee number, employee name,
department, and salary information. The primary key for the employees table should
be the employee number. Create a foreign key on the employees table that references
the departments table based on the department_id field.
*/

CREATE TABLE Employees (
    employee_number INT PRIMARY KEY,
    employee_name TEXT NOT NULL,
    department INT REFERENCES Departments(department_id),
    salary INT NOT NULL
);

-- Or the following, if we want to use the "constraint form" explicitly.

CREATE TABLE Employees (
    employee_number INT NOT NULL,
    employee_name TEXT NOT NULL,
    department INT NOT NULL,
    salary INT NOT NULL,
    CONSTRAINT employees_pk PRIMARY KEY (employee_number),
    CONSTRAINT fk_departments FOREIGN KEY (department) REFERENCES Departments(
    department_id)
);

-----------------------------------------------------
/* Question 2: Supplier

Given the following schema:
Suppliers(_sid:integer_, sname:string, city:string, street:string)
Parts(_pid:integer_, pname:string, color:string)
Catalog(_sid:integer_, _pid:integer_, cost:real)

Find the names of all suppliers who have supplied a non-blue part.
*/

SELECT S.sname
FROM Suppliers S
WHERE S.sid IN (SELECT C.sid
    FROM Catalog C
    WHERE C.pid IN (SELECT P.pid
        FROM Parts P
        WHERE P.color <> 'Blue'
        )
    )
;

-----------------------------------------------------

/* Question 3: Employees
Consider the table Employees(_empId_, name, department, salary). The columns empId
and name are of type text, while department and salary are of type integer.
*/

-- a. Find the employees (from table employees) who get higher salary than anyone in
the department 5.
```

```sql

SELECT E.empId
FROM Employees E
WHERE NOT EXISTS (SELECT *
    FROM Employees S
    WHERE S.department = 5 AND S.salary >= E.salary
    )
;

-- b. Find max salary from each department.

SELECT department, max(salary)
FROM Employees
GROUP BY department
;

-- c. Find all employee records containing the word "Joe", regardless of whether it was stored as JOE, Joe, or joe.

SELECT *
FROM Employees
WHERE UPPER(name) LIKE '%JOE%'
;

--------------------------------------------------
/* Question 4: Company

For the following relation schema:
Employees(_employeeId_, employeeName, street, city)
Companies(_companyId_, companyName, city)
Works(_employee_, _company_, salary)
Manages(_manager_, _employee_)

The information on which company an employee works for and the current salary is stored in relation works. Assume that all people work for at most one company. The information on which employees have manager roles and who do they manage is stored in relation manages.

*/

-- a. Find the names, street address, and cities of residence for all employees who work for 'First Bank Corporation' and earn more than $10000.

SELECT E.employeeName, E.street, E.city
FROM Employees E, Works W
WHERE E.employeeId = W.employee AND W.company IN (SELECT companyId
    FROM Companies
    WHERE companyName = 'First Bank Corporation'
    )
    AND W.salary > 10000
;

-- b. Find the names of all employees in the database who live in the same cities as the companies for which they work.

SELECT E.employeeName
FROM Employees E, Works W, Companies C
WHERE E.employeeId = W.employee
    AND W.company = C.companyId
    AND E.city = C.city
;

-- c. Find the names of all employees in the database who live in the same cities and on the same streets as do their managers.

SELECT E.employeeName
FROM Employees E, Employees F, Manages M
WHERE E.employeeId = M.employee
    AND M.manager = F.employeeId
    AND E.city = F.city
    AND E.street = F.street
```

```
130 ;
131
132 -- OR
133
134 SELECT TEMP.employee
135 FROM (SELECT F.employeeName AS manager, E.employeeName AS employee, F.city AS
    manager_city, F.street AS manager_street, E.city AS employee_city, E.street AS
    employee_street
136     FROM Manages M, Employees E, Employees F
137     WHERE  M.employee = E.employeeId
138         AND M.manager = F.employeeId
139     ) AS TEMP
140 WHERE TEMP.manager_city = TEMP.employee_city
141     AND TEMP.manager_street = TEMP.employee_street;
142 ;
143
144 -- d. Find the names of all employees in the database who earn more than every
    employee of 'Small Bank Corporation'.
145
146 SELECT E.employeeName
147 FROM Employees E, Works W
148 WHERE E.employeeId = W.employee AND W.salary > (SELECT max(salary)
149         FROM Works W, Companies C
150         WHERE W.company = C.companyId AND C.companyName = 'Small Bank Corporation'
151         )
152 ;
153
154 -- Could also use "ALL" operator.
155
156 SELECT E.employeeName
157 FROM Employees E, Works W
158 WHERE E.employeeId = W.employee AND W.salary > ALL (SELECT salary
159         FROM Works W, Companies C
160         WHERE W.company = C.companyId AND C.companyName = 'Small Bank Corporation'
161         )
162 ;
163
164 -- e. Find the name of the company that has the smallest payroll (or total salaries
    of employees).
165
166 SELECT companyName
167 FROM Companies C, (SELECT company
168     FROM Works W
169     GROUP BY company
170     HAVING sum(salary) <= ALL (SELECT sum(salary) FROM Works GROUP BY company)
171     ) AS TEMP
172 WHERE C.companyId = TEMP.company;
173
174 -- f. Assuming that the table employee had an email column with NULL values, write a
    query to update the values.
175
176 UPDATE Employees
177 SET email = '...@xxx.com'
178 WHERE employeeId = '..'
179 ;
180
181 -- g. How do you find all employees who are not managers?
182
183 SELECT E.employeeId, E.employeeName
184 FROM Employees E
185 WHERE E.EmployeeId NOT IN (SELECT DISTINCT manager FROM Manages)
186 ;
187
188 ------------------------------------------
189
190 /* Question 5: Hospital
191
192 A database system used by a hospital to record information about patients and wards
    has the following relations:
193
194 Wards(number, numBeds)
```

```
Patients(pid, name, year, gender)
PatientInWard(pid, ward)
Tests(patient, testDate, testHour, temperature, heartRate)

A ward is identified by its number. Attribute numBeds is the number of beds in that
ward. Patients are identified by their personal identification number. The name,
year of birth and gender (`M' or `F') of each patient is stored in the Patients
relation.

The ward to which each patient is assigned is stored in relation PatientInWard.

During their stay in hospital, patients will undergo routine tests. The date and
hour of each occasion when these tests are performed on a patient are recorded, and
for each of these tests the patient's temperature and heart rate are measured and
recorded in the database. A patient will normally undergo these routine tests
several times during their stay in hospital.
*/

-- a. Find the temperature and heart rate measured in each test carried out on
patients born before 1950

SELECT temperature, heartRate
FROM Tests T, Patients P
WHERE T.patient = P.pid AND P.year < 1950
;

-- b. create a view FreeBeds(ward numBeds) where ward is a ward number, and numBeds
is the number of available beds in that ward

CREATE VIEW FreeBeds AS
    SELECT Wards.number AS ward, Wards.numBeds - count(PatientInWard.pid) AS numBeds
    FROM Wards LEFT OUTER JOIN PatientInWard ON Wards.number = PatientInWard.ward
    GROUP BY Wards.number
;

----------------------------------------

/* Question 7

We assume that all stars have different names, and that planet names are only unique
within their star-system. A star-system has exactly one star, all planets have
circular orbits around their star at different distances. A planet's position
indicates which order it has in the star-system, e.g. Earth is the 3rd planet around
the Sun, after Mercury and Venus. If a planet has O2 or other gases, it has an
atmosphere. Without an atmosphere, a planet has no gases. The surface of a planet is
either all water, all land, or a combination of water and land, but
nothing else.

Consider the relation
Planets(star, name, distance, mass, atmosphere, oxygen, water)
*/

-- a. Write an SQL table definition with reasonable types and constraints. Store
distance in millions of km (For Earth, you would store the value 149.6).

CREATE TABLE Planets(
    star TEXT NOT NULL,
    name TEXT NOT NULL,
    distance REAL NOT NULL CHECK (distance > 0),
    mass REAL NOT NULL CHECK (mass > 0),
    atmosphere BOOLEAN NOT NULL,
    oxygen REAL NOT NULL CHECK ((oxygen = 0 AND NOT atmosphere) OR (atmosphere AND
    oxygen >= 0)),
    water REAL NOT NULL,
    PRIMARY KEY (star, name),
    UNIQUE (star, distance)
);


-- b. Write an SQL query to determine how many planets are in orbits larger than the
orbit of the fictional planet "Duna" of the fictional star "Kerbol".
```

```sql

SELECT count(*)
FROM Planets
WHERE distance > (SELECT distance FROM Planets WHERE star = 'Kerbol' AND name =
'Duna')
;

/* We define a planet as "habitable" if it satisfies all these conditions:
- orbit at a distance (in millions of km) between 100 and 200 (inclusive) from its
star,
- has an atmosphere and it has an oxygen percentage between 15% and 25% (inclusive),
- has water on its surface.
*/

-- c. Write an SQL query which returns the star and name of a planet, as well as a
column status with value "habitable" if the planet is habitable, otherwise
"uninhabitable".

(SELECT star, name, 'habitable' AS status
 FROM Planets
 WHERE distance >= 100 AND
    distance <= 200 AND
    atmosphere AND
    oxygen >= 15 AND
    oxygen <= 25 AND
    water > 0
)
UNION
(SELECT star, name, 'uninhabitable' AS status
 FROM Planets
 WHERE NOT (
    distance >= 100 AND
    distance <= 200 AND
    atmosphere AND
    oxygen >= 15 AND
    oxygen <= 25 AND
    water > 0
    )
)
;

-- OR

WITH Habitables AS (SELECT star, name FROM planets WHERE distance >= 100 AND distance
 <= 200 AND atmosphere AND oxygen >= 15 AND oxygen <= 25 AND water > 0)
SELECT star, name, 'habitable' AS status
FROM Planets
WHERE (star, name) IN (SELECT star, name FROM Habitables)
UNION
SELECT star, name, 'uninhabitable' AS status
FROM Planets
WHERE (star, name) NOT IN (SELECT star, name FROM Habitables)
;

```