CREATE TABLE addresses (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(100) NOT NULL

);

CREATE TABLE categories (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(10) NOT NULL

);

CREATE TABLE clients (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`full\_name` VARCHAR(50) NOT NULL,

`phone\_number` VARCHAR(20) NOT NULL

);

CREATE TABLE drivers (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`first\_name` VARCHAR(30) NOT NULL,

`last\_name` VARCHAR(30) NOT NULL,

`age` INT NOT NULL,

`rating` FLOAT DEFAULT 5.5

);

CREATE TABLE cars (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`make` VARCHAR(20) NOT NULL,

`model` VARCHAR(20),

`year` INT DEFAULT 0 NOT NULL,

`mileage` INT DEFAULT 0,

`condition` CHAR(1) NOT NULL,

`category\_id` INT NOT NULL,

CONSTRAINT fk\_cars\_categories

FOREIGN KEY (`category\_id`)

REFERENCES `categories` (`id`)

);

CREATE TABLE courses (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`from\_address\_id` INT NOT NULL,

`start` DATETIME NOT NULL,

`bill` DECIMAL(10, 2) DEFAULT 10,

`car\_id` INT NOT NULL,

`client\_id` INT NOT NULL,

CONSTRAINT fk\_courses\_addresses

FOREIGN KEY (`from\_address\_id`)

REFERENCES `addresses` (`id`),

CONSTRAINT fk\_courses\_cars

FOREIGN KEY (`car\_id`)

REFERENCES `cars` (`id`),

CONSTRAINT fk\_courses\_clients

FOREIGN KEY (`client\_id`)

REFERENCES `clients` (`id`)

);

CREATE TABLE cars\_drivers (

`car\_id` INT NOT NULL,

`driver\_id` INT NOT NULL,

CONSTRAINT pk\_cars\_drivers

PRIMARY KEY (`car\_id`, `driver\_id`),

CONSTRAINT fk\_cars\_drivers\_cars

FOREIGN KEY (`car\_id`)

REFERENCES `cars` (`id`),

CONSTRAINT fk\_cars\_drivers\_drivers

FOREIGN KEY (`driver\_id`)

REFERENCES `drivers` (`id`)

);

-- When drivers are not working and need a taxi to transport them, they will also be registered

-- at the database as customers.

-- You will have to insert records of data into the clients table, based on the drivers table.

-- For all drivers with an id between 10 and 20 (both inclusive), insert data in the clients table with the following values:

-- • full\_name – get first and last name of the driver separated by single space

-- • phone\_number – set it to start with (088) 9999 and the driver\_id multiplied by 2

-- o Example – the phone\_number of the driver with id = 10 is (088) 999920

INSERT INTO clients (`full\_name`, `phone\_number`)

SELECT concat\_ws(' ', first\_name, last\_name) AS full\_name, concat('(088) 9999', id \* 2) AS phone\_number

FROM drivers

WHERE id BETWEEN 10 AND 20;

-- After many kilometers and over the years, the condition of cars is expected to deteriorate.

-- Update all cars and set the condition to be 'C'. The cars must have a mileage greater than 800000 (inclusive)

-- or NULL and must be older than 2010(inclusive).

-- Skip the cars that contain a make value of Mercedes-Benz. They can work for many more years.

UPDATE cars

SET `condition` = 'C'

WHERE mileage >= 800000 OR mileage IS NULL AND `year` <= 2010 AND `make` != 'Mercedes-Benz';

-- Some of the clients have not used the services of our company recently, so we need to remove them

-- from the database.

-- Delete all clients from clients table, that do not have any courses and the count of the characters in the full\_name

-- is more than 3 characters.

DELETE c.\* FROM clients AS c

LEFT JOIN courses AS cs ON c.id = cs.client\_id

WHERE cs.client\_id IS NULL AND char\_length(c.full\_name) > 3;

-- Extract the info about all the cars.

-- Order the results by car’s id.

-- Required Columns

-- • make

-- • model

-- • condition

SELECT make, model, `condition`

FROM cars

ORDER BY id;

-- Now, we need a more detailed information about drivers and their cars.

-- Select all drivers and cars that they drive. Extract the driver’s first and last name from the drivers table and the make,

-- the model and the mileage from the cars table. Order the result by the mileage in descending order,

-- then by the first name alphabetically.

-- Skip all cars that have NULL as a value for the mileage.

-- Required Columns

-- • first\_name

-- • last\_name

-- • make

-- • model

-- • mileage

SELECT d.first\_name, d.last\_name, c.make, c.model, c.mileage

FROM drivers AS d

JOIN cars\_drivers AS cd ON d.id = cd.driver\_id

JOIN cars AS c ON cd.car\_id = c.id

WHERE mileage IS NOT NULL

ORDER BY c.mileage DESC, d.first\_name;

-- Extract from the database all the cars and the count of their courses. Also display the average

-- bill of each course by the car, rounded to the second digit.

-- Order the results descending by the count of courses, then by the car’s id.

-- Skip the cars with exactly 2 courses.

-- Required Columns

-- • car\_id

-- • make

-- • mileage

-- • count\_of\_courses

-- • avg\_bill

SELECT c.id AS car\_id, c.make, c.mileage, count(cs.id) AS count\_of\_courses, round(avg(cs.bill) , 2) AS avg\_bill

FROM cars AS c

LEFT JOIN courses AS cs ON c.id = cs.car\_id

GROUP BY c.id

HAVING count\_of\_courses != 2

ORDER BY count\_of\_courses DESC, c.id;

-- Extract the regular clients, who have ridden in more than one car. The second letter of the customer's full name must be 'a'.

-- Select the full name, the count of cars that he ridden and total sum of all courses.

-- Order clients by their full\_name.

-- Required Columns

-- • full\_name

-- • count\_of\_cars

-- • total\_sum

SELECT c.full\_name, count(ca.model) AS count\_of\_cars, SUM(cs.bill) AS total\_sum

FROM clients AS c

JOIN courses AS cs ON c.id = cs.client\_id

JOIN cars AS ca ON cs.car\_id = ca.id

GROUP BY c.id

HAVING substr(c.full\_name, 2, 1) = 'a' AND count\_of\_cars > 1

ORDER BY c.full\_name;

-- The headquarters want us to make a query that shows the complete information about all courses in the database.

-- The information that they need is the address, if the course is made in the Day (between 6 and 20(inclusive both))

-- or in the Night (between 21 and 5(inclusive both)), the bill of the course, the full name of the client, the car maker,

-- the model and the name of the category.

-- Order the results by course id.

-- Required Columns

-- • name (address)

-- • day\_time

-- • bill

-- • full\_name (client)

-- • make

-- • model

-- • category\_name (category)

SELECT a.name,

CASE

WHEN hour(cs.start) BETWEEN 6 AND 20 THEN 'Day'

ELSE 'Night'

END AS day\_time, cs.bill, cl.full\_name, ca.make, ca.model, cat.name

FROM addresses AS a

JOIN courses AS cs ON a.id = cs.from\_address\_id

JOIN clients AS cl ON cs.client\_id = cl.id

JOIN cars AS ca ON cs.car\_id = ca.id

JOIN categories AS cat ON ca.category\_id = cat.id

ORDER BY cs.id;

-- Create a user defined function with the name udf\_courses\_by\_client (phone\_num VARCHAR (20)) that receives a client’s

-- phone number and returns the number of courses that clients have in database.

DELIMITER $$

CREATE FUNCTION udf\_courses\_by\_client (phone\_num VARCHAR (20))

RETURNS INT

DETERMINISTIC

BEGIN

RETURN (SELECT count(cs.id)

FROM clients AS c

JOIN courses AS cs ON c.id = cs.client\_id

WHERE c.phone\_number = phone\_num);

END

$$

DELIMITER ;

SELECT udf\_courses\_by\_client ('(803) 6386812') as `count`;

-- Create a stored procedure udp\_courses\_by\_address which accepts the following parameters:

-- • address\_name (with max length 100)

-- Extract data about the addresses with the given address\_name. The needed data is the name of the address,

-- full name of the client, level of bill (depends of course bill – Low – lower than 20(inclusive), Medium – lower than 30(inclusive),

-- and High), make and condition of the car and the name of the category.

-- Order addresses by make, then by client’s full name.

-- Required Columns

-- • name (address)

-- • full\_name

-- • level\_of\_bill

-- • full\_name (client)

-- • make

-- • condition

-- • cat\_name (category)

DELIMITER $$

CREATE PROCEDURE udp\_courses\_by\_address (address\_name VARCHAR(100))

BEGIN

SELECT a.name, cl.full\_name,

CASE

WHEN cs.bill <= 20 THEN 'Low'

WHEN cs.bill <= 30 THEN 'Medium'

ELSE 'High'

END AS level\_of\_bill,

ca.make, ca.condition, cat.name AS cat\_name

FROM addresses AS a

JOIN courses AS cs ON a.id = cs.from\_address\_id

JOIN clients AS cl ON cs.client\_id = cl.id

JOIN cars AS ca ON cs.car\_id = ca.id

JOIN categories AS cat ON ca.category\_id = cat.id

WHERE a.name = address\_name

ORDER BY ca.make, cl.full\_name;

END

$$

CALL udp\_courses\_by\_address('66 Thompson Drive');