Assignment

Structured Query Language

Department of Computer Science
Jönköping University

Tuwe Löfström & Vi Tran
Tuwe.lofstrom@ju.se
huynhkhanhvi.tran@ju.se

Innehåll

1 Introduction

In this laboratory you will work with a database for a car booking system. You must create the car booking system's tables, enter data in each table and write a number of search questions. The tables are created in the same database you worked with during the workshops.

2 The car booking system

The car booking system has a number of *Vehicles* that are rented out from various *Rental Locations* to the company's customers. When a *Customer* books a vehicle, a *Booking* is created in the database with associated information. A booking contains, among other things, information about the customer number, the vehicle's registration number, the rental period (from and to date of the booking) and which rental location the vehicle is to be picked up at or returned to. Since a vehicle can be returned to a different rental location than the pick-up location, the *Vehicle Location* table exists to keep track of which rental location a vehicle is at as of a certain date. A booking can assume three different *Booking Statuses* (Booked, Collected, Returned). Similarly, a vehicle can have one of three *Vehicle Statuses* (Available, Maintenance, Expired) at any given time. A vehicle that has expired or is undergoing maintenance is not available for rental. A vehicle also has a certain *Vehicle Type* (Small, Medium, Large, Minibus, Environmental). The schedule for the car booking system is shown in Appendix A.

Below follows a description of each table. The tables are specified both as relations with attributes of the form **Relation**(Attribute1, Attribute2, ..., AttributeN) where the primary key is underlined, and as tables with columns in graphic form. In the graphical representation, **PK** means that a column is the primary key of the table (PK plus several columns indicates a composite primary key) and **FK** plus a column means that the column is a foreign key (in the diagram this referential integrity condition is shown as an arrow from the table with the foreign key to the table with the column referenced by the foreign key). The column in the home table referenced by the foreign key is specified in the description below. The data types are also shown alongside each column name. The three numeric data types used are INT (integer), BIT (can only assume the values 0 and 1) and DECIMAL(7,2) which specifies a decimal number with a maximum length of 7 including 2 decimal places, i.e. the largest positive number that the data type can store is 99999.99. Only the two string data types CHAR(*n*) and VARCHAR(*n*) are used, where n indicates the number of characters the data type can store, e.g. so a CHAR(5) stores five characters and a VARCHAR(5) stores a maximum of five characters. If a smaller number of characters than n is stored in a column of data type CHAR(*n*), then the column is filled with spaces until n characters are obtained. A VARCHAR(*n*), on the other hand, does not fill with spaces, i.e. it has a variable length, however, with a maximum of n characters. For example, the text 'abc' is stored as 'abc' for a CHAR(5) and as 'abc' for a VARCHAR(5). Finally, the DATETIME data type is used to store the date and time. A column in the graph that does not accept NULL values (ie NOT NULL) is indicated in bold, otherwise the column accepts NULL values.

Customer(<u>CustomerNumber</u>, FirstName, LastName, Street, PostalCode, City, TelephoneNumber)

Customer		
PK	<u>**CustomerNumber**</u>	**NUMBERPS(38;0)**

	FirstName
	LastName
	Street
	PostalCode
	City
	TelephoneNumber
		VARCHAR2(20)
		VARCHAR2(25)
		VARCHAR2(50)
		CHAR(5)
		VARCHAR2(20)
		VARCHAR2(10)

The table *Customer* contains information about the company's customers, where *CustomerNumber* is the primary key for the table. All columns do not accept NULL.

RentalLocation(<u>RentalLocationNumber</u>, Name, Street, PostalCode, City, TelephoneNumber, Email, Website)

RentalLocaxtion		
PK	<u>**RentalLocationNumber**</u>	**NUMBERPS(38;0)**

	Name
	Street
	PostalCode
	City
	TelephoneNumber
	Email
	Webbsite
		VARCHAR2(30)
		VARCHAR2(50)
		CHAR(5)
		VARCHAR2(20)
		VARCHAR2(10)
		VARCHAR2(40)
		VARCHAR2(50)

The *RentalLocation* table contains information about the company's rental locations, where *RentalLocationNumber* is the primary key for the table. All columns except Website do not accept NULL.

VehicleType(<u>VehicleTypeCode</u>, Type)

VehicleType		
PK	<u>**VehicleTypeCode**</u>	**NUMBERPS(38;0)**
	Type	**VARCHAR2(30)**

The *VehicleType* table contains information about a vehicle type (Small, Medium, Large, Minivan, Renter), where the *VehicleTypeCode* is the primary key for the table. All columns do not accept NULL.

VehicleStatus(<u>VehicleStatusCode</u>, Status)

VehicleStatus		
PK	<u>**VehicleStatusCode**</u>	**NUMBERPS(38;0)**
	Status	**VARCHAR2(15)**

The *VehicleStatus* table contains information about a vehicle's status (Available, Maintenance, Expired), where the *VehicleStatusCode* is the primary key for the table. All columns do not accept NULL.

Vehicle(<u>RegistrationNumber</u>, VehicleTypeCode, Model, Mileage, DailyRent, NumberOfPassengers, Automatic, VehicleStatusCode)

Vehicle		
PK	**<u>RegistrationNumber</u>**	**CHAR(6)**
FK1	**VehicleTypeCode**	**NUMBERPS(38;0)**
	Model	**VARCHAR2(50)**
	Mileage	**NUMBERPS(38;0)**
	DailyRent	**NUMBERPS(7;2)**
	NumberOfPassengers	**NUMBERPS(38;0)**
	Automatic	**CHAR(1)**
FK2	**VehicleStatusCode**	**NUMBERPS(38;0)**

The table *Vehicles* contains information about the company's vehicles, where the vehicle's *RegistrationNumber* is the primary key for the table. *VehicleTypeCode* is a foreign key that references *VehicleTypeCode* in the *VehicleType* table. *VehicleStatusCode* is a foreign key that references *VehicleStatusCode* in the *VehicleStatus* table. All columns do not accept NULL.

BookingStatus(<u>BookingStatusCode</u>, Status)

BookingStatus		
PK	**<u>BookingStatusCode</u>**	**NUMBERPS(38;0)**
	Status	**VARCHAR2(15)**

The table *BookingStatus* contains information about the status of a booking (Booked, Collected, Returned), where *BookingStatusCode* is the primary key for the table. All columns do not accept NULL.

Booking(<u>BookingNumber</u>, CustomerNumber, RegistrationNumber, BookingStatusCode, FromDate, ToDate, PickupLocationNumber, ReturnLocationNumber)

Booking		
PK	**<u>BookingNumber</u>**	**NUMBERPS(38;0)**
FK1	**CustomerNumber**	**NUMBERPS(38;0)**
FK2	**RegistrationNumber**	**CHAR(6)**
FK3	**BookingStatusCode**	**NUMBERPS(38;0)**
	FromDate	**DATE**
	ToDate	**DATE**
FK4	**PickingLocationNumber**	**NUMBERPS(38;0)**
FK5	**ReturnLocationNumber**	**NUMBERPS(38;0)**

The *Booking* table contains information about the customers' bookings of the company's vehicles, where the *BookingNumber* is the primary key for the table. *CustomerNumber* is a foreign key that refers to *CustomerNumber* in the *Customer* table. *RegistrationNumber* is a foreign key that references *RegistrationNumber* in the *Vehicle* table. *BookingStatusCode* is a foreign key that references *BookingStatusCode* in the *BookingStatus* table. *PickupLocationNumber* is a foreign key that references *RentalLocationNumber* in the RentalLocation table. *ReturnLocationNumber* is a foreign key that references *RentalLocationNumber* in the *RentalLocation* table. All columns do not accept NULL. When a customer books a vehicle, a new row is stored in the table, where the rental period is stored in *FromDate* and *ToDate* (both date and time). If the customer wants to return the vehicle at a rental location other than the pick-up location, the *ReturnLocationNumber* differs from the *PickupLocationNumber*.

VehicleLocation(<u>RegistrationNumber, FromDate</u>, RentalLocationNumber)

VehicleLocation		
PK,FK1 **PK**	**<u>RegistrationNumber</u>** **<u>FromDate</u>**	**CHAR(6)** **DATE**
FK2	**RentalLocationNumber**	**NUMBERPS(38;0)**

The *VehicleLocation* table contains information about which rental location a vehicle is at from a certain date, where the *RegistrationNumber* together with the *FromDate* form the composite primary key for the table. *RegistrationNumber* is a foreign key that references *RegistrationNumber* in the *Vehicle* table. *RentalLocationNumber* is a foreign key that references *RentalLocationNumber* in the *RentalLocation* table. All columns do not accept NULL. When a customer books a vehicle where the return location is different from the pick-up location, a new row is stored in the table. The *FromDate* in the *VehicleLocation* table gets the same value as the booking's *ToDate* in the *Booking* table and the *RentalLocationNumber* in the *VehicleLocation* table gets the same value as the booking's *ReturnLocationNumber* in the *Booking* table.

3 Data

The instance of the database you will work with is shown below.

Customer

Customer Number	*FirstName*	*LastName*	*Street*	*Postal Code*	*City*	*TelephoneNumber*
1	Sven	Svensson	Solbacken 5	50636	Borås	033123456
2	Lotta	Olsson	Kyrkogatan 11	50912	Borås	033112233
3	Anders	Andersson	Sandvägen 7	41137	Göteborg	031332211
4	Tore	Toresson	Sjögatan 3	50812	Borås	033542312
5	Ulla	Larsson	Fågelvägen 15	40625	Göteborg	033247278
6	Svante	Svantesson	Kungsgatan 1	50711	Borås	033772058
7	Johan	Johansson	Karlavägen 23	10342	Stockholm	0863051122

RentalLocation

Rental Location Number	*Renter*	*Street*	*Postal Code*	*City*	*Telephone Number*	*Email*	*Webbsite*
1	RL Borås	Vintergatan 5	50700	Borås	033500500	info@rl.se	www.rl.se/bs
2	RL Göteborg	Solgatan 2	40100	Göteborg	031300300	info@rl.se	wwwrl.se/gbg
3	RL Stockholm	Vårgatan 6	10425	Stockholm	0810005000	info@rl.se	*NULL*

VehicleType

VehicleTypeCode	*Type*
1	Small
2	Medium
3	Large
4	Minivan
5	Renter

VehicleStatus

VehicleStatusCode	*Status*
1	Available
2	Maintenance
3	Expired

Vehicle

Registration Number	*Vehicle Type Code*	*Model*	*Mileage*	*Daily Rent*	*NumberOf Passengers*	*Automatic*	*VehicleStatus Code*
ABC123	1	Toyota Aygo 1.0	1000	1000.00	4	0	1
ADN274	2	Volkswagen Passat 2.0	1000	1350.00	5	0	1
AVE693	2	Opel Insignia	1500	1470.00	5	0	1
BUS104	4	Volkswagen Sharan	3000	2000.00	7	0	1
GPS935	3	Hyundai Sonata	1500	1750.00	5	1	2
HAP555	3	Volvo V70	2000	1630.00	5	0	1
KAD395	4	Volkswagen Transporter	2500	2150.00	9	1	1
LJF599	1	Volkswagen Golf 1.6	3000	1300.00	5	1	1
PPP409	5	Saab 9-3	1000	1550.00	5	0	1

Booking

Booking Number	*Customer Number*	*Registration Number*	*Booking Status Code*	*FromDate*	*ToDate*	*Picking Location Number*	*Return Location Number*
1	1	ADN274	3	2022-03-25 09:00	2022-04-05 09:00	1	1
2	2	KAD395	3	2022-05-01 18:00	2022-05-15 18:00	1	2
3	3	PPP409	3	2022-05-03 10:00	2022-05-10 18:00	2	2
4	5	PPP409	3	2022-05-11 18:00	2022-05-13 18:00	2	2
5	4	LJF599	3	2022-05-07 08:00	2022-05-09 19:00	1	1
6	1	LJF599	2	2022-05-15 06:00	2022-05-15 19:00	1	1
7	5	HAP555	1	2022-07-05 09:00	2022-07-12 09:00	2	1
8	3	BUS104	1	2022-07-12 09:00	2022-07-17 09:00	2	2
9	1	ADN274	1	2022-07-11 09:00	2022-07-11 15:00	1	1
10	6	AVE693	1	2022-07-01 09:00	2022-07-21 09:00	1	1

VehicleLocation

RegistrationNumber	*FromDate*	*RentalLocationNumber*
ABC123	2022-01-01 00:00	2
ADN274	2022-01-01 00:00	1
AVE693	2022-01-01 00:00	1
BUS104	2022-01-01 00:00	2
GPS935	2022-01-01 00:00	1
HAP555	2022-01-01 00:00	2
KAD395	2022-01-01 00:00	1
LJF599	2022-01-01 00:00	1
PPP409	2022-01-01 00:00	2
KAD395	2022-05-15 18:00	2
HAP555	2022-07-12 09:00	1

BookingStatus

BookingStatusCode	*Status*
1	Booked
2	Collected
3	Returned

4 Tasks

Below are six tasks, which in turn are divided into a number of sub-tasks. Create a script for each task, ie the files must be named `LAB1.sql`, `LAB2.sql`, `LAB3.sql`, `LAB4.sql`, `LAB5.sql` and `LAB6.sql`.

Remember to make copies of, and save, the files at regular intervals!

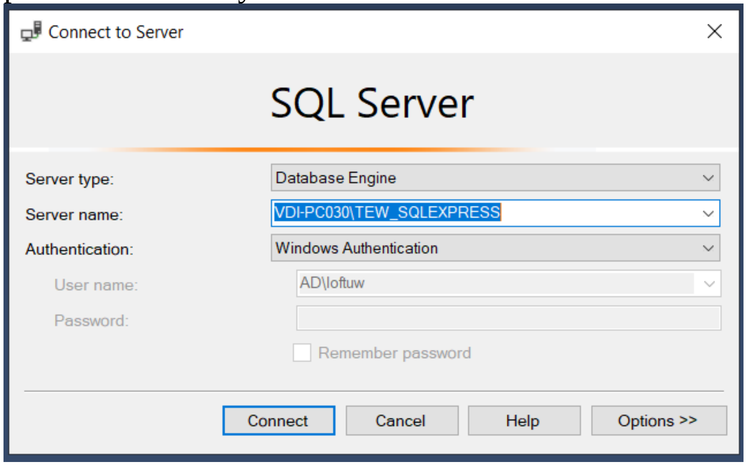
Of course, you yourself must test that the SQL files work before submitting them, but also keep in mind that a correct search is general, i.e. works in all possible instances of the database.

If a solution contains a subquery, but could have been solved without it, this is counted as an incorrect solution even if it gives the correct result.

Make sure that the group number and the names of all group members are entered as SQL comments at the beginning of each script file. A comment in the SQL language is created by entering two hyphens at the beginning of a line (then the entire line becomes a comment) or by enclosing the comment between /* and */, e.g.:

```
-- Group xx: John Doe, Mr Niemand
/* Group xx: John Doe, Mr Niemand */
```

Use the Editor in *Microsoft SQL Server Management Studio* to create and save the respective scripts. This is done by:


Connect to Server

SQL Server

Server type: Database Engine
Server name: VDI-PC030\TEW_SQLEXPRESS
Authentication: Windows Authentication
User name: AD\loftuw
Password:
Remember password

1. Start MS SQL Server Management Studio and connect to the Microsoft SQL Server instance
   SQLEXPRESS.

   ATTENTION! In the login window, "Server type" should be Database Engine and "Authentication" should be Windows Authentication.

2. Expand the "Databases" node in the *Object Explorer* and create a *New Database* from the context menu.
3. Right-click your newly created database and select *New Query* from the menu.
4. This opens a new tab in the Editor where the SQL code for the script can be entered.
5. To save the script, click the 💾-button in the toolbar below the main menu.

4.1 Task 1 (DDL)

The SQL code below must be saved in the script file LAB1.SQL.

a) Write SQL code to create the tables according to the scheme in Appendix A and the description in Chapter 2. Keep in mind that the order in which the tables are created matters because of the foreign keys.

b) Write SQL code (an ALTER TABLE statement) that adds a condition that ensures that the *ToDate* is later than the *FromDate* in the *Booking* table.

c) Write SQL code that creates the *AvailableVehicles* view that lists all available vehicles (i.e. do not show vehicles that have expired or are undergoing maintenance). Show *RegistrationNumber*, *Model*, *NumberOfPassengers* and *Automatic* (if you want, you can solve this subtask after you have added data to the tables in task 2).

The script file will thus contain a number of CREATE TABLE statements, an ALTER TABLE statement and a CREATE VIEW statement. If you receive an error message that e.g. a CREATE VIEW must come first in a SQL batch (ie the database manager interprets all the SQL code in your script file as a single "batch" of SQL commands), so you can write the reserved word GO on a new line after each SQL statement in the script file. Then each SQL statement is executed as a separate "batch", e.g.:

```
CREATE TABLE T
(
   column1 INT NULL
)
GO

ALTER TABLE T
ALTER COLUMN column1 INT NOT NULL
GO

CREATE VIEW V (column1)
AS
   SELECT 'Hello World'
GO
```

If you run your script file several times, remember that you must first delete all the objects you have created (e.g. tables) before you can create them again.

4.2 Task 2 (INSERT)

The SQL code below must be saved in the script file LAB2.SQL.

Create the database instance shown in chapter 3, i.e., write SQL code that adds all the rows to the respective table. Keep in mind that the order in which the data is added to the different tables matters because of the foreign keys.

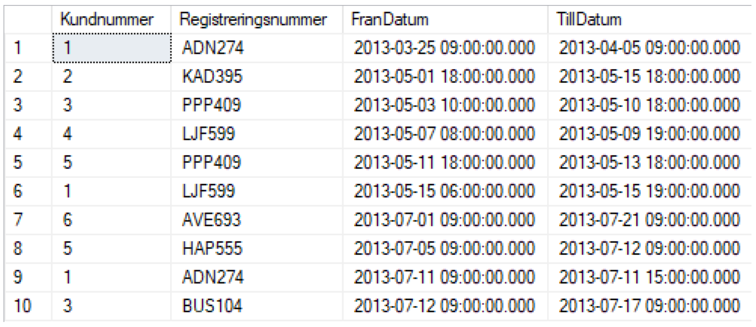
The script file will thus contain a number of INSERT INTO statements.

4.3 Task 3 (Simple SELECT with ORDER BY)

Create the script file LAB3.SQL that contains SQL expressions corresponding to the search queries below. Write a comment before each statement in the script file indicating which subtask (a-d) it answers.

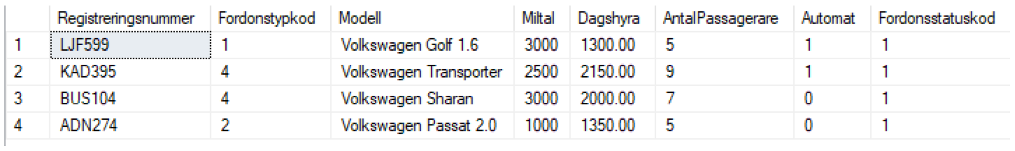
Keep in mind that a correct search is general, i.e. works in all possible instances of the database, regardless of the data contained in it.

a) Write an SQL statement that gives a list of all bookings. The result must contain *CustomerNumber*, *RegistrationNumber*, *FromDate* and *ToDate*. Sort on *FromDate* in ascending order and then on *ToDate* in ascending order.



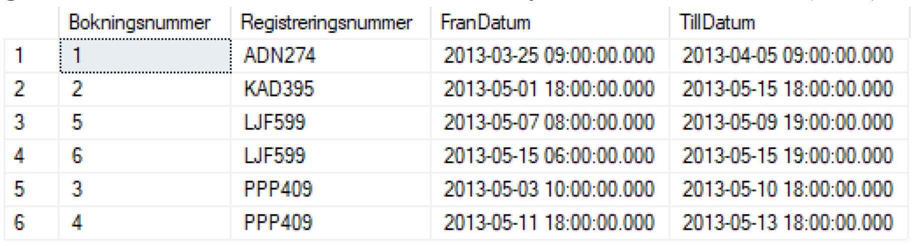
	Kundnummer	Registreringsnummer	FranDatum	TillDatum
1	1	ADN274	2013-03-25 09:00:00.000	2013-04-05 09:00:00.000
2	2	KAD395	2013-05-01 18:00:00.000	2013-05-15 18:00:00.000
3	3	PPP409	2013-05-03 10:00:00.000	2013-05-10 18:00:00.000
4	4	LJF599	2013-05-07 08:00:00.000	2013-05-09 19:00:00.000
5	5	PPP409	2013-05-11 18:00:00.000	2013-05-13 18:00:00.000
6	1	LJF599	2013-05-15 06:00:00.000	2013-05-15 19:00:00.000
7	6	AVE693	2013-07-01 09:00:00.000	2013-07-21 09:00:00.000
8	5	HAP555	2013-07-05 09:00:00.000	2013-07-12 09:00:00.000
9	1	ADN274	2013-07-11 09:00:00.000	2013-07-11 15:00:00.000
10	3	BUS104	2013-07-12 09:00:00.000	2013-07-17 09:00:00.000

b) Write an SQL statement that lists all vehicles that have the word "Volkswagen" anywhere in the *Model* column. All columns in the *Vehicle* table must be displayed in the result and the list must be sorted by *RegistrationNumber* in descending order.



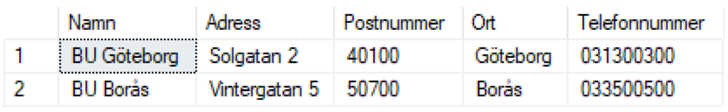
	Registreringsnummer	Fordonstypkod	Modell	Miltal	Dagshyra	AntalPassagerare	Automat	Fordonsstatuskod
1	LJF599	1	Volkswagen Golf 1.6	3000	1300.00	5	1	1
2	KAD395	4	Volkswagen Transporter	2500	2150.00	9	1	1
3	BUS104	4	Volkswagen Sharan	3000	2000.00	7	0	1
4	ADN274	2	Volkswagen Passat 2.0	1000	1350.00	5	0	1

c) Write an SQL sentence that gives a list of all bookings whose *FromDate* is between February 1 and March 31, 2013, and whose *FromDate* falls in May regardless of the year. The columns *BookingNumber*, *RegistrationNumber*, *FromDate* and *ToDate* must be included in the result. Sort by *RegistrationNumber* in ascending order. TIP: The month number for a given date can be obtained with the SQL function MONTH(date).



	Bokningsnummer	Registreringsnummer	FranDatum	TillDatum
1	1	ADN274	2013-03-25 09:00:00.000	2013-04-05 09:00:00.000
2	2	KAD395	2013-05-01 18:00:00.000	2013-05-15 18:00:00.000
3	5	LJF599	2013-05-07 08:00:00.000	2013-05-09 19:00:00.000
4	6	LJF599	2013-05-15 06:00:00.000	2013-05-15 19:00:00.000
5	3	PPP409	2013-05-03 10:00:00.000	2013-05-10 18:00:00.000
6	4	PPP409	2013-05-11 18:00:00.000	2013-05-13 18:00:00.000

d) Write an SQL statement that gives a list of all rental locations that have a web page (ie where *Website* is not equal to NULL). Show *Renter*, *Street*, *PostalCode*, *City* and *TelephoneNumber* in the result. Sort by *Renter* in descending order.



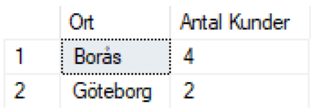
	Namn	Adress	Postnummer	Ort	Telefonnummer
1	BU Göteborg	Solgatan 2	40100	Göteborg	031300300
2	BU Borås	Vintergatan 5	50700	Borås	033500500

4.4 Task 4 (SELECT with GROUP BY and HAVING)

Create the script file LAB4.SQL that contains SQL expressions corresponding to the search queries below. Write a comment before each statement in the script file indicating which subtask (a-d) it answers.

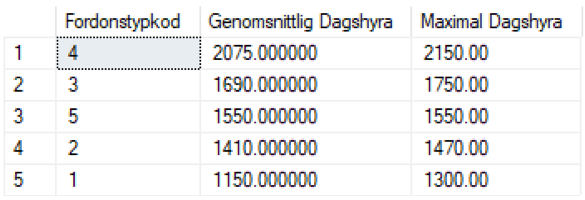
Keep in mind that a correct search is general, i.e. works in all possible instances of the database, regardless of the data contained in it.

a) Write an SQL statement that lists the number of customers per *City* who do not live in Stockholm. Display *City* and the number of customers per city as [*Number of Customers*].



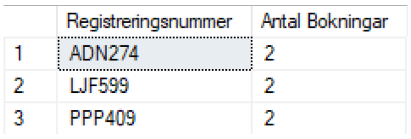
	Ort	Antal Kunder
1	Borås	4
2	Göteborg	2

b) Write an SQL sentence that reports the average and maximum daily rent per vehicle type. The list must show the *VehicleTypeCode*, the average daily rent as [*Average Daily Rent*] and the maximum daily rent as [*Maximum Daily Rent*]. The list must be sorted by the average daily rent in descending order.



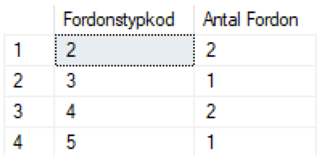
	Fordonstypkod	Genomsnittlig Dagshyra	Maximal Dagshyra
1	4	2075.000000	2150.00
2	3	1690.000000	1750.00
3	5	1550.000000	1550.00
4	2	1410.000000	1470.00
5	1	1150.000000	1300.00

c) You want to find out how many bookings there are per vehicle. Write an SQL statement that accomplishes this. Only the vehicles that have been booked at least 2 times must be included in the list. Show *RegistrationNumber* and number of bookings as [*Number of Bookings*].



	Registreringsnummer	Antal Bokningar
1	ADN274	2
2	LJF599	2
3	PPP409	2

d) Write an SQL statement that lists the number of available vehicles per vehicle type (ie do not include vehicles that have expired or are undergoing maintenance in the number). List only vehicle types with an average daily rental greater than 1200. Display *VehicleTypeCode* and the number of vehicles per vehicle type as [*Number of Vehicles*]



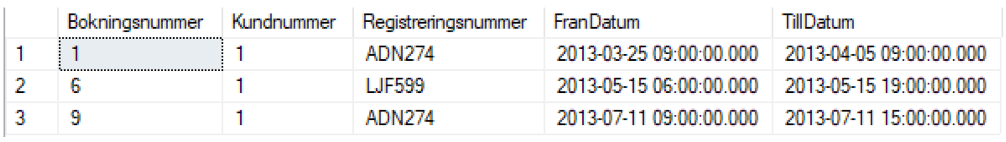
	Fordonstypkod	Antal Fordon
1	2	2
2	3	1
3	4	2
4	5	1

4.5 Task 5 (Subqueries and JOIN)

Create the script file LAB5.SQL that contains SQL expressions corresponding to the search queries below. Write a comment before each statement in the script file indicating which subtask (a-d) it answers.

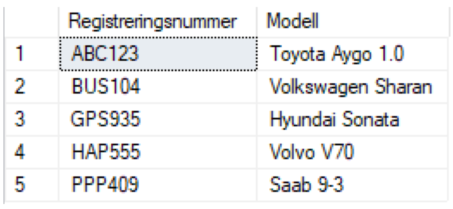
Keep in mind that a correct search is general, i.e. works in all possible instances of the database, regardless of the data contained in it.

a) Write an SQL statement that gives a list of all bookings that the customer with phone number '033123456' has made. Show Booking number, Customer number, *RegistrationNumber*, *FromDate* and *ToDate* in the result.



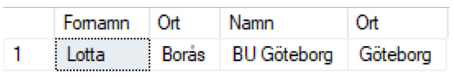
	Bokningsnummer	Kundnummer	Registreringsnummer	FranDatum	TillDatum
1	1	1	ADN274	2013-03-25 09:00:00.000	2013-04-05 09:00:00.000
2	6	1	LJF599	2013-05-15 06:00:00.000	2013-05-15 19:00:00.000
3	9	1	ADN274	2013-07-11 09:00:00.000	2013-07-11 15:00:00.000

b) Write an SQL statement that gives a list of all vehicles that have no bookings from customers in Borås (NOTE! the query must also show the vehicles for which there are no bookings in the Booking table). Show *RegistrationNumber* and *Model* in the result and sort by *RegistrationNumber* in ascending order.



	Registreringsnummer	Modell
1	ABC123	Toyota Aygo 1.0
2	BUS104	Volkswagen Sharan
3	GPS935	Hyundai Sonata
4	HAP555	Volvo V70
5	PPP409	Saab 9-3

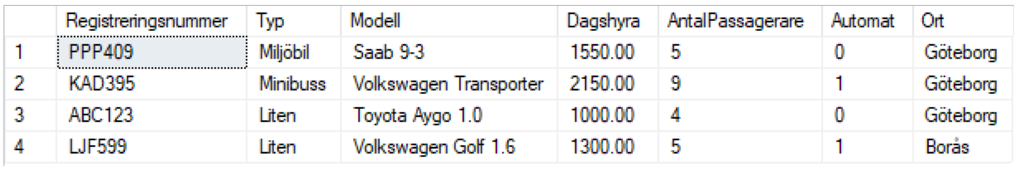
c) Write an SQL statement that gives a list of the customers who have returned a vehicle in a different *City* than where they live. Enter the customer's *FirstName* and *City* as well as the *Renter* and *City* of the rental location.



	Fornamn	Ort	Namn	Ort
1	Lotta	Borås	BU Göteborg	Göteborg

d) List all available vehicles that are free between '2022-07-07 17:00' and '2022-07-15 17:00'. Show *RegistrationNumber*, *Type*, *Model*, *DailyRent*, *NumberOfPassengers*, *Automatic* and *Location* of the rental location in the result. Sort by *VehicleTypeCode* in descending order and then by *DailyRent* in ascending order.

Subtask d) is quite extensive, so you will get some help here. Keep in mind that you must first know at which rental location the respective vehicle is located on the date '2022-07-07 17:00'. Use a subquery in the outer query's FROM clause to return the *RegistrationNumber* and the latest *FromDate* for each vehicle in the *VehicleLocation* table where *FromDate* is earlier or equal to '2022-07-07 17:00' (you need a GROUP BY in the subquery). You also need to create an alias for the subquery. The alias is specified directly after the subquery according to the syntax "FROM (subquery) x", where x is an alias for the subquery. Next, you must JOIN the subquery with the *VehicleLocation* table on *RegistrationNumber* **and** *FromDatum*. Only then is it obtained at which rental location the respective vehicle is available on the date '2022-07-07 17:00'. You then need to JOIN a further number of tables. You also need to ensure that each vehicle is free between '2022-07-07 17:00' and '2022-07-15 17:00'. To do this, you need to use a subquery in the outer query's WHERE clause. Don't forget that the respective vehicle must also be available (i.e. don't bring vehicles that have expired or are undergoing maintenance).



	Registreringsnummer	Typ	Modell	Dagshyra	AntalPassagerare	Automat	Ort
1	PPP409	Miljöbil	Saab 9-3	1550.00	5	0	Göteborg
2	KAD395	Minibuss	Volkswagen Transporter	2150.00	9	1	Göteborg
3	ABC123	Liten	Toyota Aygo 1.0	1000.00	4	0	Göteborg
4	LJF599	Liten	Volkswagen Golf 1.6	1300.00	5	1	Borås

4.6 Task 6 (UPDATE and DELETE)

Keep in mind that correct updates are general, ie. works in all possible instances of the database, regardless of what data is already in it.

a) Write an SQL statement that changes the zip code to '10420' for all rental locations that have no website (ie where Website is NULL).

b) Write an SQL statement that increases the number of miles by 10 miles for all booked vehicles that have been returned.

c) Write an SQL statement that removes all customers from the Customer table who have not booked any vehicles.

Remember that you can restore the contents of your tables (as they looked after the insertion in task 2) by first emptying the tables with the DELETE FROM statement and then running the script file from task 2 again. In that case, you also need to empty the tables and add data to the tables in a certain order due to the foreign keys.

5 Additional information

5.1 Tutoring

Supervision of the laboratory takes place on four occasions. All opportunities may, but do not have to, be used. The tutorial does not need to be booked for the first two occasions, but you all work with the laboratories in your groups in front of one or two computers, where the teacher is present during the entire tutorial session. During the tutoring session, each group can ask questions about a concept to the teacher, which is then explained by the teacher to everyone present at the tutoring session. The other two tutoring sessions take place in groups, where each group can book a tutoring session on PingPong, under Activity/Tutorial/Lab1 (the booking links become visible in connection with each tutoring session).

See the schedule for available times. Booking will be made available through the Canvas Calendar.

5.2 Submission

The assignment is submitted via Canvas, under Assignment SQL, where all script files are submitted as an archive file (zip file or rar file). Don't forget to enter the group number and the names of all group members as a comment in all script files!

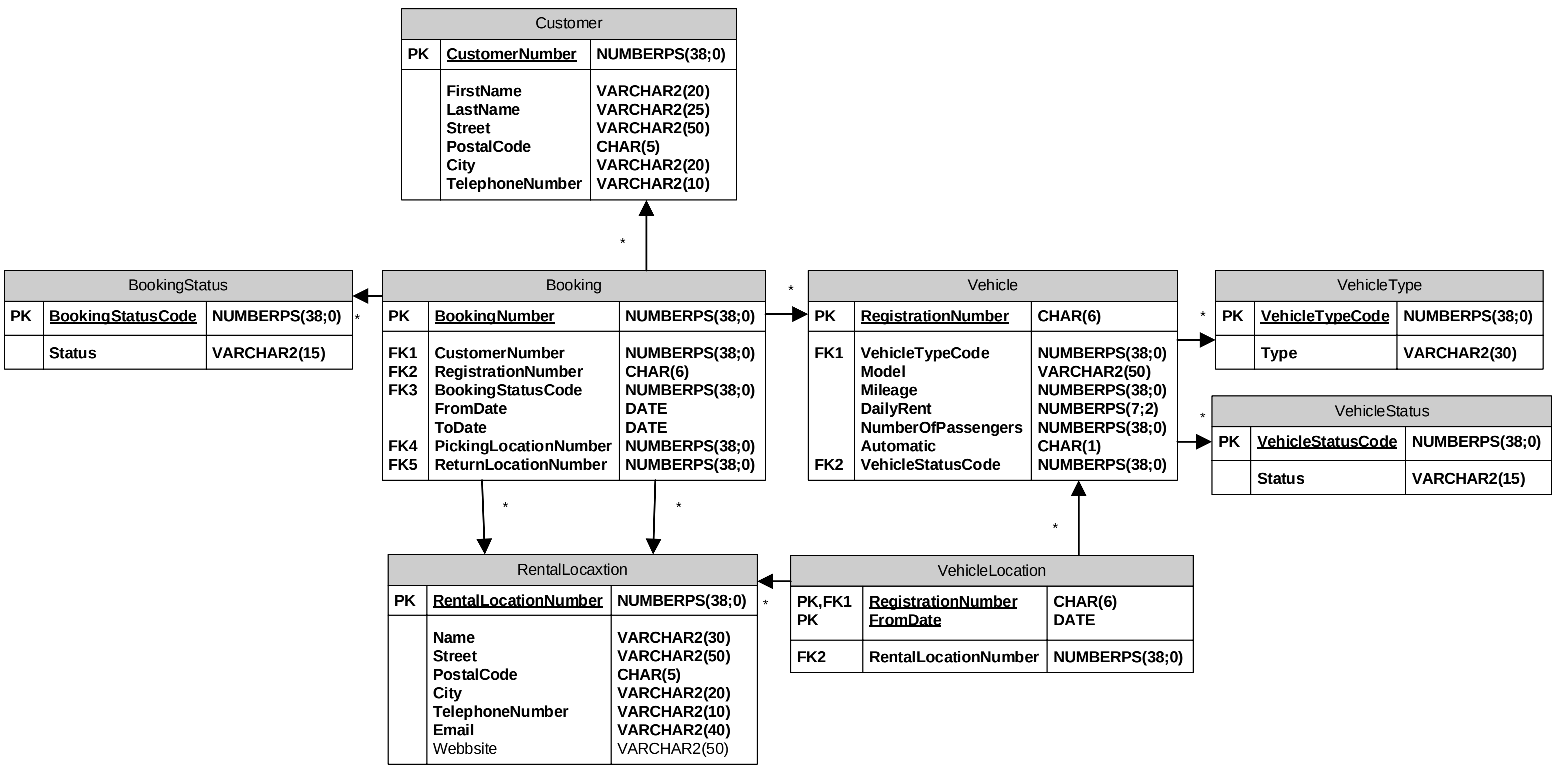
Submission and examination of the assignment takes place on three occasions according to the schedule (NOTE: A link for a certain submission occasion will only be visible in Canvas the day after the previous submission occasion). All three opportunities may be used. If you fail an assignment after the first and second examination, this must be remedied before the next examination.

First examination date is on the 29th of September 2022. The reamining two examination dates coincide with the re-exam dates as published on the exam schedule page.

5.3 Grade

Only the grades Fail or Pass are used to grade the assignment.

Appendix A (Database schema)



Customer

PK	CustomerNumber	NUMBERPS(38;0)
	FirstName	VARCHAR2(20)
	LastName	VARCHAR2(25)
	Street	VARCHAR2(50)
	PostalCode	CHAR(5)
	City	VARCHAR2(20)
	TelephoneNumber	VARCHAR2(10)

BookingStatus

PK	BookingStatusCode	NUMBERPS(38;0)
	Status	VARCHAR2(15)

Booking

PK	BookingNumber	NUMBERPS(38;0)
FK1	CustomerNumber	NUMBERPS(38;0)
FK2	RegistrationNumber	CHAR(6)
FK3	BookingStatusCode	NUMBERPS(38;0)
	FromDate	DATE
	ToDate	DATE
FK4	PickingLocationNumber	NUMBERPS(38;0)
FK5	ReturnLocationNumber	NUMBERPS(38;0)

Vehicle

PK	RegistrationNumber	CHAR(6)
FK1	VehicleTypeCode	NUMBERPS(38;0)
	Model	VARCHAR2(50)
	Mileage	NUMBERPS(38;0)
	DailyRent	NUMBERPS(7;2)
	NumberOfPassengers	NUMBERPS(38;0)
	Automatic	CHAR(1)
FK2	VehicleStatusCode	NUMBERPS(38;0)

VehicleType

PK	VehicleTypeCode	NUMBERPS(38;0)
	Type	VARCHAR2(30)

VehicleStatus

PK	VehicleStatusCode	NUMBERPS(38;0)
	Status	VARCHAR2(15)

RentalLocaxtion

PK	RentalLocationNumber	NUMBERPS(38;0)
	Name	VARCHAR2(30)
	Street	VARCHAR2(50)
	PostalCode	CHAR(5)
	City	VARCHAR2(20)
	TelephoneNumber	VARCHAR2(10)
	Email	VARCHAR2(40)
	Webbsite	VARCHAR2(50)

VehicleLocation

PK,FK1	RegistrationNumber	CHAR(6)
PK	FromDate	DATE
FK2	RentalLocationNumber	NUMBERPS(38;0)