Car Dealership

Project for Database Systems

This project was created as term project for course: Database Systems

Assignment is to create database model and database schema for topic of their own choosing. Project should have these parts:

* Topic – short description of what described database is about.
* Conceptual and logical (relational) diagrams
* Commands for creating schema in PostgreSQL database and filling it with sample data (CREATE TABLE, INSERT INTO)
* Sample queries in RA and SQL (24 in SQL, 10 in RA – commands can be same for RA and SQL). Try to get as many different categories of queries as possible list at the end. There is no set number of categories you need to do, but if you only do simple queries or queries of one type, it will influence your project final grade.

Yellow parts should be changed – white parts don’t.

**For completed project – this document must be sent to email of your LAB teacher. Your tables and sample data must also be created and filled into your assigned PostgreSQL database. Your database should be able to run all your queries (**[**https://psqlc.db.kii.pef.czu.cz/SQLCLient**](https://psqlc.db.kii.pef.czu.cz/SQLCLient)**).**

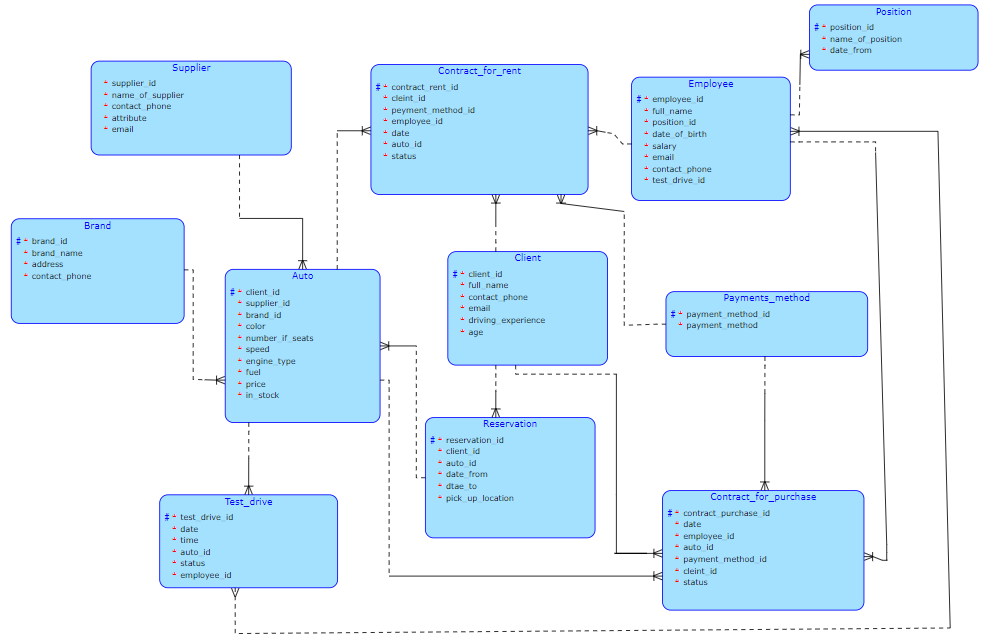
Topic

This car dealership focuses on selling and renting out new cars that belong to a company. Company has employees that are registered in various positions.(employee\_id, full name, position, date of birth, salary, contact phone, email).

Company order’s cars of different brands, colors, number of seats, engine types, fuel and prices from suppliers. Each auto registered in a local company system with full description and supplier(supplier\_id, name of supplier, contact phone, email). Cars also have a warranty from the supplier, and the period of that depends on the price of exact auto, but in any case of doubts about the quality, the client or company can appeal to the supplier. If desired, clients may reserve available cars for rent with exact dates of rental. It records in a system as a “reservation” with reservation\_id, client id, auto id, date from and to, pick-up location. Reservation for purchasing a car is also possible, but it works as an early order form the supplier, but looks the same in system. When clients come to the office for choose an auto, they are free to take a test drive that also records(test drive id, date,time, auto\_id, status). Employee of the company formulates a contract of buying or rental an auto between company and customer. Companies register clients in their local database(client\_id, full name, contact phone, email). Before buying a car client should get acquainted with the contract to learn all descriptions of the car. Afterwards the client is given the choice of payment method.It is possible for client to rent and buy cars simultaneously, for each car separate contract.

Restrictions for renting car: at least 3 years of driving experience, 25+ age, deposit 1000 euros.

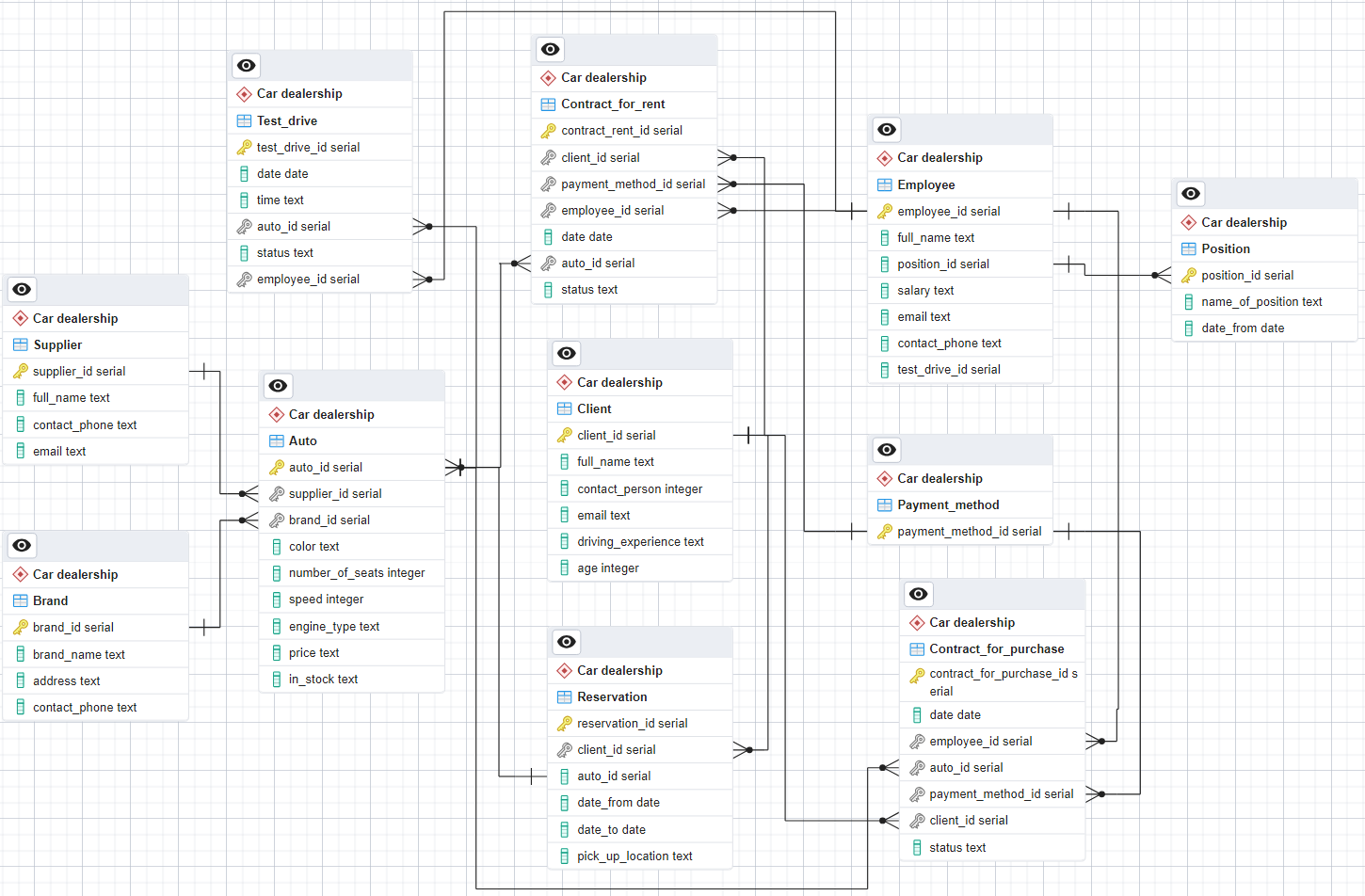
Conceptual diagram



Loops discussion

Discuss the loops dangerous and how it is solved

Logical diagram



Tables (schema)

Normal text in this chapter should be in sql comments – so it could be all copied, pasted and run in sql query windows. For commenting one row use -- at start of row. For commenting paragraph use /\* text \*/

Tables schema list:

CREATE TABLE IF NOT EXISTS "Car dealership"."Client"

(

client\_id serial NOT NULL,

full\_name text,

contact\_person integer,

email text,

driving\_experience text,

age integer,

PRIMARY KEY (client\_id)

);

CREATE TABLE IF NOT EXISTS "Car dealership"."Reservation"

(

reservation\_id serial,

client\_id serial,

auto\_id serial,

date\_from date,

date\_to date,

pick\_up\_location text,

PRIMARY KEY (reservation\_id)

);

CREATE TABLE IF NOT EXISTS "Car dealership"."Contract\_for\_rent"

(

contract\_rent\_id serial,

client\_id serial,

payment\_method\_id serial,

employee\_id serial,

date date,

auto\_id serial,

status text,

PRIMARY KEY (contract\_rent\_id)

);

CREATE TABLE IF NOT EXISTS "Car dealership"."Employee"

(

employee\_id integer NOT NULL,

full\_name text,

position\_id serial,

salary text,

email text,

contact\_phone text,

test\_drive\_id serial,

PRIMARY KEY (employee\_id),

UNIQUE (employee\_id)

);

CREATE TABLE IF NOT EXISTS "Car dealership"."Position"

(

position\_id serial NOT NULL,

name\_of\_position text,

date\_from date,

PRIMARY KEY (position\_id)

);

CREATE TABLE IF NOT EXISTS "Car dealership"."Payment\_method"

(

payment\_method\_id serial,

PRIMARY KEY (payment\_method\_id)

);

CREATE TABLE IF NOT EXISTS "Car dealership"."Contract\_for\_purchase"

(

contract\_for\_purchase\_id serial,

date date,

employee\_id serial,

auto\_id serial,

payment\_method\_id serial,

client\_id serial,

status text,

PRIMARY KEY (contract\_for\_purchase\_id)

);

CREATE TABLE IF NOT EXISTS "Car dealership"."Auto"

(

auto\_id serial NOT NULL,

supplier\_id serial,

brand\_id serial,

color text,

" number\_of\_seats" integer,

speed integer,

engine\_type text,

price text,

in\_stock text,

PRIMARY KEY (auto\_id)

);

CREATE TABLE IF NOT EXISTS "Car dealership"."Supplier"

(

supplier\_id serial NOT NULL,

full\_name text,

contact\_phone text,

email text,

PRIMARY KEY (supplier\_id)

);

CREATE TABLE IF NOT EXISTS "Car dealership"."Brand"

(

brand\_id serial NOT NULL,

brand\_name text,

address text,

contact\_phone text,

PRIMARY KEY (brand\_id)

);

CREATE TABLE IF NOT EXISTS "Car dealership"."Test\_drive"

(

test\_drive\_id serial,

date date,

"time" text,

auto\_id serial,

status text,

employee\_id serial,

PRIMARY KEY (test\_drive\_id)

);

ALTER TABLE IF EXISTS "Car dealership"."Reservation"

ADD FOREIGN KEY (client\_id)

REFERENCES "Car dealership"."Client" (client\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Contract\_for\_rent"

ADD FOREIGN KEY (client\_id)

REFERENCES "Car dealership"."Client" (client\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Contract\_for\_rent"

ADD FOREIGN KEY (employee\_id)

REFERENCES "Car dealership"."Employee" (employee\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Contract\_for\_rent"

ADD FOREIGN KEY (auto\_id)

REFERENCES "Car dealership"."Auto" (auto\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Contract\_for\_rent"

ADD FOREIGN KEY (payment\_method\_id)

REFERENCES "Car dealership"."Payment\_method" (payment\_method\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Employee"

ADD FOREIGN KEY (employee\_id)

REFERENCES "Car dealership"."Position" (position\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Contract\_for\_purchase"

ADD FOREIGN KEY (payment\_method\_id)

REFERENCES "Car dealership"."Payment\_method" (payment\_method\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Contract\_for\_purchase"

ADD FOREIGN KEY (employee\_id)

REFERENCES "Car dealership"."Employee" (employee\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Contract\_for\_purchase"

ADD FOREIGN KEY (client\_id)

REFERENCES "Car dealership"."Client" (client\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Contract\_for\_purchase"

ADD FOREIGN KEY (auto\_id)

REFERENCES "Car dealership"."Auto" (auto\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Auto"

ADD FOREIGN KEY (supplier\_id)

REFERENCES "Car dealership"."Supplier" (supplier\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Auto"

ADD FOREIGN KEY (brand\_id)

REFERENCES "Car dealership"."Brand" (brand\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Auto"

ADD FOREIGN KEY (auto\_id)

REFERENCES "Car dealership"."Reservation" (reservation\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Test\_drive"

ADD FOREIGN KEY (auto\_id)

REFERENCES "Car dealership"."Auto" (auto\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS "Car dealership"."Test\_drive"

ADD FOREIGN KEY (employee\_id)

REFERENCES "Car dealership"."Employee" (employee\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

Queries

There should be at least 24 queries in SQL total. Each query should have:

* Query Number.
* Set category from SQL Statements Cover chapter (last chapter).
* Short description.
* For 10 queries there should be variation in RA.
* Query itself – if there are multiple possibilities of how to write query, write it multiple times – write description in comment.

Normal text in this chapter should be in sql comments – so it could be all copied, pasted and run in sql query windows. For commenting one row use -- at start of row. For commenting paragraph use /\* text \*/

Query list:

/\* Query 1 - SQL Category: A, F2 \*/

-- List all employees and the autos they are associated with via test drives

RA: ("Car dealership"."Employee" \* "Car dealership"."Test\_drive" \* "Car dealership"."Auto")[full\_name, color, price]

SELECT E.full\_name, A.color, A.price

FROM "Car dealership"."Employee" E

JOIN "Car dealership"."Test\_drive" TD ON E.employee\_id = TD.employee\_id

JOIN "Car dealership"."Auto" A ON TD.auto\_id = A.auto\_id;

/\* Query 2 - SQL Category: B, H1 \*/

-- Union of all clients and all employees (assumes you want to see a list of people involved in the dealership)

RA: TempTable = "Car dealership"."Client" + "Car dealership"."Employee"

SELECT full\_name FROM "Car dealership"."Client"

UNION

SELECT full\_name FROM "Car dealership"."Employee";

/\* Query 3 - SQL Category: G1 \*/

-- Find all autos that have never been rented

RA:"Car dealership"."Auto" - ("Car dealership"."Auto" \* "Car dealership"."Contract\_for\_rent")

SELECT A.\*

FROM "Car dealership"."Auto" A

WHERE NOT EXISTS (SELECT 1 FROM "Car dealership"."Contract\_for\_rent" CR WHERE CR.auto\_id = A.auto\_id);

/\* Query 4 - SQL Category: I1 \*/

-- Count the number of contracts for each status

SELECT status, COUNT(\*)

FROM "Car dealership"."Contract\_for\_purchase"

GROUP BY status;

/\* Query 5 - SQL Category: I2 \*/

-- Show the total sales amount for each brand

SELECT B.brand\_name, SUM(CFP.contract\_for\_purchase\_id) AS TotalSales

FROM "Car dealership"."Brand" B

JOIN "Car dealership"."Auto" A ON B.brand\_id = A.brand\_id

JOIN "Car dealership"."Contract\_for\_purchase" CFP ON A.auto\_id = CFP.auto\_id

GROUP BY B.brand\_name;

/\* Query 6 - SQL Category: F4 \*/

-- List all employees and any associated contracts, including those who haven't made any contracts

RA: ("Car dealership"."Employee" \* "Car dealership"."Contract\_for\_purchase")[full\_name, \*]

SELECT E.full\_name, CFP.\*

FROM "Car dealership"."Employee" E

LEFT JOIN "Car dealership"."Contract\_for\_purchase" CFP ON E.employee\_id = CFP.employee\_id;

/\* Query 7 - SQL Category: F5 \*/

-- Show all autos and any associated rental contracts

RA: ("Car dealership"."Auto" \* "Car dealership"."Contract\_for\_rent")[\*]

SELECT A.\*, CFR.\*

FROM "Car dealership"."Auto" A

FULL JOIN "Car dealership"."Contract\_for\_rent" CFR ON A.auto\_id = CFR.auto\_id;

/\* Query 8 - SQL Category: G2 \*/

-- List the total number of rentals for each auto

SELECT A.auto\_id, COALESCE(RentalCounts.TotalRentals, 0) AS TotalRentals

FROM "Car dealership"."Auto" A

LEFT JOIN (SELECT auto\_id, COUNT(\*) AS TotalRentals FROM "Car dealership"."Contract\_for\_rent" GROUP BY auto\_id) RentalCounts ON A.auto\_id = RentalCounts.auto\_id;

/\* Query 9 - SQL Category: H1 \*/

-- Find all autos that have a listed price greater than the average price of all autos

SELECT \*

FROM "Car dealership"."Auto" A

WHERE CAST(REPLACE(A.price, ' czk', '') AS numeric) > (

SELECT AVG(CAST(REPLACE(price, ' czk', '') AS numeric))

FROM "Car dealership"."Auto"

);

/\* Query 10 - SQL Category: H3 \*/

-- Intersection of clients who have made both purchases and rentals

SELECT DISTINCT CFP.client\_id

FROM "Car dealership"."Contract\_for\_purchase" CFP

INTERSECT

SELECT CFR.client\_id

FROM "Car dealership"."Contract\_for\_rent" CFR;

/\* Query 11 - SQL Category: J \*/

-- Count the number of autos per supplier in three different ways.

-- Method 1

SELECT S.full\_name, COUNT(A.auto\_id)

FROM "Car dealership"."Supplier" S

JOIN "Car dealership"."Auto" A ON S.supplier\_id = A.supplier\_id

GROUP BY S.full\_name;

-- Method 2

SELECT full\_name, COUNT(auto\_id)

FROM "Car dealership"."Supplier"

JOIN "Car dealership"."Auto" USING (supplier\_id)

GROUP BY full\_name;

-- Method 3

SELECT S.full\_name, (SELECT COUNT(\*)

/\* Query 12 - SQL Category: K \*/

-- List autos with details and total sales, ordered by brand.

SELECT A.auto\_id, B.brand\_name, SUM(CFP.contract\_for\_purchase\_id) AS TotalSales

FROM "Car dealership"."Auto" A

JOIN "Car dealership"."Brand" B ON A.brand\_id = B.brand\_id

LEFT JOIN "Car dealership"."Contract\_for\_purchase" CFP ON A.auto\_id = CFP.auto\_id

GROUP BY A.auto\_id, B.brand\_name

ORDER BY B.brand\_name;

/\* Query 13 - SQL Category: G3 \*/

-- List employees and the number of contracts they have made.

RA: ("Car dealership"."Employee" \* "Car dealership"."Contract\_for\_purchase")[full\_name, COUNT(\*)→NumberOfContracts]

SELECT E.full\_name,

(SELECT COUNT(\*) FROM "Car dealership"."Contract\_for\_purchase" CFP WHERE CFP.employee\_id = E.employee\_id) AS NumberOfContracts

FROM "Car dealership"."Employee" E;

/\* Query 14 - SQL Category: I1 \*/

-- SELECT A.auto\_id, COUNT(R.reservation\_id) AS total\_reservations

FROM "Car dealership"."Auto" A

LEFT JOIN "Car dealership"."Reservation" R ON A.auto\_id = R.auto\_id

GROUP BY A.auto\_id;

/\* Query 15 - SQL Category: F3 \*/

-- Cross join between brands and Supplier

RA: ("Car dealership"."Brand" \* "Car dealership"."Supplier")[brand\_name, full\_name]

SELECT B.brand\_name, S.full\_name

FROM "Car dealership"."Brand" B

CROSS JOIN "Car dealership"."Supplier" S;

/\* Query 16 - SQL Category: H2 \*/

-- List employees who have formulated contracts but never conducted test drives.

SELECT E.full\_name

FROM "Car dealership"."Employee" E

WHERE EXISTS ( SELECT 1

FROM "Car dealership"."Contract\_for\_purchase" CP

WHERE CP.employee\_id = E.employee\_id)

EXCEPT SELECT E.full\_name

FROM "Car dealership"."Employee" E

JOIN "Car dealership"."Test\_drive" TD ON E.employee\_id = TD.employee\_id;

/\* Query 17- SQL Category: G2 \*/

-- List all employees along with the number of autos they have associated with in test drives.

SELECT E.full\_name, COUNT(DISTINCT TD.auto\_id) AS Number\_of\_Autos

FROM "Car dealership"."Employee" E

LEFT JOIN "Car dealership"."Test\_drive" TD ON E.employee\_id = TD.employee\_id

GROUP BY E.full\_name;

/\* Query 18 - SQL Category: K,M \*/

-- List autos with details, including the total number of test drives and reservations, ordered by brand.

SELECT B.brand\_name, A.auto\_id, A.color, A.price,

(SELECT COUNT(\*) FROM "Car dealership"."Test\_drive" TD WHERE TD.auto\_id = A.auto\_id) AS TestDriveCount,

(SELECT COUNT(\*) FROM "Car dealership"."Reservation" R WHERE R.auto\_id = A.auto\_id) AS ReservationCount

FROM "Car dealership"."Auto" A

JOIN "Car dealership"."Brand" B ON A.brand\_id = B.brand\_id

ORDER BY B.brand\_name;

/\* Query 19 - SQL Category: I2 \*/

-- List the number of autos supplied by each supplier, along with the supplier's name.

SELECT S.full\_name, COUNT(A.auto\_id) AS NumberOfAutos

FROM "Car dealership"."Supplier" S

JOIN "Car dealership"."Auto" A ON S.supplier\_id = A.supplier\_id

GROUP BY S.full\_name;

/\* Query 20 - SQL Category: D2 \*/

-- Check for any autos that have not been rented by every client who has made a reservation.

RA: SubqueryResult = "Car dealership"."Reservation"[client\_id] \* ("Car dealership"."Auto" \* "Car dealership"."Contract\_for\_rent")[auto\_id]

"Car dealership"."Auto"(auto\_id = SubqueryResult.auto\_id)[auto\_id]

SELECT A.auto\_id

FROM "Car dealership"."Auto" A

WHERE EXISTS (

SELECT R.client\_id

FROM "Car dealership"."Reservation" R

EXCEPT

SELECT CFR.client\_id

FROM "Car dealership"."Contract\_for\_rent" CFR

WHERE CFR.auto\_id = A.auto\_id

);

/\* Query 21 - SQL Category: F2 \*/

RA: ("Car dealership"."Contract\_for\_purchase" \* "Car dealership"."Auto")[\*]

-- Display all contracts for purchase with corresponding auto details using a NATURAL JOIN.

SELECT CFP.\*, A.brand\_id, A.color

FROM "Car dealership"."Contract\_for\_purchase" CFP

NATURAL JOIN "Car dealership"."Auto" A;

/\* Query 22 - SQL Category: F1 \*/

-- List all autos and their respective supplier details.

RA: ("Car dealership"."Auto" \* "Car dealership"."Supplier")[auto\_id, color, full\_name AS SupplierName]

SELECT A.auto\_id, A.color, S.full\_name AS SupplierName

FROM "Car dealership"."Auto" A

JOIN "Car dealership"."Supplier" S ON A.supplier\_id = S.supplier\_id;

/\* Query 23 - SQL Category: P \*/

-- Delete all reservations for clients who have never made a purchase.

DELETE FROM "Car dealership"."Reservation"

WHERE client\_id IN (

SELECT client\_id

FROM "Car dealership"."Client"

WHERE client\_id NOT IN (

SELECT client\_id

FROM "Car dealership"."Contract\_for\_purchase"

)

);

/\* Query 24 - SQL Category: O \*/

-- Update the status of all contracts for purchase to 'Completed' for clients who have more than 3 years of driving experience.

UPDATE "Car dealership"."Contract\_for\_purchase"

SET status = 'Completed'

WHERE client\_id IN (

SELECT client\_id

FROM "Car dealership"."Client"

WHERE driving\_experience > '3 years'

);

**RA Queries**

-- RA Query 1: Select All Attributes from the Client Entity

Client[\*]

-- RA Query 2: Select Full Name and Age from Clients

Client[full\_name, age]

-- RA Query 3: Select Full Names of All Clients

Client[full\_name]

-- RA Query 4: Select Autos with Speed Greater Than 200

"Car dealership"."Auto"(speed > 200)[\*]

-- RA Query 5: Suppliers with Supplier ID Less Than 3

"Car dealership"."Supplier"(supplier\_id < 3)[\*]

-- RA Query 6: Details of Contract for Rent Made after 2023-08-01

"Car dealership"."Contract\_for\_rent"(date >= '2023-08-01')[\*]

-- RA Query 7: Select Employees Who Have Conducted Test Drives

"Car dealership"."Employee"(test\_drive\_id IS NOT NULL)[\*]

-- RA Query 8: Select Brand Names

Brand[brand\_name]

-- RA Query 9: Select All Reservation Details

Reservation[\*]

-- RA Query 10: Selects all attributes from Autos that are marked as 'in stock'

"Car dealership"."Auto"(in\_stock = 'in stock')[\*]

Sample data

Normal text in this chapter should be in sql comments – so it could be all copied, pasted and run in sql query windows. For commenting one row use -- at start of row. For commenting paragraph use /\* text \*/

Sample data list:

INSERT INTO Client (client\_id, full\_name, contact\_phone, email, driving\_experience, age) VALUES ('1', 'Michael Potter', '770662122', 'Mick1@gmail.com', '2 years', '36');

INSERT INTO Client (client\_id, full\_name, contact\_phone, email, driving\_experience, age) VALUES ('2', 'Alisa Kim', '445566543', 'Alisa2@gmail.com', '4 years', '30');

INSERT INTO Client (client\_id, full\_name, contact\_phone, email, driving\_experience, age) VALUES ('3', 'Alina Bond', '123456789', 'Alina3@gmail.com', '3 years', '26');

INSERT INTO Client (client\_id, full\_name, contact\_phone, email, driving\_experience, age) VALUES ('4', 'Karel Jackson', '709876543', 'Karel4@gmail.com', '10 years', '56');

INSERT INTO Client (client\_id, full\_name, contact\_phone, email, driving\_experience, age) VALUES ('5', 'Rima Switch', '445667889', 'MRmima31@gmail.com', '5 years', '46');

INSERT INTO Client (client\_id, full\_name, contact\_phone, email, driving\_experience, age) VALUES ('6', 'Olesya Fritch', '445655449', 'daa1@gmail.com', '5 years', '24');

INSERT INTO "Car dealership"."Supplier" (supplier\_id, full\_name, contact\_phone, email) VALUES

(1, 'Lora Johnson', '778443554', 'Lora23@gmail.com'),

(2, 'Avrora Hahnson', '333443554', 'Avrora23@gmail.com');

INSERT INTO "Car dealership"."Brand" (brand\_id, brand\_name, address, contact\_phone) VALUES

(1, 'Mersedes', 'Na Belidle 23', '432432321'),

(2, 'Tesla', 'Palladium', '543543567'),

(3, 'BMW', 'Na Prikope 121', '444333222');

INSERT INTO "Car dealership"."Payment\_method" (payment\_method\_id) VALUES

(1),

(2),

(3);

INSERT INTO "Car dealership"."Position" (position\_id, name\_of\_position, date\_from) VALUES

(2, 'Sales Manager', '2020-01-01'),

(1, 'Product Manager', '2021-01-01'),

(3, 'Consultant', '2022-01-01');

INSERT INTO "Car dealership"."Employee"

(employee\_id, full\_name, position\_id, salary, email, contact\_phone, test\_drive\_id) VALUES

(3, 'Mark Robins', 2, '50 000 czk', 'Mark@gmail.com', '444000333', 1),

(1, 'Harry Kort', 1, '60 000 czk', 'Harry@gmail.com', '888000333', 2),

(2, 'Leila Mis', 3, '40 000 czk', 'Leila@gmail.com', '499900333', 3);

INSERT INTO "Car dealership"."Auto" (auto\_id, supplier\_id, brand\_id, color, " number\_of\_seats", speed, engine\_type, price, in\_stock) VALUES

(1, 1, 1, 'white', 5, 240, 'two-stroke', '100000 czk', 'in stock'),

(2, 2, 3, 'blue', 5, 240, 'one-stroke', '200000 czk', 'in stock'),

(3, 2, 1, 'pink', 5, 240, 'three-stroke', '150000 czk', 'in stock'),

(4, 1, 2, 'black', 5, 240, 'two-stroke', '300000 czk', 'in stock'),

(5, 1, 2, 'white', 5, 240, 'two-stroke', '100000 czk', 'in stock'),

(6, 2, 1, 'black', 5, 240, 'one-stroke', '150000 czk', 'in stock');

INSERT INTO "Car dealership"."Contract\_for\_purchase" (contract\_for\_purchase\_id, date, employee\_id, auto\_id, payment\_method\_id, client\_id, status) VALUES

(1, '2023-06-01', 3, 1, 1, 1, 'completed'),

(2, '2023-07-15', 1, 2, 2, 2, 'pending'),

(3, '2023-08-05', 2, 3, 3, 3, 'completed'),

(4, '2023-09-10', 3, 4, 1, 4, 'pending'),

(5, '2023-10-22', 1, 5, 2, 5, 'completed'),

(6, '2023-11-29', 2, 6, 3, 6, 'completed');

INSERT INTO "Car dealership"."Reservation"(reservation\_id, client\_id, auto\_id, date\_from, date\_to, pick\_up\_location) VALUES

(1, 2, 3, '2023.07.07', '2023.12.29', 'Palladium central entrance'),

(2, 3, 5, '2023.08.07', '2023.12.28', 'Novy Smichov central entrance'),

(3, 4, 1, '2023.01.07', '2023.12.20', 'Chodov central entrance'),

(4, 5, 2, '2023.04.07', '2023.12.21', 'Narodni Museum central entrance'),

(5, 1, 4, '2023.05.07', '2023.12.22', 'Palladium central entrance'),

(6, 6, 3, '2023.10.07', '2023.12.22', 'Chodoc central entrance');

INSERT INTO "Car dealership"."Contract\_for\_rent" (contract\_rent\_id, client\_id, payment\_method\_id, employee\_id, date, auto\_id, status) VALUES

(1, 2, 3, 3, '2023.07.07', 3, 'active'),

(2, 3, 2, 2, '2023.08.07', 5, 'active'),

(3, 4, 3, 1, '2023.01.07', 1, 'active'),

(4, 5, 1, 2, '2023.04.07', 2, 'active'),

(5, 1, 2, 1, '2023.05.07', 4, 'active');

INSERT INTO "Car dealership"."Test\_drive" (test\_drive\_id, date, time, auto\_id, status, employee\_id) VALUES

(1,'2023.07.06', '17:50', 3, 'succesfully completed', 3),

(2, '2023.08.06', '16:00', 5, 'succesfully completed', 2);

SQL Statements Cover

|  |  |
| --- | --- |
| Category | Category statement description |
| A | Simple (positive) query with at least two joined tables (list of patients - name, address who were examined by doctor Braun) |
| B | Simple (negative) query (list of patients - name, address who were not visiting doctor Braun) (select all doctors who had patients,...) |
| C | Choose all records, which have relation for "X" only …(list of patients - name, address who were examined by doctor Braun ONLY - the patients didn't have visit at another doctor). |
| D1 | Select all records which are at the relation with all.General quantified query (list of doctors - name, address who were visited by EACH patient, who visited doctor Braun) |
| D2 | The result validity check from category D1. For example if query {list of teachers, who had lecture during ALL semesters begin at winter 2001/2002 till summer semester 2008/2008 included} shows the teacher "123 Josef Pavlicek", so the validity check will be query {{list of all semester at the season, where was teaching Pavlicek }\ (minus){list of all semester at the season}} and the result must be empty. |
| F1 | join - JOIN ON |
| F2 | NATURAL JOIN | JOIN USING |
| F3 | CROSS JOIN |
| F4 | LEFT | RIGHT OUTER JOIN |
| F5 | FULL (OUTER) JOIN |
| G1 | Subquery inside WHERE |
| G2 | Subquery inside FROM |
| G3 | Subquery inside SELECT |
| G4 | Correlated subquery (EXISTS | NOT EXISTS) |
| H1 | Set union query - UNION |
| H2 | Set substraction query - EXCEPT or MINUS (v Oracle) |
| H3 | Set intersection - INTERSECT |
| I1 | Aggregation functions (count | sum | min | max| avg) |
| I2 | Aggregations with GROUP BY (HAVING) clause |
| J | The same query using three different formulations in SQL |
| K | All select clauses - SELECT FROM WHERE GROUP BY HAVING ORDER BY |
| L | Make VIEW |
| M | View used query |
| N | Insert statement for inserting list of records - INSERT without clause VALUES use, for example ( add customer ID 6 all green boats reservation for all needed time intervals) |
| O | UPDATE statement with subquery in WHERE/SET |
| P | DELETE statement with subquery in WHERE clause |