BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Database project assignment 2

Lukáš Král

Brno 2021

Create a query that will retrieve only selected columns from the selected table

SELECT employee_id, first_name FROM employee;

Data Output | Explain | Messages | Notifications

	employee_id bigint	first_name character varying (45)
1	2	Petr
2	3	Hana
3	4	Alois
4	5	Luboš
5	6	Markéta
6	7	Jakub
7	8	Zbyněk
8	9	Vojtěch
9	10	Štěpán
10	11	Šimon
11	12	Rostislav
12	13	Richard

Create a query that will select user/person or similar table based on the email.

SELECT first_name, surname FROM employee WHERE mail = 'petr.kal@automotive.cz';

or

SELECT first_name, surname FROM employee WHERE mail LIKE 'petr.kal@automotive.cz' ;

Data Output | Explain | Messages | Notifications

	first_name character varying (45)	surname character varying (45)
1	Petr	Kalivoda

Create at least one UPDATE, INSERT, DELETE, and ALTER TABLE query

UPDATE employee SET first_name = 'Hana' WHERE employee_id = 1

INSERT INTO car(car_id, car_brand, car_color, customer_id) VALUES (6, 'Porsche', 'červená',2);

DELETE FROM employee_has_contract WHERE employee_id = 48;

ALTER TABLE job ALTER COLUMN salary TYPE INT;

Create a series of queries that will separately use the following:

WHERE

SELECT * FROM employee WHERE first_name = 'Ludvík'

Data Output | Explain | Messages | Notifications

	employee_id [PK] bigint	first_name character varying (45)	surname character varying (45)	mail character varying (45)	building_id [PK] bigint
1	44	Ludvík	Daniš	ludva.danis@automotive....	6
2	48	Ludvík	Gajdoš	ludva.gajdos@automotiv...	5

LIKE

SELECT * FROM car WHERE car_brand LIKE 'B%';

Data Output | Explain | Messages | Notifications

	car_id [PK] bigint	car_brand character varying (45)	car_color character varying (45)	customer_id bigint
1	3	BMW	černá	[null]
2	4	BMW	bílá	[null]

NOT LIKE

SELECT * FROM car WHERE car_brand NOT LIKE 'Vol%' AND car_brand NOT LIKE 'BM%';

Data Output | Explain | Messages | Notifications

	car_id [PK] bigint	car_brand character varying (45)	car_color character varying (45)	customer_id bigint
1	5	Škoda	stříbrná	[null]
2	6	Porsche	červená	2

SUBSTRING

SELECT employee_id,substring(first_name,1,4)"First 4 letters" FROM employee WHERE employee_id BETWEEN 10 AND 30;

Data Output | Explain | Messages | Notific

	employee_id bigint	First 4 letters text
1	10	Štěp
2	11	Šimo
3	12	Rost
4	13	Rich
5	14	Adam
6	15	Bohu
7	16	Jan
8	17	Mart

TRIM

SELECT TRIM('Hana' FROM first_name) FROM employee;

	btrim text
1	Petr
2	
3	Alois
4	Luboš

COUNT

SELECT COUNT(car_color) FROM car ;

	count bigint
1	6

SUM

SELECT SUM(salary)"Salary costs" FROM manager;

	Salary costs double precision
1	395200

MIN

SELECT * FROM job where salary = (SELECT MIN(salary) FROM job);

	job_id [PK] bigint	job_type character varying (45)	salary integer
1	2	uklízečka	18000

MAX

SELECT * FROM manager where salary = (SELECT MAX(salary) FROM manager);

	manager_id [PK] bigint	manager_firstname character varying (45)	manager_surname character varying (45)	salary double precision	building_id [PK] bigint
1	3	Rudolf	Bednařík	135000	3

AVG

SELECT AVG(salary) from manager;

Data Output Explain

	avg double precision
1	79040

GROUP BY

SELECT employee_id FROM employee GROUP BY employee_id;

Data Output Explain

	employee_id bigint
1	42
2	29
3	4
4	34
5	51
6	41
7	46
8	40

GROUP BY and HAVING

SELECT manager_firstname, manager_surname, salary FROM manager
GROUP BY manager_firstname, manager_surname, salary, manager_id HAVING salary > 10000
ORDER BY salary;

	employee_id bigint
1	42
2	29
3	4
4	34
5	51
6	41

GROUP BY, HAVING, and WHERE

SELECT manager_firstname, manager_surname, salary FROM manager WHERE manager_firstname LIKE 'M%'
GROUP BY manager_firstname, manager_surname, salary, manager_id HAVING salary > 10000
ORDER BY salary;

	manager_firstname character varying (45)	manager_surname character varying (45)	salary double precision
1	Marcel	Šimčík	43200
2	Miroslav	Hladký	100000

UNION ALL

SELECT first_name FROM employee UNION ALL SELECT manager_firstname from manager;

	first_name character varying (45)
1	Petr
2	Hana
3	Alois
4	Luboš
5	Markéta
6	Jakub
7	Zbyněk
8	Vojtěch
9	Štěpán
10	Šimon
11	Rostislav
12	Richard

UNION

SELECT first_name FROM employee UNION SELECT manager_firstname from manager;

	first_name character varying (45)
1	Rudolf
2	Tomáš
3	František
4	Ludvík
5	Luděk
6	Emilie
7	Karlos

DISTINCT

SELECT DISTINCT car_brand FROM car;

1	Škoda
2	Porsche
3	BMW
4	Volkswagen

LEFT JOIN

SELECT employee_has_contract.employee_id, job.job_id FROM employee_has_contract LEFT JOIN job ON employee_has_contract.job_id=job.job_id;

Data Output | Explain | Messages | Notificatic

	employee_id bigint	job_id bigint
1	1	5
2	2	5
3	3	4
4	4	1
5	5	1
6	6	7
7	7	1
8	8	5
9	9	1
10	10	6

RIGHT JOIN

SELECT employee_has_contract.employee_id, job.job_id FROM employee_has_contract RIGHT JOIN job ON employee_has_contract.job_id=job.job_id;

	employee_id bigint	job_id bigint
1	1	5
2	2	5
3	3	4
4	4	1
5	5	1
6	6	7

FULL OUTER JOIN

SELECT * from employee FULL OUTER JOIN employee_has_contract ON employee.employee_id=employee_has_contract.employee_id;

Data Output Explain Messages Notifications

	employee_id bigint	first_name character varying (45)	surname character varying (45)	mail character varying (45)	building_id bigint	contract_expiration date	employee_id bigint	job_id bigint
1	1	Hana	Samková	alena.samkova@auto...	2	2023-01-01	1	5
2	2	Petr	Kalivoda	petr.kal@automotive.cz	3	2023-01-01	2	5
3	3	Hana	Jandová	alena.jand@automoti...	4	2023-01-01	3	4
4	4	Alois	Smola	alois.smola@automot...	1	2023-01-01	4	1
5	5	Luboš	Veverka	lubos.vever@automot...	1	2023-01-01	5	1
6	6	Markéta	Janková	mj.1@automotive.cz	5	2023-01-01	6	7
7	7	Jakub	Ferenc	jakub.ferenc@automo...	5	2023-01-01	7	1
8	8	Zbyněk	Hrnčíř	zbynek.hrncir@autom...	2	2023-01-01	8	5
9	9	Vojtěch	Sluka	vojtech.sluk@automo...	1	2023-01-01	9	1
10	10	Štěpán	Bečvár	stepan.becvar@auto...	4	2023-01-01	10	6
11	11	Šimon	Motl	simon.motl@automot...	1	[null]	11	1
12	12	Rostislav	Kropáček	rost.krop@automotiv...	1	2025-01-01	12	3
13	13	Richard	Maršík	richard.mars@autom...	1	[null]	13	1
14	14	Adam	Vybíral	nevybral@automotive...	1	[null]	14	1
15	15	Bohuslav	Berky	bohus.berky@automo...	1	[null]	15	1

Use in one query: LEFT JOIN, GROUP BY, HAVING, ORDER BY, AVG and DISTINCT

SELECT DISTINCT employee_has_address.address_type, AVG(job.salary)"prumer plat" FROM employee_has_address
LEFT JOIN job ON employee_has_address.employee_id=job.job_id GROUP BY employee_has_address.address_type
HAVING AVG(job.salary) > 5 ORDER BY employee_has_address.address_type ;

Data Output Explain Messages Notifications

	address_type character varying (45)	prumerplat numeric
1	Adresa trvalého bydliště	27600.000000000000
2	Korespondenční adresa	24400.000000000000

Create a query that will return the data from an arbitrary table for the last one and half days (1day + 12 hours, i.e., 36 hours). Do not hard code the query (e.g., created at > 7-11-2021)! Do it programmatically with DATE functions

SELECT * FROM employee_has_contract WHERE contract_expiration <= NOW() - '36 hours'::INTERVAL;

Data Output Explain Messages Notifications

contract_expiration date	employee_id [PK] bigint	job_id [PK] bigint

Create a query that will return data from the last month (starting from the first day of the month)

```
SELECT * FROM employee_has_contract
WHERE contract_expiration <= date_trunc('month', current_date - interval '1' month)
and contract_expiration < date_trunc('month', current_date);
```

Data Output | Explain | Messages | Notifications

car_id [PK] bigint	employee_id [PK] bigint	deadline date

Write a select that will remove accents on a selected string (e.g., ´a will be converted to a)

```
INSERT INTO employee (employee_id, first_name, surname, mail, building_id) VALUES
(51,'Ondřej', 'Křupatý', 'okrup@mail.cz', 1);
CREATE EXTENSION UNACCENT;
SELECT employee_id, UNACCENT(first_name), UNACCENT(surname) FROM employee WHERE
employee_id=51;
```

Data Output | Explain | Messages | Notifications

	employee_id bigint	unaccent text	unaccent text
1	51	Ondrej	Krupaty

Create a query for pagination in an application (use LIMIT and OFFSET)

```
SELECT * FROM employee ORDER BY employee_id LIMIT 6 OFFSET 3 ;
```

Data Output | Explain | Messages | Notifications

	employee_id [PK] bigint	first_name character varying (45)	surname character varying (45)	mail character varying (45)	building_id [PK] bigint
1	4	Alois	Smola	alois.smola@automotive.cz	1
2	5	Luboš	Veverka	lubos.vever@automotive.cz	1
3	6	Markéta	Janková	mj.1@automotive.cz	5
4	7	Jakub	Ferenc	jakub.ferenc@automotive.cz	5
5	8	Zbyněk	Hrnčíř	zbynek.hrncir@automotive.cz	2
6	9	Vojtěch	Sluka	vojtech.sluk@automotive.cz	1

Create a query that will use subquery in FROM

SELECT * FROM (SELECT manager_firstname, manager_surname FROM manager) AS Man;

Data Output | Explain | Messages | Notifications

	manager_firstname character varying (45)	manager_surname character varying (45)
1	Miroslav	Hladký
2	Natálie	Syrová
3	Rudolf	Bednařík
4	Alžběta	Stupková
5	Marcel	Šimčík

Create a query that will use subquery in WHERE condition

SELECT * FROM employee WHERE building_id = (SELECT building_id FROM building WHERE building_name = 'účetnictví');

Data Output | Explain | Messages | Notifications

	employee_id [PK] bigint	first_name character varying (45)	surname character varying (45)	mail character varying (45)	building_id [PK] bigint
1	8	Zbyněk	Hrnčíř	zbynek.hrncir@automotive.cz	2
2	22	Barbora	Ptáčková	bara.ptack@automotive.cz	2
3	37	Emilie	Kubátová	emilie.kubat@automotive.cz	2
4	1	Hana	Samková	alena.samkova@automotive.cz	2

Create a query that will use any aggregate function and GROUP BY with HAVING

SELECT building_id,count(employee)"Number of employees" FROM employee GROUP BY(building_id) HAVING count(employee)> 5;

Data Output | Explain | Messages | Notific

	building_id bigint	Number of employees bigint
1	1	20
2	3	8
3	5	10

Create a query that will join at least five tables

```
SELECT DISTINCT e.employee_id,e.first_name,e.surname,j.job_type, j.salary, a.city, a.street,
a.street_number
FROM employee e
JOIN employee_has_contract ehc ON e.employee_id = ehc.employee_id
JOIN job j ON ehc.job_id = j.job_id
JOIN employee_has_address eha ON eha.employee_id = e.employee_id
JOIN address a ON a.address_id = eha.address_id;
```

Data Output | Explain | Messages | Notifications

	employee_id bigint	first_name character varying (45)	surname character varying (45)	job_type character varying (45)	salary integer	city character varying (45)	street character varying (45)	street_number character varying (45)
1	3	Hana	Jandová	IT technik	35000	Rožmitál pod Tremšínem	U medvídků	2155
2	1	Hana	Samková	účetní	29500	Úhretice	Jiráskova	1045
3	3	Hana	Jandová	IT technik	35000	Louny	Louny	17123
4	2	Petr	Kalivoda	účetní	29500	Zajecov	K Lukárně	5188
5	1	Hana	Samková	účetní	29500	Nechanice	Na Kopečku	1544

Create a query that will join at least three tables and will use GROUP BY, HAVING, and WHERE

```
SELECT e.building_id, count(e.employee_id) FROM employee e
JOIN employee_has_address as eha ON e.employee_id = eha.employee_id
JOIN employee_has_contract as ehc ON eha.employee_id = ehc.job_id WHERE
ehc.contract_expiration < '2025-01-01' GROUP BY(e.building_id)
HAVING min(eha.address_type) = 'Adresa trvalého bydliště';
```

Data Output | Explain | Messages

	building_id bigint	count bigint
1	1	4
2	5	2
3	2	8

Modify the database from the first project assignment to improve integrity constraints (e.g., reduce the size for varchar columns) – Set cascading, explain places where you used cascading and why?

```
ALTER TABLE building ALTER COLUMN building_name TYPE varchar(30);
ALTER TABLE employee ALTER COLUMN first_name TYPE varchar(20);
ALTER TABLE employee ALTER COLUMN surname  TYPE varchar(20);
ALTER TABLE customer ALTER COLUMN customer_f_name TYPE varchar(20);
ALTER TABLE customer ALTER COLUMN customer_s_name  TYPE varchar(20);
ALTER TABLE car ALTER COLUMN car_color TYPE varchar(20);
ALTER TABLE car ALTER COLUMN car_brand TYPE varchar(25);
```

These columns didn’t need that much size in my opinion.

Create database indexes

WITHOUT INDEX:

EXPLAIN ANALYZE SELECT employee_id, first_name FROM employee WHERE first_name = 'Hana';

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on employee (cost=0.00..1.66 rows=2 width=15...
2	[...] Filter: ((first_name)::text = 'Hana'::text)
3	[...] Rows Removed by Filter: 51
4	Planning Time: 0.262 ms
5	Execution Time: 0.028 ms

CREATE INDEX n_idx ON employee(first_name);

WITH INDEX:

EXPLAIN ANALYZE SELECT employee_id, first_name FROM employee WHERE first_name = 'Hana';

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on employee (cost=0.00..1.66 rows=2 width=15...
2	[...] Filter: ((first_name)::text = 'Hana'::text)
3	[...] Rows Removed by Filter: 51
4	Planning Time: 0.097 ms
5	Execution Time: 0.031 ms

Searching with index is more like scanning names, it is faster than searching without index.

Create arbitrary database procedure (consider some complex case)

```
CREATE OR REPLACE PROCEDURE Insert_employee(bigint, character varying, character varying,
character varying,bigint)
LANGUAGE 'plpgsql'
AS $$
BEGIN
INSERT INTO public.employee(employee_id, first_name, surname, mail,building_id) values
($1,$2,$3,$4,$5);
COMMIT;

END;
$$;
```

Query Editor Query History

```
1  CREATE PROCEDURE Insert_employee(bigint, character varying, character varying, character varying,bigint)
2  LANGUAGE 'plpgsql'
3  AS $$
4  BEGIN
5  INSERT INTO public.employee(employee_id, first_name, surname, mail,building_id) values ($1,$2,$3,$4,$5);
6  COMMIT;
7
8  END;
9  $$;
```

CALL Insert_employee(52,'Karlos','Vémola','vymol@mail.cz',1);
SELECT * FROM employee WHERE employee_id > 50;

Data Output Explain Messages Notifications

	employee_id [PK] bigint	first_name character varying (45)	surname character varying (45)	mail character varying (45)	building_id [PK] bigint
1	51	Ondřej	Křupatý	okrup@mail.cz	1
2	52	Karlos	Vémola	vymol@mail.cz	1

Create arbitrary database trigger

```
CREATE FUNCTION new_employee() RETURNS trigger AS $tr$
BEGIN
RAISE NOTICE 'New employee in company!';
RETURN NEW;
END;
$tr$ LANGUAGE plpgsql ;
CREATE TRIGGER information AFTER INSERT ON employee
EXECUTE PROCEDURE new_employee();
INSERT INTO employee(employee_id, first_name, surname, mail,building_id) values (53, 'Lojza',
'Zlojza', 'lojzazlojza@mail.cz',4);
```

Data Output Explain Messages Notifications

```
NOTICE:  New employee in company!
INSERT 0 1

Query returned successfully in 45 msec.
```

Create arbitrary database view (consider some complex case)

```
CREATE VIEW employee_check AS
SELECT DISTINCT e.employee_id,e.first_name,e.surname,j.job_type, j.salary, a.city, a.street,
a.street_number
FROM employee e
JOIN employee_has_contract ehc ON e.employee_id = ehc.employee_id
JOIN job j ON ehc.job_id = j.job_id
JOIN employee_has_address eha ON eha.employee_id = e.employee_id
JOIN address a ON a.address_id = eha.address_id;
SELECT * FROM employee_check;
```

Data Output | Explain | Messages | Notifications

	employee_id bigint	first_name character varying (45)	surname character varying (45)	job_type character varying (45)	salary integer	city character varying (45)	street character varying (45)	street_number character varying (45)
1	3	Hana	Jandová	IT technik	35000	Rožmitál pod Tremšínem	U medvídků	2155
2	1	Hana	Samková	účetní	29500	Úhretice	Jiráskova	1045
3	6	Markéta	Janková	správce odpadů	27500	Louny	Louny	17123
4	3	Hana	Jandová	IT technik	35000	Louny	Louny	17123
5	7	Jakub	Ferenc	dělník	22000	Zajecov	K Lukárně	5188
6	5	Luboš	Veverka	dělník	22000	Nechanice	Na Kopečku	1544
7	2	Petr	Kalivoda	účetní	29500	Zajecov	K Lukárně	5188
8	1	Hana	Samková	účetní	29500	Nechanice	Na Kopečku	1544

Create database materialized view

```
CREATE MATERIALIZED VIEW car_view AS
SELECT DISTINCT e.car_id,e.car_brand,e.customer_id,chw.employee_id, cib.parts, cib.building_id
FROM car e
JOIN car_has_workers chw ON e.car_id = chw.car_id
JOIN car_in_building cib ON chw.car_id = cib.car_id;

SELECT * FROM car_view;
```

Data Output | Explain | Messages | Notifications

	car_id bigint	car_brand character varying (45)	customer_id bigint	employee_id bigint	parts character varying (45)	building_id bigint
1	1	Volkswagen	[null]	44	Motor	6
2	1	Volkswagen	[null]	12	Motor	6
3	1	Volkswagen	[null]	44	Volant	1
4	1	Volkswagen	[null]	12	Volant	1
5	1	Volkswagen	[null]	14	Motor	6
6	1	Volkswagen	[null]	14	Volant	1

Create two roles teacher and student in your database. Assign for teacher privileges to SELECT, INSERT, UPDATE, and DELETE everything in arbitrary table. Furthermore, set for teacher the possibility to view only certain fields (e.g., without salary from ” person“ or your ” user“ object). For student assign a possibility to select only certain tables.

```
CREATE ROLE teacher NOSUPERUSER;
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM teacher;
GRANT SELECT, INSERT, UPDATE, DELETE ON address TO teacher;
CREATE VIEW teachers_view AS
(SELECT employee_id, first_name, surname, mail FROM employee);
GRANT SELECT ON teachers_view TO teacher;
CREATE ROLE student NOSUPERUSER;
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM student;
GRANT SELECT ON employee, car TO student;
```