

Database project assignment 2

Lukáš Král

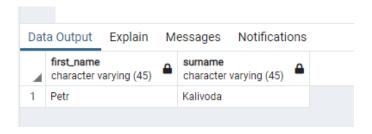
Create a query that will retrieve only selected columns from the selected table SELECT employee_id, first_name FROM employee;

Data	Output	Expl	aiı	n Messages Notific	ations
4	employee bigint	_id	1	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		1	0	Štěpán	
10		1	1	Šimon	
11		1	2	Rostislav	
12		1	2	Dichard	

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';



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'



LIKE

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



NOT LIKE

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



SUBSTRING

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



TRIM

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



COUNT

SELECT COUNT(car_color) FROM car;



SUM

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



MIN

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

4	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);

4	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;



GROUP BY

SELECT employee_id FROM employee GROUP BY employee_id;

4	employee_id bigint	<u></u>	
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;

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

GROUP BY, HAVING, and WHERE

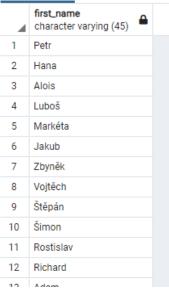
SELECT manager_firstname, manager_surname, salary FROM manager WHERE manager_firstname LIKE 'M%'

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

4	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;



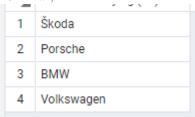
UNION

SELECT first_name FROM employee UNION SELECT manager_firstname from manager;



DISTINCT

SELECT DISTINCT car_brand FROM car;



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 E	хр	lain	Messag	es	Notificatio
4	employee_id	a:	job_id bigint	<u></u>		
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	1	0		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;

4	employee_id bigint	<u></u>	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_id=employee_has_contract.employee_id;

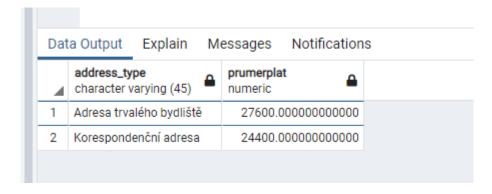
4	employee_id_ bigint	first_name character varying (45)	surname character varying (45)	mail character varying (45)	building_id_ bigint	contract_expiration date	employee_id bigint	job_id bigint
1	1	Hana	Samková	alena.samkova@auto	2	2023-01-01	1	5
2	2	Petr	Kalivoda	petr.kal@automotive.cz	3	2023-01-01	2	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	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	MotI	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

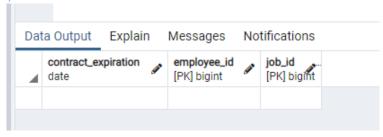
LEFT JOIN job ON employee_has_address.employee_id=job.job_id GROUP BY employee_has_address.add

 $HAVING\ AVG(job.salary) > 5\ ORDER\ BY\ employee_has_address_type\ ;$



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;



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

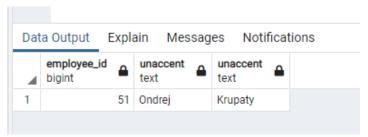
SELECT * FROM employee_has_contract
WHERE contract_expiration <= date_trunc('month', current_date - interval '1' month)
and contract_expiration < date_trunc('month', current_date);



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

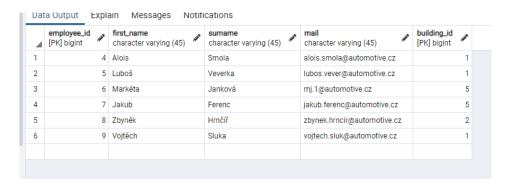
INSERT INTO employee (employee_id, first_name, surname, mail, building_id) VALUES (51,'Ondřej', 'Křupatý', 'okrup@mail.cz', 1); CREATE EXTENSION UNACCENT;

SELECT employee_id, UNACCENT(first_name), UNACCENT(surname) FROM employee WHERE employee_id=51;



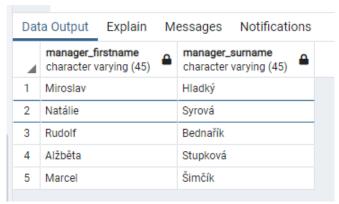
Create a query for pagination in an application (use LIMIT and OFFSET)

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



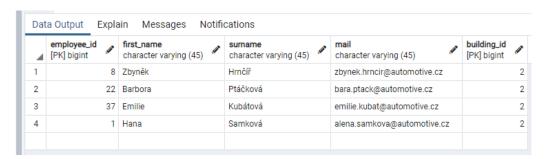
Create a query that will use subquery in FROM

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



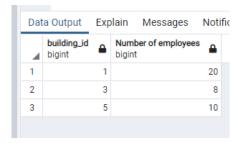
Create a query that will use subquery in WHERE condition

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



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;



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

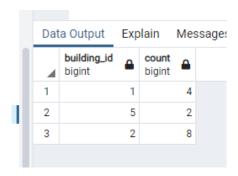
JOIN employee_has_address eha ON eha.employee_id = e.employee_id

JOIN address a ON a.address_id = eha.address_id;

4	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';



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?

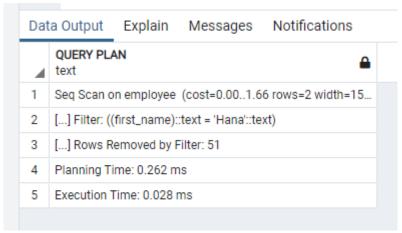
```
ALTER TABLE building ALTER COLUMN building_name TYPE varchar(30); ALTER TABLE employee ALTER COLUMN first_name TYPE varchar(20); ALTER TABLE employee ALTER COLUMN surname TYPE varchar(20); ALTER TABLE customer ALTER COLUMN customer_f_name TYPE varchar(20); ALTER TABLE customer ALTER COLUMN customer_s_name 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:

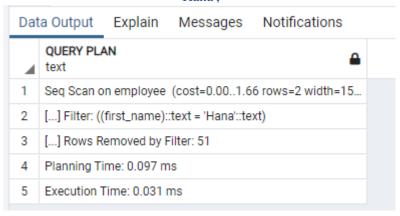
EXPLAIN ANALYZE SELECT employee_id, first_name FROM employee WHERE first_name = 'Hana';



CREATE INDEX n_idx ON employee(first_name);

WITH INDEX:

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



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;

\$\$;

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



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);



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;

Dat	Data Output Explain Messages Notifications										
4	employee_id bigint	4	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	1	Hana	Samková	účetní	29500	Úhretice	Jiráskova	1045		
3	(6	Markéta	Janková	správce odpadů	27500	Louny	Louny	17123		
4	3	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	1	2	Petr	Kalivoda	účetní	29500	Zajecov	K Lukárně	5188		
8	1	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;

Dat	ta Output	Explain Messages	Notifications			
4	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]	[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;