Julia Ruszer 247775

Dominik Gałkowski 247659

Jan Śladowski 247806

Wiktor Żelechowski 247833

**Zadanie 1**  
  
1. Utwórz nową tabelę dotyczącą odwiedzających firmę o nazwie *visitors*, która będzie zawierać wymienione pola:  
– *visitor\_id* – klucz główny automatycznie numerowany od 100 co 10,  
– *employee\_id* – klucz obcy do tabeli pracowników odwołujący się do danych pracownika odpowiedzialnego za odwiedzających,  
– *company* – nazwa firmy, którą reprezentują odwiedzający,  
– *people\_number* – liczba osób odwiedzających,  
– *parking* – informacja o tym, czy odwiedzający korzystali z firmowego parkingu,  
– *enter\_datetime* – data i godzina rozpoczęcia wizyty,  
– *exit\_datetime* – data i godzina zakończenia wizyty.  
Dodatkowo zadbaj o:  
– konieczność podania wartości w polu *employee\_id*,  
– ustawienie domyślnej wartości aktualnej daty i czasu w polu *enter\_datetime*,  
– wprowadzenie takiego ograniczenia wartości w polu *exit\_datetime*, że musi być ona późniejsza niż wartość w polu *enter\_datetime* oraz wcześniejsza albo równa aktualnej dacie i godzinie.

**PostgreSQL**

CREATE SEQUENCE visitor\_id\_seq

START WITH 100

INCREMENT BY 10;

CREATE TABLE visitors (

visitor\_id INT DEFAULT nextval('visitor\_id\_seq') PRIMARY KEY,

employee\_id INT NOT NULL,

company VARCHAR(255),

people\_number INT,

parking BOOLEAN,

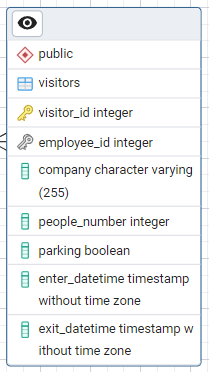
enter\_datetime TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

exit\_datetime TIMESTAMP,

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id),

CHECK (exit\_datetime > enter\_datetime AND exit\_datetime <= CURRENT\_TIMESTAMP)

);

****

**MS SQL Server**

CREATE SEQUENCE visitor\_id\_seq

START WITH 100

INCREMENT BY 10;

CREATE TABLE visitors (

visitor\_id INT DEFAULT NEXT VALUE FOR visitor\_id\_seq PRIMARY KEY,

employee\_id NUMERIC(6, 0) NOT NULL,

company VARCHAR(255),

people\_number INT,

parking BIT,

enter\_datetime DATETIME DEFAULT GETDATE(),

exit\_datetime DATETIME,

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id),

CHECK (exit\_datetime > enter\_datetime AND exit\_datetime <= GETDATE())

);



2. Dodaj dane o odwiedzających pracownika Daniela Favieta:  
– 1 pracownik firmy RSM przyszedł na trwające 1h spotkanie w dniu 13/10/2024 o godzinie 9:00 i skorzystał on z firmowego parkingu,  
– 3 pracowników firmy KPMG przyszło na trwające 1,5h spotkanie w dniu 14/10/2024 o godzinie 10:00.

**PostgreSQL**

INSERT INTO visitors (employee\_id, company, people\_number, parking, enter\_datetime, exit\_datetime)

SELECT d.employee\_id, 'RSM', 1, TRUE, '2024-10-13 09:00:00', '2024-10-13 10:00:00'

FROM employees d

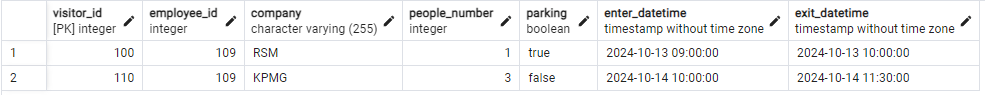
WHERE d.first\_name = 'Daniel' AND d.last\_name = 'Faviet';

INSERT INTO visitors (employee\_id, company, people\_number, parking, enter\_datetime, exit\_datetime)

SELECT d.employee\_id, 'KPMG', 3, FALSE, '2024-10-14 10:00:00', '2024-10-14 11:30:00'

FROM employees d

WHERE d.first\_name = 'Daniel' AND d.last\_name = 'Faviet';

****

**MS SQL Server**

INSERT INTO visitors (employee\_id, company, people\_number, parking, enter\_datetime, exit\_datetime)

SELECT e.employee\_id, 'RSM', 1, 1, '2024-10-13 09:00:00', '2024-10-13 10:00:00'

FROM employees e

WHERE e.first\_name = 'Daniel' AND e.last\_name = 'Faviet';

INSERT INTO visitors (employee\_id, company, people\_number, parking, enter\_datetime, exit\_datetime)

SELECT e.employee\_id, 'KPMG', 3, 0, '2024-10-14 10:00:00', '2024-10-14 11:30:00'

FROM employees e

WHERE e.first\_name = 'Daniel' AND e.last\_name = 'Faviet';

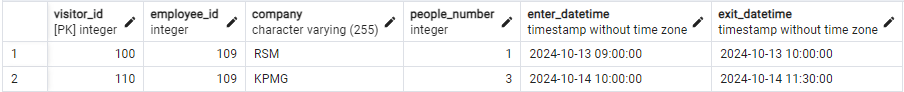


3. Usuń pole *parking* z tabeli odwiedzających.

**PostgreSQL i MS SQL Server**

ALTER TABLE visitors

DROP COLUMN parking;



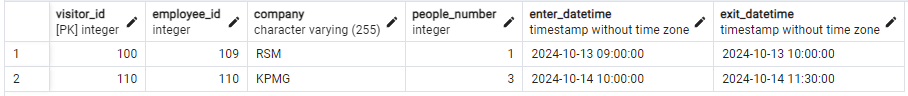
4. Przypisz odwiedzających z firmy KPMG do pracownika Johna Chena, który rzeczywiście uczestniczył w tym spotkaniu.

**PostgreSQL i MS SQL Server**

UPDATE visitors

SET employee\_id = (SELECT employee\_id FROM employees d WHERE d.first\_name = 'John' and d.last\_name = 'Chen')

WHERE company = 'KPMG';

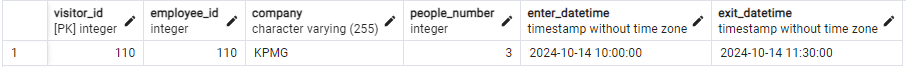


5. Usuń wszystkie informacje o odwiedzających pracownika Daniela Favieta.

**PostgreSQL i MS SQL Server**

DELETE FROM visitors

WHERE employee\_id = (SELECT employee\_id FROM employees d WHERE d.first\_name = 'Daniel' and d.last\_name = 'Faviet')



6. Usuń tabelę odwiedzających.

**PostgreSQL i MS SQL Server**

DROP TABLE visitors;

**Zadanie 2**  
  
1. Wyświetl nazwy departamentów, w których nie jest zatrudniony żaden pracownik.

**PostgreSQL i MS SQL Server**

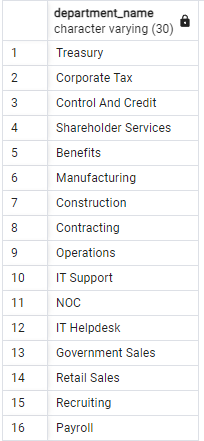
SELECT d.department\_name

FROM departments d

LEFT JOIN employees e ON d.department\_id = e.department\_id

WHERE e.employee\_id IS NULL

GROUP BY D.department\_name, D.department\_id;



2. Dla każdego menedżera wyświetl jego imię i nazwisko oraz różnicę pomiędzy jego wynagrodzeniem a średnim wynagrodzeniem pracowników bezpośrednio mu podlegających.

**PostgreSQL i MS SQL Server**

SELECT

m.first\_name AS manager\_first\_name,

m.last\_name AS manager\_last\_name,

ROUND(m.salary - COALESCE(avg\_emp\_salary.avg\_salary, 0), 2) AS salary\_difference

FROM

employees m

JOIN (

SELECT

manager\_id,

AVG(salary) AS avg\_salary

FROM

employees

WHERE

manager\_id IS NOT NULL

GROUP BY

manager\_id

) avg\_emp\_salary ON m.employee\_id = avg\_emp\_salary.manager\_id

ORDER BY salary\_difference;

****

3. Wyświetl nazwy miast oraz ich kody lokalizacji (nazwij tę kolumnę *code*) w postaci 00000\_KKMMM, gdzie:  
– 00000 – pięciocyfrowy kod pocztowy,  
– KK – kod kraju,  
– MMM – 3 ostatnie litery nazwy miasta zapisane za pomocą wielkich liter.  
Wynik ogranicz do miast, których kod pocztowy składa się z dokładnie 5 cyfr.

**PostgreSQL**

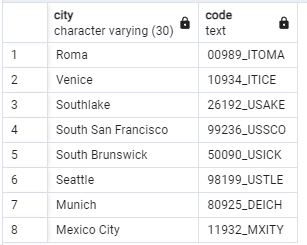
SELECT l.city, CONCAT(l.postal\_code, '\_', l.country\_id,

UPPER(RIGHT(l.city, 3))

) AS code

FROM locations l

WHERE l.postal\_code ~ '^[0-9]{5}$';

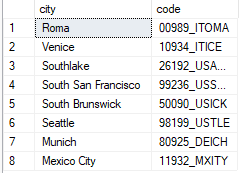
****

**MS SQL Server**

SELECT l.city, CONCAT(l.postal\_code, '\_', l.country\_id, UPPER(RIGHT(l.city, 3))) AS code

FROM locations l

WHERE l.postal\_code LIKE '[0-9][0-9][0-9][0-9][0-9]';



4. Wyświetl nazwy departamentów, w których zatrudnionych jest więcej pracowników niż wynosi średnia liczba pracowników we wszystkich departamentach.

**PostgreSQL i MS SQL Server**

SELECT d.department\_name

FROM departments d

JOIN employees e ON d.department\_id = e.department\_id

GROUP BY d.department\_id, d.department\_name

HAVING

COUNT(e.employee\_id) > (

SELECT AVG(emp\_count)

FROM (

SELECT COUNT(e2.employee\_id) AS emp\_count

FROM departments d2

JOIN employees e2 ON d2.department\_id = e2.department\_id

GROUP BY d2.department\_id

) AS avg\_emp\_count

);

****

5. Dla każdego pracownika wyświetl jego imię i nazwisko, nazwy jego stanowisk oraz informację o tym, które z nich jest aktualne:  
– "actual" w przypadku, gdy dane stanowisko jest aktualnym stanowiskiem pracownika,  
– "archive" w przypadku, gdy dane stanowisko jest poprzednim stanowiskiem pracownika.  
Wyniki posortuj alfabetycznie według nazwiska, imienia oraz nazwy stanowiska.

**PostgreSQL i MS SQL Server**

SELECT

e.first\_name,

e.last\_name,

j.job\_title,

'actual' AS job\_status

FROM

employees e

JOIN

jobs j ON e.job\_id = j.job\_id

UNION ALL

SELECT

e.first\_name,

e.last\_name,

j.job\_title,

'archive' AS job\_status

FROM

employees e

JOIN

job\_history jh ON e.employee\_id = jh.employee\_id

JOIN

jobs j ON jh.job\_id = j.job\_id

ORDER BY

last\_name,

first\_name,

job\_status DESC,

job\_title;

****

****

****

****

**Zadanie 3**  
  
1. Wyświetl imiona i nazwiska menedżerów Diany Lorentz zgodnie z hierarchią w firmie. Do każdego takiego pracownika dopisz jego poziom. Listę pracowników rozpocznij od Diany Lorentz. W rozwiązaniu wykorzystaj rekurencję, klauzulę *with* oraz operator sumy zbiorów.

**PostgreSQL**

WITH RECURSIVE hierarchy AS (

SELECT 0 AS employee\_level,

e.first\_name,

e.last\_name,

e.manager\_id,

e.employee\_id

FROM employees e

WHERE e.first\_name = 'Diana' AND e.last\_name = 'Lorentz'

UNION ALL

SELECT h.employee\_level + 1 AS employee\_level,

m.first\_name,

m.last\_name,

m.manager\_id,

m.employee\_id

FROM employees m

JOIN hierarchy h

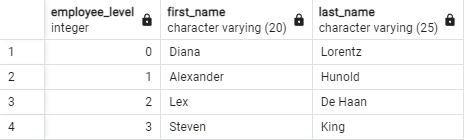
ON h.manager\_id = m.employee\_id

)

SELECT employee\_level, first\_name, last\_name

FROM hierarchy

ORDER BY employee\_level;



**MS SQL Server**

WITH hierarchy AS (

SELECT 0 AS employee\_level,

e.first\_name,

e.last\_name,

e.manager\_id,

e.employee\_id

FROM employees e

WHERE e.first\_name = 'Diana' AND e.last\_name = 'Lorentz'

UNION ALL

SELECT h.employee\_level + 1 AS employee\_level,

m.first\_name,

m.last\_name,

m.manager\_id,

m.employee\_id

FROM employees m

JOIN hierarchy h

ON h.manager\_id = m.employee\_id

)

SELECT employee\_level, first\_name, last\_name

FROM hierarchy

ORDER BY employee\_level;

