Julia Ruszer 247775

Dominik Gałkowski 247659

Jan Śladowski 247806

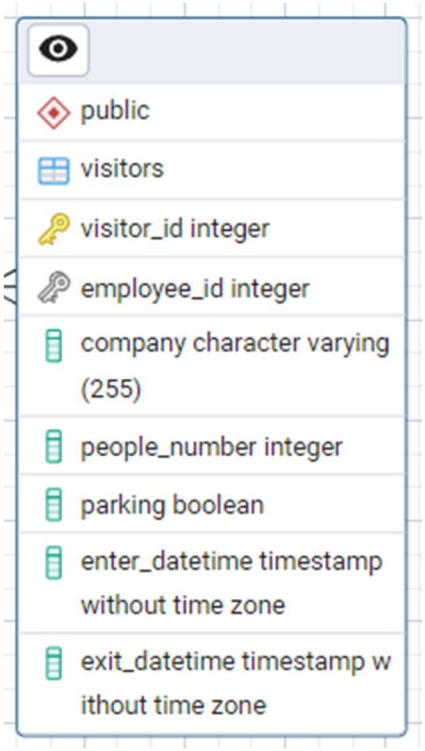
Wiktor Żelechowski 247833

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.

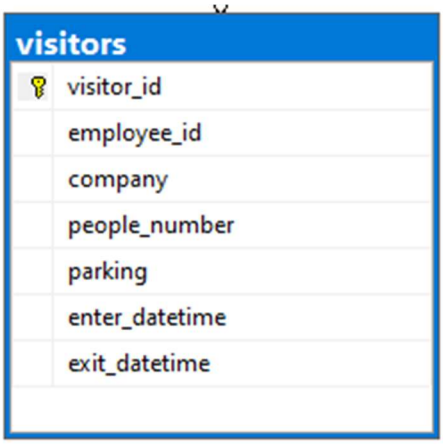
PostgreSQL

```
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';
```

	visitor_id [PK] integer	employee_id integer	company character varying (255)	people_number integer	parking boolean	enter_datetime timestamp without time zone	exit_datetime timestamp without time zone
1	100	109	RSM	1	true	2024-10-13 09:00:00	2024-10-13 10:00:00
2	110	109	KPMG	3	false	2024-10-14 10:00:00	2024-10-14 11:30:00

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';
```

	visitor_id	employee_id	company	people_number	parking	enter_datetime	exit_datetime
1	100	109	RSM	1	1	2024-10-13 09:00:00.000	2024-10-13 10:00:00.000
2	110	109	KPMG	3	0	2024-10-14 10:00:00.000	2024-10-14 11:30:00.000

3. Usuń pole *parking* z tabeli odwiedzających.

PostgreSQL i MS SQL Server

```
ALTER TABLE visitors
DROP COLUMN parking;
```

	visitor_id [PK] integer	employee_id integer	company character varying (255)	people_number integer	enter_datetime timestamp without time zone	exit_datetime timestamp without time zone
1	100	109	RSM	1	2024-10-13 09:00:00	2024-10-13 10:00:00
2	110	109	KPMG	3	2024-10-14 10:00:00	2024-10-14 11:30:00

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';
```

	visitor_id [PK] integer	employee_id integer	company character varying (255)	people_number integer	enter_datetime timestamp without time zone	exit_datetime timestamp without time zone
1	100	109	RSM	1	2024-10-13 09:00:00	2024-10-13 10:00:00
2	110	110	KPMG	3	2024-10-14 10:00:00	2024-10-14 11:30:00

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')
```

	visitor_id [PK] integer	employee_id integer	company character varying (255)	people_number integer	enter_datetime timestamp without time zone	exit_datetime timestamp without time zone
1	110	110	KPMG	3	2024-10-14 10:00:00	2024-10-14 11:30:00

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;
```

	department_name character varying (30)
1	Treasury
2	Corporate Tax
3	Control And Credit
4	Shareholder Services
5	Benefits
6	Manufacturing
7	Construction
8	Contracting
9	Operations
10	IT Support
11	NOC
12	IT Helpdesk
13	Government Sales
14	Retail Sales
15	Recruiting
16	Payroll

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;
```

	manager_first_name character varying (20)	manager_last_name character varying (25)	salary_difference numeric
1	Eleni	Zlotkey	2166.67
2	Gerald	Cambrault	2350.00
3	Kevin	Mourgos	2925.00
4	Shanta	Vollman	3262.50
5	Shelley	Higgins	3708.00
6	Alexander	Hunold	4050.00
7	Nancy	Greenberg	4088.00
8	Alberto	Errazuriz	4233.33
9	Payam	Kaufling	4950.00
10	Karen	Partners	5000.00
11	Adam	Fripp	5025.00
12	Matthew	Weiss	5237.50
13	John	Russell	5500.00
14	Michael	Hartstein	7000.00
15	Lex	De Haan	8000.00
16	Neena	Kochhar	8016.80
17	Den	Raphaely	8220.00
18	Steven	King	12900.00

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}$';
```

	city character varying (30)	code text
1	Roma	00989_ITOMA
2	Venice	10934_ITICE
3	Southlake	26192_USAKE
4	South San Francisco	99236_USSCO
5	South Brunswick	50090_USICK
6	Seattle	98199_USTLE
7	Munich	80925_DEICH
8	Mexico City	11932_MXITY

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]';
```

	city	code
1	Roma	00989_ITOMA
2	Venice	10934_ITICE
3	Southlake	26192_USA...
4	South San Francisco	99236_USS...
5	South Brunswick	50090_USICK
6	Seattle	98199_USTLE
7	Munich	80925_DEICH
8	Mexico City	11932_MXITY

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
    );
```

	department_name character varying (30)
1	Shipping
2	Sales

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;
```

	first_name character varying (20)	last_name character varying (25)	job_title character varying (35)	job_status text
1	Ellen	Abel	Sales Representative	actual
2	Sundar	Ande	Sales Representative	actual
3	Mozhe	Atkinson	Stock Clerk	actual
4	David	Austin	Programmer	actual
5	Hermann	Baer	Public Relations Representative	actual
6	Shelli	Baida	Purchasing Clerk	actual
7	Amit	Banda	Sales Representative	actual
8	Elizabeth	Bates	Sales Representative	actual
9	Sarah	Bell	Shipping Clerk	actual
10	David	Bernstein	Sales Representative	actual
11	Laura	Bissot	Stock Clerk	actual
12	Harrison	Bloom	Sales Representative	actual
13	Alexis	Bull	Shipping Clerk	actual
14	Anthony	Cabrio	Shipping Clerk	actual
15	Gerald	Cambrault	Sales Manager	actual
16	Nanette	Cambrault	Sales Representative	actual
17	John	Chen	Accountant	actual
18	Kelly	Chung	Shipping Clerk	actual
19	Karen	Colmenares	Purchasing Clerk	actual
20	Curtis	Davies	Stock Clerk	actual
21	Lex	De Haan	Programmer	archive
22	Lex	De Haan	Administration Vice President	actual
23	Julia	Dellinger	Shipping Clerk	actual
24	Jennifer	Dilly	Shipping Clerk	actual
25	Louise	Doran	Sales Representative	actual
26	Bruce	Ernst	Programmer	actual
27	Alberto	Errazuriz	Sales Manager	actual
28	Britney	Everett	Shipping Clerk	actual
29	Daniel	Faviet	Accountant	actual
30	Pat	Fay	Marketing Representative	actual

	first_name character varying (20)	last_name character varying (25)	job_title character varying (35)	job_status text
31	Kevin	Feeney	Shipping Clerk	actual
32	Jean	Fleaur	Shipping Clerk	actual
33	Tayler	Fox	Sales Representative	actual
34	Adam	Fripp	Stock Manager	actual
35	Timothy	Gates	Shipping Clerk	actual
36	Ki	Gee	Stock Clerk	actual
37	Girard	Geoni	Shipping Clerk	actual
38	William	Gietz	Public Accountant	actual
39	Douglas	Grant	Shipping Clerk	actual
40	Kimberely	Grant	Sales Representative	actual
41	Nancy	Greenberg	Finance Manager	actual
42	Danielle	Greene	Sales Representative	actual
43	Peter	Hall	Sales Representative	actual
44	Michael	Hartstein	Marketing Representative	archive
45	Michael	Hartstein	Marketing Manager	actual
46	Shelley	Higgins	Accounting Manager	actual
47	Guy	Himuro	Purchasing Clerk	actual
48	Alexander	Hunold	Programmer	actual
49	Alyssa	Hutton	Sales Representative	actual
50	Charles	Johnson	Sales Representative	actual
51	Vance	Jones	Shipping Clerk	actual
52	Payam	Kaufling	Stock Clerk	archive
53	Payam	Kaufling	Stock Manager	actual
54	Alexander	Khoo	Purchasing Clerk	actual
55	Janette	King	Sales Representative	actual
56	Steven	King	President	actual
57	Neena	Kochhar	Accounting Manager	archive
58	Neena	Kochhar	Public Accountant	archive
59	Neena	Kochhar	Administration Vice President	actual
60	Sundita	Kumar	Sales Representative	actual

	first_name character varying (20)	last_name character varying (25)	job_title character varying (35)	job_status text
61	Renske	Ladwig	Stock Clerk	actual
62	James	Landry	Stock Clerk	actual
63	David	Lee	Sales Representative	actual
64	Jack	Livingston	Sales Representative	actual
65	Diana	Lorentz	Programmer	actual
66	Jason	Mallin	Stock Clerk	actual
67	Steven	Markle	Stock Clerk	actual
68	James	Marlow	Stock Clerk	actual
69	Mattea	Marvins	Sales Representative	actual
70	Randall	Matos	Stock Clerk	actual
71	Susan	Mavris	Human Resources Representa...	actual
72	Samuel	McCain	Shipping Clerk	actual
73	Allan	McEwen	Sales Representative	actual
74	Irene	Mikkilineni	Stock Clerk	actual
75	Kevin	Mourgos	Stock Manager	actual
76	Julia	Nayer	Stock Clerk	actual
77	Donald	OConnell	Shipping Clerk	actual
78	Christopher	Olsen	Sales Representative	actual
79	TJ	Olson	Stock Clerk	actual
80	Lisa	Ozer	Sales Representative	actual
81	Karen	Partners	Sales Manager	actual
82	Valli	Pataballa	Programmer	actual
83	Joshua	Patel	Stock Clerk	actual
84	Randall	Perkins	Shipping Clerk	actual
85	Hazel	Philtanker	Stock Clerk	actual
86	Luis	Popp	Accountant	actual
87	Trenna	Rajs	Stock Clerk	actual
88	Den	Raphaely	Stock Clerk	archive
89	Den	Raphaely	Purchasing Manager	actual
90	Michael	Rogers	Stock Clerk	actual

	first_name character varying (20)	last_name character varying (25)	job_title character varying (35)	job_status text
88	Den	Raphaely	Stock Clerk	archive
89	Den	Raphaely	Purchasing Manager	actual
90	Michael	Rogers	Stock Clerk	actual
91	John	Russell	Sales Manager	actual
92	Nandita	Sarchand	Shipping Clerk	actual
93	Ismael	Sciarra	Accountant	actual
94	John	Seo	Stock Clerk	actual
95	Sarath	Sewall	Sales Representative	actual
96	Lindsey	Smith	Sales Representative	actual
97	William	Smith	Sales Representative	actual
98	Stephen	Stiles	Stock Clerk	actual
99	Martha	Sullivan	Shipping Clerk	actual
100	Patrick	Sully	Sales Representative	actual
101	Jonathon	Taylor	Sales Manager	archive
102	Jonathon	Taylor	Sales Representative	archive
103	Jonathon	Taylor	Sales Representative	actual
104	Winston	Taylor	Shipping Clerk	actual
105	Sigal	Tobias	Purchasing Clerk	actual
106	Peter	Tucker	Sales Representative	actual
107	Oliver	Tuvault	Sales Representative	actual
108	Jose Manuel	Urman	Accountant	actual
109	Peter	Vargas	Stock Clerk	actual
110	Clara	Vishney	Sales Representative	actual
111	Shanta	Vollman	Stock Manager	actual
112	Alana	Walsh	Shipping Clerk	actual
113	Matthew	Weiss	Stock Manager	actual
114	Jennifer	Whalen	Administration Assistant	archive
115	Jennifer	Whalen	Public Accountant	archive
116	Jennifer	Whalen	Administration Assistant	actual
117	Eleni	Zlotkey	Sales Manager	actual

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;
```

	employee_level integer	first_name character varying (20)	last_name character varying (25)
1	0	Diana	Lorentz
2	1	Alexander	Hunold
3	2	Lex	De Haan
4	3	Steven	King

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;
```

	employee_level	first_name	last_name
1	0	Diana	Lorentz
2	1	Alexander	Hunold
3	2	Lex	De Haan
4	3	Steven	King