Zapytania do bazy danych college – moduł Skill 1.1

11.01 (NULL w wyrażeniach i funkcjach agregujących)

a) Wykonaj zapytanie:
```
SELECT 34+NULL
```
i skomentuj wynik.

b) Wszystkie dane o tych pracownikach, dla których brakuje numeru PESEL lub daty zatrudnienia (warunek klauzuli WHERE napisz w taki sposób aby był SARG)
13 rekordów
```
SELECT *
FROM employees
WHERE employment_date IS NULL OR pesel IS NULL
```

c) Zapytanie wybierające wszystkie dane z tabeli students_modules.
Zauważ, że dla niektórych egzaminów nie wyznaczono planned_exam_date.
```
SELECT * FROM students_modules
```

d) Zapytanie, które dla każdego rekordu z tabeli students_modules zwróci informację, ile dni minęło od planowanego egzaminu (wykorzystaj funkcję DateDiff).
Dane posortowane malejąco według daty.
Zapamiętaj ile rekordów zwróciło zapytanie.
94 rekordy, w pierwszych dwóch student_id jest 17 i 1
```
SELECT student_id, module_id,
      DateDiff(day,planned_exam_date,getdate()) AS no_of_days
FROM students_modules
ORDER BY planned_exam_date DESC
```

e) Zapytanie zwracające wynik działania funkcji agregującej COUNT na polu planned_exam_date tabeli students_modules. Zwrócona wartość oznaczająca liczbę takich rekordów jest mniejsza niż liczba rekordów w tabeli. Wyjaśnij dlaczego.
16 rekordów
```
SELECT count(planned_exam_date) AS no_of_records
FROM students_modules
```

f) Zapytanie zwracające wynik działania funkcji agregującej COUNT(*) dla tabeli students_modules. Wartość oznaczająca liczbę rekordów jest równa liczbie rekordów w tabeli. Wyjaśnij dlaczego.
Zapytanie zwróciło liczbę 94
```
SELECT count(*) AS no_of_records
FROM students_modules
```

11.02 (DISTINCT)

a) Zapytanie zwracające identyfikatory studentów wraz z datami przystąpienia do egzaminów. Jeśli student danego dnia przystąpił do wielu egzaminów, jego identyfikator ma się pojawić tylko raz. Dane posortowane malejąco względem dat.
50 rekordów
```
SELECT DISTINCT student_id, exam_date
FROM student_grades
ORDER BY exam_date DESC
```

b) Zapytanie zwracające identyfikatory studentów, którzy przystąpili do egzaminu w marcu 2018 roku. Identyfikator każdego studenta ma się pojawić tylko raz. Dane posortowane malejąco według identyfikatorów studentów

10 rekordów

```
SELECT DISTINCT student_id
FROM student_grades
WHERE exam_date BETWEEN '20180301' AND '20180331'
ORDER BY student_id DESC
```

11.03

Spróbuj wykonać zapytanie:

```
SELECT student_id, surname AS family_name
FROM students
WHERE family_name='Fisher'
```

Wyjaśnij dlaczego jest ono niepoprawne a następnie je skoryguj.

11.04 (SARG)

Zapytanie zwracające module_name oraz lecturer_id z tabeli modules z tych rekordów, dla których lecturer_id jest równy 8 lub NULL.

Zapytanie napisz dwoma sposobami – raz wykorzystując funkcję COALESCE (jako drugi parametr przyjmij 0) raz tak, aby predykat podany w warunku WHERE był SARG.

9 rekordów

COALESCE:

```
SELECT module_name, lecturer_id
FROM modules
WHERE lecturer_id=8 OR coalesce(lecturer_id, 0) = 0
```

SARG:

```
SELECT module_name, lecturer_id
FROM modules
WHERE lecturer_id=8 OR lecturer_id IS NULL
```

11.05

Wykorzystaj funkcję CAST i TRY_CAST jako parametr instrukcji SELECT próbując zamienić tekst ABC na liczbę typu INT.

Skomentuj otrzymane wyniki.

```
SELECT cast('ABC' AS int)
SELECT try_cast('ABC' AS int)
```

11.06

Napisz trzy razy instrukcję SELECT wykorzystując funkcję CONVERT zamieniającą dzisiejszą datę na tekst. Jako ostatni parametr funkcji CONVERT podaj 101, 102 oraz 103.

Skomentuj otrzymane wyniki.

```
SELECT convert(varchar(12),getdate(),101)
SELECT convert(varchar(12),getdate(),102)
SELECT convert(varchar(12),getdate(),103)
```

11.07 (LIKE)

Napisz zapytania z użyciem operatora LIKE wybierające nazwy grup (wielkość liter jest nieistotna):

a) zaczynające się na DM

6 rekordów

```
SELECT group_no FROM groups
WHERE group_no LIKE 'DM%'
```

b) niemające w nazwie ciągu '10'

15 rekordów

```
SELECT group_no FROM groups
WHERE group_no NOT LIKE '%10%'
```

c) których drugim znakiem jest M

9 rekordów

```
SELECT group_no FROM groups
WHERE group_no LIKE '_M%'
```

d) których przedostatnim znakiem jest 0 (zero)

11 rekordów

```
SELECT group_no FROM groups
WHERE group_no LIKE '%0_'
```

e) których ostatnim znakiem jest 1 lub 2

12 rekordów

```
SELECT group_no FROM groups
WHERE group_no LIKE '%[12]'
```

f) których pierwszym znakiem nie jest litera D

8 rekordów

```
SELECT group_no FROM groups
WHERE group_no LIKE '[^D]%'
```

g) których drugim znakiem jest dowolna litera z zakresu A-P

10 rekordów

```
SELECT group_no FROM groups
WHERE group_no LIKE '_[A-P]%'
```

11.08 (LIKE i COLLATE)

Napisz zapytania z użyciem operatora LIKE i/lub klauzuli COLLATE:

a) wybierające nazwy wykładów, które w nazwie mają literę o (wielkość liter nie ma znaczenia)

19 rekordów

```
SELECT module_name
FROM modules
WHERE module_name LIKE '%o%'
```

b) wybierające nazwy wykładów, które w nazwie mają dużą literę O

1 rekord, Operational systems

```
SELECT module_name
FROM modules
WHERE module_name collate POLISH_CS_AS LIKE '%O%'
```

c) wybierające nazwy grup, które w nazwie mają trzecią literę i (wielkość liter nie ma znaczenia)

16 rekordów

```
SELECT group_no FROM groups
WHERE group_no LIKE '__i%'
```

d) wybierające nazwy grup, które w nazwie mają trzecią literę małe i
4 rekordy

```
SELECT group_no FROM groups
WHERE group_no collate POLISH_CS_AS LIKE '__i%' --dwa znaki 'podłoga'
```

11.09 (COLLATE)

Instrukcją CREATE utwórz tabelę #tmp (jeśli stworzymy tabelę, której nazwa będzie poprzedzona znakiem #, tabela zostanie automatycznie usunięta po zamknięciu sesji) składającą się z pól:

id int primary key
nazwisko varchar(30) collate polish_cs_as

```
CREATE TABLE #tmp (
id int primary key,
nazwisko varchar(30) collate polish_cs_as)
```

Jedną instrukcją INSERT wprowadź do tabeli #tmp następujące rekordy (zwracając uwagę na wielkość liter):

1 Kowalski
2 kowalski
3 KoWaLsKi
4 KOWALSKI

```
INSERT INTO #tmp VALUES
(1,'Kowalski')
,(2,'kowalski')
,(3,'KoWaLsKi')
,(4,'KOWALSKI')
```

a) Wybierz z tabeli #tmp rekordy, które w polu nazwisko mają (wielkość liter jest istotna):
pierwszą literę duże K
3 rekordy

```
SELECT * FROM #tmp
WHERE nazwisko LIKE 'K%'
```

napis Kowalski
1 rekord

```
SELECT * FROM #tmp
WHERE nazwisko='Kowalski'
```

drugą literę od końca duże K
2 rekordy

```
SELECT * FROM #tmp
WHERE nazwisko LIKE '%K_'
```

b) Wyświetl rekordy, które w polu nazwisko mają (wielkość liter jest nieistotna):
napis kowalski
4 rekordy

```
SELECT * FROM #tmp
WHERE nazwisko collate polish_CI_as = 'kowalski'
```

drugą literę o
4 rekordy

```
SELECT * FROM #tmp
WHERE nazwisko collate polish_CI_as LIKE '_o%'
```

Odpowiedz na pytanie, w którym przypadku, a) czy b), użycie klauzuli COLLATE było konieczne i dlaczego.

11.10

Napisz zapytanie:

```
SELECT DISTINCT surname
FROM students
ORDER BY group_no
```

Wyjaśnij na czym polega błąd i skoryguj zapytanie tak, aby zwracało nazwiska studentów z tabeli students posortowane według numeru grupy.

35 rekordów

```
SELECT surname
FROM students
ORDER BY group_no
```

11.11 (TOP)

a) Napisz zapytanie wybierające 5 pierwszych rekordów z tabeli student_grades, które w polu exam_date mają najdawniejsze daty

5 rekordów

```
SELECT top(5) *
FROM student_grades
ORDER BY exam_date
```

b) Skoryguj zapytanie z punktu a) dodając klauzulę WITH TIES. Skomentuj otrzymany wynik.

6 rekordów

```
SELECT top(5) WITH TIES *
FROM student_grades
ORDER BY exam_date
```

11.12 (TOP, OFFSET)

a) Sprawdź, ile rekordów jest w tabeli student_grades

b) Wybierz 20% początkowych rekordów z tabeli student_grades. Posortuj wynik według exam_date. Sprawdź, ile rekordów zostało zwróconych i wyjaśnij dlaczego.

12 rekordów

```
SELECT top(20) PERCENT *
FROM student_grades
ORDER BY exam_date
```

c) Pomiń pierwszych 6 rekordów i wybierz kolejnych 10 rekordów z tabeli student_grades. Posortuj wynik według exam_date.

pierwszy rekord: student_id=6 i module_id=4

```
SELECT *
FROM student_grades
ORDER BY exam_date
OFFSET 6 ROWS FETCH NEXT 10 ROWS ONLY
```

d) Wybierz wszystkie rekordy z tabeli student_grades z pominięciem pierwszych 20 (sortowanie według exam_date).

38 rekordów

```
SELECT *
FROM student_grades
ORDER BY exam_date
OFFSET 20 ROWS
```

11.13 (INTERSECT, UNION, EXCEPT)

a) Wszystkie nazwiska z tabel students i employees (każde ma się pojawić tylko raz) posortowane według nazwisk

40 rekordów

```
SELECT surname
FROM students
UNION
SELECT surname
FROM employees
ORDER BY surname
```

b) Wszystkie nazwiska z tabel students i employees (każde ma się pojawić tyle razy ile razy występuje w tabelach) posortowane według nazwisk

77 rekordów

```
SELECT surname
FROM students
UNION ALL
SELECT surname
FROM employees
ORDER BY surname
```

c) Te nazwiska z tabeli students, które nie występują w tabeli employees

21 rekordów

```
SELECT surname
FROM students
EXCEPT
SELECT surname
FROM employees
ORDER BY surname
```

d) Te nazwiska z tabeli students, które występują także w tabeli employees

1 rekord – nazwisko Craven

```
SELECT surname
FROM students
INTERSECT
SELECT surname
FROM employees
ORDER BY surname
```

e) Informację, pracownicy których katedr (departments) nie są przypisani jako potencjalni prowadzący do żadnego wykładu (użyj operatora EXCEPT)

1 rekord – Department of Foreign Affairs

```
SELECT department
FROM lecturers
EXCEPT
SELECT department
FROM modules
```

f) Informację, pracownicy których katedr są przypisani jako potencjalni prowadzący wykłady, których nazwa zaczyna się na M

2 rekordy, Department of Economics and Department of Mathematics

```
SELECT department
FROM lecturers
INTERSECT
SELECT department
FROM modules
WHERE module_name LIKE 'M%'
```

g) Te pary id_studenta, id_wykladu z tabeli student_grades, którym nie została przyznana dotychczas żadna ocena
45 rekordów

```
SELECT student_id, module_id
FROM students_modules
EXCEPT
SELECT student_id, module_id
FROM student_grades
```

h) Identyfikatory studentów, którzy zapisali się zarówno na wykład o identyfikatorze 3 jak i 12
Trzech studentów o identyfikatorach 9, 14 i 18

```
SELECT student_id
FROM students_modules
WHERE module_id=3
INTERSECT
SELECT student_id
FROM students_modules
WHERE module_id=12
```

i) Nazwiska i imiona studentów wraz z numerami grup, zapisanych do grup o nazwach zaczynających się na DMIe oraz nazwiska i imiona wykładowców wraz z nazwami katedr, w których pracują. Ostatnia kolumna ma mieć nazwę group_department. Dane posortowane rosnąco według ostatniej kolumny.
Wskazówka: W zapytaniu wybierającym dane o wykładowcach należy użyć złączenia
37 rekordów

```
SELECT surname, first_name, group_no AS group_department
FROM students
WHERE group_no LIKE 'DMIe%'
UNION
SELECT surname, first_name, department
FROM lecturers INNER JOIN employees ON lecturer_id=employee_id
ORDER BY group_department
```

Skill 1.2 – używając składni złączeń napisz zapytania do bazy danych college.

12.01

Identyfikatory i nazwy wykładów na które nie został zapisany żaden student. Dane posortowane malejąco według nazw wykładów.

4 rekordy, wykłady o identyfikatorach 26, 25, 24, 23 (w podanej kolejności)

```
SELECT m.module_id, module_name

FROM modules m LEFT JOIN students_modules sm

       ON m.module_id=sm.module_id

WHERE student_id IS NULL

ORDER BY module_name DESC
```

12.02

Identyfikatory i nazwy wykładów oraz nazwiska wykładowców prowadzących wykłady, na które nie zapisał się żaden student.

4 rekordy, wykłady o identyfikatorach 23, 24, 25, 26

```
SELECT m.module_id, module_name, surname
FROM (modules m LEFT JOIN students_modules sm
        ON m.module_id=sm.module_id)
              LEFT JOIN employees e ON m.lecturer_id=e.employee_id
WHERE sm.module_id IS NULL
```

12.03

Identyfikatory (pod nazwą lecturer_id) i nazwiska wszystkich wykładowców wraz z nazwami wykładów, które prowadzą. Dane posortowane rosnąco według nazwisk.

37 rekordów, w pierwszych dwóch rekordach są wykładowcy o identyfikatorach 5 i 12

```
SELECT employee_id AS lecturer_id, surname, module_name
FROM employees INNER JOIN lecturers ON employee_id=lecturer_id
        LEFT JOIN modules m ON m.lecturer_id=lecturers.lecturer_id
ORDER BY surname
```

12.04

Identyfikatory, nazwiska i imiona pracowników, którzy są wykładowcami.

28 rekordów

```
SELECT employee_id, surname, first_name
FROM employees INNER JOIN lecturers ON lecturer_id=employee_id
```

12.05

Identyfikatory, nazwiska i imiona pracowników, którzy nie są wykładowcami.

14 rekordów

```
SELECT employee_id, surname, first_name
FROM employees LEFT JOIN lecturers ON lecturer_id=employee_id
WHERE lecturer_id IS NULL
```

12.06

Identyfikatory, imiona, nazwiska i numery grup studentów, którzy nie są zapisani na żaden wykład. Dane posortowane rosnąco według nazwisk i imion.

9 rekordów, ostatni: 13 Layla Owen NULL

```
SELECT s.student_id, first_name, surname, group_no
FROM students s LEFT JOIN students_modules sm
        ON s.student_id=sm.student_id
WHERE sm.student_id IS NULL
ORDER BY surname, first_name
```

12.07

Nazwiska, imiona i identyfikatory studentów, którzy przystąpili do egzaminu co najmniej raz oraz daty egzaminów. Jeśli student danego dnia przystąpił do wielu egzaminów, jego dane mają się pojawić tylko raz. Dane posortowane rosnąco względem dat.

50 rekordów, ostatni: Cox Megan, 32, 2018-09-30

```
SELECT DISTINCT surname, first_name, s.student_id, exam_date
FROM students s INNER JOIN student_grades sg
        ON s.student_id=sg.student_id
ORDER BY exam_date
```

12.08

Nazwy **wszystkich** wykładów, liczby godzin przewidziane na każdy z nich oraz identyfikatory, nazwiska i imiona prowadzących. Dane posortowane rosnąco według nazw wykładów a następnie nazwisk i imion prowadzących.

26 rekordów, ostatni: Windows server services, 15, 8, Evans Thomas

```
SELECT module_name, no_of_hours, lecturer_id, surname, first_name
FROM modules LEFT JOIN employees ON lecturer_id=employee_id
ORDER BY module_name, surname, first_name
```

12.09

Identyfikatory, nazwiska i imiona studentów zapisanych na wykład z Statistics, posortowane rosnąco według nazwiska i imienia.

4 studentów o identyfikatorach (w podanej kolejności) 32, 10, 12, 2

```
SELECT s.student_id, surname, first_name
FROM (students s INNER JOIN students_modules sm
       ON s.student_id=sm.student_id)
            INNER JOIN modules m ON m.module_id=sm.module_id
WHERE module_name='Statistics'
ORDER BY surname, first_name
```

12.10

Nazwiska, imiona i stopnie/tytuły naukowe pracowników Department of Informatics. Dane posortowane rosnąco według nazwisk i imion.

7 rekordów, pierwszy: Craven Lily doctor

```
SELECT surname, first_name, acad_position
FROM employees INNER JOIN lecturers ON employee_id=lecturer_id
WHERE department='Department of Informatics'
ORDER BY surname, first_name
```

12.11

Nazwiska i imiona wszystkich pracowników, a dla tych, którzy są wykładowcami także nazwy katedr. Dane posortowane rosnąco według nazwisk oraz malejąco według imion.

42 rekordy, pierwszy: Brown John NULL

```
SELECT surname, first_name, department
FROM employees LEFT JOIN lecturers ON employee_id=lecturer_id
ORDER BY surname, first_name DESC
```

Odpowiedz na pytanie: czy John Brown, dla którego nazwa katedry jest NULL jest wykładowcą, czy na podstawie otrzymanych danych nie jesteśmy w stanie tego stwierdzić?

Czy w udzieleniu odpowiedzi na pytanie pomocny może być projekt logiczny (diagram) bazy danych?

12.12

Nazwiska i imiona wszystkich wykładowców wraz z nazwami katedr, w których pracują. Dane posortowane rosnąco według nazwisk oraz malejąco według imion.

28 rekordów, pierwszy: Brown Jacob, Department of Economics

```
SELECT surname, first_name, department
FROM employees INNER JOIN lecturers ON employee_id=lecturer_id
ORDER BY surname, first_name DESC
```

12.13

Identyfikatory, nazwiska, imiona i stopnie/tytuły naukowe wykładowców, którzy nie prowadzą żadnego wykładu. Dane posortowane malejąco według stopni naukowych.

17 rekordów, pierwszy: 35, Jones Lily, master

```
SELECT l.lecturer_id, surname, first_name, acad_position
FROM (employees INNER JOIN lecturers l ON employee_id=lecturer_id)
        LEFT JOIN modules m ON m.lecturer_id=l.lecturer_id
WHERE module_id IS NULL
ORDER BY acad_position DESC
```

12.14

Imiona i nazwiska wszystkich studentów, nazwy wykładów, na które są zapisani, nazwiska prowadzących te wykłady (pole ma mieć nazwę lecturer_surname) oraz nazwy katedr, w których każdy z wykładowców pracuje. Dane posortowane malejąco według nazw wykładów a następnie rosnąco według nazwisk wykładowców.

103 rekordy, pierwszy: Mason Ben, Web applications, Jones, Department of History

```
SELECT s.surname, s.first_name, module_name,
       e.surname AS lecturer_surname, l.department
FROM students s LEFT JOIN
       (((students_modules sm INNER JOIN modules m
            ON sm.module_id=m.module_id)
       LEFT JOIN lecturers l ON l.lecturer_id=m.lecturer_id)
       LEFT JOIN employees e ON l.lecturer_id=employee_id)
            ON s.student_id=sm.student_id
ORDER BY module_name DESC, e.surname
```

12.15

Liczba godzin wykładów, dla których nie da się ustalić kwoty, jaką trzeba zapłacić za ich przeprowadzenie.

Wskazówka: weź pod uwagę fakt, że nie jesteśmy w stanie ustalić, ile uczelnia musi zapłacić za danych wykład w dwóch przypadkach:

1. Gdy w tabeli modules wartość w polu lecturer_id jest Null
2. Gdy w tabeli modules wartość w polu lecturer_id istnieje, ale w tabeli lecturers wykładowca prowadzący ten wykład nie ma wpisanego acad_position.

Wynikiem jest liczba 165

```
SELECT SUM(no_of_hours) AS sum_hours
FROM modules m LEFT JOIN lecturers l ON m.lecturer_id=l.lecturer_id
WHERE m.lecturer_id IS NULL or acad_position IS NULL
```

12.16

Identyfikatory, nazwy wykładów oraz nazwy katedr odpowiedzialnych za prowadzenie wykładów, dla których nie można ustalić kwoty, jaką trzeba zapłacić za ich przeprowadzenie.

7 wykładów o identyfikatorach 4, 5, 7, 13, 15, 20, 22

```
SELECT module_id, module_name, m.department
FROM modules m LEFT JOIN lecturers l ON m.lecturer_id=l.lecturer_id
WHERE m.lecturer_id IS NULL OR acad_position IS NULL
```

12.17

Nazwy wszystkich wykładów, których nazwa zaczyna się od słowa **computer** (z uwzględnieniem wielkości liter – wszystkie litery małe) oraz liczbę godzin przewidzianych na każdy z tych wykładów, nazwiska prowadzących i nazwy katedr, w których pracują. Dane posortowane malejąco według nazwisk.

Wynikiem jest tabela pusta

```
SELECT module_name, no_of_hours, surname, l.department
FROM modules m LEFT JOIN lecturers l
    ON m.lecturer_id=l.lecturer_id
```

```
            LEFT JOIN employees e ON l.lecturer_id=employee_id
WHERE module_name collate polish_cs_as LIKE 'computer%'
ORDER BY surname DESC
```

12.18

Nazwy wszystkich wykładów, których nazwa zaczyna się od słowa **Computer** (z uwzględnieniem wielkości liter – pierwsza litera duża) oraz liczbę godzin przewidzianych na każdy z tych wykładów, nazwiska prowadzących i nazwy katedr, w których pracują.

Dane posortowane malejąco według nazwisk.

4 rekordy: Computer networks, Computer network devices, Computer programming oraz Computer programming II

```
SELECT module_name, no_of_hours, surname, l.department
FROM modules m LEFT JOIN lecturers l
      ON m.lecturer_id=l.lecturer_id
            LEFT JOIN employees e ON l.lecturer_id=employee_id
WHERE module_name collate polish_cs_as LIKE 'Computer%'
ORDER BY surname DESC
```

12.19

Identyfikatory i nazwiska studentów, którzy nie otrzymali dotychczas oceny z wykładów, na które się zapisali wraz z nazwami tych wykładów (dane każdego studenta mają się pojawić tyle razy z ilu wykładów nie otrzymali oceny). Dane posortowane rosnąco według identyfikatorów studentów.

45 rekordów, np. student o identyfikatorze 2 nie ma ocen z 3 wykładów

```
SELECT s.student_id, surname, module_name
FROM ((students s INNER JOIN students_modules sm
      ON s.student_id=sm.student_id)
            LEFT JOIN student_grades sg ON sg.student_id=sm.student_id
            AND sg.module_id=sm.module_id)
                  INNER JOIN modules m ON m.module_id=sm.module_id
```

```
WHERE sg.student_id IS NULL

ORDER BY s.student_id
```

12.20

Identyfikatory i nazwiska studentów, którzy otrzymali oceny z wykładów, na które się zapisali wraz z nazwami tych wykładów i otrzymanymi ocenami (dane każdego studenta mają się pojawić tyle razy z ilu wykładów nie otrzymali oceny). Dane posortowane rosnąco według identyfikatorów studentów i nazw wykładów a następnie malejąco według otrzymanych ocen.

58 rekordów, pierwszy rekord: 1, Bowen, Computer network devices, 4.5

student o identyfikatorze 2 (Palmer) ma 6 ocen,

```
SELECT s.student_id, surname, module_name, grade

FROM ((students s INNER JOIN students_modules sm

       ON s.student_id=sm.student_id)

            LEFT JOIN student_grades sg ON sg.student_id=sm.student_id
            AND sg.module_id=sm.module_id)

                  INNER JOIN modules m ON m.module_id=sm.module_id

WHERE sg.student_id IS NOT NULL

ORDER BY s.student_id, module_name, grade DESC
```

12.21

W polu department tabeli modules przechowywana jest informacja, która katedra jest odpowiedzialna za prowadzenie każdego z wykładów.

Napisz zapytanie, które zwróci nazwy wykładów, które są prowadzone przez wykładowcę, który nie jest pracownikiem katedry odpowiedzialnej za dany wykład.

11 rekordów, np.:

Wykład z Web applications jest przypisany do Department of Informatics a prowadzi go pracownik Department of History.

Wykład z Management jest przypisany do Department of Management a prowadzi go pracownik Department of Informatics.

```
SELECT module_name, m.department AS dep_responsible,

      l.department AS dpt_lecturer
```

```
FROM modules m inner join lecturers l on m.lecturer_id=l.lecturer_id
WHERE m.department<>l.department
```

12.22

Nazwiska, imiona i PESELe wykładowców, którzy prowadzą przynajmniej jeden wykład wraz z nazwami prowadzonych przez nich wykładów i napisem „wykladowca” w ostatniej kolumnie

oraz

nazwiska, imiona, numery grup **wszystkich** studentów wraz z nazwami wykładów na które się zapisali i napisem „student” w ostatniej kolumnie.

Trzecia kolumna ma mieć nazwę pesel/grupa a ostatnia student/wykladowca.

Dane posortowane rosnąco według nazw wykładów a następnie według kolumny student/wykladowca.

119 rekordów. Rekord nr 11: Chapman Grace DMZa3012 Advanced Statistics student

```
SELECT surname, first_name, PESEL AS "pesel/grupa",
       module_name, 'wykladowca' AS [student/wykladowca]
FROM employees INNER JOIN modules m
       ON employee_id=lecturer_id
UNION
SELECT surname, first_name, group_no, module_name, 'student'
FROM students s LEFT JOIN (modules m INNER JOIN students_modules sm
       ON m.module_id=sm.module_id) ON s.student_id=sm.student_id
ORDER BY module_name, [student/wykladowca]
```

Skill 2.1. Query data by using subqueries and APPLY.

21.01

Nazwiska i imiona studentów zapisanych na wykład z matematyki (Mathematics). Dane posortowane według nazwisk. Użyj składni podzapytania.

6 rekordów.

Studenci o nazwiskach (kolejno): Bowen, Foster, Holmes, Hunt, Palmer, Powell

```
SELECT surname, first_name
```

```
FROM students where student_id IN
      (SELECT student_id
       FROM students_modules
       WHERE module_id IN
            (SELECT module_id
             FROM modules
             WHERE module_name='Mathematics'))
ORDER BY surname
```

21.02

Identyfikatory i nazwy wykładów, na które nie został zapisany żaden student. Dane posortowane malejąco według nazw wykładów.

Użyj składni podzapytania.

4 rekordy, wykłady o identyfikatorach 26, 25, 24 i 23 (w podanej kolejności)

```
SELECT module_id, module_name
FROM modules
WHERE module_id NOT IN
       (SELECT module_id FROM students_modules)
        ORDER BY module_name DESC
```

21.03

Identyfikatory studentów, którzy przystąpili do egzaminu zarówno 2018-03-22 jak i 2018-09-30. Dane posortowane malejąco według identyfikatorów.

Napisz dwie wersje tego zapytania: raz używając składni podzapytania, drugi raz operatora INTERSECT.

Studenci o identyfikatorach 18 i 2 (w podanej kolejności)

Podzapytanie:

```
SELECT student_id
FROM student_grades
```

```
WHERE exam_date='20180322' AND student_id IN
      (SELECT student_id
       FROM student_grades
       WHERE exam_date='20180930')
ORDER BY student_id DESC
```

Z operatorem INTERSECT:

```
SELECT student_id
FROM student_grades
WHERE exam_date='20180322'
      INTERSECT
SELECT student_id
FROM student_grades
WHERE exam_date='20180930'
ORDER BY student_id DESC
```

21.04

Identyfikatory, nazwiska, imiona i numery grup studentów, którzy zapisali się zarówno na wykład o identyfikatorze 2 jak i 4. Dane posortowane malejąco według nazwisk.

Użyj składni podzapytania a w zapytaniu zewnętrznym także złączenia.

3 rekordy, studenci o identyfikatorach 16, 3, 20 (w podanej kolejności)

```
SELECT s.student_id,surname, first_name, group_no
FROM students s INNER JOIN students_modules sm
      ON s.student_id=sm.student_id
WHERE sm.module_id=2 AND s.student_id IN
      (SELECT sm.student_id
       FROM students_modules sm
       WHERE sm.module_id=4)
ORDER BY surname DESC
```

21.05

Imiona, nazwiska i numery grup studentów z grup, których nazwa zaczyna się na DMIe i kończy cyfrą 1 i którzy nie są zapisani na wykład z „Ancient history”.

Użyj składni zapytania negatywnego a w zapytaniu wewnętrznym także złączenia.

3 rekordy (studenci z grupy DMIe1011 o nazwiskach Hunt, Holmes i Lancaster)

```
SELECT first_name, surname, group_no
FROM students
WHERE group_no LIKE 'DMIe%1' AND student_id NOT IN
      (SELECT student_id FROM students_modules sm
            INNER JOIN modules m ON sm.module_id=m.module_id
       WHERE module_name='Ancient history')
```

21.06

Nazwy wykładów o najmniejszej liczbie godzin. Zapytanie, oprócz nazw wykładów, ma zwrócić także liczbę godzin.

Użyj operatora ALL.

Jeden wykład: Advanced Statistics, 9 godzin

```
SELECT module_name, no_of_hours

FROM modules

WHERE no_of_hours <= ALL

(SELECT no_of_hours FROM modules)
```

21.07

Identyfikatory i nazwiska studentów, którzy otrzymali ocenę wyższą od najniższej. Dane każdego studenta mają się pojawić tyle razy, ile takich ocen otrzymał.

Użyj operatora ANY. W zapytaniu nie wolno posługiwać się liczbami oznaczającymi oceny 2, 3, itd.) ani funkcjami agregującymi (MIN, MAX).

45 rekordów

```
SELECT s.student_id, surname

FROM students s INNER JOIN student_grades sg

      ON s.student_id=sg.student_id

WHERE grade > ANY

(SELECT grade FROM grades)
```

Sprawdź, czy liczba rekordów zwróconych przez zapytanie jest poprawna, wykonując odpowiednie zapytanie do tabeli student_grades (wybierające rekordy, w których ocena jest wyższa niż 2).

21.08

Napisz jedno zapytanie, które zwróci dane o najmłodszym i najstarszym studencie (do połączenia tych danych użyj jednego z operatorów typu SET).

W zapytaniu nie wolno używać funkcji agregujących (MIN, MAX).

Uwaga: należy uwzględnić fakt, że data urodzenia w tabeli students może być NULL, do porównania należy więc wybrać rekordy, które w polu date_of_birth mają wpisane daty.

Użyj operatora ALL.

Najstarszym studentem jest Melissa Hunt urodzona 1978-10-18

Najmłodszym studentem jest Layla Owen urodzona 2001-06-20

```
SELECT *
FROM students
WHERE date_of_birth <= ALL
(SELECT date_of_birth FROM students where date_of_birth is not null)
      UNION
SELECT *
FROM students
WHERE date_of_birth >= ALL
(SELECT date_of_birth FROM students where date_of_birth is not null)
```

Napisz zapytanie do tabeli students i sprawdź, czy otrzymane dane o najmłodszych i najstarszych studentach są poprawne.

21.09a

Identyfikatory, imiona i nazwiska studentów z grupy DMIe1011, którzy otrzymali oceny z egzaminu wcześniej, niż wszyscy pozostali studenci z innych grup (nie uwzględniamy studentów, którzy nie są zapisani do żadnej grupy). Dane każdego studenta mają się pojawić tylko raz.

Użyj złączenia i operatora ALL.

3 rekordy, studenci o identyfikatorach 1, 3 i 6

```
SELECT DISTINCT s.student_id, first_name, surname
FROM students s INNER JOIN student_grades sg
```

```
        ON sg.student_id=s.student_id
WHERE group_no = 'DMIe1011' AND exam_date < ALL
        (SELECT exam_date FROM students s INNER JOIN
                student_grades sg  ON sg.student_id=s.student_id
        WHERE group_no <> 'DMIe1011')
```

21.09b

Jak wyżej, ale tym razem należy uwzględnić studentów, którzy nie są zapisani do żadnej grupy.

Wynikiem jest tabela pusta

```
SELECT DISTINCT s.student_id, first_name, surname
FROM students s INNER JOIN student_grades sg
        ON sg.student_id=s.student_id
WHERE group_no = 'DMIe1011' AND exam_date < ALL
        (SELECT exam_date FROM students s INNER JOIN
                student_grades sg  ON sg.student_id=s.student_id
        WHERE group_no <> 'DMIe1011' OR group_no IS NULL)
```

Odpowiedz na pytanie, jaki jest identyfikator studenta, którego ocena spowodowała, że wynikiem jest tabela pusta?

Jest to student o identyfikatorze 2, który otrzymał ocenę 2018-02-21 i była to najwcześniej przyznana ocena (w tym dniu ocenę otrzymało jeszcze dwóch studentów o identyfikatorach 1 i 3 z grupy DMIe1011).

21.10

Nazwy wykładów, którym przypisano największą liczbę godzin (wraz z liczbą godzin).

Wykorzystaj składnię podzapytania z operatorem =. W zapytaniu wewnętrznym użyj funkcji MAX.

Jeden rekord: Econometrics, 45 godzin

```
SELECT module_name, no_of_hours
FROM modules
WHERE no_of_hours =
      (SELECT max(no_of_hours)
       FROM modules)
```

21.11

Nazwy wykładów, którym przypisano liczbę godzin większą od najmniejszej.

Użyj funkcji MIN i składni podzapytania z operatorem >.

25 rekordów

```
SELECT module_name
FROM modules
WHERE no_of_hours >
      (SELECT min(no_of_hours)
       FROM modules)
```

21.12a

Wszystkie dane o najstarszym studencie z każdej grupy.

Użyj funkcji MIN i składni podzapytania skorelowanego z operatorem =.

11 rekordów, np. w grupie DMIe1013 najstarszy jest Elliot Fisher, ur. 1998-07-19

```
SELECT *
FROM students s1
WHERE date_of_birth =
      (SELECT min(date_of_birth)
```

```
FROM students s2
WHERE s1.group_no=s2.group_no)
```

21.12b

Wszystkie numery grup z tabeli students posortowane według numerów grup. Każda grupa ma się pojawić jeden raz.

```
SELECT DISTINCT group_no FROM students
```

Zapytanie zwróciło 13 rekordów. Ponieważ jedną z wartości jest NULL, więc studenci są przypisani do 12 różnych grup. Poprzednie zapytanie, zwracające dane o najmłodszym studencie z każdej grupy, zwróciło 11 rekordów. Znajdź przyczynę tej różnicy.

Odpowiedź:

Zapytanie:

```
SELECT *
FROM students
ORDER BY group_no
```

zwróciło dane, z których wynika, że do grupy ZMIe2012 został przypisany tylko jeden student (o identyfikatorze 18), który w dodatku nie ma wpisanej daty urodzenia. Zapytanie zwracające dane o najmłodszym studencie z każdej grupy pominęło więc tę grupę.

21.13

Identyfikatory, nazwiska i imiona studentów, którzy otrzymali ocenę 5.0. Nazwisko każdego studenta ma się pojawić jeden raz.

Użyj operatora EXISTS.

6 studentów o identyfikatorach 1, 2, 14, 18, 19, 21

```
SELECT student_id, surname, first_name
FROM students s
WHERE EXISTS
      (SELECT *
      FROM student_grades sg
      WHERE s.student_id=sg.student_id AND grade=5)
```

Napisz zapytanie:

```
SELECT * FROM student_grades where grade=5
```

i sprawdź otrzymany wynik.

21.14a

Wszystkie dane o wykładach, w których uczestnictwo wymaga wcześniejszego uczestnictwa w wykładzie o identyfikatorze 3.

Użyj operatora EXISTS.

Trzy wykłady o identyfikatorach 10, 16 i 25

```
SELECT *
FROM modules m1
WHERE EXISTS
      (SELECT *
      FROM modules m2
      WHERE m1.module_id=m2.module_id AND m2.preceding_module=3)
```

Aby sprawdzić otrzymany wynik napisz zapytanie:

```
SELECT module_id
FROM modules
WHERE preceding_module=3
```

21.14b

Nazwy wykładów, w których uczestnictwo wymaga wcześniejszego uczestnictwa w wykładzie z matematyki (Mathematics).

Użyj operatora EXISTS.

Wskazówka: id. wykładu z matematyki znajdź przy pomocy odpowiedniego zapytania.

Dwa wykłady: Statistics i Mathematics II

```
SELECT module_name

FROM modules m1

WHERE EXISTS

      (SELECT *

       FROM modules m2

       WHERE m1.module_id=m2.module_id AND m2.preceding_module IN

            (SELECT module_id FROM modules

             WHERE module_name='Mathematics'))
```

21.15a

Dane studentów z grupy DMIe1011 wraz z **najwcześniejszą** datą planowanego dla nich egzaminu (pole planned_exam_date w tabeli students_modules). Zapytanie nie zwraca danych o studentach, którzy nie mają wyznaczonej takiej daty. Sortowanie rosnące według planned_exam_date a następnie student_id.

Użyj operatora APPLY.

Uwaga: należy uwzględnić fakt, że data planowanego egzaminu może być NULL.

3 rekordy, studenci o identyfikatorach 3, 29 i 1 (w takiej kolejności)

Najwcześniejsza planned_exam_date dla studenta o id=3 to 2018-03-21

```
SELECT *
```

```
FROM students s
CROSS APPLY
      (SELECT Top(1) planned_exam_date
       FROM students_modules sm
       WHERE s.student_id=sm.student_id
            and planned_exam_date IS NOT NULL
       ORDER BY planned_exam_date) AS t
WHERE group_no='DMIe1011'
ORDER BY planned_exam_date, student_id
```

21.15b

Jak wyżej, tylko zapytanie ma zwrócić **najpóźniejszą** datę planowanego egzaminu. Ponadto zapytanie ma także zwrócić dane o studentach, którzy nie mają wyznaczonej takiej daty. Sortowanie rosnące według planned_exam_date.

Użyj operatora APPLY.

6 rekordów, studenci o identyfikatorach 4, 6, 30 (dla których planned_exam_date jest NULL)

oraz 29, 3 i 1 (z istniejącą planned_exam_date).

Najwcześniejsza planned_exam_date dla studenta o id=3 to 2018-10-13

```
SELECT *
FROM students s
OUTER APPLY
      (SELECT Top(1) planned_exam_date
       FROM students_modules sm
       WHERE s.student_id=sm.student_id
            and planned_exam_date IS NOT NULL
       ORDER BY planned_exam_date DESC) AS t
WHERE group_no='DMIe1011'
ORDER BY planned_exam_date, student_id
```

Zapytanie różni się od poprzedniego tym, że należy użyć operatora OUTER APPLY (zamiast CROSS APPLY) oraz w podzapytaniu sortowanie według planned_exam_date ma być malejące (DESC).

21.16a

Identyfikatory i nazwiska studentów oraz dwie najlepsze oceny dla każdego studenta wraz z datami ich przyznania. Zapytanie uwzględnia tylko studentów, którym została przyznana co najmniej jedna ocena.

Użyj operatora APPLY.

37 rekordów.

Ostatni rekord: 33, Bowen, 2.0, 2018-09-23

Np. w przypadku studentów o id=1, 2 i 3 zwrócone zostały po dwie oceny.

W przypadku studenta o id=4 jedna ocena.

Student o id=5 nie otrzymał żadnej oceny.

```
SELECT student_id, surname, grade, exam_date
FROM students s
CROSS APPLY
(SELECT Top(2) grade, exam_date
 FROM student_grades sg
 WHERE s.student_id=sg.student_id
 ORDER BY grade DESC) AS t
```

21.16b

Identyfikatory i nazwiska studentów oraz dwie najgorsze oceny dla każdego studenta wraz z datami ich przyznania. Zapytanie uwzględnia także studentów, którym nie została przyznana żadna ocena.

Użyj operatora APPLY.

51 rekordów.

Pierwszy: 1, Bowen, 2.0, 2018-03-22

Ostatni: 35, Fisher, NULL, NULL

W kilku przypadkach (np. studenci o id: 5, 11, 13, 16) studenci nie otrzymali żadnej oceny.

```
SELECT student_id, surname, grade, exam_date
FROM students s
```

```
OUTER APPLY
(SELECT Top(2) grade, exam_date
 FROM student_grades sg
 WHERE s.student_id=sg.student_id
 ORDER BY grade) AS t
```

Zapytanie różni się od poprzedniego tym, że należy użyć operatora OUTER APPLY (zamiast CROSS APPLY) oraz w podzapytaniu sortowanie według grade ma być rosnące.

21.17

Identyfikatory i nazwiska studentów oraz kwoty dwóch ostatnich wpłat za studia wraz z datami tych wpłat. Zapytanie uwzględnia także studentów, którzy nie dokonali żadnej wpłaty.

Użyj operatora APPLY.

54 rekordy.

Trzeci: 2, Palmer, 450.00, 2018-10-30

W kilku przypadkach (np. studenci o id: 9, 10, 20) studenci nie dokonali żadnej wpłaty.

```
SELECT student_id, surname, fee_amount, date_of_payment
FROM students s
OUTER APPLY
(SELECT Top(2) fee_amount, date_of_payment
 FROM tuition_fees tf
 WHERE s.student_id=tf.student_id
 ORDER BY date_of_payment DESC) AS t
```

21.18

Nazwę modułu poprzedzającego dla modułu Databases.

Information technology

```
select module_name
from modules
```

```
where module_id in
(select preceding_module
from modules
where module_name='databases')
```

Skill 2.2. Query data by using table expressions

22.01a – Widok (view) i funkcja (function)

Nazwiska i imiona studentów zapisanych na wykład z matematyki (Mathematics). Dane posortowane według nazwisk. Użyj składni podzapytania.

6 rekordów.

Studenci o nazwiskach (kolejno): Bowen, Foster, Holmes, Hunt, Palmer, Powell

```
SELECT surname, first_name
FROM students where student_id IN
      (SELECT student_id
       FROM students_modules
       WHERE module_id IN
            (SELECT module_id
             FROM modules
             WHERE module_name='Mathematics'))
ORDER BY surname
```

22.01b – Widok (view) i funkcja (function)

Napisz funkcję o nazwie studmod_f, która zwróci nazwiska i imiona studentów zapisanych na wykład o nazwie przekazanej do funkcji przy pomocy parametru. Uruchom funkcję podając jako parametr nazwę wybranego wykładu.

SELECT * FROM studmod_f('Mathematics') zwraca 6 rekordów (jak powyższe zapytanie)

SELECT * FROM studmod_f('Statistics') zwraca 4 rekordy

SELECT * FROM studmod_f('Databases') zwraca 2 rekordy

```
CREATE OR ALTER FUNCTION studmod_f(@module AS VARCHAR(100))
RETURNS TABLE AS RETURN
SELECT surname, first_name
FROM students where student_id IN
      (SELECT student_id
       FROM students_modules
       WHERE module_id IN
            (SELECT module_id
             FROM modules
             WHERE module_name=@module))
ORDER BY surname
```

Modyfikacje, jakie należy wykonać w powyższym zapytaniu aby powstała funkcja oznaczono kolorem **zielonym**. Zwróć uwagę, że w instrukcji SELECT nie może być użyta klauzula ORDER BY.

22.01c – Widok (view) i funkcja (function)

Jedną z różnic między funkcją a widokiem jest to, że do funkcji można przekazać parametr a do widoku nie. Aby przekazać parametr do widoku, można jednak użyć context info lub session context (zobacz Chapter 1, Skill 3).

Utwórz widok o nazwie studmod_v, który zwróci nazwiska i imiona studentów zapisanych na wykład o nazwie przekazanej do widoku przy pomocy session context. Wykorzystaj mechanizm session context do przekazania nazwy wykładu do widoku i uruchom widok.

Instrukcja tworząca widok:

```
CREATE OR ALTER VIEW studmod_v AS
SELECT surname, first_name
FROM students where student_id IN
      (SELECT student_id
       FROM students_modules
       WHERE module_id IN
            (SELECT module_id
             FROM modules
             WHERE module_name=session_context(N'module')))
```

Instrukcja tworząca parę @key-@value session context i uruchamiająca widok:

```
exec sp_set_session_context @key=N'module', @value='Statistics'
select * from studmod_v
```

22.02 – Funkcja ROW_NUMBER()

Wszystkie dane z tabeli student_grades, w ramach każdego module_id (partition by) posortowane według daty egzaminu a następnie identyfikatora studenta oraz ponumerowane kolejnymi liczbami. Pole zawierające kolejny numer oceny w ramach każdego module_id ma mieć nazwę sequence_num.

58 rekordów,

pierwsze trzy dotyczą wykładu o id=1 (w kolumnie sequence_num są liczby 1, 2, 3)

kolejnych pięć dotyczy wykładu o id=2 (w kolumnie sequence_num są liczby 1-5), itd.

```
SELECT ROW_NUMBER() OVER
```

```
        (PARTITION BY module_id
         ORDER BY exam_date, student_id) AS sequence_num,
module_id, exam_date, student_id, grade
FROM student_grades;
```

22.03 – Funkcja ROW_NUMBER()

Wszystkie dane z tabeli student_grades, w ramach każdego **student_id** posortowane według daty egzaminu oraz ponumerowane kolejnymi liczbami. Zapytanie ma zwrócić jedynie dane o ocenach pozytywnych (większych niż 2). Pole zawierające kolejny numer oceny w ramach każdego student_id ma mieć nazwę sequence_num.

45 rekordów

pierwsze cztery dotyczą studenta o id=1 (w kolumnie sequence_num są liczby 1-4)

kolejne cztery dotyczą studenta o id=2 (w kolumnie sequence_num są liczby 1-4), itd.

```
SELECT ROW_NUMBER() OVER
        (PARTITION BY student_id
         ORDER BY exam_date) AS sequence_num,
student_id, exam_date, module_id, grade
FROM student_grades
WHERE grade>2;
```

22.04 – Funkcja ROW_NUMBER()

Identyfikatory i nazwiska studentów oraz daty egzaminów, w ramach każdego student_id posortowane według daty egzaminu oraz ponumerowane kolejnymi liczbami. Zapytanie ma zwrócić jedynie dane o ocenach negatywnych (równych 2). Pole zawierające kolejny numer oceny w ramach każdego student_id ma mieć nazwę sequence_num.

13 rekordów

studenci o identyfikatorach 2, 3 i 12 mają po dwie oceny 2,0

studenci o identyfikatorach 1, 6, 7, 8, 10, 20 i 33 po jednej

```
SELECT ROW_NUMBER() OVER
      (PARTITION BY sg.student_id
       ORDER BY exam_date) AS sequence_num,
       s.student_id, exam_date
FROM student_grades sg INNER JOIN students s
      ON sg.student_id=s.student_id
WHERE grade=2;
```

22.05 – Funkcja ROW_NUMBER()

Wszystkie dane z tabeli students, grupami. W ramach każdej grupy dane posortowane według daty urodzenia studenta. W ramach każdej grupy rekordy mają być ponumerowane.

35 rekordów.

Zauważ, że w pierwszych 7 rekordach group_no jest NULL i rekordy te są traktowane jako jedna partycja.

W grupie DMIe1011 jest 6 studentów (data urodzenia pierwszego jest NULL).

W grupie DMIE1014 jest jeden student.

Data urodzenia pierwszego studenta w grupie ZMIe2011 to 1990-01-30 (Melissa Hunt).

W ostatnich dwóch rekordach w polu zawierającym kolejny numer w danej grupie są wartości 5 (w przedostatnim rekordzie) i 1 (w ostatnim).

```
SELECT ROW_NUMBER() OVER
      (PARTITION BY group_no
       ORDER BY date_of_birth) AS rownum,
```

```
        group_no, student_id, surname, first_name, date_of_birth
FROM students;
```

22.06a

Identyfikator, nazwisko i datę ostatniego egzaminu dla każdego studenta. Zapytanie ma zwrócić jedynie dane o studentach, którzy przystąpili co najmniej do jednego egzaminu.

Napisz zapytanie w dwóch wersjach: raz używając składni derived tables, raz CTE.

21 rekordów

Trzeci: 3, Hunt, 2018-09-20

Ostatni: 33, Bowen, 2018-09-23

Derived tables:

```
SELECT dt.student_id, surname, exam_date
FROM
(SELECT ROW_NUMBER() OVER
       (PARTITION BY s.student_id
        ORDER BY exam_date DESC) AS rownum,
        s.student_id, surname, exam_date
 FROM students s INNER JOIN student_grades sg
      ON s.student_id=sg.student_id) AS dt
WHERE rownum=1;
```

CTE:

```
WITH CTE_table AS
      (SELECT ROW_NUMBER() OVER
       (PARTITION BY s.student_id
        ORDER BY exam_date DESC) AS rownum,
            s.student_id, surname, exam_date
       FROM students s INNER JOIN student_grades sg
            ON s.student_id=sg.student_id)
SELECT student_id, surname, exam_date
FROM CTE_table
WHERE rownum=1;
```

22.06b

Korzystając z poprzedniego zapytania utwórz widok (VIEW) o nazwie last_exam zwracający identyfikator, nazwisko i datę ostatniego egzaminu dla każdego studenta.

Uruchom widok i sprawdź poprawność jego działania.

Wskazówka: Aby utworzyć widok, należy zapytanie poprzedzić instrukcją CREATE VIEW.

Uwaga: Widok nie może mieć takiej samej nazwy jak inny obiekt w bazie danych (tabela, funkcja).

SELECT * FROM last_exam
zwraca te same dane, co powyższe zapytanie.

```
CREATE OR ALTER VIEW last_exam AS
WITH CTE_table AS
(SELECT ROW_NUMBER() OVER
       (PARTITION BY s.student_id
        ORDER BY exam_date DESC) AS rownum,
           s.student_id, surname, exam_date
        FROM students s INNER JOIN student_grades sg
           ON s.student_id=sg.student_id)
SELECT student_id, surname, exam_date
FROM CTE_table
WHERE rownum=1;
```

22.06c

Zmodyfikuj utworzony widok o nazwie last_exam, aby wywołując go instrukcją SELECT można było podać liczbę oznaczającą, ile rekordów z danymi o ostatnich egzaminach ma zostać zwróconych dla każdego studenta.

Wskazówka: widok powinien zwracać dane o wszystkich egzaminach dla każdego studenta, dzięki czemu zapytanie uruchamiające widok może zawierać klauzulę WHERE zawierającą warunek wskazujący, ile ostatnich egzaminów dla każdego studenta ma zostać zwróconych.

SELECT * FROM last_exam
WHERE rownum=1

zwraca 21 rekordów

SELECT * FROM last_exam
WHERE rownum<=3

zwraca 46 rekordów

```
CREATE OR ALTER VIEW last_exam AS

WITH CTE_table AS

(SELECT ROW_NUMBER() OVER

       (PARTITION BY s.student_id

        ORDER BY exam_date DESC) AS rownum,

            s.student_id, surname, exam_date

        FROM students s INNER JOIN student_grades sg

            ON s.student_id=sg.student_id)

SELECT student_id, surname, exam_date, rownum

FROM CTE_table

WHERE rownum=1;
```

W widoku należy **usunąć ostatnią klauzulę WHERE** oraz w ostatniej instrukcji SELECT **umieścić na liście pole rownum**.

22.06d

Korzystając z poprzedniego zapytania utwórz funkcję o nazwie last_exams zwracającą identyfikator, nazwisko i datę tylu ostatnich egzaminów każdego studenta, ile wynosi wartość parametru funkcji (np. jeśli jako parametr funkcji podana zostanie liczba 4, to funkcja ma zwrócić daty ostatnich 4 egzaminów każdego studenta).

Uruchom funkcję i sprawdź poprawność jej działania.

SELECT * FROM last_exams(1)

zwraca 21 rekordów

SELECT * FROM last_exams(2)

zwraca 37 rekordów

SELECT * FROM last_exams(4)

zwraca 52 rekordy

```
CREATE OR ALTER FUNCTION last_exams(@exam_no AS int) RETURNS TABLE AS
RETURN

WITH CTE_table AS

      (SELECT ROW_NUMBER() OVER

       (PARTITION BY s.student_id

        ORDER BY exam_date DESC) AS rownum,

            s.student_id, surname, exam_date

        FROM students s INNER JOIN student_grades sg

            ON s.student_id=sg.student_id)

SELECT student_id, surname, exam_date

FROM CTE_table

WHERE rownum<=@exam_no;
```

22.07a

Wszystkie dane o dwóch najmłodszych studentach w każdej grupie. W zapytaniu pomiń dane o studentach, którzy nie są przypisani do żadnej grupy oraz o tych, którzy nie mają przypisanej daty urodzenia.

Napisz zapytanie w dwóch wersjach: raz używając składni derived tables, raz CTE.

17 rekordów.

W grupach DMIe1014, DMZa3013, DZZa3001, ZMIe2014 oraz ZZIe2003 tylko jeden student ma wpisaną datę urodzenia.

6 rekord: Rebecca Mason z grupy DMZa3012, ur. 1988-12-10

15 rekord: Katie Powell z grupy ZZIe2012, ur. 2001-01-20

Derived tables:

```
SELECT rownum, group_no, student_id, surname, first_name,
       date_of_birth
FROM
      (SELECT ROW_NUMBER() OVER
       (PARTITION BY group_no
        ORDER BY date_of_birth DESC) AS rownum,
       group_no, student_id, surname, first_name, date_of_birth
       FROM students
       WHERE group_no IS NOT NULL and date_of_birth IS NOT NULL) AS s
WHERE rownum<=2;
```

CTE:

```
WITH CTE_table AS
      (SELECT ROW_NUMBER() OVER
       (PARTITION BY group_no
        ORDER BY date_of_birth DESC) AS rownum,
       group_no, student_id, surname, first_name, date_of_birth
      FROM students
      WHERE group_no IS NOT NULL and date_of_birth IS NOT NULL)
```

```
SELECT rownum, group_no, student_id, surname, first_name,
      date_of_birth
FROM CTE_table
WHERE rownum<=2;
```

22.07b

Korzystając z poprzedniego zapytania napisz funkcję o nazwie youngest_students, która zwróci dane o tylu najmłodszych studentach, ile wskazuje pierwszy parametr funkcji, z grupy, której nazwa zostanie podana jako drugi parametr.

Uruchom funkcję (wykorzystaj instrukcję SELECT) i sprawdź poprawność jej działania.

Wywołanie funkcji:

SELECT * FROM youngest_students(4, 'DMIe1011')

Zwraca 4 rekordy, w kolejności studentów o id: 1, 29, 30, 6

Wywołanie funkcji:

SELECT * FROM youngest_students(3, 'ZMIe2012')

Zwraca tabele pustą

Wywołanie funkcji:

SELECT * FROM youngest_students(5, 'DZZa3001')

Zwraca jeden rekord, student o id=19

```
CREATE or ALTER FUNCTION youngest_students
      (@number AS tinyint, @group AS varchar(20)) RETURNS TABLE AS
RETURN
WITH s AS
      (SELECT ROW_NUMBER() OVER
       (PARTITION BY group_no
        ORDER BY date_of_birth DESC) AS rownum,
       group_no, student_id, surname, first_name, date_of_birth
       FROM students
       WHERE group_no=@group and date_of_birth IS NOT NULL)
SELECT rownum, group_no, student_id, surname, first_name,
      date_of_birth
FROM s
WHERE rownum<=@number;
```

22.08a – recursive CTE

Module_id, module_name and no_of_hours wykładu o identyfikatorze 9 wraz z łańcuchem poprzedzających wykładów. Kolumnę zawierającą kolejny poziom nazwij distance.

4 rekordy, kolejno:

9 Econometrics 45 (distance 0)

8 Advanced Statistics 9 (1)

5 Statistics 30 (2)

4 Mathematics 15 (3)

```
WITH modulesCTE AS
(
SELECT module_id, module_name, preceding_module, no_of_hours,
	0 AS distance
FROM modules
WHERE module_id = 9
	UNION ALL
SELECT m.module_id, m.module_name, m.preceding_module, m.no_of_hours,
	e.distance + 1 AS distance
FROM modulesCTE e INNER JOIN modules m
	ON e.preceding_module = m.module_id
)
SELECT module_id, module_name, no_of_hours, distance
FROM modulesCTE;
```

22.08b

Na podstawie powyższego zapytania napisz funkcję o nazwie preceding_modules zwracającą module_id, module_no oraz no_of_hours wykładu o identyfikatorze podanym jako parametr funkcji wraz z łańcuchem poprzedzających wykładów.

Dla parametru 9 funkcja zwraca 4 rekordy (takie same jak powyższe zapytanie)

Dla parametru 5 funkcja zwraca 2 rekordy (kolejno wykłady o identyfikatorach 5 i 4)

Dla parametru 8 funkcja zwraca 3 rekordy (kolejno wykłady o identyfikatorach 8, 5, 4)

```
CREATE or ALTER FUNCTION preceding_modules
      (@number AS tinyint) RETURNS TABLE AS
RETURN
WITH modulesCTE AS
(
SELECT module_id, module_name, preceding_module, no_of_hours,
      0 AS distance
FROM modules
WHERE module_id = @number
      UNION ALL
SELECT m.module_id, m.module_name, m.preceding_module, m.no_of_hours,
      e.distance + 1 AS distance
FROM modulesCTE e INNER JOIN modules m
      ON e.preceding_module = m.module_id
)
SELECT module_id, module_name, no_of_hours, distance
FROM modulesCTE;
```

Elementy, które należy dodać do zapytania lub w nim zmienić zostały zaznaczone na **zielono**.

Skill 2.3/1. Writing grouped queries

I. Single Grouping set

23/1.01

Liczba studentów zarejestrowanych w bazie danych.

Zapytanie zwraca liczbę 35

```
SELECT count(*)
FROM students
```

23/1.02

Liczba studentów, którzy są przypisani do jakiejś grupy.

Zapytanie zwraca liczbę 28

```
SELECT count(group_no)
FROM students
```

23/1.03

Liczba studentów, którzy nie są przypisani do żadnej grupy.

Zapytanie zwraca liczbę 7

```
SELECT count(*)
FROM students
WHERE group_no IS NULL
```

23/1.04

Liczba grup, do których jest przypisany co najmniej jeden student.

Takich grup jest 12

```
SELECT count(distinct group_no)
FROM students
```

23/1.05

Nazwy grup, do których zapisany jest przynajmniej jeden student wraz z liczbą zapisanych studentów. Zapytanie ma zwrócić także informację, ilu studentów nie jest zapisanych do żadnej grupy. Kolumna zwracająca liczbę studentów ma mieć nazwę no_of_students. Dane posortowane rosnąco według liczby studentów.

13 rekordów

w pięciu grupach jest po jednym studencie, w czterech po dwóch,

w jednej czterech, w jednej pięciu, w jednej sześciu i w jednej siedmiu

```
SELECT group_no, COUNT(*) AS no_of_students
FROM students
GROUP BY group_no
ORDER BY no_of_students
```

23/1.06

Nazwy grup, do których zapisanych jest przynajmniej trzech studentów wraz z liczbą tych studentów. Kolumna zwracająca liczbę studentów ma mieć nazwę no_of_students. Dane posortowane rosnąco według liczby studentów.

4 rekordy

```
SELECT group_no, COUNT(*) AS no_of_students
FROM students
GROUP BY group_no
HAVING COUNT(*)>=3
ORDER BY no_of_students
```

23/1.07

Wszystkie możliwe oceny oraz ile razy każda z ocen została przyznana (kolumna ma mieć nazwę no_of_grades). Dane posortowane według ocen.

8 rekordów.

Ocena 2.0 została przyznane 13 razy.

Ocena 5.5 4 razy.

Ocena 6.0 nie została przyznana ani raz.

```
SELECT g.grade, COUNT(sg.student_id) AS no_of_grades
FROM grades g LEFT JOIN student_grades sg ON g.grade=sg.grade
GROUP BY g.grade
ORDER BY g.grade
```

23/1.08

Nazwy wszystkich katedr oraz ile godzin wykładów w sumie mają pracownicy zatrudnieni w tych katedrach. Kolumna zwracająca liczbę godzin ma mieć nazwę total_hours. Dane posortowane rosnąco według kolumny total_hours.

11 rekordów

Dla pierwszych sześciu total_hours jest NULL

Ostatni rekord: Department of Informatics, 117 godzin

```
SELECT d.department, SUM(no_of_hours) AS total_hours
FROM departments d LEFT JOIN lecturers l ON d.department=l.department
      LEFT JOIN modules m ON l.lecturer_id=m.lecturer_id
GROUP BY d.department
ORDER BY total_hours
```

23/1.09

Nazwisko każdego wykładowcy wraz z liczbą prowadzonych przez niego wykładów (zapytanie ma zwrócić także nazwiska wykładowców, którzy nie prowadzą żadnego wykładu). Kolumna zawierająca liczbę wykładów ma mieć nazwę no_of_modules. Dane posortowane malejąco według nazwiska.

28 rekordów.

Pierwszy: Wright, nie prowadzi żadnego wykładu.

Trzeci: White, prowadzi jeden wykład.

```
SELECT surname, no_of_modules=COUNT(module_id)
FROM (lecturers INNER JOIN employees ON lecturer_id=employee_id)
      LEFT JOIN modules ON modules.lecturer_id=lecturers.lecturer_id
GROUP BY surname, lecturers.lecturer_id
ORDER BY surname DESC
```

23/1.10

Nazwiska i imiona wykładowców prowadzących co najmniej dwa wykłady wraz z liczbą prowadzonych przez nich wykładów. Dane posortowane malejąco według liczby wykładów a następnie rosnąco według nazwiska.

6 rekordów.

Pierwszy: Harry Jones, 4 wykłady

Ostatni: Lily Taylor, 2 wykłady

```
SELECT surname, first_name, count(*) AS no_of_modules
FROM modules INNER JOIN employees ON lecturer_id=employee_id
GROUP BY employee_id, surname, first_name
HAVING count(*)>=2
ORDER BY no_of_modules DESC, surname
```

23/1.11a

Nazwiska i imiona wszystkich studentów o nazwisku Bowen, którzy otrzymali przynajmniej jedną ocenę wraz ze średnią ocen (każdego Bowena z osobna). Kolumna zwracająca średnią ma mieć nazwę avg_grade. Dane posortowane malejąco według nazwisk i malejąco według imion.

Dwa rekordy:
Harry Bowen, średnia 3.7
Charlie Bowen, średnia 2.0

```
SELECT surname, first_name, AVG(grade) AS avg_grade
FROM students s INNER JOIN student_grades sg
      ON s.student_id=sg.student_id
WHERE surname='Bowen'
GROUP BY s.student_id, surname, first_name
ORDER BY surname DESC, first_name DESC
```

23/1.11b

Na podstawie powyższego zapytania utwórz funkcję o nazwie avg_grade, która zwróci dane o studentach, których nazwisko zostanie podane jako parametr.

Pamiętaj, że w funkcji nie wolno używać klauzuli ORDER BY.

SELECT * FROM avg_grade('Fisher')

zwraca jeden rekord: Katie Fisher, średnia 4.1666

```
CREATE OR ALTER FUNCTION avg_grade (@surname varchar(100))
RETURNS TABLE AS RETURN
SELECT surname, first_name, AVG(grade) AS avg_grade
FROM students s INNER JOIN student_grades sg
      ON s.student_id=sg.student_id
WHERE surname=@surname
GROUP BY s.student_id, surname, first_name
~~ORDER BY surname DESC, first_name DESC~~
```

Zmiany, jakie należało zrobić w zapytaniu zaznaczono kolorem **zielonym**.

23/1.12a

Napisz funkcję o nazwie student_no, która zwróci liczbę studentów zapisanych na wykład o nazwie podanej jako parametr. Spraw, aby w parametrze usunięte zostały wszystkie wiodące i końcowe spacje.

SELECT * FROM student_no(' Databases ') – z ewentualnymi spacjami na początku i końcu

zwraca liczbę 2

SELECT * FROM student_no('Statistics')

zwraca liczbę 4

SELECT * FROM student_no('Macroeconomics')

zwraca liczbę 0

```
CREATE or ALTER FUNCTION student_no (@module AS varchar(100))
      RETURNS TABLE AS
RETURN
SELECT count(student_id) AS no_of_students
FROM modules m INNER JOIN students_modules sm
      ON m.module_id=sm.module_id
WHERE module_name=trim(@module)
```

23/1.12b

Zmodyfikuj poprzednią funkcję, aby zwracała nazwy wykładów wraz z liczbą studentów zapisanych na każdy z wykładów o nazwie zaczynającej się tekstem podanym jako parametr. Jeśli jako parametr zostanie podana wartość NULL, funkcja ma zwrócić tabelę pustą.

Wskazówka: w klauzuli WHERE użyj operatora +

SELECT * FROM student_no(NULL)

zwraca tabelę pustą

SELECT * FROM student_no('C')

Zwraca 5 rekordów.

Na wykład Computer network devices zapisanych jest 9 studentów.

Na Contemporary history 2 studentów.

```
CREATE or ALTER FUNCTION student_no (@module varchar(100))
RETURNS TABLE AS RETURN
SELECT module_name, count(student_id) AS no_of_students
FROM modules m INNER JOIN students_modules sm
      ON m.module_id=sm.module_id
WHERE module_name LIKE @module +'%'
GROUP BY module_name, m.module_id
```

23/1.12c

Zmodyfikuj poprzednią funkcję, aby dla parametru NULL funkcja zwracała dane o wszystkich wykładach.

Wskazówka: w klauzuli WHERE użyj funkcji CONCAT

SELECT * FROM student_no(NULL)

zwraca 26 rekordów

na cztery wykłady nie zapisał się żaden student

SELECT * FROM student_no('Macroeconomics')

zwraca 1 rekord: Marcroeconomics, 0 studentów

```
CREATE or ALTER FUNCTION student_no (@module varchar(100))
```

```
RETURNS TABLE AS RETURN

SELECT module_name, count(student_id) AS no_of_students

FROM modules m LEFT JOIN students_modules sm

      ON m.module_id=sm.module_id

WHERE module_name LIKE concat(@module,'%')

GROUP BY module_name, m.module_id
```

23/1.13a

Nazwiska i imiona wykładowców, którzy prowadzą co najmniej jeden wykład wraz ze średnią ocen jakie dali studentom (jeśli wykładowca nie dał do tej pory żadnej oceny, także ma się pojawić na liście). Kolumna zwracająca średnią ma mieć nazwę avg_grade. Dane posortowane malejąco według średniej.

11 rekordów. Pierwszy rekord: James Robinson, średnia 5.0.

Jeden wykładowca nie wystawił żadnej oceny.

```
SELECT surname, first_name, AVG(grade) AS avg_grade
FROM employees INNER JOIN modules m ON employee_id=lecturer_id
      LEFT JOIN student_grades sg ON m.module_id=sg.module_id
GROUP BY employee_id, surname, first_name
ORDER BY avg_grade DESC
```

23/1.13b

Nazwiska i imiona **wszystkich** wykładowców wraz ze średnią ocen jakie dali studentom (jeśli wykładowca nie dał do tej pory żadnej oceny, lub nie prowadzi wykładu, także ma się pojawić na liście). Kolumna zwracająca średnią ma mieć nazwę avg_grade. Dane posortowane malejąco według średniej.

28 rekordów.

W 18 przypadkach średnia wynosi NULL, co oznacza, że wykładowcy nie prowadzą żadnego wykładu lub prowadzą wykład, ale nie przyznali do tej pory żadnej oceny.

```
SELECT surname, first_name, AVG(grade) AS avg_grade
FROM employees INNER JOIN lecturers l
ON employee_id=lecturer_id
LEFT JOIN modules m ON l.lecturer_id=m.lecturer_id
LEFT JOIN student_grades sg ON m.module_id=sg.module_id
GROUP BY l.lecturer_id, surname, first_name
ORDER BY avg_grade DESC
```

23/1.14a

Nazwy wykładów oraz kwotę, jaką uczelnia musi przygotować na wypłaty pracownikom prowadzącym wykłady ze Statistics i Economics (osobno). Jeśli jest wiele wykładów o nazwie Statistics lub Economics, suma dla nich ma być obliczona łącznie. Zapytanie ma więc zwrócić dwa rekordy (jeden dla wykładów ze Statistics, drugi dla Economics).

Kwotę za jeden wykład należy obliczyć jako iloczyn stawki godzinowej prowadzącego wykładowcy oraz liczby godzin przeznaczonych na wykład.

Zapytanie zwraca jeden rekord: Economics 1200.00

```
SELECT module_name, SUM(overtime_rate*no_of_hours) sum_of_money
FROM (acad_positions ap INNER JOIN lecturers l
      ON ap.acad_position=l.acad_position)
      INNER JOIN modules m ON m.lecturer_id=l.lecturer_id
WHERE module_name in ('Statistics','Economics')
GROUP BY module_name
```

Odpowiedz na pytanie, dlaczego zapytanie nie zwróciło danych o wykładzie Statistics.

Odpowiedź:
W bazie danych jest jeden taki wykład (o id=5) i ma przypisaną liczbę godzin oraz wykładowcę (id=30). Jednak wykładowca o id=30 nie ma przypisanego stopnia naukowego a od stopnia zależy stawka za godzinę, której w takiej sytuacji nie da się wyliczyć.

23/1.14b

Zapytanie zwracające jedną liczbę: kwotę, jaką uczelnia musi przygotować na wypłaty z tytułu prowadzonych wykładów. Kwotę za jeden wykład należy wyliczyć jako iloczyn stawki godzinowej prowadzącego wykładowcy oraz liczby godzin przeznaczonych na ten wykład. Pamiętaj, aby nazwać kolumnę zwracającą szukaną kwotę.

Odpowiedz na pytanie: czy możliwe jest wyliczenie pełnej kwoty należności z tytułu przeprowadzonych wykładów? Uzasadnij odpowiedź.

Wynikiem jest kwota 20265.00

```
SELECT SUM(overtime_rate*no_of_hours) AS sum_salary
FROM (acad_positions ap INNER JOIN lecturers l
        ON ap.acad_position=l.acad_position)
             INNER JOIN modules m ON m.lecturer_id=l.lecturer_id
```

Nie jesteśmy w stanie ustalić pełnej kwoty, gdyż może być taka sytuacja, że wykład nie ma wpisanego lecturer_id, czyli nie wiadomo, kto wykład prowadzi. Może być też tak, że wykład ma wpisaną wartość lecturer_id ale w tabeli lecturers wykładowca prowadzący ten wykład nie ma wpisanego acad_position a od acad_position zależy stawka za godzinę nadliczbową.

23/1.14c

Kwotę, jaką uczelnia musi przygotować na wypłaty z tytułu prowadzenia wykładów, którym nie jest przypisany żaden wykładowca, przy założeniu, że za godzinę takiego wykładu należy zapłacić średnią z pola overtime_rate w tabeli acad_positions.

Wskazówka: wykorzystaj CTE. Wynik CTE, którym będzie obliczona średnia, połącz iloczynem kartezjańskim z tabelą modules.

7649.99

```
WITH a AS
      (SELECT avg(overtime_rate) avg_or
       FROM acad_positions)
SELECT sum(no_of_hours*a.avg_or)
FROM modules CROSS JOIN a
WHERE lecturer_id IS NULL
```

Aby sprawdzić otrzymany wynik napisz następującą instrukcję:
SELECT

(zapytanie obliczające sumę godzin wykładów, które w polu lecturer_id mają NULL)

*

(zapytanie obliczające średnią z wartości w polu overtime_rate w tabeli acad_positions)

```
SELECT
(SELECT sum(no_of_hours) FROM modules WHERE lecturer_id IS NULL)
 *
(SELECT avg(overtime_rate) FROM acad_positions)
```

23/1.14d

<u>Maksymalną</u> kwotę, jaką uczelnia musi przygotować na wypłaty z tytułu prowadzenia wykładów, dla których nie można tej kwoty obliczyć. Są to wykłady, którym nie jest przypisany żaden wykładowca lub wykładowca jest przypisany, ale nieznany jest jego stopień/tytuł naukowy.

13200

```
with maksimum as
(select max(overtime_rate) avg_or
 from acad_positions)
select sum(no_of_hours*avg_or)
from (modules m left join lecturers l
       on  m.lecturer_id=l.lecturer_id) cross join maksimum
where m.lecturer_id is null or acad_position is null
```

lub

```
SELECT
(SELECT sum(no_of_hours)
 FROM modules m left join lecturers l ON m.lecturer_id=l.lecturer_id
 WHERE l.lecturer_id IS NULL or acad_position IS NULL)
 *
(SELECT max(overtime_rate) FROM acad_positions)
```

23/1.15

Nazwiska i imiona wykładowców wraz z sumaryczną liczbą godzin wykładów prowadzonych przez każdego z nich z osobna ale tylko w przypadku, gdy suma godzin prowadzonych wykładów jest większa od 30. Kolumna zwracająca liczbę godzin ma mieć nazwę no_of_hours. Dane posortowane malejąco według liczby godzin.

5 rekordów.

Pierwszy: Jones Harry, 72 godziny.

Ostatni: Katie Davies 55 godzin.

```
SELECT surname, first_name, SUM(no_of_hours) AS no_of_hours
```

```
FROM (employees INNER JOIN lecturers l ON employee_id=lecturer_id)
      INNER JOIN modules m ON m.lecturer_id=l.lecturer_id
GROUP BY surname, first_name, m.lecturer_id
HAVING SUM(no_of_hours)>30
ORDER BY no_of_hours DESC
```

23/1.16

Nazwy wszystkich grup oraz liczbę studentów zapisanych do każdej grupy (kolumna ma mieć nazwę no_of_students). Dane posortowane rosnąco według liczby studentów a następnie numeru grupy.

23 rekordy.

Do 11 grup nie został zapisany żaden student.

Ostatni rekord: grupa DMIe1011, 6 studentów

```
SELECT g.group_no, COUNT(student_id) AS no_of_students
FROM groups g LEFT JOIN students s ON g.group_no=s.group_no
GROUP BY g.group_no
ORDER BY no_of_students, g.group_no
```

23/1.17

Nazwy wszystkich wykładów, których nazwa zaczyna się literą A oraz średnią ocen ze wszystkich tych wykładów osobno (jeśli jest wiele takich wykładów, to średnia ma być obliczona dla każdego z nich oddzielnie). Jeśli z danego wykładu nie ma żadnej oceny, także powinien on pojawić się na liście. Kolumna ma mieć nazwę average.

3 rekordy:

Advanced databases NULL

Advanced statistics 4.25

Ancient history 4.25

```
SELECT module_name, AVG(grade) AS average
FROM modules m LEFT JOIN student_grades sg
	ON m.module_id=sg.module_id
WHERE module_name LIKE 'A%'
GROUP BY m.module_id, module_name
```

23/1.18

Nazwy grup, do których jest zapisanych co najmniej dwóch studentów, liczba studentów zapisanych do tych grup (kolumna ma mieć nazwę no_of_students) oraz średnie ocen dla każdej grupy (kolumna ma mieć nazwę average_grade). Dane posortowane malejąco według średniej.

8 rekordów.

Pierwszy: ZMIe2012, liczba studentów 5, średnia 6.6

Ostatni: DMIe1014, liczba studentów 2, średnia 3.25

```
SELECT group_no, COUNT(s.student_id) AS no_of_students,
	AVG(grade) AS average_grade
FROM students s INNER JOIN student_grades sg
	ON s.student_id=sg.student_id
GROUP BY group_no
HAVING COUNT(s.student_id) >= 2
ORDER BY average_grade DESC
```

23/1.19a

Nazwy tych katedr (department), w których pracuje co najmniej 2 doktorów (doctor) wraz z liczbą doktorów pracujących w tych katedrach (ta ostatnia kolumna ma mieć nazwę no_of_dostors). Dane posortowane malejąco według liczby doktorów i rosnąco według nazw katedr.

3 rekordy.

Department of Informatics – trzech doktorów.

Department of History i Departemt of Mathematics po dwóch doktorów.

```
SELECT department, count(*) AS no_of_doctors
FROM lecturers
WHERE acad_position='doctor'
GROUP BY department
HAVING count(*)>=2
ORDER BY no_of_doctors DESC, department
```

23/1.19b

Na podstawie powyższego zapytania napisz funkcję o nazwie academics, która będzie zwracać to, co zapytanie, ale na podstawie wartości dwóch parametrów: nazwy stopnia naukowego oraz minimalnej liczby pracowników mających ten stopień.

SELECT * FROM academics('associate professor', 1)

zwraca 4 rekordy (w Department of Statistics jest 3 associate professors)

SELECT * FROM academics('doctor', 2)

zwraca 3 rekordy

```
CREATE OR ALTER FUNCTION academics (@ap varchar(100), @num int)
      RETURNS TABLE AS RETURN
SELECT department, count(*) AS no_of_doctors
FROM lecturers
WHERE acad_position=@ap
GROUP BY department
```

```
HAVING count(*)>=@num
ORDER BY no_of_doctors DESC, department
```

Zmiany, jakie należało zrobić w zapytaniu zaznaczono kolorem **zielonym**.

23/1.20

Nazwiska, imiona i nazwy katedr trzech wykładowców, którzy prowadzą największą liczbę wykładów wraz z liczbą wykładów, które prowadzi każdy z nich. Kolumna zwracająca liczbę wykładów ma mieć nazwę no_of_modules. Jeśli trzecia osoba na liście prowadzi tyle samo wykładów co kolejne, te kolejne także mają się pojawić w tabeli wynikowej zapytania.

6 rekordów.

Harry Jones prowadzi 4 wykłady,

Thomas Evans 3

oraz czterech wykładowców po 2

```
SELECT top(3) WITH TIES surname, first_name, lecturers.department,
      COUNT(m.module_id) AS no_of_modules
FROM (employees INNER JOIN lecturers ON employee_id=lecturer_id)
      LEFT JOIN modules m ON m.lecturer_id=lecturers.lecturer_id
GROUP BY lecturers.lecturer_id, surname, first_name,
      lecturers.department
ORDER BY count(m.module_id) DESC
```

23/1.21

Identyfikatory, nazwy wykładów oraz nazwy katedr odpowiedzialnych za prowadzenie wykładów, dla których nie można ustalić kwoty, jaką trzeba zapłacić za ich przeprowadzenie wraz z nazwiskiem i imieniem dowolnego spośród pracowników tych katedr.

Dane posortowane według module_id.

UWAGA: najpierw należy utworzyć pole wyliczane, które połączy nazwisko i imię w jedno pole. Użyj funkcji CONCAT lub operatora +. Pamiętaj, aby między nazwiskiem i imieniem wstawić spację.

6 rekordów. Wśród module_id są tylko identyfikatory 4, 7, 13, 15, 20, 22

```
SELECT module_id, module_name, m.department,
       min(concat(surname,' ',first_name)) as lecturer_name
FROM modules m INNER JOIN lecturers l
       ON m.department=l.department
            INNER JOIN employees ON l.lecturer_id=employee_id
WHERE m.lecturer_id IS NULL
GROUP BY module_id, module_name, m.department
ORDER BY module_id
```

II. Multiple Grouping Set

23/1.22a

Nazwy katedr, w których pracuje co najmniej jeden wykładowca, tytuły naukowe występujące w ramach każdej katedry oraz informację o liczbie prowadzonych wykładów w ramach katedr i w katedrach przez każdą z grup wykładowców (według tytułu naukowego). W zapytaniu należy pominąć wykładowców, którzy nie prowadzą żadnego wykładu.

Użyj grupowania ROLLUP.

17 rekordów.

```
SELECT l.department, acad_position,
      count(module_id) AS no_of_modules
FROM lecturers l INNER JOIN modules m ON l.lecturer_id=m.lecturer_id
GROUP BY ROLLUP(l.department, acad_position)
```

Zapytanie zwróciło 4 rekordy dla Department of History. Zinterpretuj informację znajdującą się w każdym rekordzie.

Pierwszy z tych rekordów (Department of History, NULL, 1) informuje, że jeden z wykładów prowadzi wykładowca, który jest pracownikiem Department of History ale nie ma przypisanego tytułu naukowego.

Drugi i trzeci rekord informują, że z Department of History cztery wykłady prowadzą doktorzy a jeden full profesor.

Ostatni rekord (Department of History, NULL, 6) informuje, że pracownicy tej katedry prowadzą w sumie 6 wykładów.

Zauważ, że w rekordach dotyczących Department of History dwie grupy mają w pierwszych dwóch kolumnach Department of History oraz NULL. Zinterpretuj ten fakt.

Nie wiemy, czy grupowanie jest tylko względem department, czy względem department oraz acad_position.

23/1.22b

Używając funkcji GROUPING zmodyfikuj zapytanie tak, aby zwróciło informację, względem których pól jest wykonywane grupowanie.

17 rekordów.

Różnice zaznaczono kolorem **zielonym**.

```
SELECT grouping(l.department) AS g_department,
       grouping(acad_position) AS g_acad_position,
      l.department, acad_position, count(module_id) AS no_of_modules
FROM lecturers l INNER JOIN modules m ON l.lecturer_id=m.lecturer_id
GROUP BY ROLLUP(l.department, acad_position)
```

W niektórych polach będących wynikiem działania funkcji GROUPING pojawiły się liczby 1, w niektórych 0 – zinterpretuj te wyniki.

W ostatnim rekordzie, w polach będących wynikiem działania funkcji GROUPING są dwie jedynki – zinterpretuj informację znajdującą się w tym rekordzie.

23/1.22c

Zmodyfikuj zapytanie wyświetlając informację o sposobie grupowania przy pomocy funkcji GROUPING_ID.

17 rekordów.

W pięciu rekordach w polu będącym wynikiem działania funkcji GROUPING_ID jest liczba 1, w jednym rekordzie jest liczba 3.

```
SELECT grouping_id(l.department, acad_position) AS grouping,
      l.department, acad_position, count(module_id) AS no_of_modules
FROM lecturers l INNER JOIN modules m ON l.lecturer_id=m.lecturer_id
GROUP BY ROLLUP(l.department, acad_position)
```

W ostatnim rekordzie, w polu będącym wynikiem działania funkcji GROUPING_ID znajduje się liczba 3. Zinterpretuj tę informację.

Napisz zapytanie zwracające liczbę rekordów znajdujących się w tabeli modules. Odpowiedz na pytanie, dlaczego liczba rekordów w tabeli modules (26) jest różna od liczby wykładów zwróconych w ostatnim rekordzie poprzedniego zapytania (20).

Poprzednie zapytanie wybierało tylko dane o wykładach, którym jest przypisany wykładowca.

23/1.23a

Numery grup, do których zapisany jest co najmniej jeden student, nazwy **wszystkich wykładów, na które studenci są zapisani** oraz informację o liczbie studentów w ramach każdej grupy oddzielnie zapisanych na poszczególne wykłady.

96 rerkordów.

Pierwszy NULL NULL 0

Przedostatni: ZZIe2013 NULL 2

Ostatni: NULL NULL 94

```
SELECT group_no, module_name, count(sm.student_id) no_of_students
FROM students s LEFT JOIN students_modules sm
      ON s.student_id=sm.student_id
```

```
LEFT JOIN modules m ON sm.module_id=m.module_id

GROUP BY ROLLUP(group_no, module_name)
```

W pierwszych 16 rekordach w polu group_no znajduje się wartość NULL. Zinterpretuj tę informację.

Na wykłady zapisani są studenci, którzy nie są przypisani do żadnej grupy.

29 i 30 rekord są identyczne (DMIe1013 NULL 0) – zinterpretuj tę informację.

W grupie DMIE1013 jest co najmniej jeden student i żaden z nich nie jest zapisany na wykład.

W przypadku niektórych grup (np. DMIe1011 – rekord 17) na początku listy występuje numer grupy oraz, w polu module_name wartość NULL a w przypadku niektórych grup (np. DMZa3012) rekord z numerem grupy i wartością NULL na początku nie występuje – zinterpretuj jeden i drugi przypadek.

W pierwszym przypadku w grupie są studenci, którzy nie są zapisani na żaden wykład, w drugim przypadku takich studentów nie ma – wszyscy studenci z grupy są zapisani na co najmniej jeden wykład.

23/1.23b

Zmodyfikuj poprzednie zapytanie, aby zwracało wynik działania funkcji GROUPING_ID (kolumnę nazwij grp_id). Zinterpretuj znaczenie liczb 0, 1 i 3 znajdujących się w kolumnie zawierającej wynik działania tej funkcji.

Różnice zaznaczono kolorem **zielonym**.

```
SELECT grouping_id(group_no, module_name) AS grp_id, group_no,
      module_name, count(sm.student_id) no_of_students
FROM students s LEFT JOIN students_modules sm
      ON s.student_id=sm.student_id
            LEFT JOIN modules m ON sm.module_id=m.module_id
GROUP BY ROLLUP(group_no, module_name)
```

23/1.23c

Zmodyfikuj poprzednie zapytanie, aby zwracało jedynie rekordy, które w polu grp_id mają liczbę 1.

13 rekordów.

Różnice zaznaczono kolorem **zielonym**.

```
SELECT grouping_id(group_no, module_name) AS grp_id, group_no,
      module_name, count(sm.student_id) no_of_students
FROM students s LEFT JOIN students_modules sm
      ON s.student_id=sm.student_id
            LEFT JOIN modules m ON sm.module_id=m.module_id
GROUP BY ROLLUP(group_no, module_name)
HAVING grouping_id(group_no, module_name)=1
```

Napisz zapytanie zwracające informację o liczbie RÓŻNYCH grup w tabeli students.

```
SELECT count(DISTINCT group_no) FROM students
```

Zapytanie zwróciło liczbę 12. Zapytanie zwracające rekordy, które w polu grp_id mają liczbę 1 zwróciło 13 rekordów. Wyjaśnij różnicę.

Zapytanie zwracające informację o liczbie grup nie uwzględniło braku przypisania studentów do grupy (NULL w polu group_no).

23/1.24

Nazwy stopni/tytułów naukowych, nazwy katedr oraz informację, ile godzin mają poszczególne grupy wykładowców (posiadających taki sam stopień/tytuł) w ramach każdej katedry. Użyj funkcji GROUPING_ID lub GROUPING.

17 rekordów.

associate professors mają w sumie 115 godzin zajęć

doctors: 87

full professors: 68

masters: 129

```
SELECT grouping_id(acad_position, l.department) AS grp_id,
      acad_position, l.department, sum(no_of_hours) AS sum_of_hours
FROM lecturers l INNER JOIN modules m ON l.lecturer_id=m.lecturer_id
GROUP BY ROLLUP(acad_position, l.department)
```

W pierwszych dwóch rekordach w polu acad_position jest NULL a w polu department w pierwszym rekordzie jest Department of History a w drugim NULL. Zinterpretuj tę informację.

Pierwszy rekord wskazuje, że w Department of History są pracownicy, którzy prowadzą zajęcia (w sumie 30 godzin) ale nie mają przypisanego stopnia/tytułu naukowego.