

## Bazy Danych 2 - Lab 1/2

### Spis Treści

<b>Wstęp Teoretyczny.....</b>	<b>2</b>
<b>Zadania.....</b>	<b>7</b>
<b>Basic Select .....</b>	<b>7</b>
Zadanie 1.....	7
Zadanie 2.....	8
Zadanie 3.....	8
Zadanie 4.....	9
Zadanie 5.....	9
Zadanie 6.....	10
Zadanie 7.....	11
Zadanie 8.....	11
Zadanie 9.....	12
Zadanie 10.....	13
Zadanie 11.....	14
Zadanie 12.....	15
Zadanie 13.....	16
Zadanie 14.....	17
Zadanie 15.....	18
Zadanie 16.....	19
Zadanie 17.....	20
Zadanie 18.....	21
Zadanie 19.....	22
Zadanie 20.....	23
<b>Aggregation .....</b>	<b>24</b>
Zadanie 21.....	24
Zadanie 22.....	24
Zadanie 23.....	25
Zadanie 24.....	25
Zadanie 25.....	26
Zadanie 26.....	26
Zadanie 27.....	27
Zadanie 28.....	28
Zadanie 29.....	29
Zadanie 30.....	30
Zadanie 31.....	31
Zadanie 32.....	32
Zadanie 33.....	33
Zadanie 34.....	34
Zadanie 35.....	35
Zadanie 36.....	36
Zadanie 37.....	37
<b>Basic Join .....</b>	<b>38</b>
Zadanie 38.....	38
Zadanie 39.....	39
Zadanie 40.....	40
Zadanie 41.....	41
Zadanie 42.....	42
Zadanie 43.....	43
Zadanie 44.....	44
Zadanie 45.....	45

# Wstęp Teoretyczny

## Typy baz danych

Bazy danych można podzielić na kilka kategorii:

1. **Relacyjne bazy danych (RDBMS – Relational Database Management Systems)** – dane są przechowywane w tabelach i powiązane ze sobą relacjami (np. MySQL, PostgreSQL, Oracle, SQL Server).
2. **NoSQL (Not Only SQL)** – nie stosują struktury tabelarycznej i są przeznaczone dla dużych, dynamicznych danych (np. MongoDB, Redis, Cassandra).
3. **Bazy obiektowe** – przechowują dane w postaci obiektów, jak w programowaniu obiektowym (np. db4o).
4. **Bazy grafowe** – przechowują dane w postaci grafów, używane w analizach relacji (np. Neo4j).
5. **Bazy temporalne** – pozwalają śledzić zmiany wartości w czasie.

## Podstawowe pojęcia w bazach danych

- **Encja** – obiekt przechowywany w bazie (np. użytkownik, produkt, zamówienie).
- **Atrybut** – właściwość encji (np. nazwa, wiek, cena).
- **Rekord** – jeden wiersz w tabeli.
- **Klucz główny (Primary Key)** – unikalny identyfikator rekordu w tabeli.
- **Klucz obcy (Foreign Key)** – odniesienie do innej tabeli (relacja między tabelami).
- **Normalizacja** – proces organizacji danych w bazie w celu minimalizacji redundancji i zwiększenia spójności.
- **Transakcja** – zbiór operacji na bazie danych, które muszą zostać wykonane w całości (ACID).

## 1. Wprowadzenie do SQL

SQL (Structured Query Language) to język używany do zarządzania danymi w relacyjnych bazach danych. Jego podstawowe funkcjonalności obejmują:

- **Tworzenie i modyfikowanie struktur bazy danych** (DDL – Data Definition Language)
- **Wstawianie, aktualizowanie i usuwanie danych** (DML – Data Manipulation Language)
- **Zapytania o informacje zawarte w bazie** (DQL – Data Query Language)

SQL pozwala na wykonywanie skomplikowanych operacji na danych, takich jak:

- **Agregacje i analizy danych**
- **Filtrowanie i sortowanie wyników**
- **Łączenie tabel (JOIN)**
- **Podzapytania i funkcje analityczne**

## 2. Operacje na Danych w SQL

### 2.1 Podstawowe zapytania SELECT

Podstawowe zapytanie w SQL wygląda następująco:

```
SELECT kolumny  
FROM tabela  
WHERE warunek;
```

Na przykład, aby pobrać wszystkie miasta z populacją większą niż 100000:

```
SELECT NAME, POPULATION  
FROM CITY  
WHERE POPULATION > 100000;
```

### 2.2 Sortowanie wyników – ORDER BY

Sortowanie wyników można osiągnąć przy użyciu **ORDER BY**:

```
SELECT NAME, POPULATION  
FROM CITY  
ORDER BY POPULATION DESC;
```

- **ASC (rosnąco, domyślnie)**
- **DESC (malejąco)**

## 2.3 Filtrowanie danych – WHERE

Klauzula **WHERE** umożliwia filtrowanie wyników na podstawie warunków logicznych:

```
SELECT * FROM CITY  
WHERE COUNTRYCODE = 'USA' AND POPULATION > 500000;
```

## 2.4 Grupowanie wyników – GROUP BY & HAVING

**GROUP BY** pozwala na grupowanie danych według określonej kolumny:

```
SELECT COUNTRYCODE, COUNT(*) AS TotalCities  
FROM CITY  
GROUP BY COUNTRYCODE;
```

Aby przefiltrować grupy, używamy **HAVING**:

```
SELECT COUNTRYCODE, COUNT(*) AS TotalCities  
FROM CITY  
GROUP BY COUNTRYCODE  
HAVING COUNT(*) > 5;
```

## 3. Agregacje i Funkcje Analityczne

SQL oferuje szereg funkcji agregujących do analizy danych:

Funkcja	Opis
<b>COUNT()</b>	Liczyc liczbę wierszy
<b>SUM()</b>	Sumuje wartości w kolumnie
<b>AVG()</b>	Oblicza średnią
<b>MIN() / MAX()</b>	Znajduje wartość minimalną/maksymalną

**Przykład:** Znalezienie liczby miast w USA o populacji > 100000:

```
SELECT COUNT(*) AS TotalCities  
FROM CITY  
WHERE COUNTRYCODE = 'USA' AND POPULATION > 100000;
```

## 4. Łączenie Tabel – JOIN

Relacyjne bazy danych składają się z wielu tabel połączonych relacjami. Do pobierania danych z kilku tabel używamy **JOIN**.

### 4.1 Rodzaje JOIN-ów

Rodzaj	Opis
<b>INNER JOIN</b>	Pobiera tylko pasujące rekordy z obu tabel
<b>LEFT JOIN</b>	Pobiera wszystkie rekordy z lewej tabeli i pasujące z prawej
<b>RIGHT JOIN</b>	Pobiera wszystkie rekordy z prawej tabeli i pasujące z lewej
<b>FULL JOIN</b>	Pobiera wszystkie rekordy z obu tabel

#### Przykład INNER JOIN:

```
SELECT CITY.NAME, COUNTRY.CONTINENT  
FROM CITY  
JOIN COUNTRY ON CITY.COUNTRYCODE = COUNTRY.CODE;
```

## 5. Zaawansowane Operacje SQL

### 5.1 Podzapytania (Subqueries)

Podzapytania pozwalają na wykonywanie zapytań wewnętrz zapytań:

```
SELECT NAME FROM CITY  
WHERE POPULATION = (SELECT MAX(POPULATION) FROM CITY);
```

W tym przykładzie pobieramy nazwę miasta o największej populacji.

### 5.2 Funkcje Okienkowe (Window Functions)

Funkcje okienkowe pozwalają na operacje agregujące w ramach podziału danych.

```
SELECT hacker_id, name, SUM(score) OVER(PARTITION BY hacker_id) AS total_score  
FROM SUBMISSIONS;
```

**PARTITION BY** pozwala grupować dane bez utraty szczegółowości.

## **6. Sortowanie i Grupowanie Specjalne**

### **6.1 Zaokrąglanie wyników – ROUND, FLOOR, CEIL**

SQL pozwala na manipulację wartościami liczbowymi:

- **ROUND(liczba, x)** – Zaokrągl do x miejsc po przecinku
- **FLOOR(liczba)** – Zaokrągl w dół
- **CEIL(liczba)** – Zaokrągl w góre

```
SELECT ROUND(AVG(POPULATION), 2) FROM CITY;
```

### **6.2 Ranking wyników – ROW\_NUMBER, RANK, DENSE\_RANK**

Czasami chcemy przypisać numer do każdego wiersza:

```
SELECT hacker_id, name,  
       RANK() OVER(ORDER BY SUM(score) DESC) AS rank  
  FROM SUBMISSIONS  
 GROUP BY hacker_id, name;
```

**RANK()** przydziela ranking, pomijając wartości przy remisie.

## **7. Znajdowanie Median i Innych Wartości Statystycznych**

SQL pozwala na obliczanie mediany:

```
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY LAT_N)  
  FROM STATION;
```

- **PERCENTILE\_CONT(0.5)** – Znajduje medianę

# Zadania

Aby rozpocząć ćwiczenie, należy:

1. Wejść na stronę [HackerRank](#) i założyć konto.
2. Przejść do sekcji zadań SQL, dostępnej pod adresem:  
<https://www.hackerrank.com/domains/sql>

## Basic Select

### Zadanie 1

#### Treść zadania:

Należy zapytać o wszystkie kolumny dla amerykańskich miast w tabeli **CITY**, które mają populację większą niż **100000**.

Dla USA kod kraju (**CountryCode**) to “**USA**”

Link do zadania: [Revising the Select Query 1](#)

Struktura Tabeli CITY:

CITY	
Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

## Zadanie 2

Link do zadania: [Revising the Select Query 2](#)

### Treść zadania:

Należy zapytać o kolumnę **NAME** dla amerykańskich miast w tabeli **CITY**, które mają populację większą niż **120000**.

Dla USA kod kraju (**CountryCode**) to “**USA**”.

Struktura Tabeli CITY:

CITY	
Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

## Zadanie 3

Link do zadania: <https://www.hackerrank.com/challenges/select-all-sql/problem>

### Treść zadania:

Należy zapytać o wszystkie kolumny (**attributes**) dla każdego wiersza w tabeli **CITY**.

Query all columns (attributes) for every row in the CITY table.  
The CITY table is described as follows:

CITY	
Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

## Zadanie 4

Link do zadania: <https://www.hackerrank.com/challenges/select-by-id/problem>

### Treść zadania:

Należy zapytać o wszystkie kolumny dla miasta w tabeli **CITY**, które ma **ID = 1661**.

Query all columns for a city in <b>CITY</b> with the ID 1661.													
The <b>CITY</b> table is described as follows:													
<b>CITY</b>													
<table border="1" style="width: 100%;"><thead><tr><th>Field</th><th>Type</th></tr></thead><tbody><tr><td>ID</td><td>NUMBER</td></tr><tr><td>NAME</td><td>VARCHAR2(17)</td></tr><tr><td>COUNTRYCODE</td><td>VARCHAR2(3)</td></tr><tr><td>DISTRICT</td><td>VARCHAR2(20)</td></tr><tr><td>POPULATION</td><td>NUMBER</td></tr></tbody></table>		Field	Type	ID	NUMBER	NAME	VARCHAR2(17)	COUNTRYCODE	VARCHAR2(3)	DISTRICT	VARCHAR2(20)	POPULATION	NUMBER
Field	Type												
ID	NUMBER												
NAME	VARCHAR2(17)												
COUNTRYCODE	VARCHAR2(3)												
DISTRICT	VARCHAR2(20)												
POPULATION	NUMBER												

## Zadanie 5

Link do zadania: <https://www.hackerrank.com/challenges/japanese-cities-attributes/problem>

### Treść zadania:

Należy zapytać o wszystkie atrybuty (**kolumny**) dla każdego miasta w Japonii w tabeli **CITY**.

Kod kraju (**COUNTRYCODE**) dla Japonii to “**JPN**”.

Query all attributes of every Japanese city in the <b>CITY</b> table. The <b>COUNTRYCODE</b> for Japan is <b>JPN</b> .													
The <b>CITY</b> table is described as follows:													
<b>CITY</b>													
<table border="1" style="width: 100%;"><thead><tr><th>Field</th><th>Type</th></tr></thead><tbody><tr><td>ID</td><td>NUMBER</td></tr><tr><td>NAME</td><td>VARCHAR2(17)</td></tr><tr><td>COUNTRYCODE</td><td>VARCHAR2(3)</td></tr><tr><td>DISTRICT</td><td>VARCHAR2(20)</td></tr><tr><td>POPULATION</td><td>NUMBER</td></tr></tbody></table>		Field	Type	ID	NUMBER	NAME	VARCHAR2(17)	COUNTRYCODE	VARCHAR2(3)	DISTRICT	VARCHAR2(20)	POPULATION	NUMBER
Field	Type												
ID	NUMBER												
NAME	VARCHAR2(17)												
COUNTRYCODE	VARCHAR2(3)												
DISTRICT	VARCHAR2(20)												
POPULATION	NUMBER												

## Zadanie 6

Link do zadania: <https://www.hackerrank.com/challenges/japanese-cities-name/problem>

### Treść zadania:

Należy zapytać o nazwy wszystkich japońskich miast w tabeli **CITY**.

Kod kraju (**COUNTRYCODE**) dla Japonii to “**JPN**”.

Query the names of all the Japanese cities in the **CITY** table. The **COUNTRYCODE** for Japan is **JPN**.

The **CITY** table is described as follows:

**CITY**

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

## Zadanie 7

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-1/problem>

### Treść zadania:

Należy zapytać o listę **CITY** i **STATE** z tabeli **STATION**.

Query a list of <b>CITY</b> and <b>STATE</b> from the <b>STATION</b> table.													
The <b>STATION</b> table is described as follows:													
<b>STATION</b>													
<table border="1"><thead><tr><th>Field</th><th>Type</th></tr></thead><tbody><tr><td>ID</td><td>NUMBER</td></tr><tr><td>CITY</td><td>VARCHAR2(21)</td></tr><tr><td>STATE</td><td>VARCHAR2(2)</td></tr><tr><td>LAT_N</td><td>NUMBER</td></tr><tr><td>LONG_W</td><td>NUMBER</td></tr></tbody></table>		Field	Type	ID	NUMBER	CITY	VARCHAR2(21)	STATE	VARCHAR2(2)	LAT_N	NUMBER	LONG_W	NUMBER
Field	Type												
ID	NUMBER												
CITY	VARCHAR2(21)												
STATE	VARCHAR2(2)												
LAT_N	NUMBER												
LONG_W	NUMBER												
where <b>LAT_N</b> is the northern latitude and <b>LONG_W</b> is the western longitude.													

## Zadanie 8

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-3/problem>

### Treść zadania:

Należy zapytać o listę nazw miast (**CITY**) z tabeli **STATION**, które mają **parzyste ID**.

- Wyniki mogą być zwrócone w dowolnej kolejności.
- W odpowiedzi nie mogą pojawiać się duplikaty.

STATION													
<table border="1"><thead><tr><th>Field</th><th>Type</th></tr></thead><tbody><tr><td>ID</td><td>NUMBER</td></tr><tr><td>CITY</td><td>VARCHAR2(21)</td></tr><tr><td>STATE</td><td>VARCHAR2(2)</td></tr><tr><td>LAT_N</td><td>NUMBER</td></tr><tr><td>LONG_W</td><td>NUMBER</td></tr></tbody></table>		Field	Type	ID	NUMBER	CITY	VARCHAR2(21)	STATE	VARCHAR2(2)	LAT_N	NUMBER	LONG_W	NUMBER
Field	Type												
ID	NUMBER												
CITY	VARCHAR2(21)												
STATE	VARCHAR2(2)												
LAT_N	NUMBER												
LONG_W	NUMBER												
ID	NUMBER												
CITY	VARCHAR2(21)												
STATE	VARCHAR2(2)												
LAT_N	NUMBER												
LONG_W	NUMBER												

## Zadanie 9

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-4/problem>

### Treść zadania:

Należy obliczyć różnicę między całkowitą liczbą wpisów **CITY** w tabeli **STATION** a liczbą unikalnych wartości **CITY** w tej tabeli.

### Przykład:

Jeśli w tabeli znajdują się trzy wpisy z wartościami **CITY**: "New York", "New York", "Bengaluru", to istnieją **dwa różne miasta**: "New York" i "Bengaluru".

Wynik zapytania to:

total number of records - number of unique city names =  $3 - 2 = 1$

Find the difference between the total number of **CITY** entries in the table and the number of distinct **CITY** entries in the table.

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where **LAT\_N** is the northern latitude and **LONG\_W** is the western longitude.

For example, if there are three records in the table with **CITY** values 'New York', 'New York', 'Bengalaru', there are 2 different city names: 'New York' and 'Bengalaru'. The query returns 1, because total number of records – number of unique city names =  $3 - 2 = 1$ .

## Zadanie 10

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-5/problem>

### Treść zadania:

Należy znaleźć **dwa miasta** w tabeli **STATION**:

1. Miasto o **najkrótszej nazwie**.

2. Miasto o **najdłuższej nazwie**.

Dodatkowo, jeśli jest więcej niż jedno miasto z najkrótszą lub najdłuższą nazwą, należy wybrać to, które pojawia się **pierwsze w porządku alfabetycznym**.

### Przykład:

Założmy, że w tabeli **CITY** znajdują się wpisy:

DEF, ABC, PQRS, WXY

- Długości nazw:
- ABC → 3
- DEF → 3
- PQRS → 4
- WXY → 3

**Najkrótsze nazwy:** "ABC", "DEF", "WXY" → wybieramy "ABC", bo jest pierwsze alfabetycznie.

**Najdłuższa nazwa:** "PQRS"

Query the two cities in **STATION** with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.

The **STATION** table is described as follows:

STATION	
Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where **LAT\_N** is the northern latitude and **LONG\_W** is the western longitude.

**Sample Input**

For example, **CITY** has four entries: **DEF**, **ABC**, **PQRS** and **WXY**.

**Sample Output**

```
ABC 3
PQRS 4
```

**Explanation**

When ordered alphabetically, the **CITY** names are listed as **ABC**, **DEF**, **PQRS**, and **WXY**, with lengths 3, 3, 4, and 3. The longest name is **PQRS**, but there are 3 options for shortest named city. Choose **ABC**, because it comes first alphabetically.

**Note**

You can write two separate queries to get the desired output. It need not be a single query.

## Zadanie 11

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-6/problem>

### Treść zadania:

Należy zapytać o listę nazw miast (**CITY**) z tabeli **STATION**, które zaczynają się na **samogłoskę (a, e, i, o, u)**. Wyniki nie mogą zawierać duplikatów.

Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from **STATION**. Your result cannot contain duplicates.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 12

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-7/problem>

### Treść zadania:

Należy zapytać o listę nazw miast (**CITY**) z tabeli **STATION**, które **kończą się na samogłoskę (a, e, i, o, u)**. Wyniki nie mogą zawierać duplikatów.

Query the list of CITY names ending with vowels (a, e, i, o, u) from **STATION**. Your result cannot contain duplicates.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 13

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-8/problem>

### Treść zadania:

Należy zapytać o listę nazw miast (**CITY**) z tabeli **STATION**, które zaczynają się **i kończą** na **samogłoskę (a, e, i, o, u)**. Wyniki nie mogą zawierać duplikatów.

Query the list of CITY names from **STATION** which have vowels (i.e., a, e, i, o, and u) as both their first and last characters. Your result cannot contain duplicates.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 14

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-9/problem>

### Treść zadania:

Należy zapytać o listę nazw miast (**CITY**) z tabeli **STATION**, które **nie zaczynają się na samogłoskę (a, e, i, o, u)**. Wyniki nie mogą zawierać duplikatów.

Query the list of CITY names from **STATION** that do not start with vowels. Your result cannot contain duplicates.

**Input Format**

The **STATION** table is described as follows:

STATION	
Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 15

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-10/problem>

### Treść zadania:

Należy zapytać o listę nazw miast (**CITY**) z tabeli **STATION**, które **nie kończą się na samogłoskę (a, e, i, o, u)**. Wyniki nie mogą zawierać duplikatów.

Query the list of CITY names from **STATION** that do not end with vowels. Your result cannot contain duplicates.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 16

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-11/problem>

### Treść zadania:

Należy zapytać o listę nazw miast (**CITY**) z tabeli **STATION**, które **albo nie zaczynają się na samogłoskę, albo nie kończą się na samogłoskę (a, e, i, o, u)**. Wyniki nie mogą zawierać duplikatów.

Query the list of CITY names from **STATION** that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 17

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-12/problem>

### Treść zadania:

Należy zapytać o listę nazw miast (**CITY**) z tabeli **STATION**, które **nie zaczynają się i nie kończą na samogłoskę (a, e, i, o, u)**. Wyniki nie mogą zawierać duplikatów.

Query the list of CITY names from **STATION** that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 18

Link do zadania: <https://www.hackerrank.com/challenges/more-than-75-marks/problem>

### Treść zadania:

Należy zapytać o **nazwy studentów**, którzy uzyskali **więcej niż 75 punktów** w tabeli **STUDENTS**. Wynik powinien być posortowany według **ostatnich trzech znaków nazwy** w porządku alfabetycznym. Jeśli dwóch lub więcej studentów ma takie same trzy ostatnie znaki, należy posortować ich dodatkowo według **ID rosnąco**.

Query the Name of any student in **STUDENTS** who scored higher than **75 Marks**. Order your output by the last three characters of each name. If two or more students both have names ending in the same last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending **ID**.

**Input Format**

Column	Type
<i>ID</i>	<i>Integer</i>
<i>Name</i>	<i>String</i>
<i>Marks</i>	<i>Integer</i>

The **STUDENTS** table is described as follows:  
(A-Z) and lowercase (a-z) letters.

The Name column only contains uppercase

**Sample Input**

<i>ID</i>	<i>Name</i>	<i>Marks</i>
1	Ashley	81
2	Samantha	75
4	Julia	76
3	Belvet	84

**Sample Output**

```
Ashley
Julia
Belvet
```

**Explanation**

Only Ashley, Julia, and Belvet have Marks > 75. If you look at the last three characters of each of their names, there are no duplicates and 'ley' < 'lia' < 'vet'.

## Zadanie 19

Link do zadania: <https://www.hackerrank.com/challenges/name-of-employees/problem>

### Treść zadania:

Należy zapytać o listę imion pracowników (**name**) z tabeli **Employee** i zwrócić wynik w kolejności alfabetycznej.

Write a query that prints a list of employee names (i.e.: the name attribute) from the **Employee** table in alphabetical order.

#### Input Format

The **Employee** table containing employee data for a company is described as follows:

Column	Type
employee_id	Integer
name	String
months	Integer
salary	Integer

where employee\_id is an employee's ID number, name is their name, months is the total number of months they've been working for the company, and salary is their monthly salary.

#### Sample Input

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

#### Sample Output

```
Angela
Bonnie
Frank
Joe
Kimberly
Lisa
Michael
Patrick
Rose
Todd
```

## Zadanie 20

Link do zadania: <https://www.hackerrank.com/challenges/salary-of-employees/problem>

### Treść zadania:

Należy zapytać o listę imion pracowników (**name**) z tabeli **Employee**, którzy spełniają następujące warunki:

1. Mają miesięczne wynagrodzenie **większe niż 2000**.
2. Pracują w firmie **mniej niż 10 miesięcy**.
3. Wyniki należy **posortować rosnaco według employee\_id**.

Write a query that prints a list of employee names (i.e.: the name attribute) for employees in **Employee** having a salary greater than \$2000 per month who have been employees for less than 10 months. Sort your result by ascending employee\_id.

**Input Format**

The **Employee** table containing employee data for a company is described as follows:

Column	Type
employee_id	Integer
name	String
months	Integer
salary	Integer

where employee\_id is an employee's ID number, name is their name, months is the total number of months they've been working for the company, and salary is the their monthly salary.

**Sample Input**

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

**Sample Output**

```
Angela
Michael
Todd
Joe
```

**Explanation**

Angela has been an employee for 1 month and earns \$3443 per month.  
Michael has been an employee for 6 months and earns \$2017 per month.  
Todd has been an employee for 5 months and earns \$3396 per month.  
Joe has been an employee for 9 months and earns \$3573 per month.  
We order our output by ascending employee\_id.

## Aggregation

### Zadanie 21

Link do zadania: <https://www.hackerrank.com/challenges/revising-aggregations-the-count-function/problem>

#### Treść zadania:

Należy obliczyć **liczbę miast** w tabeli **CITY**, które mają populację **większą niż 100 000**.

Query a count of the number of cities in **CITY** having a Population larger than 100,000.

**Input Format**

<b>CITY</b>	
<b>Field</b>	<b>Type</b>
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

The **CITY** table is described as follows:

### Zadanie 22

Link do zadania: <https://www.hackerrank.com/challenges/revising-aggregations-sum/problem>

#### Treść zadania:

Należy obliczyć **całkowitą populację** wszystkich miast znajdujących się w **dystrykcie “California”** w tabeli **CITY**.

Query the total population of all cities in **CITY** where District is **California**.

**Input Format**

<b>CITY</b>	
<b>Field</b>	<b>Type</b>
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

The **CITY** table is described as follows:

## Zadanie 23

Link do zadania: <https://www.hackerrank.com/challenges/revising-aggregations-the-average-function/problem>

### Treść zadania:

Należy obliczyć **średnią populację** wszystkich miast w tabeli **CITY**, które znajdują się w **dystrykcie “California”**.

Query the average population of all cities in CITY where District is California.														
<b>Input Format</b>														
<table border="1"><thead><tr><th colspan="2">CITY</th></tr><tr><th>Field</th><th>Type</th></tr></thead><tbody><tr><td>ID</td><td>NUMBER</td></tr><tr><td>NAME</td><td>VARCHAR2(17)</td></tr><tr><td>COUNTRYCODE</td><td>VARCHAR2(3)</td></tr><tr><td>DISTRICT</td><td>VARCHAR2(20)</td></tr><tr><td>POPULATION</td><td>NUMBER</td></tr></tbody></table> <p>The CITY table is described as follows:</p>	CITY		Field	Type	ID	NUMBER	NAME	VARCHAR2(17)	COUNTRYCODE	VARCHAR2(3)	DISTRICT	VARCHAR2(20)	POPULATION	NUMBER
CITY														
Field	Type													
ID	NUMBER													
NAME	VARCHAR2(17)													
COUNTRYCODE	VARCHAR2(3)													
DISTRICT	VARCHAR2(20)													
POPULATION	NUMBER													

## Zadanie 24

Link do zadania: <https://www.hackerrank.com/challenges/average-population/problem>

### Treść zadania:

Należy obliczyć **średnią populację** wszystkich miast w tabeli **CITY**, a następnie **zaokrąglić wynik w dół do najbliższej liczby całkowitej**.

Query the average population for all cities in CITY, rounded down to the nearest integer.														
<b>Input Format</b>														
<table border="1"><thead><tr><th colspan="2">CITY</th></tr><tr><th>Field</th><th>Type</th></tr></thead><tbody><tr><td>ID</td><td>NUMBER</td></tr><tr><td>NAME</td><td>VARCHAR2(17)</td></tr><tr><td>COUNTRYCODE</td><td>VARCHAR2(3)</td></tr><tr><td>DISTRICT</td><td>VARCHAR2(20)</td></tr><tr><td>POPULATION</td><td>NUMBER</td></tr></tbody></table> <p>The CITY table is described as follows:</p>	CITY		Field	Type	ID	NUMBER	NAME	VARCHAR2(17)	COUNTRYCODE	VARCHAR2(3)	DISTRICT	VARCHAR2(20)	POPULATION	NUMBER
CITY														
Field	Type													
ID	NUMBER													
NAME	VARCHAR2(17)													
COUNTRYCODE	VARCHAR2(3)													
DISTRICT	VARCHAR2(20)													
POPULATION	NUMBER													

## Zadanie 25

Link do zadania: <https://www.hackerrank.com/challenges/japan-population/problem>

### Treść zadania:

Należy obliczyć **sumę populacji** wszystkich japońskich miast w tabeli **CITY**. Kod kraju dla Japonii (**COUNTRYCODE**) to “**JPN**”.

CITY	
Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

The CITY table is described as follows:

## Zadanie 26

Link do zadania: <https://www.hackerrank.com/challenges/population-density-difference/problem>

### Treść zadania:

Należy obliczyć **różnicę między największą a najmniejszą populacją** w tabeli **CITY**.

CITY	
Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

The CITY table is described as follows:

## Zadanie 27

Link do zadania: <https://www.hackerrank.com/challenges/the-blunder/problem>

### Treść zadania:

Samantha miała obliczyć **średnią pensję miesięczną** dla wszystkich pracowników w tabeli **EMPLOYEES**, ale jej klawiatura miała uszkodzony klawisz **0**. W wyniku tego błędu Samantha **usunęła wszystkie zera** z wartości pensji przed obliczeniem średniej. Należy obliczyć **różnicę między rzeczywistą średnią pensją a błędnie obliczoną średnią pensją**. Wynik powinien być **zaokrąglony w górę do najbliższej liczby całkowitej**.

Samantha was tasked with calculating the average monthly salaries for all employees in the **EMPLOYEES** table, but did not realize her keyboard's 0 key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary.

Write a query calculating the amount of error (i.e. *actual – miscalculated* average monthly salaries), and round it up to the next integer.

#### Input Format

The **EMPLOYEES** table is described as follows:

Column	Type
<i>ID</i>	<i>Integer</i>
<i>Name</i>	<i>String</i>
<i>Salary</i>	<i>Integer</i>

**Note:** Salary is per month.

#### Constraints

$1000 < \text{Salary} < 10^6$ .

#### Sample Input

<i>ID</i>	<i>Name</i>	<i>Salary</i>
1	Kristeen	1420
2	Ashley	2006
3	Julia	2210
4	Maria	3000

#### Sample Output

2061

#### Explanation

The table below shows the salaries without zeros as they were entered by Samantha:

<i>ID</i>	<i>Name</i>	<i>Salary</i>
1	Kristeen	142
2	Ashley	26
3	Julia	221
4	Maria	3

Samantha computes an average salary of **98.00**. The actual average salary is **2159.00**.

The resulting error between the two calculations is  $2159.00 - 98.00 = 2061.00$ . Since it is equal to the integer **2061**, it does not get rounded up.

## Zadanie 28

Link do zadania: <https://www.hackerrank.com/challenges/earnings-of-employees/problem>

### Treść zadania:

**Całkowite zarobki** pracownika definiujemy jako:

total earnings = salary X months

Maksymalne **total earnings** to najwyższa wartość tej metryki dla dowolnego pracownika w tabeli **Employee**. Należy znaleźć: **Maksymalne total earnings** spośród wszystkich pracowników. **Liczę pracowników**, którzy osiągnęli tę maksymalną wartość. Wynik powinien zawierać **dwie liczby oddzielone spacją**.

We define an employee's total earnings to be their monthly `salary`  $\times$  `months` worked, and the maximum total earnings to be the maximum total earnings for any employee in the `Employee` table. Write a query to find the maximum total earnings for all employees as well as the total number of employees who have maximum total earnings. Then print these values as 2 space-separated integers.

**Input Format**

The `Employee` table containing employee data for a company is described as follows:

Column	Type
<code>employee_id</code>	Integer
<code>name</code>	String
<code>months</code>	Integer
<code>salary</code>	Integer

where `employee_id` is an employee's ID number, `name` is their name, `months` is the total number of months they've been working for the company, and `salary` is the their monthly salary.

**Sample Input**

<code>employee_id</code>	<code>name</code>	<code>months</code>	<code>salary</code>
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

**Sample Output**

```
69952 1
```

**Explanation**

<code>employee_id</code>	<code>name</code>	<code>months</code>	<code>salary</code>	<code>earnings</code>
12228	Rose	15	1968	29520
33645	Angela	1	3443	3443
45692	Frank	17	1608	27336
56118	Patrick	7	1345	9415
59725	Lisa	11	2330	25630
74197	Kimberly	16	4372	69952
78454	Bonnie	8	1771	14168
83565	Michael	6	2017	12102
98607	Todd	5	3396	16980
99989	Joe	9	3573	32157

The table and earnings data is depicted in the following diagram:

The maximum earnings value is **69952**. The only employee with earnings = **69952** is Kimberly, so we print the maximum earnings value (**69952**) and a count of the number of employees who have earned **\$69952** (which is **1**) as two space-separated values.

## Zadanie 29

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-2/problem>

### Treść zadania:

Należy zapytać o **sumę wszystkich wartości** w kolumnach:

1. **LAT\_N** (szerokość geograficzna północna)

2. **LONG\_W** (długość geograficzna zachodnia)

- Obie wartości muszą być **zaokrąglone do dwóch miejsc po przecinku**.

Query the following two values from the **STATION** table:

1. The sum of all values in LAT\_N rounded to a scale of **2** decimal places.
2. The sum of all values in LONG\_W rounded to a scale of **2** decimal places.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

#### Output Format

Your results must be in the form:

```
lat lon
```

where *lat* is the sum of all values in LAT\_N and *lon* is the sum of all values in LONG\_W. Both results must be rounded to a scale of **2** decimal places.

## Zadanie 30

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-13/problem>

### Treść zadania:

Należy obliczyć **sumę szerokości geograficznych północnych (LAT\_N)** w tabeli **STATION**, ale tylko dla wartości:

- **większych niż 38.7880**
- **mniejszych niż 137.2345**

Wynik musi być **obcięty (truncated) do 4 miejsc po przecinku**.

Query the sum of Northern Latitudes (LAT\_N) from **STATION** having values greater than **38.7880** and less than **137.2345**. Truncate your answer to 4 decimal places.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 31

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-14/problem>

### Treść zadania:

Należy zapytać o **największą wartość szerokości geograficznej północnej (LAT\_N)** z tabeli **STATION**, ale tylko dla wartości **mniejszych niż 137.2345**. Wynik musi być **obcięty (truncated) do 4 miejsc po przecinku**.

Query the greatest value of the Northern Latitudes (LAT\_N) from **STATION** that is less than 137.2345. Truncate your answer to 4 decimal places.

**Station.jpg**

**Input Format**

The **STATION** table is described as follows:

STATION	
Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 32

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-15/problem>

### Treść zadania:

Należy zapytać o **wartość długości geograficznej zachodniej (LONG\_W)** dla rekordu w tabeli **STATION**, który ma **największą szerokość geograficzną północną (LAT\_N) mniejszą niż 137.2345**. Wynik musi być **zaokrąglony do 4 miejsc po przecinku**.

Query the Western Longitude (LONG\_W) for the largest Northern Latitude (LAT\_N) in **STATION** that is less than 137.2345. Round your answer to 4 decimal places.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 33

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-16/problem>

### Treść zadania:

Należy zapytać o **najmniejszą wartość szerokości geograficznej północnej (LAT\_N)** z tabeli **STATION**, ale tylko dla wartości **większych niż 38.7780**. Wynik musi być **zaokrąglony do 4 miejsc po przecinku**.

Query the smallest Northern Latitude (LAT\_N) from **STATION** that is greater than **38.7780**. Round your answer to **4 decimal places**.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 34

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-17/problem>

### Treść zadania:

Należy zapytać o **wartość długości geograficznej zachodniej (LONG\_W)** dla rekordu w tabeli **STATION**, który ma **najmniejszą szerokość geograficzną północną (LAT\_N) większą niż 38.7780**. Wynik musi być **zaokrąglony do 4 miejsc po przecinku**.

Query the Western Longitude (LONG\_W) where the smallest Northern Latitude (LAT\_N) in **STATION** is greater than **38.7780**. Round your answer to 4 decimal places.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 35

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-18/problem>

### Treść zadania:

Mamy dwa punkty na płaszczyźnie 2D:

- $P_1(a, b)$ , gdzie:
  - $a$  to minimalna wartość szerokości geograficznej północnej (**LAT\_N**).
  - $b$  to minimalna wartość długości geograficznej zachodniej (**LONG\_W**).
- $P_2(c, d)$ , gdzie:
  - $c$  to maksymalna wartość szerokości geograficznej północnej (**LAT\_N**).
  - $d$  to maksymalna wartość długości geograficznej zachodniej (**LONG\_W**).

Należy obliczyć **Manhattan Distance** między punktami  $P_1$  i  $P_2$ , korzystając ze wzoru:

$$\text{Manhattan Distance} = |c - a| + |d - b|$$

Wynik musi być **zaokrąglony do 4 miejsc po przecinku**.

Consider  $P_1(a, b)$  and  $P_2(c, d)$  to be two points on a 2D plane.

- $a$  happens to equal the minimum value in Northern Latitude (LAT\_N in **STATION**).
- $b$  happens to equal the minimum value in Western Longitude (LONG\_W in **STATION**).
- $c$  happens to equal the maximum value in Northern Latitude (LAT\_N in **STATION**).
- $d$  happens to equal the maximum value in Western Longitude (LONG\_W in **STATION**).

Query the [Manhattan Distance](#) between points  $P_1$  and  $P_2$  and round it to a scale of 4 decimal places.

**Input Format**

The **STATION** table is described as follows:

STATION	
Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 36

Link do zadania:

### Treść zadania:

Mamy dwa punkty na płaszczyźnie 2D:

- $P_1(a, c)$ , gdzie:
  - $a$  to **minimalna wartość szerokości geograficznej północnej (LAT\_N)**.
  - $c$  to **minimalna wartość długości geograficznej zachodniej (LONG\_W)**.
- $P_2(b, d)$ , gdzie:
  - $b$  to **maksymalna wartość szerokości geograficznej północnej (LAT\_N)**.
  - $d$  to **maksymalna wartość długości geograficznej zachodniej (LONG\_W)**.

Należy obliczyć **odległość euklidesową** między punktami  $P_1$  i  $P_2$ ,

Wynik musi być **zaokrąglony do 4 miejsc po przecinku**.

Consider  $P_1(a, c)$  and  $P_2(b, d)$  to be two points on a 2D plane where  $(a, b)$  are the respective minimum and maximum values of Northern Latitude (LAT\_N) and  $(c, d)$  are the respective minimum and maximum values of Western Longitude (LONG\_W) in **STATION**.

Query the [Euclidean Distance](#) between points  $P_1$  and  $P_2$  and format your answer to display 4 decimal digits.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Zadanie 37

Link do zadania: <https://www.hackerrank.com/challenges/weather-observation-station-20/problem>

### Treść zadania:

Mediana to wartość dzieląca zbiór danych na dwie równe części – dolną i górną połowę.

Należy zapytać o **medianę szerokości geograficznych północnych (LAT\_N)** z tabeli **STATION** i zaokrąglić wynik do 4 miejsc po przecinku.

A **median** is defined as a number separating the higher half of a data set from the lower half. Query the median of the Northern Latitudes (LAT\_N) from **STATION** and round your answer to 4 decimal places.

#### Input Format

The **STATION** table is described as follows:

**STATION**

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

## Basic Join

### Zadanie 38

Link do zadania: <https://www.hackerrank.com/challenges/asian-population/problem>

#### Treść zadania:

Należy zapytać o **sumę populacji wszystkich miast** w tabeli **CITY**, które znajdują się w krajach z kontynentu '**Asia**'.

- Kolumna **CITY.CountryCode** odnosi się do kolumny **COUNTRY.Code** (klucz obcy).
- Kolumna **COUNTRY.Continent** zawiera nazwę kontynentu, więc należy wybrać tylko rekordy, gdzie **CONTINENT = 'Asia'**.
- **SUM()** służy do sumowania populacji.

Given the **CITY** and **COUNTRY** tables, query the sum of the populations of all cities where the CONTINENT is 'Asia'.

**Note:** CITY.CountryCode and COUNTRY.Code are matching key columns.

**Input Format**

CITY	
Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

The **CITY** and **COUNTRY** tables are described as follows:

COUNTRY	
Field	Type
CODE	VARCHAR2(3)
NAME	VARCHAR2(44)
CONTINENT	VARCHAR2(13)
REGION	VARCHAR2(25)
SURFACEAREA	NUMBER
INDEPYEAR	VARCHAR2(5)
POPULATION	NUMBER
LIFEEXPECTANCY	VARCHAR2(4)
GNP	NUMBER
GNPOLD	VARCHAR2(9)
LOCALNAME	VARCHAR2(44)
GOVERNMENTFORM	VARCHAR2(44)
HEADOFSTATE	VARCHAR2(32)
CAPITAL	VARCHAR2(4)

## Zadanie 39

Link do zadania: <https://www.hackerrank.com/challenges/average-population-of-each-continent/problem>

### Treść zadania:

Należy zapytać o **nazwy wszystkich kontynentów (COUNTRY.Continent)** oraz **średnią populację miast (CITY.Population)** na każdym kontynencie. Wynik powinien być zaokrąglony w dół do najbliższej liczby całkowitej.

- **CITY.CountryCode** odnosi się do **COUNTRY.Code** (klucz obcy).
- **COUNTRY.Continent** grupuje miasta według kontynentów.
- **AVG()** oblicza średnią populację miast.
- **FLOOR()** zaokrąga wynik w dół.

Given the **CITY** and **COUNTRY** tables, query the names of all the continents (**COUNTRY.Continent**) and their respective average city populations (**CITY.Population**) rounded down to the nearest integer.

**Note:** CITY.CountryCode and COUNTRY.Code are matching key columns.

**Input Format**

CITY	
Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

The **CITY** and **COUNTRY** tables are described as follows:

**COUNTRY**

Field	Type
CODE	VARCHAR2(3)
NAME	VARCHAR2(44)
CONTINENT	VARCHAR2(13)
REGION	VARCHAR2(25)
SURFACEAREA	NUMBER
INDEPYEAR	VARCHAR2(5)
POPULATION	NUMBER
LIFEEXPECTANCY	VARCHAR2(4)
GNP	NUMBER
GNPOLD	VARCHAR2(9)
LOCALNAME	VARCHAR2(44)
GOVERNMENTFORM	VARCHAR2(44)
HEADOFSTATE	VARCHAR2(32)
CAPITAL	VARCHAR2(4)
CODE2	VARCHAR2(2)

## Zadanie 40

Link do zadania: <https://www.hackerrank.com/challenges/full-score/problem>

### Treść zadania:

Należy zapytać o **hacker\_id** oraz **name** tych hakerów, którzy osiągnęli **pełne wyniki (maksymalną liczbę punktów)** w **więcej niż jednym zadaniu**.

- Wyniki powinny być **posortowane malejąco** według liczby zadań z pełnym wynikiem.
- Jeśli kilku hakerów ma tę samą liczbę zadań z pełnym wynikiem, należy ich **posortować rosnąco według hacker\_id**.

## Zadanie 41

Link do zadania: <https://www.hackerrank.com/challenges/harry-potter-and-wands/problem>

### Treść zadania:

Hermiona chce znaleźć **minimalną liczbę złotych galeonów** potrzebną do zakupu każdej róźdżki o wysokiej **mocy (power)** i **wieku (age)**, która **nie jest zła (non-evil)**.

- Wynik powinien zawierać kolumny:
  - **Id**
  - **Age**
  - **coins\_needed** (najniższa cena dla danej kombinacji power i age)
  - **power**
- Wynik powinien być **posortowany malejąco według power**.
- Jeśli dwie róźdżki mają tę samą moc, należy je **posortować malejąco według age**.

## Zadanie 42

Link do zadania: <https://www.hackerrank.com/challenges/challenges/problem>

### Treść zadania:

Należy zapytać o **hacker\_id**, **name**, oraz **łączną liczbę stworzonych wyzwań (challenges)** przez każdego studenta.

- Wynik powinien być **posortowany malejąco według liczby stworzonych wyzwań**.
- Jeśli kilku hakerów stworzyło tę samą liczbę wyzwań, to wynik powinien być **posortowany rosnąco według hacker\_id**.
- Jeśli więcej niż jeden haker stworzył tę samą liczbę wyzwań i ta liczba jest **mniejsza niż maksymalna liczba wyzwań stworzonych przez jednego hakera**, to **należy ich wykluczyć z wyników**.

## Zadanie 43

Link do zadania: <https://www.hackerrank.com/challenges/contest-leaderboard/problem>

### Treść zadania:

Należy zapytać o **hacker\_id**, **name** oraz **sumę najlepszych wyników** dla każdego hakera (**total score**).

- Wynik powinien być **posortowany malejąco według total\_score**.
- Jeśli kilku hakerów ma ten sam wynik, należy ich **posortować rosnąco według hacker\_id**.
- Należy **wykluczyć hakerów, którzy mają total\_score = 0**.

## Zadanie 44

Link do zadania: <https://www.hackerrank.com/challenges/sql-projects/problem>

### Treść zadania:

Dane zawierają informacje o zadaniach w projektach, zapisane w tabeli **Projects**.

Każde zadanie ma:

- **Task\_ID** – unikalny identyfikator zadania.
- **Start\_Date** – data rozpoczęcia zadania.
- **End\_Date** – data zakończenia zadania (**różnica między Start\_Date a End\_Date to zawsze 1 dzień**).

## Zadanie 45

Link do zadania: <https://www.hackerrank.com/challenges/placements/problem>

### Treść zadania:

Dane zawierają trzy tabele:

- **Students:** Lista studentów z ich unikalnym **ID** i **Name**.
- **Friends:** Każdy student ma dokładnie **jednego najlepszego przyjaciela** (**Friend\_ID**).
- **Packages:** Zawiera **ID** studenta i jego **oferowaną pensję** (w tysiącach dolarów).

Cel:

- Znaleźć studentów, **których najlepsi przyjaciele otrzymali wyższą pensję**.
- **Posortować wynik według pensji najlepszego przyjaciela** (rosnąco).
- **Każdy student ma unikalną pensję**, więc nie ma remisów w sortowaniu.