

# BAZY DANYCH

## ZADANIA

**z1.** [SQL] Dane są dwie tabele:

t1		
a	b	x
1	0	5
0	1	4
1	2	5

t2		
y	c	d
5	20	2
5	22	4
5	24	6

Na tych tabelach wykonano trzy operacje:

```
UPDATE t1 SET x=4 WHERE b>1;
DELETE FROM t2 WHERE c=24;
SELECT a+d AS ad FROM t1, t2 WHERE x=y;
```

Uzupełnij poniższą tabelkę tak, by przedstawiała tabelę wynikową:


tworzenie i wypełnienie tabel:

```
DROP TABLE IF EXISTS t1;
CREATE TABLE t1 (a INTEGER, b INTEGER, x INTEGER);
INSERT INTO t1 VALUES (1, 0, 5);
INSERT INTO t1 VALUES (0, 1, 4);
INSERT INTO t1 VALUES (1, 2, 5);
SELECT * FROM t1;

drop table IF EXISTS t2;
create table t2 (y INTEGER, c INTEGER, d INTEGER);
INSERT INTO t2 VALUES (5, 20, 2);
INSERT INTO t2 VALUES (5, 22, 4);
INSERT INTO t2 VALUES (5, 24, 6);
SELECT * FROM t2;
```

**z2.** [SQL] Dane są dwie tabele:

t1		
a	b	x
1	7	5
0	1	4
1	2	5

t2		
y	c	d
5	20	6
5	22	4
5	24	9

Na tych tabelach wykonano trzy operacje:

```
UPDATE t2 SET y=1 WHERE c=22;
UPDATE t1 SET x=4 WHERE b=2;
SELECT b, d FROM t1, t2 WHERE x=y;
```

Uzupełnij poniższą tabelkę tak, by przedstawiała tabelę wynikową:


**z3.** [SQL] Dane są dwie tabele:

t1			t2		
a	b	x	y	c	d
1	0	5	5	20	2
0	1	4	5	22	4
1	2	5	5	24	6

Na tych tabelach wykonano trzy operacje:

```
update t1 set x = 2 * a + b where b < 2;
update t2 set y = d where d > 2;
select distinct a,b,x from t1, t2 where d = y;
```

Uzupełnij poniższą tabelkę tak, by przedstawiała tabelę wynikową:


**z4.** [SQL] Dane są dwie tabele:

t1			t2		
a	b	x	y	c	d
1	7	5	5	20	6
0	1	4	5	22	4
1	2	5	5	24	9

Na tych tabelach wykonano trzy operacje:

```
update t1 set x = 2 * a + b where b < 2;
update t2 set y = d where d > 2;
select c,d,y from t1,t2 where b < x order by a asc;
```

Uzupełnij poniższą tabelkę tak, by przedstawiała tabelę wynikową:


**z5.** [SQL/C] Tablica *Workers* składa się z dwóch kolumn: *Name* i *Hours*. Należy uzupełnić poniższy kod języka C w taki sposób, aby realizował poniższe zapytanie języka SQL:

```
SELECT SUM(Hours) FROM Workers WHERE Name LIKE '_a%';
```

Na przykład dla poniższej tabeli

Name	Hours
Antczak	40
Nawrocki	20
Wolkowiak	30

wynikiem powinno być 50.

Tabela *Workers* jest zaimplementowana jako plik zawierający dwie kolumny: pierwsza kolumna zawiera nazwiska a druga liczbę godzin. Poniżej jest kod, który należy uzupełnić:

```
#define MaxS 100
#define MaxL 40
#include <stdio.h>
#include <string.h>
char Name[MaxS][MaxL];
int Hours[MaxS], Last;
```

```

int Query(void){
    int j, Sum= 0;
    for(j= 0; j <= Last; j++){

        Sum += Hours[j];
    }
    return Sum;
}

int main(void){
    for(Last = -1;
        scanf("%s", &Name[++Last])
            != -1;){
        scanf("%d", &Hours[Last]);
        printf("Total= %d\n", Query());
        return 0;
    }
}

```

**z6.** [SQL] Dane są trzy tabele:

Persons		Authors		Publishers	
MainKey	Name	Who	What	MainKey	Pub
1	Ala	1	1	1	Hey
2	Ewa	1	2	2	Abc
3	Jan	2	2	3	Abc
4	Ola	2	3	4	Hey
5	Ula	3	3		
		5	4		

Podaj (w jednym wierszu) imiona będące wynikiem następującego zapytania:

```

SELECT Name
FROM Persons, Authors, Publishers
WHERE Pub = 'Abc'
AND Publishers.MainKey = Authors.What
AND Authors.Who = Persons.MainKey;

```

tworzenie i wypełnienie tabel:

```

DROP TABLE IF EXISTS Persons;
CREATE TABLE Persons (MainKey INTEGER, Name VARCHAR(10));
INSERT INTO Persons VALUES (1, 'Alan');
INSERT INTO Persons VALUES (2, 'Ewa');
INSERT INTO Persons VALUES (3, 'Jan');
INSERT INTO Persons VALUES (4, 'Ola');
INSERT INTO Persons VALUES (5, 'Ula');
SELECT * FROM Persons;

drop table IF EXISTS Authors;
create table Authors (Who INTEGER, What INTEGER);
INSERT INTO Authors VALUES (1, 1);
INSERT INTO Authors VALUES (1, 2);
INSERT INTO Authors VALUES (2, 2);
INSERT INTO Authors VALUES (2, 3);
INSERT INTO Authors VALUES (3, 3);
INSERT INTO Authors VALUES (5, 4);
SELECT * FROM Authors;

drop table IF EXISTS Publishers;
create table Publishers (MainKey INTEGER, Pub VARCHAR(5));
INSERT INTO Publishers VALUES (1, 'Hey');
INSERT INTO Publishers VALUES (2, 'Abc');
INSERT INTO Publishers VALUES (3, 'Abc');
INSERT INTO Publishers VALUES (4, 'Hey');
SELECT * FROM Publishers;

```

**z7.** [SQL] Dane są trzy tabele:

Persons		Authors		Publishers	
MainKey	Name	Who	What	MainKey	Pub
1	Ala	1	1	1	Hey
2	Ewa	1	2	2	Abc
3	Jan	2	2	3	Abc
4	Ola	2	3	4	Hey
5	Ula	3	3		
		5	4		

Podaj (w jednym wierszu) imiona będące wynikiem następującego zapytania:

```
SELECT DISTINCT Name
FROM Persons, Authors, Publishers
WHERE Pub = 'Hey'
AND Publishers.MainKey = Authors.What
AND Authors.Who = Persons.MainKey;
```

**z8.** [SQL] W Strangeland wierzą, że szczęśliwe małżeństwo zależy od daty urodzenia małżonków. Dokładniej, liczba miesięcy plus liczba dni powinna być taka sama dla mężczyzny i kobiety (rok nie ma znaczenia). Załóż, że chcesz zaoferować im internetową swatkę i masz dwie tabele SQL: *MEN* i *WOMEN*, każda zawiera dzień (*DAY*) i miesiąc (*MONTH*) urodzenia, imię (*FIRST*) i nazwisko (*FAMILY*) każdej osoby. Ze względów marketingowych potrzebujesz nazwisk (imienia i nazwiska) wszystkich kobiet, które mają co najmniej jednego dobrze dopasowanego męskiego kandydata w bazie danych (chcesz zaprosić ich na imprezę taneczną). Napisz odpowiednie zapytanie SQL.

**SELECT**

**z9.** [AWK/SQL] Załóż, że tabela *Workers* jest reprezentowana jako plik tekstowy i ma trzy kolumny: *First name*, *Surname* i *Hours*. Oto przykładowy plik/tabela:

First name	Surname	Hours
Adam	Abacki	30
Ben	Bigman	26
Clara	Cimerman	10

Konieczne jest zaimplementowanie następującego zapytania SQL w AWK:

```
SELECT * FROM Workers WHERE Hours > (SELECT AVG(Hours) FROM Workers)
```

Dokończ poniższy program:

```
NR==1 {print;}
NR> 1 {L
    H
    s+=
}
END {for(i=2; i<=NR; i++)
    if(H[i] >
        print L[i];}
```