Group by
Csoportosítás lényege.

A feladat, ahova szeretnénk eljutni. **Tanáronként írjuk ki a diákok számát.**
Ezt kézzel úgy készíthetnénk el, hogy kivágjuk a sorokat és tanárok (nevei) szerint kupacokat képezünk. Minden sort (rekordot) az adott tanár nevének megfelelő kupacba rakjuk. Majd a kapott kupacok sorainak számát megszámoljuk. Hasonlóan működik ez SQL-ben is. Nézzük az előző feladatot kicsit másképp, egyszerűbben.
Alapfeladat. Csináljunk csoportokat a diákok korának megfelelően.
Eredménye SQL-ben az alábbi lesz:

```
SELECT kor
FROM tdiak
GROUP BY kor
```

Lekérdezés eredménye pedig:

Kor
16
19
21
22
28

De mindez az elméletben, háttérben, részletesen:
Végigmegyünk a diákokon, soronként – hiszen az SQL csak így tud -, majd a rekordokat a kor értéknek megfelelően szétdobáljuk külön-külön kupacokba (csoportokba).
Ehhez nézzük a diák táblánk tartalmát.

Dazon	**Nev**	**Kor**	**Ofo**
10	Zsiros B. Ödön	16	1
26	Feles Elek	19	2
32	Deb Ella	21	1
41	Ka Pál	16	1
57	Har Mónika	28	2
69	Elektrom Ágnes	22	3

A feladat szerint az alábbi kupacok jönnek létre, részletesen:
Az első sorunkat berakjuk a 16-os kupacba.

16			
10	Zsiros B. Ödön	16	1

Feles elek bekerül a második, 19-es csoportba, Deb Ella, a 21-es csoportba, Ka Pál pedig szintén a 16-os csoportba kerül. Tehát létrejönnek az alábbi kupacok, méghozzá kor szerint növekvő sorrendben, hiszen a szerint csoportosítottunk, s mégiscsak így logikus csoportjainkat sorbarakni tehát.

16	19	21	22	28
2 sor	1 sor	1 sor	1 sor	1 sor

De ez kiíráskor nem látszik, hiszen az adott sorokat nem írhatjuk ki, mert a kapott táblázatunk áttekithetetlen lenne. Ellenben függvénnyekkel kiszámíthatunk dolgokat a

kupacokban, megszámolhatjuk a csoport tagjait, azok valamely mezőnének átlagát, stb. Illetve kiírhatjuk, sőt ki is kell írjuk azokat a dolgokat, amik közösek a csoporton belül, jelen esetben a kor érték. Mivel ez különbözteti meg a csoportunkat egy másik csoporttól.

Visszatérhetünk tehát az alap feladathoz: **Tanáronként írjuk ki a diákok számát.**
Azaz mindkét táblára szükségünk van. Másrészt csoportosítani is kell, mert hiába van meg a tábláink összetett tartalma (egy képzett, párokat alkotó tábla), azt csak soronként dolgozhatjuk fel, egy sor kiértékeléséből pedig sehogyan sem tudhatjuk meg azt, hogy az adott tanárnak mennyi diákja lesz még a következő sorokban! Csoportokat kell tehát képezni!
Egyszerűsítsük tovább a feladatot úgy, hogy ismételjük az előzőekben tanultakat.
Mindkét táblára szükségünk van, tehát így kezdünk:

```
SELECT
FROM tDiak, tTanar
```

Ekkor viszont direkt szorzatot képez a kiértékelő, mi ezzel szemben szeretnénk figyelembe venni a két tábla közötti kapcsolatot is, amit az adatbáziskezelő automatikusan csak a táblák tartalmának módosításakor figyel: beillesztéskor (INSERT), módosításkor (UPDATE) és törléskor (DELETE).
A feladatunk tehát az alábbiak szerint módosul:

```
SELECT
FROM tDiak, tTanar
WHERE tDiak.ofo=tTanar.tTazon
```

Most térhetünk rá a valódi feladatra.
A diákok számát szeretnénk kiiratni, tehát a kimenet a COUNT(*) lesz.

```
SELECT count(*)
FROM tDiak, tTanar
WHERE ofo=Tazon
```

Ez megszámolja a diákokat, még az összeset. Nem tanáronként, hanem mindet, azaz kiírja, hány diákunk van.
Ha tanáronként szeretnénk, akkor csoportokat kell képezni tanáronként, azaz:

```
SELECT count(*)
FROM tDiak, tTanar
WHERE ofo=Tazon
GROUP BY ttanar.nev
```

Ez hibát fog jelezni, mert a csoportosítási mezőnek is ki kell mennie a mezőnév felsorolásban, hiszen az érték az, amelyik az adott csoportokban azonos. Ellenkező esetben a csoportjainkat nem tudnánk megkülönböztetni egymástól. Arról nem is beszélve, hogy a kapott eredmény áttekinthető is.

```
SELECT ttanar.nev, count(*)
FROM tDiak, tTanar
WHERE ofo=Tazon
GROUP BY ttanar.nev
```

Látszólag megnyugodhatunk. Elkészültünk. Azonban a feladatunk az SQL szempontjából ugyan helyes, azonban még mindig nem tökéletes. Előfordulhat ugyanis olyan eset, hogy több ugyanolyan nevű tanárunk is van, őket azonban nem vesszük külön csoportba, de mégsem azonos személyekről van szó. Ezért tehát akkor járunk el helyesen, ha a kulcs mezőt is belevesszük a csoportosításba.

```
SELECT ttanar.nev, count(*)
FROM tDiak, tTanar
WHERE ofo=Tazon
GROUP BY tazon, ttanar.nev
```

A kilistázandó mezők közül elég a nevet szerepeltetni, hiszen azok a csoportokban eltérőek, ha van két ugyanolyan nevű tanárunk, akkor ő kétszer is szerepel. Ez azonban nem gond, sőt pont ezt akartuk elérni.
Feladatunkban ez így néz ki:

Ttanar.nev	Count(*)
Zsakó László	3
Szlávi Péter	2
Pap Gáborné	1

GROUP BY tazon, ttanar.nev: Ilyenkor azok a sorok kerülnek külön csoportba, ahol a tazon és ttanar.nev érték valamelyike eltérő. Tehát például a tazon=3 és ttanar.nev=’Pap Gáborné’ és a tazon=5 és ttanar.nev=’Pap Gáborné’sorok külön csoportot alkotnak.

Feladatunk is jó eredményt ad.
Az eredmény táblába nem kerülnek bele azok a tanárok (’Illés Zoltán’), akik nem tanítanak senkit sem. Ez logikus is, hiszen a FROM és a WHERE rész kiértékelése után kialakult közös táblánkban sem volt már olyan sor, ahol a tanár neve oszlopban az érték ’Illés Zoltán’ lett volna. Pedig a kiértékelés a továbbiakban e szerint halad.

Hasonlóan kiszámítható tanáronként a legidősebb diák is.

```
SELECT ttanar.nev, max(kor)
FROM tDiak, tTanar
WHERE ofo=Tazon
GROUP BY tazon, ttanar.nev
```

Azonban azt már nem tehetjük meg, hogy kiírjuk az adott legidősebb diák nevét is tanáronként, mivel a csoportokban lévő rekordok konkrét értékei nem érhetőek el. Ehhez már összetett lekérdezéseket kell készítenünk. Méghozzá az előbb elkészített select eredményeit megkapva kikeressük azt az eredeti közös táblánkban. Ezeket, mint konkrét, fix értékeket tekintve.

HAVING:
Mi van akkor azonban, ha egy olyan feltételt szeretnénk megfogalmazni, amihez szükségünk van valaminek a kiszámolására. Például az érdekel minket, hogy ki tanít kettőnél több diákot.
Ekkor csoportos feltételt kell használnunk, ezt a célt szolgálja a HAVING.

A where-el sorokra vonatkozó feltételeket készíthetünk, annak segítségével sorokat válogathatunk ki, míg a HAVING konkrét csoportokat szűr ki.

```
SELECT ttanar.nev
FROM tdiak, ttanar
WHERE ofo=tazon
GROUP BY tazon, ttanar.nev
HAVING count(*)>2
```

Azaz, eldobálja azokat a csoportokat, amelyekben kevesebb, mint két rekord (diák) van.

Megjegyzés: A kettőnél több diák és a legalább 3 diák ugyanazt jelenti, tehát a count(*)>2 és a count(*)>=3 csak hatékonyságban és megfogalmazásában tér el egymástól.