

Adatbázis kezelés I.

SQL bevezető

Rostagni Csaba

2022.08.28

Ezen az órán... I

- 1 Bevezető
- 2 Egyszerű lekérdezések (SELECT)
- 3 Műveleti jelek
- 4 Feltételes lekérdezések (WHERE)
- 5 A NULL érték
- 6 Rendezés (ORDER BY)
- 7 Sorok számának limitálása (LIMIT)

Ezen az órán... II

- 8 Számított mezők
- 9 Matematikai függvények
- 10 Szöveg függvények
- 11 Dátum és idő függvények
- 12 Egyéb hasznos függvények
- 13 Aggregált (összesítő) függvények
- 14 Csoportosítás (GROUP BY)

Ezen az órán... III

- 15 Feltételek csoportosított mezőre (HAVING)
- 16 Többtáblás lekérdezések

Tartalom I

1 Bevezető



Speciális karakterek

A diákon különböző speciális jelek találhatók meg.

aposztróf

- A MySQL aposztóffal jelöli a szöveget.

pl.: 'szoveg'

-  + 

backtick

- Így jelöli adatbázisokat, táblákat és a mezőneveket.

pl.: `tablanev`

-  + 

Megjegyzés (egy soros)

- A # karaktertől a sor végéig
- A -- karaktersorozattól a sor végéig
 - Utána egy szóköz kell!
pl.: --*Ez egy megjegyzés*
 - Kissé eltér a szabványtól, de **ezt használjuk!**

Linkek:

- MySQL dokumentáció: Megjegyzések
- MySQL dokumentáció: A szabványtól eltérő megjegyzés

Megjegyzés (több soros)

- Nyitó karaktersorozat: `/*`
- Záró karaktersorozat: `*/`
- Szöveg közben használható
pl.: `/* Ez szövegeközi vagy több soros megjegyzes */`

Linkek:

- MySQL dokumentáció: Megjegyzések
- MySQL dokumentáció: A szabványtól eltérő megjegyzés

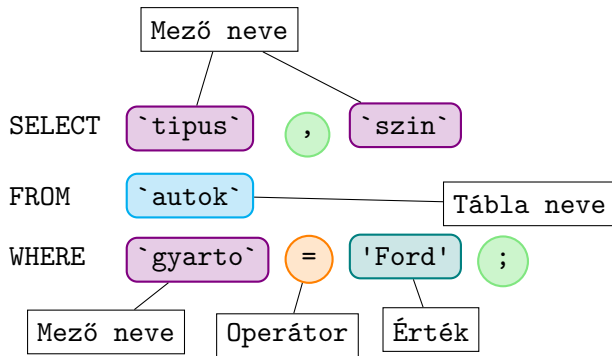
Különböző típusú értékek jelölése

- A szöveges értékeket aposztróf közé kell elhelyezni.
'Nagy Lajos'
- A dátumokat és az időket is aposztróf között fogjuk szabványosan megadni.
'2018-12-01'
'15:55:30'
'2018-12-01 12:55:30'
- Az egész számokat számmal megadhatjuk a szokásos módon.
10
- A valós számokat **tizedes ponttal** kell elválasztani!
10.5

Tartalom I

2 Egyszerű lekérdezések (SELECT)

Egyszerű SELECT felépítése



Első lekérdezésünk

Listázzunk ki minden adatot az **autok** táblából!

MySQL

```
SELECT *  
FROM `autok`;
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
XXX-111	Opel	Adam	M1	benzin	piros
AAA-555	Honda	Jazz	M1	hibrid	kék
ABC-123	Ford	Focus	M1	diesel	kék
AA-AX-1234	Ford	Fiesta	M1	benzin	sárga

Mezők kiválasztása

Listázzuk ki az összes autó **rendszámát** és **típusát**!

MySQL

```
SELECT `rendszám`,`típus`  
FROM `autok`;
```

rendszám	típus
XXX-111	Adam
AAA-555	Jazz
ABC-123	Focus
AA-AX-1234	Fiesta

Tartalom I

- 3 Műveleti jelek
 - Összehasonlító operátorok
 - Logikai operátorok

Tartalom

- 3 Műveleti jelek
 - Összehasonlító operátorok
 - Logikai operátorok

Összehasonlító operátorok

SQL	Mat.	Megnevezés
<	<	kisebb
>	>	nagyobb
<=	≤	kisebb egyenlő
>=	≥	nagyobb egyenlő
=	=	egyenlő
<>	≠	nem egyenlő

táblázat: összehasonlító operátorok

Megjegyzés:

- A != is a nem egyenlőt jelenti (MySQL, MS SQL, Oracle, ...)
- A ^= is a nem egyenlőt jelenti (Oracle)
- A <> formátum felel meg az ANSI szabványnak, ez az elvárt!

Tartalom

- 3 Műveleti jelek
 - Összehasonlító operátorok
 - Logikai operátorok

Logikai operátorok

SQL	Mat.	Megnevezés
NOT	\neg	nem
AND	\wedge	és
OR	\vee	vagy

táblázat: logikai operátorok

Az operátorok precedencia szerint csökkenő sorrendben vannak feltüntetve.

Linkek:

- MySQL dokumentáció: Operátorok precedenciája

Tartalom I

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Feltételek megadása (WHERE)

Listázzuk ki a **benzines** autók minden adatát

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag` = 'benzin';
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
XXX-111	Opel	Adam	M1	benzin	piros
AA-AX-1234	Ford	Fiesta	M1	benzin	sárga

Megadott mezők és feltételek

Listázzuk ki a **benzines** autók **gyártóját** és **típusát**!

MySQL

```
SELECT `gyarto`,`tipus`  
FROM `autok`  
WHERE `uzemanyag` = 'benzin';
```

gyarto	tipus
Opel	Adam
Ford	Fiesta

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- **Tagadás**
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Egyszerű tagadás

Listázzuk ki a **nem benzines** autókat.

MySQL

```
SELECT * FROM `autok`  
WHERE `uzemanyag` <> 'benzin';
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
AAA-555	Honda	Jazz	M1	hibrid	kék
ABC-123	Ford	Focus	M1	diesel	kék

- Használhattuk volna a != operátort, csak az SQL szabvány nem azt tartalmazza.
- Egyszerű feltételek esetén ilyen egyszerű tagadni.

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Logikai operátorok: ÉS (AND)

Listázzuk ki a **benzines Fordok** minden adatát!

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag` = 'benzin'  
      AND `gyarto` = 'Ford';
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
AA-AX-1234	Ford	Fiesta	M1	benzin	sárga

Logikai operátorok: VAGY (OR)

Listázzuk ki a **piros** vagy **sárga** színű autók minden adatát!

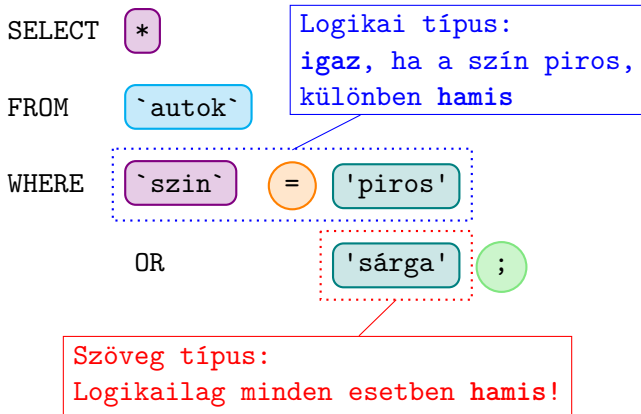
```
SELECT *  
FROM autok  
WHERE szin = 'piros'  
       OR 'sárga';
```

logikai hiba

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
XXX-111	Opel	Adam	M1	benzin	piros

- Hol van a sárga autó?

Logikai operátorok: VAGY (OR)



- Mivel a második feltétel mindig hamis, így olyan, mintha ott se lenne.

Logikai operátorok: VAGY (OR)

Listázzuk ki a **piros**, vagy **sárga** színű autók minden adatát!

MySQL

```
SELECT *  
FROM `autok`  
WHERE `szin` = 'piros'  
      OR `szin` = 'sárga';
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
XXX-111	Opel	Adam	M1	benzin	piros
AA-AX-1234	Ford	Fiesta	M1	benzin	sárga

Logikai operátorok: VAGY (OR)

Listázzuk ki a **hibrid** és a **benzines** autók minden adatát!

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag` = 'benzin'  
      OR `uzemanyag` = 'hibrid';
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
XXX-111	Opel	Adam	M1	benzin	piros
AAA-555	Honda	Jazz	M1	hibrid	kék
AA-AX-1234	Ford	Fiesta	M1	benzin	sárga

- A köznyelvben használt "és" logikailag lehet, hogy "vagy"-ot jelent.

És / Vagy operátorok

- `&&`
 - a MySQL egy nem szabványos kiegészítése,
 - a 8.0.17-es verziótól kezdve elavult, meg fog szűnni
- `||`
 - a MySQL egy nem szabványos kiegészítése,
 - a 8.0.17-es verziótól kezdve elavult, meg fog szűnni
 - Az ANSI szabvány szerint összefűzést jelent

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- **ÉS/VAGY kombinálása**
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Logikai operátorok kombinálása (hibás próbálkozás)

Listázzuk ki azoknak a **benzines** autóknak minden adatát, melyek **Ford** vagy **Honda** gyártmányúak.

```
SELECT * FROM `autok`  
WHERE `uzemanyag` = 'benzin'  
    AND `gyarto` = 'Ford'  
    OR `gyarto` = 'Honda';
```

logikai hiba

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
AAA-555	Honda	Jazz	M1	hibrid	kék
AA-AX-1234	Ford	Fiesta	M1	benzin	sárga

- **Probléma:** a Honda Jazz is megjelent, ami nem benzines!

Logikai operátorok kombinálása hiba magyarázata

Az ÉS logikai operátor precedenciája magasabb.
Elsőnek ez a feltétel teljesül.

SELECT

*

FROM

`autok`

WHERE

`uzemanyag`

=

'benzin'

AND

`gyarto`

=

'Ford'

OR

`gyarto`

=

'Honda'

;

A Honda esetén az üzemanyagra nem szűr.

Logikai operátorok kombinálása

Listázzuk ki azoknak a **benzines** autóknak minden adatát, melyek **Ford** vagy **Honda** gyártmányúak.

MySQL

```
SELECT *  
FROM `autok`  
WHERE (`uzemanyag` = 'benzin')  
      AND (`gyarto` = 'Ford' OR `gyarto` = 'Honda');
```

rendszám	gyarto	tipus	kategoria	uzemanyag
AA-AX-1234	Ford	Fiesta	M1	benzin

- Megfelelően zárójelezve rövid, tömör kódot kapunk
- Későbbi módosításkor is csak egy helyen kell módosítani

Egyszerű tagadás (NOT)

Listázzuk ki a **nem benzines** autókat.

MySQL

```
SELECT * FROM `autok`  
WHERE NOT ( `uzemanyag` = 'benzin' );
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
AAA-555	Honda	Jazz	M1	hibrid	kék
ABC-123	Ford	Focus	M1	diesel	kék

Megjegyzés:

- Ennél az egyszerű példánál a zárójelezés elhagyható, összetettebbeknél célszerű kitenni

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Választás több elem közül

Listázzuk ki a **benzines**, **diesel** és **elektromos** autók minden adatát

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag` = 'benzin'  
           OR 'diesel'  
           OR 'elektromos';
```

logikai hiba

- Az ``uzemanyag` = 'benzin'` logikai értéke lehet igaz vagy hamis.
- A `'diesel'` és az `'elektromos'` logikai értéke MINDIG HAMIS!
- Mivel a **(bármilyen) vagy hamis** értéke **(bármilyen)**, így a benzines autók összes adatát kapjuk meg.!

Választás több elem közül

Listázzuk ki a **benzines**, **diesel** és **elektromos** autók minden adatát

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag` = 'benzin'  
      OR `uzemanyag` = 'diesel'  
      OR `uzemanyag` = 'elektromos';
```

- Az `uzemanyag` mezőt fölöslegesen sokszor kellett felsorolni
- ÉS/VAGY együttes alkalmazása okozhat problémát, ha nincs jól zárójelezve

Az IN operátor

Listázzuk ki a **benzines**, **diesel** és **elektromos** autók minden adatát

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag`  
      IN ('benzin', 'diesel', 'elektromos');
```

- Csak egyenlőség vizsgálat esetén alkalmazható
- Az `uzemanyag` mezőt csak egyszer kellett felsorolni
- ÉS alkalmazásakor nem zavar be a precedencia
- Számok esetén az aposztrófok elhagyhatóak. pl.: **IN**(1,5,8)

Linkek:

- MySQL dokumentáció: Az IN operátor

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- **Azonos mezőre több lehetséges érték tagadással (NOT IN)**
- Két érték között (BETWEEN)

Azonos mezőre több lehetséges érték tagadással

Listázzuk ki azokat autókat, melyekre nem igaz, hogy **diesel** vagy **benzin** az üzemanyaga.

```
SELECT * FROM `autok`  
WHERE `uzemanyag` <> 'benzin'  
      OR `uzemanyag` <> 'diesel';
```

logikai hiba

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
XXX-111	Opel	Adam	M1	benzin	piros
AAA-555	Honda	Jazz	M1	hibrid	kék
ABC-123	Ford	Focus	M1	diesel	kék
AA-AX-1234	Ford	Fiesta	M1	benzin	sárga

- Az `uzemanyag` <> 'benzin' megjeleníti az összes NEM benzinest, így az összes dieselt is!
- Az `uzemanyag` <> 'diesel' megjeleníti az összes NEM dieselt!

Azonos mezőre több lehetséges érték tagadással

Listázzuk ki azokat autókat, melyekre nem igaz, hogy **diesel** vagy **benzin** az üzemanyaga.

MySQL

```
SELECT * FROM `autok`  
WHERE NOT (`uzemanyag` = 'benzin'  
          OR `uzemanyag` = 'diesel');
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
AAA-555	Honda	Jazz	M1	hibrid	kék

- Megnézi minden egyes sorra, hogy diesel vagy benzin üzemű az autó,
- Amennyiben a válasz nem, akkor jeleníti csak meg.

Azonos mezőre több lehetséges érték tagadással

Listázzuk ki azokat autókat, melyekre nem igaz, hogy **diesel** vagy **benzin** az üzemanyaga.

MySQL

```
SELECT * FROM `autok`  
WHERE `uzemanyag` <> 'benzin'  
AND `uzemanyag` <> 'diesel';
```

rendszer	gyarto	tipus	kategoria	uzemanyag	szin
AAA-555	Honda	Jazz	M1	hibrid	kék

- A feladatot átfogalmazva adódik egy másik megoldás:
 - "Listázzuk ki a se nem **benzines**, se nem **diesel** autók összes adatát."
- A zárójel felbontásakor a De Morgan-azonosságokat figyelembe kell venni

Linkek:

- De Morgan-azonosságok - Wikipedia

A NOT IN operátor

Listázzuk ki a se nem **benzines**, se nem **diesel** autók összes adatát.

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag`  
      NOT IN ('benzin', 'diesel');
```

- Egy zárójelben felsorolhatjuk a kizárandó értékeket.

Linkek:

- [MySQL dokumentáció: A NOT IN operátor](#)

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Két érték közötti vizsgálat

Jelenítsük meg a 10 és 20 év közötti diákokat.

MySQL

```
SELECT * FROM `diakok`  
WHERE   `kor` >= 10  
      AND `kor` <= 20;
```

- A `kor` mezőt kétszer is szerepeltetni kell!
- Oda kell figyelni, hogy egyik oldalt se maradjon le az egyenlőség!
- Oda kell figyelni, hogy **és** kapcsolat legyen a két feltétel között.
- Összetett feltételben ez okozhat gondot!

A BETWEEN ... AND ... operátor

Jelenítsük meg a 10 és 20 év közötti diákokat.

MySQL

```
SELECT * FROM `diakok`  
WHERE `kor` BETWEEN 10 AND 20;
```

- A 10 és 20, azaz a minimum és a maximum is benne lesz a szűrésben, nem lehet lefelejtteni az egyenlőséget
- Egy egységet alkot, így összetett feltételben zárójelek nélkül is használható

Linkek:

- [MySQL dokumentáció: A BETWEEN operátor](#)

A NOT BETWEEN ... AND ... operátor

Jelenítsük meg azokat a diákokat, akik élettkora nem esik 10 és 20 közé.

MySQL

```
SELECT * FROM `diakok`  
WHERE `kor` NOT BETWEEN 10 AND 20;
```

Ugyanígy működne, ha a sima BETWEEN eredményét letagadnánk.

MySQL

```
SELECT * FROM `diakok`  
WHERE NOT (`kor` BETWEEN 10 AND 20);
```

- A zárójel itt elhagyható, az átláthatóság miatt került be.

Linkek:

- [MySQL dokumentáció: A NOT BETWEEN operátor](#)

Tartalom I

5 A NULL érték

A NULL érték

- A null érték speciális érték
- A NULL nem egyenlő 0-val!
- A NULL nem egyenlő az üres szöveggel!
- Adatbázisan NULL jelentése: Az adott mező értéke nincs megadva, nincs kitöltve, ismeretlen.

A NULL vizualizálása



ábra: A NULL érték vizualizálása

Forrás: <https://www.reddit.com/r/ProgrammerHumor>

IS NULL

Az IS NULL segítségével ellenőrizhetjük, hogy a mező értéke NULL-e.

```
SELECT * FROM `autok`  
WHERE `tipus` IS NULL;
```

MySQL

IS NOT NULL

Az IS NOT NULL segítségével ellenőrizhetjük, hogy a mező értéke **nem** NULL.

```
FROM `autok`  
WHERE `tipus` IS NOT NULL;
```

MySQL

COALESCE()

```
COALESCE(value,...)
```

- Visszaadja az első **nem** NULL értéket

```
SELECT COALESCE(NULL, NULL, NULL, 'A', 'B') AS `e`  
FROM DUAL;
```

MySQL

e

A

Tartalom I

6 Rendezés (ORDER BY)

ORDER BY

MySQL

```
SELECT * FROM `autok`  
ORDER BY `gyarto`;
```

- Az **ORDER BY** után sorolható fel, hogy melyik mező(k) alapján, növekvő vagy csökkenő sorrendbe rendezve adja vissza adatokat
- **ASC** növekvő (alapértelmezett)
- **DESC** csökkenő

Növekvő sorrend ASC

Jelenítsük meg az autók minden adatát a gyártók szerinti **növekvő** sorrendben.

MySQL

```
SELECT * FROM `autok`  
ORDER BY `gyarto` ASC;
```

rendszam	gyarto	tipus	kategoria	uzemanyag	szin
ABC-123	Ford	Focus	M1	diesel	kék
AA-AX-1234	Ford	Fiesta	M1	benzin	sárga
AAA-555	Honda	Jazz	M1	hibrid	kék
XXX-111	Opel	Adam	M1	benzin	piros

Csökkenő sorrend DESC

Jelenítsük meg az autók minden adatát a gyártók szerinti **csökkenő** sorrendben.

MySQL

```
SELECT * FROM `autok`  
ORDER BY `gyarto` DESC;
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
XXX-111	Opel	Adam	M1	benzin	piros
AAA-555	Honda	Jazz	M1	hibrid	kék
ABC-123	Ford	Focus	M1	diesel	kék
AA-AX-1234	Ford	Fiesta	M1	benzin	sárga

Azonosak esetén...

A példán látható, hogy a Ford Focus és Ford Fiesta növekvő és csökkenő rendezés esetén is ugyanabban a sorrendben jelentek meg. Vesszővel felsorolhatunk több rendezési szempontot is.

Összetett rendezés példa 1

Jelenítse meg az autók adatát **gyártókszerint csökkenő**, míg **típus szerint növekvő** sorrendben!

MySQL

```
SELECT * FROM `autok`  
ORDER BY `gyarto` DESC, `tipus` ASC;
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
XXX-111	Opel	Adam	M1	benzin	piros
AAA-555	Honda	Jazz	M1	hibrid	kék
AA-AX-1234	Ford	Fiesta	M1	benzin	sárga
ABC-123	Ford	Focus	M1	diesel	kék

A termékek tábla

id	nev	kategoria	netto	penznem	afa
1	4K TV	tv	499	EUR	0.19
2	Mobil 32GB	mobil	299	EUR	0.19
3	Mobil 128GB	mobil	679	EUR	0.19
4	Olcsó laptop	laptop	269	EUR	0.19
5	Drága laptop	laptop	1729	EUR	0.19
6	Könyv	könyv	NULL	NULL	NULL

Álnév probléma!

Jelenítsük meg a termékek nevét és a bruttó árat a bruttó szerinti növekvő sorrendben.

```
SELECT `nev`, `netto` * `afa` AS 'brutto'  
FROM `termekek`  
ORDER BY 'brutto' ASC;
```

logikai hiba

- A kód le fog futni, de nem a várt eredménnyel.

Álnév hiba eredménye

`nev`	`brutto`	'brutto'
4K TV	94.80999881029129	brutto
Mobil 32GB	56.80999928712845	brutto
Mobil 128GB	129.00999838113785	brutto
Olcsó laptop	51.10999935865402	brutto
Drága laptop	328.50999587774277	brutto
Könyv	-	brutto

- Az eredmény rendezett, de egy harmadik, mesterségesen generált mező alapján.
- Fontos, hogy az álnév backtick legyen, itt ennek hiányában nem működött a rendezés.

Rendezés számított mező alapján

MySQL

```
SELECT `nev`, `netto` * (1 + `afa`) AS `brutto`  
FROM `termekek`  
ORDER BY `brutto` ASC;
```

vagy

MySQL

```
SELECT `nev`, `netto` * (1 + `afa`)  
FROM `termekek`  
ORDER BY `netto` * (1 + `afa`) ASC;
```

Rendezés sorszám alapján

MySQL

```
SELECT `nev`, `netto`  
FROM `termekek`  
ORDER BY 2;
```

- A SELECT után felsorolt n-edik mező szerint is rendezhetőek az adatok
- A mezőket 1-től indexeli
- A fenti lekérdezés a második mező, azaz a nettó ár alapján rendez

Figyelem!

Nem az eredeti tábla, hanem a SELECT után felsorolt mezők sorszáma az, ami számít!

Rendezés és feltétel

Jelenítsük meg a **kék** színű autók minden adatát a gyártók szerinti **növekvő** sorrendben.

MySQL

```
SELECT * FROM `autok`  
WHERE `szin` = 'kék'  
ORDER BY `gyarto` ASC;
```

rendszám	gyarto	tipus	kategoria	uzemanyag	szin
ABC-123	Ford	Focus	M1	diesel	kék
AAA-555	Honda	Jazz	M1	hibrid	kék

Rendezés és feltétel

Jelenítsük meg a **300 eurónál drágább** termékek nevét és **nettó árát**, utóbbi **szerint növekvő** sorrendben.

MySQL

```
SELECT `nev`, `netto`  
FROM `termekek`  
WHERE `netto` > 300  
ORDER BY `netto` ASC;
```

nev	netto
4K TV	499
Mobil 128GB	679
Drága laptop	1729

Tartalom I

7 Sorok számának limitálása (LIMIT)

Sorok számának limitálása (LIMIT)

A LIMIT segítségével megadhatjuk, hogy az eredmény hány sorát szeretnénk megkapni.

```
SELECT * FROM `tanulok`  
LIMIT 2;
```

MySQL

https://www.w3schools.com/php/php_mysql_select_limit.asp

Lapozó

Első N rekord kihagyása:

MySQL

```
SELECT id, vnev, knev  
FROM `tanulok`  
LIMIT 10, 5;
```

- Kihagyja az első 10 tanulót és onnantól kezdve jelenít meg legfeljebb 5 tanulót.
- Felhasználás a gyakorlatban:
 - Nagy mennyiségű adat esetén egy weboldalon nem jó ötlet az összes adatot egyszerre megjeleníteni, célszerű több oldalra bontani.
 - Amennyiben 1 oldalon 5 adatot jelenítünk meg, úgy a fenti példa a 3. oldal adatait kéri le
 - Az első két oldalon megjelenő 5-5, azaz összesen 10 rekord kerül kihagyásra, majd jön az aktuális oldal tartalma

Legjobb 3 tanuló

Listázzuk ki a három legjobb tanulót.

MySQL

```
SELECT id, vnev, knev  
FROM `tanulok`  
ORDER BY `atlag` DESC  
LIMIT 3;
```

- A tanulókat átlag alapján rendezi **csökkenő** sorrendbe
- Az átlag nem kerül megjelenítésre
- Az **ORDER BY** és **LIMIT** kombinálásával létrehozható toplista

A legmagasabb tanuló

Hogy hívják a legmagasabb tanulót?

MySQL

```
SELECT vnev, knev  
FROM `tanulok`  
ORDER BY `magassag` DESC  
LIMIT 1;
```

- A tanulókat a magasságuk alapján rendezi csökkenő sorrendbe
- A magasság nem kerül megjelenítésre
- Az **ORDER BY** és **LIMIT** kombinálásával meghatározható hogyan hívják a legmagasabb tanulót

A legidősebb tanuló

Hogy hívják a legidősebb tanulót?

MySQL

```
SELECT vnev, knev  
FROM `tanulok`  
ORDER BY `szuletesi_datum` ASC  
LIMIT 1;
```

- A tanulókat a születési dátumok alapján rendezi **növekvő** sorrendbe
 - Minél korábban született valaki, annál idősebb lesz
- Az **ORDER BY** és **LIMIT** kombinálásával meghatározható hogyan hívják legidősebb tanulót

Tartalom I

8 Számított mezők

Számított mezők

- SQL lekérdezésben lehetőség van számítások elvégzésére
- Szerepelhet benne:
 - Konstans érték: 1, 'hello', '2000-01-01', true
 - Egy mező az adatbázisból: `ar`
 - Valamilyen függvény: sqrt(9), round(1.975,2)

A DUAL "tábla"

MySQL

```
SELECT 10 + 5 AS `eredmeny`  
FROM dual;
```

- Előfordulhatnak olyan lekérdezések, amit nem táblától szeretnénk lekérdezni.
- A **dual** egy speciális „tábla”, ahonnan bármit lekérdezhetünk.
 - **Itt a backtick nem használható!**

MySQL

```
SELECT 10 + 5 AS `eredmeny`;
```

- Más adatbázisoknál kötelező
- A MySQL-ben elhagyható

Linkek:

- [SELECT - MySQL dokumentáció](#)

Aritmetikai operátorok

Operátor	Művelet
-	Negatív előjel
*	Szorzás
/	(Valós) Osztás
MOD vagy %	Modulo operátor / Maradék képzés
DIV	Egész osztás
+	Összeadás
-	Kivonás

- A műveletek precedencia (műveleti sorrend) szerinti sorrendben láthatóak

Linkek:

- Aritmetikai műveletek - MySQL dokumentáció

A termékek tábla

id	nev	kategoria	netto	penznem	afa
1	4K TV	tv	499	EUR	0.19
2	Mobil 32GB	mobil	299	EUR	0.19
3	Mobil 128GB	mobil	679	EUR	0.19
4	Olcsó laptop	laptop	269	EUR	0.19
5	Drága laptop	laptop	1729	EUR	0.19
6	Könyv	könyv	NULL	NULL	NULL

Számított mezők

Jelenítsük meg a termékek bruttó árait.

MySQL

```
SELECT
    `nev` AS `termek_nev`,
    `netto` * (1 + afa) AS `brutto`
FROM
    `termekek`;
```

- A lekérdezések során a tábla mezői felhasználhatóak különböző számításokhoz.

Szűrés számított mező alapján

Jelenítsük meg azokat a termékeket, melyek **bruttó ára** több, mint 400 euro

```
SELECT
    `nev` AS `termek_nev`,
    `netto` * (1 + afa) AS `brutto`
FROM `termekek`
WHERE `brutto` > 400;
```

Szintaktikai hiba

#1054 - A(z) 'brutto' oszlop ervenytelen 'where clause'-ben

Hiba!

- Az ANSI SQL szabvány szerint a **WHERE** záradékban nem használható ALIAS

Számított mezők feltételként

Jelenítsük meg azokat a termékeket, melyek **bruttó ára** több, mint 400 euro

MySQL

```
SELECT
    `nev` AS `termek_nev`,
    `netto` * (1 + afa) AS `brutto`
FROM `termekek`
WHERE `netto` * (1 + afa) > 400;
```

`termek_nev`	`brutto`
4K TV	633.73
Mobil 128GB	862.33
Drága laptop	2195.83

- A **WHERE** záradékban alkalmazhatóak számított értékek

Rendezés számított mező alapján

Jelenítsük meg a termékek nevét és a bruttó árat a bruttó szerinti növekvő sorrendben.

```
SELECT `nev`, `netto` * (1 + `afa`) AS 'brutto'  
FROM `termekek`  
ORDER BY 'brutto' ASC;
```

logikai hiba

- A kód le fog futni, de nem a várt eredménnyel.

Rendezés számított mező alapján

`nev`	`brutto`	'brutto'
4K TV	94.80999881029129	brutto
Mobil 32GB	56.80999928712845	brutto
Mobil 128GB	129.00999838113785	brutto
Olcsó laptop	51.10999935865402	brutto
Drága laptop	328.50999587774277	brutto
Könyv	-	brutto

- Az eredmény rendezett, de egy harmadik, mesterségesen generált mező alapján.
- Fontos, hogy az álnév backtick legyen, itt ennek hiányában nem működött a rendezés.

Rendezés számított mező alapján

Jelenítsük meg a termékek nevét és a bruttó árat a bruttó szerinti növekvő sorrendben.

MySQL

```
SELECT `nev`, `netto` * (1 + `afa`) AS `brutto`  
FROM `termekek`  
ORDER BY `netto` * (1 + `afa`) ASC;
```

- A rendezési feltétel kiszámítása elvégezhető az **ORDER BY** záradékban

MySQL

```
SELECT `nev`, `netto` * `afa` AS `brutto`  
FROM `termekek`  
ORDER BY `brutto` ASC;
```

- Az **ORDER BY** záradékban használható a **SELECT**-ben meghatározott álnév

Rendezés számított mező alapján

`nev`	`brutto`
Könyv	<i>NULL</i>
Olcsó laptop	51.10999935865402
Mobil 32GB	56.80999928712845
4K TV	94.80999881029129
Mobil 128GB	129.00999838113785
Drága laptop	328.50999587774277

- Rendezéskor a NULL értékeket mindennél kisebbnek tekinti a MySQL
- Növekvő sorrend esetében az elsők között szerepel
- Csökkenő sorrend esetében az utolsók között szerepel

Tartalom I

- 9 Matematikai függvények
 - Egyszerű matematikai függvények
 - Kerekítés

Tartalom

- 9 Matematikai függvények
 - Egyszerű matematikai függvények
 - Kerekítés

Matematikai függvények

`ABS(x)` $|x|$ abszolút érték

`MOD(x,y)` maradékos osztás

`POW(x,y)` x^y hatványozás

`POWER(x,y)` x^y hatványozás

`SQRT(x)` \sqrt{x} gyök

<https://dev.mysql.com/doc/refman/8.0/en/mathematical-functions.html>

Függvényhasználat: SQRT()

A lekérdezésben használhatunk függvényeket, például a gyök függvényt!

MySQL

```
SELECT SQRT(9) AS `negyzetgyok`  
FROM DUAL;
```

negyzetgyok
3

Linkek:

- [MySQL dokumentáció: Matematikai függvények](#)

PI()

PI()

- Megadja a π (pi) értékét.
- Alapértelmezetten 7 számjegyet jelenít meg
 - ebből 1 számjegy az egész résznek,
 - és 6 számjegy a tört résznek.
- Ennél nagyobb pontossággal tárolja és számol vele.

```
SELECT PI() as `pite` FROM DUAL;
```

MySQL

pite

3.141593

Tartalom

- 9 Matematikai függvények
 - Egyszerű matematikai függvények
 - Kerekítés

Kerekítő függvények

CEILING(x) $\lceil x \rceil$ felső egész rész

CEIL(x) alias a CEILING() függvényre

FLOOR(x) $\lfloor x \rfloor$ alsó egész rész

ROUND(x,n) matematikai kerekítés

TRUNCATE(x,n) nem kerekít, levágja a tizedes jegyeket

Linkek:

- MySQL dokumentáció: Matematikai függvények

Kerekítés ROUND(x,d)

x Kerekítendő érték

d tizedesek száma

A bruttó árat két tizedesre kerekítve jelenítse meg!

MySQL

```
SELECT ROUND(`netto` * (1 + afa) ,2) AS `brutto`  
FROM `termekek`;
```

- Alapvetően a matematikai kerekítést alkalmazza, 5-től felfelé kerekít
- Lebegőpontos számábrázolás esetén bizonyos rendszereken előfordul, hogy a "Round to Even", más néven "Banker's Rounding" módszert alkalmazhatja

Linkek:

- MySQL dokumentáció: Az ROUND() függvény
- Wikipedia: Szimmetrikus kerekítés (Banker's Rounding)

ROUND() példák

MySQL

```
SELECT ROUND(123.4567) as `eredmeny`  
FROM DUAL;
```

eredmeny

123

- Ha a második paraméter 0, vagy nincs, akkor egészre kerekít

MySQL

```
SELECT ROUND(123.4567,1) as `eredmeny`  
FROM DUAL;
```

eredmeny

123.5

- A második paraméter 1, így egy tizedesre kerekít

MySQL

```
SELECT ROUND(123.4567,-1) as `eredmeny`  
FROM DUAL;
```

eredmeny

120

- Mivel a második paraméter -1, így a tizes helyiértékű számra kerekíti

Linkek:

- [MySQL dokumentáció: Az ROUND\(\) függvény](#)

CEIL(), FLOOR(), és ROUND() összehasonlítása

MySQL

```
SELECT CEIL(222.111) as `eredmeny`  
FROM DUAL;
```

eredmeny

223

- A CEIL() függvény visszaadja a **tőle nem kisebb** legkisebb egész számot

MySQL

```
SELECT FLOOR(111.888) as `eredmeny`  
FROM DUAL;
```

eredmeny

111

- A FLOOR() függvény visszaadja a **tőle nem nagyobb** legnagyobb egész számot

MySQL

```
SELECT ROUND(111.888) as `eredmeny`  
FROM DUAL;
```

eredmeny

112

- A ROUND() függvény kerekítést alkalmaz

Linkek:

- [MySQL dokumentáció: Az ROUND\(\) függvény](#)

CEIL(), FLOOR(), és ROUND() negatív számokkal

MySQL

```
SELECT CEIL(-222.111) as `eredmeny`  
FROM DUAL;
```

eredmeny

-222

- A CEIL() függvény visszaadja a **tőle nem kisebb** legkisebb egész számot

MySQL

```
SELECT FLOOR(-111.888) as `eredmeny`  
FROM DUAL;
```

eredmeny

-112

- A FLOOR() függvény visszaadja a **tőle nem nagyobb** legnagyobb egész számot

MySQL

```
SELECT ROUND(-111.888) as `eredmeny`  
FROM DUAL;
```

eredmeny

-112

- A ROUND() függvény kerekítést alkalmaz

Linkek:

Tartalom I

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

CONCAT()

```
CONCAT(str1,str2,...)
```

- Összefűzi az argumentumként kapott értékeket
- A számokat átalakítja szöveggé
- Amennyiben tartalmaz **NULL** értéket, úgy a végeredmény is **NULL** lesz

Linkek:

- MySQL dokumentáció: [CONCAT\(\)](#)

CONCAT() példák

MySQL

```
SELECT CONCAT('A', 'B', 'C', 'D') as `e`  
FROM DUAL;
```

e

ABCD

- Több, mint két argumentum is megadható

MySQL

```
SELECT CONCAT('Hello', ' ', 'World') as `e`  
FROM DUAL;
```

e

Hello World

- A szóköz külön argumentumként lett megadva

MySQL

```
SELECT CONCAT(15, ' cm') as `e`  
FROM DUAL;
```

e

15 cm

- A szóköz a ' cm' értékben található meg

CONCAT() példák

MySQL

```
SELECT  
    CONCAT(`magassag` / 100, ' m') as `magassag_meterben`  
FROM `tanulok`;
```

magassag_meterben
1,72 m
1,83 m
1,85 m
...

- A `magassag` a `tanulok` tábla egyik oszlopa
- A magasság cm-ből m-re át lett számítva
- A szóköz a ' m' értékben található meg

CONCAT() és rendezés

logikai hiba

```
SELECT
    `nev`,
    CONCAT(ROUND(`netto` * (1 + afa)), ' EUR') AS `eur`
FROM `termekek`
WHERE `netto` IS NOT NULL
ORDER BY `eur` ASC;
```

Drága laptop	2058 EUR
Olcsó laptop	320 EUR
Mobil 32GB	356 EUR
4K TV	594 EUR
Mobil 128GB	808 EUR

- A CONCAT miatt a rendezés szövegek alapján történik
- Így akár '2' > '1 000 000' igaz (szöveges összehasonlítás)
- A sorrend hibás lesz

CONCAT() és rendezés

MySQL

```
SELECT
    `nev`,
    CONCAT(ROUND(`netto` * (1 + afa)), ' EUR') AS `eur`
FROM `termekek`
WHERE `netto` IS NOT NULL
ORDER BY ROUND(`netto` * (1 + afa)) ASC;
```

Olcsó laptop	320 EUR
Mobil 32GB	356 EUR
4K TV	594 EUR
Mobil 128GB	808 EUR
Drága laptop	2058 EUR

- Szabvány szerint használhatnánk az alias a rendezésben
- A CONCAT-ben található rész szerint kell rendezni:

ROUND(`netto` * (1 + afa))

CONCAT_WS()

```
CONCAT_WS(separator, str1, str2, ...)
```

- "Concatenate With Separator"
- Összefűzi az argumentumként kapott értékeket
- Az első argumentum az elválasztó karakter
- Az elválasztó karaktert a legvégére nem teszi ki

```
SELECT CONCAT_WS('*', 'alma', 'barack', 'eper') AS `e`  
FROM DUAL;
```

MySQL

e

alma*barack*eper

Linkek:

- [MySQL dokumentáció: CONCAT_WS\(\)](#)

Logikai konstansok

- A TRUE logikai konstans értéke: 1
- A False logikai konstans értéke: 0
- A kis- és nagybetűkre nem érzékeny

MySQL

```
SELECT  
    TRUE, true, True, TrUe, FALSE, false, False, fALSe  
FROM DUAL;
```

TRUE	true	True	TrUe	FALSE	false	False	fALSe
1	1	1	1	0	0	0	0

Linkek:

- [MySQL dokumentáció: Boolean Literals](#)

Szövegek és számok összehasonlítása

MySQL

```
SELECT 2 > 1999999 AS `eredmeny`  
FROM DUAL;
```

0 - FALSE

- Számként összehasonlítva az 1999999 a nagyobb

MySQL

```
SELECT '2' > '1999999' AS `eredmeny`  
FROM DUAL;
```

1 - TRUE

- Szöveggént összehasonlítva a 2 a nagyobb
- Az első karaktert összehasonlítva eldöntötte, hogy '2' > '1'

MySQL

```
SELECT '2' > '2000000' AS `eredmeny`  
FROM DUAL;
```

0 - FALSE

- Az első karaktert összehasonlítva nem állapítható meg a nagyobb, mert '2' = '1'
- Az lesz a nagyobb, amiben még találhatóak további karakterek

Összefűtés és rendezés

logikai hiba

```
SELECT
  `nev`,
  CONCAT(ROUND(`netto`*(1+`afa`),2), `penznem`) AS `brutto`
FROM `termekek`
ORDER BY `brutto` ASC;
```

nev	brutto
Könyv	NULL
Drága laptop	2057.51EUR
Olcsó laptop	320.11EUR
Mobil 32GB	355.81EUR
4K TV	593.81EUR
Mobil 128GB	808.01EUR

- A CONCAT miatt a brutto oszlop értékeit szöveggént hasonlítja össze
- A sorrend nem megfelelő

Összefűtés és rendezés

MySQL

```
SELECT
    `nev`,
    CONCAT(ROUND(`netto`*(1+`afa`),2), `penznem`) AS `brutto`
FROM `termekek`
ORDER BY ROUND(`netto`*(1+`afa`),2) ASC;
```

nev	brutto
Könyv	NULL
Olcsó laptop	320.11EUR
Mobil 32GB	355.81EUR
4K TV	593.81EUR
Mobil 128GB	808.01EUR
Drága laptop	2057.51EUR

- A kerekített értéket látjuk, így célszerű annak megfelelően rendezni
- A CONCAT ugyan szerepel a SELECT-ben, de az ORDER BY záradékban már nem
- A sorrend így már helyes

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

UPPER()

UPPER(str)

- Nagybetűssé alakítja a szöveget (str)
- Alapértelmezetten a latin1 kódolást (cp1252 West European) használja
- Amennyiben a tábla karakterkódolás jól van megadva multibyte karaktereket is jól kezel

MySQL

```
SELECT UPPER('heLLo') FROM dual;
```

e

HELLO

LOWER()

LOWER(str)

- Kisbetűssé alakítja a szöveget (str)
- Alapértelmezetten a latin1 kódolást (cp1252 West European) használja
- Amennyiben a tábla karakterkódolás jól van megadva multibyte karaktereket is jól kezel

MySQL

```
SELECT LOWER('heLLo') FROM dual;
```

e

hello

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - **Hossz**
 - Hossz
 - Keresés
 - Csere
 - Kivágás

LENGTH()

LENGTH(str)

- Megadja a szöveg (str), hosszát **byteokban**
- A multibyte karaktereket **többször** számolja

LENGTH() példák

MySQL

```
SELECT LENGTH('car') as `e`  
FROM DUAL;
```

e

3

- Az egy byteos karakterek (ASCII első 128 karaktere) hossza megegyezik a karaktereinek számával

logikai hiba

```
SELECT LENGTH('autó') as `e`  
FROM DUAL;
```

e

5

- Az "autó" 4 betűs szó, de a hosszú "ó" multibytos karakter, így lesz a végeredmény 5

CHAR_LENGTH()

```
CHAR_LENGTH(str)
```

- Megadja a szöveg (str), hosszát **karakterekben**
- A multibyte karaktereket **egyszer** számolja
- Szinonímák erre a függvényre:
 - **CHARACTER_LENGTH**(str)

CHAR_LENGTH() példák

```
SELECT CHAR_LENGTH('car') as `e`  
FROM DUAL;
```

MySQL

e

3

- Az egy byteos karakterek (ASCII első 128 karaktere) hossza megegyezik a karaktereinek számával

```
SELECT CHAR_LENGTH('autó') as `e`  
FROM DUAL;
```

MySQL

e

4

- Az "autó" 4 betűs szó, amit helyesen megállapított a függvény

LENGTH() és CHAR_LENGTH() összehasonlítása

```
SELECT LENGTH('árvíztűrőtükörfúrógép') as `e`  
FROM DUAL;
```

logikai hiba

e

30

```
SELECT CHAR_LENGTH('árvíztűrőtükörfúrógép') as `e`  
FROM DUAL;
```

MySQL

e

21

- A **LENGTH()** a byteok számát, míg a **CHAR_LENGTH()** a karakterek számát adja meg, így utóbbi a multibyteos karakterek esetén is helyesen állapítja meg a szöveg hosszát.

Tartalom

10 Szöveg függvények

- Összefűzés
- Kis- és nagybetűk
- Hossz
- **Hossz**
- Keresés
- Csere
- Kivágás

FORMAT()

```
FORMAT(X,D[,locale])
```

- Az X számot formázza ezres csoportosítással
- A tizedesek számát a D határozza meg
- A nyelvi beállítás határozza meg,
 - hogy tizedes pontot ('en_US'), vagy
 - hogy tizedes vesszőt ('hu_HU') használjon

MySQL

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

LOCATE()

```
LOCATE(substr, str)
```

vagy

```
LOCATE(substr, str, pos)
```

- Megkeresi a keresett szöveg (substr), a kezdő pozícióját a szövegben (str) a megadott számú (pos) karaktertől kezdve
- Az indexelés 1-től kezdődik
- Ha nem találja meg 0-t ad vissza.
- Az eredeti szöveget nem módosítja
- Szinonímák erre a függvényre:
 - **POSITION**(substr **IN** str)

LOCATE() példák

```
SELECT LOCATE('vár', 'Székesfehérvár') as `e`  
FROM DUAL;
```

MySQL

e

12

- A "vár" szöveg "v" betűje a 12. karakter

```
SELECT LOCATE('Székesfehérvár', 'vár') as `e`  
FROM DUAL;
```

MySQL

e

0

- A "vár" szöveg nem tartalmazza a "Székesfehérvár" szöveget, így az eredmény 0

LOCATE() példák

MySQL

```
SELECT LOCATE('é', 'Székesfehérvár') as `e`  
FROM DUAL;
```

e

3

- Az "é" betű a 3. karakter a szó legelejétől keresve

MySQL

```
SELECT LOCATE('é', 'Székesfehérvár', 3) as `e`  
FROM DUAL;
```

e

3

- Az "é" betű a 3. karakter a szó 3. karakterétől keresve

MySQL

```
SELECT LOCATE('é', 'Székesfehérvár', 4) as `e`  
FROM DUAL;
```

e

10

- Az "é" betű a 10. karakter a szó 4. karakterétől keresve

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

REPLACE()

```
REPLACE(str,from_str,to_str)
```

- Lecseréli a szövegben (str), az összes előfordulását a keresett szövegrésznek (from_str) az új szövegre (to_str)
- Az eredeti szöveget nem módosítja

REPLACE() példák

MySQL

```
SELECT  
  REPLACE('Székesfehérvár', 'é', 'e') as `e`  
FROM DUAL;
```

e

Szekesfehehervár

- Az "é" betű lett lecserélve az "e" betűre

MySQL

```
SELECT  
  REPLACE(REPLACE('Székesfehérvár', 'é', 'e'), 'á', 'a') as `e`  
FROM DUAL;
```

e

Szekesfehervar

- A függvény többszöri egymásba ágyazásával több karakter is lecserélhető

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

SUBSTRING()

`SUBSTRING(str,pos,len)`

- Kivág egy részt a szövegből (`str`), a kezdő pozíciótól (`pos`) kezdve megadott számú (`len`) karaktert
- Az eredeti szöveget nem módosítja
- Szinonímák erre a függvényre:
 - `SUBSTR()`
 - `MID()`

SUBSTRING() példák

```
SELECT SUBSTRING('Székesfehérvár',7) as `e`  
FROM DUAL;
```

MySQL

e

fehérvár

- A 1en elhagyásával a szöveget a pos-tól a legvégéig veszi

```
SELECT SUBSTRING('Székesfehérvár',7,5) as `e`  
FROM DUAL;
```

MySQL

e

fehér

- A 7. karaktertől vesz 5 karaktert

```
SELECT SUBSTRING('Székesfehérvár',-3,3) as `e`  
FROM DUAL;
```

MySQL

e

vár

- Negatív pos esetén hátulról lép vissza, majd a 1en-ben meghatározott karaktert veszi, annak elhagyásával a végéig

Tartalom I

- 11 Dátum és idő függvények
 - Dátum/idő részének kinyerése
 - Aktuális dátum/idő

Tartalom

- 11 Dátum és idő függvények
 - Dátum/idő részének kinyerése
 - Aktuális dátum/idő

Dátum/Idő részének kinyerése

- `DATE()` Megadja a dátum részt egy dátum/időből
- `TIME()` Megadja az idő rész egy dátum/időből
- `YEAR()` Megadja az évet
- `MONTH()` Megadja a hónapot
- `DAY()` Megadja a napot
- `HOUR()` Megadja az órát
- `MINUTE()` Megadja a percet
- `SECOND()` Megadja a másodpercet
- `WEEKDAY()` Megadja, hogy az adott dátum a hét hányadik napja
 - Paramétere lehet egy mező `YEAR(`született`)`,
 - vagy konkrét érték `YEAR('2022-01-12')`

Linkek:

- MySQL dokumentáció: Dátum és idő függvények

A YEAR() függvény használata

Melyik évben született az 1-es azonosítójú tanuló?

MySQL

```
SELECT YEAR(`szul_ido`)  
FROM `tanulok`  
WHERE `id` = 1;
```

- A `szul_ido` a születési dátumokat tartalmazza (pl.: '2003-03-16')
- A `YEAR()` függvény az évet nyeri ki belőle. (pl.: 2003)

A YEAR() függvény használata

A hét melyik napján született az 1-es azonosítójú tanuló?

MySQL

```
SELECT WEEKDAY(`szul_ido`)  
FROM `tanulok`  
WHERE `id` = 1;
```

- A `szul_ido` a születési dátumokat tartalmazza (pl.: '2003-03-16')
- A `WEEKDAY()` függvény a nap sorszámát határozza meg. (pl.: 6)
 - 0 - Hétfő
 - 1 - Kedd
 - ...
 - 6 - vasárnap

Tartalom

- 11 Dátum és idő függvények
 - Dátum/idő részének kinyerése
 - Aktuális dátum/idő

Aktuális dátum és idő

- Aktuális dátum
 - `CURDATE()`
 - `CURRENT_DATE()`
- Aktuális idő
 - `CURTIME()`
 - `CURRENT_TIME()`
- Aktuális dátum és idő
 - `NOW()`

Linkek:

- [MySQL dokumentáció: Dátum és idő függvények](#)

Tartalom I

12 Egyéb hasznos függvények

COALESCE()

```
COALESCE(value,...)
```

- Visszaadja az első **nem NULL** értéket

```
SELECT COALESCE(NULL, NULL, NULL, 'A', 'B') AS `e`  
FROM DUAL;
```

MySQL

e

A

Tartalom I

13 Aggregált (összesítő) függvények

- COUNT
- SUM
- AVG
- MIN/MAX

Összesítő függvények

Az összesítő (aggregált) függvények a meghatározott kifejezésen hajtanak végre különböző műveleteket.

- Alapértelmezetten a NULL értékeket nem veszik számításba
- Gyakran használt összesítő függvények:
 - COUNT()
 - SUM()
 - AVG()
 - MIN()
 - MAX()

Linkek:

- MySQL dokumentáció: Összesítő függvények

Tartalom

13 Aggregált (összesítő) függvények

- COUNT
- SUM
- AVG
- MIN/MAX

A COUNT() függvény

Megszámolja a lekérdezés által visszaadott sorokban a nem NULL értékeket.

- Van lehetőség NULL beleszámítására is
- Amennyiben nincs a feltételeknek megfelelő találat, úgy 0 lesz a függvény kimenete
- A MySQL nem csak a számokat tartalmazó mezőkön értelmezi
- Az eredmény BIGINT típusú lesz

Linkek:

- MySQL dokumentáció: [COUNT\(\)](#)

A termékek tábla

id	nev	kategoria	netto	penznem	afa
1	4K TV	tv	499	EUR	0.19
2	Mobil 32GB	mobil	299	EUR	0.19
3	Mobil 128GB	mobil	679	EUR	0.19
4	Olcsó laptop	laptop	269	EUR	0.19
5	Drága laptop	laptop	1729	EUR	0.19
6	Könyv	könyv	NULL	NULL	NULL

Példa: COUNT(`netto`) példa

MySQL

```
SELECT COUNT(`netto`) AS `darab`  
FROM `termekek`;
```

darab

5

- Ahol a netto értéke **NULL**, azt a sort kihagyja a számításból.

Példa: COUNT() - szöveget tartalmazó oszlopon

MySQL

```
SELECT COUNT(`penznem`) AS `db`  
FROM `termekek`;
```

db

5

- Ahol a penznem értéke **NULL**, azt a sort kihagyja a számításból.

Példa: COUNT(*)

MySQL

```
SELECT COUNT(*) AS `db_csillag`  
FROM `termekek`;
```

db_csillag
6

- A **COUNT(*)** a visszaadott sorok számát számolja meg, így a NULL értékeket is beleveszi a számításába!

Példa: COUNT() - a tábla elsődleges kulcsára alkalmazva

MySQL

```
SELECT COUNT(`id`) AS `darab_id_szerint`  
FROM `termekek`;
```

darab_id_szerint
6

- Az elsődleges kulcs sosem lehet **NULL**
- Érdekes az elsődleges kulcsot megadni paraméterként
- Az elsődleges kulcs gyakran id vagy Azon néven szerepel

Egyedi értékek az összesítő függvényekben

Hány **különböző** kategória található a táblában?

MySQL

```
SELECT COUNT(DISTINCT `kategoria`) AS `db_kategoria`  
FROM `termekek`;
```

db_kategoria
4

- Amennyiben a DISTINCT kulcsszót a COUNT() függvényen belül helyezzük el, úgy az azonos értékeket egyszer veszi csak számításba.

Tartalom

13 Aggregált (összesítő) függvények

- COUNT
- SUM
- AVG
- MIN/MAX

A SUM() függvény

Összeadja a meghatározott kifejezés értékeit.

- A **DISTINCT** megadásával csak az egyedi értékeket összegzi
- Amennyiben a lekérdezés egyetlen sorral sem tér vissza, úgy **NULL** lesz az eredmény
- A **NULL** értékek összege is **NULL** lesz

Linkek:

- MySQL dokumentáció: SUM()

A termékek tábla

id	nev	kategoria	netto	penznem	afa
1	4K TV	tv	499	EUR	0.19
2	Mobil 32GB	mobil	299	EUR	0.19
3	Mobil 128GB	mobil	679	EUR	0.19
4	Olcsó laptop	laptop	269	EUR	0.19
5	Drága laptop	laptop	1729	EUR	0.19
6	Könyv	könyv	NULL	NULL	NULL

Példa: SUM()

Mennyi a termékek *nettó értéke összesen*?

MySQL

```
SELECT SUM(`netto`) AS `ossz`  
FROM `termekek`;
```

OSSZ

3475

- A könyv netto értéke **NULL**
 - Nem adta hozzá az eredményhez
 - Nem lett a végeredmény **NULL**

Példa: SUM() - NULL értékek kihagyásával

MySQL

```
SELECT SUM(`netto`) AS `ossz`  
FROM `termekek`  
WHERE `netto` IS NOT NULL;
```

OSSZ

3475

- A könyv sora kimarad a számításból

Példa: SUM() - NULL értékekkel

MySQL

```
SELECT SUM(`netto`) AS `ossz`  
FROM `termekek`  
WHERE `netto` IS NULL;
```

OSSZ

NULL

- A könyv sorában lesz egyedül **NULL** érték, a végeredmény is **NULL** lett

Példa: SUM() - feltétellel

Mennyibe kerülnek a mobilok?

MySQL

```
SELECT SUM(`netto`) AS `ossz_mobil`  
FROM `termekek`  
WHERE `kategoria` = 'mobil';
```

ossz_mobil

978

- Az összegzés előtt szűr a **WHERE** feltétel alapján
- Csak azokat a sorokat veszi, ahol a **kategoria** értéke "mobil"

Példa: SUM() - számított mező összegzése

Mennyi a termékek **bruttó** értéke?

MySQL

```
SELECT  
    ROUND( SUM(`netto` * (1 + `afa`)) , 2) AS `ossz`  
FROM  
    `termekek`;
```

OSSZ

4135.25

- Kiszámítja a bruttó értéket: ``netto` * (1 + `afa`)`
- A számított értékeket összegzi: `SUM()`
- A végeredményt kerekíti 2 tizedesre: `ROUND()`

Tartalom

13 Aggregált (összesítő) függvények

- COUNT
- SUM
- **AVG**
- MIN/MAX

Az AVG() függvény

Meghatározza a megadott kifejezés átlagát.

- A **DISTINCT** megadásával csak az egyedi értékeket átlagolja
- Amennyiben a lekérdezés egyetlen sorral sem tér vissza, úgy **NULL** lesz az eredmény
- A **NULL** értékek átlaga is **NULL** lesz

Linkek:

- MySQL dokumentáció: [AVG\(\)](#)

A termékek tábla

id	nev	kategoria	netto	penznem	afa
1	4K TV	tv	499	EUR	0.19
2	Mobil 32GB	mobil	299	EUR	0.19
3	Mobil 128GB	mobil	679	EUR	0.19
4	Olcsó laptop	laptop	269	EUR	0.19
5	Drága laptop	laptop	1729	EUR	0.19
6	Könyv	könyv	NULL	NULL	NULL

Példa: AVG()

Mennyi a termékek *nettó árának* az **átlaga**?

MySQL

```
SELECT AVG(`netto`) AS `netto_atlag`  
FROM `termekek`;
```

netto_atlag
695

Példa: AVG()

Mennyi a termékek *bruttó árának* az **átlaga**?

MySQL

```
SELECT  
    AVG(`netto` * ( 1 + `afa` ) ) AS `brutto_atlag`  
FROM  
    `termekek`;
```

brutto_atlag

827.0499983429909

Tartalom

13 Aggregált (összesítő) függvények

- COUNT
- SUM
- AVG
- MIN/MAX

A MIN() függvény

A megadott kifejezés legkisebb értékével tér vissza.

- Szöveggel is működik, az eredmény a karakterkódolástól függhet
- A **DISTINCT** megadásával csak az egyedi értékeket veszi figyelembe, de itt nem számít, mivel csak egy értéket ad úgyis vissza
- Amennyiben a lekérdezés egyetlen sorral sem tér vissza, úgy **NULL** lesz az eredmény
- A **NULL** értékek minimuma is **NULL** lesz

Linkek:

- MySQL dokumentáció: MIN()

A MAX() függvény

A megadott kifejezés legnagyobb értékével tér vissza.

- Szöveggel is működik, az eredmény a karakterkódolástól függhet
- A **DISTINCT** megadásával csak az egyedi értékeket veszi figyelembe, de itt nem számít, mivel csak egy értéket ad úgyis vissza
- Amennyiben a lekérdezés egyetlen sorral sem tér vissza, úgy **NULL** lesz az eredmény
- A **NULL** értékek maximum is **NULL** lesz

Linkek:

- MySQL dokumentáció: MAX()

A termékek tábla

id	nev	kategoria	netto	penznem	afa
1	4K TV	tv	499	EUR	0.19
2	Mobil 32GB	mobil	299	EUR	0.19
3	Mobil 128GB	mobil	679	EUR	0.19
4	Olcsó laptop	laptop	269	EUR	0.19
5	Drága laptop	laptop	1729	EUR	0.19
6	Könyv	könyv	NULL	NULL	NULL

Példa: MIN()

Mennyi a **legolcsóbb** termék *nettó ára*?

```
SELECT MIN(`netto`) AS `min_netto`  
FROM `termekek`;
```

MySQL

min_netto
269

Példa: MIN()

Mi a *neve* annak a terméknek, ami **ABC**-ben az **első**?

```
SELECT MIN(`nev`) AS `min_nev`  
FROM `termekek`;
```

MySQL

min_nev

4K TV

Példa: a MIN() hibás használata

Mi a **neve** a *nettó ár* szerint **legolcsóbb** terméknek?

```
SELECT nev AS `min_nev`, MIN(`netto`)
FROM `termekek`;
```

logikai hiba

#1140 - In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 'pelda.termek.nev'; this is incompatible with sql_mode=only_full_group_by

Hiba!

- Senki sem kérte az árat, hanem csak a nevet

Példa: a MIN() hibás használata

Mi a **neve** a *nettó ár* szerint **legolcsóbb** terméknek?

```
SELECT nev  
FROM `termekek`  
WHERE MIN(`netto`);
```

logikai hiba

```
#1111 - Invalid use of group function
```

Hiba!

- Az összesítő függvények nem használhatóak a **WHERE** záradékban

Példa: a MIN() helyett ORDER BY és LIMIT

Mi a **neve** a *nettó ár* szerint **legolcsóbb** terméknek?

```
SELECT nev AS `min_nev`  
FROM `termekek`  
ORDER BY `netto`  
LIMIT 1;
```

logikai hiba

min_nev

Könyv

- A "legolcsóbb" nettó érték a **NULL** lesz a rendezés szerint
- Előre ki kell szűrni a **NULL** értékeket

Példa: a MIN() helyett ORDER BY és LIMIT

Mi a **neve** a *nettó ár* szerint **legolcsóbb** terméknek?

MySQL

```
SELECT nev AS `min_nev`  
FROM `termekek`  
WHERE `netto` IS NOT NULL  
ORDER BY `netto`  
LIMIT 1;
```

min_nev

Olcsó laptop

Példa: MIN() és MAX() egy lekérdezésben

MySQL

```
SELECT
    MIN(`netto`) AS `min_ar`,
    MAX(`netto`) AS `max_ar`
FROM
    `termekek`;
```

min_ar	max_ar
269	1729

- Egy lekérdezésben több összesítő függvény is szerepelhet
- Nem csak a **MIN()** és a **MAX()**

A termékek tábla

id	nev	kategoria	netto	penznem	afa
1	4K TV	tv	499	EUR	0.19
2	Mobil 32GB	mobil	299	EUR	0.19
3	Mobil 128GB	mobil	679	EUR	0.19
4	Olcsó laptop	laptop	269	EUR	0.19
5	Drága laptop	laptop	1729	EUR	0.19
6	Könyv	könyv	NULL	NULL	NULL

Tartalom I

14 Csoportosítás (GROUP BY)

Csoportosítás

- A GROUP BY
- <https://dev.mysql.com/doc/refman/8.0/en/group-by-functions.html>

Figyelem!

Csoportosításnál csak az szerepelhet a SELECT után, ami vagy szerepel a GROUP BY után, vagy összesítő függvényben van.

GROUP BY példa 1.

Melyik kategóriában hány termék található meg?

vagy

Határozza meg kategóriánként a termékek számát!

MySQL

```
SELECT COUNT(`id`) AS `db`  
FROM `termekek`  
GROUP BY `kategoria`;
```

- Illene megadni a kategóriát is!

db
1
2
2
1

GROUP BY példa 1.

Melyik kategóriában hány termék található meg?

vagy

Határozza meg kategóriánként a termékek számát!

MySQL

```
SELECT `kategoria`, COUNT(`id`) AS `db`  
FROM `termekek`  
GROUP BY `kategoria`;
```

kategoria	db
tv	1
mobil	2
laptop	2
konyv	1

GROUP BY példa 2.

Melyik kategóriában mennyibe kerülnek **átlagosan** a termékek?
vagy

Határozza meg kategóriánként az **átlagos** árat.

MySQL

```
SELECT  AVG(`netto`) AS `atlag`  
FROM    `termekek`  
GROUP BY `kategoria`;
```

- Itt a kategóriát is meg kell jeleníteni, különben nem tudnánk melyikhez tartozik.

GROUP BY példa 2.

Melyik kategóriában mennyibe kerülnek **átlagosan** a termékek?

vagy

Határozza meg kategóriánként az **átlagos** árat.

MySQL

```
SELECT `kategoria`, AVG(`netto`) AS `atlag`  
FROM `termekek`  
GROUP BY `kategoria`;
```

kategoria	atlag
tv	499
mobil	489
laptop	999
konyv	NULL

GROUP BY példa 3.

Melyik kategóriában mennyibe kerül a **legolcsóbb** termék?

vagy

Határozza meg kategóriánként a **legolcsóbb** árat.

```
SELECT `nev`, MIN(`netto`) AS `legolcsobb`  
FROM `termekek`;
```

logikai hiba

nev	legolcsobb
4K TV	269

- **Gyakori hiba, hogy kimarad a GROUP BY!**
- Itt megkeresi a legolcsóbb árat és hozzá írja egy tetszőleges termék nevét.

GROUP BY példa 3. (sql_mode=only_full_group_by)

Melyik kategóriában mennyibe kerül a **legolcsóbb** termék?

vagy

Határozza meg kategóriánként a **legolcsóbb** árat.

```
SELECT `nev`, MIN(`netto`) AS `legolcsobb`  
FROM `termekek`;
```

logikai hiba

#1140 - In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 'pelda.termekek.nev'; this is incompatible with sql_mode=only_full_group_by

Hiba!

- Összesítő lekérdezés csoportosítás nélkül lehetséges, de itt
- Az 1-es számú kifejezés a SELECT-ben (`nev`) nem összesítő függvényben szerepel

GROUP BY példa 4.

Melyik kategóriában mennyibe kerül a **legolcsóbb** termék?

vagy

Határozza meg kategóriánként a **legolcsóbb** árat.

```
SELECT `nev`, MIN(`netto`) AS `legolcsobb`
FROM `termekek`
GROUP BY `penznem`;
```

logikai hiba

nev	legolcsobb
Könyv	-
4K TV	269

- Oda kell figyelni, hogy mi alapján csoportosítunk!
- A pénznemben itt csak EUR és NULL szerepel.
- A kategóriát kell megjeleníteni a nev helyett!

GROUP BY példa 4. (sql_mode=only_full_group_by)

Melyik kategóriában mennyibe kerül a **legolcsóbb** termék?

vagy

Határozza meg kategóriánként a **legocslcsóbb** árat.

```
SELECT `nev`, MIN(`netto`) AS `legolcsobb`  
FROM `termekek`  
GROUP BY `penznem`;
```

logikai hiba

#1055 - Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'pelda.termek.nev' which is not functionally dependent on columns in GROUP BY clause; this is incompatible with sql_mode=only_full_group_by

Hiba!

- Az 1-es számú kifejezés a SELECT-ben (``nev``)
 - nem szerepel a GROUP BY záradékban
 - nem összesítő függvényben szerepel

GROUP BY példa 4.

Melyik kategóriában mennyibe kerül a **legolcsóbb** termék?

vagy

Határozza meg kategóriánként a **legolcsóbb** árat.

MySQL

```
SELECT `kategoria`, MIN(`netto`) AS `legolcsobb`  
FROM `termekek`  
GROUP BY `kategoria`;
```

kategoria	legolcsobb
tv	499
mobil	299
laptop	269
könyv	-

GROUP BY példa 5.

Melyik kategóriában mennyibe kerül a **legdrágább** termék?

vagy

Határozza meg kategóriánként az **legdrágább** árat.

MySQL

```
SELECT `kategoria`, MAX(`netto`) AS `legdragabb`  
FROM `termekek`  
GROUP BY `kategoria`;
```

kategoria	legdragabb
könyv	-
laptop	1729
mobil	679
tv	499

GROUP BY példa 6.

**Melyik kategóriában mennyibe kerül a legdrágább termék?
Ahol nincs ár az ne jelenjen meg!**

MySQL

```
SELECT `kategoria`, MAX(`netto`) AS `legdragabb`  
FROM `termekek`  
WHERE `netto` IS NOT NULL  
GROUP BY `kategoria`;
```

kategoria	legdragabb
laptop	1729
mobil	679
tv	499

GROUP BY példa 7.

Melyik kategóriában mennyibe kerül a **legdrágább** termék?
Ahol nincs ár az ne jelenjen meg! **A bruttó ár** jelenjen meg!

MySQL

```
SELECT
    `kategoria`,
    MAX(`netto` * (1 + `afa`) ) AS `max_brutto`
FROM `termekek`
WHERE `netto` IS NOT NULL`
GROUP BY `kategoria`;
```

kategoria	max_brutto
laptop	1729
mobil	679
tv	499

GROUP BY példa 8.

Melyik kategóriában mennyibe kerül a **legdrágább** termék? Ahol nincs ár az ne jelenjen meg! A bruttó ár jelenjen meg! Az adatok **ár szerint növekvő** sorrendben legyenek rendezettek!

MySQL

```
SELECT
    `kategoria`,
    MAX(`netto` * (1 + `afa`) ) AS `max_brutto`
FROM `termekek`
WHERE `netto` IS NOT NULL
GROUP BY `kategoria`
ORDER BY MAX( `netto` * (1 + `afa`) ) ASC;
```

GROUP BY példa 8. eredménye

Melyik kategóriában mennyibe kerül a **legdrágább** termék? Ahol nincs ár az ne jelenjen meg! A bruttó ár jelenjen meg! Az adatok **ár szerint növekvő** sorrendben legyenek rendezettek!

kategoria	max_brutto
tv	593.8099988102913
mobil	808.0099983811378
laptop	2057.5099958777428

GROUP BY példa 9.

Melyik kategóriában mennyibe kerül a **legdrágább** termék? Ahol nincs ár az ne jelenjen meg! A bruttó ár jelenjen meg! Az adatok **ár szerint csökkenő** sorrendben legyenek rendezettek!

MySQL

```
SELECT
    `kategoria`,
    MAX( `netto` * (1 + afa) ) AS `max_brutto`
FROM `termekek`
WHERE `netto` IS NOT NULL
GROUP BY `kategoria`
ORDER BY `max_brutto` DESC;
```

- MySQL-ben használhatjuk az álnevet rendezéshez.

GROUP BY példa 9. eredménye

Melyik kategóriában mennyibe kerül a **legdrágább** termék? Ahol nincs ár az ne jelenjen meg! A bruttó ár jelenjen meg! Az adatok **ár szerint csökkenő** sorrendben legyenek rendezettek!

kategoria	max_brutto
laptop	2057.5099958777428
mobil	808.0099983811378
tv	593.8099988102913

A termékek tábla

id	nev	kategoria	netto	penznem	afa
1	4K TV	tv	499	EUR	0.19
2	Mobil 32GB	mobil	299	EUR	0.19
3	Mobil 128GB	mobil	679	EUR	0.19
4	Olcsó laptop	laptop	269	EUR	0.19
5	Drága laptop	laptop	1729	EUR	0.19
6	Könyv	könyv	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>

Tartalom I

15 Feltételek csoportosított mezőre (HAVING)

HAVING példa 1.

Melyik kategóriában mennyibe kerül a **legdrágább** termék?

```
SELECT `kategoria`, MAX(`netto`) AS legdragabb  
FROM `termekek`  
GROUP BY `kategoria`  
HAVING MAX(`netto`) is not null;
```

logikai hiba

kategoria	legdragabb
laptop	1729
mobil	679
tv	499

- Ez nem az igazi! Itt a WHERE is elég lett volna!

HAVING példa 2.

Jelenítse meg a legalább két terméket tartalmazó kategóriákat!

MySQL

```
SELECT `kategoria`, COUNT(`id`) as `db`  
FROM `termekek`  
GROUP BY `kategoria`  
HAVING `db` >= 2;
```

kategoria	db
laptop	2
mobil	2

- Az nem volt kérdés, hogy hány termék van a kategóriában!

HAVING példa 3.

Jelenítse meg a legalább két kategóriát tartalmazó termékeket!

MySQL

```
SELECT `kategoria`  
FROM `termek`  
GROUP BY `kategoria`  
HAVING COUNT(`id`) >= 2;
```

kategoria

laptop

mobil

Tartalom I

16 Többtáblás lekérdezések

- Descartes szorzat
- INNER JOIN
- OUTER JOIN

Tartalom

16 Töbبتáblás lekrdézések

- Descartes szorzat
- INNER JOIN
- OUTER JOIN

Példa: a felhasználó és a cikk tábla

A két tábla az alábbi adatokat tartalmazza:

felhasznalo	
id	nev
1	Norbi
2	Bea
3	Helga

cikk		
id	felhasznalo_id	cim
1	1	Első cikk
2	3	Új motorom
3	3	Hogyan lettem videós
4	1	Új nap kezdődik

Példa: A descartes szorzat eredménye. (*felhasznalo* \times *cikk*)

id	nev	id	felhasznalo_id	cim
1	Norbi	1	1	Első cikk
1	Norbi	2	3	Új motorom
1	Norbi	3	3	Hogyan lettem videós
1	Norbi	4	1	Új nap kezdődik
2	Bea	1	1	Első cikk
2	Bea	2	3	Új motorom
2	Bea	3	3	Hogyan lettem videós
2	Bea	4	1	Új nap kezdődik
3	Helga	1	1	Első cikk
3	Helga	2	3	Új motorom
3	Helga	3	3	Hogyan lettem videós
3	Helga	4	1	Új nap kezdődik

Descartes szorzat (direkt szorzat) $A \times B$

Amennyiben a FROM után több táblát is megadunk vesszővel, akkor a lekérdezés során a táblák descartes szorzatát kapjuk.

```
SELECT * FROM `felhasznalo`, `cikk`;
```

MySQL

Probléma: Több hamis sor is keletkezett.

Valós adatok kinyerése a descartes szorzatból

MySQL

```
SELECT * FROM `felhasznalo`, `cikk`  
WHERE `felhasznalo`.`id` = `cikk`.`felhasznalo_id`;
```

id	nev	id	felhasznalo_id	cim
1	Norbi	1	1	Első cikk
1	Norbi	4	1	Új nap kezdődik
3	Helga	2	3	Új motorom
3	Helga	3	3	Hogyan lettem videós

Figyelem!

A **felhasznalo.id** (ponttal) és a **felhasznalo_id** (aláhúzással) nem összekeverendő!

Tartalom

16 Többtáblás lekérdezések

- Descartes szorzat
- **INNER JOIN**
- OUTER JOIN

INNER JOIN (2 tábla)

Két táblás lekérdezés:

```
SELECT * FROM `t1`  
    [INNER] JOIN `t2`  
        ON `t1`.`id` = `t2`.`t1_id`;
```

MySQL

```
SELECT * FROM `felhasznalo`  
    INNER JOIN `cikk`  
        ON `felhasznalo`.`id` = `cikk`.`felhasznalo_id`;
```

Figyelem!

Belső összekapcsoláskor csak azok a sorok jelennek meg, ahol van összeköthető adat a két táblában!

INNER JOIN (3 tábla)

Három táblás lekérdezés

```
SELECT * FROM `t1`  
    [INNER] JOIN `t2`  
        ON `t1`.`id` = `t2`.`t1_id`;  
    [INNER] JOIN `t3`  
        ON `t2`.`id` = `t3`.`t2_id`;
```

Tartalom

16 Töbبتáblás lekrdézések

- Descartes szorzat
- INNER JOIN
- OUTER JOIN

INNER JOIN vs. OUTER JOIN

A külső és belső összekapcsolások között annyi a különbség, hogy amíg a belső összekapcsolásnál csak azok a sorok jelennek meg, ahol mind a két táblában van összeköthető adat, addig a külső összekapcsolásnál elég ha valamelyik oldalon van adat.

Ez persze függ attól, hogy LEFT, RIGHT vagy FULL JOINról van szó.

Azon sorokhoz, melyekhez nem lehet adatot találni a másik táblában, ott a hiányzó részek NULL értékekkel lesz kitöltve.

A MySQL **nem támogatja** a FULL OUTER JOIN-t.

LEFT OUTER JOIN

A **t1** tábla **minden** sorát megjeleníti, ahol tudott hozzá adatot találni a t2-ből ott megjelenik, ahol nem, ott NULL értékek lesznek.

```
SELECT * FROM t1
  LEFT [OUTER] JOIN
    t2 ON t1.id = t2.t1_id;
```

Példa 1: LEFT OUTER JOIN

Ki nem írt még cikket a blogra?

MySQL

```
SELECT *  
FROM  
  `felhasznalo`  
    LEFT OUTER JOIN `cikk`  
      ON `felhasznalo`.`id` = `cikk`.`felhasznalo_id`  
WHERE `cikk`.`felhasznalo_id` IS NULL;
```

Példa 1: LEFT OUTER JOIN

Ki nem írt még cikket a blogra?

id	nev	id	felhasznalo_id	cim
1	Norbi	1	1	Első cikk
1	Norbi	4	1	Új nap kezdődik
2	Bea	-	-	-
3	Helga	2	3	Új motorom
3	Helga	3	3	Hogyan lettem videós

Bea még nem írt cikket, így a felhasznalo_id és a cim mezőben NULL értékek szerepelnek a lekérdezés eredményében.

nev

Bea

RIGHT OUTER JOIN

A **t2** tábla **minden** sorát megjeleníti, ahol tudott hozzá adatot találni a **t1**-ből ott megjelenik, ahol nem, ott NULL értékek lesznek.

```
SELECT * FROM t1  
RIGHT OUTER JOIN t2 ON t1.id = t2.t1_id;
```

Példa: RIGHT OUTER JOIN

Melyik felhasználó nem írt még cikket a blogra?

MySQL

```
SELECT *  
FROM  
    `cikk`  
    RIGHT OUTER JOIN `felhasznalo`  
        ON `felhasznalo`.`id` = `cikk`.`felhasznalo_id`  
WHERE `cikk`.`felhasznalo_id` IS NULL;
```

Példa: RIGHT OUTER JOIN

Ki nem írt még cikket a blogra?

id	felhasznalo_id	cim	id	nev
1	1	Első cikk	1	Norbi
4	1	Új nap kezdődik	1	Norbi
-	-	-	2	Bea
2	3	Új motorom	3	Helga
3	3	Hogyan lettem videós	3	Helga

Bea még nem írt cikket, így a felhasznalo_id és a cim mezőben NULL értékek szerepelnek a lekérdezés eredményében.

nev

Bea

Férj és feleség táblák

ferj		
id	nev	felesege
1	Tamás	-
2	Laci	3
3	Peti	1

feleseg		
id	nev	ferje
1	Andrea	3
2	Emese	-
3	Nóra	2

Feltételezzük, a monogám kapcsolati viszonyt, így a táblák között 1:1 kapcsolat áll fenn.

Figyelem!

Az itt látható táblák tervezése nem megfelelő, de a bemutatni kívánt anyagrészt megértését elősegíti. Senki se próbálja ki otthon!

feleseg LEFT OUTER JOIN ferj

MySQL

```
SELECT * FROM `feleseg`  
  LEFT OUTER JOIN `ferj`  
    ON `feleseg`.`ferje` = `ferj`.`id`;
```

id	nev	ferje	id	nev	felesege
1	Andrea	3	3	Peti	1
2	Emese	-	-	-	-
3	Nóra	2	2	Laci	3

Az összes feleség felsorolásra kerül, még az is, akinek nincs férje, csak utóbbinál NULL értékek szerepelnek a férj helyén.

ferj LEFT OUTER JOIN feleseg

MySQL

```
SELECT * FROM `ferj`  
  LEFT OUTER JOIN `felesege`  
    ON `ferj`.`felesege` = `felesege`.`id`;
```

id	nev	felesege	id	nev	ferje
1	Tamás	-	-	-	-
2	Laci	3	3	Nóra	2
3	Peti	1	1	Andrea	3

Az összes férj felsorolásra kerül, még az is, akinek nincs felesége, csak utóbbinál NULL értékek szerepelnek a feleség helyén.

feleseg RIGHT OUTER JOIN ferj

MySQL

```
SELECT * FROM `feleseg`  
  RIGHT OUTER JOIN `ferj`  
    ON `feleseg`.`ferje` = `ferj`.`id` ;
```

id	nev	ferje	id	nev	felesege
-	-	-	1	Tamás	-
3	Nóra	2	2	Laci	3
1	Andrea	3	3	Peti	1

Az összes férj megjelenik, még az is, akinek nincs felesége. Akinek van felesége, annak a felesége is megjelenik, akinek nincs, ott NULL értékek szerepelnek..

ferj RIGHT OUTER JOIN feleseg

MySQL

```
SELECT * FROM `ferj`  
  RIGHT OUTER JOIN `felesege`  
    ON `ferj`.`felesege` = `felesege`.`id`;
```

id	nev	felesege	id	nev	ferje
3	Peti	1	1	Andrea	3
-	-	-	2	Emese	-
2	Laci	3	3	Nóra	2

Az összes feleség megjelenik, még az is, akinek nincs férje. Akinek van férje, annak a férje is megjelenik, akinek nincs, ott NULL értékek szerepelnek..