Amennyiben a **DISTINCT** opciót alkalmazzuk, akkor minden azonos előfordulás csak egyszer jelenik meg az eredménytáblában. A következőkben nézzünk néhány példát az egyszerű lekérdező parancsok alkalmazására:

1. Példa

Dolgozó tábla:

A dolgozó törzsszáma	A dolgozó Neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése
T234578	Kiss István	Eger	1968. 12. 11.	120000
T456734	Nagy József	Budapest	1972. 01. 30.	150000
T429877	Kovács János	Szeged	1967. 05. 12.	120000

A teljes tábla lekérdezése a következő paranccsal történhet:

```
SELECT * FROM Dolgozó
```

2. Példa

Tegyük fel, hogy szeretnénk lekérdezni az egyes dolgozók nevét a fizetésükkel együtt. A megfelelő parancs a következő:

```
SELECT 'A dolgozó neve', 'A dolgozó fizetése' FROM
Dolgozó
```

A keletkezett eredménytábla a következőképpen néz ki:

A dolgozó neve	A dolgozó fizetése
Kiss István	120000
Nagy József	150000
Kovács János	120000

3. Példa

Készítsünk olyan eredménytáblát, amely tartalmazza a dolgozók nevét, fizetését, valamint a 10%-kal megemelt fizetéseket.

Az új oszlop neve legye **A dolgozó emelt fizetése.** A feladatot az alábbi paranccsal oldhatjuk meg:

```
SELECT 'A dolgozó neve', 'A dolgozó fizetése', 1.1*'A
dolgozó fizetése' AS 'A dolgozó emelt fizetése' FROM
Dolgozó
```

A keletkezett eredménytábla a következő:

A dolgozó neve	A dolgozó fizetése	A dolgozó emelt fizetése
Kiss István	120000	132000
Nagy József	150000	165000
Kovács János	120000	132000

4. Példa

Jelenítsük meg a dolgozók fizetését úgy, hogy a listában ne legyen két azonos érték. A feladatot az alábbi paranccsal oldhatjuk meg:

```
SELECT DISTINCT 'A dolgozó fizetése' FROM Dolgozó
```

A keletkezett eredménytábla a következő:

A dolgozó fizetése
120000
150000

5. Példa

Készítsünk olyan táblázatot, amely megadja a vállalat dolgozóinak számát, az összes és az átlagos fizetéseket.

Az oszlopok nevei legyenek rendre: **A dolgozók száma, A dolgozók összes fizetése, A dolgozók átlagos fizetése.** A feladatot a következő módon oldhatjuk meg:

```
SELECT COUNT('A dolgozó neve') AS 'A dolgozók száma'
, SUM('A dolgozó fizetése') AS 'A dolgozók összes
fizetése', AVG('A dolgozó fizetése') AS 'A dolgozók
átlagos fizetése' FROM Dolgozó
```

A keletkezett eredménytábla a következő:

A dolgozók száma	A dolgozók összes fizetése	A dolgozók átlagos fizetése
3	390000	130000

6. Példa

Készítsünk olyan táblázatot, amely megadja a vállalat dolgozóinak nevét és születési évét. A születési évet tartalmazó oszlop neve legyen A dolgozó születési éve.

A feladatot a következő módon oldhatjuk meg, feltéve, hogy a rendszerünkben létezik a **LEFT(<karaktersorozat>,<darab>)** függvény, amely a <karaktersorozat> paraméterben megadott karaktersorozat bal oldaláról levág a <darab> paraméterben megadott számú karaktert.

```
SELECT 'A dolgozó neve', LEFT('A dolgozó születési ideje',4) AS 'A dolgozó születési éve' FROM Dolgozó
```

A keletkezett eredménytábla a következő:

A dolgozó neve	A dolgozó születési éve
Kiss István	1968
Nagy József	1972
Kovács János	1967

Kiválasztó lekérdezések

A kiválasztást végrehajtó parancsnál a FROM után a következő szintaxisnak megfelelően adhatjuk meg az alparancsot:

[WHERE <feltétel>]

A feltételben operanduszok és operátorok szerepelhetnek.
Az operátorok összehasonlító (<, >, <=, >=, <>),
aritmetikai (+, -, *, /) és
logikai műveletek (AND, OR, NOT),
míg az operanduszok lehetnek **konstansok, reláció attribútumok, azaz oszlopnevek, illetve függvényhivatkozások.**

A műveletekre érvényesek a szokásos **precedencia** szabályok, amelyeket természetesen zárójelezéssel felülbírálhatunk. Ügyelnünk kell arra, hogy a kifejezésnek mindig logikai értéket kell szolgáltatni, ugyanis az eredménytáblába azok az előfordulások fognak bekerülni,
amelyekre a megadott kifejezés igaz (TRUE) értéket szolgáltat.

Létezik néhány predikátum függvény

Ezek közül az első a BETWEEN predikátum függvény, amely a következőképpen használható:

[<oszlopkifejezés> BETWEEN <alsóérték> AND <felsőérték>]

Az **<alsóérték>**, illetve **<felsőérték>** valamely ismert elemi típusnak (numerikus, dátum) megfelelő konstans,
a **<oszlopkifejezés>** ugyanilyen típusú kifejezés, amely oszlopnevekből van képezve.
Eredményként azok a rekordok fognak eleget tenni a feltételnek, amelyekre a kifejezés értéke a két konstans közé esik.
Ezért a feltételnek akkor van értelme, ha az <alsóérték> paraméterben megadott konstans kisebb, mint a <felsőérték> paraméterben megadott.

A következő az **IN predikátum,** amely a következőképpen használható:

[<oszlopkifejezés> [NOT] IN <értéklista>]

A kifejezés hatására annak vizsgálata történik meg, hogy az <oszlopkifejezés> értéke szerepel-e a megadott értéklistában, vagy nem. Amennyiben szerepel a kifejezés értéke igaz lesz. Amennyiben használjuk a NOT kulcsszót, a kifejezés akkor lesz igaz, ha az értéke nem szerepel a listában. Az <értéklista> paraméterben tehát a kifejezés típusának megfelelő értékeket kell vesszővel elválasztva felsorolni.
A harmadik fajta predikátum karakterlánc típusú kifejezésre alkalmazható. Általános formája az alábbi:

[<oszlopkifejezés> LIKE <karakterlánc>]

A <karakterlánc> konstansban idézőjelek között adhatunk meg karaktersorozatot.
A karaktersorozatban **két karakternek speciális jelentése van, ezek a % illetve az _ jelek.**
Az <oszlopkifejezés> paraméternek karakteres értéket kell szolgáltatni, amely összehasonlításra kerül a konstanssal. Amennyiben a kettő megegyezik, a feltétel igaz lesz. Ha a konstansban a % jelet használjuk, akkor a két karakterláncnak csak eddig a jelig kell egyezni ahhoz, hogy a feltétel igaz legyen. Ennek megfelelően a % jelet csak egyszer kell használnunk. Az _ jelet

többször is alkalmazhatjuk, ilyenkor az összehasonlításnál a megadott pozíción bármilyen jel szerepel, azon a helyen az egyezésnek fog számítani.

7. Példa

Készítsünk olyan táblázatot, amely megadja a vállalat azon dolgozóinak nevét és fizetését, akik 150000 Ft felett keresnek!

A feladatot a következő módon oldhatjuk meg:

```
SELECT 'A dolgozó neve', 'A dolgozó fizetése' FROM
Dolgozó WHERE 'A Dolgozó fizetése' >= 150000
```

A keletkezett eredménytábla a következő:

A dolgozó neve	A dolgozó fizetése
Nagy József	150000

8. Példa

Készítsünk olyan táblázatot, amely megadja a vállalat azon dolgozóinak nevét és fizetését, akik 1960.01.01-e után születtek, és a fizetésük 100000 Ft felett van!

A feladatot a következő módon oldhatjuk meg:

```
SELECT 'A dolgozó neve', 'A dolgozó fizetése' FROM
Dolgozó WHERE 'A dolgozó születési ideje' >
{1960.01.01.} AND 'A dolgozó fizetése' >= 100000
```

A keletkezett eredménytábla a következő:

A dolgozó neve	A dolgozó fizetése
Kiss István	120000
Nagy József	150000
Kovács János	120000

9. Példa

Készítsünk olyan táblázatot, amely megadja a vállalat azon dolgozóinak nevét és fizetését, akiknek a fizetése 100000 és 120000 Ft közé esik!

A feladatot a következő módon oldhatjuk meg:

```
SELECT 'A dolgozó neve', 'A dolgozó fizetése' FROM
Dolgozó WHERE 'A dolgozó fizetése' BETWEEN 100000 AND
120000
```

A feladatot megoldhatjuk másképpen is:

```
SELECT 'A dolgozó neve', 'A dolgozó fizetése' FROM
Dolgozó WHERE 'A dolgozó fizetése' >= 100000 AND 'A
dolgozó fizetése' <= 120000
```

A keletkezett eredménytábla a következő:

A dolgozó neve	A dolgozó fizetése
Kiss István	120000
Kovács János	120000

10.Példa

Készítsünk olyan táblázatot, amely megadja a vállalat azon dolgozóinak nevét és fizetését, akiknek a fizetése 100000 vagy 150000 Ft!

A feladatot a következő módon oldhatjuk meg:

```
SELECT 'A dolgozó neve', 'A dolgozó fizetése' FROM
Dolgozó WHERE 'A dolgozó fizetése' IN (100000,150000)
```

A keletkezett eredménytábla a következő:

A dolgozó neve	A dolgozó fizetése
Nagy József	150000

11.Példa

Készítsünk olyan táblázatot, amely megadja a vállalat azon dolgozóinak nevét és születési helyét, akik Egerben, Szegeden vagy Debrecenben születettek!

A feladatot a következő módon oldhatjuk meg:

```
SELECT 'A dolgozó neve', 'A dolgozó születési helye'
FROM Dolgozó WHERE 'A dolgozó születési helye' IN
("Eger", "Szeged", "Debrecen")
```

A keletkezett eredménytábla a következő:

A dolgozó neve	A dolgozó születési helye
Kiss István	Eger
Kovács János	Szeged

12.Példa
Készítsünk olyan táblázatot, amely megadja a vállalat Kovács nevű dolgozóinak adatait!

A feladatot a következő módon oldhatjuk meg:

```
SELECT * FROM Dolgozó WHERE 'A dolgozó neve' LIKE
"Kovács%"
```

A keletkezett eredménytábla a következő:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése
T429877	Kovács János	Szeged	1967. 05. 12.	120000

13.Példa
Készítsünk olyan táblázatot, amely megadja a vállalat Kiss vagy Koós vezetéknevű dolgozóinak adatait!
A feladatot a következő módon oldhatjuk meg:

```
SELECT * FROM Dolgozó WHERE 'A dolgozó neve' LIKE
"K__s%"
```

A keletkezett eredménytábla a következő:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése
T234578	Kiss István	Eger	1968. 12. 11.	120000

14.Példa

Készítsünk olyan táblázatot, amely megadja a vállalat T4-gyel kezdődő törzsszámú nem szegedi vagy debreceni születésű, vagy T2-vel kezdődő törzsszámú 1968-ban született dolgozóit!
Tegyük fel, hogy a rendszerünkben egy dátum típusú adat évének lekérdezésére a `YEAR(<Dátumkifejezés>)` függvény használható.

A feladatot a következő módon oldhatjuk meg:

```
SELECT * FROM Dolgozó WHERE ((LEFT('A dolgozó törzsszáma',2) LIKE "T4") AND ('A dolgozó születési helye' NOT IN ("Szeged", "Debrecen"))) OR ((LEFT('A dolgozó törzsszáma',2) LIKE "T2") AND (YEAR('A dolgozó születési ideje') = 1968))
```

A keletkezett eredménytábla a következő:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése
T234578	Kiss István	Eger	1968. 12. 11.	120000
T456734	Nagy József	Budapest	1972. 01. 30.	150000

Csoportosító lekérdezések és rendezések

A csoportosítás azt jelenti, hogy a rekordokat egy adott mező értékei szerint csoportokra bontjuk.

Legtöbbször az adott mezőben azonos értékeket felvevő rekordok kerülnek egy csoportba.

Ezután a csoportokhoz tartozó rekordokra különböző műveleteket hajthatunk végre, például alkalmazhatjuk a már ismert aggregációs függvényeket, esetleg a csoportokra vonatkozóan kiválasztó műveletet alkalmazhatunk.

A csoportosítás a következő utasítással hajtható végre:

```
GROUP BY <oszlopnév>,[<oszlopnév>]...
```

A csoportosítás a megadott oszlopnevek azonos értékei alapján fog történni. Amennyiben több oszlopot adunk meg, akkor az első oszlop azonos értékein belül a második oszlop azonos értékei szerint csoportosít, majd a harmadik szerint, stb. Az egyes mezőkhöz tartozó értékeket a megadás sorrendjében egymás mellé rakja a rendszer, és az így kapott minta alapján csoportosít.
A művelet végrehajtása után minden egyes csoportra egy sor keletkezik az eredménytáblában.

Mivel speciális utasításról van szó, használata során a SELECT parancs egyéb részeire vonatkozóan is megkötéseket kell tennünk.
Így a lekérdezendő oszlopok adataira vonatkozóan mindenképpen alkalmaznunk kell valamilyen aggregáló operátort, vagy ha ezt nem tesszük, akkor az oszlopnak szerepelnie kell a csoportosításban részt vevő oszlopok között, azaz a GROUP BY után.

Az SQL nyelv lehetőséget biztosít arra is, hogy az aggregálással keletkezett adatokra vonatkozóan feltételeket adhassunk meg. Ebben az esetben a WHERE kulcsszó helyett a HAVING szót kell használnunk.

Az alparancs formája az alábbi:

HAVING <csoportfeltétel>

A <csoportfeltétel> paraméterben a hagyományos módon adhatunk meg feltételeket, azzal a különbséggel, hogy

a feltételben szereplő oszlopneveknek tartalmazniuk kell valamilyen aggregáló operátort, és ennek ugyancsak szerepelnie kell a SELECT után.

Rendezés. Az SQL lehetőséget biztosít arra, hogy lekérdezéseink eredményét rendezetten jelenítsük meg.

ORDER BY alparancs. Formája a következő:

ORDER BY <oszlopnév|oszlopsorszám> [ASC|DESC], <oszlopnév|oszlopsorszám> [ASC|DESC]]...

A rendezés a megadott oszlopok szerint történik. Első szempontként az első oszlopot, további szempontként az utána megadott oszlopokat veszi figyelembe.
Az oszlopok kétféleképpen adhatók meg.
Egyrészt hagyományosan a nevükkel, másrészt egy számmal, ami a táblázatban az oszlop sorszáma. A számozás 1-től kezdődik a táblázat fejrészében megadott sorrend szerint.
Lényeges hogy a rendezési szempontként megadott oszlopnak szerepelni kell a SELECT parancs után is.
Fontos még az **ASC** és a **DESC** kulcsszavak jelentése.
Az ASC az **alapértelmezés,** ami azt jelenti, hogy a rendezés növekvő sorrend szerint történik.
Amennyiben a DESC kulcsszót használjuk, akkor a megadott szempontnál a rendezés **csökkenő** lesz. A következőkben nézzünk néhány példát a csoportosító, illetve a rendező SQL parancsok használatára.

15.Példa

Tegyük fel ezúttal, hogy a Dolgozó táblánk az alábbi módon néz ki:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	A dolgozó fizetése
T234578	Kiss István	Eger	1968. 12. 11.	120000
T3443234	Kiss Timót	Eger	1970. 02. 28.	105000
T456734	Nagy József	Budapest	1972. 01. 30.	150000
T768545	Vári Ödön	Budapest	1958. 07. 12.	210000
T429877	Kovács János	Szeged	1967. 05. 12.	120000

Készítsünk olyan listát, amely megadja városonként, hogy az adott városban hány dolgozó született!

Az alábbi parancsot használhatjuk:

```
SELECT 'A dolgozó születési helye' AS 'Születési hely', COUNT('A dolgozó törzsszáma') AS 'A dolgozók száma' FROM Dolgozó GROUP BY 'A dolgozó születési helye'
```

Eredményképpen a következő táblát kapjuk:

Születési hely	A dolgozók száma
Eger	2
Budapest	2
Szeged	1

16.Példa

Készítsünk olyan listát, amely megadja városonként, hogy az adott városban született dolgozóknak mennyi az összes illetve az átlagfizetése!

Az alábbi parancsot használhatjuk: