```
SELECT 'A dolgozó születési helye' AS 'Születési hely', SUM('A dolgozó fizetése') AS 'A dolgozók összes fizetése', AVG('A dolgozó fizetése') AS 'A dolgozók átlagfizetése' FROM Dolgozó GROUP BY 'A dolgozó születési helye'
```

Eredményképpen a következő táblát kapjuk:

Születési hely	A dolgozók összes fizetése	A dolgozók átlagfizetése
Eger	225000	112500
Budapest	360000	180000
Szeged	120000	120000

17.Példa

Készítsünk olyan listát, amely megadja azokat a városokat, amelyekre igaz, hogy az adott városban született dolgozóknak az átlagfizetése legfeljebb 120000 Ft!

Az alábbi parancsot használhatjuk:

```
SELECT 'A dolgozó születési helye' AS 'Születési hely', AVG('A dolgozó fizetése') AS 'A dolgozók átlagfizetése' FROM Dolgozó GROUP BY 'A dolgozó születési helye' HAVING AVG('A dolgozó fizetése') < 120000
```

Eredményképpen a következő táblát kapjuk:

Születési hely	A dolgozók átlagfizetése
Eger	112500
Szeged	120000

18.Példa

Készítsünk olyan listát, amely a dolgozók nevét és fizetését név szerint ábécé sorrendben adja meg!

Az alábbi parancsot használhatjuk:

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

A keletkezett eredménytábla az alábbi:

A dolgozó neve	A dolgozó fizetése
Kiss István	120000
Kiss Timót	105000
Kovács János	120000
Nagy József	150000
Vári Ödön	210000

19.Példa

Készítsünk olyan listát, amely a dolgozók nevét és fizetését a fizetés szerint csökkenő, illetve azonos fizetés esetén név szerint ábécé sorrendben adja meg!

Az alábbi parancsot használhatjuk:

```
SELECT 'A dolgozó neve', 'A dolgozó fizetése' FROM
Dolgozó ORDER BY 'A dolgozó fizetése' DESC, 'A
dolgozó neve'
```

A keletkezett eredménytábla az alábbi:

A dolgozó neve	A dolgozó fizetése
Vári Ödön	210000
Nagy József	150000
Kiss István	120000
Kovács	120000

János	
Kiss Timót	105000

Több táblára vonatkozó lekérdezések

A legtöbb esetben az adatbázis szerkezete olyan, hogy a szükséges információk több táblában találhatók. Különösen igaz ez, ha normalizált relációkkal dolgozunk, hiszen mint láttuk, a normalizálás alaptevékenysége a több relációra bontás. Ilyen esetekben az információk összegyűjtéséhez minden táblára szükségünk van, ezért a lekérdezéseket ki kell terjeszteni úgynevezett ***többtáblás lekérdezésekké.***

Ennek általános formája a következő:

SELECT [ALL|DISTINCT] * | <oszlopnévlista> FROM <táblanév> [<másodnév>][,<táblanév> [<másodnév>]]...

A kérdés az, hogyan kell értelmeznünk azt, amikor a FROM kulcsszó után több tábla neve szerepel. A választ a relációs modell adja.

Ilyen esetben ugyanis **a táblákon a Descartes szorzat relációalgebrai művelet** hajtódik végre, s az eredményt a szorzat reláció adja. A gyakorlatban általában egy Descartes szorzat nem mindenre használható. Egy ilyen szorzat tehát nagyon sok felesleges sort tartalmaz. Képzeljük el, ha az egyik táblában a dolgozók adatai, a másikban a fizetési adatok találhatók. Ekkor ezek Descartes szorzatában minden dolgozóhoz hozzárendelődik az összes kifizetés, az is, ami nem az adott dolgozóhoz tartozik. Ennek így semmi értelme, az ilyen felesleges sorokat ki kell szűrni a szorzatból. Erre használhatjuk a már megismert kiválasztó műveletet, vagyis a **WHERE kulcsszót a megfelelő feltétel megadásával**.

Amennyiben a lekérdezéseinkben több tábla szerepel, akkor előfordulhat, hogy valamely attribútumnak mindkét táblában ugyanaz az azonosítója.

Ilyenkor a megkülönböztetésül az **attribútum név elé oda kell írnunk a megfelelő tábla nevét.** Ugyancsak előfordulhat, hogy egy táblát saját magával szeretnénk összekapcsolni

. Ilyenkor a tábla önmagával való Descartes szorzata képződik. Azonban itt az attribútumok nevei mellett magukat a táblák példányait is el kell különítenünk egymástól.

Ilyenkor kell megadnunk a **<másodnév> paramétert**, amelyben a megfelelő táblához egy megkülönböztető nevet rendelünk. A további hivatkozásokban ezt használhatjuk.

Az SQL nyelv lehetőséget biztosít az összekapcsolás relációalgebrai művelet közvetlen megvalósítására is. Ez azt jelenti, hogy speciális utasítások állnak rendelkezésre, amelyek az összekapcsolás különböző fajtáit adják meg.

A legelső ilyen parancs magát a Descartes szorzatot hozza létre. A parancs megadásánál csak a FROM kulcsszó utáni részt definiáljuk:

<táblanév> CROSS JOIN <táblanév>

A parancs hatására a megadott két tábla Descartes szorzatát képezi a rendszer. Mint tudjuk, sokkal természetesebb a feltételen alapuló összekapcsolás. Ennek formája az alábbi:

<táblanév> JOIN <táblanév> ON <feltétel>

Feltételként tetszőleges, a WHERE parancs után használható feltételt adhatunk meg.

Az összekapcsolások egy speciális fajtáját jelentik az úgynevezett ***külső összekapcsolások.***

Ezek tulajdonképpen annak a problémának a kezelésére használhatók, ami akkor jelentkezik, ha

a **két összekapcsolandó tábla valamely sorához nem tartozik a másik táblából elem**. Ilyenkor az összekapcsolás művelet definíciója alapján ez a sor nem kerül be az eredménytáblába.

A gyakorlatban előfordulhat, hogy ezekre a "lógó" előfordulásokat is szeretnénk szerepeltetni az összekapcsolt relációban. Ilyenkor használhatjuk a **külső összekapcsolásokat.**

Egy külső összekapcsolás abban különbözik a hagyományostól, hogy az eredménybe minden olyan sor is bekerül, amely a másik tábla egyetlen sorához sem kapcsolódik. Egy külső összekapcsolás háromféle módon valósulhat meg. **Az egyik mód az, amikor mindkét tábla "lógó" sorai bekerülnek az eredménybe, a másik kettő pedig amikor csak a bal, illetve a jobboldali tábláé.**

Ennek megfelelően a következő esetek lehetnek:

<táblanév> FULL OUTER JOIN <táblanév> ON <feltétel>

Ebben az esetben mindkét tábla "lógó" sorai bekerülnek az eredmény táblába.

<táblanév> LEFT OUTER JOIN <táblanév> ON <feltétel>

Ebben az esetben csak az első <táblanév> paraméterben megadott tábla "lógó" sorai kerülnek be az eredmény táblába.

<táblanév> RIGHT OUTER JOIN <táblanév> ON <feltétel>

Ebben az esetben pedig a második <táblanév> paraméterben megadott tábla "lógó" sorai kerülnek be az eredmény táblába.

20.Példa

Tekintsük most a Dolgozó és Kifizetés táblákat, amelyek az alábbi módon néznek 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
T343234	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

A kifizetés dátuma	A kifizetett bér	A levont adóelőleg	A dolgozó törzsszáma
2000.01. 02.	76000	45000	T234578
2000.01. 02	69000	43000	T343234
2000.01. 02.	90000	50000	T456734

Készítsünk olyan listát, amely a dolgozók nevét és a számukra kifizetett összeget, adóelőleget, és a kifizetés dátumát név szerint ábécé sorrendben adja meg!

Az alábbi parancsot használhatjuk:

```
SELECT 'A dolgozó neve', 'A kifizetett bér', 'A levont adóelőleg', 'A kifizetés dátuma' FROM Dolgozó, Kifizetés WHERE Dolgozó.'A dolgozó törzsszáma' = Kifizetés.'A dolgozó törzsszáma' ORDER BY 'A dolgozó neve'
```

Ugyanez megvalósítható más módon is:

```
SELECT 'A dolgozó neve', 'A kifizetett bér', 'A levont adóelőleg', 'A kifizetés dátuma' FROM Dolgozó JOIN Kifizetés ON Dolgozó.'A dolgozó törzsszáma' = Kifizetés.'A dolgozó törzsszáma' ORDER BY 'A dolgozó neve'
```

A keletkezett eredménytábla az alábbi:

A dolgozó neve	A kifizetett bér	A levont adóelőleg	A kifizetés dátuma
Kiss István	76000	45000	2000.01.02.
Kiss Timót	69000	43000	2000.01.02
Nagy József	90000	50000	2000.01.02.

21.Példa

Készítsünk olyan listát, amely az előző példában szereplő feladatot úgy oldja meg, hogy a listába azok a dolgozók is belekerülnek, akiknek nem történt kifizetés.

A megoldás a következő:

```
SELECT 'A dolgozó neve', 'A kifizetett bér', 'A levont adóelőleg', 'A kifizetés dátuma' FROM Dolgozó LEFT OUTER JOIN Kifizetés ON Dolgozó.'A dolgozó törzsszáma' = Kifizetés.'A dolgozó törzsszáma' ORDER BY 'A dolgozó neve'
```

A keletkezett eredménytábla az alábbi:

A dolgozó neve	A kifizetett bér	A levont adóelőleg	A kifizetés dátuma
Kiss István	76000	45000	2000.01.02.

Kiss Timót	69000	43000	2000.01.02
Kovács János			
Nagy József	90000	50000	2000.01.02 .
Vári Ödön			

22.Példa

Készítsünk olyan listát, amely az azonos városban született dolgozókat listázza ki páronként, úgy hogy a listán egy pár csak egyszer szerepeljen!

Az alábbi parancsot használhatjuk:

```
SELECT Dolgozó1.'A dolgozó neve', Dolgozó2.'A dolgozó neve' FROM Dolgozó Dolgozó1, Dolgozó Dolgozó2 WHERE Dolgozó1.'A dolgozó neve' <> Dolgozó2.'A dolgozó neve' AND Dolgozó1.'A dolgozó neve' < Dolgozó2.'A dolgozó neve' AND Dolgozó1.'A dolgozó születési helye' = Dolgozó2.'A dolgozó születési helye'
```

Figyeljük meg a másodnevek használatát, valamint azt, hogy a feltételek között a Descartes szorzatból kiszűrjük az egyes rekordoknak az önmagukkal való szorzatát, továbbá a

```
Dolgozó1.'A dolgozó neve' < Dolgozó2.'A dolgozó neve'
```

feltétellel azt, hogy egy pár kétszer szerepeljen a listában.

A keletkezett eredménytábla az alábbi:

Dolgozó1.A dolgozó neve	Dolgozó2.A dolgozó neve
Kiss István	Kiss Timót
Nagy József	Vári Ödön

Beágyazott lekérdezések

Az SQL nyelv lehetőséget biztosít arra, hogy a kiválasztó lekérdezések feltételében is használjunk SQL lekérdező parancsot. Ilyenkor

megkülönböztetünk **külső és belső SELECT parancsot**, és az ilyen jellegű lekérdezéseket

beágyazott lekérdezéseknek nevezzük.

A SELECT parancsok szintaxisa megegyezik a már ismert formákkal, csupán arra kell ügyelnünk, hogy a belső lekérdezésekre vonatkozóan bizonyos megkötéseknek kell teljesülnie. A belső lekérdezés jellege alapján több esetet különböztethetünk meg.
Ezek a következők:

1. **A belső lekérdezés egyetlen értéket szolgáltat. Ez a legegyszerűbb eset, ugyanis ilyenkor minden a hagyományos módon történik, azzal a különbséggel, hogy a feltételként megadott kifejezésben a belső lekérdezés által szolgáltatott értéket használja fel a rendszer.**

2. **A belső lekérdezés egyoszlopos relációt szolgáltat. Ekkor olyan feltételeket adhatunk meg, amelyek a belső lekérdezés által szolgáltatott oszlop adatait használja fel. Ebben az esetben különböző predikátumokat használhatunk.**

Az alábbi három predikátum létezik:

<oszlopkifejezés> [NOT] IN <belső lekérdezés>

Ennél a típusnál az <oszlopkifejezés> paraméterben megadott kifejezés értékéről fogja eldönteni a rendszer, hogy szerepel-e a belső lekérdezés által előállított oszlop adatai között. Ha igen a feltétel értéke igaz, ha nem, akkor hamis lesz. A NOT kulcsszó hatására pontosan fordítva kell értelmezni a feltételt.

A következő predikátumok formája az alábbi:

[NOT] <oszlopkifejezés> <reláció> ALL|ANY <belső lekérdezés>

Ennél a típusnál az <oszlopkifejezés> paraméterben megadott kifejezés értékére vonatkozóan azt fogja vizsgálni a rendszer, hogy a megadott reláció teljesül-e a belső lekérdezés által előállított oszlop adataira. Ha az **ALL** kulcsszót használjuk, a feltétel akkor lesz igaz, ha a reláció az oszlop minden elemére teljesül, míg az **ANY** használatakor elegendő egyetlen elemre teljesülnie. A NOT kulcsszó itt is a feltétel ellentettjét jelenti.

A legáltalánosabb eset az, amikor a belső lekérdezés általános relációt szolgáltat. Ekkor csak kétféle feltételt vizsgálhatunk.

Az egyik az, hogy a keletkezett reláció üres vagy sem. Ehhez a vizsgálathoz az **EXISTS** kulcsszót kell használnunk, amit természetesen a NOT módosíthat. A parancs formája az alábbi:

[NOT] EXISTS <belső lekérdezés>

A másik eset hasonló az 2. pont első részében leírtakhoz, azzal a különbséggel, hogy a feltételben több oszlopkifejezést adhatunk meg, amelyek egyezését külön-külön fogja vizsgálni a rendszer a belső lekérdezés által adott tábla sorainak elemeivel. Természetesen a megadott lista és a lekérdezés eredménytáblája sorainak száma azonos kell hogy legyen. A pontos szintaxis a következő:

(<oszlopkifejezés> [,<oszlopkifejezés>]...) [NOT] IN <belső lekérdezés>

23.Példa

Készítsünk olyan listát, amely a Dolgozó táblából kilistázza azon dolgozók nevét és fizetését, akik az átlag alatt keresnek!

Az alábbi parancsot használhatjuk:

```
SELECT 'A dolgozó neve', 'A dolgozó fizetése' FROM
Dolgozó WHERE 'A dolgozó fizetése' < (SELECT AVG('A
dolgozó fizetése') FROM Dolgozó)
```

A keletkezett eredménytábla az alábbi:

A dolgozó neve	A dolgozó fizetése
Kiss István	120000
Kiss Timót	105000
Kovács János	120000

24.Példa

Készítsünk olyan listát, amely a Dolgozó táblából kilistázza azon dolgozók nevét és fizetését, akik a fizetése a legnagyobb fizetéstől legfeljebb csak 60000 Ft-tal tér el!

Az alábbi parancsot használhatjuk:

```
SELECT 'A dolgozó neve', 'A dolgozó fizetése' FROM
Dolgozó WHERE 'A dolgozó fizetése'+60000 > (SELECT
MAX('A dolgozó fizetése') FROM Dolgozó)
```

A keletkezett eredménytábla az alábbi:

A dolgozó neve	A dolgozó fizetése
Nagy József	150000
Vári Ödön	210000

25.Példa

Az előzőekben látott Kifizetés tábla felhasználásával készítsünk olyan listát, amely a Dolgozó táblából kilistázza azon dolgozók nevét és törzsszámát, akik számára még nem történt kifizetés!

Az alábbi parancsot használhatjuk:

```
SELECT 'A dolgozó neve', 'A dolgozó törzsszáma' FROM
Dolgozó WHERE 'A dolgozó törzsszáma' NOT IN (SELECT
'A dolgozó törzsszáma' FROM Kifizetés)
```

Egy másik lehetséges megoldás:

```
SELECT 'A dolgozó neve', 'A dolgozó törzsszáma' FROM
Dolgozó WHERE NOT EXISTS (SELECT 'A dolgozó
törzsszáma' FROM Kifizetés WHERE Kifizetés.'A dolgozó
törzsszáma' = Dolgozó.'A dolgozó törzsszáma')
```

Figyeljük meg, hogy ez a második megoldás egy olyan speciális esetet foglal magába, amikor a belső lekérdezésben felhasználjuk a külső lekérdezés táblájának egy mezőjét is.

A keletkezett eredménytábla az alábbi:

A dolgozó neve	A dolgozó törzsszáma
Vári Ödön	T768545
Kovács János	T429877