ADATBÁZIS RENDSZEREK LABOR

FELADAT TÍPUS

01	**Egyszerű SELECT, WHERE, ORDER BY**
02	**Egyszerű SELECT, IS NULL, LIKE, ORDER BY**
03	**Dátumos SELECT**
04	**GROUP BY**
05	**GROUP BY, HAVING**
06	**JOIN 2 táblára**
07	**JOIN 2 táblára, GROUP BY, HAVING**
08	**JOIN 3 táblára**
09	**Beágyazott SELECT aggregációs függvényekkel**
10	**Beágyazott SELECT aggregációs függvényekkel és IN-nel**

EASY SELECT, WHERE, ORDER BY

Alap felépítés

SELECT – Ide írjuk, hogy mit szeretnénk látni a lekérdezés eredményeképpen

FROM – Melyik táblában szerepelnek ezek az értékek

WHERE – Feltételeket adhatunk meg (pl. kisebb, mint 5)

ORDER BY – És rendezhetjük a lekért adatokat

Minta feladat:
Listázzuk ki a diák tagok olvasójegyszámát és teljes nevét! Rendezzük név szerint ABC sorrendbe!

Megoldási javaslat:

SELECT – Meg kell jeleníteni a tagok nevét, és olvasó jegyszámát

FROM – Meg kell keresni ez melyik táblában van (KONYVTAR.TAG)

WHERE – Meg kell néznünk, hogy a tag besorolása diák-e

Megoldás:

```
SELECT (tag.vezeteknev ||' '|| tag.keresztnev) AS nev,
tag.olvasojegyszam
FROM konyvtar.tag
WHERE tag.besorolas='diák'
ORDER BY nev ASC;
```

Megjegyzések:

Amikor írjuk a feltételek és a lekérdezésekkor, hogy mit szeretnénk. Pl. tag.keresztnev akkor, ha egy táblát használunk nem muszály a tag.-ot odaírni.

Ha két oszlopot szeretnénk összevonni, akkor a ||' '|| kifejezést kell használni. Az aposztrófok közé kell beilleszteni azt a jelet, ami legyen a két oszlop elválasztó karaktere. Pl. név esetén a space karakter. Ha szeretnénk, akkor adhatunk Aliast az új cellánknak, így a lekérdezés végén ezt írja ki, valamint, a továbbiakban tudjuk is azt használni. Pl. AS nev.

Fontos megjegyezni, hogy az ORACLE Developer nem szereti az idézőjelet, így minden esetben aposztrófot kell használni.

Ha megszeretnénk valamit vizsgálni, akkor meg kell adni azt hogy melyik értéket szeretnénk vizsgálni és hogy mit szeretnénk megnézni. Pl. a besorolas='diák'

Minden egyes lekérdezést le kell zárni pontos vessző segítségével.

EASY SELECT, IS NULL, LIKE, ORDER BY

Alap felépítés

SELECT – Ide írjuk, hogy mit szeretnénk látni a lekérdezés eredményeképpen

FROM – Melyik táblában szerepelnek ezek az értékek

WHERE – Feltételeket adhatunk meg (pl. kisebb, mint 5)

ORDER BY – És rendezhetjük a lekért adatokat

Minta feladat:
Listázzuk ki azon diák tagok olvasójegyszámát és teljes nevét, melyeknek a keresztneve nem 'A' betűvel kezdődik! Rendezzük ABC – szerint növekvő sorrendbe!

Megoldási javaslat:

SELECT – Meg kell jeleníteni a tagok nevét, és olvasó jegyszámát

FROM – Meg kell keresni ez melyik táblában van (KONYVTAR.TAG)

WHERE – Több feltételünk is van. Az egyik, hogy a tag besorolásának diáknak kell lennie, a másik, hogy a keresztnevének 'A' betűvel kell kezdődnie

Megoldás:

```
SELECT (tag.vezeteknev ||' '|| tag.keresztnev) AS nev, tag.olvasojegyszam
FROM konyvtar.tag
WHERE tag.besorolas='diák' and tag.keresztnev NOT LIKE 'A%'
ORDER BY nev ASC;
```

Megjegyzések:

Ha azt szeretnénk megnézni, hogy a szóban szerepel-e egy adott karakter akkor a LIKE kifejezést kell használni.
% - jel megadásakor az adott karakter után bármilyen hosszú karaktersorozat következhet.
_ - estén viszon, csak egy karakter.

Ha egyszerre több feltételt akarunk megvizsgálni, akkor használnunk kell logikai kifejezéseket. Ilyen az

- *AND,* amit akkor használunk, ha mindkét feltételnek teljesülnie kell.
- *OR,* ha csak az egyik feltételnek kell teljesülnie
- *NOT,* ha van olyan aminek nem kell teljesülnie

Ezeket tudjuk kombinálni is.

DÁTUMOS SELECT

Alap felépítés

SELECT – Ide írjuk, hogy mit szeretnénk látni a lekérdezés eredményeképpen

FROM – Melyik táblában szerepelnek ezek az értékek

WHERE – Feltételeket adhatunk meg (pl. kisebb, mint 5)

Minta feladat:
Listázzuk ki azon tagok minden adatát, akik 30 éves kornál fiatalabbak.

Megoldási javaslat:

SELECT – Meg kell jeleníteni a tagok nevét, és olvasó jegyszámát

FROM – Meg kell keresni ez melyik táblában van (KONYVTAR.TAG)

WHERE – Meg kell néznük, kik a 30 éves kornál fiatalabb tagok!

Megoldás:

```
SELECT *

FROM konyvtar.tag

WHERE months_between( sysdate , szuletesi_datum)/12<30;
```

Minta feladat:
Írassa ki az aktuális dátumot!

Megoldás:

```
SELECT sysdate

FROM dual;
```

Megjegyzések:

Ha egy tag minden adatát meg szeretnénk jeleníteni, akkor a * kifejezést kell írnunk a SELECT mezőbe.

Többféle dátumfüggvényt is használunk:

- *months_between:* megadja hány hónap telt el a két paramétereként megadott időpont között.
- *sysdate:* visszaadja az aktuális dátumot
- *last_day:* visszadja a paraméterként megadott dátum utolsó hónapjának, utolsó napját
- *Továbbiak a time függvényeknél*

Ha egy dátumot, vagy műveletet hajtunk végre ami nem igényel könyvtárhoz kapcsolást, akkor a dual-t kell a fromba írni

GROUP BY

Alap felépítés

SELECT – Ide írjuk, hogy mit szeretnénk látni a lekérdezés eredményeképpen

FROM – Melyik táblában szerepelnek ezek az értékek

WHERE – Feltételeket adhatunk meg (pl. kisebb, mint 5)

GROUP BY – Csoportosít megadott feltétel alapján

ORDER BY – És rendezhetjük a lekért adatokat

Minta feladat:
Témánként mennyi az átlagos oldalszám?

Megoldási javaslat:

SELECT – Meg kell jeleníteni, az összes témát, és az átlagos oldalszámot

FROM – Meg kell keresni ez melyik táblában van (KONYVTAR.KONYV)

GROUP BY – Ahhoz, hogy le tudjuk kérni az átlagos oldalszámot, témánként csoportosítani kell

Megoldás:

SELECT konyv.tema, avg(konyv.oldalszam)

FROM konyvtar.konyv

GROUP BY konyv.tema

Megjegyzések:

Az avg függvény fogja megadni az átlagot.

A group by segítségével téma szerint csoportosítunk, tehát innentől kezdve egy téma oldalszámait vizsgáljuk egyszerre.

GROUP BY, HAVING

Alap felépítés

SELECT – Ide írjuk, hogy mit szeretnénk látni a lekérdezés eredményeképpen

FROM – Melyik táblában szerepelnek ezek az értékek

WHERE – Feltételeket adhatunk meg (pl. kisebb, mint 5)

GROUP BY – Csoportosít megadott feltétel alapján

HAVING –Feltételeket adhatunk a csoportosított elemekre

ORDER BY – És rendezhetjük a lekért adatokat

Minta feladat:
Melyik az a téma, amelynek az átlagos oldalszáma kevesebb, mint 400?

Megoldási javaslat:

SELECT – Meg kell jeleníteni, az összes témát, és az átlagos oldalszámot

FROM – Meg kell keresni ez melyik táblában van (KONYVTAR.KONYV)

GROUP BY – Ahhoz, hogy le tudjuk kérni az átlagos oldalszámot, témánként csoportosítani kell

HAVING – Meg kell adni az oldalszámra meg adott csoportosítási feltételt

Megoldás:

SELECT konyv.tema, avg(konyv.oldalszam)

FROM konyvtar.konyv

GROUP BY konyv.tema

HAVING avg(oldalszam)<400;

Megjegyzések:

Az avg függvény fogja megadni az átlagot.

A group by segítségével téma szerint csoportosítunk, tehát innentől kezdve egy téma oldalszámait vizsgáljuk egyszerre.

A HAVING utasítás fogja megnézni a csoportosított függvény eredményére vonatkozó feltételt.

A WHERE-be nem tudunk avg és társai függvényt használni. Ezek használatára a Having alkalmas csak.

JOIN 2 TÁBLÁRA

Alap felépítés

SELECT – Ide írjuk, hogy mit szeretnénk látni a lekérdezés eredményeképpen

FROM – tabla1 … JOIN tabla2 ON tabla1…=tabla2…

WHERE – Feltételeket adhatunk meg (pl. kisebb, mint 5)

ORDER BY - Rendezés

Minta feladat:
Listázzuk a horror témájú könyvekért kapott honoráriumokat.

Megoldási javaslat:

SELECT – Meg kell jeleníteni a könyvek cím, témájukat és a honoráriumukat.

FROM – Meg kell keresni ezek melyik táblában vannak. Feltűnik, hogy ezek az adatok kettő táblában szerepelnek. Ezért össze kell őket kötni

WHERE – Meg kell nézni, hogy könyv témája horror legyen

Megoldás:

```
SELECT konyv.cim, konyv.tema,
konyvszerzo.honorarium

FROM konyvtar.konyv  kk INNER JOIN
konyvtar.konyvszerzo ksz ON
kk.konyv_azon=ksz.konyv_azon

WHERE konyv.tema='horror';
```

Megjegyzések:

Két táblából származó adatok esetén, a lekérdezés eredményének pontos megjelenítéséhez szükséges a két tábla összekapcsolása.

INNER JOIN : Azon rekordokat adja vissza, amelyekre a kapcsoló mezőkben értékek vannak mindkét táblában.

JOIN 2 TÁBLÁRA, GROUP BY, HAVING

Alap felépítés

SELECT – Ide írjuk, hogy mit szeretnénk látni a lekérdezés eredményeképpen

FROM – tabla1 … JOIN tabla2 ON tabla1…=tabla2…

WHERE – Feltételeket adhatunk meg (pl. kisebb, mint 5)

GROUP BY – Csoportosít megadott feltétel alapján

HAVING –Feltételeket adhatunk a csoportosított elemekre

ORDER BY – És rendezhetjük a lekért adatokat

Minta feladat:
Mely szerzőknek nagyobb az összhonoráriuma 1 milliónál?

Megoldási javaslat:

SELECT – Meg kell jeleníteni a szerzők nevét és az összhonoráriumot

FROM – Meg kell keresni ezek melyik táblában vannak. Feltűnik, hogy ezek az adatok kettő táblában szerepelnek. Ezért össze kell őket kötni.

GROUP BY – Csoportosítani kell szerzők szerint

HAVING – Meg kell nézni melyik összhonorarium több mint 1 millió

Megoldás:

```
SELECT sz.keresztnev ||' '|| sz.vezeteknev,
sum(honorarium)
FROM konyvtar.szerzo sz INNER JOIN
konyvtar.konyvszerzo ksz ON
GROUP BY sz.szerzo_azon, sz.keresztnev, sz.vezeteknev
HAVING sum(honorarim)>1 000 000;
```

Megjegyzések:

Ha két táblát összekapcsoltunk és a SELECT-be szeretnénk megjeleníteni az adatokat, akkor azt az adatot be kell írni a GROUP BY-ba.

A HAVING utasításrészben lehet csak használni a sum() függvényt.

JOIN 3 TÁBLÁRA

Alap felépítés

SELECT – Ide írjuk, hogy mit szeretnénk látni a lekérdezés eredményeképpen

FROM – tabla1 … JOIN tabla2 ON tabla1…=tabla2… ... JOIN tabla3 ON tabla1/2…=tabla3…

WHERE – Feltételeket adhatunk meg (pl. kisebb, mint 5)

Minta feladat:
Ki írta a 'Hasznos holmik' című könyvet?

Megoldási javaslat:

SELECT – Meg kell jeleníteni a szerző nevét

FROM – Meg kell keresni ez melyik 3 táblában van

WHERE – Meg kell néznünk, hogy a könyv címe Hasznos Holmik-e

Megoldás:

```
SELECT sz.vezeteknev, sz.keresztnev

FROM konyvtar.konyv kv full outer join
konyvtar.konyvszerzo ko ON kv.konyv_azon =
ko.konyv_azon FULL OUTER JOIN
konyvtar.szerzo kszo ON ko.szerzo_azon =
kszo.szerzo_azon

WHERE cim='Hasznos Holmik';
```

Megjegyzések:

FULL OUTER JOIN: kombinálja a LEFT és RIGHT JOIN-t, azaz az összes rekordot megjeleníti, akár csak az egyik táblában van érték, akár mindkettőben.

BEÁGYAZOTT SELECT AGGREGÁCIÓS FÜGGVÉNYEKKEL

Alap felépítés

SELECT – Ide írjuk, hogy mit szeretnénk látni a lekérdezés eredményeképpen

FROM – Melyik táblában szerepelnek ezek az értékek

WHERE – Feltételeket adhatunk meg (pl. kisebb, mint 5) – ITT LESZ A BEÁGYAZOTT SELECT!

ORDER BY – És rendezhetjük a lekért adatokat

Minta feladat:
Mi a legidősebb szerzőnk neve és születési napja szövegesen leírva (mindegy milyen nyelven)?

Megoldási javaslat:

SELECT – Meg kell jeleníteni a szerző nevét és születésnapját

FROM – Meg kell keresni ez melyik táblában van (KONYVTAR.SZERZO)

WHERE – Meg kell nézni, hogy a szerző születésnapja megegyezik-e a legkorábbi születési időponttal. Ahhoz, hogy megtudjuk mennyi a legkorábbi időpont be kell ágyazni egy SELECTET.

Megoldás:

```
SELECT szerzo.keresztnev, szerzo.vezeteknev,
to_char(szerzo.szuletesnap,'day')

FROM konyvtar.szerzo

WHERE szuletesi_datum = (SELECT
MIN(szuletesi_datum) FROM
KONYVTAR.SZERZO);
```

Megjegyzések:

A beágyazott SELECT segítségével tudunk más eredményeket összehasonlítani azzal, amit keresünk.

EASY SELECT, IS NULL, LIKE, ORDER BY

Alap felépítés

SELECT – Ide írjuk, hogy mit szeretnénk látni a lekérdezés eredményeképpen

FROM – Melyik táblában szerepelnek ezek az értékek

WHERE – Feltételeket adhatunk meg (pl. kisebb, mint 5) – ITT LESZ A BEÁGYAZOTT SELECT!

ORDER BY – És rendezhetjük a lekért adatokat

Minta feladat:
Listázzuk ki a legnagyobb értékkel rendelkező könyv(ek) címét!

Megoldási javaslat:

SELECT – Meg kell jeleníteni a könyv címét

FROM – Meg kell keresni ez melyik táblában van (KONYVTAR.KONYV)

WHERE – Először meg kell nézni, hogy melyik könyv a legdrágább, ezután meg kell nézni, hogy mi a legdrágább könyv azonosítója, végül azonosító segétségével meg kell néznünk melyik az aminek a címe ugyanaz.

Megoldás:

```
SELECT cim

FROM KONYVTAR.KONYV

WHERE KONYV_AZON IN ( SELECT konyv_azon
FROM KONYVTAR.KONYVTARI_KONYV WHERE
ertek = (SELECT MAX(ertek) FROM
KONYVTAR.KONYVTARI_KONYV)
```

Megjegyzések:

Egy lekérdezésben több beágyazott SELECTET is használhatunk. Mindig úgy kell kiindulni, hogy a feladatot belülről kifelé kell megoldani!

FÜGGVÉNYEK

SZÖVEG FÜGGVÉNYEK

Összegzés

||''||, length, instr, decode, nvl, concat, initcap, upper, lower, substr, replace, to_char

Összekapcsolás

arg1 ||'elvlaszto_kar' || arg2 : összekapcsolja arg1-et és arg2-őt a megadott karaktert használva

Hossz:

Length('arg1'): visszaadja arg1-ként megadott szöveg hosszát

INSTR:

INSTR('arg1', 'arg2'): Visszaadja, hogy arg1-ben hanyadik karaktertől szerepel arg2

DECODE:

DECODE('arg1', 'arg2'): Arg1 helyett arg2-őt adja vissza a lekérdezés kiiírások

NVL:

NVL(arg1, arg2): Lekérdezéskor arg1-et kicseréli arg2-re, ha arg1 null értékű

CONCAT:

concat('arg1',arg2): összefűzi arg1-et arg2-vel

INITCAP:

initcap('arg1'): arg1-et nagy kezdőbetűvel jeleníti meg

LOWER:

LOWER('arg1'): arg1-et kisbetűsként jeleníti meg

UPPER

UPPER('arg1'): arg1-et nagybetűsként jeleníti meg

SUBSTR

SUBSTR('arg1',arg2,arg3): az arg1ből vág ki részeket arg2- mettől és arg3 – hány darab alapján

REPLACE

REPLACE('arg1','arg2','arg3'): Megnézi, hogy arg1-ben szerepel-e arg2 és ha igen akkor az arg2 részt kicseréli arg3ra

Dátum -> Szöveg

TO_CHAR(arg1,'arg2'): arg1-dátumot alakítja át az arg2ként megadott formátumba

- yyyy – évszám
- mm – hónap
- dd – nap
- hh24:mi:ss – óra:perc:másodperc

FÜGGVÉNYEK

RENDEZÉSKOR HASZNÁLANDÓ

Összegzés

DESC, ASC, nulls first, nulls last

Növekvő sorrendbe rendezés

arg1 ASC: arg1 szerint rakja növekvő sorrendbe a lekérdezést

Csökkenő sorrend:

arg1 DESC : arg1 szerint rakja csökkenő sorrendbe a lekérdezést

Null értékek elől:

arg1 nulls first: arg1 szerint úgy rendezi az elemeket, hogy a null értékek az elsők

Null értékek hátul:

arg1 nulls last: arg1 szerint úgy rendezi az elemeket, hogy a null értékek az utolsók

FÜGGVÉNYEK

WHERE-BEN HASZNÁLANDÓ

Összegzés

=, and, or, >, <, LIKE, %, _, not, between, in, is null

Egyenlőség vizsgálata:

arg1 = arg2: azokat az elemeket adja vissza, ahol arg1 egyenlő arg2-vel

És kapcsolat:

arg1 and arg2: Ha mindkét feltétel teljesül, akkor igaz

Vagy kapcsolat:

arg1 or arg2: Akkor teljesül, ha legalább az egyik feltétel igaz

Nagyobb:

arg1 > arg2: Ha arg1 nagyobb, mint arg2, akkor igaz

Tagadás:

not (arg1): Akkor teljesül, ha az arg1 feltétel hamis

Két érték között:

between arg1 and arg2: Igaz, ha az érték arg1 és arg2 között van

Tartalmazza:

arg1 in ('arg2','arg3','argn'): Azokat az értékeket adja vissza, amikor arg1 felveszi a zárójelben megadott értékeket

Kisebb:

arg1 < arg2: Ha arg1 kisebb, mint arg2, akkor igaz

Null érték vizsgálata:

arg1 is null: Azokat adja vissza, mikor arg1 null értéket vesz fel

Szerepel-e a karakter:

arg1 LIKE 'arg2':
% - jel megadásakor az adott karakter után bármilyen hosszú karaktersorozat következhet. Pl. cim LIKE 'Re%'
_ - estén viszon, csak egy karakter. Pl. második karatkert nézünk: cim LIKE '_a%'

FÜGGVÉNYEK

MATEMATIKAI FÜGGVÉNYEK

Összegzés

trunc, round, abs, sqrt, min, max, sum, avg, count, sum

TRUNC

TRUNC (arg1, arg2): arg1-et kerekíti arg2-ben megadott értékig. pl TRUNC(15.79, 1) = 15.7

ROUND

ROUND (arg1, arg2): arg1-et kerekíti arg2-ben megadott értékig. pl ROUND(15.79, 1) = 15.7.

Ebben a függvényben nem adhatunk arg2-nek negatív indexet

Abszolútérték:

ABS(arg1): visszaadja arg1 abszolútértékét

Gyökvonás:

SQRT(arg1): visszaadja arg1 gyökét

MIN:

min(arg1): visszaadja a minimumot

MAX:

max(arg1): visszaadja a maximumot

SUM:

sum(arg1): összeadja az értékeket

AVG:

avg(arg1): visszaadja az átlagukat

COUNT:

count(arg1): visszaadja a sorok számát

FÜGGVÉNYEK

TIME FÜGGVÉNYEK

Összegzés

sysdate, datum +-, add_months, months_between, to_date

AKTUÁLIS DÁTUM

sysdate: visszaadja az aktuális dátumot

DÁTUM +- NAP

datum+1 – növeli a dátumot 1 nappal

datum – 7 –csökkenti a datumot 7 nappal

DÁTUM +- HÓNAP

add_months(datum,12) – hozzaad a dátumhoz 12 hónapot

add_months(datum,-12) – elvesz a dátumból 12 hónapot

DÁTUM +-ÓRA

datum – ½ -csökkenti a dátumot 12 órával

datum + ½ - növeli a dátumot 12 órával

Szöveg to Dátum

to_date('2000.01.01', 'yyyy.mm.dd')- meg kell adni a dátumot, és a formátumát

Months_between:

months_between(arg1,arg2): visszaadja hány hónap telt el az arg1 és arg2 ként megadott dátum között

Last_day:

last_day(arg1): *visszadja a paraméterként megadott dátum utolsó hónapjának, utolsó napját*

FÜGGVÉNYEK

SELECT, JOIN

Összegzés

user, distinct

DISTINCT:

distinct: minden elemet csak egyszer jelenít meg

USER:

USER: visszaadja a felhasználónevünket

INNER JOIN

INNER JOIN : Azon rekordokat adja vissza, amelyekre a kapcsoló mezőkben értékek vannak mindkét táblában.

LEFT OUTER JOIN

LEFT OUTER JOIN: Azon rekordokat adja vissza, amelyekre a kapcsoló mezőkben értékek vannak mindkét táblában, valamint azokat is, amelyekre csak az egyik táblában van érték. Például: ha van egy könyv, amire nincs honorárium, az is megjelenik

RIGHT OUTER JOIN

RIGHT OUTER JOIN: Azon rekordokat adja vissza, amelyekre a kapcsoló mezőkben értékek vannak mindkét táblában, valamint azokat is, amelyekre csak az egyik táblában van érték. Például: ha van egy honorárium, ahol nincs könyv akkor is megjelenik.

FULL OUTER JOIN

FULL OUTER JOIN: kombinálja a LEFT és RIGHT JOIN-t, azaz az összes rekordot megjeleníti, akár csak az egyik táblában van érték, akár mindkettőben.