BAZE PODATAKA – PETI I ŠESTI TERMIN VJEŽBI

Agregatne funkcije, ALTER, Složeni primarni ključevi. Složeni SELECT upit. UNION

1. a) Napisati upit koji kao rezultat daje spisak zaposlenih kojima je plata manja od prosječne plate, iz tabele **EMPLOYEES** korisnika **HR**. Zatim u ovaj spisak uključiti i spisak zaposlenih kojima se plata nalazi u opsegu od 50% do 60% od prosječne plate svih zaposlenih (iz tabele **EMPLOYEES**) umanjene za 100. Spisak sadrži sljedeće kolone: ID, IME, PREZIME, PLATA.

b) Upit modifikovati tako da se eliminišu redovi koji se ponavljaju. Novi spisak ne treba da sadrži iznose plata.

2. Tabele **STUDENT**, **SMJEROVI**, **ODSJECI** i **PREDMET** kreirane su na prvom terminu vježbi iz Baza podataka.

- Kolone IND iz tabele **STUDENT**, SS iz tabele **SMJEROVI** i SO iz tabele **ODSJECI** treba modifikovati tako da budu primarni ključevi. Budući da primarni ključ mora biti jedinstven i ne smije imati `null` vrijednost, neophodno je prethodno obrisati svaki postojeći unos koji ova ograničenja ne zadovoljava. Tabeli **PREDMET** potrebno je dodati kolonu **RB** (redni broj) koja sadrži maksimalno četvorocifrene cjelobrojne vrijednosti, koje se ne ponavljaju, pri čemu vrijednost `null` nije dozvoljena. Kolone RB i SO zajedno treba da čine primarni ključ tabele **PREDMET**.
- U tabeli **STUDENT** i tabeli **PREDMET** kolone SS i SO treba da budu spoljni ključevi odgovarajućih kolona iz tabela **SMJEROVI** i **ODSJECI**, respektivno.
- U tabelu **PREDMET** unijeti bar po jedan predmet za svaki smjer.
- Koje predmete sluša student sa indeksom 25/09, na kojem je odsjeku i kojem smjeru?

Posmatra se skup tabela (šema) korisnika **HR**, za koje imate pravo pregleda. Tabele koje se razmatraju su: **EMPLOYEES**, **DEPARTMENTS**, **JOBS**, **COUNTRIES**, **REGIONS, LOCATIONS.** Ispitati strukture ovih tabela. Obratiti pažnju na primarne ključeve, i strane ključeve (foreign key). U sljedećim zadacima potrebno je koristiti podatke iz ovih tabela.

4. Prikazati kolike su zarade po odjeljenjima zaposlenih iz tabele **EMPLOYEES**. Prikazati ID odjeljenja kao i njihov naziv. Prikazati koliko ima zaposlenih u svakom odjeljenju. Zarade zaokružiti na dvije decimale.

5. Iz prethodnog zadatka odvojiti ona odjeljenja koja imaju više od 5 zaposlenih. Rezultate upisati u tabelu **ODJELJENJA** koju je potrebno kreirati za ovu namjenu.

6. Prikazati mjesto zaposlenja radnika, čiji je naziv posla '*President*'.

7. Prikazati sve radnike koji imaju platu iz opsega zarade '*Administration Assistant*'-a.

Posmatra se skup tabela dostupnih preko javnih sinonima **NASTAVNICI**, **JEDINICE**, **ZVANJA**, **PREDMETI**, **OPT** i **SPR**. Ispitati strukture ovih tabela. Obratiti pažnju na primarne ključeve. U sljedećim zadacima potrebno je koristiti podatke iz ovih tabela.

8. Pronaći sve predmete koji se slušaju na studijskom programu čiji je SPRID=10 na jedinici čiji je JID=13, u 5 semestru. Kako se zove jedinica na kojoj postoji spomenuti studijski program?

9. Pronaći imena svih nastavnika koji su angažovani na predmetu '*Adaptivni diskretni sistemi i neuralne mreže'*.

10. Pronaći sve predmete na kojima nastavu izvode nastavnici koji su angažovani na predmetu '*Adaptivni diskretni sistemi i neuralne mreže*'. Zatim za svaki od tih predmeta prikazati fond časova. Ponovljene redove neophodno je eliminisati sa spiska.

11. Pronaći ukupan broj časova koje drži nastavnik čiji je NID=130143.

12. Napisati upit koji će pored naziva svake jedinice ispisati ukupan broj nastavnika sa zvanjem docent.

ČETVRTI TERMIN VJEŽBI – PREDLOG RJEŠENJA

1. a) Pošto je potrebno odvojiti zaposlene koji imaju plate manje od prosječne, neophodno je odrediti iznos prosječne plate. To ćemo postići primjenom agregatne funkcije AVG. Agregatne funkcije primjenjuju se na čitave kolone, a kao rezultat daju *jednu* vrijednost. Jedino u slučaju grupisanja pomoću GROUP BY moguće je dobiti kao rezultat više vrijednosti, pri čemu se za svaku od grupa dobija po jedna vrijednost. Upit `select avg(salary) from hr.employees` kao rezultat će vratiti prosječnu platu zaposlenih iz tabele HR.EMPLOYEES. Da je postojao WHERE dio u prethodnom upitu, agregatna funkcija bi se primijenila na redove kolone koji su rezultat uslova u WHERE dijelu. Osim funkcije AVG, nabrojimo još neke agregatne funkcije: SUM (računa sumu redova u koloni), MAX (pronalazi maksimalni element u koloni na koju se primjenjuje), MIN (pronalazi minimalni element u koloni na koju se primjenjuje), COUNT (broji redove u rezultujućoj koloni). Ako se funkcija COUNT primijeni u kombinaciji sa GROUP BY, ona će vratiti broj redova za svaku grupu.

   Obratiti pažnju da se u glavnom SELECT upitu u WHERE dijelu pojavljuje prethodno navedeni SELECT upit *stavljen u zagradama*, koji vraća vrijednost prosječne zarade u tabeli HR.EMPLOYEES:

```
select employee_id ID, first_name ime, last_name prezime, salary
plata from hr.employees where salary<(select avg(salary) from
hr.employees);
```

   Rezultujućim kolonama su dati alias-i koji su definisani u postavci zadatka: ID, IME, PREZIME i PLATA.

   U drugom dijelu zadatka potrebno je dodati u prethodni spisak i one zaposlene koji imaju zaradu u opsegu od 50% do 60% od prosječne zarade, pri čemu ona mora biti umanjena za 100. Logika SELECT upita biće ista kao i u prethodnom slučaju, osim što se u WHERE dijelu postavljaju odgovarajući uslovi iz zadatka (npr. dio upita `0.5*(select avg(salary) from hr.employees)` daje 50% od prosječne plate):

```
select employee_id ID, first_name ime, last_name prezime, salary-100
plata  from hr.employees where salary>0.5*(select avg(salary) from
hr.employees) and salary<0.6*(select avg(salary));
```

   Sada je rezultate prethodna dva upita potrebno spojiti u jednu rezultujuću tabelu. Za spajanje rezultata SELECT upita koristi se operator <upit1> UNION <upit2>. Rezultat je tabela sastavljena od svih vrsta koje se pojavljuju u rezultatima upita spojenih ovim operatorom. Duplikati će biti automatski obrisani, osim ako se ne koristi varijanta UNION ALL, koja zadržava ponovljene vrste u rezultujućoj tabeli. Da je bilo potrebno naći redove koji se pojavljuju u rezultatima oba upita, <upit1> i <upit2>, koristio bi se operator INTERSECT (u suštini daje presjek dvije ili više rezultujućih tabela). <upit1> MINUS <upit2> će vratiti one vrste koje se pojavljuju u tabeli koja je rezultat upita <upit1> ali se ne pojavljuju u tabeli koja je rezultat upita <upit2>. Obratite pažnju da se spomenuti operatori izvršavaju nakon upita koje oni spajaju. *Važno je napomenuti da upiti koji se spajaju sa ovim operatorima moraju imati ISTI broj kolona i ISTE tipove podataka u kolonama rezultujućih tabela , inače spajanje (kombinovanje) njihovih podataka neće biti moguće.* <upit1> *i* <upit2> *ne stavljati u zagradama.* Krajnje rješenje zadatka je:

```
select employee_id ID, first_name ime, last_name prezime, salary
  plata from hr.employees where salary<(select avg(salary) from
  hr.employees)
union
select employee_id ID, first_name ime, last_name prezime, salary-100
  plata  from hr.employees where salary>0.5*(select avg(salary) from
  hr.employees) and salary<0.6*(select avg(salary) from
  hr.employees);
```

b) U rezultujućem upitu postoje zaposleni koji se ponavljaju, s tim što se u jednom slučaju pojavljuju sa pravim iznosom plate, a u drugom sa iznosom plate koji je umanjen za 100 (iz postavke zadatka). Ovi redovi su zadovoljavali oba SELECT upita koja su spojena operatorom UNION. Da bismo eliminisali ponavljanja, potrebno je kolonu PLATA isključiti iz rezultujućeg upita (ona sadrži vrijednosti koje su različite za istog zaposlenog). Na rezultat dobijen u zadatku pod a) ćemo primijeniti SELECT upit, i korišćenjem ključne riječi DISTINCT eliminisati ponavljanja. U ovom SELECT upitu ćemo koristiti kolone ID, IME i PREZIME. Obratiti pažnju da je upit iz zadatka pod a) *stavljen u zagradama* u WHERE dijelu novog SELECT upita.

```
select distinct id, ime, prezime from (select employee_id ID,
first_name ime, last_name prezime, salary plata from hr.employees
where salary<(select avg(salary) from hr.employees)union select
employee_id ID, first_name ime, last_name prezime, salary-100 plata
from hr.employees where salary>0.5*(select avg(salary) from
hr.employees) and salary<0.6*(select avg(salary) from
hr.employees));
```

Ovdje treba napomenuti da *je neophodno* koristiti nazive kolona ID, IME, PREZIME u novom SELECT upitu, jer su to nazivi u rezultujućoj tabeli dobijenoj pod a). Javila bi se greška da je korišćeno npr. FIRST_NAME IME (gdje bi IME bio alias), umjesto naziva kolone IME ispred ključne riječi FROM u SELECT upitu iz zadatka pod b) (boldovani dio).

2. Budući da su tabele već ranije kreirane, ograničenja iz postavke zadatka ćemo dodavati izmjenom postojećih tabela, pomoću ALTER TABLE. *Važno je napomenuti da je, prilikom dodavanja ovih ograničenja neophodno iz tabela kojima se dodaju eliminisati sve redove u kojima postoje podaci koji ograničenja ne zadovoljavaju. Tako na primjer, ukoliko je potrebno postaviti da je neka kolona primarni ključ, u toj koloni ne smiju postojati ponovljeni podaci ili* `null` *vrijednost (primarni ključ mora biti jedinstven, i ne smije imati null vrijednost!).*
Primarni ključ je kolona (ili skup kolona) koja u nekoj tabeli služi kao jedinstveni identifikator svakog unosa. Prilikom kombinovanja podataka iz više tabela, upravo će nam primarni ključevi omogućiti da ovo kombinovanje bude nedvosmisleno, odnosno, da se kao rezultat upita uvijek dobije baš ono što je traženo. Dodavanje ovog ograničenja postići ćemo korišćenjem naredbe ALTER TABLE.
U opštem slučaju ALTER TABLE služi za redefiniciju tabela/kolona.

```
alter table <naziv_tabele>
add (<kolona> <tip podatka> [default <vrijednost>]
[ogranicenje kolone]);
```

'<, >' označavaju obavezan sadržaj, '[,]' označavaju opcioni sadržaj.
Slična sintaksa je i u slučaju da je postojeću definiciju tabele/kolona neophodno modifikovati:

```
alter table <naziv_tabele>
modify (<kolona> <tip podatka> [default <vrijednost>]
[ogranicenje kolone]);
```

Napomena: U starijim verzijama Oracle-a nije moguće brisanje kolona, a tabeli/koloni nije moguće mijenjati naziv. Počevši od verzije 9i to je moguće postići korišćenjem ALTER TABLE, a u novijim verzijama postoji i klauzula za brisanje kolona, DROP COLUMN.
Dodavanje ograničenja moguće je postići na sljedeći način:

```
alter table <naziv_tabele> add (<ogranicenja>);
```

- Ovu varijantu naredbe ALTER TABLE koristićemo u našem zadatku. Kolona IND u tabeli STUDENT treba da bude primarni ključ.

```
alter table student add(constraint student_ogr1 primary key(ind));
```

U prethodnoj naredbi `student_ogr1` je naziv ograničenja koje se dodaje, dok `primary key(ind)` označava da je primarni ključ dodat koloni IND. Ekvivalentno ovom ćemo uraditi i za preostale tabele:

```
alter table smjerovi add(constraint smjerovi_ogr1 primary key(ss));
alter table odsjeci add(constraint ogr557 primary key(so));
```

Dodavanje nove kolone biće postignuto kao što je ranije objašnjeno:

```
alter table predmet add(rb number(4) not null unique);
```

Iza naziva kolone RB stavlja se tip podatka kao i odgovarajuća ograničenja (koja mogu, a ne moraju biti imenovana). U našem slučaju to su ograničenja `not null` i `unique`. Ovo indicira da je neophodno obrisati sve stare vrijednosti iz tabele PREDMETI prije dodavanja ove kolone (jer će podacima u ovoj koloni po default-u biti dodijeljene `null` vrijednosti, što će izazvati grešku).

Za tabelu PREDMETI dvije kolone treba da budu zajedno primarni ključ. U ovom slučaju one zajedno, a ne pojedinačno, treba da zadovoljavaju uslov o jedinstvenosti vrijednosti (dakle, podaci iz obije kolone zajedno ne smiju biti ponovljeni, dok je unutar jedne kolone dozvoljeno). `Null` vrijednost *ne smije biti unešena ni u jednoj koloni*. ALTER TABLE će biti sljedećeg oblika:

```
alter table predmet add(constraint predmet_ogr1 primary key(rb,so));
```

Provjere možemo vršiti unošenjem eksperimentalnih podataka, ili naredbom:

```
describe naziv_tabele;
```

- Spoljni ključ (FOREIGN KEY) je ograničenje tabele/kolona definisano na sljedeći način:

```
[constraint <naziv_ogranicenja>] [foreign key (<kolone>)]
                                  references <tabela>[(<kolone>)];
```

Ovo ograničenje specificira kolonu ili listu kolona tabele na koju se odnosi kao spoljni ključ na referenciranu tabelu `<tabela>`. Referencirana tabela se naziva *parent-table* dok se tabela koja ima spoljni ključ na nju zove *child-table.* Klučna riječ `foreign key` mora se koristiti ukoliko spoljni ključ uključuje više od jedne kolone, dok za jednu kolonu *nije obavezna*. Iza klauzule `references` stoji naziv *parrent-table*-a kao i spisak njenih kolona na koje se vrši referenciranje. Ukoliko se ne zada ovaj spisak kolona, tada se podrazumijeva spisak kolona koje čine primarni ključ *parrent-table*-a. Jako je bitno napomenuti da spoljni ključ mora referencirati kompletan primarni ključ *parrent table*-a. Vrijednosti u koloni(ama) *child table-a* koje su spoljni ključ moraju zadovoljavati jedan od dva moguća uslova:

- vrijednosti moraju biti iz skupa vrijednosti odgovarajuće kolone(a) *parrent table*-a ili
- vrijednosti mogu biti `null` vrijednost

U našem slučaju, u tabeli STUDENT kolona SS treba da bude spoljni ključ na kolonu SS iz tabele SMJEROVI:

```
alter table student add(constraint student_ogr2 foreign key(ss)
references smjerovi(ss));
```

Ključna riječ `foreign key` nije bila obavezna, jer je samo jedna kolona, SS spoljni ključ. Obratiti pažnju da je u *parrent-table*-u odnosno referenciranoj tabeli SMJEROVI kolona SS primarni ključ. `student_ogr2` je naziv ovog ograničenja.

Slično je potrebno odraditi i za drugi spoljni ključ tebele STUDENT, kolonu SO:

```
alter table studenti add(constraint student_ogr3 foreign key(so)
references odsjeci(so));
```

Analogno prethodnom, postavićemo i spoljne ključeve za tabelu PREDMET:

```
alter table predmet add(constraint predmet_ogr2 foreign key(ss)
references smjerovi(ss))
alter table predmet add(constraint predmet_ogr3 foreign key(so)
references odsjeci(so));
```

Sva ograničenja nalaze se u tabeli `user_constraints`. Može se zadati odgovarajući upit za dobijanje liste ograničenja sa informacijama o njima tabele sa datim nazivom:

```
select * from user_constraints where table_name='PREDMET';
```

Za brisanje ograničenja može se koristiti naredba:

```
alter table naziv drop constraint naziv_ogranicenja;
```

- Unesimo neke vrijednosti u tabelu PREDMETI:

```
insert into predmet(so, ss, gs, naziv, rb)
values(12,12001,4,'VLSI',1);
insert into predmet(so, ss, gs, naziv, rb)
values(12,12001,4,'Multimedijalni sistemi',2);
insert into predmet(so, ss, gs, naziv, rb)
values(12,12001,4,'Fizičko-tehnička mjerenja',3);
```

- Spisak predmeta dobićemo sljedećim upitom:

```
select naziv from student s, predmet p where ind='25/09' and
s.so=p.so and s.ss=p.ss;
```

Neophodno je kombinovati podatke iz tabela STUDENT i PREDMET. Ovim tabelama dodijeljeni su kraći nazivi (aliasi) S i P respektivno. Aliasi se dodjeljuju u cilju pojednostavljenja upita. *Bitno je naglasiti da kada se uvedu aliasi, u dijelu SELECT upita ispred FROM moraju se koristiti uvedeni aliasi umjesto pravih naziva tabela.* Kada se pristupa odgovarajućim kolonama tabele, koristi se operator tačka (.), u obliku `naziv_tabele.naziv_kolone` (umjesto naziva tabele može stajati alias). Pošto su kolone NAZIV i IND jedinstvene za tabele PREDMET odnosno STUDENT, nije neophodno koristiti pristup koloni operatorom tačka (zna se kojoj tabeli koja kolona pripada). U uslovu SELECT upita postavlja se da IND ima vrijednost iz postavke zadatka. Da bi bilo izvršeno spajanje tabela STUDENT i PREDMET, kao uslov SELECT upita postavlja se da SO i SS u obije tabele mora imati istu vrijednost. Upit će pronaći red u tabeli STUDENT sa odgovarajućom vrijednošću u koloni IND. Istovremeno će potražiti sve redove u tabeli PREDMET u kojoj stoje iste vrijednosti u kolonama SO i SS kao i u redu iz tabele STUDENT u kojem stoji odgovarajući broj indeksa. Rezultujuća tabela će imati redove sastavljene iz dva dijela: dio u kojem se svaki put ponavlja red iz tebele STUDENT sa odgovarajućim brojem indeksa, i drugi dio u kojem stoje pridruženi redovi iz tabele PREDMET koji su imali isti SO i SS kao red iz tabele STUDENT koji se ponavlja. Konačno, pošto prije FROM dijela opisanog SELECT upita stoji samo kolona NAZIV, iz rezultujuće tabele biće prikazani jedino nazivi predmeta koje student sluša.

Slična je logika upita koji daju odsjek i smjer na kojem je student upisan. U ovom slučaju kombinujemo podatke iz tri tabele. Jedna moguća varijanta upita je sljedeća:

```
select s.naziv, o.naziv from smjerovi s, odsjeci o, studenti st
where st.ss=s.ss and st.so=o.so;
```

3. Ukoliko bi se agregatna funkcija AVG primijenila na kolonu SALARY, rezultat bi bila samo jedna vrijednost – prosječna zarada zaposlenih iz tabele HR.EMPLOYEES. Pošto je potrebno prikazati prosječne zarade po odjeljenjima (department), to je neophodno koristiti klauzulu GROUP BY. Opšti oblik upita sa ovom klauzulom je oblika:

```
select <kolone>
from <tabela>
where <uslov>
group by <kolone_za_grupisanje>
[having <uslovi_za_grupe>]
```

Ona se, dakle, koristi kad je potrebno prvo grupisati vrste koje imaju određene vrijednosti u nekoj (nekim) kolonama, pa tek onda primijeniti agregatnu funkciju. One kolone koje se nalaze iza ključne riječi SELECT a imaju iste vrijednosti u redovima se grupišu po tim vrijednostima. Zatim se izvršavaju agregatne funkcije iz SELECT dijela, *na svaku grupu odvojeno. Samo one kolone koje se nalaze u GROUP BY dijelu mogu se naći u SELECT dijelu upita. U protivnom, javlja se greška u upitu.* Odabir vrsta od interesa vrši se standardno u WHERE dijelu upita. Kada se formiraju grupe, i neke od njih mogu biti eliminisane, korišćenjem klauzule HAVING, koja se, dakle, može opciono koristiti.

Redosljed izvršavanja djelova upita sa grupisanjem je sljedeći:

- Selektuju se sve vrste koje zadovoljavaju uslove iz iz WHERE dijela upita
- Od ovih vrsta se prave grupe, u skladu sa GROUP BY klauzulom
- Eliminišu se sve grupe koje ne zadovoljavaju uslove iz HAVING klauzule (ako ona postoji)
- Primjenjuju se agregatne funkcije na svaku grupu
- Dodjeljuju se vrijednosti za kolone i agregacije koje se nalaze u SELECT dijelu.

U našem zadatku vrši se spajanje tabela HR.EMPLOYEES i HR.DEPARTMENTS. Povezivanje će se vršiti na osnovu uslova jednakosti vrijednosti u koloni DEPARTMENT_ID. Za svaki DEPARTMENT_ID (i njemu odgovarajući DEPARTMENT_NAME), odnosno za svako odjeljenje, treba izračunati prosječnu zaradu. Primijeniće se agregatna funkcija AVG, a pošto je potrebno odrediti prosjek za sve zaposlene koji pripadaju jednom odjeljenju, odnosno imaju isti DEPARTMENT_ID, vršiće se grupisanje po vrijednostima ove kolone. Tako će svima koji imaju DEPARTMENT_ID 100 na primjer biti izračunat prosjek plate. Zatim za zaposlene sa drugim DEPARTMENT_ID-em itd. Pošto je potrebno ispisati i DEPARTMENT_NAME, mora se i ova kolona dodati u GROUP_BY dio, a podaci za nju se vade iz tabele HR.DEPARMENTS. Budući da u tabeli HR.DEPARTMENTS jednom DEPARTMENT_ID-u odgovara jedan DEPARTMENT_NAME, ovakvo grupisanje je prirodno. Rezultate koje bude dala agregatna funkcija AVG potrebno je zaokružiti na dvije decimale, što se postiže funkcijom `ROUND(vrijednost, broj_decimala)`. Ranije je napomenuto da agregatna funkcija COUNT daje broj redova u rezultujućoj koloni. Ako je prilikom grupisanja učestvovalo N redova u datoj grupi, za vrijednost date grupe funkcija COUNT daje rezultat N. Broj redova je u našem slučaju broj zaposlenih koji imaju isti DEPARTMENT_ID (odnosno DEPARMENT_NAME, jer smo i njega uključili u grupisanje), i koji su za dati DEPARMENT_ID učestvovali u računanju prosjeka. Traženi upit je:

```
select e.department_id, d.department_name, round(avg(e.salary),2),
count(*) "broj zaposlenih" from hr.employees e, hr.departments d
where e.department_id=d.department_id group by e.department_id,
department_name;
```

Obratiti pažnju da je potrebno upotrebljavati iste aliase kolona u GROUP BY i u SELECT dijelu.

4. Potrebno je iz rezultata upita iz prethodnog zadatka eliminisati ona odjeljenja koja imaju manje od 5 zaposlenih, ili drugim riječima, grupe koje imaju u sebi manje od 5 redova, ili njih tačno toliko. Ovo ćemo postići korišćenjem klauzule HAVING, koja će eliminisati grupe koje ne zadovoljavaju odgovarajući uslov, koji je, u našem slučaju, da je broj zaposlenih veći od 5:

```
select e.department_id, d.department_name, round(avg(e.salary),2),
count(*) "broj zaposlenih" from hr.employees e, hr.departments d
```

```
where e.department_id=d.department_id group by e.department_id,
department_name having count(*)>5;
```

Obratiti pažnju da je u HAVING dijelu za uslove moguće koristiti agregatne funkcije, to čak ne moraju biti one koje su korišćene u SELECT dijelu. Jedino se mogu koristiti kolone iz SELECT dijela upita. Ne mogu se koristiti aliasi kolona, jer HAVING ipak služi za generisanje krajnjeg rezultata upita.

Sada je rezultat prethodnog upita (tabelu) potrebno smjestiti u kreiranu tabelu ODJELJENJA. Kreiranje i upis podataka može se najjednostavnije postići na sljedeći način:

```
create table odj2 as select e.department_id, d.department_name,
round(avg(e.salary),2) "prosječna plata", count(*) "broj zaposlenih"
from hr.employees e, hr.departments d where
e.department_id=d.department_id group by e.department_id,
department_name having count(*)>5
```

SELECT upit se može po želji staviti u običnim zagradama. Bitno je naglasiti da je kolonu u kojoj je rezultat agregatne funkcije AVG neophodno imenovati (kolona "`prosječna plata`"), u protivnom javlja se greška.

5. Informacije o lokacijama u šemi HR nalaze se u tabeli HR.LOCATIONS. To su lokacije odjeljenja u kojima zaposleni rade. HR.JOBS sadrži informacije o poslovima zaposlenih, uključujući i nazive poslova. Ovu tabelu moramo uključiti u rezultujući upit. JOB_ID je kolona koja će povezati ovu tabelu sa tabelom HR.EMPLOYEES. U našem slučaju, u zadatku je odabran naziv posla koji obavlja samo jedna osoba. U opštem slučaju, može se desiti da više osoba iz istog odjeljenja obavlja posao sa istim nazivom, tako da je u tom smislu postavka zadatka specifična. U tabeli HR.EMPLOYEES postoji kolona DEPARTMENT_ID koja će nam poslužiti za povezivanje sa tabelom HR.DEPARTMENTS, koja će opet dati vezu preko LOCATION_ID-a sa tablelom HR.LOCATIONS. Kao rješenje biće prikazan sadržaj odgovarajućih kolona tabele HR.LOCATIONS:

```
select STREET_ADDRESS, POSTAL_CODE, CITY, STATE_PROVINCE, COUNTRY_ID
from hr.employees e, hr.jobs j, hr.departments d, hr.locations l
where job_title='President' and e.job_id=j.job_id and
e.department_id=d.department_id and d.location_id=l.location_id;
```

6. U tabeli HR.JOBS postoje kolone MIN_SALARY i MAX_SALARY koje daju informacije o minimalnoj i maksimalnoj plati, respektivno, za određenu vrstu posla. Rezultujući upit je:

```
select first_name ime, last_name prezime, salary plata from
hr.employees where salary between (select min_salary from hr.jobs
where job_title='Administration Assistant') and (select max_salary
from hr.jobs where job_title='Administration Assistant');
```

Obratiti pažnju da su SELECT upiti u WHERE dijelu glavnog SELECT upita stavljeni u malim zagradama. BETWEEN je bilo moguće koristiti jer ono *uključuje* i granične slučajeve, odnosno minimalnu i maksimalnu platu. Ekvivalentno rješenje dobija se korišćenjem operatora poređenja '>=' i '<='.

7. Tabela SPR ima primarni ključ sastavljen od dvije kolone SPRID i JID. Tabela PREDMETI sadrži takođe ove dvije kolone. Postoji više jedinica koje imaju iste SPRID-ove, pa je stoga neophodno prilikom povezivanja tabela SPR i PREDMETI koristiti obije kolone. Upit za spisak predmeta je sljedeći:

```
select p.naziv from spr s, predmeti p where s.sprid=10 and s.jid=13
and p.sprid=s.sprid and s.jid=p.jid and p.semestar=5;
```

dok je upit za naziv jedinice:

```
select naziv from jedinice where jid=13;
```

8. Jedno moguće rješenje je:

```
select n.ime from predmeti p, opt o, nastavnici n where
p.naziv='Adaptivni diskretni sistemi i neuralne mreže' and
o.pid=p.pid and n.nid=o.nid;
```

Zadatak je mogao biti odrađen i na sljedeći način:

```
select ime from nastavnici where nid in (select o.nid from predmeti
p, opt o where p.naziv='Adaptivni diskretni sistemi i neuralne
mreže' and o.pid=p.pid);
```

Ova varijanta rješenja je ekvivalentna prethodnoj, s tim što je SELECT upit u WHERE dijelu glavnog upita poslužio da se dobije spisak NID-ova nastavnika iz tabele opterećenja, a zatim se u glavnom upitu iz tabele NASTAVNICI dobija spisak imena nastavnika sa ovim vrijednostima NID-ova. Ovdje je izuzetno bitno istaći da u where dijelu glavnog SELECT upita nije bilo moguće koristiti operator '=' jer upit `select o.nid from predmeti p, opt o where p.naziv='Adaptivni diskretni sistemi i neuralne mreže' and o.pid=p.pid)` kao rezultat vraća *spisak* NID-ova, a ne samo jednu vrijednost, pa je nemoguće izjednačiti NID iz tabele NASTAVNICI sa nizom NID-ova iz ovog upita. Izjednačavanje se može vršiti jedino u slučaju kada se kao rezultat upita sa kojim se postavlja uslov jednakosti dobija samo jedna vrijednost! Zato je u ovom slučaju neophodno korišćenje operatora IN!

9. Za rješenje ovog zadatka koristićemo upit iz prethodnog, s tim što ćemo kao rezultat upita staviti vrijednost NID-a umjesto imena nastavnika koji su angažovani na predmetu 'Adaptivni diskretni sistemi i neuralne mreže':

```
select n.nid from predmeti p, opt o, nastavnici n where
p.naziv='Adaptivni diskretni sistemi i neuralne mreže' and
o.pid=p.pid and n.nid=o.nid
```

Tabela PREDMETI sadrži informacije koje se traže u postavci, dok je veza predmeta sa nastavnicima koji ih drže data u tabeli opterećenja OPT. Tabele je jednostavno povezati izjednačavajući PID iz tabele PREDMETI i tabele OPT, dok je u tabeli OPT potrebno odrediti one redove koji imaju NID-ove nastavnika koji su angažovani na predmetu iz postavke zadatka. `o.nid` je potrebno da ima skup vrijednosti NID-ova koje daje gore navedeni upit, pa će, kao što je objašnjeno u prethodnom zadatku biti korišćen operator IN, jer operator '=' se ne može koristiti za izjednačavanje jedne vrijednosti u vrsti kolone NID iz tabele OPT sa nizom vrijednosti NID-a koje daje gore navedeni upit.

```
select distinct p.naziv, fcp "Predavanja", fcv "Vježbe", fcl
"Laboratorija" from opt o, predmeti p where p.pid=o.pid and o.nid
in(select n.nid from predmeti p, opt o, nastavnici n where
p.naziv='Adaptivni diskretni sistemi i neuralne mreže' and
o.pid=p.pid and n.nid=o.nid);
```

10. Tabela OPT u sebi sadrži kolone CP (časovi predavanja), CV (časovi vježbi), CL (časovi laboratorije), GV (broj grupa za vježbe) i GL (broj grupa za časove laboratorije). Pošto nastavnik može držati vježbe u više grupa, to je ukupan broj časova vježbi jednak broju časova vježbi pomnoženim sa brojem grupa za vježbe. Slično važi i za laboratorijske vježbe. Rješenje je upit:

```
select sum(cp+cv*gv+cl*gl) from opt where nid=130143;
```

Operacije množenja i sabiranja će se izvršiti za svaki red rezultujućeg upita (to jest za svaki predmet), dok je za krajnji rezultat potrebno sabrati sve redove, odnosno primijeniti agregatnu funkciju SUM.