UNIVERZITET U BEOGRADU
SAOBRAĆAJNI FAKULTET
PTT ODSEK

PROJEKTOVANJE INFORMACIONIH SISTEMA

- skripta sa primerima i ispitnim zadacima -

1. deo:
RELACIONA ALGEBRA

2. deo:
STRUCTURED QUERY LANGUAGE (SQL)

Ivan Milović TS 2000-1-003
milovici@gmail.com

Relaciona algebra

- σ restrikcija
- π projekcija (menja se struktura relacije)
- $\cup$ unija (da bi se ostvarila unija obe relacije treba da imaju istu strukturu)
- $-$ razlika
- $\cap$ presek
- $\times$ Dekartov proizvod
- $\bowtie$ prirodno spajanje
- $\overline{\bowtie}$ spoljno spajanje
- $\overline{\cup}$ spoljna unija

Primer 1:

R1:

BI	Ime
121	Pera
318	Mika
417	Laza
509	Zika
601	Ana

R2:

BI	Ime
318	Mika
509	Zika
511	Slavka

σ restrikcija π projekcija

$\sigma_{BI<400}$ (R1)

BI	**Ime**
121	Pera
318	Mika

π_{Ime} (R2)

Ime
Mika
Zika
Slavka

$\cup$ unija

R1 $\cup$ R2

BI	Ime
121	Pera
318	Mika
417	Laza
509	Zika
601	Ana
511	Slavka

$-$ razlika

R1 $-$ R2

BI	Ime
121	Pera
417	Laza
601	Ana

$\cap$ presek

R1 $\cap$ R2

BI	Ime
318	Mika
509	Zika

Primer 2: $\times$ Dekartov proizvod

R1:

A
1
2

R2:

B
A
B
C

R1 $\times$ R2

A	B
1	A
1	B
1	C
2	A
2	B
2	C

Zadatak 1: Data je relacija STUDENT. Dekomponovati ovu relaciju na njene projekcije: R1(BI, IDPRED, IME) i R2(IME, OCENA). Utvrditi da li je ovo dekompozicija bez gubitka informacije.

BI	IDPRED	IME	OCENA
123	7	DEJAN	7
123	11	DEJAN	7
354	7	VLADAN	8
723	11	DEJAN	7
723	3	DEJAN	8

Rešenje:

R1(BI, IDPRED, IME)
$R1 = \pi_{BI,\ IDPRED,\ IME}\ (R)$

R2(IME, OCENA)
$R2 = \pi_{IME,\ OCENA}\ (R)$

R1:

BI	IDPRED	IME
123	7	DEJAN
123	11	DEJAN
354	7	VLADAN
723	11	DEJAN
723	3	DEJAN

R2:

IME	OCENA
DEJAN	7
DEJAN	7
VLADAN	8
DEJAN	7
DEJAN	8

Prirodno spajanje (uzima se prva vrsta iz R1 i uklapa s čim god može iz R2)

$R1 \bowtie_{IME} R2$

BI	IDPRED	IME	OCENA	
123	7	DEJAN	7	√
123	7	DEJAN	8	—
123	11	DEJAN	7	√
123	11	DEJAN	8	—
354	7	VLADAN	8	√
723	11	DEJAN	7	√
723	11	DEJAN	8	—
723	3	DEJAN	7	—
723	3	DEJAN	8	√

Datom dekompozicijom je došlo je do gubitka informacija zbog neodređenosti (ne znamo da li je ocena 7 ili je ocena 8.

ZA DOAMĆI:

R1(BI, IME)
R2(IDPRED, OCENA)

R1(BI, IDPRED, IME)
R2(BI, IDPRED, OCENA)

Zadatak 2: Date su relacije R1 i R2. Potrebno je prikazati rezultate sledećih operacija: projekcije relacije R1 po atributu C, prirodno spajanje relacije R1 i R2 po atributu A i D, spoljno spajanje relacije R1 i R2 po A i D i spoljne unije relacije R1 i R2.

R1:

A	B	C
1	?	4
2	MISIC	33
2	MARJANOVIC	12
?	CVETKOVIC	5

R2:

D	E
2	LAZAREVIC
1	ZIVKOVIC
3	?

Rešenje:

π_C R1

C
4
33
12
5

R1 $\bowtie_{A,D}$ R2

A	B	C	D	E
1	?	4	1	ZIVKOVIC
2	MISIC	33	2	LAZAREVIC
2	MARJANOVIC	12	2	LAZAREVIC

$\bowtie$ Spoljno spajanje - spajaju se i one preostale vrste koje nisu mogle da se spoje

R1 $\bowtie_{A,D}$ R2

A	B	C	D	E
1	?	4	1	ZIVKOVIC
2	MISIC	33	2	LAZAREVIC
2	MARJANOVIC	12	2	LAZAREVIC
?	CVETKOVIC	5	?	?
?	?	?	3	?

$\cup$ Spoljna unija – uključe se svi atributi iz obe relacije

R1 $\cup$ R2

A	B	C	D	E
1	?	4	?	?
2	MISIC	33	?	?
2	MARJANOVIC	12	?	?
?	CVETKOVIC	5	?	?
?	?	?	2	LAZAREVIC
?	?	?	1	ZIVKOVIC
?	?	?	3	?

Zadatak 3: Data je relaciona šema sa relacijama:

p: posecuje (pijanica, bar)
s: sluzi (bar, pivo)
v: voli (pijanica, pivo).

p:

pijanica	bar
MIKA	DOMOVINA
MIKA	SANSA
LAZA	DOMOVINA
PERA	ORASAC
PERA	1001 NOC

s:

bar	pivo
SANSA	LAV
SANSA	NP
DOMOVINA	LAV
DOMOVINA	VALJEVSKO
ORASAC	TUBORG
ORASAC	JELEN
1001 NOC	NP

v:

pijanica	pivo
MIKA	LAV
LAZA	NP
MIKA	VALJEVSKO
PERA	NP

Za sledeće upite napisati odgovarajuće iskaze relacione algebre:

1) Prikazati nazive barova koji služe LAV pivo;
2) Prikazati imena pijanica koji vole LAV pivo;
3) Prikazati imena pijanica koji posećuju bar jedan bar u kome se služi pivo koje dotični pijanica voli;
4) Naći pijanice koji piju u istom baru sa pijanicama koji vole NP (Nikšićko pivo);
5) Naći pijanice koji ne posećuju bar koji služi pivo koje oni vole;
6) Naći pijanice koji ne piju u istom baru sa pijanicama koji vole pivo koje taj bar služi i pivo JELEN.

Rešenje:

1) Nazivi barova koji služe LAV pivo:

$\sigma_{\text{pivo=LAV}}$ (s)

bar	pivo
SANSA	LAV
DOMOVINA	LAV

$\Rightarrow$

$\pi_{\text{bar}}(\sigma_{\text{pivo=LAV}}(\text{s}))$

bar
SANSA
DOMOVINA

2) Imena pijanica koji vole LAV pivo:

$R1 = \sigma_{\text{pivo=LAV}}$ (v)

pijanica	pivo
MIKA	LAV

$\Rightarrow$

$R = \pi_{\text{pijanica}}(R1)$

pijanica
MIKA

3) Imena pijanica koji posećuju bar jedan bar u kome se služi pivo koje dotični pijanica voli:

$$R1 = p \bowtie_{bar} s$$

pijanica	bar	pivo
MIKA	DOMOVINA	LAV
MIKA	DOMOVINA	VALJEVSKO
MIKA	SANSA	LAV
MIKA	SANSA	NP
LAZA	DOMOVINA	LAV
LAZA	DOMOVINA	VALJEVSKO
PERA	ORASAC	TUBORG
PERA	ORASAC	JELEN
PERA	1001 NOC	NP

$$\Downarrow$$

$$R2 = R1 \bowtie_{pijanica,\ pivo} v$$

pijanica	bar	pivo
MIKA	DOMOVINA	LAV
MIKA	DOMOVINA	VALJEVSKO
MIKA	SANSA	LAV
PERA	1001 NOC	NP

$$\Downarrow$$

$$R = \pi_{pijanica}(R2)$$

pijanica
MIKA
PERA

4) Pijanice koji piju u istom baru sa pijanicama koji vole NP (Nikšićko pivo):

$$R1 = \sigma_{pivo=NP}(v)$$

pijanica	pivo
LAZA	NP
PERA	NP

$$\Downarrow$$

$$R2 = R1 \bowtie_{pijanica} p$$

pijanica	pivo	bar
LAZA	NP	DOMOVINA
PERA	NP	ORASAC
PERA	NP	1001 NOC

$$\Downarrow$$

$$R3 = R2 \bowtie_{bar} p$$

pijanica R2	pivo	bar	pijanica P
LAZA	NP	DOMOVINA	MIKA
LAZA	NP	DOMOVINA	LAZA
PERA	NP	ORASAC	PERA
PERA	NP	1001 NOC	PERA

$$\Downarrow$$

$$R = \pi_{pijanica\ P}(R3)$$

pijanica
MIKA
LAZA
PERA

$$\Downarrow$$

$$\sigma_{pijanica\ R2 \neq pijanica\ P}(R3)$$

pijanica
MIKA

5) Pijanice koji ne posećuju bar koji služi pivo koje oni vole:

$$R1 = v \bowtie_{pivo} s$$

pijanica	pivo	bar
MIKA	LAV	SANSA
MIKA	LAV	DOMOVINA
LAZA	NP	SANSA
LAZA	NP	1001 NOC
MIKA	VALJEVSKO	DOMOVINA
PERA	NP	SANSA
PERA	NP	1001 NOC

$$\Downarrow$$

$$R2 = \pi_{pijanica,\ bar}\ R1$$

pijanica	bar
MIKA	SANSA
MIKA	DOMOVINA
LAZA	SANSA
LAZA	1001 NOC
MIKA	DOMOVINA
PERA	SANSA
PERA	1001 NOC

$$\Downarrow$$

$$R3 = R2 - p$$

pijanica	bar
LAZA	SANSA
LAZA	1001 NOC
PERA	SANSA

$$\Downarrow$$

$$R = \pi_{pijanica}(R3)$$

pijanica
LAZA
PERA

Ili, zapisano u jednom redu, rešenje glasi:

$$R = \pi_{pijanica}\ (\pi_{pijanica,\ bar}\ (v \bowtie_{pivo} s) - p)$$

6) Naći pijanice koji ne piju u istom baru sa pijanicama koji vole pivo koje taj bar služi i pivo JELEN.

Structured Query Language (SQL)

- **NAZIVI**

- Slova, brojevi
- Mogu da se koriste _, #, $, @ > identifikatori

- **TIPOVI PODATAKA (DATA TYPES)**

  - **INTEGER** / **INT** – može sadržati samo brojeve
  - **FLOAT** (length, decimal) – redni broj
  - **CHAR** (length) – može sadržati brojeve, slova i ostale znakove
  - **NUMBER** – realan broj
  - **DECIMAL** (10,2) – prvi broj definiše koliko ima ukupno cifara, a drugi koliko cifara ima iza zareza
  - **DATE** (yyyymmdd) – datum
  - **DATETIME**
  - **TIMESTAMP**
  - **TIME**
  - **TEXT**

- Svaka tabela je neka relacija, na primer:

INT	*CHAR*	*DATE*	*TIME*
C_1	**C_2**	**C_3**	**C_4**

- Podaci se u tabelu smestaju vrsta po vrsta.
- Podaci iz iste vrste su u relaciji

Primer: Relacija STUDENT

char(30)	*char(30)*	*int*
broj indeksa	ime	godina
101	TINA	3
153	MARKO	4
160	TIJANA	2
205	MILICA	3
182	DARKO	4
207	MIRKO	? NULL

- **ARITMETIČKI OPERATORI**

  + , - , / , * , %

- **OPERATORI POREĐENJA**

  = , > , >= , <= , < , <> ili !=

- **REZULTATI OPERATORA POREĐENJA**

- **TRUE**
- **FALSE**
- **UNKNOWN** (u slučaju da postoji nedefinisana vrednost)

- **OPERATORI ZA RAD NAD KARAKTERIMA**

- **LIKE** / **NOT LIKE**
- **STARTING WITH**

Primer:

- ime **LIKE** 'T%' ili ime **STARTING WITH** 'T' (**LIKE** je generalnije)
- Dva džoker znaka:
  % zamenjuje više znakova
  _ zamenjuje tačno jedan karakter na određenoj poziciji

- ime **LIKE** 'M_RKO'
- ime **LIKE** '%RK%'

- Komanda **SELECT [ALL** / **DISTINCT]** izrazi – odabira, tj. selektuje podatke iz baze

  FROM tabele

- Klauzule:

  [WHERE predikat**]**
  [GROUP BY polje**]**
  [HAVING predikat**]**
  [ORDER BY polje**]**

  - **DISTINCT** – samo različite vrednosti koje su pronađene
  - **ALL** – sve vrednosti
  - **[ ]** – može da se piše, ali i ne mora
  - ***** – zamenjuje sve podatke ili se navode svi podaci redom

Primer:

```
SELECT *
FROM  STUDENT
WHERE  ime LIKE 'T%';
```

Rezultat datog upita je:

broj indeksa	ime	godina
101	TINA	3
160	TIJANA	2

- U **WHERE** klauzuli se zadaju kriterijumi po kojima se nalazi informacija
- **HAVING** i **GROUP BY** se odnose na obavljanje operacija nad grupama podataka
- U **SELECT** mogu da se traže samo vrednosti onih polja koja su zadata u **GROUP BY**

- **GROUP BY** – ne sme se koristiti sa varijablama koje nisu pozvane sa **SELECT**
- Ako se **HAVING** komanda koristi sama, bez **GROUP BY**, uz nju mora biti i jedna agregatna funkcija (**COUNT**, **SUM**, **AVG**...)
- **ORDER BY** se koristi za sortiranje po kolonama
    - **ASC** / **DESC** – rastući, odnosno opadajući redosled sortiranja; ako se ne navede ništa sortiranje se vrši po rastućem redosledu.

- Svaka tabela može da ima KLJUČ (**KEY**). To je ona vrednost koja jednoznačno određuje svaku vrstu u tabeli. To se zove PRIMARNI KLJUČ (**PRIMARY KEY**) i govori nam da ne mogu postojati dva reda sa istom vrednošću u toj koloni. STRANI KLJUČ se koristi za pozivanje informacija iz druge tabele, spajanjem stranog ključa iz jedne tabele sa odgovarajućim (primarnim) ključem uz druge tabele. Strani ključ nema to ograničenje da moraju svi podaci u tabeli da budu različiti.

- **JOIN** – komanda za spajanje tabela

    - **INNER JOIN** - pokazuje samo podatke ako ih stvarno ima; ako ih nema, ti redovi neće biti pokazani

Primer:

```
SELECT tab_name1.col_name1, tab_name2.col_name2
FROM tab_name1
INNER JOIN tab_name2
ON tab_name1.col_name3 = tab_name2.col_name3;
```

- **LEFT JOIN** – pokazuje sve podatke iz prve tabele, bez obzira da li se slažu sa rezultatima iz druge tabele i svi redovi će biti prikazani

Primer:

```
SELECT tab_name1.col_name1, tab_name2.col_name2
FROM tab_name1
LEFT JOIN tab_name2
ON tab_name1.col_name3 = tab_name2.col_name3;
```

- **RIGHT JOIN** – pokazuje sve podatke iz druge tabele, bez obzira da li se slažu sa rezultatima iz prve tabele i svi redovi će biti prikazani.

Primer:

```
SELECT tab_name1.col_name1, tab_name2.col_name2
FROM tab_name1
RIGHT JOIN tab_name2
ON tab_name1.col_name3 = tab_name2.col_name3;
```

- Operator **spajanje** (sa stringovima) ||

Primer:

```
SELECT ime||','||prezime
FROM STUDENT
```

Prikazaće se imena i prezimena spojena zarezom.

- **LOGIČKI OPERATORI**

- **AND** – mora da zadovolji oba uslova
- **OR** – baram jedan od uslova mora biti zadovoljen
- **NOT** – negacija

- Ako se kombinuju **AND** i **OR**, moraju se odvojiti zagradom

- **SKUPOVNI OPERATORI**

- **UNION** – unija; spaja sume rezultata dve selekcije, pri cemu broj kolona u oba rezultata mora biti isti i kolone moraju biti istog tipa (DATA TYPE).
- **INTERSECT** – presek; daje samo one kolone koje se nalaze u obe selekcije.
- **MINUS** – razlika; kao rezultat daje samo one kolone iz prve selekcije (znači prve tabele), koje nemaju korespondirajuće redove u drugoj selekciji, tj. tabeli.

- **MEŠOVITI OPERATORI**

- **BETWEEN**
- **IN** / **NOT IN**
- **ALL** – rezultat je tačan ako se sve vrednosti slažu
- **ANY** – minimalno jedna vrednost se mora slagati
- **EXISTS** / **NOT EXISTS** – proveravamo da li jedna selekcija uopšte sadrži podatke
- **IF** – dodatni predikat (**IF EXISTS**...)

- **AGREGATNE FUNKCIJE**

- **COUNT** – prebrojava selektovane podatke
- **DISTINCT COUNT** – prebrojava selektovane, ali različite podatke iz tabele
- **SUM** – sumira
- **AVG** – daje prosek
- **MIN** – daje minimalnu vrednost
- **MAX** – daje maksimalnu vrednost

SELECT *
FROM tabele
WHERE uslov
GROUP BY polje
HAVING agregatni uslov

- **SELECT COUNT (*)** – prebrojava koliko redova ima tabela

- **KREIRANJE BAZE PODATAKA**

- **CREATE DATABASE** db_name

- **BRISANJE BAZE PODATAKA**

- **DROP DATABASE** db_name

- **ODABIRANJE BAZE PODATAKA NAD KOJOM ĆE SE IZVRŠITI UPITI**

- **USE DATABASE** db_name

- **KREIRANJE TABELE**

- **CREATE TABLE** tbl_name [create_definition]

- create_definition:

    - col_name – naziv kolone

        - DATA TYPE [**NOT NULL** / **NULL**] – da li kolona može ili ne može prihvatiti nedefinisane vrednosti

        - [**DEFAULT** default_value] – podrazumevane vrednosti

        - [**AUTO_INCREMENT**] – pri svakom unosu nove vrste ta kolona se povećava za jedan

        - **PRIMARY KEY** [col_name, ...]

Primer:

CREATE TABLE STUDENT **(**broj indeksa **CHAR(**30**)**,
ime **CHAR(**30**)**,
godina **INT,**
PRIMARY KEY (broj indeksa**));**

- **UNOŠENJE VREDNOSTI U TABELU**

- **INSERT INTO** tbl_name **VALUES** (’value1’, ’value2’,...) [**WHERE**...] – definiše se gde**;**

Primer:

INSERT INTO dobavljac **VALUES (**1, ’DOBAVLJAC 1’, **NULL**, ’BEOGRAD’**);**

- Vrednosti koje nisu navedene će biti **NULL** vrednosti.

- **UNOŠENJE U TABELU VREDNOSTI IZ DRUGE TABELE**

- **SELECT** [col_names...] **INTO** new_table **FROM** source_table**;**

Primer:

SELECT * INTO tabela_backup **FROM** tabela**;**

- **BRISANJE VREDNOSTI IZ TABELE**

- **DELETE FROM** tbl_name [**WHERE**...]

- **IZMENA VREDNOSTI U TABELI**

- **UPDATE** tbl_name **SET** col_name=exp... [**WHERE**...] – definiše se gde

Primer:

UPDATE projekat **SET** grad='BEOGRAD' **WHERE** grad='BG'**;**

UPDATE radnik **SET** ld=ld*1,1 **WHERE** ld>50000**;**

- **PROMENA STRUKTURE TABELE**

- **ALTER TABLE** tbl_name [alter_spec...]

- alter_spec:

    - **ADD** [**COLUMN**] create_definition [**FIRST** / **AFTER** col_name]

        - [**COLUMN**] – može da se stavlja, ali ne mora

        - [**FIRST** / **AFTER** col_name] – može, a ne mora, određuje gde će se postaviti nove kolone; ako se ništa ne navede, onda se kolona postavlja na kraju

    - **CHANGE** old_name [create_definition]

    - **DROP** [**COLUMN**] col_name

    - **RENAME** [**TO**] new_table_name

Primeri:

ALTER TABLE radnik **ADD** prezime **CHAR(**30**) AFTER** ime**;**

ALTER TABLE radnik **CHANGE** prezime prezime_radnika **CHAR(**40**);**

ALTER TABLE radnik **DROP** prezime_radnika**;**

- **BRISANJE TABELE**

- **DROP TABLE** tbl_name

- Date su relacije:

avio

ime
TINA
MARKO
MIRKO

ptt

ime
ANA
MIRKO
JOVANA

1)

```
SELECT * FROM avio
UNION
SELECT * FROM ptt;
```

ime
TINA
MARKO
MIRKO
ANA
JOVANA

2)

```
SELECT * FROM avio
UNION ALL
SELECT * FROM ptt;
```

ime
TINA
MARKO
MIRKO
ANA
MIRKO
JOVANA

3)

```
SELECT * FROM avio
INTERSECT
SELECT * FROM ptt;
```

ime
MIRKO

4)

```
SELECT * FROM avio
MINUS
SELECT * FROM ptt;
```

ime
TINA
MARKO

5) Sledeći upiti rade isto:

a)
```
SELECT *
FROM STUDENT
WHERE godina=3 OR godina=4;
```

b)
```
SELECT *
FROM STUDENT
WHERE godina BETWEEN 3 AND 4;
```

c)
```
SELECT *
FROM STUDENT
WHERE godina IN (3,4) ;
```

Rezultat je:

broj indeksa	ime	godina
101	TINA	3
153	MARKO	4
205	MILICA	3
182	DARKO	4

Zadaci:

1. Data je relaciona šema sa relacijama:

   PREDUZECE (pred, naziv, grad)
   RADNIK (rad, ime, posao, datzap, ld, premija, pred)

Napisati SQL naredbe za sledeće slučajeve:

1.1. Prikazati nazive i šifre svih preduzeća.

```
SELECT naziv, pred
FROM PREDUZECE;
```

1.2. Prikazati sve podatke o svim preduzećima.

```
SELECT *
FROM PREDUZECE;
```

1.3. Prikazati sve poslove koje radnici obavljaju.

```
SELECT posao
FROM RADNIK;
```

ili

```
SELECT DISTINCT posao
FROM RADNIK;
```

- **SELECT DISTINCT** prikazuje samo različite poslove iz tabele RADNIK.

1.4. Prikazati sve radnike koji rade u preduzeću čija je šifra 30.

```
SELECT *
FROM RADNIK
WHERE pred=30;
```

1.5. Prikazati ime, posao i lični dohodak svakog radnika iz preduzeća sa šifrom 20, čiji je lični dohodak veći od 20000

```
SELECT ime, posao, ld
FROM RADNIK
WHERE pred=20 AND ld>20000;
```

1.6. Prikazati ime, posao i pretpostavljena godišnja primanja svakog radnika iz preduzeća sa šifrom 20 čiji je lični dohodak veći od 20.

```
SELECT ime, posao, ld*12
FROM RADNIK
WHERE pred=20 AND ld>20000;
```

ime	posao	ld*12

Da se ne bi prikazao ceo izraz ld*12, može se u produžetku navesti šta će zameniti taj izraz (**pseudonim ili alias**):

Na primer: ld*12 **AS** ‘godisnja_primanja’

1.7. Prikazati sve podatke o rukovodiocima preduzeća i predsednicima firme.

```
SELECT *
FROM RADNIK
WHERE posao=”RUKOV’ OR posao=‘PRED’;
```

ili

```
SELECT *
FROM RADNIK
WHERE posao IN (‘RUKOV’,’PRED’);
```

1.8. Prikazati ime, posao i lični dohodak radnika koji zarađuje između 15000 i 30000.

```
SELECT ime, posao, ld
FROM RADNIK
WHERE ld BETWEEN 15000 AND 30000;
```

1.9. Prikazati ime, posao, šifru preduzeća i šifru radnika čije ime počinje sa slovom M.

```
SELECT ime, posao, pred, rad
FROM RADNIK
WHERE ime LIKE ‘M%’;
```

1.10. Prikazati ime, posao i lični dohodak radnika koji ne primaju premiju.

```
SELECT ime, posao, ld
FROM RADNIK
WHERE premija=0;
```

ili

```
SELECT ime, posao, ld
FROM RADNIK
WHERE premija IS NULL;
```

1.11. Pronaći sve radnike koji rade u Beogradu.

```
SELECT rad, ime
FROM PREDUZECE P, RADNIK R
WHERE P.pred=R.pred AND grad=‘BEOGRAD’;
```

1.12. Napisati SQL upit kojim se prikazuje šifra preduzeća, naziv i grad za koje postoji barem jedan radnik koji je zaposlen nakon 1. avgusta 2004.

```
SELECT DISTINCT P.pred, P.naziv, P.grad
FROM PREDUZECE P, RADNIK R
WHERE  P.pred=R.pred AND datzap>‘01/08/2004’;
```

1.13. Prikazati ukupan broj preduzeća koja postoje.

```
SELECT COUNT(*)
FROM PREDUZECE;
```

1.14. Prikazati broj poslova koje obavlja radnik sa šifrom MM1.

```
SELECT COUNT(*) AS broj_poslova
FROM RADNIK
WHERE rad='MM1';
```

1.15. Prikazati prosečnu maksimalnu i minimalnu platu radnika koji radi u preduzeću sa šifrom 40.

```
SELECT AVG(ld), MAX(ld), MIN(ld)
FROM RADNIK
WHERE pred=40;
```

- Rezultat je jedna vrsta sa prosečnom, maksimalnom i minimalnom vrednošću

1.16. Prikazati ukupnu sumu primljenih premija po radnicima koji rade kao trgovački putnici.

```
SELECT rad, SUM(premija)
FROM RADNIK
WHERE posao='TRGOVACKI PUTNIK'
GROUP BY rad;
```

1.17. Prikazati prosečnu premiju, po preduzećima koja imaju više od 100 radnika

```
SELECT pred, AVG(premija), COUNT(*) AS broj_radnika
FROM RADNIK
GROUP BY pred
HAVING COUNT(*)>100;
```

- **HAVING** - izvršiće se operacija samo za one grupe koje zadovoljavaju uslov u toj klauzuli

1.18. Prikazati nazive preduzeća u kojima su 2000. godine primljeni novi radnici.

```
SELECT P.naziv
FROM RADNIK R, PREDUZECE P
WHERE R.pred=P.pred AND R.datzap>='01/01/2000';
```

1.19. Po gradovima i vrsti posla prikazati ukupan broj radnika koji imaju platu veću od 20000, sortirati po opadajućem redosledu broja radnika.

```
SELECT P.grad, R.posao, COUNT(*) AS broj_radnika
FROM PREDUZECE P, RADNIK R
WHERE R.ld>20000 AND R.pred=P.pred
GROUP BY P.grad, R.posao
ORDER BY COUNT(*) DESC;
```

- Rezultat je dat, na primer, na sledeći način:

P.grad	R.posao	broj_radnika
NS	TRGOVACKI PUTNIK	10
NS	SEKRETAR	100
BG	TRGOVACKI PUTNIK	18
BG	SEKRETAR	184

1.20. Prikazati po nazivu parove preduzeća iz istog grada.

```
SELECT P1.naziv, P2.naziv
FROM PREDUZECE P1, PREDUZECE P2
WHERE P1.grad=P2.grad AND P1.pred>P2.pred;
```

- Uslov P1.pred>P2.pred sprečava da dođe do ponavljanja i spajanja preduzeća sa istom šifrom.

1.21. Prikazati šifre svih radnika koji rade u bar jednom preduzeću u kojem radi Marko Marković.

```
SELECT R.rad
FROM RADNIK R
WHERE R.pred IN (SELECT MM.pred
                 FROM RADNIK MM
                 WHERE MM.ime='MARKO MARKOVIC');
```

1.22. Prikazati nazive svih preduzeća koja se nalaze u istom gradu kao i preduzeće sa šifrom 40.

```
SELECT P1.naziv
FROM PREDUZECE P1
WHERE P1.grad=(SELECT P2.grad
               FROM PREDUZECE P2
               WHERE P2 pred=40);
```

- Unutrašnji **SELECT** (podupit) daje tačno jednu vrednost pa uslov može da bude znak =.

1.23. Prikazati šifre svih radnika koji imaju platu veću od prosečne plate u Beogradu.

```
SELECT R.rad
FROM RADNIK R
WHERE R.ld>(SELECT AVG(R2.ld)
            FROM RADNIK R2, PREDUZECE P
            WHERE R2.pred=P.pred AND P.grad='BEOGRAD');
```

1.24. Prikazati nazive preduzeća u kojima su 2001. godine primljeni novi radnici.

```
SELECT P.naziv
FROM  PREDUZECE P
WHERE EXISTS (SELECT *
              FROM RADNIK R
              WHERE R.pred=P.pred AND R.datzap>='01/01/2001');
```

- Dakle, ako imamo tabele:

PREDUZECE

pred	naziv	grad
1	SF	BEOGRAD
2	MINAKVA	NOVI SAD
3	TELEKOM	BEOGRAD

RADNIK

rad	ime	posao	datzap	ld	premija	pred
121			2001			1
118			2000			2
203			2003			2
221			2000			1
283			2000			3

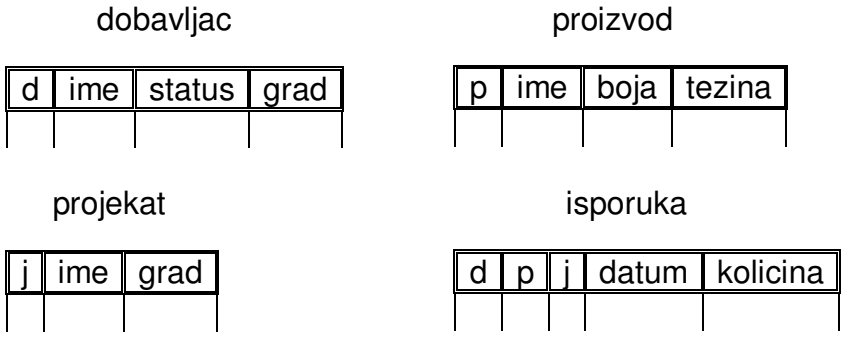
- Rezultat je sledeći:

PREDUZECE

naziv
SF
MINAKVA

2. Data je relaciona šema sa tabelama:

dobavljac (d, ime, status, grad)
proizvid (p, ime, boja, tezina)
projekat (j, ime, grad)
isporuka (d, p, j, datum, kolicina)



2.1. Napisati SQL naredbe kojim se deklarišu navedene relacije.

```
CREATE TABLE dobavljac (d INT NOT NULL AUTO_INCREMENT,
                        ime CHAR(30),
                        status CHAR(30),
                        grad CHAR(30),
                        PRIMARY KEY (d));

CREATE TABLE proizvod (p INT NOT NULL AUTO_INCREMENT,
                       ime CHAR(30), boja CHAR(30),
                       PRIMARY KEY (p));

CREATE TABLE isporuka (d INT DEFAULT '0' NOT NULL,
                       p INT DEFAULT '0' NOT NULL,
                       j  INT DEFAULT '0' NOT NULL,
                       datum DATE,
                       kolicina DECIMAL (10,2));

CREATE TABLE projekat (j INT NOT NULL AUTO_INCREMENT ,
                       ime CHAR(30),
                       grad CHAR(30),
                       PRIMARY KEY (j));
```

2.2. Napisati SQL naredbu kojom se dodaje kolona adresa tabeli dobavljac ispred kolone status.

```
ALTER TABLE dobavljac ADD adresa CHAR(30) AFTER naziv;
```

2.3. Napisati SQL naredbu kojom se težina proizvoda uvećava za 10% za sve proizvode plave boje.

```
UPDATE proizvod SET tezina=1,1 * tezina WHERE boja='plava';
```

2.4. Prikazati podatke o dobavljačima koji nisu ništa isporučili 8. marta 2004.

```
SELECT *
FROM  dobavljac d
WHERE NOT EXISTS (SELECT *
                  FROM isporuka i, dobavljac d2
                  WHERE i.d=d2.d AND i.datum='08/03/2004');
```

2.5. Prikazati upit za pronalaženje ukupne količine proizvoda pod šifrom 1, koju iporučuje dobavljač pod šifrom 3.

```
SELECT SUM (i.kolicina)
FROM  isporuka i
WHERE i.p=1 AND i.d=3;
```

2.6. Napisati upit za pronalaženje ukupne količine proizvoda pod imenom TELEFON koju isporučuje dobavljač pod imenom MOTOROLA.

```
SELECT SUM (i.kolicina)
FROM  isporuka i, dobavljac d, proizvod p
WHERE i.d=d.d AND i.p=p.p AND d.ime='MOTOROLA' AND p.ime='TELEFON';
```

2.7. Prikazati sve parove gradova, tako da dobavljač u prvom gradu isporučuje neki proizvod projektu u drugom gradu.

```
SELECT DISTINCT d.grad, j.grad
FROM dobavljac d, projekat j, isporuka i
WHERE i.d=d.d AND i.j=j.j AND d.grad<>j.grad;
```

2.8. Prikazati šifre projekata koje snabdeva isključivo dobavljač pod šifrom 1.

```
SELECT DISTINCT i1.j
FROM  isporuka i1
WHERE i1.d=1 AND NOT EXISTS (SELECT *
                             FROM isporuka i2
                             WHERE i2.j=i1.j AND i2.d<>1);
```

isporuka i1

i1.d	i1.p	i1.j	i1.datum	i1.kolicina

isporuka i2

i2.d	i2.p	i2.j	i2.datum	i2.kolicina

isporuka

d	p	j	datum	kolicina
5		3	10.10.2004.	100
1		3	11.10.2004.	50
8		4	09.09.2004	130
1		5	08.08.2004	80

- Može i iz 'projekat p', bez **DISTINCT**, ali to ne daje tačno rešenje, jer bismo dobili i selektovane projekte za koje nije izvršena nikakva isporuka.

2.9. Pronaći nazive projekata koji su kupili više od 10 proizvoda jednog dobavljača i imena tih dobavljača i broj kupljnih proizvoda (pod brojem proizvoda podrazumeva se broj isporuke).

```
SELECT j.ime, d.ime, COUNT(*) AS broj
FROM  projekat j, dobavljac d, isporuka i
WHERE i.j=j.j AND d.d=i.d
GROUP BY d.ime, j.ime
HAVING COUNT(*)>10;
```

2.10. Prikazati spisak proizvoda i srednje vrednosti iporučene količine za proizvode CRVENE boje, koje isporučuju samo dobavljači iz LONDONA.

```
SELECT p.ime, AVG(i.kolicina)
FROM proizvod p, isporuka i
WHERE p.p=i.p AND p.boja='CRVENA' AND NOT EXISTS
                    (SELECT *
                     FROM dobavljac d, isporuka i1
                     WHERE d.d=i1.d AND d.grad <>'LONDON')
GROUP BY p.ime;
```

2.11. Prikazati spisak gradova u kojima postoji neki dobavljač i više od jednog projekta ili neki projekat i više od jednog dobavljača.

```
SELECT DISTINCT d.grad
FROM  dobavljac d
WHERE d.grad IN (SELECT j.grad
                 FROM projekat j
                 GROUP BY j.grad
                 HAVING COUNT(*)>1)
UNION
SELECT DISTINCT j1.grad
FROM  projekat j1
WHERE j1.grad IN (SELECT d1.grad
                  FROM dobavljac d1
                  GROUP BY d1.grad
                  HAVING COUNT (*)>1);
```

2.12. Prikazati isporučene ukupne količine proizvoda svakog dobavljača projektima iz Novog Sada koji se snabdevaju isključivo od dobavljača iz Beograda.

```
SELECT SUM(i.kolicina), p.p
FROM isporuka i, projekat j, proizvod p
WHERE i.j=j.j AND p.p=i.p AND j.grad='NOVI SAD' AND NOT EXISTS
                        (SELECT *
                         FROM dobavljac d, isporuka i1
                         WHERE d.d=i1.d AND d.grad <>'BEOGRAD' AND i1.p=p.p )
GROUP BY p.p
ORDER BY p.p;
```

2.13. Pronaći koliko je svaki projekat kupio određenih proizvoda crvene boje, koliko je u jednoj isporuci bilo najviše i najmanje istih proizvoda i koliko je prosečno i ukupno isporučeno istih. Izostaviti iz razmatranja isporuke koje su se obavile u istom gradu. Ne mešati iste proizvode različitih dobavljača.

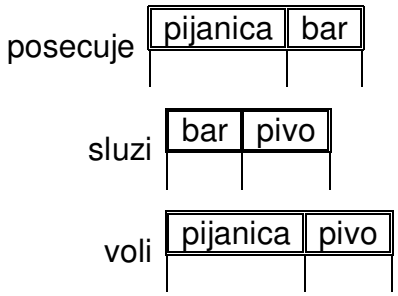
```
SELECT COUNT(*), MAX(i.kolicina), MIN(i.kolicina), AVG(i.kolicina), SUM(i.kolicina), d.d
FROM isporuka i, proizvod p, dobavljac d
WHERE i.p=p.p AND i.d=d.d AND p.boja='CRVENA' AND d.grad IN
                       (SELECT d1.grad
                        FROM dobavljac d1
                        WHERE d1.grad <>d.grad)
GROUP BY d.d
ORDER BY d.d;
```

2.14. Prikazati podatke o dobavljačima koji ništa nisu isporučili 5. maja 2003. godine. Sortirati po gradu dobavljača u opadajućem redosledu.

```
SELECT *
FROM dobavljac d
WHERE NOT EXISTS  (SELECT *
                           FROM isporuka i
                           WHERE d.d=i.d AND i.datum='05/05/2003')
ORDER BY d.grad DESC;
```

3. Data je relaciona šema:

posecuje (pijanica, bar)
sluzi (bar, pivo)
voli (pijanica, pivo)



3.1. Kreirati bazu podataka pijanica, sve tabele i unos podataka.

```
CREATE DATABASE pijanice;
USE pijanice;
CREATE TABLE posecuje (pijanica CHAR(30), bar CHAR(30));
CREATE TABLE sluzi (bar CHAR(30), pivo CHAR(30));
CREATE TABLE voli (pijanica CHAR(30), pivo CHAR(30));
INSERT INTO posecuje VALUES ('MIKA','DOMOVINA');
INSERT INTO sluzi VALUES ('SANSA','LAV');
INSERT INTO voli VALUES  ('MIKA','LAV');  ...
```

3.2. Naći pijanice koji posećuju bar u kome se služi pivo Lav.

```
SELECT DISTINCT p.pijanica
FROM  posecuje p, sluzi s
WHERE p.bar=s.bar AND s.pivo='LAV';
```

3.3. Naći pijanice koje posećuju bar u kome se služi pivo koje oni vole.

```
SELECT DISTINCT p.pijanica
FROM  posecuje p, sluzi s, voli v
WHERE p.pijanica=v.pijanica AND p.bar=s.bar AND s.pivo=v.pivo;
```

3.4. Naći pijanice koje piju u istom baru sa pijanicama koji vole Nikšićko pivo.

```
SELECT DISTINCT p1.pijanica
FROM  posecuje p1
WHERE p1.bar  IN (SELECT p2.bar
                  FROM posecuje p2, voli v
                  WHERE p2.pijanica=v.pijanica AND
                        v.pivo='NP' AND
                        p1.pijanica<>p2.pijanica);
```

3.5. Naći pijanice koje ne posećuju bar u kome se služi pivo koje oni vole.

```
SELECT DISTINCT v.pijanica
FROM  sluzi s, voli v
WHERE s.pivo=v.pivo AND NOT EXISTS (SELECT *
                                   FROM  posecuje p
                                   WHERE p.pijanica=v.pijanica AND p.bar=s.bar);
```

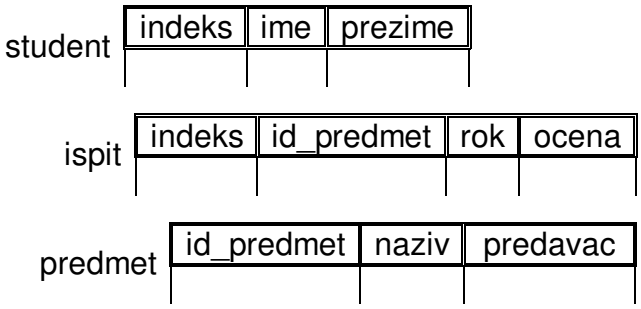
3.6. Naći pijanice koji piju u istom baru sa pijanicama koji vole pivo koje taj bar služi i pivo Jelen.

4. Data je relaciona šema:

student (indeks, ime, prezime)

ispit (indeks, id_premet, rok, ocena)

predmet (id_predmet, naziv, predavac)



4.1. Kreirati bazu podataka studenti i sve tabele.

```
CREATE DATABASE studenti;

USE studenti;

CREATE TABLE student (indeks CHAR(8) NOT NULL,
                      ime CHAR(30) NOT NULL,
                      prezime CHAR(30) NOT NULL,
                      PRIMARY KEY (indeks));

CREATE TABLE ispit (indeks CHAR(8) NOT NULL,
                    id_predmet INT NOT NULL,
                    rok CHAR(20) NOT NULL,
                    ocena INT NOT NULL);

CREATE TABLE predmet (id_predmet INT NOT NULL,
                      naziv CHAR(50) NOT NULL,
                      predavac CHAR(30) NOT NULL,
                      PRIMARY KEY (id_predmet));
```

4.2. Prikazati broj indeksa, ime, naziv predmeta, rok i ocenu i sortirati ih po predmetu.

```
SELECT s.indeks, s.ime, p.naziv, i.rok, i.ocena
FROM  student s, predmet p, ispit i
WHERE s.indeks=i.indeks AND p.id_predmet=i.id_predmet AND i.ocena>5
ORDER BY p.naziv;
```

4.3. Prikazati broj indeksa, ime, prezime, srednju, maksimalnu i minimalnu ocenu i broj položenih ispita. Sortirati po broju položenih ispita u rastućem redosledu i po prosečnoj oceni u opadajućem redosledu.

```
SELECT s.indeks, s.ime, s.prezime,
       AVG(i.ocena), MAX(i.ocena), MIN(i.ocena), COUNT(*)
FROM  student s, ispit i
WHERE s.indeks=i.indeks AND i.ocena>5
GROUP BY s.indeks, s.ime, s.prezime
ORDER BY COUNT(*) ASC, AVG(i.ocena) DESC;
```

4.4. Prikazati broj indeksa, ime, prezime, srednju, maksimalnu i minimalnu ocenu i broj položenih ispita za studente koji imaju više od tri položena ispita; sortirati po broju položenih ispita u rastućem redosledu i po prosečnoj oceni u opadajućem redosledu.

```
SELECT s.indeks, s.ime, s.prezime,
        AVG(i.ocena), MAX(i.ocena), MIN(i.ocena), COUNT(*)
FROM  student s, ispit i
WHERE s.indeks=i.indeks AND i.ocena>5
GROUP BY s.indeks, s.ime, s.prezime
HAVING COUNT(*)>3
ORDER BY COUNT(*) ASC, AVG(i.ocena) DESC;
```

4.5. Prikazati po predavacima srednju, maksimalnu, minimalnu ocenu i broj položenih ispita.

```
SELECT p.predavac, AVG(i.ocena) AS srednja_ocena,
        MAX(i.ocena) AS max_ocena, MIN(i.ocena) AS min_ocena, COUNT(*) AS broj
FROM  ispit i, predmet p
WHERE  i.id_predmet=p.id_predmet  AND i.ocena>5
GROUP BY p.predavac;
```

4.6. Prikazati prosečnu ocenu i broj položenih ispita za sve studente koji su položili više od 10 ispita, ne računajući predmet pod nazivom 'MATEMATIKA'. Rezultat sortirati po prosečnoj oceni i u opadajućem redosledu.

```
SELECT  s.indeks, s.ime, s.prezime, AVG(i.ocena) AS prosek, COUNT(*) AS broj
FROM  student s, ispit i, predmet p
WHERE  s.indeks=i.indeks AND p.id_predmet=i.id_predmet  AND
        p.naziv <>'MATEMATIKA' AND i.ocena>5
GROUP BY s.indeks, s.ime, s.prezime
HAVING COUNT(*)>10
ORDER BY AVG(i.ocena) DESC;
```

4.7. Pronaći predavače kod kojih su položeni ispiti sa ocenom 9.

```
SELECT DISTINCT predavac
FROM  predmet p, ispit i
WHERE  p.id_predmet=i.id_predmet  AND i.ocena=9;
```

4.8. Pronaći prosečnu, maksimalnu i minimalnu ocenu za svakog studenta koji ima prosečnu ocenu veću ili jednaku 9.

```
SELECT  s.indeks, s.ime, s.prezime, AVG(i.ocena) AS prosek,
        MAX(i.ocena) AS max_ocena, MIN(i.ocena) AS  min_ocena
FROM  student s, ispit i
WHERE  s.indeks=i.indeks
GROUP BY s.indeks, s.ime, s.prezime
HAVING AVG(i.ocena)>=9;
```

- **HAVING** služi za agregatne uslove koji ne smeju da se stavljaju u **WHERE** klauzulu

4.9. Pronaći prosečnu, maksimalnu i minimalnu ocenu za svakog studenta koji ima prosečnu ocenu veću ili jednaku 9, ne računajući ocene kod predavača Zorana Avramovića i sortirati po prezimenu i imenu studenta.

```
SELECT  s.indeks, s.ime, s.prezime, AVG(i.ocena) AS prosek,
        MAX(i.ocena) AS max_ocena, MIN(i.ocena) AS  min_ocena
FROM  student s, ispit i, predmet p
WHERE  s.indeks=i.indeks AND predavac <>'ZORAN AVRAMOVIC'
GROUP BY s.indeks, s.ime, s.prezime
HAVING AVG(i.ocena)>=9
ORDER BY s.prezime, s.ime;
```

4.10. Promenjena je šifra za Vazdušni smer sa VZ na VAZ. Ažurirati bazu student u skladu sa nastalim promenama.

```
UPDATE student SET smer='VAZ' WHERE smer='VZ';
```

4.11. Student sa brojem indeksa 98-1-006 poništio je položeni ispit iz predmeta sa šifrom 1. Ažurirati bazu podataka u skladu sa nastalim promenama.

```
DELETE FROM ispit WHERE indeks='98-1-006' AND id_predmet=1;
```

4.12. Za svaki položeni ispit potrebno je voditi evidenciju o datumu polaganja. Ažurirajte tabelu ispit tako da se dodaje još jedno polje pod nazivom datum, iza polja rok.

```
ALTER TABLE ispit ADD datum DATE AFTER rok;
```

4.13. Napisati SQL naredbu kojom se dodaje kolona smer tabeli student.

```
ALTER TABLE student ADD smer CHAR(4);
```

4.14. Prikazati broj indeksa, ime, prezime i prosečnu ocenu za sve studente koji su položili barem jedan ispit kod profesora Zorana Avramovića.

```
SELECT  s.indeks, s.ime, s.prezime, AVG(i.ocena) AS prosek
FROM  student s, ispit i
WHERE  s.indeks=i.indeks AND s.indeks IN
          (SELECT DISTINCT i1.indeks
           FROM ispit i1, predmet p
           WHERE  i1.id_predmet=p.id_predmet AND
                    p.predavac='ZORAN AVRAMOVIC' AND i1.ocena>5)
GROUP BY s.indeks, s.ime, s.prezime
HAVING COUNT(*)>=1;
```

4.15. Prikazati rangiranu listu po prosečnoj oceni svih položenih ispita za studente koji imaju prosek najmanje 7.

```
SELECT  s.indeks, s.ime, s.prezime, i.id_predmet, AVG(i.ocena) AS prosek
FROM  student s, ispit i
WHERE  s.indeks=i.indeks AND i.ocena>5
GROUP BY i.id_predmet, s.prezime, s.ime, s.indeks
HAVING AVG(i.ocena)>=7
ORDER BY AVG(i.ocena) DESC;
```

4.16. Napisati upit koji pronalazi broj studenata po predmetima koji nisu položili ispit iz prva tri polaganja. Prikazati samo podatke za one predmete za koje postoji više od 10 studenata koji nisu uspeli da ga polože iz prva tri pokušaja. Ne uzimati u obzir studente pete godine. Sortirati po opadajućem redosledu broja studenata.

- U ovom slučaju relacije student i ispit su izmenjene dodavanjem atributa:

```
student (indeks,ime, prezime, smer, godina)
ispit (indeks, id_predmet, datum, rok, ocena, prijava)
predmet (id_predmet, naziv, predavac)
```

```
SELECT  p.naziv, COUNT(*) AS broj
FROM  ispit i, predmet p, student s
WHERE  i.indeks=s.indeks AND i.id_predmet=p.id_predmet AND s.godina<5 AND
          i.prijava>=3 AND i.ocena=5
GROUP BY p.naziv
HAVING COUNT(*)>10
ORDER BY COUNT(*) DESC;
```

4.17. Napisati SQL upit za prepravljanje ocena svih studenata koji su položili u oktobarskom roku 2002. ispit PIS, sa ocene 7 na ocenu 8.

```
UPDATE ispit SET ocena=8
WHERE ocena=7 AND id_predmet='PIS' AND
        datum BETWEEN '01/01/2002' AND '31/12/2002' AND
        rok='OKTOBARSKI';
```

4.18. Prikazati prosečnu ocenu i broj položenih ispita za sve studente koji imaju najmanje prosek 8 i koji su položili barem jedan ispit iz predmeta na kojem je ostvareno manje od tri desetke. Sortirati po prosečnoj oceni u opadajućem redosledu.

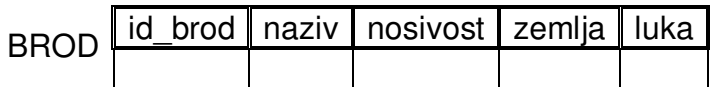
```
SELECT AVG(i.ocena) AS prosek, COUNT(*) AS broj, s.ime, s.prezime, s.indeks
FROM  ispit i, student s, predmet p
WHERE  s.indeks=i.indeks AND p.id_predmet IN (SELECT i1.id_predmet
                                              FROM ispit i1
                                              WHERE i1.ocena=10
                                              GROUP BY  i1.id_predmet
                                              HAVING COUNT(*)<3)
GROUP BY s.indeks, s.prezime, s.ime
HAVING AVG(i.ocena)>=8 AND COUNT(*)>=1
ORDER BY AVG(i.ocena) DESC;
```

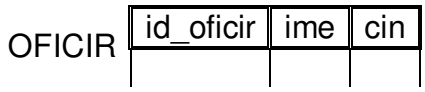
4.19. Prikazati imena onih profesora koji drže barem jedan predmet na kome je samo jedan student položio ispit iz prvog pokušaja.

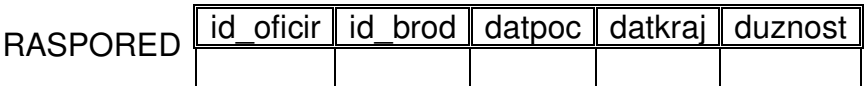
```
SELECT p.predavac
FROM  predmet p
WHERE  p.id_predmet IN (SELECT i.id_predmet
                        FROM ispit i
                        WHERE i.ocena>5 AND i.prijava=1
                        GROUP BY i.id_predmet, i.prijava
                        HAVING  COUNT(*)=1)
GROUP BY p.predavac
HAVING COUNT(*)>=1;
```

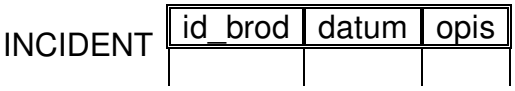
5. Data je relaciona šema:

BROD (id_brod, naziv, nosivost, zemlja, luka)
OFICIR (id_oficir, ime, cin)
RASPORED (id_oficir, id_brod, datpoc, datkraj, duznost)
INCIDENT (id_brod, datum, opis)









5.1. Napisati SQL naredbe kojim se deklarišu navedene relacije.

```
CREATE TABLE BROD (id_brod INTEGER AUTO_INCREMENT NOT NULL,
                   naziv CHAR(10),
                   nosivost INTEGER,
                   zemlja CHAR(20)
                   luka CHAR(20)
                   PRIMARY KEY (id_brod));

CREATE TABLE RASPORED (id_oficir INTEGER AUTO_INCREMENT NOT NULL,
                       id_brod INTEGER NOT NULL,
                       datpoc DATE,
                       datkraj DATE,
                       duznost CHAR(20));

CREATE TABLE OFICIR (id_oficir INTEGER AUTO_INCREMENT NOT NULL,
                     ime CHAR(30),
                     cin CHAR(20)
                     PRIMARY KEY(id_oficir));

CREATE TABLE INCIDENT (id_brod  INTEGER AUTO_INCREMENT NOT NULL,
                       datum DATE
                       opis CHAR(20));
```

5.2. Prikazati imena kapetana čiji su se brodovi nasukali u periodu od 5. marta do 2. juna 2002.

```
SELECT DISTINCT O.ime
FROM  OFICIR O, INCIDENT I, RASPORED R
WHERE  O.id_oficir=R.id_oficir AND I.id_brod=R.id_brod AND O.cin='KAPETAN' AND
       I.datum BETWEEN '05/03/2002' AND '02/06/2002' AND I.opis='NASUKAN';
```

5.3. Prikazati imena svih oficira koji su zapovedali brodom 'BISMARK'.

```
SELECT DISTINCT O.ime
FROM  OFICIR O, BROD B, RASPORED R
WHERE B.id_brod=R.id_brod AND O.id_oficir=R.id_oficir AND
        B.naziv='BISMARK' AND  R.duznost='ZAPOVEDNIK';
```

5.4. Prikazati imena oficira koji su zapovedali brodom 'BISMARK' i na njemu nisu imali nijedan incident.

```
SELECT DISTINCT O.ime
FROM  OFICIR O, BROD B, RASPORED R
WHERE B.id_brod=R.id_brod AND O.id_oficir=R.id_oficir AND
        B.naziv='BISMARK' AND  R.duznost='ZAPOVEDNIK'
        AND R.id_oficir NOT IN (SELECT R2.id_oficir
                                FROM INCIDENT I, RASPORED R2
                                WHERE I.id_brod=R2.id_brod AND
                                      I.datum BETWEEN R2.datpoc AND R2.datkraj);
```

5.5. Prikazati imena onih kapetana koji su na svakom od brodova kojima su zapovedali, imali po neki incident.

```
SELECT O.ime
FROM  OFICIR O, RASPORED R, BROD B
WHERE  R.id_oficir=O.id_oficir AND R.id_brod=B.id_brod AND O.cin='KAPETAN' AND
        R.duznost='ZAPOVEDNIK' AND B.id_brod IN (SELECT I.id_brod
                                                 FROM INCIDENT I
                                                 GROUP BY I.id_brod
                                                 HAVING COUNT(*)>=1);
```

5.6. Prikazati imena svih oficira koji su bili raspoređeni na brodovima koji su se nasukali u periodu od 5. do 25. juna 2004. godine.

```
SELECT DISTINCT O.ime
FROM  OFICIR O, INCIDENT I, RASPORED R
WHERE  O.id_oficir=R.id_oficir AND I.id_brod=R.id_brod AND I.opis='NASUKAN' AND
        I.datum BETWEEN '05/06/2004' AND '25/06/2004';
```

5.7. Napisati SQL naredbu kojom se ukljanjaju svi incidenti sa opisom požar.

```
DELETE FROM INCIDENT WHERE opis='POZAR';
```

5.8. Prikazati imena kapetana koji su zapovedali brodovima, naziv broda na kojem su zapovedali, vremenski period u kojem su zapovedali, koliko incidenata su imali na tom brodu i kada.

```
SELECT O.ime, B.naziv, R.datpoc, R.datkraj, COUNT(*) AS broj_incidenata, I.datum
FROM OFICIR O, BROD B, RASPORED R, INCIDENT I
WHERE O.id_oficir=R.id_oficir AND B.id_brod=R.id_brod AND
        I.id_brod=R.id_brod  AND O.cin='KAPETAN' AND
        R.duznost='ZAPOVEDNIK' AND
        I.datum BETWEEN R.datpoc AND R.datkraj
GROUP BY O.ime, B.naziv, R.datpoc, R.datkraj, I.datum;
```

6. Organizacija podataka objekata u trodimenzionalnom prostoru je sledeća: objekat se sastoji od niza poligona, poligon se sastoji od niza tačaka u prostoru (vertex), verteks se sastoji od koordinata x, y i z.

6.1. Projektujte bazu podataka u kojoj će se čuvati ovi podaci.

CREATE DATABASE 3D_OBJEKTI**;**

USE 3D_OBJEKTI**;**

6.2. Za svaku tabelu napišite SQL upite za kreiranje tabele.

CREATE TABLE VERTEKS **(**x **INTEGER**, y **INTEGER**, z **INTEGER**,
id_verteks **INTEGER NOT NULL**,
id_poligon **INTEGER NOT NULL**,
id_objekat **INTEGER NOT NULL;**
PRIMARY KEY (id_vertex**);**

CREATE TABLE POLIGON **(**id_poligon **INTEGER NOT NULL**,
id_objekat **INTEGER NOT NULL**,
PRIMARY KEY (id_poligon**);**

CREATE TABLE OBJEKAT **(**id_objekat **INTEGER NOT NULL**,
naziv **CHAR(**20**) NOT NULL**,
PRIMARY KEY (id_objekat**));**

6.3. Za svaku tabelu napišite po jedan SQL upit za unos podataka.

INSERT INTO VERTEKS **VALUES (**10, 10, 10, 1, 1, 1**);**
INSERT INTO POLIGON **VALUES (**1, 1**);**
INSERT INTO OBJEKAT **VALUES (**1, 'LOPTA'**);**

6.4. Neka je data sfera u prostoru sa centrom (x_0, y_0, z_0) i poluprečnikom r_0. Napišite upit koji pronalazi sve objekte koji sadrže najmanje 5 verteksa koji se nalaze u prostoru date sfere. Prikažite identifikator objekta i broj verteksa koji se nalaze u sferi. Prikaz sortirati u opadajućem redosledu broja obuhvaćenih verteksa.

```
SELECT O.id_objekat, O.naziv, COUNT(*) AS broj_verteksa
FROM  OBJEKAT O
WHERE  O.naziv='SFERA' AND O.id_objekat IN
            (SELECT O1.id_objekat
             FROM OBJEKAT O1, POLIGON P, VERTEKS V
             WHERE V.id_objekat=O1.id_objekat AND
                   V.id_poligon=P.id_poligon AND
                   V.x BETWEEN (x0 – r0) AND (x0 + r0) AND
                   V.y BETWEEN (y0 – r0) AND (y0 + r0) AND
                   V.z BETWEEN (z0 – r0) AND (z0 + r0)
            GROUP BY V.id_verteks, O1.id_objekat
            HAVING COUNT(*)>= 5)
GROUP BY O.id_objekat, O.naziv
ORDER BY COUNT(*) DESC;
```