

Svim nastavnicima čiji je koeficijent manji od prosječnog smanjiti koeficijent za 10%.

Check column mode: 4. PERMISSIVE: try 3 (to match by names); if not - try 2 (use column order).

```
1 UPDATE nastavnik
2 SET koef = koef * 0.9
3 WHERE koef < (SELECT AVG(koef) FROM nastavnik);
```

Run

Save

Result (186 rows)

😊 Correct! Well done!



masterexam03

Primarni ključ u relaciji **predmetGrupa** je skup atributa: {sifPredmet, akGodina, oznGrupa}. **Jednom** SQL naredbom osigurajte entitetski integritet i integritet ključa u toj relaciji. Također osigurati referencijski integritet među relacijama **upisanPredmet** i **predmetGrupa**.

Napomena: višestruke SQL naredbe odvojiti znakom ";". Koristiti sintaksu za naknadnu izmjenu postojeće tablice: 'ALTER TABLE ime_tablice ADD CONSTRAINT ime_ogranicenja opis_ogranicenja'.

Check column mode: 4. PERMISSIVE: try 3 (to match by names); if not - try 2 (use column order).

```
1 ALTER TABLE predmetGrupa
2 ADD CONSTRAINT pkPredmetGrupa
3     PRIMARY KEY(sifPredmet, akGodina, oznGrupa);
4
5 ALTER TABLE upisanPredmet
6 ADD CONSTRAINT fkUpisanPredmet
7     FOREIGN KEY(sifPredmet, akGodina, oznGrupa) REFERENCES predmetGrupa(sifPredmet, akGodina, oznGrupa);
```

Run

Save

Result (3 rows)

😊 Correct! Well done!



masterexam03

Za relaciju `student` kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

```
SELECT * FROM student WHERE datumrod > '01.01.1996' ORDER BY datumrod;
```

```
SELECT * FROM student  
WHERE prezimeStudent = 'Salopek'  
AND imeStudent = 'Krešimir';
```

```
SELECT * FROM student WHERE prezimeStudent = 'Salopek';
```

```
SELECT * FROM student ORDER BY datumrod DESC;
```

```
SELECT * FROM student WHERE oib = '08707549086';
```

```
SELECT * FROM student WHERE datumrod > '01.01.1996';
```

```
SELECT * FROM student ORDER BY datumrod, jmbag;
```

Check column mode: 4. PERMISSIVE: try 3 (to match by names); if not - try 2 (use column order).

```
1 CREATE INDEX index1 ON student(datumRod, jmbag);  
2 CREATE INDEX index2 ON student(prezimeStudent, imeStudent);  
3 CREATE INDEX index3 ON student(oib);
```

Run

Save

Result (10 rows)

😊 Correct! Well done!



masterexam07

Napraviti 2 virtualne relacije:

1. Napraviti virtualnu relaciju **stanBr3** sa shemom relacije **STANBR3 = pbrstan, broj3** koja će omogućiti pregled broja studenata koji su s **ocjenom 3** položili ispit čiji je datum roka bio u **2019. kalendarskoj godini** po **poštanskom broju stanovanja**.

Primjer rezultata:

```
SELECT * FROM stanBr3;
```

pbrstan	broj3
99999	1
(null)	5
11111	3
...	...

Primijetite mogućnost postojanja *n-torke* koja predstavlja studente čije mjesto stanovanja nije poznato.

2. Napraviti virtualnu relaciju **mjestoBr3** sa shemom relacije **MJESTOBR3 = nazmjesto, broj3** koja će po **SVIM mjestima** omogućiti pregled broja studenata koji su s **ocjenom 3** položili ispit čiji datum roka bio u **2019. kalendarskoj godini** i koji **stanuju** u tom mjestu. Uključiti i studente čije mjesto stanovanja nije poznato (pogledajte primjer).

Virtualnu relaciju **mjestoBr3** obavezno napraviti pomoću relacije **stanBr3**.

Primjer rezultata:

```
SELECT * FROM mjestoBr3;
```

nazmjesto	broj3
Zadar	10
(null)	20
Zagreb	5
Hum	(null)
Makarska	(null)
...	...

Primijetite mogućnost postojanja *n-torke* koja predstavlja studente čije mjesto stanovanja nije poznato i mogućnost postojanja *n-torki* koje predstavljaju mjesta za koja nema položenih predmeta s ocjenom 3 u zadanoj kalendarskoj godini.

```
1 CREATE VIEW stanBr3 (pbrStan, broj3) AS
2 SELECT pbrStanStudent, COUNT(DISTINCT student.jmbag) AS broj3
3 FROM ispit NATURAL JOIN student
4 WHERE datumRok BETWEEN '2018/10/01' AND '2019/09/30'
5     AND ocjena = 3
6 GROUP BY pbrStanStudent;
7
8 CREATE VIEW mjestoBr3 (nazMjesto, broj3) AS
9 SELECT nazMjesto, Broj3
10 FROM stanBr3 FULL OUTER JOIN mjesto ON stanBr3.pbrStan = mjesto.pbr;
```

Run

Save

Result (285 rows)

😊 Correct! Well done!



masterexam06