b) Kreiranje tabela ▪ Kreiranje sekvenci ▪ Kreiranje ograničenja

--vrsta poslovnice, kreiramo sekvencu i tabelu sa ogranicenjima

```
CREATE SEQUENCE vrsta_poslovnice_sekv

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;
```

```
CREATE TABLE vrsta_poslovnice (vrsta_poslovnice_id NUMBER NOT NULL, naziv VARCHAR2(50) NOT NULL, CONSTRAINT prim_v_p PRIMARY KEY(vrsta_poslovnice_id));
```

--zaposlenici, kreiramo sekvencu i tabelu sa ogranicenjima

```
CREATE SEQUENCE zaposlenici_sekv

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;
```

```
CREATE TABLE zaposlenici(zaposlenik_id NUMBER NOT NULL, odjel VARCHAR2(50) NOT NULL, funkcija VARCHAR2(50) NOT NULL, naziv VARCHAR2(50) NOT NULL, CONSTRAINT prim_zap PRIMARY KEY(zaposlenik_id));
```

--drzava, kreiramo sekvencu i tabelu sa ogranicenjima

```
CREATE SEQUENCE drzava_sekv

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;
```

CREATE TABLE drzava(drzava_id NUMBER NOT NULL, naziv VARCHAR2(50) NOT NULL, porez NUMBER(2,2) NOT NULL,CONSTRAINT prim_drz PRIMARY KEY(drzava_id));

--statistika, kreiramo sekvencu i tabelu sa ogranicenjima

CREATE SEQUENCE statistika_sekv

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE TABLE statistika(statistika_id NUMBER NOT NULL,igraci VARCHAR2(100), timovi VARCHAR2(50),liga VARCHAR2(50), mec NUMBER, CONSTRAINT prim_stat PRIMARY KEY(statistika_id));

--informacije dodatne o vrsti opklade, kreiramo sekvencu i tabelu sa ogranicenjima

CREATE SEQUENCE opklada_info_sekv

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE TABLE opklada_info(opklada_info_id NUMBER NOT NULL, broj_uplacenih_listica NUMBER, pogodnosti VARCHAR2(100),statistika_id NUMBER,CONSTRAINT prim_op_inf PRIMARY KEY(opklada_info_id),CONSTRAINT opk_inf_fk FOREIGN KEY(statistika_id) REFERENCES statistika(statistika_id));

--vrsta opklade, kreiramo sekvencu i tabelu sa ogranicenjima

CREATE SEQUENCE vrsta_sekv

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE TABLE vrsta(vrsta_id NUMBER NOT NULL,naziv VARCHAR2(50) NOT NULL, broj_subjekata NUMBER NOT NULL , opklada_info_id NUMBER,CONSTRAINT prim_vrst PRIMARY KEY(vrsta_id),CONSTRAINT vr_fk FOREIGN KEY(opklada_info_id) REFERENCES opklada_info(opklada_info_id));

--listici, kreiramo sekvencu i tabelu sa ogranicenjima

CREATE SEQUENCE listici_sekv

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE TABLE listici(listic_id NUMBER NOT NULL, ishod VARCHAR2(50) NOT NULL,parovi NUMBER , vrsta_id NUMBER,CONSTRAINT prim_lis PRIMARY KEY (listic_id),CONSTRAINT li_fk FOREIGN KEY(vrsta_id) REFERENCES vrsta(vrsta_id));

--klijenti, kreiramo sekvencu i tabelu sa ogranicenjima

CREATE SEQUENCE klijenti_sekv

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE TABLE klijenti(klijent_id NUMBER NOT NULL, naziv VARCHAR(50) NOT NULL, racun NUMBER , listic_id NUMBER,CONSTRAINT prim_kli PRIMARY KEY(klijent_id),CONTRAINT kli_fk FOREIGN KEY(listic_id) REFERENCES listici(listic_id));

--poslovnica, kreiramo sekvencu i tabelu sa ogranicenjima

```
CREATE SEQUENCE poslovnica_sekv

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;


CREATE TABLE poslovnica(poslovnica_id NUMBER NOT NULL, zaposlenik_id NUMBER,klijent_id
NUMBER, drzava_id NUMBER ,stanje_novca NUMBER NOT NULL ,vrsta_poslovnice_id NUMBER,
CONSTRAINT prim_posl PRIMARY KEY(poslovnica_id), CONSTRAINT pos_zap_fk FOREIGN
KEY(zaposlenik_id) REFERENCES zaposlenici(zaposlenik_id), CONSTRAINT pos_kli_fk FOREIGN
KEY(klijent_id) REFERENCES klijenti(klijent_id), CONSTRAINT pos_drz_fk FOREIGN KEY(drzava_id)
REFERENCES drzava(drzava_id), CONSTRAINT pos_vrst_posl_fk FOREIGN KEY(vrsta_poslovnice_id)
REFERENCES vrsta_poslovnice(vrsta_poslovnice_id));
```

c) Popunjavanje tabela (u prosjeku 10 slogova po tabeli)

--vrsta poslovnice, mogu biti obicne, regionalne, i glavna

```
INSERT INTO vrsta_poslovnice

VALUES(vrsta_poslovnice_sekv.NEXVAL,'obicna');

INSERT INTO vrsta_poslovnice

VALUES(vrsta_poslovnice_sekv.NEXVAL,'regionalna');

INSERT INTO vrsta_poslovnice

VALUES(vrsta_poslovnice_sekv.NEXVAL,'glavna');
```

--zaposlenici, unosimo 10 zaposlenika sa razlicitim funkcijama i odjelima

```
INSERT INTO  zaposlenici

VALUES(zaposlenici_sekv.NEXTVAL,'racunovodstvo','racunovodja','Mujo Mujic');

INSERT INTO  zaposlenici

VALUES(zaposlenici_sekv.NEXTVAL,'pult','blagajnica','Maja Majic');
```

```
INSERT INTO  zaposlenici
VALUES(zaposlenici_sekv.NEXTVAL,'cistacice','cistacica','Fata Fatic');
INSERT INTO  zaposlenici
VALUES(zaposlenici_sekv.NEXTVAL,'racunovodstvo','racunovodja','Muki Mukic');
INSERT INTO  zaposlenici
VALUES(zaposlenici_sekv.NEXTVAL,'security','security','Suljo Suljic');
INSERT INTO  zaposlenici
VALUES(zaposlenici_sekv.NEXTVAL, 'security','security',Huso Husic');
INSERT INTO  zaposlenici
VALUES(zaposlenici_sekv.NEXTVAL,'marketing','dizajner','Haris Hari');
INSERT INTO  zaposlenici
VALUES(zaposlenici_sekv.NEXTVAL,'pult','blagajnica','Ajla Ajli');
INSERT INTO  zaposlenici
VALUES(zaposlenici_sekv.NEXTVAL,'pult','blagajnica','Emina Emi');
INSERT INTO  zaposlenici
VALUES(zaposlenici_sekv.NEXTVAL,'marketing','media planer','Edo Edi');
--drzava, unosimo 10 drzava sa razlicitim porezima
INSERT INTO drzava
VALUES(drzava_sekv.NEXTVAL,'Bosna i Hercegovina',17);
INSERT INTO drzava
VALUES(drzava_sekv.NEXTVAL,'Srbija',18);
INSERT INTO drzava
VALUES(drzava_sekv.NEXTVAL,'Hrvatska',10);
INSERT INTO drzava
VALUES(drzava_sekv.NEXTVAL,'Njemacka',8);
INSERT INTO drzava
VALUES(drzava_sekv.NEXTVAL,'Austrija',11);
INSERT INTO drzava
VALUES(drzava_sekv.NEXTVAL,'Slovenija',16);
```

```
INSERT INTO drzava
VALUES(drzava_sekv.NEXTVAL,'Italija',15);
INSERT INTO drzava
VALUES(drzava_sekv.NEXTVAL,'Norveska',16);
INSERT INTO drzava
VALUES(drzava_sekv.NEXTVAL,'Velika Britanija',18);
INSERT INTO drzava
VALUES(drzava_sekv.NEXTVAL,'Madjarska',17);
```

--statistika,neke vrste opklada mozda nemaju timove ili igrace ili ostalo, trenutno cemo postaviti sve na null, pa mozemo naknadno preko UPDATE da unesemo rezultate meceva i sl

```
INSERT INTO statistika
VALUES (statistika_sekv.NEXTVAL,null,null,null,null);
INSERT INTO statistika
VALUES (statistika_sekv.NEXTVAL,null,null,null,null);
INSERT INTO statistika
VALUES (statistika_sekv.NEXTVAL,null,null,null,null);
INSERT INTO statistika
VALUES (statistika_sekv.NEXTVAL,null,null,null,null);
INSERT INTO statistika
VALUES (statistika_sekv.NEXTVAL,null,null,null,null);
INSERT INTO statistika
VALUES (statistika_sekv.NEXTVAL,null,null,null,null);
INSERT INTO statistika
VALUES (statistika_sekv.NEXTVAL,null,null,null,null);
INSERT INTO statistika
VALUES (statistika_sekv.NEXTVAL,null,null,null,null);
INSERT INTO statistika
VALUES (statistika_sekv.NEXTVAL,null,null,null,null);
INSERT INTO statistika
```

VALUES (statistika_sekv.NEXTVAL,null,null,null,null);

--informacije dodatne o vrsti opklade, unosimo broj listica odigranih ako ih ima za datu vrstu opklade, pogodnosti ako ima, i vrstu

INSERT INTO opklada_info

VALUES (opklada_info_sekv.NEXTVAL,null,null,1);

INSERT INTO opklada_info

VALUES (opklada_info_sekv.NEXTVAL,null,' odbijanje poreza ’,2);

INSERT INTO opklada_info

VALUES (opklada_info_sekv.NEXTVAL,null,' odbijanje poreza ’,3);

INSERT INTO opklada_info

VALUES (opklada_info_sekv.NEXTVAL,null,null,4);

INSERT INTO opklada_info

VALUES (opklada_info_sekv.NEXTVAL,15,null,5);

INSERT INTO opklada_info

VALUES (opklada_info_sekv.NEXTVAL,300,null,6);

INSERT INTO opklada_info

VALUES (opklada_info_sekv.NEXTVAL,100,' povećana stopa dobitka’,7);

INSERT INTO opklada_info

VALUES (opklada_info_sekv.NEXTVAL,69,null,8);

INSERT INTO opklada_info

VALUES (opklada_info_sekv.NEXTVAL,91,null,9);

INSERT INTO opklada_info

VALUES (opklada_info_sekv.NEXTVAL,15, null,10);

--vrsta opklade, unosimo naziv opklade i na koliko subjekata se moze kladiti i broj stranog kljuca za datu vrstu

INSERT INTO vrsta

VALUES(vrsta_sekv.NEXTVAL,'fudbal',2,1);

INSERT INTO vrsta

VALUES(vrsta_sekv.NEXTVAL,'fudbal',1,2);

```
INSERT INTO vrsta
VALUES(vrsta_sekv.NEXTVAL,'kosarka',2,3);

INSERT INTO vrsta
VALUES(vrsta_sekv.NEXTVAL,'kosarka',1,4);
INSERT INTO vrsta
VALUES(vrsta_sekv.NEXTVAL,'skijanje',1,5);
INSERT INTO vrsta
VALUES(vrsta_sekv.NEXTVAL,'trke pasa',1,6);
INSERT INTO vrsta
VALUES(vrsta_sekv.NEXTVAL,'formula 1',1,7);
INSERT INTO vrsta
VALUES(vrsta_sekv.NEXTVAL,'trke kornjaca',1,8);
INSERT INTO vrsta
VALUES(vrsta_sekv.NEXTVAL,'trke zeceva',1,9);
INSERT INTO vrsta
VALUES(vrsta_sekv.NEXTVAL,'trke nojeva',1,10);
--listici, unosimo ono sto je klijent odigrao, ishode, parove i koja je vrsta
INSERT INTO listici
VALUES(listici_sekv.NEXTVAL,'drugi pas po redu prvi',null,6);
INSERT INTO listici
VALUES(listici_sekv.NEXTVAL,'prva kornjaca posljednja',null,8);
INSERT INTO listici
VALUES(listici_sekv.NEXTVAL,'formula pod rednim brojem 6 prva',null,7);
INSERT INTO listici
VALUES(listici_sekv.NEXTVAL,'skijas 8 prvi',null,5);
INSERT INTO listici
VALUES(listici_sekv.NEXTVAL,'zec 1 prvi',null,9);
INSERT INTO listici
```

VALUES(listici_sekv.NEXTVAL,'noj 6 posljednji',null,10);

--klijenti, unosimo podatke o klijentu, ime, racun i njegov listic

INSERT INTO klijenti

VALUES (klijenti_sekv.NEXTVAL,'Muji Mujic',150,5);

INSERT INTO klijenti

VALUES (klijenti_sekv.NEXTVAL,'Fari Faric',100,6);

INSERT INTO klijenti

VALUES (klijenti_sekv.NEXTVAL,'Timi Timic',50,7);

INSERT INTO klijenti

VALUES (klijenti_sekv.NEXTVAL,'Keno Kenic',80,8);

INSERT INTO klijenti

VALUES (klijenti_sekv.NEXTVAL,'Kan Kanic',99,9);

INSERT INTO klijenti

VALUES (klijenti_sekv.NEXTVAL,'Dino Dinic',500,10);

--poslovnica, sve sto smo prije unijeli ce biti sadrzano u poslovnici, od zaposlenika do klijenata, stanja racuna poslovnice i u kojoj se drzavi nalazi

INSERT INTO poslovnica

VALUES(poslovnica_sekv.NEXTVAL,1,1,1,15000,1);

INSERT INTO poslovnica

VALUES(poslovnica_sekv.NEXTVAL,2,2,1,15000,1);

INSERT INTO poslovnica

VALUES(poslovnica_sekv.NEXTVAL,3,3,2,10000,2);

INSERT INTO poslovnica

VALUES(poslovnica_sekv.NEXTVAL,4,4,10,150000,3);

INSERT INTO poslovnica

VALUES(poslovnica_sekv.NEXTVAL,5,5,5,1000,1);

INSERT INTO poslovnica

```
VALUES(poslovnica_sekv.NEXTVAL,6,6,5,1000,1);
INSERT INTO poslovnica
VALUES(poslovnica_sekv.NEXTVAL,7,7,5,1000,1);

INSERT INTO poslovnica
VALUES(poslovnica_sekv.NEXTVAL,8,8,6,1900,2);
INSERT INTO poslovnica
VALUES(poslovnica_sekv.NEXTVAL,9,9,6,1900,2);
INSERT INTO poslovnica
VALUES(poslovnica_sekv.NEXTVAL,10,10,6,1900,2);
```

d) 30 SQL upita, a od toga :

10 jednostavnijih upita po volji

```
--1, ispisujemo sve sto se nalazi u poslovnici
SELECT *
FROM poslovnica;
--2,svi zaposlenici u poslovnici
SELECT z.naziv
FROM poslovnica p, zaposlenici z
WHERE p.zaposlenici_id=z.zaposlenici_id;
--3,naziv drzava i porez gdje se nalazi  data poslovnica
SELECT d.naziv, d.porez
FROM poslovnica p, drzava d
WHERE p.drzava_id=d.drzava_id AND d.porez<18;
--4,naziv opklade koju je odigrao klijent
SELECT v.naziv
FROM klijent k, listici l, vrsta v
WHERE k.listic_id=l.listic_id AND l.vrsta_id=v.vrsta_id;
```

--5 sve pogodnosti na osnovu listica koji je odigrao klijent

```
SELECT o.pogodnosti

FROM klijent k, listici l, vrsta v, opklada_info o

WHERE k.listic_id=l.listic_id AND l.vrsta_id=v.vrsta_id AND v.vrsta_info_id=o.vrsta_info_id ;
```

--6, ako nema pogodnosti umjesto null napisati nema pogodnosti, u suprotnom ostaje isto

```
SELECT Decode(Nvl(pogodnosti,0),0,'nema pogodnosti',pogodnosti)

FROM opklada_info;
```

--7, izracunati broj zaposlenika I broj klijenata po poslovnici

```
SELECT Count(zaposlenik_id), Count(klijent_id)

FROM poslovnica;
```

--8, sve vrste poslovnica u Srbiji

```
SELECT v.naziv

FROM poslovnica p, drzava d,vrsta_poslovnice v

WHERE p.drzava_id=d.drzava_id AND p.vrsta_poslovnice_id=v.vrsta_poslovnice_id  AND
d.drzava='Srbija';
```

--9, stanje novca u svim obicnim poslovnicama

```
SELECT p.stanje_novca

FROM poslovnica p, vrsta_poslovnice v

WHERE p.vrsta_poslovnice_id=v.vrsta_poslovnice_id  AND v.vrsta_poslovnice='obicna';
```

--10, za klijenta ciji je id 1, ispisati naziv I njegov racun

```
SELECT naziv,racun

FROM klijent

WHERE klijent_id=1;
```

▪ 5 upita sa grupnim funkcijama (minimalno 2 sa HAVING klauzulom)

--1,po vrsti poslovnice izracunati broj zaposlenika

```
SELECT v.naziv, Count(z.zaposlenik_id)
FROM poslovnica p, zaposlenici z,vrsta_poslovnice v
WHERE p.zaposlenik_id=z.zaposlenik_id  AND p.vrsta_poslovnice=v.vrsta_poslovnice
GROUP BY v.naziv;
```

--2, po vrsti poslovnice izracunati broj klijenata

```
SELECT v.naziv, Count(k.klijent_id)
FROM poslovnica p, klijenti k, vrsta_poslovnice v
WHERE p.klijent_id=k.klijent_id  AND p.vrsta_poslovnice=v.vrsta_poslovnice
GROUP BY v.naziv;
```

--3, po vrsti poslovnice izracunati sumu svih racuna klijenata

```
SELECT v.naziv,sum(k.racun)
FROM poslovnica p, klijenti k, vrsta_poslovnice v
WHERE p.klijent_id=k.klijent_id AND  p.vrsta_poslovnice=v.vrsta_poslovnice
GROUP BY v.naziv;
```

--4, po poslovnicama izracunati sumu racuna klijenata I prosjek svih racuna, gdje suma racuna treba biti veca od 1000

```
SELECT p.poslovnica_id, sum(k.racun), avg(k.racun)
FROM poslovnica p, klijenti k
WHERE p.klijent_id=k.klijent_id
GROUP BY p.poslovnica_id
HAVING sum(k.racun)>1000;
```

--5, po poslovnicama izracunati broj zaposlenika gdje broj zaposlenika treba biti veci od 5

```
SELECT p.poslovnica_id, Count(z.zaposlenik_id)
FROM zaposlenici z,poslovnice p
WHERE p.zaposlenik_id=z.zaposlenik_id
GROUP BY p.poslovnica_id
HAVING Count(z.zaposlenik_id)>5;
```

▪ 5 upita sa korištenjem podupita

--1, naziv klijenta ciji je racun veci od svih ostalih klijenata

```
SELECT naziv
FROM klijent
WHERE racun > ALL(SELECT k.racun
                  FROM klijent k);
```

--2,naziv poslovnice u Bosni cije je stanje novca manje od svih ostalih u Bosni

```
SELECT p.naziv
FROM poslovnica p, drzava d
WHERE p.drzava_id=d.drzava_id AND d.naziv='Bosna i Hercegovina'
AND p.stanje_racuna< ALL( SELECT p1.stanje_racuna
                          FROM poslovnica p1, drzava d1
                          WHERE p1.drzava_id=d1.drzava_id AND d1.naziv='Bosna i Hercegovina');
```

--3,naziv odjel i funcija zaposlenika u poslovnicama Njemacke

```
SELECT z.naziv, z.odjel, z.funkcija
FROM poslovnica p, zaposlenici z
WHERE p.zaposlenik_id=z.zaposlenik_id  AND p.poslovnica_id=ANY(SELECT p1.poslovnica_id
                                                               FROM poslovnica p1, drzava d1
                                                               WHERE p1.drzava_id=d1.drzava_id AND
                                                                     d.naziv='Njemacka');
```

--4,naziv poslovnica u Sloveniji

```
SELECT v.naziv
FROM poslovnica p, vrsta_poslovnice v
WHERE p.vrsta_poslovnice_id=v.vrsta_poslovnice_id AND p.poslovnica_id=ANY(
                                                                  SELECT p1.poslovnica_id
                                                                  FROM poslovnica p1, drzava d1
                                                                  WHERE p1.drzava_id=d1.drzava_id AND
                                                                        d1.naziv='Slovenija');
```

--5, klijentov racun i naziv ciji je racun veci od svih ostalih u poslovnicama 1 2 3

```
SELECT k.racun,k.naziv
FROM klijent k
WHERE  k.racun>ALL(SELECT k1.racun
                   FROM poslovnica p, klijent k1
                   WHERE p.klijent_id=k1.klijent_id AND p.poslovnica_id IN (1,2,3));
```

▪ 5 upita sa više od jednog nivoa podupita

--1, naziv klijenta ciji je racun veci od svih ostalih u poslovnicama koje se nalaze u Austriji

```
SELECT naziv
FROM klijent
WHERE racun > ALL(SELECT k.racun
                  FROM klijent k, poslovnica p
                  WHERE p.klijent_id=k.klijent_id AND p.poslovnica_id=ANY(SELECT p.poslovnica_id
                                                                          FROM poslovnica p, drzava d
                                                                          WHERE
                                                                          p.drzava_id=d.drzava_id AND d.naziv='Austrija'));
```

--2, naziv klijenta u poslovnicama u kojima imena klijenata pocinju sa A ili imaju na nekom drugom mjestu slovo a I te se poslovnice nalaze u Hrvatskoj

```
SELECT k.naziv
FROM klijent k, poslovnica p
WHERE p.klijent_id = k.klijent_id AND p.poslovnica_id = ANY(SELECT p1.poslovnica_id
                                                           FROM poslovnica p1,klijent k1
                                                           WHERE p1.klijent_id=k1.klijent_id AND (k1.naziv
                                                           LIKE ‘%a’ OR k1.naziv LIKE ‘A%’) AND p1.poslovnica_id=ANY(SELECT p2.poslovnica_id
                                                           FROM poslovnica p2, drzava d
                                                           WHERE p2.drzava_id=d.drzava_id AND
                                                           d.naziv=’Hrvatska’));
```

--3, ispisati naziv odjel i funkciju zaposlenika gdje radi Mujo Mujic, ako je stanje novca poslovnice u kojoj on radi vece od svih ostalih stanja u drugim poslovnicama

```
SELECT z.naziv,z.odjel,z.funkcija

FROM zaposlenici z, poslovnica p

WHERE p.zaposlenik_id=z.zaposlenik_id AND p.poslovnica_id=(SELECT p1.poslovnica_id

                                                    FROM poslovnica p1, zaposlenici z1

                                                WHERE p1.zaposlenik_id=z1.zaposlenik_id
AND z1.naziv='Mujo Mujic' AND p1.stanje_novca> ALL(SELECT p2.stanje_novca

                                                FROM poslovnica p2)) AND z.naziv<>'Mujo
Mujic';
```

--4, ispisati vrstu poslovnice, stanje novca, I naziv drzave poslovnice cije je stanje vece od svih ostalih koje se nalaze u Bosni

```
SELECT v.naziv, p.stanje_novca, d.naziv

FROM poslovnica p, vrsta_poslovnice v, drzava d

WHERE p.vrsta_poslovnice_id=v.vrsta_poslovnice_id AND p.drzava_id=d.drzava_id AND
p.stanje_novca>ALL(SELECT p1.stanje_novca

                    FROM poslovnica p1

                    WHERE p1.poslovnica_id=ANY(SELECT p2.poslovnica_id

                                                FROM poslovnica p2,drzava d1

                                            WHERE p2.drzava_id=d1.drzava_id AND
                                                d1.naziv='Bosna I Hercegovina'));
```

--5, naziv klijenata poslovnice cije je stanje novca vece od svih ostalih koje se nalaze u Austriji i u kojima radi Maja Majic

```
SELECT k.naziv

FROM poslovnica p, klijent k

WHERE p.klijent_id=k.klijent_id AND p.stanje_novca>ALL(SELECT p1.stanje_novca

                                                       FROM poslovnica p1

                                                  WHERE p1.poslovnica_id=ANY(SELECT p2.poslovnica_id

                                                    FROM poslovnica p2,drzava d1,zaposlenici z

                                                        WHERE p2.drzava_id=d1.drzava_id AND
                                                  p2.zaposlenik_id=z.zaposlenik_id AND d1.naziv='Austrija' AND

                                                  z.zaposlenik_id=(SELECT z1.zaposlenik_id

                                                                   FROM z1.zaposlenici

                                                                   WHERE z1.naziv='Maja Majic')));
```

▪ 2 upita sa subtotalima (ROLLUP, CUBE, GROUPING SETS)

--1 , ispisati broj klijenata po zaposleniku I drzavi I po drzavi I poslovnici,gdje se prvo ispisuje po drzavi I po poslovnici, zatim po zaposleniku i drzavi

```
SELECT zaposlenik_id, vrsta_poslovnice_id,drzava_id, Count(klijent_id)

FROM poslovnica

GROUP BY GROUPING SETS

((zaposlenik_id,drzava_id),(drzava_id,vrsta_poslovnice_id));
```

--2, izracunaj stanje novca po drzavi I vrsti poslovnice, takodjer izracunati I ukupno stanje po vrsti poslovnice I drzave, I na kraju izracunati ukupno stanje za sve drzave I vrste, a upravo to cemo dobiti preko ROLLUP, da smo koristili CUBE isao bi po manje bitnim kolonama I to tako da bi prvo izracunao ukupno stanje za sve drzave I vrste poslovnica, a zaim bi izracunao po drzavama I na kraju bi racunao jedan red po vrsti poslovnice, drugi red po vrsti I drzavi itd..

```
SELECT vrsta_poslovnice_id, drzava_id, sum(stanje_novca)

FROM poslovnica

GROUP BY ROLLUP(vrsta_poslovnice_id,drzava_id);
```

▪ 2 upita sa TOP N analizom

--1, 5 poslovnica sa najvecim brojem stanja novca, vrsta I u kojoj drzavi

```
SELECT ROWNUM as RANK, poslovnica,drzava(
SELECT v.naziv poslovnica, d.naziv drzava
FROM poslovnica p, vrsta_poslovnice v, drzava d
WHERE p.vrsta_poslovnice_id= v. vrsta_poslovnice_id AND p.drzava_id=d.drzava_id
ORDER BY p.stanje_novca DESC)
WHERE ROWNUM<=5;
```

--2,vrsta poslovnice I broj zaposlenih za 3 poslovnice koje imaju najveci broj zaposlenika

```
SELECT ROWNUM as RANK, poslovnica, broj_zaposlenih(
SELECT v.naziv poslovnica, Count(p.zaposlenik_id)
FROM poslovnica p, vrsta_poslovnice v, zaposlenici z
WHERE p.vrsta_poslovnice_id= v. vrsta_poslovnice_id AND p.zaposlenik_id=z.zaposlenik_id
ORDER BY Count(p.zaposlenik_id) DESC)
WHERE ROWNUM<=3;
```

▪ 1 upit sa korištenjem UNION, SET

--1, jedinstvena lista klijenata, ako zelimo da saznamo koliko ima ukupno klijenata u svim poslovnicama

```
SELECT klijent_id, To_Char(null) naziv, To_Number(null) racun
FROM poslovnica
UNION
SELECT klijent_id,naziv, racun
FROM klijenti;
```

e) 5 indeksa uz detaljno objašnjenje kako ti indeksi doprinose bržem izvršenju nekih od napisanih upita pod d)

--1, zbog ucestalog koristenja poslovnica_id u WHERE klauzi, ubrzava proces vraćanja slogova iz baze podataka korištenjem pointera, po primarnom kljucu poslovnice

```
CREATE INDEX posl_posl_id_idx ON poslovnica(poslovnica_id);
```

--2, zbog ucestalog koristenja drzava_id u WHERE klauzi, ubrzava proces vraćanja slogova iz baze podataka korištenjem pointera, po stranom kljucu drzava_id

```
CREATE INDEX posl_drz_id_idx ON poslovnica(drzava_id);
```

--3, zbog ucestalog koristenja naziv drzave u WHERE klauzi, ubrzava proces vraćanja slogova iz baze podataka korištenjem pointera, po nazivu drzave iz tabele drzava

```
CREATE INDEX drz_drz_naz_idx ON drzava(naz);
```

--4,ovi podaci mogu biti obimni i cesto koristeni u WHERE kaluzi, pa da se brze moze pretraziti stavit cemo index koji ubrzava proces

```
CREATE INDEX posl_kli_id_idx ON poslovnica(klijent_id);
```

--5,ovi podaci mogu biti obimni i cesto koristeni u WHERE kaluzi, pa da se brze moze pretraziti stavit cemo index koji ubrzava proces

```
CREATE INDEX posl_zap_id_idx ON poslovnica(zaposlenik_id);
```

f) 10 funkcija

--1,funkcija prima kao parameter broj koji predstavlja id poslovnice, vraca broj zaposlenika

```
CREATE OR REPLACE FUNCTION brojZaposlenika
(v_id IN poslovnica.poslovnica_id%TYPE)
RETURN NUMBER
IS
brojzap NUMBER(10):=0;
BEGIN
SELECT Count(zaposlenici_id) INTO brojzap
FROM poslovnica
WHERE poslovnica_id=v_id;
```

```
RETURN (brojzap);
END brojZaposlenika;
/
--2,sada fja vraca broj klijenata date poslovnice
CREATE OR REPLACE FUNCTION brojKlijenata
(v_id IN poslovnica.poslovnica_id%TYPE)
RETURN NUMBER
IS
brojkli  NUMBER(10):=0;
BEGIN
SELECT Count(klijenti_id)  INTO brojkli
FROM poslovnica
WHERE poslovnica_id=v_id;
RETURN (brojkli);
END brojKlijenata;
/
--3,stanje novca svih  poslovnica po vrsti
CREATE OR REPLACE FUNCTION stanjeNovcaVrsta
(v_naziv IN vrsta_poslovnice.naziv%TYPE)
RETURN NUMBER
IS
novac NUMBER(10):=0;
BEGIN
SELECT Sum(p.stanjeNovca) INTO novac
FROM poslovnica p, vrsta_poslovnice v
WHERE p.vrsta_poslovnice_id=v.vrsta_poslovnice_id AND v.naziv=v_naziv;
RETURN (novac);
END stanjeNovcaVrsta;
/
```

--4, stanje novca poslovnica po drzavi

```
CREATE OR REPLACE FUNCTION stanjeNovcaDrzava
(v_naziv IN vrsta_poslovnice.naziv%TYPE)
RETURN NUMBER
IS
novac NUMBER(10):=0;
BEGIN
SELECT Sum(p.stanjeNovca) INTO novac
FROM poslovnica p, drzava d
WHERE p.drzava_id=d.drzava_id AND d.naziv=v_naziv;
RETURN (novac);
END stanjeNovcaDrzava;
/
```

--5,stanje novca poslovnica po drzavi i po vrsti poslovnice

```
CREATE OR REPLACE FUNCTION stanjeNovcaVrstaDrzava
(v_naziv IN vrsta_poslovnice.naziv%TYPE, v_naz IN drzava.naziv%TYPE)
RETURN NUMBER
IS
novac NUMBER(10):=0;
BEGIN
SELECT Sum(p.stanjeNovca) INTO novac
FROM poslovnica p, vrsta_poslovnice v,drzava d
WHERE p.vrsta_poslovnice_id=v.vrsta_poslovnice_id AND p.drzava_id=d.drzava_id AND v.naziv=v_naziv
AND d.naziv=v_naz;
RETURN (novac);
END stanjeNovcaVrstaDrzava;
```

```
/
```

--6,suma racuna svih klijenata po poslovnicama

```
CREATE OR REPLACE FUNCTION racunKlijenata
(v_id IN poslovnica.poslovnica_id%TYPE)
RETURN NUMBER
IS
racunn NUMBER(10):=0;
BEGIN
SELECT Sum(k.racun)
INTO racunn
FROM poslovnica p , klijenti k
WHERE p.klijent_id=k.klijent_id AND p.poslovnica_id=v_id;
RETURN(racunn);
END racunKlijenata;
/
```

--7, suma racuna svih klijenata po vrsti poslovnica

```
CREATE OR REPLACE FUNCTION racunKlijenataVrsta
(v_naziv IN vrsta_poslovnice.naziv%TYPE)
RETURN NUMBER
IS
racunn NUMBER(10):=0;
BEGIN
SELECT Sum(k.racun)
INTO racunn
FROM poslovnica p , klijenti k,vrsta_poslovnice v
WHERE p.klijent_id=k.klijent_id AND p. vrsta_poslovnice _id=v. vrsta_poslovnice _id AND
v.naziv=v_naziv;
```

```
RETURN(racunn);
END racunKlijenataVrsta;
/
```

--8, suma racuna svih klijenata po vrsti poslovnica i drzavi

```
CREATE OR REPLACE FUNCTION racunKlijenataVrstaDrzava
(v_naziv IN vrsta_poslovnice.naziv%TYPE, v_naz IN drzava.naziv%TYPE)
RETURN NUMBER
IS
racunn NUMBER(10):=0;
BEGIN
SELECT Sum(k.racun)
INTO racunn
FROM poslovnica p , klijenti k,vrsta_poslovnice v,drzava d
WHERE p.klijent_id=k.klijent_id AND p. vrsta_poslovnice _id=v. vrsta_poslovnice _id AND
p.drzava_id=d.drzava_id AND v.naziv=v_naziv AND d.naziv=v_naz;
RETURN(racunn);
END racunKlijenataVrstaDrzava;
/
```

--9, prosjecni broj listica po vrsti poslovnice

```
CREATE OR REPLACE FUNCTION prosjekListica
(v_naziv IN vrsta_poslovnice.naziv%TYPE)
IS
broj NUMBER(10,3) :=0
BEGIN
SELECT DISTINCT avg(o.broj_listica)
INTO broj
FROM poslovnica p, vrsta_poslovnice v, klijent k, listici l ,vrsta v, opklada_info o
```

```
WHERE p.klijent_id=k.klijent_id AND p. vrsta_poslovnice _id=v. vrsta_poslovnice _id AND
k.listic_id=l.listic_id AND l.vrsta_id=v.vrsta_id AND v.opklada_info_id=o.opklada_info_id AND
v.naziv=v_naziv;

RETURN(broj);

END prosjekListica;

/
```

--10, prosjecni broj listica po id poslovnice

```
CREATE OR REPLACE FUNCTION prosjekListica

(v_id IN poslovnica.poslovnica_id%TYPE)

IS

broj NUMBER(10,3) :=0

BEGIN

SELECT DISTINCT avg(o.broj_listica)

INTO broj

FROM poslovnica p, klijent k, listici l ,vrsta v, opklada_info o

WHERE p.klijent_id=k.klijent_id AND  k.listic_id=l.listic_id AND l.vrsta_id=v.vrsta_id AND
v.opklada_info_id=o.opklada_info_id AND p.poslovnica_id=v_id;

RETURN(broj);

END prosjekListica;

/
```

g) 10 procedura

napravit cemo procedure koje su najvise vezane za azuriranje podataka, jer je to najpotrebnije s obzirom da se radi o kladionici, za svaku cemo osigurati putem 'bacanja izuzetaka' da se ne desi neka greska

--1, unos klijenta u tabelu klijenata

```
CREATE OR REPLACE PROCEDURE proceduraKlijent

(k_id IN klijent.klijent_id%TYPE, k_naz IN klijent.naziv%TYPE, k_rac IN klijent.racun%TYPE, k_listic_id IN
klijent.listic_id%TYPE)

IS

BEGIN

INSERT INTO klijent
```

```
VALUES (k_id,k_naz,k_rac,k_listic_id);
 EXCEPTION WHEN DUP_VAL_ON_INDEX
THEN ROLLBACK;
Raise_application_error (-20585, 'Id vec postoji');
WHEN VALUE_ERROR
THEN ROLLBACK;
Raise_application_error (-20585, 'Vrijednost nekog od podataka pogresna');
COMMIT;
END proceduraKlijent;
/
```

--2, unos drzave u kojoj su otvorene poslovnice

```
CREATE OR REPLACE PROCEDURE proceduraDrzava
(d_id IN drzava.drzava_id%TYPE, d_naz IN drzava.naziv%TYPE, d_porez IN drzava.porez%TYPE)
IS
BEGIN
INSERT INTO drzava
VALUES (d_id,d_naz,d_porez);
EXCEPTION WHEN DUP_VAL_ON_INDEX
THEN ROLLBACK;
Raise_application_error (-20585, 'Id vec postoji');
WHEN VALUE_ERROR
THEN ROLLBACK;
Raise_application_error (-20585, 'Vrijednost nekog od podataka pogresna');
COMMIT;
END proceduraDrzava;
/
```

--3, unos novih poslovnica

```
CREATE OR REPLACE PROCEDURE proceduraPoslovnica
```

```
(p_zap IN poslovnica.zaposlenik_id%TYPE ,p_kli IN poslovnica.klijent_id%TYPE,p_drz IN
poslovnica.drzava_id%TYPE,p_stanje IN poslovnica.stanje_novca%TYPE,p_vrsta IN
poslovnica.vrsta_poslovnice%TYPE)

IS

BEGIN

INSERT INTO poslovnica

VALUES (p_zap,p_kli,p_drz,p_stanje,p_vrsta);

EXCEPTION WHEN DUP_VAL_ON_INDEX

THEN ROLLBACK;

Raise_application_error (-20585, 'Id vec postoji');

WHEN VALUE_ERROR

THEN ROLLBACK;

Raise_application_error (-20585, 'Vrijednost nekog od podataka pogresna');

COMMIT;

END proceduraPoslovnica;

/
```

--4, unos novih vrsta opklada

```
CREATE OR REPLACE PROCEDURE proceduraVrstaOpklada

(v_id IN vrsta.vrsta_id%TYPE,v_naz IN vrsta.naziv%TYPE, v_br IN vrsta.broj_subjekata%TYPE,v_op_id IN
vrsta.opklada_info_id%TYPE)

IS

BEGIN

INSERT INTO vrsta

VALUES (v_id,v_naz,v_br,v_op_id);

EXCEPTION WHEN DUP_VAL_ON_INDEX

THEN ROLLBACK;

Raise_application_error (-20585, 'Id vec postoji');

WHEN VALUE_ERROR

THEN ROLLBACK;

Raise_application_error (-20585, 'Vrijednost nekog od podataka pogresna');
```

```
COMMIT;

END proceduraVrstaOpklada;

/
```

--5, azuriranje statistike

```
CREATE OR REPLACE PROCEDURE proceduraStatistike

(s_id IN statistika.statistika_id%TYPE ,s_igr IN statistika.igraci%TYPE,s_tim IN statistika.timovi%TYPE,s_lig IN  statistika.liga%TYPE,  s_mec IN statistika.mec%TYPE )

IS

BEGIN

UPDATE statistika

SET (statistika_id, igraci, timovi, liga, mec)=(s_id,s_igr,s_tim,s_lig,s_mec);

EXCEPTION WHEN DUP_VAL_ON_INDEX

THEN ROLLBACK;

Raise_application_error (-20585, 'Id vec postoji');

WHEN VALUE_ERROR

THEN ROLLBACK;

Raise_application_error (-20585, 'Vrijednost nekog od podataka pogresna');

COMMIT;

END proceduraStatistike;

/
```

--6, azuriranje zaposlenih

```
CREATE OR REPLACE PROCEDURE proceduraZaposlenih

(z_id IN zaposlenici.zaposlenik_id%TYPE ,z_odj IN zaposlenici.odjel%TYPE,z_fja IN zaposlenici.funkcija%TYPE,z_naz IN zaposlenici.naziv%TYPE  )

IS

BEGIN

UPDATE zaposlenici

SET (zaposlenik_id, odjel,funkcija,naziv)=(z_id,z_odj,z_fja,z.naz);
```

```
EXCEPTION WHEN DUP_VAL_ON_INDEX

THEN ROLLBACK;

Raise_application_error (-20585, 'Id vec postoji');

WHEN VALUE_ERROR

THEN ROLLBACK;

Raise_application_error (-20585, 'Vrijednost nekog od podataka pogresna');

COMMIT;

END proceduraZaposlenih;

/
```

--7,azuriranje novih informacija o opkladi

```
CREATE OR REPLACE PROCEDURE proceduraInfo

(o_id IN opklada_info.opklada_info_id%TYPE ,o_br_li IN
opklada_info.broj_uplacenih_listica%TYPE,o_pog IN opklada_info.pogodnosti%TYPE, o_stat IN
opklada_info.statistika_id%TYPE)

IS

BEGIN

UPDATE opklada_info

SET (opklada_info_id,broj_uplacenih_listica,pogodnosti,statistika_id)=(o_id,o_br_li,o_pog,o_stat);

EXCEPTION WHEN DUP_VAL_ON_INDEX

THEN ROLLBACK;

Raise_application_error (-20585, 'Id vec postoji');

WHEN VALUE_ERROR

THEN ROLLBACK;

Raise_application_error (-20585, 'Vrijednost nekog od podataka pogresna');

COMMIT;

END proceduraInfo;

/
```

--8, unos odigranog listica datog klijenta

```
CREATE OR REPLACE PROCEDURE proceduraListica
```

```
(l_id IN listici.listici_id%TYPE, l_ish IN listici.ishod%TYPE, l_par IN listici.parovi%TYPE, l_vrsta IN
listici.vrsta_id%TYPE)

IS

BEGIN

INSERT INTO listici

VALUES (l_id,l_ish,l_par,l_vrsta);

EXCEPTION WHEN DUP_VAL_ON_INDEX

THEN ROLLBACK;

Raise_application_error (-20585, 'Id vec postoji');

WHEN VALUE_ERROR

THEN ROLLBACK;

Raise_application_error (-20585, 'Vrijednost nekog od podataka pogresna');

COMMIT;

END proceduraListica;

/
```

--9, unos novih zaposlenika

```
CREATE OR REPLACE PROCEDURE proceduraZaposlenihUnos

(z_id IN zaposlenici.zaposlenik_id%TYPE ,z_odj IN zaposlenici.odjel%TYPE,z_fja IN
zaposlenici.funkcija%TYPE,z_naz IN zaposlenici.naziv%TYPE  )

IS

BEGIN

INSERT INTO zaposlenici

VALUES (z_id,z_odj,z_fja,z.naz);

EXCEPTION WHEN DUP_VAL_ON_INDEX

THEN ROLLBACK;

Raise_application_error (-20585, 'Id vec postoji');

WHEN VALUE_ERROR

THEN ROLLBACK;

Raise_application_error (-20585, 'Vrijednost nekog od podataka pogresna');
```

```
COMMIT;

END proceduraZaposlenihUnos;

/
```

--10, unos nov statistike

```
CREATE OR REPLACE PROCEDURE proceduraStatistikeUnos

(s_id IN statistika.statistika_id%TYPE ,s_igr IN statistika.igraci%TYPE,s_tim IN statistika.timovi%TYPE,s_lig
IN  statistika.liga%TYPE,  s_mec IN statistika.mec%TYPE )

IS

BEGIN

INSERT INTO statistika

VALUES (s_id,s_igr,s_tim,s_lig,s_mec);

EXCEPTION WHEN DUP_VAL_ON_INDEX

THEN ROLLBACK;

Raise_application_error (-20585, 'Id vec postoji');

WHEN VALUE_ERROR

THEN ROLLBACK;

Raise_application_error (-20585, 'Vrijednost nekog od podataka pogresna');

COMMIT;

END proceduraStatistikeUnos;

/
```

--11, napisat cemo jos proceduru za brisanje zaposlenih, jer je i to precesto potrebno

```
CREATE PROCEDURE izbrisiZaposlenog

(z_id IN zaposlenici.zaposlenik_id%TYPE)

BEGIN

DELETE FROM zaposlenici

WHERE zaposlenici.zaposlenik_id=z_id;

EXCEPTION WHEN DUP_VAL_ON_INDEX
```

```
THEN ROLLBACK;
Raise_application_error (-20585, 'Id vec postoji');
COMMIT;
END izbrisiZaposlenog;
/
```

--12, takodjer ako se neka poslovnica zatvori u nekoj drzavi treba je izbrisati iz tabele

```
CREATE PROCEDURE izbrisiPoslovnicu
(p_id IN poslovnica.poslovnica_id%TYPE)
BEGIN
DELETE FROM poslovnica
WHERE poslovnica_id=p_id;
EXCEPTION WHEN DUP_VAL_ON_INDEX
THEN ROLLBACK;
Raise_application_error (-20585, 'Id vec postoji');
COMMIT;
END izbrisiPoslovnicu;
/
```

h) 10 triggera

--1,u poslovnici gdje je stari dobio otkaz novi zaposlenik dolazi

```
CREATE OR REPLACE TRIGGER novi_zaposlenik
AFTER UPDATE OF zaposlenik_id ON zaposlenici
FOR EACH ROW
BEGIN
UPDATE poslovnica
SET poslovnica.zaposlenik_id=: new.zaposlenik_id
WHERE poslovnica.zaposlenik_id=:old.zaposlenik_id;
END novi_zaposlenik;
/
```

--2,otkazi I novi radnici se mogu azurirati samo radnim danima i to ne poslije 6 sati

```
CREATE OD REPLACE TRIGGER osiguraj_unos_zap

BEFORE INSERT OR DELETE OR UPDATE zaposlenik_id ON poslovnica

FOR EACH ROW

BEGIN

IF (To_Char(SYSDATE,'DY') IN ('SAT', 'SUN')) OR (To_Char(SYSDATE, 'HH24) NOT BETWEEN '08' AND '18')

THEN

Raise_Application_Error(-20500,'Azuriranje zaposlenika moze se desavati samo radnim danima tokom radnog vremena');

END IF;

END osiguraj_unos_zap;

/
```

--3,citava tabela poslovnica se moze azurirati samo u toku radnog vremena

```
CREATE OD REPLACE TRIGGER osiguraj_poslovnicu

BEFORE INSERT OR DELETE OR UPDATE zaposlenik_id ON poslovnica

FOR EACH ROW

BEGIN

IF (To_Char(SYSDATE, 'HH24) NOT BETWEEN '08' AND '22')

THEN

Raise_Application_Error(-20500,'Kraj je radnog vremena, ne mozete unijeti nikakve podatke');

END IF;

END osiguraj_poslovnicu;

/
```

--4,unos u listic se ne moze mijenjati tj mora se zabraniti updating

```
CREATE OR REPLACE TRIGGER osiguraj_listic

BEFORE UPDATING listici

BEGIN

Raise_Application_Error(-20500, 'Podaci sa listica se ne mogu mijenjati');

END osiguraj_listic;

/
```

--5,ako u poslovnici nema vise novca, klijenti vise ne mogu igrati I treba zabraniti ikakav unos

```
CREATE OR REPLACE TRIGGER osiguraj_stanje
BEFORE INSERT OR UPDATE OR DELETE klijent_id ON poslovnica
FOR EACH ROW
DECLARE
stanje IN poslovnica.stanje_novca%TYPE;
BEGIN
SELECT stanje_novca
INTO stanje
FROM poslovnica
WHERE klijent_id=:new.klijent_id;
IF stanje =0
THEN
Raise_Application_Error(-20500,'Stanje racuna je 0 KM');
END IF;
END osiguraj_stanje;
/
```

--6,ako je klijentov racun presao 1000, mora prvo sve platiti da bi nastavio igrati

```
CREATE OR REPLACE TRIGGER osiguraj_racun_klijenta
BEFORE UPDATE OR DELETE listic_id ON klijenti
FOR EACH ROW
DECLARE
rac IN klijenti.racun%TYPE
BEGIN
SELECT racun
INTO rac
FROM klijenti
WHERE listic_id=:new.listic_id;
IF rac NOT BETWEEN '0' AND '1000'
```

```
THEN
Raise_Application_Error(-20500,'Klijent prije uplate novog listica, da izmiri racun');
END IF;
END osiguraj_racun_klijenta;
/
```

--7, regulisanje pogodnosti, ako je broj listica neke vrste opklade mali, postoji mogucnost za neku pogodnosti, u suprotnom pogodnost mora biti null

```
CREATE OR REPLACE TRIGGER trigerPogodnosti
BEFORE INSERT OR UPDATE pogodnosti ON opklada_info
FOR EACH ROW
WHEN(new.pogodnosti<> NULL)
DECLARE
br_list  IN opklada_info.broj_uplacenih_listica%TYPE;
BEGIN
SELECT broj_uplacenih_listica
INTO br_list
FROM opklada_info
WHERE pogodnosti=:new.pogodnosti;
IF br_list NOT BETWEEN '1' AND '100'
THEN
Raise_Application_Error(-20500,'Mogucnost za neke pogodnosti nije moguc');
END IF;
END trigerPogodnosti;
/
```

--8,ako se unese zaposlenik sa istim id zaposlenika koji je vec unesen

```
CREATE OR REPLACE TRIGGER osiguraj_id_zaposenika
BEFORE INSERT zaposlenik_id ON zaposlenici
```

```
FOR EACH ROW

WHEN(new.zaposlenik_id=old.zaposlenik_id)

BEGIN

Raise_Application_Error(-20500,'Zaposlenik sa tim id vec postoji!');

END osiguraj_id_zaposlenika;

/
```

--9, ako se u poslovnicu unese zaposlenik koji ne postoji

```
CREATE OR REPLACE TRIGGER osiguraj_unos_zaposlenika_poslovnica

BEFORE INSERT zaposlenik_id ON poslovnica

FOR EACH ROW

DECLARE

z_id IN zaposlenici.zaposlenik_id%TYPE;

BEGIN

SELECT zaposlenik_id

INTO z_id

FROM zaposlenici

WHERE zaposlenik_id=:new.zaposlenik_id;

IF z_id= NULL

THEN

Raise_Application_Error(-20500,'Zaposlenika kojeg pokusavate  unijeti u arhivu poslovnice, ne postoji u arhivi zaposlenih');

END IF;

END osiguraj_unos_zaposlenika_poslovnica;

/
```

i) Napisati jednu smislenu i kompleksnu PL/SQL skriptu koja će automatizirati određenu akciju u poslovanju firme, te istu detaljno opisati

```
CREATE OR REPLACE FUNCTION racunKlijenata
(v_id IN poslovnica.poslovnica_id%TYPE)
RETURN NUMBER
IS
racunn NUMBER(10):=0;
BEGIN
SELECT Sum(k.racun)
INTO racunn
FROM poslovnica p , klijenti k
WHERE p.klijent_id=k.klijent_id AND p.poslovnica_id=v_id;
RETURN(racunn);
END racunKlijenata;
/

CREATE PROCEDURE izbrisiPoslovnicu
(p_id IN poslovnica.poslovnica_id%TYPE)
BEGIN
DELETE FROM poslovnica
WHERE poslovnica_id=p_id;
EXCEPTION WHEN DUP_VAL_ON_INDEX
THEN ROLLBACK;
Raise_application_error (-20585, 'Id vec postoji');
COMMIT;
END izbrisiPoslovnicu;
/
```

```
CREATE OR REPLACE TRIGGER osiguraj_racun_klijenta
BEFORE UPDATE OR DELETE listic_id ON klijenti
FOR EACH ROW
DECLARE
rac IN klijenti.racun%TYPE
BEGIN
SELECT racun
INTO rac
FROM klijenti
WHERE listic_id=:new.listic_id;
IF rac NOT BETWEEN ‘0’ AND ‘1000’
THEN
Raise_Application_Error(-20500,’Klijent prije uplate novog listica, da izmiri racun’);
END IF;
END osiguraj_racun_klijenta;
/
```