Zadatak 36 – Analiza ukupne prodaje po godini, kvartalu i mesecu

Kod:

```
SELECT
  YEAR(datum_prodaje) AS Godina,
  DATEPART(QUARTER, datum_prodaje) AS Kvartal,
  MONTH(datum_prodaje) AS Mesec,
  SUM(iznos) AS UkupnaProdaja
FROM
  Prodaje
GROUP BY
  ROLLUP(YEAR(datum_prodaje),
         DATEPART(QUARTER, datum_prodaje),
         MONTH(datum_prodaje)
  );
```

Zadatak 37 – Grupisanje zaposlenih

Kod:

```
SELECT
  OpsegPlata,
  COUNT(*) AS BrojZaposlenih
FROM
  (SELECT
     ID_zaposlenog,
     Ime,
     Prezime,
     Plata,
     CASE
       WHEN Plata BETWEEN 0 AND 999 THEN '0-999'
       WHEN Plata BETWEEN 1000 AND 1999 THEN '1000-1999'
       WHEN Plata BETWEEN 2000 AND 2999 THEN '2000-2999'
       -- Dodajte više opsega po potrebi
       ELSE '3000+' END AS OpsegPlata
   FROM
     Zaposleni
  ) AS TabelaSaOpsegom
GROUP BY
  OpsegPlata
ORDER BY
  OpsegPlata;
```

Zadatak 38 – Prodaja za svaki mesec unutar godine

Kod;

```
SELECT
  YEAR(datum_prodaje) AS Godina,
  MONTH(datum_prodaje) AS Mesec,
  SUM(iznos_prodaje) AS MesečnaProdaja,
  SUM(SUM(iznos_prodaje)) OVER (
    PARTITION BY YEAR(datum_prodaje)
    ORDER BY MONTH(datum_prodaje)
  ) AS TekućiZbir
FROM
  Prodaje
GROUP BY
  YEAR(datum_prodaje),
  MONTH(datum_prodaje)
ORDER BY
  Godina,
  Mesec;
```

Zadatak 39 – Razlika u iznosu prodaje

Kod:

```
SELECT
  ID_prodaje,
  ID_klijenta,
  datum_prodaje,
  iznos_prodaje,
  YEAR(datum_prodaje) AS Godina,
  iznos_prodaje - LAG(iznos_prodaje) OVER (
    PARTITION BY ID_klijenta, YEAR(datum_prodaje)
    ORDER BY datum_prodaje
  ) AS Delta
FROM
  Prodaje
ORDER BY
  ID_klijenta,
  datum_prodaje;
```

Zadatak 40 – Godišnja razlika u ukupnoj prodaji po klijentima

Kod:

```
SELECT
  ID_klijenta,
  Godina,
  UkupnaProdaja,
  UkupnaProdaja - LAG(UkupnaProdaja) OVER (
    PARTITION BY ID_klijenta
    ORDER BY Godina
  ) AS YearOverYearDelta
FROM
  (SELECT
     ID_klijenta,
     YEAR(datum_prodaje) AS Godina,
     SUM(iznos_prodaje) AS UkupnaProdaja
   FROM
     Prodaje
   GROUP BY
     ID_klijenta,
     YEAR(datum_prodaje)
  ) AS GodisnjeProdaje
ORDER BY
  ID_klijenta,
  Godina;
```

Zadatak 41 – Rekurzivni upit za kreiranje hijerarhijske strukture zaposlenih

Kod:

```
WITH RECURSIVE HijerarhijaZaposlenih AS (
  SELECT
    ID_zaposlenog,
    ime,
    prezime,
    ID_menadzera,
    CAST(ime AS VARCHAR(255)) AS Hijerarhija
  FROM
    Zaposleni
  WHERE
    ID_menadzera IS NULL
  UNION ALL
  SELECT
    z.ID_zaposlenog,
```

```
    z.ime,
    z.prezime,
    z.ID_menadzera,
    h.Hijerarhija || ' -> ' || z.ime AS Hijerarhija
  FROM
    Zaposleni z
    INNER JOIN HijerarhijaZaposlenih h
    ON z.ID_menadzera = h.ID_zaposlenog
)
SELECT
  *
FROM
  HijerarhijaZaposlenih
ORDER BY
  Hijerarhija;
```

Kod:

```
SELECT
  a.ID_zaposlenog AS 'ID Zaposlenog',
  a.ime AS 'Ime Zaposlenog',
  a.prezime AS 'Prezime Zaposlenog',
  b.ID_zaposlenog AS 'ID Menadžera',
  b.ime AS 'Ime Menadžera',
  b.prezime AS 'Prezime Menadžera'
FROM
  Zaposleni a
  LEFT JOIN Zaposleni b
  ON a.ID_menadzera = b.ID_zaposlenog
ORDER BY
  a.ID_menadzera, a.ID_zaposlenog;
```

Zadatak 42 – Prozorske funkcije

Kod:

```
WITH NumerisaniDani AS (
  SELECT
    ID_prodavca,
    datum_prodaje,
    iznos_prodaje,
    ROW_NUMBER() OVER (
      PARTITION BY ID_prodavca
      ORDER BY datum_prodaje
    ) AS RedniBroj
  FROM
    DnevnaProdaja
```

```
  WHERE
    iznos_prodaje > 0
),
Serije AS (
  SELECT
    ID_prodavca,
    datum_prodaje,
    iznos_prodaje,
    DATEDIFF(day, '1900-01-01', datum_prodaje) - RedniBroj
    AS Serija
  FROM
    NumerisaniDani
)
SELECT
  ID_prodavca,
  MIN(datum_prodaje) AS PocetakSerije,
  MAX(datum_prodaje) AS KrajSerije,
  COUNT(*) AS DuzinaSerije
FROM
  Serije
GROUP BY
  ID_prodavca,
  Serija
ORDER BY
  ID_prodavca,
  PocetakSerije;
```

Zadatak 43 – Prilagođavanje i poboljšanje strukture tabele

Kod:

```
ALTER TABLE Zaposleni
ADD Email VARCHAR(255);

Kod:
UPDATE Zaposleni
SET Email = 'email_adresa@primer.com'
WHERE ID_zaposlenog = 1;
-- Pretpostavljajući da je 1 šifra zaposlenog
-- kome ažuriramo imejl adresu
```

Kod:

```
ALTER TABLE Zaposleni
ADD CONSTRAINT PlataVecaOdNule CHECK (Plata > 0);

Kod:
```

```
ALTER TABLE Zaposleni
ADD CONSTRAINT EmailJedinstven UNIQUE (Email);
```

Zadatak 44 – Rešavanje problema sa kolonom

Kod:

```
ALTER TABLE Klijenti
ADD Ime VARCHAR(255),
ADD Prezime VARCHAR(255);
```

Kod:

```
UPDATE Klijenti
SET Ime = LEFT(
   ImePrezime,
   CHARINDEX(' ', ImePrezime + ' ') - 1
   ),
   Prezime = SUBSTRING(
   ImePrezime,
   CHARINDEX(' ', ImePrezime + ' ') + 1,
   LEN(ImePrezime)
   );
```

Kod:

```
ALTER TABLE Klijenti
DROP COLUMN ImePrezime;
```

Zadatak 45 – Rešavanje problema sa viševrednosnim kolonama

Kod:

```
CREATE TABLE TelefoniKontakata (
  ID INT AUTO_INCREMENT PRIMARY KEY,
  ID_Kontakta INT,
  BrojTelefona VARCHAR(255),
  FOREIGN KEY (ID_Kontakta) REFERENCES Kontakti(ID)
);
```

Kod:

```
INSERT INTO TelefoniKontakata (ID_Kontakta, BrojTelefona)
SELECT ID_Kontakta, value
FROM Kontakti
CROSS APPLY STRING_SPLIT(Telefoni, ',');
```

```
Kod:
INSERT INTO TelefoniKontakata (ID_Kontakta, BrojTelefona)
SELECT ID_Kontakta, unnest(string_to_array(Telefoni, ','))
FROM Kontakti;
```

Kod:

```
ALTER TABLE TelefoniKontakata
ADD UNIQUE (BrojTelefona);
```

Zadatak 46 – Osiguranje dobro definisane i stabilne strukture baze podataka

```
Kod:
CREATE TABLE KnjigeAutori (
  ID_Knjige INT,
  ID_Autora INT,
  FOREIGN KEY (ID_Knjige) REFERENCES Knjige(ID_Knjige),
  FOREIGN KEY (ID_Autora) REFERENCES Autori(ID_Autora),
  PRIMARY KEY (ID_Knjige, ID_Autora)
);
```

-

Kod:

```
CREATE TABLE IzdateKnjige (
  ID_Izdavanja INT AUTO_INCREMENT PRIMARY KEY,
  ID_Knjige INT,
  ID_Clana INT,
-- Pretpostavljamo da postoji tabela Clanovi
-- sa ovom šifrom
  DatumIzdavanja DATE,
  DatumVracanja DATE,
  FOREIGN KEY (ID_Knjige) REFERENCES Knjige(ID_Knjige)
);
```

Zadatak 47 – Problem nepotrebnih dupliranih kolona

Kod:

```
ALTER TABLE DetaljiNarudzbina
DROP COLUMN DatumNarudzbine,
DROP COLUMN StatusNarudzbine;
```

Kod:

```
ALTER TABLE DetaljiNarudzbina
ADD CONSTRAINT fk_narudzbina
FOREIGN KEY (ID_Narudzbine)
REFERENCES Narudzbine(ID_Narudzbine);
```

Zadatak 48 – Osiguranje jasne identifikacije svakog reda

```
Kod:
ALTER TABLE Prodaje
ADD CONSTRAINT pk_prodaje PRIMARY KEY (prodaja_id);
ALTER TABLE Klijenti
ADD CONSTRAINT pk_klijenti PRIMARY KEY (klijent_id);
ALTER TABLE Proizvodi
ADD CONSTRAINT pk_proizvodi PRIMARY KEY (proizvod_id);
```

Kod:

```
prodaja_id INT IDENTITY(1,1),
klijent_id INT IDENTITY(1,1),
proizvod_id INT IDENTITY(1,1),

Kod:
ALTER TABLE Prodaje
ADD CONSTRAINT fk_klijent_id
FOREIGN KEY (klijent_id)
REFERENCES Klijenti(klijent_id),
ADD CONSTRAINT fk_proizvod_id
FOREIGN KEY (proizvod_id)
REFERENCES Proizvodi(proizvod_id);
```

Zadatak 49 – Implementiranje upita

Kod:

```
SELECT
    p.ID_Zaposlenog,
```

```
    SUM(p.IznosProdaje) AS UkupnaProdajaZaposlenog,
    (SELECT AVG(SUM(IznosProdaje))
     FROM Prodaja
     GROUP BY ID_Zaposlenog
     ) AS ProsečnaProdajaSvihZaposlenih
FROM
    Prodaja p
GROUP BY
    p.ID_Zaposlenog
HAVING
    SUM(p.IznosProdaje)  >  (SELECT  AVG(SUM(IznosProdaje))
FROM Prodaja GROUP BY ID_Zaposlenog);
```

Zadatak 50 – Implementiranje upita za identifikovanje zaposlenih

Kod:

```
SELECT
    z.ID_Zaposlenog,
    z.Ime,
    z.Prezime,
    CASE
        WHEN pz.ID_Projekta IS NOT NULL
        THEN 'Učestvuje u projektu'
        ELSE 'Ne učestvuje u projektu'
    END AS StatusUčešća
FROM
    Zaposleni z
LEFT JOIN
    ProjektiZaposlenih pz
ON
    z.ID_Zaposlenog = pz.ID_Zaposlenog
WHERE
    z.ID_Zaposlenog  IN  (/*  ovde  navedite  skup  šifri
zaposlenih */)
GROUP BY
    z.ID_Zaposlenog, z.Ime, z.Prezime, pz.ID_Projekta;
```

Zadatak 51 – Pretraga imena zaposlenih

Kod:

```
SELECT
    ID_Zaposlenog,
```

```
    Ime,
    Prezime,
    Email
FROM
    Zaposleni
WHERE
    Ime LIKE 'Mi%' OR
    Prezime LIKE 'Mi%' OR
    Ime LIKE '%nov%' OR
    Prezime LIKE '%nov%';
```

Zadatak 52 – Zajednički elementi dva skupa rezultata

Kod:

```
SELECT
    Z.ID_zaposlenog
FROM
    Zaposleni Z
JOIN
    ProjektiZaposlenih PZ
ON
    Z.ID_zaposlenog = PZ.ID_zaposlenog
JOIN
    Projekti P
ON
    PZ.ID_projekta = P.ID_projekta
WHERE
    P.StatusProjekta = 'Završen'
INTERSECT
SELECT
    Z.ID_zaposlenog
FROM
    Zaposleni Z
JOIN
    ProjektiZaposlenih PZ
ON
    Z.ID_zaposlenog = PZ.ID_zaposlenog
JOIN
    Projekti P
ON
    PZ.ID_projekta = P.ID_projekta
WHERE
    P.StatusProjekta = 'U toku';
```

Zadatak 52 – Redovi jedne tabele povezani sa drugom tabelom

Kod:

```
SELECT
    K.ID_Kupca,
    K.Ime,
    K.Prezime,
    K.Email
FROM
    Kupci K
INNER JOIN
    Narudzbine N ON K.ID_Kupca = N.ID_Kupca
WHERE
  N.DatumNarudzbine  >  CURRENT_DATE  -  INTERVAL  '30'  DAY;
```

Zadatak 53 – Proizvodi koji nikad nisu naručeni

Kod:

```
SELECT
    P.ID_Proizvoda,
    P.NazivProizvoda
FROM
    Proizvodi P
LEFT JOIN
    DetaljiNarudzbina DN ON P.ID_Proizvoda = DN.ID_Proizvoda
WHERE
    DN.ID_Proizvoda IS NULL;
```

Zadatak 54 – Svi kontakti koji odgovaraju određenom kriterijumu

Kod:

```
SELECT
    ID_Kontakta,
    Ime,
    Email
FROM
    Kontakti
WHERE
    Ime LIKE '%Alex%'
```

```
    OR Email LIKE '%@example.com';
```

Zadatak 55 – Podupiti za zaposlene koji imaju platu veću od prosečne plate

Kod:

```
SELECT
    Z.ID_Zaposlenog,
    Z.Ime,
    Z.Prezime,
    Z.Plata,
    Z.ID_Odeljenja
FROM
    Zaposleni Z
WHERE
    Z.Plata > (
        SELECT AVG(Z1.Plata)
        FROM Zaposleni Z1
        WHERE Z1.ID_Odeljenja = Z.ID_Odeljenja
    );
```

Zadatak 56 -Podupiti za proizvode kojima je cena niža od prosečne

Kod:

```
SELECT P.ID_Proizvoda, P.Naziv
FROM Proizvodi P
WHERE P.Cena < (
    SELECT AVG(P1.Cena)
    FROM Proizvodi P1
) AND P.ID_Proizvoda NOT IN (
    SELECT N.ID_Proizvoda
    FROM Narudzbine N
);
```

Zadatak 57 – Skalarni podupit za izračunavanje razlike u plati

Kod:

```
SELECT
```

```
    ID_zaposlenog,
    Ime,
    Prezime,
    Plata,
    (SELECT MAX(Plata) FROM Zaposleni) - Plata
      AS RazlikaUPlati
FROM
    Zaposleni;
```

Zadatak 58 – Podupit za izračuvanje broja dana

```
SELECT
    K.Ime,
    K.Prezime,
    DATEDIFF(day, GETDATE(), MIN(N.DatumNarudzbine))
      AS DaniDoNajblizeNarudzbine
FROM
    Kupci K
LEFT JOIN
    Narudzbine N ON K.ID_kupca = N.ID_kupca
GROUP BY
    K.Ime, K.Prezime;
```

Zadatak 59 – Podupit kao izraz kolona

Kod:

```
SELECT
    K.Ime,
    K.Prezime,
    (
        SELECT
            DATEDIFF(day, GETDATE(), MIN(N.DatumNarudzbine))
        FROM
            Narudžbine N
        WHERE
            N.ID_kupca = K.ID_kupca
            AND N.DatumNarudzbine > GETDATE()
    ) AS DaniDoNajblizeNarudzbine
FROM
    Kupci K;
```

Zadatak 60 – Podupit kao filter za identifikaciju zaposlenih

Kod:

```
SELECT
    Z.ID_zaposlenog,
    Z.Ime,
    Z.Prezime
FROM
    Zaposleni Z
WHERE
    EXISTS (
        SELECT
            1
        FROM
            Projekti P
        WHERE
            P.ID_zaposlenog = Z.ID_zaposlenog
            AND P.Oblast = 'IT'
    );
```

Zadatak 61 – Generisanje sledeće vrednosti primarnog ključa

Kod:

```
INSERT INTO Narudzbine
   (DatumNarudzbine, ID_kupca, UkupanIznos)
VALUES
   (GETDATE(), @ID_kupca, @UkupanIznos);
```

Zadatak 62 – Nevezani podaci

Kod:

```
SELECT
    Z.ID_zaposlenog,
    Z.Ime,
    Z.Prezime
FROM
    Zaposleni Z
LEFT JOIN
```

```
    ProjektiZaposlenih PZ
ON
    Z.ID_zaposlenog = PZ.ID_zaposlenog
WHERE
    PZ.ID_projekta IS NULL;
```

Zadatak 63 – Referentna tabela za rešavanje problema

Kod:

```
SELECT
    D.Termin
FROM
    Driver D
LEFT JOIN
    Termini T ON D.Termin >= T.Početak
    AND D.Termin < T.Završetak
WHERE
    T.ID_termina IS NULL
AND
    D.Termin BETWEEN '2023-01-01' AND '2023-01-31';
```

Zadatak 64 – Identifikacija kupaca koji su izvršili više od jedne kupovine

Kod:

```
SELECT
    K.ID_kupca,
    COUNT(*) AS BrojKupovina
FROM
    Kupovine K
WHERE
    K.DatumKupovine BETWEEN DATEADD(day, -30, GETDATE())
    AND GETDATE()
    AND K.ID_kupca NOT IN (
        SELECT DISTINCT K2.ID_kupca
        FROM Kupovine K2
        WHERE   K2.DatumKupovine   <   DATEADD(month,   -6,
DATEADD(day, -30, GETDATE()))
    )
GROUP BY
    K.ID_kupca
HAVING
    COUNT(*) > 1;
```

Zadatak 65 – Identifikacija potencijalnih grešaka u unosu podataka

Kod:

```
SELECT
    ID_zaposlenog,
    Ime,
    Prezime,
    Email,
    DatumRođenja,
    Plata
FROM
    Zaposleni
WHERE
-- Provera nevažećih email adresa
    Email NOT LIKE '%@%'

-- Pretpostavljamo da su plate iznad 100,000 neobično visoke
    OR Plata > 100000
-- Neobični datumi rođenja
    OR YEAR(DatumRođenja) < 1900
    OR YEAR(DatumRođenja) > YEAR(GETDATE()) - 18;
```

Zadatak 66 – Rešavanje problema redundacije podataka

Kod:

```
SELECT
    Z.ID_zaposlenog,
    Z.Ime,
    Z.Prezime,
    K.VrstaKontakta,
    MAX(K.VrednostKontakta) AS VrednostKontakta
FROM
    Zaposleni Z
JOIN
    Kontakti K ON Z.ID_zaposlenog = K.ID_zaposlenog
GROUP BY
    Z.ID_zaposlenog,
    Z.Ime,
    Z.Prezime,
    K.VrstaKontakta;
```

Zadatak 67 – Kontrola zaključavanja za optimizaciju performansi

Kod

```
BEGIN TRANSACTION;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE Narudžbine
SET status_narudžbine = 'Obradjena'
WHERE ID_kupca = @IDKupca
  AND status_narudžbine = 'Na čekanju';

COMMIT TRANSACTION;
```

Zadatak 68 - Rešavanje konflikta između konkurentnih operacija

Kod:

```
BEGIN TRANSACTION;

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

SELECT dostupna_kolicina
FROM Zalihe
WHERE ID_artikla = @IDArtikla FOR UPDATE;

UPDATE Zalihe
SET dostupna_količina = dostupna_kolicina - @Kolicina
WHERE ID_artikla = @IDArtikla;

COMMIT TRANSACTION;
```

Zadatak 69 – Dodeljivanje privilegija za pristup i manipulaciju podacima

Kod:

```
-- Dodeljivanje SELECT, INSERT i UPDATE privilegija
-- na tabelu Prodaja
GRANT SELECT, INSERT, UPDATE ON KompanijaDB.Prodaja
TO AnalitikaTim;
```

```
-- Dodeljivanje SELECT, INSERT i UPDATE privilegija
-- na tabelu Kupci
GRANT SELECT, INSERT, UPDATE ON KompanijaDB.Kupci
TO AnalitikaTim;
```

Zadatak 70 – Dizajniranje „od vrha do dna” baze podataka

Kod:

```
SELECT
   P.ID_proizvoda,
   P.naziv_proizvoda,
   SUM(Pr.iznos_prodaje) AS ukupan_iznos_prodaje
FROM Prodaja Pr
JOIN Proizvodi P ON Pr.ID_proizvoda = P.ID_proizvoda
WHERE YEAR(Pr.datum_prodaje) = YEAR(CURRENT_DATE)
GROUP BY P.ID_proizvoda, P.naziv_proizvoda
ORDER BY ukupan_iznos_prodaje DESC
LIMIT 10;
```