1. Wyświetl średnią zarobków dla każdego oddziału.
2. Do wyników poprzedniego zapytania dodaj wiersz wyświetlający średnią firmy.
3. Wyświetl sumę zarobków dla poszczególnych identyfikatorów stanowisk dla każdego identyfikatora działu.
4. Do wyników poprzedniego zapytania dodaj wiersz podsumowań dla poszczególnych identyfikatorów stanowisk i podsumowanie całościowe.
5. Do wyników poprzedniego zapytania dodaj wiersze podsumowań dla poszczególnych działów.
6. Zmodyfikuj poprzednie zapytanie tak aby zamiast identyfikatorów wyświetlały się nazwy działów i nazwy stanowisk.
7. Wyświetl wartość funkcji grouping dla zapytania wyświetlającego maksymalną wartość pensji dla poszczególnych stanowisk z użyciem operatora rollup.
8. Wykorzystaj wyrażenie case do wyświetlenia słowa ‘Razem’ dla podsumowań z poprzedniego zadania.
9. Wyświetl minimalną i maksymalna pensję dla każdego stanowiska w każdym dziale oraz podsumowania całościowe i częściowe. Zadbaj aby w wierszach podsumowań znalazły się wyrażenia ‘Wszystkie stanowiska’, ‘Wszystkie oddziały’.
10. Wyświetl sumę wynagrodzeń dla każdego działu i każdego stanowiska (wykorzystaj GROUPING SETS i UNION ALL).
11. Wyświetl podsumowania wynagrodzeń dla poszczególnych stanowisk w każdym dziale, dla każdego stanowiska i podsumowanie całościowe (wykorzystaj GROUPING SETS i UNION ALL).
12. Oblicz średnią pensję pracowników na poziomach: całej firmy, każdego przedziału lat pracy (`EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM hire_date)`), każdego działu w danym przedziale lat pracy.
13. Oblicz średnią liczbę lat pracy w podziale na departamenty, stanowiska i lokalizacje.
14. Oblicz sumę pensji na poziomach: całej firmy, każdego regionu, każdego kraju w regionie.
15. Policz liczbę zatrudnionych pracowników dla różnych poziomów: cała firma, każdy rok zatrudnienia, każdy dział w danym roku.
16. Oblicz sumę premii na poziomach: całej firmy, każdego działu, każdego stanowiska w dziale.
17. Podsumuj średnie wynagrodzenie na poziomie: każdego managera, departamentu, całej firmy.
18. Przygotuj raport z liczbą pracowników dla różnych poziomów: każdego stanowiska, departamentu, przełożonego.

```
--1
SELECT department_name, round(AVG(salary),0)
FROM hr.employees JOIN hr.departments USING(department_id)
GROUP BY department_name;
--2 UNION
SELECT department_name, round(AVG(salary),0)
FROM hr.employees JOIN hr.departments USING(department_id)
GROUP BY department_name
UNION ALL
SELECT 'Razem', round(AVG(salary),0)
FROM hr.employees JOIN hr.departments USING(department_id);
-- ROLLUP
SELECT department_name, round(AVG(salary),0)
FROM hr.employees JOIN hr.departments USING(department_id)
GROUP BY ROLLUP(department_name);
-- GROUPING SETS
SELECT decode(grouping(department_name),1,'Razem',department_name) as departament,
round(AVG(salary),0)
FROM hr.employees JOIN hr.departments USING(department_id)
GROUP BY GROUPING SETS((department_name),()); --()-podsum. all
--3
SELECT department_id, job_id, SUM(salary) AS total_salary
FROM hr.employees
GROUP BY department_id, job_id;
--4
--UNION
SELECT department_id, job_id, SUM(salary) AS total_salary
FROM hr.employees
GROUP BY department_id, job_id
UNION ALL
SELECT NULL,job_id, SUM(salary)
FROM hr.employees
GROUP BY job_id
UNION ALL
SELECT NULL, NULL, SUM(salary)
FROM hr.employees;
--ROLLUP
SELECT department_id, job_id, SUM(salary) AS total_salary
FROM hr.employees
GROUP BY department_id,ROLLUP(job_id)
UNION ALL
SELECT NULL, NULL, SUM(salary)
FROM hr.employees;
--GROUPING SET
SELECT department_id, job_id, SUM(salary) AS total_salary
FROM hr.employees
GROUP BY GROUPING SETS ((department_id, job_id), (job_id), ()); --()-podsum.all
--5
SELECT e.department_id, e.job_id, SUM(e.salary) AS suma_zarobkow
FROM hr.employees e
GROUP BY e.department_id, e.job_id
UNION ALL
SELECT department_id, NULL, SUM(salary)
FROM hr.employees
GROUP BY department_id
UNION ALL
SELECT NULL, job_id, SUM(salary)
FROM hr.employees
GROUP BY job_id
```

```
UNION ALL
SELECT NULL, NULL, SUM(salary)
FROM hr.employees;
--CUBE
SELECT e.department_id, e.job_id, SUM(e.salary) AS suma_zarobkow
FROM hr.employees e
GROUP BY CUBE(e.department_id, e.job_id);
--CUBE --wyswietla nazwy dla podsumowa (zamiast null)
SELECT
decode(grouping(e.department_id),1,'Razem departamenty',e.department_id) as
id_departamentu,
decode(grouping(e.job_id),1,'Razem stanowiska',e.job_id) as id_stanowiska,
SUM(e.salary) AS suma_zarobkow
FROM hr.employees e
GROUP BY CUBE(e.department_id, e.job_id);
--6
SELECT
decode(grouping(d.department_name),1,'Razem departamenty',d.department_name) as
id_departamentu,
decode(grouping(j.job_title),1,'Razem stanowiska',j.job_title) as id_stanowiska,
SUM(e.salary) AS suma_zarobkow
FROM hr.employees e join hr.departments d using(department_id) join hr.jobs j
using(job_id)
GROUP BY CUBE(d.department_name, j.job_title);
--7
SELECT JOB_ID, MAX(SALARY), GROUPING(JOB_ID)
FROM hr.employees
GROUP BY ROLLUP(JOB_ID);
--8
SELECT CASE WHEN job_id IS NULL THEN 'Razem' ELSE job_id END AS job_id,
MAX(SALARY), GROUPING(JOB_ID)
FROM hr.employees
GROUP BY ROLLUP(JOB_ID);
--zad 9
SELECT CASE
         WHEN GROUPING(d.department_name) = 1 THEN 'Wszystkie oddzia y'
         ELSE d.department_name
       END AS department_name,
       CASE
         WHEN GROUPING(j.job_title) = 1 THEN 'Wszystkie stanowiska'
         ELSE j.job_title
       END AS job_title,
       MIN(e.salary) AS min_salary,
       MAX(e.salary) AS max_salary
FROM hr.employees e
JOIN hr.departments d ON e.department_id = d.department_id
JOIN hr.jobs j ON e.job_id = j.job_id
GROUP BY CUBE(d.department_name, j.job_title);
--10
--GROUPING SETS
SELECT department_id, job_id, SUM(salary) AS total_salary
FROM HR.employees
GROUP BY GROUPING SETS ((department_id, job_id), (department_id), (job_id), ());
--UNION ALL
SELECT department_id, job_id, SUM(salary) AS total_salary
FROM HR.employees
GROUP BY department_id, job_id
UNION ALL
SELECT department_id, NULL AS job_id, SUM(salary) AS total_salary
```

```
FROM HR.employees
GROUP BY department_id
UNION ALL
SELECT NULL AS department_id, job_id, SUM(salary) AS total_salary
FROM HR.employees
GROUP BY job_id
UNION ALL
SELECT NULL AS department_id, NULL AS job_id, SUM(salary) AS total_salary
FROM HR.employees;
--12
SELECT
    DECODE(GROUPING(department_id), 1, 'Razem departamenty', department_id) AS
id_departamentu,
    DECODE(GROUPING(EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM hire_date)), 1,
'Razem lata pracy',
           EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM hire_date)) AS
years_worked,
    ROUND(AVG(salary), 0) AS avg_salary
FROM hr.employees
GROUP BY GROUPING SETS(
    (department_id, EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM hire_date)),
    (department_id),
    (EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM hire_date)),
    ()
);
```

1. Wyświetl średnią zarobków w firmie.
2. Wyświetl ranking pracowników według zarobków.
3. Wyświetl średnią zarobków dla każdego oddziału.
4. Zbuduj ranking zarobków pracowników w każdym dziale.
5. Wyznacz skumulowaną sumę zarobków dla pracowników (kolejność zatrudnienia).
6. Zbuduj ranking zarobków pracowników według stanowisk, a następnie dla każdego stanowiska według grup podległych każdemu kierownikowi.
7. Zbuduj ranking zarobków w każdym dziale, ale tylko wśród pracowników zatrudnionych w latach 2003-2005.
8. Nadaj kolejne numery rekordom w rankingu pracowników, zbudowanym z podziałem na kolejne lata zatrudnienia.
9. Wyznacz procentowy rozkład zarobków pracowników w działach (ile osób procentowo w dziale zarabia **tyle samo lub mniej** ile dany pracownik, uwzględniamy danego pracownika).
10.Wyznacz procentowy rozkład zarobków pracowników na poszczególnych stanowiskach (ile osób procentowo na danym stanowisku zarabia **tyle samo lub mniej** ile dany pracownik, ale bez uwzględnienia niego).
11. Zbuduj ranking zarobków pracowników w każdym dziale. Wyświetl tylko po 3 najlepiej zarabiających.
12. Wylicz średnią wartość zarobków w departamencie 80 z pracowników zatrudnionych w bieżącym miesiącu i trzech miesiącach poprzedzających bieżący miesiąc.
13. Wyznacz ranking hipotetyczny dla pracownika o zarobkach 4200.
14. Na którym miejscu w rankingu stanowisk znalazłoby się stanowisko, dla którego średnia wynagrodzeń wynosi 17000. Ile % stanowisk poprzedzałoby to stanowisko.
15. Dla każdego pracownika wyświetl imię i nazwisko, id oddziału, liczbę pracowników w oddziale, skumulowaną sumę zarobków w oddziałach, udział kwoty wynagrodzenia pracownika w całości wynagrodzenia w oddziale.
16. Pogrupuj pracowników na dobrze i słabo zarabiających. Analizowany próg wynagrodzenia to 8000.
17. Porównaj sumy zarobków pracowników zatrudnionych w kolejnych latach. Wyświetl różnicę sumy zarobków w stosunku do poprzedniego roku i następnego.

```
--1
SELECT AVG(SALARY) FROM HR.EMPLOYEES;
--2
--rank()--przerwa w numeracji
SELECT first_name, last_name, salary,
       RANK() OVER (ORDER BY salary DESC) AS salary_rank
FROM hr.employees;
--dense_rank()--bez przerwy w numeracji
SELECT first_name, last_name, salary,
       DENSE_RANK() OVER (ORDER BY salary DESC) AS salary_rank
FROM hr.employees;
--3
SELECT department_id, round(AVG(salary),2) AS avg_salary
FROM hr.employees
group by department_id ;
--
SELECT  distinct department_id, AVG(salary) OVER (PARTITION BY department_id) AS
avg_salary
FROM hr.employees;
--4
SELECT first_name, last_name, salary, department_id,
       RANK() OVER (PARTITION BY department_id ORDER BY salary DESC) AS
salary_rank_in_department,
       DENSE_RANK() OVER (PARTITION BY department_id ORDER BY salary DESC) AS
salary_dense_rank_in_department
FROM hr.employees;
--5
SELECT employee_id, first_name || ' ' || last_name AS employee_name, hire_date,
salary,
SUM(salary) OVER (ORDER BY hire_date) AS cumulative_salary
FROM hr.employees;
--6
select first_name || ' ' ||last_name as employee, job_id, manager_id,salary,
RANK() OVER (PARTITION BY job_id, manager_id ORDER BY salary DESC) AS salary_rank
from hr.employees;
--7
SELECT department_id, employee_id, last_name, salary,
DENSE_RANK() OVER (PARTITION BY department_id ORDER BY salary DESC) AS
ranking_zarob
FROM hr.employees
WHERE hire_date BETWEEN DATE '2003-01-01' AND DATE '2005-12-31';
--8
select first_name as imie, last_name as nazwisko, extract(year from hire_date) as
rok_zatrudnienia,
row_number() over(partition by 3 order by extract(year from hire_date)) as
ranking_zatrudnienia
from hr.employees;
--9 --Jaki % osob zarabia wiecej w danej partycji
--(order by salary desc; mniej- order by asc) niz  biezacy rekord (z biezacym
rekordem)
select department_id, last_name, salary,
cume_dist() over (partition by department_id order by salary desc) as procent
from hr.employees;
--10 --Jaki % osob zarabia wiecej w danej partycji
--(order by salary desc; mniej- order by asc) niz  biezacy rekord (bez biezacego
rekordu)
select department_id, last_name, salary,
percent_rank() over (partition by department_id order by salary desc) as procent
from hr.employees;
```

```
--11
select department_id,last_name,first_name,salary,
rank from
(select department_id,last_name,first_name,salary,
rank() OVER (PARTITION by department_id order by salary desc) as rank
from hr.employees)
where rank <=3;
--12 --okno:w biezacym miesiacu i trzech miesiacach poprzedzajacych biezacy miesiac
SELECT salary, hire_date, department_id,
(sum(salary) OVER (ORDER BY hire_date RANGE BETWEEN interval '3'
month PRECEDING AND interval '0' month FOLLOWING)) srednia
FROM HR.employees
WHERE department_id = 80
ORDER BY hire_date;
--okno: biezacy rekord i 3 nastepne
SELECT salary, hire_date, department_id,
AVG(salary) OVER (ORDER BY hire_date rows between current row and 3 following)
srednia
FROM HR.employees WHERE department_id = 80 ORDER BY hire_date;

--okno: 5 wierszy- biezacy i po 2 przed i 2 po biezacym wierszu --w calej firmie
SELECT salary, hire_date, department_id,
sum(salary) OVER (ORDER BY hire_date rows BETWEEN 2 PRECEDING AND 2 FOLLOWING) AS
suma
FROM hr.employees
ORDER BY hire_date;
--okno: wiersze biezacy i 4 wiersze przed biezacym rekordzie --w calej firmie
select salary, hire_date, department_id,
sum(salary) over (order by hire_date rows between 4 preceding and current row) suma
from hr.employees order by hire_date;
--13
select rank(4200) within group (order by salary) rank_hipoteczny from hr.employees;
--14
--dla departamentu
select rank(17000) within group (order by avg(salary) desc) pozycja,
percent_rank(17000)within group (order by avg(salary) desc) procentowo
from hr.employees group by department_id;
--dla stanowiska
select rank(17000) within group (order by avg(salary)) pozycja,
percent_rank(17000)within group (order by avg(salary)) procentowo
from hr.employees group by job_id;
--15
SELECT e.first_name, e.last_name, e.department_id, e.salary, COUNT(*) OVER
(PARTITION BY e.department_id) AS liczba_pracownikow,
SUM(e.salary) OVER (PARTITION BY e.department_id) AS skumulowana_suma_zarobkow,
(e.salary / SUM(e.salary) OVER (PARTITION BY e.department_id)) * 100 AS
udzial_procentowy_w_oddziale
FROM employees e;
--
SELECT e.first_name, e.last_name, e.department_id, e.salary, COUNT(*) OVER
(PARTITION BY e.department_id) AS liczba_pracownikow,
SUM(e.salary) OVER (PARTITION BY e.department_id) AS skumulowana_suma_zarobkow,
ratio_to_report(salary) OVER (PARTITION BY e.department_id) AS
udzial_procentowy_w_oddziale
FROM employees e;
--16
--podziel pensje w firmie na 2 przedzialy
SELECT last_name, salary,
WIDTH_BUCKET(salary, (select min(salary) from HR.employees),
```

```
                    (select max(salary) from HR.employees) , 2) as w_bucket
FROM HR.employees ORDER BY salary;
--case
SELECT
    employee_id, first_name,last_name,salary,
    CASE
        WHEN salary >= 8000 THEN 'Dobrze zarabiaj cy'
        ELSE 'S abo zarabiaj cy'
    END AS status_zarobku
FROM
    hr.employees order by salary;
--WIDTH_BUCKET
SELECT last_name, salary,
CASE
WHEN WIDTH_BUCKET(salary, 8000, 8000 , 1) = 0 THEN 'Slabo zarabiajacy'
ELSE 'Dobrze zarabiajacy'
END as w_bucket FROM HR.employees ORDER BY salary;
--17
SELECT hire_year,
       sum_salary,
       sum_salary - LAG(sum_salary) OVER (ORDER BY hire_year) AS
diff_from_prev_year,
       LEAD(sum_salary) OVER (ORDER BY hire_year) - sum_salary AS diff_to_next_year
FROM (
    SELECT EXTRACT(YEAR FROM hire_date) AS hire_year,
           SUM(salary) AS sum_salary
    FROM employees
    GROUP BY EXTRACT(YEAR FROM hire_date)
)
ORDER BY hire_year;

--17
SELECT
    EXTRACT(YEAR FROM hire_date) AS year,
    SUM(salary) AS total_salary,
    SUM(salary) - LAG(SUM(salary)) OVER (ORDER BY EXTRACT(YEAR FROM hire_date)) AS
salary_diff_previous_year,
    SUM(salary) - LEAD(SUM(salary)) OVER (ORDER BY EXTRACT(YEAR FROM hire_date)) AS
salary_diff_next_year
FROM
    hr.employees
GROUP BY
    EXTRACT(YEAR FROM hire_date)
ORDER BY
    year;
```

1. Wyświetl w postaci drzewa hierarchię zatrudnienia na podstawie tabeli employees. Wyniki mają być posortowane w taki sposób, by na początku byli wymienieni pracownicy najwyższego stopnia, a następnie kolejni wg stopnia.
2. Napisz zapytanie, które wyświetli imiona, nazwiska i identyfikatory pracowników oraz identyfikatory przełożonych i identyfikatory stanowisk w porządku hierarchicznym rozpoczynając od prezesa.
3. Wyświetl podwładnych Higginsa.
4. Wyświetl przełożonych Higginsa.
5. Wyświetl pracownika o identyfikatorze 105 i jego zwierzchników
6. Dla każdego pracownika (imię połączone z nazwiskiem) wyświetl numer poziomu w hierarchii.
7. Uzyskaj następujący efekt:
a)

POZIOM	IDENTYFIKATOR	NAZWISKO	HIERARCHIA
1	100	King	*King
2	101	Kochhar	*King*Kochhar
3	108	Greenberg	*King*Kochhar*Greenberg
4	109	Faviet	*King*Kochhar*Greenberg*Faviet
4	110	Chen	*King*Kochhar*Greenberg*Chen
4	111	Sciarra	*King*Kochhar*Greenberg*Sciarra
4	112	Urman	*King*Kochhar*Greenberg*Urman
4	113	Popp	*King*Kochhar*Greenberg*Popp
3	200	Whalen	*King*Kochhar*Whalen
3	203	Mavris	*King*Kochhar*Mavris
3	204	Baer	*King*Kochhar*Baer

b)

POZIOM	IDENTYFIKATOR	NAZWISKO	HIERARCHIA
1	100	King	King Steven
2	101	Kochhar	Kochhar Neena
3	108	Greenberg	Greenberg Nancy
4	109	Faviet	Faviet Daniel
4	110	Chen	Chen John
4	111	Sciarra	Sciarra Ismael
4	112	Urman	Urman Jose Manuel
4	113	Popp	Popp Luis
3	200	Whalen	Whalen Jennifer
3	203	Mavris	Mavris Susan
3	204	Baer	Baer Hermann

8. Dla każdego pracownika (identyfikator, imię, nazwisko) podaj nazwisko prezesa.
9. Dla każdego pracownika (identyfikator, imię, nazwisko) podaj imię i nazwisko prezesa (dane prezesa w jednej kolumnie).
10. Napisz zapytanie, które wyświetli zawartość tabeli employees w porządku hierarchicznym rozpoczynając od prezesa, ale bez Greenberga i jego podwładnych.
11. Napisz zapytanie, które wyświetli zawartość tabeli employees w porządku hierarchicznym rozpoczynając od Kochhara, ale bez Chena i Poppa.
12. Wyświetl identyfikator, imię, nazwiska, datę zatrudnienia, stanowisko, wydział oraz poziom zagnieżdżenia w hierarchii firmy, ale tylko pracowników najniższego szczebla.

```
--1
SELECT LEVEL, LPAD(' ', 4*(LEVEL-1)) || first_name || ' ' || last_name AS pracowncy
FROM HR.employees
CONNECT by PRIOR employee_id = manager_id
START WITH manager_id IS NULL  --prezes -- employee_id=100 --min employee_id
ORDER by LEVEL;
----2
SELECT level,employee_id, first_name, last_name, manager_id, job_id
FROM hr.employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
ORDER by LEVEL;
--3
SELECT level,employee_id, first_name, last_name
from hr.employees
connect by prior employee_id=manager_id
start with employee_id=
(select employee_id from hr.employees where last_name='Higgins');
--4
SELECT level, employee_id, first_name, last_name
from hr.employees
connect by prior manager_id=employee_id
start with employee_id=
(select employee_id from hr.employees where last_name='Higgins');

--5
SELECT level, employee_id, first_name, last_name, job_id, manager_id
from hr.employees
CONNECT by PRIOR manager_id = employee_id
START WITH employee_id=105;
--6
SELECT first_name || ' ' || last_name AS "Employee", LEVEL AS "Hierarchia poziom"
FROM hr.employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
ORDER BY "Hierarchia poziom";
--7a
SELECT
    LEVEL AS "POZIOM",
    employee_id AS "IDENTYFIKATOR",
    last_name AS "NAZWISKO",
    SYS_CONNECT_BY_PATH(first_name || ' ' || last_name, '*') AS "HIERARCHIA"
FROM hr.employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
ORDER SIBLINGS BY last_name; --ORDER SIBLINGS BY -- hierarchie
--7b
SELECT LEVEL AS "POZIOM",
       employee_id AS "IDENTYFIKATOR",
       last_name AS "NAZWISKO",
       LPAD(' ', 4*(LEVEL-1)) || first_name || ' ' || last_name AS "Hierarchy"
FROM hr.employees
START WITH manager_id IS NULL  -- employee_id=100 --min employee_id
CONNECT BY PRIOR employee_id = manager_id
ORDER SIBLINGS BY last_name;
--8
SELECT LEVEL, EMPLOYEE_ID,first_name, last_name, CONNECT_BY_ROOT last_name AS
"PREZES"
FROM HR.EMPLOYEES
```

```
START WITH EMPLOYEE_ID = 100
CONNECT BY manager_id = PRIOR EMPLOYEE_ID;
--
SELECT e.employee_id, e.first_name, e.last_name, p.last_name as "Prezes Nazwisko"
FROM hr.employees e
CROSS JOIN hr.employees p
WHERE p.employee_id = (
      SELECT employee_id
      FROM hr.employees
      START WITH manager_id IS NULL
      CONNECT BY PRIOR employee_id = manager_id
      FETCH FIRST 1 ROW ONLY
)
ORDER BY e.employee_id;
--9
SELECT e.employee_id, e.first_name, e.last_name,
       (SELECT p.first_name || ' ' || p.last_name
        FROM hr.employees p
        START WITH p.manager_id IS NULL
        CONNECT BY PRIOR p.employee_id = p.manager_id
        FETCH FIRST 1 ROW ONLY) AS "Prezes"
FROM hr.employees e
ORDER BY e.employee_id;
--10
SELECT LPAD(' ', 4*(LEVEL-1)) || first_name || ' ' || last_name AS "Hierarchy"
FROM hr.employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
AND last_name != 'Greenberg'
ORDER SIBLINGS BY employee_id;
--11
SELECT LPAD(' ', 4*(LEVEL-1)) || first_name || ' ' || last_name AS "Hierarchy"
FROM hr.employees
WHERE last_name NOT IN ('Chen', 'Popp')
START WITH last_name = 'Kochhar'
CONNECT BY PRIOR employee_id = manager_id
ORDER SIBLINGS BY last_name;
--12
SELECT
    e.employee_id,
    e.first_name,
    e.last_name,
    e.hire_date,
    e.job_id,
    e.department_id,
    LEVEL as "Poziom Zagnie d enia"
FROM hr.employees e
WHERE LEVEL = (
    SELECT MAX(LEVEL)
    FROM hr.employees
    START WITH manager_id IS NULL
    CONNECT BY PRIOR employee_id = manager_id
)
START WITH e.manager_id IS NULL
CONNECT BY PRIOR e.employee_id = e.manager_id
ORDER BY e.employee_id;
--
SELECT
    e.employee_id,
```

```
    e.first_name,
    e.last_name,
    e.hire_date,
    e.job_id,
    e.department_id,
    LEVEL as "Poziom Zagnie d enia"
FROM hr.employees e
WHERE LEVEL = 4  --mniej uniwersalne
START WITH e.manager_id IS NULL
CONNECT BY PRIOR e.employee_id = e.manager_id
ORDER BY e.employee_id;
```

1) Utwórz tabelę
Studenci (nr_indeksu number(4), imię varchar2(20), nazwisko varchar2(50), pkt_egz number(3), zal_egz number(1) domyślnie 0)
Studenci_egzamin(nr_indeksu number(4), pkt_egz number(3)).
Wstaw do tabeli Studenci 6 rekordów, do tabeli Studenci_egzamin 4 rekordy spośród 6.
Scal dane w tabeli Studenci w następujący sposób:
-przepisz punkty z egzaminu
-zaznacz zaliczenie egzaminu dla studentów, którzy uzyskali więcej niż 50 punktów z egzaminu (zal_egz: 0 lub 1(zadał))
-studenci, którzy nie przyszli na egzamin powinni mieć punktację zerową.

2) Stwórz tabele:
Towary(IdTow, Nazwa, Cena) i wstaw kilka produktów.
TowaryUaktualnione(IdTow, Nazwa, Cena) – kopia tabeli Towary z cenami po 10% podwyżce i kilka nowych produktów.
Scal dane: jeśli IdTow są równe to dane w tabeli Towary powinny zostać uaktualnione zgodnie z tym co jest zapisane w tabeli TowaryUaktualnione, jeżeli w tabeli Towary nie występuje jakiś towar to powinien zostać dopisany do tabeli.

3) Stwórz tabelę Gazownia(id_klienta, stan_licznika_poprz, stan_licznika_akt) i dodaj 10 rekordów oraz tabelę Odczyt_gaz(id_klienta, stan_licznika) – te same rekordy do w tabeli Gazownia, stan_licznika to stan_licznika_akt z Gazownia zwiększony o 10%.
Wprowadź zmiany po odczycie licznika do tabeli Gazownia, tak aby stan_licznika_poprz był aktualnym, a aktualny z odczytów.

4) Utwórz tabelę Dluznicy_spoldzielni(id_klienta, zaleglosc) i wstaw 10 rekordów oraz stwórz tabelę Dluznicy_aktualizacja(id_klienta,zaleglosc) – kopia tabeli Dluznicy_spoldzielni bez 3 rekorów i zaleglosc zwiększona o 5%.
Uaktualnij listę dłużników: część dłużników zwiększa swoje zaległości, a część je spłaca, osoby które spłacą należność zostają skreślone z listy dłużników spółdzielni.

5) Utwórz tabele pracownicy:
prac_id NUMBER(5)
imie VARCHAR2(20)
nazwisko VARCHAR2(20)
id_od NUMBER(5)
pensja NUMBER(10)
i premie:
prac_id NUMBER(5)
premia NUMBER(4,2) – wartość domyślna 0.2.
Wstaw dane 5 pracowników do tabeli pracownicy i dla 4 pracowników ustaw domyślną wartość premii.
Scal dane w tabeli premie w następujący sposób: jeśli identyfikatory pracowników w obu tabelach są równe to ustal premie na poziomie 10% pensji pod warunkiem, że pensja wynosi co najmniej 4800 zł, jeśli pensja jest mniejsza od 4800 zł to pracownik nie ma premii (jego dane nie powinny pojawić się tabeli premie), jeśli identyfikator pracownika nie występuje w tabeli premie to wstaw dane pracownika do tabeli premie i ustal premie na poziomie 20% pensji pod warunkiem, że pensja wynosi co najmniej 4800zł.

1) Skopiuj 10 najlepiej zarabiających pracowników do tabeli TOP_EMPLOYEES.(INSERT INTO SELECT)
2) Podziel pracowników na dwie tabele: SENIOR_EMPLOYEES dla osób zarabiających powyżej 10 000 oraz JUNIOR_EMPLOYEES dla pozostałych.(INSERT ALL)
3) Podnieś pensję pracowników poniżej średniej w ich dziale do wartości średniej. (MERGE)
4) Zlicz liczbę pracowników w każdym dziale i zapisz te dane do tabeli DEPARTMENT_STATS. (WITH + INSERT INTO SELECT)
5) Przenieś wszystkich menedżerów do tabeli MANAGERS.(INSERT ALL)
6) Przenieś pracowników, którzy pracują dłużej niż 25 lat, do tabeli RETIREMENT_EMPLOYEES.(WITH + INSERT INTO SELECT)
7) Zwiększ pensję menedżerów o 15% lub dodaj ich do tabeli BONUS_MANAGERS, jeśli jeszcze tam nie są.(MERGE)
8) Znajdź najlepiej zarabiającego pracownika w każdym dziale i zapisz dane do tabeli TOP_EARNERS.(WITH + INSERT INTO SELECT)
9) Podziel pracowników na HIGH_SALARY_EMPLOYEES (> 15 000) i MID_SALARY_EMPLOYEES (7 000 – 15 000).(INSERT ALL)
10) Jeśli menedżer istnieje w MANAGER_INFO, aktualizuj jego pensję. Jeśli nie, dodaj go.(MERGE)
11) Zlicz liczbę pracowników w regionach i zapisz do REGION_EMPLOYEE_COUNT.(WITH + INSERT INTO SELECT)
12) Zapisz do TOP_PERFORMERS najlepiej zarabiającego pracownika w każdym dziale.(WITH + INSERT INTO SELECT)
13) Zapisz do US_EMPLOYEES wszystkich pracowników pracujących w USA.(INSERT INTO SELECT)
14) Zaktualizuj w tabeli JOBS maksymalną pensję dla każdego stanowiska na podstawie rzeczywistych zarobków.(MERGE)
15) Zapisz do SALARY_HISTORY pensje pracowników sprzed podwyżki.(WITH + INSERT INTO SELECT)

```
--1
CREATE TABLE StudenciA (
  nr_indeksu NUMBER(4),
  imie VARCHAR2(20),
  nazwisko VARCHAR2(50),
  pkt_egz NUMBER(3),
  zal_egz NUMBER(1) DEFAULT 0
);

CREATE TABLE StudenciA_egzamin (
  nr_indeksu NUMBER(4),
  pkt_egz NUMBER(3)
);

INSERT INTO StudenciA (nr_indeksu, imie, nazwisko, pkt_egz) VALUES (1234, 'Jan',
'Kowalski', 0);
INSERT INTO StudenciA (nr_indeksu, imie, nazwisko, pkt_egz) VALUES (2345,
'Aleksander', 'Kot', 0);
INSERT INTO StudenciA (nr_indeksu, imie, nazwisko, pkt_egz) VALUES (3456, 'Tomasz',
'Dom', 0);
INSERT INTO StudenciA (nr_indeksu, imie, nazwisko, pkt_egz) VALUES (4567,
'Dominika', 'Super', 0);
INSERT INTO StudenciA (nr_indeksu, imie, nazwisko, pkt_egz) VALUES (5678, 'Ola',
'Kowalska', 0);
INSERT INTO StudenciA (nr_indeksu, imie, nazwisko, pkt_egz) VALUES (6789, 'Klara',
'Grosicka', 0);
INSERT INTO StudenciA_egzamin (nr_indeksu, pkt_egz) VALUES (1234, 55);
INSERT INTO StudenciA_egzamin (nr_indeksu, pkt_egz) VALUES (2345, 65);
INSERT INTO StudenciA_egzamin (nr_indeksu, pkt_egz) VALUES (3456, 45);
INSERT INTO StudenciA_egzamin (nr_indeksu, pkt_egz) VALUES (4567, 0);

MERGE INTO StudenciA s
USING (SELECT nr_indeksu, pkt_egz FROM StudenciA_egzamin) e
ON (s.nr_indeksu = e.nr_indeksu)
WHEN MATCHED THEN
  UPDATE SET s.pkt_egz = e.pkt_egz,
             s.zal_egz = CASE WHEN e.pkt_egz > 50 THEN 1 ELSE 0 END;
--2

CREATE TABLE TowaryKC (
    IdTow INT PRIMARY KEY,
    Nazwa VARCHAR2(255),
    Cena NUMBER(10, 2)
);

CREATE TABLE TowaryUaktualnioneKC (
    IdTow INT PRIMARY KEY,
    Nazwa VARCHAR2(255),
    Cena NUMBER(10, 2)
);

INSERT INTO TowaryKC (IdTow, Nazwa, Cena)
VALUES
(1, 'Laptop', 3000.00);

INSERT INTO TowaryKC (IdTow, Nazwa, Cena)
VALUES
(2, 'Smartphone', 1500.00);
```

```
INSERT INTO TowaryKC (IdTow, Nazwa, Cena)
VALUES
(3, 'Tablet', 1000.00);

INSERT INTO TowaryUaktualnioneKC (IdTow, Nazwa, Cena)
VALUES
(1, 'Laptop', 3300.00);

INSERT INTO TowaryUaktualnioneKC (IdTow, Nazwa, Cena)
VALUES
(2, 'Smartphone', 1650.00);

INSERT INTO TowaryUaktualnioneKC (IdTow, Nazwa, Cena)
VALUES
(4, 'Smartwatch', 500.00);

Merge into TowaryKC
using TowaryUaktualnioneKC
On ( TowaryKC.idtow = TowaryUaktualnioneKC.idtow)
when matched then
    Update Set TowaryKc.Nazwa = TowaryUaktualnioneKC.Nazwa, TowaryKc.Cena =
TowaryUaktualnioneKC.Cena
when not matched then
    INSERT (IdTow, Nazwa, Cena)
    VALUES (TowaryUaktualnioneKC.IdTow, TowaryUaktualnioneKC.Nazwa,
TowaryUaktualnioneKC.Cena);

--zad 3
CREATE TABLE GazowniaKC (
    id_klienta INT PRIMARY KEY,
    stan_licznika_poprz NUMBER(10, 2),
    stan_licznika_akt NUMBER(10, 2)
);

CREATE TABLE Odczyt_gazKC (
    id_klienta INT PRIMARY KEY,
    stan_licznika NUMBER(10, 2)
);

INSERT INTO GazowniaKC (id_klienta, stan_licznika_poprz, stan_licznika_akt) VALUES
(1, 100.00, 110.00);
INSERT INTO GazowniaKC (id_klienta, stan_licznika_poprz, stan_licznika_akt) VALUES
(2, 200.00, 220.00);
INSERT INTO GazowniaKC (id_klienta, stan_licznika_poprz, stan_licznika_akt) VALUES
(3, 150.00, 165.00);
INSERT INTO GazowniaKC (id_klienta, stan_licznika_poprz, stan_licznika_akt) VALUES
(4, 300.00, 330.00);
INSERT INTO GazowniaKC (id_klienta, stan_licznika_poprz, stan_licznika_akt) VALUES
(5, 250.00, 275.00);
INSERT INTO GazowniaKC (id_klienta, stan_licznika_poprz, stan_licznika_akt) VALUES
(6, 400.00, 440.00);
INSERT INTO GazowniaKC (id_klienta, stan_licznika_poprz, stan_licznika_akt) VALUES
(7, 500.00, 550.00);
INSERT INTO GazowniaKC (id_klienta, stan_licznika_poprz, stan_licznika_akt) VALUES
(8, 600.00, 660.00);
INSERT INTO GazowniaKC (id_klienta, stan_licznika_poprz, stan_licznika_akt) VALUES
(9, 700.00, 770.00);
INSERT INTO GazowniaKC (id_klienta, stan_licznika_poprz, stan_licznika_akt) VALUES
(10, 800.00, 880.00);
```

```
INSERT INTO Odczyt_gazKC (id_klienta, stan_licznika)
SELECT id_klienta, stan_licznika_akt * 1.10
FROM GazowniaKC;
INSERT INTO odczyt_gazKC (id_klienta, stan_licznika) VALUES (11, 700.00);
INSERT INTO odczyt_gazKC (id_klienta, stan_licznika) VALUES (12, 800.00);

Merge into GazowniaKC
using Odczyt_gazKC
on( GazowniaKC.id_klienta = odczyt_gazKC.id_klienta)
when matched then
    Update set GazowniaKC.stan_licznika_poprz = GazowniaKC.stan_licznika_akt,
GazowniaKC.stan_licznika_akt = Odczyt_gazKC.stan_licznika
    WHERE GazowniaKC.stan_licznika_akt != Odczyt_gazKC.stan_licznika
when not matched then
    INSERT (Id_klienta, stan_licznika_poprz, stan_licznika_akt)
    VALUES (Odczyt_gazKC.id_klienta, Odczyt_gazKC.stan_licznika,
Odczyt_gazKC.stan_licznika);

--4
--tworzenie tabeli Dluznicy_spoldzielni
CREATE TABLE Dluznicy_spoldzielni(id_klienta NUMBER(3), zaleglosc NUMBER(9,2));

INSERT INTO Dluznicy_spoldzielni VALUES (1, 100.00);
INSERT INTO Dluznicy_spoldzielni VALUES (2, 150.00);
INSERT INTO Dluznicy_spoldzielni VALUES (3, 200.00);
INSERT INTO Dluznicy_spoldzielni VALUES (4, 250.00);
INSERT INTO Dluznicy_spoldzielni VALUES (5, 300.00);
INSERT INTO Dluznicy_spoldzielni VALUES (6, 350.00);
INSERT INTO Dluznicy_spoldzielni VALUES (7, 400.00);
INSERT INTO Dluznicy_spoldzielni VALUES (8, 450.00);
INSERT INTO Dluznicy_spoldzielni VALUES (9, 500.00);
INSERT INTO Dluznicy_spoldzielni VALUES (10, 550.00);

-- tworzenie tabeli Dluznicy_aktualizacja
--kopia struktury bez rekordow
CREATE TABLE Dluznicy_aktualizacja as select * from Dluznicy_spoldzielni where 0=1;


INSERT INTO Dluznicy_aktualizacja (id_klienta, zaleglosc)
SELECT id_klienta, zaleglosc * 1.05
FROM Dluznicy_spoldzielni
WHERE id_klienta NOT IN (8, 9, 10);

--merge do edycji / usuwania element w w Dluznicy_spoldzielni
MERGE INTO Dluznicy_spoldzielni d
USING Dluznicy_aktualizacja a
ON (d.id_klienta = a.id_klienta)
WHEN MATCHED THEN
    UPDATE SET d.zaleglosc = a.zaleglosc
    DELETE WHERE a.zaleglosc >= d.zaleglosc * 1.05
WHEN NOT MATCHED THEN
    INSERT (id_klienta, zaleglosc)
    VALUES (a.id_klienta, a.zaleglosc);
```

```
--1
DROP TABLE TOP_EMPLOYEES;
CREATE TABLE TOP_EMPLOYEES AS SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY,
JOB_ID, DEPARTMENT_ID FROM HR.EMPLOYEES WHERE 0=1;

INSERT INTO TOP_EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, JOB_ID,
DEPARTMENT_ID)
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, JOB_ID, DEPARTMENT_ID
FROM HR.EMPLOYEES
ORDER BY SALARY DESC
FETCH FIRST 10 ROWS WITH TIES;

SELECT * FROM TOP_EMPLOYEES;
--2
DROP TABLE SENIOR_EMPLOYEES;
CREATE TABLE SENIOR_EMPLOYEES AS SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY,
JOB_ID, DEPARTMENT_ID FROM HR.EMPLOYEES WHERE 0=1;
DROP TABLE JUNIOR_EMPLOYEES;
CREATE TABLE JUNIOR_EMPLOYEES AS SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY,
JOB_ID, DEPARTMENT_ID FROM HR.EMPLOYEES WHERE 0=1;

INSERT ALL
    WHEN SALARY > 10000 THEN
        INTO SENIOR_EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, JOB_ID,
DEPARTMENT_ID)
        VALUES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, JOB_ID, DEPARTMENT_ID)
    WHEN SALARY <= 10000 THEN
        INTO JUNIOR_EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, JOB_ID,
DEPARTMENT_ID)
        VALUES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, JOB_ID, DEPARTMENT_ID)
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, JOB_ID, DEPARTMENT_ID
FROM HR.EMPLOYEES;

SELECT * FROM JUNIOR_EMPLOYEES;
SELECT * FROM SENIOR_EMPLOYEES;
--3
DROP TABLE PRACOWNICY_JW;
CREATE TABLE PRACOWNICY_JW AS SELECT * FROM HR.EMPLOYEES;

MERGE INTO PRACOWNICY_JW e
USING (
    SELECT DEPARTMENT_ID, AVG(SALARY) AS AVG_SAL
    FROM PRACOWNICY_JW
    GROUP BY DEPARTMENT_ID
) d
ON (e.DEPARTMENT_ID = d.DEPARTMENT_ID)
WHEN MATCHED THEN
    UPDATE SET e.SALARY = d.AVG_SAL WHERE e.SALARY < d.AVG_SAL;

SELECT *FROM PRACOWNICY_JW;
DESCRIBE PRACOWNICY_JW;

--4
DROP TABLE DEPARTMENT_STATS;
CREATE TABLE DEPARTMENT_STATS (DEPARTMENT_ID NUMBER(4),EMPLOYEE_COUNT NUMBER(3));

INSERT INTO DEPARTMENT_STATS (DEPARTMENT_ID, EMPLOYEE_COUNT)
WITH DeptCount AS (
    SELECT DEPARTMENT_ID, COUNT(*) AS EMP_COUNT
```

```
    FROM HR.EMPLOYEES
    GROUP BY DEPARTMENT_ID
)
SELECT DEPARTMENT_ID, EMP_COUNT FROM DeptCount;

select * from DEPARTMENT_STATS;
--13
CREATE TABLE US_EMPLOYEES AS SELECT * FROM HR.employees WHERE 1=0;

INSERT INTO US_EMPLOYEES
SELECT e.* FROM HR.employees e
                    JOIN HR.departments d ON e.department_id = d.department_id
                    JOIN HR.locations l ON d.location_id = l.location_id
                    JOIN HR.countries c ON l.country_id = c.country_id
WHERE c.country_name = 'United States of America';

--14
CREATE TABLE JOBS_COPY AS SELECT * FROM HR.jobs;

MERGE INTO JOBS_COPY j
USING (
    SELECT job_id, MAX(salary) AS max_salary
    FROM HR.employees
    GROUP BY job_id
) e
ON (j.job_id = e.job_id)
WHEN MATCHED THEN
    UPDATE SET j.max_salary = e.max_salary;
```