Template to numeru zadania:

```
--ZAD 1
SELECT 'zadanie 1' FROM DUAL;
```

SKŁADNIA POLECEŃ KTÓRE WYSTĄPIŁY

DISTINCT

- Usuwa duplikaty w wynikach zapytania.
- **Przykład:** Wybierz unikalne wartości z kolumny `kolumna` w tabeli `tabela`.
  `SELECT DISTINCT kolumna FROM tabela;`

LIKE / ON / IN / BETWEEN

- Służą do wyszukiwania wzorców:
  - **LIKE**: Używane głównie do wzorców tekstowych.
    Przykład: Wyszukaj rekordy, w których wartość kolumny `kolumna` kończy się na „ski”.
    `SELECT * FROM tabela WHERE kolumna LIKE '%ski';`
  - **ON**: Wykorzystywane w złączeniach tabel.
    Przykład: Połącz dwie tabele na podstawie klucza `kolumna`.
    `SELECT * FROM tabela1 JOIN tabela2 ON tabela1.kolumna = tabela2.kolumna;`
  - **IN**: Umożliwia wybór wartości z określonej listy.
    Przykład: Wybierz rekordy, gdzie kolumna `kolumna` ma wartość 1, 2 lub 3.
    `SELECT * FROM tabela WHERE kolumna IN (1, 2, 3);`
  - **BETWEEN**: Sprawdza przedziały liczbowe lub dat.
    Przykład: Wybierz rekordy z wartościami kolumny `kolumna` między 10 a 20.
    `SELECT * FROM tabela WHERE kolumna BETWEEN 10 AND 20;`

Operatory specjalne w `LIKE`

- **%**: Zastępuje dowolną liczbę znaków.
  Przykład: Wyszukaj nazwiska kończące się na „ski”.
  `SELECT * FROM tabela WHERE nazwisko LIKE '%ski';`
- **_**: Zastępuje dokładnie jeden znak.
  Przykład: Wyszukaj nazwiska, gdzie druga litera to „a”.
  `SELECT * FROM tabela WHERE nazwisko LIKE '_a%';`

EXISTS

- Sprawdza, czy wynik zapytania istnieje.
  Przykład: Sprawdź, czy w tabeli `inna_tabela` istnieją rekordy powiązane z tabelą `tabela`.
  ```
  SELECT * FROM tabela WHERE EXISTS (SELECT 1 FROM inna_tabela
  WHERE inna_tabela.kolumna = tabela.kolumna);
  ```

TO_CHAR

- Formatuje dane (np. daty) do postaci tekstowej.
  Przykład: Wyświetl datę zatrudnienia w formacie „DD-MM-YYYY”.
  ```
  SELECT TO_CHAR(zatrudniony, 'DD-MM-YYYY') FROM pracownicy;
  ```

Konkatenacja (`||`)

- Łączy wartości w jeden ciąg znaków.
  Przykład: Połącz nazwisko pracownika z jego numerem zespołu.
  ```
  SELECT nazwisko || ' PRACUJE W ZESPOLE ' || id_zesp FROM
  pracownicy;
  ```

COALESCE

- Zamienia wartość `NULL` na wartość zapasową.
  Przykład: Zastąp wartość `NULL` w kolumnie `kolumna` tekstem „Brak danych”.
  ```
  SELECT COALESCE(kolumna, 'Brak danych') FROM tabela;
  ```

SYSDATE

- Pobiera bieżącą datę i czas.
  Przykład: Wyświetl aktualny czas.
  ```
  SELECT SYSDATE AS aktualny_czas FROM DUAL;
  ```
- `CIEKAWOSTKA DROPSA MOZNA UZYC CURRENT_DATE OD TOMKA`

DATE, TIME, TIMESTAMP

- **DATE:** Definiuje datę w formacie `RRRR-MM-DD`.
  Przykład: Wybierz pracowników zatrudnionych przed 1 stycznia 2000 roku.
  ```
  SELECT * FROM pracownicy WHERE zatrudniony < DATE
  '2000-01-01';
  ```
- **TIME:** Definiuje czas w formacie `HH:MI:SS.MMMM`.
  Przykład: Wybierz wydarzenia między godziną 8:00 a 16:00.
  ```
  SELECT * FROM harmonogram WHERE czas_poczatku BETWEEN TIME
  '08:00:00' AND TIME '16:00:00';
  ```
- **TIMESTAMP:** Definiuje znacznik czasowy w formacie `RRRR-MM-DD HH:MI:SS.MMMM`.
  Przykład: Wybierz logi zarejestrowane po 1 stycznia 2020 roku.
  ```
  SELECT * FROM log WHERE data_wyswietlania > TIMESTAMP
  '2020-01-01 00:00:00';
  ```

EXTRACT

- Wydobywa określoną część z daty, czasu lub interwału.
  Przykład: Wybierz nazwiska pracowników zatrudnionych w 2000 roku.
  ```
  SELECT nazwisko FROM pracownicy WHERE EXTRACT(YEAR FROM
  zatrudniony) = 2000;
  ```

ROUND

- Zaokrągla liczbę do określonej liczby miejsc dziesiętnych.
  Przykład: Zaokrąglij wynagrodzenie podstawowe do dwóch miejsc po przecinku.
  ```
  SELECT ROUND(placa_pod, 2) FROM pracownicy;
  ```

LENGTH

- Zwraca długość ciągu znaków.
  Przykład: Wyświetl długość nazwisk pracowników.
  ```
  SELECT LENGTH(nazwisko) FROM pracownicy;
  ```

SUBSTR

- Wyodrębnia fragment ciągu znaków.
  Przykład: Pobierz pierwsze trzy litery nazwiska pracownika.
  ```
  SELECT SUBSTR(nazwisko, 1, 3) FROM pracownicy;
  ```

CASE-WHEN

- Umożliwia wykonywanie instrukcji warunkowych w zapytaniach.
  Przykład: Oznacz wynagrodzenie jako „WYSOKIE” lub „NISKIE” w zależności od jego wartości.
  ```
  SELECT nazwisko, CASE WHEN placa_pod > 5000 THEN 'WYSOKIE'
  ELSE 'NISKIE' END AS status FROM pracownicy;
  ```

—----------

AVG

- **Działanie:** Oblicza średnią wartość z wybranego zbioru danych.
- **Przykład:** Oblicz średnią wartość wynagrodzenia podstawowego dla zespołu o ID 20.
  ```
  SELECT AVG(placa_pod) FROM pracownicy WHERE id_zesp = 20;
  ```

SUM

- **Działanie:** Oblicza sumę wartości w kolumnie.
- **Przykład:** Oblicz sumę wynagrodzeń podstawowych dla wszystkich pracowników.
  ```
  SELECT SUM(placa_pod) FROM pracownicy;
  ```

COUNT

- **Działanie:** Zlicza liczbę rekordów w kolumnie lub całej tabeli.
- **Przykład 1:** Zlicz wszystkich pracowników.
  ```
  SELECT COUNT(*) FROM pracownicy;
  ```
- **Przykład 2:** Zlicz różne rodzaje etatów w tabeli.
  ```
  SELECT COUNT(DISTINCT etat) FROM pracownicy;
  ```

MAX

- **Działanie:** Zwraca maksymalną wartość w zbiorze danych.
- **Przykład:** Znajdź maksymalne wynagrodzenie podstawowe wśród asystentów.
  ```
  SELECT MAX(placa_pod) FROM pracownicy WHERE etat = 'ASYSTENT';
  ```

MIN

- **Działanie:** Zwraca minimalną wartość w zbiorze danych.
- **Przykład:** Znajdź minimalne wynagrodzenie podstawowe w tabeli pracowników.
  ```
  SELECT MIN(placa_pod) FROM pracownicy;
  ```

GROUP BY

- **Działanie:** Grupuje dane według wartości wyrażenia grupującego.
- **Przykład:** Pokaż sumę wynagrodzeń dla każdego rodzaju etatu.
  ```
  SELECT etat, SUM(placa_pod) FROM pracownicy GROUP BY etat;
  ```

HAVING

- **Działanie:** Filtruje wyniki grupowania (dla funkcji grupowych).
- **Przykład:** Pokaż sumę wynagrodzeń dla etatów, gdzie suma przekracza 3000.
  ```
  SELECT etat, SUM(placa_pod) FROM pracownicy GROUP BY etat HAVING
  SUM(placa_pod) > 3000;
  ```

LISTAGG

- **Działanie:** Łączy wartości w grupie w jeden ciąg znaków (w SQL).
- **Przykład:** Wyświetl listę nazwisk pracowników zgrupowanych według etatu.
  ```
  SELECT etat, LISTAGG(nazwisko, ', ') WITHIN GROUP (ORDER BY nazwisko)
  AS pracownicy FROM pracownicy GROUP BY etat;
  ```

VARIANCE

- **Działanie:** Oblicza wariancję dla zbioru danych.
- **Przykład:** Oblicz wariancję wynagrodzenia podstawowego w tabeli pracowników.
  ```
  SELECT VARIANCE(placa_pod) FROM pracownicy;
  ```

STDDEV

- **Działanie:** Oblicza odchylenie standardowe dla zbioru danych.
- **Przykład:** Oblicz odchylenie standardowe wynagrodzenia podstawowego w tabeli pracowników.
  ```
  SELECT STDDEV(placa_pod) FROM pracownicy;
  ```

INSERT INTO

- **Działanie:** Wstawia nowy rekord do tabeli.
- **Przykład:** Dodanie nowego pracownika z unikalnym identyfikatorem.
  ```
  INSERT INTO pracownicy(id_prac, nazwisko, id_zesp) VALUES ((SELECT
  COALESCE(MAX(id_prac), 0) + 1 FROM pracownicy), 'WOLNY', NULL);
  ```

DELETE

- **Działanie:** Usuwa rekordy z tabeli spełniające warunek.
- **Przykład:** Usunięcie pracownika o nazwisku "WOLNY".
  ```
  DELETE FROM pracownicy WHERE nazwisko = 'WOLNY';
  ```

NVL (Oracle) / COALESCE

- **Działanie:** Zastępuje wartości `NULL` podaną wartością.
- **Przykład 1:** Zastąpienie wartości `NULL` w kolumnie `id_zesp` pustym ciągiem znaków.
  ```
  SELECT NVL(TO_CHAR(zespoly.id_zesp), '') AS id_zesp FROM zespoly;
  ```
- **Przykład 2:** Wyświetlenie wartości lub „brak zespołu” w przypadku `NULL`.
  ```
  SELECT COALESCE(zespoly.nazwa, 'brak zespołu') AS nazwa FROM zespoly;
  ```

UNION i UNION ALL

- **Działanie:** Łączy wyniki dwóch zapytań:
  - **UNION:** Usuwa duplikaty.
  - **UNION ALL:** Zachowuje wszystkie rekordy.
- **Przykład:** Grupowanie pracowników na podstawie wynagrodzenia.
  ```
  SELECT 'Poniżej 480 złotych' AS prog FROM pracownicy WHERE placa_pod
  < 480 UNION ALL SELECT 'Dokładnie 480 złotych' AS prog FROM
  pracownicy WHERE placa_pod = 480;
  ```

INTERSECT

- **Działanie:** Zwraca wspólne rekordy dla dwóch zapytań.
- **Przykład:** Znalezienie wspólnych etatów dla pracowników zatrudnionych w latach 1992 i 1993.
  ```
  SELECT DISTINCT etat FROM pracownicy WHERE EXTRACT(YEAR FROM zatrudniony) = 1992 INTERSECT SELECT DISTINCT etat FROM pracownicy WHERE EXTRACT(YEAR FROM zatrudniony) = 1993;
  ```

CROSS JOIN

- **Działanie:** Tworzy iloczyn kartezjański dwóch tabel.
- **Przykład:** Połączenie wszystkich nazwisk pracowników z nazwami zespołów.
  ```
  SELECT pracownicy.nazwisko, zespoly.nazwa FROM pracownicy CROSS JOIN zespoly;
  ```

NATURAL JOIN

- **Działanie:** Łączy tabele automatycznie na podstawie kolumn o tych samych nazwach.
- **Przykład:** Połączenie pracowników z zespołami.
  ```
  SELECT nazwisko, nazwa FROM pracownicy NATURAL JOIN zespoly;
  ```

INNER JOIN

- **Działanie:** Łączy rekordy z dwóch tabel, które spełniają warunek.
- **Przykład:** Wyświetlenie nazwisk pracowników i nazw ich zespołów.
  ```
  SELECT nazwisko, nazwa FROM pracownicy INNER JOIN zespoly ON pracownicy.id_zesp = zespoly.id_zesp;
  ```

OUTER JOIN (LEFT, RIGHT, FULL)

- **Działanie:** Zwraca rekordy dopasowane i niedopasowane z jednej lub obu tabel:
  - **LEFT JOIN:** Wszystkie rekordy z pierwszej tabeli i dopasowania z drugiej.
  - **RIGHT JOIN:** Wszystkie rekordy z drugiej tabeli i dopasowania z pierwszej.
  - **FULL JOIN:** Wszystkie rekordy z obu tabel.
- **Przykład:** Wyświetlenie zespołów i nazwisk pracowników (uwzględniając zespoły bez pracowników).
  ```
  SELECT zespoly.nazwa, pracownicy.nazwisko FROM zespoly LEFT JOIN pracownicy ON zespoly.id_zesp = pracownicy.id_zesp;
  ```

SELF JOIN

- **Działanie:** Łączy tabelę "samą ze sobą".
- **Przykład:** Wyświetlenie relacji pracownik-szef.
  ```
  SELECT p.nazwisko AS pracownik, s.nazwisko AS szef FROM pracownicy p
  INNER JOIN pracownicy s ON p.id_szefa = s.id_prac;
  ```

USING

- **Działanie:** Łączy tabele na podstawie podanych kolumn.
- **Przykład:** Wyświetlenie nazwisk pracowników i nazw ich zespołów, łącząc tabele przez `id_zesp`.
  ```
  SELECT nazwisko, nazwa FROM pracownicy INNER JOIN zespoly USING
  (id_zesp);
  ```

Operator IN

- **Działanie:** Sprawdza, czy wartość znajduje się w zbiorze wyników podzapytania.
- **Przykład:** Wyświetl nazwiska i wynagrodzenia pracowników należących do zespołu "ADMINISTRACJA".
  ```
  SELECT nazwisko, placa_pod FROM pracownicy WHERE id_zesp IN (SELECT
  id_zesp FROM zespoly WHERE nazwa = 'ADMINISTRACJA');
  ```

Operator ANY / SOME

- **Działanie:** Warunek jest spełniony, jeśli porównanie zwróci wynik prawdziwy dla dowolnej wartości ze zbioru wyników podzapytania.
- **Przykład:** Wyświetl pracowników zarabiających więcej niż dowolny pracownik z zespołu o ID 30.
  ```
  SELECT nazwisko, placa_pod FROM pracownicy WHERE placa_pod > ANY
  (SELECT placa_pod FROM pracownicy WHERE id_zesp = 30);
  ```

Operator ALL

- **Działanie:** Warunek jest spełniony, jeśli porównanie zwróci wynik prawdziwy dla wszystkich wartości ze zbioru wyników podzapytania.
- **Przykład:** Wyświetl pracowników zarabiających więcej niż wszyscy pracownicy z zespołu o ID 30.
  ```
  SELECT nazwisko, placa_pod FROM pracownicy WHERE placa_pod > ALL
  (SELECT placa_pod FROM pracownicy WHERE id_zesp = 30);
  ```

Podzapytanie w klauzuli HAVING

- **Działanie:** Używane do filtrowania grup utworzonych przez `GROUP BY`.
- **Przykład:** Wyświetl zespoły, których średnie wynagrodzenie jest wyższe niż średnia wszystkich pracowników.
  ```
  SELECT nazwa, AVG(placa_pod) AS srednia FROM zespoly JOIN pracownicy USING(id_zesp) GROUP BY nazwa HAVING AVG(placa_pod) > (SELECT AVG(placa_pod) FROM pracownicy);
  ```

Podzapytanie skorelowane

- **Działanie:** Podzapytanie, które odwołuje się do wartości z zapytania głównego.
- **Przykład:** Wyświetl pracowników zarabiających więcej niż średnie wynagrodzenie w ich zespole.
  ```
  SELECT nazwisko, placa_pod FROM pracownicy p WHERE placa_pod > (SELECT AVG(placa_pod) FROM pracownicy WHERE id_zesp = p.id_zesp);
  ```

Podzapytanie w klauzuli FROM

- **Działanie:** Tworzy zbiór danych, który może być użyty w zapytaniu głównym.
- **Przykład:** Wyświetl nazwiska i roczną sumę wynagrodzenia pracowników, którzy zarabiają ponad 15 000 zł rocznie.
  ```
  SELECT nazwisko, suma FROM (SELECT nazwisko, (placa_pod * 12) + COALESCE(placa_dod, 0) AS suma FROM pracownicy) alias WHERE suma > 15000;
  ```

Podzapytanie w klauzuli ORDER BY

- **Działanie:** Wykorzystuje wynik podzapytania do sortowania wyników zapytania głównego.
- **Przykład:** Posortuj pracowników według liczby członków ich zespołu (malejąco).
  ```
  SELECT nazwisko FROM pracownicy ORDER BY (SELECT COUNT(*) FROM pracownicy WHERE id_zesp = pracownicy.id_zesp) DESC;
  ```

Wielopoziomowe zagnieżdżanie zapytań

- **Działanie:** Podzapytania są zagnieżdżane na wielu poziomach, a wykonanie rozpoczyna się od najgłębszego podzapytania.

- **Przykład:** Wyświetl pracowników zarabiających więcej niż maksymalne wynagrodzenie w zespole "ALGORYTMY".
```
SELECT nazwisko, placa_pod FROM pracownicy WHERE placa_pod > (SELECT MAX(placa_pod) FROM pracownicy WHERE id_zesp = (SELECT id_zesp FROM zespoly WHERE nazwa = 'ALGORYTMY'));
```

FETCH FIRST N ROWS ONLY

- **Działanie:** Ogranicza wynik zapytania do pierwszych N rekordów.
- **Przykład:** Wyświetl najwcześniej zatrudnionego profesora.
```
SELECT nazwisko, zatrudniony FROM pracownicy WHERE etat = 'PROFESOR' ORDER BY zatrudniony FETCH FIRST 1 ROWS ONLY;
```

ABS

- **Działanie:** Oblicza wartość bezwzględną wyrażenia.
- **Przykład:** Wyświetl parę pracownik-szef, gdzie różnica w ich wynagrodzeniach jest najmniejsza.
```
SELECT p1.nazwisko AS pracownik, p2.nazwisko AS szef FROM pracownicy p1 JOIN pracownicy p2 ON p1.id_szefa = p2.id_prac WHERE ABS(p1.placa_pod - p2.placa_pod) = (SELECT MIN(ABS(p3.placa_pod - p4.placa_pod)) FROM pracownicy p3 JOIN pracownicy p4 ON p3.id_szefa = p4.id_prac);
```

FETCH FIRST / FETCH NEXT

- **Działanie:** Ogranicza liczbę zwracanych wierszy w zapytaniu.
- **Przykład:** Wyświetl 3 pracowników z najwyższymi wynagrodzeniami.
```
SELECT nazwisko, placa_pod FROM pracownicy ORDER BY placa_pod DESC FETCH FIRST 3 ROWS ONLY;
```

OFFSET

- **Działanie:** Pomija określoną liczbę rekordów od początku zbioru wynikowego.
- **Przykład:** Pomiń 3 pierwsze rekordy, a następnie wyświetl kolejne 5.
```
SELECT nazwisko, placa_pod FROM pracownicy ORDER BY placa_pod DESC OFFSET 3 ROWS FETCH NEXT 5 ROWS ONLY;
```

ROWNUM (Oracle)

- **Działanie:** Numeruje rekordy w zbiorze wynikowym. Numeracja odbywa się przed sortowaniem.

- **Przykład:** Wyświetl numery wierszy wraz z nazwiskami pracowników i ich wynagrodzeniami.
  ```
  SELECT ROWNUM, nazwisko, placa_pod FROM pracownicy ORDER BY placa_pod DESC;
  ```

ROWID (Oracle)

- **Działanie:** Unikalnie identyfikuje rekord w tabeli na podstawie jego lokalizacji w bazie danych.
- **Przykład:** Wyświetl identyfikatory wierszy i nazwiska pracowników o nazwisku "HAPKE".
  ```
  SELECT ROWID, nazwisko FROM pracownicy WHERE nazwisko = 'HAPKE';
  ```

Common Table Expression (CTE)

- **Działanie:** Tworzy tymczasowe zbiory wynikowe, które można używać w zapytaniach.
- **Przykład:** Wyświetl nazwiska i wynagrodzenia pracowników zarabiających więcej niż 1000 zł.
  ```
  WITH prac_zesp AS (SELECT nazwisko, placa_pod FROM pracownicy WHERE placa_pod > 1000) SELECT * FROM prac_zesp;
  ```

ZADANIA

Proste zapytania

Wyświetl całość informacji z relacji ZESPOLY. Wynik posortuj rosnąco wg identyfikatorów zespołów.

SELECT* FROM ZESPOLY ORDER BY ID_ZESP;

Wyświetl całość informacji z relacji PRACOWNICY. Wynik posortuj rosnąco wg identyfikatorów pracowników.

SELECT* FROM PRACOWNICY ORDER BY ID_PRAC;

Wyświetl nazwiska i roczne dochody (dwunastokrotność płacy podstawowej) pracowników. Zmień nazwę kolumny z roczną płacą jak przedstawiono poniżej. Posortuj dane rosnąco wg nazwisk pracowników.

SELECT Nazwisko, PLACA_POD * 12 as Roczna_Placa FROM PRACOWNICY ORDER BY Nazwisko;

Wyświetl nazwiska pracowników, nazwy etatów na których pracują oraz sumaryczne miesięczne dochody pracowników (z uwzględnieniem płac dodatkowych). Zmień nazwę kolumny z zarobkami jak przedstawiono poniżej. Dane posortuj malejąco wg miesięcznych zarobków.

SELECT NAZWISKO, ETAT, PLACA_POD + COALESCE(PLACA_DOD, 0) AS MIESIECZNE_ZAROBKI
FROM PRACOWNICY
ORDER by MIESIECZNE_ZAROBKI DESC;

Wyświetl całość informacji o zespołach sortując wynik rosnąco według nazw zespołów.

SELECT * FROM ZESPOLY ORDER BY NAZWA;

Wyświetl listę etatów (bez duplikatów) na których zatrudnieni są pracownicy Instytutu. Etaty posortuj rosnąco.

SELECT DISTINCT ETAT FROM PRACOWNICY ORDER BY ETAT;

Wyświetl wszystkie informacje o asystentach pracujących w Instytucie. Wynik posortuj wg nazwisk pracowników.

SELECT * FROM PRACOWNICY WHERE ETAT = 'ASYSTENT' ORDER BY NAZWISKO;

Wyświetl poniższe dane o pracownikach zespołów 30 i 40 w kolejności malejących zarobków.

SELECT ID_PRAC, NAZWISKO, ETAT, PLACA_POD, ID_ZESP FROM PRACOWNICY
WHERE ID_ZESP IN (30, 40) ORDER BY PLACA_POD DESC;

Wyświetl dane o pracownikach których płace podstawowe mieszczą się w przedziale 300 do 800 zł. Wynik posortuj rosnąco wg nazwisk pracowników.

SELECT NAZWISKO, ID_ZESP, PLACA_POD FROM PRACOWNICY
WHERE PLACA_POD BETWEEN 300 AND 800 ORDER BY NAZWISKO;

Wyświetl poniższe informacje o pracownikach, których nazwisko kończy się na „SKI”. Wynik posortuj rosnąco wg nazwisk pracowników.

SELECT NAZWISKO, ETAT, ID_ZESP FROM PRACOWNICY
WHERE NAZWISKO LIKE '%SKI' ORDER BY NAZWISKO;

Wyświetl poniższe informacje o tych pracownikach, którzy zarabiają powyżej 1000 złotych i posiadają szefa.

SELECT ID_PRAC, ID_SZEFA, NAZWISKO, PLACA_POD FROM PRACOWNICY
WHERE PLACA_POD > 1000 AND ID_SZEFA IS NOT NULL;

Wyświetl nazwiska i identyfikatory zespołów pracowników zatrudnionych w zespole nr 20, których nazwisko zaczyna się na „M” lub kończy na „SKI”. Wynik posortuj wg nazwisk.

SELECT NAZWISKO, ID_ZESP FROM PRACOWNICY WHERE ID_ZESP = 20 AND
(NAZWISKO LIKE 'M%' OR NAZWISKO LIKE '%SKI') ORDER BY NAZWISKO;

Wyświetl nazwiska, etaty i stawki godzinowe tych pracowników, którzy nie są ani adiunktami ani asystentami ani stażystami i którzy nie zarabiają w przedziale od 400 do 800 złotych. Wyniki uszereguj według stawek godzinowych pracowników (przyjmij 20-dniowy miesiąc pracy i 8- godzinny dzień pracy). Wynik posortuj wg wartości stawek w porządku rosnącym.

SELECT NAZWISKO, ETAT, PLACA_POD / (20 * 8) AS STAWKA FROM PRACOWNICY
WHERE ETAT NOT IN ('ADIUNKT', 'ASYSTENT', 'STAZYSTA') AND PLACA_POD NOT
BETWEEN 400 AND 800 ORDER BY STAWKA;

Wyświetl poniższe informacje o pracownikach, dla których suma płacy podstawowej i dodatkowej jest wyższa niż 1000 złotych. Wyniki uporządkuj według nazw etatów. Jeżeli dwóch pracowników ma ten sam etat, to posortuj ich według nazwisk.

SELECT NAZWISKO, ETAT, PLACA_POD, COALESCE(PLACA_DOD, 0) AS PLACA_DOD
FROM PRACOWNICY

```
WHERE PLACA_POD + COALESCE(PLACA_DOD, 0) > 1000 ORDER BY ETAT,
NAZWISKO;
```

Wyświetl poniższe informacje o profesorach, wyniki uporządkuj według malejących płac (nie zwracaj uwagi na format prezentacji daty).

```
SELECT NAZWISKO,     'PRACUJE OD ' || TO_CHAR(ZATRUDNIONY, 'DD-MM-YYYY') || ' I
ZARABIA ' || PLACA_POD AS PROFESOROWIE FROM PRACOWNICY WHERE ETAT =
'PROFESOR'
ORDER BY PLACA_POD DESC;
```

Zaawansowana selekcja danych

Dla każdego pracownika wygeneruj kod składający się z dwóch pierwszych liter jego etatu i jego numeru identyfikacyjnego.

```
SELECT NAZWISKO, SUBSTR(ETAT,1,2) || ID_PRAC AS KOD
FROM PRACOWNICY;
```

Wydaj wojnę literom „K”, „L”, „M” zamieniając je wszystkie na literę „X” w nazwiskach pracowników.

```
SELECT NAZWISKO, REPLACE(REPLACE(REPLACE(NAZWISKO, 'K', 'X'), 'L',
'X'), 'M', 'X') AS WOJNA_LITEROM
FROM PRACOWNICY;
```

Wyświetl nazwiska pracowników którzy posiadają literę „L” w pierwszej połowie swojego nazwiska.

```
SELECT NAZWISKO
FROM PRACOWNICY
WHERE NAZWISKO LIKE '%L%' AND
LENGTH(NAZWISKO)/2>=INSTR(NAZWISKO,'L');
```

Wyświetl nazwiska i płace pracowników powiększone o 15% i zaokrąglone do liczb całkowitych.

```
SELECT NAZWISKO, ROUND(PLACA_POD*1.15) AS POWIEKSZONA_PLACA
FROM PRACOWNICY;
```

Każdy pracownik odłożył 20% swoich miesięcznych zarobków na 10-letnią lokatę oprocentowaną 10% w skali roku i kapitalizowaną co roku. Wyświetl informację o tym, jaki zysk będzie miał każdy pracownik po zamknięciu lokaty.

```
SELECT NAZWISKO, PLACA_POD , ROUND(PLACA_POD*0.2,2) AS INWESTYCJA,
ROUND(PLACA_POD*0.2 * POWER(1+0.1,10),6) AS KAPITAL,
ROUND(PLACA_POD * 0.2 * POWER(1+0.1,10)-(PLACA_POD*0.2),6) AS ZYSK
FROM PRACOWNICY;
```

Policz, jaki staż miał każdy pracownik 1 stycznia 2000 roku.

```
SELECT NAZWISKO, ETAT,
      FLOOR(MONTHS_BETWEEN(DATE '2000-01-01', ZATRUDNIONY) / 12) AS
STAZ_W_2000
FROM PRACOWNICY
WHERE ZATRUDNIONY < DATE '2000-01-01'
ORDER BY STAZ_W_2000 DESC;
```

Wyświetl poniższe informacje o datach przyjęcia pracowników zespołu 20.

```
SELECT NAZWISKO,
TO_CHAR(ZATRUDNIONY, 'Month, DD YYYY','NLS_DATE_LANGUAGE = POLISH') AS
DATA_ZATRUDNIENIA FROM PRACOWNICY
WHERE ID_ZESP = 20;
```

Sprawdź, jaki mamy dziś dzień tygodnia.

```
SELECT TO_CHAR(SYSDATE, 'Day', 'NLS_DATE_LANGUAGE = POLISH') AS DZIS
FROM DUAL;
```

Przyjmij, że Mielżyńskiego i Strzelecka należą do dzielnicy Stare Miasto, Piotrowo należy do dzielnicy Nowe Miasto a Włodkowica należy do dzielnicy Grunwald. Wyświetl poniższy raport (skorzystaj z wyrażenia CASE).

```
SELECT NAZWA, ADRES,
CASE
   WHEN ADRES LIKE 'PIOTROWO%' THEN 'NOWE MIASTO'
   WHEN ADRES LIKE 'STRZELECKA%' OR ADRES LIKE 'MIELZYNSKIEGO%' THEN
'STARE MIASTO'
   WHEN ADRES LIKE 'WLODKOWICA%' THEN 'GRUNWALD'
   ELSE 'BRAK'
   END AS DZIELNICA
   FROM ZESPOLY;
```

Dla każdego pracownika wyświetl informację o tym, czy jego pensja jest mniejsza niż, równa lub większa niż 480 złotych (skorzystaj z wyrażenia CASE).

```
SELECT NAZWISKO, PLACA_POD,
CASE
   WHEN PLACA_POD < 480 THEN 'PONIŻEJ 480'
   WHEN PLACA_POD = 480 THEN 'DOKŁADNIE 480'
   ELSE 'POWYŻEJ 480'
END AS PRÓG
FROM PRACOWNICY
ORDER BY PLACA_POD DESC;
```

W poniższej tabelce przedstawiono średnie płace w poszczególnych zespołach (nie brano pod uwagę płac dodatkowych).

```
SELECT NAZWISKO, ID_ZESP, PLACA_POD
FROM PRACOWNICY
WHERE (ID_ZESP = 10 AND PLACA_POD >= 1070.10)
  OR (ID_ZESP = 20 AND PLACA_POD >= 616.60)
  OR (ID_ZESP = 30 AND PLACA_POD >= 502.00)
  OR (ID_ZESP = 40 AND PLACA_POD >= 1350.00)
ORDER BY ID_ZESP, PLACA_POD;
```

Funkcje grupowe

Wyświetl najniższą i najwyższą pensję w firmie. Wyświetl informację o różnicy dzielącej najlepiej i najgorzej zarabiających pracowników.

```
SELECT MIN(PLACA_POD) AS MINIMUM, MAX(PLACA_POD) AS MAKSIMUM,
      MAX(PLACA_POD) - MIN(PLACA_POD) AS RÓŻNICA
FROM PRACOWNICY;
```

Wyświetl średnie pensje dla wszystkich etatów. Wyniki uporządkuj wg malejącej średniej pensji.

```
SELECT ETAT, AVG(PLACA_POD) AS SREDNIA
FROM PRACOWNICY
GROUP BY ETAT
ORDER BY SREDNIA DESC;
```

Wyświetl liczbę profesorów zatrudnionych w Instytucie.

```
SELECT COUNT(*) AS PROFESOROWIE
FROM PRACOWNICY
WHERE ETAT = 'PROFESOR';
```

Znajdź sumaryczne miesięczne płace dla każdego zespołu. Nie zapomnij o płacach dodatkowych.

```
SELECT ID_ZESP, SUM(PLACA_POD + COALESCE(PLACA_DOD, 0)) AS
SUMARYCZNE_PLACE
```

```
FROM PRACOWNICY
GROUP BY ID_ZESP;
```

Zmodyfikuj zapytanie z zadania poprzedniego w taki sposób, aby jego wynikiem była sumaryczna miesięczna płaca w zespole, który wypłaca swoim pracownikom najwięcej pieniędzy.

```
SELECT MAX(SUMARYCZNE_PLACE) AS MAKS_SUM_PLACA
FROM (SELECT SUM(PLACA_POD + COALESCE(PLACA_DOD, 0)) AS
SUMARYCZNE_PLACE
    FROM PRACOWNICY
    GROUP BY ID_ZESP);
```

Dla każdego pracownika, który posiada podwładnych, wyświetl pensję najgorzej zarabiającego podwładnego. Wyniki uporządkuj wg malejącej pensji.

```
SELECT ID_SZEFA, MIN(PLACA_POD) AS MINIMALNA
FROM PRACOWNICY
WHERE ID_SZEFA IS NOT NULL
GROUP BY ID_SZEFA
ORDER BY MINIMALNA DESC, ID_SZEFA ASC;
```

Wyświetl numery zespołów wraz z liczbą pracowników w każdym zespole. Wyniki uporządkuj wg malejącej liczby pracowników.

```
SELECT ID_ZESP, COUNT(*) AS ILU_PRACUJE
FROM PRACOWNICY
GROUP BY ID_ZESP
ORDER BY ILU_PRACUJE DESC;
```

Zmodyfikuj zapytanie z zadania poprzedniego, aby wyświetlić numery tylko tych zespołów, które zatrudniają więcej niż 3 pracowników.

```
SELECT ID_ZESP, COUNT(*) AS ILU_PRACUJE
FROM PRACOWNICY
GROUP BY ID_ZESP
HAVING COUNT(*) > 3
ORDER BY ILU_PRACUJE DESC;
```

Sprawdź, czy identyfikatory pracowników są unikalne. Wyświetl zdublowane wartości identyfikatorów.

```
SELECT ID_PRAC, COUNT(*)
FROM PRACOWNICY
GROUP BY ID_PRAC
HAVING COUNT(*) > 1;
```

Wyświetl średnie pensje wypłacane w ramach poszczególnych etatów i liczbę zatrudnionych na danym etacie. Pomiń pracowników zatrudnionych po 1990 roku.

```
SELECT ETAT, AVG(PLACA_POD) AS SREDNIA, COUNT(*) AS LICZBA
FROM PRACOWNICY
WHERE TO_CHAR(ZATRUDNIONY, 'YYYY') <= 1990
GROUP BY ETAT
```

```
ORDER BY ETAT ASC;
```

Zbuduj zapytanie, które wyświetli średnie i maksymalne pensje asystentów i profesorów w poszczególnych zespołach (weź pod uwagę zarówno płace podstawowe jak i dodatkowe). Dokonaj zaokrąglenia pensji do wartości całkowitych.

```
SELECT ID_ZESP, ETAT, ROUND(AVG(PLACA_POD + COALESCE(PLACA_DOD, 0))) AS
SREDNIA,
    ROUND(MAX(PLACA_POD + COALESCE(PLACA_DOD, 0))) AS MAKSYMALNA
FROM PRACOWNICY
WHERE ETAT IN ('ASYSTENT', 'PROFESOR')
GROUP BY ID_ZESP, ETAT
ORDER BY ID_ZESP, ETAT;
```

Zbuduj zapytanie, które wyświetli, ilu pracowników zostało zatrudnionych w poszczególnych latach. Wynik posortuj rosnąco ze względu na rok zatrudnienia.

```
SELECT TO_CHAR(ZATRUDNIONY, 'YYYY') AS ROK, COUNT(*) AS
ILU_PRACOWNIKOW
FROM PRACOWNICY
GROUP BY TO_CHAR(ZATRUDNIONY, 'YYYY')
ORDER BY ROK;
```

Zbuduj zapytanie, które policzy liczbę liter w nazwiskach pracowników i wyświetli liczbę nazwisk z daną liczbą liter. Wynik zapytania posortuj rosnąco wg liczby liter w nazwiskach.

```
SELECT LENGTH(NAZWISKO) AS ILE_LITER, COUNT(*) AS W_ILU_NAZWISKACH
FROM PRACOWNICY
GROUP BY LENGTH(NAZWISKO)
ORDER BY ILE_LITER;
```

Zbuduj zapytanie, które wyliczy, ilu pracowników w swoim nazwisku posiada chociaż jedną literę „a” lub „A”.

```
SELECT COUNT(*) AS ILE_NAZWISK_Z_A
FROM PRACOWNICY
WHERE LOWER(NAZWISKO) LIKE '%a%';
```

Zmień poprzednie zapytanie w taki sposób, aby oprócz kolumny, pokazującej ilu pracowników w swoim nazwisku posiada chociaż jedną literę „a” lub „A”, pojawiła się kolumna pokazująca liczbę pracowników z chociaż jedną literą „e” lub „E” w nazwisku.

```
SELECT COUNT(CASE WHEN LOWER(NAZWISKO) LIKE '%a%' THEN 1 END) AS
ILE_NAZWISK_Z_A,
    COUNT(CASE WHEN LOWER(NAZWISKO) LIKE '%e%' THEN 1 END) AS
ILE_NAZWISK_Z_E
FROM PRACOWNICY;
```

Dla każdego zespołu wyświetl jego identyfikator, sumę płac pracowników w nim zatrudnionych oraz listę pracowników w formie: nazwisko:podstawowa płaca pracownika. Dane pracowników na liście mają zostać oddzielone średnikami.

```
SELECT ID_ZESP, SUM(PLACA_POD) AS SUMA_PLAC,
      LISTAGG(NAZWISKO || ':' || PLACA_POD, ';') WITHIN GROUP (ORDER BY
NAZWISKO) AS PRACOWNICY
FROM PRACOWNICY
GROUP BY ID_ZESP
ORDER BY ID_ZESP;
```

Podstawy połączeń

Wyświetl nazwiska, etaty, numery zespołów i nazwy zespołów wszystkich pracowników. Wynik uporządkuj wg nazwisk pracowników.

```
SELECT PRACOWNICY.NAZWISKO, PRACOWNICY.ETAT, PRACOWNICY.ID_ZESP,
ZESPOLY.NAZWA
FROM PRACOWNICY
JOIN ZESPOLY ON PRACOWNICY.ID_ZESP = ZESPOLY.ID_ZESP
ORDER BY PRACOWNICY.NAZWISKO;
```

Ogranicz wynik poprzedniego zapytania do tych pracowników, którzy pracują w zespołach zlokalizowanych przy ul. Piotrowo 3a.

```
SELECT PRACOWNICY.NAZWISKO, PRACOWNICY.ETAT, PRACOWNICY.ID_ZESP,
ZESPOLY.NAZWA
FROM PRACOWNICY
JOIN ZESPOLY ON PRACOWNICY.ID_ZESP = ZESPOLY.ID_ZESP
WHERE ZESPOLY.ADRES = 'PIOTROWO 3A'
ORDER BY PRACOWNICY.NAZWISKO;
```

Znajdź nazwiska, etaty i pensje podstawowe pracowników. Wyświetl również minimalne i maksymalne pensje dla etatów, na których pracują pracownicy (użyj tabeli Etaty). Wynik posortuj wg nazw etatów i nazwisk pracowników.

```
SELECT PRACOWNICY.NAZWISKO, PRACOWNICY.ETAT, PRACOWNICY.PLACA_POD,
ETATY.PLACA_MIN, ETATY.PLACA_MAX
FROM PRACOWNICY
JOIN ETATY ON PRACOWNICY.ETAT = ETATY.NAZWA
ORDER BY PRACOWNICY.ETAT, PRACOWNICY.NAZWISKO;
```

Zmień poprzednie zapytanie w taki sposób, aby w zbiorze wynikowym pojawiła się kolumna czy_pensja_ok. Ma w niej pojawić wartość „OK” jeśli płaca podstawowa pracownika zawiera się w przedziale wyznaczonym przez płace: minimalną i maksymalną dla etatu, na którym pracownik pracuje lub wartość „NIE” w przeciwnym wypadku.

```
SELECT PRACOWNICY.NAZWISKO, PRACOWNICY.ETAT, PRACOWNICY.PLACA_POD,
ETATY.PLACA_MIN, ETATY.PLACA_MAX,
      CASE
         WHEN PRACOWNICY.PLACA_POD BETWEEN ETATY.PLACA_MIN AND
ETATY.PLACA_MAX THEN 'OK'
         ELSE 'NIE'
```

```
    END AS CZY_PENSJA_OK
FROM PRACOWNICY
JOIN ETATY ON PRACOWNICY.ETAT = ETATY.NAZWA
ORDER BY PRACOWNICY.ETAT, PRACOWNICY.NAZWISKO;
```

Wykorzystaj dodaną w p. 4. kolumnę aby znaleźć pracowników, którzy zarabiają więcej lub mniej niż to jest przewidziane dla etatów, na których pracują.

```
SELECT PRACOWNICY.NAZWISKO, PRACOWNICY.ETAT, PRACOWNICY.PLACA_POD,
ETATY.PLACA_MIN, ETATY.PLACA_MAX
FROM PRACOWNICY
JOIN ETATY ON PRACOWNICY.ETAT = ETATY.NAZWA
WHERE PRACOWNICY.PLACA_POD NOT BETWEEN ETATY.PLACA_MIN AND
ETATY.PLACA_MAX
ORDER BY PRACOWNICY.NAZWISKO;
```

Dla każdego pracownika wyświetl jego nazwisko, płacę podstawową, etat, kategorię płacową i widełki płacowe, w jakich mieści się pensja pracownika. Kategoria płacowa to nazwa etatu (z tabeli Etaty), do którego pasuje płaca podstawowa pracownika (zawiera się w przedziale płac dla etatu). Wynik posortuj wg nazwisk i kategorii płacowych pracowników.

```
SELECT PRACOWNICY.NAZWISKO, PRACOWNICY.PLACA_POD, PRACOWNICY.ETAT,
ETATY.NAZWA AS KAT_PLAC, ETATY.PLACA_MIN, ETATY.PLACA_MAX
FROM PRACOWNICY
JOIN ETATY ON PRACOWNICY.PLACA_POD BETWEEN ETATY.PLACA_MIN AND
ETATY.PLACA_MAX
ORDER BY PRACOWNICY.NAZWISKO, KAT_PLAC;
```

Powyższy zbiór ogranicz do tych pracowników, których rzeczywiste zarobki odpowiadają widełkom płacowym przewidzianym dla sekretarek. Wynik posortuj wg nazwisk pracowników.

```
SELECT PRACOWNICY.NAZWISKO, PRACOWNICY.PLACA_POD, PRACOWNICY.ETAT,
ETATY.NAZWA AS KAT_PLAC, ETATY.PLACA_MIN, ETATY.PLACA_MAX
FROM PRACOWNICY
JOIN ETATY ON PRACOWNICY.PLACA_POD BETWEEN ETATY.PLACA_MIN AND
ETATY.PLACA_MAX
WHERE ETATY.NAZWA = 'SEKRETARKA'
ORDER BY PRACOWNICY.NAZWISKO;
```

Wyświetl nazwiska i numery pracowników wraz z numerami i nazwiskami ich szefów. Wynik posortuj wg nazwisk pracowników. W zbiorze wynikowym mają się pojawić tylko ci pracownicy, którzy mają szefów.

```
SELECT P1.NAZWISKO AS PRACOWNIK, P1.ID_PRAC, P2.NAZWISKO AS SZEF,
P1.ID_SZEFA
FROM PRACOWNICY P1
JOIN PRACOWNICY P2 ON P1.ID_SZEFA = P2.ID_PRAC
ORDER BY P1.NAZWISKO;
```

Wyświetl nazwiska i daty zatrudnienia pracowników, którzy zostali zatrudnieni nie później niż 10 lat po swoich przełożonych. Wynik uporządkuj wg dat zatrudnienia i nazwisk pracowników.

```
SELECT P1.NAZWISKO AS PRACOWNIK,
    TO_CHAR(P1.ZATRUDNIONY, 'YYYY.MM.DD') AS PRAC_ZATRUDNIONY,
    P2.NAZWISKO AS SZEF,
    TO_CHAR(P2.ZATRUDNIONY, 'YYYY.MM.DD') AS SZEF_ZATRUDNIONY,
    TRUNC(MONTHS_BETWEEN(P1.ZATRUDNIONY, P2.ZATRUDNIONY) / 12) AS LATA
FROM PRACOWNICY P1
JOIN PRACOWNICY P2 ON P1.ID_SZEFA = P2.ID_PRAC
WHERE TRUNC(MONTHS_BETWEEN(P1.ZATRUDNIONY, P2.ZATRUDNIONY) / 12) <=
10
ORDER BY P1.ZATRUDNIONY, P1.NAZWISKO;
```

Dla każdego zespołu, który zatrudnia pracowników, wyświetl liczbę zatrudnionych w nim pracowników i ich średnią płacę podstawową. Wynik posortuj wg nazw zespołów.

```
SELECT ZESPOLY.NAZWA, COUNT(PRACOWNICY.ID_PRAC) AS LICZBA,
AVG(PRACOWNICY.PLACA_POD) AS SREDNIA_PLACA
FROM ZESPOLY
JOIN PRACOWNICY ON ZESPOLY.ID_ZESP = PRACOWNICY.ID_ZESP
GROUP BY ZESPOLY.NAZWA
ORDER BY ZESPOLY.NAZWA;
```

Poetykietuj zespoły w zależności od liczby zatrudnionych pracowników. Jeśli zespół zatrudnia do dwóch pracowników, przydziel mu etykietę “mały”. Zespołom zatrudniającym od 3 do 6 pracowników, przydziel etykietę “średni”. Jeśli departament zatrudnia 7 i więcej pracowników, powinien otrzymać etykietę “duży”. Pomiń departamenty bez pracowników.

```
SELECT ZESPOLY.NAZWA,
    CASE
       WHEN COUNT(PRACOWNICY.ID_PRAC) <= 2 THEN 'mały'
       WHEN COUNT(PRACOWNICY.ID_PRAC) BETWEEN 3 AND 6 THEN 'średni'
       ELSE 'duży'
    END AS ETYKIETA
FROM ZESPOLY
JOIN PRACOWNICY ON ZESPOLY.ID_ZESP = PRACOWNICY.ID_ZESP
GROUP BY ZESPOLY.NAZWA
HAVING COUNT(PRACOWNICY.ID_PRAC) > 0;
```

Połączenia zaawansowane. Operatory zbiorowe

Wstaw „na chwilę” pracownika nieprzypisanego do żadnego zespołu. Osiągniesz to poniższym poleceniem:

```
INSERT INTO pracownicy(id_prac, nazwisko)
VALUES ((SELECT max(id_prac) + 1 FROM pracownicy), 'WOLNY');
```

Następnie skonstruuj zapytanie, które wyświetli nazwiska, numery zespołów i nazwy zespołów wszystkich pracowników. W zbiorze wynikowym mają pojawić się również pracownicy, którzy nie należą do żadnego zespołu. Wynik uporządkuj wg nazwisk pracowników.

```
SELECT p.nazwisko, z.id_zesp, z.nazwa
FROM pracownicy p
LEFT JOIN zespoly z ON p.id_zesp = z.id_zesp
ORDER BY p.nazwisko;
```

Tym razem wyświetl nazwy wszystkich zespołów. Jeśli w zespole pracują pracownicy, wyświetl ich nazwiska. Dla zespołów, które nie mają pracowników, wyświetl tekst „brak pracowników”. Uporządkuj wynik według nazw zespołów i nazwisk pracowników.

```
SELECT z.nazwa, z.id_zesp,
    COALESCE(p.nazwisko, 'brak pracowników') AS pracownik
FROM zespoly z
LEFT JOIN pracownicy p ON z.id_zesp = p.id_zesp
ORDER BY z.nazwa, p.nazwisko;
```

Połącz wyniki dwóch poprzednich zapytań w jeden wynik. Dla pracowników pracujących w zespołach wyświetl nazwisko pracownika i nazwę zespołu. Dla pracowników bez zespołów wyświetl w miejscu nazwy zespołu tekst „brak zespołu”. Dla zespołów, które nie mają pracowników, wyświetl tekst „brak pracowników”. Uporządkuj wynik według nazw zespołów i nazwisk pracowników. Nazwiska pracowników bez zespołów powinny znaleźć się na końcu raportu, posortowane w porządku rosnącym.

```
SELECT z.nazwa AS zespol, COALESCE(p.nazwisko, 'brak pracownikow') AS pracownik
FROM zespoly z
LEFT JOIN pracownicy p ON z.id_zesp = p.id_zesp
UNION ALL
SELECT 'brak zespołu' AS zespol, nazwisko AS pracownik
FROM pracownicy
WHERE id_zesp IS NULL
ORDER BY zespol, pracownik;
```

Usuń dodanego w punkcie 1. pracownika poniższym poleceniem:

```
DELETE FROM pracownicy
WHERE nazwisko = 'WOLNY';
```

Dla każdego zespołu znajdź liczbę pracowników, których zatrudnia oraz sumę ich płac. W zbiorze wynikowym uwzględnij również zespoły bez pracowników.

```
SELECT z.nazwa AS zespol,
    COUNT(p.id_prac) AS liczba,
```

```
    SUM(p.placa_pod) AS suma_plac
FROM zespoly z
LEFT JOIN pracownicy p ON z.id_zesp = p.id_zesp
GROUP BY z.nazwa
ORDER BY z.nazwa ASC;
```

Wyświetl nazwy zespołów, które nie zatrudniają pracowników. Wynik posortuj wg nazw zespołów.

```
SELECT nazwa
FROM zespoly z
LEFT JOIN pracownicy p ON z.id_zesp = p.id_zesp
WHERE p.id_prac IS NULL
ORDER BY z.nazwa;
```

Wyświetl nazwiska i numery pracowników wraz z numerami i nazwiskami ich szefów. Wynik posortuj wg nazwisk pracowników. W zbiorze wynikowym mają się pojawić również ci pracownicy, którzy nie mają szefów.

```
SELECT p1.nazwisko AS pracownik, p1.id_prac,
    p2.nazwisko AS szef, p1.id_szefa
FROM pracownicy p1
LEFT JOIN pracownicy p2 ON p1.id_szefa = p2.id_prac
ORDER BY p1.nazwisko;
```

Dla każdego pracownika wyświetl liczbę jego bezpośrednich podwładnych.

```
SELECT p.nazwisko AS pracownik,
    COUNT(sub.id_prac) AS liczba_podwladnych
FROM pracownicy p
LEFT JOIN pracownicy sub ON p.id_prac = sub.id_szefa
GROUP BY p.nazwisko
ORDER BY p.nazwisko ASC;
```

Wyświetl następujące informacje o każdym pracowniku: nazwisko, etat, płaca podstawowa, nazwa zespołu, do którego należy oraz nazwisko szefa. Wynik uporządkuj wg nazwisk pracowników. Weź pod uwagę, że pracownik może nie mieć szefa i może nie być zatrudniony w żadnym zespole.

```
SELECT p.nazwisko, p.etat, p.placa_pod,
    z.nazwa AS zespol,
    s.nazwisko AS szef
FROM pracownicy p
LEFT JOIN zespoly z ON p.id_zesp = z.id_zesp
LEFT JOIN pracownicy s ON p.id_szefa = s.id_prac
ORDER BY p.nazwisko;
```

Wygeneruj iloczyn kartezjański relacji Pracownicy i Zespoly. W zbiorze wynikowym umieść jedynie wartości kolumn nazwisko i nazwa.

```
SELECT p.nazwisko, z.nazwa
FROM pracownicy p, zespoly z
ORDER BY p.nazwisko ASC;
```

Policz, ile rekordów będzie zawierał iloczyn kartezjański trzech relacji: Etaty, Pracownicy i Zespoły.

```
SELECT COUNT(*)
FROM etaty e, pracownicy p, zespoly z;
```

Wyświetl nazwy etatów, na które przyjęto pracowników zarówno w 1992 jak i 1993 roku. Wynik posortuj wg nazw etatów.

```
SELECT DISTINCT etat
FROM pracownicy
WHERE EXTRACT(YEAR FROM pracownicy.ZATRUDNIONY) = 1992
INTERSECT
SELECT DISTINCT etat
FROM pracownicy
WHERE EXTRACT(YEAR FROM pracownicy.ZATRUDNIONY) = 1993
ORDER BY etat;
```

Wyświetl numer zespołu, który nie zatrudnia żadnych pracowników.

```
SELECT z.id_zesp
FROM zespoly z
LEFT JOIN pracownicy p ON z.id_zesp = p.id_zesp
WHERE p.id_prac IS NULL;
```

Zmień powyższe zapytanie w taki sposób, aby oprócz numeru poznać również nazwę zespołu bez pracowników.

```
SELECT z.id_zesp, z.nazwa
FROM zespoly z
LEFT JOIN pracownicy p ON z.id_zesp = p.id_zesp
WHERE p.id_prac IS NULL;
```

Wyświetl poniższy raport. Nie używaj wyrażenia CASE.

```
SELECT nazwisko, placa_pod,
      DECODE(SIGN(placa_pod - 480), -1, 'Poniżej 480 złotych',
            0, 'Dokladnie 480 złotych',
            1, 'Powyżej 480 złotych') AS prog
FROM pracownicy
ORDER BY pracownicy.PLACA_POD ASC;
```

Podstawy podzapytań

Wyświetl nazwiska i etaty pracowników pracujących w tym samym zespole co pracownik o nazwisku Brzeziński. Wynik uporządkuj wg nazwisk pracowników.

```
SELECT p.nazwisko, p.etat, p.id_zesp
FROM pracownicy p
WHERE p.id_zesp IN (SELECT id_zesp FROM pracownicy WHERE nazwisko =
'BRZEZINSKI')
ORDER BY p.nazwisko;
```

Zmodyfikuj treść poprzedniego zapytania w taki sposób, aby zamiast identyfikatora zespołu pojawiła się jego nazwa.

```
SELECT p.nazwisko, p.etat, z.nazwa AS nazwa_zespolu
FROM pracownicy p
JOIN zespoly z ON p.id_zesp = z.id_zesp
WHERE p.id_zesp = (SELECT id_zesp FROM pracownicy WHERE nazwisko =
'BRZEZINSKI')
ORDER BY p.nazwisko;
```

Wstaw „na chwilę” asystenta nieprzypisanego do żadnego zespołu, zatrudnionego 1 lipca 1968 r. Osiągniesz to poniższym poleceniem:

```
INSERT INTO pracownicy (id_prac, nazwisko, etat, zatrudniony)
VALUES ((SELECT MAX(id_prac) + 1 FROM pracownicy),
      'WOLNY', 'ASYSTENT', DATE '1968-07-01');
```

Następnie wyświetl nazwisko, etat i datę zatrudnienia najdłużej zatrudnionego profesora.

```
SELECT nazwisko, etat, TO_CHAR(zatrudniony, 'YYYY/MM/DD') AS zatrudniony
FROM pracownicy
WHERE etat = 'PROFESOR'
AND zatrudniony <= ALL (
SELECT zatrudniony
FROM pracownicy
WHERE etat = 'PROFESOR'
);
```

Wyświetl najkrócej pracujących pracowników każdego zespołu. Uszereguj wyniki zgodnie z kolejnością zatrudnienia.

```
SELECT p.nazwisko, TO_CHAR(p.zatrudniony, 'YYYY/MM/DD') AS zatrudniony, p.id_zesp
FROM pracownicy p
WHERE (p.zatrudniony, p.id_zesp) IN (
   SELECT MAX(zatrudniony), id_zesp
   FROM pracownicy
   GROUP BY id_zesp
)
ORDER BY p.zatrudniony;
```

Wyświetl informacje o zespołach, które nie zatrudniają pracowników.

```
SELECT z.id_zesp, z.nazwa, z.adres
FROM zespoly z
```

```
WHERE NOT EXISTS (
   SELECT 1 FROM pracownicy p WHERE p.id_zesp = z.id_zesp
);
```

Usuń dodanego w punkcie 3. pracownika poniższym poleceniem:

```
DELETE FROM pracownicy
WHERE nazwisko = 'WOLNY';
```

Wyświetl nazwiska tych profesorów, którzy wśród swoich podwładnych nie mają żadnych stażystów.

```
SELECT DISTINCT szef.nazwisko
FROM pracownicy szef
WHERE szef.etat = 'PROFESOR'
  AND NOT EXISTS (
      SELECT 1
      FROM pracownicy p
      WHERE p.id_szefa = szef.id_prac
        AND p.etat = 'STAZYSTA'
  );
```

Wyświetl numer zespołu wypłacającego miesięcznie swoim pracownikom najwięcej pieniędzy.

```
SELECT id_zesp, suma_plac
FROM (
   SELECT p.id_zesp, SUM(p.placa_pod) AS suma_plac
   FROM pracownicy p
   GROUP BY p.id_zesp
)
WHERE suma_plac = (
   SELECT MAX(SUM(p.placa_pod))
   FROM pracownicy p
   GROUP BY p.id_zesp
);
```

Zmodyfikuj poprzednie zapytanie w taki sposób, aby zamiast numeru zespołu wyświetlona została jego nazwa.

```
SELECT nazwa, suma_plac
FROM (
   SELECT z.nazwa, SUM(p.placa_pod) AS suma_plac
   FROM pracownicy p
   JOIN zespoly z ON p.id_zesp = z.id_zesp
   GROUP BY z.nazwa
)
WHERE suma_plac = (
   SELECT MAX(SUM(p.placa_pod))
   FROM pracownicy p
   JOIN zespoly z ON p.id_zesp = z.id_zesp
   GROUP BY z.nazwa
```

);

Znajdź zespoły zatrudniające więcej pracowników niż zespół ADMINISTRACJA. Wynik posortuj wg nazw zespołów.

```
SELECT z.nazwa, COUNT(p.id_prac) AS ilu_pracownikow
FROM pracownicy p
JOIN zespoly z ON p.id_zesp = z.id_zesp
GROUP BY z.nazwa
HAVING COUNT(p.id_prac) > (
   SELECT COUNT(p.id_prac)
   FROM pracownicy p
   JOIN zespoly z ON p.id_zesp = z.id_zesp
   WHERE z.nazwa = 'ADMINISTRACJA'
)
ORDER BY z.nazwa;
```

Znajdź etat (etaty), który jest najliczniej reprezentowany w zbiorze pracowników.

```
SELECT etat
FROM pracownicy
GROUP BY etat
HAVING COUNT(*) = (
   SELECT MAX(cnt)
   FROM (SELECT COUNT(*) AS cnt FROM pracownicy GROUP BY etat)
)
ORDER BY etat ASC;
```

Uzupełnij wynik poprzedniego zapytania o listę nazwisk pracowników na znalezionych etatach.

```
WITH popularne AS (
   SELECT etat
   FROM pracownicy
   GROUP BY etat
   HAVING COUNT(*) = (
      SELECT MAX(cnt)
      FROM (SELECT COUNT(*) AS cnt FROM pracownicy GROUP BY etat)
   )
)
SELECT e.etat, LISTAGG(p.nazwisko, ',') WITHIN GROUP (ORDER BY p.nazwisko) AS
pracownicy
FROM pracownicy p
JOIN popularne e ON p.etat = e.etat
GROUP BY e.etat;
```

Znajdź parę: pracownik – szef, dla której różnica między płacą pracownika a płacą jego szefa jest najniższa.

```
SELECT pracownik, szef
FROM (
   SELECT
```

```
        e.nazwisko AS pracownik,
        szef.nazwisko AS szef,
        ABS((e.placa_pod + NVL(e.placa_dod, 0)) - (szef.placa_pod + NVL(szef.placa_dod,
0))) AS roznica
    FROM pracownicy e
    JOIN pracownicy szef ON e.id_szefa = szef.id_prac
)
WHERE roznica = (
    SELECT MIN(
        ABS((e1.placa_pod + NVL(e1.placa_dod, 0)) - (s1.placa_pod + NVL(s1.placa_dod, 0)))
    )
    FROM pracownicy e1
    JOIN pracownicy s1 ON e1.id_szefa = s1.id_prac
);
```

Podzapytania zaawansowane

Wyświetl informacje o zespołach, które nie zatrudniają pracowników. Rozwiązanie powinno korzystać z podzapytania skorelowanego.

```
SELECT ID_ZESP, NAZWA, ADRES
FROM ZESPOLY Z
WHERE NOT EXISTS (
    SELECT 1
    FROM PRACOWNICY P
    WHERE P.ID_ZESP = Z.ID_ZESP
);
```

Wyświetl nazwiska, płace podstawowe i nazwy etatów pracowników zarabiających więcej niż średnia pensja dla ich etatu. Wynik uporządkuj malejąco wg wartości płac podstawowych. Czy da się ten problem rozwiązać podzapytaniem zwykłym (bez korelacji)?

```
SELECT NAZWISKO, PLACA_POD, ETAT
FROM PRACOWNICY P
WHERE PLACA_POD > (
    SELECT AVG(PLACA_POD)
    FROM PRACOWNICY
    WHERE ETAT = P.ETAT
)
ORDER BY PLACA_POD DESC;
```

Wyświetl nazwiska i pensje pracowników którzy zarabiają co najmniej 75% pensji swojego szefa. Wynik uporządkuj wg nazwisk.

```
SELECT P.NAZWISKO, P.PLACA_POD
FROM PRACOWNICY P
WHERE P.PLACA_POD >= 0.75 * (
   SELECT SZ.PLACA_POD
   FROM PRACOWNICY SZ
   WHERE SZ.ID_PRAC = P.ID_SZEFA
)
ORDER BY P.NAZWISKO;
```

Wyświetl nazwiska tych profesorów, którzy wśród swoich podwładnych nie mają żadnych stażystów. Użyj podzapytania skorelowanego.

```
SELECT NAZWISKO
FROM PRACOWNICY P
WHERE ETAT = 'PROFESOR'
  AND NOT EXISTS (
   SELECT 1
   FROM PRACOWNICY P2
   WHERE P2.ID_SZEFA = P.ID_PRAC AND P2.ETAT = 'STAZYSTA'
);
```

Wyświetl zespół z najwyższą sumaryczną pensją wśród zespołów. Użyj tylko podzapytań w klauzuli FROM: pierwsze ma znaleźć maksymalną sumaryczną płacę wśród zespołów (pojedyncza wartość), drugie wyliczy sumę płac w każdym zespole (zbiór rekordów, struktura zbioru: identyfikator zespołu, suma płac w zespole). Zapytanie główne ma wykonać dwa połączenia: pierwsze połączy zbiory wynikowe obu podzapytań do znalezienia szukanego zespołu, drugie, z tabelą Zespoly, uzupełni zbiór wynikowy o nazwę zespołu.

```
SELECT Z.NAZWA, MAX_PLAC.MAKS_SUMA_PLAC
FROM (
   SELECT ID_ZESP, SUM(PLACA_POD) AS SUMA_PLAC
   FROM PRACOWNICY
   GROUP BY ID_ZESP
) ZESP_SUM,
(
   SELECT MAX(SUMA_PLAC) AS MAKS_SUMA_PLAC
   FROM (
      SELECT SUM(PLACA_POD) AS SUMA_PLAC
      FROM PRACOWNICY
      GROUP BY ID_ZESP
   )
) MAX_PLAC,
ZESPOLY Z
WHERE ZESP_SUM.SUMA_PLAC = MAX_PLAC.MAKS_SUMA_PLAC
  AND Z.ID_ZESP = ZESP_SUM.ID_ZESP;
```

Wyświetl nazwiska i pensje trzech najlepiej zarabiających pracowników. Uporządkuj ich zgodnie z wartościami pensji w porządku malejącym. Zastosuj podzapytanie skorelowane.

```
SELECT NAZWISKO, PLACA_POD
FROM PRACOWNICY P1
WHERE 3 > (
   SELECT COUNT(*)
   FROM PRACOWNICY P2
   WHERE P2.PLACA_POD > P1.PLACA_POD
)
ORDER BY PLACA_POD DESC;
```

Dla każdego pracownika podaj jego nazwisko, płacę podstawową oraz różnicę między jego płacą podstawową a średnią płacą podstawową w zespole, do którego pracownik należy. Zaproponuj dwa rozwiązania, wykorzystujące: (1) podzapytanie w klauzuli SELECT (2) podzapytanie w klauzuli FROM.

```
SELECT P.NAZWISKO, P.PLACA_POD,
      P.PLACA_POD - (
         SELECT AVG(PLACA_POD)
         FROM PRACOWNICY
         WHERE ID_ZESP = P.ID_ZESP
      ) AS ROZNICA
FROM PRACOWNICY P
ORDER BY P.NAZWISKO;
```

```
SELECT P.NAZWISKO, P.PLACA_POD, P.PLACA_POD - Z.AVG_PLACA AS ROZNICA
FROM PRACOWNICY P
JOIN (
   SELECT ID_ZESP, AVG(PLACA_POD) AS AVG_PLACA
   FROM PRACOWNICY
   GROUP BY ID_ZESP
) Z ON P.ID_ZESP = Z.ID_ZESP
ORDER BY P.NAZWISKO;
```

Ogranicz poprzedni zbiór tylko do tych pracowników, którzy zarabiają więcej niż średnia w ich zespole (czyli mających dodatnią wartość różnicy między ich płacą podstawową a średnią płacą w ich zespole). Modyfikacji poddaj oba rozwiązania z poprzedniego punktu.

```
SELECT P.NAZWISKO, P.PLACA_POD,
      P.PLACA_POD - (
         SELECT AVG(PLACA_POD)
         FROM PRACOWNICY
         WHERE ID_ZESP = P.ID_ZESP
      ) AS ROZNICA
FROM PRACOWNICY P
WHERE P.PLACA_POD > (
   SELECT AVG(PLACA_POD)
   FROM PRACOWNICY
```

```
   WHERE ID_ZESP = P.ID_ZESP
)
ORDER BY P.NAZWISKO;

SELECT P.NAZWISKO, P.PLACA_POD, P.PLACA_POD - Z.AVG_PLACA AS ROZNICA
FROM PRACOWNICY P
JOIN (
   SELECT ID_ZESP, AVG(PLACA_POD) AS AVG_PLACA
   FROM PRACOWNICY
   GROUP BY ID_ZESP
) Z ON P.ID_ZESP = Z.ID_ZESP
WHERE P.PLACA_POD > Z.AVG_PLACA
ORDER BY P.NAZWISKO;
```

Wyświetl nazwiska profesorów, zatrudnionych na Piotrowie, wraz liczbą ich podwładnych. Wynik uporządkuj wg liczby podwładnych w porządku malejącym. Zastosuj podzapytanie w klauzuli SELECT.

```
SELECT P.NAZWISKO, (
   SELECT COUNT(*)
   FROM PRACOWNICY P2
   WHERE P2.ID_SZEFA = P.ID_PRAC
) AS PODWLADNI
FROM PRACOWNICY P
JOIN ZESPOLY Z ON P.ID_ZESP = Z.ID_ZESP
WHERE P.ETAT = 'PROFESOR' AND Z.ADRES LIKE '%PIOTROW%'
ORDER BY PODWLADNI DESC;
```

Dla każdego zespołu wylicz średnią płacę jego pracowników. Następnie porównaj średnią w zespole z ogólną średnią płac i odpowiednio oznacz nastroje w zespole: umieść :) jeśli średnia w zespole jest wyższa lub równa średniej ogólnej i :(w przeciwnym wypadku. Jeśli zespół nie ma pracowników, nastrój oznacz jako nieokreślony używając ???.

```
SELECT Z.NAZWA AS NAZWA,
      ROUND(NVL(SZ.AVG_PLACA, 0), 2) AS SREDNIA_W_ZESPOLE,
      ROUND(GLOBAL.AVG_PLACA_GLOBAL, 2) AS SREDNIA_OGOLNA,
      CASE
         WHEN SZ.AVG_PLACA IS NULL THEN '???'
         WHEN SZ.AVG_PLACA >= GLOBAL.AVG_PLACA_GLOBAL THEN ':)'
         ELSE ':('
      END AS NASTROJE
FROM ZESPOLY Z
LEFT JOIN (
   SELECT ID_ZESP, AVG(PLACA_POD) AS AVG_PLACA
   FROM PRACOWNICY
   GROUP BY ID_ZESP
) SZ ON Z.ID_ZESP = SZ.ID_ZESP,
(
   SELECT AVG(PLACA_POD) AS AVG_PLACA_GLOBAL
```

```
   FROM PRACOWNICY
) GLOBAL
ORDER BY Z.NAZWA;
```

Wyświetl wszystkie informacje o etatach z tabeli Etaty. Wynik zaprezentuj w porządku malejącym, ustalonym przez liczbę pracowników, zatrudnionych na poszczególnych etatach. Jeśli na dwóch lub więcej etatach pracowałoby tylu samo pracowników, uporządkuj etaty wg ich nazw. Posłuż się podzapytaniem w klauzuli ORDER BY.

```
SELECT E.NAZWA, E.PLACA_MIN, E.PLACA_MAX
FROM ETATY E
JOIN (
   SELECT ETAT, COUNT(*) AS LICZBA_PRAC
   FROM PRACOWNICY
   GROUP BY ETAT
) P ON E.NAZWA = P.ETAT
ORDER BY P.LICZBA_PRAC DESC, E.NAZWA;
```

Zaawansowane mechanizmy w zapytaniach(18.01.2025 15:44-koniec promptów 😕)

```
--Zad nr 1
SELECT 'zadanie nr 1' FROM DUAL;

-- Z wykorzystaniem FETCH FIRST
SELECT NAZWISKO, PLACA_POD
FROM PRACOWNICY
ORDER BY PLACA_POD DESC
FETCH FIRST 3 ROWS ONLY;

-- Z wykorzystaniem ROWNUM
SELECT NAZWISKO, PLACA_POD
FROM (
  SELECT NAZWISKO, PLACA_POD
  FROM PRACOWNICY
  ORDER BY PLACA_POD DESC
)
WHERE ROWNUM <= 3;

--Zad nr 2
```

```sql
SELECT 'zadanie nr 2' FROM DUAL;

-- Z wykorzystaniem OFFSET
SELECT NAZWISKO, PLACA_POD
FROM PRACOWNICY
ORDER BY PLACA_POD DESC
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;

-- Z wykorzystaniem ROWNUM
SELECT NAZWISKO, PLACA_POD
FROM (
  SELECT NAZWISKO, PLACA_POD, ROWNUM AS RN
  FROM (
    SELECT NAZWISKO, PLACA_POD
    FROM PRACOWNICY
    ORDER BY PLACA_POD DESC
  )
)
WHERE RN BETWEEN 6 AND 10;

--Zad nr 3
SELECT 'zadanie nr 3' FROM DUAL;

WITH ZESPOL_SREDNIE AS (
  SELECT ID_ZESP, AVG(PLACA_POD) AS SREDNIA_PLACA
  FROM PRACOWNICY
  GROUP BY ID_ZESP
)
SELECT P.NAZWISKO, P.PLACA_POD, P.PLACA_POD - Z.SREDNIA_PLACA AS ROZNICA
FROM PRACOWNICY P
JOIN ZESPOL_SREDNIE Z ON P.ID_ZESP = Z.ID_ZESP
WHERE P.PLACA_POD > Z.SREDNIA_PLACA
ORDER BY NAZWISKO ASC;

--Zad nr 4
SELECT 'zadanie nr 4' FROM DUAL;

WITH LATA AS (
  SELECT EXTRACT(YEAR FROM ZATRUDNIONY) AS ROK, COUNT(*) AS LICZBA
  FROM PRACOWNICY
  GROUP BY EXTRACT(YEAR FROM ZATRUDNIONY)
)
SELECT ROK, LICZBA
FROM LATA
ORDER BY LICZBA DESC;

--Zad nr 5
SELECT 'zadanie nr 5' FROM DUAL;

WITH LATA AS (
  SELECT EXTRACT(YEAR FROM ZATRUDNIONY) AS ROK, COUNT(*) AS LICZBA
```

```
  FROM PRACOWNICY
  GROUP BY EXTRACT(YEAR FROM ZATRUDNIONY)
)
SELECT ROK, LICZBA
FROM LATA
WHERE LICZBA = (SELECT MAX(LICZBA) FROM LATA);

--Zad nr 6
SELECT 'zadanie nr 6' FROM DUAL;

WITH ASYSTENCI AS (
  SELECT * FROM PRACOWNICY WHERE ETAT = 'ASYSTENT'
),
PIOTROWO AS (
  SELECT * FROM ZESPOLY WHERE ADRES LIKE '%PIOTROWO%'
)
SELECT A.NAZWISKO, A.ETAT, Z.NAZWA, Z.ADRES
FROM ASYSTENCI A
JOIN PIOTROWO Z ON A.ID_ZESP = Z.ID_ZESP;

--Zad nr 7
SELECT 'zadanie nr 7' FROM DUAL;

WITH SUMY_PLAC AS (
  SELECT Z.NAZWA, SUM(P.PLACA_POD) AS MAKS_SUMA_PLAC
  FROM ZESPOLY Z
  JOIN PRACOWNICY P ON Z.ID_ZESP = P.ID_ZESP
  GROUP BY Z.NAZWA
)
SELECT NAZWA, MAKS_SUMA_PLAC
FROM SUMY_PLAC
WHERE MAKS_SUMA_PLAC = (SELECT MAX(MAKS_SUMA_PLAC) FROM SUMY_PLAC);

--Zad nr 8
SELECT 'zadanie nr 8' FROM DUAL;

-- Z rekurencyjną klauzulą WITH z aliasami kolumn
WITH REKURSJA (ID_PRAC, NAZWISKO, POZYCJA_W_HIERARCHII) AS (
  SELECT ID_PRAC, NAZWISKO, 1 AS POZYCJA_W_HIERARCHII
  FROM PRACOWNICY
  WHERE NAZWISKO = 'BRZEZINSKI'
  UNION ALL
  SELECT P.ID_PRAC, P.NAZWISKO, R.POZYCJA_W_HIERARCHII + 1
  FROM PRACOWNICY P
  JOIN REKURSJA R ON P.ID_SZEFA = R.ID_PRAC
)
SELECT NAZWISKO, POZYCJA_W_HIERARCHII
FROM REKURSJA;


-- Zapytanie hierarchiczne w Oracle
```

```
SELECT NAZWISKO, LEVEL AS POZYCJA_W_HIERARCHII
FROM PRACOWNICY
START WITH NAZWISKO = 'BRZEZINSKI'
CONNECT BY PRIOR ID_PRAC = ID_SZEFA
ORDER BY POZYCJA_W_HIERARCHII;

--Zad nr 9
SELECT 'zadanie nr 9' FROM DUAL;

SELECT LPAD(' ', LEVEL * 2) || NAZWISKO AS NAZWISKO, LEVEL AS POZYCJA_W_HIERARCHII
FROM PRACOWNICY
START WITH NAZWISKO = 'BRZEZINSKI'
CONNECT BY PRIOR ID_PRAC = ID_SZEFA
ORDER BY POZYCJA_W_HIERARCHII;

--Zad nr 10
SELECT 'zadanie nr 10' FROM DUAL;

WITH CYFRA_NAZWA AS (
  SELECT 0 AS CYFRA, 'zero' AS NAZWA FROM DUAL
  UNION ALL SELECT 1, 'jeden' FROM DUAL
  UNION ALL SELECT 2, 'dwa' FROM DUAL
  UNION ALL SELECT 3, 'trzy' FROM DUAL

)
SELECT
  P.NAZWISKO,
  'zarobki w tysiącach: ' || C.NAZWA AS ZAROBKI
FROM
  PRACOWNICY P
JOIN
  CYFRA_NAZWA C
ON
  FLOOR((NVL(P.PLACA_POD, 0) + NVL(P.PLACA_DOD, 0)) / 1000) = C.CYFRA
ORDER BY
  P.NAZWISKO ASC;
```

Język manipulowania danymi (DML)

DELETE FROM pracownicy;

```
INSERT INTO PRACOWNICY VALUES (100,'WEGLARZ'    ,'DYREKTOR'
,NULL,to_date('01-01-1968','DD-MM-YYYY'),1730.00,420.50,10);
INSERT INTO PRACOWNICY VALUES (110,'BLAZEWICZ'  ,'PROFESOR'  ,100
,to_date('01-05-1973','DD-MM-YYYY'),1350.00,210.00,40);
INSERT INTO PRACOWNICY VALUES (120,'SLOWINSKI'  ,'PROFESOR'  ,100
,to_date('01-09-1977','DD-MM-YYYY'),1070.00,  NULL,30);
INSERT INTO PRACOWNICY VALUES (130,'BRZEZINSKI' ,'PROFESOR'  ,100
,to_date('01-07-1968','DD-MM-YYYY'), 960.00,  NULL,20);
INSERT INTO PRACOWNICY VALUES (140,'MORZY'      ,'PROFESOR'  ,130
,to_date('15-09-1975','DD-MM-YYYY'), 830.00,105.00,20);
INSERT INTO PRACOWNICY VALUES (150,'KROLIKOWSKI','ADIUNKT'   ,130
,to_date('01-09-1977','DD-MM-YYYY'), 645.50,  NULL,20);
INSERT INTO PRACOWNICY VALUES (160,'KOSZLAJDA'  ,'ADIUNKT'   ,130
,to_date('01-03-1985','DD-MM-YYYY'), 590.00,  NULL,20);
INSERT INTO PRACOWNICY VALUES (170,'JEZIERSKI'  ,'ASYSTENT'  ,130
,to_date('01-10-1992','DD-MM-YYYY'), 439.70, 80.50,20);
INSERT INTO PRACOWNICY VALUES (190,'MATYSIAK'   ,'ASYSTENT'  ,140
,to_date('01-09-1993','DD-MM-YYYY'), 371.00,  NULL,20);
INSERT INTO PRACOWNICY VALUES (180,'MAREK'      ,'SEKRETARKA',100
,to_date('20-02-1985','DD-MM-YYYY'), 410.20,  NULL,10);
INSERT INTO PRACOWNICY VALUES (200,'ZAKRZEWICZ' ,'STAZYSTA'  ,140
,to_date('15-07-1994','DD-MM-YYYY'), 208.00,  NULL,30);
INSERT INTO PRACOWNICY VALUES (210,'BIALY'      ,'STAZYSTA'  ,130
,to_date('15-10-1993','DD-MM-YYYY'), 250.00,170.60,30);
INSERT INTO PRACOWNICY VALUES (220,'KONOPKA'    ,'ASYSTENT'  ,110
,to_date('01-10-1993','DD-MM-YYYY'), 480.00,  NULL,20);
INSERT INTO PRACOWNICY VALUES (230,'HAPKE'      ,'ASYSTENT'  ,120
,to_date('01-09-1992','DD-MM-YYYY'), 480.00, 90.00,30);
SELECT '' AS "Zadanie_1" FROM DUAL;
INSERT INTO PRACOWNICY (ID_PRAC, NAZWISKO, ETAT, ID_SZEFA, ZATRUDNIONY,
PLACA_POD, PLACA_DOD, ID_ZESP)
VALUES (250, 'KOWALSKI', 'ASYSTENT', NULL, TO_DATE('2015-01-13', 'YYYY-MM-DD'),
1500, NULL, 10);

INSERT INTO PRACOWNICY (ID_PRAC, NAZWISKO, ETAT, ID_SZEFA, ZATRUDNIONY,
PLACA_POD, PLACA_DOD, ID_ZESP)
VALUES (260, 'ADAMSKI', 'ASYSTENT', NULL, TO_DATE('2014-09-10', 'YYYY-MM-DD'),
1500, NULL, 10);

INSERT INTO PRACOWNICY (ID_PRAC, NAZWISKO, ETAT, ID_SZEFA, ZATRUDNIONY,
PLACA_POD, PLACA_DOD, ID_ZESP)
VALUES (270, 'NOWAK', 'ADIUNKT', NULL, TO_DATE('1990-05-01', 'YYYY-MM-DD'), 2050,
540, 20);

SELECT ID_PRAC, NAZWISKO, ETAT, ID_SZEFA, TO_CHAR(ZATRUDNIONY,
'YYYY-MM-DD') AS ZATRUDNIONY, PLACA_POD, PLACA_DOD, ID_ZESP
FROM PRACOWNICY
WHERE ID_PRAC IN (250, 260, 270);
```

```
SELECT '' AS "Zadanie_2" FROM DUAL;
UPDATE PRACOWNICY
SET
   PLACA_POD = PLACA_POD * 1.1,
   PLACA_DOD = CASE
               WHEN PLACA_DOD IS NULL THEN 100
               ELSE PLACA_DOD * 1.2
            END
WHERE ID_PRAC IN (250, 260, 270);

SELECT * FROM PRACOWNICY
WHERE ID_PRAC IN (250, 260, 270);

SELECT '' AS "Zadanie_3" FROM DUAL;
INSERT INTO ZESPOLY (ID_ZESP, NAZWA, ADRES)
VALUES (60, 'BAZY DANYCH', 'PIOTROWO 2');

SELECT * FROM ZESPOLY
WHERE ID_ZESP = 60;

SELECT '' AS "Zadanie_4" FROM DUAL;
UPDATE PRACOWNICY
SET ID_ZESP = (SELECT ID_ZESP FROM ZESPOLY WHERE NAZWA = 'BAZY
DANYCH')
WHERE ID_PRAC IN (250, 260, 270);

SELECT * FROM PRACOWNICY
WHERE ID_ZESP = (SELECT ID_ZESP FROM ZESPOLY WHERE NAZWA = 'BAZY
DANYCH');

SELECT '' AS "Zadanie_5" FROM DUAL;
UPDATE pracownicy
SET id_szefa = (SELECT id_prac FROM pracownicy WHERE nazwisko = 'MORZY')
WHERE id_zesp = (SELECT id_zesp FROM zespoly WHERE nazwa = 'BAZY DANYCH');

SELECT * FROM pracownicy
WHERE id_szefa = (SELECT id_prac FROM pracownicy WHERE nazwisko = 'MORZY')
ORDER BY id_prac asc;

SELECT '' AS "Zadanie_6" FROM DUAL;
DELETE FROM ZESPOLY
WHERE NAZWA = 'BAZY DANYCH';

SELECT '' AS "Zadanie_7" FROM DUAL;
DELETE FROM PRACOWNICY
WHERE ID_ZESP = (SELECT ID_ZESP FROM ZESPOLY WHERE NAZWA = 'BAZY
DANYCH');
```

```
DELETE FROM ZESPOLY
WHERE NAZWA = 'BAZY DANYCH';

SELECT * FROM PRACOWNICY;
SELECT * FROM ZESPOLY;

SELECT '' AS "Zadanie_8" FROM DUAL;
SELECT NAZWISKO, PLACA_POD,
      ROUND(0.1 * (SELECT AVG(PLACA_POD)
                  FROM PRACOWNICY p2
                  WHERE p2.ID_ZESP = p.ID_ZESP), 2) AS PODWYZKA
FROM PRACOWNICY p
ORDER by nazwisko;

SELECT '' AS "Zadanie_9" FROM DUAL;
UPDATE PRACOWNICY p1
SET PLACA_POD = PLACA_POD +
   ROUND(0.1 * (SELECT AVG(PLACA_POD)
               FROM PRACOWNICY p2
               WHERE p2.ID_ZESP = p1.ID_ZESP), 2);

SELECT NAZWISKO, PLACA_POD
FROM PRACOWNICY
ORDER by pracownicy.nazwisko;

SELECT '' AS "Zadanie_10" FROM DUAL;
SELECT ID_PRAC, NAZWISKO, ETAT, ID_SZEFA, TO_CHAR(ZATRUDNIONY,
'YYYY-MM-DD') AS ZATRUDNIONY, PLACA_POD, PLACA_DOD, ID_ZESP
FROM PRACOWNICY
WHERE PLACA_POD = (SELECT MIN(PLACA_POD) FROM PRACOWNICY);

SELECT '' AS "Zadanie_11" FROM DUAL;
UPDATE PRACOWNICY
SET PLACA_POD = ROUND((SELECT AVG(PLACA_POD) FROM PRACOWNICY), 2)
WHERE PLACA_POD = (SELECT MIN(PLACA_POD) FROM PRACOWNICY);

SELECT ID_PRAC, NAZWISKO, ETAT, ID_SZEFA, TO_CHAR(ZATRUDNIONY,
'YYYY-MM-DD') AS ZATRUDNIONY, PLACA_POD, PLACA_DOD, ID_ZESP
FROM PRACOWNICY
WHERE ID_PRAC = 200;

SELECT '' AS "Zadanie_12" FROM DUAL;
SELECT NAZWISKO, PLACA_DOD
FROM PRACOWNICY
WHERE ID_ZESP = 20
ORDER BY pracownicy.nazwisko;
```

```
UPDATE PRACOWNICY
SET PLACA_DOD = (
   SELECT AVG(PLACA_POD)
   FROM PRACOWNICY
   WHERE ID_SZEFA = (SELECT ID_PRAC FROM PRACOWNICY WHERE NAZWISKO =
'MORZY')
)
WHERE ID_ZESP = 20;


SELECT NAZWISKO, PLACA_DOD
FROM PRACOWNICY
WHERE ID_ZESP = 20;


SELECT '' AS "Zadanie_13" FROM DUAL;
-- Wyświetlenie płac podstawowych przed aktualizacją
SELECT P.NAZWISKO, P.PLACA_POD
FROM PRACOWNICY P
JOIN ZESPOLY Z ON P.ID_ZESP = Z.ID_ZESP
WHERE Z.NAZWA = 'SYSTEMY ROZPROSZONE';

-- Aktualizacja płac podstawowych z podwyżką o 25%
UPDATE PRACOWNICY P
SET PLACA_POD = PLACA_POD * 1.25
WHERE P.ID_ZESP = (SELECT ID_ZESP FROM ZESPOLY WHERE NAZWA = 'SYSTEMY
ROZPROSZONE');

-- Wyświetlenie płac podstawowych po aktualizacji
SELECT P.NAZWISKO, P.PLACA_POD
FROM PRACOWNICY P
JOIN ZESPOLY Z ON P.ID_ZESP = Z.ID_ZESP
WHERE Z.NAZWA = 'SYSTEMY ROZPROSZONE'
ORDER BY P.NAZWISKO;


SELECT '' AS "Zadanie_14" FROM DUAL;
-- Wyświetlenie bezpośrednich podwładnych pracownika MORZY przed usunięciem
SELECT P1.NAZWISKO AS PRACOWNIK, P2.NAZWISKO AS SZEF
FROM PRACOWNICY P1
JOIN PRACOWNICY P2 ON P1.ID_SZEFA = P2.ID_PRAC
WHERE P2.NAZWISKO = 'MORZY';

-- Usunięcie bezpośrednich podwładnych pracownika MORZY
DELETE FROM PRACOWNICY
```

```
WHERE ID_SZEFA = (SELECT ID_PRAC FROM PRACOWNICY WHERE NAZWISKO =
'MORZY');

-- Sprawdzenie, czy usunięci podwładni już nie istnieją
SELECT P1.NAZWISKO AS PRACOWNIK, P2.NAZWISKO AS SZEF
FROM PRACOWNICY P1
JOIN PRACOWNICY P2 ON P1.ID_SZEFA = P2.ID_PRAC
WHERE P2.NAZWISKO = 'MORZY';


SELECT '' AS "Zadanie_15" FROM DUAL;
SELECT * FROM PRACOWNICY
ORDER BY NAZWISKO;

SELECT '' AS "Zadanie_16" FROM DUAL;
CREATE SEQUENCE PRAC_SEQ
START WITH 300
INCREMENT BY 10;

SELECT '' AS "Zadanie_17" FROM DUAL;
INSERT INTO PRACOWNICY (ID_PRAC, NAZWISKO, ETAT, PLACA_POD)
VALUES (PRAC_SEQ.NEXTVAL, 'TRĄBCZYŃSKI', 'STAZYSTA', 1000);

SELECT * FROM PRACOWNICY
WHERE NAZWISKO = 'TRĄBCZYŃSKI';

SELECT '' AS "Zadanie_18" FROM DUAL;
UPDATE PRACOWNICY
SET PLACA_DOD = PRAC_SEQ.CURRVAL
WHERE NAZWISKO = 'TRĄBCZYŃSKI';

SELECT * FROM PRACOWNICY
WHERE NAZWISKO = 'TRĄBCZYŃSKI';

SELECT '' AS "Zadanie_19" FROM DUAL;
DELETE FROM PRACOWNICY
WHERE NAZWISKO = 'TRĄBCZYŃSKI';

SELECT * FROM PRACOWNICY
WHERE NAZWISKO = 'TRĄBCZYŃSKI';

SELECT '' AS "Zadanie_20" FROM DUAL;
CREATE SEQUENCE MALA_SEQ
START WITH 1
INCREMENT BY 1
MAXVALUE 10;

SELECT MALA_SEQ.NEXTVAL FROM DUAL;
```

```
SELECT '' AS "Zadanie_21" FROM DUAL;
DROP SEQUENCE MALA_SEQ;
```

Język definiowania danych (DDL) część 1

```
SELECT '' as "Zadanie 1" from dual;
CREATE TABLE PROJEKTY (
    ID_PROJEKTU INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    OPIS_PROJEKTU VARCHAR(20),
    DATA_ROZPOCZECIA DATE DEFAULT CURRENT_DATE,
    DATA_ZAKONCZENIA DATE,
    FUNDUSZ NUMBER(7,2)
);
SELECT '' as "Zadanie 2" from dual;
INSERT INTO PROJEKTY (OPIS_PROJEKTU, DATA_ROZPOCZECIA,
DATA_ZAKONCZENIA, FUNDUSZ)
VALUES ('Indeksy bitmapowe', TO_DATE('1999-04-02', 'YYYY-MM-DD'),
TO_DATE('2001-08-31', 'YYYY-MM-DD'), 25000);


INSERT INTO PROJEKTY (OPIS_PROJEKTU, DATA_ROZPOCZECIA, FUNDUSZ)
VALUES (
    'Sieci kręgosłupowe',
    TO_DATE('2017-02-21', 'YYYY-MM-DD'),
    19000
);

SELECT '' as "Zadanie 3" from dual;
SELECT ID_PROJEKTU, OPIS_PROJEKTU
FROM PROJEKTY;

SELECT '' as "Zadanie 4" from dual;
-- Próba wstawienia z ręcznym ID_PROJEKTU
INSERT INTO PROJEKTY (ID_PROJEKTU, OPIS_PROJEKTU, DATA_ROZPOCZECIA,
DATA_ZAKONCZENIA, FUNDUSZ)
VALUES (10, 'Indeksy drzewiaste', '2013-12-24', '2014-01-01', 1200);

--Błąd: Pole ID_PROJEKTU jest automatycznie generowane. Nie można podać wartości
ręcznie.
```

```
INSERT INTO PROJEKTY (OPIS_PROJEKTU, DATA_ROZPOCZECIA,
DATA_ZAKONCZENIA, FUNDUSZ)
VALUES (
   'Indeksy drzewiaste',
   TO_DATE('2013-12-24', 'YYYY-MM-DD'),
   TO_DATE('2014-01-01', 'YYYY-MM-DD'),
   1200
);

SELECT ID_PROJEKTU, OPIS_PROJEKTU
FROM PROJEKTY;


SELECT '' as "Zadanie 5" from dual;

UPDATE PROJEKTY
SET ID_PROJEKTU = 10
WHERE OPIS_PROJEKTU = 'Indeksy drzewiaste';
-- TAK SAMO ID JEST PRZYDZIELANE AUTOMATYCZNIE WIEC NIE MOZNA GO
NADPISAC UPDATE
SELECT '' as "Zadanie 6" from dual;

UPDATE PROJEKTY
SET DATA_ROZPOCZECIA = TO_DATE('2017-02-21', 'YYYY-MM-DD')
WHERE OPIS_PROJEKTU = 'Sieci kręgosłupowe';


CREATE TABLE PROJEKTY_KOPIA AS
SELECT * FROM PROJEKTY;
SELECT
   ID_PROJEKTU,
   OPIS_PROJEKTU,
   TO_CHAR(DATA_ROZPOCZECIA, 'YYYY-MM-DD') AS DATA_ROZPOCZECIA,
   TO_CHAR(DATA_ZAKONCZENIA, 'YYYY-MM-DD') AS DATA_ZAKONCZENIA,
   FUNDUSZ
FROM PROJEKTY;



SELECT '' as "Zadanie 7" from dual;
INSERT INTO PROJEKTY_KOPIA (ID_PROJEKTU, OPIS_PROJEKTU,
DATA_ROZPOCZECIA, DATA_ZAKONCZENIA, FUNDUSZ)
VALUES (
   10, 'Sieci lokalne', SYSDATE, ADD_MONTHS(SYSDATE, 12),  24500);
--W tabeli PROJEKTY_KOPIA kolumna ID_PROJEKTU nie ma ograniczenia GENERATED
ALWAYS AS IDENTITY.
```

```
SELECT '' as "Zadanie 8" from dual;
DELETE FROM PROJEKTY
WHERE OPIS_PROJEKTU = 'Indeksy drzewiaste';
SELECT * FROM PROJEKTY;
SELECT * FROM PROJEKTY_KOPIA;


SELECT '' as "Zadanie 9" from dual;
SELECT TABLE_NAME
FROM USER_TABLES;
```

Język definiowania danych (DDL) część 2

```
SELECT ' ' as "Zadanie 1" from dual;

ALTER TABLE PROJEKTY DROP PRIMARY KEY;
ALTER TABLE PROJEKTY ADD CONSTRAINT PK_PROJEKTY PRIMARY KEY
(ID_PROJEKTU);

ALTER TABLE PROJEKTY
ADD CONSTRAINT UK_PROJEKTY UNIQUE (OPIS_PROJEKTU);

ALTER TABLE PROJEKTY
MODIFY OPIS_PROJEKTU NOT NULL;

ALTER TABLE PROJEKTY
ADD CONSTRAINT CHK_DATA CHECK (DATA_ZAKONCZENIA > DATA_ROZPOCZECIA);

ALTER TABLE PROJEKTY
ADD CONSTRAINT CHK_FUNDUSZ CHECK (FUNDUSZ > 0);

SELECT
    A.CONSTRAINT_NAME AS CONSTRAINT_NAME,
    A.CONSTRAINT_TYPE AS C_TYPE,
    A.SEARCH_CONDITION AS SEARCH_CONDITION,
    B.COLUMN_NAME AS COLUMN_NAME
FROM USER_CONSTRAINTS A
LEFT JOIN USER_CONS_COLUMNS B ON A.CONSTRAINT_NAME =
B.CONSTRAINT_NAME
WHERE A.TABLE_NAME = 'PROJEKTY';
```

```
SELECT '' as "Zadanie 2" from dual;
-- Zadanie 2: Próba wstawienia duplikatu OPIS_PROJEKTU
INSERT INTO PROJEKTY (OPIS_PROJEKTU, DATA_ROZPOCZECIA,
DATA_ZAKONCZENIA, FUNDUSZ)
VALUES ('Indeksy bitmapowe', TO_DATE('12-04-2015', 'DD-MM-YYYY'),
TO_DATE('30-09-2016', 'DD-MM-YYYY'), 40000);
-- index nie jesy unique

SELECT '' as "Zadanie 3" from dual;

-- Zadanie 3: Tworzenie tabeli PRZYDZIALY
CREATE TABLE PRZYDZIALY (
   ID_PROJEKTU INT NOT NULL,
   NR_PRACOWNIKA INT NOT NULL,
   PRZYDZIAL_OD DATE DEFAULT SYSDATE,
   PRZYDZIAL_DO DATE,
   STAWKA NUMBER(7, 2),
   ROLA VARCHAR2(20),
   CONSTRAINT PK_PRZYDZIALY PRIMARY KEY (ID_PROJEKTU, NR_PRACOWNIKA),
   CONSTRAINT FK_PRZYDZIALY_01 FOREIGN KEY (ID_PROJEKTU) REFERENCES
PROJEKTY (ID_PROJEKTU),
   CONSTRAINT FK_PRZYDZIALY_02 FOREIGN KEY (NR_PRACOWNIKA)
REFERENCES PRACOWNICY (ID_PRAC),
   CONSTRAINT CHK_PRZYDZIALY_DATY CHECK (PRZYDZIAL_DO > PRZYDZIAL_OD),
   CONSTRAINT CHK_PRZYDZIALY_STAWKA CHECK (STAWKA > 0),
   CONSTRAINT CHK_PRZYDZIALY_ROLA CHECK (ROLA IN ('KIERUJĄCY', 'ANALITYK',
'PROGRAMISTA'))
);

SELECT '' as "Zadanie 4" from dual;

-- Zadanie 4: Wstawianie rekordów do PRZYDZIALY
INSERT INTO PRZYDZIALY (ID_PROJEKTU, NR_PRACOWNIKA, PRZYDZIAL_OD,
PRZYDZIAL_DO, STAWKA, ROLA)
VALUES (
   (SELECT ID_PROJEKTU FROM PROJEKTY WHERE OPIS_PROJEKTU = 'Indeksy
bitmapowe'), -- Pobiera ID projektu
   170,
   TO_DATE('10-04-1999', 'DD-MM-YYYY'),
   TO_DATE('10-05-1999', 'DD-MM-YYYY'),
   1000,
   'KIERUJĄCY'
);

INSERT INTO PRZYDZIALY (ID_PROJEKTU, NR_PRACOWNIKA, PRZYDZIAL_OD,
PRZYDZIAL_DO, STAWKA, ROLA)
VALUES (
```

```
   (SELECT ID_PROJEKTU FROM PROJEKTY WHERE OPIS_PROJEKTU = 'Indeksy
bitmapowe'), -- Pobiera ID projektu
   140,
   TO_DATE('01-12-2000', 'DD-MM-YYYY'),
   NULL, -- PRZYDZIAL_DO nie jest podany
   1500,
   'ANALITYK'
);

INSERT INTO PRZYDZIALY (ID_PROJEKTU, NR_PRACOWNIKA, PRZYDZIAL_OD,
PRZYDZIAL_DO, STAWKA, ROLA)
VALUES (
   (SELECT ID_PROJEKTU FROM PROJEKTY WHERE OPIS_PROJEKTU = 'Sieci
kręgosłupowe'), -- Pobiera ID projektu
   140,
   TO_DATE('14-09-2015', 'DD-MM-YYYY'),
   NULL, -- PRZYDZIAL_DO nie jest podany
   2500,
   'KIERUJĄCY'
);

-- Sprawdzenie zawartości tabeli PRZYDZIALY
SELECT
   ID_PROJEKTU,
   NR_PRACOWNIKA,
   TO_CHAR(PRZYDZIAL_OD, 'YYYY-MM-DD') AS OD,
   TO_CHAR(PRZYDZIAL_DO, 'YYYY-MM-DD') AS PDO,
   STAWKA,
   ROLA
FROM PRZYDZIALY;


SELECT '' as "Zadanie 5" from dual;
-- Zadanie 5: Dodanie atrybutu GODZINY
ALTER TABLE PRZYDZIALY
ADD GODZINY NUMBER(4) NOT NULL CHECK (GODZINY <= 9999);

--ALTER TABLE PRZYDZIALY ADD (GODZINY INTEGER CHECK (GODZINY <= 9999));


SELECT '' as "Zadanie 6" from dual;
-- Zadanie 6: Wyłączenie unikalności opisu projektów (OracleDB)
ALTER TABLE PRZYDZIALY
ADD GODZINY NUMBER(4) CHECK (GODZINY <= 9999);
UPDATE PRZYDZIALY
SET GODZINY = 40;
ALTER TABLE PRZYDZIALY
MODIFY GODZINY NOT NULL;
```

```
SELECT '' as "Zadanie 7" from dual;
-- Zadanie 7: Wyłączenie unikalności opisu projektów (OracleDB)
ALTER TABLE PROJEKTY DISABLE CONSTRAINT UK_PROJEKTY;



-- Sprawdzenie statusu ograniczeń
SELECT CONSTRAINT_NAME, STATUS FROM USER_CONSTRAINTS WHERE
CONSTRAINT_NAME = 'UK_PROJEKTY';

SELECT '' as "Zadanie 8" from dual;

INSERT INTO PROJEKTY (OPIS_PROJEKTU, DATA_ROZPOCZECIA,
DATA_ZAKONCZENIA, FUNDUSZ)
VALUES ('Indeksy bitmapowe', DATE '2015-04-12', DATE '2016-09-30', 40000);

SELECT ID_PROJEKTU, OPIS_PROJEKTU, TO_CHAR(DATA_ROZPOCZECIA,
'YYYY-MM-DD') AS DATA_ROZPOCZECIA , TO_CHAR(DATA_ZAKONCZENIA,
'YYYY-MM-DD') AS DATA_ZAKONCZENIA, FUNDUSZ
FROM PROJEKTY;



SELECT '' as "Zadanie 9" from dual;

--ZAD9
ALTER TABLE PROJEKTY
ENABLE CONSTRAINT UK_PROJEKTY;
-- JESLI SA DUPLIKATY MUSIMY SIE ICH POZABYC ABY WŁĄCZYĆ OGRANICZENIE.

SELECT '' as "Zadanie 10" from dual;

-- Zadanie 10: Zmiana opisu projektu
UPDATE PROJEKTY SET OPIS_PROJEKTU = 'Inne indeksy'
WHERE ID_PROJEKTU = 5;

-- Włączenie ograniczenia
ALTER TABLE PROJEKTY ENABLE CONSTRAINT UK_PROJEKTY;

--TERAZ JUZ MOZNA

SELECT '' as "Zadanie 11" from dual;

-- Zadanie 11: Zmiana rozmiaru OPIS_PROJEKTU
ALTER TABLE PROJEKTY MODIFY OPIS_PROJEKTU VARCHAR(10);
```

```
SELECT '' as "Zadanie 12" from dual;

-- Zadanie 12: Usunięcie rekordu z opisem 'Sieci kręgosłupowe'
DELETE FROM PROJEKTY WHERE OPIS_PROJEKTU = 'Sieci kręgosłupowe';
--W tabeli PRZYDZIALY istnieje kolumna, która zawiera klucz obcy, odnoszący się do klucza głównego w tabeli PROJEKTY.
SELECT '' as "Zadanie 13" from dual;

-- Zadanie 13: Zmiana klucza obcego FK_PRZYDZIALY_01
ALTER TABLE PRZYDZIALY DROP CONSTRAINT FK_PRZYDZIALY_01;
ALTER TABLE PRZYDZIALY ADD CONSTRAINT FK_PRZYDZIALY_01 FOREIGN KEY (ID_PROJEKTU) REFERENCES PROJEKTY(ID_PROJEKTU) ON DELETE CASCADE;

DELETE FROM PROJEKTY
WHERE OPIS_PROJEKTU = 'Sieci kręgosłupowe';

SELECT * FROM PROJEKTY;
SELECT * FROM PRZYDZIALY;

SELECT '' as "Zadanie 14" from dual;

-- Zadanie 14: Sprawdzenie ograniczeń integralnościowych w relacji
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION,
R_CONSTRAINT_NAME, STATUS
FROM USER_CONSTRAINTS
WHERE TABLE_NAME = 'PRZYDZIALY';

SELECT '' as "Zadanie 15" from dual;

--15 Usunięcie relacji PROJEKTY wraz z kluczami obcymi
DROP TABLE PROJEKTY CASCADE CONSTRAINTS;

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, STATUS
FROM USER_CONSTRAINTS
WHERE TABLE_NAME = 'PRZYDZIALY';

SELECT '' as "Zadanie 16" from dual;
/*
--16
DROP TABLE PRZYDZIALY;
DROP TABLE PRZYDZIALY_KOPIA;
DROP TABLE PROJEKTY;
DROP TABLE PROJEKTY_KOPIA;

SELECT TABLE_NAME
FROM USER_TABLES;

*/
```

ROZWIĄZANE KOLOSY

ZESTAW 1

```
--ZAD1
Dla danych pokazanych obok podaj wyniki poniższych zapytań (koniecznie zapisz
wyniki operacji
pośrednich np. połączeń, grupowań, wyliczenia wartości wyrażeń, podzapytań itp.):
CREATE TABLE A_B (
    A NUMBER,
    B NUMBER
);
INSERT INTO A_B (A, B) VALUES (1, 1);
INSERT INTO A_B (A, B) VALUES (2, NULL);
INSERT INTO A_B (A, B) VALUES (3, 2);
INSERT INTO A_B (A, B) VALUES (1, NULL);
INSERT INTO A_B (A, B) VALUES (NULL, 2);

SELECT SUM(A)
FROM A_B
GROUP BY A+B
zwróci
  SUM(A)
----------
         1
         3
         3
poniewaz pierw stworzy grupy a+b np 1+1 = 2, a jak null to zawsze null wynikowy
wiec beda grupy nu 2, null, 5

    SUM(A)        A+B
```

```
---------- ----------
         1          2 bo wiersz 1 a=1
         3            bo wiersz 2 a=2 wiersz 3 a=1 wiersz 5 a=null wiec 2+1=3
         3          5 bo wiersz 3 a=3

SELECT SUM(DISTINCT A)
FROM A_B b
WHERE b.B = (SELECT COUNT(*)
             FROM A_B a
             WHERE a.B = b.B);
zwroci 4 potem wkleje screna w paincie to rozpisalem
---Skrpty DDL i DML ktory towrzy te realcje z przykladowymi danymi
Skrypt DDL
-- Tabela PRACOWNICY
CREATE TABLE PRACOWNICY (
    ID_PRAC NUMBER(4) CONSTRAINT PK_ID_PRAC PRIMARY KEY,
    NAZWISKO VARCHAR2(51),
    STANOWISKO VARCHAR2(51),
    PLACA_MIESIECZNA NUMBER(6,2)
);

-- Tabela PROJEKTY
CREATE TABLE PROJEKTY (
    ID_PROJEKTU NUMBER(4) CONSTRAINT PK_ID_PROJEKTU PRIMARY KEY,
    NAZWA VARCHAR2(20),
    BUDZET NUMBER(8,2),
    ID_OPIEKUNA NUMBER(4) CONSTRAINT FK_ID_OPIEKUNA REFERENCES PRACOWNICY(ID_PRAC)
);

-- Tabela PRZYDZIALY
CREATE TABLE PRZYDZIALY (
    ID_PROJEKTU NUMBER(4) CONSTRAINT FK_ID_PROJEKTU REFERENCES
PROJEKTY(ID_PROJEKTU),
    ID_PRAC NUMBER(4) CONSTRAINT FK_ID_PRAC REFERENCES PRACOWNICY(ID_PRAC),
    ROLA VARCHAR2(20),
    DATA_ROZPOCZECIA DATE DEFAULT CURRENT_DATE,
    STAWKA_MIESIECZNA NUMBER(6,2),
    LICZBA_MIESIECY NUMBER,
    CONSTRAINT PK_PRZYDZIAL PRIMARY KEY (ID_PROJEKTU, ID_PRAC, ROLA)
);

Skrypt DML
-- Wstawianie danych do tabeli PRACOWNICY
INSERT INTO PRACOWNICY (ID_PRAC, NAZWISKO, STANOWISKO, PLACA_MIESIECZNA) VALUES (1,
'Kowalski', 'ADMINISTRATOR', 3000);
INSERT INTO PRACOWNICY (ID_PRAC, NAZWISKO, STANOWISKO, PLACA_MIESIECZNA) VALUES (2,
'Nowak', 'PROGRAMISTA', 2500);
INSERT INTO PRACOWNICY (ID_PRAC, NAZWISKO, STANOWISKO, PLACA_MIESIECZNA) VALUES (3,
'Wiśniewski', 'TECHNICZNA', 1800);
INSERT INTO PRACOWNICY (ID_PRAC, NAZWISKO, STANOWISKO, PLACA_MIESIECZNA) VALUES (4,
'Jankowski', 'KIEROWNIK', 4000);
```

```
INSERT INTO PRACOWNICY (ID_PRAC, NAZWISKO, STANOWISKO, PLACA_MIESIECZNA) VALUES (5,
'Zieliński', 'ADMINISTRATOR', 2100);
INSERT INTO PRACOWNICY (ID_PRAC, NAZWISKO, STANOWISKO, PLACA_MIESIECZNA) VALUES
(9999, 'Nowak', 'TESTER', 4000);


-- Wstawianie danych do tabeli PROJEKTY
INSERT INTO PROJEKTY (ID_PROJEKTU, NAZWA, BUDZET, ID_OPIEKUNA) VALUES (100,
'Projekt A', 15000, 1);
INSERT INTO PROJEKTY (ID_PROJEKTU, NAZWA, BUDZET, ID_OPIEKUNA) VALUES (200,
'Projekt B', 8000, 2);
INSERT INTO PROJEKTY (ID_PROJEKTU, NAZWA, BUDZET, ID_OPIEKUNA) VALUES (300, 'XML',
12000, 4);
INSERT INTO PROJEKTY (ID_PROJEKTU, NAZWA, BUDZET, ID_OPIEKUNA) VALUES (400,
'Projekt C', 5000, 5);

-- Wstawianie danych do tabeli PRZYDZIALY
INSERT INTO PRZYDZIALY (ID_PROJEKTU, ID_PRAC, ROLA, DATA_ROZPOCZECIA,
STAWKA_MIESIECZNA, LICZBA_MIESIECY) VALUES (100, 1, 'ADMINISTRACYJNY',
TO_DATE('2024-01-01', 'YYYY-MM-DD'), 2500, 6);
INSERT INTO PRZYDZIALY (ID_PROJEKTU, ID_PRAC, ROLA, DATA_ROZPOCZECIA,
STAWKA_MIESIECZNA, LICZBA_MIESIECY) VALUES (100, 2, 'PROGRAMISTYCZNA',
TO_DATE('2024-02-01', 'YYYY-MM-DD'), 2600, 12);
INSERT INTO PRZYDZIALY (ID_PROJEKTU, ID_PRAC, ROLA, DATA_ROZPOCZECIA,
STAWKA_MIESIECZNA, LICZBA_MIESIECY) VALUES (200, 3, 'TECHNICZNA',
TO_DATE('2024-03-01', 'YYYY-MM-DD'), 1800, 3);
INSERT INTO PRZYDZIALY (ID_PROJEKTU, ID_PRAC, ROLA, DATA_ROZPOCZECIA,
STAWKA_MIESIECZNA, LICZBA_MIESIECY) VALUES (300, 4, 'KIEROWNICZA',
TO_DATE('2024-04-01', 'YYYY-MM-DD'), 3000, 6);
INSERT INTO PRZYDZIALY (ID_PROJEKTU, ID_PRAC, ROLA, DATA_ROZPOCZECIA,
STAWKA_MIESIECZNA, LICZBA_MIESIECY) VALUES (300, 5, 'ADMINISTRACYJNY',
TO_DATE('2024-05-01', 'YYYY-MM-DD'), 2100, 12);
INSERT INTO PRZYDZIALY (ID_PROJEKTU, ID_PRAC, ROLA, DATA_ROZPOCZECIA,
STAWKA_MIESIECZNA, LICZBA_MIESIECY) VALUES (400, 2, 'PROGRAMISTYCZNA',
TO_DATE('2024-06-01', 'YYYY-MM-DD'), 2200, 6);
INSERT INTO PRZYDZIALY (ID_PROJEKTU, ID_PRAC, ROLA, DATA_ROZPOCZECIA,
STAWKA_MIESIECZNA, LICZBA_MIESIECY) VALUES (100, 5, 'ADMINISTRACYJNY',
TO_DATE('2024-07-01', 'YYYY-MM-DD'), 2200, 4);

COMMIT;

--ZAD3
SELECT nazwisko,placa_miesieczna from PRACOWNICY
where stanowisko in ('admin','prog') and placa_miesieczna >2000;
--ZAD4
select p.rola, sum(p.stawka_miesieczna) as suma from przydzialy p
group by p.rola
having avg(p.stawka_miesieczna)>2000;
--ZAD5
SELECT
    P.NAZWISKO,
```

```
    PJ.NAZWA AS NAZWA_PROJEKTU,
    PR.ROLA
FROM PRZYDZIALY PR
JOIN PRACOWNICY1 P ON PR.ID_PRAC = P.ID_PRAC
JOIN PROJEKTY PJ ON PR.ID_PROJEKTU = PJ.ID_PROJEKTU
WHERE PR.LICZBA_MIESIECY >= 3
  AND PJ.BUDZET > 10000;
--ZAD6
SELECT PRA.NAZWISKO,NVL(LISTAGG(PRO.NAZWA,', ')WITHIN GROUP (ORDER BY PRO.NAZWA)
,'BRAK PROJEKTOW') FROM PROJEKTY PRO
LEFT JOIN PRACOWNICY PRA ON PRO.ID_OPIEKUNA = PRA.ID_PRAC;
--ZAD7
SELECT NAZWISKO FROM PRACOWNICY PRA
WHERE PRA.ID_PRAC NOT IN (
    SELECT PRZ.ID_PRAC
    FROM PRZYDZIALY PRZ
    WHERE PRZ.ROLA = 'TECHNICZNA'
);
--ZAD 8
SELECT PRA.NAZWISKO FROM PRACOWNICY PRA
JOIN PRZYDZIALY PRZ ON PRA.ID_PRAC = PRZ.ID_PRAC
GROUP by PRA.NAZWISKO
HAVING SUM(CASE WHEN PRZ.ROLA = 'TECHNICZNA' THEN PRZ.stawka_miesieczna ELSE 0 END)
> (SUM(PRZ.stawka_miesieczna)/2);
--ZAD 9
UPDATE PROJEKTY SET BUDZET = BUDZET *1.1
WHERE NAZWA = 'XML';
--ZAD10
UPDATE PRACOWNICY PRA
SET PRA.PPLACA_MIESIECZNA  = PRA.PLACA_MIESIECZNA + (0.1 *(
    SELECT AVG(PRA2.PLACA_MIESIECZNA)
    FROM PRACOWNICY PRA2
    WHERE PRA2.STANOWISKO = PRA.STANOWISKO
))
WHERE PRA.ID_PRAC IN (
    SELECT PRZ.ID_PRAC
    FROM PRZYDZIALY PRZ
    GROUP BY PRZ.ID_PRAC
    HAVING COUNT(DISTINCT PRZ.ID_PROJEKTU)>10
);
--ZAD 11
INSERT INTO PRZYDZIALY (ID_PRAC,ID_PROJEKTU,ROLA)
VALUES (100,250,'ADMIN');
--ZAD 12
DELETE FROM PRZYDZIALY PRZ
WHERE PRZ.ID_PRAC IN (
    SELECT PRA.ID_PRAC
    FROM PRACOWNICY PRA
    JOIN PROJEKTY PRO ON PRA.ID_PRAC = PRO.ID_OPIEKUNA
    WHERE PRA.STANOWISKO = 'ADMIN'
);
```

```
--Zad 13
CREATE TABLE PRZYDZIALY (
    ID_PROJEKTU NUMBER(4) CONSTRAINT FK_ID_PROJEKTU REFERENCES
PROJEKTY(ID_PROJEKTU),
    ID_PRAC NUMBER(4) CONSTRAINT FK_ID_PRAC REFERENCES PRACOWNICY(ID_PRAC),
    ROLA VARCHAR2(20) CHECK (ROLA IN ('ADMINISTRACYJNA', 'BADAWCZA',
'TECHNICZNA')),
    DATA_ROZPOCZECIA DATE DEFAULT CURRENT_DATE NOT NULL,
    STAWKA_MIESIECZNA NUMBER(6,2) CHECK(STAWKA_MIESIECZNA>0) ,
    LICZBA_MIESIECY NUMBER CHECK(LICZBA_MIESIECY>0),
    CONSTRAINT UK_PRZYDZIALY UNIQUE (ID_PROJEKTU, DATA_ROZPOCZECIA),
    CONSTRAINT PK_PRZYDZIAL PRIMARY KEY (ID_PROJEKTU, ID_PRAC, ROLA)
);
```

ZESTAW 2

```
--ZAD 1
select * from etaty
where nazwa != 'PROFESOR';
--ZAD 2
select nazwisko from artysci
where narodowosc = 'amerykanska' and stawka_wyjsciowa between 1000000 and
100000000000;
--ZAD 3
select typ_roli, avg(stawka_za_rola) from role
where liczba_dni > 10
group by typ_roli;
--ZAD 4
SELECT a.nazwisko,f,tytul,r.postac from artysci a
join role r on r.id_artysty = a.id_artysty
join filmy f on f.id_filmu = r.id_filmu
where r.typ_roli = 'pierwszoplanowa' and a.narodowosc not like 'polak';
--ZAD 5
select a.nazwisko ,NVL(listagg(f.tytul,', '), 'brak filmow') as filmy
from artysci a
left join role r on a.id_artysty = r.id_artysty
left join filmy f on r.id_filmu = f.id_filmu;
--ZAD 6
select a.nazwisko from artysci a
join role r on r.id_artysty = a.id_artysty
where NVL(r.rola,'brak') != 'pierwszoplanowa';
--ZAD 7
select a.nazwiska a from aktorzy a
join role r on r.id_artysty = a.id_artysty
group by a.nazwiska
```

```
having
count(r.rola)/2 < SUM(CASE WHEN r.rola = 'pp' then 1 else 0 end);
--ZAD 8
update artysci set stawka_wyjscia= stawka_wyjscia * 1.02
where nazwisko like 'A%';
--ZAD 9
update filmy f set budzet =(
    select 1.1 * sum(r.stawka_za_role)
    from role r
    where r.id_filmu = f.id_filmu)
where f.id_filmu in (
    select f.id_filmu
    from filmy f
    group by f.id_filmu
    having sum(f.stawka_za_role) > 20000
);
--ZAD 10
insert into role (id_filmu,id_artysty,postac)
VALUES(140,130,'SUPERMAN');
--ZAD 11
delete from role r where r.id_artysty in(
    select distinct a.id_artysty from artysci a
    where a.narodowosc = 'POLSKA!'
    AND EXISTS (SELECT 1 FROM FILMY F WHERE F.id_rezysera = A.id_artysty)
);
--ZAD 12
create table ROLE (
    id_filmu NUMBER NOT NULL,
    id_artysty NUMBER NOT NULL,
    postac VARCHAR2(100) NOT NULL,
    typ_roli VARCHAR2(20) CHECK (typ_roli in ('pp','dp','e')),
    stawka_za_role NUMBER CHECK (stawka_za_role >0) NOT NULL,
    liczba_dni NUMBER CHECK (liczba_dni >0),
    PRIMARY KEY (id_filmu,id_artysty,postac),
    UNIQUE (id_filmu,postac),
    CONSTRAINT fk_id_filmu FOREIGN KEY (id_filmu) REFERENCES FILMY(id_filmu),
    CONSTRAINT fk_id_artysty FOREIGN KEY (id_artysty) REFERENCES
ARTYSCI(id_artysty)
);
drop table role;
```

ZESTAW 3

```
--ZAD2
Create Table REZERWACJE (
```

```
    NR_REZERWACJI INT NOT NULL CHECK(NR_REZERWACJI <= 999999) ,
    ID_KLIENTA INT NOT NULL CHECK (ID_KLIENTA <= 999999),
    DATA_REZERWACJI DATE DEFAULT CURRENT_DATA NOT NULL,
    CENA DECIMAL(6,2) NOT NULL CHECK (CENA >= 0 AND CENA <= 99999.99),
    ZALICZKA DECIMAL(6,2) NOT NULL CHECK (ZALICZKA >= 0 AND ZALICZKA <= CENA),
    IMPREZA CHAR(10) NOT NULL,
    PRIMARY KEY (NR_REZERWACJI),
    CONSTRAINT REZ_FK_1 FOREIGN KEY (ID_KLIENTA) REFERENCES KLIENCI (ID_KLIENTA),
    CONSTRAINT REZ_FK_2 FOREIGN KEY (IMPREZA) REFERENCES IMPREZY (SYMBOL_IMPREZY),
    CONSTRAINT REZ_UK UNIQUE (ID_KLIENTA, IMPREZA)
    );
nie działa /sperka do zaliczki chte triggera
--ZAD3
SELECT SYMBOL_IMPREZY,KRAJ,MIASTO FROM IMPREZY
WHERE TYP_IMPREZY LIKE 'POBYT' and KONTYNET IN ('AZJA','EUROPA');

--ZAD4
SELECT K.NAZWISKO,K.IMIE,R.NR_REZERWACJI,I.KRAJ,I.MIASTO FROM REZERWACJE R
JOIN KLIENCI K ON R.ID_KLIENTA = K.ID_KLIENTA
JOIN IMPREZY I ON I.SYMBOL_IMPREZY = R.IMPREZA
WHERE R.DATA_REZERWACJI BETWEEN '2005-03-01' AND '2005-03-31' AND K.MIASTO LIKE
'POZNAN';

--ZAD5
SELECT I.SYMBOL_IMPREZY,I.MIASTO,COUNT(R.NR_REZERWACJI) AS LICZBA_REZERWACJI FROM
IMPREZY I
LEFT JOIN REZERWACJE R ON I.SYMBOL_IMPREZY = R.IMPREZA
GROUP BY
    I.SYMBOL_IMPREZY,
    I.MIASTO;

--ZAD6
SELECT K.IMIE,K.NAZWISKO,I.CENA_ZA_OSOBE FROM REZERWACJE R
LEFT JOIN KLIENCI K ON R.ID_KLIENTA = K.ID_KLIENTA
LEFT JOIN IMPREZY I ON I.SYMBOL_IMPREZY = R.IMPREZA
WHERE K.MIASTO LIKE 'WARSZAWA' AND I.KONTYNET LIKE 'EUROPA' AND I.CENA_ZA_OSOBE = (
    SELECT MAX(CENA_ZA_OSOBE) FROM IMPREZY
);
// też chuj wie za mało
--ZAD7
SELECT I.SYMBOL_IMPREZY, I.KRAJ FROM REZERWACJE R
LEFT JOIN IMPREZY I ON I.SYMBOL_IMPREZY = R.IMPREZA
WHERE
    I.TYP_IMPREZY = 'pobyt'
    AND R.NR_REZERWACJI IS NULL;

--ZAD8
SELECT K.NAZWISKO,K.IMIE FROM KLIENCI K
LEFT JOIN REZERWACJE R ON R.ID_KLIENTA = K.ID_KLIENTA
WHERE IMPREZA LIKE 'AZ2990'
ORDER BY R.ZALICZKA DESC
```

```
FETCH FIRST 5 ROWS ONLY;

--ZAD9 (JAKIES ZJEBANE CHYBA GIT IDK)
UPDATE REZERWACJE R
SET R.CENA = R.CENA * (1 - (
    SELECT LEAST(COUNT(*) * 5, 50) / 100.0
    FROM REZERWACJE R2004
    WHERE R2004.ID_KLIENTA = R.ID_KLIENTA
      AND EXTRACT(YEAR FROM R2004.DATA_REZERWACJI) = 2004
))
WHERE EXTRACT(YEAR FROM R.DATA_REZERWACJI) = 2005
  AND EXISTS (
    SELECT 1
    FROM REZERWACJE R2004
    WHERE R2004.ID_KLIENTA = R.ID_KLIENTA
      AND EXTRACT(YEAR FROM R2004.DATA_REZERWACJI) = 2004
);

--ZAD10
DELETE FROM REZERWACJE
WHERE ID_KLIENTA IN (
    SELECT ID_KLIENTA
    FROM KLIENCI
    WHERE MIASTO IN ('Warszawa', 'Kraków')
)
AND ZALICZKA < CENA;
```

kolos 4

```
CREATE TABLE FILMY (
    ID_FILMU        NUMBER(6)
        CONSTRAINT FILMY_PK PRIMARY KEY,    -- Klucz główny
    TYTUL           VARCHAR2(100) NOT NULL,                         -- Tytuł filmu
    REZYSER         VARCHAR2(100) NOT NULL,                        -- Reżyser filmu
    KOSZT_PRODUKCJI NUMBER(12, 2) CHECK ( KOSZT_PRODUKCJI > 0 ), -- Koszt produkcji
(dodatni)
    KATEGORIA       VARCHAR2(50) NOT NULL,                      -- Kategoria filmu
    DATA_PREMIERY   DATE NOT NULL                             -- Data premiery
);

CREATE TABLE KINA (
    SYMBOL_KINA     CHAR(10)
        CONSTRAINT KINA_PK PRIMARY KEY,  -- Klucz główny
    NAZWA           VARCHAR2(100) NOT NULL,                         -- Nazwa kina
    ADRES           VARCHAR2(200) NOT NULL,                         -- Adres kina
    MIASTO          VARCHAR2(50) NOT NULL,                         -- Miasto, w
którym znajduje się kino
    REZERW_INTERNET CHAR(1) CHECK ( REZERW_INTERNET IN ( 'T', 'N' ) ), -- Czy
obsługuje rezerwacje internetowe ('T' - tak, 'N' - nie)
```

```
    DOTACJA         NUMBER(10, 2)                                   -- Dotacja dla
kina (opcjonalna)
);

--2 ZAD
CREATE TABLE SPRZEDAZ_BILETOW (
    NR_TRANSAKCJI        NUMBER(6)
        CONSTRAINT SB_PK PRIMARY KEY,
    FILM                 NUMBER(6) NOT NULL,
    SYMBOL_KINA          CHAR(10) NOT NULL,
    DATA_TRANSAKCJI      DATE DEFAULT SYSDATE NOT NULL,
    ILOSC                NUMBER(6) NOT NULL,
    CENA_JEDNOSTK_BILETU NUMBER(5, 2) CHECK ( CENA_JEDNOSTK_BILETU > 0 ) NOT NULL,
    ADNOTACJA            VARCHAR2(200),
    CONSTRAINT SB_FK_1 FOREIGN KEY ( FILM )
        REFERENCES FILMY ( ID_FILMU ),
    CONSTRAINT SB_FK_2 FOREIGN KEY ( SYMBOL_KINA )
        REFERENCES KINA ( SYMBOL_KINA ),
    CONSTRAINT SB_UK UNIQUE ( FILM,
                              SYMBOL_KINA )
);

--3 ZAD

SELECT
    NAZWA,
    ADRES
FROM
    KINA
WHERE
    REZERW_INTERNET = 'T';

--4 ZAD

SELECT
    F.TYTUL,
    S.ILOSC,
    K.NAZWA
FROM
         SPRZEDAZ_BILETOW S
    JOIN FILMY F ON F.ID_FILMU = S.FILM
    JOIN KINA  K ON K.SYMBOL_KINA = S.SYMBOL_KINA
WHERE
    EXTRACT(YEAR FROM F.DATA_PREMIERY) = 2005;

-- ZAD 5
SELECT
    K.SYMBOL_KINA,
    K.MIASTO,
    NVL(SUM(S.ILOSC),
        0) AS SUMA_BILETOW
```

```
FROM
    KINA             K
    LEFT JOIN SPRZEDAZ_BILETOW S ON S.SYMBOL_KINA = K.SYMBOL_KINA
GROUP BY
    K.SYMBOL_KINA,
    K.MIASTO;
-- ZAD 6
```

Zad. 6 (10 pkt)
Podaj tytuły filmów z kategorii dramat, które były wyświetlane w kinie w Warszawie, otrzymującym najwyższą dotację wśród wszystkich kin.

```
SELECT F.TYTUL FROM FILMY F
JOIN SPRZEDAZ_BILETOW S ON F.ID_FILMU = S.FILM
JOIN KINA K ON K.SYMBOL_KINA= S.SYMBOL_KINA
WHERE F.KATEGORIA='DRAMAT' AND K.MIASTO='WARSZAWA'
ORDER BY K.DOTACJA DESC
FETCH FIRST 1 ROW ONLY;

SELECT DISTINCT F.TYTUL
FROM FILMY F
JOIN SPRZEDAZ_BILETOW S ON F.ID_FILMU = S.FILM
JOIN KINA K ON S.SYMBOL_KINA = K.SYMBOL_KINA
WHERE F.KATEGORIA = 'dramat'
  AND K.MIASTO = 'Warszawa'
  AND K.DOTACJA = (
    SELECT MAX(DOTACJA)
    FROM KINA  );



-- ZAD7
SELECT F.TYTUL FROM FILMY F
JOIN SPRZEDAZ_BILETOW S ON F.ID_FILMU= S.FILM
GROUP BY F.ID_FILMU , F.TYTUL , F.KOSZT_PRODUKCJI
HAVING F.KOSZT_PRODUKCJI > 2* SUM(S.CENA_JEDNOSTK_BILETU* S.ILOSC);

-- ZAD8
SELECT F.TYTUL FROM FILMY F
JOIN SPRZEDAZ_BILETOW S ON F.ID_FILMU = S.FILM
WHERE F.KATEGORIA = 'SENSACYJNY'
ORDER BY S.CENA_JEDNOSTK_BILETU DESC
FETCH FIRST 3 ROWS ONLY;

-- ZAD 9
UPDATE KINA K
SET DOTACJA = DOTACJA + (0.1 * (SELECT AVG(DOTACJA) FROM KINA))
WHERE K.SYMBOL_KINA IN (
    SELECT S.SYMBOL_KINA
    FROM SPRZEDAZ_BILETOW S
    WHERE S.DATA_TRANSAKCJI BETWEEN TO_DATE('2004-01-01', 'YYYY-MM-DD')
                                AND TO_DATE('2004-12-31', 'YYYY-MM-DD')
```

```sql
    GROUP BY S.SYMBOL_KINA
    HAVING SUM(S.CENA_JEDNOSTK_BILETU * S.ILOSC) >= 10000
);

-- 10 ZAD
DELETE FROM SPRZEDAZ_BILETOW S
WHERE S.FILM IN (
    SELECT F.ID_FILMU
    FROM FILMY F
    WHERE EXTRACT(YEAR FROM F.DATA_PREMIERY) = 2005
      AND EXTRACT(MONTH FROM F.DATA_PREMIERY) = 4
) AND S.SYMBOL_KINA IN (
    SELECT K.SYMBOL_KINA
    FROM KINA K
    WHERE K.MIASTO IN ('Poznań', 'Katowice')
);
```

SQL Cheat Sheet

DQL

```
with
  <name> as (<select>), ...
select
  [distinct] <expr> [ as <name> ], ...
from
  {<table> <join>}
where
  <cond>
group by
  <name> [asc | desc], ...
having
  <cond>
order by
  <name> [asc | desc] [nulls {last | first}], ...
fetch first <n> rows with ties
```

Joins

```
join := [ inner | {full | left | right [outer] ] join <table> on {
<cond> | using (<name>, ...) }
join := natural [ inner | {full | left | right} [outer] ] join <table>
```

Aggregation

```
avg, count, max, min, sum, variance, stddev
count(*)
count([all | distinct] <attr>)
```

Sets

```
<select> {union | intersect | minus} <select>
```

DML

```
insert into <table> (<attr>, ...) values (<expr>, ...)
delete from <table> where <cond>
update <table> set (<attr>, ...) = (<select>) where <cond>
```

DDL

```
create table <table> (
  <attr> <type> [default <expr>] [[constraint <name>] <constr>]
[generated always as identity],
  ...
  [constraint <name>] <constr>,
  ...
)
```

```
alter table <table> {enable | disable} constraint <name>;
alter table <table> add (...) modify (...);
drop table <table> [cascade constraints];
describe <table>;
```

Constraints

```
<constr>
  | primary key(<attr>, ...) [on delete cascade]
  | [foreign key <attr>] references <table>(<attr>, ...) [on delete
cascade]
```

```
  | unique(<attr>)
  | check(<expr>)
  | not null
```

Views

```
create [or replace] view <view> (<attr>, ...)
as <select>
[with {check option [constraint <name>] | read only}];
```

```
drop view <view>;
```

Library

Tables

```
user_constraints: (table_name, constraint_name, constraint_type,
search_condition)
user_updatable_columns: (table_name, column_name, updatable,
insertable, deletable)
user_tables: (table_name)
dual: (dummy)
```

Switch

```
case [<expr>]
when <expr> then <expr>
when <expr> then <expr>
else <expr>
end
```

Functions

```
-- Operators
T in [T]
T between T and T
Str like Str
T? is null
date 'YYYY-MM-DD'
```

```
timestamp 'YYYY-MM-DD HH:MI:SS.MMMM'
interval 'REPR' _ [ to _ ]
listagg(<attr>, Str) within group (order by <attr>, ...)

-- Optional
coalesce/nvl(T?, T): T

-- Strings
length(Str): Int
replace(Str, sub: Str, new: Str): Str
substr(Str, start: Int, length: Int): Str
instr(Str, sub: Str, start: Int, count: Int): Int
upper/lower(Str): Str
translate(Str, old: Str, new: Str): Str
trim([leading|trailing|both] Str from Str): Str
lpad/rpad(Str, length: Int, chars: Str): Str

-- Numbers
round/trunc(Num, Int?): Num
floor/ceil(Num): Num
power/mod(Num, Num): Num
sqrt/abs/sign(Num): Num
greatest/least(Num...): Num

-- Date & Time
extract([year|month|day|hour|minute|second] from Date)
current_date/current_time/current_timestamp
months_between(Date, Date): Int
add_months(Date, Int): Date
next_day(Date, ): Date
last_day(Date): Date

-- Conversions
cast(Any as Type)
to_char(Date, fmt: Str): Str
to_char(Num, fmt: Str): Str
to_number(Str, fmt: Str): Num
to_date(Str, fmt: Str): Date

-- Oracle Shit
decode(value, c1, r1, ..., ci, ri, other)
```

Formatting

Date & Time
```
SCC BC AD - Century, Era
D DD DDD DAY fmDAY - Day
MM MON MONTH fmMONTH - Month
YYYY - Year
```

```
HH HH24 - Hour
MI - Minute
SS - Second
AM PM - Time of day
```

Number
```
D - fraction separator
G - group separator
9 - digit
0 - zero padding
```

Nulls

- Arithmetic operations with NULL return NULL
- Boolean comparisons with NULL return UNKNOWN
- `count(*)` counts all NULL and non-NULL tuples
- `count(attr)` counts all tuples where attr is not NULL
- Other aggregate functions ignore NULL