LISTA 1

1. Dla każdego departamentu wyświetl liczbę zatrudnionych w nim pracowników i ich średnią z ich minimalnego wynagrodzenia.

```
SELECT DEP.DEPARTMENT_ID, COUNT(EMP.EMPLOYEE_ID), AVG(J.MIN_SALARY)
FROM DEPARTMENTS DEP LEFT JOIN EMPLOYEES EMP
ON DEP.DEPARTMENT_ID = EMP.DEPARTMENT_ID
LEFT JOIN JOBS J ON EMP.JOB_ID = J.JOB_ID
GROUP BY (DEP.DEPARTMENT_ID)
ORDER BY (DEP.DEPARTMENT_ID);
```

2. Dla każdego pracownika wyświetl jego kategorię płacową i widełki płacowe w jakich mieści się pensja pracownika.

```
SELECT EMP.FIRST_NAME, EMP.LAST_NAME,J.JOB_TITLE, EMP.SALARY,J.MIN_SALARY,
J.MAX_SALARY, J2.JOB_TITLE AS STANOWISKO,
J2.MIN_SALARY AS PLACA_MIN, J2.MAX_SALARY AS PLACA_MAX
FROM EMPLOYEES EMP, JOBS J, JOBS J2
WHERE EMP.JOB_ID = J.JOB_ID AND EMP.SALARY BETWEEN J2.MIN_SALARY AND J2.MAX_SALARY
ORDER BY EMP.LAST_NAME;
```

3. Dla każdego pracownika posiadającego podwładnych wyświetl ich liczbę. Wyniki posortuj zgodnie z malejącą liczbą podwładnych.

```
SELECT EMP.FIRST_NAME, EMP.LAST_NAME, EMP.EMPLOYEE_ID, COUNT(*) AS ILOSC_PODWLADNYCH
FROM EMPLOYEES EMP2
JOIN EMPLOYEES EMP ON EMP.EMPLOYEE_ID=EMP2.MANAGER_ID
GROUP BY EMP.FIRST_NAME, EMP.LAST_NAME, EMP.EMPLOYEE_ID
ORDER BY COUNT(*) DESC;
```

4. Wyświetl numer departamentu, który nie zatrudnia żadnych pracowników.

```
SELECT DISTINCT DEP.DEPARTMENT_ID
FROM DEPARTMENTS DEP
WHERE DEP.MANAGER_ID IS NULL
ORDER BY DEP.DEPARTMENT_ID ASC;
```

5. Wyświetl dla każdego roku liczbę zatrudnionych w nim pracowników. Wynik uporządkuj zgodnie z liczbą zatrudnionych.

```
SELECT DISTINCT EXTRACT(YEAR FROM EMP.HIRE_DATE) AS HIRE_YEAR
FROM EMPLOYEES EMP
ORDER BY HIRE_YEAR DESC;
```

6. Wyświetl id departamentu, nazwę departamentu, nazwisko, wynagrodzenie, dla tych pracowników, którzy zarabiają mniej niż średnia płaca dla ich etatu.

```
SELECT EMP.FIRST_NAME, EMP.LAST_NAME, EMP.EMPLOYEE_ID, COUNT(*) AS ILOSC_PODWLADNYCH,
SUM(EMP2.SALARY) AS SUMA_WYNAGRODZ, AVG(EMP2.SALARY) AS SR_WYNAGRODZ,
SUM(EMP2.SALARY)/COUNT(*) AS SR
FROM EMPLOYEES EMP2
JOIN EMPLOYEES EMP ON EMP.EMPLOYEE_ID=EMP2.MANAGER_ID
GROUP BY EMP.FIRST_NAME, EMP.LAST_NAME, EMP.EMPLOYEE_ID
ORDER BY COUNT(*) DESC;
```

LISTA 2 FUNKCJE AGREGUJĄCE

1. Zbuduj zapytanie, które znajdzie liczbę wszystkich klientów (tabela CUSTOMERS).

```
SELECT COUNT(*)
FROM CUSTOMERS;
```

2. Zmodyfikuj powyższe zapytanie w taki sposób, aby otrzymać liczbę klientów z podziałem na płeć (atrybut gender). Wynik posortuj wg płci.

```
SELECT GENDER, COUNT(*)
FROM CUSTOMERS
GROUP BY GENDER
ORDER BY GENDER;
```

3. Policz, ile zamówień realizowali klienci w roku 2007 (tabela ORDERS).

```
SELECT COUNT(*)
FROM ORDERS
WHERE EXTRACT(year from ORDER_DATE) = 2007;
```

4. Rozszerz poprzednie zapytanie w taki sposób, aby otrzymać liczbę zamówień w roku 2007 z podziałem na tryb zamówienia: direct lub online.

```
SELECT ORDER_MODE, COUNT(*)
FROM ORDERS
WHERE EXTRACT(year from ORDER_DATE) = 2007
GROUP BY ORDER_MODE;
```

5. Ogranicz analizy zamówień do zamówień online. Poznaj rozkład statusów w tym trybie (rozkład statusów – liczba wystąpień danego statusu zamówienia).

```
SELECT ORDER_MODE, ORDER_STATUS, COUNT(*)
FROM ORDERS
WHERE EXTRACT(year from ORDER_DATE) = 2007 AND ORDER_MODE LIKE 'online'
GROUP BY ORDER_MODE, ORDER_STATUS
ORDER BY ORDER_STATUS ASC;
```

6. Rozszerz poprzednie zapytanie w taki sposób aby pominąć rekordy o statusie=0.

```
SELECT ORDER_MODE, ORDER_STATUS, COUNT(*)
FROM ORDERS
WHERE EXTRACT(year from ORDER_DATE) = 2007 AND ORDER_MODE LIKE 'online'
AND NOT ORDER_STATUS = 0
GROUP BY ORDER_MODE, ORDER_STATUS
ORDER BY ORDER_STATUS ASC;
```

7. Tym razem podziel zbiór zamówień online na trzy kategorie: "entered": 0 i 1, "canceled": 2 oraz 3 i "shipped": 4,5,6,7,8,9,10. Znajdź liczbę zamówień w każdej kategorii. Wykorzystaj konstrukcję CASE.

```
SELECT COUNT(*),
CASE
        WHEN ORDER_STATUS IN (0,1) THEN 'ENTERED'
        WHEN ORDER_STATUS IN (2,3) THEN 'CANCELD'
        ELSE 'SHIPPED'
END AS KATEGORIA
FROM ORDERS
        WHERE EXTRACT(year from ORDER_DATE) = 2007 AND ORDER_MODE LIKE 'online'
GROUP BY
CASE
        WHEN ORDER_STATUS IN (0,1) THEN 'ENTERED'
        WHEN ORDER_STATUS IN (2,3) THEN 'CANCELD'
        ELSE 'SHIPPED'
END;
```

8. Policz, ilu klientów przypada na poszczególnych pracowników. Posortuj wynik wg malejącej liczby klientów.

```
SELECT EMPLOYEES.FIRST_NAME, EMPLOYEES.LAST_NAME, CUSTOMERS.ACCOUNT_MGR_ID,
COUNT(CUSTOMERS.ACCOUNT_MGR_ID)
FROM EMPLOYEES LEFT JOIN CUSTOMERS
ON EMPLOYEES.EMPLOYEE_ID = CUSTOMERS.ACCOUNT_MGR_ID
GROUP BY EMPLOYEES.FIRST_NAME, EMPLOYEES.LAST_NAME, CUSTOMERS.ACCOUNT_MGR_ID
ORDER BY CUSTOMERS.ACCOUNT_MGR_ID ASC;
```

9. Rozbuduj poprzednie zapytanie w taki sposób, aby pominąć pracowników, którzy obsługują mniej niż 70 osób (użyj w zapytaniu klauzuli HAVING).

```
SELECT EMPLOYEES.EMPLOYEE_ID, CUSTOMERS.ACCOUNT_MGR_ID, COUNT(CUSTOMERS.ACCOUNT_MGR_ID)
FROM EMPLOYEES LEFT JOIN CUSTOMERS
ON EMPLOYEES.EMPLOYEE_ID=CUSTOMERS.ACCOUNT_MGR_ID
GROUP BY EMPLOYEES.EMPLOYEE_ID, CUSTOMERS.ACCOUNT_MGR_ID
HAVING COUNT(CUSTOMERS.ACCOUNT_MGR_ID)>=70
ORDER BY CUSTOMERS.ACCOUNT_MGR_ID ASC;
```

LISTA 2 PÓŁKOSTKA I KOSTKA

1. Zbuduj zapytanie, które ponownie znajdzie liczbę klientów z podziałem na płcie. Tym razem w wyniku ma się pojawić wiersz z całkowitą liczbą klientów (użyj klauzuli ROLLUP).

```
SELECT GENDER, COUNT(*)
FROM CUSTOMERS
GROUP BY ROLLUP(GENDER)
ORDER BY GENDER;
```

2. Zmodyfikuj zapytanie z punktu 1. tak, aby w wierszu podsumowania pojawił się napis "wszyscy" (użyj funkcji GROUPING z konstrukcją CAS).

```
SELECT GENDER,
CASE WHEN GROUPING(GENDER)=1 THEN 'WSZYSCY'
END AS GR,
COUNT(*)
FROM CUSTOMERS
GROUP BY ROLLUP(GENDER)
ORDER BY GENDER;
```

3. Podaj liczby zamówień, realizowane w kolejnych latach przez klientów, w rozbiciu na lata. Wyświetl również podsumowanie zawierające całkowitą liczbę zamówień (użyj klauzuli ROLLUP).

```
SELECT EXTRACT(YEAR FROM ORDER_DATE) AS YEAR ,COUNT(*)
FROM ORDERS
GROUP BY ROLLUP(EXTRACT(YEAR FROM ORDER_DATE));
```

4. Rozszerz powyższe zapytanie o dodatkowy poziom sumowania – wyświetl również liczbę zamówień realizowanych w poszczególnych miesiącach.

```
SELECT EXTRACT(YEAR FROM ORDER_DATE) AS YEAR,
EXTRACT(MONTH FROM ORDER_DATE) AS MONTH,
COUNT(*)
FROM ORDERS
GROUP BY ROLLUP(EXTRACT(YEAR FROM ORDER_DATE),
EXTRACT(MONTH FROM ORDER_DATE));
```

5. Zmodyfikuj powyższy wynik, w taki sposób, aby oznaczyć podsumowania tekstem „Miesiące razem”.

```
SELECT EXTRACT(YEAR FROM ORDER_DATE) AS YEAR,
EXTRACT(MONTH FROM ORDER_DATE) AS MONTH,
CASE GROUPING_ID(EXTRACT(YEAR FROM ORDER_DATE),EXTRACT(MONTH FROM ORDER_DATE))
        WHEN 1 THEN 'Miesiace razem'
        WHEN 3 THEN 'Lata razem'
        END AS GR,
        COUNT(*)
FROM ORDERS
GROUP BY ROLLUP(EXTRACT(YEAR FROM ORDER_DATE),
EXTRACT(MONTH FROM ORDER_DATE));
```

6. Zbuduj zapytanie, które wyliczy, ilu klientów znajduje się w każdym regionie, w każdym kraju, i w każdym mieście. Dołącz do wyniku również podsumowanie zawierające całkowitą liczbę klientów (użyj klauzuli ROLLUP).

```
SELECT C.CUST_ADDRESS.CITY AS CITY,
C.CUST_ADDRESS.STATE_PROVINCE AS PROVINCE,
C.CUST_ADDRESS.COUNTRY_ID AS COUNTRY_ID,
COUNT(*) AS COUNT
FROM CUSTOMERS C
GROUP BY ROLLUP(C.CUST_ADDRESS.COUNTRY_ID,
C.CUST_ADDRESS.STATE_PROVINCE,
C.CUST_ADDRESS.CITY);

/*V2
SELECT REGIONS.REGION_NAME AS REGION,
CC.COUNTRY_NAME AS KRAJ,
COUNT(*) AS LICZBA_KLIENTOW
FROM CUSTOMERS C JOIN COUNTRIES CC
ON C.CUST_ADDRESS.COUNTRY_ID = TO_CHAR(CC.COUNTRY_ID)
JOIN REGIONS ON CC.REGION_ID = REGIONS.REGION_ID
GROUP BY ROLLUP(REGIONS.REGION_NAME,CC.COUNTRY_NAME, C.CUST_ADDRESS.CITY);
```

7. Zbuduj zapytanie, które pokaże, ilu klientów z podziałem na płcie pochodzi z poszczególnych krajów (zastosuj zwykłe grupowanie).

```
SELECT NLS_TERRITORY, GENDER, COUNT(*)
FROM CUSTOMERS
GROUP BY NLS_TERRITORY,GENDER
ORDER BY NLS_TERRITORY;
```

8. Rozszerz polecenie z poprzedniego punktu w ten sposób, aby w wyniku otrzymać również podsumowanie liczby klientów w każdym kraju bez podziału na płcie, podsumowanie każdej płci bez względu na kraj oraz podsumowanie całkowite. Wynik będzie pełną kostką danych.

```
SELECT NLS_TERRITORY, GENDER, GROUPING(NLS_TERRITORY, GENDER) AS G1, COUNT(*)
FROM CUSTOMERS
GROUP BY CUBE(NLS_TERRITORY,GENDER);

/*v2
SELECT CC.COUNTRY_NAME, C.GENDER, COUNT(*)
FROM CUSTOMERS C JOIN COUNTRIES CC
ON C.CUST_ADDRESS.COUNTRY_ID = TO_CHAR(CC.COUNTRY_ID)
GROUP BY CUBE(CC.COUNTRY_NAME,C.GENDER);
```

Lista 2 Transformacja danych wierszowych do układu kolumnowego

1. Zbuduj zapytanie, które pokaże, ilu klientów z podziałem na płcie pochodzi z poszczególnych krajów (zastosuj zwykłe grupowanie).

```
SELECT NLS_TERRITORY, GENDER, COUNT(*)
FROM CUSTOMERS
GROUP BY (NLS_TERRITORY,GENDER)
ORDER BY NLS_TERRITORY;
```

2. Przekształć powyższy wynik w taki sposób, aby wyliczenia liczby klientów w poszczególnych krajach pojawiły się w dwóch kolumnach: jednej dla kobiet, drugiej dla mężczyzn (użyj klauzuli PIVOT).

```
SELECT * FROM
(SELECT C.CUST_ADDRESS.COUNTRY_ID AS KRAJ, C.GENDER GEN, CCUSTOMER_ID C_ID
        FROM CUSTOMERS C)
        PIVOT (COUNT(C_ID)
        FOR GEN
        IN('F' FEMALE, 'M' MALE)
 )
ORDER BY KRAJ;
```

3. Zbuduj zapytanie, które tym razem pokaże w kolejnych kolumnach, osobno dla kobiet i mężczyzn, średnie kwotę wydaną na zamówienia w podziale na pracowników obsługujących klientów

```
SELECT * FROM
(SELECT O.SALES_REP_ID S_ID, CC.GENDER GEN, O.ORDER_TOTAL
        FROM CUSTOMERS CC JOIN ORDERS O
        ON CC.CUSTOMER_ID=O.CUSTOMER_ID)
        PIVOT (AVG(ORDER_TOTAL)
        FOR GEN
        IN ('F' FEMALE,'M' MALE)
)
ORDER BY S_ID;
```

4. Przekształć powyższe zapytanie w taki sposób, aby średnie kwota były prezentowana z dokładnością do dwóch miejsc po przecinku.

```
SELECT S_ID,ROUND(FEMALE,2),ROUND(MALE,2) FROM
(SELECT O.SALES_REP_ID S_ID, CC.GENDER GEN, O.ORDER_TOTAL
        FROM CUSTOMERS CC JOIN ORDERS O
        ON CC.CUSTOMER_ID=O.CUSTOMER_ID)
        PIVOT (AVG(ORDER_TOTAL)
        FOR GEN
        IN ('F' FEMALE,'M' MALE)
)
ORDER BY S_ID;
```

5. Z powyższego zapytania utwórz perspektywę o nazwie SREDNIE_PIVOT. Sprawdź, jakie dane udostępnia perspektywa.

```
SELECT S_ID,ROUND(FEMALE,2),ROUND(MALE,2) FROM
(SELECT O.SALES_REP_ID S_ID, CC.GENDER GEN, O.ORDER_TOTAL
        FROM CUSTOMERS CC JOIN ORDERS O
        ON CC.CUSTOMER_ID=O.CUSTOMER_ID)
        PIVOT (AVG(ORDER_TOTAL)
        FOR GEN
        IN ('F' FEMALE,'M' MALE)
 )
ORDER BY S_ID;
```

6. Zbuduj zapytanie z klauzulą UNPIVOT, które przekształci dane perspektywy SREDNIE_PIVOT do układu wierszowego.

```
SELECT * FROM(
(SELECT  S_ID, FEMALE, MALE FROM SREDNIE_PIVOT_5 SP)
        UNPIVOT( (SREDNIA)
        FOR GENDER3 IN (FEMALE AS 'FEMALE3', MALE AS 'MALE3')
));
```

LISTA 3 FUNKCJE ANALITYCZNE

1. Zbuduj zapytanie, które dla każdej podkategorii znajdzie liczbę produktów do niej należących. Następnie utwórz ranking podkategorii ze względu liczbę na produktów.

```
SELECT PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID, COUNT(*),
RANK() OVER(ORDER BY COUNT(*) DESC) RANKING
FROM H_PRODUCTS
GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID);
```

2. Zmodyfikuj zapytanie z p. 1 w taki sposób, aby w zbiorze wynikowym pojawiła się dodatkowa kolumna pokazująca ranking gęsty. Czy występują różnice pomiędzy rankingami?

```
SELECT PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID, COUNT(*),
RANK() OVER(ORDER BY COUNT(*) DESC) RANKING,
DENSE_RANK() OVER(ORDER BY COUNT(*) DESC) RANKING_DENSE
FROM H_PRODUCTS
GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID);
```

3. Zmodyfikuj zapytanie z poprzedniego punktu w taki sposób, aby otrzymać dane jedynie trzech pierwszych podkategorii w rankingu (weź pod uwagę ranking zwykły).

```
SELECT * FROM (
 SELECT PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID, COUNT(*),
 RANK() OVER(ORDER BY COUNT(*) DESC) RANKING,
 DENSE_RANK() OVER(ORDER BY COUNT(*) DESC) RANKING_DENSE
 FROM H_PRODUCTS
 GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID))
WHERE RANKING < 4;
```

4. Dokonaj kolejnej modyfikacji zapytania, tym razem chcemy uzyskać informacje o pięciu najmniej licznych podkategoriach (ponownie użyj zwykłego rankingu).

```
SELECT * FROM (
 SELECT PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID, COUNT(*),
 RANK() OVER(ORDER BY COUNT(*) ASC) RANKING
 FROM H_PRODUCTS
 GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID))
WHERE RANKING < 5;
```

5. Przekształć ranking, uzyskany w zadaniu 1., w ranking procentowy (użyj funkcji PERCENT_RANK).
Ogranicz wynik do dwóch pozycji po przecinku.

```
SELECT PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID, COUNT(*),
ROUND(PERCENT_RANK() OVER(ORDER BY COUNT(*) DESC),2) RANKING
FROM H_PRODUCTS
GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID);
```

6. Zmodyfikuj zapytanie z punktu poprzedniego w taki sposób, aby otrzymać informacje o podkategoriach, które lokują się w 25% najliczniej obsadzonych podkategorii.

```
WITH RANKING_TAB AS (
        SELECT PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID, COUNT(*),
        ROUND(PERCENT_RANK() OVER(ORDER BY COUNT(*) DESC),2) AS RANKING
 FROM H_PRODUCTS
 GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID))
SELECT * FROM RANKING_TAB
WHERE RANKING < 0.25;
```

7. Dodaj do wyniku zadania 6. kolumnę wyliczającą percentyle (funkcja CUME_DIST). Porównaj wyniki uzyskane w kolumnach RANKING_PROC i PERCENTYL.

```
WITH RANKING_TAB AS (
        SELECT PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID, COUNT(*),
        ROUND(PERCENT_RANK() OVER(ORDER BY COUNT(*) DESC),2) AS RANKING,
        CUME_DIST() OVER(ORDER BY COUNT(*) DESC) AS PERCENTYL
        FROM H_PRODUCTS
        GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID))
SELECT * FROM RANKING_TAB
WHERE RANKING < 0.25;
```

8. Podaj hipotetyczną pozycję w rankingu podkategorii, która zawiera dokładnie 9 produktów. Użyj rankingu zwykłego.

```
SELECT PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID, COUNT(*),
RANK() OVER(ORDER BY COUNT(*) DESC) RANKING
FROM H_PRODUCTS
GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID);
SELECT RANK(9) WITHIN GROUP
        (ORDER BY COUNT(*) DESC) AS POSITION
FROM H_PRODUCTS
GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID);
```

9. Przydziel każdej pozycji w rankingu podkategorii z punktu 1. unikalny numer porządkowy (wykorzystaj funkcję ROW_NUMBER). Porównaj numer porządkowy rekordu z pozycją w rankingu.

```
SELECT PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID, COUNT(*),
RANK() OVER(ORDER BY COUNT(*) DESC) RANKING,
ROW_NUMBER() OVER(ORDER BY COUNT(*) DESC) AS ROW_NUMBER
FROM H_PRODUCTS
GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID);
```

10. Podziel podkategorie na cztery "koszyki" w zależności od ich pozycji w rankingu zbudowanym wg liczby produktów. W każdym koszyku powinno znaleźć się tyle samo podkategorii (liczby podkategorii w poszczególnych koszykach mogą się różnić o co najwyżej 1).

```
SELECT PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID, COUNT(*),
RANK() OVER(ORDER BY COUNT(*) DESC) RANKING,
NTILE(4) OVER(ORDER BY COUNT(*) DESC) BUCKET
FROM H_PRODUCTS
GROUP BY (PROD_SUBCATEGORY,PROD_SUBCATEGORY_ID);
```