1. TEORIE

Funcţii SQL

➔ Sunt funcţii predefinite în sistemul Oracle.

➔ Pot fi clasificate în:
1. Funcții *single-row*
2. Funcții *multiple-row*

1. **Funcțiile *single row*** (laboratorul trecut)

2. **Funcţiile *multiple row (agregat)***

➔ pot fi utilizate pentru a returna informația corespunzătoare fiecăruia dintre grupurile obținute în urma divizării liniilor tabelului cu ajutorul clauzei GROUP BY

Clauza GROUP BY

➔ utilizată pentru a diviza datele identice în grupuri

```
SELECT nume_coloana, functie_agregat(nume_coloana)
FROM nume_tabel
GROUP BY nume_coloana;
```

OBS: Toate expresiile din clauza SELECT, cu excepția funcțiilor de agregare, se trec în clauza GROUP BY.

➔ pot apărea în clauzele:
- ❖ SELECT
- ❖ ORDER BY
- ❖ HAVING

➔ Server-ul Oracle aplică aceste funcții fiecărui grup de linii şi returnează un singur rezultat pentru fiecare mulțime

➔ Dintre funcțiile grup definite în sistemul Oracle, se pot enumera:

1. `AVG (nume_coloana)` - returnează valoarea medie a unei coloane numerice
2. `SUM (nume_coloana)` - returnează suma totală a unei coloane numerice
3. `MAX (nume_coloana)` - returnează valoarea maximă din coloană
4. `MIN (nume_coloana)` - returnează valoarea minimă din coloană
5. `COUNT (nume_coloana)` - returnează numarul de linii care respectă un anumit criteriu
6. `VARIANCE (nume_coloana)` - returnează dispersia valorilor
7. `STDDEV (nume_coloana)` -returnează deviația standard a valorilor

Observații:

- Funcțiile AVG, SUM, STDDEV şi VARIANCE operează numai asupra valorilor numerice.
- Funcțiile MAX şi MIN pot opera asupra valorilor numerice, caracter sau dată calendaristică.
- Absența clauzei GROUP BY conduce la aplicarea funcției grup pe mulțimea tuturor liniilor tabelului.
- Toate funcțiile grup, cu excepția lui COUNT(*), ignoră valorile null. COUNT(expresie) returnează numărul de linii pentru care expresia dată nu are valoarea null.

  Funcția COUNT returnează un număr mai mare sau egal cu zero şi nu întoarce niciodată valoarea null.
- Când este utilizată clauza GROUP BY, server-ul sortează implicit mulțimea rezultată în ordinea crescătoare a valorilor coloanelor după care se realizează gruparea.

Exemplu:

Considerăm următorul tabel:

CUSTOMERS

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Ramesh	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	kaushik	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Dorim să afişăm salariul total al fiecărui client, astfel:

```
SELECT name, SUM(salary)
FROM customers
GROUP BY name;
```

Rezultatul obţinut este următorul:

NAME	SUM(SALARY)
Hardik	8500.00
kaushik	8500.00
Komal	4500.00
Muffy	10000.00
Ramesh	3500.00

Clauza HAVING

- ➔ permite restricționarea grupurilor de linii returnate, la cele care îndeplinesc o anumită condiție
- ➔ a fost adăugată în SQL deoarece clauza WHERE nu poate conține funcții agregat
- ➔ Așadar, clauza WHERE pune condiția pentru coloanele selectate, în timp ce HAVING pune condiția pentru grupurile create de GROUP BY

```
SELECT nume_coloana, functie_agregat(nume_coloana)
FROM nume_tabel
WHERE nume_coloana operator valoare
GROUP BY nume_coloana
HAVING functie_agregat(nume_coloana) operator valoare;
```

Exemplu:

Considerăm următorul tabel:

CUSTOMERS

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Dorim să grupăm clienții după vârstă și să afișăm doar acea vârstă care apare de cel puțin 2 ori:

```
SELECT age
FROM customers
GROUP BY age
HAVING COUNT(age) >= 2;
```

Rezultatul obținut este următorul:

AGE
25

Sursă exemple: https://www.tutorialspoint.com

Operatorii ROLLUP, CUBE şi GROUPING SETS

➔ extensii ale clauzei GROUP BY

1. ROLLUP

➔ creează grupări prin deplasarea într-o singură direcție, de la dreapta la stânga, de-a lungul listei de coloane specificate în clauza GROUP BY. Apoi, se aplică funcția agregat acestor grupări.

```
SELECT nume_coloana_1, nume_coloana_2,...nume_coloana_n
FROM nume_tabel
GROUP BY ROLLUP(nume_coloana_1, nume_coloana_2,..., nume_coloana_n)
```

Datele vor fi grupate astfel:

P_1: nume_coloana_1, nume_coloana_2,..., nume_coloana_n

P_2: nume_coloana_1, nume_coloana_2,..., nume_coloana_n-1

P_3: nume_coloana_1, nume_coloana_2,..., nume_coloana_n-2

……..

P_n: nume_coloana_1

P_{n+1}: ()

Aşadar, dacă sunt specificate n expresii în operatorul ROLLUP, numărul de grupări generate va fi n + 1. Liniile care se bazează pe valoarea primelor n expresii se numesc linii obişnuite, iar celelalte se numesc linii *superagregat*.

Cum poate fi simulată folosirea operatorului ROLLUP? Cum este mai eficient?

R: Prin folosirea a n + 1 instrucțiuni SELECT conectate prin UNION ALL. Aceasta ar face execuția cererii ineficientă pentru că fiecare instrucțiune SELECT determină accesarea tabelului.

Exemplu:

Considerăm tabelul EMPLOYEES.

Pentru departamentele având codul mai mic decât 50, să se afişeze:

- ➢ pentru fiecare departament şi pentru fiecare an al angajării (corespunzător departamentului respectiv), valoarea totală a salariilor angajaților în acel an
- ➢ valoarea totală a salariilor pe departamente (indiferent de anul angajării)
- ➢ valoarea totală a salariilor (indiferent de anul angajării şi de departament)

```
SELECT department_id, TO_CHAR(hire_date, 'yyyy') AN, SUM(salary) SALARIU
FROM employees
WHERE department_id < 50
GROUP BY ROLLUP(department_id, TO_CHAR(hire_date, 'yyyy'));
```

Rezultatul va fi următorul:

	DEPARTMENT_ID	AN	SALARIU
1	10	1987	4400
2	10	(null)	4400
3	20	1996	13000
4	20	1997	6000
5	20	(null)	19000
6	30	1994	11000
7	30	1995	3100
8	30	1997	5700
9	30	1998	2600
10	30	1999	2500
11	30	(null)	24900
12	40	1994	6500
13	40	(null)	6500
14	(null)	(null)	54800

Se pot distinge 3 tipuri de linii:

1. *Unde datele au fost grupate după toate coloanele specificate în ROLLUP:* Prima linie reprezintă suma salariilor angajaților în 1987 din departamentul care are codul 10. În mod similar se interpretează liniile din rezultat care au toate coloanele completate.
2. *Unde datele au fost grupate numai după prima coloană specificată în ROLLUP:* Linia a doua conține valoarea totală a salariilor din departamentul al cărui cod este 10. La fel se interpretează toate liniile care se disting prin faptul că valoarea coloanei TO_CHAR(hire_date, ‘dd’) este null.
3. *Unde datele nu au fost grupate după nicio condi ie specificată în ROLLUP:* Ultima linie conține suma salariilor tuturor angajatiilor din departamentele al căror cod este mai mic decat 50. Întrucât această linie corespunde totalului general,ea conține valoarea null pe toate coloanele, cu excepția câmpului SUM(salary).

2. CUBE

➔ grupează liniile selectate pe baza valorilor tuturor combinațiilor posibile ale expresiilor specificate

```
SELECT nume_coloana_1, nume_coloana_2,...nume_coloana_n
FROM nume_tabel
GROUP BY CUBE(nume_coloana_1, nume_coloana_2,..., nume_coloana_n)
```

Dacă există n coloane sau expresii în clauza GROUP BY, vor exista 2^n combinații posibile. De exemplu, pentru n = 3, datele vor fi grupate astfel:

P_1: nume_coloana_1, nume_coloana_2, nume_coloana_3

P_2: nume_coloana_1, nume_coloana_2

P_3: nume_coloana_2, nume_coloana_3

P_4: nume_coloana_1, nume_coloana_3

P_5: nume_coloana_1

P_6: nume_coloana_2

P_7: nume_coloana_3

P_8: ()

De câte instrucțiuni SELECT conectate prin UNION ALL ar fi nevoie pentru a simula operatorul CUBE?

R: 2^n

Exemplu:

Considerăm tabelul EMPLOYEES.

Pentru departamentele având codul mai mic decât 50 să se afişeze:

- valoarea totală a salariilor corespunzătoare fiecărui an de angajare, din cadrul fiecărui departament
- valoarea totală a salariilor din fiecare departament (indiferent de anul angajarii)
- valoarea totală a salariilor corespunzatoare fiecărui an de angajare (indiferent de departament)
- valoarea totală a salariilor (indiferent de departament si de anul angajarii)

```
SELECT department_id, TO_CHAR(hire_date, 'yyyy') AN, SUM(salary) SALARIU
FROM employees
WHERE department_id < 50
GROUP BY CUBE(department_id, TO_CHAR(hire_date, 'yyyy'));
```

Rezultatul va fi următorul:

	DEPARTMENT_ID	AN	SALARIU
1	(null)	(null)	54800
2	(null)	1987	4400
3	(null)	1994	17500
4	(null)	1995	3100
5	(null)	1996	13000
6	(null)	1997	11700
7	(null)	1998	2600
8	(null)	1999	2500
9	10	(null)	4400
10	10	1987	4400
11	20	(null)	19000
12	20	1996	13000
13	20	1997	6000
14	30	(null)	24900
15	30	1994	11000
16	30	1995	3100
17	30	1997	5700
18	30	1998	2600
19	30	1999	2500
20	40	(null)	6500
21	40	1994	6500

În plus față de rezultatul corespunzător operației ROLLUP, operatorul CUBE va produce linii care reprezintă suma salariilor pentru fiecare an de angajare corespunzător unui departament având codul mai mic decât 50. Aceste linii se disting prin faptul că valoarea coloanei department_id este null.

3. GROUPING SETS

➔ datele sunt grupate numai după expresiile specificate in clauza GROUPING SETS

```
SELECT nume_coloana_1, nume_coloana_2,...nume_coloana_n
FROM nume_tabel
GROUP BY GROUPING SETS ((nume_coloana_11, nume_coloana_12, ...,
nume_coloana_1n), (nume_coloana_21, nume_coloana_22, ...nume_coloana_2m), ...)
```

Exemplu:

Considerăm tabelul EMPLOYEES.

Pentru departamentele având codul mai mic decât 50 să se afişeze:

- ➢ valoarea totală a salariilor corespunzătoare fiecărui an de angajare, din cadrul fiecărui departament
- ➢ valoarea totală a salariilor corespunzatoare fiecărui an de angajare (indiferent de departament)

```
SELECT department_id, TO_CHAR(hire_date, 'yyyy') AN, SUM(salary) SALARIU
FROM employees
WHERE department_id < 50
GROUP BY GROUPING SETS((department_id, TO_CHAR(hire_date, 'yyyy')),
(TO_CHAR(hire_date, 'yyyy')));
```

Rezultatul va fi următorul:

	DEPARTMENT_ID	AN	SALARIU
1	10	1987	4400
2	(null)	1987	4400
3	30	1994	11000
4	40	1994	6500
5	(null)	1994	17500
6	30	1995	3100
7	(null)	1995	3100
8	20	1996	13000
9	(null)	1996	13000
10	20	1997	6000
11	30	1997	5700
12	(null)	1997	11700
13	30	1998	2600
14	(null)	1998	2600
15	30	1999	2500
16	(null)	1999	2500

OBS: Pentru determinarea modului în care a fost obținută o valoare totalizatoare cu ROLLUP, CUBE sau GROUPING SETS, se utilizează funcția: **GROUPING(expresie)**. Aceasta întoarce:

- ❖ valoarea 0, dacă expresia a fost utilizată pentru calculul valorii agregat
- ❖ valoarea 1, dacă expresia nu a fost utilizată.

```
SELECT nume_coloana_1, nume_coloana_2,...nume_coloana_n,
GROUPING(nume_coloana_1), GROUPING(nume_coloana_2), …, GROUPING(nume_coloana_n)
FROM nume_tabel
GROUP BY CUBE(nume_coloana_1, nume_coloana_2,..., nume_coloana_n)
```

2. EXERCIŢII

1. Să se afişeze cel mai mare salariu, cel mai mic salariu, suma şi media salariilor tuturor angajaților. Etichetați coloanele Maxim, Minim, Suma, respectiv Media. Sa se rotunjeasca rezultatele.

2. Să se afişeze minimul, maximul, suma şi media salariilor pentru fiecare job.

3. Să se afişeze numărul de angajați pentru fiecare job.

4. Să se determine numărul de angajați care sunt şefi. Etichetati coloana “Nr. manageri”.

5. Să se afişeze diferența dintre cel mai mare si cel mai mic salariu. Etichetati coloana.

6. Scrieți o cerere pentru a se afişa numele departamentului, locația, numărul de angajați şi salariul mediu pentru angajații din acel departament. Coloanele vor fi etichetate corespunzător.

7. Pentru fiecare şef, să se afişeze codul său şi salariul celui mai prost platit subordonat. Se vor exclude cei pentru care codul managerului nu este cunoscut. De asemenea, se vor exclude grupurile în care salariul minim este mai mic de 1000$. Sortați rezultatul în ordine descrescătoare a salariilor.

8. Pentru departamentele in care salariul maxim depăşeşte 3000$, să se obțină codul, numele acestor departamente şi salariul maxim pe departament.

9. Care este salariul mediu minim al job-urilor existente? Salariul mediu al unui job va fi considerat drept media arirmetică a salariilor celor care îl practică.

OBS: Într-o imbricare de funcții agregat, criteriul de grupare specificat în clauza GROUP BY se referă doar la funcția agregat cea mai interioară. Astfel, într-o clauză SELECT în care există funcții agregat imbricate nu mai pot apărea alte expresii.

10. Să se afişeze codul, numele departamentului şi suma salariilor pe departamente.

11. Să se afişeze maximul salariilor medii pe departamente.
12. Să se afişeze salariul mediu din firmă doar dacă acesta este mai mare decât 2500. (clauza HAVING fără GROUP BY)

13. Să se afişeze suma salariilor pe departamente şi, în cadrul acestora, pe job-uri. Ordonați rezultatul în ordinea departamentelor.

14. Să se afiseze codul, numele departamentului şi numărul de angajaţi care lucrează în acel departament pentru departamentele în care lucrează mai puţin de 4 angajaţi;

15. Să se obțină numărul departamentelor care au cel puțin 15 angajați.

16. Să se obțină codul departamentelor şi suma salariilor angajaților care lucrează în acestea, în ordine crescătoare. Se consideră departamentele care au mai mult de 10 angajați şi al căror cod este diferit de 30.

17. Scrieti o cerere pentru a afisa, pentru departamentele avand codul > 80, salariul total pentru fiecare job din cadrul departamentului. Se vor afisa orasul, numele departamentului, jobul si suma salariilor. Se vor eticheta coloanele corespunzator.

18. Care sunt angajatii care au mai avut cel putin doua joburi?

19. Să se calculeze comisionul mediu din firmă, luând în considerare toate liniile din tabel.

20.
    a. Să se afişeze numele departamentelor, id-urile job-urilor şi valoarea medie rotunjită a salariilor, pentru:
        - fiecare departament şi, în cadrul său pentru fiecare job;
        - fiecare departament (indiferent de job);
        - întreg tabelul,
    b. Analog cu a, afişând şi o coloană care arată intervenția coloanelor department_name, job_id, în obținerea rezultatului.

21.
    a. Să se afişeze numele departamentelor, id-urile job-urilor şi valoarea medie rotunjită a salariilor, pentru:
        - fiecare departament şi, în cadrul său pentru fiecare job;
        - fiecare departament (indiferent de job);
        - fiecare job (indiferent de departament)
        - întreg tabelul.
    b. Cum intervin coloanele în obținerea rezultatului? Să se afişeze 'Dep', dacă departamentul a intervenit în agregare, şi 'Job', dacă job-ul a intervenit în agregare.

22. Să se afişeze numele departamentelor, numele job-urilor, codurile managerilor, maximul şi suma salariilor pentru:

- fiecare departament şi, în cadrul său, fiecare job;
- fiecare job şi, în cadrul său, pentru fiecare manager;
- întreg tabelul.