

# Grouping by More Than One Column

EMP

DEPTNO	JOB	SAL
10	MANAGER	2450
	PRESIDENT	5000
	CLERK	1300
20	CLERK	800
	CLERK	1100
	ANALYST	3000
	ANALYST	3000
20	MANAGER	2975
	SALESMAN	1600
	MANAGER	2850
30	SALESMAN	1250
	CLERK	950
	SALESMAN	1500
	SALESMAN	1250

“sum salaries in the EMP table for each job, grouped by department”

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
	MANAGER	2450
	PRESIDENT	5000
20	ANALYST	6000
	CLERK	1900
	MANAGER	2975
	CLERK	950
30	MANAGER	2850
	SALESMAN	5600

# Using the GROUP BY Clause on Multiple Columns

```
SQL> SELECT      deptno, job, sum(sal)
  2  FROM        emp
  3  GROUP BY    deptno, job;
```

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
...		

9 rows selected.

# Illegal Queries Using Group Functions

- Any column or expression in the SELECT list that is not an aggregate function must be in the GROUP BY

```
SQL> SELECT    deptno, COUNT(ename)  
  2  FROM      emp;
```

Column missing in the GROUP BY clause

# Excluding Group Results

EMP

DEPTNO	SAL
10	2450
	5000
	1300
20	800
	1100
	3000
	3000
	2975
30	1600
	2850
	1250
	950
	1500
	1250

5000

3000

2850

“maximum  
salary  
per department  
greater than  
\$2900”

DEPTNO	MAX (SAL)
10	5000
20	3000

# Excluding Group Results: HAVING Clause

- Use the HAVING clause to restrict groups
  - Rows are grouped.
  - The group function is applied.
  - Groups matching the HAVING clause are displayed.

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column] ;
```

# Using the HAVING Clause

```
SQL> SELECT      deptno,  max(sal)
  2  FROM        emp
  3  GROUP BY   deptno
  4  HAVING      max(sal)>2900;
```

DEPTNO	MAX (SAL)
10	5000
20	3000

# Using the HAVING Clause

```
SQL> SELECT      job, SUM(sal) AS PAYROLL  
  2  FROM        emp  
  3  WHERE       job NOT LIKE 'SALES%'  
  4  GROUP BY    job  
  5  HAVING      SUM(sal)>5000  
  6  ORDER BY    SUM(sal);
```

JOB	PAYROLL
ANALYST	6000
MANAGER	8275

# Summary of aggregating data

```
SELECT      column, group_function(column)
FROM        table
[WHERE       condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column] ;
```

- Order of evaluation of the clauses:
  - WHERE clause
  - GROUP BY clause
  - HAVING clause

# SUBQUERIES



# Using a Subquery to Solve a Problem

- “Who has a salary greater than Jones?”

## Main Query



“Which employees have a salary greater than Jones’ salary?”



## Subquery



“What is Jones’ salary?”

# Subqueries

```
SELECT      select_list
FROM        table
WHERE       expr operator
            (SELECT      select_list
             FROM       table) ;
```

- The subquery (inner query) executes once before the main query.
- The result of the subquery is used by the main query (outer query).

# Using a Subquery

```
SQL> SELECT ename  
  2  FROM emp  
  3 WHERE sal > 2975  
  4  
  5  
  6
```

```
(SELECT sal  
  FROM emp  
 WHERE empno=7566);
```

ENAME

-----

KING

FORD

SCOTT

# Guidelines for Using Subqueries

- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison operator.
- Use single-row operators with single-row subqueries.
- Use multiple-row operators with multiple-row subqueries.

# Types of Subqueries

- Single-row subquery



- Multiple-row subquery



# Single-Row Subqueries

- Return only one row
- Use single-row comparison operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
≠	Not equal to

# Executing Single-Row Subqueries

```
SQL> SELECT      ename, job
  2  FROM        emp
  3  WHERE       job =          ← CLERK
  4
  5
  6
  7  AND        sal >          ← 1100
  8
  9
 10
```

(SELECT job  
 FROM emp  
 WHERE empno = 7369)

(SELECT sal  
 FROM emp  
 WHERE empno = 7876);

ENAME	JOB
-----	-----
MILLER	CLERK

# Using Group Functions in a Subquery

```
SQL> SELECT      ename, job, sal
  2  FROM        emp
  3  WHERE       sal =          800
  4
  5          (SELECT      MIN(sal)
   FROM        emp) ;
```

ENAME	JOB	SAL
SMITH	CLERK	800

# HAVING Clause with Subqueries

- The Oracle Server executes subqueries first.
- The Oracle Server returns results into the HAVING clause of the main query.

```
SQL> SELECT      deptno, MIN(sal)
  2  FROM          emp
  3  GROUP BY      deptno
  4 HAVING        MIN(sal) > 800
  5
  6
  7
```

```
(SELECT      MIN(sal)
  FROM          emp
  WHERE         deptno = 20);
```



# What Is Wrong with This Statement?

```
SQL> SELECT empno, ename  
2   FROM emp  
3 WHERE sal =  
4  
5  
6
```

```
(SELECT      MIN(sal)  
  FROM        emp  
 GROUP BY    deptno);
```

Single-row operator with  
multiple-row subquery

ERROR:

ORA-01427: single-row subquery returns more than  
one row

no rows selected

# Will This Statement Work?

```
SQL> SELECT ename, job  
2   FROM emp  
3  WHERE job =  
4  
5  
6          (SELECT job  
      FROM emp  
     WHERE ename= 'SMYTHE' ) ;
```

```
no rows selected
```

Subquery returns no values

# Multiple-Row Subqueries

- Return more than one row
- Use multiple-row comparison operators

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

# Using ANY Operator in Multiple-Row Subqueries

```
SQL> SELECT empno, ename, job
  2  FROM emp
  3 WHERE sal < ANY
  4
  5
  6
  7 AND job <> 'CLERK';
```

```
(SELECT sal
  FROM emp
 WHERE job = 'CLERK')
```

EMPNO	ENAME	JOB
7654	MARTIN	SALESMAN
7521	WARD	SALESMAN

# Using ALL Operator in Multiple-Row Subqueries

```
SQL> SELECT    emno, ename, job
  2  FROM      emp
  3 WHERE     sal > ALL
  4
  5
  6
```

```
(SELECT          avg(sal)
   FROM           emp
   GROUP BY      deptno);
```

EMPNO	ENAME	JOB
7839	KING	PRESIDENT
7566	JONES	MANAGER
7902	FORD	ANALYST
7788	SCOTT	ANALYST

# Summary of Subqueries

- Subqueries are useful when a query is based on unknown values.

```
SELECT      select_list
FROM        table
WHERE       expr operator
            (SELECT select_list
             FROM    table);
```