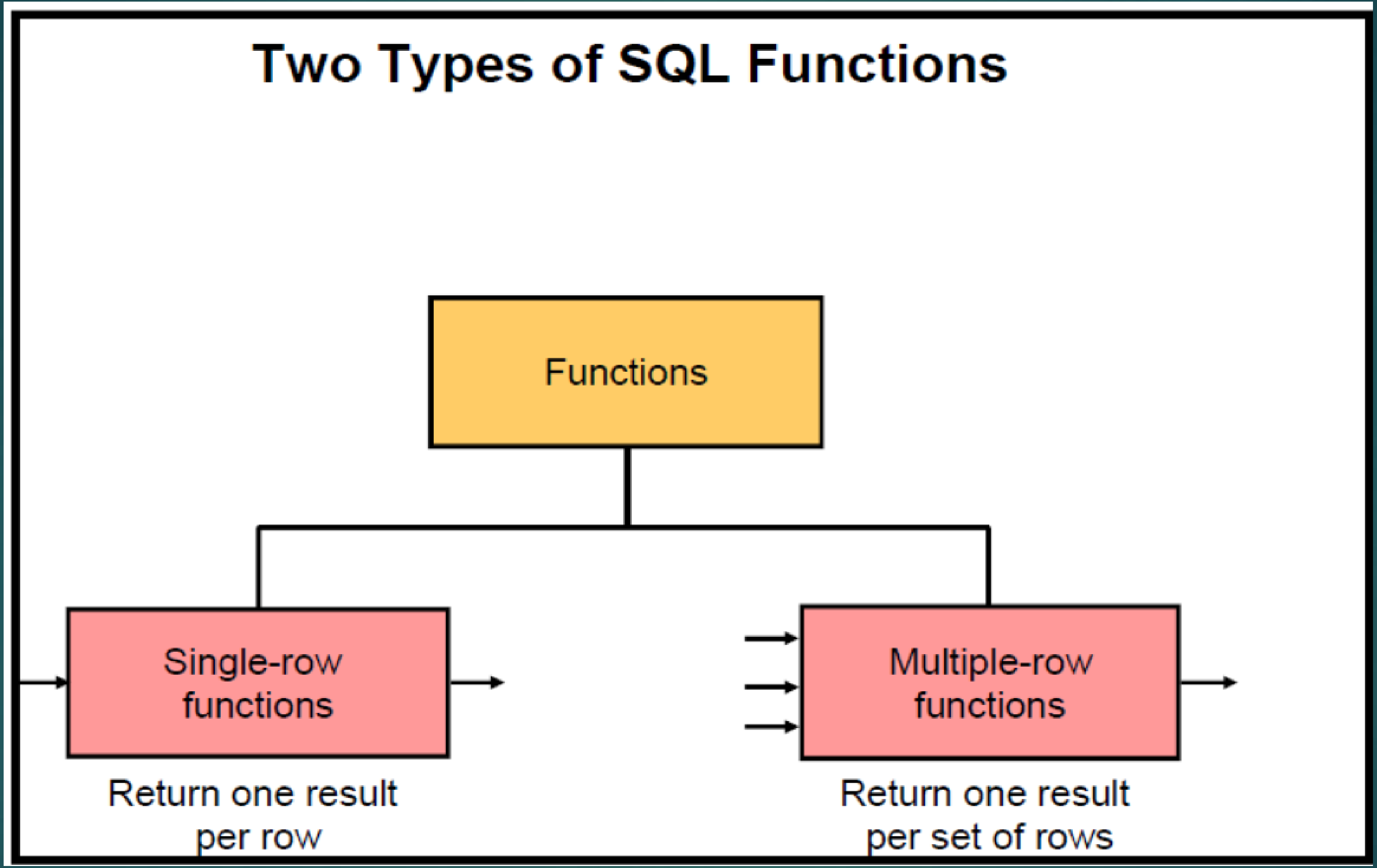
Grouping and Aggregating Data Using SQL

- ❖ Review Group functions and group by clause .
- ❖ A very nice examples to refresh your knowledge.
- ❖ Use the ROLLUP operator.
- ❖ Use the CUBE operator.
- ❖ Use the GROUPING function.
- ❖ Use GROUPING SETS.
- ❖ Work with composite columns.
- ❖ Use concatenated groupings to generate useful combinations of groupings.



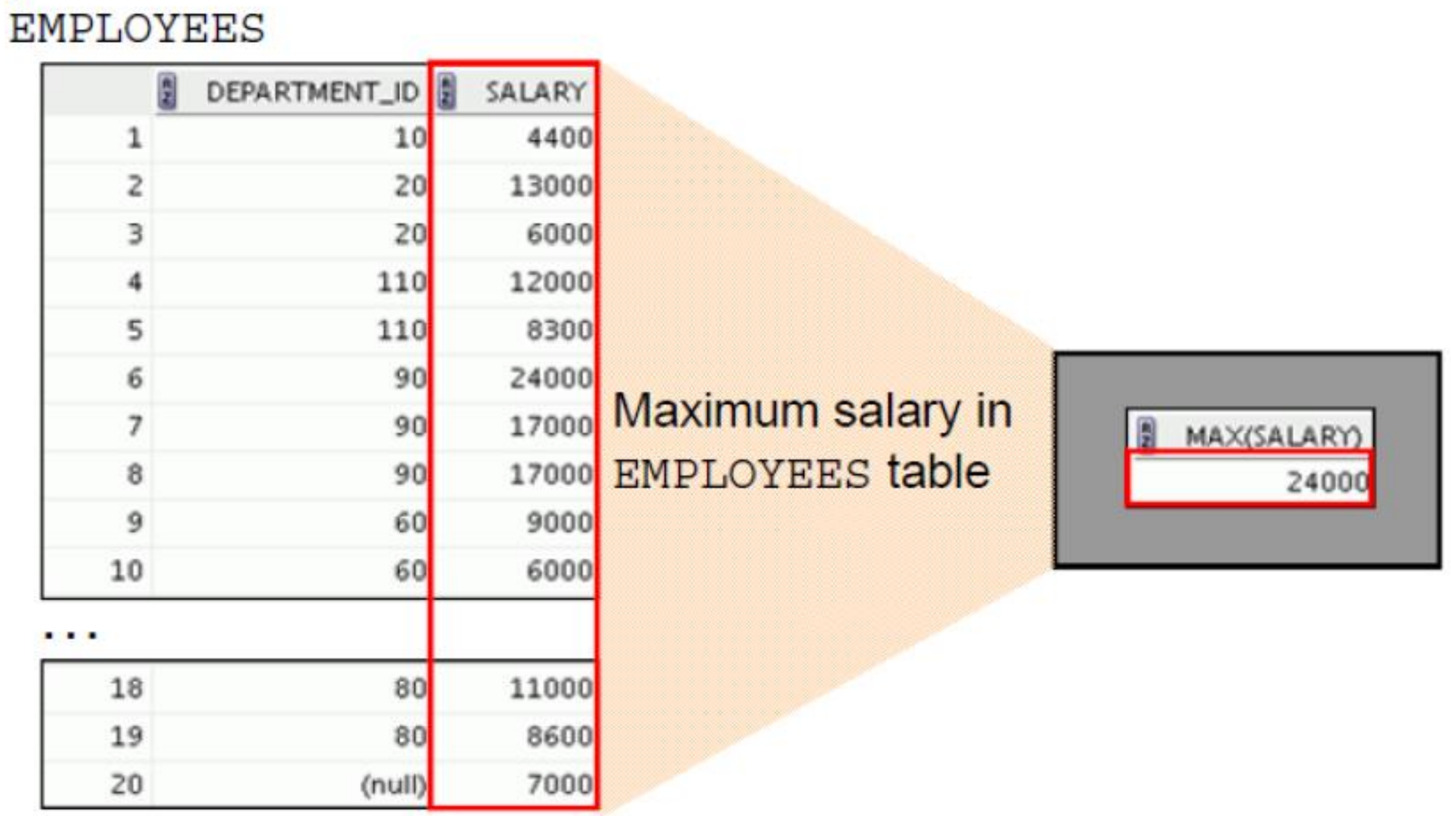
Two Types of SQL Functions

Return one result per row

Return one result per set of rows

What Are Group Functions?

Group functions operate on sets of rows to give one result per group.



EMPLOYEES

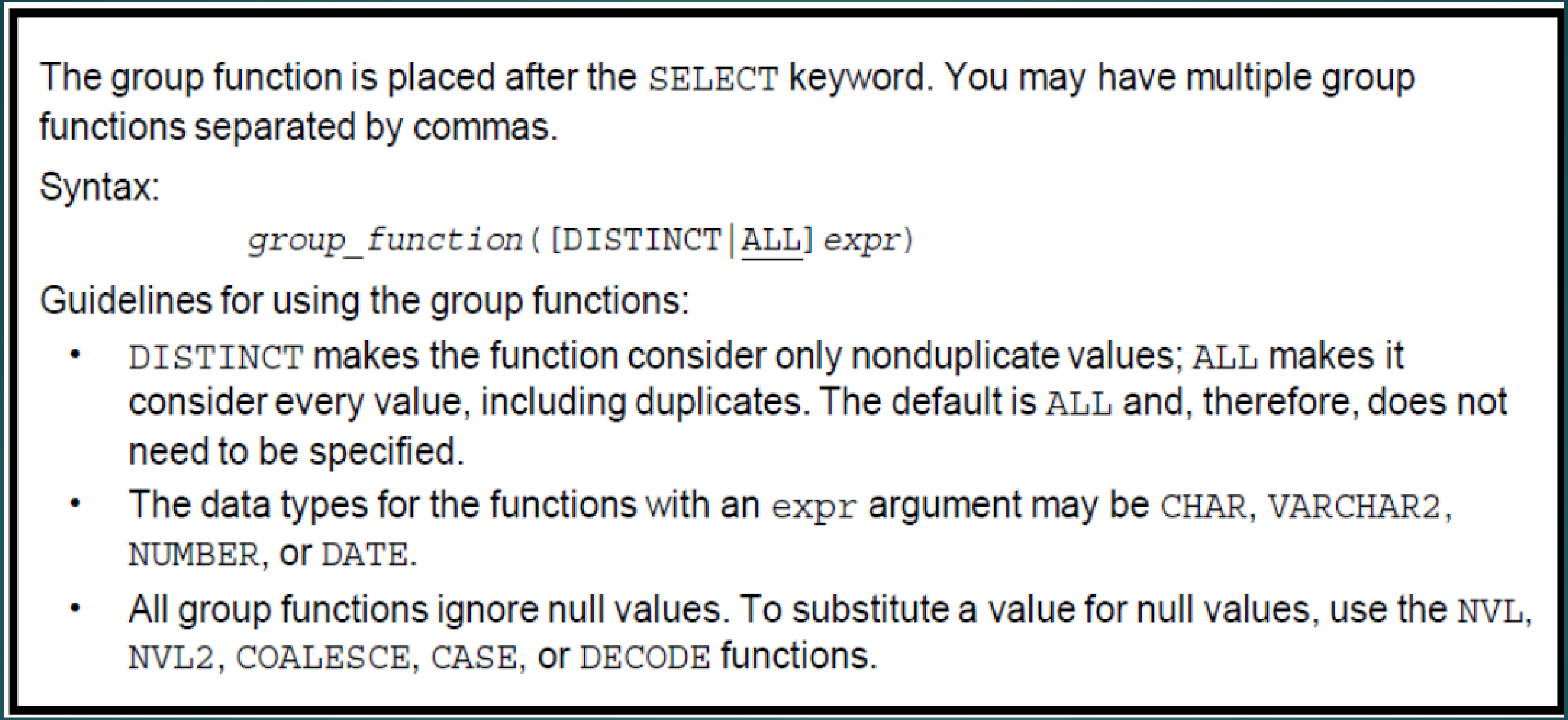
	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	110	12000
5	110	8300
6	90	24000
7	90	17000
8	90	17000
9	60	9000
10	60	6000
. . .		
18	80	11000
19	80	8600
20	(null)	7000

Maximum salary in EMPLOYEES table

MAX(SALARY)
24000

The most Popular group functions:

- ❖ AVG
- ❖ SUM
- ❖ MIN
- ❖ MAX
- ❖ COUNT
- ❖ STDDEV
- ❖ VARIANCE



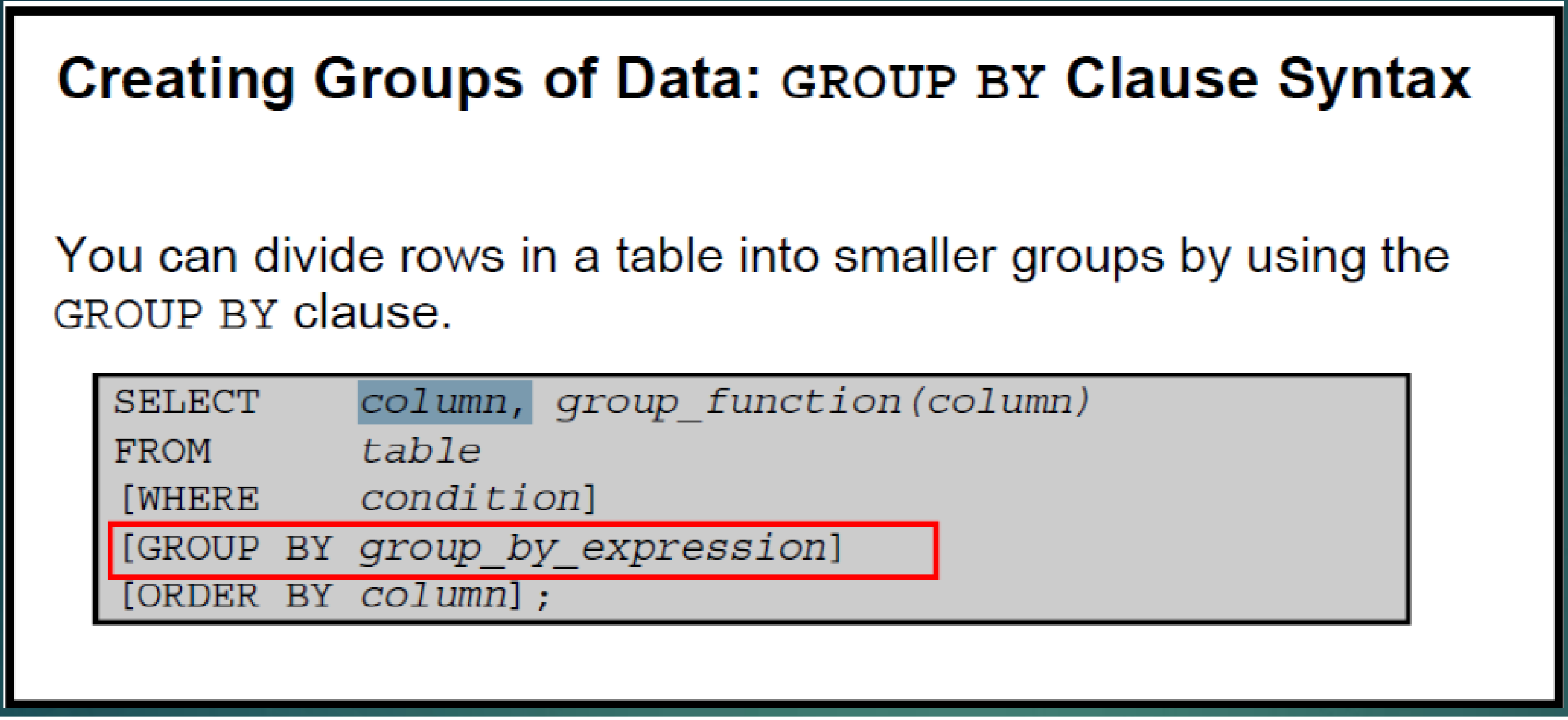
The group function is placed after the SELECT keyword. You may have multiple group functions separated by commas.

Syntax:

```
group_function([DISTINCT|ALL] expr)
```

Guidelines for using the group functions:

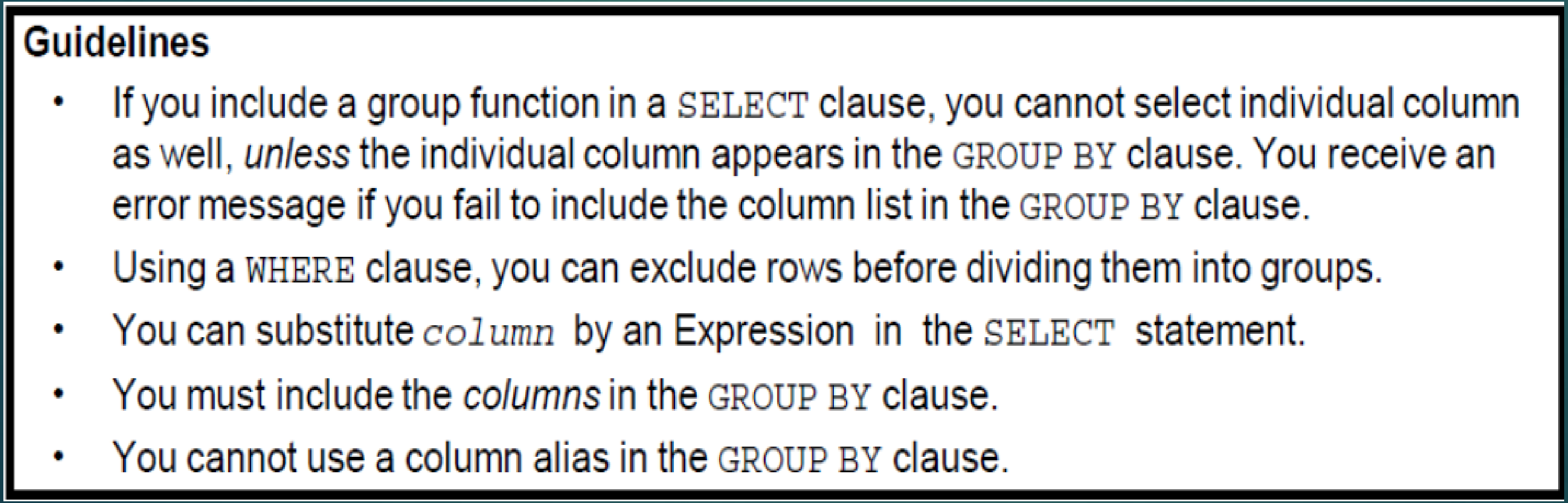
- DISTINCT makes the function consider only nonduplicate values; ALL makes it consider every value, including duplicates. The default is ALL and, therefore, does not need to be specified.
- The data types for the functions with an expr argument may be CHAR, VARCHAR2, NUMBER, or DATE.
- All group functions ignore null values. To substitute a value for null values, use the NVL, NVL2, COALESCE, CASE, or DECODE functions.



Creating Groups of Data: GROUP BY Clause Syntax

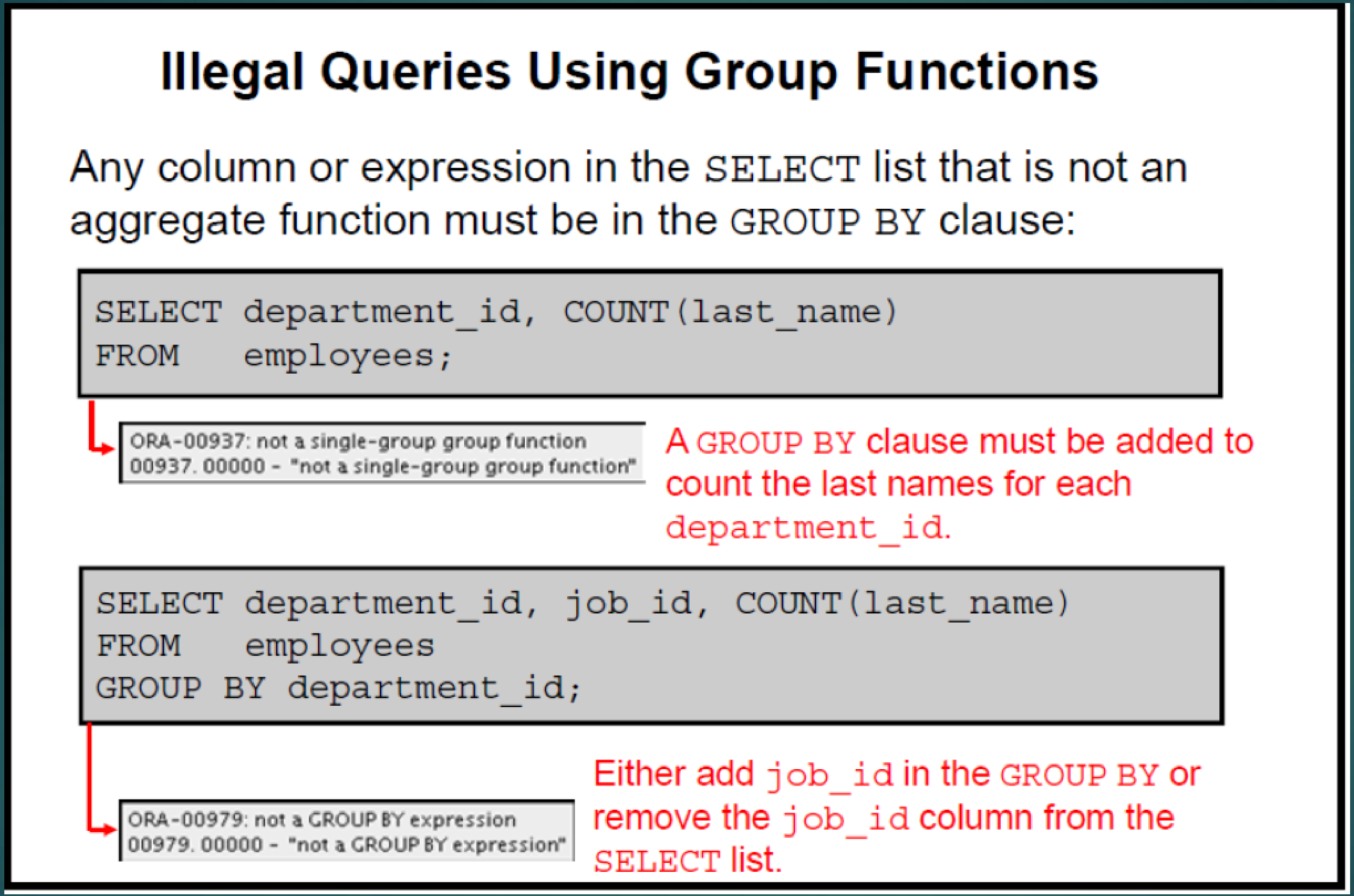
You can divide rows in a table into smaller groups by using the GROUP BY clause.

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



Guidelines

- If you include a group function in a SELECT clause, you cannot select individual column as well, *unless* the individual column appears in the GROUP BY clause. You receive an error message if you fail to include the column list in the GROUP BY clause.
- Using a WHERE clause, you can exclude rows before dividing them into groups.
- You can substitute *column* by an Expression in the SELECT statement.
- You must include the *columns* in the GROUP BY clause.
- You cannot use a column alias in the GROUP BY clause.



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` clause:

```
SELECT department_id, COUNT(last_name)
FROM   employees;
```

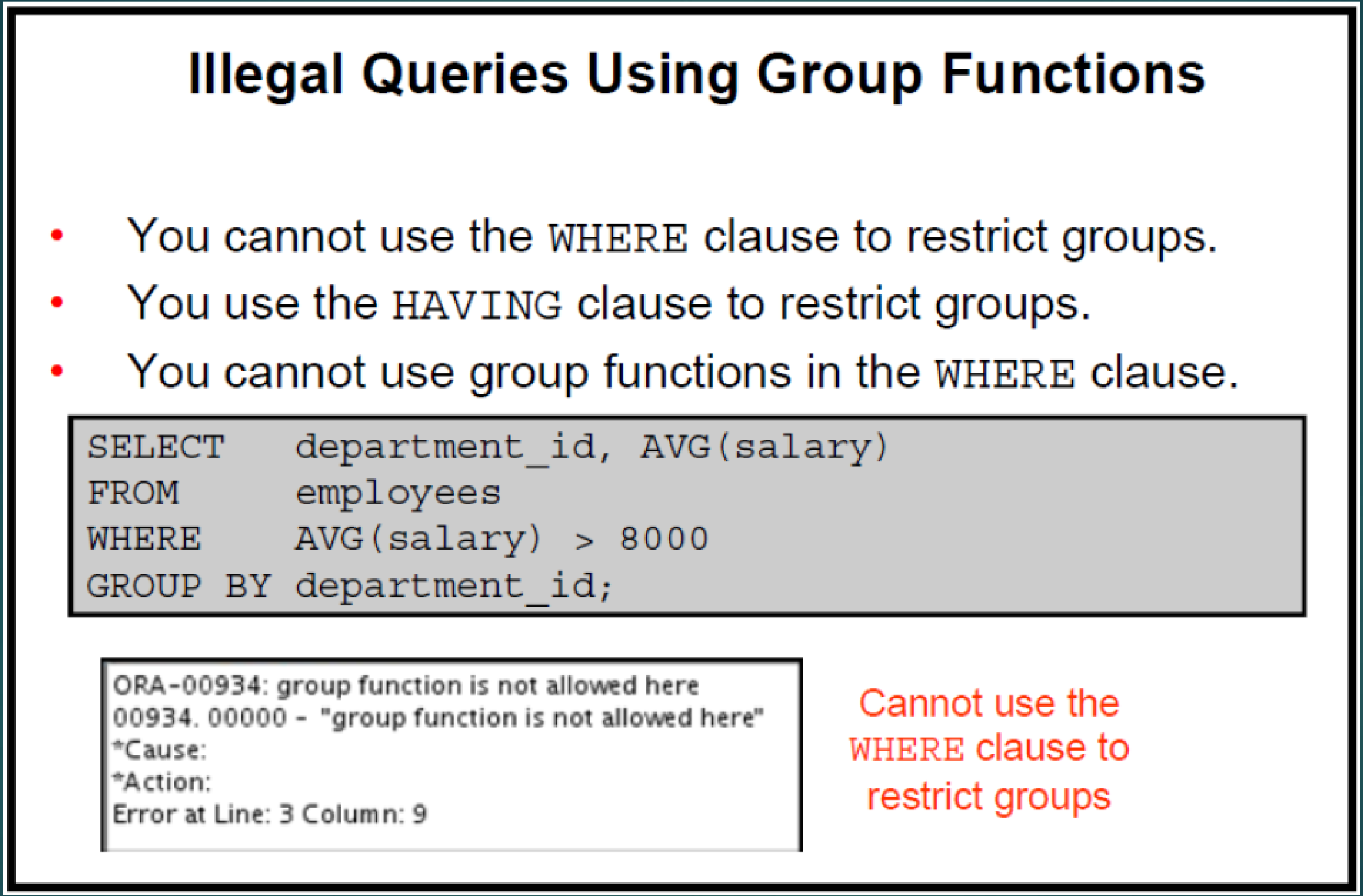
```
ORA-00937: not a single-group group function
00937. 00000 -  "not a single-group group function"
```

A `GROUP BY` clause must be added to count the last names for each `department_id`.

```
SELECT department_id, job_id, COUNT(last_name)
FROM   employees
GROUP BY department_id;
```

```
ORA-00979: not a GROUP BY expression
00979. 00000 -  "not a GROUP BY expression"
```

Either add `job_id` in the `GROUP BY` or remove the `job_id` column from the `SELECT` list.



Illegal Queries Using Group Functions

- You cannot use the `WHERE` clause to restrict groups.
- You use the `HAVING` clause to restrict groups.
- You cannot use group functions in the `WHERE` clause.

```
SELECT   department_id, AVG(salary)
FROM     employees
WHERE    AVG(salary) > 8000
GROUP BY department_id;
```

```
ORA-00934: group function is not allowed here
00934. 00000 - "group function is not allowed here"
*Cause:
*Action:
Error at Line: 3 Column: 9
```

Cannot use the `WHERE` clause to restrict groups

ROLLUP is an extension of the GROUP BY clause.
Use the ROLLUP operation to produce cumulative aggregates, such as Grand totals & subtotals.

The ROLLUP operator creates groupings by moving
in one direction,
from right to left, along the list
of columns specified in the GROUP BY clause.
It then applies the aggregate function to these groupings.

ROLLUP produces only a fraction of possible subtotal combinations

Group by rollup (col1);

We will have 2 types of totals

1. The total coming from the group statement
2. The Grand total

Group by rollup (col1, col2);

We will have 3 types of totals

N+1

1. The total coming from the group statement
2. The subtotal for col1
3. The grand total

Group by rollup (col1, col2, col3);

We will have 4 types of totals

1. The total coming from the group statement
2. The subtotal for col1 , col2
3. The subtotal total for col1
4. The grand total

N+1

Group by rollup (col1, col2, col3, col4);

We will have 5 types of totals

1. The total coming from the group statement
2. The subtotal for col1 , col2,col3
3. The subtotal total for col1,col2
4. The subtotal total for col1
5. The grand total

Grouping and Aggregating Data Using SQL 6.sql

GROUPING SETS

- The GROUPING SETS syntax is used to define multiple groupings in the same query.

- All groupings specified in the GROUPING SETS clause are computed and the results of individual groupings are combined with a UNION ALL operation.

- Grouping set efficiency:

– Only one pass over the base table is required.
– There is no need to write complex UNION statements.
– The more elements GROUPING SETS has, the greater the performance benefit.