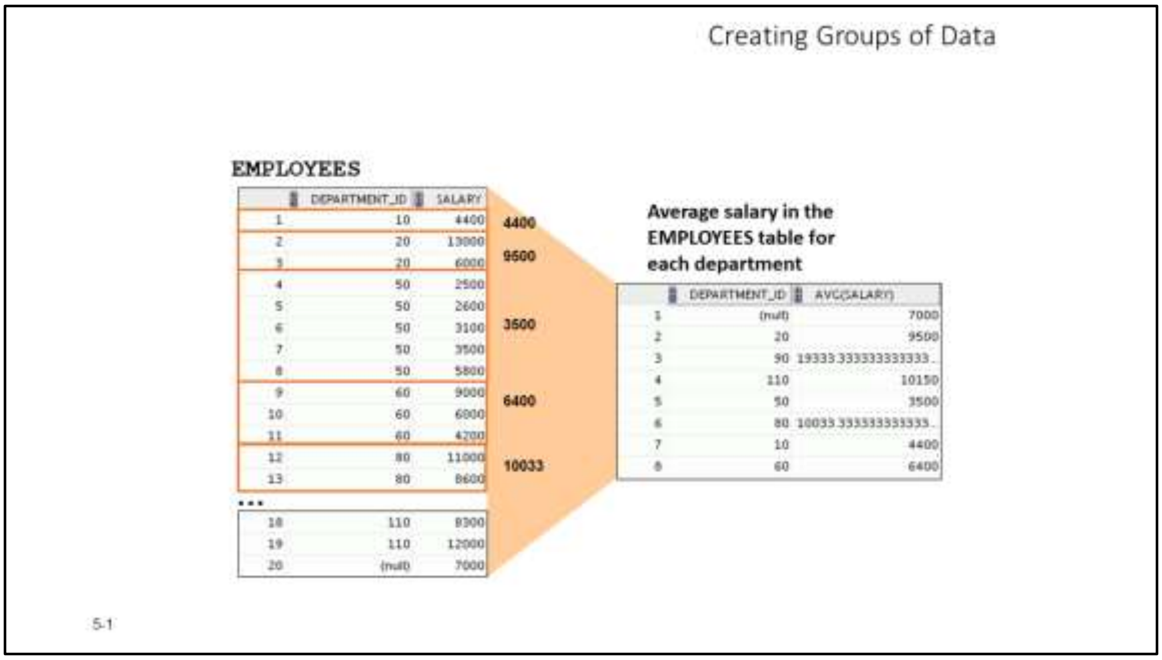


Creating Groups of Data

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	2500
5	50	2600
6	50	3100
7	50	3500
8	50	5800
9	60	9000
10	60	6000
11	60	4200
12	80	11000
13	80	8600
...		
18	110	8300
19	110	12000
20	(null)	7000

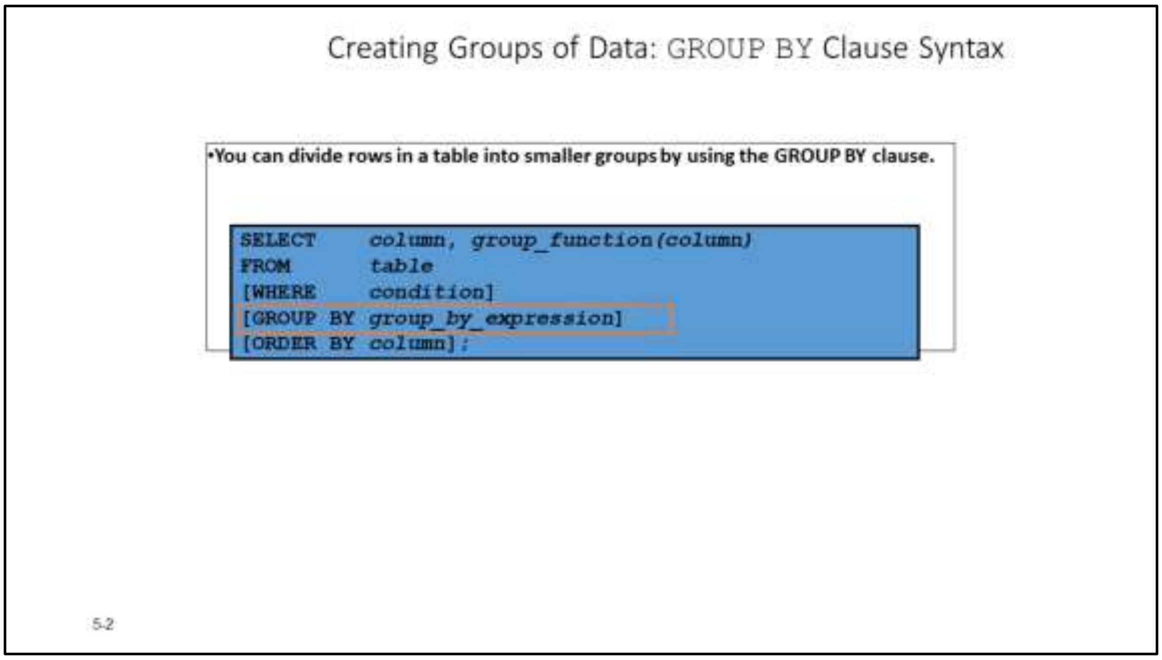
Average salary in the EMPLOYEES table for each department

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	20	9500
3	90	19333.3333333333333...
4	110	10150
5	50	3500
6	80	10033.3333333333333...
7	10	4400
8	60	6400

5-1

Creating Groups of Data

Until this point in the discussion, all group functions have treated the table as one large group of information. At times, however, you need to divide the table of information into smaller groups. This can be done by using the `GROUP BY` clause.



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];
```

5-2

Creating Groups of Data: GROUP BY Clause Syntax

You can use the GROUP BY clause to divide the rows in a table into groups. You can then use the group functions to return summary information for each group.

In the syntax:

group_by_expression Specifies the columns whose values determine the basis for grouping rows

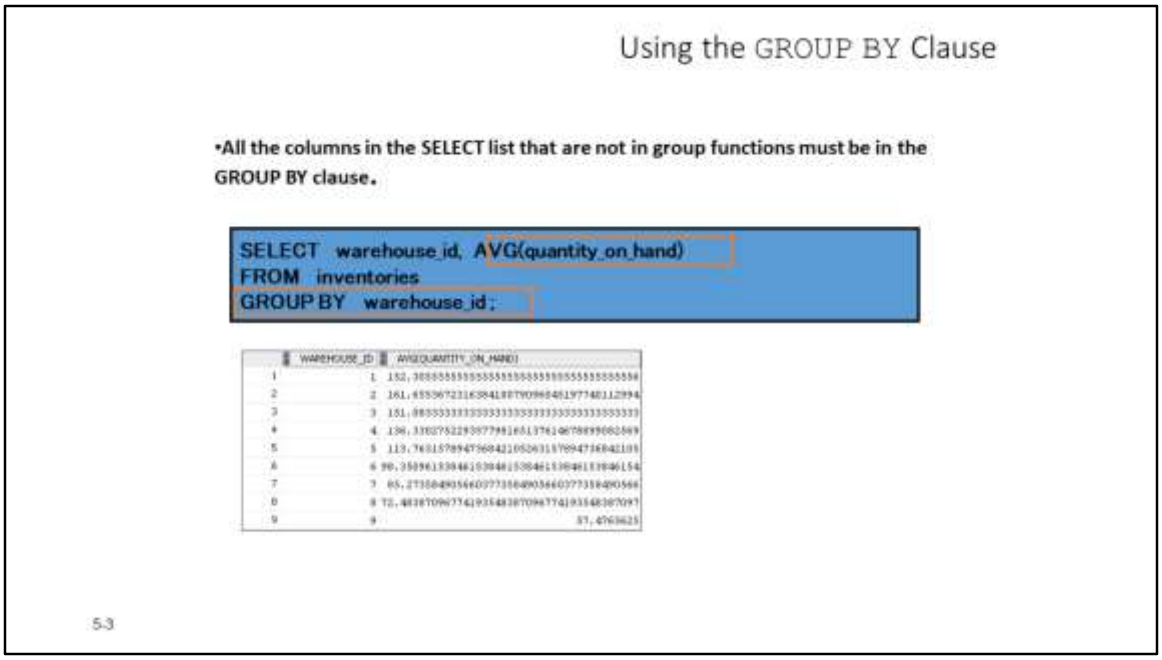
Guidelines

If you include a group function in a SELECT clause, you cannot select individual results 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 must include the *columns* in the GROUP BY clause.

You cannot use a column alias in the GROUP BY clause.



Using the GROUP BY Clause

•All the columns in the SELECT list that are not in group functions must be in the GROUP BY clause.

```
SELECT   warehouse_id,  AVG(quantity_on_hand)
FROM   inventories
GROUP BY   warehouse_id ;
```

```
     WAREHOUSE_ID  AVG(QUANTITY_ON_HAND)
1               1   152,3055555555555555555555555555555555556
2               2   161,6553672316384180790960451977401129944
3               3   151,0833333333333333333333333333333333333
4               4   138,3302752293577981651376146788990825688
5               5   113,7631578947368421052631578947368421053
6               6  90,35096153846153846153846153846153846154
7               7   85,2735849056603773584905660377358490566
8               8  72,48387096774193548387096774193548387097
9               9                                 37,4763625
```

5-3

Using the GROUP BY Clause

When using the GROUP BY clause, make sure that all columns in the SELECT list that are not group functions are included in the GROUP BY clause. The example in the slide displays the department number and the average salary for each department. Here is how this SELECT statement, containing a GROUP BY clause, is evaluated:

The SELECT clause specifies the columns to be retrieved, as follows:
Warehouse ID column in the INVENTORIES table.
The average of quantity on hand in the group that you specified in the GROUP BY clause

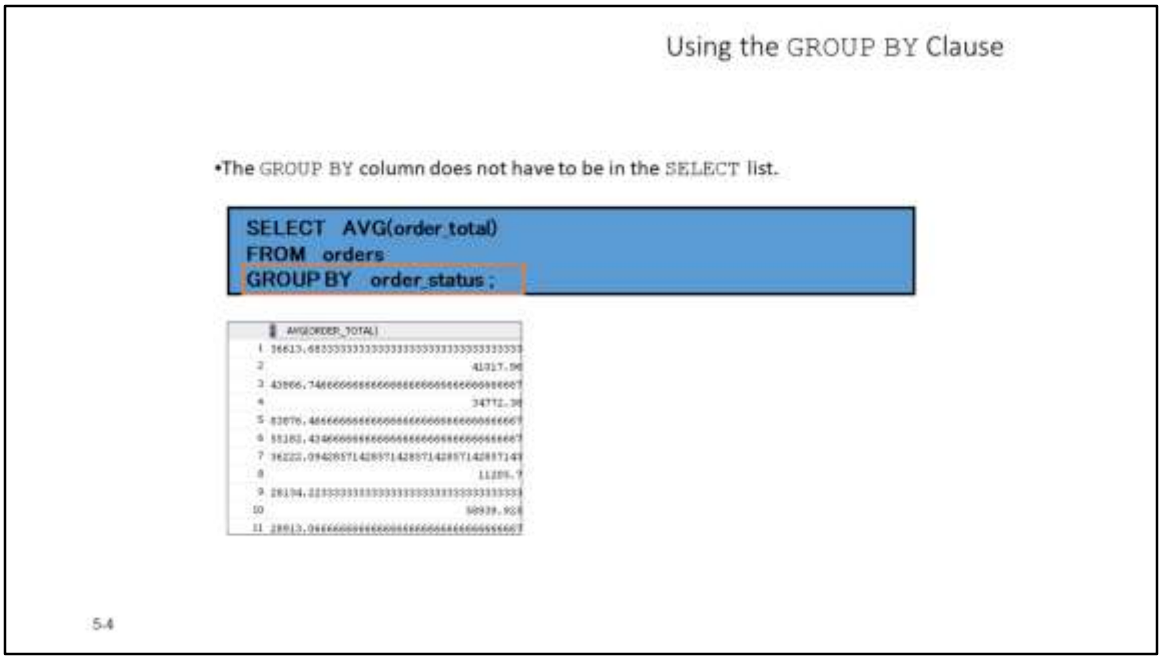
The FROM clause specifies the tables that the database must access: the INVENTORIES table.

The WHERE clause specifies the rows to be retrieved. Because there is no WHERE clause, all rows are retrieved by default.

The GROUP BY clause specifies how the rows should be grouped. The rows are grouped by warehouse_id column, so the AVG function that is applied to the quantity_on_hand column which calculates the average quantity_on_hand for each department.

Note: To order the query results in ascending or descending order, include the

`ORDER BY` clause in the query.



Using the GROUP BY Clause

•The GROUP BY column does not have to be in the SELECT list.

```
SELECT   AVG(order_total)
FROM   orders
GROUP BY   order_status ;
```

```
   AVG(ORDER_TOTAL)
 1 36613.68333333333333333333333333333333
 2                                 41017.98
 3 43966.74666666666666666666666666666667
 4                                 34772.38
 5 83076.48666666666666666666666666666667
 6 55183.43466666666666666666666666666667
 7 36222.09428571428571428571428571428571
 8                                  11205.7
 9 28134.22333333333333333333333333333333
10                                 58939.928
11 28913.06666666666666666666666666666667
```

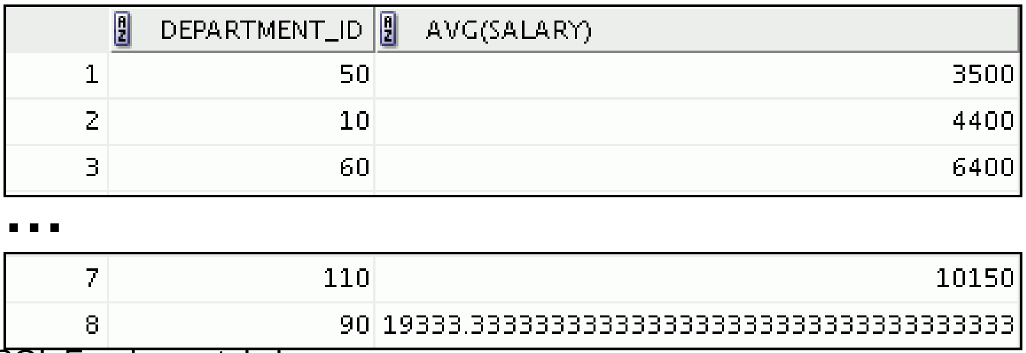
5-4

Using the GROUP BY Clause (continued)

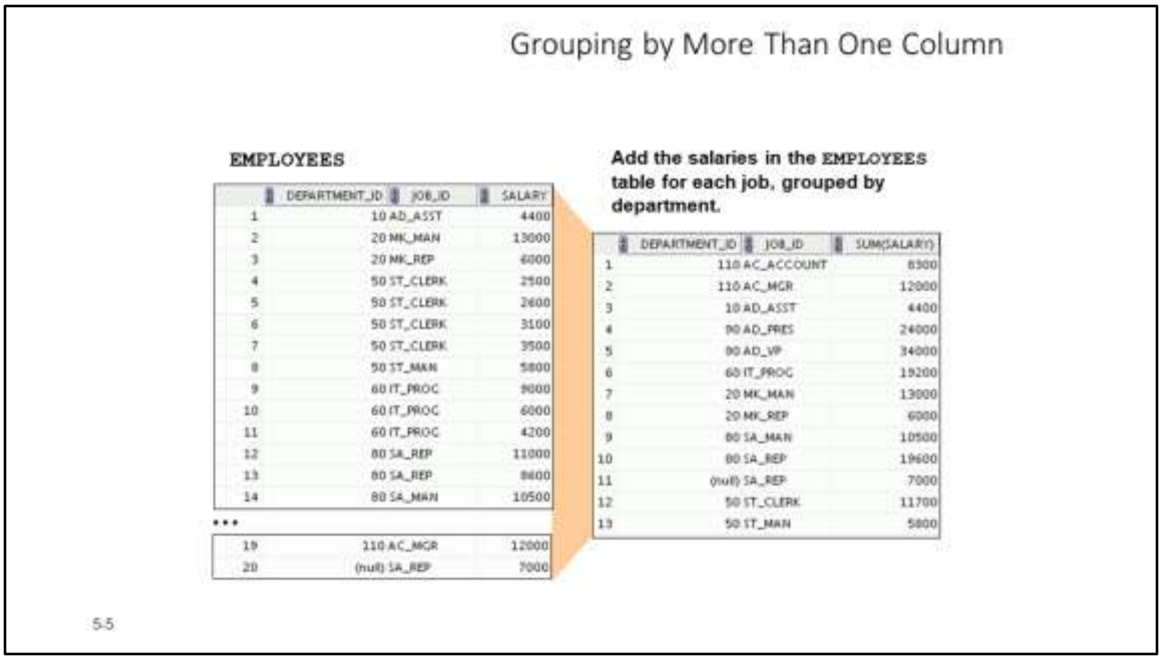
The GROUP BY column does not have to be in the SELECT clause. For example, the SELECT statement in the slide displays the average order value for each order status without displaying the respective status. Without the status, however, the results do not look meaningful.

You can also use the group function in the ORDER BY clause:

```
SELECT   order_status, AVG(order_total)
FROM     orders
GROUP BY order_status
ORDER BY AVG(order_total);
```



	DEPARTMENT_ID	AVG(SALARY)
1	50	3500
2	10	4400
3	60	6400
…		
7	110	10150
8	90	19333.3333333333333333333333333333333333



Grouping by More Than One Column

EMPLOYEES

	DEPARTMENT_ID	JOB_ID	SALARY
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_CLERK	2500
5	50	ST_CLERK	2600
6	50	ST_CLERK	3100
7	50	ST_CLERK	3500
8	50	ST_MAN	5800
9	60	IT_PROG	9000
10	60	IT_PROG	6000
11	60	IT_PROG	4200
12	80	SA_REP	11000
13	80	SA_REP	8600
14	80	SA_MAN	10500
•••			
19	110	AC_MGR	12000
20	(null)	SA_REP	7000

Add the salaries in the EMPLOYEES table for each job, grouped by department.

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	110	AC_ACCOUNT	8300
2	110	AC_MGR	12000
3	10	AD_ASST	4400
4	90	AD_PRES	24000
5	90	AD_VP	34000
6	60	IT_PROG	19200
7	20	MK_MAN	13000
8	20	MK_REP	6000
9	80	SA_MAN	10500
10	80	SA_REP	19600
11	(null)	SA_REP	7000
12	50	ST_CLERK	11700
13	50	ST_MAN	5800

5-5

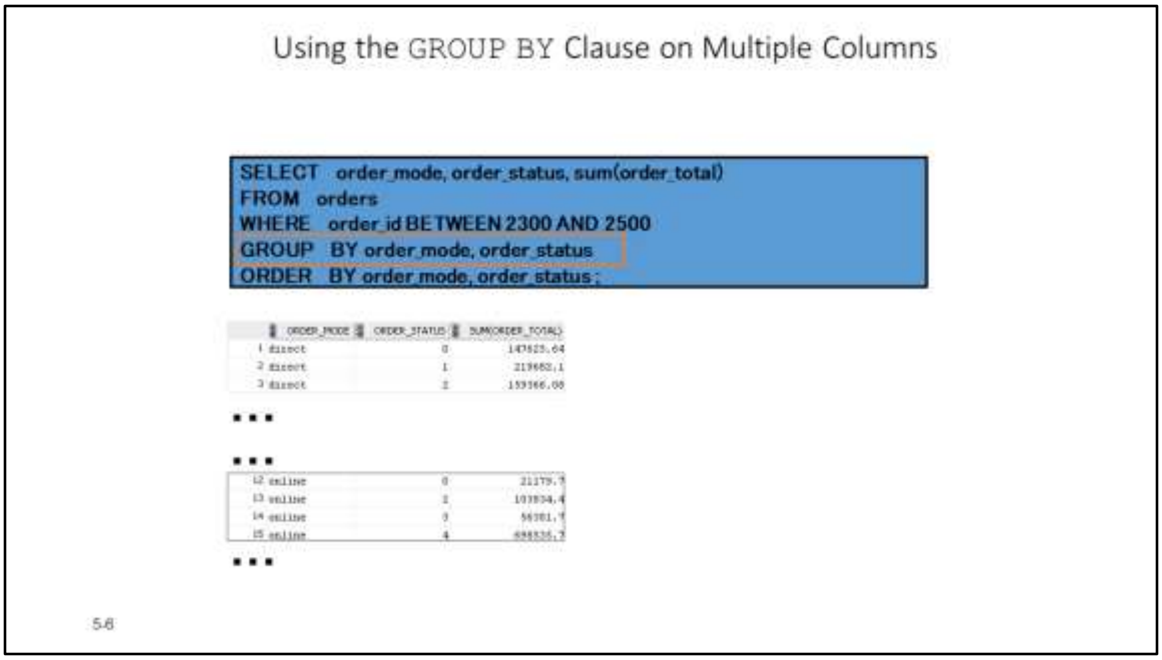
Grouping by More Than One Column

Sometimes, you need to see results for groups within groups. The slide shows a report that displays the total salary that is paid to each job title in each department.

The `EMPLOYEES` table is grouped first by the department number, and then by the job title within that grouping. For example, the four stock clerks in department 50 are grouped together, and a single result (total salary) is produced for all stock clerks in the group.

The following `SELECT` statement returns the result shown in the slide:

```
SELECT   department_id, job_id, sum(salary)
FROM     employees
GROUP BY department_id, job_id
ORDER BY job_id;
```



Using the GROUP BY Clause on Multiple Columns

```
SELECT   order_mode, order_status, sum(order_total)
FROM   orders
WHERE   order_id BETWEEN 2300 AND 2500
GROUP   BY order_mode, order_status
ORDER   BY order_mode, order_status ;
```

```
    ORDER_MODE   ORDER_STATUS   SUM(ORDER_TOTAL)
  1 direct                  0          147823.64
  2 direct                  1           219682.1
  3 direct                  2          159366.08
...

...
 12 online                  0            21179.7
 13 online                  2           103834.4
 14 online                  3            56301.7
 15 online                  4           698535.7
...
```

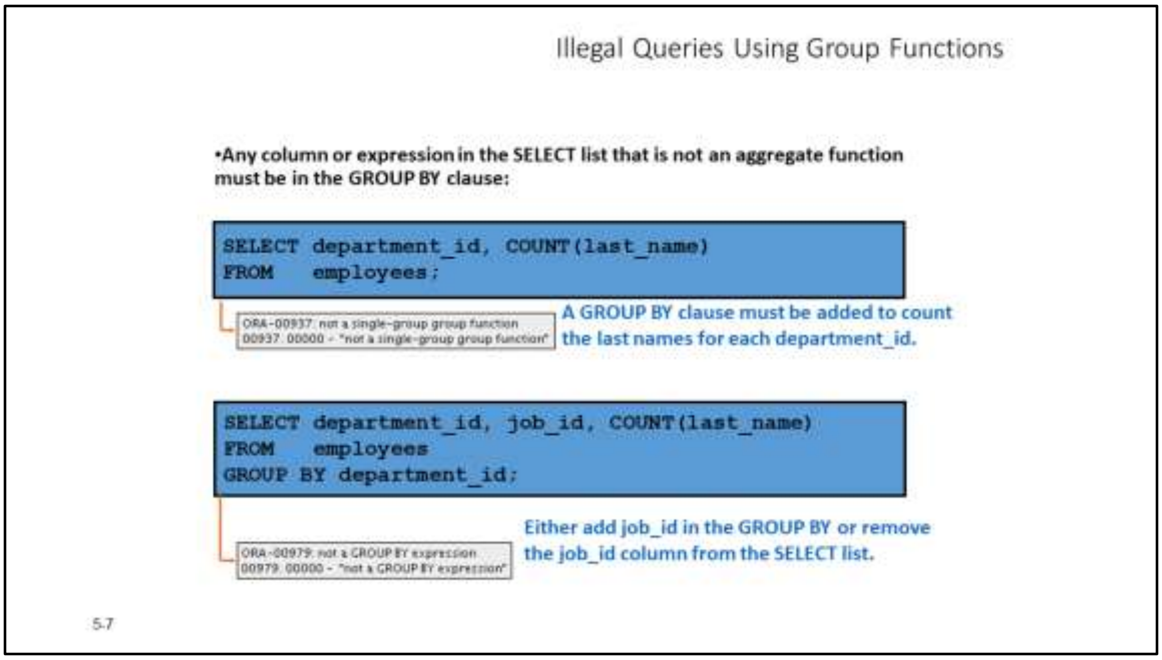
5-6

Using the Group By Clause on Multiple Columns

You can return summary results for groups and subgroups by listing multiple GROUP BY columns. The GROUP BY clause groups rows but does not guarantee the order of the result set. To order the groupings, use the ORDER BY clause.

In the example in the slide, the SELECT statement that contains a GROUP BY clause is evaluated as follows:

- The SELECT clause specifies the column to be retrieved:
  - order_mode in the ORDERS table
  - order_status in the ORDERS table
  - The sum of all order_total in the group that you specified in the GROUP BY clause
- The FROM clause specifies the tables that the database must access: the ORDERS table.
- The WHERE clause reduces the result set to those rows where order_id is between 2300 and 2500.
- The GROUP BY clause specifies how you must group the resulting rows:
  - First, the rows are grouped by the order_mode.
  - Second, the rows are grouped by order_status in the order_mode groups.
- The ORDER BY clause sorts the results by order_mode and order_status.



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.

5-7

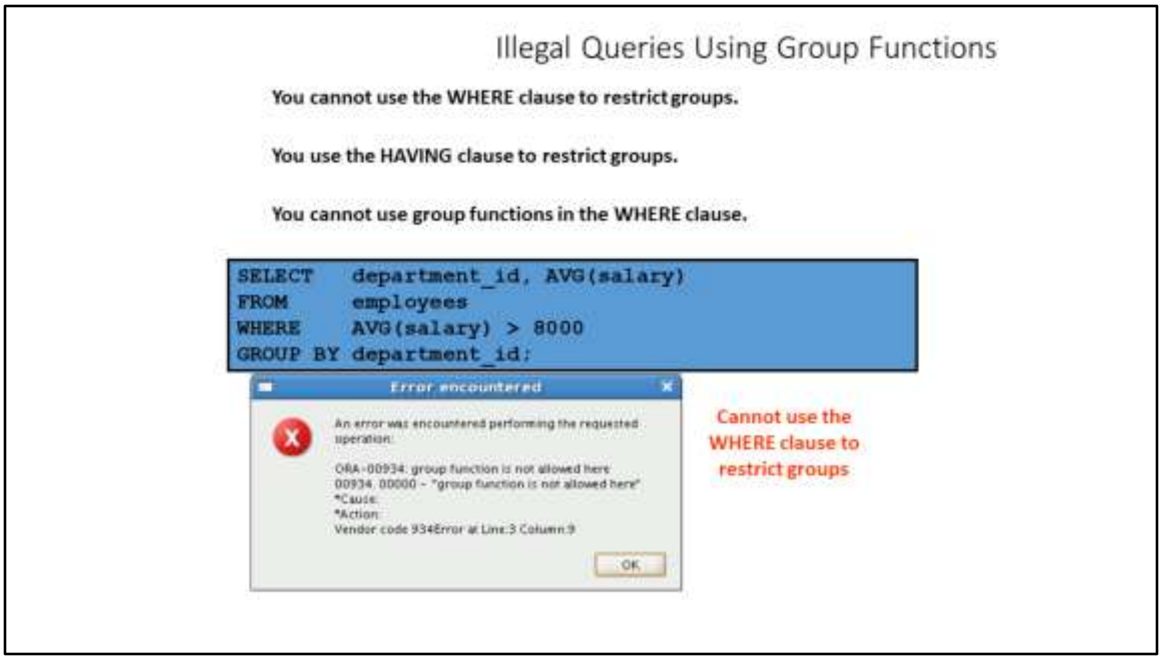
Illegal Queries Using Group Functions

Whenever you use a mixture of individual items (DEPARTMENT_ID) and group functions (COUNT) in the same SELECT statement, you must include a GROUP BY clause that specifies the individual items (in this case, DEPARTMENT_ID). If the GROUP BY clause is missing, the error message "not a single-group group function" appears and an asterisk (*) points to the offending column. You can correct the error in the first example in the slide by adding the GROUP BY clause:

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

Any column or expression in the SELECT list that is not an aggregate function must be in the GROUP BY clause. In the second example in the slide, job_id is neither in the GROUP BY clause nor is it being used by a group function, so there is a "not a GROUP BY expression" error. You can correct the error in the second slide example by adding job_id in the GROUP BY clause.

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



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;
```

```
Error encountered
An error was encountered performing the requested operation:

ORA-00934: group function is not allowed here
00934. 00000 - "group function is not allowed here"
*Cause:
*Action:
Vendor code 934Error at Line:3 Column:9
```

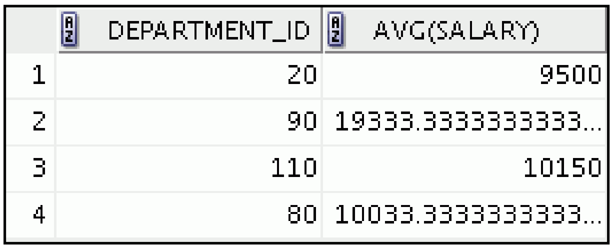
Cannot use the WHERE clause to restrict groups

Illegal Queries Using Group Functions (continued)

The `WHERE` clause cannot be used to restrict groups. The `SELECT` statement in the example in the slide results in an error because it uses the `WHERE` clause to restrict the display of the average salaries of those departments that have an average salary greater than $8,000.

However, you can correct the error in the example by using the `HAVING` clause to restrict groups:

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



	DEPARTMENT_ID	AVG(SALARY)
1	20	9500
2	90	19333.3333333333...
3	110	10150
4	80	10033.3333333333...