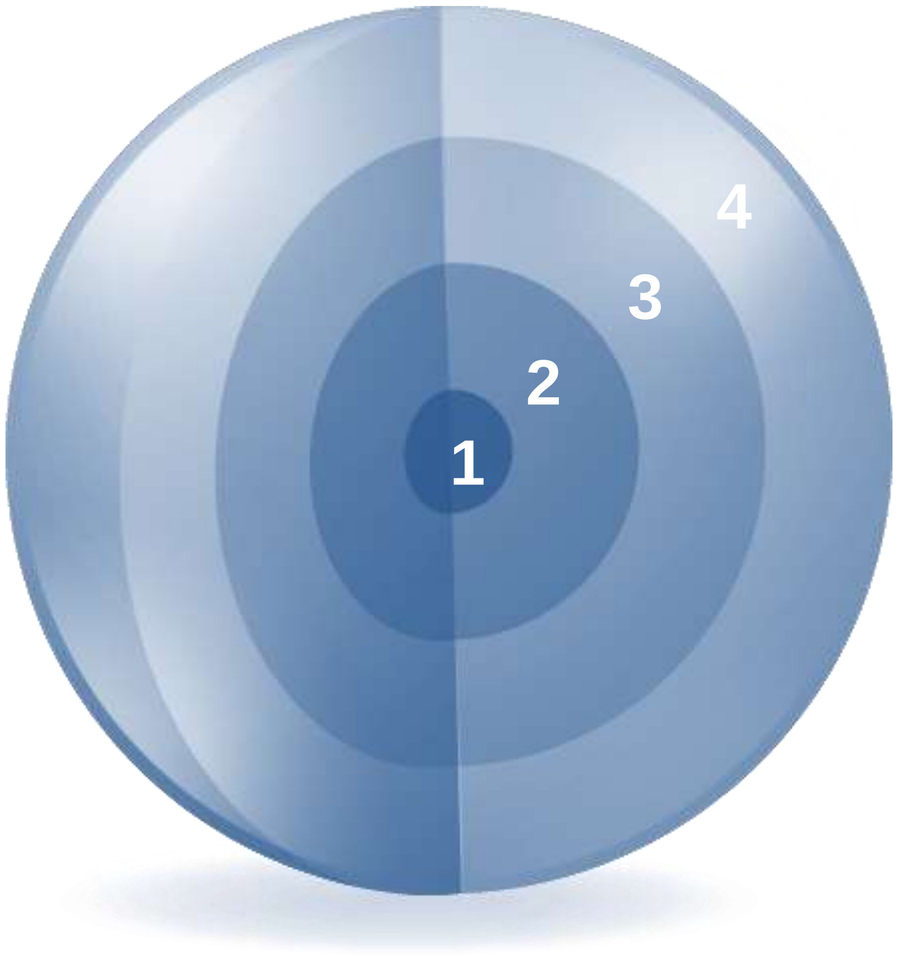
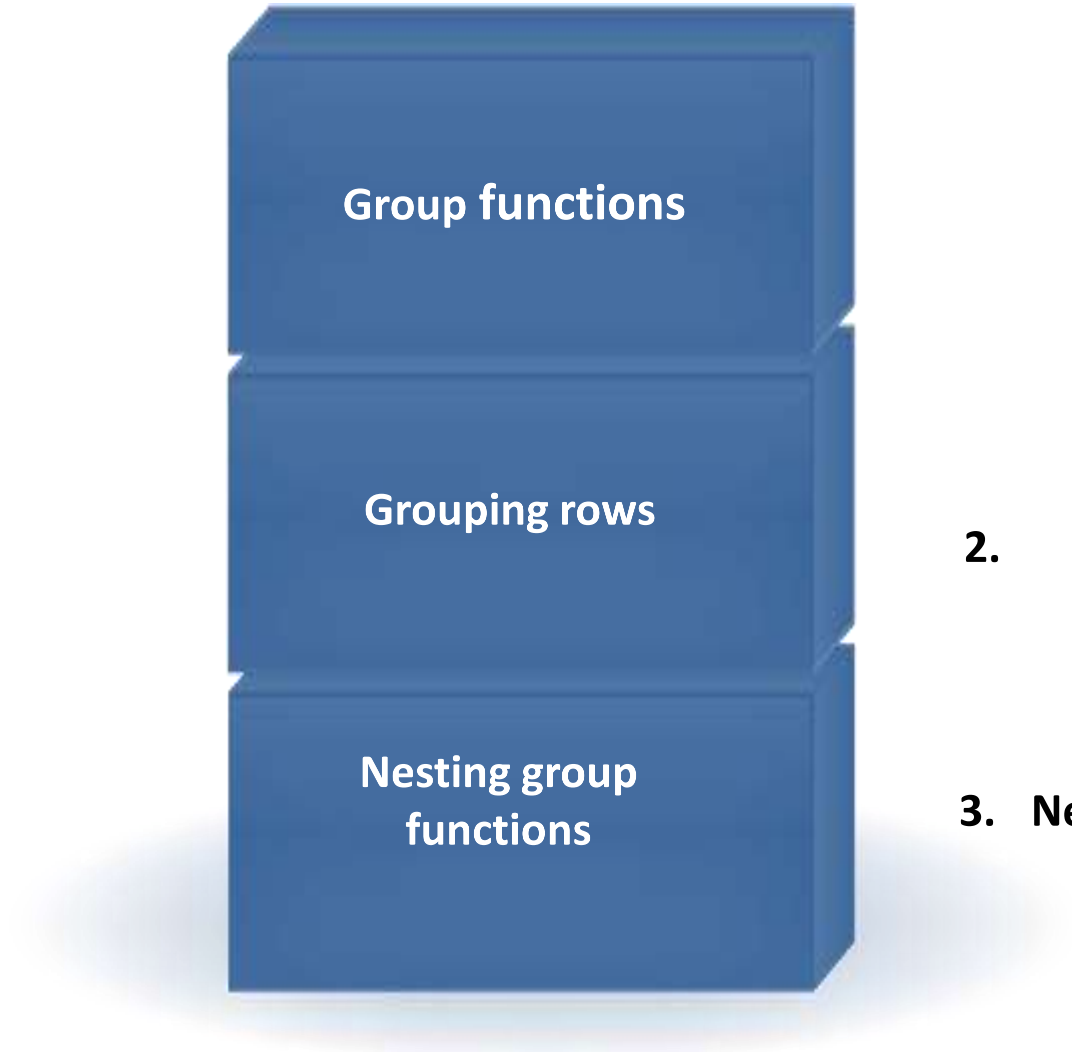
Reporting Aggregated Data Using the Group Functions

What You will learn at the end of the Session?



1. **Identify the available group functions**
2. **Describe the use of group functions.**
3. **Group data by using the GROUP BY clause**
4. **Include or exclude grouped rows by using the HAVING clause**

Session Plan



1. Group functions:
   - Types and syntax
   - Use AVG, SUM, MIN, MAX, COUNT
   - Use the DISTINCT keyword within group functions
   - NULL values in a group function

2. Grouping rows:

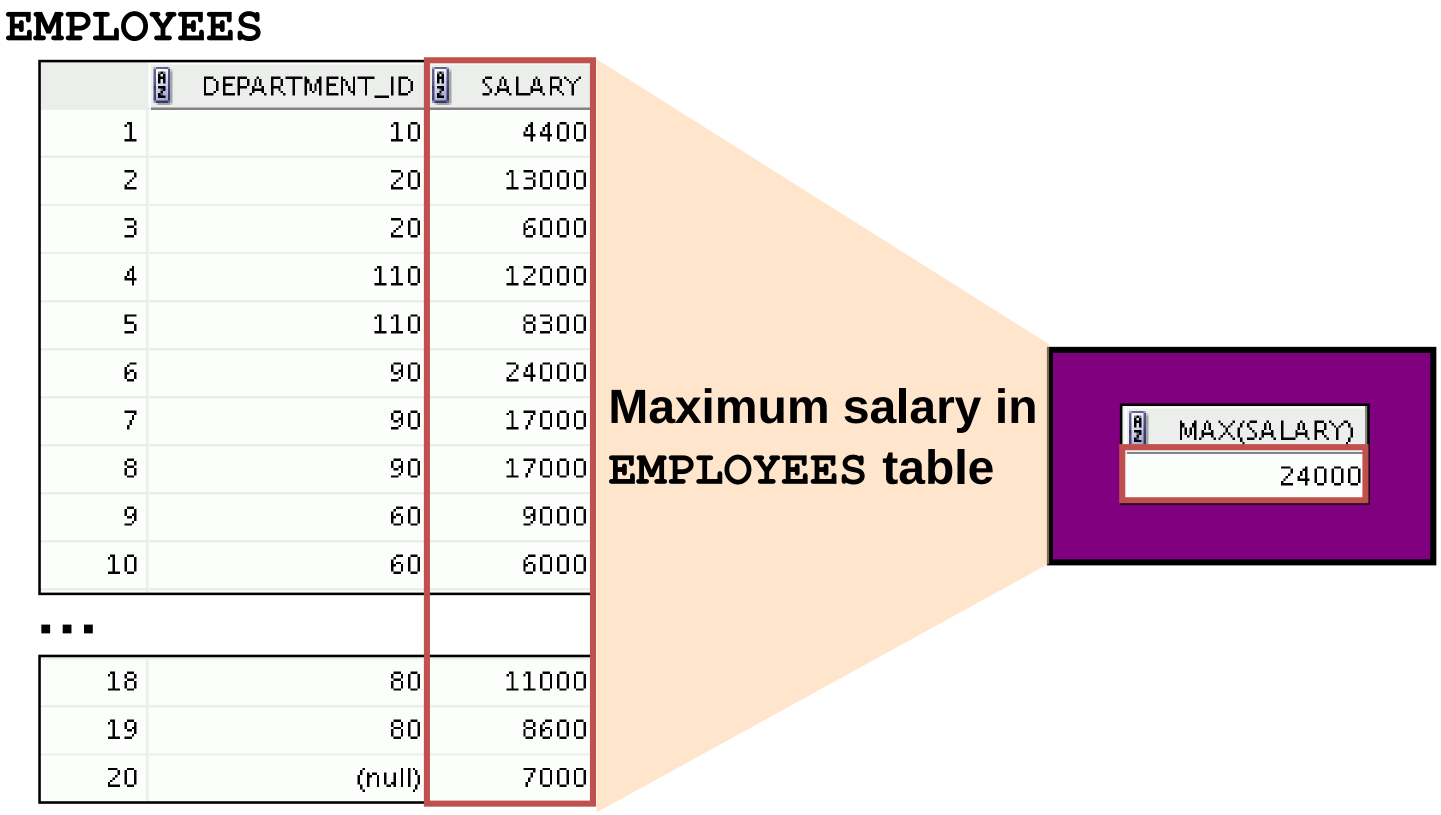
   GROUP BY clause

   HAVING clause

3. Nesting group functions

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

…

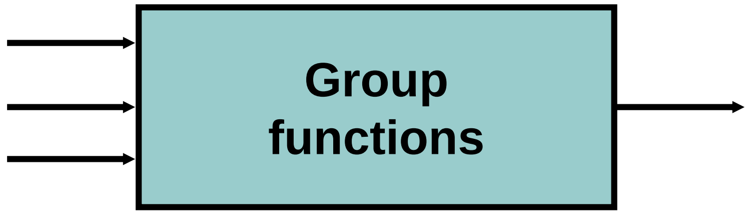
18	80	11000
19	80	8600
20	(null)	7000

Maximum salary in EMPLOYEES table

MAX(SALARY)
24000

Types of Group Functions

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

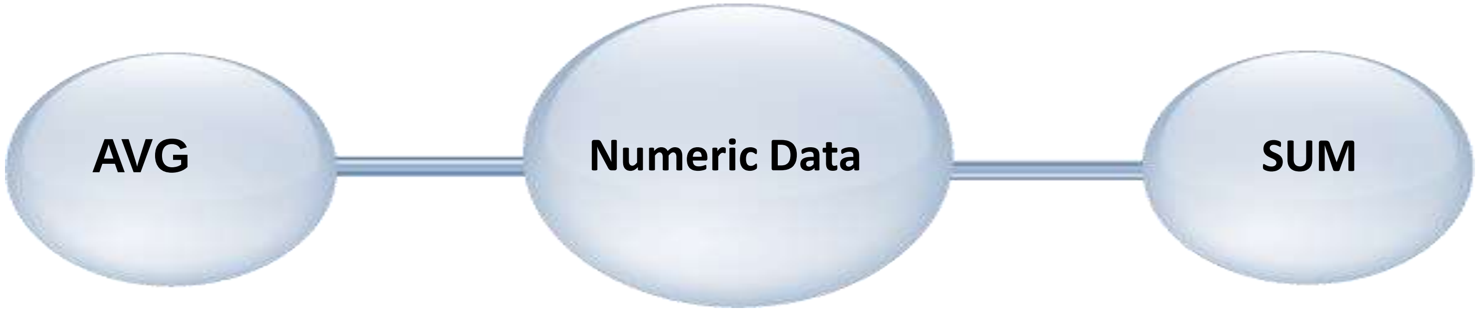


Group Functions: Syntax

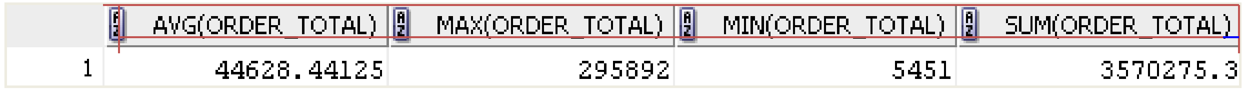
```
SELECT       group_function(column), ...
FROM         table
[WHERE       condition]
[ORDER BY    column];
```

Using the AVG and SUM Functions

- You can use AVG and SUM for numeric data.



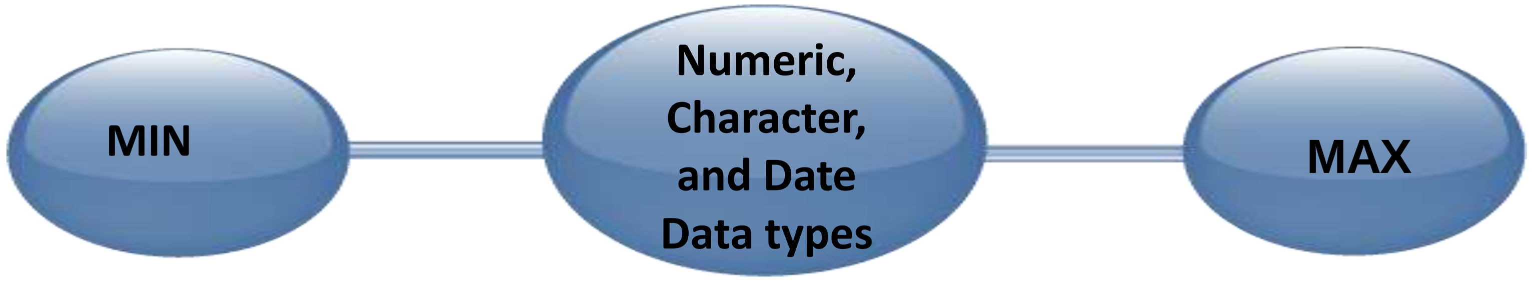
```
SELECT   AVG(order_total),  MAX(order_total),
         MIN (order_total), SUM( order_total)
FROM   orders;
```



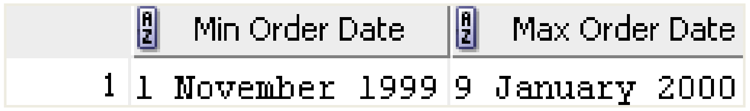
```
   AVG(ORDER_TOTAL)  MAX(ORDER_TOTAL)  MIN(ORDER_TOTAL)  SUM(ORDER_TOTAL)
1       44628.44125            295892              5451         3570275.3
```

Using the MIN and MAX Functions

•You can use MIN and MAX for numeric, character, and date data types.



```
SELECT   MIN(to_char(order_date, 'fmDD Month YYYY'))
AS "Min Order Date",
MAX(to_char(order_date, 'fmDD Month YYYY'))
AS "Max Order Date"
FROM   orders ;
```



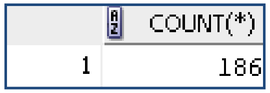
	Min Order Date	Max Order Date
1	1 November 1999	9 January 2000

Using the COUNT Function

•COUNT(*) returns the number of rows in a table:

1

```
SELECT  count(*)
FROM   inventories
WHERE   warehouse_id = 8;
```

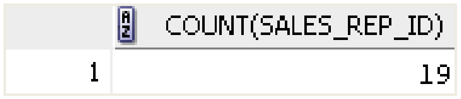


```
     COUNT(*)
1         186
```

•COUNT(*expr*) returns the number of rows with non-null values for *expr*:

2

```
SELECT  COUNT(sales_rep_id)
FROM   orders
WHERE   order_status <=3;
```



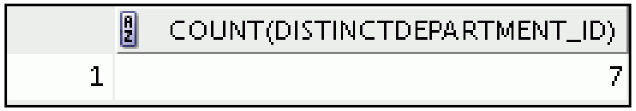
```
     COUNT(SALES_REP_ID)
1                     19
```

ORACLE

Using the DISTINCT Keyword

- COUNT(DISTINCT expr) returns the number of distinct non-null values of *expr*.
- To display the number of distinct department values in the EMPLOYEES table:

```
SELECT COUNT(DISTINCT department_id)
FROM   employees;
```



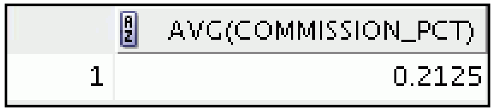
```
     COUNT(DISTINCTDEPARTMENT_ID)
1                               7
```

Group Functions and Null Values

- Group functions ignore null values in the column:

1

```
SELECT AVG(commission_pct)
FROM   employees;
```

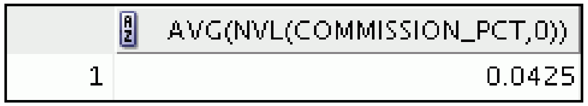


```
     AVG(COMMISSION_PCT)
1                 0.2125
```

The NVL function forces group functions to include null values:

2

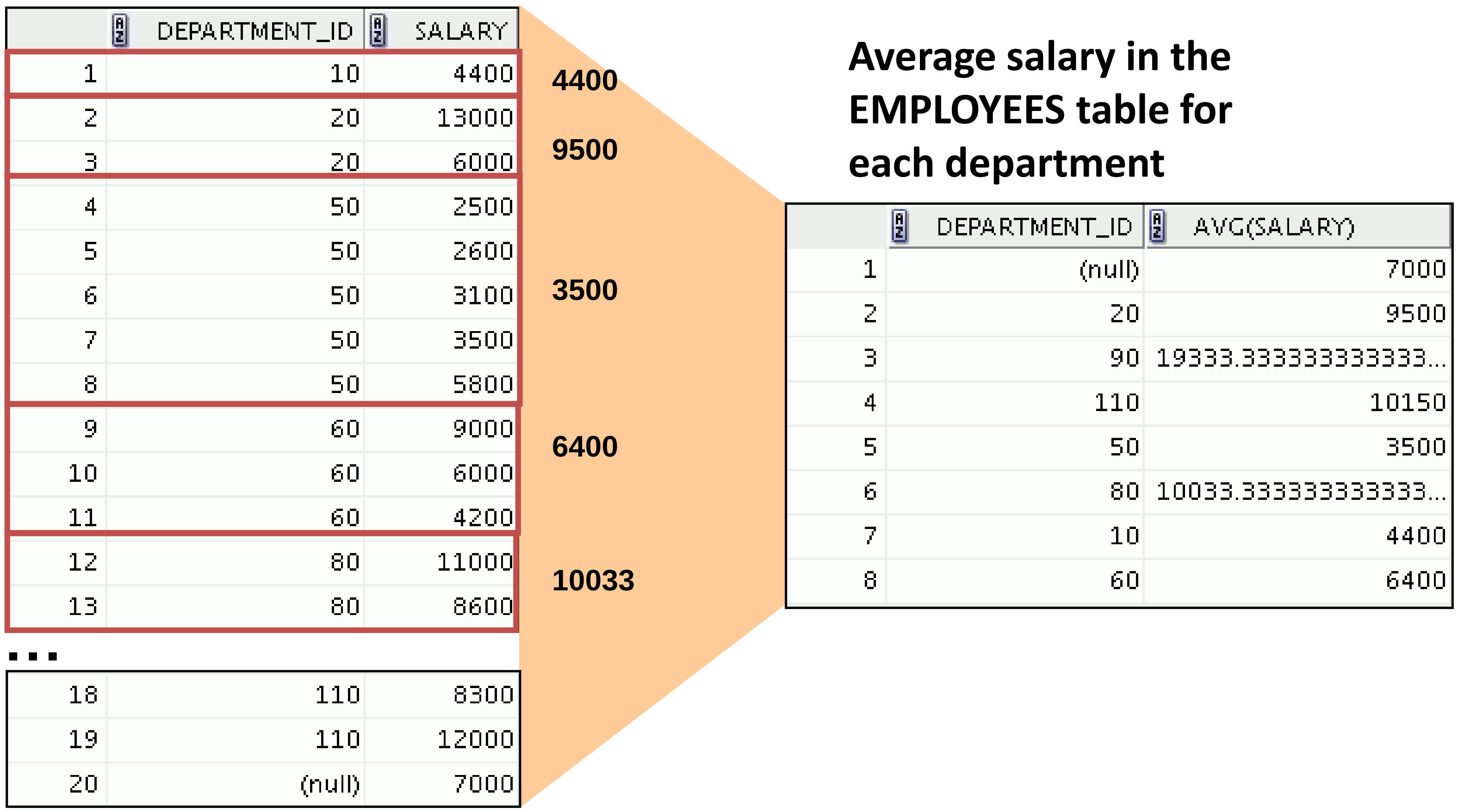
```
SELECT AVG(NVL(commission_pct, 0))
FROM   employees;
```



```
     AVG(NVL(COMMISSION_PCT,0))
1                        0.0425
```

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

Creating Groups of Data: GROUP BY Clause Syntax

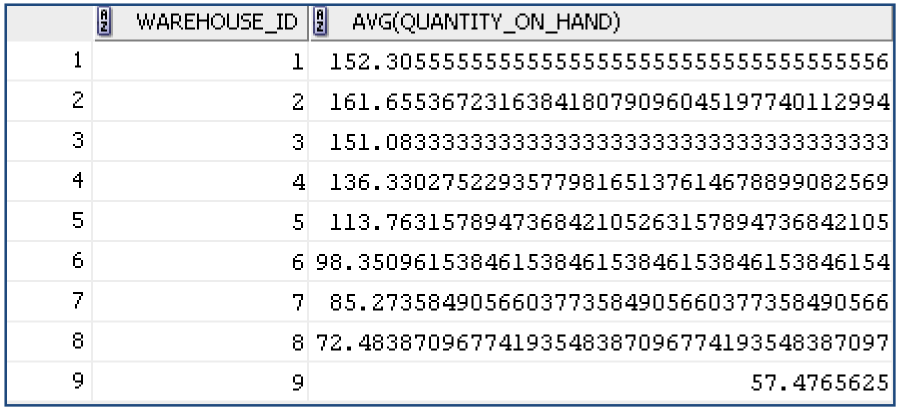
- You can divide rows in a table into smaller groups by using the GROUP BY clause.

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

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 ;
```

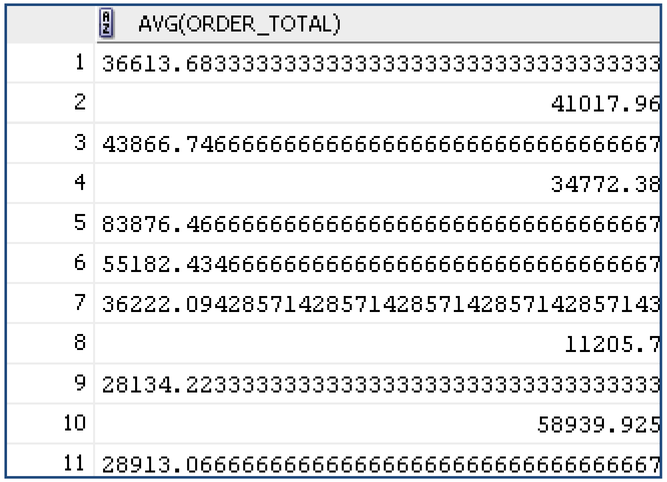


	WAREHOUSE_ID	AVG(QUANTITY_ON_HAND)
1	1	152.305555555555555555555555555555555556
2	2	161.655367231638418079096045197740112994
3	3	151.083333333333333333333333333333333333
4	4	136.330275229357798165137614678899082569
5	5	113.763157894736842105263157894736842105
6	6	98.35096153846153846153846153846153846154
7	7	85.2735849056603773584905660377358490566
8	8	72.48387096774193548387096774193548387097
9	9	57.4765625

Using the GROUP BY Clause

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

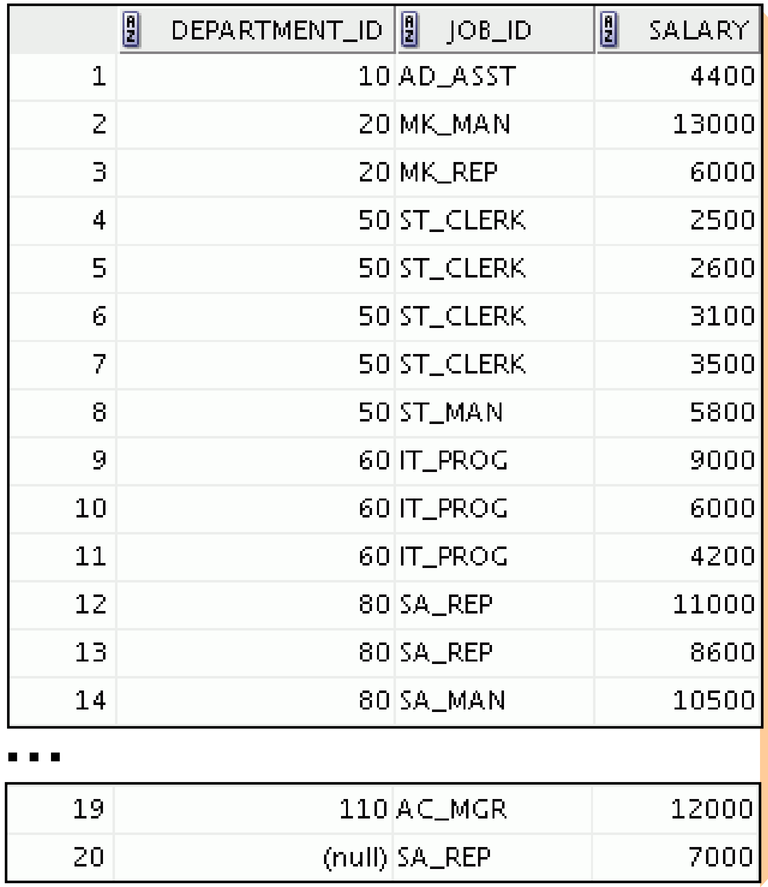
```
SELECT   AVG(order_total)
FROM   orders
GROUP BY   order_status ;
```



	AVG(ORDER_TOTAL)
1	36613.683333333333333333333333333333
2	41017.96
3	43866.746666666666666666666666666667
4	34772.38
5	83876.466666666666666666666666666667
6	55182.434666666666666666666666666667
7	36222.094285714285714285714285714143
8	11205.7
9	28134.223333333333333333333333333333
10	58939.925
11	28913.066666666666666666666666666667

Grouping by More Than One Column

EMPLOYEES

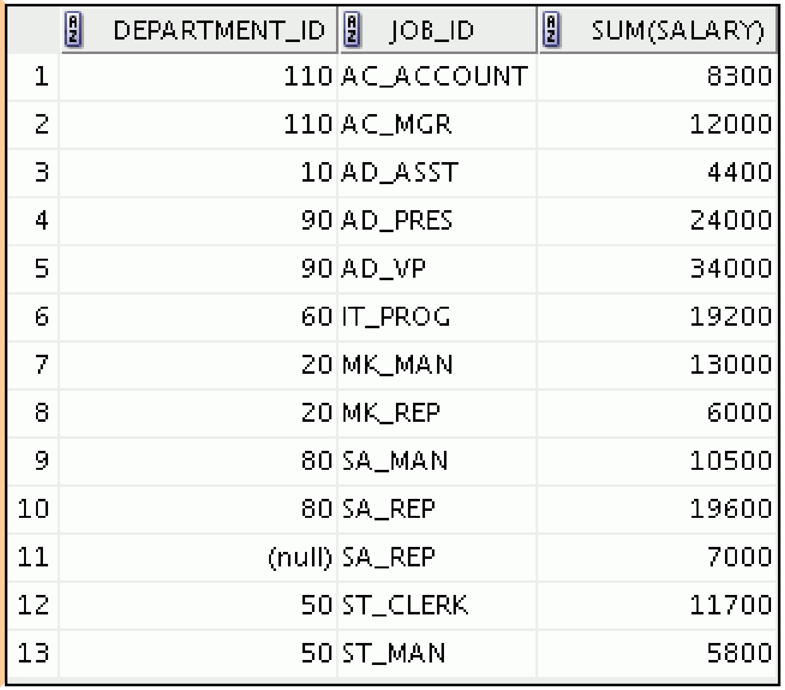


	DEPARTMENT_ID	JOB_ID	SALARY
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_CLERK	2500
5	50	ST_CLERK	2600
6	50	ST_CLERK	3100
7	50	ST_CLERK	3500
8	50	ST_MAN	5800
9	60	IT_PROG	9000
10	60	IT_PROG	6000
11	60	IT_PROG	4200
12	80	SA_REP	11000
13	80	SA_REP	8600
14	80	SA_MAN	10500

...

19	110	AC_MGR	12000
20	(null)	SA_REP	7000

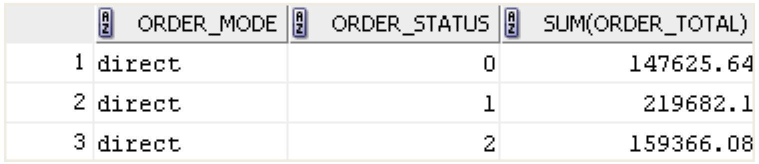
Add the salaries in the EMPLOYEES table for each job, grouped by department.



	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

Using the GROUP BY Clause on Multiple Columns

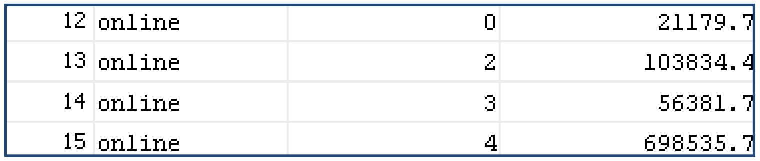
```
SELECT   order_mode, order_status, sum(order_total)
FROM   orders
WHERE   order_id BETWEEN 2300 AND 2500
GROUP   BY order_mode, order_status
ORDER   BY order_mode, order_status ;
```



	ORDER_MODE	ORDER_STATUS	SUM(ORDER_TOTAL)
1	direct	0	147625.64
2	direct	1	219682.1
3	direct	2	159366.08

■ ■ ■

■ ■ ■



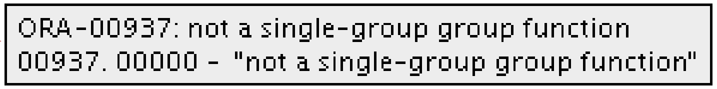
12	online	0	21179.7
13	online	2	103834.4
14	online	3	56381.7
15	online	4	698535.7

■ ■ ■

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:

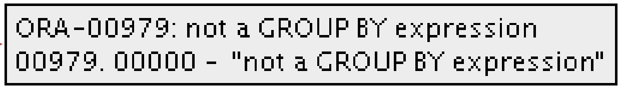
```
SELECT department_id, COUNT(last_name)
FROM   employees;
```



```
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.

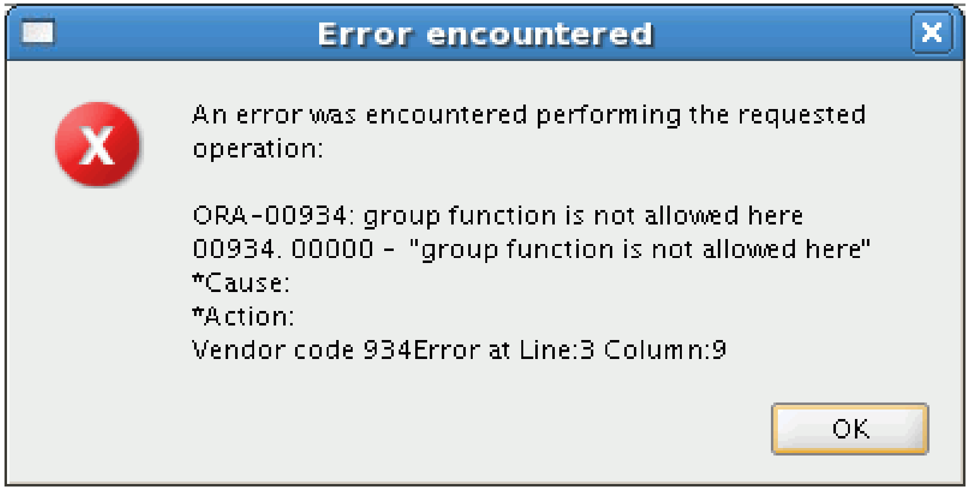
Illegal Queries Using Group Functions

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;
```



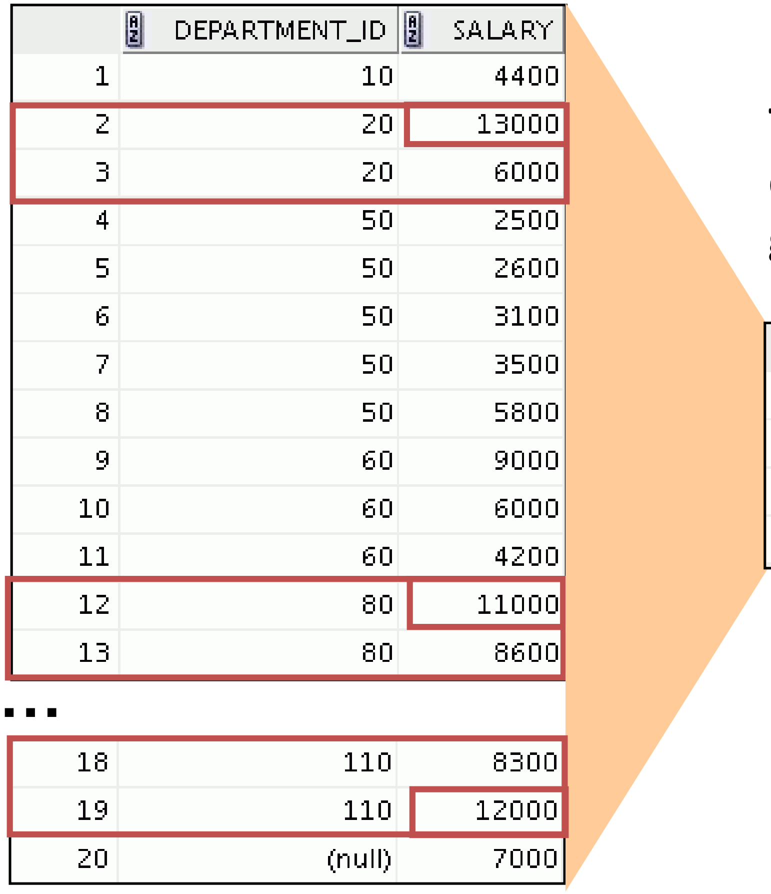
```
An error was encountered performing the requested
operation:

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

Restricting Group Results

EMPLOYEES

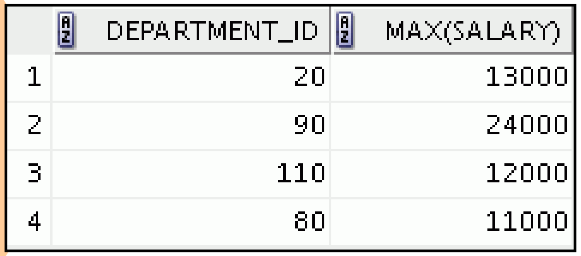


	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	2500
5	50	2600
6	50	3100
7	50	3500
8	50	5800
9	60	9000
10	60	6000
11	60	4200
12	80	11000
13	80	8600

…

18	110	8300
19	110	12000
20	(null)	7000

The maximum salary per department when it is greater than $10,000



	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

Restricting Group Results with the HAVING Clause

- When you use the HAVING clause, the Oracle server restricts groups as follows:

    Rows are grouped.

    The group function is applied.

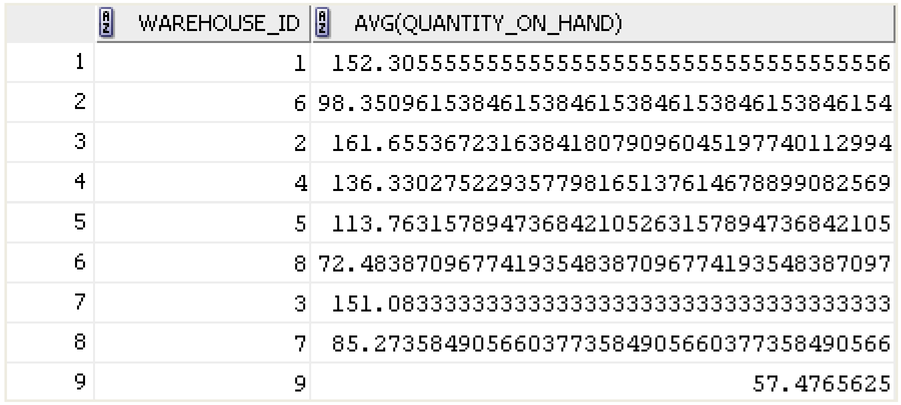
    Groups matching the HAVING clause are displayed.

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

ORACLE

Using the HAVING Clause

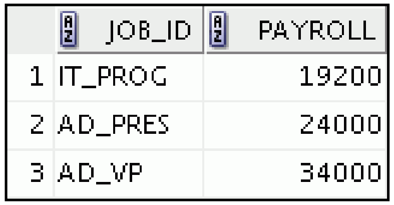
```
SELECT   warehouse_id,  AVG(quantity_on_hand)
FROM   inventories
GROUP BY   warehouse_id
HAVING  MAX  (quantity_on_hand) > 130 ;
```



	WAREHOUSE_ID	AVG(QUANTITY_ON_HAND)
1	1	152.30555555555555555555555555555555556
2	6	98.350961538461538461538461538461538462
3	2	161.65536723163841807909604519774011299
4	4	136.33027522935779816513761467889908257
5	5	113.76315789473684210526315789473684211
6	8	72.483870967741935483870967741935483871
7	3	151.08333333333333333333333333333333333
8	7	85.273584905660377358490566037735849057
9	9	57.4765625

Using the HAVING Clause

```
SELECT    job_id, SUM(salary) PAYROLL
FROM      employees
WHERE     job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING    SUM(salary) > 13000
ORDER BY SUM(salary);
```

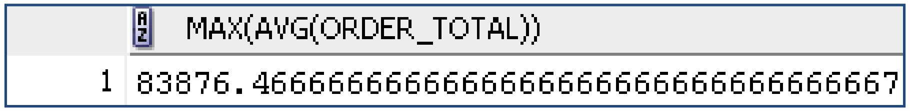


	JOB_ID	PAYROLL
1	IT_PROG	19200
2	AD_PRES	24000
3	AD_VP	34000

Nesting Group Functions

•**Display the maximum average salary:**

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



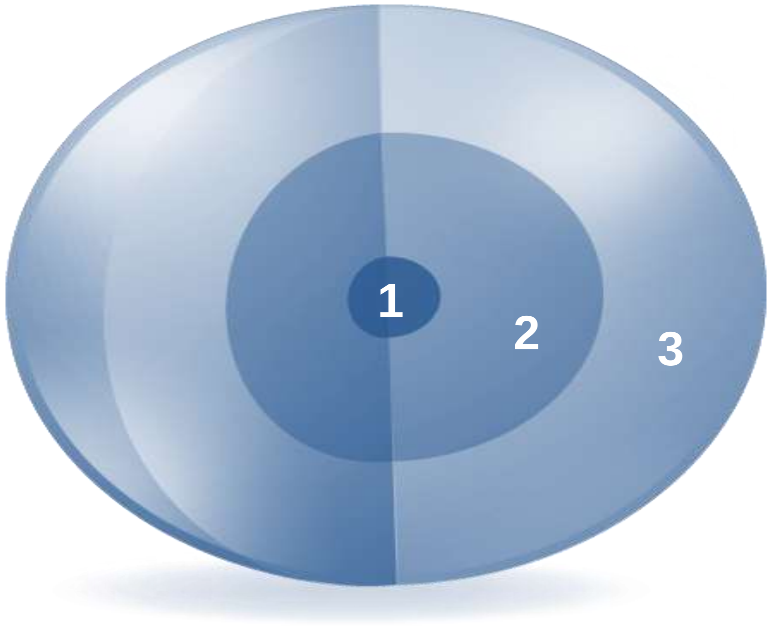
```
     MAX(AVG(ORDER_TOTAL))
1  83876.466666666666666666666666666666667
```

Quiz

•Identify the guidelines for group functions and the GROUP BY clause.

1. You cannot use a column alias in the GROUP BY clause.
2. The GROUP BY column must be in the SELECT clause.
3. By using a WHERE clause, you can exclude rows before dividing them into groups.
4. The GROUP BY clause groups rows and ensures order of the result set.
5. If you include a group function in a SELECT clause, you cannot select individual results as well.

Session Summary

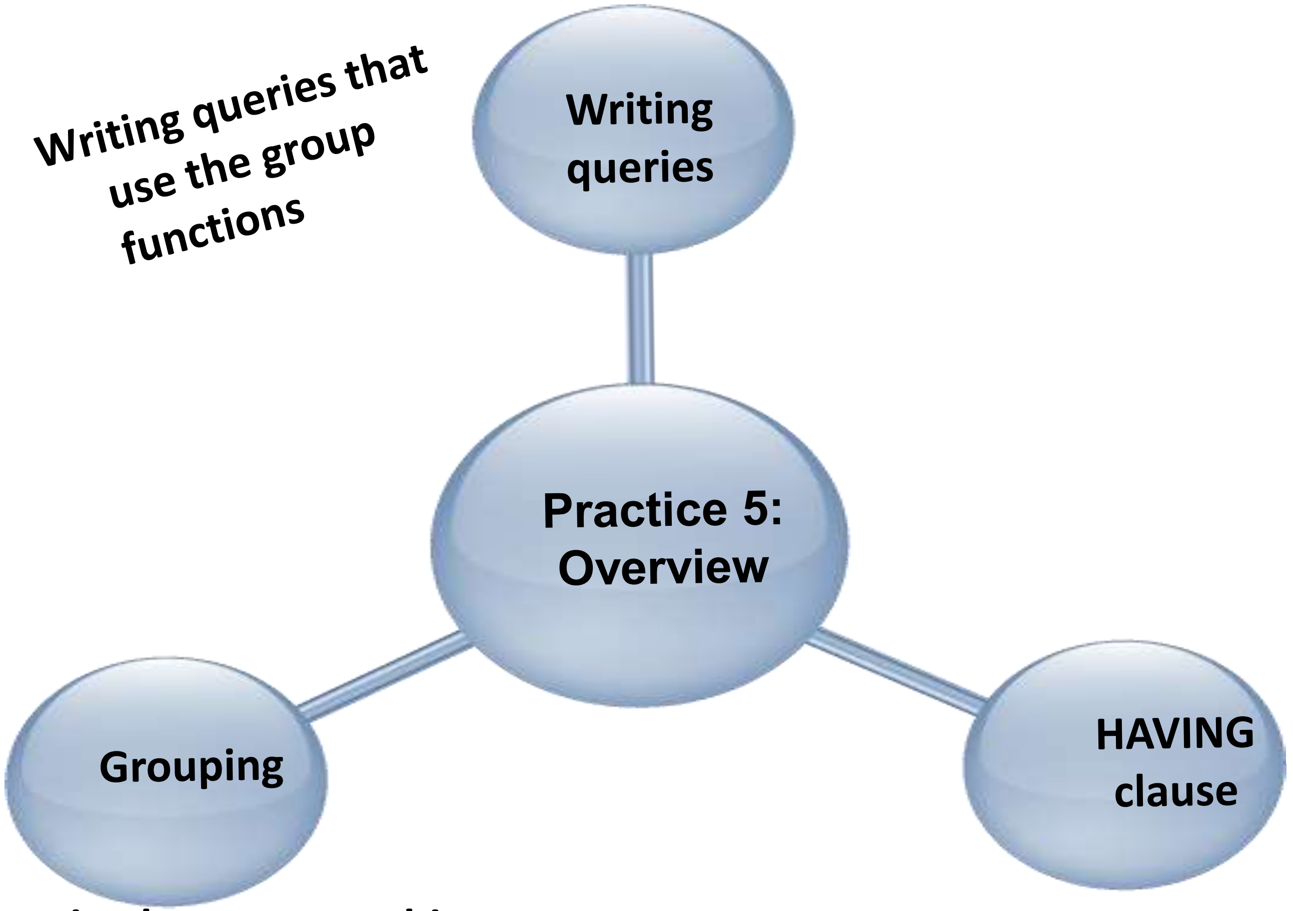


1. **Use the group functions COUNT, MAX, MIN, SUM, and AVG**
2. **Write queries that use the GROUP BY clause**
3. **Write queries that use the HAVING clause**

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

Practice 5: Overview

This practice covers the following topics:



Writing queries that use the group functions

Grouping by rows to achieve more than one result

Restricting groups by using the HAVING clause