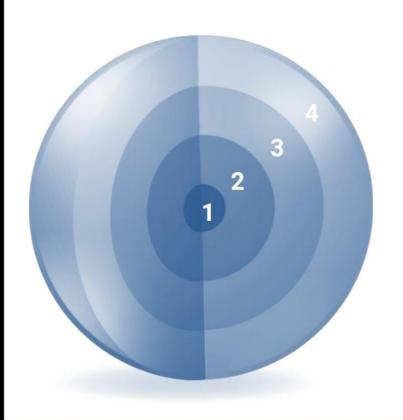
Lesson 5

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

Group functions Grouping rows Nesting group functions

- 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	Maximum salary in
8	90	17000	EMPLOYEES table 24000
9	60	9000	
10	60	6000	
•••			
18	80	11000	
19	80	8600	
20	(null)	7000	

Types of Group Functions

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

ORACLE!

Group

functions

Group Functions: Syntax

```
SELECT group_function(column), ...

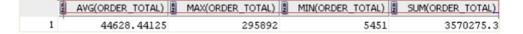
FROM table
[WHERE condition]
[ORDER BY column];
```

Using the ${\tt AVG}$ and ${\tt SUM}$ Functions

•You can use AVG and SUM for numeric data.

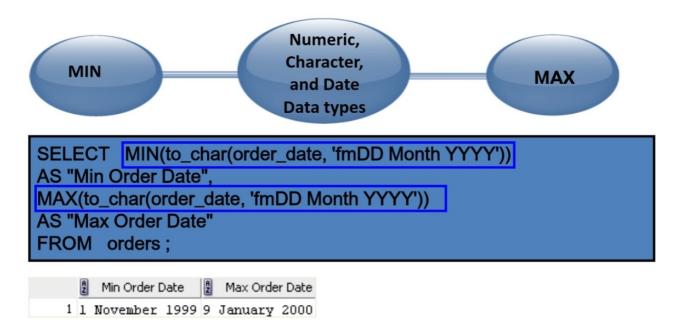


SELECT AVG(order_total), MAX(order_total), MIN (order_total), SUM(order_total)
FROM orders;



Using the MIN and MAX Functions

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



Using the COUNT Function

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

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

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

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



Using the ${\tt DISTINCT}$ Keyword

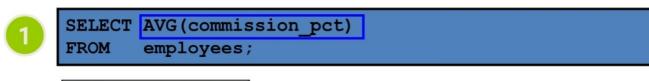
- COUNT(DISTINCT expr) returns the number of distinct non-null values of expr.
- To display the number of distinct department values in the EMPLOYEES table:

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



Group Functions and Null Values

•Group functions ignore null values in the column:



```
AVG(COMMISSION_PCT)

1 0.2125
```

The NVL function forces group functions to include null values:

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

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

1 0.0425
```

Creating Groups of Data

EMPLOYEES

	SALARY	DEPARTMENT_ID	A
4400	4400	10	1
	13000	20	2
9500	6000	20	3
	2500	50	4
	2600	50	5
3500	3100	50	6
	3500	50	7
	5800	50	8
6400	9000	60	9
	6000	60	10
	4200	60	11
10033	11000	80	12
	8600	80	13
	8300	110	18
	12000	110	19
	7000	(null)	20
-			

Average salary in the EMPLOYEES table for each department

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	20	9500
3	90	19333.33333333333
4	110	10150
5	50	3500
6	80	10033.333333333333
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.

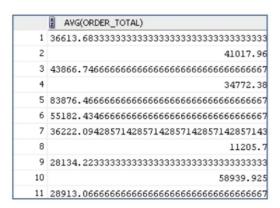
```
SELECT warehouse_id, AVG(quantity_on_hand)
FROM inventories
GROUP BY warehouse_id;
```

	WAREHOUSE_ID	AVG(QUANTITY_ON_HAND)
1	1	152.30555555555555555555555555555555
2	2	161.655367231638418079096045197740112994
3	3	151.08333333333333333333333333333333333
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;



Grouping by More Than One Column

EMPLOYEES

	DEPARTMENT_ID		2 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.

	A	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

. . .

. . .

Γ	12	online	0	21179.7
ľ	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.

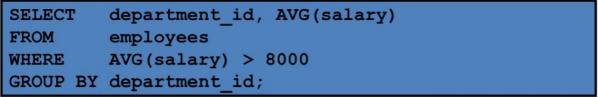


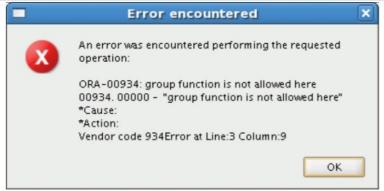
Illegal Queries Using Group Functions

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.





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];
```

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	152.30555555555555555555555555555555
2	6	98.35096153846153846153846153846154
3	2	161.655367231638418079096045197740112994
4	4	136.330275229357798165137614678899082569
5	5	113.763157894736842105263157894736842105
6	8	72.48387096774193548387096774193548387097
7	3	151.0833333333333333333333333333333333
8	7	85.2735849056603773584905660377358490566
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:

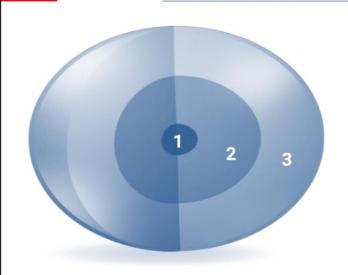
SELECT MAX(AVG(order_total))
FROM orders
GROUP BY order_status;

MAX(AVG(ORDER_TOTAL))

1 83876.46666666666666666666666666666666666

- •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];
```

