

Chapter 5 Aggregate Functions, Group By, and Having

Aggregate Functions

- Are **multi-row** functions
- Work on **groups** of rows and return a **single** result for each group

Function	Description
COUNT(*)	returns the number of rows in the query, including nulls and duplicates
COUNT(DISTINCT ALL expr)	returns the number of nonblank rows
SUM(DISTINCT ALL expr)	returns a numerical sum
AVG(DISTINCT ALL expr)	returns the numerical average
MAX(DISTINCT ALL expr)	returns largest value (any datatype)
MIN(DISTINCT ALL expr)	returns smallest value (any datatype)

Aggregate Functions and NULLS

- Aggregate functions ignore nulls
 - except for COUNT(*)
- Can use NVL or COALESCE to provide a substitute value for a Null



Aggregate Function Practice

- How many records are in the writer table?
- How many writers have a phone number listed?
- How many writers don't have a phone number listed?
- What is the total length and average length of all articles? Also show the length of the longest and shortest article.
- Modify the previous query to show summary results only for business and law articles.

4



GROUP BY Clause

- Divides rows in a table into subgroups
- When used with aggregate functions, the query returns one row of aggregate results for each group
 - vers <9l automatically sorts result rows by the grouping expression

```
SQL> SELECT type, AVG(length)
      FROM article
      GROUP BY type;

TYP  AVG(LENGTH)
-----
ADV          2190.5
BUS    1398.66667
EXP    2031.5743
...
```

5



GROUP BY Clause

- All columns in the SELECT clause that are not group functions must be in the GROUP BY clause
 - error 00979: not a GROUP BY expression

```
SQL> SELECT type, AVG(length)
      FROM article
      GROUP BY type;
```

```
SQL> SELECT type, length, AVG(length)
2  FROM article
3  GROUP BY type;
SELECT type, length, AVG(length)
*
ERROR at line 1:
ORA-00979: not a GROUP BY expression
```

6



GROUP BY Practice

- Show each section and the number of students enrolled in the section.

SECTION_ID	ENROLLMENT
80	1
81	3
82	2
83	2
84	2
85	5
86	6
87	7
88	5
89	12
...	

7



Multi-Level GROUP BY Clause

- How many sections of each course are scheduled in each classroom?

```
SQL> SELECT course_no, location, COUNT(*)
      FROM section
      GROUP BY course_no, location;
COURSE_NO LOCATION          COUNT(*)
-----
10 L214                1
20 L210                2
20 L214                1
20 L509                1
25 L210                2
25 L214                1
25 L507                2
...
```

8



GROUP BY and NULLS

- When a GROUP BY column contains Nulls, they're aggregated together
- By default, Nulls appear at top of sorted results
 - include ORDER BY with NULLS LAST clause to have nulls at bottom

```
SQL> SELECT contact, COUNT(*)
      FROM writer
      GROUP BY contact;
CONT  COUNT(*)
-----
L350  2
L350  8
W432  6
```

```
SQL> SELECT contact, COUNT(*)
      FROM writer
      GROUP BY contact
      ORDER BY contact NULLS LAST;
CONT  COUNT(*)
-----
L350  8
W432  6
      3
```

9

HAVING Clause

- Restricts which **groups** are returned
- Must be used in conjunction with a GROUP BY clause
- The columns in the HAVING clause must appear in the GROUP BY clause or must be aggregate functions

10

HAVING Clause: Example

- For any type of article with an average length exceeding 2000 words, show how many articles there are and the average article length.

```
SQL> SELECT type, COUNT(*), AVG(length)
      FROM article
      GROUP BY type
      HAVING AVG(length) > 2000;
```

TYP	COUNT(TYPE)	AVG(LENGTH)
ADV	2	2190.5
EXP	7	2031.57143
FMR	1	2395
POL	6	2225.5
TEC	1	2222

11

HAVING Practice

- Show each section and the number of students enrolled in the section. Show only sections with fewer than 5 students enrolled.

```
SECTION_ID ENROLLMENT
-----
      80          1
      81          3
      82          2
      83          2
      84          2
      ...
     153          3
     154          4
47 rows selected.
```

12

Using WHERE and HAVING Together

- WHERE specifies which rows are selected from the table to participate in a query
 - executed before GROUP BY
- HAVING specifies which groups to display results for
 - executed after GROUP BY

13

Using WHERE and HAVING Together

- For any article published after the 80's, show how many articles there are and the average length, but only for types whose average length is at least 2000 words.

```
SQL> SELECT type, COUNT(TYPE), AVG(LENGTH)
      FROM article
      WHERE issue >= TO_DATE('01/01/1990', 'mm/dd/yyyy')
      GROUP BY type
      HAVING AVG(length) > 2000;
```

TYP	COUNT(TYPE)	AVG(LENGTH)
ADV	1	3285
EXP	3	2122.33333
LAW	2	2303.5
POL	2	3981
TEC	1	2222

14

Clause Order

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
```

15
