JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Database Management

BU.330.770

Session 7 (part I)

Instructor: Changmi Jung, Ph.D.

Group Functions

Announcement

- Final exam is next week
  - 2 hours in total
  - Around 50 - 55 questions
  - In-person test (No exception)
  - Please arrive 2-3 minutes before class
  - Cheat sheet: one letter size sheet is allowed.
    - Use both front and back, Letter size
    - No restrictions on the font size, margin, line space, etc.
    - Handwritten, printed, all allowed, but it must be your creation
    - Write your name in the top right corner
    - Must submit it to the proctor before you exit the classroom

Session Objectives (1/2)

- Differentiate between single-row and multiple-row functions
- Use the SUM and AVG functions for numeric calculations
- Use the COUNT function to return the number of records containing non-NULL values
- Use COUNT(*) to include all records including NULL values
- Use the MIN and MAX functions with non-numeric fields

works with → Datetype
→ character
→ numeric

Session Objectives (2/2)

- Determine when to use the GROUP BY clause to group data
- Identify when the HAVING clause should be used
- Understand the order for evaluating WHERE, GROUP BY, and HAVING clauses
- Nest a group function inside of a single-row function
- Calculate the standard deviation and variance of a set of data using the STDDEV and VARIANCE functions

Group Functions

- Return one result per group of rows processed
- Are also called multiple-row or aggregate functions
- All group functions ignore NULL values except COUNT(*)
- Use DISTINCT to suppress duplicate values

Added Clauses to Perform Group Functions

SELECT * | *columnname, columnname*…

FROM *tablename*

[WHERE *condition*]

[GROUP BY *columnname, columnname*…]

[HAVING *group condition*]

[ORDER BY *columnname, columnname*…];

Added clauses for Group functions

SUM Function

- Calculates the total amount stored in a numeric column for a group of rows

Total sales made on a particular date?

Worksheet | Query Builder

```
SELECT SUM(paideach * quantity) "Total_Sales"
FROM orderitems oi JOIN orders o USING(order#)
WHERE orderdate = '02-APR-19';
```

Query Result

SQL | All Rows Fetched: 1 in 0.085 seconds

	Total_Sales
1	375.75

AVG Function

- Calculates the average of numeric values in a specified column

Worksheet | Query Builder

```
SELECT AVG(retail-cost) "Average Profit"
FROM books
WHERE category = 'COMPUTER';
```

Query Result

SQL | All Rows Fetched: 1 in 0.11 seconds

	Average Profit
1	18.2625

Average profit generated by all books in the Computer

Back to Grouping

AVG Function with NULL?

- Group functions ignore NULL: records containing NULL value in a specified column will be dropped from the aggregation

Worksheet | Query Builder

```
SELECT empno, lname, mthsal, bonus
FROM employees;
```

Query Result

SQL | All Rows Fetched: 5 in 0.035 seconds

	EMPNO	LNAME	MTHSAL	BONUS
1	7839	KING	6000	3000
2	8888	JONES	4200	1200
3	7344	SMITH	4900	1500
4	7355	POTTS	4900	1900
5	8844	STUART	3700	(null)

Stuart's record will be omitted from AVG(bonus) calculation

Worksheet | Query Builder

```
SELECT (3000+1200+1500+1900)/4
FROM dual;
```

Query Result

SQL | All Rows Fetched: 1 in 0.023 seconds

	(3000+1200+1500+1900)/4
1	1900

Use NVL Function to Address NULL

- Replace null values with a specified value for a given column

Dropping Stuart's record

Worksheet | Query Builder

```
SELECT avg(bonus)
FROM employees;
```

Query Result

SQL | All Rows Fetched: 1 in 0.023 seconds

	(3000+1200+1500+1900)/4
1	1900

Replace Stuart's bonus with zero

Worksheet | Query Builder

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

Query Result

SQL | All Rows Fetched: 1 in 0.02 sec

	AVG(NVL(BONUS,0))
1	1520

If the NULL value represents a bonus of zero, it must be included in the calculation, right?

COUNT Function

- Count non-NULL values: specify a column name as an argument
  - Ex. **COUNT(bonus)**: count the number of records that include any value (not null) in the column *bonus*.

- Count total records, including those with NULL values: use an asterisk as an argument
  - Ex. **COUNT(*)**

Worksheet | Query Builder

```
SELECT COUNT(*), COUNT(bonus)
FROM employees;
```

Query Result

SQL | All Rows Fetched: 1 in 0.133 seconds

	COUNT(*)	COUNT(BONUS)
1	5	4

COUNT Function: Non-NULL Values

- Include column name in argument to count number of occurrences

How many unique categories exist in BOOKS table?

Worksheet | Query Builder

```
SELECT COUNT(DISTINCT category) "Number of Categories"
FROM books;
```

Query Result

SQL | All Rows Fetched: 1 in 0.023 seconds

	Number of Categories
1	8

COUNT Function: Count all with *

- Include asterisk in argument to count the total number of rows

Worksheet | Query Builder

```
SELECT count(*)
FROM orders
WHERE shipdate IS NULL;
```

Query Result

SQL | All Rows Fetched: 1 in 0.02 se

	COUNT(*)
1	6

Out of the total 21 records in the ORDERS table, 6 records contain a NULL value in the *shipdate* column.

Can you count the number of orders that have not been shipped yet by using the SUM function?

MAX Function

- Returns the largest value in the specified column

Worksheet | Query Builder

```
SELECT MAX(retail-cost) "Highest Profit"
FROM books;
```

Query Result

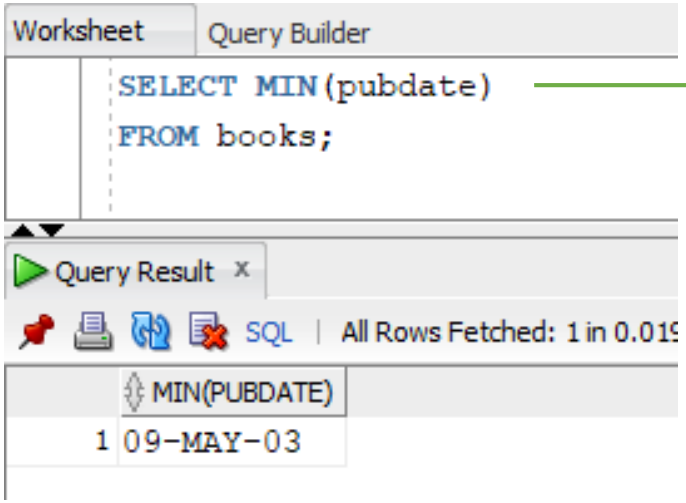
SQL | All Rows Fetched: 1 in 0.059 seconds

	Highest Profit
1	41.95

We can't tell which book generates this maximum profit of $41.95. What will happen if you add the title column in the SELECT clause?

MIN Function

- Returns the smallest value in the specified column



Finds the earliest publication date

Note: **COUNT, MIN**, and **MAX** functions can be used on values with character, numeric, and date datatypes

Grouping Data

- **GROUP BY** clause rules
  - Used to group data
  - If a group function is used in the SELECT clause, any single (non-aggregate) columns listed in the SELECT clause must be listed in the GROUP BY clause
  - Columns used in the GROUP BY clause don't have to be listed in the SELECT clause. (But I do recommend listing it for identification purposes)
  - Cannot reference column aliases

What if we need the average profit for each and every category, not just for the Computer category?

GROUP BY Example

Non-aggregate single column

Worksheet | Query Builder

```
SELECT category, TO_CHAR(AVG(retail - cost), '$999.99') "Profit"
FROM books
GROUP BY category;
```

Query Result

SQL | All Rows Fetched: 8 in 0.017 seconds

	CATEGORY	Profit
1	FITNESS	$12.20
2	FAMILY LIFE	$24.88
3	CHILDREN	$12.89
4	COMPUTER	$18.26
5	COOKING	$8.60
6	SELF HELP	$12.10
7	BUSINESS	$16.55
8	LITERATURE	$18.10

Average profit for each category of books

Common Error

- A common error is missing a GROUP BY clause for non-aggregated columns in the SELECT clause

Worksheet | Query Builder

```
SELECT category, TO_CHAR(AVG(retail - cost), '$999.99') "Profit"
FROM books;
```

Query Result

SQL | Executing:SELECT category, TO_CHAR(AVG(retail - cost), '$999.99') "Profit"FROM books in 0

```
ORA-00937: not a single-group group function
00937. 00000 - "not a single-group group function"
*Cause:
*Action:
Error at Line: 67 Column: 8
```

Non-aggregated (single) column must be included in the GROUP BY clause

Restricting Aggregated Output

- HAVING clause serves as the WHERE clause for grouped data

HAVING *group_function comparison_operator value*

Worksheet | Query Builder

```
SELECT category, TO_CHAR(AVG(retail-cost), '$999.99') "Profit"
FROM books
GROUP BY category
HAVING AVG(retail-cost) > 15;
```

Can use the column alias Profit

Query Result

SQL | All Rows Fetched: 4 in 0.023 seconds

	CATEGORY	Profit
1	FAMILY LIFE	$24.88
2	COMPUTER	$18.26
3	BUSINESS	$16.55
4	LITERATURE	$18.10

Display a list of book categories with an average profit of more than $15

The Order of Evaluation

- When a SELECT statement includes all three clauses, the clauses are evaluated in the order of:

  1. WHERE
  2. GROUP BY
  3. HAVING

The Order of Evaluation in Practice

Worksheet | Query Builder

```
SELECT category, TO_CHAR(AVG(retail-cost), '$999.99') "Profit"
FROM books
WHERE pubdate > '01-JAN-15'
GROUP BY category
HAVING AVG(retail-cost) > 15;
```

1 → WHERE
2 → GROUP BY
3 → HAVING

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.023 seconds

	CATEGORY	Profit
1	COMPUTER	$16.17
2	LITERATURE	$18.10

Nesting Group Functions

- Inner function is resolved first
- Maximum nesting depth: 2

How to calculate the average sales amount per order?

Worksheet | Query Builder

```
SELECT AVG(SUM(paideach * quantity)) "Average Order Amount"
FROM orderitems
GROUP BY order#;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.016 seconds

	Average Order Amount
1	80.3833333333333333333333333333333333333

Statistical Group Functions

- Based on normal distribution
- Includes:
  - STDDEV: calculates the standard deviation for a specified field
  - VARIANCE: calculates the variance for a specified field
  - MEDIAN: find the median value for a specified field

STDDEV, VARIANCE, etc.

Worksheet | Query Builder

```
SELECT category, COUNT(*) total,
       TO_CHAR(AVG(retail-cost), '$999.99') "Avg_Profit",
       TO_CHAR(STDDEV(retail-cost), '999.9999') "Std Dev",
       TO_CHAR(VARIANCE(retail-cost), '999.9999') "Variance",
       MEDIAN(retail-cost) "Median",
       MIN(retail-cost) "Minimum",
       MAX(retail-cost) "Maximum"
FROM books
GROUP BY category;
```

Query Result

SQL | All Rows Fetched: 8 in 0.022 seconds

	CATEGORY	TOTAL	Avg_Profit	Std Dev	Variance	Median	Minimum	Maximum
1	BUSINESS	1	$16.55	.0000	.0000	16.55	16.55	16.55
2	CHILDREN	2	$12.89	13.0956	171.4952	12.89	3.63	22.15
3	COMPUTER	4	$18.26	11.2267	126.0390	20.575	3.2	28.7
4	COOKING	2	$8.60	1.6263	2.6450	8.6	7.45	9.75
5	FAMILY LIFE	2	$24.88	24.1477	583.1113	24.875	7.8	41.95
6	FITNESS	1	$12.20	.0000	.0000	12.2	12.2	12.2
7	LITERATURE	1	$18.10	.0000	.0000	18.1	18.1	18.1
8	SELF HELP	1	$12.10	.0000	.0000	12.1	12.1	12.1

Summary (1/2)

- The AVG, SUM, STDDEV, and VARIANCE functions are used only with numeric fields
- The COUNT, MAX, and MIN functions can be applied to any datatype

  dtypes allowed → Numeric, Date, characters
- The AVG, SUM, MAX, MIN, STDDEV, and VARIANCE functions all ignore NULL values
- By default, the AVG, SUM, MAX, MIN, COUNT, STDDEV, and VARIANCE functions include duplicate values
- The STDDEV and VARIANCE functions are used to perform statistical analyses on a set of data

Summary (2/2)

- The GROUP BY clause is used to divide table data into groups
- If a SELECT clause contains both an individual (single) field name and a group function, the non-aggregated (single) field name must also be included in a GROUP BY clause
- The HAVING clause is used to restrict groups in a group function
- Group functions can be nested to a depth of only two. The inner function is always performed first, using the specified grouping. The results of the inner function are used as input for the outer function.