Oracle SQL Chapter 7: Group Functions

Certification Objective 7.01

Overview of Group Functions

Group functions return one result for a group of rows. They are essential for data aggregation and analysis in Oracle SQL.

Types of Group Functions

1. **Aggregate Functions** - Operate on entire result sets (multirow)
2. **Analytic Functions** - Process row subsets with awareness of their position

> **Important Rule**: You cannot mix scalar functions (e.g., `ROUND(price)`) and group functions (e.g., `AVG(price)`) at the same aggregation level without proper nesting.

Common Group Functions

Function	Description	Aggregate	Analytic
COUNT, SUM, MIN, MAX, AVG	Basic statistical functions	Yes	Yes
MEDIAN	Middle value of sorted set	Yes	No*
VARIANCE, STDDEV	Statistical measures	Yes	Yes
RANK, DENSE_RANK	Ranking functions	Yes	Yes
GROUP_ID, GROUPING	Grouping features	Yes	No

*Note: MEDIAN is not supported in analytic form; use RANK/DENSE_RANK instead.

Detailed Function Reference

COUNT(expr)

- Counts **non-NULL** values in the expression
- `COUNT(*)` counts **all rows**, including those with only NULL values
- `COUNT(DISTINCT col)` ignores duplicate values
- `COUNT(ALL col)` includes duplicates (default behavior)
- **Never returns NULL** - returns 0 if no rows match

SUM(expr)

- Adds up numeric values (ignores NULL values)
- Example:

```sql
SELECT SUM(subtotal) FROM orders;
```

MIN(expr) and MAX(expr)

- Work with numeric, date, and character data types
- NULL values are ignored unless all values are NULL
- Ordering rules: numeric < date < character

AVG(expr)

- Returns average of numeric column values
- Ignores NULL values in calculation
- Can be nested with scalar functions:

```sql
SELECT ROUND(AVG(salary), 2) FROM pay_history;
```

MEDIAN(expr)

- Returns the middle value of a sorted set
- Ignores NULL values
- Interpolates when there's an even number of values
- Example:

```sql
SELECT MEDIAN(a) FROM test_median;
```

RANK() and DENSE_RANK()

As Analytic Functions:

```sql
RANK() OVER (PARTITION BY col1 ORDER BY col2)
DENSE_RANK() OVER (PARTITION BY col1 ORDER BY col2)
```

- RANK: Creates gaps in ranking (e.g., 1,1,1,4)
- DENSE_RANK: No gaps in ranking (e.g., 1,1,1,2)

As Aggregate Functions:

```sql
RANK(c1) WITHIN GROUP (ORDER BY e1)
DENSE_RANK(c1) WITHIN GROUP (ORDER BY e1)
```

- Matches c1 against ordered e1 set and returns rank

FIRST and LAST

Returns first or last value after ordering:

```sql
aggregate_function KEEP (DENSE_RANK FIRST|LAST ORDER BY expr)
```

Example:

```sql
SELECT MAX(sq_ft) KEEP (DENSE_RANK FIRST ORDER BY guests)
FROM ship_cabins;
```

GROUP BY Clause

Purpose

- Groups rows that share common values
- Creates "mini SELECTs" within a main SELECT
- Enables aggregate functions to operate on each group

Syntax

```sql
SELECT column1, AGG_FUNC(column2)
FROM table
WHERE condition
GROUP BY column1;
```

Key Rules

- Columns in SELECT must either be:
  - Part of the GROUP BY clause, or
  - Used in an aggregate function
- GROUP BY doesn't require selected columns to be in the SELECT list

Examples

Basic Grouping:

```sql
SELECT ROOM_STYLE, ROUND(AVG(SQ_FT), 2)
FROM SHIP_CABINS
WHERE SHIP_ID = 1
GROUP BY ROOM_STYLE;
```

Multiple Aggregates:

```sql
SELECT ROOM_STYLE,
       ROUND(AVG(SQ_FT), 2) AS avg_sq_ft,
       MIN(GUESTS) AS min_guests,
       COUNT(SHIP_CABIN_ID) AS total_cabins
FROM SHIP_CABINS
WHERE SHIP_ID = 1
GROUP BY ROOM_STYLE;
```

Grouping by Multiple Columns:

```sql
SELECT ROOM_STYLE, ROOM_TYPE, COUNT(*)
FROM SHIP_CABINS
WHERE SHIP_ID = 1
GROUP BY ROOM_STYLE, ROOM_TYPE;
```

- Groups by combinations of values
- Order matters: groups by ROOM_STYLE first, then by ROOM_TYPE

ORDER BY with GROUP BY

- Must use columns from GROUP BY or aggregate functions
- Can sort by position or alias:

```sql
ORDER BY 2 DESC  -- sorts by second item in SELECT
```

HAVING Clause

Purpose

- Filters **groups** of rows after GROUP BY is applied
- Acts like WHERE clause but for groups, not individual rows

Key Rules

- Can only be used with GROUP BY
- Must appear **after** GROUP BY and **before** ORDER BY
- Cannot reference individual rows - must refer to grouped data or aggregates

Valid Expressions

- Aggregate functions (MIN, MAX, COUNT, SUM, etc.)
- Scalar functions applied to groups or aggregates
- Boolean operators (AND, OR, NOT)

Example

sql

```sql
SELECT ROOM_STYLE, ROOM_TYPE, TO_CHAR(MIN(SQ_FT), '9,999') "Min"
FROM SHIP_CABINS
WHERE SHIP_ID = 1
GROUP BY ROOM_STYLE, ROOM_TYPE
HAVING ROOM_TYPE IN ('Standard', 'Large') OR MIN(SQ_FT) > 1200
ORDER BY 3;
```

This query:

1. Groups data by ROOM_STYLE and ROOM_TYPE
2. Filters groups to include only those where:
   - ROOM_TYPE is 'Standard' or 'Large', OR
   - MIN(SQ_FT) > 1200

SQL Clause Order

Sequence	Clause	Required?	Notes
1	SELECT	Required	
2	FROM	Required	
3	WHERE	Optional	Filters rows before grouping
4	GROUP BY	Optional	Needed if using aggregates or grouping
5	HAVING	Optional	Filters after grouping (requires GROUP BY)
6	ORDER BY	Optional	Sorts the result set

Function Nesting Rules

Scalar Functions

- Can be nested unlimited times
- Example: `TO_CHAR(ROUND(AVG(salary), 2), '999,999')`

Aggregate Functions

- Can only be nested up to 2 levels
- ✅ Valid: `ROUND(AVG(MAX(SQ_FT)))`
- ❌ Invalid: `COUNT(AVG(MAX(SQ_FT)))` - Error: too deeply nested

Mixed Nesting

- Scalar functions can wrap aggregate functions:

```sql
SELECT TO_CHAR(MEDIAN(SQ_FT), '999.99') FROM SHIP_CABINS;
```

Common Errors

ORA-00937

- Occurs when mixing grouped and ungrouped columns in SELECT
- Fix: Ensure every selected column is either:
  - In the GROUP BY clause, or
  - Used in an aggregate function

Usage Guidelines

Where Group Functions Can Be Used:

- SELECT clause
- ORDER BY clause
- GROUP BY clause
- HAVING clause

Best Practices:

1. Be careful mixing scalar and group functions without proper nesting
2. DISTINCT/ALL can be used with COUNT, AVG, etc., but not with COUNT(*)
3. Remember that aggregate functions return one row per group
4. Use nesting to create higher-level summaries of your data