# Chapter 11: Oracle SQL Set Operators

## Overview of Set Operators

Oracle SQL provides four set operators for combining results from multiple SELECT statements:

| Operator | Description |
|---|---|
| **UNION** | Combines row sets from two SELECTs and removes duplicates |
| **UNION ALL** | Combines row sets from two SELECTs and keeps all rows including duplicates |
| **INTERSECT** | Returns only rows that appear in both SELECT results |
| **MINUS** | Returns rows from the first SELECT that are not in the second SELECT |

## Rules for Using Set Operators

When using set operators, you must follow these rules:

1. **Column Count**: The number of columns must match in all SELECT statements
2. **Data Types**: Corresponding columns must have compatible data types (from the same group, e.g., numeric with numeric)
3. **Column Names**: Column names don't need to match; the result set uses column names from the first SELECT
4. **ORDER BY**: Must appear only once, after the last SELECT in the series
5. **LOB Restrictions**: BLOB and CLOB data types cannot be used with set operators
6. **No Join Requirements**: SELECT statements don't require joins, keys, or matching column/table names—only compatible data types and column counts

## Set Operator Behavior

### UNION

- Automatically removes duplicate rows
- Checks for uniqueness across all columns in the row
- Final column names come from the first SELECT statement
- More processing overhead due to duplicate removal

### UNION ALL

- Keeps all rows from both queries, including duplicates

- No filtering for uniqueness
- More performance-efficient than UNION
- Use when duplicates are acceptable or when you know there are no duplicates

### INTERSECT

- Returns only rows that exist in both SELECT statements
- Automatically removes duplicates from the result
- Useful for finding common data between two queries

### MINUS

- Returns rows from the first SELECT that don't exist in the second
- Order matters: A MINUS B ≠ B MINUS A
- Also removes duplicates from the result

## Combining Multiple Set Operators

You can chain multiple set operators together:

```sql
SELECT ...
UNION
SELECT ...
INTERSECT
SELECT ...
```

### Precedence and Parentheses

- All set operators have equal precedence
- Execution occurs from left to right by default
- Use parentheses to control execution order:

```sql
SELECT ...
UNION (
  SELECT ...
  INTERSECT
  SELECT ...
)
```

## ORDER BY with Set Operators

### General Rules

- Only one ORDER BY clause is allowed per set operation
- Must appear at the end of the entire set operation
- Sorting is based on the output columns from the first SELECT
- Can sort by:
  - Column position (number)
  - Column reference (name or alias from first SELECT)

### ORDER BY Using Column Position

Use column numbers (starting at 1) from the SELECT list:

```sql
SELECT 'Individual',
       LAST_NAME || ', ' || FIRST_NAME
FROM CRUISE_CUSTOMERS
UNION
SELECT CATEGORY,
       VENDOR_NAME
FROM VENDORS
ORDER BY 2; -- Sort by the second column
```

This approach is useful when column names differ across SELECT statements.

### ORDER BY Using Column Reference

Use a column alias or column name from the first SELECT:

```sql
SELECT 'Individual' CONTACT_CATEGORY,
       LAST_NAME || ', ' || FIRST_NAME POINT_OF_CONTACT
FROM CRUISE_CUSTOMERS
UNION
SELECT CATEGORY,
       VENDOR_NAME
FROM VENDORS
ORDER BY POINT_OF_CONTACT;
```

You must alias columns in the first SELECT if you want to reference them by name in ORDER BY.

## Practical Considerations

### Working with Primary Keys

- Be cautious when combining SELECTs that include primary key values from different tables
- Only use them together if the keys are semantically related (they usually aren't)

### Duplicate Row Handling

- UNION checks entire rows for uniqueness, not individual columns
- Rows with different combinations of column values are considered unique
- If you need all rows regardless of duplicates, use UNION ALL for better performance

### Use Cases

Set operators are particularly useful when:

- Combining data from unrelated tables (no FK/PK relationship)
- Creating reports that merge different data sources
- Finding common or different records between datasets
- Building complex queries without joins

## Key Exam Points

When answering questions about set operators:

- Remember that ORDER BY must be the final clause
- ORDER BY can only reference columns from the first SELECT
- All set operators require matching column counts and compatible data types
- UNION removes duplicates; UNION ALL keeps them
- MINUS result depends on the order of SELECT statements