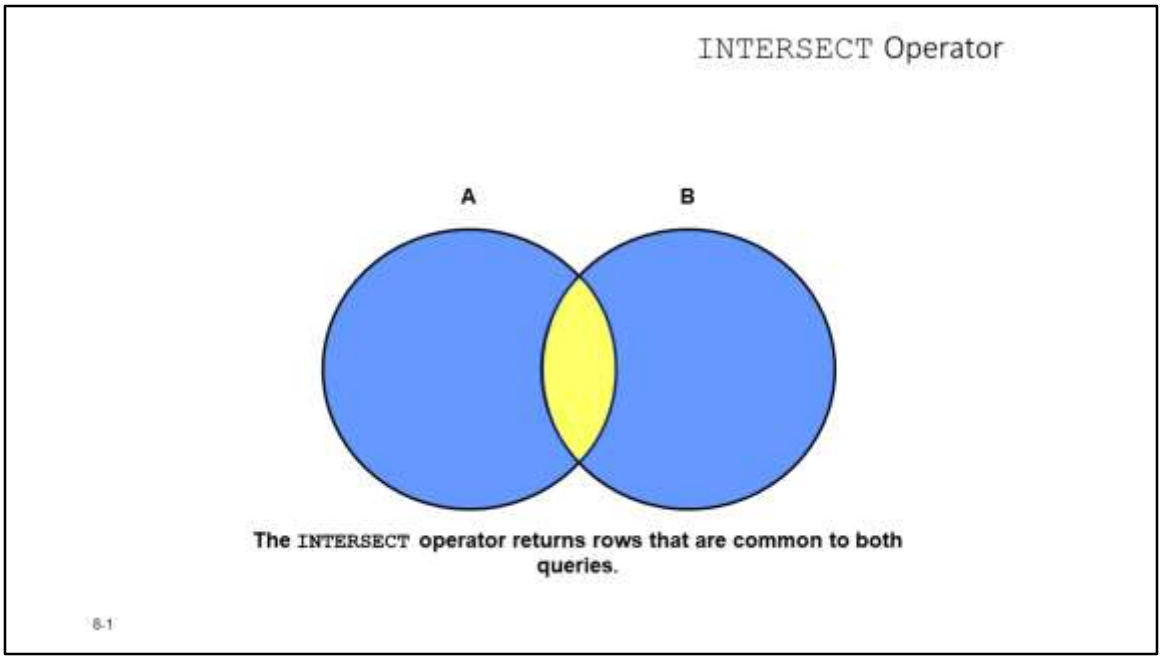


INTERSECT Operator

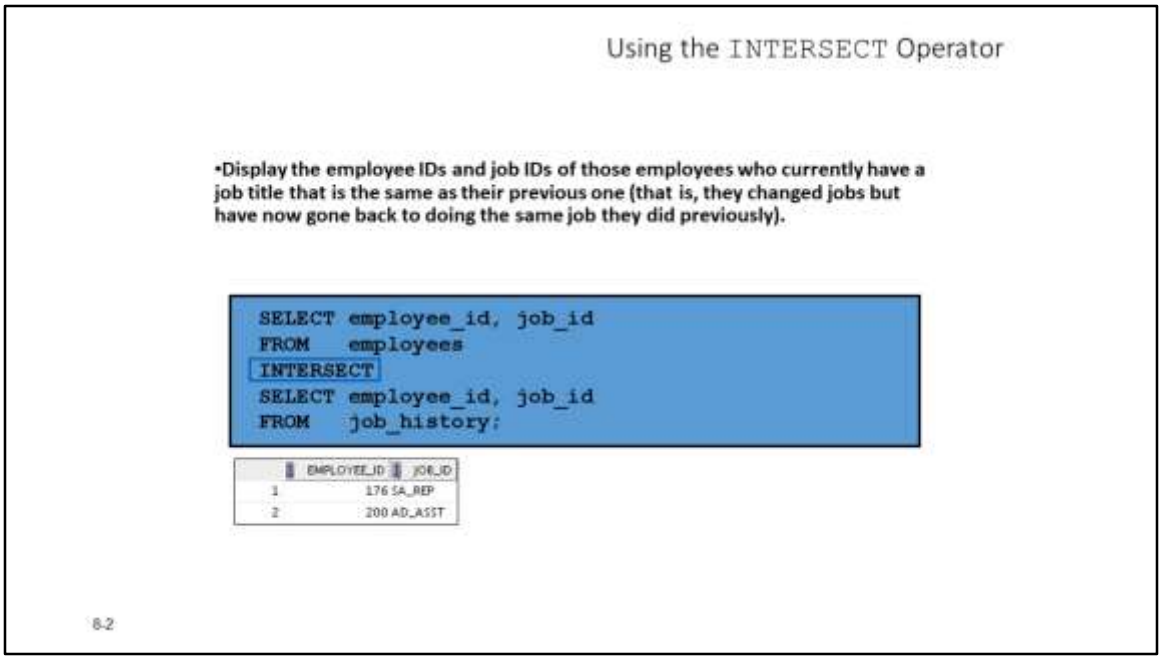
Use the INTERSECT operator to return all rows that are common to multiple queries.

Guidelines

The number of columns and the data types of the columns being selected by the SELECT statements in the queries must be identical in all the SELECT statements used in the query. The names of the columns, however, need not be identical.

Reversing the order of the intersected tables does not alter the result.

INTERSECT does not ignore NULL values.



Using the `INTERSECT` Operator

In the example in this slide, the query returns only those records that have the same values in the selected columns in both tables.

What will be the results if you add the `DEPARTMENT_ID` column to the `SELECT` statement from the `EMPLOYEES` table and add the `DEPARTMENT_ID` column to the `SELECT` statement from the `JOB_HISTORY` table, and run this query? The results may be different because of the introduction of another column whose values may or may not be duplicates.

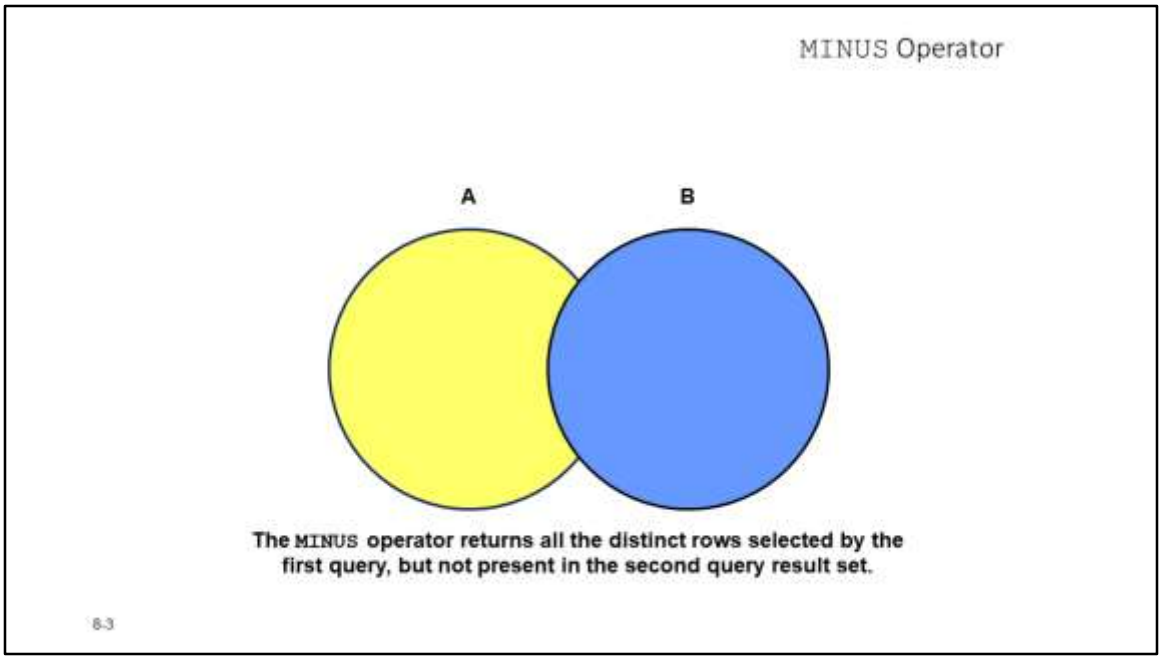
Example:

```
SELECT employee_id, job_id, department_id
FROM   employees
INTERSECT
SELECT employee_id, job_id, department_id
FROM   job_history;
```

Er

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	176	SA_REP	80

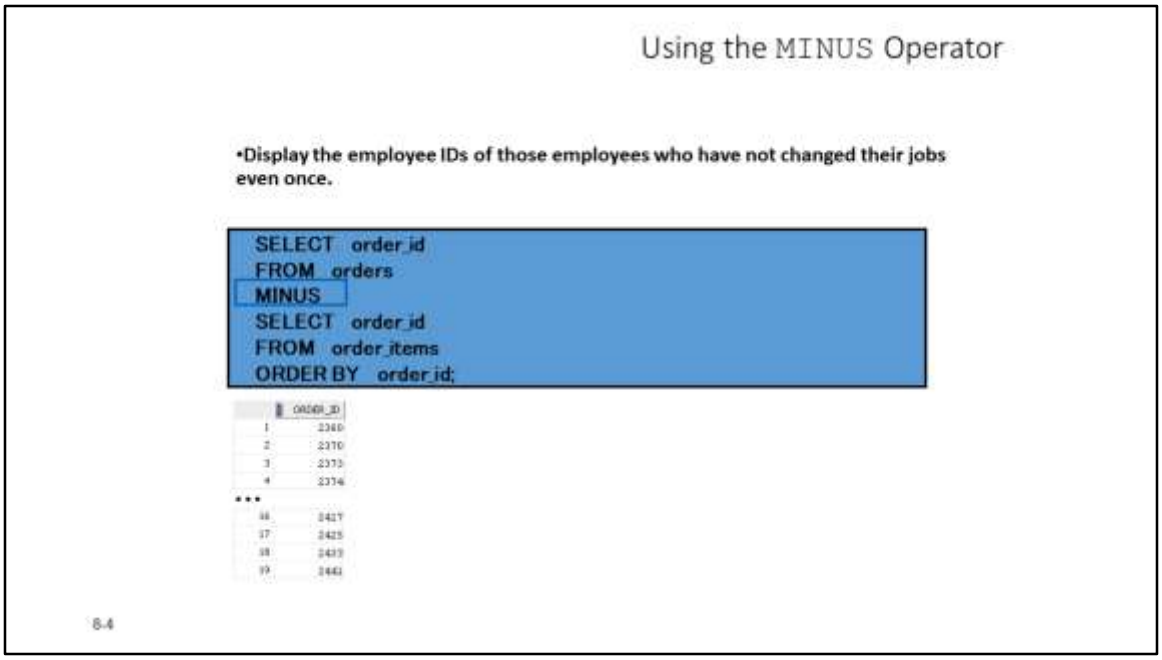
EMPLOYEES.DEPARTMENT_ID value is different from the JOB_HISTORY.DEPARTMENT_ID value.



MINUS Operator

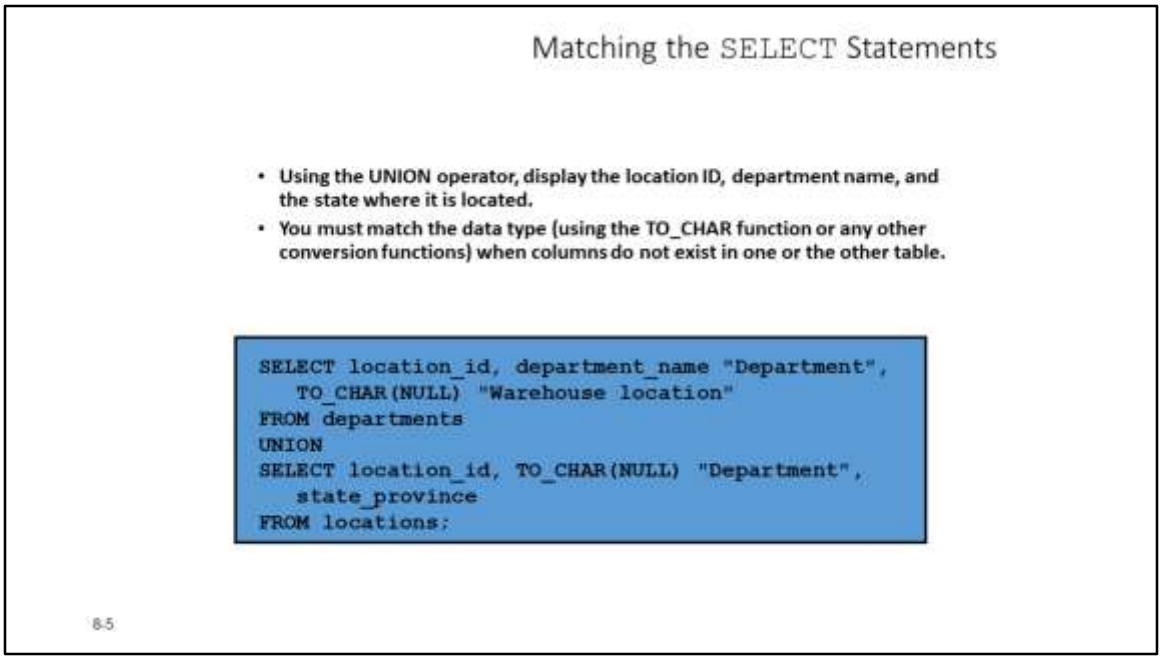
Use the MINUS operator to return all distinct rows selected by the first query, but not present in the second query result set (the first SELECT statement MINUS the second SELECT statement).

Note: The number of columns must be the same and the data types of the columns being selected by the SELECT statements in the queries must belong to the same data type group in all the SELECT statements used in the query. The names of the columns, however, need not be identical.



Using the `MINUS` Operator

In the example in the slide, the order_id's in the ORDER_ITEM table are subtracted from those in the ORDERS table. The results set displays the orders remaining after the subtraction; they are represented by rows that exist in the `ORDERS` table, but do not exist in the `ORDER_ITEMS` table.

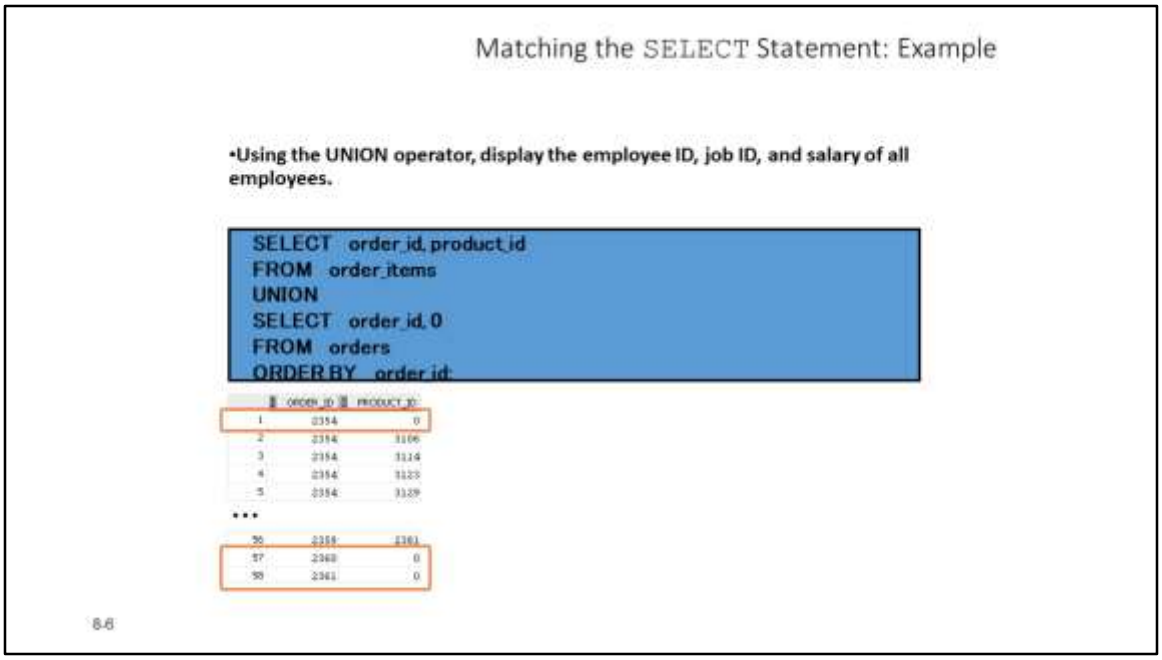


Matching the SELECT Statements

Because the expressions in the SELECT lists of the queries must match in number, you can use the dummy columns and the data type conversion functions to comply with this rule. In the slide, the name, Warehouse location, is given as the dummy column heading. The TO_CHAR function is used in the first query to match the VARCHAR2 data type of the state_province column that is retrieved by the second query. Similarly, the TO_CHAR function in the second query is used to match the VARCHAR2 data type of the department_name column that is retrieved by the first query.

The output of the query is shown:

	LOCATION_ID	Department	Warehouse location
1	1400	IT	(null)
2	1400	(null)	Texas
3	1500	Shipping	(null)
4	1500	(null)	California
5	1700	Accounting	(null)
6	1700	Administration	(null)
7	1700	Contracting	(null)
8	1700	Executive	(null)

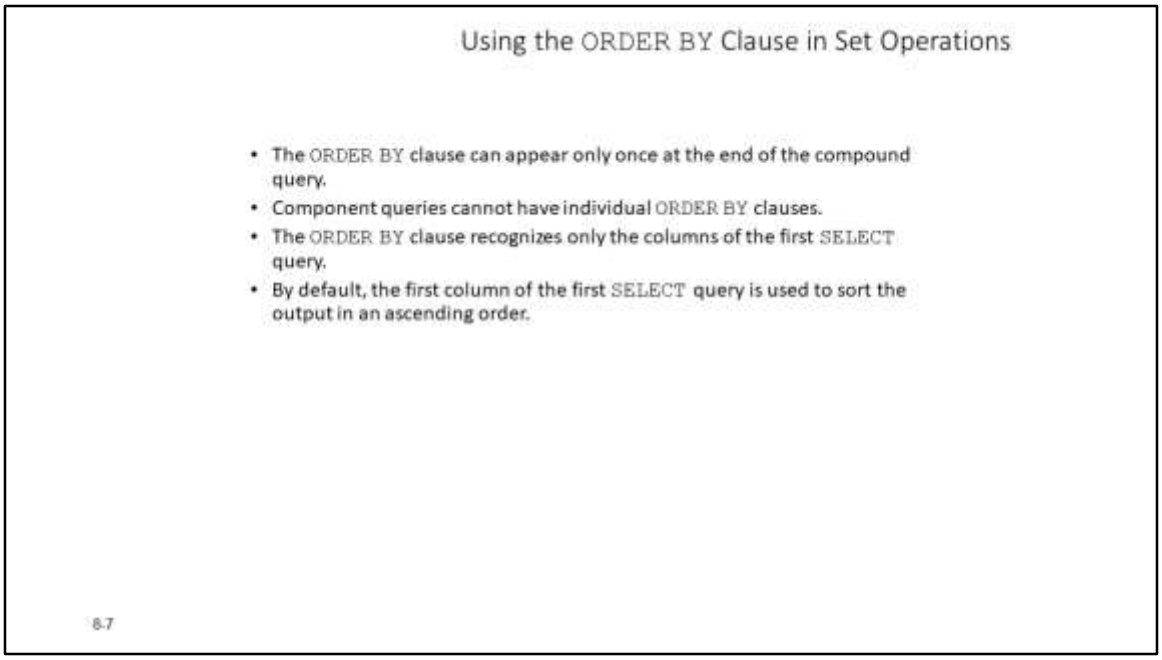


Matching the SELECT Statement: Example

The ORDERS and ORDER_ITEMS tables have one columns in common (for example, object_id)). But what if you want the query to display the order ID and the product ID, using the UNION operator, knowing that the product_id exists only in the OBJECT_ITEMS table?

The code example in the slide matches the OBJECT_ID and PRODUCT_ID columns in the ORDERS and ORDER_ITEMS tables. A literal value of 0 is added to the ORDERS table's SELECT statement to match the numeric PRODUCT_ID column in the ORDER_ITEM table's SELECT statement.

In the results shown in the slide, each row in the output that corresponds to a record from the ORDERS table contains a 0 in the PRODUCT_ID column.



Using the ORDER BY Clause in Set Operations

The ORDER BY clause can be used only once in a compound query. If used, the ORDER BY clause must be placed at the end of the query. The ORDER BY clause accepts the column name or an alias. By default, the output is sorted in ascending order in the first column of the first SELECT query.

Note: The ORDER BY clause does not recognize the column names of the second SELECT query. To avoid confusion over column names, it is a common practice to ORDER BY column positions.

For example, in the following statement, the output will be shown in ascending order of job_id.

```
SELECT order_id, product_id
FROM   order_items
UNION
SELECT order_id,0
FROM   orders
ORDER BY 2;
```

If you omit ORDER BY, by default, the output will be sorted in ascending order of object_id. You cannot use the columns from the second query to sort the output.

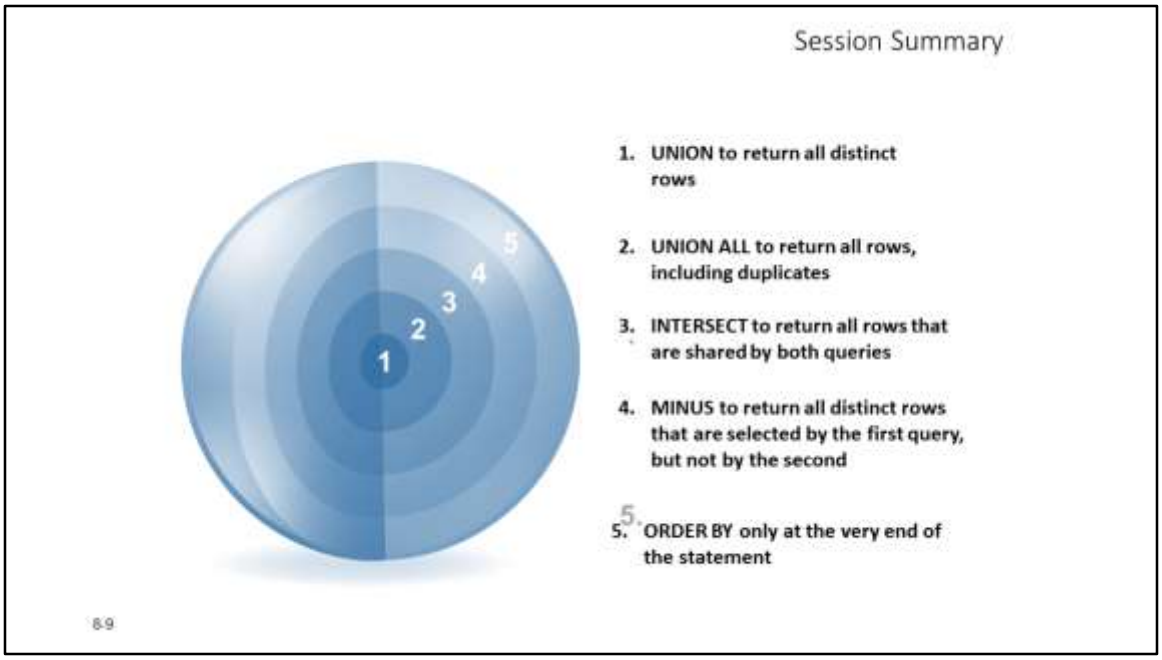
Quiz

- Identify the set operator guidelines.

1. The expressions in the SELECT lists must match in number.
2. Parentheses may not be used to alter the sequence of execution.
3. The data type of each column in the second query must match the data type of its corresponding column in the first query.
4. The ORDER BY clause can be used only once in a compound query, unless a UNION ALL operator is used.

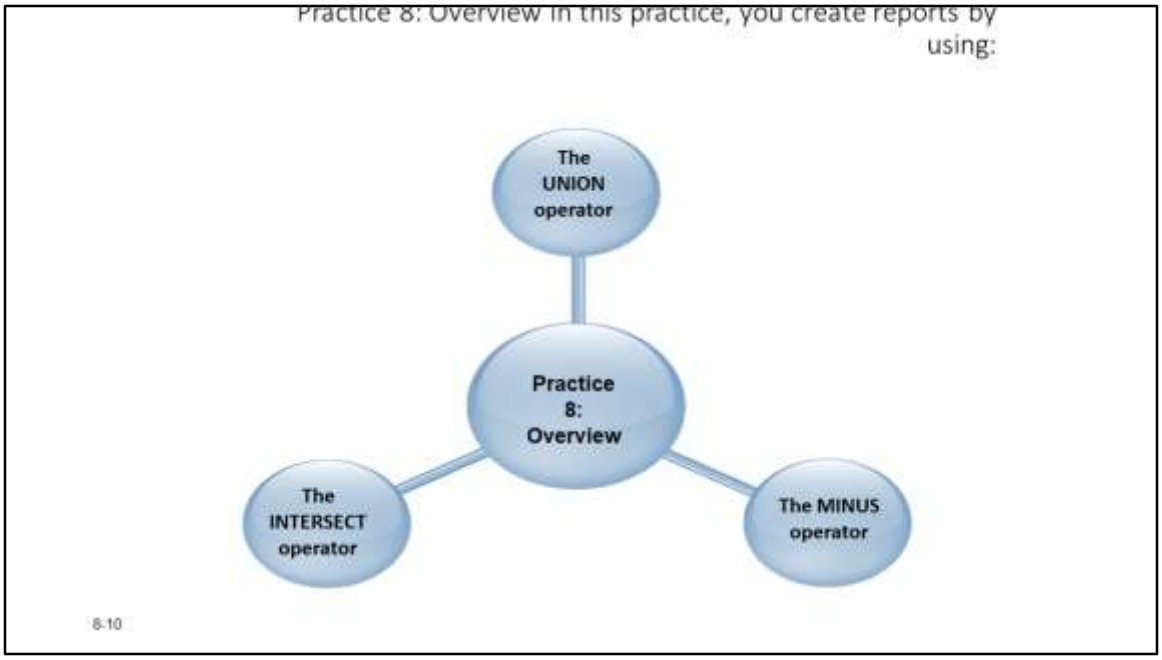
8-8

Answer: 1, 3



Session Summary

The `UNION` operator returns all the distinct rows selected by each query in the compound query. Use the `UNION` operator to return all rows from multiple tables and eliminate any duplicate rows.
Use the `UNION ALL` operator to return all rows from multiple queries. Unlike the case with the `UNION` operator, duplicate rows are not eliminated and the output is not sorted by default.
Use the `INTERSECT` operator to return all rows that are common to multiple queries.
Use the `MINUS` operator to return rows returned by the first query that are not present in the second query.
Remember to use the `ORDER BY` clause only at the very end of the compound statement.
Make sure that the corresponding expressions in the `SELECT` lists match in number and data type.



Practice 8: Overview

In this practice, you write queries using the set operators.