

Database Programming with SQL 9-3: Set Operators

Practice Activities

# Objectives

* Define and explain the purpose of SET operators
* Use a set operator to combine multiple queries into a single query
* Control the order of rows returned using set operators

# Vocabulary

Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **UNION** | operator that returns all rows from both tables and eliminates duplicates |
| **TO\_CHAR(NULL)** or **TO\_DATE(NULL)** or **TO\_NUMBER(NULL)** | columns that were made up to match queries in another table that are not in both tables |
| **UNION ALL** | operator that returns all rows from both tables, including duplicates |
| **set operators** | used to combine results into one single result from multiple SELECT statements |
| **MINUS** | operator that returns rows that are unique to each table |
| **INTERSECT** | operator that returns rows common to both tables |

# Try It / Solve It

1. Name the different Set operators?

**UNION**

**UNION ALL**

**MINUS**

**INTERSECT**

1. Write one query to return the employee\_id, job\_id, hire\_date, and department\_id of all employees and a second query listing employee\_id, job\_id, start\_date, and department\_id from the job\_history table and combine the results as one single output. Make sure you suppress duplicates in the output.

**SELECT employee\_id, job\_id, hire\_date, department\_id**

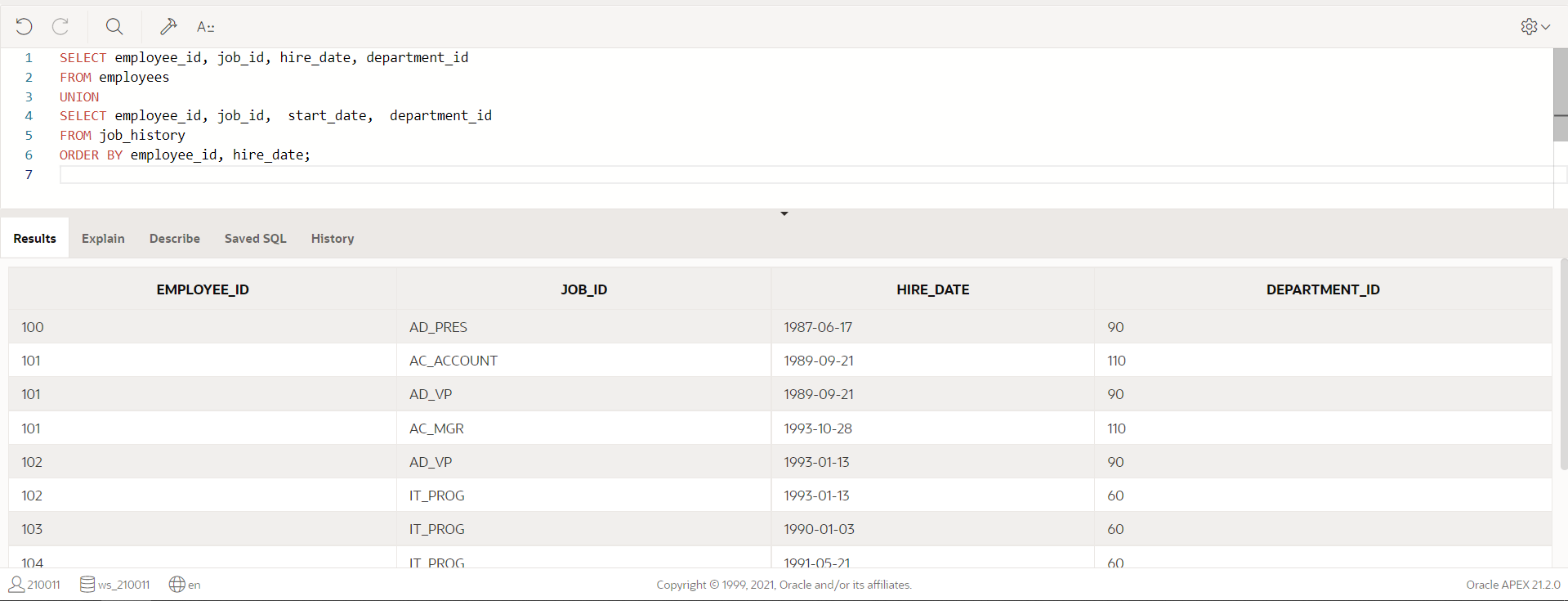
**FROM employees**

**UNION**

**SELECT employee\_id, job\_id,  start\_date,  department\_id**

**FROM job\_history**

**ORDER BY employee\_id, hire\_date;**



1. Amend the previous statement to not suppress duplicates and examine the output. How many extra rows did you get returned and which were they? Sort the output by employee\_id to make it easier to spot.

**SELECT employee\_id, job\_id, hire\_date, department\_id**

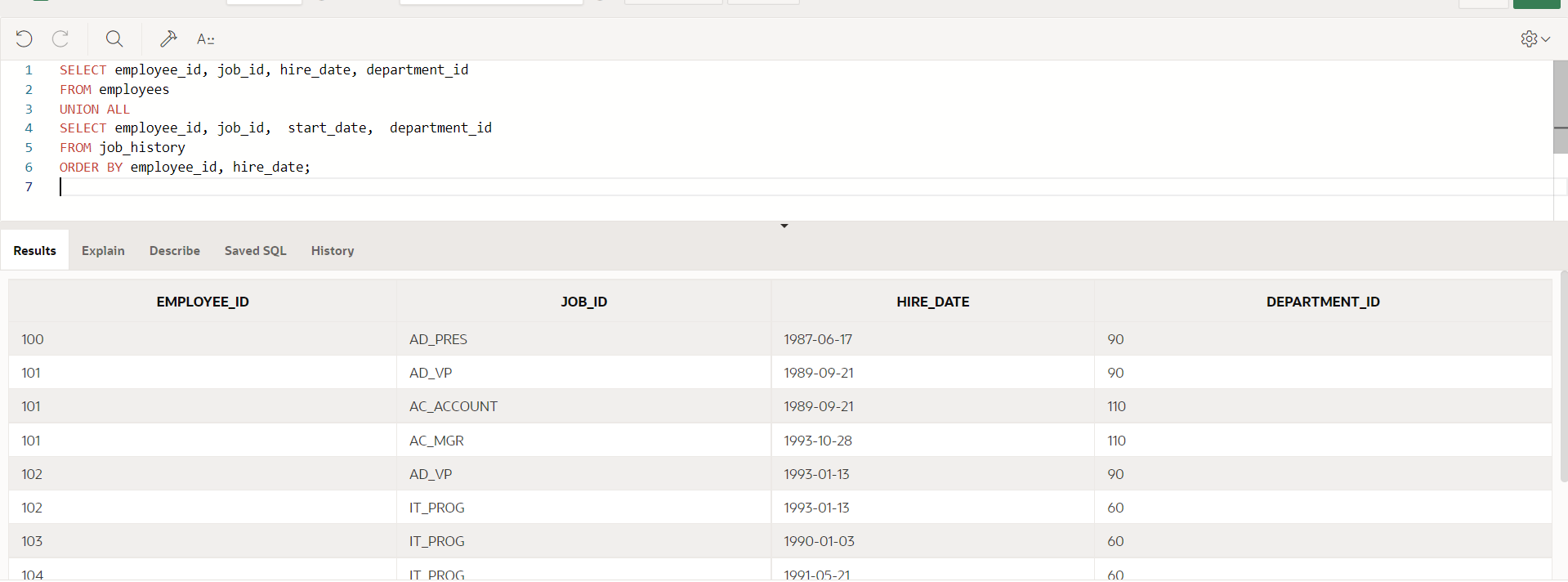
**FROM employees**

**UNION ALL**

**SELECT employee\_id, job\_id,  start\_date,  department\_id**

**FROM job\_history**

**ORDER BY employee\_id, hire\_date;**



1. List all employees who have not changed jobs even once. (Such employees are not found in the job\_history table)

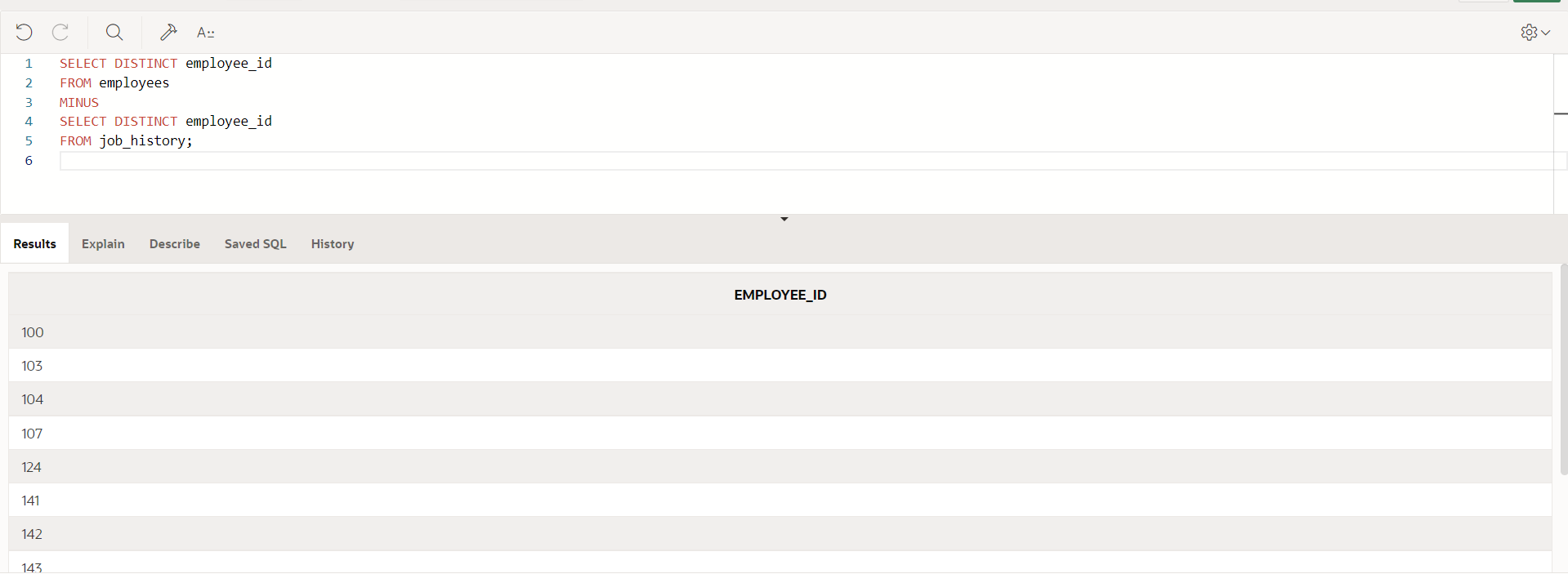
**SELECT DISTINCT employee\_id**

**FROM employees**

**MINUS**

**SELECT DISTINCT employee\_id**

**FROM job\_history;**



1. List the employees that HAVE changed their jobs at least once.

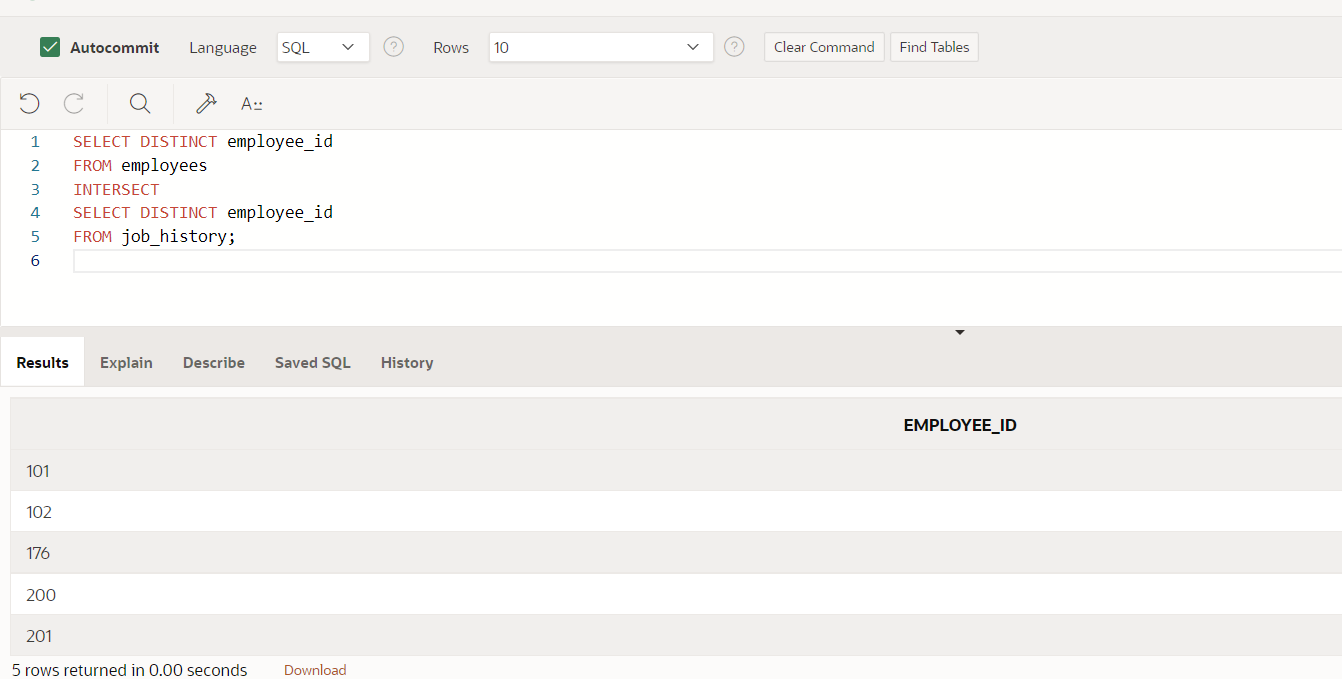
**SELECT DISTINCT employee\_id**

**FROM employees**

**INTERSECT**

**SELECT DISTINCT employee\_id**

**FROM job\_history;**



1. Using the UNION operator, write a query that displays the employee\_id, job\_id, and salary of ALL present and past employees. If a salary is not found, then just display a 0 (zero) in its place.

**SELECT employee\_id, job\_id, NVL(salary, 0)**

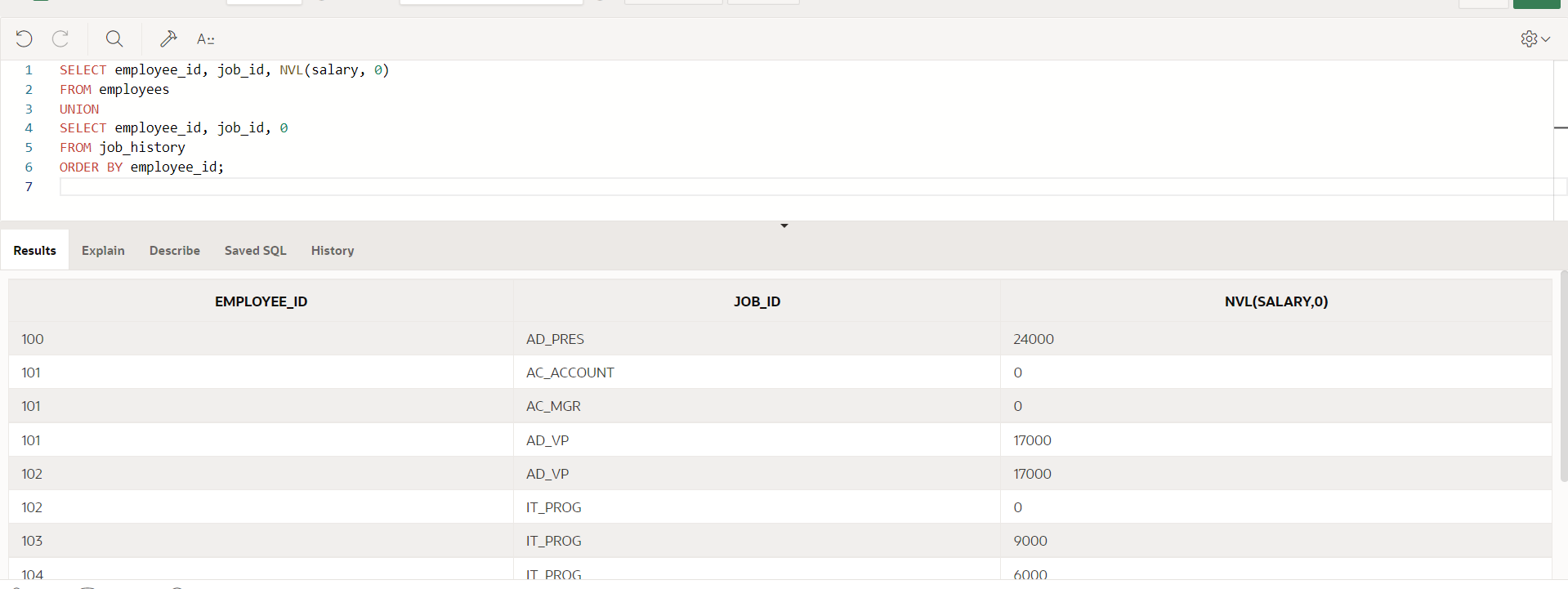
**FROM employees**

**UNION**

**SELECT employee\_id, job\_id, 0**

**FROM job\_history**

**ORDER BY employee\_id;**



Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.