

# ITC 5104 DATABASE Design and SQL

Lecture 2

Chapter 8 Oracle 12c SQL

Restricting Rows and Sorting Data

# Objectives

- Use a WHERE clause to restrict rows returned by a query
- Create a search condition using mathematical comparison operators
- Use the BETWEEN ... AND comparison operator to specify records within a range of values
- Specify a list of values for a search condition using the IN comparison operator
- Search for patterns using the LIKE comparison operator
- Identify the purpose of the % and \_ wildcard operators

# Objectives

- Join multiple search conditions using the appropriate logical operator
- Perform searches for NULL values
- Specify the order for the presentation of query results using ORDER BY, DESC, ASC and the SELECT clause

# Introduction

- In chapter 2 we learned how to retrieve specific fields from a table
- Unless the **UNIQUE** or **DISTINCT** keywords were used, the results returned included every row from the table
- In some cases, you may only want to see records that meet a certain condition or conditions. This is a process called **selection**
- Selection reduces the number of records retrieved by a query
- We will use the **WHERE** clause with the **SELECT** statement and the **ORDER BY** clause to present selected results in a specific sequence

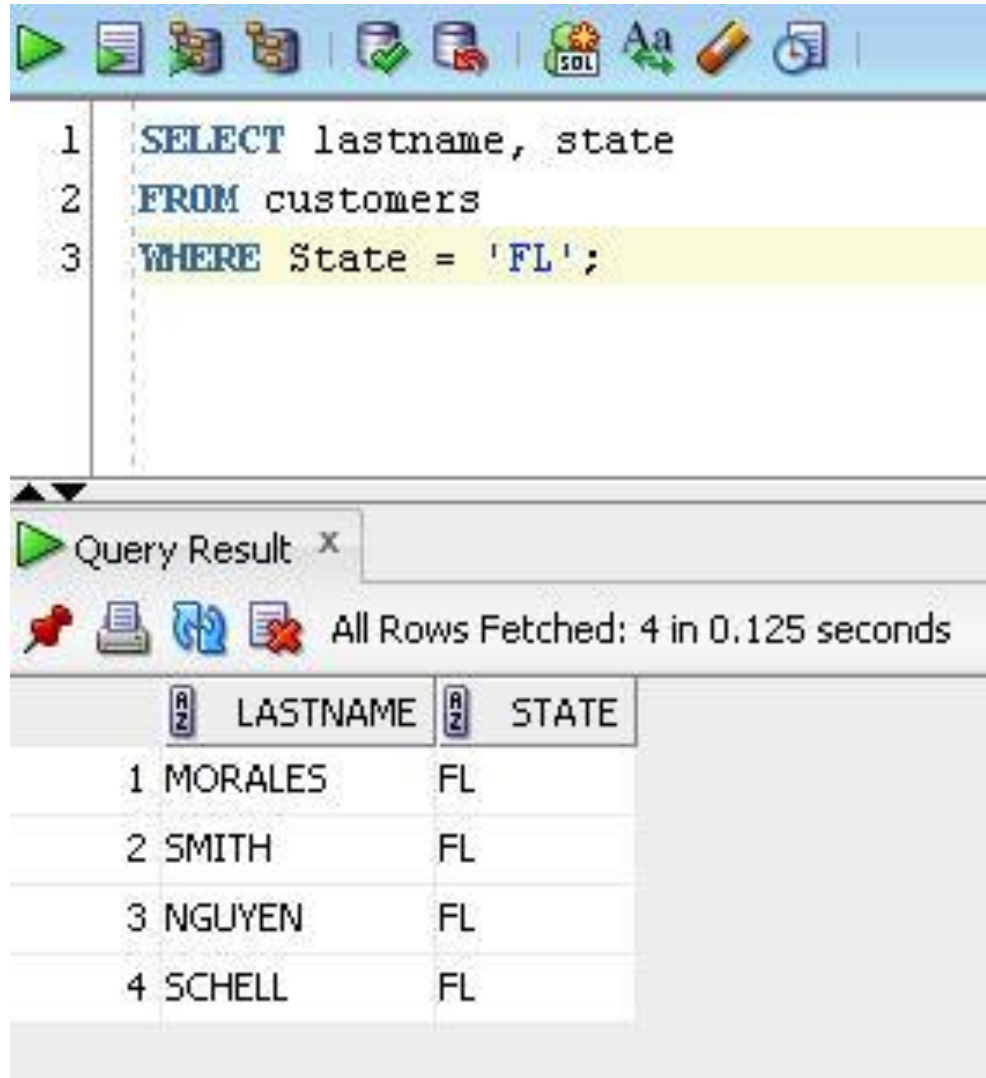
# Syntax of Select Statement

```
SELECT [DISTINCT | UNIQUE] (*, columnname [ AS alias], ...)  
      FROM      tablename  
      [WHERE    condition]  
      [GROUP BY group_by_expression]  
      [HAVING   group_condition]  
      [ORDER BY columnname];
```

# WHERE Clause

- The **WHERE** clause is optional
- Listed beneath the **FROM** clause
- A condition identifies what must exist or a requirement that must be met
- Oracle 12c searches through each record to determine whether the condition is **TRUE**
- If the record meets the condition it will be returned in the results of the query
- **WHERE** clause uses the following format:
  - **<column name> <comparison operator> <another column name or a physical value>**

# WHERE Clause



The screenshot shows a SQL query editor window with a toolbar at the top. The query text is as follows:

```
1 SELECT lastname, state
2 FROM customers
3 WHERE State = 'FL';
```

The third line of the query is highlighted in yellow. Below the editor is a "Query Result" window. It displays the status "All Rows Fetched: 4 in 0.125 seconds". The results are shown in a table with two columns: "LASTNAME" and "STATE".

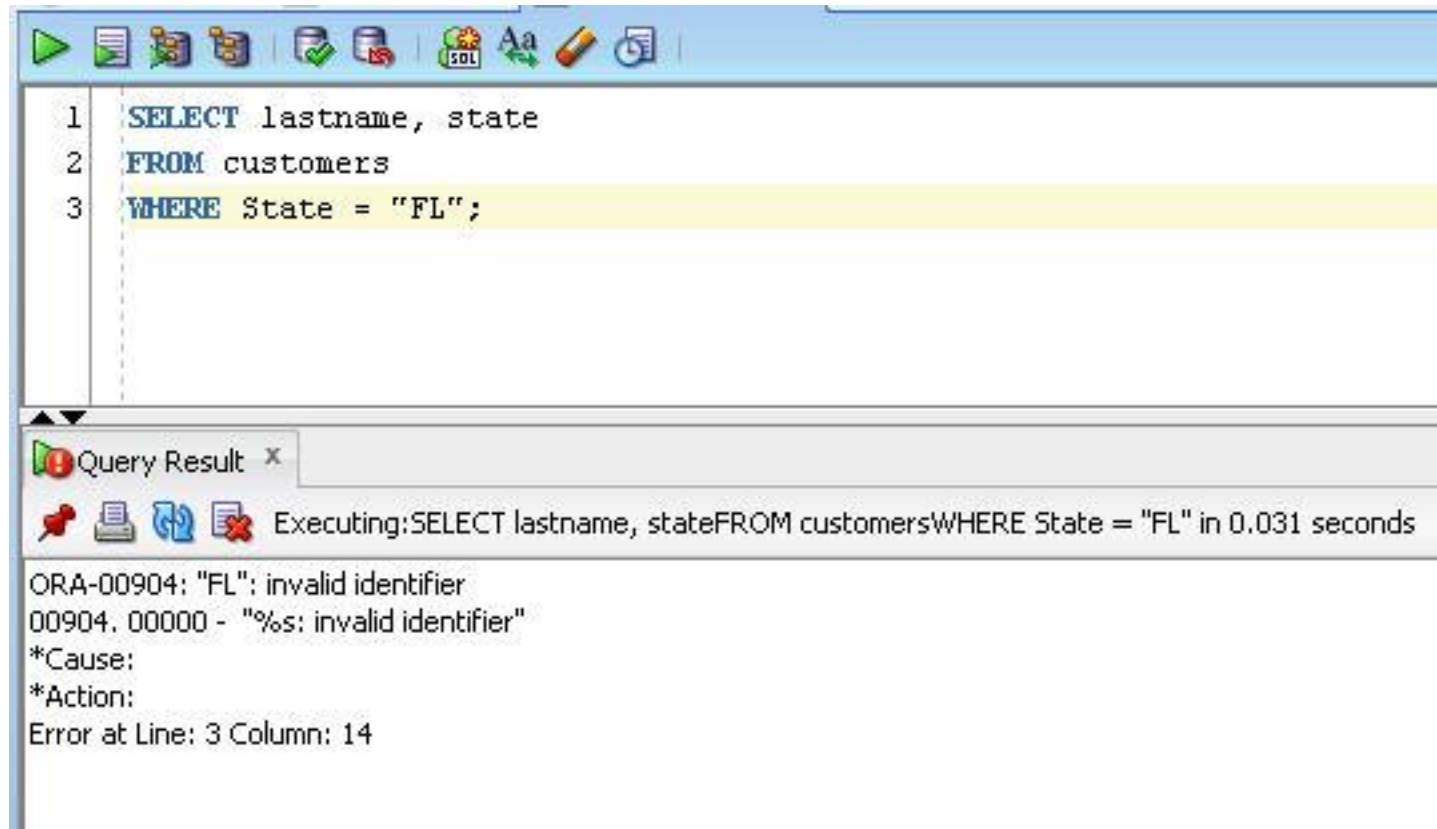
	LASTNAME	STATE
1	MORALES	FL
2	SMITH	FL
3	NGUYEN	FL
4	SCHELL	FL

# WHERE Clause

- This query specifies the Lastname and State columns stored in the Customers table as the data to be listed in the output
- However, you only want to see the records of the customers who have the the letters FL stored in the State column
  - **WHERE** is the keyword
  - **State** is the name of the column being searched
  - **=** is the comparison operator
  - **FL** is the specific value being searched
- All character data for the Just Lee Books database is in **upper-case** because the original data was entered all in capitals

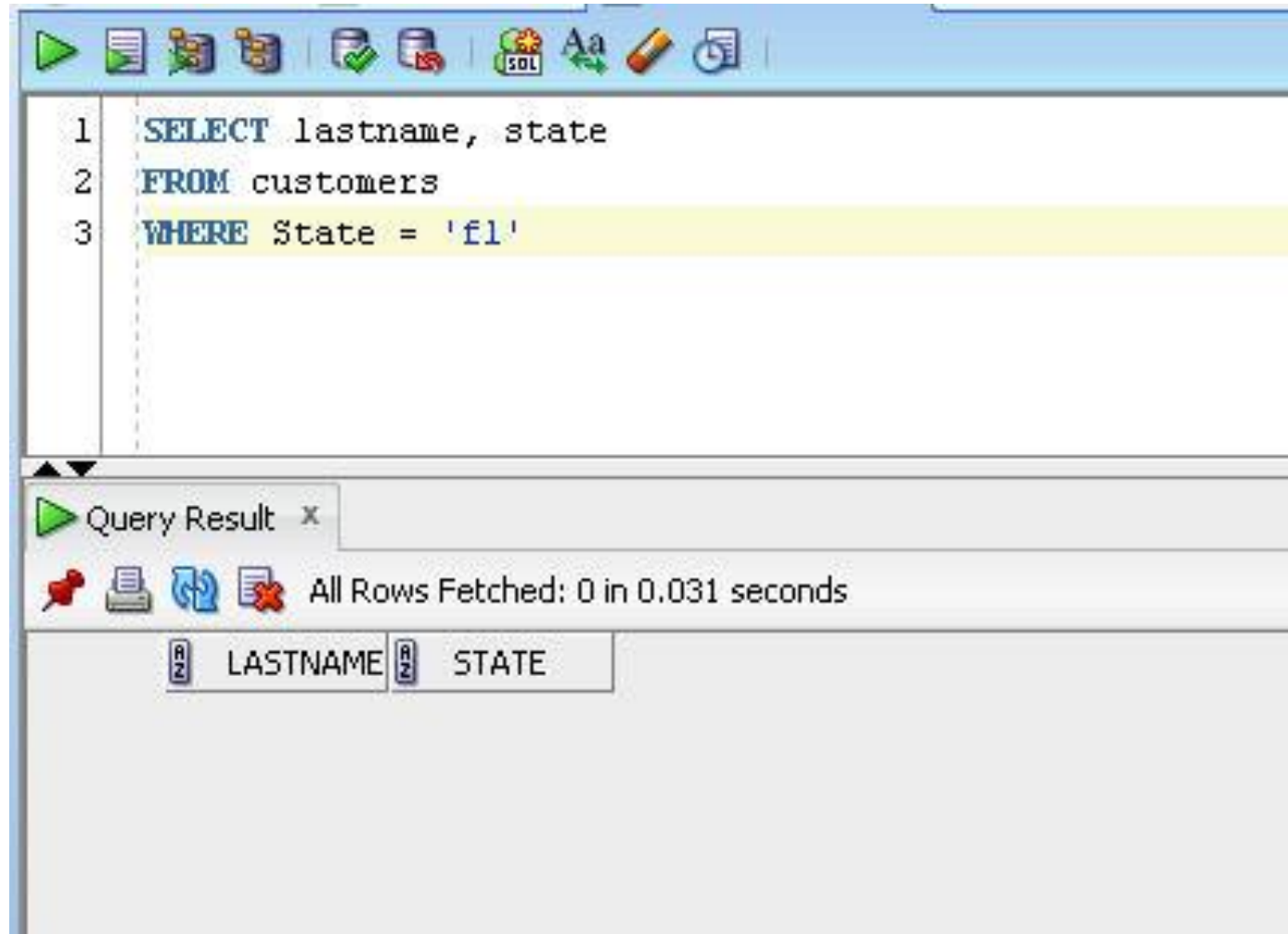


# Possible Errors

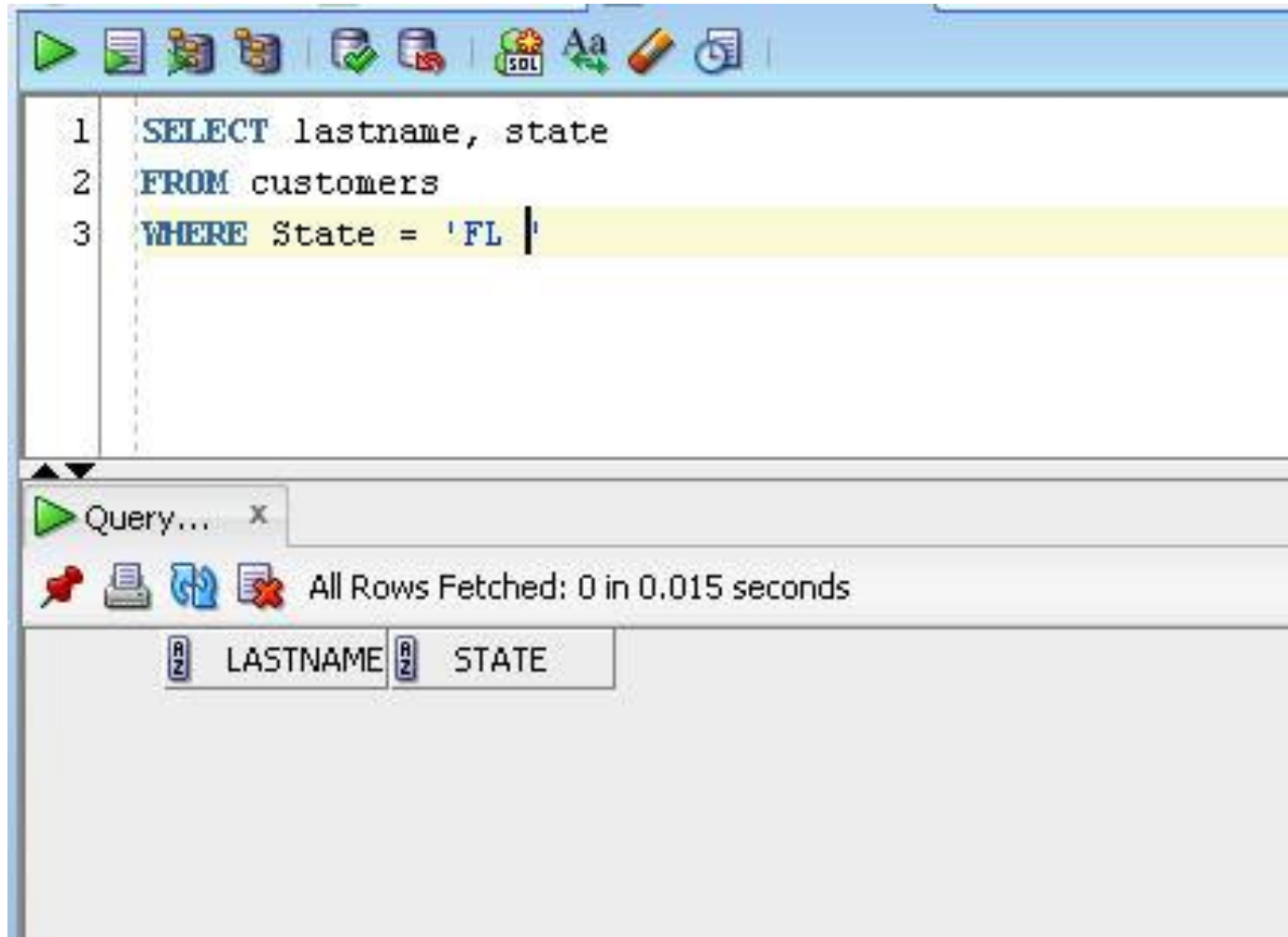


Can you figure out why the commands on the next three slides failed?

# Possible Errors



# Possible Errors



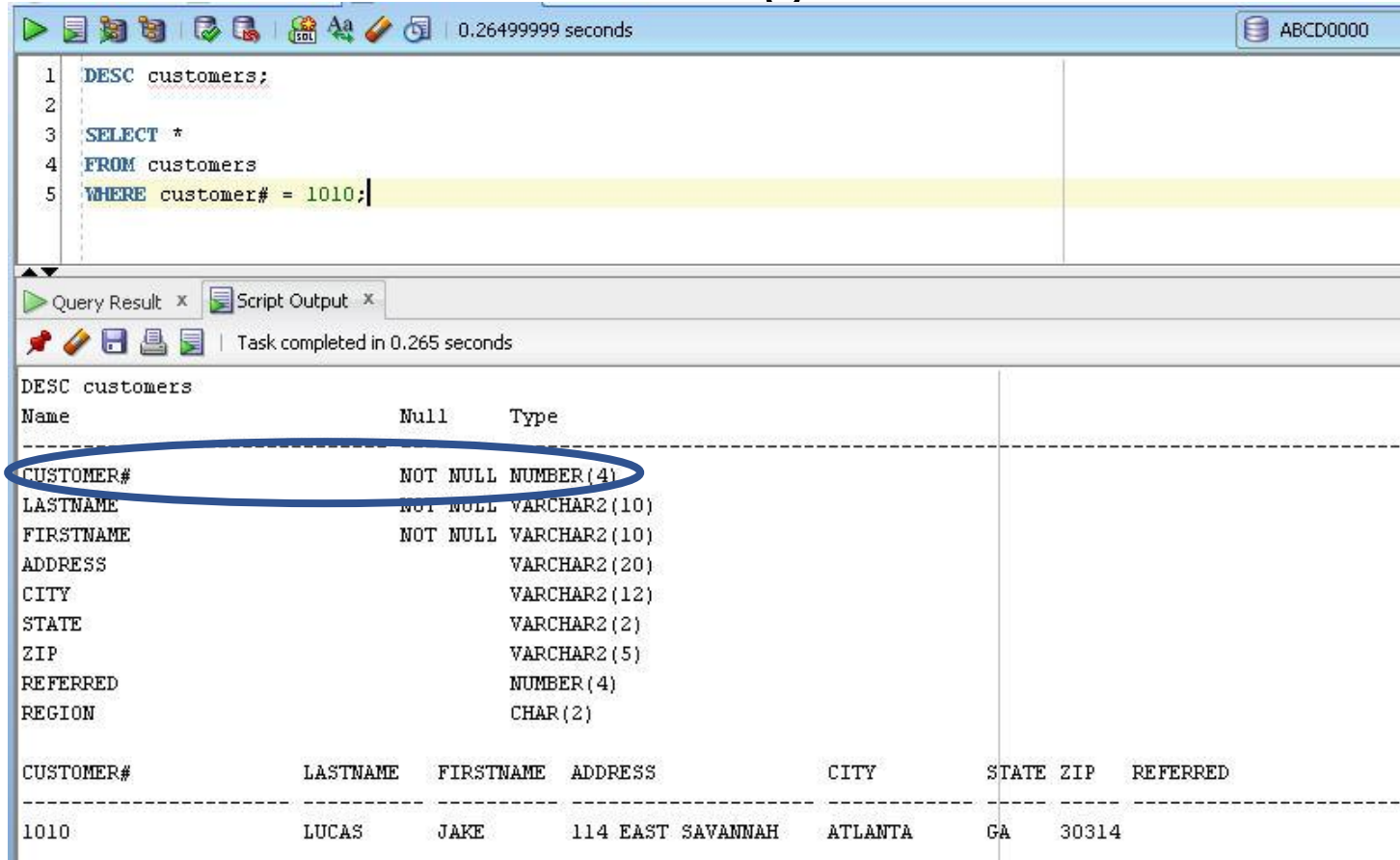
# Errors

- In the first example, the value of FL was entered in **double quotation marks**, these are reserved for **column alias's**
- In the second example, the letters FL are entered in lowercase, but the data is stored in uppercase in the table. A string entered in the search condition must be in the same case as it is stored. Although the Oracle 12c syntax is not case sensitive, the data stored in the record is **case sensitive**
- In the third example, although the search condition is in uppercase, a blank space was added so it does **not exactly match** because of the extra space in the search condition

# Rules for Character Strings

- As shown in the previous SQL command, the value of FL is shown in **single quotation marks**. Whenever a string literal is used as part of a search condition, it must be enclosed in single quotation marks
- It will be interpreted exactly as listed inside the **single quotation marks**
- If the target value contains only numbers, the single quotation marks are not required
- If you entered the previous query with no single quotation marks you would get an error

# Rules for Character Strings



The screenshot shows an SQL IDE interface. The top toolbar includes icons for running queries, saving, and other standard database operations. The status bar at the top right shows 'ABCD0000' and a timer at '0.26499999 seconds'. The main editor contains the following SQL code:

```
1 DESC customers;
2
3 SELECT *
4 FROM customers
5 WHERE customer# = 1010;
```

Below the editor, the 'Query Result' tab is active, showing the results of the 'DESC customers' query. The results are displayed in a table with three columns: 'Name', 'Null', and 'Type'. The 'CUSTOMER#' row is circled in blue. Below this, the results of the 'SELECT \* FROM customers WHERE customer# = 1010;' query are shown in a table with columns: 'CUSTOMER#', 'LASTNAME', 'FIRSTNAME', 'ADDRESS', 'CITY', 'STATE', 'ZIP', and 'REFERRED'.

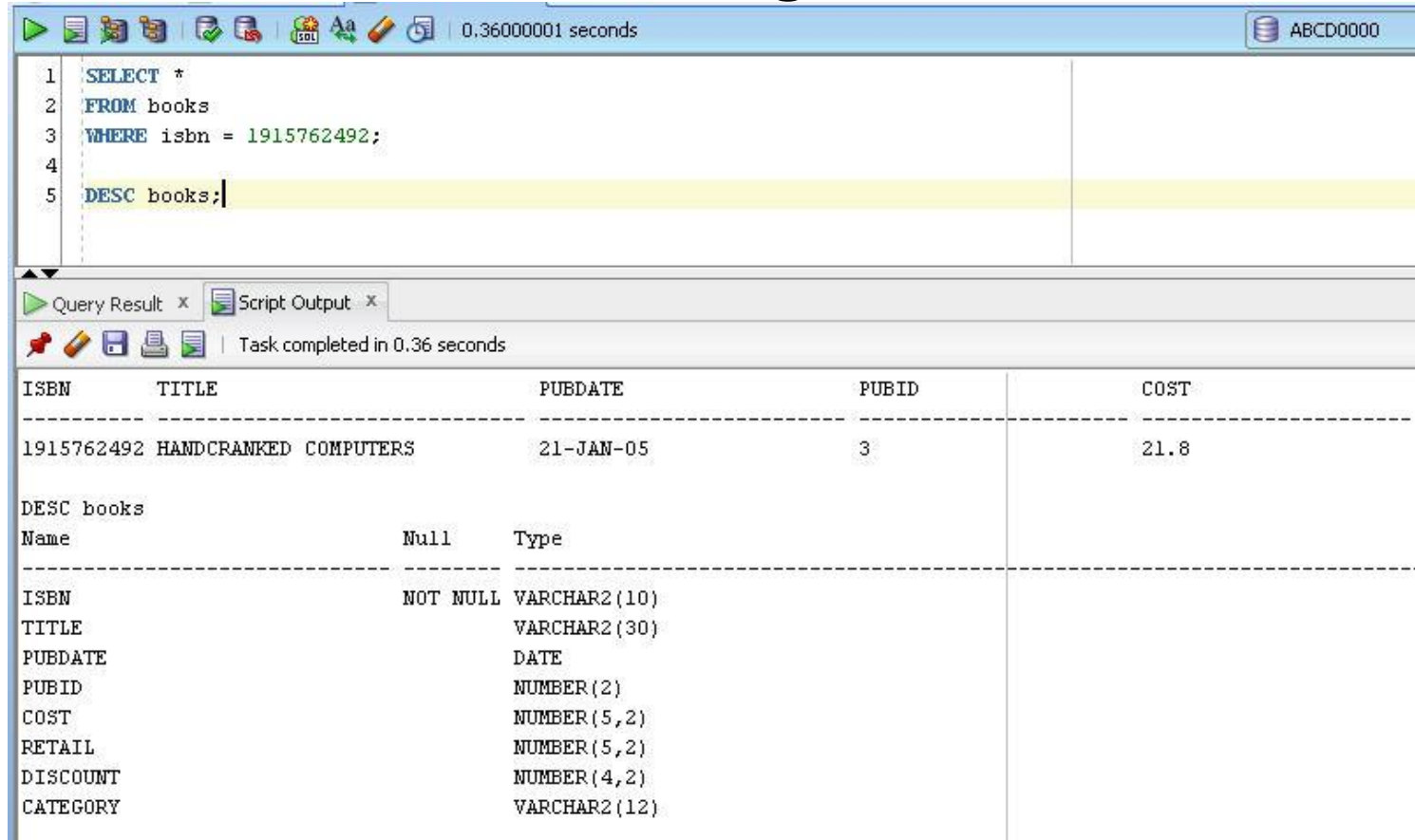
NAME	NULL	TYPE
CUSTOMER#	NOT NULL	NUMBER(4)
LASTNAME	NOT NULL	VARCHAR2(10)
FIRSTNAME	NOT NULL	VARCHAR2(10)
ADDRESS		VARCHAR2(20)
CITY		VARCHAR2(12)
STATE		VARCHAR2(2)
ZIP		VARCHAR2(5)
REFERRED		NUMBER(4)
REGION		CHAR(2)

CUSTOMER#	LASTNAME	FIRSTNAME	ADDRESS	CITY	STATE	ZIP	REFERRED
1010	LUCAS	JAKE	114 EAST SAVANNAH	ATLANTA	GA	30314	

The value 1010 for the Customer# column is not enclosed in single quotes because it is defined as a numeric column

# Rules for Character Strings



0.36000001 seconds

ABCD0000

```
1 SELECT *
2 FROM books
3 WHERE isbn = 1915762492;
4
5 DESC books;
```

Query Result x Script Output x

Task completed in 0.36 seconds

ISBN	TITLE	PUBDATE	PUBID	COST
1915762492	HANDCRANKED COMPUTERS	21-JAN-05	3	21.8

DESC books

Name	Null	Type
ISBN	NOT NULL	VARCHAR2(10)
TITLE		VARCHAR2(30)
PUBDATE		DATE
PUBID		NUMBER(2)
COST		NUMBER(5,2)
RETAIL		NUMBER(5,2)
DISCOUNT		NUMBER(4,2)
CATEGORY		VARCHAR2(12)

Look at the output of this query. In addition to the description of the books table, do you notice anything peculiar?

# Rules for Character Strings

- The ISBN of the BOOKS table is defined as a character (VARCHAR2) or text field rather than a numeric field, since some ISBN values could contain letters
- In this particular instance, none of the values stored in the ISBN contain any letters
- This means you can search the field using a search condition specified as numeric (without any quotation marks)
- However, if one of the records being searched contained a letter in the ISBN field, an error would have been returned by Oracle 12c



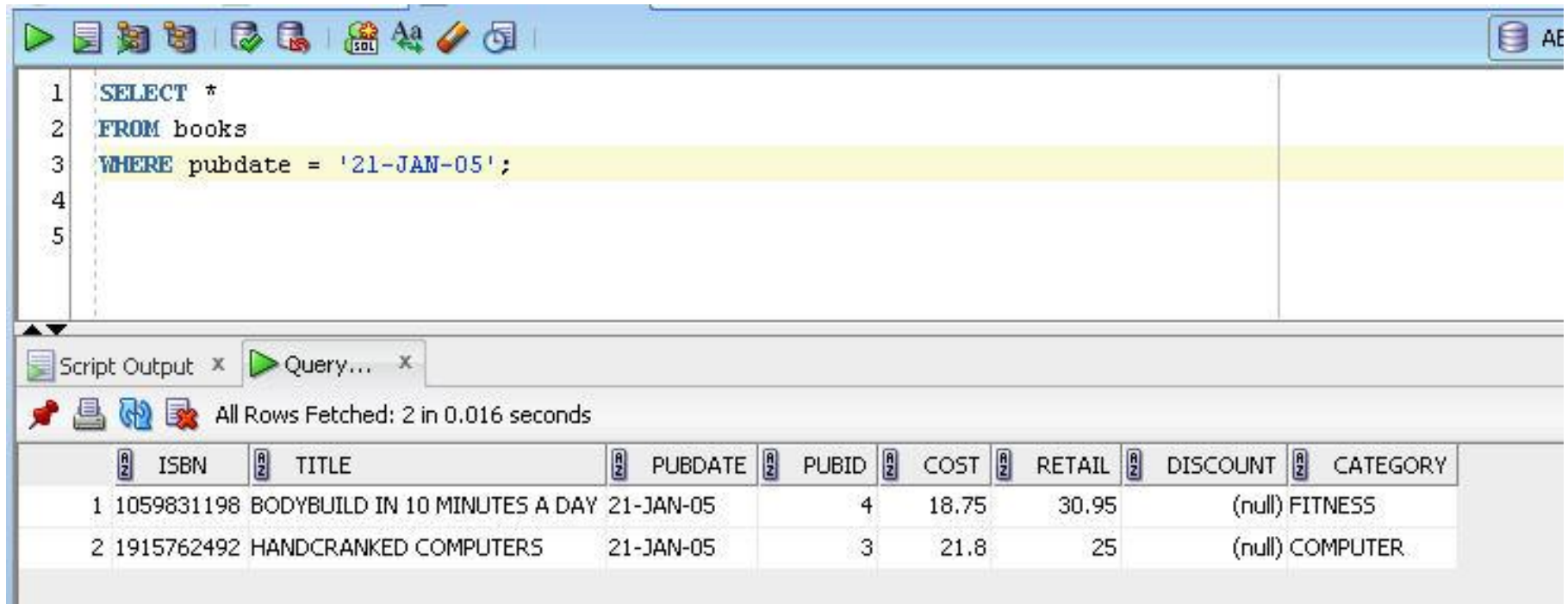
# Rules for Character Strings

- In other words, it did work in this case, but it might not always work
- Using single quotation marks ultimately depends on whether the field is defined to hold text or only numeric data
- Therefore, always use single quotation marks if the column is defined as anything other than a numeric field
- Remember, if you don't know if a column is defined to hold numeric values, use the **DESCRIBE table\_name** command to see the table columns definitions

# Rules for Dates

- Sometimes, you might need to use a date as a search condition
- Oracle 12c displays dates in the default format of DD-MON-RR, with MON being the standard three letter abbreviation for the month, you will see it in our text book as DD-MON-YY, RR is the Y2K compliant value for the date and Oracle considers it to be the default
- In the BOOKS table, the PUBLISHDATE field contains letters and hyphens
- Dates are not considered numeric values when Oracle 12c performs searches
- This means date fields must be enclosed in single quotation marks

# Rules for Dates



The screenshot shows a database query tool interface. The top toolbar contains icons for execution, saving, and editing. The main text area displays a SQL query:

```
1 SELECT *
2 FROM books
3 WHERE pubdate = '21-JAN-05';
4
5
```

Below the query editor, the 'Script Output' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 2 in 0.016 seconds'. The results are displayed in a table with the following columns: ISBN, TITLE, PUBDATE, PUBID, COST, RETAIL, DISCOUNT, and CATEGORY.

	ISBN	TITLE	PUBDATE	PUBID	COST	RETAIL	DISCOUNT	CATEGORY
1	1059831198	BODYBUILD IN 10 MINUTES A DAY	21-JAN-05	4	18.75	30.95	(null)	FITNESS
2	1915762492	HANDCRANKED COMPUTERS	21-JAN-05	3	21.8	25	(null)	COMPUTER

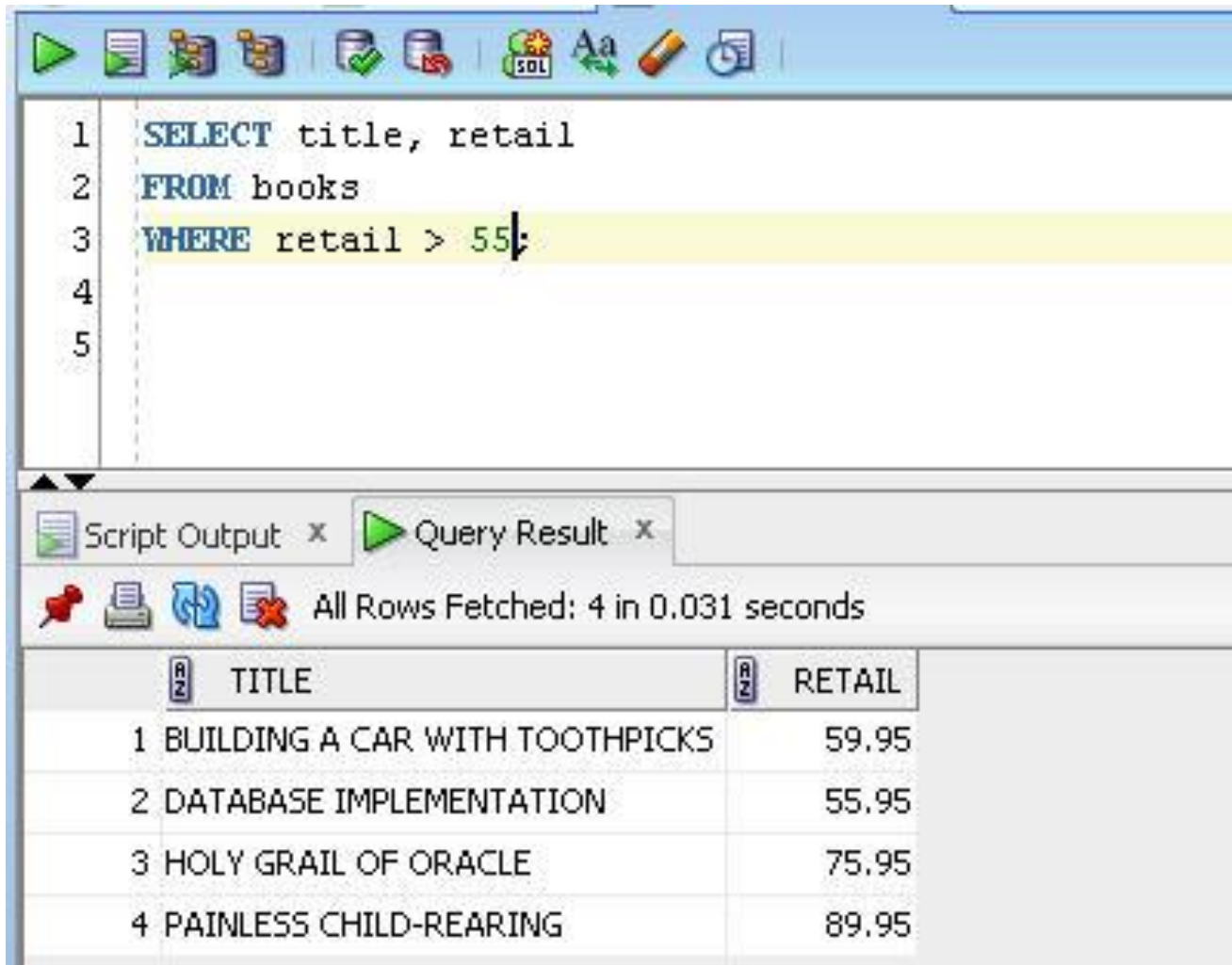
# Comparison Operators

Operator	Description
=	Equality, equal to
>	Greater than
<	Less than
<>, !=, ^=	Not equal to, syntax can be entered three different ways
<=	Less than or equal to
>=	Greater than or equal to

# Comparison Operators

- So far, we have used the equal sign (the equality operator) to evaluate search conditions
- Oracle 12c has returned results containing the exact value that was provided
- There are many situations that are not based on an “equal to” condition
- A **comparison operator** indicates how the data should relate to a given search value
- When working with dollar amounts, remember that Oracle 12c does not have a currency data type. Therefore, you cannot use search conditions with \$ or , in the search condition

# Comparison Operators



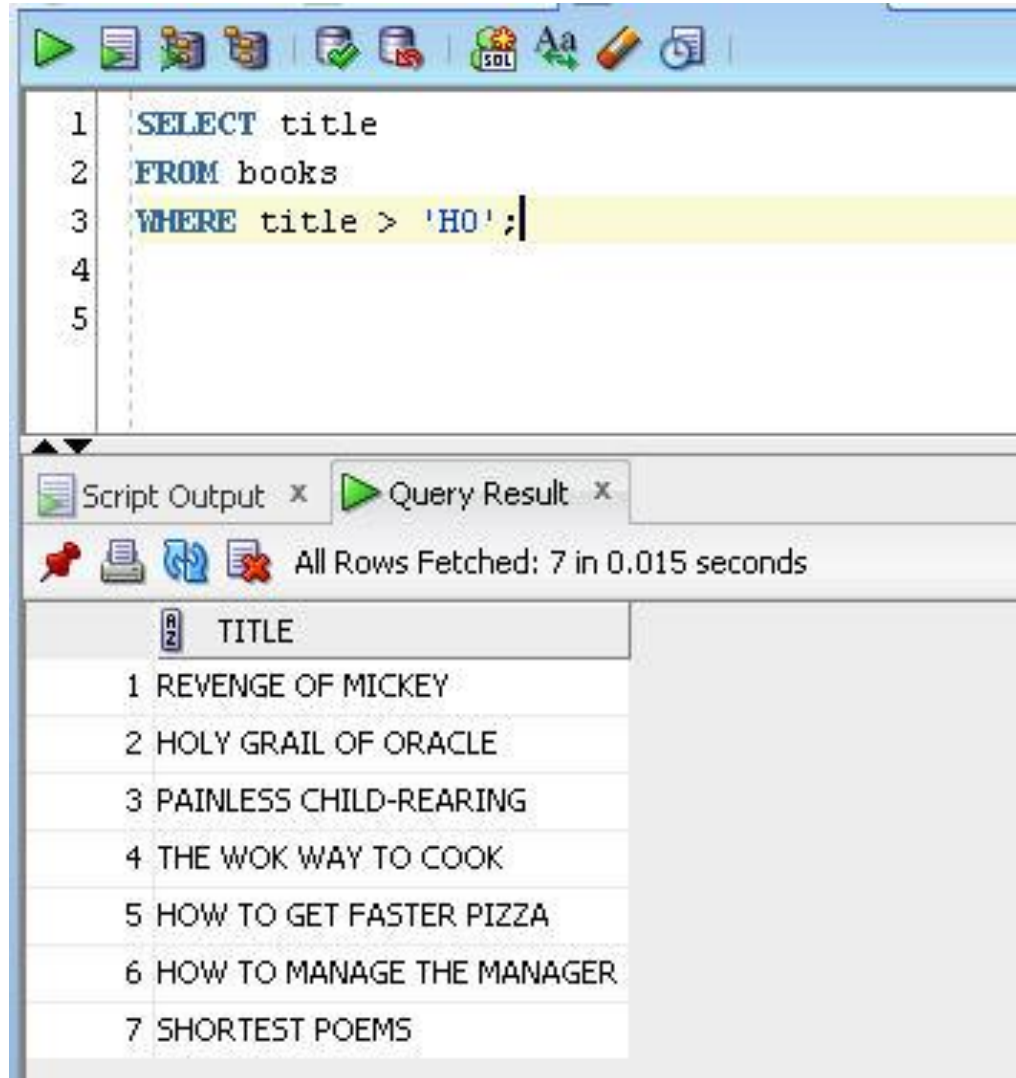
The screenshot shows a SQL IDE interface. The top toolbar contains icons for running queries, saving, and other database functions. The main text area displays the following SQL query:

```
1 SELECT title, retail
2 FROM books
3 WHERE retail > 55;
```

Below the query editor, the 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 4 in 0.031 seconds'. The results are displayed in a table with two columns: 'TITLE' and 'RETAIL'.

	TITLE	RETAIL
1	BUILDING A CAR WITH TOOTHPICKS	59.95
2	DATABASE IMPLEMENTATION	55.95
3	HOLY GRAIL OF ORACLE	75.95
4	PAINLESS CHILD-REARING	89.95

# Comparison Operators



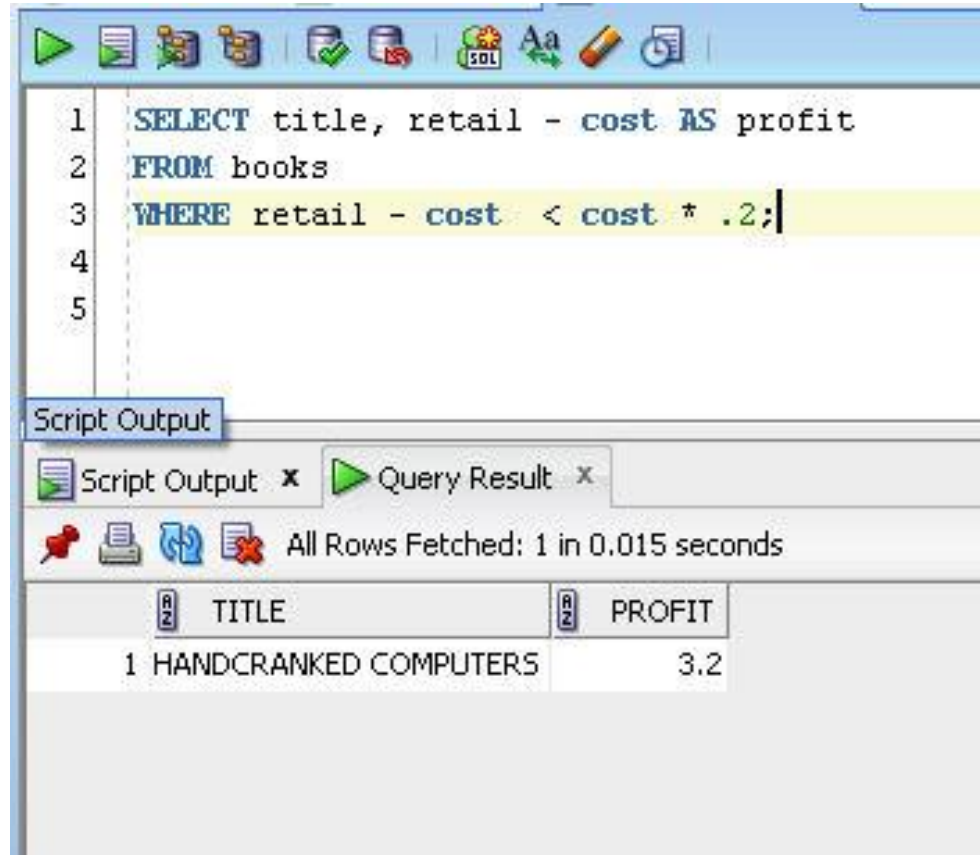
The screenshot shows a SQL IDE interface. The top toolbar contains icons for running queries, saving, and editing. The main text area displays the following SQL query:

```
1 SELECT title
2 FROM books
3 WHERE title > 'HO';
4
5
```

Below the query editor, the 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 7 in 0.015 seconds'. The results are displayed in a table with one column, 'TITLE'.

TITLE
1 REVENGE OF MICKEY
2 HOLY GRAIL OF ORACLE
3 PAINLESS CHILD-REARING
4 THE WOK WAY TO COOK
5 HOW TO GET FASTER PIZZA
6 HOW TO MANAGE THE MANAGER
7 SHORTEST POEMS

# Comparison Operators



The screenshot shows a SQL query editor with the following query:

```
1 SELECT title, retail - cost AS profit
2 FROM books
3 WHERE retail - cost < cost * .2;
4
5
```

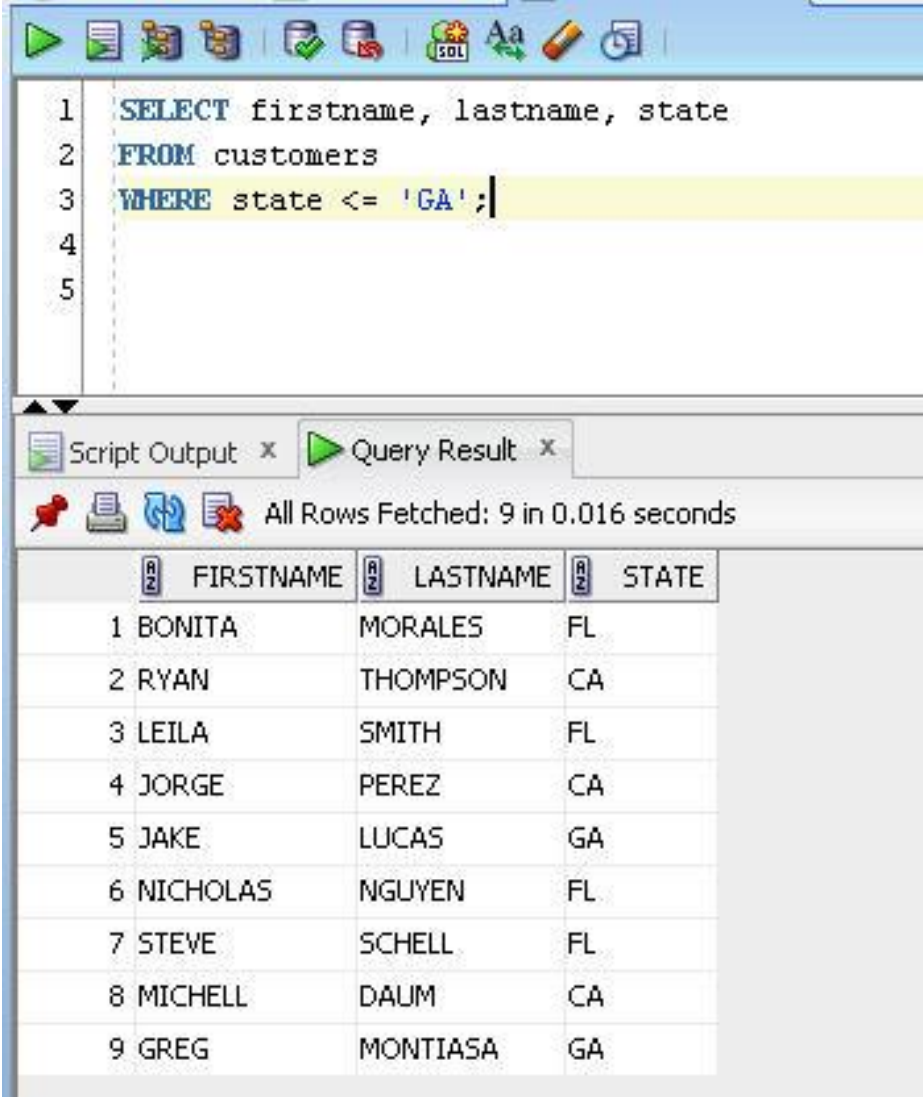
The query is executed, and the results are displayed in a table. The table has two columns: TITLE and PROFIT. The results are as follows:

	TITLE	PROFIT
1	HANDCRANKED COMPUTERS	3.2

Comparison operator used between two expressions



# Comparison Operators



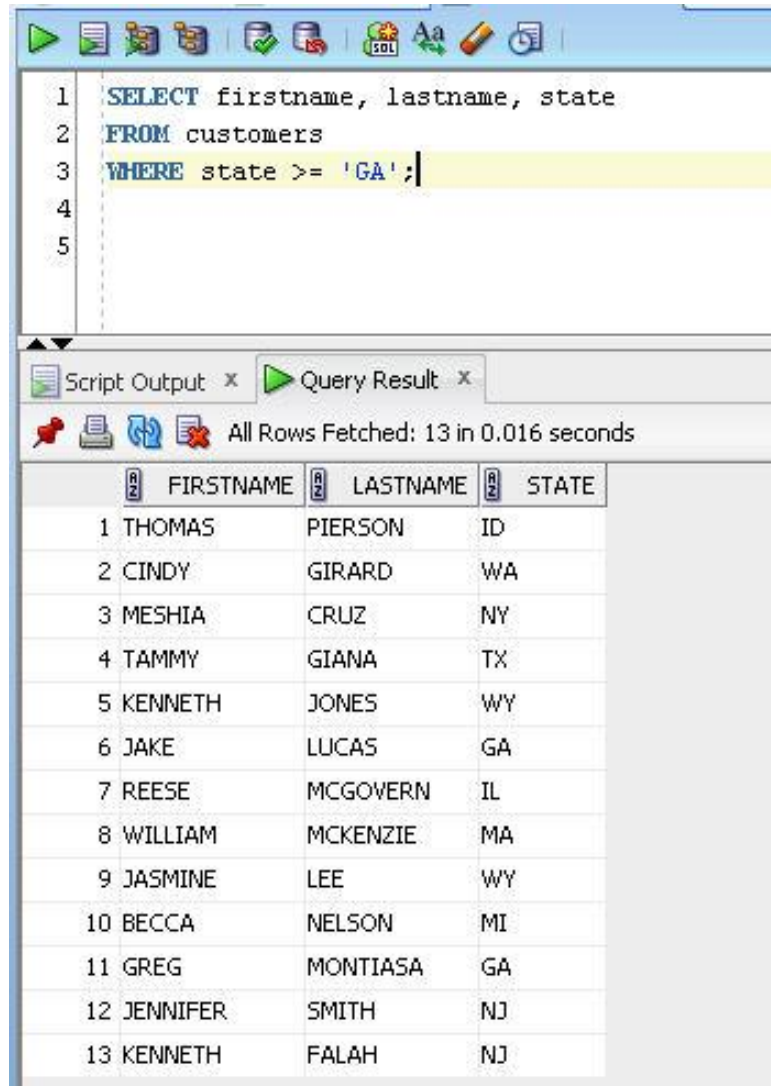
The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 SELECT firstname, lastname, state
2 FROM customers
3 WHERE state <= 'GA';
4
5
```

Below the query editor, there is a 'Script Output' tab and a 'Query Result' tab. The 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 9 in 0.016 seconds'.

	FIRSTNAME	LASTNAME	STATE
1	BONITA	MORALES	FL
2	RYAN	THOMPSON	CA
3	LEILA	SMITH	FL
4	JORGE	PEREZ	CA
5	JAKE	LUCAS	GA
6	NICHOLAS	NGUYEN	FL
7	STEVE	SCHELL	FL
8	MICHELL	DAUM	CA
9	GREG	MONTIASA	GA

# Comparison Operators



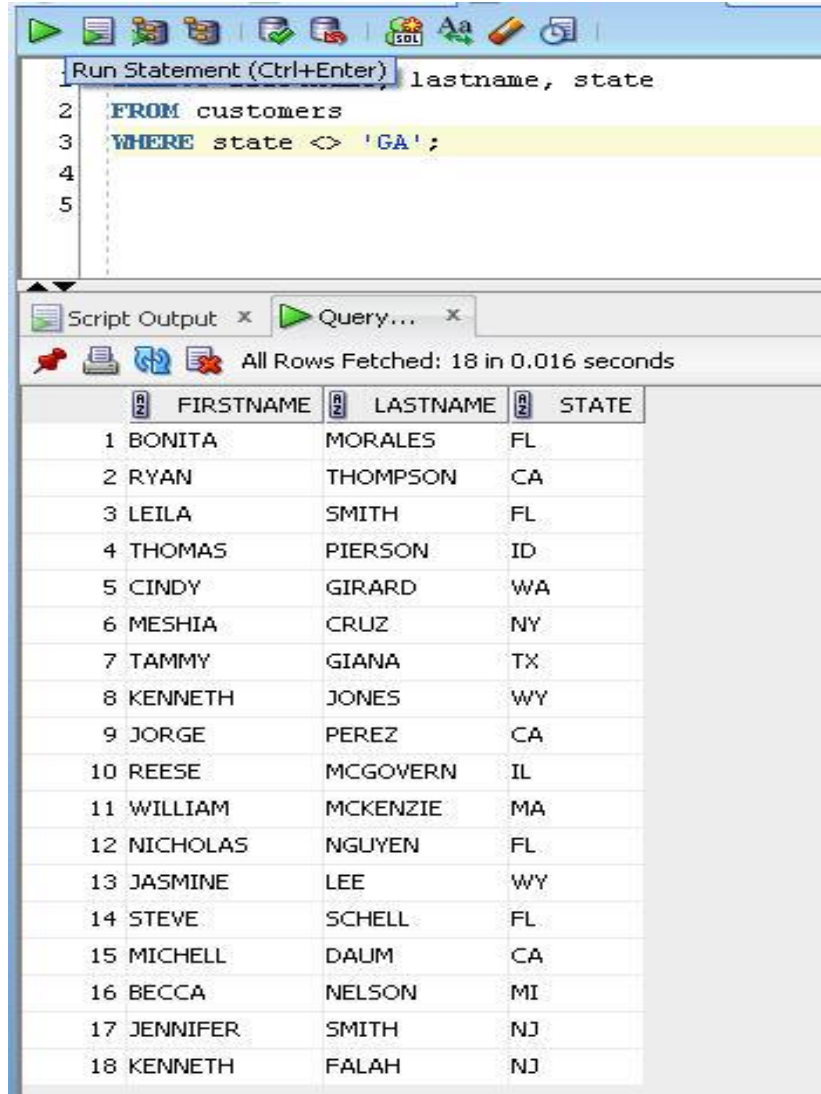
1 `SELECT` firstname, lastname, state  
2 `FROM` customers  
3 `WHERE` state >= 'GA';  
4  
5

Script Output x Query Result x

All Rows Fetched: 13 in 0.016 seconds

	FIRSTNAME	LASTNAME	STATE
1	THOMAS	PIERSON	ID
2	CINDY	GIRARD	WA
3	MESHIA	CRUZ	NY
4	TAMMY	GIANA	TX
5	KENNETH	JONES	WY
6	JAKE	LUCAS	GA
7	REESE	MCGOVERN	IL
8	WILLIAM	MCKENZIE	MA
9	JASMINE	LEE	WY
10	BECCA	NELSON	MI
11	GREG	MONTIASA	GA
12	JENNIFER	SMITH	NJ
13	KENNETH	FALAH	NJ

# Comparison Operators



The screenshot shows a SQL IDE window. The top toolbar includes icons for running queries, saving, and editing. The main text area contains a SQL query: `SELECT lastname, state FROM customers WHERE state <> 'GA';`. The query is highlighted in yellow. Below the query, the 'Script Output' tab is active, displaying the results of the query. The results are shown in a table with columns: FIRSTNAME, LASTNAME, and STATE. The table contains 18 rows of data, all of which are displayed. The status bar at the bottom of the results pane indicates 'All Rows Fetched: 18 in 0.016 seconds'.

	FIRSTNAME	LASTNAME	STATE
1	BONITA	MORALES	FL
2	RYAN	THOMPSON	CA
3	LEILA	SMITH	FL
4	THOMAS	PIERSON	ID
5	CINDY	GIRARD	WA
6	MESHIA	CRUZ	NY
7	TAMMY	GIANA	TX
8	KENNETH	JONES	WY
9	JORGE	PEREZ	CA
10	REESE	MCGOVERN	IL
11	WILLIAM	MCKENZIE	MA
12	NICHOLAS	NGUYEN	FL
13	JASMINE	LEE	WY
14	STEVE	SHELL	FL
15	MICHELL	DAUM	CA
16	BECCA	NELSON	MI
17	JENNIFER	SMITH	NJ
18	KENNETH	FALAH	NJ

# Other Comparison Operators

Operator	Description
[NOT] BETWEEN x AND y	Allows a user to specify a range of values
[NOT] IN (x, y, ...)	Similar to the OR operator
[NOT] LIKE	Use for searching patterns, uses the wildcards of % and _
IS [NOT] NULL	Allows users to search for records that have no entry in the specified field

# BETWEEN ... AND Operator

- Searches a range between a low boundary and a high boundary
  - Boundaries must be specified in that order
- Begin and end points are inclusive

# BETWEEN ... AND Operator

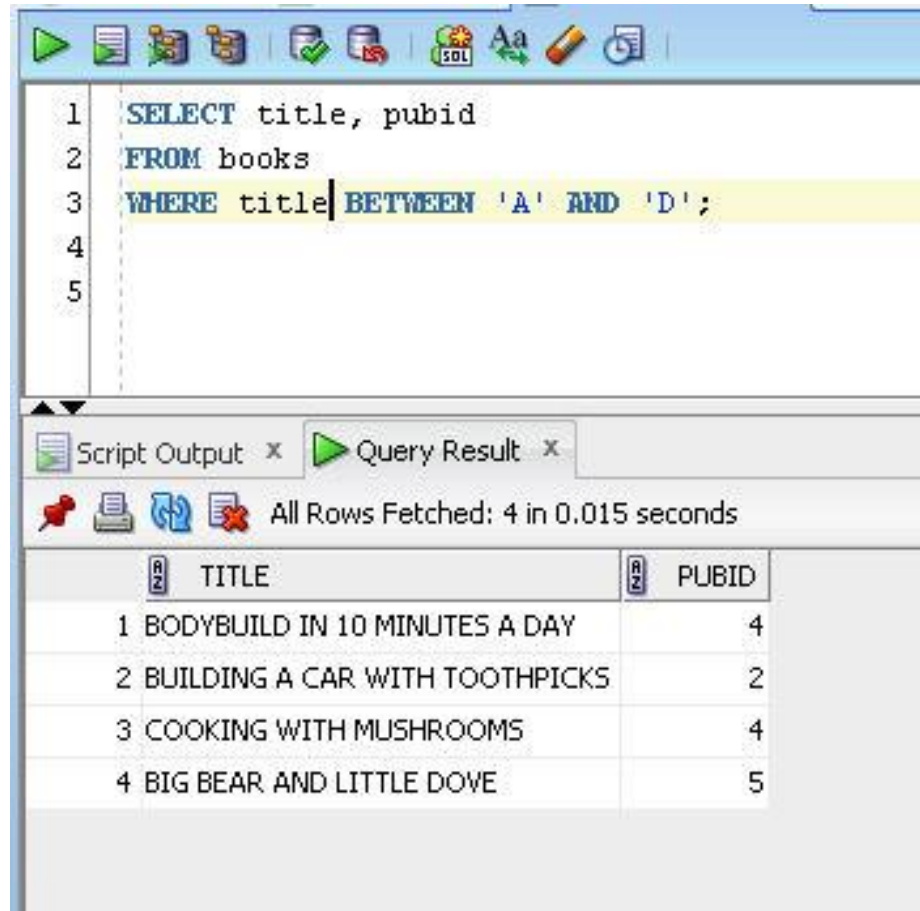
The screenshot shows a SQL IDE interface. The top toolbar contains icons for running queries, saving, and other database functions. The main text area displays the following SQL query:

```
1 SELECT title, pubid
2 FROM books
3 WHERE pubid BETWEEN 1 AND 3;
4
5
```

Below the query editor, the 'Script Output' window shows the execution status: 'All Rows Fetched: 7 in 0.016 seconds'. Below this, a table displays the results of the query:

	TITLE	PUBID
1	REVENGE OF MICKEY	1
2	BUILDING A CAR WITH TOOTHPICKS	2
3	DATABASE IMPLEMENTATION	3
4	HOLY GRAIL OF ORACLE	3
5	HANDCRANKED COMPUTERS	3
6	E-BUSINESS THE EASY WAY	2
7	HOW TO MANAGE THE MANAGER	1

# BETWEEN ... AND Operator



The screenshot shows a SQL query editor with a toolbar at the top. The query text is as follows:

```
1 SELECT title, pubid
2 FROM books
3 WHERE title BETWEEN 'A' AND 'D';
4
5
```

Below the query editor, there is a 'Script Output' tab and a 'Query Result' tab. The 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 4 in 0.015 seconds'.

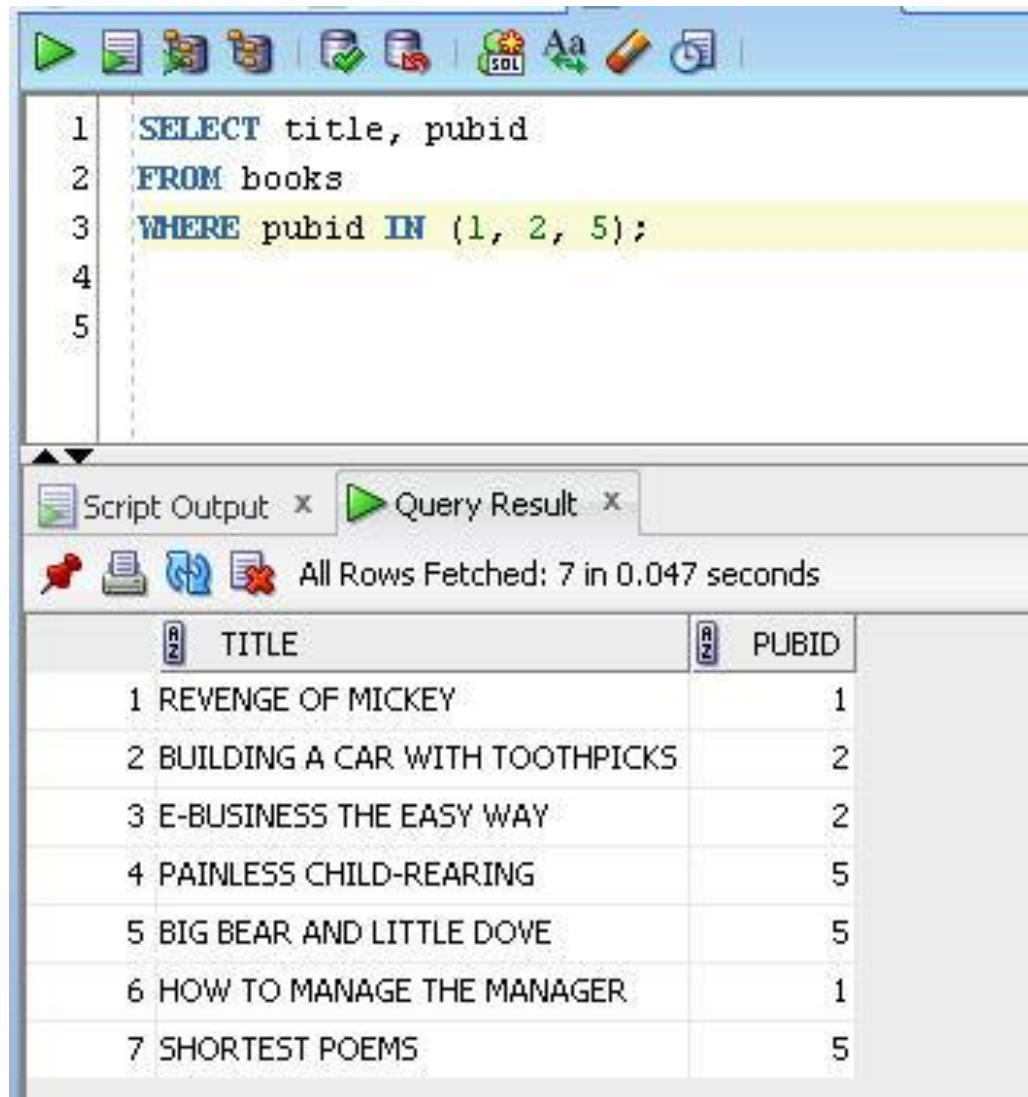
	TITLE	PUBID
1	BODYBUILD IN 10 MINUTES A DAY	4
2	BUILDING A CAR WITH TOOTHPICKS	2
3	COOKING WITH MUSHROOMS	4
4	BIG BEAR AND LITTLE DOVE	5

# The IN Operator

- Returns values that match one of the values specified in the value list
- List must be enclosed in parentheses and the values in the list must be separated by commas



# The IN Operator



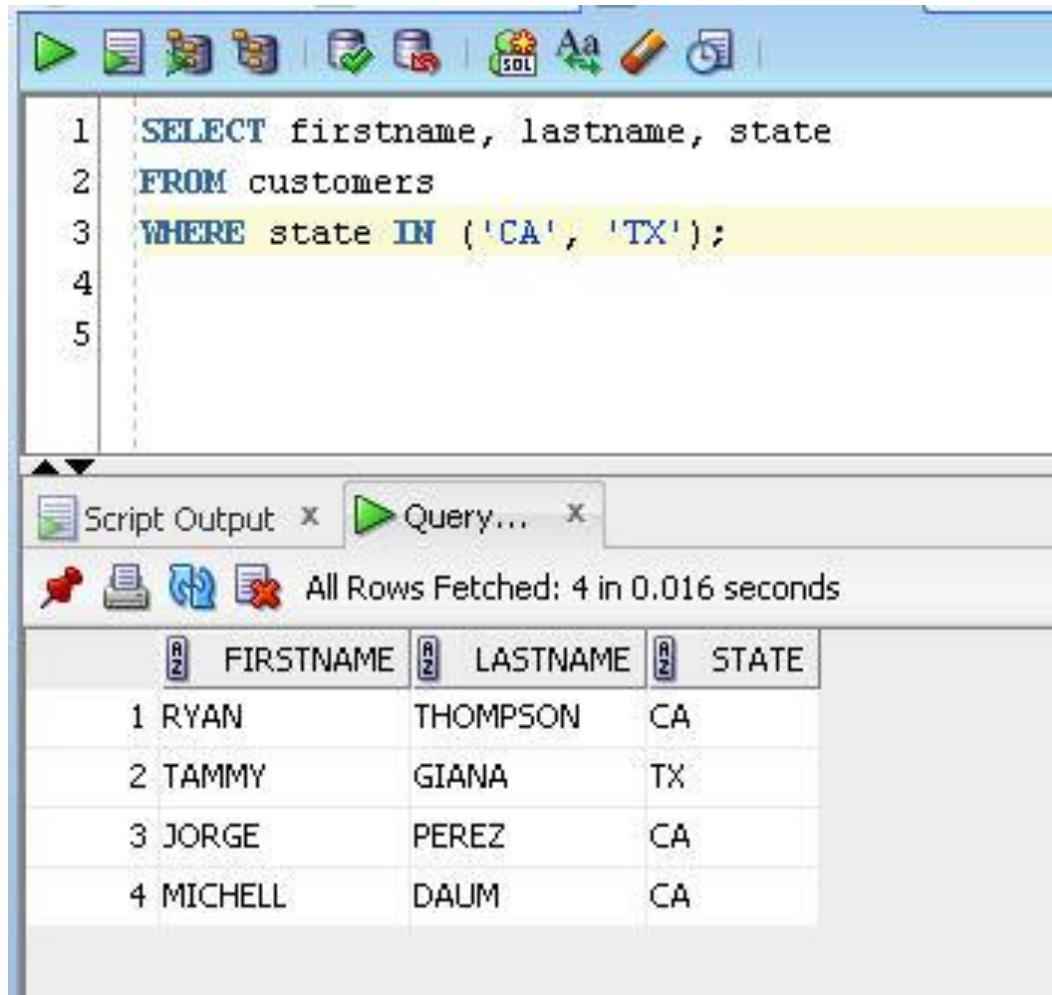
The screenshot displays a SQL query editor window. The query is as follows:

```
1 SELECT title, pubid
2 FROM books
3 WHERE pubid IN (1, 2, 5);
4
5
```

Below the query editor, the 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 7 in 0.047 seconds'.

	TITLE	PUBID
1	REVENGE OF MICKEY	1
2	BUILDING A CAR WITH TOOTHPICKS	2
3	E-BUSINESS THE EASY WAY	2
4	PAINLESS CHILD-REARING	5
5	BIG BEAR AND LITTLE DOVE	5
6	HOW TO MANAGE THE MANAGER	1
7	SHORTEST POEMS	5

# The IN Operator



The screenshot displays a SQL development environment. The top pane shows a query using the IN operator to filter customers by state. The bottom pane shows the results of the query, which are four rows of customer data.

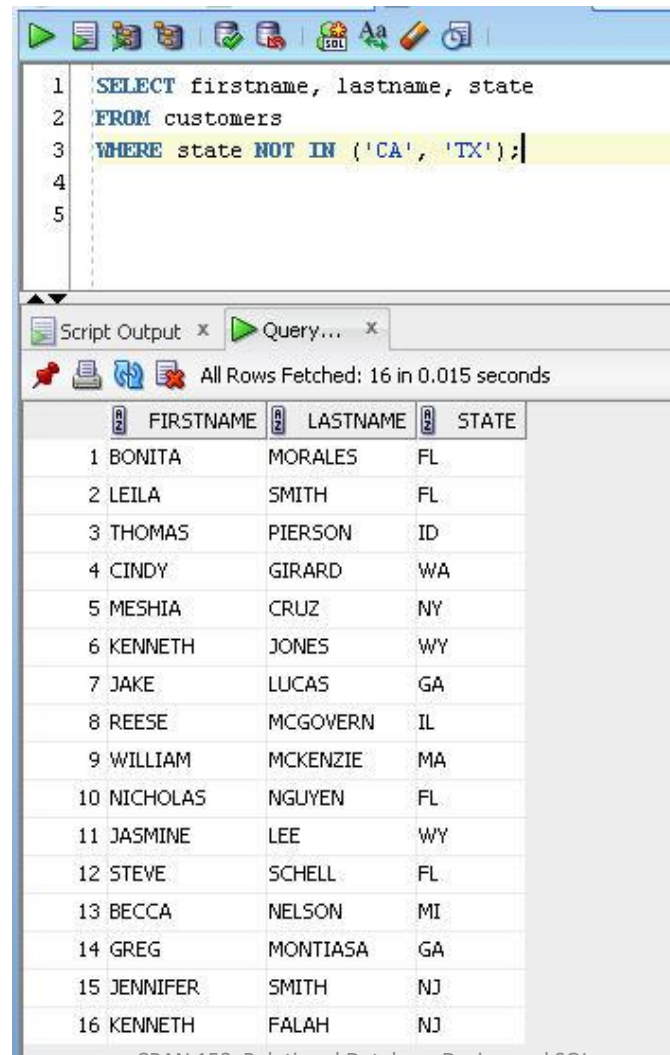
```
1 SELECT firstname, lastname, state
2 FROM customers
3 WHERE state IN ('CA', 'TX');
4
5
```

Script Output x Query... x

All Rows Fetched: 4 in 0.016 seconds

	FIRSTNAME	LASTNAME	STATE
1	RYAN	THOMPSON	CA
2	TAMMY	GIANA	TX
3	JORGE	PEREZ	CA
4	MICHELL	DAUM	CA

# The IN Operator



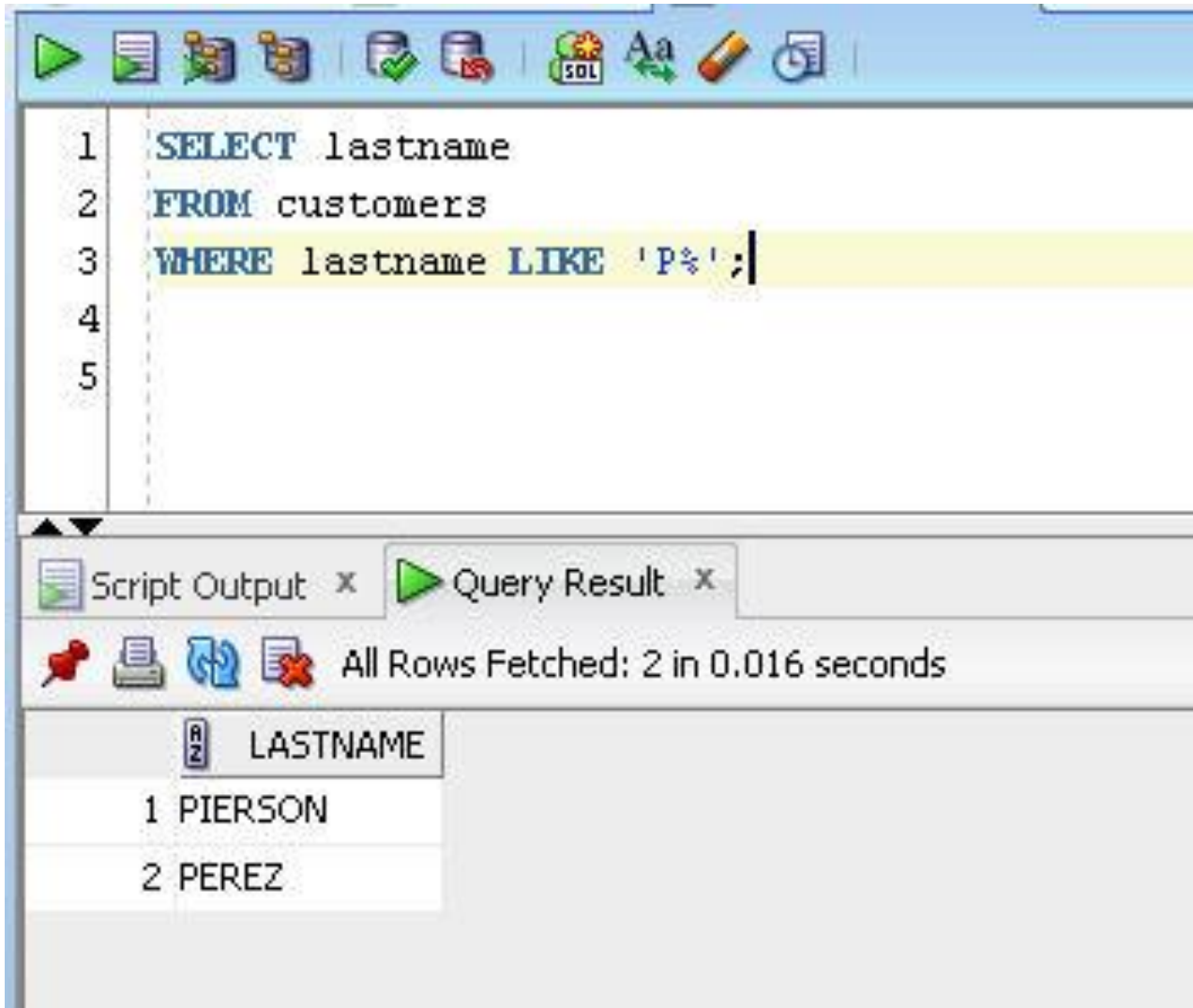
The screenshot shows a SQL IDE window. The top pane contains a SQL query: `SELECT firstname, lastname, state FROM customers WHERE state NOT IN ('CA', 'TX');`. The bottom pane shows the results of the query, which are 16 rows of customer data. The results are displayed in a table with columns: FIRSTNAME, LASTNAME, and STATE. The rows are numbered 1 through 16.

	FIRSTNAME	LASTNAME	STATE
1	BONITA	MORALES	FL
2	LEILA	SMITH	FL
3	THOMAS	PIERSON	ID
4	CINDY	GIRARD	WA
5	MESHIA	CRUZ	NY
6	KENNETH	JONES	WY
7	JAKE	LUCAS	GA
8	REESE	MCGOVERN	IL
9	WILLIAM	MCKENZIE	MA
10	NICHOLAS	NGUYEN	FL
11	JASMINE	LEE	WY
12	STEVE	SHELL	FL
13	BECCA	NELSON	MI
14	GREG	MONTIASA	GA
15	JENNIFER	SMITH	NJ
16	KENNETH	FALAH	NJ

# LIKE Operator

- The **LIKE** operator is used with wild cards to search for patterns
- Wildcard characters can be used to represent one or more alphanumeric characters
- The wildcard characters available for Oracle 10g are **%** and underscore **\_**
- **%** is used to represent any number of characters, while **underscore** is used to represent exactly one character

# LIKE Operator



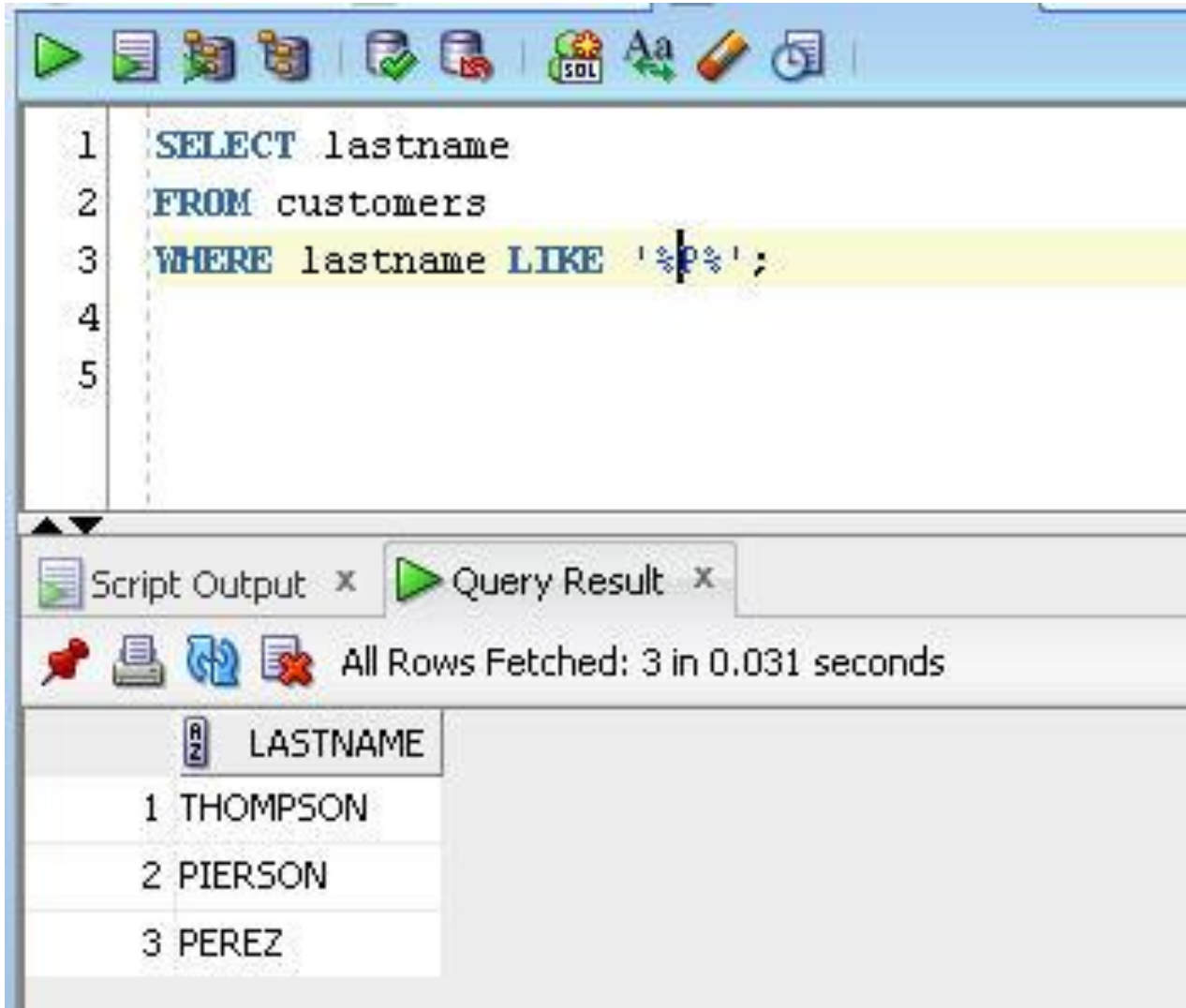
The screenshot shows a SQL IDE interface. The top toolbar contains icons for running queries, saving, and other database functions. The main text area displays a SQL query:

```
1 SELECT lastname
2 FROM customers
3 WHERE lastname LIKE 'P%';
4
5
```

Below the query editor, the 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 2 in 0.016 seconds'.

LASTNAME
1 PIERSON
2 PEREZ

# LIKE Operator



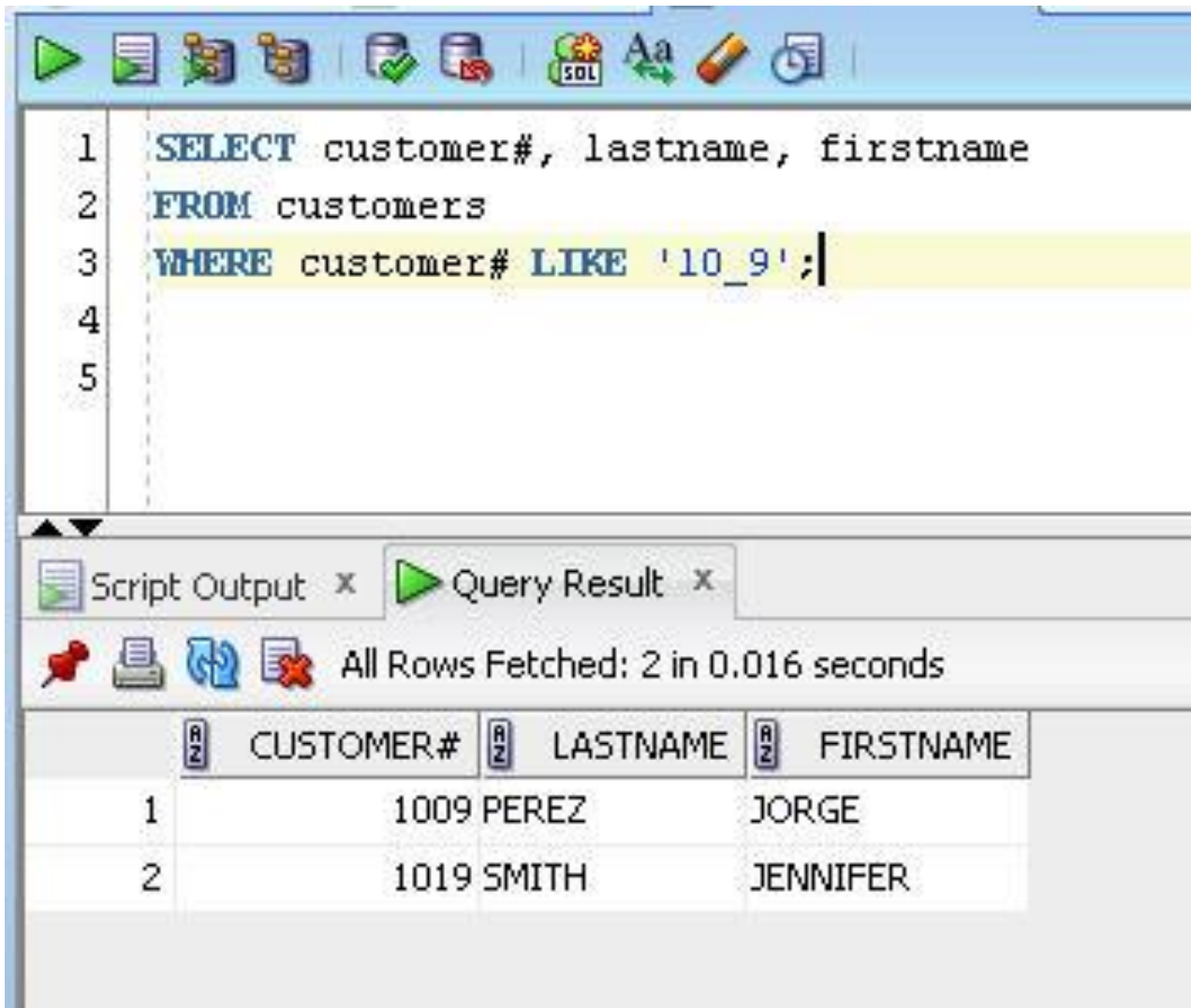
The screenshot shows a SQL IDE interface. The main editor window contains the following SQL query:

```
1 SELECT lastname
2 FROM customers
3 WHERE lastname LIKE '%P%';
4
5
```

The third line of the query is highlighted in yellow. Below the editor, there is a tabbed interface with two tabs: "Script Output" and "Query Result". The "Query Result" tab is active, displaying the results of the query. Above the results table, it says "All Rows Fetched: 3 in 0.031 seconds".

	LASTNAME
1	THOMPSON
2	PIERSON
3	PEREZ

# LIKE Operator



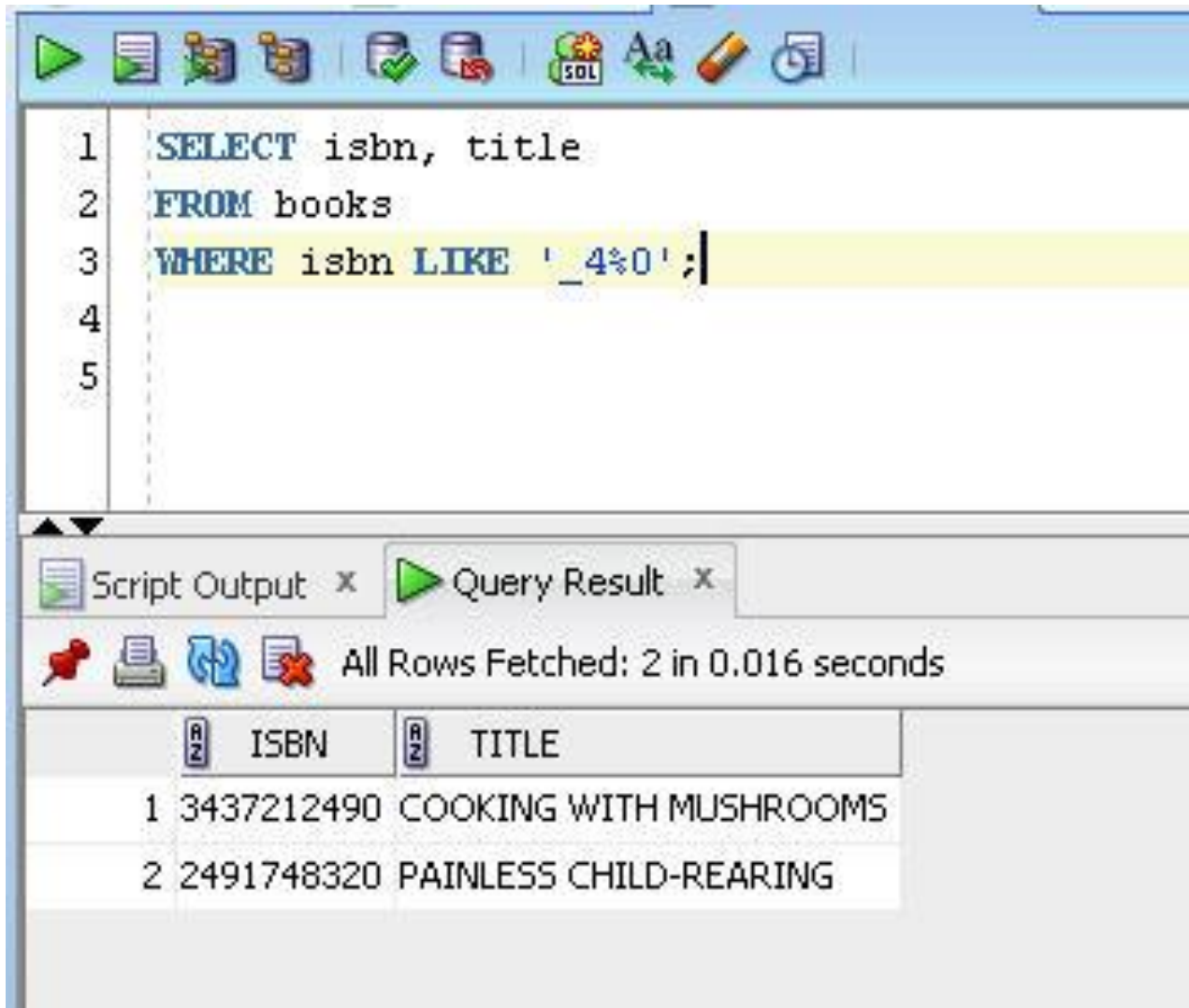
The screenshot displays a SQL IDE interface. The top toolbar includes icons for execution, saving, undo, redo, and other standard database operations. The main text area contains the following SQL query:

```
1 SELECT customer#, lastname, firstname
2 FROM customers
3 WHERE customer# LIKE '10_9';
4
5
```

Below the query editor, the 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 2 in 0.016 seconds'. The results are displayed in a table with three columns: CUSTOMER#, LASTNAME, and FIRSTNAME.

	CUSTOMER#	LASTNAME	FIRSTNAME
1	1009	PEREZ	JORGE
2	1019	SMITH	JENNIFER

# LIKE Operator



The screenshot shows a database query tool interface. The top toolbar contains icons for execution, saving, and editing. The main text area displays a SQL query:

```
1 SELECT isbn, title
2 FROM books
3 WHERE isbn LIKE '_4%0';
4
5
```

Below the query editor, the 'Query Result' tab is active, showing the status 'All Rows Fetched: 2 in 0.016 seconds'. The results are displayed in a table with two columns: ISBN and TITLE.

	ISBN	TITLE
1	3437212490	COOKING WITH MUSHROOMS
2	2491748320	PAINLESS CHILD-REARING



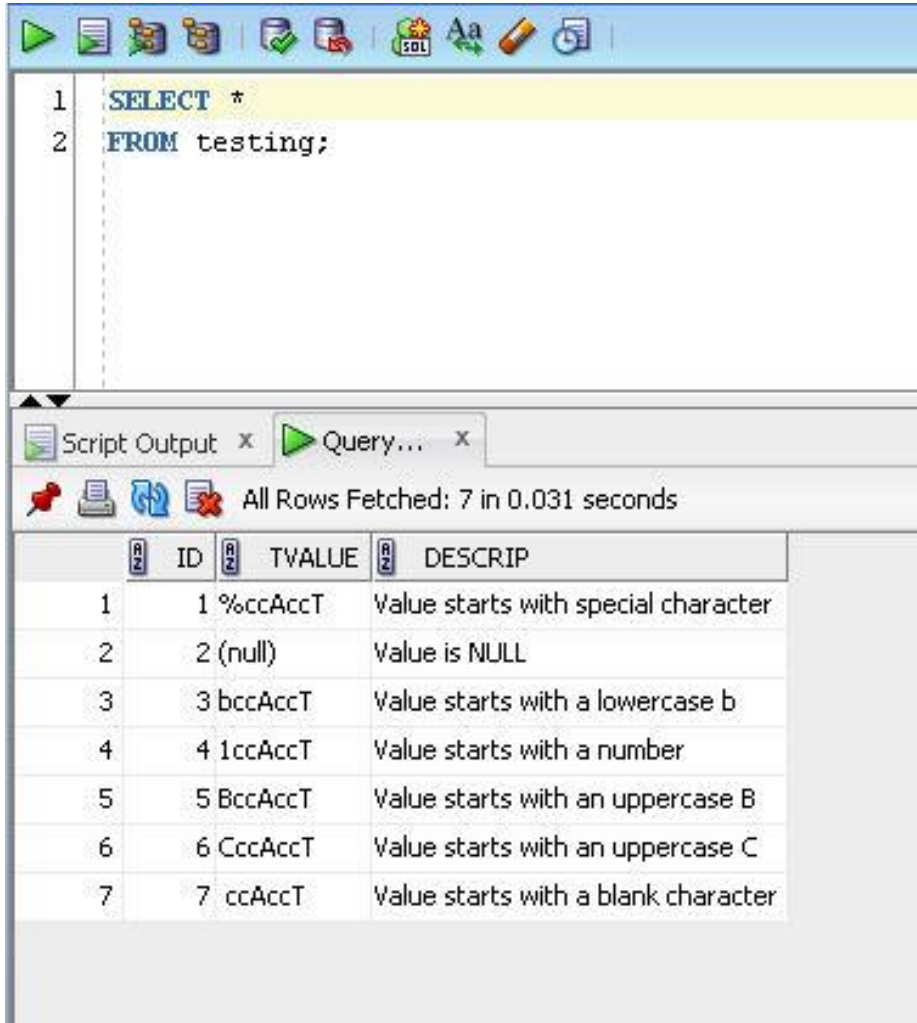
# LIKE Operator

- What if you need to use the LIKE operator to search for patterns but also need to search for a wildcard character as a literal in your value
- You need to search value that starts with a % symbol, and contains an uppercase A as the fourth character, and ends with an uppercase T
- In this query you need to use the wildcard characters \_ and % with the LIKE operator but you also need to search for a literal % symbol as the first character

# LIKE Operator

- The LIKE operator includes the ESCAPE option for indicating when wildcard symbols should be used as literals rather than translated as wildcard characters
- This option allows the user to select the escape character
- The escape character must precede any wildcard characters in the search pattern that should be interpreted literally, not as wildcard characters

# LIKE Operator



The screenshot shows a SQL IDE interface. The top toolbar contains icons for running queries, saving, and other database functions. The main editor window displays the following SQL query:

```
1 SELECT *
2 FROM testing;
```

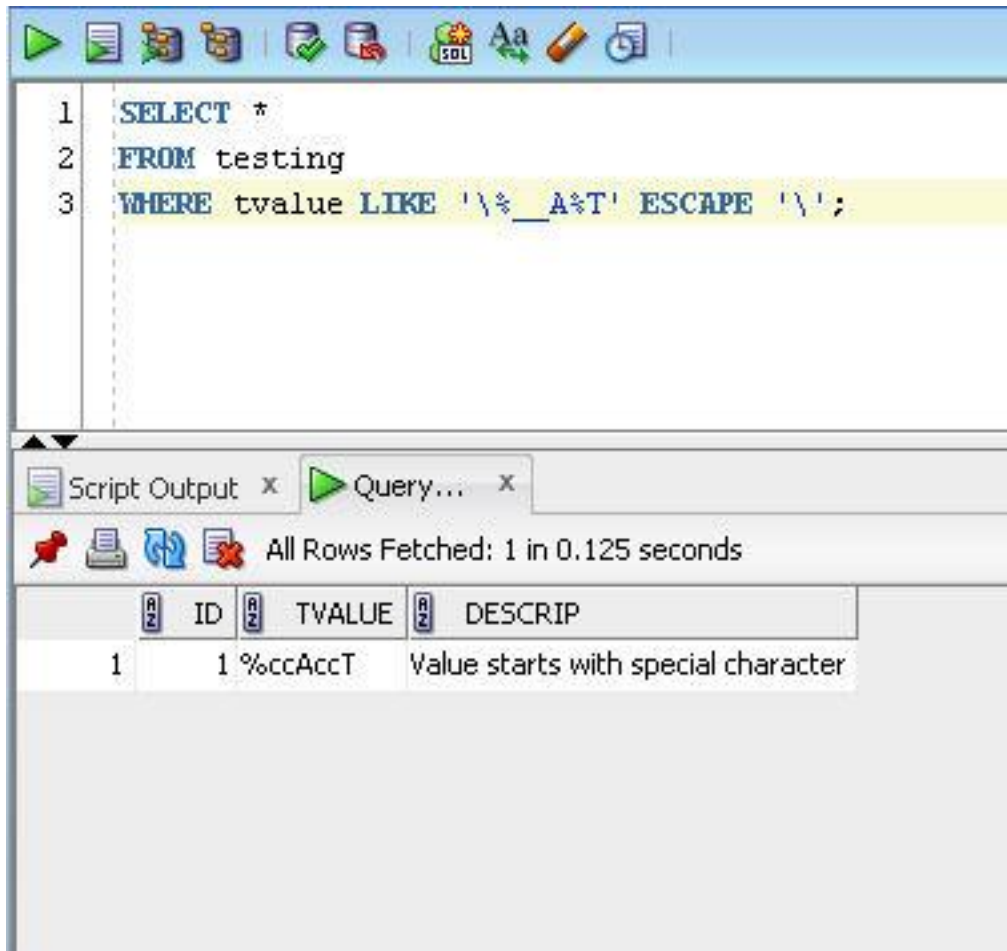
Below the editor, the 'Script Output' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 7 in 0.031 seconds'. The results are displayed in a table with three columns: ID, TVALUE, and DESCRIP.

ID	TVALUE	DESCRIP
1	%ccAccT	Value starts with special character
2	(null)	Value is NULL
3	bccAccT	Value starts with a lowercase b
4	1ccAccT	Value starts with a number
5	BccAccT	Value starts with an uppercase B
6	CccAccT	Value starts with an uppercase C
7	ccAccT	Value starts with a blank character

Notice the first row in this table, it begins with a % symbol

This table is not part of the Just Lee script, I will provide a new script for you to run to create this table

# LIKE Operator



The screenshot shows a SQL query editor with the following query:

```
1 SELECT *
2 FROM testing
3 WHERE tvalue LIKE '\\%_A*T' ESCAPE '\\';
```

The query is executed, and the results are displayed in a table with the following columns: ID, TVALUE, and DESCRIP.

ID	TVALUE	DESCRIP
1	1 %ccAccT	Value starts with special character

Notice the \ character in front of the first % symbol

This is the ESCAPE character, its function is defined with the ESCAPE keyword followed by the \ symbol defining it as the escape symbol

The first % symbol is taken as a literal % symbol, where the second % in the statement is used a wildcard because the ESCAPE symbol is not used in front of it

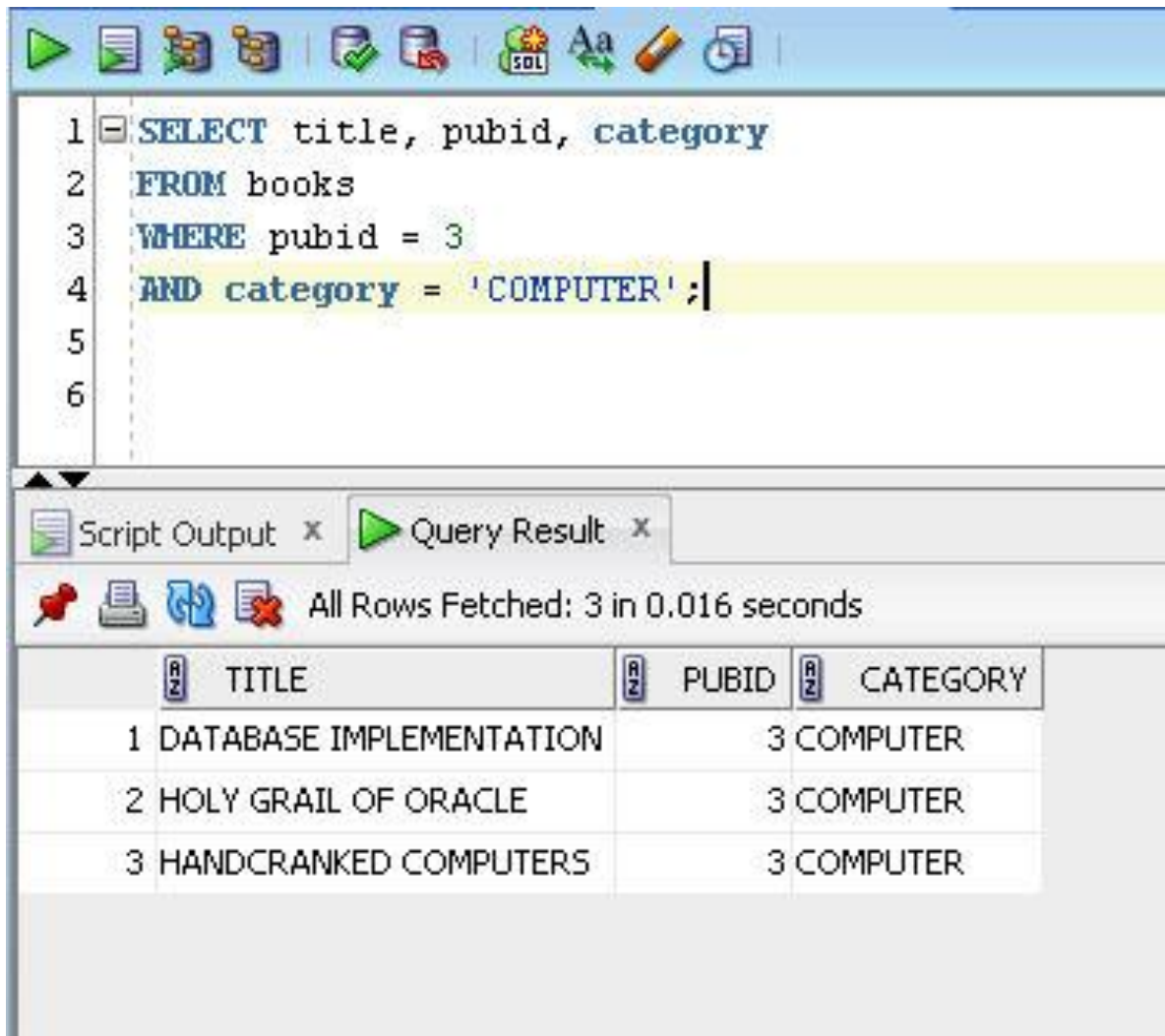
# Logical Operators

- There may be times when you need to search for records based on two or more search conditions
- You use **logical operators** to combine search conditions
- The logical operators **AND** and **OR** are the most common
- The **NOT** operator we mentioned earlier is also a logical operator. It is used to reverse the meaning of a search condition, rather than to combine them

# Logical Operators

- When queried with a **WHERE** clause, each record in the table is compared to the condition
- If the condition is **TRUE** when compared to a record the record is included in the results
- When the **AND** operator is used in a **WHERE** clause, both conditions (combined by the **AND** operator), must evaluate as being **TRUE** for the record to be included in the results

# Logical Operators



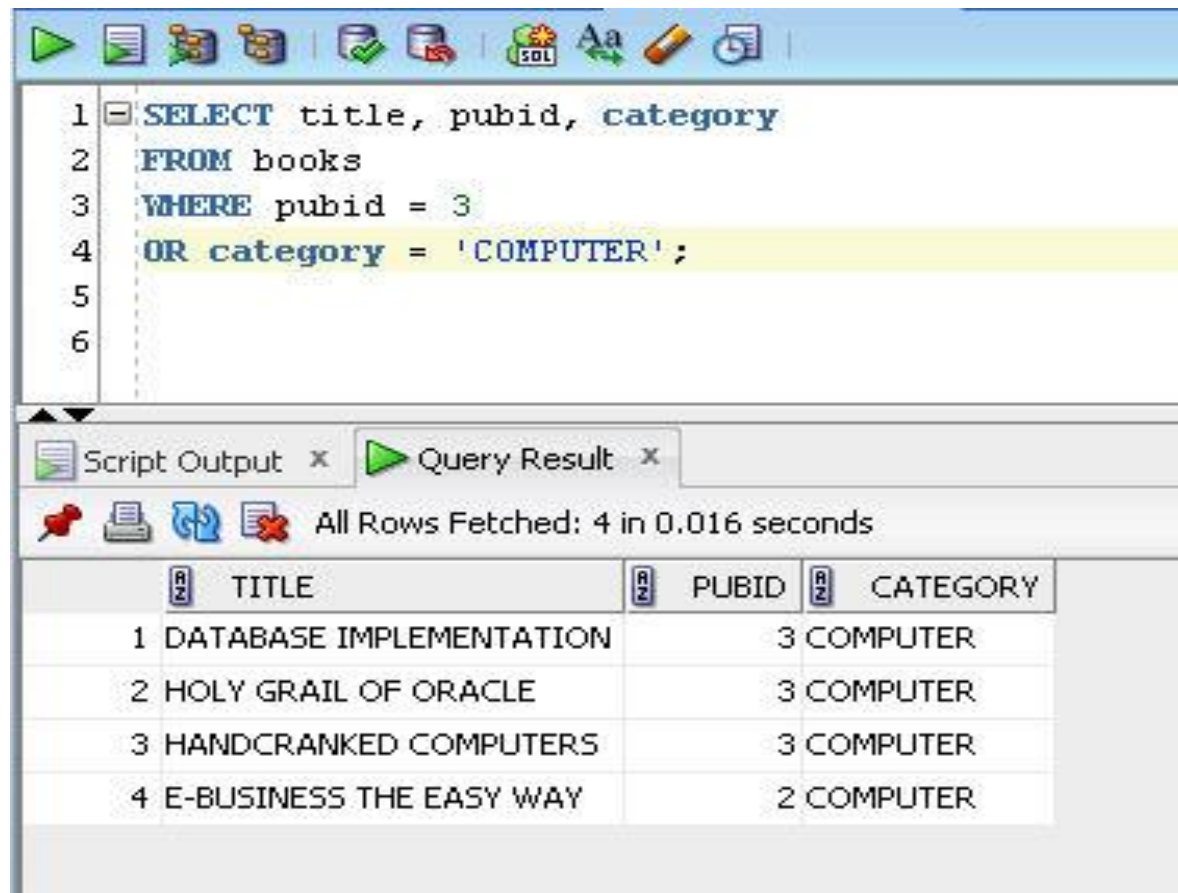
The screenshot displays a SQL query editor with a toolbar at the top. The query text is as follows:

```
1 SELECT title, pubid, category
2 FROM books
3 WHERE pubid = 3
4 AND category = 'COMPUTER';
5
6
```

Below the query editor, the 'Query Result' tab is active, showing the execution status: 'All Rows Fetched: 3 in 0.016 seconds'. The results are presented in a table with three columns: TITLE, PUBID, and CATEGORY.

	TITLE	PUBID	CATEGORY
1	DATABASE IMPLEMENTATION	3	COMPUTER
2	HOLY GRAIL OF ORACLE	3	COMPUTER
3	HANDCRANKED COMPUTERS	3	COMPUTER

# Logical Operators



The screenshot shows a SQL query editor with a toolbar at the top. The query text is as follows:

```
1 SELECT title, pubid, category
2 FROM books
3 WHERE pubid = 3
4 OR category = 'COMPUTER';
5
6
```

Below the query editor, the 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 4 in 0.016 seconds'. The results are displayed in a table with three columns: TITLE, PUBID, and CATEGORY.

	TITLE	PUBID	CATEGORY
1	DATABASE IMPLEMENTATION	3	COMPUTER
2	HOLY GRAIL OF ORACLE	3	COMPUTER
3	HANDCRANKED COMPUTERS	3	COMPUTER
4	E-BUSINESS THE EASY WAY	2	COMPUTER

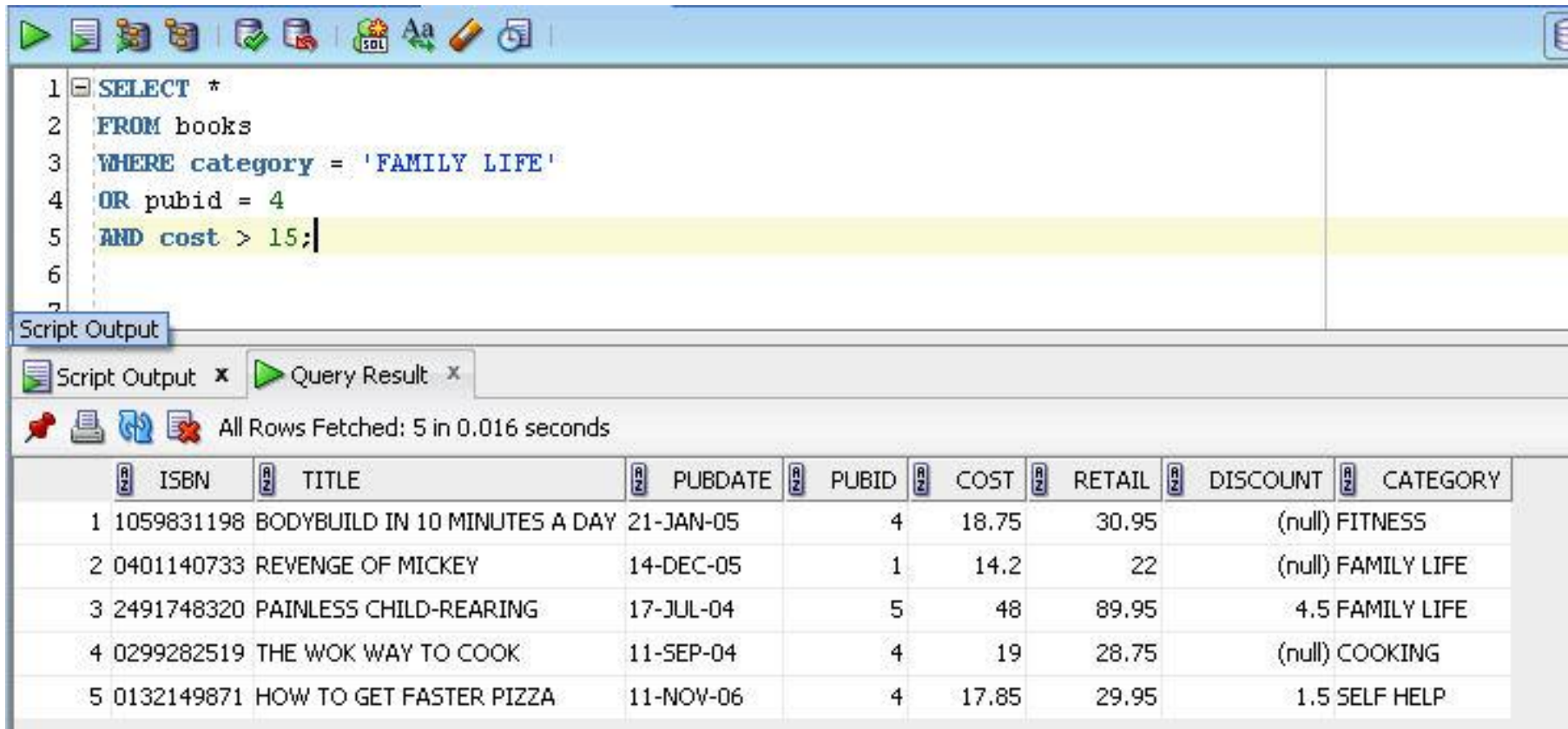
If you are interested in a list of books published by Publisher 3 OR in the COMPUTER Category, you can use the OR operator



# Logical Operators

- With the **OR** operator only one of the conditions must evaluate to **TRUE** to have the record included in the results
- Using a series of **OR** logical operators to join conditions that are based on the same column is identical to using the **IN** comparison operator

# Logical Operators



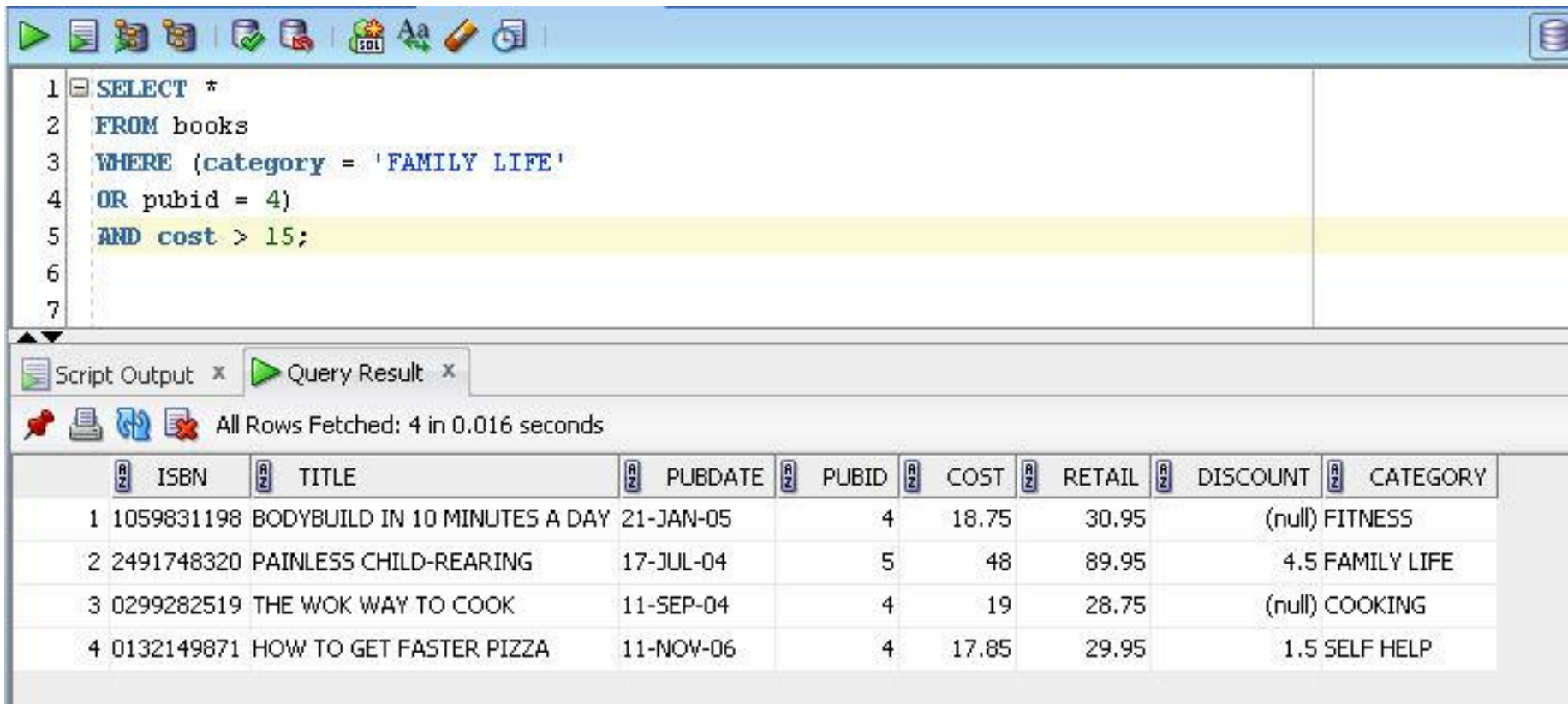
The screenshot shows a SQL query editor with a toolbar at the top. The query text is as follows:

```
1 SELECT *
2 FROM books
3 WHERE category = 'FAMILY LIFE'
4 OR pubid = 4
5 AND cost > 15;
6
7
```

Below the query editor, the 'Script Output' tab is active, displaying the query results. The status bar indicates 'All Rows Fetched: 5 in 0.016 seconds'.

	ISBN	TITLE	PUBDATE	PUBID	COST	RETAIL	DISCOUNT	CATEGORY
1	1059831198	BODYBUILD IN 10 MINUTES A DAY	21-JAN-05	4	18.75	30.95	(null)	FITNESS
2	0401140733	REVENGE OF MICKEY	14-DEC-05	1	14.2	22	(null)	FAMILY LIFE
3	2491748320	PAINLESS CHILD-REARING	17-JUL-04	5	48	89.95	4.5	FAMILY LIFE
4	0299282519	THE WOK WAY TO COOK	11-SEP-04	4	19	28.75	(null)	COOKING
5	0132149871	HOW TO GET FASTER PIZZA	11-NOV-06	4	17.85	29.95	1.5	SELF HELP

# Logical Operators



The screenshot shows a SQL query editor window with a toolbar at the top. The query text is as follows:

```
1 SELECT *
2 FROM books
3 WHERE (category = 'FAMILY LIFE'
4 OR pubid = 4)
5 AND cost > 15;
6
7
```

Below the query editor, there is a 'Query Result' tab. It shows the status 'All Rows Fetched: 4 in 0.016 seconds'. The results are displayed in a table with the following columns: ISBN, TITLE, PUBDATE, PUBID, COST, RETAIL, DISCOUNT, and CATEGORY.

	ISBN	TITLE	PUBDATE	PUBID	COST	RETAIL	DISCOUNT	CATEGORY
1	1059831198	BODYBUILD IN 10 MINUTES A DAY	21-JAN-05	4	18.75	30.95	(null)	FITNESS
2	2491748320	PAINLESS CHILD-REARING	17-JUL-04	5	48	89.95	4.5	FAMILY LIFE
3	0299282519	THE WOK WAY TO COOK	11-SEP-04	4	19	28.75	(null)	COOKING
4	0132149871	HOW TO GET FASTER PIZZA	11-NOV-06	4	17.85	29.95	1.5	SELF HELP

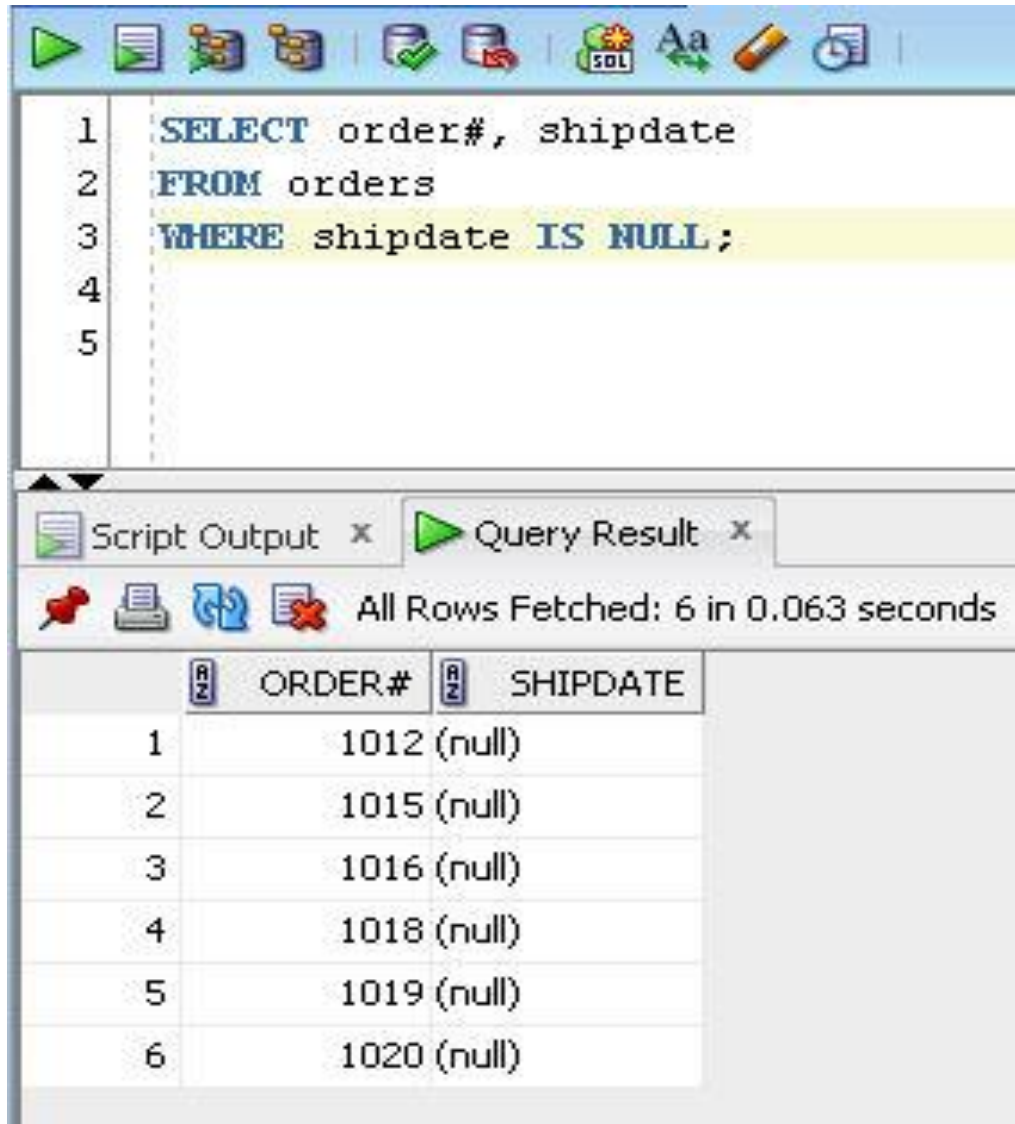
# Logical Operators

- The order of the logical operators is important
- The **WHERE** clause can contain multiple types of operators. You have to know the order in which they are resolved:
  - Arithmetic operations are resolved first
  - Comparison operators (<, >, =, LIKE, etc) are solved next
  - Logical operators are lower in precedence and are evaluated last – in the order of NOT, AND and then OR
  - To change the order of evaluation, parentheses are used

# Treatment of NULL Values

- When performing arithmetic operations or search conditions, **NULL** values can return unexpected results
- A **NULL** value simply means that no value has been stored in a particular field
- A **NULL** is not a blank space. It is an unknown value since a value has not been supplied to a field
- When searching for **NULL** values you cannot use the **=** sign because there is not a value available to use for comparison in the search condition. This is where **IS NULL** is used

# Treatment of NULL Values



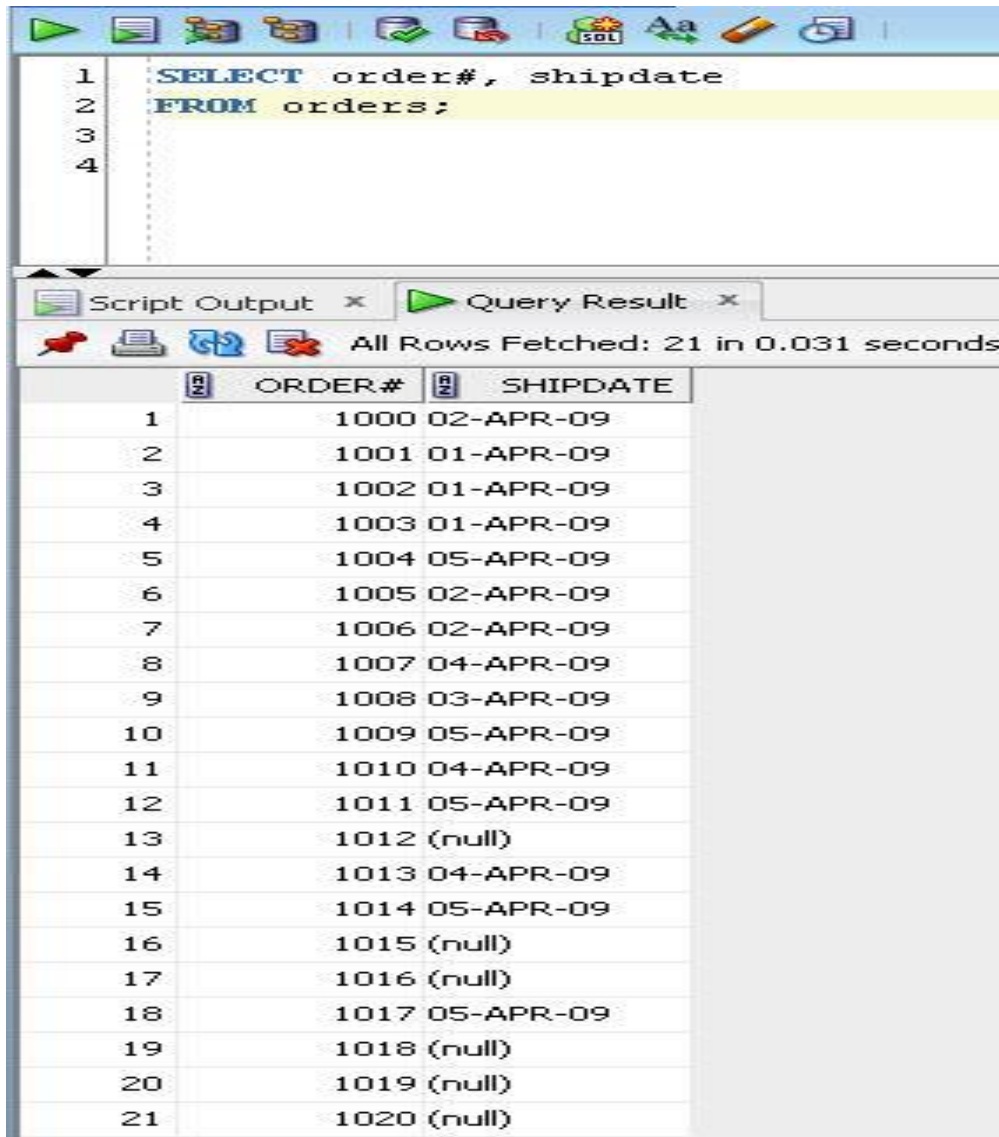
The screenshot shows a SQL query editor with a toolbar at the top. The query text is as follows:

```
1 SELECT order#, shipdate
2 FROM orders
3 WHERE shipdate IS NULL;
4
5
```

Below the query editor, there is a tabbed interface with 'Script Output' and 'Query Result'. The 'Query Result' tab is active, showing a status bar that reads 'All Rows Fetched: 6 in 0.063 seconds'. Below the status bar is a table with the following data:

	ORDER#	SHIPDATE
1	1012	(null)
2	1015	(null)
3	1016	(null)
4	1018	(null)
5	1019	(null)
6	1020	(null)

# Treatment of NULL Values



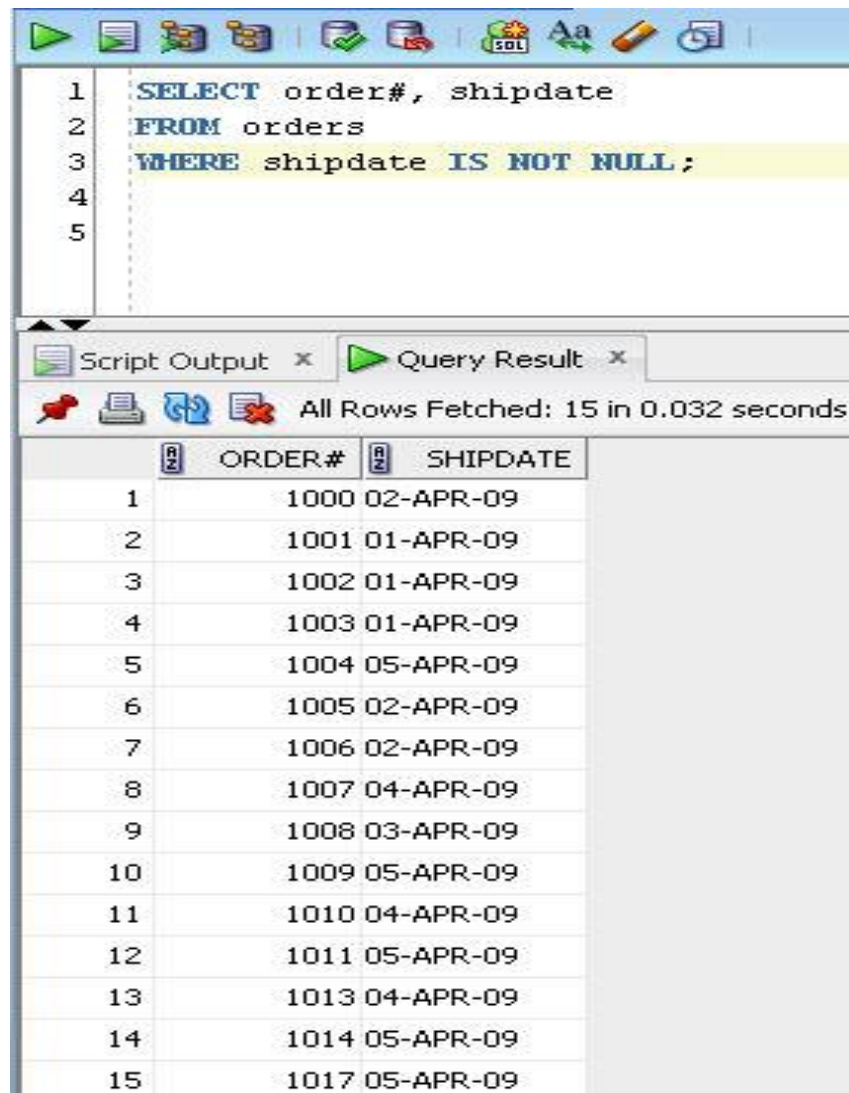
The screenshot shows a SQL query execution window. The query is: `SELECT order#, shipdate FROM orders;`. The results are displayed in a table with two columns: ORDER# and SHIPDATE. The table contains 21 rows. Rows 1 through 11 have valid dates. Rows 12 through 21 have NULL values for SHIPDATE, indicated by "(null)".

	ORDER#	SHIPDATE
1	1000	02-APR-09
2	1001	01-APR-09
3	1002	01-APR-09
4	1003	01-APR-09
5	1004	05-APR-09
6	1005	02-APR-09
7	1006	02-APR-09
8	1007	04-APR-09
9	1008	03-APR-09
10	1009	05-APR-09
11	1010	04-APR-09
12	1011	05-APR-09
13	1012	(null)
14	1013	04-APR-09
15	1014	05-APR-09
16	1015	(null)
17	1016	(null)
18	1017	05-APR-09
19	1018	(null)
20	1019	(null)
21	1020	(null)

Notice ORDER# 1012, 1015, 1016, 1018, 1019, and 1020 all have a NULL value for the SHIPDATE

This was confirmed on the previous slide

# Treatment of NULL Values



The screenshot displays a SQL query in a text editor and its corresponding results in a table. The query is as follows:

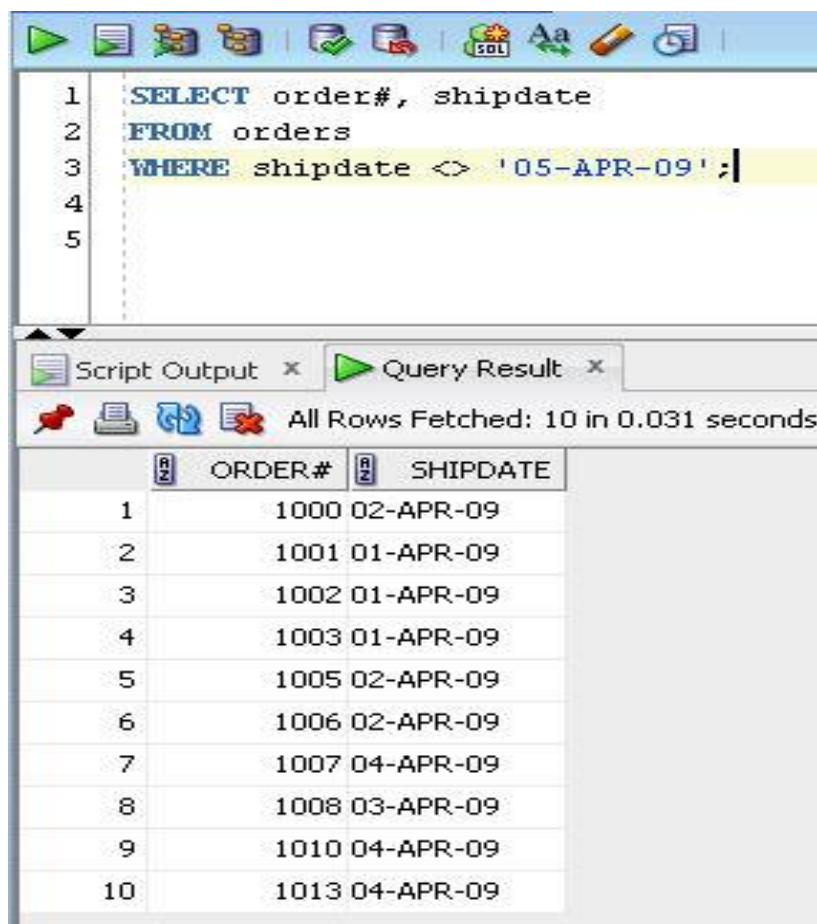
```
1 SELECT order#, shipdate
2 FROM orders
3 WHERE shipdate IS NOT NULL;
4
5
```

The results are shown in a table with the following columns: ORDER# and SHIPDATE. The table contains 15 rows of data, all of which have non-null ship dates.

	ORDER#	SHIPDATE
1	1000	02-APR-09
2	1001	01-APR-09
3	1002	01-APR-09
4	1003	01-APR-09
5	1004	05-APR-09
6	1005	02-APR-09
7	1006	02-APR-09
8	1007	04-APR-09
9	1008	03-APR-09
10	1009	05-APR-09
11	1010	04-APR-09
12	1011	05-APR-09
13	1013	04-APR-09
14	1014	05-APR-09
15	1017	05-APR-09



# Treatment of NULL Values



```
1 SELECT order#, shipdate
2 FROM orders
3 WHERE shipdate <> '05-APR-09';
4
5
```

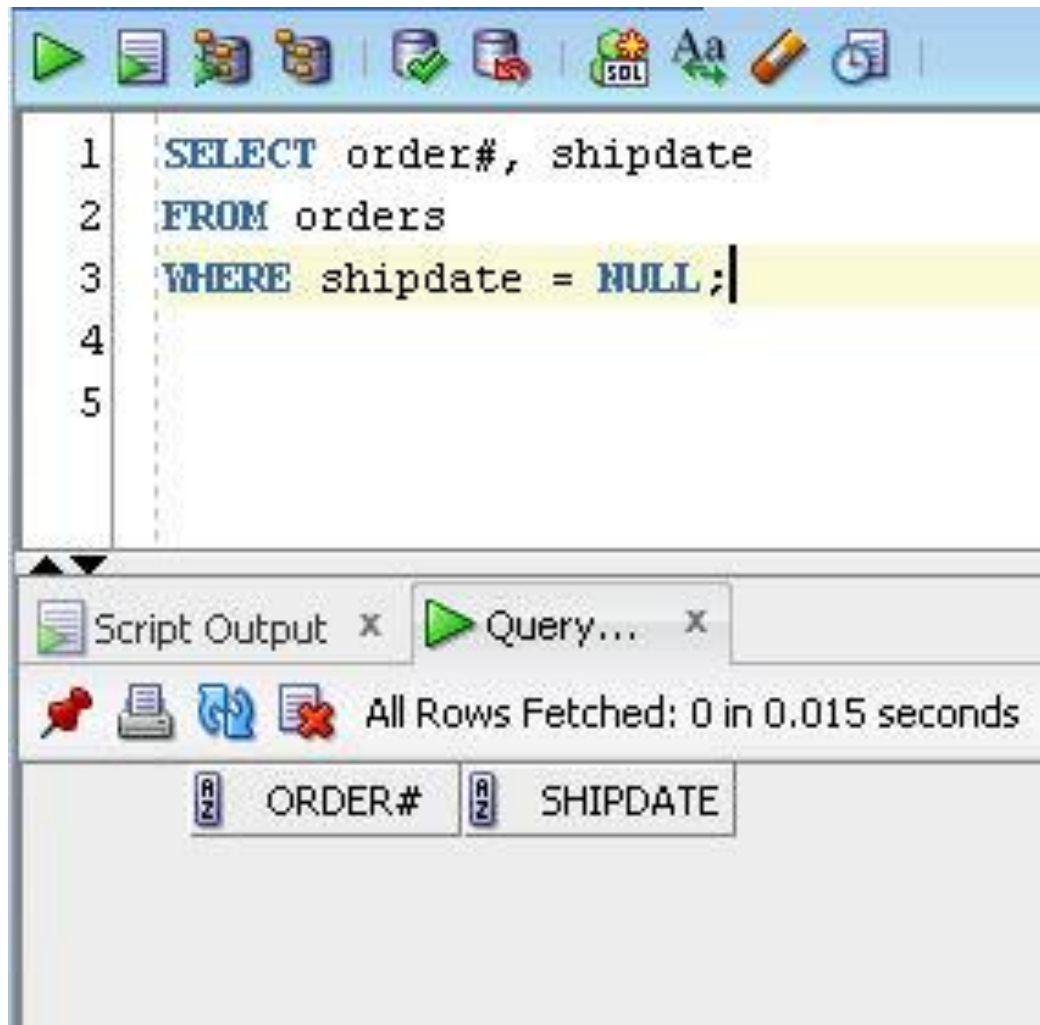
Script Output x Query Result x

All Rows Fetched: 10 in 0.031 seconds

	ORDER#	SHIPDATE
1	1000	02-APR-09
2	1001	01-APR-09
3	1002	01-APR-09
4	1003	01-APR-09
5	1005	02-APR-09
6	1006	02-APR-09
7	1007	04-APR-09
8	1008	03-APR-09
9	1010	04-APR-09
10	1013	04-APR-09

Notice that the records for ORDER# 1012, 1015, 1016, etc. are not visible since these do not contain a value for SHIPDATE

# Treatment of NULL Values



Be aware that using  
= NULL will never  
return an error

However it always  
returns no rows

# ORDER BY Clause

- The **ORDER BY** clause is used for displaying the results of a query in sorted order
- The **ORDER BY** clause is listed at the end of the SELECT statement
- You can sort output in either ascending (**ASC**) or descending (**DESC**) sequence

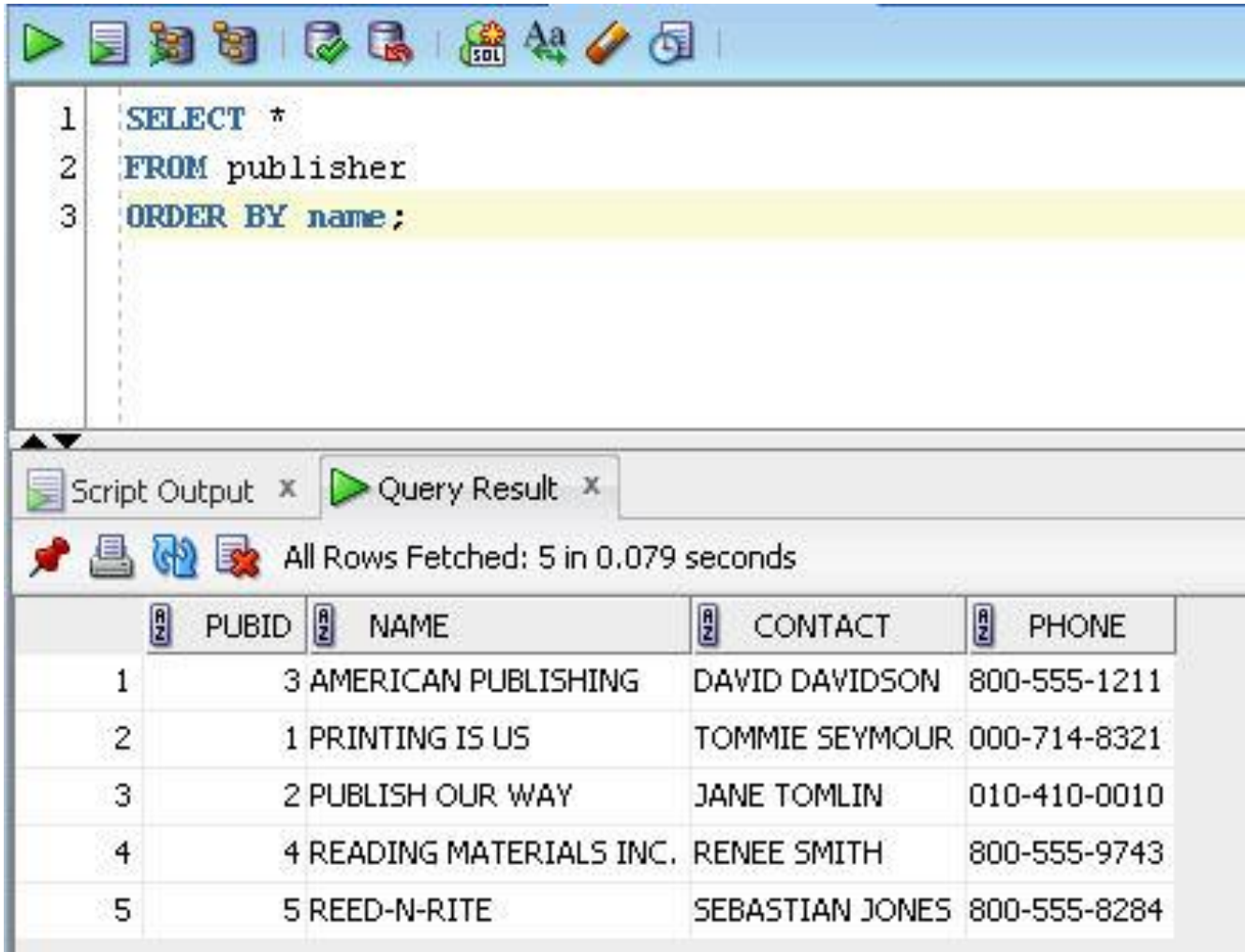
# ORDER BY Clause

```
SELECT [DISTINCT | UNIQUE] (*, columnname [ AS alias], ...)  
      FROM tablename  
      [WHERE condition]  
      [GROUP BY group_by_expression]  
      [HAVING group_condition]  
      [ORDER BY columnname];
```

# ORDER BY Clause

- When sorting in ascending order, values will be listed in this sequence:
  - Numeric values
  - Character values
  - NULL values
- If the order is not specified, **ASC** is the default

# ORDER BY Clause



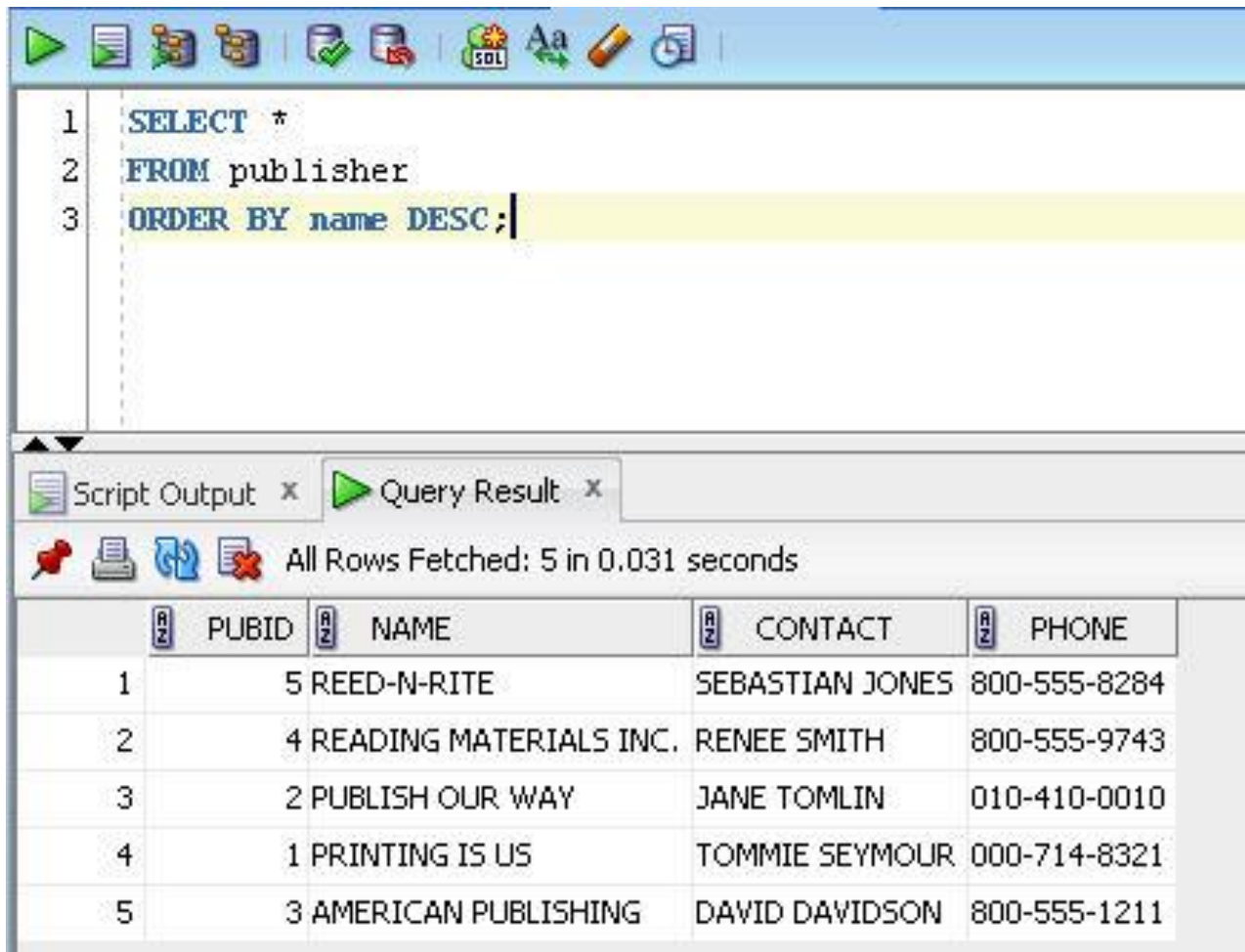
The screenshot shows a SQL query editor window. The query is as follows:

```
1 SELECT *
2 FROM publisher
3 ORDER BY name;
```

The third line of the query, `ORDER BY name;`, is highlighted in yellow. Below the query editor, there is a tabbed interface with two tabs: "Script Output" and "Query Result". The "Query Result" tab is active, showing the results of the query. Above the results table, it says "All Rows Fetched: 5 in 0.079 seconds".

	PUBID	NAME	CONTACT	PHONE
1	3	AMERICAN PUBLISHING	DAVID DAVIDSON	800-555-1211
2	1	PRINTING IS US	TOMMIE SEYMOUR	000-714-8321
3	2	PUBLISH OUR WAY	JANE TOMLIN	010-410-0010
4	4	READING MATERIALS INC.	RENEE SMITH	800-555-9743
5	5	REED-N-RITE	SEBASTIAN JONES	800-555-8284

# ORDER BY Clause



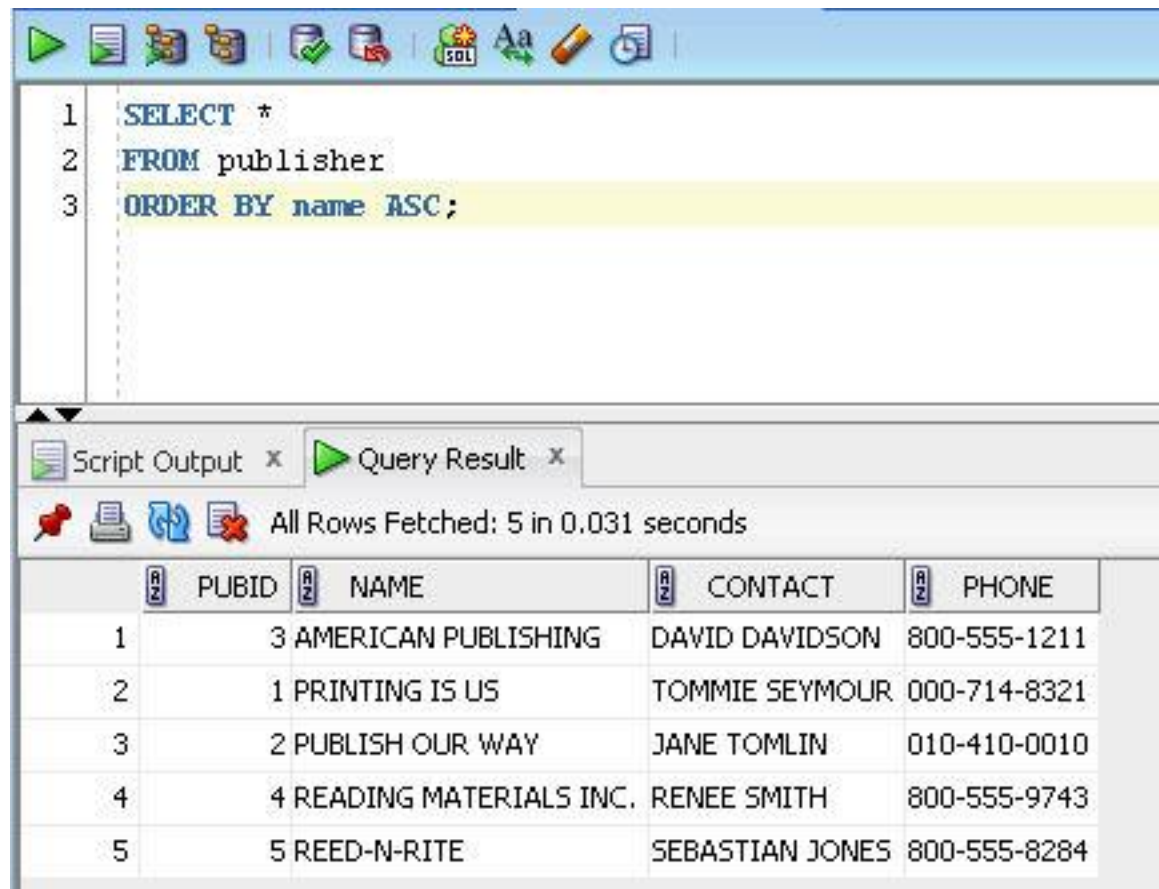
```
1 SELECT *
2 FROM publisher
3 ORDER BY name DESC;
```

Script Output x Query Result x

All Rows Fetched: 5 in 0.031 seconds

	PUBID	NAME	CONTACT	PHONE
1	5	REED-N-RITE	SEBASTIAN JONES	800-555-8284
2	4	READING MATERIALS INC.	RENEE SMITH	800-555-9743
3	2	PUBLISH OUR WAY	JANE TOMLIN	010-410-0010
4	1	PRINTING IS US	TOMMIE SEYMOUR	000-714-8321
5	3	AMERICAN PUBLISHING	DAVID DAVIDSON	800-555-1211

# ORDER BY Clause



The screenshot shows a database query tool interface. The top toolbar contains icons for running queries, saving, and other database functions. The main text area displays the following SQL query:

```
1 SELECT *  
2 FROM publisher  
3 ORDER BY name ASC;
```

Below the query editor, the 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 5 in 0.031 seconds'. The results are displayed in a table with the following columns: PUBID, NAME, CONTACT, and PHONE. The rows are ordered by NAME in ascending order.

	PUBID	NAME	CONTACT	PHONE
1	3	AMERICAN PUBLISHING	DAVID DAVIDSON	800-555-1211
2	1	PRINTING IS US	TOMMIE SEYMOUR	000-714-8321
3	2	PUBLISH OUR WAY	JANE TOMLIN	010-410-0010
4	4	READING MATERIALS INC.	RENEE SMITH	800-555-9743
5	5	REED-N-RITE	SEBASTIAN JONES	800-555-8284

It is common to use the ASC keyword with the ORDER BY clause to eliminate any possible confusion



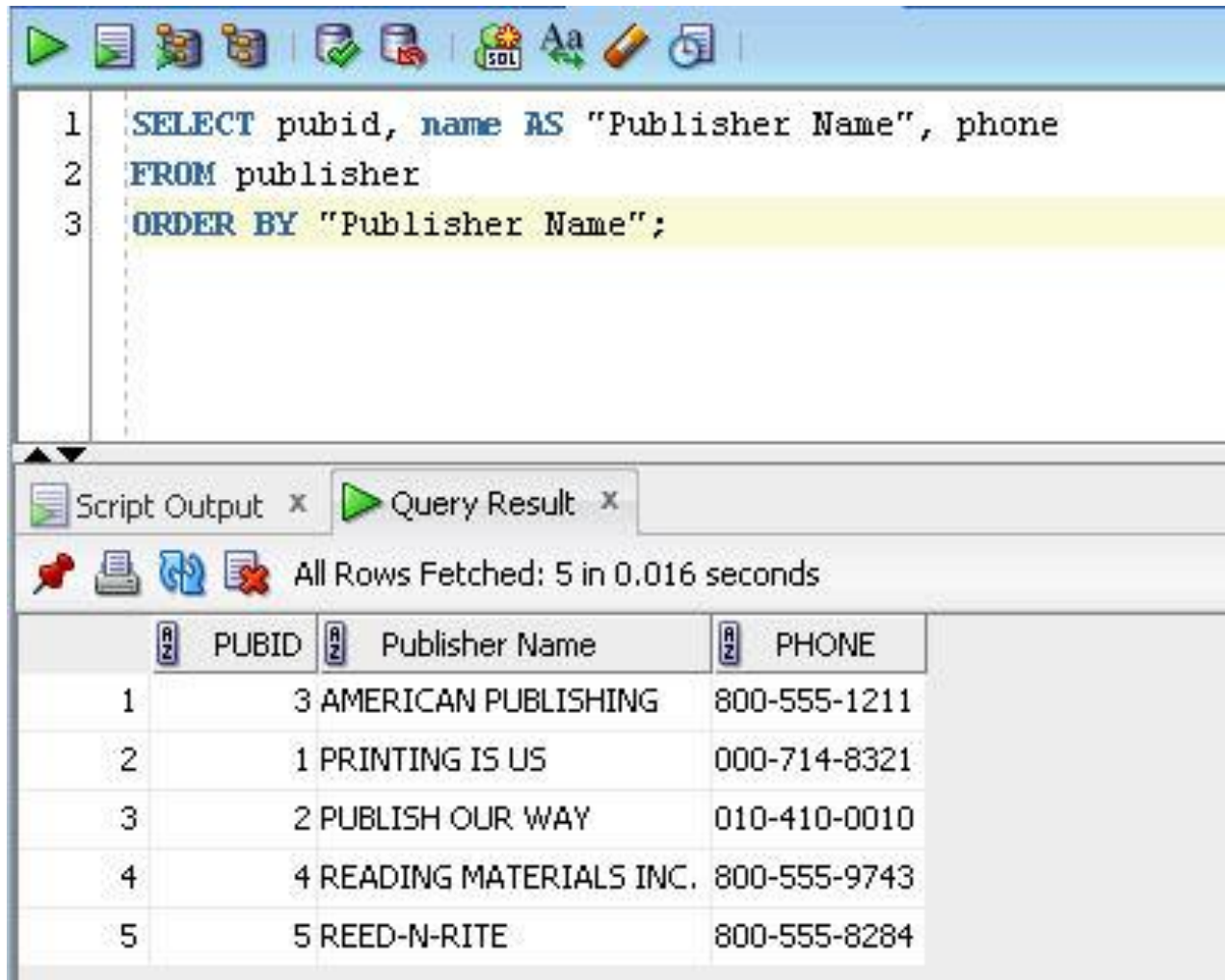
# ORDER BY Clause

- When sorting is ascending order, values are listed in this order:
  1. Blank and special characters
  2. Numeric values
  3. Character values (uppercase first)
  4. NULL values

# ORDER BY Clause

- If a column alias is given to a field in the SELECT clause, and that field is also used in the ORDER BY clause, you can use the column alias in the ORDER BY clause, although it is not required
- This is shown on the next two slides

# ORDER BY Clause



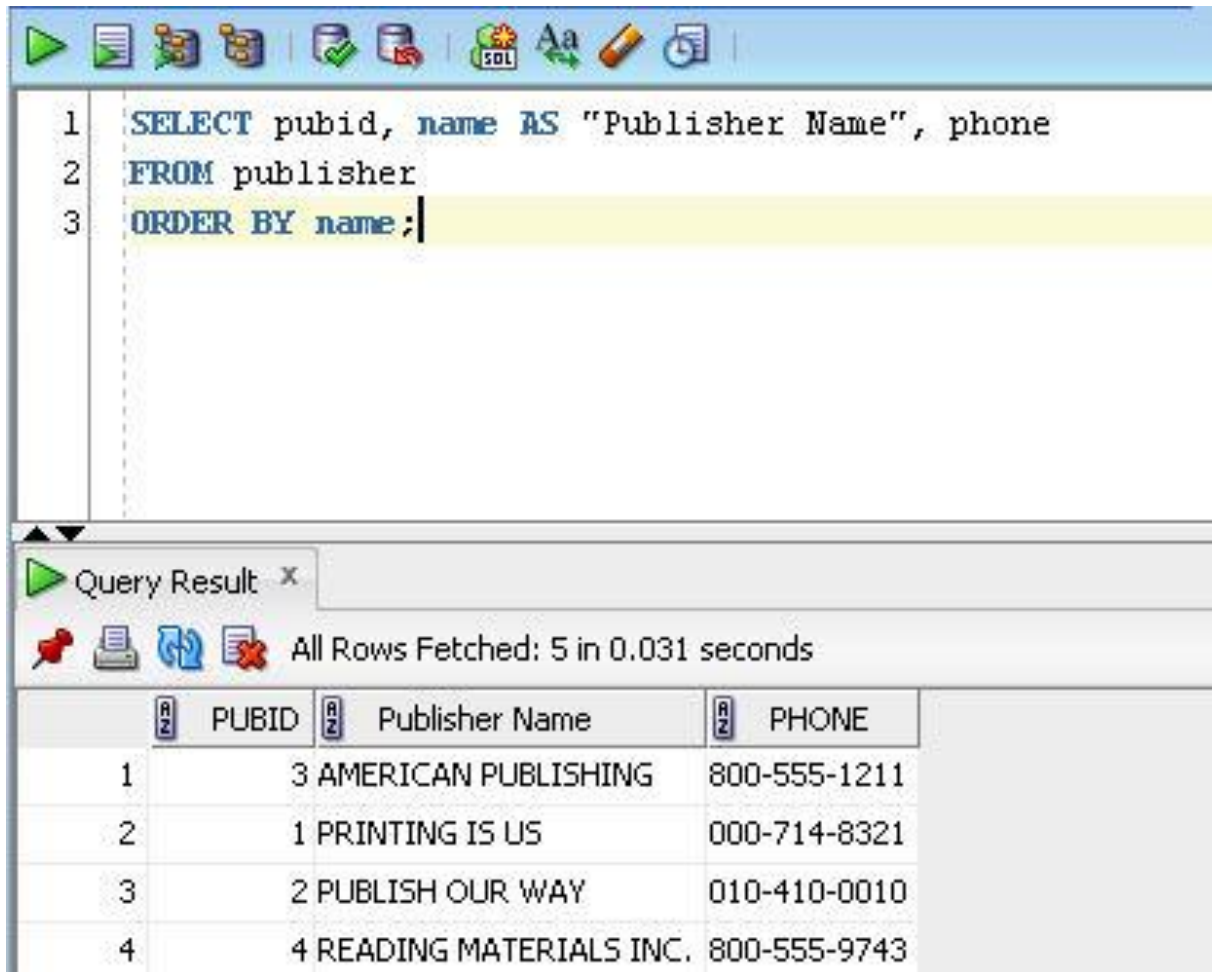
The screenshot shows a SQL query editor window. The query is as follows:

```
1 SELECT pubid, name AS "Publisher Name", phone
2 FROM publisher
3 ORDER BY "Publisher Name";
```

The query results are displayed in a table below the editor. The table has three columns: PUBID, Publisher Name, and PHONE. The results are ordered by the Publisher Name in ascending order.

	PUBID	Publisher Name	PHONE
1	3	AMERICAN PUBLISHING	800-555-1211
2	1	PRINTING IS US	000-714-8321
3	2	PUBLISH OUR WAY	010-410-0010
4	4	READING MATERIALS INC.	800-555-9743
5	5	REED-N-RITE	800-555-8284

# ORDER BY Clause



The screenshot shows a SQL query editor window with a toolbar at the top. The query text is as follows:

```
1 SELECT pubid, name AS "Publisher Name", phone
2 FROM publisher
3 ORDER BY name;
```

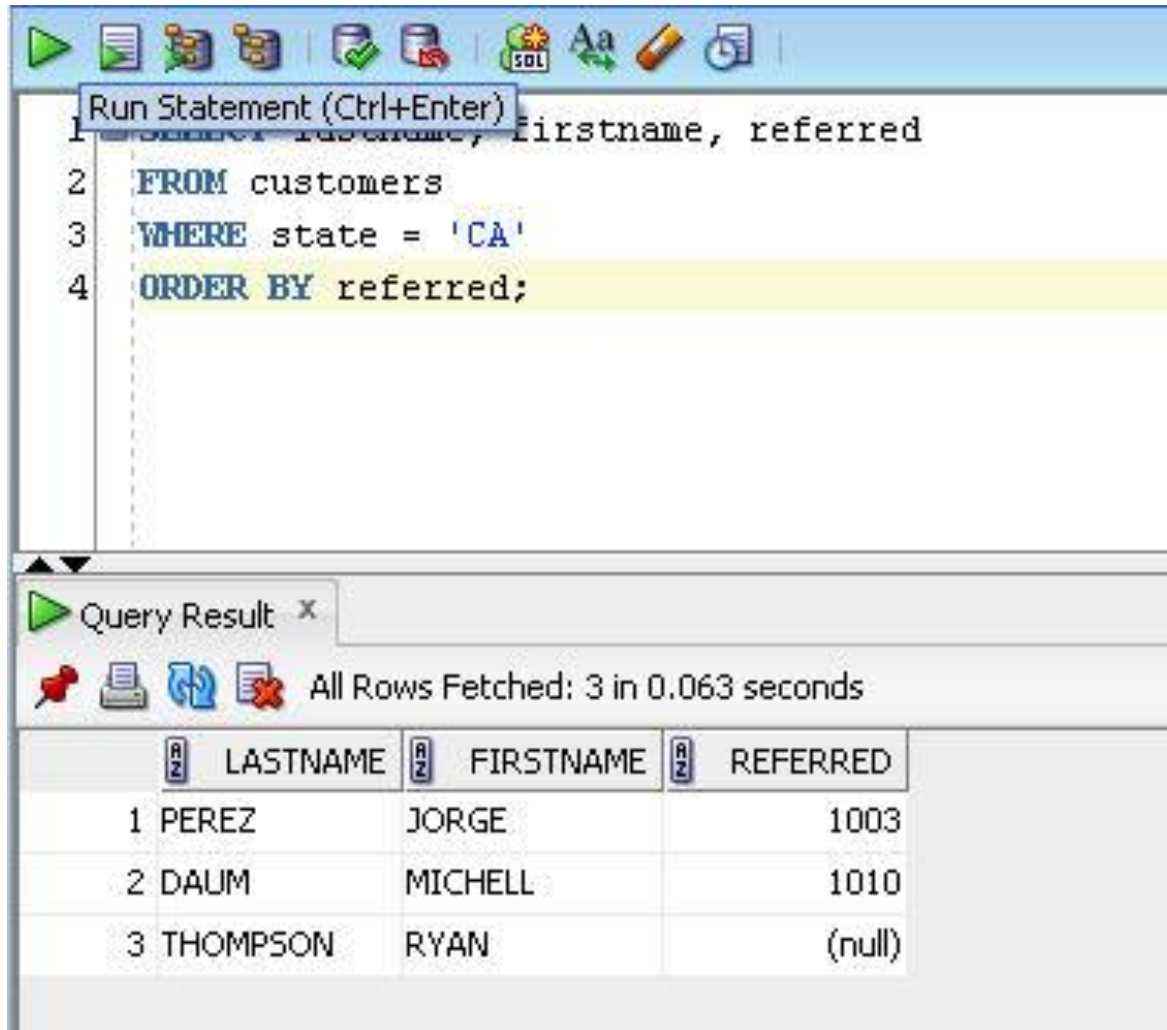
Below the query editor is a "Query Result" window. It displays the status "All Rows Fetched: 5 in 0.031 seconds". The results are shown in a table with four columns: PUBID, Publisher Name, and PHONE. The table contains four rows of data, sorted by Publisher Name in ascending order.

	PUBID	Publisher Name	PHONE
1	3	AMERICAN PUBLISHING	800-555-1211
2	1	PRINTING IS US	000-714-8321
3	2	PUBLISH OUR WAY	010-410-0010
4	4	READING MATERIALS INC.	800-555-9743

# ORDER BY Clause

- The **ORDER BY** clause can be used with the optional **NULLS FIRST** or **NULLS LAST** keywords to change the order in which the **NULL** values will be listed
- By default, nulls are listed last when the results are sorted in ascending order, and first when sorted in descending order

# ORDER BY Clause



The screenshot shows a SQL IDE window. The top toolbar includes icons for running queries, saving, and editing. A tooltip for the 'Run Statement (Ctrl+Enter)' button is visible. The SQL editor contains the following query:

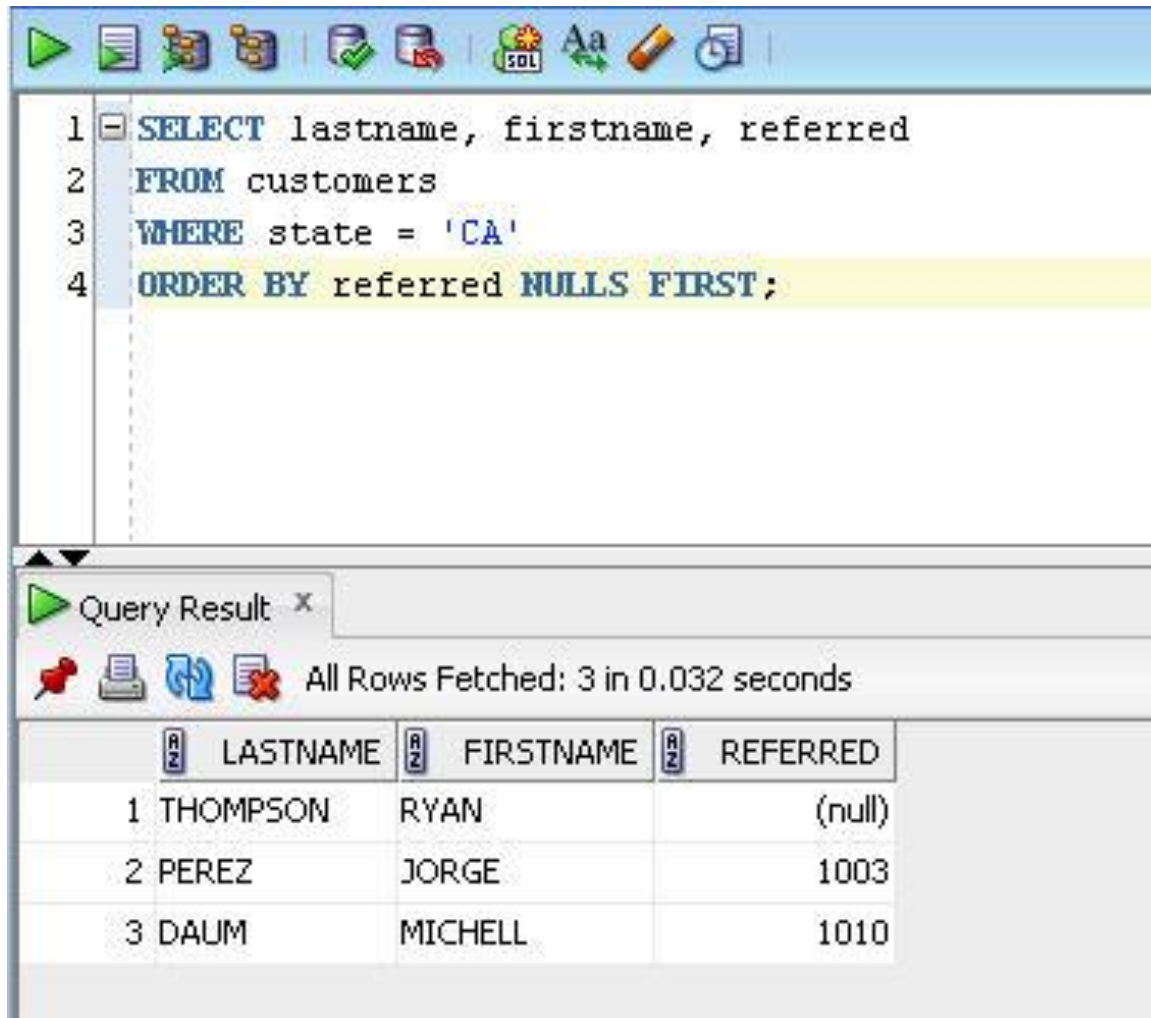
```
1 SELECT Lastname, Firstname, referred
2 FROM customers
3 WHERE state = 'CA'
4 ORDER BY referred;
```

The fourth line of the query is highlighted in yellow. Below the editor, the 'Query Result' tab is active, showing the results of the query. It indicates that all rows were fetched in 0.063 seconds. The results are displayed in a table with three columns: LASTNAME, FIRSTNAME, and REFERRED.

	LASTNAME	FIRSTNAME	REFERRED
1	PEREZ	JORGE	1003
2	DAUM	MICHELL	1010
3	THOMPSON	RYAN	(null)

The NULL value is listed last in the output for REFERRED, this is the default

# ORDER BY Clause



The screenshot shows a SQL query editor window with a toolbar at the top. The query text is as follows:

```
1 SELECT lastname, firstname, referred
2 FROM customers
3 WHERE state = 'CA'
4 ORDER BY referred NULLS FIRST;
```

The fourth line of the query is highlighted in yellow. Below the query editor is a 'Query Result' window. It displays the status 'All Rows Fetched: 3 in 0.032 seconds'. The results are shown in a table with three columns: LASTNAME, FIRSTNAME, and REFERRED. The rows are ordered by the REFERRED column, with NULL values appearing first.

	LASTNAME	FIRSTNAME	REFERRED
1	THOMPSON	RYAN	(null)
2	PEREZ	JORGE	1003
3	DAUM	MICHELL	1010

The NULL value is listed first in the output for REFERRED, when NULLS FIRST is specified

# ORDER BY Clause - Secondary

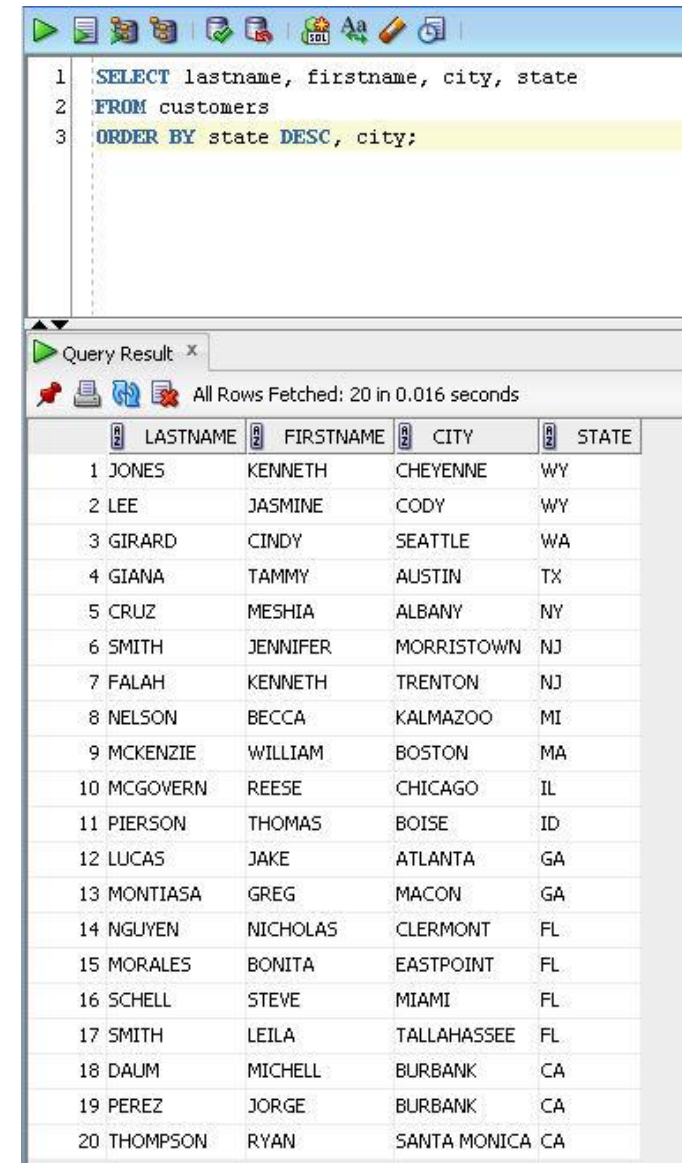
- When you only specify one clause in the **ORDER BY**, it is referred to as a **primary sort**
- In some cases, you may also want to include a secondary sort
- A **secondary sort** provides an alternative field to use if an exact match occurs between two or more rows in the primary sort
- The limit on the number of columns that can be used in the **ORDER BY** clause is **255**



# ORDER BY Clause – Secondary Sort

Notice the output, first of all look at the STATE column, it is sorted in descending sequence, so WYOMING (WY) is listed first

Now look at the CITY column, it is sorted in ascending sequence within each state, so CHEYENNE is listed before CODY for WYOMING (WY)



```
1 SELECT lastname, firstname, city, state
2 FROM customers
3 ORDER BY state DESC, city;
```

Query Result x

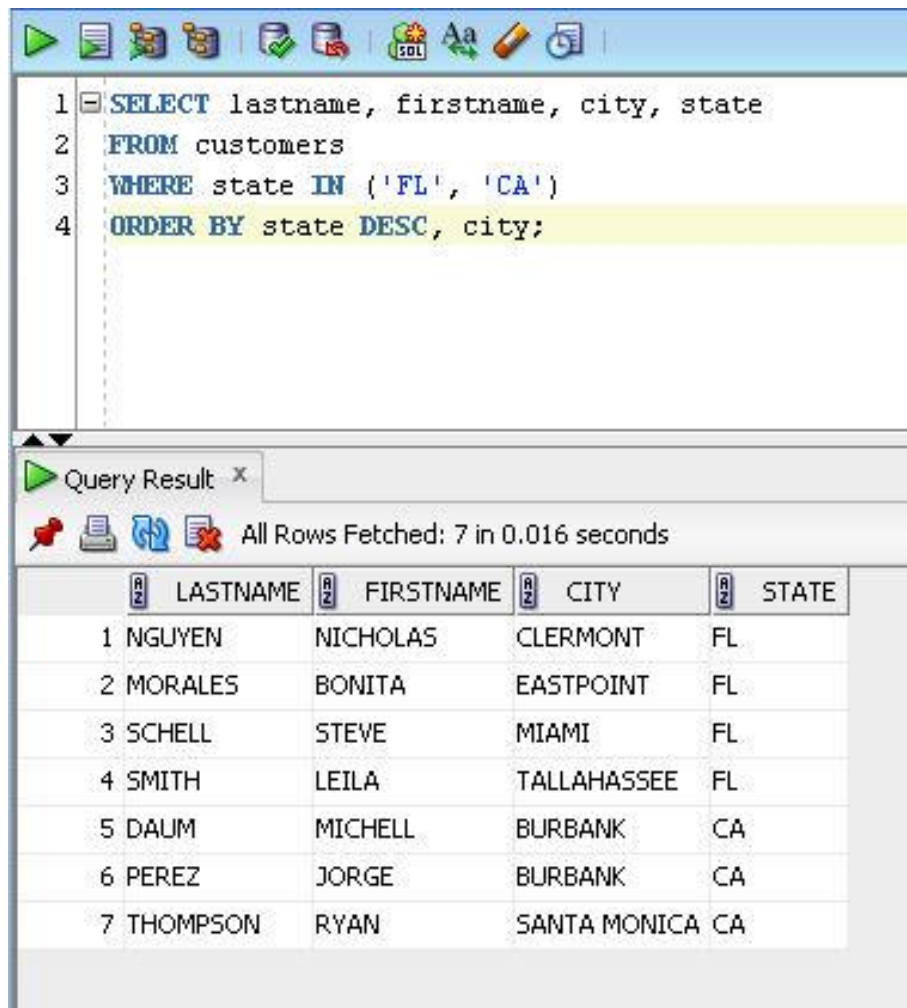
All Rows Fetched: 20 in 0.016 seconds

	LASTNAME	FIRSTNAME	CITY	STATE
1	JONES	KENNETH	CHEYENNE	WY
2	LEE	JASMINE	CODY	WY
3	GIRARD	CINDY	SEATTLE	WA
4	GIANA	TAMMY	AUSTIN	TX
5	CRUZ	MESHIA	ALBANY	NY
6	SMITH	JENNIFER	MORRISTOWN	NJ
7	FALAH	KENNETH	TRENTON	NJ
8	NELSON	BECCA	KALMAZOO	MI
9	MCKENZIE	WILLIAM	BOSTON	MA
10	MCGOVERN	REESE	CHICAGO	IL
11	PIERSON	THOMAS	BOISE	ID
12	LUCAS	JAKE	ATLANTA	GA
13	MONTIASA	GREG	MACON	GA
14	NGUYEN	NICHOLAS	CLERMONT	FL
15	MORALES	BONITA	EASTPOINT	FL
16	SHELL	STEVE	MIAMI	FL
17	SMITH	LEILA	TALLAHASSEE	FL
18	DAUM	MICHELL	BURBANK	CA
19	PEREZ	JORGE	BURBANK	CA
20	THOMPSON	RYAN	SANTA MONICA	CA

# Sorting By SELECT Order

- Another alternative for specifying the sort order is to use the selected column number rather than the column name
- This is also referred to as **ordering by column ordinals**

# Sorting By SELECT Order



```
1 SELECT lastname, firstname, city, state
2 FROM customers
3 WHERE state IN ('FL', 'CA')
4 ORDER BY state DESC, city;
```

Query Result x

All Rows Fetched: 7 in 0.016 seconds

	LASTNAME	FIRSTNAME	CITY	STATE
1	NGUYEN	NICHOLAS	CLERMONT	FL
2	MORALES	BONITA	EASTPOINT	FL
3	SHELL	STEVE	MIAMI	FL
4	SMITH	LEILA	TALLAHASSEE	FL
5	DAUM	MICHELL	BURBANK	CA
6	PEREZ	JORGE	BURBANK	CA
7	THOMPSON	RYAN	SANTA MONICA	CA

Another example showing a WHERE clause to restrict the rows being displayed to FL and CA

Again STATE is in descending sequence and CITY is in ascending sequence

This is the standard method we have used so far

# Sorting By SELECT Order

```
1 SELECT lastname, firstname, city, state
2 FROM customers
3 WHERE state IN ('FL', 'CA')
4 ORDER BY 4 DESC, 3;
```

Query Result x

All Rows Fetched: 7 in 0.015 seconds

	A Z	LASTNAME	A Z	FIRSTNAME	A Z	CITY	A Z	STATE
1		NGUYEN		NICHOLAS		CLERMONT		FL
2		MORALES		BONITA		EASTPOINT		FL
3		SHELL		STEVE		MIAMI		FL
4		SMITH		LEILA		TALLAHASSEE		FL
5		DAUM		MICHELL		BURBANK		CA
6		PEREZ		JORGE		BURBANK		CA
7		THOMPSON		RYAN		SANTA MONICA		CA

Another example showing the WHERE clause to restrict the rows being displayed to FL and CA

Again STATE is in descending sequence and CITY is in ascending sequence

This method is using the column ordinals so the sort is being done by the fourth column listed in the SELECT (STATE) and then the secondary sort on the third column listed in the SELECT (CITY)