

In this exercise you will retrieve data that is stored in the database system by using a SELECT statement.

Using the SELECT * statement show all data stored in the following tables:

- Worksheet Query Builder

SELECT* FROM customers;

Query Result x

All Rows Fetched: 6 in 0.146 seconds

	CTR_NUMBER	EMAIL	FIRST_NAME	LAST_NAME	PHONE_NUMBER	CURRENT_BALANCE	SRE_ID	TEM_ID	LOYALTY_CARD_NUMBER
1	c00001	bob.thornberry@heatmail.com	Robert	Thornberry	01234567898	150	sr01	t001	(null)
2	c00012	Jjones@freemail.com	Jennifer	Jones	01505214598	0	(null)	(null)	1c1015
3	c00101	unknown@here.com	John	Doe	03216547808	987.5	sr01	t002	(null)
4	c00103	MurciaA@globaltech.com	Andrew	Murcia	07715246890	85	(null)	(null)	1c2341
5	c01986	margal87@delphiview.com	Maria	Galant	01442736589	125.65	sr03	t003	(null)
6	c02001	brianrog@hootech.com	Brian	Rogers	01654564898	50	(null)	(null)	1c4587

2. teams.

Worksheet Query Builder

```
SELECT* FROM teams;
```

Query Result x

SQL | All Rows Fetched: 3 in 0.008 seconds

	ID	NAME	NUMBER_OF_PLAYERS	DISCOUNT
1	t001	Rockets	25	10
2	t002	Celtics	42	20
3	t003	Rovers	8	(null)

3. items

Worksheet

Query Builder

SELECT* FROM items;

Query Result x

SQL | All Rows Fetched: 5 in 0.006 seconds

	ITEM_NUMBER	NAME	DESCRIPTION	CATEGORY	COLOR	Size	ILT_ID
1	im01101044	gloves	catcher mitt	clothing	brown	m	i1010230124
2	im01101045	under shirt	top worn under the game top	clothing	white	s	i1010230125
3	im01101046	socks	team socks with emblem	clothing	range	l	i1010230126
4	im01101047	game top	team shirt with emblem	clothing	range	m	i1010230127
5	im01101048	premium bat	high quaity baseball bat	equipment	(null)	(null)	i1010230128

Part 2: Selecting Specific Columns

1. Display the customer number, first name, last name, email and phone number of the customers.

Worksheet

Query Builder

```
SELECT ctr_number, first_name, last_name, email, phone_number
FROM customers;
```

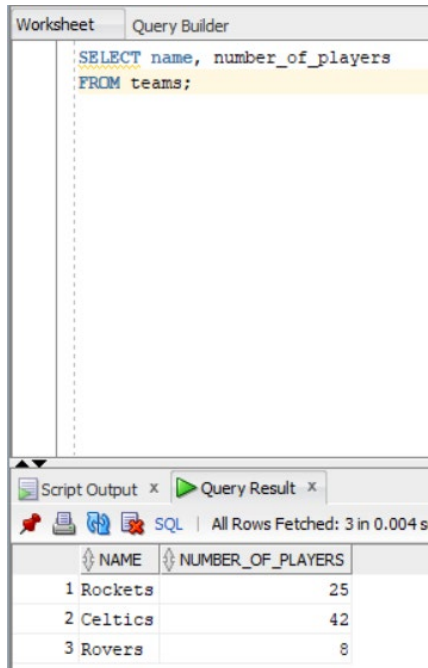
Script Output x

Query Result x

SQL | All Rows Fetched: 6 in 0.003 seconds

CTR_NUMBER	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER
1 c00001	Robert	Thornberry	bob.thornberry@heatmail.com	01234567898
2 c00012	Jennifer	Jones	Jjones@freemail.com	01505214598
3 c00101	John	Doe	unknown@here.com	03216547808
4 c00103	Andrew	Murcia	MurciaA@globaltech.com	07715246890
5 c01986	Maria	Galant	margal87@delphiview.com	01442736589
6 c02001	Brian	Rogers	brianrog@hootech.com	01654564898

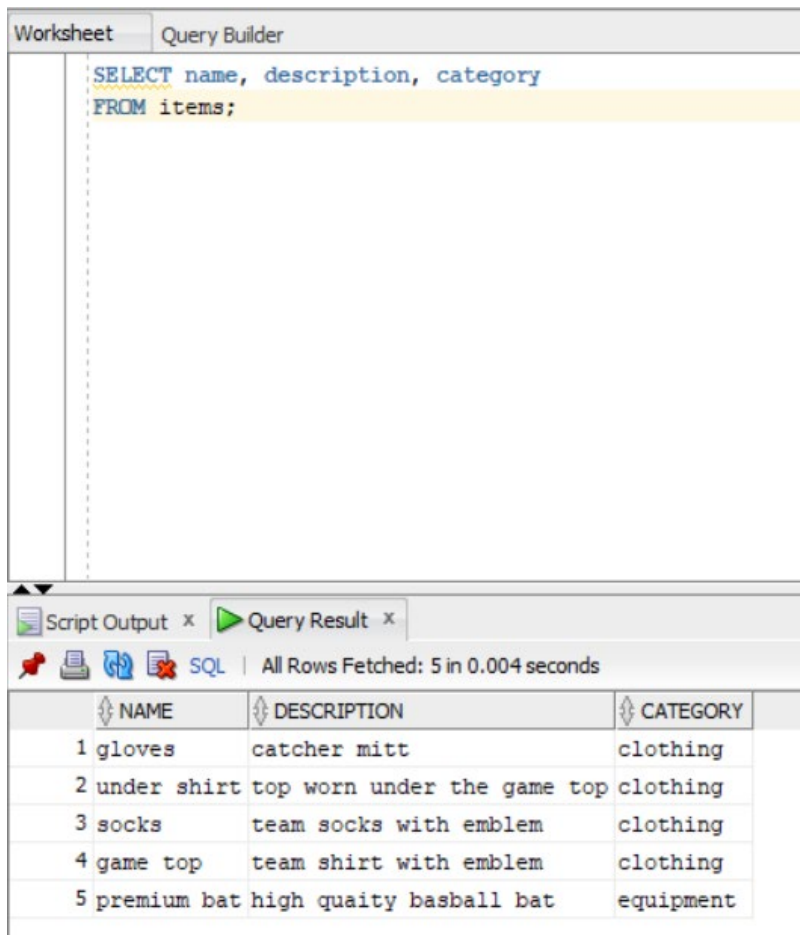
2. Display the name and number of players for each team.



The screenshot shows a 'Query Builder' window with a 'Worksheet' tab. The SQL query entered is: `SELECT name, number_of_players
FROM teams;`. Below the query, the 'Query Result' tab is active, displaying a table with 3 rows and 2 columns: 'NAME' and 'NUMBER_OF_PLAYERS'. The status bar indicates 'All Rows Fetched: 3 in 0.004 s'.

	NAME	NUMBER_OF_PLAYERS
1	Rockets	25
2	Celtics	42
3	Rovers	8

3. Display the name, description and category for every item in the table.



The screenshot shows a 'Query Builder' window with a 'Worksheet' tab. The SQL query entered is: `SELECT name, description, category
FROM items;`. Below the query, the 'Query Result' tab is active, displaying a table with 5 rows and 3 columns: 'NAME', 'DESCRIPTION', and 'CATEGORY'. The status bar indicates 'All Rows Fetched: 5 in 0.004 seconds'.

	NAME	DESCRIPTION	CATEGORY
1	gloves	catcher mitt	clothing
2	under shirt	top worn under the game top	clothing
3	socks	team socks with emblem	clothing
4	game top	team shirt with emblem	clothing
5	premium bat	high quaity baseball bat	equipment

Write and Execute SELECT statements (S6L6 Objective 2)

Part 1: Using Arithmetic Operators

- [illegible]

- ```
SELECT first_name, last_name, ctr_number, current_balance, current_balance - 5.00
FROM customers;
```



[illegible]

## Part 3: Using Literal Character Strings

1. Write a query that will display the team information in the following format:

The Rockets team has 25 players and receives a discount of 10 percent.

Use **Team Information** as the column alias.

```
SELECT 'The ' || name || ' team has ' || number_of_players || ' players and receives a discount of ' ||
discount || ' percent '
```

## AS "Team Information"

FROM teams;


Worksheet

Query Builder

```
SELECT 'The ' || name || ' team has ' || number_of_players || ' players and receives a discount of ' || discount || ' percent '
AS "Team Information"
FROM teams;
```

Script Output x

Query Result x

 All Rows Fetched: 3 in 0.04 seconds

| Team Information                                                        |
|-------------------------------------------------------------------------|
| 1 The Rockets team has 25 players and receives a discount of 10 percent |
| 2 The Celtics team has 42 players and receives a discount of 20 percent |
| 3 The Rovers team has 8 players and receives a discount of percent      |

- Why does the last team not show a discount?

The last team discount is a null value.

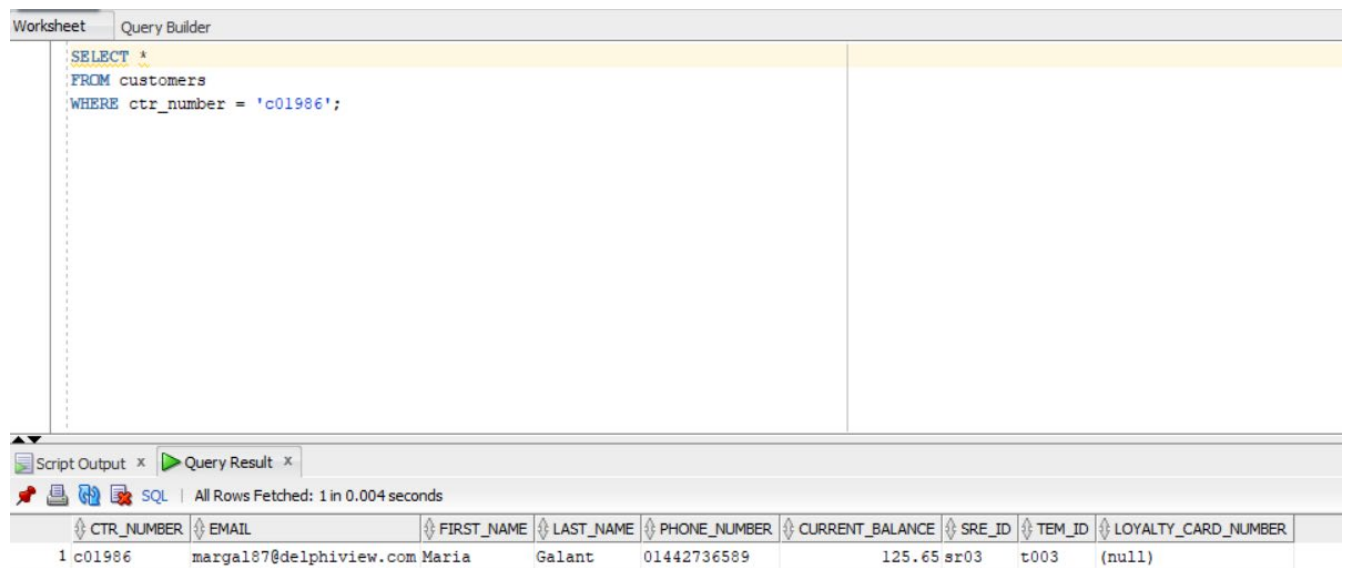
## Section 6 Lesson 7 Exercise 1: Restricting Data Using WHERE

### Limit rows using WHERE (S6L7 Objective 1)

In this exercise you will refine the data that is returned in your query by adding a WHERE clause to your SELECT statement.

#### Part 1: Using the WHERE Clause.

- Using the unique customer number in the where clause display all columns for Maria Galant.



The screenshot shows a SQL query builder interface. The top section is labeled 'Worksheet' and 'Query Builder'. The query text is as follows:

```
SELECT *
FROM customers
WHERE ctr_number = 'c01986';
```

The bottom section shows the 'Query Result' tab. It indicates 'All Rows Fetched: 1 in 0.004 seconds'. Below this is a table with the following columns and data:

| CTR_NUMBER | EMAIL                   | FIRST_NAME | LAST_NAME | PHONE_NUMBER | CURRENT_BALANCE | SRE_ID | TEM_ID | LOYALTY_CARD_NUMBER |
|------------|-------------------------|------------|-----------|--------------|-----------------|--------|--------|---------------------|
| 1 c01986   | margal87@delphiview.com | Maria      | Galant    | 01442736589  | 125.65 sr03     | t003   | (null) |                     |

- Display the first name, last name and customer number for all customers who have a current balance of greater than 100. Use an appropriate alias for your column headings.



Worksheet    Query Builder

```
SELECT first_name "First Name", last_name "Last Name", ctr_number "Customer Number"
FROM customers
WHERE current_balance > 100;
```

Script Output x    Query Result x

SQL | All Rows Fetched: 3 in 0.001 seconds

|   | First Name | Last Name  | Customer Number |
|---|------------|------------|-----------------|
| 1 | Robert     | Thornberry | c00001          |
| 2 | John       | Doe        | c00101          |
| 3 | Maria      | Galant     | c01986          |

3. Display the order id, date and time of all orders that were placed before the 28<sup>th</sup> of May 2019. Use an appropriate alias for your column headings.

Worksheet    Query Builder

```
SELECT id "Order ID", odr_date "Order Date", odr_time "Order Time"
FROM orders
WHERE odr_date < '28-MAY-2019';
```

Script Output x    Query Result x

SQL | All Rows Fetched: 5 in 0.001 seconds

|   | Order ID  | Order Date | Order Time |
|---|-----------|------------|------------|
| 1 | or0101250 | 17/04/2017 | 17/04/2017 |
| 2 | or0101350 | 24/05/2017 | 24/05/2017 |
| 3 | or0101425 | 28/05/2017 | 28/05/2017 |
| 4 | or0101681 | 02/06/2017 | 02/06/2017 |
| 5 | or0101750 | 18/06/2017 | 18/06/2017 |

## Part 2: Range Conditions: BETWEEN Operator

1. Display the inventory id, cost and number of units using appropriate aliases for all items that have a trade cost of between 3.00 and 15.00.

The screenshot shows a database query builder interface. At the top, there are tabs for 'Worksheet' and 'Query Builder'. The 'Query Builder' tab is active, displaying a SQL query in a text area:

```
SELECT id "Inventory ID", cost "Cost", units "Number of units"
FROM inventory_list
WHERE cost BETWEEN 3.00 AND 15.00;
```

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: 2 in 0.01 seconds'. The results are displayed in a table with three columns: 'Inventory ID', 'Cost', and 'Number of units'.

|   | Inventory ID | Cost | Number of units |
|---|--------------|------|-----------------|
| 1 | i1010230125  | 7.99 | 250             |
| 2 | i1010230126  | 5.24 | 87              |

## Part 3: Membership Conditions: IN Operator

1. Display the inventory id, cost and number of units using appropriate aliases for all items that have 50, 100, 150 or 200 units in stock.

Worksheet Query Builder

```
SELECT id "Inventory ID", cost "Cost", units "Number of units"
FROM inventory_list
WHERE units IN (50,100,150,200);
```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.003 seconds

|   | Inventory ID | Cost | Number of units |
|---|--------------|------|-----------------|
| 1 | i1010230124  | 2.5  | 100             |

#### Part 4: Membership Conditions: NOT IN Operator

1. Display the inventory id, cost and number of units using appropriate aliases for all items that do not have 50, 100, 150 or 200 units in stock.

Worksheet Query Builder

```
SELECT id "Inventory ID", cost "Cost", units "Number of units"
FROM inventory_list
WHERE units NOT IN (50,100,150,200);
```

Script Output x Query Result x

SQL | All Rows Fetched: 4 in 0.001 seconds

|   | Inventory ID | Cost  | Number of units |
|---|--------------|-------|-----------------|
| 1 | i1010230125  | 7.99  | 250             |
| 2 | i1010230126  | 5.24  | 87              |
| 3 | i1010230127  | 18.95 | 65              |
| 4 | i1010230128  | 97.46 | 8               |

## Part 5: Pattern Matching: LIKE Operator

1. Display item number and name of all items that have a name that begins with g. Use an appropriate alias for your column headings.

The screenshot shows a database query builder interface. At the top, there are two tabs: "Worksheet" and "Query Builder". The "Query Builder" tab is active, displaying a SQL query in a text area. The query is:

```
SELECT itm_number "Item number", name "Name of item"
FROM items
WHERE name LIKE 'g%';
```

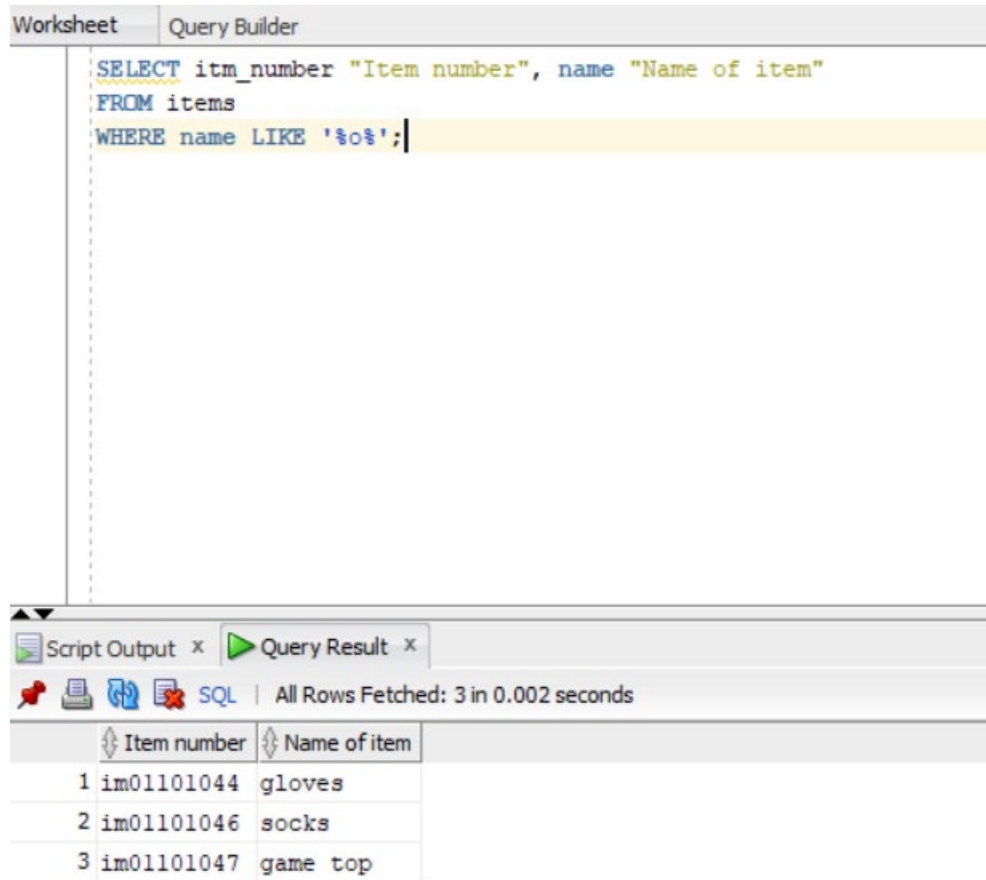
Below the query editor, there is a section for the query results. It includes a "Script Output" tab and a "Query Result" tab. The "Query Result" tab is active, showing a table with two columns: "Item number" and "Name of item". The table contains two rows of data:

|   | Item number | Name of item |
|---|-------------|--------------|
| 1 | im01101044  | gloves       |
| 2 | im01101047  | game top     |

At the bottom of the interface, there is a status bar that reads "All Rows Fetched: 2 in 0.003 seconds".

## Part 6 : Pattern Matching: Combining Wildcard Characters with the LIKE Operator

1. Display item number and name of all items that have a name that contain a lowercase o. Use an appropriate alias for your column headings.



The screenshot shows a SQL Query Builder window with a query editor and a results pane. The query editor contains the following SQL statement:

```
SELECT itm_number "Item number", name "Name of item"
FROM items
WHERE name LIKE '%o%';
```

The results pane shows the output of the query, displaying 3 rows of data. The columns are labeled "Item number" and "Name of item".

|   | Item number | Name of item |
|---|-------------|--------------|
| 1 | im01101044  | gloves       |
| 2 | im01101046  | socks        |
| 3 | im01101047  | game top     |

## Section 6 Lesson 7 Exercise 2: Restricting Data Using WHERE

### Limit rows using WHERE (S6L7 Objective 1)

In this exercise you will refine the data that is returned in your query by adding a WHERE clause to your SELECT statement.

#### Part 1: Using the NULL Conditions

1. Write a query that will display information for teams that don't receive a discount in the following format:

The Rovers team has 25 players and does not receive a discount.

Use **Team Information** as the column alias.

```
SELECT 'The ' || name || ' ' || 'team has ' || number_of_players || ' players and does not receive a discount'
AS "Team Information"
FROM teams
WHERE discount IS NULL;
```

2. Write a query that will display information for only teams that receive a discount in the following format:

The Rockets team has 25 players and receives a discount of 10 percent.

Use **Team Information** as the column alias.

```
SELECT 'The ' || name || ' ' || 'team has ' || number_of_players || ' players and receives a discount of 10 percent' AS "Team Information"
FROM teams
WHERE discount = '10';
```

## Part 2: Logical Operators: AND

1. Write a query that will display the customer number, address line 1 and postal code for customers that live in the starford area of Liverpool. Use Customer Number, Street Address and Postal Code as the column aliases.

```
SELECT id "Customer Number", address_line_1 "Street Address", zip_code "Postal Code"
FROM customers_addresses
WHERE address_line_2 = 'Starford'
AND city = 'Liverpool';
```

## Part 3: Logical Operators: OR

1. Write a query that will display the customer number, address line 1 and postal code for customers that live in either starford or Liverpool in general. Use Customer Number, Street Address and Postal Code as the column aliases.

```
SELECT id "Customer Number", address_line_1 "Street Address", zip_code "Postal Code"
FROM customers_addresses
WHERE address_line_2 = 'Starford'
OR city = 'Liverpool';
```

## Part 4: Logical Operators: NOT Equal To

1. Write a query that will display the customer number, address line 1 and postal code for customers that do not live in Liverpool. Use Customer Number, Street Address and Postal Code as the column aliases.

```
SELECT id "Customer Number", address_line_1 "Street Address", zip_code "Postal Code"
FROM customers_addresses
WHERE city <> 'Liverpool';
```

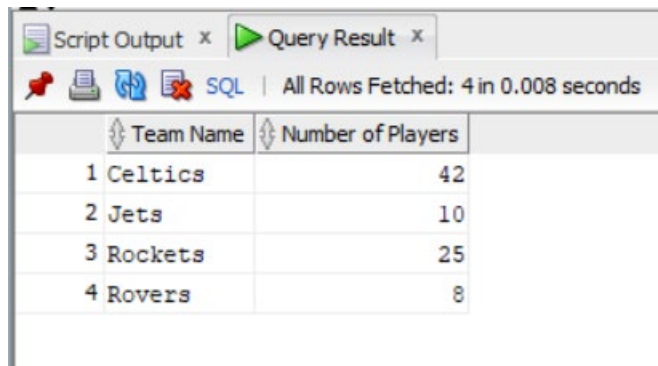
## Section 6 Lesson 8 Exercise 1: Sorting Data Using ORDER BY

### Use the ORDER BY Clause to Sort SQL Results (S6L8 Objective 1)

In this exercise you will sort the order of the data that is returned in your query by adding an ORDER BY clause to the end of your SELECT statement.

1. Display the team name and number of players alphabetically in order of team name. Use an appropriate alias for your column headings.

```
SELECT name "Team Name", number_of_players "Number of Players"
FROM teams
ORDER BY name;
```

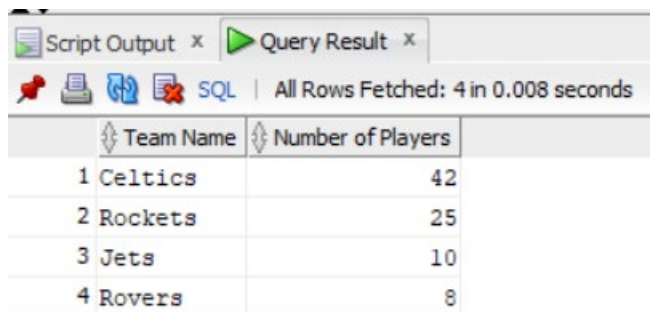


The screenshot shows a SQL query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with two columns: 'Team Name' and 'Number of Players'. The table contains four rows of data, sorted alphabetically by team name. The status bar at the top indicates 'All Rows Fetched: 4 in 0.008 seconds'.

|   | Team Name | Number of Players |
|---|-----------|-------------------|
| 1 | Celtics   | 42                |
| 2 | Jets      | 10                |
| 3 | Rockets   | 25                |
| 4 | Rovers    | 8                 |

2. Display the team name and number of players in descending order of number of players. Use an appropriate alias for your column headings.

```
SELECT name "Team Name", number_of_players "Number of Players"
FROM teams
ORDER BY number_of_players DESC;
```



The screenshot shows a SQL query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with two columns: 'Team Name' and 'Number of Players'. The table contains four rows of data, sorted in descending order of the number of players. The status bar at the top indicates 'All Rows Fetched: 4 in 0.008 seconds'.

|   | Team Name | Number of Players |
|---|-----------|-------------------|
| 1 | Celtics   | 42                |
| 2 | Rockets   | 25                |
| 3 | Jets      | 10                |
| 4 | Rovers    | 8                 |

3. Display the team name and number of players alphabetically in order of team name. Use Team Name for the name alias and Players for the number of players. Sort the output in descending order of name using the alias in the ORDER BY clause.

```
SELECT name "Team Name", number_of_players "Players"
FROM teams
ORDER BY "Team Name" DESC;
```

Script Output x

Query Result x

SQL

All Rows Fetched: 4 in 0.003 seconds

|   | Team Name | Players |
|---|-----------|---------|
| 1 | Rovers    | 8       |
| 2 | Rockets   | 25      |
| 3 | Jets      | 10      |
| 4 | Celtics   | 42      |

## Section 6 Lesson 8 Exercise 2: Sorting Data Using ORDER BY

### Part 1 : TOP-N-ANALYSIS (S6L8 Objective 3)

- The customers are numbered sequentially with each new customer being assigned a higher customer number. Use TOP-N-ANALYSIS to only show the First and last name of the first three customers. Show the customers first and last name in the same column using Customer Name as the column alias.

```
SELECT ROWNUM AS "Latest Customers", first_name || ' ' || last_name AS "Customer Name"
FROM (SELECT first_name, last_name
 FROM customers)
WHERE ROWNUM <=3;
```

Script Output x

Query Result x

SQL | All Rows Fetched: 3 in 0.055 seconds

|   | Latest Customers | Customer Name     |
|---|------------------|-------------------|
| 1 | 1                | Robert Thornberry |
| 2 | 2                | John Doe          |
| 3 | 3                | Andrew Murcia     |

### Part 2 : Using a Substitution Variable (S6L8 Objective 4)

- Use a substitution variable that will allow you to enter the commission rate for the sales representatives. The first and last names should be displayed to screen for any sales representatives that earn that commission rate and the output should be ordered by their last name. Use an appropriate alias for your column headings.

```
SELECT first_name "First Name", last_name "Last Name"
FROM sales_representatives
WHERE commission_rate = :commission_rate
ORDER BY last_name;
```



Enter Binds

commission\_rate

Name: commission\_rate

☐ NULL

Value: 10

Help Apply Cancel

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.003 seconds

|   | First Name | Last Name |
|---|------------|-----------|
| 1 | Charles    | Raymond   |