

## Lab 4

Name : Khoo Teong Lee

Matric No : A22EC0174

Section : 08

### SQL4-DML 3 PART 1

#### Section 6 Lesson 9 Exercise 1: Joining Tables Using JOIN

#### Write SELECT Statements Using Data From Multiple Tables Using Equijoins and Non-Equijoins (S6L9 Objective 1)


In this exercise you will write SELECT statements to access data from more than one table.

##### Part 1: Creating Natural Joins.

1. Display all of the information about sales representatives and their addresses using a natural join.

Ans :

```
SELECT id, email, first_name, last_name, phone_number, commission_rate, supervisor_id,
address_line_1, address_line_2, city, zip_code
FROM sales_representatives NATURAL JOIN sales_rep_addresses;
```



The screenshot shows a SQL Developer window with a query editor and a results grid. The query editor contains the following SQL statement:

```
1 SELECT id, email, first_name, last_name, phone_number, commission_rate, supervisor_id, address_line_1, address_line_2, city, zip_code
2 FROM sales_representatives NATURAL JOIN sales_rep_addresses;
3
```

The results grid displays the following data:

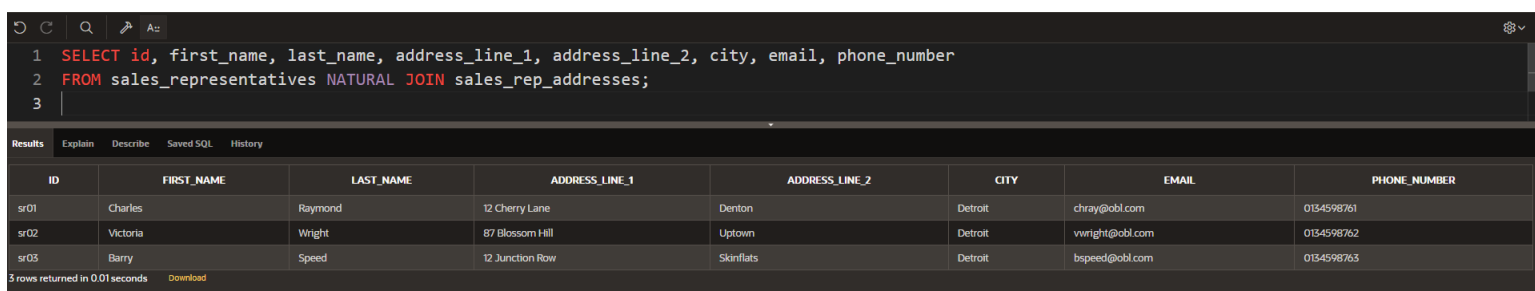
ID	EMAIL	FIRST_NAME	LAST_NAME	PHONE_NUMBER	COMMISSION_RATE	SUPERVISOR_ID	ADDRESS_LINE_1	ADDRESS_LINE_2	CITY	ZIP_CODE
sr01	chray@obl.com	Charles	Raymond	0134598761	10	sr01	12 Cherry Lane	Denton	Detroit	DT48211
sr02	vwright@obl.com	Victoria	Wright	0134598762	5	sr01	87 Blossom Hill	Uptown	Detroit	DT52314
sr03	bspeed@obl.com	Barry	Speed	0134598763	5	sr01	12 Junction Row	Skinflats	Detroit	DT52564

3 rows returned in 0.01 seconds

2. Adapt the query from the previous question to only show the id, first name, last name, address line 1, address line 2, city, email and phone\_number for the sales representatives.

Ans :

```
SELECT id, first_name, last_name, address_line_1, address_line_2, city, email, phone_number
FROM sales_representatives NATURAL JOIN sales_rep_addresses;
```



The screenshot shows a SQL Developer window with a query editor and a results grid. The query editor contains the following SQL statement:

```
1 SELECT id, first_name, last_name, address_line_1, address_line_2, city, email, phone_number
2 FROM sales_representatives NATURAL JOIN sales_rep_addresses;
3
```

The results grid displays the following data:

ID	FIRST_NAME	LAST_NAME	ADDRESS_LINE_1	ADDRESS_LINE_2	CITY	EMAIL	PHONE_NUMBER
sr01	Charles	Raymond	12 Cherry Lane	Denton	Detroit	chray@obl.com	0134598761
sr02	Victoria	Wright	87 Blossom Hill	Uptown	Detroit	vwright@obl.com	0134598762
sr03	Barry	Speed	12 Junction Row	Skinflats	Detroit	bspeed@obl.com	0134598763

3 rows returned in 0.01 seconds

## Part 2: Creating Joins with the USING Clause

1. Adapt the previous query answer to use the USING clause instead of a natural join.

Ans :

```
SELECT id, first_name, last_name, address_line_1, address_line_2, city, email, phone_number
FROM sales_representatives JOIN sales_rep_addresses
USING (id);
```



The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```
1 SELECT id, first_name, last_name, address_line_1, address_line_2, city, email, phone_number
2 FROM sales_representatives JOIN sales_rep_addresses
3 USING (id);
4
```

The results pane shows the following table:

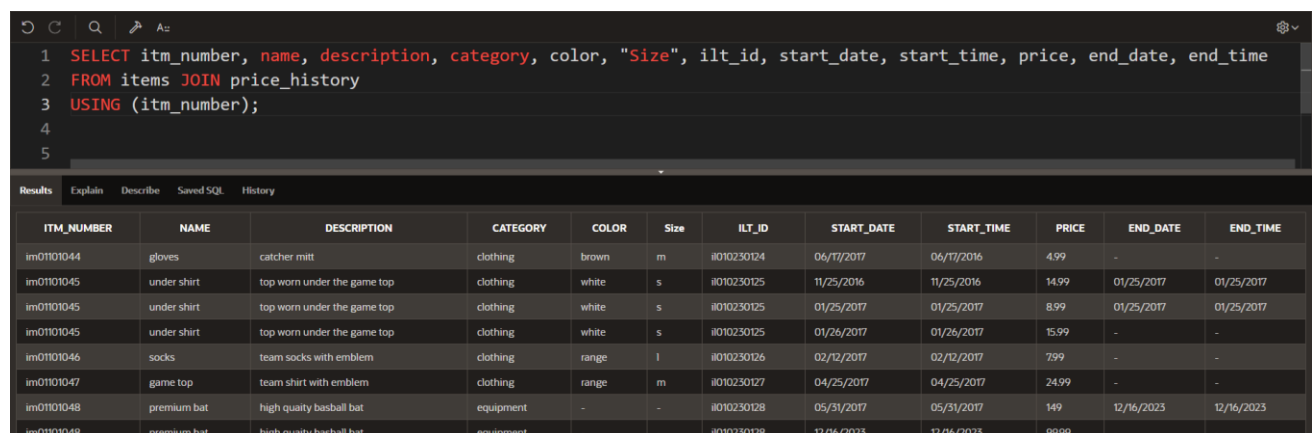
ID	FIRST_NAME	LAST_NAME	ADDRESS_LINE_1	ADDRESS_LINE_2	CITY	EMAIL	PHONE_NUMBER
sr01	Charles	Raymond	12 Cherry Lane	Denton	Detroit	chray@obl.com	0134598761
sr02	Victoria	Wright	87 Blossom Hill	Uptown	Detroit	vwright@obl.com	0134598762
sr03	Barry	Speed	12 Junction Row	Skinflats	Detroit	bspeed@obl.com	0134598763

5 rows returned in 0.02 seconds

2. Display all of the information about items and their price history by joining the items and price\_history tables.

Ans :

```
SELECT itm_number, name, description, category, color, "Size", ilt_id, start_date, start_time, price, end_date,
end_time
FROM items JOIN price_history
USING (itm_number);
```



The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```
1 SELECT itm_number, name, description, category, color, "Size", ilt_id, start_date, start_time, price, end_date, end_time
2 FROM items JOIN price_history
3 USING (itm_number);
4
5
```

The results pane shows the following table:

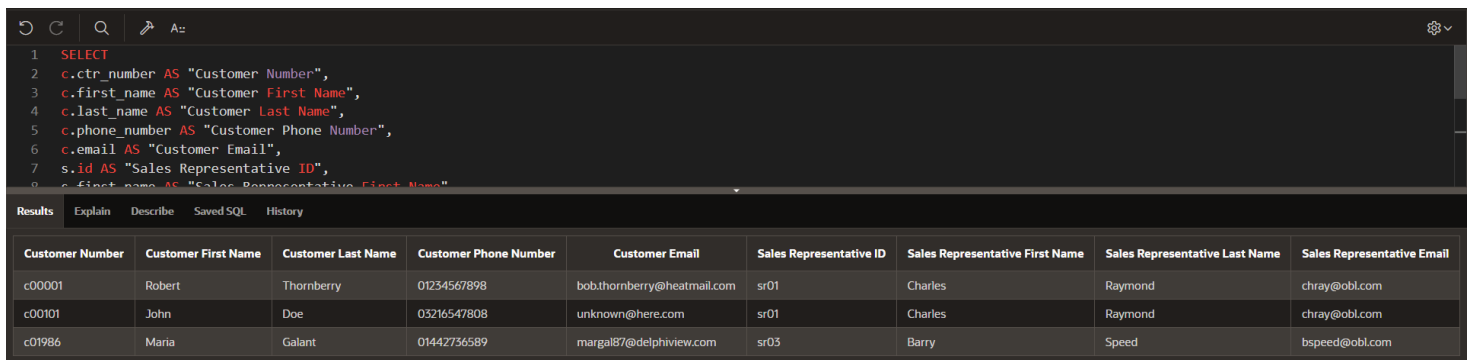
ITEM_NUMBER	NAME	DESCRIPTION	CATEGORY	COLOR	Size	ILT_ID	START_DATE	START_TIME	PRICE	END_DATE	END_TIME
im0101044	gloves	catcher mitt	clothing	brown	m	il010230124	06/11/2017	06/11/2016	4.99	-	-
im0101045	under shirt	top worn under the game top	clothing	white	s	il010230125	11/25/2016	11/25/2016	14.99	01/25/2017	01/25/2017
im0101045	under shirt	top worn under the game top	clothing	white	s	il010230125	01/25/2017	01/25/2017	8.99	01/25/2017	01/25/2017
im0101045	under shirt	top worn under the game top	clothing	white	s	il010230125	01/26/2017	01/26/2017	15.99	-	-
im0101046	socks	team socks with emblem	clothing	range	l	il010230126	02/12/2017	02/12/2017	7.99	-	-
im0101047	game top	team shirt with emblem	clothing	range	m	il010230127	04/25/2017	04/25/2017	24.99	-	-
im0101048	premium bat	high quality baseball bat	equipment	-	-	il010230128	05/31/2017	05/31/2017	149	12/16/2023	12/16/2023
im0101048	premium bat	high quality baseball bat	equipment	-	-	il010230128	12/16/2023	12/16/2023	99.99	-	-

### Part 3: Creating Joins with the ON Clause

1. Use an ON clause to join the customer and sales representative table so that you display the customer number, customer first name, customer last name, customer phone number, customer email, sales representative id, sales representative first name, sales representative last name and sales representative email. You will need to use a table alias in your answer as both tables have columns with the same name.

Ans :

```
SELECT
    c.ctr_number AS "Customer Number",
    c.first_name AS "Customer First Name",
    c.last_name AS "Customer Last Name",
    c.phone_number AS "Customer Phone Number",
    c.email AS "Customer Email",
    s.id AS "Sales Representative ID",
    s.first_name AS "Sales Representative First Name",
    s.last_name AS "Sales Representative Last Name",
    s.email AS "Sales Representative Email"
FROM customers c JOIN sales_representatives s
ON s.id = c.sre_id;
```



The screenshot shows a SQL IDE interface. The top panel contains a SQL query that joins the 'customers' and 'sales\_representatives' tables. The bottom panel, titled 'Results', displays the query output as a table with 9 columns and 3 rows. The columns are: Customer Number, Customer First Name, Customer Last Name, Customer Phone Number, Customer Email, Sales Representative ID, Sales Representative First Name, Sales Representative Last Name, and Sales Representative Email. The three rows represent different customer-representative pairs.

Customer Number	Customer First Name	Customer Last Name	Customer Phone Number	Customer Email	Sales Representative ID	Sales Representative First Name	Sales Representative Last Name	Sales Representative Email
c00001	Robert	Thornberry	01234567898	bob.thornberry@heatmail.com	sr01	Charles	Raymond	chray@obl.com
c00101	John	Doe	03216547808	unknown@here.com	sr01	Charles	Raymond	chray@obl.com
c01986	Maria	Galant	01442736589	margal87@delphiview.com	sr03	Barry	Speed	bspeed@obl.com

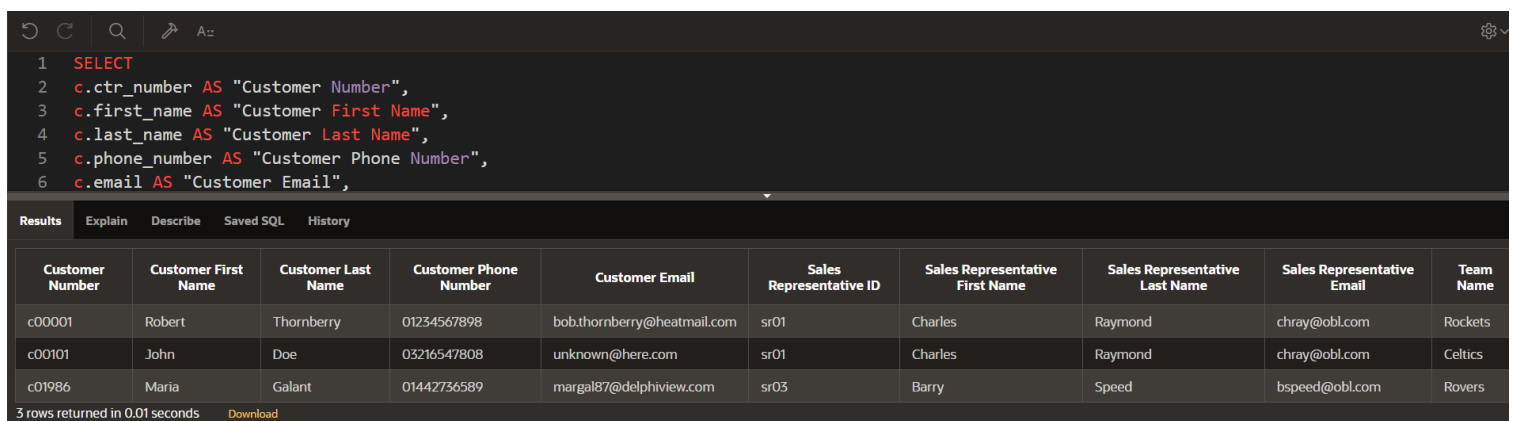
3 rows returned in 0.01 seconds

## Part 4- Creating Three-Way Joins with the ON Clause

1. Using the answer to Task 3 add a join that will allow the team name that the customer represents to be included in the results.

Ans :

```
SELECT
    c.ctr_number AS "Customer Number",
    c.first_name AS "Customer First Name",
    c.last_name AS "Customer Last Name",
    c.phone_number AS "Customer Phone Number",
    c.email AS "Customer Email",
    s.id AS "Sales Representative ID",
    s.first_name AS "Sales Representative First Name",
    s.last_name AS "Sales Representative Last Name",
    s.email AS "Sales Representative Email",
    t.name AS "Team Name"
FROM customers c JOIN sales_representatives s
ON c.sre_id=s.id
JOIN teams t
ON c.tem_id = t.id ;
```



The screenshot shows a SQL IDE interface. The top panel contains a SQL query. The bottom panel shows the results of the query, which is a table with 10 columns and 3 rows. The table is titled 'Results' and has tabs for 'Explain', 'Describe', 'Saved SQL', and 'History'. The query is a three-way join between the 'customers', 'sales\_representatives', and 'teams' tables. The results show three rows of data, each representing a customer, their sales representative, and the team they belong to.

Customer Number	Customer First Name	Customer Last Name	Customer Phone Number	Customer Email	Sales Representative ID	Sales Representative First Name	Sales Representative Last Name	Sales Representative Email	Team Name
c00001	Robert	Thornberry	01234567898	bob.thornberry@heatmail.com	sr01	Charles	Raymond	chray@obl.com	Rockets
c00101	John	Doe	03216547808	unknown@here.com	sr01	Charles	Raymond	chray@obl.com	Celtics
c01986	Maria	Galant	01442736589	margal87@delphiview.com	sr03	Barry	Speed	bspeed@obl.com	Rovers

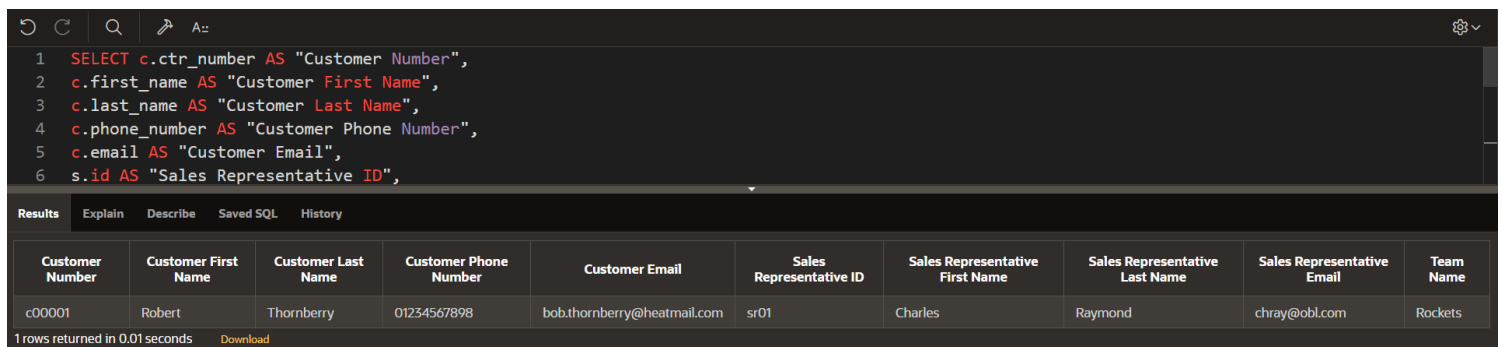
3 rows returned in 0.01 seconds [Download](#)

## Part 5: Applying Additional Conditions to a Join

1. Using the answer to Task 4 add an additional condition to only show the results for the customer that has the number - c00001.

Ans :

```
SELECT
    c.ctr_number AS "Customer Number",
    c.first_name AS "Customer First Name",
    c.last_name AS "Customer Last Name",
    c.phone_number AS "Customer Phone Number",
    c.email AS "Customer Email",
    s.id AS "Sales Representative ID",
    s.first_name AS "Sales Representative First Name",
    s.last_name AS "Sales Representative Last Name",
    s.email AS "Sales Representative Email",
    t.name AS "Team Name"
FROM customers c JOIN sales_representatives s
ON c.sre_id=s.id
JOIN teams t
ON c.tem_id = t.id
WHERE c.ctr_number = 'c00001';
```



The screenshot shows a SQL IDE interface. The top part displays the SQL query being executed. Below the query editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing a table with 10 columns: Customer Number, Customer First Name, Customer Last Name, Customer Phone Number, Customer Email, Sales Representative ID, Sales Representative First Name, Sales Representative Last Name, Sales Representative Email, and Team Name. The table contains one row of data for customer c00001. At the bottom left, it says '1 rows returned in 0.01 seconds' and there is a 'Download' button.

Customer Number	Customer First Name	Customer Last Name	Customer Phone Number	Customer Email	Sales Representative ID	Sales Representative First Name	Sales Representative Last Name	Sales Representative Email	Team Name
c00001	Robert	Thornberry	01234567898	bob.thornberry@heatmail.com	sr01	Charles	Raymond	chray@obl.com	Rockets

1 rows returned in 0.01 seconds [Download](#)

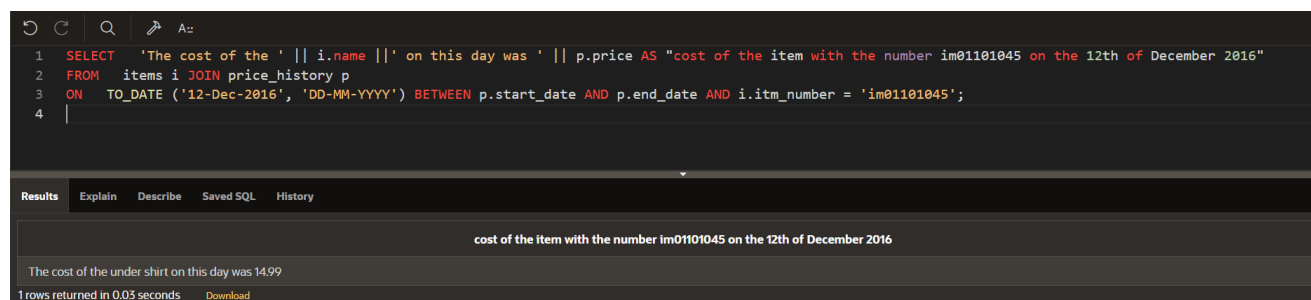
## Part 6: Retrieving Records with Nonequijoins

1. Write a query that will display name and cost of the item with the number im01101045 on the 12th of December 2016. The output of the query should look like this:

The cost of the under shirt on this day was 14.99

Ans :

```
SELECT 'The cost of the ' || i.name || ' on this day was ' || p.price AS "cost of the item with the number im01101045 on the 12th of December 2016"
FROM items i JOIN price_history p
ON TO_DATE ('12-Dec-2016', 'DD-MM-YYYY') BETWEEN p.start_date AND p.end_date AND i.itm_number = 'im01101045';
```



The screenshot shows a SQL query execution interface. The query is as follows:

```
1 SELECT 'The cost of the ' || i.name || ' on this day was ' || p.price AS "cost of the item with the number im01101045 on the 12th of December 2016"
2 FROM items i JOIN price_history p
3 ON TO_DATE ('12-Dec-2016', 'DD-MM-YYYY') BETWEEN p.start_date AND p.end_date AND i.itm_number = 'im01101045';
4
```

The interface includes tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, showing the following output:

cost of the item with the number im01101045 on the 12th of December 2016
The cost of the under shirt on this day was 14.99

At the bottom, it indicates "1 rows returned in 0.03 seconds" and provides a Download link.

## SQL4-DML 3 PART 2

### Section 6 Lesson 9 Exercise 2: Joining Tables Using JOIN

Write SELECT Statements Using Data From Multiple Tables Using Equijoins and Non-Equijoins (S6L9 Objective 1)

#### Part 1 : Use a Self-Join to Join a Table to Itself (S6L9 Objective 2)

1. Write a query that will display who the supervisor is for each of the sales representatives. The information should be displayed in two columns, the first column will be the first name and last name of the sales representative and the second will be the first name and last name of the supervisor. The column aliases should be Rep and Supervisor.

Ans :

```
SELECT
    c.first_name || ' ' || c.last_name AS "Rep",
    s.first_name || ' ' || s.last_name AS "Supervisor"
FROM customers c JOIN sales_representatives s
ON c.sre_id = s.supervisor_id;
```

The screenshot shows a SQL IDE interface. At the top, there's a 'SQL Commands' tab and a 'Schema' dropdown set to 'WKSP\_TEONGLEE'. Below this is a toolbar with 'Language' set to 'SQL', 'Rows' set to '10', and buttons for 'Clear Command', 'Find Tables', 'Save', and 'Run'. The main editor area contains the following SQL query:

```
1 SELECT
2 c.first_name || ' ' || c.last_name AS "Rep",
3 s.first_name || ' ' || s.last_name AS "Supervisor"
4 FROM customers c JOIN sales_representatives s
5 ON c.sre id = s.supervisor id;
```

Below the editor is a 'Results' tab with sub-tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' sub-tab is active, showing a table with two columns: 'Rep' and 'Supervisor'. The table contains 6 rows of data:

Rep	Supervisor
Robert Thornberry	Charles Raymond
Robert Thornberry	Victoria Wright
Robert Thornberry	Barry Speed
John Doe	Charles Raymond
John Doe	Victoria Wright
John Doe	Barry Speed

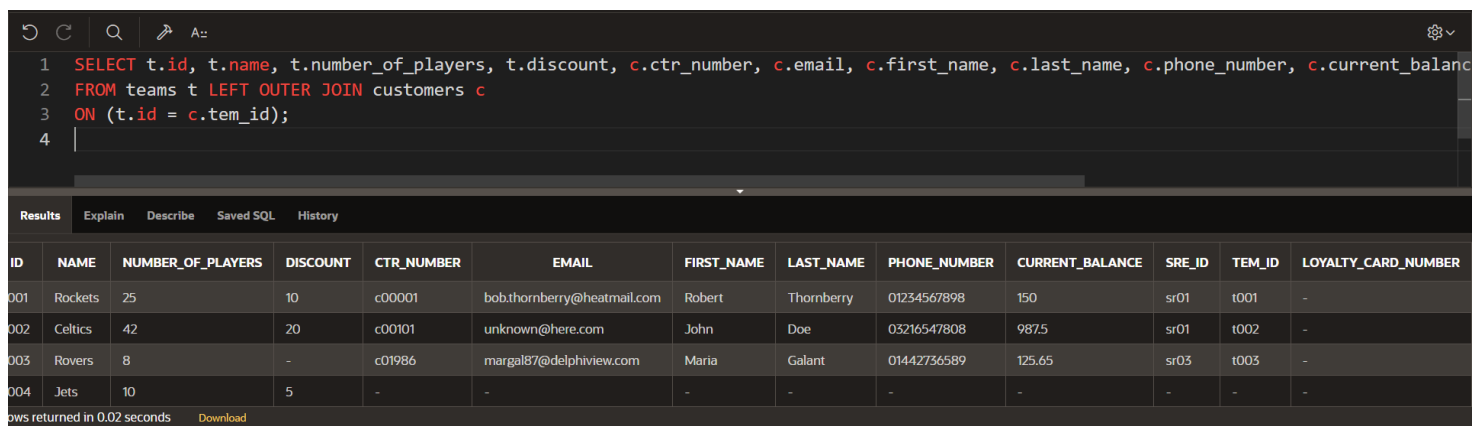
At the bottom, it says '6 rows returned in 0.02 seconds' and has a 'Download' button.

## Part 2 : Use OUTER joins (S6L9 Objective 3)

1. Write a query that will display all of the team and customer information even if there is no match with the table on the left (team).

Ans :

```
SELECT t.id, t.name, t.number_of_players, t.discount, c.ctr_number, c.email, c.first_name,
c.last_name, c.phone_number, c.current_balance, c.sre_id, c.tem_id, c.loyalty_card_number
FROM teams t LEFT OUTER JOIN customers c
ON (t.id = c.tem_id);
```



The screenshot shows a SQL query editor with a dark theme. The query is as follows:

```
1 SELECT t.id, t.name, t.number_of_players, t.discount, c.ctr_number, c.email, c.first_name, c.last_name, c.phone_number, c.current_balance
2 FROM teams t LEFT OUTER JOIN customers c
3 ON (t.id = c.tem_id);
4
```

Below the query editor, there is a 'Results' tab with sub-tabs for 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, displaying a table with 13 columns: ID, NAME, NUMBER\_OF\_PLAYERS, DISCOUNT, CTR\_NUMBER, EMAIL, FIRST\_NAME, LAST\_NAME, PHONE\_NUMBER, CURRENT\_BALANCE, SRE\_ID, TEM\_ID, and LOYALTY\_CARD\_NUMBER. The table contains 4 rows of data.

ID	NAME	NUMBER_OF_PLAYERS	DISCOUNT	CTR_NUMBER	EMAIL	FIRST_NAME	LAST_NAME	PHONE_NUMBER	CURRENT_BALANCE	SRE_ID	TEM_ID	LOYALTY_CARD_NUMBER
001	Rockets	25	10	c00001	bob.thornberry@heatmail.com	Robert	Thornberry	01234567898	150	sr01	t001	-
002	Celtics	42	20	c00101	unknown@here.com	John	Doe	05216547808	987.5	sr01	t002	-
003	Rovers	8	-	c01986	margal87@delphiview.com	Maria	Galant	01442736589	125.65	sr03	t003	-
004	Jets	10	5	-	-	-	-	-	-	-	-	-

At the bottom left, it says '4 rows returned in 0.02 seconds' and there is a 'Download' button.



### Part 3 : Generating a Cartesian Product (S6L9 Objective 4)

1. Create a Cartesian product between the customer and sales representative tables.

Ans :

```
SELECT c.ctr_number, c.email, c.first_name, c.last_name, c.phone_number, c.current_balance,
c.sre_id, c.tem_id, c.loyalty_card_number, s.id, s.email, s.first_name, s.last_name, s.phone_number,
s.commission_rate, s.supervisor_id
FROM customers c, sales_representatives s;
```

```
1 SELECT c.ctr_number, c.email, c.first_name, c.last_name, c.phone_number, c.current_balance, c.sre_id, c.tem_id, c.loyalty_card_number, s.id, s.email,
2 FROM customers c, sales_representatives s;
3
```

Results

Explain

Describe

Saved SQL

History

CTR_NUMBER	EMAIL	FIRST_NAME	LAST_NAME	PHONE_NUMBER	CURRENT_BALANCE	SRE_ID	TEM_ID	LOYALTY_CARD_NUMBER	ID	EMAIL	FIRST_NAME	LAST_NAME	PHONE_NUMBER	COMMISSION_RATE	SUPERVISOR_ID
c00001	bob.thornberry@heatmail.com	Robert	Thornberry	01234567898	150	sr01	t001	-	sr01	chray@obl.com	Charles	Raymond	0134598761	10	sr01
c00012	j.jones@freemail.com	Jennifer	Jones	01505214598	0	-	-	k1015	sr01	chray@obl.com	Charles	Raymond	0134598761	10	sr01
c00101	unknown@here.com	John	Doe	03216547808	987.5	sr01	t002	-	sr01	chray@obl.com	Charles	Raymond	0134598761	10	sr01
c00103	MurciaA@globaltech.com	Andrew	Murcia	07715246890	85	-	-	k2341	sr01	chray@obl.com	Charles	Raymond	0134598761	10	sr01
c01986	margalit@delphiview.com	Maria	Galant	01442736589	125.65	sr03	t003	-	sr01	chray@obl.com	Charles	Raymond	0134598761	10	sr01
c02001	brianrog@hootech.com	Brian	Rogers	01654564898	50	-	-	k4587	sr01	chray@obl.com	Charles	Raymond	0134598761	10	sr01
c00001	bob.thornberry@heatmail.com	Robert	Thornberry	01234567898	150	sr01	t001	-	sr02	vwright@obl.com	Victoria	Wright	0134598762	5	sr01
c00012	j.jones@freemail.com	Jennifer	Jones	01505214598	0	-	-	k1015	sr02	vwright@obl.com	Victoria	Wright	0134598762	5	sr01
c00101	unknown@here.com	John	Doe	03216547808	987.5	sr01	t002	-	sr02	vwright@obl.com	Victoria	Wright	0134598762	5	sr01
c00103	MurciaA@globaltech.com	Andrew	Murcia	07715246890	85	-	-	k2341	sr02	vwright@obl.com	Victoria	Wright	0134598762	5	sr01

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.02 secondsDownload