



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
UTM Johor Bahru

SECD2523

SECTION

10

LECTURER: ROZILAWATI BINTI DOLLAH @ MD ZAIN

PROJECT TITLE:

SQL LAB4

(DML3)

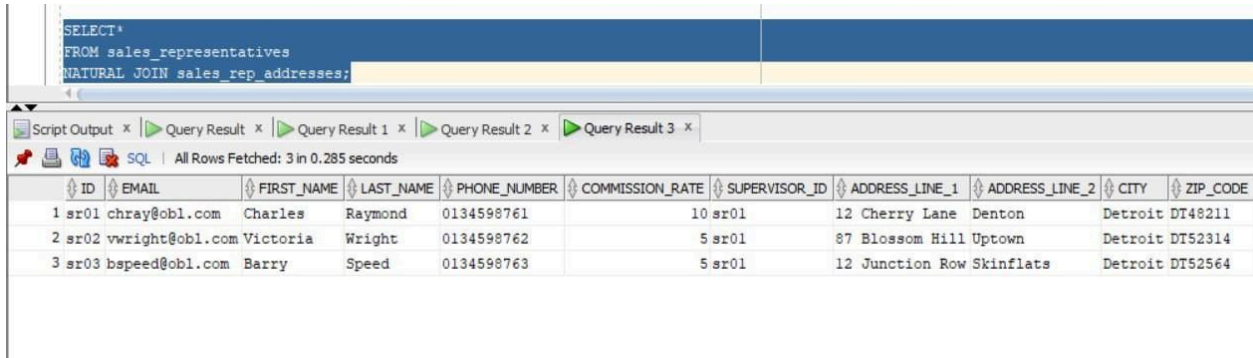
PART1

| NAME | MATRIC NUMBER |
|--|---------------|
| NURFAZRINA SYAKILA BINTI BAHARUDDIN | A21SC0276 |

Part 1: Creating Natural Joins.

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

```
SELECT*
FROM sales_representatives
NATURAL JOIN
sales_rep_addresses;
```



The screenshot shows a SQL query execution window with the following query:

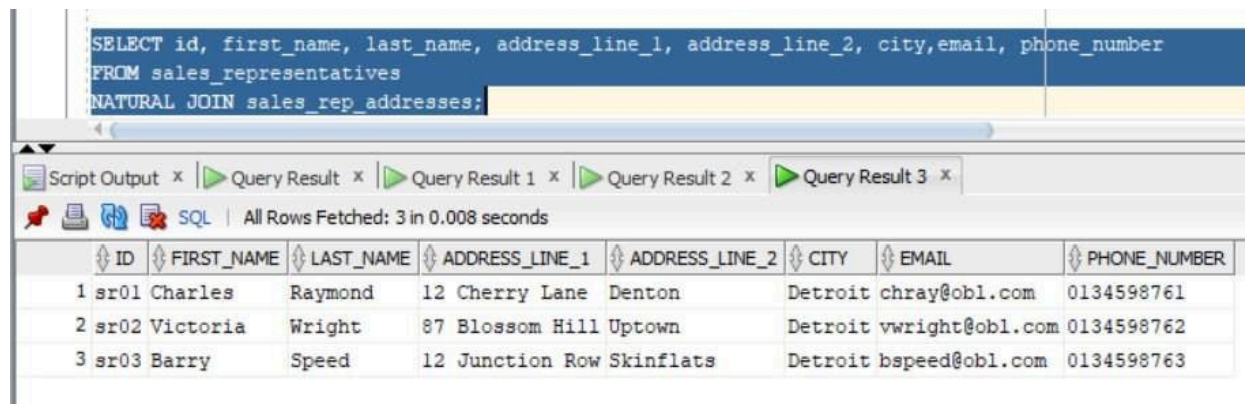
```
SELECT*
FROM sales_representatives
NATURAL JOIN sales_rep_addresses;
```

The results are displayed in a table with 11 columns: ID, EMAIL, FIRST_NAME, LAST_NAME, PHONE_NUMBER, COMMISSION_RATE, SUPERVISOR_ID, ADDRESS_LINE_1, ADDRESS_LINE_2, CITY, and ZIP_CODE. There are 3 rows of data.

| ID | EMAIL | FIRST_NAME | LAST_NAME | PHONE_NUMBER | COMMISSION_RATE | SUPERVISOR_ID | ADDRESS_LINE_1 | ADDRESS_LINE_2 | CITY | ZIP_CODE |
|--------|-----------------|------------|-----------|--------------|-----------------|---------------|-----------------|----------------|---------|----------|
| 1 sr01 | chray@obl.com | Charles | Raymond | 0134598761 | 10 | sr01 | 12 Cherry Lane | Denton | Detroit | DT48211 |
| 2 sr02 | vwright@obl.com | Victoria | Wright | 0134598762 | 5 | sr01 | 87 Blossom Hill | Uptown | Detroit | DT52314 |
| 3 sr03 | bspeed@obl.com | Barry | Speed | 0134598763 | 5 | sr01 | 12 Junction Row | Skinflats | Detroit | DT52564 |

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.

```
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 query execution window with the following query:

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

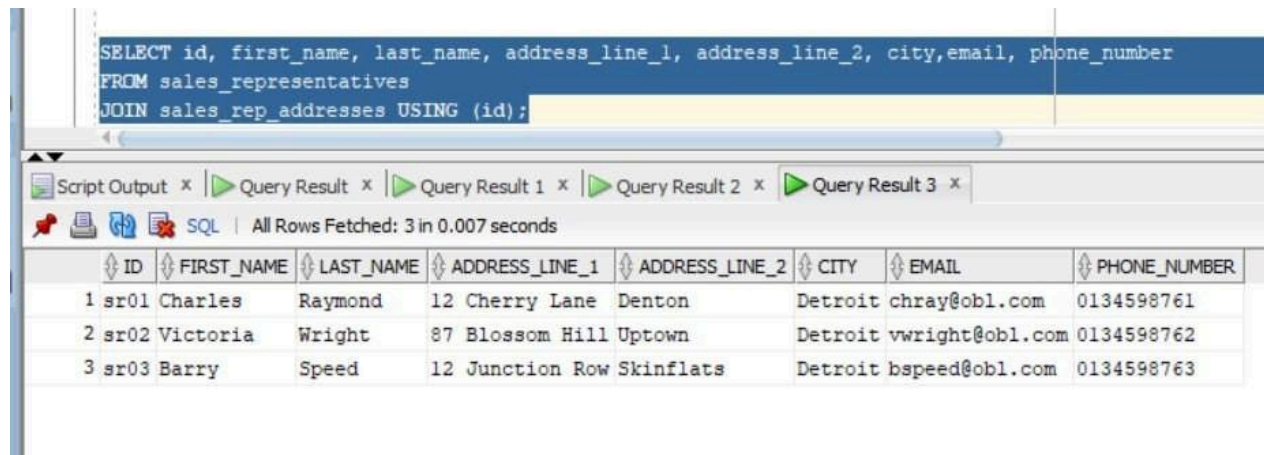
The results are displayed in a table with 8 columns: ID, FIRST_NAME, LAST_NAME, ADDRESS_LINE_1, ADDRESS_LINE_2, CITY, EMAIL, and PHONE_NUMBER. There are 3 rows of data.

| ID | FIRST_NAME | LAST_NAME | ADDRESS_LINE_1 | ADDRESS_LINE_2 | CITY | EMAIL | PHONE_NUMBER |
|--------|------------|-----------|-----------------|----------------|---------|-----------------|--------------|
| 1 sr01 | Charles | Raymond | 12 Cherry Lane | Denton | Detroit | chray@obl.com | 0134598761 |
| 2 sr02 | Victoria | Wright | 87 Blossom Hill | Uptown | Detroit | vwright@obl.com | 0134598762 |
| 3 sr03 | Barry | Speed | 12 Junction Row | Skinflats | Detroit | bspeed@obl.com | 0134598763 |

Part 2: Creating Joins with the USING Clause

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

```
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 at the top containing the following SQL query:

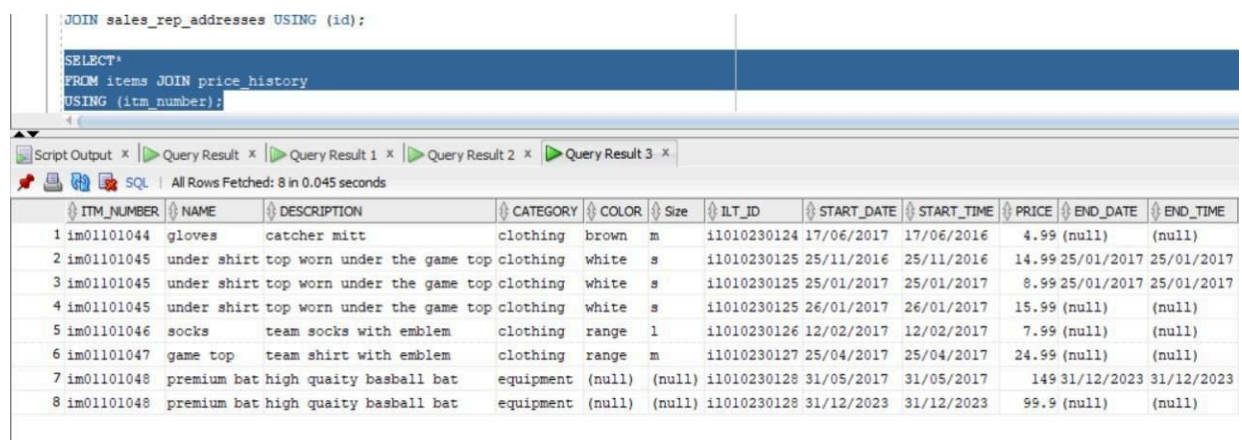
```
SELECT id, first_name, last_name, address_line_1, address_line_2, city,email, phone_number  
FROM sales_representatives  
JOIN sales_rep_addresses USING (id);
```

Below the query editor, the 'Query Result' tab is active, displaying the results of the query. The status bar indicates 'All Rows Fetched: 3 in 0.007 seconds'. The results are shown in a table with 9 columns: ID, FIRST_NAME, LAST_NAME, ADDRESS_LINE_1, ADDRESS_LINE_2, CITY, EMAIL, and PHONE_NUMBER.

| ID | FIRST_NAME | LAST_NAME | ADDRESS_LINE_1 | ADDRESS_LINE_2 | CITY | EMAIL | PHONE_NUMBER |
|--------|------------|-----------|-----------------|----------------|---------|-----------------|--------------|
| 1 sr01 | Charles | Raymond | 12 Cherry Lane | Denton | Detroit | chray@obl.com | 0134598761 |
| 2 sr02 | Victoria | Wright | 87 Blossom Hill | Uptown | Detroit | vwright@obl.com | 0134598762 |
| 3 sr03 | Barry | Speed | 12 Junction Row | Skinflats | Detroit | bspeed@obl.com | 0134598763 |

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

```
SELECT*  
FROM items JOIN  
price_history USING  
(itm_number);
```



The screenshot shows a SQL IDE with a query editor at the top containing the following SQL query:

```
JOIN sales_rep_addresses USING (id);  
  
SELECT*  
FROM items JOIN price_history  
USING (itm_number);
```

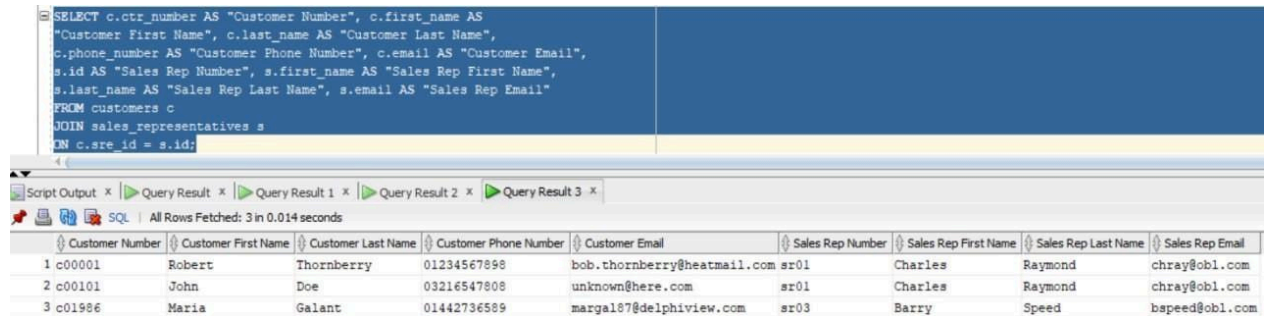
Below the query editor, the 'Query Result' tab is active, displaying the results of the query. The status bar indicates 'All Rows Fetched: 8 in 0.045 seconds'. The results are shown in a table with 12 columns: ITM_NUMBER, NAME, DESCRIPTION, CATEGORY, COLOR, Size, ILT_ID, START_DATE, START_TIME, PRICE, END_DATE, and END_TIME.

| ITM_NUMBER | NAME | DESCRIPTION | CATEGORY | COLOR | Size | ILT_ID | START_DATE | START_TIME | PRICE | END_DATE | END_TIME |
|--------------|---|------------------------|----------|--------|-------------|-------------|------------|------------|-------------|------------|------------|
| 1 im01101044 | gloves | catcher mitt | clothing | brown | m | 11010230124 | 17/06/2017 | 17/06/2016 | 4.99 (null) | (null) | (null) |
| 2 im01101045 | under shirt top worn under the game top | clothing | white | s | 11010230125 | 25/11/2016 | 25/11/2016 | 14.99 | 25/01/2017 | 25/01/2017 | 25/01/2017 |
| 3 im01101045 | under shirt top worn under the game top | clothing | white | s | 11010230125 | 25/01/2017 | 25/01/2017 | 8.99 | 25/01/2017 | 25/01/2017 | 25/01/2017 |
| 4 im01101045 | under shirt top worn under the game top | clothing | white | s | 11010230125 | 26/01/2017 | 26/01/2017 | 15.99 | (null) | (null) | (null) |
| 5 im01101046 | socks | team socks with emblem | clothing | range | l | 11010230126 | 12/02/2017 | 12/02/2017 | 7.99 | (null) | (null) |
| 6 im01101047 | game top | team shirt with emblem | clothing | range | m | 11010230127 | 25/04/2017 | 25/04/2017 | 24.99 | (null) | (null) |
| 7 im01101048 | premium bat high quaity baseball bat | equipment | (null) | (null) | (null) | 11010230128 | 31/05/2017 | 31/05/2017 | 149 | 31/12/2023 | 31/12/2023 |
| 8 im01101048 | premium bat high quaity baseball bat | equipment | (null) | (null) | (null) | 11010230128 | 31/12/2023 | 31/12/2023 | 99.9 | (null) | (null) |

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.

```
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 Rep Number", s.first_name AS "Sales Rep First
Name", s.last_name AS "Sales Rep Last Name", s.email AS "Sales Rep
Email"
FROM customers c
JOIN sales_representatives s ON c.sre_id = s.id;
```



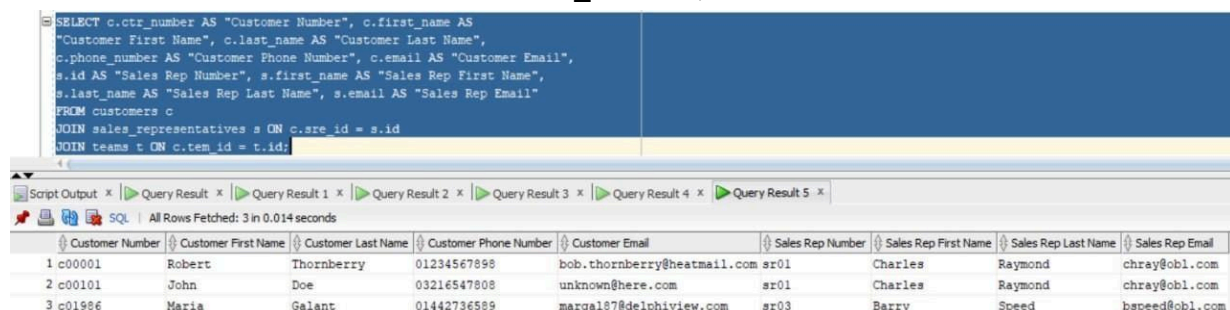
The screenshot shows the SQL Developer interface with the query from Part 3 executed. The 'Query Result' tab is active, displaying 3 rows of data. The columns are: Customer Number, Customer First Name, Customer Last Name, Customer Phone Number, Customer Email, Sales Rep Number, Sales Rep First Name, Sales Rep Last Name, and Sales Rep Email.

| Customer Number | Customer First Name | Customer Last Name | Customer Phone Number | Customer Email | Sales Rep Number | Sales Rep First Name | Sales Rep Last Name | Sales Rep Email |
|-----------------|---------------------|--------------------|-----------------------|-----------------------------|------------------|----------------------|---------------------|-----------------|
| 1 c00001 | Robert | Thornberry | 01234567898 | bob.thornberry@heatmail.com | sr01 | Charles | Raymond | chray@obl.com |
| 2 c00101 | John | Doe | 03216547808 | unknown@here.com | sr01 | Charles | Raymond | chray@obl.com |
| 3 c01986 | Maria | Galant | 01442736589 | margal87@delphiview.com | sr03 | Barry | Speed | bspeed@obl.com |

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.

```
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 Rep Number", s.first_name AS "Sales Rep First
Name", s.last_name AS "Sales Rep Last Name", s.email AS "Sales Rep
Email"
FROM customers c
JOIN sales_representatives s ON c.sre_id =
s.id JOIN teams t ON c.tem_id = t.id;
```



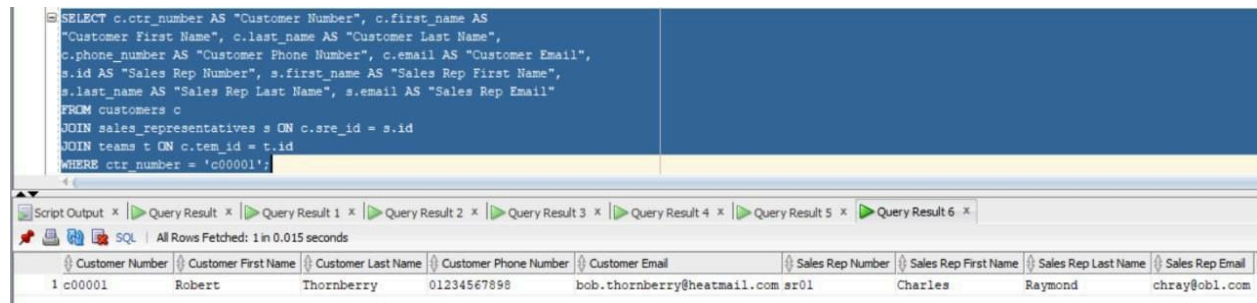
The screenshot shows the SQL Developer interface with the query from Part 4 executed. The 'Query Result' tab is active, displaying 3 rows of data. The columns are: Customer Number, Customer First Name, Customer Last Name, Customer Phone Number, Customer Email, Sales Rep Number, Sales Rep First Name, Sales Rep Last Name, and Sales Rep Email. The results are identical to Part 3, as the additional join to the 'teams' table does not filter the data.

| Customer Number | Customer First Name | Customer Last Name | Customer Phone Number | Customer Email | Sales Rep Number | Sales Rep First Name | Sales Rep Last Name | Sales Rep Email |
|-----------------|---------------------|--------------------|-----------------------|-----------------------------|------------------|----------------------|---------------------|-----------------|
| 1 c00001 | Robert | Thornberry | 01234567898 | bob.thornberry@heatmail.com | sr01 | Charles | Raymond | chray@obl.com |
| 2 c00101 | John | Doe | 03216547808 | unknown@here.com | sr01 | Charles | Raymond | chray@obl.com |
| 3 c01986 | Maria | Galant | 01442736589 | margal87@delphiview.com | sr03 | Barry | Speed | bspeed@obl.com |

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.

```
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 Rep Number", s.first_name AS "Sales Rep First  
       Name", s.last_name AS "Sales Rep Last Name", s.email AS "Sales Rep  
       Email"  
FROM customers c  
JOIN sales_representatives s ON c.sre_id =  
s.id JOIN teams t ON c.tem_id = t.id  
WHERE ctr_number = 'c00001';
```



| | Customer Number | Customer First Name | Customer Last Name | Customer Phone Number | Customer Email | Sales Rep Number | Sales Rep First Name | Sales Rep Last Name | Sales Rep Email |
|---|-----------------|---------------------|--------------------|-----------------------|-----------------------------|------------------|----------------------|---------------------|-----------------|
| 1 | c00001 | Robert | Thornberry | 01234567898 | bob.thornberry@heatmail.com | sr01 | Charles | Raymond | chray@ob1.com |

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

```
SELECT 'The cost of the ' || i.name || ' on this day was ' || y.price AS "Output"  
FROM items i  
JOIN price_history y ON i.itm_number = y.itm_number  
WHERE i.itm_number = 'im01101045'  
AND TO_DATE('12-DEC-2016', 'DD-MON-YYYY') BETWEEN y.start_date  
AND y.end_date;
```

```
SELECT 'The cost of the ' || i.name || ' on this day was ' || y.price AS "Output"  
FROM items i  
JOIN price_history y ON i.itm_number = y.itm_number  
WHERE i.itm_number = 'im01101045'  
AND TO_DATE('12-DEC-2016', 'DD-MON-YYYY') BETWEEN y.start_date AND y.end_date;
```

Script Output x Query Result x Query Result 1 x Query Result 2 x Query Result 3 x Query Result 4 x

SQL | All Rows Fetched: 1 in 0.058 seconds

Output

1 The cost of the under shirt on this day was 14.99