Lab 3: SQL4 - Data Manipulation Language 3

SECD - Database

Universiti Teknologi Malaysia

Objective:

1. To write SELECT statements to access data from more than one table

Reference material: ORACLE Academy; academy.oracle.com

Name : Nur Hanisah Izzati Binti Abdul Haniff

Metric No : A22EC0097

Section : 06

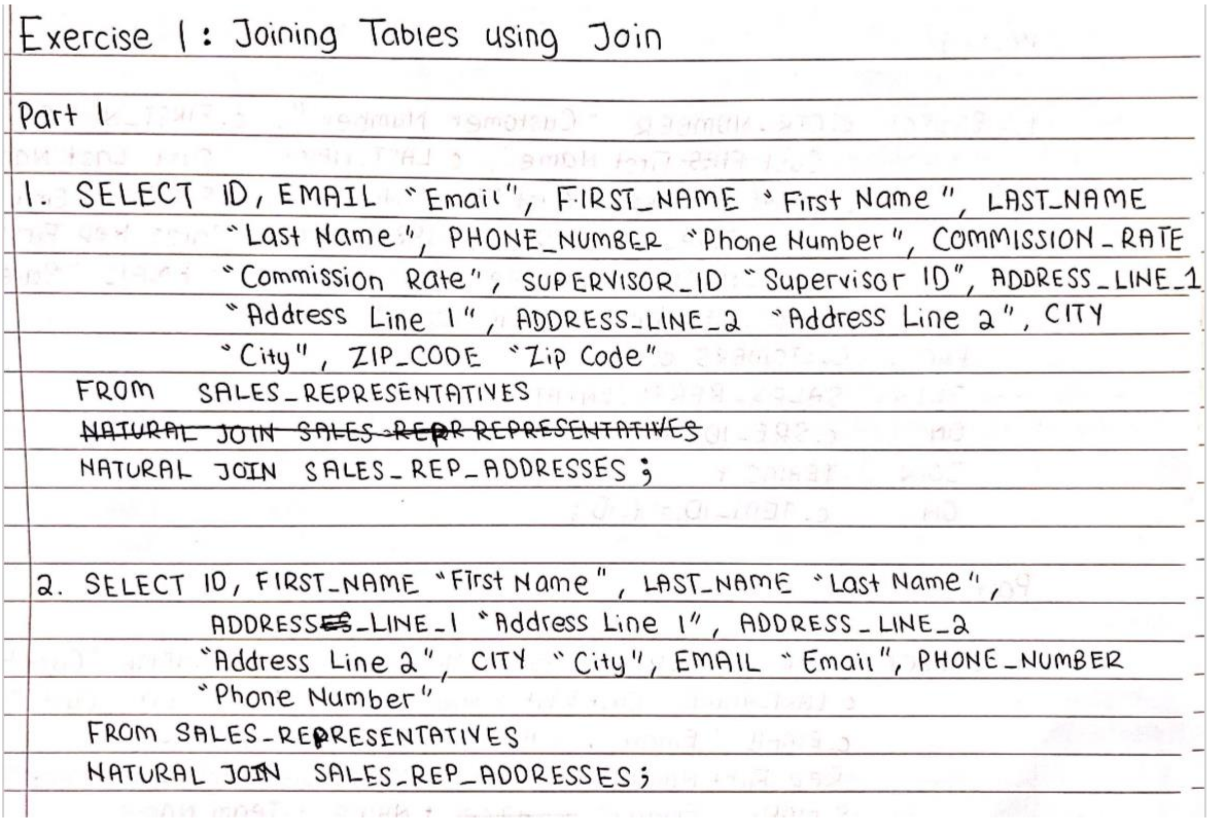
Mark

Exercise 1: Joining Tables Using JOIN

Part 1: Creating Natural Joins.

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

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.



Exercise 1 : Joining Tables using Join

Part 1

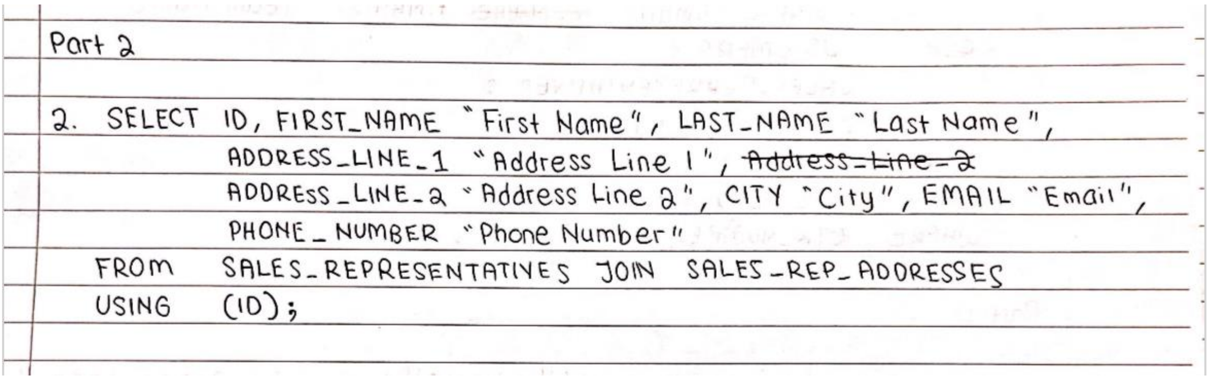
```
1. SELECT ID, EMAIL "Email", FIRST_NAME "First Name", LAST_NAME
          "Last Name", PHONE_NUMBER "Phone Number", COMMISSION_RATE
          "Commission Rate", SUPERVISOR_ID "Supervisor ID", ADDRESS_LINE_1
          "Address Line 1", ADDRESS_LINE_2 "Address Line 2", CITY
          "City", ZIP_CODE "Zip Code"
   FROM SALES_REPRESENTATIVES
   ~~NATURAL JOIN SALES_REPR REPRESENTATIVES~~
   NATURAL JOIN SALES_REP_ADDRESSES;


2. SELECT ID, FIRST_NAME "First Name", LAST_NAME "Last Name",
          ADDRESS~~ES~~_LINE_1 "Address Line 1", ADDRESS_LINE_2
          "Address Line 2", CITY "City", EMAIL "Email", PHONE_NUMBER
          "Phone Number"
   FROM SALES_REPRESENTATIVES
   NATURAL JOIN SALES_REP_ADDRESSES;
```

Part 2: Creating Joins with the USING Clause

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

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

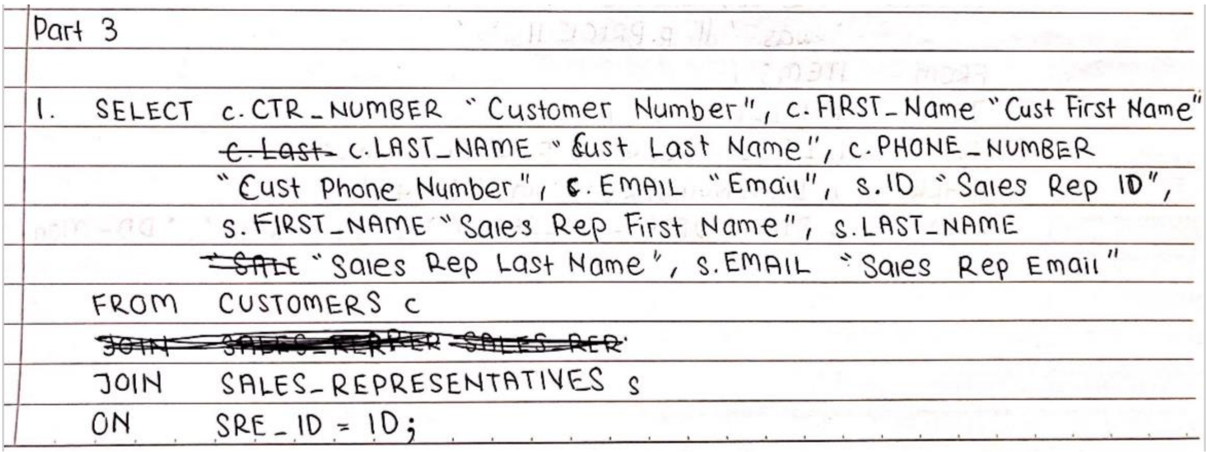


Part 2

2. SELECT ID, FIRST_NAME "First Name", LAST_NAME "Last Name",
ADDRESS_LINE_1 "Address Line 1", ~~Address_Line_2~~
ADDRESS_LINE_2 "Address Line 2", CITY "City", EMAIL "Email",
PHONE_NUMBER "Phone Number"
FROM SALES_REPRESENTATIVES JOIN SALES_REP_ADDRESSES
USING (ID);

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 fist 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.

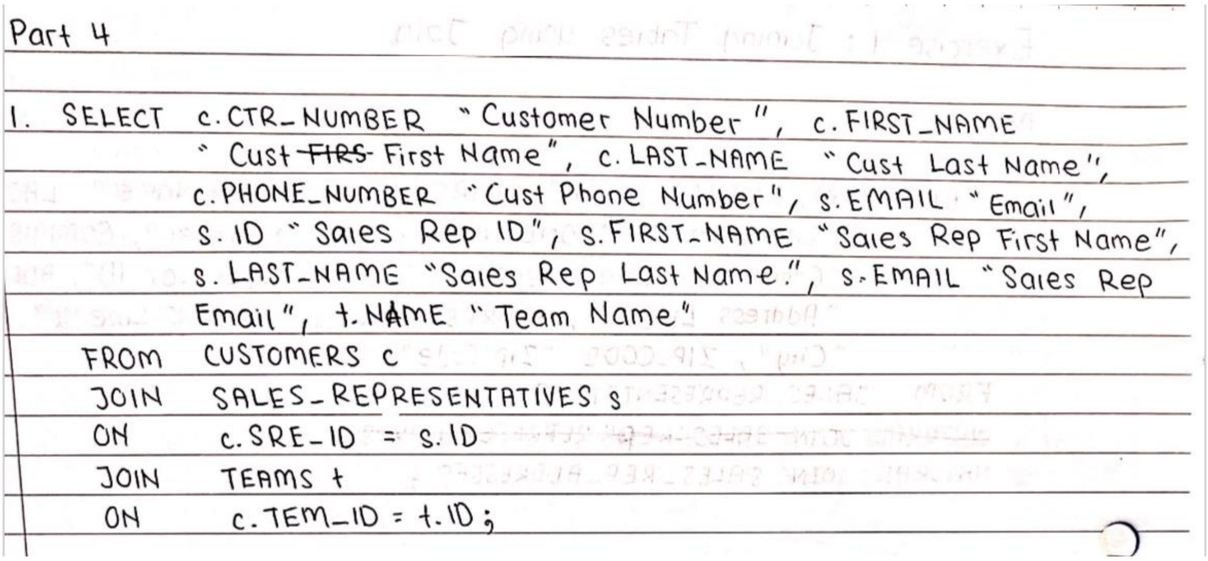


Part 3

1. SELECT c.CTR_NUMBER "Customer Number", c.FIRST_Name "Cust First Name"
~~c.Last~~ c.LAST_NAME "Cust Last Name", c.PHONE_NUMBER
"Cust Phone Number", c.EMAIL "Email", s.ID "Sales Rep ID",
s.FIRST_NAME "Sales Rep First Name", s.LAST_NAME
~~SALE~~ "Sales Rep Last Name", s.EMAIL "Sales Rep Email"
FROM CUSTOMERS c
~~JOIN SALES_REPREP SALES_REP~~
JOIN SALES_REPRESENTATIVES s
ON SRE_ID = ID;

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's name that the customer represents to be included in the results.

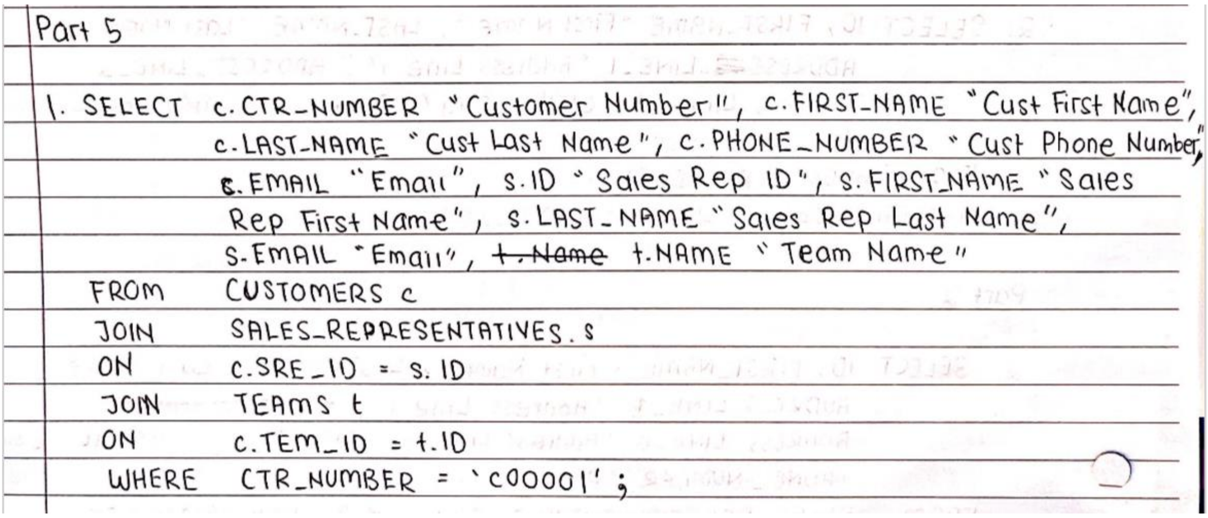


Part 4

```
1. SELECT c.CTR_NUMBER "Customer Number", c.FIRST_NAME
          " Cust ~~FIRS~~ First Name", c.LAST_NAME "Cust Last Name",
          c.PHONE_NUMBER "Cust Phone Number", s.EMAIL "Email",
          s.ID "Sales Rep ID", s.FIRST_NAME "Sales Rep First Name",
          s.LAST_NAME "Sales Rep Last Name", s.EMAIL "Sales Rep
          Email", t.NAME "Team Name"
   FROM   CUSTOMERS c
   JOIN   SALES_REPRESENTATIVES s
   ON     c.SRE_ID = s.ID
   JOIN   TEAMS t
   ON     c.TEM_ID = t.ID;
```

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.



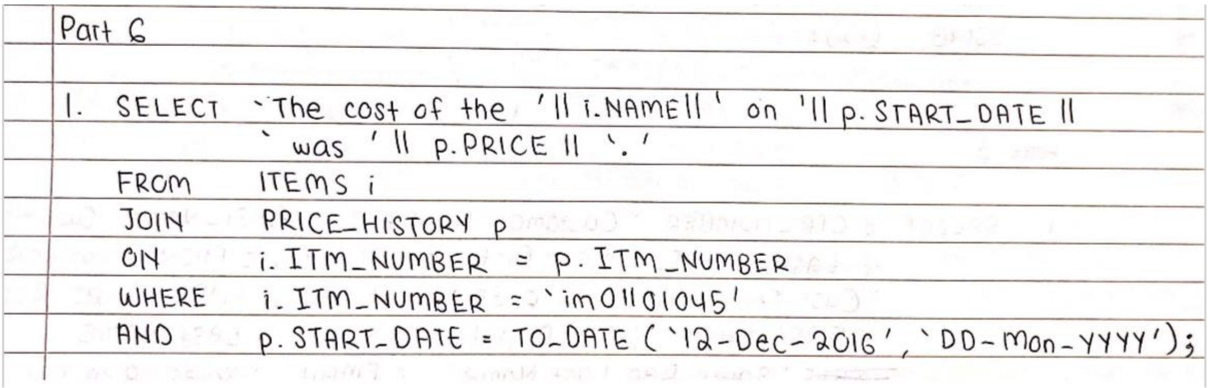
Part 5

```
1. SELECT c.CTR_NUMBER "Customer Number", c.FIRST_NAME "Cust First Name",
          c.LAST_NAME "Cust Last Name", c.PHONE_NUMBER "Cust Phone Number",
          s.EMAIL "Email", s.ID "Sales Rep ID", s.FIRST_NAME "Sales
          Rep First Name", s.LAST_NAME "Sales Rep Last Name",
          s.EMAIL "Email", ~~t.Name~~ t.NAME "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  CTR_NUMBER = 'C00001';
```

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.



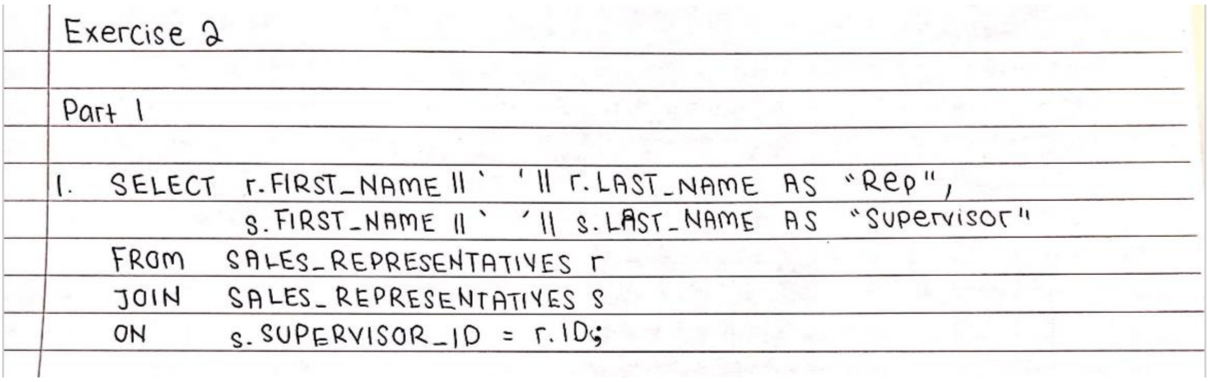
Part 6

```
1. SELECT  `The cost of the '|| i.NAME|| ' on '|| p.START_DATE ||
           ` was ' || p.PRICE || `.'
   FROM    ITEMS i
   JOIN    PRICE_HISTORY p
   ON      i.ITM_NUMBER = p.ITM_NUMBER
   WHERE   i.ITM_NUMBER = `im01101045'
   AND     p.START_DATE = TO_DATE (`12-Dec-2016', `DD-Mon-YYYY');
```

Exercise 2: Joining Tables Using JOIN

Part 1 : Use a Self-Join to Join a Table to Itself

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.



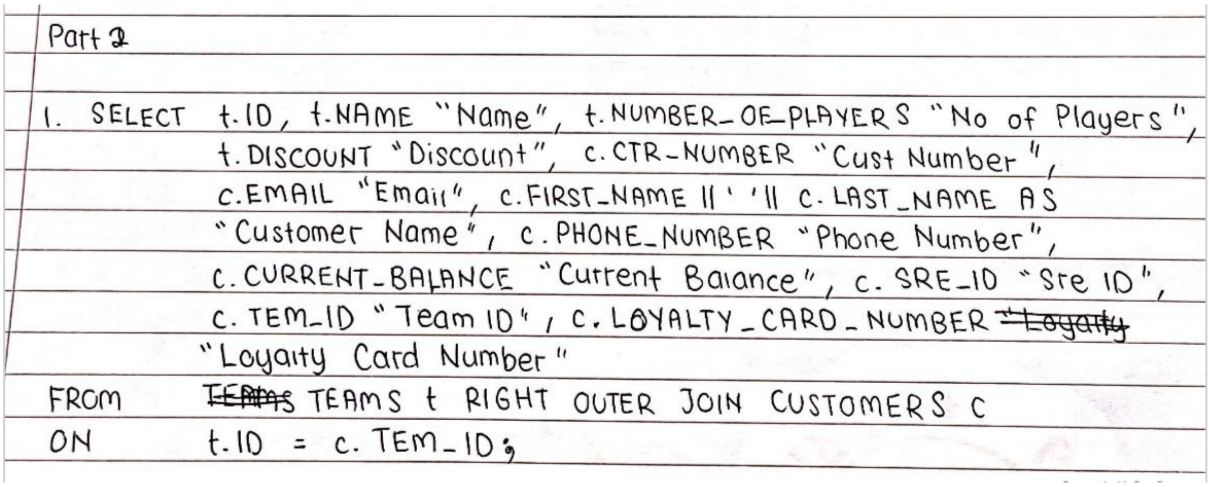
Exercise 2

Part 1

```
1.  SELECT  r.FIRST_NAME || ' ' || r.LAST_NAME  AS  "Rep",
            s.FIRST_NAME || ' ' || s.LAST_NAME  AS  "Supervisor"
    FROM    SALES_REPRESENTATIVES r
    JOIN    SALES_REPRESENTATIVES s
    ON      s.SUPERVISOR_ID = r.ID;
```

Part 2 : Use OUTER joins

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).

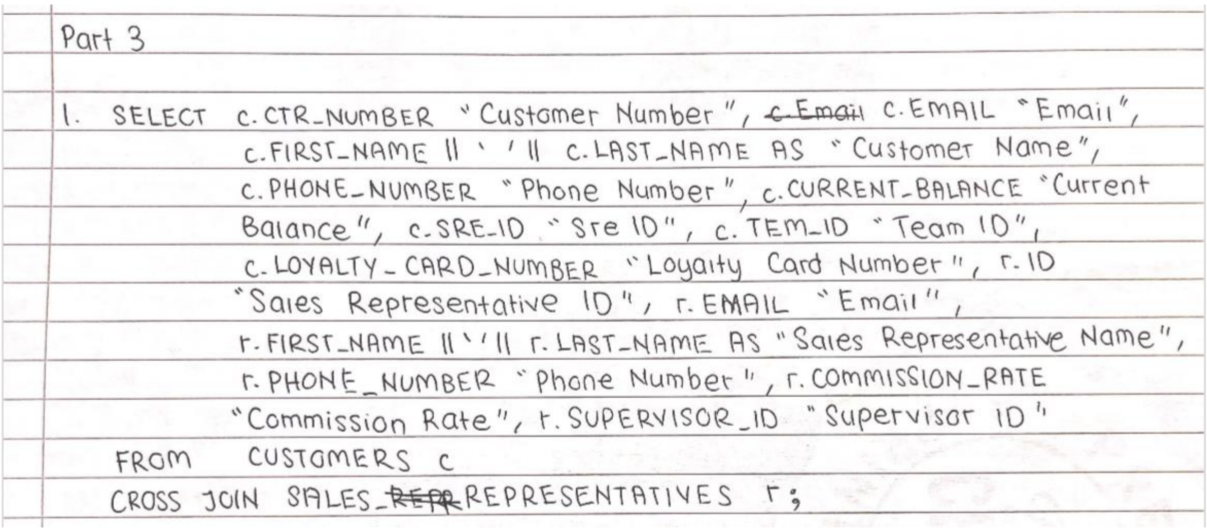


Part 2

```
1.  SELECT  t.ID, t.NAME "Name", t.NUMBER_OF_PLAYERS "No of Players",
            t.DISCOUNT "Discount", c.CTR_NUMBER "Cust Number",
            c.EMAIL "Email", c.FIRST_NAME || ' ' || c.LAST_NAME AS
            "Customer Name", c.PHONE_NUMBER "Phone Number",
            c.CURRENT_BALANCE "Current Balance", c.SRE_ID "Sre ID",
            c.TEM_ID "Team ID", c.LOYALTY_CARD_NUMBER ~~"Loyalty~~
           "Loyalty Card Number"
FROM        ~~TEAMS~~ TEAMS t RIGHT OUTER JOIN CUSTOMERS c
ON          t.ID = c.TEM_ID;
```

Part 3 : Generating a Cartesian Product

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



Part 3

```
1.  SELECT  c.CTR_NUMBER "Customer Number", ~~c.Email~~ c.EMAIL "Email",
            c.FIRST_NAME || ' ' || c.LAST_NAME AS "Customer Name",
            c.PHONE_NUMBER "Phone Number", c.CURRENT_BALANCE "Current
            Balance", c.SRE_ID "Sre ID", c.TEM_ID "Team ID",
            c.LOYALTY_CARD_NUMBER "Loyalty Card Number", r.ID
            "Sales Representative ID", r.EMAIL "Email",
            r.FIRST_NAME ||''|| r.LAST_NAME AS "Sales Representative Name",
            r.PHONE_NUMBER "Phone Number", r.COMMISSION_RATE
            "Commission Rate", r.SUPERVISOR_ID "Supervisor ID"
    FROM    CUSTOMERS c
    CROSS JOIN SALES_~~REPR~~REPRESENTATIVES r;
```