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.

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

2. SELECT id, first_name, last_name, address_line_1, address_line_2, city, email, 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.

```
1. 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);

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

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.

```
1. 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 "Phone Number"
     c.email as "Customer Email", s.id as "Sales Representatives Id"
     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.ctr_number = s.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 name that the customer represents to be included in the results.

```
1. 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 "Phone Number"
     c.email as "Customer Email", s.id as "Sales Representatives Id"
     s.first_name as "Sales Rep First Name", s.last_name as "Sales Rep Last Name"
     s.email as "Sales Rep Email", t.name as "Team Name"
   FROM customers c JOIN sales_representatives s
   ON (c.ctr_number = s.id)
   JOIN teams t ON (c.team 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.

```
1. 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 "Phone Number"
     c.email as "Customer Email", s.id as "Sales Representatives Id"
     s.first_name as "Sales Rep First Name", s.last_name as "Sales Rep Last Name"
     s.email as "Sales Rep Email", t.name as "Team Name"
   FROM customers c JOIN sales_representatives s
   ON (c.ctr_number = s.id)
   JOIN teams t ON (c.team id = t.id)
   WHERE c.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

```
SELECT 'the cost of the under ' || i.name || 'on this day was' || p.price
FROM items i JOIN price_history p
WHERE i.itm_number = 'im 01101045' AND '12/12/2016' between p.start_date and P.end_date;
```

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.

```
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       (r.supervisor_id = supervisor.id);
```

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).

```
SELECT *
FROM teams  LEFT OUTER JOIN;
```

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

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

```
SELECT *
FROM customers, sales_representatives;
```