Database Design Project

Oracle Baseball League Store Database

Project Scenario:

You are a small consulting company specializing in database development. You have just been awarded the contract to develop a data model for a database application system for a small retail store called Oracle Baseball League (OBL).

The Oracle Baseball League store serves the entire surrounding community selling baseball kit. The OBL has two types of customer, there are individuals who purchase items like balls, cleats, gloves, shirts, screen printed t-shirts, and shorts. Additionally customers can represent a team when they purchase uniforms and equipment on behalf of the team.

Teams and individual customers are free to purchase any item from the inventory list, but teams get a discount on the list price depending on the number of players. When a customer places an order we record the order items for that order in our database.

OBL has a team of three sales representatives that officially only call on teams but have been known to handle individual customer complaints.

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.

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.

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 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 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.

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 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 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 * FROM sales_representatives JOIN sales_rep_addresses
   USING (id);

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

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 "Customer Phone Number",
          c.email AS "Customer Email",
          s.id AS "Sales ID",
          s.first_name AS "Sales First Name",
          s.last_name AS "Sales Last Name",
          s.email AS "Sales 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
   AND 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

```
1. SELECT 'The cost of the ' ||i.name|| ' on this day 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 < '12-DEC-2016' AND p.end_date > '12-DEC-2016';
```

Database Design Project

Oracle Baseball League Store Database

Project Scenario:

You are a small consulting company specializing in database development. You have just been awarded the contract to develop a data model for a database application system for a small retail store called Oracle Baseball League (OBL).

The Oracle Baseball League store serves the entire surrounding community selling baseball kit. The OBL has two types of customer, there are individuals who purchase items like balls, cleats, gloves, shirts, screen printed t-shirts, and shorts. Additionally customers can represent a team when they purchase uniforms and equipment on behalf of the team.

Teams and individual customers are free to purchase any item from the inventory list, but teams get a discount on the list price depending on the number of players. When a customer places an order we record the order items for that order in our database.

OBL has a team of three sales representatives that officially only call on teams but have been known to handle individual customer complaints.

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.

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).

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

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

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 s.first_name || ' ' || s.last_name || ' As Sales Representatives',
          v.first_name || ' ' || v.last_name || ' As Supervisor'
   FROM sales_representative s JOIN sales_representativer v
   ON (s.id = v.supervisor_id);
```

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).

```
1. SELECT * FROM teams t LEFT OUTER JOIN customers c
   ON (t.id = c.ten.id);
```

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

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

```
1. SELECT * FROM customers c, sales_representatives s;
```