

# **Oracle Academy**

## **Database Programming with SQL**

### **Instructor Resource Guide**

#### **Notes for PowerPoint Slides:**

#### **SECTION 3 LESSON 1 – Destinations: What's in My Future?**

##### **Slide 1: Destinations: What's in My Future?**

###### **Lesson Preparation**

Internet access is required for this lesson. The following Internet sites are good places for students to begin looking:

Search engine keywords: best colleges, trade schools

[www.scholarstuff.com](http://www.scholarstuff.com)

Military careers

###### **What to Watch For**

Students are not always realistic about what career options they have. Ask students to discuss their plans with a parent or guardian.

Students often don't know how to find information about local community colleges, colleges, universities, and trade schools.

You may want to have parents review and sign each student's planning documents.

For those students who already have concrete plans for their thirteenth year, ask them to plan another option. What if something happened and they could not pursue what seems to be so certain right now? This will eliminate the "I already know what I'm going to do so I don't have to do this" response.

###### **Connections**

Relate the process of developing goals, changing plans, and maybe having to restart a few times in life to the same process students experienced in data modeling. Was the first model the one you eventually adopted? Did all your plans as a group materialize? What changed along the way?

Change is the only thing you can count on. It's best to have more than one play in your playbook - more options instead of fewer.

##### **Slide 2: What Will I Learn?**

Explain the purpose of a 10-year vision. Explain that long-term goals are things that happen over time.

##### **Slide 3: Why Learn It?**

**No instructor notes for this slide**

##### **Slide 4: Tell Me / Show Me – Planning ahead is one step to reaching...**

Students will struggle with 10-year goals, but that's the point. Students are not used to thinking that far into the future. This is a good mental challenge. Just try to keep students thinking realistically -- they may not all be millionaires.

##### **Slide 5: Tell Me / Show Me – Although you might take many turns along ...**

**No instructor notes for this slide**

**Slide 6: Tell Me / Show Me – Add to your plan the monthly activities ...**  
**No instructor notes for this slide**

**Slide 7: Summary – In this lesson you have learned to:**  
**No instructor notes for this slide**

**Slide 8: Summary - Practice Guide**  
**No instructor notes for this slide**

## **SECTION 3 LESSON 2 – Cartesian Product and the Join Operations**

### **Slide 1: Cartesian Product and the Join Operations**

#### **Lesson Preparation**

To introduce joins, discuss the processes involved in taking attendance each day. If your school uses an online attendance system, ask students how they think each teacher's data gets aggregated into the daily totals. If attendance is done manually, how does the office aggregate the data?

Somewhere all the data is combined into one master list.

Explain that SQL has commands that do the job of joining (or combining) data.

#### **Connections**

Now that students are able to select data from more than one table, they should see the purpose behind a relational database and the power to organize and retrieve data easily. Relate this to drawing relationships between entities and identifying attributes and selecting unique identifiers. All the work in data modeling comes to life when the data tables are created.

### **Slide 2: What Will I Learn?**

It is helpful for students to have a copy of the table structure for the DJs on Demand database table layout in order to be able to draw connections between shared primary- and foreign-key columns. Table structures can be located in the course outline.

### **Slide 3: Why Learn It?**

Explain the process of a join using the graphic.

### **Slide 4: Tell Me / Show Me – There are two sets of commands or syntax ...**

**No instructor notes for this slide**

### **Slide 5: Tell Me / Show Me – ORACLE PROPRIETARY JOINS**

**No instructor notes for this slide**

### **Slide 6: Tell Me / Show Me – ORACLE PROPRIETARY JOINS (continued)**

**No instructor notes for this slide**

### **Slide 7: Tell Me / Show Me – In the example at right, which...**

**No instructor notes for this slide**

### **Slide 8: Tell Me / Show Me – EQUIJOIN**

Students should recognize this join from a prior example. Ask students how rows were chosen in this query. Answer: If the columns selected shared that same data, they were selected; if not, they were omitted. In this query, song\_id 50 in the d\_track\_listings table was not returned because the d\_play\_list\_item table does not have this song\_id.

### **Slide 9: Tell Me / Show Me – CARTESIAN PRODUCT JOIN**

**No instructor notes for this slide**

### **Slide 10: Tell Me / Show Me – As with single-table queries, the ...**

**No instructor notes for this slide**

### **Slide 11: Tell Me / Show Me – ALIASES**

**No instructor notes for this slide**

**Slide 12: Tell Me / Show Me –Table aliases precede the column name.**

**No instructor notes for this slide**

**Slide 13: Tell Me /Show Me – Terminology**

Cartesian product-Results from an invalid or omitted join condition; all combinations of rows are displayed

Equijoin-Values in a column in one table must be equal to a value in another table; also called an inner join or simple join

Proprietary join-Connection command exclusive to a specific company

Alias-gives a table another name to simplify queries and improve performance

Join conditions-Display data from two or more related tables

**Slide 14: Summary – In this lesson you have learned to:**

**No instructor notes for this slide**

**Slide 15: Summary - Practice Guide**

**No instructor notes for this slide**

## **SECTION 3 LESSON 3 – Nonequijoins**

### **Slide 1: Nonequijoins**

#### **Lesson Preparation**

None.

#### **What to Watch For**

The nonequijoin should not be difficult for students. Use other verbal examples to make the point of joining two tables using BETWEEN...AND.

### **Slide 2: What Will I Learn?**

Read the objective for this lesson. Ask students if they can guess what a nonequijoin might be designed to do. Accept all answers.

### **Slide 3: Why Learn It?**

**No instructor notes for this slide**

### **Slide 4: Tell Me / Show Me – NONEQUIJOIN**

**No instructor notes for this slide**

### **Slide 5: Tell Me / Show Me – NONEQUIJOIN (continued)**

**To join your number grade...**

**No instructor notes for this slide**

### **Slide 6: Tell Me / Show Me – NONEQUIJOIN (continued)**

**The query shown joins the ...**

Ask students to look at the D\_EVENTS and D\_PACKAGES tables. Ask "What package does the birthday bash event fall into?" Ask them how they arrived at their answers. Look for responses that indicate use of a nonequijoin and the BETWEEN...AND operator.

### **Slide 7: Summary – In this lesson you have learned to:**

**No instructor notes for this slide**

### **Slide 8: Summary - Practice Guide**

**No instructor notes for this slide**

## **SECTION 3 LESSON 4 – Outer Joins**

### **Slide 1: Outer Joins**

#### **Lesson Preparation**

This lesson has a career-exploration activity. Verify Internet access to the following site:

<http://jobstar.org/tools/career/career.cfm>

#### **What to Watch For**

Joins take work and practice. Be sure students understand the use of the plus sign (+) and on which side it needs to be placed in the join condition.

In the nontraditional-career activity, encourage students to find a realistic job.

#### **Connections**

Using the information students found about a nontraditional career, relate it to the development of their personal portfolios, and the importance of having a plan. No matter what kind of career anyone chooses, everyone in any career attended high school at one time, just like you! The plans they made and the goals they accomplished after high school began in high school and continued throughout their lives.

### **Slide 2: What Will I Learn?**

**No instructor notes for this slide**

### **Slide 3: Why Learn It?**

**No instructor notes for this slide**

### **Slide 4: Tell Me / Show Me – An outer join is used to see rows...**

Begin this lesson with the question when do you know whether to use an equijoin or a nonequijoin? In equijoins, the values in both tables must be equal. Frequently, this type of join involves primary- and foreign-key values. A nonequijoin is a join condition containing something other than the equality operator. Use a nonequijoin when the joining of two columns in different tables does not have matching values. Instead, the match occurs within a range of values. Use equijoin when you can match two columns in different tables using equal values (WHERE employees.department\_id = departments.department\_id).

### **Slide 5: Tell Me / Show Me – The query below uses the plus sign ...**

Ask students if we could get an accurate count of all the students that attend our school by just counting the students in class. Obviously, there are students that are home sick, students who are working in the office or attending classes somewhere else part of the day. There has to be another way to count all of the students whether they are in class now or not. Possible Answers: Use an outer join that return student names whether they are in class or not. It's always a challenge for students (and instructors!) to remember which side the plus sign (+) goes on. A good way to remember this is: Think of the plus being used to fill in missing data in a column. Put the plus next to that column. Visualize plus signs filling in all the gaps.

### **Slide 6: Summary – In this lesson you have learned to:**

**No instructor notes for this slide**

### **Slide 7: Summary - Practice Guide**

**No instructor notes for this slide**

## **SECTION 3 LESSON 5 – Self Joins**

### **Slide 1: Self Joins**

#### **Lesson Preparation**

None.

#### **What to Watch For**

Make sure students understand the basics of joins and self-joins.

#### **Connections**

Relate joins to data modeling and the direct relationship between how a data model is designed and how the database functions to store and retrieve information. Relate hierarchical data models to self-joins. Relate unique identifiers and table primary keys. A primary key placed in a related table becomes a foreign key in that table. These table relationship columns become the links that enable joins between tables. It is these relationships between tables that enable a database to have many hundreds of tables from which data can be selected.

### **Slide 2: What Will I Learn?**

**No instructor notes for this slide**

### **Slide 3: Why Learn It?**

Ask students: Who is the manager of employee 17? Who does Bob Jones manage?

### **Slide 4: Tell Me / Show Me – SELF-JOIN**

Ask students: Who is the manager of employee 17? Who does Bob Jones manage?

### **Slide 5: Tell Me /Show Me - Terminology**

**Nonequijoin-A join condition containing something other than an equality operator; values in a column in one table must be conditional to but not equal to a value(s) in another table**

**Self join-Joins a table to itself**

**Outer join>Returns rows that do not meet the join condition**

### **Slide 6: Summary – In this lesson you have learned to:**

**No instructor notes for this slide**

### **Slide 7: Summary - Practice Guide**

**No instructor notes for this slide**

## **Notes For Practice Activities:**

### **Destinations: What's In My Future? S03 L01**

#### ***Try It / Solve It***

Individual Responses

### **Cartesian Product and the Join Operations S03 L02**

#### ***Vocabulary***

<u>Cartesian product</u>	Results from an invalid or omitted join condition; all combinations of rows are displayed
<u>Equijoin</u>	Values in a column in one table must be equal to a value in another table; also called an inner join or simple join
<u>Proprietary join</u>	Connection command exclusive to a specific company
<u>Alias</u>	Gives a table another name to simplify queries and improve performance
<u>Join conditions</u>	Display data from two or more related tables

#### ***Try It / Solve It***

1. SELECT d.event\_id, d.song\_id, d.comments, t.cd\_number, t.track  
FROM d\_play\_list\_items d, d\_track\_listings t;

2. SELECT d.event\_id, d.song\_id, d.comments, t.cd\_number, t.track  
FROM d\_play\_list\_items d, d\_track\_listings t  
WHERE d.song\_id = t.song\_id;

3. SELECT s.title, s.artist, t.description  
FROM d\_songs s, d\_types t  
WHERE s.type\_code = t.code;

4. SELECT s.title, s.artist, t.description  
FROM d\_songs s, d\_types t  
WHERE s.type\_code = t.code AND s.ID IN(47,48);

5. SELECT c.last\_name, e.name, j.job\_date  
FROM d\_clients c, d\_events e, d\_job\_assignments j  
WHERE c.client\_number = e.client\_number AND e.id = j.event\_id;  
(Answers may vary.)

6. SELECT t.song\_id, c.title  
FROM d\_track\_listings t, d\_cds c



WHERE t.cd\_number = c.cd\_number;

7. F\_\_\_a. A join is a type of query that gets data from more than one table based on columns with the same name. (Joined based on common values existing in corresponding columns)

T\_\_\_b. To join tables, there must be a common column in both tables and that column is usually a primary key in one of the tables.

T\_\_\_c. A Cartesian product occurs because the query does not specify a WHERE clause.

F\_\_\_d. Table aliases are required to create a join condition. (Table aliases give a column another name and help keep SQL code smaller. If two tables do not have columns with the same name, the join does not have to use the table.column syntax.)

T\_\_\_e. If a table alias is used for a table name in the FROM clause, it must be substituted for the table name throughout the SELECT statement.

F\_\_\_f. Table aliases must be only one character in length. (Table aliases can be up to 30 characters in length, but shorter is better.)

T\_\_\_g. A simple join or inner join is the same as an equijoin.

8. Being able to combine data from different tables allows you to select and organize data in many different ways.

## Nonequijoins S03 L03

### *Try It / Solve It*

1. 

```
SELECT e.name, p.code
FROM d_events e, d_packages p
WHERE e.cost BETWEEN p.low_range AND p.high_range;
```
2. 

```
SELECT e.last_name, e.salary, j.grade_level
FROM employees e, job_grades j
WHERE e.salary
BETWEEN j.lowest_sal AND j.highest_sal;
```
3. Join condition containing something other than an equality operator
4. 

```
WHERE a.ranking >= g.lowest_rank AND a.ranking <= g.highest_rank
```
5. Use an alias when two tables have columns with the same name, although you can always safely use an alias to qualify column names.
6. nonequijoin

## Outer Joins S03 L04

### *Try It / Solve It*

1. SELECT c.id, o.order\_date,c.last\_name,o.order\_total  
FROM f\_customers c, f\_orders o  
WHERE c.id = o.cust\_id(+);

2. SELECT e.last\_name, e.department\_id, d.department\_name  
FROM employees e, departments d  
WHERE e.department\_id = d.department\_id (+);

3. SELECT e.last\_name, e.department\_id, d.department\_name  
FROM employees e, departments d  
WHERE e.department\_id(+) = d.department\_id ;

4. a. Cannot have (+) on both sides. How can you return data from two tables that is not in either one? One possible corrected solution is: WHERE e.department\_id(+) = d.department\_id;

4b. Missing aliases in the FROM line; column names must have underscores: location\_id; cannot have space between alias and column name. One possible solution is:  
SELECT e.employee\_id, e.last\_name, d.location\_id  
FROM employees e, departments d  
WHERE e.department\_id = d.department\_id;

5. SELECT t.song\_id, c.title  
FROM d\_track\_listings t, d\_cds c  
WHERE t.cd\_number(+) = c.cd\_number;

6. The purpose of this activity is to expand students' thinking about all the possibilities that exist to make a living and enjoy what they do. Help women focus on nontraditional careers. Internet search keywords for this topic include: "odd jobs," nontraditional careers," "nontraditional job titles" or <http://jobstar.org/tools/career/career.cfm>  
Ask students to prepare a short description to share with the class about a job they may be interested in doing.

## Self Joins S03 L05

### *Vocabulary*

<u>Nonequijoin</u>	A join condition containing something other than an equality operator; values in a column in one table must be conditional to but not equal to a value(s) in another table
<u>Self join</u>	Joins a table to itself
<u>Outer join</u>	Returns rows that do not meet the join condition along with rows that do meet the join condition.

### *Try It / Solve It*

1. 

```
SELECT e.last_name as "Employee", e.employee_id AS "Emp#", m.last_name AS "Manager", m.employee_id AS "Mgr#"
FROM employees e, employees m
WHERE e.manager_id = m.employee_id;
```
2. 

```
SELECT e.last_name as "Employee", e.employee_id AS "Emp#", m.last_name AS "Manager", m.employee_id AS "Mgr#"
FROM employees e, employees m
WHERE e.manager_id = m.employee_id(+)
ORDER BY e.last_name;
```
3. 

```
SELECT e.last_name AS "Employee", e.hire_date AS "Emp Hired", m.last_name AS "Manager", m.hire_date AS "Mgr Hired"
FROM employees e, employees m
WHERE e.manager_id = m.employee_id
AND e.hire_date < m.hire_date;
```