Les05-Problems to solve

# PROBLEM 1:

Given these 2 tables: Give Program code and count of students in that program

Programs - Data

|  |  |
| --- | --- |
| ProgramCode | ProgramDescription |
| ACC | Accounting |
| BUS | Business |
| CPA | Computer Programming |
| MKT | Marketing |

Student Data

|  |  |  |
| --- | --- | --- |
| SID | NAME & OTHER DATA | PROGCODE |
| 1111 | first | ACC |
| 2222 | second | ACC |
| 3333 | third | BUS |
| 4444 | fourth | ACC |
| 5555 | fifth | MKT |

I want to know how all the programs at Seneca are doing. One of the things I want to look at is the

(a) program name and

(b) the number of students enrolled in the program.

If did it manually

RESULT – if done manually

Accounting 3

Business 1

Computer Programming 0

Marketing 1

**Since you have done JOINS already these will be simple review for you.**

**1 FIRST STEP If data is found in more than 1 table – Find a common field and JOIN.**

**PROGRAMS data STUDENT data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACC | Accounting | 1111 | first | ACC |
| ACC | Accounting | 2222 | second | ACC |
| ACC | Accounting | 4444 | fourth | ACC |
| BUS | Business | 3333 | third | BUS |
| MKT | Marketing | 5555 | fifth | MKT |
|  |  |  |  |  |

**2 Then selecting rows to display and check if there is a where condition restricting the rows**

**3 and columns from the joined data based on the select**

# PROBLEM 2:

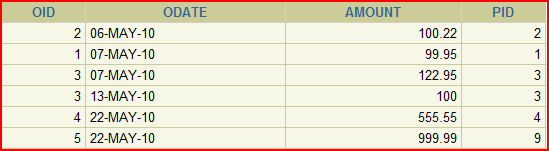
**Given these 2 tables**

**PROVIDE A LIST OF CUSTOMERS AND THEIR SALES AMOUNTS**

CUSTOMERS



ORDERS



The data about customers is in one table and the data about how much was sold is in the ORDERS table.

From your design class you know that there needs to be a common field in order to get the data from 2 or more tables.

The common field is PID (meaning Person ID of the Customer and the FK of PID in the order table)

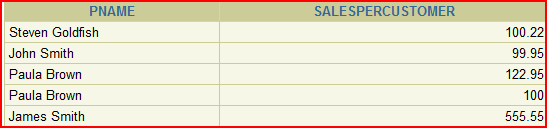
**SELECT pname,**

**Amount FROM Customers, Orders**

**WHERE Customers.pid = orders.pid; *🡸 have to clarify which PID***

The condition was to join on the common field

**RESULT**



THESE ARE CALLED EQUIJOINS

# Problem 3:

Retrieve Department id, Department Name, Location id, city

The data is found in 2 tables 🡪 Department and Locations

Name Null Type

-------------- -------- ------------

LOCATION\_ID NOT NULL NUMBER(4)

STREET\_ADDRESS VARCHAR2(40)

POSTAL\_CODE VARCHAR2(12)

CITY NOT NULL VARCHAR2(30)

STATE\_PROVINCE VARCHAR2(25)

COUNTRY\_ID CHAR(2)

Name Null Type

-------------- -------- ------------

EMPLOYEE\_ID NOT NULL NUMBER(6)

FIRST\_NAME VARCHAR2(20)

LAST\_NAME NOT NULL VARCHAR2(25)

EMAIL NOT NULL VARCHAR2(25)

PHONE\_NUMBER VARCHAR2(20)

HIRE\_DATE NOT NULL DATE

JOB\_ID NOT NULL VARCHAR2(10)

SALARY NUMBER(8,2)

COMMISSION\_PCT NUMBER(2,2)

MANAGER\_ID NUMBER(6)

DEPARTMENT\_ID NUMBER(4)

**EQUIJOIN – what you just did previous page**

**SELECT DEPARTMENT\_ID, DEPARTMENT\_NAME, D.LOCATION\_ID, CITY**

**FROM DEPARTMENTS D, LOCATIONS L**

**WHERE D.LOCATION\_ID = L.LOCATION\_ID**

**NATURAL**

**SELECT DEPARTMENT\_ID, DEPARTMENT\_NAME, LOCATION\_ID, CITY**

**FROM DEPARTMENTS**

**NATURAL JOIN LOCATIONS;**

## 🡪🡪🡪 NB It knows to look for the common named column

* Cannot use a qualifier on location\_id in the select

# But … MUST BE SAME DATA TYPE

**SELECT department\_id, department\_name,**

Added a where clause to limit the rows of output

**location\_id, city**

**FROM departments**

**NATURAL JOIN locations**

**WHERE department\_id IN (20, 50);**

Problem with Natural Join – read it yourself

1. The common column might be another set of columns with matching names
2. Since don’t know what was chosen to be joined sometimes the number of rows might be different

Natural join is like a “shortcut” to save from typing and has been around for a long time.

NOTE: from Wikepedia

Most experts agree that NATURAL JOINs are dangerous and therefore strongly discourage their use.[[3]](http://en.wikipedia.org/wiki/Join_(SQL)#cite_note-3) The danger comes from inadvertently adding a new column, named the same as another column in the other table. An existing natural join might then "naturally" use the new column for comparisons, making comparisons/matches using different criteria (from different columns) than before. Thus an existing query could produce different results, even though the data in the tables have not been changed, but only augmented.

Note that natural is not mentioned as a STANDARD

ANSI standard SQL specifies four types of JOIN: INNER, OUTER, LEFT, and RIGHT.

As a special case, a table (base table, [view](http://en.wikipedia.org/wiki/View_(database)), or joined table) can JOIN to itself in a *self-join*.

Improvement on Natural JOIN with USING

Sample:

**VALID:**

SELECT l.city, D.department\_name

FROM locations L JOIN departments D USING (location\_id)

WHERE location\_id = 1400;

**In this method you ensure what column is being used**

**NOT VALID**: The referenced column (location\_id can not be qualified anywhere)

SELECT l.city, D.department\_name

FROM locations L JOIN departments D USING (location\_id)

WHERE D.location\_id = 1400;

ORA-25154: column part of USING clause cannot have qualifier

PROBLEM 4:

Find the

1) employee id,

2) employee name,

3) department ID,

4) location ID

How many tables to find the data?

This data is found in the EMPLOYEES and DEPARTMENTS tables

SELECT EMPLOYEES.EMPLOYEE\_ID,

EMPLOYEES.LAST\_NAME,

NOTE: Oracle did not like it if bracket missing

DEPARTMENTS.LOCATION\_ID,

DEPARTMENT\_ID

FROM EMPLOYEES JOIN DEPARTMENTS

USING (DEPARTMENT\_ID);

Use table name prefix to qualify an ambiguous column name

🡪 one that is found in both tables

Use table prefixes to improve performance

If no join type mentioned the default is INNER

DO NOT use aliases on columns used with a USING clause and listed elsewhere also

NOTE: … very wordy

Can use aliases just not on the one used in the using statement

Creating Joins with the ON Clause

**🡪** The join condition for the natural join is basically

**an equijoin of all columns with the same name**

**🡪**Use the ON clause to specify arbitrary conditions

**or specify columns to join**

**🡪**The join condition lets you separate the join from other WHERE conditions

**🡪🡪🡪 The ON clause makes the code easy to understand**

SAMPLE:

SELECT e.employee\_id,

e.last\_name,

Notice – no WHERE clause. Separates the 2 ideas of JOIN and WHERE

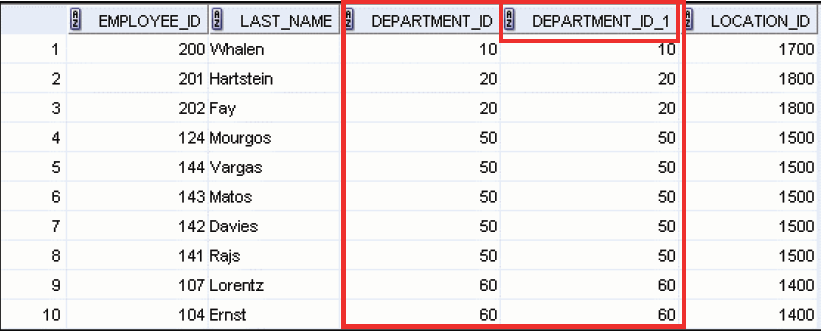
e.department\_id,

d.department\_id,

d.location\_id

FROM employees e JOIN departments d

ON (e.department\_id = d.department\_id);



**NOTE:** Can also use ON for joins on different names

Three-Way Joins

Joining more than 2 tables

**SELECT employee\_id, city, department\_name**

**FROM employees e**

**JOIN departments d**

**ON d.department\_id = e.department\_id**

**JOIN locations l**

**ON d.location\_id = l.location\_id;**

The order of the joins is from LEFT to RIGHT, or in this case first then second

This is mentioned so you will understand the condition in the ON must reference only columns in the tables being joined.

PROBLEM 5:

Find the last\_name of Lorentz's manager

**Here is the employye table**

Lorentz and the manager are all employees. There isn't a need for a manager table.

TABLE: Employees

EMPLOYEE\_ID LAST\_NAME MANAGER\_ID

----------- ------------------------- ----------

100 King

101 Kochhar 100

102 De Haan 100

103 Hunold 102

104 Ernst 103

107 Lorentz 103

124 Mourgos 100

141 Rajs 124

142 Davies 124

143 Matos 124

144 Vargas 124

149 Zlotkey 100

174 Abel 149

176 Taylor 149

178 Grant 149

200 Whalen 101

201 Hartstein 100

202 Fay 201

205 Higgins 101

206 Gietz 205

20 rows selected

HOW TO DO IT

1) Find Lorentz in the employees table by looking up the name in the last\_names column

2) Find the manager number on the same row 🡪 103

3) Use the manager number to search back through the employee table to find a match for employee number 103

NOTE: You are looking in the same table twice

SELF JOIN(p190-slide)

USED TO SOLVE ABOVE PROBLEM:

**SELECT e.last\_name AS emp,**

**m.last\_name AS mgr**

**FROM employees e JOIN employees m**

**ON (e.manager\_id = m.employee\_id)**

**WHERE e.last\_name like 'Lorentz';**

SELF JOIN can have conditions

Show only those with manager 149

SELECT e.employee\_id, e.last\_name, e.department\_id

d.department\_id, d.location\_id

FROM employees s JOIN departments d

ON (e.department\_id = d.department\_id)

AND e.manager\_id = 149;

SELF JOIN can have conditions

Show only those with manager 149

SELECT e.employee\_id, e.last\_name, e.department\_id,

D.Department\_Id, D.Location\_Id

FROM employees E JOIN departments D

ON (E.Department\_Id = D.Department\_Id)

AND E.manager\_id = 149;

Alternately you can use a WHERE clause

SELECT e.employee\_id, e.last\_name, e.department\_id,

d.department\_id, d.location\_id

FROM employees E JOIN departments D

ON (e.department\_id = d.department\_id)

WHERE e.manager\_id = 149;

The above is what is meant when it separates the join from the where condition

Below is the older style using and EQUIJOIN and a WHERE

SELECT e.employee\_id, e.last\_name, e.department\_id,

D.Department\_Id, D.Location\_Id

FROM Employees E, Departments D

Where E.Department\_Id = D.Department\_Id

AND E.Manager\_Id = 149;

NOTE: Equijoins are based on equality or = signs

PROBLEM 6:

Display Managers Last Name and the employees last name working for that manager.

**USING EQUIJOIN**

Select M.Last\_Name As Manager, W.last\_name AS Worker

From Employees M , Employees W

WHERE W.manager\_id = M.employee\_id

Order By 1;

Result:

MANAGER WORKER

------------------------- -------------------------

De Haan Hunold

Hartstein Fay

Higgins Gietz

Hunold Ernst

Hunold Lorentz

King De Haan

King Kochhar

King Hartstein

King Zlotkey

King Mourgos

Kochhar Higgins

Kochhar Whalen

Etc.…. 19 rows

USING ON method

Select M.Last\_Name As Manager, W.Last\_Name As Worker

From Employees M Join Employees W

ON W.Manager\_Id = M.Employee\_Id

order by M.last\_name

AGAIN NOTE: The ON is like the WHERE

PROBLEM 7:

**Display Employee ID, Employee Last\_name and Department Name that they work in**

**RESULTS**

EMPLOYEE\_ID LAST\_NAME DEPARTMENT\_NAME

----------- ------------------------- ----------------

200 Whalen Administration

201 Hartstein Marketing

202 Fay Marketing

124 Mourgos Shipping

141 Rajs Shipping

142 Davies Shipping

143 Matos Shipping

144 Vargas Shipping

103 Hunold IT

104 Ernst IT

107 Lorentz IT

149 Zlotkey Sales

174 Abel Sales

176 Taylor Sales

100 King Executive

101 Kochhar Executive

102 De Haan Executive

205 Higgins Accounting

206 Gietz Accounting

**EXPLICIT METHOD – means you say INNER**

**SELECT employee\_id, last\_name, department\_name**

**FROM employees INNER JOIN departments**

**ON employees.Department\_ID = departments.Department\_ID;**

**IMPLICIT METHOD**

**SELECT employee\_id, last\_name, department\_name**

**FROM employees, departments**

**WHERE employees.Department\_ID = departments.Department\_ID;**

🡺🡺 The ON tends to make it more readable

Repeat of problem from before to

Introduce another problem with a JOIN

Another look at the other set of tables

|  |  |
| --- | --- |
| CUSTOMERS | ORDERS |

**PROBLEM**

**PROVIDE A LIST OF CUSTOMERS AND THEIR SALES AMOUNTS –** shown before and just repeated here

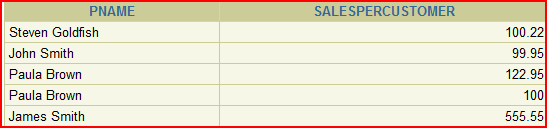
SELECT pname,

Amount AS SalesPerCustomer

FROM Customers, Orders

WHERE Customers.pid = orders.pid

***The condition was to join on the common field***



PROBLEM 8:

Notice Paula Brown has 2 orders

|  |  |
| --- | --- |
| CUSTOMERS | ORDERS |

🡪 **PROVIDE A LIST OF CUSTOMERS AND THEIR SALES AMOUNTS**

🡪 Show only total amount from each of the customers

🡺 Need to use a SUM function

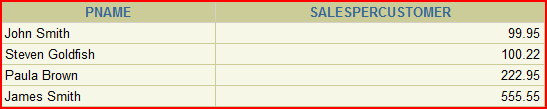
**SELECT pname, SUM(Amount)**

**FROM Customers JOIN Orders**

**ON Customers.pid = orders.pid**

**GROUP BY PNAME** 🡨 MUST HAVE THIS

SINGLE AND GROUP FUNCTIONS ON SELECT



PROBLEM 9:

??? This does not answer a question such as …

**List all customers** and what orders they have placed.

PROBLEM:

Uncle Joe, PID 5, does not show because there is no order for PID 5

The join is based on finding a value in the joining columns.

If no order has been placed then no data will show, but you want all customers

* 🡺🡺 🡺… leads to other types of joins ---

OUTER JOIN

|  |  |
| --- | --- |
| CUSTOMERS | ORDERS |

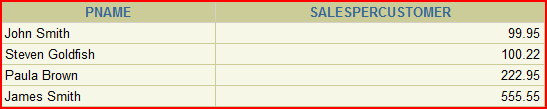
To find what customers ordered we previously used this SQL

**SELECT pname, SUM(Amount) AS SalesPerCustomer**

**FROM Customers JOIN Orders**

**ON Customers.pid = orders.pid**

**GROUP BY PNAME** 🡨 MUST HAVE THIS IN AS HAVE A GROUP AND SINGLE ROW



🡪 🡪🡪 Management wants ALL customers and the resulting sales totals.

**🡺🡺 LEADS to OUTER JOINS**

2 types of Joins

INNER and

OUTER

If you don’t state INNER or OUTER, the default is INNER

🡪 INNER JOIN then is the same as JOIN

INNER JOINS

The **INNER JOIN** will select all rows from both tables 🡪 as long as there is a match between the columns we are matching on.

If a customer has not placed an order or has not placed an order in the time we might specify, then this customer will not be listed as there is no common field.

OUTER JOINS

**Solves the problem of displaying ALL customers and their sales including customers with no sales.**

3 types of OUTER JOINS

🡪 LEFT

🡪 RIGHT

🡪 FULL

**HOW IT WORKS**

SQL:1999

1) Joins of 2 tables that return only matching rows 🡪 INNER JOIN

2) Joins between 2 tables that return

a) result of INNER join

b) any unmatched rows from the left (or right) tables

🡪 called an OUTER JOIN

3) Joins between2 tables that returns the result of

a) an INNER join and

b) all results of both left and right non-matching rows

🡪 called a FULL OUTER JOIN

LEFT JOIN

|  |  |
| --- | --- |
| CUSTOMERS | ORDERS |

SELECT pname,

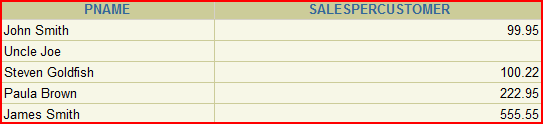
SUM(Amount) AS SalesPerCustomer

FROM Customers **LEFT** JOIN Orders

ON Customers.pid = orders.pid

GROUP BY PNAME

**Now there are 5**



RIGHT JOIN

|  |  |
| --- | --- |
| CUSTOMERS | ORDERS |

SELECT pname,

SUM(Amount) AS SalesPerCustomer

FROM Customers **RIGHT** JOIN Orders

ON Customers.pid = orders.pid

GROUP BY PNAME

Look at the first row.

# 

# Why is this? 🡨 BAD DESIGN

This is a case of the system allowing an order for a non-existent customer == BAD DESIGN

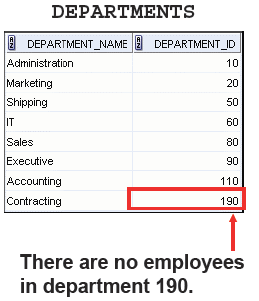
But it does allow us to demonstrate a RIGHT JOIN

MORE OUTER EXAMPLES

Using EMPLOYEES and DEPARTMENTS

Show last\_name, department ID, department\_name

Notice Contracting has no employees and Grant has no department



SELECT E.LAST\_NAME, D.DEPARTMENT\_ID, D.DEPARTMENT\_NAME

FROM EMPLOYEES E FULL OUTER JOIN DEPARTMENTS D

ON (E.DEPARTMENT\_ID = D.DEPARTMENT\_ID);

LAST\_NAME DEPARTMENT\_ID DEPARTMENT\_NAM

------------------------- ------------- --------------

King 90 Executive

Kochhar 90 Executive

De Haan 90 Executive

Hunold 60 IT

Ernst 60 IT

Lorentz 60 IT

Mourgos 50 Shipping

Rajs 50 Shipping

Davies 50 Shipping

Matos 50 Shipping

Vargas 50 Shipping

Zlotkey 80 Sales

Abel 80 Sales

Taylor 80 Sales

Grant

Whalen 10 Administration

Hartstein 20 Marketing

Fay 20 Marketing

Higgins 110 Accounting

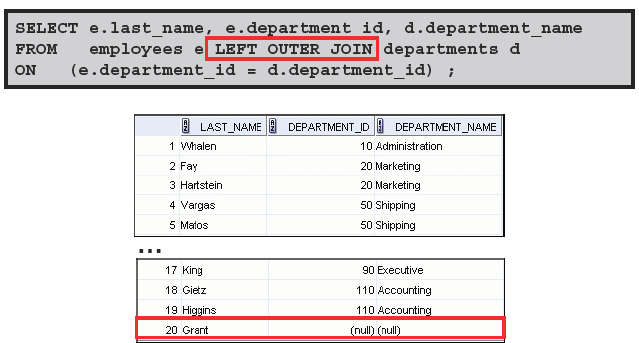
Gietz 110 Accounting

190 Contracting

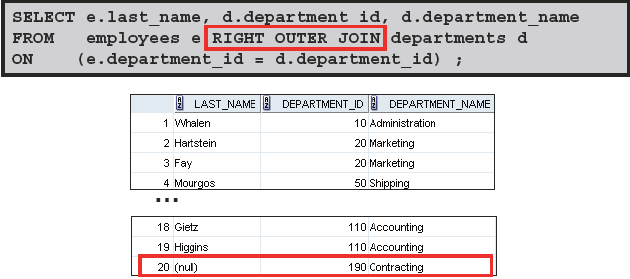
21 rows selected.

Sample of 3 outer joins

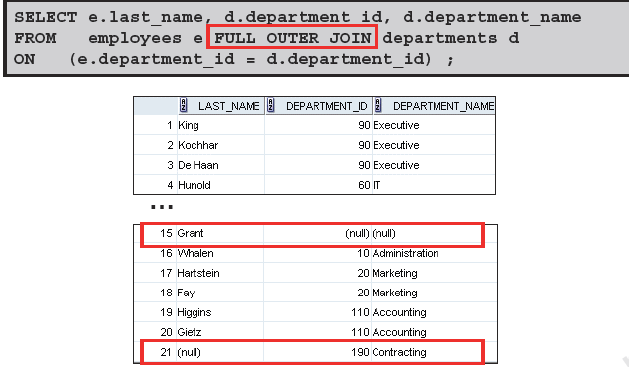
LEFT



RIGHT



FULL



NON-EQUIJOINS

**PROBLEM 10:**

**Find the last name, the salary and what salary grade letter applies to their salary**

Look at the 2 tables JOB\_GRADES and EMPLOYEES

|  |  |
| --- | --- |
| GRADE LOWEST\_SAL HIGHEST\_SAL  - ---------- -----------  A 1000 2999  B 3000 5999  C 6000 9999  D 10000 14999  E 15000 24999  F 25000 40000 | LAST\_NAME SALARY  ------------------------- ----------  King 24000  Kochhar 17000  De Haan 17000  Hunold 9000  Ernst 6000  Lorentz 4200  Mourgos 5800  Rajs 3500  Davies 3100  Matos 2600  … |

Non-equijoins simply mean a join on other than equals or equality

To do this we use an operator other than equals.

In the above tables salary for employee

King has a salary of 24000. It falls in the fifth JOB\_GRADE which is an E

Ernst with salary of 6000 will have a GRADE\_LEVEL of C

SELECT E.LAST\_NAME, E.SALARY, J.GRADE

FROM EMPLOYEES E JOIN JOB\_GRADES J

ON E.SALARY

BETWEEN J.LOWEST\_SAL AND J.HIGHEST\_SAL;

LAST\_NAME SALARY G

------------------------- ---------- -

Vargas 2500 A

Matos 2600 A

Davies 3100 B

Rajs 3500 B

Lorentz 4200 B

Whalen 4400 B

Mourgos 5800 B

Ernst 6000 C

Fay 6000 C

Etc ….

**PRACTICE**

**PRACTICE**

**PRACTICE**