**LAB 5 : JOINS and VIEWS**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Purpose - In this lab you will learn how :

A. To use a**JOIN** for reportsthat needs data from more than 1 table.

B. To create a **VIEW**

- a VIEW contains no data

- the VIEW appears to the user as a table.

- the VIEW runs a “SELECT” to create a temporary table.

- the user can use this temporary table to produce reports or anything else that a table can be used

for.

C. To copy data from one table to another.

D. To use calculated fields that are derived from two or more attributes

.*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

Take screenshot after every successful sql statement.

**Submission instruction: take frequent screenshot, write down the answers, mainly sql statements of the questions only as you proceed with this lab, no print out is required. There may be hand in instruction at the end of each lab.**

**How to Submit: Demonstration in Class only (check the deadline from course agenda and policies, late penalty 100%)**

**PART A:**What if we want to produce a report that contains data from CUSTOMER and SALESREP?

Fields: Customer\_Number

CUSTOMER.First\_Name

CUSTOMER.Last\_Name

SALESREP.First\_Name

SALESREP.Last\_Name

Tables: PREMIERE.CUSTOMER, PREMIERE.SALESREP

The Salesrep data would be the customer’s corresponding Salesrep.   
  
- We will use the common column Sales\_Rep\_Number.

- It isused in both the CUSTOMER and the SALESREP tables.

- The Sales\_Rep\_Number is the primary key in the SALESREP table

- The Sales\_Rep\_Numberis the foreign key in the Customer table.

- When we put a row of the CUSTOMER table together with the corresponding row of theSALESREP table we get a temporary table called a **join.**

We can JOIN CUSTOMER with SALESREP with a SELECT statement:

SELECT c.CUSTOMER\_NUMBER, c.FIRST\_NAME, c.LAST\_NAME,

s.FIRST\_NAME, s.LAST\_NAME

FROM PREMIERE.CUSTOMER c, PREMIERE.SALESREP s

WHERE c.SALES\_REP\_NUMBER = s.SALES\_REP\_NUMBER

ORDER BY c.CUSTOMER\_NUMBER

NOTE:

The characters c and s are aliases.

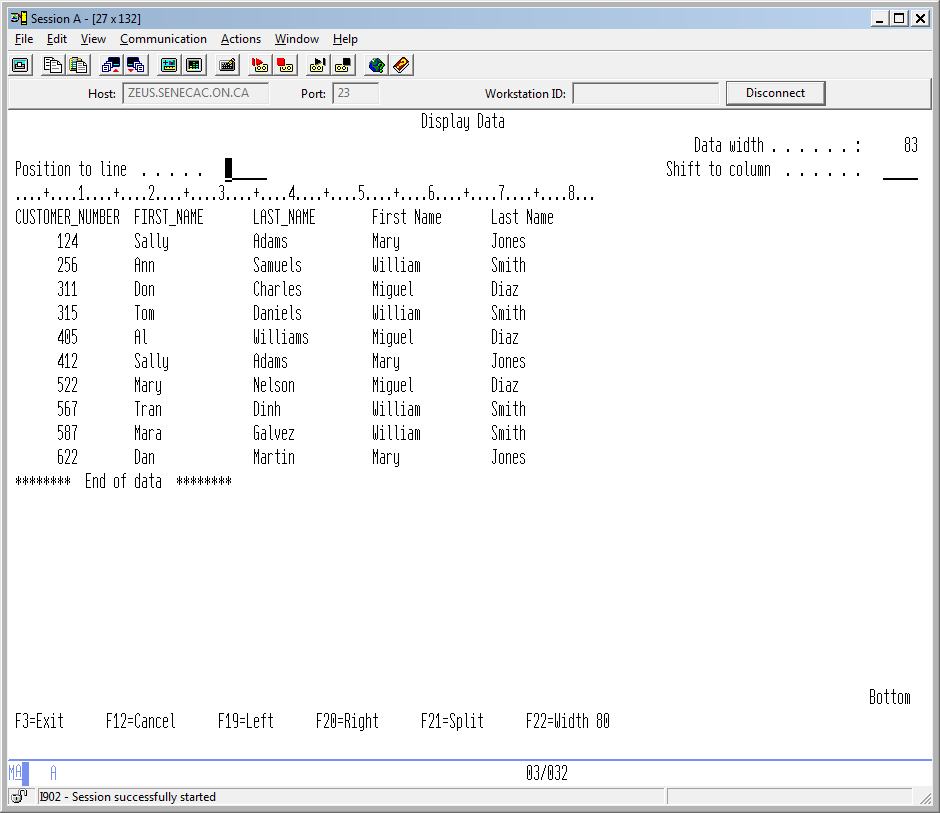
c stands for PREMIERE.CUSTOMER

s stands for PREMIERE.SALESREP

Let's create this report. In Client Access, type in the SELECT exactly as previously provided

.

You should obtain the following display.



What is the problem with this report? By looking at the report we cannot tell which is the Customer Nameand which is the Sales Rep Name

We can fix this. The **concatenation** operator joins two character strings together.

It is || (“||” is the pipe key “|” pressed twice).

For example, ‘ABC’ || ‘/’ || ‘def’ will produce a result of ‘ABC/def’.  
Retrieve the previous command (F9) and change it as follows:

SELECT c.CUSTOMER\_NUMBER,

c.FIRST\_NAME || c.LAST\_NAME as CUST\_NAME,

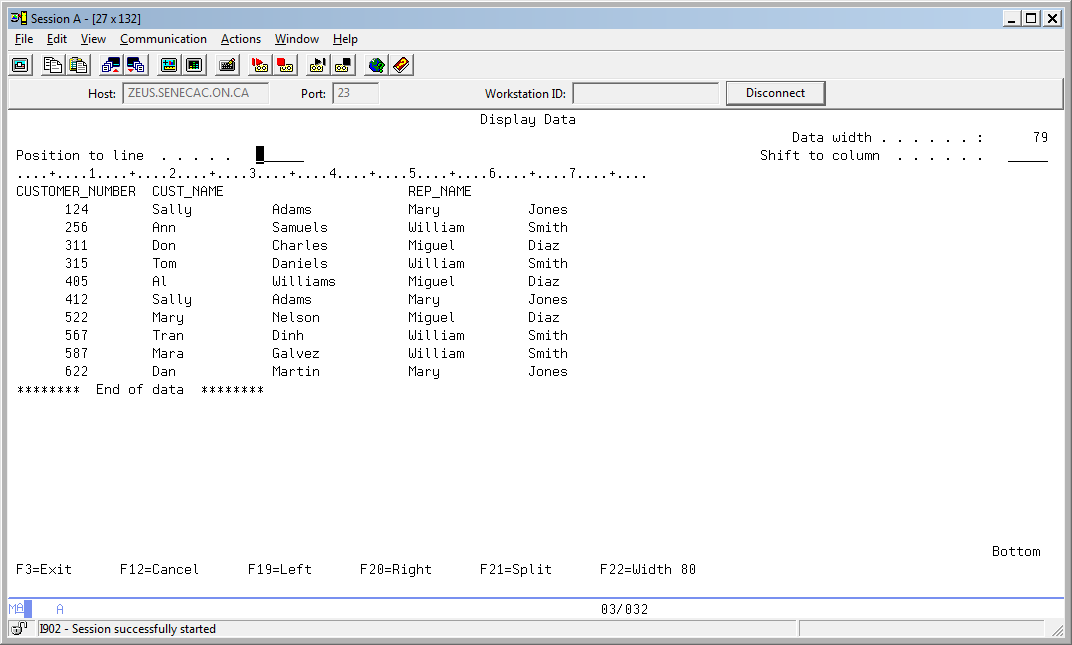
s.FIRST\_NAME || s.LAST\_NAME as REP\_NAME

FROM PREMIERE.CUSTOMER c, PREMIERE.SALESREP s

WHERE c.SALES\_REP\_NUMBER = s.SALES\_REP\_NUMBER

ORDER BY c.CUSTOMER\_NUMBER

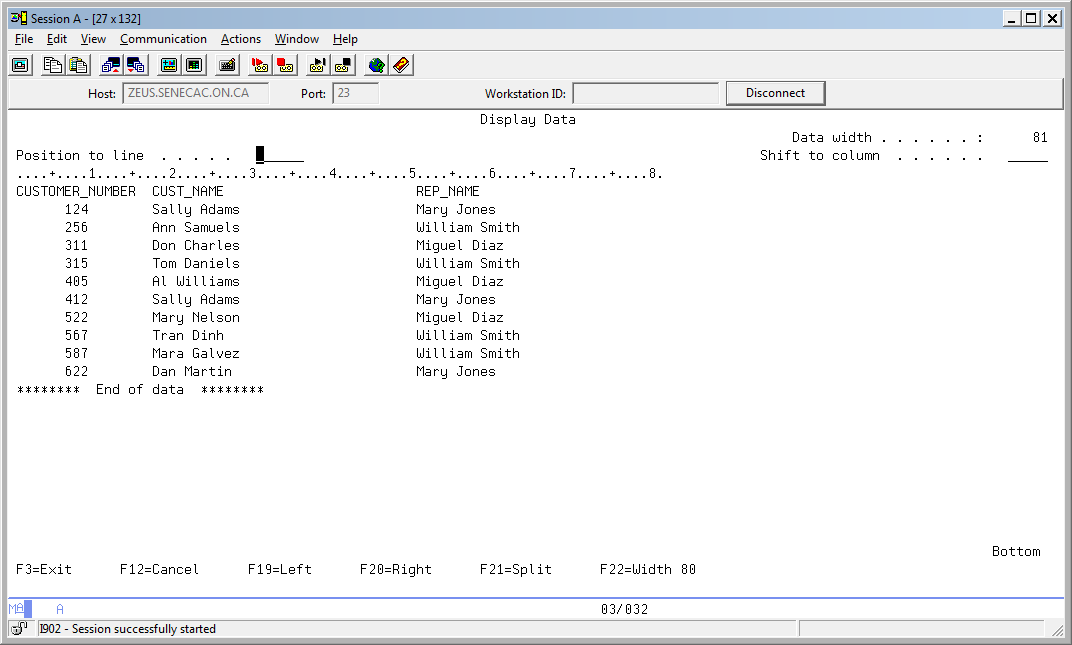
You should get the following display:



**Much Better!**

**Let’s get rid of the wasted space between first and last names.**

**Use the TRIM function to provide the following output:**



**(you will retrieve most of this statement and write all this down later when creating the view)**

SELECT c.CUSTOMER\_NUMBER,

Trim(c.FIRST\_NAME) || ‘ ‘ || trim(c.LAST\_NAME) as CUST\_NAME,

Trim(s.FIRST\_NAME)|| ‘ ‘ || trim(s.LAST\_NAME) as REP\_NAME

FROM PREMIERE.CUSTOMER c, PREMIERE.SALESREP s

WHERE c.SALES\_REP\_NUMBER = s.SALES\_REP\_NUMBER

ORDER BY c.CUSTOMER\_NUMBER

**PART B: Creating a View**

In the previous section, we went to a lot of trouble to generate a report using multiple files, concatenation, and a change of column titles. What if this SELECT were required many, many times? It is possible to save that select statement as a *View*.

The general syntax to create a view is:  
  
 **CREATE VIEW *viewname AS***

**(SELECT …..[ insert your select statement here] ).**

Find and retrieve the SELECT that you used to create the previous report. With your cursor at the beginning of the command, press F15 (Split line). This should insert a new line at the beginning of the command. Now we can enter our CREATE VIEW AS command before the SELECT statement and save having to re-enter the whole thing!

**An ORDER BY clause is not allowed when creating a view, so you will have to remove that clause when creating the view.**

Provide the full statement to produce the CUST\_REP view where names show as “Sally Adams” for the customer and “Mary Jones” for the sales rep. Assume the Set schema command was not run and refer to PRMIER??? in the appropriate areas. This view will be created in your collection and be based on the tables in your collection.

Make sure your written statement is readable; it will be part of the submission for this lab.

===> **CREATE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Press enter to execute the command. You should receive the following message:

**View CUST\_REP created in PRMIERxxx.**

Once a VIEW is created, it can be used in much the same way as a TABLE. You can use SELECT statements to see the data. Run the SELECT statement to look at all the data in the view. (SELECT \* FROM PRMIER???.CUST\_REP.) Your report should look the same as before.

A VIEW can be thought of as a virtual table.It is virtual in the sense that it is not an actual table that stores data, but rather it is a way of presenting data from other tables to you in table format.

You can also think of it as a way of "saving" a complicated select statement so you don't have to re-enter it everytime you want to run that query.

**PART B: Copying data from one table to another**

Remove all the data in the SalesRep table

==>DELETE FROM SALESREP

You should get a message above your delete statement preventing the delete from being executed like the following:

Delete prevented by referential constraint CUSTOMER\_FK in PRMIERB45

**(You may not see this message if the previous lab was not done properly)**

This error message is a good thing. Why?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

So, you need to remove all the rows from Customer first, than remove all the rows from SalesRep next. Note that you are to delete from the prmier???. Do this.

Now that our tables are empty, let’s get the data from Premiere.Customer and the data from Premiere.SalesRep.

The statement to do this is similar to the following (**read ahead**):

INSERT INTO PRMIER???.CUSTOMER SELECT \* FROM PREMIERE.CUSTOMER

Note that this statement is not referring to the column names in either table. That is because the column names were specified in the same order with the CREATE TABLE statement for both tables. Also the columns for both tables agree in type and size. If Customer\_number was defined as a character type with a length of 3 in the first table, it should also be a character type with a length of 3 in the second table.

if you did not create the tables in the previous lab the way you were guided to and are having problems with this step, see if you can run a more sophisticated version of the insert into statement with prompting. Instead of using an asterisk you will have to match up the column names. The last option is to go back to previous lab, drop the tables and recreate carefully.

We needed to run the Delete Statements in a certain order, do we have to follow a certain order when loading your Customer and SaleRep tables with data from the PREMIERE collection tables? If so, what is the order?

Provide the statements here**(insert into salesrep first, why?):**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

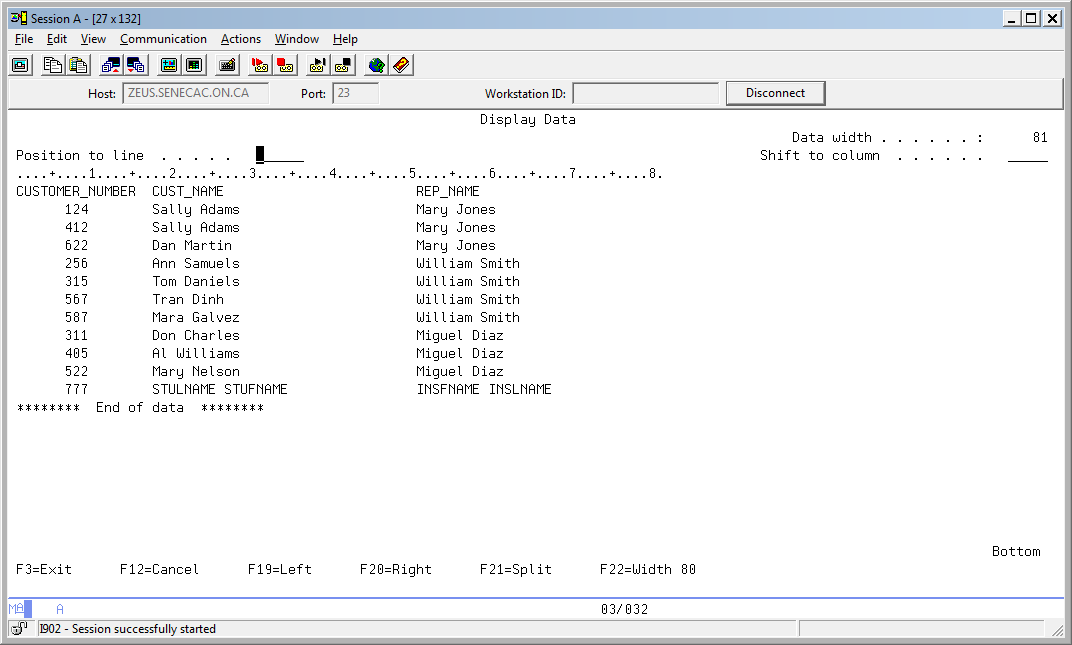
Why does the order matter? (cannot insert non-existent foreign key)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

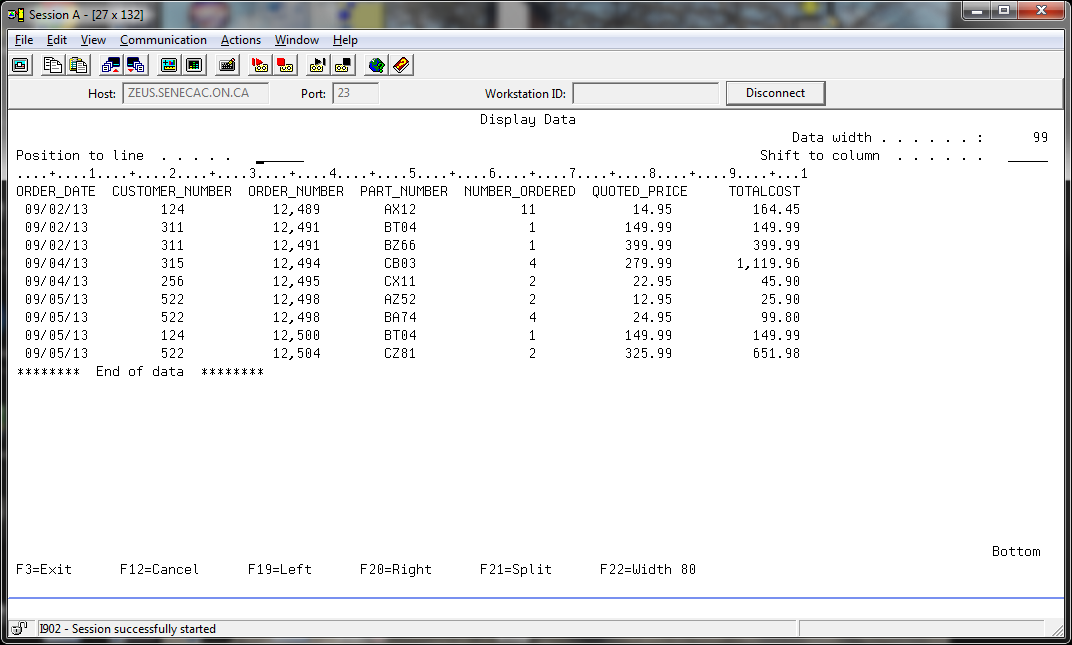
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Insert one row into the SalesRep table with your instructors name and one row in the customer table with your name. Use ‘99’ for the sales rep number for both tables. Your instructor should show up as your SalesRep in the following report. (do not forget the effect of check constraints you created in previous lab)



**PART D: Using calculated fields that are derived from two or more attributes**

AnORDER\_INFO report should look like this



**Try this**

**Set schema prmier???**

**select o.order\_number, o.order\_date, o.customer\_number, ol.part\_numb**

**er, ol.number\_ordered, ol.quoted\_price,(ol.number\_ordered\*ol.quoted\_**

**price) as TotalCOST**

**from premiere.orders o,**

**premiere.orderline ol where**

**o.order\_number = ol.order\_number**

You are to create view using the above sql statement. Look into no 3 of next section for the view name and regarding orders and orderline tables.To create view order\_info in any particular library start the statement like “Create view libraryname.order\_info as (seleact……)”

To Hand In:  
There are 4 items to hand in.

1. A screenshot of the Sales Rep report showing above with your name as a Customer, your instructors name as the SalesRep and the data from the PREMIERE Customer and SalesRep tables in your versions of these tables.

2. The formatted **written** SQL statement used to create the Cust\_Repview (entered on page 4)

3. Create a view in your PRMIERxxx collection and produce a report.

The view should be called ORDER\_INFO. This view will use tables in the PREMIERE database and will be stored in your library (DR201???). The VIEW should contain the Order Date, Customer Number and the Order Number from the ORDERS table and the Part Number, Number Ordered and Quoted Price from the ORDERLINE table. The TotalCost column is a virtual field determined by the Number Ordered and the Quoted Price.

Provide the formatted **written** SQL statement that you used to create the ORDER\_INFO view.

4. Employ a SELECT statement that uses the ORDER\_INFO view to produce a report that displays all

fields in the view. Include screenshot for that report. A sample report shows on the previous page.

**REMEMBER: when you select columns from more than 1 table, you must tell the database which column to match on using the "WHERE" clause.**