**Purpose:**

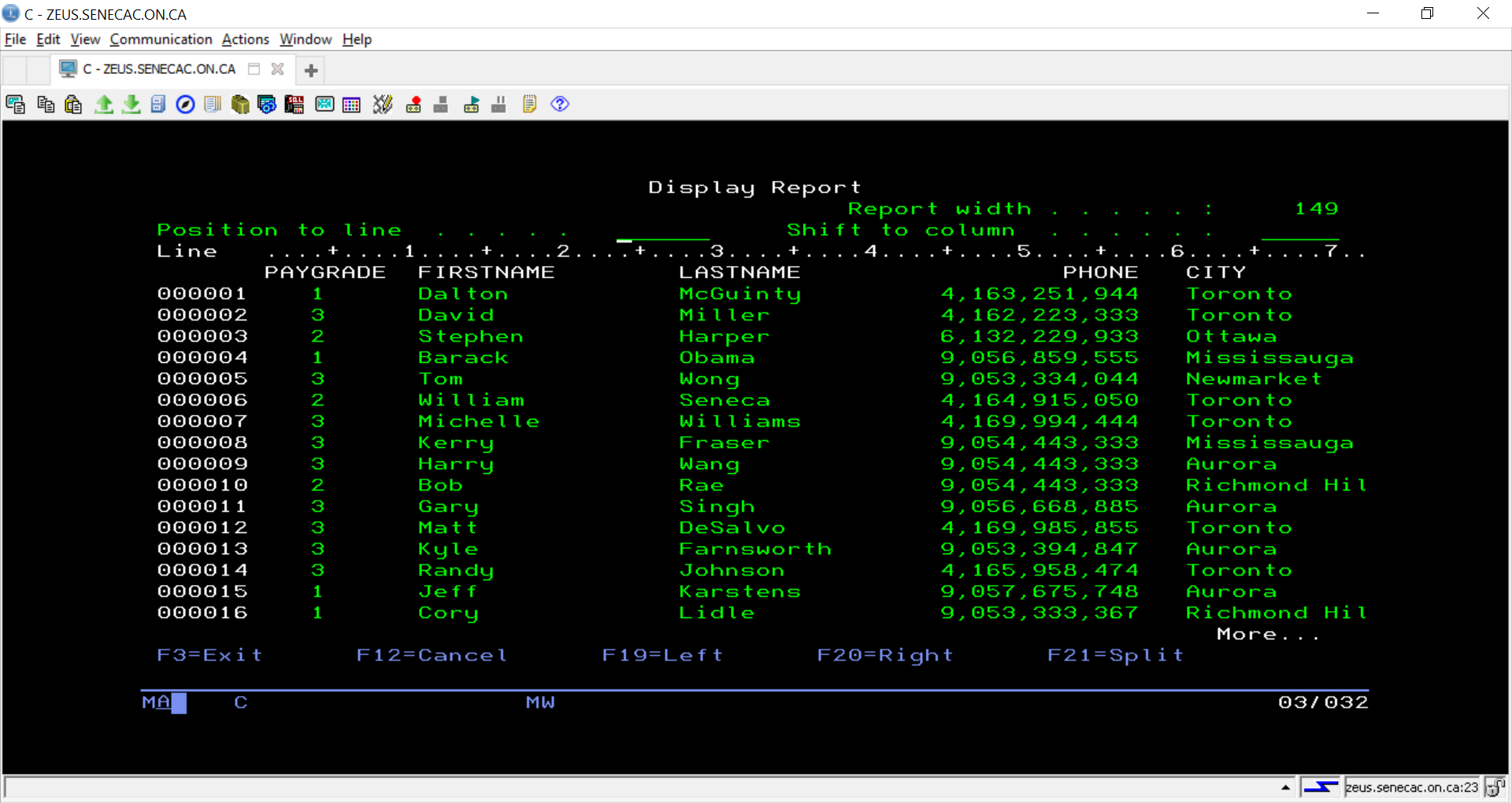
* **To learn the use of interactive SQL on the IBMi.**
* **To create a collection**
* **To create tables for your collection using SQL**
* **To designate one or more attribute(s) as the primary key for a table**

### Requirements to pass the lab:

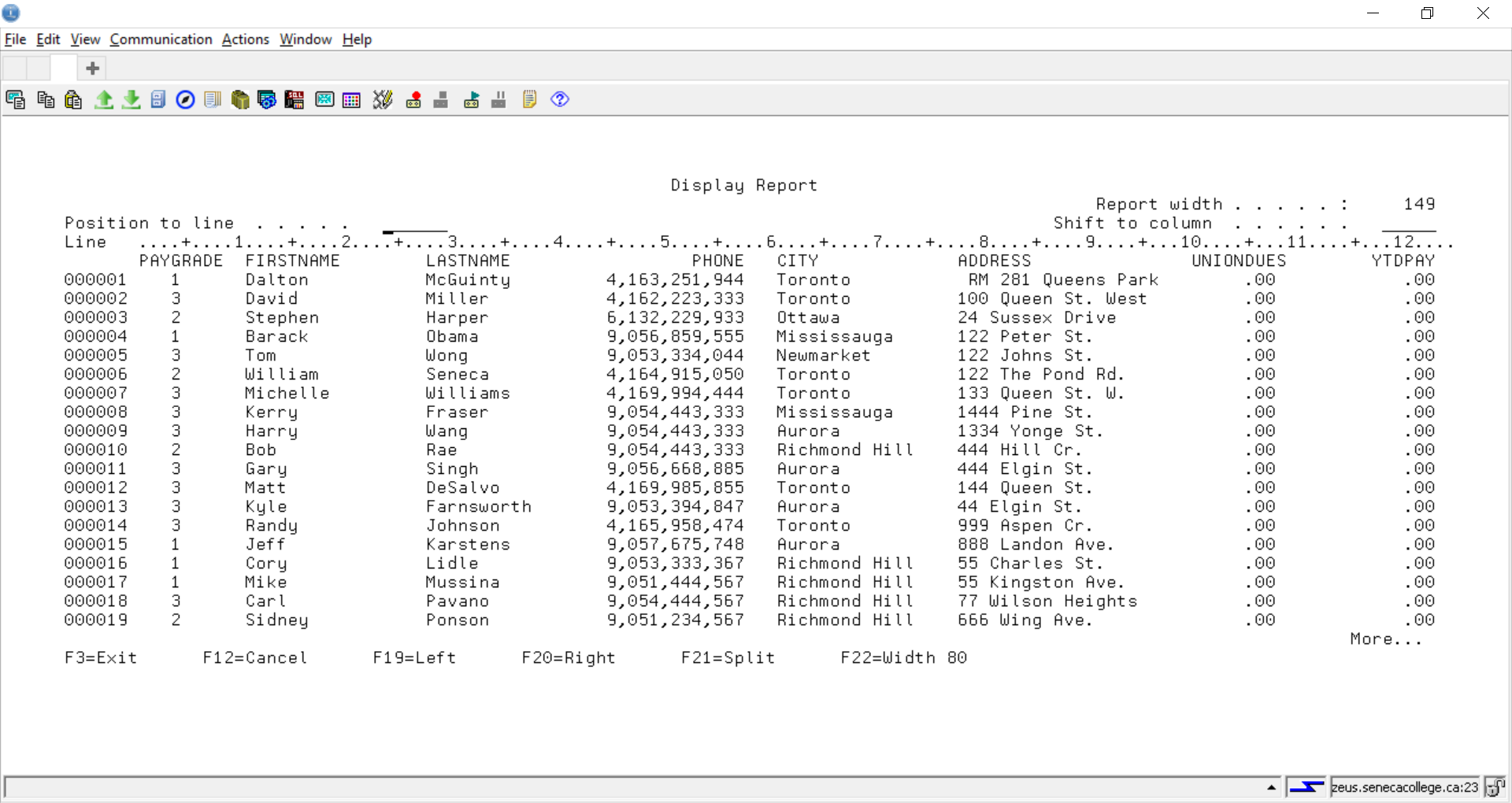
* **Hand in a system generated printout of your CUSTOMER table with your name printed at bottom page and the appropriate data in the table**
* **Hand in a system generated printout of your SALESREP table with your name printed at bottom page and the appropriate data in the table**
* **Answer the questions on the answer sheet**

1. Sign on to the system using the 5250 emulator from Access Client Solutions. The emulator can be configured to use a Whitescreen signon. (This displays 132 columns instead of 80 columns relieving you of pressing function keys to see the left or right sides of any reports that are wider than 80 columns.)

This is a GreenScreen.hod – probably the default. You can click on File at the top menu and select a different hod file.



This is the White Screen hod file showing the same data – more data shows on the screen.



Use the IBM5250-WhiteScreen.hod file. This is available from the DBS201 class page.

Initiate an interactive SQL session.

==> STRSQL

Let’s test out some SQL statements.

The following shows some statements to try and the expected results.

==> SELECT \* FROM PREMIERE.CUSTOMER

(should see the customer data from the CUSTOMER table located in Premiere Collection)

Press F9 to retrieve your previous statement and run it removing the collection name and the period

==> SELECT \* FROM CUSTOMER

(should have an error locating CUSTOMER )

How does the system respond about your failure to execute the SQL statement? (Use answer sheet)

Note, if you have a Customer table in your current library (unlikely at this time) then there obviously would not be a problem finding this table. Also, if a student creates a Customer table with public access in QGPL then the above statement would work because all library list libraries are searched.

Let’s make sure an unqualified reference to CUSTOMER will be directed to the PREMIERE collection

==> SET SCHEMA PREMIERE

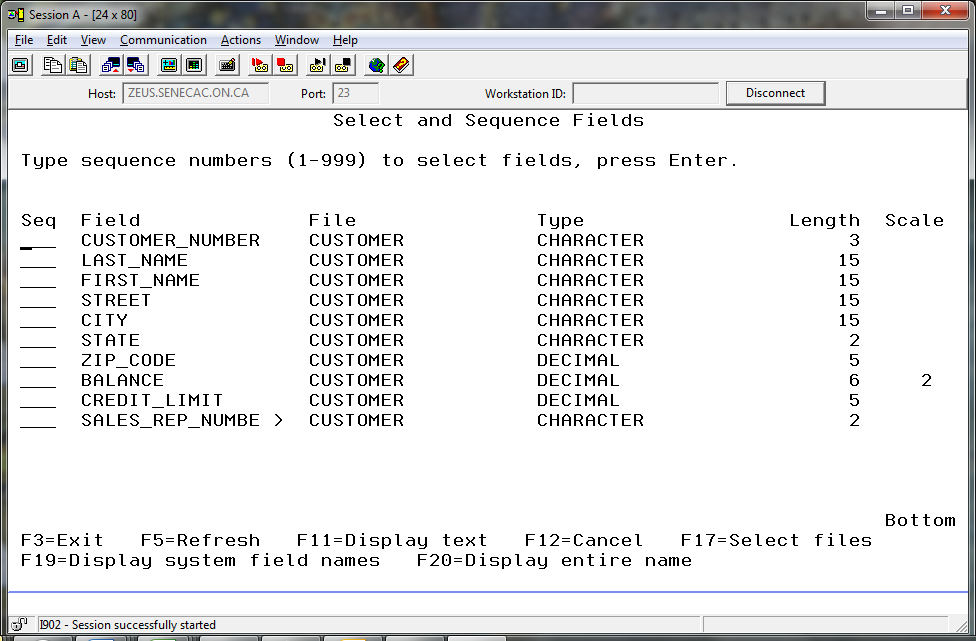
==> SELECT \* FROM CUSTOMER

(should see the customer table rows from that table in the Premiere Collection)

Note that when you quit interactive SQL, you lose the setting that Premiere is the current schema.

==> SELECT \* FROM CUSTOMER (Press F4 here instead of pressing enter)

Go to the second line Select Fields prompt and press F4



We will be creating this table in our own collection soon. The screenshot above is a handy reference for how we will be defining the columns (fields) for this table. One piece of information that is not showing is the full name for the Sales Rep column. You can select just this column name with a “1” beside it. When you press enter, the full name should show. If the asterisk is still there, you can remove it and run the statement. The full column name shows at the top.

SALES\_REP\_NUMBER

03

06

…

Is there a shorter name available for Sales\_Rep\_Number?

On the screen above that lets you Select and Sequence fields you can press F19 (Shift + F7) and you will see the shorter names available. Record them now on the answer sheet in the same order that the longer fields show. You will use this information when you create your tables.

Press F3 to exit from interactive SQL, retain option 1 to save your session on the next screen and press enter.

You have now lost the current schema setting. When you sign back onto the system, the unqualified reference to CUSTOMER will not work.

Restart your SQL session.

2. You will create a **collection** or **database** similar to the Premiere collection.  On the IBMi platform a collection is similar to a library.  All collection or library names must be unique. (If you attempt to create a collection with the name PAYROLL and it is already defined in the system catalog you would get the following message:  **PAYROLL in \*N type \*LIB already exists**.  This happens when another user has already used the name **PAYROLL** as a database name**.**

Because many students will use similar collection or library names, you are to include the last three characters of your ID and shorten the PREMIERE name to ensure the collection name will be unique.  For example, the collection name used by the owner of the **WS201A01** account should be **PREMIERA01**.  A maximum of 10 characters can be used in a collection name.

Type: **CREATE   COLLECTION  PREMIERxxx** (where xxx are the last three characters of your ID) and **press Enter**.

example: **CREATE   COLLECTION  PREMIERA01**

(The creation of your collection may take several minutes.  Some of the time used in creating a collection is devoted to the creation and set-up of some special objects called a **journal** and **journal receiver**.  These objects record the before-image and after-image of any record that is changed in the collection. This allows recovery of the database due to hardware or software failures and provides an audit trail when investigating unauthorized changes to tables. This is part of Transaction Control Language (TCL).

Now **delete** your newly created collection with the **DROP** command as follows:

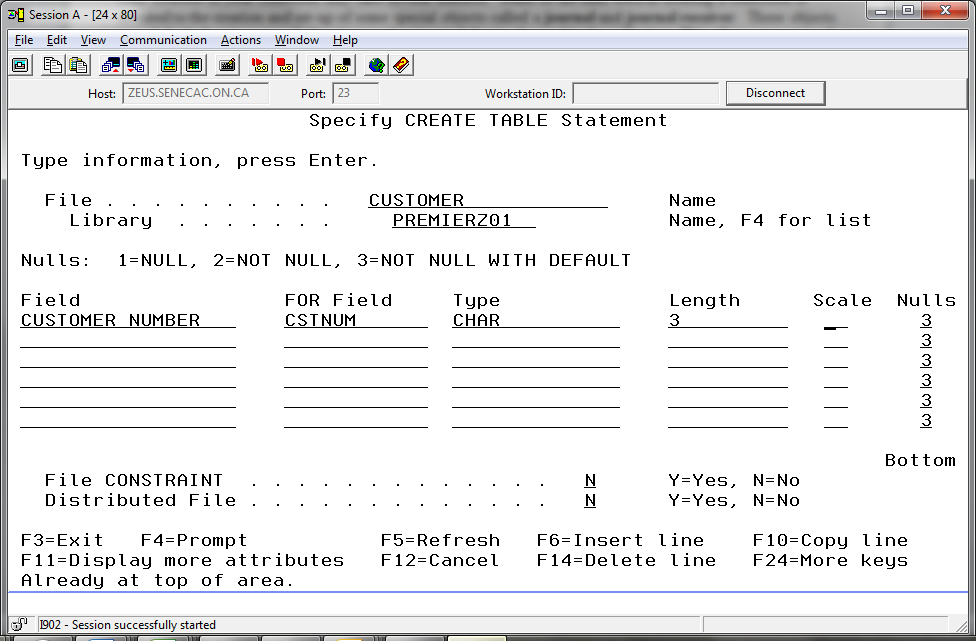
DROP COLLECTION PREMIERxxx (where xxx are the last three characters of your ID)

Now recreate the collection by moving the cursor up to the CREATE COLLECTION statement and **pressing F9 to recall your previous statement to create your collection and re-execute the command**.

3. Let us create two tables similar to the ones in the PREMIERE collection.  Both SALESREP and CUSTOMER need to have the same column definitions in your collection as the

ones in the PREMIERE collection. This will make copying of the data easier in a later lab.

Type CREATE TABLE PREMIERxxx.CUSTOMER  (where xxx is from your ID) and press F4.



Place your cursor under Type (before entering the Customer\_Number). On the answer sheet record the 26 types of columns or fields you can use.

Refer back to the information recorded in the screenshot on page 2 and the information that you wrote down and provide all the required information to create your table. We will use option 3 for the Nulls column above.

When you get to the Zip Code there is no more room to type new field (column) names. Press F6 to insert a new line and then press page down. Press F6 two more times. You now have room to include your Balance, Credit Limit and Sales Rep fields. Note the Balance column will have a scale of 2. Six digits with two decimal positions.

When you have entered all the required column number names, aliases, types, lengths and scales for the CUSTOMER table, press enter. You should have created a new table in your collection.

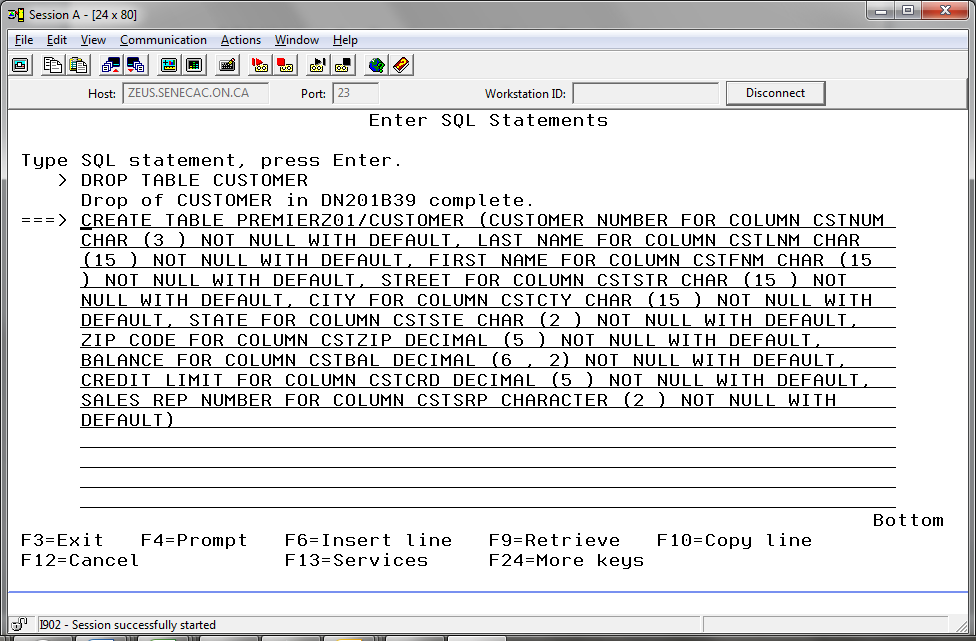
Press F9 to see the statement you have just run. It should look quite confusing. Prompting is great when building an SQL statement with interactive SQL and you are not familiar with how to put a statement together. Prompting is also a great way to save on typing when you are running an ad hoc statement to get a quick answer.

But, you do not want to write your statements that way when you embed them in programs, or use them in scripts, or write them on tests. You will want to clean them up a bit so they are readable.

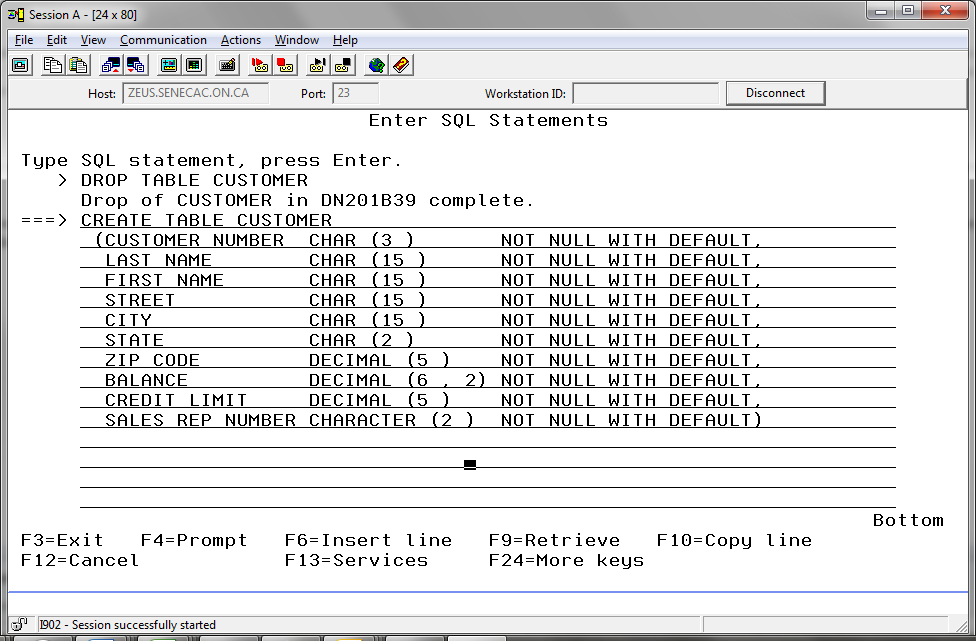
Before we compare the messy and clean version of these statements, let’s try running your SQL statement without a reference to the collection it will be placed in. Press the key that retrieves previous statements \_\_\_\_\_\_\_\_\_ and remove the collection name and period from your statement. By default, the CUSTOMER table will be placed in your current library – not your newly created collection. Indicate on the answer sheet what warning do you get when CUSTOMER is created in your library?

You did not get this warning when CUSTOMER was created in your collection.

Let’s compare the two statements. With the first screen shot, you can see a CUSTOMER table was removed from the current library and a CUSTOMER table was placed in a newly created collection. It is difficult to read.



With the second screen shot we can see aliases were not used and some formatting was done to enhance readability.



**Which statement would you rather have to support!**

You can achieve the nicely formatted SQL statement by retrieving the messy prompted statement and splitting lines where appropriate with the F15 key (Shift + F3). You can also cut and paste where appropriate. This is done by selecting a small block of text and clicking on the Edit menu selection at the top of the screen to cut or to paste. (File Edit Communication … shows at top left corner)

You can also type in the entire statement from scratch.

Get a nicely formatted CREATE TABLE statement in your SQL statement history by pressing F9 and using the advice given above. If the statement runs with a message that the file was already created, you have it correct. If there is some other error response, you need to fix something.

Some explanations:

**Not Null With Default**

This means that when the user does not supply a value for this field, the system will set the field to it’s default value. For a character field that would be spaces. For a numeric or decimal field that would be a 0. For a date field that would be the current date.

It also means a field will not be allowed to be set as null. Null indicates that the value is unknown. If we did allow nulls with a field set up to store the middle initial of someone’s name we would want to store a space in that field when there is no middle initial, a character in that field to represent the known middle initial and a null value there when we do not know anything about the middle initial field for that individual.

**Decimal vs Numeric**

Two of the number data-types to be used are **Numeric** and **Decimal**.  Numeric is Zoned Decimal and Decimal is Packed Decimal.  We will use the data-type of Decimal because less storage space is required and arithmetic operations execute faster using packed decimal fields.

A new feature, the Primary Key

Every table must have a **Primary Key**.  You are now to specify the Primary Key for the CUSTOMER table.  You do this by defining a **primary key constraint** on the CUSTOMER table.    A **constraint** restricts values that can be entered into this table.

Let’s get rid of our successfully created table in our new collection.

**==>DROP TABLE PREMIER???.CUSTOMER** (where ??? is your unique id)

Now, find the nicely formatted version of your create table command. You want to use the prompter to determine how a primary key constraint is added to this statement.

You can use the F9 key to cycle through your previous commands or use the page up key and search for the appropriate command. If you page up and find the command you want, put your cursor on the statement and press F9 to bring it as the current statement to execute.

Press F4 to see if the prompter provides any clues on how to add a primary key constraint.

**Tab down to the File constraint parameter, change the 'N' to 'Y' and press Enter.**

File CONSTRAINT . . . . . . . . . . . . Y

Now press **enter** to Add the constraint.Identify the constraint **type** as **Primary Key** and press the Enter key. You may now name the constraint and choose the column for this constraint. **Name** this constraint **Customer\_ PK** (the convention used for constraint names is the table name plus PK for primary key). Either type in the correct column name or use F4 to locate the column. If you chose the **F4** method, simply write a **1** beside the field of your choice in the left hand column.

**Press Enter** until you receive a message stating that the table has been created. If you have errors, they will be highlighted and you must correct them.  An example of an error would be an incorrect spelling of your collection name.

After your statement correctly runs with the primary key constraint, press F9 to see how it looks. It will be all messed up again. You need to copy the new part and press F9 to get the nicely formatted version of your statement. Add the new part to this readable version.

Be careful, the last Not Null with Default clause should have a comma following it and then the

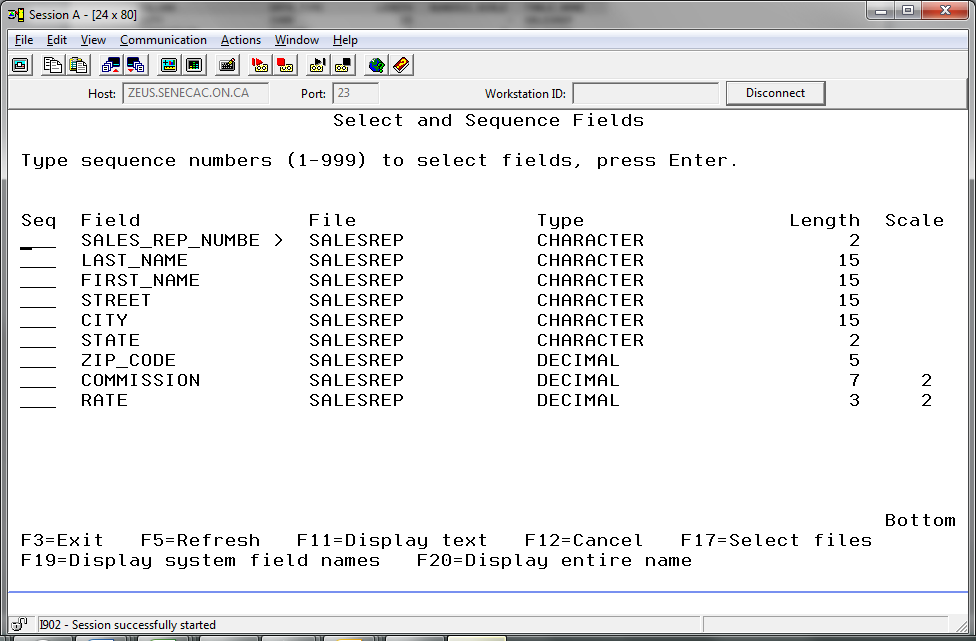
Next line should start with the CONSTRAINT clause that you cut out.

How does that last part look?

SALES\_REP\_NUMBER CHAR (2) NOT NULL WITH DEFAULT,

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

4. The SALESREP table also needs to be created. Use the following information to create the SALESREP table. You must also specify a primary key constraint on the column of Sales\_Rep\_Number.

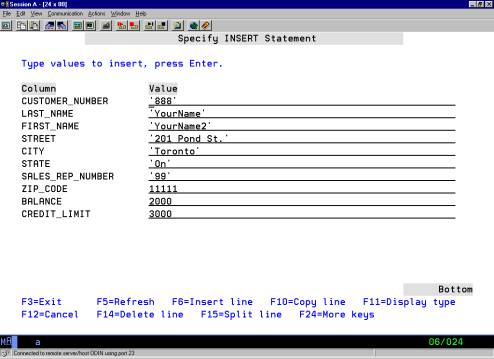


Write out the nicely formatted SQL statement you used to create the SALESREP table on the answer sheet.

5. You now are to place some data in each table.  Add a couple of rows to the CUSTOMER table with one row containing your name and do the same with the SALESREP table using your instructor’s name in the first row.

Type INSERT INTO PREMIERxxx.CUSTOMER  (where xxx  are the last 3 characters of your  id)  and press F4.

Accept the defaults on the first screen and press enter. On the second screen use your real first and last name and keep the suggested data for the other columns



**Press Enter** to submit**.**

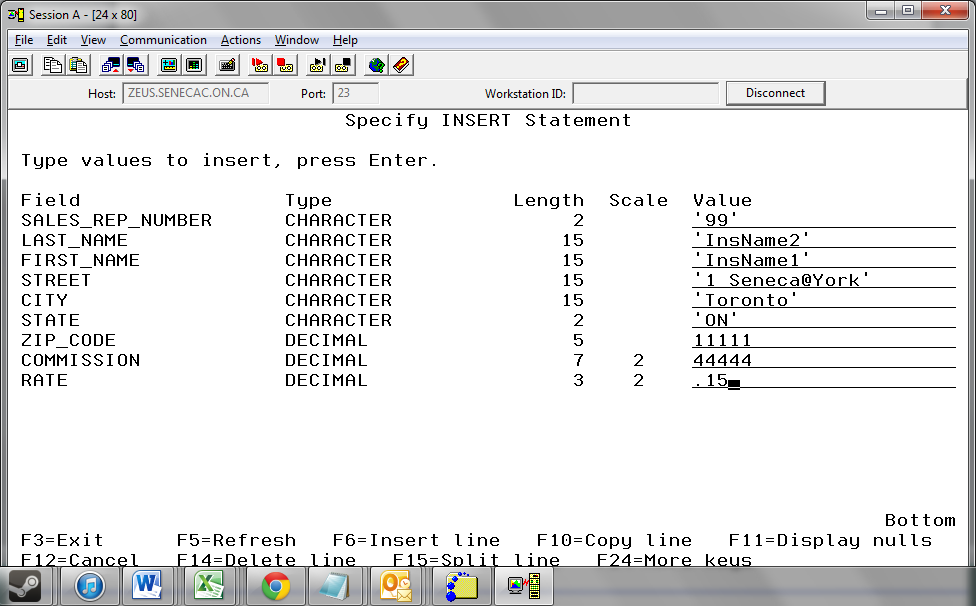
Write a properly formatted SQL statement for your insert command on the answer sheet. It goes much faster than prompting. You will not have the assistance of the function keys on exams, so it’s a good idea to learn the commands without prompting.

Press F9 to enter a second row to the Customer table. Keep the same data for all the columns except the first and last names.

What happens when you press enter?

Make the adjustment that will allow you to include a second row in the Customer Table.

**Now enter a SALESREP row using your instructor’s name and a Sales\_Rep\_Number of ‘99’ and the information shown in the provided screenshot.** Type INSERT (F4) (Enter) and then F11.

Note that this time the insert statement was prompted and F11 was used. F11 is useful in letting you know if you need to put quotes around a value. For example you may see the prompt Sales\_Rep\_Number and assume a value without quotes. But now Type shows and you can see this is a CHARACTER entry.  


Using F9 add a second row to the SalesRep table. This time change the Sales\_Rep\_Number and the first and last names.

Exit interactive SQL and then restart it.

Set the Schema to your new collection and then use unqualified Select statements to see the rows in your two new tables.

**Producing a Proper Report using SQL on the IBMi (review from lab1)**

1) Always make sure your SELECT is displaying the report correctly on the screen first.

2) Change the output so that it goes to a print file by:

– Press F13 (i.e. Shift and the F1 key at the same time)

- Take option 1: Change Session Attributes

- Change the “Select Output” from 1 (display) to 2 (print)

3) Return to SQL by hitting <Enter> three times.

4) Use F9 as many times as necessary to bring back the successful SELECT statement.

Press <Enter> to submit the command.

5) You will not get an error message and you will not see the results on the display, but a print file has been created.

6) Make sure you reset your output back to the display after printing out your two reports. Sometimes students forget to do this and when they sign in again and start interactive SQL the output is not directed to the screen.

7) Print the file according to the instructions of Lab 1.using System i Navigator OR

Exit interactive SQL

Type WRKSPLF

Run a SELECT statement to display the contents of the CUSTOMER table you created today, print it out and hand it in.

Run a SELECT statement to display the contents of the SALESREP table you created today, print it out and hand it in.

Lab 2 Summary

1. Collection names must be unique on the iSeries platform.
2. Unqualified references will default to objects in the libraries in your library list. If you want to make unqualified references to objects in a schema that is not part of your library list – set the schema.
3. A CREATE command produces a new collection, table or other types of SQL object. A Primary Key constraint should always be specified for a table.
4. Function keys make it easier to enter SQL code:  F4 to prompt for long column names and save on typing, F15 to split a line when more room is needed and F6 to insert a line.
5. Deletion of collections, tables and other SQL objects is achieved with a DROP command.
6. There are many choices for a column data-type.  Char, Date, Decimal and Numeric are some of the more commonly used selections.