Assignment 4 Hands-on – INSERT, UPDATE, DELETE AND SEQUENCES psp CASE

***J. Packy Laverty***

***laverty@rmu.edu***

Contents

[**Introduction Assignment 4 – Hands-on 4**](#_Toc486604546)

[**How to complete Hand-on Tutorial Requirements and Review Questions 4**](#_Toc486604547)

[**Oracle Error Codes Summary 5**](#_Toc486604548)

[**1.0 INSERT, UPDATE and DELETE 6**](#_Toc486604549)

[**1.1 INSERT Statement 6**](#_Toc486604550)

[**1.1 INSERT Rules 7**](#_Toc486604551)

[**1.2 NSERT INTO TABLE WITHOUT the COLUMN NAMES 7**](#_Toc486604552)

[**1.3 INSERT DANGERS - Not Specifying the Column Names 8**](#_Toc486604553)

[**ORA-00947 Error Not Enough Values 8**](#_Toc486604554)

[**1.4 INSERT INTO TABLE\_NAME Specifying Column Names 9**](#_Toc486604555)

[**1.5 NO DATA VALUES MEANS NULL 10**](#_Toc486604556)

[**ORA-00001 - Error Unique Constraint 11**](#_Toc486604557)

[**ORA-01400 Error Cannot Insert NULL 11**](#_Toc486604558)

[**ORA-01401 Error Inserted Value Too Large 12**](#_Toc486604559)

[**ORA-01722 Error Invalid Number 12**](#_Toc486604560)

[**ORA-02290 - Error Check Constraint Violated 13**](#_Toc486604561)

[**1.2 Inserting Dates using the TO\_DATE() Function 13**](#_Toc486604562)

[**1.3 Commit Statement 14**](#_Toc486604563)

[**1.3 INSERT VIA SELECT Inserting Data from Other Tables 15**](#_Toc486604564)

[**1.4 INSERT ALL 15**](#_Toc486604565)

[**1.5 Creating a Sequence Object 16**](#_Toc486604566)

[**1.6 Using a Sequence Object in the INSERT Statement 17**](#_Toc486604567)

[**1.6.1 What Is My Next Sequence Number? 17**](#_Toc486604568)

[**1.6.2 Altering a Sequence Object 19**](#_Toc486604569)

[**1.6.3 show\_sequences.sql 19**](#_Toc486604570)

[**1.6.4 Using a Sequence Object with Two Different Tables 20**](#_Toc486604571)

[**1.6.5 Generating Test Data Using a Sequence Object 20**](#_Toc486604572)

[**1.7 Foreign Key Consideration for Inserting Data into a Child Table 21**](#_Toc486604573)

[**ORA-02291 - Error Integrity Constraint Violated 21**](#_Toc486604574)

[**1.8 insert\_rmu2.sql script 22**](#_Toc486604575)

[**2.0 Update Statement 23**](#_Toc486604576)

[**2.1 Updating one or more columns for one or more table rows 24**](#_Toc486604577)

[**3.0 Delete Statement 25**](#_Toc486604578)

[**3.1 DELETE FROM ONE OR MORE ROWS FROM A TABLE 25**](#_Toc486604579)

[**3.2 Do You Know What You Are Deleting? 25**](#_Toc486604580)

[**3.3 Do Not Want to Delete the Entire Row. 26**](#_Toc486604581)

[**3.4 Comparing TRUNCATE TABLE and DELETE FROM 27**](#_Toc486604582)

[**4.0 COMMIT, ROLLBACK AND SAVEPOINTS 27**](#_Toc486604583)

[**4.1 SAVEPOINT EXAMPLE 28**](#_Toc486604584)

[**4.2 CONCURRENCY and Multi-user Issues 28**](#_Toc486604585)

[**4.3 ORACLE Read Consistency 28**](#_Toc486604586)

[**4.4 Set Transaction Read Only 29**](#_Toc486604587)

[**4.5 Documents Requirements 30**](#_Toc486604588)

[**5.0 PSP Case Database 31**](#_Toc486604589)

[**PSP Tables 31**](#_Toc486604590)

[**5.1 General Oracle name rules 33**](#_Toc486604591)

[**5.2 Oracle Data Types 34**](#_Toc486604592)

[**5.3 Character Data Types 34**](#_Toc486604593)

[**5.4 Oracle LOB Data Types 36**](#_Toc486604594)

[**5.5 Oracle Numeric Data Types 37**](#_Toc486604595)

[**5.6 Oracle Date /Time Data Types 38**](#_Toc486604596)

[**Step 1. CODE AND INCREMENTALLY PSP TEST YOUR TABLES, COLUMNS, NOT NULLS and DEFAULT VALUES 39**](#_Toc486604597)

[**Step 2. Use the ALTER TABLE statement to incrementally add the Primary Key for Each PSP Table 40**](#_Toc486604598)

[**Step 3. Use the ALTER TABLE statement to incrementally add Foreign Keys for Each Table. 41**](#_Toc486604599)

[**Step 4. Use the ALTER TABLE statement to incrementally add Check Constraints for Each Table. 41**](#_Toc486604600)

[**Step 5. Use the CREATE INDEX statement to indexes for Each Table. 41**](#_Toc486604601)

[**Step 6. Use the CREATE SEQUENECE statement to create a sequence object to auto increment the following Primary keys. 42**](#_Toc486604602)

[**5.7 Document your PSP Tables 43**](#_Toc486604603)

[**5.8 Document your Customer Table 43**](#_Toc486604604)

[**5.9 Document your Drawing Table 44**](#_Toc486604605)

[**5.10 Document your Option Table 44**](#_Toc486604606)

[**5.11 Document your Drawing Option Table 45**](#_Toc486604607)

[**5.12 Document your Raw Materials Table 45**](#_Toc486604608)

[**5.13 Document your Packlist Table 45**](#_Toc486604609)

[**5.14 Document your PSP Primary Key, Foreign Key, and Check Constraints 46**](#_Toc486604610)

[**5.15 Document your PSP INDEXES 47**](#_Toc486604611)

[**6.0 Enter Test Data into the PSP Database 48**](#_Toc486604612)

[**6.1 Document the Test Data that You Inserted into your PSP Tables 49**](#_Toc486604613)

**Enter your Name Here 🡺**

# Introduction Assignment 4 – Hands-on

**Assignment 4 is organized into two documents: Assignment 4 – Theory and Assignment 4- Hands-on.**

**Assignment 4 – Theory will be allocated 30% of the assignment points**

**Assignment 4 – Hands-on will be allocated 70% of the Assignment points.**

## How to complete Hand-on Tutorial Requirements and Review Questions

This tutorial will provide directions and demonstration examples to guide the student to perform a hands-on requirement. Using these tutorial demonstration examples the student will be required to perform a similar hands-on task. The following is an example of typical tutorial hands-on requirement.

**Use a graphical snipping tool to document your successful logon with a display of your ISPF Primary Option Menu below.**

The directions specify the use of a graphical snipping tool, such as the Window's Snipping Tool. (Directions to use the Window's Snipping Tool will be presented next.) Any graphic snipping tool may be used to demonstrate that you have successfully completed the required hands-on task.

A grey or colored box will be provided after the requirement directions. You are required to provide a graphic image, e.g., using cut-and-paste, to document that the hands-on requirement was correctly completed.

Review questions are also be provided at the end of the tutorial. The following is an example of a review question format. Since type the answer in provided grey or colored box.

1. What is the purpose of a partitioned data set? Answer:

Type in the answer to the question into the grey or colored box.

**It is recommended that you use Table of Contents at the beginning of the tutorial to review and navigate to the concept presented in the review question. Students will find that using the document FIND tool or searching GOOGLE may also be valuable for researching the review question answer.**

## Oracle Error Codes Summary

**Important!**

The following error codes are discussed at least once in this assignment.

**ORA-00904 - Error Invalid Column Name**

**ORA-00907 - Error Missing Right Parenthesis**

**ORA-00922 - Error Missing or Invalid Option**

**ORA-00942 - Error Table or View Does Not Exist**

**ORA-00955 - Name Is Already Used by Existing Object**

**ORA-00957 - Error Duplicate Column Name**

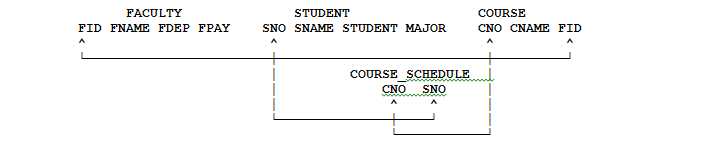
**ORA-02267 - Error Column Type Incompatible (Foreign Key Relationship)**

**ORA-02449 - Error Unique/Primary Keys in Table Referenced by Foreign Keys**

# 1.0 INSERT, UPDATE and DELETE

Data Manipulation Language (DML) statements are used to store, modify, remove, and retrieve data values contained within the database objects created with DDL statements. The four Data Manipulation Language statements available are: INSERT, UPDATE, DELETE, and SELECT.

For the purpose of discussion assume that the following table names, column names and interfile relationships exist in one database.



## 1.1 INSERT Statement

The INSERT statement inserts rows into a table or view or activates the INSTEAD OF INSERT trigger. The table or view can be at the current server or any DB2® subsystem with which the current server can establish a connection. Inserting a row into a view inserts the row into the table on which the view is based if no INSTEAD OF INSERT trigger is defined on the specified view. If an INSTEAD OF INSERT trigger is defined, the trigger is activated instead of the INSERT statement.

There are three forms of this statement:

* The **INSERT via VALUES** form is used to insert a single row into the table or view using the values provided or referenced.
* The **INSERT via SELECT** form is used to insert one or more rows into the table or view using values from other tables, or views, or both.
* The **INSERT via ALL (Multiple) Rows** form is used to insert multiple rows into the table or view using one or more values sets provided or referenced. This form of INSERT is supported in PL/SQL procedure applications.

**Video -**

**Inserting data into an Oracle Table (Theory) -**

[**http://www.youtube.com/watch?v=Z6XYBFcDmCs&feature=channel&list=UL**](http://www.youtube.com/watch?v=Z6XYBFcDmCs&feature=channel&list=UL)

**Video - Insert Data into an Oracle Table -** [**http://www.youtube.com/watch?v=Z6XYBFcDmCs&feature=BFa&list=ULUb\_PtqVUZok**](http://www.youtube.com/watch?v=Z6XYBFcDmCs&feature=BFa&list=ULUb_PtqVUZok)

**Video - SQL Insert Command**

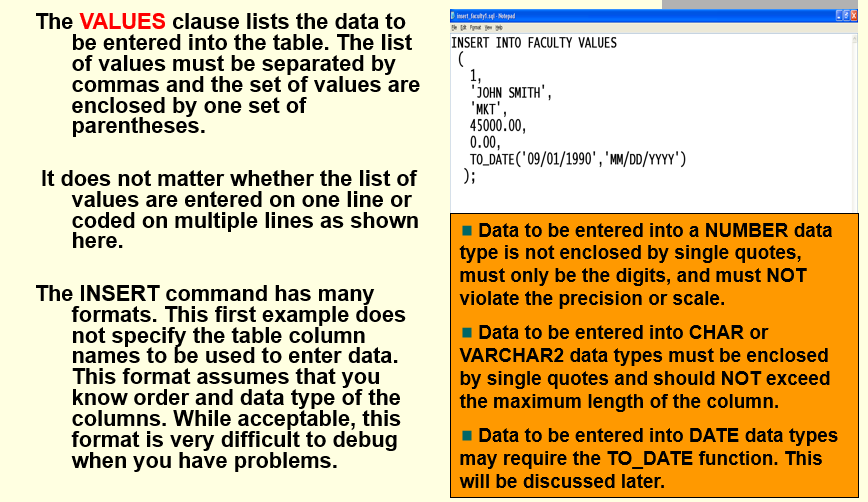
**http://www.youtube.com/watch?v=A4xTl\_xjJTc&feature=fvwrel**

### 1.1 INSERT Rules

**Insert rules -** Insert values must satisfy the following rules.

* **Default values.** The value inserted in any column that is not in the column list is the default value of the column. Columns without a default value must be included in the column list. Similarly, if you insert into a view, the default value is inserted into any column of the base table that is not included in the view. Hence, all columns of the base table that are not in the view must have a default value.
* **Length.** If the insert value of a column is a number, the column must be a numeric column with the capacity to represent the integral part of the number. If the insert value of a column is a string, the column must be either a string column with a length attribute at least as great as the length of the string, or a datetime column if the string represents a date, time, or timestamp.
* **Uniqueness constraints.** If the identified table or the base table of the identified view has one or more unique indexes, each row inserted into the table must conform to the constraints imposed by those indexes.
* **Referential constraints.** Each nonnull insert value of a foreign key must be equal to some value of the parent key of the parent table in the relationship.
* **Check constraints.** The identified table or the base table of the identified view might have one or more check constraints. Each row inserted must conform to the conditions imposed by those constraints. Thus, each check condition must be true or unknown.'

### 1.2 NSERT INTO TABLE WITHOUT the COLUMN NAMES

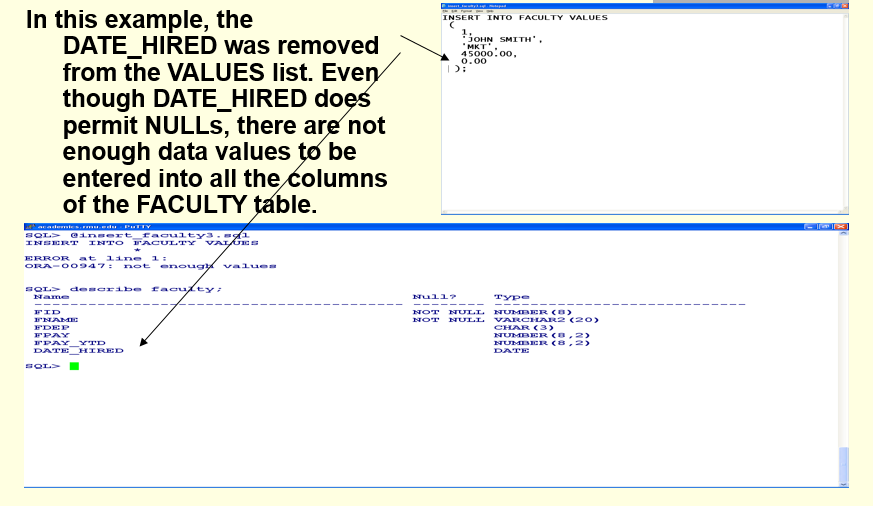


### 1.3 INSERT DANGERS - Not Specifying the Column Names

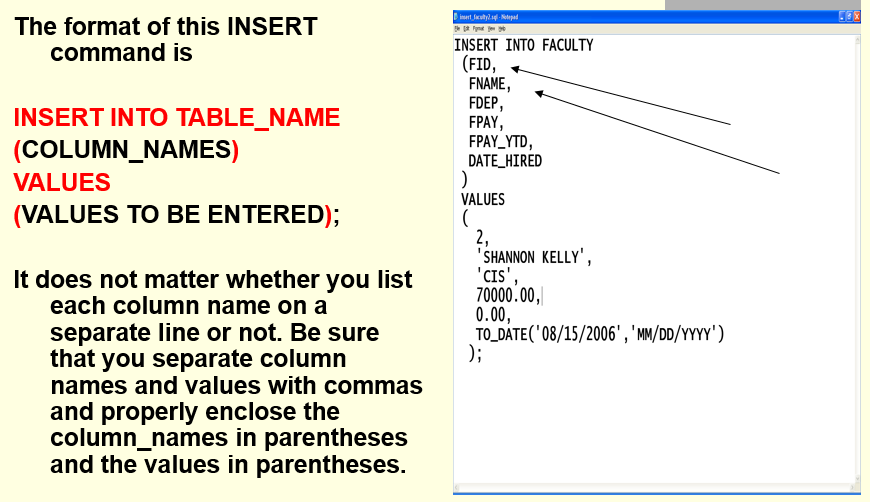
While not specifying the column names may save you coding time, the disadvantages definitely outweigh the advantages.

* The VALUES must match the exact order and data type of ALL columns of the table, even if NULLs are permitted.
* The coding of the INSERT statement may be difficult to comprehend.
* If a new column has been added to an existing table, your previously coded INSERT statement may not work.
* There is also the danger that the INSERT statement may successfully execute, but you may have inserted data into the wrong column if your order of values was incorrect**.**

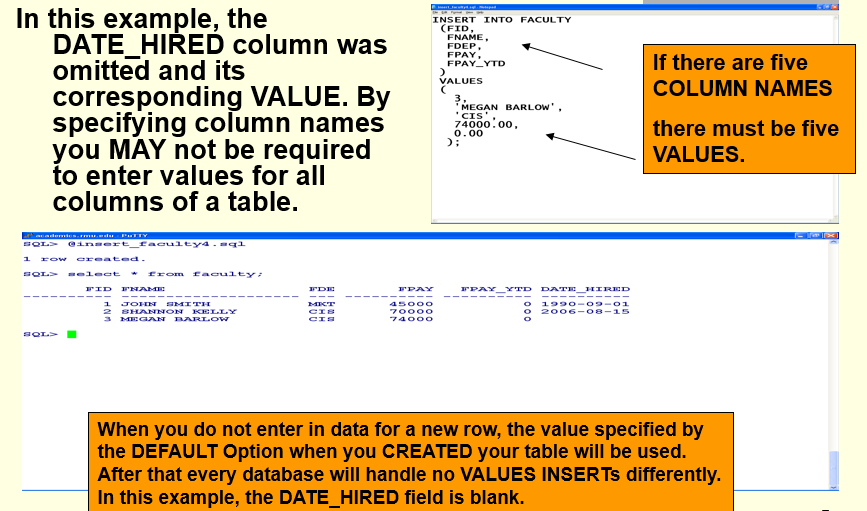
### ORA-00947 Error Not Enough Values



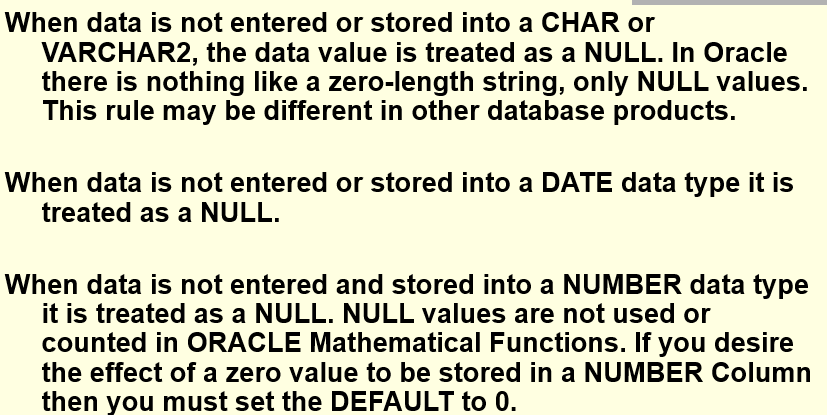
### 1.4 INSERT INTO TABLE\_NAME Specifying Column Names



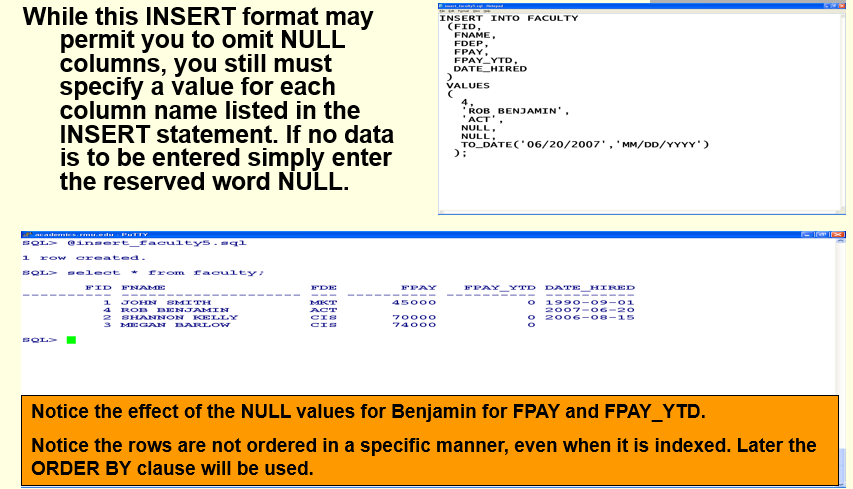
**Do I Need Values for EVERY Column?**



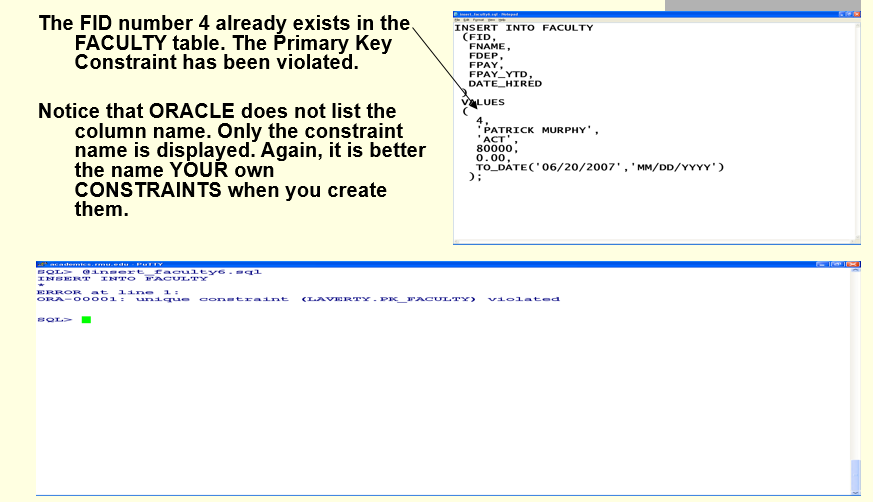
### 1.5 NO DATA VALUES MEANS NULL



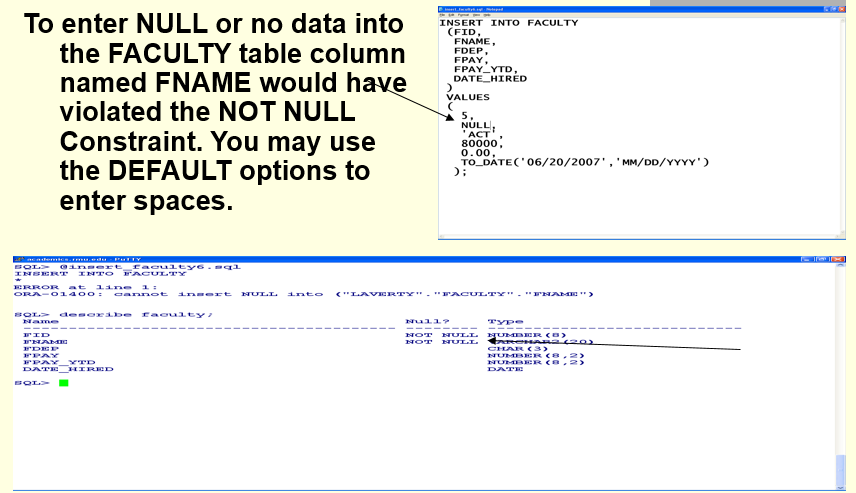
**EVERY COLUMN NAME MUST HAVE A VALUE**



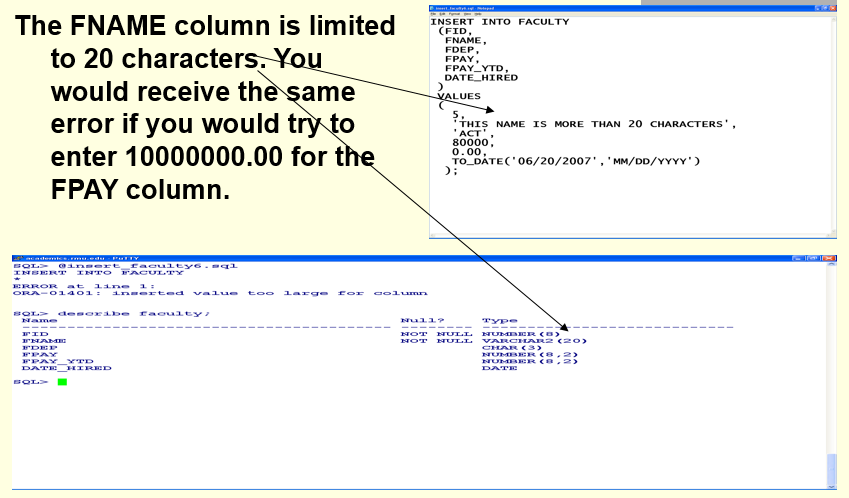
### ORA-00001 - Error Unique Constraint



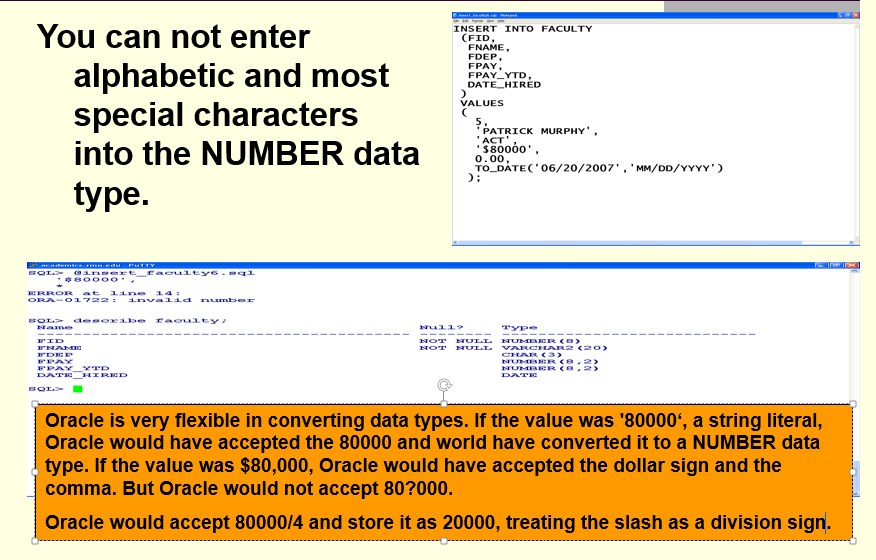
### ORA-01400 Error Cannot Insert NULL



### ORA-01401 Error Inserted Value Too Large



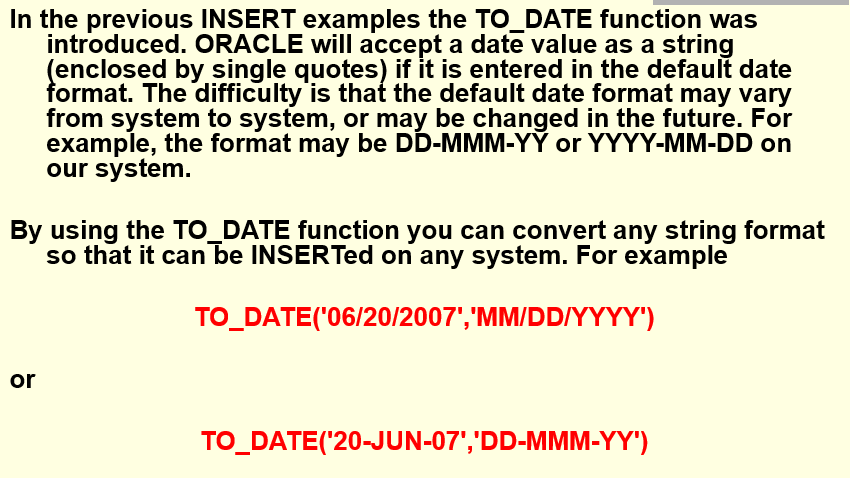
### ORA-01722 Error Invalid Number



### ORA-02290 - Error Check Constraint Violated

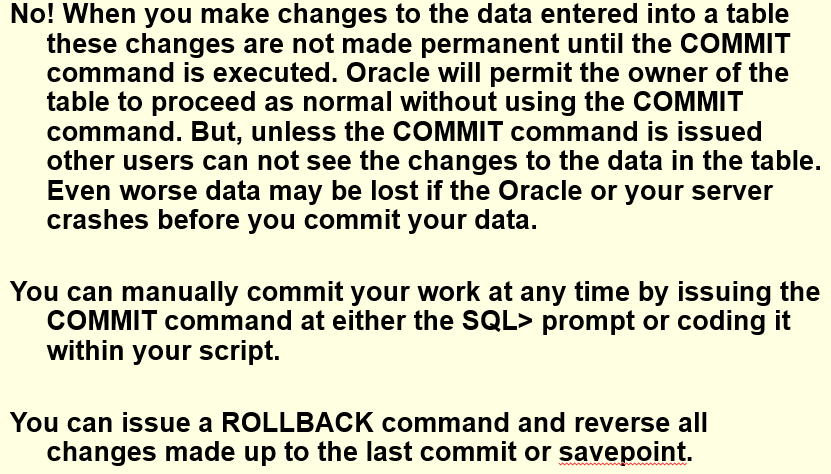


## 1.2 Inserting Dates using the TO\_DATE() Function

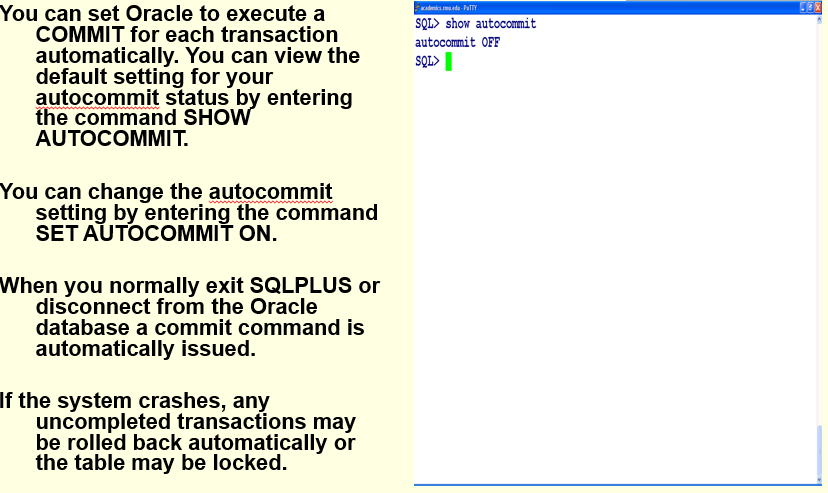


## 1.3 Commit Statement

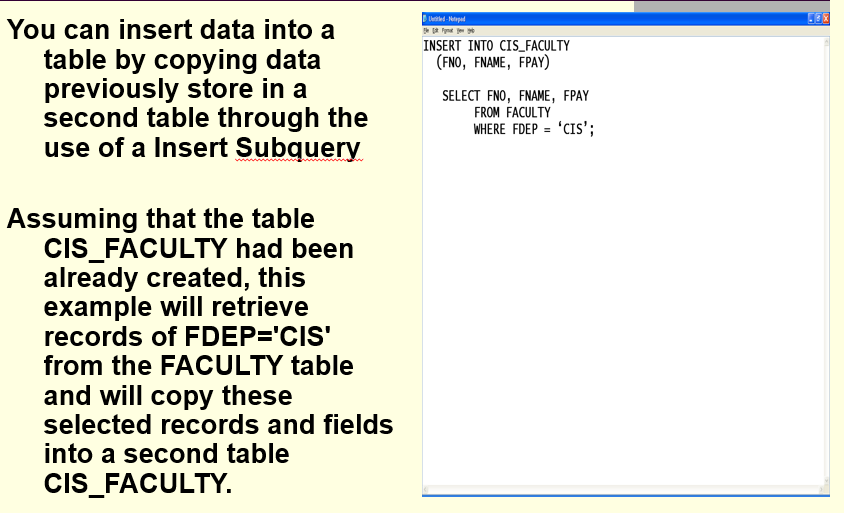
**Are We Finished Inserting Data?**



**Autocommit and Disconnect**

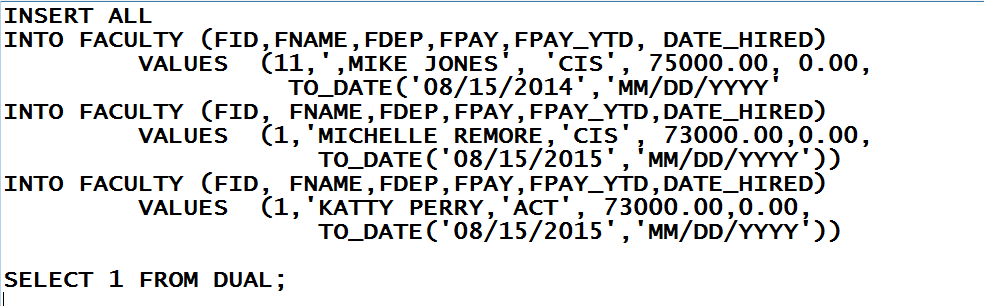


## 1.3 INSERT VIA SELECT Inserting Data from Other Tables



## 1.4 INSERT ALL

The Oracle INSERT ALL statement is used to add multiple rows with a single INSERT statement. The rows can be inserted into one table or multiple tables using only one SQL command.



The INSERT ALL uses the statement SELECT 1 FROM DUAL at the end of the statement. DUAL is a dummy table (to be discussed later) which will be use to terminate the INSERT statement by providing a trailing semicolon. The advantage of the INSERT ALL statement as compare to multiple INSERT statements is 1) it executes faster since each INSERT is not automatically committed and 2) you can insert date into two separate tables with one inset statement.

## 1.5 Creating a Sequence Object

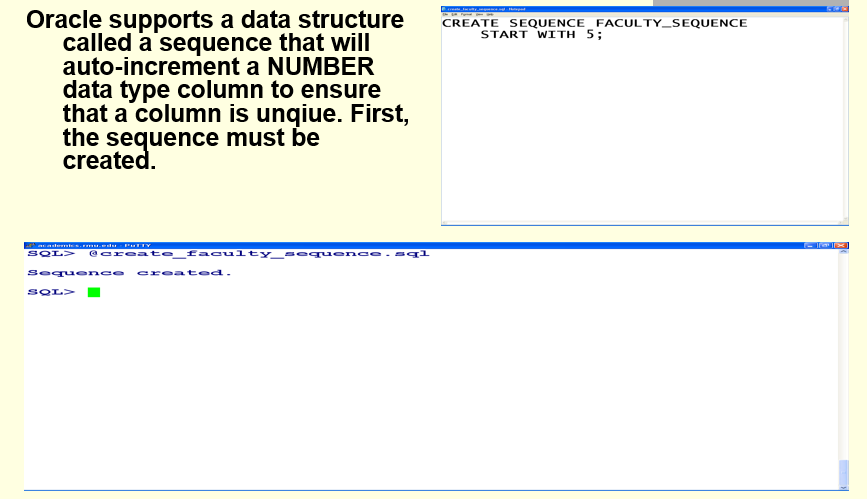
Use the CREATE SEQUENCE statement to create a sequence, which is a database object from which multiple users may generate unique integers. You can use sequences to automatically generate primary key values.

When a sequence number is generated, the sequence is incremented, independent of the transaction committing or rolling back. If two users concurrently increment the same sequence, then the sequence numbers each user acquires may have gaps, because sequence numbers are being generated by the other user. One user can never acquire the sequence number generated by another user. After a sequence value is generated by one user, that user can continue to access that value regardless of whether the sequence is incremented by another user.

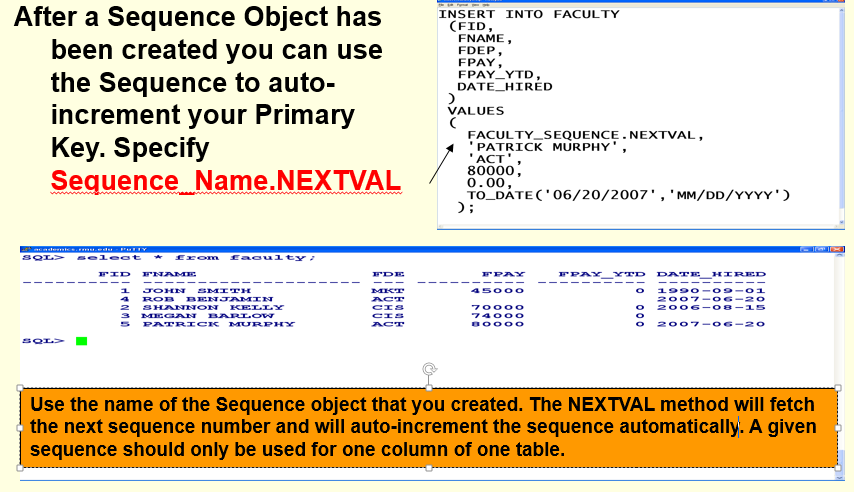
Sequence numbers are generated independently of tables, so the same sequence can be used for one or for multiple tables. It is possible that individual sequence numbers will appear to be skipped, because they were generated and used in a transaction that ultimately rolled back. Additionally, a single user may not realize that other users are drawing from the same sequence.

After a sequence is created, you can access its values in SQL statements with the CURRVAL pseudocolumn, which returns the current value of the sequence, or the NEXTVAL pseudocolumn, which increments the sequence and returns the new value.

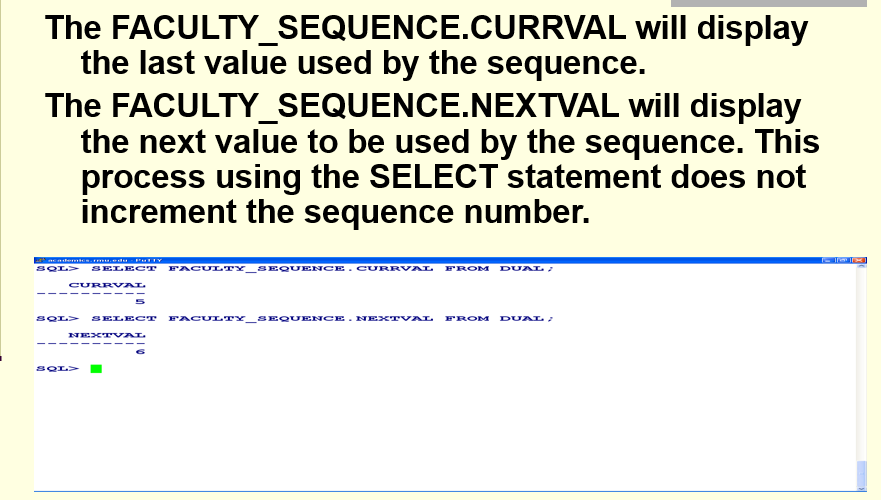
**Creating a Sequence Object**



## 1.6 Using a Sequence Object in the INSERT Statement

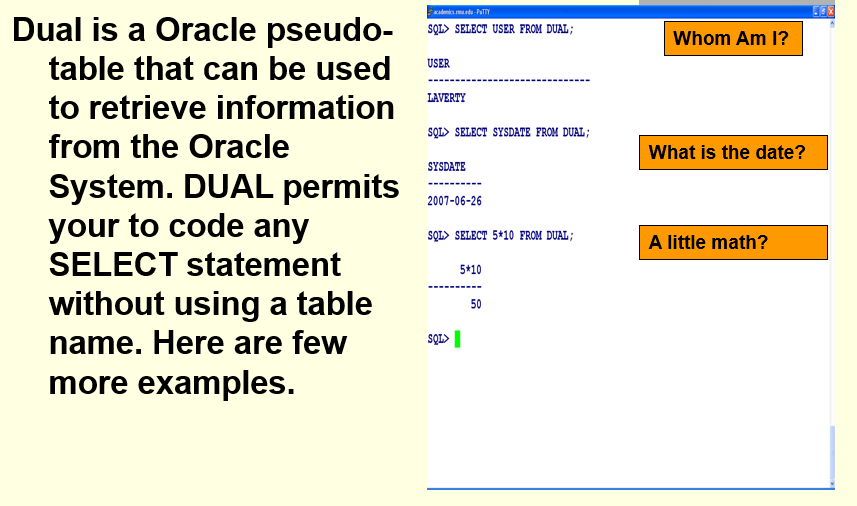


### 1.6.1 What Is My Next Sequence Number?

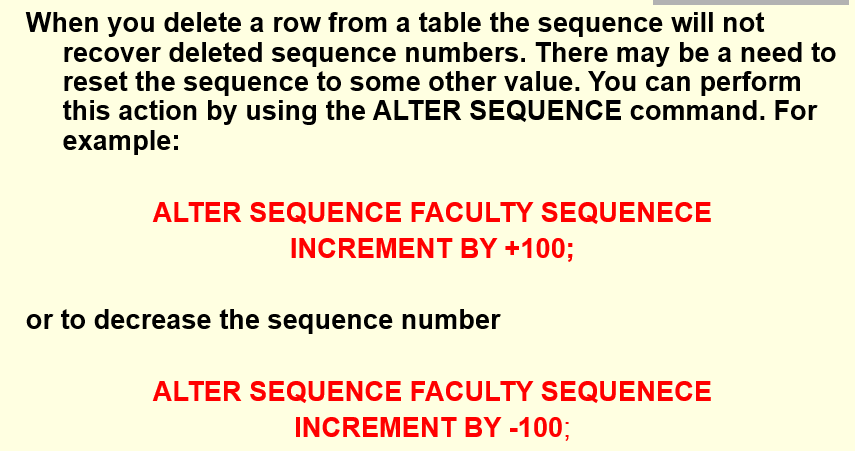


**What is DUAL?**

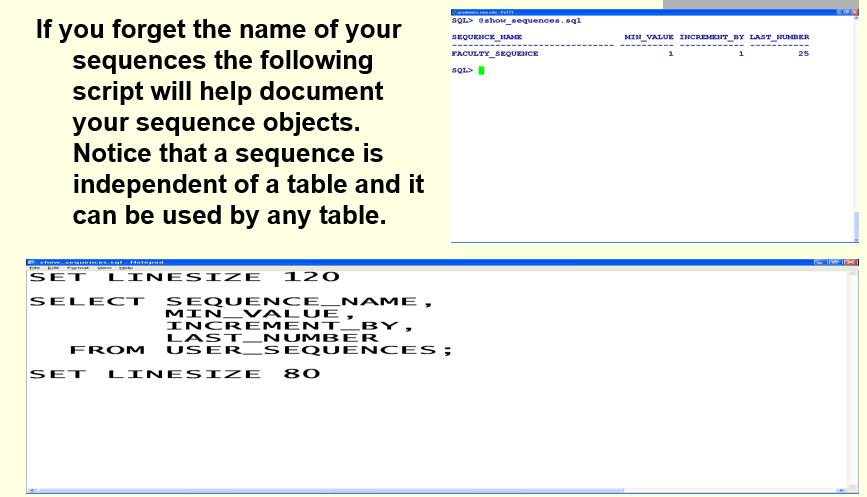
The DUAL table is a special one-row, one-column table present by default in Oracle and other database installations. In Oracle, the table has a single VARCHAR2(1) column called DUMMY that has a value of 'X'. It is suitable for use in selecting a pseudo column such as SYSDATE or USER. IBM DB2 has a view that resolves DUAL when using Oracle Compatibility



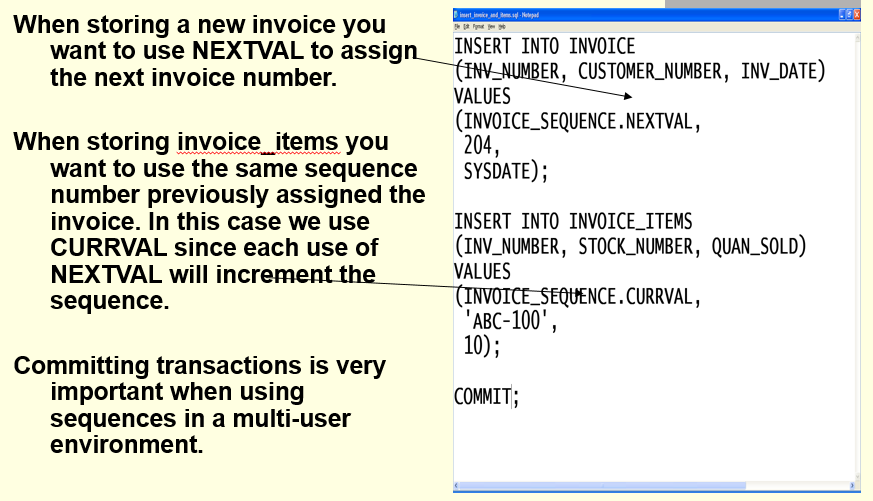
### 1.6.2 Altering a Sequence Object



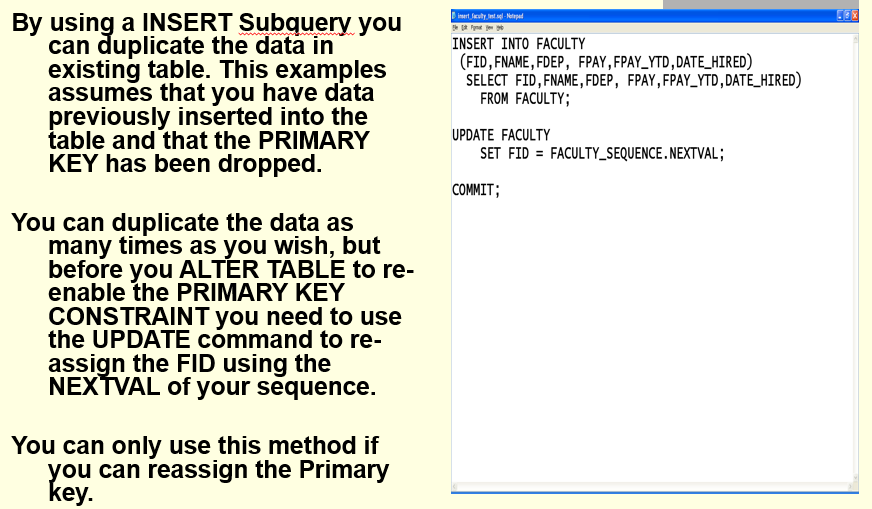
### 1.6.3 show\_sequences.sql



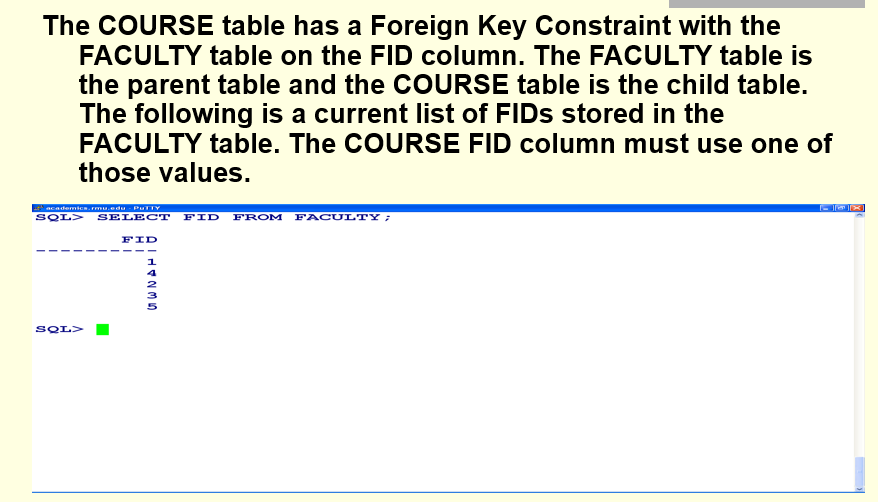
### 1.6.4 Using a Sequence Object with Two Different Tables



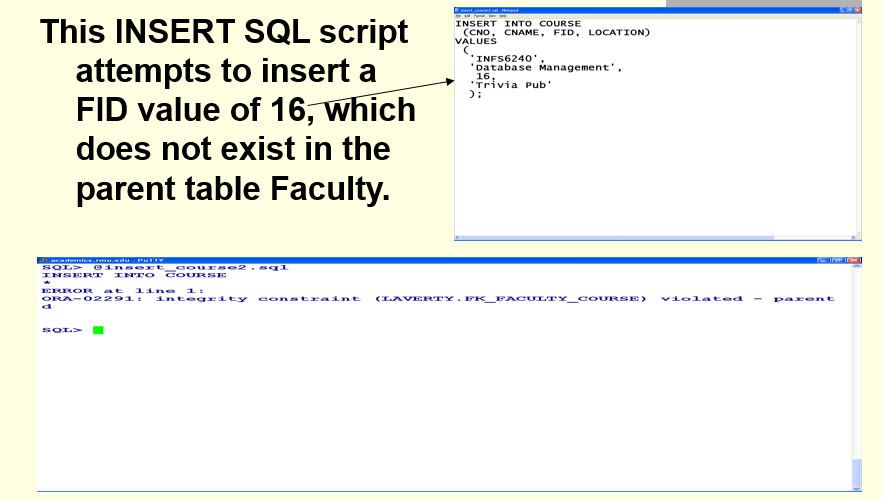
### 1.6.5 Generating Test Data Using a Sequence Object



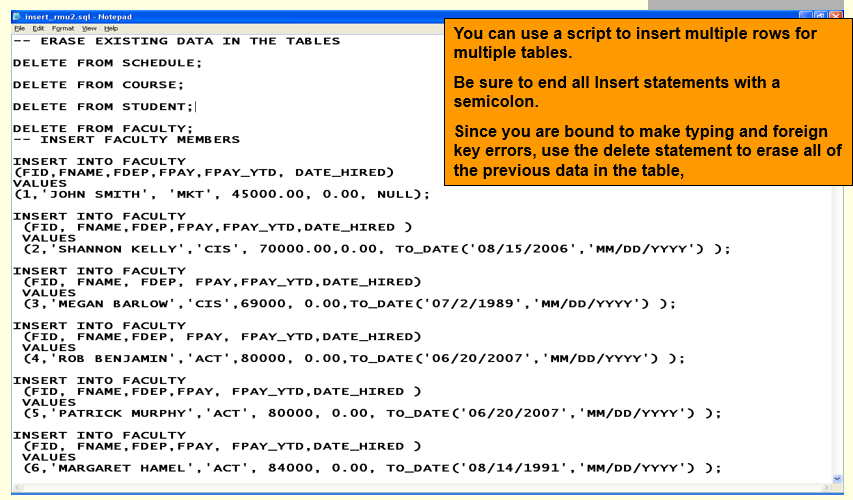
## 1.7 Foreign Key Consideration for Inserting Data into a Child Table



### ORA-02291 - Error Integrity Constraint Violated



## 1.8 insert\_rmu2.sql script



# 2.0 Update Statement

**Video - Updating data in an Oracle Table -** [**http://www.youtube.com/watch?v=9yOq6P8I8iM&feature=BFa&list=ULZ6XYBFcDmCs**](http://www.youtube.com/watch?v=9yOq6P8I8iM&feature=BFa&list=ULZ6XYBFcDmCs)

**Inserting dates into a table**

[**http://www.youtube.com/watch?v=-BxSFQZqbTA&feature=fvwrel**](http://www.youtube.com/watch?v=-BxSFQZqbTA&feature=fvwrel)

**Video SQL Update Command**

[**http://www.youtube.com/watch?v=9s40bvjgRn4&feature=fvwrel**](http://www.youtube.com/watch?v=9s40bvjgRn4&feature=fvwrel)

**Video- SQL Update Command**

[**http://www.youtube.com/watch?v=bm77\_LevpFE&feature=fvwrel**](http://www.youtube.com/watch?v=bm77_LevpFE&feature=fvwrel)

**Video Deleting data in a table -**

[**http://www.youtube.com/watch?v=UlgOsnoRP9s&feature=fvwrel**](http://www.youtube.com/watch?v=UlgOsnoRP9s&feature=fvwrel)

Video- Oracle SEQUENCES – Beginner - <https://www.youtube.com/watch?v=QJXpjrrZHFE>

Video - Oracle Sequences and Synonyms - <https://www.youtube.com/watch?v=NXgd7rGGpn4>

Video - he DUAL Table Pseudocolumns SYSDATE - <https://www.youtube.com/watch?v=cCCghdJl7WY>

Video - Oracle 11g SQL - Beginner to Professional (1z0-051) Certification Exam (Part 1) - <https://www.youtube.com/watch?v=_qNK7T4bn60>

Video - Oracle 11g SQL - Beginner to Professional (1z0-051) Certification Exam (Part 2) - <https://www.youtube.com/watch?v=sFJrJxw8YcQ>

Video - oracle date and time functions - <https://www.youtube.com/watch?v=H18UWBoHhHY>

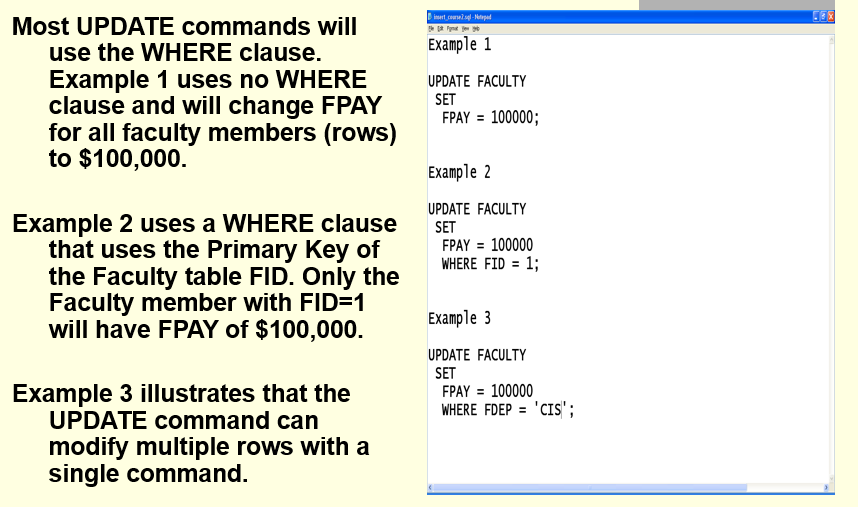
Video - Oracle SQL Video Tutorial 19 - DATE functions - <https://www.youtube.com/watch?v=JbaE6XqEQS4>

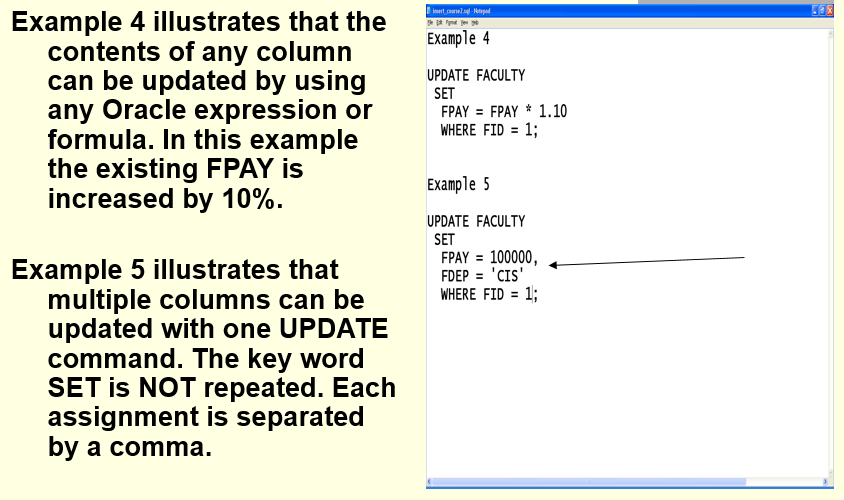
Video - Oracle SQL Video Tutorial 20 - TO\_CHAR Functions - <https://www.youtube.com/watch?v=LMSKnfbvm74>

Video - Oracle SQL Tutorial - Querying Data - Part 9 - Intro to Single-Row Functions - <https://www.youtube.com/watch?v=yjXXntX0sh8>

Video - Oracle SQL Tutorial - Querying Data - Part 10(a) - Single-Row Character Functions - <https://www.youtube.com/watch?v=P1edEqD51TI>

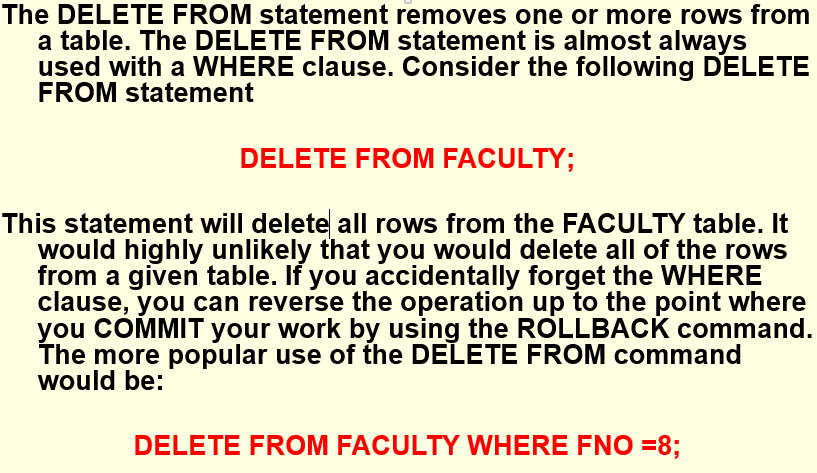
## 2.1 Updating one or more columns for one or more table rows



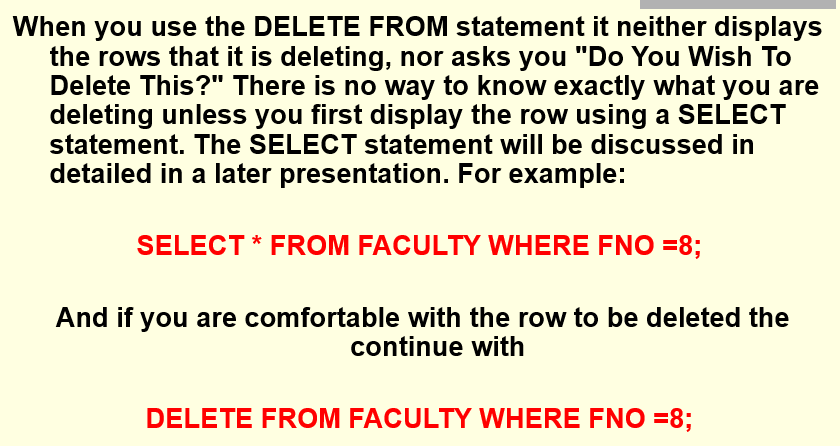


# 3.0 Delete Statement

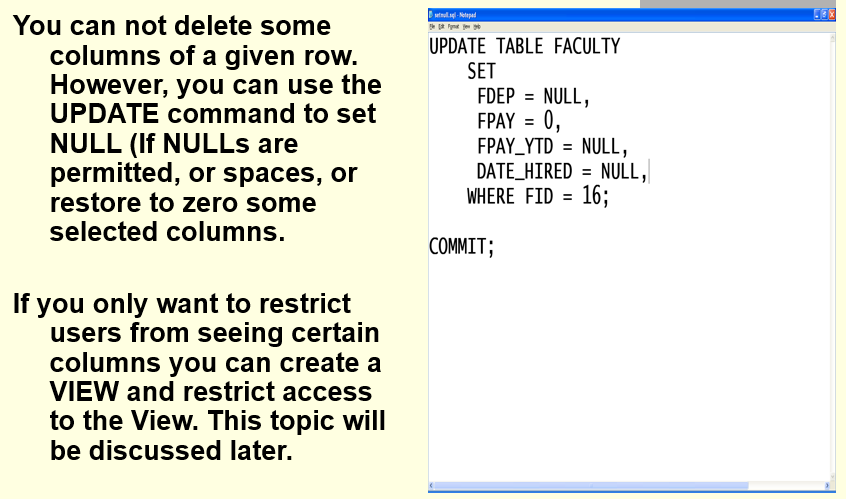
## 3.1 DELETE FROM ONE OR MORE ROWS FROM A TABLE



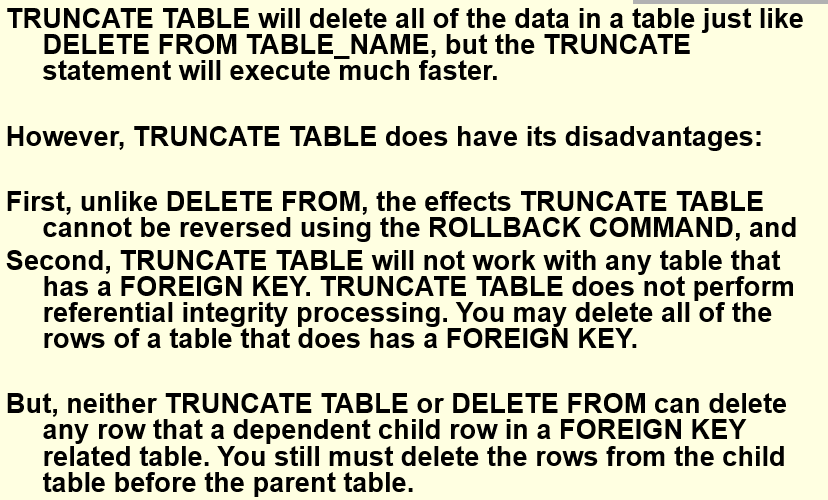
## 3.2 Do You Know What You Are Deleting?



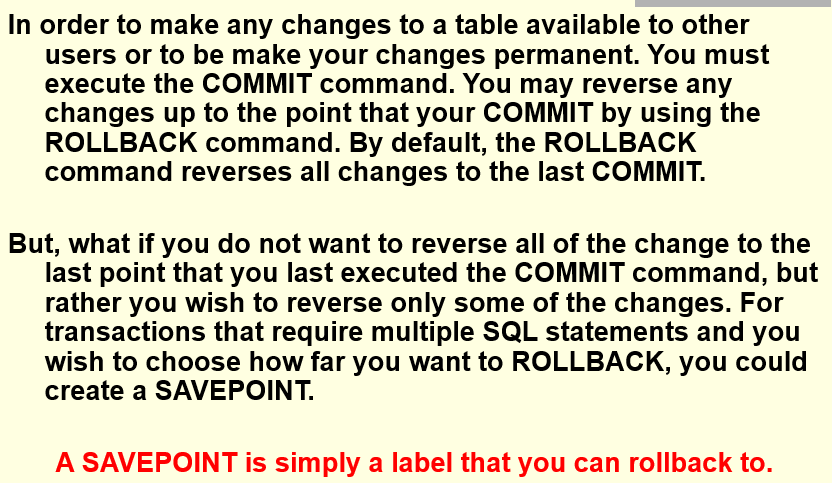
## 3.3 Do Not Want to Delete the Entire Row.



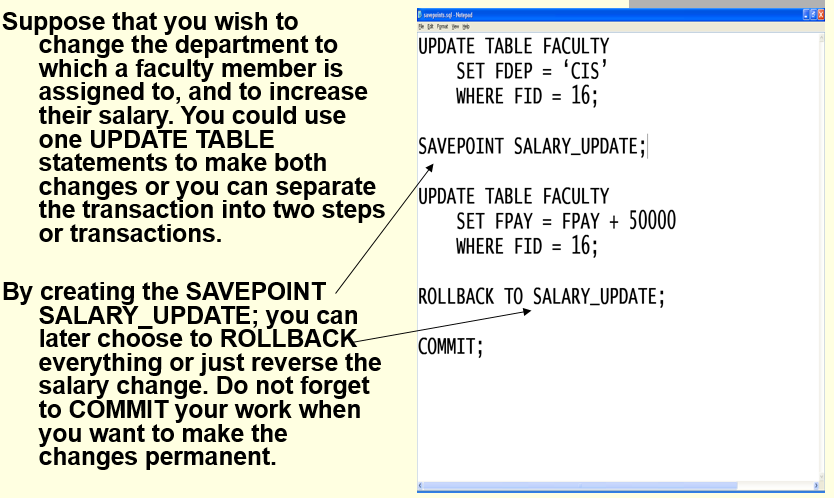
## 3.4 Comparing TRUNCATE TABLE and DELETE FROM



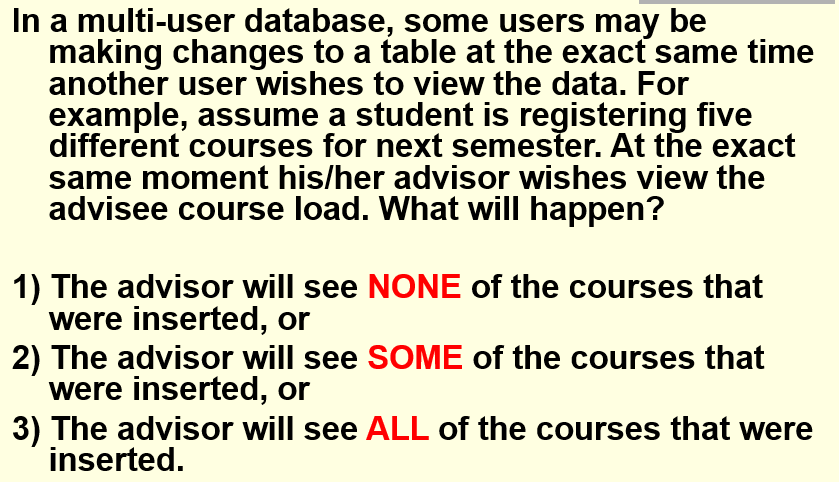
# 4.0 COMMIT, ROLLBACK AND SAVEPOINTS



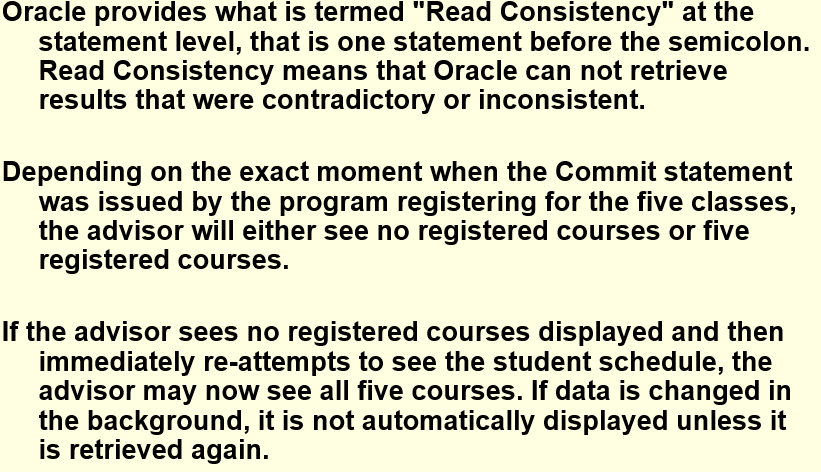
## 4.1 SAVEPOINT EXAMPLE



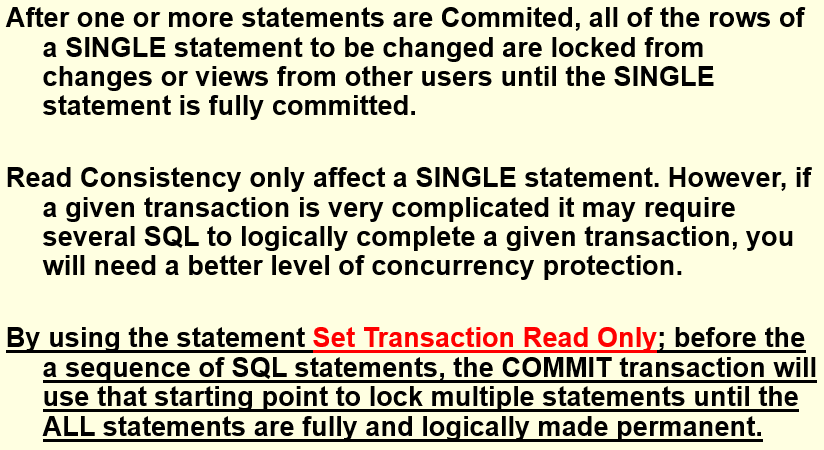
## 4.2 CONCURRENCY and Multi-user Issues



## 4.3 ORACLE Read Consistency



## 4.4 Set Transaction Read Only



# 5.0 Required Professional Appearance of Outputted Results

While you may have coded a SQL statement that meets the requirement objectives, your overall grade will be slightly decreased if your documented results are not professional in appearance. You are required to demonstrate your ability to apply SQL\*Plus line and page formatting, PuTTY formatting, and SQL formatting, e.g., SQL SUBSTR() and TO\_CHAR() Function and Columns Aliases (AS).

This does not mean that you should waste excessive time formatting and not completing the assignment. While you will not be penalized for each and every improperly formatted result requirement, the bottom line is if the instructor cannot read your SQL statement and printed result you will receive NO CREDIT.

On the other hand, you may ask, "How picky will the instructor in grading for professional appearance?" Properly applying basic SQL\*Plus, Putty, and SPL formatting is important to some degree and the instructor will not "nit pick" if you have demonstrated that knowledge. The more important issue is what do you have to show a potential employer that you can?

"Design, implement, insert test data, and test a simple academic database and manufacturing cost accounting data base". (Sound like a good line to add to your resume!)

Better yet. Bind all of your final professionally prepared copies of this course assignments into a folder and take it to your internship of future job interview. It does not matter whether or not that the interview involves a database. What matters is that you can professionally apply and document any type of computer technology if given the opportunity. Don't just prepare these course assignments for a final grade, prepare them for a better and financially rewarding career opportunity.

## 5.1 Improving the Professional Appearance of the SQL Statement and Results using SQL\*Plus

The SQL\*Plus default LINESIZE is 80 characters. Your output line may exceed 80 characters and then wrap to the next line this making your documentation unreadable. Increase the LINESIZE at the SQL> prompt or by adding the following in your script, e.g., SET LINESIZE 130 [this is a SQL\*Plus command, not a SQL statement, and should not end with a semicolon

Setting the LINESIZE does not affect the font size in Putty. By default SQL\*Plus display 80 characters on a line not matter what the font size is in Putty. SQL\*Plus will force the display of the result of the SELECT statement is wrap to the next line at 80 characters. Increasing the SQL\*Plus LINESIZE may still be wrapped by Putty. You can decrease the font size in Putty.

## 5.2 Improving the Professional Appearance of the SQL Statement and Results using PuTTY

**Change the point size (or font) under Window/Appearance**. Click CHANGE under Font Settings. A point size of 14 that is bolded is easier to read, but you may get word wrap problems. Clicking on OK only affects the current session.



**Video - Formatting SQL\*Plus Output -** [**https://www.youtube.com/watch?v=\_46MeUAFDO4**](https://www.youtube.com/watch?v=_46MeUAFDO4)

**Video - How-to: Change the PAGESIZE and LINESIZE in SQL\*Plus -** [**https://www.youtube.com/watch?v=hSQFcRt8kqI**](https://www.youtube.com/watch?v=hSQFcRt8kqI)

## 5.3 Using the SQL SUBSTR() and TO\_CHAR() Function and Columns Aliases (AS)

Similar to the previous CREATE TABLEs, start simple and then add more features.

**Step 1.** Start by listing columns those columns that are to be displayed and FROM which table stores those columns?

Use the SQL Script method or type the following example at the SQL> prompt and execute.

**SELECT CUSTOMER\_NAME,**

**CUSTOMER\_NUMBER,**

**CUSTOMER\_TYPE,**

**ATTENTION\_NAME**

**FROM CUSTOMER;**

**SELECT JOBCOST\_REFERENCE\_NUMBER,**

**PACKLIST\_NUMBER,**

**TRANSACTION\_DATE,**

**TRANSACTION CODE,**

**JOBCOST\_UNITS \* COST\_PER\_UNIT**

**FROM JOB\_COST;**

Notice that each previous example coded each column on a separate line and a little indentation was used to mage the SQL code easier to read.

## 5.4 SUBSTR() Function

**Step 2.** Format the column width using the SUBSTR() function. Students frequently make column names or the width of the CHAR data types to wide. For example, you may have decided to create the CUSTOMER\_NAME column as CHAR(25). Depending on the business requirements a column width of 25 characters may be acceptable or even recommended. But, if you display five columns, each with a column width of 25 characters, then the line will be a least 125 characters wide and the white space will make the output unreadable.

Therefore, it is recommended to use the SUBSTR() function to reduce the number of characters displayed, which will have no effect on the number of characters stored. see (<http://www.techonthenet.com/oracle/functions/substr.php>)

Video - Oracle Database11g tutorials 13 || SQL substr function / SQL substring function - <https://www.youtube.com/watch?v=f52uKYyYFoU>

Use the SQL Script method or type the following example at the SQL> prompt and execute.

SELECT **SUBSTR(CUSTOMER\_NAME, 1, 10),**

CUSTOMER\_NUMBER,

CUSTOMER\_TYPE,

**SUBSTR(ATTENTION\_NAME, 1, 10)**

FROM CUSTOMER;

SELECT **SUBSTR(JOBCOST\_REFERENCE\_NUMBER, 1, 6),**

**SUBSTR(PACKLIST\_NUMBER, 1, 6),**

**SUBSTR(TRANSACTION\_DATE, 1, 10),**

**SUBSTR(TRANSACTION CODE, 1, 2),**

JOBCOST\_UNITS \* COST\_PER\_UNIT

FROM JOB\_COST;

## 5.5 Documents Requirements

The requirements for this script are to

* Insert two (2) faculty rows using a faculty name that are members of your family. You may enter other data as you may determine to be appropriate.
* Insert two (2) course rows using any data as you may determine to be appropriate. According to the rules of referential integrity you may not enter a faculty id, i.e., FID, that was not previously entered into the faculty table.
* Insert two (2) student rows using student names that are members of your family. You may enter other data as you may determine to be appropriate.
* Insert two (2) schedule rows. According to the rules of referential integrity you may not enter a course no, i.e., CNO, or a student number, i.e., SNO, that were not previously entered.

**Use a graphical snipping tool to document an image of these requirements below. Make sure your Putty banner is displayed.**

Document the data the data that you have entered into your table by using the previous INSERT statements. The Select \* from Faculty; SQL statement will display all columns and all rows of data entered in FACULTY table. Enter each Alt-Prtscn below:

1. **Enter the SQLPLUS command Clear Screen. Execute and insert and image for your SELECT \* FROM FACULTY; to document your test data below:**

Answer =>

**2. Enter the SQLPLUS command Clear Screen. Execute and insert and image for your SELECT \* FROM COURSE; to document your test data below:**

Answer =>

**3. Enter the SQLPLUS command Clear Screen. Execute and insert and image for your SELECT \* FROM STUDENT; to document your test data below:**

Answer =>

**4. Enter the SQLPLUS command Clear Screen. Execute and insert and image for your SELECT \* FROM SCHEDULE; to document your test data below:**

Answer =>

# 6.0 PSP Case Database

## PSP Tables

Create the following tables (shaded name) and columns form the PSP case. The table names and column names do not follow the rules of naming object in Oracle, e.g., Oracle objects are not permitted to have embedded spaces. Each column name must be assigned an appropriate date types. Details on Oracle data types are presented after the list.

A column's data type specifies the 1) the type of information that may be stored in a column, and the operations that may be performed. Oracle supports five data types, Character, Numeric, Temporal (Date/Time), Large Object (LOB), and Row ID (not discussed). These details are provided after the PSP table list.

**CUSTOMER**

**CUSTOMER NAME**

**CUSTOMER NUMBER**

**CUSTOMER TYPE**

**ATTENTION NAME**

**ADDRESS 1 BILL**

**ADDRESS 2 BILL**

**CITY BILL**

**STATE BILL**

**ZIPCODE BILL**

**COUNTRY BILL**

**DRAWING**

**DRAWING NUMBER**

**DRAWING PREFIX**

**REVISION LEVEL**

**DATE LAST REVISED**

**CUSTOMER NUMBER**

**CUSTOMER PART NUMBER**

**PRIMARY RAW MATERIAL NUMBER**

**OD**

**WALL**

**LENGH**

**CUT LENGTH**

**RAW MATERIAL PER UNIT**

**GL\_NUMBER**

**OPTION**

**OPTION NUMBER**

**OPTION TYPE**

**OPTION DESCRIPITON**

**OPTION UNIT COST**

**OPTION UNIT PRICE**

**OPTION PRECENTAGE**

**DRAWING OPTION**

**DRAWING NUMBER**

**OPTION NUMBER**

**UNITS-OF-OPTION**

**PRICE LIST**

**OD**

**WALL**

**LENGH**

**TYPE**

**PRICE 0 TO 5**

**PRICE 6 TO 25**

**PRICE 26 TO 50**

**PRICE 50 TO 100**

**PRICE 101 TO 250**

**PRICE OVER 250**

**RAW MATERIAL**

**RAW MATERIAL ID**

**RAW MATERIAL TYPE**

**CURRENT UNIT COST**

**RAW MATERIAL REQUIREMENTS**

**PACKLIST NUMBER {common column}**

**RAW MATERIAL ID**

**SCHEDULED SHIP DATE**

**UNITS REQUIRED**

**VENDOR**

**VENDOR NUMBER**

**VENDOR NAME**

**LABOR OPERATION**

**OPERATION NUMBER**

**OPERATION DESCRIPTION**

**SCHEDULE CATAGORY**

**PACKLIST**

**PACKLIST NUMBER**

**DRAWING NUMBER**

**REVISION LEVEL**

**CUSTOMER NUMBER**

**CUSTOMER PART NUMBER**

**CUSTOMER PURCHASE ORDER NUMBER**

**ORDER BY**

**DATE ORDERED**

**SCHEDULED SHIP DATE**

**ORGIINAL SHIP DATE**

**ACTUAL SHIPPED DATE**

**ORDER ACKNOWLEDGENT DATE**

**UNITS**

**PRICE PER UNIT**

**GROUP BY**

**QUANTITY SHIPPED**

**CUSTOMER SHIPPING ID**

**SATISFIED**

**JOBCOST**

**JOBCOST REFERENCE NUMBER**

**PACKLIST NUMBER**

**TRANSACTION DATE**

**TRANSACTION CODE**

**TRANSACTION CROSS REFERENCE**

**CLOCK NUMBER**

**JOBCOST UNITS**

**COST PER UNIT**

## 6.1 General Oracle name rules

Oracle objects include tables, views, columns, and indexes. They follow these naming conventions.

* These objects must have names of the following length in characters: column (1–30), index (1–18), **table (1–18)**, view (1–18), alias (1–18), synonym (1–18), or correlation (1–128).
* A name must begin with a letter. If the name is in quotation marks, it can start with and contain any character. Depending on how your string delimiter is set, quoted strings can contain quotation marks such as “O'Malley”.
* A name can contain the quotation marks ("), 0 through 9, ! % ( ) { } . - ^ ~ \_ (underscore) @, #, $, and space, and \ (backslash). There are other special characters that might work separately depending on your operating system and where you are working with the DB2® database. However, while they might work, there is no guarantee that they will work. It is not recommended that you use these other special characters when naming objects in your database.
* Microsoft Access permits embedded spaces in table names. But, if you plan to code your own SQL statements with objects and identifier names that contain spaces, you must include these table names in square brackets which can be both a typing and coding nightmare, e.g., [COLLEGE FACULTY]. Columns name follow the same rules except that a column name may be reused in a different table.
* The letters A through Z, and a through z, as defined in the basic (7-bit) ASCII character set. When used in identifiers for objects created with SQL statements, lowercase characters "a" through "z" are converted to uppercase unless they are delimited with
* Names are not case sensitive. For example, CUSTOMER and Customer are the same. However, if the name of the object is in quotation marks, it is case sensitive.
* A name cannot be a reserved word in DB2, e.g., CREATE, INSERT, SELECT.
* A name cannot be the same as another DB2 object within the same schema, table, or view. For example, each column name within the same table must be unique.
* Rules exist for the naming of all database objects, user names, passwords, groups, files, and paths. Some of these rules are specific to the platform you are working on.

## 6.2 Oracle Data Types

A column's data type specifies the 1) the type of information that may be stored in a column, and the operations that may be performed. Oracle supports five data types, Character, Numeric, Temporal, Large Object (LOB), and Row ID (not discussed)

**Video - SQL Data Types -** [**https://www.youtube.com/watch?v=AFzXa7wO6iY**](https://www.youtube.com/watch?v=AFzXa7wO6iY)

**Video- Oracle - Datatypes -** [**https://www.youtube.com/watch?v=Y9psXcPA-6w**](https://www.youtube.com/watch?v=Y9psXcPA-6w)

**Video - Class - 01 DataTypes in ORACLE** [**https://www.youtube.com/watch?v=dO8QHTCw\_sg**](https://www.youtube.com/watch?v=dO8QHTCw_sg)

**Video - Oracle 10g new Data types -** [**https://www.youtube.com/watch?v=9JHNkByQFvY**](https://www.youtube.com/watch?v=9JHNkByQFvY)

## 6.3 Character Data Types

Oracle supports several three types of string data: standard 8-bit fixed length, fixed length Unicode *character strings, and variable-length ANSI or Unicode strings. IBM DB2 o*

* Fixed-length character strings contain text and can be either a fixed-length or a varying-length. Uses 8-bit encoding that support traditional ASCII and ECBDIC (IBM Mainframe) character sets.
* Fixed-length Unicode strings will used 8-bit, 16-biyt, 32-bits or other encoding schemes to store one character. A Unicode character generally may use the latin character set such as [a-zA-Z]. or other characters such as ñéóíúá, and characters in German, French, etc. character sets. Unicode Graphic characters are more about the ones in other languages, such as Mandarin (Chinese), Hebrew, Arab, Japanese, and even characters in Braile. They are very different from any latin character, Hebrew and Arab are written right to left, and Mandarin is up-down, and because of this they are like graphics for the West world (which is based on latin chars.)

What is the difference between an encoding scheme and a character set? An encoding scheme indicates how many bytes are used to store the binary representation of a single character. ASCII and EBCDIC use 8- bits, or one byte, to store one character. This mean one can store up to 256 combinations of characters. On the other hand Unicode uses 16-bits, or a possible combination 65,556 characters. Chinese, Hebrew, Japanese or Arabic may store their characters in 16-bit length Unicode, but they need to be displayed differently. This is the function of the character set. The character set represents of each of the stored combinations of characters are DISPLAYED, while the encoding scheme determines how of each of the stored combinations of characters are STORED in a column.

* Variable-length data types may store multiple ASCII or UNICODE either ASCII strings contain strings of binary bytes and can be either a fixed-length or a varying-length. All of these types of string data can be represented as large objects. A binary string column is used for storing non-character data, such as encoded or compressed data, pictures, voice, and mixed media. A binary string column can also hold structured data for use by distinct types, user-defined functions, and stored procedures. The BINARY function returns a BINARY (fixed-length binary string) representation of a string of any type or a row ID type.

Multinational companies that engage in international trade often store data from more than one country in the same table. Some countries use different coded character set identifiers. Oracle supports the Unicode encoding scheme, which represents many different geographies and languages. If you need to perform character conversion on Unicode data, the conversion is more likely to preserve all of your information.

In some cases, you might need to convert characters to a different encoding representation. The process of conversion is known as character conversion. Most users do not need a knowledge of character conversion. When character conversion does occur, it does so automatically and a successful conversion is invisible to the application and users.

Oracle data types are compatible with the encoding schemes used by DB2, ANSI and other program language data types. COBOL data types are provided in the following tables. You are required to use Oracle data types when coding ORACLE columns.

|  |  |  |
| --- | --- | --- |
| **Oracle Character Types** | | |
| **Oracle Data Type** | **Compatible DB2, ANSI and COBOL Data Types** | **Description** |
| **CHAR(n)** | **CHARACTER(n)**  **CHAR(n)**  03 char-field1 pic x(5).  03 char-field2 pic x(254). | Fixed-length character strings with a length of n bytes. n must be greater than 0 and not greater than 2000. The default length is 1. Uses 8-bit encoding. |
| **VARCHAR(Size)** | **CHAR VARYING (N)**  **VARCHAR2(n)**  firstname varchar(12)  03 firstname.  49 firstname-len pic 9(4) comp.  49 firstname-data pic x(12). | Varying-length character strings with a maximum length of n bytes. Size must be greater than 0 and less than a 4000. However the limit may be increased to a maximum length of 32704 8-bit characters. |
| **NCHAR(Size)** | **NATIONAL CHARACTER (n)**  **NATIONAL CHAR (n)**  **GRAPHIC(n)**  01 name PIC G(n) DISPLAY-1. | Fixed-length Unicode characters where size is the number of characters. Size must be greater than 0 with an upper limit of 2000 bytes. Unlike VARCHAR2 this is the length of bytes, not the number of characters stored. If the character set uses 2-bytes and the length is 2000 bytes, then only 1000 Unicode characters may be stored in 2000 bytes. The following is an example of a. fixed-length Unicode field which will store 2000 characters using the 16-bit Japanese character set. If this column would be stored store on a EBCDIC mainframe the character set name would be JA16DBCS (http://www.daylight.com/meetings/emug04/Delany/charsets.html)  MY\_UNICODE\_COLUMN NCHAR(2000)  CHARACTER SET JA16VMS |
| **NCHAR2(size)** | **NATIONAL CHARACTER VARYING (n)**  **NATIONAL CHAR VARYING (n)** | Fixed-length Unicode characters where size is the number of characters. Size must be greater than 0 with an upper limit of 4000 bytes. If the character set uses 2-bytes and the length is 4000 bytes, then only 2000 Unicode characters may be stored in 4000 bytes. |
| Both VARCHAR2 and NVARCHAR2 data types may increase the maximum size to 32,767 bytes if the parameter EXTENDED is used for the MAX\_STRING\_SIZE in the init.ora startup file when using Oracle 12c | | |

## 6.4 Oracle LOB Data Types

|  |  |  |
| --- | --- | --- |
| **Oracle LOB Data Types** | | |
| Large object data types can store text, XML, Word documents, PSFS, image, audio, video files, or other unstructured data | | |
| **Oracle Data Type** | **Compatible DB2, ANSI and COBOL Data Types** | **Description** |
| **CLOB** |  | Character Large Object Stores up to 8 gigabytes of 8-bit ASCII character (text) or binary data. |
| **NCLOB** |  | National Character Large Object Stores up to 8 gigabytes of Unicode character (text) or binary data. |
| **BLOB** |  | Binary Large Object Stores up to 8 gigabytes of unstructured data binary data. |
| **BFILE** |  | Binary file. Stores a pointer to a large binary file outside the database in the file system of the host computer. |
| **RAW(size)** |  | Oracle 8 old data type. Stores up to 2000 bytes of binary data that is not intended to be converted when moving between different versions of Oracle. |
| **LONG** |  | Oracle 8 old data type. Stores up to 2 gigabytes of character data. |
| **LONG RAW** |  | Oracle 8 old data type. Stores up to 2 gigabytes bytes of binary data that is not intended to be converted when moving between different versions of Oracle. |

## 6.5 Oracle Numeric Data Types

|  |  |  |
| --- | --- | --- |
| **Oracle Numeric Data Types** | | |
| **Oracle Data Type** | **Compatible DB2, ANSI and COBOL Data Types** | **Description** |
|  | 01 name PIC S9(4) COMP-5. | Small integers. A small integer is binary integer with a precision of 15 bits. The range is -32768 to +32767. |
| **NUMBER(precision, scale)** | DECIMAL(length,position)  NUMERIC(length,position)  01 name PIC S9(m)V9(n) COMP-3. | Stores fixed decimal point numbers. For example, DECIMAL(7,3) will permit up to 7 digits and there will be 3 decimal positions. Precision is the maximum number digits stored, and sale is the placement of the decimal point |
| **NUMBER (precision)** | **SMALLINT (8 bits)**  01 name PIC S9(4) COMP-5.  **INTEGER or INT (16 bits)**  01 name PIC S9(9) COMP-5. | Stores integer numbers. The precision can range from 1 to 38 |
| **FLOAT (Precision)** | FLOAT (126) is double precision (16 significant digits)  FLOAT (63) is single precision (7 significant digits) | Stores a floating-point number, unlike fixed decimal point numbers, A FLOAT values are stored as a number, but has less of a significant number of significant digits as a result FLOAT does not always provide for exact values. |
| **BINARY FLOAT** | REAL  01 name USAGE IS COMP-1.  Similar to the datatypes FLOAT in Java | Stores single-precision, 32-bit, floating numbers; cannot store integers. Stores approximate representations of decimal values, rather than exact representations. |
| **BINARY DOUBLE** | DOUBLE  01 name USAGE IS COMP-2.  Similar to the datatypes DOUBLE in Java | Stores DOUBLE-precision, 64-bit, floating numbers; ; cannot store integers. Stores approximate representations of decimal values, rather than exact representations. |

## 6.6 Oracle Date /Time Data Types

Although storing dates and times as numeric values is possible, using datetime data types is recommended

|  |  |  |
| --- | --- | --- |
| **Oracle Date /Time Data Types** | | |
| **Oracle Data Type** | **Compatible DB2, ANSI and COBOL Data Types** | **Description** |
| **DATE** | 01 identifier PIC X(7). | Stores the year (including the century), the month, the day, the hours, the minutes, and the seconds (after midnight). The date data type does not store fractional second or time zone.  Oracle Database uses its own internal format to store dates. Date data is stored in fixed-length fields of seven bytes each, corresponding to century, year, month, day, hour, minute, and second.  For input and output of dates, the standard Oracle date format is DD-MON-YY, as follows: '13-JUL-15'. You can change this default date format for an instance with the parameter NLS\_DATE\_FORMAT.  It is typically good for representing data for when something has happened or should happen in the future. The problem with the DATE datatype is its' granularity when trying to determine a time interval between two events when the events happen within a second of each other.  Use the TO\_CHAR() function to format the date when displayed.  Uses the TO\_DATE() to format the date to be stored. |
| **TIMESTAMP** | 01 identifier PIC X(26). | TIMESTAMP datatype which stores all the information that the DATE datatype stores, but also includes fractional seconds. he size is 7 or 11 bytes, depending on the precision. This data type contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. It contains fractional seconds but does not have a time zone. |
| **TIMESTAMP [(fractional\_seconds\_precision)] WITH TIME ZONE** |  | All values of TIMESTAMP as well as time zone displacement value, where fractional\_seconds\_precision is the number of digits in the fractional part of the SECOND datetime field. |
| **TIMESTAMP [(fractional\_seconds\_precision)] WITH local TIME ZONE** |  | Data is normalized to the database time zone when it is stored in the database.  When the data is retrieved, users see the data in the session time zone. |
| **INTERVAL YEAR [(year\_precision)] TO MONTH** |  | Makes it easier to work with time intervals such as 2 days, 2 hours and 2 minutes. |
| **INTERVAL DAY [(day\_precision)] TO SECOND [(fractional\_seconds\_precision)]** |  |  |

## Step 1. CODE AND INCREMENTALLY PSP TEST YOUR TABLES, COLUMNS, NOT NULLS and DEFAULT VALUES

When assigning a data type you should ask the following questions.

1. Is the column going to store date or time data? Use DATE or TIMESTAMP

2. Is the column going to store string data that will NOT be used in a numerical calculation or Oracle mathematic operation? Use CHAR or VARCHAR2. Determine the estimated maximum size of the string. While in the real world, the size should be estimated on the high side. In this course keep the size limited since you will have more work to do to format the output of the SELECT statements in a future assignment.

3. Will the column be used in a numerical calculation or Oracle mathematic operation? The Oracle data type NUMBER is very flexible since it can be used with both integer (without a decimal point) or floating point number (with a decimal point). Be careful with your percussion and scale. NUMBER (7,2) will only be able to store a number to a limit of 99999.99, or less than 100,000.

***It is highly recommended to follow the development procedure that was used when coding the FACULTY, COURSE, STUDENT and SCHEDULE tables. Start SIMPLE.***

***Start with one table and a few columns. Drop the table. Add a few more columns. Continue this process until the table has been coded. Start with the second table and continue the same incremental process. If you insist on coding all of the tables and columns before testing it, your local bartender or barmaid will be very appreciative as you drown yourself in wine or beer.***

***I do not recommend that you code the Primary and Foreign Key constraints, or Check constraints and Indexes at this time. Of course, you may choose to do so.***

## Step 2. Use the ALTER TABLE statement to incrementally add the Primary Key for Each PSP Table

* **CUSTOMER (PRIMARY KEY CUSTOMER NUMBER)**
* **DRAWING (PRIMARY KEY DRAWING NUMBER)**
* **OPTION (Primary Key OPTION NUMBER)**
* **DRAWING OPTION (Primary Key DRAWING NUMBER + OPTION NUMBER)**
* **PRICE LIST (PRIMARY KEY OD + WALL + LENGTH + TYPE)**
* **RAW MATERIAL (Primary Key Raw Material ID)**
* **RAW MATERIAL REQUIREMENTS (Primary Key PACKLIST NUMBER + RAW MATERIAL ID)**
* **VENDOR (Primary Key VENDOR NUMBER)**
* **LABOR OPERATION (Primary Key OPERATION NUMBER)**
* **PACKLIST (Primary key PACKLIST NUMBER)**
* **JOBCOST (Primary Key JOB COST REFERENCE NUMBER)**

**For example:**

**ALTER TABLE CUSTOMER**

**ADD CONSTRAINT CUSTOMER\_PK PRIMARY KEY (CUSTOMER\_NUMBER);**

**ALTER TABLE CUSTOMER**

**DROP CONSTRAINT PK\_CUSTOMER;**

## Step 3. Use the ALTER TABLE statement to incrementally add Foreign Keys for Each Table.

* DRAWING (FOREIGN KEY CUSTOMER NUMBER)
* DRAWING OPTION (FOREIGN KEY DRAWING NUMBER)
* RAW MATERIAL REQUIREMENTS (FOREIGN KEYS DRAWING NUMBER (DELETE CASCADE))
* PACKLIST (FOREIGN KEY CUSTOMER NUMBER)
* PACKLIST (FOREIGN KEY DRAWING NUMBER)

**ALTER TABLE DRAWING**

**ADD CONSTRAINT FK\_DRAWING\_CUST\_NUM**

**FOREIGN KEY (CUSTOMER\_NUMBER)**

**REFERENCES CUSTOMER(CUSTOMER\_NUMBER);**

## Step 4. Use the ALTER TABLE statement to incrementally add Check Constraints for Each Table.

* CUSTOMER (CUSTOMER TYPE) must be 'B','C','S'
* OPTION (OPTION TYPE) must be % or $
* RAW MATERIAL (RAW MATERIAL TYPE) must be "T","S" or "O"
* JOBCOST (TRANSACTION CODE) must be "R", "S", "D", "L", "O", "C", "U"
* JOBCOST (TRANSACTION CROSS REFERENCE) must be "R", "S", "D", "L", "O", "C", "U"

**ALTER TABLE CUSTOMER**

**ADD CONSTRAINT CK\_CUSTOMER\_CTYPE**

**CHECK (CUSTOMER\_TYPE IN ('B','C','S'));**

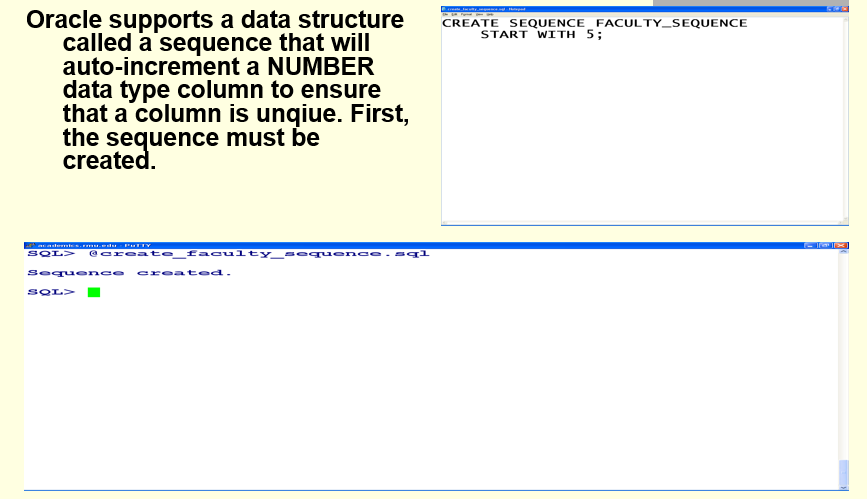
## Step 5. Use the CREATE INDEX statement to indexes for Each Table.

* **CUSTOMER (CUSTOMER NAME)**
* **DRAWING (CUSTOMER NUMBER + CUSTOMER PART NUMBER)**
* **DRAWING (CUSTOMER PART NUMBER)**
* **OPTION (OPTION DESCRIPTION)**
* **DRAWING OPTION (OPTION NUMBER)**
* **RAW MATERIAL (RAW MATERIAL ID)**
* **PACKLIST (DRAWING NUMBER)**
* **PACKLIST (CUSTOMER PURCHASE ORDER NUMBER)**
* **PACKLIST (CUSTOMER PART NUMBER)**
* **PACKLIST (SCHEDULED SHIP DATE)**
* **JOBCOST (PACKLIST NUMBER + TRANSACTION DATE)**

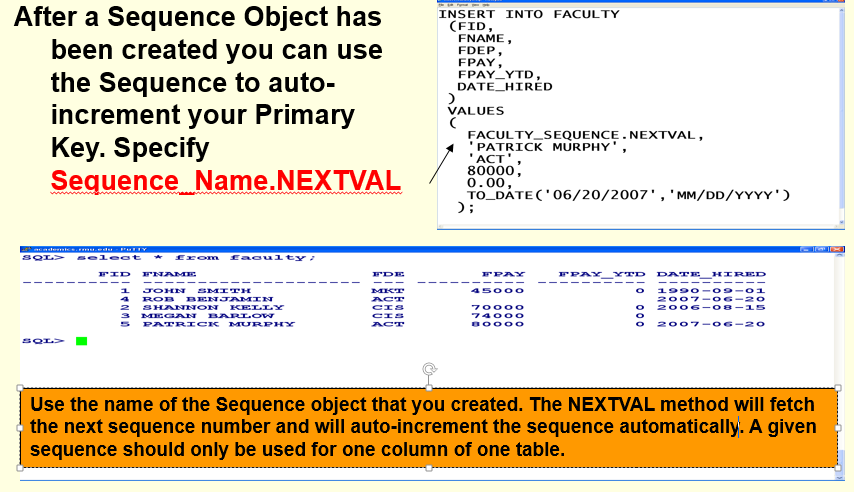
## Step 6. Use the CREATE SEQUENECE statement to create a sequence object to auto increment the following Primary keys.

* **JOBCOST (Primary Key JOB COST REFERENCE NUMBER)**
* **VENDOR (Primary Key VENDOR NUMBER)**

**Creating a Sequence Object**



**Using a Sequence Object in the INSERT Statement**



**Refer to Assignment 3 Hands-on to provide specific directions.**

**Use a graphical snipping tool to document an image of these requirements below. Make sure your Putty banner is displayed.**

## 6.7 Document your PSP Tables

**Enter the SQL\*Plus command Clear Screen. Enter the SQL\*Plus command Clear Screen. Execute and insert an Image for your SELECT TABLE\_NAME FROM TABS; below.** Answer =>

**I must see all of the tables that you created for this assignment.**

## 6.8 Document your Customer Table

**Enter the SQL\*Plus command Clear Screen. Enter the SQL\*Plus command Clear Screen. Execute and insert an Image for your DESCRIBE CUSTOMER Table below.** Answer =>

**Use a graphical snipping tool to document an image of these requirements below. Make sure your Putty banner is displayed.**

## 6.9 Document your Drawing Table

**Enter the SQL\*Plus command Clear Screen. Enter the SQL\*Plus command Clear Screen. Execute and insert an Image for your DESCRIBE DRAWING Table below.** Answer =>

## 6.10 Document your Option Table

**Enter the SQL\*Plus command Clear Screen. Enter the SQL\*Plus command Clear Screen. Execute and insert an Image for your DESCRIBE OPTION Table below.** Answer =>

**Use a graphical snipping tool to document an image of these requirements below. Make sure your Putty banner is displayed.**

## 6.11 Document your Drawing Option Table

**Enter the SQL\*Plus command Clear Screen. Enter the SQL\*Plus command Clear Screen. Execute and insert an Image for your DESCRIBE DRAWING\_OPTION Table below.** Answer =>

## 6.12 Document your Raw Materials Table

**Enter the SQL\*Plus command Clear Screen. Enter the SQL\*Plus command Clear Screen. Execute and insert an Image for your DESCRIBE RAW-MATERIALS Table below.** Answer =>

## 6.13 Document your Packlist Table

**Enter the SQL\*Plus command Clear Screen. Enter the SQL\*Plus command Clear Screen. Execute and insert an Image for your DESCRIBE PACKLIST Table below.** Answer =>

## 6.14 Document your PSP Primary Key, Foreign Key, and Check Constraints

**Enter the SQL\*Plus command Clear Screen. Enter the SQL\*Plus command Clear Screen. Execute and insert an Image for your show\_constraints.sql script below.** Answer =>

**I must see all of the constraints that you created for this assignment.**

**Use a graphical snipping tool to document an image of these requirements below. Make sure your Putty banner is displayed.**

## 6.15 Document your PSP INDEXES

**Enter the SQL\*Plus command Clear Screen. Enter the SQL\*Plus command Clear Screen. Execute and insert an Image for your show\_indexes.sql script below.** Answer =>

**I must see all of the constraints that you created for this assignment.**

# 7.0 Enter Test Data into the PSP Database

The requirements for this script are to (you should test one row and simply copy and slightly edit the remaining insert row statements).

* Insert eight (8) customer rows using a faculty name that are members of your family. You must enter at least one customer for each customer type. You may enter other data as you may determine to be appropriate.
* Insert ten (10) drawing rows using any data as you may determine to be appropriate. According to the rules of referential integrity you may not enter a customer number that was not previously entered into the customer table. At least four customers must have drawings
* Insert four (4) price list rows using any data as you may determine to be appropriate.
* Insert eight (8) option rows using any data as you may determine to be appropriate.
* Insert eight (8) drawing option rows using any data as you may determine to be appropriate.
* Insert four (4) raw material rows using any data as you may determine to be appropriate.
* Insert 15 packlist rows using any data as you may determine to be appropriate. You must insert at least six different drawings.
* Insert 15 raw material requirement rows s using any data as you may determine to be appropriate.
* Insert 25 job costs using any data as you may determine to be appropriate. You must insert at least six ten different packlists, which will also use a selection of customers and drawings.

Document the data the data that you have entered into your table by using the previous INSERT statements.

**Use a graphical snipping tool to document an image of these requirements below. Make sure your Putty banner is displayed.**

## 7.1 Document the Test Data that You Inserted into your PSP Tables

1. **Enter the SQLPLUS command Clear Screen. Execute and insert an image for your SELECT \* FROM CUSTOMER; to document your test data below:**

Answer =>

**2. Enter the SQLPLUS command Clear Screen. Execute and insert an image for your SELECT \* FROM DRAWING; to document your test data below:**

Answer =>

**3. Enter the SQLPLUS command Clear Screen. Execute and insert an image for your SELECT \* FROM OPTION; to document your test data below:**

Answer =>

**Use a graphical snipping tool to document an image of these requirements below. Make sure your Putty banner is displayed.**

**4. Enter the SQLPLUS command Clear Screen. Execute and insert an image for your SELECT \* FROM DRAWING OPTION; to document your test data below:**

Answer =>

**5. Enter the SQLPLUS command Clear Screen. Execute and insert an image for your SELECT \* FROM RAW MATERIALS; to document your test data below:**

Answer =>

**6. Enter the SQLPLUS command Clear Screen. Execute and insert an image for your SELECT \* FROM PACKLIST; to document your test data below:**

Answer =>

**7. Enter the SQLPLUS command Clear Screen. Execute and insert an image for r your SELECT \* FROM JOBCOST; to document your test data below:**

Answer =>