Assignment 5 Part 2 Hands-on

SUmmer 2018

***J. Packy Laverty***

***laverty@rmu.edu***

Contents

[**Introduction 5**](#_Toc508283274)

[**How to complete Hand-on Tutorial Requirements and Review Questions 5**](#_Toc508283275)

[**Oracle Error Codes Summary 6**](#_Toc508283276)

[**Documentation Requirements 6**](#_Toc508283277)

[**1.0 Required Professional Appearance of Outputted Results 7**](#_Toc508283278)

[**1.1 Improving the Professional Appearance of the SQL Statement and Results using SQL\*Plus 7**](#_Toc508283279)

[**1.2 Improving the Professional Appearance of the SQL Statement and Results using PuTTY 7**](#_Toc508283280)

[**1.3 Using the SQL SUBSTR() and TO\_CHAR() Function and Columns Aliases (AS) 8**](#_Toc508283281)

[**1.4 SUBSTR() Function 9**](#_Toc508283282)

[**1.5 AS (Column Alias) 10**](#_Toc508283283)

[**1.6 TO\_CHAR() Function 11**](#_Toc508283284)

[**2.0 Sample Tables for the PSP Case 13**](#_Toc508283285)

[**CUSTOMER Table (PRIMARY KEY CUSTOMER NUMBER) 13**](#_Toc508283286)

[**CUSTOMER SHIPPING ADDRESS TABLE (PRIMARY KEY CUSTOMER SHIPPING ID) 13**](#_Toc508283287)

[**DRAWING TABLE (PRIMARY KEY DRAWING NUMBER) 14**](#_Toc508283288)

[**PRICE LIST TABLE (PRIMARY KEY OD + WALL + LENGTH + TYPE) 15**](#_Toc508283289)

[**OPTION TABLE (Primary Key OPTION NUMBER) 15**](#_Toc508283290)

[**DRAWING OPTION TABLE (Primary Key DRAWING NUMBER + OPTION NUMBER) 15**](#_Toc508283291)

[**TUBULAR TABLE (Primary Key Raw Material ID) 16**](#_Toc508283292)

[**SUPPLIES TABLE (Primary Key Raw Material ID) 16**](#_Toc508283293)

[**RAW MATERIAL REQUIREMENTS TABLE (Primary Key PACKLIST NUMBER + RAW MATERIAL ID 16**](#_Toc508283294)

[**RAW MATERIAL FIFO TABLE (Primary Key FIFO ID, auto sequence) 16**](#_Toc508283295)

[**RAW MATERIAL DROP INVENTORY (Primary Key DROP REFERENCE NUMBER, auto sequence) 17**](#_Toc508283296)

[**VENDOR TABLE (Primary Key VENDOR NUMBER auto sequence, auto sequence) 17**](#_Toc508283297)

[**INVENTORY VENDOR (Primary Key VENDOR NUMBER + RAW MATERIAL NUMBER 17**](#_Toc508283298)

[**LABOR OPERATION TABLE (Primary Key OPERATION NUMBER) 17**](#_Toc508283299)

[**PACKLIST TABLE (Primary key PACKLIST NUMBER) 18**](#_Toc508283300)

[**JOBCOST TABLE (Primary Key JOB COST REFERENCE NUMBER, auto sequence) 18**](#_Toc508283301)

[**EMPLOYEE TABLE (Primary Key Clock Number) 19**](#_Toc508283302)

[**TIME CARD TABLE (Primary Key TIME CARD REFERENCE NUMBER, auto sequence) 19**](#_Toc508283303)

[**3.0 Introductory SQL DML Statements 20**](#_Toc508283304)

[**3.1 Questions - ALTER TABLE 21**](#_Toc508283305)

[**3.2 UPDATE TABLE 22**](#_Toc508283306)

[**3.3 Questions - UPDATE TABLE 23**](#_Toc508283307)

[**3.4 Questions - DELETE TABLE 24**](#_Toc508283308)

[**3.5 Questions - ORDER BY Clause 25**](#_Toc508283309)

[**3.6 WHERE Clause 26**](#_Toc508283310)

[**3.7 Questions - WHERE Clause 28**](#_Toc508283311)

[**3.8 Comparison Operators 28**](#_Toc508283312)

[**3.9 LIKE operator 29**](#_Toc508283313)

[**3.10 IN operator 31**](#_Toc508283314)

[**3.11 TO\_DATE() Function (Specifying a DATE in a WHERE clause. 32**](#_Toc508283315)

[**3.12 BETWEEN/AND Operators 32**](#_Toc508283316)

[**3.14 ROWNUM, TOP, BOTTOM (Limiting the number of rows retrieved) 33**](#_Toc508283317)

[**3.15 Questions - Concatenation 34**](#_Toc508283318)

[**3.16 Questions- RTRIM() Function 34**](#_Toc508283319)

[**3.14 Questions- SYSDATE() Function 34**](#_Toc508283320)

[**3.17 Mathematical Expressions and DECODE 36**](#_Toc508283321)

[**3.18 Questions - Aggregate Functions and Group By 38**](#_Toc508283322)

[**3.19 Questions- GROUP and HAVING 42**](#_Toc508283323)

[**3.19.1 Apply Aggregate (column) functions 42**](#_Toc508283324)

[**3.19.2 Apply GROUP BY Clauses 42**](#_Toc508283325)

[**3.19.3 Apply WHERE and GROUP BY Clauses 43**](#_Toc508283326)

[**3.20 Questions - Apply the LIKE Operator 43**](#_Toc508283327)

[**4.0 Dynamic Queries (Interactive Queries) and Oracle Reports 44**](#_Toc508283328)

[**4.1 Introduction to Interactive Queries 45**](#_Toc508283329)

[**4.2 Questions – Dynamic Inquires 47**](#_Toc508283330)

[**4.3 Oracle Reports in SQL\*Plus 47**](#_Toc508283331)

[**4.4 Questions - Oracle Reports in SQL\*Plus 48**](#_Toc508283332)

[**5.0 - Your Customized Examples of SQL SELECT Statements 49**](#_Toc508283333)

[**5.1 ORDER BY, DESC, ROWNUM, TOP or BOTTOM Examples 52**](#_Toc508283334)

[**5.1.1 Your Example 1 52**](#_Toc508283335)

[**5.1.2 Your Example 2 52**](#_Toc508283336)

[**5.1.3 Your Example 3 53**](#_Toc508283337)

[**5.1.4 Your Example 4 53**](#_Toc508283338)

[**5.1.5 Your Example 5 54**](#_Toc508283339)

[**5.2 WHERE, AND, OR, TO\_DATE(), TO\_CHAR(), or IN examples 55**](#_Toc508283340)

[**5.2.1 Your Example 1 55**](#_Toc508283341)

[**5.2.2 Your Example 2 56**](#_Toc508283342)

[**5.2.3 Your Example 3 56**](#_Toc508283343)

[**5.2.4 Your Example 4 57**](#_Toc508283344)

[**5.2.5 Your Example 5 57**](#_Toc508283345)

[**5.3 SUM(), COUNT(), ANG(), MIN(), MAX(), GROUP BY, or HAVING Examples 58**](#_Toc508283346)

[**5.3.1 Your Example 1 58**](#_Toc508283347)

[**5.3.2 Your Example 2 59**](#_Toc508283348)

[**5.3.3 Your Example 3 60**](#_Toc508283349)

[**5.4.4 Your Example 4 60**](#_Toc508283350)

[**5.3.5 Your Example 5 61**](#_Toc508283351)

[**5.4 ACCEPT, SQL\*Plus mathematical expression formatting in the SELECT LIST 62**](#_Toc508283352)

[**5.4.1 Your Example 1 62**](#_Toc508283353)

[**5.4.2 Your Example 2 62**](#_Toc508283354)

[**5.4.3 Your Example 3 63**](#_Toc508283355)

[**5.4.4 Your Example 4 63**](#_Toc508283356)

[**5.4.5 Your Example 5 64**](#_Toc508283357)

**Enter your Name Here 🡺 Puja Ghosh**

# Introduction

**Assignment 5 is organized into two documents: Assignment 5 – Part 1 and Assignment 5- Part 2 Hands-on.**

**Assignment 5 – Part 1 will be allocated 30% of the assignment points**

**Assignment 5 – Part 2 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**

## Documentation Requirements

The following are requirements to code, execute, test and provide documentation of SQL statement

If you encounter a problem coding of a complete SQL statement that meets all requirements, then code a partial SQL statement as close to the requirement objectives as possible. Don’t leave the requirement blank. Change your Requirement header to reflect the partial SQL statement. If you are close enough I will give you credit.

**1. Using a Snipping Tool document the code of required SQL statement –** properly formatted.

**2. Using a Snipping Tool document the Results of the execution of the SQL Statement – properly formatted**

# 1.0 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.

## 1.1 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.

## 1.2 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)

## 1.3 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,**

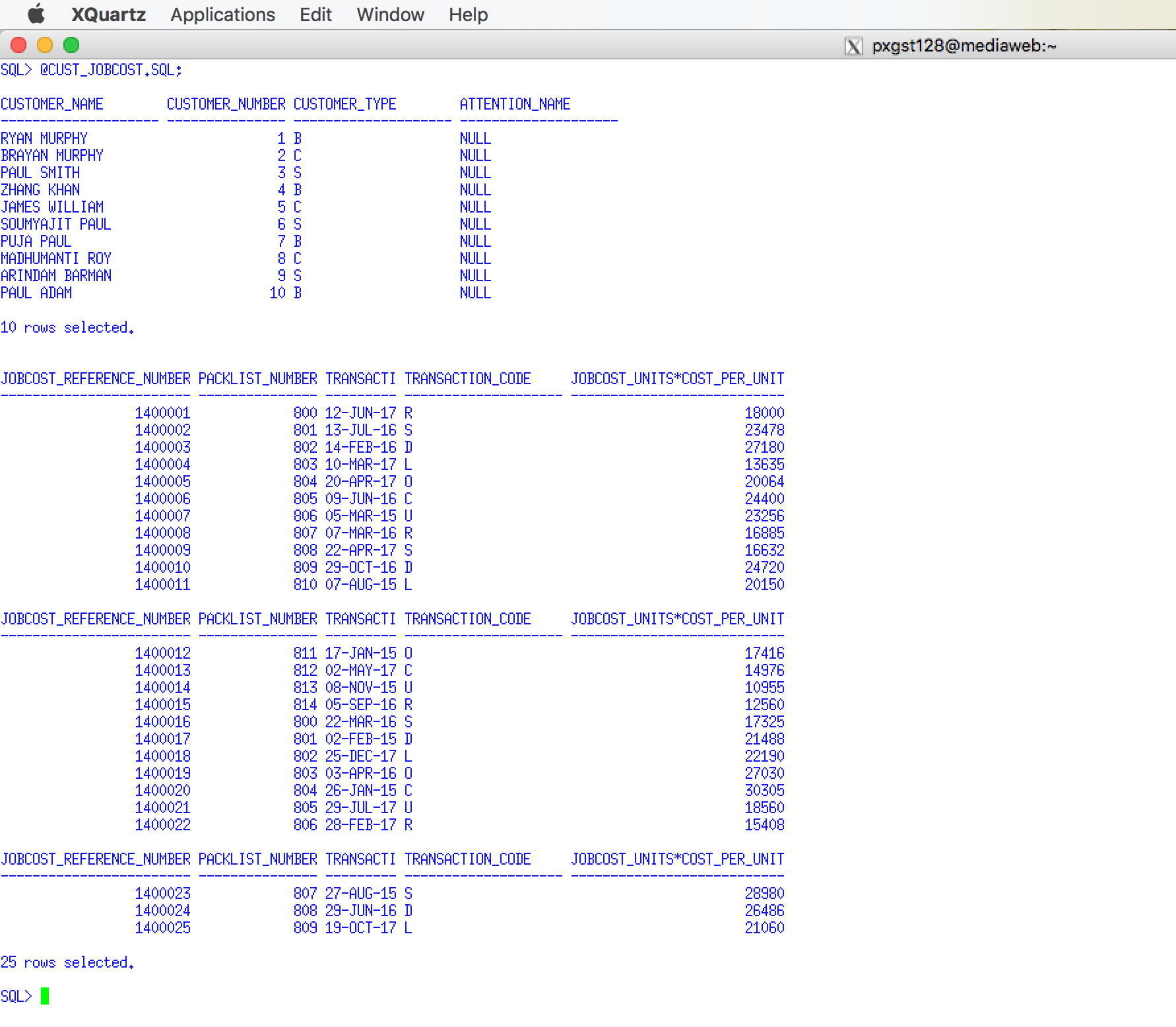
**TRANSACTION\_DATE,**

**TRANSACTION CODE,**

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

**FROM JOB\_COST;**

**Using a Snipping Tool document Your Results**



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.

## 1.4 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),**

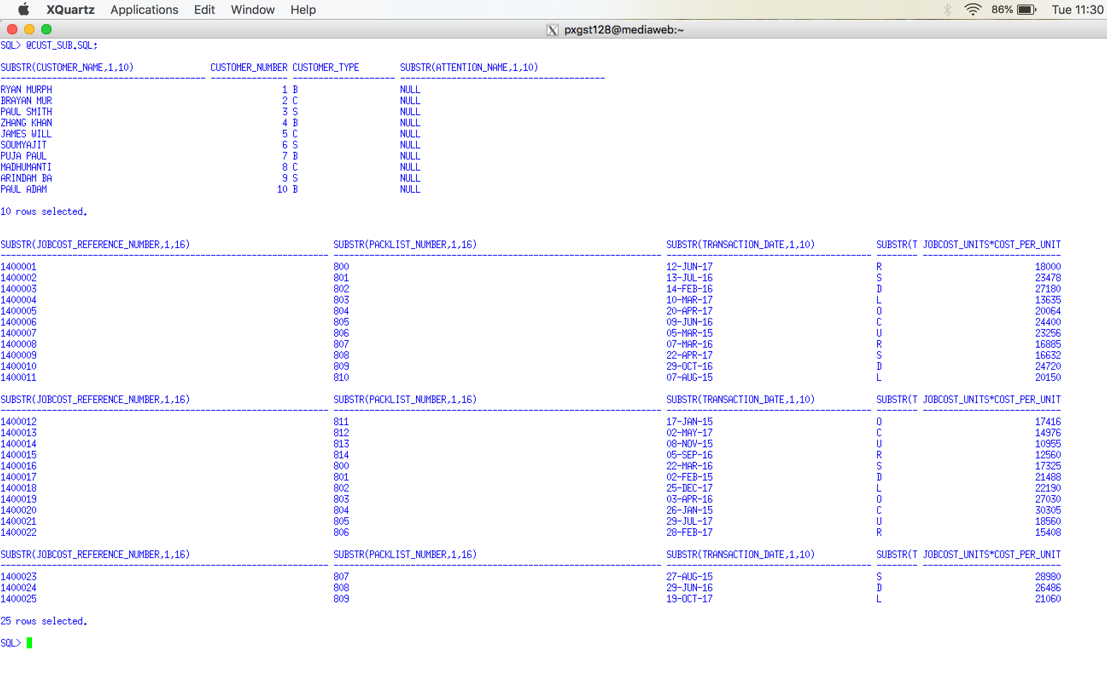
**SUBSTR(TRANSACTION\_DATE, 1, 10),**

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

JOBCOST\_UNITS \* COST\_PER\_UNIT

FROM JOB\_COST;

**Using a Snipping Tool document Your Results**



## 1.5 AS (Column Alias)

**Step 3.** Provide a Column Alias to make the column header more meaning full. By default Oracle uses the column name as the displayed column header. When we use the SUBSTR() function we may truncate the column header. A column alias may be appropriate. A column alias may be appropriate to may a column header of a calculation more descriptive. A column alias uses the AS clause. Do not get a column alias confused with a table alias. Notice that a column alias is not permitted to have embedded spaces UNLESS it is enclosed in quotes (single or double). You may use upper or lower case letters depending on the professional appearance desired. SQL Aliases - <http://www.w3schools.com/sql/sql_alias.asp>

Using Column Alias in SELECT Statement - <http://www.geeksengine.com/database/basic-select/column-alias.php>

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

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

CUSTOMER\_NUMBER ,

CUSTOMER\_TYPE **AS TYPE** ,

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

FROM CUSTOMER;

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

SUBSTR(PACKLIST\_NUMBER, 1, 6) **AS PL#** ,

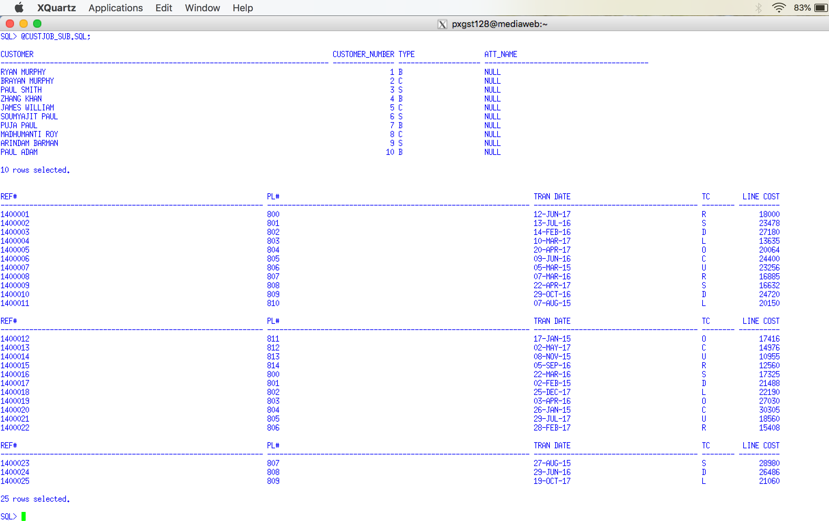
SUBSTR(TRANSACTION\_DATE, 1, 10) **AS "TRAN DATE"** ,

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

JOBCOST\_UNITS \* COST\_PER\_UNIT **AS "LINE COST"**

FROM JOB\_COST;

**Using a Snipping Tool document Your Results**



## 1.6 TO\_CHAR() Function

**Step 4.** Use the TO\_CHAR() function to format date and numeric values as appropriate.

TO\_CHAR Function http://www.techonthenet.com/oracle/functions/to\_char.php

TO\_CHAR http://www.java2s.com/Tutorial/Oracle/0300\_\_Conversion-Functions/0140\_\_TO\_CHAR.htm

TO\_CHAR - Convert Datetime to String - Oracle to SQL Server Migration - http://www.sqlines.com/oracle-to-sql-server/to\_char\_datetime

Video - TO\_CHAR(datetime) Function in SQL Query - <https://www.youtube.com/watch?v=00VT7Sgo5ig>

Vide - Concatenation Operator, To\_Char Currency, Money Symbol in Oracle SQL - <https://www.youtube.com/watch?v=9ePcZrzQcYQ>

Video - Oracle TO\_CHAR Function- <https://www.youtube.com/watch?v=X-uQDdmORvI>

Video - TO\_CHAR(number) Function in SQL Query- <https://www.youtube.com/watch?v=fuXkFmD1FKY>

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

SELECT SUBSTR(JOBCOST\_REFERENCE\_NUMBER, 1, 6) AS REF#,

SUBSTR(PACKLIST\_NUMBER, 1, 6) AS PL# ,

**TO\_CHAR(**SUBSTR(TRANSACTION\_DATE, 1, 10) **, 'DD/MM/YYYY' )** AS "TRAN DATE" ,

SUBSTR(TRANSACTION CODE, 1, 2) AS TC,

**TO\_CHAR**( (JOBCOST\_UNITS \* COST\_PER\_UNIT) **, '9999.99')** AS "LINE COST"

FROM JOB\_COST;

**Using a Snipping Tool document Your Results**

**Support Videos**

Oracle Express is used in these videos. You are not required to use Oracle Express. It does not matter if you are using the SQL\*Plus client or Oracle Express. The SQL coding concepts are the same.

**Introduction to SQL - SQL LESSON 1**

**http://www.youtube.com/watch?v=UeJKioNqe5w&feature=relmfu**

**Express Tour of SQL - PART 1 - SQL LESSON 2**

**http://www.youtube.com/watch?v=HOrpREPUs2o&feature=relmfu**

**Express Tour of SQL - PART 2 - SQL LESSON 3**

**http://www.youtube.com/watch?v=5bcLy2PT8Fk&feature=relmfu**

**Installing Oracle Express - PART 1 - SQL LESSON 4**

**http://www.youtube.com/watch?v=4EwvoCYlGmo&feature=relmfu**

**Installing Oracle Express - PART 2 - SQL LESSON 5**

**http://www.youtube.com/watch?v=0TVwQoU8vU4&feature=relmfu**

**SQL Conditions, Expressions, and Operators - PART 1 - SQL LESSON 6 - http://www.youtube.com/watch?v=lPzW1H1nbaE&feature=relmfu**

**SQL Conditions, Expressions, and Operators - PART 2 - SQL LESSON 7**

**http://www.youtube.com/watch?v=3mnc7LROq-0&feature=relmfu**

**SQL Conditions, Expressions, and Operators - PART 3 - SQL LESSON 8 -**

**http://www.youtube.com/watch?v=zUAE2Ao1m-M&feature=relmfu**

**SQL Conditions, Expressions, and Operators - PART 4 - SQL LESSON 9 - http://www.youtube.com/watch?v=RzS7eDHfGZk&feature=relmfu**

**SQL Logical Operators - PART 1 - SQL LESSON 10 -**

**http://www.youtube.com/watch?v=jhSm8tFvWSI&feature=relmfu**

**SQL Logical Operators - PART 2 - SQL LESSON 11 -**

**http://www.youtube.com/watch?v=oVF1s8WiDDk&feature=fvwrel**

**SQL Logical Operators - PART 3 - SQL LESSON 12 -**

**http://www.youtube.com/watch?v=btZ39GrWPuQ&feature=relmfu**

**SQL Multiple Logical Operators - PART 1 - SQL LESSON 13 -**

**http://www.youtube.com/watch?v=1FNlnqjW2vk&feature=relmfu**

**SQL Multiple Logical Operators - PART 2 - SQL LESSON 14 -**

**http://www.youtube.com/watch?v=gaISO0Xdpi4&feature=relmfu**

**SQL Multiple Logical Operators - PART 3 - SQL LESSON 15**

**http://www.youtube.com/watch?v=yZbml5Kga0I&feature=relmfu**

**Using Brackets in SQL to Simplify the Where Clause - PART 1 - SQL LESSON 15**

**http://www.youtube.com/watch?v=k29P\_8zSD7w&feature=relmfu**

**Using Brackets in SQL to Simplify the Where Clause - PART 2 - SQL LESSON 16**

**http://www.youtube.com/watch?v=0U5DgEV0S0Q&feature=relmfu**

**Equality and Inequality Conditions in SQL Where Clause - SQL LESSON 17**

**http://www.youtube.com/watch?v=MmzZDq3HGzw&feature=relmfu**

**Membership Conditions in SQL Where Clause (Subqueries) - Part 1 - SQL LESSON 18 -**

**http://www.youtube.com/watch?v=7MKQGXcw1aw&feature=relmfu**

**Membership Conditions in SQL Where Clause (Subqueries) - Part 2 - SQL LESSON 19 - http://www.youtube.com/watch?v=\_X1oiNqDVeE&feature=fvwrel**

**Range Conditions in SQL Where Clause (Between) - SQL LESSON 20 -**

**http://www.youtube.com/watch?v=z5YCPGzq-Ac&feature=fvwrel**

**Matching Conditions in SQL Where Clause (IN) - Part 1 - SQL LESSON 21 - http://www.youtube.com/watch?v=MIHA7ptVom0&feature=relmfu**

**Matching Conditions in SQL Where Clause (IN) - Part 2 - SQL LESSON 22 -**

**http://www.youtube.com/watch?v=bK1EEnTFQl0&feature=relmfu**

**SQL Comparison Conditions - SQL LESSON 22 B -**

**http://www.youtube.com/watch?v=iSX4Yb3\_TPY&feature=relmfu**

**Concept of a Null Value in SQL - Part 1 - SQL LESSON 23 -**

**http://www.youtube.com/watch?v=XoH-T\_0Ik9k&feature=relmfu**

**Problems with Null Value in SQL - SQL LESSON 25 -**

**http://www.youtube.com/watch?v=4cFfPaJCkxw&feature=relmfu**

**Sorting in SQL Using Order by Clause - Part 1 - SQL LESSON 27 -**

**http://www.youtube.com/watch?v=0ljgw-EFqJE&feature=relmfu**

**Sorting in SQL Using Order by Clause - Part 2 - SQL LESSON 28**

**http://www.youtube.com/watch?v=PnlgMtMh4i0&feature=channel&list=UL**

**Sorting in Descending Order in SQL Using Order By Desc Clause - SQL LESSON 29**

**http://www.youtube.com/watch?v=kBDIf6fEt-E&feature=channel&list=UL**

# 2.0 Sample Tables for the PSP Case

The following table names and columns names without an assigned data type were provided for you in a previous assignment. Please use the table names, columns names, and data types that you used in your coded for your tables assignment. You are only required required to use the red- columns

## CUSTOMER Table (PRIMARY KEY CUSTOMER NUMBER)

**CUSTOMER NAME**

**CUSTOMER NUMBER {common column}**

**CUSTOMER TYPE must be 'B','C','S' (where B means Blanket, C means Contract, S means Standard)**

**ATTENTION NAME**

**DATE ACTIVATED**

**ADDRESS 1 BILL**

**CITY BILL**

**STATE BILL**

**ZIPCODE BILL**

**COUNTRY BILL**

CUSTOMER SALES PERSON NO

PREFERRED SHIP METHOD

CUSTOMER CLASS

F O B (What Does FOB Mean in Shipping? https://www.freightquote.com/blog/what-does-fob-mean-in-freight-shipping)

BLANKET PO

DATE EXPIRE

FREIGHT FLAG

PRICE CHANGE FLAG

DATE ACTIVATED

SIC (Standard Industrial Classification - https://en.wikipedia.org/wiki/Standard\_Industrial\_Classification)

CUSTOMER PRIMARY PHONE NUMBER

CUSTOMER PRIMARY EMAIL CONTACT

**Customer Table Indexes CUSTOMER NAME**

## CUSTOMER SHIPPING ADDRESS TABLE (PRIMARY KEY CUSTOMER SHIPPING ID)

CUSTOMER SHIPPING ID

CUSTOMER NUMBER {common column}

DATE ADDED

DATE MODIFIED

ADDRESS 1 SHIP

ADDRESS 2 SHIP

CITY SHIP

STATE SHIP

ZIPCODE SHIP

COUNTRY SHIP

SHIP TYPE must be 'W', 'M','D', or 'O' where W means Warehouse, M means Manufacturing Plant, D means Drop Ship, and O means Other

Customer Shipping Table Indexes CUSTOMER NUMBER

Customer Shipping Address Table Foreign Keys CUSTOMER NUMBER (delete cascade)

## DRAWING TABLE (PRIMARY KEY DRAWING NUMBER)

**DRAWING NUMBER {common column}**

**DRAWING PREFIX (A prefix categorizes the general type of drawing, e.g., "S" Shaft, "R", Bar "B", etc.)**

**REVISION LEVEL (A revision level specifies the current version of the drawing to be used for the next customer order, which may be different from past revision levels or those on the current schedule.)**

**ORIGINAL DATE ENTERED**

**DATE LAST REVISED**

**CUSTOMER NUMBER**

**CUSTOMER PART NUMBER**

**PRIMARY RAW MATERIAL ID (Every drawing has a primary raw material used to start the packing list production process. It may be a hollow tube, a solid tube, and solid bard, plastic, type of wood, etc. All other production materials charged to the will be called supplies, not raw material.)**

**PPRIMARY RAW MATERIAL UNIT (The primary raw material unit represents how the raw material will be disbursed to the packing list, by tube, by foot, by pound, etc.**

**OD**

**WALL**

**LENGH**

**CUT LENGTH**

**GL\_NUMBER**

**Drawing Table Indexes CUSTOMER NUMBER + CUSTOMER PART NUMBER**

**PRIMARY RAW MATERIAL NUMBER**

**CUSTOMER PART NUMBER**

**Drawing Table Foreign Keys CUSTOMER NUMBER (delete cascade)**

**Note: When a Primary Key, Foreign Key, or Index specifies two columns combined by a plus-sign "+", e.g., column\_1 + column\_2, this is an indication of a composite key.**

## PRICE LIST TABLE (PRIMARY KEY OD + WALL + LENGTH + TYPE)

**OD**

**WALL**

**LENGH**

**TYPE (matches one of the categories of drawing prefixes)**

**PRICE 0 TO 5 (to store**

**PRICE 6 TO 25**

**PRICE 26 TO 50**

**PRICE 50 TO 100**

**PRICE 101 TO 250**

**PRICE OVER 250**

**Comments: This Company was originally started as a customized tubular manufacturer and expanded its product lines into other customized areas. Bob's original pricing formulas and quantity discounts was based on a raw material tube dimensions OD (outside diameter), Wall (the thickness of a tube) and Length of a tube. The type of a tube was contrast non-tube products, e.g., bar product, and howitzer shell casings. I wanted to make primary key more independent of a tubular product line but Bob would not approve it. The only area where the price table was important was to help Tom quote prices and then store this primary key in the Packing List table to document that set of prices used to quote the order.**

## OPTION TABLE (Primary Key OPTION NUMBER)

**OPTION NUMBER {common column}**

**OPTION TYPE must be % or $**

**OPTION DESCRIPITON**

**OPTION UNIT COST**

**OPTION UNIT PRICE**

**OPTION PRECENTAGE**

**Option Table Indexes OPTION DESCRIPTION**

## DRAWING OPTION TABLE (Primary Key DRAWING NUMBER + OPTION NUMBER)

**DRAWING NUMBER {common column}**

**OPTION NUMBER {common column}**

**UNITS-OF-OPTION**

**Drawing Option Table Indexes OPTION NUMBER**

**Drawing Option Table Foreign Keys DRAWING NUMBER (delete cascade)**

**RAW MATERIAL (Primary Key Raw Material ID)**

**RAW MATERIAL ID**

**RAW MATERIAL TYPE must be "T","S" or "O" ("T" means Tubular, "S" Supplies, "O" means Other)**

**MATERIAL DESCIPTION**

**UNIT TYPE**

**CURRENT UNIT COST**

Stores common information for tubular and supplies tables. This is an example of generalization and specialization

## TUBULAR TABLE (Primary Key Raw Material ID)

**RAW MATERIAL ID {common column}**

**OD**

**WALL**

**LENGH**

**CURRENT UNIT COST**

**CURRENT COST PER FOOT**

**MINIMUM DROP LENGTH**

**Tubular Inventory Table Indexes OD + WALL + LENGTH**

**Tubular Inventory Table Foreign Keys RAW MATERIAL ID (delete cascade)**

## SUPPLIES TABLE (Primary Key Raw Material ID)

**RAW MATERIAL ID {common column}**

**CURRENT UNIT COST**

**REORDER POINT**

**SAFETY STOCK**

**Tubular Inventory Table Foreign Keys RAW MATERIAL ID (delete cascade)**

## RAW MATERIAL REQUIREMENTS TABLE (Primary Key PACKLIST NUMBER + RAW MATERIAL ID

**PACKLIST NUMBER {common column}**

**RAW MATERIAL ID**

**SCHEDULED SHIP DATE**

**UNITS REQUIRED**

**Raw Material Table Indexes RAW MATERIAL ID**

**Raw Material Requirements Foreign Keys DRAWING NUMBER (delete cascade)**

(Comments: this table was used to determine future raw material requirements. One row was entered when the PL was recorded. As the raw material was used during the manufacturing process, the units required was changed. When the PL was completed this row was deleted.)

## RAW MATERIAL FIFO TABLE (Primary Key FIFO ID, auto sequence)

FIFO ID

RAW MATERIAL ID

DATE DELIVERED

UNITS ON HAND

UNIT COST

VENDOR NUMBER

MFG CODE

Raw Material Fifo Indexes RAW MATERIAL ID + DATE DELIVERED

Raw Material Fifo Table Foreign Keys RAW MATERTIAL ID (delete cascade)

## RAW MATERIAL DROP INVENTORY (Primary Key DROP REFERENCE NUMBER, auto sequence)

DROP REFERENCE NUMBER **{common column}**

RAW MATERIAL ID

DROP LENGTH

COST PER FOOT

DROP UNITS

Raw Material Drop Inventory Indexes RAW MATERIAL ID + DROP LENGTH

Raw Material Drop Inventory Foreign Keys RAW MATERTIAL ID (delete cascade)

## VENDOR TABLE (Primary Key VENDOR NUMBER auto sequence, auto sequence)

**VENDOR NUMBER {common column}**

**VENDOR NAME**

**Vendor Table Indexes VENDOR NAME**

## INVENTORY VENDOR (Primary Key VENDOR NUMBER + RAW MATERIAL NUMBER

RAW MATERIAL ID **{common column}**

VENDOR NUMBER **{common column}**

MFG CODE

LAST COST PER UNIT

Inventory Vendor Table Indexes VENDOR NUMBER

Inventory Vendor Table Foreign Keys RAW MATERTIAL ID (delete cascade)

VENDOR NUMBER (delete cascade

## LABOR OPERATION TABLE (Primary Key OPERATION NUMBER)

**OPERATION NUMBER {common column}**

**OPERATION DESCRIPTION**

**SCHEDULE CATAGORY must be 0 thru 20**

**Labor Operation Table Indexes OPTION DESCRIPTON**

**(Comments: This table was used to the type of labor operation used for labor costs in the Job Cost table. Also the type of labor operation was recorded in the Time Card. The Time Card table was used by applications to determine the job progress of a Packing List. Most labor operations were assigned a schedule category from a number 0 (no job progress) to 20 (ready for delivery). The schedule category roughly control the sequence of a packing list as it moved through production. Several labor operations could be assigned schedule category 1, i.e., cutting. If there were 20 units ordered on a Packing List and Tom saw 20 units cut on the job progress report, he would know that cutting step had been completed. It didn't matter which type of cutting was performed. Not all packing lists used all schedule category steps.)**

## PACKLIST TABLE (Primary key PACKLIST NUMBER)

**PACKLIST NUMBER {common column}**

**DRAWING NUMBER {common column}**

**REVISION LEVEL**

**CUSTOMER NUMBER {common column}**

**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**

**PACKLIST Table Indexes DRAWING NUMBER**

**CUSTOMER PURCHASE ORDER NUMBER**

**CUSTOMER NUMBER + CUSTOMER PURCHASE ORDER NUMBER**

**CUSTOMER PART NUMBER**

**SCHEDULED SHIP DATE**

## JOBCOST TABLE (Primary Key JOB COST REFERENCE NUMBER, auto sequence)

**JOBCOST REFERENCE NUMBER**

**PACKLIST NUMBER {common column}**

**TRANSACTION DATE**

**TRANSACTION CODE must be "R", "S", "D", "L", "O", "C", "U", where**

**"R" means Raw Materials,**

**"S" means Supplies,**

**"D" means Drops,**

**"L" means Regular Labor**

**"O" means Overtime Labor**

**"C" means Outside Contract**

**"U" means Undefined**

**TRANSACTION CROSS REFERENCE**

**"R" will store Raw Material ID**

**"S" will store Supply ID**

**"D" will store Drop Reference Number**

**"L" will store Labor Operation**

**"O" will store Labor Operation**

**"C" will store AP Invoice Number**

**"U" means Undefined**

**CLOCK NUMBER**

**JOBCOST UNITS**

**COST PER UNIT**

**Jobcost Table Indexes PACKLIST NUMBER + TRANSACTION DATE**

**TRANSACTION CODE + TRANSACTION CROSS REFERENCE**

**TRANSACTION DATE**

**When the Job cost type is a labor type, you should have inserted one of your labor operation numbers into the cross reference number. You can insert a new Job COST row or alter a row in the Job Cost table. Likewise the cross reference number should have been a raw material number for material transactions.**

## EMPLOYEE TABLE (Primary Key Clock Number)

SOCIAL SECURITY NUMBER

CLOCK NUMBER **{common column}**

EMPLOYEE LAST NAME

EMPLOYEE FIRST NAME

EMPLOYEE MIDDLE INITIAL

HOURLY RATE

BONUS CHARGE must be "Y" or "N"

Employee Table Indexes SOCIAL SECURITY NUMBER

EMPLOYEE LAST NAME

## TIME CARD TABLE (Primary Key TIME CARD REFERENCE NUMBER, auto sequence)

TIME CARD REFERENCE NUMBER **{common column}**

CLOCK NUMBER

TIME CARD DATE

SHIFT NUMBER must be 1, 2 or 3

PACK LIST NUMBER **{common column}**

OPERATION NUMBER

HOURS must be 1 to 24

MINUTES must be 0, 15, 30 or 45

GOOD PIECES

BAD PIECES

REWORED PIECES

OVER TIME

HOURLY RATE

Time Card Table Indexes CLOCK NUMBER + TIME CARD DATE

(This index was used to determine the amount and type of work performed by an employee for that day's shift.)

TIME CARD DATE + SHIFT NUMBER

PACKLIST NUMBER, OPERATION NUMBER + TIME CARD DATE

(This index was used by job progress applications)

The employee would enter his work in a time card application program. Over time this application was modernized to include bar codes and scanning. An employee could work on several packing list per shift. It the employee worked on five packing lists during that shift then five rows would enter five lines or rows in the time card table.) The employee could update the time card at any time until the supervisor finalized the entries.

When the employee entered a row on his time card, a job cost row would automatically be entered into job cost table by the time card application. When the time card was finalized, a payroll row would automatically be entered into the payroll table (not relevant in this case.)

Since changes may be made both before and after finalizing, the update application was very sophisticated. Since the system was real-time, all previously entries in all tables must be erased or changed and the changes then recorded.

Using the schedule category of the labor operation, the time card table successfully provided detail data to watch job as it progressed through the shop. The bad and reworked pieces provided an excellent opportunity to manage quality control.

# 3.0 Introductory SQL DML Statements

**Your documentation is required to display the Putty Header and you should use a snipping tool**

**Important**

The results should be easy to read and grade. The following is some guidelines that may help you minimize some formatting.

SQL String Functions - http://webspace.cs.odu.edu/~ibl/450/common/sqlstring.html

To\_Date Function - <http://www.techonthenet.com/oracle/functions/to_date.php>

Oracle TO\_DATE Function – SQL Syntax Examples - <http://www.mandsconsulting.com/oracle-to_date-function-sql-syntax-examples>

Oracle TO\_CHAR Function – SQL Syntax Examples (Most With Dates, TO\_DATE) - <http://www.mandsconsulting.com/oracle-to_char-function-sql-syntax-examples-most-with-dates-to_date>

Oracle Built in Functions - <http://beginner-sql-tutorial.com/oracle-functions.htm>

a. **I cannot grade wrapped text.** By default SQLPlus has a default line size of 80 characters. If your results are greater than 80 characters wide it will wrap to the next line. While you may correct the problem in Word, a better solution is change SQLplus's linesize, e.g., set linesize 150

b. A problem with the set linesize is that the text will be display past the right margin of the Putty screen. One solution would be to change Putty's font size. You may also change Word's font size or even switch to landscape.

d. The to-char function, e.g., see <http://www.techonthenet.com/oracle/functions/to_char.php>, can better display dates and times. For example,

Select to\_char(date\_shipped, 'Month DD, YYYY') from invoice; would display July 09, 2012

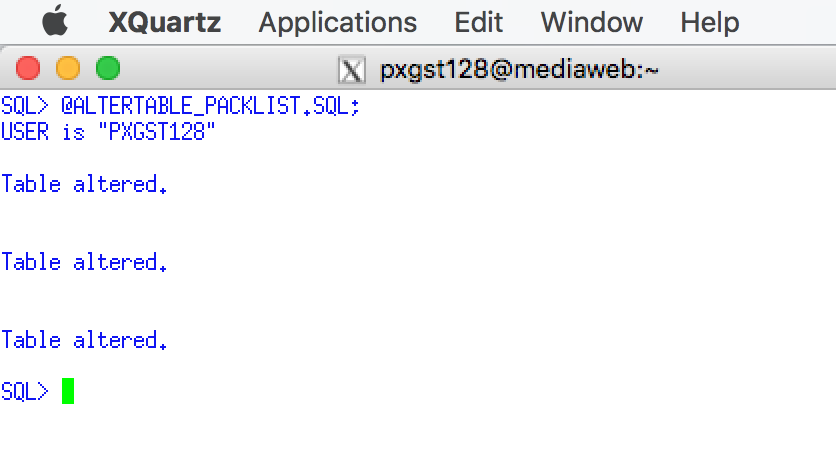
## 3.1 Questions - ALTER TABLE

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

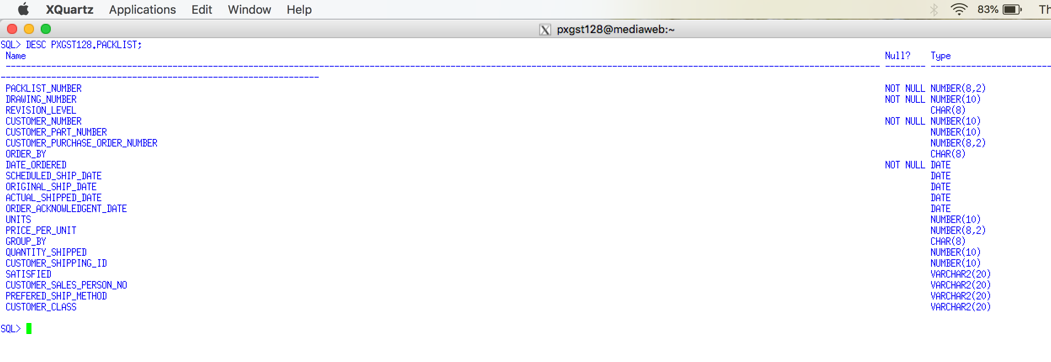
**A schema is a collection of database objects (as far as this hour is concerned—tables) associated with one particular database username. This username is called the schema owner, or the owner of the related group of objects. Given JPLST1.PACKINGLIST, the schema or owner name is JPLST1, then followed by a period, then followed the table or column name.**

**Requirement A.** Alter the Packing List Table to add the CUSTOMER SALES PERSON NO

PREFERRED SHIP METHOD, and CUSTOMER CLASS that you didn’t previously use. Document your efforts by describing the altered table. SQL Code which also displays your schema name🡺

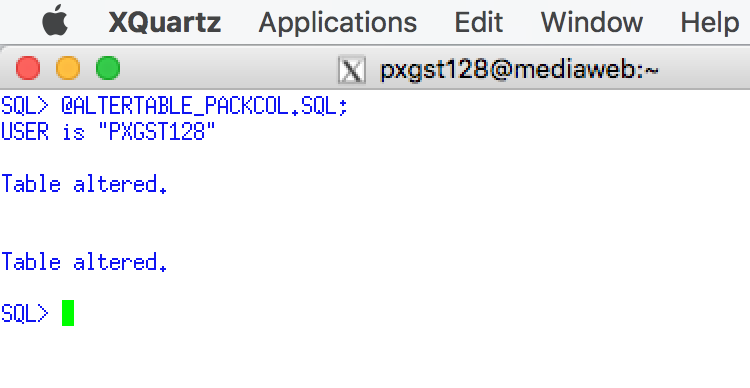


**Use the DESCRIBE command to document that the new column has been added. It is required to document the DESCRIBE command. You must a schema name, e.g., DESCRIBE JPLST1.PACKINGLIST. The schema name specifies the owner of the table. Leave blank if you encounter an error 🡺**



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

**Requirement B.** Alter the Packing list table to increase the size of any existing column. Document your efforts by describing the altered table. SQL Code which also displays your schema name 🡺



**Use the DESCRIBE command to document that the size of the column has been altered It is required to document the DESCRIBE command. You must a schema name, e.g., DESCRIBE JPLST1.PACKINGLIST. Leave blank if you encounter an error 🡺**

## 3.2 UPDATE TABLE

**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)

**UPDATE (to change a field for one row or multiple rows)**

**UPDATE JPL1ST1.FACULTY**

**SET FPAY = 100000**

**WHERE FNAME = ‘PACKY’;**

**This example will replace PACKY's meager pay with a more appropriate salary.**

**UPDATE JPLST1.FACULTY**

**SET FPAY = JPLST1.FPAY \* 1.10**

**WHERE FNAME = ‘PACKY’;**

**This example will replace PACKY's meager pay with a 10% raise.**

**UPDATE JPLST1.FACULTY**

**SET FPAY = FPAY \* 1.10**

**WHERE FDEP = ‘CIS’;**

**Multiple rows may be updated simultaneously. Therefore care should be taken when updating a field when it is being retrieve by something else than the primary key.**

**UPDATE JPLST1.FACULTY**

**SET FPAY = 100000,**

**FDEP = 'CIS'**

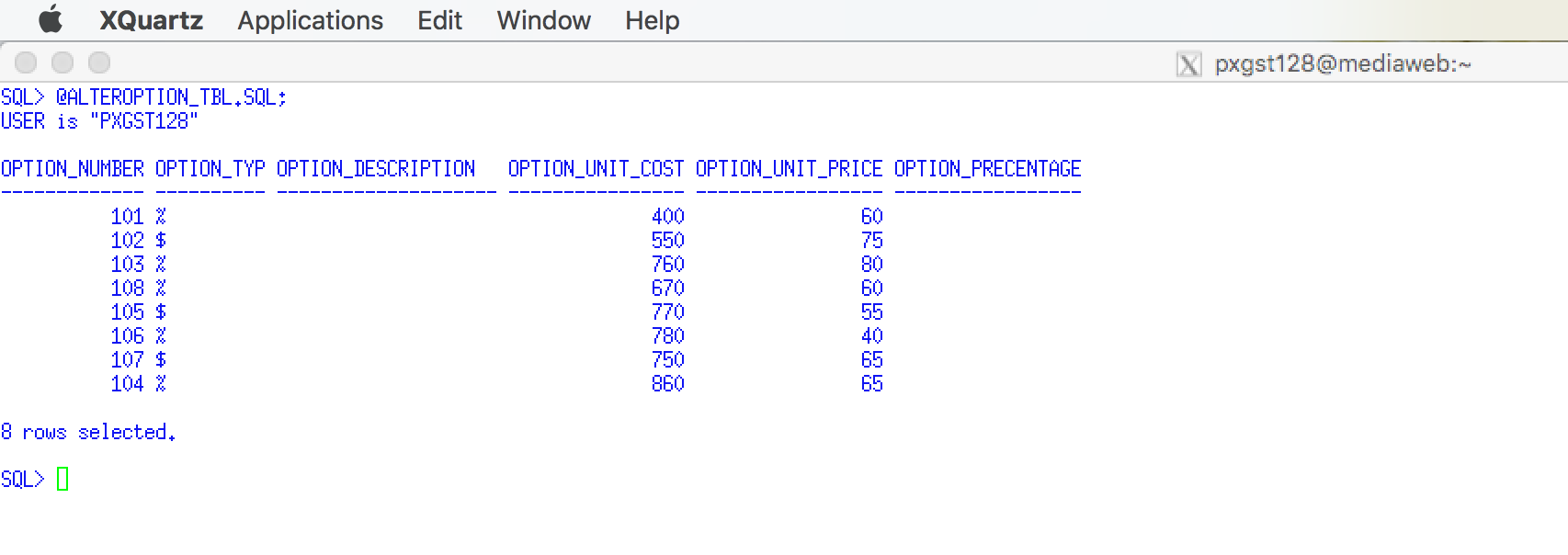
**WHERE FNAME = ‘PACKY’;**

**Multiple columns can be updated with one UPDATE command. The key word SET is NOT repeated. Each assignment is separated by a comma, except the last column**

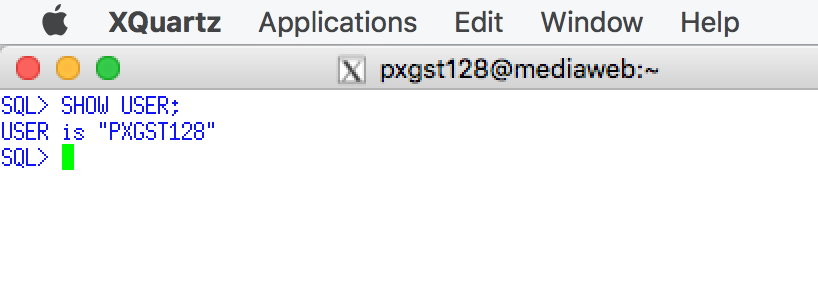
## 3.3 Questions - UPDATE TABLE

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

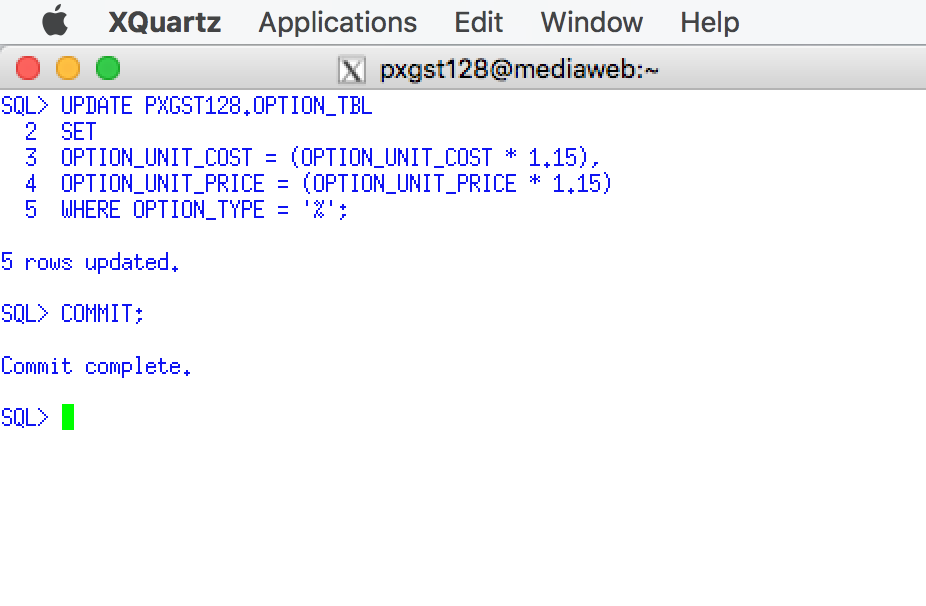
Select all the records and columns to document your option table values BEFORE this requirement. **You must a schema name, e.g., SELECT \* from JPLST1.OPTION. The schema name specifies the owner of the table.** SQL Code which also displays your schema name 🡺



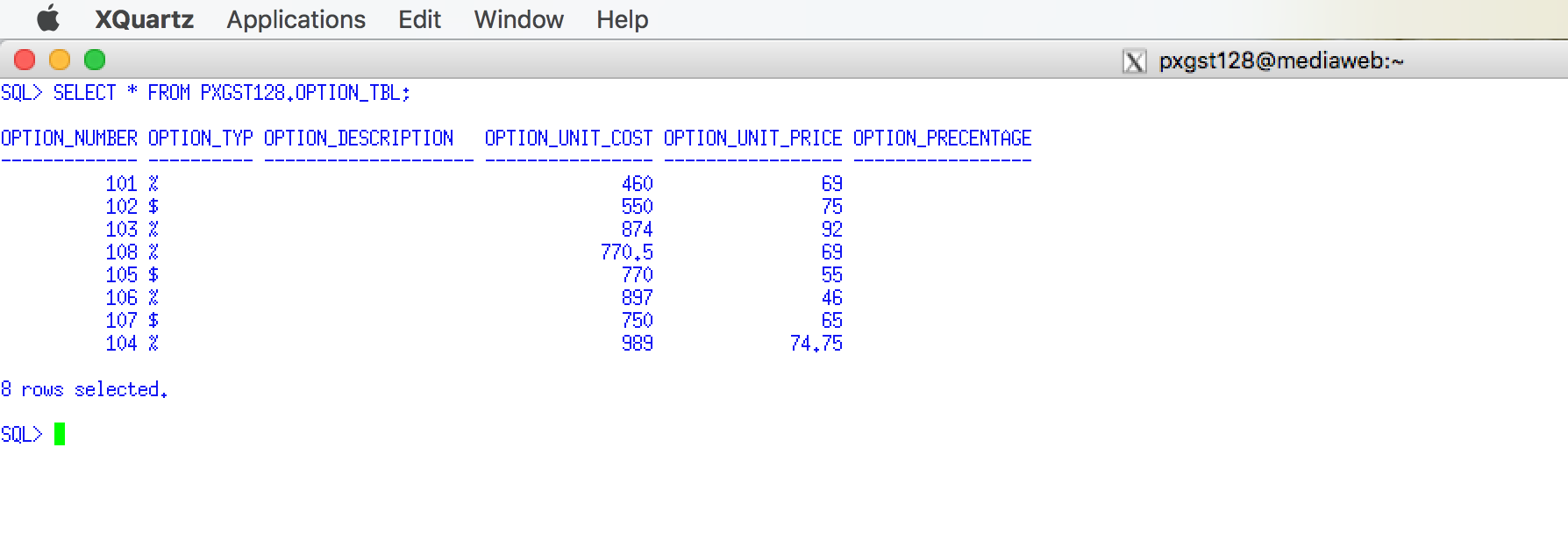
**Requirement C.**  Increase the only the **%** options for each option row by 15%. Select all the records to document your result. SQL Code which also displays your schema name which also displays your schema name 🡺



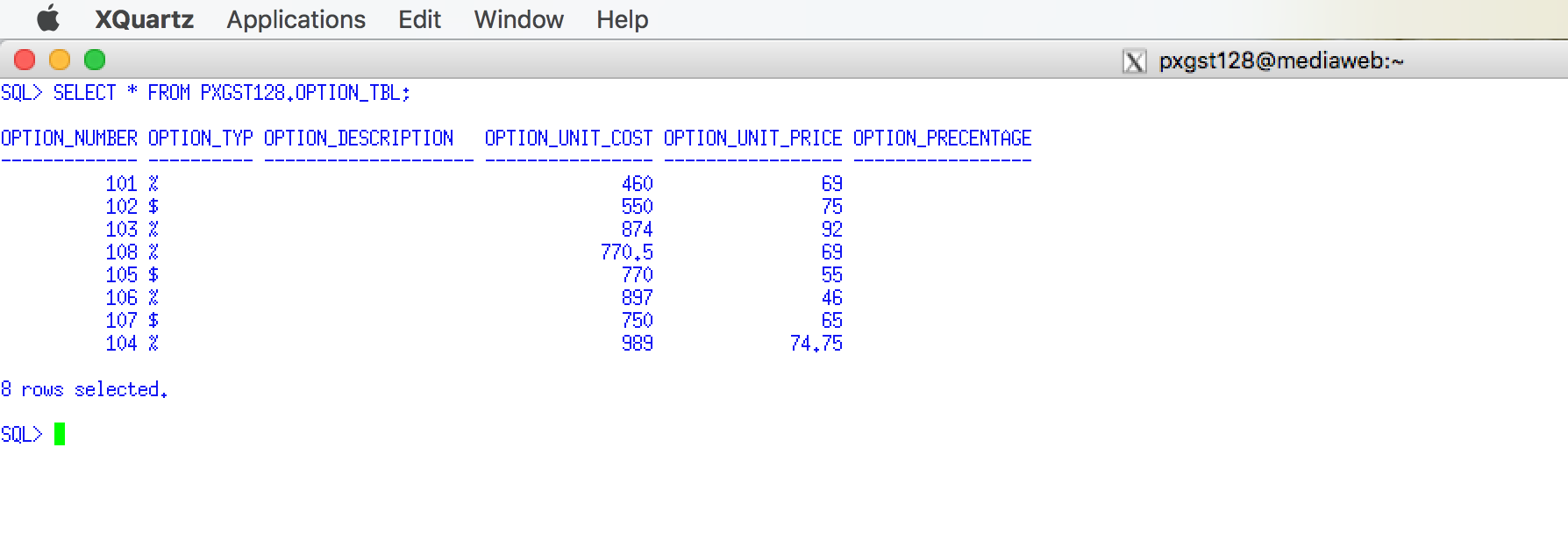
SQL Code 🡺



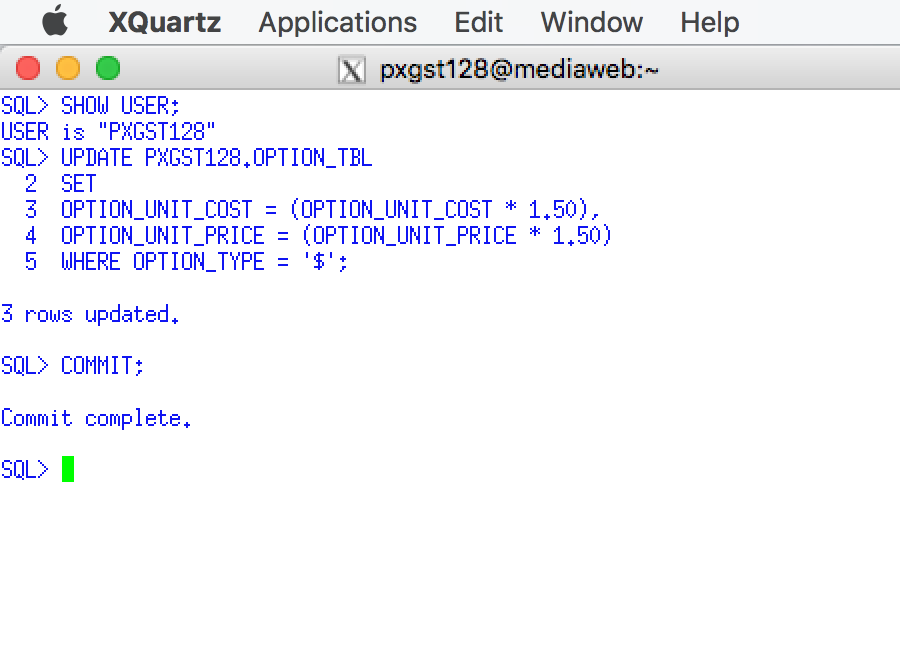
Select all the records to document your option table values AFTER Executing Requirement C. SQL Code which also displays your schema name 🡺



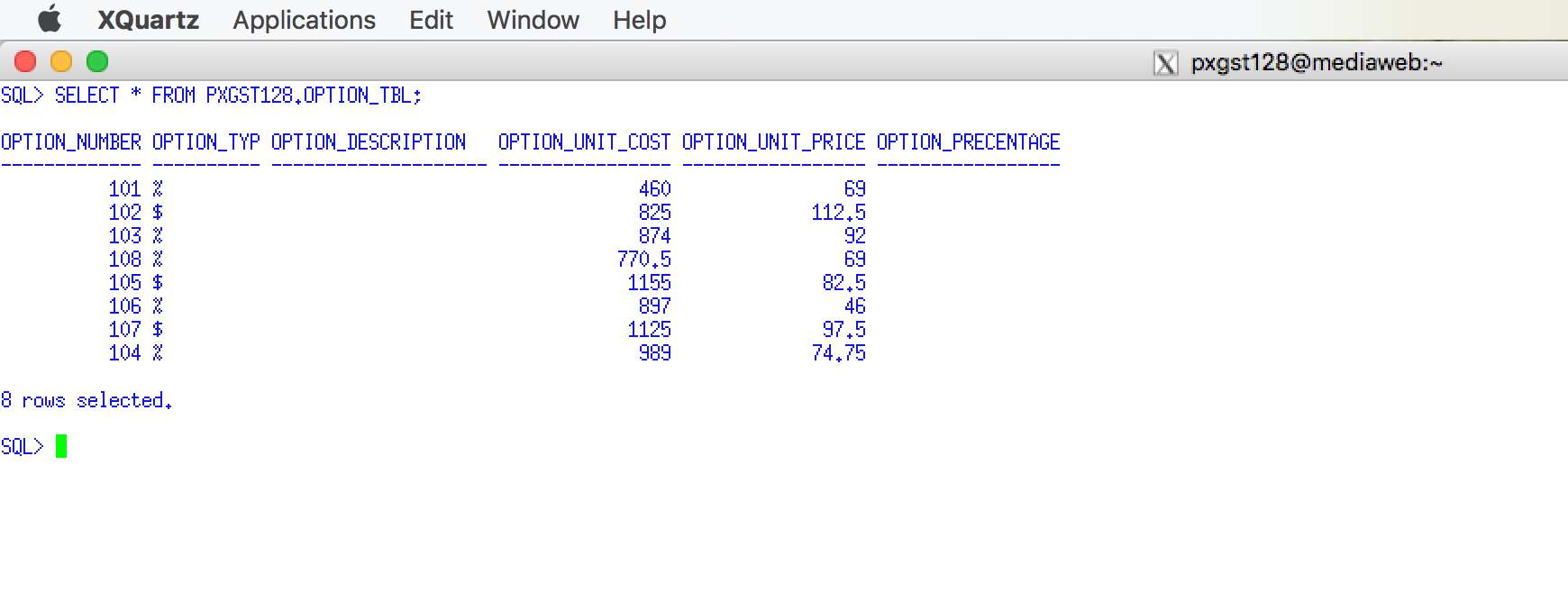
Select all the records and columns to document your option table values BEFORE Requirement D. SQL Code 🡺



**Requirement D.** Increase only the **$** options for each option row by 50%. Select all the records to document your result. SQL Code which also displays your schema name 🡺



Select all the records and columns to document your option table values AFTER Executing Requirement D. SQL Code which also displays your schema name 🡺



## 3.4 Questions - DELETE TABLE

**DELETE FROM ONE OR MORE ROWS FROM A TABLE**

**DELETE FROM FACULTY;**

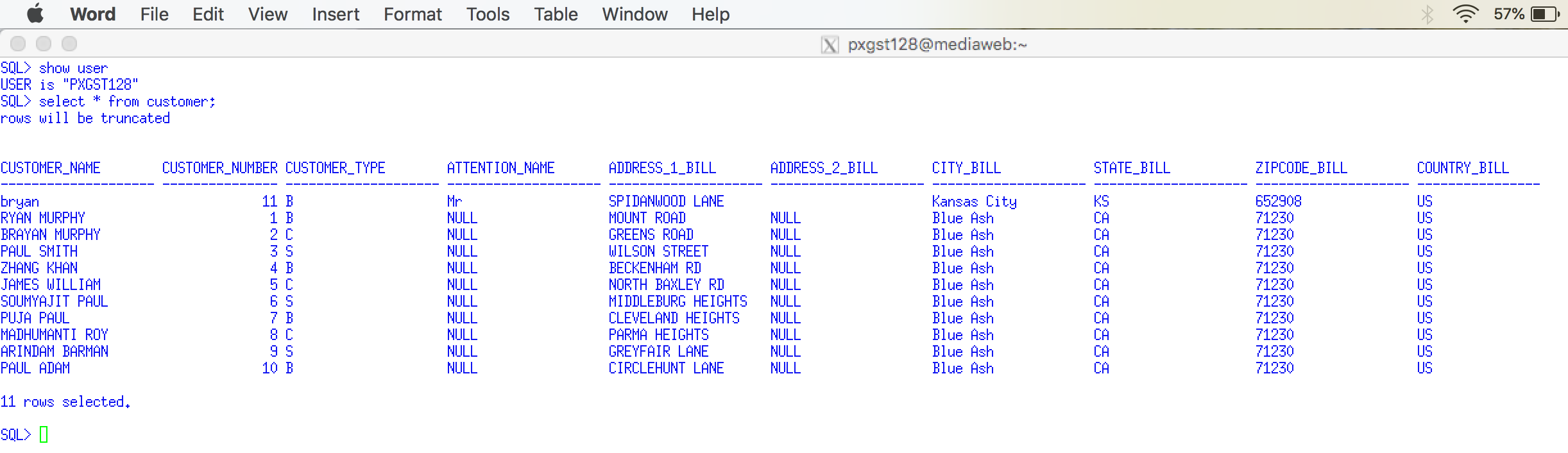
**It would highly unlikely that you would delete all of the rows from a given table. If you accidentally forget the WHERE clause, you can reverse the operation up to the point where you COMMIT your work by using the ROLLBACK command. It would highly unlikely that you would delete all of the rows from a given table. If you accidentally forget the WHERE clause, you can reverse the operation up to the point where you COMMIT your work by using the ROLLBACK command.**

**DELETE FROM FACULTY WHERE FID =8;**

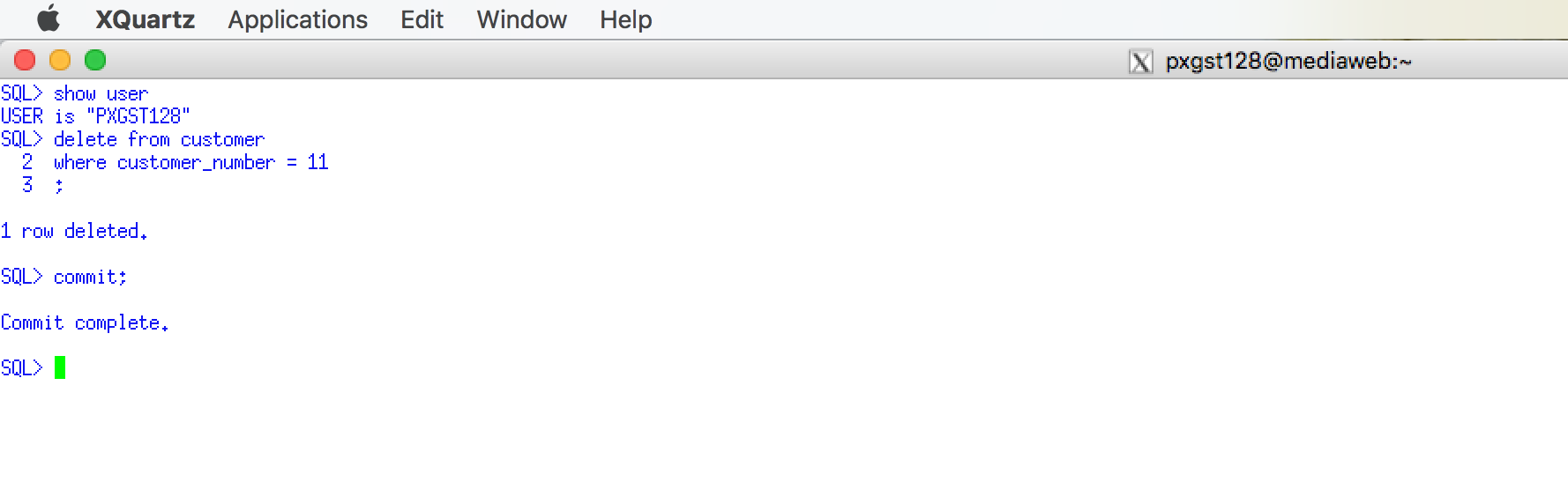
**This statement will delete the FACULTY row where a FID = 8**

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

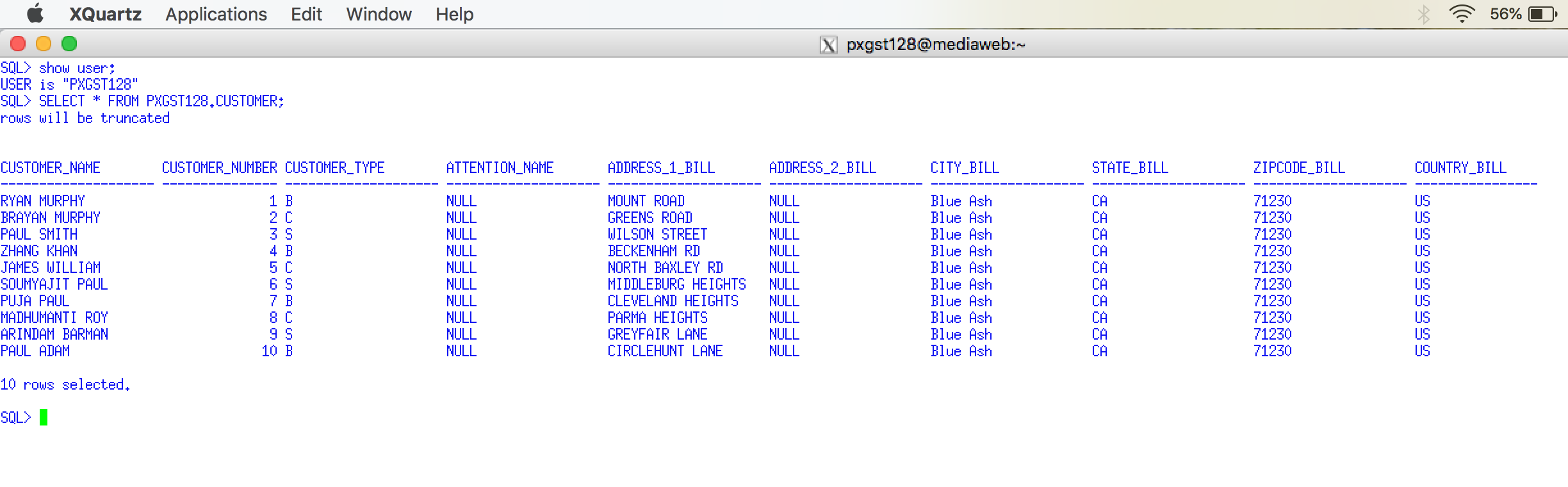
Select all the records and columns to document your customer table. SQL Code which also displays your schema name 🡺



**Requirement E.** Delete any customer record. SQL Code which also displays your schema name 🡺

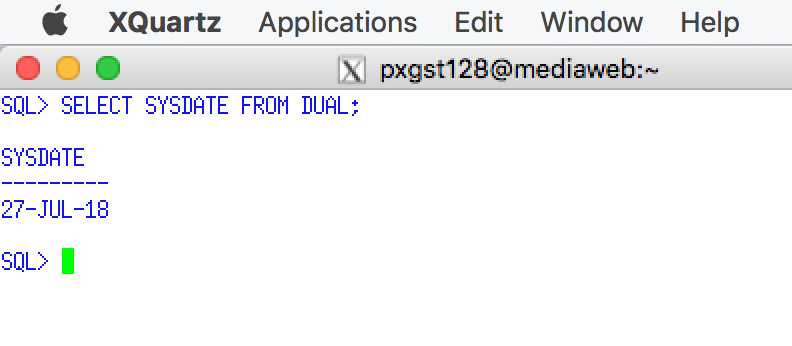


Select all the records and columns to document your customer table AFTER Requirement E. SQL Code which also displays your schema name 🡺



SYSDATE returns the current date and time set for the operating system on which the database resides. The datatype of the returned value is DATE, and the format returned depends on the value of the NLS\_DATE\_FORMAT initialization parameter. Execute the following **SELECT SYSDATE FROM DUAL;**

**Formatted Results of command - Leave blank if you encounter an error 🡺**



The Oracle/PLSQL CURRENT\_TIMESTAMP function returns the current date and time in the time zone of the current SQL session as set by the ALTER SESSION command. It returns a TIMESTAMP WITH TIME ZONE value.

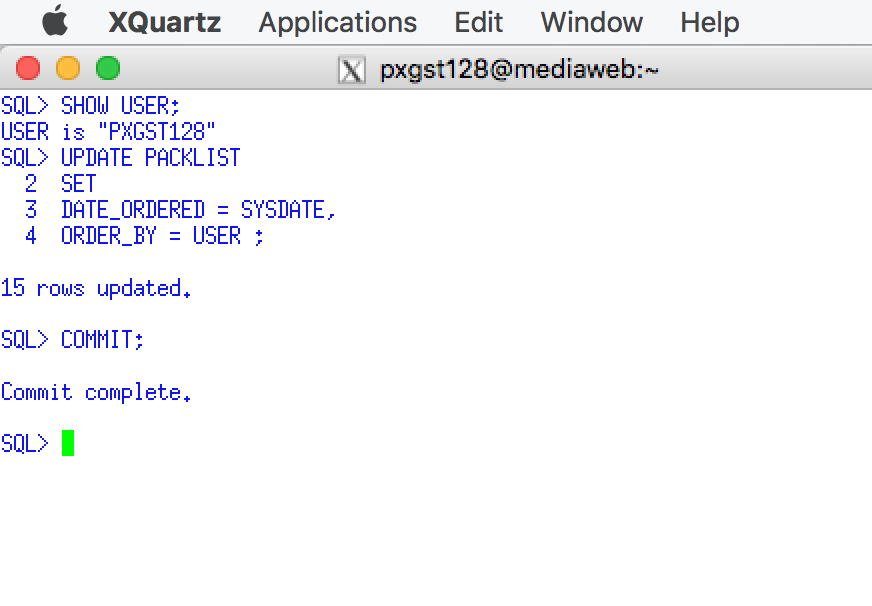
The Oracle/PLSQL USER function returns the user\_id from the current Oracle session. Execute the following **SELECT USER FROM DUAL;**

**Formatted Results of command - Leave blank if you encounter an error 🡺**



## 3.5 Questions - ORDER BY Clause

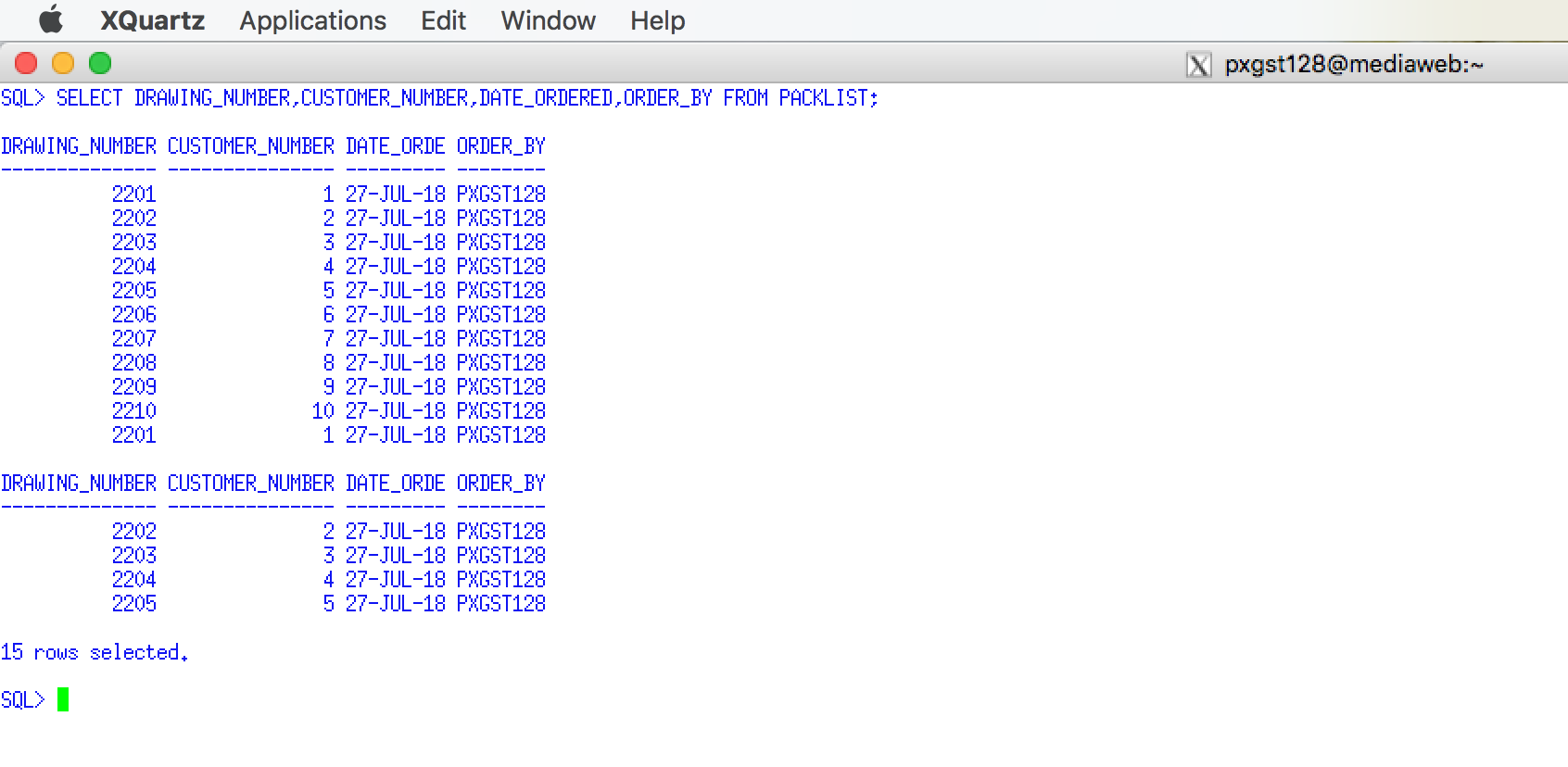
**Requirement F.** Using the Packing List Table change all DATE ORDERED column content to the SYSDATE and ORDERED BY column to the system user. Select all the records to document your result. SQL Code which also displays your schema name 🡺



**Use a graphical snipping tool to document an image of these requirements below.**

**Make sure your Putty banner is displayed.**

**Formatted Results of command - Leave blank if you encounter an error 🡺**



The SELECT statement is used to retrieve and values from one or more tables. The basic format of the select statement is:

SELECT predicate fieldnames FROM table name [IN external database name]

or

SELECT {COLUMNS OR EXPRESSIONS TO BE DISPLAYED)

FROM (THIS TABLE)

ORDER BY (WHICH COLUMN IS USE TO SORT THE DATA)

WHERE (WHICH ROWS ARE TO BE RETRIEVED)

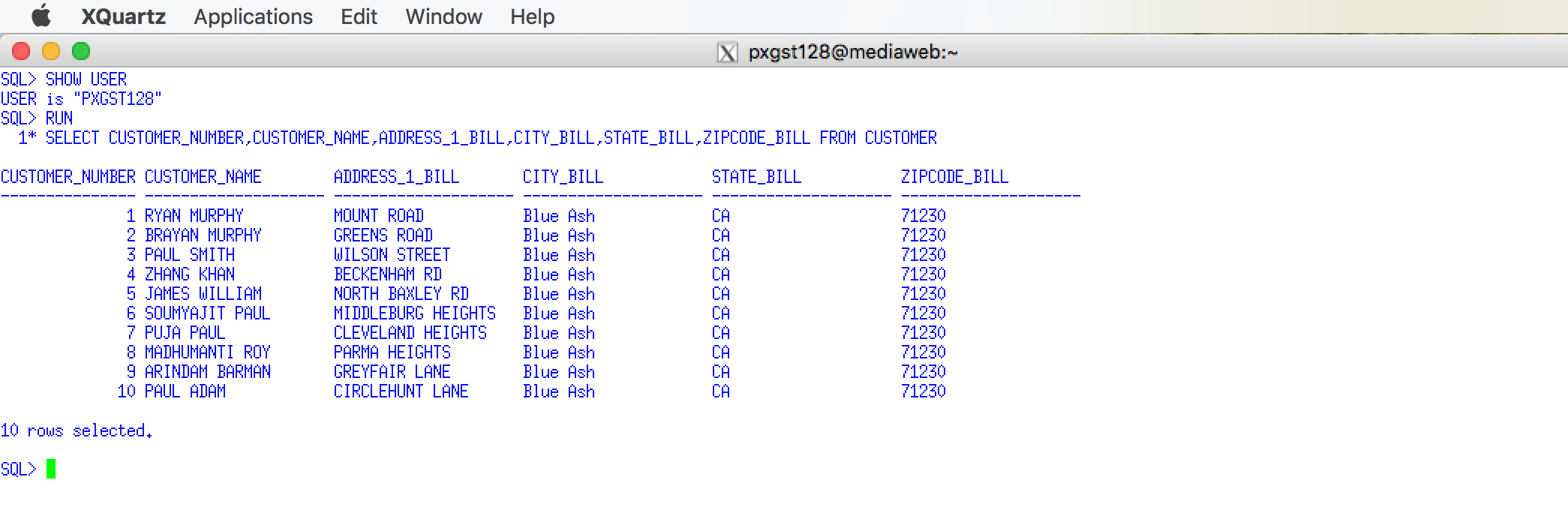
Example: SELECT \* FROM STUDENT;

Retrieve all records and all fields will be retrieved from the STUDENT table.

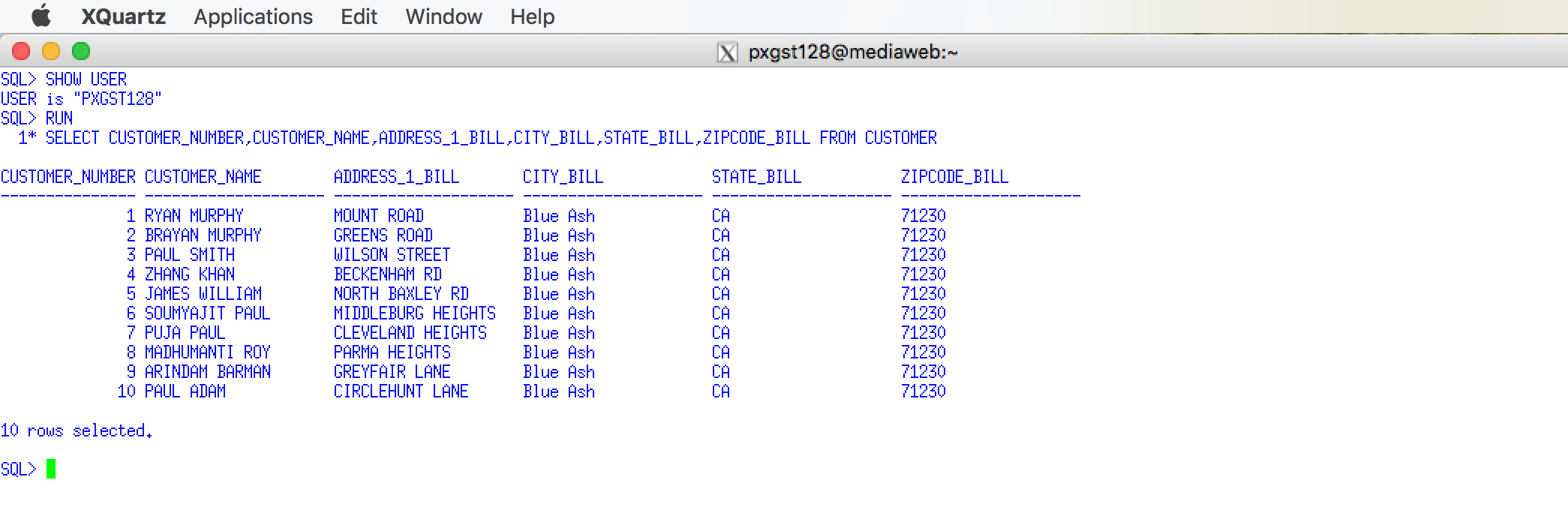
The asterisk '\*' indicates that all columns are to be retrieved. If the table STUDENT is in the current schema and database.

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

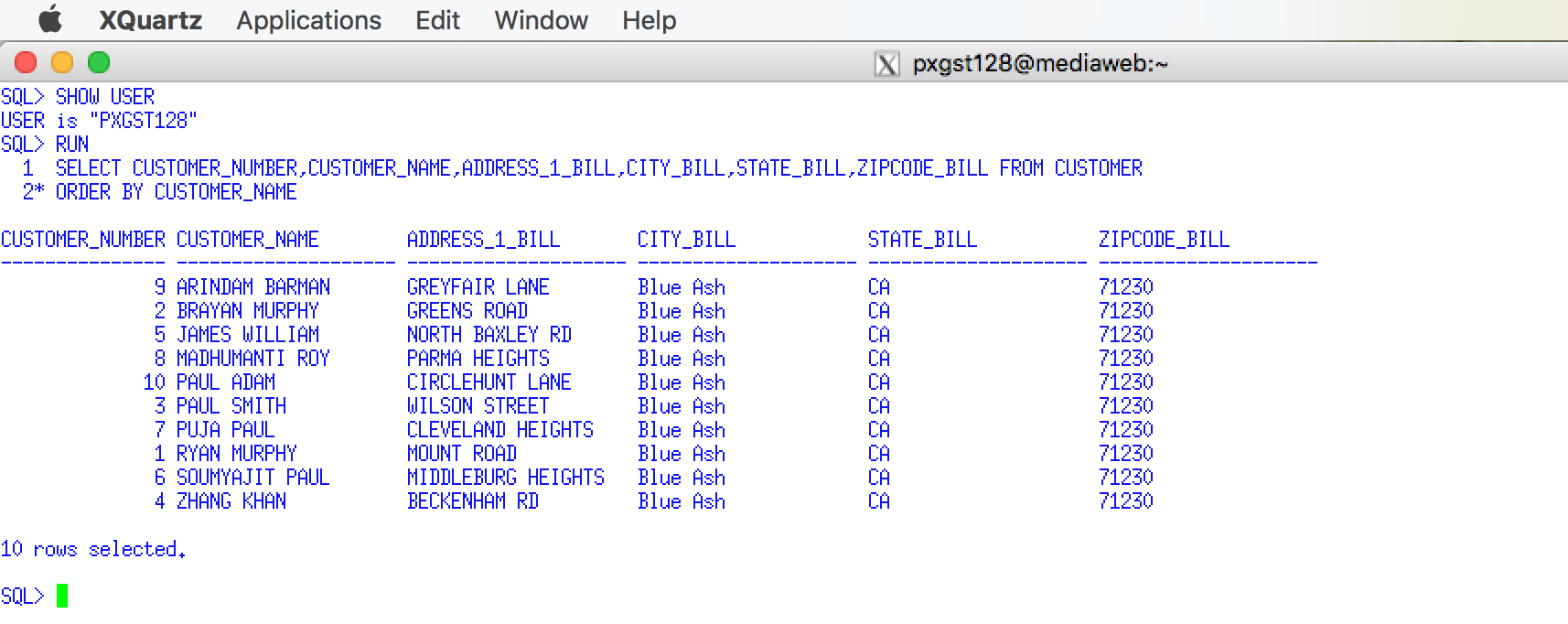
**Requirement G.** Select the Customer Table and Print the Account Number, Customer's Name, Address, City State, and Zip code in record order (not sorted). SQL Code which also displays your schema name 🡺



**Formatted Results of command - Leave blank if you encounter an error 🡺**



**Requirement H.** Select the Customer Table and Print the Account Number, Customer's Name, Address, City State, and Zip code in Customer Name order. SQL Code which also displays your schema name 🡺



**Formatted Results of command - Leave blank if you encounter an error 🡺**

SQL Conditions, Expressions, and Operators - PART 1 - SQL LESSON 6 - http://www.youtube.com/watch?v=lPzW1H1nbaE&feature=relmfu

SQL Conditions, Expressions, and Operators - PART 2 - SQL LESSON 7

http://www.youtube.com/watch?v=3mnc7LROq-0&feature=relmfu

SQL Conditions, Expressions, and Operators - PART 3 - SQL LESSON 8 -

http://www.youtube.com/watch?v=zUAE2Ao1m-M&feature=relmfu

SQL Conditions, Expressions, and Operators - PART 4 - SQL LESSON 9 -

Sorting in SQL Using Order by Clause - Part 1 - SQL LESSON 27 -

http://www.youtube.com/watch?v=0ljgw-EFqJE&feature=relmfu

Sorting in SQL Using Order by Clause - Part 2 - SQL LESSON 28

http://www.youtube.com/watch?v=PnlgMtMh4i0&feature=channel&list=UL

Sorting in Descending Order in SQL Using Order by Desc Clause - SQL LESSON 29

http://www.youtube.com/watch?v=kBDIf6fEt-E&feature=channel&list=UL

## 3.6 WHERE Clause

**WHERE (restricting the row retrieved)**

SELECT FACULTY.FID, FACULTY.FNAME FROM FACULTY

WHERE FACULTY.FID = 34;

Retrieve any row(s) that has a FACULTY.FID = 34 and display the FCULTY.FID and FACULTY.FNAME from the table FACULTY. This example is using the Table\_name.Column\_name format. Normally specifying the table name or schema name is optional if you retrieve data from one table or the table is in the current schedule, e.g., it is your user table that you created or own. .

SELECT \* FROM FACULTY WHERE FACULTY.FDEP = ‘CIS’;

Retrieve any row(s) that is a member of the CIS department from the FACULTY table and display all columns. When specifying text fields you must enclose the selection criteria in quotes.

SELECT \* FROM FACULTY WHERE FACULTY.FDEP != ‘CIS’;

or

SELECT \* FROM FACULTY WHERE NOT FACULTY.FDEP = ‘CIS’;

Retrieve any rows(s) that is NOT a member of the CIS department from the FACULTY table and display all fields. There are many ways to express the same syntax in Oracle SQL and the "not equals" operator may be expressed as "<>" or "!=".

SELECT \* FROM FACULTY WHERE FACULTY.FPAY >= 10000;

Retrieve any row(s) for faculty having pay greater than or equal to 10000 from the FACULTY table and display all fields.

SELECT \* FROM FACULTY

WHERE FACULTY.FPAY >= 10000 AND FACULTY.DEP = ‘CIS’;

Retrieve any rows(s) for a member of the CIS department whose faculty pay greater than or equal to 10000.

SELECT \* FROM FACULTY

WHERE FACULTY.PAY BETWEEN 10000 AND 20000;

Retrieve any row(s) for a member of the CIS department whose faculty pay greater than or equal to 10000 and less than or equal to 20000. Is the value in a range?

SELECT \* FROM FACULTY

WHERE FACULTY.PAY NOT BETWEEN 10000 AND 20000;

Retrieve any row(s) for a member of the CIS department whose faculty pay greater than or equal to 20000 or less than or equal to 10000. Is the value outside in a range, or not in the range?

SELECT \* FROM FACULTY

WHERE FACULTY.DEP =’CIS’ OR FACULTY.DEP = ‘MKT’;

Retrieve any row(s) for a member of the CIS or MKT departments

SELECT \* FROM FACULTY

WHERE FACULTY.DEP =’CIS’ OR FACULTY.DEP = ‘MKT’

OR FACULTY.DEP = ‘ACC’;

Retrieve any row(s) for a member of the CIS, MKT or ACC departments

OR

SELECT \* FROM FACULTY

WHERE FACULTY.DEP IN (‘CIS’,’MKT’, ‘ACC’);

The IN operator allows you to specify multiple values in a WHERE clause. The SQL IN operator is used to help reduce the need for multiple OR conditions in a SELECT, INSERT, UPDATE, or DELETE statement. It is frequently used with subqueries to be demonstrated later.

SELECT \* FROM FACULTY

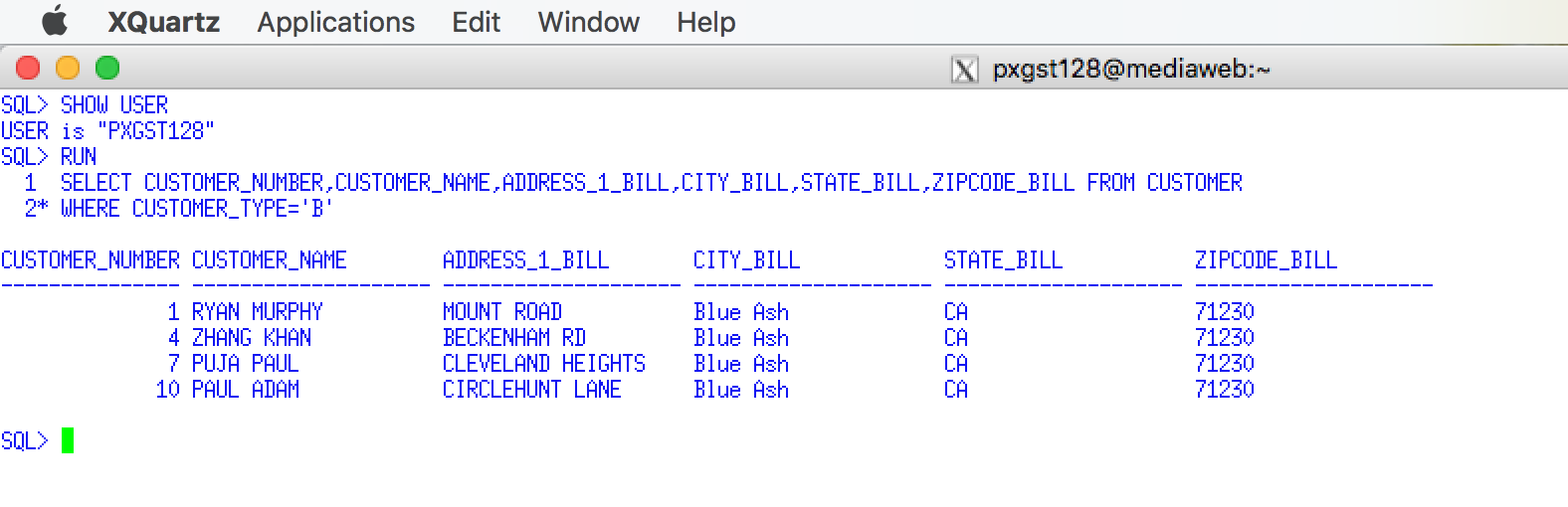
WHERE FACULTY.DEP NOT IN (‘CIS’,’MKT’, ‘ACC’);

Retrieve any row(s) who is NOT a member of the CIS, MKT or ACC departments

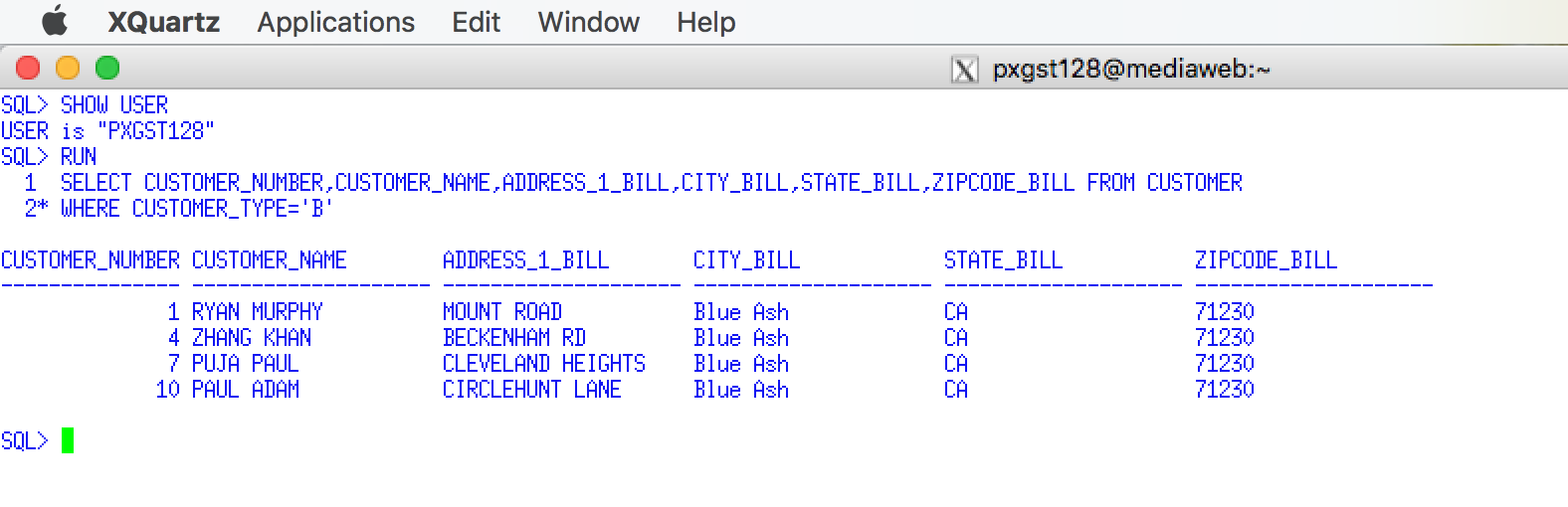
## 3.7 Questions - WHERE Clause

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

**Requirement I.**  Select the Customer Table and Print the Customer Type, Account Number, Customer's Name, Address, City State, and Zip code for only blanket customers. SQL Code which also displays your schema name 🡺



**Formatted Results of command - Leave blank if you encounter an error 🡺**



SQL Logical Operators - PART 1 - SQL LESSON 10 -

http://www.youtube.com/watch?v=jhSm8tFvWSI&feature=relmfu

SQL Logical Operators - PART 2 - SQL LESSON 11 -

http://www.youtube.com/watch?v=oVF1s8WiDDk&feature=fvwrel

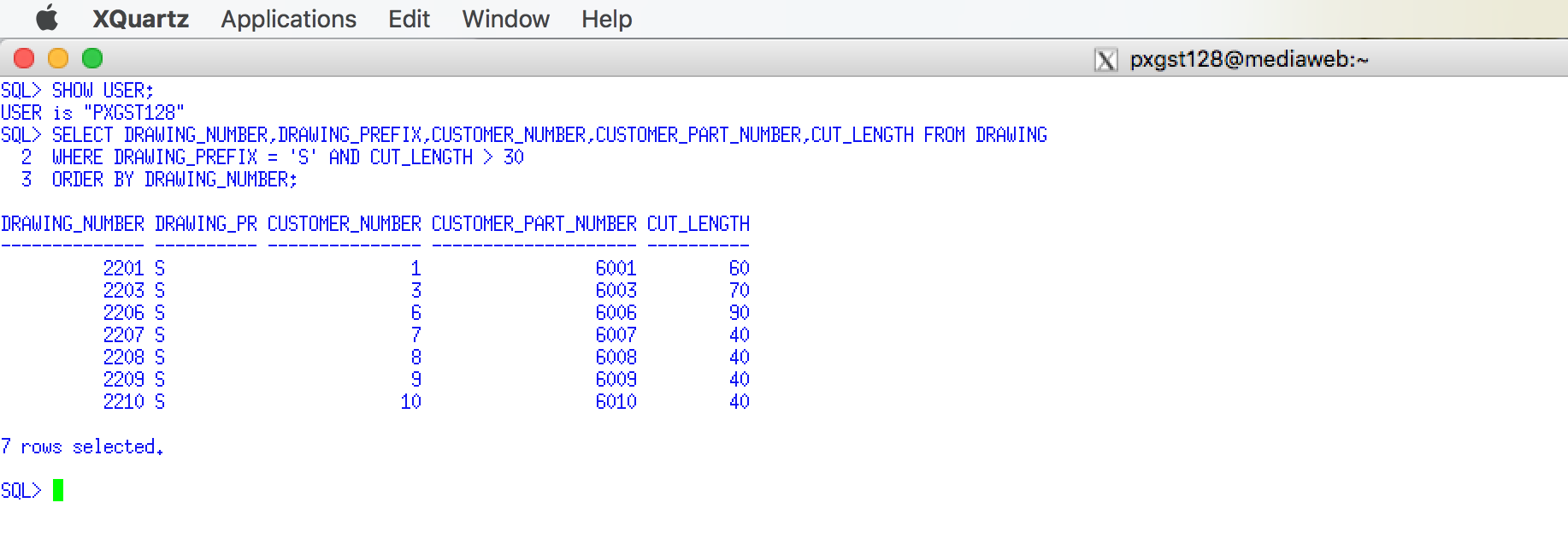
SQL Logical Operators - PART 3 - SQL LESSON 12 -

http://www.youtube.com/watch?v=btZ39GrWPuQ&feature=relmfu

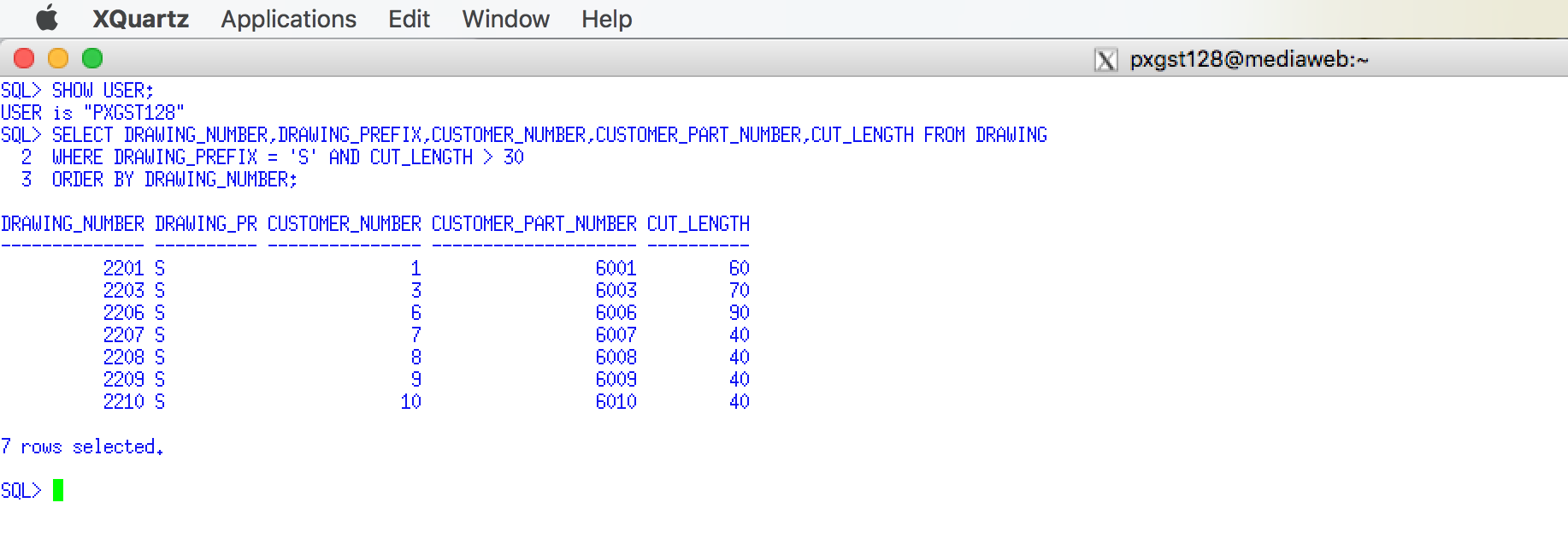
## 3.8 Comparison Operators

**Requirement J.** Select the Drawing table and Print the Drawing Number, Drawing Prefix, Customer Number, Customer Part Number, and Cut Length in drawing number order, for those drawings with an S prefix and whose cut length is above 30 inches. (Insert data if needed to display some rows. DO NOT DISPLAY NO OUTPUT) (Use of logical operators)

SQL Code which also displays your schema name 🡺



**Formatted Results of command - Leave blank if you encounter an error 🡺**



<http://www.1keydata.com/sql/sql-concatenate.html>

There are other comparison keywords available in SQL which are used to enhance the search capabilities of a SQL query. They are "IN", "BETWEEN...AND", "IS NULL", "LIKE".

|  |  |
| --- | --- |
| **Comparison Operator** | **Description** |
| LIKE | Column value is similar to specified character(s). |
| IN | Column value is equal to any one of a specified set of values. |
| BETWEEN...AND | Column value is between two values, including the end values specified in the range. |
| IS NULL | Column value does not exist. |

## 3.9 LIKE operator

The LIKE operator is used to list all rows in a table whose column values match a specified pattern. It is useful when you want to search rows to match a specific pattern, or when you do not know the entire value. For this purpose we use a wildcard character '%'.

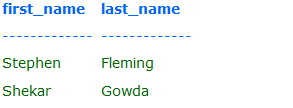
For example: To select all the students whose name begins with 'S'

**SELECT first\_name, last\_name**

**FROM student\_details**

**WHERE first\_name LIKE 'S%';**

The output would be similar to:



The above select statement searches for all the rows where the first letter of the column first\_name is 'S' and rest of the letters in the name can be any character.

There is another wildcard character you can use with LIKE operator. It is the underscore character, ' \_ ' . In a search string, the underscore signifies a single character.

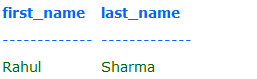
For example: to display all the names with 'a' second character,

**SELECT first\_name, last\_name**

**FROM student\_details**

**WHERE first\_name LIKE '\_a%';**

The output would be similar to:



NOTE:Each underscore act as a placeholder for only one character. So you can use more than one underscore. Eg: ' \_\_i% '-this has two underscores towards the left, 'S\_\_j%' - this has two underscores between character 'S' and 'i'.

Other examples of the LIKE Clause

SELECT \* FROM FACULTY

WHERE FACULTY.FNAME LIKE ‘P%’

Retrieve any record(s) with a faculty name beginning with the letter P followed by any sequence of characters.

SELECT \* FROM FACULTY

WHERE FACULTY.FNAME LIKE ‘%P%’;

Retrieve any record(s) with a faculty name that contains a capital the letter P.

SELECT \* FROM FACULTY

WHERE FACULTY.FNAME LIKE ‘P\_\_’;

Retrieve any record(s) with a faculty name beginning with the letter P and followed by any two single characters.

**SQL BETWEEN ... AND Operator**

The operator BETWEEN and AND, are used to compare data for a range of values.

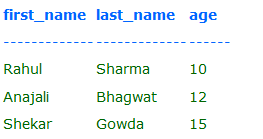
For Example: to find the names of the students between age 10 to 15 years, the query would be like,

**SELECT first\_name, last\_name, age**

**FROM student\_details**

**WHERE age BETWEEN 10 AND 15;**

The output would be similar to:



## 3.10 IN operator

**SQL IN Operator:**

The IN operator is used when you want to compare a column with more than one value. It is similar to an OR condition.

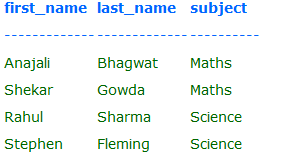
For example: If you want to find the names of students who are studying either Maths or Science, the query would be like,

SELECT first\_name, last\_name, subject

FROM student\_details

WHERE subject IN ('Maths', 'Science');

The output would be similar to:



You can include more subjects in the list like ('maths','science','history')

NOTE: The data used to compare is case sensitive.

**SQL IS NULL Operator**

A column value is NULL if it does not exist. The IS NULL operator is used to display all the rows for columns that do not have a value.

For Example: If you want to find the names of students who do not participate in any games, the query would be as given below

SELECT first\_name, last\_name

FROM student\_details

WHERE games IS NULL

There would be no output as we have every student participate in a game in the table student\_details, else the names of the students who do not participate in any games would be displayed.

## 3.11 TO\_DATE() Function (Specifying a DATE in a WHERE clause.

SELECT \* FROM FACULTY

WHERE DATE\_HIRED > TO\_DATE('2015-07-07', 'YYYY-MM-DD');

Retrieve any record(s) with for a faculty member being hired after 7/7/2015. The previous examples of INSERT and UPDATE used the TO\_DATE() function to convert a date value into a format that Oracle understands, e.g., the default Oracle date format is 07-JUL-15 where are using a YYYY-MM-DD format. Therefore the WHERE clause must use the TO-DATE() function.

## 3.12 BETWEEN/AND Operators

SELECT \* FROM FACULTY

WHERE DATE\_HIRED BETWEEN TO\_DATE('2000-01-01', 'YYYY-MM-DD') AND

TO\_DATE('2015-07-07', 'YYYY-MM-DD');

Retrieve any record(s) with for a faculty member being hired in the period hired between 1/1/2000 and 7/7/2015.

Display the 50% of the most recent faculty members hired. The BOTTOM clause does not sort the data.

SELECT FID, FNAME, FDEP, TO\_CHAR(DATE\_HIRED, "MM'DD'YYYY')

FROM FACULTY

WHERE DATE\_HIRED BETWEEN TO\_DATE('2000-01-01', 'YYYY-MM-DD') AND

TO\_DATE('2015-07-07', 'YYYY-MM-DD');

Retrieve any record(s) with for a faculty member being hired in the period hired between 1/1/2000 and 7/7/2015 and display the columns FID, FNAME, FDEP, and DATE\_HIRED formatted as "MM'DD'YYYY'. The TO\_CHAR function is used on the SELECT line to format the date values to be displayed. The TO\_DATE function is used on the WHERE line to format the date values for Oracle to internally understand.

**ORDER BY (ordering the rows retrieved)**

SELECT \* FROM FACULTY ORDER BY FACULTY.FNAME;

Retrieve all records for faculty and sort the records in ascending name order. The ORDER BY does not require any key fields or indexes. If an adequate index does not exist the data will be automatically sorted.

SELECT \* FROM FACULTY ORDER BY FACULTY.FNAME DESC;

Retrieve all faculty records and sort these records in descending name order.

SELECT \* FROM FACULTY ORDER BY FACULTY.DEP, FACULTY.FNAME;

Retrieve all faculty records and sort these records first by order of the department name and then, within a department, ordered by faculty name.

SELECT \* FROM FACULTY

WHERE FACULTY.FDEP = ‘CIS’

ORDER BY FDEP, FNAME;

The WHERE and the ORDER BY clause may specify multiple columns. For example, retrieve all CIS faculty records and sort first by FDEP (categorically group by FDEP, and then within each department sort by faculty name.

## 3.14 ROWNUM, TOP, BOTTOM (Limiting the number of rows retrieved)

The WHERE clause will retrieve all that meets a certain condition. This may be no rows or hundreds of rows. To limit the numbers of rows retrieved use the ROWNUM value.

SELECT \* FROM FACULTY

WHERE ROWNUM <=5 and FDEP ='CIS';

Only the first five rows or less who are members of the CIS department will be retrieved. If you use the ORBER BY CLAUSE you can specify the order of the rows.

SELECT \* FROM FACULTY

WHERE ROWNUM <=5 and FDEP ='CIS'

ORDER BY DATE\_HIRED;

Display the FIVE most senior faculty members

SELECT TOP 2 \* FROM FACULTY

ORDER BY DATE\_HIRED;

Display the top two most senior faculty members. The TOP clause does not sort the data.

SELECT TOP 50 PERCENT \* FROM FACULTY

ORDER BY DATE\_HIRED;

Display the top 50% most senior faculty members. The TOP clause does not sort the data.

SELECT BOTTOM 2 \* FROM FACULTY

ORDER BY DATE\_HIRED;

Display the two most recent faculty members hired. The BOTTOM clause does not sort the data.

SELECT BOTTOM 50 PERCENT \* FROM FACULTY

ORDER BY DATE\_HIRED;

Membership Conditions in SQL Where Clause (Subqueries) - Part 1 - SQL LESSON 18 -

http://www.youtube.com/watch?v=7MKQGXcw1aw&feature=relmfu

Membership Conditions in SQL Where Clause (Subqueries) - Part 2 - SQL LESSON 19 - http://www.youtube.com/watch?v=\_X1oiNqDVeE&feature=fvwrel

Range Conditions in SQL Where Clause (Between) - SQL LESSON 20 -

http://www.youtube.com/watch?v=z5YCPGzq-Ac&feature=fvwrel

Matching Conditions in SQL Where Clause (IN) - Part 1 - SQL LESSON 21 - http://www.youtube.com/watch?v=MIHA7ptVom0&feature=relmfu

Matching Conditions in SQL Where Clause (IN) - Part 2 - SQL LESSON 22 -

http://www.youtube.com/watch?v=bK1EEnTFQl0&feature=relmfu

SQL Comparison Conditions - SQL LESSON 22 B -

http://www.youtube.com/watch?v=iSX4Yb3\_TPY&feature=relmfu

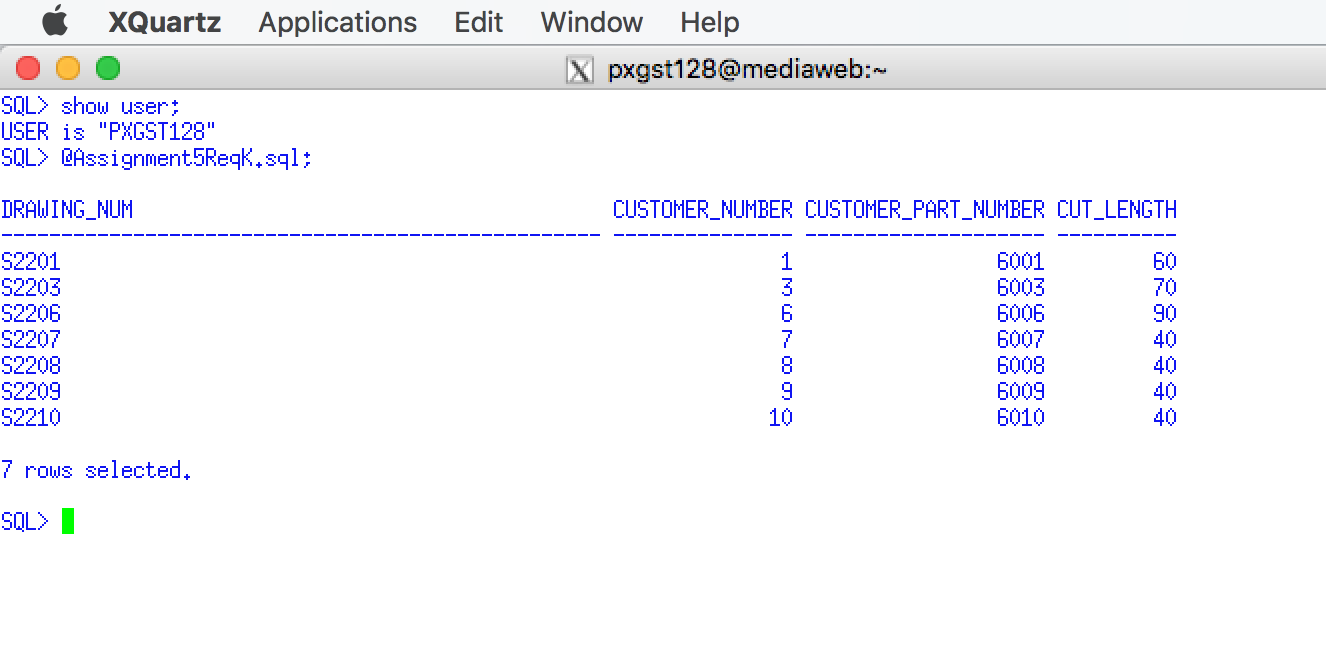
Concept of a Null Value in SQL - Part 1 - SQL LESSON 23 -

http://www.youtube.com/watch?v=XoH-T\_0Ik9k&feature=relmfu

## 3.15 Questions - Concatenation

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

**Requirement K.** Repeat Requirement J, but display the Drawing Number and the Prefix as a concatenated field and use appropriate column aliases. SQL Code which also displays your schema name 🡺



## 3.16 Questions- RTRIM() Function

**Requirement L.** Repeat Requirement H but use the RTRIM function and Concatenation to combine address, city, state and zip code as appropriate. However, use the substitution value to prompt the user for the customer number to be displayed. SQL Code which also displays your schema name 🡺

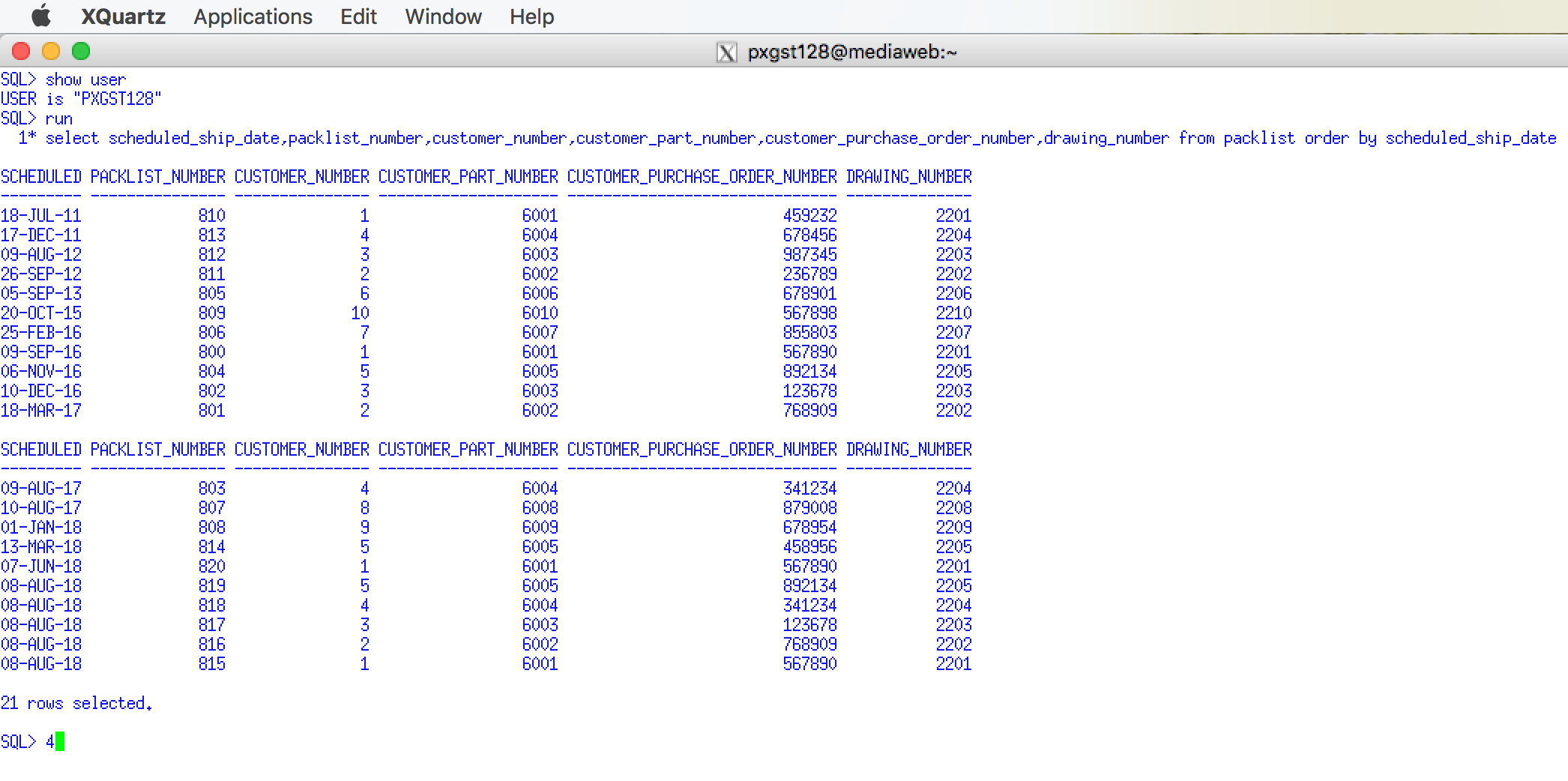


**Formatted Results of command - Leave blank if you encounter an error 🡺**

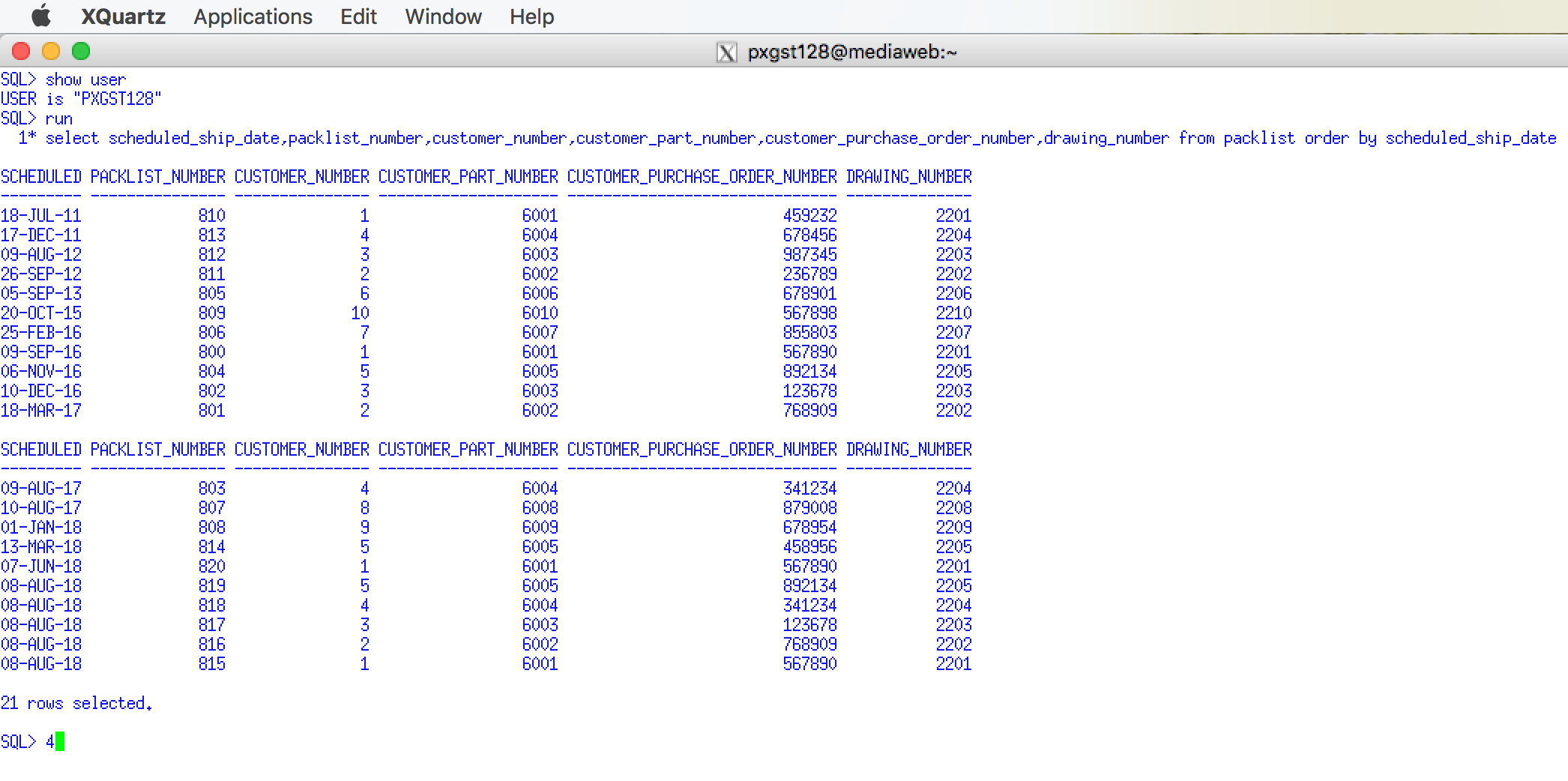


**Requirement M.** Select the Packing list table and print the Date Planned Shipped, Packing list Number, Account Number, Customer Part no, Customer PO, and Drawing Number in Date Planned Shipped Order.

SQL Code which also displays your schema name 🡺

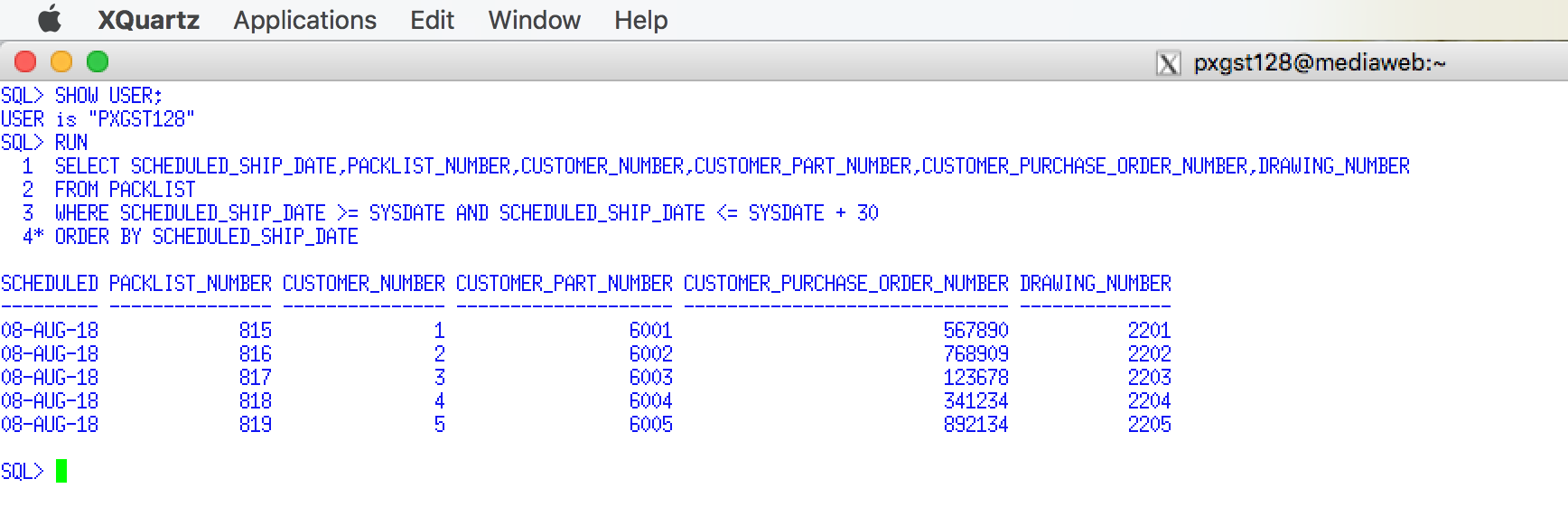


**Formatted Results of command - Leave blank if you encounter an error 🡺**

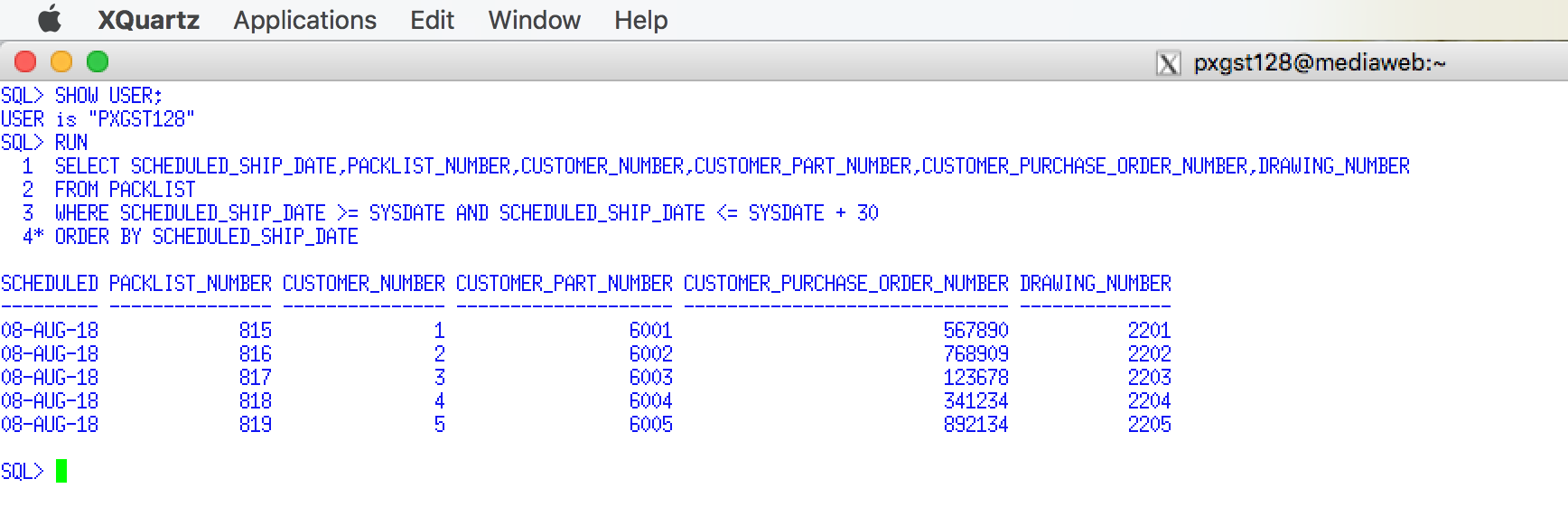


## 3.14 Questions- SYSDATE() Function

**Requirement N.** Select the Packing list table and print the Date Planned Shipped, Packing list Number, Account Number, Customer Part no, Customer PO, and Drawing Number in Date Planned Shipped Order ONLY for those packing list to be shipped during the next 30 days (sysdate). Be sure that your packing list table includes records both greater than and less than 30 days. SQL Code which also displays your schema name 🡺

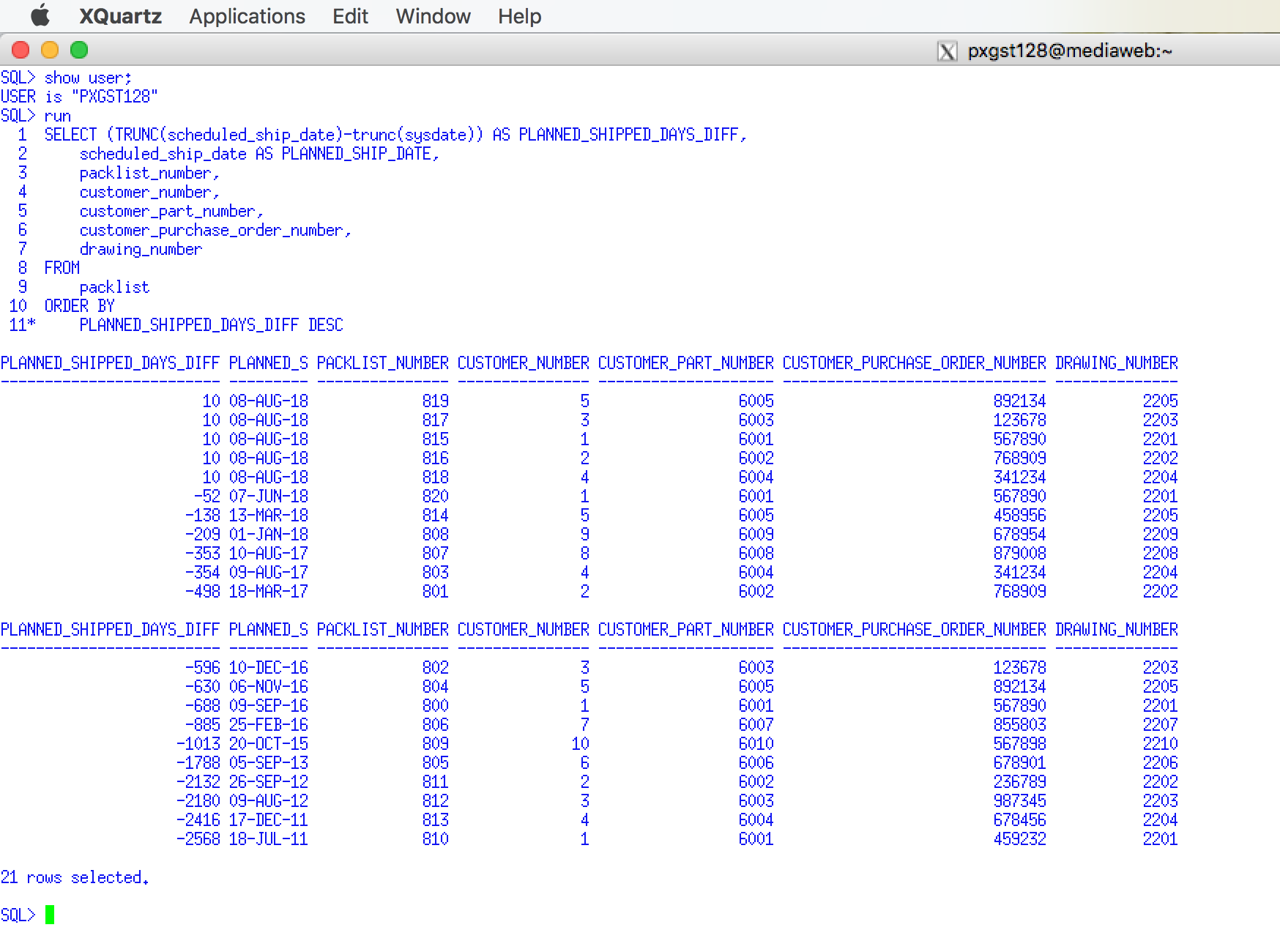


**Formatted Results of command - Leave blank if you encounter an error 🡺**

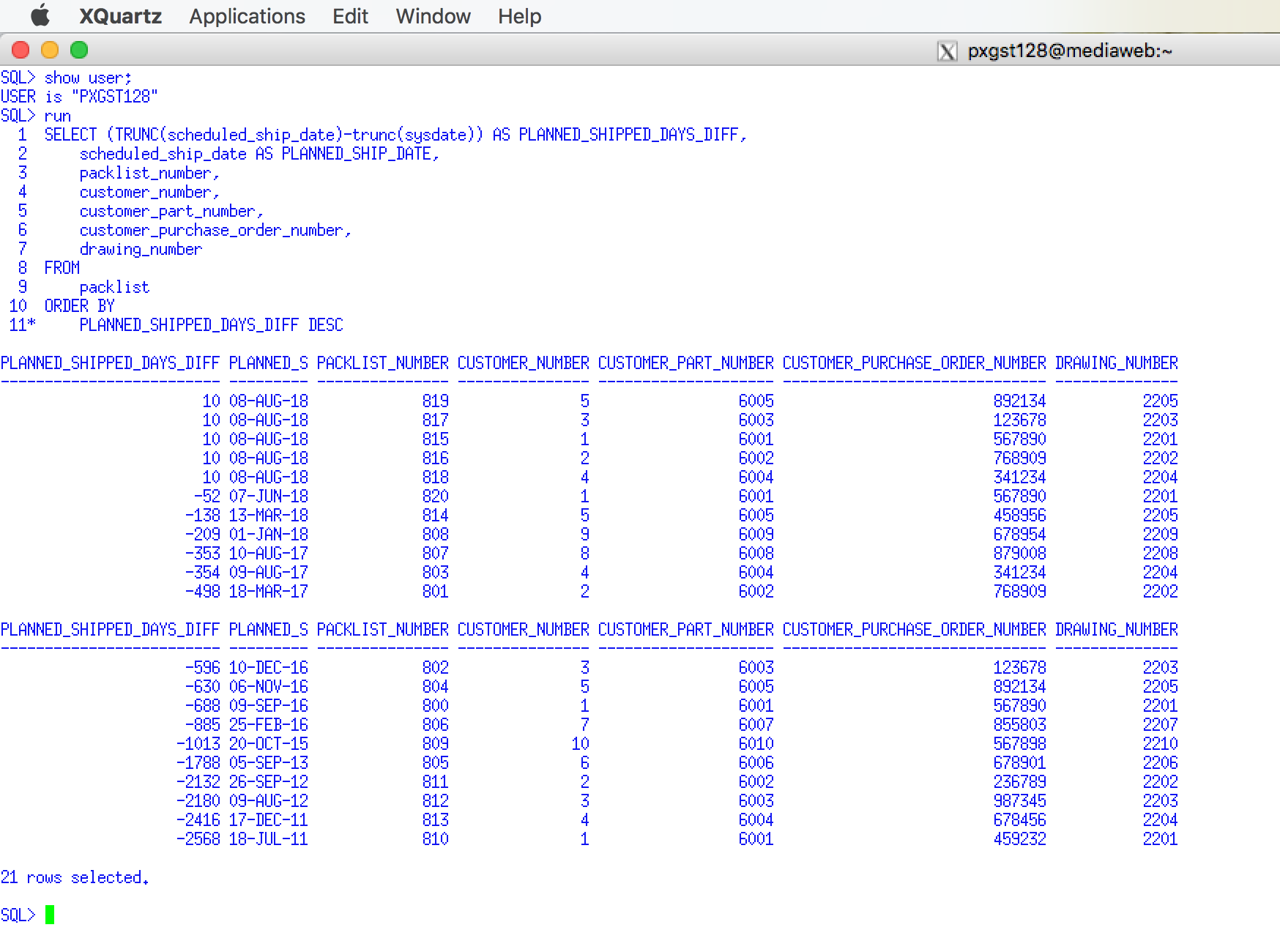


**Use a graphical snipping tool to document an image of these requirements below. Make sure your Putty banner is displayed.**

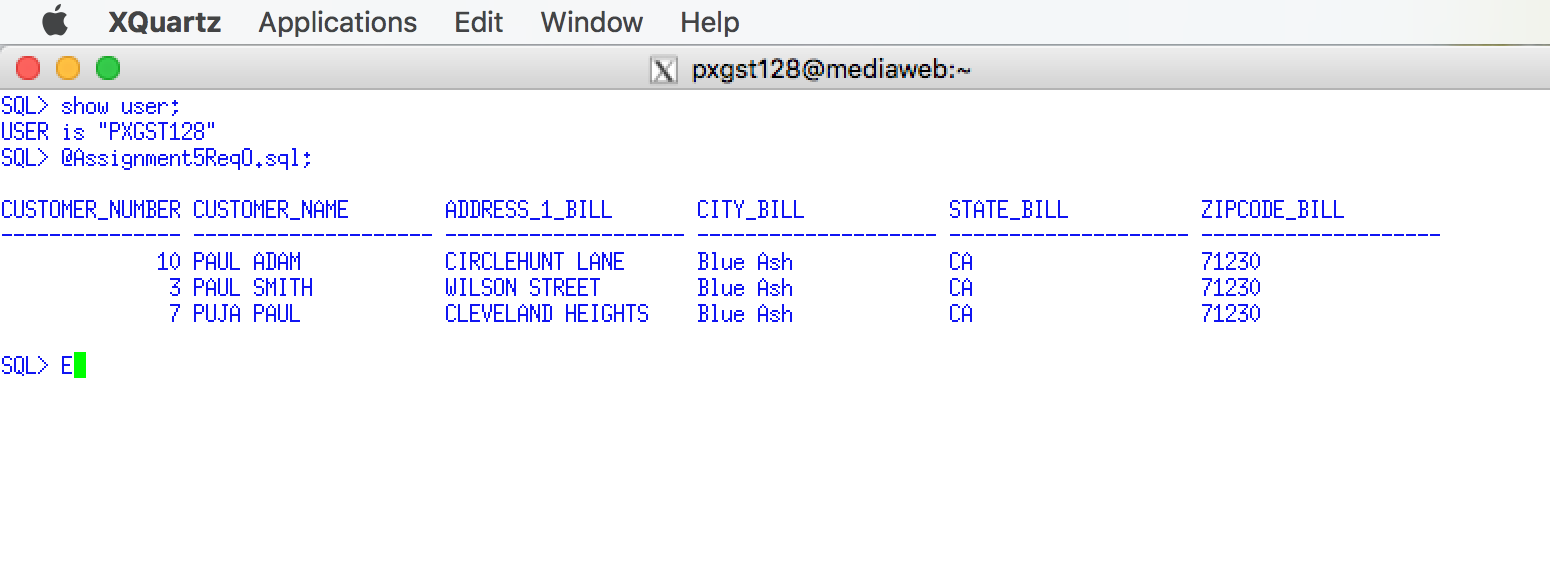
**Requirement N2.** Select the Packing list table and print the difference between today’s date and the Date Planned Shipped, Packing list Number, Account Number, Customer Part no, Customer PO, and Drawing Number ordered by the difference for all packing lists. SQL Code which also displays your schema name 🡺



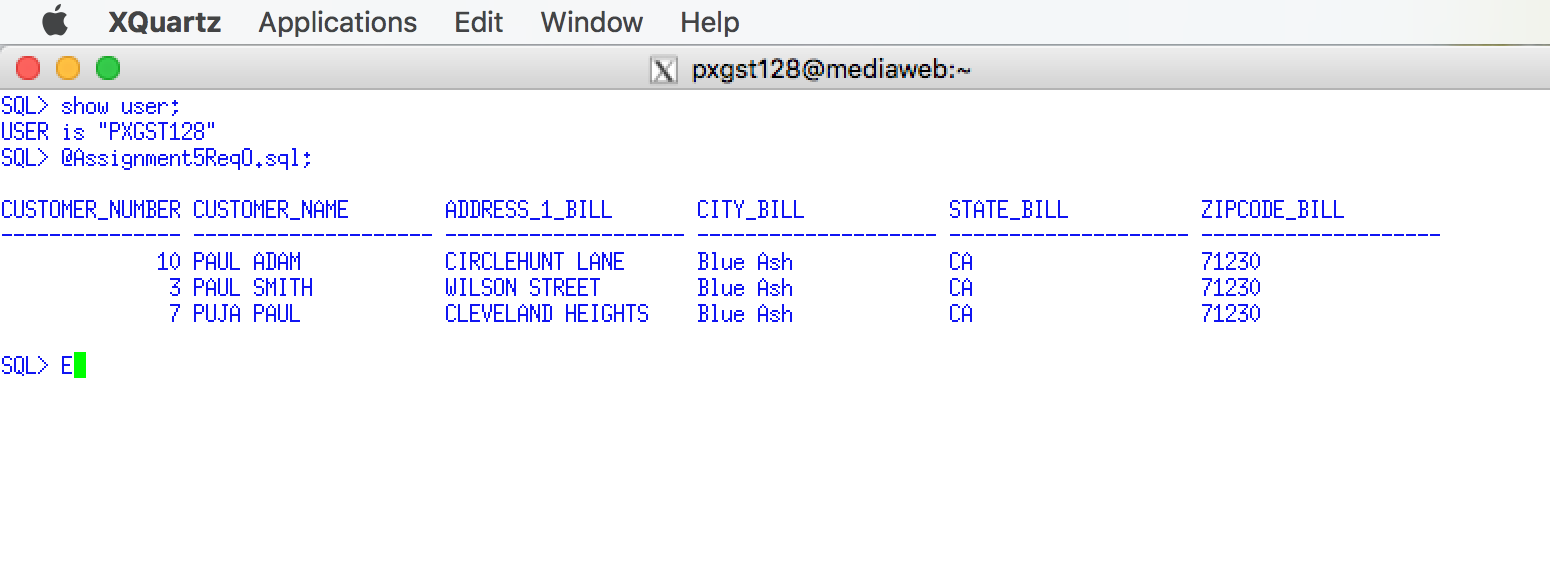
**Formatted Results of command - Leave blank if you encounter an error 🡺**



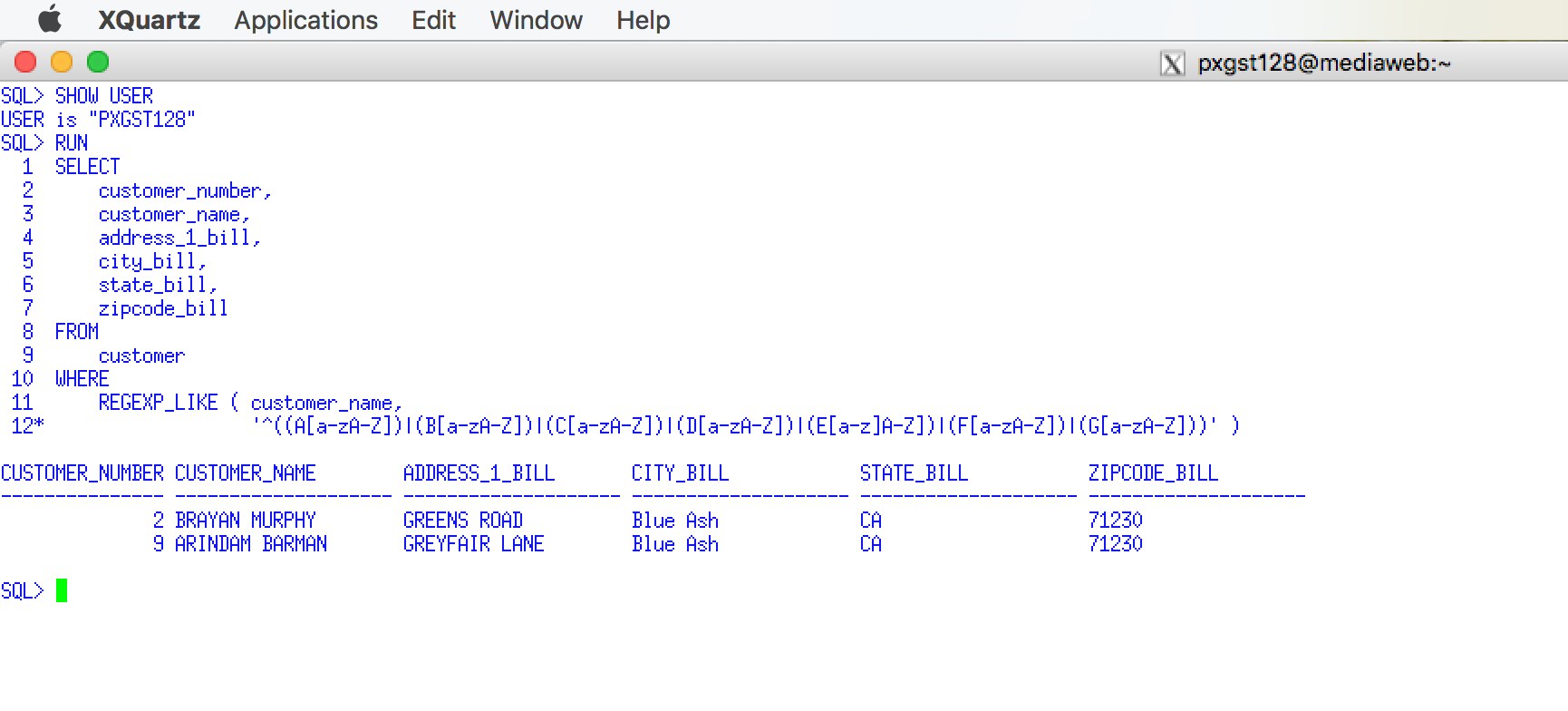
**Requirement O.** Select the Customer Table, Print the Account Number, Customer's Name, Address, City State, and Zipcode in Account Number order for all customers whose name begins with the letter A (or begins with any other single letter of the alphabet that you choose). SQL Code which also displays your schema name 🡺



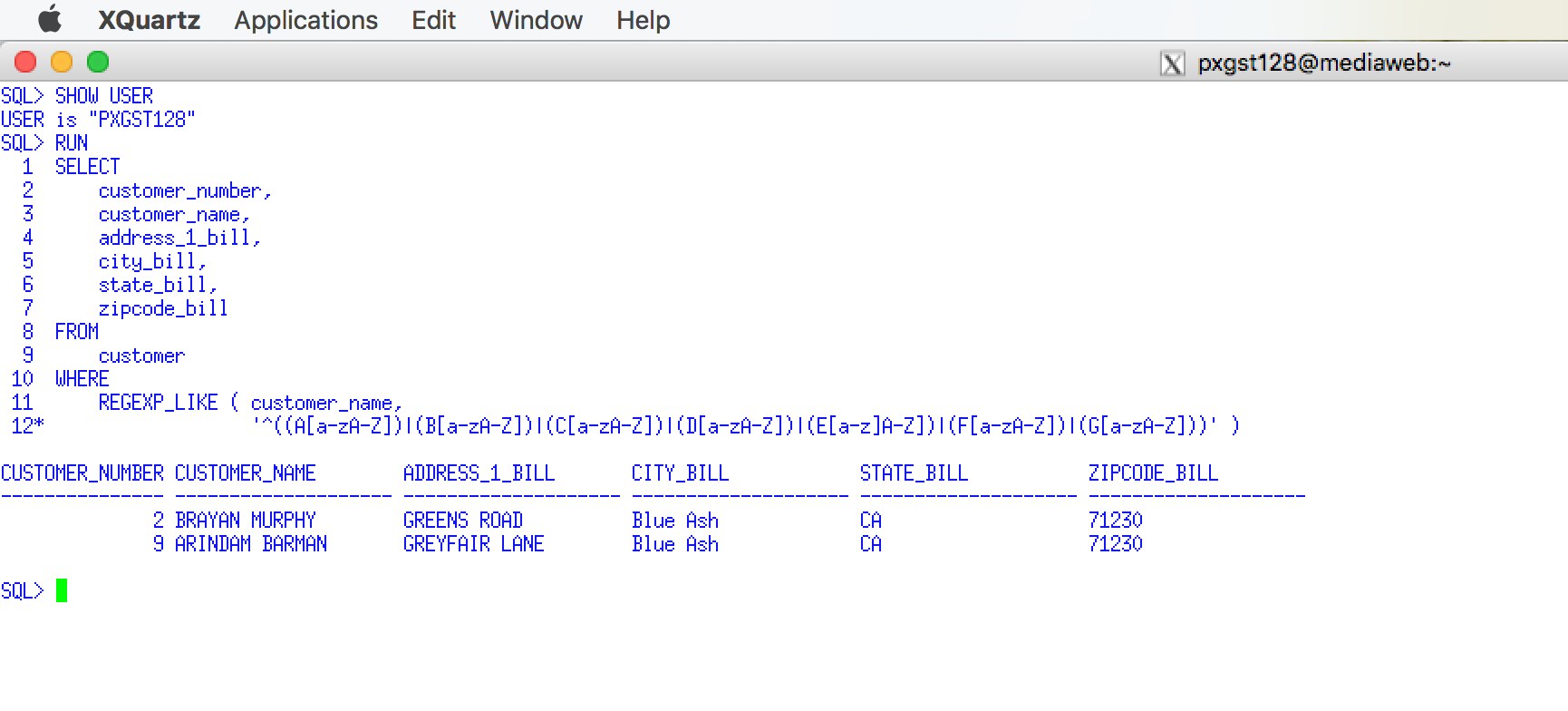
**Formatted Results of command - Leave blank if you encounter an error 🡺**



**Requirement P**. Select the Customer Table, Print the Account Number, Customer's Name, Address, City State, and Zipcode in Account Number order for all customers whose name begins with the letter A through G (or begins with any other range of letters of the alphabet that you choose). SQL Code which also displays your schema name 🡺



**Formatted Results of command - Leave blank if you encounter an error 🡺**



**Requirement Q.** Select the Customer Table, Print the Account Number, Customer's Name, Address, City State, and Zipcode in Account Number order for all customers whose name includes the letter R (or includes any letter or combinations of letters of the alphabet that you choose). SQL Code which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

## 3.17 Mathematical Expressions and DECODE

**Using Column Aliases to make your Outputs look nicer**.

Suppose that we code the following Select Statement that displays the current faculty pay and the proposed increase of 10% of the faculty pay.

SELECT FDEP, FNAME, FPAY, FPAY \* 1.10

FROM FACULTY ORDERED BY FDEP, FNAME;

The output would appear similar to the following:

FDEP FNAME FPAY FPAY \* 1.10

Cis Packy 30,000 33,000

Cis Tom 50,000 55,000

Mkt Mary 40,000 44,000

The column headers while comfortable to the designer, may not be understandable to o the user. Consider the following use of the “AS” clause to assign a more user-friendly column alias to the report.

SELECT FDEP AS DEPARTMENT, FNAME AS FACULTY\_NAME,

FPAY AS CURRENT\_SALARY, FPAY \* 1.10 AS PROPOSED\_SALARY

FROM FACULTY ORDERED BY FDEP, FNAME;

DEPARTMENT FACULTY\_NAME CURRENT\_SALARY PROPOSED\_SALARY

Cis Packy 30,000 33,000

Cis Tom 50,000 55,000

Mkt Mary 40,000 44,000

**Using the DECODE function to translate a numeric value into a string.**

DECODE Function - http://www.1keydata.com/sql/sql-decode.html

Sometimes the designer stores a numeric code as a substitute for a string. In the previous examples, we stored a string abbreviation for the department name. If on the other hand, we had stored the number 1 to represent the Computer Information Science department, the number 2 to represent the Marketing Department the preceding SQL statement, example 62 would appear as:

DEPARTMENT FACULTY\_NAME CURREENT\_SALARY PROPOSED\_SALARY

1 Packy 30,000 33,000

1 Tom 50,000 55,000

2 Mary 40,000 44,000

However, the use of the numeric codes would now permit the application of Oracle’ DECODE function.

SELECT FDEP AS DEPARTMENT, FNAME AS FACULTY\_NAME,

DECODE (FDEP, 1, ‘Computer Information Science’

2, ‘Marketing’)

FPAY AS CURRENT\_SALARY, FPAY \* 1.10 AS PROPOSED\_SALARY

FROM FACULTY ORDERED BY FDEP, FNAME;

DEPARTMENT FACULTY\_NAME CUREENT\_SALARY PROPOSED\_SALARY

Computer Information Science Packy 30,000 33,000

Computer Information Science Tom 50,000 55,000

Marketing Mary 40,000 44,000

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

**Requirement R.** Assuming that it is expected that prices may be raised by 10%, prepare a query that selects standard price list and prints the Price list Number, Price Type, and the first 4 prices raised (not changed by 10%). The Columns names should be changed to Est.Price 1, Est. Price 2 and so forth using the AS clause. This is not a SQL UPDATE requirement.

SQL Code which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

**Requirement S.** Select the Packing List Table and print the Total Quantity shipped for all packing List records. Only the totals should be printed, not the detail records. SQL Code which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

**Requirement T.** Select the Job Cost table and print the transaction date, packing list number, transaction type, transaction info, transaction units, unit cost and extended cost (transaction units multiplied by unit cost) in transaction date order. SQL Code which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

## 3.18 Questions - Aggregate Functions and Group By

<http://www.youtube.com/watch?v=E3rXVrTTROU>

SQL GROUP BY, HAVING and aggregate functions, video tutorial

<http://www.youtube.com/watch?v=098hXV0q9ps>

In SQL, what’s the difference between the having clause and the where clause? <http://www.programmerinterview.com/index.php/database-sql/having-vs-where-clause/>

<http://www.1keydata.com/sql/sqlgroupby.html>

**Subtotals, Control breaks, or GROUP BY**

SELECT FDEP, COUNT(\*) FROM FACULTY

GROUP BY FDEP;

Report the number of faculty members in each department. The field that will be used to group or categorize the data will be specified in the SELECT list and the GROUP BY list. The SELECT list must contain only column names specified in the GROUP BY list. For example, the following SQL example will cause a syntax error because FPAY is not in the GROUP BY list:

SELECT FDEP, FPAY, COUNT(\*) FROM FACULTY

GROUP BY FDEP

SELECT FDEP, AVG(FPAY) FROM FACULTY

GROUP BY FDEP;

Report the average faculty pay by department listing departments in insertion order (no order).

SELECT FDEP, AVG(FPAY) FROM FACULTY

GROUP BY FDEP

ORDER BY FDEP;

Report the average faculty pay by department listing the department names in alphabetical order.

SELECT FDEP, AVG(FPAY) FROM FACULTY

WHERE FPAY > 30000

GROUP BY FDEP;

Before averaging the faculty pay for each department, SELECT only those records that have FPAY > 30000. Faculty members earning less than 30000 are not included in the calculation of the departmental average.

SELECT FDEP, AVG(FPAY) FROM FACULTY

GROUP BY FDEP

HAVING AVG(FPAY) > 40000;

After averaging all the faculty member's pay for each department, report only those departments that have a departmental average pay greater than 40000. The WHERE clause will limit the selection of records to be averaged. The WHERE clause should only contain field names, logical operators, etc., but no aggregate functions. The HAVING clause can only specify SQL function names, logical operators and constants. Simple field names are not permitted in a HAVING clause.

SELECT FDEP, AVG(FPAY) FROM FACULTY

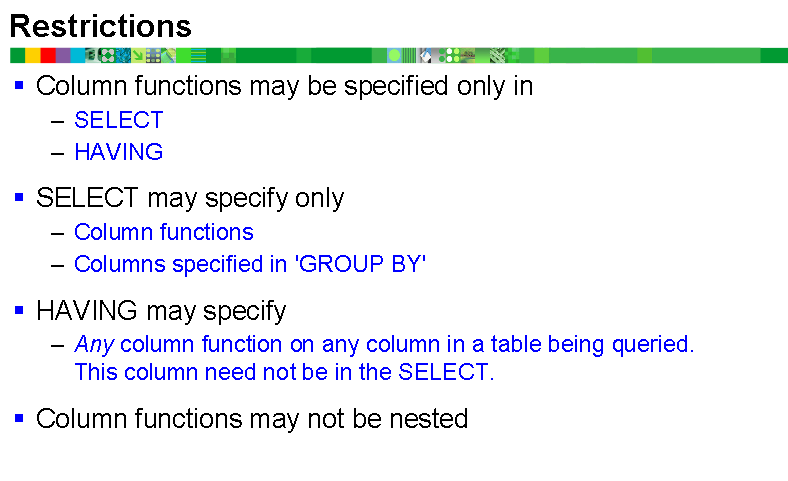
WHERE FPAY > 30000

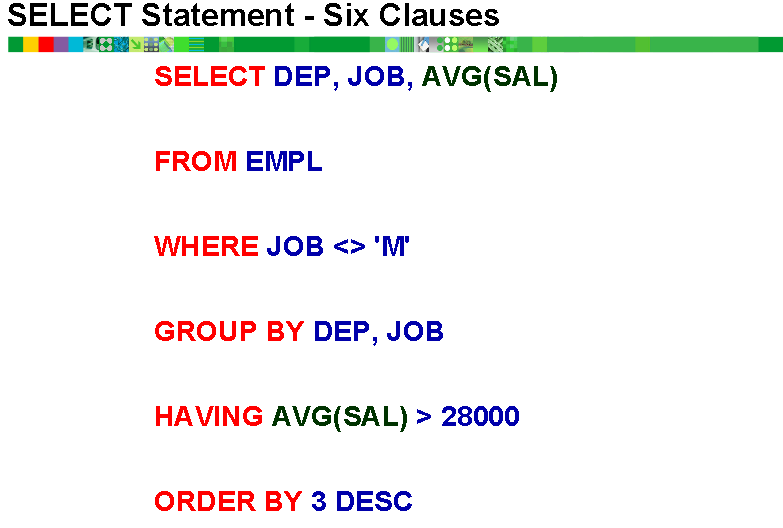
GROUP BY FDEP

HAVING AVG(FPAY) > 40000

ORDER BY FDEP;

Before averaging the faculty pay for each department, SELECT only those records that have FPAY > 30000. After averaging all faculty member's pay for each department, report only those departments that have a departmental average pay greater than 40000. Then list each department's name in alphabetical order.





**Requirement U.** Select the Job Cost Table and print the total cost (transaction units multiplied by unit cost) for all job cost records. Only the totals should be printed, not the detail records. SQL Code which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

**Requirement V.** Select the Job Cost Table and print the total cost (transaction units multiplied by unit cost) for all job cost records that occurred during a month that you select. Only the totals should be printed, not the detail records. SQL Code 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

**Requirement W.** Select the Job Cost Table and print the total cost (transaction units multiplied by unit cost) for all job cost records grouped by transaction type. Only the totals should be printed, not the detail records. SQL Code 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

**Requirement X**. Select the Job Cost Table and print the total cost (transaction units multiplied by unit cost) for all job cost records grouped for each packing list number ordered by packing list number. Only the totals should be printed, not the detail records. SQL Code 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

## 3.19 Questions- GROUP and HAVING

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

### 3.19.1 Apply Aggregate (column) functions

Code a select statement that will display the number of customers classified by customer type listed in customer type order. hint: a count function.

Use a Snippit to document your SQL output =>

SQL Code which also displays your schema name =>

### 3.19.2 Apply GROUP BY Clauses

Display the Account Number, count of the rows that have either two or two drawings for the customer. List the output in count number order. Remember the WHERE clause selects row and the HAVING clause selects aggregate totals.

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

Use a Snippit to document your SQL output =>

SQL Code which also displays your schema name =>

### 3.19.3 Apply WHERE and GROUP BY Clauses

Select the Job Cost Table and display the total cost (transaction units multiplied by unit cost) for only labor over time **grouped** for each packing list number **ordered by** packing list number. Only the totals should be printed, not the detail records. You need multiple job costs rows for multiple PLs. This is a summary analysis. DO not show the details of each row. This is similar to displaying the QPA for each student from the transcript table, rather that the grade for each course for each student.

**Make sure your Putty banner is displayed.**

Use a Snippit to document your SQL output =>

SQL Code which also displays your schema name =>

## 3.20 Questions - Apply the LIKE Operator

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

Select those drawings and print Drawing Number, Prefix, Account Number, Customer Part No for those Customer Part Numbers that contain the word “roll”. (Hint: Use the Like Operator)

Use a Snippit to document your SQL output =>

SQL Code which also displays your schema name =>

# 4.0 Dynamic Queries (Interactive Queries) and Oracle Reports

**Important**

The results should be easy to read and grade. The following is some guidelines that may help you minimize some formatting.

SQL String Functions - http://webspace.cs.odu.edu/~ibl/450/common/sqlstring.html

To\_Date Function - <http://www.techonthenet.com/oracle/functions/to_date.php>

Oracle TO\_DATE Function – SQL Syntax Examples - <http://www.mandsconsulting.com/oracle-to_date-function-sql-syntax-examples>

Oracle TO\_CHAR Function – SQL Syntax Examples (Most With Dates, TO\_DATE) - <http://www.mandsconsulting.com/oracle-to_char-function-sql-syntax-examples-most-with-dates-to_date>

Oracle Built in Functions - <http://beginner-sql-tutorial.com/oracle-functions.htm>

a. **I cannot grade wrapped text.** By default SQLPlus has a default line size of 80 characters. If your results are greater than 80 characters wide it will wrap to the next line. While you may correct the problem in Word, a better solution is change SQLplus's linesize, e.g., set linesize 150

b. A problem with the set linesize is that the text will be display past the right margin of the Putty screen. One solution would be to change Putty's font size. You may also change Word's font size or even switch to landscape.

c. The substr function, e.g., see <http://www.techonthenet.com/oracle/functions/substr.php>, can help by displaying a smaller column width than is stored.

For example Select substr(Cname ,1, 4) from customer; would only display the first four characters of the customer name.

d. The to-char function, e.g., see <http://www.techonthenet.com/oracle/functions/to_char.php>, can better display dates and times. For example,

Select to\_char(date\_shipped, 'Month DD, YYYY') from invoice; would display July 09, 2012

## 4.1 Introduction to Interactive Queries

Given the syntax of the following Select statement.

Example 1

**Select fname, fpay**

**From faculty**

**Where fdep = ‘CIS’;**

This Select statement will display all of the faculty members of the CIS department. But, what if a user wanted to display all of the faculty members of the MKT department? No problem. Change the value in the WHERE Clause. What about the ACT department? What if the user doesn’t know how to write or edit a Select statement?

Oracle provides a several methods to provide search criteria during execution. In other words, Oracle will prompt (ask) you for the department you want to Select.

**Substitution Values**

One method to enable this interactivity Oracle is to use Substitution Values (variables). Substitution Values are varaibles that begin with an ampersand (&). Consider the new improved version of the previous Select statement.

**Example 2**

**Select fname, fpay**

**From faculty**

**Where fdep = &faculty\_department;**

**When this statement is executed Oracle will provide the following prompt:**

**Enter Value for faculty\_department:**

Whatever you enter as a substitution value Oracle will search for. In response to the prompt you type ‘CIS’ (don’t forget the single quotes for varchar fields), then the CIS faculty members will be displayed.

In response to the prompt you type ‘MKT’, then the CIS faculty members will be displayed.

Notice that the prompt does not look pretty, especially when it displays faculty\_department with the underscore character. The underscore character is required since this is Oracle fields are not permitted to have embedded spaces.

**Accept Command**

To provide a better prompt the Accept command can be used in conjunction with substitution values. Consider the following code:

**Example 3**

**ACCEPT faculty\_department PROMPT ‘Please Enter the Faculty Department Abbreviation to be Displayed:’**

**Select fname, fpay**

**From faculty**

**Where fdep = &faculty\_department;**

You can probably guess that the PROMPT clause of the ACCEPT command will provide a more user-friendly prompt. But, there are several things to highlight from Example 3.

First, there are two commands (and perhaps more in real life) in this example and it will probably would be better to code this commands in a script rather than at the SQL-PLUS prompt.

Second, there is no ampersand preceding the faculty\_department variable in the ACCEPT command, but there is one specified in the Where clause. The Select statement expects field names from our tables. The ACCEPT command does not. Therefore, we need a method to distinguish table field names from other variables in the Select statements, but not the ACCEPT command. Therefore, the ampersand is only required in the Select statement.

Third, notice there is only one semicolon at the end of the Select statement, but not after ACCEPT statement. Remember, the semicolon means to execute. We code the semicolon at the end of the script, which tells Oracle –“ It is not time to execute the previous commands”.

The execution will appear similar to the following:

Please Enter the Faculty Department Abbreviation to be displayed: CIS

Select fname, fpay

From faculty

Where fdep = &faculty\_department

Where fdep = ‘CIS’

\*\*\*\* then the result will be displayed \*\*\*\*

Notice that the select statement will be displayed, and the Where clause will be displayed twice. The first WHERE clause displayed the original code, the second displays the verification of the substitution value entered. To suppress the display of verification lines, enter SET VERIFY OFF either at the SQL prompt or inside the script.

## 4.2 Questions – Dynamic Inquires

**Your documentation is required to display your Putty Header and you should use a snipping tool**

Requirement Y. Create a Dynamic Inquiry using the Packlist table that uses substitution values and ACCEPT statement. SQL Code 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

## 4.3 Oracle Reports in SQL\*Plus

Video - Formatting SQL\*Plus Output - <https://www.youtube.com/watch?v=_46MeUAFDO4>

Video - Oracle Scripting Tips and Techniques - <https://www.youtube.com/watch?v=E1ka-cnVjgs>

The normal output of a Select command is poorly formatted. You can create a script file in which the output of the Select command will be better formatted. The TITLE command will specify the title of the report to appear at the top of each page, which also can be aligned as LEFT, RIGHT and CENTER. The SKIP command will specify any extra lines to be skipped. For example, the command SKIP 1 will actually double-space the next printed line. After you print a report YOU MUST use the TITLE OFF or BTITLE OFF command since these titles will also appear on any subsequent queries.

BTITLE will specify a title that may appear at the bottom of each page. The SET LINESIZE command will specify the width (no of printed characters) of the printed output without scrolling to the next line. SET PAGESIZE will number of lines that may appear on a printed page. Be sure to adjust for your headers. For example, if your normal printed page may contain 60 printed lines, then PAGESIZE must be less than 60 to make room for your headers.

Probably the most useful command is the COLUMN command. The COLUMN command is followed by the field name (as it is stored in the table. The HEADING command is followed by the text that will be displayed as the column header, similar to the ALIAS column name. The FORMAT command provides you a method specify the width of a column and the format of the column. The A command is used for CHAR and VARCHAR2 data; it will left justify the data. The 9 command will be used for numeric data and will right or decimal justify the data. For example, Format 999, or Format 9,999.99 or Format $99,999.99.

Customer\_report.sql

**TITLE CENTER ‘PSP, Inc.’ SKIP 1**

**CENTER ‘\*\*\*\*\*\*\*’ SKIP 2**

**LEFT ‘Customer Report’**

**BTITLE ‘This appears at the bottom of each Page’**

**SET LINESIZE 80**

**SET PAGESIZE 55**

**COLUMN customer\_no HEADING ‘C# ‘ FORMAT 999**

**COLUMN customer\_name HEADING ‘Customer Name‘ FORMAT A20**

**COLUMN customer\_address HEADING ‘Customer Address‘ FORMAT A25**

**COLUMN customer\_city HEADING ‘City‘ FORMAT A20**

**COLUMN customer\_state HEADING ‘State‘ FORMAT A5**

**COLUMN customer\_zipcode HEADING ‘Zipcode‘ FORMAT A10**

**Select customer\_no, cutomer\_name, customer\_address, customer\_state, customer\_zipcode**

**From customer; { notice only one trailing semicolon**

## 4.4 Questions - Oracle Reports in SQL\*Plus

**Use a graphical snipping tool to document an image of these requirements below.**

Requirement Z. Improve Requirement Y to include SQL report formatting commands, e.g., Title, Center, Left, BTitle, Line size and page size. Use appropriate column formatting and headers. SQL Code 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

# 5.0 - Your Customized Examples of SQL SELECT Statements

After one stores data values into a table column, the data may be retrieved to produce a business document, report, analyze data or provide management information to answer a question. Data analytics (DA) is the science of examining raw data with the purpose of drawing conclusions about that information. Data analytics is used in many industries to allow companies and organization to make better business decisions and in the sciences to verify or disprove existing models or theories. The underlying foundation of data analytic tools is the SQL SELECT statement. A data analytic tool may have fancy GUI drag-and-drop features or may provide fancy reports, analysis or graphs, but underneath these fancy features is a SQL SELECT statement.

RMU has a successful Actuarial Science Degrees. Actuarial science is the discipline that applies mathematical and statistical methods to assess risk in insurance, finance and other industries and professions. Many of RMU actuarial science, accounting and marketing alumni, will tell you about the importance of SQL in their careers.

In this requirement, you will be required to design and code your own SQL statement that may be used to provide information to the PSP managers while demonstrating your knowledge of SQL. It is recommended to review the PSP case and tables design (provided at the end of this document) for ideas. Your grade will be based on applying the required SQL statements and clauses. Better students will take some extra initiative and see if they could design a better SELECT statement than is minimally required that would provide value valuable information to the PSP administration.

**In addition to the SQL statement and Output you will be required to provide an INFORMATON OR REPORT TITLE:** **The Title Header should provide a User-friendly description that SPECIFIC description of the information provided by your SQL statement example.**

**It would be excellent idea to read the PSP case at this time. Why do we store data in a database? 1) To record transactions and information about the entities that is the foundation for PSP to survive. If PSP does know the names of their customers, drawing specification Pack List orders and job manufacturing costs they would go out of business. 2) To make decisions about the current status of manufacturing operations and discover business strategies to improve future operations.**

**For example, PSP has customers, drawings, and packing lists (which are orders). If a customer asked PSP to give them a quote of a drawing, but they never placed an order, e.g., a packing list, should you want to know WHY? If PSP was producing drawings on packing lists, but the total job cost exceeds the total selling revenue, should you want to know WHY?**

**Students often try to work together, but they shouldn't submit the same customized scripts. In real life, there were four RMU students working on the PSP project. After designing, coding and creating the database, and entering test data, it took six months for these four students to run of ideas to code hundreds of customized SQL statements. There were over 50 customized packing list and job cost join statements, which use the previous SQL techniques to provide information to explain why PSP was operating at a loss.**

**While some student's examples may be similar the title will clearly explain that one student was investigating raw materials which the other student was investigating labor time and labor operations. The SQL code may be similar, a little change in the SELECT columns and a WHERE statement**

Do not forget the WHERE clause, e.g., BLANKET CUSTOMER REPORT, or JOBOSTS BETWEEN XXXXX and YYYY DATEs, or the use of the GROUP BY clause,, e.g., JOB COSTS by CUSTOMER PART NUMBER., or NUMBER OF DRAWINGS BY CUSTOMER. The ORDER BY clause might provide information such as DRAWINGS listed by CUSTOMER or CUSTOMER PART NUMBERS.

DO NOT USE PREVIOUS SQL EXAMPLES USED IN THIS ASSIGNMENT. You may use any tables that you wish and long as you provide useful information.

Column Aliases and user-friendly formatting should be used or the requirement will be penalized by 50% of the assignment grade.

**Your documentation is required to display the Putty Header and you should use a snipping tool**

**Important**

The results should be easy to read and grade. The following is some guidelines that may help you minimize some formatting.

SQL String Functions - http://webspace.cs.odu.edu/~ibl/450/common/sqlstring.html

To\_Date Function - <http://www.techonthenet.com/oracle/functions/to_date.php>

Oracle TO\_DATE Function – SQL Syntax Examples - <http://www.mandsconsulting.com/oracle-to_date-function-sql-syntax-examples>

Oracle TO\_CHAR Function – SQL Syntax Examples (Most With Dates, TO\_DATE) - <http://www.mandsconsulting.com/oracle-to_char-function-sql-syntax-examples-most-with-dates-to_date>

Oracle Built in Functions - <http://beginner-sql-tutorial.com/oracle-functions.htm>

a. **I cannot grade wrapped text.** By default SQLPlus has a default line size of 80 characters. If your results are greater than 80 characters wide it will wrap to the next line. While you may correct the problem in Word, a better solution is change SQLplus's linesize, e.g., set linesize 150

b. A problem with the set linesize is that the text will be display past the right margin of the Putty screen. One solution would be to change Putty's font size. You may also change Word's font size or even switch to landscape.

c. The substr function, e.g., see <http://www.techonthenet.com/oracle/functions/substr.php>, can help by displaying a smaller column width than is stored.

For example Select substr(Cname ,1, 4) from customer; would only display the first four characters of the customer name.

d. The to-char function, e.g., see <http://www.techonthenet.com/oracle/functions/to_char.php>, can better display dates and times. For example,

Select to\_char(date\_shipped, 'Month DD, YYYY') from invoice; would display July 09, 2012

## 5.1 ORDER BY, DESC, ROWNUM, TOP or BOTTOM Examples

**The Title Header should provide a User-friendly description that SPECIFIC description of the information provided by your SQL statement example.**

Apply, design, code, execute and document five useful examples demonstrating your knowledge and apply ORDER BY, DESC, ROWNUM, TOP or BOTTOM causes.

**Your documentation is required to display your Putty Header and you should use a snipping tool**

### 5.1.1 Your Example 1

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.1.2 Your Example 2

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

**Your documentation is required to display your Putty Header and you should use a snipping tool**

### 5.1.3 Your Example 3

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.1.4 Your Example 4

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

**Your documentation is required to display your Putty Header and you should use a snipping tool**

### 5.1.5 Your Example 5

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

## 5.2 WHERE, AND, OR, TO\_DATE(), TO\_CHAR(), or IN examples

**The Title Header should provide a User-friendly description that SPECIFIC description of the information provided by your SQL statement example.**

**It would be excellent idea to read the PSP case at this time. Why do we store data in a database? 1) To record transactions and information about the entities that is the foundation for PSP to survive. If PSP does know the names of their customers, drawing specification Pack List orders and job manufacturing costs they would go out of business. 2) To make decisions about the current status of manufacturing operations and discover business strategies to improve future operations.**

**For example, PSP has customers, drawings, and packing lists (which are orders). If a customer asked PSP to give them a quote of a drawing, but they never placed an order, e.g., a packing list, should you want to know WHY? If PSP was producing drawings on packing lists, but the total job cost exceeds the total selling revenue, should you want to know WHY?**

**Students often try to work together, but they shouldn't submit the same customized scripts. In real life, there were four RMU students working on the PSP project. After designing, coding and creating the database, and entering test data, it took six months for these four students to run of ideas to code hundreds of customized SQL statements. There were over 50 customized packing list and job cost join statements, which use the previous SQL techniques to provide information to explain why PSP was operating at a loss.**

**While some student's examples may be similar the title will clearly explain that one student was investigating raw materials which the other student was investigating labor time and labor operations. The SQL code may be similar, a little change in the SELECT columns and a WHERE statement.**

**Your documentation is required to display your Putty Header and you should use a snipping tool**

### 5.2.1 Your Example 1

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.2.2 Your Example 2

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

**Your documentation is required to display your Putty Header and you should use a snipping tool**

### 5.2.3 Your Example 3

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.2.4 Your Example 4

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.2.5 Your Example 5

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

## 5.3 SUM(), COUNT(), ANG(), MIN(), MAX(), GROUP BY, or HAVING Examples

**In addition to the SQL statement and Output you will be required to provide an INFORMATON OR REPORT TITLE:** **The Title Header should provide a User-friendly description that SPECIFIC description of the information provided by your SQL statement example.**

**It would be excellent idea to read the PSP case at this time. Why do we store data in a database? 1) To record transactions and information about the entities that is the foundation for PSP to survive. If PSP does know the names of their customers, drawing specification Pack List orders and job manufacturing costs they would go out of business. 2) To make decisions about the current status of manufacturing operations and discover business strategies to improve future operations.**

**For example, PSP has customers, drawings, and packing lists (which are orders). If a customer asked PSP to give them a quote of a drawing, but they never placed an order, e.g., a packing list, should you want to know WHY? If PSP was producing drawings on packing lists, but the total job cost exceeds the total selling revenue, should you want to know WHY?**

**Students often try to work together, but they shouldn't submit the same customized scripts. In real life, there were four RMU students working on the PSP project. After designing, coding and creating the database, and entering test data, it took six months for these four students to run of ideas to code hundreds of customized SQL statements. There were over 50 customized packing list and job cost join statements, which use the previous SQL techniques to provide information to explain why PSP was operating at a loss.**

**While some student's examples may be similar the title will clearly explain that one student was investigating raw materials which the other student was investigating labor time and labor operations. The SQL code may be similar, a little change in the SELECT columns and a WHERE statement.**

**Your documentation is required to display your Putty Header and you should use a snipping tool**

### 5.3.1 Your Example 1

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.3.2 Your Example 2

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

**Your documentation is required to display your Putty Header and you should use a snipping tool**

### 5.3.3 Your Example 3

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.4.4 Your Example 4

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.3.5 Your Example 5

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

## 5.4 ACCEPT, SQL\*Plus mathematical expression formatting in the SELECT LIST

4. Apply, design, code, execute and document five useful examples demonstrating your knowledge of apply dynamic queries using the ACCEPT statement, SQL\*Plus formatting or a mathematical expression in the SELECT LIST.

**Your documentation is required to display your Putty Header and you should use a snipping tool**

### 5.4.1 Your Example 1

**Information Header or Title**  🡺

**SQL Code w**hich also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.4.2 Your Example 2

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

**Your documentation is required to display your Putty Header and you should use a snipping tool**

### 5.4.3 Your Example 3

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.4.4 Your Example 4

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

### 5.4.5 Your Example 5

**Information Header or Title**  🡺

**SQL Code** which also displays your schema name 🡺

**Formatted Results of command - Leave blank if you encounter an error 🡺**

