Assignment 6

Hands-on

Summer 2018

***J. Packy Laverty***

***laverty@rmu.edu***

Contents

[**Introduction 4**](#_Toc488148103)

[**How to complete Hand-on Tutorial Requirements and Review Questions 4**](#_Toc488148104)

[**Oracle Error Codes Summary 5**](#_Toc488148105)

[**Documentation Requirements 5**](#_Toc488148106)

[**Required Professional Appearance of Outputted Results 5**](#_Toc488148107)

[**Improving the Professional Appearance of the SQL Statement and Results using SQL\*Plus 6**](#_Toc488148108)

[**Improving the Professional Appearance of the SQL Statement and Results using PuTTY 6**](#_Toc488148109)

[**Improving the Professional Appearance of the Results using the SQL SUBSTR() and TO-CHAR() Function and Columns Aliases (AS) 7**](#_Toc488148110)

[**SUBSSTR() Function 8**](#_Toc488148111)

[**AS (Column Alias) 9**](#_Toc488148112)

[**TO\_CHAR() Function 10**](#_Toc488148113)

[**1.0 Retrieving Data from Multiple Tables 19**](#_Toc488148114)

[**1.1 Introducing a Join 20**](#_Toc488148115)

[**1.2 Correlation Names and Table Aliases 21**](#_Toc488148116)

[**1.3 Questions - Multiple Table Joins 22**](#_Toc488148117)

[**1.4 Questions - Types of Join 22**](#_Toc488148118)

[**1.5 Questions - Avoiding Cartesian Product Joins 23**](#_Toc488148119)

[**1.6 Question – Correlation names, Table Alias, and Synonym 24**](#_Toc488148120)

[**1.7 Question – Joining Two Tables 25**](#_Toc488148121)

[**1.8 Questions - Joining Three Tables 28**](#_Toc488148122)

[**1.9 Questions - Self-Join 35**](#_Toc488148123)

[**1.10 Questions - Joining a Table with Itself 30**](#_Toc488148124)

[**1.11 Questions - Your Join Examples 31**](#_Toc488148125)

[**1.12 Questions - Left and Right Joins 32**](#_Toc488148126)

[**2.0 Unions, Intersections and Difference 35**](#_Toc488148127)

[**2.1 Questions - UNION ALL 36**](#_Toc488148128)

[**2.2 Questions - DECODE 38**](#_Toc488148129)

[**2.3 Questions - UNION versus a JOIN 41**](#_Toc488148130)

[**2.4 Questions - INTERSECT 42**](#_Toc488148131)

[**2.5 Questions - EXCEPT 44**](#_Toc488148132)

[**3.0 Subqueries 45**](#_Toc488148133)

[**3.1 Single-Value Subqueries 45**](#_Toc488148134)

[**3.2 Multiple-Value Subqueries 46**](#_Toc488148135)

[**3.3 When to Use Joins and Subqueries 46**](#_Toc488148136)

[**3.4 Questions – Subqueries 47**](#_Toc488148137)

[**3.5 Questions - Subquery with IN Operator 49**](#_Toc488148138)

[**4.0 Unions and Subqueries 57**](#_Toc488148139)

[**4.1 Questions – Customized Subqueries 57**](#_Toc488148140)

[**4.1 Questions – Customized Unions 59**](#_Toc488148141)

[**5.0 Table Layout for the PSP Case 12**](#_Toc488148142)

[**6.0 Apply SQL to PSP Case – Level 1** Error! Bookmark not defined.](#_Toc488148143)

[**6.1 Apply the LIKE Operator 64**](#_Toc488148144)

[**6.2 Apply GROUP BY and COUNT(\*).** Error! Bookmark not defined.](#_Toc488148145)

[**6.3 Apply LIKE Operator and COUNT(\*).** Error! Bookmark not defined.](#_Toc488148146)

[**6.4 Apply HAVING Clause** Error! Bookmark not defined.](#_Toc488148147)

[**6.4 Apply WHERE and GROUP BY Clauses** Error! Bookmark not defined.](#_Toc488148148)

[**6.5 Apply an Inner Join between Customer and Drawing tables 65**](#_Toc488148149)

[**6.6 Apply an Inner Join between Customer and Packing List tables 65**](#_Toc488148150)

[**6.7 Apply an Inner Join between Customer and Packing List tables 66**](#_Toc488148151)

[**6.8 Apply an Inner Join between Customer and Packing List tables 66**](#_Toc488148152)

[**6.9 Apply an Inner Join between Customer, Packing List and Job Cost tables 67**](#_Toc488148153)

[**6.10 Apply an Inner Join between Customer, Packing List and Job Cost tables 67**](#_Toc488148154)

[**6.11 Apply an Inner Join between Packing List and Job Cost (decode) 68**](#_Toc488148155)

[**6.12 Apply an Inner Join between Job Ccost and Labor Operation 68**](#_Toc488148156)

[**6.13 Apply Inner Join between Job Cost, Time Clock, and Employee 69**](#_Toc488148157)

[**6.14 NESTED QUERIES (SUBQUERIES) Background 52**](#_Toc488148158)

[**6.15 Subquery Review 55**](#_Toc488148159)

**Enter your Name Here 🡺 Puja Ghosh**

# Introduction

**Assignment 6 is organized into two documents: Assignment 6 Theory and Assignment 6 Hands-on.**

**Assignment 6 Theory will be allocated 30% of the assignment points**

**Assignment 6 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**

# 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.

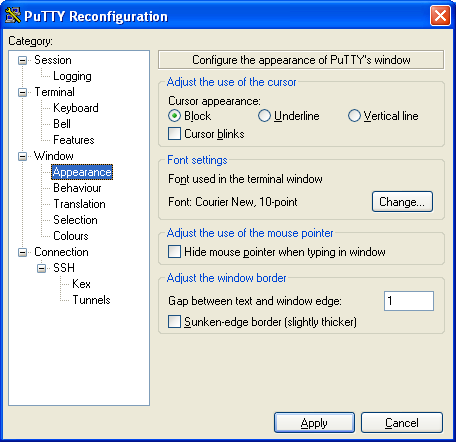
## Improving the Professional Appearance of the SQL Statement and Results using SQL\*Plus

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.

## 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)

## Improving the Professional Appearance of the Results 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,**

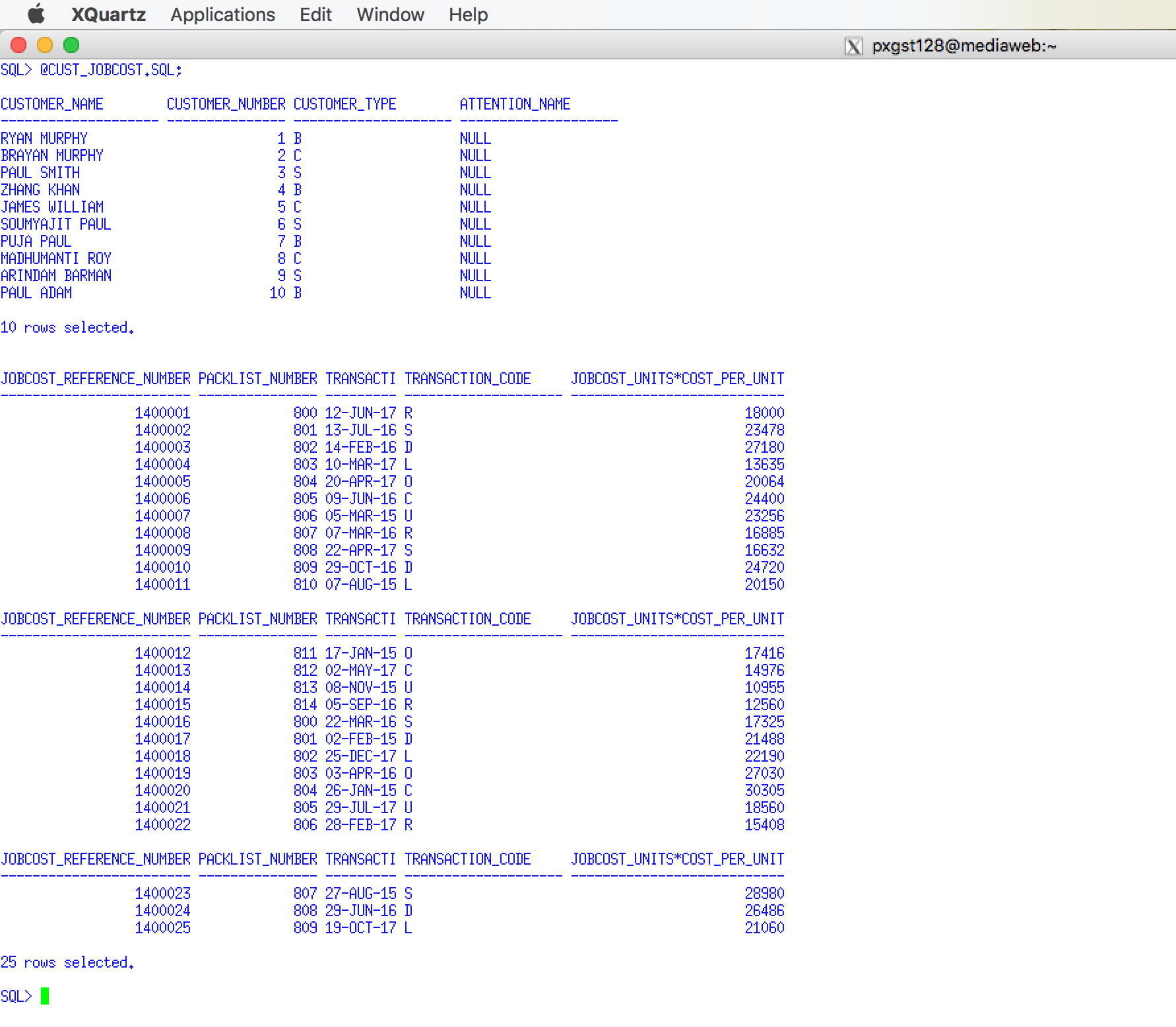
**TRANSACTION\_DATE,**

**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.

## SUBSSTR() Function

**Step 2.** Format the column width using the SUBTR() 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),**

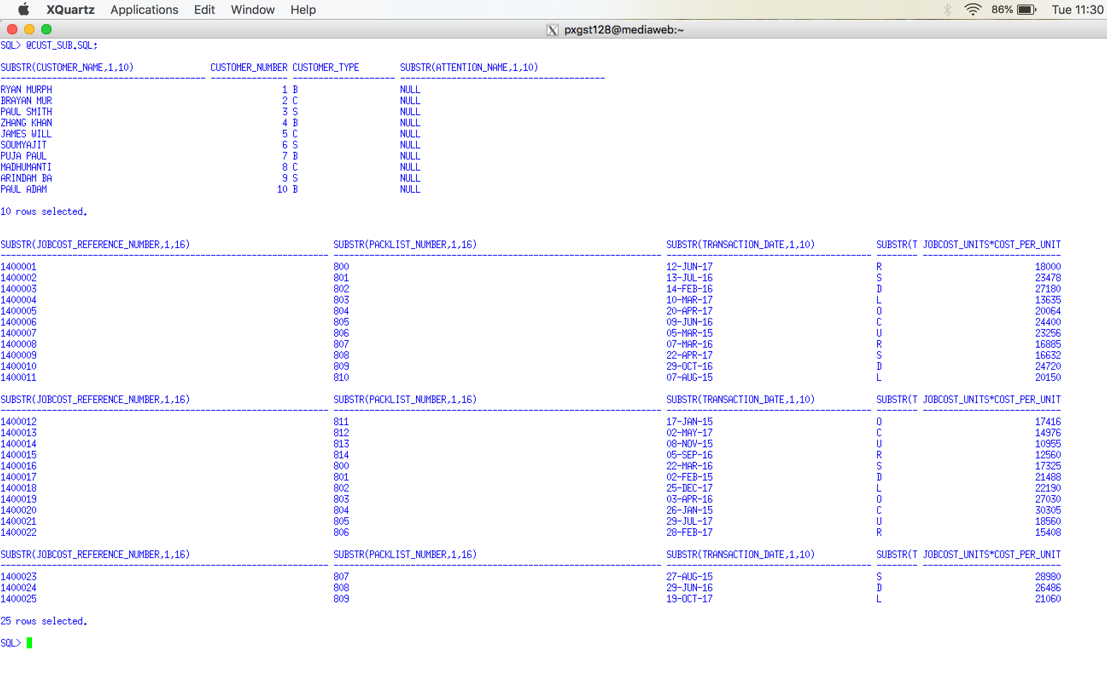
**SUBSTR(TRANSACTION\_DATE, 1, 10),**

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

JOBCOST\_UNITS \* COST\_PER\_UNIT

FROM JOB\_COST;

**Using a Snipping Tool document Your Results**



## 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#** ,

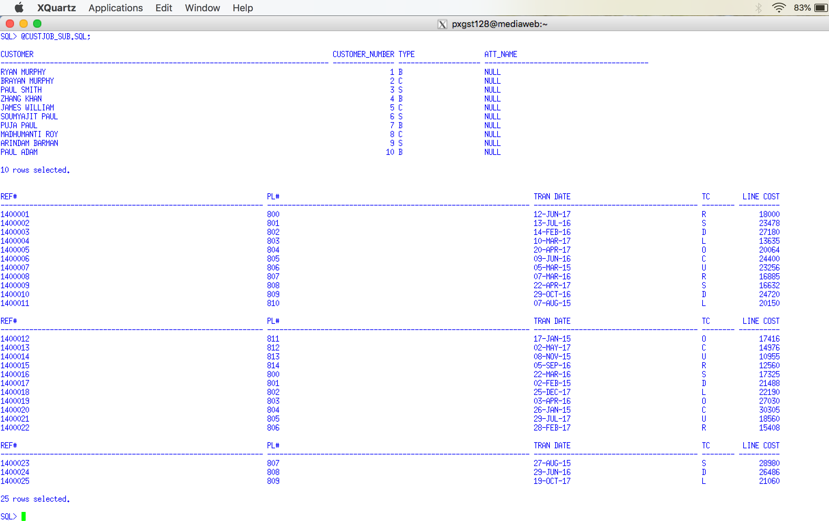
SUBSTR(TRANSACTION\_DATE, 1, 10) **AS "TRAN DATE"** ,

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. Make sure your Putty banner is displayed.**



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

**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**

# PSP Case - Table Layouts

## 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.

# 1.0 Retrieving Data from Multiple Tables

SQL Joins - <http://www.w3schools.com/sql/sql_join.asp>

SQL INNER JOIN Keyword - <http://www.w3schools.com/sql/sql_join_inner.asp>

SQL LEFT JOIN Keyword - <http://www.w3schools.com/sql/sql_join_left.asp>

SQL RIGHT JOIN Keyword - <http://www.w3schools.com/sql/sql_join_right.asp>

SQL FULL JOIN Keyword - <http://www.w3schools.com/sql/sql_join_full.asp>

Join (SQL) - <http://en.wikipedia.org/wiki/Join_%28SQL%29>

In SQL, what’s the difference between an inner and outer join? - <http://www.programmerinterview.com/index.php/database-sql/inner-vs-outer-joins/>

What is a self-join? <http://www.programmerinterview.com/index.php/database-sql/what-is-a-self-join/>

Distinct Keyword - http://www.programmerinterview.com/index.php/database-sql/retrieve-unique-rows-with-distinct/

Joining Two Tables

<http://www.youtube.com/watch?v=DwhQl2_8slA&feature=relmfu>

SQL: Understanding the JOIN clause in the SELECT statement

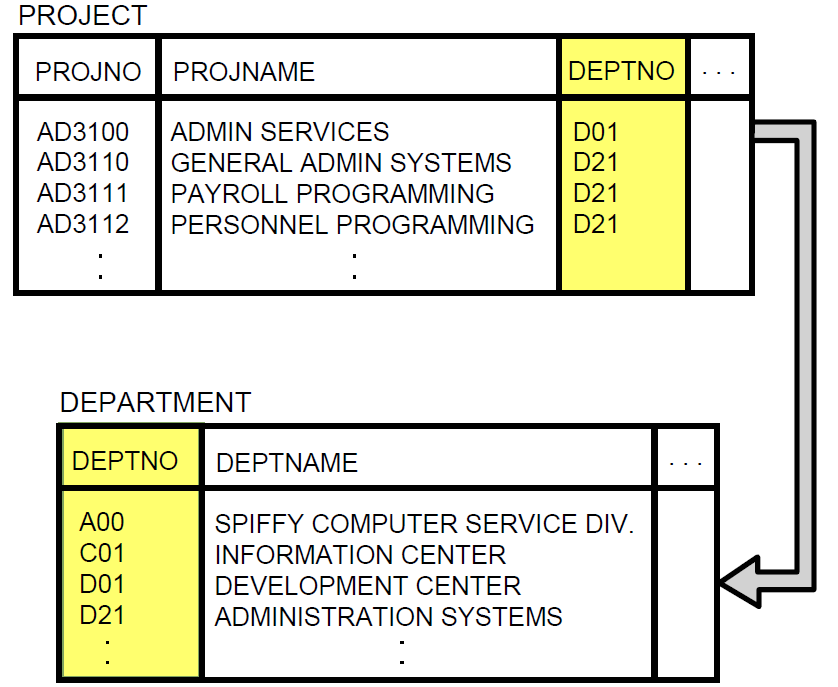
<http://www.youtube.com/watch?v=M3Dj6UWDj-4&feature=channel&list=UL>

SQL: Understanding the JOIN clause in the SELECT statement

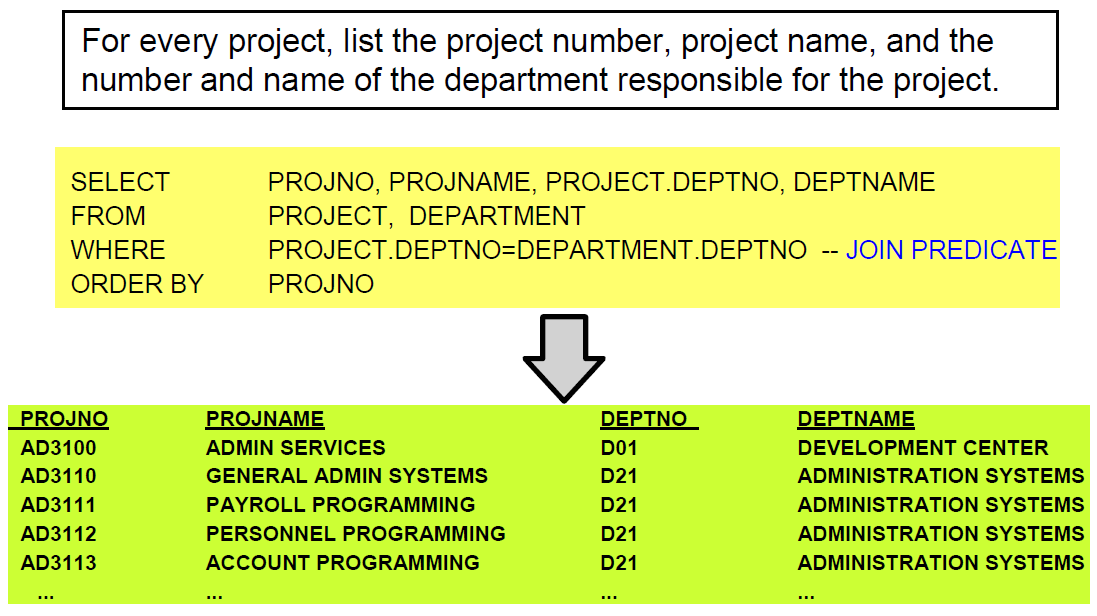
<http://www.youtube.com/watch?v=M3Dj6UWDj-4&feature=BFa&list=ULM3Dj6UWDj-4>

## 1.1 Introducing a Join

**Relationship between the Project and Department Table**



**Introducing the JOIN Statement**

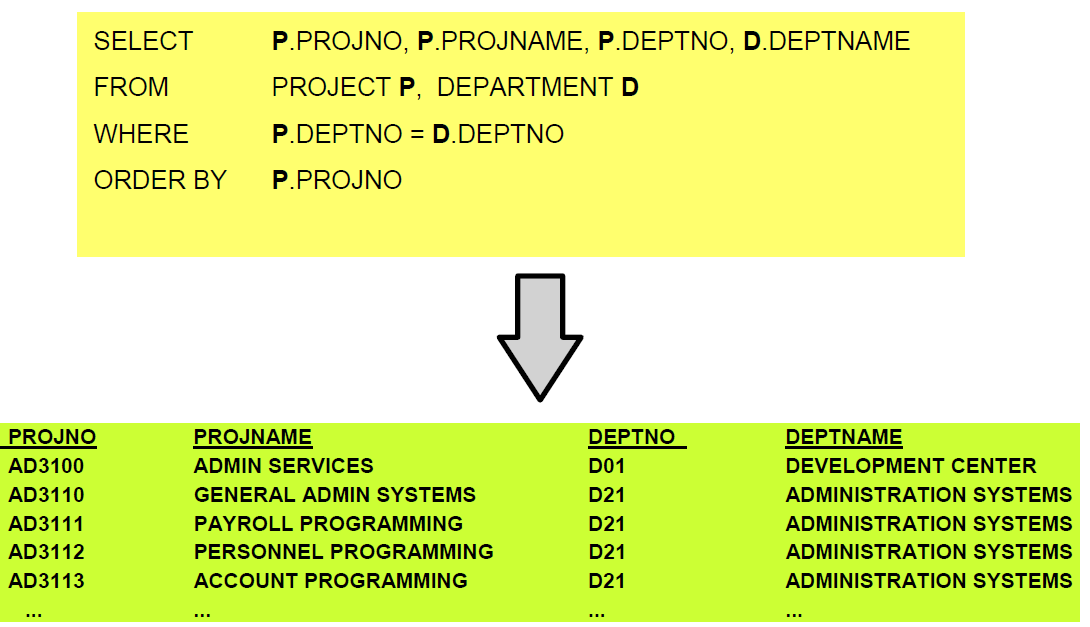


A join is an SQL statement that links several tables with one another and returns a join table as its result. In previous discussions, the SQL WHERE clause and relational expression was used to limit the number of rows retrieved.

A JOIN predicate is a SQL WHERE clause and relational expression which specifies "How the tables are to be joined". Notice that the FROM clause specifies the names of the tow tables to be joined. In this example, the common column DEPTNO is used to link both table when the DEPTNO's match. In order to use a common column to join a table the column names are NOT required to match, but the data type does. There is NO requirement to create a Foreign Key Constraint. But, since the WHERE clause specifies identical column names, e.g., DEPTNO, from each table, SQL will get confused unless you specify the table name as a qualifier, e.g., PROJECT.DEPTNO = DEPARTMENT.DEPTNO

## 1.2 Correlation Names and Table Aliases

**Correlation Names and Table Aliases**



A Correlation Name, or Table Alias, provides a temporary name to a table which simplifies the coding of complex join and SQL statements. In the previous example, the table named PROJECT was assigned a correlation name or table alias as P, table named DEPARTMENT was assigned a correlation name or table alias as D. As a result, the JOIN Predicate was coded as P.DEPTNO = D.DEPTNO. The correlation name is defined in the FROM clause and may be used anywhere in the SQL statement.

If there is a need to use this code simplification technique across multiple SQL statements, database or schemas, one will need to CREATE a SYNONMON for the table name. Database synonyms are objects that provides an alternative name to another object, e.g., similar to a nickname.

The advantages of using database synonyms also include:

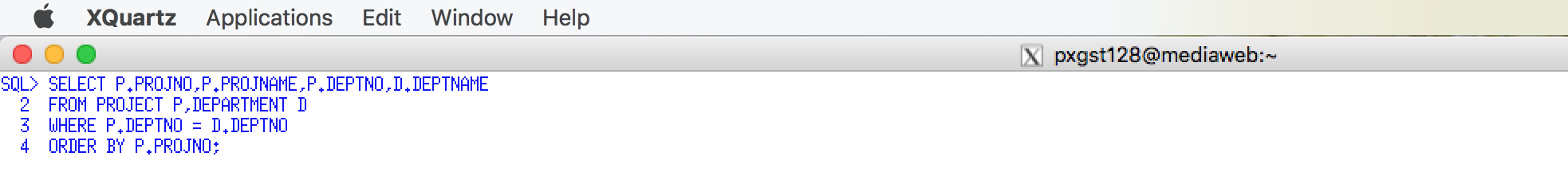
* Providing easy to use names for remote tables, i.e. over database links
* Tables that you need to be accessible to all users, i.e. public synonyms
* Testing - t is excellent for staging mock tables when testing. For example, if your source tables contain millions of records and you want to test a small subset of data, you can use synonyms to re-direct the source table to a smaller table you control so you can stage various scenarios. In this way, you can update/delete the data in the mock table without affecting your source table. When you are ready to use the source table, all you need to do is re-direct the synonym.
* Security benefits – Assume that the DBA wishes to separate database objects into different schemas, but wants/needs some of these objects to be visible to other schemas (but doesn't want to give direct access to them). An example I've seen most recently: Several web apps run by the same company. Users usually have access to more than one of these apps, and the will only have one user account to access these apps. User-account information is stored in a USER\_ACCOUNTS schema, and all other apps are in their own schemas and access the USER\_ACCOUNTS schema via synonyms.
* Database object names hard-coded within existing application code. Using synonyms might spare you the agony of rewriting old code, sometimes from multiple sources, which has its own ideas of the table or database names.

All SQL statements and output need to be documented using the Windows snipping tool or other comparable tool. To receive credit the QMF function key line must be clearly displayed.

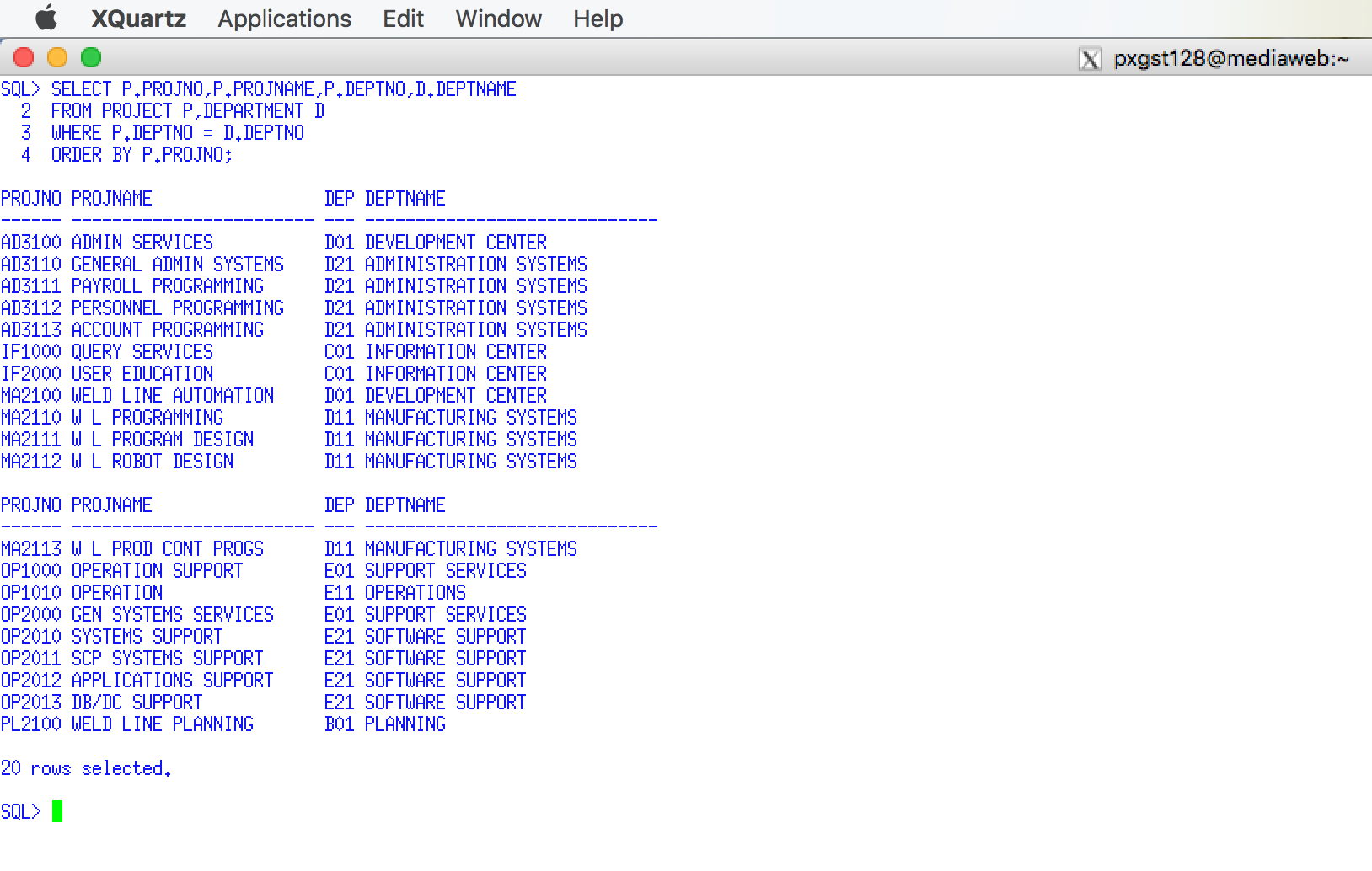
## 1.3 Questions - Multiple Table Joins

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

1. Retrieving Data from Multiple Tables (JOIN) - The previous example illustrates a simple join between the project and department tables which displays the project number, project name, department number, and department name. Execute this example and display the result. Use a snipping tool to document your SQL statement =>



Use a snipping tool to document your SQL output =>



2 What is a JOIN PREDICATE? Answer =>

A JOIN predicate is a SQL WHERE clause and relational expression which specifies the way in which the tables are to be joined.

Summer 2018

## 1.4 Questions - Types of Join

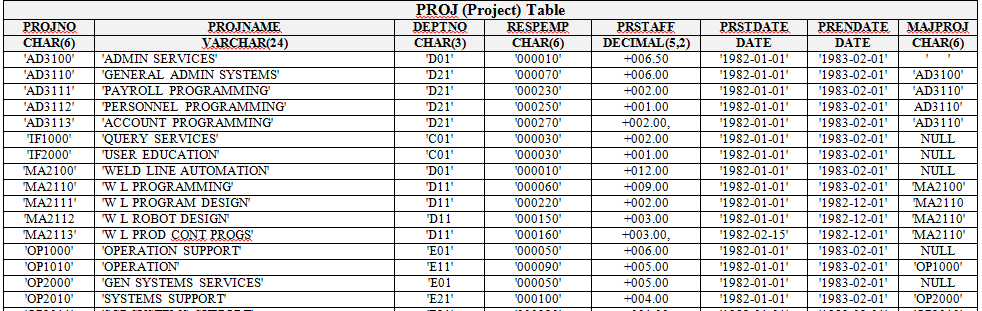
3. Using the following table answer the following questions by applying your answer to the Project and Department tables.

|  |  |
| --- | --- |
| **Type of Join** | **Definition using the Project and Department Tables as an Example**  Summer 2018 |
| **Inner Join** | Inner Join selects keyword that have matching records in two tables. In PROJECT table and DEPARTMENT table there is one common column name ‘DEP’. Inner join can be done based upon this column. |
| **Outer Join** | The FULL join keyword return all records when there is a match in either left (PROJECT) or right (DEPARTMENT) table records. Full join returns all records from the two tables with their matching values. |
| **Left Join** | The LEFT JOIN keyword returns all records from the left table here table1(PROJECT), and the matched records from the right table2 (DEPARTMENT). The result will be NULL from the right side, if there is no match. Here LEFT JOIN will display all records from PROJECT table regardless whether it will match to the DEPARTMENT table or not, but the DEPARTMENT table will return those rows which have matching values with PROJECT table. |
| **Right Join** | The RIGHT JOIN keyword returns all records from the right table2(DEPARTMENT), and the matched records from the left table1(PROJECT). The result will be NULL from the left side, when there is no match. Here RIGHT JOIN will display all records from the DEPARTMENT table, but the PROJECT table will return those rows which have matching values with DEPARTMENT table. |
| **Full Join** | The FULL join keyword return all records when there is a match in either left (PROJECT) or right (DEPARTMENT) table records. Full join returns all records from the two tables with their matching values. |

4. What is a Self-Join? Answer =>

SQL SELF JOIN is used to join or compare a table to itself. SQL self joins are used to compare values of a column with values of another column in the same table.

5. Given the following Project Table columns and sample data.



List two columns that would be in a Self-Join. List the reason WHY these columns should be candidates for a self-join. Answer =>

PROJNAME, DEPTNO would be the two columns for self join. one Department can have multiple employees. It can be sorted out by the PROJNAME.

Summer 2018

## 1.5 Questions - Avoiding Cartesian Product Joins

Avoiding Cartesian Product Joins –

<http://www.rampant-books.com/t_hpsdba_77_cartesian_join_operations.htm>

6. What is a Cartesian Product Join? What is the cause of this popular error? Answer =>

Cartesian product, also referred to as a cross-join, returns all the tables listed in the query.

Each row in the first table is paired with all the rows in the second table. This happen when there is no relationship defined between two tables.

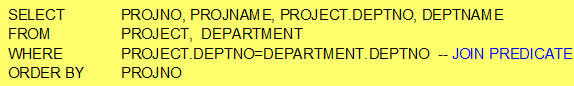
A Cartesian product returns all combinations of rows from the tables referenced by objects in the query.As two tables in the query have not paired together through a common relational column that is present in both tables, therefore, SQL server will have to utilize a Cartesian join. And this join multiplies the number of rows in the first table by the number of rows in the second table and the result set becomes large and rarely correct.

Summer 2018

## 1.6 Question – Correlation names, Table Alias, and Synonym

Correlation names - <http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=%2Fdb2%2Frbafzmstc2cornm.htm>

SQL joins are created by matching common fields, which frequently has the same column name. For example, the previous join matched DEPTNOs, hence the same name. All versions of SQL require one to uniquely define a column name. Let's review the syntax of the first join example



What is a Correlation Name? Answer =>

A correlation name is given to a table expression in a FROM clause as a new name or alias for that table.

Summer 2018

What is the relationship between a Correlation Name and a Table Alias? Answer =>

Aliases provide the ability to reduce the amount of code required for a query, and to make queries simpler to under stand. A table alias is also called correlation name.

What are the differences between a Correlation Name, Table Alias and a Database Synonym? Answer =>

A synonym is a database object that provides an alternative name for another database object, referred to as the base object, that can exist on a local or remote server.

An alias of a table is a way to give the table a name to more easily reference the table. Alias is temporary and used with one quirry. Synonym is permanent and not used a s alias.

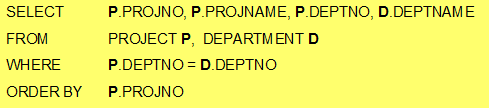
List some advantages of a Database Synonym? Answer =>

SYNONYMs provide s layer of abstraction over the refrenced object. It allows changes to complicated and lengthy names with a simplified alias as a same server resident object. It privides flexibility for changing the location of objects without changing existing code. SYNONYMs can be created in the same database to provide backward compatibility for older applications in case of drop or rename of objects.

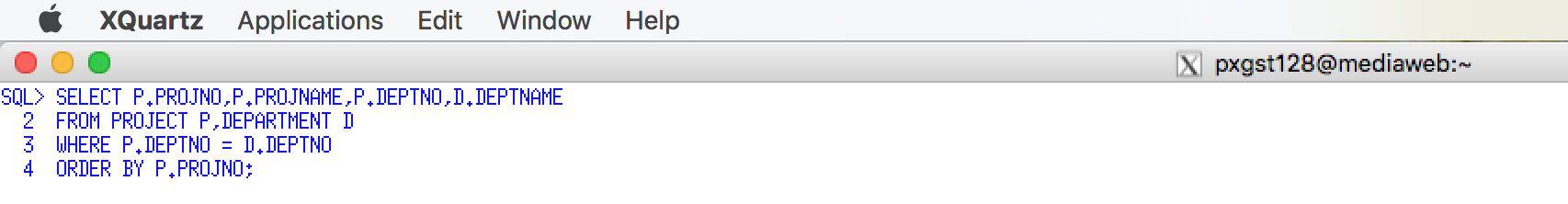
Summer 2018

The SELECT and WHERE clause uniquely describes duplicate column names by prefixing the column name with the table name, e.g., PROJECT.DEPTNO and DEPARTMENT.DEPTNO. The TableName.ColumnName format gets to be very cumbersome in complicate joins.

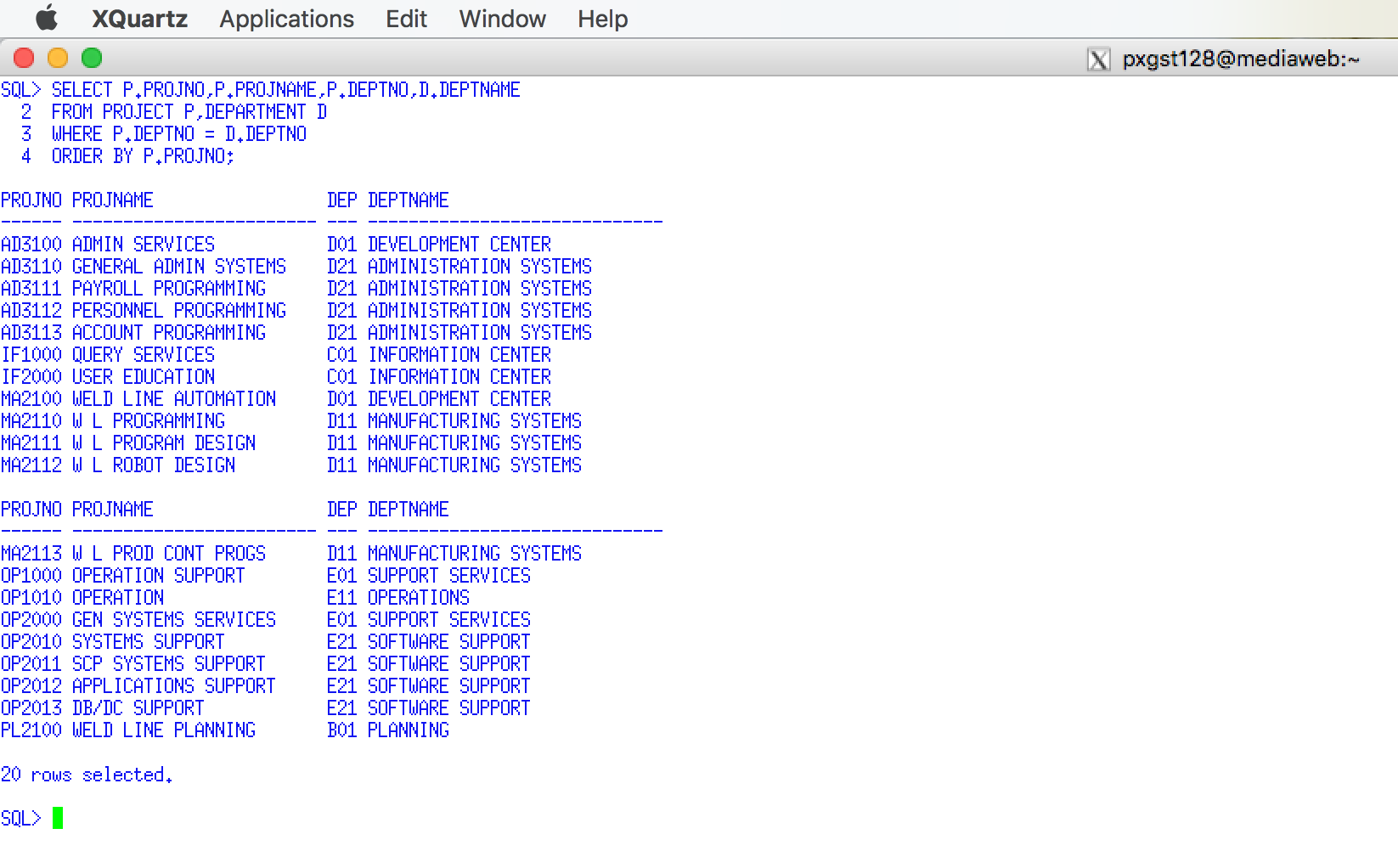
A SQL correlation simply provides an abbreviation for the table name. It is defined using the FROM clause. The correlations (abbreviations) are P and D. Review the following correlation version of the previous join.



7. Execute the previous SQL Join Correlation. Use a Snippit to document your SQL statement =>



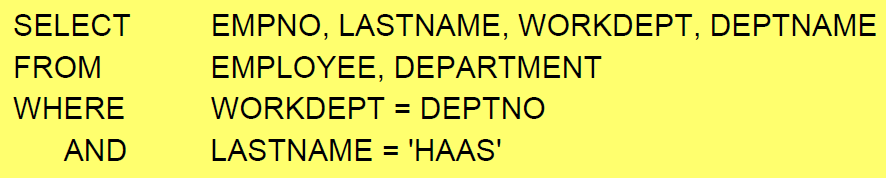
Use a Snippit to document your SQL output =>



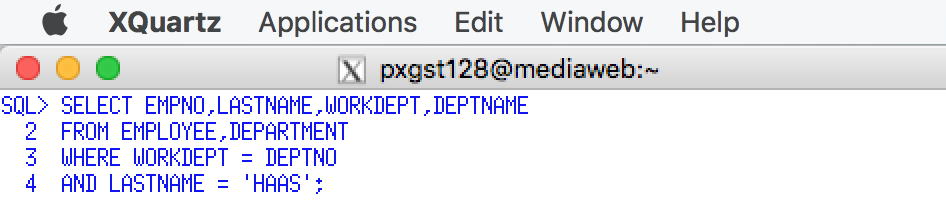
**Use a graphical snipping tool to document an image of these requirements below. Make sure your Putty banner is displayed.**

## 1.7 Question – Joining Two Tables

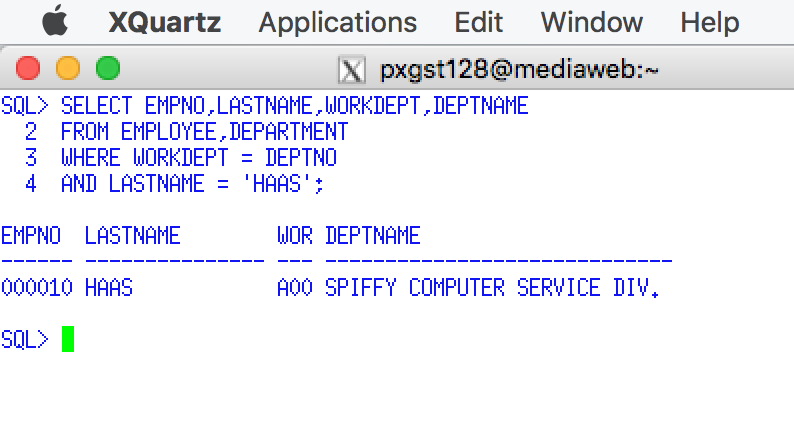
**8. JOIN Syntax 1** - You are required to display employees with a last name of HAAS, display the employee number, last name, and the department number and department name of the department they are working in.



Execute this example and display the result🡺



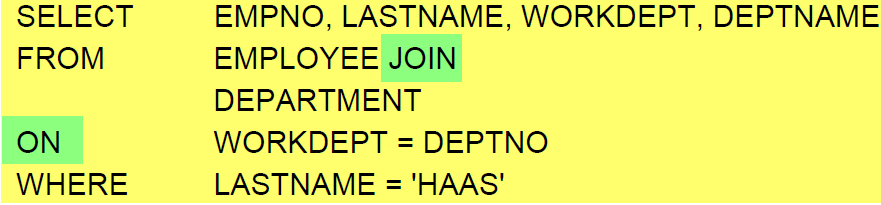
Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



**9. JOIN Syntax 2 (JOIN Keyword) -** You are required to display employees with a last name of HAAS, display the employee number, last name, and the department number and department name of the department they are working by using the JOIN keyword.

This example uses the JOIN and ON keywords instead of a JOIN predicate. This will simplify the coding and separation of the join predicate and other conditional expressions. Notice that the ON clause specifies the join predicate and the WHERE clause lists other conditional expressions.

If you simply code the JOIN operator, it is the same as coding INNER JOIN.



**SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME**

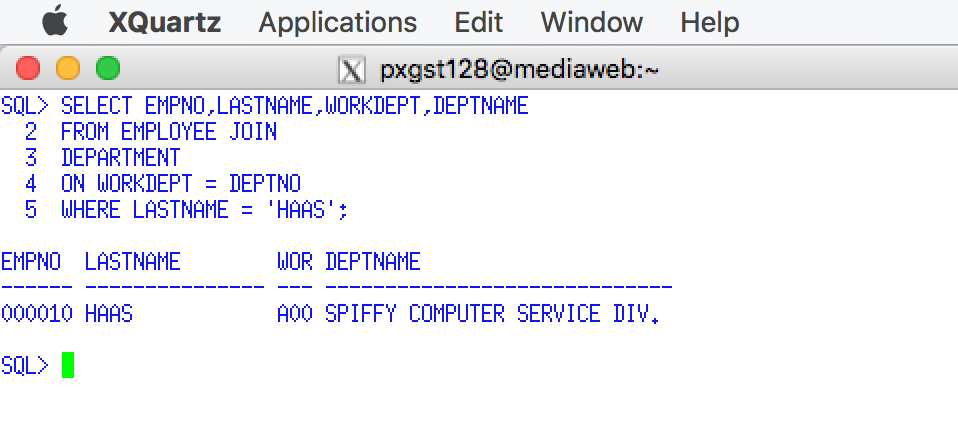
**FROM EMP JOIN**

**DEPT**

**ON WORKDEPT = DEPTNO**

**WHERE LASTNAME = 'HAAS'**

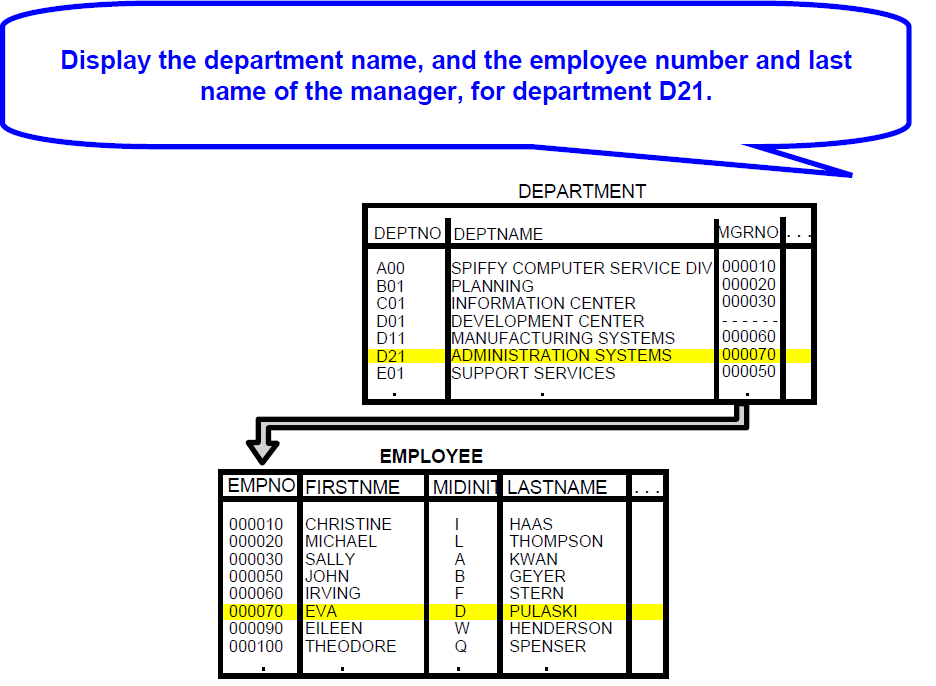
Execute this example and display the result =>

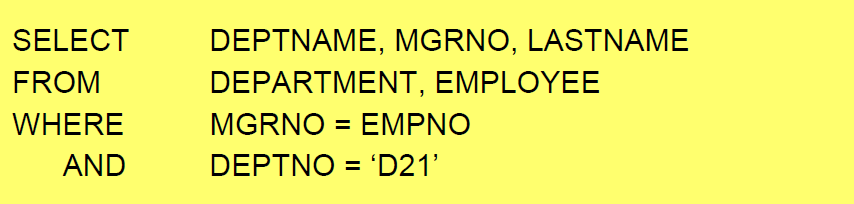


Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

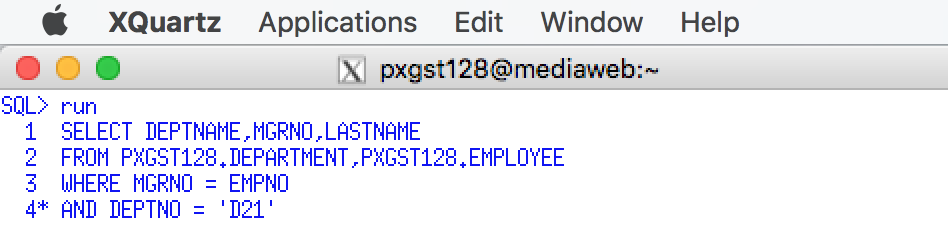


10. Using a Join display the department name, and the employee number and last name of the manager, for department D21. Execute this example.

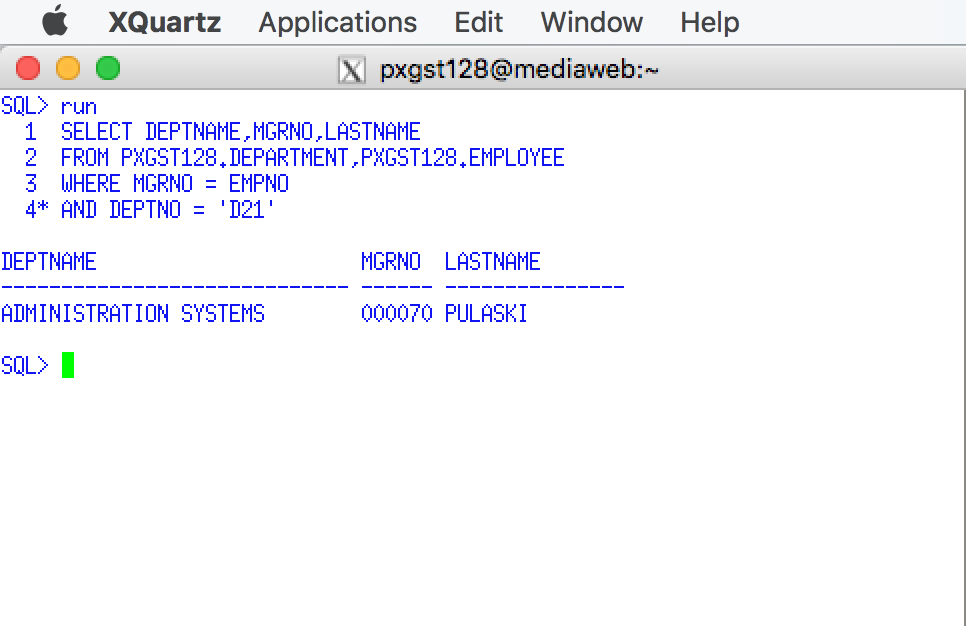




Use a Snippit to document your SQL statement =>

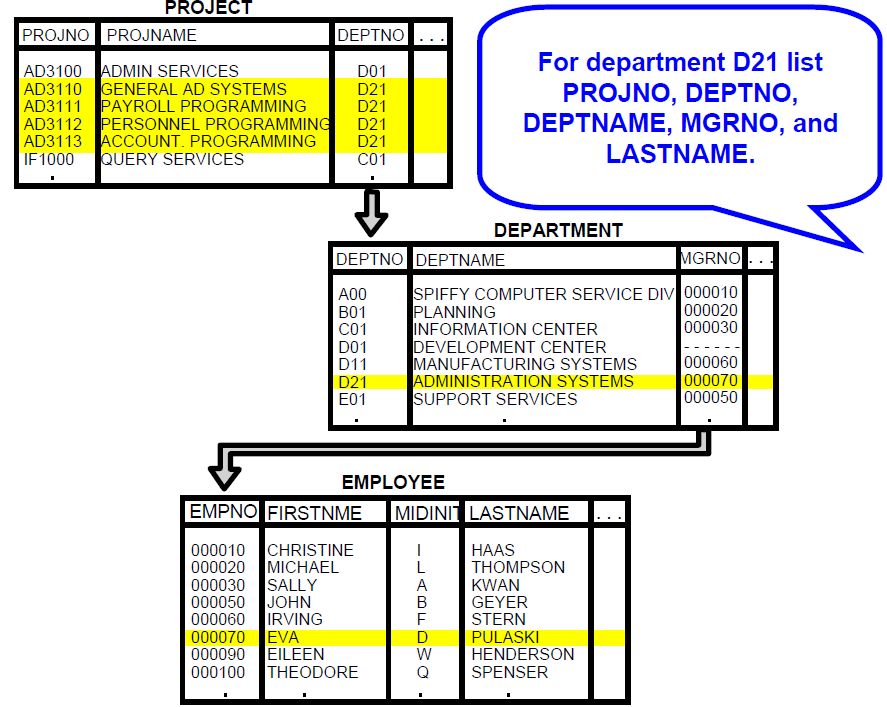


Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



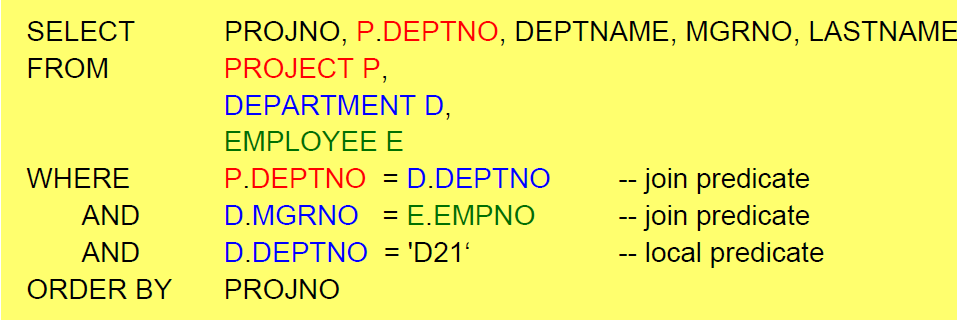
## 1.8 Questions - Joining Three Tables

**11. Join three tables** - For department D21 list project number, department number, department name, manager number, and employee last name. Execute this example and display the result.

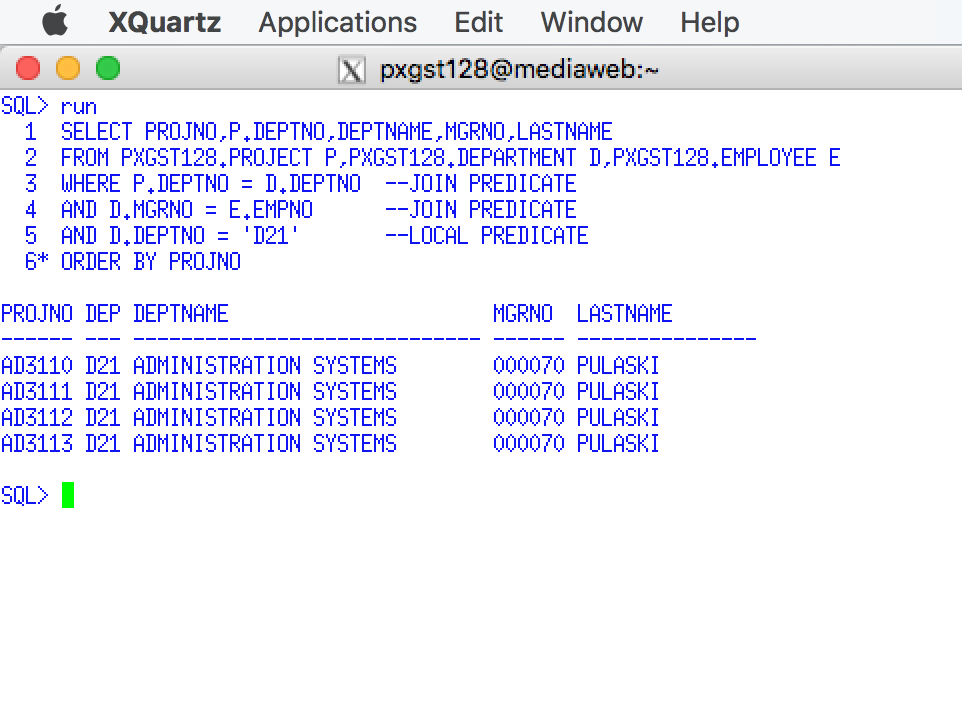


**Use a graphical snipping tool to document an image of these requirements below.**

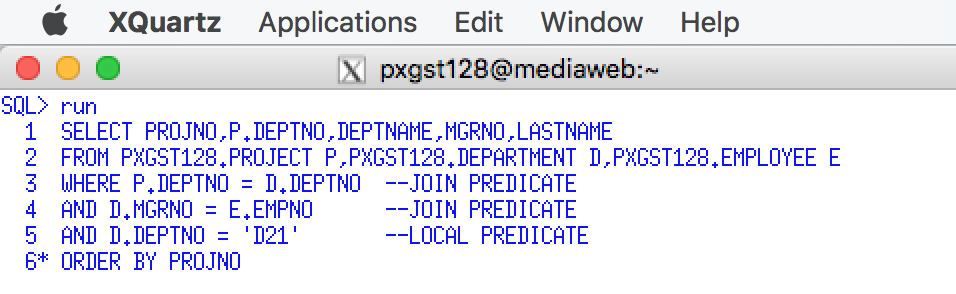
**Make sure your Putty banner is displayed.**



Use a Snippit to document your SQL output =>



Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



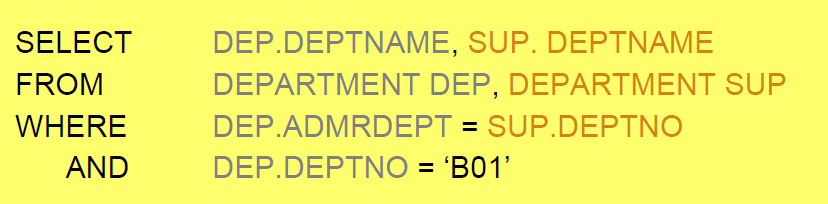
Review: You will notice the two hyphens in front of the join predicate, e.g., --join predicate. What is the purpose of the two hyphens in SQL? Answer =>

Two hyphens mean the command is not an executable command.

## 1.9 Questions - Self-Join

**12. Joining a Table with Itself (Self-Join)**. Display the name of department B01 and the name of the department it reports to. The column named ADMRDEPT contains the value of DEPTNO that this row reports to. For example, DEPTNO A00, reports to itself, but DEPTNO B01 reports to department A00

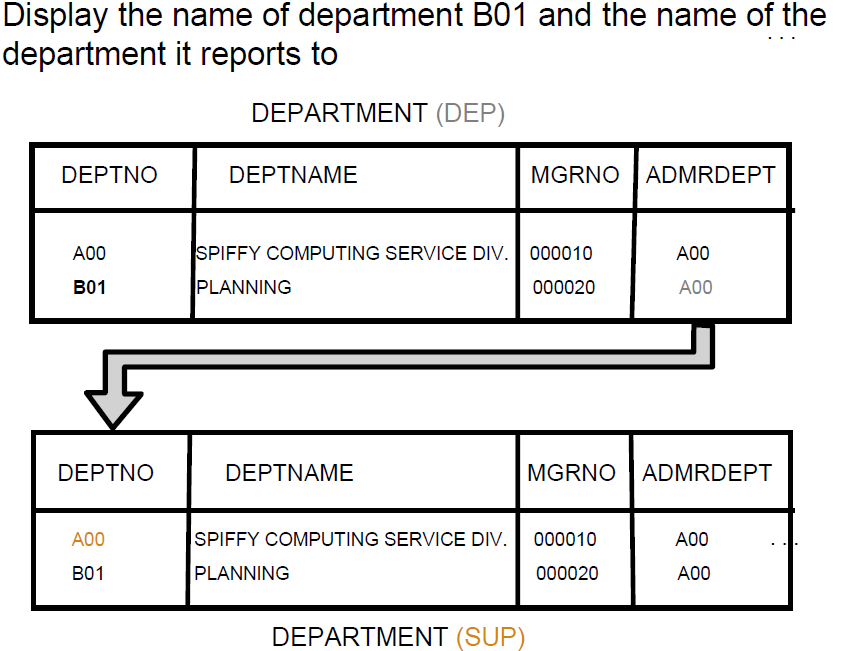




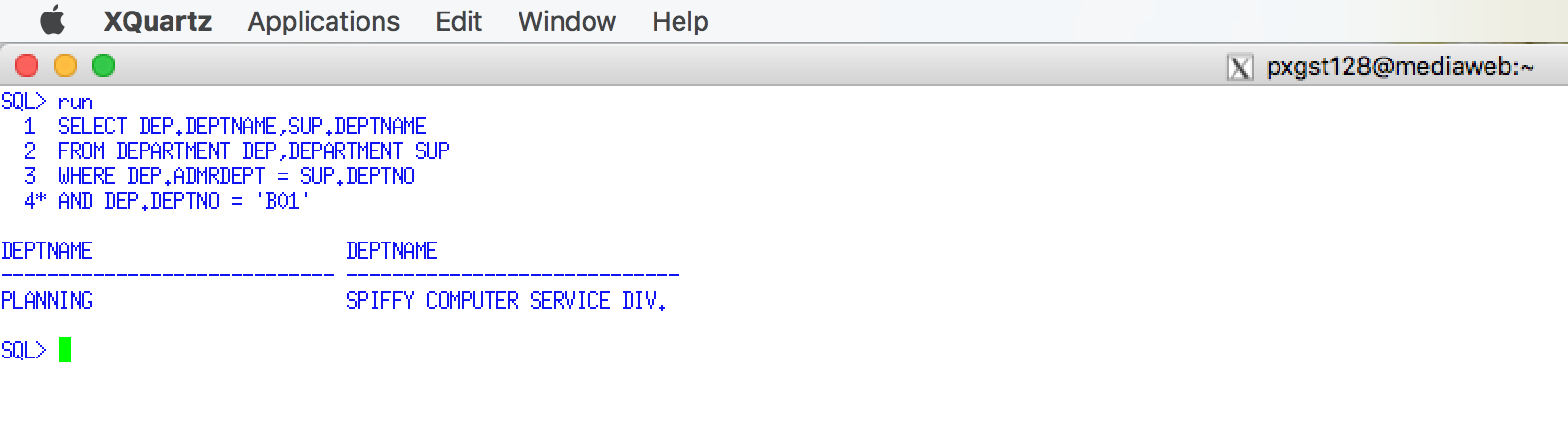
Since the FROM clause specifies the DEPARTMENT table or view twice and correlation name or table alias is used to distinguish between the two copies of the same table. A self-join enables SQL to simultaneously compare two rows of the same table. SQL can view both pages of the text book simultaneously, but you must finger flip between the two pages.

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

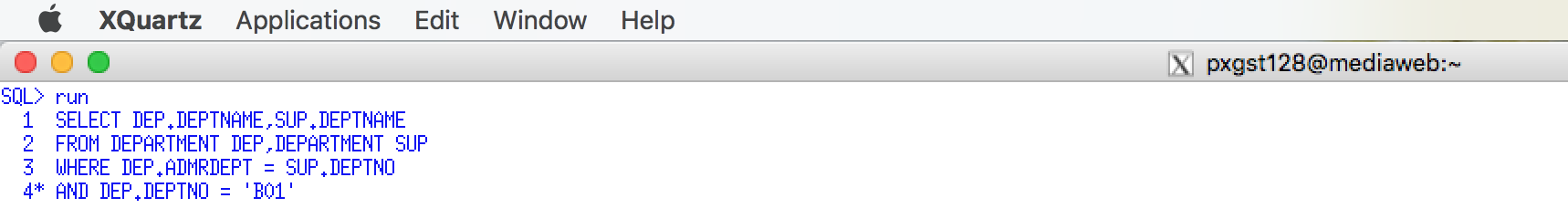
**Make sure your Putty banner is displayed.**



Execute this example and display the result. Use a Snippit to document your SQL output =>

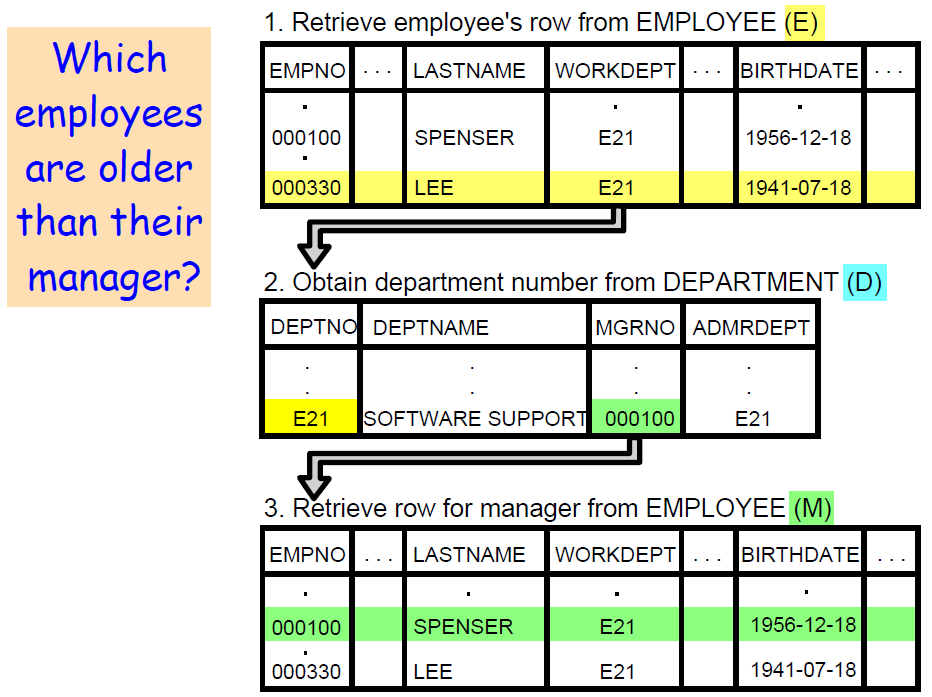


Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



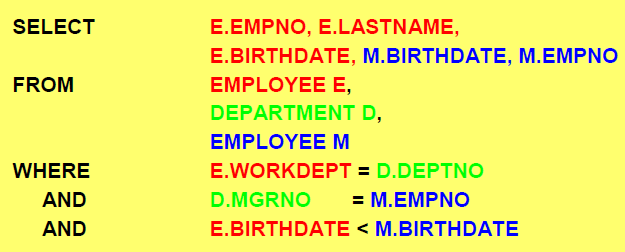
## 1.10 Questions - Joining a Table with Itself

**13. Joining a Table with Itself.** Display the employee name and number of employees who are older than their manager.

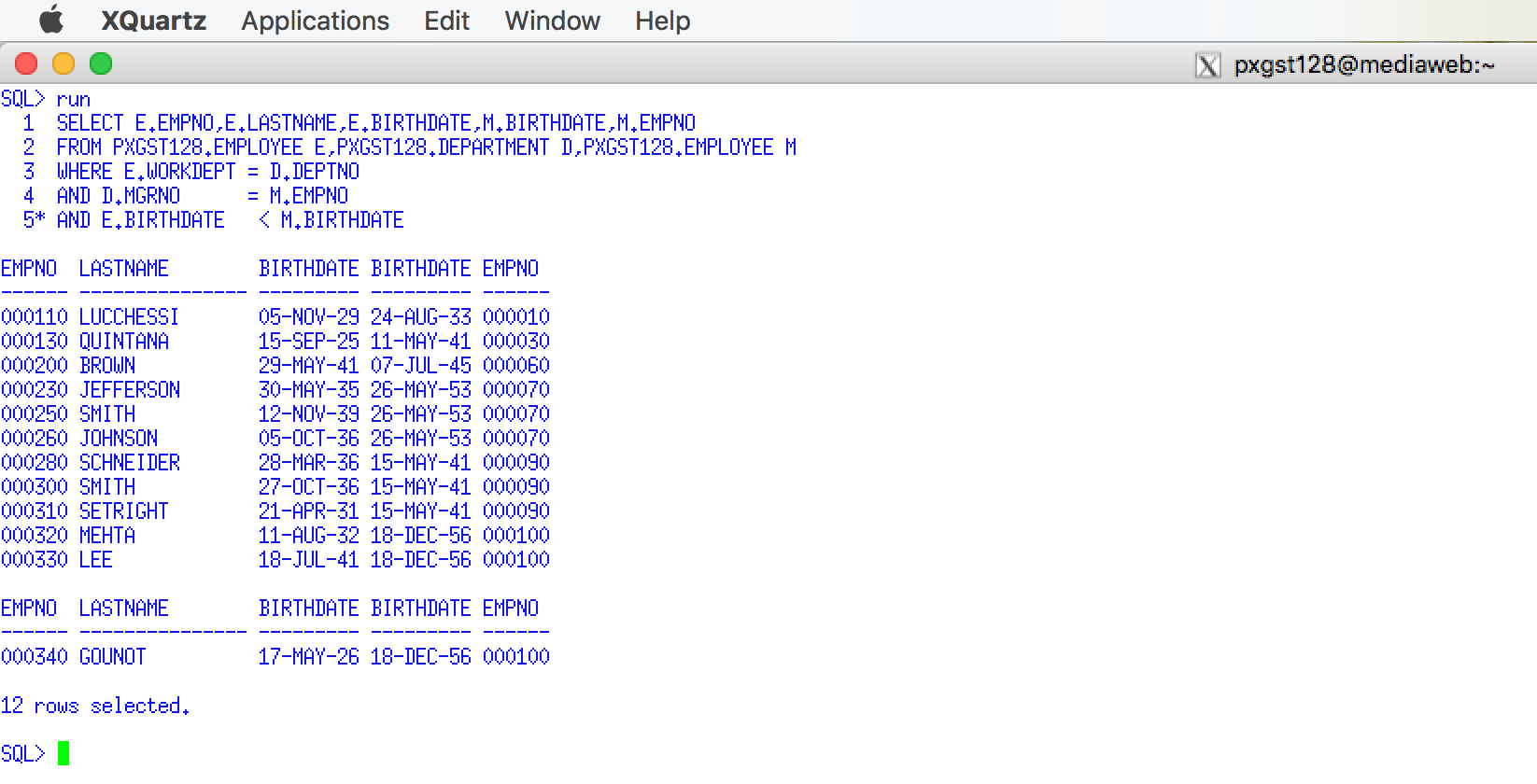


**Use a graphical snipping tool to document an image of these requirements below.**

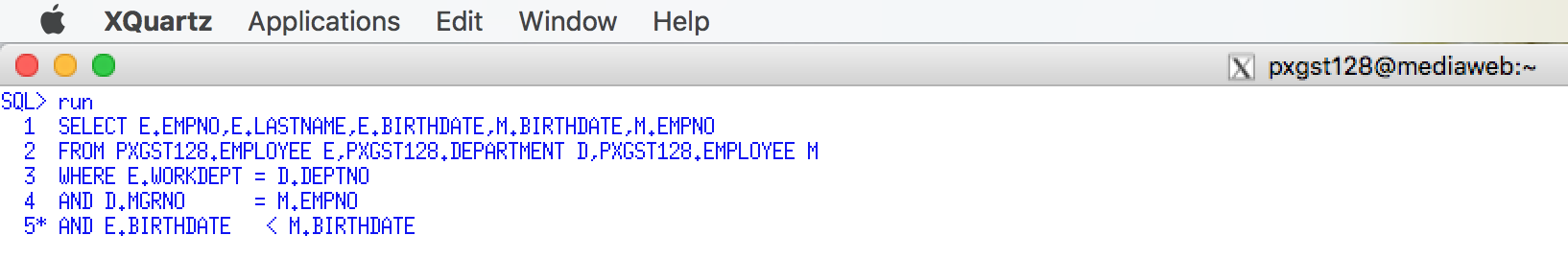
**Make sure your Putty banner is displayed.**



Execute this example and display the result. Use a Snippit to document your SQL output =>



Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



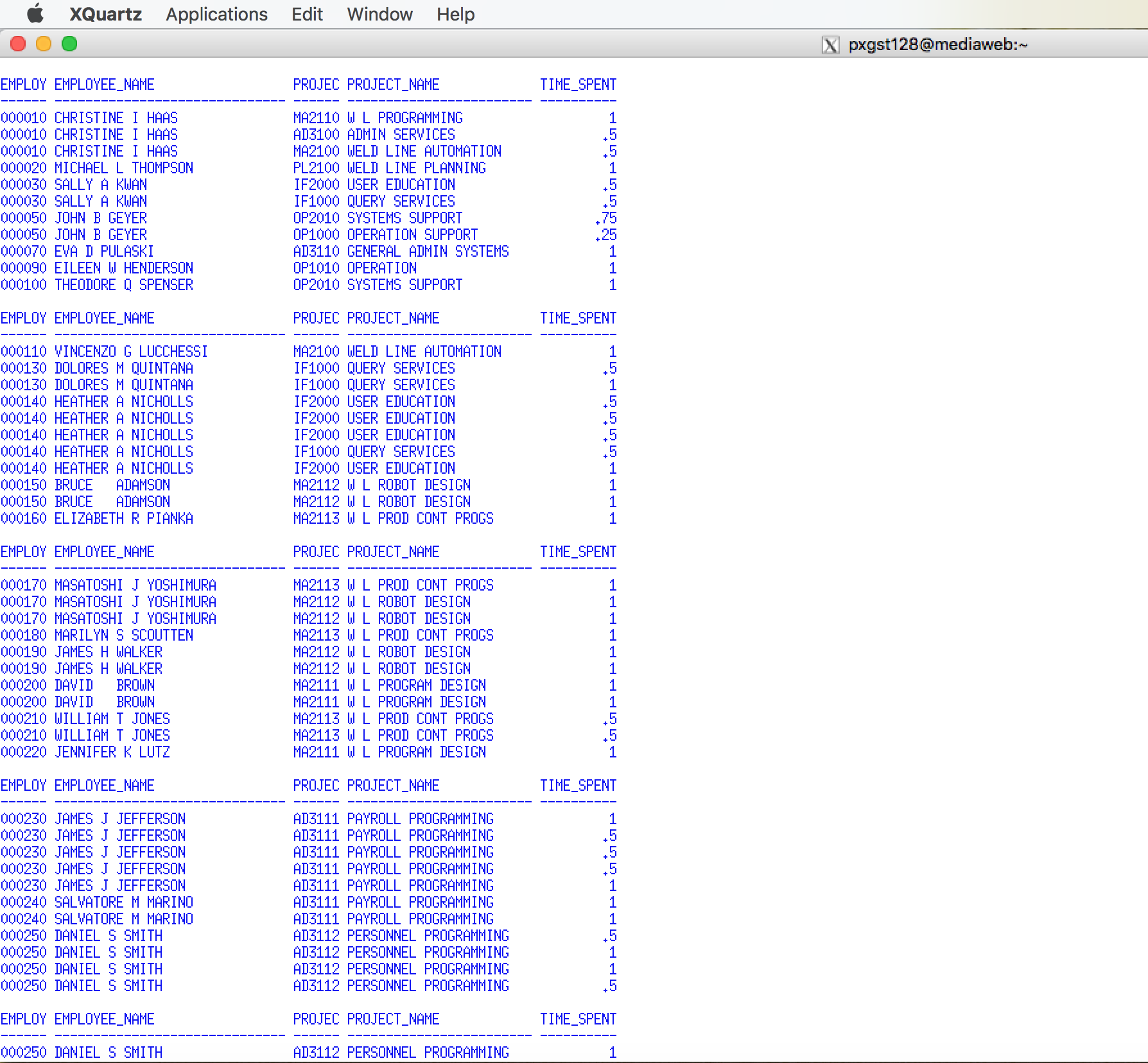
## 1.11 Questions - Your Join Examples

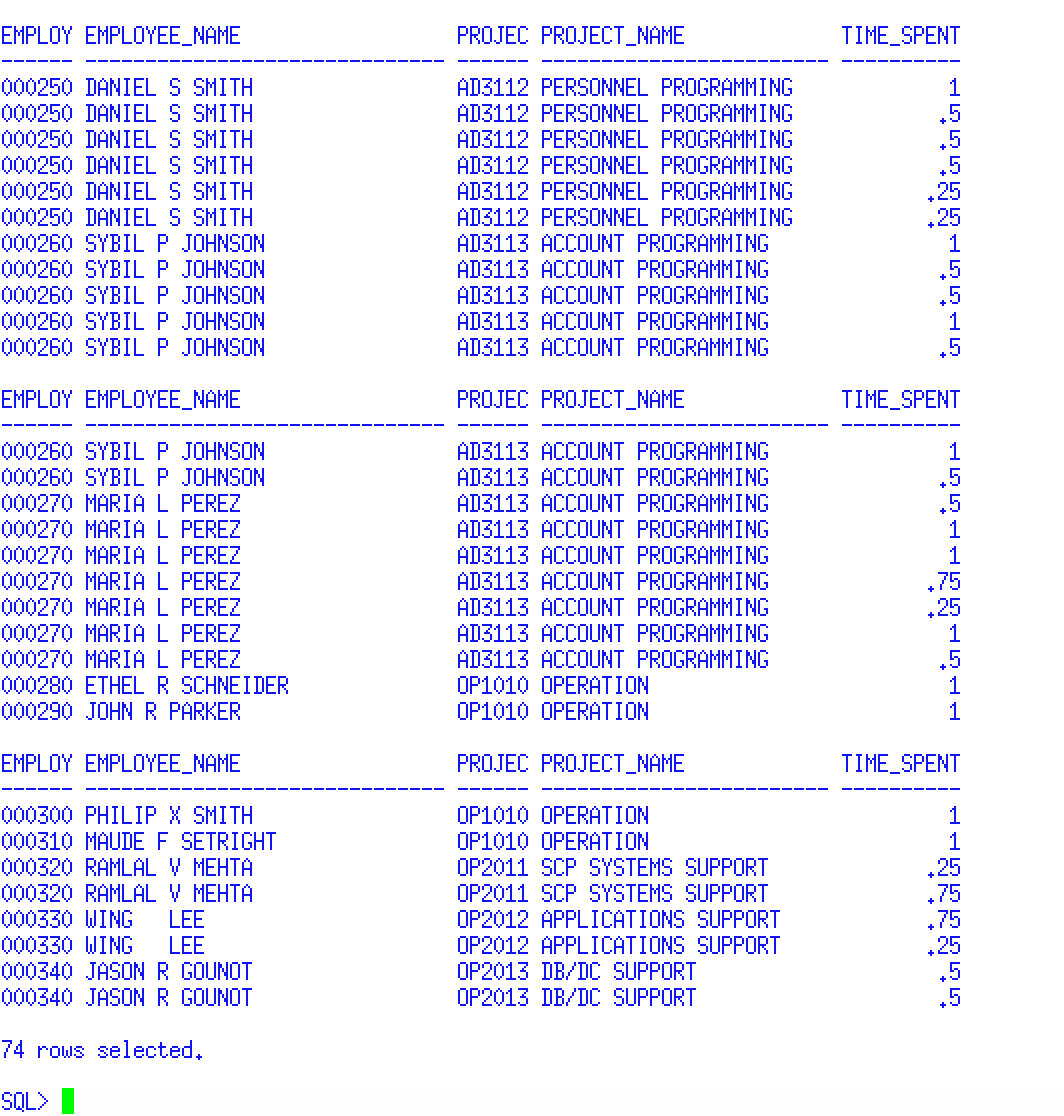
**Use a graphical snipping tool to document an image of these requirements below.**

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

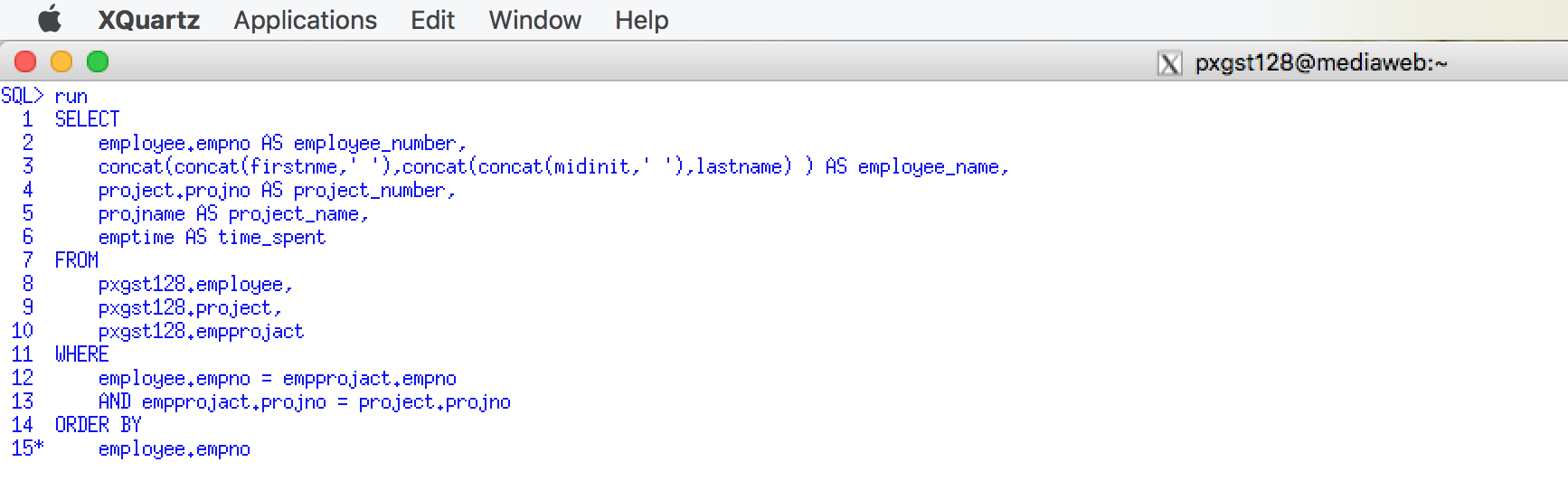
**14. Your Requirement -** Using the EMPLOYEE, PROJECT and EMPPROJACT tables display the employee number, employee name, project number, project name and time spent (emptime), sorted by employee number.

Execute this example. Use a Snippit to document your SQL statement =>



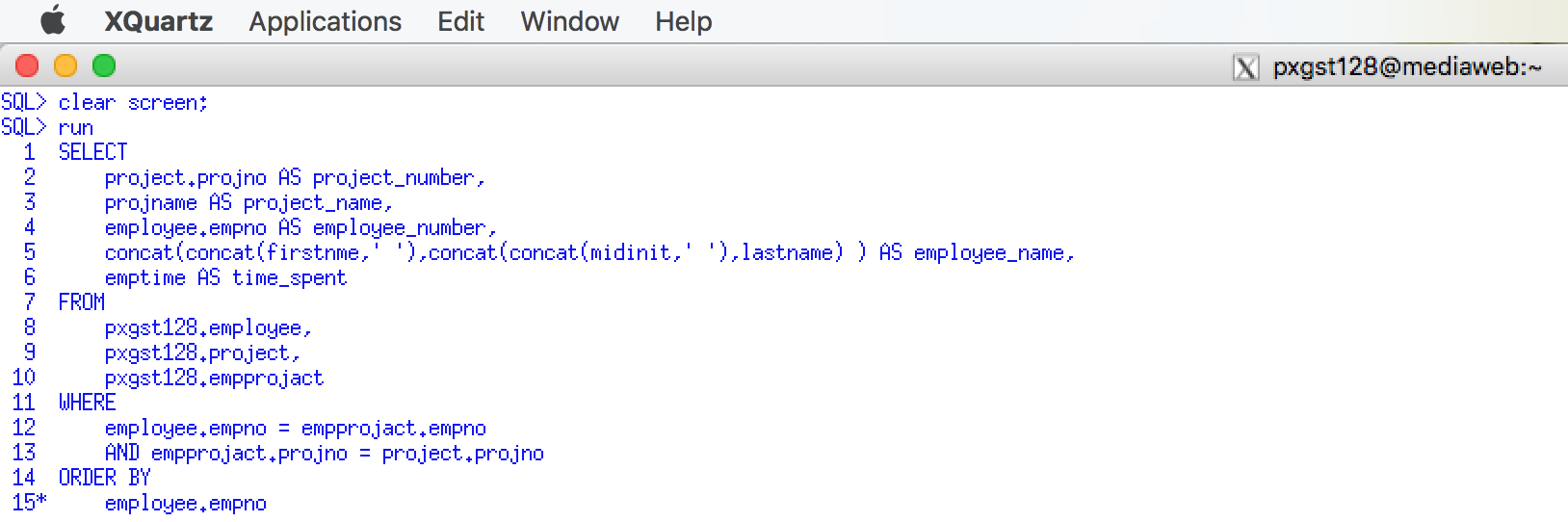


Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

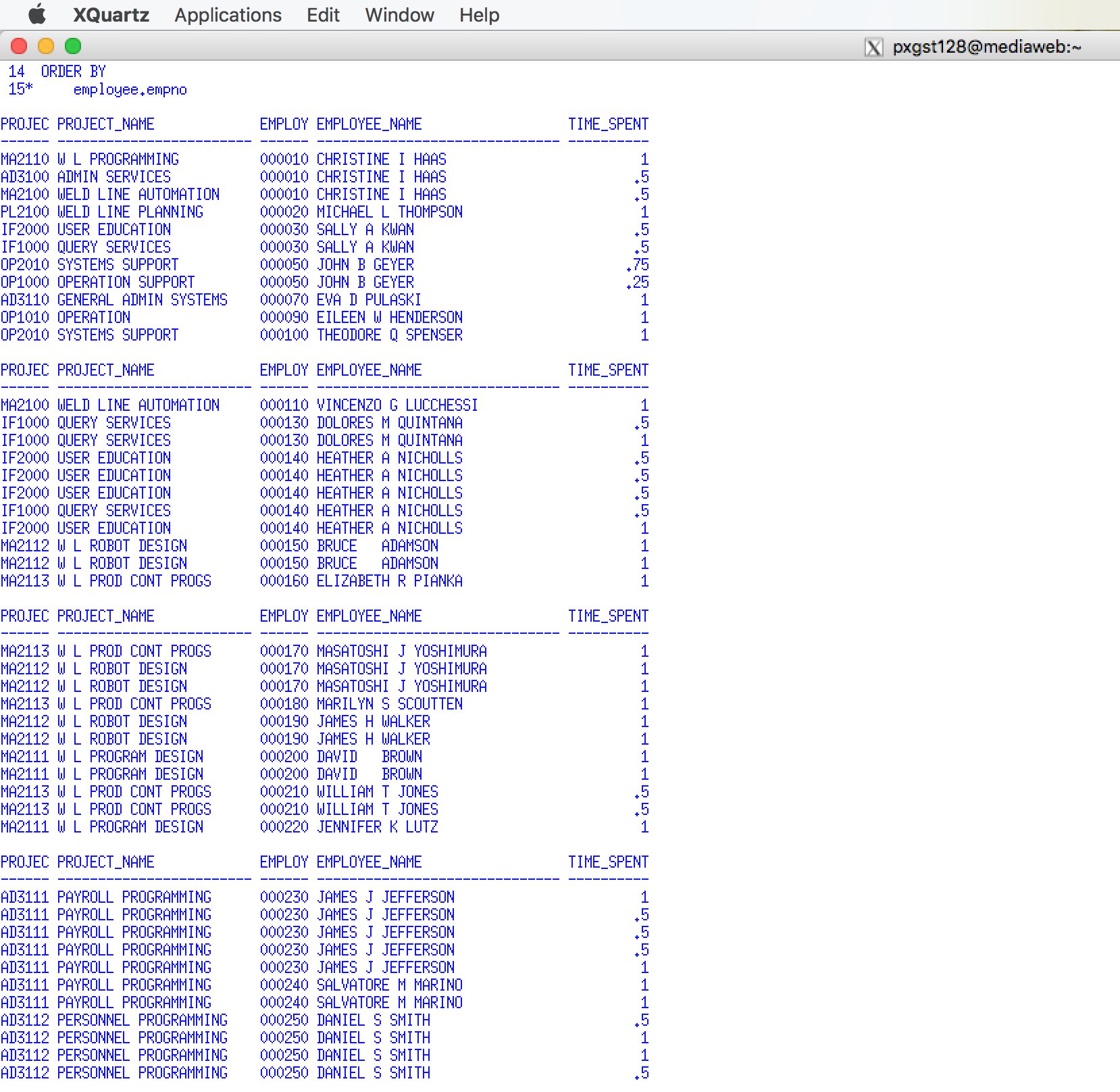


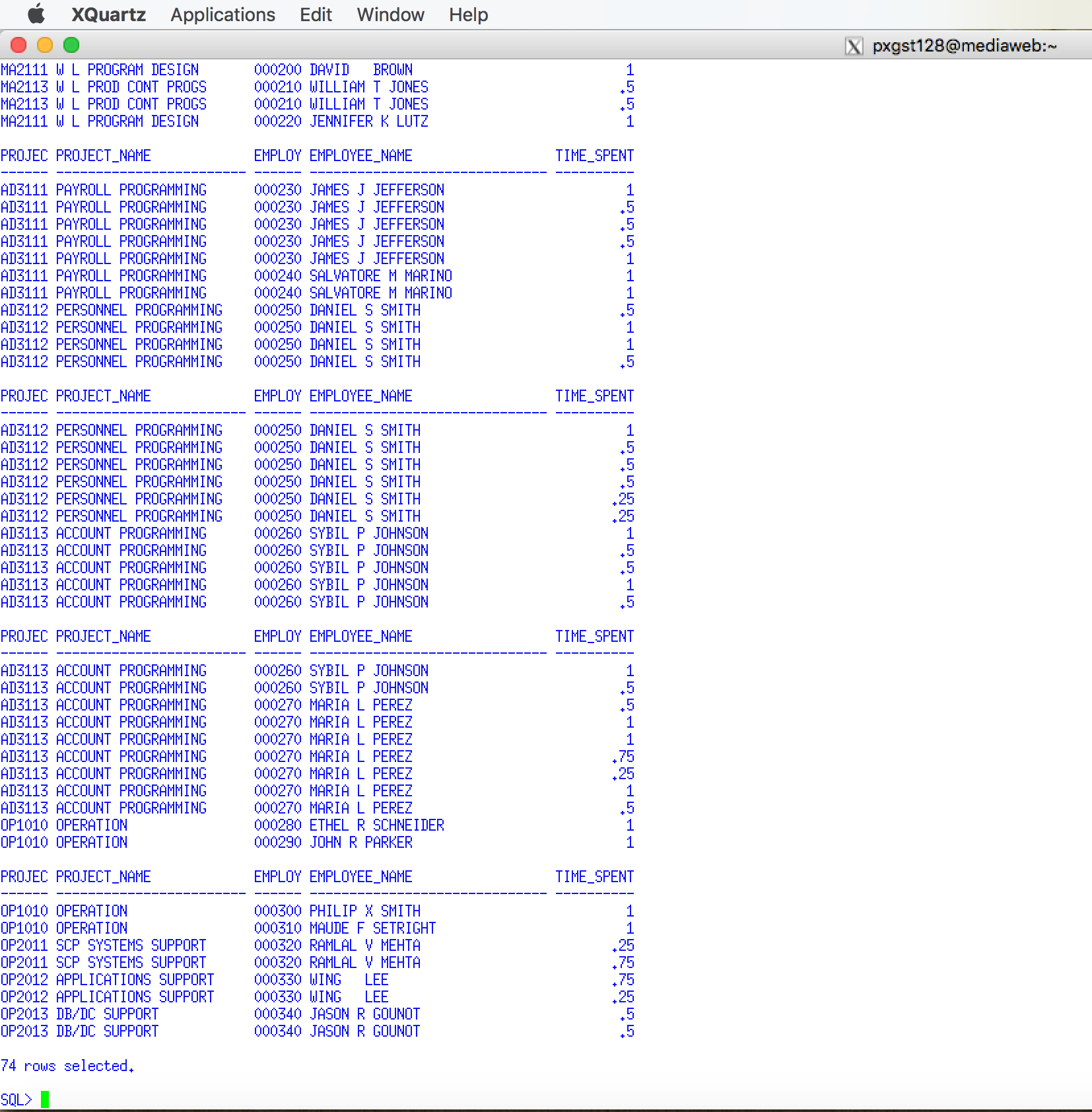
**15. Your Requirement -** Using the EMPLOYEE, PROJECT and EMPPROJACT tables display the project number, , project name, employee number, employee and time spent (emptime), sorted by employee number Execute this example.

Use a Snippit to document your SQL statement =>



Use a Snippit to document your SQL output =>





## 1.12 Questions - Left and Right Joins

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

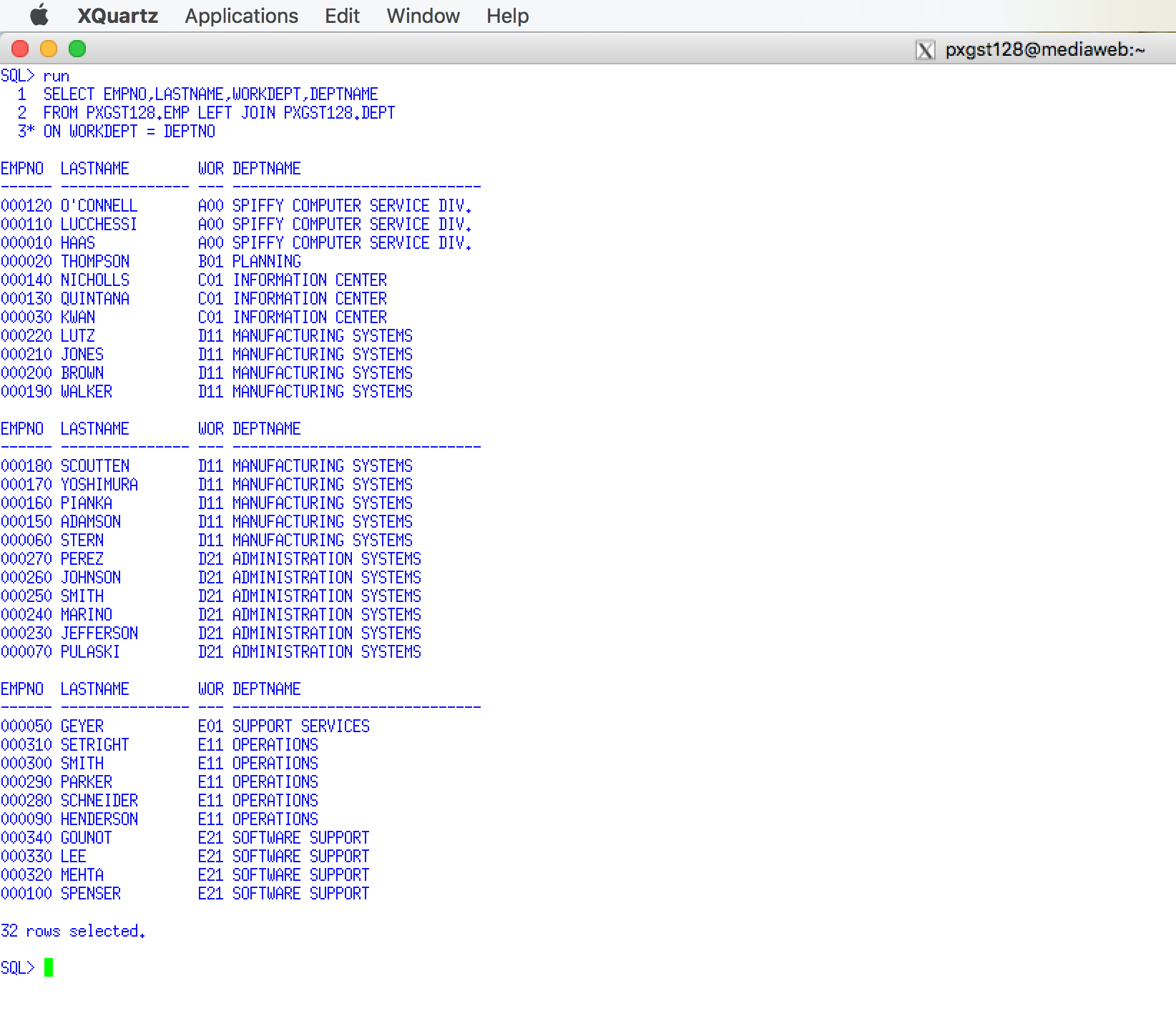
**Make sure your Putty banner is displayed.**

**16. Left Join.** Display ALL employee numbers, last names, and the WORKDEPT number and department names of the department where they are working for. The inner join will only display rows in which an employee has been assigned to a valid department number stored in the department table. If a match is not found, this LEFT JOIN example will display ALL employees and list a NULL value for the department name.

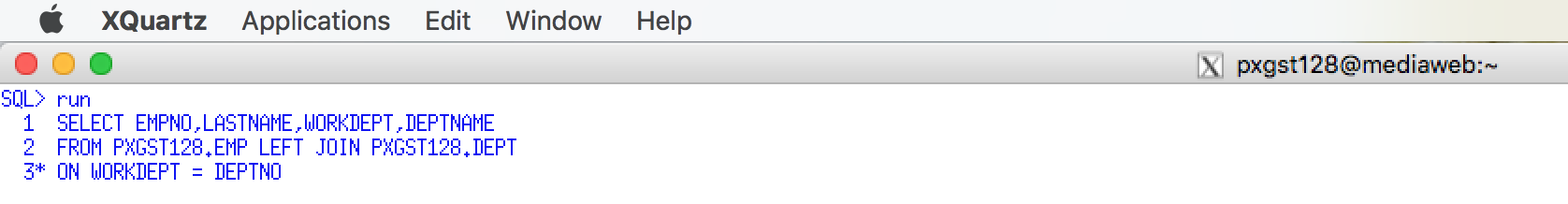
Some business requirements permit an employee not to be assigned to a department or a DEPTNO not found in the DEPARTMENT table. A Left Join will assist finding these employees. On the other hand, other business requirements require an employee to be assigned to a valid department. Your sample database does not use a Foreign Key constraint to enforce this referential integrity, but the authors who created the sample data did ensure that this constraint was manually enforced.



Use a Snippit to document your SQL statement =>



Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



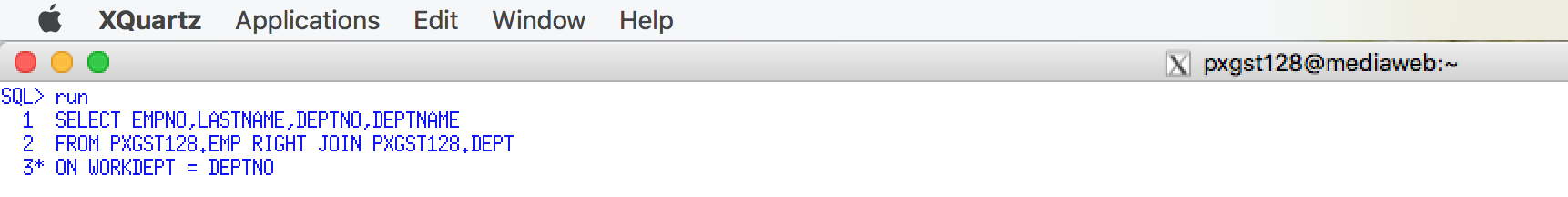
**17. Right Join.** Display ALL employee numbers, last names, and the department (DEPTNO) number and department names of the department for ALL departments, whether or not any employee has been assigned to the department. The inner join will only display rows in which an employee has been assigned to a valid department number stored in the department table. If a match is not found, this LEFT JOIN will list ALL department names and will display NULL values for EMPNO and LASTNAME.

Most business requirements will a department to exist with any assigned employees. From the point of a view of business requirements, a foreign key constraint would not be appropriate. From the point of view of SQL, you cannot code a circular foreign key constraint between two tables. It is illogical that a parent of a child, that that child is the parent of their parent.

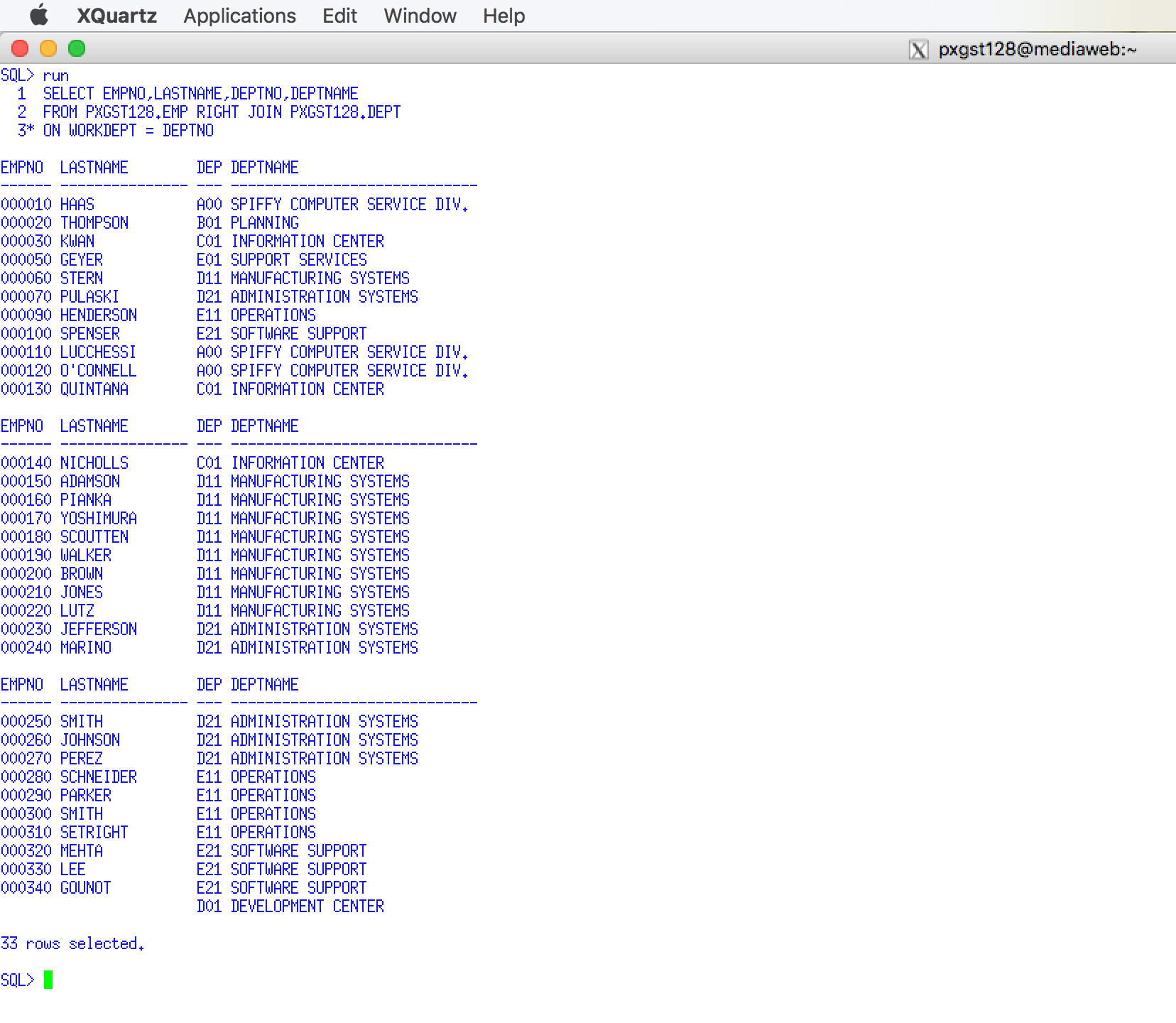


**Make sure your Putty banner is displayed.**

Use a Snippit to document your SQL statement =>



Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



The previous SQL example used the JOIN predicate format. The following will use the JOIN/ON SQL statement. When you join multiple tables it is very easy to make coding errors. The previous example joined three tables. Based on the principle "Keep it Simple", we will incrementally code the following example by simply coding and testing two tables at a time.

**Example 1. Join the PROJ and DEPT tables**

**SELECT PROJNO, P.DEPTNO, DEPTNAME, MGRNO**

**FROM PROJ P**

**INNER JOIN DEPT D**

**ON P.DEPTNO = D.DEPTNO**

**Example 2. Join the PROJ and DEPT tables, and then join the EMP table and add EMP column names to the SELECT line**

**SELECT PROJNO, P.DEPTNO, DEPTNAME, MGRNO, LASTNAME**

**FROM PROJ P**

**INNER JOIN DEPT D**

**ON P.DEPTNO = D.DEPTNO**

**INNER JOIN EMP E**

**ON D.MGRNO = E.EMPNO**

**Example 3. Add any other WHERE or ORDER BY statements, formatting or user-friendly column aliases**

**SELECT PROJNO, P.DEPTNO, DEPTNAME, MGRNO, LASTNAME**

**FROM PROJ P**

**INNER JOIN DEPT D**

**ON P.DEPTNO = D.DEPTNO**

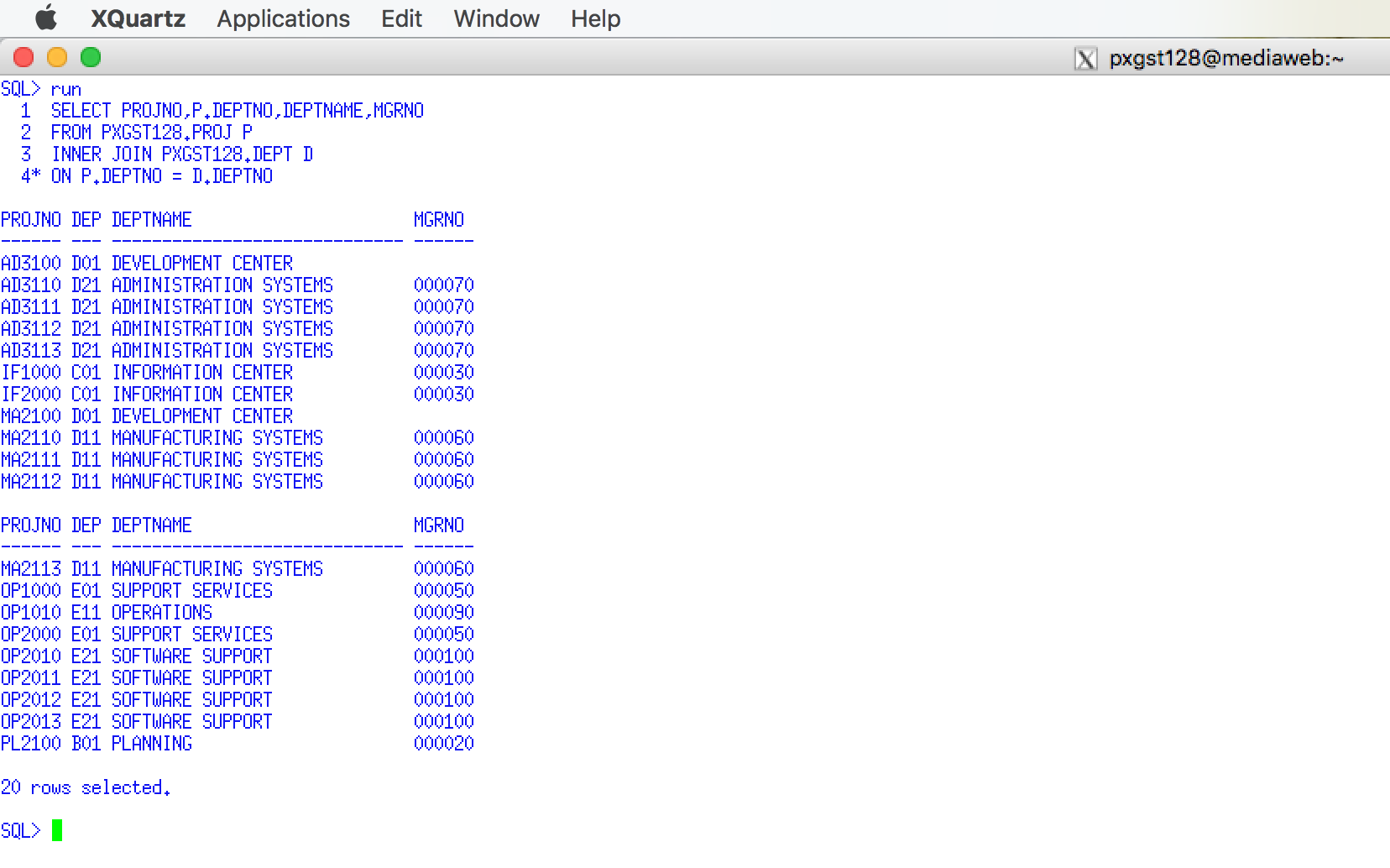
**INNER JOIN EMP E**

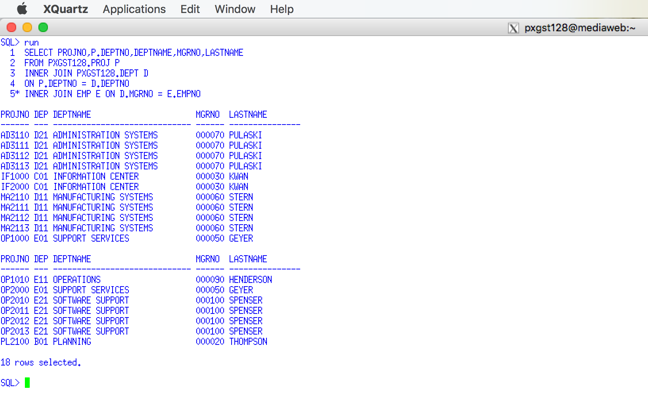
**ON D.MGRNO = E.EMPNO**

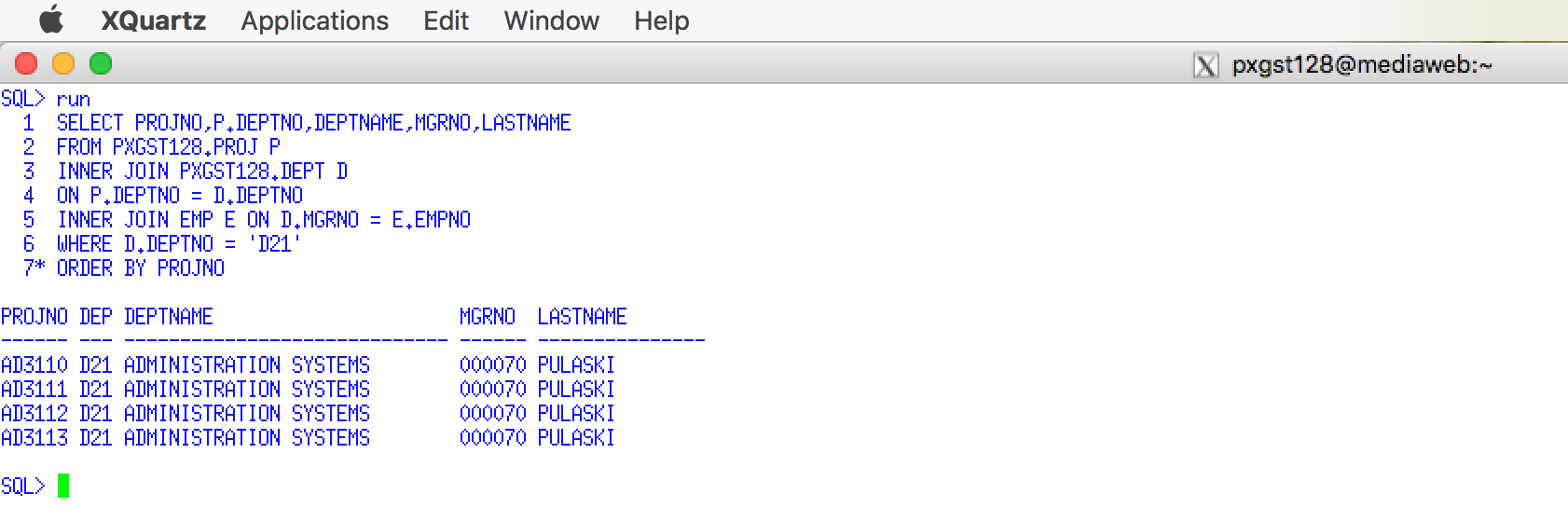
**WHERE D.DEPTNO ='D21'**

**ORDER BY PROJNO**

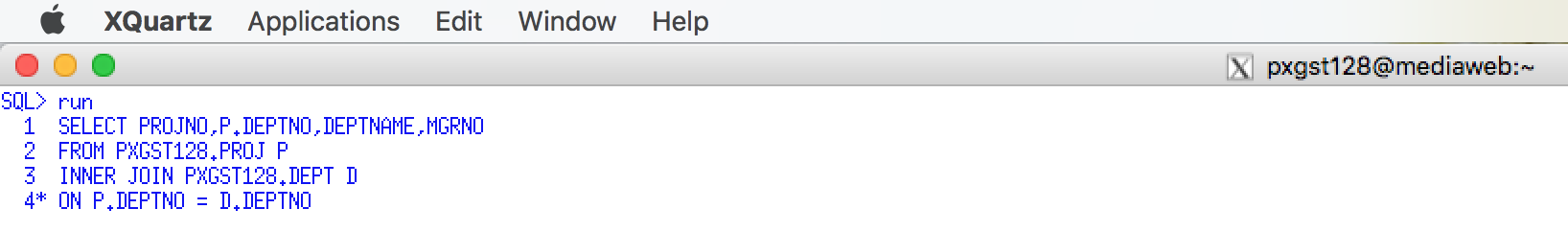
Use a Snippit to document your SQL output =>

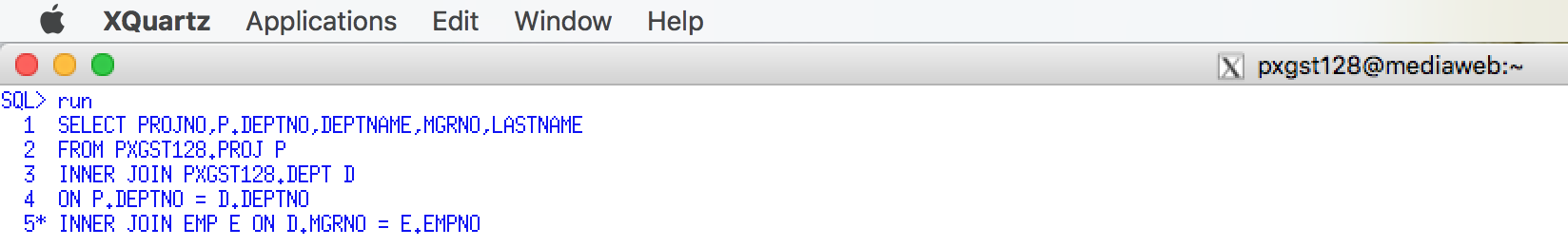


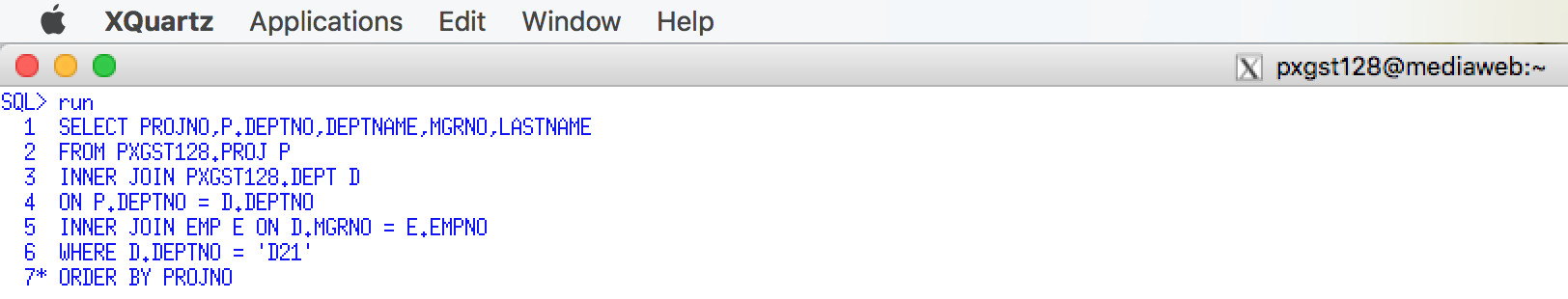




Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>







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

# 2.0 Unions, Intersections and Difference

The SQL UNION Operator - http://www.w3schools.com/sql/sql\_union.asp

Using the UNION keyword to combine subselects - <http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=%2Fsqlp%2Frbafykeyu.htm>

SQL: UNION Query - <http://www.techonthenet.com/sql/union.php>

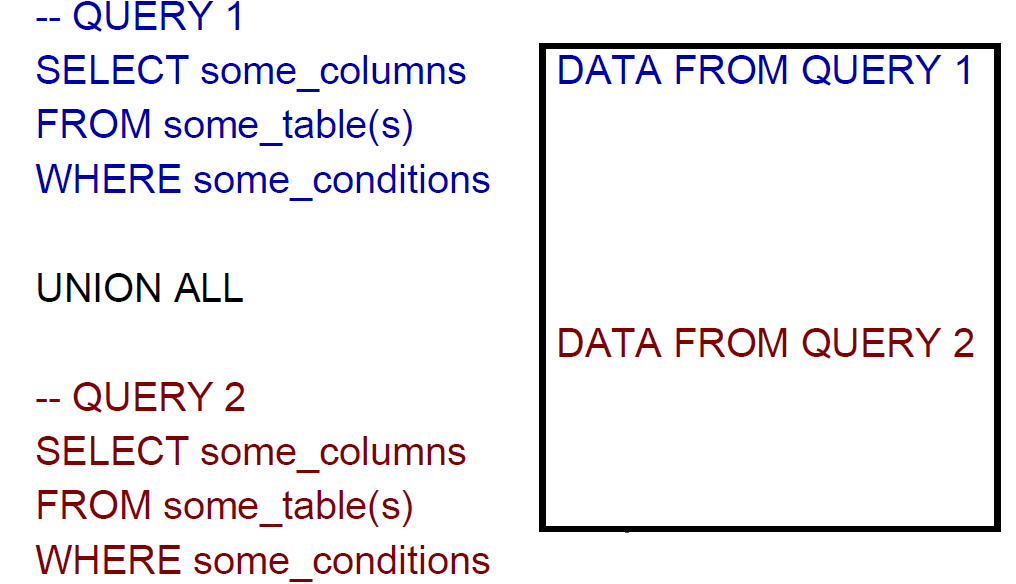
Specifying the UNION ALL keyword -

<http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=%2Fsqlp%2Frbafykeyuall.htm>

The UNION [ALL], INTERSECT, MINUS Operators - <https://docs.oracle.com/cd/B19306_01/server.102/b14200/queries004.htm>

A UNION, MINUS, or INTERSECT expression combines sets of columns into new sets of columns. When SQL encounters the UNION keyword, it processes each subselect to form an interim result table, then it combines the interim result table of each subselect and deletes duplicate rows to form a combined result table. You can use different clauses and techniques when coding select-statements.

**UNIONing Queries Together - Single Report**



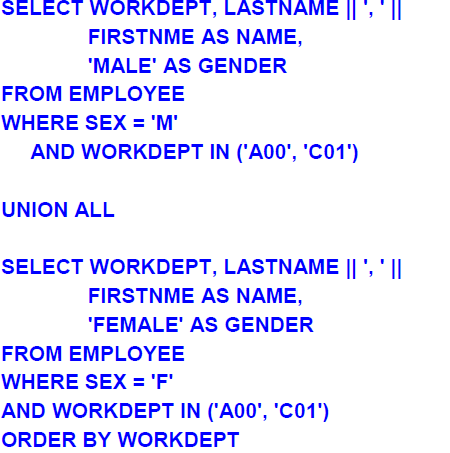
Each query must SELECT the same number of columns with the same data types.

## 2.1 Questions - UNION ALL

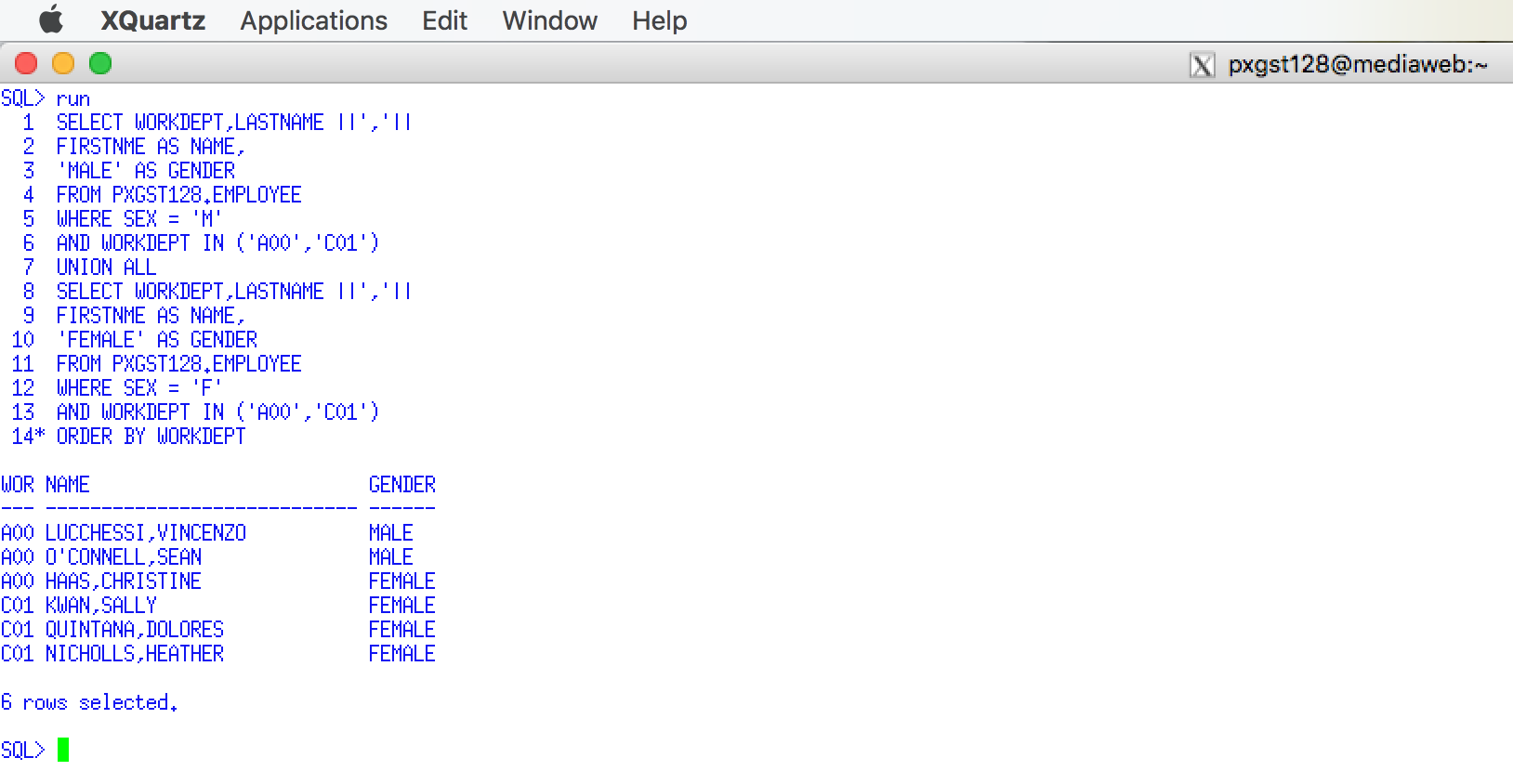
**1. UNION ALL - Example 1.** Display workdept, last name concatenated to first name and in a GENDER column print MALE or FEMALE where appropriate for workdepts C01 and A00. Since is a complicated SQL statement you should apply the "Keep it Simple" principle. First, code and test the first query. Second, copy, paste and modify the first query to the second query .Add the UNION all statement. And then add any other WHERE or ORDER BY statements, formatting or user-friendly column aliases. Summer 2018

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

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



Execute this example and display the result. Use a Snippit to document your SQL output =>



Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



Notes: In order for the UNION or UNION ALL clause the two different queries must be UNION compatible. This means that underlying SELECT must list the same number of columns, length and data types. Vertically through the selects the column's data types must be compatible. The data types don't have to be exactly alike, but must be in the same general category, that is, CHAR and VARCHAR, regardless of their lengths, are compatible; SMALLINT, INT, DECIMAL, FLOAT, and REAL are all compatible numeric data types.

For numeric columns the column's precision and scale will be that of the largest precision and scale of any of the numbers within the column, that is, if we had three queries in the stack and the third column of these queries had the following data types and lengths respectively: SMALLINT, DEC(7,2), DEC(23,3). The report column would have a data type of DEC(23,3).

The underlying tables and column names need not be the same, but the SELECT list must the previous rules.

Rules for using UNION - <http://pic.dhe.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.qmf9.doc.ref%2Fdsqi2dsq1018382.htm>

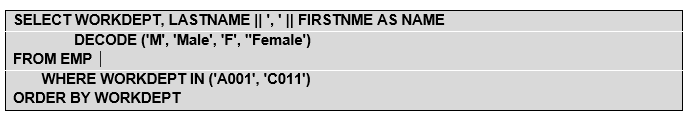
## 2.2 Questions - DECODE

2. In the previous example, the objective is to display MALE or FEMALE rather than M or F. The example is weak since one could have used the Oracle DECODE function instead.

DECODE - <http://pic.dhe.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.db2z10.doc.sqlref%2Fsrc%2Ftpc%2Fdb2z_bif_decode.htm>

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

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



Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

SELECT WORKDEPT,LASTNAME ||','|| FIRSTNME AS NAME

DECODE('M','MALE','F','FEMALE')

FROM PXGST128.EMP

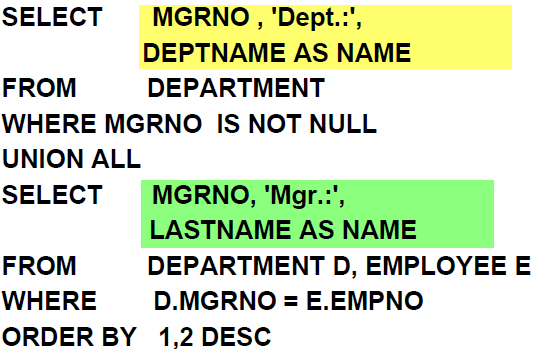
WHERE WORKDEPT IN('A001','C011')

ORDER BY WORKDEPT;

Note: The ORDER BY clause must be the last clause on the last query. This means that the data will be sorted AFTER the UNION operation.

If we want to ORDER BY a specific column (for example the second column) AND another column has the same column name in every SELECT query stack that the column name can appear in the ORDER BY clause. Otherwise, a relative position number must be used, e.g., ORDER by 2.

**3. UNION ALL - Example 2.** Use two different formatted lines to display department information. On line one print the manager's information, on line two print department information. Summer 2018

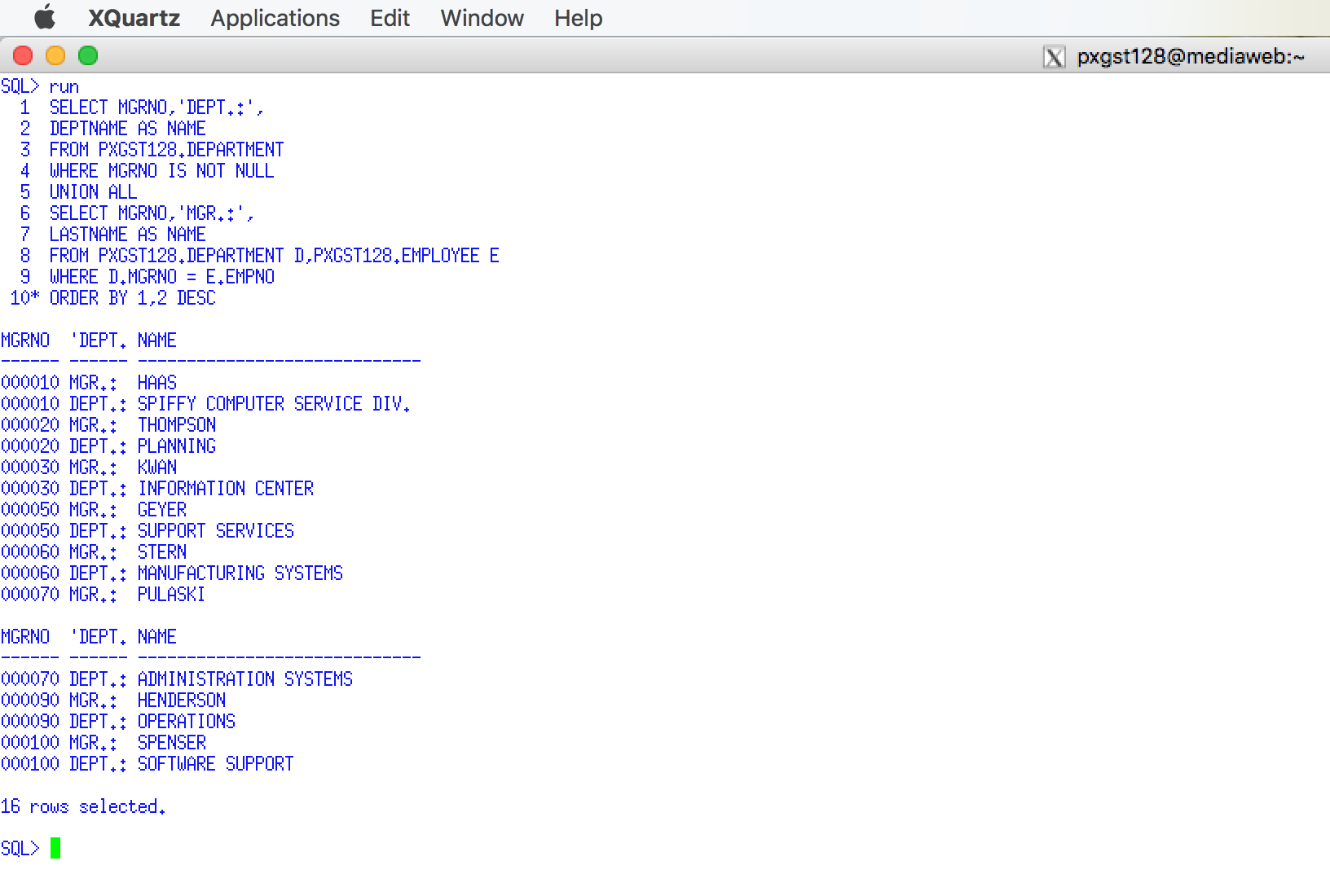


The first SELECT establishes the column headings of the result. If a column you wish to sort by has the same name in each query in the stack, then the column name can be used in the ORDER BY clause. Otherwise, the column's position number must be used.

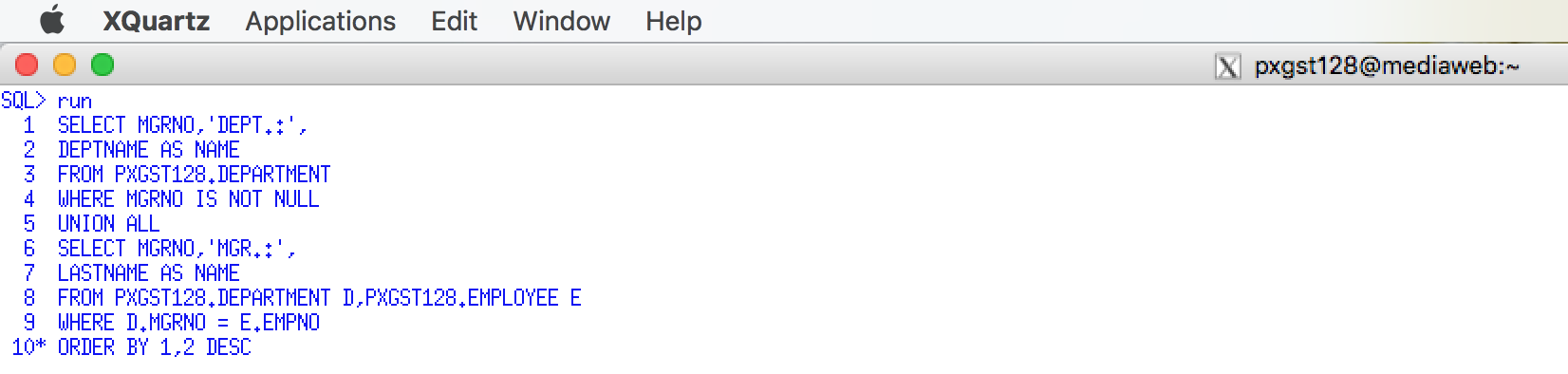
In this example, the position number (2) must be used for the second column in the ORDER BY clause because the column does not have a name in the result table. For this example, another valid ORDER BY clause may be ORDER BY MGRNO, 2 DESC. If we assign a common name to the second column in each of the SELECTs (for example, 'Dept.:' AS TEXT and 'Mgr.:' AS TEXT) we might write ORDER BY MGRNO, TEXT DESC.

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

Execute this example and display the result. Use a Snippit to document your SQL output =>



Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

,

**4. UNION- Example 3.** As per the presentation example use two lines to display department information. On line one print the manager's information, on line two print department information.

UNION combines two sets of rows and removes duplicates. Therefore, UNION ALL often performs better than UNION.

The problem to list last name and education level of employees who are analysts or have an education level of 18 can be solved by means of a UNION performing the following three steps:

1. Each SELECT is evaluated separately

2. The results are merged and ordered

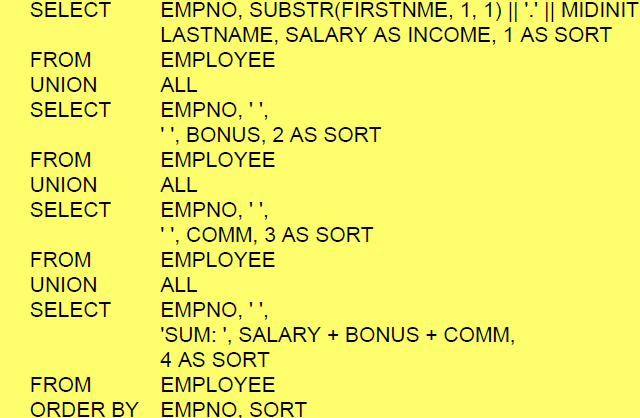
3. Duplicates are removed

Alternatively, the problem can be solved by means of a single SELECT:

SELECT LASTNAME, EDLEVEL FROM EMPLOYEE

WHERE JOB = 'ANALYST' OR EDLEVEL = 18

ORDER BY 1



Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

SELECT EMPNO,SUBSTR(FIRSTNME,1,1) ||'.'|| MIDINIT

LASTNAME,SALARY AS INCOME, 1 AS SORT

FROM PXGST128.EMPLOYEE

UNION ALL

SELECT EMPNO,'','',BONUS,2 AS SORT

FROM PXGST128.EMPLOYEE

UNION ALL

SELECT EMPNO,'','',COMM,3 AS SORT

FROM PXGST128.EMPLOYEE

UNION ALL

SELECT EMPNO,'','SUM:',SALARY + BONUS + COMM,4 AS SORT

FROM PXGST128.EMPLOYEE

ORDER BY EMPNO,SORT;

## 2.3 Questions - UNION versus a JOIN

When to use UNION versus when to join tables - http://pic.dhe.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.qmf9.doc.ref%2Fdsqi2dsq1018382.htm

5. What is the difference between a UNION and a JOIN? What are the advantages of each? Answer =>

Joins combine data into new columns. If two tables are joined together, then the data from the first table is shown in one set of column alongside the second table’s column in the same row.

Unions combines data into new rows. If two tables are “unioned” together, then the data from the first table is in one set of rows, and the data from the second table in another set. The rows are the same result.

The main advantage of a join is that it executes faster. The performance increase might not be noticeable by the end user.However because the columns are specifically named and indexed and optimized by the database engine, the retrival time almost always will be faster than that of a sub query.

The advantage of using Union is that they only select unique or distinct rows into the result set because duplicity is not allowed in Union.

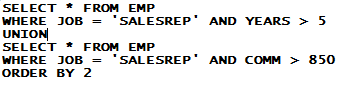
Summer 2018

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

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

**6. UNION implies that only DISTINCT rows** are selected from the columns named in both SELECT statements. If you want to keep duplicates in the result of a UNION operation, specify the optional keyword ALL after UNION. When UNION ALL is specified, duplicate rows are not eliminated from the result.

The following example selects all salespeople in EMP who have been employed for more than five years, or who earn a commission greater than $850. The salespeople who meet both conditions appear twice in the resulting report.



Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

SELECT \* FROM PXGST128.EMP

WHERE JOB = 'SALESREP' AND YEARS > 5

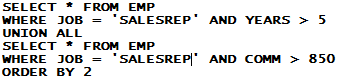
UNION

SELECT \* FROM PXGST128.EMP

WHERE JOB = 'SALESREP' AND COMM > 850

ORDER BY 2;

**7. Change the previous example of UNION ALL to UNION**



Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

SELECT \* FROM EMP

WHERE JOB = 'SALESREP' AND YEARS > 5

UNION ALL

SELECT \* FROM EMP

WHERE JOB = 'SALESREP' AND COMM > 850

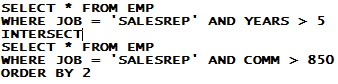
ORDER BY 2;

## 2.4 Questions - INTERSECT

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

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

**8. INTERSECT Operator Query**. The SQL INTERSECT query allows you to return the results of 2 or more "select" queries. However, it only returns the rows selected by all queries. If a record exists in one query and not in the other, it will be omitted from the INTERSECT results.



Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

SELECT \* FROM EMP

WHERE JOB = 'SALESREP' AND YEAR > 5

INTERSECT

SELECT \* FROM EMP

WHERE JOB = 'SALESREP' AND COMM > 850

ORDER BY 2;

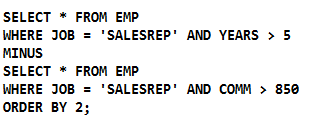
After reviewing the output of this example, explain the difference between the UNION operator in requirement 6 and INTERSECT operator in this example. ? Answer =>

## 2.5 Questions - MINUS

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

The UNION [ALL], INTERSECT, MINUS Operators - <https://docs.oracle.com/cd/B19306_01/server.102/b14200/queries004.htm>

**9. MINUS Operator Query -** An MINUS operation retrieves the set of distinct data values (not rows) that exist in the first the table but not in the second.



Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

SELECT \* FROM EMP

WHERE JOB = 'SALESREP' AND YEAR > 5

MINUS

SELECT \* FROM EMP

WHERE JOB = 'SALESREP' AND COMM > 850

ORDER BY 2;

After reviewing the output of this example, explain the difference between the INTERSECT operator in requirement 8 and MINUS operator in this example. Answer =>

# 3.0 Subqueries

While a table join combines multiple tables into a new table, a subquery (enclosed in parentheses) selects rows from one table based on values in another table. A subquery, or inner query, is a query-expression that is nested as part of another query-expression. Depending on the clause that contains it, a subquery can return a single value or multiple values. Subqueries are most often used in the WHERE and the HAVING expressions.

## 3.1 Single-Value Subqueries

A single-value subquery returns a single row and column. It can be used in a WHERE or HAVING clause with a comparison operator. The subquery must return only one value, or else the query fails and an error message is printed to the log.

This query uses a subquery in its WHERE clause to select U.S. states that have a population greater than Belgium. The subquery is evaluated first, and then it returns the population of Belgium to the outer query.

SELECT STATE.NAME, POPULATION

FROM US\_STATES

WHERE POPULATION >

(SELECT POPULATION FROM COUNTRIES

WHERE NAME = "BELGIUM");

Internally, this is what the query looks like after the subquery has executed:

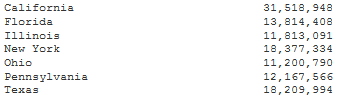
SELECT STATE.NAME, POPULATION

FROM US\_STATES

WHERE POPULATION > 101626614;

The outer query lists the states whose populations are greater than the population of Belgium.

Single-Value Subquery Results



## 3.2 Multiple-Value Subqueries

A multiple-value subquery can return more than one value from one column. It is used in a WHERE or HAVING expression that contains IN or a comparison operator that is modified by ANY or ALL. This example displays the populations of oil-producing countries. The subquery first returns all countries that are found in the OILPROD table. The outer query then matches countries in the COUNTRIES table to the results of the subquery.

SELECT COUNTRY, POPULATION

FROM .COUNTRIES

WHERE NAME IN

(SELECT COUNTRY FROM OILPROD\_COUNTRIES);



SELECT COUNTRY, POPULATION

FROM .COUNTRIES

WHERE NAME NOT IN

(SELECT COUNTRY FROM OILPROD\_COUNTRIES);



## 3.3 When to Use Joins and Subqueries

Use a join or a subquery any time that you reference information from multiple tables. Joins and subqueries are often used together in the same query. In many cases, you can solve a data retrieval problem by using a join, a subquery, or both. Here are some guidelines for using joins and queries.

* If your report needs data that is from more than one table, then you must perform a join. Whenever multiple tables (or views) are listed in the FROM clause, those tables become joined.
* If you need to combine related information from different rows within a table, then you can join the table with itself.
* Use subqueries when the result that you want requires more than one query and each subquery provides a subset of the table involved in the query.
* If a membership question is asked, then a subquery is usually used. If the query requires a NOT EXISTS condition, then you must use a subquery because NOT EXISTS operates only in a subquery; the same principle holds true for the EXISTS condition.

## 3.4 Questions – Subqueries

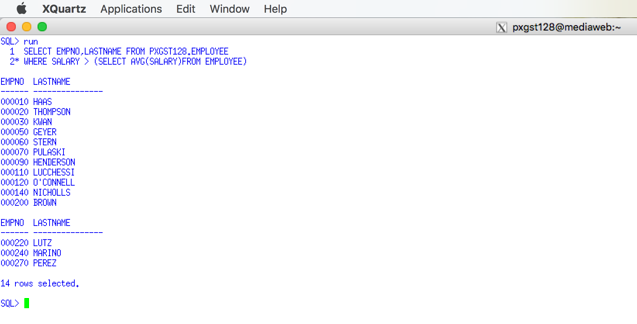
**Use a graphical snipping tool to document an image of these requirements below.**

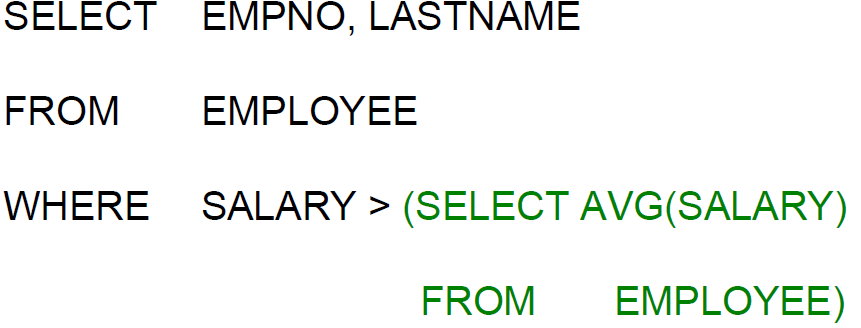
**Make sure your Putty banner is displayed.**

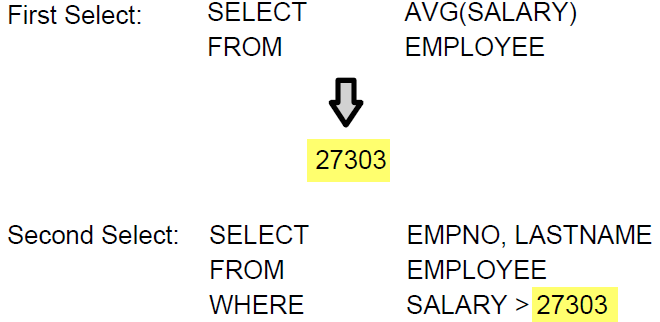
### 3.4.1 Question - Simple Subquery

1. Display the employee number and last name where the employee's salary is higher than the average salary.

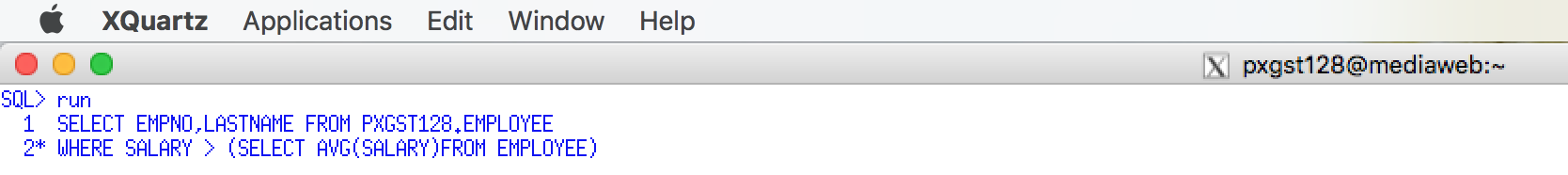
Execute this example and display the result. Use a Snippit to document your SQL output =>







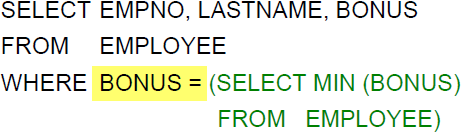
Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



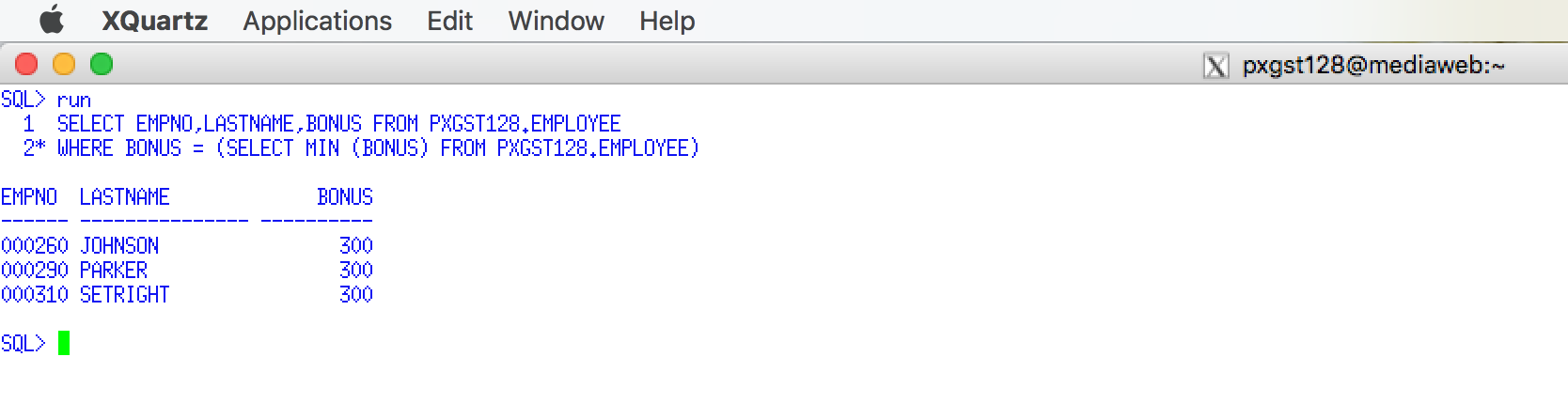
**Make sure your Putty banner is displayed.**

### 3.4.2. Questions - Subquery with Basic Predicates

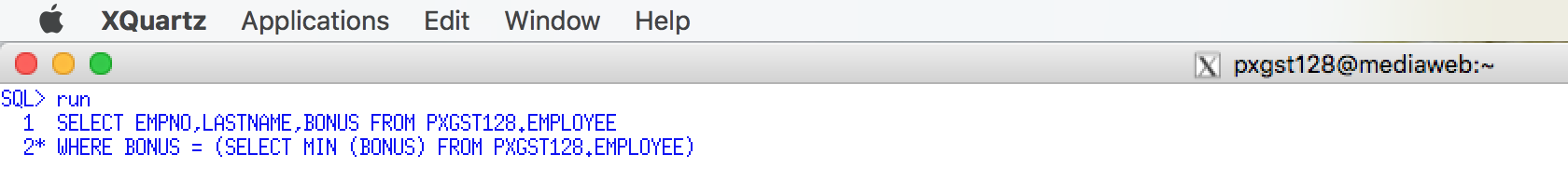
Display the employee number, last name, and bonus of employee(s) who made the lowest bonus.



Execute this example and display the result. Use a Snippit to document your SQL output =>

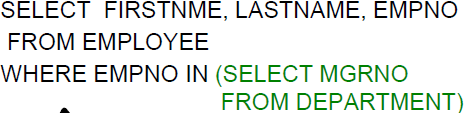


Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



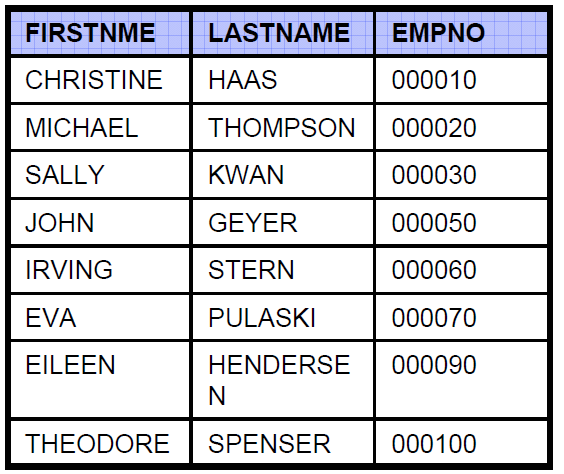
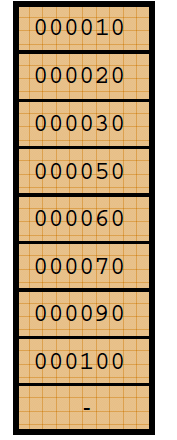
### 3.4.3 Questions - Subquery with IN Operator

3. Subquery using IN - Display the employee first name, last name, and employee number who are managers of a department.



Result of the Final results of the Outer Query

subquery



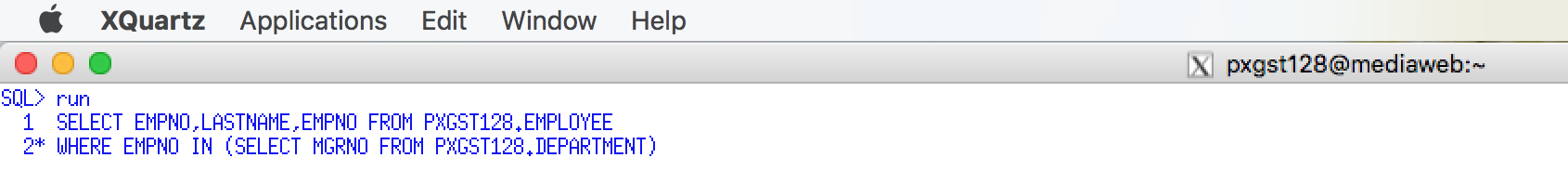
**Use a graphical snipping tool to document an image of these requirements below**

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

Execute this example and display the result. Use a Snippit to document your SQL output =>



Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



4. Subquery using IN - Display the employee first name, last name, work department and job for those employees who have the same job as SMITH. Summer 2018



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

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

Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

SELECT EMPNO,LASTNAME,WORKDEPT,JOB

FROM PXGST128.EMPLOYEE

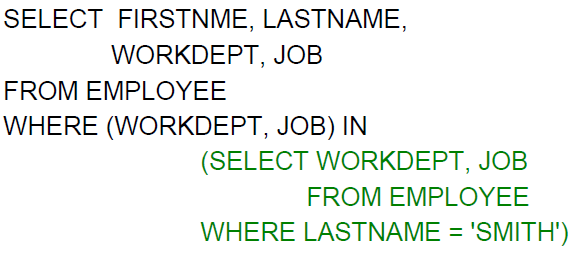
WHERE (WORKDEPT,JOB) = ('D21','CLERK');

### 3.4.4 Questions - Subquery using NOT IN

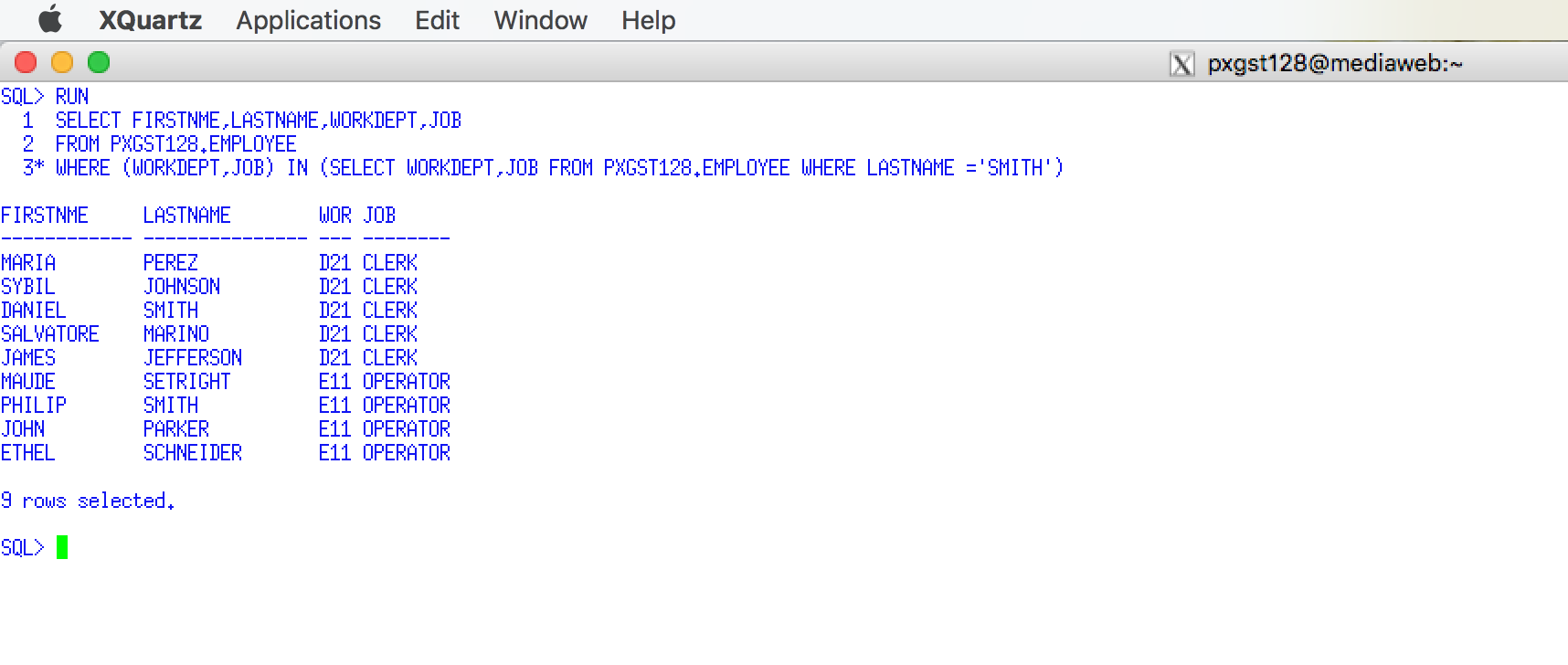
5. Display the department number and department name for those departments which do not have a project assigned to them. Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

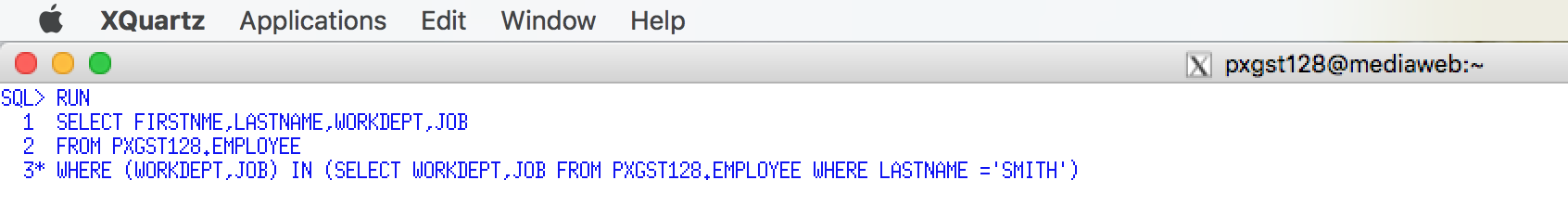
5. Subquery within a HAVING Clause - As per the presentation example display the department number and average employee salary for non-managers is higher than the company-wide average for non-managers. The department with the highest average should be listed first.



Execute this example and display the result. Use a Snippit to document your SQL output =>



Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>



## 3.5 NESTED QUERIES (SUBQUERIES) Background

**Video - SUB Queries in ORACLE 11g - https://www.youtube.com/watch?v=sgzZyhFMgag**

**Video - Oracle SQL Sub Queries Basic Video Tutorials.**

**https://www.youtube.com/watch?v=QE8N4jhh9is**

Suppose you wish to report any faculty member's name that is making more than the faculty average pay. You may try to code the following SQL statement:

SELECT FNAME FROM FACULTY

WHERE FPAY > AVG(FPAY);

While at first the above SQL statement appears to be logical, it includes a syntax error. To understand the nature of this syntax error it is important is to realize the fact that SQL must process table twice to perform the above request. The first execution will calculate the faculty average; the second execution will compare each individual's faculty pay to the average. To perform this and other similar tasks, one should code a nested query, or a subquery.

You can use a subquery to narrow a search condition that is based on information in an interim table. **You can specify a subquery in either a WHERE clause or a HAVING clause.**

**Example 52** **SELECT FNAME, FDEP, FPAY FROM FACULTY**

**WHERE FPAY >**

**(SELECT AVG(FPAY) FROM FACULTY);**

Report those faculty members who are making more than the average pay. The last (or inner) SELECT, i.e., (SELECT AVG(FPAY) FROM FACULTY) statement is executed first. An average is calculated and the result is stored in a temporary table. Then the first (outer) SELECT statement is executed comparing individual faculty's pay to the previously calculated average.

The inner select is executed and gets the result of an average of $63500. Then the outer select statement is executed.

**SELECT FNAME, FDEP, FPAY FROM FACULTY**

**WHERE FPAY >**

**(635000)**

**Example 53.** **SELECT FNAME FROM FACULTY**

**WHERE FID IN**

**(SELECT DISTINCT FID FROM COURSE\_SCHEDULE);**

Report those faculty names that are teaching this semester. The inner SELECT process all records in the COURSE\_SCHEDULE table and will create a file containing only the FID found in the COURSE\_SCHEDULE table. Then SQL will process all of the records in the FACULTY table and compare the FIDs to the found list.

The inner select is executed and gets the result contains a set of all faculty distinct FIDs who are on the course schedule.

**SELECT FNAME FROM FACULTY**

**WHERE FID IN**

**(22, 45, 666, 9810);**

**Example 54**  **SELECT FNAME FROM FACULTY**

**WHERE FPAY >**

**(SELECT FPAY FROM FACULTY WHERE FNAME = ‘PACKY’);**

Report those faculty names who are earning a salary greater than Packy's salary. {Special note: Be sure that you have enough disk space to store all of the names}

**Example 55**  **SELECT FNAME FROM FACULTY**

**WHERE FDEP =**

**(SELECT FDEP FROM FACULTY WHERE FNAME = ‘PACKY’);**

Report those faculty names who are working in the same department as PACKY.

**Other Subquery Examples**

**Example 56:**  **SELECT JOB, MAX(SALARY), MIN(SALARY)**

**FROM EMP**

**GROUP BY JOB**

**HAVING COUNT(\*) > 1 AND MAX(SALARY) > 50000;**

Show the job code, maximum salary, and minimum salary for each group of rows of EMP with the same job code, but only for groups with more than one row and with a maximum salary greater than 50000.

**Example 57: SELECT EMPNO, ACTNO, CHAR(EMSTDATE,USA), CHAR(EMENDATE,USA)**

**FROM EMPPROJACT**

**WHERE EMPNO IN**

**(SELECT EMPNO FROM EMP**

**WHERE WORKDEPT = 'E11');**

For each employee in department E11, get the following information from the table EMPPROJACT: display the employee number, activity number, activity start date, and activity end date. Using the CHAR function, convert the start and end dates to their USA formats. Get the needed department information from the table EMP.

**Example 58: SELECT WORKDEPT, MAX(SALARY)**

**FROM EMP**

**GROUP BY WORKDEPT**

**HAVING MAX(SALARY) <**

**(SELECT AVG(SALARY)**

**FROM EMP);**

Show the department number and maximum departmental salary for all departments whose maximum salary is less than the average salary for all employees. (In this example, the subquery would be executed only one time.)

**Example 59 SELECT WORKDEPT, MAX(SALARY)**

**FROM EMP Q**

**GROUP BY WORKDEPT**

**HAVING MAX(SALARY) <**

**(SELECT AVG(SALARY)**

**FROM EMP**

**WHERE NOT WORKDEPT = Q.WORKDEPT);**

A correlated subquery is a subquery that Oracle reevaluates when it examines a new row (in a WHERE clause) or a group of rows (in a HAVING clause) as it executes the outer SELECT statement. Show the department number and maximum departmental salary for all departments whose maximum salary is less than the average salary for employees in all other departments. (In contrast to Example 59 the subquery in this statement, containing a correlated reference, would need to be executed for each group.)

**Example 60 SELECT EMPNO, LASTNAME, WORKDEPT, EDLEVEL**

**FROM EMP X**

**WHERE EDLEVEL >**

**(SELECT AVG(EDLEVEL)**

**FROM EMP**

**WHERE WORKDEPT = X.WORKDEPT);**

Suppose that you want a list of all the employees whose education levels are higher than the average education levels in their respective departments. To get this information, ORACLE must search the EMP table. For each employee in the table, ORACLE needs to compare the employee's education level to the average education level for that employee's department.

For this example, you need to use a correlated subquery, which differs from an uncorrelated subquery. An uncorrelated subquery compares the employee's education level to the average of the entire company, which requires looking at the entire table. A correlated subquery evaluates only the department that corresponds to the particular employee.

In the subquery, you tell ORACLE to compute the average education level for the department number in the current row. The following query performs this action:

A correlated subquery looks like an uncorrelated one, except for the presence of one or more correlated references. In the example, the single correlated reference is the occurrence of X.WORKDEPT in the WHERE clause of the subselect. In this clause, the qualifier X is the correlation name that is defined in the FROM clause of the outer SELECT statement. X designates rows of the first instance of EMP. At any time during the execution of the query, X designates the row of EMP to which the WHERE clause is being applied.

Consider what happens when the subquery executes for a given row of EMP. Before it executes, X.WORKDEPT receives the value of the WORKDEPT column for that row. Suppose, for example, that the row is for Christine Haas. Her work department is A00, which is the value of WORKDEPT for that row. Therefore, the following is the subquery that is executed for that row:

(**SELECT AVG(EDLEVEL)**

**FROM EMP**

**WHERE WORKDEPT = 'A00');**

The subquery produces the average education level of Christine's department. The outer SELECT then compares this average to Christine's own education level. For some other row for which WORKDEPT has a different value than the value appears in the subquery in place of A00. For example, in the row for Michael L Thompson, this value is B01, and the subquery for his row delivers the average education level for department B01.

The result table that is produced by the query is similar to the following output:

EMPNO LASTNAME WORKDEPT EDLEVEL

====== ========= ======== =======

000010 HASS A00 18

000030 KWAN C01 20

000070 PULASKI D21 16

000090 HENDERSON E11 16

**Example 61 SELECT HIREYEAR, AVG(SALARY)**

**FROM**

**(SELECT YEAR(HIREDATE) AS HIREYEAR, SALARY**

**FROM DSN8A10.EMP) AS NEWEMP**

**GROUP BY HIREYEAR;**

For each group of employees hired during the same year, show the year-of-hire and current average salary. (This example demonstrates how to use the AS clause in a FROM clause to name a derived column that you want to refer to in a GROUP BY clause.)

## 3.6 Questions - Subquery Review

1. Explain the process how a Subquery executes, e.g., inner and outer queries? Answer =>

2. Given this following SQL Statement SELECT FNAME FROM FACULTY WHERE FPAY > AVG(FPAY). Why is this statement illegal? Answer =>

3. Given this following SQL Statement SELECT FNAME FROM FACULTY WHERE FPAY > AVG(FPAY). How would you use a Subquery to get the information results? Answer =>

4. Why is the IN or NOT IN operators popular in Subqueries? Answer =>

# 4.0 Questions - Unions and Subqueries

## 4.1 Questions – Customized Subqueries

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

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

1. List those employees that have a salary which is greater than or equal to the average salary of all employees plus $5,000. Display department number, employee number, last name, and salary. Sort the list by the department number and employee number.

Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

2,.List employee number and last name of all employees not assigned to any projects. This means that table EMP\_ACT does not contain a row with their employee number.

Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

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

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

3. List project number and duration (in days) of the project with the shortest duration.

Name the derived column DAYS. (Hint: The inner Select is SELECT MIN(DAYS(PRENDATE)-DAYS(PRSTDATE). Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

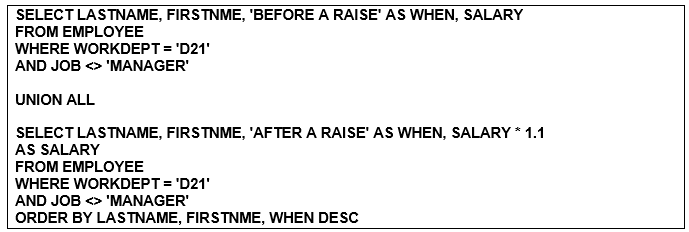
4. List department number, department name, last name, and first name of all those employees in departments that have only male employees. (Hint: The inner Select should list department numbers which have for female employees. Then the outer Select would search for those departments NOT IN that set.).

Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

## 4.1 Questions – Customized Unions

5. List the names and salaries for the non-managers working in department D21 showing the effects of a 10 percent raise. Use the following output as a guide. Apply and appropriate ORDER BY clause to achieve the required results.



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

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

Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

6. List the department number, employee number, and salaries of all employees in department A00. For the last line of the report, display the sum of all the salaries.

**SELECT WORKDEPT, EMPNO, SALARY**

**FROM EMPLOYEE**

**WHERE WORKDEPT = 'A00'**

**UNION ALL**

**SELECT WORKDEPT, 'SUM', SUM(SALARY)**

**FROM EMPLOYEE**

**WHERE WORKDEPT = 'A00'**

**GROUP BY WORKDEPT**

**ORDER BY 3**

Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

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

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

7. For all departments, display department number and the sum of all salaries for each department. Name the derived column SUM\_SALARY. Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snippit to document your SQL statement =>

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

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

8. For all departments, display the department number and the number of employees. Name the derived column EMP\_COUNT. Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

9. Display those departments which have more than 3 employees. Name the derived column EMP\_COUNT. Execute this example and display the result. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

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

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

10. Display the average salary for men and the average salary for women for each department. Display the work department, the sex, the average salary, average bonus, average commission, and the number of people in each group. Include only those groups that have two or more people. Show only two decimal places in the averages. Use the following names for the derived columns: AVG\_SALARY, AVG\_BONUS, AVG\_COMM, and COUNT. Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

# 5.0 PSP Join Questions

**Videos**

**Video - Introduction to Intermediate SQL -**

**http://www.youtube.com/watch?v=QOoA5ucZasE&feature=relmfu**

**Video - Introduction to Advanced SQL -**

**http://www.youtube.com/watch?v=JhdsY32zqo0&feature=relmfu**

**Video - Joining Two Tables**

**http://www.youtube.com/watch?v=DwhQl2\_8slA&feature=relmfu**

**Video - SQL: Understanding the JOIN clause in the SELECT statement**

**http://www.youtube.com/watch?v=M3Dj6UWDj-4&feature=channel&list=UL**

**Video - SQL: Understanding the JOIN clause in the SELECT statement**

**http://www.youtube.com/watch?v=M3Dj6UWDj-4&feature=BFa&list=ULM3Dj6UWDj-4**

**Video - Union Statement**

**http://www.youtube.com/watch?v=gdtRK\_jCDH8&feature=related**

## 5.1 Apply an Inner Join between Customer and Drawing tables

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

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

Use a **simple inner join** to display the Drawing, Number Account Number, Customer Name, Customer Part Number, and GL Account Number ***ordered by drawing number***.

Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

## 5.2 Apply an Inner Join between Customer and Packing List tables

Use a **simple inner join** to display the Date Planned Shipped, Packing List Number, Account Number, Customer Name, Customer Part Number, and qty to be shipped in plan shipped date order. Within a ship date, the packing list should be ***ordered by customer type***.

Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

## 5.3 Apply an Inner Join between Customer and Packing List tables

Use a **simple inner join** to display the Date Planned Shipped, Packing List Number, Account Number, Customer Name, Customer Part Number, and qty to be shipped in plan shipped date order ***only for blanket customers***. Your must have stored one or more blanket customer types. If not add it now with an Insert statement.

Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

## 5.4 Apply an Inner Join between Customer and Packing List tables

Use a **simple inner join** display the Date Planned Shipped, Packing List Number, Account Number, Customer Name, Customer Part Number, and qty to be shipped listed in ***descending by plan shipped date order***.

**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 =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

## 5.5 Apply an Inner Join between Customer, Packing List and Job Cost tables

Using an **inner join** between the customer, packing list and job cost table display the reference number, transaction date, packing list number, drawing number, customer name, transaction type and extended cost (unit cost times transaction units) ***in reference number order.***

Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

## 5.6 Apply an Inner Join between Customer, Packing List and Job Cost tables

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

Using an **inner join** between the customer, packing list and job cost table display the reference number, transaction date, packing list number, drawing number, customer name, transaction type and extended cost (unit cost times transaction units) in reference number order ***only for blanket customers.***

Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

## 5.7 Apply an Inner Join between Packing List and Job Cost (decode)

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

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

Using an **inner join** between the packing list and job cost table display the Packing list number, drawing number, transaction date, transaction description based upon transaction type and extended cost (unit cost times transaction units) in packing list, transaction date order. Using the **decode function** replace the transaction type with a descriptive statement, e.g., L means direct labor. You must have stored multiple Packing Lists which have multiple direct labor costs, e.g., L

Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

## 5.8 Apply an Inner Join between Job Ccost and Labor Operation

Using a join between the job cost table and labor operation table display the Packing list number, transaction date, transaction info, labor operation description and extended cost (unit cost times transaction units) for only labor job costs in packing list, transaction date order. You might have to use the RTRIM function to join the transaction info with the operation code in the labor operation table.

Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>

## 5.9 Apply Inner Join between Job Cost, Time Clock, and Employee

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

Using an inner join between the job cost table, time clock and employee table display the Packing list number, transaction date, time clock key, employee number, employee name and transaction units for only labor job costs in packing list, transaction date order.

Use a Snippit to document your SQL output =>

Use a Snipping tool document your SQL code, which also displays your schema name for each table name =>