# --Group#7-- 75/80

# --Adam Springett 130751209 aspringett@myseneca.ca--

# --Yihsun Lee 129713202 ylee212@myseneca.ca--

# --Sanghwa Suh 100711209 ssuh6@myseneca.ca--

# --Nozomi Tsuchiya 154666192 ntsuchiya1@myseneca.ca--

# Assignment 1, Section B Worth 20% Due by Friday, June 25th by 9pm

## Submission

***Your submission will be a Document file containing all 10 Select statements Plus their Outputs. Also you will attach a single text-based SQL file (.sql) with appropriate header and commenting.*** *Please ensure your file runs when the entire file is executed in SQL Developer or SQL Plus.*

Save these files as **Asgn1\_GrX.docx** and **Asgn1\_GrX.sql**, here is X your Group Number.  
**Only one e-mail submission per group please.**

## Submission Checklist Use the following checklist, to make sure you have completed the assignment successfully.

|  |  |  |
| --- | --- | --- |
| **Tasks to be completed** | **Yes** | **No** |
| You have read the assignment group submission and completion policies and all instructions provided in the assignment document and have not missed a word. | X |  |
| Student information and the assignment information have been added to the header of the submission. (Same as the template provided in the assignment documents) | X |  |
| All questions are answered in a text file and are saved as a ***.sql*** file. | X |  |
| Comments are included. (questions definition or any additional explanation) | X |  |
| All SQL statements are executed successfully without errors. (Use "Run Script" or @scriptname to execute all statements together.) | X |  |
| The result of your SQL statements contains the given sample row and the header in the assignment document. | X |  |

## Group Work

This assignment is to be completed in groups of 3 or 4. Please only one submission per group. The comment header MUST have all students’ name and student number.

It is suggested that you **ALL do it individually** and then meet to compare answers. Those not doing the work may be barred from your group resulting in a zero and incomplete on the assignment.

## Tasks

For each question, the Header and the First Sample Row (or Rows) must match the sample row or rows given in that question. If you are using SQL Plus, then you may format the width of certain items (columns) output with SUBSTR(item,1,length)

1. Display the employee number, full employee name, job title, and hire date of all employees hired outside the Fall months (Sep, Oct, Nov), including only employees with ‘A’ jobs (their job title starts with **A**), but that title can not exceed 20 characters. Sort the output so that the most recently hired employees are displayed first.

Here is shown the exact Output with the first TWO rows. Your code must accomplish the Exact look of the Output.

**Emp# Full Name Job Start Date ----------------------------------------------------------------------**

**14 Elliot Brooks Accountant Seventh of December, 2016**

**10 Ryan Gray Accountant Sixteenth of August, 2016**

**SELECT employee\_id AS "Emp#",**

**first\_name || ' ' || last\_name AS "Full Name",**

**job\_title AS "Job",**

**TO\_CHAR(hire\_date, 'fmDdspth "of" Month"," YYYY') AS "Start Date"**

**FROM employees**

**WHERE NOT(hire\_date BETWEEN '01-SEP-16' AND '30-NOV-16')**

**AND LENGTH(job\_title) <= 20 AND**

**UPPER(job\_title) LIKE 'A%'**

**ORDER BY hire\_date DESC;**

**-- Months condition should be valid for Any Year, you should use TO\_CHAR with MM option**

**Graphical user interface, text, application

Description automatically generated**

1. The company wants to see the total sale amount per sales person (salesman) for all orders. Display the salesman full name and the total sale amount for that employee.

Exclude sales persons whose first name end on **y** (like Lily or Daisy).

Sort the result so that bigger sales are shown first.

Note; Do NOT use LIKE operator.

Here is shown the exact Output with the first TWO rows. Your code must accomplish the Exact look of the Output.

**Emp Name Total Sale**

**------------------------- ----------------**

**Gomez, Freya $8,081,332.30**

**Freeman, Florence $4,341,842.14**

**SELECT last\_name || ', ' || first\_name AS "Emp Name",**

**TO\_CHAR(SUM(unit\_price \* quantity),CONCAT('$','999,999,999.99')) AS "Total Sale"**

**FROM employees e**

**INNER JOIN orders o**

**ON e.employee\_id = o.salesman\_id**

**INNER JOIN order\_items oi**

**ON o.order\_id = oi.order\_id**

**WHERE SUBSTR(first\_name, -1, 1) != 'y'**

**GROUP BY last\_name || ', ' || first\_name**

**ORDER BY 2 DESC;**

**Graphical user interface, application

Description automatically generated with medium confidence**

1. Variation of Question 2.

The company wants to see the total sale amount per sales person (salesman) for all orders. Assume that some online orders do not have any sales person involved. Display the salesman full name and the total sale amount for that employee.

Include also the Total Sale amount for all Orders without involvement of a salesman.

Exclude sales persons whose first name end on **y** (like Lily or Daisy).

Include only those sales persons who were responsible for more than 4 million dollars in sale.

Sort the result so that bigger sales are shown first.

Note; Do NOT use LIKE operator.

Here is shown the exact Output with the first TWO rows. Your code must accomplish the Exact look of the Output.

**Emp Name Total Sale**

**------------------------- ----------------**

**, $18,245,463.50**

**Gomez, Freya $8,081,332.30**

**SELECT TO\_CHAR(',') AS "Emp Name",**

**TO\_CHAR(SUM(unit\_price \* quantity),CONCAT('$','999,999,999.99')) AS "Total Sale"**

**FROM order\_items oi**

**JOIN orders o**

**ON oi.order\_id = o.order\_id**

**WHERE o.salesman\_id IS NULL**

**UNION**

**SELECT last\_name || ', ' || first\_name AS "Emp Name",**

**TO\_CHAR(SUM(unit\_price \* quantity),CONCAT('$','999,999,999.99')) AS "Total Sale"**

**FROM employees e**

**JOIN orders o**

**ON e.employee\_id = o.salesman\_id**

**INNER JOIN order\_items oi**

**ON o.order\_id = oi.order\_id**

**WHERE SUBSTR(first\_name, -1, 1) != 'y'**

**GROUP BY last\_name || ', ' || first\_name**

**HAVING SUM(unit\_price \* quantity) > 4000000**

**ORDER BY 2 DESC;**

**Graphical user interface, text, application

Description automatically generated**

1. Display number of orders for customers with name starting on **B** or **L,** but only if their total number of orders is less than 2. Include the customers with no orders in your report as well.

The report should show only customers who contain **l** in their name (if their name starts on **B**) and also customers who contain **b** in their name (if their name starts on **L**). Exclude Banks from your report (like Bank of New York Mellon).

Sort the result by the value of total orders ascending, followed by name ascending.

Note; Do NOT use LIKE operator.

Here is shown the exact Output with the first TWO rows. Your code must accomplish the Exact look of the Output.

**CustId Name Total Orders**

**---------- ------------------------------ ------------**

**116 Baxter International 0**

**289 BlackRock 0**

**SELECT c.customer\_id AS "CustID", c.name AS "Name", COUNT(o.customer\_id) AS "Total Orders"**

**FROM customers c**

**FULL OUTER JOIN orders o**

**ON c.customer\_id = o.customer\_id**

**WHERE INSTR(UPPER(name), 'BANK')= 0**

**AND (SUBSTR(UPPER(name), 1, 1) = 'B'**

**AND INSTR(UPPER(name), 'L') >1)**

**OR (SUBSTR(UPPER(name), 1, 1) = 'L'**

**AND INSTR(UPPER(name), 'B') >1)**

**GROUP BY c.customer\_id, c.name**

**HAVING COUNT(o.customer\_id) < 2**

**ORDER BY 3, 2;**

**-- Full Join is Not needed, just Left Join**

**Text

Description automatically generated**

* 1. Display customer number, customer name, and the order id and the order date of all orders placed by contact persons from Area Phone Code **319** (Eastern Iowa in US). Be careful to avoid phone numbers having these digits somewhere else (like +1 212 319 4567). Show also the total number of items ordered, the total amount of each customer’s order and last name of the contact person.
  2. Exclude Orders with the Total Amount exceeding 800 thousand dollars
  3. Exclude orders placed by Contact person **Norris.**
  4. Sort the result from the highest to lowest total order amount.

Note: use Multiple Tables JOIN method with USING clause

Here is shown the exact Output with the FIRST row only. Your code must accomplish the Exact look of the Output.

**Cust# Customer Name Order# Order Date Total# Total Amount LName**

**----- ---------------- ------ ---------- ------ ------------- -------**

**42 Becton Dickinson 25 24-AUG-16 700 $484,279.39 Jarvis**

SELECT customer\_id AS "Cust#",

name AS "Customer Name",

order\_id AS "Order#",

order\_date AS "Order Date",

SUM(quantity) AS "Total#",

TO\_CHAR(SUM(unit\_price \* quantity),CONCAT('$','999,999,999.99')) AS "Total Amount",

last\_name AS "LName"

FROM customers

JOIN orders

USING(customer\_id)

JOIN contacts

USING(customer\_id)

JOIN order\_items

USING (order\_id)

WHERE INSTR(phone, '319') = 4

AND UPPER(last\_name) != 'NORRIS'

HAVING SUM(unit\_price \* quantity) < 800000

GROUP BY customer\_id, name, order\_id, order\_date, last\_name

ORDER BY 7; Graphical user interface, text, application, email

Description automatically generated

1. Display Warehouse Id, warehouse name, product category Id, product category name, and the lowest product list price for this combination.

* In your result, include the rows that the lowest list price is less then $50.
* Also, include the rows that the lowest cost is more than $200.
* Observe only warehouses that are located in Countries that start on **C**
* Sort the output according to Warehouse Id and then product category Id

Here is shown the exact Output with the first TWO rows. Your code must accomplish the Exact look of the Output.

**Wrhs# Warehouse Name Category# Category Name Lowest Price**

**----- --------------- --------- -------------- ------------**

**5 Toronto 1 CPU $554.99**

**5 Toronto 2 Video Card $739.99**

**SELECT w.warehouse\_id "Wrhs#",**

**w.warehouse\_name "Warehouse Name",**

**pc.category\_id "Category#",**

**pc.category\_name "Category Name",**

**TO\_CHAR(MIN(p.list\_price), CONCAT('$', '999.99')) AS "Lowest Price"**

**FROM product\_categories pc**

**JOIN products p ON pc.category\_id = p.category\_id -- USING(category\_id)**

**JOIN inventories i ON p.product\_id = i.product\_id --USING(product\_id)**

**JOIN warehouses w ON i.warehouse\_id = w.warehouse\_id -- USING(warehouse\_id)**

**JOIN locations l ON w.location\_id = l.location\_id -- USING(location\_id)**

**JOIN countries c ON l.country\_id = c.country\_id -- USING(country\_id)**

**WHERE c.country\_name LIKE 'C%'**

**GROUP BY w.warehouse\_id, w.warehouse\_name, pc.category\_id, pc.category\_name**

**HAVING MIN(p.list\_price) < 50**

**OR MIN(p.standard\_cost) > 200**

**ORDER BY 1,3;**

**Text

Description automatically generated with low confidence**

1. Display product Id, category Id, product Name, and list Price for products that were purchased in orders handled by salesmen whose last name begins on **E** or **F,** but only for product category that starts on **Video**. Sort the output by product Id ascending.

Note; You must NOT use JOIN method, just use SubQueries. Also you must use LIKE operator (not SUBSTR or INSTR).

Here is shown the exact Output with the first THREE rows. Your code must accomplish the Exact look of the Output.

**ProdId Category# Product Name Lprice**

**---------- ----------- -------------------- ---------**

**3 2 Corsair CB-9060011-W 799.99**

**5 2 PNY VCQK6000-PB 2290.79**

**6 2 Zotac ZT-P10810A-10P 849.99**

**SELECT product\_id "ProdId",**

**category\_id "Category#",**

**product\_name "Product Name",**

**list\_price "Lprice"**

**FROM products**

**WHERE category\_id IN (SELECT category\_id**

**FROM product\_categories**

**WHERE UPPER(category\_name) LIKE 'VIDEO%')**

**AND product\_id IN (SELECT product\_id**

**FROM order\_items**

**WHERE order\_id IN (SELECT order\_id**

**FROM orders**

**WHERE salesman\_id IN (SELECT employee\_id**

**FROM employees**

**WHERE UPPER(last\_name) LIKE 'E%'**

**OR UPPER(last\_name) LIKE 'F%')))**

**ORDER BY 1;**

**Text, application

Description automatically generated**

**Text

Description automatically generated**

1. Variation of Q7

Display product Id, product Name, and list Price for products that were purchased in orders handled by salesmen whose last name begins on **E** or **F** and who were hired in **December** of any year**,** but only for product category that starts on **Video**. Include only products with list Price smaller than Any Average Price per product Category. Sort the output by product Id ascending.

Note; You must NOT use JOIN method, just SubQueries. Also,

you must NOT use LIKE operator.

Here is shown the exact Output with the first TWO rows. Your code must accomplish the Exact look of the Output.

**ProdId Product name Lprice**

**---------- -------------------- ----------**

**48 AMD FirePro S7000 1218.5**

**89 PNY VCGGTX780T3XPB-O 749.99**

SELECT

product\_id AS "ProdID",

product\_name AS "Product Name",

list\_price AS "Lprice"

From products

WHERE product\_id IN (SELECT product\_id

FROM order\_items

WHERE order\_id IN ( SELECT order\_id

FROM orders

WHERE salesman\_id IN (SELECT employee\_id

FROM employees

WHERE upper(last\_name) LIKE 'E%'

OR upper(last\_name) LIKE 'F%'

AND TO\_CHAR(hire\_date, 'MM') = '12')))

AND category\_id IN (SELECT category\_id

FROM product\_categories

WHERE upper(category\_name) LIKE 'VIDEO%')

AND list\_price < ANY (SELECT AVG(list\_price)

FROM products

GROUP BY category\_id)

ORDER BY 1;

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**-- should Not use Like, use SUBSTR(last\_name,1,1)**

1. Display product Id, product Name, and list Price for products that their list price is less than Any lowest product standard cost per warehouse inside regions of **Europe** or **Asia.**

(You need to find the lowest standard cost for each warehouse that is located inside these two regions. Then you need to return all products that their list price is less than any lowest standard cost of those warehouses.).

Include only products with Quantity greater than Any highest quantity per warehouse (anywhere).

Sort the result according to list price descending.

Note; You should use both methods here, Subquery first, then followed by JOIN/USING method.

Here is shown the exact Output with the first THREE rows. Your code must accomplish the Exact look of the Output.

**Product ID Product Name List Price**

**---------- -------------------- ------------**

**227 Gigabyte GA-X99-UD5 $305.00**

**260 Crucial CT1050MX300S $267.99**

**286 Samsung MZ-V6E500 $234.00**

SELECT p.product\_id "Product ID",

p.product\_name "Product Name",

TO\_CHAR(p.list\_price, '$999.99') "List Price"

FROM products p JOIN inventories i ON p.product\_id = i.product\_id

WHERE p.list\_price < ANY (SELECT list\_price

FROM products

WHERE p.list\_price < ANY (SELECT MIN(standard\_cost)

FROM products pr

JOIN inventories i USING(product\_id)

JOIN warehouses w ON i.warehouse\_id = w.warehouse\_id

JOIN locations l USING(location\_id)

JOIN countries c USING(country\_id)

JOIN regions r USING(region\_id)

WHERE UPPER(r.region\_name) IN ('EUROPE', 'ASIA')

GROUP BY w.warehouse\_id)

)

AND i.quantity > ANY (SELECT MAX(quantity)

FROM inventories

WHERE warehouse\_id IN (SELECT warehouse\_id

FROM warehouses)

GROUP BY warehouse\_id)

GROUP BY p.product\_id, p.product\_name, p.list\_price

ORDER BY 3 DESC;

Graphical user interface, text, application

Description automatically generated with medium confidence

**-- should Not Join upfront, just a Main Query**

**-- last SQ should be over product\_id , and the nested SQ over warehouse\_id goes out**

Text

Description automatically generated