--===============================================================================  
--=================================**Exercise 7** =====================================

--=============================**Individual Only** =====================================  
--===============================================================================

**\*\* In this exercise, you will produce your second artifact from this course.   
\*\*This exercise challenges your understanding of all the SQL elements we learned in class.**

--==============================Create **A SINGLE** SELECT STATEMENT======================

--Write **1 complex query** using **STAYWELL Property** Tables and incorporate as many checklist elements as possible.

/\*Out of the 30 checklist elements, the query must contain at least 20 to obtain full points, strive to contain all 30 elements (which will be the goal for exercise part 2)

If 20 in one query is unobtainable, add a second or third query to capture additional elements\*/

--Write **the corresponding business question** for that query.

--Ensure proper formatting. Meaning it should be easy to read, and if I copy all and paste to execute in SQL, it should execute without error.

--Proper comments within the code is key to demonstrating your understanding of each component, keeping in mind that future coworkers or other teams should be able to reasonably understand your code and its purpose based on the query and comments alone.

-- Check off the elements as you create. Notice that some elements overlap, meaning incorporating one may check off more than one element.

--===================Prof Wang’s Example SINGLE SELECT STATEMENT======================

SELECT CUST\_ID, I.INVOICE\_NUM, INVOICE\_DATE, SUM(QUANTITY \* QUOTED\_PRICE) AS INVOICE\_TOTAL

FROM INVOICES I INNER JOIN INVOICE\_LINE IL ON I.INVOICE\_NUM = IL.INVOICE\_NUM

WHERE CUST\_ID IN (SELECT CUST\_ID

FROM CUSTOMER

WHERE CREDIT\_LIMIT > BALANCE)

GROUP BY CUST\_ID, I.INVOICE\_NUM, INVOICE\_DATE

HAVING SUM(QUANTITY \* QUOTED\_PRICE) > 250

ORDER BY I.INVOICE\_NUM ASC;

---SQL Query Clauses:

SELECT

FROM

WHERE

GROUP BY

HAVING

ORDER BY

---Aggregate Functions:

SUM/AVG/COUNT/MAX/MINDISTINCT

---Simple & Compound Conditions:

---Other Operators/Keywords/Characters:

Comparison Operators < > = !

ASC/DESC

IN (,,,)

---Computed Column:

Computed Column

AS Alias

---Subquery/Nesting Query/Nested Query:

Nested Query

---Table Joins

Inner Join

--======================= Example Corresponding business question =========================

List the customer ID, invoice number, invoice date, and invoice total

for each invoice with a total that exceeds $250,

placed by a customer in good credit standing.

Assign the column name INVOICE\_TOTAL to the column that displays invoice totals.

Order the results by invoice number.

--================== Create your Complex Query below ======================

SELECT o.FIRST\_NAME, o.LAST\_NAME, SUM(p.MONTHLY\_RENT) AS TOTAL\_RENT\_PAID

FROM OWNER o

LEFT JOIN PROPERTY p ON o.OWNER\_NUM = p.OWNER\_NUM

WHERE p.MONTHLY\_RENT IS NOT NULL AND p.MONTHLY\_RENT = ANY (SELECT p.MONTHLY\_RENT FROM PROPERTY WHERE p.MONTHLY\_RENT BETWEEN 1000 AND 3000)

GROUP BY o.FIRST\_NAME, o.LAST\_NAME

HAVING COUNT(p.PROPERTY\_ID) >1

ORDER BY TOTAL\_RENT\_EARNED DESC;

--================== Create your Business Question below ======================

For ANY owner in the owner table with more than one property from the property table that is not null and between 1000 and 3000, show the sum of rent from the property table as total rent paid. sort by descending.

--====================== Query Creation Checklist - 30 Items ======================

---SQL Query Clauses:

SELECT

FROM

WHERE

GROUP BY

HAVING

ORDER BY

---Aggregate Functions:

SUM/AVG/COUNT/MAX/MIN/DISTINCT

---Simple & Compound Conditions:

AND

OR

NOT

---Other Operators/Keywords/Characters:

Comparison Operators < > = !

BETWEEN

IS NULL/IS NOT NULL

ANY/ALL

TOP N

LIKE %

ASC/DESC

IN (,,,)

EXISTS

UNION

INTERSECT

PRODUCT

---Computed Column:

Computed Column

AS Alias

---Subquery/Nesting Query/Nested Query:

Nested Query

---Table Joins

Inner Join

Left Join/Right Join

Full Outer Join

Cartesian Join

Self Join