--===============================================================================  
--=========================Exercise 8 SQL Element Checklist ===========================

**\*\* 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.**

**6 Clauses + 1 Aggregate + 2 Conditions + 6 Operators + 2 Computed Columns + 3 Table Joins = 20**

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

---SQL Query Clauses: 6

SELECT

FROM

WHERE

GROUP BY

HAVING

ORDER BY

---Aggregate Functions:

SUM/AVG/COUNT/MAX/MIN/DISTINCT 1

---Simple & Compound Conditions: 3

AND

OR

NOT

---Other Operators/Keywords/Characters: 12

Comparison Operators < > = !

BETWEEN

IS NULL/IS NOT NULL

ANY/ALL

TOP N

LIKE %

ASC/DESC

IN (,,,)

EXISTS

UNION

INTERSECT

PRODUCT

---Computed Column: 2

Computed Column

AS Alias

---Subquery/Nesting Query/Nested Query: 1

Nested Query

---Table Joins 5

Inner Join

Left Join/Right Join

Full Outer Join

Cartesian Join

Self Join

1)

The company that manages all of the listed Staywell properties wants a comprehensive list of all possible combinations of residents if they were to share the property they reside in with one other resident to see if it cuts down on costs. List both residents’ IDs and full names; the property ID, full address, and owner’s number and full name. To save on the number of columns, you may combine the full names into one (e.g. (FIRST\_NAME + SURNAME) AS FULL\_NAME)

SELECT R1.RESIDENT\_ID, (R1.FIRST\_NAME + R1.SURNAME) AS FULL\_NAME,

R2.RESIDENT\_ID, (R2.FIRST\_NAME + R2.SURNAME) AS FULL NAME,

R2.PROPERTY\_ID, P.ADDRESS, O.OWNER\_NUM, (O.FIRST\_NAME + ‘ ’ + O.LAST\_NAME) AS FULL\_NAME

FROM RESIDENTS R1 INNER JOIN RESIDENTS R2 ON R1.PROPERTY\_ID = R2.PROPERTY\_ID

INNER JOIN PROPERTY P ON R2.PROPERTY\_ID = P.PROPERTY\_ID

INNER JOIN OWNER O ON P.OWNER\_NUM = O.OWNER\_NUM

WHERE R1.RESIDENT\_ID < R2.RESIDENT\_ID

SELECT

FROM

WHERE

Comparison Operators < > = !

Computed Column

AS Alias

Inner Join

Self Join

2)

StayWell has received a large number of service requests / complaints regarding several properties that are either partially resolved or completely unresolved. They want you to list the property address; as well as the number of reports, total estimated hours, and total spent hours for each property. Only show the top five results where the total spent hours is less than or equal to half of the total estimated hours (in other words, total spent hours \* 2 is less than or equal to total estimated hours). Then sort the results so that the most complaints appear first, followed by the least amount of spent hours, followed by the most amount of estimated hours.

SELECT TOP(5) P.ADDRESS, COUNT(R.PROPERTY\_ID) AS NUM\_REPTS,

SUM(R.EST\_HOURS) AS TOTAL\_ESTHRS, SUM(R.SPENT\_HOURS) AS TOTAL\_SPENTHRS

FROM SERVICE\_REQUEST R RIGHT JOIN PROPERTY P ON R.PROPERTY\_ID = P.PROPERTY\_ID

GROUP BY P.ADDRESS

HAVING SUM(R.SPENT\_HOURS) \* 2 <= SUM(R.EST\_HOURS)

ORDER BY COUNT(R.PROPERTY\_ID) DESC,

SUM(R.SPENT\_HOURS) ASC,

SUM(R.EST\_HOURS) DESC;

SELECT

FROM

GROUP BY

HAVING

ORDER BY

SUM/AVG/COUNT/MAX/MIN/DISTINCT

Comparison Operators < > = !

TOP N

ASC/DESC

AS Alias

Left Join/Right Join

3)

Following the previous business question, StayWell wants to notify the residents associated with any property that currently has reports about a temporary change in monthly rent and when to next expect a service (if any). For each resident and report, use an outer join to list the resident’s ID and full name (using the same method as the first business question); the property ID, address, and monthly rent cost; and the service request’s description, status, and next service date (if any). Be sure to only include residents/properties where the request status is not null or currently open, and if the next service date is within a week to a month (31 days) of today’s date: October 2nd, 2019. Sort the results by date, with the soonest first.

As a tip, you can use DATEDIFF (year/month/day, date1, date2) to calculate the amount of years, months, or days inbetween.

SELECT R.RESIDENT\_ID, (R.FIRST\_NAME + R.SURNAME) AS RES\_FULLNAME,

P.PROPERTY\_ID, P.ADDRESS, P.MONTHLY\_RENT, S.DESCRIPTION, S.STATUS, S.NEXT\_SERVICE\_DATE

FROM RESIDENTS R LEFT JOIN PROPERTY P ON R.PROPERTY\_ID = P.PROPERTY\_ID

LEFT JOIN SERVICE\_REQUEST S ON P.PROPERTY\_ID = S.PROPERTY\_ID

WHERE (S.STATUS IS NOT NULL OR S.STATUS NOT LIKE ‘%Open%’)

AND DATEDIFF(day, ‘2019-10-02’, S.NEXT\_SERVICE\_DATE) BETWEEN 7 AND 31

ORDER BY NEXT\_SERVICE\_DATE;

SELECT

FROM

WHERE

ORDER BY

AND

OR

BETWEEN

IS NULL/IS NOT NULL

LIKE %

ASC/DESC

AS Alias

Left Join/Right Join