```
/* ************************************************************************
        CUNY ACE UPSKILLING:  INTRODUCTION TO STRUCTURED QUERY LANGUAGE
                        SF21JOB#2, 2021/11/08 to 2021/12/13
                  https://folvera.commons.gc.cuny.edu/?cat=30
************************************************************************

 SESSION #3 (2021/11/15): MANIPULATING DATA

 1. Using built-in functions for strings
 2. Querying two or more datasets (tables or views) using `INNER JOIN`,
    `[OUTER] LEFT JOIN` and `[OUTER] RIGHT JOIN`
************************************************************************

1. A function, in any programming environment, lets you encapsulate reusable
   logic and build software that is ``composable``, i.e. built of pieces that
   can be reused and put together in a number of different ways to meet the
   needs of the users.  Functions hide the steps and the complexity from other
   code.
   https://www.simple-talk.com/sql/t-sql-programming/sql-server-functions-the-
    basics/

   1.1. Go to https://techonthenet.com/sql_server/functions/index_alpha.php
        for a detailed list of functions.

        1.1.1. As we mentioned before, so functions affect strings.

               CONCAT()    allows you to concatenate strings together
                           https://techonthenet.com/sql_server/functions/
                   concat.php
                           https://techonthenet.com/sql_server/functions/
                   concat2.php
               LEFT()      allows you to extract a substring from a string,
                           starting from the left-most character
                           https://techonthenet.com/sql_server/functions/
                   left.php
               LTRIM()     removes all space characters from the left-hand side
                           of a string
                           https://techonthenet.com/sql_server/functions/
                   ltrim.php
               LOWER()     converts all letters in the specified string to
                           lowercase
                           https://techonthenet.com/sql_server/functions/
                   lower.php
               REPLACE()   replaces a sequence of characters in a string with
                           another set of characters, not case-sensitive
                           https://techonthenet.com/sql_server/functions/
                   replace.php
               RIGHT()     allows you to extract a substring from a string,
                           starting from the right-most character
                           https://techonthenet.com/sql_server/functions/
                   right.php
               RTRIM()     removes all space characters from the right-hand
```

```
                            side of a string
                            https://techonthenet.com/sql_server/functions/
                        rtrim.php
                    SUBSTRING() allows you to extract a substring from a string
                            https://techonthenet.com/sql_server/functions/
                        substring.php
                    UPPER()     converts all letters in the specified string to
                            uppercase
                            https://techonthenet.com/sql_server/functions/
                        upper.php

            1.1.2. We also have functions that affect numeric values.

                    AVG()       returns the average value of an expression
                            https://techonthenet.com/sql_server/functions/avg.php
                    CEILING()   returns the smallest integer value that is greater
                            than or equal to a number
                            https://techonthenet.com/sql_server/functions/
                        ceiling.php
                    COUNT()     returns the count of an expression
                            https://techonthenet.com/sql_server/functions/
                        count.php
                    FLOOR()     returns the largest integer value that is equal to
                            or less than a number
                            https://techonthenet.com/sql_server/functions/
                        floor.php
                    LEN()       returns the length of the specified string... does
                            not include trailing space characters at the end the
                            string when calculating the length
                            https://techonthenet.com/sql_server/functions/len.php
                    MAX()       returns the maximum value of an expression
                            https://techonthenet.com/sql_server/functions/max.php
                    MIN()       returns the minimum value of an expression
                            https://techonthenet.com/sql_server/functions/min.php
                    RAND()      returns a random number or a random number within a
                            range
                            https://techonthenet.com/sql_server/functions/
                        rand.php
                    ROUND()     returns a number rounded to a certain number of
                            decimal places
                            https://techonthenet.com/sql_server/functions/
                        round.php
                    SUM()       returns the summed value of an expression
                            https://techonthenet.com/sql_server/functions/sum.php

        1.2. Note that every time you have a function, you need parenthesis.  Go to
             https://techonthenet.com/sql_server/functions/index_alpha.php for a
             complete list of built-in functions.

        1.3. As you might have noticed, some built-in functions manipulate strings.
             When working with numerical values, first we would have to convert
             them into strings as we will see later in the course.
```

```
 89
 90      1.4. Some other built-in functions ``return a single value, calculated from
 91           values in a column``.  These are referred to as aggregate functions
 92           (https://msdn.microsoft.com/en-us/library/ms173454.aspx).
 93
 94   2. Understanding the concepts above, we can now use them.
 95
 96      2.1. In the example below, we concatenate (put strings together) columns
 97           `FirstName` and `LastName` from table `AP1.ContactUpdates`.
 98   ********************************************************************** */
 99
100  SELECT CONCAT (
101      FirstName,
102      ' ',
103      LastName
104      ) AS NAME
105  FROM AP1.ContactUpdates;
106
107
108  /* ***********************************************************************
109      2.2. In the example below, we concatenate (put strings together) columns
110           `WE `, `ARE `, `LEARNING `, `SQL!`.
111   ********************************************************************** */
112
113  SELECT
114    CONCAT('WE ', 'ARE ', 'LEARNING ', 'SQL!');   -- returns `WE ARE LEARNING
115                                                  --         SQL!`
116
117
118  /* ***********************************************************************
119      2.3. In the example below, we concatenate (put strings together) columns
120           `FirstName` and `LastName` from table `AP1.ContactUpdates`, just like
121           the previous example.
122
123           2.3.1. We also use `LTRIM()` and `RTRIM()` to remove leading and
124                  trailing spaces from `FirstName` with `LTRIM(RTRIM(FirstName))`
125                  and `LastName` with `LTRIM(RTRIM(LastName))`.
126   ********************************************************************** */
127
128  SELECT CONCAT (
129      LTRIM(RTRIM(LastName)),
130      ', ',
131      LTRIM(RTRIM(FirstName))
132      ) AS NAME
133  FROM AP1.ContactUpdates;
134
135
136  /* ***********************************************************************
137      2.4. In the examples below, we use `UPPER()` to change a string to upper
138           case.
139   ********************************************************************** */
140
```

```
141 SELECT UPPER('this string is in upper case');    -- returns `THIS STRING SHOULD
142                                                  --          IN UPPER CASE`
143
144
145 /* ***************************************************************************
146     2.5. In the examples below, we use `LOWER()` to change a string to lower
147          case.
148  *************************************************************************** */
149
150 SELECT LOWER('BUT THIS STRING IS IN LOWER CASE.');
151                                                  -- returns `but this string is
152                                                  --      in lower case.`
153
154
155 /* ***************************************************************************
156     2.6. In the examples below, we use `RIGHT()` to extract characters from the
157          right.
158  *************************************************************************** */
159
160 SELECT RIGHT('apple', 2);                        -- returns `le`
161
162
163 /* ***************************************************************************
164     2.7. In the examples below, we use `LEFT()` to extract characters from the
165          left.
166  *************************************************************************** */
167
168 SELECT LEFT('apple', 2);                         -- returns `ap`
169
170
171 /* ***************************************************************************
172     2.8. In the examples below, we use `SUBSTRING()` to extract characters from
173          the middle -- same as built-in function `MID()` in other database
174          management systems like Oracle.
175  *************************************************************************** */
176
177 SELECT SUBSTRING('apple tree #5', 6, 10);        -- returns ` tree #5`
178
179
180 /* ***************************************************************************
181     2.9. In the example below, we use `LEN()` to retrieve the length of a
182          string.
183  *************************************************************************** */
184
185 SELECT LEN('tree      #5');                      -- returns 12
186
187
188 /* ***************************************************************************
189     2.10. In the examples below, we use `LTRIM()` and `RTRIM()` to remove any
190           leading and/or trailing spaces from the strings in single quotes.
191
192           2.10.1. Function `TRIM()` has not been implemented although
```

```
                    advertised by Microsoft
                    (https://docs.microsoft.com/en-us/sql/t-sql/functions/trim-
                     transact-sql).
 ********************************************************************** */

SELECT LTRIM('       tree'),                    -- 1. trimming leading spaces
  RTRIM('tree       '),                         -- 2. trimming trailing spaces
  LTRIM(RTRIM('       tree       '));           -- 3. trimming leading and
                                                --    trailing spaces


/* ***********************************************************************
    2.11. In the example below, we use `REPLACE()` to replace pattern `mstake`
          with `mistake`.  Since `mstake` exists in string `This is a mstake`,
          `REPLACE()` returns `This is a mistake`.
 ********************************************************************** */

SELECT REPLACE('This is a mstake', 'mstake', 'mistake');


/* ***********************************************************************
          2.11.1. In the example below, we use `REPLACE()` to replace pattern
                  `gg` with `mistake`.  Since `gg` does not exist in `This is a
                  mstake`, `REPLACE()` returns the original value.
 ********************************************************************** */

SELECT REPLACE('This is a mstake', 'gg', 'mistake');


/* ***********************************************************************
    2.12. In the example below, since there is no function to make the first
          letter of a string upper case and the rest lower case, we can use
          a combination of functions `UPPER()`, `LOWER()`, `RIGHT()`, `LEFT()`
          and `CONCAT()` working from the inside out.
 ********************************************************************** */

SELECT CONCAT (
    UPPER(LEFT('hELLO', 1)                      -- 1. retrieving first
                                                --    character from `hELLO`;
                                                --    returns `h`
    )                                           -- 2. making `h` upper case;
                                                --    returns `H`
    ,
    LOWER(SUBSTRING('hELLO', 2, LEN('hELLO')    -- 3. retrieving variable
      )                                         --    number of characters
                                                --    from character two (2)
                                                --    to the length of the
                                                --    string (integer value
                                                --    of 5); returns `ELLO`
    )                                           -- 4. making `ELLO` lower
                                                --    case; returns `ello`
    );                                          -- 5. concatenating all
```

```
                                            --    previous sections;
                                            --    returns `Hello`


/* ***************************************************************************
    2.13. In the example below, we use `REPLACE()` to change pattern `  ` (two
          spaces, `CHAR(32)+CHAR(32)` with ` ` (a single space, `CHAR(32)`).

                    SELECT REPLACE('tree      #5', '  ', ' ');

          2.12.1. Since string `tree      #5` has more than two spaces, we need
                  run several passes of `REPLACE()`.

          2.12.2. The statement runs from the inside out (3, 2, 1, 2, 3).

                    function 3                  -- 3. beginning of function #3:
                                                --    * receiving value of
                                                --      function #2
                      function 2                -- 2. beginning of function #2:
                                                --    * receiving value of
                                                --      function #1
                        function 1              -- 1. function #1:
                                                --    * receiving original
                                                --      value #0
                                                --    * returning new value #1
                      function 2                -- 2. end function of #2:
                                                --    * returning new value #2
                    function 3                  -- 3. end function of #3:
                                                --    * returning new value #3
                                                --     (final value)
 *************************************************************************** */


SELECT
  REPLACE(                                  -- 3. pass #3 to replace
                                            --    `CHAR(32)+CHAR(32)` for
                                            --    `CHAR(32)`
                                            --    * returns `tree #5` with
                                            --      1 space
    REPLACE(                                -- 2. pass #2 to replace
                                            --    `CHAR(32)+CHAR(32)` for
                                            --    `CHAR(32)`
                                            --    * returns `tree  #5` with
                                            --      2 space, which feeds
                                            --      pass #3
      REPLACE('tree      #5',               -- 1. pass #1 to replace
                                            --    `CHAR(32)+CHAR(32)` for
                                            --    `CHAR(32)`
                                            --    * returns `tree   #5`
                                            --      with 3 spaces, which
      '  ', ' '),                           --      feeds pass #2
    '  ', ' '),                             -- 2. end of pass #2
```

```
296     '  ', ' ');                                    -- 3. end of pass #3
297
298
299   /* ************************************************************************
300       2.13. In the example below, we use `REPLACE()` to replace pattern `tree`
301             for `fruit`.
302
303             2.13.1. Since pattern `tree` exists in `      tree      `  with
304                     leading and trailing spaces around `tree`, `REPLACE()`
305                     returns `      fruit      ` with leading and trailing
306                     spaces around word `fruit`.
307
308             2.13.2. We also use `RTRIM()` and `LTRIM()` to remove trailing and
309                     leading spaces respectively to get `fruit` without leading
310                     and trailing spaces.
311    ************************************************************************ */
312
313   SELECT RTRIM(LTRIM(REPLACE('      tree      ', 'tree', 'fruit')));
314
315
316   /* ************************************************************************
317       2.14. In the example below, we use `REPLACE()` to replace pattern `Box` for
318             `PO Box`.
319
320             2.14.1. The first pass (inner) of `REPLACE()` changes some fields to
321                     `PO PO Box`.
322
323             2.14.2. The second pass (outer) of `REPLACE()` changes the previous
324                     error (`PO PO Box`) to `PO Box`.
325    ************************************************************************ */
326
327   SELECT AP1.Vendors.VendorID,                    -- 1. fields using format
328     AP1.Vendors.VendorName,                       --    `schema.table.field`
329     REPLACE(                                      -- 2. second pass of
330                                                   --    `REPLACE()` working from
331                                                   --    inside out
332       REPLACE(AP1.Vendors.VendorAddress1,         -- 3. first pass of `REPLACE()`
333       'Box', 'PO Box'),                           --    working from inside out
334     'PO PO Box', 'PO Box') AS VendorAddress1,
335     AP1.Vendors.VendorAddress2,
336     AP1.Vendors.VendorCity,
337     AP1.Vendors.VendorState,
338     AP1.Vendors.VendorZipCode,
339     AP1.Vendors.VendorPhone,
340     AP1.Vendors.VendorContactLName,
341     AP1.Vendors.VendorContactFName,
342     AP1.Vendors.DefaultTermsID,
343     AP1.Vendors.DefaultAccountNo,
344     AP1.Terms.TermsID,
345     AP1.Terms.TermsDescription,
346     AP1.Terms.TermsDueDays
347   FROM AP1.Vendors
```

```
INNER JOIN AP1.Terms                                -- 4. `INNER JOIN` to retrieve
                                                    --    data in the first (left)
                                                    --    table (`AP1.Vendors`)
                                                    --    that is also in the
                                                    --    second (right) table
                                                    --    (`AP1.Terms`)
  ON AP1.Vendors.DefaultTermsID = AP1.Terms.TermsID;
                                                    -- 5. `ON` two fields with the
                                                    --    same values/data, but in
                                                    --    this case NOT the same
                                                    --    name (`DefaultTermsID`
                                                    --    and `TermsID`)


/* ***************************************************************************
    2.14. In the example below, we use the functions that we have covered to
          manipulate strings (any array of characters, such as letters and
          numbers).
 *************************************************************************** */

SELECT VendorID,
  LEFT(VendorName, 8) AS VendorNameL,               -- 1. retrieving eight (8)
                                                    --    characters from the left
                                                    --    of each string value in
                                                    --    column `VendorName`;
                                                    --    returns `US Posta`
                                                    --    (row 1)
  RIGHT(VendorName, 8) AS VendorNameR,              -- 2. retrieving eight (8)
                                                    --    characters from the right
                                                    --    of each string value in
                                                    --    column `VendorName`;
                                                    --    returns ` Service`
                                                    --    including the leading
                                                    --    space (row 1)
  CONCAT (                                          -- 3. concatenating the string
    VendorAddress1,                                 --    value in column
    ' ',                                            --    `VendorAddress1`, a space
    VendorAddress2                                  --    and the value in
    ) AS VendorAddess,                              --    column `VendorAddress2`;
                                                    --    returns `PO Box 96621 `
                                                    --    including the space since
                                                    --    there was no value in
                                                    --    `VendorAddress2` (row 2)
  UPPER(VendorCity) AS VendorCity,                  -- 4. changing the the string
                                                    --    value in column
                                                    --    `VendorCity` to upper
                                                    --    case; returns `MADISON`
                                                    --    (row 1)
  LOWER(VendorState) AS VendorState,                -- 5. changing the the string
                                                    --    value in column
                                                    --    `VendorState` to lower
                                                    --    case; returns `dc`
```

```
                                                --    (row 2)
  VendorZipCode,
  SUBSTRING(VendorPhone, 4, 3) AS VendorPhone,  -- 6. retrieving three (3)
                                                --    characters starting from
                                                --    the character four (4)
                                                --    of each string value in
                                                --    column `VendorName`;
                                                --    returns `555` (row 1) and
                                                --    `255` (row 73)
  REPLACE(VendorContactLName, 'en', 'XX')       -- 7. replacing pattern `en` in
    AS VendorContactLName,                      --    each string value in
                                                --    column
                                                --    `VendorContactLName` with
                                                --    pattern `XX` when found;
                                                --    returns `MaegXX` (row 7)
                                                --    and `AileXX` (row 16)
  VendorContactFName,
  LEN(VendorContactFName)                       -- 8. retrieving the length as
    AS VendorContactFNameLEN,                   --    an integer of each string
                                                --    value in column
                                                --    `VendorContactFName`;
                                                --    returns 9 (row 1)
  DefaultTermsID,
  DefaultAccountNo
FROM AP1.Vendors;


/* ****************************************************************************
    2.14. In the example below, we write a query (`SELECT` statement) calling
          all shared data (`INNER JOIN`) from tables `AP1.Vendors`,
          `AP1.Invoices` and `AP1.InvoiceLineItems` using the following syntax.

                    SELECT table1.field1,
                      table1.field2 ...
                      table2.field1,
                      table2.field2 ...
                      table3.field1,
                      table3.field2 ...
                    FROM table1
                    INNER JOIN table2
                      ON table1.common_field1(id1) = table2.common_field1(id1)
                    INNER JOIN table3
                      ON table1.common_field2(id2) = table3.common_field2(id2);

          Then we can delete or rename using an alias the duplicate name of the
          columns.
 **************************************************************************** */

SELECT AP1.Vendors.VendorID,
  AP1.Vendors.VendorName,
  AP1.Vendors.VendorAddress1,
  AP1.Vendors.VendorAddress2,
```

```
452    AP1.Vendors.VendorCity,
453    AP1.Vendors.VendorState,
454    AP1.Vendors.VendorZipCode,
455    AP1.Vendors.VendorPhone,
456    AP1.Vendors.VendorContactLName,
457    AP1.Vendors.VendorContactFName,
458    AP1.Vendors.DefaultTermsID,
459    AP1.Vendors.DefaultAccountNo,
460    AP1.Invoices.InvoiceID,
461    -- AP1.Invoices.VendorID,                    -- 1. duplicate column name
462                                                --    (`VendorID`), which can
463                                                --    be removed (commented
464                                                --    out, in this case)
465                                                --    without affecting the
466                                                --    query output; could also
467                                                --    be renamed
468    AP1.Invoices.InvoiceNumber,
469    AP1.Invoices.InvoiceDate,
470    AP1.Invoices.InvoiceTotal,
471    AP1.Invoices.PaymentTotal,
472    AP1.Invoices.CreditTotal,
473    AP1.Invoices.TermsID,
474    AP1.Invoices.InvoiceDueDate,
475    AP1.Invoices.PaymentDate,
476    -- AP1.InvoiceLineItems.InvoiceID,           -- 2. duplicate column name
477                                                --    (`InvoiceID`), which can
478                                                --    be removed (commented
479                                                --    out, in this case)
480                                                --    without affecting the
481                                                --    query output; could also
482                                                --    be renamed
483    AP1.InvoiceLineItems.InvoiceSequence,
484    AP1.InvoiceLineItems.AccountNo,
485    AP1.InvoiceLineItems.InvoiceLineItemAmount,
486    AP1.InvoiceLineItems.InvoiceLineItemDescription
487  FROM AP1.Vendors                              -- 3. from table `AP1.Vendors`
488  INNER JOIN AP1.Invoices                       -- 4. `INNER JOIN` to retrieve
489                                                --    data in the first (left)
490                                                --    table (`AP1.Vendors`)
491                                                --    that is also in the
492                                                --    second (right) table
493                                                --    (`AP1.Invoices`)
494    ON AP1.Vendors.VendorID = AP1.Invoices.VendorID
495                                                -- 5. `ON` two fields with the
496                                                --    same values/data and the
497                                                --    same name (`VendorID`);
498                                                --    specifying the relation
499                                                --    between tables
500                                                --    `AP1.Vendors` and
501                                                --    `AP1.Invoices`
502  INNER JOIN AP1.InvoiceLineItems               -- 6. `INNER JOIN` to retrieve
503                                                --    data in the first (left)
```

```
                                                --    table (`AP1.Invoices`)
                                                --    that is also in the
                                                --    second (right) table
                                                --    (`AP1.InvoiceLineItems`)
  ON AP1.Invoices.InvoiceID = AP1.InvoiceLineItems.InvoiceID;
                                                -- 7. `ON` two fields with the
                                                --    same values/data and the
                                                --    same name (`InvoiceID`);
                                                --    specifying the relation
                                                --    between tables
                                                --    `AP1.Invoices` and
                                                --    `AP1.InvoiceLineItems`


/* ****************************************************************************
 4. LAB #2
    Write a query without duplicate rows (`SELECT DISTINCT`)
    4.1. to call all columns from `AP1.Vendors` and `AP1.Invoices`, shared data
         only (`INNER JOIN`)
    4.2. to present `VendorPhone` in `(123) 456-7890` structure.
 **************************************************************************** */

SELECT DISTINCT -- 1. to retrieve unique rows
  AP1.Vendors.VendorID,
  AP1.Vendors.VendorName,
  AP1.Vendors.VendorAddress1,
  AP1.Vendors.VendorAddress2,
  AP1.Vendors.VendorCity,
  AP1.Vendors.VendorState,
  AP1.Vendors.VendorZipCode,
  CONCAT (
    '(',                                        -- 1. concatenation of a
                                                --    opening parenthesis,
    LEFT(AP1.Vendors.VendorPhone, 3),           --    three (3) characters from
                                                --    the left of
    ') ',                                       --    `VendorPhone`, closing
                                                --    parenthesis, the
    SUBSTRING(AP1.Vendors.VendorPhone, 4, 3),   --    substring from character
                                                --    four (4) taking three (3)
                                                --    characters of
    '-',                                        --    `VendorPhone`, a hyphen
                                                --    between branch and the
                                                --    subscriber number and
    RIGHT(AP1.Vendors.VendorPhone, 4)           --    four (4) characters from
                                                --    the right of
                                                --    `VendorPhone` with alias
    AS VendorPhone,                             --    `VendorPhone`
  AP1.Vendors.VendorContactLName,
  AP1.Vendors.VendorContactFName,
  AP1.Vendors.DefaultTermsID,
  AP1.Vendors.DefaultAccountNo,
  AP1.Invoices.InvoiceID,
```

```
556    -- AP1.Invoices.VendorID AS Expr1,            -- 2. commenting out duplicate
557                                                   --    field `VendorID`
558    AP1.Invoices.InvoiceNumber,
559    AP1.Invoices.InvoiceDate,
560    AP1.Invoices.InvoiceTotal,
561    AP1.Invoices.PaymentTotal,
562    AP1.Invoices.CreditTotal,
563    AP1.Invoices.TermsID,
564    AP1.Invoices.InvoiceDueDate,
565    AP1.Invoices.PaymentDate
566  FROM AP1.Vendors
567  INNER JOIN AP1.Invoices
568    ON AP1.Vendors.VendorID = AP1.Invoices.VendorID;
569
570
571  /* ***************************************************************************
572   https://folvera.commons.gc.cuny.edu/?p=1009
573   *************************************************************************** */
```