```sql
/* ********************************************************************************
          INTRODUCTION TO STRUCTURED QUERY LANGUAGE FOR DATA ANALYTICS
                    WS24SQL10001, 2024/03/11 - 2024/04/10
                  https://folvera.commons.gc.cuny.edu/?cat=34
 ********************************************************************************

  SESSION #3 (2024/03/18): MANIPULATING DATA

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

 1. LAB 1
    Write a query calling all shared rows/records (`INNER JOIN`) from
    `AP1.Invoices`, `AP1.Terms` and `AP1.Vendors`.
    * Delete or rename the duplicate name of the columns.
 ******************************************************************************** */

SELECT AP1.Invoices.InvoiceID,
  AP1.Invoices.VendorID,
  AP1.Invoices.InvoiceNumber,
  AP1.Invoices.InvoiceDate,
  AP1.Invoices.InvoiceTotal,
  AP1.Invoices.PaymentTotal,
  AP1.Invoices.CreditTotal,
  AP1.Invoices.TermsID,
  AP1.Invoices.InvoiceDueDate,
  AP1.Invoices.PaymentDate,
  -- AP1.Terms.TermsID,                          -- 1. duplicate column name
                                                 --    (`TermsID`), which can
                                                 --    be removed (commented
                                                 --    out, in this case)
                                                 --    without affecting the
                                                 --    query output; could also
                                                 --    be renamed
  AP1.Terms.TermsDescription,
  AP1.Terms.TermsDueDays,
  -- AP1.Vendors.VendorID,                       -- 2. duplicate column name
                                                 --    (`VendorID`), which can
                                                 --    be removed (commented
                                                 --    out, in this case)
                                                 --    without affecting the
                                                 --    query output; could also
                                                 --    be renamed
  AP1.Vendors.VendorName,
  AP1.Vendors.VendorAddress1,
  AP1.Vendors.VendorAddress2,
  AP1.Vendors.VendorCity,
  AP1.Vendors.VendorState,
  AP1.Vendors.VendorZipCode,
  AP1.Vendors.VendorPhone,
  AP1.Vendors.VendorContactLName,
  AP1.Vendors.VendorContactFName,
  AP1.Vendors.DefaultTermsID,
  AP1.Vendors.DefaultAccountNo
FROM AP1.Invoices                                -- 3. from table `AP1.Invoices`
INNER JOIN AP1.Terms                             -- 4. `INNER JOIN` to retrieve
                                                 --    data in the first (left)
                                                 --    table (`AP1.Invoices`)
                                                 --    that is also in the
                                                 --    second (right) table
                                                 --    (`AP1.Terms`)
  ON AP1.Invoices.TermsID = AP1.Terms.TermsID    -- 5. `ON` two fields with the
                                                 --    same values/data and the
```

```
                                                --    same name (`TermsID`);
                                                --    specifying the relation
                                                --    between tables
                                                --    `AP1.Invoices` and
                                                --    `AP1.Terms`
INNER JOIN AP1.Vendors                          -- 6. `INNER JOIN` to retrieve
                                                --    data in the second (left)
                                                --    table (`AP1.Terms`) that
                                                --    is also in the third
                                                --    (right) table
                                                --    (`AP1.Vendors`)
  ON AP1.Vendors.VendorID = AP1.Invoices.VendorID
                                                -- 7. `ON` two fields with the
                                                --    same values/data and the
                                                --    same name (`VendorID`);
                                                --    specifying the relation
                                                --    between tables
                                                --    `AP1.Vendors` and
                                                --    `AP1.Invoices`
    AND AP1.Vendors.DefaultTermsID = AP1.Terms.TermsID;
                                                -- 7. `AND` two other fields
                                                --    with the same values/data
                                                --    and different names
                                                --    (`DefaultTermsID` in
                                                --    `AP1.Vendors`, which has
                                                --    the same data as
                                                --    `TermsID` in
                                                --    `AP1.Terms`); specifying
                                                --    the relation between
                                                --    tables `AP1.Vendors` and
                                                --    `AP1.Terms`


/* ***************************************************************************
    * As an alternative, the code above can also be written using an alias
      (`AS`) for each table in order to simplify the code.

                    `i` for `AP1.Invoices`
                    `t` for `AP1.Terms`
                    `v` for `AP1.Vendors`

      Note that, if you use an alias (`AS`) for a table (for example, `v` for
      `AP1.Vendors`), you must use the alias (`v`) when calling the table
      (`AP1.Vendors`) in the query (for example, calling `AP1.Vendors.VendorID`
      as `v.VendorID`).
 *************************************************************************** */

SELECT i.InvoiceID,
  i.VendorID,
  i.InvoiceNumber,
  i.InvoiceDate,
  i.InvoiceTotal,
  i.PaymentTotal,
  i.CreditTotal,
  i.TermsID,
  i.InvoiceDueDate,
  i.PaymentDate,
  t.TermsDescription,
  t.TermsDueDays,
  v.VendorName,
  v.VendorAddress1,
  v.VendorAddress2,
  v.VendorCity,
  v.VendorState,
  v.VendorZipCode,
```

```
    v.VendorPhone,
    v.VendorContactLName,
    v.VendorContactFName,
    v.DefaultTermsID,
    v.DefaultAccountNo
FROM AP1.Invoices AS i
INNER JOIN AP1.Terms AS t
  ON i.TermsID = t.TermsID
INNER JOIN AP1.Vendors AS v
  ON v.VendorID = i.VendorID
    AND v.DefaultTermsID = t.TermsID;


/* ***************************************************************************
 2. 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/

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

    As we mentioned before, so functions affect strings.

          CONCAT()    allows you to concatenate strings together
                      https://techonthenet.com/sql_server/functions/concat.php

          + (plus)    also allows you to concatenate strings together although
                      adding NULL returns a NULL
                      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

          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

          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

    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

    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.

    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.

    Some other built-in functions ``return a single value, calculated from
    values in a column``.  These are referred to as aggregate functions
    (https://msdn.microsoft.com/en-us/library/ms173454.aspx).

 2. Understanding the concepts above, we can now use them.

    2.01. In the example below, we concatenate (put strings together) columns
          `FirstName` and `LastName` from table `AP1.ContactUpdates`.
 ********************************************************************** */

SELECT CONCAT (
    FirstName,
    ' ',
    LastName
    ) AS NAME
FROM AP1.ContactUpdates;
```

```
/* ************************************************************************
    2.02. In the example below, we concatenate (put strings together) columns
          `WE `, `ARE `, `LEARNING `, `SQL!` and print the result to the
          console.
 ************************************************************************ */

PRINT CONCAT('WE ','ARE ','LEARNING ','SQL!');  -- returns `WE ARE LEARNING
                                                --    SQL!`


/* ************************************************************************
    2.03. In the example below, we concatenate (put strings together) columns
          `FirstName` and `LastName` from table `AP1.ContactUpdates`, just like
          the previous example.

          We also use `LTRIM()` and `RTRIM()` to remove leading and trailing
          spaces from `FirstName` with `LTRIM(RTRIM(FirstName))` and `LastName`
          with `LTRIM(RTRIM(LastName))`.
 ************************************************************************ */

SELECT CONCAT (
    LTRIM(RTRIM(LastName)),
    ', ',
    LTRIM(RTRIM(FirstName))
    ) AS NAME
FROM AP1.ContactUpdates;


/* ************************************************************************
    2.04. In the examples below, we use `UPPER()` to change a string to upper
          case and print the result to console.
 ************************************************************************ */

PRINT UPPER('this string is in upper case');    -- returns `THIS STRING SHOULD
                                                --    IN UPPER CASE`


/* ************************************************************************
    2.05. In the examples below, we use `LOWER()` to change a string to lower
          case.
 ************************************************************************ */

PRINT LOWER('BUT THIS STRING IS IN LOWER CASE.');
                                                -- returns `but this string is
                                                --    in lower case.`


/* ************************************************************************
    2.06. In the examples below, we use `RIGHT()` to extract characters from
          the right.
 ************************************************************************ */

PRINT RIGHT('apple', 2);                        -- returns `le`


/* ************************************************************************
    2.07. In the examples below, we use `LEFT()` to extract characters from the
          left.
 ************************************************************************ */

PRINT LEFT('apple', 2);                         -- returns `ap`


/* ************************************************************************
    2.08. In the examples below, we use `SUBSTRING()` to extract characters
```

```
          from the middle -- same as the built-in function `MID()` in other
          relational database management systems (RDBMS) like Oracle -- and
          print the result to the console
 ************************************************************************** */

PRINT SUBSTRING('apple tree #5', 6, 10);          -- returns ` tree #5`


/* **************************************************************************
    2.09. In the example below, we use `LEN()` to retrieve the length of a
          string.
 ************************************************************************** */

PRINT LEN('tree      #5');                        -- returns 12 (numeric value)


/* **************************************************************************
    2.10. In the examples below, we use `LTRIM()` and `RTRIM()` to remove any
          leading and/or trailing spaces from the strings in single quotes and
          print the result to the console.

          We could also use function `TRIM()` only in SQL Server
          (https://docs.microsoft.com/en-us/sql/t-sql/functions/trim-transact-sql).
 ************************************************************************** */

PRINT 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`.
 ************************************************************************** */

PRINT REPLACE('This is a mstake', 'mstake', 'mistake');
                                                 -- returns `This is a mistake`


/* **************************************************************************
          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.
 ************************************************************************** */

PRINT REPLACE('This is a mstake', 'gg', 'mistake');
                                                 -- returns `This is a mstake`


/* **************************************************************************
    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 and print the result to
          the console.
 ************************************************************************** */

PRINT 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)`).

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

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

          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)
 ************************************************************************ */

PRINT
  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
  '  ', ' ');                                    -- 3. end of pass #3


/* ************************************************************************
```

```
    2.14. In the example below, we use `REPLACE()` to replace pattern `tree`
          for `fruit`.

          Since pattern `tree` exists in `     tree     ` with leading and
          trailing spaces around `tree`, `REPLACE()` returns `     fruit     `
          with leading and trailing spaces around word `fruit`.

          We also use `RTRIM()` and `LTRIM()` to remove trailing and leading
          spaces respectively to get `fruit` without leading and trailing
          spaces.
 ************************************************************************** */

PRINT RTRIM(LTRIM(REPLACE('     tree     ', 'tree', 'fruit')));
                                              -- returns `     fruit     `;
                                              --    empty spaces untouched


/* **************************************************************************
    2.15. In the example below, we use `REPLACE()` to replace pattern `Box` for
          `PO Box`.  The first pass (inner) of `REPLACE()` changes some fields
          to `PO PO Box`.  The second pass (outer) of `REPLACE()` changes the
          previous logical error (`PO PO Box`) to `PO Box`.
 ************************************************************************** */

SELECT AP1.Vendors.VendorID,                  -- 1. fields using format
  AP1.Vendors.VendorName,                     --    `schema.table.field`
  REPLACE(                                    -- 2. second pass of
                                              --    `REPLACE()` working from
                                              --    inside out
    REPLACE(AP1.Vendors.VendorAddress1,       -- 3. first pass of `REPLACE()`
    'Box', 'PO Box'),                         --    working from inside out
  'PO PO Box', 'PO Box') AS VendorAddress1,
  AP1.Vendors.VendorAddress2,
  AP1.Vendors.VendorCity,
  AP1.Vendors.VendorState,
  AP1.Vendors.VendorZipCode,
  REPLACE(                                    -- 4. replacing `() -` from the
                                              --    concatenation in #5
                                              --    instead of a CASE clause
                                              --    (logic block), which we
                                              --    will cover later
    CONCAT (                                  -- 5. concatenating an opening
      '(',                                    --    parenthesis, first three
      LEFT(Vendors.VendorPhone, 3),           --    characters of
      ') ',                                   --    `VendorPhone`, a closing
      SUBSTRING(Vendors.VendorPhone, 4, 3),   --    parenthesis with a space,
      '-',                                    --    the substring of
      RIGHT(Vendors.VendorPhone, 4)           --    `VendorPhone` starting
      ), '() -', '') AS VendorPhone           --    from the fourth character
                                              --    and taking 3, a hyphen
                                              --    and the last four
                                              --    characters of
                                              --    `VendorPhone`
  AP1.Vendors.VendorContactLName,
  AP1.Vendors.VendorContactFName,
  AP1.Vendors.DefaultTermsID,
  AP1.Vendors.DefaultAccountNo,
  AP1.Terms.TermsID,
  AP1.Terms.TermsDescription,
  AP1.Terms.TermsDueDays
FROM AP1.Vendors
INNER JOIN AP1.Terms                          -- 6. `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;
                                                    -- 7. `ON` two fields with the
                                                    --    same values/data, but in
                                                    --    this case NOT the same
                                                    --    name (`DefaultTermsID`
                                                    --    and `TermsID`) as in many
                                                    --    real world databases


/* ***************************************************************************
    2.16. The query above can also be written using an alias for each table.
 *************************************************************************** */

SELECT v.VendorID,
  v.VendorName,
  REPLACE(
    REPLACE(v.VendorAddress1,
    'Box', 'PO Box'),
  'PO PO Box', 'PO Box') AS VendorAddress1,
  v.VendorAddress2,
  v.VendorCity,
  v.VendorState,
  v.VendorZipCode,
  REPLACE(CONCAT (
      '(',
      LEFT(v.VendorPhone, 3),
      ') ',
      SUBSTRING(v.VendorPhone, 4, 3),
      '-',
      RIGHT(v.VendorPhone, 4)
      ), '() -', '') AS VendorPhone,
  v.VendorContactLName,
  v.VendorContactFName,
  v.DefaultTermsID,
  v.DefaultAccountNo,
  t.TermsID,
  t.TermsDescription,
  t.TermsDueDays
FROM AP1.Vendors AS v                               -- 1. using alias `v` for table
                                                    --    `AP1.Vendors`
INNER JOIN AP1.Terms AS t                           -- 2. using alias `t` for table
                                                    --    `AP1.Terms`
  ON v.DefaultTermsID = t.TermsID;


/* ***************************************************************************
    2.17. 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;


/* ********************************************************************************
    2.18. In the example below, we take the original value of
          `AP1.Vendors.VendorPhone` (`8005551205`) and divide it into three (3)
          sections -- the area code (`800'), the branch exchange (`555`) and
          the subscriber number (`1205`).  Then we concatenate the latter
          values with hard-coded values (opening and closing parenthesis and a
          hyphen) to present the phone number in a more legible way
          (`(800) 555-1205`).
 ******************************************************************************** */

SELECT VendorPhone AS original_number,              -- 1. dividing original value
                                                    --    of VendorPhone
                                                    --    (`8005551205`) to get
                                                    --    `(800) 555-1205`
  LEFT(VendorPhone, 3) AS area_code,                -- 2. retrieving three (3)
                                                    --    characters from the left
                                                    --    (`800', area code)
  SUBSTRING(VendorPhone, 4, 3) AS branch_exch,      -- 3. retrieving the middle
```

```
                                                    --    three (3) characters
                                                    --    starting for the fourth
                                                    --    (4th) character (`555`,
                                                    --    branch exchange)
  RIGHT(VendorPhone, 4) AS subscriber_number,       -- 4. retrieving four (4)
                                                    --    characters from the right
                                                    --    (`1205`, subscriber
                                                    --    number)
  CONCAT (                                          -- 5. concatenating the latter
                                                    --    values plus hard-coded
                                                    --    values (opening and
                                                    --    closing parenthesis and a
                                                    --    hyphen)
    '(',                                            --    5.1. value 1, the opening
                                                    --         parenthesis
    LEFT(VendorPhone, 3),                           --    5.2. value 2, area code
    ') ',                                           --    5.3. value 3, the closing
                                                    --         parenthesis and a
                                                    --         space
    SUBSTRING(VendorPhone, 4, 3),                   --    5.4. value 4, branch
                                                    --         exchange
    '-',                                            --    5.5. value 5, hyphen
    RIGHT(VendorPhone, 4)                           --    5.6. value 6, subscriber
                                                    --         number
    ) AS legible_phone_number
FROM AP1.Vendors;


/* ***************************************************************************
 https://folvera.commons.gc.cuny.edu/?p=1264
 *************************************************************************** */
```