```
/* ******************************************************************************
        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 #4 (2021/11/17): MANIPULATING DATA

  1. Using built-in functions for numeric values including aggregate functions
     and `GROUP BY`
  2. Using clauses `ORDER BY`, `CASE`, `WHERE` and operators
  3. Sub-queries
 *******************************************************************************

 1. ``In mathematical sets, the null set, also called the empty set, is the set
    that does not contain anything. It is symbolized ∅ or { }.  There is only
    one null set.  This is because there is logically only one way that a set
    can contain nothing.
    The null set makes it possible to explicitly define the results of
    operations on certain sets that would otherwise not be explicitly
    definable.  The intersection of two disjoint sets (two sets that contain no
    elements in common) is the null set. For example:
    {1, 3, 5, 7, 9, ...}∩{2, 4, 6, 8, 10, ...} = ∅                [∩ = U+2229]
                                                                  [∅ = U+2205]
    The null set provides a foundation for building a formal theory of numbers.
    In axiomatic mathematics, zero is defined as the cardinality of (that is,
    the number of elements in) the null set.  From this starting point,
    mathematicians can build the set of natural numbers, and from there, the
    sets of integers and rational numbers.``
    http://whatis.techtarget.com/definition/null-set

    As such, NULL refers to a memory allocation with no value -- not an empty
    space since the latter has a value of `CHAR(32)`.

    Note that concatenating any VARCHAR (ANSI-complaint accepting ASCII,
    UTF-8) or NVARCHAR (Microsoft proprietary data type, not ANSI-complaint
    accepting ASCII, UTF-8 and especially Unicode) field to a NULL (no value,
    not a blank character) field using `+` instead of using the `CONCAT()`
    function will return NULL.

    In the example below, we lose data when concatenating `VendorAddress1`
    and `VendorAddress2` in the `AP1.Vendors` table when using `+`.

******************************************************************************
 2. ``An aggregate function performs a calculation on a set of values, and
    returns a single value. Except for COUNT, aggregate functions ignore null
    values.  Aggregate functions are often used with the GROUP BY clause of the
    SELECT statement.``
    https://docs.microsoft.com/en-us/sql/t-sql/functions/aggregate-functions-transact-sql

    2.1. In the example below, we search for the count of records from table
```

```
         `AP1.Vendors` where column `VendorState` has a value of `NY` and `NJ`.
         Since a field (a single data allocation) cannot have two values at the
         same time, the query returns no values.
 ***************************************************************************** */

SELECT COUNT(VendorState) AS CountVendorState
FROM AP1.Vendors
WHERE VendorState = 'NJ'
  AND VendorState = 'NY';                          -- returns 0 (zero)


/* ******************************************************************************
    2.2. In the example below, we search for the count of records from table
         `AP1.Vendors` where column `VendorState` has a value of `NY` or `NJ`.
         In other words, the field can have either value.
 ***************************************************************************** */

SELECT COUNT(VendorState) AS CountVendorState
FROM AP1.Vendors
WHERE VendorState = 'NJ'
  OR VendorState = 'NY';                           -- returns 7 (4 `NJ` & 3 `NY`)


/* ******************************************************************************
    2.3. In the example below, we search for the count of records from table
         `AP1.Vendors` with `DISTINCT` values in column `VendorState` -- in
         other words, the number of unique states.
 ***************************************************************************** */

SELECT COUNT(DISTINCT VendorState) AS CountVendorState
FROM AP1.Vendors;                                  -- returns 22


/* ******************************************************************************
    2.4. In the example below, we search for the count of records from table
         `AP1.Vendors`.  We can use `*` (read as ``all``) since we are looking
         for the number of all values -- in other words, of all records.
 ***************************************************************************** */

SELECT COUNT(*) AS CountOfRows
FROM AP1.Vendors;                                  -- returns 114


/* ******************************************************************************
    2.5. In the examples below, we retrieve the sum of values in column
         `InvoiceTotal` (`SUM(InvoiceTotal)`), average value of column
         `InvoiceTotal` (`AVG(InvoiceTotal)`), maximum value of column
         `InvoiceTotal` (`MAX(InvoiceTotal)`) and minimum value of column
         `InvoiceTotal` (`MIN(InvoiceTotal)`) from table `AP1.Invoices`.

         Note that these values do not have commas as dividers (1,000) or
         currency symbols.  If you need to include dividers, you would need to
```

```sql
          use the `FORMAT()` function.
 *************************************************************************** */

SELECT SUM(InvoiceTotal) AS InvoiceTotalSUM,    -- returns 214290.51
  AVG(InvoiceTotal) AS InvoiceTotalAVG,         -- returns 1879.7413
  MAX(InvoiceTotal) AS InvoiceTotalMAX,         -- returns 37966.19
  MIN(InvoiceTotal) AS InvoiceTotalMIN          -- returns 6.00
FROM AP1.Invoices;


/* ****************************************************************************
    2.6. In the examples below, we search for the sum, average, maximum and
         minimum value of column `InvoiceTotal` from table `AP1.Invoices`
         respectively as (nested queries) sub-queries.
 *************************************************************************** */

SELECT InvoiceID,
  VendorID,
  InvoiceNumber,
  InvoiceDate,
  InvoiceTotal,
  (                                             -- 1. beginning of nested query
    SELECT                                      --    between parenthesis to
      MAX(InvoiceTotal)                         --    get `MAX(InvoiceTotal)`
    FROM AP1.Invoices                           --    from table `AP1.Invoices`
    ) AS InvoiceTotalMAX,                       --    with alias
                                                --    `InvoiceTotalMAX` for
                                                --    nested query
  (                                             -- 2. beginning of nested query
    SELECT                                      --    between parenthesis to
      MIN(InvoiceTotal)                         --    get `MIN(InvoiceTotal)`
    FROM AP1.Invoices                           --    from table `AP1.Invoices`
    ) AS InvoiceTotalMIN,                       --    with alias
                                                --    `InvoiceTotalMIN` for
                                                --    nested query
  ROUND                                         -- 3. rounding value of the sub
  (                                             --    query to 2 decimal spaces
    (                                           --    3.1. beginning of nested
      SELECT                                    --         query between
        AVG(InvoiceTotal)                       --         parenthesis to get
                                                --         `AVG(InvoiceTotal)`
      FROM AP1.Invoices                         --         from table
      ),                                        --         `AP1.Invoices`
    2)                                          --    3.2. rounding the value
                                                --         the nested query to
                                                --         2 decimal spaces
  AS InvoiceTotalAVG,                           -- 4. with alias
                                                --    `InvoiceTotalAVG` for
                                                --    nested query & `ROUND()`
  PaymentTotal,
  CreditTotal,
  TermsID,
  InvoiceDueDate,
  PaymentDate
FROM AP1.Invoices
ORDER BY VendorID,
  InvoiceTotal;


/* ****************************************************************************
    2.7. When using aggregate functions, we need to use `GROUP BY`.  Otherwise
         we would get the following error.

                    ``Msg 8120, Level 16, State 1, Line 2
                    Column `AP1.Invoices.InvoiceID` is invalid in the select
                    list because it is not contained in either an aggregate
                    function or the GROUP BY clause.``

         When using `GROUP BY`, we need to list each column that we are calling
         (from `InvoiceID` to `PaymentDate`) not affected by the aggregate
         function.

         Note that `AVG(InvoiceTotal)` returns the same value as `InvoiceTotal`
         since the average only affects a single value (`InvoiceTotal`) within
         a single row.
 *************************************************************************** */

SELECT InvoiceID,
  VendorID,
  InvoiceNumber,
  InvoiceDate,
  InvoiceTotal,
  AVG(InvoiceTotal) AS InvoiceTotalAVG,          -- aggregate function `AVG()`
                                                 -- only affecting field
                                                 -- `AP1.Invoices.InvoiceTotal`
  PaymentTotal,
  CreditTotal,
  TermsID,
  InvoiceDueDate,
  PaymentDate
FROM AP1.Invoices
GROUP BY                                         -- must use `GROUP BY` because
  InvoiceID,                                     -- of the aggregate function;
  VendorID,                                      -- no exceptions to this rule
  InvoiceNumber,
  InvoiceDate,
  InvoiceTotal,
  PaymentTotal,
  CreditTotal,
  TermsID,
  InvoiceDueDate,
  PaymentDate
ORDER BY VendorID,                               -- `ORDER BY` placed after
  InvoiceTotal;                                  -- `GROUP BY`; no exceptions
```

```sql
                                                -- to this rule


/* ***************************************************************************
 3. LAB #3

    Write a query
    3.1. to call all columns and values from `AP1.Vendors` any related values
         from `AP1.ContactUpdates` (`LEFT JOIN`)

                    CONCAT (
                      AP1.ContactUpdates.FirstName,
                      ' ',
                      AP1.ContactUpdates.LastName
                      ) AS ContactName

    3.2. to put together `FirstName` and `LastName` in one field with alias
         `ContactName`,

                    LOWER(CONCAT (
                      LEFT(AP1.ContactUpdates.FirstName, 1),
                      AP1.ContactUpdates.LastName,
                      '@',
                      REPLACE(
                        REPLACE(
                          REPLACE(AP1.Vendors.VendorName, ' ', ''),
                        '&', ''),
                      ',', ''),
                      '.com'
                      )) AS ContactEmail

 *************************************************************************** */

SELECT AP1.ContactUpdates.VendorID,
  CONCAT (                                      -- 1. concatenation of
    AP1.ContactUpdates.FirstName,               --    `FirstName`, a single
    ' ',                                        --    space and `LastName` with
    AP1.ContactUpdates.LastName                 --    alias `ContactName`
    ) AS ContactName,
  LOWER(CONCAT (                                -- 2. concatenation of one
      LEFT(AP1.ContactUpdates.FirstName, 1),    --    character from left of
                                                --    `FirstName`, `LastName`
      AP1.ContactUpdates.LastName,              --    the `@` symbol,
      '@',                                      --    `VendorName` after minor
                                                --    cleaning (#2.1 to #2.4
                                                --    processed from innermost
                                                --    to outermost function in
                                                --    chain)
      REPLACE(                                  --    2.4. pass #4 of
                                                --         `REPLACE()` to
                                                --         change apostrophes
                                                --         (````) to no space
```

```sql
                                                --         (``)
        REPLACE(                                --    2.3. pass #3 of
                                                --         `REPLACE()` to
                                                --         change commas (`,`)
                                                --         to no space (``)
          REPLACE(                              --    2.2. pass #2 of
                                                --         `REPLACE()` to
                                                --         change `&` to no
                                                --         space (``)
            REPLACE(AP1.Vendors.VendorName,     --    2.1. pass #1 of
              ' ', ''),                         --         `REPLACE()` to
                                                --         change single spaces
                                                --         (` `) to no space
                                                --         (``); processed
                                                --         from the inside out
          '&', ''),                             --    2.2. closing pass #2
        ',', ''),                               --    2.3. closing pass #3
      '''', ''),                                --    2.4. closing pass #4
      '.com'                                    --    and hard-coded string
                                                --    `.com` with alias
      )) AS ContactEmail,                       --    `ContactEmail`
  AP1.Vendors.VendorName,
  AP1.Vendors.VendorAddress1,
  AP1.Vendors.VendorAddress2,
  AP1.Vendors.VendorCity,
  AP1.Vendors.VendorState,
  AP1.Vendors.VendorZipCode,
  AP1.Vendors.VendorPhone,
  CONCAT (                                      -- 3. same as #1
    AP1.Vendors.VendorContactFName,
    ' ',
    AP1.Vendors.VendorContactLName
    ) AS VendorContactName,
  LOWER(CONCAT (                                -- 4. same as #2
      LEFT(AP1.Vendors.VendorContactFName, 1),
      AP1.Vendors.VendorContactLName,
      '@',
      REPLACE(
        REPLACE(
          REPLACE(AP1.Vendors.VendorName, ' ', ''),
        '&', ''),
      ',', ''),
      '.com'
      )) AS VendorContactEmail,
  AP1.Vendors.DefaultTermsID,
  AP1.Vendors.DefaultAccountNo
FROM AP1.Vendors
LEFT JOIN AP1.ContactUpdates
  ON AP1.Vendors.VendorID = AP1.ContactUpdates.VendorID;


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

```sql
   3.3. to put together the first letter of `FirstName`, the complete
         `LastName`, `@`, `VendorName` (removing empty spaces between words and
         special characters like `&` and `,`) and `.com` as `ContactEmail`
         presenting the output in lower case,

                    CONCAT (
                        AP1.Vendors.VendorContactFName,
                        ' ',
                        AP1.Vendors.VendorContactLName
                        ) AS VendorContactName,

                    LOWER(CONCAT (
                          LEFT(AP1.Vendors.VendorContactFName, 1),
                          AP1.Vendors.VendorContactLName,
                          '@',
                          REPLACE(
                            REPLACE(
                              REPLACE(AP1.Vendors.VendorName, ' ', ''),
                            '&', ''),
                          ',', ''),
                          '.com'
                          )) AS VendorContactEmail

    3.4. and to put together `VendorContactFName` and `VendorContactLName` with
         aliases `VendorContactName` and `VendorContactEmail` (like #3.2).
 *************************************************************************** */

SELECT AP1.ContactUpdates.VendorID,
  CONCAT (                                      -- 1. concatenation of
    AP1.ContactUpdates.FirstName,               --    `FirstName`, a space
    ' ',                                        --    and `LastName` with alias
    AP1.ContactUpdates.LastName                 --    `ContactName`
    ) AS ContactName,
  LOWER(CONCAT (                                -- 2. lower case of
                                                --    concatenation of
      LEFT(AP1.ContactUpdates.FirstName, 1),    --    2.1. one character from
                                                --         the left from
                                                --         `FirstName`
      -- REPLACE(AP1.ContactUpdates.LastName,   --    2.2. replacing a single
      --  '''', ''),                            --         quote with an empty
                                                --         string (solution #1)
                                                --         commented  out
      REPLACE(AP1.ContactUpdates.LastName,      --    2.3. replacing CHAR(39)
        CHAR(39), ''),                          --         (single quote
                                                --         character) with an
                                                --         empty string
                                                --         (solution #2)
      '@',                                      --    2.4. the at (`@`) sign
      REPLACE(                                  --    2.5. replacing a comma
                                                --         with an empty string
        REPLACE(                                --    2.6. replacing an
                                                --         ampersand (`&`) with
                                                --         an empty string
          REPLACE(AP1.Vendors.VendorName,       --    2.7. replacing a space
            ' ', ''),                           --         with an empty string
                                                --         in field
                                                --         `VendorName`
        '&', ''),                               --         * closing #2.6
      ',', ''),                                 --         * closing #2.5
      '.com'                                    --    2.8. `.com` to complete
                                                --         email
      )) AS ContactEmail,                       -- 3. with alias `ContactEmail`
  -- AP1.Vendors.VendorID AS Expr1,             -- 4. duplicate column
                                                --    commented out (excluded)
  AP1.Vendors.VendorName,
  AP1.Vendors.VendorAddress1,
  AP1.Vendors.VendorAddress2,
  AP1.Vendors.VendorCity,
  AP1.Vendors.VendorState,
  AP1.Vendors.VendorZipCode,
  AP1.Vendors.VendorPhone,
  CONCAT (                                      -- 5. concatenation of
    AP1.Vendors.VendorContactFName,             --    `VendorContactFName`, an
    ' ',                                        --    empty space (` `),
    AP1.Vendors.VendorContactLName              --    `VendorContactLName`
  ) AS VendorContactName,                       --    from table `AP1.Vendors`
                                                --    with alias
                                                --    `VendorContactName`
  LOWER(CONCAT (                                -- 6. lower case of
                                                --    concatenation of
    LEFT(AP1.Vendors.VendorContactFName, 1),    --    6.1. one character from
                                                --         the left from
                                                --         `VendorContactFName`
    AP1.Vendors.VendorContactLName,             --    6.2  `VendorContactLName`
      '@',                                      --    6.3. the at (`@`) sign
      REPLACE(                                  --    6.4. replacing a comma
                                                --         with an empty string
                                                --         (``)
        REPLACE(                                --    6.5. replacing an
                                                --         ampersand (`&`) with
                                                --         an empty string
          REPLACE(AP1.Vendors.VendorName,       --    6.6. replacing a space
            ' ', ''),                           --         with an empty string
                                                --         (``) in field
                                                --         `VendorName`
        '&', ''),                               --         * closing #6.5
      ',', ''),                                 --         * closing #6.4
      '.com'                                    --    6.7. `.web` to complete
                                                --         email
      )) AS VendorContactEmail,                 -- 7. with alias
                                                --    `VendorContactName`
  AP1.Vendors.DefaultTermsID,
  AP1.Vendors.DefaultAccountNo
FROM AP1.Vendors                                -- 8. from table `AP1.Vendors`
```

```sql
LEFT JOIN AP1.ContactUpdates                        --    left-joined to table
                                                    --    `AP1.ContactUpdates`
  ON AP1.Vendors.VendorID = AP1.ContactUpdates.VendorID;
                                                    --    on shared data from
                                                    --    `VendorID` in tables
                                                    --    `AP1.ContactUpdate`
                                                    --    and `AP1.Vendors`


/* ***********************************************************************
    3.5. We can avoid getting a NULL when concatenating with the `+` sign using
         a `CASE` clause (a logic block).

                    CASE
                      WHEN condition1
                        THEN action1
                      WHEN condition2
                        THEN action2
                      ELSE escape_action
                    END
 *********************************************************************** */

SELECT AP1.Vendors.VendorID,
  AP1.Vendors.VendorName,
  CASE                                              -- 1. start of `CASE` clause
    WHEN AP1.Vendors.VendorAddress2 IS NOT NULL -- 2. condition #1 for clause
      THEN AP1.Vendors.VendorAddress1               -- 3. action to take if
      + AP1.Vendors.VendorAddress2                  --    condition #1 is satisfied
    ELSE AP1.Vendors.VendorAddress1                 -- 4. escape action when
                                                    --    previous conditions fail
    END AS VendorAddress,                           -- 5. end of `CASE` clause with
                                                    --    alias `VendorAddress`
  AP1.Vendors.VendorCity,
  AP1.Vendors.VendorState,
  CASE                                              -- 6. beginning of `CASE`
    WHEN AP1.Vendors.VendorZipCode IS NOT NULL  -- 7. condition #1 for clause
      THEN AP1.Vendors.VendorZipCode                -- 8. action to take if
      + '-0001'                                     --    condition #1 is satisfied
    ELSE ''                                         -- 9. escape action when
                                                    --    previous conditions fail
    END AS VendorZipCodePlus4,                      -- 10. end of `CASE` clause
                                                    --     with alias
                                                    --     `VendorZipCodePlus4`
  AP1.Vendors.VendorPhone,
  AP1.Vendors.VendorContactLName
    + ', '
    + AP1.Vendors.VendorContactFName AS VendorContactFName,
  AP1.Vendors.VendorContactFName
    + AP1.Vendors.VendorContactLName AS VendorContactFName,
  AP1.Vendors.VendorContactFName
    + AP1.Vendors.VendorContactLName
    + '@example.com' AS VendorContactEmail,
  AP1.Vendors.DefaultTermsID,
  AP1.Vendors.DefaultAccountNo,
  AP1.ContactUpdates.VendorID AS 'Vendor Check',
  AP1.ContactUpdates.LastName,
  AP1.ContactUpdates.FirstName,
  'New Column' AS NewColumn
FROM AP1.Vendors
LEFT JOIN AP1.ContactUpdates
  ON AP1.Vendors.VendorID = AP1.ContactUpdates.VendorID;


/* ****************************************************************************
    3.5. Therefore, we should use the `CONCAT()` function to avoid losing data.
 ************************************************************************** */

SELECT AP1.Vendors.VendorID,
  AP1.Vendors.VendorName,
  CONCAT (                                          -- 1. concatenating fields with
    AP1.Vendors.VendorAddress1,                     --    comma between them and
    ' ',                                            --    empty spaces (` `) for a
    AP1.Vendors.VendorAddress2                      --    logical display followed
    ) AS VendorAddress,                             --    by an alias to name
                                                    --    column in output; no need
                                                    --    for `CASE` as in #1.3
  AP1.Vendors.VendorCity,
  AP1.Vendors.VendorState,
  CONCAT (
    AP1.Vendors.VendorZipCode,
    '-0001'
    ) AS VendorZipCodePlus4,
  AP1.Vendors.VendorPhone,
  CONCAT (
    AP1.Vendors.VendorContactLName,
    ', ',
    AP1.Vendors.VendorContactFName
    ) AS VendorContactFName,
  CONCAT (
    AP1.Vendors.VendorContactFName,
    ' ',
    AP1.Vendors.VendorContactLName
    ) AS VendorContactFName,
  CONCAT (
    AP1.Vendors.VendorContactFName,
    AP1.Vendors.VendorContactLName,
    '@example.com'
    ) AS VendorContactEmail,
  AP1.Vendors.DefaultTermsID,
  AP1.Vendors.DefaultAccountNo,
  AP1.ContactUpdates.VendorID AS 'Vendor Check',
  AP1.ContactUpdates.LastName,
  AP1.ContactUpdates.FirstName,
  'New Column' AS NewColumn                         -- 2. value not in table, added
```

```
                                                 --    in the query
FROM AP1.Vendors
LEFT JOIN AP1.ContactUpdates
  ON AP1.Vendors.VendorID = AP1.ContactUpdates.VendorID;
                                                 -- 3. relation between the two
                                                 --    tables on shared field
                                                 --    `VendorID`

/* ***************************************************************************
 https://folvera.commons.gc.cuny.edu/?p=1015
 ************************************************************************* */
```