

```
1  /* *****
2      CUNY ACE UPSKILLING:  INTRODUCTION TO STRUCTURED QUERY LANGUAGE
3          SF21JOB#2, 2021/11/08 to 2021/12/13
4          https://folvera.commons.gc.cuny.edu/?cat=30
5  *****
6
7  SESSION #4 (2021/11/17): MANIPULATING DATA
8
9  1. Using built-in functions for numeric values including aggregate functions
10     and `GROUP BY`
11  2. Using clauses `ORDER BY`, `CASE`, `WHERE` and operators
12  3. Sub-queries
13  *****
14
15  1. ``In mathematical sets, the null set, also called the empty set, is the set
16     that does not contain anything. It is symbolized  $\emptyset$  or  $\{ \}$ . There is only
17     one null set. This is because there is logically only one way that a set
18     can contain nothing.
19     The null set makes it possible to explicitly define the results of
20     operations on certain sets that would otherwise not be explicitly
21     definable. The intersection of two disjoint sets (two sets that contain no
22     elements in common) is the null set. For example:
23     {1, 3, 5, 7, 9, ...}n{2, 4, 6, 8, 10, ...} =  $\emptyset$            [n = U+2229]
24                                                                [  $\emptyset$  = U+2205]
25
26     The null set provides a foundation for building a formal theory of numbers.
27     In axiomatic mathematics, zero is defined as the cardinality of (that is,
28     the number of elements in) the null set. From this starting point,
29     mathematicians can build the set of natural numbers, and from there, the
30     sets of integers and rational numbers.``
31     http://whatis.techtarget.com/definition/null-set
32
33     As such, NULL refers to a memory allocation with no value -- not an empty
34     space since the latter has a value of `CHAR(32)`.
35
36     Note that concatenating any VARCHAR (ANSI-complaint accepting ASCII,
37     UTF-8) or NVARCHAR (Microsoft proprietary data type, not ANSI-complaint
38     accepting ASCII, UTF-8 and especially Unicode) field to a NULL (no value,
39     not a blank character) field using `+` instead of using the `CONCAT()`
40     function will return NULL.
41
42     In the example below, we lose data when concatenating `VendorAddress1`
43     and `VendorAddress2` in the `AP1.Vendors` table when using `+`.
44  *****
45  2. ``An aggregate function performs a calculation on a set of values, and
46     returns a single value. Except for COUNT, aggregate functions ignore null
47     values. Aggregate functions are often used with the GROUP BY clause of the
48     SELECT statement.``
49     https://docs.microsoft.com/en-us/sql/t-sql/functions/aggregate-functions-
50     transact-sql
51
52     2.1. In the example below, we search for the count of records from table
```

```
52     `AP1.Vendors` where column `VendorState` has a value of `NY` and `NJ`.
53     Since a field (a single data allocation) cannot have two values at the
54     same time, the query returns no values.
55     ***** */
56
57 SELECT COUNT(VendorState) AS CountVendorState
58 FROM AP1.Vendors
59 WHERE VendorState = 'NJ'
60     AND VendorState = 'NY';           -- returns 0 (zero)
61
62
63 /* *****
64     2.2. In the example below, we search for the count of records from table
65     `AP1.Vendors` where column `VendorState` has a value of `NY` or `NJ`.
66     In other words, the field can have either value.
67     ***** */
68
69 SELECT COUNT(VendorState) AS CountVendorState
70 FROM AP1.Vendors
71 WHERE VendorState = 'NJ'
72     OR VendorState = 'NY';           -- returns 7 (4 `NJ` & 3 `NY`)
73
74
75 /* *****
76     2.3. In the example below, we search for the count of records from table
77     `AP1.Vendors` with `DISTINCT` values in column `VendorState` -- in
78     other words, the number of unique states.
79     ***** */
80
81 SELECT COUNT(DISTINCT VendorState) AS CountVendorState
82 FROM AP1.Vendors;                   -- returns 22
83
84
85 /* *****
86     2.4. In the example below, we search for the count of records from table
87     `AP1.Vendors`. We can use `*` (read as ``all``) since we are looking
88     for the number of all values -- in other words, of all records.
89     ***** */
90
91 SELECT COUNT(*) AS CountOfRows
92 FROM AP1.Vendors;                   -- returns 114
93
94
95 /* *****
96     2.5. In the examples below, we retrieve the sum of values in column
97     `InvoiceTotal` (`SUM(InvoiceTotal)`), average value of column
98     `InvoiceTotal` (`AVG(InvoiceTotal)`), maximum value of column
99     `InvoiceTotal` (`MAX(InvoiceTotal)`) and minimum value of column
100    `InvoiceTotal` (`MIN(InvoiceTotal)`) from table `AP1.Invoices`.
101
102    Note that these values do not have commas as dividers (1,000) or
103    currency symbols. If you need to include dividers, you would need to
```

```

104         use the `FORMAT()` function.
105  *****
106
107  SELECT SUM(InvoiceTotal) AS InvoiceTotalSUM,      -- returns 214290.51
108         AVG(InvoiceTotal) AS InvoiceTotalAVG,     -- returns 1879.7413
109         MAX(InvoiceTotal) AS InvoiceTotalMAX,     -- returns 37966.19
110         MIN(InvoiceTotal) AS InvoiceTotalMIN      -- returns 6.00
111  FROM AP1.Invoices;
112
113
114  /* *****
115     2.6. In the examples below, we search for the sum, average, maximum and
116         minimum value of column `InvoiceTotal` from table `AP1.Invoices`
117         respectively as (nested queries) sub-queries.
118  ***** */
119
120  SELECT InvoiceID,
121         VendorID,
122         InvoiceNumber,
123         InvoiceDate,
124         InvoiceTotal,
125         (
126             SELECT
127                 MAX(InvoiceTotal)
128             FROM AP1.Invoices
129             ) AS InvoiceTotalMAX,
130
131         (
132             SELECT
133                 MIN(InvoiceTotal)
134             FROM AP1.Invoices
135             ) AS InvoiceTotalMIN,
136
137         ROUND
138         (
139             (
140                 (
141                     SELECT
142                         AVG(InvoiceTotal)
143                     FROM AP1.Invoices
144                 ),
145                 2)
146             ) AS InvoiceTotalAVG,
147
148         PaymentTotal,
149         CreditTotal,
150         TermsID,

```

```

156 InvoiceDueDate,
157 PaymentDate
158 FROM AP1.Invoices
159 ORDER BY VendorID,
160 InvoiceTotal;
161
162
163 /* *****
164 2.7. When using aggregate functions, we need to use `GROUP BY`. Otherwise
165 we would get the following error.
166
167 `Msg 8120, Level 16, State 1, Line 2
168 Column `AP1.Invoices.InvoiceID` is invalid in the select
169 list because it is not contained in either an aggregate
170 function or the GROUP BY clause.`
171
172 When using `GROUP BY`, we need to list each column that we are calling
173 (from `InvoiceID` to `PaymentDate`) not affected by the aggregate
174 function.
175
176 Note that `AVG(InvoiceTotal)` returns the same value as `InvoiceTotal`
177 since the average only affects a single value (`InvoiceTotal`) within
178 a single row.
179 ***** */
180
181 SELECT InvoiceID,
182 VendorID,
183 InvoiceNumber,
184 InvoiceDate,
185 InvoiceTotal,
186 AVG(InvoiceTotal) AS InvoiceTotalAVG, -- aggregate function `AVG()`
187 -- only affecting field
188 -- `AP1.Invoices.InvoiceTotal`
189 PaymentTotal,
190 CreditTotal,
191 TermsID,
192 InvoiceDueDate,
193 PaymentDate
194 FROM AP1.Invoices
195 GROUP BY -- must use `GROUP BY` because
196 InvoiceID, -- of the aggregate function;
197 VendorID, -- no exceptions to this rule
198 InvoiceNumber,
199 InvoiceDate,
200 InvoiceTotal,
201 PaymentTotal,
202 CreditTotal,
203 TermsID,
204 InvoiceDueDate,
205 PaymentDate
206 ORDER BY VendorID, -- `ORDER BY` placed after
207 InvoiceTotal; -- `GROUP BY`; no exceptions

```

```

208 -- to this rule
209
210
211 /* *****
212 3. LAB #3
213
214 Write a query
215 3.1. to call all columns and values from `AP1.Vendors` any related values
216 from `AP1.ContactUpdates` (`LEFT JOIN`)
217
218         CONCAT (
219             AP1.ContactUpdates.FirstName,
220             ' ',
221             AP1.ContactUpdates.LastName
222         ) AS ContactName
223
224 3.2. to put together `FirstName` and `LastName` in one field with alias
225 `ContactName`,
226
227         LOWER(CONCAT (
228             LEFT(AP1.ContactUpdates.FirstName, 1),
229             AP1.ContactUpdates.LastName,
230             '@',
231             REPLACE(
232                 REPLACE(
233                     REPLACE(AP1.Vendors.VendorName, ' ', ''),
234                     '&', ''),
235                 ',',''),
236                 '.com'
237             )) AS ContactEmail
238
239 ***** */
240
241 SELECT AP1.ContactUpdates.VendorID,
242        CONCAT (
243            AP1.ContactUpdates.FirstName,
244            ' ',
245            AP1.ContactUpdates.LastName
246        ) AS ContactName,
247        LOWER(CONCAT (
248            LEFT(AP1.ContactUpdates.FirstName, 1),
249            AP1.ContactUpdates.LastName,
250            '@',
251            REPLACE(
252                REPLACE(
253                    REPLACE(AP1.Vendors.VendorName, ' ', ''),
254                    '&', ''),
255                    ',',''),
256                    '.com'
257                )) AS ContactEmail
258
259
-- 1. concatenation of
-- `FirstName`, a single
-- space and `LastName` with
-- alias `ContactName`
-- 2. concatenation of one
-- character from left of
-- `FirstName`, `LastName`
-- the `@` symbol,
-- `VendorName` after minor
-- cleaning (#2.1 to #2.4
-- processed from innermost
-- to outermost function in
-- chain)
-- 2.4. pass #4 of
-- `REPLACE()` to
-- change apostrophes
-- (```) to no space

```

```

260                                     --      (` `)
261      REPLACE(                         --      2.3. pass #3 of
262                                     --      `REPLACE()` to
263                                     --      change commas (`,`),`
264                                     --      to no space (` `)
265      REPLACE(                         --      2.2. pass #2 of
266                                     --      `REPLACE()` to
267                                     --      change `&` to no
268                                     --      space (` `)
269      REPLACE(AP1.Vendors.VendorName,  --      2.1. pass #1 of
270      ' ', ' '),                       --      `REPLACE()` to
271                                     --      change single spaces
272                                     --      (` `) to no space
273                                     --      (` `); processed
274                                     --      from the inside out
275      '&', ' '),                       --      2.2. closing pass #2
276      ', ', ' '),                     --      2.3. closing pass #3
277      ' ', ' '),                     --      2.4. closing pass #4
278      '.com'                          --      and hard-coded string
279                                     --      `.com` with alias
280    )) AS ContactEmail,              --      `ContactEmail`
281    AP1.Vendors.VendorName,
282    AP1.Vendors.VendorAddress1,
283    AP1.Vendors.VendorAddress2,
284    AP1.Vendors.VendorCity,
285    AP1.Vendors.VendorState,
286    AP1.Vendors.VendorZipCode,
287    AP1.Vendors.VendorPhone,
288    CONCAT (                          -- 3. same as #1
289      AP1.Vendors.VendorContactFName,
290      ' ',
291      AP1.Vendors.VendorContactLName
292    ) AS VendorContactName,
293    LOWER(CONCAT (                    -- 4. same as #2
294      LEFT(AP1.Vendors.VendorContactFName, 1),
295      AP1.Vendors.VendorContactLName,
296      '@',
297      REPLACE(
298        REPLACE(
299          REPLACE(AP1.Vendors.VendorName, ' ', ' '),
300          '&', ' '),
301          ', ', ' '),
302          '.com'
303        )) AS VendorContactEmail,
304    AP1.Vendors.DefaultTermsID,
305    AP1.Vendors.DefaultAccountNo
306 FROM AP1.Vendors
307 LEFT JOIN AP1.ContactUpdates
308 ON AP1.Vendors.VendorID = AP1.ContactUpdates.VendorID;
309
310
311 /* *****

```

312 3.3. to put together the first letter of `FirstName`, the complete
 313 `LastName`, `@`, `VendorName` (removing empty spaces between words and
 314 special characters like `&` and `,`) and `.com` as `ContactEmail`
 315 presenting the output in lower case,
 316

```

317         CONCAT (
318             AP1.Vendors.VendorContactFName,
319             ' ',
320             AP1.Vendors.VendorContactLName
321         ) AS VendorContactName,
322
323         LOWER(CONCAT (
324             LEFT(AP1.Vendors.VendorContactFName, 1),
325             AP1.Vendors.VendorContactLName,
326             '@',
327             REPLACE(
328                 REPLACE(
329                     REPLACE(AP1.Vendors.VendorName, ' ', ''),
330                     '&', ''),
331                 ', ', ''),
332             '.com'
333         )) AS VendorContactEmail
334
```

335 3.4. and to put together `VendorContactFName` and `VendorContactLName` with
 336 aliases `VendorContactName` and `VendorContactEmail` (like #3.2).

337 ***** */

```

338
339 SELECT AP1.ContactUpdates.VendorID,
340        CONCAT (
341            AP1.ContactUpdates.FirstName,
342            ' ',
343            AP1.ContactUpdates.LastName
344        ) AS ContactName,
345        LOWER(CONCAT (
346            LEFT(AP1.ContactUpdates.FirstName, 1),
347            AP1.ContactUpdates.LastName,
348            '@',
349            REPLACE(
350                REPLACE(
351                    REPLACE(AP1.ContactUpdates.LastName,
352                        CHAR(39), ''),
353                    '&', ''),
354                ', ', ''),
355            '.com'
356        )) AS VendorContactEmail,
357
358        REVERSE(
359            REPLACE(
360                REPLACE(
361                    REPLACE(
362                        REPLACE(
363
```

-- 1. concatenation of
 -- `FirstName`, a space
 -- and `LastName` with alias
 -- `ContactName`

-- 2. lower case of
 -- concatenation of
 -- 2.1. one character from
 -- the left from
 -- `FirstName`
 -- 2.2. replacing a single
 -- quote with an empty
 -- string (solution #1)
 -- commented out
 -- 2.3. replacing CHAR(39)
 -- (single quote
 -- character) with an
 -- empty string
 -- (solution #2)
 -- 2.4. the at (`@`) sign
 -- 2.5. replacing a comma
 -- with an empty string
 -- 2.6. replacing an
 -- ampersand (`&`) with

```

364 -- an empty string
365         REPLACE(AP1.Vendors.VendorName, -- 2.7. replacing a space
366             ' ', ''), -- with an empty string
367 -- in field
368 -- `VendorName`
369         '&', ''), -- * closing #2.6
370         ', ', ''), -- * closing #2.5
371         '.com' -- 2.8. `.com` to complete
372 -- email
373     )) AS ContactEmail, -- 3. with alias `ContactEmail`
374 -- AP1.Vendors.VendorID AS Expr1, -- 4. duplicate column
375 -- commented out (excluded)
376 AP1.Vendors.VendorName,
377 AP1.Vendors.VendorAddress1,
378 AP1.Vendors.VendorAddress2,
379 AP1.Vendors.VendorCity,
380 AP1.Vendors.VendorState,
381 AP1.Vendors.VendorZipCode,
382 AP1.Vendors.VendorPhone,
383 CONCAT ( -- 5. concatenation of
384     AP1.Vendors.VendorContactFName, -- `VendorContactFName`, an
385     ' ', -- empty space (` `),
386     AP1.Vendors.VendorContactLName -- `VendorContactLName`
387 ) AS VendorContactName, -- from table `AP1.Vendors`
388 -- with alias
389 -- `VendorContactName`
390 LOWER(CONCAT ( -- 6. lower case of
391 -- concatenation of
392     LEFT(AP1.Vendors.VendorContactFName, 1), -- 6.1. one character from
393 -- the left from
394 -- `VendorContactFName`
395     AP1.Vendors.VendorContactLName, -- 6.2 `VendorContactLName`
396     '@', -- 6.3. the at (`@`) sign
397     REPLACE( -- 6.4. replacing a comma
398 -- with an empty string
399 -- (``)
400     REPLACE( -- 6.5. replacing an
401 -- ampersand (`&`) with
402 -- an empty string
403     REPLACE(AP1.Vendors.VendorName, -- 6.6. replacing a space
404         ' ', ''), -- with an empty string
405 -- (``) in field
406 -- `VendorName`
407         '&', ''), -- * closing #6.5
408         ', ', ''), -- * closing #6.4
409         '.com' -- 6.7. `.web` to complete
410 -- email
411     )) AS VendorContactEmail, -- 7. with alias
412 -- `VendorContactName`
413 AP1.Vendors.DefaultTermsID,
414 AP1.Vendors.DefaultAccountNo
415 FROM AP1.Vendors -- 8. from table `AP1.Vendors`

```



```

416 LEFT JOIN AP1.ContactUpdates          -- left-joined to table
417                                     -- `AP1.ContactUpdates`
418 ON AP1.Vendors.VendorID = AP1.ContactUpdates.VendorID;
419                                     -- on shared data from
420                                     -- `VendorID` in tables
421                                     -- `AP1.ContactUpdate`
422                                     -- and `AP1.Vendors`
423
424
425 /* *****
426     3.5. We can avoid getting a NULL when concatenating with the `+` sign using
427         a `CASE` clause (a logic block).
428
429             CASE
430             WHEN condition1
431                 THEN action1
432             WHEN condition2
433                 THEN action2
434             ELSE escape_action
435             END
436 ***** */
437
438 SELECT AP1.Vendors.VendorID,
439        AP1.Vendors.VendorName,
440        CASE                                -- 1. start of `CASE` clause
441            WHEN AP1.Vendors.VendorAddress2 IS NOT NULL -- 2. condition #1 for clause
442                THEN AP1.Vendors.VendorAddress1 -- 3. action to take if
443                + AP1.Vendors.VendorAddress2 -- 4. condition #1 is satisfied
444            ELSE AP1.Vendors.VendorAddress1 -- 5. escape action when
445                                                -- previous conditions fail
446            END AS VendorAddress, -- 6. end of `CASE` clause with
447                                                -- alias `VendorAddress`
448        AP1.Vendors.VendorCity,
449        AP1.Vendors.VendorState,
450        CASE                                -- 6. beginning of `CASE`
451            WHEN AP1.Vendors.VendorZipCode IS NOT NULL -- 7. condition #1 for clause
452                THEN AP1.Vendors.VendorZipCode -- 8. action to take if
453                + '-0001' -- 9. condition #1 is satisfied
454            ELSE '' -- 9. escape action when
455                                                -- previous conditions fail
456            END AS VendorZipCodePlus4, -- 10. end of `CASE` clause
457                                                -- with alias
458                                                -- `VendorZipCodePlus4`
459        AP1.Vendors.VendorPhone,
460        AP1.Vendors.VendorContactLName
461        + ', '
462        + AP1.Vendors.VendorContactFName AS VendorContactFName,
463        AP1.Vendors.VendorContactFName
464        + AP1.Vendors.VendorContactLName AS VendorContactFName,
465        AP1.Vendors.VendorContactFName
466        + AP1.Vendors.VendorContactLName
467        + '@example.com' AS VendorContactEmail,

```

```

468 AP1.Vendors.DefaultTermsID,
469 AP1.Vendors.DefaultAccountNo,
470 AP1.ContactUpdates.VendorID AS 'Vendor Check',
471 AP1.ContactUpdates.LastName,
472 AP1.ContactUpdates.FirstName,
473 'New Column' AS NewColumn
474 FROM AP1.Vendors
475 LEFT JOIN AP1.ContactUpdates
476 ON AP1.Vendors.VendorID = AP1.ContactUpdates.VendorID;
477
478
479 /* *****
480 3.5. Therefore, we should use the `CONCAT()` function to avoid losing data.
481 ***** */
482
483 SELECT AP1.Vendors.VendorID,
484 AP1.Vendors.VendorName,
485 CONCAT ( -- 1. concatenating fields with
486 AP1.Vendors.VendorAddress1, -- comma between them and
487 ' ', -- empty spaces (` `) for a
488 AP1.Vendors.VendorAddress2 -- logical display followed
489 ) AS VendorAddress, -- by an alias to name
490 -- column in output; no need
491 -- for `CASE` as in #1.3
492 AP1.Vendors.VendorCity,
493 AP1.Vendors.VendorState,
494 CONCAT (
495 AP1.Vendors.VendorZipCode,
496 '-0001'
497 ) AS VendorZipCodePlus4,
498 AP1.Vendors.VendorPhone,
499 CONCAT (
500 AP1.Vendors.VendorContactLName,
501 ' ',
502 AP1.Vendors.VendorContactFName
503 ) AS VendorContactFName,
504 CONCAT (
505 AP1.Vendors.VendorContactFName,
506 ' ',
507 AP1.Vendors.VendorContactLName
508 ) AS VendorContactFName,
509 CONCAT (
510 AP1.Vendors.VendorContactFName,
511 AP1.Vendors.VendorContactLName,
512 '@example.com'
513 ) AS VendorContactEmail,
514 AP1.Vendors.DefaultTermsID,
515 AP1.Vendors.DefaultAccountNo,
516 AP1.ContactUpdates.VendorID AS 'Vendor Check',
517 AP1.ContactUpdates.LastName,
518 AP1.ContactUpdates.FirstName,
519 'New Column' AS NewColumn -- 2. value not in table, added

```

```
520 -- in the query
521 FROM AP1.Vendors
522 LEFT JOIN AP1.ContactUpdates
523 ON AP1.Vendors.VendorID = AP1.ContactUpdates.VendorID;
524 -- 3. relation between the two
525 -- tables on shared field
526 -- `VendorID`
527
528 /* *****
529 https://folvera.commons.gc.cuny.edu/?p=1015
530 ***** */
```