```
/* ************************************************************************
                    DATABASE ADMINISTRATION FUNDAMENTALS:
                  INTRODUCTION TO STRUCTURED QUERY LANGUAGE
                    SF21SQL1001, 2021/11/02 - 2021/12/09
                  https://folvera.commons.gc.cuny.edu/?cat=29
************************************************************************/

 SESSION #2 (2021/11/04): UNDERSTANDING CORE DATABASE CONCEPTS

 1. Learning history of SQL and basic concepts of the structure of a
    relational database
 2. Understanding structured programming
 3. Understanding naming convention
 4. Understanding basic syntax to query one table
************************************************************************/

1. Database professionals in the labor economy is on the rise.  By 2024, the
   US Bureau of Labor Statistics has projected an 18.8% job increase for
   `Software developers, applications` with a median annual income of $100,080
   and 20.9% for `Computer systems analysts` with a median annual income of
   $87,220 as of 04/14/2017 (https://bls.gov/emp/ep_table_104.htm) creating a
   surge for individuals who possess the right skills to store and query large
   data sets.

   +----------------------------------------+--------------+--------------+
   | Computer Systems Analysts              | $88,270 / yr | $42.44 / hr  |
   +----------------------------------------+--------------+--------------+
   | https://bls.gov/ooh/computer-and-information-technology/computer-systems-
    analysts.htm
   +----------------------------------------+--------------+--------------+
   | Database Administrators                | $87,020 / yr | $41.84 / hr  |
   +----------------------------------------+--------------+--------------+
   | https://bls.gov/ooh/computer-and-information-technology/database-
    administrators.htm
   +----------------------------------------+--------------+--------------+
   | Web Developers                         | $67,990 / yr | $32.69 / hr  |
   +----------------------------------------+--------------+--------------+
   | https://bls.gov/ooh/computer-and-information-technology/web-developers.htm
   +----------------------------------------+--------------+--------------+
   | Operations Research Analysts           | $81,390 / yr | $39.13 / hr  |
   +----------------------------------------+--------------+--------------+
   | https://bls.gov/ooh/math/operations-research-analysts.htm              |
   +------------------------------------------------------------------------

2. The following are concepts that you need to know.

   2.1. SQL (Structured Query Language) is a standardized programming language
        used for managing relational databases and performing various
        operations on the data in them.  Initially created in the 1970s, SQL
        is regularly used by database administrators, as well as by developers
        writing data integration scripts and data analysts looking to set up
        and run analytical queries.
```

```
51          https://searchsqlserver.techtarget.com/definition/SQL
52
53          The SQL programming language was first developed in the 1970s by IBM
54          researchers Raymond Boyce and Donald Chamberlin.  The programming
55          language, known then as SEQUEL, was created following the publishing
56          of Edgar Frank Todd's paper, ``A Relational Model of Data for Large
57          Shared Data Banks,`` in 1970.
58          https://businessnewsdaily.com/5804-what-is-sql.html
59
60          Refer to https://ibm.com/ibm/history/ibm100/us/en/icons/reldb/ for
61          more information on Edgar Frank Todd's paper.
62
63     2.2. T-SQL (Transact-SQL) is a set of programming extensions from Sybase
64          and Microsoft that add several features to the Structured Query
65          Language (SQL), including transaction control, exception and error
66          handling, row processing and declared variables.
67          https://searchsqlserver.techtarget.com/definition/T-SQL
68
69     2.3. Microsoft SQL Server is a relational database management system, or
70          RDBMS, that supports a wide variety of transaction processing,
71          business intelligence and analytics applications in corporate IT
72          environments.  It's one of the three market-leading database
73          technologies, along with Oracle Database and IBM's DB2.
74          https://searchsqlserver.techtarget.com/definition/SQL-Server
75
76     2.4. A server is a computer program that provides a service to another
77          computer programs (and its user).  In a data center, the physical
78          computer that a server program runs in is also frequently referred to
79          as a server.  That machine may be a dedicated server or it may be used
80          for other purposes as well.
81          In the client/server programming model, a server program awaits and
82          fulfills requests from client programs, which may be running in the
83          same or other computers.  A given application in a computer may
84          function as a  client with requests for services from other programs
85          and also as a server of requests from other programs.
86          https://whatis.techtarget.com/definition/server
87
88     2.5. A relational database management system (RDBMS) is a collection of
89          programs and capabilities that enable IT teams and others to create,
90          update, administer and otherwise interact with a relational database.
91          Most commercial RDBMSes use Structured Query Language (SQL) to access
92          the database, although SQL was invented after the initial development
93          of the relational model and is not necessary for its use.
94          https://searchdatamanagement.techtarget.com/definition/RDBMS-relational- ↵
             database-management-system
95
96     2.6. In computer programming, a schema (pronounced SKEE-mah) is the
97          organization or structure for a database.  The activity of data
98          modeling leads to a schema. (The plural form is schemata.  The term is
99          from a Greek word for ``form`` or ``figure.``  Another word from the
100         same source is ``schematic.``)  The term is used in discussing both
101         relational databases and object-oriented databases.  The term
```

```
102          sometimes seems to refer to a visualization of a structure and
103          sometimes to a formal text-oriented description.
104          https://searchsqlserver.techtarget.com/definition/schema
105
106     2.7. In computer programming, a table is a data structure used to organize
107          information, just as it is on paper.  There are many different types
108          of computer-related tables, which work in a number of different ways.
109          The following are examples of the more common types.
110          1) In data processing, a table (also called an array) is a organized
111             grouping of fields.  Tables may store relatively permanent data, or
112             may be frequently updated.  For example, a table contained in a
113             disk volume is updated when sectors are being written.
114          2) In a relational database, a table (sometimes called a file)
115             organizes the information about a single topic into rows and
116             columns.  For example, a database for a business would typically
117             contain a table for customer information, which would store
118             customers' account numbers, addresses, phone numbers, and so on as
119             a series of columns.  Each single piece of data (such as the
120             account number) is a field in the table. A column consists of all
121             the entries in a single field, such as the telephone numbers of all
122             the customers.  Fields, in turn, are organized as records, which
123             are complete sets of information (such as the set of information
124             about a particular customer), each of which comprises a row.  The
125             process of normalization determines how data will be most
126             effectively organized into tables.
127          https://whatis.techtarget.com/definition/table
128
129  3. Before we start, you should be familiar with the naming convention used in
130     T-SQL (https://searchsqlserver.techtarget.com/definition/T-SQL) using the
131     database for this course.
132
133                     PC12345\MSSQLSERVER         (server; name depending on the
134                      |                          machine you are using where
135                      |                          `PC12345` is the `HOSTNAME` and
136                      |                          `MSSQLSERVER` is the database
137                      |                          instance)
138                      |
139                      +- SF21SQL1001             (database in server
140                           |                     `PC12345\SQLSERVEREXPRESS`)
141                           |
142                           +- AP1                (schema in database
143                                |                `SF21SQL1001`)
144                                |
145                                +- ContactUpdates (table in schema `AP1`)
146                                     |
147                                     +- VendorID   (column in table
148                                                   `ContactUpdates`)
149
150     3.1. Using the structure above, `SF21SQL1001` is the database
151          (https://searchsqlserver.techtarget.com/definition/database).  A
152          database (DB) is a collection of related data like schemata, tables,
153          views, functions, procedures and other related objects.
```

```
154
155      3.2. `AP1` (`SF21SQL1001.AP1`) is a schema
156           (https://searchsqlserver.techtarget.com/definition/schema) in database
157           `SF21SQL1001`.  A schema is a collection of tables, views, functions
158           and other related objects often used for organizational purposes only.
159
160      3.3. `ContactUpdates` (`SF21SQL1001.AP1.ContactUpdates`) is a table
161           (https://whatis.techtarget.com/definition/table) in schema `AP1`
162           calling the schema because the schema is not `dbo` (`database owner`
163           default schema in T-SQL, which does not need to be called when used).
164           A table is a collection of columns/fields and rows/records.
165
166      3.4. `VendorID` (`SF21SQL1001.AP1.ContactUpdates.VendorID`) is a
167           column/field (https://searchoracle.techtarget.com/definition/field) in
168           table `AP1.ContactUpdates`.  A column/field is an allocation of data
169           in a record/row.
170
171           This column stores the row identifier for the table.
172
173           It is best practice for a row identifier (usually an integer, a whole
174           number) to be a unique identifier, preferably not related to the rest
175           of the data in the row.
176
177      3.5. A record/row (https://searchoracle.techtarget.com/definition/record)
178           is a collection of related data
179           (https://searchdatamanagement.techtarget.com/definition/data), not
180           referred to with a name but rather its row identifier or position in
181           the table.
182
183   4. As a quick review, in SQL, in order to retrieve data, we use a `SELECT`
184      statement where the simplest syntax is the following.
185
186                      SELECT field1, field2 ...
187                      FROM table1;
188
189      4.1. In the example below, we retrieve all columns (fields) and all rows
190           (records) from `AP1.ContactUpdates` calling each one of the columns.
191   ********************************************************************** */
192
193  SELECT VendorID,
194    VendorName,
195    VendorAddress1,
196    VendorAddress2,
197    VendorCity,
198    VendorState,
199    VendorZipCode,
200    VendorPhone,
201    VendorContactLName,
202    VendorContactFName,
203    DefaultTermsID,
204    DefaultAccountNo
205  FROM AP1.ContactUpdates;
```

```
206
207
208 /* ********************************************************************
209     4.2. In the example below, we retrieve all columns (fields) and all rows
210          (records) from `AP1.Vendors` using wild card `*` (read as `all`).
211  ******************************************************************** */
212
213 SELECT *                                        -- read as `all`
214 FROM AP1.Vendors;
215
216
217 /* ********************************************************************
218     4.3. In the example below, we retrieve all columns and rows from tables
219          `AP1.ContactUpdates` and `AP1.Vendors` using wild card `*` (read as
220          `all`).
221
222          Since we are calling a second table, we have to `JOIN` them on the
223          common field (data, value) as this indicates the relation between the
224          two tables.
225
226          We are going to cover three `JOIN` alternatives.  Each `JOIN` returns
227          a different population.
228
229          `INNER JOIN` returns shared data (rows) between the two tables.  Any
230          data found only in one table is not returned.
231
232          `LEFT [OUTTER] JOIN` returns all the data (rows) from the left table
233          (first table called, `AP1.ContactUpdates`) and any shared data (rows)
234          in the right table (second table called, `AP1.Vendors`).  No data is
235          ignored.
236
237          `RIGHT [OUTTER] JOIN` returns all the data (rows) from the right table
238          (second table called, `AP1.Vendors`) and any shared data (rows) in the
239          left table (first table called, `AP1.ContactUpdates`).  Note that this
240          may be confusing for anyone reading your code.  You might want to avoid
241          using `RIGHT JOIN`.
242
243          4.3.1. As mentioned, in the example below, we retrieve all shared data
244                 (rows) from tables `AP1.ContactUpdates` and `AP1.Vendors`.
245  ******************************************************************** */
246
247 SELECT *
248 FROM AP1.ContactUpdates                         -- 1. all shared data (rows)
249                                                 --    from `AP1.ContactUpdates`
250 INNER JOIN AP1.Vendors                          -- 2. all shared data (rows)
251                                                 --    from `AP1.Vendors`
252   ON AP1.ContactUpdates.VendorID = AP1.Vendors.VendorID;
253                                                 -- 3. on common data (rows)
254                                                 --    `VendorID`
255
256
257 /* ********************************************************************
```

```
258          4.3.2. In the example below, we retrieve all data (rows) from table
259                 `AP1.ContactUpdates` and any shared data (rows) from
260                 `AP1.Vendors`.
261   ********************************************************************** */
262
263  SELECT *
264  FROM AP1.ContactUpdates                        -- 1. all data (rows) from main
265                                                 --    table
266                                                 --    `AP1.ContactUpdates`
267  LEFT JOIN AP1.Vendors                          -- 2. any shared data (rows)
268                                                 --    from `AP1.Vendors`
269    ON AP1.ContactUpdates.VendorID = AP1.Vendors.VendorID;
270                                                 -- 3. on common data (rows)
271                                                 --    `VendorID`
272
273
274  /* ********************************************************************
275          4.3.3. In the example below, we retrieve all data (rows) from table
276                 `AP1.Vendors` and any shared data (rows) from
277                 `AP1.ContactUpdates`.
278   ********************************************************************** */
279
280  SELECT *
281  FROM AP1.ContactUpdates                        -- 1. any shared data (rows)
282                                                 --    from `AP1.ContactUpdates`
283  RIGHT JOIN AP1.Vendors                         -- 2. all data (rows) from main
284                                                 --    table `AP1.Vendors`
285    ON AP1.ContactUpdates.VendorID = AP1.Vendors.VendorID;
286                                                 -- 3. on common data (rows)
287                                                 --    `VendorID`
288
289
290  /* ********************************************************************
291      4.4. In the example below, we retrieve all columns (fields) and rows
292           (columns) from `AP1.Vendors` calling each one of the columns.
293
294           4.4.1. We use `AS` to assign aliases to ``create a temporary name for
295                  columns or tables.``
296
297                  We can use aliases on columns ``to make column headings in your
298                  result set easier to read.``
299
300                  We can use aliases on tables ``to shorten your SQL to make it
301                  easier to read or when you are performing a self join (ie:
302                  listing the same table more than once in the FROM clause).``
303                  https://techonthenet.com/sql_server/alias.php
304
305           4.4.2. We also use some functions
306                  (https://techonthenet.com/sql_server/functions/index_alpha.php).
307
308                  CONCAT()    allows you to concatenate strings together
309                              https://techonthenet.com/sql_server/functions/
```

```
                    concat.php
                            https://techonthenet.com/sql_server/functions/
                    concat2.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
 ********************************************************************** */

SELECT VendorID,
  UPPER(VendorName) AS VendorName,              -- 1. using an alias (`AS`)
                                                --    since losing column name
                                                --    with when using function
                                                --    `UPPER()` to make all
                                                --    characters lower upper
                                                --    case
  CONCAT (
```

```
353      VendorAddress1,
354      ' ',
355      VendorAddress2
356      ) AS VendorAddress,                          -- 2. using an alias (`AS`)
357                                                   --    since losing column name
358                                                   --    with when using function
359                                                   --    `CONCAT()` to concatenate
360                                                   --    (to put strings together)
361    LOWER(VendorCity) AS VendorCity,               -- 3. using an alias (`AS`)
362                                                   --    since losing column name
363                                                   --    with when using function
364                                                   --    `LOWER()` to make all
365                                                   --    characters lower upper
366                                                   --    case
367    RIGHT(VendorCity, 4) AS VendorCityRight,       -- 4. using an alias (`AS`)
368                                                   --    since losing column name
369                                                   --    with when using function
370                                                   --    `RIGHT()` to retrieve
371                                                   --    four (4) characters from
372                                                   --    the right
373    LEFT(VendorCity, 3) AS VendorCityLeft,         -- 5. using an alias (`AS`)
374                                                   --    since losing column name
375                                                   --    with when using function
376                                                   --    `LEFT()` to retrieve
377                                                   --    three (3) characters from
378                                                   --    the left
379    SUBSTRING(VendorCity, 3, 4) AS VendorCitySubstring,
380                                                   -- 6. using an alias (`AS`)
381                                                   --    since losing column name
382                                                   --    with when using function
383                                                   --    `SUBSTRING()` to retrieve
384                                                   --    four (4) characters
385                                                   --    starting from the the
386                                                   --    third (3rd) character
387    LEN(VendorCity) AS VendorCityLen,              -- 7. using an alias (`AS`)
388                                                   --    since losing column name
389                                                   --    with when using function
390                                                   --    `LEN()` to retrieve the
391                                                   --    length of string in field
392    REPLACE(VendorState, 'CA', 'California') AS VendorState,
393                                                   -- 8. using an alias (`AS`)
394                                                   --    since losing column name
395                                                   --    with when using function
396                                                   --    `REPLACE()` to replace
397                                                   --    string `CA` with string
398                                                   --    `California`
399    VendorZipCode,
400    VendorPhone,
401    VendorContactLName AS 'Vendor Contact Last Name',
402                                                   -- 9. using an alias (`AS`)
403                                                   --    to change the name of
404                                                   --    column
```

```
405    VendorContactFName AS 'Vendor Contact First Name',
406                                                 -- 10. using an alias (`AS`)
407                                                 --     to change the name of
408                                                 --     column
409    DefaultTermsID,
410    DefaultAccountNo
411  FROM AP1.Vendors;
412
413
414  /* ***************************************************************************
415   3. LAB 1
416      Write a query calling all shared fields (`INNER JOIN`) from `AP1.Invoices`,
417      `AP1.Terms` and `AP1.Vendors`.
418      * Delete or rename the duplicate name of the columns.
419   *************************************************************************** */
420
421  SELECT AP1.Invoices.InvoiceID,
422    AP1.Invoices.VendorID,
423    AP1.Invoices.InvoiceNumber,
424    AP1.Invoices.InvoiceDate,
425    AP1.Invoices.InvoiceTotal,
426    AP1.Invoices.PaymentTotal,
427    AP1.Invoices.CreditTotal,
428    AP1.Invoices.TermsID,
429    AP1.Invoices.InvoiceDueDate,
430    AP1.Invoices.PaymentDate,
431    -- AP1.Terms.TermsID,                       -- 1. duplicate column name
432                                                --    (`TermsID`), which can
433                                                --    be removed (commented
434                                                --    out, in this case)
435                                                --    without affecting the
436                                                --    query output; could also
437                                                --    be renamed
438    AP1.Terms.TermsDescription,
439    AP1.Terms.TermsDueDays,
440    -- AP1.Vendors.VendorID,                    -- 2. duplicate column name
441                                                --    (`VendorID`), which can
442                                                --    be removed (commented
443                                                --    out, in this case)
444                                                --    without affecting the
445                                                --    query output; could also
446                                                --    be renamed
447    AP1.Vendors.VendorName,
448    AP1.Vendors.VendorAddress1,
449    AP1.Vendors.VendorAddress2,
450    AP1.Vendors.VendorCity,
451    AP1.Vendors.VendorState,
452    AP1.Vendors.VendorZipCode,
453    AP1.Vendors.VendorPhone,
454    AP1.Vendors.VendorContactLName,
455    AP1.Vendors.VendorContactFName,
456    AP1.Vendors.DefaultTermsID,
```

```
457    AP1.Vendors.DefaultAccountNo
458  FROM AP1.Invoices                                -- 3. from table `AP1.Invoices`
459  INNER JOIN AP1.Terms                             -- 4. `INNER JOIN` to retrieve
460                                                   --    data in the first (left)
461                                                   --    table (`AP1.Invoices`)
462                                                   --    that is also in the
463                                                   --    second (right) table
464                                                   --    (`AP1.Terms`)
465    ON AP1.Invoices.TermsID = AP1.Terms.TermsID    -- 5. `ON` two fields with the
466                                                   --    same values/data and the
467                                                   --    same name (`TermsID`);
468                                                   --    specifying the relation
469                                                   --    between tables
470                                                   --    `AP1.Invoices` and
471                                                   --    `AP1.Terms`
472  INNER JOIN AP1.Vendors                           -- 6. `INNER JOIN` to retrieve
473                                                   --    data in the second (left)
474                                                   --    table (`AP1.Terms`) that
475                                                   --    is also in the third
476                                                   --    (right) table
477                                                   --    (`AP1.Vendors`)
478    ON AP1.Vendors.VendorID = AP1.Invoices.VendorID;
479                                                   -- 7. `ON` two fields with the
480                                                   --    same values/data and the
481                                                   --    same name (`VendorID`);
482                                                   --    specifying the relation
483                                                   --    between tables
484                                                   --    `AP1.Vendors` and
485                                                   --    `AP1.Invoices`
486
487  /* ***************************************************************************
488   https://folvera.commons.gc.cuny.edu/?p=988
489   ************************************************************************** */
```