```
/* ************************************************************************
        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 #9 (2021/12/06):  CREATING DATABASE OBJECTS

  1. Parameters, user-defined functions and stored procedures
 ************************************************************************

 1. LAB #10 (Procedures)
    1.1. Understanding that the following is the structure for a procedure
         (https://techonthenet.com/sql_server/procedures.php)

                    CREATE PROCEDURE procedure_name [@input_param data_type]
                    AS
                    BEGIN
                      [DECLARE @output_param data_type
                      SET @output_param = some_value]
                      executable_code
                    END;

         that we EXECUTE (EXEC) in order for it to run,

                    EXEC procedure_name [@input_param]

         write a procedure `strings2_sp` in schema `lab10` in database `ace` to
         concatenate two (2) strings with an empty space (` `) between the two
         strings.

         HINT: two (2) input parameters to produce one (1) output parameter
         with the minimal size of the sum of the sizes of the first input
         parameter and the second input parameter

    1.2. To test that procedure `lab10.strings2_sp` works, execute the
         procedure passing first name and last name.

         HINT: EXEC procedure_name(@in_param1, @in_param2)
 ************************************************************************ */

CREATE SCHEMA lab10;

CREATE PROCEDURE lab10.string2_sp @in_string1 VARCHAR(50),
  @in_string2 VARCHAR(50)
AS
BEGIN
  DECLARE @out_string VARCHAR(101)              -- to accept VARCHAR(50) for
                                                -- `@in_string1`, VARCHAR(1)
                                                -- for a space + VARCHAR(50)
                                                -- for `@in_string2`
  SET @out_string = CONCAT (
```

```
53        @in_string1,
54        ' ',
55        @in_string2
56        )
57    PRINT @out_string
58  END;
59
60
61  /* ************************************************************************
62      1.3. Then we execute procedure `lab10.string2_sp` passing two (2) values.
63           Passing more or fewer values will return an error.
64
65                      Msg 201, Level 16, State 4, Procedure lab10.string2_sp,
66                      Line 0 [Batch Start Line 53]
67                      Procedure or function 'string2_sp' expects parameter
68                      '@in_string2', which was not supplied.
69   ************************************************************************ */
70
71  EXEC lab10.string2_sp 'John', 'Smith';
72
73
74  /* ************************************************************************
75   2. LAB #11 (Functions)
76      2.1. Understanding that the following is the structure for a function
77           (https://techonthenet.com/sql_server/functions.php)
78
79                      CREATE FUNCTION funtion_name (@input_param data_type)
80                      RETURNS data_type
81                      AS
82                      BEGIN
83                        DECLARE @output_param data_type
84                        SET @output_param = some_value
85                        executable_code
86                        RETURN output_param
87                      END;
88
89           that affects a field or other value,
90
91                      funtion_name(field)
92
93           write a function `strings2_udf()` in schema `lab11` in database `ace`
94           to concatenate two (2) strings with an empty space (` `) between the
95           two strings.
96
97           HINT: two (2) input parameters to produce one (1) output parameter
98           with the minimal size of the sum of the sizes of the first input
99           parameter and the second input parameter
100
101     2.2. To test that function `lab11.strings2_udf()` works, write a query
102          calling all values from `AP1.ContactUpdates` using function
103          `lab11.strings2_udf()` on `first_name` and `last_name`.
104
```

```
105           HINT: SELECT function_name(@in_param1, @in_param2)
106   ************************************************************************ */
107
108   CREATE SCHEMA lab11;
109
110   CREATE FUNCTION lab11.string2_udf (
111     @in_string1 VARCHAR(50),
112     @in_string2 VARCHAR(50)
113     )
114   RETURNS VARCHAR(101)                              -- same datatype and size as
115                                                     -- `@out_string`, in this case
116   AS
117   BEGIN
118     DECLARE @out_string VARCHAR(101)                -- to accept VARCHAR(50) for
119                                                     -- `@in_string1`, VARCHAR(1)
120                                                     -- for a space + VARCHAR(50)
121                                                     -- for `@in_string2`
122     SET @out_string = CONCAT (
123         @in_string1,
124         ' ',
125         @in_string2
126         )
127     RETURN @out_string
128   END;
129
130   /* ***************************************************************************
131       2.3. Then we use function `lab11.string2_udf` passing two (2) values.  Note
132            that passing more or fewer values will return an error.
133
134                     Msg 313, Level 16, State 2, Line 101
135                     An insufficient number of arguments were supplied for the
136                     procedure or function lab11.string2_udf.
137    ************************************************************************ */
138
139   SELECT lab11.string2_udf('John', 'Smith');
140
141
142   /* ***************************************************************************
143    3. In the example below, we make a function to dress up phone numbers as
144       `(xxx) xxx-xxxx` in schema `lab12`.
145    ************************************************************************ */
146
147   CREATE SCHEMA lab12;
148
149   CREATE FUNCTION lab12.phones_udf (@in_phone VARCHAR(15))
150   RETURNS VARCHAR(15)                               -- need to remember that a
151                                                     -- function RETURNS a value
152   AS
153   BEGIN
154     DECLARE @out_phone VARCHAR(15)
155     SET @out_phone = CASE                           -- `CASE` clause to check if
156         WHEN @in_phone IS NOT NULL                  -- `CONCAT` needs to be run
```

```
157         OR @in_phone <> ''
158         OR @in_phone NOT LIKE ('(%)%-%')        -- checking for formatted phone
159         THEN CONCAT (
160             '(',
161             LEFT(@in_phone, 3),
162             ') ',
163             SUBSTRING(@in_phone, 4, 3),
164             '-',
165             RIGHT(@in_phone, 4)
166             )
167       ELSE @in_phone
168       END                                       -- ending/closing `CASE` clause
169   RETURN @out_phone
170 END;                                            -- ending/closing function
171
172
173 /* ***************************************************************************
174     3.1 Then we use function `lab11.string2_udf` passing two (2) values when
175          querying `SF21SQL1001.AP1.Vendors`.
176
177          Since accessing another objects in another database, you need to call
178          the full name the function (`labs.lab11.string2_udf`) and/or the table
179          (`SF21SQL1001.AP1.Vendors`) depending in which database you are in.
180  *************************************************************************** */
181
182 SELECT VendorID,
183   SF21SQL1001.AP1.Vendors.VendorName,
184   labs.lab11.strings2_udf(SF21SQL1001.AP1.Vendors.VendorAddress1,
185     SF21SQL1001.AP1.Vendors.VendorAddress2)
186     AS VendorAddress,                          -- `labs.lab11.strings2_udf`
187                                                -- on `VendorAddress1` and
188                                                -- `VendorAddress2`
189   SF21SQL1001.AP1.Vendors.VendorCity,
190   SF21SQL1001.AP1.Vendors.VendorState,
191   SF21SQL1001.AP1.Vendors.VendorZipCode,
192   labs.lab12.phones_udf(VendorPhone)
193     AS VendorPhone                             -- using function
194                                                -- `labs.lab12.phones_udf`
195 FROM SF21SQL1001.AP1.Vendors;
196
197
198 /* ***************************************************************************
199  4. This marks the end of new material.
200
201     4.1. As a developer, you should have a list of resources -- websites, books
202          or people whom you can contact for help.  The following is only a list
203          of resources -- not a recommendation of goods and/or services.
204
205          Analytics Vidhya (data science community)
206          https://analyticsvidhya.com/
207
208          Apache Spark - Unified Analytics Engine
```

```
209          https://spark.apache.org/
210
211          Apache Spark - Unified Analytics Engine - Spark SQL & DataFrames
212          https://spark.apache.org/sql/
213
214          Azure Cosmos DB
215          https://azure.microsoft.com/en-us/services/cosmos-db/
216
217          Azure SQL - Azure SQL documentation - Microsoft Docs
218          https://docs.microsoft.com/en-us/azure/azure-sql/
219
220          Cockroach Labs - CockroachDB
221          https://cockroachlabs.com/
222
223          DBeaver - Universal Database Tool
224          https://dbeaver.com/
225
226          DBeaver Community (client)
227          https://dbeaver.io/
228
229          EverSQL - SQL Query Optimizer Tool Online
230          https://eversql.com/sql-syntax-check-validator/
231
232          HeidiSQL (client)
233          https://heidisql.com/
234
235          Infrastructure as SQL (iaSQL)
236          https://iasql.com/
237
238          MariaDB (MySQL fork, not related to Oracle)
239          https://mariadb.org/
240
241          Microsoft Azure
242          https://portal.azure.com/
243
244          Microsoft Azure - Quickstart Templates
245          https://azure.microsoft.com/en-us/resources/templates/
246
247          Microsoft Power Automate
248          https://flow.microsoft.com/en-us/desktop/
249
250          Microsoft Power BI (business intelligence)
251          https://powerbi.microsoft.com/
252
253          Microsoft SQL Server
254          https://microsoft.com/en-us/sql-server/
255
256          Microsoft SQL Server - Get Started
257          https://microsoft.com/en-us/sql-server/developer-get-started/
258
259          MongoDB
260          https://mongodb.com/
```

```
261
262         MongoDB Blog
263         https://mongodb.com/blog
264
265         mycli (CLI MariaDB, MySQL & Percona)
266         https://mycli.net/
267
268         MySQL (Oracle)
269         https://mysql.com/
270
271         Oracle
272         http://oracle.com/
273
274         phpMyAdmin (administration tool for MySQL/MariaDB)
275         https://phpmyadmin.net/
276
277         Poor SQL (code formatter)
278         https://poorsql.com/
279
280         PostgreSQL
281         https://postgresql.org/
282
283         Slack - codebar - sql
284         https://app.slack.com/client/T08CJBA82/CHPE04RU7
285
286         SQLite
287         https://sqlite.org/
288
289         SQLZOO
290         https://sqlzoo.net/
291
292         Tech on the Net - SQL Server
293         https://techonthenet.com/sql_server/
294
295         Vespa (big data AI, Oath/Yahoo)
296         http://vespa.ai/
297
298 ************************************************************************
299 https://folvera.commons.gc.cuny.edu/?p=1061
300 ************************************************************************ */
```