```
INTRODUCTION TO STRUCTURED OUERY LANGUAGE FOR DATA ANALYTICS
2
3
                 WS23SQL1001, 2023/04/03 to 2023/05/03
4
               https://folvera.commons.gc.cuny.edu/?cat=33
   ***********************************
5
6
7
    SESSION #2 (2023/04/05): UNDERSTANDING CORE DATABASE CONCEPTS
8
9
    1. Learning history of SQL and basic concepts of the structure of a
10
      relational database
    2. Understanding structured programming
11
    3. Understanding naming convention
12
    4. Understanding basic syntax to query one table
13
   *********************
14
15
16
   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
17
     `Software developers, applications` with a median annual income of $100,080
18
     and 20.9% for `Computer systems analysts` with a median annual income of
19
20
     $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
21
22
23
     +-----+
24
     | Computer Systems Analysts | $88,270 / yr | $42.44 / hr |
25
     +-----+
26
     https://bls.gov/ooh/computer-and-information-technology/computer-systems-
27
     analysts.htm
     +-----+
28
     29
     +-----+
30
     https://bls.gov/ooh/computer-and-information-technology/database-
31
     administrators.htm
     +-----
32
33
     | Web Developers
                                  | $67,990 / yr | $32.69 / hr |
     +-----+
34
     https://bls.gov/ooh/computer-and-information-technology/web-developers.htm
35
     +------
36
     | Operations Research Analysts | $81,390 / yr | $39.13 / hr | +------
37
38
     https://bls.gov/ooh/math/operations-research-analysts.htm
39
     +-----
40
41
42
   2. The following are concepts that we need to know.
43
44
     2.01. SQL (Structured Query Language) is a standardized programming
         language used for managing relational databases and performing
45
46
         various operations on the data in them. Initially created in the
         1970s, SQL is regularly used by database administrators, as well as
47
48
         by developers writing data integration scripts and data analysts
49
         looking to set up and run analytical queries.
         https://searchsqlserver.techtarget.com/definition/SQL
50
```

51

52 The SQL programming language was first developed in the 1970s by IBM 53 researchers Raymond Boyce and Donald Chamberlin. The programming 54 language, known then as SEQUEL, was created following the publishing 55 of Edgar Frank Todd's paper, ``A Relational Model of Data for Large 56 Shared Data Banks, `` in 1970. 57

https://businessnewsdaily.com/5804-what-is-sql.html

58 59

Refer to https://ibm.com/ibm/history/ibm100/us/en/icons/reldb/ for more information on Edgar Frank Todd's paper.

60 61 62

63 64

65

2.02. T-SQL (Transact-SQL) is a set of programming extensions from Sybase and Microsoft that add several features to the Structured Query Language (SQL), including transaction control, exception and error handling, row processing and declared variables. https://searchsqlserver.techtarget.com/definition/T-SQL

66 67 68

69 70

71

72

2.03. Microsoft SQL Server is a relational database management system, or RDBMS, that supports a wide variety of transaction processing, business intelligence and analytics applications in corporate IT environments. It's one of the three market-leading database technologies, along with Oracle Database and IBM's DB2. https://searchsqlserver.techtarget.com/definition/SQL-Server

73 74 75

76

77 78

79

80 81

82

83

2.04. A server is a computer program that provides a service to another computer programs (and its user). In a data center, the physical computer that a server program runs in is also frequently referred to as a server. That machine may be a dedicated server or it may be used for other purposes as well. In the client/server programming model, a server program awaits and fulfills requests from client programs, which may be running in the same or other computers. A given application in a computer may function as a client with requests for services from other programs and also as a server of requests from other programs. https://whatis.techtarget.com/definition/server

84 85 86

87

88 89

90

91

92

93

2.05. A relational database management system (RDBMS) is a collection of programs and capabilities that enable IT teams and others to create, update, administer and otherwise interact with a relational database. Most commercial RDBMSes use Structured Query Language (SQL) to access the database, although SQL was invented after the initial development of the relational model and is not necessary for its use. https://searchdatamanagement.techtarget.com/definition/RDBMSrelational-database-management-system

94 95

96 97

98 99

100

101

2.06. In computer programming, a schema (pronounced SKEE-mah) is the organization or structure for a database. The activity of data modeling leads to a schema. (The plural form is schemata. The term is from a Greek word for ``form`` or ``figure.`` Another word from the same source is ``schematic.``) The term is used in discussing both relational databases and object-oriented databases. The term sometimes seems to refer to a visualization of a structure and

P

```
sometimes to a formal text-oriented description.
102
103
               https://searchsqlserver.techtarget.com/definition/schema
104
105
        2.07. In computer programming, a table is a data structure used to organize
106
               information, just as it is on paper. There are many different types
               of computer-related tables, which work in a number of different ways.
107
108
               The following are examples of the more common types.
               1) In data processing, a table (also called an array) is a organized
109
110
                  grouping of fields. Tables may store relatively permanent data,
                  or may be frequently updated. For example, a table contained in a
111
                  disk volume is updated when sectors are being written.
112
               2) In a relational database, a table (sometimes called a file)
113
114
                  organizes the information about a single topic into rows and
115
                  columns. For example, a database for a business would typically
116
                  contain a table for customer information, which would store
                  customers' account numbers, addresses, phone numbers, and so on as
117
                  a series of columns. Each single piece of data (such as the
118
                  account number) is a field in the table. A column consists of all
119
                  the entries in a single field, such as the telephone numbers of
120
                  all the customers. Fields, in turn, are organized as records,
121
122
                  which are complete sets of information (such as the set of
                  information about a particular customer), each of which comprises
123
124
                  a row. The process of normalization determines how data will be
125
                  most effectively organized into tables.
126
                  https://whatis.techtarget.com/definition/table
127
128
     3. Before we start, we should be familiar with the naming convention used in
        T-SQL (https://searchsqlserver.techtarget.com/definition/T-SQL) using the
129
130
         database for this course.
131
                                                     (server; name depending on the
132
                         PC12345\MSSOLSERVER
133
                                                     machine we are using where
                                                     `PC12345` is the `HOSTNAME` and
134
                                                     `MSSQLSERVER` is the database
135
136
                                                     instance)
137
138
                          +- WS23SQL1001
                                                      (database in server\instance
139
                                                     `PC12345\SQLSERVEREXPRESS`)
140
141
                              +- AP1
                                                     (schema in database
142
                                                     `WS23SQL1001`)
143
144
                                  +- ContactUpdates (table in schema `AP1`)
145
                                                     (column in table
146
                                      +- VendorID
147
                                                     `ContactUpdates`)
148
149
         3.01. Using the structure above, `WS23SQL1001` is the database
150
               (https://searchsqlserver.techtarget.com/definition/database). A
```

database (DB) is a collection of related data like schemata, tables,

views, functions, procedures and other related objects.

152153

151

```
154
        3.02. `AP1` (`WS23SQL1001.AP1`) is a schema
155
              (https://searchsqlserver.techtarget.com/definition/schema) in
156
              database `WS23SQL1001`. A schema is a collection of tables, views,
157
              functions and other related objects often used for organizational
158
              purposes only.
159
        3.03. `ContactUpdates` (`WS23SQL1001.AP1.ContactUpdates`) is a table
160
              (https://whatis.techtarget.com/definition/table) in schema `AP1`
161
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
        3.04. `VendorID` (`WS23SQL1001.AP1.ContactUpdates.VendorID`) is a
167
              column/field (https://searchoracle.techtarget.com/definition/field)
168
              in table `AP1.ContactUpdates`. A column/field is an allocation of
169
              data in a record/row.
170
              This column stores the row identifier for the table.
171
172
              It is best practice for a row identifier (usually an integer, a whole
173
174
              number) to be a unique identifier, preferably not related to the rest
175
              of the data in the row.
176
177
        3.05. 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
     4. In order to retrieve data, we use a `SELECT` statement where the simplest
183
184
        syntax is the following.
185
186
                        SELECT field1, field2 ...
187
                        FROM table1;
188
189
        4.01. 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.
      VendorAddress2,
196
197
      VendorCity,
198
      VendorState,
      VendorZipCode.
199
200
      VendorPhone.
201
      VendorContactLName,
202
      VendorContactFName,
203
      DefaultTermsID,
      DefaultAccountNo
204
205 FROM AP1.ContactUpdates;
```

```
206
207
209
       4.02. 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` as in
214 FROM AP1. Vendors;
                                              -- `SELECT all FROM AP1.Vendors`
215
216
    217
218
       4.03. 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
            Since we are calling a second table, we have to `JOIN` them on the
222
             common field (data, value) as this indicates the relation between the
223
224
            two tables.
225
            We are going to cover three `JOIN` alternatives. Each `JOIN` returns
226
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
            (first table called, `AP1.ContactUpdates`) and any shared data (rows)
233
234
            in the right table (second table called, `AP1.Vendors`). No data is
235
            ignored.
236
            `RIGHT [OUTTER] JOIN` returns all the data (rows) from the right
237
            table (second table called, `AP1.Vendors`) and any shared data (rows)
238
             in the left table (first table called, `AP1.ContactUpdates`). Note
239
240
            that this may be confusing for anyone reading the code. You might
241
            want to avoid using `RIGHT JOIN`.
242
243
            We also use `AS` to assign aliases to ``create a temporary name for
244
            columns or tables.`` We can use aliases on columns ``to make column
            headings in wer result set easier to read. `` We can use aliases on
245
            tables ``to shorten wer SQL to make it easier to read or when we
246
247
             are performing a self join (ie: listing the same table more than once
            in the FROM clause).`
248
249
            https://techonthenet.com/sql_server/alias.php
250
            As mentioned, in the example below, we retrieve all shared data
251
             (rows) from tables `AP1.ContactUpdates` and `AP1.Vendors`.
252
     253
254
255 SELECT *
                                              -- 01. all fields (columns)
                                              -- 02. all shared data (rows)
256 FROM AP1.ContactUpdates
                                                    from `AP1.ContactUpdates`
257
```

```
258 INNER JOIN AP1. Vendors
                                          -- 03. all shared data (rows)
259
                                                from `AP1.Vendors`
260
     ON AP1.ContactUpdates.VendorID = AP1.Vendors.VendorID;
                                          -- 4. on common data (columns)
261
262
                                                `VendorID`
263
264
266
           As an alternative, the code above can also be written using an alias
267
           (`AS`) for each table in order to simplify the code. Note that, if
           we use an alias for a table (for example, `v` for `AP1.Vendors`), we
268
           must use the alias (`v`) when calling the table anywhere else in the
269
           query (`v.VendorID` instead of `AP1.Vendors.VendorID`).
270
    271
272
273 SELECT *
                                          -- 01. all fields (columns)
274 FROM AP1.ContactUpdates AS c
                                          -- 02. all shared data (rows)
                                                from table
275
276
                                                `AP1.ContactUpdates`
                                                using alias `c`
277
278 INNER JOIN AP1. Vendors AS v
                                          -- 03. all shared data (rows)
                                               from table
                                                `AP1.Vendors` using
280
281
                                                alias `v`
282
     ON c.VendorID = v.VendorID;
                                          -- 04. on common data (columns)
283
                                                `VendorTD`
284
285
287
           In the example below, we retrieve all data (rows) from table
           `AP1.ContactUpdates` and any shared data (rows) from `AP1.Vendors`.
288
    289
290
291 SELECT *
                                          -- 01. all fields (columns)
292 FROM AP1.ContactUpdates
                                          -- 02. all data (rows) from
                                                main table
293
                                          --
294
                                                `AP1.ContactUpdates`
295 LEFT JOIN AP1. Vendors
                                          -- 03. any shared data (rows)
296
                                                from secondary table
297
                                                `AP1.Vendors`
298
     ON AP1.ContactUpdates.VendorID = AP1.Vendors.VendorID;
299
                                          -- 04. on common data (columns)
300
                                                `VendorID`
301
302
   303
304
           As an alternative, the code above can also be written using an alias
305
           (`AS`) for each table in order to simplify the code. Note that, if
           we use an alias for a table (for example, `v` for `AP1.Vendors`), we
306
307
           must use the alias (`v`) when calling the table anywhere else in the
           query (`v.VendorID` instead of `AP1.Vendors.VendorID`).
308
          309
```

```
310
311 SELECT *
                                          -- 01. all fields (columns)
                                          -- 02. all data (rows) from
312 FROM AP1.ContactUpdates AS c
313
                                                main table
314
                                                `AP1.ContactUpdates`
315
                                                using alias `c`
316 LEFT JOIN AP1. Vendors AS v
                                          -- 03. any shared data (rows)
                                                from secondary table
318
                                                `AP1.Vendors` using
319
                                                alias `v`
320
    ON c.VendorID = v.VendorID;
321
                                          -- 04. on common data (columns)
322
                                                `VendorID`
323
324
In the example below, we retrieve all data (rows) from table
326
          `AP1.Vendors` and any shared data (rows) from `AP1.ContactUpdates`.
    328
329
330 SELECT *
                                          -- 01. all fields (columns)
331 FROM AP1.ContactUpdates
                                          -- 02. any shared data (rows)
332
                                                from secondary table
333
                                                `AP1.ContactUpdates`
334 RIGHT JOIN AP1. Vendors
                                          -- 03. all data (rows) from
335
                                                main table `AP1.Vendors`
336
     ON AP1.ContactUpdates.VendorID = AP1.Vendors.VendorID;
337
                                          -- 04. on common data (columns)
338
                                                `VendorID`
339
340
As an alternative, the code above can also be written using an alias
343
           (`AS`) for each table in order to simplify the code. Note that, if
344
          we use an alias for a table (for example, `v` for `AP1.Vendors`), we
345
          must use the alias (`v`) when calling the table anywhere else in the
346
          query (`v.VendorID` instead of `AP1.Vendors.VendorID`).
    347
348
349 SELECT *
                                          -- 01. all fields (columns)
                                          -- 02. any shared data (rows)
350 FROM AP1.ContactUpdates AS c
                                                from secondary table
351
352
                                                `AP1.ContactUpdates`
                                          ___
353
                                                using alias `c`
354 RIGHT JOIN AP1. Vendors AS v
                                          -- 03. all data (rows) from
                                                main table `AP1.Vendors`
355
356
                                          --
                                                using alias `v`
                                          -- 04. on common data (columns)
357
    ON c.VendorID = v.VendorID;
358
                                                `VendorID`
359
```

••••	1.000 1.300	. (20230403)	. M3233QL1001 \M3233QL1001_20	230-03.30
362	4.04. In the example below, we retrieve all columns (fields) and rows			
363		(records) from `AP1.Vendors` calling each one of the columns. We also		
364		use some string (array of letters, numbers, symbols, etc.) functions		
365		(https://techonthenet.com/sql_server/functions/index_alpha.php).		
366		, , , , , , , , , , , , , , , , , , ,		
367		CONCAT()	allows we to concatenate s	trings together
368		()		ql_server/functions/concat.php
369			Treeps 1, 7 economence recom, 3	qr_server, runeerons, conederpnp
370		`+`	also allows we to concaten	ate strings together although
371		'	adding NULL returns a NULL	
372				ql_server/functions/concat2.php
			irceps.//techonchenet.com/so	q1_servery runctions/concac2.pnp
373		LEET()	-11	stains form a stains stanting
374		LEFT()		string from a string, starting
375			from the left-most characte	
376			https://techonthenet.com/s	ql_server/functions/left.php
377				
378		LEN()		specified string does not
379				racters at the end the string
380			when calculating the lengt	
381			https://techonthenet.com/s	ql_server/functions/len.php
382				
383		LTRIM()	removes all space characte	rs from the left-hand side of a
384			string	
385			https://techonthenet.com/se	ql_server/functions/ltrim.php
386				
387		LOWER()	converts all letters in the	e specified string to lowercase
388		.,		ql_server/functions/lower.php
389			•	
390		REPLACE()	replaces a sequence of char	racters in a string with another
391			set of characters, not case	
392				ql_server/functions/replace.php
393				
394		RIGHT()	allows we to extract a sub	string from a string, starting
395		()	from the right-most charac	
396				ql_server/functions/right.php
397				q=_ser (e. ) : aee=es) : =8evpp
398		RTRIM()	removes all space character	rs from the right-hand side of a
399		1111 <u>1</u> 11()	string	13 11 0 m ene 1 1gire nana 31 ae 01 a
400				ql_server/functions/rtrim.php
401			receps., / economenciae.com/ so	q1_3cr vcr / runccion3/1 cr im.pnp
402	SUBSTRING() allows we to extract a substring from a string			
403	https://techonthenet.com/sql_server/functions/substring.php			
404			nccps.//techonchenet.com/s	dr_servery rancerons/sanserring.bub
		UDDED/)	convents all letters in th	a specified string to upperson
405		UPPER()		e specified string to uppercase
406	https://techonthenet.com/sql_server/functions/upper.php ***********************************			
407	11 11 11 11 11 11 11 11 11 11 11 11 11	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·
408	CELECT V			
409	SELECT Vei		AC MandanNama	04
410	UPPER(Ve	endorName)	AS VendorName,	01. using an alias (`AS`)
411				since losing column name
412				with when using function
413				`UPPER()` to make all

```
414
                                                              characters lower upper
415
                                                              case
416
       CONCAT (
417
         VendorAddress1,
418
         . .
419
         VendorAddress2
420
         ) AS VendorAddress,
                                                       -- 02. using an alias (`AS`)
421
                                                              since losing column name
422
                                                              with when using function
423
                                                              `CONCAT()` to
                                                       ___
424
                                                              concatenate (to put two
425
                                                              or more strings
                                                              together)
426
427
       LOWER(VendorCity) AS VendorCity,
                                                       -- 03. using an alias (`AS`)
428
                                                              since losing column name
429
                                                              with when using function
430
                                                              `LOWER()` to make all
431
                                                              characters lower upper
432
                                                              case
433
       RIGHT(VendorCity, 4) AS VendorCityRight,
                                                       -- 04. using an alias (`AS`)
434
                                                              since losing column name
                                                       ___
435
                                                              with when using function
                                                              `RIGHT()` to retrieve
436
437
                                                              four (4) characters from
438
                                                       --
                                                              the right
439
       LEFT(VendorCity, 3) AS VendorCityLeft,
                                                       -- 05. using an alias (`AS`)
440
                                                              since losing column name
441
                                                              with when using function
                                                       ___
442
                                                              `LEFT()` to retrieve
443
                                                              three (3) characters
444
                                                              from the left
445
       SUBSTRING(VendorCity, 3, 4) AS VendorCitySubstring,
446
                                                       -- 06. using an alias (`AS`)
447
                                                              since losing column name
448
                                                              with when using function
                                                       --
449
                                                       --
                                                              `SUBSTRING()` to
450
                                                              retrieve four (4)
451
                                                              characters starting from
452
                                                              the third (3rd)
453
                                                              character
                                                       -- 07. using an alias (`AS`)
454
       LEN(VendorCity) AS VendorCityLen,
455
                                                              since losing column name
456
                                                              with when using function
                                                       ___
457
                                                              `LEN()` to retrieve the
458
                                                              length of string in
459
                                                              field
460
       REPLACE(VendorState, 'CA', 'California')
461
                                                       -- 08. using an alias (`AS`)
462
                                                              since losing column name
463
                                                              with when using function
                                                       ___
464
                                                              `REPLACE()` to replace
                                                              string `CA` with string
465
```

```
466
                                              `California`
467
     VendorZipCode,
468
     VendorPhone,
     VendorContactLName AS 'Vendor Contact Last Name',
469
                                         -- 09. using an alias (`AS`)
470
471
                                              to change the name of
472
                                              column; not a good idea
473
                                              to have two-word names
     VendorContactFName AS 'Vendor Contact First Name',
474
475
                                         -- 10. using an alias (`AS`)
476
                                              to change the name of
477
                                              column; not a good idea
478
                                              to have two-word names
479
     DefaultTermsID,
480
     DefaultAccountNo
481 FROM AP1. Vendors;
482
483
485
    https://folvera.commons.gc.cuny.edu/?p=1209
    486
```