

JSON DATA TYPE

Database Management Systems

MSCS_542L_256_23S

Super Six



Marist College

School of Computer Science and Mathematics

Submitted To: Dr. Reza Sadeghi

Spring 2023

Report of JSON DATA TYPE SQL Files

Team Name

Super Six

Team members

Pradeep Reddy Baireddy.	pradeepreddy.baireddy1@marist.edu (Team Head)
Prajakta kshirsagar	Prajakta.kshirsagar1@marist.edu (Team member)
Deepthi Niharika Gadepalli	DeepthiNiharika.Gadepalli1@marist.edu (Team member)
Bharadwaja Thota	bharadwaja.thota1@marist.edu (Team member)
Pradeep Reddy Macha	pradeepreddy.macha1@marist.edu (Team member)
Chakradhar Reddy Marepally	ChakradharReddy.Marepally1@marist.edu (Team member)

Table of Contents

Creation:.....	3
Insertion:	3
Selection and Search:.....	5
Merge:.....	8
Other Functions:	10
GitHub repository address:	10

Creation:

We created the JSON data type using the command “create table J_table (json_col1 JSON, task varchar(100))”. We use the keyword JSON as data type for antialiasing the attribute.

The screenshot shows the SQL Developer interface with a query window titled 'Query 1' containing the following SQL code:

```

1 create database IF NOT EXISTS JSON_Dtype;
2 use JSON_Dtype;
3
4 -- creating table J_table for JSON data type demo
5
6 create table J_table (json_col1 JSON, task varchar(100));
7
8 select * from J_table;

```

The 'Result Grid' shows the table structure with columns 'json_col1' and 'task'. The 'Output' pane displays the execution log:

#	Time	Action	Message	Duration / Fetch
2	02:06:00	drop database JSON_Dtype	1 row(s) affected	0.110 sec
3	02:06:43	create database IF NOT EXISTS JSON_Dtype	1 row(s) affected	0.015 sec
4	02:06:43	use JSON_Dtype	0 row(s) affected	0.000 sec
5	02:06:43	create table J_table (json_col1 JSON, task varchar(100))	0 row(s) affected	0.047 sec
6	02:06:43	select * from J_table LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

Insertion:

Direct Insertion

The screenshot shows the SQL Developer interface with a query window titled 'Query 1' containing the following SQL code:

```

8 -- Insertion of JSON data type values into table (Direct Insertion)
9
10 -- JSON object insertion
11 Insert into J_table(json_col1, task) values('{"key1": "value1", "key2": "value2"}', 'Direct Insertion');
12 -- JSON array insertion
13 Insert into J_table(json_col1, task) values('[3, "string1"]', 'Direct Insertion');
14
15 -- display final table
16
17 select * from J_table;

```

The 'Result Grid' shows the table structure with columns 'json_col1' and 'task'. The 'Output' pane displays the execution log:

#	Time	Action	Message	Duration / Fetch
8	02:09:00	Insert into J_table(json_col1, task) values('{"key1": "value1", "key2": "value2"}', 'Direct Insertion')	1 row(s) affected	0.016 sec
9	02:09:00	select * from J_table LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Insertion using functions

Query 1 | Creation_Insertion* | SELECTION_AND_SEARCHING | OTHER_FUNCTIONS | MERGE*

Limit to 1000 rows

```

11 • Insert into J_table(json_col1, task) values('{"key1":"value1","key2":"value2"}','Direct Insertion');
12 -- JSON array insertion
13 • Insert into J_table(json_col1, task) values('[3,"string1"]','Direct Insertion');
14
15 -- Insertion of JSON data type values into table (Insertion using functions)
16
17 • Insert into J_table(json_col1, task) values(JSON_object("key3","value3","key4","value4'),'Insertion through json method
18 • Insert into J_table(json_col1, task) values(JSON_array("string2",3,{"key5":"value5","key6":"value6"}),'Insertion throu
19
20 -- display final table
21
22 • select * from J_table;

```

Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help

json_col1	task
["key1": "value1", "key2": "value2"]	Direct Insertion
[3, "string1"]	Direct Insertion
["key3": "value3", "key4": "value4"]	Insertion through json method
["string2", 3, {"key5": "value5", "key6": "value6"}]	Insertion through json method

J_table 3 x | Read Only | Context Help | Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
11	02:11:44	Insert into J_table(json_col1, task) values(JSON_array("string2",3,{"key5":"value5","key6":...	1 row(s) affected	0.016 sec
12	02:11:54	select * from J_table LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Nested Insertion

Query 1 | Creation_Insertion* | SELECTION_AND_SEARCHING | OTHER_FUNCTIONS | MERGE*

Limit to 1000 rows

```

11 • Insert into J_table(json_col1, task) values('{"key1":"value1","key2":"value2"}','Direct Insertion');
12 -- JSON array insertion
13 • Insert into J_table(json_col1, task) values('[3,"string1"]','Direct Insertion');
14
15 -- Insertion of JSON data type values into table (Insertion using functions)
16
17 • Insert into J_table(json_col1, task) values(JSON_object("key3","value3","key4","value4'),'Insertion through json method
18 • Insert into J_table(json_col1, task) values(JSON_array("string2",3,{"key5":"value5","key6":"value6"}),'Insertion throu
19
20 -- Nested Insertion
21
22 -- JSON object in an array

```

Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help

json_col1	task
[3, "string1"]	Direct Insertion
["key3": "value3", "key4": "value4"]	Insertion through json method
["string2", 3, {"key5": "value5", "key6": "value6"}]	Insertion through json method
[{"key1": "value1", "key2": "value2", "string2": 3}]	Nested type-1
[{"key1": ["string3", 6], "key2": "value2"}]	Nested type-2

J_table 4 x | Read Only | Context Help | Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
14	02:14:01	Insert into J_table(json_col1, task) values([{"key1": ["string3", 6], "key2": "value2"}], "Nested t...	1 row(s) affected	0.000 sec
15	02:14:01	select * from J_table LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Selection and Search:

JSON_EXTRACT()

Query 1 Creation_Insertion SELECTION_AND_SEARCHING... OTHER_FUNCTIONS MERGE*

Limit to 1000 rows

```

1 -- Search and Selection JSON documents using functions
2
3 -- JSON_EXTRACT() FUNCTION FOR SEARCH
4
5 -- searching for values with key as 'key1'
6 • select json_col1, JSON_EXTRACT(json_col1,'$.key1') from J_table;

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

json_col1	JSON_EXTRACT(json_col1,'\$.key1')
{ "key1": "value1", "key2": "value2" }	"value1"
[3, "string1"]	NULL
{ "key3": "value3", "key4": "value4" }	NULL
["string2", 3, { "key5": "value5", "key6": "value6" }]	NULL
{ "key1": "value1", "key2": "value2", "string2": 3 }	NULL

Result 1 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
15	02:14:01	select * from J_table LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
16	02:16:53	select json_col1, JSON_EXTRACT(json_col1,'\$.key1') from J_table LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Query 1 Creation_Insertion SELECTION_AND_SEARCHING... OTHER_FUNCTIONS MERGE*

Limit to 1000 rows

```

1 -- Search and Selection JSON documents using functions
2
3 -- JSON_EXTRACT() FUNCTION FOR SEARCH
4
5 • select json_col1,
6
7 -- searching for values with key as 'key1'
8 JSON_EXTRACT(json_col1,'$.key1'),
9 -- searching for values in an array with key as 'key1' and index of object is 0
10 JSON_EXTRACT(json_col1,'$[0].key1')
11
12 from J_table;

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

json_col1	JSON_EXTRACT(json_col1,'\$.key1')	JSON_EXTRACT(json_col1,'\$[0].key1')
[3, "string1"]	NULL	NULL
{ "key3": "value3", "key4": "value4" }	NULL	NULL
["string2", 3, { "key5": "value5", "key6": "value6" }]	NULL	NULL
{ "key1": "value1", "key2": "value2", "string2": 3 }	"value1"	
{ "key1": "string3", "key2": "value2" }	"string3", 6]	"string3", 6]

Result 2 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
16	02:16:53	select json_col1, JSON_EXTRACT(json_col1,'\$.key1') from J_table LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
17	02:20:42	select json_col1, -- searching for values with key as 'key1' JSON_EXTRACT(json_col1,'\$.k...	6 row(s) returned	0.000 sec / 0.000 sec

JSON_INSERT()

Query 1 Creation_Insertion SELECTION_AND_SEARCHING... OTHER_FUNCTIONS MERGE*

Limit to 1000 rows

```

4
5 -- JSON_EXTRACT() FUNCTION FOR SEARCH
6 -- searching for values with key as 'key1'
7 JSON_EXTRACT(json_col1, '$.key1'),
8 -- searching for values in an array with key as 'key1' and index of object is 0
9 JSON_EXTRACT(json_col1, '$[0].key1'),
10
11 -- JSON_INSERT() function
12 -- insert new object with key 'keynew' and value 'newval'
13 JSON_INSERT(json_col1, '$.keynew', "newval")
14
15 from J_table;

```

Result Grid

json_col1	JSON_EXTRACT(json_col1, '\$.key1')	JSON_EXTRACT(json_col1, '\$[0].key1')	JSON_INSERT(json_col1, '\$.keynew', 'newval')
{'key1': 'value1', 'key2': 'value2'}	'value1'	'value1'	{'key1': 'value1', 'key2': 'value2', 'keynew': 'newval'}
[3, 'string1']	NULL	NULL	[3, 'string1']
{'key3': 'value3', 'key4': 'value4'}	NULL	NULL	{'key3': 'value3', 'key4': 'value4', 'keynew': 'newval'}
['string2', 3, {'key5': 'value5', 'key6': 'value6'}]	NULL	NULL	['string2', 3, {'key5': 'value5', 'key6': 'value6'}]
[{'key1': 'value1', 'key2': 'value2'}, 'string2', 3]	NULL	'value1'	[{'key1': 'value1', 'key2': 'value2'}, 'string2', 3]

Result 4 x

Output

Action Output

#	Time	Action	Message
✓ 18	02:22:28	select json_col1. -- JSON_EXTRACT() FUNCTION FOR SEARCH -- searching for values wit...	6 row(s) returned
✓ 19	02:24:11	select json_col1. -- JSON_EXTRACT() FUNCTION FOR SEARCH -- searching for values wit...	6 row(s) returned

Automat disabled. manually current c toggle

JSON_SET()

Query 1 Creation_Insertion SELECTION_AND_SEARCHING... OTHER_FUNCTIONS MERGE*

Limit to 1000 rows

```

8 -- searching for values in an array with key as 'key1' and index of object is 0
9 -- JSON_EXTRACT(json_col1, '$[0].key1'),
10
11 -- JSON_INSERT() function
12 -- insert new object with key 'keynew' and value 'newval'
13 -- JSON_INSERT(json_col1, '$.keynew', "newval"),
14
15 -- JSON_SET() function
16 -- set object with key 'key1' as 'val3'
17 JSON_SET(json_col1, '$.key1', "val3")
18
19 from J_table;

```

Result Grid

json_col1	JSON_SET(json_col1, '\$.key1', 'val3')
{'key1': 'value1', 'key2': 'value2'}	{'key1': 'val3', 'key2': 'value2'}
[3, 'string1']	[3, 'string1']
{'key3': 'value3', 'key4': 'value4'}	{'key1': 'val3', 'key3': 'value3', 'key4': 'valu...
['string2', 3, {'key5': 'value5', 'key6': 'value6'}]	['string2', 3, {'key5': 'value5', 'key6': 'V...
[{'key1': 'value1', 'key2': 'value2'}, 'string2', 3]	[{'key1': 'value1', 'key2': 'value2'}, 'string2', 3]

Result 8 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 22	02:27:02	select json_col1. -- JSON_EXTRACT() FUNCTION FOR SEARCH -- searching for values wit...	6 row(s) returned	0.000 sec / 0.00
✓ 23	02:28:18	select json_col1. -- JSON_EXTRACT() FUNCTION FOR SEARCH -- searching for values wit...	6 row(s) returned	0.000 sec / 0.00

Automatic context disabled. Use the to manually get help i current caret positio toggle automatic

JSON_REMOVE()

Query 1: Creation_Insertion, SELECTION_AND_SEARCHING, OTHER_FUNCTIONS, MERGE*

```

13 -- JSON_INSERT(json_col1, '$.keynew', "newval1"),
14
15 -- JSON_SET() function
16 -- set object with key 'key1' as 'val3'
17 -- JSON_SET(json_col1, '$.key1', "val3")
18
19 -- JSON_REMOVE() function
20 -- remove value of first json object of array
21 JSON_REMOVE(json_col1, '$[0]')
22
23
24 from j_table;

```

Result Grid:

json_col1	JSON_REMOVE(json_col1, '\$[0]')
{'key1': 'value1', 'key2': 'value2'}	{'key1': 'value1', 'key2': 'value2'}
[3, 'string1']	[3, 'string1']
{'key3': 'value3', 'key4': 'value4'}	{'key3': 'value3', 'key4': 'value4'}
[string2, 3, {'key5': 'value5', 'key6': 'value6'}]	[3, {'key5': 'value5', 'key6': 'value6'}]
[{'key1': 'value1', 'key2': 'value2', 'string2': 3}]	[string2, 3]

Result 9 x

Output:

#	Time	Action	Message	Duration / Fetch
23	02:28:18	select json_col1, --JSON_EXTRACT() FUNCTION FOR SEARCH -- searching for values wit...	6 row(s) returned	0.000 sec / 0.000 sec
24	02:29:55	select json_col1, --JSON_EXTRACT() FUNCTION FOR SEARCH -- searching for values wit...	6 row(s) returned	0.000 sec / 0.000 sec

Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help

JSON_REPLACE()

Query 1: Creation_Insertion, SELECTION_AND_SEARCHING, OTHER_FUNCTIONS, MERGE*

```

17 -- JSON_SET(json_col1, '$.key1', "val3")
18
19 -- JSON_REMOVE() function
20 -- remove value of first json object of array
21 -- JSON_REMOVE(json_col1, '$[0]')
22
23 -- JSON_REPLACE() function
24 -- replace value of object with key as key1 with val3
25 JSON_REPLACE(json_col1, '$.key1', "val3")
26
27
28 from j_table;

```

Result Grid:

json_col1	JSON_REPLACE(json_col1, '\$.key1', 'val3')
{'key1': 'value1', 'key2': 'value2'}	{'key1': 'val3', 'key2': 'value2'}
[3, 'string1']	[3, 'string1']
{'key3': 'value3', 'key4': 'value4'}	{'key3': 'value3', 'key4': 'value4'}
[string2, 3, {'key5': 'value5', 'key6': 'value6'}]	[string2, 3, {'key5': 'value5', 'key6': 'value6'}]
[{'key1': 'value1', 'key2': 'value2', 'string2': 3}]	[{'key1': 'value1', 'key2': 'value2', 'string2': 3}]

Result 10 x

Output:

#	Time	Action	Message	Duration / Fetch
24	02:29:55	select json_col1, --JSON_EXTRACT() FUNCTION FOR SEARCH -- searching for values wit...	6 row(s) returned	0.000 sec / 0.000 sec
25	02:31:42	select json_col1, --JSON_EXTRACT() FUNCTION FOR SEARCH -- searching for values wit...	6 row(s) returned	0.000 sec / 0.000 sec

Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help

Merge:

JSON_MERGE_PRESERVE()

Merging JSON data type columns with JSON array and vice versa

Automatic context help disabled. Use the toolbar manually get help for current caret position or toggle automatic help

```

1  -- Merging JSON attribute with other JSON attributes
2
3  -- Using JSON_MERGE_PRESERVE()
4
5  • select json_col1,
6
7  -- merging JSON datatype columns with JSON array and vice versa
8  JSON_MERGE_PRESERVE(json_col1, '[1, 2]'),
9  JSON_MERGE_PRESERVE('[1, 2]', json_col1)
10
11 from J_table;

```

json_col1	JSON_MERGE_PRESERVE(json_col1, '[1, 2]')	JSON_MERGE_PRESERVE('[1, 2]', json_col1)
{ "key1": "value1", "key2": "value2" }	{ "key1": "value1", "key2": "value2", 1, 2 }	[1, 2, { "key1": "value1", "key2": "value2" }]
[3, "string1"]	[3, "string1", 1, 2]	[1, 2, 3, "string1"]
{ "key3": "value3", "key4": "value4" }	{ "key3": "value3", "key4": "value4", 1, 2 }	[1, 2, { "key3": "value3", "key4": "value4" }]
["string2", 3, { "key5": "value5", "key6": "value6" }]	["string2", 3, { "key5": "value5", "key6": "value6", 1, 2 }]	[1, 2, "string2", 3, { "key5": "value5", "key6": "value6" }]
[{ "key1": "value1", "key2": "value2", "string2": 3 }]	[{ "key1": "value1", "key2": "value2", "string2": 3, 1, 2 }]	[1, 2, { "key1": "value1", "key2": "value2", "string2": 3 }]
{ "key1": ["string3", 6], "key2": "value2" }	{ "key1": ["string3", 6], "key2": "value2", 1, 2 }	[1, 2, { "key1": ["string3", 6], "key2": "value2" }]

Result 1 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
26	02:32:45	select json_col1, -- JSON_EXTRACT() FUNCTION FOR SEARCH -- searching for values wit...	6 row(s) returned	0.000 sec / 0.000 s
27	02:35:45	select json_col1, -- merging JSON datatype columns with JSON array and vice versa JSON_...	6 row(s) returned	0.000 sec / 0.000 s

Merging JSON data type columns with JSON object and vice versa

Automatic context help disabled. Use the toolbar manually get help for current caret position or toggle automatic help

```

6
7  -- merging JSON datatype columns with JSON array and vice versa
8  -- JSON_MERGE_PRESERVE(json_col1, '[1, 2]'),
9  -- JSON_MERGE_PRESERVE('[1, 2]', json_col1),
10
11  -- merging JSON datatype columns with JSON object and vice versa
12  JSON_MERGE_PRESERVE(json_col1, '{"keymerge": "valuemerge"}'),
13  JSON_MERGE_PRESERVE('{"keymerge": "valuemerge"}', json_col1)
14
15  from J_table;
16

```

json_col1	JSON_MERGE_PRESERVE(json_col1, '{"keymerge": "valuemerge"}')	JSON_MERGE_PRESERVE('{"keymerge": "valuemerge"}', json_col1)
{ "key1": "value1", "key2": "value2" }	{ "key1": "value1", "key2": "value2", "keymerge": "valuemerge" }	{ "key1": "value1", "key2": "value2", "keymerge": "valuemerge" }
[3, "string1"]	[3, "string1", { "keymerge": "valuemerge" }]	[{ "keymerge": "valuemerge" }, 3, "string1"]
{ "key3": "value3", "key4": "value4" }	{ "key3": "value3", "key4": "value4", "keymerge": "valuemerge" }	{ "key3": "value3", "key4": "value4", "keymerge": "valuemerge" }
["string2", 3, { "key5": "value5", "key6": "value6" }]	["string2", 3, { "key5": "value5", "key6": "value6", "keymerge": "valuemerge" }]	[{ "keymerge": "valuemerge" }, "string2", 3, { "key5": "value5", "key6": "value6" }]
[{ "key1": "value1", "key2": "value2", "string2": 3 }]	[{ "key1": "value1", "key2": "value2", "string2": 3, "keymerge": "valuemerge" }]	[{ "keymerge": "valuemerge" }, { "key1": "value1", "key2": "value2", "string2": 3 }]
{ "key1": ["string3", 6], "key2": "value2" }	{ "key1": ["string3", 6], "key2": "value2", "keymerge": "valuemerge" }	[{ "key1": ["string3", 6], "key2": "value2", "keymerge": "valuemerge" }]

Result 2 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
27	02:35:45	select json_col1, -- merging JSON datatype columns with JSON array and vice versa JSON_...	6 row(s) returned	0.000 sec / 0.000 sec
28	02:37:44	select json_col1, -- merging JSON datatype columns with JSON array and vice versa -- JSON_...	6 row(s) returned	0.000 sec / 0.000 sec

JSON_MERGE_PATCH()

Merging JSON data type columns with JSON array and vice versa

Query 1 Creation_Insertion SELECTION_AND_SEARCHING OTHER_FUNCTIONS **MERGE*** x

Limit to 1000 rows

SQLAdditions: Automatic context help disabled. Use the toolbar to manually get help for the current caret position or toggle automatic help

```

14
15 -- Using JSON_MERGE_PRESERVE()
16
17 -- merging JSON datatype columns with JSON array and vice versa
18 JSON_MERGE_PATCH(json_col1, '[1, 2]'),
19 JSON_MERGE_PATCH('[1, 2]', json_col1)
20
21 from J_table;
22
23

```

Result Grid

json_col1	JSON_MERGE_PATCH(json_col1, [1, 2])	JSON_MERGE_PATCH([1, 2], json_col1)
{'key1': 'value1', 'key2': 'value2'}	[1, 2]	{'key1': 'value1', 'key2': 'value2'}
[3, 'string1']	[1, 2]	[3, 'string1']
{'key3': 'value3', 'key4': 'value4'}	[1, 2]	{'key3': 'value3', 'key4': 'value4'}
['string2', 3, {'key5': 'value5', 'key6': 'value6'}	[1, 2]	['string2', 3, {'key5': 'value5', 'key6': 'value6'}
[{'key1': 'value1', 'key2': 'value2'}, 'string2', 3]	[1, 2]	[{'key1': 'value1', 'key2': 'value2'}, 'string2', 3]
{'key1': ['string3', 6], 'key2': 'value2'}	[1, 2]	{'key1': ['string3', 6], 'key2': 'value2'}

Result 3 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
28	02:37:44	select json_col1, -- merging JSON datatype columns with JSON array and vice versa -- JSON...	6 row(s) returned	0.000 sec / 0.0
29	02:39:41	select json_col1, -- merging JSON datatype columns with JSON array and vice versa -- JSON...	6 row(s) returned	0.000 sec / 0.0

Merging JSON data type columns with JSON object and vice versa

Query 1 Creation_Insertion SELECTION_AND_SEARCHING OTHER_FUNCTIONS **MERGE*** x

Limit to 1000 rows

SQLAdditions: Automatic context help disabled. Use the toolbar to manually get help for the current caret position or toggle automatic help

```

16
17 -- merging JSON datatype columns with JSON array and vice versa
18 -- JSON_MERGE_PATCH(json_col1, '[1, 2]'),
19 -- JSON_MERGE_PATCH('[1, 2]', json_col1)
20
21 -- merging JSON datatype columns with JSON object and vice versa
22 JSON_MERGE_PATCH(json_col1, '{"keymerge": "valuemerge"}'),
23 JSON_MERGE_PATCH('{"keymerge": "valuemerge"}', json_col1)
24
25 from J_table;
26

```

Result Grid

json_col1	JSON_MERGE_PATCH(json_col1, {'keymerge': 'valuemerge'})	JSON_MERGE_PATCH({'keymerge': 'valuemerge'}, json_col1)
{'key1': 'value1', 'key2': 'value2'}	{'key1': 'value1', 'key2': 'value2', 'keymerge': 'valuemerge'}	{'key1': 'value1', 'key2': 'value2', 'keymerge': 'valuemerge'}
[3, 'string1']	{'keymerge': 'valuemerge'}	[3, 'string1']
{'key3': 'value3', 'key4': 'value4'}	{'key3': 'value3', 'key4': 'value4', 'keymerge': 'valuemerge'}	{'key3': 'value3', 'key4': 'value4', 'keymerge': 'valuemerge'}
['string2', 3, {'key5': 'value5', 'key6': 'value6'}	{'keymerge': 'valuemerge'}	['string2', 3, {'key5': 'value5', 'key6': 'value6'}
[{'key1': 'value1', 'key2': 'value2'}, 'string2', 3]	{'keymerge': 'valuemerge'}	[{'key1': 'value1', 'key2': 'value2'}, 'string2', 3]
{'key1': ['string3', 6], 'key2': 'value2'}	{'key1': ['string3', 6], 'key2': 'value2', 'keymerge': 'valuemerge'}	{'key1': ['string3', 6], 'key2': 'value2', 'keymerge': 'valuemerge'}

Result 4 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
29	02:39:41	select json_col1, -- merging JSON datatype columns with JSON array and vice versa -- JSON...	6 row(s) returned	0.000 sec / 0.000 sec
30	02:40:54	select json_col1, -- merging JSON datatype columns with JSON array and vice versa -- JSON...	6 row(s) returned	0.000 sec / 0.000 sec

Other Functions:

The screenshot displays a SQL IDE interface with a query editor and an output pane. The query editor contains several SQL statements demonstrating JSON functions. The output pane shows the execution results for two of these queries.

Query 1: OTHER_FUNCTIONS

```

3  -- Appending integer element into index 2 of an array
4  • select json_col1,JSON_ARRAY_APPEND(json_col1 ,'$[1]',1) from J_table;
5
6  -- Inserting integer element into index 2 of an array
7  • select JSON_ARRAY_INSERT(json_col1 ,'$[0]',1) from J_table;
8
9  -- JSON column objects storage size calculation
10 • select json_col1, JSON_STORAGE_SIZE(json_col1) from J_table;
11
12 -- JSON column element Type
13 • select json_col1, JSON_TYPE(json_col1) from J_table;
14
15 -- JSON column element length (gives length of array or number of key-value pairs in JSON object)
16 • select json_col1, JSON_LENGTH(json_col1) from J_table;
17
18 -- confirming validity of json objects before inserting (returns 1 if valid 0 if not)
19 • select JSON_VALID('string'), JSON_VALID('"string1"');
20
21

```

Output:

#	Time	Action	Message	Duration / Fetch
54	03:25:09	select json_col1, JSON_LENGTH(json_col1) from J_table LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 s
55	03:25:09	select JSON_VALID('string'), JSON_VALID('"string1"') LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 s

GitHub repository address:

https://github.com/PradeepReddy-Baireddy/MSCS_542L_256_23S_College-Data-Management-System_Super-Six