The following are selected examples from the Second Edition of MySQL and JSON - A Practical Programming Guide. Please use them to follow along with the book and as a starting point to try out various facets of the MySQL JSON data type.

2-1

```
mysql> CREATE TABLE foo (oldJson char(250));
Query OK, 0 rows affected (0.32 sec)

mysql> INSERT INTO foo
     VALUES ('{ "name" : "Bond", "first" : "James", "ID" :
"007" }');
Query OK, 1 row affected (0.04 sec)

mysql> SELECT * FROM foo;
```

2-2

```
mysql>  SELECT *
        FROM foo
        WHERE oldJson REGEXP 'Bond';
```

2-3

```
mysql> SELECT *
  FROM foo
  WHERE oldJson REGEXP 'J*m';
```

2-4

```
mysql> CREATE TABLE bar (our_data JSON);
Query OK, 0 rows affected (0.40 sec)
mysql> INSERT INTO bar
       VALUES ('{ "name" : "Bond", "first" : "James", "ID"
: "007" }');
```

2-5

```
mysql> SELECT * FROM bar;
```

3-1

```
mysql> USE world_x;
Database changed
mysql> DESCRIBE countryinfo;
```

3-2

```
mysql> SELECT doc
```

```
        FROM countryinfo
        WHERE _id='USA';
```

3-3

```
mysql> SELECT JSON_PRETTY(doc)
       FROM countryinfo
       WHERE _id='USA';
```

3-4

```
SELECT JSON_KEYS(doc)
       FROM countryinfo
       WHERE _id='USA';
```

3-5

```
mysql> SELECT JSON_PRETTY(JSON_KEYS(doc))
       FROM countryinfo
  WHERE _id='USA';
```

3-6

```
mysql> SELECT JSON_KEYS(doc,"$.geography")
       FROM countryinfo
       WHERE _id='USA';
```

3-7

```
mysql> SELECT JSON_EXTRACT(doc,"$.geography")
       FROM countryinfo
  WHERE _id='USA';
```

3-8

```
mysql> SELECT JSON_EXTRACT(doc,"$.geography.Region")
       FROM countryinfo
  WHERE _id='USA';
```

3-9

```
mysql> SELECT JSON_EXTRACT(doc,"$.*.Region")
       FROM countryinfo
  WHERE _id='USA';
```

4-1

```
mysql> SELECT JSON_KEYS(doc)
       FROM countryinfo
  WHERE _id = 'USA';
```

4-2

```
mysql> SELECT JSON_KEYS(doc,"$.geography")
       FROM countryinfo
```

```
  WHERE _id = 'USA';

4-3
     mysql> SELECT JSON_CONTAINS_PATH(doc,"ONE","$.geography")
            FROM countryinfo
            WHERE _id='USA';
4-4
          mysql> SELECT
          JSON_CONTAINS_PATH(doc,"ONE","$.geography","$.governme
          nt") FROM countryinfo
          WHERE _id='USA';

4-5
     mysql> SELECT
JSON_CONTAINS_PATH(doc,"ALL","$.geography","$.governmentx") FROM
countryinfo
     WHERE _id='USA';

4-6

     mysql> SELECT JSON_CONTAINS(doc,"1776","$.IndepYear")
            FROM countryinfo
       WHERE _id='USA';

4-7
          mysql> SELECT JSON_SEARCH(doc,"ONE", "United States")
            FROM countryinfo
            WHERE _id='usa';
4-8
     mysql> SELECT JSON_SEARCH(doc,"ONE", "North America")
            FROM countryinfo
            WHERE _id='usa';

4-9
     mysql> CREATE TABLE booltest (id INT UNSIGNED, doc JSON);
     Query OK, 0 rows affected (0.0644 sec)
     mysql> INSERT INTO booltest
            VALUES (1, '{"finished" : "true" }');
     Query OK, 1 row affected (0.0124 sec)
     mysql> INSERT INTO booltest
            VALUES (99, '{  "finished" : "true" }');
     Query OK, 1 row affected (0.0054 sec)
     mysql> INSERT INTO booltest
            VALUES (94, '{  "finished" : "false" }');
     Query OK, 1 row affected (0.0048 sec)
     mysql> SELECT id
            FROM booltest
```

```
            WHERE
             INSTR(JSON_EXTRACT(doc, "$.finished"), 'true');


4-10
     SELECT 'Joe' MEMBER OF ('["Joe","Betty","Hakeem"]');

4-11
     SELECT 5 MEMBER OF ('[1, 3, 5, 7, 9, 11]');
     SELECT 5 MEMBER OF
         ('[1, 3, 5, "Fred", 7, 9, "Lenka", 11]');

4-12

     mysql> SELECT JSON_CONTAINS('{"Moe": 1, "Larry":
2}','{"Moe": 1}');

4-13

     mysql> SELECT JSON_CONTAINS('{"Moe": 7, "Larry": 2}',
                                 '7','$.Moe');


4-14
     mysql> SELECT JSON_OVERLAPS("[1,3,5,7]","[2,3,4,5]");

4-15
     mysql> SELECT JSON_OVERLAPS("[1,3,5,7]","[2,4,6]");


4-16
     mysql> SELECT JSON_OVERLAPS("[1,3,5,7]","[1,3,5,9]");
     mysql> SELECT JSON_CONTAINS("[1,3,5,7]","[1,3,5,9]");
4-17

     mysql> SELECT
     JSON_EXTRACT(doc,"$.demographics.LifeExpectancy") as raw,
     JSON_VALUE(doc,"$.demographics.LifeExpectancy" RETURNING
DECIMAL(4,2)) as trimmed
     FROM countryinfo
     WHERE _id='USA';


4-18

     CREATE TABLE inventory( items JSON,
      INDEX i3 ( (JSON_VALUE(items, '$.quantity'
```

```
                    RETURNING UNSIGNED)) )
      );

4-19
     SELECT *
          FROM inventory
          WHERE JSON_VALUE(items, '$.price' RETURNING
                DECIMAL(5,2)) <= 100.01;
4-20
     SELECT (JSON_VALUE("{ 'first_name' : 'Dave'}",'$.last_name'
          DEFAULT 'No Last Name' ON ERROR)) as "last name";
5-1
     mysql> CREATE DATABASE testjson; USE testjson;
     Database changed
     mysql> CREATE TABLE y (x JSON);
     Query OK, 0 rows affected (0.05 sec)
     mysql> INSERT INTO y VALUES (JSON_ARRAY('A','B','C'));

5-2

     SELECT * FROM y;

5-3

     UPDATE y SET x=JSON_ARRAY_APPEND(x,"$[0]","*");
     SELECT * FROM y;
5-4

     UPDATE y SET x=JSON_ARRAY_APPEND(x,"$","#");

5-5
     UPDATE y
            SET x=JSON_ARRAY_APPEND(x,"$[1]","@","$[3]","+");

5-6
     UPDATE y SET x=JSON_ARRAY_INSERT(x,"$[0]","&");

5-7

     UPDATE y
          SET x=JSON_ARRAY_INSERT(x,"$[1]","777","$[3]","999");

5-8
     TRUNCATE y;
     INSERT INTO y
     VALUES('{ "key1" : "value1" }');
```

```
5-9
     UPDATE y
            SET x = JSON_INSERT(x,'$.key2','value2');

5-10

     UPDATE y SET x =
JSON_INSERT(x,'$.key1','value1x',"$.key3","value3");

5-11

     UPDATE y SET x =
JSON_REPLACE(x,"$.key1","Value1A","$.key3","VALUE-3");

5-12
     UPDATE y
            SET x = JSON_REMOVE(x,"$.key2");

5-13

     UPDATE y
          SET x =
     JSON_SET(x,"$.key1","Value 1X","$.key99","Value-99");

5-14
     SELECT JSON_EXTRACT(x,"$.key1") FROM y;
     SELECT JSON_UNQUOTE(JSON_EXTRACT(x,"$.key1"))
       FROM y;
          SELECT x->>"$.key1"
       FROM y;
5-15
     SELECT
           JSON_MERGE('{ "odds" : 1, "evens" : 2 }',
      '{ "odds": 3, "evens" : 4 }');
          SELECT
            JSON_MERGE_PRESERVE('{ "odds" : 1, "evens" : 2 }',
       '{ "odds": 3, "evens" : 4 }');
          select
                 JSON_MERGE_PATCH('{ "odds" : 1, "evens" : 2 }',
        '{ "odds": 3, "evens" : 4 }');

5-16

     select JSON_MERGE('{ "odds" : 1, "evens" : 2 }','{ "odds":
     3, "evens" : 4 }');

5-17
```

```
     SELECT * from y;
     SELECT JSON_MERGE(x->"$",'{ "key2" : "Buzz" }')
       FROM y;
5-18
     SELECT JSON_MERGE('{ "odds" : 1, "evens" : 2 }',
       '{ "odds": 3, "evens" : 4 }');

5-19

     SELECT JSON_DEPTH(doc),JSON_KEYS(doc) FROM countryinfo
WHERE _id = 'USA';

5-20
     SELECT JSON_KEYS(doc),
            JSON_LENGTH(doc)
            FROM countryinfo
            LIMIT 1;

5-21
     SELECT JSON_KEYS(doc,'$.geography'),
                 JSON_LENGTH(doc,'$.geography')
                 FROM countryinfo LIMIT 1;
5-22

     SELECT JSON_TYPE('[1,2,3]'),
                               JSON_TYPE('{ "x":"y"}'),
                               JSON_TYPE('123'),
                               JSON_TYPE(NULL)\G
5-23

     SELECT JSON_VALID('{ "A" : 1}'),
                  JSON_VALID('A'),
                  JSON_VALID('"A"')\G

5-24

     SELECT JSON_EXTRACT(doc,'$.Name'),
                  JSON_STORAGE_SIZE(doc)
                  FROM countryinfo
                  WHERE _id IN ("USA","BRA");
5-25

CREATE DATABASE  IF NOT EXISTS test;USE test;
CREATE TABLE x (id INT UNSIGNED, doc JSON);
INSERT INTO x (id,doc) VALUES (1,'{ "a" : 1 }');
```

```
UPDATE x SET doc = JSON_SET(doc,'$[0]','{ "a" : "This is a
string" }');
UPDATE x  SET doc = JSON_SET(doc,'$.a','{ "a" : "a" }');
SELECT JSON_STORAGE_FREE(doc) FROM x;
```

6-1

```
     SELECT city.Name,
            country.Name
     FROM city
     JOIN country ON (city.CountryCode=country.Code)
     LIMIT 5;
```

6-2

```
     SELECT
         JSON_OBJECT("City", city.Name, "Country", country.Name)
         FROM city
         JOIN country ON (city.CountryCode=country.Code)
         LIMIT 5;
```

6-3

```
     SELECT JSON_ARRAY(Code, Name, Capital)
            FROM country
            LIMIT 1;
```

6-4

```
     SELECT JSON_TYPE(CAST('[1,2]' AS JSON));
     SELECT JSON_TYPE(CAST('1' AS JSON));
```

6-5

```
     SELECT
CAST(JSON_EXTRACT(doc,"$.demographics.LifeExpectancy") AS
unsigned)
 FROM countryinfo WHERE _id = 'USA';
```

6-6

```
     SELECT json_extract(doc,"$.demographics.LifeExpectancy")
FROM countryinfo  WHERE _id = 'USA';
     SELECT
     CAST(json_extract(doc,"$.demographics.LifeExpectancy") AS
     UNSIGNED)
     FROM countryinfo  WHERE _id = 'USA';
```

6-7

```
     SELECT
     CAST(json_extract(doc,"$.demographics.LifeExpectancy") AS
     UNSIGNED)
     FROM countryinfo
     WHERE _id = 'USA';
```

6-8

```
     SELECT country_name,
                  IndyYear
          FROM countryinfo,
          JSON_TABLE(doc, "$" COLUMNS (
            country_name CHAR(20) PATH "$.Name",
            IndyYear INT PATH "$.IndepYear")) as stuff
          WHERE IndyYear > 1992;
```

6-9

```
     SELECT country_name,
                  IndyYear
          FROM countryinfo,
          JSON_TABLE(doc, "$" COLUMNS (
            country_name CHAR(20) PATH "$.Name",
            IndyYear INT PATH "$.IndepYear")) as stuff
          WHERE IndyYear > 1992;
```

6-11

```
     SELECT * FROM t1,
            JSON_TABLE(doc,"$" COLUMNS (
            xHasValue INT PATH "$.x" DEFAULT '999' ON EMPTY,
            hasname CHAR(10) EXISTS PATH "$.name",
            mojo CHAR(5) EXISTS PATH "$.mojo"))
            AS t2;
```

6-12

```
     SELECT * FROM t1,
            JSON_TABLE(doc,"$" COLUMNS (
            xHasValue INT PATH "$.x" DEFAULT '999' ON EMPTY,
            hasname CHAR(10) EXISTS PATH "$.name",
            mojo CHAR(5) EXISTS PATH "$.mojo"))
            AS t2
            WHERE hasname=1 and xHasValue = 1;
```

6-13

```
     SELECT * FROM t2,
JSON_TABLE(doc, "$" COLUMNS (
myX INT PATH "$.x",
               NESTED PATH "$.y[*]" COLUMNS (
```

```
                        myID FOR ORDINALITY,
                        myZ CHAR(10) PATH "$.z")))
                AS tt;


7-1
     CREATE TABLE taxCalc (itemPrice DECIMAL(10,3),
     taxRate DECIMAL(10,3),
     taxAmount DECIMAL(10,3) AS (itemPrice * taxRate));
     INSERT INTO taxCalc (itemPrice, taxRate)
            VALUES (10.0,0.08), (100.0,0.25);
     SELECT * FROM taxCalc;
7-3
     ALTER TABLE countryinfo
                 ADD COLUMN PopulationCountry INT AS
              (JSON_UNQUOTE(doc->"$.demographics.Population"));
7-6
     ALTER TABLE countryinfo
          DROP COLUMN PopulationCountry;
     ALTER TABLE countryinfo
            ADD COLUMN PopulationCountry INT AS
     (JSON_UNQUOTE(doc->"$.demographics.Population")) STORED;


8-1

     SELECT ST_AsText(ST_GeomFromGeoJSON('{ "type" : "Point",
"coordinates" : [99.1, 1.1]}'));

8-2

     SELECT  ST_GeomFromGeoJSON('{ "type" : "Point",
     "coordinates" : [99.1, 1.1]}',4);

8-3
     SELECT
     ST_AsGeoJSON(ST_GeomFromText('POINT(12.3456 23.4567)'),2);

8-4

     SELECT
     ST_AsGeoJSON(ST_GeomFromText('POINT(12.3456
23.4567)'),2,1);


8-5
```

```
     SELECT
     ST_AsGeoJSON(ST_GeomFromText('POINT(12.3456
     23.4567)'),2,4);
     SELECT
     ST_AsGeoJSON(ST_GeomFromText('POINT(12.3456
23.4567)'),2,5);

10-2
     use test;
     create table zipcode (doc JSON,
     _id char(5) GENERATED ALWAYS AS
     (JSON_UNQUOTE(JSON_EXTRACT(doc,'$._id'))) STORED NOT NULL,
          PRIMARY KEY (_id));

12-1
          import mysqlx

          # Connect to server on localhost
          session = mysqlx.get_session({
               'host': 'localhost',
               'port': 33060,
               'user': 'dave',
               'password': 'S3cR3T!',
               'ssl-mode' : mysqlx.SSLMode.DISABLED #Remove this
          line if SSL enabled
          })

          schema = session.get_schema('world_x')

          # Use the collection 'countryinfo'
          collection = schema.get_collection('countryinfo')

          # Specify which document to find with
          Collection.find()
          result = collection.find('_id like
          :param').bind('param', 'USA').execute()

          # Print document
          docs = result.fetch_all()
          print('id: {0}'.format(docs[0]['Name']))
12-2
          // Simple example to grab one record and print it
          const mysqlx = require('@mysql/xdevapi');
          const options = {
            host: 'localhost',
            port: 33060,
            dbUser: 'dave',
```

```
  dbPassword: 'S3cR3t!!'
};

mysqlx
  .getSession(options)
  .then (session => {
  var schema = session.getSchema('world_x');

//equivalent of SELECT doc FROM countryinfo where _id
= 'USA'
  var coll = schema.getCollection('countryinfo');
  var query = "$._id == 'USA'";

      // Print doc
     return Promise.all([
       coll.find(query).execute(function (doc) {
       console.log(doc);
      }),
      session.close()
     ]);
    })
    .catch(err => {
        console.log(err.message);
        console.log(err.stack);
    });
```

12-3

```
#!/usr/bin/php
<?PHP
// Connection parameters
  $user = 'dave';
  $passwd = 'S3cR3t!';
  $host = 'localhost';
  $port = '33060';
  $connection_uri =
'mysqlx://'.$user.':'.$passwd.'@'.$host.':'.$port;
  echo $connection_uri . "\n";

// Connect as a Node Session
  $nodeSession =
mysql_xdevapi\getNodeSession($connection_uri);
// "USE world_x"
  $schema = $nodeSession->getSchema("world_x");
// Specify collection to use
  $collection = $schema->getCollection("countryinfo");

// Query the Document Store
```

```
  $result = $collection->find('_id = "USA"')->fields(['Name
as Country','geography as
Geo','geography.Region'])->execute();

// Fetch/Display data
  $data = $result->fetchAll();
  var_dump($data);
?>
```

14-1

```
CREATE TABLE s (id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
   name CHAR(20) NOT NULL,
   j JSON,
   INDEX nbrs( (CAST(j->'$.nbr' AS UNSIGNED ARRAY)))
   );
```

14-10

```
SELECT id, data->>"$.nbr"
FROM a
WHERE data->>"$.nbr[2]" = 99999
```

15-1

```
set @s='{"type": "object",
     "properties": {
     "myage": {
          "type" : "number",
          "minimum": 28,
          "maximum": 99
     }
    }
}';
```

10-2

```
set @d='{  "myage": 33}';
```

10-3

```
`    select JSON_SCHEMA_VALID(@s,@d);
```