ORACLE

Python and the New MySQL Shell

Dave Stokes
MySQL Community Manager
MySQL Community Team

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, timing, and pricing of any features or functionality described for Oracle’s products may change and remains at the sole discretion of Oracle Corporation.

Dave Stokes

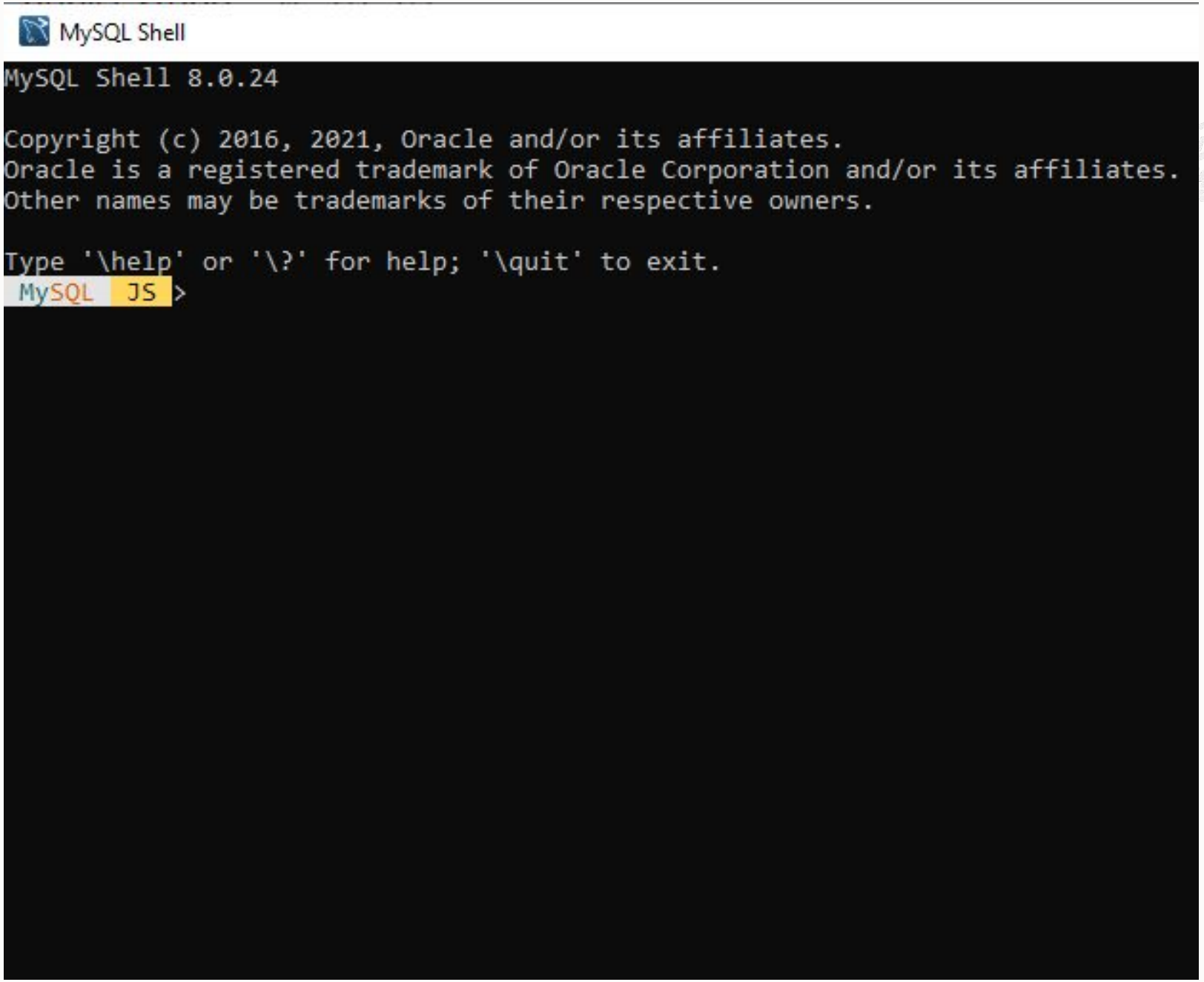
MySQL Community Team
Oracle Corporation
@Stoker
https://elephantdolphin.blogspot.com/
David.Stokes@Oracle.com

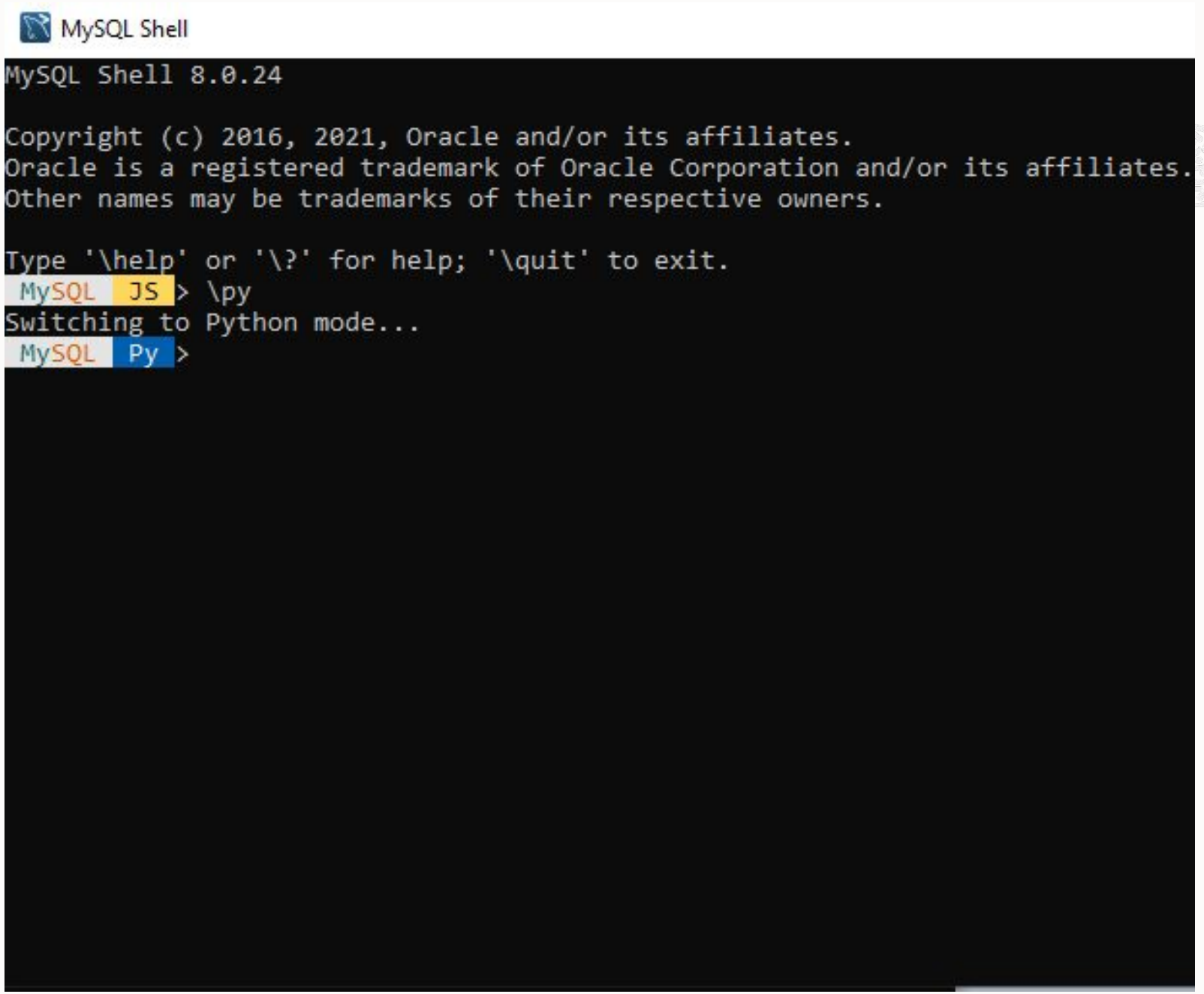
Slides are available at Slideshare.net/davestokes

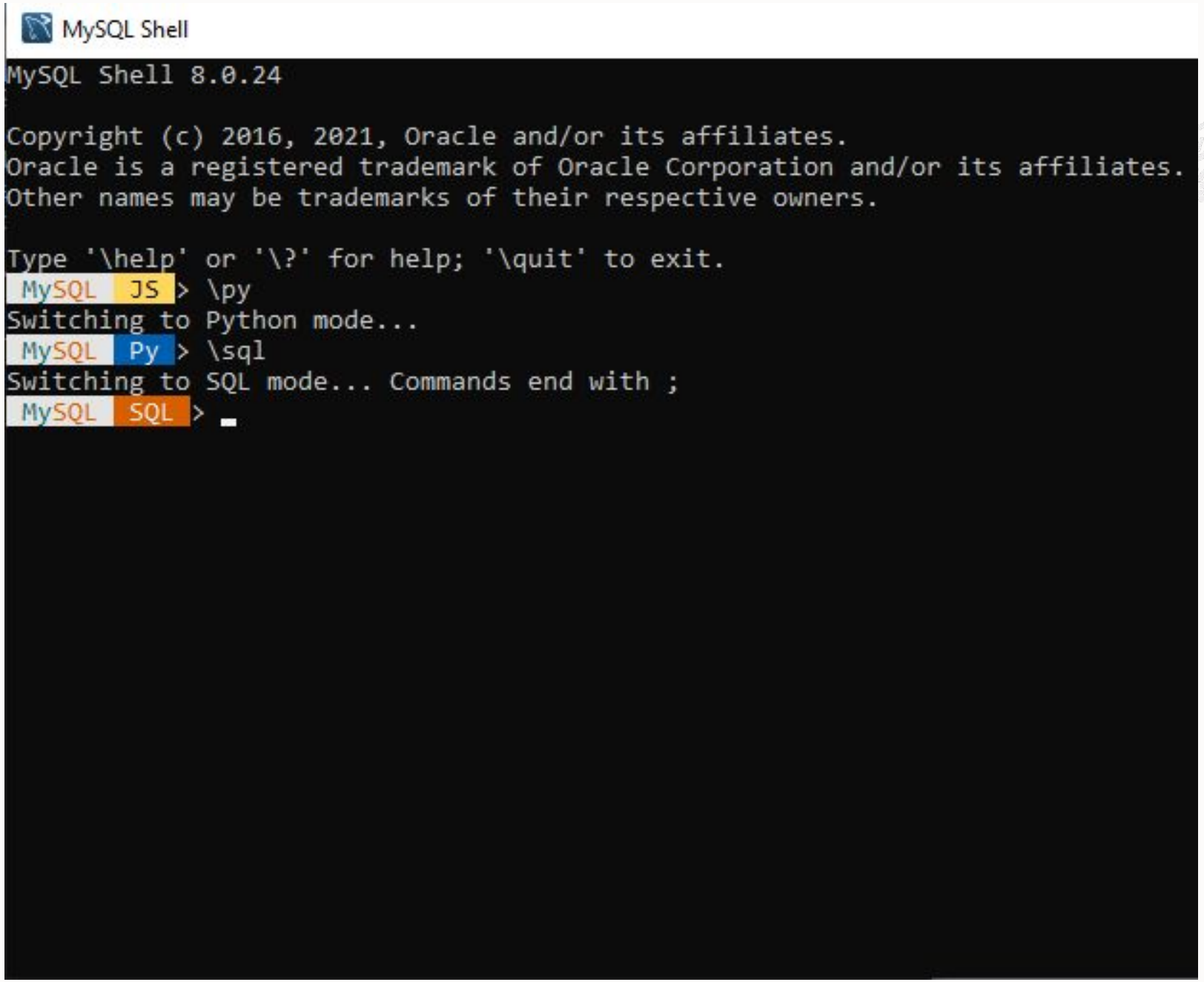
mysqlsh

MySQL Shell processes code written in JavaScript, Python and SQL.

Any executed code is processed as one of these languages, based on the language that is currently active.


MySQL Shell
MySQL Shell 8.0.24
Copyright (c) 2016, 2021, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.
Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS >


MySQL Shell
MySQL Shell 8.0.24
Copyright (c) 2016, 2021, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.
Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS > \py
Switching to Python mode...
MySQL Py >


MySQL Shell
MySQL Shell 8.0.24
Copyright (c) 2016, 2021, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.
Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS > \py
Switching to Python mode...
MySQL Py > \sql
Switching to SQL mode... Commands end with ;
MySQL SQL >

\s for status

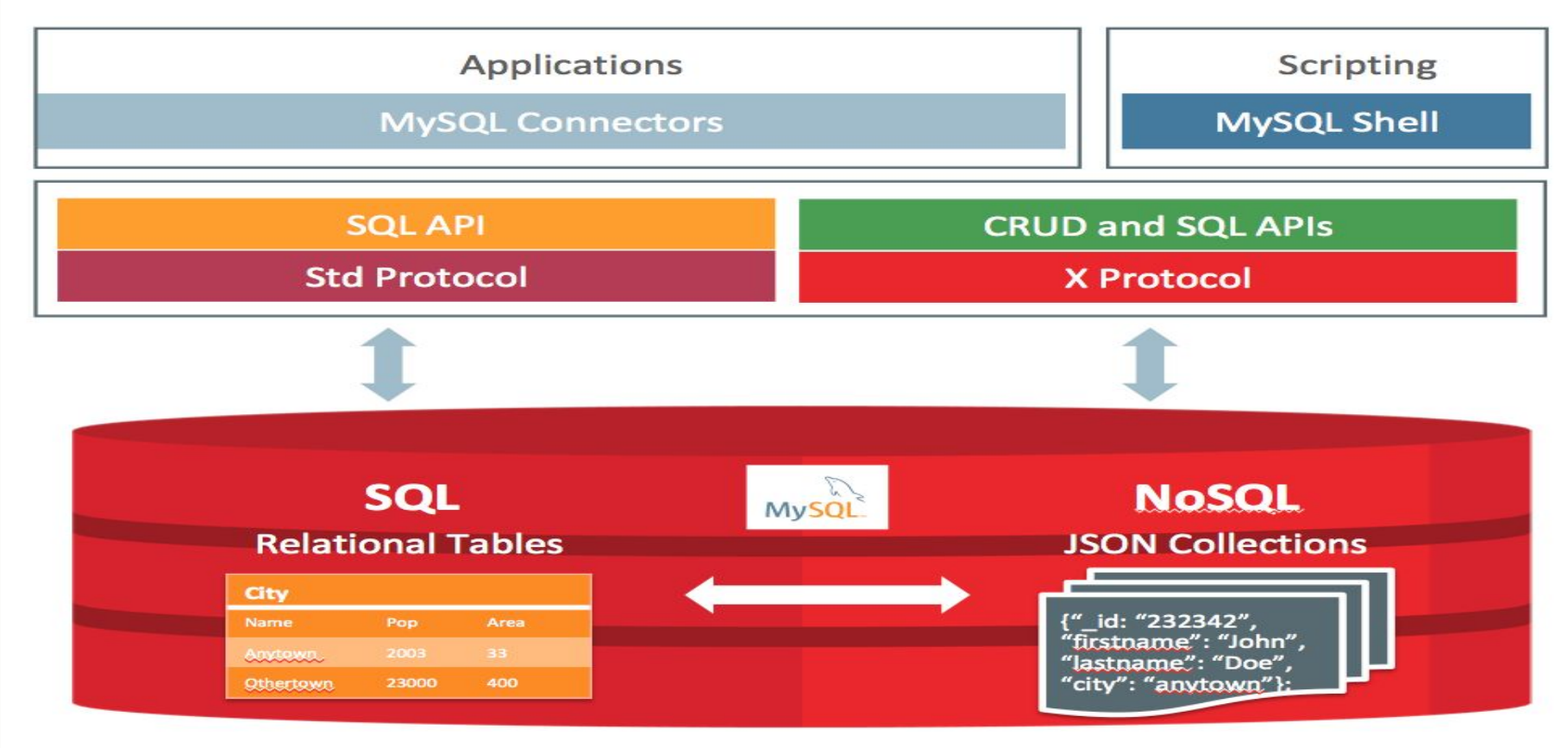
Please note that by default the connection is encrypted with AES 256, is using the UTF8MB4 character set, and compression is enabled.

```
MySQL  localhost:33060+ ssl  demo  Py > \s
MySQL Shell version 8.0.25

Connection Id:                  9
Default schema:                 demo
Current schema:                 demo
Current user:                   root@localhost
SSL:                            Cipher in use: TLS_AES_256_GCM_SHA384 TLSv1.3
Using delimiter:                ;
Server version:                 8.0.25 MySQL Community Server - GPL
Protocol version:               X protocol
Client library:                 8.0.25
Connection:                     localhost via TCP/IP
TCP port:                       33060
Server characterset:            utf8mb4
Schema characterset:            utf8mb4
Client characterset:            utf8mb4
Conn. characterset:             utf8mb4
Result characterset:            utf8mb4
Compression:                    Enabled (DEFLATE_STREAM)
Uptime:                         2 hours 21 min 26.0000 sec


MySQL  localhost:33060+ ssl  demo  Py > _
```

Two types of session

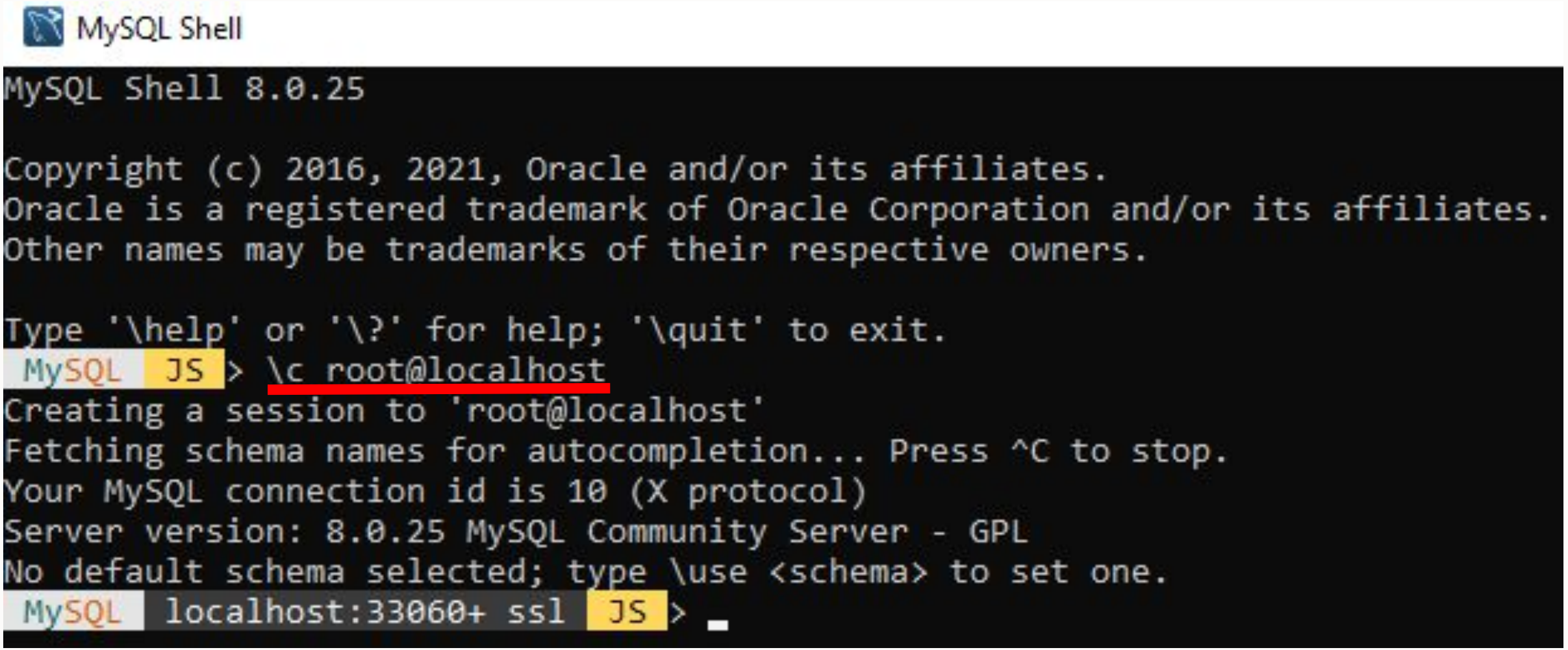
Classic - port 3306

X DevAPI -- SQL & NoSQL -- port 33060



You can use functions available in JavaScript and Python mode to create multiple session objects of your chosen types and assign them to variables.

These session objects let you establish and manage concurrent connections to work with multiple MySQL Server instances, or with the same instance in multiple ways, from a single MySQL Shell instance.


MySQL Shell
MySQL Shell 8.0.25
Copyright (c) 2016, 2021, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.
Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS > \c root@localhost
Creating a session to 'root@localhost'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 10 (X protocol)
Server version: 8.0.25 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
MySQL localhost:33060+ ssl JS >

Or create multiple sessions

```
MySQL Shell 8.0.24

Copyright (c) 2016, 2021, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
 MySQL  JS > var session1 = mysql.getClassicSession('root@localhost:3306','hidave');
 MySQL  JS > session1
<ClassicSession:root@localhost:3306>
 MySQL  JS >
```


Want encryption?

```
session = mysqlx.get_session({
    'host': '127.0.0.1',
    'user': 'root',
    'password': '',
    'tls-versions"': ["TLSv1.1", "TLSv1.2"],
    'tls-ciphersuites': ["DHE-RSA-AES256-SHA"],
})
res = session.sql("SHOW STATUS LIKE 'Mysqlx_ssl_version'").execute().fetch_all()
print("Mysqlx_ssl_version: {}".format(res[0].get_string('Value')))
res = session.sql("SHOW STATUS LIKE 'Mysqlx_ssl_cipher'").execute().fetch_all()
print("Mysqlx_ssl_cipher: {}".format(res[0].get_string('Value')))
session.close()
```


Or compression?

```
shell> mysqlsh --mysqlx -u user -h localhost -C required --compression-algorithms=lz4,zstd --compression-level=5
```


```
validation = {
    "level": "STRICT",
    "schema": {
        "id": "http://json-schema.org/geo",
        "$schema": "http://json-schema.org/draft-07/schema#",
        "title": "Longitude and Latitude Values",
        "description": "A geographical coordinate",
        "required": ["latitude", "longitude"],
        "type": "object",
        "properties": {
            "latitude": {
                "type": "number",
                 "minimum": -90,
                "maximum": 90
            },
            "longitude": {
                "type": "number",
                "minimum": -180,
                "maximum": 180
            }
        },
    }
}
# Create 'my_collection' in schema with a schema validation
schema.create_collection('my_collection', validation=validation)
```

JSON Schema Validation

Keeps bad data out of database

Tests for

- data type
- minimum value
- maximum value
- required fields

Optional password storage

```
mysql-js> \connect user@localhost:3310
Creating a session to 'user@localhost:3310'
Please provide the password for 'user@localhost:3310': ********
Save password for 'user@localhost:3310'? [Y]es/[N]o/Ne[v]er (default No):
y
```


Using MySQL Shell to execute the contents of the file code.py as Python code

```
shell> mysqlsh < code.py
```


Use the libraries you already know

```
MySQL  localhost:33060+ ssl  demo  Py > from datetime import datetime
MySQL  localhost:33060+ ssl  demo  Py > datetime.now()
datetime.datetime(2021, 5, 17, 12, 33, 1, 356269)
MySQL  localhost:33060+ ssl  demo  Py >
```


X DevAPI for using MySQL as a NoSQL JSON Document Store

```
MySQL  localhost:33060+ ssl  world_x  Py > db.get_collections()
[
    <Collection:countryinfo>
]
MySQL  localhost:33060+ ssl  world_x  Py > db.countryinfo.find('Name like :param').bind('param', 'S%').limit(1)
{
    "GNP": 264478,
    "_id": "CHE",
    "Name": "Switzerland",
    "IndepYear": 1499,
    "geography": {
        "Region": "Western Europe",
        "Continent": "Europe",
        "SurfaceArea": 41284
    },
    "government": {
        "HeadOfState": "Adolf Ogi",
        "GovernmentForm": "Federation"
    },
    "demographics": {
        "Population": 7160400,
        "LifeExpectancy": 79.5999984741211
    }
}
1 document in set (0.0009 sec)
MySQL  localhost:33060+ ssl  world_x  Py >
```

X DevAPI for using MySQL as a NoSQL JSON Document Store

```
MySQL  localhost:33060+ ssl  world_x  Py > db.get_collections()
[
    <Collection:countryinfo>
]
MySQL  localhost:33060+ ssl  world_x  Py > db.countryinfo.find('Name like :param').bind('param', 'S%').limit(1)
{
    "GNP": 264478,
    "_id": "CHE",
    "Name": "Switzerland",
    "IndepYear": 1499,
    "geography": {
        "Region": "Western Europe",
        "Continent": "Europe",
        "SurfaceArea": 41284
    },
    "government": {
        "HeadOfState": "Adolf Ogi",
        "GovernmentForm": "Federation"
    },
    "demographics": {
        "Population": 7160400,
        "LifeExpectancy": 79.5999984741211
    }
}
1 document in set (0.0009 sec)
MySQL  localhost:33060+ ssl  world_x  Py >
```

X DevAPI for using MySQL as a NoSQL JSON Document Store

```
MySQL  localhost:33060+ ssl  world_x  Py > db.get_collections()
[
    <Collection:countryinfo>
]
MySQL  localhost:33060+ ssl  world_x  Py > db.countryinfo.find('Name like :param').bind('param', 'S%').limit(1)
{
    "GNP": 264478,
    "_id": "CHE",
    "Name": "Switzerland",
    "IndepYear": 1499,
    "geography": {
        "Region": "Western Europe",
        "Continent": "Europe",
        "SurfaceArea": 41284
    },
    "government": {
        "HeadOfState": "Adolf Ogi",
        "GovernmentForm": "Federation"
    },
    "demographics": {
        "Population": 7160400,
        "LifeExpectancy": 79.5999984741211
    }
}
1 document in set (0.0009 sec)
MySQL  localhost:33060+ ssl  world_x  Py >
```

Running SQL from Python

```
 MySQL  localhost:33060+ ssl  demo  Py > session.run_sql('SELECT user, host FROM mysql.user')
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| Foo              | %         |
| dstokes          | %         |
| mike             | %         |
| myuser           | %         |
| bar              | localhost |
| bill             | localhost |
| davetest         | localhost |
| demo             | localhost |
| demo2            | localhost |
| dstokes          | localhost |
| foo              | localhost |
| foobar           | localhost |
| jack             | localhost |
| mary             | localhost |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
+------------------+-----------+
18 rows in set (0.0017 sec)
 MySQL  localhost:33060+ ssl  demo  Py >
 MySQL  localhost:33060+ ssl  demo  Py >
```

Running SQL from Python, JSON output

```
MySQL  localhost:33060+ ssl  demo  Py >
MySQL  localhost:33060+ ssl  demo  Py > shell.options.set('resultFormat','json')
MySQL  localhost:33060+ ssl  demo  Py > session.run_sql('SELECT user, host FROM mysql.user LIMIT 2')
{
    "user": "Foo",
    "host": "%"
}
{
    "user": "dstokes",
    "host": "%"
}
2 rows in set (0.0008 sec)
MySQL  localhost:33060+ ssl  demo  Py >
MySQL  localhost:33060+ ssl  demo  Py >
```

Also available in table and vertical formats

Using MySQL without SQL! No need to normalize data or set up tables

Here we create a document collection

Add a JSON document

And then find a record

The _id is a primary key for the record and you can supply your own unique value or let the system generate it for you.

```
MySQL  localhost:33060+ ssl  demo  JS > db.createCollection('openjs')
<Collection:openjs>
MySQL  localhost:33060+ ssl  demo  JS > db.openjs.add({
                                    -> "Name": "Dave"})
                                    ->
Query OK, 1 item affected (0.0180 sec)
MySQL  localhost:33060+ ssl  demo  JS > db.openjs.find()
{
    "_id": "00006091464f0000000000000001",
    "Name": "Dave"
}
1 document in set (0.0008 sec)
MySQL  localhost:33060+ ssl  demo  JS >
```

Use your favorite JavaScript libraries or use your own

dave.js:

```
exports.hello = function () {
  println('Hello there!')
}

exports.mysql = function() {
  println('MySQL 8.0 is great!')
}
```

Extensible

You can define extensions to the base functionality of MySQL Shell in the form of reports and extension objects.

Reports and extension objects can be created using JavaScript or Python, and can be used regardless of the active MySQL Shell language.

You can persist reports and extension objects in plugins that are loaded automatically when MySQL Shell starts.

Easy to extend

```
def show_tables(session=None):
   if session is None:
      session = shell.get_session()
   if session is None:
      print("No session specified - pass a session or use an existing connection to database")
      return
   if session is not None:
      r = session.run_sql("SELECT * FROM world.city")
      shell.dump_rows(r)


plugin_obj = shell.create_extension_object()

shell.add_extension_object_member(plugin_obj, "worldinfo", show_tables, {"brief" : "Dave's demo 2"} )
```


Easy to extend

```
def show_tables(session=None):
   if session is None:
      session = shell.get_session()
   if session is None:
      print("No session specified - pass a session or use an existing connection to database")
      return
   if session is not None:
      r = session.run_sql("SELECT * FROM world.city")
      shell.dump_rows(r)


plugin_obj = shell.create_extension_object()

shell.add_extension_object_member(plugin_obj, "worldinfo", show_tables, {"brief" : "Dave's demo 2"} )
```


Easy to extend

```
def show_tables(session=None):
   if session is None:
      session = shell.get_session()
   if session is None:
      print("No session specified - pass a session or use an existing connection to database")
      return
   if session is not None:
      r = session.run_sql("SELECT * FROM world.city")
      shell.dump_rows(r)


plugin_obj = shell.create_extension_object()

shell.add_extension_object_member(plugin_obj, "worldinfo", show_tables, {"brief" : "Dave's demo 2"} )
```


Easy to extend

```
def show_tables(session=None):
   if session is None:
      session = shell.get_session()
   if session is None:
      print("No session specified - pass a session or use an existing connection to database")
      return
   if session is not None:
      r = session.run_sql("SELECT * FROM world.city")
      shell.dump_rows(r)
```

```
plugin_obj = shell.create_extension_object()

shell.add_extension_object_member(plugin_obj, "worldinfo", show_tables, {"brief" : "Dave's demo 2"} )
```


Registering a report

You can create and register a user-defined report for MySQL Shell in either of the supported scripting languages, JavaScript and Python.

The reporting facility handles built-in reports and user-defined reports using the same API frontend scheme.

Reports can specify a list of report-specific options that they accept, and can also accept a specified number of additional arguments.

Your report can support both, one, or neither of these inputs.

When your request help for a report, MySQL Shell provides a listing of options and arguments, and any available descriptions of these that are provided when the report is registered.

Registering MySQL Shell Reports

To register your user-defined report with MySQL Shell, call the shell.**registerReport()** method in JavaScript or shell.**register_report()** in Python.

The syntax for the method is as follows:

shell.registerReport(name, type, report[, description])

Where:

- name is a string giving the unique name of the report.
- type is a string giving the report type which determines the output format, either “list”, “report”, or “print”.
- report is the function to be called when the report is invoked.
- description is a dictionary with options that you can use to specify the options that the report supports, additional arguments that the report accepts, and help information that is provided in the MySQL Shell help system.

Write your own reports

```
def sessions(session, args, options):
    sys = session.get_schema('sys')
    session_view = sys.get_table('session')
    query = session_view.select(
     'thd_id', 'conn_id', 'user', 'db', 'current_statement', 'statement_latency AS latency', 'current_memory AS memory')
    if (options.has_key('limit')):
        limit = int(options['limit'])
        query.limit(limit)

    result = query.execute()
    report = [result.get_column_names()]
    for row in result.fetch_all():
        report.append(list(row))

    return {'report': report}

shell.register_report(
    'sessions',
    'list',
    sessions,
    {
      'brief': 'Shows which sessions exist.',
      'details': ['You need the SELECT privilege on sys.session view and the underlying tables and functions used by
it.'],
      'options': [{'name': 'limit', 'brief': 'The maximum number of rows to return.',  'shortcut': 'l', 'type': 'integer'
}
        ],
        'argc': '0'
    }
)
```

Write your own reports

```
def sessions(session, args, options):
    sys = session.get_schema('sys')
    session_view = sys.get_table('session')
    query = session_view.select(
        'thd_id', 'conn_id', 'user', 'db', 'current_statement',
        'statement_latency AS latency', 'current_memory AS memory')
    if (options.has_key('limit')):
        limit = int(options['limit'])
        query.limit(limit)

    result = query.execute()
```

\watch sessions --interval=2.0

```
+--------+---------+---------------+------+-----------------------------------------------------------+---------+-------------+
| thd_id | conn_id | user          | db   | current_statement                                         | latency | memory      |
+--------+---------+---------------+------+-----------------------------------------------------------+---------+-------------+
| 53     | 15      | mysqlx/worker | demo | NULL                                                      | NULL    | 210.68 KiB |
| 54     | 16      | mysqlx/worker | test | SELECT `thd_id`,`conn_id`,`use ...  `memory` FROM `sys`.`session` | 32.84 ms | 1.42 MiB   |
+--------+---------+---------------+------+-----------------------------------------------------------+---------+-------------+
```

```
shell.register_report(
    'sessions',
    'list',
    sessions,
    {
        'brief': 'Shows which sessions exist.',
        'details': ['You need the SELECT privilege on sys.session view and the underlying tables and functions used by it.'],
        'options': [
            {
                'name': 'limit',
                'brief': 'The maximum number of rows to return.',
                'shortcut': 'l',
                'type': 'integer'
            }
        ],
        'argc': '0'
    }
)
```

Built-in reports

```
mysql-js> \watch query --interval=0.5 show global status like 'Com%'
```

```
mysql-js> \show threads --foreground -o
tid,ptid,cid,user,host,progname,command,memory
```

```
mysql-py>  \show thread --tid 53 --general --client --locks
```


MySQL Shell includes the following utilities for working with MySQL:

An upgrade checker utility to verify whether MySQL server version 5.7 instances are ready for upgrade to version 8.0 . Use **util.checkForServerUpgrade()** to access the upgrade checker.

A JSON import utility to import JSON documents to a MySQL Server collection or table.

```
util.importjson("/tmp/products.json", {"schema": "mydb", "collection": "products"})
```

A parallel table import utility that splits up a single data file and uses multiple threads to load the chunks into a MySQL table.

```
util.importTable("/tmp/productrange.csv", {schema: "mydb", table: "products", dialect: "csv-unix", skipRows: 1, showProgress: true})
```


MySQL Shell includes the following utilities for working with MySQL:

Parallel Table Import Utility

```
mysql-js> util.importTable("/tmp/productrange.csv", {schema: "mydb", table: "products", dialect: "csv-unix", skipRows: 1, showProgress: true})
```

Table export for use with MySQL Shell's parallel table import utility

```
shell-js> util.exportTable("hr.employees", "file:///home/hanna/exports/employees.txt")
```

Instance Dump Utility, Schema Dump Utility, and Table Dump Utility

```
shell-js> util.dumpInstance("C:/Users/hanna/worlddump", {dryRun: true, ocimds: true})
```

```
shell-js> util.dumpTables("hr", [ "employees", "salaries" ], "emp")
```

```
shell-py> util.dump_schemas(["world"], "worlddump", { "osBucketName": "hanna-bucket", "osNamespace": "idx28w1ckztq",
"ocimds": "true", "compatibility":["strip_definers", "strip_restricted_grants"]})
```

```
shell-js> util.loadDump("/mnt/data/worlddump")
```

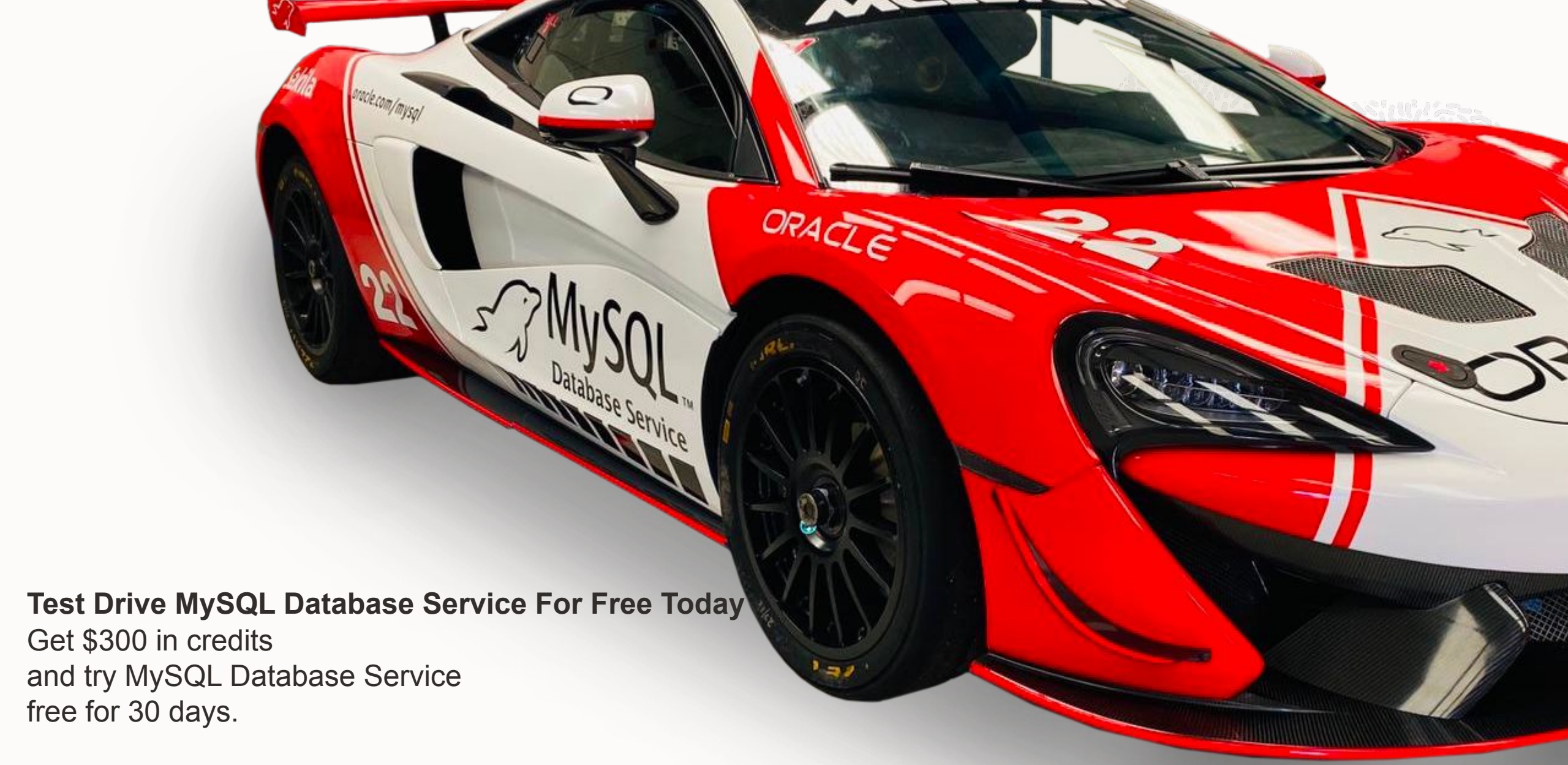

Where To Learn More

MySQL Shell 8.0
https://docs.oracle.com/cd/E17952_01/mysql-shell-8.0-en/mysql-shell-8.0-en.pdf

Example how to write a plugin
https://developpaper.com/technology-sharing-how-to-write-mysql-shell-plug-in/

Example Plugins
https://www.slideshare.net/lefred.descamps/mysql-shell-the-best-dba-tool-197880094

& https://github.com/lefred/mysqlshell-plugins



Test Drive MySQL Database Service For Free Today
Get $300 in credits
and try MySQL Database Service
free for 30 days.

https://www.oracle.com/cloud/free/

Follow us on Social Media

Connect with us

mysql.com
twitter.com/mysql
facebook.com/mysql
linkedin.com/company/mysql

MySQLCommunity.slack.com

Oracle for Startups - enroll at oracle.com/startup
A Virtuous Cycle of Innovation, Everybody Wins.

Startups get cloud credits and a 70% discount for 2 years, global exposure via marketing, events, digital promotion, and media, plus access to mentorship, capital and Oracle's 430,000+ customers

Customers meet vetted startups in transformative spaces that help them stay ahead of their competition

Oracle stays at the competitive edge of innovation with solutions that complement its technology stack

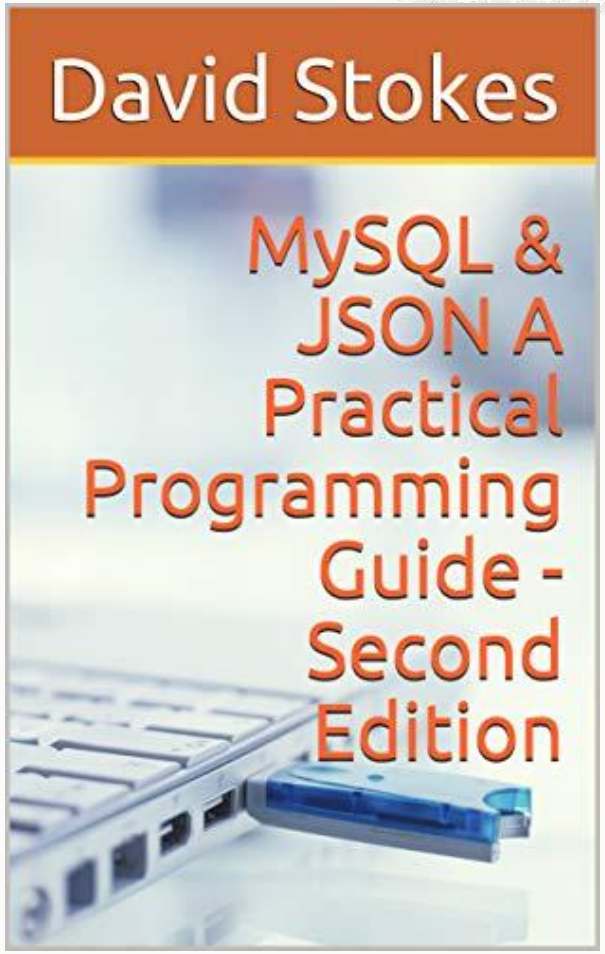
We have saved around 40% of our costs and are able to reinvest that back into the business. And we are scaling across EMEA, and that's basically all because of Oracle."

—Asser Smidt

CEO and Cofounder, BotSupply

Interested in using JSON with MySQL?

Then please consider buying my book on the JSON data type, how to use the supporting functions, and it is filled with example code to get you up to speed!



Q&A

Thank You!

David.Stokes@oracle.com
@Stoker
slideshare.net/davestokes


ORACLE