
Types of MySQL Variables – Local Variables
365 DataScience

Types of MySQL Variables – Local Variables

scope

the region of a computer program where a phenomenon, such as a variable, is considered valid

Types of MySQL Variables – Local Variables

scope

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN

DECLARE v_avg_salary DECIMAL(10,2);

SELECT
    AVG(s.salary)
INTO v_avg_salary FROM
    employees e
```

Types of MySQL Variables – Local Variables

scope

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN

DECLARE v_avg_salary DECIMAL(10,2);

SELECT
    AVG(s.salary)
INTO v_avg_salary FROM
    employees e
```

Types of MySQL Variables – Local Variables

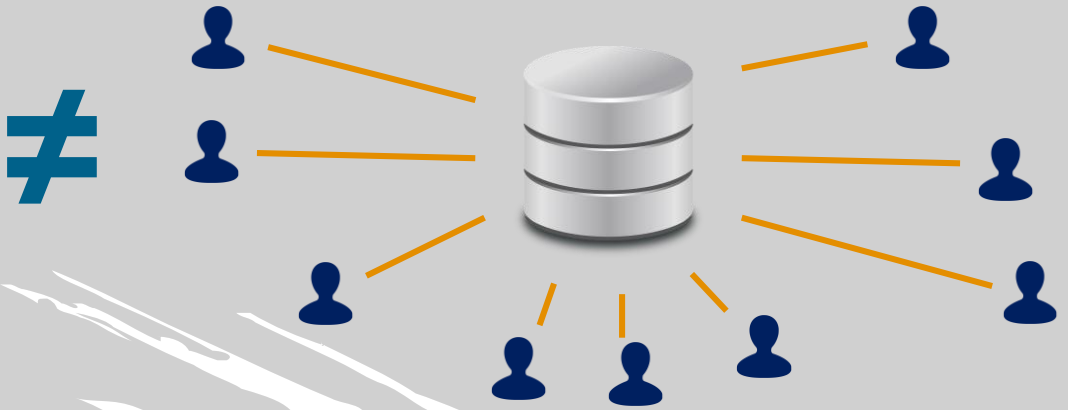
scope

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN

DECLARE v_avg_salary DECIMAL(10,2);

SELECT
    AVG(s.salary)
INTO v_avg_salary FROM
    employees e
```



Types of MySQL Variables – Local Variables

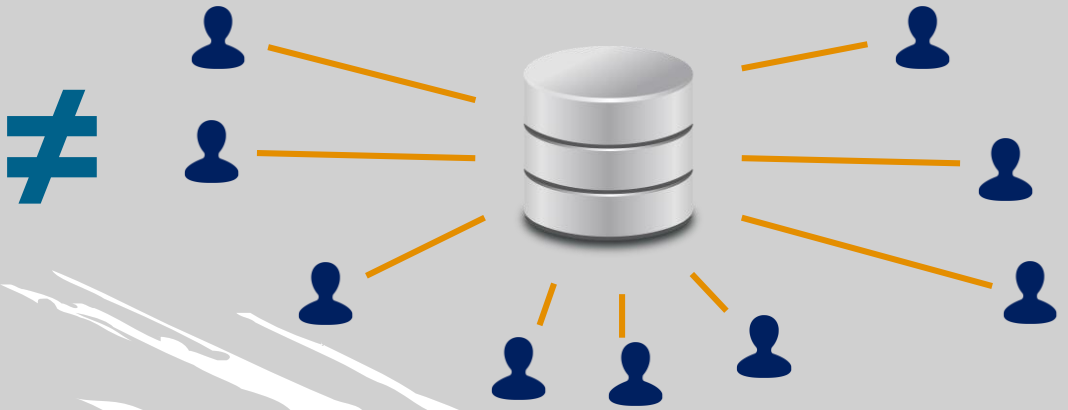
scope

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN

DECLARE v_avg_salary DECIMAL(10,2);

SELECT
    AVG(s.salary)
INTO v_avg_salary FROM
    employees e
```



365 DataScience

Types of MySQL Variables – Local Variables

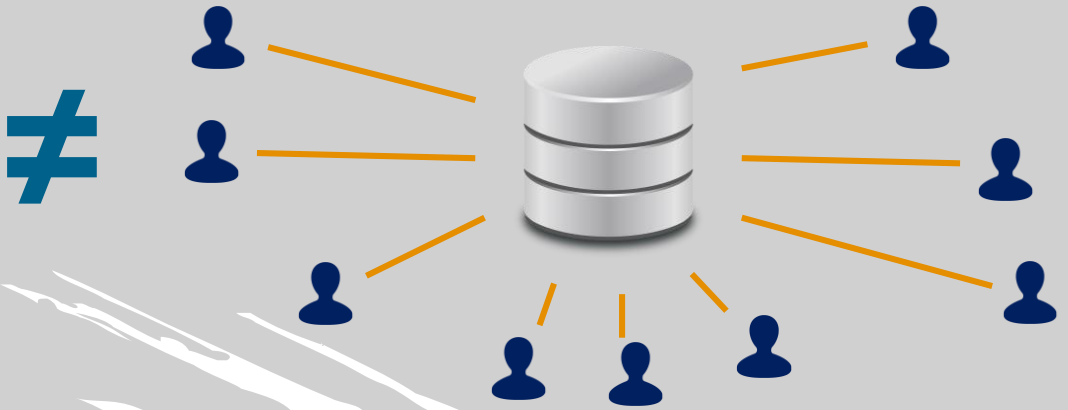
scope =

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN

DECLARE v_avg_salary DECIMAL(10,2);

SELECT
    AVG(s.salary)
INTO v_avg_salary FROM
    employees e
```



365 DataScience

Types of MySQL Variables – Local Variables

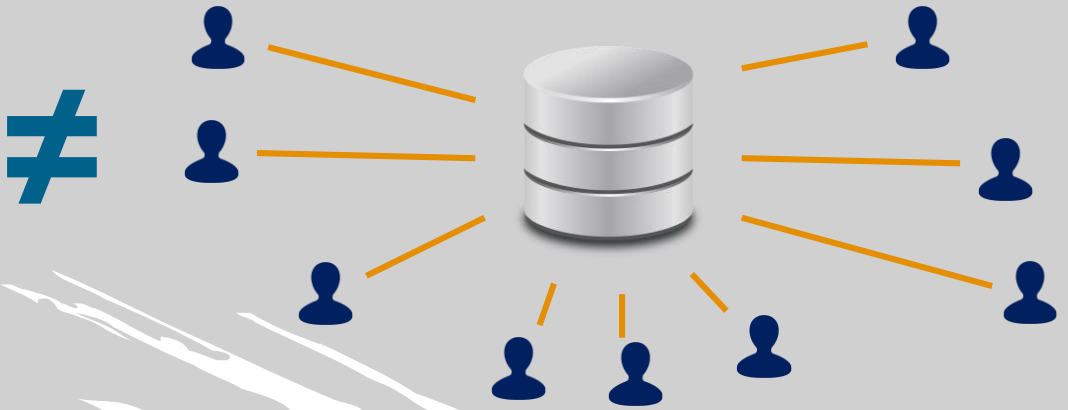
scope = visibility

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN

DECLARE v_avg_salary DECIMAL(10,2);

SELECT
    AVG(s.salary)
INTO v_avg_salary FROM
    employees e
```

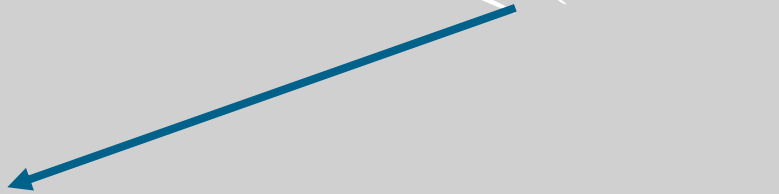


365 DataScience

Types of MySQL Variables – Local Variables

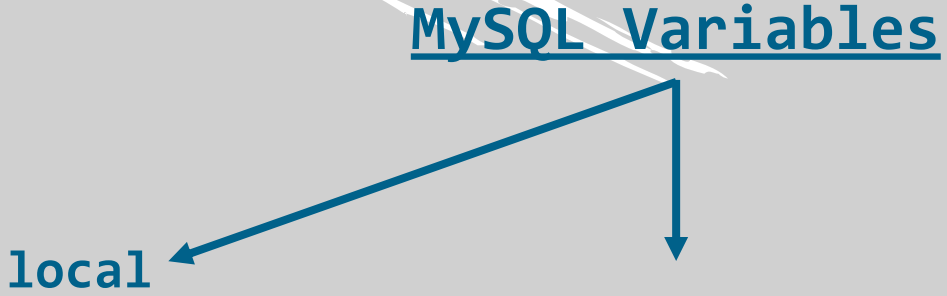
MySQL Variables

Types of MySQL Variables – Local Variables



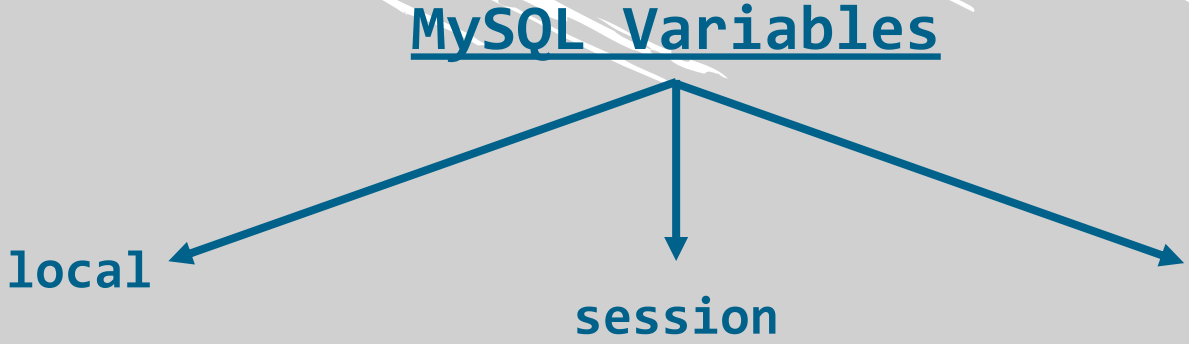
Types of MySQL Variables – Local Variables

Types of MySQL Variables – Local Variables

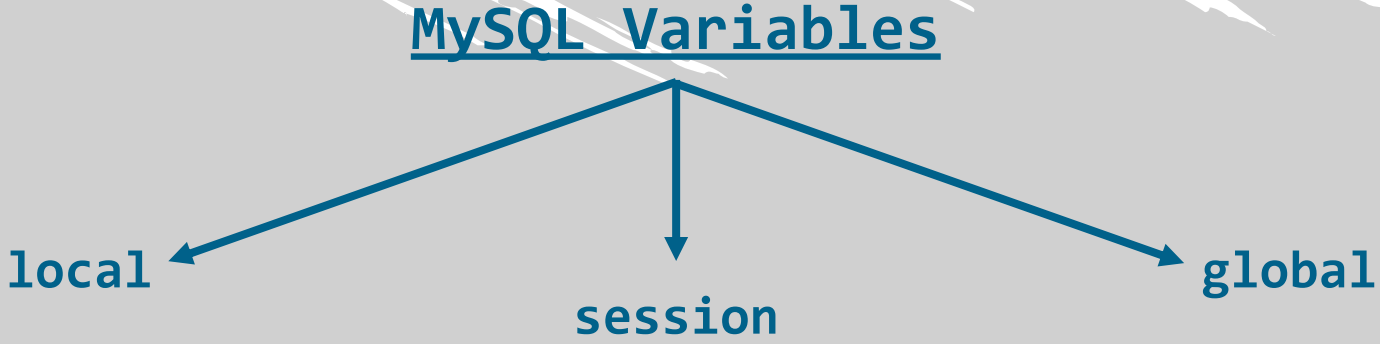


Types of MySQL Variables – Local Variables

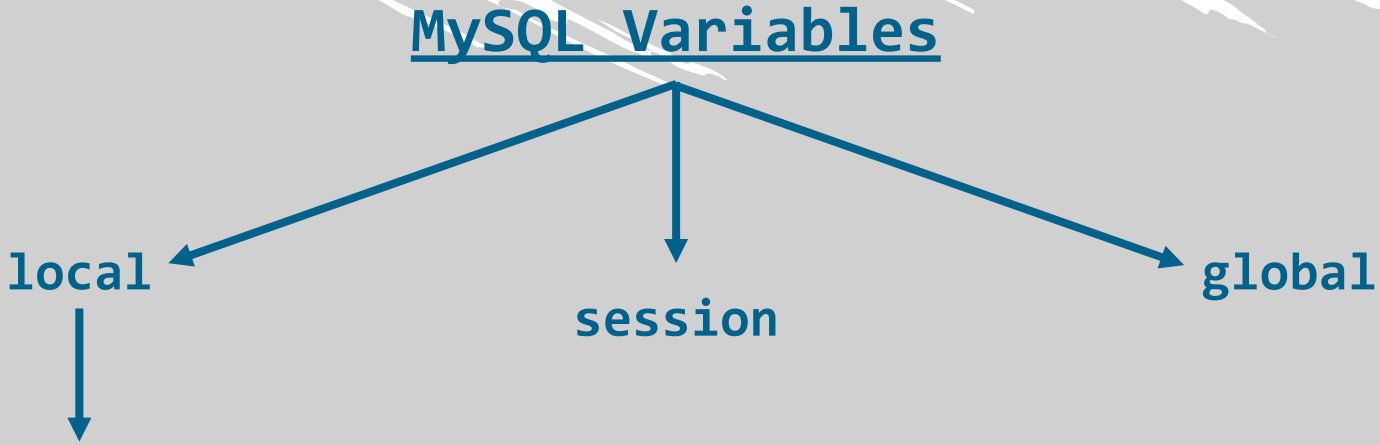
Types of MySQL Variables – Local Variables



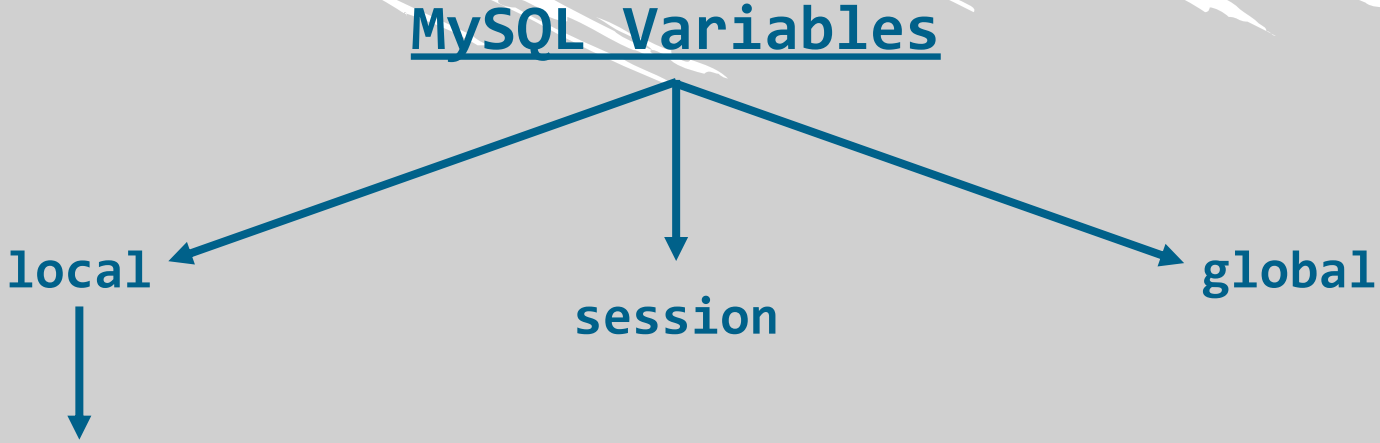
Types of MySQL Variables – Local Variables



Types of MySQL Variables – Local Variables



Types of MySQL Variables – Local Variables



local variable

a variable that is visible only in the **BEGIN – END** block in which it was created

Types of MySQL Variables – Local Variables

- DECLARE is a keyword that can be used when creating *local* variables only
- *v_avg_salary* is visible only in the BEGIN - END block

Session Variables

Session Variables

`session`

a series of information exchange interactions, or a dialogue, between a computer and a user

Session Variables

session

a series of information exchange interactions, or a dialogue, between a computer and a user

- e.g. a dialogue between the MySQL server and a client application like MySQL Workbench

Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Session Variables

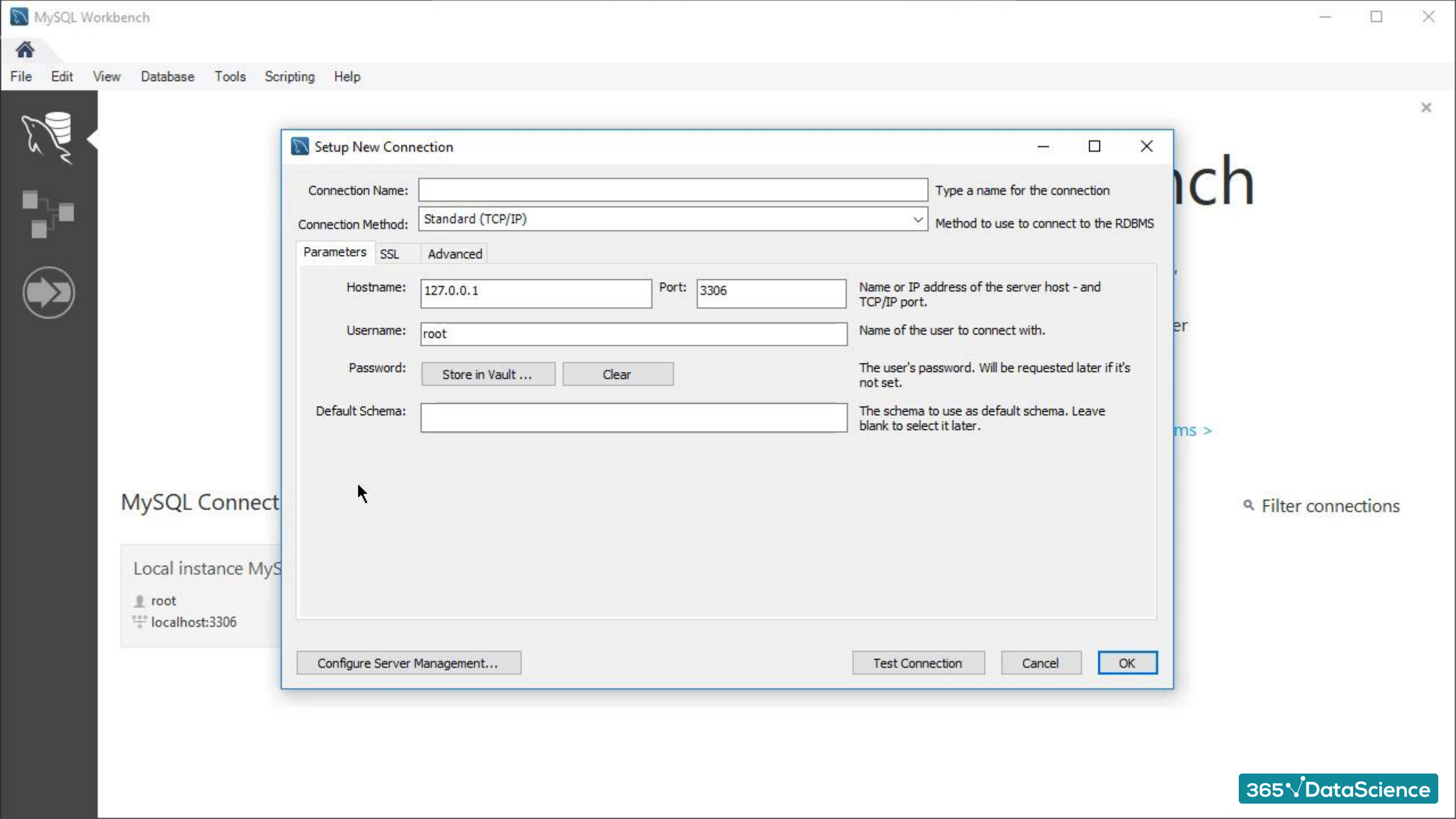
- a *session* begins at a certain point in time and terminates at another, later point

Session Variables

a *session* begins at a certain point in time and terminates at another, later point

Step 1

set up a connection


MySQL Workbench
File Edit View Database Tools Scripting Help
Setup New Connection
Connection Name:
Type a name for the connection
Connection Method:
Standard (TCP/IP)
Method to use to connect to the RDBMS
Parameters SSL Advanced
Hostname:
127.0.0.1
Port:
3306
Name or IP address of the server host - and TCP/IP port.
Username:
root
Name of the user to connect with.
Password:
Store in Vault ...
Clear
The user's password. Will be requested later if it's not set.
Default Schema:
The schema to use as default schema. Leave blank to select it later.
Configure Server Management...
Test Connection
Cancel
OK
MySQL Connect
Local instance MyS
root
localhost:3306
Filter connections
365 DataScience

Session Variables

a *session* begins at a certain point in time and terminates at another, later point

Step 1

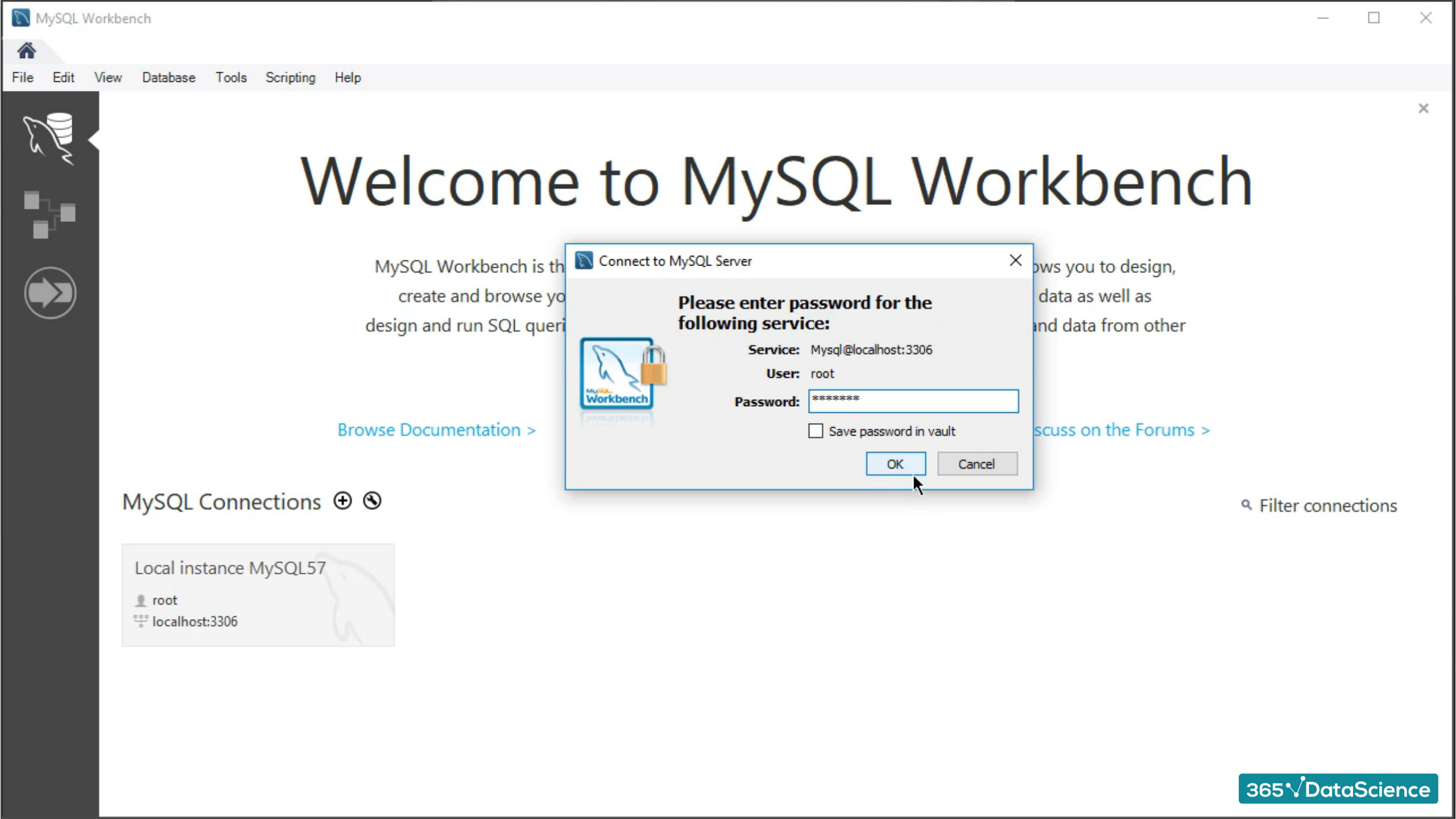
set up a connection

Session Variables

a *session* begins at a certain point in time and terminates at another, later point

Step 1

set up a connection


MySQL Workbench
File Edit View Database Tools Scripting Help
Welcome to MySQL Workbench
MySQL Workbench is th
ows you to design,
create and browse yo
data as well as
design and run SQL queri
nd data from other
Browse Documentation >
scuss on the Forums >
Connect to MySQL Server
Please enter password for the following service:
Service: Mysql@localhost:3306
User: root
Password: *******
Save password in vault
OK
Cancel
MySQL Connections
Filter connections
Local instance MySQL57
root
localhost:3306
365 DataScience

Session Variables

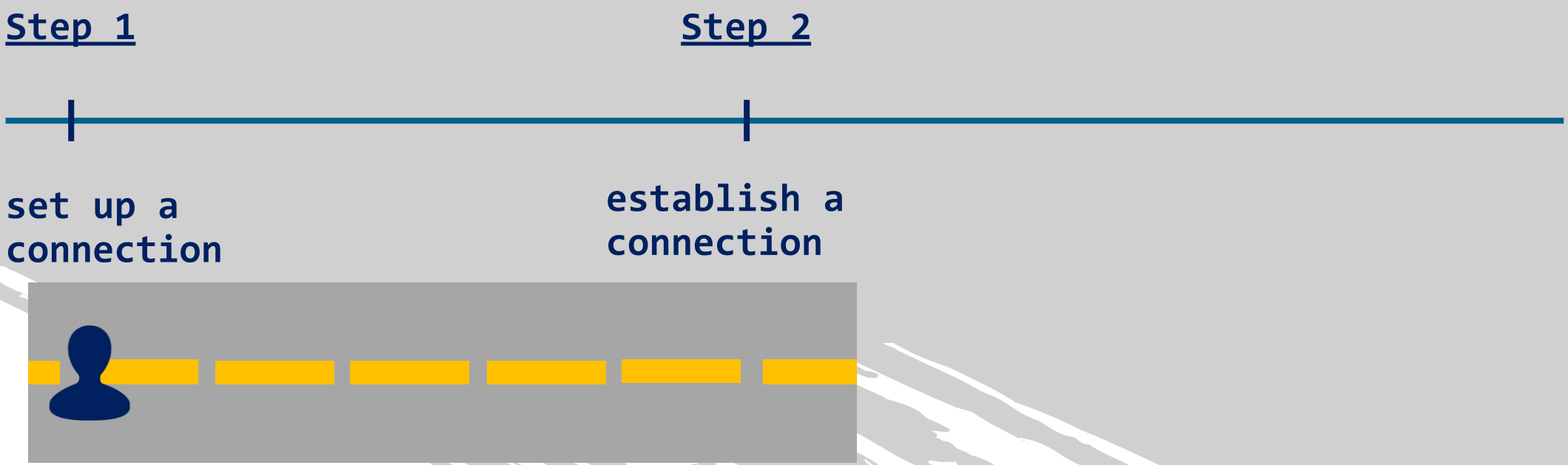
a *session* begins at a certain point in time and terminates at another, later point

Step 1

set up a connection

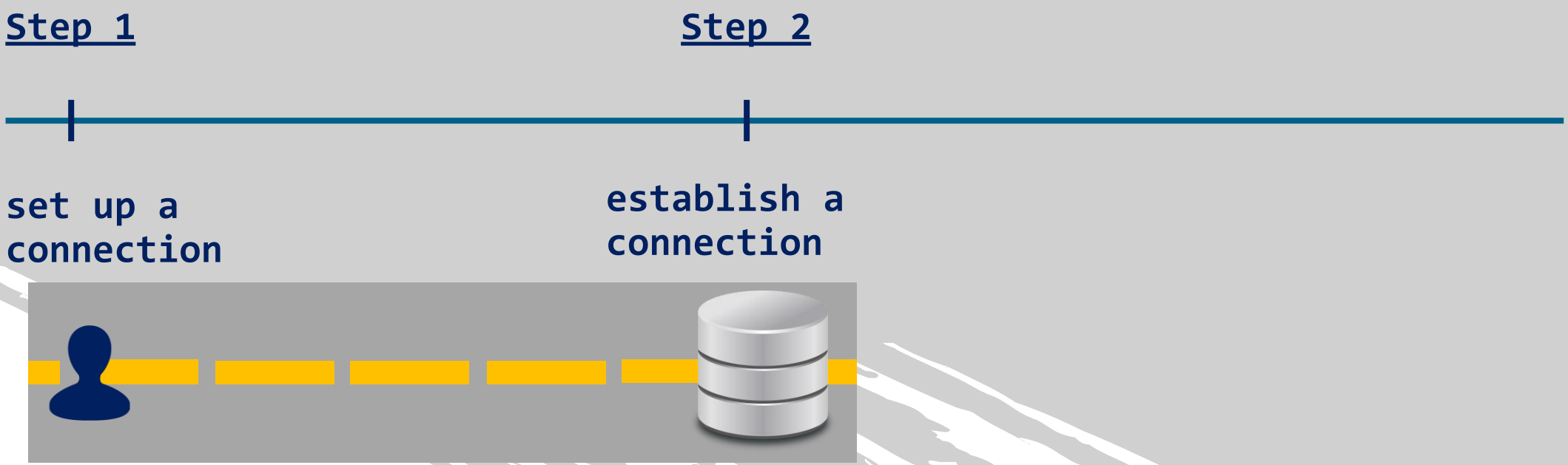
Session Variables

a *session* begins at a certain point in time and terminates at another, later point



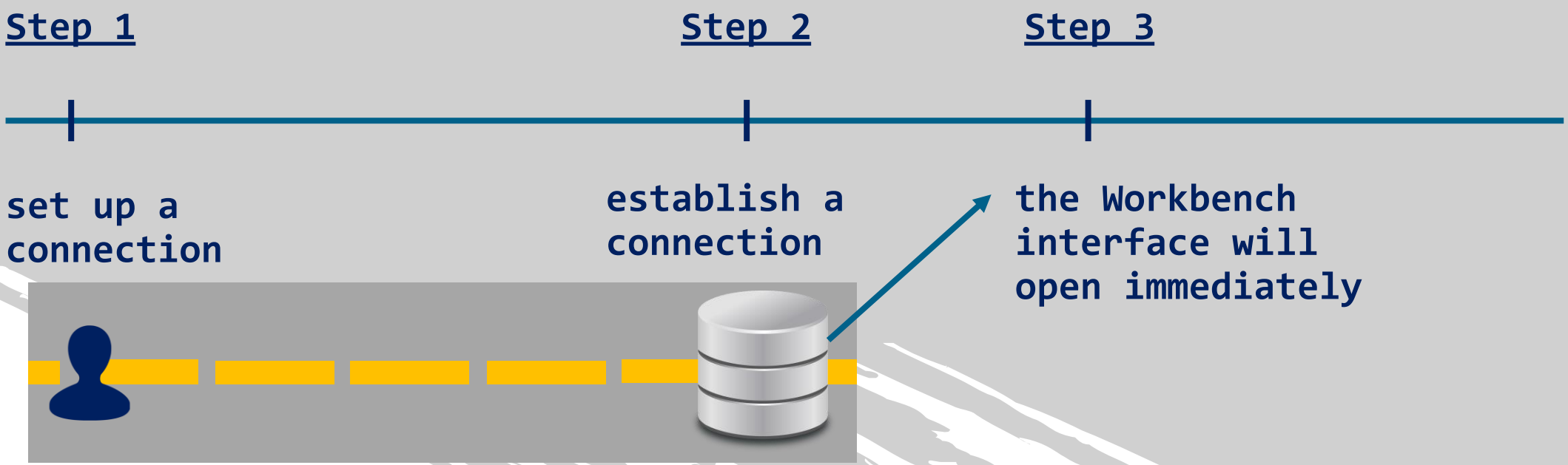
Session Variables

- a *session* begins at a certain point in time and terminates at another, later point



Session Variables

a *session* begins at a certain point in time and terminates at another, later point



Session Variables

a *session* begins at a certain point in time and terminates at another, later point

Step 1

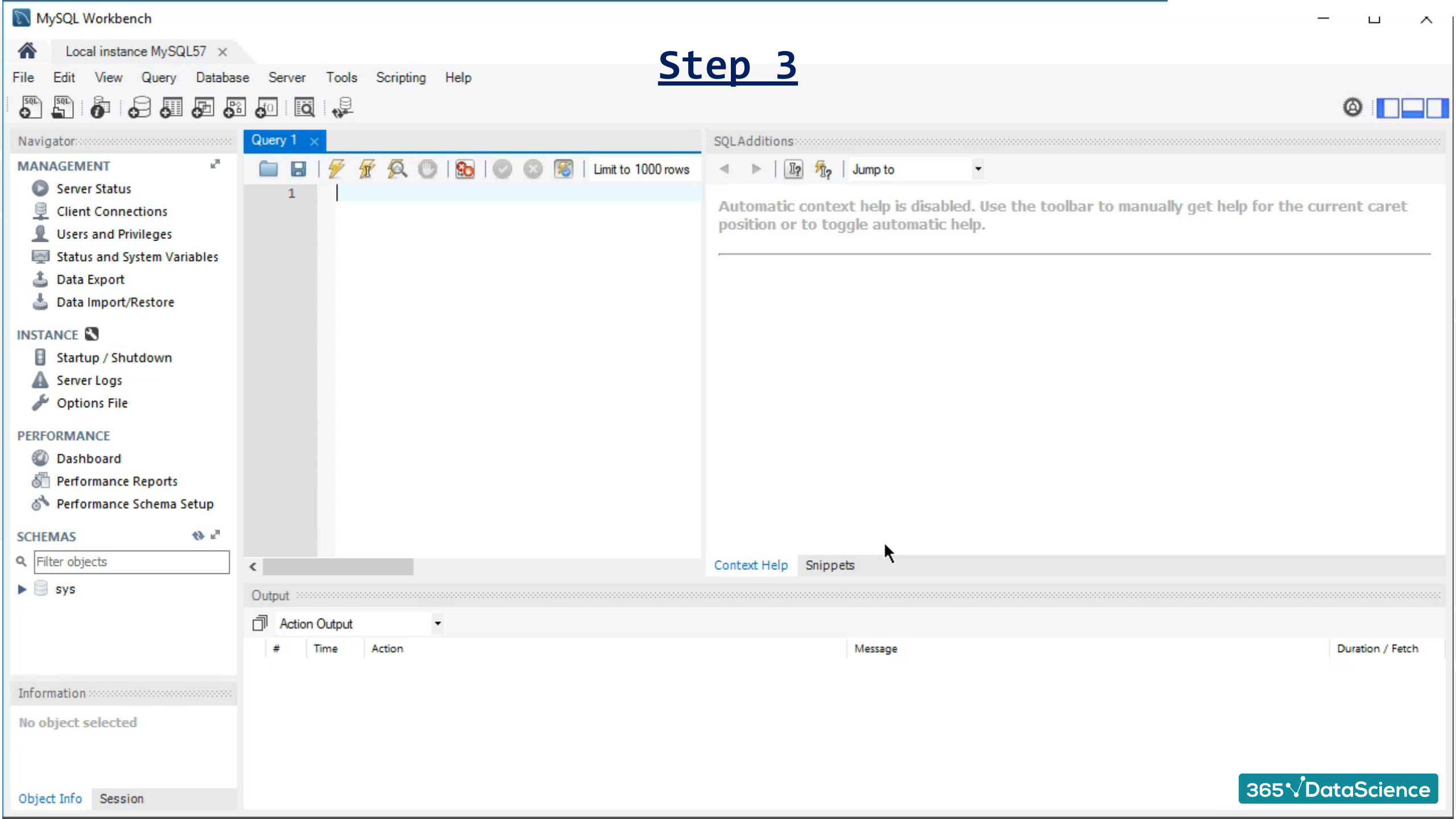
set up a connection

Step 2

establish a connection

Step 3

the Workbench interface will open immediately


MySQL Workbench
Local instance MySQL57
Step 3
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
SCHEMAS
Filter objects
sys
Information
No object selected
Object Info
Session
Query 1
Limit to 1000 rows
1
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
365 DataScience

Session Variables

a *session* begins at a certain point in time and terminates at another, later point

Step 1

set up a connection

Step 2

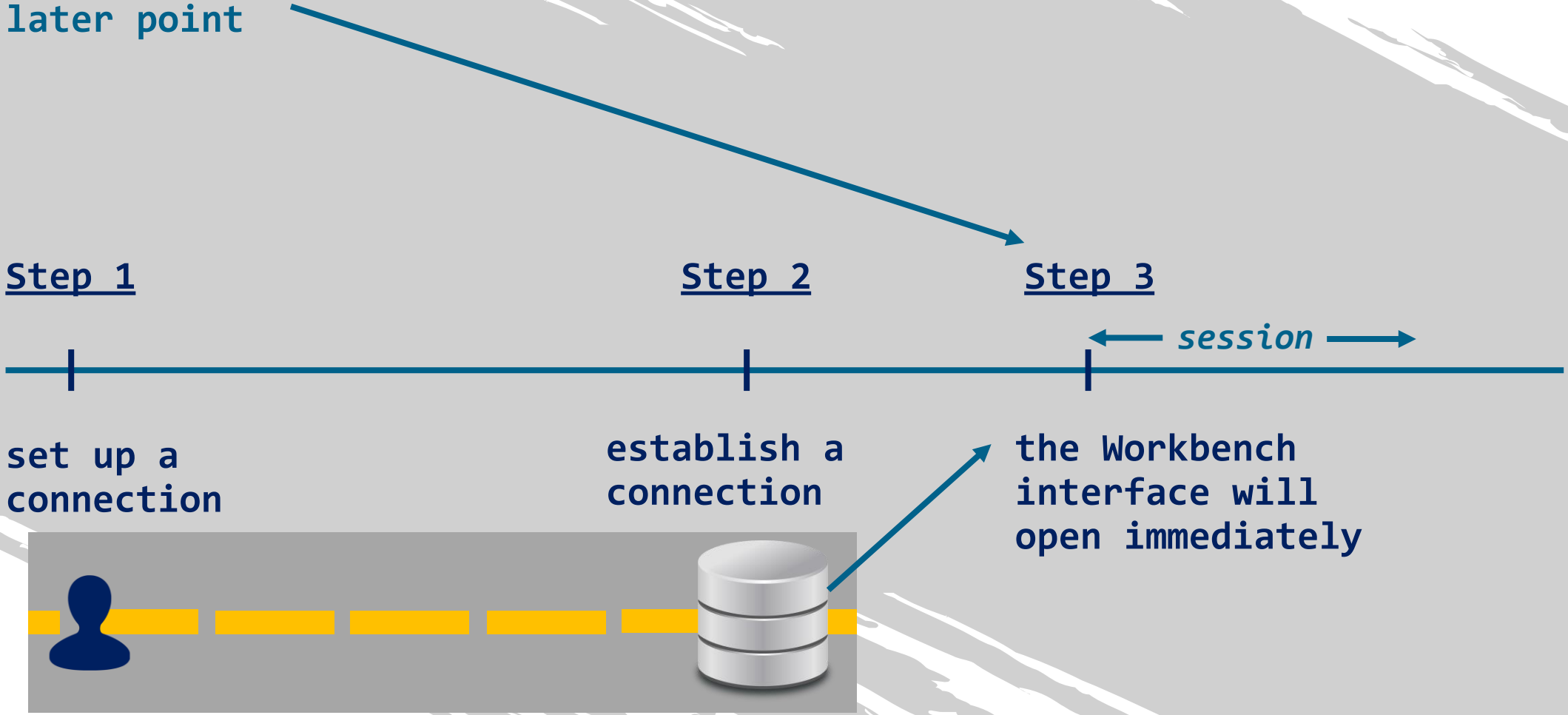
establish a connection

Step 3

the Workbench interface will open immediately

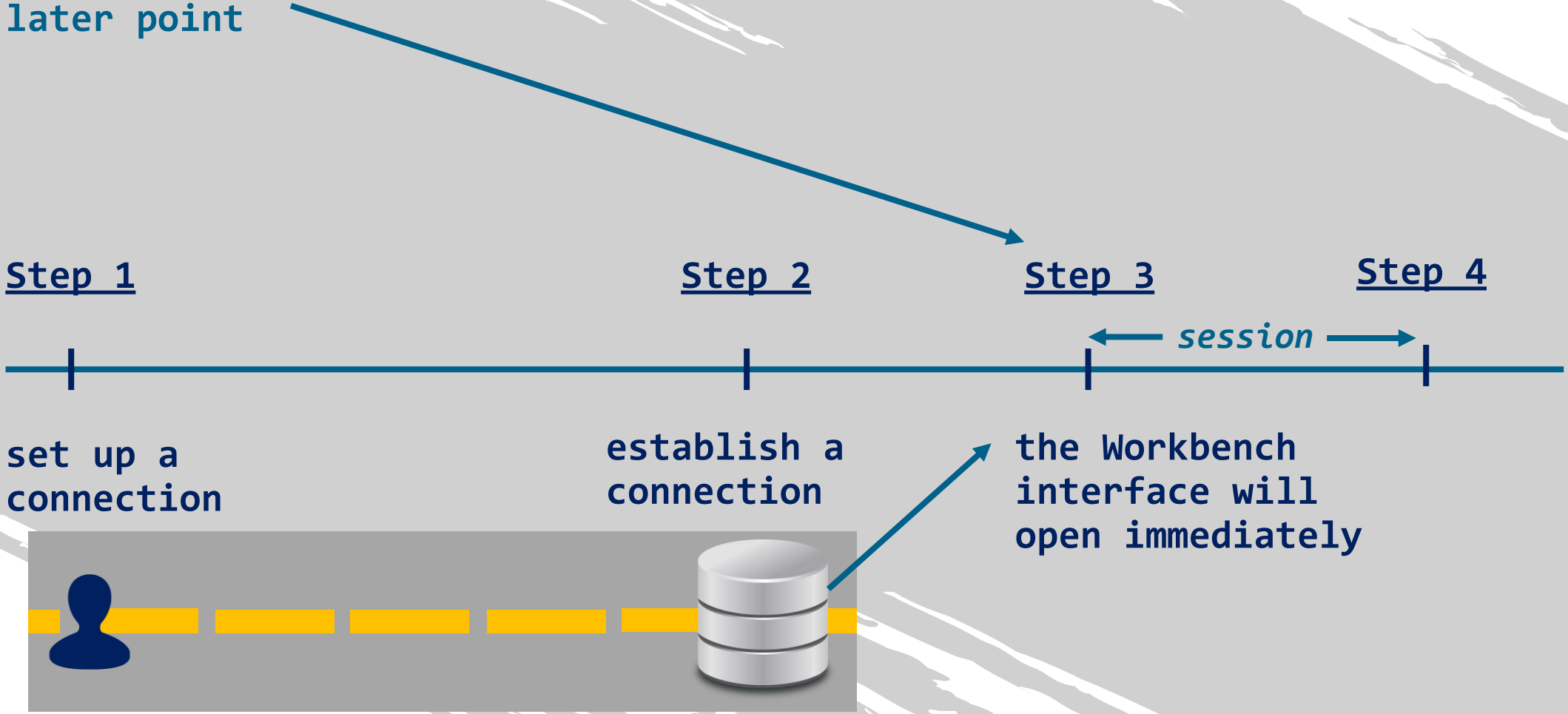
Session Variables

a *session* begins at a certain point in time and terminates at another, later point



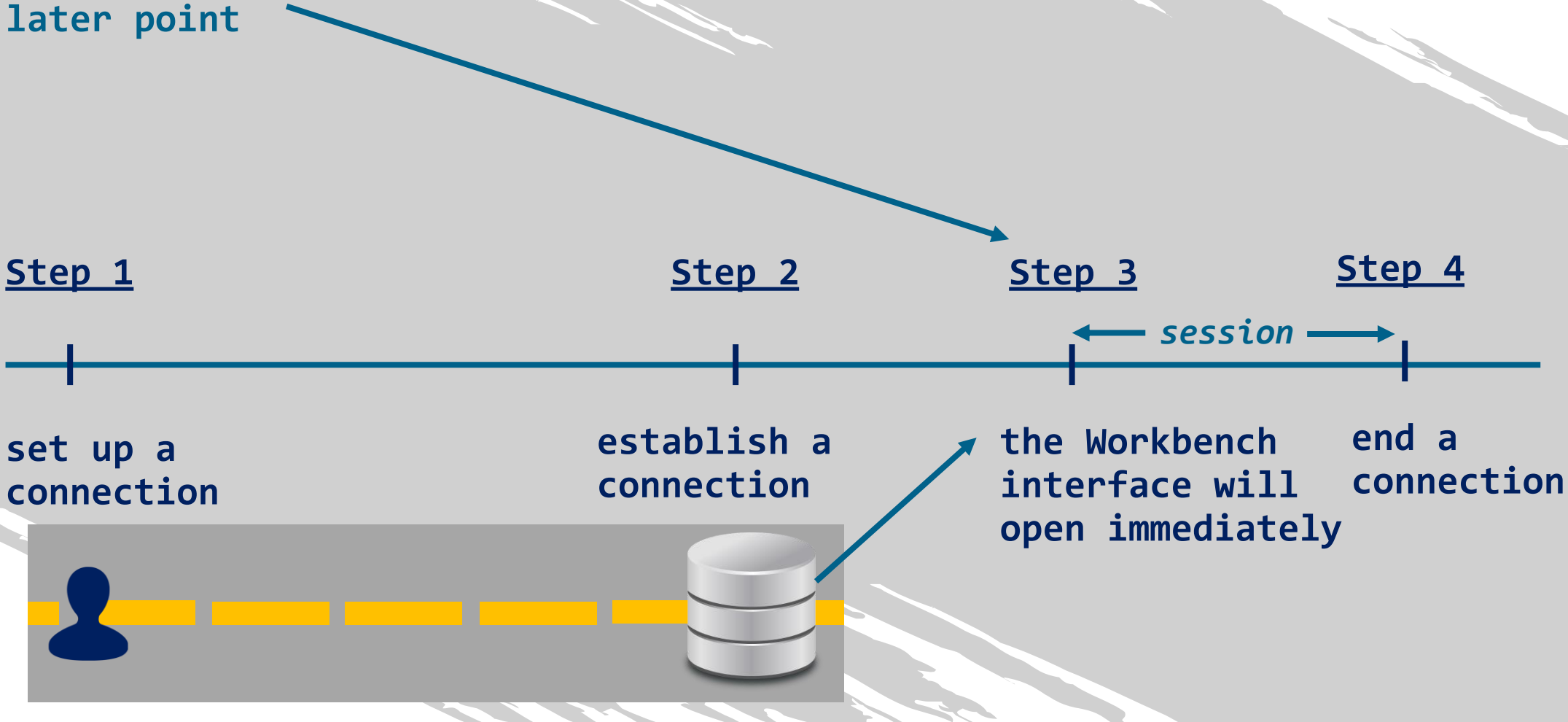
Session Variables

a *session* begins at a certain point in time and terminates at another, later point



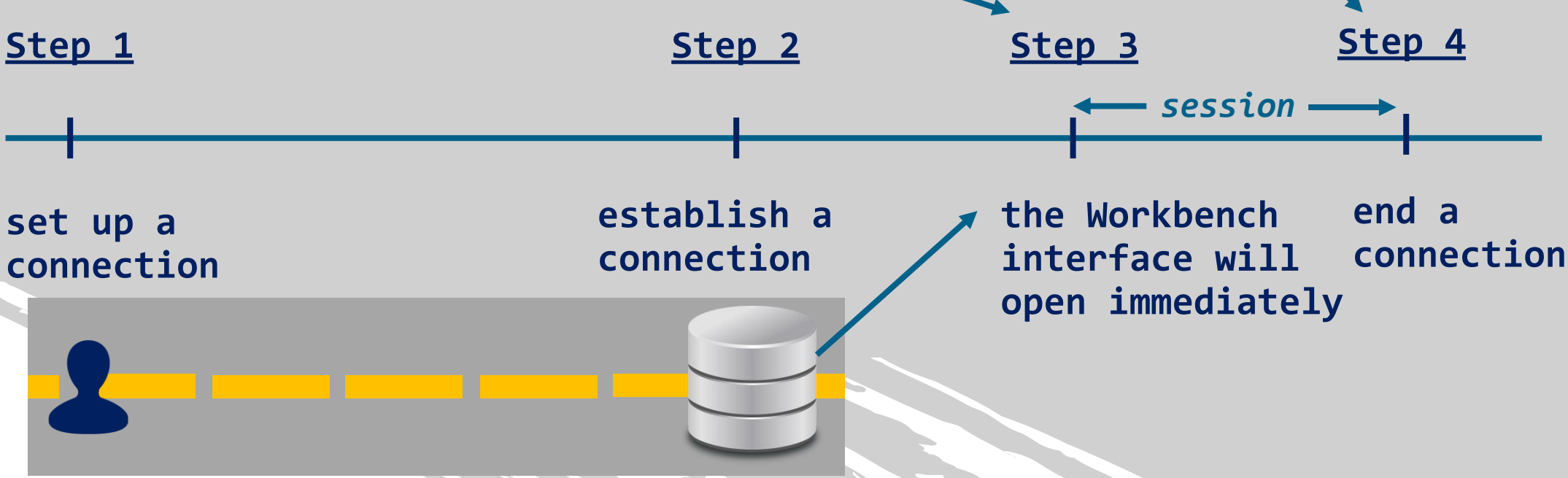
Session Variables

a *session* begins at a certain point in time and terminates at another, later point



Session Variables

a *session* begins at a certain point in time and terminates at another, later point



Session Variables

a *session* begins at a certain point in time and terminates at another, later point

Step 1

Step 2

Step 3

Step 4

set up a connection

establish a connection

the Workbench interface will open immediately

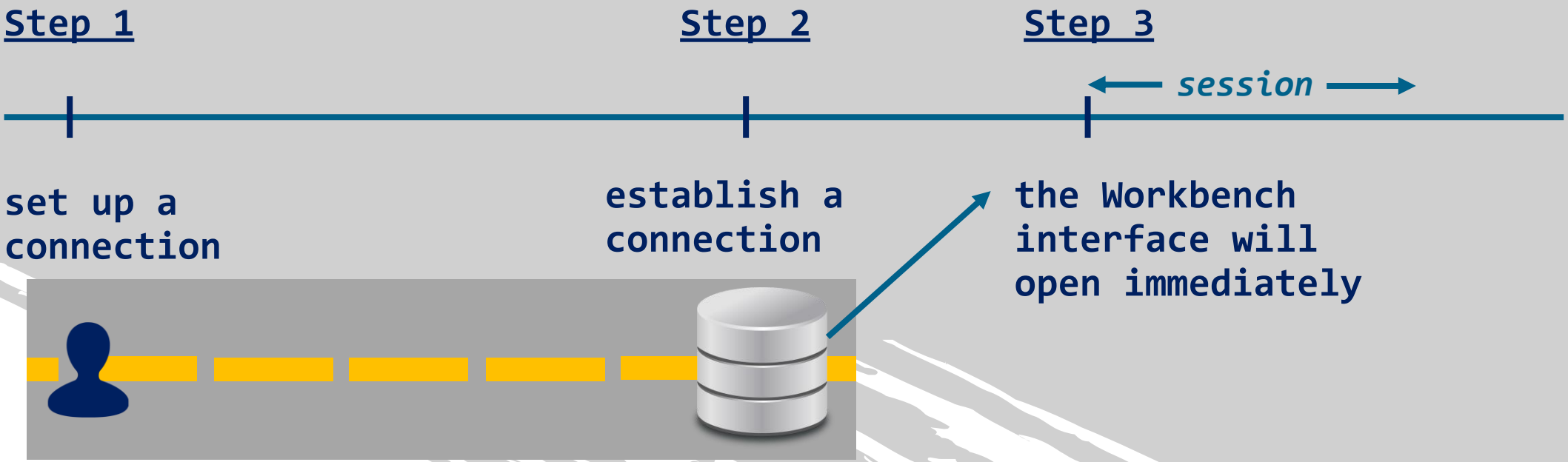
end a connection

Session Variables

- there are certain SQL objects that are valid for a specific session only

Session Variables

- there are certain SQL objects that are valid for a specific session only



Session Variables

there are certain SQL objects that are valid for a specific session only

emp_no	from_date	to_date
10001	1986-06-26	9999-01-01
10002	1996-08-03	9999-01-01
10003	1995-12-03	9999-01-01
10004	1986-12-01	9999-01-01
10005	1989-09-12	9999-01-01

Step 1

Step 2

session

set up a connection

establish a connection

the Workbench interface will open immediately

Session Variables

there are certain SQL objects that are valid for a specific session only

emp_no	from_date	to_date
10001	1986-06-26	9999-01-01
10002	1996-08-03	9999-01-01
10003	1995-12-03	9999-01-01
10004	1986-12-01	9999-01-01
10005	1989-09-12	9999-01-01

Step 1

Step 2

session

set up a connection

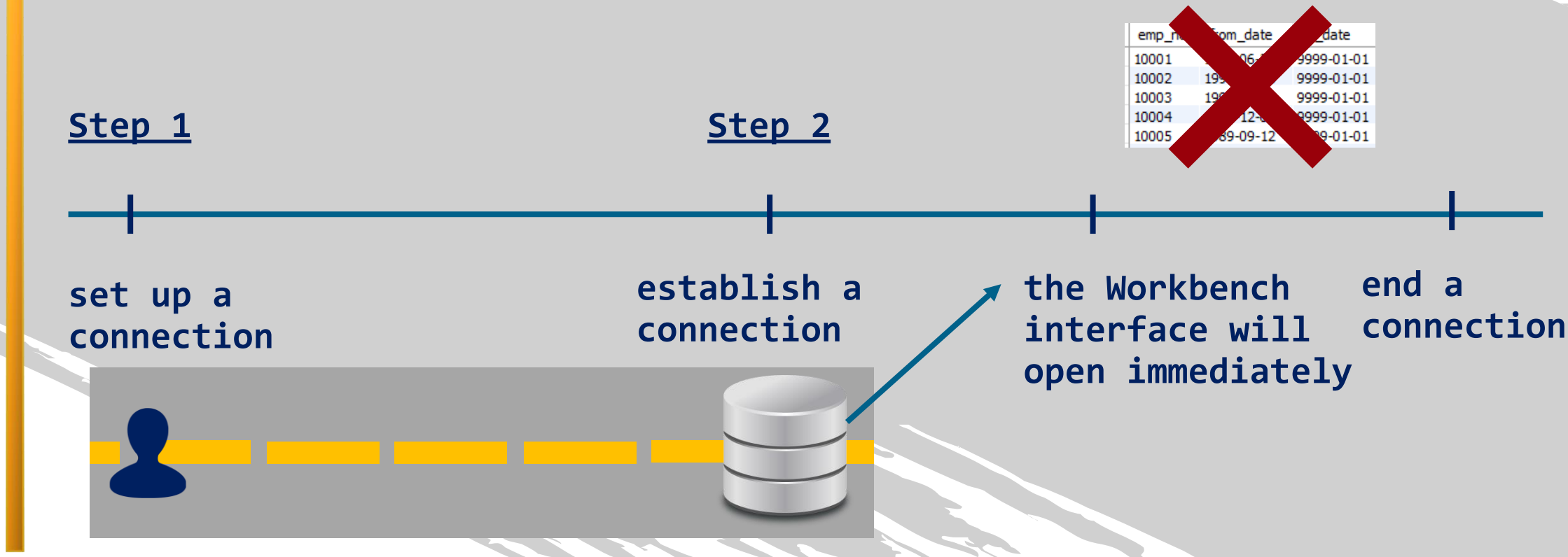
establish a connection

the Workbench interface will open immediately

end a connection

Session Variables

there are certain SQL objects that are valid for a specific session only



Session Variables

session variable

a variable that exists only for the session in which you are operating

Session Variables

session variable

a variable that exists only for the session in which you are operating

- it is *defined* on our server, and it lives there

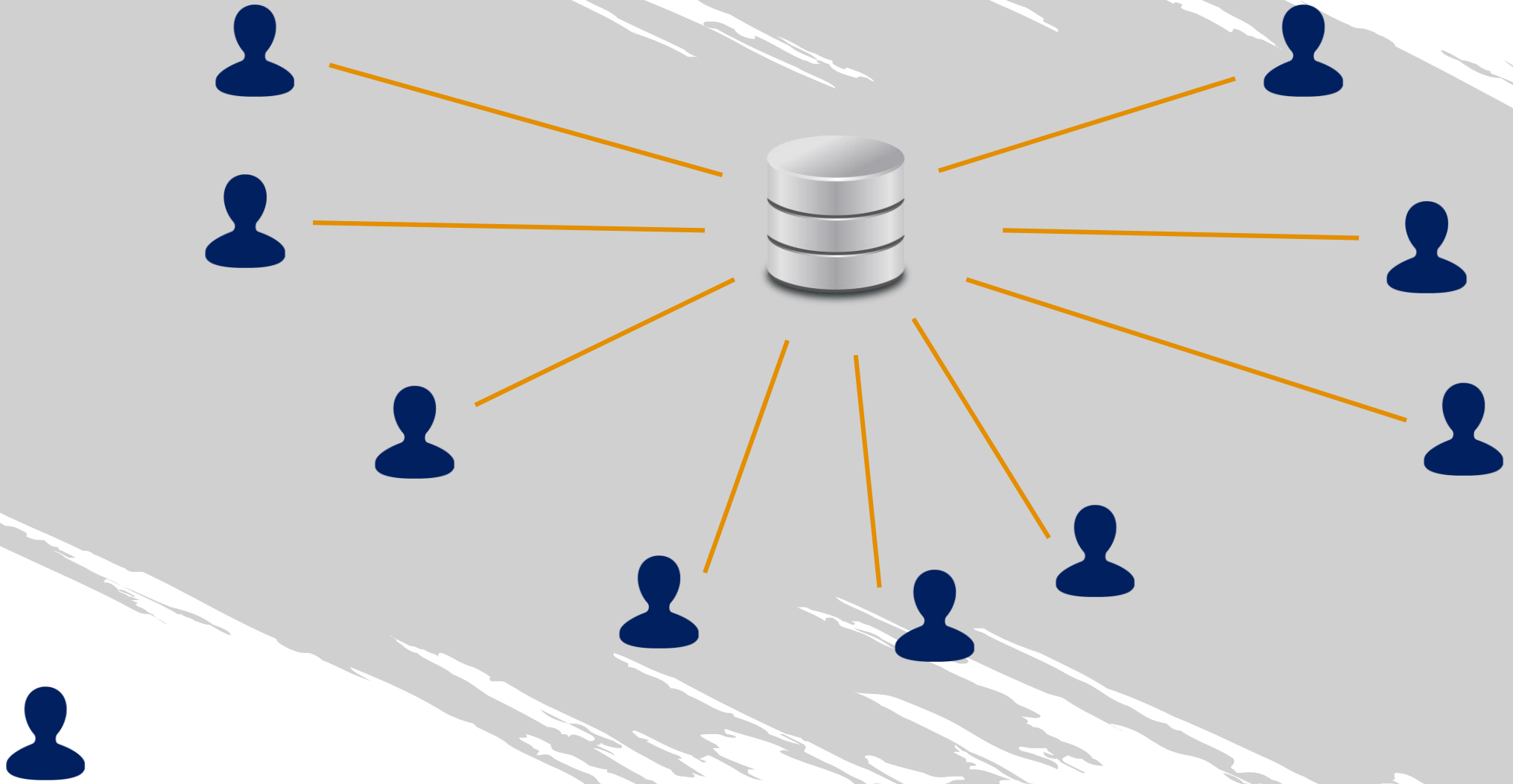
Session Variables

session variable

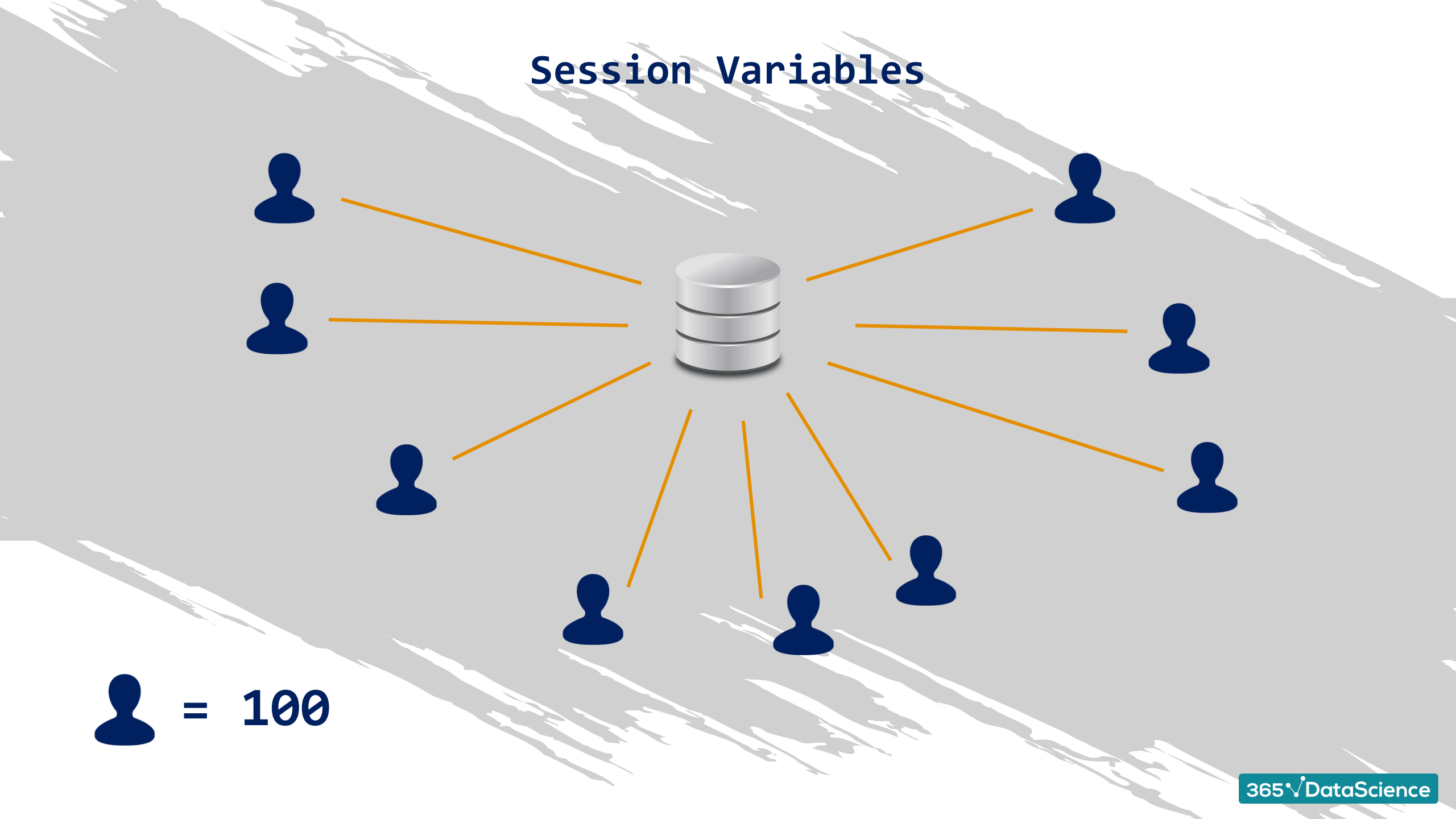
a variable that exists only for the session in which you are operating

- it is *defined* on our server, and it lives there
- it is *visible* to the connection being used only

Session Variables

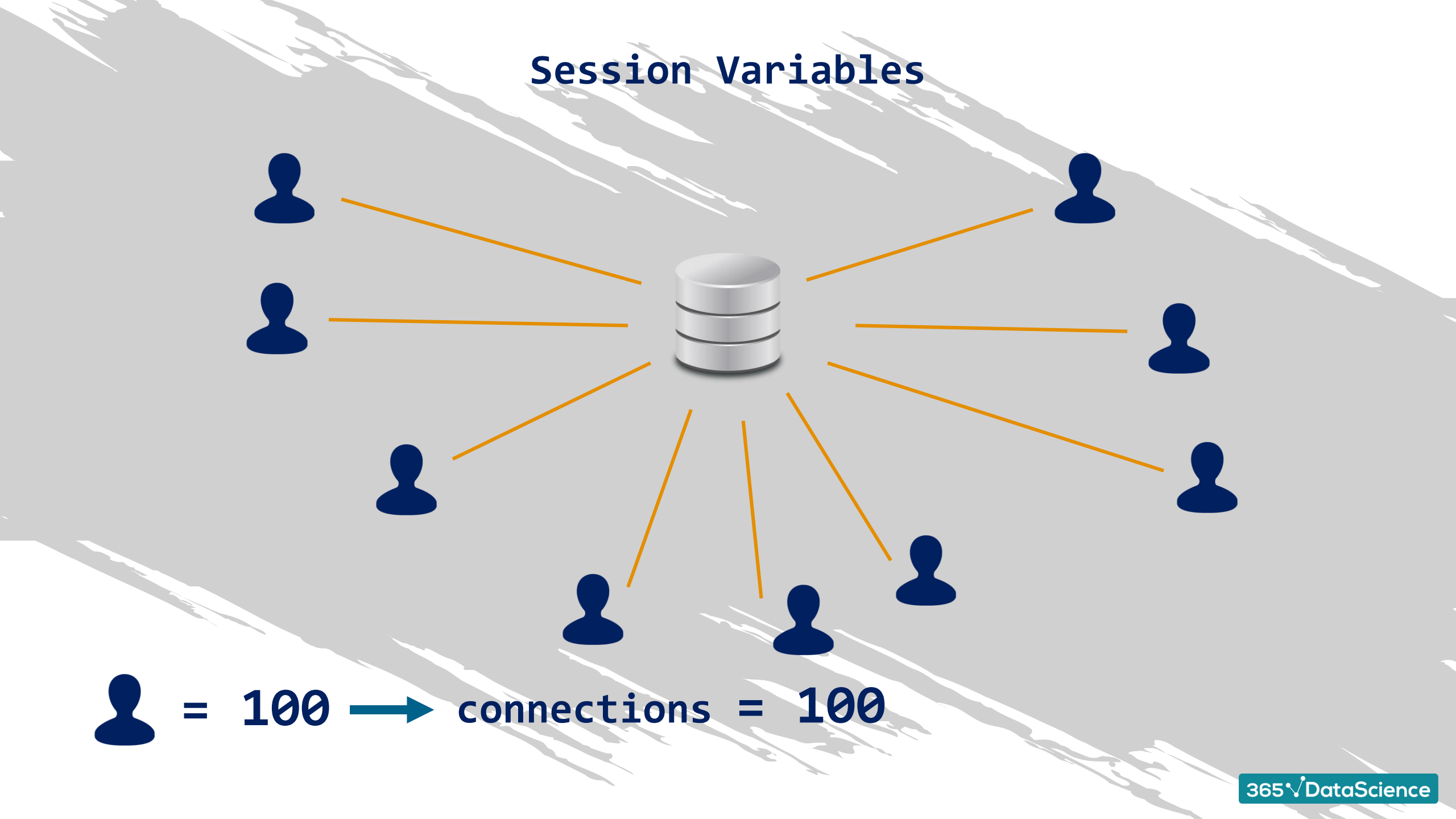


Session Variables



= 100

Session Variables



= 100 → connections = 100

Session Variables

= 100 → connections = 100 → sessions = 100

Session Variables

= 100 → connections = 100 → sessions = 100

Session Variables

= 100 → connections = 100 → sessions = 100

Session Variables


```SQL
SET @var_name = value;
```

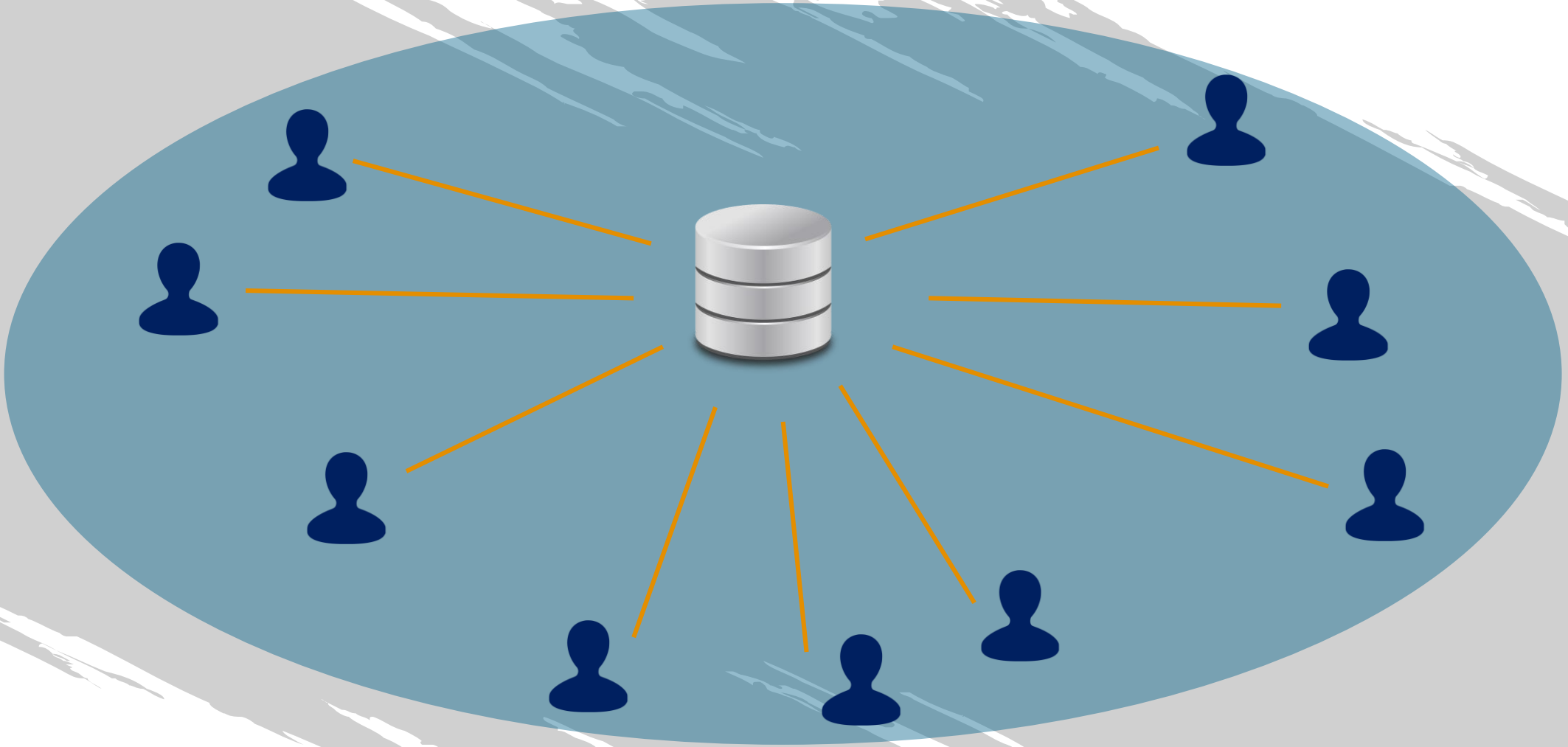

Global Variables
365 DataScience

Global Variables

`global variables`

global variables apply to all connections related to a specific server

Global Variables



Global Variables

Global Variables


```
SET GLOBAL var_name = value;
```

Global Variables


```sql
SET GLOBAL var_name = value;
```


```sql
SET @@global.var_name = value;
```

Global Variables

- you cannot set just any variable as global

Global Variables

- you cannot set just any variable as global

  - a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables

Global Variables

- you cannot set just any variable as global

  - a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables

Global Variables

you cannot set just any variable as global

- a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables

---

```
.max_connections()
```

Global Variables

you cannot set just any variable as global

- a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables

---

`.max_connections()`

`.max_join_size()`

Global Variables

you cannot set just any variable as global

- a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables

---

`.max_connections()` - indicates the *maximum number of connections* to a server that can be established at a certain point in time

`.max_join_size()`

Global Variables

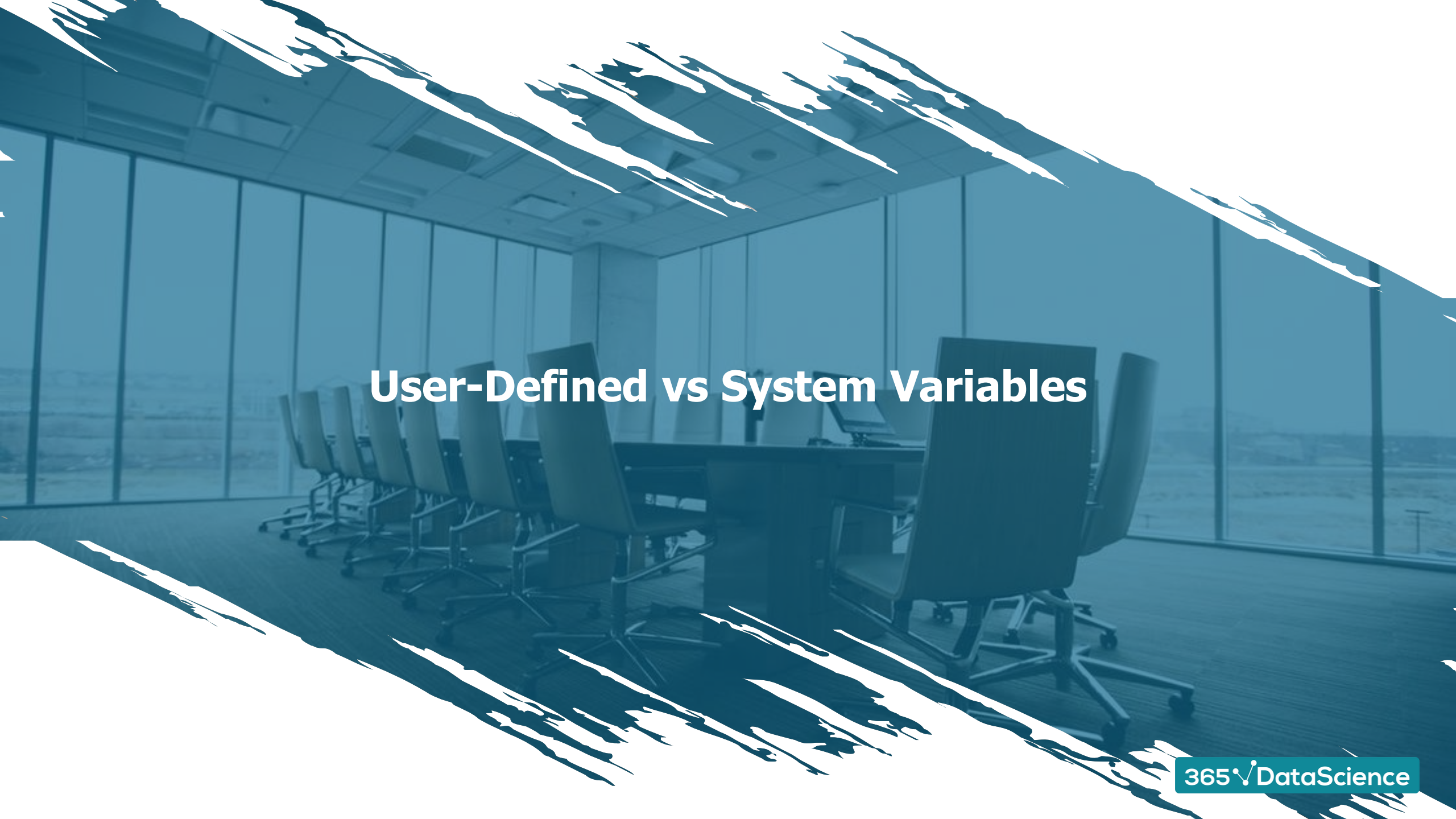
you cannot set just any variable as global

- a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables

---

`.max_connections()` - indicates the *maximum number of connections* to a server that can be established at a certain point in time

`.max_join_size()` - sets the *maximum memory space* allocated for the joins created by a certain connection


User-Defined vs System Variables
365 DataScience

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

`user-defined`

User-Defined vs System Variables

variables in MySQL can be characterized according to the way they have been created

- user-defined
- system

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

user-defined — variables that can be *set by the user manually*

system

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

user-defined — variables that can be *set by the user manually*

system — variables that are *pre-defined* on our *system* – the MySQL server

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global	
user-defined	✓			
system				

- local variables can be user-defined only

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global	
user-defined	✓			
system	✗			

- local variables can be user-defined only

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global	
user-defined	✓			
system	✗		✓	

- only system variables can be set as global

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global	
user-defined	✓		✗	
system	✗		✓	

- only system variables can be set as global

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global	
user-defined	✓		✗	
system	✗		✓	

- both user-defined and system variables can be set as session variables

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global	
user-defined	✓	✓	✗	
system	✗	✓	✓	

- both user-defined and system variables can be set as session variables

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global	
user-defined	✓	✓	✗	
system	✗	✓	✓	

- both user-defined and system variables can be set as session variables

there are limitations to this rule!

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global	
user-defined	✓	✓*	✗	
system	✗	✓*	✓	

- both user-defined and system variables can be set as session variables

there are limitations to this rule!

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global	
user-defined	✓	✓*	✗	
system	✗	✓*	✓	

- some of the system variables can be defined as global only; they cannot be session variables

User-Defined vs System Variables

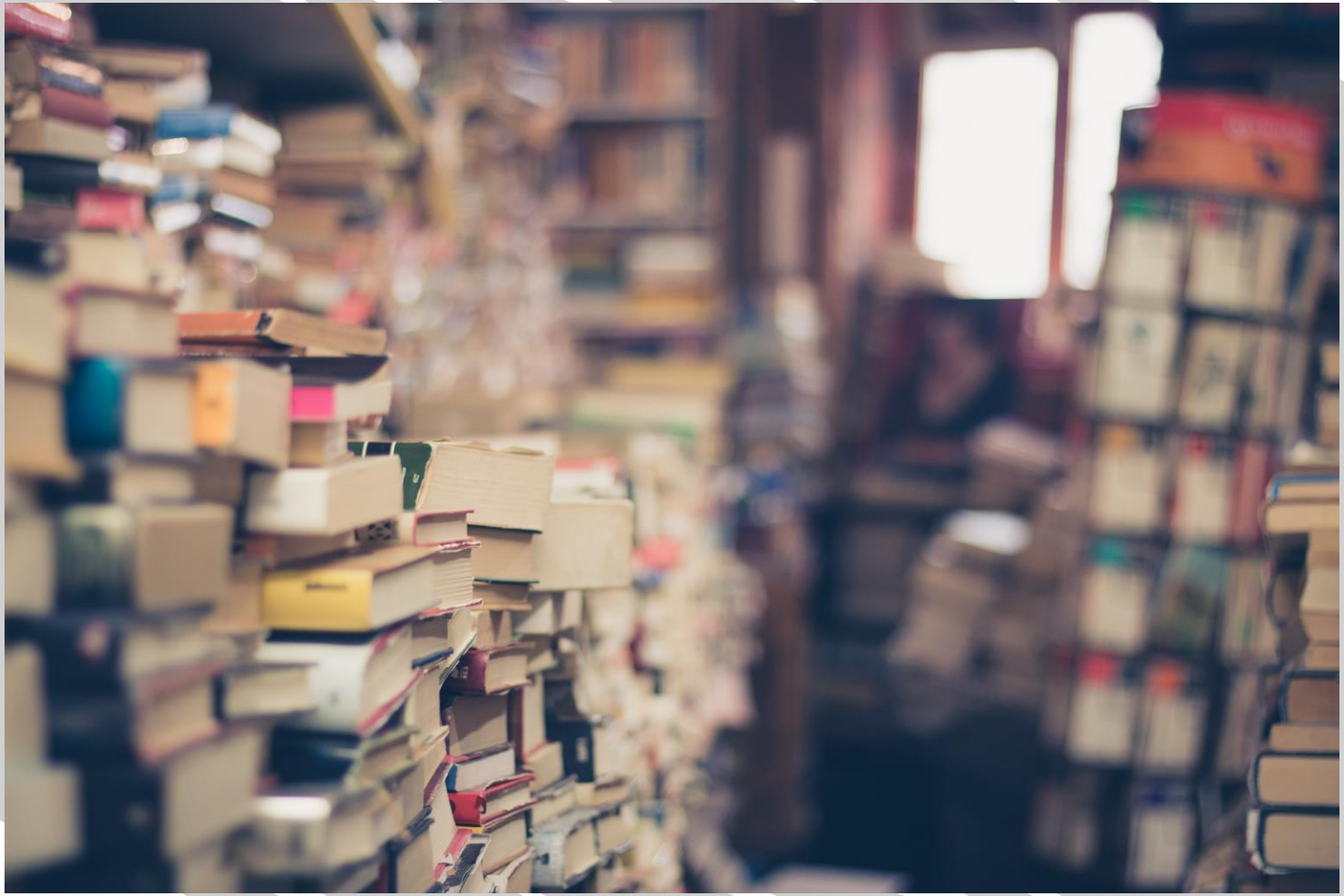
- variables in MySQL can be characterized according to the way they have been created

	local	session	global	
user-defined	✓	✓*	✗	
system	✗	✓*	✓	

- some of the system variables can be defined as global only; they cannot be session variables (e.g. `.max_connections()`)


MySQL Indexes
365 DataScience

MySQL Indexes



MySQL Indexes

MySQL Indexes

- the index of a table functions like the index of a book

MySQL Indexes

- the index of a table functions like the index of a book

  - data is taken from a column of the table and is stored in a certain order in a distinct place, called an index

MySQL Indexes

- the index of a table functions like the index of a book

  - data is taken from a column of the table and is stored in a certain order in a distinct place, called an index

- your datasets will typically contain 100,000+ or even 1,000,000+ records

MySQL Indexes

- the index of a table functions like the index of a book

  - data is taken from a column of the table and is stored in a certain order in a distinct place, called an index

- your datasets will typically contain 100,000+ or even 1,000,000+ records

  - the *larger* a database is, the *slower* the process of finding the record or records you need

MySQL Indexes

- we can use an index that will increase the speed of searches related to a table

MySQL Indexes

- we can use an index that will increase the speed of searches related to a table


```
CREATE INDEX index_name
ON table_name (column_1, column_2, …);
```

MySQL Indexes

- we can use an index that will increase the speed of searches related to a table

SQL

```
CREATE INDEX index_name
ON table_name (column_1, column_2, …);
```

the parentheses serve us to indicate the column names on which our search will be based

MySQL Indexes

- we can use an index that will increase the speed of searches related to a table

SQL

```
CREATE INDEX index_name
ON table_name (column_1, column_2, …);
```

these must be fields from your data table you will search frequently

MySQL Indexes

- composite indexes

MySQL Indexes

composite indexes

- applied to *multiple* columns, not just a single one

MySQL Indexes

composite indexes

- applied to *multiple* columns, not just a single one


```SQL
CREATE INDEX index_name
ON table_name (column_1, column_2, …);
```

MySQL Indexes

composite indexes

- applied to *multiple* columns, not just a single one


```SQL
CREATE INDEX index_name
ON table_name (column_1, column_2, …);
```

- carefully pick the columns that would optimize your search!

MySQL Indexes

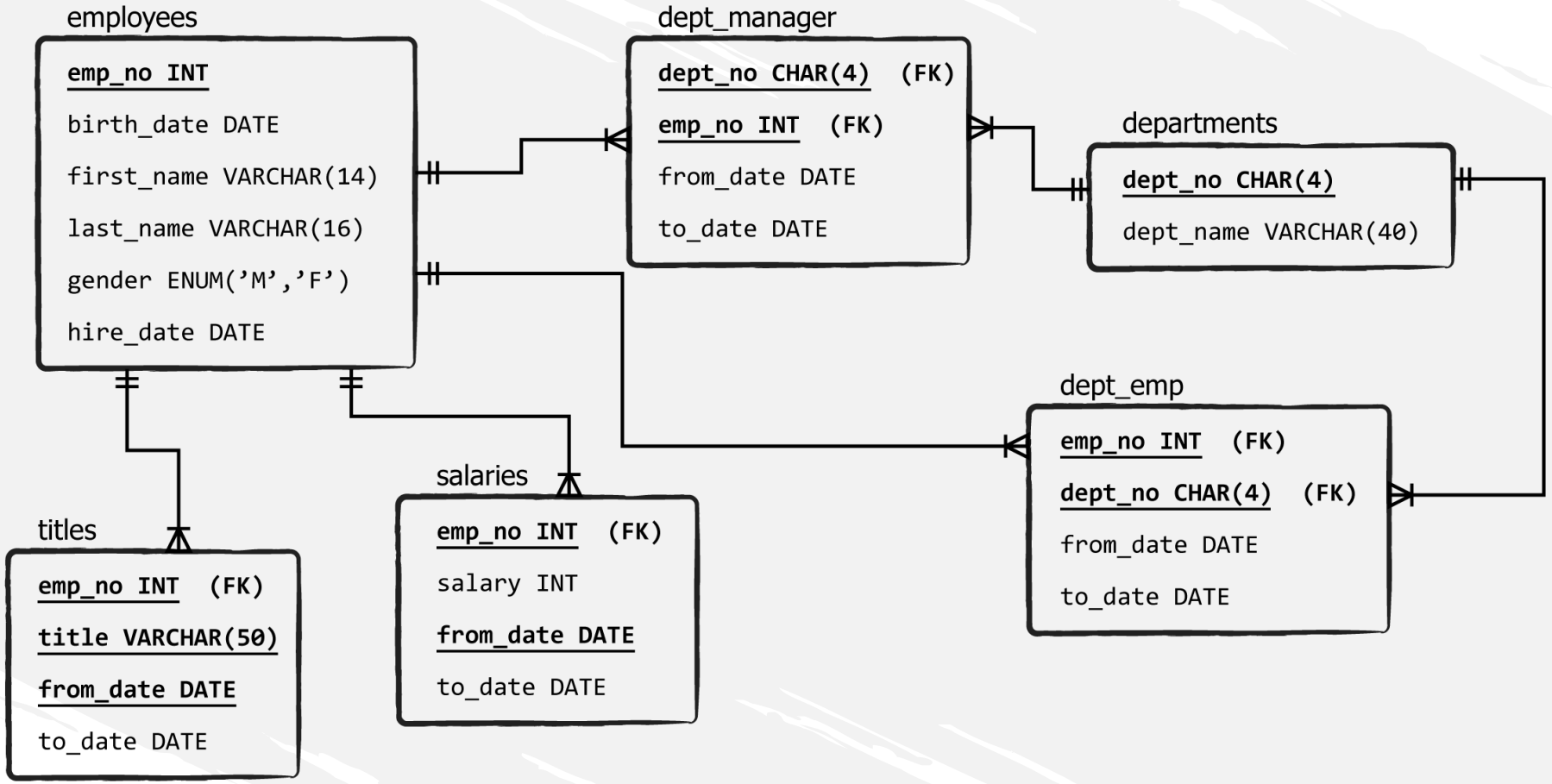
- *primary* and *unique keys* are MySQL indexes

MySQL Indexes

- *primary* and *unique keys* are MySQL indexes

  - they represent columns on which a person would typically base their search

MySQL Indexes



MySQL Indexes

- SQL specialists are always aiming for a good balance between the *improvement of speed search* and the *resources used for its execution*

MySQL Indexes

SQL specialists are always aiming for a good balance between the *improvement of speed search* and the *resources used for its execution*

small datasets	the costs of having an index might be higher than the benefits

MySQL Indexes

SQL specialists are always aiming for a good balance between the *improvement of speed search* and the *resources used for its execution*

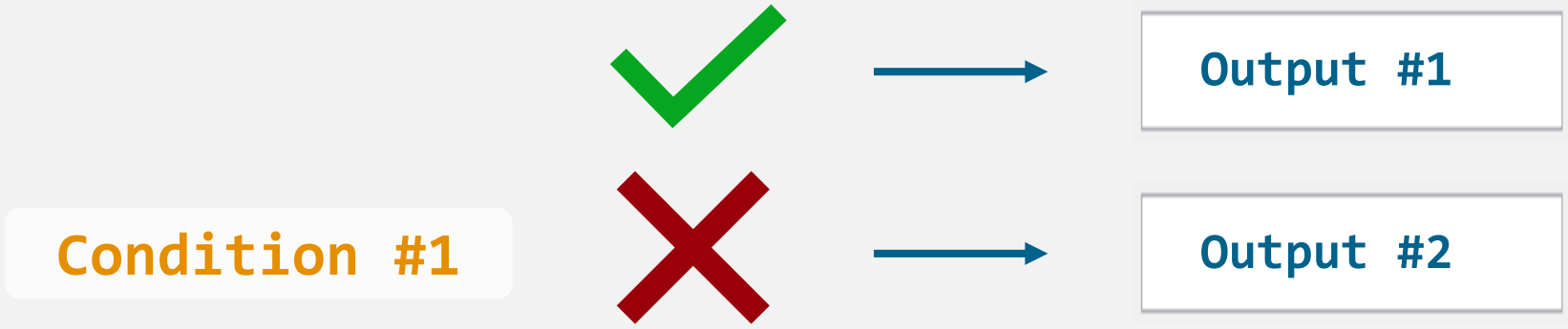
small datasets	the costs of having an index might be higher than the benefits
large datasets	a well-optimized index can make a *positive* impact on the search process

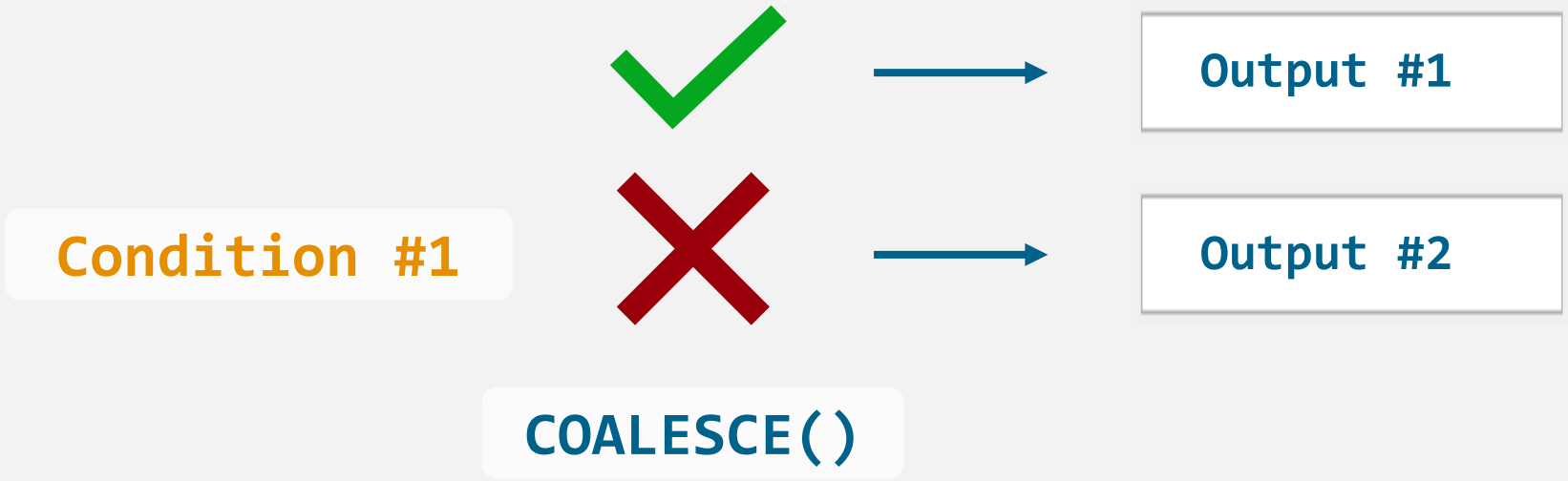
The CASE Statement

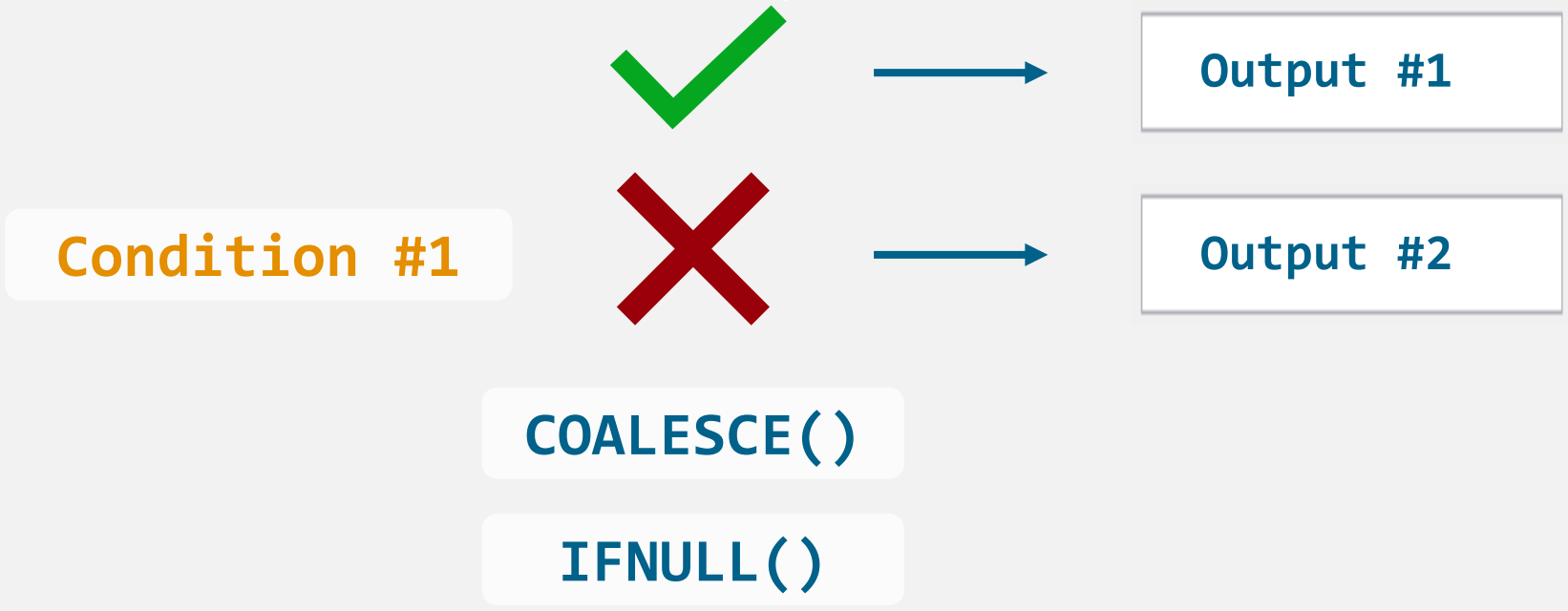
Condition #1

Condition #1

Condition #1

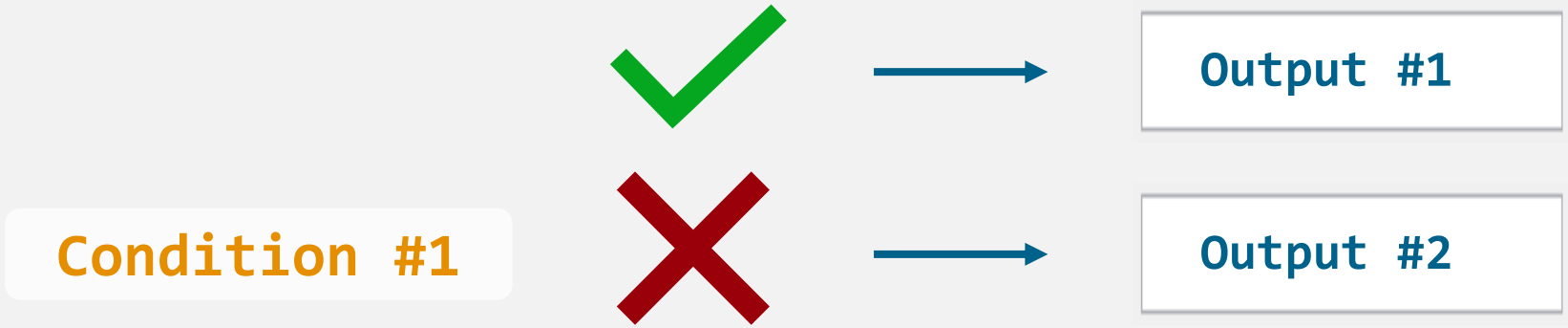






COALESCE()

IFNULL()



COALESCE()

IFNULL()

The CASE Statement

THE CASE STATEMENT

SQL

```
SELECT
     column_name(s)
     CASE
          WHEN condition_1 THEN result_1
          WHEN condition_2 THEN result_2
          …
          ELSE
     END AS
FROM
     table_name;
```

THE CASE STATEMENT

SQL

```
SELECT
     column_name(s)
     CASE condition_field
          WHEN condition_field_value_1 THEN result_1
          WHEN condition_field_value_2 THEN result_2
          …
          ELSE
     END AS
FROM
     table_name;
```