DATA SOCIETY®

Introduction to SQL - Part 2

"One should look for what is and not what he thinks should be."
-Albert Einstein.

Warm up

Before we begin, check out this case study of using MySQL in hospitality industry: ***https://www.mysql.com/why-mysql/case-studies/hospitality-app-boosts-productivity-hotel-profits/***

Welcome back!

In the last class we started looking into MySQL.

Today we will:

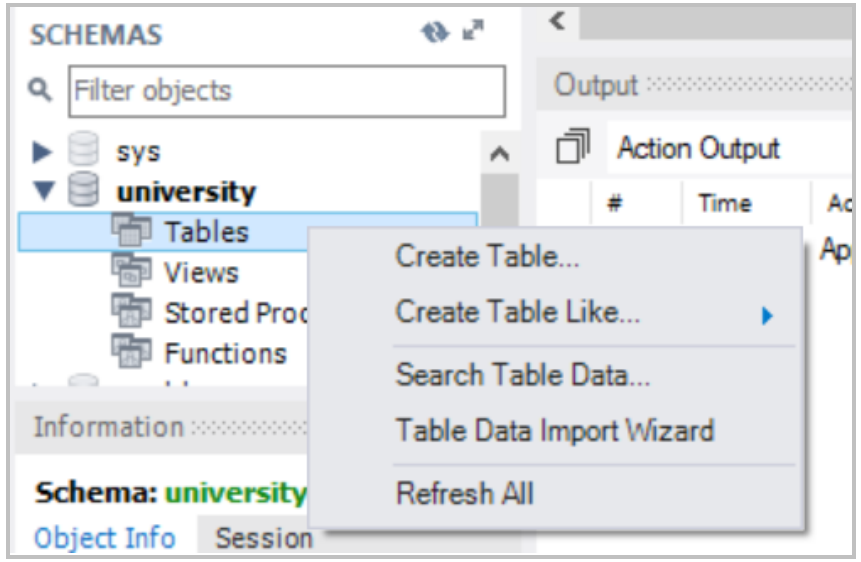
- create tables in SQL
- query datasets and perform data manipulation using SQL commands

Module completion checklist

Objective	Complete
Execute SQL DDL commands	
Query and filter data	

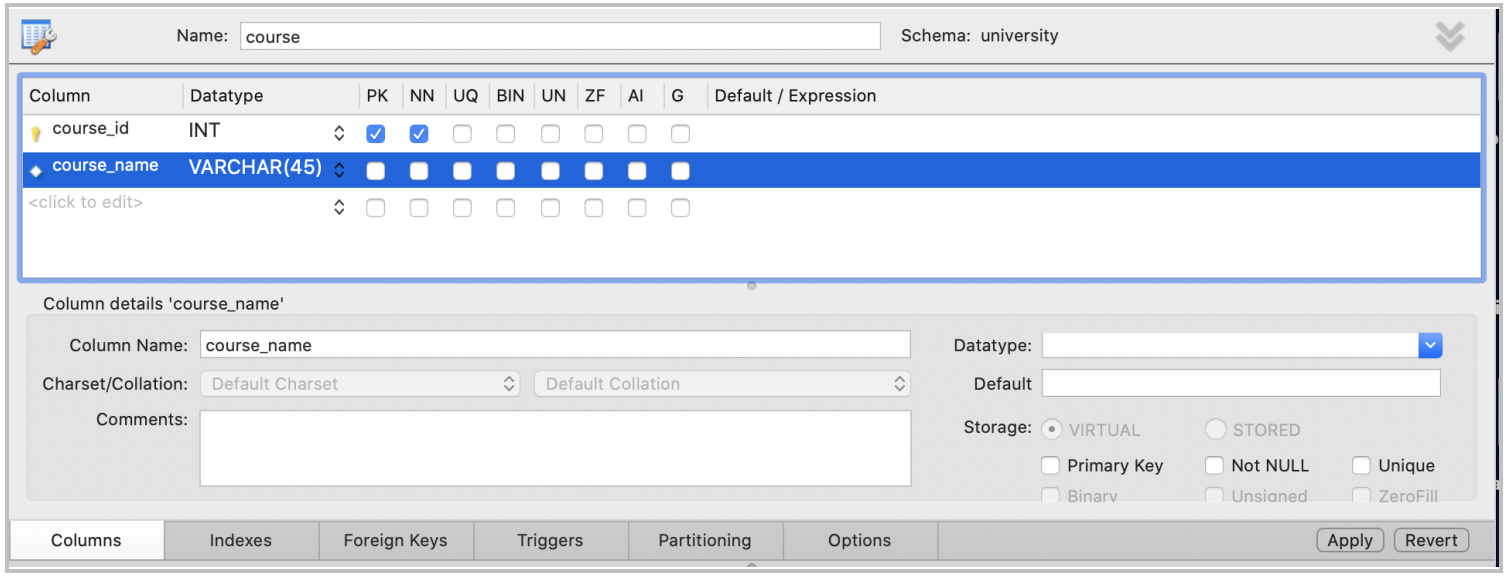
DDL commands

- Recall that SQL's **Data Definition Language (DDL)** is used to define the data structures stored in the database
- We'll now use DDL commands to create, update, and delete tables and data
- Let's start by adding a table to a database



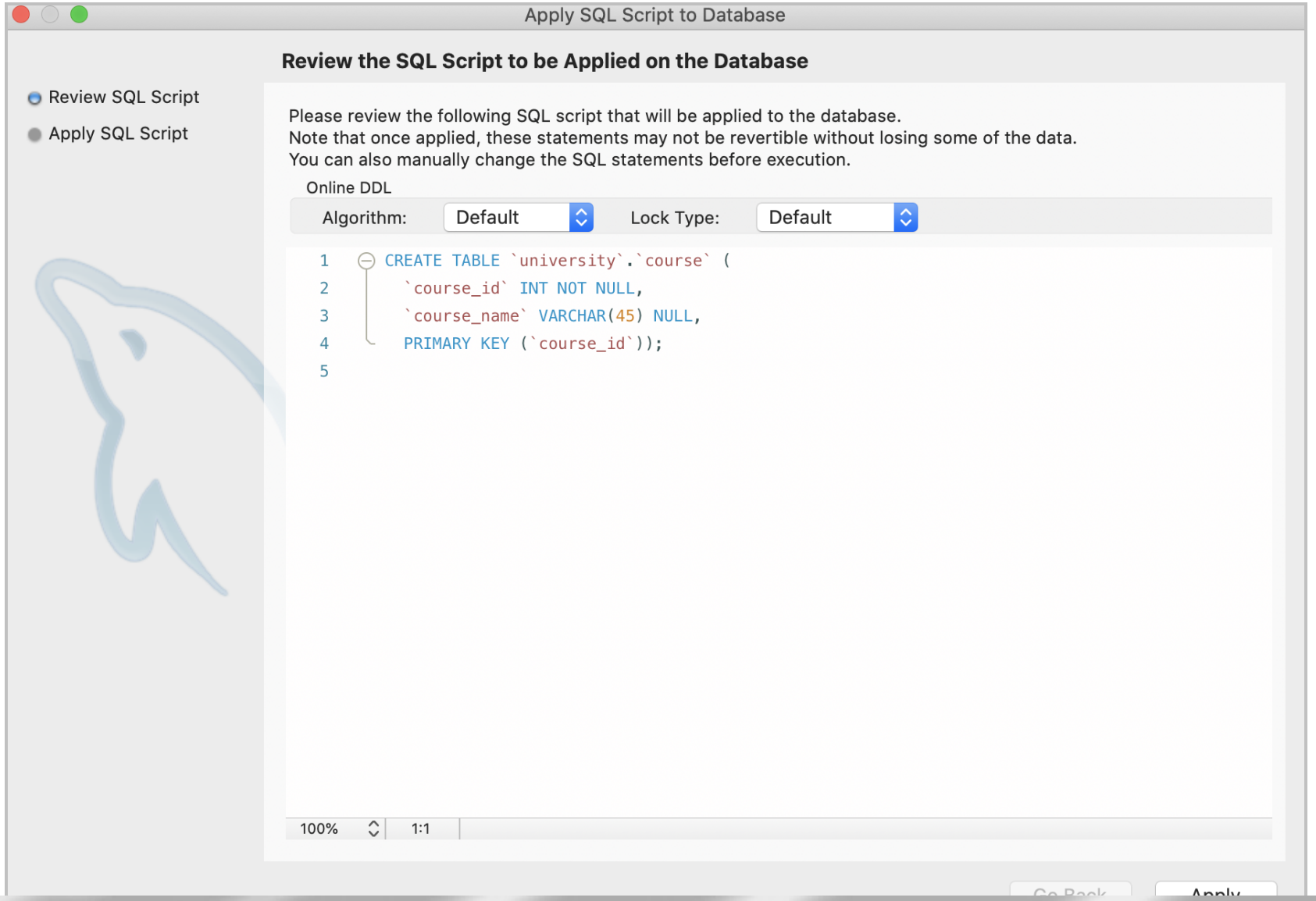
Add table to database

- In the open window, type in `course` for Table Name
- Create `course_id` column of `INT` type and click in `NN` box to set constraint to `NOT NULL` (as in picture below)



- Then, click `PK` for primary key
- After that, create another column `course_name` of type `varchar(45)` and click the **Apply** button

Add table to database



DESC

- To inspect the structure of the table, in the query tab use the **DESC** clause

```
DESC university.course;
```

CREATE TABLE

- We can type the code to create a table in the database using the **CREATE TABLE** clause

Note: The foreign key is defined *only when there is a foreign key in a table*, but primary key is defined for every table created

```
USE university;                                    -- define database to avoid referencing everytime

CREATE TABLE faculty                                  -- table name
(faculty_id INT(10),                                  -- column 1 - faculty id
faculty_name VARCHAR(20),                             -- column 2 - faculty_name
course_id INT(11),                                    -- column 3 - course_id
CONSTRAINT pk_faculty PRIMARY KEY(faculty_id),        -- define primary key
CONSTRAINT fk_faculty_course_id FOREIGN KEY(course_id) -- define foreign key
REFERENCES course(course_id)                          -- referencing table
);

DESC faculty;
```

Result Grid Filter Rows: Export:

#	Field	Type	Null	Key	Default	Extra
1	faculty_id	int	NO	PRI	NULL	
2	faculty_name	varchar(20)	YES		NULL	
3	course_id	int	YES	MUL	NULL	

INSERT

- To insert values into the table, use the **INSERT INTO** clause

```
INSERT INTO course        -- table name
(course_id, course_name) -- column/attributes name
VALUES (011, "DBMS"), (012, "Operating Systems"), (015, "Data Science");    -- values to the attributes
```

SELECT

- To view all the data available in the table, use the **SELECT** clause

```
SELECT * from course; -- * to view all the data in table
```

Result Grid Filter Rows:

#	course_id	course_name
1	11	DBMS
2	12	Operating Systems
3	15	Data Science
*	NULL	NULL

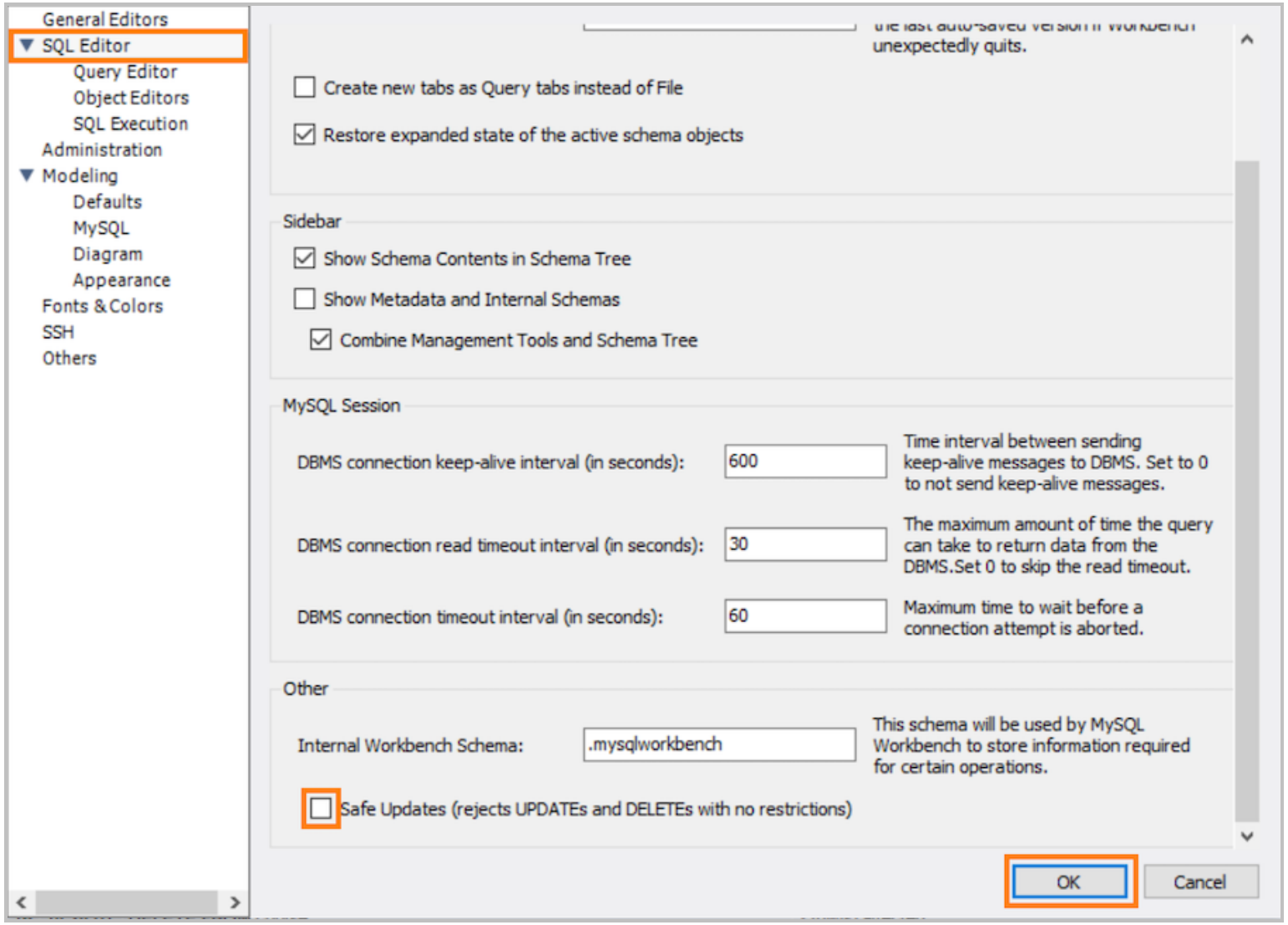
UPDATE

- We can update the table structure and data inserted using the **UPDATE** clause
- Condition for updating can be given using the **WHERE** clause
- Single row can be updated by `'='` operator
- Multiple rows can be updated by `'<', '>', '<=', '>=', '!='` operators
- New values are updated using the **SET** clause

Note: Before we run the update queries in MySQL, let's disable the `safe update` option

- Go to `preferences` -> uncheck the `safe update` option and reconnect your MySQL
- If you have a:
  - `windows/linux:` user `preferences` can be found in `edit` tab
  - `mac`: user `preferences` can be found in `MySQLWorkbench` tab on the top

UPDATE


General Editors
SQL Editor
Query Editor
Object Editors
SQL Execution
Administration
Modeling
Defaults
MySQL
Diagram
Appearance
Fonts & Colors
SSH
Others
unexpectedly quits.
Create new tabs as Query tabs instead of File
Restore expanded state of the active schema objects
Sidebar
Show Schema Contents in Schema Tree
Show Metadata and Internal Schemas
Combine Management Tools and Schema Tree
MySQL Session
DBMS connection keep-alive interval (in seconds):
600
Time interval between sending keep-alive messages to DBMS. Set to 0 to not send keep-alive messages.
DBMS connection read timeout interval (in seconds):
30
The maximum amount of time the query can take to return data from the DBMS.Set 0 to skip the read timeout.
DBMS connection timeout interval (in seconds):
60
Maximum time to wait before a connection attempt is aborted.
Other
Internal Workbench Schema:
.mysqlworkbench
This schema will be used by MySQL Workbench to store information required for certain operations.
Safe Updates (rejects UPDATEs and DELETEs with no restrictions)
OK
Cancel

UPDATE

```
UPDATE course                                        -- update the table
SET course_name = "Database Management Systems"      -- set new value
WHERE course_id = 011;                               -- condition to set the data
```

Before updating

Result Grid Filter Rows:

#	course_id	course_name
1	11	DBMS
2	12	Operating Systems
3	15	Data Science
*	NULL	NULL

After updating

Result Grid Filter Rows:

#	course_id	course_name
1	11	Database Management Systems
2	12	Operating Systems
3	15	Data Science
*	NULL	NULL

ALTER TABLE

- We can change the table structure using the **ALTER TABLE** clause
- **ALTER TABLE** can be used to
  - **Add** a new column
  - **Modify** the existing column
  - **Drop** the existing column

ALTER TABLE - ADD

- To add a new column, use the **ADD** clause and define the column name & type

```
ALTER TABLE faculty   -- table name
ADD dateofBirth DATE; -- add new column 'dateofBirth'
```

```
DESC faculty;
```

#	Field	Type	Null	Key	Default	Extra
1	faculty_id	int	NO	PRI	NULL	
2	faculty_name	varchar(20)	YES		NULL	
3	course_id	int	YES	MUL	NULL	
4	dateofBirth	date	YES		NULL	

ALTER TABLE - MODIFY & DROP

- To modify the column structure, use the **MODIFY** clause

```
ALTER TABLE faculty       -- table name
MODIFY dateofBirth YEAR;  -- alter 'dateofBirth'
                          --  date to year type
```

```
DESC faculty;
```

Result Grid Filter Rows: Export:

#	Field	Type	Null	Key	Default	Extra
1	faculty_id	int	NO	PRI	NULL	
2	faculty_name	varchar(20)	YES		NULL	
3	course_id	int	YES	MUL	NULL	
4	dateofBirth	year	YES		NULL	

- To delete a column, use the **DROP** clause

```
ALTER TABLE faculty       -- table name
DROP COLUMN dateofBirth;  -- delete column
                          --   'dateofBirth'
```

```
DESC faculty;
```

Result Grid Filter Rows: Export:

#	Field	Type	Null	Key	Default	Extra
1	faculty_id	int	NO	PRI	NULL	
2	faculty_name	varchar(20)	YES		NULL	
3	course_id	int	YES	MUL	NULL	

DELETE - data

- We can delete the data from the table by using the **DELETE FROM** clause
- Use the **WHERE** clause to set the condition to delete the data
- Single row can be deleted by `'='` operator
- Multiple rows can be deleted by `'<'`, `'>'`, `'<='`, `'>='`, `'!='` operators

- **Before deleting**

```
SELECT * from course;
```

Result Grid Filter Rows:

#	course_id	course_name
1	11	DBMS
2	12	Operating Systems
3	15	Data Science
*	NULL	NULL

- **After deleting**

```
DELETE FROM course WHERE course_id = 12;
```

Result Grid Filter Rows:

#	course_id	course_name
1	11	Database Management Systems
2	15	Data Science
*	NULL	NULL

DELETE - table & column

- To delete the entire data from the table

```
DELETE FROM course;
```

- To remove the table itself from the database, use the **DROP TABLE**

```
DROP TABLE faculty;  -- child table having course_id as foreign key
DROP TABLE course;   -- parent table
```

- **Note: always delete the child table with the foreign key first**
- Table `faculty` has a foreign key from `course` table, so delete `faculty` table first
- `course` table is the referenced table - if we delete course table first, it will throw an error
- Delete faculty table first and then course table

43	01:12:16	drop table course	Error Code: 3730. Cannot drop table 'course' referenced by a foreign key constraint '...	0.016 sec
44	01:12:30	drop table faculty	0 row(s) affected	0.031 sec
45	01:12:37	drop table course	0 row(s) affected	0.031 sec

When good SQL statements go bad

- The following common errors should be avoided:
  - Non-unique **primary key**
  - Non-existent **foreign key**
  - Column value violation
  - Invalid **date** conversion

Knowledge check 1

Exercise 1

Module completion checklist

Objective	Complete
Execute SQL DDL commands	✔
Query and filter data	

Our starting scenario

- Your manager would like a database of information about the world from which he can pull information for your organization's annual report
- To meet this objective, we will be creating and querying a database called `world`
- The database will be built using the `world.sql` file, which came from the MYSQL website and contains information about countries, cities, and languages

'World' database schema

country table

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	Field	Type	Null	Key	Default	Extra
1	Code	char(3)	NO	PRI		
2	Name	char(52)	NO			
3	Continent	enum('Asia','Europe','North Americ...	NO		Asia	
4	Region	char(26)	NO			
5	SurfaceArea	decimal(10,2)	NO		0.00	
6	IndepYear	smallint	YES		NULL	
7	Population	int	NO		0	
8	LifeExpectancy	decimal(3,1)	YES		NULL	
9	GNP	decimal(10,2)	YES		NULL	
10	GNPOld	decimal(10,2)	YES		NULL	
11	LocalName	char(45)	NO			
12	Government...	char(45)	NO			
13	HeadOfState	char(60)	YES		NULL	
14	Capital	int	YES		NULL	
15	Code2	char(2)	NO			

city table

Result Grid | Filter Rows: | Export: | Wrap

#	Field	Type	Null	Key	Default	Extra
1	ID	int	NO	PRI	NULL	auto_increment
2	Name	char(35)	NO			
3	CountryCode	char(3)	NO	MUL		
4	District	char(20)	NO			
5	Population	int	NO		0	

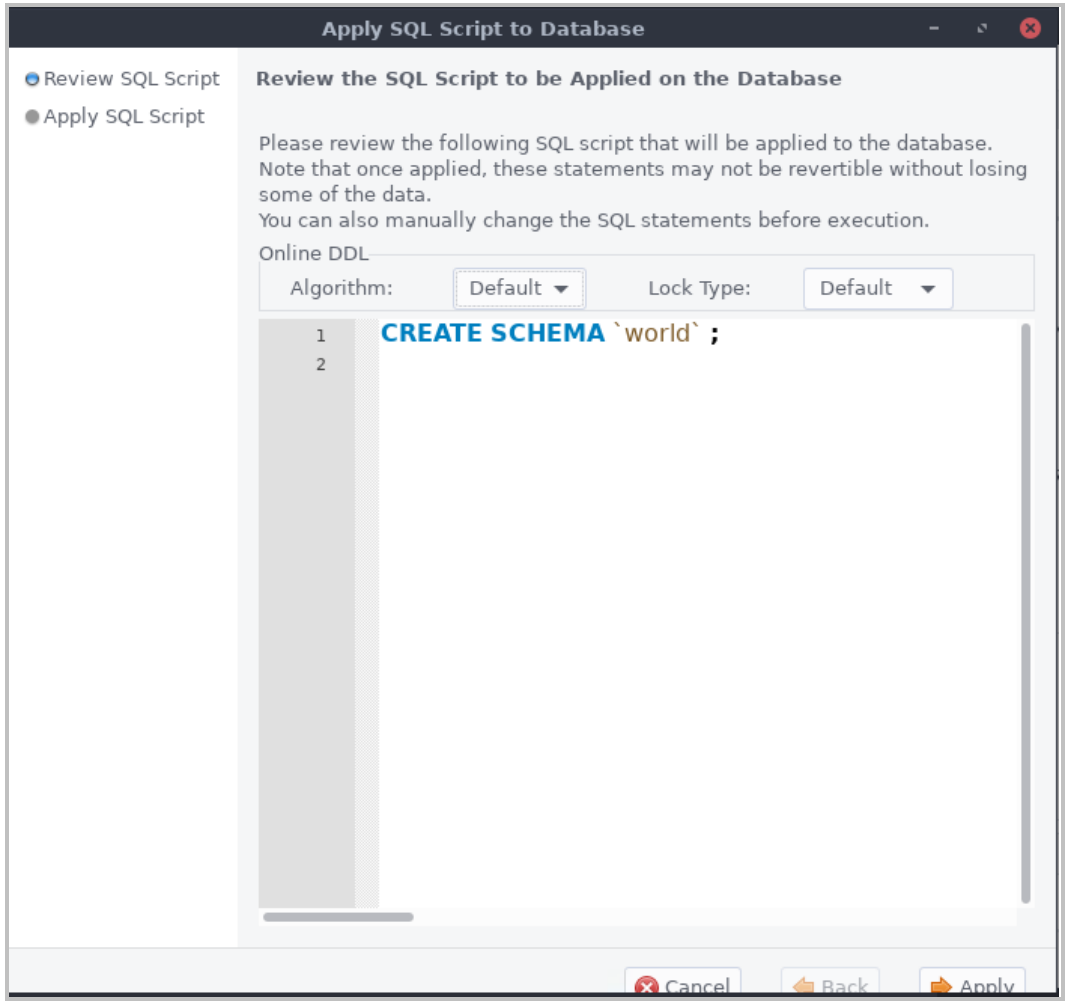
country language table

Result Grid | Filter Rows: | Export:

#	Field	Type	Null	Key	Default	Extra
1	CountryCode	char(3)	NO	PRI		
2	Language	char(30)	NO	PRI		
3	IsOfficial	enum('T','F')	NO		F	
4	Percentage	decimal(4,1)	NO		0.0	

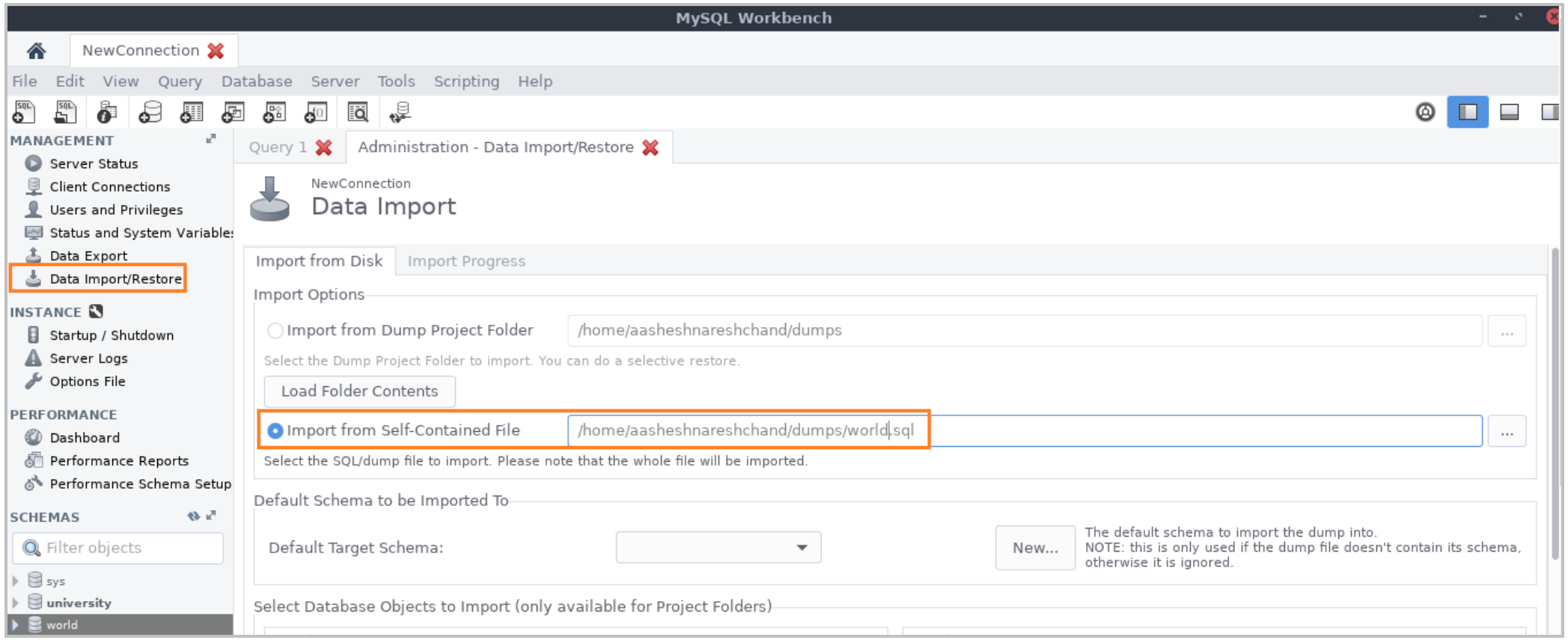
Create new database

- Create a new database named **world**



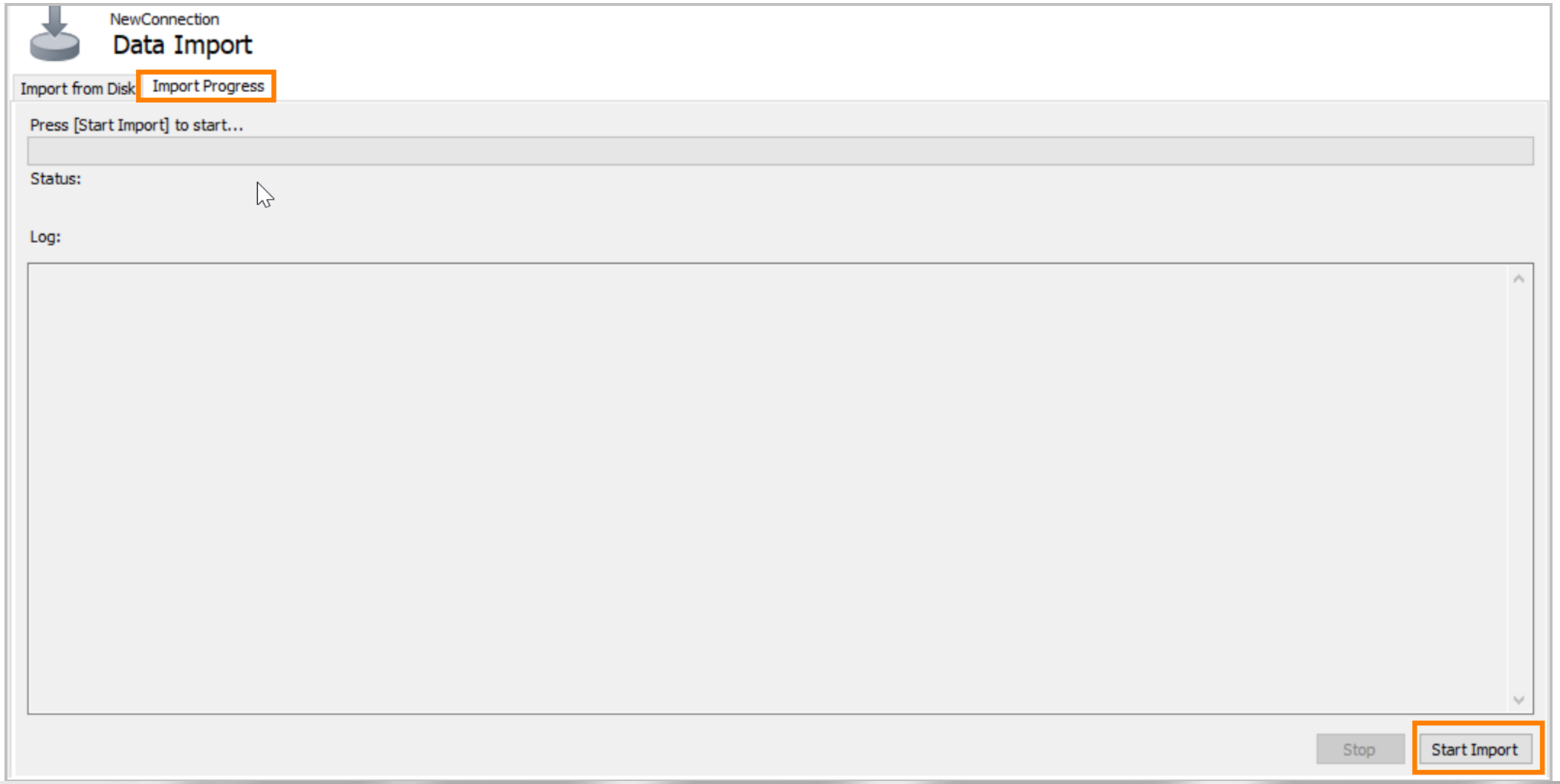
Import data

- We are going to import the schema and data from the .sql file
- Click on the **Data import/Restore** and navigate to the data folder to select the `world.sql` file



Import data (cont'd)

- Click on **import progress** and **start import**



Query clauses

- Now that we have our database, let's explore how to manipulate and query the data inside
- In SQL, there are six clauses used for this purpose:

Clause	Definition
SELECT	Determines which columns to include in the query's result set
FROM	Identifies the table from which to draw the data and how the tables should be joined
WHERE	Filters out unwanted data
GROUP BY	Used to group rows by common column values
HAVING	Filters out unwanted groups
ORDER BY	Orders the rows ascending or descending of the final result set by one or more columns

Order of the clause matters

- Always remember that the order of the clauses matters!

1 SELECT	2 FROM	3 WHERE
4 GROUP BY	5 HAVING	6 ORDER BY

SELECT & FROM

- **SELECT** can be used to select all columns in a table

```
-- Switch to world database for convenience
USE world;

-- To select all data, use * operator
SELECT * FROM city;
```

Result Grid Filter Rows: Edit:

#	ID	Name	CountryCode	District	Population
1	1	Kabul	AFG	Kabol	1780000
2	2	Qandahar	AFG	Qandahar	237500
3	3	Herat	AFG	Herat	186800
4	4	Mazar-e-Sharif	AFG	Balkh	127800
5	5	Amsterdam	NLD	Noord-Holland	731200
6	6	Rotterdam	NLD	Zuid-Holland	593321

- **SELECT** can be used to select some columns in a table

```
-- To select specific columns.
SELECT Name, Population FROM city;
```

Result Grid Filter Rows:

#	Name	Population
1	Kabul	1780000
2	Qandahar	237500
3	Herat	186800
4	Mazar-e-Sharif	127800
5	Amsterdam	731200
6	Rotterdam	593321

SELECT & FROM

- Other operations that can be done on the final result set include:
  - adding a new column with same data across all rows
  - performing mathematical operations on numeric columns
  - performing string operations on character columns

```
SELECT ID,                  -- select ID
'City_population',          -- add a new column with value City_population
Population/100,             -- divide the population column by 100
LOWER(CountryCode)          -- convert the country code column to lower case
FROM city;
```

Result Grid Filter Rows: Export: Wrap

#	ID	City_populatio	Population/100	LOWER(CountryCode
1	1	City_population	17800.0000	afg
2	2	City_population	2375.0000	afg
3	3	City_population	1868.0000	afg
4	4	City_population	1278.0000	afg
5	5	City_population	7312.0000	nld
6	6	City_population	5933.2100	nld

DISTINCT

- It may happen that a query returns duplicate rows of data
- To return *only* distinct rows, we use the **DISTINCT** clause

```
SELECT Language FROM countrylanguage;

-- Returns 984 rows
```

Result Grid | Filter Ro

#	Language
1	Dutch
2	English
3	Papiamento
4	Spanish
5	Balochi
6	Dari

```
SELECT DISTINCT Language FROM countrylanguage;

-- Returns 457 rows
```

Result Grid | Filter Ro

#	Language
1	Dutch
2	English
3	Papiamento
4	Spanish
5	Balochi
6	Dari

✓	12	14:59:02	SELECT Language FROM countrylanguage LIMIT 0, 1000	984 row(s) returned
✓	13	15:00:27	SELECT DISTINCT Language FROM countrylanguage LIMIT ...	457 row(s) returned

AS

- The **AS** clause is used for aliasing table names and column names
- A **column alias** gives a new name to the existing column in the final result set
- A **table alias** gives a new name to the existing table

```
SELECT ci.Name AS city_name,    -- alias column name as city_name
ci.District AS city_district    -- alias column district as city_district
FROM city AS ci;                -- alias table city as ci
```

Result Grid Filter Rows:

#	city_name	city_district
1	Kabul	Kabol
2	Qandahar	Qandahar
3	Herat	Herat
4	Mazar-e-Sharif	Balkh
5	Amsterdam	Noord-Holland
6	Rotterdam	Zuid-Holland
7	Haag	Zuid-Holland

WHERE

- MySQL allows you to use the **WHERE** clause to add filter conditions, including:

Condition type	Syntax
Logical condition	`OR, AND, NOT`
Equality or matching condition	`=`
Comparison condition	`<, <=, >, >=, <>, !=`
Range condition	BETWEEN & AND
Membership condition	IN
Null condition	IS NULL; IS NOT NULL;

WHERE

```
-- Select all data from 'countrylanguage' table where the language is official to the country and
-- the percentage spoken is greater than 70%.
SELECT * FROM countrylanguage -- table name
  WHERE                       -- WHERE clause to filter
  IsOfficial = 'T'            -- equality condition '='
  AND                         -- logical condition 'AND'
  Percentage > 70;            -- comparison condition '>'
```

Result Grid Filter Rows: Edit:

#	CountryCode	Language	IsOfficial	Percentage
1	ALB	Albaniana	T	97.9
2	ANT	Papiamento	T	86.2
3	ARG	Spanish	T	96.8
4	ARM	Armenian	T	93.4
5	ASM	Samoan	T	90.6
6	AUS	English	T	81.2
7	AUT	German	T	92.0
8	AZE	Azerbaijani	T	89.0

WHERE

```sql
-- Select name and population of the cities
where population is between 180000 and 190000.

SELECT Name, Population FROM city
WHERE population BETWEEN 180000 AND 190000;
```

```sql
-- Select all the data from countrylanguage of
three countries USA, Australia, and India.

SELECT * FROM countrylanguage
WHERE CountryCode IN ('USA', 'AUS', 'IND');
```

Result Grid Filter Rows:

#	Name	Population
1	Herat	186800
2	Batna	183377
3	Santiago del…	189947
4	Mymensingh	188713
5	Liège	185639
6	São Leopoldo	189258
7	Marília	188691
8	São Carlos	187122

Result Grid Filter Rows: Edit:

#	CountryCode	Language	IsOfficia	Percentage
1	AUS	Arabic	F	1.0
2	AUS	Canton Chinese	F	1.1
3	AUS	English	T	81.2
4	AUS	German	F	0.6
5	AUS	Greek	F	1.6
6	AUS	Italian	F	2.2
7	AUS	Serbo-Croatian	F	0.6
8	AUS	Vietnamese	F	0.8

ORDER BY

- Use **ORDER BY** to arrange data in ascending order (default)

```
-- Select all data from the 'city' table order
by population in ascending order.

SELECT * FROM city
  ORDER BY Population;
```

Result Grid Filter Rows: Edit:

#	ID	Name	CountryCode	District	Population
1	2912	Adamstown	PCN	–	42
2	2317	West Island	CCK	West Island	167
3	3333	Fakaofo	TKL	Fakaofo	300
4	3538	Città del Vati...	VAT	–	455
5	2316	Bantam	CCK	Home Island	503
6	2728	Yaren	NRU	–	559
7	62	The Valley	AIA	–	595
8	2805	Alofi	NIU	–	682

- Use **ORDER BY** to arrange data in descending order

```
-- Select all data from the 'city' table order
by population in descending order.

SELECT * FROM city
  ORDER BY Population
  DESC;
```

Result Grid Filter Rows: Edit:

#	ID	Name	CountryCode	District	Population
1	1024	Mumbai (Bo...	IND	Maharashtra	10500000
2	2331	Seoul	KOR	Seoul	9981619
3	206	São Paulo	BRA	São Paulo	9968485
4	1890	Shanghai	CHN	Shanghai	9696300
5	939	Jakarta	IDN	Jakarta Raya	9604900
6	2822	Karachi	PAK	Sindh	9269265
7	3357	Istanbul	TUR	Istanbul	8787958
8	2515	Ciudad de M...	MEX	Distrito Federal	8591309

A note on NULL

- **NULL** values are appropriate:
  - when values are not available or applicable
  - when values are not yet known, but will be added later
  - when values are undefined
- To test whether an expression is null, use **IS NULL** or **IS NOT NULL** operators

NULL

```sql
-- Select Name and IndepYear from country
-- where IndepYear is null.
SELECT Name, IndepYear FROM country
WHERE IndepYear IS NULL;
```

Name	IndepYear
Aruba	NULL
Anguilla	NULL
Netherlands Antilles	NULL
American Samoa	NULL
Antarctica	NULL
French Southern territories	NULL
Bermuda	NULL
Bouvet Island	NULL
Cocos (Keeling) Islands	NULL

```sql
-- Select Name and IndepYear from country
-- where IndepYear is  not null.
SELECT Name, IndepYear FROM country
WHERE IndepYear IS NOT NULL;
```

Name	IndepYear
Afghanistan	1919
Angola	1975
Albania	1912
Andorra	1278
United Arab Emirates	1971
Argentina	1816
Armenia	1991
Antigua and Barbuda	1981
Australia	1901

Knowledge check 2

Exercise 2

Module completion checklist

Objective	Complete
Execute SQL DDL commands	✔
Query and filter data	✔

Next steps

In the next session you will:

- query datasets and perform data manipulation using SQL commands
- learn to implement join and set operations

This completes our module

Congratulations!