Relational DBs & the SQL language

{`auth`:[`ap`, `ss`, `da`, `am`]}

This unit:

- ✓Introduction to structured vs. unstructured data, relational databases.

1. Introduction to Database Management systems

- Some theory on databases

2. Introduction to SQL and SQL by Python

```
SELECT Pizza
FROM Freezer
WHERE Pepperoni = 10
AND Olives = 'Black'
```

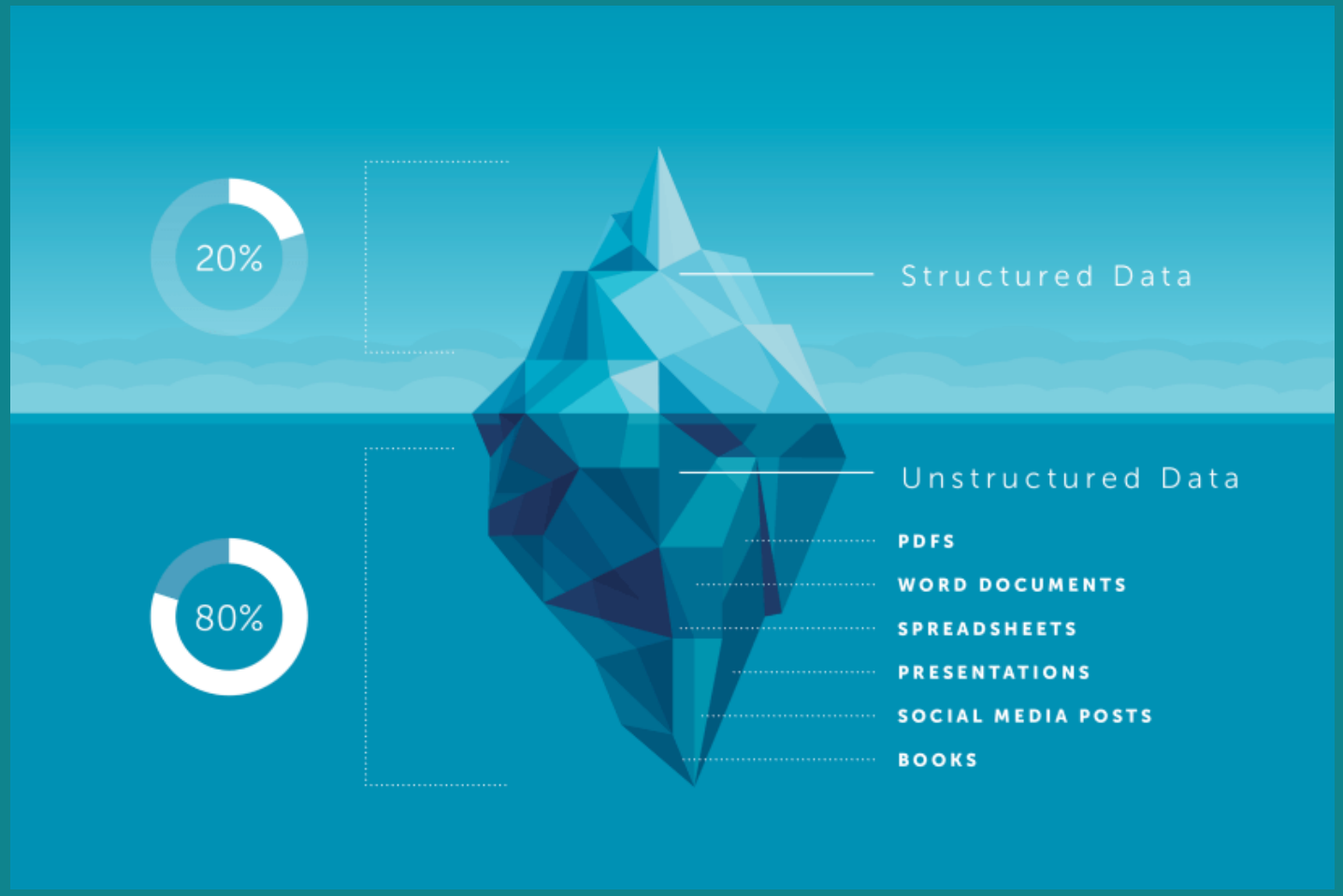
Data organisation

- Structured data
  - Clearly defined data types
  - A data model (called schema) defines the data
    - Data type (e.g. integer), data format (e.g. +44 (079) 444 4444
  - Data model defines rules and constraints!
    - E.g. salary should be >0, first name cannot be empty etc.

- Unstructured data
  - Internal structure is not pre defined
  - There is no data model

Structured vs Unstructured data

- Structured data resides in relational databases:
  - a database structured to recognise relations between stored items of data
- Unstructured data is everything else!
  - Text files, emails, websites, media, social media etc.



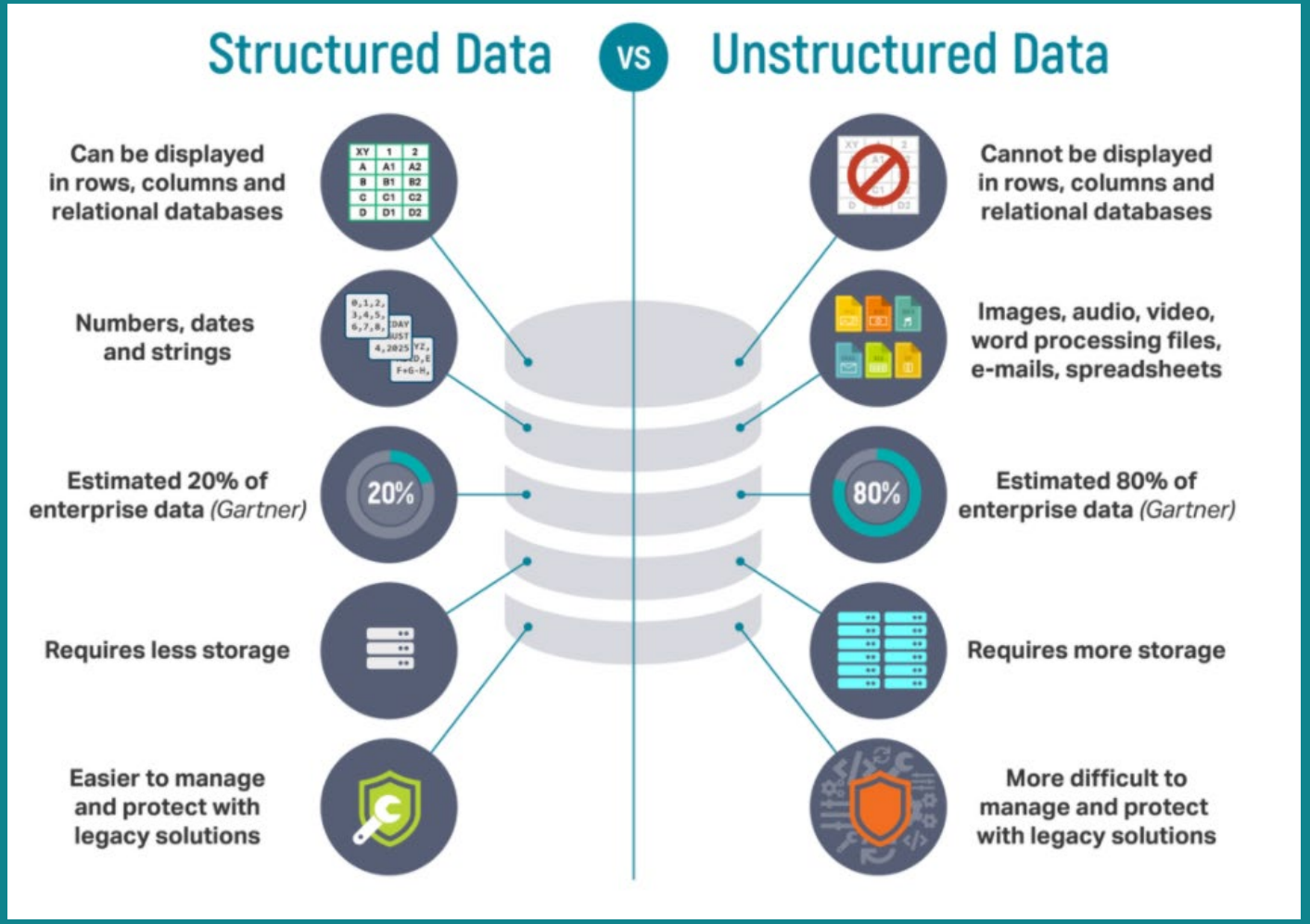
In detail … Structured data

- Clearly defined data types
- Designed in a way to be easy to search data
- Usually stored in relational databases
  - Relational database management systems (RDBMS)
- Fields (column titles) easy to manage using constraints
- To interact with RDBMS we use the *Structured Query Language* (SQL)
  - Extract and manipulate data based on queries
  - the top programming language by mentions in job adverts for technical positions [IEEE Spectrum, Aug. 2023]

In detail … Unstructured data

- Have some internal structure
- There is no clear data model
- It could be difficult to search
- Unstructured data stored in NoSQL database systems.
- Examples:
  - Text files
  - Emails

Structured vs Unstructured



Semi-structured data

- **Semi-structured data** is a form of **structured data** that does not obey the formal **structure** of **data** models associated with relational databases or other forms of **data** tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the **data**. [Wikipedia]
- Examples:
  - JSON
  - XML
- NoSQL systems: Ideally systems to store Semi-structured data

Example of XML file

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Source: https://en.wikipedia.org/wiki/XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
<food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
   Two of our famous Belgian Waffles with plenty of real maple syrup
   </description>
    <calories>650</calories>
</food>
<food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>
    Light Belgian waffles covered with strawberries and whipped cream
    </description>
    <calories>900</calories>
</food>
<food>
    <name>Berry-Berry Belgian Waffles</name>
    <price>$8.95</price>
    <description>
    Belgian waffles covered with assorted fresh berries and whipped cream
    </description>
    <calories>900</calories>
</food>
<food>
    <name>French Toast</name>
    <price>$4.50</price>
    <description>
    Thick slices made from our homemade sourdough bread
    </description>
    <calories>600</calories>
</food>
<food>
    <name>Homestyle Breakfast</name>
    <price>$6.95</price>
    <description>
    Two eggs, bacon or sausage, toast, and our ever-popular hash browns
    </description>
    <calories>950</calories>
</food>
</breakfast_menu>
```

Example of JSON file

```
{
    "fruit": "Apple",
    "size": "Large",
    "color": "Red"
}
```

In computing, JavaScript Object Notation (JSON) (/ˈdʒeɪsən/ "Jason"[1][2]) is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value).
Source: https://en.wikipedia.org/wiki/JSON

Summary

- Structured data:
  - Examples:
    - Airline reservation system
    - Inventory control
- Semi-structured data:
  - Examples:
    - Data generated from systems
    - Data collected from a pulse monitoring device
- Unstructured data:
  - Examples:
    - Email clients
    - Word processing documents
    - Logs and reports

Relational databases: a logical view

- data is structured in tables
- intuitive organisation
- easier to control/search
- more restrictive and less ad hoc/ambigous than arbitrary spreadsheet tables
- *table* is defined as a 2-D data structure with rows (records) and columns (attributes/labels)
- the attributes row (aka *schema*) is special: names must be unique & have fixed data type
- a *relational database* is a set of such tables

				Attribute
	User_id	**Fname**	**Lname**	**Phone**
	1001	Stelios	Sotiriadis	07912341234
record	1002	Niki	Andrea	07976543421
	1003	George	Martin	07912341234

Table characteristics

- A table includes attributes:
  - **Keys**
    - Primary Key (PK): An attribute that uniquely identifies any given entity (row of record)
    - Keys are used in all systems
      - Passport ID
      - NINO
      - NHS Number
  - Key characteristics:
    - Cannot be empty for a record (Not Null)
    - Has to be unique!
      - No duplicated values are allowed.
    - It’s very useful when you create relationships between tables
      - Helps to avoid redundancy (records that are repeated)

What is a schema (data model) ?

- A data model (or datamodel) is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities.
  - [Source]

Example: there is a canonical, almost immutable Relational *schema* to describe a university.
Typical tables are Students, Modules/Classes, Completed exams, Graduates, Instructors

There are dozens of relational database instances, sometimes hosted in the same server/cloud:
You are into the unimi.it instance.

What's wrong with this table?

Book_ID	Book_Title	Book_Author	Author_Address	Authors_Phone_Number
1	Hobbit	Tolkien	Bournemouth	01202 345234
2	LOTR 1	Tolkien	Bournemouth	01202 345234
3	LOTR 2	Tolkien	Bournemouth	01202 345234
4	LOTR 3	Tolkien	Bournemouth	01202 345234
5	The Silmarillion	Tolkien	Bournemouth	01202 345234

5 rows * 5 columns = 25 data

Quiz 1: Can we redesign it?

- Produce a better design.
  - Define ”better”…

Book_ID	Book_Title	Book_Author	Author_Address	Authors_Phone_Number
1	Hobbit	Tolkien	Bournemouth	01202 345234
2	LOTR 1	Tolkien	Bournemouth	01202 345234
3	LOTR 2	Tolkien	Bournemouth	01202 345234
4	LOTR 3	Tolkien	Bournemouth	01202 345234
5	The silmarillion	Tolkien	Bournemouth	01202 345234

Designing a “better” data model

These are the same…

Book_ID	Book_Title	Author_ID
1	Hobbit	A1
2	LOTR 1	A1
3	LOTR 2	A1
4	LOTR 3	A1
5	The silmarillion	A1

5 rows * 3 columns = 15 data

1 row * 4 columns = 4 data

19 data

(*less data…*)

Author_ID	Book_Author	Author_Address	Authors_Phone_Number
A1	Tolkien	Bournemouth	01202 345234

Database creation

- A database is an organized collection of structured information, or data, typically stored electronically in a computer system.
- A database is usually controlled by a (heavy) software called database management system (DBMS).
- Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

[Source]

Popular RDBMSs

- IBM DB2
- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- SQLite: a Python port
- MongoDB: not a SQL database
- Check the updated rankings

What is SQL

- SQL (pronounced "ess-que-el") stands for Structured Query Language.
- SQL is used to communicate with a database.
- SQL provides a set of commands to interact with a database system.

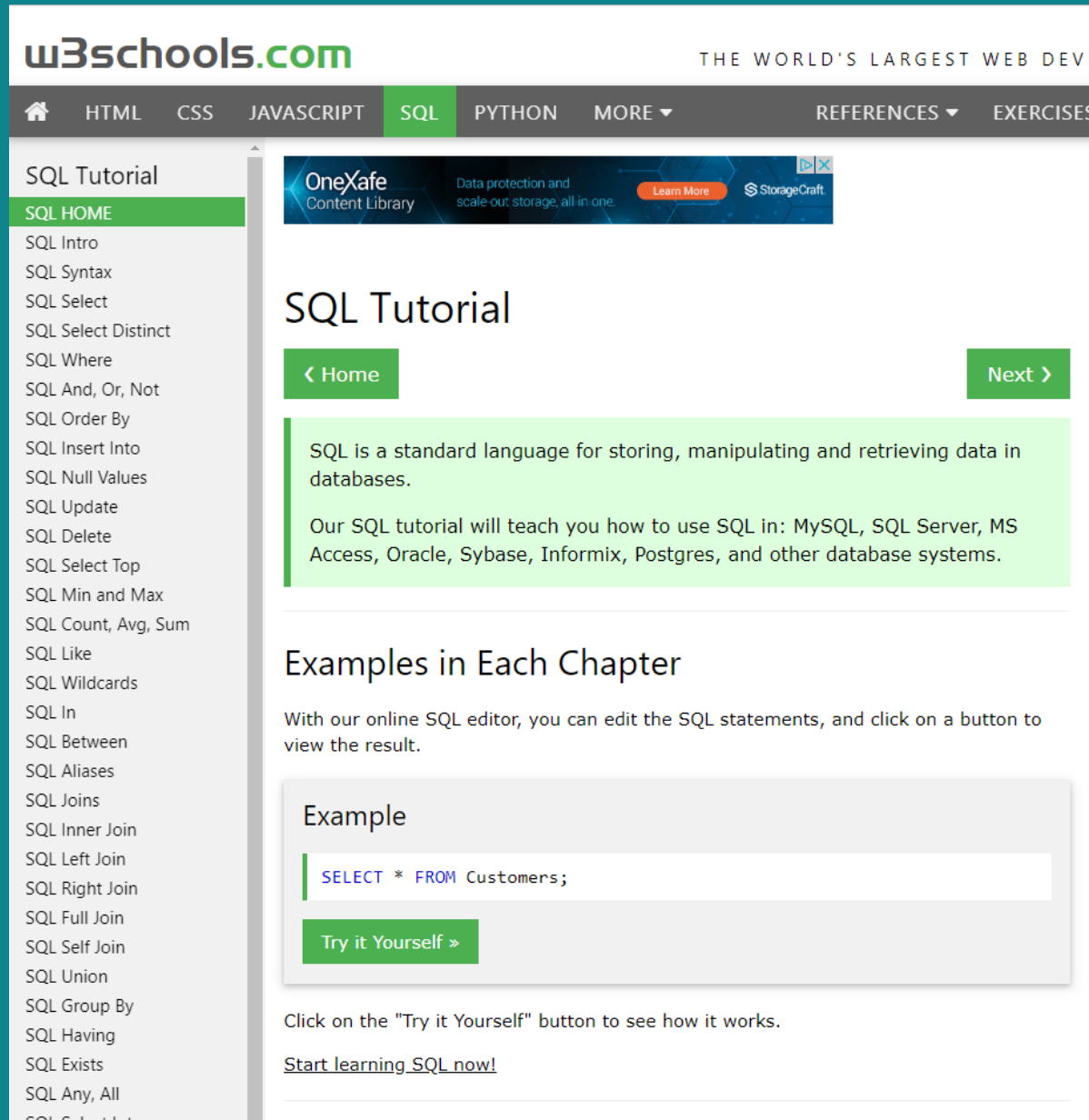
Create a database

Create a database

```
CREATE DATABASE bl_data;
```

Delete a database

```
DROP DATABASE bl_data;
```



Create a new table

- We want to store data about urban areas
  - **city_id** will be numeric (Integer) and my primary key
    - No duplicates/Not Null values
  - **name** will be string (characters), in SQL called varchar!
    - Also, the size of the name could be up to 24 characters…
    - name, cannot be empty in my table!
  - **country** will be a varchar, up to 2 characters
    - E.g. UK, US, GR, NL etc.

Table Name: Cities

city_id	name	country
1	London	GB
2	New York City	US

Test it on w3schools!

```
SQL> CREATE TABLE cities (city_id INTEGER PRIMARY KEY,
                          name VARCHAR(24) NOT NULL,
                          country VARCHAR(2)
                          );
```

SQL Datatypes

- VARCHAR(size):
  - A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535
- TEXT
  - A field with a maximum length of 65535 characters
- INTEGER(M):
  - M is optional, and denotes the display length.
  - A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295.
- FLOAT(M,D)
  - A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D).

SQL Describe a table to see internal structure

```
DESCRIBE cities;
```

```
mysql> DESCRIBE cities;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| city_id | int(11)     | NO   | PRI | NULL    |       |
| name    | varchar(24) | NO   |     | NULL    |       |
| country | varchar(2)  | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

SQL Insert data

- RDBMSs allow slightly different INSERT commands
- MySQL use the following:

```
INSERT INTO cities (city_id, name, country) VALUES
(1,'London','GB');
```

```
INSERT INTO cities (city_id, name, country) VALUES
(2,'New York City','US');
```

```
mysql> INSERT INTO cities (city_id, name, country) VALUES (2,'New York City','US');
Query OK, 1 row affected (0.01 sec)
```

SQL Insert data, 2

When used from inside Python, PostgreSQL use the following:

```
"""INSERT INTO cities (city_id, name, country)
    VALUES (%s, %s, %s)""",
    (2,'New York City','US')
```

Select data

The SELECT statement is used to select data from a database.

Table Name: cities

city_id	name	country
1	London	GB
2	New York City	US

```
SELECT * FROM cities;
```

```
mysql> SELECT * FROM cities;
+---------+---------------+---------+
| city_id | name          | country |
+---------+---------------+---------+
|       1 | London        | GB      |
|       2 | New York City | US      |
+---------+---------------+---------+
2 rows in set (0.00 sec)
```

```
SELECT *
FROM cities
WHERE country = 'GB';
```

```
mysql> SELECT * FROM cities WHERE country = 'GB';
+---------+--------+---------+
| city_id | name   | country |
+---------+--------+---------+
|       1 | London | GB      |
+---------+--------+---------+
1 row in set (0.00 sec)
```

Update data

```
UPDATE cities SET country='UK' WHERE city_id=1;
```

SQL> SELECT * FROM cities;

```
mysql> SELECT * FROM cities;
+---------+---------------+---------+
| city_id | name          | country |
+---------+---------------+---------+
|       1 | London        | UK      |
|       2 | New York City | US      |
+---------+---------------+---------+
2 rows in set (0.00 sec)
```

Quiz! Let's create another table

Cannot be empty

writer_id	writer_fname	writer_lname	writer_gender	writer_birth_year
100	Agatha	Christie	F	1890
101	J.R.R.	Tolkien	M	1892
102	Charles	Dickens	M	1812
103	J.K.	Rowling	F	1965

```
CREATE TABLE writers
(
        writer_id INTEGER PRIMARY KEY,
        writer_fname VARCHAR(24) NOT NULL,
        writer_lname varchar(24) NOT NULL,
        writer_gender varchar(1),
        writer_birth_year INTEGER

);
```

Insert data by the row

```
INSERT INTO WRITERS (WRITER_ID, WRITER_FNAME, WRITER_LNAME, WRITER_GENDER,
WRITER_BIRTH_YEAR)
      VALUES (100, 'AGATHA','CHRISTIE','F',1890);


INSERT INTO WRITERS (WRITER_ID, WRITER_FNAME, WRITER_LNAME, WRITER_GENDER,
WRITER_BIRTH_YEAR)
      VALUES  (101, 'J.R.R.','TOLKIEN','M',1892);

INSERT INTO WRITERS (WRITER_ID, WRITER_FNAME, WRITER_LNAME, WRITER_GENDER,
WRITER_BIRTH_YEAR)
      VALUES  (105, 'Rowland','Atkinson', NULL, NULL);
```

Insert several rows

```
INSERT INTO writers (writer_id, writer_fname, writer_lname,
writer_gender, writer_birth_year)
      VALUES (100, 'Agatha','Christie','F',1890),
             (101, 'J.R.R.','Tolkien','M',1892),
              (102, 'Charles','Dickens','M',1812),
              (103,'J.K.', 'Rowling','F',1965);
```

```
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Quiz

- Let us assume that we have a **customers** table

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

- Identify the customers primary key
- What is the data type of **CustomerName**?
- What is the data type of **Address**?

Quiz solution

- Let us assume that we have a **customers** table

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

- Identify the customers primary key: CustomerID
- What is the data type of **CustomerName**? VARCHAR(25)
- What is the data type of **Address**? TEXT

Select data using WHERE clause to filter records

```
SELECT writer_fname, writer_lname
FROM writers
WHERE writer_gender = 'F';
```

writer_id	writer_fname	writer_lname	writer_gender	writer_birth_year
100	Agatha	Christie	F	1890
101	J.R.R.	Tolkien	M	1892
102	Charles	Dickens	M	1812
103	J.K.	Rowling	F	1965

```
mysql> SELECT writer_fname, writer_lname FROM writers WHERE writer_gender = 'F';
+--------------+--------------+
| writer_fname | writer_lname |
+--------------+--------------+
| Agatha       | Christie     |
| J.K.         | Rowling      |
+--------------+--------------+
2 rows in set (0.00 sec)
```

Select data using WHERE clause to filter records cont.

```
SELECT writer_fname, writer_lname
FROM writers
WHERE writer_birth_year<1900;
```

writer_id	writer_fname	writer_lname	writer_gender	writer_birth_year
100	Agatha	Christie	F	1890
101	J.R.R.	Tolkien	M	1892
102	Charles	Dickens	M	1812
103	J.K.	Rowling	F	1965

```
mysql> SELECT writer_fname, writer_lname FROM writers WHERE writer_birth_year<1900;
+--------------+--------------+
| writer_fname | writer_lname |
+--------------+--------------+
| Agatha       | Christie     |
| J.R.R.       | Tolkien      |
| Charles      | Dickens      |
+--------------+--------------+
3 rows in set (0.00 sec)
```

Select data using where and logical condition

```
SELECT writer_fname, writer_lname,
writer_birth_year
FROM writers WHERE writer_birth_year>1900 AND
writer_gender='F';
```

writer_id	writer_fname	writer_lname	writer_gender	writer_birth_year
100	Agatha	Christie	F	1890
101	J.R.R.	Tolkien	M	1892
102	Charles	Dickens	M	1812
103	J.K.	Rowling	F	1965

```
mysql> SELECT writer_fname, writer_lname, writer_birth_year FROM writers WHERE writer_birth_year>1900 AND writer_gender=
'F';
+--------------+--------------+-------------------+
| writer_fname | writer_lname | writer_birth_year |
+--------------+--------------+-------------------+
| J.K.         | Rowling      |              1965 |
+--------------+--------------+-------------------+
1 row in set (0.00 sec)
```

Quiz: the "Customers" table

1. Select "CustomerName" and "City"

2. Select data on from customers *living in Berlin*

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Quiz solution

1. Select the "CustomerName" and "City" columns from the "Customers" table.
2. Select all the data from customer name, address and city for customers living in Berlin.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

1.
```
SELECT CustomerName, City
FROM customers;
```

2.
```
SELECT CustomerName, Address
FROM customers
WHERE City='Berlin';
```

Count data using count function

```
SELECT COUNT(writer_gender)
FROM writers
WHERE writer_birth_year>1900;
```

writer_id	writer_fname	writer_lname	writer_gender	writer_birth_year
100	Agatha	Christie	F	1890
101	J.R.R.	Tolkien	M	1892
102	Charles	Dickens	M	1812
103	J.K.	Rowling	F	1965

```
mysql> SELECT COUNT(writer_gender) FROM writers WHERE writer_birth_year>1900;
+----------------------+
| COUNT(writer_gender) |
+----------------------+
|                    1 |
+----------------------+
1 row in set (0.00 sec)
```

More on Select data

- The “GROUP BY” statement, groups rows that have the same values into summary rows:
  “count the number of genders".

```
SELECT writer_gender, COUNT(writer_gender)
FROM writers
GROUP BY writer_gender;
```

writer_id	writer_fname	writer_lname	writer_gender	writer_birth_year
100	Agatha	Christie	F	1890
101	J.R.R.	Tolkien	M	1892
102	Charles	Dickens	M	1812
103	J.K.	Rowling	F	1965

```
mysql> SELECT writer_gender, COUNT(writer_gender) FROM writers GROUP BY writer_gender;
+---------------+----------------------+
| writer_gender | COUNT(writer_gender) |
+---------------+----------------------+
| F             |                    2 |
| M             |                    2 |
+---------------+----------------------+
2 rows in set (0.00 sec)
```

Delete data by filtering records

```
DELETE FROM writers
WHERE writer_birth_year=1965;
```

writer_id	writer_fname	writer_lname	writer_gender	writer_birth_year
100	Agatha	Christie	F	1890
101	J.R.R.	Tolkien	M	1892
102	Charles	Dickens	M	1812
103	J.K.	Rowling	F	1965

SQL> SELECT * FROM writers;

```
mysql> SELECT * FROM writers;
+-----------+--------------+--------------+---------------+-------------------+
| writer_id | writer_fname | writer_lname | writer_gender | writer_birth_year |
+-----------+--------------+--------------+---------------+-------------------+
|       100 | Agatha       | Christie     | F             |              1890 |
|       101 | J.R.R.       | Tolkien      | M             |              1892 |
|       102 | Charles      | Dickens      | M             |              1812 |
+-----------+--------------+--------------+---------------+-------------------+
3 rows in set (0.00 sec)
```

ALTER TABLE: Change the structure (schema)

```
ALTER TABLE writers ADD number_of_books INTEGER;
```

SQL> DESCRIBE writers;

```
mysql>  DESCRIBE writers;
+-------------------+-------------+------+-----+---------+-------+
| Field             | Type        | Null | Key | Default | Extra |
+-------------------+-------------+------+-----+---------+-------+
| writer_id         | int(11)     | NO   | PRI | NULL    |       |
| writer_fname      | varchar(24) | NO   |     | NULL    |       |
| writer_lname      | varchar(24) | NO   |     | NULL    |       |
| writer_gender     | varchar(1)  | YES  |     | NULL    |       |
| writer_birth_year | int(11)     | YES  |     | NULL    |       |
| number_of_books   | int(11)     | YES  |     | NULL    |       |
+-------------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Let’s add new values (update…)

```
UPDATE writers
SET number_of_books=73
WHERE writer_id=100;

UPDATE writers
SET number_of_books=26
WHERE WRITER_ID=101;

UPDATE writers
SET number_of_books=20
WHERE writer_id=102;
```

The AVG() function returns the average value of a numeric column.

```
SELECT AVG(number_of_books) FROM writers WHERE
writer_gender='M';
```

writer_id	writer_fname	writer_lname	writer_gender	writer_birth_year	number_of_books
100	Agatha	Christie	F	1890	73
101	J.R.R.	Tolkien	M	1892	26
102	Charles	Dickens	M	1812	20

```
mysql> SELECT AVG(number_of_books) FROM writers WHERE writer_gender='M';
+----------------------+
| AVG(number_of_books) |
+----------------------+
|              23.0000 |
+----------------------+
1 row in set (0.00 sec)
```

Summary on SQL functions

COUNT() Syntax

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

AVG() Syntax

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

SUM() Syntax

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

Combine function with a group by…

```
SELECT AVG(number_of_books)
FROM writers
GROUP BY writer_gender;
```

writer_id	writer_fname	writer_lname	writer_gender	writer_birth_year	number_of_books
100	Agatha	Christie	F	1890	73
101	J.R.R.	Tolkien	M	1892	26
102	Charles	Dickens	M	1812	20

```
mysql> SELECT AVG(number_of_books) FROM writers GROUP BY writer_gender;
+----------------------+
| AVG(number_of_books) |
+----------------------+
|              73.0000 |
|              23.0000 |
+----------------------+
2 rows in set (0.00 sec)
```

More on select…

```
SELECT writer_gender,SUM(number_of_books),
AVG(number_of_books) FROM writers GROUP BY writer_gender;
```

writer_id	writer_fname	writer_lname	writer_gender	writer_birth_year	number_of_books
100	Agatha	Christie	F	1890	73
101	J.R.R.	Tolkien	M	1892	26
102	Charles	Dickens	M	1812	20

```
mysql> SELECT writer_gender,SUM(number_of_books),AVG(number_of_books) FROM writers GROUP BY writer_gender;
+---------------+----------------------+----------------------+
| writer_gender | SUM(number_of_books) | AVG(number_of_books) |
+---------------+----------------------+----------------------+
| F             |                   73 |              73.0000 |
| M             |                   46 |              23.0000 |
+---------------+----------------------+----------------------+
2 rows in set (0.00 sec)
```