EMERITUS

INSTITUTE OF MANAGEMENT

WEEK 6

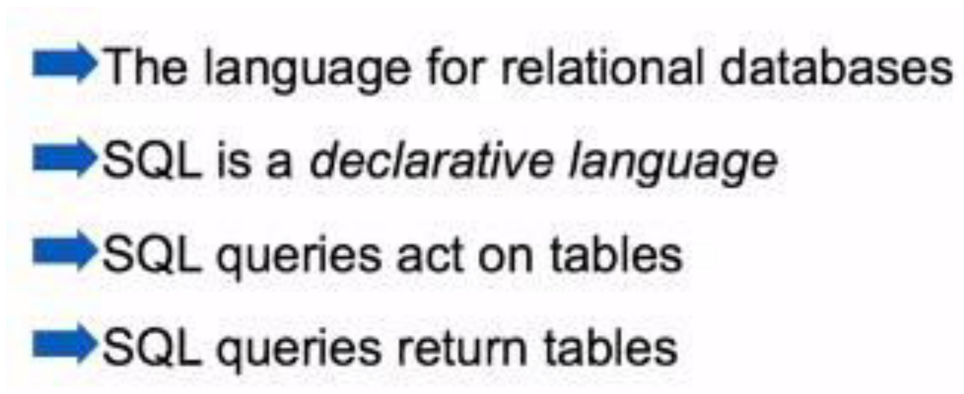
SQL — UBIQUITOUS DATABASE FORMAT/LANGUAGE

MITSloan Executive Education

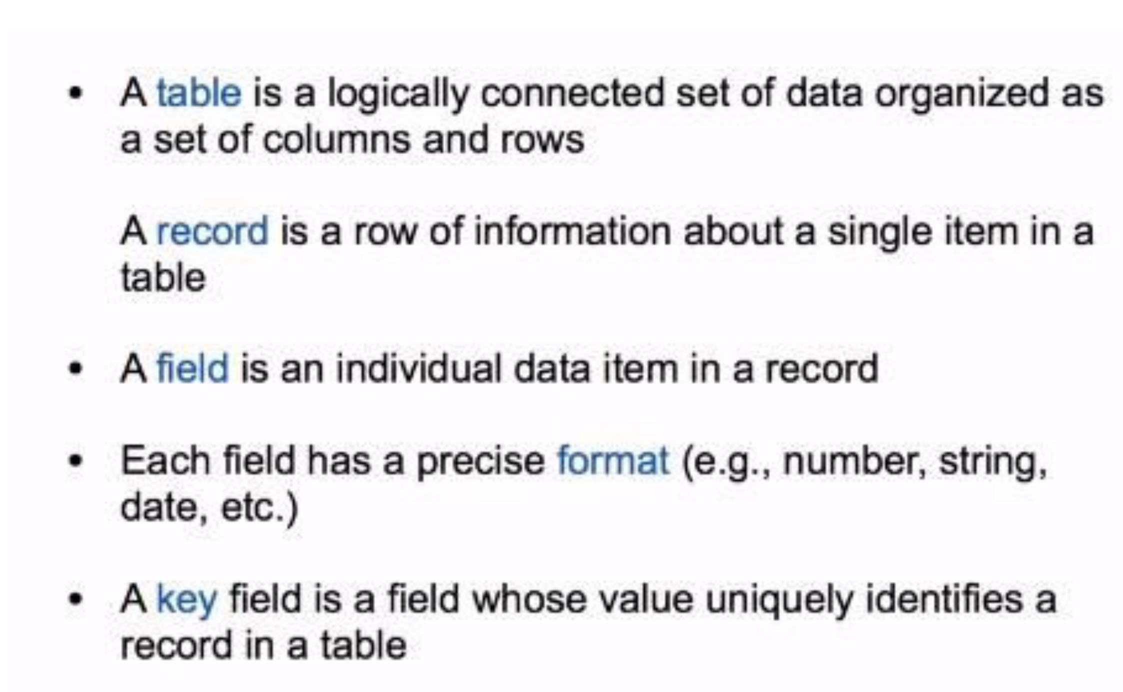
Columbia Business School
AT THE VERY CENTER OF BUSINESS™
EXECUTIVE EDUCATION

Tuck
EXECUTIVE EDUCATION
at Dartmouth

SQL: Structured Query Language and its Elements

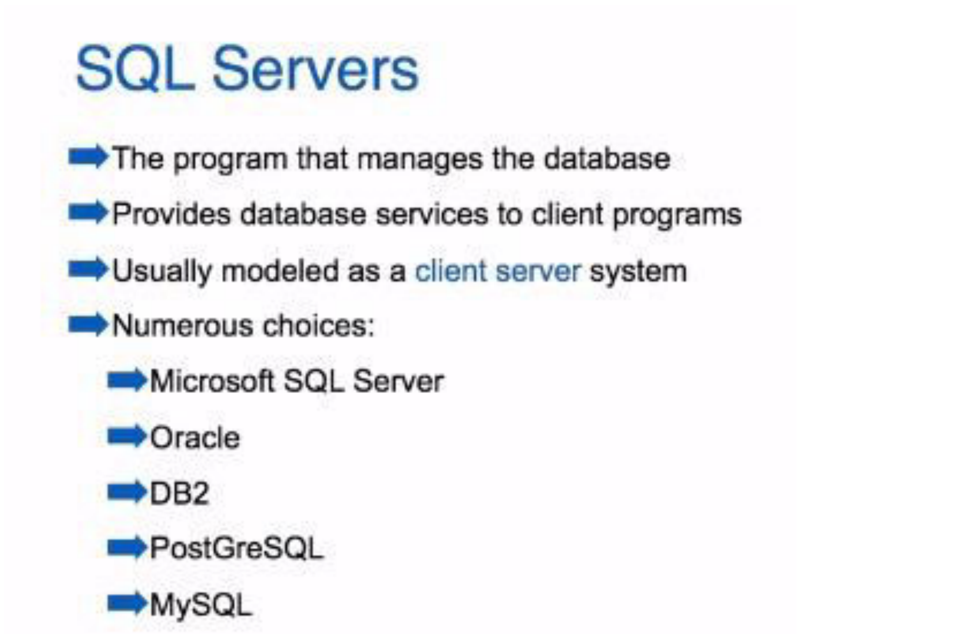


- The language for relational databases
- SQL is a *declarative language*
- SQL queries act on tables
- SQL queries return tables



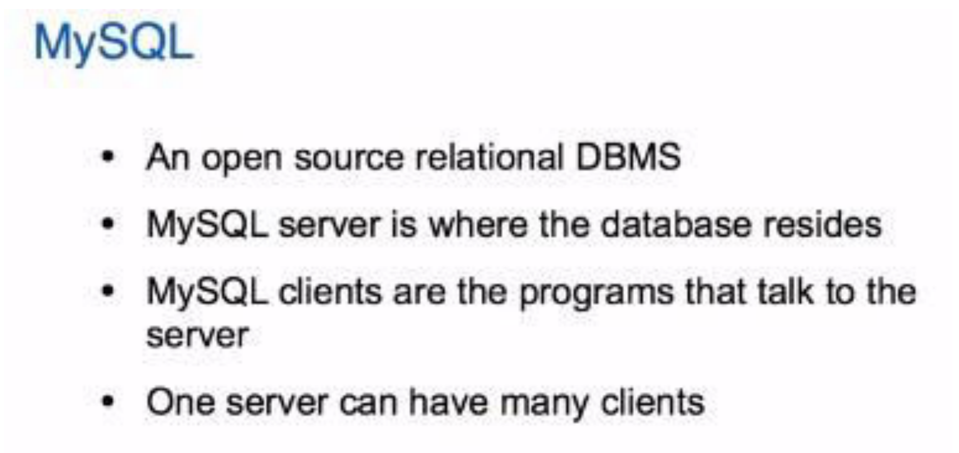
- A table is a logically connected set of data organized as a set of columns and rows

  A record is a row of information about a single item in a table
- A field is an individual data item in a record
- Each field has a precise format (e.g., number, string, date, etc.)
- A key field is a field whose value uniquely identifies a record in a table



SQL Servers

- The program that manages the database
- Provides database services to client programs
- Usually modeled as a client server system
- Numerous choices:
  - Microsoft SQL Server
  - Oracle
  - DB2
  - PostGreSQL
  - MySQL



MySQL

- An open source relational DBMS
- MySQL server is where the database resides
- MySQL clients are the programs that talk to the server
- One server can have many clients

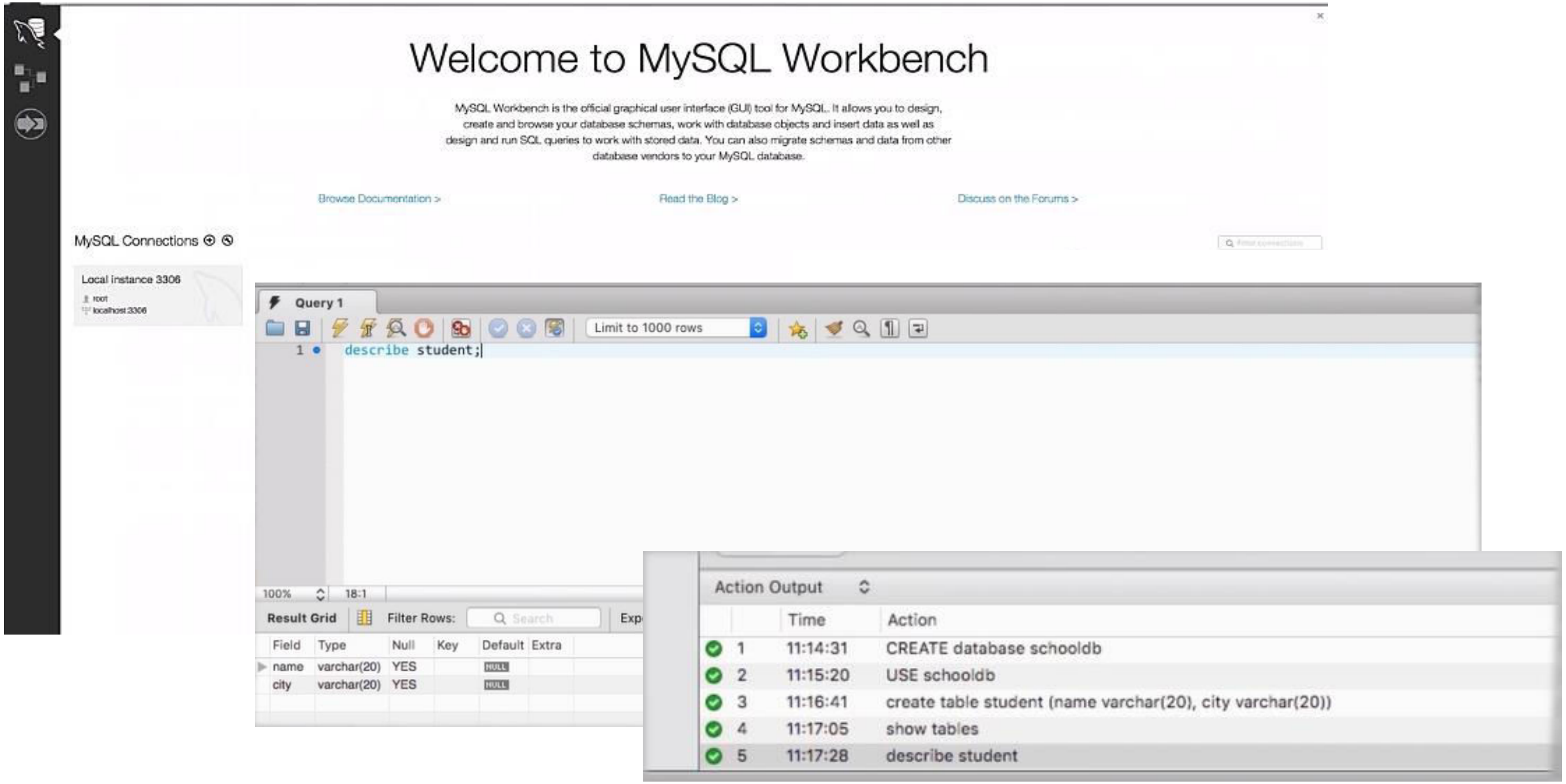
Query Types

- Data definitional
  create, alter, drop
- Data manipulation
  insert, delete, update
- Data querying
  select
  select where
  select join

- List existing databases:
  - SHOW databases;
- Create new database:
  - CREATE database IF NOT EXISTS SchoolDB;
- Choose a DB to work on
  - USE SchoolDB;
- Delete a database:
  - DROP DATABASE SchoolDB;

Introduction to MySQL Workbench

MySQL Workbench is a GUI client for MySQL



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

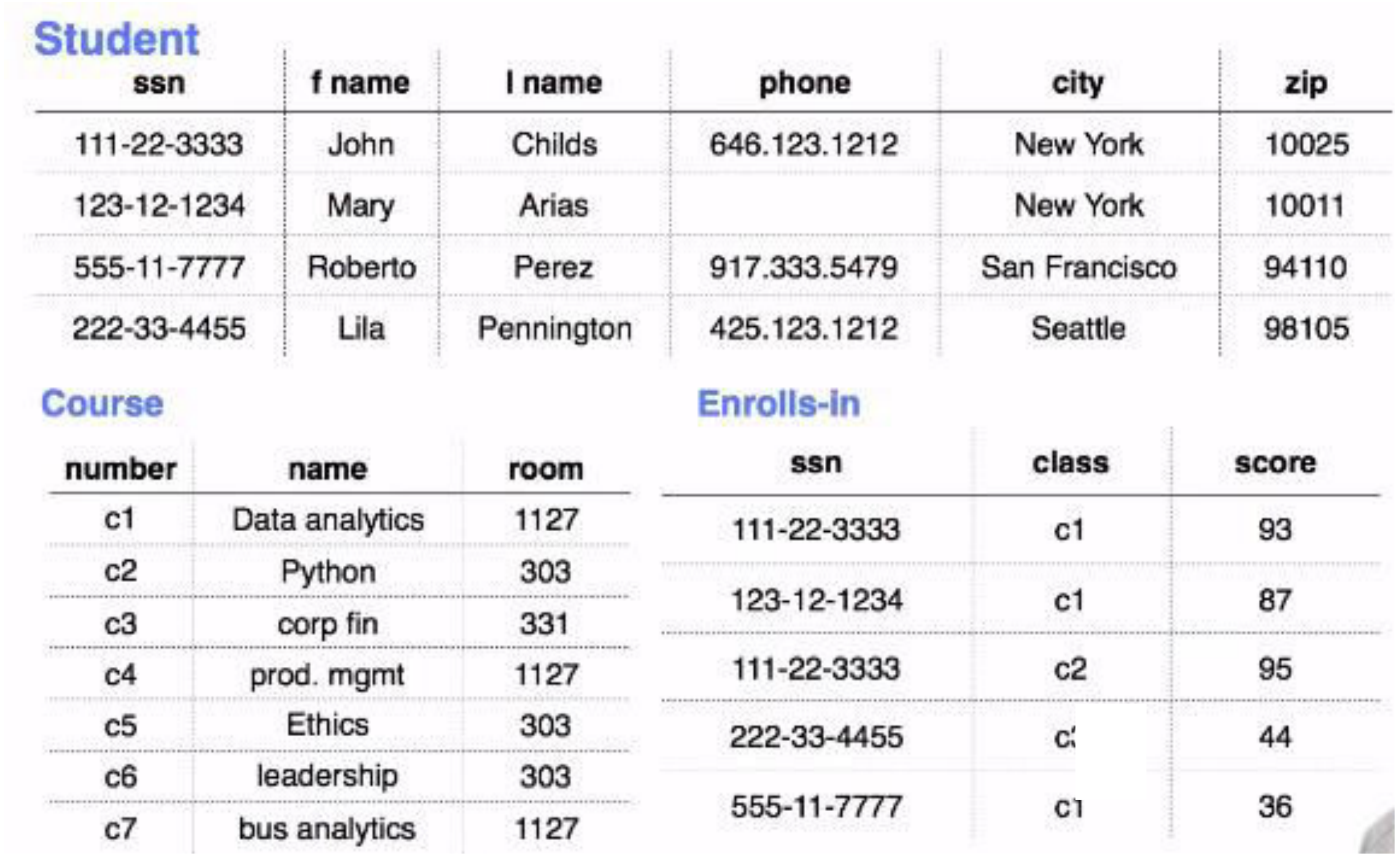
```
describe student;
```

Field	Type	Null	Key	Default	Extra
name	varchar(20)	YES		NULL	
city	varchar(20)	YES		NULL	

Action Output

		Time	Action
✓	1	11:14:31	CREATE database schooldb
✓	2	11:15:20	USE schooldb
✓	3	11:16:41	create table student (name varchar(20), city varchar(20))
✓	4	11:17:05	show tables
✓	5	11:17:28	describe student

Example Database



Student

ssn	f name	l name	phone	city	zip
111-22-3333	John	Childs	646.123.1212	New York	10025
123-12-1234	Mary	Arias		New York	10011
555-11-7777	Roberto	Perez	917.333.5479	San Francisco	94110
222-33-4455	Lila	Pennington	425.123.1212	Seattle	98105

Course

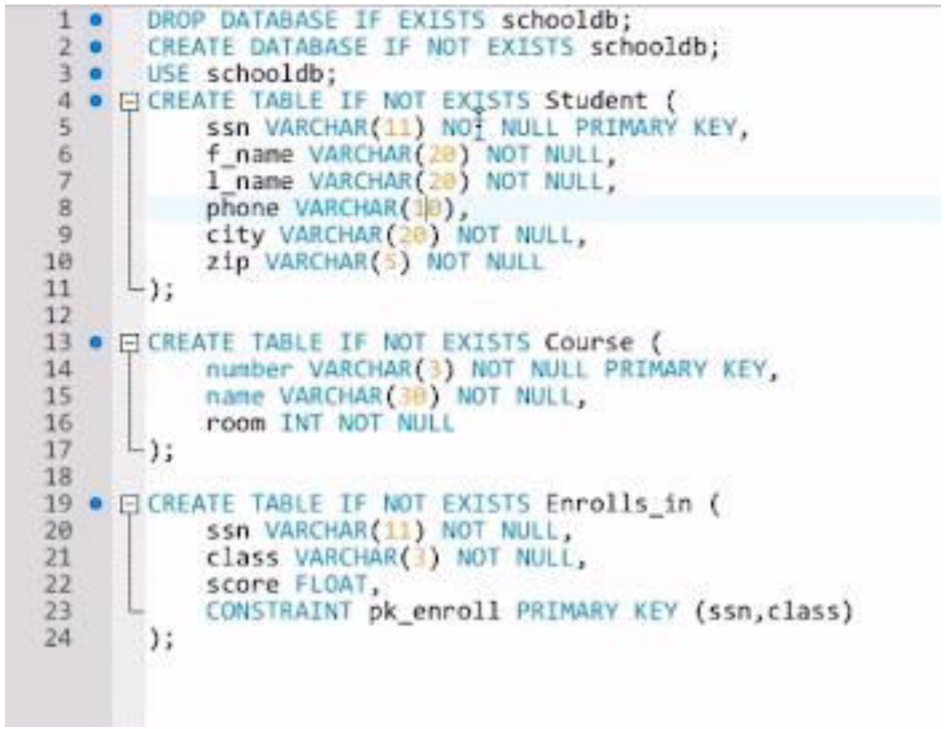
number	name	room
c1	Data analytics	1127
c2	Python	303
c3	corp fin	331
c4	prod. mgmt	1127
c5	Ethics	303
c6	leadership	303
c7	bus analytics	1127

Enrolls-in

ssn	class	score
111-22-3333	c1	93
123-12-1234	c1	87
111-22-3333	c2	95
222-33-4455	c3	44
555-11-7777	c1	36

Setup the Database

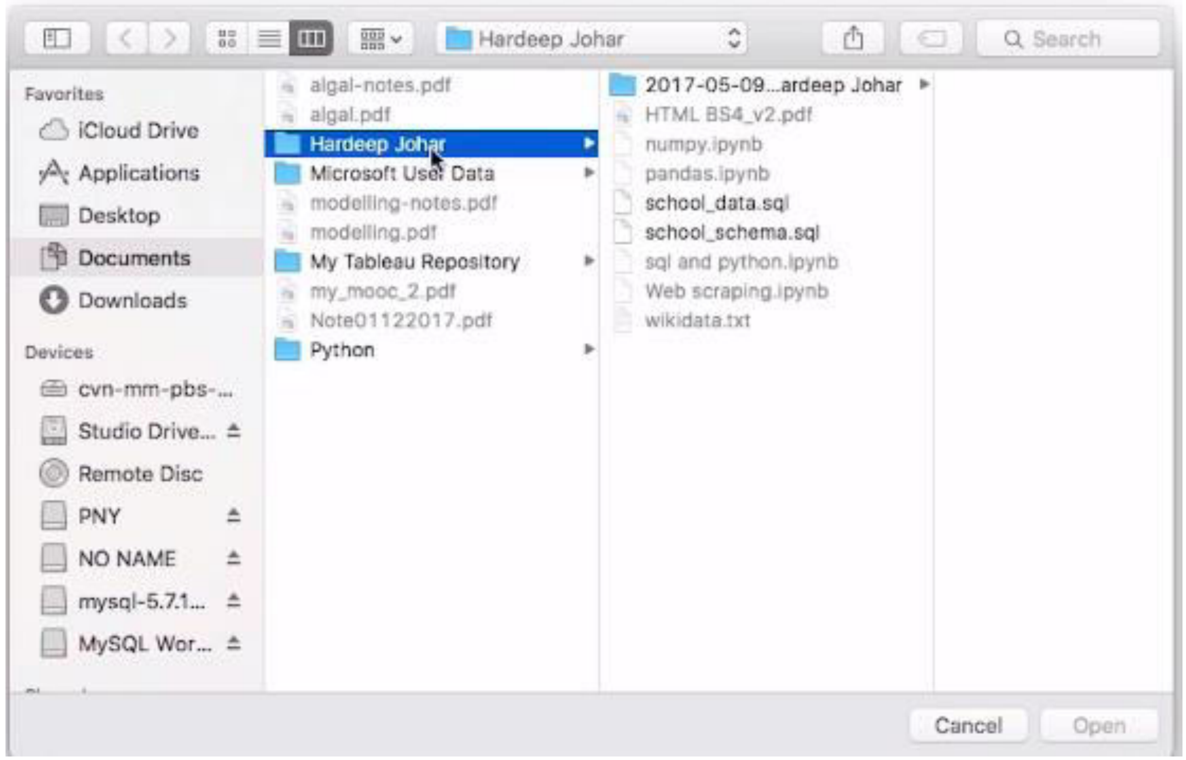
- Download the files
  - school_schema.sql
  - school_data.sql
- Open mysql workbench (see installation instructions)
- Open school_schema.sql (click th icon)
- Execute the script (click th icon)
- Repeat with the fine school_data.sql



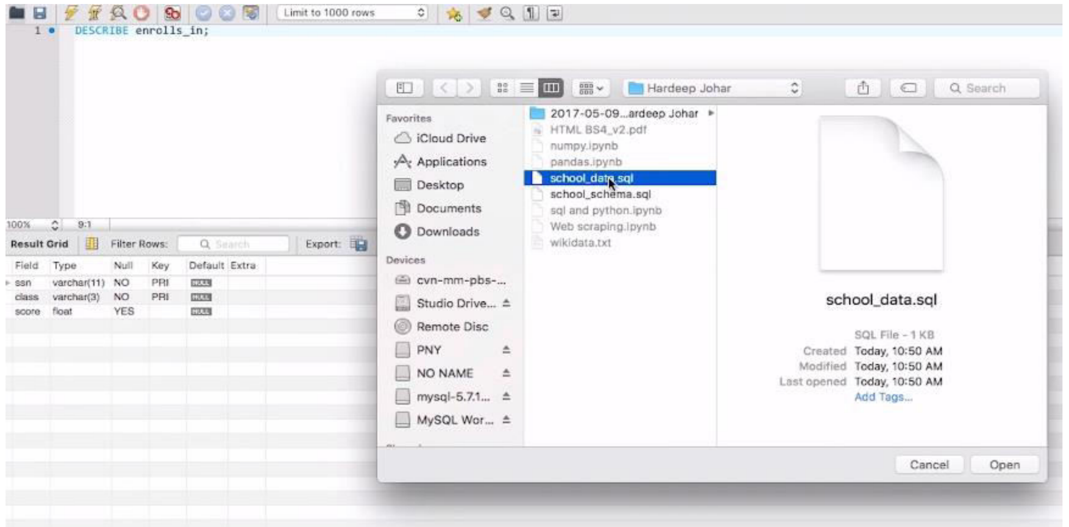
```sql
DROP DATABASE IF EXISTS schooldb;
CREATE DATABASE IF NOT EXISTS schooldb;
USE schooldb;
CREATE TABLE IF NOT EXISTS Student (
    ssn VARCHAR(11) NOT NULL PRIMARY KEY,
    f_name VARCHAR(20) NOT NULL,
    l_name VARCHAR(20) NOT NULL,
    phone VARCHAR(10),
    city VARCHAR(20) NOT NULL,
    zip VARCHAR(5) NOT NULL
);

CREATE TABLE IF NOT EXISTS Course (
    number VARCHAR(3) NOT NULL PRIMARY KEY,
    name VARCHAR(30) NOT NULL,
    room INT NOT NULL
);

CREATE TABLE IF NOT EXISTS Enrolls_in (
    ssn VARCHAR(11) NOT NULL,
    class VARCHAR(3) NOT NULL,
    score FLOAT,
    CONSTRAINT pk_enroll PRIMARY KEY (ssn,class)
);
```

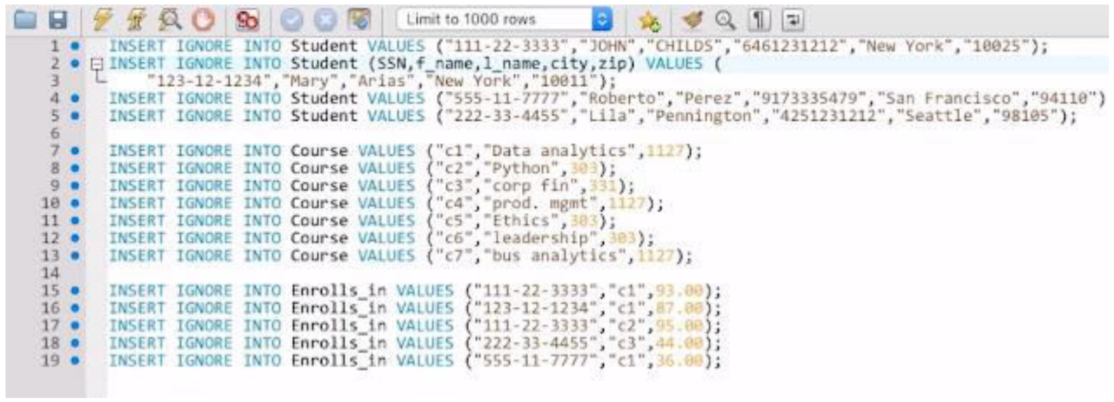


Database Example



```
DESCRIBE enrolls_in;
```

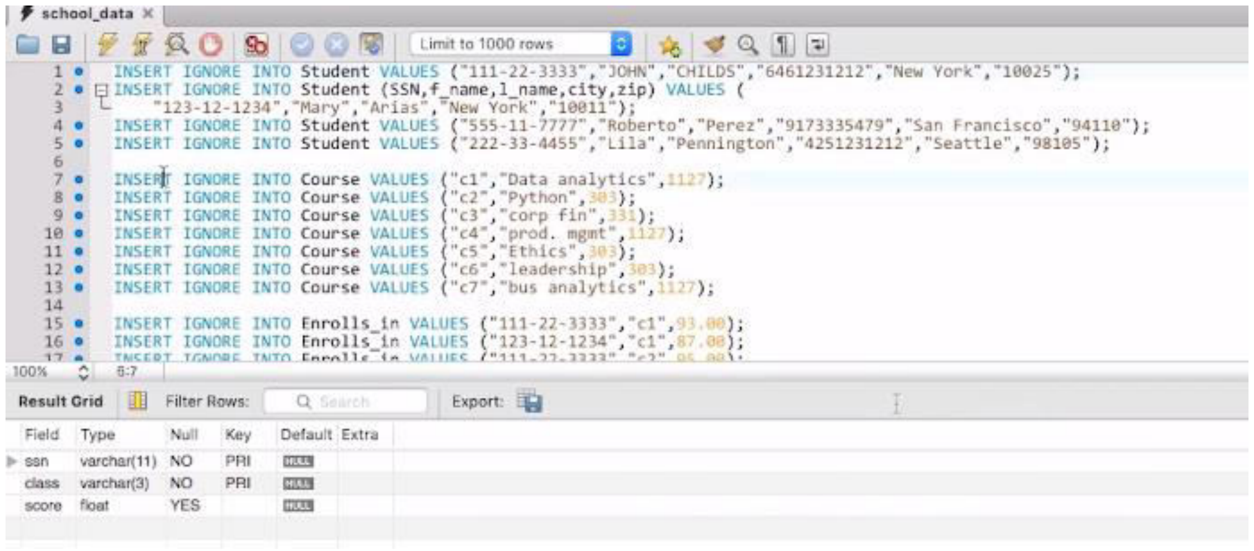
Field	Type	Null	Key	Default	Extra
ssn	varchar(11)	NO	PRI	NULL	
class	varchar(3)	NO	PRI	NULL	
score	float	YES		NULL	



```
INSERT IGNORE INTO Student VALUES ("111-22-3333","JOHN","CHILDS","6461231212","New York","10025");
INSERT IGNORE INTO Student (SSN,f_name,l_name,city,zip) VALUES (
    "123-12-1234","Mary","Arias","New York","10011");
INSERT IGNORE INTO Student VALUES ("555-11-7777","Roberto","Perez","9173335479","San Francisco","94110")
INSERT IGNORE INTO Student VALUES ("222-33-4455","Lila","Pennington","4251231212","Seattle","98105");

INSERT IGNORE INTO Course VALUES ("c1","Data analytics",1127);
INSERT IGNORE INTO Course VALUES ("c2","Python",303);
INSERT IGNORE INTO Course VALUES ("c3","corp fin",331);
INSERT IGNORE INTO Course VALUES ("c4","prod. mgmt",1127);
INSERT IGNORE INTO Course VALUES ("c5","Ethics",303);
INSERT IGNORE INTO Course VALUES ("c6","leadership",303);
INSERT IGNORE INTO Course VALUES ("c7","bus analytics",1127);

INSERT IGNORE INTO Enrolls_in VALUES ("111-22-3333","c1",93.00);
INSERT IGNORE INTO Enrolls_in VALUES ("123-12-1234","c1",87.00);
INSERT IGNORE INTO Enrolls_in VALUES ("111-22-3333","c2",95.00);
INSERT IGNORE INTO Enrolls_in VALUES ("222-33-4455","c3",44.00);
INSERT IGNORE INTO Enrolls_in VALUES ("555-11-7777","c1",36.00);
```

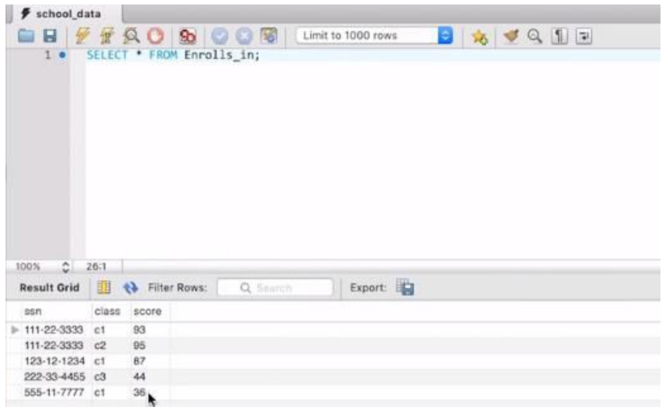


```
INSERT IGNORE INTO Student VALUES ("111-22-3333","JOHN","CHILDS","6461231212","New York","10025");
INSERT IGNORE INTO Student (SSN,f_name,l_name,city,zip) VALUES (
    "123-12-1234","Mary","Arias","New York","10011");
INSERT IGNORE INTO Student VALUES ("555-11-7777","Roberto","Perez","9173335479","San Francisco","94110");
INSERT IGNORE INTO Student VALUES ("222-33-4455","Lila","Pennington","4251231212","Seattle","98105");

INSERT IGNORE INTO Course VALUES ("c1","Data analytics",1127);
INSERT IGNORE INTO Course VALUES ("c2","Python",303);
INSERT IGNORE INTO Course VALUES ("c3","corp fin",331);
INSERT IGNORE INTO Course VALUES ("c4","prod. mgmt",1127);
INSERT IGNORE INTO Course VALUES ("c5","Ethics",303);
INSERT IGNORE INTO Course VALUES ("c6","leadership",303);
INSERT IGNORE INTO Course VALUES ("c7","bus analytics",1127);

INSERT IGNORE INTO Enrolls_in VALUES ("111-22-3333","c1",93.00);
INSERT IGNORE INTO Enrolls_in VALUES ("123-12-1234","c1",87.00);
```

Field	Type	Null	Key	Default	Extra
ssn	varchar(11)	NO	PRI	NULL	
class	varchar(3)	NO	PRI	NULL	
score	float	YES		NULL	



```
SELECT * FROM Enrolls_in;
```

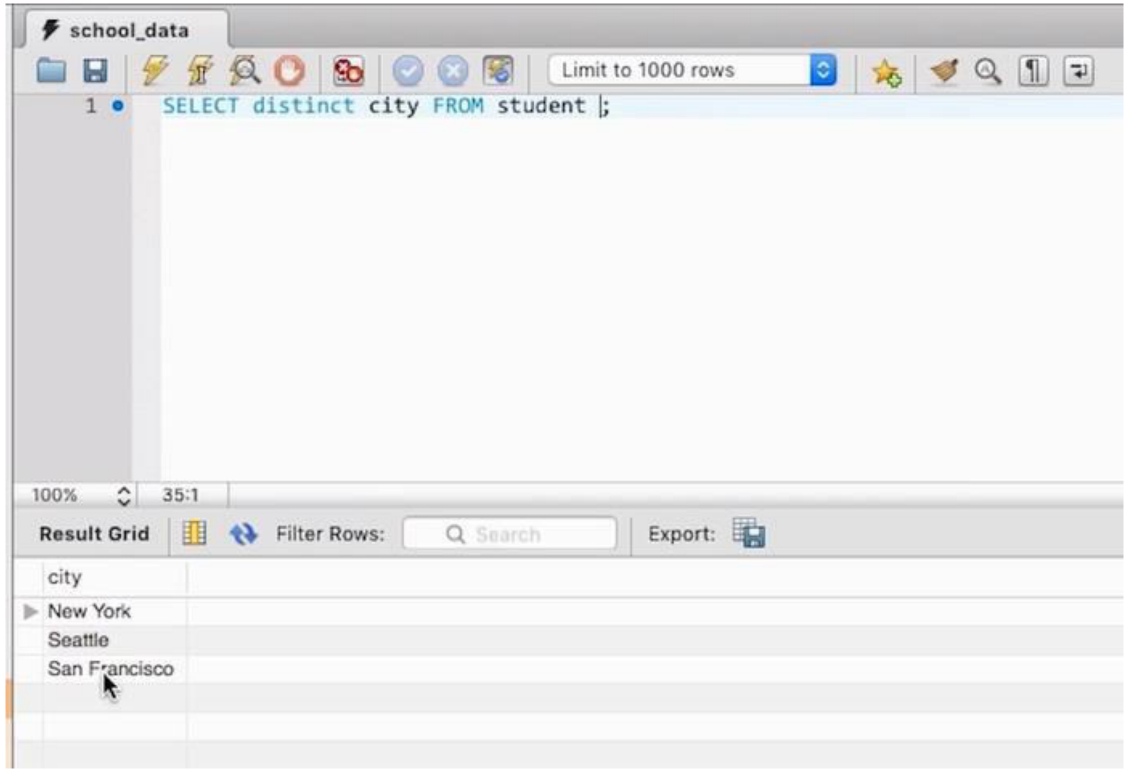
ssn	class	score
111-22-3333	c1	93
111-22-3333	c2	95
123-12-1234	c1	87
222-33-4455	c3	44
555-11-7777	c1	36

The Select Statement Part 1

- SELECT returns a set of records from one or more tables in a database
- SELECT describes the result set - processing is left to the server

Select

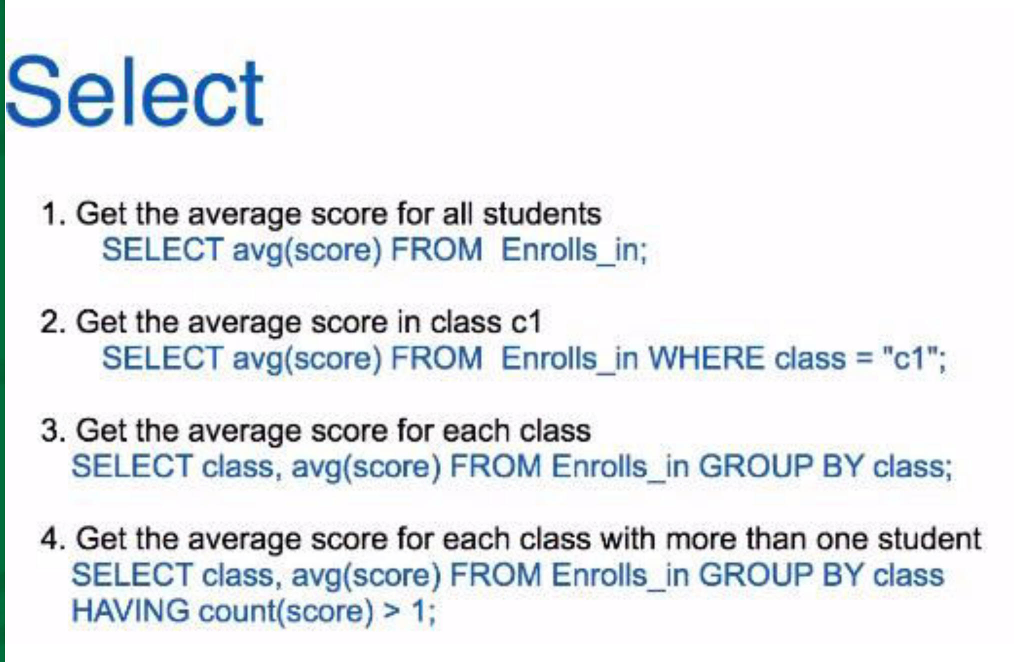
1. Get everything in a table
   SELECT * FROM Student;
2. Get all students from New York
   SELECT * FROM Student WHERE City = "New York";
3. Get only the names of students from New York
   SELECT f_name, l_name FROM Student WHERE city = "New York";
4. Get the ssn of students ordered by f_name
   SELECT ssn FROM Student ORDER BY f_name ASC;
   SELECT ssn FROM Student ORDER BY f_name DESC;
5. List the set of unique cities in the Student table
   SELECT DISTINCT city FROM Student;



```
SELECT distinct city FROM student ;
```

```
city
New York
Seattle
San Francisco
```

The Select Statement Part 2



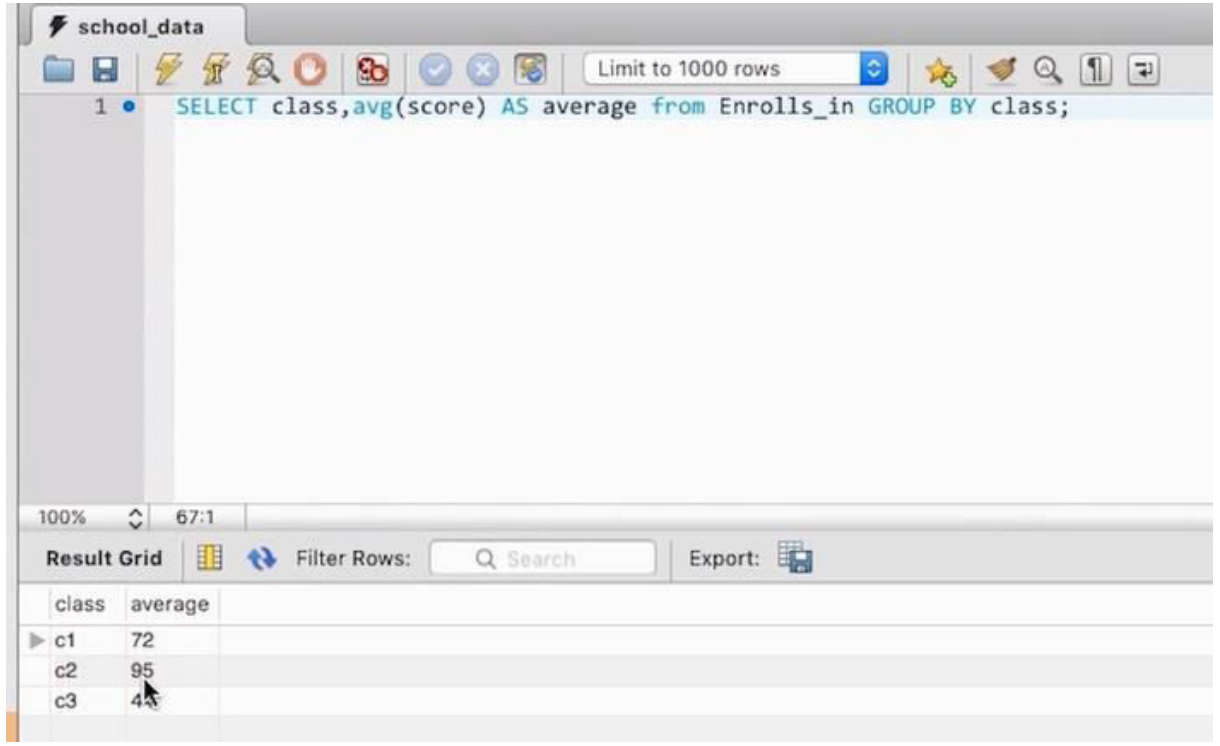
Select

1. Get the average score for all students
   SELECT avg(score) FROM Enrolls_in;

2. Get the average score in class c1
   SELECT avg(score) FROM Enrolls_in WHERE class = "c1";

3. Get the average score for each class
   SELECT class, avg(score) FROM Enrolls_in GROUP BY class;

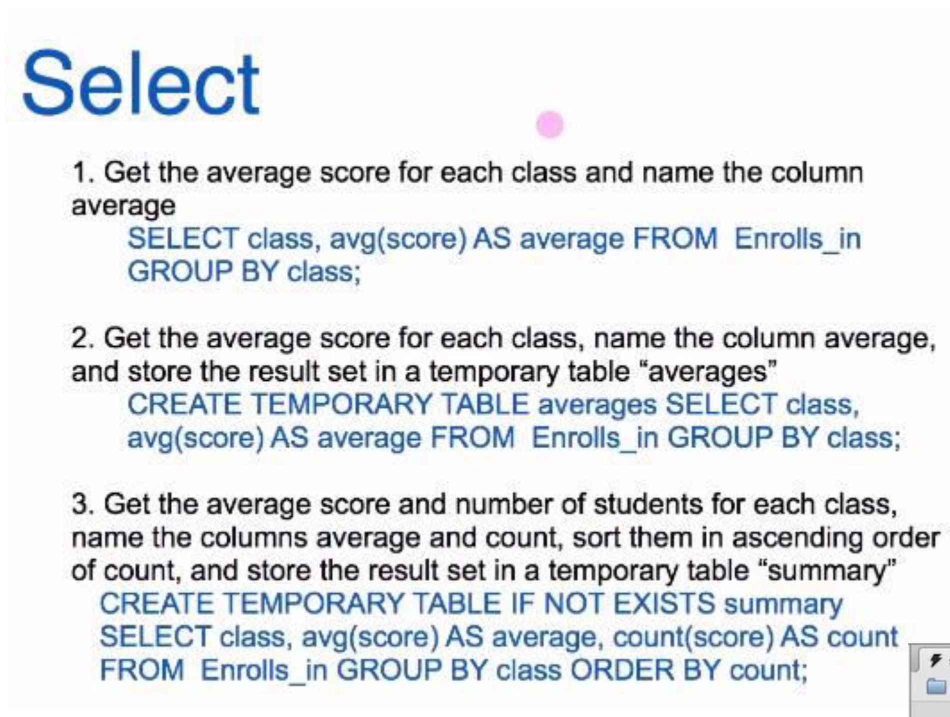
4. Get the average score for each class with more than one student
   SELECT class, avg(score) FROM Enrolls_in GROUP BY class
   HAVING count(score) > 1;



```
SELECT class,avg(score) AS average from Enrolls_in GROUP BY class;
```

class	average
c1	72
c2	95
c3	4[illegible]

The Select Statement Part 3

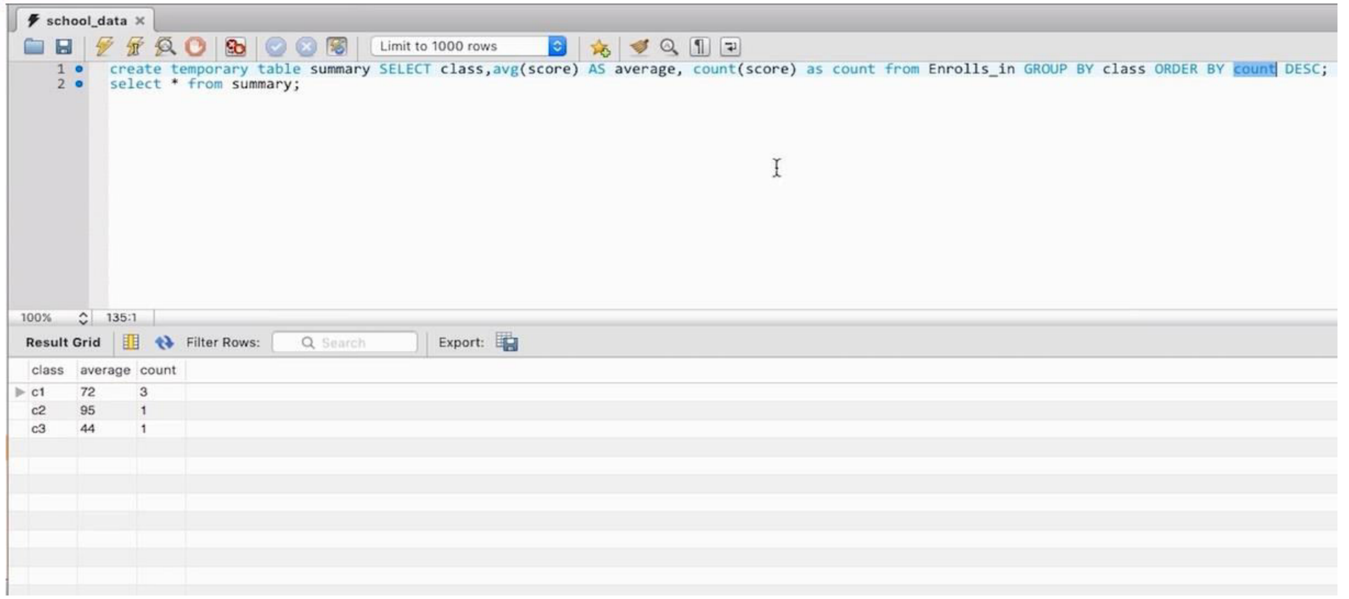


Select

1. Get the average score for each class and name the column average
   SELECT class, avg(score) AS average FROM Enrolls_in GROUP BY class;

2. Get the average score for each class, name the column average, and store the result set in a temporary table "averages"
   CREATE TEMPORARY TABLE averages SELECT class, avg(score) AS average FROM Enrolls_in GROUP BY class;

3. Get the average score and number of students for each class, name the columns average and count, sort them in ascending order of count, and store the result set in a temporary table "summary"
   CREATE TEMPORARY TABLE IF NOT EXISTS summary SELECT class, avg(score) AS average, count(score) AS count FROM Enrolls_in GROUP BY class ORDER BY count;



```
create temporary table summary SELECT class,avg(score) AS average, count(score) as count from Enrolls_in GROUP BY class ORDER BY count DESC;
select * from summary;
```

```
class  average  count
c1     72       3
c2     95       1
c3     44       1
```

Working Across Multiple Tables Part 1

1. Get a list of student names and classes

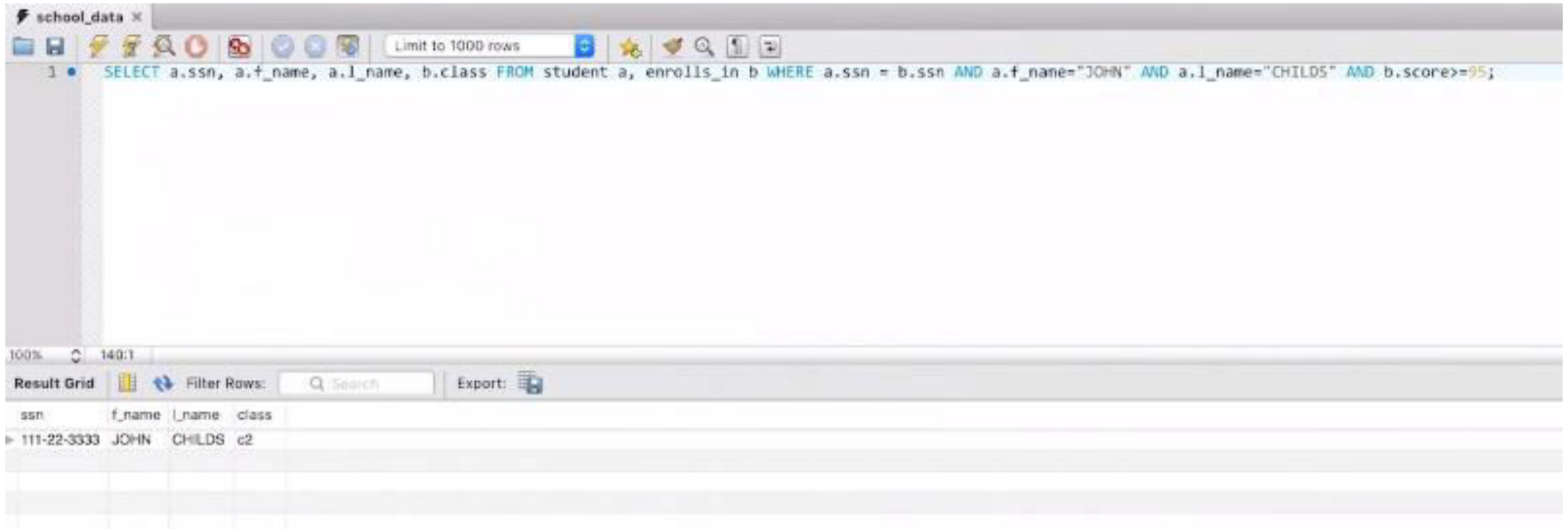
```
SELECT f_name,l_name,class FROM Student a, Enrolls_in b
WHERE a.ssn = b.ssn;
```

2. Get the list of classes that John Childs is enrolled in and his grade is 95 or greater

```
SELECT class FROM Student a, Enrolls_in b WHERE
a.f_name = "JOHN" AND a.l_name = "CHILDS" AND a.ssn =
b.ssn AND b.score >=95;
```

3. Get the names of the courses that John Childs is enrolled in

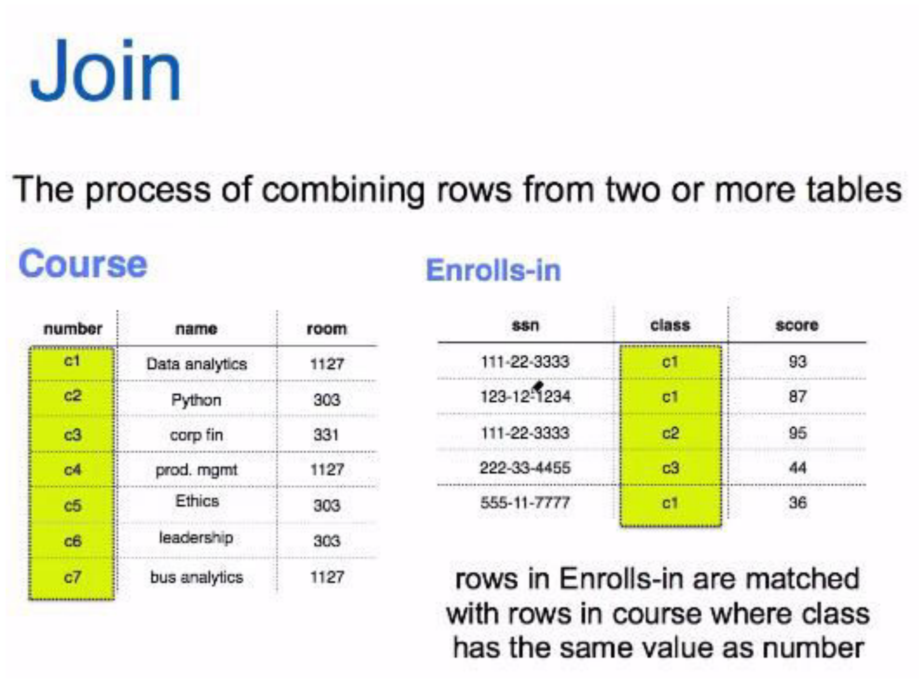
```
SELECT name from Student a, Enrolls_in b, Course c WHERE
a.f_name = "JOHN" AND a.l_name = "CHILDS" AND a.ssn =
b.ssn AND b.class = c.number;
```



```
SELECT a.ssn, a.f_name, a.l_name, b.class FROM student a, enrolls_in b WHERE a.ssn = b.ssn AND a.f_name="JOHN" AND a.l_name="CHILDS" AND b.score>=95;
```

ssn	f_name	l_name	class
111-22-3333	JOHN	CHILDS	c2

Working Across Multiple Tables Part 2



Join

The process of combining rows from two or more tables

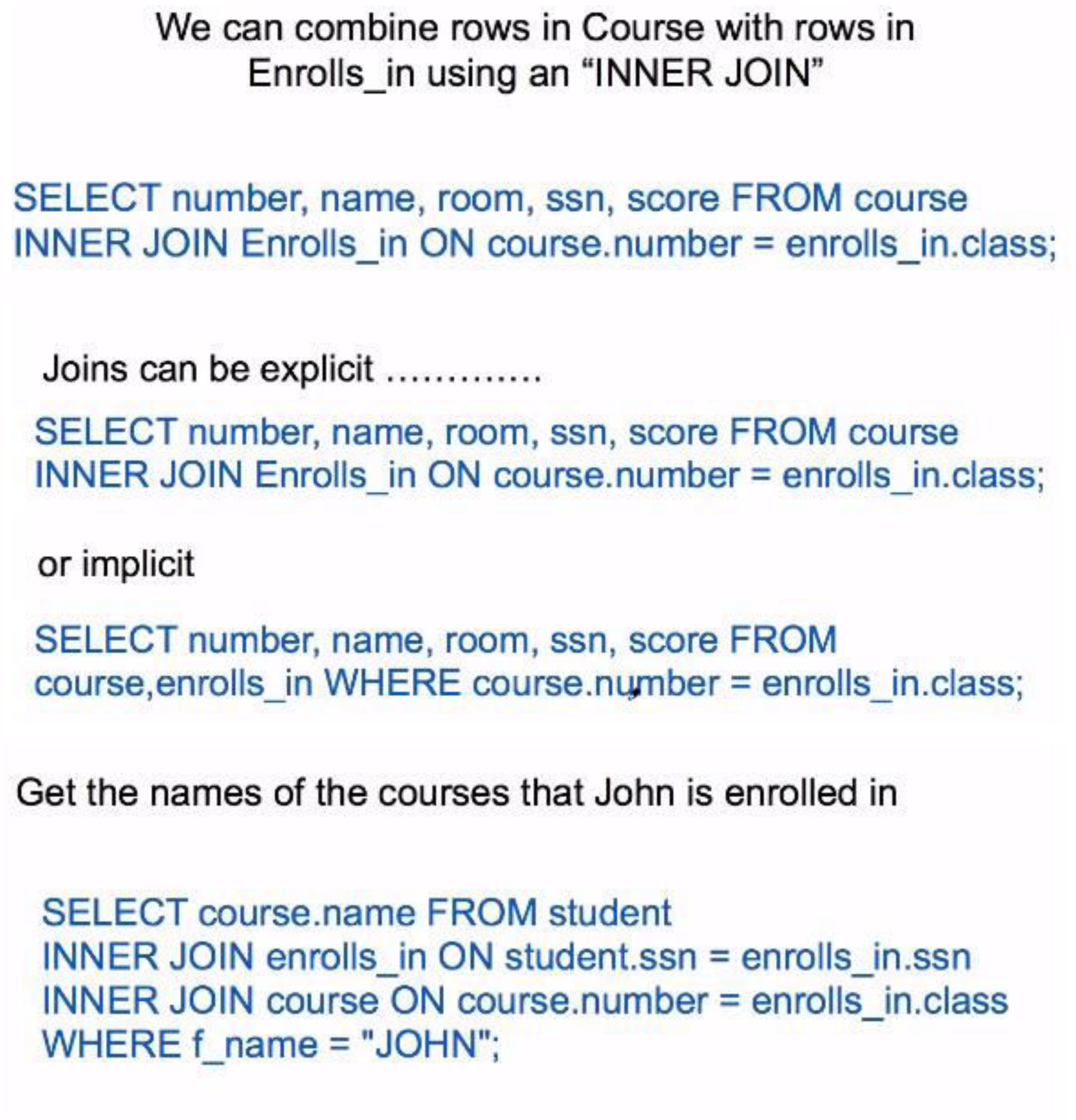
Course

number	name	room
c1	Data analytics	1127
c2	Python	303
c3	corp fin	331
c4	prod. mgmt	1127
c5	Ethics	303
c6	leadership	303
c7	bus analytics	1127

Enrolls-in

ssn	class	score
111-22-3333	c1	93
123-12-1234	c1	87
111-22-3333	c2	95
222-33-4455	c3	44
555-11-7777	c1	36

rows in Enrolls-in are matched with rows in course where class has the same value as number



We can combine rows in Course with rows in Enrolls_in using an "INNER JOIN"

```
SELECT number, name, room, ssn, score FROM course
INNER JOIN Enrolls_in ON course.number = enrolls_in.class;
```

Joins can be explicit ....

```
SELECT number, name, room, ssn, score FROM course
INNER JOIN Enrolls_in ON course.number = enrolls_in.class;
```

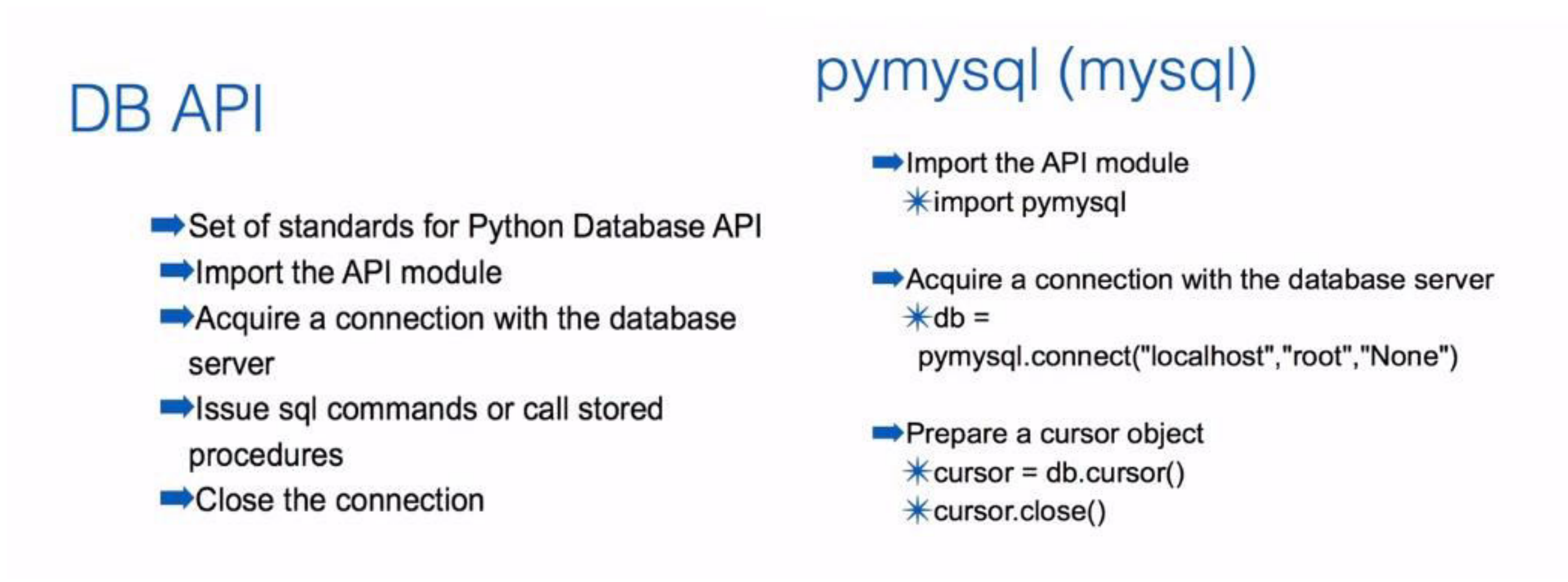
or implicit

```
SELECT number, name, room, ssn, score FROM
course,enrolls_in WHERE course.number = enrolls_in.class;
```

Get the names of the courses that John is enrolled in

```
SELECT course.name FROM student
INNER JOIN enrolls_in ON student.ssn = enrolls_in.ssn
INNER JOIN course ON course.number = enrolls_in.class
WHERE f_name = "JOHN";
```

Using Python for SQL



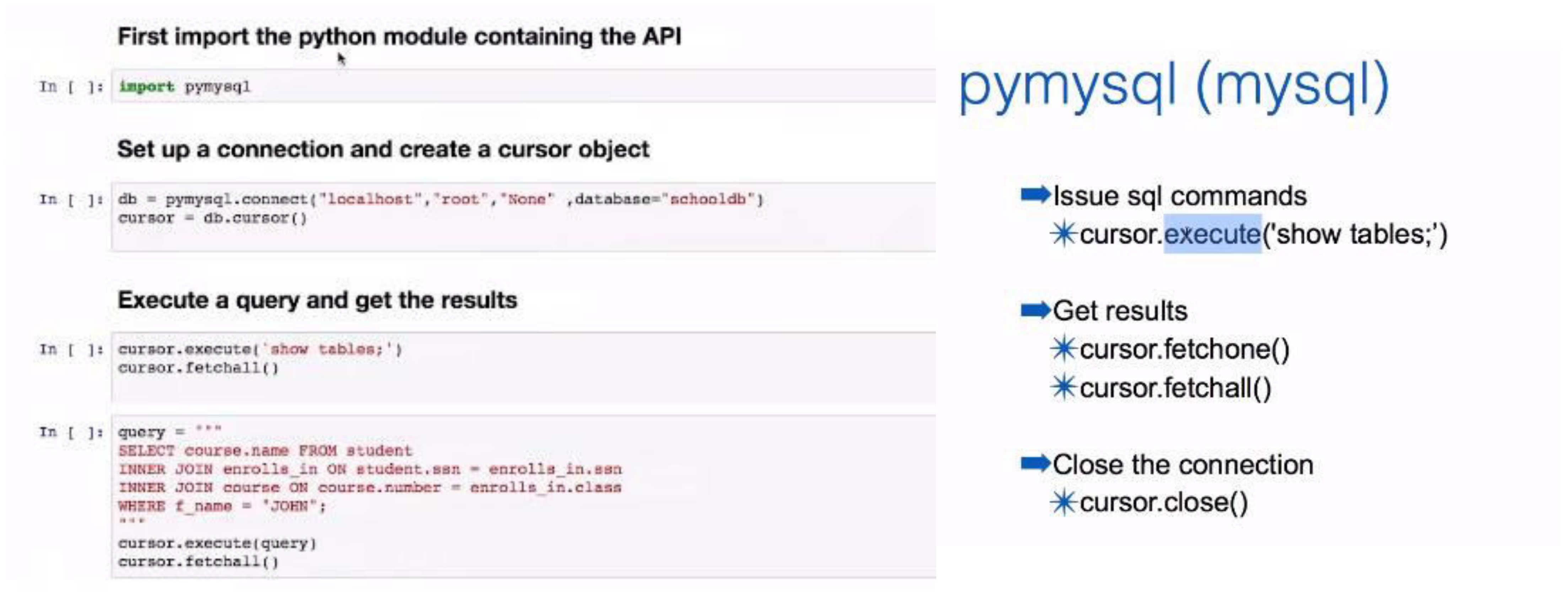
DB API

- ➡Set of standards for Python Database API
  - ➡Import the API module
  - ➡Acquire a connection with the database server
  - ➡Issue sql commands or call stored procedures
  - ➡Close the connection

pymysql (mysql)

- ➡Import the API module
  - ✳import pymysql
- ➡Acquire a connection with the database server
  - ✳db = pymysql.connect("localhost","root","None")
- ➡Prepare a cursor object
  - ✳cursor = db.cursor()
  - ✳cursor.close()

Python and MYSQL



First import the python module containing the API

```
In [ ]: import pymysql
```

Set up a connection and create a cursor object

```
In [ ]: db = pymysql.connect("localhost","root","None" ,database="schooldb")
        cursor = db.cursor()
```

Execute a query and get the results

```
In [ ]: cursor.execute('show tables;')
        cursor.fetchall()
```

```
In [ ]: query = """
        SELECT course.name FROM student
        INNER JOIN enrolls_in ON student.ssn = enrolls_in.ssn
        INNER JOIN course ON course.number = enrolls_in.class
        WHERE f_name = 'JOHN';
        """
        cursor.execute(query)
        cursor.fetchall()
```

pymysql (mysql)

- ➡Issue sql commands
  - ✶cursor.execute('show tables;')
- ➡Get results
  - ✶cursor.fetchone()
  - ✶cursor.fetchall()
- ➡Close the connection
  - ✶cursor.close()

GLOBAL IVY EDUCATION

www.emeritus.org