

SQL – Part I

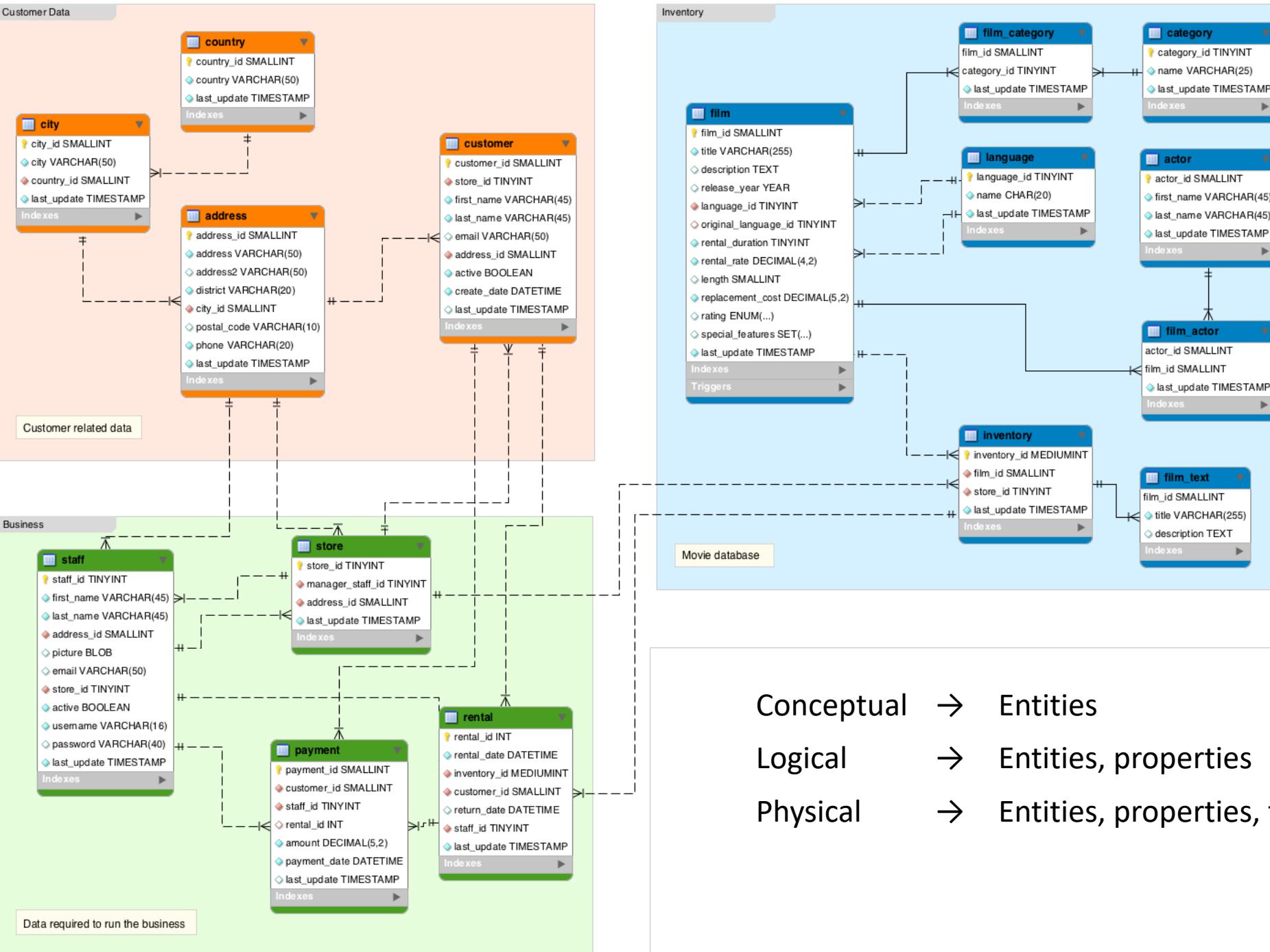
Structured Query Language

Database Design

conceptual, logical, physical

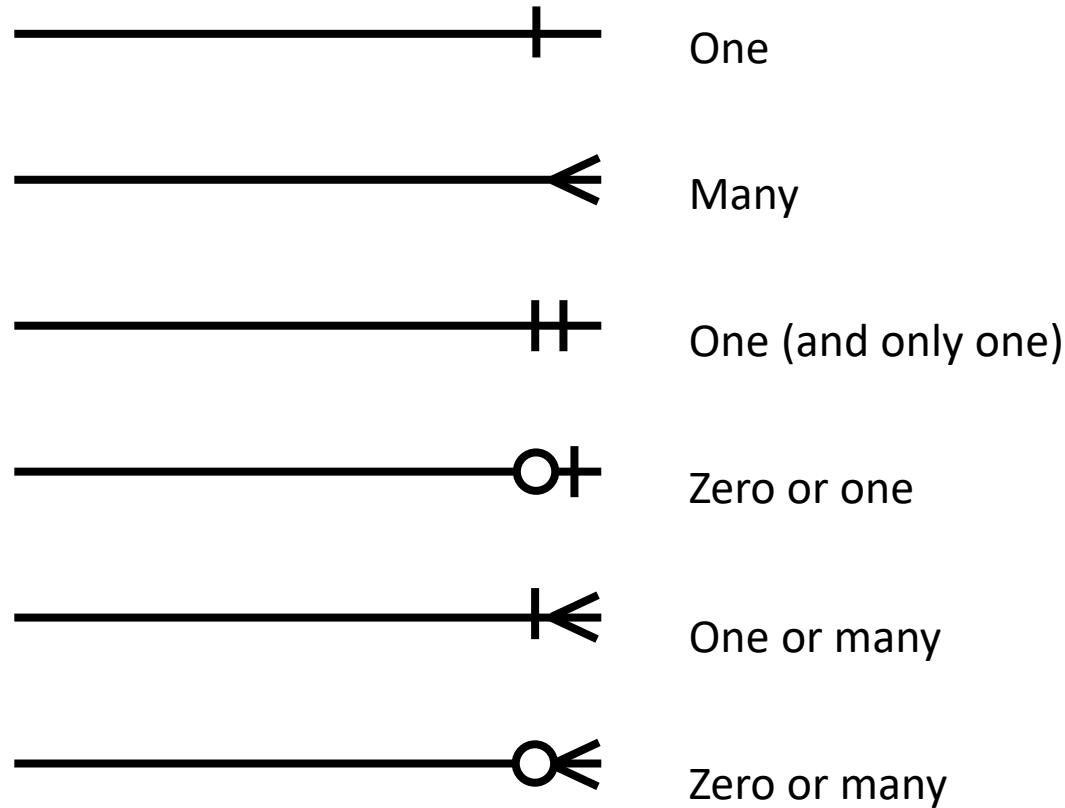
Sakila

Sample Database



Conceptual → Entities
 Logical → Entities, properties
 Physical → Entities, properties, types

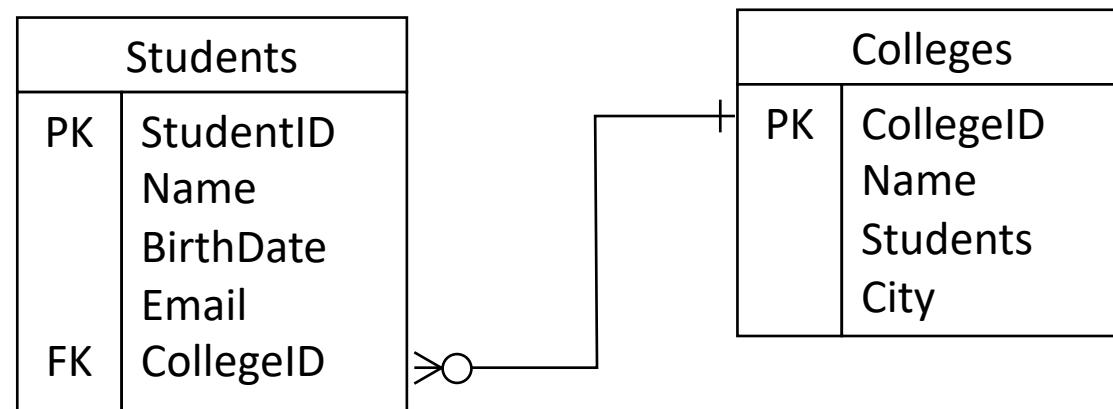
Cardinality – how one table relates to another



Entities, properties, PK, FK

Entity	
PK	Property Property Property Property Property
FK	Property

Example



Database Platform Installation

MySQL



Workbench



Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

SQL SQL i f o f o SQL Additions

Navigator

SCHEMAS Filter objects

- sakila
- sys
- world
 - Tables
 - city
 - country
 - countrylanguage
 - Views
 - Stored Procedures
 - Functions

Administration Schemas Object information

Table: country

Columns:

Code	char(3) PK
Name	char(52)
Continent	enum('Asia', 'Europe', 'North America', 'Africa', 'South America')
Region	char(26)
SurfaceArea	float(10,2)
IndepYear	smallint
Population	int
LifeExpectancy	float(3,1)
GNP	float(10,2)
GNPOld	float(10,2)
LocalName	char(45)

Object Info Session

country ×

Executes SQL statement

country

1 • SELECT * FROM country;

SQL query

SQL query output

Code	Name	Continent	Region
ABW	Aruba	North America	Caribbean
AFG	Afghanistan	Asia	Southern and Central Asia
AGO	Angola	Africa	Central Africa
AIA	Anguilla	North America	Caribbean
ALB	Albania	Europe	Southern Europe
AND	Andorra	Europe	Southern Europe
ANT	Netherlands Antilles	North America	Caribbean
ARE	United Arab Emirates	Asia	Middle East
ARG	Argentina	South America	South America
ARM	Armenia	Asia	Middle East

Result Grid Form Editor Field Types

country 1 ×

Action Output

Log

Output

1 11:25:26 SELECT * FROM world.country LIMIT 0, 1000

Message 239 row(s) returned

Duration / Fetch 0.000 sec / 0.000 sec

Hide/show panels

SELECT Syntax:

SELECT is used to retrieve rows selected from one or more tables, and can include UNION statements and subqueries. See [union](#), and [subqueries](#). A SELECT statement can start with a WITH clause to define common table expressions accessible within the SELECT. See [with](#).

The most commonly used clauses of SELECT statements are these:

- Each **select_expr** indicates a column that you want to retrieve. There must be at least one **select_expr**.
- table_references** indicates the table or tables from which to retrieve rows. Its syntax is described in [join](#).
- SELECT supports explicit partition

Context Help Snippets



Local instance MySQL80 ×

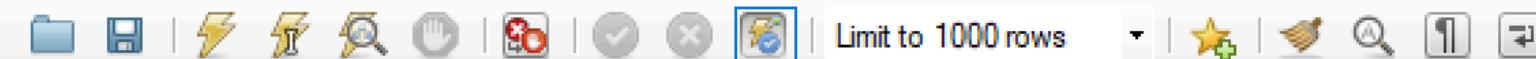
[File](#) [Edit](#) [View](#) [Query](#) [Database](#) [Server](#) [Tools](#) [Scripting](#) [Help](#)

Navigator

SCHEMAS Filter objects

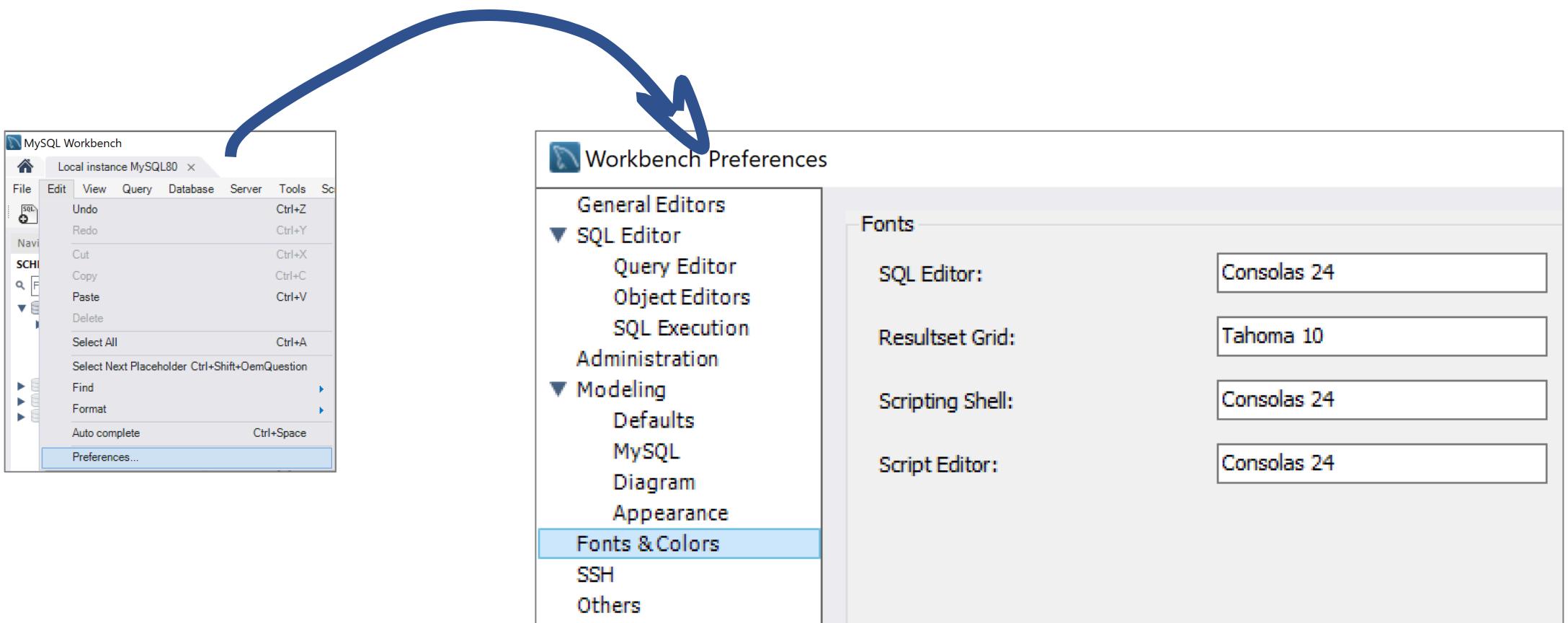
- ▶ [sakila](#)
- ▶ [sys](#)
- ▶ [world](#)

Query 1 ×

**1 • SHOW DATABASES**[Result Grid](#) | [Filter Rows:](#) [Export:](#) [Wrap Cell Content:](#)

	Database
▶	information_schema
▶	mysql
▶	performance_schema
▶	sakila
▶	sys
▶	world

Customize





DBeaver Community

Free Universal Database Tool



Star

19,388

Follow @dbeaver_news

search here ...

Go

[Home](#) [About](#) [Download](#) [Sources](#) [Documentation](#) [News](#) [Support](#) [Enterprise Edition](#) [CloudBeaver](#)

Universal Database Tool

Free multi-platform database tool for developers, database administrators, analysts and all people who need to work with databases. Supports all popular databases: MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, Teradata, Firebird, Apache Hive, Phoenix, Presto, etc.

#	Address	T1	City	T1	State	T1	Country	T1	Value	MetaData	
13	907 Bloos St	Buenos	DF	Brasil	73	Name	Label	#	Time	Table Name: Wan	
14	8210 111 St NW	Edmonton	AB	Canada	78	Na...	Quar...	0	INTEGER	Customer: 2,147,482	
15	790 W Pender Street	Vancouver	BC	Canada	79	Na...	Freight...	1	NVARCHAR	Customer: 2,147,482	
16	1800 Amphitheatre Parkway	Mountain View	CA	USA	94	Na...	LastNa...	2	NVARCHAR	Customer: 2,147,482	
17	1 Microsoft Way	Redmond	WA	USA	98	Co...	Compa...	3	NVARCHAR	Customer: 2,147,482	
18	627 Broadway	New York	NY	USA	10	Na...	Add...	Address	4	NVARCHAR	Customer: 2,147,482
19	7 Infinite Loop	Cupertino	CA	USA	95	Na...	City	5	NVARCHAR	Customer: 2,147,482	
20	541 Del Medio Avenue	Mountain View	CA	USA	94	Na...	State	6	NVARCHAR	Customer: 2,147,482	
21	801 W 4th Street	Reno	NV	USA	99	Na...	Country	7	NVARCHAR	Customer: 2,147,482	
22	120 S Orange Ave	Orlando	FL	USA	32	Na...	Prest...	8	NVARCHAR	Customer: 2,147,482	
23	Teuroniversale 8	Berlin	Germany	10	Na...	Phone	Phone	9	NVARCHAR	Customer: 2,147,482	
24	62 Salem Street	Boston	MA	USA	21	Na...	Fax	10	NVARCHAR	Customer: 2,147,482	
25	182 E Superior Street	Chicago	IL	USA	80	Na...	Email	11	NVARCHAR	Customer: 2,147,482	
26	319 N. Frances Street	Madison	WI	USA	53	Na...	Supp...	12	INTEGER	Customer: 2,147,482	
27	2211 W Glerry Street	Fort Worth	TX	USA	78						
28	1033 N Park Ave	Tucson	AZ	USA	85						
29	302 S 700 E	Salt Lake City	UT	USA	84						
30	790 Dundas Street West	Toronto	ON	Canada	46						
31	330 Elgin Street	Ottawa	ON	Canada	42						
32	194K Chain Lake Drive	Halifax	NS	Canada	83						
33	898 Osborne Street	Winnipeg	MB	Canada	82						
34	5112 48 Street	Yellowknife	NT	Canada	31						

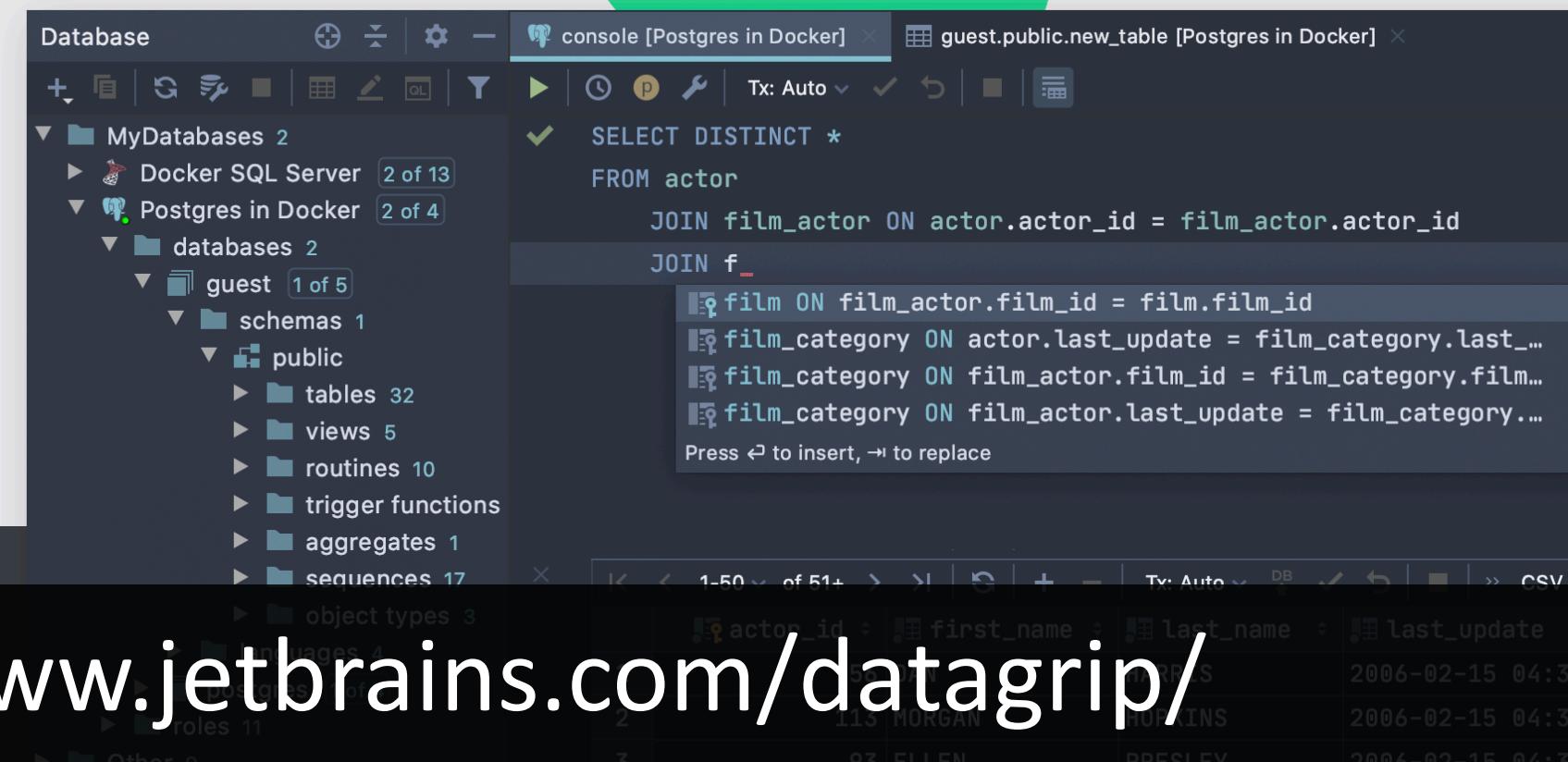


Download <https://dbeaver.io/>

ENJOY WORKING WITH DATABASES

Meet DataGrip, our new database IDE that is tailored to suit the specific needs of professional SQL developers.

 TAKE A TOUR



<https://www.jetbrains.com/datagrip/>

Allows you to execute queries in

SQL Statements

SQL division of tasks

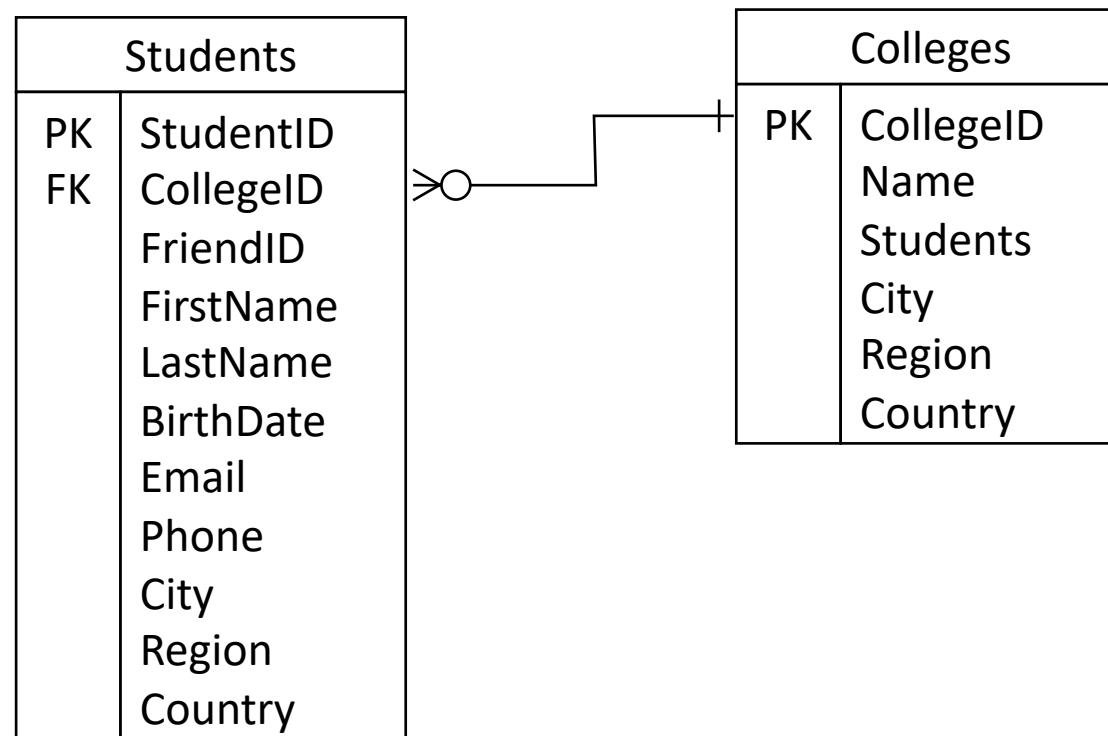
DQL, data query language, is used for querying data - includes like SELECT.

DDL, data definition language, is used to create and modify tables, views, users, other objects in the database. It affects the structure, but not the contents. There are three common commands: CREATE, ALTER, and DROP.

DCL, data control language, is used for access control.

DML, data manipulation language, is used to act on the data itself. The commands are INSERT, UPDATE, and DELETE.

Simple database



school_plus.sql

```

1 DROP DATABASE IF EXISTS `education`;
2 CREATE DATABASE IF NOT EXISTS `education`;
3 USE `education`;
4
5 SET NAMES UTF8MB4;
6 SET character_set_client = UTF8MB4;
7
8 -- -----
9 -- TABLE COLLEGES
10 --
11
12 CREATE TABLE `Colleges` (
13     `CollegeID` int NOT NULL AUTO_INCREMENT,
14     `Name` varchar (20) NOT NULL,
15     `Students` int NULL,
16     `City` varchar (15) NULL ,
17     `Region` varchar (15) NULL ,
18     `Country` varchar (15) NULL ,
19     PRIMARY KEY (`CollegeID`),
20     INDEX `CollegeID` (`CollegeID` ASC),
21     INDEX `Name` (`Name` ASC)
22 ) ENGINE=InnoDB DEFAULT CHARSET=UTF8MB4 COLLATE=utf8mb4_0900_ai_ci;
23
24 -- -----
25 -- TABLE STUDENTS
26
27
28 CREATE TABLE `Students` (
29     `StudentID` int NOT NULL AUTO_INCREMENT,
30     `CollegeID` int NOT NULL,
31     `FriendID` int NULL,
32     `FirstName` varchar (20) NOT NULL ,
33     `LastName` varchar (20) NOT NULL ,
34     `BirthDate` date NULL ,
35     `Email` varchar (30) NULL ,
36     `Phone` varchar (24) NULL ,
37     `City` varchar (15) NULL ,
38     `Region` varchar (15) NULL ,
39     `Country` varchar (15) NULL ,
40     PRIMARY KEY (`StudentID`),
41     INDEX `StudentID` (`StudentID` ASC),
42     INDEX `LastName` (`LastName` ASC),
43     INDEX `FirstName` (`FirstName` ASC),
44     FOREIGN KEY (`CollegeID`) REFERENCES `Colleges` (`CollegeID`)
45         ON DELETE NO ACTION
46         ON UPDATE NO ACTION
47 ) ENGINE=InnoDB DEFAULT CHARSET=UTF8MB4 COLLATE=utf8mb4_0900_ai_ci;

```

school_plus.sql

```

49
50     -- POPULATE COLLEGES
51
52
53     INSERT INTO `Colleges` VALUES(1,'MIT',11,'Cambridge','MA','USA');
54     INSERT INTO `Colleges` VALUES(2,'Brown',9,'Providence','RI','USA');
55     INSERT INTO `Colleges` VALUES(3,'Dartmouth',6,'Hanover','NH','USA');
56     INSERT INTO `Colleges` VALUES(4,'Stanford',17,'Stanford','CA','USA');
57     INSERT INTO `Colleges` VALUES(5,'Yale',12,'New Haven','CT','USA');
58     INSERT INTO `Colleges` VALUES(6,'Columbia',31,'New York','NY','USA');
59     INSERT INTO `Colleges` VALUES(7,'Harvard',23,'Cambridge','MA','USA');
60     INSERT INTO `Colleges` VALUES(8,'Princeton',9,'Princeton','NJ','USA');
61     INSERT INTO `Colleges` VALUES(9,'Johns Hopkins',24,'Baltimore','MD','USA');
62     INSERT INTO `Colleges` VALUES(10,'Northwestern',21,'Evanston','IL','USA');
63
64     INSERT INTO `Colleges` VALUES(11,'Duke',15,'Durham','NC','USA');
65     INSERT INTO `Colleges` VALUES(12,'Cornell',22,'Ithaca','NY','USA');
66     INSERT INTO `Colleges` VALUES(13,'Notre Dame',9,'Notre Dame','IN','USA');
67     INSERT INTO `Colleges` VALUES(14,'UCLA',32,'Los Angeles','CA','USA');
68     INSERT INTO `Colleges` VALUES(15,'Berkeley',42,'Berkeley','CA','USA');
69     INSERT INTO `Colleges` VALUES(16,'Georgetown',5,'Washington','DC','USA');
70     INSERT INTO `Colleges` VALUES(17,'Michigan',45,'Ann Arbor','MI','USA');
71     INSERT INTO `Colleges` VALUES(18,'USC',44,'Los Angeles','CA','USA');
72     INSERT INTO `Colleges` VALUES(19,'Tufts',11,'Medford','MA','USA');
73     INSERT INTO `Colleges` VALUES(20,'NYU',51,'New York','NY','USA');
74
75
76     -- POPULATE STUDENTS
77
78
79
80     INSERT INTO `Students` VALUES(1,1,10,'Nancy','Davolio','1948-12-08','nancy@gmail.com'
81         , '(360) 234-8488','Seattle','WA','USA');
82     INSERT INTO `Students` VALUES(2,9,5,'Andrew','Fuller','1952-02-19','andrew@yahoo.com'
83         ,NULL,'Dallas','TX','USA');
84     INSERT INTO `Students` VALUES(3,8,1,'Janet','Leverling','1963-08-30',''
85         ,janet@hotmail.com', '(786) 634-4522','Miami','FL','USA');
86     INSERT INTO `Students` VALUES(4,3,9,'Margaret','Peacock','1937-09-19',''
87         ,maggie@outlook.com',NULL,'Phoenix','AZ','USA');
88     INSERT INTO `Students` VALUES(5,4,2,'Steven','Buchanan','1955-03-04','steve@apple.com'
89         ,NULL,'Denver','CO','USA');
90     INSERT INTO `Students` VALUES(6,7,8,'Michael','Suyama','1963-07-02','mike@icloud.com'
91         , '(541) 544-7733','Portland','OR','USA');
92     INSERT INTO `Students` VALUES(7,6,3,'Robert','King','1960-05-29','rob@gmail.com',NULL
93         , 'San Francisco','CA','USA');
94     INSERT INTO `Students` VALUES(8,5,7,'Laura','Callahan','1958-01-09','laura@gmail.com'
95         , '(901) 425-8913','Memphis','TN','USA');
96     INSERT INTO `Students` VALUES(9,2,4,'Anne','Dodsworth','1966-01-27','anne@msn.com')
97 
```

Download installation script

<http://bit.ly/3tiLF3H>

SQL Statements

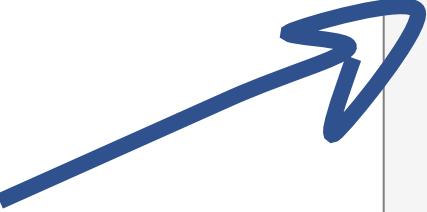
Clauses

CLAUSE	PURPOSE
SELECT	Selects which columns to include
FROM	The tables from which to retrieve data
WHERE	Filters out unwanted data
GROUP BY	Groups rows together
HAVING	Filters groups
ORDER BY	Sorts rows

SELECT

<https://dev.mysql.com/doc/>

This block
is a career



```
SELECT
  [ALL | DISTINCT | DISTINCTROW ]
  [HIGH_PRIORITY]
  [STRAIGHT_JOIN]
  [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
  [SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
  select_expr [, select_expr] ...
  [into_option]
  [FROM table_references
    [PARTITION partition_list]]
  [WHERE where_condition]
  [GROUP BY {col_name | expr | position}, ... [WITH ROLLUP]]
  [HAVING where_condition]
  [WINDOW window_name AS (window_spec)
    [, window_name AS (window_spec)] ...]
  [ORDER BY {col_name | expr | position}
    [ASC | DESC], ... [WITH ROLLUP]]
  [LIMIT {[offset,] row_count | row_count OFFSET offset}]
  [into_option]
  [FOR {UPDATE | SHARE}
    [OF tbl_name [, tbl_name] ...]
    [NOWAIT | SKIP LOCKED]
    | LOCK IN SHARE MODE]
  [into_option]

  into_option: {
    INTO OUTFILE 'file_name'
      [CHARACTER SET charset_name]
      export_options
    | INTO DUMPFILE 'file_name'
    | INTO var_name [, var_name] ...
  }
```

Keep clauses in syntactical order

*Order of
Clauses
Matters*

SELECT

```
select_expr [, select_expr] ...
[FROM table_references]
[WHERE where_condition]
[GROUP BY {col_name | expr | position}]
[ORDER BY {col_name | expr | position}[ASC | DESC]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Statements

Several clauses make up a select statement.

You will often use two or three clauses.

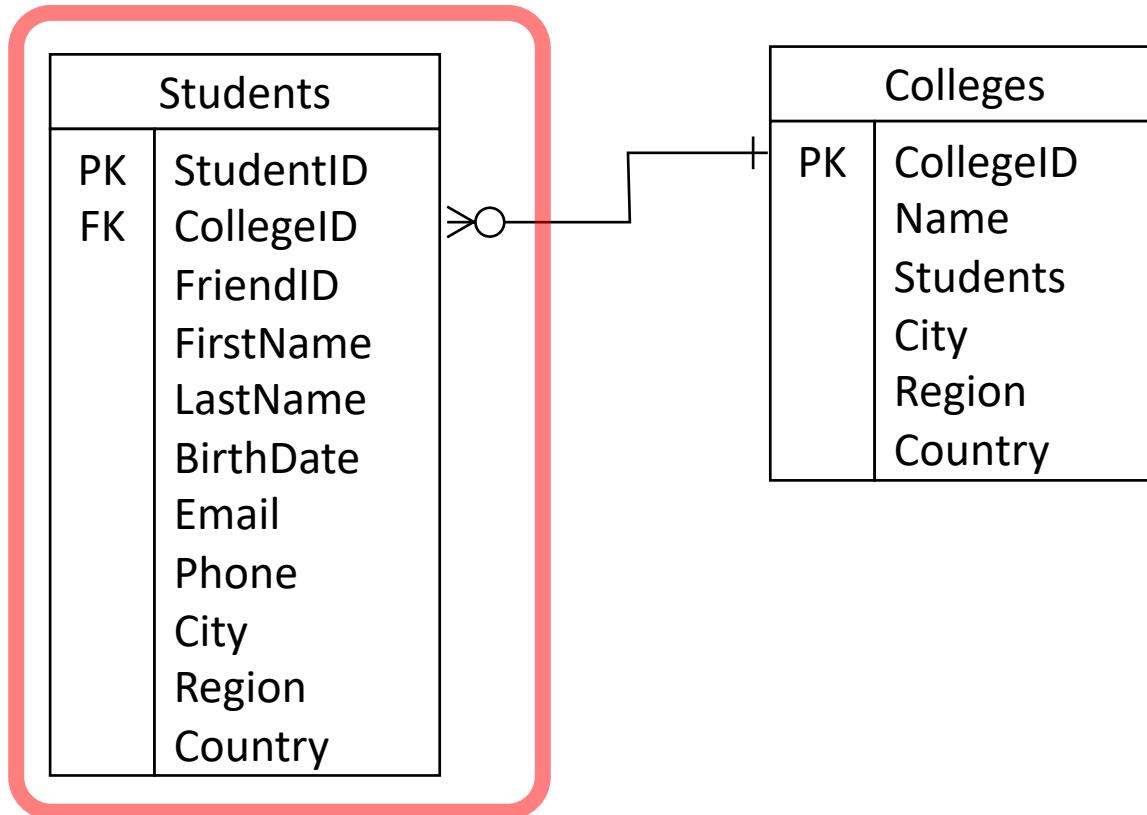
Order of clauses matters.

Only the select clause is mandatory.

QUERY

SQL statement

Simple database



Simple select

SELECT *columns*
FROM *table*

Select

SELECT *columns*

FROM *table*

WHERE *condition*

ORDER BY *columns*

SELECT

Retrieving records

Case sensitive – some times

- SQL Keywords – case insensitive but usually written in caps
- Tables and columns are case sensitive depending on platform/OS
- Example, MySQL case sensitive on Linux, insensitive on Windows

Select basics

SELECT *list_of_columns*

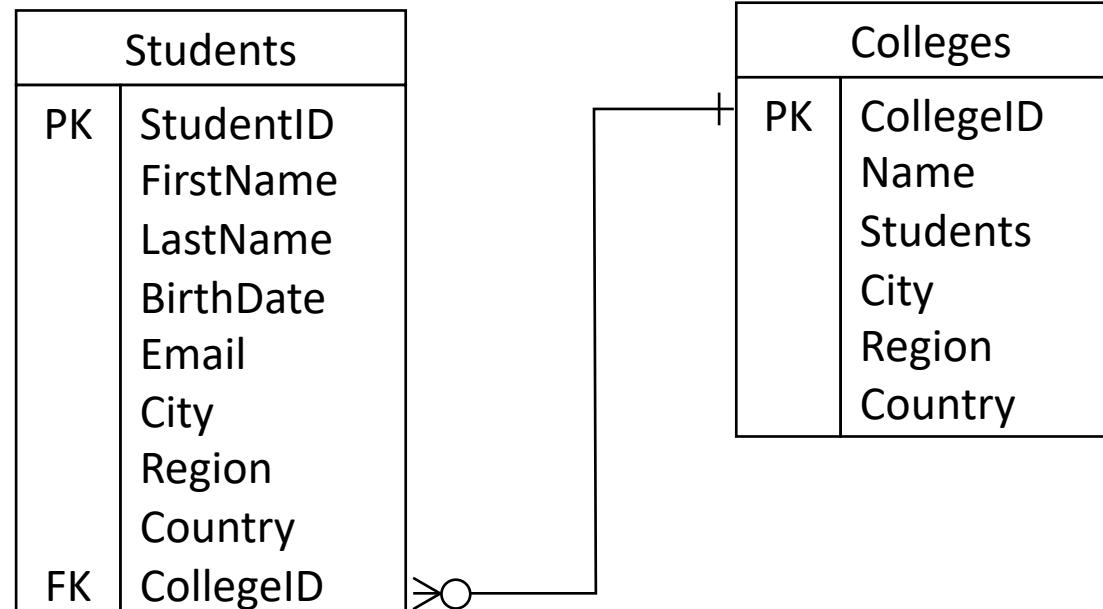
FROM *table[s]*

[**WHERE** *search_conditions*]

SELECT *

FROM Colleges

WHERE City='Cambridge'



Specify columns

```
SELECT FirstName, LastName  
FROM Students
```

Renaming columns and naming expressions

-- rename column

```
SELECT Name AS University  
FROM Colleges
```

-- expression plus rename

```
SELECT Name AS University, Students*1000 AS 'number of students'  
FROM Colleges
```

Pretty name for column

```
SELECT FirstName,  
        LastName,  
        CONCAT(FirstName, " ", LastName) AS FullName  
FROM students
```

A *select_expr* can be given an alias using *AS alias_name*. The alias is used as the expression's column name and can be used in GROUP BY, ORDER BY, or HAVING clauses.

Table aliases

```
SELECT c.Name, c.Students
```

```
FROM Colleges c
```

The screenshot shows a MySQL Workbench query editor window. The query text area contains two numbered lines:

1. `SELECT c.Name, c.Students`
2. `FROM Colleges c`

Below the query text is a toolbar with various icons for database management. The result grid is titled "Result Grid" and displays the following data:

	Name	Students
▶	MIT	11
	Brown	9
	Dartmouth	6
	Stanford	17

Active learning: write the query for the following output

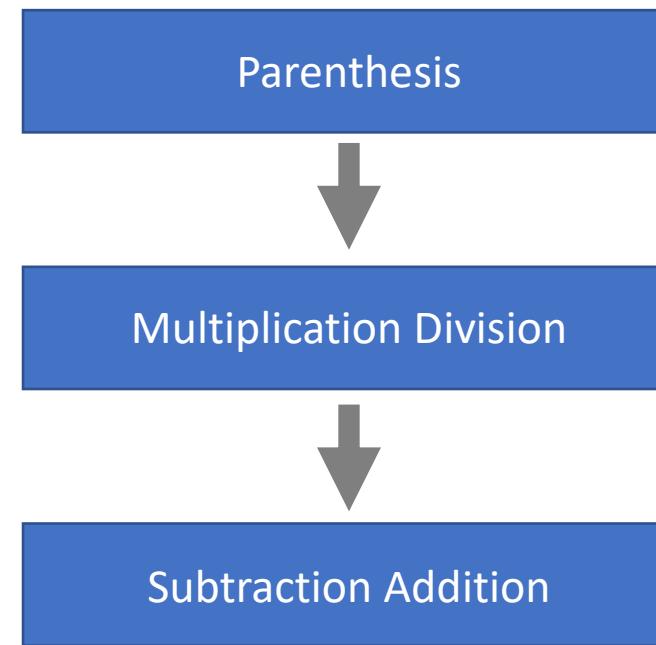
	FirstName	LastName	Birthdate	Age
▶	Nancy	Davolio	1948-12-08	72
	Andrew	Fuller	1952-02-19	69
	Janet	Leverling	1963-08-30	57
	Margaret	Peacock	1937-09-19	83
	Steven	Buchanan	1955-03-04	66
	Michael	Suyama	1963-07-02	57
	Robert	King	1960-05-29	60
	Laura	Callahan	1958-01-09	63
	Anne	Dodsworth	1966-01-27	55
	Ivy	Johnson	1986-01-20	35
	Ana	Trujillo	1998-10-08	22
	Thomas	Hardy	1992-12-09	28
	Antonio	Moreno	1993-03-23	27
	Elizabeth	Brown	1997-01-11	24
	Ann	Devon	1995-04-24	25
	Ariel	Cruz	1993-02-12	28
	Giovanni	Rovelli	1990-09-19	30
	Marie	Bertrand	1998-09-29	22
	Philip	Cramer	1996-07-17	24
	Michael	Holz	1996-02-25	25

You can use `TIMESTAMPDIFF(unit, datetime_expr1, datetime_expr2)`

Distinct

**SELECT DISTINCT Region
FROM Colleges**

Arithmetic Operator Precedence



Active learning: change precedence to match output

1 • SELECT Name, Students * 1000 - 100 As modified
2 FROM Colleges

Name	modified
MIT	10900
Brown	8900
Dartmouth	5900
Stanford	16900

1 • SELECT Name, **Change Precedence** As modified
2 FROM Colleges

Name	modified
MIT	9900
Brown	8100
Dartmouth	5400
Stanford	15300

Active learning

Return:

College name

Student population * 1000

Projected growth - 20% increase

	Name	Population	ProjectedGrowth
▶	MIT	11000	13200.0
	Brown	9000	10800.0
	Dartmouth	6000	7200.0
	Stanford	17000	20400.0
	Yale	12000	14400.0
	Columbia	31000	37200.0
	Harvard	23000	27600.0
	Princeton	9000	10800.0
	Johns Hopkins	24000	28800.0
	Northwestern	21000	25200.0
	Duke	15000	18000.0
	Cornell	22000	26400.0
	Notre Dame	9000	10800.0
	UCLA	32000	38400.0
	Berkeley	42000	50400.0
	Georgetown	5000	6000.0
	Michigan	45000	54000.0
	USC	44000	52800.0
	Tufts	11000	13200.0
	NYU	51000	61200.0

WHERE

The where clause

The where clause specifies the search conditions

SELECT *columns_list*

FROM *table_list*

WHERE *search_conditions*

Where region is equal to

SELECT *

FROM *Students*

WHERE *Region* = 'TX'

Search condition categories

- Comparison operators – e.g. =,<,>)
- Logical operators – e.g. AND,OR, NOT
- Ranges – between and not between
- Lists – IN, NOT IN
- Unknown values – IS NULL, IS NOT NULL
- Character matches – LIKE, NOT LIKE

Comparison operators

WHERE *expression comparison_operator expression*

Name	Description
>	Greater than operator
>=	Greater than or equal operator
<	Less than operator
<>, !=	Not equal operator
<=	Less than or equal operator
<=>	NULL-safe equal to operator
=	Equal operator

Where region is not equal to ...

```
SELECT *
FROM Students
WHERE Region <> 'TX'
```

* Not case sensitive

Active learning

Find students born after January 1st, 1990

	StudentID	CollegeID	FriendID	FirstName	LastName	BirthDate	Email	Phone	City	Region	Country
▶	11	1	NULL	Ana	Trujillo	1998-10-08	ana@gmail.com	(360) 457-2258	Seattle	WA	USA
	12	9	NULL	Thomas	Hardy	1992-12-09	tom@yahoo.com	NULL	Dallas	TX	USA
	13	5	NULL	Antonio	Moreno	1993-03-23	tony@hotmail.com	NULL	Miami	FL	USA
	14	7	NULL	Elizabeth	Brown	1997-01-11	beth@outlook.com	(480) 324-2178	Phoenix	AZ	USA
	15	3	NULL	Ann	Devon	1995-04-24	ann@apple.com	NULL	Denver	CO	USA
	16	2	NULL	Ariel	Cruz	1993-02-12	ariel@icloud.com	(541) 652-4565	Portland	OR	USA
	17	6	NULL	Giovanni	Rovelli	1990-09-19	gio@gmail.com	(415) 665-2255	San Francisco	CA	USA
	18	10	NULL	Marie	Bertrand	1998-09-29	marie@gmail.com	NULL	Memphis	TN	USA
	19	4	NULL	Philip	Cramer	1996-07-17	phil@msn.com	(207) 4436-6524	Portland	ME	USA
	20	8	NULL	Michael	Holz	1996-02-25	michael@gmail.com	NULL	Chicago	IL	USA

Where birth date is greater than

SELECT *

FROM Students

WHERE BirthDate > '1996-01-01'

* Date format 'YYYY-MM-DD'

Comparison operators

Name	Description
BETWEEN ... AND ...	Value is within a range
COALESCE()	Return the first non-NULL argument
GREATEST()	Return the largest argument
IN()	Value is within a set of values
INTERVAL()	Index of the argument that is less
IS	Test a value against a boolean
IS NOT	Test a value against a boolean
IS NOT NULL	NOT NULL value test
IS NULL	NULL value test
ISNULL()	Test whether the argument is NULL
LEAST()	Return the smallest argument
LIKE	Simple pattern matching
NOT BETWEEN ... AND ...	Value is not within a range of values
NOT IN()	Value is not within a set of values
NOT LIKE	Negation of simple pattern matching
STRCMP()	Compare two strings

Conditions are composed of expressions and operators

Boolean expressions	fetch the data based on matching a single value
<pre>SELECT column FROM table_name WHERE single_value_matching_expression</pre>	
Example: <pre>SELECT * FROM Colleges WHERE Region = 'MA'</pre>	

Numeric expressions	Perform mathematical operation in a query
<pre>SELECT numerical_expression as operation_name FROM table_name</pre>	
Example: <pre>SELECT (4 + 3) AS Addition</pre>	

Date expressions	Results in datetime value
Example: <pre>SELECT now()</pre>	

LOGICAL OPERATORS

AND, NOT, OR

Logical operators

Name	Description
AND, &&	Logical AND
NOT, !	Negates value
OR,	Logical OR
XOR	Logical XOR

AND

SELECT *

FROM Students

WHERE

BirthDate > '1990-01-01' AND Region = 'TX'

OR

SELECT *

FROM Students

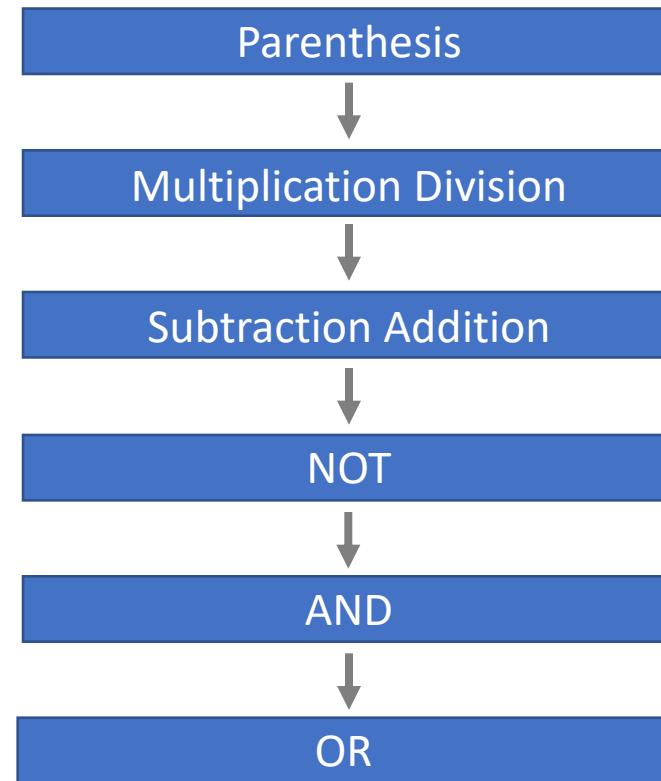
WHERE

BirthDate > '1990-01-01' OR Region = 'TX'

NOT

```
SELECT *
FROM Students
WHERE
NOT (BirthDate > '1990-01-01')
```

Logical Operator Precedence



Active learning

Find students born after 1950, from Texas, and Austin

IN

expr IN (value, ...)

OR

SELECT *

FROM Students

WHERE

Region = 'AZ' **OR**

Region = 'TX' **OR**

Region = 'FL'

IN

SELECT *

FROM *Students*

WHERE *Region* **IN** ('AZ', 'TX', 'FL')

NOT IN

SELECT *

FROM *students*

WHERE *region* **NOT IN** ('Z', 'TX', 'FL')

Active learning

Students from Austin, Boston, and Chicago

BETWEEN

expr BETWEEN *min* AND *max*

Where birth date is between ... and ...

```
SELECT *
FROM Students
WHERE
    StudentID >= 1 AND
    StudentID <= 5
```

Where StudentID is between ... and ...

```
SELECT *
FROM Students
WHERE StudentID
      BETWEEN 1 AND 5
```

Active learning

Find students born 01/01/1990 to 01/01/2000

NULL

WHERE column IS NULL

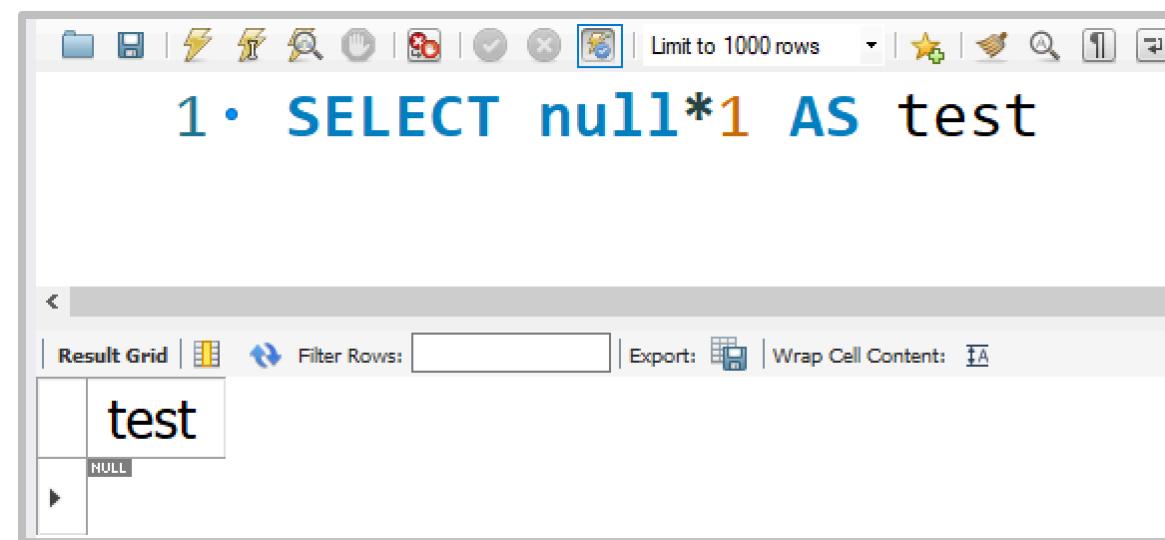
NULL is a placeholder for unknown information. It's not zero or blank

```
SELECT *
FROM Students
WHERE Country IS NULL
```

* IS NOT NULL

Arithmetic operations on *null* are *null*

```
SELECT null*1 AS test
```



The screenshot shows the MySQL Workbench interface with a query results window. The title bar of the window displays the query: "1 • SELECT null*1 AS test". The window contains a single row in a grid, with the column header "test" and the value "NULL" below it. The MySQL Workbench toolbar at the top includes various icons for file operations, search, and connection management.

test
NULL

Active learning

Find students with a phone