11.

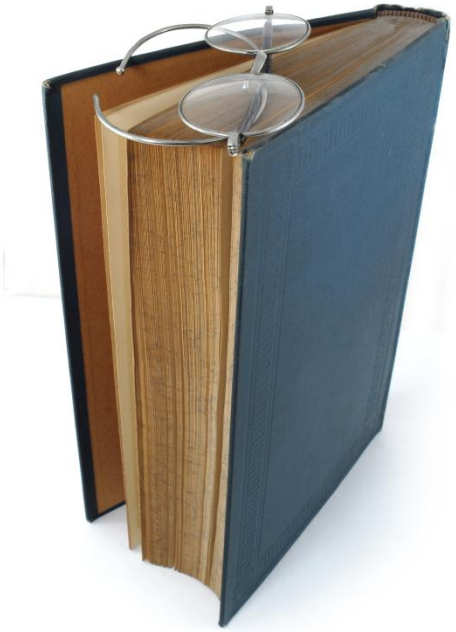
Agenda

- What is a Database, DBMS and RDBMS?
- What is SQL?Why I need to know SQL?
- SQL queries (+ practice)

Agenda

- **What is a Database, DBMS and RDBMS?**
- What is SQL? Why I need to know SQL?
- SQL queries (+ practice)

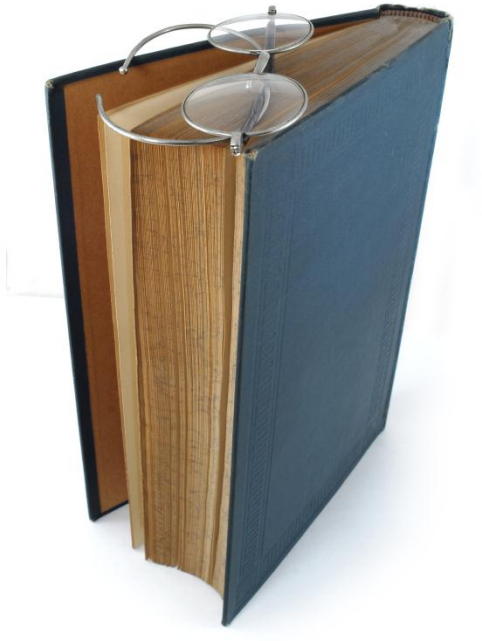
What is a Database (DB)?



A **database** is a collection of information that is organized so that it can be easily accessed, managed and updated.

Databases are managed using a **database management system** (**DBMS**).

What is DBMS?



Database Management System (DBMS) is a collection of programs which enables its users to access database, manipulate data, enables reporting / representation of data .

It also helps to control access to the database.

Database types

Relational DBs

=> SQL DBs

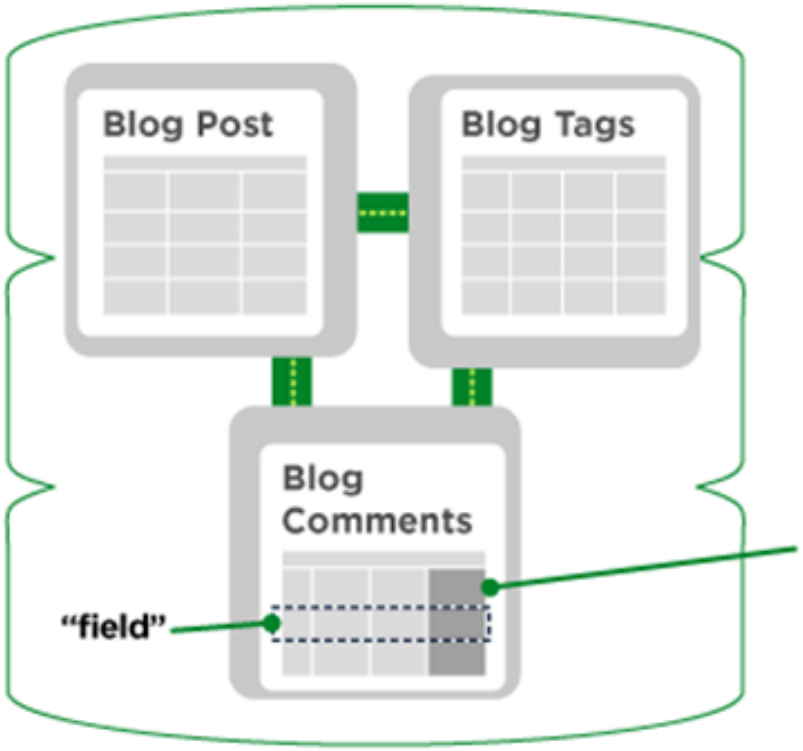
- Microsoft SQL Server
- Oracle Database
- MySQL
- IBM DB2
- …

Non-relational DBs

=> NoSQL DBs:

- MongoDB
- DocumentDB
- Cassandra
- Coachbase
- Hbase
- Redis,
- Neo4j
- …

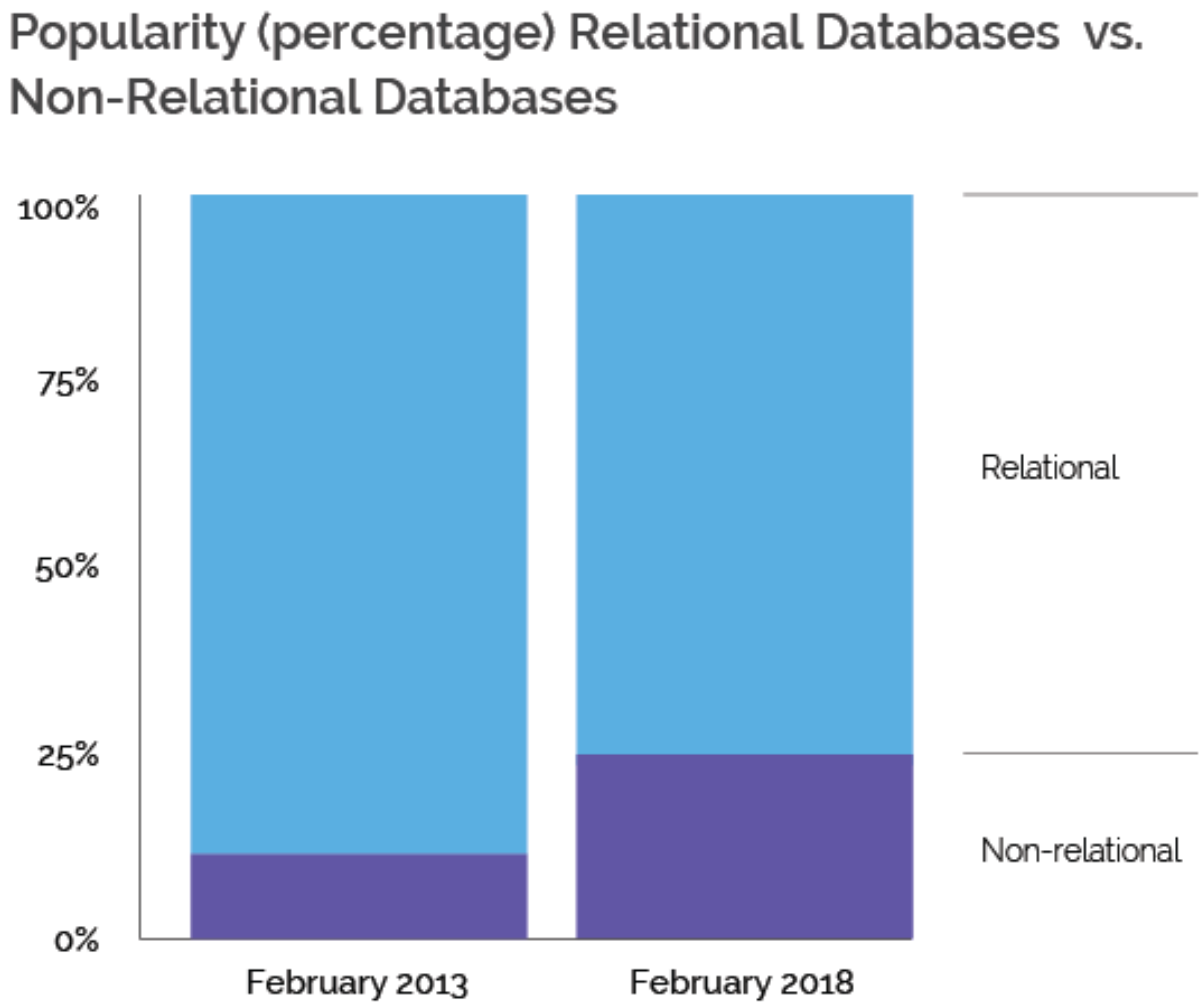
Relational vs Non-relational DBs



A non-relational database does not incorporate the table model. Instead, data can be stored in a single document file.

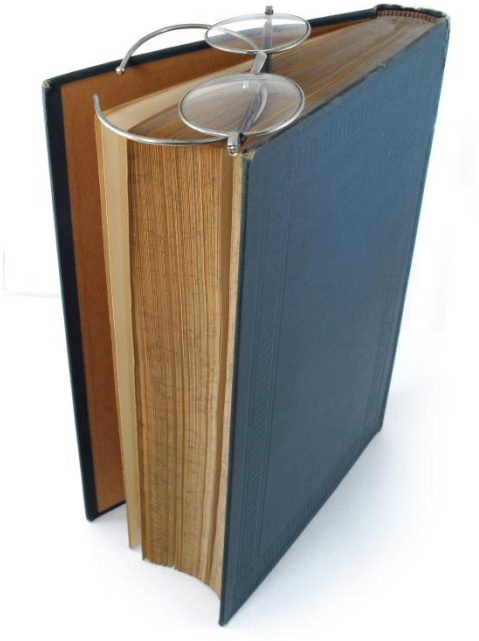
A relational database table organizes structured data fields into defined columns.

Statistics



Source: https://db-engines.com/en/ranking_trend

Relational database



Relational databases are made up of a set of tables with data that fits into a predefined category. Each table has at least one data category in a column, and each row has a certain data instance for the categories which are defined in the columns.

They uses a structure that allows us to identify and access data in relation to another piece of data in the database.

Table structure - content

Customers

Column

Row

ID	First Name	Name	Street Address	City	State
1	Tracey	am	7 East Walker Dr.	Raleigh	NC
2	Lucinda	George	789 Brewer St.	Cary	NC
3	Jerrod	Smith	211 St. George Ave.	Raleigh	NC
4	Brett	Newkirk	47 Hillsborough St.	Raleigh	NC
5	Chloe	Jones	23 Solo Ln.	Raleigh	NC
6	Quinton	Boyd	4 Cypress Cr.	Durham	NC
7	Alex	Hinton	1011 Hodge Ln.	Cary	NC
8	Nisha	Hall	123 Huntington St.	Raleigh	NC
9	Hillary	Clayton	2516 Newman	Raleigh	NC
10	Kiara	Williams	9014 Miller Ln.	Durham	NC
11	Katy	Jones	456 Denver Rd.	Cary	NC
12	Beatrix	Joslin	85 North West St.	Raleigh	NC
13	Mariah	Allen	12 Jupe	Raleigh	NC
14	Jennifer	Hill	2100 Field Ave.	Raleigh	NC
15	Jaleel	Smith	123 Hill Top Drive	Garner	NC

Table structure – primary key (PK)

Primary key (PK) is a column or a group of columns in a table that uniquely identifies the rows in that table.

CUSTOMERS

CustomerNo	FirstName	LastName
1	Sally	Thompson
2	Sally	Henderson
3	Harry	Henderson
4	Sandra	Wellington

Table structure – foreign key (FK)

Primary key in the ORDERS table

ORDERS

OrderNo	*EmployeeNo*	*CustomerNo*	Supplier	Price	Item
1	*1*	*42*	Harrison	$235	Desk
2	*4*	*1*	Ford	$234	Chair
3	*1*	*68*	Harrison	$415	Table
4	*2*	*112*	Ford	$350	Lamp
5	*3*	*42*	Ford	$234	Chair
6	*2*	*112*	Ford	$350	Lamp
7	*2*	*42*	Harrison	$235	Desk

Primary key in the CUSTOMERS table

CUSTOMERS

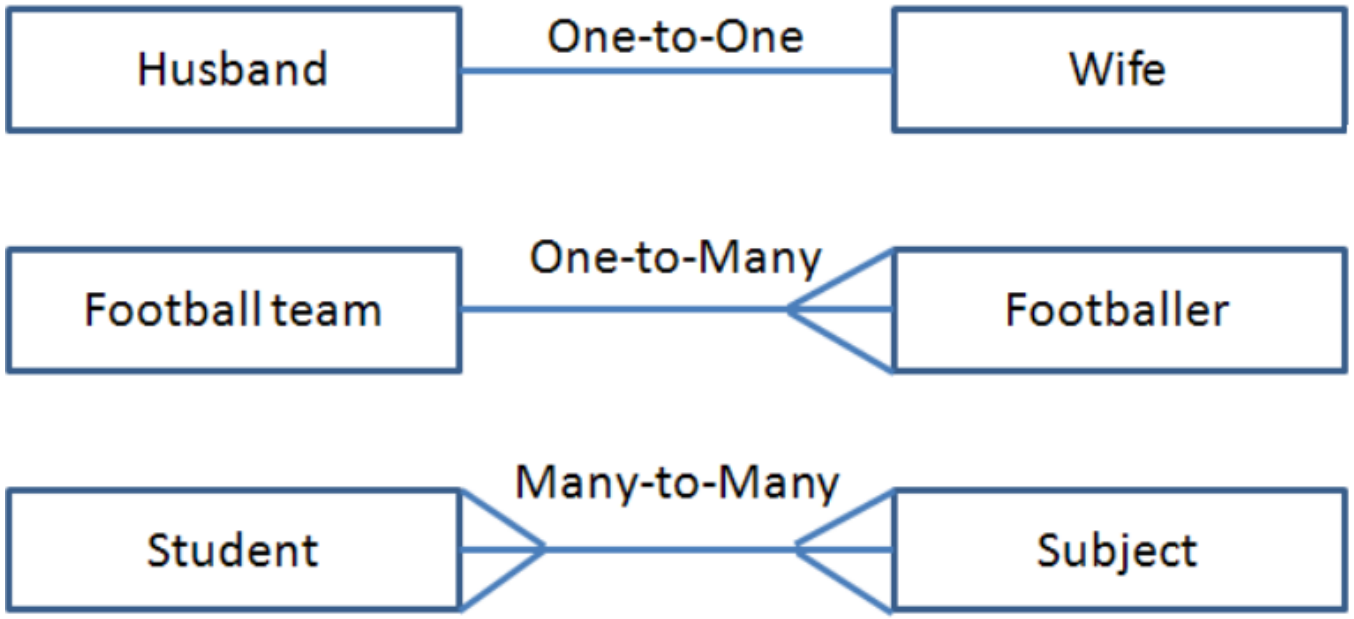
CustomerNo	FirstName	LastName
1	Sally	Thompson
2	Sally	Henderson
3	Harry	Henderson
4	Sandra	Wellington

Foreign key (FK) is a column or a group of columns that point to primary key columns in other database tables

Foreign key in the ORDERS table

Table structure – relationships

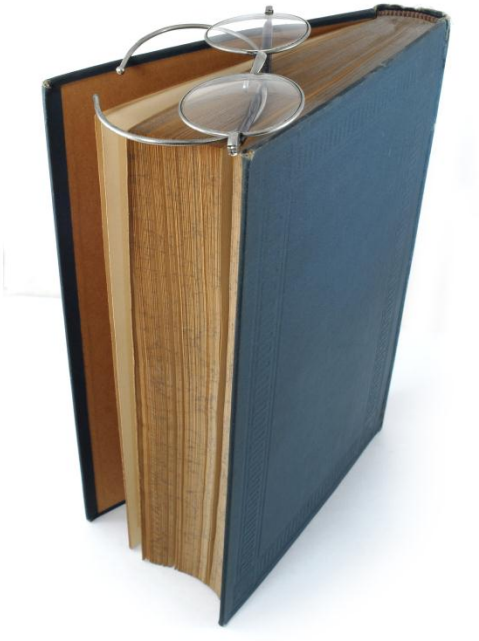
One-to-one relationships - each entry in the first table has one, and only one, counterpart in the second table.



Many-to-many relationships - each record in Table A corresponds to one or more records in Table B, and each record in Table B corresponds to one or more records in Table A.

One-to-many relationships - each record in Table A corresponds to one or more records in Table B, but each record in Table B corresponds to only one record in Table A.

What is RDBMS?

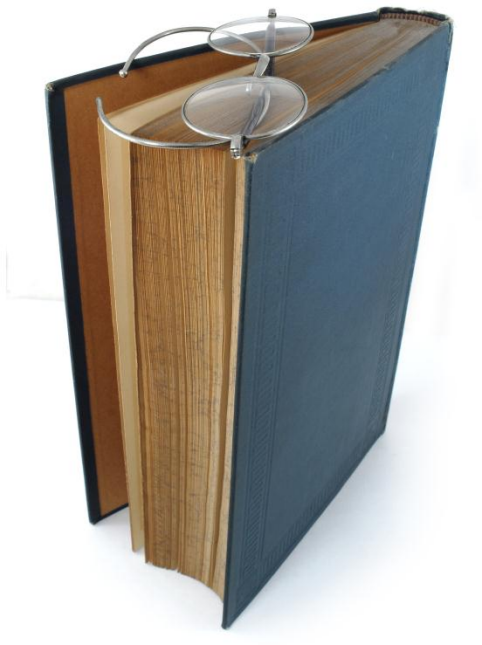


A **relational database management system (RDBMS)** is a program that allows you to create, update, and administer a **relational database**. Relational database management systems use the **SQL language** to access the database.

Agenda

- What is a Database, DBMS and RDBMS?
- **What is SQL? Why I need to know SQL?**
- SQL queries (+ practice)

What is SQL?



SQL (**S**tructured **Q**uery **L**anguage) is a programming language used to communicate with data stored in a relational database management system.

SQL syntax is similar to the English language, which makes it relatively easy to write, read, and interpret.

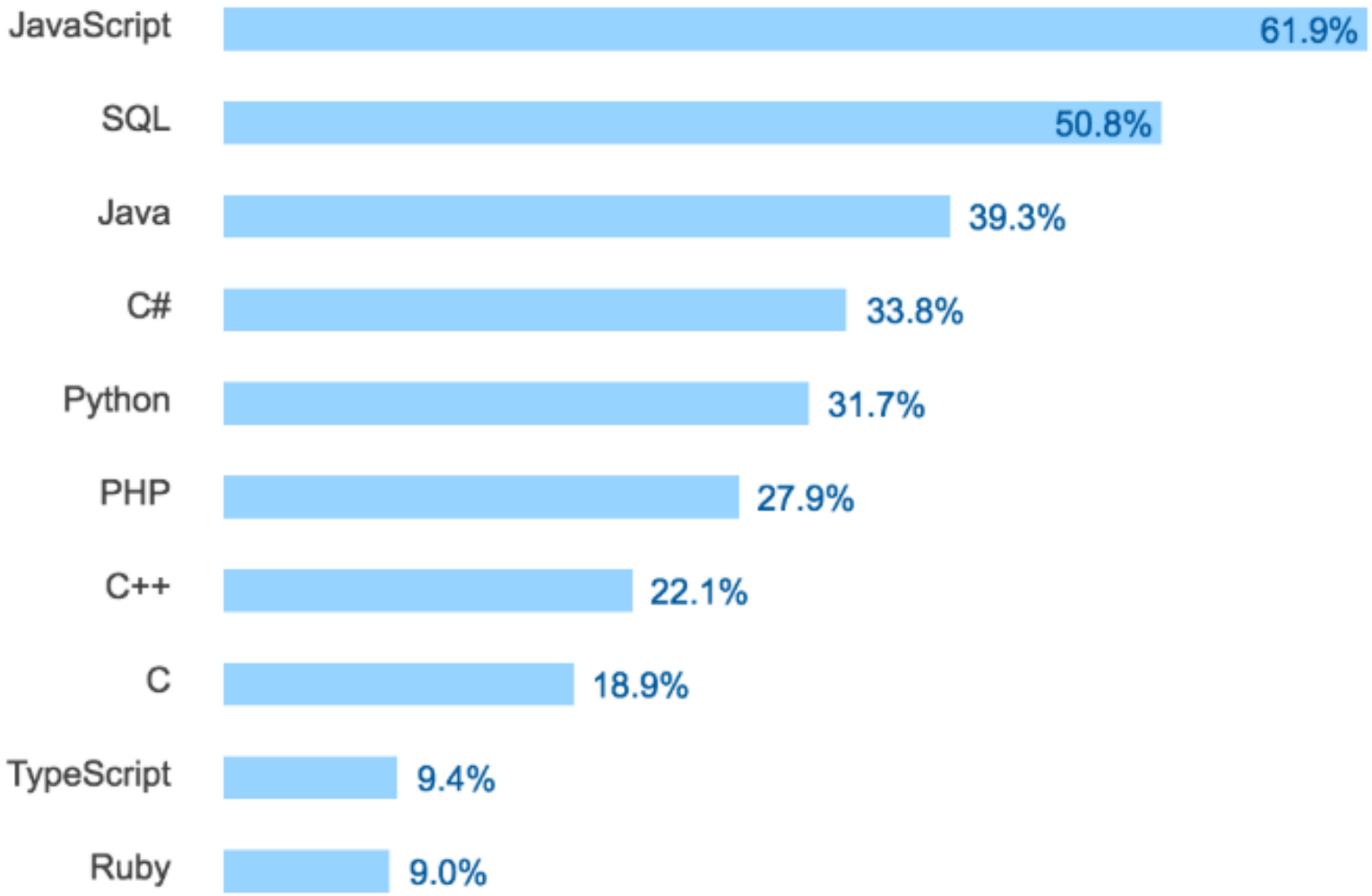
What can SQL do?

- execute queries against a database
- retrieve data from a database
- insert records in a database
- update records in a database
- delete records from a database
- create new databases
- create new tables in a database
- drop tables
- …

Why do I need to know SQL?

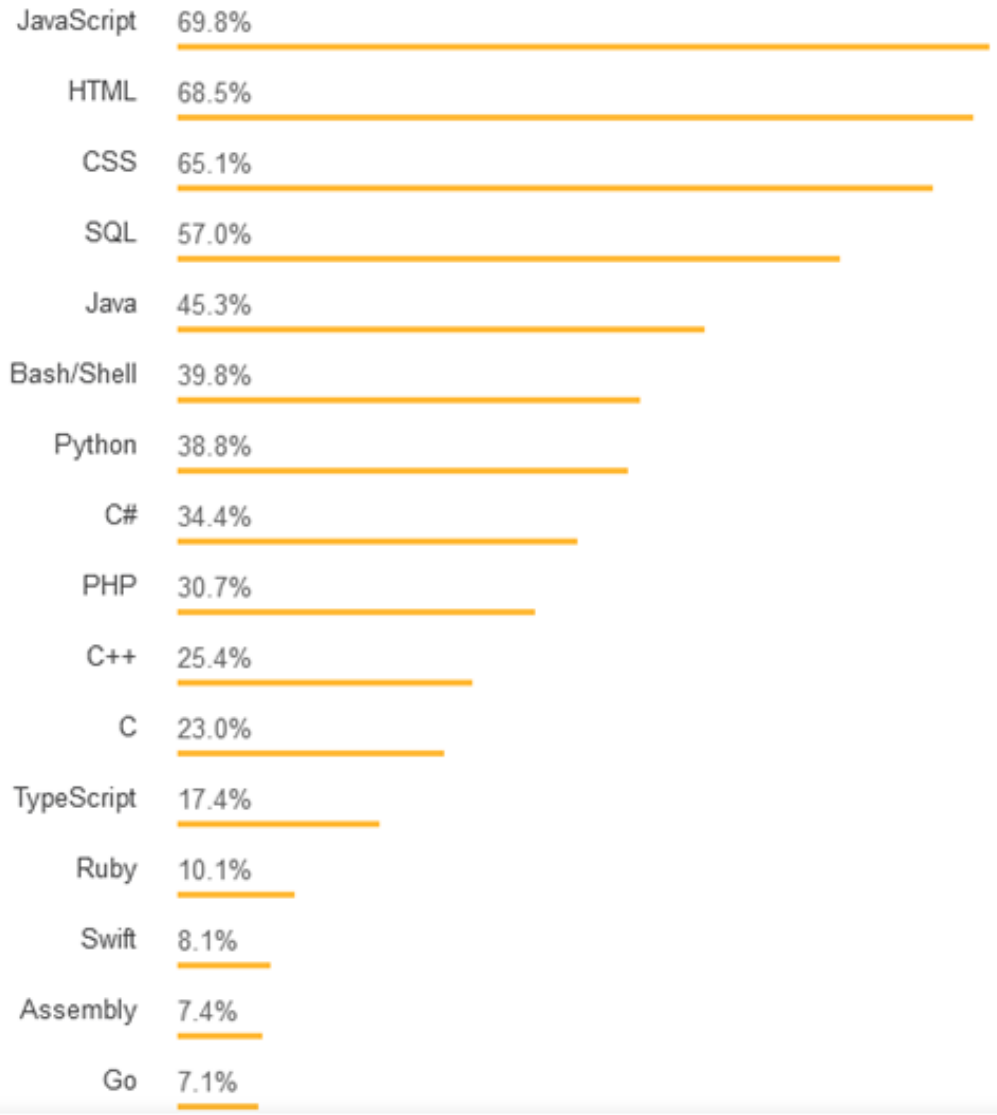
- To work with any kind of data
- To check what data is written in database
- To update or to write queries for test scripts
- Basic skill required in IT

SQL popularity in 2017



Some of the most common programming languages, Stack Overflow 2017

SQL popularity in 2018



Agenda

- What is a Database, DBMS and RDBMS?
- What is SQL? Why I need to know SQL?
- **SQL queries (+ practice)**

Let’s open the link…

https://sqlbolt.com/lesson/select_queries_introduction

SELECT queries

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103

```
SELECT * FROM movies;
```

SELECT *
FROM *table_name*;

SELECT *column1, column2, ...*
FROM *table_name*;

Queries with constraints - 1

SELECT *column1, column2, …*
FROM *table_name*
WHERE *condition* AND/OR
another_condition AND/OR …;

SELECT column1, column2
FROM table_name
WHERE column1 **= 100**;

Operator	Condition	SQL Example
=, !=, < <=, >, >=	Standard numerical operators	col_name **!=** 4
BETWEEN ... AND ...	Number is within range of two values (inclusive)	col_name **BETWEEN** 1.5 **AND** 10.5
NOT BETWEEN ... AND ...	Number is not within range of two values (inclusive)	col_name **NOT BETWEEN** 1 **AND** 10
IN (...)	Number exists in a list	col_name **IN** (2, 4, 6)
NOT IN (...)	Number does not exist in a list	col_name **NOT IN** (1, 3, 5)

Queries with constraints - 2

SELECT *column1, column2, …*
FROM *table_name*
WHERE *condition* AND/OR
another_condition AND/OR …;

SELECT column1, column2
FROM table_name
WHERE column1 **LIKE** “QA%”;

Operator	Condition	Example
=	Case sensitive exact string comparison (**notice the single equals**)	col_name **=** "abc"
!= or <>	Case sensitive exact string inequality comparison	col_name **!=** "abcd"
LIKE	Case insensitive exact string comparison	col_name **LIKE** "ABC"
NOT LIKE	Case insensitive exact string inequality comparison	col_name **NOT LIKE** "ABCD"
%	Used anywhere in a string to match a sequence of zero or more characters (only with LIKE or NOT LIKE)	col_name **LIKE** "%AT%" (matches "AT", "ATTIC", "CAT" or even "BATS")
_	Used anywhere in a string to match a single character (only with LIKE or NOT LIKE)	col_name **LIKE** "AN_" (matches "AND", but not "AN")
IN (...)	String exists in a list	col_name **IN** ("A", "B", "C")
NOT IN (...)	String does not exist in a list	col_name **NOT IN** ("D", "E", "F")

Filtering and sorting query results - 1

SELECT **DISTINCT** *column1, column2, …*
FROM *table_name*
WHERE *condition(s);*

SELECT DISTINCT *column1, column2, …*
FROM *table_name*
WHERE *condition(s)*
ORDER BY *column2* ASC/DESC

SELECT DISTINCT director
FROM movies
ORDER BY director ASC;

Director
Andrew Stanton
Brad Bird
Brenda Chapman
Dan Scanlon
John Lasseter
Lee Unkrich
Pete Docter

Filtering and sorting query results - 2

SELECT *column1, column2, …*
FROM *table_name*
WHERE *condition(s)*
ORDER BY *column* ASC/DESC
LIMIT num_limit OFFSET num_offset;

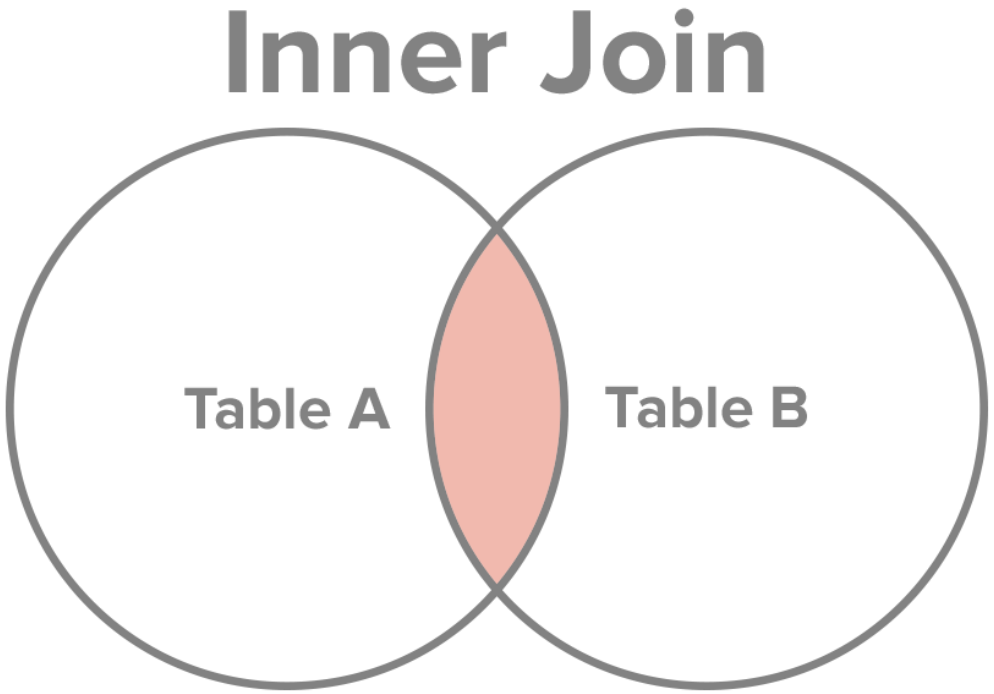
SELECT title
FROM movies
ORDER BY title ASC
LIMIT 5 OFFSET 5;

Table: Movies

Title
Monsters University
Monsters, Inc.
Ratatouille
The Incredibles
Toy Story

Multi-table queries: (INNER) JOIN

SELECT *column1, column2, …* FROM *table1_name*
INNER JOIN *table2_name*
ON *table1_name.column_name = table2_name.column_name*
WHERE *condition(s)* ORDER BY *column*, … ASC/DESC
LIMIT num_limit OFFSET num_offset;



(INNER) JOIN example

Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000

```
SELECT title, domestic_sales, international_sales
FROM movies
    JOIN boxoffice
    ON movies.id = boxoffice.movie_id;
```

Query Results

Title	Domestic_sales	International_sales
Finding Nemo	380843261	555900000
Monsters University	268492764	475066843
Ratatouille	206445654	417277164
Cars 2	191452396	368400000
Toy Story 2	245852179	239163000
The Incredibles	261441092	370001000

Another example

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3001	Brad Guzan	London		5005
3004	Fabian Johns	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Camero	Berlin	100	5003
3008	Julian Green	London	300	5002

Sample table: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5003	Lauson Hen		0.12
5007	Paul Adam	Rome	0.13

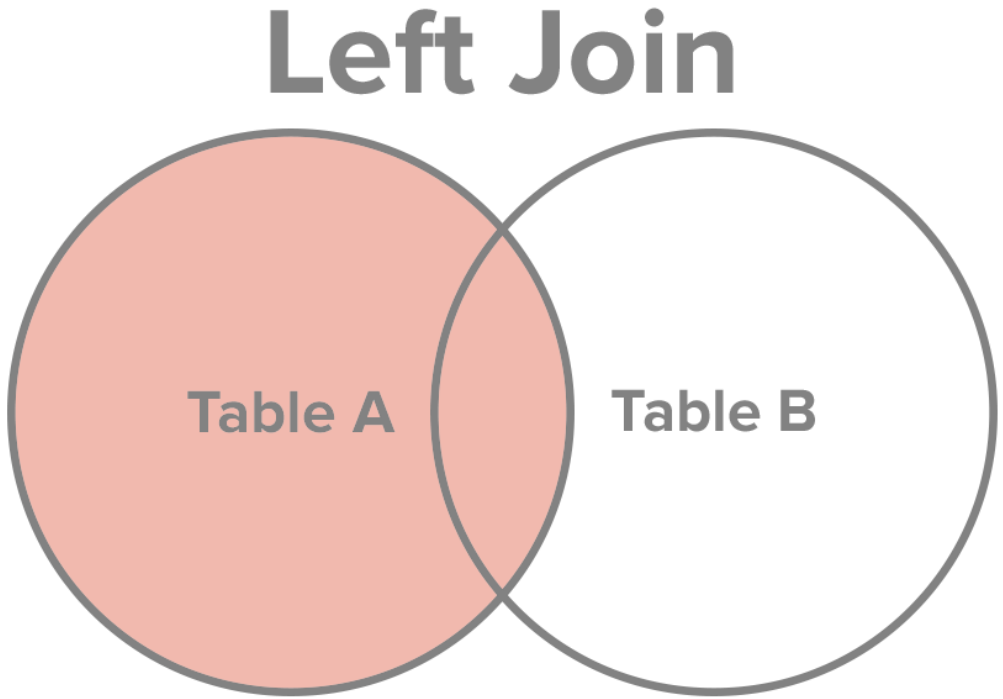
```
SELECT c.cust_name AS "Customer Name", c.city,
s.name AS "Salesman", s.commission
FROM customer c
        INNER JOIN salesman s
        ON c.salesman_id=s.salesman_id;
```

Output of the Query:

Customer Name	city	Salesman	commission
Nick Rimando	New York	James Hoog	0.15
Brad Davis	New York	James Hoog	0.15
Graham Zusi	California	Nail Knite	0.13
Julian Green	London	Nail Knite	0.13
Fabian Johnson	Paris	Mc Lyon	0.14
Geoff Cameron	Berlin	Lauson Hen	0.12
Jozy Altidor	Moscow	Paul Adam	0.13
Brad Guzan	London	Pit Alex	0.11

Multi-table queries: LEFT (OUTER) JOIN

SELECT *column1, column2, …* FROM *table1_name*
LEFT JOIN *table2_name*
    ON *table1_name.column_name = table2_name.column_name*
WHERE *condition(s)*
ORDER BY *column, …* ASC/DESC
LIMIT num_limit OFFSET num_offset;



(OUTER) LEFT JOIN example

Table: Buildings (Read-Only)

Building_name	Capacity
1e	24
1w	32
2e	16
2w	20

Table: Employees (Read-Only)

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2
Artist	Sherman D.	2w	8

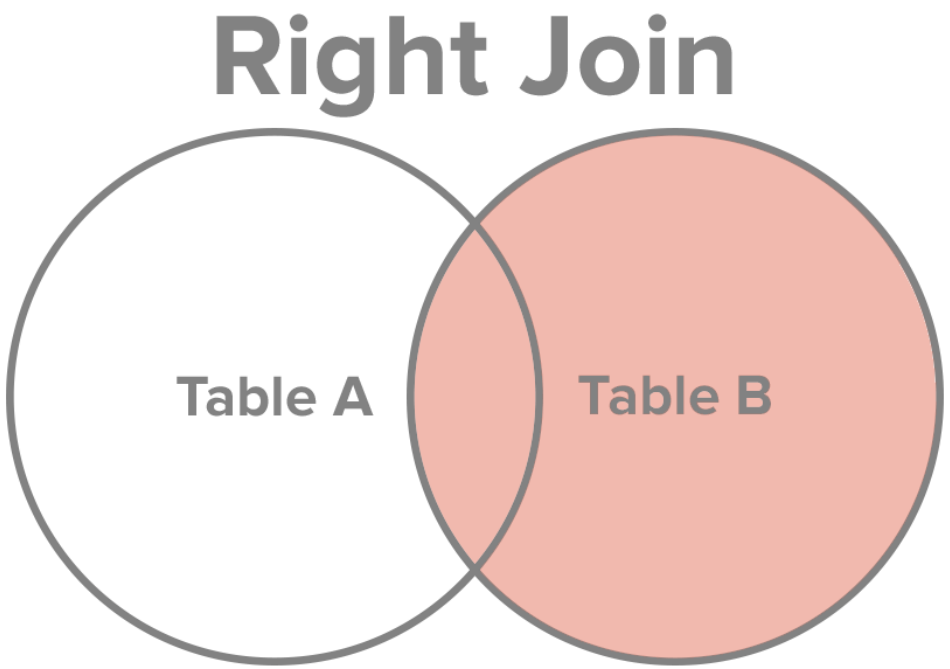
```
SELECT DISTINCT building_name,
role
FROM buildings
    LEFT JOIN employees
    ON building_name = building;
```

Query Results

Building_name	Role
1e	Engineer
1e	Manager
1w	
2e	
2w	Artist
2w	Manager

Multi-table queries: RIGHT (OUTER) JOIN

SELECT *column1, column2, …* FROM *table1_name*
RIGHT JOIN *table2_name*
    ON *table1_name.column_name = table2_name.column_name*
WHERE *condition(s)*
ORDER BY *column*, … ASC/DESC
LIMIT num_limit OFFSET num_offset;



(OUTER) RIGHT JOIN example

CUSTOMERS Table

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

ORDERS Table

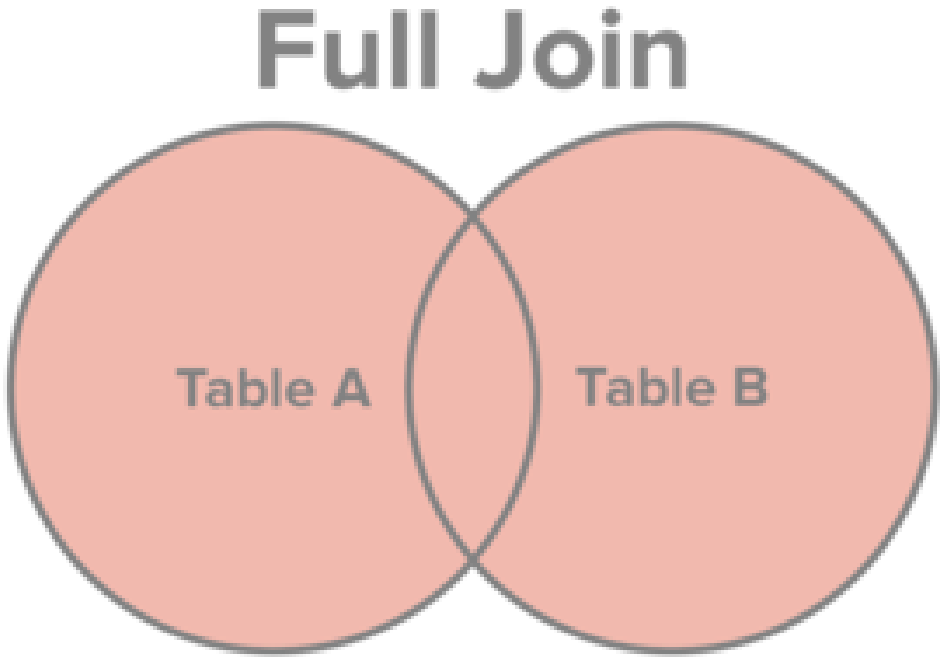
OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

```
SELECT  id, name, amount, date
FROM customers
  RIGHT JOIN orders
  ON customers.id = orders.customer_id;
```

ID	NAME	AMOUNT	DATE
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
2	Khilan	1560	2009-11-20 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00

Multi-table queries: FULL (OUTER) JOIN

SELECT *column1, column2, …* FROM *table1_name*
FULL JOIN *table2_name*
    ON *table1_name.column_name = table2_name.column_name*
WHERE *condition(s)*
ORDER BY *column, …* ASC/DESC
LIMIT num_limit OFFSET num_offset;



(OUTER) FULL JOIN example

CUSTOMERS Table

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

ORDERS Table

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

```
SELECT  id, name, amount, date
FROM customers
  FULL JOIN orders
  ON customers.id = orders.customer_id;
```

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-20 00:00:00
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
2	Khilan	1560	2009-11-20 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00

Queries with expressions: AS

SELECT ***column_name* AS *new_column_name***, …
FROM *table_name*;

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

SELECT COUNT(*) **AS** "RECORDS"
FROM CUSTOMERS;

RECORDS
7

Common aggregate functions

SELECT AGG_FUNC(*column_or_expression*) AS aggregate_description, …
FROM table_name
WHERE *constraint_expression*
GROUP BY *column;*

Function	Description
COUNT(*), **COUNT(*column*)**	A common function used to counts the number of rows in the group if no column name is specified. Otherwise, count the number of rows in the group with non-NULL values in the specified column.
MIN(*column*)	Finds the smallest numerical value in the specified column for all rows in the group.
MAX(*column*)	Finds the largest numerical value in the specified column for all rows in the group.
AVG(*column*)	Finds the average numerical value in the specified column for all rows in the group.
SUM(*column*)	Finds the sum of all numerical values in the specified column for the rows in the group.

Inserting rows

INSERT INTO *table_name*
VALUES *(value1, another_value1, …),*
(value2, another_value2,…), …;

INSERT INTO *table_name**(column1, column2, …)***
VALUES *(value1, another_value1, …),*
(value2, another_value2, …), …;

```
INSERT INTO CUSTOMERS
VALUES (7, 'Muffy', 24, 'Indore', 10000.00 );
```

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00 );
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Updating rows

UPDATE *table_name*
SET *column1 = value1_or_expr1,*
column2 = value2_or_expr2, …
WHERE *condition;*

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	El Directore	1998	95

UPDATE movies
SET director = "John Lasseter“
WHERE id = 2;

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95

Deleting rows

DELETE FROM *table_name*
WHERE *condition*;

DELETE FROM movies
WHERE year < 2005;

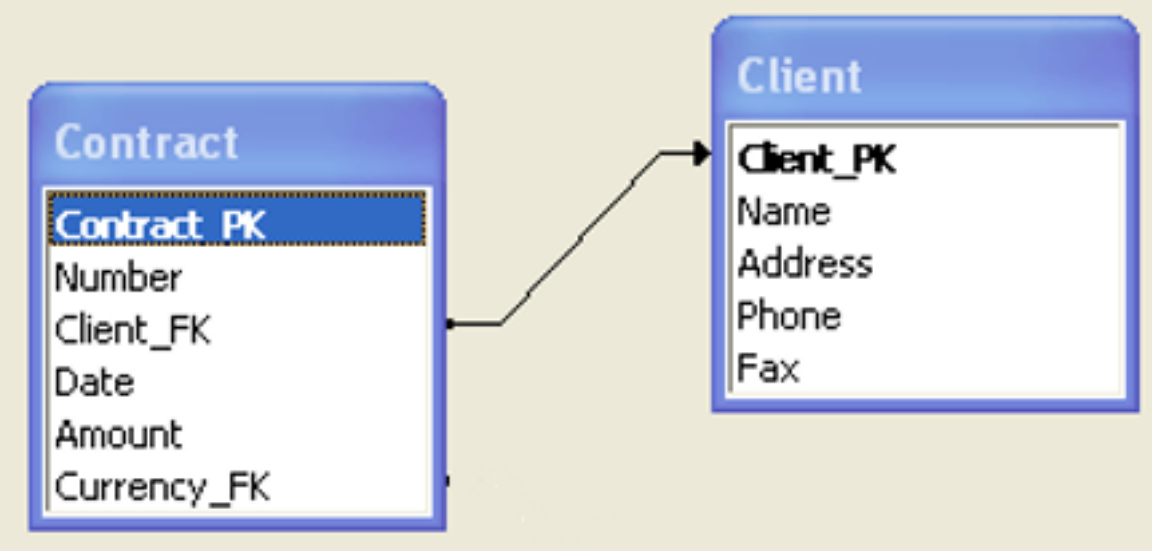
Table: Movies

Id	Title	Director	Year	Length_minutes
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101

Complex queries (subqueries)



Select contracts with client names started with “Jack”.

1. SELECT Client_PK FROM Client
   WHERE Name LIKE “Jack%”

2. SELECT * FROM Contract
   WHERE Client_FK in (…, …, …, …)

```
SELECT * FROM Contract
WHERE Client_FK in
        (SELECT Client_PK FROM Client
         WHERE Name LIKE “Jack%”)
```

Task - 1

Sample table: movie

mov_id	mov_title	mov_year	mov_time	mov_lang	mov_dt_rel	mov_rel_country
901	Vertigo	1958	128	English	1958-08-24	UK
902	The Innocents	1961	100	English	1962-02-19	SW
903	Lawrence of Arabia	1962	216	English	1962-12-11	UK
904	The Deer Hunter	1978	183	English	1979-03-08	UK
905	Amadeus	1984	160	English	1985-01-07	UK
906	Blade Runner	1982	117	English	1982-09-09	UK
907	Eyes Wide Shut	1999	159	English		UK
908	The Usual Suspects	1995	106	English	1995-08-25	UK
909	Chinatown	1974	130	English	1974-08-09	UK

Write an SQL query to find names and years of movies

Task - 2

Sample table: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5003	Lauson Hen		0.12
5007	Paul Adam	Rome	0.13

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3001	Brad Guzan	London		5005
3004	Fabian Johns	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Camero	Berlin	100	5003
3008	Julian Green	London	300	5002

Write an SQL statement to prepare a list **“Salesman”** with a **salesman name**, a **“Customer”** with a **customer name** and their **“City”** for a salesman and a customer, who belongs to the **same city**

Task - 3

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	5002
3001	Brad Guzan	London		5005
3004	Fabian Johns	Paris	300	5006
3007	Brad Davis	New York	200	5001
3009	Geoff Camero	Berlin	100	5003
3008	Julian Green	London	300	5002

Sample table: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5003	Lauson Hen		0.12
5007	Paul Adam	Rome	0.13

Write an SQL statement to find out which **salesman** works for which customer. The output result should contain **2 columns**: “Salesman” and “Customer”.

Task - 4

Sample table: emp_details

EMP_IDNO	EMP_FNAME	EMP_LNAME	EMP_DEPT
127323	Michale	Robbin	57
526689	Carlos	Snares	63
843795	Enric	Dosio	57
328717	Jhon	Snares	63
444527	Joseph	Dosni	47
659831	Zanifer	Emily	47
847674	Kuleswar	Sitaraman	57
748681	Henrey	Gabriel	47
555935	Alex	Manuel	57

Write an SQL query to display all the data of employees whose last name begins with ‘D’

Task - 5

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3005	Graham Zusi	California	200	
3001	Brad Guzan	London		5005
3004	Fabian Johns	Paris	300	5006
3007	Brad Davis	New York	200	
3009	Geoff Camero	Berlin	100	5003
3008	Julian Green	London	300	5002

Sample table: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5003	Lauson Hen		0.12
5007	Paul Adam	Rome	0.13

Write an SQL statement to make a list in **ascending order** for customers who hold a **grade less than 300** and work either through a salesman or by his own. The output result should contain **customer name, customer city, grade, salesman name, salesman city**

Task - 6

Sample table: Salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5003	Lauson Hen		0.12
5007	Paul Adam	Rome	0.13

Sample table: Orders

70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003
70012	250.45	2012-06-27	3008	5002
70011	75.29	2012-08-17	3003	5007
70013	3045.6	2012-04-25	3002	5001

Write an SQL statement(s) to display **all orders** from the Orders table issued by the salesman **Paul Adam**

Task - 7

Sample table: nurse

employeeid	name	position	registered	ssn
101	Carla Espinosa	Head Nurse	t	111111110
102	Laverne Roberts	Nurse	t	222222220
103	Paul Flowers	Nurse	f	333333330

Sample table: on_call

nurse	blockfloor	blockcode	oncallstart	oncallend
101	1	1	2008-11-04 11:00:00	2008-11-04 19:00:00
101	1	2	2008-11-04 11:00:00	2008-11-04 19:00:00
102	1	3	2008-11-04 11:00:00	2008-11-04 19:00:00
103	1	1	2008-11-04 19:00:00	2008-11-05 03:00:00
103	1	2	2008-11-04 19:00:00	2008-11-05 03:00:00
103	1	3	2008-11-04 19:00:00	2008-11-05 03:00:00

Sample table: room

roomnumber	roomtype	blockfloor	blockcode	unavailable
101	Single	1	1	f
102	Single	1	1	f
103	Single	1	1	f
111	Single	1	2	f
112	Single	1	2	t
113	Single	1	2	f
121	Single	1	3	f
122	Single	1	3	f
123	Single	1	3	f

Write an SQL query to display names of all nurses who have ever been on call for the room 122

Task - 8

Sample table: employees

emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
68319	KAYLING	PRESIDENT		1991-11-18	6000.00		1001
66928	BLAZE	MANAGER	68319	1991-05-01	2750.00		3001
67832	CLARE	MANAGER	68319	1991-06-09	2550.00		1001
65646	JONAS	MANAGER	68319	1991-04-02	2957.00		2001
67858	SCARLET	ANALYST	65646	1997-04-19	3100.00		2001
69062	FRANK	ANALYST	65646	1991-12-03	3100.00		2001
63679	SANDRINE	CLERK	69062	1990-12-18	900.00		2001
64989	ADELYN	SALESMAN	66928	1991-02-20	1700.00	400.00	3001
65271	WADE	SALESMAN	66928	1991-02-22	1350.00	600.00	3001

Sample table: department

dep_id	dep_name	dep_location
1001	FINANCE	SYDNEY
2001	AUDIT	MELBOURNE
3001	MARKETING	PERTH
4001	PRODUCTION	BRISBANE

Write an SQL query to list (salary + commission) of all sales persons of the MARKETING department

Homework

Complete all 8 tasks (slides 43-50) and send to your teacher.

Links for self-training

- https://sqlbolt.com/
- https://www.w3schools.com/sql/default.asp
- https://sqlzoo.net

Reference

- Database types - https://www.dataversity.net/review-pros-cons-different-databases-relational-versus-non-relational/


THANK YOU FOR YOUR ATTENTION
YouTube