2.1 Overview of SQL

Introduction to SQL

In this section, you will learn about what SQL is, how it is related to databases and how to use SQL for various database operations.
Watch the following 4-minute 26-second video to learn what is SQL.

What is SQL? [in 4 minutes for beginners]

https://youtu.be/27axs9dO7AE

Source: (Danielle Thé, 2019)

2.1.1 Introduction to SQL

What is SQL?

Structural Query Language (SQL) is the standard language for dealing with **Relational Databases**. As a language, SQL allows to handle the information **using tables** and shows a language to **query these tables** and other objects related (views, functions, procedures, etc.). It is used to **insert**, **search**, **update**, and **delete** database records.

In addition to basic operations, SQL can also be used for other **advanced database operations** like optimising and maintaining the database. However, these advanced features depend highly on the type of databases (for example, **SQL Server**, **Oracle**, **PostgreSQL**, **MySQL**, and **SQLite**). So, some commands of SQL do vary according to the database.

Background to SQL

In 1970, E.F. Codd published a paper called "A Relational Model of Data for Large Shared Data Banks", in which he proposed a language for working with data in a relational database.

Based on Codd's proposal, IBM developed a language called Structured English Query Language (SEQUEL), which later became SQL.
In 1979, Relational Software Inc. (now Oracle) introduced the first commercially available implementation of SQL.

Today, the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) have both accepted SQL as the standard language for working with relational databases.

There are various dialects of SQL, such as MySQL, PostgresQL, and Oracle SQL. Although they differ slightly, for most of the fundamental parts of the language they are the same. In these slides you will be learning MySQL, but most of what you learn is identical for PostgresQL and Oracle SQL.

Additional information

1. SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.
2. SQL is pronounced as "S-Q-L" or sometimes as "See-Quel".

Watch the 1-minute 56-second video below to gain an overview of SQL.

What is SQL (Structured Query Language)? Why is it Important?

https://youtu.be/T8ngx84oHFY

Source: (Eye on Tech, 2020)

SQL is not a general-purpose programming language.

Unlike general-purpose programming languages like Java, C, C++, Python etc., SQL is only limited to databases, and you can not create complete applications / Software using SQL. In simple terms, SQL is used for any application or software data part.

Who and why SQL?

SQL is a legacy method for handling data, and many roles (**Backend developer, QA engineer, ETL developer, system engineer, Data analyst, System admin**) in the software industry require to use of SQL.

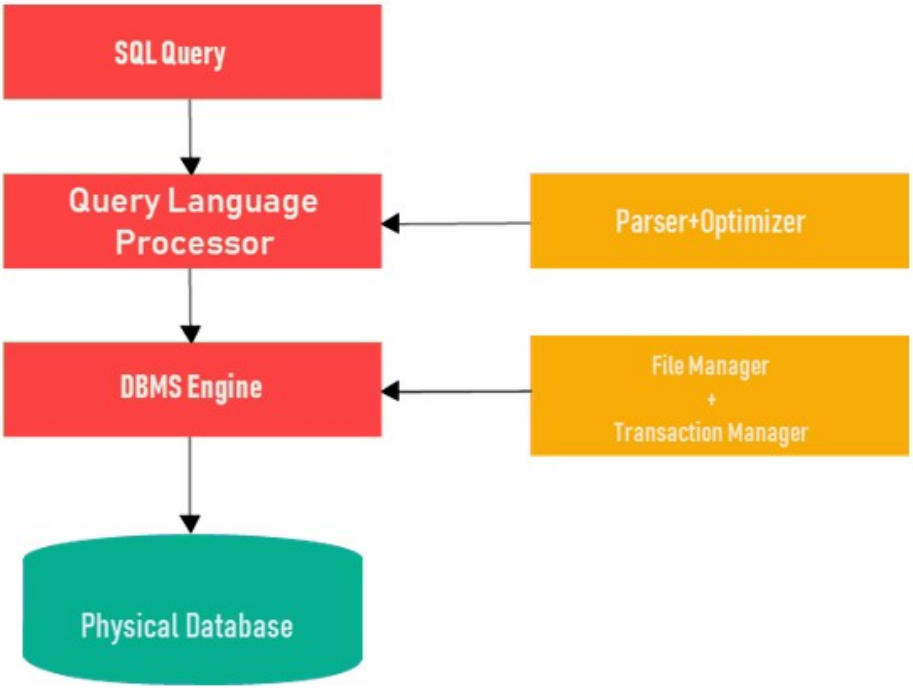
Recently, with the popularity of Big data and its application, SQL usage extended for handling big data, and Data analyst and **Data scientists have become new users of SQL.**

How does SQL work?

When you want to execute an SQL command for any DBMS system, you need to find the best method to carry out your request, and SQL engine determines how to interpret that specific task.

Important components included in this SQL process are:

- SQL Query Engine
- Optimisation Engines
- Query Dispatcher
- Classic Query Engine



Source: SQL process (Allen, 2024)

Read the following articles to find out more about SQL.

Key readings

- Allen, M. (2024). What is SQL? Learn SQL basics, SQL full form & how to use. Guru99. https://www.guru99.com/what-is-sql.html
- Kanade, V. (2022). What is SQL? Definition, elements, examples, and uses in 2022. Spiceworks. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-sql/

2.1.2 Reflection

How is data stored by software?

Data is written to a file.

Data is stay in primary memory.

Data is stored in a particular structure and handled by specialized software.

Data is stored like how we show them in the table.

Data is stored and accessed by SQL.

2.1.3 Database and SQL

What is Database and SQL?

Watch the following 6-minute 20-second video to explore Database and SQL.

What is Database & SQL?

https://youtu.be/FR4QIeZaPeM

Source: (Guru99, 2013)

A database is an organised collection of data that can be easily accessed. To manage these databases, Database Management Systems (DBMS) are used.

Types of DBMS:

1. Non-Relational
2. Relational

RDBMS stands for Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

SQL queries and other operations take the form of commands written as statements and are aggregated into programs that enable users to add, modify or retrieve data from database tables.

A table is the most basic unit of a database and consists of rows and columns of data. A single table holds records, and each record is stored in a row of the table.

Tables are the most used type of database objects or structures that hold or reference data in a relational database.

Other types of database objects include the following:

- Views are logical representations of data assembled from one or more database tables.
- Indexes are lookup tables that help speed up database lookup functions.
- Reports consist of data retrieved from one or more tables, usually a subset of that data that is selected based on search criteria.

Note: Remember the following order:

DBMS-> RDBMS-> Tables -> Rows + Columns -> Cell -> Value

Check out the following sites to learn more about Database and SQL:

Readings

- Programiz (n.d.). Introduction to Databases and SQL (with examples). https://www.programiz.com/sql/database-introduction
- Rai, V. (2021). What is a Database - A beginners guide. Stackby Blog. https://stackby.com/blog/what-is-a-database/

2.1.4 SQL Skills in Demand and Data Modelling

Most organisations need someone with SQL knowledge. Salaries for SQL-based positions vary depending on job type and experience but are generally above average.

Some positions that require SQL skills include:

- **Database administrator (DBA)**: This is someone who specialises in making sure data is being stored and managed properly and efficiently. Databases are most valuable when they allow users to retrieve desired combinations of data quickly and easily.
- **Database migration engineer**: This person specialises in moving data from various databases onto an SQL server.
- **Data scientist**: This is a position very similar to that of a data analyst, but data scientists typically are tasked with handling data in far greater volumes and accumulating it at much higher speeds.
- **Big data architect**: Someone in this role builds products for handling large volumes of data.

What is Data Modelling?

Refer to the key reading below to find out more about data modelling.

Key reading

- Ambler, S. (n.d.) Data modeling 101. AgileData. https://agiledata.org/essays/datamodeling101.html

2.1.5 Reflection

1. SQL syntax is made of complex technical terms that make execution fast.

   ANS: False
2. SQL is a modern programming language for developing complete applications.

   ANS: False
3. SQL is used by database administrators as well as data scientists.

   ANS: True
4. SQL stands for _____.

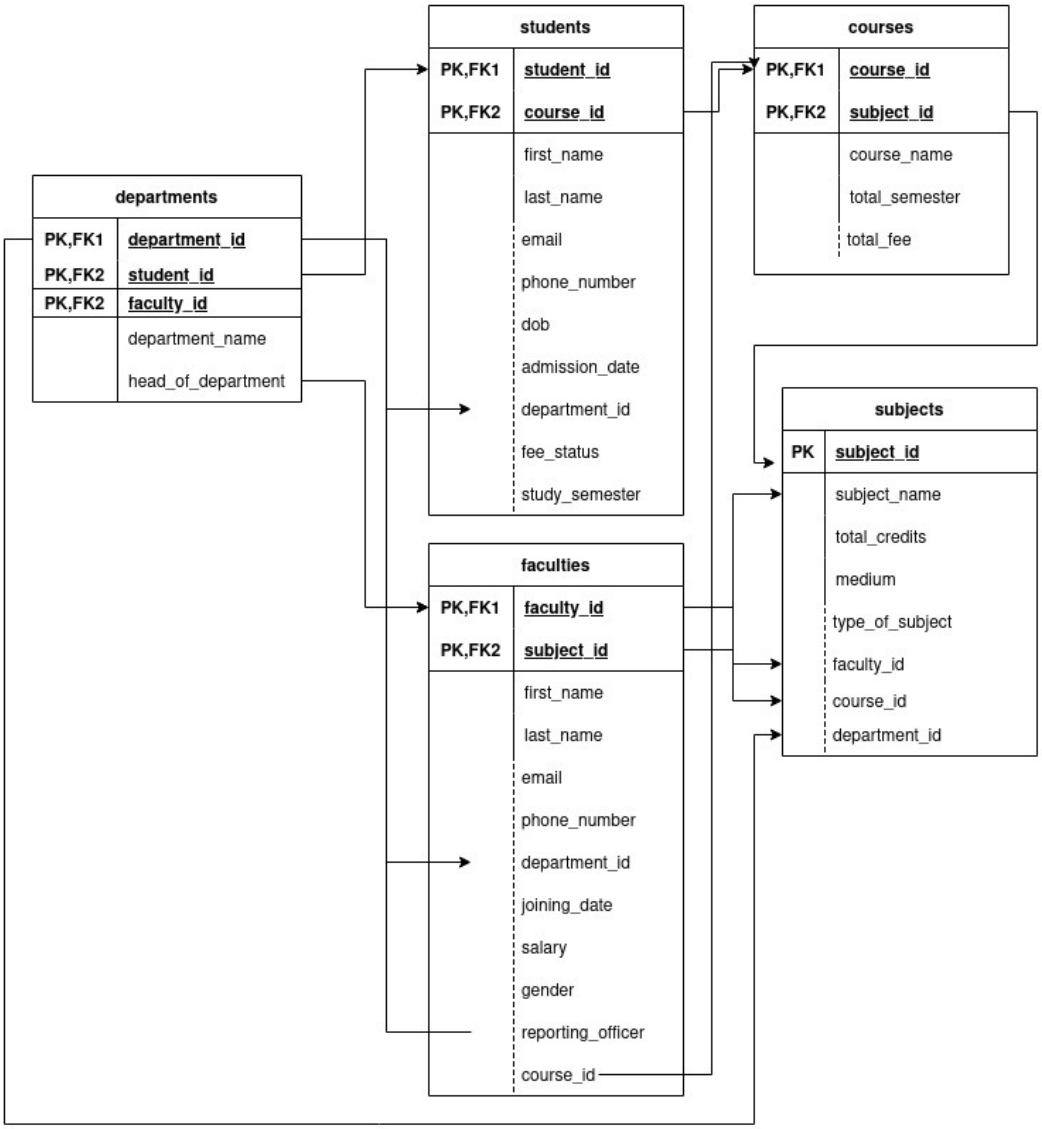
   ANS: Structural Query Language
5. In simple terms, SQL is used to create, update, query/read/retrieve the database.

   ANS: True

2.2 Sample database

In the previous sections, we have studied a lot about SQL. What is a database? Why it is essential and other dimensions of the database. This section will focus on learning about databases through a real example database.

We have designed a minimal version of the university database we will use throughout the session. Below is the schema or entity-relationship diagram of the university database. We focused more on linked tables and various attributes.



Source: Schema diagram of a university database (Sunway University, 2022)

In the designed database, we assume that a university has many departments, and each department has many students and faculties. Each student will be admitted to one of the courses, and each course will have more than one subject. Every faculty will also have one or more subjects assigned and will be given one or more courses. So, the university database contains information about five entities: departments, students, faculties, courses and subjects.

Each department has an id number (primary key), name and head of department (faculty_id). It will also have student_id and faculty_id as attributes (foreign key).

Each student has a student_id (primary key), a name (first and last name), email, phone_number, date of birth (dob), admission_date, fee_Status, and study_semester. It will also have course_id and department_id as attributes (foreign key).

Similar to the student, each faculty has faculty_id (primary key), a name (first and last name), email, phone_number, gender, joining_date, fee_Status, and reporting_officier (faculty_id). It will also have course_id and department_id as attributes (foreign key).

Each course has a course_id (primary key), course_name, total_semester, and total_fee. It will also have subject_id as the foreign key.

Each course comprises various subjects so that each subject will have a subject_id (primary key), subject_name, total_credits, medium, and type_of_subject. It will also have faculty_id, course_id, and department_id as attributes (foreign key).

You will learn how to use SQL to inspect the data in these tables as we proceed.

2.2.1 SQL sandpit

As you learn SQL during these slides you can use the space on the right to write and execute SQL statements to experiment. The sample HR database is loaded for you and ready to query.

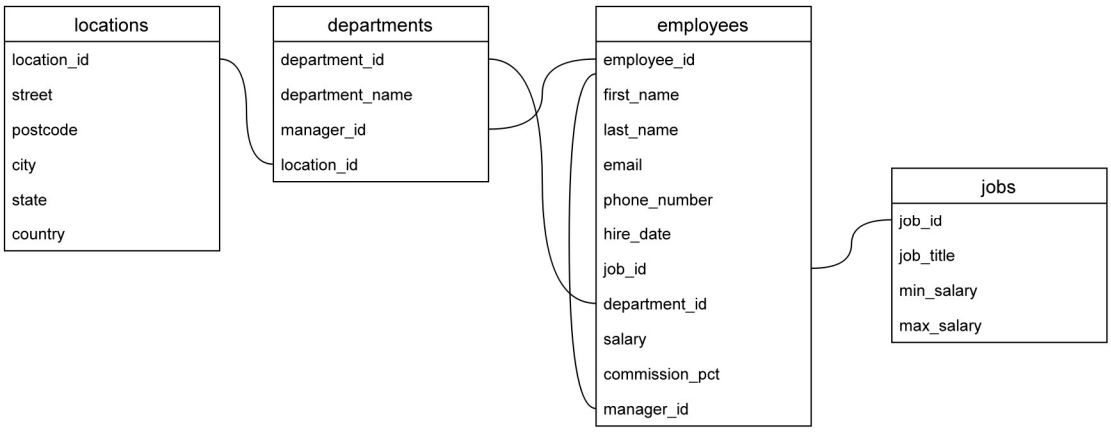
Try the following as an example (you don't need to understand it at this stage - we'll get to that). Copy and paste the following code into the panel on the right and then click "Run":

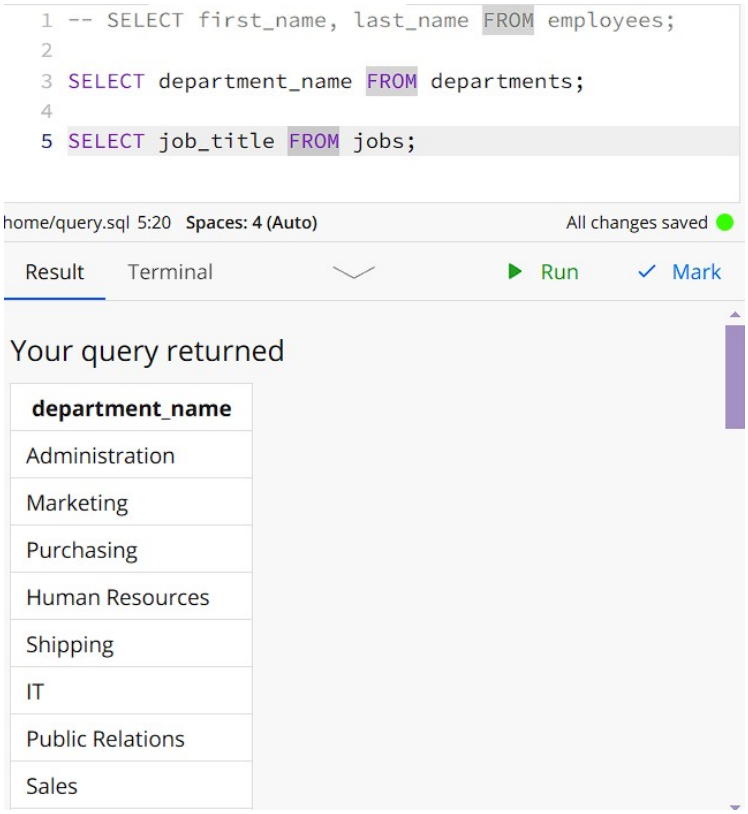
```
SELECT first_name, last_name FROM employees;
```

The query will be executed and the results of the query will be displayed below it.

You can come back to this slide whenever you like, and experiment with writing your own SQL queries.

To write your SQL statements you need to understand the structure of the HR database. You were given that in the previous slide, but here is the schema again, repeated for your convenience:





2.2.2 Getting Started with SQL

With the name Structured Query Language (SQL), it is evident that SQL is a query language, so like any other programming language, it has its syntax and rule. To work with the database, we must follow the syntax, rules and many conventions (not necessary but helpful to follow standard conventions).

Note: There is little variation in SQL syntax as per the databases, but mostly all follow the standard rules and syntax.

SQL basic rule

- SQL is based on keywords, and one must know them to use and avoid using keywords using as the names of other places. For example, "CREATE", "SELECT", and "FROM", etc. are keywords, and one should not use this as a table name.
- It is good practice to use a semicolon to end a SQL statement. However, many a time, it is not required but a good practice.

- Using CamelCase or underscore to join the table or column name is a good practice. StudentID or student_id is better than student id. If there is space between names, that must always be used with double quotes like "student id".
- SQL is case insensitive. For example, "CREATE" and "create" will be created the same by the SQL engine.
- Like any programming language, the comment is also available in SQL. Single line comment is given by using two dashes -- , and a multi-line comment can be used by placing text within /* and */.

```mysql
-- This is a whole line comment
/* This is a multi-line
 line comment */
SELECT subject_name -- This is an end of line comment
FROM courses;
```

Data types

Like any programming language, SQL uses different datatypes to store attribute values. Available data types vary as per databases.

The following are data types available in SQL Server.

- Numbers: integer and float
- Strings (Unicode and ASCII Character): "Hello", ""
- Binary
- Dates: "2020-06-24"
- Misc

You can read more about data types in this Digital Ocean tutorial (https://www.digitalocean.com/community/tutorials/sql-data-types).

Operator in SQL

An operator in SQL either comes from a set of reserved keywords or a set of special characters. These operators use the WHERE clause to perform various comparisons and arithmetic operations. The operator helps to specify conditions in SQL statements and helps to chain multiple conditions in a statement.

There are three main sets of operators:

- Arithmetic operators

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Modulus(%)

- Comparison operators (=, !=, <>, <, > etc.)
- Logical operators (Examples)
  - ALL
  - AND
  - ANY
  - BETWEEN
  - IN
  - LIKE
  - OR
  - NOT
  - UNIQUE

The function and working of these operators will be explained in their first use in the following section. You can also explore and learn more about operations in SQL from the following article.

Key reading

Clark, D. (2022). SQL operators: 6 different types (w/ 45 code examples). Dataquest. https://www.dataquest.io/blog/sql-operators/

2.2.3 Working with Database (CREATE, DROP, and USE)

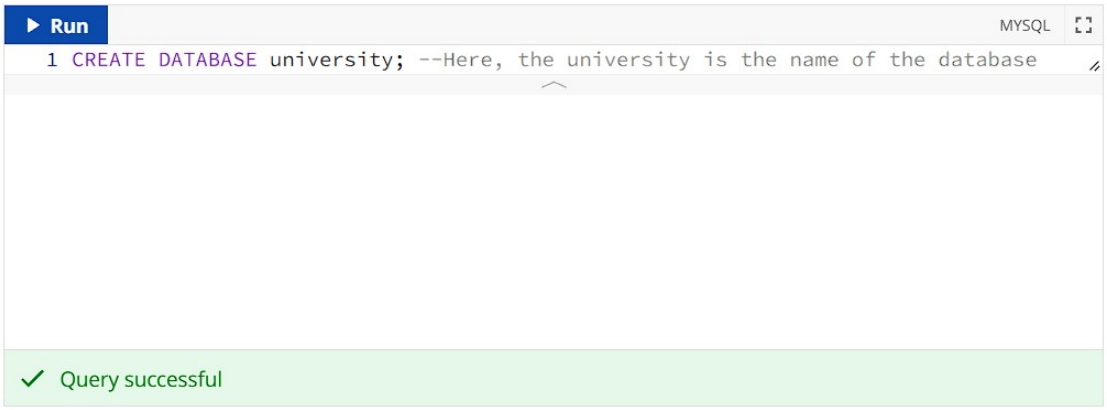
The first step of working with SQL or RDBMS is to create a database.
As mentioned in the previous section, we will be following the university database for all the examples and SQL learning, so first, we will create the university dataset.

Creating a database

The syntax for creating a database in SQL is very simple, and a database can be created by using two keywords: CREATE and DATABASE, following the name of the database. So, the complete SQL query for creating the university database will be following:

```
CREATE DATABASE university; --Here, the university is the name of the database
```

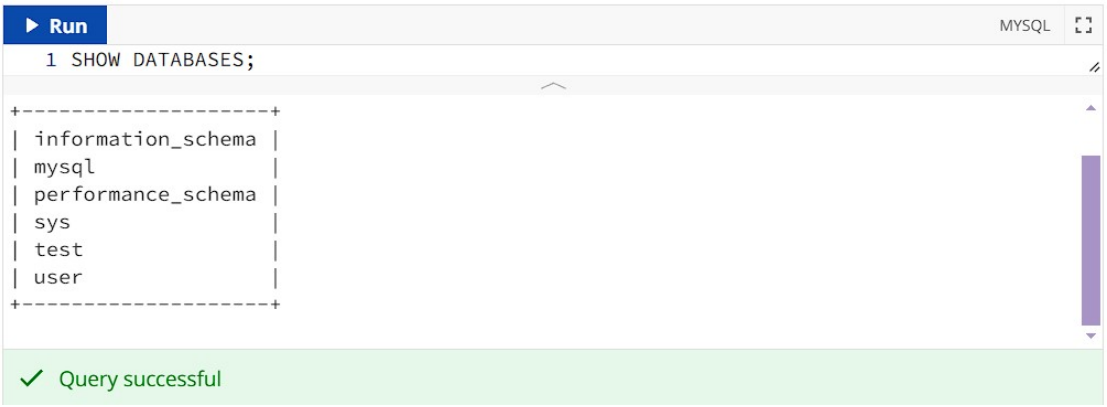
- The database name should be unique, i.e. if there is already a database with the name "university", then the above query will throw an error.
- It is good practice to avoid space in the database names and to give a long name.



Show databases

One can list and see all the available databases by running the SHOW query. Example, Note: If you are unsure about the available database, it is good practice to list and see the databases before creating a new one.

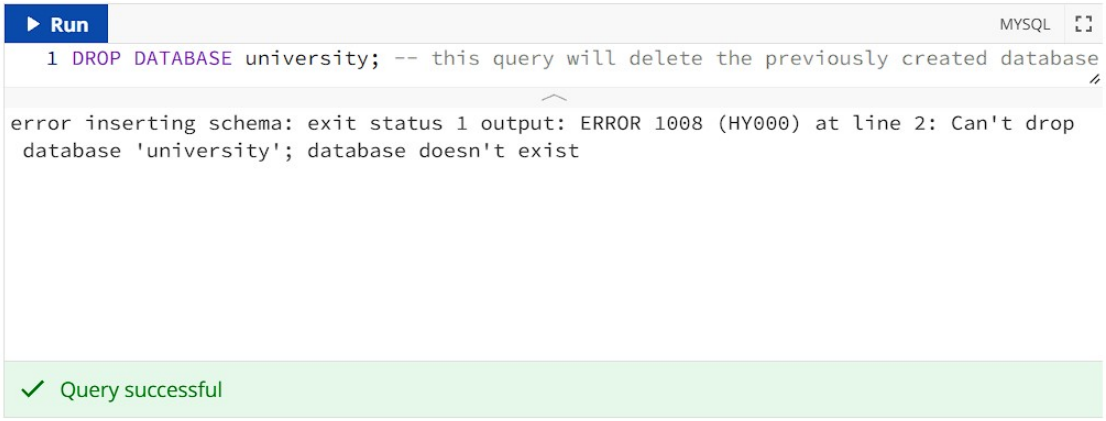
```
SHOW DATABASES; -- this will list all the databases
```



Drop a database

A database can be deleted by using the DROP and DATABASE keywords with the database name.

```
DROP DATABASE university; -- this query will delete the previously created database university.
```



Using a database

As mentioned previously, there can be many databases, so one must choose a database before applying any changes or running a SQL query.

It is very simple to choose a database to use, one can simply run a USE query with the database name. For example, to work with the university database, we can run the following command (USE is mainly the first query to start working with an existing database).

```
USE university; -- This query will set the current database to university, and all further SQL queries will be applied to the university database.
```

Note: Again, if we don't know the database name, then we can use the SHOW DATABASES; query to list the available databases.

2.2.4 Working with tables (CREATE, INSERT)

In the previous section, we created our demo database "university", and we have selected it to work upon.

However, our university database is empty and does not have any tables. So, we must create all the required tables (departments, students, faculties, courses, and subjects).

Creating a table

To create a table, we will again use two important keywords, CREATE and TABLE. To create a table, we need to provide a table name and all its column names with the column's data type.
So, the normal query syntax for creating a table will be:

```
CREATE TABLE table_name (
columnName1 datatype,
columnName2 datatype,
...
columnNameN datatype,
PRIMARY KEY (one or more column name/s)
);
CREATE TABLE departments (
department_id INT NOT NULL, student_id INT NOT NULL, faculty_id INT NOT NULL,
department_name VARCHAR (20) NOT NULL, head_of_department INT NOT NULL,
PRIMARY KEY (department_id, student_id, faculty_id)
);
```

```mysql
CREATE TABLE departments (

department_id INT NOT NULL,  student_id INT NOT NULL, faculty_id INT NOT NULL,

department_name VARCHAR (20) NOT NULL,  head_of_department INT NOT NULL,

PRIMARY KEY (department_id, student_id, faculty_id)

);
```

Query successful

Similar to the "departments" table, all other required tables can be created.

```mysql
CREATE TABLE students (
student_id INT NOT NULL,
course_id INT NOT NULL,
first_name VARCHAR (20) NOT NULL,
last_name VARCHAR (20) NOT NULL,
email VARCHAR (20) NOT NULL,
phone_number VARCHAR (10) NOT NULL,
dob DATE  NOT NULL,
admission_date DATE NOT NULL,
department_id INT NOT NULL,
fee_status VARCHAR (6) NOT NULL, -- paid and unpaid
study_semester INT NOT NULL, -- 1,2,3,4,5,6,7,8
PRIMARY KEY (student_id, course_id, department_id )
);
```

The following SQL query will create faculties table:

```mysql
faculty_id INT NOT NULL,
subject_id INT NOT NULL,
first_name VARCHAR (20) NOT NULL,
last_name VARCHAR (20) NOT NULL,
email VARCHAR (20) NOT NULL,
phone_number VARCHAR (10) NOT NULL,
department_id INT NOT NULL,
joining_date DATE NOT NULL,
salary DECIMAL (18, 2),
gender CHAR(1), -- M, F, O
reporting_officer INT NOT NULL,
course_id INT NOT NULL,
PRIMARY KEY (faculty_id, subject_id)
);
```

Similarly, the courses and subjects table will be created using the following queries:

```mysql
CREATE TABLE courses (
  course_id INT NOT NULL,
  subject_id INT NOT NULL,
  course_name VARCHAR (20) NOT NULL,
  total_semester INT NOT NULL,
  total_fee DECIMAL (18, 2),
  PRIMARY KEY (course_id, subject_id)
);
```

```mysql
CREATE TABLE subjects (
  subject_id INT NOT NULL,
  subject_name VARCHAR (20) NOT NULL,
  total_credits INT NOT NULL,
  medium VARCHAR (10) NOT NULL, -- English, Non-english
  type_of_subject VARCHAR (8) NOT NULL, -- theory , pratical
  faculty_id INT NOT NULL,
  course_id INT NOT NULL,
  department_id INT NOT NULL,
  PRIMARY KEY (subject_id)
);
```

2.2.5 Showing the Table schema

You created the database "university" and all five required tables in the previous two sections. To list the created tables, you can use the SHOW (similar to the database) with TABLES keywords.

Following SQL query with the list of all the tables under the "university" database.

```mysql
SHOW TABLES;
```

```
+----------------+
| Tables_in_user |
+----------------+
| departments    |
| employees      |
| jobs           |
| locations      |
+----------------+
```

Query successful

During creating the table (using CREATE TABLE), you have given various datatype and other constraints. You can verify each table structure using DESCRIBE keyword with the table name. For example, you can run the following SQL query to know the various department attributes with its datatype.

Run Schema Query MYSQL

```
1 DESCRIBE departments;
```

```
+-----------------+----------+------+-----+---------+-------+
| Field           | Type     | Null | Key | Default | Extra |
+-----------------+----------+------+-----+---------+-------+
| department_id   | int(11)  | YES  |     | NULL    |       |
| department_name | tinytext | YES  |     | NULL    |       |
| manager_id      | int(11)  | YES  |     | NULL    |       |
| location_id     | int(11)  | YES  |     | NULL    |       |
+-----------------+----------+------+-----+---------+-------+
```

✓ Query successful

Similar to the departments, you can run DESCRIBE query to all the other tables of the university database as follows:

Run Schema Query MYSQL

```
1 DESCRIBE employees;
```

```
| phone_number   | tinytext     | YES  |     | NULL    |       |
| hire_date      | date         | YES  |     | NULL    |       |
| job_Id         | int(11)      | YES  |     | NULL    |       |
| department_id  | int(11)      | YES  |     | NULL    |       |
| salary         | int(11)      | YES  |     | NULL    |       |
| commission_pct | decimal(3,2) | YES  |     | NULL    |       |
| manager_id     | int(11)      | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
```

✓ Query successful

Run Schema Query MYSQL

```
1 DESCRIBE jobs;
```

```
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| job_id     | int(11)  | YES  |     | NULL    |       |
| job_title  | tinytext | YES  |     | NULL    |       |
| min_salary | int(11)  | YES  |     | NULL    |       |
| max_salary | int(11)  | YES  |     | NULL    |       |
+------------+----------+------+-----+---------+-------+
```

✓ Query successful

Run Schema Query MYSQL

```
1 DESCRIBE locations;
```

```
+-------------+----------+------+-----+---------+-------+
| location_id | int(11)  | YES  |     | NULL    |       |
| street      | tinytext | YES  |     | NULL    |       |
| postcode    | tinytext | YES  |     | NULL    |       |
| city        | tinytext | YES  |     | NULL    |       |
| state       | tinytext | YES  |     | NULL    |       |
| country     | tinytext | YES  |     | NULL    |       |
+-------------+----------+------+-----+---------+-------+
```

✓ Query successful

2.2.6 Putting Data in Table (INSERT)

The work you have put in the last there sections/slides helped you to create:

- A database: university
- Five tables under the university database: departments, students, facilities, courses, and subjects

Till now, there is no actual data (rows in the table) in any of these tables, i.e. we have just created the structure of your tables.

Now, in this section, we will use SQL query to putting data (rows) in each table. For the example, we will use dummy data per table (at least 5-10 entries in each table). You can try adding more as practice.

You can use the INSERT keyword to create a SQL query to put data into the table. There are two ways to use the INSERT query:

1. With column name and value
2. Only values for each column (shorthand)

1) With column name and value

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN) VALUES
(value1, value2, value3,...valueN);INSERT INTO TABLE_NAME VALUES
(value1,value2,value3,...valueN);
```

2) Only values for each column (shorthand)

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

For the shorthand method, you must provide values for all columns and keep the same order of values per column.

For example, we can use either of the SQL queries to insert values in the "departments" table.

```
INSERT INTO departments (department_id, student_id, faculty_id,
department_name, head_of_department) VALUES (1, 101, 1001, 'Computer
Science', 1005 );
```

```
or
```

```
INSERT INTO departments VALUES (1, 101, 1001, 'Computer Science', 1005 );
```

```
Run                                                    MYSQL
1 INSERT INTO departments (department_id, student_id, faculty_id, department_name, he
2 VALUES (1, 101, 1001, 'Computer Science', 1005 );
```

```
Run                                                    MYSQL
1 INSERT INTO departments VALUES (1, 101, 1001, 'Computer Science', 1005 );
```

We can see that the shorthand notation of the SQL query is simple for inserting data.

So, we can create multiple entries to the table using this notation.

```
Run                                                    MYSQL
1  INSERT INTO departments VALUES (1, 101, 1001, 'Computer Science', 1005 );
2  INSERT INTO departments VALUES (1, 102, 1002, 'Computer Science', 1005 );
3  INSERT INTO departments VALUES (1, 103, 1003, 'Computer Science', 1005 );
4  INSERT INTO departments VALUES (1, 104, 1004, 'Computer Science', 1005 );
5  INSERT INTO departments VALUES (1, 105, 1005, 'Computer Science', 1005 );
6  INSERT INTO departments VALUES (2, 201, 2001, 'Science', 2005 );
7  INSERT INTO departments VALUES (2, 202, 2002, 'Science', 2005 );
8  INSERT INTO departments VALUES (2, 203, 2003, 'Science', 2005 );
9  INSERT INTO departments VALUES (2, 204, 2004, 'Science', 2005 );
10 INSERT INTO departments VALUES (2, 205, 2005, 'Science', 2005 );
11 INSERT INTO departments VALUES (2, 206, 2006, 'Science', 2005 );
12 INSERT INTO departments VALUES (3, 301, 3001, 'Commerce', 3005 );
13 INSERT INTO departments VALUES (4, 401, 4001, 'History', 4005 );
14 INSERT INTO departments VALUES (5, 501, 5001, 'Managment', 5005 );
```

The above queries will insert 14 rows to the departments table, as follows:

department_id	student_id	faculty_id	department_name	head_of_department
1	101	1001	Computer Science	1005
1	102	1002	Computer Science	1005
1	103	1003	Computer Science	1005
1	104	1004	Computer Science	1005
1	105	1005	Computer Science	1005
2	201	2001	Science	2005
2	202	2002	Science	2005
2	203	2003	Science	2005
2	204	2004	Science	2005
2	205	2005	Science	2005
2	206	2006	Science	2005
3	301	3001	Commerce	3005
4	401	4001	History	4005
5	501	5001	Managment	5005

Similar to the departments table, we can insert data into other tables.

```
Run                                                                    MYSQL
1  INSERT INTO students (student_id, course_id, first_name ,last_name, email,
2  phone_number,dob,admission_date,department_id,fee_status,study_semester)
3  VALUES (101,10001,"Ajit", "Kumar", "example@gmail.com", "0123456789", '2000-7-04',
4
5  INSERT INTO students VALUES
6  (102,10001,"Miccky", "Mouse", "example2@gmail.com", "0123456789", '2000-8-04', '202
7  INSERT INTO students VALUES
8  (103,10001,"Ajit", "Kumar", "example@gmail.com", "0123456789", '2000-7-04', '2022-7
9  INSERT INTO students VALUES
10 (104,10001,"Miccky", "Mouse", "example2@gmail.com", "0123456789", '2000-8-04', '202
11 INSERT INTO students VALUES
12 (105,10001,"Ajit", "Kumar", "example@gmail.com", "0123456789", '2000-7-04', '2022-7
13 INSERT INTO students VALUES
14 (201,10002,"Miccky", "Mouse", "example2@gmail.com", "0123456789", '2000-8-04', '202
```

student_id	course_id	first_name	last_name	email	phone_number	dob	admission_date	department_id	fee_status	study_semester
101	10001	Ajit	Kumar	example@gmail.com	0123456789	2000-07-04	2022-07-04	1	paid	4
102	10001	Miccky	Mouse	example2@gmail.com	0123456789	2000-08-04	2022-07-04	1	unpaid	4
103	10001	Ajit	Kumar	example@gmail.com	0123456789	2000-07-04	2022-07-04	1	paid	4
104	10001	Miccky	Mouse	example2@gmail.com	0123456789	2000-08-04	2022-07-04	1	unpaid	4
105	10001	Ajit	Kumar	example@gmail.com	0123456789	2000-07-04	2022-07-04	1	paid	4
201	10002	Miccky	Mouse	example2@gmail.com	0123456789	2000-08-04	2022-07-04	2	paid	4

2.3 Retrieve values from Table (SELECT)

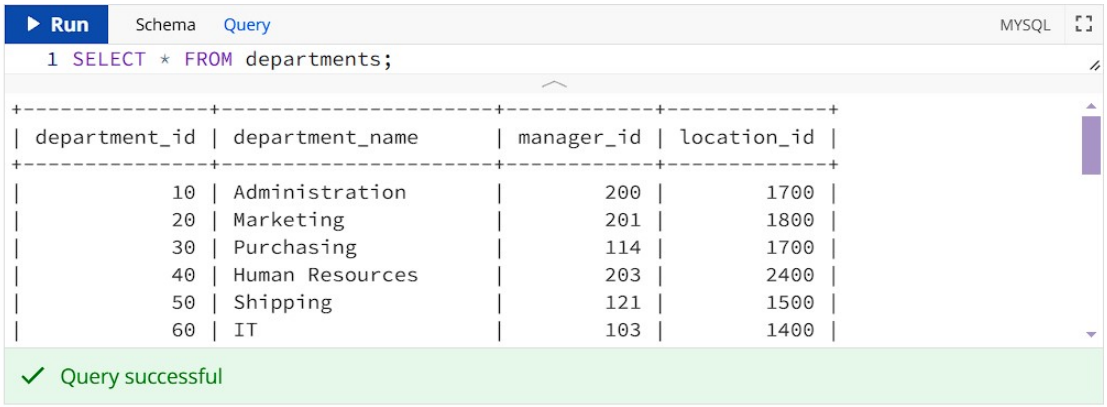
Congratulations! by now, you have to store data in your database under various tables. Storing and Accessing data from the database are two key requirements. So, in this section, we will learn how to retrieve data from the tables. Depending upon various scenarios, we will seek different parts of stored data.

Using the SELECT keyword, we can retrieve data from a database. The SELECT keyword generally follows the column/s name and table name to get the data. Further, we will learn that we can filter out our data based on various conditions and constraints while querying or executing the SELECT statement and the WHERE keyword helps us to achieve data filtration. Filtering data while accessing the table is a key requirement for Bigdata Analytics.

Selecting all columns

If we need to access all data (rows) from all columns (attributes) from a table, then we can use an asterisk (*) instead of writing all the column names. For example, to

see all values from departments, we can run the following SQL query (SELECT statement):



department_id	student_id	faculty_id	department_name	head_of_department
1	101	1001	Computer Science	1005
1	102	1002	Computer Science	1005
1	103	1003	Computer Science	1005
1	104	1004	Computer Science	1005
1	105	1005	Computer Science	1005
2	201	2001	Science	2005
2	202	2002	Science	2005
2	203	2003	Science	2005
2	204	2004	Science	2005
2	205	2005	Science	2005
2	206	2006	Science	2005
3	301	3001	Commerce	3005
4	401	4001	History	4005
5	501	5001	Managment	5005

Selecting single column

If we are only interested to see value from some selected columns (one or more), then we need to provide the name of the columns (separated by commas). For example: To know the head of the department of each department and first names of students, we can run the following queries:

```
Run  Schema  Query                                                    MYSQL
1 SELECT head_of_department FROM departments;
2
3 SELECT first_name FROM students;
```

Selecting multiple columns:

Similar to the single-column selection, we can pass multiple columns' names to get the values for these selected columns.

```
Run  Schema  Query                                                    MYSQL
1 SELECT department_name, head_of_department FROM departments;
2
3 SELECT first_name, last_name, email FROM students;
```

Conditional Selection: Where

In the above queries, we were focused on the columns selection ('*' for all or passing names of columns). However, many a time, we need to filter out rows based on one or more conditions applied to column values. We need to use an extra SQL keyword, "WHERE", for the conditional selection of rows. WHERE allows us to write multiple SQL queries and chain them as one. However, we will focus on a single sub-query with conditions for the SELECT statement.

For example, if we want to know about students who have not paid the course fee, we can use "WHERE fee_status = 'unpaid' as a subquery to the SELECT statement.

```
Run                                                                  MYSQL
1 SELECT first_name, last_name, email FROM students
2 WHERE fee_status = 'unpaid';
```

In addition to '=', we can use many other operators to apply conditions and constraints. Such as greater than (<), less than (<), LIKE, NOT etc.

Later, we will learn to use WHERE to apply conditions on multiple tables.

You can read more about various comparison operators from the below link:

SQL: Comparison Operators

(https://www.techonthenet.com/sql/comparison_operators.php)

Try it

Try selecting the first name and last name of faculty who has a salary greater than 50000.00:

2.3.1 Output Formatting (Renaming columns)

When designing the schema or ER diagram or creating the table, the focus was on the backend or database. But the result was shown to the user when we started querying the database. One can observe that the column name is not in proper English-readable format. So, SQL offers an alias to rename the columns while we execute a SELECT query. You can give an alias to any column while passing it to SELECT, the alias will be used as a new heading to that column. Even alias can be referred to in the same query too. For example, instead of showing the SELECT result with the department_id heading, we can give "Department ID" as an alias and the result with having the proper heading.

You can use the "AS" keyword to provide an alias to a column. However, you can also omit the "AS" and can just provide the new column name with a space.

Note: Care should be taken when the new column name has space, so we need to put that within the double quote.

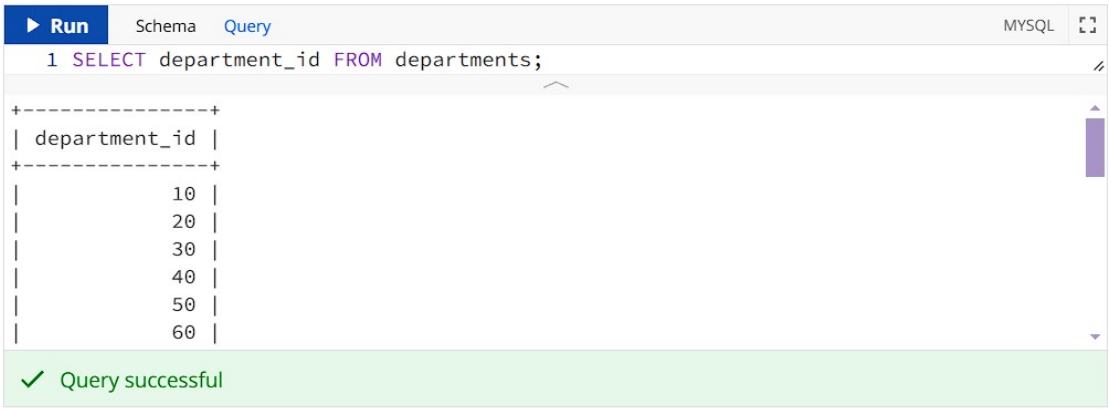
For example:

2.3.4 Removing duplicate rows

Now, we can apply selection on columns and rows and also can rename the columns. It is time to look for more options to filter data access from tables.

Getting All value (With duplicates)

A SELECT statement will default return all matching rows, even if repeated values exist. For example, the following statement will return all department_id from the "departments" table because there will be a repeated id for every department student.

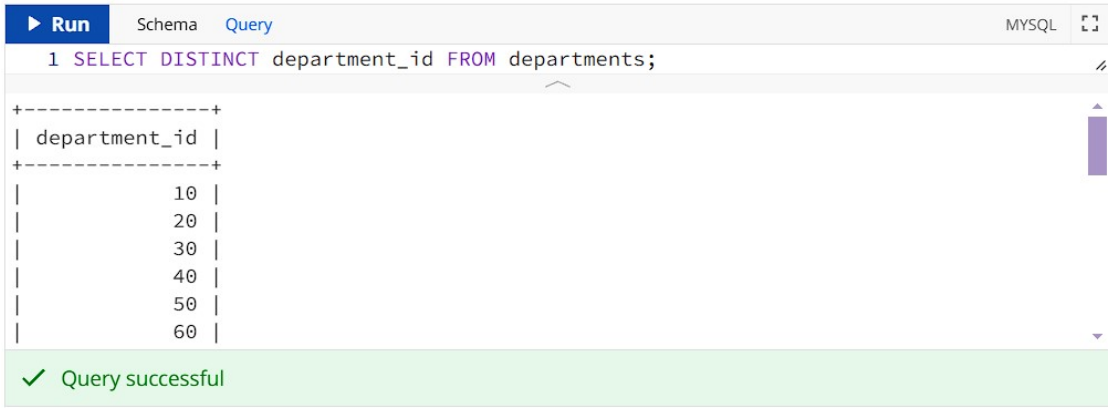


Getting Unique value (No duplicates): Single Columns

Usually, If we are only interested in knowing the ids of all the departments, we need to customize the SELECT statement to return unique rows and eliminate duplicates.

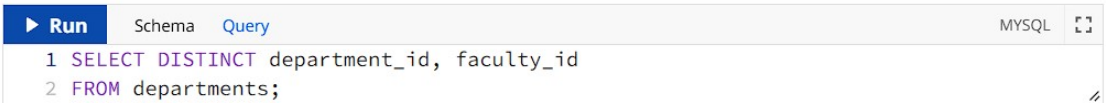
SQL has a DISTINCT keyword that lets us remove the duplicate values from the outcome of the "SELECT" statement and only shows the unique values.

So, if we modify the previous SQL query (SELECT statement) by adding DISTINCT, then the following query returns a list of unique department ids.



Getting Unique value (No duplicates): Multiple Columns

We can extend the use of the DISTINCT keyword to multiple columns; in this case, it will include the unique combinations of those columns. For example, if we want to know the faculties for each department, then we can use DISTINCT on department_id and faculty_id, which will show each unique row with department id and faculty id.



2.3.5 Sorting rows

Sorting is one of the required functions for data analytics; for example, next week, you will learn page ranking, which sorts the search result before displaying them to the user. Similarly, there are many use cases for sorting. So, SQL has a good set of sorting features.

The ORDER BY is the easiest way to sort the outcome of the SELECT statement. We can use ORDER BY to sort result in ascending or descending order, and the sorting works on single or multiple columns.

Sorting by a single column

For example, the following SQL query shows the result from the "students" table sorted based on the admission date.

Note: The ORDER BY clause should always be the last clause in your SELECT statement.

Sorting by multiple columns

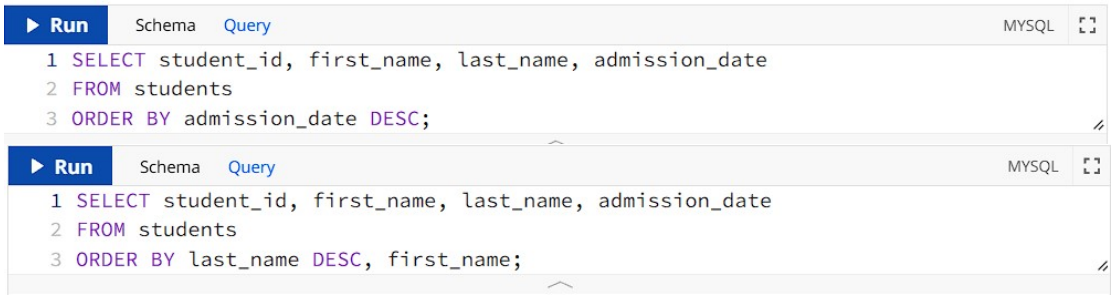
Sorting of results can be done based on multiple columns. The following query sorts rows first by last_name and then by first_name:

Sort direction

We didn't specify the soring direction in the previous two queries, so what happened? By default, SQL sorts the result in ascending order.

However, default, i.e. ascending order, is not always required, and If you want to apply to sort in descending order, you can add DESC after the column name. For example, descending order sorting will be done on admission_date and last_name in the following two queries.

Note: Notice the use of DESC after the column name; it is important to add the keyword next to the column on which sorting need to be applied.



Note: Some databases may have different default behaviour for soring, so there is an ASC keyword for forcing soring to ascending order. However, if the default is ascending order, so most of the time, you don't need to use ASC.

2.3.6 Manipulating columns

The process of data analytics involves performing many operations on the existing data to make meaningful information. So, SQL has many features that can use for data mainipulations and finding patterns.

You can manipulate columns by combining them or performing calculations. The concatenation of columns and performing arithmetic operations on column values are two important features.

Concatenating columns

We have designed our table such that first name and last name are in different columns, and every time we perform the SELECT operation, values are shown in different columns. So, if we want to display names by using first and last names, then we can combine the first_name and last_name columns by using the CONCAT() function.

```
Run  Schema  Query                                                    MYSQL
1 SELECT CONCAT(first_name, last_name)
2 FROM students;
```

You can concatenate as many columns as you like, and you can add literal values too:

```
Run  Schema  Query                                                    MYSQL
1 SELECT CONCAT("Full name: ", first_name, " ", last_name)
2 FROM students;
```

When using numeric literals, you should not use quotes. You can use single quotes inside double quotes and double quotes inside single quotes:

```
Run  Schema  Query                                                    MYSQL
1 SELECT CONCAT(first_name, "'s last name is ", '"', last_name, '"')
2 FROM students;
```

Performing calculations

In the previous section, we used concatenation on columns. However, it is not suitable for numeric values and arithmetic operations. So, SQL offers basic arithmetic operations, i.e. addition, subtraction, multiplication and division (+, -, * and /).

```
Run  Schema  Query                                                    MYSQL
1 SELECT
2     first_name,
3     last_name,
4     salary AS "Current Salary",
5     salary * 0.10 AS "Increase",
6     salary + (salary * 0.10) AS "New Salary"
7 FROM faculties;
```

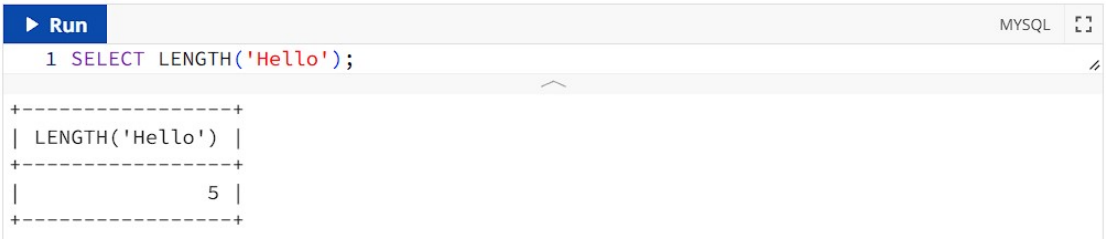
2.3.7 String functions

During the data analytics process, we often need to work with text data (for natural language processing tasks), or some part of data is text like name, address etc., In just cases, we need to perform many pre-processing tasks to text, for example,

converting all text to lower-case, removing special symbols etc. Now, doing it manually or with simple SQL options is not possible. However, SQL has various string functions that can directly apply to data and can be used during SQL queries. In this section, we will learn some key string functions available in SQL.

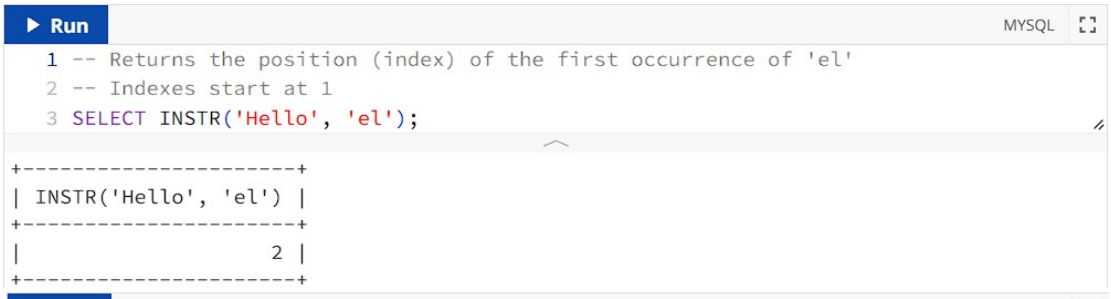
In many of the examples below a SELECT statement is executed without any reference to a table. This is perfectly fine, as long as you are just working with non-database values. It's a bit like using SQL as a calculator.

Length of a string

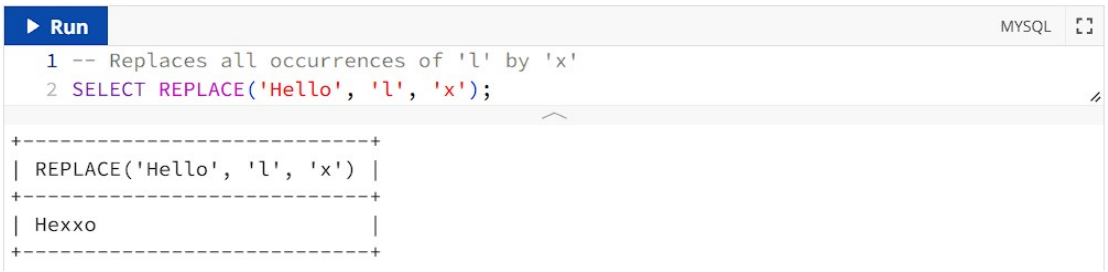


Change Case

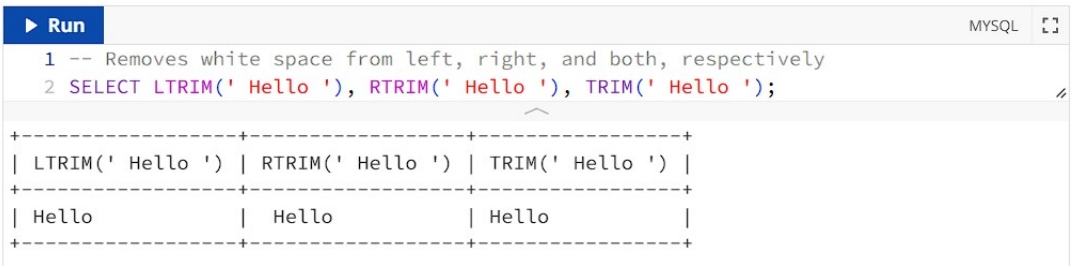
Finding substring



Replace String



Trimming



Many more string functions are available, and you can explore them from the links below.

Readings

1. Free Learning Platform For Better Future. (n.d.). SQL server string functions. https://www.javatpoint.com/sql-server-string-functions
2. GeeksforGeeks. (2019). SQL | String functions. https://www.geeksforgeeks.org/sql-string-functions/

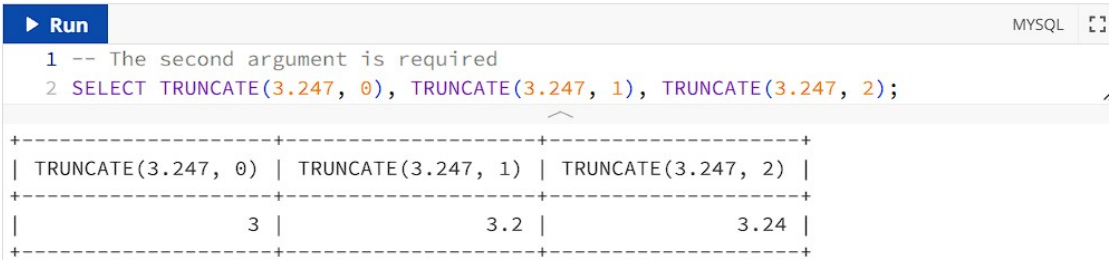
2.3.8 Numeric functions

There are various functions that you can use to work with numbers.

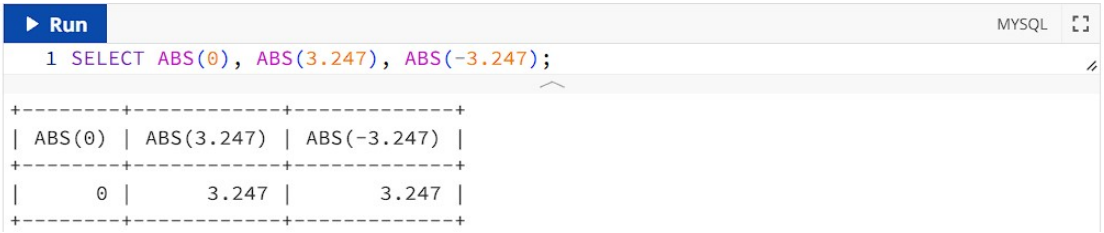
To round a number to a given number of decimal places:

To round a number up or down to the nearest integer:

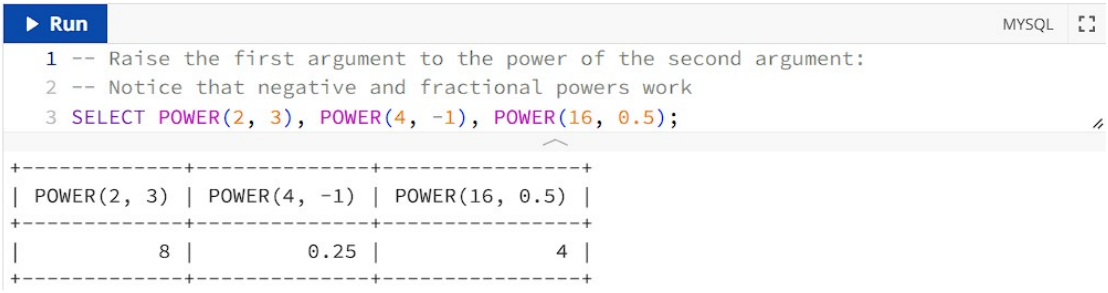
To truncate a number to a given number of decimal places:



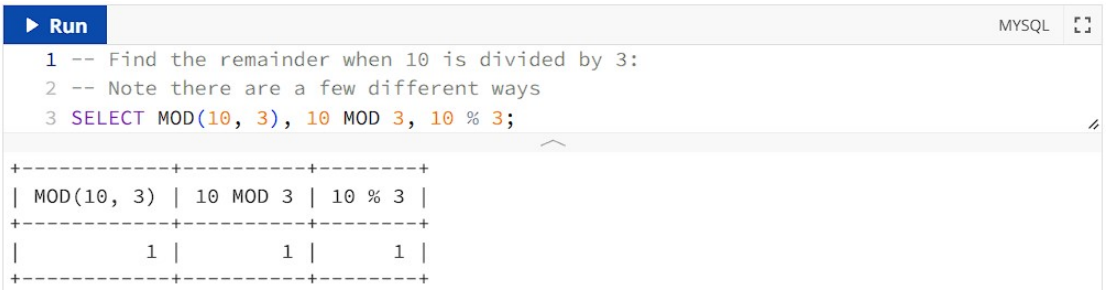
To find the absolute value of a number:



To raise one number to the power of another:



To find the remainder when one number is divided by another:



More functions

To learn a complete list of MySQL numeric functions, visit the following sites:

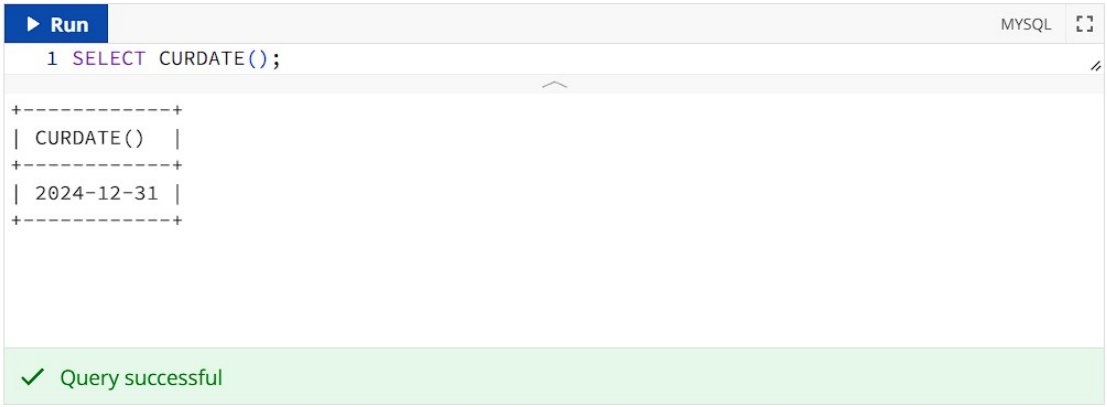
Readings

1. MySQL. (n.d.) 14.6 Numeric functions and operators.
   https://dev.mysql.com/doc/refman/8.0/en/numeric-functions.html
2. W3Schools. (n.d.). MySQL functions.
   https://www.w3schools.com/sql/sql_ref_mysql.asp

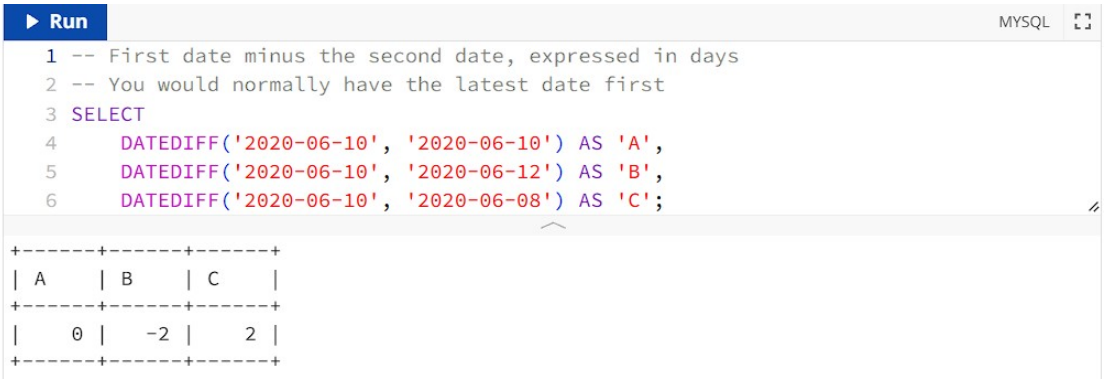
2.3.9 Date functions

The default format of a date in MySQL is YYYY-MM-DD. Different SQL languages handle dates differently, so you have to check the date format, function and how the data is handled in that language. In Oracle, it is DD-MON-RR, where "MON" stands for a three-letter month name and "RR" stands for a two-digit year number.

To get the current date:



To get the number of days between two dates:



To get the various parts of a date:

To format a date in a certain way:

What do all those symbols mean? Here is a good reference

(https://www.w3schools.com/sql/func_mysql_date_format.asp).

More functions

To learn a full list of MySQL date functions, visit the following links:

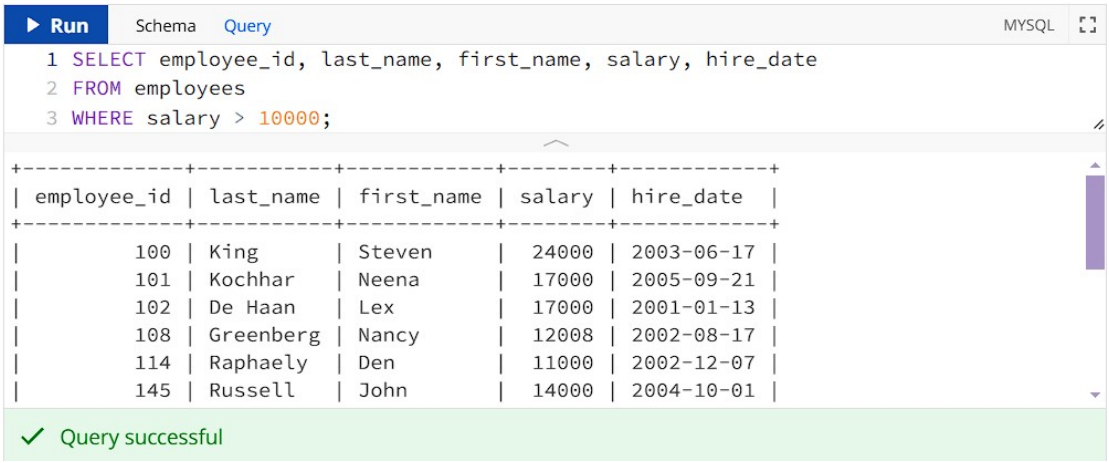
Readings

1. MySQL. (n.d.) 14.7 Date and time functions.
   https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html
2. W3Schools. (n.d.-a). MySQL DATE_FORMAT() function.
   https://www.w3schools.com/sql/func_mysql_date_format.asp
3. W3Schools. (n.d.-b). MySQL functions.
   https://www.w3schools.com/sql/sql_ref_mysql.asp

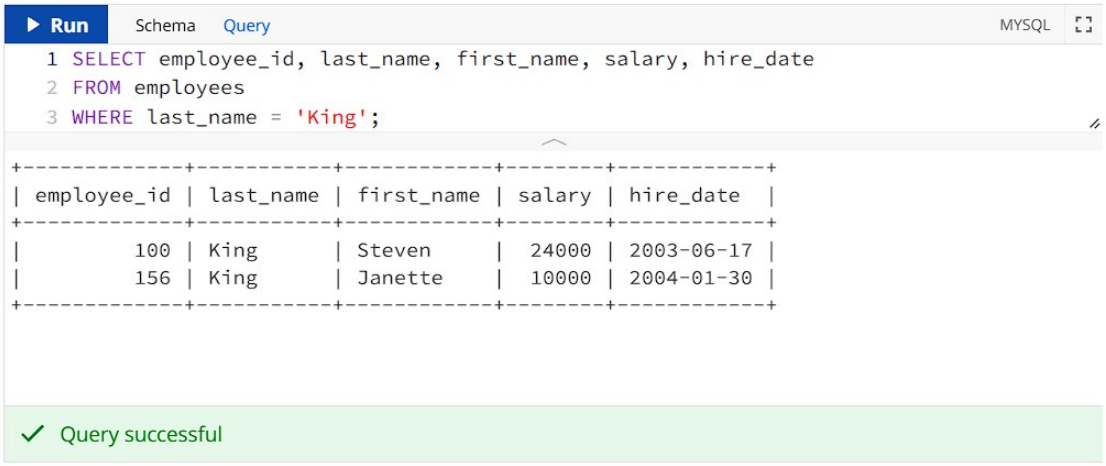
Application activity: Test of SQL syntax

You can select only certain rows by adding a WHERE clause to your query.

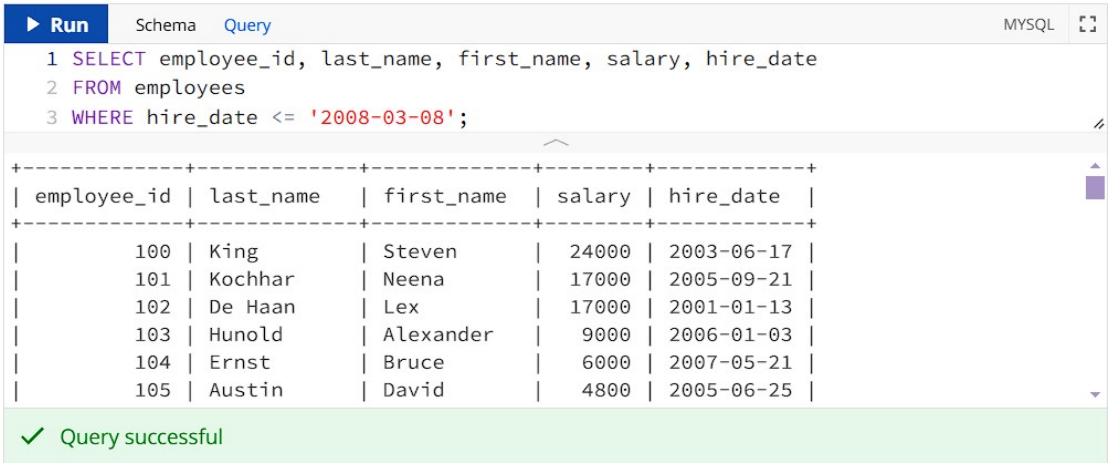
The following query selects employees whose salary is greater than $10,000.



The following query selects employees whose last name is "King".

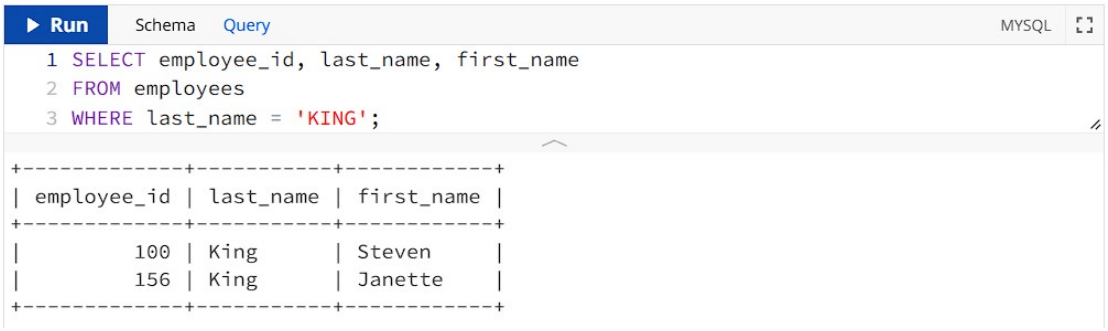


The following query selects employees who were hired on or before 8th March 2008.



When filtering string or date columns you should use quotes around the filtering value, but not for numeric columns. For dates, make sure you are aware of the format in which dates are stored. In MySQL the default is YYYY-MM-DD (e.g. 2011-01-16). In Oracle it is DD-MON-YY (e.g. 16-JAN-11).

Filtering strings is NOT case sensitive in MySQL - filtering by 'King' produces the same results as filtering by 'KING'. Be aware that in some versions of SQL the filtering is case sensitive.



2.4 SQL Schema Languages

What is Schema in SQL?

In a SQL database, a schema is a list of logical structures of data.

A database schema is like a skeletal structure representing a logical view of a whole database. It devises all the constraints applied to the data in a particular database. Whenever organizations engage in data modeling, it leads to a schema.

A database user owns the schema, which has the same name as the **database manager**.

As of SQL Server 2005, a schema is an individual entity (container of objects) distinct from the user who constructs the object.

In other words, schemas are similar to separate namespaces or containers used to handle database files. Schemas may be assigned security permissions, making them an effective method for distinguishing and defending database objects based on user

access privileges. It increases the database's stability for security-related management.

What is a database schema?

Watch the following 4-minute 46-second video to learn about database schema.

What is a database schema?

https://youtu.be/3BZz8R7mqu0

Source: (Intricity101, 2019)

A database schema is (generally) broadly divided into two categories:

1. Physical database schema: It defines how the data-like files are stored.

2. Logical database schema: It describes all the logical constraints -- including integrity, tables and views -- applied to the stored data.

Advantages of using schema

The following are some of the main advantages of using a schema in SQL:

- A SQL schema can be easily transferred to another user.
- A schema may be shared by several users.
- It enables you to transfer database objects between schemas.
- We gain greater power over the access and protection of database objects.
- A user can be removed without removing the database items that are connected with the user.
- Database objects can be grouped into logical groups using schemas. This is advantageous when several people are collaborating on the same database program and the architecture team needs to keep the database tables' credibility.
- Since a schema allows for the logical aggregation of database objects, it can assist us in cases where the database object name is the same but falls into a separate logical category.

Visit the following links to learn more about Schema.

Readings

1. Awati R. & Zola, A. (2021). Schema. TechTarget. https://www.techtarget.com/searchdatamanagement/definition/schema
2. dbForge Team. (2021). Understanding a SQL Schema. Devart. https://blog.devart.com/understanding-a-sql-schema.html
3. Ravikiran, A.S. (2023). What is a Schema in SQL and advantages of using Schema. Simplilearn. https://www.simplilearn.com/tutorials/sql-tutorial/schema-in-sql

What is the SQL language?

Watch the following 5-minute 40-second video to learn what is the SQL language.

What is the SQL Language?

https://youtu.be/Bn-nTzkhEpQ

Source: (CBT Nuggets, 2019)

2.4.1 How to Create a Schema?

How to Create a Schema?

Creating schemas can be useful when objects have circular references, such as when we need to construct two tables, one with a foreign key referencing the other table. You can create a schema in SQL by following the below syntax.

Syntax

```
CREATE SCHEMA [schema_name] [AUTHORIZATION owner_name]
[DEFAULT CHARACTER SET char_set_name]
[PATH schema_name[, ...]]
[ ANSI CREATE statements [...] ]
[ ANSI GRANT statements [...] ];
```

2.4.2 How to Alter a Schema?

How to Alter a Schema?

The ALTER SCHEMA statement is used to rename a schema or specify a new owner, who must be a pre-existing database user.

Syntax for Altering a Schema:

```
ALTER SCHEMA schema_name [RENAME TO new_schema_name] [OWNER TO new_user_name]
```

Here, `new_schema_name` refers to the name to which you want to rename the existing schema and new_user_name refers to the new owner of the schema.

Example:

Suppose we want to rename the previously created schema- STUDENT as `STUDENT_DETAILS` and pass the ownership to new user `DAVID`. The following query will result in the desired result.

```
ALTER SCHEMA STUDENT [RENAME TO STUDENT_DETAILS] [OWNER TO DAVID]
```

2.4.3 How to Drop a Schema?

How to Drop a Schema?

The `DROP SCHEMA` in SQL is used to delete all tables present in that particular schema.

Syntax:

```
DROP SCHEMA <schema name>
```

Example:

If you want to delete the schema `STUDENT_DETAILS`, then use the following SQL query.

```
MYSQL
DROP SCHEMA STUDENT_DETAILS
```

2.4.4 Hands-on with SQL for data Modelling and Query

1. Create a Database using SQL. (choose the name of your database)
2. Create two tables naming customers and products with the following fields for each table
    a) Customers
    - customer_id: numeric
    - customer_name:string
    - date_of_birth:date
    - gender: string
    - address: string

b) Products

- product_id: numeric
- product_name: string
- cost: floating
- manufacture_year: numeric
- manufacture_country: string
- total_count: numeric

3. Write an insert SQL query to add a few (at least five) rows to both tables.

4. Create a table name purchase with the following fields:

- customer_id
- product_id
- Date

5. Create an ER diagram/schema diagram for the above database.

2.4.5 Big Data analysis: Flight Data

Flights Dataset Overview

Download the flight data, then answer the basic questions concerning the dataset:

1. What is the best time to fly to minimise flight delays?
2. Which aircraft types are most prone to delays, and what Is the impact of aircraft age on delays?
3. How often do delays cascade and lead to further delays in other flights?
4. What Is the impact of airlines on delay frequency?

2.4.6 NoSQL

What is NoSQL?

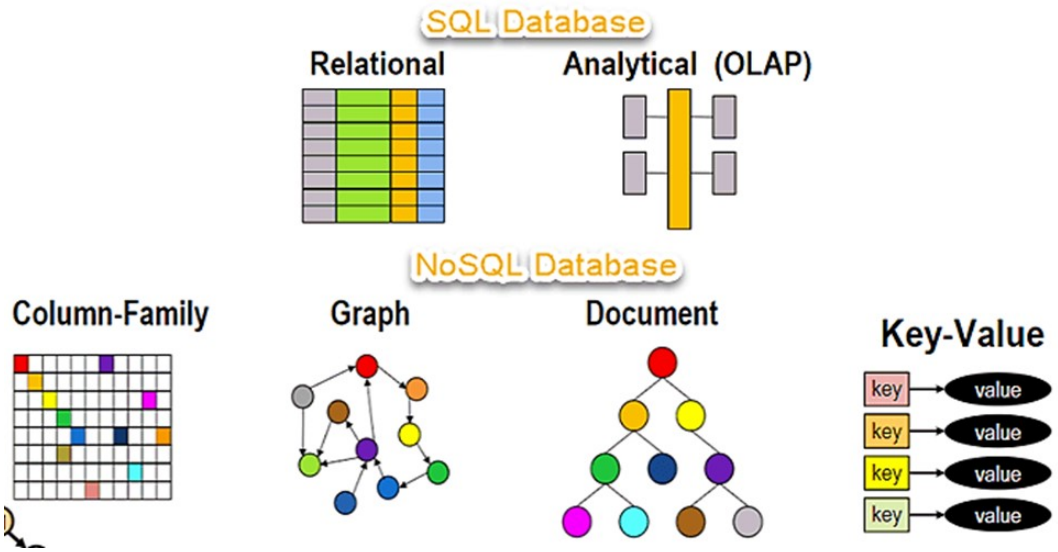
NoSQL Database is a non-relational Data Management System, that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs.

NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.

NoSQL database stands for “Not Only SQL” or “Not SQL.” Though a better term would be “NoREL”, NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

Let’s understand about NoSQL with a diagram in this NoSQL database tutorial:



Source: SQL and NoSQL Database (Twain, 2024)

Watch the following 2-minute video to learn further about What is NoSQL.

What is NoSQL? Database tutorial

https://youtu.be/qUV2j3XBRHc

Source: (Guru99, 2013)

2.4.7 NoSQL tutorial

What is NoSQL?

NoSQL Database, which is also known as “Not Only SQL” is an alternative to SQL database.

Refer to the site below to find out more about NoSQL Tutorial.

Key reading

- Twain, S. (2024). NoSQL tutorial: What is, types of NoSQL databases & example. Guru99. https://www.guru99.com/nosql-tutorial.html

Further readings

To learn further about SQL Introduction and Syntax, visit the following links:

For a gentle guide to SQL:

- W3Schools. (n.d.). SQL tutorial. https://www.w3schools.com/sql/default.asp

A complete tutorial on SQL:

- Tutorialspoint. (n.d.). SQL tutorial. https://www.tutorialspoint.com/sql/index.htm

If you would like to see a more theoretical discussion, you could look at **Part 10** of the following textbook:

- Elmasri, R. & Navathe, S.B. (2017). Fundamentals of database systems (7th ed.). Pearson Education Limited. https://research.ebsco.com/linkprocessor/plink?id=a95816b6-9a13-324e-bed9-74c9b817b1b6

SQL Schema:

- Ravikiran, A.S. (2023). What is a Schema in SQL and advantages of using Schema. Simplilearn. https://www.simplilearn.com/tutorials/sql-tutorial/schema-in-sql

NoSQL is a database technology driven by Cloud Computing, Web, Big Data and Big Users. Find out more about NoSQL in the following article:

- ProjectPro. (2024). NoSQL vs SQL- 4 reasons why NoSQL is better for big data applications. https://www.projectpro.io/article/nosql-vs-sql-4-reasons-why-nosql-is-better-for-big-data-applications/86

2.4.8 Discussion: How SQL can be useful in Big Data

Time: 30 minutes

Purpose: To discuss how SQL can be useful in Big Data

Task: Visit the following links to learn how SQL can be useful in Big data and share your responses to the following question in the discussion forum.

Readings

- Dewani, R. (2020). 8 SQL techniques to perform data analysis for analytics and data science. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/07/8-sql-techniques-data-analysis-analytics-data-science/
- Neelu. (2024). SQL for data science: A beginner's guide! Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/sql-for-data-science-a-beginners-guide/
- Tutorialspoint. (n.d.). Big data analytics - Introduction to SQL. https://www.tutorialspoint.com/big_data_analytics/introduction_to_sql.htm

Question: Share your takeaways on how SQL can be useful in Big Data.