Table of Contents

[SQL basics 2](#_Toc108518145)

[SELECT statement 2](#_Toc108518146)

[COUNT, DISTINCT, and LIMIT 2](#_Toc108518147)

[INSERT 3](#_Toc108518148)

[UPDATE and DELETE 3](#_Toc108518149)

[Some more features 4](#_Toc108518150)

[Introduction to Relational Databases and Tables 4](#_Toc108518151)

[Relational Database concepts 4](#_Toc108518152)

[Types of SQL statements 5](#_Toc108518153)

[CREATE TABLE 6](#_Toc108518154)

[ALTER, DROP, and TRUNCATE 7](#_Toc108518155)

[Intermediate SQL 8](#_Toc108518156)

[Using String patterns, ranges, and sets 8](#_Toc108518157)

[Sorting result sets 8](#_Toc108518158)

[Grouping result sets 9](#_Toc108518159)

[Assignment 10](#_Toc108518160)

[Built-in database functions 12](#_Toc108518161)

[Date and Time functions 13](#_Toc108518162)

[Assignment 13](#_Toc108518163)

[Sub-queries and nested selects 15](#_Toc108518164)

[Accessing Databases using Python 16](#_Toc108518165)

[Writing code using DB-API 16](#_Toc108518166)

SQL

# SQL basics

## SELECT statement

Select statement is used to retrieve the rows from a database with some conditions. Some examples:

* **SELECT col1, col2, … FROM table;**
* **SELECT \* FROM table;** # retrieves all rows from a table.
* **SELECT \* FROM table**

**WHERE country\_code = ‘CA’;** # single quotes are used!

## COUNT, DISTINCT, and LIMIT

COUNT( ) – is a built-in function that retrieves the number of rows matching the query criteria.

* Get the number of rows in a table:

**SELECT COUNT( \* ) FROM table;**

* Retrieve the number of rows where the medal recipient is from Canada:

**SELECT COUNT( country ) FROM medals**

**WHERE country = ‘Canada’;**

DISTINCT is used to remove duplicate values from the resulting set.

* Retrieve unique values in a column:

**SELECT DISTINCT col FROM table;**

* List unique countries that received gold medals:

**SELECT DISTINCT country FROM medals**

**WHERE medaltype = ‘Gold’;**

LIMIT is a function that restricts the number of rows retrieved from the database.

* Retrieve the first 10 rows from a table:

**SELECT \* FROM table LIMIT 10;**

* Retrieve 5 rows in the MEDALS table for a particular year:

**SELECT \* FROM medals**

**WHERE year = 2018 LIMIT 5;**

## INSERT

Insert statement is used to add new data to the table by adding rows to it.

* Syntax of the statement:

**INSERT INTO table**

**<( col1, col2, … )>**

**VALUES**

**( val1, val2, … );**

* Adding new row to the Authors table:

**INSERT INTO Authors**

**( id, lastname, firstname, email, city, country )**

**VALUES**

**( ‘A1’, ‘Chong’, ‘Raul’, ‘ex@gmail.com’, ‘Toronto’, ‘CA’ );**

* Several columns can also be inserted at a time:

**INSERT INTO Authors**

**( id, lastname, firstname, email, city, country )**

**VALUES**

**( ‘A1’, ‘Chong’, ‘Raul’, ‘ex1@gmail.com’, ‘Toronto’, ‘CA’ )**

**( ‘A2’, ‘Ahuja’, ‘Rav’, ‘ex2@gmail.com’, ‘Toronto’, ‘CA’ );**

## UPDATE and DELETE

Update is a function that can alter (change) the data in a table.

* Syntax of the statement:

**UPDATE table**

**SET [col\_name = new\_val]**

**<WHERE condition>;**

* Updating the previously created table:

**UPDATE Authors**

**SET lastname = ‘Me’**

**firstname = ‘Me’**

**WHERE id = ‘A2’;** # if where clause is not specified, all rows of the

# table will be updated.

Delete statement is used whenever there is a need to remove 1 or more rows from a table.

* Syntax of the statement:

**DELETE FROM table**

**<WHERE condition>;**

* Deleting the rows from the previous Authors table:

**DELETE FROM Authors**

**WHERE id IN ( ‘A2’, ‘A3’ );** # if the where clause is not specified,

# all rows will be removed.

## Some more features

* Getting the number of unique directors of movies released in the 21st century:

**SELECT COUNT( DISTINCT Director ) FROM Films**

**WHERE ReleaseYear >= 2001;**

* Getting only 3 rows of data after the first 5 rows (skip 5 rows and get 3):

**SELECT \* FROM table LIMIT 3 OFFSET 5;**

# Introduction to Relational Databases and Tables

## Relational Database concepts

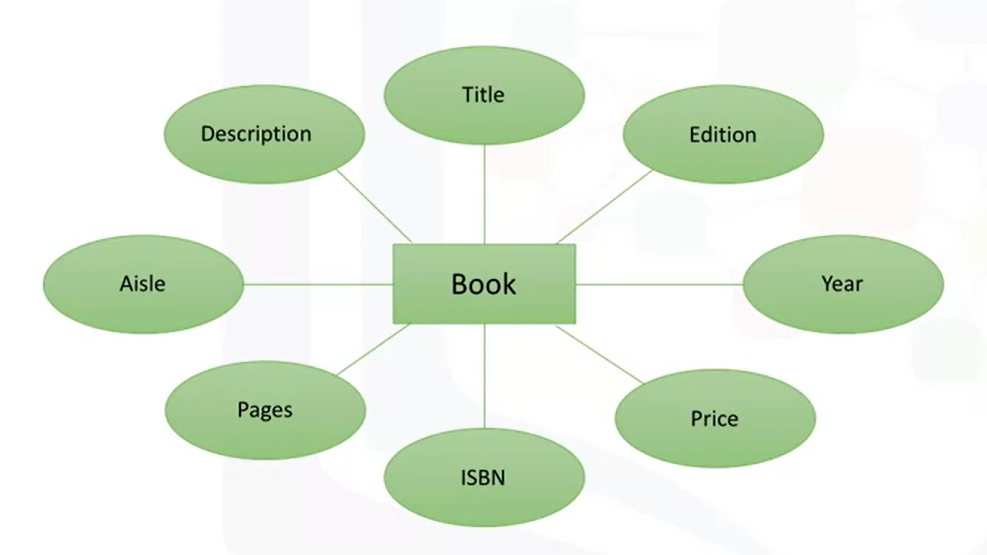
Relational model

* Is the most used data model,
* because it allows for data independence.
* Data is stored in entities called tables.

Entity-Relationship model proposes thinking of a database as a collection of entities. ER is not used as a model on its own, but rather is used as a tool to design relational databases.

* In the ER model entities are objects that exist independently of any other entities in the database.
* **Entity** is a table of a database.
* Entities have **attributes** which are the elements that characterize the entity.

In the following image, “Book” is an example of the entity. Its attributes are “Title”, “Edition”, “Year”, and so on.



An entity becomes a table in the relational database, whereas its attributes become the columns of that table.

The **primary key** of a relational table uniquely identifies each row in a table, thus preventing duplication of data and providing a way of defining relationships between tables in a database.

Tables can also contain **foreign keys** – these are the primary keys defined in other tables, different from the one where that primary key is defined. Foreign keys enable to create a link between the tables.

## Types of SQL statements

Statement types:

1. **Data Definition Language** statements (DDL)
   * 1. Used to define, change, or drop data.
     2. Common DDL statements:
        + 1. CREATE – to create tables and define their columns.
          2. ALTER – to alter tables, add and drop columns, and modify data types.
          3. TRUNCATE – to delete data in a table, but not the table itself.
          4. DROP – to delete tables.
2. **Data Manipulation Language** statements (DML)
   * 1. Used to read and modify data in tables.
     2. CRUD operations (Create, Read, Update, and Delete rows).
     3. Common DML statements:
        + 1. INSERT – to insert rows to a table.
          2. SELECT – to retrieve rows from a table.
          3. UPDATE – to edit rows in a table.
          4. DELETE – to remove rows from a table.

## CREATE TABLE

This is the most common DDL statement used to create a table in a relational database.

* Syntax of the statement:

**CREATE TABLE table (**

**col1 datatype optional\_parameter,**

**col2 datatype,**

**…**

**coln datatype**

**);**

* Creating a table for provinces in Canada:

**CREATE TABLE provinces (**

**id char( 2 ) PRIMARY KEY NOT NULL,**

**name varchar( 24 )**

**);**

* Creating the author table:

**CREATE TABLE Authors (**

**author\_id char( 2 ) PRIMARY KEY NOT NULL,**

**lastname varchar( 15 ) NOT NULL,**

**firstname varchar( 15 ) NOT NULL,**

**email varchar( 40 )**

**);**

## ALTER, DROP, and TRUNCATE

ALTER TABLE statement is used to add or remove columns from a table, to modify the datatype of columns, to add or remove keys, and to add or remove constraints.

* Syntax of the statement:

**ALTER TABLE table**

**ADD COLUMN col1 datatype**

**…**

**ADD COLUMN coln datatype;**

* Adding a telephone number column to the author table:

**ALTER TABLE Authors**

**ADD COLUMN phone\_number BIGINT;**

* Changing the datatype of a column:

**ALTER TABLE Authors**

**ALTER COLUMN phone\_number SET DATA TYPE char( 20 );**

* Removing a column:

**ALTER TABLE Authors**

**DROP COLUMN phone\_number;**

* Renaming a column:

**ALTER TABLE Authors**

**RENAME COLUMN id TO unique\_id;**

DROP TABLE statement is used to delete the whole table from a database.

* Syntax of the statement:

**DROP TABLE table;**

* Deleting the author table:

**DROP TABLE Authors;**

TRUNCATE TABLE statement is used to delete the whole content of a table, but not the table itself.

* Syntax of the statement:

**TRUNCATE TABLE table**

**IMMEDIATE;**

* Deleting all rows from the author table:

**TRUNCATE TABLE Authors**

**IMMEDIATE;**

# Intermediate SQL

## Using String patterns, ranges, and sets

We can retrieve some rows from a table based on a WHERE clause. However, this statement requires a predicate and what if I do not know the exact values of rows that I want to retrieve, but I know that the values start with some letter? This is when I can use the **LIKE statement** with string patterns.

* Syntax of the statement:

**WHERE col LIKE <string pattern>;**

* Getting rows where author’s name starts with R:

**SELECT id, title FROM Book**

**WHERE author\_name LIKE R%;**

The statement **BETWEEN AND** is used when I want to get a range from a table. For example, I only want to get books that are more than 290 and less than 300 pages long:

* I could write it like this:

**SELECT title, pages FROM Book**

**WHERE pages >= 290 AND pages <= 300;**

* But the following statement is easier:

**SELECT title, pages FROM Book**

**WHERE pages BETWEEN 290 AND 300;**

The **IN** operator allows us to specify a set of values in a WHERE clause to compare against the list of expressions.

* Getting only authors that are from Australia, Brazil, or Canada:

**SELECT firstname, lastname, country FROM Authors**

**WHERE country IN ( ‘AU’, ‘BR’, ‘CA’ );**

## Sorting result sets

I can sort the resultant set of a select statement using the **ORDER BY** clause, which, by default, sorts in ascending order. I also need to specify the column by which I want to sort the result set.

* Sorting in an ascending order:

**SELECT title FROM Book**

**ORDER BY title;**

* Sorting in a descending order:

**SELECT title FROM Book**

**ORDER BY title DESC;**

* Indicating the column sequence number when sorting the result:

**SELECT title, pages FROM Book**

**ORDER BY 2;** # sorts by the 2nd specified column (pages).

## Grouping result sets

Suppose there is a table with 20 rows of different authors who come from 6 distinct countries. What if I want to know how many authors come from the same country? I can use the **GROUP BY** clause in this case.

* The above example:

**SELECT country, COUNT( country ) FROM Authors**

**GROUP BY country;**

[out]: Table

Description automatically generated

* **SELECT country, COUNT( country ) as Count FROM Authors**

**GROUP BY country;**

[out]: Table

Description automatically generated

After grouping the resultant set from a table, I can further restrict it using some conditions with the **HAVING** clause. For instance, I want to check if there are more than 4 authors from the same country:

* The above example:

**SELECT country, COUNT( country ) as Count FROM Authors**

**GROUP BY country**

**HAVING COUNT( country ) > 4;**

HAVING clause works only with the GROUP BY clause.

### Assignment

Table

Description automatically generated

1. Retrieve all employees whose address is in Elgin, IL.

**SELECT \* FROM employees**

**WHERE address LIKE ‘%Elgin,IL%’;**

1. Retrieve all employees who were born during the 1970’s.

**SELECT \* FROM employees**

**WHERE b\_date LIKE ‘197%’;**

1. Retrieve all employees in department 5 whose salary is between 60000 and 70000.

**SELECT \* FROM employees**

**WHERE ( salary BETWEEN 60000 AND 70000 ) AND dep\_id = 5;**

1. Retrieve a list of employees ordered by department ID.

**SELECT f\_name, l\_name, dep\_id FROM employees**

**ORDER BY dep\_id;**

1. Retrieve a list of employees ordered in descending order by department ID and within each department ordered alphabetically in descending order by last name.

**SELECT f\_name, l\_name, dep\_id FROM employees**

**ORDER BY dep\_id DESC, l\_name DESC;**

1. In problem above, use department name instead of department ID. Retrieve a list of employees ordered by department name, and within each department ordered alphabetically in descending order by last name.

**SELECT D.dep\_name, E.f\_name, E.l\_name**

**FROM employees as E, departments as D**

**WHERE E.dep\_id = D.dept\_id\_dep**

**ORDER BY D.dep\_name, E.l\_name DESC;**

1. For each department ID retrieve the number of employees in the department.

**SELECT dep\_id, COUNT( \* ) as Count FROM employees**

**GROUP BY dep\_id;**

1. For each department retrieve the number of employees in the department, and the average employee salary in the department.

**SELECT dep\_id, COUNT( \* ), AVG( salary ) FROM employees**

**GROUP BY dep\_id;**

1. Label the computed columns in the result set of the previous problem as NUM\_EMPLOYEES and AVG\_SALARY.

**SELECT dep\_id, COUNT( \* ) as ‘NUM\_EMPLOYEES’, AVG( salary ) as ‘AVG\_SALARY’**

**FROM employees**

**GROUP BY dep\_id;**

1. In the previous problem, order the result set by Average Salary.

**SELECT dep\_id, COUNT( \* ) as ‘NUM\_EMPLOYEES’, AVG( salary ) as ‘AVG\_SALARY’**

**FROM employees**

**GROUP BY dep\_id**

**ORDER BY AVG\_SALARY;**

1. In the previous problem, limit the result to departments with fewer than 4 employees.

**SELECT dep\_id, COUNT( \* ) as ‘NUM\_EMPLOYEES’, AVG( salary ) as ‘AVG\_SALARY’**

**FROM employees**

**GROUP BY dep\_id**

**HAVING COUNT( \* ) < 4**

**ORDER BY AVG\_SALARY;**

## Built-in database functions

SUM( col ) – adds up all the values in the specified column.

* Example:

**SELECT SUM( cost ) as Sum\_of\_cost FROM petrescue;**

MIN( col ) – returns the lowest value of the specified column.

MAX( col ) – returns the greatest value of the specified column.

These aggregate functions can also be used with a subset of data instead of an entire column:

* Example:

**SELECT MIN( id ) FROM petrescue**

**WHERE animal = ‘dog’;**

Mathematical operations can be performed between columns.

* Calculating the average cost per ‘dog’:

**SELECT AVG( cost / quantity ) FROM petrescue**

**WHERE animal = ‘dog’;**

ROUND( col ) – rounds the values in the specified column to the nearest integer.

LENGTH( col ) – retrieves the length of each value in the specified column.

UCASE( col ) – retrieves the upper-case version of the values in the column.

LCASE( col ) – returns the lower-case version of the values in the column.

* Using functions in the WHERE clause:

**SELECT \* FROM petrescue**

**WHERE LCASE( animal ) = ‘cat’;**

## Date and Time functions

There are built-in functions that operate with dates and times in SQL. These include:

DAY( col ) – returns the day number.

MONTH( col ) – returns the month number.

YEAR( col ) – returns the year number.

CURRENT\_DATE – returns the current date.

CURRENT\_TIME – returns the current time.

### Assignment

1. Enter a function that calculates the total cost of all animal rescues in the PETRESCUE table.

**SELECT SUM( cost ) FROM petrescue;**

1. Enter a function that displays the total cost of all animal rescues in the PETRESCUE table in a column called SUM\_OF\_COST.

**SELECT SUM( cost ) as Sum\_of\_cost FROM petrescue;**

1. Enter a function that displays the maximum quantity of animals rescued.

**SELECT MAX( quantity ) FROM petrescue;**

1. Enter a function that displays the average cost of animals rescued.

**SELECT AVG( cost ) FROM petrescue;**

1. Enter a function that displays the average cost of rescuing a dog.

**SELECT AVG( cost / quantity ) FROM petrescue**

**WHERE animal = ‘dog’;**

1. Enter a function that displays the rounded cost of each rescue.

**SELECT ROUND( cost ) FROM petrescue;**

1. Enter a function that displays the length of each animal name.

**SELECT LENGTH( animal ) FROM petrescue;**

1. Enter a function that displays the animal name in each rescue in uppercase.

**SELECT UCASE( animal ) FROM petrescue;**

1. Enter a function that displays the animal name in each rescue in uppercase without duplications.

**SELECT DISTINCT( UCASE( animal ) ) FROM petrescue;**

1. Enter a query that displays all the columns from the PETRESCUE table, where the animal(s) rescued are cats. Use cat in lower case in the query.

**SELECT \* FROM petrescue**

**WHERE LCASE( animal ) = ‘cat’;**

1. Enter a function that displays the day of the month when cats have been rescued.

**SELECT DAY( date ) FROM petrescue**

**WHERE LCASE( animal ) = ‘cat’;**

1. Enter a function that displays the number of rescues on the 5th month.

**SELECT SUM( quantity ) FROM petrescue**

**WHERE MONTH( date ) = ‘05’;**

1. Enter a function that displays the number of rescues on the 14th day of the month.

**SELECT SUM( quantity ) FROM petrescue**

**WHERE DAY( date ) = ‘14’;**

1. Animals rescued should see the vet within three days of arrivals. Enter a function that displays the third day from each rescue.

**SELECT ( date + 3 DAYS ) FROM petrescue;**

1. Enter a function that displays the length of time the animals have been rescued; the difference between today’s date and the rescue date.

**SELECT ( CURRENT\_DATE – date ) FROM petrescue;**

## Sub-queries and nested selects

Suppose I want to get only the employees who earn more than the average salary. Then I need the sub-query.

* The above example:

**SELECT id, f\_name, l\_name, salary FROM employees**

**WHERE salary > ( SELECT AVG( salary ) FROM employees );**

* Substituting the column name with a sub-query:

**SELECT id, salary, ( SELECT AVG( salary ) FROM employees )**

**as Avg\_salary FROM employees;**

Suppose, the following two tables are given:

Table

Description automatically generated

* Retrieving only the employee records that correspond to departments in the DEPARTMENTS table:

**SELECT \* FROM employees**

**WHERE dep\_id IN ( SELECT dept\_id\_dep FROM departments );**

* Retrieving only the list of employees from a specific location:

**SELECT \* FROM employees**

**WHERE dep\_id IN ( SELECT dept\_id\_dep FROM departments**

**WHERE loc\_id = ‘L0002’ );**

* Retrieving the department ID and name for employees who earn more than 70000:

**SELECT dept\_id\_dep, dep\_name FROM departments**

**WHERE dept\_id\_dep IN ( SELECT dep\_id FROM employees**

**WHERE salary > 70000 );**

Python with SQL

# Accessing Databases using Python

## Writing code using DB-API

DB-API is a Python’s standard API for accessing relational databases. The Python code connects to the database using DB-API calls.

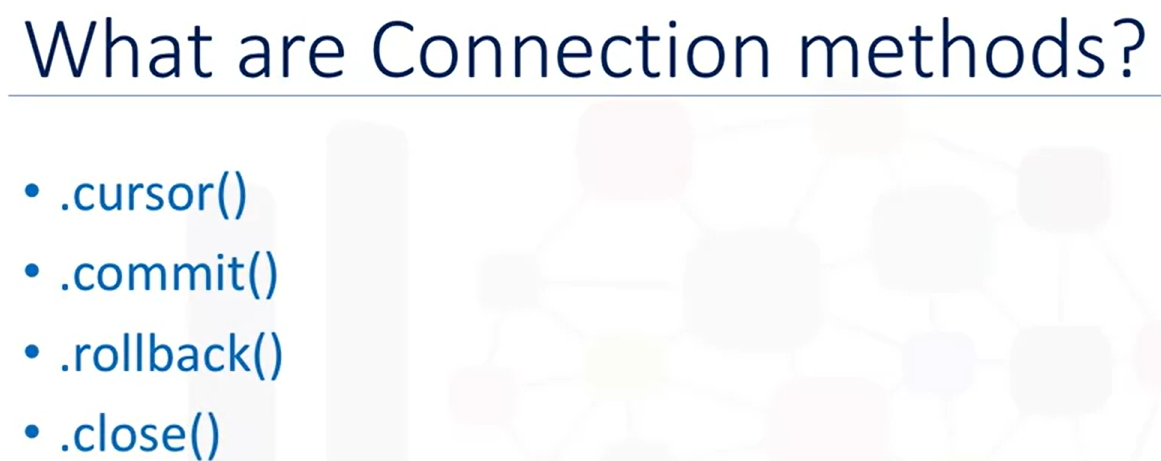
The 2 main concepts in the Python DB-API are **connection** and **query objects**.

**Connection objects are used for:**

* Database connections
* Transactions management

**Cursor objects are used for:**

* Database queries
* Scrolling through result set
* Retrieving results



Text

Description automatically generated with medium confidence

Sample code:

|  |
| --- |
| from dbmodule import connect  Connection = connect( ‘database\_name’, ‘username’, ‘password’ )  Cursor = Connection.cursor( )  Cursor.execute( ‘SELECT \* FROM table’ )  results = Cursor.fetchall( )  Cursor.close( )  Connection.close( ) |