Coding Summary

SQL & Python

IYKRA Batch 5 (Week 2)

Diemas Aksya Fachriza - DS1

Outline

- SQL
  - What is SQL?
  - Basic SQL Command
  - Function
  - Joins
- Python
  - Data Types
  - Conditional Statements
  - Loops
  - Function

SQL
(Structured Query Language)

What is SQL?

SQL is a domain-specific language used in programming and designed for **managing data** held in a **relational database management system**, or for **stream processing** in a relational data stream management system.

RDBMS (Relational Database Management System) using SQL :

- MySQL
- PostgreSQL
- Microsoft SQL Server
- Oracle

Terminologies

id	ISSN-L	ISSNs	PublisherId	Journal_Title
0	2056-9890	2056-9890	1	Acta Crystallographica Section E Crystallographic Communications
1	2077-0472	2077-0472	2	Agriculture
2	2073-4395	2073-4395	2	Agronomy
3	2076-2615	2076-2615	2	Animals
4	2076-3417	2076-3417	2	Applied Sciences
5	2306-5354	2306-5354	2	Bioengineering
6	2079-7737	2079-7737	2	
7	2079-6374	2079-6374	2	

Basic SQL Command

Data Definition Language

- CREATE
- ALTER
- DROP

Data Manipulation Language

- SELECT
- INSERT
- UPDATE
- DELETE

Data Control Language

- GRANT
- REVOKE

Data Definition Language (DDL)

CREATE

To create table in a database

```
CREATE TABLE
table_name (
column_1 datatype,
column_2 datatype,
column_3 datatype);
```

ALTER

To add or remove field/columns from a table

```
ALTER TABLE
table_name ADD
column_name
datatype;
```

DROP

To remove table from a database

```
DROP TABLE
table_name;
```

Learn more about data type in SQL here : https://www.w3schools.com/sql/sql_datatypes.asp

Data Manipulation Language (DML)

SELECT

To extract data from a table

```
SELECT * FROM table_name;
```

```
SELECT
    column1
    ,column2
FROM table_name;
```

INSERT

To add new data into a table

```
INSERT INTO table_name
(column1, column2)
VALUES (value1, value2);
```

UPDATE

To update data in a table

```
UPDATE table_name
SET column_name = value
WHERE condition;
```

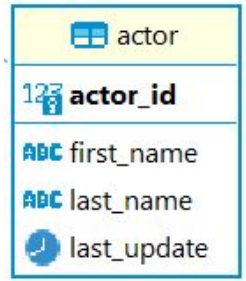
DELETE

To remove data from a table

```
DELETE FROM table_name
WHERE condition;
```

The symbol (*) above meaning “choose all columns”

Data Manipulation Language - SELECT Example



Fetch data from all fields/columns

```
select * from actor
```

actor_id	first_name	last_name	last_update
1	Penelope	Guiness	2013-05-26 14:47:57
2	Nick	Wahlberg	2013-05-26 14:47:57
3	Ed	Chase	2013-05-26 14:47:57
4	Jennifer	Davis	2013-05-26 14:47:57
5	Johnny	Lollobrigida	2013-05-26 14:47:57
6	Bette	Nicholson	2013-05-26 14:47:57
7	Grace	Mostel	2013-05-26 14:47:57
8	Matthew	Johansson	2013-05-26 14:47:57
9	Joe	Swank	2013-05-26 14:47:57
10	Christian	Gable	2013-05-26 14:47:57
11	Zero	Cage	2013-05-26 14:47:57
12	Karl	Berry	2013-05-26 14:47:57
13	Uma	Wood	2013-05-26 14:47:57
14	Vivien	Bergen	2013-05-26 14:47:57
15	Cuba	Olivier	2013-05-26 14:47:57
16	Fred	Costner	2013-05-26 14:47:57

Fetch data from fields/columns called "**actor_id**" and "**first_name** "

```
select actor_id, first_name from actor
```

actor_id	first_name
1	Penelope
2	Nick
3	Ed
4	Jennifer
5	Johnny
6	Bette
7	Grace
8	Matthew
9	Joe
10	Christian
11	Zero
12	Karl
13	Uma
14	Vivien
15	Cuba
16	Fred

Sample Dataset : https://www.postgresqltutorial.com/postgresql-sample-database/

Additional Commands

DISTINCT	: Get unique rows from a table	**SELECT DISTINCT(first_name) FROM actor**
WHERE	: Conditional statement.	**SELECT * FROM actor WHERE actor_id=1**
OR Resulting TRUE if either the condition is valid	: Multiple statement.	**SELECT * FROM actor WHERE actor_id=1 OR actor_id=5**
AND Resulting TRUE if 2 conditional is valid	: Multiple statement.	**SELECT * FROM actor WHERE actor_id >= 1 AND actor_id <= 5**
BETWEEN Resulting TRUE between specific range of value	: Multiple statement.	**SELECT * FROM actor WHERE actor_id BETWEEN 1 AND 5**
NOT Resulting TRUE if the otherwise condition is valid	: Multiple statement.	**SELECT * FROM actor WHERE NOT actor_id >= 1 AND actor_id <= 5**

Your turn!

Open the link below to try it yourself using a different dataset.

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

You can always try another DML queries and alter the dataset.

Don’t worry, you won’t break their database 😉

Function - Aggregate

Open the previous link, then show all data from table "**Products**"

SUM	: Get total value from a column	**SELECT SUM(Price) FROM Products**
AVG	: Get average value from a column	**SELECT AVG(Price) FROM Products**
COUNT	: Get how many rows fetched	**SELECT COUNT(*) FROM Products**
MIN	: Get lowest value from a column	**SELECT MIN(Price) FROM Products**
MAX	: Get highest value from a column	**SELECT MAX(Price) FROM Products**

Function - Group By

The GROUP BY statement groups rows that have the same values into summary rows

The GROUP BY statement is often used with aggregate functions **(COUNT, MAX, MIN, SUM, AVG)** to group the result-set by one or more columns.

Without Group By, returns how many rows fetched from a column

```
SELECT COUNT(CustomerID), Country
FROM Customers
```

COUNT(CustomerID)	Country
91	Germany

With Group By, returns how many rows fetched from a column on each “Country”

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

COUNT(CustomerID)	Country
3	Argentina
2	Austria
2	Belgium
9	Brazil
3	Canada
2	Denmark
2	Finland

Function - Order By

The ORDER BY keyword is used to **sort** the result-set in **ascending** or **descending** order.

The ORDER BY keyword sorts the records in **ascending order by default**. To sort the records in descending order, use the **DESC** keyword.

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID);
```

COUNT(CustomerID)	Country
1	Ireland
1	Norway
1	Poland
2	Austria
2	Belgium

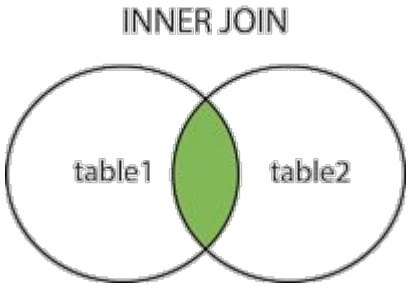
```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

COUNT(CustomerID)	Country
13	USA
11	Germany
11	France
9	Brazil
7	UK

JOIN

A JOIN clause is used to **combine rows** from **two or more tables**, based on a **related column between them**.

By default, JOIN in SQL is interpreted as INNER JOIN



Product Name	Supplier ID
Planet Oat Oatmilk	1
Honey Nut Frosted Flakes	2
Magnum Double Tub	5
Sour Patch Marshmallows	3
Ferrero Eggs	4

Supplier ID	Supplier Name
1	John
2	Anne
3	Robert
4	Jerry
5	Tim

Product Name	Supplier Name
Planet Oat Oatmilk	John
Honey Nut Frosted Flakes	Anne
Sour Patch Marshmallows	Robert
Ferrero Eggs	Jerry
Magnum Double Tub	Tim

Combine 2 tables by “**SupplierID**” column

JOIN

Table "**Orders**"

Number of Records: 196

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	1996-07-04	3
10249	81	6	1996-07-05	1
10250	34	4	1996-07-08	2
10251	84	3	1996-07-08	1
10252	76	4	1996-07-09	2

Table "**Customers**"

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany

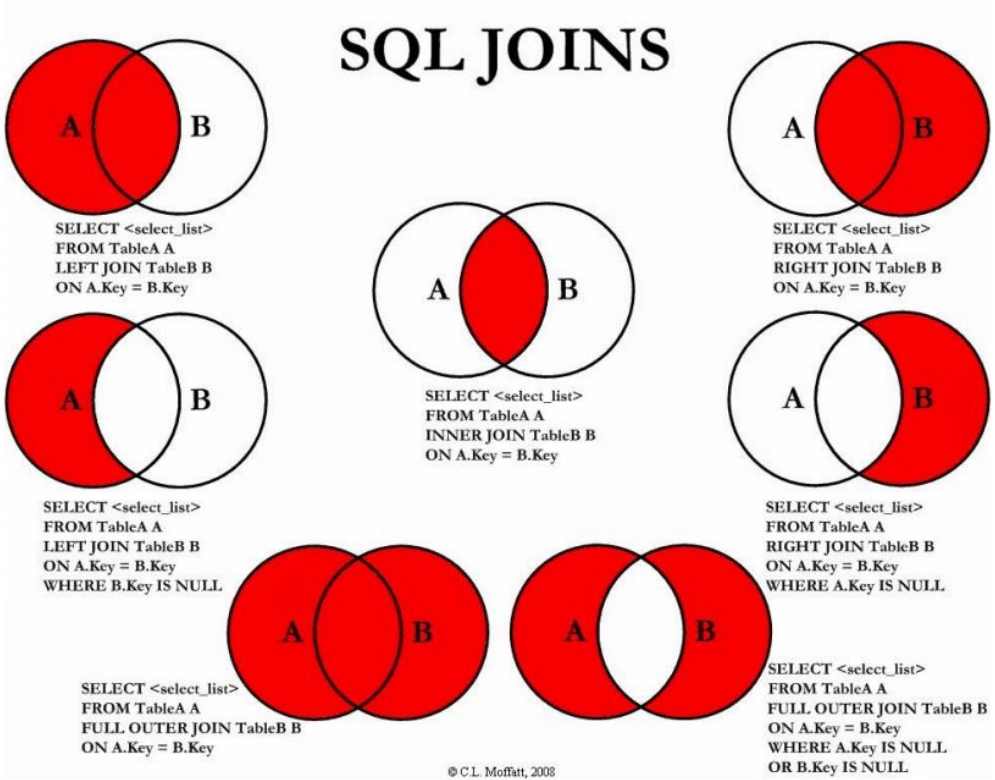
Get "**OrderID**", "**customerName**", and "**OrderDate**" by matching "**CustomerID**" column

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers
ON Orders.CustomerID=Customers.CustomerID;
```

Number of Records: 196

OrderID	CustomerName	OrderDate
10248	Wilman Kala	1996-07-04
10249	Tradição Hipermercados	1996-07-05
10250	Hanari Carnes	1996-07-08
10251	Victuailles en stock	1996-07-08
10252	Suprêmes délices	1996-07-09

Result


SQL JOINS
A
B
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
© C.L. Moffatt, 2008

Python

Data Types & Variable Assignment

Data Types

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`

Variables

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
```

Casting

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Print Variable

```
print(x)
print(y)
```

Data Type (Collections)

List : Store multiple items in a single variable.

Tuple : Store multiple items, unchangeable.

Set : Unordered and unindexed

Dictionary : Store data in key-value pairs

```
thislist = ["apple", "banana", "cherry"]

thistuple = ("apple", "banana", "cherry")

thisset = {"apple", "banana", "cherry"}

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

Indexing and Slicing

List = [0, 1, 2, 3, 4, 5]

0	1	2	3	4	5

List[0] = 0

List[1] = 1

List[2] = 2

List[3] = 3

List[4] = 4

List[5] = 5

List[0:] = [0,1,2,3,4,5]

List[:] = [0,1,2,3,4,5]

List[2:4] = [2, 3]

List[1:3] = [1, 2]

List[:4] = [0, 1, 2, 3]

str = "HELLO"

H	E	L	L	O
0	1	2	3	4

str[0] = 'H'

str[1] = 'E'

str[2] = 'L'

str[3] = 'L'

str[4] = 'O'

str[:] = 'HELLO'

str[0:] = 'HELLO'

str[:5] = 'HELLO'

str[:3] = 'HEL'

str[0:2] = 'HE'

str[1:4] = 'ELL'

Conditional Statements

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

Try the code below by clicking here

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Change the value of "**a**" and "**b**" variable however you like

Loops

While loops will execute a set of statements as long as a condition is true

Click here to try it yourself

Code

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Output

```
1
2
3
4
5
```

For loops is used for **iterating over a sequence** (that is either a **list**, a **tuple**, a **dictionary**, a **set**, or a **string**).

Click here to try it yourself

Code

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Output

```
apple
banana
cherry
```

Loops (Cont'd)

With the **break** statement we can stop the loop even if the while condition is true:

Code

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

Output

```
1
2
3
```

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
apple
banana
```

With the **continue** statement we can stop the current iteration of the loop, and continue with the next:

Code

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

Output

```
1
2
4
5
6
```

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

```
apple
cherry
```

Functions

A function is a block of code which only **runs when it is called**. You can pass data, known as **parameters**, into a function. A function can **return data as a result**.

Without using parameter

Code

```
def my_function():
  print("Hello from a function")

my_function()
```

Output

```
Hello from a function
```

Using parameter

Code

```
def my_function(x):
  return 5 * x

number = [my_function(3),
          my_function(4),
          my_function(5)]
print(number)
```

Output

```
[15, 20, 25]
```

Classes/Objects

A Class is like an object constructor, or a "blueprint" for creating objects.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def introduction(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.introduction()
```

```
Hello my name is John
```

References

- IYKRA Data Fellowship Batch 5 - Coding Module (SQL & Python)
- https://www.w3schools.com/

Thank you!

See you next time 😊