COLUMN Series

species

	code	species
0	m	maple
1	p	pine

COLUMN Series

trees

	tree	x	y	species	diameter	priority
0	A	10	4	m	8	71
1	B	20	4	m	10	100
2	C	30	4	p	6	30
3	D	40	4	p	8	40
4	E	50	4	m	12	99

```
import sqlite3
c = sqlite3.connect("worksheet.db")

def qry(sql):
    return pd.read_sql(sql, c)

species = qry("SELECT * FROM species")
trees = qry("SELECT * FROM trees")
```

DataFrame

① List creation, indexing, slicing
② Dict lookup
③ Pandas Series & DataFrame

---

1 `trees[trees["priority"] > 90][["x", "y"]]` # convert to SQL

FROM overloading [..] WHERE — LOOKUP SELECT

	x	y
1	20	4
4	50	4

```
SELECT x, y
FROM trees
WHERE priority > 90
```

2 qry("**SELECT** x+y **FROM** trees **WHERE** species = 'm'") # convert to Pandas

	x+y
0	14
1	24
2	54

```
maples = trees[trees["species"] == "m"]
maples["x"] + maples["y"]
```

3
```
cd = species["code"][species["species"]=="maple"].iloc[0]
trees[trees["species"] == cd]["tree"]   # convert to 2 SQL queries
```

cd = "m" — "m"

0	A
1	B
4	E

⇒ type : Pandas Series

4 qry("**SELECT** species **FROM** trees **ORDER BY** priority DESC")

	species
0	m
1	m
2	m
3	p
4	p

100, 99, 71, 40, 30

species

5

	code	species
0	m	maple
1	p	pine

trees

	tree	x	y	species	diameter	priority
0	A	10	4	m	8	71
1	B	20	4	m	10	100
2	C	30	4	p	6	30
3	D	40	4	p	8	40
4	E	50	4	m	12	99

```
list(qry("SELECT tree, priority FROM trees " +
         "ORDER BY priority DESC LIMIT 1").iloc[0])
```

	tree	priority
0	B	100

['B', 100]

* or tree or priority

6

```
qry("""SELECT COUNT(SPECIES) AS c1,
          COUNT(DISTINCT SPECIES) as c2
       FROM trees""")
```

	c1	c2
0	5	2

7

```
qry("""SELECT species, COUNT(SPECIES) AS count,
          AVG(diameter) AS size
       FROM trees
       GROUP BY species ORDER BY count DESC""")
```

	species	count	size
0	m	3	10.0
1	p	2	7.0

hydrants

	year	color	style	owner	alt	active
0	1999	red	K-81	private	1179	0
1	2000	red	M-3	public	1065	0
2	2001	green	Pacer	private	1058	1
3	2010	blue	Pacer	public	1081	1
4	2014	blue	Pacer	public	1052	1
5	2018	blue	Pacer	public	1109	1

```
hydrants = qry("""
    SELECT * FROM hydrants
""")
```

1 MILLION ROWS

8 qry("**SELECT** color, year **FROM** hydrants **WHERE** color = 'blue' ") FASTER !

	color	year
0	blue	2010
1	blue	2014
2	blue	2018

Reasons:
1. database performance optimizations
2. Dataframe → 3 rows

9 → df = qry("**SELECT** color, year **FROM** hydrants")
→ df[df.color == "blue"]

df.color ⟺ df["color"]

SLOWER !

Step 1 : Dataframe → 1 M rows

Step 2 : processing a Dataframe with 1M rows

df :

	color	year
0	red	1999
1	red	2000
2	green	2001
...	...	...

exactly same

10 qry("**SELECT** year **FROM** hydrants **WHERE** owner='private' AND active")

	year
0	2001

11 → df = qry("**SELECT** year, style, active **FROM** hydrants")
→ df[df.active == 1]["style"]

df :

	year	style	active
0	1999	K-81	0
1	2000	M-3	0
2		...	
...			

Pandas Series

2	Pacer
3	Pacer
4	Pacer
5	Pacer

hydrants

year	color	style	owner	alt	active
1999	red	K-81	private	1179	0
2000	red	M-3	public	1065	0
2001	green	Pacer	private	1058	1
2010	blue	Pacer	public	1081	1
2014	blue	Pacer	public	1052	1
2018	blue	Pacer	public	1109	1

```
hydrants = qry("""
    SELECT * FROM hydrants
""")
```

COLUMN Series

unique value counts from pandas Series
return value : Series
index : unique value
value : count

dict or list

```
12 hydrants["color"].value_counts() # convert to SQL
```

type : pandas Series

blue	3
red	2
green	1

{'blue': 3, 'red': 2, 'green': 1}

[3, 2, 1]

```
13 qry("""SELECT color, COUNT(*) FROM hydrants
          WHERE active GROUP BY color""")
```

→ ORDER BY in order to dictate how ordering

	color	COUNT(*)
0	blue	3
1	green	1

```
14 qry("""SELECT color, COUNT(*) AS count FROM hydrants
          GROUP BY color HAVING count > 1""")
```

	color	count
0	blue	3
1	red	2

```
15 qry("""SELECT color, COUNT(*) AS count
          FROM hydrants WHERE year >= 2000
          GROUP BY color HAVING count < 2""")
```

	color	count
0	green	1
1	red	1