Introduction to Data, Pandas and SQL

Part B – Pandas and SQL

Pavlos Protopapas

Lecture Outline

Part A: Data and Databases

What is data and how can we store it?

Part B: Pandas and SQL

Tools to inspect data

Relational Databases and Tables

- A collection of tables related to each other through common data values.
- Rows represent attributes of something.
- Everything in a column is values of one attributes.
- A cell is expected to be atomic.
- Tables are related to each other if they have columns called keys which represent the same values.

Structured Query Language (SQL)

What if our dataset doesn’t fit in RAM?

What if we cannot download the entire dataset on our computer?

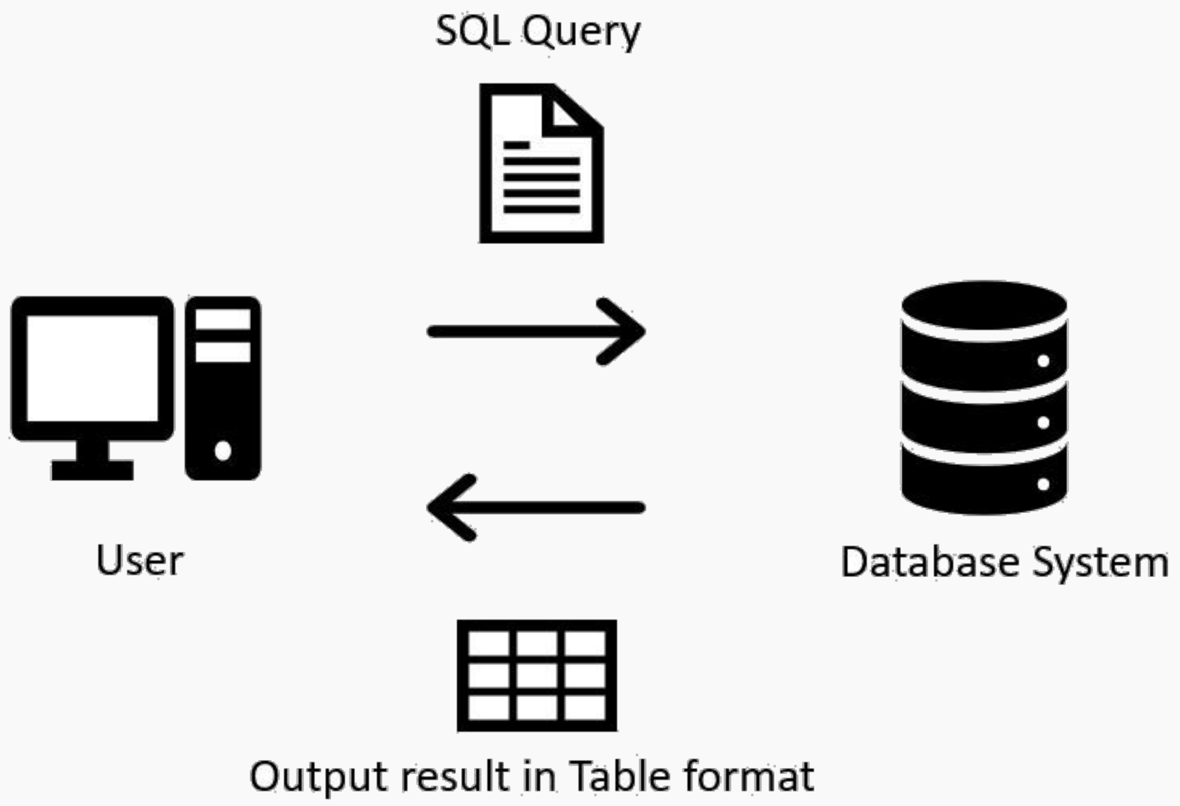
How to pronounce SQL
Credit : u/thefizzynator

SQL

Structured Query Language

SQL is a language that allows us to access and manipulate data stored in relational databases.

With SQL we can create, read, update and delete records stored in these databases.



SQL Queries

Queries are statements used to add, modify, query, or remove data from an SQL database

There are several SQL data statements, most used are:

- SELECT – to retrieve data from the database
- UPDATE – to modify data in the table
- DELETE – to delete data from the table
- INSERT – to add/populate the table in database

SQL Queries

SELECT is used to retrieve data from one or more tables in the database

Syntax of a SELECT statement,

```
SELECT columns or expressions
FROM tables
WHERE condition
GROUP BY column to group rows
HAVING condition
ORDER BY column to order rows
LIMIT number of rows to be returned
INTO TEMP  save results of query in a temporary table
```

SQL Queries - Example

Consider a SQL table **fortune500** below with information about fortune 500 companies

company_name	country	num_of_employees	revenues_millions	profit_millions	Assets_millions
Walmart	USA	2300000	485873	13643.0	198825
State Grid	China	926067	315199	9571.3	198825
Sinopec Group	China	713288	267518	1257.9	310726
Toyota Motor	Japan	364445	254694	16899.3	437575
...					

What is the SQL Query to display all columns from **fortune500** table?

```
SELECT *
FROM fortune500
```

SQL Queries - Example

Consider a SQL table **fortune500** below with information about fortune 500 companies

company_name	**country**	**num_of_employees**	**revenues_millions**	**profit_millions**	**Assets_millions**
Walmart	USA	2300000	485873	13643.0	198825
State Grid	China	926067	315199	9571.3	198825
Sinopec Group	China	713288	267518	1257.9	310726
Toyota Motor	Japan	364445	254694	16899.3	437575
...					

What is the SQL Query to display the first 4 rows with only *company_name*, *country* columns from **fortune500** table?

```
SELECT company_name, country
FROM fortune500
LIMIT 4
```

SQL Queries - Example

What is the SQL Query to display the first 4 rows with only *company_name*, *country* columns from **fortune500** table?

```
SELECT company_name, country
FROM fortune500
LIMIT 4
```

What will be the output of the above code?

company_name	country
Walmart	USA
State Grid	China
Sinopec Group	China
Toyota Motor	Japan

SQL Queries - Example

Consider a SQL table **fortune500** below with information about fortune 500 companies

company_name	**country**	**num_of_employees**	**revenues_millions**	**profit_millions**	**Assets_millions**
Walmart	USA	2300000	485873	13643.0	198825
State Grid	China	926067	315199	9571.3	198825
Sinopec Group	China	713288	267518	1257.9	310726
Toyota Motor	Japan	364445	254694	16899.3	437575
...					

Which companies have more than a million employees?

```
SELECT company_name, country, num_of_employees
FROM fortune500
WHERE num_of_employees > 1000000
```

SQL Queries - Example

Which companies have more than a million employees?

```
SELECT company_name, country, num_of_employees
FROM fortune500
WHERE num_of_employees > 1000000
```

What will be the output of the above code?

company_name	country	num_of_employees
Wallmart	USA	2300000
China National Petroluem	China	1512048

SQL Queries - Example

Consider a SQL table **fortune500** below with information about fortune 500 companies

company_name	**country**	**num_of_employees**	**revenues_millions**	**profit_millions**	**Assets_millions**
Walmart	USA	2300000	485873	13643.0	198825
State Grid	China	926067	315199	9571.3	198825
Sinopec Group	China	713288	267518	1257.9	310726
Toyota Motor	Japan	364445	254694	16899.3	437575
...					

What is the total revenue of fortune 500 companies in each country? Display only the first 4 rows.

```
SELECT country, SUM(revenues_millions)
FROM fortune500
GROUP BY country
LIMIT 4
```

GROUP BY can be used with an aggregation function like COUNT(), AVG(), SUM(), MIN(), MAX()

SQL Queries - Example

What is the total revenue of fortune 500 companies in each country? Display only the first 4 rows.

```
SELECT country, SUM(revenues_millions) AS total_revenue
FROM fortune500
GROUP BY country
LIMIT 4
```

You can use AS to set/rename columns in the output

What will be the output of the above code?

country	total_revenue
Australia	235821
Belgium	45905
Brazil	364172
Britain	1179837

SQL Queries - Example

What is the total revenue of fortune 500 companies in each country ordered by total revenue?

```
SELECT country, SUM(revenues_millions) AS total_revenue
FROM fortune500
GROUP BY country
ORDER BY total_revenue DESC
```

What will be the output of the above code?

country	**total_revenue**
USA	8476825
China	6038369
Japan	2711366
Germany	1853535

UPDATE

UPDATE statement is used to modify one or more rows in a table

Syntax of a UPDATE statement,

```
UPDATE    Table

SET       column1 = expression1,
          column2 = expression2

WHERE     condition
```

Data scientists don't usually modify the databases, they usually query data. Therefore, the UPDATE query is not used as much as the SELECT query.

UPDATE

Consider a SQL table **fortune500** below with information about fortune 500 companies

company_name	country	num_of_employees	revenues_millions	profit_millions	Assets_millions
Walmart	USA	2300000	485873	13643.0	198825
State Grid	China	926067	315199	9571.3	198825
Sinopec Group	China	713288	267518	1257.9	310726
Toyota Motor	Japan	364445	254694	16899.3	437575
...					
IBM	USA	414400	79919	11872.0	117470
...					

Change the name of ‘IBM’ company in fortune500 table to ‘International Business Machines’

```
UPDATE fortune500
SET company_name = 'International Business Machines'
WHERE company_name = 'IBM'
```

UPDATE

Change the name of IBM company in fortune500 table to ’International Business Machines’

```
UPDATE fortune500
SET company_name = 'International Business Machines'
WHERE company_name = 'IBM'
```

What will be the output of the above code?

company_name	country	num_of_employees	revenues_millions	profit_millions	Assets_millions
Wallmart	USA	2300000	485873	13643.0	198825
State Grid	China	926067	315199	9571.3	198825
Sinopec Group	China	713288	267518	1257.9	310726
Toyota Motor	Japan	364445	254694	16899.3	437575
. . .					
International Business Machines	USA	414400	79919	11872.0	117470
. . .					

SQL programmers be like
NathanTheCreator
Caps Lock

SQLite

How would you run a SQL query in Python?

How would you query a database?

SQLite

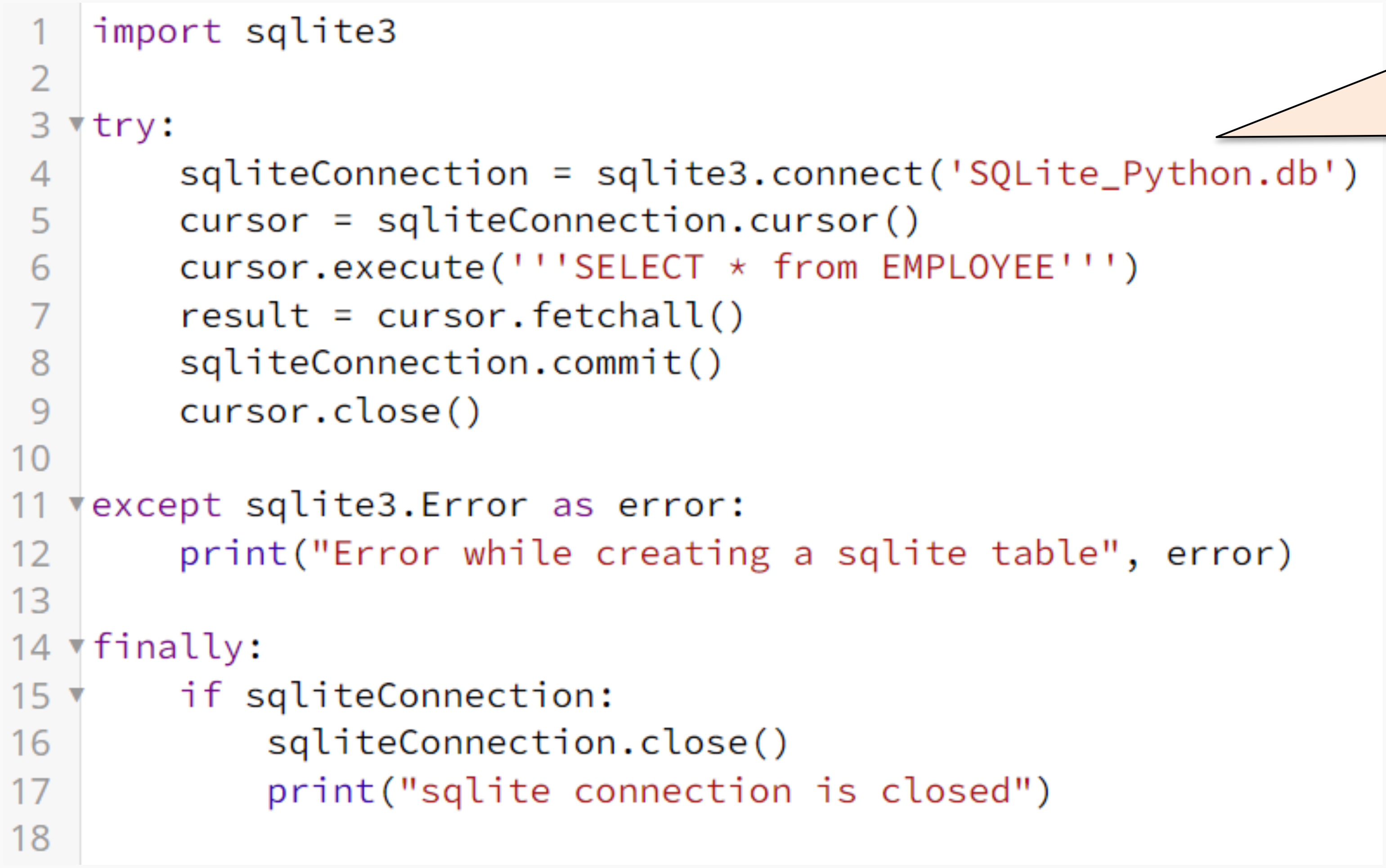
SQLite is a software library that provides a relational database management system.

Python implements a standard database API called **DBAPI2**. DB-API2 is a library that lets python connect to a database server. So, SQLite is used to process DB-API2 method calls and query the database.

There is an even higher-level API available called SQLAlchemy.

SQLite

SQLite



```python
import sqlite3

try:
    sqliteConnection = sqlite3.connect('SQLite_Python.db')
    cursor = sqliteConnection.cursor()
    cursor.execute('''SELECT * from EMPLOYEE''')
    result = cursor.fetchall()
    sqliteConnection.commit()
    cursor.close()

except sqlite3.Error as error:
    print("Error while creating a sqlite table", error)

finally:
    if sqliteConnection:
        sqliteConnection.close()
        print("sqlite connection is closed")

```

This opens a connection to the SQLite database file. If database is opened successfully, it returns a connection object.

SQLite

```python
import sqlite3

try:
    sqliteConnection = sqlite3.connect('SQLite_Python.db')
    cursor = sqliteConnection.cursor()
    cursor.execute('''SELECT * from EMPLOYEE''')
    result = cursor.fetchall()
    sqliteConnection.commit()
    cursor.close()

except sqlite3.Error as error:
    print("Error while creating a sqlite table", error)

finally:
    if sqliteConnection:
        sqliteConnection.close()
        print("sqlite connection is closed")

```

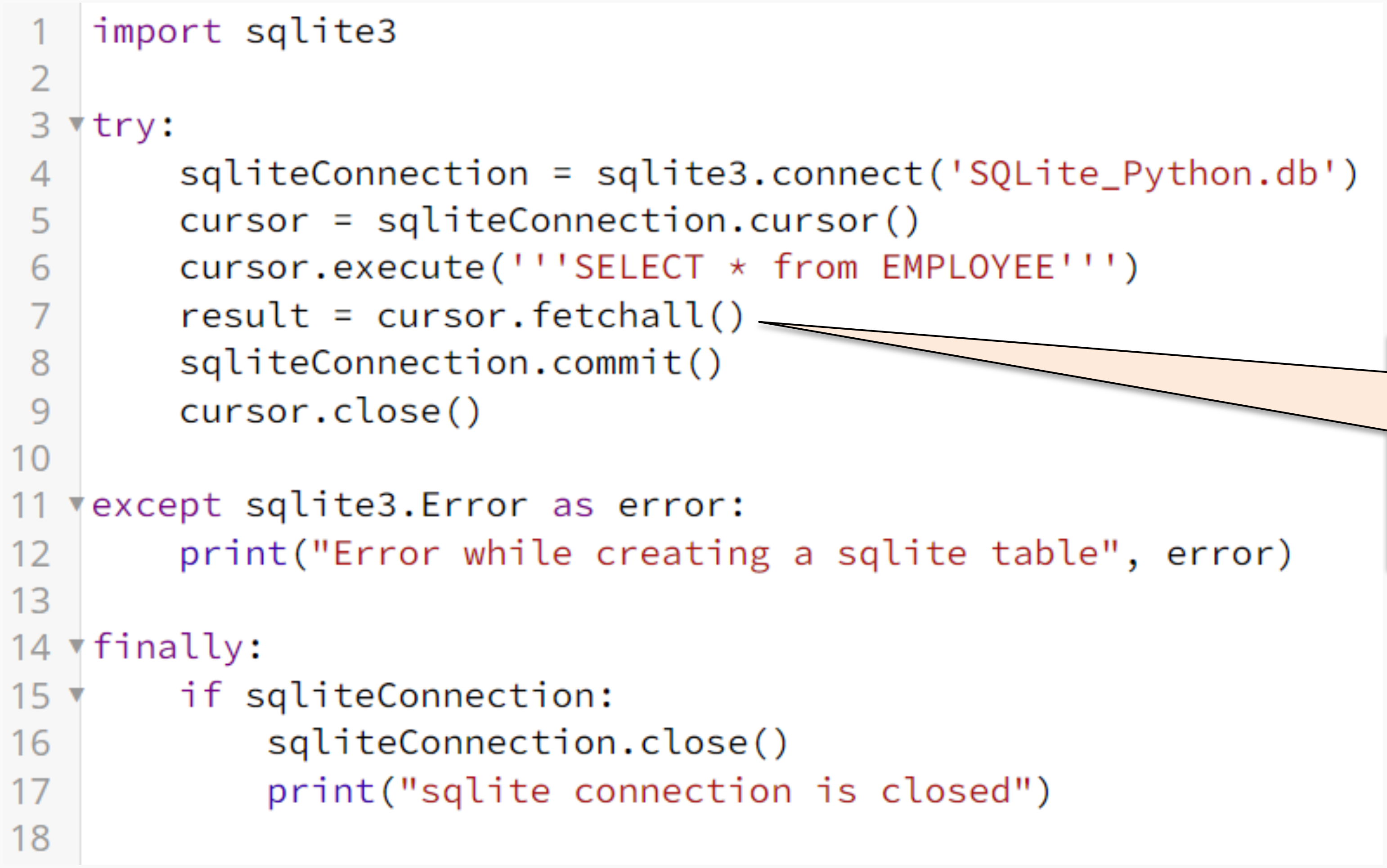
A cursor is an instance that you can use to invoke methods that execute SQLite statements and fetch data from the result sets of the queries.

SQLite

```python
import sqlite3

try:
    sqliteConnection = sqlite3.connect('SQLite_Python.db')
    cursor = sqliteConnection.cursor()
    cursor.execute('''SELECT * from EMPLOYEE''')
    result = cursor.fetchall()
    sqliteConnection.commit()
    cursor.close()

except sqlite3.Error as error:
    print("Error while creating a sqlite table", error)

finally:
    if sqliteConnection:
        sqliteConnection.close()
        print("sqlite connection is closed")

```

This routine executes an SQL statement.

SQLite



```
import sqlite3

try:
    sqliteConnection = sqlite3.connect('SQLite_Python.db')
    cursor = sqliteConnection.cursor()
    cursor.execute('''SELECT * from EMPLOYEE''')
    result = cursor.fetchall()
    sqliteConnection.commit()
    cursor.close()

except sqlite3.Error as error:
    print("Error while creating a sqlite table", error)

finally:
    if sqliteConnection:
        sqliteConnection.close()
        print("sqlite connection is closed")

```

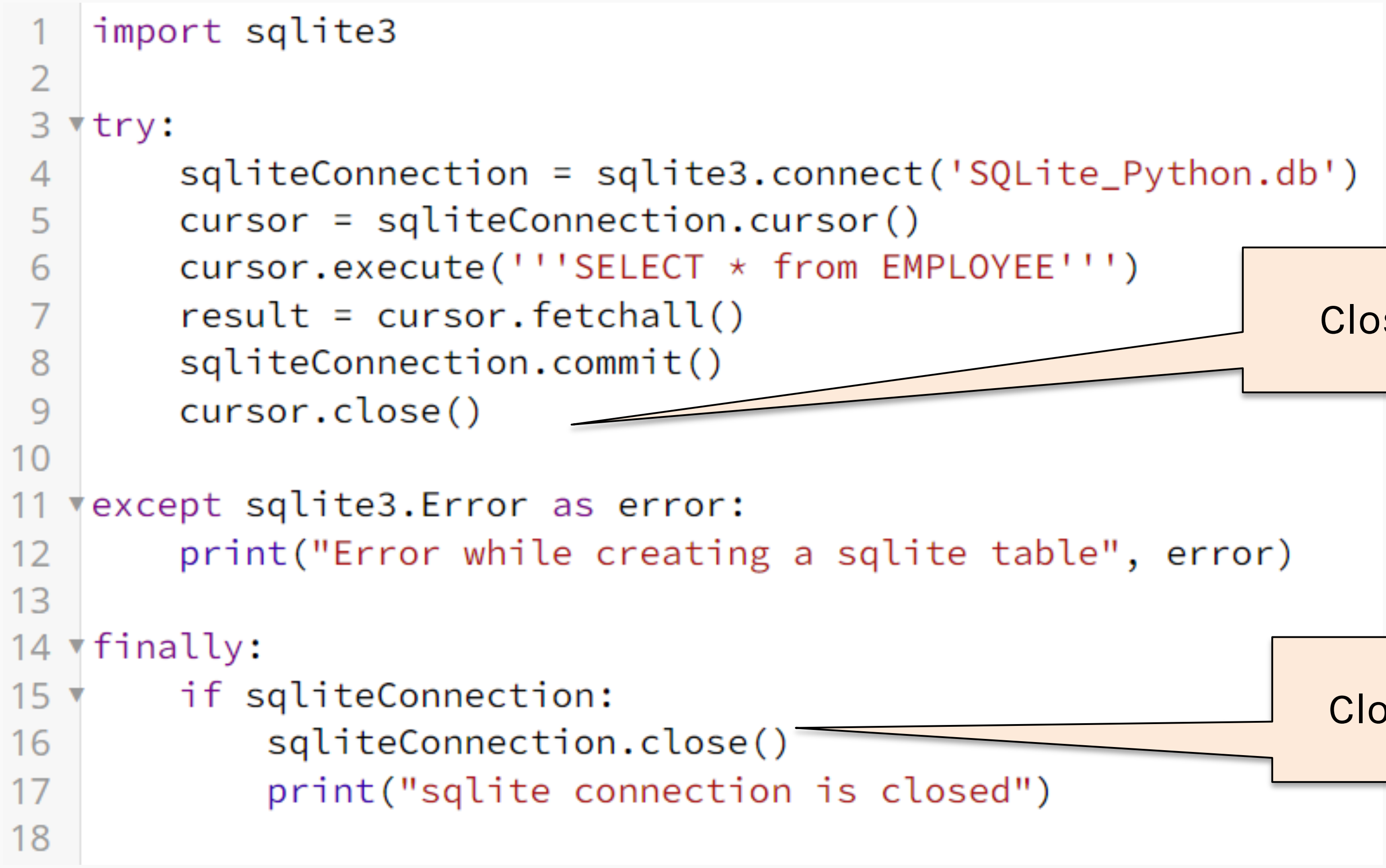
This routine fetches all rows of a query result, returning a list.

SQLite

```python
import sqlite3

try:
    sqliteConnection = sqlite3.connect('SQLite_Python.db')
    cursor = sqliteConnection.cursor()
    cursor.execute('''SELECT * from EMPLOYEE''')
    result = cursor.fetchall()
    sqliteConnection.commit()
    cursor.close()

except sqlite3.Error as error:
    print("Error while creating a sqlite table", error)

finally:
    if sqliteConnection:
        sqliteConnection.close()
        print("sqlite connection is closed")

```

This method commits the current transaction. If you don't call this method, anything you did since the last call to commit() is not visible from other database connections.

SQLite

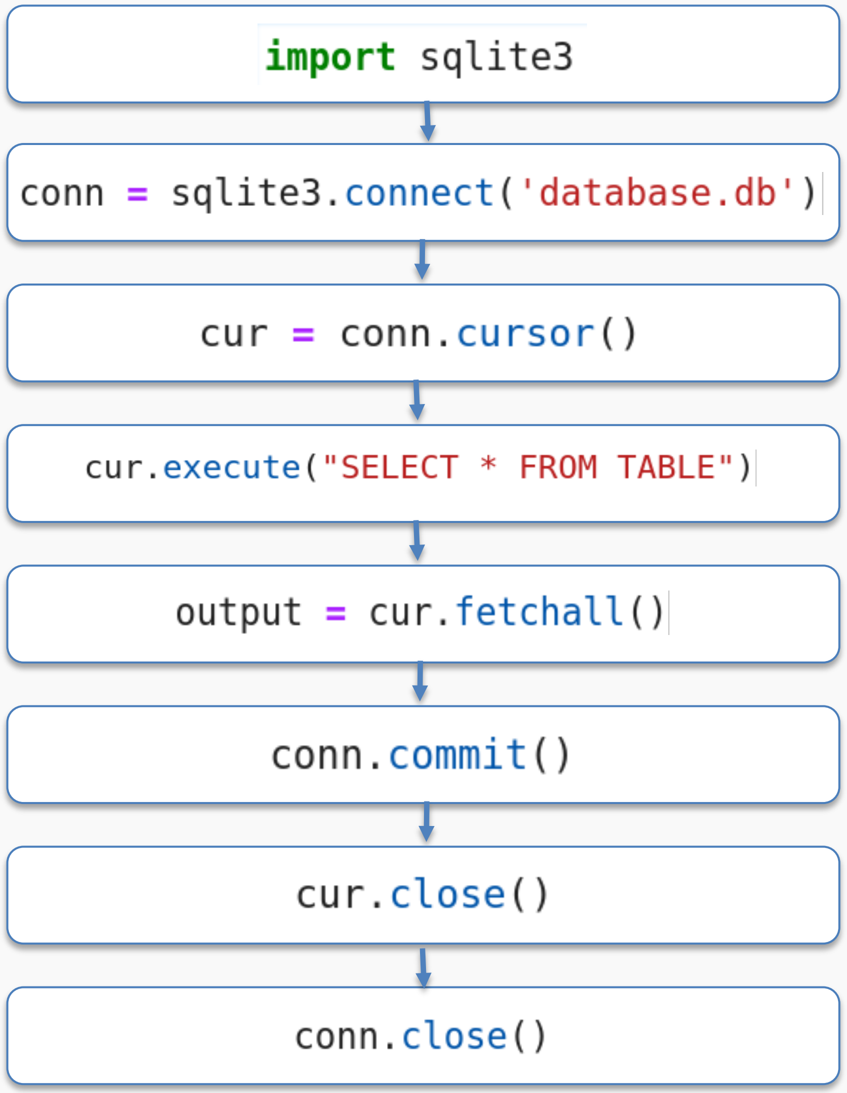


```python
import sqlite3

try:
    sqliteConnection = sqlite3.connect('SQLite_Python.db')
    cursor = sqliteConnection.cursor()
    cursor.execute('''SELECT * from EMPLOYEE''')
    result = cursor.fetchall()
    sqliteConnection.commit()
    cursor.close()

except sqlite3.Error as error:
    print("Error while creating a sqlite table", error)

finally:
    if sqliteConnection:
        sqliteConnection.close()
        print("sqlite connection is closed")

```

Closes the cursor connection.

Closes the database connection.

SQLite



```
import sqlite3
conn = sqlite3.connect('database.db')
cur = conn.cursor()
cur.execute("SELECT * FROM TABLE")
output = cur.fetchall()
conn.commit()
cur.close()
conn.close()
```

SQLite and Pandas

SQLite and Pandas

VERBS	PANDAS	SQL
QUERY/SELECTION	query() (and loc[], iloc[])	SELECT WHERE
SORT	sort_values()	ORDER BY
SELECT-DISTINCT	unique(), drop_duplicates()	SELECT DISTINCT COLUMN
ASSIGN	assign	ALTER/UPDATE

SQLite and Pandas

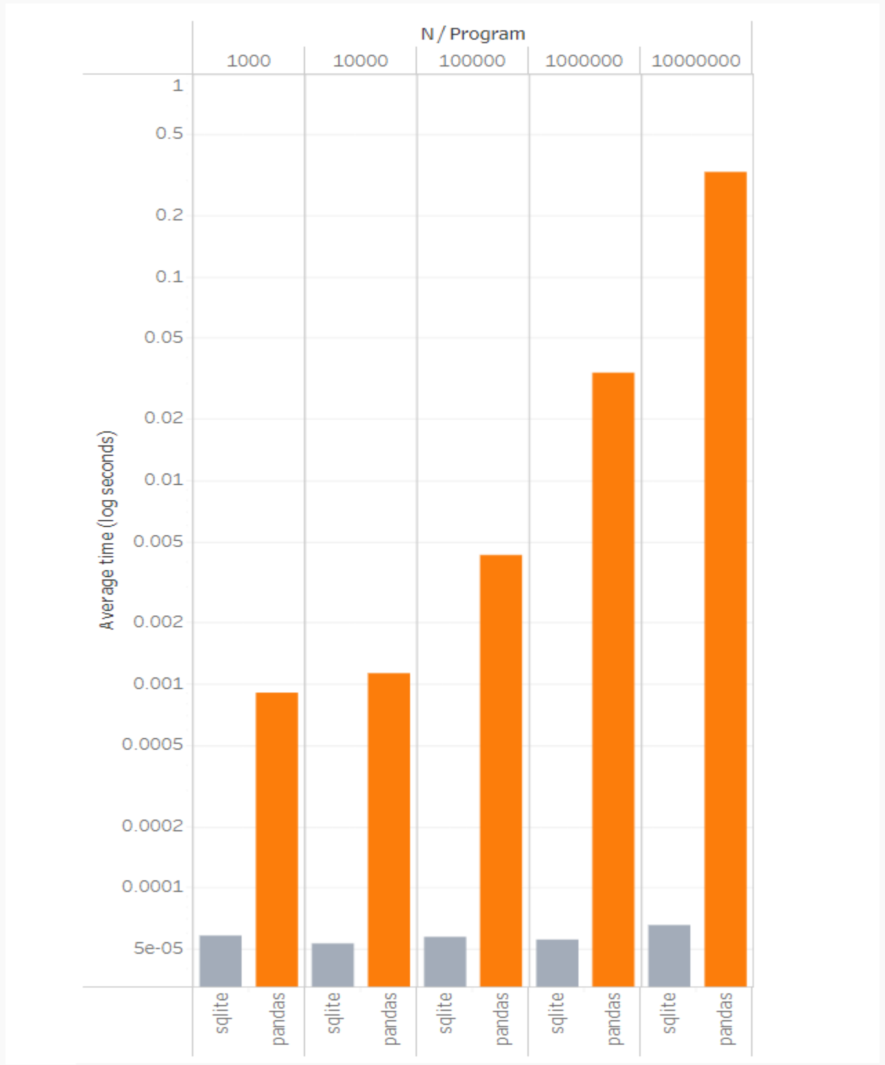
VERBS	PANDAS	SQL
AGGREGATE	describe(), mean(), max()	None, AVG(),MAX()
SAMPLE	sample()	implementation dep, use RAND()
GROUP-AGG	groupby/agg, count, mean	GROUP BY
DELETE	drop/masking	DELETE/WHERE

Structured Query Language

SQL performs faster under the following conditions:

- When dealing with highly structured and relational data.
- When the mathematical operations involved are kept simple.
- When there is no need to transform data into other formats.

Moreover, relational databases are essential for managing data that does not fit into memory.



Performance comparison for 'select' queries

Combining Tables

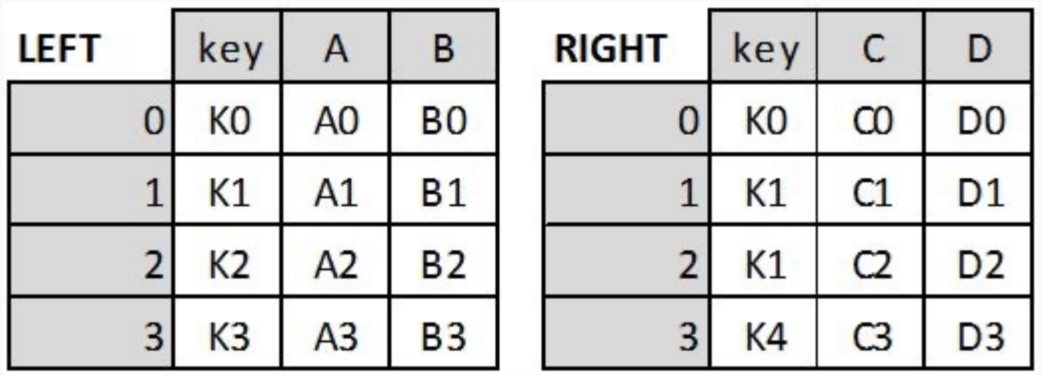
Mutating joins add new variables to one table by matching rows.

Common types of joins:

1. Inner Join
2. Left (Outer) Join
3. Right (Outer) Join
4. Full (Outer) Join

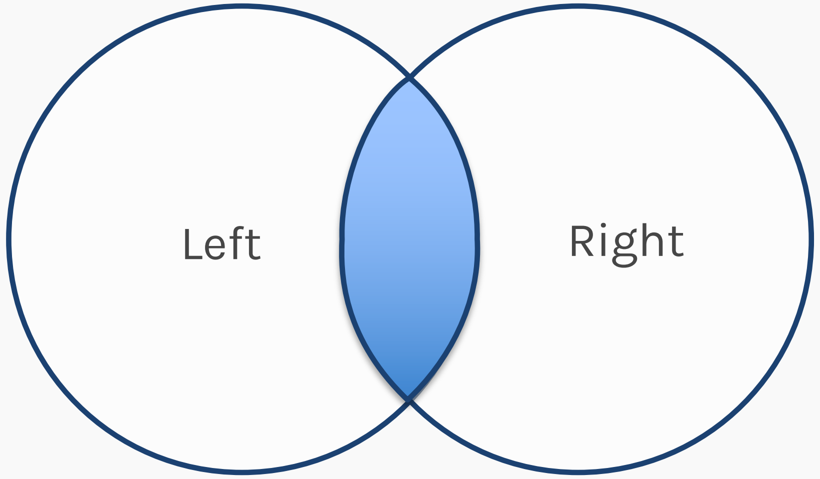
Note: SQLite supports Inner Join and Left (Outer) Join

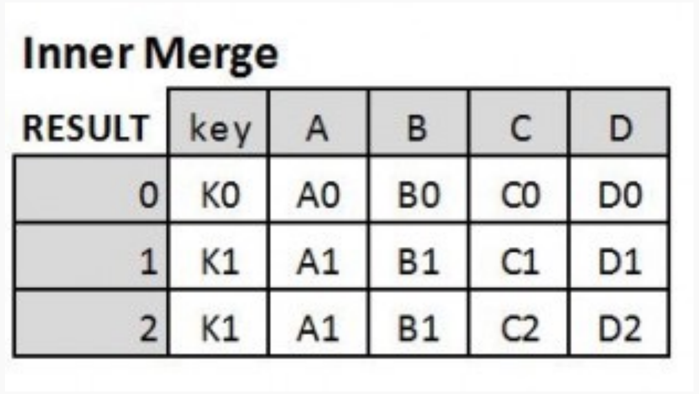
Inner Join



LEFT	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

RIGHT	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K1	C2	D2
3	K4	C3	D3





Inner Merge

RESULT	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K1	A1	B1	C2	D2

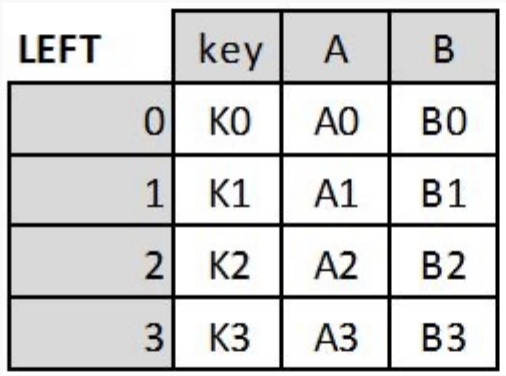
Pandas

```
pd.merge(left, right, on='key', how='inner')
```

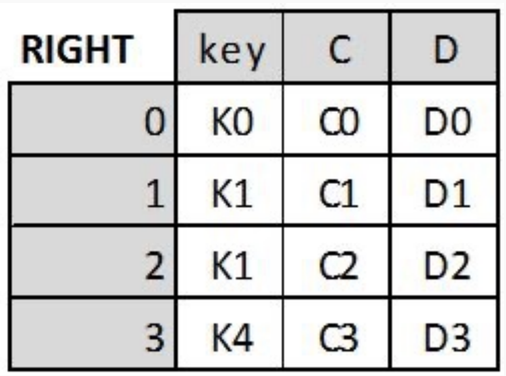
SQL

```
SELECT left.id,left.KEY,left.A,left.B,right.C,right.D FROM left
INNER JOIN right ON left.key = right.key;
```

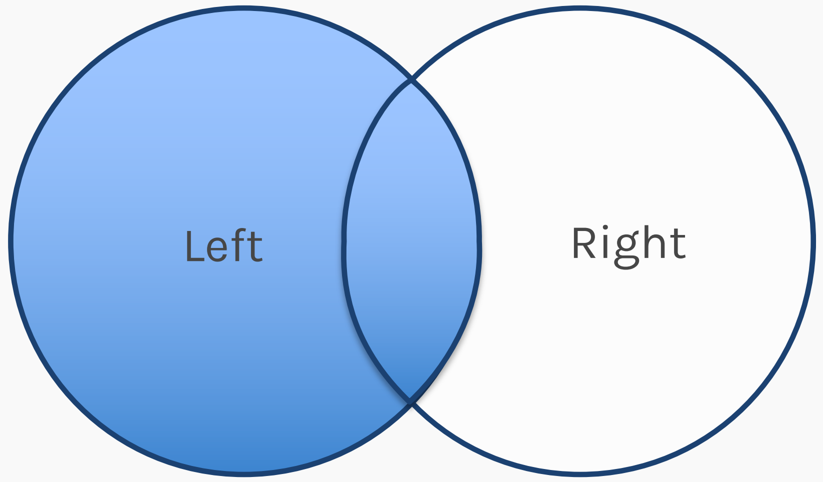
Left Outer Join

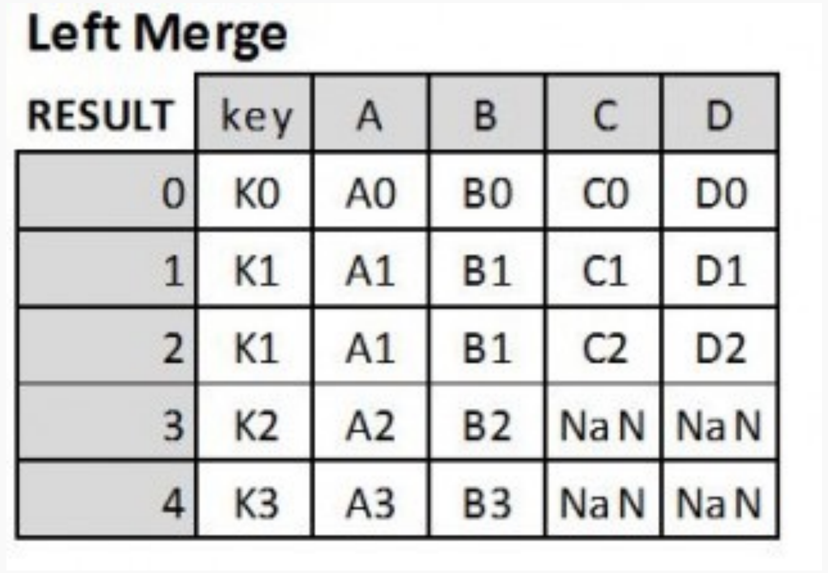


LEFT	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3



RIGHT	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K1	C2	D2
3	K4	C3	D3





Left Merge

RESULT	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K1	A1	B1	C2	D2
3	K2	A2	B2	NaN	NaN
4	K3	A3	B3	NaN	NaN

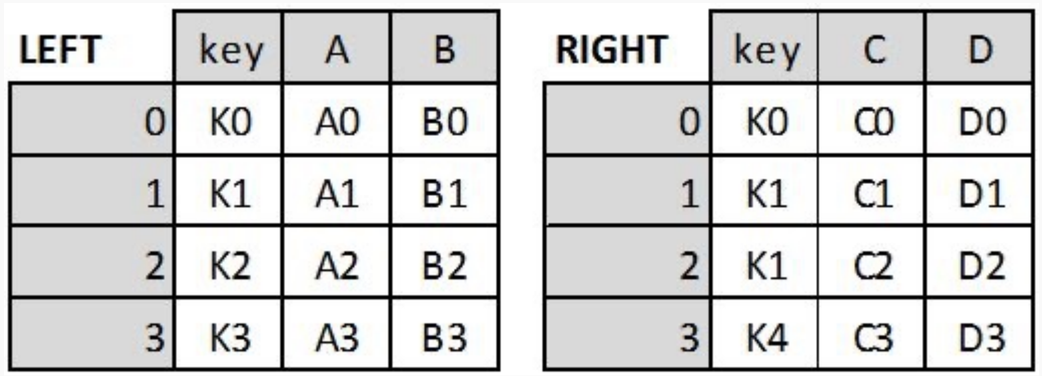
Pandas

```
pd.merge(left, right, on='key', how='left')
```

SQL

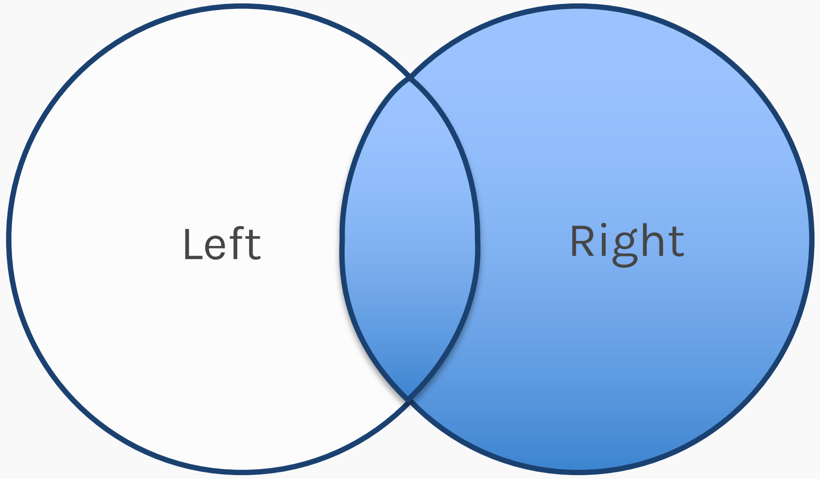
```
SELECT left.id,left.KEY,left.A,left.B,right.C,right.D FROM left
LEFT JOIN right ON left.key = right.key;
```

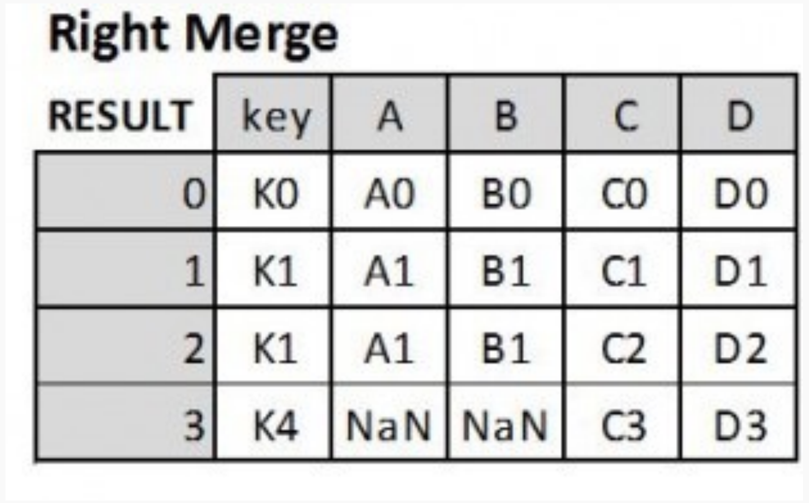
Right Outer Join



LEFT	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

RIGHT	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K1	C2	D2
3	K4	C3	D3





Right Merge

RESULT	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K1	A1	B1	C2	D2
3	K4	NaN	NaN	C3	D3

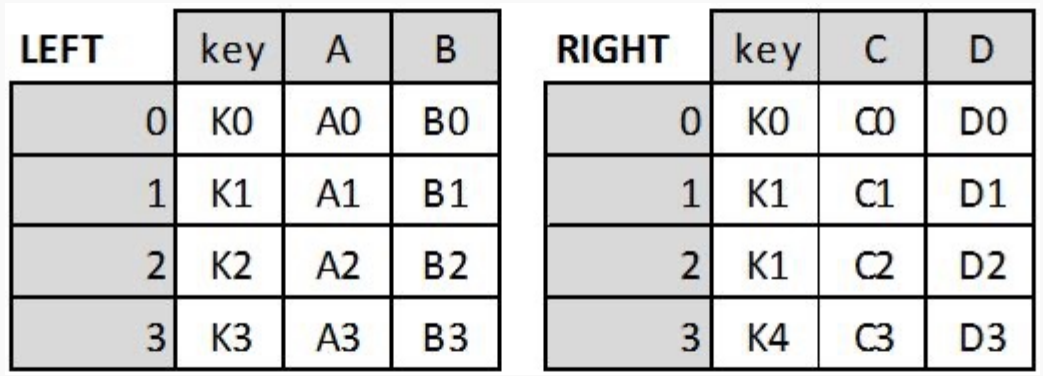
Pandas

```
pd.merge(left, right, on='key', how='right')
```

SQL

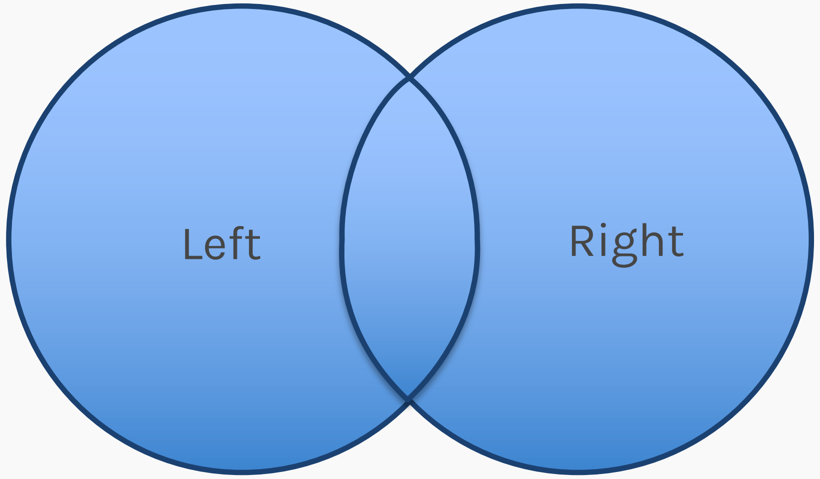
```
SELECT right.id,right.KEY,left.A,left.B,right.C,right.D FROM right
LEFT JOIN left ON right.key = left.key;
```

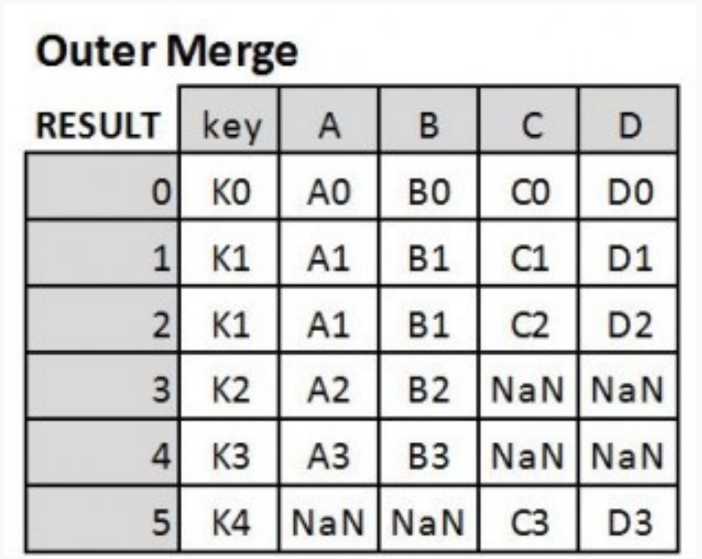
Full Outer Join



LEFT	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

RIGHT	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K1	C2	D2
3	K4	C3	D3





Outer Merge

RESULT	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K1	A1	B1	C2	D2
3	K2	A2	B2	NaN	NaN
4	K3	A3	B3	NaN	NaN
5	K4	NaN	NaN	C3	D3

Pandas

```
pd.merge(left, right, on='key', how='outer')
```

SQL

Left Outer Join U Right Outer Join

DATA SCIENC
LEFT JOIN
RIGHT JOIN
INNER JOIN
FULL JOIN

Thank you