SQL vs. Pandas

Create a DataFrame

```
data = {
    'name': ['John', 'Jane', 'Mary'],
    'age': [25, 30, 27]
}
df = pd.DataFrame(data)
```

Create DataFrame from database

```
conn = sqlite3.connect('harry-potter.db')
```

Option 1. Extract using `fetchall()`

```
results = conn.execute("SELECT * FROM students").fetchall() # list of tuples
df = pd.DataFrame(results, columns=["id", "fname", "lname", "year"])
```

Option 2. Extract using `pd.read_sql()`

```
df = pd.read_sql("SELECT * FROM students", conn)
```

SQL vs. Pandas

- Inspection
- Selection
- Filtering
- Sorting
- Aggregation
- Grouping
- Joining

Read `harry-potter.db` into Pandas DataFrame

```
import sqlite3
import pandas as pd

conn = sqlite3.connect('harry-potter.db')
df = pd.read_sql("SELECT * FROM students", conn)
```

Inspection

SQL

```sql
DESCRIBE students
SELECT * FROM students LIMIT 5
```

Pandas

```python
df.info()
df.describe()
df.head()
df.tail(3)
```

Selection

SQL

```sql
SELECT first_name FROM students;
SELECT first_name, last_name FROM students;
SELECT * FROM students;
```

Pandas

```python
df['first_name']

cols = ['first_name', 'last_name']
df[cols]
df[['first_name', 'last_name']]

df['first_name', 'last_name']   # Error

df
```

Create new columns

```
# vectorized operations
df['two'] = 2
df['age'] = 1997 - df['birthyear']
df['age2'] = df['age'] * 2
df['age3'] = df['age'] + df['age2']
```

Filtering

- `query()` : **SQL-like syntax**
- `loc[]` : label-based
- `iloc[]` : position-based`

...

Filtering

SQL

```sql
SELECT * FROM students WHERE age = 10;
SELECT first_name, house FROM students WHERE age > 10;
SELECT * FROM students WHERE age in (10, 11);
```

Pandas - `query`

```python
df.query('age == 10')
df.query('age > 10')[['first_name', 'house']]
df.query('age in (10, 11)')
```

Pattern matching

SQL

```sql
SELECT * FROM students WHERE first_name LIKE 'J%';
SELECT * FROM students WHERE first_name LIKE '%J';
SELECT * FROM students WHERE first_name LIKE '%a%';
```

Pandas - `query`

```
df.query("first_name.str.startswith('J')")
df.query("first_name.str.endswith('J')")
df.query("first_name.str.contains('a')")
df.query("first_name.str.contains('a', case=False)")
```

Sorting

SQL

```sql
SELECT * FROM students ORDER BY age;
SELECT * FROM students ORDER BY age desc;
SELECT * FROM students ORDER BY age, first_name;
```

Pandas

```python
df.sort_values(by='age')
df.sort_values(by='age', ascending=False)
df.sort_values(by=['age', 'first_name'])
```

💻 Query `harrypotter.db` with Pandas

Find the answers to the following questions using Pandas `query()` function.

```
df = pd.read_sql("SELECT * FROM students", conn)
```

- In what year was Harry Potter born?
- List the names of students born after 1980.
- Who is the youngest student?

Aggregation

```
df.agg({'column_name': 'function_name'})
```

Aggregation

SQL

```sql
SELECT AVG(age) FROM students;
SELECT AVG(age), MAX(age) FROM students;
```

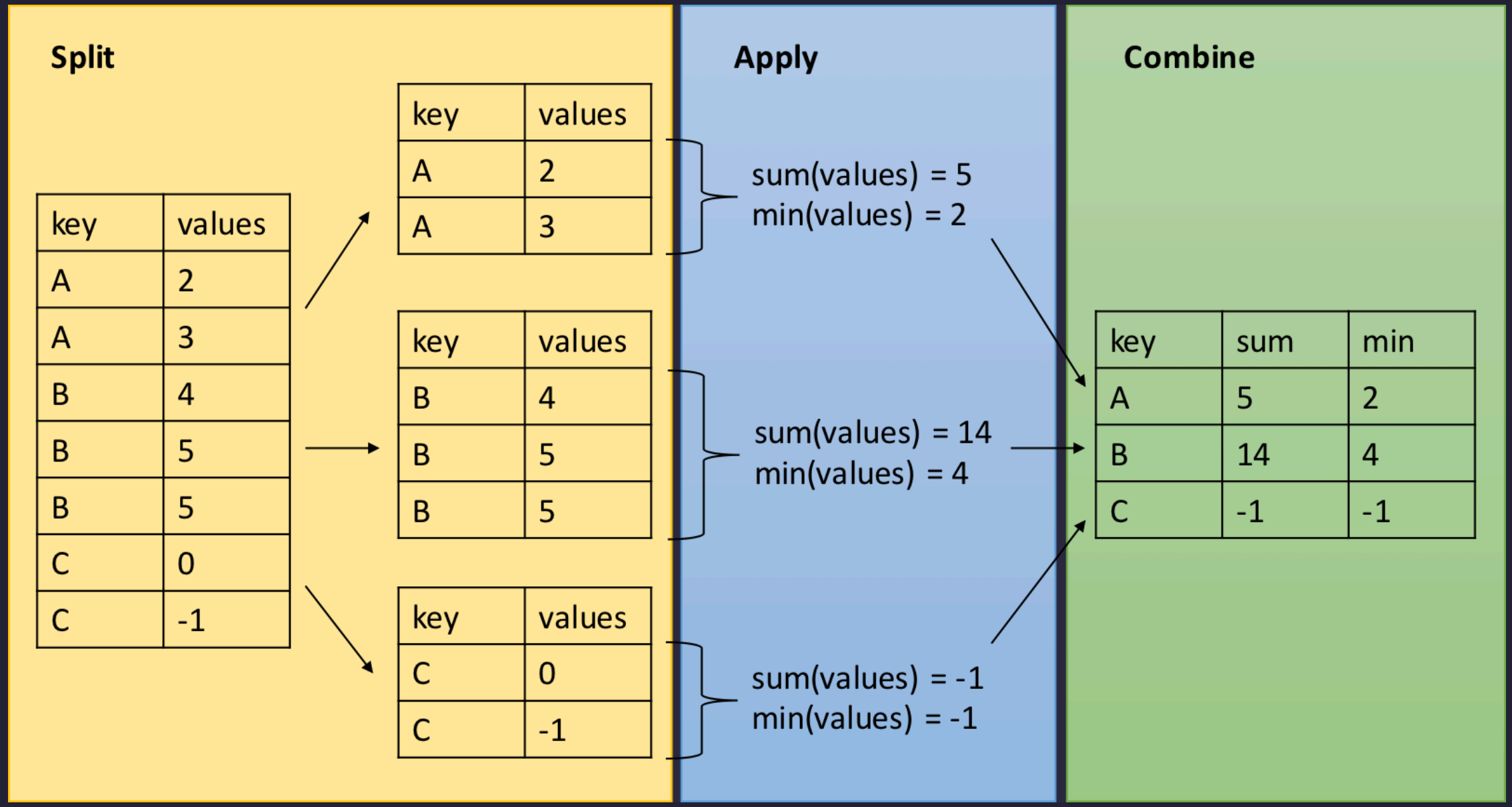
Pandas

```python
df.agg({'age': 'mean'})
df.agg({'age':['mean', 'max']})
```

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.aggregate.html

Grouping

```
df.groupby('column_name').agg({'column_name': 'function_name'})
```


Split
Apply
Combine
key
values
A
2
A
3
B
4
B
5
B
5
C
0
C
-1
key
values
A
2
A
3
key
values
B
4
B
5
B
5
key
values
C
0
C
-1
sum(values) = 5
min(values) = 2
sum(values) = 14
min(values) = 4
sum(values) = -1
min(values) = -1
key
sum
min
A
5
2
B
14
4
C
-1
-1

Grouping

SQL

```sql
SELECT house_id, AVG(age) FROM students GROUP BY house_id;
SELECT house_id, AVG(age), MAX(age) FROM students GROUP BY house_id;
```

Pandas

```python
df.groupby('house_id').agg({'age': 'mean'})
df.groupby('house_id').agg({'age': ['mean', 'max']})
```

---

https://realpython.com/pandas-groupby/

Joining

```
pd.merge(df1, df2, left_on='column_name', right_on='column_name')
df1.merge(df2, left_on='column_name', right_on='column_name')
```

Joining

SQL

```sql
SELECT * FROM students JOIN houses ON students.house_id = houses.id;
```

Pandas

```python
# Join on column (default inner join)
pd.merge(students, houses, left_on='house_id', right_on='id', how='inner')
pd.merge(students, houses, left_on='house_id', right_on='id')
```

💻 SQL Murder Mystery

- Use SQL only to fetch relevant tables
- Use Pandas `query()` to filter the records to get the same output as the SQL statement in each question.

Choosing between SQL and Pandas

SQL:

- If doing a task in SQL can cut the amount of data returned to the client (e.g. by filtering)
- Data extraction, filtering, simple data analysis

Pandas:

- If the amount of data returned to the client remains unchanged or grows by doing it in SQL (e.g. adding columns)
- Complex data analysis, formatting, etc.

If it's painful or ugly, do it in Pandas

---

https://towardsdatascience.com/sql-vs-pandas-how-to-balance-tasks-between-server-and-client-side-9e2f6c95677

	SQL	Pandas
Selection	`select name, age from students`	`df[['name', 'age']]`
Filtering	`select * from students where age > 10`	`df.query('age > 10')`
Sorting	`select * from students order by age`	`df.sort_values(by = 'age')`

	SQL	Pandas
Aggregation	`SELECT AVG(age) FROM students`	`df.agg({'age':'mean'})`
Grouping	`SELECT house, AVG(age), MAX(age) FROM students GROUP BY house`	`df.groupby('house').agg({'age': ['mean', 'max']})`
Joining	`SELECT * from students join houses on students.house_id = houses.id`	`pd.merge(df, df2, left_on = 'house_id', right_on = 'id')`

References

- https://www.datacamp.com/tutorial/pandas
- https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html