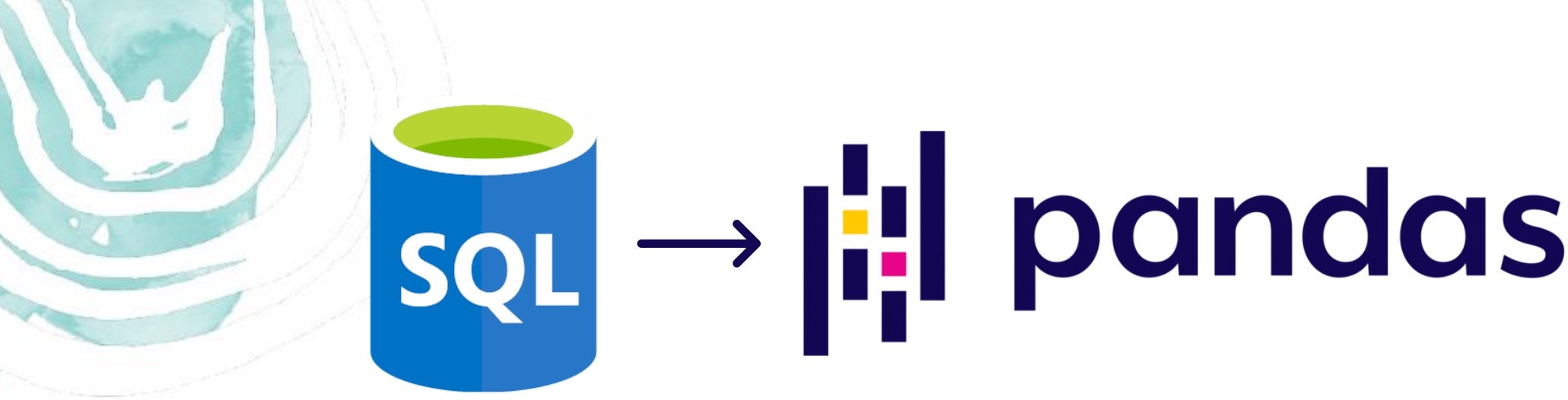


Introduction to pandas for SQL developers

Owen Price - linkedin.com/in/owenhprice

*SELECT*ing columns

```sql
SELECT column1, column2
FROM table;
```

```python
# 1 - slicing:
df[['column1', 'column2']]

# 2 - df.loc:
df.loc[:, ['column1', 'column2']]

# 3 - df.filter:
df.filter(['column1', 'column2'])
```

1. Slice the column axis of the DataFrame by passing a list of column names inside square brackets.

2. Access a group of rows and columns by label(s) or a Boolean array.

   The colon in the first position indicates “all rows”.

3. Subset rows or columns according to index labels.

   This first argument is ‘items’. If we omit the axis argument, the default axis is ‘columns’.

WHERE a condition is true

```sql
SELECT column1, column2
FROM table
WHERE column3 = 'value';
```

1 Here, we are passing the Boolean vector df['column3'] == 'value' as the row filter.

```python
# 1 - df.loc with Boolean vector to select rows:
df.loc[df['column3'] == 'value', ['column1', 'column2']]

# 2 - .eq function to create Boolean vector:
df.loc[df['column3'].eq('value'), ['column1', 'column2']]

# 3 - df.query:
df.query("column3 == 'value'")[['column1', 'column2']]
```

3 Query the columns of a DataFrame with a boolean expression.

2 Another way of creating the Boolean vector. i.e. if column 3 is equal to 'value'.

*JOIN*ing tables

```sql
SELECT t1.column1, t2.column4
FROM table1 t1
INNER JOIN table2 t2
ON t1.column3 = t2.column3;
```

1 Merge two DataFrames with (at minimum) a column or list of columns to join on.

```python
# 1 - pandas.merge:
pd.merge(df1, df2, on='column3')[['column1', 'column4']]

# 2 - df.merge:
df1.merge(df2, on='column3')[['column1', 'column4']]

# 3 - df.join:
df1.join(df2.set_index('column3'),
         on='column3')[['column1', 'column4']]
```

3 Joins columns of another DataFrame to a DataFrame.

2 Call merge as a method of a DataFrame.

GROUP BY columns

```sql
SELECT column1, AVG(column2)
FROM table
GROUP BY column1;
```

1 Group by column1, get mean of column2.

reset_index() so column1 is a column and not the index.

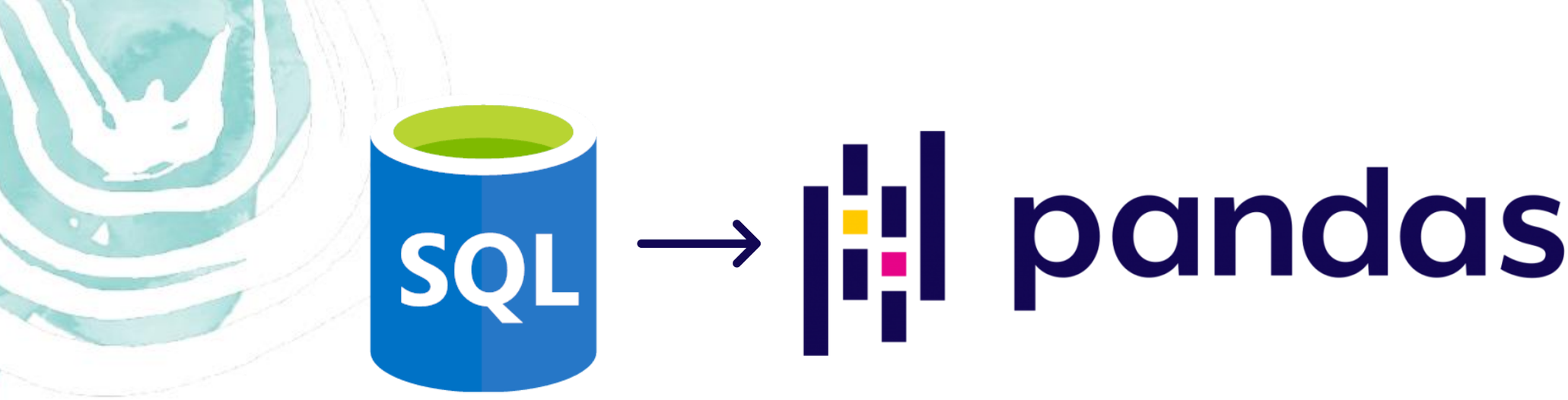
```python
# 1 - df.groupby
df.groupby('column1')['column2'].mean().reset_index()


# 2 - df.groupby alternative:
df.groupby('column1', as_index=False)['column2'].mean()


# 3 - explicit renaming of output column with df.groupby.agg:
df.groupby('column1').agg(col2_mean=('column2','mean')).reset_index()
```

3 Use .agg to rename output columns. Apply multiple aggregates to as many columns as needed.

2 Optionally use as_index=False to achieve the same result



Further reading

Indexing and selecting data

groupby

merge, join