

Unlearning SQL

Leverage the SQL design patterns in Python

S.Lott

<https://fosstodon.org/@slott56>

<https://github.com/slott56>

14-May-2025

Link to Slide Deck

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion



<https://github.com/slott56/unlearning-sql>

Table of Contents

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

1 SQL Overuse

2 SQL Design Patterns

3 Python Implementation Patterns

4 About Those Join Algorithms

5 Group By and Having — The Good Stuff

6 Conclusion

SQL is Helpful

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Folks know SQL.

- They are fluent in SQL's design patterns.
- Some find it hard to convert SQL designs to Python.

This talk should help clarify SQL from a Python perspective.

Database Overuse

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Given a processing problem...

Database Overuse

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Given a processing problem...

- ① Define (and normalize) tables
- ② Write, and debug a load script
- ③ Write, and debug the SQL

Easy, right?

Database Overuse

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Given a processing problem...

- ① Define (and normalize) tables
- ② Write, and debug a load script
- ③ Write, and debug the SQL

Easy, right?

Maybe not

SQL Overheads

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

The database engine has overheads

Lots of them.

Locking. Storage management. Permissions. Serialization.

War Story:

Developer struggling with transient data processing.

The app does repeated **Create-Load-Query-Drop** cycles.

The DROP (it turns out) is both unpredictable and slow.

(Even SQLite introduces overheads.)

How do we unlearn SQL?

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Two steps to moving past SQL:

- ① Understand the SQL design patterns.
- ② Rework those design elements in Python.

SQL Design Patterns

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Consider the core Select statement:

SELECT *expr*, ...

FROM *table*, ...

WHERE *condition*

We'll get to GROUP BY and HAVING later.

SELECT works like this

Unlearning
SQL

S.Lott

SQL Overuse

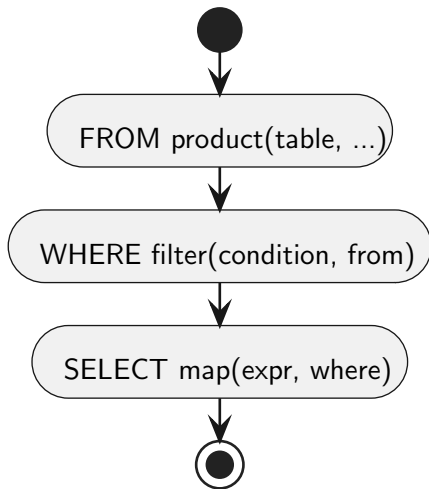
SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion



SELECT in Python

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

```
# FROM t1, t2, ...
from_ = itertools.product(t1, t2, ...)

# WHERE c
where = (row_tuple
         for row_tuple in from_
         if c(row_tuple))

# SELECT ex1, ex2, ...
result = list(
    (ex1(row_tuple), ex2(row_tuple), ...)
    for row_tuple in where)
```

Good and Bad

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

- The SQL maps directly to Python code.
- Python has more syntax:
`WHERE expr becomes (r for r in from_ if expr).`
- `SELECT expr, expr, expr` is even **more** complicated-looking.

Let's look at details.

All the syntax means there are a lot of places to add processing.

The From Clause

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Each of the FROM tables needs to be iterable sequences.

```
list[dict[str, Any]]
```

```
from itertools import product  
from_ = product(t1, t2, t3)
```

Yes. It's the Cartesian product.

Aha!

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

“Gotcha!”

- A *real* database doesn't do cartesian products all the time.
- It has fancy query algorithms and optimizations.

Aha!

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

“Gotcha!”

- A *real* database doesn't do cartesian products all the time.
- It has fancy query algorithms and optimizations.

“Your nonsense is clearly unworkable in general.”

Query optimization

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

- Requires extended syntax to suggest query optimizations.
- Require someone to design the right indexes.
- Requires detailed statistics on key distribution.

You **can** do query optimization in Python, also.

Query optimization

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

- Requires extended syntax to suggest query optimizations.
- Require someone to design the right indexes.
- Requires detailed statistics on key distribution.

You **can** do query optimization in Python, also.

We'll get to it.

The Where Condition

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Context:

```
where = (row for row in from_ if c(row))
```

Or.

```
where = filter(c, from_)
```

```
def c(row: tuple[dict[str, Any], ...]) -> bool:
    t1, t2, t3 = row
    return (
        t1['rowid'] == t2['foreign_key']
        and t2['some_key'] == t3['whatever']
    )
```

The Select Clause

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Context:

```
result = ((ex1(row_tuple), ex2(row_tuple), ...)
          for row_tuple in where)
```

Or.

```
result = map(row_builder, where)
```

```
def ex1(row: tuple[dict[str, Any], ...]) -> Any:
    t1, t2, t3 = row
    if t1['value'] % 2 == 0:
        return t1['value'] // 2
    else:
        return t1['value'] * 3 + 1
```

Key Point 1

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

You're not limited

You don't have to write SQL expressions.

You have the **Vast Python Ecosystem** available.

Key Point 1

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

You're not limited

You don't have to write SQL expressions.

You have the **Vast Python Ecosystem** available.

(Turn on echo effect)

Unlimited Computing Power!

Key Point 1

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

You're not limited

You don't have to write SQL expressions.

You have the **Vast Python Ecosystem** available.

(Turn on echo effect)

Unlimited Computing Power!

The `From` clause needs work.

The Cartesian Product Problem

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Databases do “cart-prod” joins all the time.

Very small tables are easier to fetch from disk into cache ignoring any indexes.

Two common alternative algorithms:

- Sort-Merge Join
- Lookup Join

Going to take a shallow look at each.

Sort-Merge Join

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

For very large tables.

- ① Sort each table into a consistent order by the join key for that table.
May need multiple files and the os utility **sort** command.
- ② Create row tuples for matching rows from each sorted table.

A variation on this can do any of the outer join algorithms.

See <https://toolz.readthedocs.io>; they offer `merge_sorted()`

Lookup Join

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Create for many small tables and one big table.
The “Star Schema” design pattern.

- ① Transform each small table into a Python dictionary.

```
small_1 = {r['pk']: row for row in table_1}
etc.
```

- ② Join.

```
from_ = (
    (r, small_1[r['fk_1']], small_2[r['fk_2']])
    for r in big_table
)
```

Key Point 2

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

You're not limited

You don't have to wrestle with database index and query optimizations.

You have the **Vast Python Ecosystem** available.

Key Point 2

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

You're not limited

You don't have to wrestle with database index and query optimizations.

You have the **Vast Python Ecosystem** available.

(Turn on echo effect)

Unlimited Computing Power!

Key Point 2

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

You're not limited

You don't have to wrestle with database index and query optimizations.

You have the **Vast Python Ecosystem** available.

(Turn on echo effect)

Unlimited Computing Power!

What about Group by and Having?
They can't be simple.

Group By Clause(s)

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

The **Group By** process involves two separate steps.

- 1 **Partition.** The expressions in the GROUP BY clause define keys to build groups.
- 2 **Aggregate.** The aggregate functions from the SELECT (and HAVING) are reduce() operations to create single group values.

Syntax Oddity: Group-By aggregates in the SELECT clause.
And in the HAVING clause.

Group By Implementation

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

```
from collections import defaultdict
from operator import itemgetter

# Partition

groups = defaultdict(list)
for row_tuple in where:
    key = (k_1(row_tuple), k_2(row_tuple), ...)
    groups[key].append(row_tuple)

# Aggregate

group_by = []
for key, group in groups:
    agg_1 = some_function(group)
    agg_2 = mean(row['value'] for row in group)
    agg_3 = sum(map(itemgetter('name'), group))
    group_by.append((key, agg_1, agg_2, agg_3))
```

Having Clause

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

The **Having** process is (nearly) the same as the **Where** process. It's an expression to filter the groups.

SQL syntax uses aggregate functions in the HAVING clause.

- These are yet more group-by aggregates.
- The result values are only used for filtering.

Key Point 3

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

You're not limited

The Group-By operation is a `defaultdict(list)`.
Maybe a Counter.

You have the **Vast Python Ecosystem** available.

Key Point 3

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

You're not limited

The Group-By operation is a `defaultdict(list)`.
Maybe a Counter.

You have the **Vast Python Ecosystem** available.

(Turn on echo effect)

Unlimited Computing Power!

Conclusion

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

- SQL has helpful patterns to describe a desired result.
- SQL can be limiting.
- An actual database engine introduces a lot of overhead. Avoid it.

Think of SQL as a design language.

Conclusion

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

- SQL has helpful patterns to describe a desired result.
- SQL can be limiting.
- An actual database engine introduces a lot of overhead. Avoid it.

Think of SQL as a design language.
Not an implementation choice.

SQL

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

- SQL describes a pipeline of steps:

From \rightarrow Where \rightarrow Select \rightarrow Group By \rightarrow Having

- Or, nested functions:

$$H\left(G_a\left(G_p\left(S\left(W\left(F(t_1, t_2, \dots)\right)\right)\right)\right)\right)$$

Important: The select-from-where ordering of clauses is confusing.

That's not how it works.

SQL to Python

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

In Python, Select is a stack of generator expressions

```
from_ = itertools.product(...)
where = filter(condition, from_)
select = map(row_builder, where)
groups = group_reduce(select)
aggregates = map(agg_row_builder, groups)
result = filter(having_condition, aggregates)
```

Most steps are lazy and don't compute big intermediate results.

SQL to Python

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

In Python, Select is a stack of generator expressions

```
from_ = itertools.product(...)
where = filter(condition, from_)
select = map(row_builder, where)
groups = group_reduce(select)
aggregates = map(agg_row_builder, groups)
result = filter(having_condition, aggregates)
```

Most steps are lazy and don't compute big intermediate results.
The `group_reduce()` function does compute a big result.

Call to Action

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

Stop using SQL (and a database) as a data transformation tool.

Continue using SQL as a design aid.

SQL design patterns are useful.

Look at SQL as a pipeline of functional transformations.

More Information

Unlearning
SQL

S.Lott

SQL Overuse

SQL Design
Patterns

Python Imple-
mentation
Patterns

About Those
Join
Algorithms

Group By and
Having — The
Good Stuff

Conclusion

- *Unlearning SQL* book (Available from Amazon and Lulu)
- <https://github.com/slott56/functional-SQL>
- <https://github.com/slott56/unlearning-sql>
- <https://fosstodon.org/@slott56>

