# Say No To SQL

The main idea is to implement an SQL select statement generating program from the given input and output example.

There is this tool: <https://github.com/Mestway/Scythe>

Which does the same in python, but it doesn’t find out the best SQL query. So no ranking and cannot be useful for a layman with no prior knowledge of SQL.

We want to have a C# implementation which can generate SELECT queries.

## What are we using?

PROSE (**PR**ogram **S**ynthesis using **E**xample) is an SDK which can do program synthesis using example.

Read more about PROSE here : <https://microsoft.github.io/prose/documentation/prose/tutorial/>

This is written in C#

## Essential Parts?

### DSL:

We need a domain specific language to limit our search space for programs

### Witness Function:

Inverse function for each operator in the DSL.

### Ranking Function:

Choose the best program from all the possible outcomes.

## Our Grammar:

Any general SELECT statement looks like the following:

SELECT select\_list

[ INTO new\_table\_name ]

FROM table\_list

[ WHERE search\_conditions ]

[ GROUP BY group\_by\_list ]

[ HAVING search\_conditions ]

[ ORDER BY order\_list [ ASC | DESC ] ]

This changes to the following our grammar:

@input TableList<Table> tableList;

@start sql := Aggregation(select) //We won’t implement it, it has SUM, COUNT and other stuff

| select

select := SelectWithoutWhere(columnList, tableList) // No WHERE clause, SELECT columnList FROM tableList

|SelectWithWhere(whereClause) // for statements like SELECT columnList FROM tableList WHERE condition

whereClause := Where(condition) // WHERE condition , it will return the whole table with all columns

condition := cmpStatement // This has one single the comparison statement

|Logical(cmpStatement, @recurse[5] condition, logicSymbol) // combining two comparison statements

cmpStatement = Comparator(columnList, column, tableList, cmpSymbol) // column.name [cmpSymbol] Const

cmpSymbol := ‘>’|’<’|’>=’|’<=’|’==’

logicSymbol := ‘AND’|’OR’

List<Column> columnList;

Column column

**Table**:

**List<Column>** Columns

**Object[][]** DataPoints

**Column:**

**string** Name

**Type** Type //Only string and reals (doubles and int)

**#Note:** the data types for cmpStatement, condition, whereClause, select and sql all are **Table**.