SQL rocks!

- “Everybody” knows SQL
- All major data management and database systems support SQL

→ TableAPI
Table clicksTable = tableEnv.toTable(clicks, "url, userId");

Table clickCounts = clicks
  .groupBy("user")
  .select("userId, url.count as count")
  .filter("count > 4");

Table activeUsers = users.join(clickCounts)
  .where("id = userIdId && count > 10")
  .select("username, count");
Under the Hood

Logical Plan

\[ a = b \&\& c > 3 \]

Execution Plan
Plan Translation

Aggregate expansion

Plan generation

Batch Program

Streaming Program
String operations

- **Sub strings:**
  
  "hello world" → "hello"

  ```
in.select("a.substring(0, b.avg + 3) as aShort")
  ```

- **String concatenation:**
  
  "hello" + " world" → "hello world"

  ```
in.select("a.count + ' is the count'")
  ```
Expression Translation

\[ a = c \land b > 3 \]

\[
\text{def rules} = \text{Seq(}
\begin{array}{l}
\text{new ResolveFieldReferences,}
\text{new InsertAutoCasts,}
\text{new TypeCheck,}
\text{new VerifyNoAggregates,}
\text{new VerifyBoolean,}
\text{new ExtractEquiJoins}
\end{array}
\)
\]

\[
\text{in1.join(in2).where(0).equalTo(0) } = \text{ join(\{0\}, \{0\})}
\]

\[
\text{if (r.get(1).asInstanceOf[Int] > 3) {}
\text{val result = Row(4)}
\text{result.set(0, l.get(0))}
\text{...}
\text{result.set(3, r.get(1))}
\text{out.collect(result)}
\text{}}
\]
public static class Out {
    public String c;
    public Integer d;
}

DataSet<Tuple2<Integer,String>> input = ...

TableEnvironment tableEnv = new TableEnvironment();
Table in = tableEnv.toTable(input,"a,b");
Table result = in.groupBy("b").select("b, a.avg");

DataSet<Out> result =
    tableEnv.toSet(result.as("c,d"), Out.class);

- Supports POJOs, Case classes, Tuples
What Works?

- Relational queries from both Java and Scala
- Translation to batch programs
- Preliminary translation to streaming programs
Future Work

- Relational Optimization
  - Filter/Projection push down
  - Join order
- Operator Fusion
- Extend expressions
  - string operations, casting, explode/gather, date/time, ...
- Windowing operations (streaming)
- Columnar execution?
- SQL
Hands-on

- **Task:**
  - A German bus company asks for your help.
  - They provide 20 movies in their bus entertainment center. They want to have a nice chart about the average rating for all movies which have been rated by more than 1000 passengers.

- **Download the git repository:**
  
  ```
  git clone https://github.com/FelixNeutatz/flink-hands-on-TableAPI.git
  ```

  ```
  Start IntelliJ → File → Open → ../flink-hands-on-TableAPI/pom.xml
  Complete de.tubulin.dima.flinkhandson.tables.TablesSolution
  ```

  **Bonus Task:** Generate a similar chart for a single genre :)