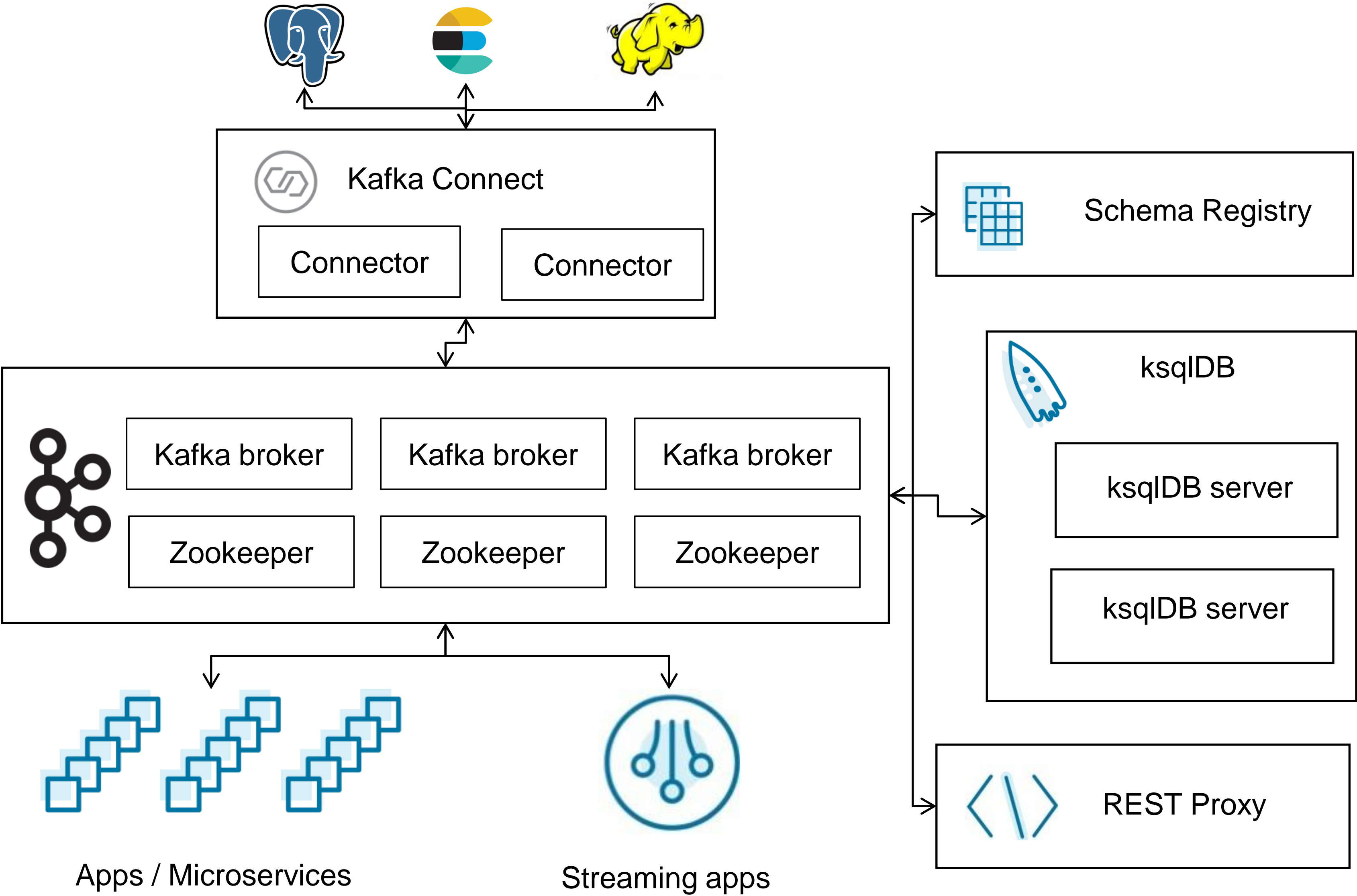
JavaCro'21

Streaming with ksqlDB

Ivan Turčinović
ivan.turcinovic@inovatrend.com

JavaCro'21

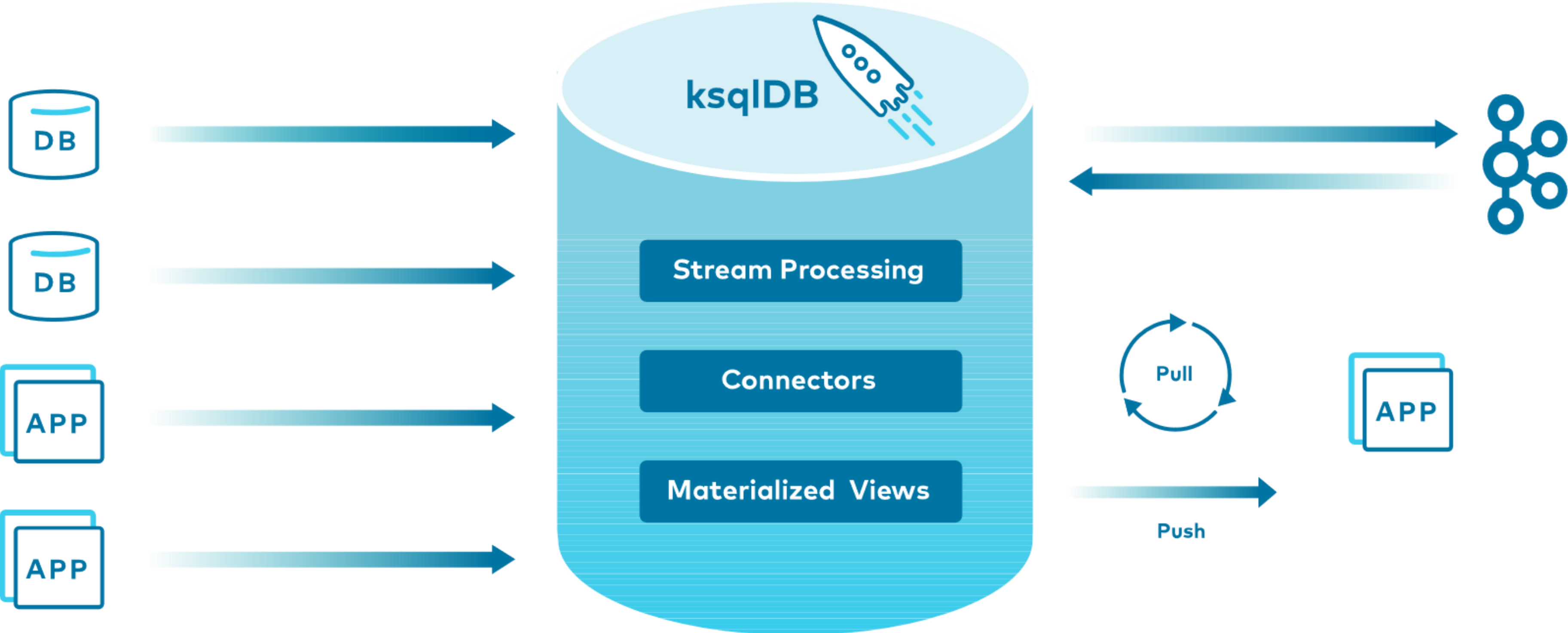
Kafka ecosystem



INNOVATREND inovativni trendovi d.o.o.

JavaCro'21

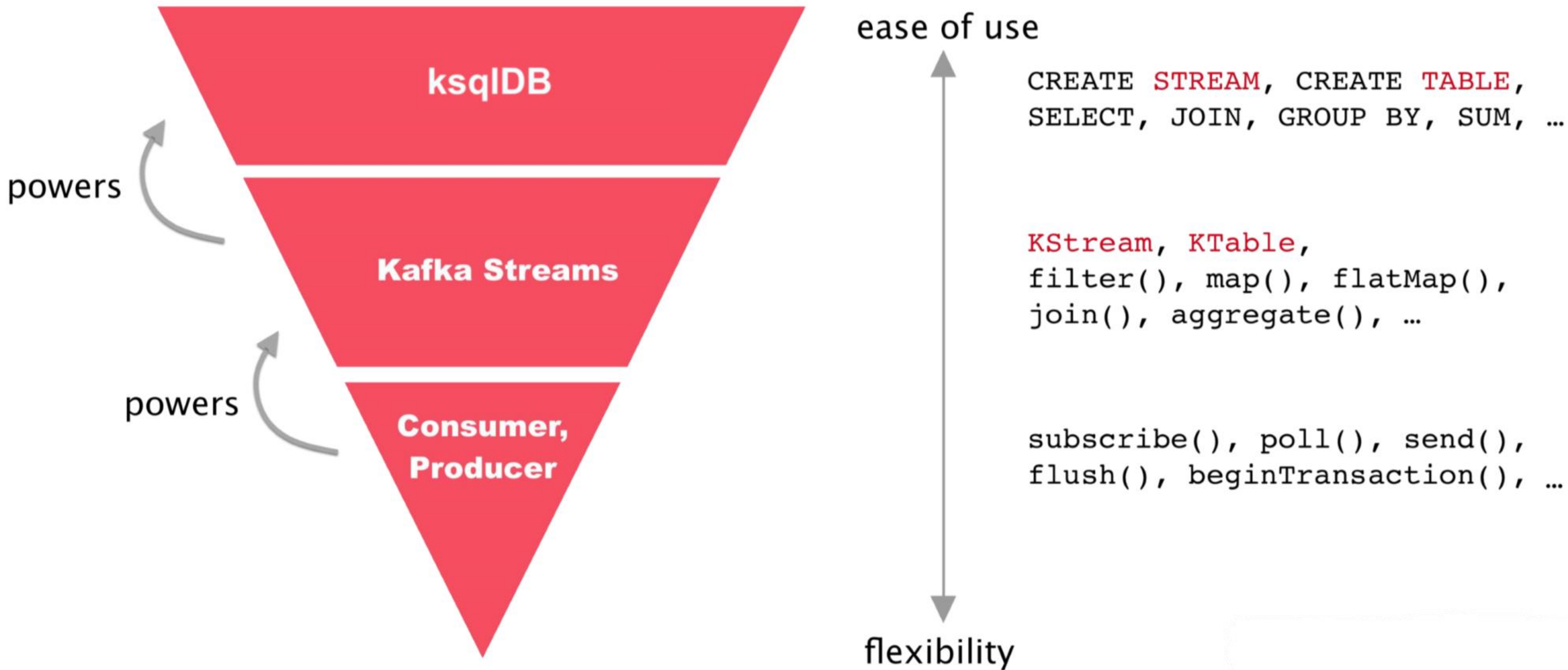
ksqlDB



https://images.ctfassets.net/8vofjvai1hpv/5KfaCoL62wOmqFFDgnvkK6/0167db54b960de083cc8ef1a782e6010/20210121-DIA-DEV_ksqlDB.svg

INNOVATREND Inovativni trendovi d.o.o.

ksqlDB streams stack



https://docs.ksqldb.io/en/latest/img/ksqldb-kafka-streams-core-kafka-stack.png

INNOVATREND Inovativni trendovi d.o.o.

Stream – Table duality

- An event stream records the history of what has happened as a sequence of events
- A table represents the state at a particular point in time, typically "now."
- We can turn a stream into a table by aggregating the stream
- We can turn a table into a stream by capturing the changes made to the table—inserts, updates, and deletes—into a "change stream."

Streams record history

"The sequence of moves"

Tables represent state

"The state of the board"

https://www.confluent.io/blog/kafka-streams-tables-part-1-event-streaming/
https://www.confluent.io/blog/streams-tables-two-sides-same-coin/

REST API/Clients

REST API - allows clients to interact with the ksqlDB server

Deployment:

- Interactive mode
  - can submit new queries anytime by using the REST API.
  - interactive experience, allows the ksqlDB server to create and tear down streams, tables, queries, and connectors dynamically
- Headless mode
  - creating a file containing any persistent queries you want the SQL engine to execute
  - Static

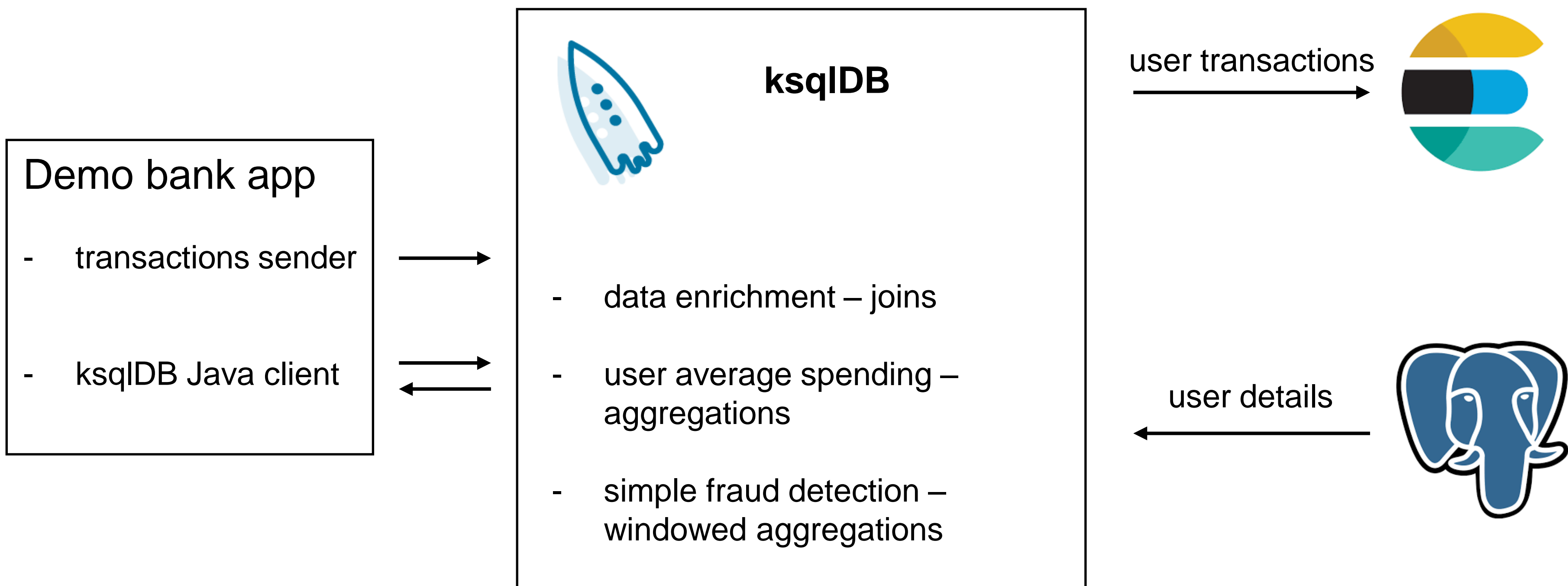
ksqlDB Clients

- ksqlDB CLI
- ksqlDB UI – available only on Confluent platfom
- Java Client - official client
- .NET Client - unofficial client. contributed and maintained by community members

JavaCro'21

Demo

Demo



INOVATREND inovativni trendovi d.o.o.

Materialized views

- Same as relational DB views:
  - derived from a query against another collection.
  - can be queried (called pull queries in ksqlDB).
- Different from relational DB views:
  - for now could only be computed from aggregate queries.
  - refreshed automatically as new data comes in
- Materialized view is a TABLE in ksqlDB

Connector management

- ksqlDB gives you ability to configure and manage Kafka Connect Connectors
- ksqlDB can run connectors in two different modes: embedded or external.

```
CREATE SOURCE CONNECTOR jdbc_bank_client WITH (
    "connector.class"='io.confluent.connect.jdbc.JdbcSourceConnector',
    "connection.url"='jdbc:postgresql://localhost:5433/bank',
    "connection.user"= 'bank',
    "connection.password"= 'bank',
    "mode"='incrementing',
    "incrementing.column.name"='id',
    "topic.prefix"='jdbc_',
    "table.whitelist"='bank_client',
    "key"='id',
    "key.converter" = 'org.apache.kafka.connect.converters.LongConverter'
);
```

JavaCro'21

Stream processing – Create collections

```
CREATE STREAM bank_transactions_stream
    WITH (
     KAFKA_TOPIC='bank_transactions',
     VALUE_FORMAT='AVRO'
    );

CREATE TABLE bank_clients_table (
      USER_ID BIGINT PRIMARY KEY,
      FIRST_NAME STRING,
      LAST_NAME STRING,
       ADDRESS STRING
   ) WITH (
     KAFKA_TOPIC = 'jdbc_bank_client',
     VALUE_FORMAT = 'AVRO');
```


INNOVATREND
Inovativni trendovi d.o.o.

Stream processing – Queries

Pull queries:

- retrieve results at a point in time - “now” – at time of query execution.

```
SELECT * FROM bank_client_avg_spending WHERE  userid = 32
```

Push queries:

- stream a query result changes to the client

```
SELECT * FROM bank_client_avg_spending WHERE  userid = 32
emit changes;
```

Persistent Queries:

- Query which write result back to Kafka.

```
CREATE STREAM kiehn_transactions AS
  SELECT * FROM bank_transactions_enriched
     WHERE lastname = 'Kiehn'
EMIT CHANGES;
```


INNOVATREND inovativni trendovi d.o.o.

Stream processing – joins

- Stateful operation
- INNER JOIN, LEFT JOIN, FULL JOIN

```
CREATE STREAM bank_transactions_enriched AS
  SELECT
        bank_transactions_stream.userid,
        bank_transactions_stream.amount,
        bank_transactions_stream.accountnumber,
        bank_transactions_stream.merchantname,
        bank_transactions_stream.timestamp,
        bank_clients_table.first_name AS firstname,
        bank_clients_table.last_name  AS lastname,
        bank_clients_table.address    AS address
  FROM  bank_transactions_stream
  INNER JOIN bank_clients_table
  ON bank_transactions_stream.userid = bank_clients_table.u
ser_id
EMIT CHANGES;
```


iNNOVATREND Inovativni trendovi d.o.o.

Stream processing – aggregations

Two steps for aggregating data:

- Create a SELECT statement that has some aggregate function.
- Group related records using the GROUP BY clause.

```
CREATE TABLE bank_client_avg_spending AS
  SELECT
    USERID,
    AVG(AMOUNT) AS avg
  FROM bank_transactions_enriched
  GROUP BY userid
EMIT CHANGES;
```


INOVATREND Inovativni trendovi d.o.o.

Stream processing – windowed aggregations

Window types:

- Tumbling, Hopping, Session

```
CREATE TABLE bank_client_possible_fraud
    WITH ( KAFKA_TOPIC= 'bank_client_possible_fraud')
        AS
    SELECT userid,
           count(*) AS n_attempts,
           SUM(amount) AS total_amount,
           collect_list(merchantname) AS merchants,
           windowstart AS start_boundary,
           windowend AS end_boundary
        FROM bank_transactions_enriched
    WINDOW TUMBLING(size 30 seconds, retention 10 days)
        GROUP BY userid
        HAVING   count(*) >= 3
EMIT CHANGES;
```


INNOVATREND Inovativni trendovi d.o.o.

UDF

- ksqlDB has a lot of built-in functions
  - AVG, CEIL, CONCAT, COUNT, EARLIEST_BY_OFFSET …
- To list the available functions, we can use the SHOW FUNCTIONS statement

User Defined Functions:

- Java interface that enables you to write your own functions
- User-defined functions, or UDFs
  - Scalar functions
  - Tabular functions – UDTFs
  - Aggregation functions - UDAFs

INOVATREND Inovativni trendovi d.o.o.

Thank You!