```
# Import an ML model and a data set

type Item = {name:String, price:f64, mass:f64};
type Order = {id:u32, items: Vec[Item], t:Time, src_id:u32, dst_id:u32};
type Area = {id:u32, lon:f32, lat:f32};

# Import an ML model and a data set
val model: Model[[f32;3],[f32;1]] = model(path("cost_predictor.pb"), tf());
val areas: Set[Area] = set(path("areas.csv"), csv());

# Create an external Rust UDF for filtering
---rust
fn is_valid_order(o: Order) -> bool { /* Rust code */ }
---

from o:Order in source(kafka("127.0.0.1:9092", "orders"), json())
where is_valid_order(o)
join a0 in areas on a0.id == o.src_id
join a1 in areas on a1.id == o.dst_id
val features = [o.items.sum_by(_.mass), f32(o.t.day()), distance(a0, a1)]
val [shipment_cost] = model.predict(features)
select {o.id, shipment_cost}
into sink(kafka("127.0.0.1:9092", "cost_predictions"), json());
```