KIP-149 / 159 / 182: what should they look like in the end

This is a tentative summary of what the DSL would be end-up looking like after these three KIPs.

Things to think about:

1. For the RichValueXX / RichInitializer / RichReducer functions under discussion in KIP-159, whether it should also include the key in addition to the context itself? Here are my thoughts about pros and cons:

a) If we do not do that, we would have overloaded function as ValueFunc / ValueFuncWithKey / RichValueFunc / RichValueFuncWithKey, to accept value / key+value / context+value / context+key+value. This is simply too many overloads.

b) If we have only two overloads, with value (or key+value, for KeyValueMapper, etc) and key+value+context, then for people who only wants to access key+value or context+value, they need to use the overload function with dummy parameters (i.e. "key, value, _" in Scala).

The following first draft assumes a). But under that option, the next question to ask is would we be subsuming all of KIP-149 in KIP-159 then?

2. In the current KIP-159 design doc, for "aggregate" functions we enrich the `Aggregator` with `RichAggregator` but not `Initializer` with `RichInitializer`. I think it is better to also have RichInitializer for aggregate functions where we are using RichAggregators. So the overloads would be, for example "initializer+aggregator" and "richInitializer+richAggregator".

KStream Non-stateful

```
<VR> KStream<K, VR> mapValues(final ValueMapper<? super V, ? extends VR> mapper);                      // old
<VR> KStream<K, VR> mapValues(final ValueMapperWithKey<? super K, ? super V, ? extends VR> mapper);    // KIP-
149
<VR> KStream<K, VR> mapValues(final RichValueMapper<? super K, ? super V, ? extends VR> mapper);       // KIP-
159

<VR> KStream<K, VR> transformValues(final ValueTransformerSupplier< ? super V, ? extends VR>
valueTransformerWithKeySupplier, final String... stateStoreNames);                       // old
<VR> KStream<K, VR> transformValues(final ValueTransformerWithKeySupplier<? super K, ? super V, ? extends VR>
valueTransformerWithKeySupplier, final String... stateStoreNames);      // KIP-149
<VR> KStream<K, VR> transformValues(final RichValueTransformerSupplier<? super K, ? super V, ? extends VR>
valueTransformerWithKeySupplier, final String... stateStoreNames);         // KIP-159

<VR> KStream<K, VR> flatMapValues(final ValueMapper<? super V, ? extends Iterable<? extends VR>>
processor);                          // old
<VR> KStream<K, VR> flatMapValues(final ValueMapperWithKey<? super K, ? super V,
? extends Iterable<? extends VR>> processor);       // KIP-149
<VR> KStream<K, VR> flatMapValues(final RichValueMapper<? super K, ? super V, ? extends Iterable<? extends VR>>
processor);          // KIP-159

KStream<K, V> filter(Predicate<? super K, ? super V> predicate);              // old
KStream<K, V> filter(RichPredicate<? super K, ? super V> predicate);          // KIP-159
KStream<K, V> filterNot(Predicate<? super K, ? super V> predicate);           // old
KStream<K, V> filterNot(RichPredicate<? super K, ? super V> predicate);       // KIP-159

<KR> KStream<KR, V> selectKey(KeyValueMapper<? super K, ? super V, ? extends KR> mapper);           // old
<KR> KStream<KR, V> selectKey(RichKeyValueMapper<? super K, ? super V, ? extends KR> mapper);       // KIP-159

<KR, VR> KStream<KR, VR> map(KeyValueMapper<? super K, ? super V, ? extends KeyValue<? extends KR,
? extends VR>> mapper);          // old
<KR, VR> KStream<KR, VR> map(RichKeyValueMapper<? super K, ? super V, ? extends KeyValue<? extends KR,
? extends VR>> mapper);      // KIP-159

<KR, VR> KStream<KR, VR> flatMap(final KeyValueMapper<? super K, ? super V,
? extends Iterable<? extends KeyValue<? extends KR, ? extends VR>>> mapper);            // old
<KR, VR> KStream<KR, VR> flatMap(final RichKeyValueMapper<? super K, ? super V,
? extends Iterable<? extends KeyValue<? extends KR, ? extends VR>>> mapper);        // KIP-159

void print(final Printed<K, V> printed);                                                 // KIP-182
void foreach(final ForeachAction<? super K, ? super V> action);                          // old
void foreach(final RichForeachAction<? super K, ? super V> action);                      // KIP-159
KStream<K, V> peek(final ForeachAction<? super K, ? super V> action);                    // old
KStream<K, V> peek(final RichForeachAction<? super K, ? super V> action);                // KIP-159
KStream<K, V>[] branch(final Predicate<? super K, ? super V>... predicates);             // old
KStream<K, V>[] branch(final RichPredicate<? super K, ? super V>... predicates);         // KIP-159

KStream<K, V> through(final String topic);                                               // old
KStream<K, V> through(final String topic, final Produced<K, V> partitioned);             // KIP-182
void to(final String topic);                                                             // old
void to(final String topic, final Produced<V, V> partitioned);                           // KIP-182
```

KTable Non-stateful

```
KTable<K, V> filter(final Predicate<? super K, ? super V>
predicate);                                                                    // old
KTable<K, V> filter(final RichPredicate<? super K, ? super V>
predicate);                                                                  // KIP-159
KTable<K, V> filter(final Predicate<? super K, ? super V> predicate, final Materialized<K, V, KeyValueStore<K,
V>> materialized);         // KIP-182
KTable<K, V> filter(final RichPredicate<? super K, ? super V> predicate, final Materialized<K, V,
KeyValueStore<K, V>> materialized);     // KIP-182/159

KTable<K, V> filterNot(final Predicate<? super K, ? super V>
predicate);                                                                     // old
KTable<K, V> filterNot(final RichPredicate<? super K, ? super V>
predicate);                                                                  // KIP-159
KTable<K, V> filterNot(final Predicate<? super K, ? super V> predicate, final Materialized<K, V,
KeyValueStore<K, V>> materialized);      // KIP-182
KTable<K, V> filterNot(final RichPredicate<? super K, ? super V> predicate, final Materialized<K, V,
KeyValueStore<K, V>> materialized);  // KIP-182/159

<VR> KTable<K, VR> mapValues(final ValueMapper<? super V, ? extends VR>
mapper);                                                                  // old
<VR> KTable<K, VR> mapValues(final ValueMapper<? super V, ? extends VR> mapper, final Materialized<K, V,
KeyValueStore<K, V>> materialized);  // KIP-182
<VR> KTable<K, VR> mapValues(final RichValueMapper<? super V, ? extends VR>
mapper);                                                                // KIP-159
<VR> KTable<K, VR> mapValues(final RichValueMapper<? super V, ? extends VR> mapper, final Materialized<K, V,
KeyValueStore<K, V>> materialized);    // KIP-182/159

<KR> KStream<KR, V> toStream(final KeyValueMapper<? super K, ? super V, ? extends KR> mapper);           // old
<KR> KStream<KR, V> toStream(final RichKeyValueMapper<? super K, ? super V, ? extends KR> mapper);       // KIP-
159

void to(final String topic);                                 // old
void to(final String topic, final Produced<V, V> options);   // KIP-182

KTable<K, V> through(final String topic);                                    // old
KTable<K, V> through(final String topic, final Materialized<K, V> options);  // KIP-182
```

KStream Joins

```
<VO, VR> KStream<K, VR> join(final KStream<K, VO> other, final ValueJoiner<? super V, ? super VO, ? extends VR>
joiner, final JoinWindows windows, final Joined<K, V, VO> options); // KIP-182
<VT, VR> KStream<K, VR> join(final KTable<K, VT> other, final ValueJoiner<? super V, ? super VT, ? extends VR>
joiner, final Joined<K, V, VT> options);                             // KIP-182

<VO, VR> KStream<K, VR> join(final KStream<K, VO> other, final ValueJoinerWithKey<? super V, ? super VO,
? extends VR> joiner, final JoinWindows windows, final Joined<K, V, VO> options);   // KIP-182/149, can this be
subsumed by KIP-159?
<VT, VR> KStream<K, VR> join(final KTable<K, VT> other, final ValueJoinerWithKey<? super V, ? super VT,
? extends VR> joiner, final Joined<K, V, VT> options);                               // KIP-182/149, can this
be subsumed by KIP-159?

<VO, VR> KStream<K, VR> join(final KStream<K, VO> other, final RichValueJoiner<? super V, ? super VO,
? extends VR> joiner, final JoinWindows windows, final Joined<K, V, VO> options);      // KIP-182/159
<VT, VR> KStream<K, VR> join(final KTable<K, VT> other, final RichValueJoiner<? super V, ? super VT,
? extends VR> joiner, final Joined<K, V, VT> options);                                  // KIP-182/159

// same for KStream#leftJoin, KStream#outerJoin
```

KTable Joins

```
<VO, VR> KTable<K, VR> join(final KTable<K, VO> other, final ValueJoiner<? super V, ? super VO, ? extends VR>
joiner, final Materialized<K, VR, KeyValueStore<K, VR>> materialized);           // KIP-182
<VO, VR> KTable<K, VR> join(final KTable<K, VO> other, final ValueJoinerWithKey<? super V, ? super VO,
? extends VR> joiner, final Materialized<K, VR, KeyValueStore<K, VR>> materiaized);     // KIP-182/149, can
this be subsumed by KIP-159?
<VO, VR> KTable<K, VR> join(final KTable<K, VO> other, final RichValueJoiner<? super V, ? super VO,
? extends VR> joiner, final Materialized<K, VR, KeyValueStore<K, VR>> materialized);       // KIP-182/159

// same for KTable#leftJoin, KTable#outerJoin
```

KStream Aggregations

```
KGroupedStream<K, V> groupByKey();                                    // old
KGroupedStream<K, V> groupByKey(final Serialized<K, V> serialized);  // KIP-182

<KR> KGroupedStream<KR, V> groupBy(final KeyValueMapper<? superK, ? superV, KR>
selector);                                        // old
<KR> KGroupedStream<KR, V> groupBy(final KeyValueMapper<? superK, ? superV, KR> selector, final Serialized<KR,
V> serialized);   // KIP-182
<KR> KGroupedStream<KR, V> groupBy(final RichKeyValueMapper<? superK, ? superV, KR>
selector);                                       // old
<KR> KGroupedStream<KR, V> groupBy(final RichKeyValueMapper<? superK, ? superV, KR> selector, final
Serialized<KR, V> serialized);  // KIP-182
```

KGroupedStream / WindowedKStream Aggregations

For WindowedKStream the only difference is typed K to Windowed<K>, and hence is omitted here:

```
KTable<K, Long> count();                                                    // old
KTable<K, Long> count(final Materialized<K, Long> materialized);    // KIP-182

KTable<K, V> reduce(final Reducer<V> reducer);                                                                  //
old
KTable<K, V> reduce(final Reducer<V> reducer, final Materialized<K, V, KeyValueStore<K, V>> materialized);   //
KIP-182

KTable<K, V> reduce(final ReducerWithKey<V>
reducer);                                                                    // KIP-149, can this be subsumed by
KIP-159?
KTable<K, V> reduce(final ReducerWithKey<V> reducer, final Materialized<K, V, KeyValueStore<K, V>>
materialized);     // KIP-182/149, can this be subsumed by KIP-182/159?

KTable<K, V> reduce(final RichReducer<V>
reducer);                                                                  // KIP-159
KTable<K, V> reduce(final RichReducer<V> reducer, final Materialized<K, V, KeyValueStore<K, V>>
materialized);    // KIP-182/159

<VR> KTable<K, VR> aggregate(final Initializer<VR> initializer,
                             final Aggregator<? super K, ? super V, VR> aggregator);                // old
<VR> KTable<K, VR> aggregate(final Initializer<VR> initializer,
                             final Aggregator<? super K, ? super V, VR> aggregator,
                             final Materialized<K, VR, KeyValueStore<K, VR>> materialized);         // KIP-182

<VR> KTable<K, VR> aggregate(final InitializerWithKey<K, VR> initializer,
                             final Aggregator<? super K, ? super V, VR> aggregator);                // KIP-149,
can this be subsumed?
<VR> KTable<K, VR> aggregate(final InitializerWithKey<K, VR> initializer,
                             final Aggregator<? super K, ? super V, VR> aggregator,
                             final Materialized<K, VR, KeyValueStore<K, VR>> materialized);         // KIP-182
/149, can this be subsumed?

<VR> KTable<K, VR> aggregate(final RichInitializer<K, VR> initializer,
                             final Aggregator<? super K, ? super V, VR> aggregator);                // KIP-159
<VR> KTable<K, VR> aggregate(final RichInitializer<K, VR> initializer,
                             final RichAggregator<? super K, ? super V, VR> aggregator,
                             final Materialized<K, VR, KeyValueStore<K, VR>> materialized);         // KIP-182
/159
```

KTable Aggregations

```
<KR, VR> KGroupedTable<KR, VR> groupBy(final KeyValueMapper<? super K, ? super V, KeyValue<KR, VR>>
selector);                                      // old
<KR, VR> KGroupedTable<KR, VR> groupBy(final KeyValueMapper<? super K, ? super V, KeyValue<KR, VR>> selector,
Serialized<KR, VR> serialized);      // KIP-182
<KR, VR> KGroupedTable<KR, VR> groupBy(final RichKeyValueMapper<? super K, ? super V, KeyValue<KR, VR>>
selector);                                    // KIP-159
<KR, VR> KGroupedTable<KR, VR> groupBy(final RichKeyValueMapper<? super K, ? super V, KeyValue<KR, VR>>
selector, Serialized<KR, VR> serialized);  // KIP-182/159
```

KGroupedTable Aggregations

```
KTable<K, Long> count();                                                                    // old
KTable<K, Long> count(final Materialized<K, V, KeyValueStore<K, V>> materialized);      // KIP-182

KTable<K, V> reduce(final Reducer<V> adder, final Reducer<V>
subtractor);                                                                  // old
KTable<K, V> reduce(final Reducer<V> adder, final Reducer<V> subtractor, final Materialized<K, V,
KeyValueStore<K, V>> materialized);    // KIP-182

KTable<K, V> reduce(final RichReducer<V> adder, final RichReducer<V>
subtractor);                                                                  // KIP-159
KTable<K, V> reduce(final RichReducer<V> adder, final RichReducer<V> subtractor, final Materialized<K, V,
KeyValueStore<K, V>> materialized);    // KIP-182/KIP-149

<VR> KTable<K, VR> aggregate(final Initializer<VR> initializer,
                             final Aggregator<? super K, ? super V, VR> aggregator,
                             final Aggregator<? super K, ? super V, VR> subtractor);              // old
<VR> KTable<K, VR> aggregate(final Initializer<VR> initializer,
                             final Aggregator<? super K, ? super V, VR> aggregator,
                             final Aggregator<? super K, ? super V, VR> subtractor,
                             final Materialized<K, VR, KeyValueStore<K, VR>> materialized);       // KIP-182
<VR> KTable<K, VR> aggregate(final RichInitializer<VR> initializer,
                             final RichAggregator<? super K, ? super V, VR> aggregator,
                             final RichAggregator<? super K, ? super V, VR> subtractor);          // KIP-159
<VR> KTable<K, VR> aggregate(final RichInitializer<VR> initializer,
                             final RichAggregator<? super K, ? super V, VR> aggregator,
                             final RichAggregator<? super K, ? super V, VR> subtractor,
                             final Materialized<K, VR, KeyValueStore<K, VR>> materialized);       // KIP-182/159
```