

LINQ Query Operators Cheat Sheet

Aggregation Operators

OPERATOR	DESCRIPTION
Aggregate	Returns a value by applying an accumulator function over the sequence ¹
Average	Returns the average value of the elements in the sequence – <i>or</i> – the average as determined by the specified transform function ²
Count	Returns the count of the number of elements in the sequence – <i>or</i> – the count where the specified predicate function is <i>true</i> ³
LongCount	Returns the count of the number of elements in the sequence – <i>or</i> – the count where the specified predicate function is <i>true</i>
Max	Returns the maximum element in the sequence – <i>or</i> – the maximum as determined by the specified transform function ²
Min	Returns the minimum element in the sequence – <i>or</i> – the minimum as determined by the specified transform function ²
Sum	Returns the sum of the elements in the sequence – <i>or</i> – the sum as determined by the specified transform function ²

- 1) Throws an exception when the sequence is empty and no seed value is specified
 2) Throws an exception when the sequence is empty and TSource is not nullable
 3) Use LongCount when you want to allow the result to be greater than Int32.MaxValue

Conversion Operators

OPERATOR	DESCRIPTION
AsEnumerable	Returns the sequence as IEnumerable<TSource>
Cast	Returns an IEnumerable<TResult> containing the sequence cast to the specified type ⁴
OfType	Returns an IEnumerable<TResult> containing the elements of the sequence filtered by the specified type
ToArray	Returns an array TSource[] from the sequence
ToDictionary	Returns a Dictionary<TKey, TSource> from the sequence using the specified key and optionally element selector functions ⁵
ToList	Returns a List<TSource> from the sequence
ToLookup	Returns a Lookup<TKey, TSource> from the sequence using the specified key and optionally element selector functions

- 4) Throws an exception when an element cannot be cast to TResult
 5) Throws an exception when the key selector produces a key that is null or a duplicate key

Element Operators

OPERATOR	DESCRIPTION
DefaultIfEmpty	Returns the sequence; otherwise it returns an IEnumerable<TSource> containing the default element if the sequence is empty ⁶
ElementAt	Returns the element in the sequence at the specified index ⁷
ElementAtOrDefault	Returns the element in the sequence at the specified index; otherwise it returns the default element if the index is out of bounds ⁶
First	Returns the first element in the sequence – <i>or</i> – the first where the specified predicate function is <i>true</i> ⁸
FirstOrDefault	Returns the first element in the sequence – <i>or</i> – the first where the specified predicate function is <i>true</i> ; otherwise it returns the default element if the sequence is empty – <i>or</i> – no element satisfies the condition in the predicate ⁶
Last	Returns the last element in the sequence – <i>or</i> – the last where the specified predicate function is <i>true</i> ⁸
LastOrDefault	Returns the last element in the sequence – <i>or</i> – the last where the specified predicate function is <i>true</i> ; otherwise it returns the default element if the sequence is empty – <i>or</i> – no element satisfies the condition in the predicate ⁶
Single	Returns the only element in the sequence – <i>or</i> – the only where the specified predicate function is <i>true</i> ^{8,9}
SingleOrDefault	Returns the only element in the sequence – <i>or</i> – the only where the specified predicate function is <i>true</i> ; otherwise it returns the default element if the sequence is empty – <i>or</i> – no element satisfies the condition in the predicate ^{6,9}

- 6) The default value for TSource is null if it is a nullable or reference type and zero if it is a value type
 7) Throws an exception if the index is outside the bounds of the sequence
 8) Throws an exception when the sequence is empty – *or* – no element satisfies the condition in the predicate
 9) Throws an exception when the sequence contains more than one element – *or* – more than one element satisfies the condition in the predicate

Equality Operators

OPERATOR	DESCRIPTION
SequenceEqual	Returns a value indicating whether the specified sequence is equal to the sequence

Generation Operators

OPERATOR	DESCRIPTION
Empty	Returns an empty IEnumerable<TResult> of the specified type
Range	Returns an IEnumerable<Int32> that contains a sequence of integers within a specified range
Repeat	Returns an IEnumerable<TResult> that contains a sequence of a repeated value

Grouping Operators

OPERATOR	DESCRIPTION
GroupBy	Returns the sequence grouped using the specified key and optionally element and result selector functions

Joining Operators

OPERATOR	DESCRIPTION
GroupJoin	Returns an IEnumerable<TResult> that contains the matching elements in the sequence and the specified sequence joined and grouped using the specified key and result selector functions
Join	Returns an IEnumerable<TResult> that contains the matching elements in the sequence and the specified sequence joined using the specified key and result selector functions

Ordering Operators

OPERATOR	DESCRIPTION
OrderBy	Returns an IOOrderedEnumerable<TSource> that contains the elements in the sequence sorted using the specified key selector function
OrderByDescending	Returns an IOOrderedEnumerable<TSource> that contains the elements in the sequence in descending order using the specified key selector function
ThenBy	Returns an IOOrderedEnumerable<TSource> that performs a subsequent ordering of the elements using the specified key selector function
ThenByDescending	Returns an IOOrderedEnumerable<TSource> that performs a subsequent ordering of the elements in descending order using the specified key selector function
Reverse	Returns an IEnumerable<TSource> that contains the sequence with the elements in reverse order

Partitioning Operators

OPERATOR	DESCRIPTION
Skip	Returns an IEnumerable<TSource> that contains the sequence bypassing the specified number of elements
SkipWhile	Returns an IEnumerable<TSource> that contains the sequence bypassing elements as long as the specified predicate function is <i>true</i>
Take	Returns an IEnumerable<TSource> that contains the specified number of contiguous elements in the sequence
TakeWhile	Returns an IEnumerable<TSource> that contains contiguous elements in the sequence as long as the specified predicate function is <i>true</i>

Quantifier Operators

OPERATOR	DESCRIPTION
All	Returns a value indicating whether the sequence contains all elements where specified predicate function is <i>true</i>
Any	Returns a value indicating whether the sequence contains any elements – <i>or</i> – any where the specified predicate function is <i>true</i>
Contains	Returns a value indicating whether the specified element is contained in the sequence

Restriction Operators

OPERATOR	DESCRIPTION
Where	Returns an IEnumerable<TSource> that contains the elements in the sequence where the specified predicate function is <i>true</i>

Selection Operators

OPERATOR	DESCRIPTION
Select	Returns an IEnumerable<TResult> that contains the elements in the sequence transformed by the specified selector function
SelectMany	Returns an IEnumerable<TResult> that contains the elements in the sequence transformed to a one-to-many projection by the specified selector function

Set Operators

OPERATOR	DESCRIPTION
Concat	Returns an IEnumerable<TSource> that contains the elements in the sequence and the specified sequence
Distinct	Returns an IEnumerable<TSource> that contains the distinct elements in the sequence
Except	Returns an IEnumerable<TSource> that contains the difference of the elements between the sequence and the specified sequence ¹⁰
Intersect	Returns an IEnumerable<TSource> that contains the intersection of the elements between the sequence and the specified sequence ¹⁰
Union	Returns an IEnumerable<TSource> that contains the union of the elements between the sequence and the specified sequence

10) Returns a sequence containing distinct elements