

Filters

- [What filter types are available?](#)
- [How do I make my filter invertible?](#)
- [How do I create a custom admin filter?](#)

What filter types are available?

As of 15/09/2026, there are eleven filter types available:

Type	Description	Invertible?
Dropdown	Allows users to select a single option from a dropdown menu.	Yes
Radio List	Allows users to select exactly one option from a list of radio buttons.	Yes
Checkbox List	Allows users to select one or more options from a list of checkboxes.	Yes
Min Max	Allows users to specify minimum and maximum values for a numeric range.	No
Min Max Price	Allows users to specify minimum and maximum values for a price range.	No
Date Range	Allows users to filter results by selecting a start date and end date.	No
Options Date Range	A Date Range Filter that also allows users to select a predefined date range option, such as Today , Last 7 days , or Last 30 days .	No
Searchable Select	Allows users to search for and select an option from a searchable list.	Yes
Searchable Select When Over Count	Functions as a Checkbox List Filter when there's fewer than a certain number of options, in which case it then functions as a Searchable Select Filter.	Yes
Searchable Select With Modal	A Searchable Select Filter that supports searching via modal.	Yes

Searchable Select With Product Picker	A Searchable Select Filter for products that supports searching via product picker.	Yes
---------------------------------------	---	-----

Dropdown Filter

```
class ProductStockLevelViewAdminFilter extends \Aero\Admin\Filters\DropdownAdminFilter
{
    protected function handleDropdown($selected, $query)
    {
        if ($selected === 'only-out-of-stock') {
            $query->where('stock_level', '<=', 0);
        } elseif ($selected === 'only-in-stock') {
            $query->where('stock_level', '>', 0);
        } elseif ($selected === 'only-negative') {
            $query->where('stock_level', '<', 0);
        }
    }

    protected function dropdowns(): array
    {
        return [
            [
                'name' => 'Show all',
                'value' => '',
            ],
            [
                'name' => 'Only show out of stock',
                'value' => 'only-out-of-stock',
            ],
            [
                'name' => 'Only show in stock',
                'value' => 'only-in-stock',
            ],
            [
                'name' => 'Only show negative stock',
                'value' => 'only-negative',
            ],
        ];
    }
}
```

```

    }

    public function baseName(): string
    {
        return 'View';
    }
}

```

Radio List Filter

```

class OrderFlagStatusAdminFilter extends \Aero\Admin\Filters\RadioListAdminFilter
{
    protected function handleRadioList($selected, $query)
    {
        return $query
            ->when($selected === 'not', function ($query) {
                $query->whereDoesntHave('flags');
            })
            ->when($selected === 'has', function ($query) {
                $query->whereHas('flags');
            });
    }

    protected function radios(): array
    {
        return [
            [
                'value' => 'has',
                'name' => 'Has been flagged',
            ],
            [
                'value' => 'not',
                'name' => 'Has not been flagged',
            ],
        ];
    }

    public function baseName(): string

```

```
{  
    return 'FlagStatus';  
}  
}
```

Checkbox List Filter

```
class RedirectsStatusAdminFilter extends \Aero\Admin\Filters\CheckboxListAdminFilter  
{  
    protected function handleCheckboxList($selected, $query)  
    {  
        $query->whereIn('status', $selected);  
    }  
  
    protected function checkboxes(): array  
    {  
        return [  
            [  
                'id' => '301',  
                'name' => 'Moved Permanently (301)',  
                'url' => $this->getUrlFor('301'),  
            ],  
            [  
                'id' => '302',  
                'name' => 'Found Redirect (302)',  
                'url' => $this->getUrlFor('302'),  
            ],  
            [  
                'id' => '308',  
                'name' => 'Permanent Redirect (308)',  
                'url' => $this->getUrlFor('308'),  
            ],  
        ];  
    }  
  
    protected function baseName(): string  
    {  
        return 'Status';  
    }  
}
```

```
}  
}
```

`getUrlFor` is a helper method provided by the `Aero\Admin\Filters\Traits\AdminFilterHelpers` trait.

Min Max Filter

```
class ProductStockLevelStockAdminFilter extends \Aero\Admin\Filters\MinMaxAdminFilter  
{  
    public function handleMinMax($min, $max, $query)  
    {  
        $query  
            ->when($min !== null, function ($query) use ($min) {  
                $query->where('stock_level', '>=', $min);  
            })  
            ->when($max !== null, function ($query) use ($max) {  
                $query->where('stock_level', '<=', $max);  
            });  
    }  
  
    public function baseName(): string  
    {  
        return 'Stock';  
    }  
}
```

Min Max Price Filter

```
class VoucherBalanceAdminFilter extends \Aero\Admin\Filters\MinMaxPriceAdminFilter  
{  
    public function handleMinMax($min, $max, $query)  
    {  
        return $query  
            ->when($min !== null, function ($query) use ($min) {  
                $query->where('balance', '>=', $min);  
            });  
    }  
}
```

```

    })
    ->when($max !== null, function ($query) use ($max) {
        $query->where('balance', '<=', $max);
    });
}

public function baseName(): string
{
    return 'Balance';
}
}

```

Date Range Filter

```

class CustomerLastLoginAtAdminFilter extends \Aero\Admin\Filters\DateRangeAdminFilter
{
    protected function handleDateRange($startDate, $endDate, $query)
    {
        $query->whereBetween('last_login_at', [$startDate, $endDate]);
    }

    protected function baseName(): string
    {
        return 'LastLoginAtDate';
    }
}

```

Options Date Range Filter

```

class ExportLogCreatedAtDateAdminFilter extends
\Aero\Admin\Filters\OptionsDateRangeAdminFilter
{
    protected $extraDates = true;

    protected function handleDateRange($startDate, $endDate, $query): void
    {

```

```

        $query->whereBetween('created_at', [$startDate, $endDate]);
    }

    protected function baseName(): string
    {
        return 'CreatedAtDate';
    }
}

```

Searchable Select Filter

```

class OrderDiscountAdminFilter extends \Aero\Admin\Filters\SearchableSelectAdminFilter
{
    protected $multiple = true;

    protected function handleSearchableSelect(array $selected, $query): void
    {
        $query->whereHas('discounts', function ($query) use ($selected) {
            $query->whereIn('id', $selected);
        });
    }

    protected function model(): \Aero\Common\Models\Model
    {
        return new \Aero\Cart\Models\Discount();
    }

    protected function modelName(\Aero\Common\Models\Model $model)
    {
        return $model->code ?? ($model->name ?? 'Discount #'.$model->id);
    }

    protected function searchRoute(): string
    {
        return route('admin.discounts.search');
    }

    protected function baseName(): string

```

```
{  
    return 'Discount';  
}  
}
```

Searchable Select When Over Count Filter

```
class CurrencyAdminFilter extends \Aero\Admin\Filters\SearchableSelectWhenOverCountAdminFilter  
{  
    protected function handleSearchableSelectWhenOverCount(array $selected, $query)  
    {  
        $query->whereHas('currency', function ($query) use ($selected) {  
            $query->whereIn('code', $selected);  
        });  
    }  
  
    protected function model(): \Aero\Common\Models\Model  
    {  
        return new \Aero\Common\Models\Currency();  
    }  
  
    protected function modelName(\Aero\Common\Models\Model $model): string  
    {  
        return $model->name . ' (' . $model->code . ')';  
    }  
  
    protected function searchRoute(): string  
    {  
        return route('admin.configuration.currencies.search');  
    }  
  
    protected function whenOverCount(): int  
    {  
        return 10;  
    }  
  
    protected function baseName(): string  
    {
```

```
        return 'Currency';
    }
}
```

Searchable Select With Modal Filter

```
class PriceListCategoryAdminFilter extends
    \Aero\Admin\Filters\SearchableSelectWithModalAdminFilter
{
    protected $multiple = true;

    protected function handleSearchableSelect(array $selected, $query)
    {
        $query->whereHas('entries', function ($query) use ($selected) {
            $query->whereHasMorph('catalog', [\Aero\Catalog\Models\Category::class], function
($query) use ($selected) {
                $query->whereIn('id', $selected);
            });
        });
    }

    protected function model(): \Aero\Common\Models\Model
    {
        return new \Aero\Catalog\Models\Category();
    }

    protected function modelName(\Aero\Common\Models\Model $model)
    {
        return $model->breadcrumb->map->name->implode(' » ');
    }

    protected function searchRoute(): string
    {
        return route('admin.catalog.categories.search');
    }

    protected function modalRoute(): string
    {

```

```
        return route('admin.catalog.categories.search.tree');
    }

    public function baseName(): string
    {
        return 'Category';
    }
}
```

Searchable Select With Product Picker Filter

```
class PriceListProductAdminFilter extends
    \Aero\Admin\Filters\SearchableSelectWithProductPickerAdminFilter
{
    protected function handleSearchableSelect(array $selected, $query)
    {
        $query->whereHas('entries', function ($query) use ($selected) {
            $query->whereHasMorph('catalog', [\Aero\Catalog\Models\Product::class], function
($query) use ($selected) {
                $query->whereIn('id', $selected);
            });
        });
    }

    public function baseName(): string
    {
        return 'Product';
    }
}
```

How do I make my filter invertible?

As of `15/09/2026`, the following filters support being inverted:

Type	Handle Method	Inverted Handle Method
Dropdown	<code>handleDropdown</code>	<code>handleInvertedDropdown</code>
Radio List	<code>handleRadioList</code>	<code>handleInvertedRadioList</code>
Checkbox List	<code>handleCheckboxList</code>	<code>handleInvertedCheckboxList</code>
Searchable Select	<code>handleSearchableSelect</code>	<code>handleInvertedSearchableSelect</code>
Searchable Select When Over Count	<code>handleSearchableSelectWhenOverCount</code>	<code>handleInvertedSearchableSelectWhenOverCount</code>
Searchable Select With Modal	<code>handleSearchableSelect</code>	<code>handleInvertedSearchableSelect</code>
Searchable Select With Product Picker	<code>handleSearchableSelect</code>	<code>handleInvertedSearchableSelect</code>

To enable inverted handling, set `$invertible` to `true` on the filter:

```
class RedirectsStatusAdminFilter extends \Aero\Admin\Filters\CheckboxListAdminFilter
{
    protected $invertible = true;

    protected function handleCheckboxList($selected, $query)
    {
        $query->whereIn('status', $selected);
    }

    protected function handleInvertedCheckboxList(array $selected, $query): void
    {
        $query->whereNotIn('status', $selected);
    }

    protected function checkboxes(): array
    {
        return [
```

```

        [
            'id' => '301',
            'name' => 'Moved Permanently (301)',
            'url' => $this->getUrlFor('301'),
        ],
        [
            'id' => '302',
            'name' => 'Found Redirect (302)',
            'url' => $this->getUrlFor('302'),
        ],
        [
            'id' => '308',
            'name' => 'Permanent Redirect (308)',
            'url' => $this->getUrlFor('308'),
        ],
    ];
}

protected function baseName(): string
{
    return 'Status';
}
}

```

The inverted handler is optional for Laravel 10+ (in most cases), as it is automatically generated using `whereNot`.

How do I create a custom admin filter?

To create a custom admin filter you must create a class that extends `\Aero\Admin\Filters\AdminFilter` and implement a `handle` method:

```
class OrderStatusAdminFilter extends \Aero\Admin\Filters\AdminFilter
{
    use \Aero\Admin\Filters\Traits\HasDefault;
    use \Aero\Admin\Filters\Traits\AdminFilterHelpers;

    protected $view = 'admin::filters.order-status';

    protected function selectedOptions(): array
    {
        return request()->input($this->key(), []) ?? [];
    }

    public function handle(\Symfony\Component\HttpFoundation\ParameterBag $parameters, $query)
    {
        $view = $parameters->get('view');

        if ($view === null && strlen($parameters->get('q', ''))) {
            $view = 'all';

            request()->request->set('view', $view);
        } elseif ($view === null) {
            $view = static::getDefault();
        }

        $selectedStatusIds = $view === 'incomplete'
            ? collect()
            : collect($parameters->get($this->key()))
                ->map(function ($value) {
                    return intval($value);
                });
    }
}
```

```

        })
        ->filter()
        ->values();

$query->when($view === 'incomplete', function ($query) {
    $query->whereNull('ordered_at');
})->when($view === 'outstanding' || $view === 'new', function ($query) use
($selectedStatusIds) {
    $outstandingStatusIds =
\Aero\Cart\Models\OrderStatus::forState(...setting('admin.outstanding_order_states'))-
>pluck('id');

    $statusIds = $selectedStatusIds->isNotEmpty()
        ? $outstandingStatusIds->intersect($selectedStatusIds)
        : $outstandingStatusIds;

    if ($statusIds->isNotEmpty()) {
        $query->whereIn('order_status_id', $statusIds);
    }
})->when($view === 'all', function ($query) use ($selectedStatusIds) {
    $query->whereNotNull('ordered_at');

    if ($selectedStatusIds->isNotEmpty()) {
        $query->whereIn('order_status_id', $selectedStatusIds);
    }
});
}

protected function viewData(): array
{
    $selectedStatuses = collect(request()->input($this->key()));
    $selectedView = request()->input('view', static::getDefault());

    $orderStatuses = \Aero\Cart\Models\OrderStatus::ordered()->get()
        ->map(function ($status) use ($selectedStatuses) {
            return [
                'id' => $status->id,
                'name' => $status->name,
                'state' => $status->state,
                'url' => $this->getUrlFor($status->id),
            ];
        });
}

```

```

        ];
    });

    return [
        'selectedStatuses' => $selectedStatuses,
        'statuses' => collect($orderStatuses)
            ->filter(function ($status) use ($selectedView) {
                switch ($selectedView) {
                    case 'all': return $status['state'] !== 'incomplete';
                    case 'new':
                    case 'outstanding': return in_array($status['state'],
setting('admin.outstanding_order_states')->toArray());
                }

                return false;
            })
            ->toArray(),
    ];
}

protected function baseName(): string
{
    return 'Status';
}

public function stateFields(): array
{
    return [$this->key(), 'view'];
}

protected static function defaultValue(): string
{
    return 'outstanding';
}
}

```

And then create a view for rendering the filter, in this case `admin::filters.order-status`:

```

<div class="border-b border-grey-light" data-filter-id="status">
    <button class="flex justify-between items-center w-full p-4" type="button"

```

```

onClick="toggleContent(this)">
    <span class="text-sm font-medium">{{ $title }}</span>
    <span class="[ js-accordion-arrow ] w-4 h-4 flex items-center justify-center transform
rotate-90">@include('admin::icons.caret-down')</span>
</button>

<div class="p-4 pt-2 list-reset [ js-accordion-content ] hidden">
    <div class="select w-full -mt-1">
        <select id="view_all" name="view" class="w-full" autocomplete="off"
onChange="updateUrlQuery('view', this.value)">
            <option value="outstanding" {{ request('view',
\\Aero\\Admin\\Filters\\Order\\OrderStatusAdminFilter::getDefault()) === 'outstanding' ||
request('view', \\Aero\\Admin\\Filters\\Order\\OrderStatusAdminFilter::getDefault()) === 'new' ?
'selected' : '' }} data-class="new">Outstanding Orders</option>
            <option value="all" data-class="all"
                @if(request('view',
\\Aero\\Admin\\Filters\\Order\\OrderStatusAdminFilter::getDefault()) == 'all') selected data-
filter-active="true" data-filter-group-name="View" data-filter-name="All Orders" data-filter-
key="view" @endif>All Orders</option>
            <option value="incomplete" data-class="incomplete"
                @if(request('view',
\\Aero\\Admin\\Filters\\Order\\OrderStatusAdminFilter::getDefault()) == 'incomplete') selected
data-filter-active="true" data-filter-group-name="View" data-filter-name="Incomplete Orders"
data-filter-key="view" @endif>Incomplete Orders</option>
        </select>
    </div>

    @if(count($statuses) > 0)
        <div class="pt-4 flex flex-col bg-white mt-2 overflow-hidden">
            <ul class="list-reset overflow-auto leading-loose" style="max-height: 216px">
                @foreach($statuses as $status)
                    <li class="mb-2 filter">
                        <label for="status-{{ $status['id'] }}"><label class="checkbox">
                            <input id="status-{{ $status['id'] }}" type="checkbox"
name="{{ $key }}" autocomplete="off" value="{{ $status['id'] }}" onclick="location.href =
'{{ $status['url'] }}? ' }}" ;"
                                @if($selectedStatuses->contains($status['id']))
checked data-filter-active="true" data-filter-group-name="{{ $title }}" data-filter-name="{{
$status['name'] }}" data-filter-url="{{ $status['url'] }}" @endif
                            > <span></span></label>

```

```
                {{ $status['name'] }}
            </label>
        </li>
    @endforeach
</ul>
</div>
@endif
</div>
</div>
```

Which receives the `viewData` defined by your filter type as well as the `options` defined by the base class (`$title` and `$key`).