// Text Search demo script (v2190912)
// This script is to be executed against your 'application' store and will install the Graphileon Function nodes required for the demo applications.
//
// The script searches for your default dashboard and connects to it with a SHORTCUT relation.
// If you default dashboard has another name than "Default" then update the first line of this script.
//
// This application also requires that you create an index on your data store. Depending on the contents of your datastore this should be similar to this:
// CALL db.index.fulltext.createNodeIndex("my_index",["Movie", "Person"],["title", "tagline", "name"])
//
// WARNING
// The Cypher statement below does not take into account any existing data in database you are running it against.
// It may cause conflicts (e.g. with existing constraints), or produce other undesired side effects (e.g. change results of existing queries).
// It is therefore highly recommended to create a backup of your graph database before executing the statement.
//
// = start script =
//

```
MATCH (dashb:IA_Dashboard{name:'Default'})
WITH dashb
CREATE (_62:`IA_Function`:`IA_HtmlView`:`SEARCHDEMO` {
    `template`: '<div class="flexbox flex flex-column full-height">

    <div class="v-pad-2 h-pad-0">
        <form>
            <div>
                <label>Search <i class="fa fa-question-circle trigger-click"
data-action="help"></i> </label>
                <input type="search" name="search" class="form-input" style="width:
100%">
            </div>
            <div class="v-pad-2 h-pad-0">
                <button>Search</button>
            </div>
        </form>
    </div>
    <div class="flexbox flex flex-column" id="prologram-area-item-list-container" >
    </div>
</div>
',
    `$container`: '{
      width: 1200,
      height: 800
}',
    `name`: 'Search form',
    `$css`: '.full-height {
    height: 100%;
}

.h-pad-0 {
    padding-left: 0px;
    padding-right: 0px;
}
```

```
.v-pad-2 {
    padding-top: 4px;
    padding-bottom: 4px;
}
',
    `$container:evaluate`: 'full',
    `$_instance`: 'search_list'
})
CREATE (_63:`IA_Function`:`IA_HtmlView`:`SEARCHDEMO` {
    `template`: '
    Fields used in searching: "title", "tagline", "name".

<h4>Term search</h4>
    <code>good</code> - all records that contain the term "good"

<h4>Multiple terms</h4>
    <code>few man</code> - all records that contain one or both terms

<h4>Phrase search</h4>
    <code>"good as it"</code> - all records that contain the terms in sequence or in
near vicinity

<h4>Boolean</h4>
    <code>good AND men</code> - all records that contain both words

<h4>Wildcard</h4>
    <code>m?n</code> - ? is wildcard for one character (matches words like "man",
"men") <br/>
    <code>*man</code> - * is wildcard for a group of 0 or more characters (matches
words like \'woman\', \'man\' ....)

<h4>Approximate (fuzzy)</h4>
    <code>tim~</code> - search for similar words (ex: tim, time, tom)

<h4>Field search</h4>
    <code>title: matr*</code> - all records that have the field "title" starting
with "matr"
<br/><br/>
<p>
Any combination of these types of searches is allowed.
</p>',
    `name`: 'Help text',
    `$container.width`: '600',
    `css`: 'h4 {
    margin: 8px 0px 2px;
}',
    `$container.title`: 'How to search'
})
CREATE (_64:`IA_Function`:`IA_CypherQuery`:`SEARCHDEMO` {
    `name`: 'Search for Items',
    `cypher`: 'CALL db.index.fulltext.queryNodes("my_index", toString($search))
YIELD node, score
WITH node, score
WITH node{id: id(node),
          .*,
```

```
          labels: REDUCE(a=\'\',l IN labels(node) | a
                                                    + CASE WHEN a = \'\' THEN \'\'
ELSE \',\' END
                                                     + l
                            ),
            label:COALESCE(node.name,node.title),
            year:COALESCE(node.released,node.born),
            score: score} AS item
ORDER BY item.score DESC
RETURN COLLECT(item) AS items'
})
CREATE (_67:`IA_Function`:`SEARCHDEMO`:`IA_AgGridView` {
    `$options.rowSelection`: 'multiple',
    `$options.suppressContextMenu`: 'true',
    `$options.suppressMenuColumnPanel`: 'true',
    `$options.enableSorting`: 'true',
    `$container.state`: 'maximized',
    `$options.columnDefs`: '{
    id: {
        field: \'id\',
        headerName: \'id\',
        hide:true
    },

    labels: {
        field: \'labels\',
        headerName: \'Labels\',
              checkboxSelection: true,
              headerCheckboxSelectionFilteredOnly:true,
              headerCheckboxSelection: true,
    },

    label: {
        field: \'label\',
            headerName: "Name/Title",
        width: 200
    },

    year: {
          field: \'year\',
        headerName: "Year",
        width: 200
    },

    tagline: {
        field: \'tagline\',
            headerName: "Tagline",
        width: 300
    },

    \'score\': {
        field:\'score\',
            headerName: "Matching score"
    }
}',
    `$options.suppressMenuMainPanel`: 'true',
```

```
    `$options.floatingFilter`: 'false',
    `$container.title`: 'Items',
    `$options.suppressRowClickSelection`: 'true',
    `$area`: 'item-list-container',
    `name`: 'Item list',
    `$options.pagination`: 'false',
    `$options.paginationAutoPageSize`: 'false',
    `$options.columnDefs:evaluate`: 'full',
    `$container.height`: '10000',
    `$container.id`: 'item-list'
})
CREATE (_66:`IA_Function`:`IA_CypherQuery`:`SEARCHDEMO` {
    `name`: 'Get elements',
    `process`: 'true',
    `cypher`: 'MATCH (n)
WHERE id(n) IN $selIds
RETURN n'
})
CREATE (_65:`IA_Function`:`IA_NetworkView`:`SEARCHDEMO` {
    `$nodeId`: '',
    `$canDownloadSVG`: '1',
    `$canSwitchAutoCompleteStatus`: '1',
    `$container.width`: '800',
    `$addOnExecute`: '',
    `$autoCompleteStatus`: 'true',
    `_instance`: 'nvsearch',
    `$diagramName`: 'null',
    `$container.title`: 'Nodes from Search',
    `$canSwitchZoomToFitStatus`: '1',
    `$area`: 'content',
    `name`: 'Search NetworkView',
    `$canSwitchAutoLayoutStatus`: '1',
    `$zoomToFitStatus`: '1',
    `$container.height`: '800',
    `exploreFilter`: 'false',
    `$canSetForceParameters`: '1',
    `$container.id`: '_search'
})
CREATE (_62)-[:`TRIGGER` {
    `type`: 'click',
    `data.action`: 'help'
}]->(_63)
CREATE (_62)-[:`TRIGGER` {
    `type`: 'submit',
    `$params.search`: '(%).data.search',
    `data.search.trim().length > 0`: 'true'
}]->(_64)
CREATE (dashb)-[:`SHORTCUT` {
    `name`: 'Search demo',
    `icon`: 'search',
    `style`: 'background-color: #309cff;',
    `hoverStyle`: 'background-color: #6fb9fd;'
}]->(_62)
CREATE (_64)-[:`TRIGGER` {
    `type`: 'success',
    `#options.rowData`: '(%).data[0].items'
```

```
}]->(_67)
CREATE (_67)-[:`TRIGGER` {
    `icon`: 'share-alt',
    `action`: 'Add selection',
    `index`: '99',
    `#params.selIds`: '(%)._function.state.selected[#].id',
    `type`: 'batch'
}]->(_66)
CREATE (_66)-[:`TRIGGER` {
    `type`: 'success',
    `! (@).instances.nvsearch`: 'true',
    `#nodes`: '(%).processed.nodes'
}]->(_65)
CREATE (_66)-[:`TRIGGER` {
    `#_update.add.nodes`: '(%).processed.nodes',
    `type`: 'success',
    `!! (@).instances.nvsearch`: 'true'
}]->(_65)

// return the demo configuration
WITH dashb
MATCH (n:SEARCHDEMO) return dashb, n

// = end script =
```