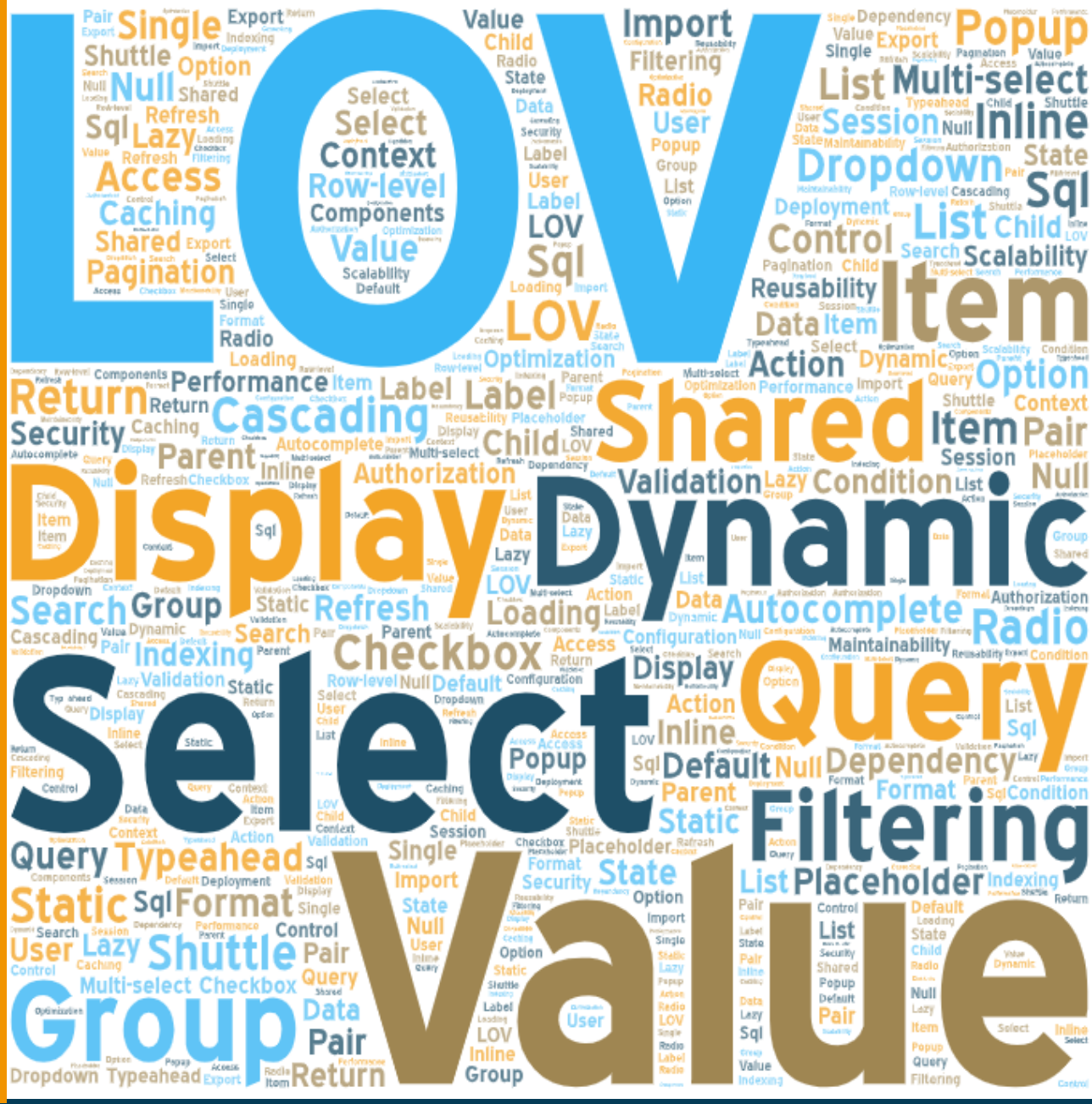
Gain full control over LOVs!

Bart Limbeek



Who am I?

- Bart Limbeek
- Transfer Solutions
- 9+ years APEX experience
- Love to watch:
  - Speedskating
  - Olympic Games

What's the presentation's theme?

- Practical LOV challenges
- From Front-end to Backend
- Design alternatives
- Context
  - Olympic Games
  - Speed skating

What's on the agenda?

- UX/UI issues in Popup LOVs
- Managing LOVs
- Data integrity issues
- Alternatives for LOV items
  - Representation of data
  - Sorting your results

UX/UI issues in Popup LOVs

How to give a good user experience and user interface?

UX/UI issues in Popup LOVs

Standard situation

- **Equal column widths by default**
- **Columns can be resized manually**
- **No text wrapping within cells**
- **Columns can be sorted**
  - Sorting cannot be reset to default

UX/UI issues in Popup LOVs

Set widths of columns

- **LOV item → Advanced → Initialization JavaScript Function**
- **Column names are defined in the Shared Component**
- **Columns without defined widths are evenly distributed**

```
function(options) {

    options.minWidth = 800;

    options.columns.YEAR.width = 60;
    options.columns.COUNTRY_NAME.width = 100;
    options.columns.CITY.width = 300;
    options.columns.OPENING_CEREMONY.width = 50;

    return options;
}
```

UX/UI issues in Popup LOVs

Avoid irreversible options

- **Column resizing is remembered in local storage**
- **Resize columns by default is enabled**
- **Apply to every popup LOV**
- **Manual sorting can not be reset**

```
function(options) {

    options.defaultGridOptions = {
        resizeColumns: false
    ,   columnSort: false
    }

    return options;
}
```

UX/UI issues in Popup LOVs

Multi line cells in LOV

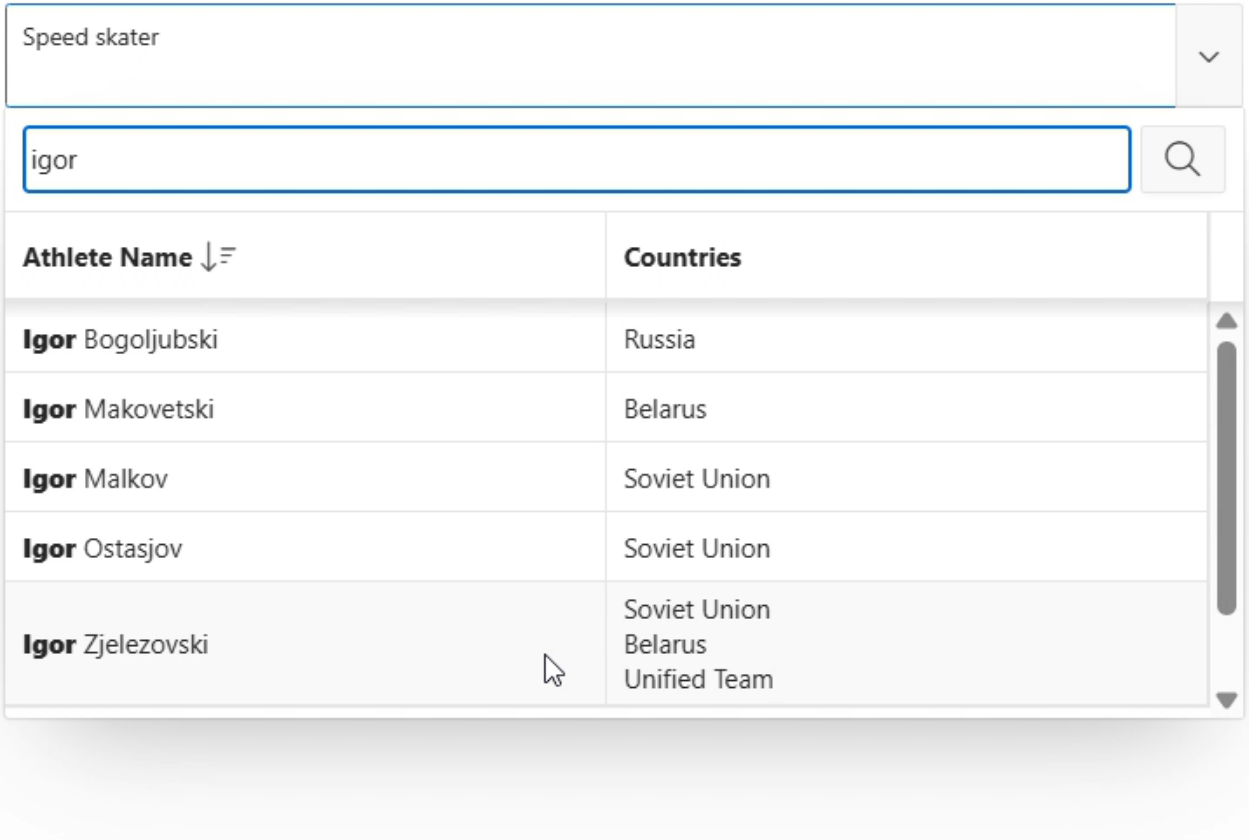
- **Useful for grouped data**
- **Avoid possible column space**
- **No
 “Breaks” are possible**
- **Chr(10) → for line breaks**
- **CSS to make it visible**
  - white-space: pre-line

```
select  athlete_id
    ,   athlete_name
    ,   listagg(
            country_name
        ,   chr(10)
        ) as countries
from    athletes
join    athlete_countries using (athlete_id)
join    countries using (ioc_code)
group by athlete_id
    ,   athlete_name
```

```
.a-PopupLOV-results  .a-GV-cell{
    white-space: pre-line;
}
```

UX/UI issues in Popup LOVs

Multi line cells in LOV



Managing LOVs

What is a good way to use parameterized LOVs?

Managing LOVs

Best Practice 1:

- **Define your LOVs centrally in Shared Components**
  - Single source of truth
  - Reusable across pages
  - Easier maintenance & consistency
  - Reduces duplication

Best Practice 2:

- **Put logic in the database (use views)**
  - Centralized business logic
  - Better performance & scalability
  - Simplifies LOVs in APEX
  - Easier to maintain and secure

But: Not possible to parameterize!

Managing LOVs

Situation 1: Domain table

- **Multiple LOVs on the same table**
- **Same return and display value**
- **Domain name could be a parameter**
- **Show active rows only**

```
select  value as display_value
    ,   code as return_value
from    domains
where   domain_name = 'GENDER'

select  value as display_value
    ,   code as return_value
from    domains
where   domain_name = 'OLYMPIC_SPORTS'

select  value as display_value
    ,   code as return_value
from    domains
where   domain_name = 'OLYMPIC_SPORTS'
and     (
            end_date is null
        or
            end_date < sysdate
        )
```

Managing LOVs

Situation 2: Cascading LOVs

- Multiple LOVs on the same table
- Comparable return and display value
- Items could be parameters
- Even in select statements possible

```
select  athlete_name as display_value
    ,   athlete_id as return_value
from    athletes
where   gender = :P2_GENDER
or      :P2_GENDER is null

select  athlete_name as display_value
    ,   athlete_id as return_value
from    athletes
where   gender = :P3_GENDER
or      :P3_GENDER is null

select  athlete_name || ' ' || '(' ||
        country_name ||
        ')' as display_value
    ,   athlete_id as return_value
from    athletes
join    athlete_countries using (athlete_id)
join    countries using (ioc_code)
where   ioc_code = :P2_IOC_CODE
```

Managing LOVs

SQL table macro

- Return a SQL query (not data) as a reusable function
- Expands at parse time
- Acts like a view with parameters
- Called in a SQL query from clause

```
function domain_values(
    p_domain_name in varchar2
,   p_active_only in boolean
) return varchar2 sql_macro
is
begin
    return q'{
        select  value as display_value
            ,   code as return_value
        from    domains
        where   domain_name = p_domain_name
        and     (
                    not p_active_only
                or
                    nvl(end_date,sysdate) >= sysdate
                )
    }';
end domain_values
```

Managing LOVs

Situation 1: Domain table

```
select  display_value
    ,   return_value
from    domain_values(
            p_domain_name => 'GENDER'
        ,   p_active_only => false
        );

select  *
from    domain_values(
            p_domain_name => 'OLYMPIC_SPORTS'
        ,   p_active_only => false
        );

select  *
from    domain_values(
            p_domain_name => 'OLYMPIC_SPORTS'
        ,   p_active_only => true
        );
```

```
function domain_values(
    p_domain_name in varchar2
,   p_active_only in boolean
) return varchar2 sql_macro
is
begin
    return q'{
        select  value as display_value
            ,   code as return_value
        from    domains
        where   domain_name = p_domain_name
        and     (
                    not p_active_only
                or
                    nvl(end_date,sysdate) >= sysdate
                )
    }';
end domain_values
```

Managing LOVs

Situation 2: Cascading LOVs

```
select  *
from    athletes(
            p_gender => :P2_GENDER
        );

select  *
from    athletes(
            p_gender => :P3_GENDER
        );

select  *
from    athletes(
            p_gender => null
        ,   p_ioc_code => :P2_IOC_CODE
        ,   p_show_country => true
        );
```

```
function athletes(
    p_gender in varchar2
,   p_ioc_code in varchar2 := null
,   p_show_country in boolean := false
) is
begin
    return q'{
        select  athlete_name ||
                case when p_show_country then
                    ' (' || country_name || ')'
                end as display_value
            ,   athlete_id as return_value
        from    athletes
        join    countries using (ioc_code)
        where   (
                    countries.ioc_code = p_ioc_code
                or  p_ioc_code is null
                )
        and     (
                    athletes.gender = p_gender
                or  p_gender is null
                )
    }';
end athletes;
```

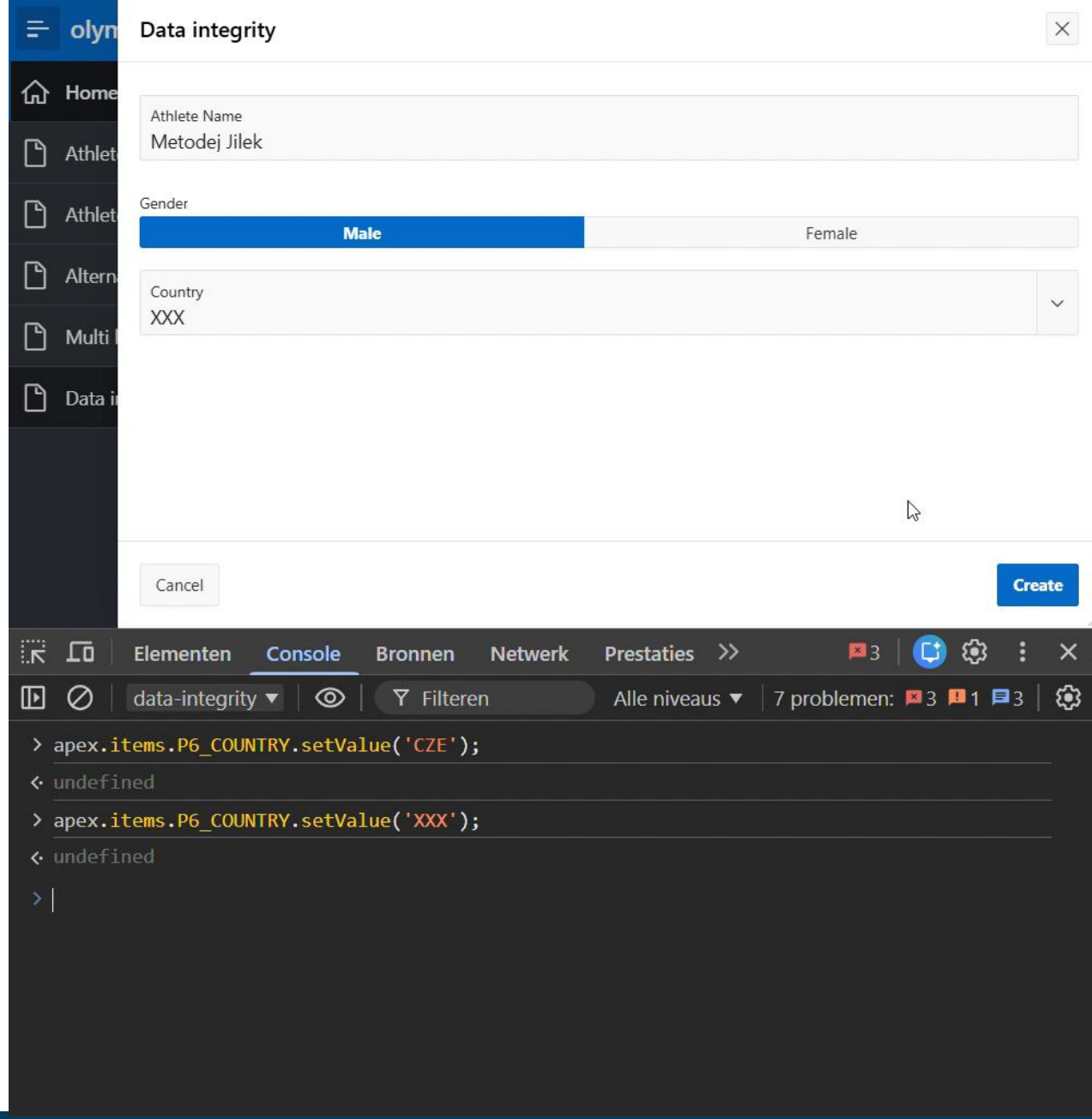
Data integrity

How to secure your date when using LOVs?

Data integrity

How is it insecure?

- LOV showed valid results
- Set value by Javascript
- Check constraint prevents non-existent values
- Non-constraint checked values are allowed



Data integrity

When is it a problem?

- **Before APEX 24.2**
- **Dynamic items**
  - Popup LOV
  - Select One
  - Select Many
  - List manager
- **The lack of a database constraint**
- **Cascading/Parameterized LOVs**

When is it safe?

- **APEX 24.2**
- **Safe items**
  - Select One
  - Combo box
  - Radio group
  - Checkboxes
- **When all values are shown and a database constraint**
- **When data is validated**

Data integrity

How to solve?

- **Upgrade to APEX 24.2**
- **Validate your value at submit**

```
declare
    cursor c_country(
        p_ioc_code in varchar2
    )
    is
        select  ioc_code
        from    countries
        where   ioc_code = p_ioc_code
        and     end_date is null;
begin
    open c_country(p_ioc_code => :P6_COUNTRIES);
    fetch c_country into r_country;

    if c_country%notfound then
        apex.error.add_error(
            p_message => 'It is not a valid country'
        ,   p_display_location =>
                apex_error.c_inline_in_notification
        );
    end if;

    close c_countries;
end;
```

Alternative for LOV item

Can you get more developer freedom with an alternative solution?

Alternative for LOV item

Pros of LOV items (requirements)

- **Direct searching with typing**
- **Shown as an inline popup**
- **Search over multiple columns**
- **Suitable for large datasets**

Cons of LOV Items (restrictions)

- **Limited UI flexibility**
- **(Almost) no options for images/html**
- **Limited sorting options**
- **Only 1 search filter**

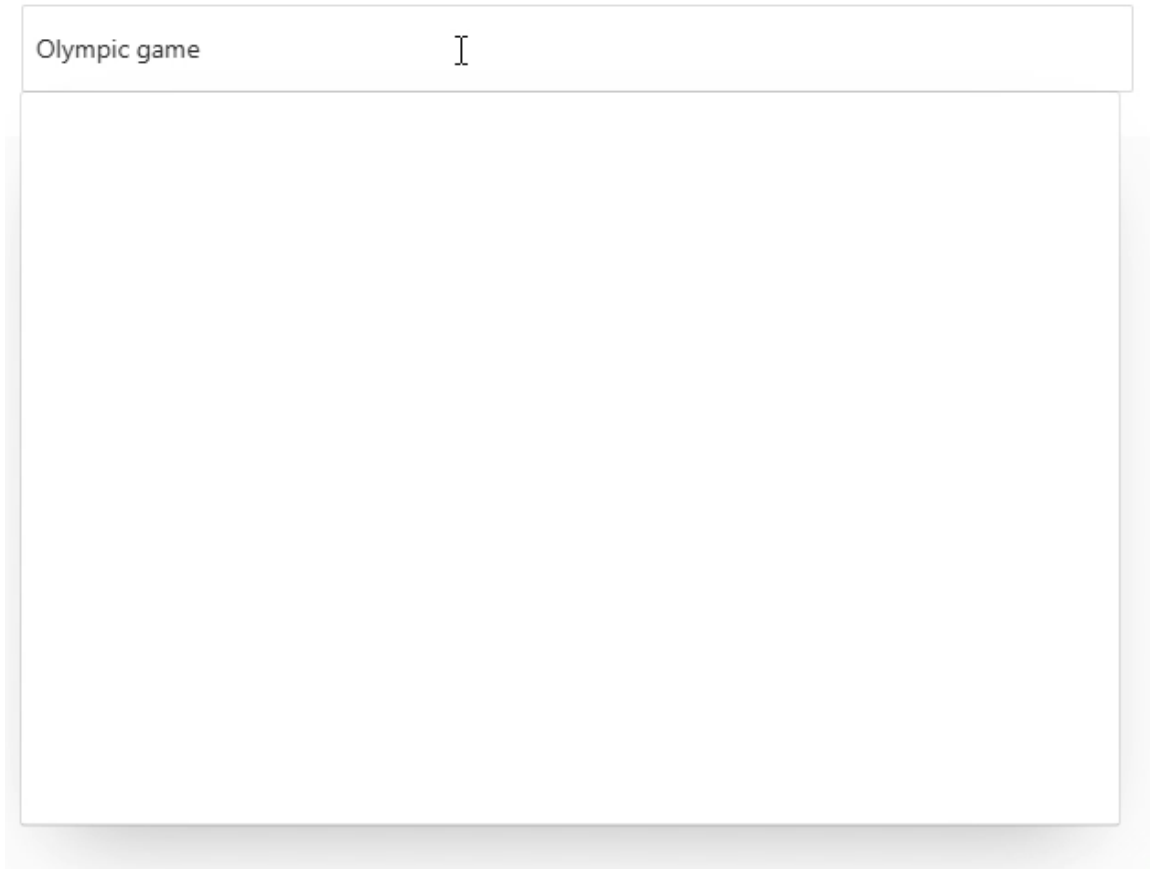
Alternative for LOV item – Basic solution

Step 1: Open an inline popup with an item

- **Items**
  - Text field (for search/display value)
  - Hidden (for return value)
- **Region – Static content**
  - Slot: Dialogs, Drawers and Popups
  - Template: Inline Popup
  - Custom Attributes: data-parent-element="#P1_ITEM"
- **Dynamic action – Key down**
  - Event: Key Down
  - Selection Type: Item(s)
  - True action: Open Region

Alternative for LOV item – Basic solution

Step 1: Open an inline popup with an item



Tip about anchored inline popups: https://hartenfeller.dev/blog/apex-anchored-inline-popups

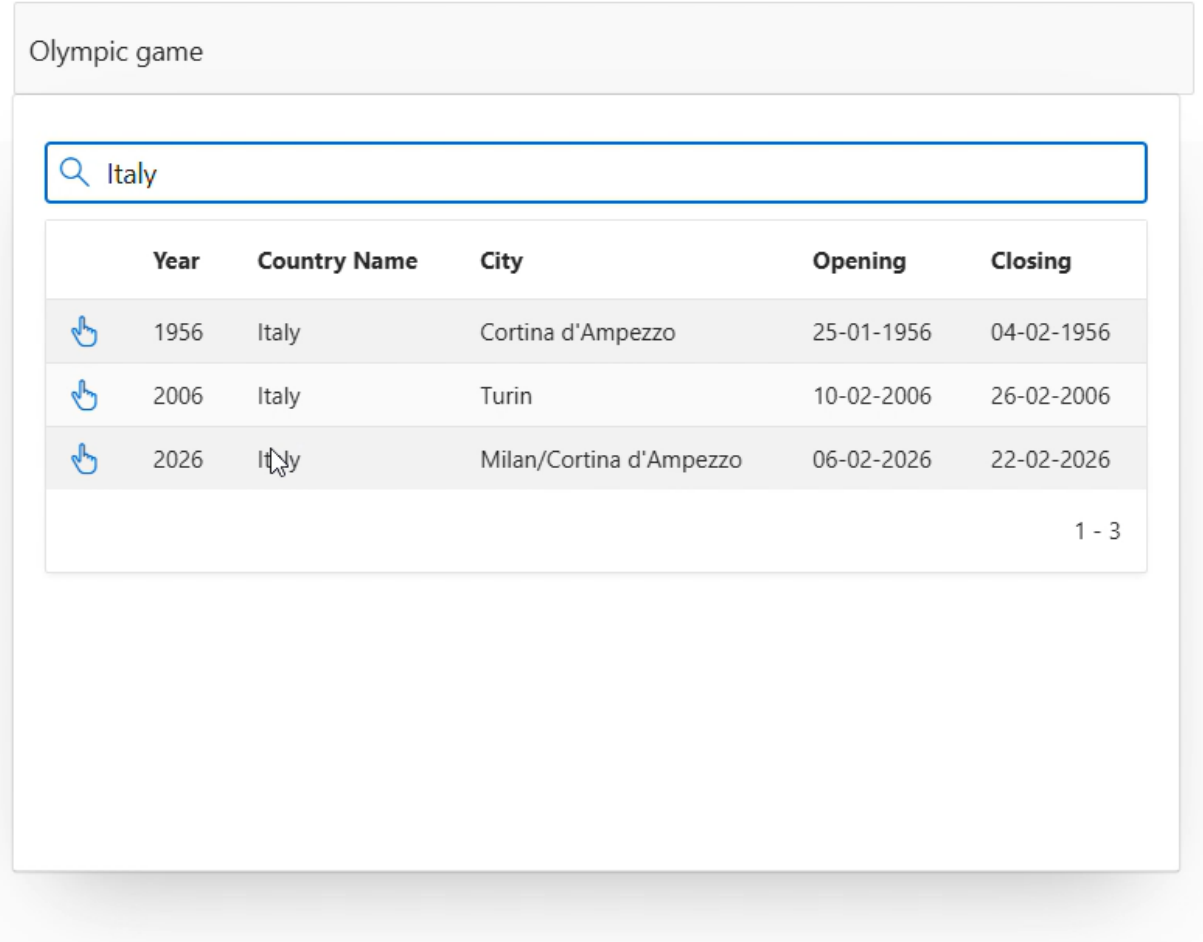
Alternative for LOV item – Basic solution

Step 2: Search in the results and select the value

- **Subregions within Inline Popup region**
  - Classic Report
  - Faceted Search
- **Facets**
  - Search
  - External Page Items
- **Link column**
  - Set values for return and display item

Alternative for LOV item – Basic solution

Step 2: Search in the results and select the value



Alternative for LOV item – Basic solution

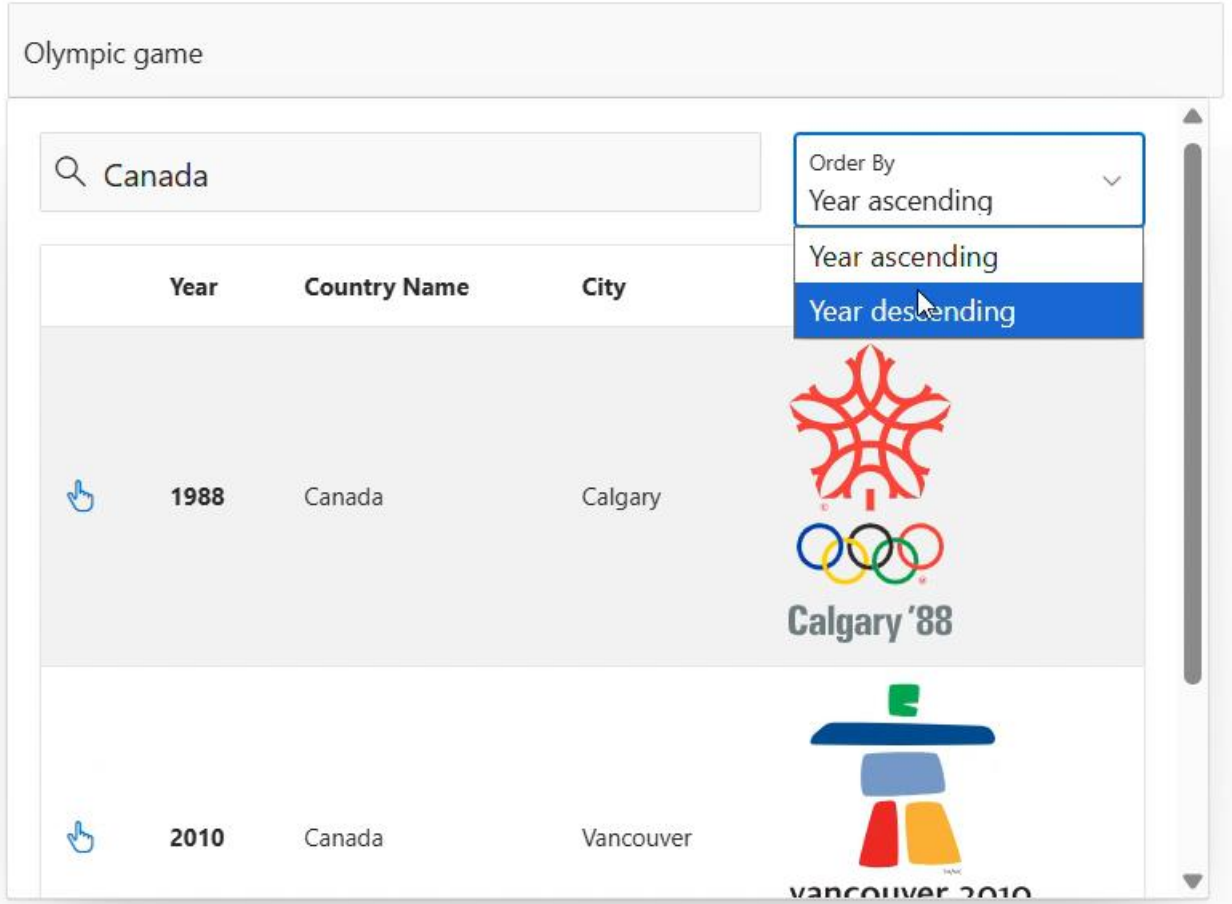
Step 3: Final touches (UX/UI)

- **Adjusted width and height settings**
- **Button to open popup by click**
- **Exclude tab-key to open popup**
- **Clear the Search value when searching again**
- **Entire row in search results has to be clickable**

Alternative for LOV item

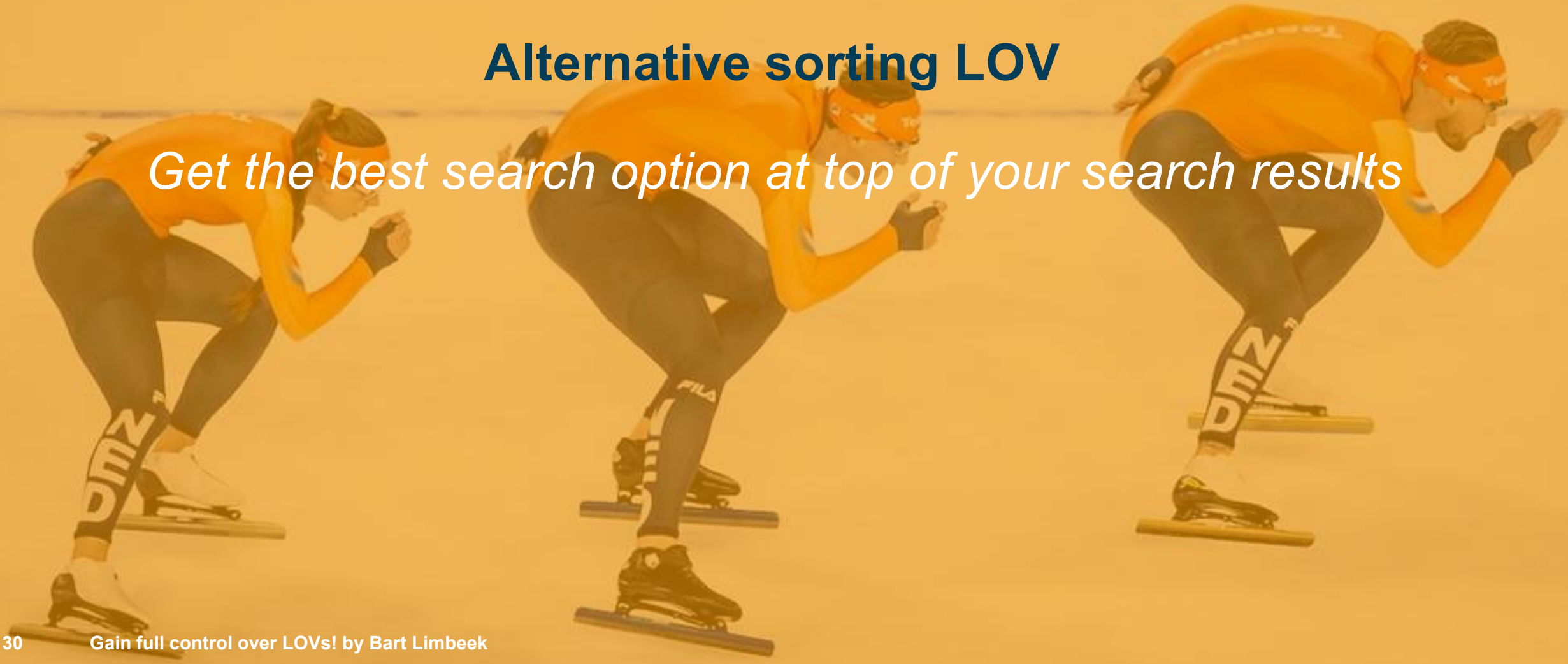
Extensive options

- Images in your results
- HTML in code
- Order by Item
- Extra Faceted search filters



Alternative sorting LOV

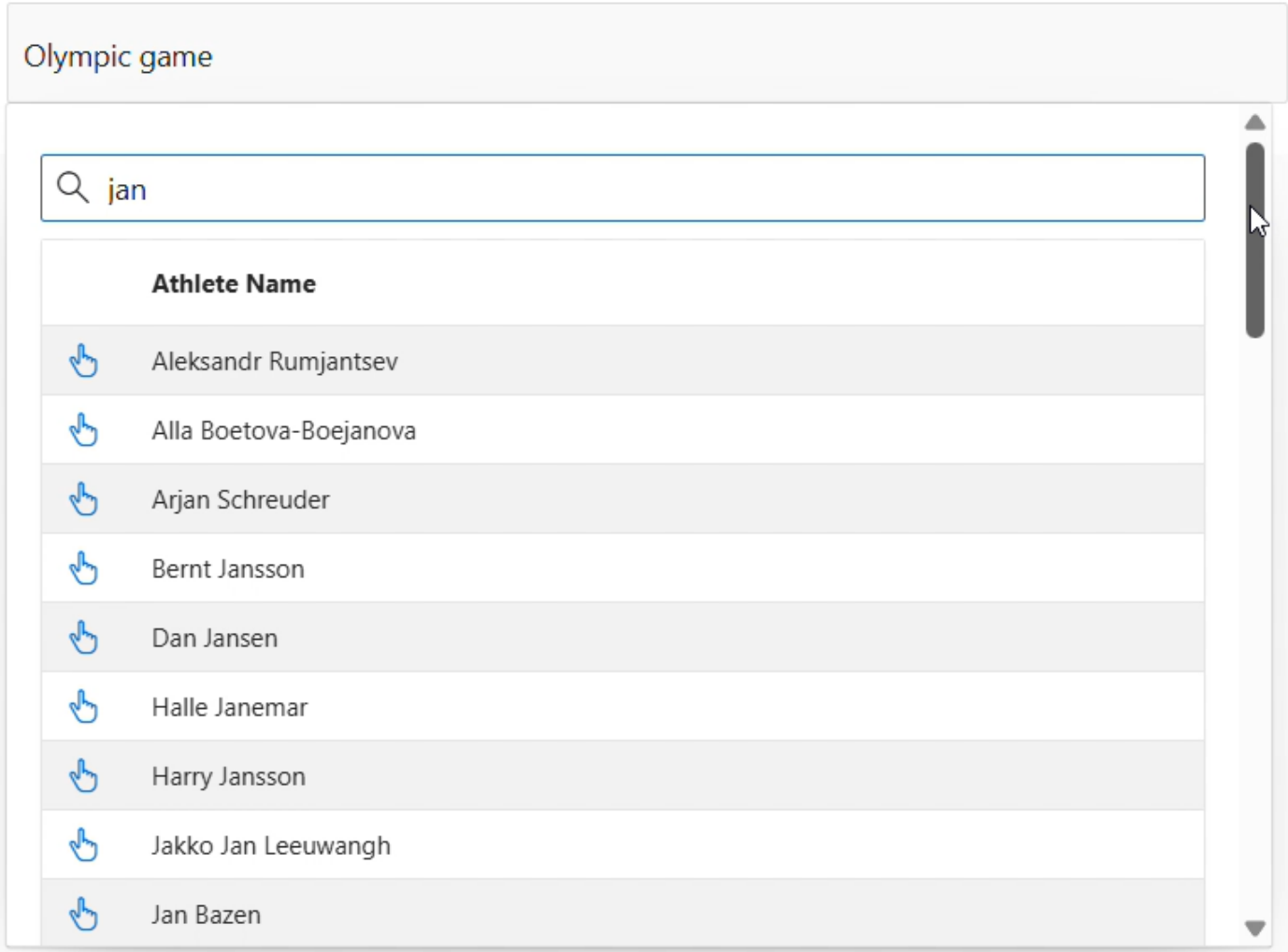
Get the best search option at top of your search results



Alternative sorting LOV

Standard situation

- **Default sorting**
  - Alphabetical
  - ID-based
- **Best result not shown at top**
- **Requires scrolling through results**
- **User must input more characters**



Alternative sorting LOV

Algorithm 1: Levenstein (Edit Distance)

- **Measures the minimum number of single-character edits to transform one string into another**
- **Every error is treated equally**
- **Best for**
  - Technical strings
  - Addresses
  - Shorter strings

Alternative sorting LOV

Algorithm 2: Jaro-Winkler

- **Calculates character matches and transpositions (swapped letters), with a specific bonus for matching prefixes.**
- **Errors at the end of a word are penalized less than at the beginning**
- **Best for**
  - Person names
  - Company names
  - Typos or misspelled words

Alternative sorting LOV

How to get the best option highest in my query?

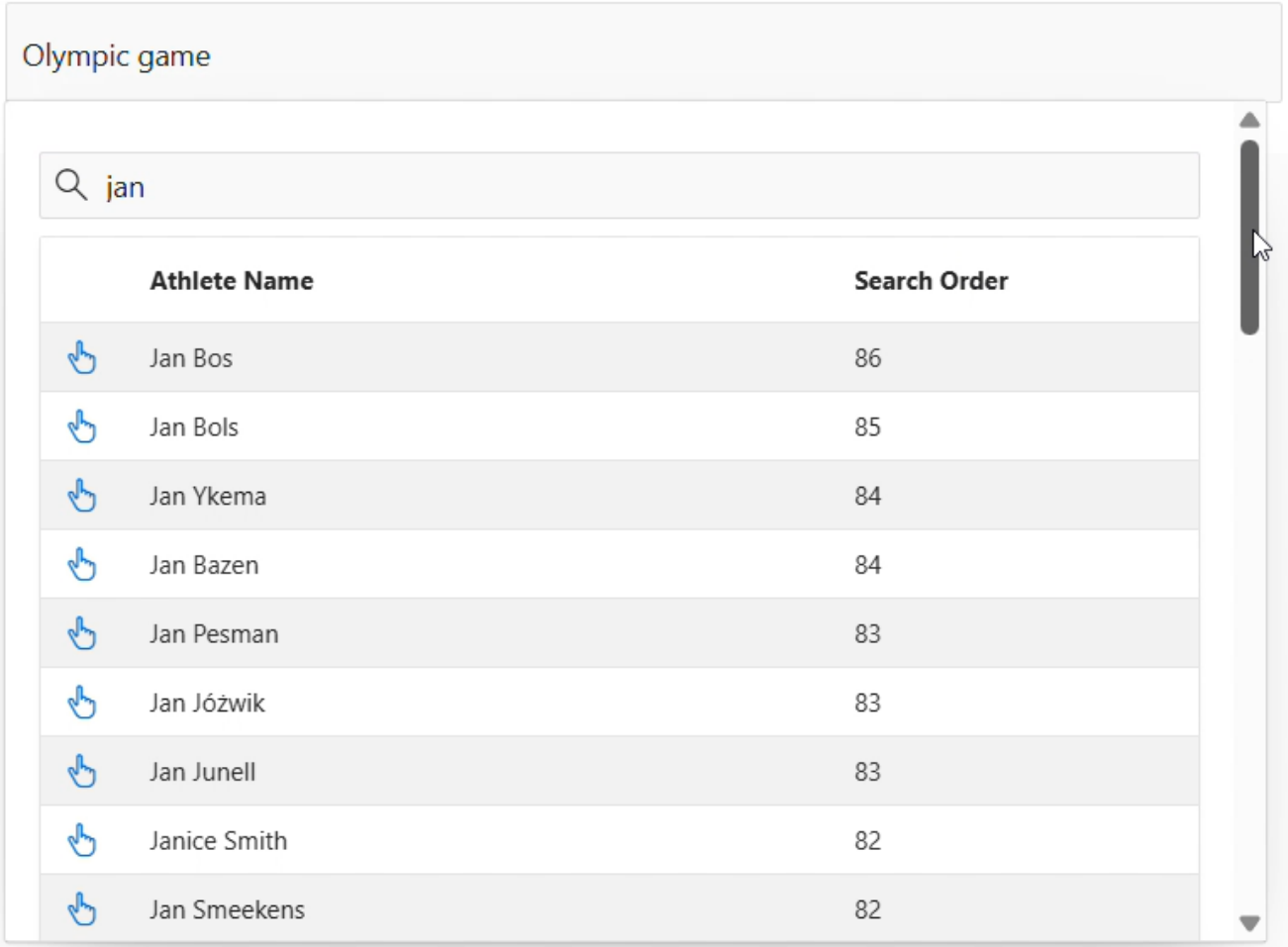
- **Fuzzy_match is only available in sql (26ai)**
- **Utl_match.jaro_winkler_similarity (19c)**
- **Results ranked by similarity (0-100)**
- **Results are case-senstive**

```
select  athlete_id
    ,   name
    ,   fuzzy_match(
            jaro_winkler
        ,   upper(name)
        ,   upper(:P2_SEARCH)
        ) as search_order
from    athletes
order by search_order
```

Alternative sorting LOV

Using fuzzy match

- Best match ranked at the top
- Higher scores for prefix matches (same starting letters)
- High performance and fast response time
- Fuzzy search possible in where clause

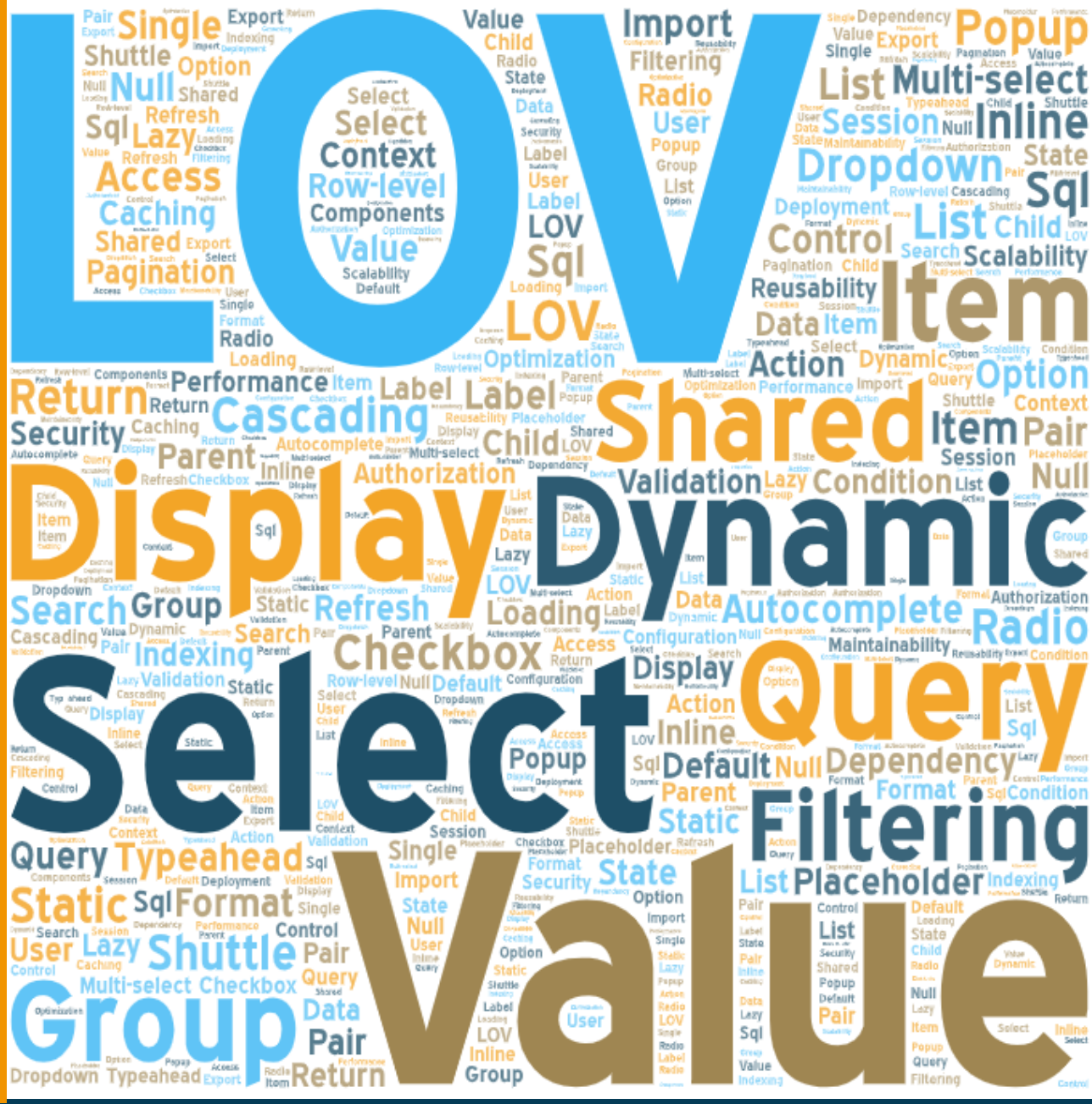


What are my closing thoughts?

- Default approach isn’t always the best.
- Focus on the user
- Build something great
- Ensure strong maintainability & security
- Hope you learned something new

TRANSFER SOLUTIONS

Questions?


LOV
Select
Value
Display
Dynamic
Shared
Query
Filtering
Item
Context
Row-level
Components
Autocomplete
Checkbox
Typeahead
Group
Shuttle
Multi-select
Inline
Popup
Dropdown
Session
Control
Reusability
Cascading
Performance
Optimization
Authorization
Validation
Maintainability
Dependency
Placeholder
Indexing
Pagination
Caching
Access
Refresh
Search
Parent
Child
Label
Security
Scalability
Deployment
Configuration