

# Mastering EXISTS Statements

*Your Dream Workfront Report EXISTS!*

Skye Hansen | Senior Product Manager for Workfront - Starbucks

Nathan Johnson | Technical Support Engineer III - Adobe



**Two approaches,  
one mission: make  
EXISTS make sense.**

## ORIGIN STORIES

In 2018, Skye got permission to delete 10,000 projects. Months later, she started discovering empty portfolios... #consequences

When Nathan joined Support, his background in SQL landed him in the Reporting pod. On one of his first customer calls, the customer wanted to create a proof approval report that was filtered by project status.

# Our Promise: EXISTS Will Finally Make Sense

1. **What is EXISTS??**

2. **Why EXISTS?**

The real-world problems EXISTS statements solve.

3. **Nathan's Approach**

Technical look at how EXISTS work and how to understand them

4. **Skye's Approach**

A business-driven method to build EXISTS statements efficiently.

5. **Putting It All Together**

Walk away ready to build your dream Workfront report.

# What is EXISTS? (context)

| Type of filter                       | Reason to use  | Examples   |
|--------------------------------------|--|--|
| Beginner:<br>Basic filters           | <ul style="list-style-type: none"><li>• 1:1 (each item in your report has one “yes” answer to the filter)</li><li>• 85% standard mode, 15% easy text mode</li></ul>                            | <ul style="list-style-type: none"><li>• Project report looking for projects created last week (each project has 1 entry date)</li><li>• Task report looking for the primary assigned user (each task has 1 task “owner”)</li></ul> |
| Intermediate:<br>Collections filters | <ul style="list-style-type: none"><li>• 1:Many (each item in your report has more than one possible answer, at least one of which is “yes”)</li><li>• 95% text mode, 5% cheat fields</li></ul> | <ul style="list-style-type: none"><li>• Program report looking for programs where the projects were created last week</li><li>• Task report looking for all assigned users (each task can have multiple assignees)</li></ul>       |

# Advanced: EXISTS filters

**EXISTS:1:\$\$OBJCODE=PROJ**

**EXISTS:1:tasks:ID=FIELD:ID**

**EXISTS:1:program:ownerID=\$\$USER.ID**

These are:

- Advanced, text mode filters, which allow us to work around limitations in standard reporting
- 100% text mode
- Many to many: items in one group are checked against items in another group; a match occurs when at least one pair meets the condition

# What can EXISTS do for you?

Can you get me a list of all templates that have not been used. Users have too many choices, so we want to retire some.

The screenshot shows the 'New Template Report' dialog box. At the top, there is a title bar with 'New Template Report' and a 'Report Settings' button. Below the title bar, there are four tabs: 'Columns (View)', 'Groupings', 'Filters', and 'Chart'. The 'Filters' tab is selected. Below the tabs, there is a section titled 'Set Filter Rules for your Report:' with two buttons: 'Apply an Existing Filter' and 'Switch to Standard Mode'. Below this section, there is a text area containing the following filter rules:  
EXISTS:B:\$\$EXISTSMOD=NOTEXISTS  
EXISTS:B:\$\$OBJCODE=PROJ  
EXISTS:B:templateID=FIELD:ID  
To the right of the text area is an 'Edit Text Mode' button. At the bottom of the dialog box, there are three buttons: 'Save + Close', 'Apply', and 'Cancel'.

We are spending too much on licenses. Can you determine who isn't using theirs.

The screenshot shows the 'New User Report' dialog box. At the top, there is a title bar with 'New User Report' and a 'Report Settings' button. Below the title bar, there are four tabs: 'Columns (View)', 'Groupings', 'Filters', and 'Chart'. The 'Filters' tab is selected. Below the tabs, there is a section titled 'Set Filter Rules for your Report:' with two buttons: 'Apply an Existing Filter' and 'Switch to Standard Mode'. Below this section, there is a text area containing the following filter rules:  
EXISTS:1:\$\$EXISTSMOD=NOTEXISTS  
EXISTS:1:\$\$OBJCODE=HOUR  
EXISTS:1:entryDate\_Mod=notnull  
EXISTS:1:ownerID=FIELD:ID  
To the right of the text area is an 'Edit Text Mode' button. At the bottom of the dialog box, there are three buttons: 'Save + Close', 'Apply', and 'Cancel'.

## EXERCISE 1

### **First challenge:**

Word Problem: Show me TASKS that are in PROGRAMS  
that are OWNED by the LOGGED IN USER

## First challenge:

Word Problem: Show me TASKS that are in PROGRAMS that are OWNED by the LOGGED IN USER

### First attempt: Creating a normal filter

The screenshot shows a web interface for creating a report. At the top is a text input field labeled "New Task Report". Below it are tabs for "Columns (View)" and "Groupings". A section titled "Set Filter Rules for your Report:" contains a button labeled "Select a field" and a link "Add another Filter Rule". A modal dialog titled "Select a field" is open, showing a search input with the text "program owner id" and a clear button. Below the search bar is the placeholder text "Type to filter items...". At the bottom of the modal, a message states "It doesn't look like that matches any available fields..." followed by a sad face emoji.

New Task Report

Columns (View) Groupings

Set Filter Rules for your Report:

Select a field

[Add another Filter Rule](#)

Select a field

program owner id

Type to filter items...

It doesn't look like that matches any available fields...



## Second attempt: Creating a normal text mode filter

### Handy Hint 1

#### Avoid creating any text mode from scratch

We often create similar **views** and **filters** in **standard mode** and then change to **text mode** and “add on”

e.g. Looking for the Task’s Program’s Owner ID?

#### Starting Point:

Started with Task’s  
Project Program ID  
in standard mode;  
added on in text mode

The screenshot shows a configuration window titled 'New Task Report'. It has two tabs: 'Columns (View)' and 'Groupings'. The 'Columns (View)' tab is active, showing a section 'Show in this column:' followed by a list of configuration parameters:

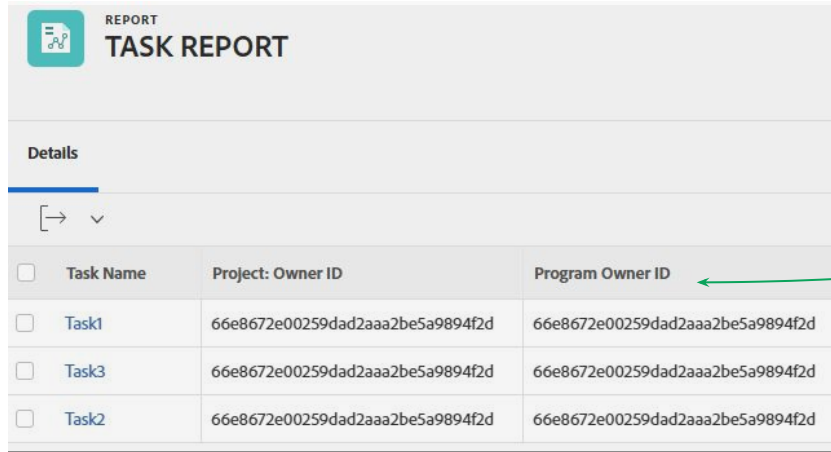
```
valuefield=project:programID
querysort=project:programID
valueformat=HTML
linkedname=project
namekey=view.relatedcolumn
namekeyargkey.0=project
namekeyargkey.1=programID
```

## First challenge:

Word Problem: Show me TASKS that are in PROGRAMS that are OWNED by the LOGGED IN USER

## Second attempt: Creating a normal text mode filter

Test your field in a column to see if it gives you results...



| <input type="checkbox"/> | Task Name | Project: Owner ID                | Program Owner ID                 |
|--------------------------|-----------|----------------------------------|----------------------------------|
| <input type="checkbox"/> | Task1     | 66e8672e00259dad2aaa2be5a9894f2d | 66e8672e00259dad2aaa2be5a9894f2d |
| <input type="checkbox"/> | Task3     | 66e8672e00259dad2aaa2be5a9894f2d | 66e8672e00259dad2aaa2be5a9894f2d |
| <input type="checkbox"/> | Task2     | 66e8672e00259dad2aaa2be5a9894f2d | 66e8672e00259dad2aaa2be5a9894f2d |

TASK REPORT

Columns (View) Groupings Filters

Show in this column:

```
valuefield=project:program:ownerID
valueformat=HTML
linkedname=project:program
namekey=view.relatedcolumn
displayname=Program Owner ID
namekeyangkey.0=project:program
namekeyangkey.1=ownerID
querysort=project:program:ownerID
```

## First challenge:

Word Problem: Show me TASKS that are in PROGRAMS that are OWNED by the LOGGED IN USER

## Second attempt: Creating a normal text mode filter

Our text mode gave us data in the column. Now try the exact same thing in a filter...

The screenshot shows a report builder interface with three tabs: 'Columns (View)', 'Groupings', and 'Filters'. The 'Filters' tab is selected and highlighted with a red box. Below the tabs, the text 'Set Filter Rules for your Report:' is followed by a text input field containing the filter rule: `project:program:ownerID=$$USER.ID` and `project:program:ownerID_Mod=in`. A red arrow points from this input field to a large red error box on the right. The error box contains a warning icon and the text: 'An error has occurred and we are working to resolve the issue. To continue with your work, try refreshing this browser page.' Below this text is a red banner with a white warning icon and the message: 'Invalid Parameter: Search Parameter value "project:program:ownerID"'. A red 'X' icon is in the top right corner of the banner.

What the heck happened?? 🙄

# **It's Too Far Away**

## **Why Columns Can See What Filters Can't**

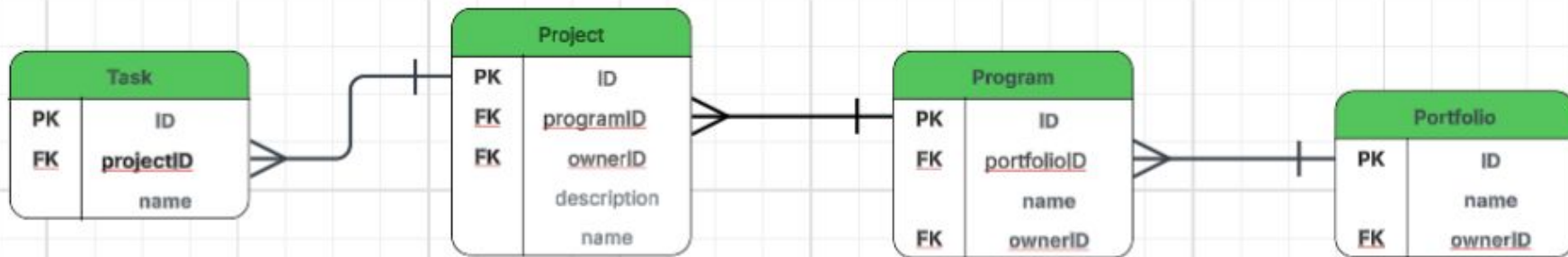
### **Why it works in Columns:**

- Our engineers designed columns to allow up to 6 table jumps.
- Columns don't filter rows; they just display fields that exist.

### **Why it fails in Filters:**

- Our engineers designed filters to allow one table jump.
- We limit how many joins (table jumps) a filter can make for performance and data consistency.

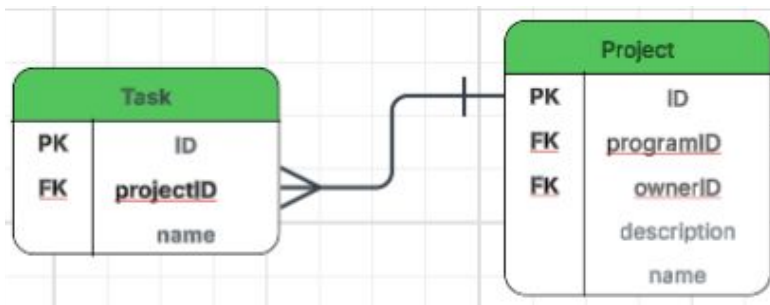
# How Objects Are Related



All objects in Workfront are linked to each other by their unique identifier.

- **Primary Key (PK)**: A unique identifier for each object record — like a phone number. No two records share the same PK.
- **Foreign Key (FK)**: A field on one object that links to the **Primary Key** of another. It creates a relationship between records.
- **One-to-Many**: A single project can connect to *multiple* tasks, but each of those tasks links back to only one project.

# Relationships in the API Explorer



A **reference** is a *link to one specific related object* — a **foreign key** pointing *upward* to its parent. For example, a Task has a reference to its only Project

A **collection** is a *list of related child objects* — all the records that point *back* to this one via their foreign key. A Project has a collection of many Tasks.

## Objects

Filter

API Version 19.0 ▼

Task

fields

references

collections

search

actions

TASK

accessRules

accessRules

allConditions

allConditions

allPriorities

allPriorities

allStatuses

allStatuses

approverStatuses

approverStatuses

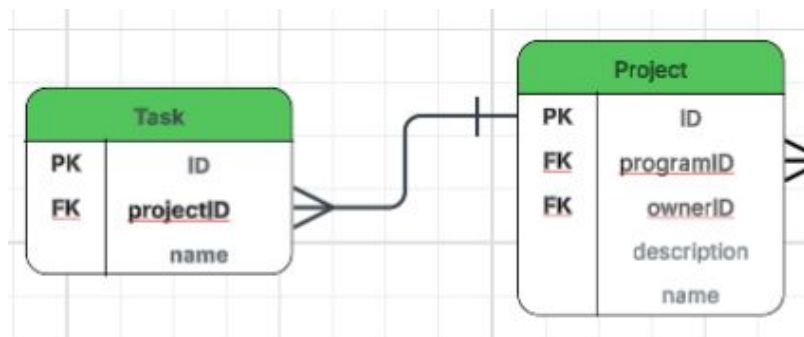
Assignments

assignments

awaitingApprovals

awaitingApprovals

An EXISTS filter requires us to choose a linking object so that the resulting filter does not violate the “one-table-jump rule”. We use the foreign key to unlock access to a table that is close enough to our target.



- **EXISTS:1:\$OBJCODE=PROJ**
  - Object Line - Program information cannot be reached from a Task report, but it can be from a Project report.
- **EXISTS:1:tasks:ID=FIELD:ID**
  - Linking Line - This is how I connect my report object type, Task, with the Project object type.
  - The right side of the equal sign uses FIELD as a stand in for the object I am on.
  - The left side is written from the linking object table.

## Word Problem Reminder:

**Show me TASKS (in projects) that are in PROGRAMS that are OWNED by the LOGGED IN USER**

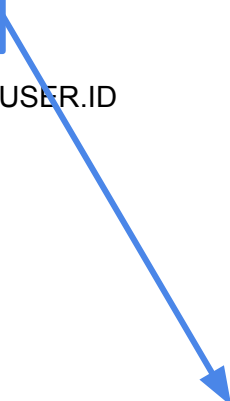
Now that we know how to link our objects, we only need the final line. This is our filter.

- **EXISTS:1:\$\$OBJCODE=PROJ**
- **EXISTS:1:tasks:ID=FIELD:ID**
- **EXISTS:1:program:ownerID=\$\$USER.ID**
  - Our filter. Now that we have successfully linked to the PROJECT object, we build a filter that would work on a project report
  - We are allowed one table jump in filters, so I can jump from project to program and filter on the ownerID.



# Skye's Easy text mode Filter Hack

EXISTS:1:\$\$OBJCODE=**PROJ**  
EXISTS:1:tasks:ID=FIELD.ID  
EXISTS:1:program:ownerID=\$\$USER.ID



Information needed is in Project Report

1. Set aside your Task Report, open a new tab and separately create a **Project** Report.
2. Build your **Project** filter in standard mode (*also a good way to test your results*).
3. Switch over to Text Mode.
4. Copy and paste the text mode filter\* into your exists statement (preface with "EXISTS:1:").

\* "in" is always assumed; no need to add

# Skye's Easy text mode Filter Hack - Visualized

EXISTS:1:\$\$OBJCODE=PROJ  
EXISTS:1:tasks:ID=FIELD:ID  
EXISTS:1:program:ownerID=\$\$USER.ID

Create Project Report

The diagram illustrates the process of creating a project report using a filter hack in text mode. It consists of two screenshots of the report configuration interface, connected by numbered arrows (1-4).

**Top Screenshot (Initial State):**

- ①** Points to the "Create Project Report" text.
- ②** Points to the "Switch to Text Mode" button.
- ③** Points to the text input field containing the filter rules.
- ④** Points to the "Program >> Owner ID" filter rule.

**Bottom Screenshot (Text Mode):**

- The "Switch to Text Mode" button is now labeled "Switch to Standard Mode".
- The text input field contains the filter rules: `program:ownerID=$$USER.ID` and `program:ownerID_Mod=in *`.
- An "Edit Text Mode" button is visible.

\* "in" is always assumed; no need to add

# Skye's Easy Textmode Filter Hack v.2

Tasks in Projects where the you are the program owner **AND** the Project's program was created this year

Columns (View) Groupings **Filters** Chart

Set Filter Rules for your Report:

|                       |   |         |   |             |                       |           |
|-----------------------|---|---------|---|-------------|-----------------------|-----------|
| Program >> Owner ID   | ⊖ | Equal   | ▼ |             | ▼                     | 🔍         |
|                       |   |         |   | 🔗 \$USER.ID | ✕                     |           |
| <b>AND</b> ▼          |   |         |   |             |                       |           |
| Program >> Entry Date | ⊖ | Between | ▼ | \$TODAYby   | 🔕 Set relative date ⓘ | \$TODAYey |
|                       |   |         |   |             | 🔕 Set relative date ⓘ |           |

[Add another Filter Rule](#)

EXISTS:1:\$OBJCODE=**PROJ**  
EXISTS:1:tasks:ID=FIELD:ID  
EXISTS:1:**program:ownerID=\$USER.ID**  
EXISTS:1:**program:entryDate=\$TODAYby**  
EXISTS:1:**program:entryDate\_Mod=between**  
EXISTS:1:**program:entryDate\_Range=\$TODAYey**

# Skye's Easy Textmode Filter Hack v.3

Tasks in Projects where the you are the program owner **OR** the Project's program was created this year

Columns (View) Groupings **Filters** Chart

Set Filter Rules for your Report:

Program >> Owner ID  Equal

[Add another Filter Rule](#)

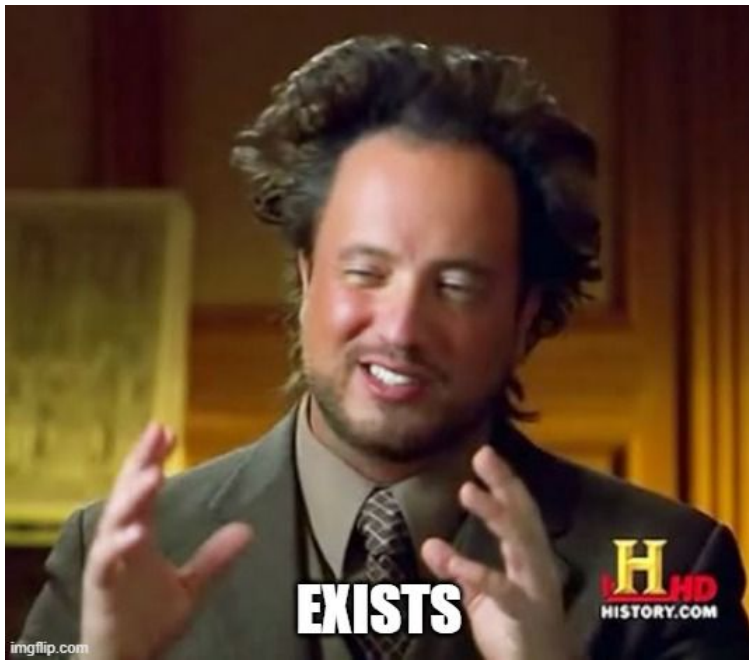
**OR**

Program >> Entry Date  Between  \$\$TODAYby ☐ Set relative date ⓘ  \$\$TODAYey ☐ Set relative date ⓘ

[Add another Filter Rule](#)

EXISTS:1:\$\$OBJCODE=**PROJ**  
EXISTS:1:tasks:ID=FIELD:ID  
EXISTS:1:**program:ownerID=\$\$USER.ID**

OR:1:EXISTS:1:\$\$OBJCODE=**PROJ**  
OR:1:EXISTS:1:tasks:ID=FIELD:ID  
OR:1:EXISTS:1:**program:entryDate=\$\$TODAYby**  
OR:1:EXISTS:1:**program:entryDate\_Mod=between**  
OR:1:EXISTS:1:**program:entryDate\_Range=\$\$TODAYey**



## Why is this so hard??

- Forgetting (or never realizing) that **EXISTS even exist**.
- Not knowing *when* to use it — or why your results don't match expectations.
- Linking lines feel like dark magic.
- OBJCODE headaches — which object links to which?

# Why EXISTS?

| Reason to use EXISTS   | Example ask  | Translated to plain English  |
|--|--|--|
| 1. <b>Many to Many</b> (A project can be shared to many users, and a user can have many projects shared to them)       | Show <b>projects</b> that are <b>shared</b> with a specific user (or team... or group... or role...) | Can you search through multiple sharing settings/rules from the project? (thing you are looking for doesn't have just one answer per object) |
| 2. <b>"At least one / Just one"</b> (I don't want a list of projects, I want a list of owners)                         | Show <b>users</b> who have created at least one <b>project</b> in the past 90 days                   | I don't care about this list of projects – can you check to see who is using their license?  |
| 3. Check for <b>"something outside your immediate family"</b> (task connects to project, but how do I get to program?) | Show a <b>program owner</b> all the <b>tasks</b> in their program                                    | Can you check for the task's project's program's owner ID? (sorry... what?)  |
| 4. Find something that isn't there ( <b>doesn't exist</b> , wouldn't show up in a report to begin with)                | Show <b>projects</b> that have <b>NO</b> completed <b>tasks</b>                                      | Can you look for missing items? i.e. not looking for items that are there and blank – they are absolutely not there.                         |

## EXERCISE 2

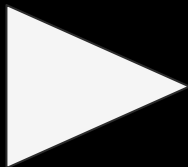
Take a breath



# Now Let's Level Up

**Word Problem:** Show me all projects that are shared with a specific user.

It's a common problem - you're trying to find out what was shared, and to whom. Let me DEMO how I would start →



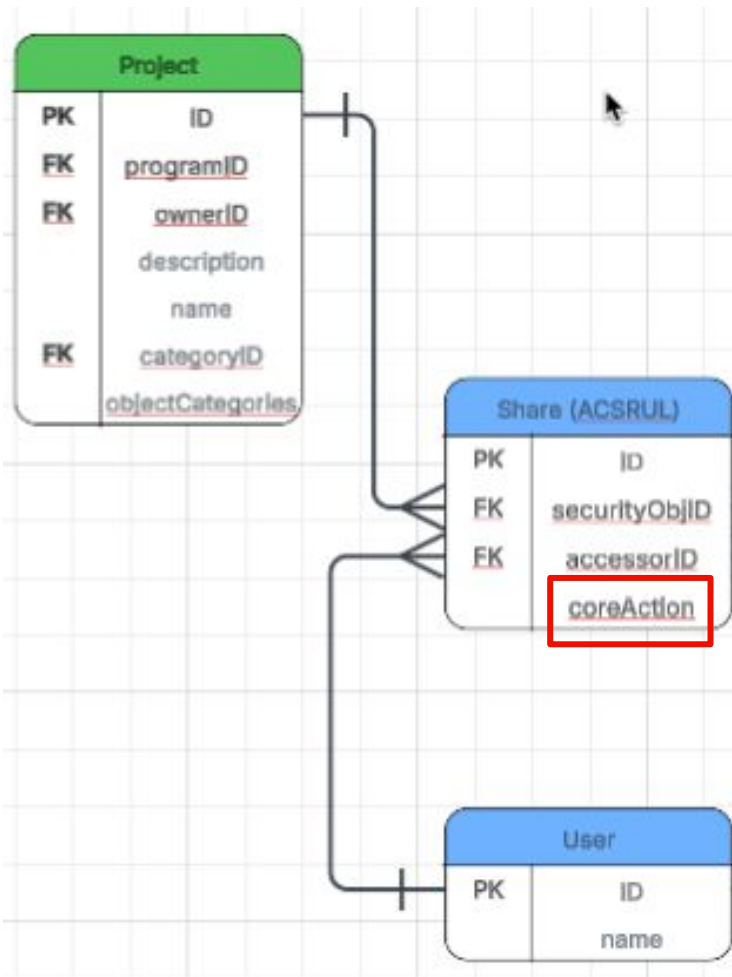
You need to have enough knowledge to know that a relationship between one project and multiple people means that we are looking at a relationship to MANY things (users) and so this will have to be a collection.



The screenshot shows the Adobe Developer console interface. At the top, there's a browser window with the URL 'developer.adobe.com/workfr...'. Below the browser, the Adobe Developer logo is visible. A 'Filter' section contains a 'Project' input field and an 'API Version 19.0' dropdown. The main content area is titled 'Project (Financial Data)' with a 'FINDAT' button. Below this, there's a 'Project' section with tabs for 'fields', 'references', 'collections', 'search', and 'actions'. The 'fields' tab is active, displaying a list of fields with their corresponding API names:

| Project                      | fields | references | collections | search | actions | PROJ                             |
|------------------------------|--------|------------|-------------|--------|---------|----------------------------------|
| BC Completion State          |        |            |             |        |         | <i>BCCompletionState</i>         |
| ID                           |        |            |             |        |         | <i>ID</i>                        |
| URL                          |        |            |             |        |         | <i>URL</i>                       |
| accessorIDs                  |        |            |             |        |         | <i>accessorIDs</i>               |
| Actual Benefit               |        |            |             |        |         | <i>actualBenefit</i>             |
| Actual Billable Expense Cost |        |            |             |        |         | <i>actualBillableExpenseCost</i> |
| Actual Completion Date       |        |            |             |        |         | <i>actualCompletionDate</i>      |
| Actual Cost                  |        |            |             |        |         | <i>actualCost</i>                |
| actualDurationExpression     |        |            |             |        |         | <i>actualDurationExpression</i>  |
| Actual Duration Minutes      |        |            |             |        |         | <i>actualDurationMinutes</i>     |
| Actual Expense Cost          |        |            |             |        |         | <i>actualExpenseCost</i>         |





**Many-to-Many:** In Workfront, a **user** can be shared to many **projects**, and each **project** can be shared with many **users**.

Since databases can't represent this directly, they use a **linking object** between the two.

Linking objects include assignments, objectCategories, userRoles, and so on. When you have a **many-to-many** relationship, the linking object line is easy.

EXISTS:A:\$\$OBJCODE=ACSRUL  
EXISTS:A:securityObjID=FIELD:ID

# Handy Hint 2

## Reference collections in a view (Use collections text mode in your report view)

Instead of making educated guesses, just brute-force-add all the likely suspects into one column and search through the results to see which ones could add to your linking line or your filter lines.

### Template: Collections text mode

```
valueformat=HTML  
textmode=true  
type=iterate  
listdelimiter=<p>  
displayname=Column Name  
listmethod=nested(collection object name).lists  
valueexpression=CONCAT("field: ",{field},"field2: ",{field2})
```

# Finding the Right Fields - Report Value Expression

Here goes  
nothing...

|                  |        |            |             |        |         |                  |
|------------------|--------|------------|-------------|--------|---------|------------------|
| Share            | fields | references | collections | search | actions | ACSRUL           |
| ID               |        |            |             |        |         | ID               |
| accessorID       |        |            |             |        |         | accessorID       |
| accessorObjCode  |        |            |             |        |         | accessorObjCode  |
| ancestorID       |        |            |             |        |         | ancestorID       |
| ancestorObjCode  |        |            |             |        |         | ancestorObjCode  |
| coreAction       |        |            |             |        |         | coreAction       |
| Customer ID      |        |            |             |        |         | customerID       |
| forbiddenActions |        |            |             |        |         | forbiddenActions |
| isInherited      |        |            |             |        |         | isInherited      |
| secondaryActions |        |            |             |        |         | secondaryActions |
| securityObjCode  |        |            |             |        |         | securityObjCode  |
| securityObjID    |        |            |             |        |         | securityObjID    |

## Edit Text Mode

```
displayname=Sharing Settings
listdelimiter=<p>
listmethod=nested(accessRules).lists
type=iterate
valueexpression=CONCAT('ID:',{ID},' - accessorID:',{accessorID},' - accessorObjCode: ',
{accessorObjCode},' - coreAction:',{coreAction},' - securityObjCode:',{securityObjCode},' -
securityObjID:',{securityObjID})
valueformat=HTML
```

## Details



Gantt

Report Default

Report Default

Report Default

Report Default

Report Default

Report Default

Report Default

Report Default

Report Default

| <input type="checkbox"/> Name              | Project ID (Linking Line)       | Who I'm Looking For (Filter)     | Sharing Settings  |
|--|---------------------------------|----------------------------------|---|
| <input type="checkbox"/> Bill Test Project | 639b99b10017a2b9d75f90c3b653b7b | 560da5f1000c1d14de09352e70ff2205 | ID: 639b99b10017a2e8ed5897174e44140d - accessorID: 560da5f1000c1d14de09352e70ff2205 - accessorObjCode: USER - coreAction: DELETE - securityObjCode: PROJ - securityObjID: 639b99b10017a2b9d75f90c3b653b7b |

# Exercise 2 Solution

Show me all projects that are incorrectly shared with a specific user  
(i.e. they needed manage, not view permissions)

EXISTS:A:\$\$OBJCODE=ACSRUL

EXISTS:A:securityObjID=FIELD:ID

EXISTS:A:accessorID=5c1e831f02dc1e4ae10d3701979cbd08

EXISTS:A:coreAction=VIEW

coreAction

Field Name: **coreAction**

Flags: **NOT\_GROUPABLE** (NOT\_GROUPABLE)

Field Type: **string**

Enum Type: **com.attask.common.constants.ActionTypeEnum**

Possible Values: **ADD** (Add)

**EDIT** (Edit)

**LIMITED\_EDIT** (Limited Edit)

**VIEW** (View)

**DELETE** (Delete)

# NOTEXISTS

What if I only want things **without** that match? By adding one more line, we can turn the EXISTS filter into a NOTEXISTS filter.

Show me projects that have NOT been shared with Nathan. He is a super user and should have access to everything.

```
EXISTS:A:$$EXISTSMOD=NOTEXISTS
EXISTS:A:$$OBJCODE=ACSRUL
EXISTS:A:securityObjID=FIELD:ID
EXISTS:A:accessorID=5c1e831f02dc1e4ae10d3701979cbd08
```

Show me tasks in Programs that I do NOT own.

```
EXISTS:1:$$EXISTSMOD=NOTEXISTS
EXISTS:1:$$OBJCODE=PROJ
EXISTS:1:tasks:ID=FIELD:ID
EXISTS:1:program:ownerID=$$USER.ID
```

# Why EXISTS? (clues)

| Reason to use EXISTS  | Example ask   | Report / OBJCODE / link  |
|---|---|--|
| 1. Many to Many (A project can be shared to many users, and a user can have many projects shared to them)       | Show <b>projects</b> that are <b>shared</b> with a specific <b>user</b> (or team... or group... or role...) | Report: <b>Project</b><br>OBJCODE: <b>ACSRUL</b><br>Link: <b>securityObjID</b> |
| 2. “At least one / Just one” (I don’t want a list of projects, I want a list of owners)                         | Show <b>users</b> who have <b>created</b> at least one <b>project</b> in the past 90 days                   | Report: <b>User</b><br>OBJCODE: <b>PROJ</b><br>Link: <b>enteredByID</b>        |
| 3. Check for “something outside your immediate family” (task connects to project, but how do I get to program?) | Show a <b>program owner</b> all the <b>(project) tasks</b> in their program                                 | Report: <b>Task</b><br>OBJCODE: <b>PROJ</b><br>Link: <b>tasks:ID</b>           |
| 4. Find something that isn’t there (doesn’t exist, wouldn’t show up in a report to begin with)                  | Show <b>projects</b> that have NO completed <b>tasks</b>  | Report: <b>Project</b><br>OBJCODE: <b>TASK</b><br>Link: <b>projectId</b>       |

# Tips

- **Search Community** – do a keyword search on the Adobe Experience League for unique OBJCODEs like ACSRUL or ARVSTS – these words don't exist (haha) in the english language so it's easier to pull up any EXISTS filters to help you get a quick start.
- **Make yourself a template you can paste as a starting prompt**, in your own words. Mine looks like this:

In the **project report you are in, show projects**

Without                      EXISTS:a:\$EXISTSMOD=NOTEXISTS

Tasks                        EXISTS:a:\$OBJCODE=TASK

On the Project            EXISTS:a:projectID=FIELD:ID

Task named “test”      EXISTS:a:name=test

This is because the fastest thing I remember is: on the **task**, it's called “projectID” and on the **project**, it's called “ID”

- **Start your own reference library**: keep samples of successful filters so you can mix and match. Always include a description!
- Shout out to **keyboard macro tools** such as Keyboard Maestro, Espanso or AutoHotkey, and **reference library tools** such as Airtable

# Homework! 🧐

## **Week 1 — "Under the Hood"** (*anchor → explore*)

Pick a type of report you build often (for example, a project, task, or issue report). Open the API Explorer, and locate that object. From there, trace out to collections that could expand what your report can show. How far can you get before you feel like EXISTS would be necessary? (*for beginners who need a safe starting point*)

## **Week 2 — "Getting to Carnegie Hall"** (*practice, practice, practice*)

Run a search in the Adobe Experience League looking for EXISTS as a keyword in the Workfront questions forum, and without looking at the answer, try to come up with your answer independently. (*for learners who want to test themselves*)

## **Week 3 — "Chain Breaker"** (*link → extended link*)

Find an object in the API Explorer that clearly links to another object (for example, Task → Assignment). Then push one step further: what happens if you need to filter based on the *next* link in the chain? Sketch how EXISTS could step in to make that connection. (*for learners ready to stretch their understanding*)

## **Week 4 — "Six Degrees of Kevin Bacon's Workfront"** (*explore → anchor*)

Open the API Explorer and choose an object you don't normally work with. Can you trace a path from that object back to something you know well (like project, task, issue, user, or custom form)? If so, sketch how EXISTS might help you bridge the gap. (*for learners who enjoy puzzles or exploration*)

## **Week 5 — "Unsupported Explorer Hunt"** (*unsupported → insight*)

Open the unsupported API Explorer and pick an object or field that doesn't appear in the supported one. What do you notice about it? Can you imagine a report where pulling that data in with EXISTS might be useful? What questions does this raise for you? (*for advanced learners or those curious about the backend*)

COME CHAT WITH US AT OUR AMA (ask me anything) IN DECEMBER!