RX 2026: Lava Tool Templates

A copy-and-paste reference for the Lava AI tools built in the "Creating AI Skills and Tools in Lava" session. Each tool starts on its own page and includes its full template, ready to paste into the Rock tool editor.

Tags skill (Demo 1)

1. LookupPersonTags Lookup
2. AddOrUpdatePersonTag AddOrUpdate
3. GetPersonTag Get
4. DeletePersonTag Delete
5. ListTaggedPeople List, ...ForTag
6. TagPerson Action
7. UntagPerson Action/Delete

Fundraising skill (Demo 2)

1. LookupFundraisingOpportunities Lookup
2. GetOpportunityInsights Insights
3. ListUpcomingTripsProgress Insights, cross-entity
4. GetTripReadiness Insights, cross-domain

LookupPersonTags

Skill: Tags | Lookup | Execute Lava

Looks up the organization and personal tags for the current person.

```
{% comment %}
  ========================================================================
  Tool:   LookupPersonTags   (taxonomy: Lookup)
  Skill:  Tags
  Type:   Execute Lava
  Description: Looks up the organization and personal tags for the current person.
  Params: (none)
  Returns: the Person tags this person can use, as { IdKey, Name, IsPersonal }.
  Notes:  Filtered to the Person entity type since this skill only tags people.
          securityenabled:'true' scopes visibility, so a person's personal tags
          stay private to them via Rock's tag security. We keep the payload
          tiny on purpose (Lookup feeds other tools; add a Get later if you
          need detail).
  ========================================================================
{% endcomment %}

{% comment %} Person tags only - resolve the Person EntityType id first. {% endcomment %}
{% entitytype where:'Name == "Rock.Model.Person"' securityenabled:'false' iterator:'entityTypes' %}
{% endentitytype %}
{% assign personEntityType = entityTypes | First %}
{% assign personEntityTypeId = personEntityType.Id %}

{% comment %} 1. LOAD: active Person tags, ordered by name. {% endcomment %}
{% tag where:'IsActive == true && EntityTypeId == {{ personEntityTypeId }}' securityenabled:'true' sort:'Name' iterator:'tags' %}
{% endtag %}

{% comment %} 2. FORMAT: keep only the fields the agent needs. Entity commands expose IdKey directly. {% endcomment %}
{% assign items = null %}
{% for tag in tags %}
    {% assign isPersonal = false %}
    {% if tag.OwnerPersonAliasId != null %}
        {% assign isPersonal = true %}
    {% endif %}

    {% assign item = '' | AddToDictionary:'IdKey',tag.IdKey %}
    {% assign item = item | AddToDictionary:'Name',tag.Name %}
    {% assign item = item | AddToDictionary:'IsPersonal',isPersonal %}
    {% assign items = items | AddToArray:item %}
{% endfor %}

{% comment %} 3. RETURN: empty list auto-collapses to NoData. {% endcomment %}
{% assign result = 'Success' | AgentToolResult:items %}
```

AddOrUpdatePersonTag

Skill: Tags | AddOrUpdate | Execute Lava

Adds a new or updates an existing personal tag for the current person.

```
{% comment %}
  ===============================================================================
  Tool:   AddOrUpdatePersonTag   (taxonomy: AddOrUpdate)
  Skill:  Tags
  Type:   Execute Lava
  Description: Adds a new or updates an existing personal tag for the current person.
  Params: tagIdKey    (String, optional)  - omit to CREATE, supply to UPDATE
          name        (String, optional)  - required when creating (no tagIdKey)
          description (String, optional)  - the tag description
  Returns: the saved tag as { IdKey, Name }, or an Error with the reason.
  Notes:  On create we make a PERSONAL tag on the Person entity, owned by the
          current person; a name is required. Update is limited to the caller's
          own personal tags - organization tags are read-only here.
          IMPORTANT: the modify command pulls [[ property ]] blocks out of the
          RAW markup with a regex, so a {% if %} around a property does NOT gate
          it - the property is always applied. That is why update defaults each
          field to the existing value (to avoid blanking), and why create-time
          values are read inline rather than via pre-assigned variables.
  ===============================================================================
{% endcomment %}

{% comment %} IdKey in -> int. No/invalid key means create (id 0). {% endcomment %}
{% assign tagId = tagIdKey | FromIdHash | Default:0 %}
{% assign canProceed = true %}

{% if tagId == 0 %}
    {% comment %} CREATE guard: need a name, and we must know who will own it. {% endcomment %}
    {% if name == null or name == '' %}
        {% assign canProceed = false %}
        {% assign result = 'Error' | AgentToolResult:'A name is required when creating a tag.' %}
    {% elsif CurrentPerson == null or CurrentPerson.PrimaryAliasId == null %}
        {% assign canProceed = false %}
        {% assign result = 'Error' | AgentToolResult:'Could not determine the current person to own the tag.' %}
    {% endif %}
{% else %}
    {% comment %} UPDATE guard: only the caller's own personal tags may be changed. {% endcomment %}
    {% tag id:'{{ tagId }}' securityenabled:'false' include:'OwnerPersonAlias' iterator:'existingTags' %}
    {% endtag %}
    {% assign existingTag = existingTags | First %}

    {% if existingTag == null %}
        {% assign canProceed = false %}
        {% assign result = 'Error' | AgentToolResult:'That tag was not found.' %}
    {% elsif existingTag.OwnerPersonAliasId == null %}
```

```
        {% assign canProceed = false %}
        {% assign result = 'Error' | AgentToolResult:'That is an organization tag. This tool only changes your personal tags.' %}
    {% elsif existingTag.OwnerPersonAlias.PersonId != CurrentPerson.Id %}
        {% assign canProceed = false %}
        {% assign result = 'Error' | AgentToolResult:'That personal tag belongs to someone else.' %}
    {% endif %}
{% endif %}

{% if canProceed and tagId == 0 %}
    {% comment %} CREATE. Property values render with the live context; owner and
       type are read inline so they never depend on a pre-assigned variable. {% endcomment %}
    {% modifytag id:'0' securityenabled:'true' return:'modifyResult' %}
        [[ property name:'Name' ]]{{ name }}[[ endproperty ]]
        [[ property name:'Description' ]]{{ description }}[[ endproperty ]]
        [[ property name:'OwnerPersonAliasId' ]]{{ CurrentPerson.PrimaryAliasId }}[[ endproperty ]]
        [[ property name:'IsActive' ]]true[[ endproperty ]]
        [[ property name:'EntityTypeId' ]]{% entitytype where:'Name == "Rock.Model.Person"' securityenabled:'false' %}{{ entitytype.Id }}{% endentitytype %}[[ endproperty ]]
    {% endmodifytag %}
{% elsif canProceed %}
    {% comment %} UPDATE. A {% if %} cannot gate a property, so default each field to
       the existing value - an omitted field is preserved, not blanked. {% endcomment %}
    {% modifytag id:'{{ tagId }}' securityenabled:'true' return:'modifyResult' %}
        [[ property name:'Name' ]]{{ name | Default:existingTag.Name }}[[ endproperty ]]
        [[ property name:'Description' ]]{{ description | Default:existingTag.Description }}[[ endproperty ]]
    {% endmodifytag %}
{% endif %}

{% if canProceed %}
    {% comment %} RETURN the saved item, or a clean Error. The result exposes the
       entity under its friendly name (modifyResult.Tag) - there is no 'Object'
       wrapper despite what the docs show. {% endcomment %}
    {% if modifyResult.Success %}
        {% assign savedTag = modifyResult.Tag %}
        {% assign item = '' | AddToDictionary:'IdKey',savedTag.IdKey %}
        {% assign item = item | AddToDictionary:'Name',savedTag.Name %}
        {% assign result = 'Success' | AgentToolResult:item %}
    {% else %}
        {% assign result = 'Error' | AgentToolResult:modifyResult.ErrorMessage %}
    {% endif %}
{% endif %}
```

GetPersonTag

Skill: Tags | Get | Execute Lava

Gets the details of a tag (organization or personal) for the current person.

```
{% comment %}
  ========================================================================
  Tool:   GetPersonTag   (taxonomy: Get)
  Skill:  Tags
  Type:   Execute Lava
  Description: Gets the details of a tag (organization or personal) for the current person.
  Params: personTagIdKey  (String, required)  - the tag to fetch
  Returns: the tag as { IdKey, Name, Description, IsPersonal, IsActive },
           or an Error if it is not found.
  Notes:  securityenabled:'true' scopes visibility, so a Get on someone else's
          personal tag comes back as not found. Returns Error (not NoData) when
          the one requested item is missing.
  ========================================================================
{% endcomment %}

{% comment %} IdKey in -> int. {% endcomment %}
{% assign tagId = personTagIdKey | FromIdHash | Default:0 %}

{% comment %} LOAD the one tag (security scopes visibility). {% endcomment %}
{% tag id:'{{ tagId }}' securityenabled:'true' iterator:'tags' %}
{% endtag %}
{% assign tag = tags | First %}

{% comment %} FORMAT and RETURN, or Error if missing. {% endcomment %}
{% if tag == null %}
    {% assign result = 'Error' | AgentToolResult:'That tag was not found.' %}
{% else %}
    {% assign isPersonal = false %}
    {% if tag.OwnerPersonAliasId != null %}
        {% assign isPersonal = true %}
    {% endif %}

    {% assign item = '' | AddToDictionary:'IdKey',tag.IdKey %}
    {% assign item = item | AddToDictionary:'Name',tag.Name %}
    {% assign item = item | AddToDictionary:'Description',tag.Description %}
    {% assign item = item | AddToDictionary:'IsPersonal',isPersonal %}
    {% assign item = item | AddToDictionary:'IsActive',tag.IsActive %}
    {% assign result = 'Success' | AgentToolResult:item %}
{% endif %}
```

DeletePersonTag

Skill: Tags | Delete | Execute Lava

Deletes an existing personal tag for the current person.

```
{% comment %}
  ============================================================================
  Tool:   DeletePersonTag   (taxonomy: Delete)
  Skill:  Tags
  Type:   Execute Lava
  Description: Deletes an existing personal tag for the current person.
  Params: personTagIdKey  (String, required)  - the tag to delete
  Returns: a success message, or an Error with the reason.
  Notes:  Limited to the caller's own personal tags - organization tags and
          other people's personal tags cannot be deleted here. Deletion is
          permanent; secure this tool and/or require confirmation at the agent
          level (the Lava template cannot prompt).
  ============================================================================
{% endcomment %}

{% comment %} IdKey in -> int. {% endcomment %}
{% assign tagId = personTagIdKey | FromIdHash | Default:0 %}
{% assign canProceed = true %}

{% comment %} GUARD: only the caller's own personal tag may be deleted. {% endcomment %}
{% tag id:'{{ tagId }}' securityenabled:'false' include:'OwnerPersonAlias' iterator:'existingTags' %}
{% endtag %}
{% assign existingTag = existingTags | First %}

{% if existingTag == null %}
    {% assign canProceed = false %}
    {% assign result = 'Error' | AgentToolResult:'That tag was not found.' %}
{% elsif existingTag.OwnerPersonAliasId == null %}
    {% assign canProceed = false %}
    {% assign result = 'Error' | AgentToolResult:'That is an organization tag. This tool only deletes your personal tags.' %}
{% elsif existingTag.OwnerPersonAlias.PersonId != CurrentPerson.Id %}
    {% assign canProceed = false %}
    {% assign result = 'Error' | AgentToolResult:'That personal tag belongs to someone else.' %}
{% endif %}

{% if canProceed %}
    {% comment %} DELETE (permanent). {% endcomment %}
    {% deletetag id:'{{ tagId }}' securityenabled:'true' return:'deleteResult' %}
    {% enddeletetag %}

    {% if deleteResult.Success %}
        {% assign result = 'Success' | AgentToolResult:'The tag was deleted.' %}
    {% else %}
        {% assign result = 'Error' | AgentToolResult:deleteResult.ErrorMessage %}
    {% endif %}
```

```
46  {% endif %}
```

ListTaggedPeople

Skill: Tags | List, ...ForTag | Execute Lava

Lists the people tagged with a given tag.

```
{% comment %}
  ========================================================================
  Tool:   ListTaggedPeople   (taxonomy: List, ...ForTag)
  Skill:  Tags
  Type:   Execute Lava
  Description: Lists the people tagged with a given tag.
  Params: tagIdKey  (String, required)  - the tag whose people to list
  Returns: the tagged people as a list of { IdKey, Name, Email }.
  Notes:  SQL earns it here - the join is across TaggedItem's polymorphic
          EntityGuid to Person.Guid, which an entity command cannot express
          cleanly. List tools stay out of chat history.
  ========================================================================
{% endcomment %}

{% comment %} IdKey in -> int. {% endcomment %}
{% assign tagId = tagIdKey | FromIdHash | Default:0 %}

{% comment %} LOAD: people tagged with this tag. Pass tagId as a SQL parameter
   (@tagId) rather than embedding the value - the fully injection-safe pattern. {% endcomment %}
{% sql tagId:'{{ tagId }}' %}
    SELECT
        [p].[Id]
        , [p].[NickName]
        , [p].[LastName]
        , [p].[Email]
    FROM [TaggedItem] AS [ti]
    INNER JOIN [Person] AS [p] ON [p].[Guid] = [ti].[EntityGuid]
    WHERE [ti].[TagId] = @tagId
    ORDER BY [p].[LastName], [p].[NickName]
{% endsql %}

{% comment %} FORMAT: SQL rows have a bare Id, so hash it with ToIdHash. {% endcomment %}
{% assign items = null %}
{% for row in results %}
    {% assign idKey = row.Id | ToIdHash %}
    {% capture fullName %}{{ row.NickName }} {{ row.LastName }}{% endcapture %}
    {% assign fullName = fullName | Trim %}

    {% assign item = '' | AddToDictionary:'IdKey',idKey %}
    {% assign item = item | AddToDictionary:'Name',fullName %}
    {% assign item = item | AddToDictionary:'Email',row.Email %}
    {% assign items = items | AddToArray:item %}
{% endfor %}

{% comment %} RETURN: list tools stay out of chat history. {% endcomment %}
{% assign result = 'Success' | AgentToolResult:items %}
```

```
47  {% assign result = result | AgentToolNoHistory %}
```

TagPerson

Skill: Tags | Action | Execute Lava

Applies a tag to a person.

```
{% comment %}
  ========================================================================
  Tool:   TagPerson   (taxonomy: Action)
  Skill:  Tags
  Type:   Execute Lava
  Description: Applies a tag to a person.
  Params: personIdKey  (String, required)  - the person to tag
          tagIdKey     (String, required)  - the tag to apply
  Returns: a success message, or an Error with the reason.
  Notes:  Works with any tag the caller can use (organization or their own
          personal) - securityenabled:'true' on the tag load hides others'
          personal tags. The link is created with a parameterized SQL INSERT
          because the modify command cannot set a Guid property (EntityGuid).
          Idempotent via NOT EXISTS.
  ========================================================================
{% endcomment %}

{% assign tagId = tagIdKey | FromIdHash | Default:0 %}
{% assign personId = personIdKey | FromIdHash | Default:0 %}
{% assign canProceed = true %}

{% comment %} Resolve the tag (security-scoped) and the person. {% endcomment %}
{% tag id:'{{ tagId }}' securityenabled:'true' iterator:'tags' %}
{% endtag %}
{% assign tag = tags | First %}

{% person id:'{{ personId }}' securityenabled:'false' iterator:'people' %}
{% endperson %}
{% assign person = people | First %}

{% if tag == null %}
    {% assign canProceed = false %}
    {% assign result = 'Error' | AgentToolResult:'That tag was not found.' %}
{% elsif person == null %}
    {% assign canProceed = false %}
    {% assign result = 'Error' | AgentToolResult:'That person was not found.' %}
{% endif %}

{% if canProceed %}
    {% comment %} Create the link with a parameterized INSERT (idempotent). {% endcomment %}
    {% assign entityTypeId = tag.EntityTypeId %}
    {% assign personGuid = person.Guid %}

    {% sql statement:'command' tagId:'{{ tagId }}' entityTypeId:'{{ entityTypeId }}' entityGuid:'{{ personGuid }}' %}
        INSERT INTO [TaggedItem] ([Guid], [TagId], [EntityTypeId], [EntityGuid], [IsSystem])
        SELECT NEWID(), @tagId, @entityTypeId, @entityGuid, 0
```

```
        WHERE NOT EXISTS (
            SELECT 1 FROM [TaggedItem]
            WHERE [TagId] = @tagId AND [EntityGuid] = @entityGuid
        )
    {% endsql %}

    {% assign result = 'Success' | AgentToolResult:'Tag applied.' %}
{% endif %}
```

UntagPerson

Skill: Tags | Action/Delete | Execute Lava

Removes a tag from a person.

```
{% comment %}
  ===============================================================================
  Tool:   UntagPerson   (taxonomy: Action/Delete)
  Skill:  Tags
  Type:   Execute Lava
  Description: Removes a tag from a person.
  Params: personIdKey  (String, required)  - the person to untag
          tagIdKey     (String, required)  - the tag to remove
  Returns: a success message, or an Error with the reason.
  Notes:  Removes the TaggedItem that links the tag to the person with a
          parameterized SQL DELETE (the join references the person by Guid).
          Idempotent - removing a link that is not there is a no-op success.
  ===============================================================================
{% endcomment %}

{% assign tagId = tagIdKey | FromIdHash | Default:0 %}
{% assign personId = personIdKey | FromIdHash | Default:0 %}
{% assign canProceed = true %}

{% comment %} Resolve the tag (security-scoped) and the person. {% endcomment %}
{% tag id:'{{ tagId }}' securityenabled:'true' iterator:'tags' %}
{% endtag %}
{% assign tag = tags | First %}

{% person id:'{{ personId }}' securityenabled:'false' iterator:'people' %}
{% endperson %}
{% assign person = people | First %}

{% if tag == null %}
    {% assign canProceed = false %}
    {% assign result = 'Error' | AgentToolResult:'That tag was not found.' %}
{% elsif person == null %}
    {% assign canProceed = false %}
    {% assign result = 'Error' | AgentToolResult:'That person was not found.' %}
{% endif %}

{% if canProceed %}
    {% comment %} Remove the link with a parameterized DELETE. {% endcomment %}
    {% assign personGuid = person.Guid %}

    {% sql statement:'command' tagId:'{{ tagId }}' entityGuid:'{{ personGuid }}' %}
        DELETE FROM [TaggedItem]
        WHERE [TagId] = @tagId AND [EntityGuid] = @entityGuid
    {% endsql %}

    {% assign result = 'Success' | AgentToolResult:'Tag removed from the person.' %}
```

```
47  {% endif %}
```

LookupFundraisingOpportunities

Skill: Fundraising | Lookup | Execute Lava

Looks up the active fundraising opportunities (mission trips, projects).

```
{% comment %}
  ========================================================================
  Tool:   LookupFundraisingOpportunities   (taxonomy: Lookup)
  Skill:  Fundraising
  Type:   Execute Lava
  Description: Looks up the active fundraising opportunities (mission trips, projects).
  Params: (none)
  Returns: the opportunities as a list of { IdKey, Name }.
  Notes:  An opportunity is a Group of the Fundraising Opportunity group type
          (4BE7FC44-332D-40A8-978E-47B7035D7A0C). The whole Fundraising skill
          uses SQL - the raised-vs-goal math needs it - and passes every value
          as a parameter.
  ========================================================================
{% endcomment %}

{% sql fundraisingTypeGuid:'4BE7FC44-332D-40A8-978E-47B7035D7A0C' %}
    SELECT [g].[Id], [g].[Name]
    FROM [Group] [g]
    INNER JOIN [GroupType] [gt] ON [gt].[Id] = [g].[GroupTypeId]
    WHERE [gt].[Guid] = @fundraisingTypeGuid
      AND [g].[IsActive] = 1
    ORDER BY [g].[Name]
{% endsql %}

{% assign items = null %}
{% for row in results %}
    {% assign idKey = row.Id | ToIdHash %}

    {% assign item = '' | AddToDictionary:'IdKey',idKey %}
    {% assign item = item | AddToDictionary:'Name',row.Name %}
    {% assign items = items | AddToArray:item %}
{% endfor %}

{% assign result = 'Success' | AgentToolResult:items %}
```

GetOpportunityInsights

Skill: Fundraising | Insights | Execute Lava

Gets fundraising progress (raised vs goal) for one opportunity and its participants.

```
{% comment %}
  ============================================================================
  Tool:   GetOpportunityInsights   (taxonomy: Insights)
  Skill:  Fundraising
  Type:   Execute Lava
  Description: Gets fundraising progress (raised vs goal) for one opportunity and its participants.
  Params: opportunityIdKey  (String, required)  - the fundraising opportunity (group)
  Returns: { Opportunity, TotalGoal, TotalRaised, Participants: [{ Name, Goal, Raised }] }.
  Notes:  Goal per participant = their IndividualFundraisingGoal attribute, else
          the group's default. Raised = sum of FinancialTransactionDetail.Amount
          where the detail points at that participant's GroupMember row
          (EntityType GroupMember, EntityId = GroupMember.Id). Trip totals are
          the same whether the opportunity is Individual or Family mode.
  ============================================================================
{% endcomment %}

{% assign groupId = opportunityIdKey | FromIdHash | Default:0 %}

{% comment %} Load the opportunity for its name and to confirm it exists. {% endcomment %}
{% group id:'{{ groupId }}' securityenabled:'false' iterator:'groups' %}
{% endgroup %}
{% assign opportunity = groups | First %}

{% if opportunity == null %}
    {% assign result = 'Error' | AgentToolResult:'That fundraising opportunity was not found.' %}
{% else %}
    {% comment %} LOAD per-participant goal and raised. {% endcomment %}
    {% sql groupId:'{{ groupId }}' %}
        DECLARE @gmEntityTypeId INT = (SELECT [Id] FROM [EntityType] WHERE [Guid] = '49668B95-FEDC-43DD-8085-D2B0D6343C48');
        DECLARE @groupEntityTypeId INT = (SELECT [Id] FROM [EntityType] WHERE [Name] = 'Rock.Model.Group');
        DECLARE @gmGoalAttributeId INT = (SELECT TOP 1 [Id] FROM [Attribute] WHERE [Key] = 'IndividualFundraisingGoal' AND [EntityTypeId] = @gmEntityTypeId);
        DECLARE @groupGoalAttributeId INT = (SELECT TOP 1 [Id] FROM [Attribute] WHERE [Key] = 'IndividualFundraisingGoal' AND [EntityTypeId] = @groupEntityTypeId);
        DECLARE @groupDefaultGoal DECIMAL(18,2) = (
            SELECT TRY_CONVERT(DECIMAL(18,2), [av].[Value])
            FROM [AttributeValue] [av]
            WHERE [av].[AttributeId] = @groupGoalAttributeId AND [av].[EntityId] = @groupId
        );

        SELECT
            [p].[NickName] + ' ' + [p].[LastName] AS [Name],
            COALESCE(TRY_CONVERT(DECIMAL(18,2), [mav].[Value]), @groupDefaultGoal, 0) AS [Goal],
            COALESCE([raised].[Total], 0) AS [Raised]
        FROM [GroupMember] [gm]
```

```
        INNER JOIN [Person] [p] ON [p].[Id] = [gm].[PersonId]
        LEFT JOIN [AttributeValue] [mav] ON [mav].[AttributeId] = @gmGoalAttributeId AND [mav].
[EntityId] = [gm].[Id]
        LEFT JOIN (
            SELECT [ftd].[EntityId] AS [GroupMemberId], SUM([ftd].[Amount]) AS [Total]
            FROM [FinancialTransactionDetail] [ftd]
            WHERE [ftd].[EntityTypeId] = @gmEntityTypeId AND [ftd].[EntityId] IS NOT NULL
            GROUP BY [ftd].[EntityId]
        ) [raised] ON [raised].[GroupMemberId] = [gm].[Id]
        WHERE [gm].[GroupId] = @groupId
          AND [gm].[GroupMemberStatus] = 1
          AND [gm].[IsArchived] = 0
        ORDER BY [p].[LastName], [p].[NickName]
    {% endsql %}

    {% comment %} FORMAT: per-participant rows + trip totals. {% endcomment %}
    {% assign participants = null %}
    {% assign totalGoal = 0 %}
    {% assign totalRaised = 0 %}
    {% for row in results %}
        {% assign totalGoal = totalGoal | Plus:row.Goal %}
        {% assign totalRaised = totalRaised | Plus:row.Raised %}

        {% assign p = '' | AddToDictionary:'Name',row.Name %}
        {% assign p = p | AddToDictionary:'Goal',row.Goal %}
        {% assign p = p | AddToDictionary:'Raised',row.Raised %}
        {% assign participants = participants | AddToArray:p %}
    {% endfor %}

    {% assign item = '' | AddToDictionary:'Opportunity',opportunity.Name %}
    {% assign item = item | AddToDictionary:'TotalGoal',totalGoal %}
    {% assign item = item | AddToDictionary:'TotalRaised',totalRaised %}
    {% assign item = item | AddToDictionary:'Participants',participants %}
    {% assign result = 'Success' | AgentToolResult:item %}
    {% assign result = result | AgentToolNoHistory %}
{% endif %}
```

ListUpcomingTripsProgress

Skill: Fundraising | Insights, cross-entity | Execute Lava

Lists fundraising progress across all upcoming opportunities, worst-funded first.

```
{% comment %}
  ========================================================================
  Tool:   ListUpcomingTripsProgress   (taxonomy: Insights, cross-entity)
  Skill:  Fundraising
  Type:   Execute Lava
  Description: Lists fundraising progress across all upcoming opportunities, worst-funded first.
  Params: (none)
  Returns: a list of { Opportunity, TotalGoal, TotalRaised, PercentComplete,
           StartDate, EndDate }, ordered least-funded first.
  Notes:  Same raised-vs-goal math as GetOpportunityInsights, rolled up per
          trip across every active Fundraising Opportunity group. "Upcoming" =
          the trip's end date is today or later; the date comes from the
          comma-delimited OpportunityDateRange attribute, parsed in Lava the
          way Rock's own fundraising blocks do (Split:',' then the Date filter).
          Excluded from history (it can be large).
  ========================================================================
{% endcomment %}

{% comment %} LOAD: per-trip totals + percent + the raw date range. {% endcomment %}
{% sql fundraisingTypeGuid:'4BE7FC44-332D-40A8-978E-47B7035D7A0C' %}
    DECLARE @gmEntityTypeId INT = (SELECT [Id] FROM [EntityType] WHERE [Guid] = '49668B95-FEDC-43DD-8085-D2B0D6343C48');
    DECLARE @groupEntityTypeId INT = (SELECT [Id] FROM [EntityType] WHERE [Name] = 'Rock.Model.Group');
    DECLARE @gmGoalAttributeId INT = (SELECT TOP 1 [Id] FROM [Attribute] WHERE [Key] = 'IndividualFundraisingGoal' AND [EntityTypeId] = @gmEntityTypeId);
    DECLARE @groupGoalAttributeId INT = (SELECT TOP 1 [Id] FROM [Attribute] WHERE [Key] = 'IndividualFundraisingGoal' AND [EntityTypeId] = @groupEntityTypeId);
    DECLARE @dateRangeAttributeId INT = (SELECT TOP 1 [Id] FROM [Attribute] WHERE [Key] = 'OpportunityDateRange' AND [EntityTypeId] = @groupEntityTypeId);

    ;WITH [MemberData] AS (
        SELECT
            [g].[Id] AS [GroupId],
            [g].[Name] AS [GroupName],
            COALESCE(TRY_CONVERT(DECIMAL(18,2), [mav].[Value]), TRY_CONVERT(DECIMAL(18,2), [gav].[Value]), 0) AS [Goal],
            COALESCE([raised].[Total], 0) AS [Raised]
        FROM [Group] [g]
        INNER JOIN [GroupType] [gt] ON [gt].[Id] = [g].[GroupTypeId] AND [gt].[Guid] = @fundraisingTypeGuid
        INNER JOIN [GroupMember] [gm] ON [gm].[GroupId] = [g].[Id] AND [gm].[GroupMemberStatus] = 1 AND [gm].[IsArchived] = 0
        LEFT JOIN [AttributeValue] [mav] ON [mav].[AttributeId] = @gmGoalAttributeId AND [mav].[EntityId] = [gm].[Id]
        LEFT JOIN [AttributeValue] [gav] ON [gav].[AttributeId] = @groupGoalAttributeId AND [gav].[EntityId] = [g].[Id]
```

```
        LEFT JOIN (
            SELECT [ftd].[EntityId] AS [GroupMemberId], SUM([ftd].[Amount]) AS [Total]
            FROM [FinancialTransactionDetail] [ftd]
            WHERE [ftd].[EntityTypeId] = @gmEntityTypeId AND [ftd].[EntityId] IS NOT NULL
            GROUP BY [ftd].[EntityId]
        ) [raised] ON [raised].[GroupMemberId] = [gm].[Id]
        WHERE [g].[IsActive] = 1
    )
    SELECT
        [md].[GroupName],
        SUM([md].[Goal]) AS [TotalGoal],
        SUM([md].[Raised]) AS [TotalRaised],
        CASE WHEN SUM([md].[Goal]) > 0 THEN ROUND(SUM([md].[Raised]) / SUM([md].[Goal]) * 100, 1) ELSE 0 END AS [PercentComplete],
        (SELECT TOP 1 [dav].[Value] FROM [AttributeValue] [dav] WHERE [dav].[AttributeId] = @dateRangeAttributeId AND [dav].[EntityId] = [md].[GroupId]) AS [DateRange]
    FROM [MemberData] [md]
    GROUP BY [md].[GroupId], [md].[GroupName]
    ORDER BY [PercentComplete] ASC, [md].[GroupName]
{% endsql %}

{% comment %} FORMAT: parse the date range, drop clearly-past trips, keep worst-first order. {% endcomment %}
{% assign nowKey = 'Now' | Date:'yyyyMMdd' | AsInteger %}
{% assign items = null %}
{% for row in results %}
    {% assign isUpcoming = true %}
    {% assign startDate = '' %}
    {% assign endDate = '' %}

    {% if row.DateRange and row.DateRange != '' %}
        {% assign parts = row.DateRange | Split:',' %}
        {% assign partsSize = parts | Size %}

        {% if partsSize >= 1 %}
            {% assign startDate = parts[0] | Date:'yyyy-MM-dd' %}
            {% assign endRaw = parts[0] %}
            {% if partsSize == 2 %}
                {% assign endRaw = parts[1] %}
            {% endif %}
            {% assign endDate = endRaw | Date:'yyyy-MM-dd' %}
            {% assign endKey = endRaw | Date:'yyyyMMdd' | AsInteger %}
            {% if endKey < nowKey %}
                {% assign isUpcoming = false %}
            {% endif %}
        {% endif %}
    {% endif %}

    {% if isUpcoming %}
        {% assign item = '' | AddToDictionary:'Opportunity',row.GroupName %}
        {% assign item = item | AddToDictionary:'TotalGoal',row.TotalGoal %}
        {% assign item = item | AddToDictionary:'TotalRaised',row.TotalRaised %}
        {% assign item = item | AddToDictionary:'PercentComplete',row.PercentComplete %}
```

```
        {% assign item = item | AddToDictionary:'StartDate',startDate %}
        {% assign item = item | AddToDictionary:'EndDate',endDate %}
        {% assign items = items | AddToArray:item %}
    {% endif %}
{% endfor %}

{% assign result = 'Success' | AgentToolResult:items %}
{% assign result = result | AgentToolNoHistory %}
```

GetTripReadiness

Skill: Fundraising | Insights, cross-domain | Execute Lava

Shows each trip participant's fundraising status and any unmet trip requirements (paperwork).

```
{% comment %}
  ========================================================================
  Tool:   GetTripReadiness   (taxonomy: Insights, cross-domain)
  Skill:  Fundraising
  Type:   Execute Lava
  Description: Shows each trip participant's fundraising status and any unmet trip requirements (paperwork).
  Params: opportunityIdKey  (String, required)  - the fundraising opportunity (group)
  Returns: { Opportunity, Participants: [{ Name, Goal, Raised,
           IsBehindOnFundraising, UnmetRequirements: [names] }] }.
  Notes:  Fuses two domains. Fundraising reuses the GetOpportunityInsights math.
          A requirement is unmet when GroupMemberRequirement.RequirementMetDateTime
          is NULL; names come from GroupRequirementType via GroupRequirement.
          STRING_AGG requires SQL Server 2017+. Excluded from history.
  ========================================================================
{% endcomment %}

{% assign groupId = opportunityIdKey | FromIdHash | Default:0 %}

{% comment %} Load the opportunity for its name and to confirm it exists. {% endcomment %}
{% group id:'{{ groupId }}' securityenabled:'false' iterator:'groups' %}
{% endgroup %}
{% assign opportunity = groups | First %}

{% if opportunity == null %}
    {% assign result = 'Error' | AgentToolResult:'That fundraising opportunity was not found.' %}
{% else %}
    {% comment %} LOAD per-participant fundraising + unmet requirements. {% endcomment %}
    {% sql groupId:'{{ groupId }}' %}
        DECLARE @gmEntityTypeId INT = (SELECT [Id] FROM [EntityType] WHERE [Guid] = '49668B95-FEDC-43DD-8085-D2B0D6343C48');
        DECLARE @groupEntityTypeId INT = (SELECT [Id] FROM [EntityType] WHERE [Name] = 'Rock.Model.Group');
        DECLARE @gmGoalAttributeId INT = (SELECT TOP 1 [Id] FROM [Attribute] WHERE [Key] = 'IndividualFundraisingGoal' AND [EntityTypeId] = @gmEntityTypeId);
        DECLARE @groupGoalAttributeId INT = (SELECT TOP 1 [Id] FROM [Attribute] WHERE [Key] = 'IndividualFundraisingGoal' AND [EntityTypeId] = @groupEntityTypeId);
        DECLARE @groupDefaultGoal DECIMAL(18,2) = (
            SELECT TRY_CONVERT(DECIMAL(18,2), [av].[Value])
            FROM [AttributeValue] [av]
            WHERE [av].[AttributeId] = @groupGoalAttributeId AND [av].[EntityId] = @groupId
        );

        SELECT
            [p].[NickName] + ' ' + [p].[LastName] AS [Name],
            COALESCE(TRY_CONVERT(DECIMAL(18,2), [mav].[Value]), @groupDefaultGoal, 0) AS [Goal],
            COALESCE([raised].[Total], 0) AS [Raised],
```

```
            [unmet].[UnmetRequirements] AS [UnmetRequirements]
        FROM [GroupMember] [gm]
        INNER JOIN [Person] [p] ON [p].[Id] = [gm].[PersonId]
        LEFT JOIN [AttributeValue] [mav] ON [mav].[AttributeId] = @gmGoalAttributeId AND [mav].
[EntityId] = [gm].[Id]
        LEFT JOIN (
            SELECT [ftd].[EntityId] AS [GroupMemberId], SUM([ftd].[Amount]) AS [Total]
            FROM [FinancialTransactionDetail] [ftd]
            WHERE [ftd].[EntityTypeId] = @gmEntityTypeId AND [ftd].[EntityId] IS NOT NULL
            GROUP BY [ftd].[EntityId]
        ) [raised] ON [raised].[GroupMemberId] = [gm].[Id]
        LEFT JOIN (
            SELECT [gmr].[GroupMemberId], STRING_AGG([grt].[Name], ', ') AS [UnmetRequirements]
            FROM [GroupMemberRequirement] [gmr]
            INNER JOIN [GroupRequirement] [gr] ON [gr].[Id] = [gmr].[GroupRequirementId]
            INNER JOIN [GroupRequirementType] [grt] ON [grt].[Id] = [gr].[GroupRequirementTypeId]
            WHERE [gmr].[RequirementMetDateTime] IS NULL
            GROUP BY [gmr].[GroupMemberId]
        ) [unmet] ON [unmet].[GroupMemberId] = [gm].[Id]
        WHERE [gm].[GroupId] = @groupId
          AND [gm].[GroupMemberStatus] = 1
          AND [gm].[IsArchived] = 0
        ORDER BY [p].[LastName], [p].[NickName]
    {% endsql %}

    {% comment %} FORMAT: per-participant money + paperwork status. {% endcomment %}
    {% assign participants = null %}
    {% for row in results %}
        {% assign isBehind = false %}
        {% if row.Goal > 0 and row.Raised < row.Goal %}
            {% assign isBehind = true %}
        {% endif %}

        {% assign unmetList = null %}
        {% if row.UnmetRequirements and row.UnmetRequirements != '' %}
            {% assign unmetList = row.UnmetRequirements | Split:', ' %}
        {% endif %}

        {% assign item = '' | AddToDictionary:'Name',row.Name %}
        {% assign item = item | AddToDictionary:'Goal',row.Goal %}
        {% assign item = item | AddToDictionary:'Raised',row.Raised %}
        {% assign item = item | AddToDictionary:'IsBehindOnFundraising',isBehind %}
        {% assign item = item | AddToDictionary:'UnmetRequirements',unmetList %}
        {% assign participants = participants | AddToArray:item %}
    {% endfor %}

    {% assign item = '' | AddToDictionary:'Opportunity',opportunity.Name %}
    {% assign item = item | AddToDictionary:'Participants',participants %}
    {% assign result = 'Success' | AgentToolResult:item %}
    {% assign result = result | AgentToolNoHistory %}
{% endif %}
```