AccessPlan.prototype (/js/AccessPlan).unnestInner

```
AccessPlan.prototype.unnestInner(
   inputColumn as String,
   valueColumn as String,
   [ordinalColumn as String?]
) as ModifyPlan
```

Summary

This function flattens an array value into multiple rows.Then performs a prototype.joinInner (./ModifyPlan.prototype.joinInner) on the rest of the rows.

Parameters	
inputColumn	The input column, which contains an array, to flatten into rows. This can be a string of the column name or an op.col (./op.col). Use op.viewCol (./op.viewCol) or op.schemaCol (./op.schemaCol) if you need to identify columns in the two views that have the same column name.
valueColumn	The output column which contains the flattened array values. This can be a string of the column name or an op.col (./op.col). Use op.viewCol (./op.viewCol) or op.schemaCol (./op.schemaCol) as needed.
ordinalColumn	The ordinalColumn is optional. If specified, an additional column will be added to the rows of flattened array values, starting from 1. This can be a string of the column name or an op.col (./op.col). Use op.viewCol (./op.viewCol) or op.schemaCol (./op.schemaCol) as needed.

Usage Notes

`unnestInner` is a method of the following classes:

- AccessPlan (/js/AccessPlan)
- ModifyPlan (/js/ModifyPlan)

Example

```
const op = require('/MarkLogic/optic');
const rows =
  [
    {rowId: 1, desc: ['ball', 'box']},
    {rowId: 2, desc: 'square'},
    {rowId: 3, desc: null}
  ];
const outputCols   = [
      {"column":"rowId",   "type":"integer"},
      {"column":"desc",    "type":"none", "nullable":true}
      ];
op.fromParam('rows', "", outputCols)
  .unnestInner('desc','descUnnest','ordinality')
  .orderBy(['rowId', 'ordinality'])
  .result('object', {"rows":rows});

/* This returns
[{"rowId":1, "desc":["ball", "box"], "descUnnest":"ball", "ordinality":1},
{"rowId":1, "desc":["ball", "box"], "descUnnest":"box", "ordinality":2},
{"rowId":2, "desc":"square", "descUnnest":"square", "ordinality":1}]
*/
```

Example

```
// Insert a template
declareUpdate();
const tde = require("/MarkLogic/tde.xqy");
const template = xdmp.toJSON(
{
  "template":{
    "context":"office",
    "rows":[
      {
        "schemaName":"unnestSchema",
        "viewName":"unnestView",
        "columns":[
          {
            "name":"department",
            "scalarType":"string",
            "val":"department",
            "nullable":true,
            "invalidValues":"ignore"
          },
          {
            "name":"teamMembers",
            "scalarType":"string",
            "val":"teamMembers",
            "nullable":true,
            "invalidValues":"ignore"
          }
        ]
      }
    ]
  }
}
);
tde.templateInsert('/optic/unnest/unnestTemplate.json', template)

//insert a document
declareUpdate();
let doc =  {office:[
            {department:"Engineering", teamMembers: 'Bob,Alice'},
            {department:"Sales", teamMembers:'Robert,Cindy'},
            {department:"Marketing", teamMembers:""},
            {department:"CEO-Office", teamMembers:null}
]};
 xdmp.documentInsert('/optic/unnest/doc1.json', doc)

 //perform a unnestInner
const op = require('/MarkLogic/optic');
op.fromView("unnestSchema","unnestView")
   .bind([
       op.as('teamMemberSequence', op.fn.tokenize(op.col('teamMembers'),','))
       ])
```

```
   .unnestInner('teamMemberSequence','teamMember', 'ordinality')
   .orderBy(['department', 'ordinality'])
   .result();

/* This returns
[{"unnestSchema.unnestView.department":"Engineering", "teamMemberSequence":["Bob", "Alice"], "unnestSc
{"unnestSchema.unnestView.department":"Engineering", "teamMemberSequence":["Bob", "Alice"], "unnestSch
{"unnestSchema.unnestView.department":"Sales", "teamMemberSequence":["Robert", "Cindy"], "unnestSchema
{"unnestSchema.unnestView.department":"Sales", "teamMemberSequence":["Robert", "Cindy"], "unnestSchema
*/
```