FILTER

Cria uma nova coluna retirando o elemento especificado do array de cada campo.

```
SELECT
        categories,
        FILTER (categories, category -> category <> "Engineering Blog") woEngineering
FROM DatabricksBlog
```

```
1  SELECT
2    temps,
3    FILTER(temps, t -> t > 18) highTemps
4  FROM DeviceData
5
```

▸ (1) Spark Jobs

	temps	highTemps
1	▸ [16, 13, 19, 11, 9, 23, 18, 13, 18, 17, 12, 12]	▸ [19, 23]
2	▸ [26, 17, 19, 13, 9, 12, 10, 12, 1, 13, 16, 12]	▸ [26, 19]
3	▸ [11, 13, 19, 8, 14, 16, 13, 14, 14, 9, 7, 12]	▸ [19]
4	▸ [20, 18, 20, 18, 11, 14, 17, 24, 17, 15, 19, 22]	▸ [20, 20, 24, 19, 22]
5	▸ [3, 1, 7, 8, 2, -4, 7, 1, 0, 8, -2, 3]	[]

Showing the first 1000 rows.

FILTER + WHERE

Cria uma nova coluna retirando o elemento especificado do array de cada campo. Mas, apenas quando a condição do WHERE for atendida.

```
SELECT
  *
FROM
  (
    SELECT
      authors, title,
      FILTER(categories, category -> category = "Engineering Blog") AS blogType
```

```
    FROM
      DatabricksBlog
  )
WHERE
  size(blogType) > 0
```

EXISTS

Cria uma nova coluna com uma flag se valor especificado existe no array de determinada célula.

```
SELECT
        categories,
        EXISTS (categories, c -> c = "Company Blog") companyFlag
FROM DatabricksBlog
```

```
1  SELECT
2    temps,
3    EXISTS(temps, t -> t > 23) highTempsFlag
4  FROM DeviceData
```

▸ (1) Spark Jobs

	temps	highTempsFlag
1	▸ [16, 13, 19, 11, 9, 23, 18, 13, 18, 17, 12, 12]	false
2	▸ [26, 17, 19, 13, 9, 12, 10, 12, 1, 13, 16, 12]	true
3	▸ [11, 13, 19, 8, 14, 16, 13, 14, 14, 9, 7, 12]	false
4	▸ [20, 18, 20, 18, 11, 14, 17, 24, 17, 15, 19, 22]	true
5	▸ [3, 1, 7, 8, 2, -4, 7, 1, 0, 8, -2, 3]	false

Showing the first 1000 rows.

TRANSFORM

Cria uma coluna transformando elementos de outra coluna com base nas especificações.

```
SELECT
  TRANSFORM(categories, cat -> LOWER(cat)) lwrCategories
FROM DatabricksBlog
```

```
SELECT
  temps temps_C,
  TRANSFORM (temps, t -> ((t * 9) div 5) + 32 ) temps_F
FROM DeviceData;
```

```
1  SELECT
2    temps temps_C,
3    TRANSFORM (temps, t -> ((t * 9) div 5) + 32 ) temps_F
4  FROM DeviceData;
5
```

▸ (1) Spark Jobs

	temps_C	temps_F
1	▸ [16, 13, 19, 11, 9, 23, 18, 13, 18, 17, 12, 12]	▸ [60, 55, 66, 51, 48, 73, 64, 55, 64, 62, 53, 53]
2	▸ [26, 17, 19, 13, 9, 12, 10, 12, 1, 13, 16, 12]	▸ [78, 62, 66, 55, 48, 53, 50, 53, 33, 55, 60, 53]
3	▸ [11, 13, 19, 8, 14, 16, 13, 14, 14, 9, 7, 12]	▸ [51, 55, 66, 46, 57, 60, 55, 57, 57, 48, 44, 53]
4	▸ [20, 18, 20, 18, 11, 14, 17, 24, 17, 15, 19, 22]	▸ [68, 64, 68, 64, 51, 57, 62, 75, 62, 59, 66, 71]
5	▸ [3, 1, 7, 8, 2, -4, 7, 1, 0, 8, -2, 3]	▸ [37, 33, 44, 46, 35, 25, 44, 33, 32,

Showing the first 1000 rows.

REDUCE

Faz o merge dos campos presentes no array para chegar em apenas um valor, depois aplica uma função chegando ao valor final.

```
CREATE OR REPLACE TEMPORARY VIEW Co2LevelsTemporary
AS
```

```
  SELECT
    dc_id,
    device_type,
    co2Level,
    REDUCE(co2Level, 0, (c, acc) -> c + acc, acc ->(acc div size(co2Level))) as
averageCo2Level

  FROM DeviceData
  SORT BY averageCo2Level DESC;

SELECT * FROM Co2LevelsTemporary
```

```sql
CREATE OR REPLACE TEMPORARY VIEW Co2LevelsTemporary
AS
  SELECT
    dc_id,
    device_type,
    co2_level,
    REDUCE(co2_level, 0, (c, acc) -> c + acc, acc ->(acc div size(co2_level))) as averageCo2Level

  FROM DeviceData
  SORT BY averageCo2Level DESC;

SELECT * FROM Co2LevelsTemporary
```

▸ (1) Spark Jobs

	dc_id	device_type	co2_level	averageCo2Level
1	dc-103	sensor-istick	▸[1819, 1705, 1658, 1753, 1616, 1871]	1737
2	dc-103	sensor-ipad	▸[1617, 1607, 1835, 1783, 1726, 1568]	1689
3	dc-103	sensor-inest	▸[1595, 1684, 1682, 1631, 1688, 1754]	1672
4	dc-103	sensor-inest	▸[1633, 1651, 1753, 1913, 1526, 1552]	1671
5	dc-103	sensor-inest	▸[1633, 1708, 1550, 1649, 1829,	1660

PIVOT TABLE

Normalmente utilizada para transformar linhas em colunas e fazer o agrupamento pelo campo especificado.

```
SELECT * FROM (
  SELECT device_type, averageCo2Level
  FROM Co2LevelsTemporary
)
PIVOT (
```

```
ROUND(AVG(averageCo2Level), 2) avg_co2
FOR device_type IN ('sensor-ipad', 'sensor-inest',
  'sensor-istick', 'sensor-igauge')
);
```

```
1  SELECT * FROM (
2    SELECT device_type, averageCo2Level
3    FROM Co2LevelsTemporary
4  )
5  PIVOT (
6    ROUND(AVG(averageCo2Level), 2) avg_co2
7    FOR device_type IN ('sensor-ipad', 'sensor-inest',
8      'sensor-istick', 'sensor-igauge')
9    );
```

▸ (3) Spark Jobs

	sensor-ipad	sensor-inest	sensor-istick	sensor-igauge
1	1245.98	1250.41	1244.86	1247.56

GROUP BY + ROLLUPS

Os dados serão sumarizados com base nos campos passados no parâmetro. Novas linhas serão criadas contendo subtotais.

```
SELECT
  COALESCE(dc_id, "All data centers") AS dc_id,
  COALESCE(device_type, "All devices") AS device_type,
  ROUND(AVG(averageCo2Level))  AS avgCo2Level
FROM Co2LevelsTemporary
GROUP BY ROLLUP (dc_id, device_type)
ORDER BY dc_id, device_type;
```

```
SELECT
  COALESCE(dc_id, "All data centers") AS dc_id,
  COALESCE(device_type, "All devices") AS device_type,
  ROUND(AVG(averageCo2Level))  AS avgCo2Level
FROM Co2LevelsTemporary
GROUP BY ROLLUP (dc_id, device_type)
ORDER BY dc_id, device_type;
```

▸ (2) Spark Jobs

	dc_id	device_type	avgCo2Level
1	All data centers	All devices	1247
2	dc-101	All devices	1197
3	dc-101	sensor-igauge	1202
4	dc-101	sensor-inest	1197
5	dc-101	sensor-ipad	1194
6	dc-101	sensor-istick	1196
7	dc-102	All devices	1296
8	dc-102	sensor-igauge	1303

Showing all 21 rows.

GROUP BY + CUBE

O subtotal será gerado para todas as combinações possíveis dos campos informados.

```
SELECT
  COALESCE(dc_id, "All data centers") AS dc_id,
  COALESCE(device_type, "All devices") AS device_type,
  ROUND(AVG(averageCo2Level))  AS avgCo2Level
FROM Co2LevelsTemporary
GROUP BY CUBE (dc_id, device_type)
ORDER BY dc_id, device_type;
```

CREATE WIDGET

Permite a entrada de parâmetros nas queries, sendo possível por exemplo conectar estes parâmetros com inputs do usuário.

```
CREATE WIDGET DROPDOWN selectedDeviceType DEFAULT "sensor-inest" CHOICES
SELECT
  DISTINCT device_type
FROM
  DeviceData
```

USE WIDGET

```
SELECT
  device_type,
  ROUND(AVG(avg_daily_temp_c),4) AS avgTemp,
  ROUND(STD(avg_daily_temp_c), 2) AS stdTemp
FROM AvgTemps
WHERE device_type = getArgument("selectedDeviceType")
GROUP BY device_type
```

DELETE WIDGET

```
REMOVE WIDGET selectedDeviceType
```

WINDOW FUNCTION

Window functions calculate a return variable for every input row of a table based on a group of rows selected by the user, the frame.

```
SELECT
  dc_id,
  month(date),
  avg_daily_temp_c,
  AVG(avg_daily_temp_c)
  OVER (PARTITION BY month(date), dc_id) AS avg_monthly_temp_c
```

```
FROM AvgTemps
WHERE month(date)="8" AND dc_id = "dc-102";
```

WINDOW FUNCTION + CTE

```
WITH DiffChart AS
(
SELECT
  dc_id,
  date,
  avg_daily_temp_c,
  AVG(avg_daily_temp_c)
  OVER (PARTITION BY month(date), dc_id) AS avg_monthly_temp_c
FROM AvgTemps
)
SELECT
  dc_id,
  date,
  avg_daily_temp_c,
  avg_monthly_temp_c,
  avg_daily_temp_c - ROUND(avg_monthly_temp_c) AS degree_diff
FROM DiffChart;
```