Name: Chandra Kiran Mudiraj

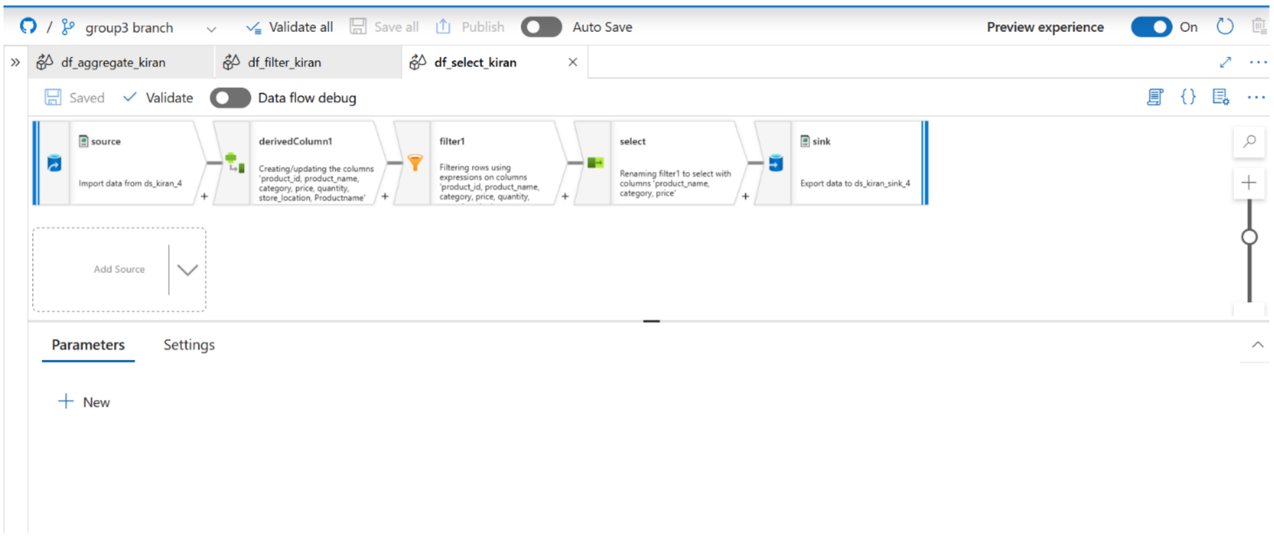
Mobile: 9490993762

Email: 2150897@cognizant.com

Requirement 1:

- **Use Case**: Select only certain columns from the dataset (e.g., exclude unnecessary columns).
- **Example**:
  - Select only the product_name, category, and price columns from the products.csv file.

Architecture:


source
Import data from ds_kiran_4

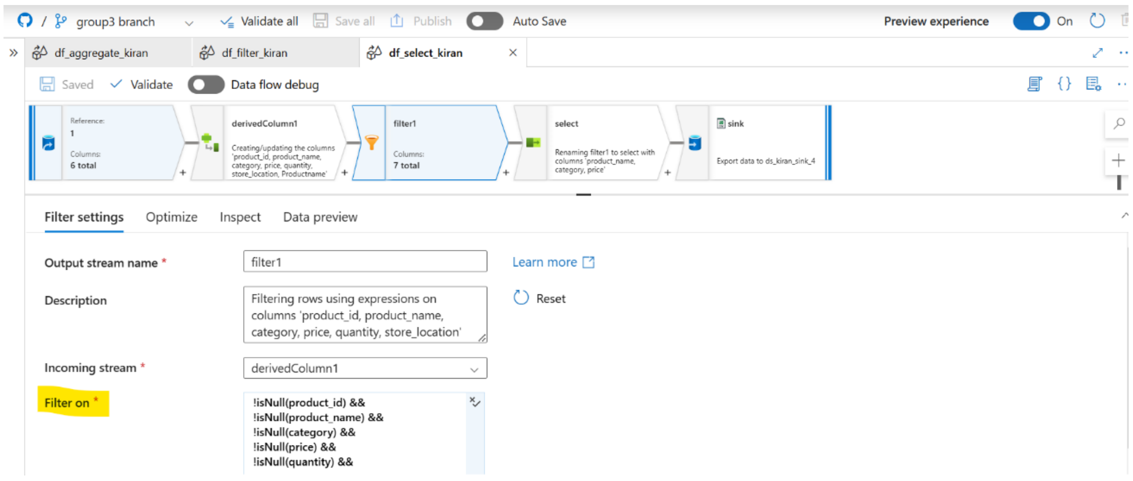
derivedColumn1
Creating/updating the columns 'product_id, product_name, category, price, quantity, store_location, Productname'

filter1
Filtering rows using expressions on columns 'product_id, product_name, category, price, quantity,

select
Renaming filter1 to select with columns 'product_name, category, price'

sink
Export data to ds_kiran_sink_4

Logic:



Output stream name * filter1

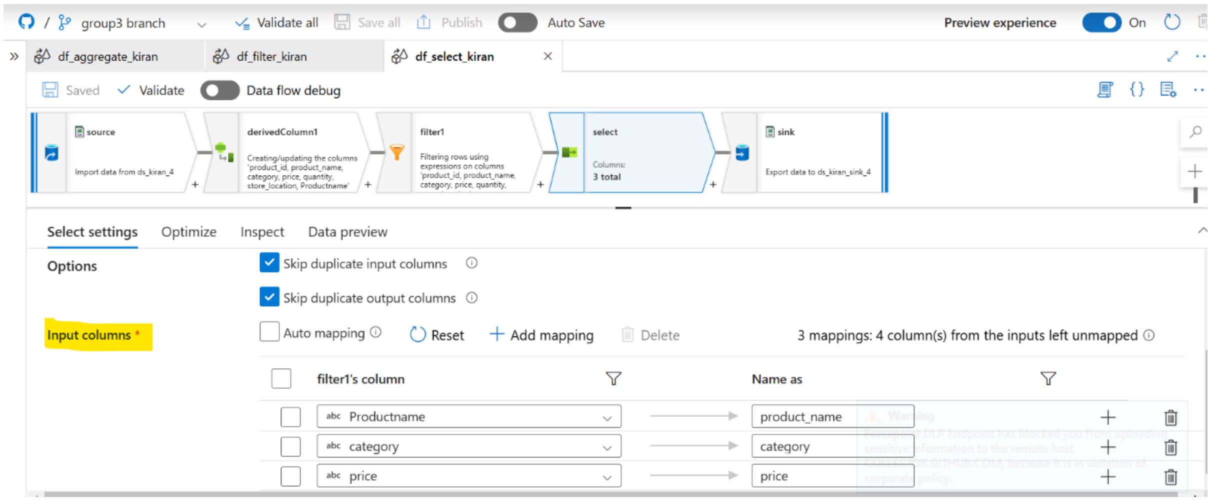
Description: Filtering rows using expressions on columns 'product_id, product_name, category, price, quantity, store_location'

Incoming stream * derivedColumn1

Filter on *

```
!isNull(product_id) &&
!isNull(product_name) &&
!isNull(category) &&
!isNull(price) &&
!isNull(quantity) &&
```

Selecting required columns(product_name, category, and price):


group3 branch
Validate all
Save all
Publish
Auto Save
Preview experience
On
df_aggregate_kiran
df_filter_kiran
df_select_kiran
Saved
Validate
Data flow debug
source
Import data from ds_kiran_4
derivedColumn1
Creating/updating the columns 'product_id, product_name, category, price, quantity, store_location, Productname'
filter1
Filtering rows using expressions on columns 'product_id, product_name, category, price, quantity,
select
Columns: 3 total
sink
Export data to ds_kiran_sink_4
Select settings
Optimize
Inspect
Data preview
Reset
Add mapping
Delete

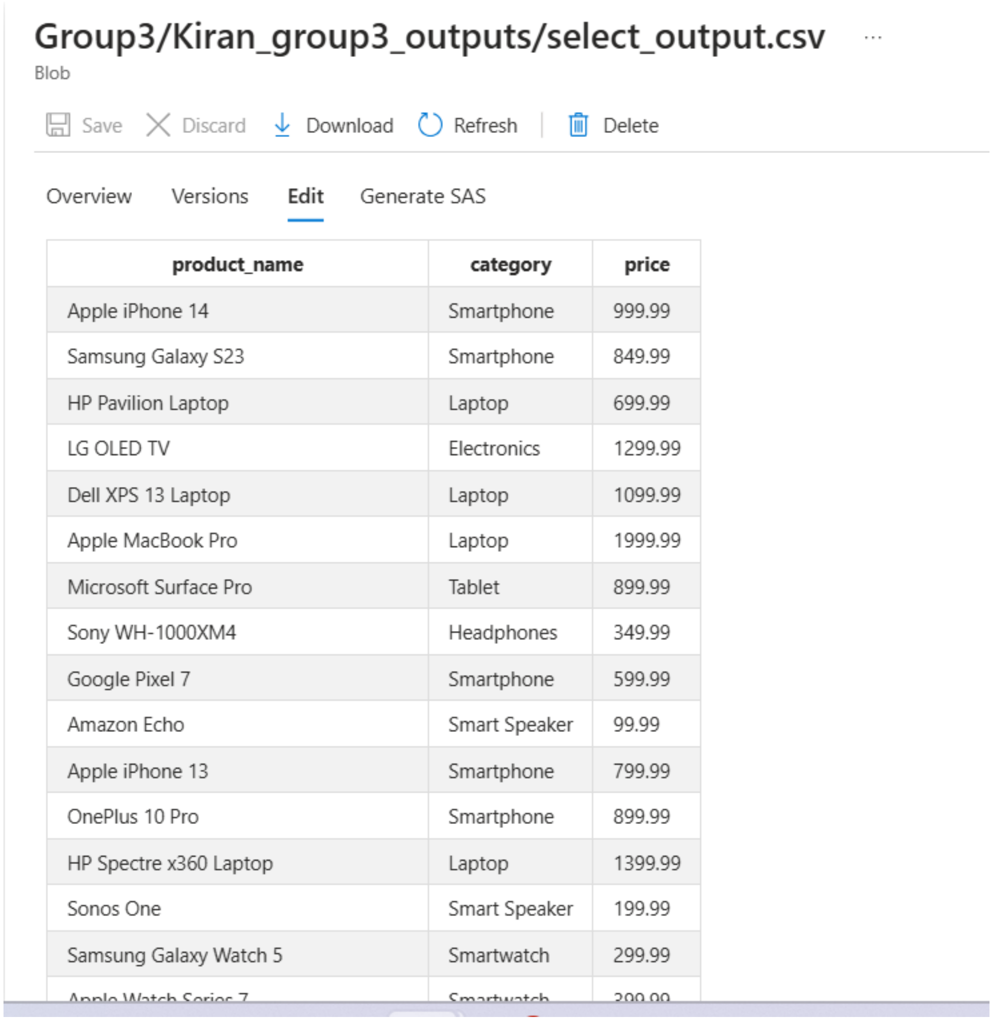
Options
- [x] Skip duplicate input columns
- [x] Skip duplicate output columns

Input columns *
- [ ] Auto mapping

3 mappings: 4 column(s) from the inputs left unmapped

filter1's column	Name as
Productname	product_name
category	category
price	price

Outcomes:

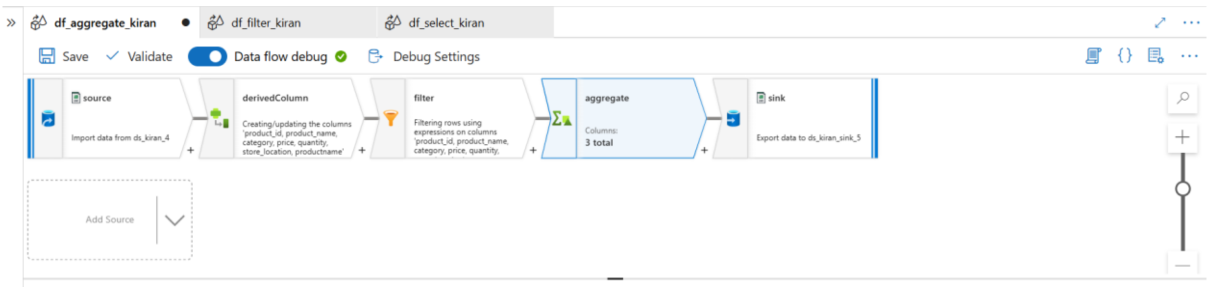


product_name	category	price
Apple iPhone 14	Smartphone	999.99
Samsung Galaxy S23	Smartphone	849.99
HP Pavilion Laptop	Laptop	699.99
LG OLED TV	Electronics	1299.99
Dell XPS 13 Laptop	Laptop	1099.99
Apple MacBook Pro	Laptop	1999.99
Microsoft Surface Pro	Tablet	899.99
Sony WH-1000XM4	Headphones	349.99
Google Pixel 7	Smartphone	599.99
Amazon Echo	Smart Speaker	99.99
Apple iPhone 13	Smartphone	799.99
OnePlus 10 Pro	Smartphone	899.99
HP Spectre x360 Laptop	Laptop	1399.99
Sonos One	Smart Speaker	199.99
Samsung Galaxy Watch 5	Smartwatch	299.99
Apple Watch Series 7	Smartwatch	399.99

Requirement 2:

- **Use Case**: Perform aggregate operations like summing, counting, or averaging.
- **Example**:
  - Calculate the total quantity of products available by category.
  - Calculate the average price of products in each category.

Architecture:



source
Import data from ds_kiran_4

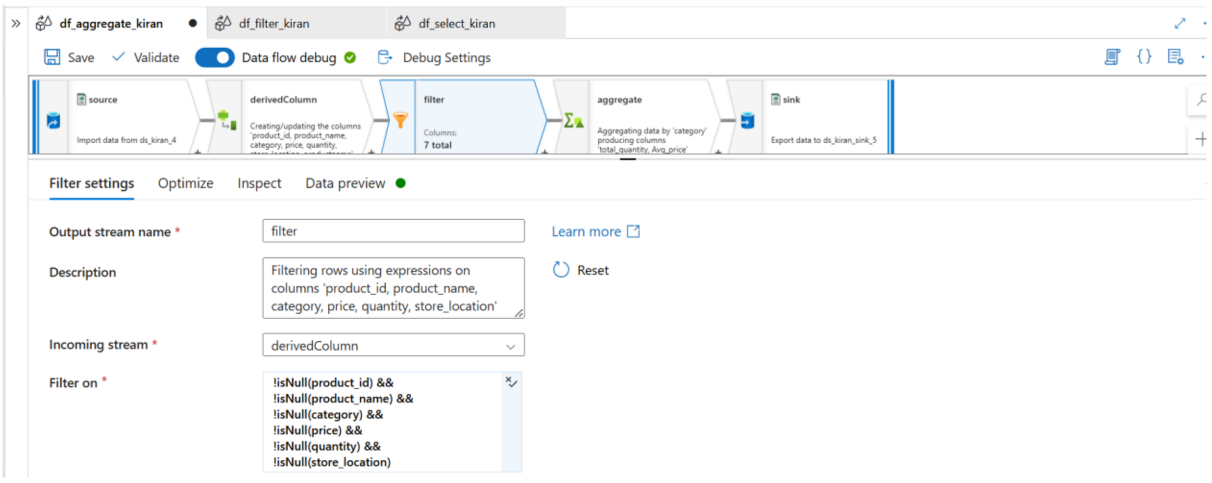
derivedColumn
Creating/updating the columns 'product_id, product_name, category, price, quantity, store_location, productname'

filter
Filtering rows using expressions on columns 'product_id, product_name, category, price, quantity,

aggregate
Columns:
3 total

sink
Export data to ds_kiran_sink_5

Logic:



source
Import data from ds_kiran_4

derivedColumn
Creating/updating the columns 'product_id, product_name, category, price, quantity,

filter
Columns:
7 total

aggregate
Aggregating data by 'category' producing columns 'total_quantity, Avg_price'

sink
Export data to ds_kiran_sink_5

Output stream name * filter

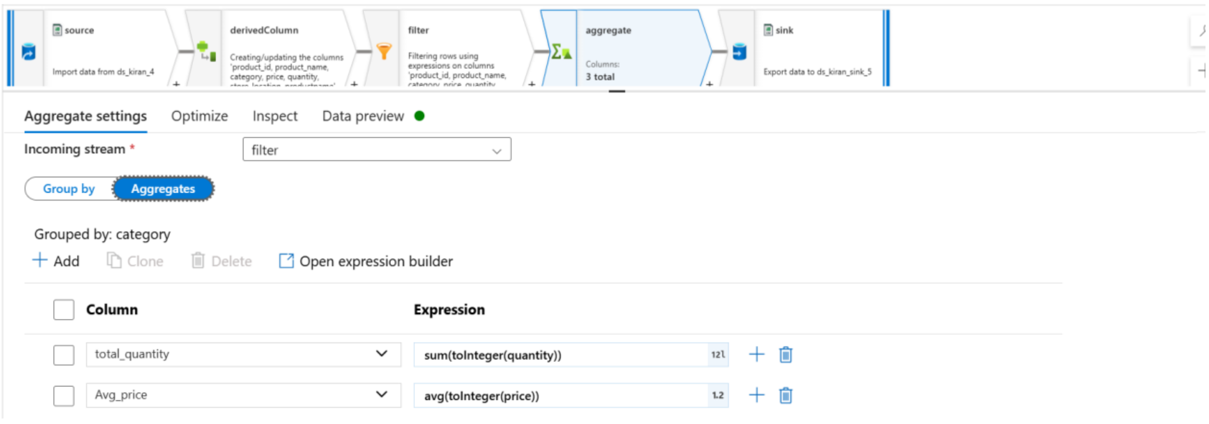
Description: Filtering rows using expressions on columns 'product_id, product_name, category, price, quantity, store_location'

Incoming stream * derivedColumn

Filter on *

```
!isNull(product_id) &&
!isNull(product_name) &&
!isNull(category) &&
!isNull(price) &&
!isNull(quantity) &&
!isNull(store_location)
```

To aggregate:

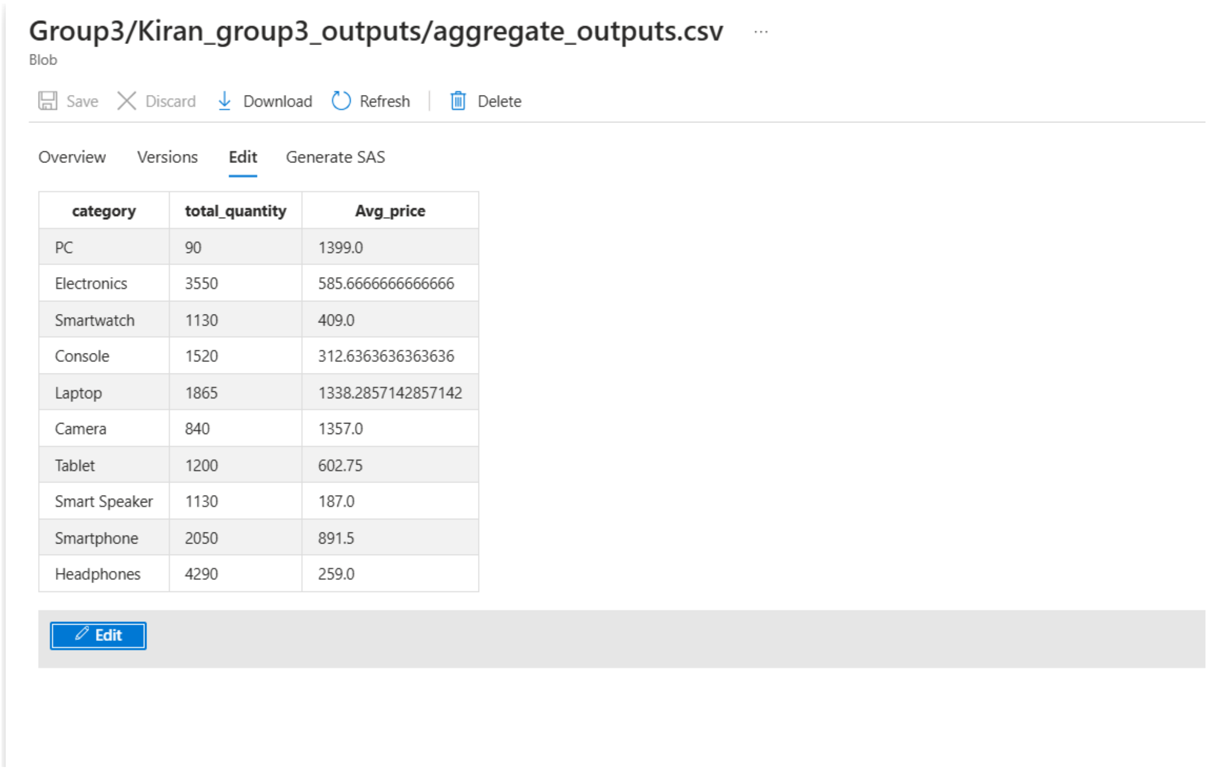

source
Import data from ds_kiran_4
derivedColumn
Creating/updating the columns 'product_id, product_name, category, price, quantity,
filter
Filtering rows using expressions on columns 'product_id, product_name,
aggregate
Columns:
3 total
sink
Export data to ds_kiran_sink_5
Aggregate settings
Optimize
Inspect
Data preview
Group by
Aggregates
Add
Clone
Delete
Open expression builder

Incoming stream * filter

Grouped by: category

Column	Expression
total_quantity	sum(toInteger(quantity))
Avg_price	avg(toInteger(price))

Outcomes:



Group3/Kiran_group3_outputs/aggregate_outputs.csv

Blob

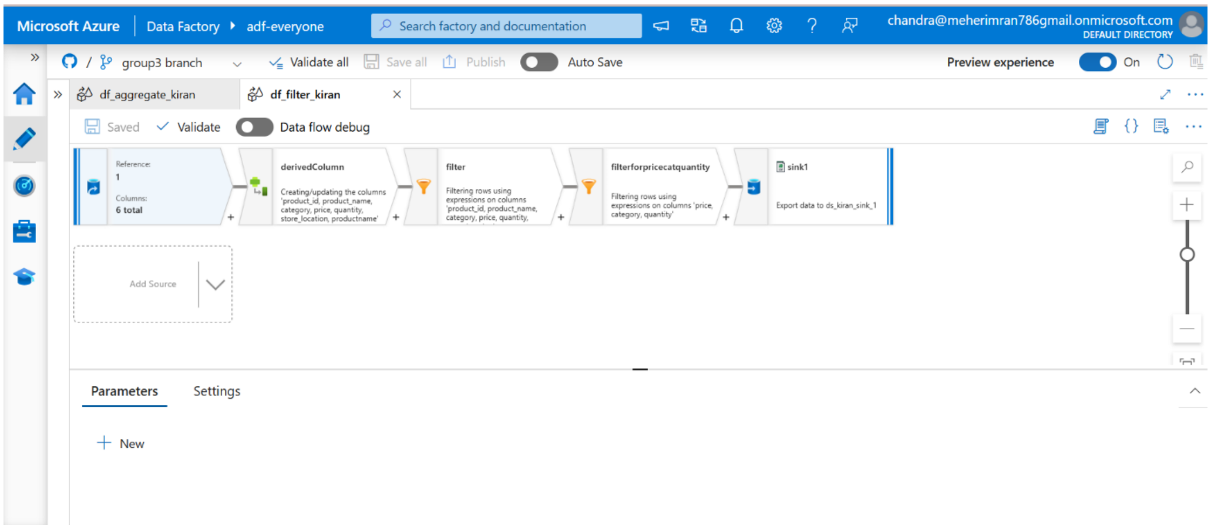
category	total_quantity	Avg_price
PC	90	1399.0
Electronics	3550	585.6666666666666
Smartwatch	1130	409.0
Console	1520	312.6363636363636
Laptop	1865	1338.2857142857142
Camera	840	1357.0
Tablet	1200	602.75
Smart Speaker	1130	187.0
Smartphone	2050	891.5
Headphones	4290	259.0

Requirement 3:

- **Use Case:** Filter products based on certain conditions like category, price range, or quantity.
- **Example**:

- Filter products where price > 1000.
- Filter products from the “Smartphone” category.
- Filter products that are available in stock (i.e., quantity > 0).

Architecture :



Reference: 1
Columns: 6 total

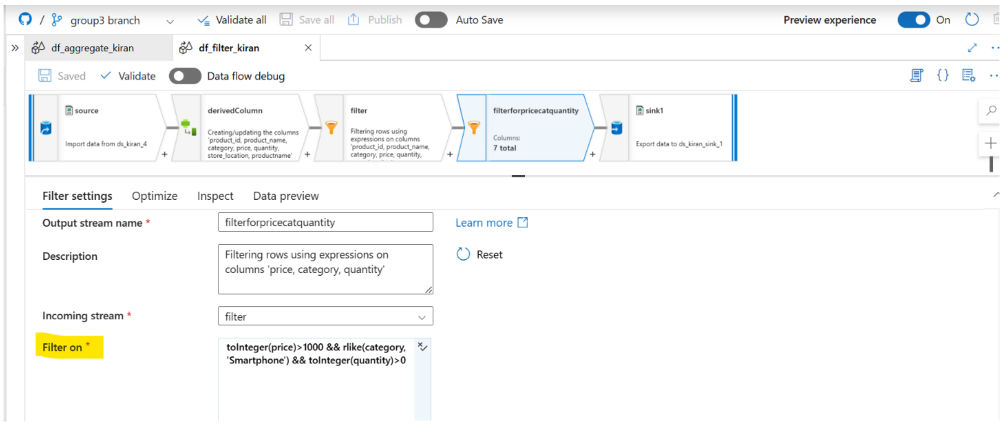
derivedColumn
Creating/updating the columns 'product_id, product_name, category, price, quantity, store_location, productname'

filter
Filtering rows using expressions on columns 'product_id, product_name, category, price, quantity,

filterforpricecatquantity
Filtering rows using expressions on columns 'price, category, quantity'

sink1
Export data to ds_kiran_sink_1

Logic:



source
Import data from ds_kiran_4

derivedColumn
Creating/updating the columns 'product_id, product_name, category, price, quantity, store_location, productname'

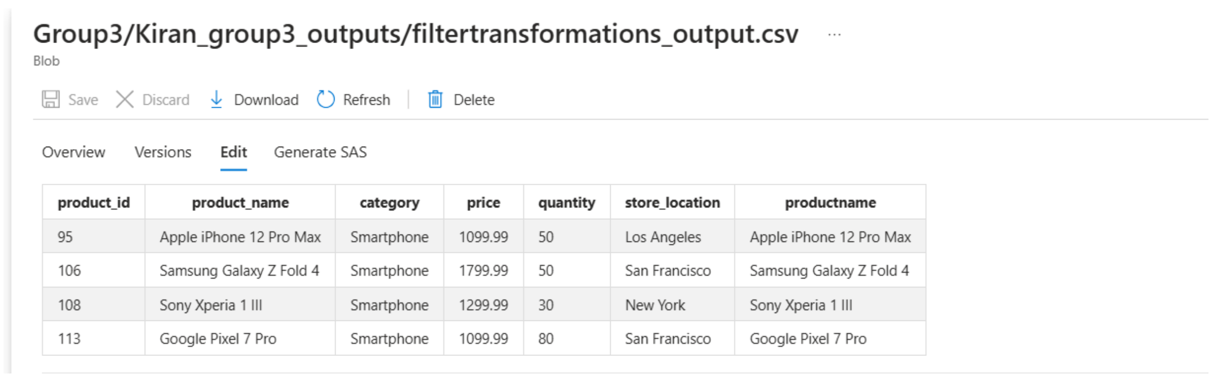
filter
Filtering rows using expressions on columns 'product_id, product_name, category, price, quantity,

filterforpricecatquantity
Columns: 7 total

sink1
Export data to ds_kiran_sink_1

Output stream name *	filterforpricecatquantity
Description	Filtering rows using expressions on columns 'price, category, quantity'
Incoming stream *	filter
Filter on *	toInteger(price)>1000 && rlike(category, 'Smartphone') && toInteger(quantity)>0

Outcomes:


Group3/Kiran_group3_outputs/filtertransformations_output.csv

Blob

product_id	product_name	category	price	quantity	store_location	productname
95	Apple iPhone 12 Pro Max	Smartphone	1099.99	50	Los Angeles	Apple iPhone 12 Pro Max
106	Samsung Galaxy Z Fold 4	Smartphone	1799.99	50	San Francisco	Samsung Galaxy Z Fold 4
108	Sony Xperia 1 III	Smartphone	1299.99	30	New York	Sony Xperia 1 III
113	Google Pixel 7 Pro	Smartphone	1099.99	80	San Francisco	Google Pixel 7 Pro