QUESTION 1

What is UBER mode ?

When a particular Hadoop job is invoked from client machine, RM will create separate container for that application. Uber configuration, will allow running that job in the same process as the Application Master (AM).

Uber jobs:

Uber jobs are jobs that are executed within the ApplicationMaster. It do not need to communicate with RM to create the containers on different worker nodes. As the job is given is evaluated small so the AM runs tasks within its own process and avoided the overhead of launching and communicate with remote containers located in different worker nodes can be considered as advantage in certain cases.

Why?

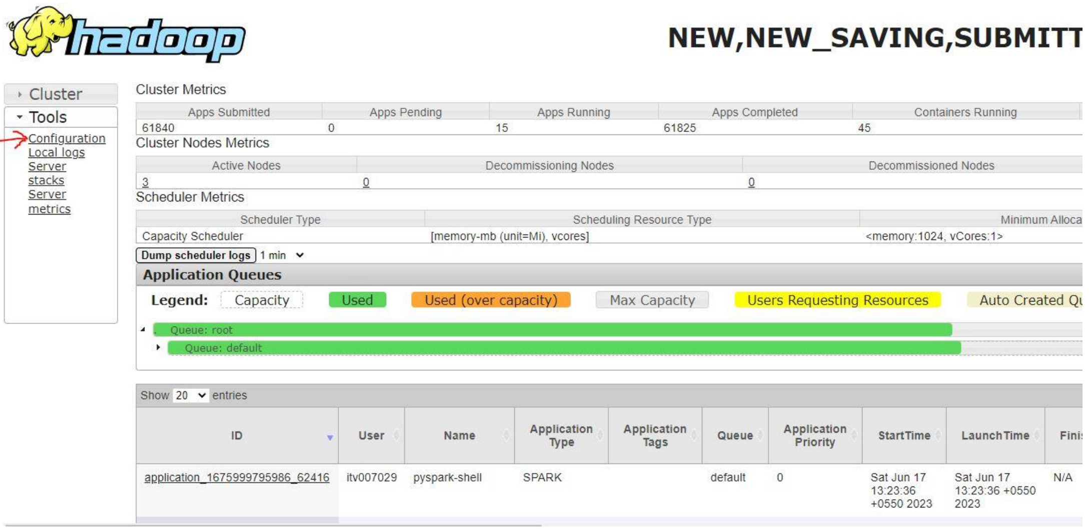
If you have a small dataset, Uber configuration will help you out, by reducing additional time required to complete the certain Hadoop job.

Configuration options for Uber Jobs

Property	**Description**
mapreduce.job.ubertask.enable	Whether to enable the small-jobs "ubertask" optimization, which runs "sufficiently small" jobs sequentially within a single JVM. "Small" is defined by the following maxmaps, maxreduces, and maxbytes settings. Users can override this value. Default = false
mapreduce.job.ubertask.maxmaps	The threshold for the number of maps beyond which a job is considered too large for the ubertasking optimization. Users can override this value, but only downward. Default = 9
mapreduce.job.ubertask.maxreduces	The threshold for the number of reduces beyond which a job is considered too large for the uber tasking optimization. CURRENTLY THE CODE CANNOT SUPPORT MORE THAN ONE REDUCE and will ignore larger values (zero is a valid maximum value, however). Users can override this value, but only downward. Default = 1
mapreduce.job.ubertask.maxbytes	The threshold for the number of input bytes beyond which a job is considered too large for the uber tasking optimization. If no value is specified, dfs.block.size is used as the default. Be sure to specify a default value in mapred-site.xml if the underlying file system is not HDFS. Users can override this value, but only downward. Default = HDFS Block Size

<u>SOP for accessing UBER mode in Resource Manager –</u>

1. Go to http://172.16.1.104:19088/ by enabling foxy proxy server.
2. Click on Configuration option under Tool tab on left side of screen.



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running
61840	0	15	61825	45

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes
3	0	0

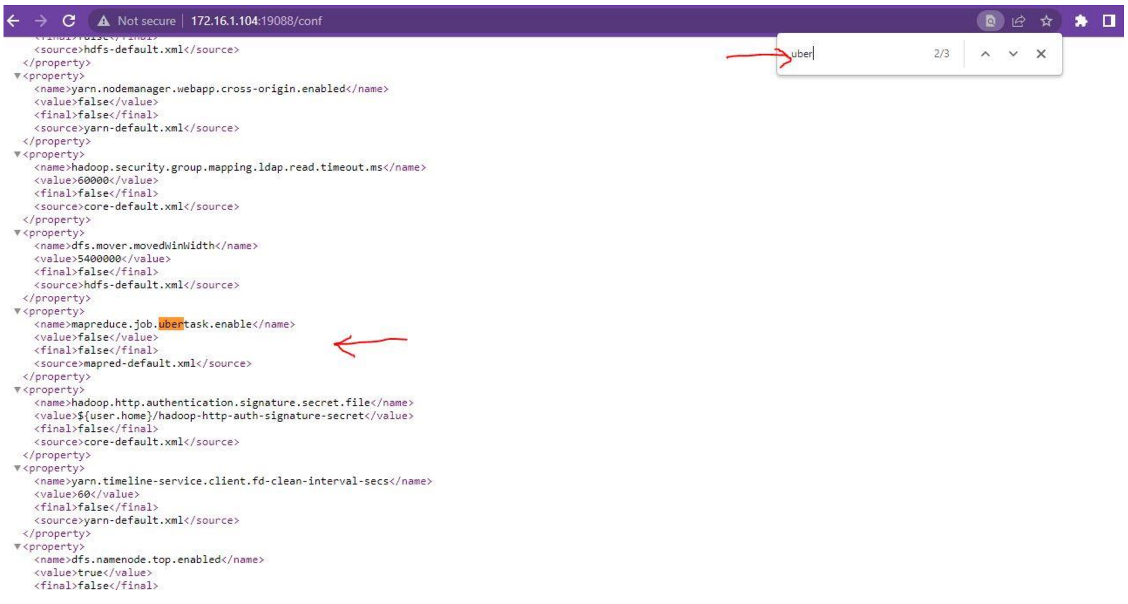
Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Alloca
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>

Application Queues

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	Fini
application_1675999795986_62416	itv007029	pyspark-shell	SPARK		default	0	Sat Jun 17 13:23:36 +0550 2023	Sat Jun 17 13:23:36 +0550 2023	N/A

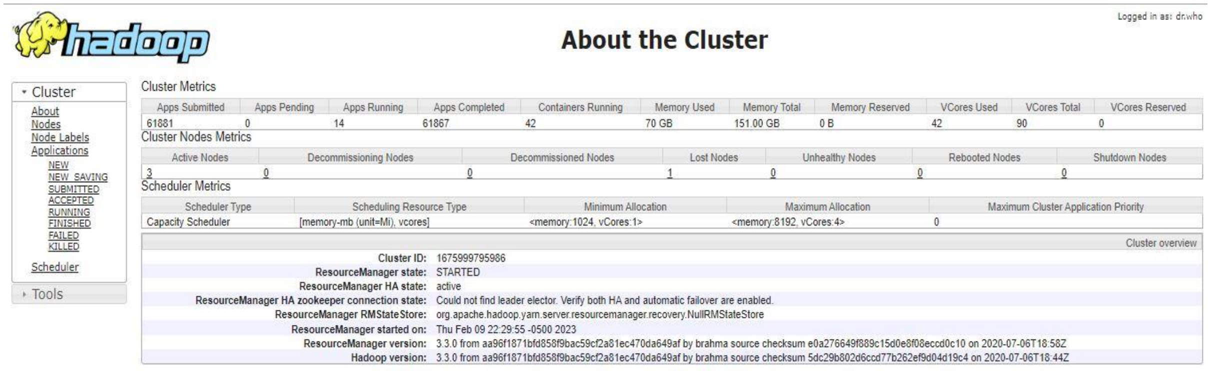
3. Find in page for the keyword "Uber". Setting will appear change as mentioned.



```
    <source>hdfs-default.xml</source>
  </property>
▼<property>
    <name>yarn.nodemanager.webapp.cross-origin.enabled</name>
    <value>false</value>
    <final>false</final>
    <source>yarn-default.xml</source>
  </property>
▼<property>
    <name>hadoop.security.group.mapping.ldap.read.timeout.ms</name>
    <value>60000</value>
    <final>false</final>
    <source>core-default.xml</source>
  </property>
▼<property>
    <name>dfs.mover.movedWinWidth</name>
    <value>5400000</value>
    <final>false</final>
    <source>hdfs-default.xml</source>
  </property>
▼<property>
    <name>mapreduce.job.ubertask.enable</name>
    <value>false</value>
    <final>false</final>
    <source>mapred-default.xml</source>
  </property>
▼<property>
    <name>hadoop.http.authentication.signature.secret.file</name>
    <value>${user.home}/hadoop-http-auth-signature-secret</value>
    <final>false</final>
    <source>core-default.xml</source>
  </property>
▼<property>
    <name>yarn.timeline-service.client.fd-clean-interval-secs</name>
    <value>60</value>
    <final>false</final>
    <source>yarn-default.xml</source>
  </property>
▼<property>
    <name>dfs.namenode.top.enabled</name>
    <value>true</value>
    <final>false</final>
```

QUESTION 2

ABOUT - Gives the information about the cluster and it's metrics eg – memory used, total memory, v cores etc. Other information like Cluster Node Metrics and Scheduler Metrics (min and max vcore allocation info). We can also find certain details related to commissioning of Resource Manager.



About the Cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
61881	0	14	61867	42	70 GB	151.00 GB	0 B	42	90	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
3	0	0	1	0	0	0

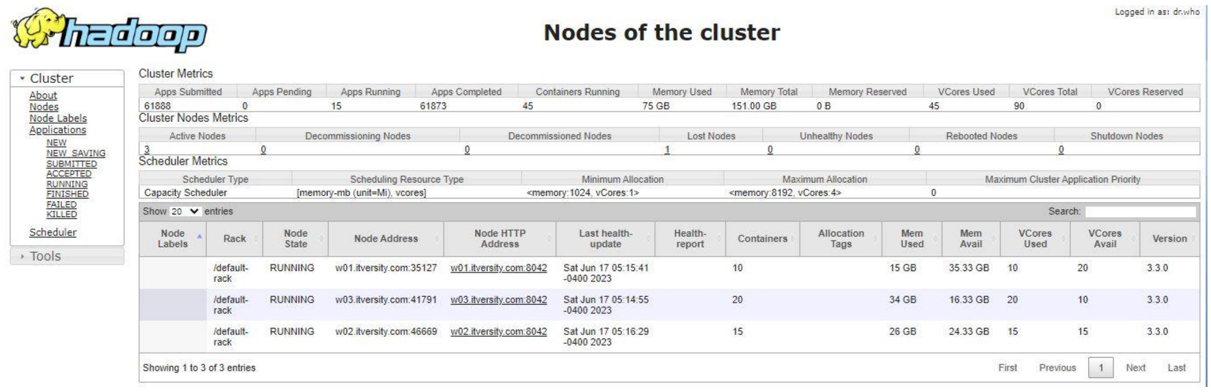
Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Cluster overview

Cluster ID:	1675999795986
ResourceManager state:	STARTED
ResourceManager HA state:	active
ResourceManager HA zookeeper connection state:	Could not find leader elector. Verify both HA and automatic failover are enabled.
ResourceManager RMStateStore:	org.apache.hadoop.yarn.server.resourcemanager.recovery.NullRMStateStore
ResourceManager started on:	Thu Feb 09 22:29:55 -0500 2023
ResourceManager version:	3.3.0 from aa96f1871bfd858f9bac59cf2a81ec470da649af by brahma source checksum e0a276649f889c15d0e8f08eccd0c10 on 2020-07-06T18:58Z
Hadoop version:	3.3.0 from aa96f1871bfd858f9bac59cf2a81ec470da649af by brahma source checksum 5dc29b802d6ccd77b262ef9d04d19c4 on 2020-07-06T18:44Z

Nodes – Shows information about 3 worker nodes present on this cluster.



Nodes of the cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
61888	0	15	61873	45	75 GB	151.00 GB	0 B	45	90	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
3	0	0	1	0	0	0

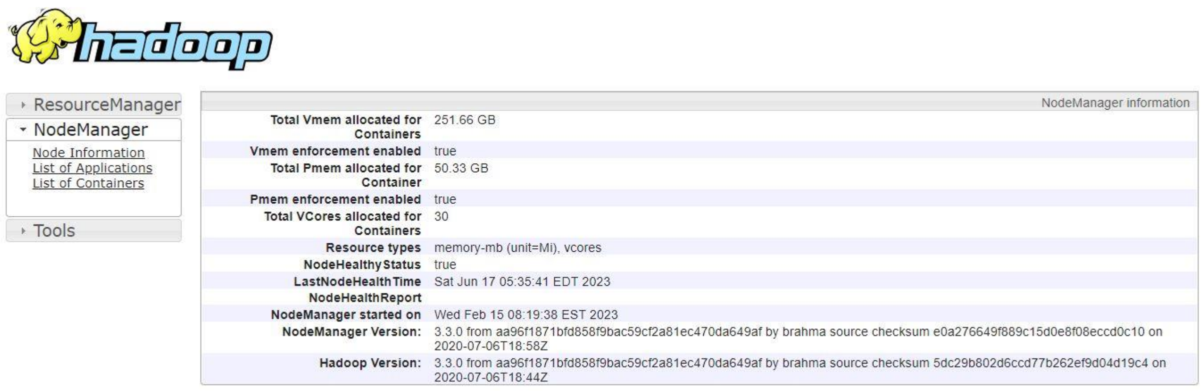
Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
	/default-rack	RUNNING	w01.itversity.com:35127	w01.itversity.com:8042	Sat Jun 17 05:15:41 -0400 2023		10		15 GB	35.33 GB	10	20	3.3.0
	/default-rack	RUNNING	w03.itversity.com:41791	w03.itversity.com:8042	Sat Jun 17 05:14:55 -0400 2023		20		34 GB	16.33 GB	20	10	3.3.0
	/default-rack	RUNNING	w02.itversity.com:46669	w02.itversity.com:8042	Sat Jun 17 05:16:29 -0400 2023		15		26 GB	24.33 GB	15	15	3.3.0

Showing 1 to 3 of 3 entries

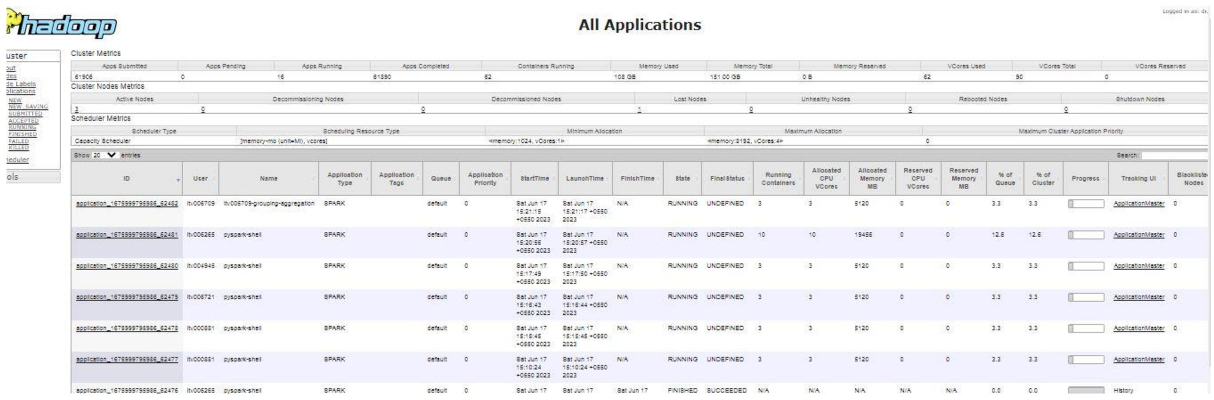
By clicking on link provided under Node HTTP Address of particular worker node we can get info about that particular node and as well **node manager** information.



NodeManager information

Total Vmem allocated for Containers	251.66 GB
Vmem enforcement enabled	true
Total Pmem allocated for Container	50.33 GB
Pmem enforcement enabled	true
Total VCores allocated for Containers	30
Resource types	memory-mb (unit=Mi), vcores
NodeHealthyStatus	true
LastNodeHealthTime	Sat Jun 17 05:35:41 EDT 2023
NodeHealthReport	
NodeManager started on	Wed Feb 15 08:19:38 EST 2023
NodeManager Version:	3.3.0 from aa96f1871bfd858f9bac59cf2a81ec470da649af by brahma source checksum e0a276649f889c15d0e8f08eccd0c10 on 2020-07-06T18:58Z
Hadoop Version:	3.3.0 from aa96f1871bfd858f9bac59cf2a81ec470da649af by brahma source checksum 5dc29b802d6ccd77b262ef9d04d19c4 on 2020-07-06T18:44Z

Applications – Shows all the links and key information about all the application present on this cluster, where different tab under applications can show particular job which are running, finished or killed etc.



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
61906	0	16	61890	62	108 GB	151.00 GB	0 B	62	90	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
3	0	0	1	0	0	0

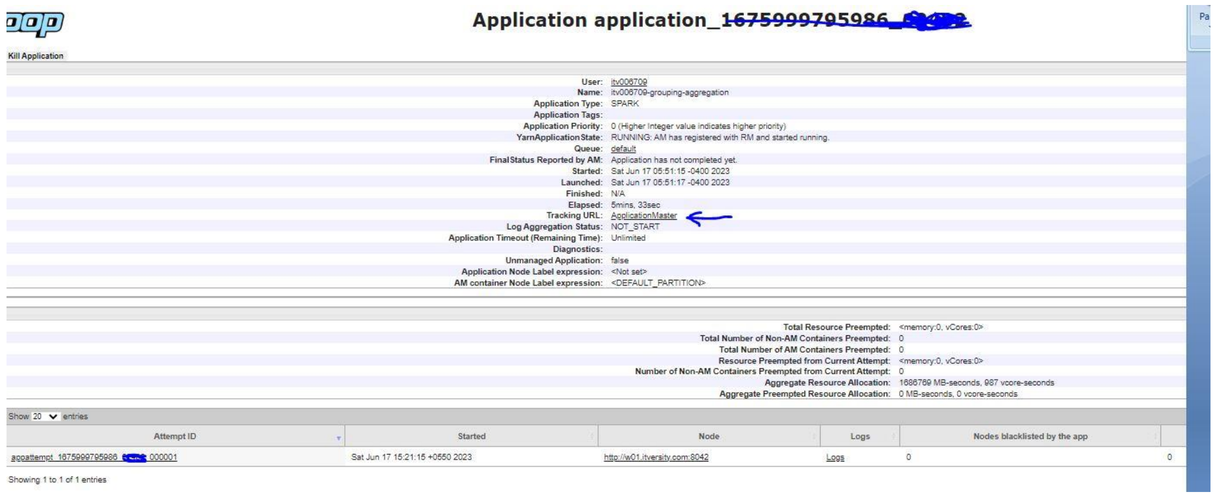
Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries Search:

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1675999795986_62482	itv006709	itv006709-grouping-aggregation	SPARK		default	0	Sat Jun 17 15:21:15 +0550 2023	Sat Jun 17 15:21:17 +0550 2023	N/A	RUNNING	UNDEFINED	3	3	5120	0	0	3.3	3.3		ApplicationMaster	0
application_1675999795986_62481	itv006265	pyspark-shell	SPARK		default	0	Sat Jun 17 15:20:56 +0550 2023	Sat Jun 17 15:20:57 +0550 2023	N/A	RUNNING	UNDEFINED	10	10	19456	0	0	12.6	12.6		ApplicationMaster	0
application_1675999795986_62480	itv004945	pyspark-shell	SPARK		default	0	Sat Jun 17 15:17:49 +0550 2023	Sat Jun 17 15:17:50 +0550 2023	N/A	RUNNING	UNDEFINED	3	3	5120	0	0	3.3	3.3		ApplicationMaster	0
application_1675999795986_62479	itv006721	pyspark-shell	SPARK		default	0	Sat Jun 17 15:16:43 +0550 2023	Sat Jun 17 15:16:44 +0550 2023	N/A	RUNNING	UNDEFINED	3	3	5120	0	0	3.3	3.3		ApplicationMaster	0
application_1675999795986_62478	itv000881	pyspark-shell	SPARK		default	0	Sat Jun 17 15:15:45 +0550 2023	Sat Jun 17 15:15:45 +0550 2023	N/A	RUNNING	UNDEFINED	3	3	5120	0	0	3.3	3.3		ApplicationMaster	0
application_1675999795986_62477	itv000881	pyspark-shell	SPARK		default	0	Sat Jun 17 15:10:24 +0550 2023	Sat Jun 17 15:10:24 +0550 2023	N/A	RUNNING	UNDEFINED	3	3	5120	0	0	3.3	3.3		ApplicationMaster	0
application_1675999795986_62476	itv006265	pyspark-shell	SPARK		default	0	Sat Jun 17	Sat Jun 17	Sat Jun 17	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0

Link under the ID of particular running application will lead to that particular application information in detailed view.



Application application_1675999795986_[redacted]

Kill Application

User:	itv006709
Name:	itv006709-grouping-aggregation
Application Type:	SPARK
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	RUNNING: AM has registered with RM and started running.
Queue:	default
FinalStatus Reported by AM:	Application has not completed yet.
Started:	Sat Jun 17 05:51:15 -0400 2023
Launched:	Sat Jun 17 05:51:17 -0400 2023
Finished:	N/A
Elapsed:	5mins, 33sec
Tracking URL:	ApplicationMaster
Log Aggregation Status:	NOT_START
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

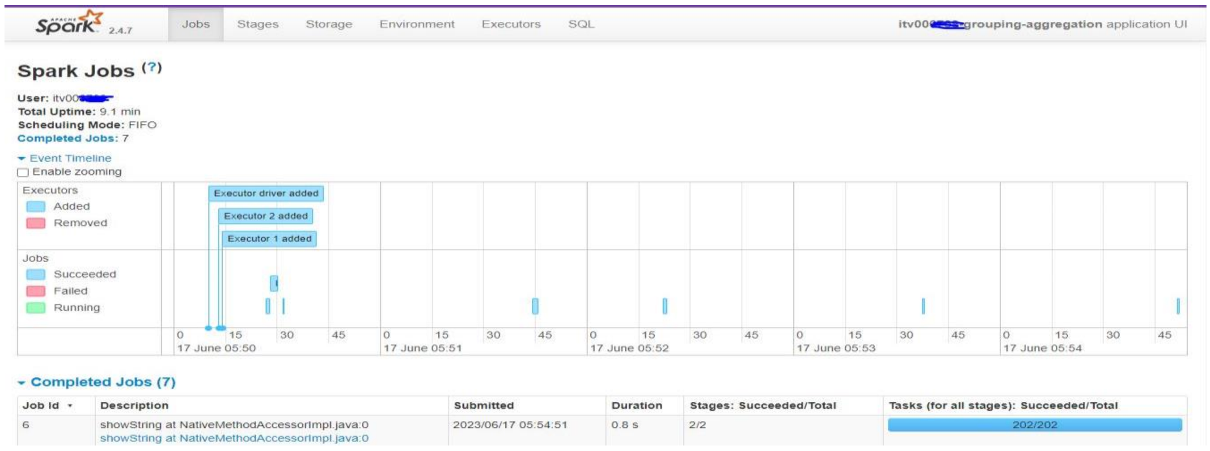
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	1686769 MB-seconds, 987 vcore-seconds
Aggregate Preempted Resource Allocation:	0 MB-seconds, 0 vcore-seconds

Show 20 entries

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	
appattempt_1675999795986_[redacted]_000001	Sat Jun 17 15:21:15 +0550 2023	http://w01.itversity.com:8042	Logs	0	0

Showing 1 to 1 of 1 entries

By clicking on link under Tracking URL named as Application Master will lead us to **Spark UI** of that particular job and we can navigate through various information about the job in that particular application.



itv00[redacted]grouping-aggregation application UI

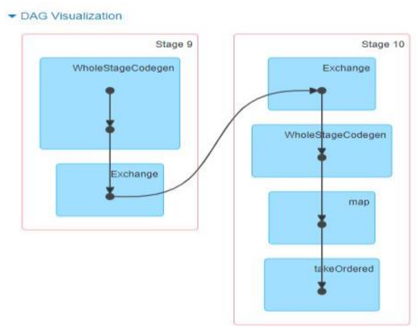
Spark Jobs (?)

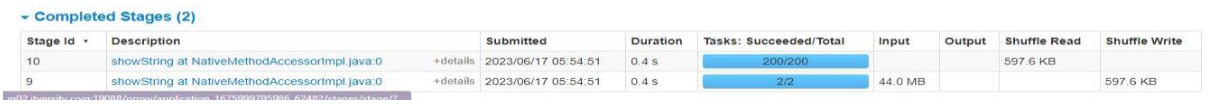
User: itv00[redacted]
Total Uptime: 9.1 min
Scheduling Mode: FIFO
Completed Jobs: 7

Completed Jobs (7)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
6	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2023/06/17 05:54:51	0.8 s	2/2	202/202

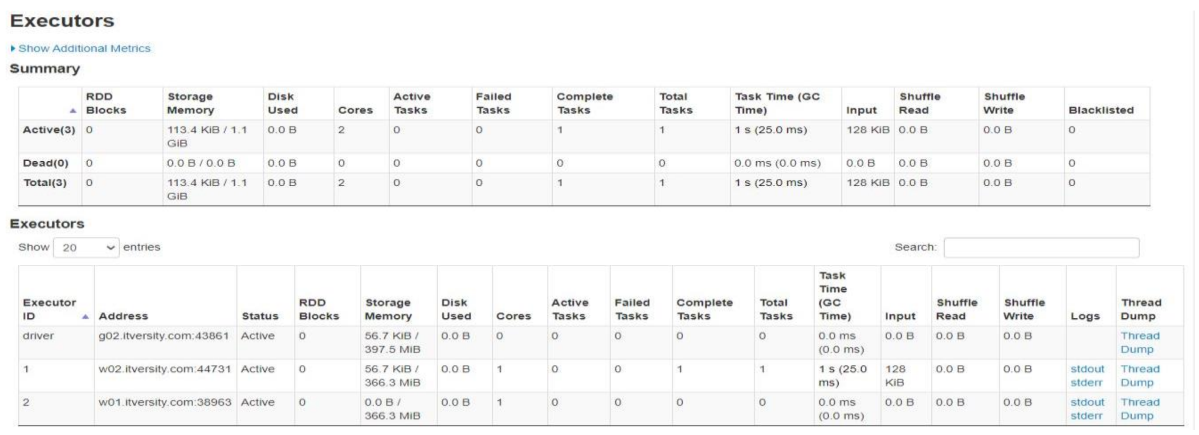
Link under Description section will lead us to DAG Visualization and stages information. **Stages tab** show completed stages. Storage tab show the cached results. **Environment tab** gives us information about spark properties, system properties etc. **Executors tab** show the information about information about driver and executors involved for this application in order to complete a job. When clicked on **SQL tab** shows the completed queries and when clicked in description details of the query with physical, optimized, analyzed and parsed logical plan.





Completed Stages (2)

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
10	showString at NativeMethodAccessorImpl.java:0	+details	2023/06/17 05:54:51	0.4 s	200/200			597.6 KB	
9	showString at NativeMethodAccessorImpl.java:0	+details	2023/06/17 05:54:51	0.4 s	2/2	44.0 MB			597.6 KB



Executors

Show Additional Metrics

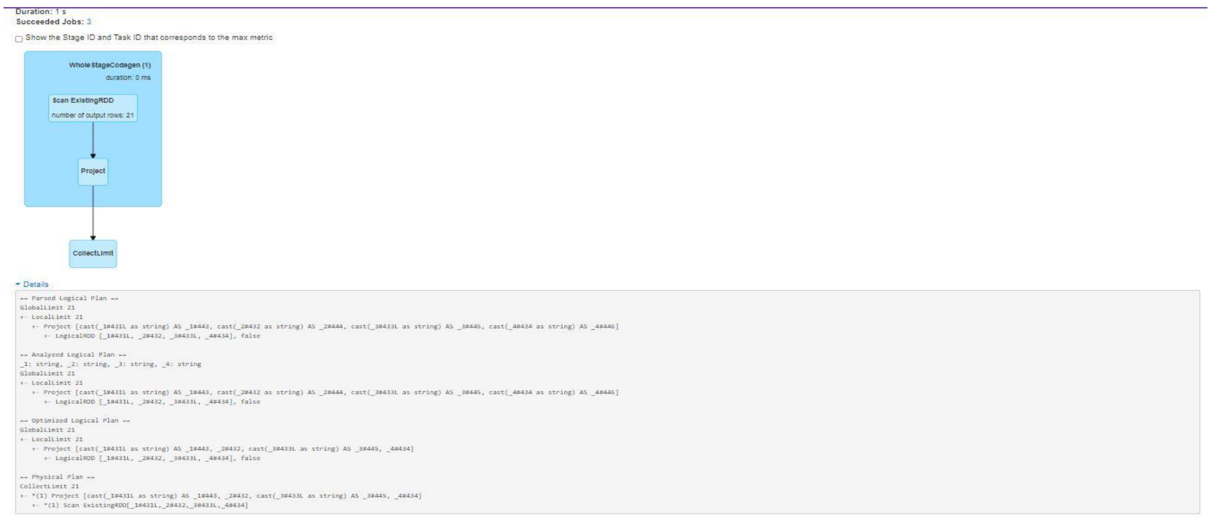
Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Blacklisted
Active(3)	0	113.4 KiB / 1.1 GiB	0.0 B	2	0	0	1	1	1 s (25.0 ms)	128 KiB	0.0 B	0.0 B	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Total(3)	0	113.4 KiB / 1.1 GiB	0.0 B	2	0	0	1	1	1 s (25.0 ms)	128 KiB	0.0 B	0.0 B	0

Executors

Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
driver	g02.itversity.com:43861	Active	0	56.7 KiB / 397.5 MiB	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B		Thread Dump
1	w02.itversity.com:44731	Active	0	56.7 KiB / 366.3 MiB	0.0 B	1	0	0	1	1	1 s (25.0 ms)	128 KiB	0.0 B	0.0 B	stdout stderr	Thread Dump
2	w01.itversity.com:38963	Active	0	0.0 B / 366.3 MiB	0.0 B	1	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump



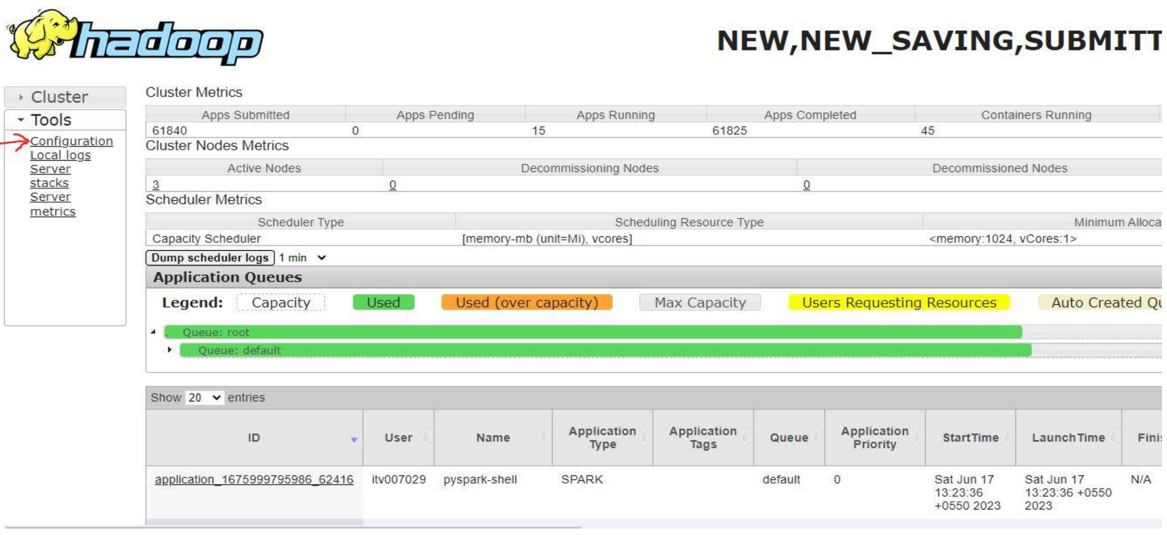
Duration: 1 s
Succeeded Jobs: 3

☐ Show the Stage ID and Task ID that corresponds to the max metric

Details

```
== Parsed Logical Plan ==
GlobalLimit 21
+- LocalLimit 21
   +- Project [cast(_1#431L as string) AS _1#443, cast(_2#432 as string) AS _2#444, cast(_3#433L as string) AS _3#445, cast(_4#434 as string) AS _4#446]
      +- LogicalRDD [_1#431L, _2#432, _3#433L, _4#434], false

== Analyzed Logical Plan ==
_1: string, _2: string, _3: string, _4: string
GlobalLimit 21
+- LocalLimit 21
   +- Project [cast(_1#431L as string) AS _1#443, cast(_2#432 as string) AS _2#444, cast(_3#433L as string) AS _3#445, cast(_4#434 as string) AS _4#446]
      +- LogicalRDD [_1#431L, _2#432, _3#433L, _4#434], false

== Optimized Logical Plan ==
GlobalLimit 21
+- LocalLimit 21
   +- Project [cast(_1#431L as string) AS _1#443, _2#432, cast(_3#433L as string) AS _3#445, _4#434]
      +- LogicalRDD [_1#431L, _2#432, _3#433L, _4#434], false

== Physical Plan ==
CollectLimit 21
+- *(1) Project [cast(_1#431L as string) AS _1#443, _2#432, cast(_3#433L as string) AS _3#445, _4#434]
   +- *(1) Scan ExistingRDD[_1#431L,_2#432,_3#433L,_4#434]
```

TOOLS – under tools section we can open configuration tab in order to edit predefined configurations and property related to resource manager e.g. – changing uber mode configuration.



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running
61840	0	15	61825	45

Cluster Nodes Metrics

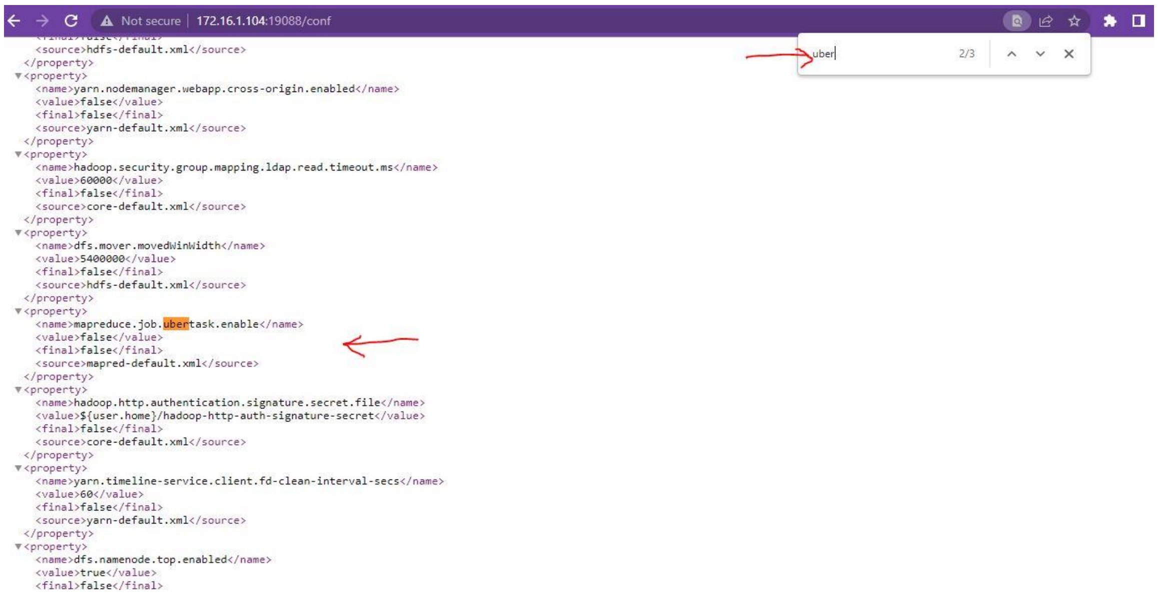
Active Nodes	Decommissioning Nodes	Decommissioned Nodes
3	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Alloca
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>

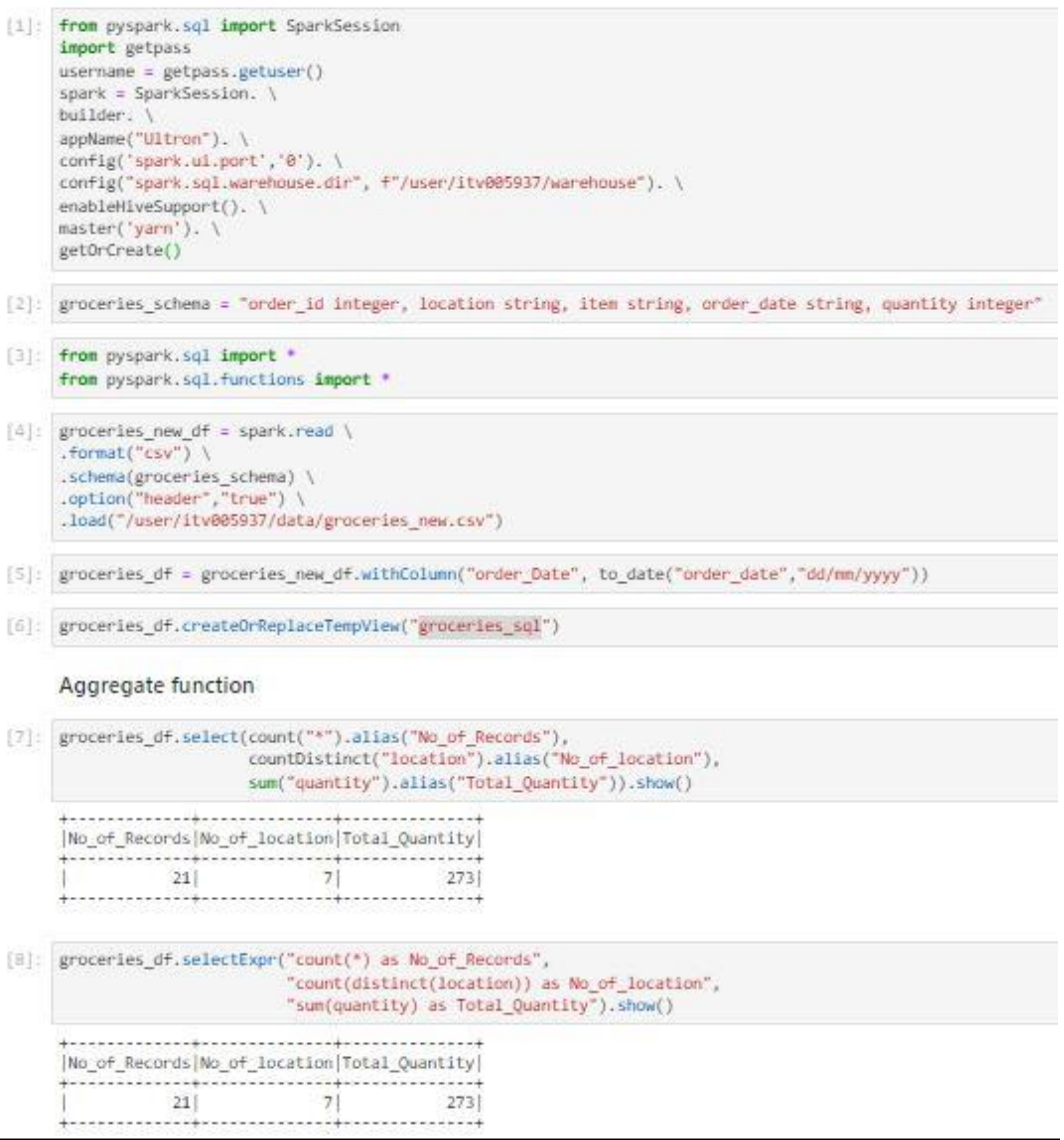
Dump scheduler logs
1 min
Application Queues
Legend: Capacity Used Used (over capacity) Max Capacity Users Requesting Resources Auto Created Qu
Queue: root
Queue: default
Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	Fini
application_1675999795986_62416	itv007029	pyspark-shell	SPARK		default	0	Sat Jun 17 13:23:36 +0550 2023	Sat Jun 17 13:23:36 +0550 2023	N/A



```
    <source>hdfs-default.xml</source>
  </property>
▼<property>
    <name>yarn.nodemanager.webapp.cross-origin.enabled</name>
    <value>false</value>
    <final>false</final>
    <source>yarn-default.xml</source>
  </property>
▼<property>
    <name>hadoop.security.group.mapping.ldap.read.timeout.ms</name>
    <value>60000</value>
    <final>false</final>
    <source>core-default.xml</source>
  </property>
▼<property>
    <name>dfs.mover.movedWinWidth</name>
    <value>5400000</value>
    <final>false</final>
    <source>hdfs-default.xml</source>
  </property>
▼<property>
    <name>mapreduce.job.ubertask.enable</name>
    <value>false</value>
    <final>false</final>
    <source>mapred-default.xml</source>
  </property>
▼<property>
    <name>hadoop.http.authentication.signature.secret.file</name>
    <value>${user.home}/hadoop-http-auth-signature-secret</value>
    <final>false</final>
    <source>core-default.xml</source>
  </property>
▼<property>
    <name>yarn.timeline-service.client.fd-clean-interval-secs</name>
    <value>60</value>
    <final>false</final>
    <source>yarn-default.xml</source>
  </property>
▼<property>
    <name>dfs.namenode.top.enabled</name>
    <value>true</value>
    <final>false</final>
```

QUESTION 3



```
[1]: from pyspark.sql import SparkSession
     import getpass
     username = getpass.getuser()
     spark = SparkSession. \
     builder. \
     appName("Ultron"). \
     config('spark.ui.port','0'). \
     config("spark.sql.warehouse.dir", f"/user/itv005937/warehouse"). \
     enableHiveSupport(). \
     master('yarn'). \
     getOrCreate()
```

```
[2]: groceries_schema = "order_id integer, location string, item string, order_date string, quantity integer"
```

```
[3]: from pyspark.sql import *
     from pyspark.sql.functions import *
```

```
[4]: groceries_new_df = spark.read \
     .format("csv") \
     .schema(groceries_schema) \
     .option("header","true") \
     .load("/user/itv005937/data/groceries_new.csv")
```

```
[5]: groceries_df = groceries_new_df.withColumn("order_Date", to_date("order_date","dd/mm/yyyy"))
```

```
[6]: groceries_df.createOrReplaceTempView("groceries_sql")
```

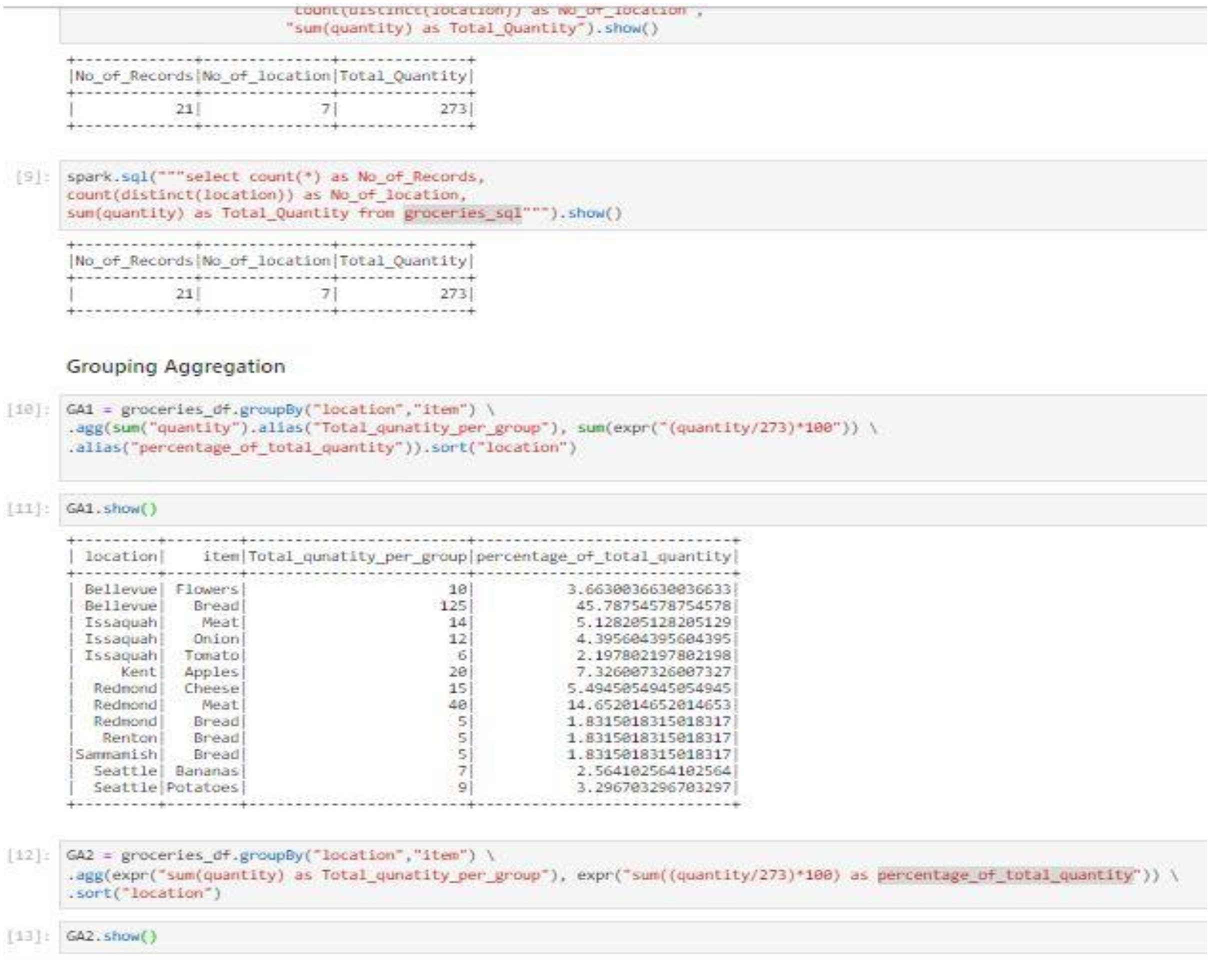
Aggregate function

```
[7]: groceries_df.select(count("*").alias("No_of_Records"),
                         countDistinct("location").alias("No_of_location"),
                         sum("quantity").alias("Total_Quantity")).show()
```

```
+-------------+--------------+--------------+
|No_of_Records|No_of_location|Total_Quantity|
+-------------+--------------+--------------+
|           21|             7|           273|
+-------------+--------------+--------------+
```

```
[8]: groceries_df.selectExpr("count(*) as No_of_Records",
                             "count(distinct(location)) as No_of_location",
                             "sum(quantity) as Total_Quantity").show()
```

```
+-------------+--------------+--------------+
|No_of_Records|No_of_location|Total_Quantity|
+-------------+--------------+--------------+
|           21|             7|           273|
+-------------+--------------+--------------+
```



```
                        count(distinct(location)) as No_of_location",
                        "sum(quantity) as Total_Quantity").show()
```

```
+--------------+--------------+--------------+
|No_of_Records|No_of_location|Total_Quantity|
+--------------+--------------+--------------+
|           21|             7|           273|
+--------------+--------------+--------------+
```

```
[9]: spark.sql("""select count(*) as No_of_Records,
     count(distinct(location)) as No_of_location,
     sum(quantity) as Total_Quantity from groceries_sql""").show()
```

```
+--------------+--------------+--------------+
|No_of_Records|No_of_location|Total_Quantity|
+--------------+--------------+--------------+
|           21|             7|           273|
+--------------+--------------+--------------+
```

Grouping Aggregation

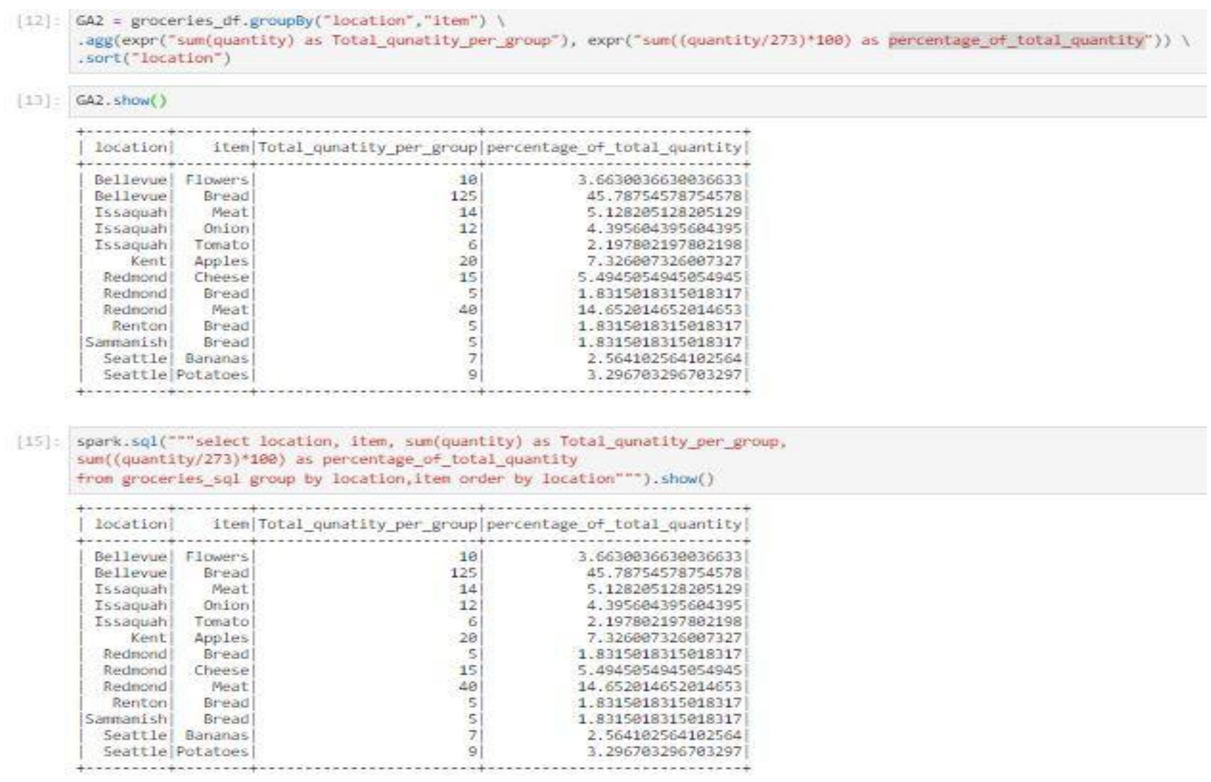
```
[10]: GA1 = groceries_df.groupBy("location","item") \
      .agg(sum("quantity").alias("Total_qunatity_per_group"), sum(expr("(quantity/273)*100")) \
      .alias("percentage_of_total_quantity")).sort("location")
```

```
[11]: GA1.show()
```

```
+---------+--------+------------------------+----------------------------+
| location|    item|Total_qunatity_per_group|percentage_of_total_quantity|
+---------+--------+------------------------+----------------------------+
| Bellevue| Flowers|                      10|          3.6630036630036633|
| Bellevue|   Bread|                     125|           45.78754578754578|
| Issaquah|    Meat|                      14|           5.128205128205129|
| Issaquah|   Onion|                      12|           4.395604395604395|
| Issaquah|  Tomato|                       6|           2.197802197802198|
|     Kent|  Apples|                      20|           7.326007326007327|
|  Redmond|  Cheese|                      15|          5.4945054945054945|
|  Redmond|    Meat|                      40|          14.652014652014653|
|  Redmond|   Bread|                       5|          1.8315018315018317|
|   Renton|   Bread|                       5|          1.8315018315018317|
|Sammamish|   Bread|                       5|          1.8315018315018317|
|  Seattle| Bananas|                       7|           2.564102564102564|
|  Seattle|Potatoes|                       9|           3.296703296703297|
+---------+--------+------------------------+----------------------------+
```

```
[12]: GA2 = groceries_df.groupBy("location","item") \
      .agg(expr("sum(quantity) as Total_qunatity_per_group"), expr("sum((quantity/273)*100) as percentage_of_total_quantity")) \
      .sort("location")
```

```
[13]: GA2.show()
```



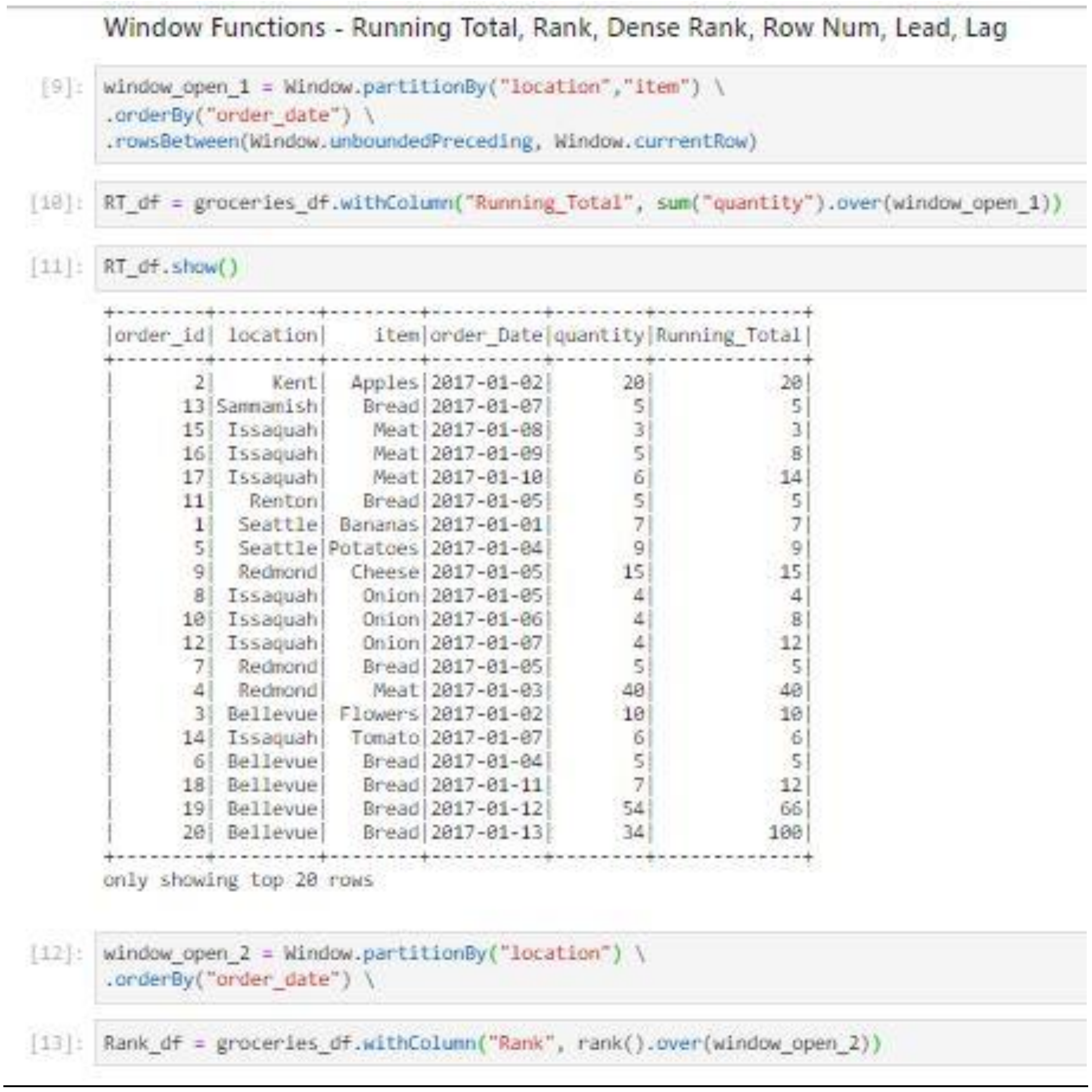
```
[12]: GA2 = groceries_df.groupBy("location","item") \
      .agg(expr("sum(quantity) as Total_qunatity_per_group"), expr("sum((quantity/273)*100) as percentage_of_total_quantity")) \
      .sort("location")
```

```
[13]: GA2.show()
```

```
+---------+--------+------------------------+----------------------------+
| location|    item|Total_qunatity_per_group|percentage_of_total_quantity|
+---------+--------+------------------------+----------------------------+
| Bellevue| Flowers|                      10|          3.6630036630036633|
| Bellevue|   Bread|                     125|           45.78754578754578|
| Issaquah|    Meat|                      14|           5.128205128205129|
| Issaquah|   Onion|                      12|           4.395604395604395|
| Issaquah|  Tomato|                       6|           2.197802197802198|
|     Kent|  Apples|                      20|           7.326007326007327|
|  Redmond|  Cheese|                      15|          5.4945054945054945|
|  Redmond|   Bread|                       5|          1.8315018315018317|
|  Redmond|    Meat|                      40|          14.652014652014653|
|   Renton|   Bread|                       5|          1.8315018315018317|
|Sammamish|   Bread|                       5|          1.8315018315018317|
|  Seattle| Bananas|                       7|           2.564102564102564|
|  Seattle|Potatoes|                       9|           3.296703296703297|
+---------+--------+------------------------+----------------------------+
```

```
[15]: spark.sql("""select location, item, sum(quantity) as Total_qunatity_per_group,
      sum((quantity/273)*100) as percentage_of_total_quantity
      from groceries_sql group by location,item order by location""").show()
```

```
+---------+--------+------------------------+----------------------------+
| location|    item|Total_qunatity_per_group|percentage_of_total_quantity|
+---------+--------+------------------------+----------------------------+
| Bellevue| Flowers|                      10|          3.6630036630036633|
| Bellevue|   Bread|                     125|           45.78754578754578|
| Issaquah|    Meat|                      14|           5.128205128205129|
| Issaquah|   Onion|                      12|           4.395604395604395|
| Issaquah|  Tomato|                       6|           2.197802197802198|
|     Kent|  Apples|                      20|           7.326007326007327|
|  Redmond|   Bread|                       5|          1.8315018315018317|
|  Redmond|  Cheese|                      15|          5.4945054945054945|
|  Redmond|    Meat|                      40|          14.652014652014653|
|   Renton|   Bread|                       5|          1.8315018315018317|
|Sammamish|   Bread|                       5|          1.8315018315018317|
|  Seattle| Bananas|                       7|           2.564102564102564|
|  Seattle|Potatoes|                       9|           3.296703296703297|
+---------+--------+------------------------+----------------------------+
```



Window Functions - Running Total, Rank, Dense Rank, Row Num, Lead, Lag

[9]:
```
window_open_1 = Window.partitionBy("location","item") \
.orderBy("order_date") \
.rowsBetween(Window.unboundedPreceding, Window.currentRow)
```

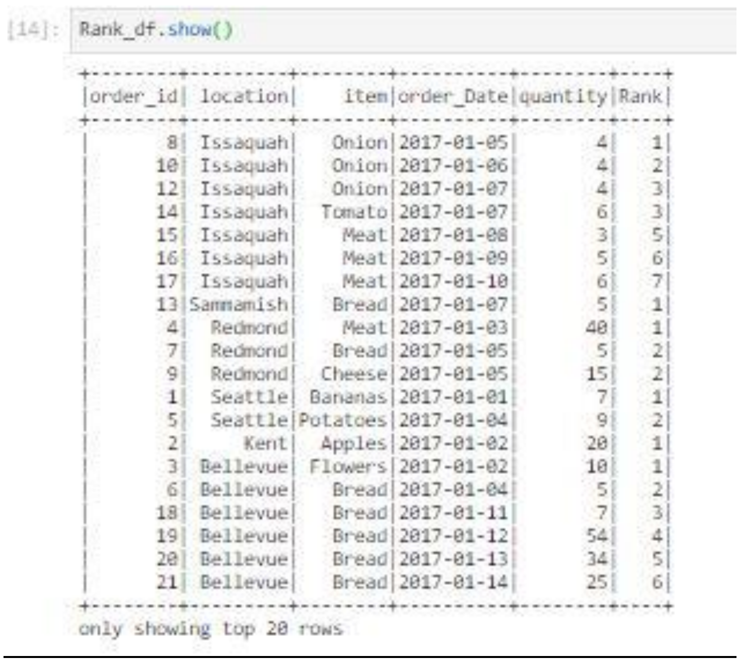
[10]:
```
RT_df = groceries_df.withColumn("Running_Total", sum("quantity").over(window_open_1))
```

[11]:
```
RT_df.show()
```

```
+--------+---------+--------+----------+--------+-------------+
|order_id| location|    item|order_Date|quantity|Running_Total|
+--------+---------+--------+----------+--------+-------------+
|       2|     Kent|  Apples|2017-01-02|      20|           20|
|      13|Sammamish|   Bread|2017-01-07|       5|            5|
|      15| Issaquah|    Meat|2017-01-08|       3|            3|
|      16| Issaquah|    Meat|2017-01-09|       5|            8|
|      17| Issaquah|    Meat|2017-01-10|       6|           14|
|      11|   Renton|   Bread|2017-01-05|       5|            5|
|       1|  Seattle| Bananas|2017-01-01|       7|            7|
|       5|  Seattle|Potatoes|2017-01-04|       9|            9|
|       9|  Redmond|  Cheese|2017-01-05|      15|           15|
|       8| Issaquah|   Onion|2017-01-05|       4|            4|
|      10| Issaquah|   Onion|2017-01-06|       4|            8|
|      12| Issaquah|   Onion|2017-01-07|       4|           12|
|       7|  Redmond|   Bread|2017-01-05|       5|            5|
|       4|  Redmond|    Meat|2017-01-03|      40|           40|
|       3| Bellevue| Flowers|2017-01-02|      10|           10|
|      14| Issaquah|  Tomato|2017-01-07|       6|            6|
|       6| Bellevue|   Bread|2017-01-04|       5|            5|
|      18| Bellevue|   Bread|2017-01-11|       7|           12|
|      19| Bellevue|   Bread|2017-01-12|      54|           66|
|      20| Bellevue|   Bread|2017-01-13|      34|          100|
+--------+---------+--------+----------+--------+-------------+
only showing top 20 rows
```

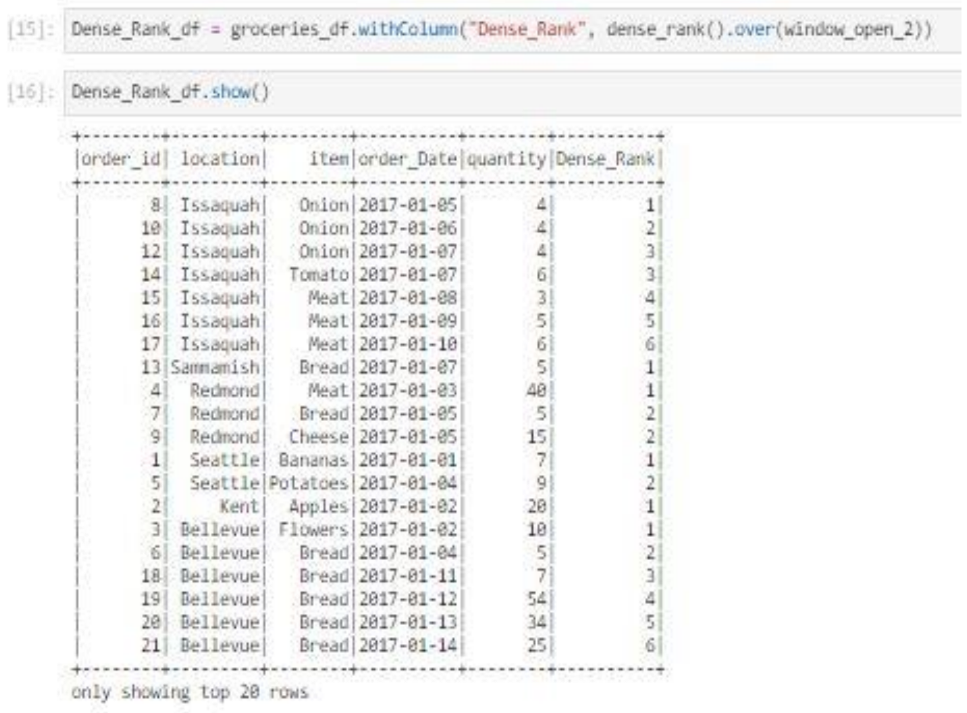
[12]:
```
window_open_2 = Window.partitionBy("location") \
.orderBy("order_date") \
```

[13]:
```
Rank_df = groceries_df.withColumn("Rank", rank().over(window_open_2))
```



[14]:
```
Rank_df.show()
```

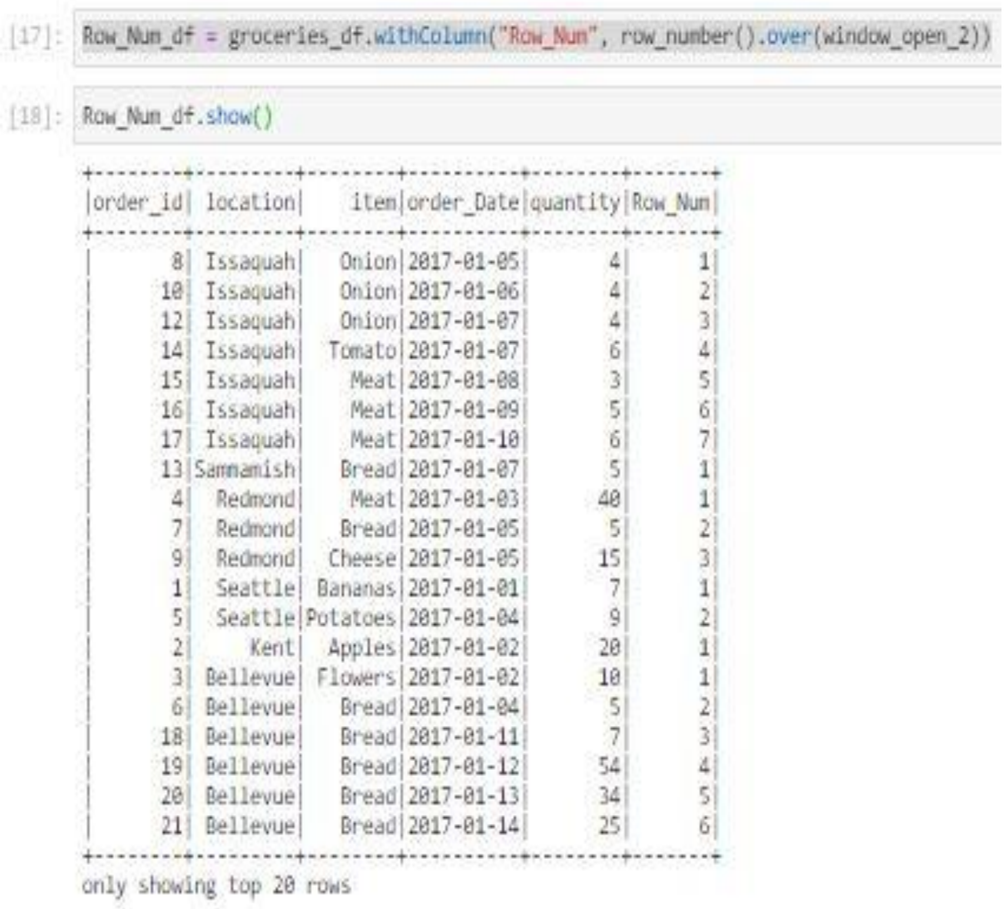
```
+--------+---------+--------+----------+--------+----+
|order_id| location|    item|order_Date|quantity|Rank|
+--------+---------+--------+----------+--------+----+
|       8| Issaquah|   Onion|2017-01-05|       4|   1|
|      10| Issaquah|   Onion|2017-01-06|       4|   2|
|      12| Issaquah|   Onion|2017-01-07|       4|   3|
|      14| Issaquah|  Tomato|2017-01-07|       6|   3|
|      15| Issaquah|    Meat|2017-01-08|       3|   5|
|      16| Issaquah|    Meat|2017-01-09|       5|   6|
|      17| Issaquah|    Meat|2017-01-10|       6|   7|
|      13|Sammamish|   Bread|2017-01-07|       5|   1|
|       4|  Redmond|    Meat|2017-01-03|      40|   1|
|       7|  Redmond|   Bread|2017-01-05|       5|   2|
|       9|  Redmond|  Cheese|2017-01-05|      15|   2|
|       1|  Seattle| Bananas|2017-01-01|       7|   1|
|       5|  Seattle|Potatoes|2017-01-04|       9|   2|
|       2|     Kent|  Apples|2017-01-02|      20|   1|
|       3| Bellevue| Flowers|2017-01-02|      10|   1|
|       6| Bellevue|   Bread|2017-01-04|       5|   2|
|      18| Bellevue|   Bread|2017-01-11|       7|   3|
|      19| Bellevue|   Bread|2017-01-12|      54|   4|
|      20| Bellevue|   Bread|2017-01-13|      34|   5|
|      21| Bellevue|   Bread|2017-01-14|      25|   6|
+--------+---------+--------+----------+--------+----+
only showing top 20 rows
```



```
[15]: Dense_Rank_df = groceries_df.withColumn("Dense_Rank", dense_rank().over(window_open_2))
```

```
[16]: Dense_Rank_df.show()
```

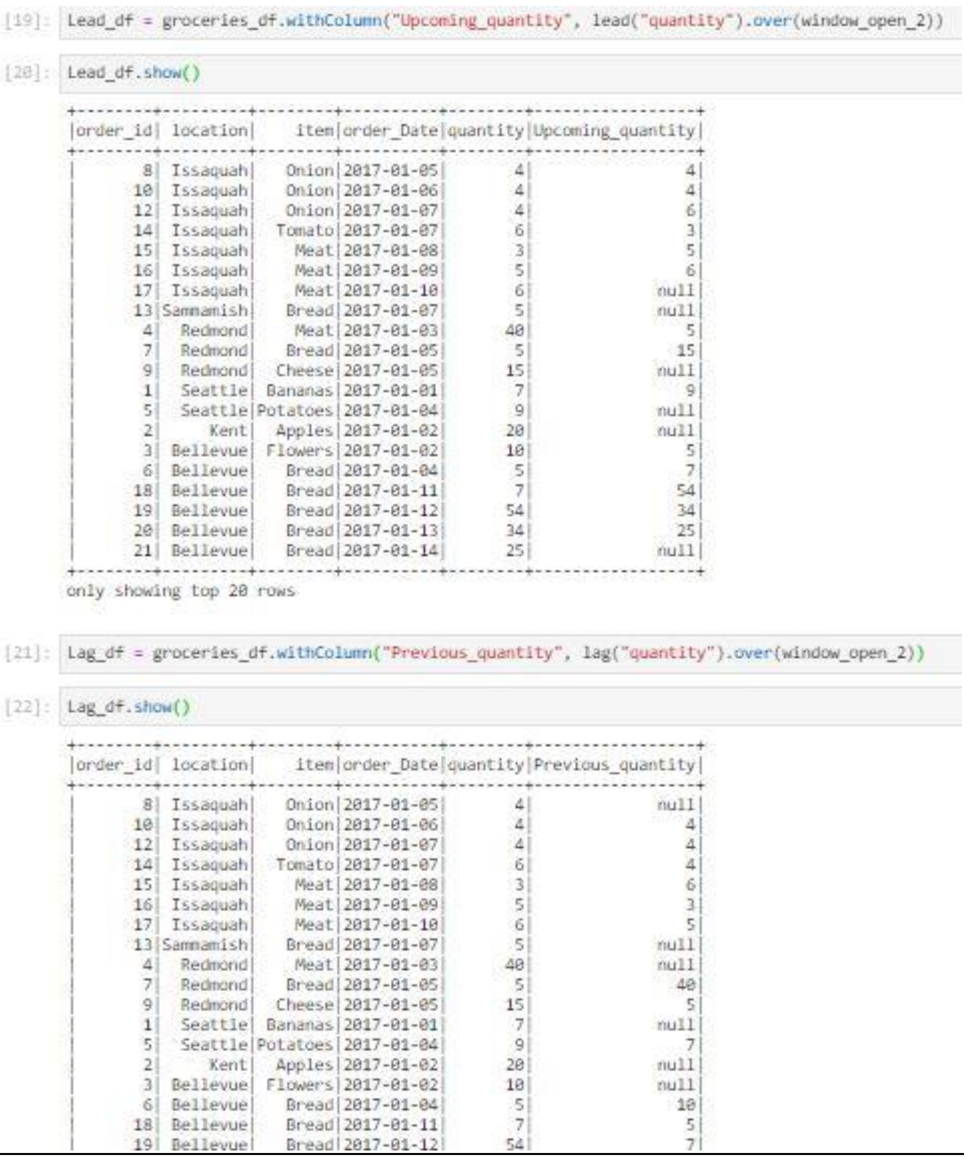
```
+--------+---------+--------+----------+--------+----------+
|order_id| location|    item|order_Date|quantity|Dense_Rank|
+--------+---------+--------+----------+--------+----------+
|       8| Issaquah|   Onion|2017-01-05|       4|         1|
|      10| Issaquah|   Onion|2017-01-06|       4|         2|
|      12| Issaquah|   Onion|2017-01-07|       4|         3|
|      14| Issaquah|  Tomato|2017-01-07|       6|         3|
|      15| Issaquah|    Meat|2017-01-08|       3|         4|
|      16| Issaquah|    Meat|2017-01-09|       5|         5|
|      17| Issaquah|    Meat|2017-01-10|       6|         6|
|      13|Sammamish|   Bread|2017-01-07|       5|         1|
|       4|  Redmond|    Meat|2017-01-03|      40|         1|
|       7|  Redmond|   Bread|2017-01-05|       5|         2|
|       9|  Redmond|  Cheese|2017-01-05|      15|         2|
|       1|  Seattle| Bananas|2017-01-01|       7|         1|
|       5|  Seattle|Potatoes|2017-01-04|       9|         2|
|       2|     Kent|  Apples|2017-01-02|      20|         1|
|       3| Bellevue| Flowers|2017-01-02|      10|         1|
|       6| Bellevue|   Bread|2017-01-04|       5|         2|
|      18| Bellevue|   Bread|2017-01-11|       7|         3|
|      19| Bellevue|   Bread|2017-01-12|      54|         4|
|      20| Bellevue|   Bread|2017-01-13|      34|         5|
|      21| Bellevue|   Bread|2017-01-14|      25|         6|
+--------+---------+--------+----------+--------+----------+
only showing top 20 rows
```



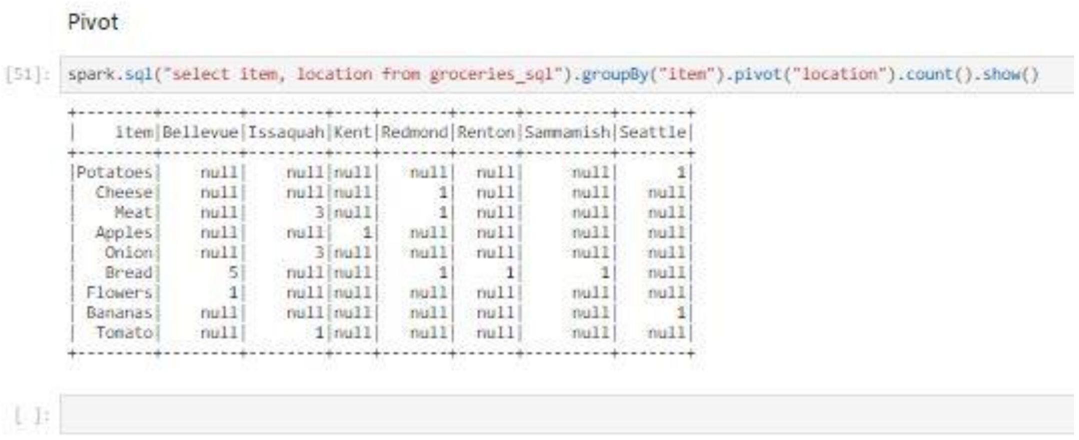
```
[17]: Row_Num_df = groceries_df.withColumn("Row_Num", row_number().over(window_open_2))
```

```
[18]: Row_Num_df.show()
```

```
+--------+---------+--------+----------+--------+-------+
|order_id| location|    item|order_Date|quantity|Row_Num|
+--------+---------+--------+----------+--------+-------+
|       8| Issaquah|   Onion|2017-01-05|       4|      1|
|      10| Issaquah|   Onion|2017-01-06|       4|      2|
|      12| Issaquah|   Onion|2017-01-07|       4|      3|
|      14| Issaquah|  Tomato|2017-01-07|       6|      4|
|      15| Issaquah|    Meat|2017-01-08|       3|      5|
|      16| Issaquah|    Meat|2017-01-09|       5|      6|
|      17| Issaquah|    Meat|2017-01-10|       6|      7|
|      13|Sammamish|   Bread|2017-01-07|       5|      1|
|       4|  Redmond|    Meat|2017-01-03|      40|      1|
|       7|  Redmond|   Bread|2017-01-05|       5|      2|
|       9|  Redmond|  Cheese|2017-01-05|      15|      3|
|       1|  Seattle| Bananas|2017-01-01|       7|      1|
|       5|  Seattle|Potatoes|2017-01-04|       9|      2|
|       2|     Kent|  Apples|2017-01-02|      20|      1|
|       3| Bellevue| Flowers|2017-01-02|      10|      1|
|       6| Bellevue|   Bread|2017-01-04|       5|      2|
|      18| Bellevue|   Bread|2017-01-11|       7|      3|
|      19| Bellevue|   Bread|2017-01-12|      54|      4|
|      20| Bellevue|   Bread|2017-01-13|      34|      5|
|      21| Bellevue|   Bread|2017-01-14|      25|      6|
+--------+---------+--------+----------+--------+-------+
only showing top 20 rows
```



```
[19]: Lead_df = groceries_df.withColumn("Upcoming_quantity", lead("quantity").over(window_open_2))

[20]: Lead_df.show()

+--------+---------+--------+----------+--------+-----------------+
|order_id| location|    item|order_Date|quantity|Upcoming_quantity|
+--------+---------+--------+----------+--------+-----------------+
|       8| Issaquah|   Onion|2017-01-05|       4|                4|
|      10| Issaquah|   Onion|2017-01-06|       4|                4|
|      12| Issaquah|   Onion|2017-01-07|       4|                6|
|      14| Issaquah|  Tomato|2017-01-07|       6|                3|
|      15| Issaquah|    Meat|2017-01-08|       3|                5|
|      16| Issaquah|    Meat|2017-01-09|       5|                6|
|      17| Issaquah|    Meat|2017-01-10|       6|             null|
|      13|Sammamish|   Bread|2017-01-07|       5|             null|
|       4|  Redmond|    Meat|2017-01-03|      40|                5|
|       7|  Redmond|   Bread|2017-01-05|       5|               15|
|       9|  Redmond|  Cheese|2017-01-05|      15|             null|
|       1|  Seattle| Bananas|2017-01-01|       7|                9|
|       5|  Seattle|Potatoes|2017-01-04|       9|             null|
|       2|     Kent|  Apples|2017-01-02|      20|             null|
|       3| Bellevue| Flowers|2017-01-02|      10|                5|
|       6| Bellevue|   Bread|2017-01-04|       5|                7|
|      18| Bellevue|   Bread|2017-01-11|       7|               54|
|      19| Bellevue|   Bread|2017-01-12|      54|               34|
|      20| Bellevue|   Bread|2017-01-13|      34|               25|
|      21| Bellevue|   Bread|2017-01-14|      25|             null|
+--------+---------+--------+----------+--------+-----------------+
only showing top 20 rows

[21]: Lag_df = groceries_df.withColumn("Previous_quantity", lag("quantity").over(window_open_2))

[22]: Lag_df.show()

+--------+---------+--------+----------+--------+-----------------+
|order_id| location|    item|order_Date|quantity|Previous_quantity|
+--------+---------+--------+----------+--------+-----------------+
|       8| Issaquah|   Onion|2017-01-05|       4|             null|
|      10| Issaquah|   Onion|2017-01-06|       4|                4|
|      12| Issaquah|   Onion|2017-01-07|       4|                4|
|      14| Issaquah|  Tomato|2017-01-07|       6|                4|
|      15| Issaquah|    Meat|2017-01-08|       3|                6|
|      16| Issaquah|    Meat|2017-01-09|       5|                3|
|      17| Issaquah|    Meat|2017-01-10|       6|                5|
|      13|Sammamish|   Bread|2017-01-07|       5|             null|
|       4|  Redmond|    Meat|2017-01-03|      40|             null|
|       7|  Redmond|   Bread|2017-01-05|       5|               40|
|       9|  Redmond|  Cheese|2017-01-05|      15|                5|
|       1|  Seattle| Bananas|2017-01-01|       7|             null|
|       5|  Seattle|Potatoes|2017-01-04|       9|                7|
|       2|     Kent|  Apples|2017-01-02|      20|             null|
|       3| Bellevue| Flowers|2017-01-02|      10|             null|
|       6| Bellevue|   Bread|2017-01-04|       5|               10|
|      18| Bellevue|   Bread|2017-01-11|       7|                5|
|      19| Bellevue|   Bread|2017-01-12|      54|                7|
```



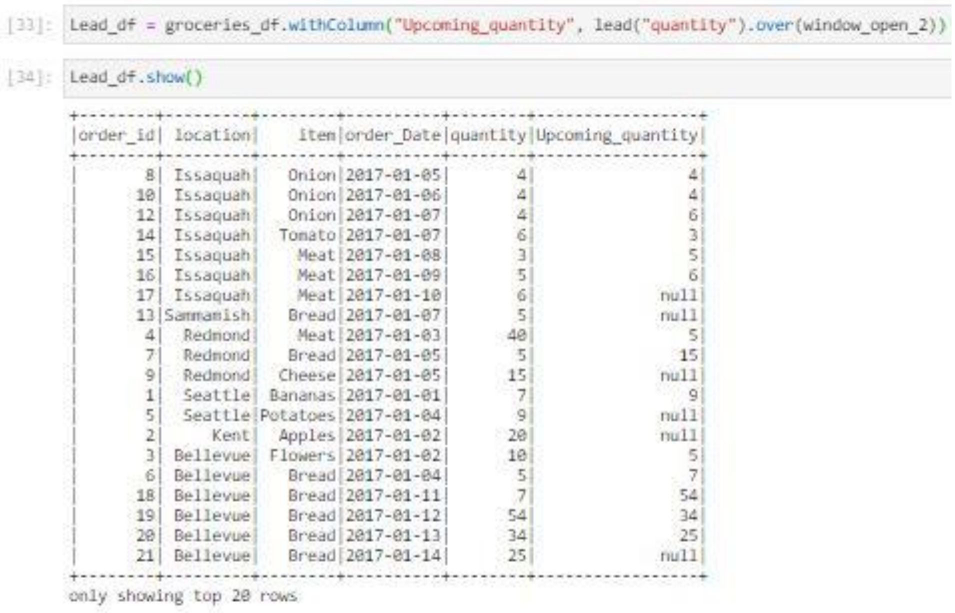
Pivot

```
[51]: spark.sql("select item, location from groceries_sql").groupBy("item").pivot("location").count().show()

+--------+--------+--------+----+-------+------+---------+-------+
|    item|Bellevue|Issaquah|Kent|Redmond|Renton|Sammamish|Seattle|
+--------+--------+--------+----+-------+------+---------+-------+
|Potatoes|    null|    null|null|   null|  null|     null|      1|
|  Cheese|    null|    null|null|      1|  null|     null|   null|
|    Meat|    null|       3|null|      1|  null|     null|   null|
|  Apples|    null|    null|   1|   null|  null|     null|   null|
|   Onion|    null|       3|null|   null|  null|     null|   null|
|   Bread|       5|    null|null|      1|     1|        1|   null|
| Flowers|       1|    null|null|   null|  null|     null|   null|
| Bananas|    null|    null|null|   null|  null|     null|      1|
|  Tomato|    null|       1|null|   null|  null|     null|   null|
+--------+--------+--------+----+-------+------+---------+-------+

[ ]:
```

QUESTION 4

Ways to handle Null Values : -

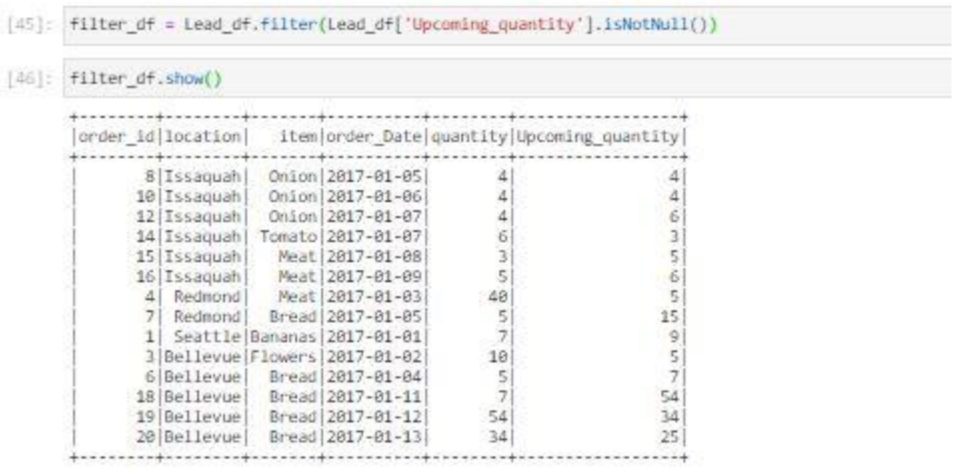
1. **Filtering –** With combination of filter() and isNotNull() function we can filter out the rows containing Null values.



```
[33]: Lead_df = groceries_df.withColumn("Upcoming_quantity", lead("quantity").over(window_open_2))
```

```
[34]: Lead_df.show()
```

```
+--------+---------+--------+----------+--------+-----------------+
|order_id| location|    item|order_Date|quantity|Upcoming_quantity|
+--------+---------+--------+----------+--------+-----------------+
|       8| Issaquah|   Onion|2017-01-05|       4|                4|
|      10| Issaquah|   Onion|2017-01-06|       4|                4|
|      12| Issaquah|   Onion|2017-01-07|       4|                6|
|      14| Issaquah|  Tomato|2017-01-07|       6|                3|
|      15| Issaquah|    Meat|2017-01-08|       3|                5|
|      16| Issaquah|    Meat|2017-01-09|       5|                6|
|      17| Issaquah|    Meat|2017-01-10|       6|             null|
|      13|Sammamish|   Bread|2017-01-07|       5|             null|
|       4|  Redmond|    Meat|2017-01-03|      40|                5|
|       7|  Redmond|   Bread|2017-01-05|       5|               15|
|       9|  Redmond|  Cheese|2017-01-05|      15|             null|
|       1|  Seattle| Bananas|2017-01-01|       7|                9|
|       5|  Seattle|Potatoes|2017-01-04|       9|             null|
|       2|     Kent|  Apples|2017-01-02|      20|             null|
|       3| Bellevue| Flowers|2017-01-02|      10|                5|
|       6| Bellevue|   Bread|2017-01-04|       5|                7|
|      18| Bellevue|   Bread|2017-01-11|       7|               54|
|      19| Bellevue|   Bread|2017-01-12|      54|               34|
|      20| Bellevue|   Bread|2017-01-13|      34|               25|
|      21| Bellevue|   Bread|2017-01-14|      25|             null|
+--------+---------+--------+----------+--------+-----------------+
only showing top 20 rows
```



```
[45]: filter_df = Lead_df.filter(Lead_df['Upcoming_quantity'].isNotNull())
```

```
[46]: filter_df.show()
```

```
+--------+--------+-------+----------+--------+-----------------+
|order_id|location|   item|order_Date|quantity|Upcoming_quantity|
+--------+--------+-------+----------+--------+-----------------+
|       8|Issaquah|  Onion|2017-01-05|       4|                4|
|      10|Issaquah|  Onion|2017-01-06|       4|                4|
|      12|Issaquah|  Onion|2017-01-07|       4|                6|
|      14|Issaquah| Tomato|2017-01-07|       6|                3|
|      15|Issaquah|   Meat|2017-01-08|       3|                5|
|      16|Issaquah|   Meat|2017-01-09|       5|                6|
|       4| Redmond|   Meat|2017-01-03|      40|                5|
|       7| Redmond|  Bread|2017-01-05|       5|               15|
|       1| Seattle|Bananas|2017-01-01|       7|                9|
|       3|Bellevue|Flowers|2017-01-02|      10|                5|
|       6|Bellevue|  Bread|2017-01-04|       5|                7|
|      18|Bellevue|  Bread|2017-01-11|       7|               54|
|      19|Bellevue|  Bread|2017-01-12|      54|               34|
|      20|Bellevue|  Bread|2017-01-13|      34|               25|
+--------+--------+-------+----------+--------+-----------------+
```

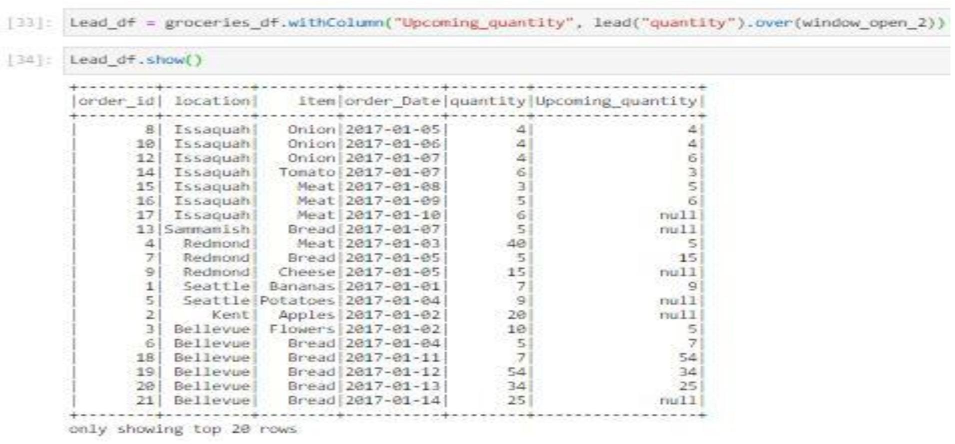
2. **Dropping Null values –** The function drop() is used to remove rows containg Null values from your dataframes. This will create a new DataFrame without Null values.

DF_without_Nulls = df.dropna()

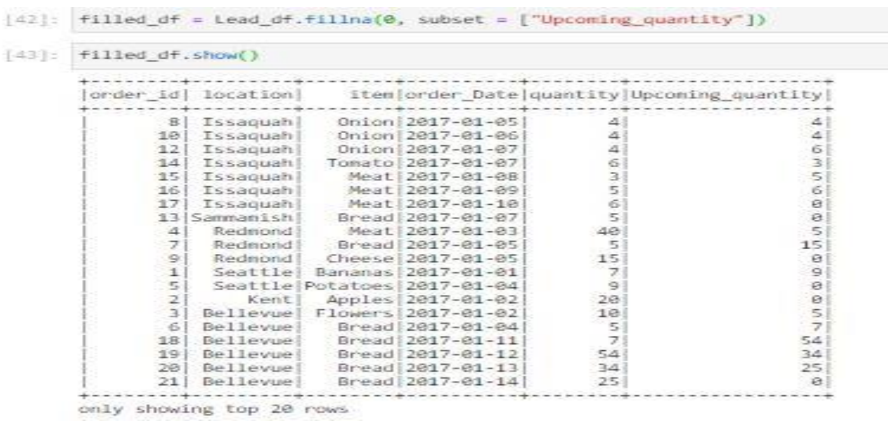
3. **<u>Filling Null values</u> -** fillna() is used to fill a value in place of null values present in particular column of a dataframe. We can use this to fill value in single column as well as all the columns containing nulls.

Single Column – filled_df = df.fillna(replacement_value, subset = [‘column name’])

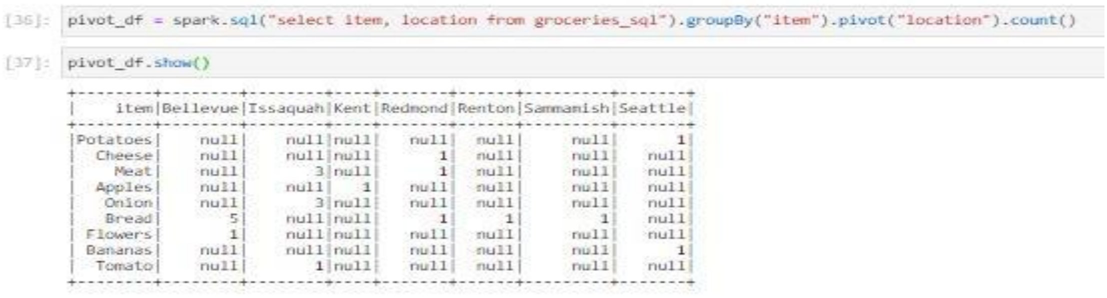
All column – df.fillna(replacement_value)



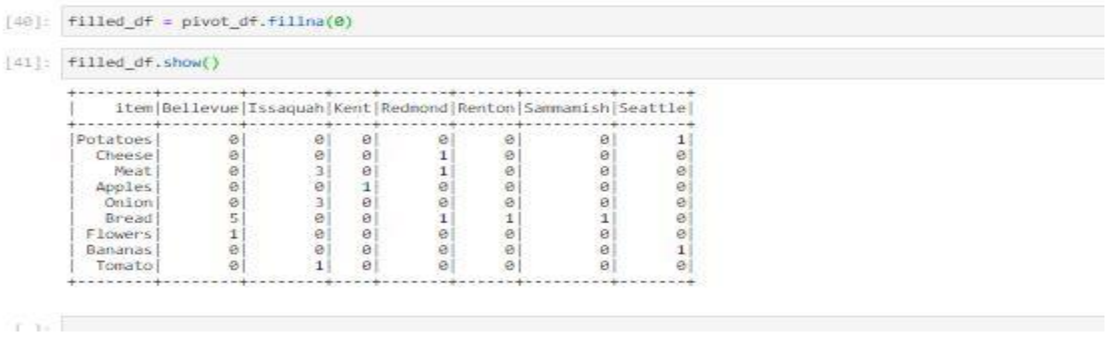
```
[33]: Lead_df = groceries_df.withColumn("Upcoming_quantity", lead("quantity").over(window_open_2))

[34]: Lead_df.show()

+--------+---------+--------+----------+--------+-----------------+
|order_id| location|    item|order_Date|quantity|Upcoming_quantity|
+--------+---------+--------+----------+--------+-----------------+
|       8| Issaquah|   Onion|2017-01-05|       4|                4|
|      10| Issaquah|   Onion|2017-01-06|       4|                4|
|      12| Issaquah|   Onion|2017-01-07|       4|                6|
|      14| Issaquah|  Tomato|2017-01-07|       6|                3|
|      15| Issaquah|    Meat|2017-01-08|       3|                5|
|      16| Issaquah|    Meat|2017-01-09|       5|                6|
|      17| Issaquah|    Meat|2017-01-10|       6|             null|
|      13|Sammamish|   Bread|2017-01-07|       5|             null|
|       4|  Redmond|    Meat|2017-01-03|      40|                5|
|       7|  Redmond|   Bread|2017-01-05|       5|               15|
|       9|  Redmond|  Cheese|2017-01-05|      15|             null|
|       1|  Seattle| Bananas|2017-01-01|       7|                9|
|       5|  Seattle|Potatoes|2017-01-04|       9|             null|
|       2|     Kent|  Apples|2017-01-02|      20|             null|
|       3| Bellevue| Flowers|2017-01-02|      10|                5|
|       6| Bellevue|   Bread|2017-01-04|       5|                7|
|      18| Bellevue|   Bread|2017-01-11|       7|               54|
|      19| Bellevue|   Bread|2017-01-12|      54|               34|
|      20| Bellevue|   Bread|2017-01-13|      34|               25|
|      21| Bellevue|   Bread|2017-01-14|      25|             null|
+--------+---------+--------+----------+--------+-----------------+
only showing top 20 rows
```



```
[42]: filled_df = Lead_df.fillna(0, subset = ["Upcoming_quantity"])

[43]: filled_df.show()

+--------+---------+--------+----------+--------+-----------------+
|order_id| location|    item|order_Date|quantity|Upcoming_quantity|
+--------+---------+--------+----------+--------+-----------------+
|       8| Issaquah|   Onion|2017-01-05|       4|                4|
|      10| Issaquah|   Onion|2017-01-06|       4|                4|
|      12| Issaquah|   Onion|2017-01-07|       4|                6|
|      14| Issaquah|  Tomato|2017-01-07|       6|                3|
|      15| Issaquah|    Meat|2017-01-08|       3|                5|
|      16| Issaquah|    Meat|2017-01-09|       5|                6|
|      17| Issaquah|    Meat|2017-01-10|       6|                0|
|      13|Sammamish|   Bread|2017-01-07|       5|                0|
|       4|  Redmond|    Meat|2017-01-03|      40|                5|
|       7|  Redmond|   Bread|2017-01-05|       5|               15|
|       9|  Redmond|  Cheese|2017-01-05|      15|                0|
|       1|  Seattle| Bananas|2017-01-01|       7|                9|
|       5|  Seattle|Potatoes|2017-01-04|       9|                0|
|       2|     Kent|  Apples|2017-01-02|      20|                0|
|       3| Bellevue| Flowers|2017-01-02|      10|                5|
|       6| Bellevue|   Bread|2017-01-04|       5|                7|
|      18| Bellevue|   Bread|2017-01-11|       7|               54|
|      19| Bellevue|   Bread|2017-01-12|      54|               34|
|      20| Bellevue|   Bread|2017-01-13|      34|               25|
|      21| Bellevue|   Bread|2017-01-14|      25|                0|
+--------+---------+--------+----------+--------+-----------------+
only showing top 20 rows
```



```
[36]: pivot_df = spark.sql("select item, location from groceries_sql").groupBy("item").pivot("location").count()

[37]: pivot_df.show()

+--------+--------+--------+----+-------+------+---------+-------+
|    item|Bellevue|Issaquah|Kent|Redmond|Renton|Sammamish|Seattle|
+--------+--------+--------+----+-------+------+---------+-------+
|Potatoes|    null|    null|null|   null|  null|     null|      1|
|  Cheese|    null|    null|null|      1|  null|     null|   null|
|    Meat|    null|       3|null|      1|  null|     null|   null|
|  Apples|    null|    null|   1|   null|  null|     null|   null|
|   Onion|    null|       3|null|   null|  null|     null|   null|
|   Bread|       5|    null|null|      1|     1|        1|   null|
| Flowers|       1|    null|null|   null|  null|     null|   null|
| Bananas|    null|    null|null|   null|  null|     null|      1|
|  Tomato|    null|       1|null|   null|  null|     null|   null|
+--------+--------+--------+----+-------+------+---------+-------+
```

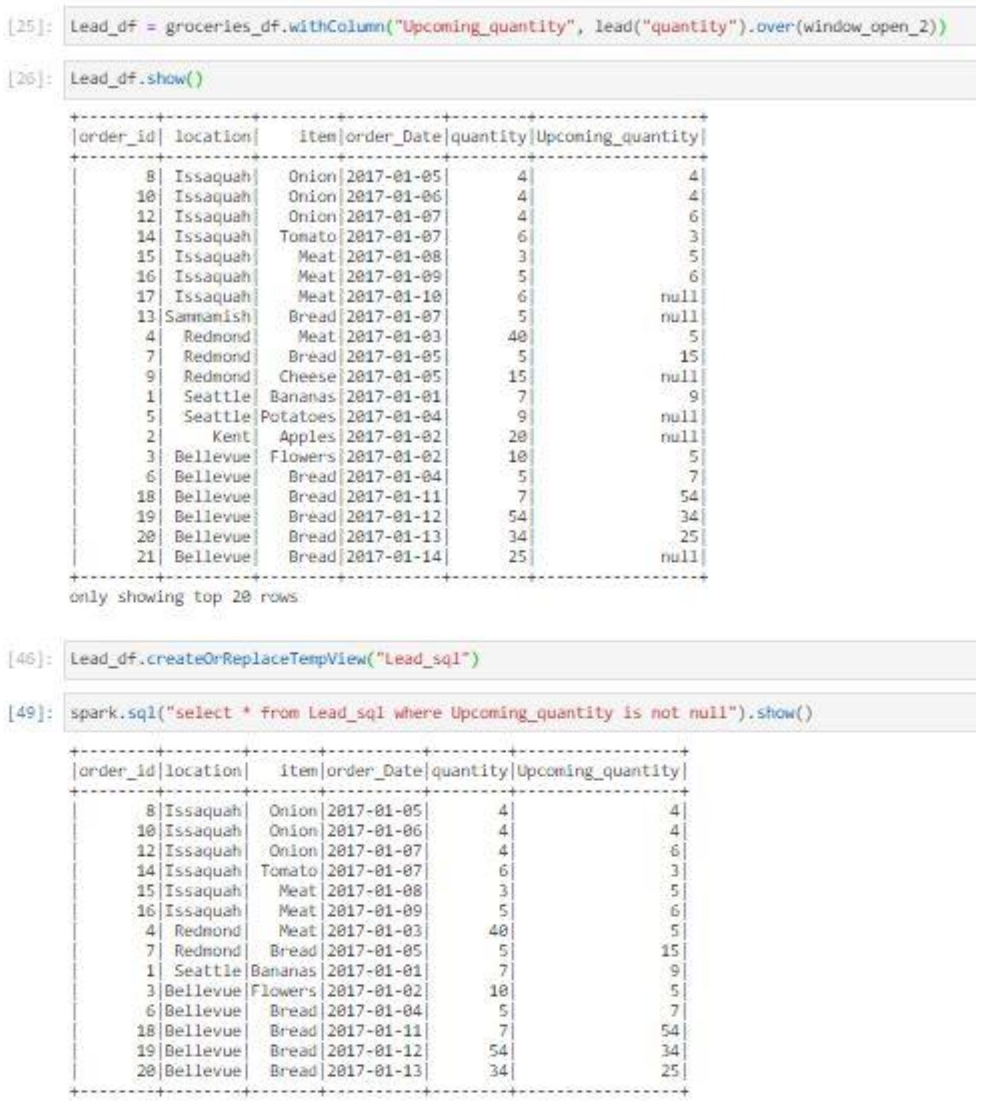


```
[40]: filled_df = pivot_df.fillna(0)

[41]: filled_df.show()

+--------+--------+--------+----+-------+------+---------+-------+
|    item|Bellevue|Issaquah|Kent|Redmond|Renton|Sammamish|Seattle|
+--------+--------+--------+----+-------+------+---------+-------+
|Potatoes|       0|       0|   0|      0|     0|        0|      1|
|  Cheese|       0|       0|   0|      1|     0|        0|      0|
|    Meat|       0|       3|   0|      1|     0|        0|      0|
|  Apples|       0|       0|   1|      0|     0|        0|      0|
|   Onion|       0|       3|   0|      0|     0|        0|      0|
|   Bread|       5|       0|   0|      1|     1|        1|      0|
| Flowers|       1|       0|   0|      0|     0|        0|      0|
| Bananas|       0|       0|   0|      0|     0|        0|      1|
|  Tomato|       0|       1|   0|      0|     0|        0|      0|
+--------+--------+--------+----+-------+------+---------+-------+

[ ]:
```

4. **Nulls in Mathematical operations –** Use the na propery to handle nulls in mathematical operations. For example - null values can be replaced by a value when performing arithmetic operation.

Math_df = df.na.fill(replacement_value)

5. **Spark.sql query filter -** We can write custom spark.sql query to filter out null values using where column_name is not null.



```
[25]: Lead_df = groceries_df.withColumn("Upcoming_quantity", lead("quantity").over(window_open_2))
```

```
[26]: Lead_df.show()
```

```
+--------+--------+--------+----------+--------+-----------------+
|order_id| location|    item|order_Date|quantity|Upcoming_quantity|
+--------+--------+--------+----------+--------+-----------------+
|       8| Issaquah|   Onion|2017-01-05|       4|                4|
|      10| Issaquah|   Onion|2017-01-06|       4|                4|
|      12| Issaquah|   Onion|2017-01-07|       4|                6|
|      14| Issaquah|  Tomato|2017-01-07|       6|                3|
|      15| Issaquah|    Meat|2017-01-08|       3|                5|
|      16| Issaquah|    Meat|2017-01-09|       5|                6|
|      17| Issaquah|    Meat|2017-01-10|       6|             null|
|      13|Sammamish|   Bread|2017-01-07|       5|             null|
|       4|  Redmond|    Meat|2017-01-03|      40|                5|
|       7|  Redmond|   Bread|2017-01-05|       5|               15|
|       9|  Redmond|  Cheese|2017-01-05|      15|             null|
|       1|  Seattle| Bananas|2017-01-01|       7|                9|
|       5|  Seattle|Potatoes|2017-01-04|       9|             null|
|       2|     Kent|  Apples|2017-01-02|      20|             null|
|       3| Bellevue| Flowers|2017-01-02|      10|                5|
|       6| Bellevue|   Bread|2017-01-04|       5|                7|
|      18| Bellevue|   Bread|2017-01-11|       7|               54|
|      19| Bellevue|   Bread|2017-01-12|      54|               34|
|      20| Bellevue|   Bread|2017-01-13|      34|               25|
|      21| Bellevue|   Bread|2017-01-14|      25|             null|
+--------+--------+--------+----------+--------+-----------------+
only showing top 20 rows
```

```
[46]: Lead_df.createOrReplaceTempView("Lead_sql")
```

```
[49]: spark.sql("select * from Lead_sql where Upcoming_quantity is not null").show()
```

```
+--------+--------+-------+----------+--------+-----------------+
|order_id|location|   item|order_Date|quantity|Upcoming_quantity|
+--------+--------+-------+----------+--------+-----------------+
|       8|Issaquah|  Onion|2017-01-05|       4|                4|
|      10|Issaquah|  Onion|2017-01-06|       4|                4|
|      12|Issaquah|  Onion|2017-01-07|       4|                6|
|      14|Issaquah| Tomato|2017-01-07|       6|                3|
|      15|Issaquah|   Meat|2017-01-08|       3|                5|
|      16|Issaquah|   Meat|2017-01-09|       5|                6|
|       4| Redmond|   Meat|2017-01-03|      40|                5|
|       7| Redmond|  Bread|2017-01-05|       5|               15|
|       1| Seattle|Bananas|2017-01-01|       7|                9|
|       3|Bellevue|Flowers|2017-01-02|      10|                5|
|       6|Bellevue|  Bread|2017-01-04|       5|                7|
|      18|Bellevue|  Bread|2017-01-11|       7|               54|
|      19|Bellevue|  Bread|2017-01-12|      54|               34|
|      20|Bellevue|  Bread|2017-01-13|      34|               25|
+--------+--------+-------+----------+--------+-----------------+
```