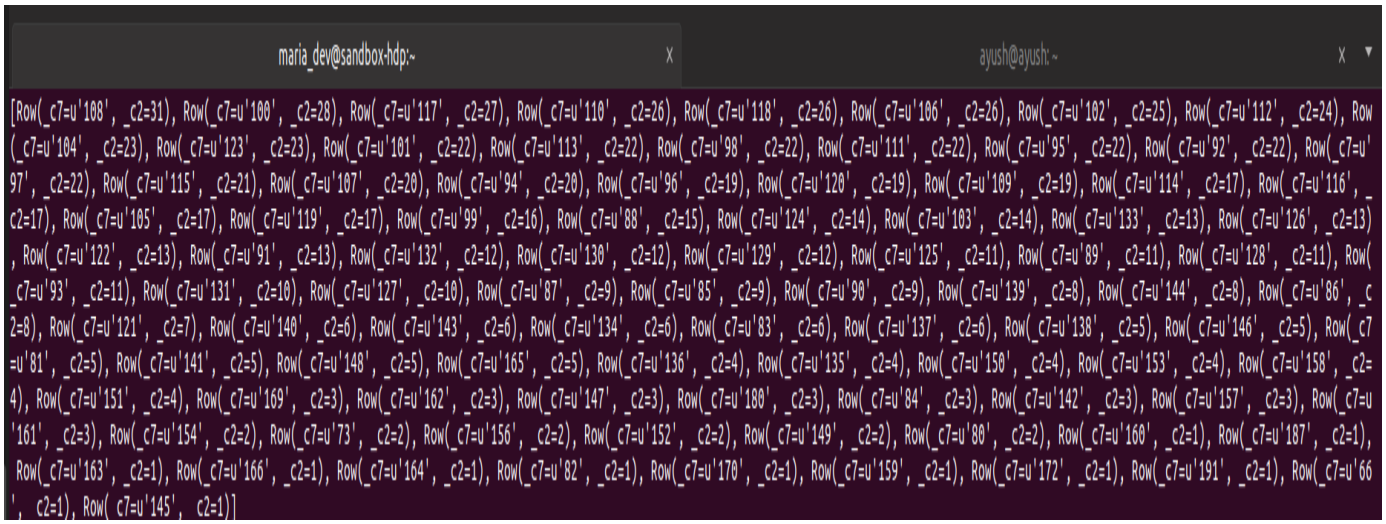
1.

result1 = spark.sql("SELECT _c7, count(*) as _c2 from df group by _c7 order by _c2 DESC")

OUTPUT

>>> result1.collect()

[Row(_c7=u'108', _c2=31), Row(_c7=u'100', _c2=28), Row(_c7=u'117', _c2=27), Row(_c7=u'110', _c2=26), Row(_c7=u'118', _c2=26), Row(_c7=u'106', _c2=26), Row(_c7=u'102', _c2=25), Row(_c7=u'112', _c2=24), Row(_c7=u'104', _c2=23), Row(_c7=u'123', _c2=23), Row(_c7=u'101', _c2=22), Row(_c7=u'113', _c2=22), Row(_c7=u'98', _c2=22), Row(_c7=u'111', _c2=22), Row(_c7=u'95', _c2=22), Row(_c7=u'92', _c2=22), Row(_c7=u'97', _c2=22), Row(_c7=u'115', _c2=21), Row(_c7=u'107', _c2=20), Row(_c7=u'94', _c2=20), Row(_c7=u'96', _c2=19), Row(_c7=u'120', _c2=19), Row(_c7=u'109', _c2=19), Row(_c7=u'114', _c2=17), Row(_c7=u'116', _c2=17), Row(_c7=u'105', _c2=17), Row(_c7=u'119', _c2=17), Row(_c7=u'99', _c2=16), Row(_c7=u'88', _c2=15), Row(_c7=u'124', _c2=14), Row(_c7=u'103', _c2=14), Row(_c7=u'133', _c2=13), Row(_c7=u'126', _c2=13), Row(_c7=u'122', _c2=13), Row(_c7=u'91', _c2=13), Row(_c7=u'132', _c2=12), Row(_c7=u'130', _c2=12), Row(_c7=u'129', _c2=12), Row(_c7=u'125', _c2=11), Row(_c7=u'89', _c2=11), Row(_c7=u'128', _c2=11), Row(_c7=u'93', _c2=11), Row(_c7=u'131', _c2=10), Row(_c7=u'127', _c2=10), Row(_c7=u'87', _c2=9), Row(_c7=u'85', _c2=9), Row(_c7=u'90', _c2=9), Row(_c7=u'139', _c2=8), Row(_c7=u'144', _c2=8), Row(_c7=u'86', _c2=8), Row(_c7=u'121', _c2=7), Row(_c7=u'140', _c2=6), Row(_c7=u'143', _c2=6), Row(_c7=u'134', _c2=6), Row(_c7=u'83', _c2=6), Row(_c7=u'137', _c2=6), Row(_c7=u'138', _c2=5), Row(_c7=u'146', _c2=5), Row(_c7=u'81', _c2=5), Row(_c7=u'141', _c2=5), Row(_c7=u'148', _c2=5), Row(_c7=u'165', _c2=5), Row(_c7=u'136', _c2=4), Row(_c7=u'135', _c2=4), Row(_c7=u'150', _c2=4), Row(_c7=u'153', _c2=4), Row(_c7=u'158', _c2=4), Row(_c7=u'151', _c2=4), Row(_c7=u'169', _c2=3), Row(_c7=u'162', _c2=3), Row(_c7=u'147', _c2=3), Row(_c7=u'180', _c2=3), Row(_c7=u'84', _c2=3), Row(_c7=u'142', _c2=3), Row(_c7=u'157', _c2=3), Row(_c7=u'161', _c2=3), Row(_c7=u'154', _c2=2), Row(_c7=u'73', _c2=2), Row(_c7=u'156', _c2=2), Row(_c7=u'152', _c2=2), Row(_c7=u'149', _c2=2), Row(_c7=u'80', _c2=2), Row(_c7=u'160', _c2=1), Row(_c7=u'187', _c2=1), Row(_c7=u'163', _c2=1), Row(_c7=u'166', _c2=1), Row(_c7=u'164', _c2=1), Row(_c7=u'82', _c2=1), Row(_c7=u'170', _c2=1), Row(_c7=u'159', _c2=1), Row(_c7=u'172', _c2=1), Row(_c7=u'191', _c2=1), Row(_c7=u'66', _c2=1), Row(_c7=u'145', _c2=1)]



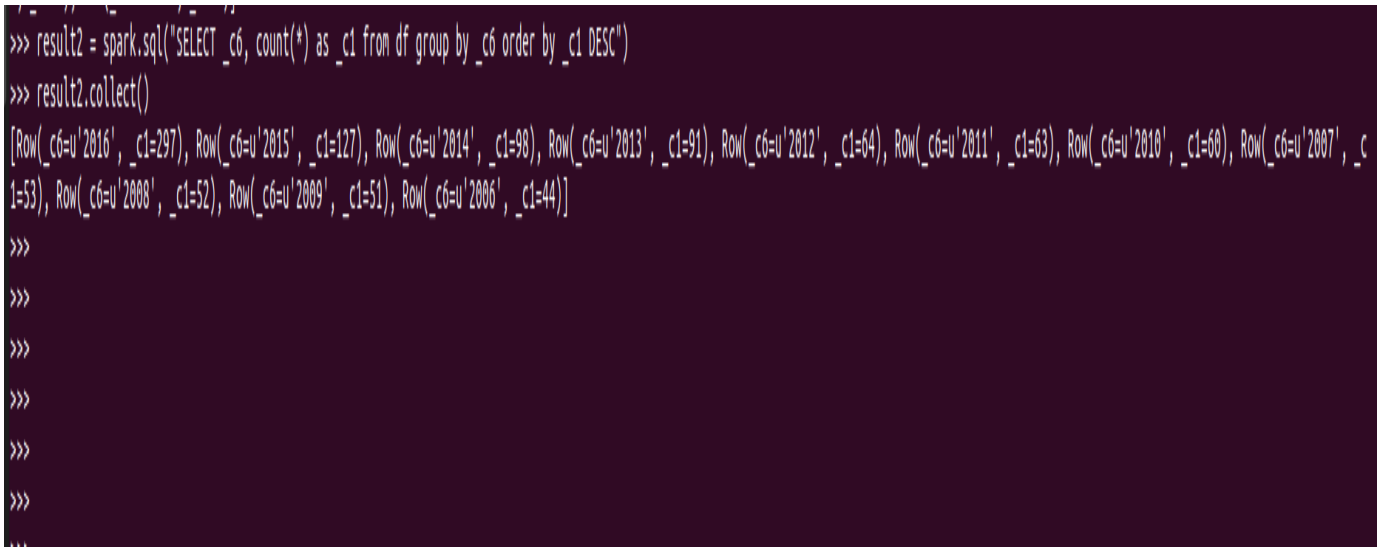
2.

result2 = spark.sql("SELECT _c6, count(*) as _c1 from df group by _c6 order by _c1 DESC")

OUTPUT:

>>> result2.collect()

[Row(_c6=u'2016', _c1=297), Row(_c6=u'2015', _c1=127), Row(_c6=u'2014', _c1=98), Row(_c6=u'2013', _c1=91), Row(_c6=u'2012', _c1=64), Row(_c6=u'2011', _c1=63), Row(_c6=u'2010', _c1=60), Row(_c6=u'2007', _c1=53), Row(_c6=u'2008', _c1=52), Row(_c6=u'2009', _c1=51), Row(_c6=u'2006', _c1=44)]

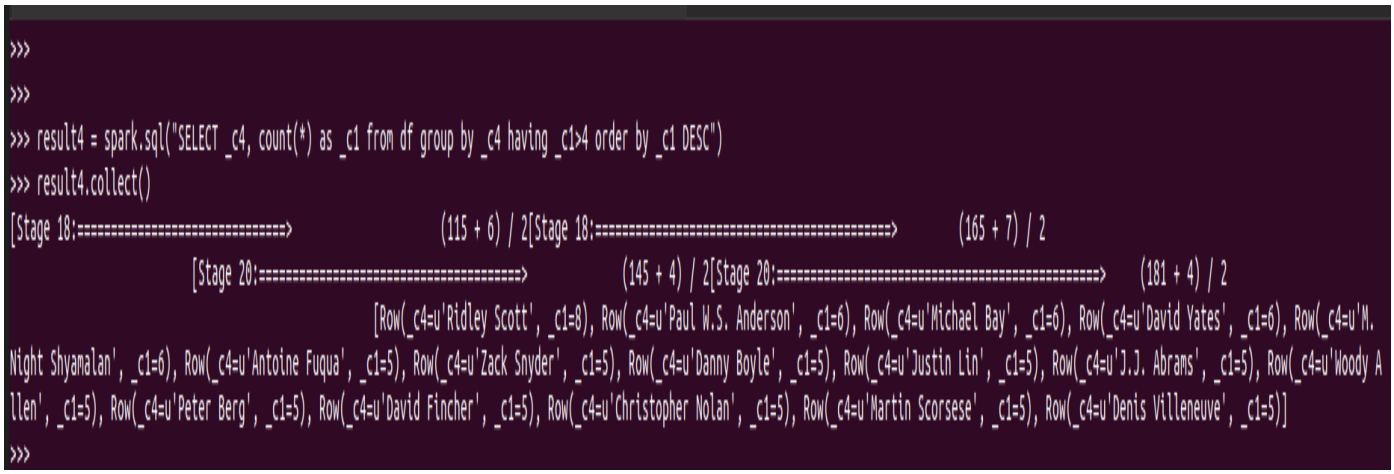


3.result4 = spark.sql("SELECT _c4, count(*) as _c1 from df group by _c4 having _c1>4 order by _c1 DESC")

OUTPUT:

>>> result4.collect()

[Stage 18:==============================> (115 + 6) / 2[Stage 18:==> (165 + 7) / 2
[Stage 20:======================================> (145 + 4) / 2[Stage 20:==> (181 + 4) / 2
[Row(_c4=u'Ridley Scott', _c1=8), Row(_c4=u'Paul W.S. Anderson', _c1=6), Row(_c4=u'Michael Bay', _c1=6), Row(_c4=u'David Yates', _c1=6), Row(_c4=u'M. Night Shyamalan', _c1=6), Row(_c4=u'Antoine Fuqua', _c1=5), Row(_c4=u'Zack Snyder', _c1=5), Row(_c4=u'Danny Boyle', _c1=5), Row(_c4=u'Justin Lin', _c1=5), Row(_c4=u'J.J. Abrams', _c1=5), Row(_c4=u'Woody Allen', _c1=5), Row(_c4=u'Peter Berg', _c1=5), Row(_c4=u'David Fincher', _c1=5), Row(_c4=u'Christopher Nolan', _c1=5), Row(_c4=u'Martin Scorsese', _c1=5), Row(_c4=u'Denis Villeneuve', _c1=5)]



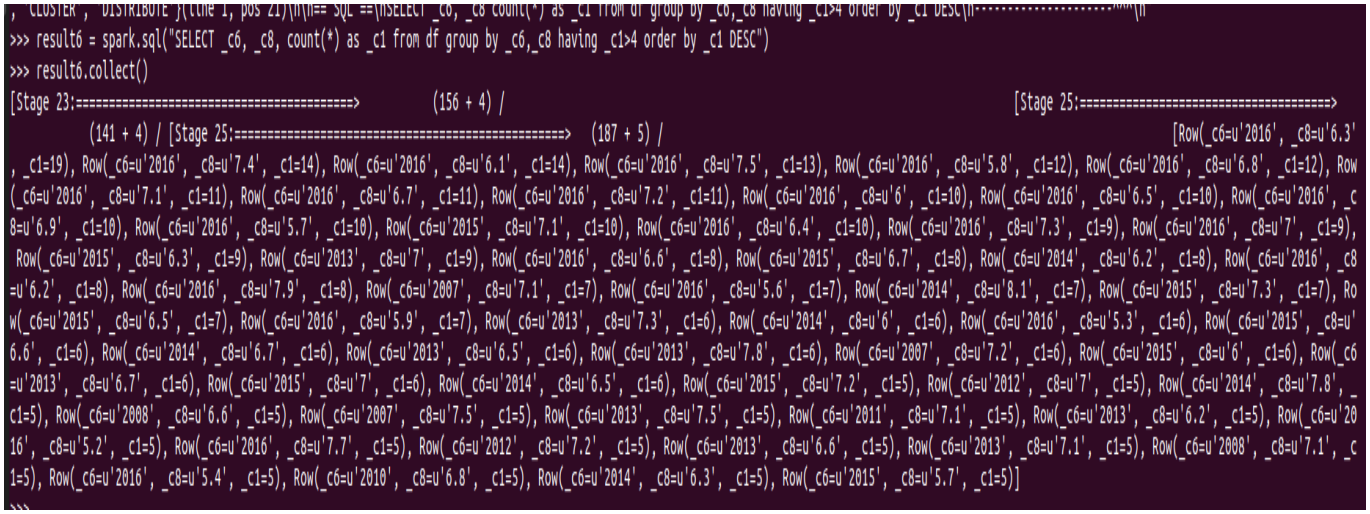
4.

result6 = spark.sql("SELECT _c6, _c8, count(*) as _c1 from df group by _c6,_c8 having _c1>4 order by _c1 DESC")

OUTPUT:

>>> result6.collect()

[Stage 23:===> (156 + 4) /
[Stage 25:=====================================> (141 + 4) / [Stage
25:==> (187 + 5) /
[Row(_c6=u'2016', _c8=u'6.3', _c1=19), Row(_c6=u'2016', _c8=u'7.4', _c1=14), Row(_c6=u'2016',
_c8=u'6.1', _c1=14), Row(_c6=u'2016', _c8=u'7.5', _c1=13), Row(_c6=u'2016', _c8=u'5.8', _c1=12),
Row(_c6=u'2016', _c8=u'6.8', _c1=12), Row(_c6=u'2016', _c8=u'7.1', _c1=11), Row(_c6=u'2016',
_c8=u'6.7', _c1=11), Row(_c6=u'2016', _c8=u'7.2', _c1=11), Row(_c6=u'2016', _c8=u'6', _c1=10),
Row(_c6=u'2016', _c8=u'6.5', _c1=10), Row(_c6=u'2016', _c8=u'6.9', _c1=10), Row(_c6=u'2016',
_c8=u'5.7', _c1=10), Row(_c6=u'2015', _c8=u'7.1', _c1=10), Row(_c6=u'2016', _c8=u'6.4', _c1=10),
Row(_c6=u'2016', _c8=u'7.3', _c1=9), Row(_c6=u'2016', _c8=u'7', _c1=9), Row(_c6=u'2015',
_c8=u'6.3', _c1=9), Row(_c6=u'2013', _c8=u'7', _c1=9), Row(_c6=u'2016', _c8=u'6.6', _c1=8),
Row(_c6=u'2015', _c8=u'6.7', _c1=8), Row(_c6=u'2014', _c8=u'6.2', _c1=8), Row(_c6=u'2016',
_c8=u'6.2', _c1=8), Row(_c6=u'2016', _c8=u'7.9', _c1=8), Row(_c6=u'2007', _c8=u'7.1', _c1=7),
Row(_c6=u'2016', _c8=u'5.6', _c1=7), Row(_c6=u'2014', _c8=u'8.1', _c1=7), Row(_c6=u'2015',
_c8=u'7.3', _c1=7), Row(_c6=u'2015', _c8=u'6.5', _c1=7), Row(_c6=u'2016', _c8=u'5.9', _c1=7),
Row(_c6=u'2013', _c8=u'7.3', _c1=6), Row(_c6=u'2014', _c8=u'6', _c1=6), Row(_c6=u'2016',
_c8=u'5.3', _c1=6), Row(_c6=u'2015', _c8=u'6.6', _c1=6), Row(_c6=u'2014', _c8=u'6.7', _c1=6),
Row(_c6=u'2013', _c8=u'6.5', _c1=6), Row(_c6=u'2013', _c8=u'7.8', _c1=6), Row(_c6=u'2007',
_c8=u'7.2', _c1=6), Row(_c6=u'2015', _c8=u'6', _c1=6), Row(_c6=u'2013', _c8=u'6.7', _c1=6),
Row(_c6=u'2015', _c8=u'7', _c1=6), Row(_c6=u'2014', _c8=u'6.5', _c1=6), Row(_c6=u'2015',
_c8=u'7.2', _c1=5), Row(_c6=u'2012', _c8=u'7', _c1=5), Row(_c6=u'2014', _c8=u'7.8', _c1=5),
Row(_c6=u'2008', _c8=u'6.6', _c1=5), Row(_c6=u'2007', _c8=u'7.5', _c1=5), Row(_c6=u'2013',
_c8=u'7.5', _c1=5), Row(_c6=u'2011', _c8=u'7.1', _c1=5), Row(_c6=u'2013', _c8=u'6.2', _c1=5),
Row(_c6=u'2016', _c8=u'5.2', _c1=5), Row(_c6=u'2016', _c8=u'7.7', _c1=5), Row(_c6=u'2012',
_c8=u'7.2', _c1=5), Row(_c6=u'2013', _c8=u'6.6', _c1=5), Row(_c6=u'2013', _c8=u'7.1', _c1=5),
Row(_c6=u'2008', _c8=u'7.1', _c1=5), Row(_c6=u'2016', _c8=u'5.4', _c1=5), Row(_c6=u'2010',
_c8=u'6.8', _c1=5), Row(_c6=u'2014', _c8=u'6.3', _c1=5), Row(_c6=u'2015', _c8=u'5.7', _c1=5)]


, 'CLUSTER', 'DISTRIBUTE')(line 1, pos 21)\n\n== SQL ==\nSELECT _c6, _c8 count(*) as _c1 from df group by _c6,_c8 having _c1>4 order by _c1 DESC\n^^^\n
>>> result6 = spark.sql("SELECT _c6, _c8, count(*) as _c1 from df group by _c6,_c8 having _c1>4 order by _c1 DESC")
>>> result6.collect()
[Stage 23:==========> (156 + 4) / [Stage 25:==========>
(141 + 4) / [Stage 25:==========> (187 + 5) / [Row(_c6=u'2016', _c8=u'6.3'
, _c1=19), Row(_c6=u'2016', _c8=u'7.4', _c1=14), Row(_c6=u'2016', _c8=u'6.1', _c1=14), Row(_c6=u'2016', _c8=u'7.5', _c1=13), Row(_c6=u'2016', _c8=u'5.8', _c1=12), Row(_c6=u'2016', _c8=u'6.8', _c1=12), Row
(_c6=u'2016', _c8=u'7.1', _c1=11), Row(_c6=u'2016', _c8=u'6.7', _c1=11), Row(_c6=u'2016', _c8=u'7.2', _c1=11), Row(_c6=u'2016', _c8=u'6', _c1=10), Row(_c6=u'2016', _c8=u'6.5', _c1=10), Row(_c6=u'2016', _c
8=u'6.9', _c1=10), Row(_c6=u'2016', _c8=u'5.7', _c1=10), Row(_c6=u'2015', _c8=u'7.1', _c1=10), Row(_c6=u'2016', _c8=u'6.4', _c1=10), Row(_c6=u'2016', _c8=u'7.3', _c1=9), Row(_c6=u'2016', _c8=u'7', _c1=9),
Row(_c6=u'2015', _c8=u'6.3', _c1=9), Row(_c6=u'2013', _c8=u'7', _c1=9), Row(_c6=u'2016', _c8=u'6.6', _c1=8), Row(_c6=u'2015', _c8=u'6.7', _c1=8), Row(_c6=u'2014', _c8=u'6.2', _c1=8), Row(_c6=u'2016', _c8
=u'6.2', _c1=8), Row(_c6=u'2016', _c8=u'7.9', _c1=8), Row(_c6=u'2007', _c8=u'7.1', _c1=7), Row(_c6=u'2016', _c8=u'5.6', _c1=7), Row(_c6=u'2014', _c8=u'8.1', _c1=7), Row(_c6=u'2015', _c8=u'7.3', _c1=7), Ro
w(_c6=u'2015', _c8=u'6.5', _c1=7), Row(_c6=u'2016', _c8=u'5.9', _c1=7), Row(_c6=u'2013', _c8=u'7.3', _c1=6), Row(_c6=u'2014', _c8=u'6', _c1=6), Row(_c6=u'2016', _c8=u'5.3', _c1=6), Row(_c6=u'2015', _c8=u'
6.6', _c1=6), Row(_c6=u'2014', _c8=u'6.7', _c1=6), Row(_c6=u'2013', _c8=u'6.5', _c1=6), Row(_c6=u'2013', _c8=u'7.8', _c1=6), Row(_c6=u'2007', _c8=u'7.2', _c1=6), Row(_c6=u'2015', _c8=u'6', _c1=6), Row(_c6
=u'2013', _c8=u'6.7', _c1=6), Row(_c6=u'2015', _c8=u'7', _c1=6), Row(_c6=u'2014', _c8=u'6.5', _c1=6), Row(_c6=u'2015', _c8=u'7.2', _c1=5), Row(_c6=u'2012', _c8=u'7', _c1=5), Row(_c6=u'2014', _c8=u'7.8', _
c1=5), Row(_c6=u'2008', _c8=u'6.6', _c1=5), Row(_c6=u'2007', _c8=u'7.5', _c1=5), Row(_c6=u'2013', _c8=u'7.5', _c1=5), Row(_c6=u'2011', _c8=u'7.1', _c1=5), Row(_c6=u'2013', _c8=u'6.2', _c1=5), Row(_c6=u'20
16', _c8=u'5.2', _c1=5), Row(_c6=u'2016', _c8=u'7.7', _c1=5), Row(_c6=u'2012', _c8=u'7.2', _c1=5), Row(_c6=u'2013', _c8=u'6.6', _c1=5), Row(_c6=u'2013', _c8=u'7.1', _c1=5), Row(_c6=u'2008', _c8=u'7.1', _c
1=5), Row(_c6=u'2016', _c8=u'5.4', _c1=5), Row(_c6=u'2010', _c8=u'6.8', _c1=5), Row(_c6=u'2014', _c8=u'6.3', _c1=5), Row(_c6=u'2015', _c8=u'5.7', _c1=5)]
>>>

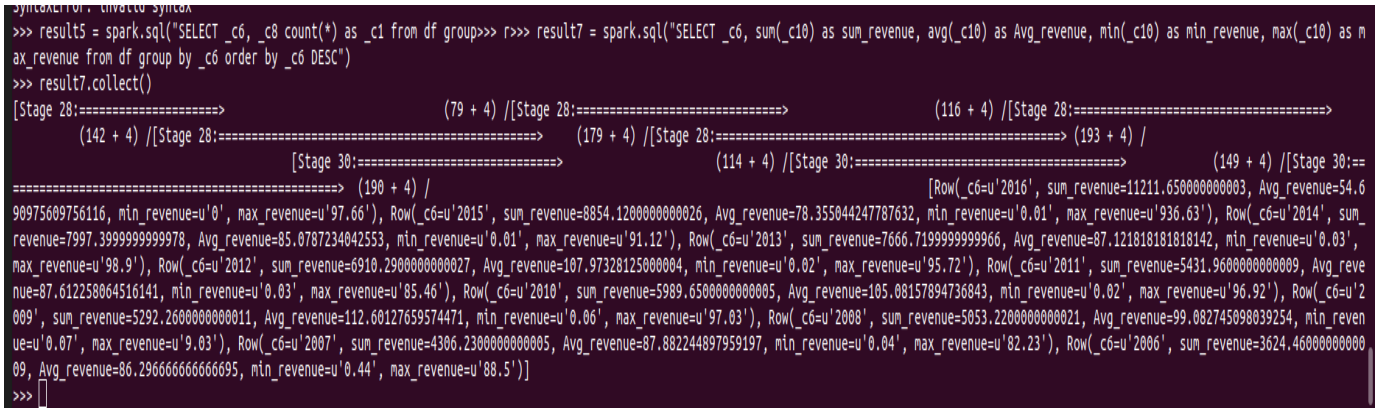
5.

result7 = spark.sql("SELECT _c6, sum(_c10) as sum_revenue, avg(_c10) as Avg_revenue, min(_c10) as min_revenue, max(_c10) as max_revenue from df group by _c6 order by _c6 DESC")

OUTPUT:

>>> result7.collect()

[Stage 28:====================> (79 + 4) /[Stage
28:==============================> (116 + 4) /[Stage
28:======================================> (142 + 4) /[Stage
28:==> (179 + 4) /[Stage
28:==> (193 + 4) /
[Stage 30:==============================> (114 + 4) /[Stage
30:======================================> (149 + 4) /[Stage
30:==> (190 + 4) /
[Row(_c6=u'2016', sum_revenue=11211.650000000003, Avg_revenue=54.690975609756116,
min_revenue=u'0', max_revenue=u'97.66'), Row(_c6=u'2015', sum_revenue=8854.1200000000026,
Avg_revenue=78.355044247787632, min_revenue=u'0.01', max_revenue=u'936.63'),
Row(_c6=u'2014', sum_revenue=7997.3999999999978, Avg_revenue=85.0787234042553,
min_revenue=u'0.01', max_revenue=u'91.12'), Row(_c6=u'2013',
sum_revenue=7666.7199999999966, Avg_revenue=87.121818181818142, min_revenue=u'0.03',
max_revenue=u'98.9'), Row(_c6=u'2012', sum_revenue=6910.2900000000027,
Avg_revenue=107.97328125000004, min_revenue=u'0.02', max_revenue=u'95.72'),
Row(_c6=u'2011', sum_revenue=5431.9600000000009, Avg_revenue=87.612258064516141,
min_revenue=u'0.03', max_revenue=u'85.46'), Row(_c6=u'2010',
sum_revenue=5989.6500000000005, Avg_revenue=105.08157894736843, min_revenue=u'0.02',
max_revenue=u'96.92'), Row(_c6=u'2009', sum_revenue=5292.2600000000011,
Avg_revenue=112.60127659574471, min_revenue=u'0.06', max_revenue=u'97.03'),
Row(_c6=u'2008', sum_revenue=5053.2200000000021, Avg_revenue=99.082745098039254,
min_revenue=u'0.07', max_revenue=u'9.03'), Row(_c6=u'2007',
sum_revenue=4306.2300000000005, Avg_revenue=87.882244897959197, min_revenue=u'0.04',
max_revenue=u'82.23'), Row(_c6=u'2006', sum_revenue=3624.4600000000009,
Avg_revenue=86.296666666666695, min_revenue=u'0.44', max_revenue=u'88.5')]



6.

result8 = spark.sql("SELECT _c4, avg(_c10) as Avg_revenue from df group by _c4 HAVING Avg_revenue > 100 order by Avg_revenue DESC")

OUTPUT:

>>> result8.collect()

[Stage 33:==================================> (126 + 4) [Stage 33:===> (177 + 5) [Stage 35:======================> (80 + 5) [Stage 35:==============================> (118 + 4) [Stage 35:==> (157 + 4)
[Row(_c4=u'James Cameron', Avg_revenue=760.50999999999999), Row(_c4=u'Colin Trevorrow', Avg_revenue=652.17999999999995), Row(_c4=u'Joss Whedon', Avg_revenue=541.13499999999999), Row(_c4=u'Lee Unkrich', Avg_revenue=414.98000000000002), Row(_c4=u'Gary Ross', Avg_revenue=408.0), Row(_c4=u'Chris Buck', Avg_revenue=400.74000000000001), Row(_c4=u'Chris Renaud', Avg_revenue=368.31), Row(_c4=u'Gareth Edwards', Avg_revenue=366.41499999999996), Row(_c4=u'Tim Miller', Avg_revenue=363.01999999999998), Row(_c4=u'Byron Howard', Avg_revenue=341.25999999999999), Row(_c4=u'J.J. Abrams', Avg_revenue=336.68999999999994), Row(_c4=u'Kyle Balda', Avg_revenue=336.02999999999997), Row(_c4=u'Anthony Russo', Avg_revenue=333.91499999999996), Row(_c4=u'Francis Lawrence', Avg_revenue=324.95249999999999), Row(_c4=u'Pete Docter', Avg_revenue=324.71500000000003), Row(_c4=u'Pierre Coffin', Avg_revenue=309.77499999999998), Row(_c4=u'Christopher Nolan', Avg_revenue=303.01800000000003), Row(_c4=u'David Slade', Avg_revenue=300.51999999999998), Row(_c4=u'Bill Condon', Avg_revenue=286.78999999999996), Row(_c4=u'Sam Raimi', Avg_revenue=285.71499999999997), Row(_c4=u'David Yates', Avg_revenue=271.75166666666667), Row(_c4=u'Christophe Lourdelet', Avg_revenue=270.31999999999999), Row(_c4=u'Dan Scanlon', Avg_revenue=268.49000000000001), Row(_c4=u'Andrew Stanton', Avg_revenue=261.05333333333334), Row(_c4=u'Jon Favreau', Avg_revenue=256.39999999999998), Row(_c4=u'Robert Stromberg', Avg_revenue=241.41), Row(_c4=u'Mark Andrews', Avg_revenue=237.28), Row(_c4=u'Michael Bay', Avg_revenue=236.88666666666666), Row(_c4=u'Shane Black', Avg_revenue=222.62), Row(_c4=u'Don Hall', Avg_revenue=222.49000000000001), Row(_c4=u'John Lasseter', Avg_revenue=217.75), Row(_c4=u'Mark Osborne', Avg_revenue=215.40000000000001), Row(_c4=u'Peter Jackson', Avg_revenue=215.11250000000001), Row(_c4=u'Gore Verbinski', Avg_revenue=207.45499999999998), Row(_c4=u'Nathan Greno', Avg_revenue=200.81), Row(_c4=u'Dean DeBlois', Avg_revenue=197.19499999999999), Row(_c4=u'Bryan Singer', Avg_revenue=196.43666666666664), Row(_c4=u'Phil Lord', Avg_revenue=195.9433333333333), Row(_c4=u'Zack Snyder', Avg_revenue=195.148), Row(_c4=u'Catherine Hardwicke', Avg_revenue=191.44999999999999), Row(_c4=u'Rich Moore', Avg_revenue=189.41), Row(_c4=u'Marc Forster', Avg_revenue=185.36000000000001), Row(_c4=u'Rob Marshall', Avg_revenue=184.53), Row(_c4=u'Tim Johnson', Avg_revenue=177.34), Row(_c4=u'Joe Johnston', Avg_revenue=176.63999999999999), Row(_c4=u'Ron Clements', Avg_revenue=176.56),

Row(_c4=u'George Miller', Avg_revenue=175.81), Row(_c4=u'Sam Mendes', Avg_revenue=175.77000000000001), Row(_c4=u'Brad Bird', Avg_revenue=169.74000000000001), Row(_c4=u'Genndy Tartakovsky', Avg_revenue=169.69), Row(_c4=u'Theodore Melfi', Avg_revenue=169.27000000000001), Row(_c4=u'Sam Taylor-Johnson', Avg_revenue=166.15000000000001), Row(_c4=u'Marc Webb', Avg_revenue=165.75666666666666), Row(_c4=u'Paul Greengrass', Avg_revenue=165.46666666666667), Row(_c4=u'Justin Lin', Avg_revenue=164.958), Row(_c4=u'Clint Eastwood', Avg_revenue=164.7475), Row(_c4=u'James Wan', Avg_revenue=160.96749999999997), Row(_c4=u'Todd Phillips', Avg_revenue=160.16499999999999), Row(_c4=u'Steven Spielberg', Avg_revenue=156.7475), Row(_c4=u'Brad Peyton', Avg_revenue=155.18000000000001), Row(_c4=u'Rupert Sanders', Avg_revenue=155.11000000000001), Row(_c4=u'Alfonso Cuar\xf3n', Avg_revenue=154.685), Row(_c4=u'Walt Dohrn', Avg_revenue=153.69), Row(_c4=u'Brett Ratner', Avg_revenue=153.50999999999999), Row(_c4=u'David Ayer', Avg_revenue=150.56999999999999), Row(_c4=u'Stephen Sommers', Avg_revenue=150.16999999999999), Row(_c4=u'Peyton Reed', Avg_revenue=149.435), Row(_c4=u'Tom McGrath', Avg_revenue=148.34), Row(_c4=u'Alan Taylor', Avg_revenue=148.04500000000002), Row(_c4=u'Kenneth Branagh', Avg_revenue=144.24000000000001), Row(_c4=u'Phyllida Lloyd', Avg_revenue=143.69999999999999), Row(_c4=u'Carlos Saldanha', Avg_revenue=143.62), Row(_c4=u'Alessandro Carloni', Avg_revenue=143.52000000000001), Row(_c4=u'Paul Feig', Avg_revenue=141.95500000000001), Row(_c4=u'Martin Campbell', Avg_revenue=141.80000000000001), Row(_c4=u'Scott Derrickson', Avg_revenue=140.32999999999998), Row(_c4=u'Rawson Marshall Thurber', Avg_revenue=138.875), Row(_c4=u'Tim Burton', Avg_revenue=138.505), Row(_c4=u'Jonathan Liebesman', Avg_revenue=137.255), Row(_c4=u'James Mangold', Avg_revenue=132.55000000000001), Row(_c4=u'John Stockwell', Avg_revenue=131.56), Row(_c4=u'Joseph Kosinski', Avg_revenue=130.535), Row(_c4=u'Kevin Lima', Avg_revenue=127.70999999999999), Row(_c4=u'Christopher McQuarrie', Avg_revenue=126.84), Row(_c4=u'Josh Boone', Avg_revenue=124.87), Row(_c4=u'Dennis Dugan', Avg_revenue=124.67999999999999), Row(_c4=u'Michael Patrick King', Avg_revenue=123.98499999999999), Row(_c4=u'Tate Taylor', Avg_revenue=122.51000000000001), Row(_c4=u'Seth Gordon', Avg_revenue=117.53), Row(_c4=u'John Lee Hancock', Avg_revenue=117.34666666666668), Row(_c4=u'F. Gary Gray', Avg_revenue=117.185), Row(_c4=u'Gabriele Muccino', Avg_revenue=116.27000000000001), Row(_c4=u'Mark Steven Johnson', Avg_revenue=115.8), Row(_c4=u'Angelina Jolie', Avg_revenue=115.59999999999999), Row(_c4=u'Anne Fletcher', Avg_revenue=114.60999999999999), Row(_c4=u'Seth MacFarlane', Avg_revenue=114.17), Row(_c4=u'Roland Emmerich', Avg_revenue=114.11666666666667), Row(_c4=u'James Gunn', Avg_revenue=113.73999999999999), Row(_c4=u'Jason Reitman', Avg_revenue=113.65000000000001), Row(_c4=u'Tony Gilroy', Avg_revenue=113.17), Row(_c4=u'Jon Lucas', Avg_revenue=113.08), Row(_c4=u'Quentin Tarantino', Avg_revenue=112.48), Row(_c4=u'Guy Ritchie', Avg_revenue=111.74250000000001), Row(_c4=u'Ryan Coogler', Avg_revenue=109.70999999999999), Row(_c4=u'Adam McKay', Avg_revenue=109.535), Row(_c4=u'Peter Billingsley', Avg_revenue=109.18000000000001), Row(_c4=u'Judd Apatow', Avg_revenue=108.75333333333333), Row(_c4=u'David O. Russell', Avg_revenue=108.05500000000001), Row(_c4=u'Clay Kaytis', Avg_revenue=107.51000000000001), Row(_c4=u'Noam Murro', Avg_revenue=106.37), Row(_c4=u'Louis Leterrier', Avg_revenue=105.56750000000001), Row(_c4=u'David Fincher', Avg_revenue=105.54400000000001), Row(_c4=u'Rupert Wyatt', Avg_revenue=105.185), Row(_c4=u'Harald Zwart', Avg_revenue=103.88), Row(_c4=u'Sean Anders',

Avg_revenue=102.36499999999999), Row(_c4=u'Peter Berg', Avg_revenue=102.26599999999999), Row(_c4=u'Steven Soderbergh', Avg_revenue=102.16333333333334), Row(_c4=u'Matt Reeves', Avg_revenue=100.23333333333333), Row(_c4=u'Tom Hooper', Avg_revenue=100.09666666666668)]

7.

result9 = spark.sql("SELECT _c6, sum(_c8) as sum_rating, avg(_c8) as avg_rating, min(_c8) as min_ratings, max(_c8) as max_rating from df group by _c6 order by _c6 DESC")

OUTPUT:

>>> result9.collect()

[Stage 38:==> (160 + 4[Stage 38:===>(197 + 3
[Stage 40:============================> (100 + 4[Stage 40:======================================> (146 + 4[Stage 40:==> (191 + 4
[Row(_c6=u'2016', sum_rating=1911.7, avg_rating=6.436700336700337, min_ratings=u'2.7', max_rating=u'8.8'), Row(_c6=u'2015', sum_rating=838.50000000000023, avg_rating=6.602362204724411, min_ratings=u'3.5', max_rating=u'8.3'), Row(_c6=u'2014', sum_rating=670.10000000000002, avg_rating=6.8377551020408163, min_ratings=u'5.1', max_rating=u'8.6'), Row(_c6=u'2013', sum_rating=619.89999999999986, avg_rating=6.812087912087911, min_ratings=u'4.3', max_rating=u'8.2'), Row(_c6=u'2012', sum_rating=443.19999999999993, avg_rating=6.9249999999999989, min_ratings=u'5.3', max_rating=u'8.5'), Row(_c6=u'2011', sum_rating=430.80000000000007, avg_rating=6.8380952380952396, min_ratings=u'4.9', max_rating=u'8.6'), Row(_c6=u'2010', sum_rating=409.60000000000008, avg_rating=6.826666666666668, min_ratings=u'4.2', max_rating=u'8.8'), Row(_c6=u'2009', sum_rating=355.0, avg_rating=6.9607843137254903, min_ratings=u'2.7', max_rating=u'8.4'), Row(_c6=u'2008', sum_rating=352.79999999999995, avg_rating=6.7846153846153836, min_ratings=u'1.9', max_rating=u'9'), Row(_c6=u'2007', sum_rating=378.09999999999991, avg_rating=7.1339622641509415, min_ratings=u'4.7', max_rating=u'8.5'), Row(_c6=u'2006', sum_rating=313.50000000000006, avg_rating=7.1250000000000009, min_ratings=u'5.6', max_rating=u'8.5')]