Leverage the power of SQL Analytical functions in Business Intelligence and Analytics

Viana Rumao, Asher Dmello

Abstract— It is said that the world's most powerful resource is no longer oil, but data. As rightly said by E.O Wilson, we are drowning in data, but striving for wisdom. Big Data, Analytics, Business Intelligence, Data Science require processing huge amounts of data in various and complex ways using a vast array of statistical methods and tools. Market increasingly needs manpower with both databases and data warehouses technological skills and statistical competencies to decipher the business patterns and trends hidden in the mountains of data[1] This paper presents an overview of SQL and its role in business intelligence and data analytics. The paper further focuses on the power of analytical functions in SQL. We experimentally prove the efficiency of the same using Spark-SQL in Python.

Index Terms— SQL, Analytical Functions, Business Intelligence, Analytics, Python, Spark, Data Science

1 Introduction

In today's world, the revolution in technology and business has bought in the large data storage from phones, computers and various other processes. This has indirectly made us store large amount of data as data used is required to analyse and interpret decisions and statistics in the field of medicine and sports and various upcoming fields. People, businesses, and devices have all become data factories that are pumping out incredible amounts of information to the web each day.[2][3]. The challenge to manage and store this data and get meaningful insights out of the same has given rise to Business Intelligence. We discuss the significance of the field of BI and how Analytical functions in SQL play a pivotal role in the same.

2 Significance of Business Intelligence and Analytics

BI is a concept that usually involves the delivery and integration of relevant and useful business information in an organization. Companies use BI to detect significant events and identify/monitor business trends to adapt quickly to their changing environment and a scenario.

In today's world, we use business intelligence for better decision making and to interpret, collect and analyse data. Business intelligence involves activities that include data mining, online analytical processing, querying and reporting. Business intelligence involves making accurate business decisions and correcting the mistakes made in the previous business decisions.

Business analytics is a methodology or tool to make a sound commercial decision. It is used to analyse primary and secondary data. It is used as predictive analysis which is based on past predictive analytics. It is also used for constructing a business model or a project based upon the past models.[4][5]

3 SQL for data storage, retrieval and analytics

According to Jacobs (2009), Big Data should be defined at any point in time as data whose size forces us to look beyond the tried and true methods that are prevalent at that time, whereas for Cuzzocrea et al. (2011) Big Data refers to enormous amounts of unstructured data produced by high-performance applications falling in a wide and heterogeneous family of application scenarios: from scientific computing applications to social networks, from e-government applications to medical information systems, and so forth.

Big volumes of data, but small analytics usually involves using regular SQL queries (SELECT with MIN, MAX, SUM, COUNT, AVG, GROUP BY, HAVING functions and clauses) on large datasets. All types of SQL (relational) databases, commercial (Oracle, IBM DB2, Microsoft SQL Server) or open-source (PostgreSQL, MySQL) could be platforms/tools for this type of processing. Big data analytics on big volumes of data requires combining ETL (Extract-Transform-Load) tools with statistical packager. Big data analytics denote regression, data mining, machine learning and other types of more complex processing. Data could be extracted from various data sources using SQL queries and/or ETL tools. Complex analyses require packages such as SPSS, R, SAS, etc. and sometimes a good deal of coding.[1] An efficient but a not-so-popular technique to handle complex analysis is the use of analytical functions in SQL. In the next section, we unleash the power and efficiency of the same

4 Analytical Functions in SQL

For an IT professional getting into data warehousing (DWH) or business intelligence, the need is to go beyond plain SQL and start digging into the analytical functions (also known as window functions, windowing functions, or OVER clauses). We demonstrate SQL Analytical functions over Spark framework using the pySpark library in Python on Jupyter notebook.

- **Partition and Over Clause:**

The analytical function consists of a function with an over keyword and an optional list of columns for ordering. The ordering indicates criteria for function evaluation, not the final ordering of results.

- **RANK()**

RANK is a simple analytical function that does not take any

column list arguments. It returns a number or a rank based on the ordering of the rows; the ordering is based on a defined condition.

- **DENSE_RANK()**

DENSE RANK works like RANK in that it needs no additional arguments and it ranks items in descending or ascending order. The only difference is that DENSE RANK does not allow "gaps" between groups.

- **LAG()**

LAG is the opposite of LEAD. We can even implement LAG using LEAD and vice versa. The difference is in the direction we look for the offset value.

- **LEAD()**

LEAD returns an offset (incrementally increased) value of an argument column. The offset amount can be defined in the code; its default amount is "1". The new value is returned in the same row.

- **Windowing**

The window concept is used extensively with analytical functions. A window is a collection of rows in which an analytical function is calculated.

We tie window concepts to changes in numeric variables. Windows are typically defined by time intervals, like years or months.

Windows can be specified as units of physical rows using the ROWS keyword, or they can be defined in logical rows using the RANGE keyword. Physical windows are specified by absolute values of rows we offset before and after the current row. Logical windows are specified by values, such as a number of days. Numeric variables are calculated in moving windows so that the variables can be compared across windows.

Windows can be cumulative or sliding, as we will see in the coming examples.

Examples of physical-row windows are:

- ROWS UNBOUNDED PRECEDING – The window is the current row and every preceding row.
- ROWS 2 PRECEDING – The window is the current row and 2 preceding physical rows.
- ROWS 2 FOLLOWING – The window is the current row and 2 following physical rows.

Please refer appendix 1 for examples.

5 THE POWER OF ANALYTICAL FUNCTIONS

As a part of the experiment, we execute a query on 1000 rows employee dataset using analytical functions and the execute the same using joins. We see that the query with analytical functions took 4.35 seconds to execute, where joins the query executed in 13.592 seconds. The analytical functions query was shorter and a one-step process. The joins query was a complex 2 step process. This explains the power of analytical functions in SQL. Please refer appendix 2 for implementation screenshots.

4 CONCLUSION AND FUTURE WORK

The paper provides an overview of the field of BI and the role of SQL analytical functions. We prove their efficiency compare to normal SQL as well. We can further explore on how Analytical functions can revolutionize the field of BI.

REFERENCES

[1] Marin Fotache, Catalin Strimbei, SQL and Data Analysis. Some Implications for Data Analysts and Higher Education, Procedia Economics and Finance, Volume 20, 2015,

[2] https://www.cleverism.com/what-is-big-data-understanding-large-amount-data/

[3] https://blog.microfocus.com/how-much-data-is-created-on-the-internet-each-day/

[4] https://www.managementstudyguide.com/business-analytics.htm

[5] https://financesonline.com/purpose-business-intelligence-business/

[6] Dataset: http://eforexcel.com/wp/downloads-16-sample-csv-files-data-sets-for-testing/

[7] http://www.vertabelo.com/blog/technical-articles/oracle-sql-analytical-functions-for-beginners-a-gentle-introduction-to-common-sql-window-functions

Appendix 1

Over and Partition Clause

Compare an employee's salary against the average salary of all employees joined in the same year

```
result3 = sqlContext.sql("""Select Emp_ID,First_Name,Last_Name,Gender,Date_of_Joining, Salary  , avg(Salary)
                 over  (partition by Year_of_Joining) as avg_sal
                from EMP
                """)
```

```
result3.show(10)
```

```
+------+----------+-----------+------+---------------+--------+------------------+
|Emp_ID|First_Name|  Last_Name|Gender|Date_of_Joining|  Salary|           avg_sal|
+------+----------+-----------+------+---------------+--------+------------------+
|330816|   Jasmine|    Freeman|     F|       2/1/1983|154216.0| 97408.33333333333|
|835545|     Janie|  Velasquez|     F|      5/14/1983| 90653.0| 97408.33333333333|
|909541|   Clement|      Myers|     M|      6/20/1983| 47356.0| 97408.33333333333|
|269523|    Kendra| Villarreal|     F|      3/30/2007|144517.0|127131.88888888889|
|248449|   Carlene|       Cain|     F|      6/25/2007|107647.0|127131.88888888889|
|749765|     Brian|Fitzpatrick|     M|      11/9/2007|151448.0|127131.88888888889|
|178602|  Kristina|    Mcgowan|     F|      5/11/2007|185016.0|127131.88888888889|
|768457|     Robby|     Jordan|     M|      3/12/2007|119793.0|127131.88888888889|
|556491|     Larry|       Tran|     M|      3/15/2007|129203.0|127131.88888888889|
|505780|     Ester|    Houston|     F|       7/1/2007| 65748.0|127131.88888888889|
+------+----------+-----------+------+---------------+--------+------------------+
only showing top 10 rows
```

Rank()

Rank the employees with respect to their salaries in highest-paid-first order. Gaps Allowed

```
result4 = sqlContext.sql("""Select Emp_ID,First_Name,Last_Name,Gender,Date_of_Birth, Date_of_Joining, Salary  ,
                 RANK () over  (order by Salary desc) as Rank_Sal
                from EMP
                """)
```

```
result4.show()
```

```
+------+----------+---------+------+-------------+---------------+--------+--------+
|Emp_ID|First_Name|Last_Name|Gender|Date_of_Birth|Date_of_Joining|  Salary|Rank_Sal|
+------+----------+---------+------+-------------+---------------+--------+--------+
|601071|       Liz|    Frank|     F|    3/15/1986|      2/14/2009|199943.0|       1|
|887692|    Joseph|   Tucker|     M|    10/3/1979|       8/22/2006|199923.0|       2|
|475176|    Harley|   Murphy|     M|    2/23/1967|      10/25/2008|199411.0|       3|
|326407|    Sophie|     Bass|     F|    1/16/1990|      10/13/2011|199238.0|       4|
|982922|Bernadette|    Glass|     F|    5/24/1994|       1/29/2016|199184.0|       5|
|509810|   Susanne|   Joseph|     F|   12/25/1963|       7/28/1985|199131.0|       6|
|752997|     Lelia|   Sawyer|     F|     5/5/1980|       7/18/2015|199120.0|       7|
|217117|      Edna| Petersen|     F|    4/29/1958|        7/7/2000|198888.0|       8|
|994907|    Lilian|    Stein|     F|    6/23/1996|       6/23/2017|198733.0|       9|
|754091|  Kimberly|    Cantu|     F|    10/7/1976|       10/4/2002|198580.0|      10|
|159472|     Moses|     Reid|     M|   12/22/1964|       8/14/1999|198160.0|      11|
|662168|    Marlon| Whitaker|     M|    4/10/1982|       4/19/2008|197971.0|      12|
|953967|   Cornell|   Daniel|     M|    5/27/1960|      10/31/2004|197951.0|      13|
|408036|    Brandy|     Carr|     F|   11/21/1976|       9/21/2008|197345.0|      14|
|479754|   Shannon|Singleton|     F|    1/27/1960|       3/23/1987|197109.0|      15|
|780963|      Jami| Cantrell|     F|    3/30/1986|       4/20/2007|196942.0|      16|
|846178|     Lilia|  Collins|     F|     6/7/1980|        2/4/2004|196782.0|      17|
|223379|    Mathew|     Duke|     M|    1/17/1993|        9/6/2015|196685.0|      18|
|287114|    Joanna|  Perkins|     F|    8/10/1959|        9/9/1992|196619.0|      19|
|670526|    Adrian|   Dunlap|     F|    1/16/1985|       8/26/2013|196585.0|      20|
+------+----------+---------+------+-------------+---------------+--------+--------+
only showing top 20 rows
```

DENSE_RANK()

Rank the employees with respect to their number of years with the company. No gaps allowed

```
from time import time
t0 = time()

result5 = sqlContext.sql("""Select Emp_ID,First_Name,Last_Name,Gender,Date_of_Birth, Date_of_Joining, Salary, Age_in_Company,
                 DENSE_RANK () over  (order by Age_in_Company desc) as Rank_Exp
                from EMP
                """)

result5.show(10)

tt = time() - t0
print("Query performed in {} seconds".format(round(tt,3)))
```

```
+------+----------+---------+------+-------------+---------------+--------+--------------+--------+
|Emp_ID|First_Name|Last_Name|Gender|Date_of_Birth|Date_of_Joining|  Salary|Age_in_Company|Rank_Exp|
+------+----------+---------+------+-------------+---------------+--------+--------------+--------+
|155916|   Rosanne|  Mendoza|     F|    9/22/1958|      1/11/1980|120595.0|         37.57|       1|
|669618|   Phyllis|   Dudley|     F|    11/2/1957|      1/17/1980|105069.0|         37.55|       2|
|993099|     Ricky|   Snider|     M|    6/29/1959|      2/12/1981| 57113.0|         36.48|       3|
|273325|     Jayne|   Thomas|     F|     1/9/1960|       3/5/1981|162559.0|         36.42|       4|
|222807|    Janice|     Berg|     F|    5/25/1958|      11/6/1981|185898.0|         35.75|       5|
|709899|     Clair|   Morris|     M|     7/7/1958|     12/13/1981|155562.0|         35.65|       6|
|678436|   Theresa|   Bender|     F|    2/27/1959|      2/27/1982|112711.0|         35.44|       7|
|855670|    Adrian|   Robles|     F|     6/8/1958|     10/14/1982|166464.0|         34.81|       8|
|330816|   Jasmine|  Freeman|     F|    8/31/1961|       2/1/1983|154216.0|         34.51|       9|
|835545|     Janie|Velasquez|     F|    4/19/1958|       5/14/1983| 90653.0|         34.23|      10|
+------+----------+---------+------+-------------+---------------+--------+--------------+--------+
only showing top 10 rows

Query performed in 5.867 seconds
```

Lag()

Compare employee's experience in the company to the person paid lower than him

```
]: result6 = sqlContext.sql("""Select Emp_ID,First_Name,Last_Name,Gender,Date_of_Birth, Date_of_Joining, Salary,Age_in_Company,
                 lag(Age_in_Company,1,null) OVER (ORDER BY Salary ) as lag_exp
                from EMP
                """)

   result6.show()
```

```
+------+----------+-----------+------+-------------+---------------+-------+--------------+-------+
|Emp_ID|First_Name|  Last_Name|Gender|Date_of_Birth|Date_of_Joining| Salary|Age_in_Company|lag_exp|
+------+----------+-----------+------+-------------+---------------+-------+--------------+-------+
|176214| Josephine|      Pitts|     F|   12/29/1972|      8/30/2008|40040.0|          8.92|   null|
|712170|  Theodore|       Dale|     M|   12/17/1994|       7/1/2017|40096.0|          0.07|   8.92|
|414557|     Corey|    Harrell|     M|    7/23/1971|      5/16/2003|40261.0|         14.21|   0.07|
|620634|     Janna|    Hopkins|     F|   11/21/1987|     12/21/2011|40265.0|          5.61|  14.21|
|220061|   Norbert|     Joyner|     M|    4/18/1966|     11/24/2008|40749.0|          8.68|   5.61|
|648744|    Edmund|Blankenship|     M|   11/11/1965|      3/22/2017|40783.0|          0.35|   8.68|
|278391|      Jami|  Velasquez|     F|    7/23/1992|      8/19/2016|40883.0|          0.94|   0.35|
|795934|   Melinda|      Lopez|     F|    9/15/1984|     11/22/2005|41287.0|         11.69|   0.94|
|430633|    Sondra|      Reyes|     F|    4/18/1993|       7/6/2016|41680.0|          1.06|  11.69|
|900863|    Barbra|      Hobbs|     F|     5/9/1991|      10/6/2013|41746.0|          3.81|   1.06|
|144114|     Nancy|  Gillespie|     F|    4/26/1980|       2/3/2011|41796.0|          6.48|   3.81|
|125371|    Lorrie|      Rocha|     F|    1/16/1995|     12/21/2016|42245.0|           0.6|   6.48|
|387407|    Damian|    Ellison|     M|    4/26/1962|       9/2/2010|42547.0|          6.91|    0.6|
|654747|     Edwin|      Brock|     M|     8/5/1994|     10/30/2015|42553.0|          1.75|   6.91|
|481137|      Gena|     Holman|     F|     5/5/1989|      6/18/2013|42626.0|          4.11|   1.75|
|597006|     Hazel|       Case|     F|     8/3/1962|      10/7/1987|42726.0|         29.83|   4.11|
|337955|  Jeremiah|     Dotson|     M|    7/16/1988|      1/21/2015|42840.0|          2.52|  29.83|
|567269|       Fay|Villarreal|     F|     4/4/1989|     11/29/2012|42988.0|          4.66|   2.52|
|371751|      Thad|       Luna|     M|    2/22/1969|       4/2/2007|43176.0|         10.33|   4.66|
|839558|     Teddy|    Merritt|     M|    6/10/1977|      5/15/2010|43192.0|          7.21|  10.33|
+------+----------+-----------+------+-------------+---------------+-------+--------------+-------+
only showing top 20 rows
```

Lead()

Compare an employee's salary to the employee joined after him

```
result7 = sqlContext.sql("""Select Emp_ID,First_Name,Last_Name,Gender,Date_of_Birth, Date_of_Joining, Salary, Age_in_Company,
                lead(Salary,1,null) OVER (ORDER BY Age_in_Company desc) as Lead_Salary
                from EMP ORDER BY Age_in_Company desc
                """)

result7.show()
```

```
+------+----------+---------+------+-------------+---------------+--------+--------------+-----------+
|Emp_ID|First_Name|Last_Name|Gender|Date_of_Birth|Date_of_Joining|  Salary|Age_in_Company|Lead_Salary|
+------+----------+---------+------+-------------+---------------+--------+--------------+-----------+
|155916|   Rosanne|  Mendoza|     F|    9/22/1958|      1/11/1980|120595.0|         37.57|   105069.0|
|669618|   Phyllis|   Dudley|     F|    11/2/1957|      1/17/1980|105069.0|         37.55|    57113.0|
|993099|     Ricky|   Snider|     M|    6/29/1959|      2/12/1981| 57113.0|         36.48|   162559.0|
|273325|     Jayne|   Thomas|     F|     1/9/1960|       3/5/1981|162559.0|         36.42|   185898.0|
|222807|    Janice|     Berg|     F|    5/25/1958|      11/6/1981|185898.0|         35.75|   155562.0|
|709899|     Clair|   Morris|     M|     7/7/1958|     12/13/1981|155562.0|         35.65|   112711.0|
|678436|   Theresa|   Bender|     F|    2/27/1959|      2/27/1982|112711.0|         35.44|   166464.0|
|855670|    Adrian|   Robles|     F|     6/8/1958|     10/14/1982|166464.0|         34.81|   154216.0|
|330816|   Jasmine|  Freeman|     F|    8/31/1961|       2/1/1983|154216.0|         34.51|    90653.0|
|835545|     Janie|Velasquez|     F|    4/19/1958|       5/14/1983| 90653.0|         34.23|    47356.0|
|909541|   Clement|    Myers|     M|   11/26/1958|       6/20/1983| 47356.0|         34.13|    96974.0|
|901588|   Mariana|   Mcneil|     F|    3/25/1961|       2/18/1984| 96974.0|         33.46|    91713.0|
|430843|    Karina|    Owens|     F|   10/30/1957|       6/30/1984| 91713.0|          33.1|   125299.0|
|874971|     Roman|    Clark|     M|   12/25/1962|        1/8/1985|125299.0|         32.57|    77106.0|
|864339|     Tania|      May|     F|    4/3/1960|       1/16/1985| 77106.0|         32.55|    65123.0|
|768500|      Jami|   Golden|     F|   11/21/1960|       1/21/1985| 65123.0|         32.54|   144108.0|
|149904|    Leanne|  Mcmahon|     F|    1/11/1964|        5/8/1985|144108.0|         32.24|   138878.0|
|205347|    Frieda|   Hebert|     F|    8/23/1957|       6/11/1985|138878.0|         32.15|   182540.0|
|280326|   Agustin|   Gentry|     M|   11/27/1962|       6/16/1985|182540.0|         32.14|   199131.0|
|509810|   Susanne|   Joseph|     F|   12/25/1963|       7/28/1985|199131.0|         32.02|    53313.0|
+------+----------+---------+------+-------------+---------------+--------+--------------+-----------+
only showing top 20 rows
```

Windowing

Compare an employee's salary to the average of salaries of 2 employees joined prior to him

```
result7 = sqlContext.sql("""Select Emp_ID,First_Name,Last_Name,Gender,Date_of_Birth, Date_of_Joining, Salary, Age_in_Company,
                Avg(Salary) OVER (ORDER BY Age_in_Company desc rows 2 preceding) as Avg_2_Salary
                from EMP order by Age_in_Company desc
                """)

result7.show()
```

```
+------+----------+---------+------+-------------+---------------+--------+--------------+------------------+
|Emp_ID|First_Name|Last_Name|Gender|Date_of_Birth|Date_of_Joining|  Salary|Age_in_Company|      Avg_2_Salary|
+------+----------+---------+------+-------------+---------------+--------+--------------+------------------+
|155916|   Rosanne|  Mendoza|     F|    9/22/1958|      1/11/1980|120595.0|         37.57|          120595.0|
|669618|   Phyllis|   Dudley|     F|    11/2/1957|      1/17/1980|105069.0|         37.55|          112832.0|
|993099|     Ricky|   Snider|     M|    6/29/1959|      2/12/1981| 57113.0|         36.48|           94259.0|
|273325|     Jayne|   Thomas|     F|     1/9/1960|       3/5/1981|162559.0|         36.42|          108247.0|
|222807|    Janice|     Berg|     F|    5/25/1958|      11/6/1981|185898.0|         35.75|          135190.0|
|709899|     Clair|   Morris|     M|     7/7/1958|     12/13/1981|155562.0|         35.65|168006.33333333334|
|678436|   Theresa|   Bender|     F|    2/27/1959|      2/27/1982|112711.0|         35.44|151390.33333333334|
|855670|    Adrian|   Robles|     F|     6/8/1958|     10/14/1982|166464.0|         34.81|144912.33333333334|
|330816|   Jasmine|  Freeman|     F|    8/31/1961|       2/1/1983|154216.0|         34.51|144463.66666666666|
|835545|     Janie|Velasquez|     F|    4/19/1958|       5/14/1983| 90653.0|         34.23|          137111.0|
|909541|   Clement|    Myers|     M|   11/26/1958|       6/20/1983| 47356.0|         34.13| 97408.33333333333|
|901588|   Mariana|   Mcneil|     F|    3/25/1961|       2/18/1984| 96974.0|         33.46| 78327.66666666667|
|430843|    Karina|    Owens|     F|   10/30/1957|       6/30/1984| 91713.0|          33.1|           78681.0|
|874971|     Roman|    Clark|     M|   12/25/1962|        1/8/1985|125299.0|         32.57|          104662.0|
|864339|     Tania|      May|     F|    4/3/1960|       1/16/1985| 77106.0|         32.55| 98039.33333333333|
|768500|      Jami|   Golden|     F|   11/21/1960|       1/21/1985| 65123.0|         32.54|           89176.0|
|149904|    Leanne|  Mcmahon|     F|    1/11/1964|        5/8/1985|144108.0|         32.24| 95445.66666666667|
|205347|    Frieda|   Hebert|     F|    8/23/1957|       6/11/1985|138878.0|         32.15|116036.33333333333|
|280326|   Agustin|   Gentry|     M|   11/27/1962|       6/16/1985|182540.0|         32.14|155175.33333333334|
|509810|   Susanne|   Joseph|     F|   12/25/1963|       7/28/1985|199131.0|         32.02|173516.33333333334|
+------+----------+---------+------+-------------+---------------+--------+--------------+------------------+
only showing top 20 rows
```

Appendix 2

Compare an employee's salary against the average salary of all employees joined in the same year

With Analytical functions

```
from time import time
t0 = time()

result8 = sqlContext.sql("""Select Emp_ID,First_Name,Last_Name,Gender,Date_of_Joining, Year_of_Joining, Salary  , avg(Salary)
                 over  (partition by Year_of_Joining) as avg_sal
                from EMP
                """)
result8.show()

tt = time() - t0
print("Query performed in {} seconds".format(round(tt,3)))
```

```
+------+----------+-----------+------+---------------+---------------+--------+------------------+
|Emp_ID|First_Name|  Last_Name|Gender|Date_of_Joining|Year_of_Joining|  Salary|           avg_sal|
+------+----------+-----------+------+---------------+---------------+--------+------------------+
|330816|   Jasmine|    Freeman|     F|       2/1/1983|           1983|154216.0| 97408.33333333333|
|835545|     Janie|  Velasquez|     F|      5/14/1983|           1983| 90653.0| 97408.33333333333|
|909541|   Clement|      Myers|     M|      6/20/1983|           1983| 47356.0| 97408.33333333333|
|269523|    Kendra| Villarreal|     F|      3/30/2007|           2007|144517.0|127131.88888888889|
|248449|   Carlene|       Cain|     F|      6/25/2007|           2007|107647.0|127131.88888888889|
|749765|     Brian|Fitzpatrick|     M|      11/9/2007|           2007|151448.0|127131.88888888889|
|178602|  Kristina|    Mcgowan|     F|      5/11/2007|           2007|185016.0|127131.88888888889|
|768457|     Robby|     Jordan|     M|      3/12/2007|           2007|119793.0|127131.88888888889|
|556491|     Larry|       Tran|     M|      3/15/2007|           2007|129203.0|127131.88888888889|
|505780|     Ester|    Houston|     F|       7/1/2007|           2007| 65748.0|127131.88888888889|
|780963|      Jami|   Cantrell|     F|      4/20/2007|           2007|196942.0|127131.88888888889|
|297502|  Angelina|       Lynn|     F|       3/8/2007|           2007|157805.0|127131.88888888889|
|532285|     Leann|   Melendez|     F|     12/17/2007|           2007| 88889.0|127131.88888888889|
|182719|     Tammy|    Vincent|     F|     11/19/2007|           2007|111372.0|127131.88888888889|
|825433|      Tara|    Goodwin|     F|     11/24/2007|           2007| 84023.0|127131.88888888889|
|234429|  Chrystal|    Swanson|     F|      4/27/2007|           2007|144528.0|127131.88888888889|
|885774|     Miles|      Giles|     M|      2/12/2007|           2007|105720.0|127131.88888888889|
|282678|     Marci|   Alvarado|     F|       9/2/2007|           2007|174673.0|127131.88888888889|
|280781|     Ollie|    William|     F|       9/14/2007|          2007|128589.0|127131.88888888889|
|463579|  Bernardo|    Langley|     M|       8/30/2007|          2007|174027.0|127131.88888888889|
+------+----------+-----------+------+---------------+---------------+--------+------------------+
only showing top 20 rows

Query performed in 4.35 seconds
```

Without Analytical Functions

```
from time import time
t0 = time()

result9 = sqlContext.sql("""Select Year_of_Joining, avg(Salary) as AvSal
                from EMP
                group by  Year_of_Joining
                """)

result9.show()
```

```
+---------------+------------------+
|Year_of_Joining|             AvSal|
+---------------+------------------+
|           1983| 97408.33333333333|
|           2007|127131.88888888889|
|           2014|113365.21153846153|
|           1988|          102412.0|
|           1986|117770.66666666667|
|           2012|117157.83018867925|
|           1991|          146727.4|
|           2016|116287.17647058824|
|           1994|          130503.8|
|           1987|129019.08333333333|
|           1999|118290.23333333334|
|           1997|          131170.4|
|           2010|115296.82926829268|
|           2009|119052.86538461539|
|           2006|119197.07692307692|
|           2017| 120964.9705882353|
|           1998|129771.33333333333|
|           2013|114665.04545454546|
|           1984|           94343.5|
|           2004|127442.83870967742|
+---------------+------------------+
only showing top 20 rows
```

```
result9.registerTempTable("EMP1")
```

```
result10 = sqlContext.sql("""Select a.Emp_ID, a.First_Name, a.Last_Name,a.Gender, a.Date_of_Joining,
               b.Year_of_Joining, a.Salary, b.AvSal
                from EMP a, EMP1 b
                where a.Year_of_Joining = b.Year_of_Joining
                """)

result10.show()
```

```
+------+----------+-----------+------+---------------+---------------+--------+------------------+
|Emp_ID|First_Name|  Last_Name|Gender|Date_of_Joining|Year_of_Joining|  Salary|             AvSal|
+------+----------+-----------+------+---------------+---------------+--------+------------------+
|330816|   Jasmine|    Freeman|     F|       2/1/1983|           1983|154216.0| 97408.33333333333|
|835545|      Janie|  Velasquez|     F|       5/14/1983|           1983| 90653.0| 97408.33333333333|
|909541|   Clement|      Myers|     M|       6/20/1983|           1983| 47356.0| 97408.33333333333|
|269523|    Kendra| Villarreal|     F|       3/30/2007|           2007|144517.0|127131.88888888889|
|248449|   Carlene|       Cain|     F|       6/25/2007|           2007|107647.0|127131.88888888889|
|749765|     Brian|Fitzpatrick|     M|       11/9/2007|           2007|151448.0|127131.88888888889|
|178602|  Kristina|    Mcgowan|     F|       5/11/2007|           2007|185016.0|127131.88888888889|
|768457|     Robby|      Jordan|     M|       3/12/2007|           2007|119793.0|127131.88888888889|
|556491|     Larry|        Tran|     M|       3/15/2007|           2007|129203.0|127131.88888888889|
|505780|     Ester|    Houston|     F|        7/1/2007|           2007| 65748.0|127131.88888888889|
|780963|      Jami|   Cantrell|     F|       4/20/2007|           2007|196942.0|127131.88888888889|
|297502|  Angelina|       Lynn|     F|        3/8/2007|           2007|157805.0|127131.88888888889|
|532285|     Leann|   Melendez|     F|      12/17/2007|           2007| 88889.0|127131.88888888889|
|182719|     Tammy|    Vincent|     F|      11/19/2007|           2007|111372.0|127131.88888888889|
|825433|      Tara|    Goodwin|     F|      11/24/2007|           2007| 84023.0|127131.88888888889|
|234429|  Chrystal|    Swanson|     F|       4/27/2007|           2007|144528.0|127131.88888888889|
|885774|     Miles|      Giles|     M|       2/12/2007|           2007|105720.0|127131.88888888889|
|282678|     Marci|   Alvarado|     F|        9/2/2007|           2007|174673.0|127131.88888888889|
|280781|     Ollie|    William|     F|       9/14/2007|           2007|128589.0|127131.88888888889|
|463579|  Bernardo|    Langley|     M|       8/30/2007|           2007|174027.0|127131.88888888889|
+------+----------+-----------+------+---------------+---------------+--------+------------------+
only showing top 20 rows
```

```
tt = time() - t0
print("Query performed in {} seconds".format(round(tt,3)))
```

```
Query performed in 13.592 seconds
```