# Create data

**Employee.data**

|  |
| --- |
| 1,Mark,Male,5000,USA  2,John,Male,4500,India  3,Pam,Female,5500,USA  4,Sara,Female,4000,India  5,Todd,Male,3500,India  5,Mary,Female,5000,UK  7,Ben,Male,6500,UK  8,Elizabeth,Female,7000,USA  9,Tom,Male,5500,UK  10,Ron,Male,5000,USA |

**Create employee table**

|  |
| --- |
| create table employee (id int,name string,gender string,salary int,country string) Row format delimited fields terminated by ','; |

**Load data**

|  |
| --- |
| load data local inpath '/home/shalaj/employee.data' into table employee; |

|  |
| --- |
| hive> **select \* from employee;**  1 Mark Male 5000 USA  2 John Male 4500 India  3 Pam Female 5500 USA  4 Sara Female 4000 India  5 Todd Male 3500 India  5 Mary Female 5000 UK  7 Ben Male 6500 UK  8 Elizabeth Female 7000 USA  9 Tom Male 5500 UK  10 Ron Male 5000 USA |

# UNION ALL

**Sum of salary by country and gender**

|  |
| --- |
| **select country,gender,sum(salary) from employee group by country,gender;**  India Female 4000  India Male 8000  UK Female 5000  UK Male 12000  USA Female 12500  USA Male 10000 |

**Along with sum of salary by country and gender we also want sum of salary by just by country**

|  |
| --- |
| **select country,gender,sum(salary) from employee group by country,gender**  **UNION ALL**  **select country,NULL as gender,sum(salary) from employee group by country;**  India Female 4000  India Male 8000  UK Female 5000  UK Male 12000  USA Female 12500  USA Male 10000  India NULL 12000  UK NULL 17000  USA NULL 22500 |

**Now we also want to append sum of salary by just gender**

|  |
| --- |
| **select country,gender,sum(salary) from employee group by country,gender**  **UNION ALL**  **select country,NULL as gender,sum(salary) from employee group by country**  **UNION ALL**  **select NULL as country, gender,sum(salary) from employee group by gender;**  India Female 4000  India Male 8000  UK Female 5000  UK Male 12000  USA Female 12500  USA Male 10000  India NULL 12000  UK NULL 17000  USA NULL 22500  NULL Female 21500  NULL Male 30000 |

**Now we want to append grand total**

|  |
| --- |
| **select country,gender,sum(salary) from employee group by country,gender**  **UNION ALL**  **select country,NULL as gender,sum(salary) from employee group by country**  **UNION ALL**  **select NULL as country, gender,sum(salary) from employee group by gender**  **UNION ALL**  **select NULL as country, NULL as gender,sum(salary) from employee;**  Query ID = shalaj\_20170516093636\_74480bd3-f393-4775-8271-1ab6bbaf3195  Total jobs = 5  Launching Job 1 out of 5  Number of reduce tasks not specified. Estimated from input data size: 1  In order to change the average load for a reducer (in bytes):  set hive.exec.reducers.bytes.per.reducer=<number>  In order to limit the maximum number of reducers:  set hive.exec.reducers.max=<number>  In order to set a constant number of reducers:  set mapreduce.job.reduces=<number>  Starting Job = job\_1493893836939\_0199, Tracking URL = http://mac127:8088/proxy/application\_1493893836939\_0199/  Kill Command = /opt/cloudera/parcels/CDH-5.10.1-1.cdh5.10.1.p0.10/lib/hadoop/bin/hadoop job -kill job\_1493893836939\_0199  Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  2017-05-16 09:37:04,418 Stage-1 map = 0%, reduce = 0%  2017-05-16 09:37:11,675 Stage-1 map = 100%, reduce = 0%  2017-05-16 09:37:20,976 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.48 sec  MapReduce Total cumulative CPU time: 3 seconds 480 msec  Ended Job = job\_1493893836939\_0199  Launching Job 2 out of 5  Number of reduce tasks not specified. Estimated from input data size: 1  In order to change the average load for a reducer (in bytes):  set hive.exec.reducers.bytes.per.reducer=<number>  In order to limit the maximum number of reducers:  set hive.exec.reducers.max=<number>  In order to set a constant number of reducers:  set mapreduce.job.reduces=<number>  Starting Job = job\_1493893836939\_0200, Tracking URL = http://mac127:8088/proxy/application\_1493893836939\_0200/  Kill Command = /opt/cloudera/parcels/CDH-5.10.1-1.cdh5.10.1.p0.10/lib/hadoop/bin/hadoop job -kill job\_1493893836939\_0200  Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1  2017-05-16 09:37:35,313 Stage-3 map = 0%, reduce = 0%  2017-05-16 09:37:43,589 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.62 sec  2017-05-16 09:37:51,873 Stage-3 map = 100%, reduce = 100%, Cumulative CPU 4.16 sec  MapReduce Total cumulative CPU time: 4 seconds 160 msec  Ended Job = job\_1493893836939\_0200  Launching Job 3 out of 5  Number of reduce tasks not specified. Estimated from input data size: 1  In order to change the average load for a reducer (in bytes):  set hive.exec.reducers.bytes.per.reducer=<number>  In order to limit the maximum number of reducers:  set hive.exec.reducers.max=<number>  In order to set a constant number of reducers:  set mapreduce.job.reduces=<number>  Starting Job = job\_1493893836939\_0201, Tracking URL = http://mac127:8088/proxy/application\_1493893836939\_0201/  Kill Command = /opt/cloudera/parcels/CDH-5.10.1-1.cdh5.10.1.p0.10/lib/hadoop/bin/hadoop job -kill job\_1493893836939\_0201  Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 1  2017-05-16 09:38:07,585 Stage-4 map = 0%, reduce = 0%  2017-05-16 09:38:14,884 Stage-4 map = 100%, reduce = 0%, Cumulative CPU 1.56 sec  2017-05-16 09:38:23,159 Stage-4 map = 100%, reduce = 100%, Cumulative CPU 4.13 sec  MapReduce Total cumulative CPU time: 4 seconds 130 msec  Ended Job = job\_1493893836939\_0201  Launching Job 4 out of 5  Number of reduce tasks determined at compile time: 1  In order to change the average load for a reducer (in bytes):  set hive.exec.reducers.bytes.per.reducer=<number>  In order to limit the maximum number of reducers:  set hive.exec.reducers.max=<number>  In order to set a constant number of reducers:  set mapreduce.job.reduces=<number>  Starting Job = job\_1493893836939\_0202, Tracking URL = http://mac127:8088/proxy/application\_1493893836939\_0202/  Kill Command = /opt/cloudera/parcels/CDH-5.10.1-1.cdh5.10.1.p0.10/lib/hadoop/bin/hadoop job -kill job\_1493893836939\_0202  Hadoop job information for Stage-5: number of mappers: 1; number of reducers: 1  2017-05-16 09:38:39,400 Stage-5 map = 0%, reduce = 0%  2017-05-16 09:38:47,691 Stage-5 map = 100%, reduce = 0%, Cumulative CPU 1.63 sec  2017-05-16 09:38:56,981 Stage-5 map = 100%, reduce = 100%, Cumulative CPU 4.19 sec  MapReduce Total cumulative CPU time: 4 seconds 190 msec  Ended Job = job\_1493893836939\_0202  Launching Job 5 out of 5  Number of reduce tasks is set to 0 since there's no reduce operator  Starting Job = job\_1493893836939\_0203, Tracking URL = http://mac127:8088/proxy/application\_1493893836939\_0203/  Kill Command = /opt/cloudera/parcels/CDH-5.10.1-1.cdh5.10.1.p0.10/lib/hadoop/bin/hadoop job -kill job\_1493893836939\_0203  Hadoop job information for Stage-2: number of mappers: 4; number of reducers: 0  2017-05-16 09:39:13,072 Stage-2 map = 0%, reduce = 0%  2017-05-16 09:39:24,638 Stage-2 map = 75%, reduce = 0%, Cumulative CPU 5.64 sec  2017-05-16 09:39:30,823 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 7.25 sec  MapReduce Total cumulative CPU time: 7 seconds 250 msec  Ended Job = job\_1493893836939\_0203  MapReduce Jobs Launched:  Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.48 sec HDFS Read: 7182 HDFS Write: 278 SUCCESS  Stage-Stage-3: Map: 1 Reduce: 1 Cumulative CPU: 4.16 sec HDFS Read: 7255 HDFS Write: 169 SUCCESS  Stage-Stage-4: Map: 1 Reduce: 1 Cumulative CPU: 4.13 sec HDFS Read: 7226 HDFS Write: 148 SUCCESS  Stage-Stage-5: Map: 1 Reduce: 1 Cumulative CPU: 4.19 sec HDFS Read: 6871 HDFS Write: 116 SUCCESS  Stage-Stage-2: Map: 4 Cumulative CPU: 7.25 sec HDFS Read: 15035 HDFS Write: 177 SUCCESS  Total MapReduce CPU Time Spent: 23 seconds 210 msec  OK  India Female 4000  India Male 8000  UK Female 5000  UK Male 12000  USA Female 12500  USA Male 10000  India NULL 12000  UK NULL 17000  USA NULL 22500  NULL Female 21500  NULL Male 30000  NULL NULL 51500  Time taken: 161.761 seconds, Fetched: 12 row(s) |

These queries scan employee table multiple time and also run many map reduce jobs

# GOUPING SETS

We can get the same result by using GROUPING SETS

|  |
| --- |
| **SELECT country,gender,sum(salary) totalsalary**  **from employee**  **GROUP BY country,gender**  **GROUPING SETS (**  **(country,gender),**  **(country),**  **(gender),**  **()**  **)**  Query ID = shalaj\_20170516093535\_c7e0b558-d1a4-42a7-b4de-58c03e00e2b5  Total jobs = 1  Launching Job 1 out of 1  Number of reduce tasks not specified. Estimated from input data size: 1  In order to change the average load for a reducer (in bytes):  set hive.exec.reducers.bytes.per.reducer=<number>  In order to limit the maximum number of reducers:  set hive.exec.reducers.max=<number>  In order to set a constant number of reducers:  set mapreduce.job.reduces=<number>  Starting Job = job\_1493893836939\_0198, Tracking URL = http://mac127:8088/proxy/application\_1493893836939\_0198/  Kill Command = /opt/cloudera/parcels/CDH-5.10.1-1.cdh5.10.1.p0.10/lib/hadoop/bin/hadoop job -kill job\_1493893836939\_0198  Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  2017-05-16 09:35:50,206 Stage-1 map = 0%, reduce = 0%  2017-05-16 09:35:57,499 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.65 sec  2017-05-16 09:36:06,843 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.78 sec  MapReduce Total cumulative CPU time: 3 seconds 780 msec  Ended Job = job\_1493893836939\_0198  MapReduce Jobs Launched:  Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.78 sec HDFS Read: 8406 HDFS Write: 177 SUCCESS  Total MapReduce CPU Time Spent: 3 seconds 780 msec  OK  NULL NULL 51500  NULL Female 21500  NULL Male 30000  India NULL 12000  India Female 4000  India Male 8000  UK NULL 17000  UK Female 5000  UK Male 12000  USA NULL 22500  USA Female 12500  USA Male 10000  Time taken: 34.785 seconds, Fetched: 12 row(s) |

As we can see by using GROUPING SETS option we got same result and here we scan employee table just once and processing is done by single map reduce job. This query run faster than union all query

Union all query took 161.761 seconds while this query executed in 34.785 seconds only

The result is same but not in same order

# ROLLUP

Rollup is used to do aggregate operation on multiple levels in hierarchy

**Retrieving salary by country along with grand total**

|  |
| --- |
| **SELECT country,sum(salary) totalsalary**  **from employee**  **GROUP BY country WITH ROLLUP**  NULL 51500  India 12000  UK 17000  USA 22500 |

**Group salary by country and gender also compute subtotal at COUNTRY level and grand total**

|  |
| --- |
| **SELECT country,gender,sum(salary) totalsalary**  **from employee**  **GROUP BY country,gender WITH ROLLUP**  NULL NULL 51500  India NULL 12000  India Female 4000  India Male 8000  UK NULL 17000  UK Female 5000  UK Male 12000  USA NULL 22500  USA Female 12500  USA Male 10000 |

**Group salary by country and gender also compute subtotal at GENDER level and grand total**

|  |
| --- |
| **SELECT country,gender,sum(salary) totalsalary**  **from employee**  **GROUP BY gender,country WITH ROLLUP**  NULL NULL 51500  NULL Female 21500  India Female 4000  UK Female 5000  USA Female 12500  NULL Male 30000  India Male 8000  UK Male 12000  USA Male 10000 |

# CUBE

CUBE produces the result set by generating all combination of columns specified in GROUP BY clause

Write a query to retrieve sum of salary grouped by all combination of 2 columns (country and gender) as well as grand total

* Group by country,gender
* Group by only country
* Group by only gender
* Grand total

|  |
| --- |
| SELECT country,gender,sum(salary) totalsalary  from employee  GROUP BY country,gender WITH CUBE  Query ID = shalaj\_20170516101414\_98d47979-7b4c-4890-93ad-2458df40f80b  Total jobs = 1  Launching Job 1 out of 1  Number of reduce tasks not specified. Estimated from input data size: 1  In order to change the average load for a reducer (in bytes):  set hive.exec.reducers.bytes.per.reducer=<number>  In order to limit the maximum number of reducers:  set hive.exec.reducers.max=<number>  In order to set a constant number of reducers:  set mapreduce.job.reduces=<number>  Starting Job = job\_1493893836939\_0208, Tracking URL = http://mac127:8088/proxy/application\_1493893836939\_0208/  Kill Command = /opt/cloudera/parcels/CDH-5.10.1-1.cdh5.10.1.p0.10/lib/hadoop/bin/hadoop job -kill job\_1493893836939\_0208  Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1  2017-05-16 10:14:38,651 Stage-1 map = 0%, reduce = 0%  2017-05-16 10:14:45,872 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.63 sec  2017-05-16 10:14:54,122 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.74 sec  MapReduce Total cumulative CPU time: 3 seconds 740 msec  Ended Job = job\_1493893836939\_0208  MapReduce Jobs Launched:  Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.74 sec HDFS Read: 8406 HDFS Write: 177 SUCCESS  Total MapReduce CPU Time Spent: 3 seconds 740 msec  OK  NULL NULL 51500  NULL Female 21500  NULL Male 30000  India NULL 12000  India Female 4000  India Male 8000  UK NULL 17000  UK Female 5000  UK Male 12000  USA NULL 22500  USA Female 12500  USA Male 10000  Time taken: 34.941 seconds, Fetched: 12 row(s) |

This is the same result the we achieved through GROUPING SETS, but in grouping sets we have liberty to choose our grouping conditions

# Grouping Function in SQL Server

Grouping function is not supported in hive but before going to Grouping\_ID we need to understand Grouping function available in SQL Server

**Sales.data**

|  |
| --- |
| Asia,India,Bangalore,1000  Asia,India,Chennai,2000  Asia,Japan,Tokyo,4000  Asia,Japan,Hiroshima,5000  Europe,UK,London,1000  Europe,UK,Manchester,2000  Europe,France,Paris,4000  Europe,France,Cannes,5000 |

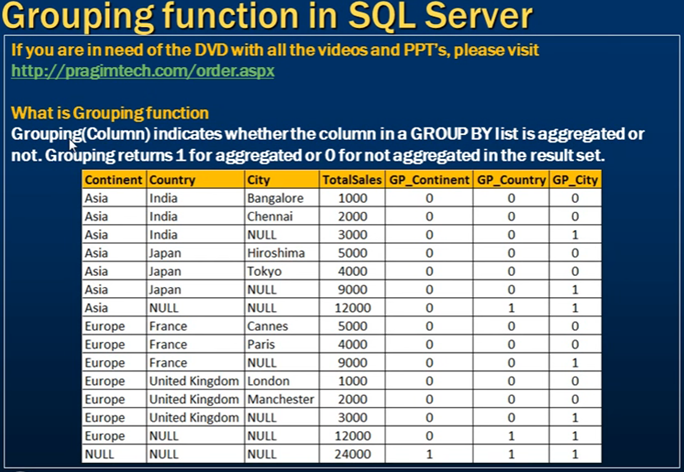
**Table**

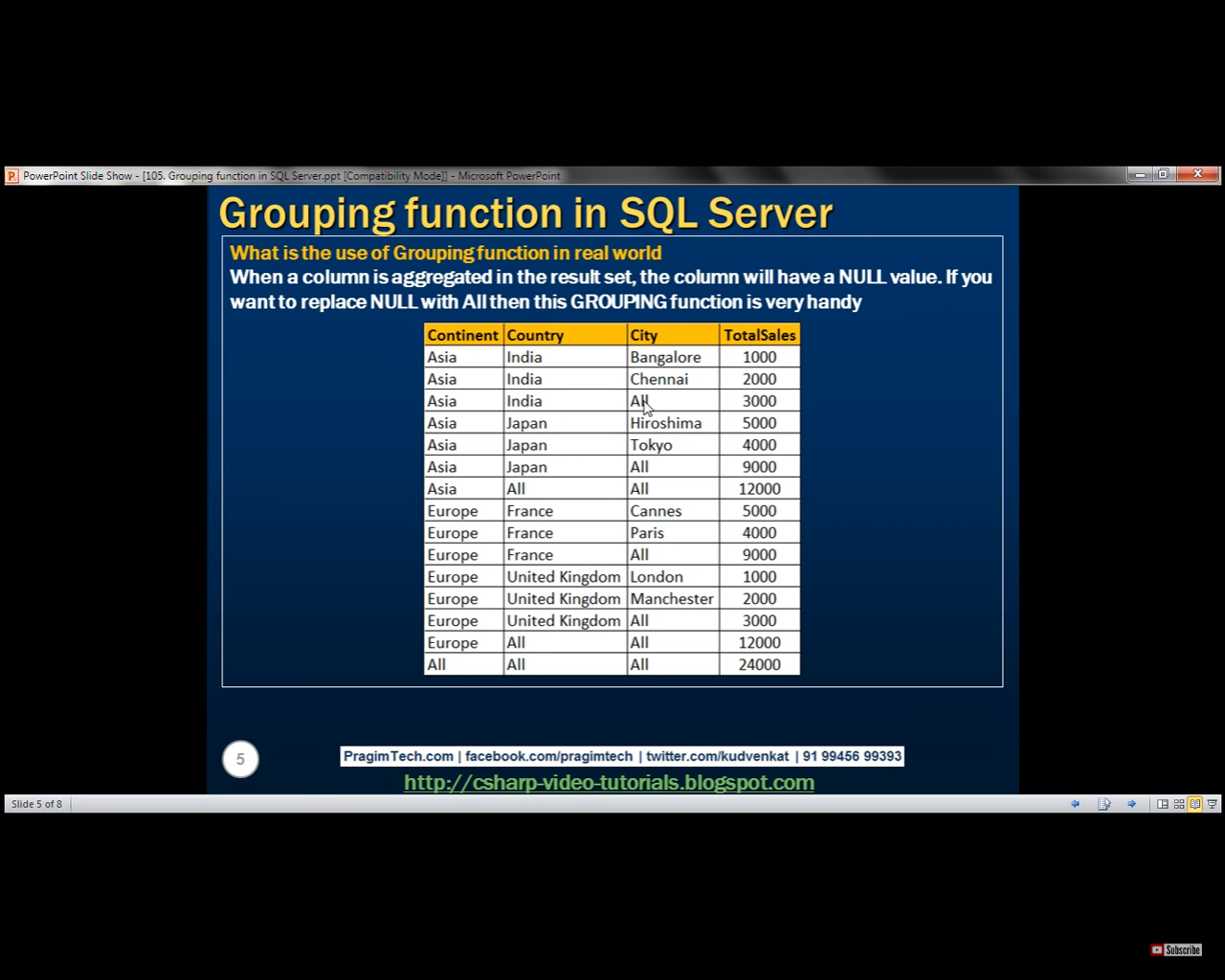
|  |
| --- |
| create table sales(continent string,country string,city string,saleamount int) Row Format delimited fields terminated by ','; |

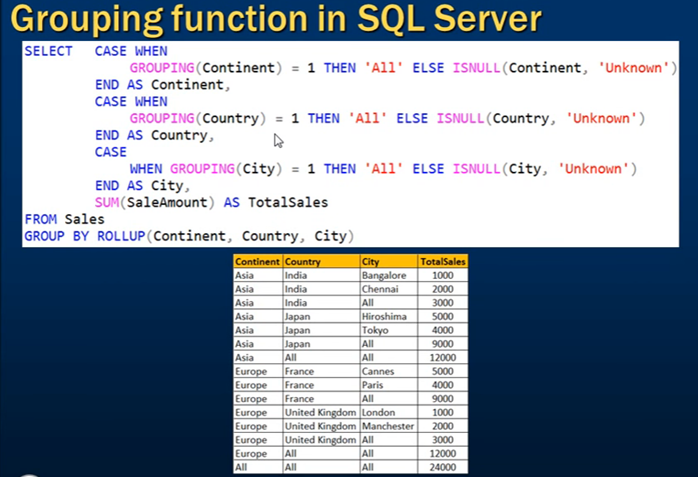
**Load data**

|  |
| --- |
| load data local inpath '/home/shalaj/sales.data' into table sales; |

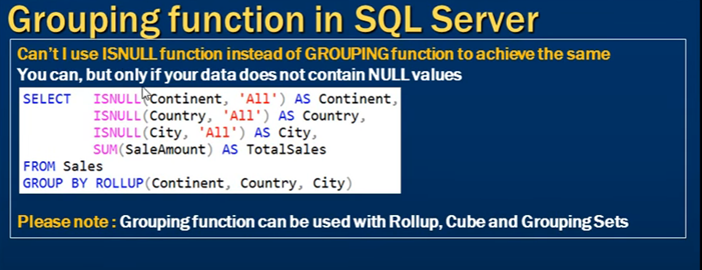
|  |
| --- |
| Asia India Bangalore 1000  Asia India Chennai 2000  Asia Japan Tokyo 4000  Asia Japan Hiroshima 5000  Europe UK London 1000  Europe UK Manchester 2000  Europe France Paris 4000  Europe France Cannes 5000 |

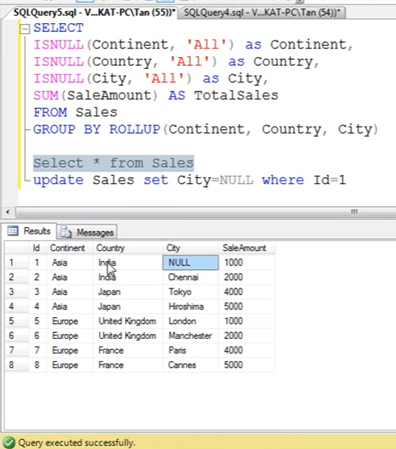




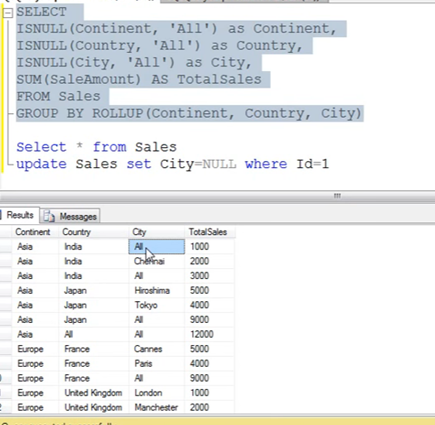


Instead of NULL if we want to display ALL we can use



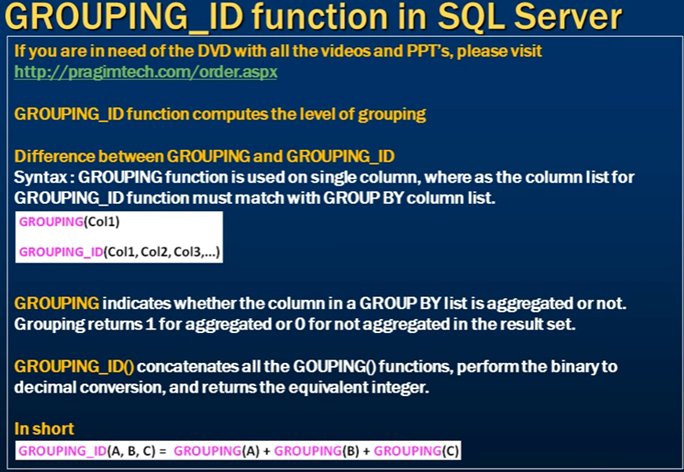


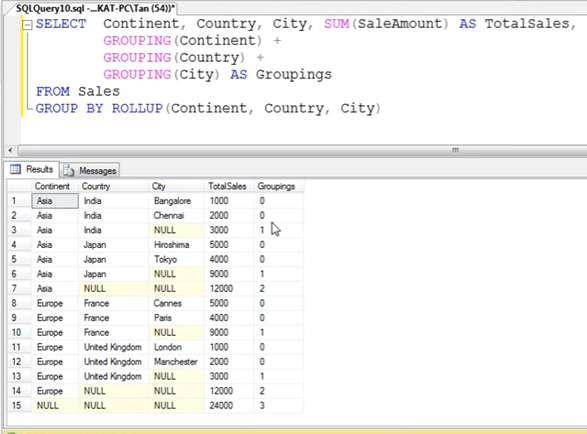
So let’s say if City is NULL in one of the column



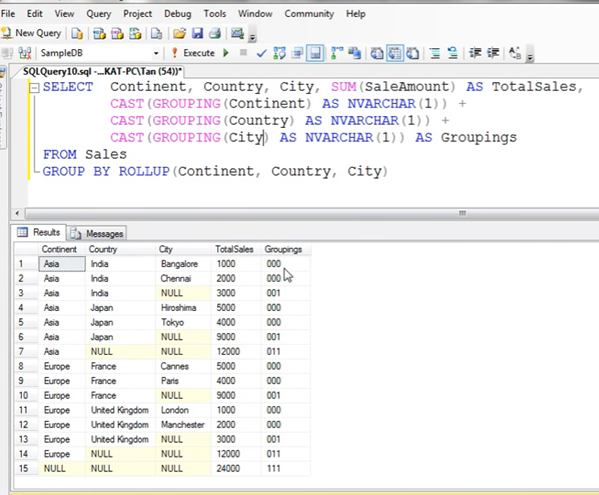
Now this query produced wrong result in first row , it looks like the first row is aggregated by city

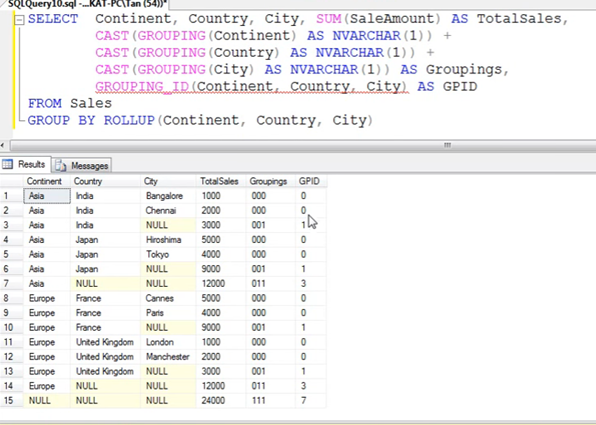
# GROUPING\_ID In SQL Server

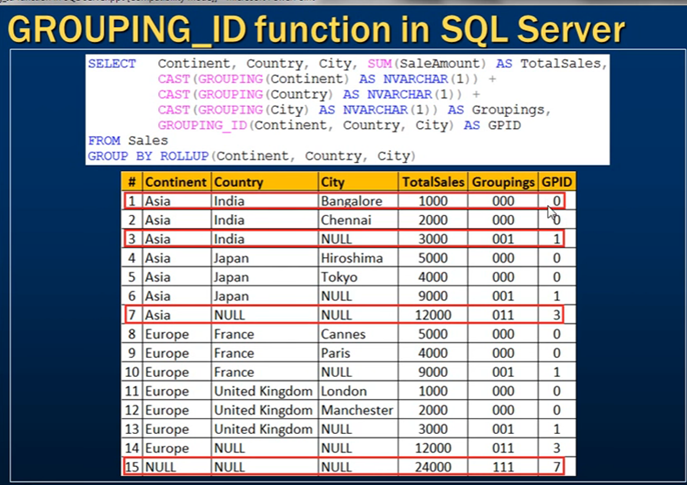


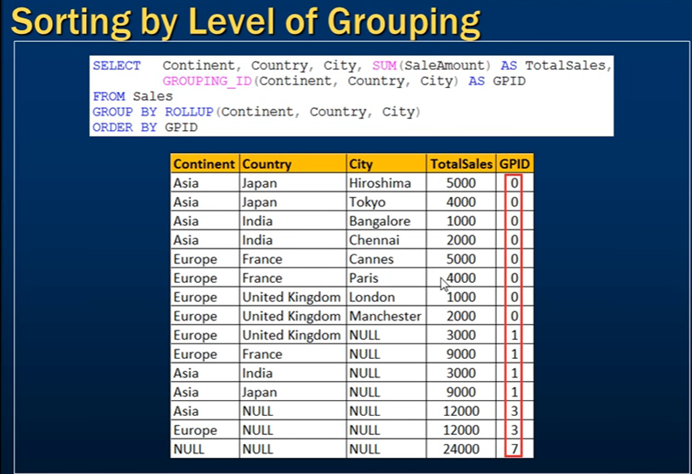


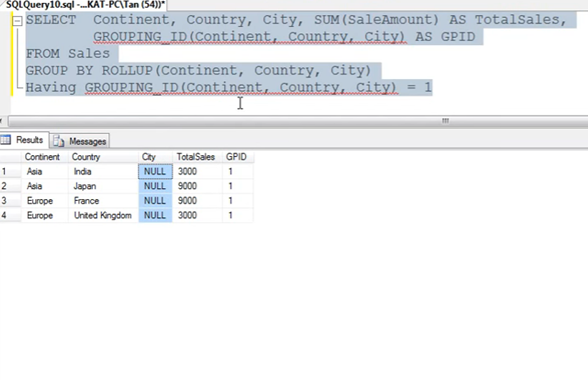
Plus operation gives us numeric values to get bitwise value we need to cast column values











# Grouping\_\_ID in hive

|  |
| --- |
| select continent,country,city,sum(saleamount) as totalsales,  Grouping\_\_ID as GP\_CN  from sales  group by continent,country,city with rollup  order by GP\_CN DESC;  Europe UK Manchester 2000 7  Europe UK London 1000 7  Europe France Paris 4000 7  Europe France Cannes 5000 7  Asia Japan Tokyo 4000 7  Asia Japan Hiroshima 5000 7  Asia India Chennai 2000 7  Asia India Bangalore 1000 7  Asia India NULL 3000 3  Europe UK NULL 3000 3  Asia Japan NULL 9000 3  Europe France NULL 9000 3  Asia NULL NULL 12000 1  Europe NULL NULL 12000 1  NULL NULL NULL 24000 0 |

When aggregates are displayed for a column its value is null. This may conflict in case the column itself has some null values. There needs to be some way to identify NULL in column, which means aggregate and NULL in column, which means value. GROUPING\_\_ID function is the solution to that.

This function returns a bitvector corresponding to whether each column is present or not. For each column, a value of "1" is produced for a row in

The result set if that column has been aggregated in that row, otherwise the value is "0". This can be used to differentiate when there are nulls in the data.

Consider the following example:

| **Column1 (key)** | **Column2 (value)** |
| --- | --- |
| 1 | NULL |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 3 | NULL |
| 4 | 5 |

|  |
| --- |
| SELECT key, value, GROUPING\_\_ID as GID, count(\*) as cnt from T1 GROUP BY key, value WITH ROLLUP ORDER BY GROUPING\_\_ID desc; |

| **Key** | **value** | **GID** | **CNT** |
| --- | --- | --- | --- |
| 4 | 5 | 3 | 1 |
| 3 | 3 | 3 | 1 |
| 3 | NULL | 3 | 1 |
| 2 | 2 | 3 | 1 |
| 1 | 1 | 3 | 1 |
| 1 | NULL | 3 | 1 |
| 4 | NULL | 1 | 1 |
| 3 | NULL | 1 | 2 |
| 2 | NULL | 1 | 1 |
| 1 | NULL | 1 | 2 |
| NULL | NULL | 0 | 6 |

* For GID having value 3 both the column are being selected
* For GID having value 1 only first column is being selected
* For last row none of the columns are being selected