**Spark SQL**

CREATE TABLE "CLIENT"."DEPARTMENT"

( "DEPT\_ID" NUMBER(2,0) NOT NULL ENABLE,

"DEPT\_NAME" VARCHAR2(200 BYTE),

"DEPT\_LOC" VARCHAR2(200 BYTE),

CONSTRAINT "DEPARTMENT\_PK" PRIMARY KEY ("DEPT\_ID")

)

CREATE TABLE "CLIENT"."EMPLOYEE"

( "EMP\_ID" NUMBER(5,0) NOT NULL ENABLE,

"F\_NAME" VARCHAR2(200 BYTE),

"SALARY" NUMBER(5,0),

"DEPT\_ID" NUMBER(5,0),

"L\_NAME" VARCHAR2(200 BYTE),

CONSTRAINT "EMPLOYEE\_PK" PRIMARY KEY ("EMP\_ID"),

CONSTRAINT "EMPLOYEE\_FK" FOREIGN KEY ("DEPT\_ID")

REFERENCES "CLIENT"."DEPARTMENT" ("DEPT\_ID") ENABLE

)

val departments = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "client.DEPARTMENT"));

val employees = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "client.EMPLOYEE"));

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No | Operations | SQL | RDD |
| 1. | Select | SELECT \* FROM EMPLOYEE; | employees.select("\*").show()  employees.show()  employees.select($"F\_NAME",$"SALARY"+10).show()  employees.select($"F\_NAME",$"SALARY"+10,concat($"F\_NAME",lit(" "),$"L\_NAME").as("FullName")).show() |
|  | Select | select F\_NAME, L\_NAME from EMPLOYEE | employees.select("F\_NAME","L\_NAME").show() |
|  | Select | select F\_NAME, L\_NAME, SALARY, 100 as BONUS from EMPLOYEE | employees.select($"F\_NAME",$"L\_NAME",$"SALARY",$"SALARY", lit(100)as "BONUS").show() |
|  | Select | select F\_NAME, L\_NAME, SALARY, SALARY + 200 as REVISED\_SALARY from EMPLOYEE | employees.select($"F\_NAME",$"L\_NAME",$"SALARY",$"SALARY"+200 as "REVISED\_SALARY").show() |
|  |  |  | employees.select(abs(sum("SALARY")) as "total").show() |
| 2. | Where | SELECT \* FROM EMPLOYEE WHERE F\_NAME='E3'; | val where =employees.filter("F\_NAME='E3'").show()  val where =employees.filter("SALARY in(100,200,300)").show()  val where =employees.filter("SALARY between 200 and 400").show()  val where =employees.filter("SALARY >= 300").show()  employees.where($"salary" > 300).show()  employees.where($"salary" in(100,200,300)).show()  departments.where("dept\_loc in ('PUNE','THANE')").show() |
|  | Where | select emp\_id, f\_name, l\_name, salary from employee  where salary > 100 and dept\_id > 1 | employees.where($"SALARY" > 100 and $"DEPT\_ID" > 1).select("F\_NAME","L\_NAME","SALARY").show() |
|  | Where | select dept.dept\_id, dept.dept\_name, dept.dept\_loc, emp.salary  from employee emp, department dept  where emp.dept\_id = dept.dept\_id  and (dept.dept\_loc = 'PUNE' or emp.salary > 100) | employees.as("emp").join(departments.as("dept"),col("emp.dept\_id")===col("dept.dept\_id")).where($"dept.dept\_loc" === "PUNE" or $"emp.salary" > 100).selectExpr("dept.dept\_id","dept.dept\_name","dept.dept\_loc","emp.salary").show() |
|  |  |  | employees.select("\*").where($"SALARY" >= 200 and $"SALARY" <= 400).show() |
| 3. | Distinct | SELECT DISTINCT (F\_NAME)  FROM EMPLOYEE; | employees.select("F\_NAME").distinct.show()  employees.as("emp").join(departments.as("dept"),col("emp.dept\_id")===col("dept.dept\_id")).where($"dept.dept\_loc" === "PUNE" or $"emp.salary" > 100).selectExpr("dept.dept\_id","dept.dept\_name","dept.dept\_loc","emp.salary").distinct.show() |
| 4. | Order By | SELECT \* FROM EMPLOYEE ORDER BY F\_NAME ASC; | employees.orderBy(asc("F\_NAME")).show() |
| 5. | Group By |  | employees.groupBy("F\_NAME").count.show()  employees.groupBy("F\_NAME").min("SALARY").show()  employees.groupBy("F\_NAME").max("SALARY").show() |
| 6. | Having |  |  |
| 7. | Rank | select b.Zone, c.FISCAL\_YEAR, c.FISCAL\_QUARTER\_SHORT\_DESC, sum(a.NO\_OF\_EMPS)  --,row\_number() OVER (order by b.Zone, b.state\_new) AS row\_number  ,rank() OVER (order by b.Zone) AS rank\_1  --dense\_rank() OVER (order by b.Zone) AS dense\_rank\_1  from  NEW\_FACT\_BRANCH\_PL a, new\_dim\_branchwise b, mst\_calendar c  where a.BRANCHWISE\_SK = b.BRANCHWISE\_SK  and a.date\_sk = c.date\_tk  group by b.Zone, c.FISCAL\_YEAR, c.FISCAL\_QUARTER\_SHORT\_DESC | val factBranch = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "client.NEW\_FACT\_BRANCH\_PL"));  val branchwise = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "client.new\_dim\_branchwise"));  val calendar = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "client.mst\_calendar"));  val join1=factBranch.join(branchwise, factBranch.col("BRANCHWISE\_SK")===branchwise("BRANCHWISE\_SK"))  val join2=join1.join(calendar,join1.col("date\_sk")===calendar("date\_tk"))  val join3=join2.groupBy("Zone","FISCAL\_YEAR","FISCAL\_QUARTER\_SHORT\_DESC").sum("NO\_OF\_EMPS")  import org.apache.spark.sql.expressions.Window  val orderBy = Window.orderBy("Zone")  val rankOfZone1 = rank.over(orderBy)  val ranked = join3.withColumn("Rank", rankOfZone1)  ranked.show() |
| 8. | Dense Rank | select b.Zone, c.FISCAL\_YEAR, c.FISCAL\_QUARTER\_SHORT\_DESC, sum(a.NO\_OF\_EMPS)  --,row\_number() OVER (order by b.Zone, b.state\_new) AS row\_number  --,rank() OVER (order by b.Zone, b.state\_new) AS rank\_1  ,dense\_rank() OVER (order by b.Zone) AS dense\_rank\_1  from  NEW\_FACT\_BRANCH\_PL a, new\_dim\_branchwise b, mst\_calendar c  where a.BRANCHWISE\_SK = b.BRANCHWISE\_SK  and a.date\_sk = c.date\_tk  group by b.Zone, c.FISCAL\_YEAR, c.FISCAL\_QUARTER\_SHORT\_DESC; | val orderBy = Window.orderBy("Zone")  val rankOfZone = denseRank.over(orderBy)  val ranked = join3.withColumn("Dense\_Rank", rankOfZone)  ranked.show() |
| 9. | Pagination |  |  |
| 10. | Aggregation(  Min  Max  Avg  Sum  ) | SELECT MAX(SALARY),sum(SALARY),AVG(SALARY),MAX(SALARY),MIN(SALARY),EMP\_ID FROM EMPLOYEE GROUP BY DEPT\_ID,EMP\_ID; | employees.groupBy("DEPT\_ID","EMP\_ID").agg(max("SALARY"),sum("SALARY"),avg("SALARY"),min("SALARY")).show; |
|  |  | **SELECT**  DEPARTMENT.DEPT\_LOC, SUM(EMPLOYEE.SALARY) AS LOC\_SALARY  **FROM**  EMPLOYEE EMPLOYEE,  DEPARTMENT DEPARTMENT  **WHERE**  EMPLOYEE.DEPT\_ID = DEPARTMENT.DEPT\_ID  **GROUP BY** DEPARTMENT.DEPT\_LOC  **HAVING**  SUM(EMPLOYEE.SALARY) > 300 |  |
| 11. | Null Handling |  |  |
| 12. | Not in |  | departments.where("dept\_loc not in ('PUNE','THANE')").show() |
| 13. | Joins(  Inner  left-outer  right-outer  full-outer  self  ) | SELECT \* FROM EMPLOYEE e, DEPARTMENT d where e.DEPT\_ID=d.DEPT\_ID  SELECT \* FROM EMPLOYEE e FULL OUTER JOIN DEPARTMENT d on e.DEPT\_ID=d.DEPT\_ID ORDER BY e.EMP\_ID;  SELECT \* FROM EMPLOYEE e LEFT OUTER JOIN DEPARTMENT d on e.DEPT\_ID=d.DEPT\_ID ORDER BY e.EMP\_ID;  SELECT \* FROM EMPLOYEE e RIGHT OUTER JOIN DEPARTMENT d on e.DEPT\_ID=d.DEPT\_ID ORDER BY e.EMP\_ID;  SELECT \* FROM EMPLOYEE e INNER JOIN DEPARTMENT d on e.DEPT\_ID=d.DEPT\_ID ORDER BY e.EMP\_ID;  SELECT \* FROM EMPLOYEE e LEFT JOIN DEPARTMENT d on e.DEPT\_ID=d.DEPT\_ID ORDER BY e.EMP\_ID;  SELECT e1.EMP\_ID,e2.EMP\_ID as "MANAGER\_ID" FROM EMPLOYEE e1, EMPLOYEE e2 where e1.MANAGER=e2.EMP\_ID | employees.join(departments, employees.col("DEPT\_ID")===departments("DEPT\_ID")).show  employees.join(departments, employees.col("DEPT\_ID")===departments("DEPT\_ID"),"outer").orderBy(employees.col("EMP\_ID")).show  employees.join(departments, employees.col("DEPT\_ID")===departments("DEPT\_ID"),"left\_outer").orderBy(employees.col("EMP\_ID")).show  employees.join(departments, employees.col("DEPT\_ID")===departments("DEPT\_ID"),"right\_outer").orderBy(employees.col("EMP\_ID")).show  employees.join(departments, employees.col("DEPT\_ID")===departments("DEPT\_ID"),"inner").orderBy(employees.col("EMP\_ID")).show  employees.join(departments, employees.col("DEPT\_ID")===departments("DEPT\_ID"),"leftsemi").show  employees.as("e1").join(employees.as("e2"), col("e1.EMP\_ID")===col("e2.MANAGER")).selectExpr("e2.EMP\_ID","e1.EMP\_ID as MANAGER\_ID").show |
| 14. | Rollup | SELECT e.DEPT\_ID,e.MANAGER,SUM(salary) "Total" FROM EMPLOYEE e GROUP BY ROLLUP( e.DEPT\_ID,e.MANAGER) ORDER BY e.DEPT\_ID; | employees.rollup($"DEPT\_ID",$"MANAGER").sum("SALARY").orderBy($"DEPT\_ID").show |
| 15. | Cube | SELECT e.DEPT\_ID,e.MANAGER,SUM(salary) FROM EMPLOYEE e GROUP BY CUBE( e.DEPT\_ID,e.MANAGER) ORDER BY e.DEPT\_ID; | employees.cube($"DEPT\_ID",$"MANAGER").sum("SALARY").orderBy($"DEPT\_ID").show |
| 16. | CASE Expression | select Dept\_loc,  case when Dept\_loc = 'THANE' then 'CENTRAL'  when Dept\_loc = 'MUMBAI' then 'WEST'  else 'OTHER'  end as "REGION"  from department | departments.select($"Dept\_loc",when(departments("Dept\_loc") === "'THANE'", "CENTRAL").when(departments("Dept\_loc") === "MUMBAI", "WEST").otherwise("REGION").as("REGION")).show |
|  | select + case | select \*  from employee  where case when salary > 200 then 'Good'  else 'Average'  end = 'Average' |  |
| 17. | Like | select \* from employee where F\_NAME like "E3%" | employees.filter($"F\_NAME" like "E3%").show |
| 18. | Functions(RND,FLR,Conversions) | 1. select ceil(7.536432),   round(7.536432,2),  floor(7.536432),  mod(10,3),  TRUNC(7.536432,2),  from dual | math.ceil(7.536432)  math.round(7.536432)  math.floor(7.536432) |
|  |  | select nvl(dept\_id,0) from employee |  |
|  |  | select F\_Name, l\_name,  (select dept\_name FROM department D WHERE D.DEPT\_ID = e.DEPT\_ID) AS DEPT\_NAME  FROM EMPLOYEE E |  |
| 19. | Variance |  |  |
| 20 | Union | select employee.emp\_id,employee.f\_name,employee.salary,department.dept\_name,department.dept\_loc  from employee  inner join department  on employee.dept\_id=department.dept\_id  union  select employee2.emp\_id,employee2.f\_name,employee2.salary,department.dept\_name,department.dept\_loc  from employee2  inner join department  on employee2.dept\_id=department.dept\_id | val employees1 = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "employee"));  val employees2 = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "employee2"));  val emp1=employees1.as("emp").join(departments.as("dept"),$"emp.dept\_id"===$"dept.dept\_id","inner").selectExpr("emp.emp\_id","emp.f\_name","emp.salary","dept.dept\_id","dept.dept\_name")  val emp2=employees1.as("emp").join(departments.as("dept"),$"emp.dept2\_id"===$"dept.dept\_id","inner").selectExpr("emp.emp\_id","emp.f\_name","emp.salary","dept.dept\_id","dept.dept\_name")  emp1.unionAll(emp2).distinct.show |
| 21. | Union All | select employee.emp\_id,employee.f\_name,employee.salary,department.dept\_name,department.dept\_loc  from employee  inner join department  on employee.dept\_id=department.dept\_id  union all  select employee2.emp\_id,employee2.f\_name,employee2.salary,department.dept\_name,department.dept\_loc  from employee2  inner join department  on employee2.dept\_id=department.dept\_id | val employees1 = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "employee"));  val employees2 = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "employee2"));  val emp1=employees1.as("emp").join(departments.as("dept"),$"emp.dept\_id"===$"dept.dept\_id","inner").selectExpr("emp.emp\_id","emp.f\_name","emp.salary","dept.dept\_id","dept.dept\_name")  val emp2=employees1.as("emp").join(departments.as("dept"),$"emp.dept2\_id"===$"dept.dept\_id","inner").selectExpr("emp.emp\_id","emp.f\_name","emp.salary","dept.dept\_id","dept.dept\_name")  emp1.unionAll(emp2).show |
| 22. | Subqueries | select dept\_name, dept\_loc, state\_new, zone  from department,  (select distinct zone, state\_new, city  from NEW\_DIM\_BRANCHWISE) location  where dept\_loc = upper(location.city) | departments.as("dept").join(branchwise.select("zone","state\_new","city").distinct.as("location"),col("dept.dept\_loc")===upper(col("location.city"))).show |
|  |  | SELECT \*  FROM employees  WHERE emp\_id IN (SELECT emp\_id  FROM employees  WHERE SALARY > 200) ; | val items = employees.where($"salary" > 100).select($"emp\_id").collect().map{ row=>row.get(0)}  employees.where($"emp\_id".isin(items:\_\*)).show |
| 23. | Running Sum |  |  |
| 24. | lead | SELECT emp\_id,f\_name,salary, LEAD(salary, 1, 0) OVER (ORDER BY salary) AS salary\_next,salary - LEAD(salary, 1, 0) OVER (ORDER BY salary) AS salary\_diff FROM employee ; | val windowSpec = Window.orderBy("salary")  employee.select(employee("F\_NAME"),employee("emp\_id"),employee("salary"),lead(employee("salary"),1,0).over(windowSpec).alias("Next")).show() |
| 25. | lag | SELECT emp\_id, f\_name,salary, LAG(salary, 1, 0) OVER (ORDER BY salary) AS salary\_prev, salary - LAG(salary, 1, 0) OVER (ORDER BY salary) AS salary\_diff FROM employee ; | import org.apache.spark.sql.expressions.Window  val windowSpec = Window.orderBy("salary")  employee.select(employee("F\_NAME"),employee("emp\_id"),employee("salary"),lag(employee("salary"),1,0).over(windowSpec).alias("Prev")).show() |
| 26. | Formatter |  | val formatter = java.text.NumberFormat.getIntegerInstance  employee.select(employee("F\_NAME"),employee("emp\_id"),employee("salary")).collect().foreach(x=> println(formatter.format(x(2)))) |
| 27. | between | SELECT \* FROM EMPLOYEE WHERE salary BETWEEN 100 AND 500; | emp.select($"salary" between (100,500)).show  employees.filter($"doj" between ("2015-11-01 00:00:00","2015-11-23 00:00:00")).show  val s= employees.select($"F\_NAME",$"L\_NAME",$"SALARY",$"SALARY",$"DOJ", to\_date(lit("2015-11-30"))as "CL\_DATE")  s.filter(lit("2015-11-23") between($"DOJ",$"CL\_DATE").show |

select b.Zone, b.state\_new, sum(a.NO\_OF\_EMPS)from FACT\_BRANCH\_PL\_NEW a, new\_dim\_branchwise b where a.BRANCHWISE\_SK = b.BRANCHWISE\_SK group by b.Zone, rollup(b.state\_new)

val factBranch = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "client.NEW\_FACT\_BRANCH\_PL"));

val branchwise = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "client.new\_dim\_branchwise"));

factBranch.join(branchwise, factBranch.col("BRANCHWISE\_SK")===branchwise("BRANCHWISE\_SK")).rollup("Zone","state\_new").sum("NO\_OF\_EMPS").orderBy("Zone").show(30).

factBranch.join(branchwise, factBranch.col("BRANCHWISE\_SK")===branchwise("BRANCHWISE\_SK")).rollup("Zone","state\_new","CITY").sum("NO\_OF\_EMPS").show(30)

**custom function**

val coder: (Int => String) = (arg: Int) => {if (arg < 1001) "little" else "big"}

val sqlfunc = udf(coder)

employees.withColumn("Code", sqlfunc(col("SALARY"))).show

val MST\_ACCOUNTS = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:idfc\_adf/cel1bons@host8.pragmatix.lan:1521/orcldb", "dbtable" -> "idfc\_adf.MST\_ACCOUNTS"));

val ASSET\_BALANCES = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:idfc\_adf/cel1bons@host8.pragmatix.lan:1521/orcldb", "dbtable" -> "idfc\_adf.ASSET\_BALANCES"));

val PENDING\_CASHFLOW\_INVT = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:idfc\_adf/cel1bons@host8.pragmatix.lan:1521/orcldb", "dbtable" -> "idfc\_adf.PENDING\_CASHFLOW\_INVT"));

val PENDING\_CASHFLOW\_TREASURY = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:idfc\_adf/cel1bons@host8.pragmatix.lan:1521/orcldb", "dbtable" -> "idfc\_adf.PENDING\_CASHFLOW\_TREASURY"));

SELECT MA.CUSTOMER\_CODE, ABS(SUM(AB.LOAN\_AMT\_OS\_INR)) BORROWER\_LEVEL\_EXPOSURE

FROM IDFC\_ADF.MST\_ACCOUNTS MA

INNER JOIN IDFC\_ADF.ASSET\_BALANCES AB ON MA.ACCOUNT\_NO = AB.LOAN\_NO

WHERE '21-FEB-2015' BETWEEN MA.FROM\_DATE AND MA.TO\_DATE

AND AB.BALANCE\_DATE = '21-FEB-2015'

GROUP BY MA.CUSTOMER\_CODE

UNION ALL SELECT CUSTOMER\_CODE,SUM(CASHFLOW\_AMT)

FROM IDFC\_ADF.PENDING\_CASHFLOW\_TREASURY

WHERE BALANCE\_DATE = '21-FEB-2015'

GROUP BY CUSTOMER\_CODE

UNION ALL SELECT ISSUER\_UCIC,SUM(CASHFLOW\_AMT)

FROM IDFC\_ADF.PENDING\_CASHFLOW\_INVT

WHERE BALANCE\_DATE = '21-FEB-2015'

GROUP BY ISSUER\_UCIC

MST\_ACCOUNTS.as("MA").join(ASSET\_BALANCES.as("AB"),col("MA.ACCOUNT\_NO")===col("AB.LOAN\_NO")).where("21-FEB-2015" BETWEEN $"MA.FROM\_DATE" AND $"MA.TO\_DATE" AND $"AB.BALANCE\_DATE" === "21-FEB-2015").groupBy("MA.CUSTOMER\_CODE").unionAll(PENDING\_CASHFLOW\_TREASURY.select("CUSTOMER\_CODE",SUM("CASHFLOW\_AMT")).where($"BALANCE\_DATE" === "21-FEB-2015").groupBy("CUSTOMER\_CODE").unionAll(PENDING\_CASHFLOW\_INVT.select("ISSUER\_UCIC",SUM("CASHFLOW\_AMT")).where($"BALANCE\_DATE" === "21-FEB-2015").groupBy("ISSUER\_UCIC"))).select("MA.CUSTOMER\_CODE", ABS(SUM("AB.LOAN\_AMT\_OS\_INR") as "BORROWER\_LEVEL\_EXPOSURE")).show()

**Partition in JDBC using JdbcRDD**

scala> val driver = "com.oracle.driver.OracleDriver";

scala> val username = "client";

scala> val password = "password";

scala> val url="jdbc:oracle:thin:@host4.pragmatix.lan:1521:orcldb1"

scala> import org.apache.spark.rdd.JdbcRDD

scala> import java.sql.{Connection, DriverManager, ResultSet}

scala> Class.forName("oracle.jdbc.driver.OracleDriver").newInstance

scala> val myRDD = new JdbcRDD( sc, () =>DriverManager.getConnection(url,username,password),"select \* from employee where ? <=EMP\_ID and EMP\_ID <= ? ",1, 40, 2)

scala> myRDD.setName("empRDD").cache

scala> myRDD.map(x => x).collect

**Partition in JDBC using DataFrame**

scala> val employees = sqlContext.load("jdbc", Map("url" -> "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "dbtable" -> "employee","partitionColumn" -> "EMP\_ID","lowerBound" -> "1" ,"upperBound" -> "40","numPartitions" -> "2"));

scala> employees.cache.show

**Partition in JDBC without specifying column**

val FACT\_SECURITY = sqlContext.read.jdbc( "jdbc:oracle:thin:client/password@host4.pragmatix.lan:1521:orcldb1", "(select rownum as Sr\_No , c.\* from FACT\_SECURITY c)",Array("Sr\_No BETWEEN 1 and 10","Sr\_No BETWEEN 11 and 20"), new java.util.Properties);

**calculated measure approach**

factBranch.join(branchwise, factBranch.col("BRANCHWISE\_SK")===branchwise("BRANCHWISE\_SK")).groupBy("Zone").agg(sum("NO\_OF\_EMPS").alias("nofepm"), sum("NO\_OF\_MT").alias("nofmt")).orderBy("Zone").selectExpr("(nofmt \* nofepm)\*2 as ab").show

OR(Correcy one)

var a = factBranch.join(branchwise, factBranch.col("BRANCHWISE\_SK")===branchwise("BRANCHWISE\_SK")).groupBy("Zone").agg(sum("NO\_OF\_EMPS").alias("nofepm"), sum("NO\_OF\_MT").alias("nofmt")).orderBy("Zone")

var b = a.selectExpr("\*", "(nofmt \* nofepm)\*2 as calc")

b.show

OR

var b= a.withColumn("cal",(new org.apache.spark.sql.Column("nofmt") \* new org.apache.spark.sql.Column("nofepm")/2))

b.show

**VitalSparkDriver**

import java.util.Properties

var prop = new Properties

prop.put("user", "client");

prop.put("password", "password");

val time = sqlContext.read.jdbc("jdbc:oracle:thin:@host4.pragmatix.lan:1521:orcldb1", "client.MST\_CALENDAR", prop);

val fact = sqlContext.read.jdbc("jdbc:oracle:thin:@host4.pragmatix.lan:1521:orcldb1", "client.NEW\_FACT\_BRANCH\_PL", prop);

val region = sqlContext.read.jdbc("jdbc:oracle:thin:@host4.pragmatix.lan:1521:orcldb1", "client.NEW\_DIM\_BRANCHWISE", prop);

val employees = sqlContext.read.jdbc("jdbc:oracle:thin:@host4.pragmatix.lan:1521:orcldb1", "client.EMPLOYEE", prop);

employees.printSchema

root

|-- EMP\_ID: decimal(5,0) (nullable = false)

|-- F\_NAME: string (nullable = true)

|-- SALARY: decimal(5,0) (nullable = true)

|-- DEPT\_ID: decimal(5,0) (nullable = true)

|-- L\_NAME: string (nullable = true)

|-- MANAGER: decimal(5,0) (nullable = true)

|-- DOJ: timestamp (nullable = true)

employees.show

+------+------+------+-------+------+-------+--------------------+

|EMP\_ID|F\_NAME|SALARY|DEPT\_ID|L\_NAME|MANAGER| DOJ|

+------+------+------+-------+------+-------+--------------------+

| 11| E11| 100| 1| L11| 1|2015-11-22 10:18:...|

| 12| E12| 200| 1| L12| 1|2015-11-22 10:18:...|

| 13| E13| 700| 2| L13| 2|2015-11-22 10:18:...|

| 14| E14| 400| 3| L14| 1|2015-11-25 10:18:...|

| 15| E15| 300| 2| L15| 2|2015-11-22 10:18:...|

| 16| E16| 2000| 4| L16| 2|2015-11-25 10:18:...|

| 17| E17| 3000| 4| L17| 1|2015-11-25 10:18:...|

| 18| E18| 4000| 2| L18| 2|2015-11-24 10:18:...|

| 19| E19| 1500| 2| L19| 1|2015-11-25 10:18:...|

| 20| E20| 1000| 2| L20| 1|2015-11-25 10:18:...|

| 21| E21| 100| 1| L21| 1|2015-11-22 10:18:...|

| 22| E22| 200| 1| L22| 1|2015-11-22 10:18:...|

| 23| E23| 700| 2| L23| 2|2015-11-22 10:18:...|

| 24| E24| 400| 3| L24| 1|2015-11-25 10:18:...|

| 25| E25| 300| 2| L25| 2|2015-11-22 10:18:...|

| 26| E26| 2000| 4| L26| 2|2015-11-25 10:18:...|

| 27| E27| 3000| 4| L27| 1|2015-11-25 10:18:...|

| 28| E28| 4000| 2| L28| 2|2015-11-24 10:18:...|

| 29| E29| 1500| 2| L29| 1|2015-11-25 10:18:...|

| 30| E30| 1000| 2| L30| 1|2015-11-25 10:18:...|

+------+------+------+-------+------+-------+--------------------+

employees.registerTempTable("emp")

sqlContext.sql("Select \* from emp").show

**client-Branch cube**

SELECT B.ZONE,B.STATE\_NEW,C.FISCAL\_YEAR,C.FISCAL\_QUARTER,C.FISCAL\_QUARTER\_SHORT\_DESC

, SUM(F.NO\_OF\_EMPS) as NO\_OF\_EMPS FROM MST\_CALENDAR C,NEW\_FACT\_BRANCH\_PL F,NEW\_DIM\_BRANCHWISE B

WHERE F.BRANCHWISE\_SK = B.BRANCHWISE\_SK AND F.DATE\_SK=C.DATE\_TK

GROUP BY B.ZONE,B.STATE\_NEW,C.FISCAL\_YEAR,C.FISCAL\_QUARTER,C.FISCAL\_QUARTER\_SHORT\_DESC;