ECOMMERCE SALES DATA ANALYSIS

Big Data Analytics

Dharshan Kumar K S
Siva Prakash R

DATASET

Data from 04-09-2016 to 03-09-2018
Dataset rows : 1,16,573
Dataset columns : 21
Dataset size : 27.4 MB

order_id	customer_id	quantity	price_MRP	payment	timestamp	rating	product_category	product_id	payment_type	order_status	product_weig	product_length	product_height	product_width_	customer_city	customer_state	seller_id	seller_city	seller_state	payment_installment
9045fa8	2f8d6af8						health_be	0a37e05									ccc4bbb5			
41de351	cb3875d6	1	1074.38	1095.65	21-11-2017 22:34	5	auty	528984f	credit_card	delivered	800	40	20	30	osasco	SP	f32a6ab2	curitiba	PR	5
404c087	1ce70910						health_be	67473aa							campos		ccc4bbb5			
c1f6618	bc164814	1	145	161.71	24-02-2017 22:55	5	auty	97e981b	credit_card	delivered	400	38	12	25	dos	RJ	f32a6ab2	curitiba	PR	7
d6d7c43	5b477d52						health_be	67473aa							nova		ccc4bbb5			
1275f00	5dd0c1eb	1	145	161.71	19-01-2017 14:28	4	auty	97e981b	credit_card	delivered	400	38	12	25	friburgo	RJ	f32a6ab2	curitiba	PR	1
c0e0261	c2f18177						health_be	b60a0c8							porto		ccc4bbb5			
3bf1537	8a120395	1	555	585.95	01-06-2018 22:22	4	auty	bd0333c	boleto	delivered	650	16	10	11	velho	RO	f32a6ab2	curitiba	PR	1
1bf38e3	21a9772e						health_be	0846252							montes		ccc4bbb5			
450f15b	10934cf4	1	226.8	250.32	15-06-2018 20:36	5	auty	8607b71	credit_card	delivered	650	16	10	11	claros	MG	f32a6ab2	curitiba	PR	8
d1ff908	190508c5						health_be	0846252							sao		ccc4bbb5			
b4e21d4	83e9da28	1	207.9	238.61	04-08-2018 21:57	5	auty	8607b71	credit_card	delivered	650	16	10	11	goncalo	RN	f32a6ab2	curitiba	PR	8
591eeea	ed0e1816						health_be	2483416							uberlandi		ccc4bbb5			
2a5c1e6	a16e9dc9	1	860	882.46	20-06-2017 14:53	4	auty	c14aa8d	credit_card	delivered	500	40	20	30	a	MG	f32a6ab2	curitiba	PR	10
c1808aa	2d88e5d5						health_be	2483416									ccc4bbb5			
ca844e2	23368eaa	1	589	606.87	24-11-2017 10:23	5	auty	c14aa8d	credit_card	delivered	500	40	20	30	gravatai	RS	f32a6ab2	curitiba	PR	1
4da0b6d	6d51e47d						health_be	2483416							ribeirao		ccc4bbb5			
6c8c1ec	2de21f74	1	589	607.87	28-11-2017 09:37	3	auty	c14aa8d	credit_card	delivered	500	40	20	30	preto	SP	f32a6ab2	curitiba	PR	10
5d4f3f9	44ce310a						health_be	2483416							lagoa dos		ccc4bbb5			
b6abf5c	65512fae	1	650	696.58	06-12-2017 13:05	4	auty	c14aa8d	credit_card	delivered	500	40	20	30	gatos	PE	f32a6ab2	curitiba	PR	3
9f1dcbf0	9b9681cf						health_be	76951ac									ccc4bbb5			
e12d0df	b00f0a6f	1	1597.35	1650.56	20-08-2017 11:40	5	auty	b342040	credit_card	delivered	650	40	20	30	maragogi	AL	f32a6ab2	curitiba	PR	7

DATASET DESCRIPTION

S.No	Name	Description
1	order_id	unique id for each order (32 fixed-size number)
2	customer_id	unique id for each customer (32 fixed-size number)
3	quantity	1-21
4	price_MRP	cost price, 0.85-6735
5	payment	selling price, 0-13664.8
6	timestamp	order purchase time (local, day-month-year hour:min:sec AM/PM)
7	rating	1-5
8	product_category	category under which product belongs
9	product_id	unique id for each product (32 fixed-size number)
10	payment_type	Type of payment - credit card/debit card/boleto/voucher
11	order_status	delivered/shipped/invoiced
12	product_weight_g	weight of product (in grams), 0-40425
13	product_length_cm	length of product (in centimeter), 7-105
14	product_height_cm	height of product (in centimeter), 2-105
15	product_width_cm	width of product (in centimeter), 6-118
16	customer_city	city where order is placed
17	customer_state	state where order is placed
18	seller_id	unique id for each seller (32 fixed-size number)
19	seller_city	city where order is picked up
20	seller_state	state where order is picked up
21	payment_installments	no. of installments taken by customer to pay bill, 0-24

ANALYSIS

1. **Customer Segmentation**
   Categorizing customers on their spendings
   [Bar-graph]

2. **Monthly Trend Forecasting**
   Visualising the monthly trend of sales
   [Bar-graph]

3. **Hourly Sales Analysis**
   Which hour has more no. of sales?
   [Timeseries-Plot]

4. **Payment Preference**
   What are the most commonly used payment types?
   Count of Orders With each No. of Payment Installments
   [Pie-Chart]

5. **Potential Customer's Location**
   Where do most customers come from?
   [Pie-chart]

6. **Seller Rating**
   Which seller sold more?
   Which seller got more rating?
   [Bar-graph]

ANALYSIS

7. **Product Based Analysis**
   - Which category product has sold more?
   - Which category product has more rating?
   - Which product has sold more?
   - Top 10 highest & least product rating?
   - Order Count for each rating
   - [Bar-graph]

8. **Logistics based Optimization Insights**
   - Which city buys heavy weight products and low weight products?
   - [Pie-chart]
   - How much products sold within seller state?
   - [Bar-graph]

9. **Machine Learning Model**
   - Predicting future sales
   - ML - Linear regression

Pre-Processing

```
val eRdd = sc.textFile("ecom_data.csv")
// eRdd: org.apache.spark.rdd.RDD[String] = ecom_data_2.csv MapPartitionsRDD[1] at textFile at <console>:24

// To remove 1st header row
val eRdd1 = eRdd.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
// eRdd1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at mapPartitionsWithIndex at <console>:26

val ePairRdd = eRdd1.map{ l =>
      val str0 = l.split(',')
      val (oid,cid,qt,cp,sp) = (str0(0),str0(1),str0(2).toInt,str0(3).toFloat,str0(4).toFloat)
      val (ts,rating,pCat,pId,sp_type) = (str0(5),str0(6).toInt,str0(7),str0(8),str0(9))
      val (oStat,pWt,pLen,pHt,pWidth) = (str0(10),str0(11).toInt,str0(12).toInt,str0(13).toInt,str0(14).toInt)
      val (cCity,cState,sId,sCity,sState) = (str0(15),str0(16),str0(17),str0(18),str0(19))
      val (sInstal) = (str0(20).toInt)
      (oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal) }
// ePairRdd: org.apache.spark.rdd.RDD[(String, String, Int, Float, Float, String, Int, String, String, String, String,
//       Int, Int, Int, Int, String, String, String, String, String, Int)] = MapPartitionsRDD[3] at map at <console>:28
```

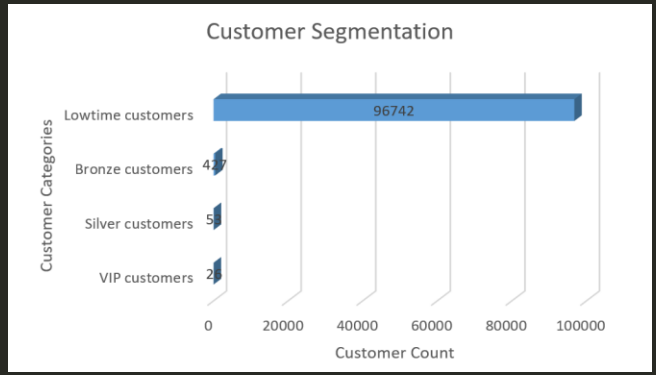
oid	cid	qt	cp	sp	ts	rating	pCat	pId	sp_type	oStat	pWt	pLen	pHt	pWidth	cCity	cState	sId
9045fa841de3514ce...	2f8d6af0cb3875d69...	1	1074.38	1095.65	21-11-2017 22:34	5	health_beauty	0a37e0552898447c...	credit_card	delivered	800	40	20	30	osasco	SP	ccc4bbb5f32a6ab2b...
404c087c1f661898f...	1ce70910bc164814...	1	145.0	161.71	24-02-2017 22:55	4	health_beauty	67473aa97e981b3b8...	credit_card	delivered	400	38	12	25	campos dos goytac...	RJ	ccc4bbb5f32a6ab2b...
d6d7c431275f0029d...	5b477d525dd0c1eb9...	1	145.0	161.71	19-01-2017 14:28	4	health_beauty	67473aa97e981b3b8...	credit_card	delivered	400	38	12	25	nova friburgo	RJ	ccc4bbb5f32a6ab2b...
c0e02613bf15378f1...	c2f181778a1203953...	1	555.0	585.95	01-06-2018 22:22	4	health_beauty	b60a0c8bd0333c69e...	boleto	delivered	650	16	10	11	porto velho	RO	ccc4bbb5f32a6ab2b...
1bf38e3450f15beea...	21a9772e10934cf45...	1	226.8	250.32	15-06-2018 20:36	5	health_beauty	08462528607b71ea6...	credit_card	delivered	650	16	10	11	montes claros	MG	ccc4bbb5f32a6ab2b...
d1ff908b4e21d4fff...	190508c583e9da289...	1	207.9	238.61	04-08-2018 21:57	5	health_beauty	08462528607b71ea6...	credit_card	delivered	650	16	10	11	sao goncalo do am...	RN	ccc4bbb5f32a6ab2b...
591eeea2a5c1e61b8...	ed0e1816a16e9dc92...	1	860.0	882.46	20-06-2017 14:53	4	health_beauty	2483416c14aa8d93d...	credit_card	delivered	500	40	20	30	uberlandia	MG	ccc4bbb5f32a6ab2b...
c1808aaca844e2210...	2d88e5d523368eaa7...	1	589.0	606.87	24-11-2017 10:23	5	health_beauty	2483416c14aa8d93d...	credit_card	delivered	500	40	20	30	gravatai	RS	ccc4bbb5f32a6ab2b...
4da0b6d6c8c1ecb45...	6d51e47d2de21f745...	1	589.0	607.87	28-11-2017 09:37	3	health_beauty	2483416c14aa8d93d...	credit_card	delivered	500	40	20	30	ribeirao preto	SP	ccc4bbb5f32a6ab2b...
5d4f3f9b6abf5c659...	44ce310a65512faeb...	1	650.0	696.58	06-12-2017 13:05	4	health_beauty	2483416c14aa8d93d...	credit_card	delivered	500	40	20	30	lagoa dos gatos	PE	ccc4bbb5f32a6ab2b...
9f1dcbf0e12d0df2e...	9b9681cfb00f0a6f1...	1	1597.35	1650.56	20-08-2017 11:40	5	health_beauty	76951acb34204078c...	credit_card	delivered	650	40	20	30	maragogi	AL	ccc4bbb5f32a6ab2b...
e21a007c09bc0fffa...	e3e4e76e7ffd5eb15...	1	1597.35	1629.33	18-07-2017 14:26	5	health_beauty	76951acb34204078c...	credit_card	delivered	650	40	20	30	salvador	BA	ccc4bbb5f32a6ab2b...
09744659d8e64274c...	75321717fe3af2a31...	1	364.0	428.8	16-07-2018 12:41	5	health_beauty	6cdd53843498f9289...	credit_card	delivered	900	25	12	38	belo jardim	PE	ccc4bbb5f32a6ab2b...
0713f1dd3cdb0f62e...	e1e49b430a59e7067...	1	349.9	365.37	01-08-2017 07:31	5	health_beauty	6cdd53843498f9289...	boleto	delivered	900	25	12	38	cascavel	PR	ccc4bbb5f32a6ab2b...
f810273bc84ceae52...	6b8d7030c9c549219...	1	349.9	381.88	22-02-2018 17:06	4	health_beauty	6cdd53843498f9289...	credit_card	delivered	900	25	12	38	recife	PE	ccc4bbb5f32a6ab2b...
b2bc998b56f86f7a3...	c6747b392110a7fba...	1	364.0	398.69	19-06-2018 20:01	2	health_beauty	6cdd53843498f9289...	credit_card	shipped	900	25	12	38	aracaju	SE	ccc4bbb5f32a6ab2b...
897f61323b0f93ac1...	790edc01cc91b3933...	1	349.9	369.6	05-02-2018 17:04	5	health_beauty	6cdd53843498f9289...	credit_card	delivered	900	25	12	38	marilia	SP	ccc4bbb5f32a6ab2b...
91a31ae0b2ce6ff70...	a7e7ac71f48352583...	1	349.9	378.89	08-06-2017 17:50	5	health_beauty	6cdd53843498f9289...	boleto	delivered	900	25	12	38	guacui	ES	ccc4bbb5f32a6ab2b...
6ae8d599fd407bfba...	43e4c6f57db5fdfea...	1	349.9	379.2	22-07-2017 12:59	4	health_beauty	6cdd53843498f9289...	credit_card	delivered	900	25	12	38	aracaju	SE	ccc4bbb5f32a6ab2b...
4389819b71f1f2356...	0d3740ed03979ba91...	1	349.9	373.17	04-04-2018 12:47	4	health_beauty	6cdd53843498f9289...	credit_card	delivered	900	25	12	38	itu	SP	ccc4bbb5f32a6ab2b...

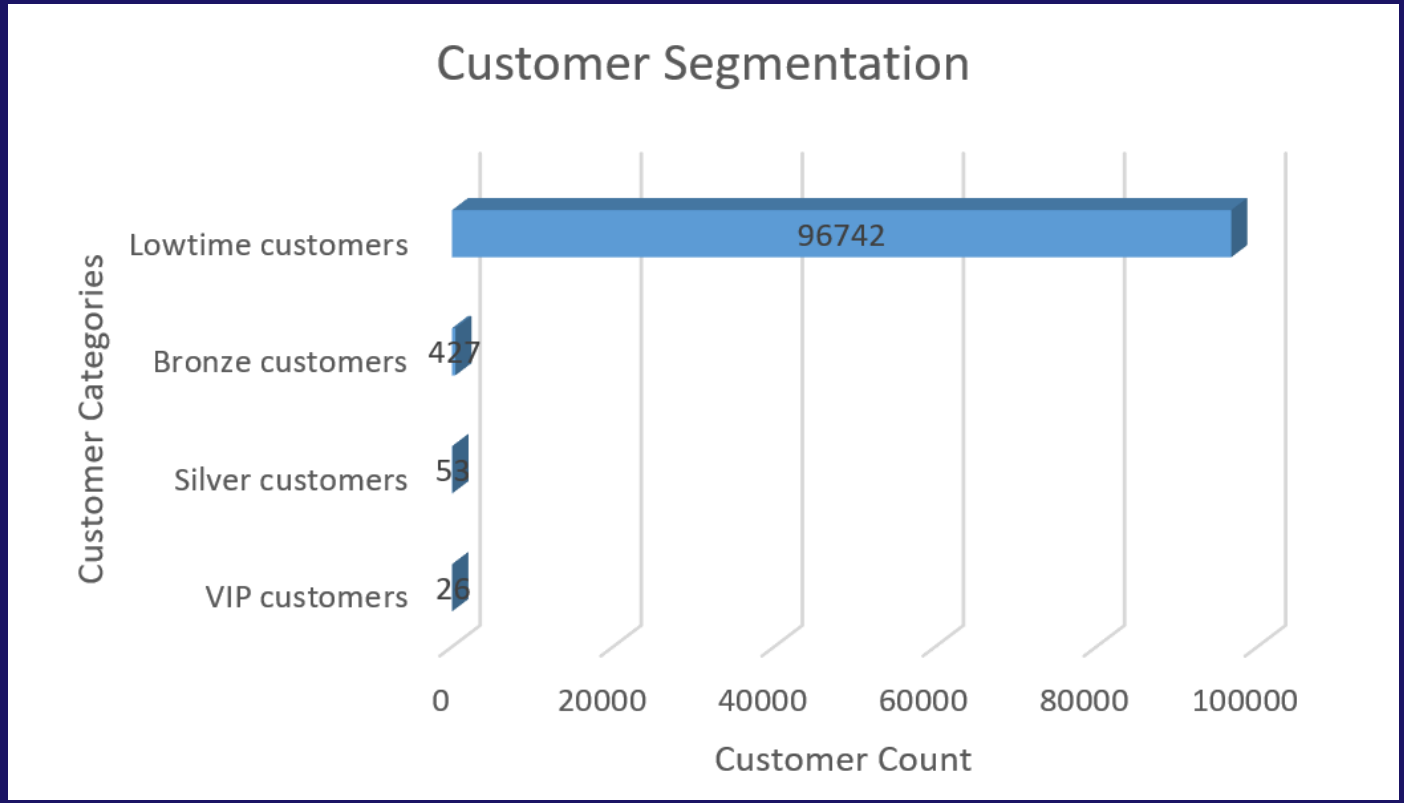
01

Customer Segmentation

```
val ePairRdd2 = ePairRdd.map{ case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal) =>
      val amount = qt*sp
      val amount_roundoff = (math rint amount*100)/100
      (cid,amount_roundoff) }.reduceByKey(_+_).sortBy(_._2,false)
// ePairRdd2: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[14] at sortBy at <console>:35

ePairRdd2.toDF("Customer ID","Bill Amount").show
/*
+--------------------+-----------------+
|         Customer ID|      Bill Amount|
+--------------------+-----------------+
|1617b1357756262bf...|        491906.88|
|bd5d39761aa56689a...|         475188.0|
|be1b70680b9f9694d...|         462504.0|
|1ff773612ab8934db...|         166023.0|
|10de381f8a8d23fff...|         147078.0|
|8c20d9bfbc96c5d39...|         131244.3|
|05455dfa7cd02f13d...|        127712.34|
|e7d6802668de6e74d...|        122906.29|
|91f92cfee46b79581...|        115614.72|
|be1c4e52bb71e0c54...|         112721.4|
|d5f2b3f597c7ccafb...|         106472.1|
|daf15f1b940cc6a72...|         97263.66|
|4a60b2ce1ee8c7b82...|          97190.5|
|adb32467ecc74b535...|          93960.0|
|78fc46047c4a639e8...|          79205.5|
|cb87122c4871e2027...|          76534.7|
|0c792d32a3251b4f6...|          73411.2|
|88324c93ce11436ae...|         72887.85|
|ec5b2ba62e5743423...|          72748.8|
|f4c13379ddd0ed4f4...|         69188.46|
+--------------------+-----------------+
only showing top 20 rows
*/
```

```scala
val all_customers = ePairRdd2.map{ case (cid,amount) =>
      if (amount>=50000) {( "VIP customers", List((cid,amount)) )}
      else if ((amount>=20000)&&(amount<50000)) {( "Silver customers", List((cid,amount)) )}
      else if ((amount>=5000) &&(amount<20000)) {( "Bronze customers", List((cid,amount)) )}
      else {( "Lowtime customers", List((cid,amount)) )}
      }.reduceByKey(_ ++ _)
// all_customers: org.apache.spark.rdd.RDD[(String, List[(String, Double)])] = ShuffledRDD[19] at reduceByKey at <console>:39
all_customers.toDF("Customer Type","Customer_id & Bill amount").show
/*
+----------------+--------------------------------------------------------+
|    Customer Type|                                Customer_id & Bill amount|
+----------------+--------------------------------------------------------+
|Lowtime customers|[[99a32bf8f0c54702217b584a4d220761,4992.119999999999], [...|
|    VIP customers|[[1617b1357756262bfa56ab541c47bc16,491906.88], [bd5d3976...|
| Silver customers|[[159132ab31eb3c8a1061ed18967f720b,45873.8], [fc3d1daec3...|
| Bronze customers|[[17f9863585a471e9ffe77c4d4f26ecea,19831.84], [fb51887c8...|
+----------------+--------------------------------------------------------+
*/

val all_customers_count = all_customers.map { case (customer_type,list)=> (customer_type, list.length) }.sortBy(_._2)
// all_customers_count: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[28] at sortBy at <console>:36
all_customers_count.toDF("Customer Type","Customer count").show
/*
+----------------+--------------+
|    Customer Type|Customer count|
+----------------+--------------+
|    VIP customers|            26|
| Silver customers|            53|
| Bronze customers|           427|
|Lowtime customers|         96742|
+----------------+--------------+
*/
```



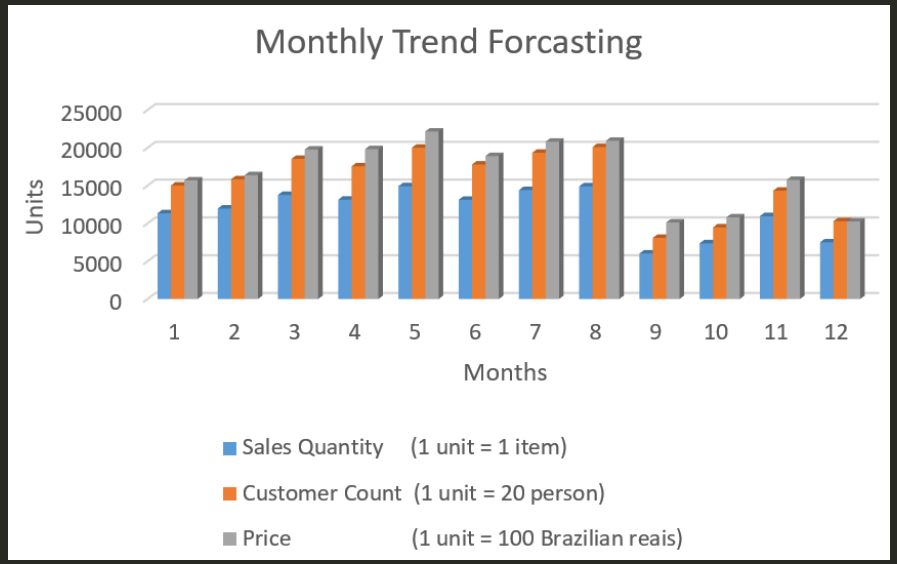

Customer Segmentation
Customer Categories
Lowtime customers
Bronze customers
Silver customers
VIP customers
96742
427
53
26
0
20000
40000
60000
80000
100000
Customer Count

02

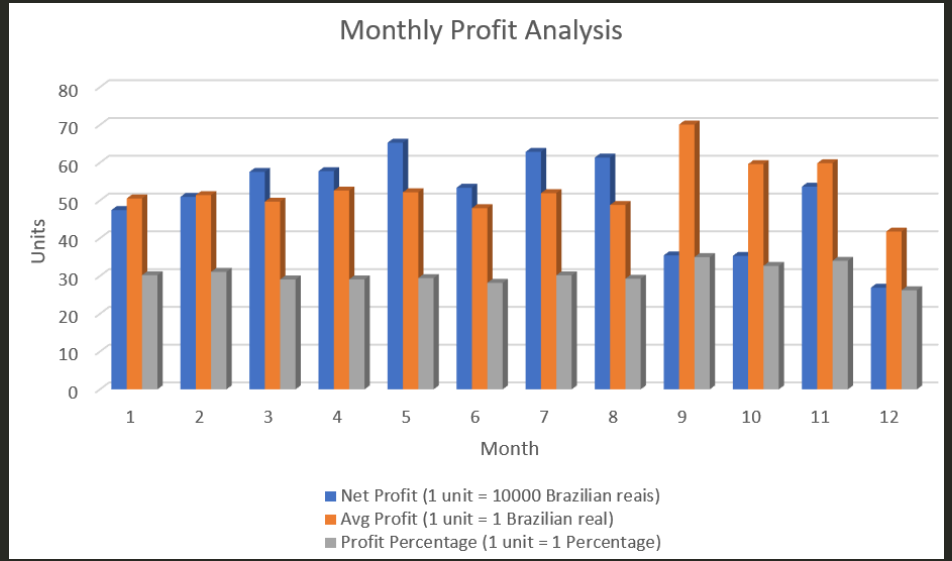
Monthly Trend Forecasting

```scala
val all_months_count = ePairRdd.map{ case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal) =>
      val ds_ts = ts.split(" ")
      val month = ds_ts(0).split("-")
      (month(1).toInt,qt) }.reduceByKey(_+_).sortBy(_._1)
// all_months_count: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[44] at sortBy at <console>:33

val monthly_customers_count = ePairRdd.map{ case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal) =>
      val ds_ts = ts.split(" ")
      val month = ds_ts(0).split("-")
      (month(1).toInt,cid) }.reduceByKey(_++_).map { case (month,cid)=> (month,cid.length) }.sortBy(_._1)
// monthly_customers_count: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[57] at sortBy at <console>:32

val monthly_price = ePairRdd.map{ case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal) =>
      val ds_ts = ts.split(" ")
      val month = ds_ts(0).split("-")
      (month(1).toInt,sp) }.reduceByKey(_+_).sortBy(_._1)
// monthly_price: org.apache.spark.rdd.RDD[(Int, Float)] = MapPartitionsRDD[64] at sortBy at <console>:33
```

```scala
// Joining 3 RDDs
val monthly_analysis_joint = (all_months_count.join(monthly_customers_count)).join(monthly_price).sortBy(_._1)
// monthly_analysis_joint: org.apache.spark.rdd.RDD[(Int, ((Int, Int), Float))] = MapPartitionsRDD[466] at sortBy at <console>:43

val monthly_analysis = monthly_analysis_joint.map{ case(month,q_c_p)=>
      val price = q_c_p._2
      val q_c = q_c_p._1
      val qt = q_c._1
      val customer_count = q_c._2
      (month,qt,customer_count,price) }
// monthly_analysis_joint: org.apache.spark.rdd.RDD[(Int, ((Int, Int), Float))] = MapPartitionsRDD[75] at sortBy at <console>:40
monthly_analysis.toDF("Month","Quantity","Customer count","Price").show
/*
+-----+--------+--------------+---------+
|Month|Quantity|Customer count|    Price|
+-----+--------+--------------+---------+
|    1|   11347|        300512|1570399.0|
|    2|   11972|        317120|1638516.1|
|    3|   13779|        370784|1978603.5|
|    4|   13150|        351264|1983463.4|
|    5|   14940|        400480|2216750.5|
|    6|   13140|        356096|1892519.4|
|    7|   14417|        387360|2082414.8|
|    8|   14914|        402208|2094366.4|
|    9|    6019|        161888|1013045.5|
|   10|    7375|        189632|1081166.9|
|   11|   10972|        286816|1576857.2|
|   12|    7508|        206176|1026283.5|
+-----+--------+--------------+---------+
*/
```



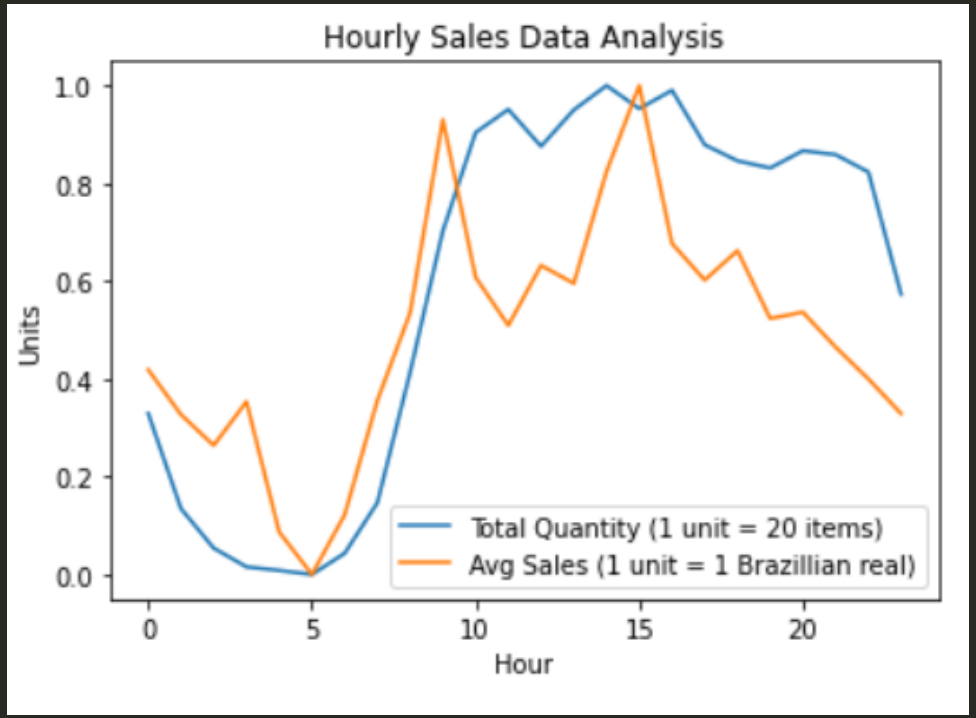
```
val pRdd=ePairRdd.map{case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal)=>
       val t1=ts.split(" ")(0).split("-")(1)
       (t1,(List(sp-cp),List(sp)))}.reduceByKey{case(a,b)=>(a._1++b._1,a._2++b._2)}.map{case(a,(b,c))=>
       val ps=b.sum
       val ss=c.sum
       val n=b.length
      (a,ps,ps/n,ps*100.0/ss)}.sortBy(x=>x._1)
// pRdd: org.apache.spark.rdd.RDD[(Int, Float, Float, Double)] = MapPartitionsRDD[101] at sortBy at <console>:36

pRdd.toDF("Month","Net Profit","Avg Profit","Profit Percentage").show
/*
+-----+----------+----------+------------------+
|Month|Net Profit|Avg Profit| Profit Percentage|
+-----+----------+----------+------------------+
|    1| 474628.84|  50.54082|30.223571026488273|
|    2| 509555.25|  51.41829| 31.09874382631755|
|    3|  575717.6| 49.686512|29.097068946684363|
|    4|  577632.4| 52.622063|29.122411952779316|
|    5|  652936.8| 52.172337|29.454731917392007|
|    6| 533786.75| 47.967896| 28.20511939851539|
|    7| 628680.44|   51.9356|30.189937930903252|
|    8| 613505.25|  48.81098|29.293048057615376|
|    9| 354642.28| 70.101265|35.007539456985114|
|   10|  353360.9| 59.628906|32.683278766536816|
|   11| 536567.44| 59.864716| 34.02762391391047|
|   12| 269167.22|  41.77669| 26.22733314487044|
+-----+----------+----------+------------------+
*/
```



03

Hourly Sales Analysis

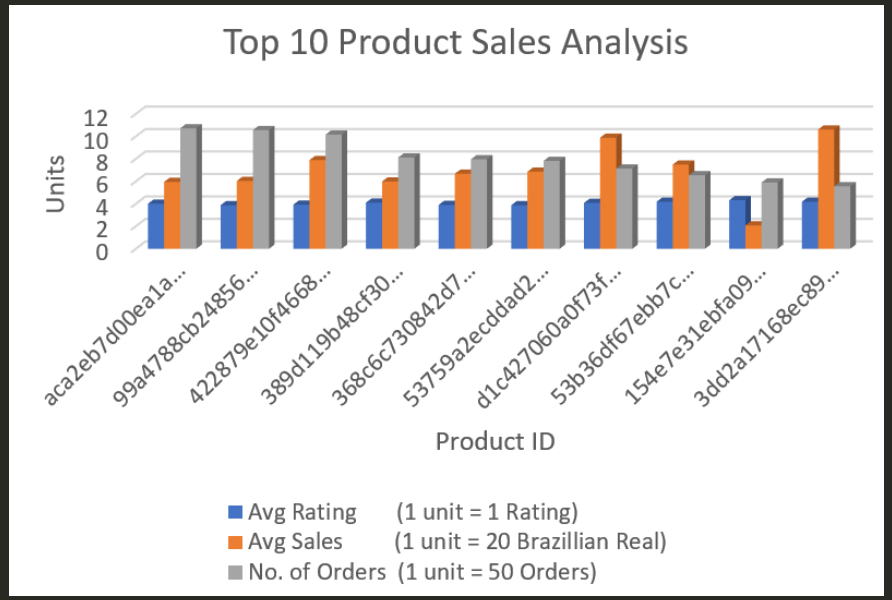
```scala
val TRdd=ePairRdd.map{case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal)=>
      val t1=ts.split(" ")(1).split(":")(0)
      (t1,(List(qt),List(sp)))}.reduceByKey{case(a,b)=>(a._1++b._1,a._2++b._2)}.map{case(a,(b,c))=>(a,b.sum,c.sum/c.length)}.sortBy(_._1)
// TRdd: org.apache.spark.rdd.RDD[(Int, Int, Float)] = MapPartitionsRDD[112] at sortBy at <console>:32

TRdd.toDF("Hour","Quantity","Avg Price").show
/*
+----+--------+---------+
|Hour|Quantity|Avg Price|
+----+--------+---------+
|   0|    3357|161.53378|
|   1|    1520|155.86949|
|   2|     757|151.99045|
|   3|     396|157.50314|
|   4|     327| 141.0595|
|   5|     249|135.69179|
|   6|     657| 143.2227|
|   7|    1635|157.67194|
|   8|    4151|168.71812|
|   9|    6882|193.09906|
|  10|    8781|  173.212|
|  11|    9228| 167.1342|
|  12|    8515|174.70575|
|  13|    9217|172.44116|
|  14|    9688|186.55557|
|  15|    9241| 197.4104|
|  16|    9596|177.47131|
|  17|    8544|172.85225|
|  18|    8235|176.57355|
|  19|    8096|168.01033|
+----+--------+---------+
only showing top 20 rows
*/
```



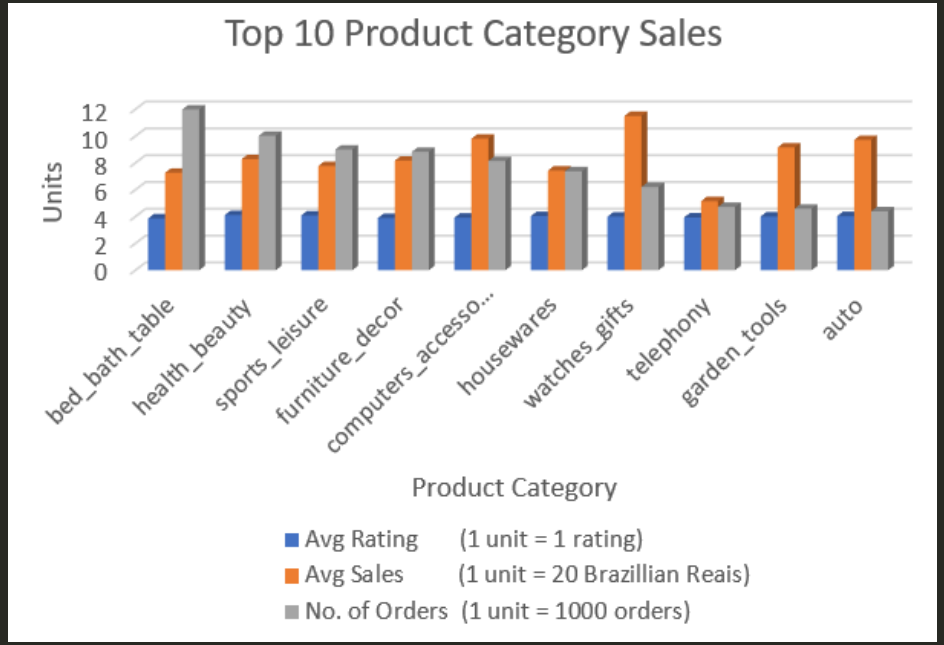
04

Product Based Analysis

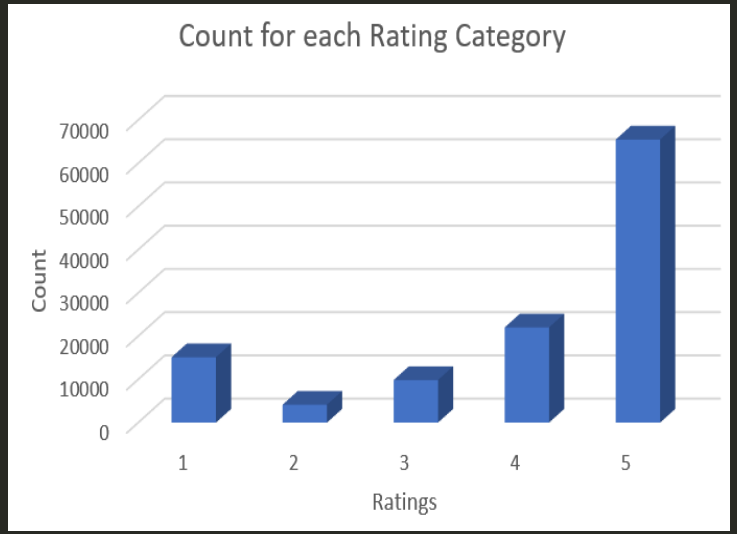
```
val prodRdd=ePairRdd.map{case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal)=>
      (pId,(List(rating),List(sp)))}.reduceByKey{case(a,b)=>(a._1++b._1,a._2++b._2)}.map{case(a,(b,c))=>
      (a,b.sum*1.0/b.length,c.sum/c.length,c.length)}
// prodRdd: org.apache.spark.rdd.RDD[(String, Double, Float, Int)] = MapPartitionsRDD[142] at map at <console>:31

// Total number of unique products sold:
prodRdd.count
//res28: Long = 6068

val pSorted=prodRdd.sortBy( x => (x._4), ascending = true).collect.toList
// Top-10 Most Sold products:
pSorted.reverse.toDF("Product ID","Avg Rating","Avg Sales","No. of Orders").show
/*
+--------------------+-----------------+----------+-------------+
|          Product ID|       Avg Rating| Avg Sales|No. of Orders|
+--------------------+-----------------+----------+-------------+
|aca2eb7d00ea1a7b8...| 4.009328358208955| 119.00755|          536|
|99a4788cb24856965...|3.8768939393939394|120.632454|          528|
|422879e10f4668299...| 3.923228346456693| 157.77785|          508|
|389d119b48cf3043d...| 4.100985221674877| 119.74496|          406|
|368c6c730842d7801...|3.9020100502512562| 133.42976|          398|
|53759a2ecddad2bb8...| 3.877237851662404| 137.05765|          391|
|d1c427060a0f73f6b...| 4.078431372549019| 197.64127|          357|
|53b36df67ebb7c415...|  4.18348623853211| 149.82967|          327|
|154e7e31ebfa09220...|  4.31864406779661|  41.60651|          295|
|3dd2a17168ec895c7...| 4.183453237410072| 212.09384|          278|
|2b4609f8948be1887...| 4.087272727272727| 106.26064|          275|
|7c1bd920dbdf22470...| 3.847107438016529|  80.54882|          242|
|a62e25e09e05e6faf...|3.8552631578947367| 277.04956|          228|
|bb50f2e236e5eea01...| 4.219047619047619| 391.55322|          210|
|e0d64dcfaa3b6db5c...| 3.702970297029703| 180.54382|          202|
|e53e557d5a159f5aa...|3.4583333333333335| 165.92116|          192|
|35afc973633aaeb6b...|3.9576719576719577| 128.44373|          189|
|42a2c92a0979a949c...| 3.924324324324324| 99.171715|          185|
|b532349fe46b38fbc...|3.5580110497237567|125.344475|          181|
|f1c7f353075ce59d8...| 4.304878048780488| 215.97105|          164|
+--------------------+-----------------+----------+-------------+
only showing top 20 rows
*/
```

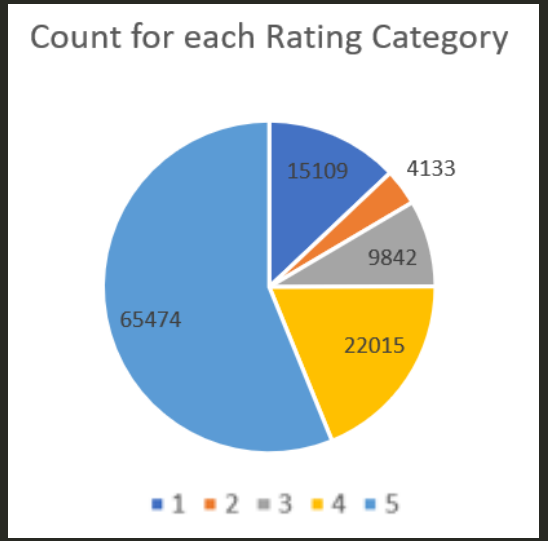


```
val pCatRdd=ePairRdd.map{case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal)=>
      (pCat,(List(rating),List(sp)))}.reduceByKey{case(a,b)=>(a._1++b._1,a._2++b._2)}.map{case(a,(b,c))=>
      (a,b.sum*1.0/b.length,c.sum/c.length,c.length)}
// pCatRdd: org.apache.spark.rdd.RDD[(String, Double, Float, Int)] = MapPartitionsRDD[158] at map at <console>:31

//Total number of unique product categories sold
pCatRdd.count
//res23: Long = 71

val cSorted=pCatRdd.sortBy( x => (x._4), ascending = true).collect.toList
//Top 10 categories sold
cSorted.reverse.toDF("Product Category","Avg Rating","Avg Sales","No. of Orders").show
/*
+--------------------+-----------------+---------+-------------+
|    Product Category|       Avg Rating|Avg Sales|No. of Orders|
+--------------------+-----------------+---------+-------------+
|      bed_bath_table|3.8670558798999166|145.47154|        11990|
|       health_beauty| 4.119142572283151|165.80989|        10030|
|      sports_leisure| 4.092837312604109|155.49915|         9005|
|     furniture_decor| 3.895279067134609|163.52063|         8833|
|computers_accesso...| 3.926144031407189|196.24254|         8151|
|          housewares| 4.045257452574526|148.76648|         7380|
|       watches_gifts| 4.000483014007406|230.19086|         6211|
|           telephony| 3.938214134574693|103.08746|         4726|
|        garden_tools| 4.011328976034858|183.16377|         4590|
|                auto|   4.0386803185438|194.36702|         4395|
|                toys| 4.142490072412988|144.82033|         4281|
|         cool_stuff| 4.125781445361341|195.53168|         3999|
|           perfumery|  4.12287545277236|141.69337|         3589|
|                baby|3.9953168904152356|168.09666|         3203|
|         electronics| 4.027397260273973|91.366974|         2847|
|          stationery| 4.176380952380953|121.18257|         2625|
|fashion_bags_acce...|4.1165898617511525|101.29753|         2170|
|           pet_shop| 4.188669950738916|153.48903|         2030|
|    office_furniture|3.5167785234899327|364.66248|         1788|
|      consoles_games|3.9916107382550337|164.45575|         1192|
+--------------------+-----------------+---------+-------------+
only showing top 20 rows
*/
```



```scala
//Ratings given
val ratingRdd=ePairRdd.map{case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal)=>
      (rating,1)}.reduceByKey(_+_)
//ratingRdd: org.apache.spark.rdd.RDD[(Int, Int)] = ShuffledRDD[98] at reduceByKey at <console>:26

ratingRdd.sortBy(x=>x._1).toDF("Rating","Count of Ratings").show
/*
+------+----------------+
|Rating|Count of Ratings|
+------+----------------+
|     1|           15109|
|     2|            4133|
|     3|            9842|
|     4|           22015|
|     5|           65474|
+------+----------------+
*/
```

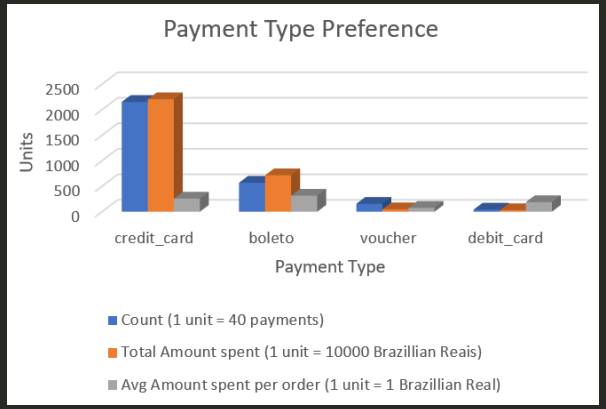


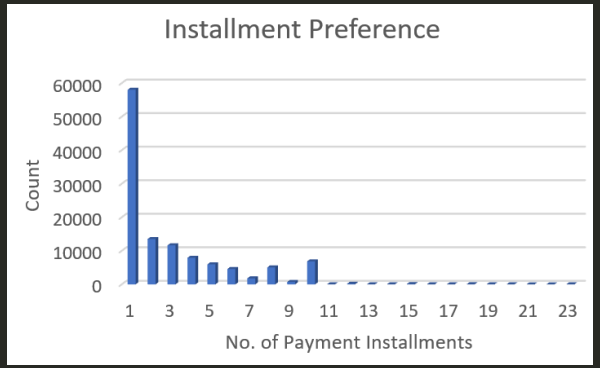


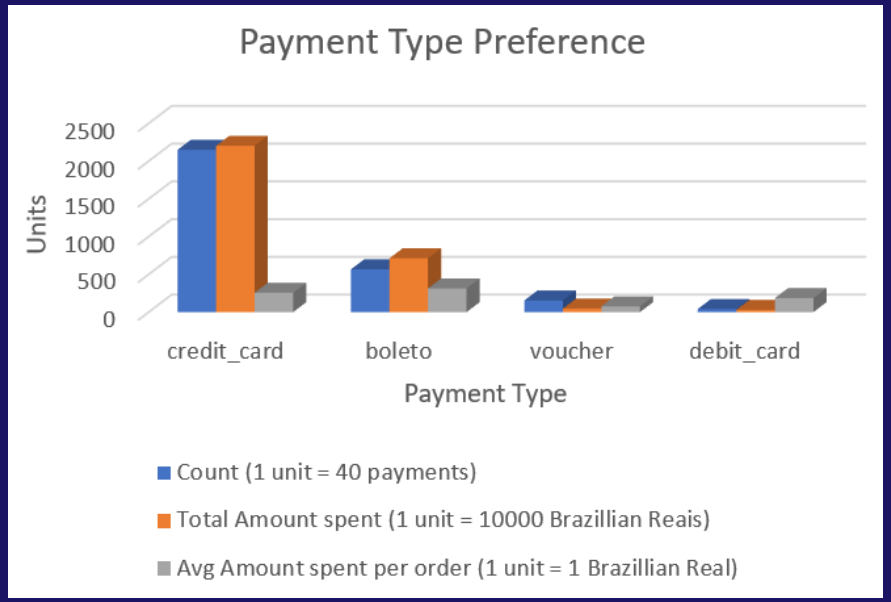
05

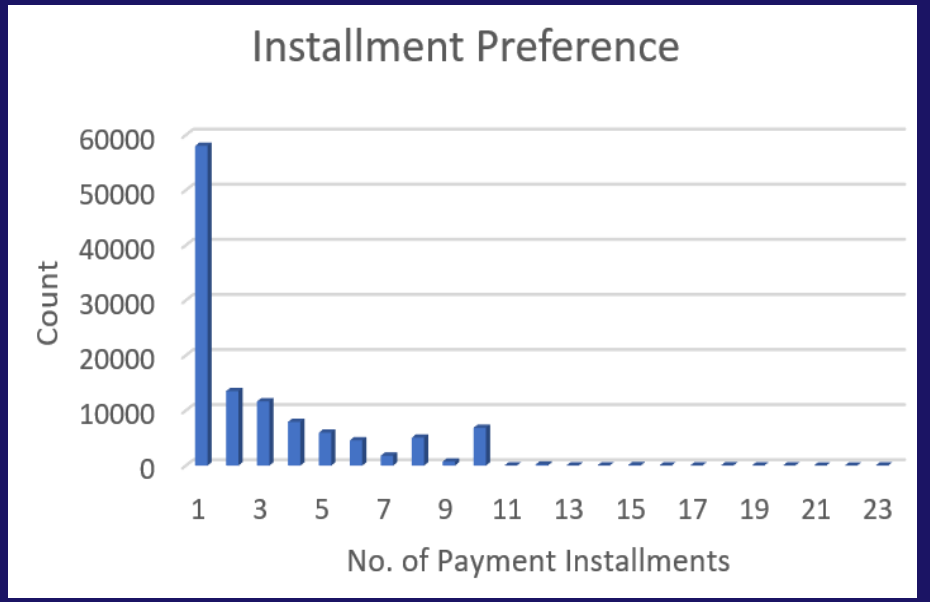
Payment Preference

```
val prodRdd11 = ePairRdd.map{case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal)=>
      (sp_type,(List(1),List(sp*qt)))}.reduceByKey{case(a,b)=>(a._1++b._1,a._2++b._2)}.map{case(sp_type,(qt,price))=>
      (sp_type,qt.length,price.sum,price.sum/qt.length)}.sortBy(_._2,false)
// ePairRdd11: org.apache.spark.rdd.RDD[(String, Int, Float, Float)] = MapPartitionsRDD[210] at map at <console>:36
ePairRdd11.toDF("Payment Type","Count","Total Amount spent","Avg Amount spent per order").show
/*
+------------+-----+------------------+-------------------------+
|Payment Type|Count|Total Amount spent|Avg Amount spent per order|
+------------+-----+------------------+-------------------------+
| credit_card|86004|       2.2028932E7|                256.13846|
|      boleto|22692|         7130513.5|                 314.2303|
|     voucher| 6211|         476525.38|                 76.72281|
|  debit_card| 1666|          310508.3|                186.37953|
+------------+-----+------------------+-------------------------+
*/
// Count of Orders With each No. of Payment Installments:
val ePairRdd12 = ePairRdd.map{ case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal) => (sInstal,1) }.reduceByKey(_+_).sortBy(_._1)
// ePairRdd12: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[225] at sortBy at <console>:32
ePairRdd12.toDF("No. of Payment Installments","Count for each installment").show
/*
+-------------------------+-------------------------+
|No. of Payment Installments|Count for each installment|
+-------------------------+-------------------------+
|                        1|                    58030|
|                        2|                    13515|
|                        3|                    11650|
|                        4|                     7926|
|                        5|                     5982|
|                        6|                     4587|
|                        7|                     1803|
|                        8|                     5055|
|                        9|                      724|
|                       10|                     6864|
|                       11|                       25|
|                       12|                      166|
|                       13|                       19|
|                       14|                       16|
|                       15|                       93|
|                       16|                        7|
|                       17|                        7|
|                       18|                       38|
|                       20|                       21|
+-------------------------+-------------------------+
only showing top 20 rows
*/
```






Payment Type Preference
Units
2500
2000
1500
1000
500
0
credit_card
boleto
voucher
debit_card
Payment Type
Count (1 unit = 40 payments)
Total Amount spent (1 unit = 10000 Brazillian Reais)
Avg Amount spent per order (1 unit = 1 Brazillian Real)


Installment Preference
Count
60000
50000
40000
30000
20000
10000
0
1
3
5
7
9
11
13
15
17
19
21
23
No. of Payment Installments

06

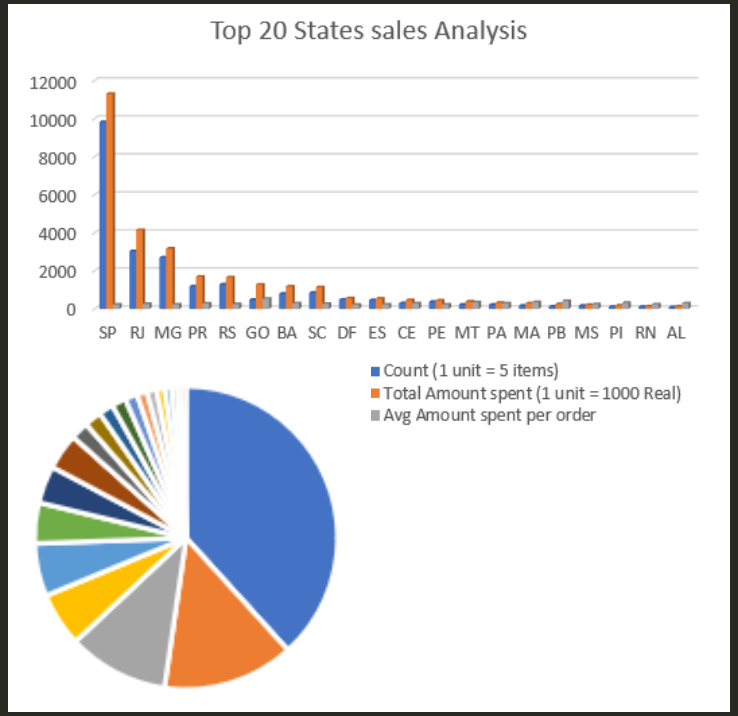
Seller Ranking

```
val sellerRankRdd = ePairRdd.map{case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal)=>
      (sId,(List(1),List(sp*qt),List(rating)))}.reduceByKey{case(a,b)=>
      (a._1++b._1,a._2++b._2,a._3++b._3)}.map{case(sp_type,(qt,price,rating_all))=>
      val rating = (rating_all.sum/qt.sum).toFloat
      (sp_type,qt.sum,price.sum,rating)}.sortBy(_._3,false)
// sellerRankRdd: org.apache.spark.rdd.RDD[(String, Int, Float, Float)] = MapPartitionsRDD[264] at sortBy at <console>:45
sellerRankRdd.toDF("Seller ID","Customers Reached","Total Sales","Avg Rating").show
/*
+--------------------+----------------+-----------+----------+
|           Seller ID|Customers Reached|Total Sales|Avg Rating|
+--------------------+----------------+-----------+----------+
|7c67e1448b00f6e96...|            1463|  1263053.1|      3.39|
|1f50f920176fa81da...|            2017|   705228.7|      3.98|
|1025f0e2d44d7041d...|            1477|   599293.06|     3.85|
|b37c4c02bda3161a7...|              15|   574920.8|       1.4|
|8e6d7754bc7e0f22c...|             135|   563415.6|      3.01|
|ce27a3cc3c8cc1ea7...|             173|   490234.34|     3.47|
|25c5c91f63607446a...|             271|   454122.22|     3.68|
|da8622b14eb17ae28...|            1662|   405415.4|      4.07|
|4a3ca9315b744ce9f...|            2155|   396226.62|     3.78|
|955fee9216a65b617...|            1530|   393402.56|     4.04|
|53243585a1d6dc264...|             437|   383077.7|      4.06|
|4869f7a5dfa277a7d...|            1186|   283238.2|       4.1|
|634964b17796e6430...|             336|   221408.97|     3.87|
|6560211a19b47992c...|            2130|   214515.6|      3.89|
|fa1c13f2614d7b5c4...|             609|   209730.97|     4.33|
|7e93a43ef30c4f03f...|             352|   187920.69|      4.2|
|a1043bafd471dff53...|             808|   177841.66|     4.14|
|7a67c85e85bb2ce85...|            1245|   176774.38|     4.21|
|620c87c171fb2a6dd...|             831|   169319.55|     4.19|
|c394e193cda3b4225...|              12|   166615.86|     4.67|
+--------------------+----------------+-----------+----------+
only showing top 20 rows
*/
```

07

Potential Customer's Location

```scala
val customerStateRdd = ePairRdd.map{case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal)=>
      (cState,(List(1),List(sp*qt)))}.reduceByKey{case(a,b)=>(a._1++b._1,a._2++b._2)}.map{case(cState,(qt,price))=>
      (cState,qt.length,price.sum,price.sum/qt.length)}.sortBy(_._3,false)
// customerStateRdd: org.apache.spark.rdd.RDD[(String, Int, Float, Float)] = MapPartitionsRDD[296] at sortBy at <console>:38
customerStateRdd.toDF("Customer State","Count","Total Amount spent","Avg Amount spent per order").show
/*
+--------------+-----+------------------+-------------------------+
|Customer State|Count|Total Amount spent|Avg Amount spent per order|
+--------------+-----+------------------+-------------------------+
|            SP|49131|       1.1303769E7|                230.07407|
|            RJ|15184|         4144816.0|                 272.9726|
|            MG|13529|         3173359.0|                234.55975|
|            PR| 5908|         1699214.4|                287.61246|
|            RS| 6435|         1663555.9|                258.51685|
|            GO| 2396|         1283653.6|                 535.7486|
|            BA| 3992|         1180009.0|                295.59344|
|            SC| 4254|         1139681.0|                267.90808|
|            DF| 2467|          563629.9|                228.46773|
|            ES| 2332|          542943.44|                232.82309|
|            CE| 1538|          460363.8|                299.32626|
|            PE| 1877|          444950.28|                237.05396|
|            MT| 1116|          391750.97|                351.03134|
|            PA| 1103|          322363.75|                292.26086|
|            MA|  838|          299734.25|                 357.6781|
|            PB|  630|          263676.5|                418.53412|
|            MS|  847|          211858.75|                250.12839|
|            PI|  567|          181782.84|                320.60468|
|            RN|  565|          136535.22|                241.65526|
|            AL|  458|          135510.22|                295.87384|
+--------------+-----+------------------+-------------------------+
only showing top 20 rows
*/
```

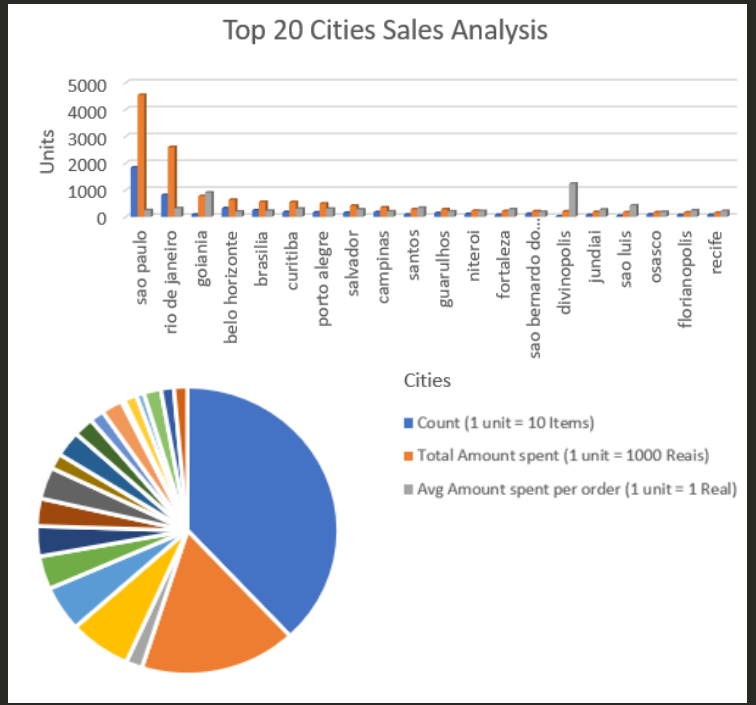


```scala
val customerCityRdd = ePairRdd.map{case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal)=>
      (cCity,(List(1),List(sp*qt)))}.reduceByKey{case(a,b)=>(a._1++b._1,a._2++b._2)}.map{case(cCity,(qt,price))=>
      (cCity,qt.length,price.sum,price.sum/qt.length)}.sortBy(_._3,false)
// customerCityRdd: org.apache.spark.rdd.RDD[(String, Int, Float, Float)] = MapPartitionsRDD[328] at sortBy at <console>:38
customerCityRdd.toDF("Customer City","Count","Total Amount spent","Avg Amount spent per order").show
/*
```

Customer City	Count	Total Amount spent	Avg Amount spent per order
sao paulo	18411	4542964.5	246.75273
rio de janeiro	8123	2599620.0	320.032
goiania	852	769202.06	902.81934
belo horizonte	3218	632311.9	196.49219
brasilia	2452	557939.7	227.54474
curitiba	1799	547542.25	304.35922
porto alegre	1640	494683.94	301.63654
salvador	1496	412581.22	275.78958
campinas	1722	350104.6	203.31277
santos	837	283820.94	339.0931
guarulhos	1388	280024.47	201.74673
niteroi	1047	226313.84	216.15457
fortaleza	753	211911.94	281.42355
sao bernardo do c...	1100	207171.22	188.33748
divinopolis	164	202474.16	1234.5985
jundiai	667	180321.73	270.34744
sao luis	400	170410.92	426.0273
osasco	894	166611.61	186.36646
florianopolis	672	163219.97	242.88686
recife	699	155056.12	221.82564

only showing top 20 rows

```scala
*/
```

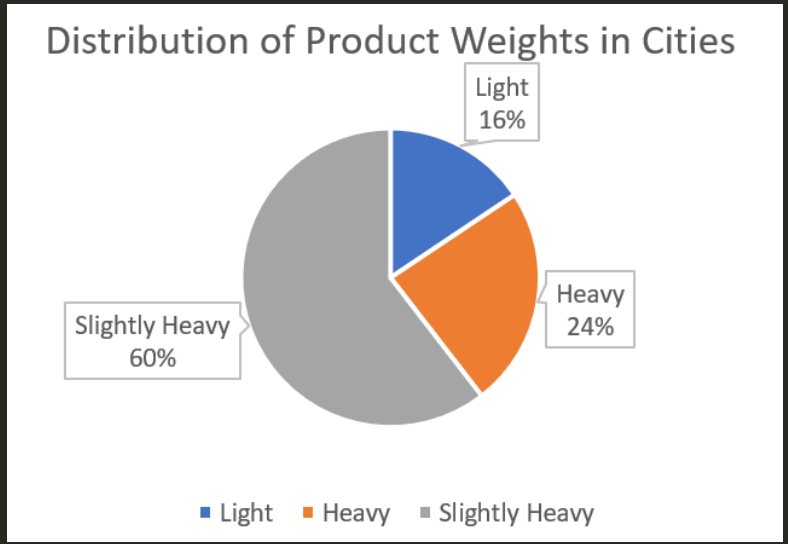


08

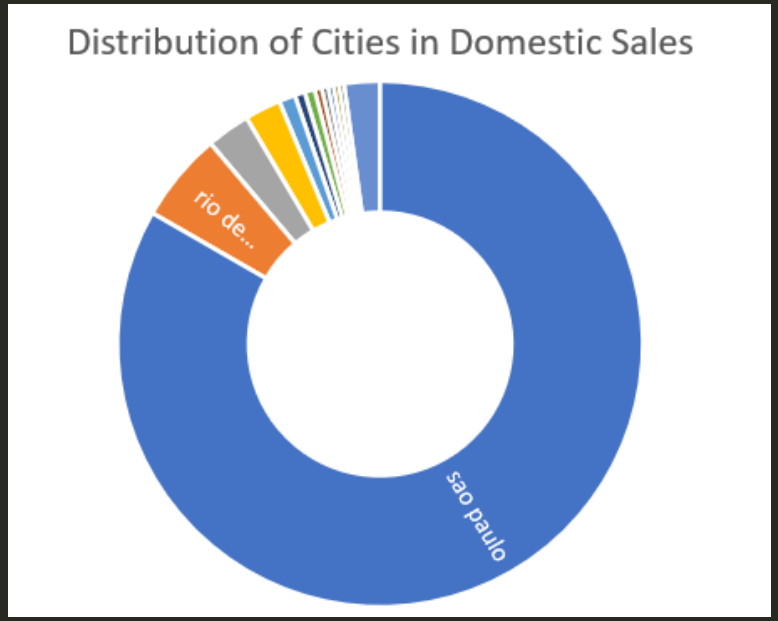
Logistics based Optimization Insights

```
// Which city buys more heavy wieght products?
val city_weight = ePairRdd.map{ case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal) =>
       var no_of_w = 0
       if ((pWt<3000)&&(pWt>=1000)) {no_of_w += 1}
       else if ((pWt<1000)&&(pWt>=200)) {no_of_w += 2}
       else if (pWt<200) {no_of_w += 3}
       else {no_of_w += 4}
       (cCity,List(no_of_w)) }.reduceByKey(_++_)
// city_weight: org.apache.spark.rdd.RDD[(String, List[Int])] = ShuffledRDD[333] at reduceByKey at <console>:36

val city_weight_2 = city_weight.map{ case(cCity,weight_list) =>
       val res = weight_list.map(x => (x,1)).reduce( (a,b) => ( a._1 + b._1,a._2 + b._2 ) )
       val avg_weight = (res._1/res._2).toFloat
       (cCity,avg_weight) }
// city_weight_2: org.apache.spark.rdd.RDD[(String, Float)] = MapPartitionsRDD[334] at map at <console>:32
city_weight_2.toDF("City","Weight Category").show
/*
+--------------------+---------------+
|                City|Weight Category|
+--------------------+---------------+
|sao francisco de ...|            1.0|
|        bandeirantes|            1.0|
|   nova guataporanga|            1.0|
|          guaranesia|            2.0|
|              cubati|            1.0|
|             pelotas|            2.0|
|               vigia|            1.0|
|santana do livram...|            2.0|
|         dois irmaos|            2.0|
|sao gabriel da palha|            2.0|
|               cuite|            3.0|
|       capao do leao|            2.0|
|           goncalves|            3.0|
|           carapebus|            2.0|
|       caxias do sul|            2.0|
|      estancia velha|            2.0|
|     atilio vivacqua|            1.0|
|             mucambo|            1.0|
|         ressaquinha|            4.0|
|              garuva|            3.0|
+--------------------+---------------+
only showing top 20 rows
*/
```

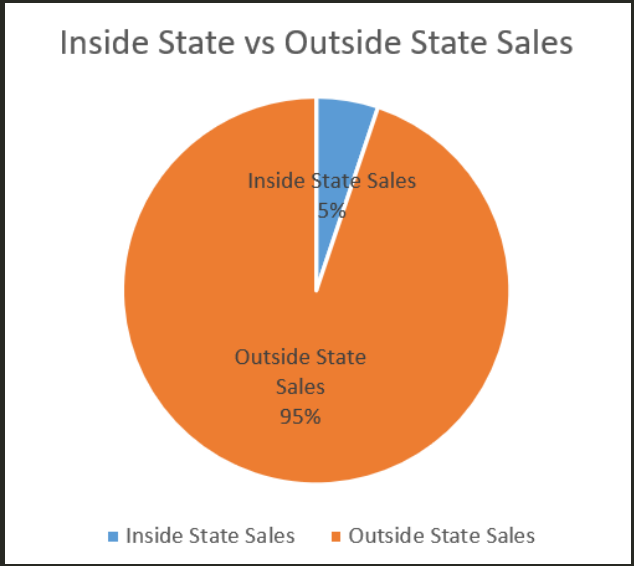
```
val city_weight_list = city_weight_2.map{ case(cCity,weight) =>
       var label = "None"
       if (weight==1) label="Heavy"
       else if (weight==2) label="Slightly Heavy"
       else if (weight==2) label="Medium"
       else label="Light"
       (label,List(cCity)) }.reduceByKey(_++_).map{ case(label,cCity_list) =>
       (label,cCity_list.size) }
// city_weight_list: org.apache.spark.rdd.RDD[(String, List[String], Int)] = MapPartitionsRDD[341] at map at <console>:37
city_weight_list.toDF("Weight Category","City Count").show
/*
+---------------+----------+
|Weight Category|City Count|
+---------------+----------+
|          Light|       640|
|          Heavy|       979|
| Slightly Heavy|      2476|
+---------------+----------+
*/
```



```
// How many proucts are domestic sales(Sold within seller city itself)?
val sales_loc = ePairRdd.map{ case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal) =>
       var count = 0
       if (cCity == sCity) count += 1
       (cCity,sCity,count) }.filter(x => (x._1 == x._2)).map{ x=> (x._1,x._3) }.reduceByKey(_+_)
// sales_loc: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[355] at reduceByKey at <console>:33
sales_loc.toDF("City","Count").show
/*
+--------------------+-----+
|                City|Count|
+--------------------+-----+
|           sao paulo| 4947|
|      rio de janeiro|  328|
|      belo horizonte|  159|
|            curitiba|  132|
|            brasilia|   60|
|            campinas|   37|
|       porto alegre|   37|
|          guarulhos|   26|
|         santo andre|   23|
|             maringa|   20|
|      ribeirao preto|   20|
|sao jose do rio p...|   19|
|            salvador|   13|
|     mogi das cruzes|   13|
|              natal|   12|
|           sorocaba|   11|
|         piracicaba|   11|
|           sao luis|   10|
|             santos|   10|
|           cascavel|    9|
+--------------------+-----+
only showing top 20 rows
*/
```



```scala
// Domestic vs Foreign sales:
val insideState_sales_count = sales_loc.map{ x => (x._2)}.sum.toInt
// insideState_sales_count: Int = 6028
val outsideState_sales_count = (ePairRdd.count - insideState_sales_count).toInt
// outsideState_sales_count: Int = 110545
val location_count = sc.parallelize( Array((insideState_sales_count,outsideState_sales_count)) )
//location_count: org.apache.spark.rdd.RDD[(Int, Int)] = ParallelCollectionRDD[385] at parallelize at <console>:36
location_count.toDF("Inside State Sales","Outside State Sales").show
/*
+-----------------+------------------+
|Inside State Sales|Outside State Sales|
+-----------------+------------------+
|             6028|            110545|
+-----------------+------------------+
*/
```



09

Predicting Future Sales

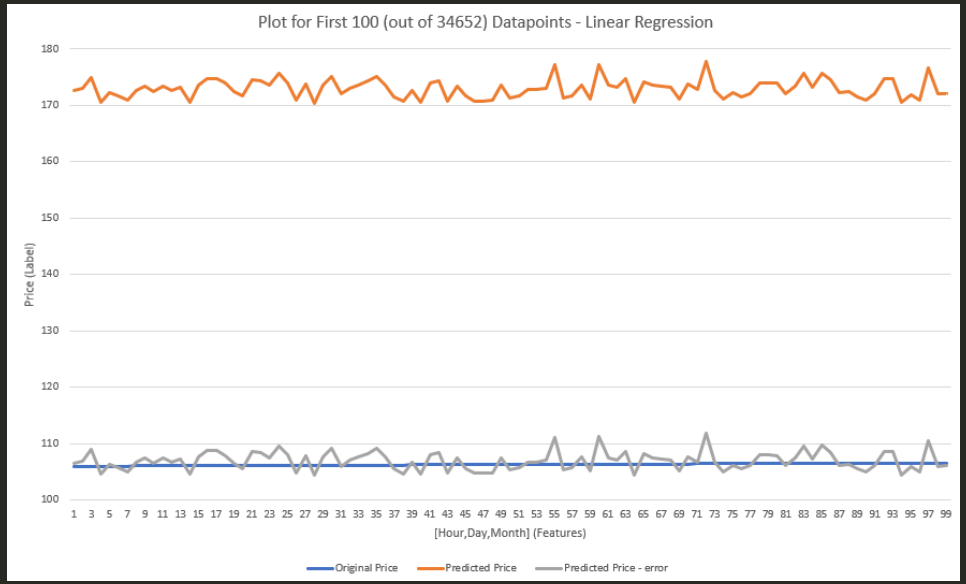
Machine Learning Model - Linear Regression

```scala
// Note: Predicting Sales Based on Hour, Day & Month
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.feature.Normalizer

val inputRdd = ePairRdd.map{ case(oid,cid,qt,cp,sp,ts,rating,pCat,pId,sp_type,
      oStat,pWt,pLen,pHt,pWidth,cCity,cState,sId,sCity,sState,sInstal) =>
      val ds_ts = ts.split(" ")
      val d_m_y = ds_ts(0).split("-")
      val hour = ds_ts(1).split(":")
      (sp.toFloat, hour(0).toInt,d_m_y(0).toInt,d_m_y(1).toInt) }.sortBy(_._1,false)
// inputRdd: org.apache.spark.rdd.RDD[(Float, Int, Int, Int)] = MapPartitionsRDD[394] at sortBy at <console>:37
inputRdd.toDF("Sales","Hour","Day","Month").show
/*
+--------+----+---+-----+
|   Sales|Hour|Day|Month|
+--------+----+---+-----+
|13664.08|  15| 29|    9|
|13664.08|  15| 29|    9|
|13664.08|  15| 29|    9|
|13664.08|  15| 29|    9|
|13664.08|  15| 29|    9|
|13664.08|  15| 29|    9|
|13664.08|  15| 29|    9|
|13664.08|  15| 29|    9|
| 7274.88|  14| 15|    7|
| 7274.88|  14| 15|    7|
| 7274.88|  14| 15|    7|
| 7274.88|  14| 15|    7|
| 6929.31|  20| 12|    2|
| 6922.21|  18| 25|    7|
| 6726.66|  18| 24|    5|
| 6081.54|  11| 24|   11|
| 6081.54|  11| 24|   11|
| 6081.54|  11| 24|   11|
| 6081.54|  11| 24|   11|
| 6081.54|  11| 24|   11|
+--------+----+---+-----+
*/
```

```scala
val df = inputRdd.toDF("label","Hour","Day","Month")
// df: org.apache.spark.sql.DataFrame = [label: float, Hour: int ... 2 more fields]

val assembler1 = new VectorAssembler().
  setInputCols(Array("Hour","Day","Month")).
  setOutputCol("features").
  transform(df)
// assembler1: org.apache.spark.sql.DataFrame = [label: float, Hour: int ... 3 more fields]
assembler1.show()
/*
+--------+----+---+-----+---------------+
|   label|Hour|Day|Month|       features|
+--------+----+---+-----+---------------+
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|
| 7274.88|  14| 15|    7| [14.0,15.0,7.0]|
| 7274.88|  14| 15|    7| [14.0,15.0,7.0]|
| 7274.88|  14| 15|    7| [14.0,15.0,7.0]|
| 7274.88|  14| 15|    7| [14.0,15.0,7.0]|
| 6929.31|  20| 12|    2| [20.0,12.0,2.0]|
| 6922.21|  18| 25|    7| [18.0,25.0,7.0]|
| 6726.66|  18| 24|    5| [18.0,24.0,5.0]|
| 6081.54|  11| 24|   11|[11.0,24.0,11.0]|
| 6081.54|  11| 24|   11|[11.0,24.0,11.0]|
| 6081.54|  11| 24|   11|[11.0,24.0,11.0]|
| 6081.54|  11| 24|   11|[11.0,24.0,11.0]|
| 6081.54|  11| 24|   11|[11.0,24.0,11.0]|
+--------+----+---+-----+---------------+
*/
```

```
val normalizer = new Normalizer().
  setInputCol("features").
  setOutputCol("normFeatures").
  setP(2.0).
  transform(assembler1)
// normalizer: org.apache.spark.sql.DataFrame = [label: float, Hour: int ... 4 more fields]
normalizer.show()
/*
+--------+----+---+-----+---------------+--------------------+
|   label|Hour|Day|Month|       features|        normFeatures|
+--------+----+---+-----+---------------+--------------------+
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|[0.44290394651811...|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|[0.44290394651811...|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|[0.44290394651811...|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|[0.44290394651811...|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|[0.44290394651811...|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|[0.44290394651811...|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|[0.44290394651811...|
|13664.08|  15| 29|    9| [15.0,29.0,9.0]|[0.44290394651811...|
| 7274.88|  14| 15|    7| [14.0,15.0,7.0]|[0.64577184562021...|
| 7274.88|  14| 15|    7| [14.0,15.0,7.0]|[0.64577184562021...|
| 7274.88|  14| 15|    7| [14.0,15.0,7.0]|[0.64577184562021...|
| 7274.88|  14| 15|    7| [14.0,15.0,7.0]|[0.64577184562021...|
| 6929.31|  20| 12|    2| [20.0,12.0,2.0]|[0.85435765771676...|
| 6922.21|  18| 25|    7| [18.0,25.0,7.0]|[0.56978004404962...|
| 6726.66|  18| 24|    5| [18.0,24.0,5.0]|[0.59183635429928...|
| 6081.54|  11| 24|   11|[11.0,24.0,11.0]|[0.38460598070128...|
| 6081.54|  11| 24|   11|[11.0,24.0,11.0]|[0.38460598070128...|
| 6081.54|  11| 24|   11|[11.0,24.0,11.0]|[0.38460598070128...|
| 6081.54|  11| 24|   11|[11.0,24.0,11.0]|[0.38460598070128...|
| 6081.54|  11| 24|   11|[11.0,24.0,11.0]|[0.38460598070128...|
+--------+----+---+-----+---------------+--------------------+
*/
```

```scala
val Array(trainingData, testData) = normalizer.randomSplit(Array(0.70, 0.30))
// trainingData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [label: float, Hour: int ... 4 more fields]
// testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [label: float, Hour: int ... 4 more fields]

trainingData.count
// Long = 81922
testData.count
// Long = 34651

val lr = new LinearRegression().
  setLabelCol("label").
  setFeaturesCol("normFeatures").
  setMaxIter(10).
  setRegParam(1.0).
  setElasticNetParam(1.0)
// lr: org.apache.spark.ml.regression.LinearRegression = linReg_f72585771c7b

val lrModel = lr.fit(trainingData)
// lrModel: org.apache.spark.ml.regression.LinearRegressionModel = linReg_f72585771c7b

lrModel.
  transform(testData).
  select("features","label","prediction").
  show()
/*
+---------------+-----+------------------+
|       features|label|        prediction|
+---------------+-----+------------------+
|  [20.0,5.0,5.0]|105.95|172.54876571676618|
|  [20.0,6.0,6.0]|105.95|172.97477934217812|
| [10.0,14.0,9.0]|105.96|174.94695419232792|
|  [21.0,3.0,1.0]|105.99|170.56329677228612|
| [11.0,31.0,7.0]|106.02| 172.2586847169489|
| [15.0,29.0,5.0]|106.02| 171.661097436566|
| [21.0,13.0,2.0]|106.02| 170.9172572137698|
|  [10.0,6.0,3.0]|106.03|172.69018221534373|
|  [8.0,16.0,6.0]|106.06| 173.4150760503963|
|  [11.0,7.0,3.0]|106.07|172.42800714769788|
|[14.0,30.0,11.0]|106.07|173.38683990123567|
| [21.0,12.0,6.0]|106.07|172.60212984586786|
|[23.0,26.0,11.0]|106.07|173.24749127118145|
|  [23.0,3.0,1.0]|106.08| 170.5210312172166|
|[11.0,29.0,11.0]| 106.1|173.58619736348862|
|[20.0,14.0,12.0]| 106.1| 174.7112125477511|
|[20.0,14.0,12.0]| 106.1| 174.7112125477511|
|[16.0,23.0,11.0]|106.11| 173.9145763553082|
| [16.0,26.0,7.0]|106.11| 172.4202413660314|
| [22.0,15.0,4.0]|106.15|171.63141300037034|
+---------------+-----+------------------+
*/
```



```
println(s"Coefficients: ${lrModel.coefficients} Intercept: ${lrModel.intercept}")
//Coefficients: [0.0,0.0,10.526619961906] Intercept: 170.0676175974204

val trainingSummary = lrModel.summary
println(s"numIterations: ${trainingSummary.totalIterations}")
//numIterations: 7
println(s"objectiveHistory: [${trainingSummary.objectiveHistory.mkString(",")}]")
//objectiveHistory: [0.5,0.4999857123444021,0.49998078700776555,0.4999807204943856,0.49998071959616674,0.49998071958403695,0.4999807195838732]
println(s"RMSE: ${trainingSummary.rootMeanSquaredError}")
// RMSE: 274.30629592771396
println(s"r2: ${trainingSummary.r2}")
// r2: 8.383535671985243E-5
```

THANKS !

For Viewing this GitHub Repo