Comparative Analysis of Query Performance in Oracle DB and Hadoop

By:
Anaja Bajpeyi
Harshada Khandekar
Ketki Kokate
Ramya Umamaheswaran
Sairamya Kothapalli
Swetha Rajaraman

Project Usecase

The proposed system focuses on a Sales Transaction Management System to keep track of the orders placed by customers from different states. The system will allow the sales admin to add products into the system, retrieve the orders of the customers. The customer can also review the product. This will help sales admin draw insights from these transactions.

Implementation Details

- **Hive/Hadoop Section**
  We ran the queries on the hadoop cluster where data is stored using Hive QL on the design center account provided. Using this we noted the time taken for the queries to execute on hive.

- **Oracle DB and Explain plan Tool**
  Execution of all the queries done using oracle db11g on the design center account provided to see the performance difference. Oracle explain plan tool is used to have detailed description of how it plans on executing the query. This is useful in tuning queries to the database to get them to perform better. Knowing query execution plans your query helps to change the environment to run the query faster.

Cluster Information(Overall Cluster)

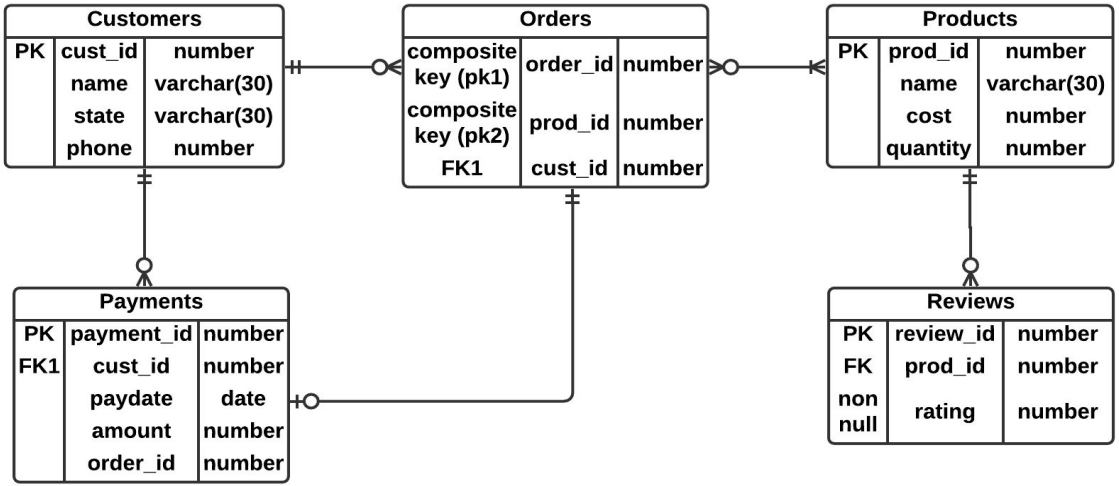
As mentioned in the Link:

- Environment: Cloudera CDH-5.16 - YARN (MapReduce v2) and Spark (2.4)
- Worker Nodes: 24
- Cores: 96
- Threads: 192
- RAM: 768GB
- HDFS Storage (Raw): 261TB
- HDFS Storage (Usable): 80TB (After factoring replication overhead)

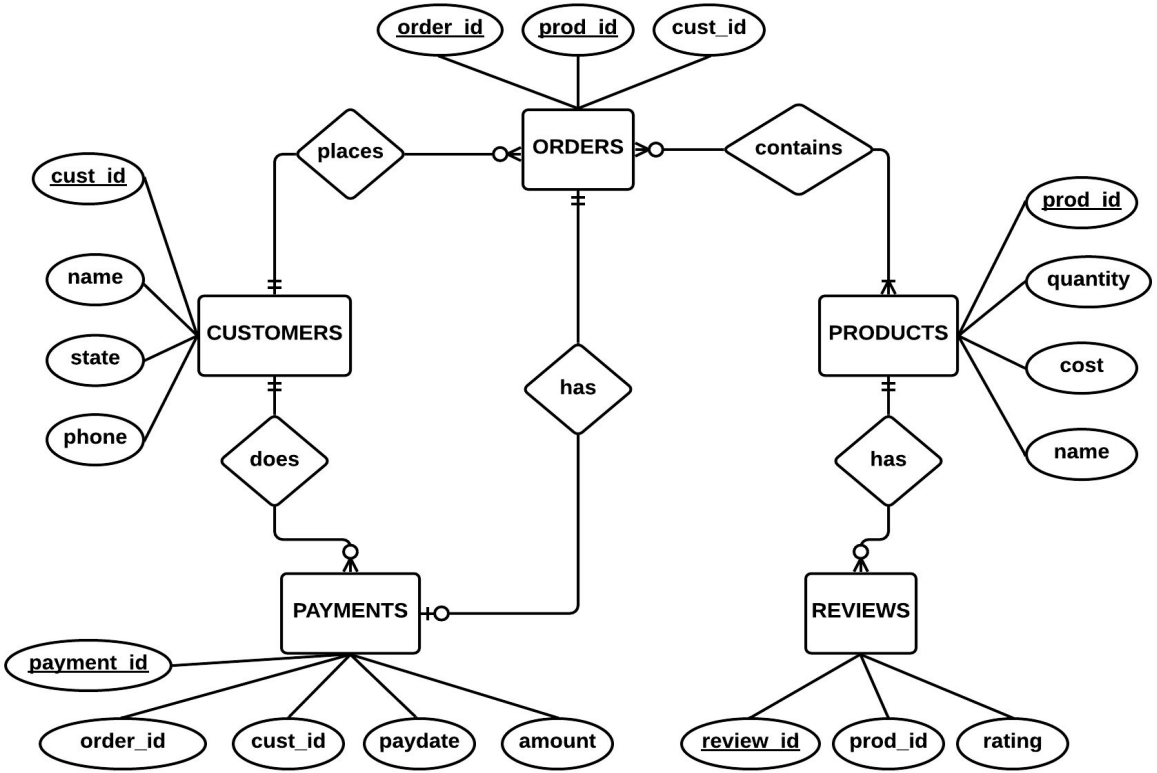
Dataset

- Our Dataset for each table is 50 records.

Schema Diagram



ER Diagram


order_id
prod_id
cust_id
ORDERS
places
contains
cust_id
name
state
phone
CUSTOMERS
does
has
PRODUCTS
prod_id
quantity
cost
name
has
PAYMENTS
REVIEWS
payment_id
order_id
cust_id
paydate
amount
review_id
prod_id
rating

Oracle Explain Plan

The EXPLAIN PLAN statement displays execution plans chosen by the Oracle optimizer for SELECT, UPDATE, INSERT, and DELETE statements. The EXPLAIN PLAN helps you to understand the optimizer decisions, such as why the optimizer chose a nested loops join instead of a hash join, and lets you understand the performance of a query.

The optimizer performs the following steps:

- The optimizer generates a set of potential plans for the SQL statement based on available access paths and hints.
- The optimizer estimates the cost of each plan based on statistics in the data dictionary. Statistics include information on the data distribution and storage characteristics of the tables, indexes, and partitions accessed by the statement.
- The optimizer compares the plans and chooses the plan with the lowest cost.

Queries

1. Aggregate- COUNT

The below query gives the number of customers of each state.

Oracle Query:

```
SELECT state, count(*) as Num_Customers from Customers group by state;
```

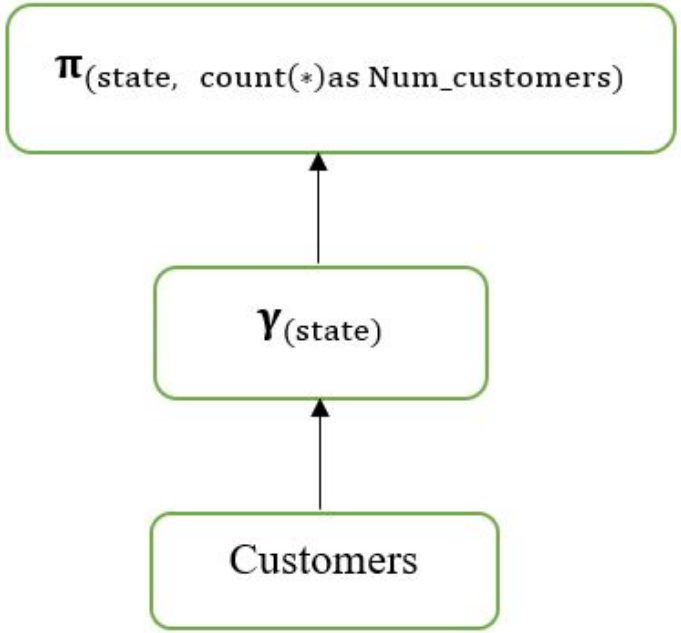
Hive Query:

```
SELECT state, count(*) as Num_Customers from Customers group by state;
```

Oracle Explain Plan

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
Plan hash value: 1577413243

------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |    50 |   850 |     4  (25)| 00:00:01 |
|   1 |  HASH GROUP BY     |           |    50 |   850 |     4  (25)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| CUSTOMERS |    50 |   850 |     3   (0)| 00:00:01 |
------------------------------------------------------------------------------

Note

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
-----
   - dynamic sampling used for this statement (level=2)

13 rows selected.
```

Proposed Query Plan



Oracle Output

```
SQL> SELECT state, count(*) as Num_Customers from Customers group by state;

STATE                          NUM_CUSTOMERS
------------------------------ -------------
ARIZONA                                    2
IOWA                                       1
MICHIGAN                                   2
OHIO                                       5
COLORADO                                   3
NEW JERSEY                                 1
FLORIDA                                    2
VIRGINIA                                   2
ALABAMA                                    1
NORTH CAROLINA                             2
PENNSYLVANIA                               2

STATE                          NUM_CUSTOMERS
------------------------------ -------------
HAWAII                                     2
TEXAS                                      2
GEORGIA                                    3
NEW YORK                                   2
MAINE                                      1
NEVADA                                     1
NEW MEXICO                                 1
CALIFORNIA                                 6
ALASKA                                     2
OREGON                                     2
INDIANA                                    3

STATE                          NUM_CUSTOMERS
------------------------------ -------------
WISCONSIN                                  1
DELAWARE                                   1

24 rows selected.
```

Time to execute on Oracle: 00:00:09.81 seconds

Hive Output

```
bigdata01@linux11114:~
Starting Job = job_1605809401632_0092, Tracking URL = http://name1.hadoop.dc.engr.scu.edu:8088/proxy/application_1605809401632_0092/
Kill Command = /DCNFS/applications/cdh/5.16/app/CDH-5.16.2-1.cdh5.16.2.p0.8/lib/hadoop/bin/hadoop job  -kill job_1605809401632_0092
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-11-30 14:06:48,615 Stage-1 map = 0%,  reduce = 0%
2020-11-30 14:06:52,872 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.34 sec
2020-11-30 14:06:59,242 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.84 sec
MapReduce Total cumulative CPU time: 2 seconds 840 msec
Ended Job = job_1605809401632_0092
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.84 sec   HDFS Read: 9607 HDFS Write: 254 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 840 msec
OK
ALABAMA 1
ALASKA  2
ARIZONA 2
CALIFORNIA      6
COLORADO        3
DELAWARE        1
FLORIDA 2
GEORGIA 3
HAWAII  2
INDIANA 3
IOWA    1
MAINE   1
MICHIGAN        2
NEVADA  1
NEW JERSEY      1
NEW MEXICO      1
NEW YORK        2
NORTH CAROLINA  2
OHIO    5
OREGON  2
PENNSYLVANIA    2
TEXAS   2
VIRGINIA        2
WISCONSIN       1
Time taken: 15.995 seconds, Fetched: 24 row(s)
hive>
```

Time to execute on Hive: 15.995 seconds

2. Aggregate- MAX

The below Query gives the highest number of product sold.

Oracle Query:

```
SELECT MAX(SALES) FROM (SELECT prod_id, COUNT(*) SALES FROM ORDERS GROUP BY
prod_id) t;
```

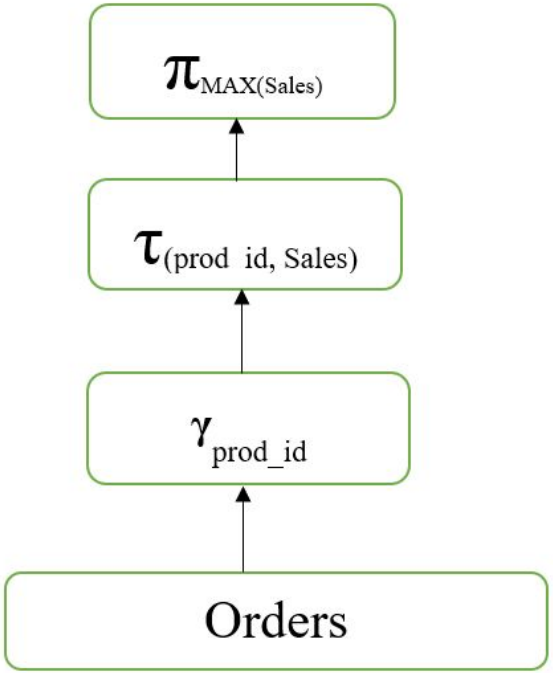
Hive Query:

```
SELECT MAX(SALES) FROM (SELECT prod_id, COUNT(*) SALES FROM ORDERS GROUP BY
prod_id) t;
```

Oracle Explain Plan

```
abajpeyi@linux11115:~

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------
Plan hash value: 2037207053

-------------------------------------------------------------------------------------
| Id  | Operation          | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |               |     1 |    13 |     2  (50)| 00:00:01 |
|   1 |  SORT AGGREGATE    |               |     1 |    13 |            |          |
|   2 |   VIEW             |               |    11 |   143 |     2  (50)| 00:00:01 |
|   3 |    HASH GROUP BY   |               |    11 |    33 |     2  (50)| 00:00:01 |
|   4 |     INDEX FULL SCAN| SYS_C00868250 |    15 |    45 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

11 rows selected.

SQL>
```

Proposed Query Plan



Oracle Output

```
SQL> SELECT MAX(SALES) FROM (SELECT prod_id, COUNT(*) SALES FROM ORDERS GROUP BY prod_id) t;

MAX(SALES)
----------
         5

SQL>
```

Time to execute on Oracle: 00:00:14.02 seconds

```
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_1605809401632_0052, Tracking URL = http://name1.hadoop.dc.engr.scu.edu:8088/proxy/application_1605809401632_0052/
Kill Command = /DCNFS/applications/cdh/5.16/app/CDH-5.16.2-1.cdh5.16.2.p0.8/lib/hadoop/bin/hadoop job  -kill job_1605809401632_0052
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-11-29 13:00:53,415 Stage-2 map = 0%,  reduce = 0%
2020-11-29 13:00:58,718 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.08 sec
2020-11-29 13:01:01,902 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.81 sec
MapReduce Total cumulative CPU time: 2 seconds 810 msec
Ended Job = job_1605809401632_0052
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.04 sec   HDFS Read: 7617 HDFS Write: 114 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.81 sec   HDFS Read: 4644 HDFS Write: 2 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 850 msec
OK
5
Time taken: 31.526 seconds, Fetched: 1 row(s)
```

Time to execute on Hive: 31.526 seconds

3. Aggregate- MIN

The below Query gives the least number of product sold.

Oracle Query:

```
SELECT MIN(SALES) FROM (SELECT prod_id, COUNT(*) SALES FROM ORDERS GROUP BY
prod_id) t;
```

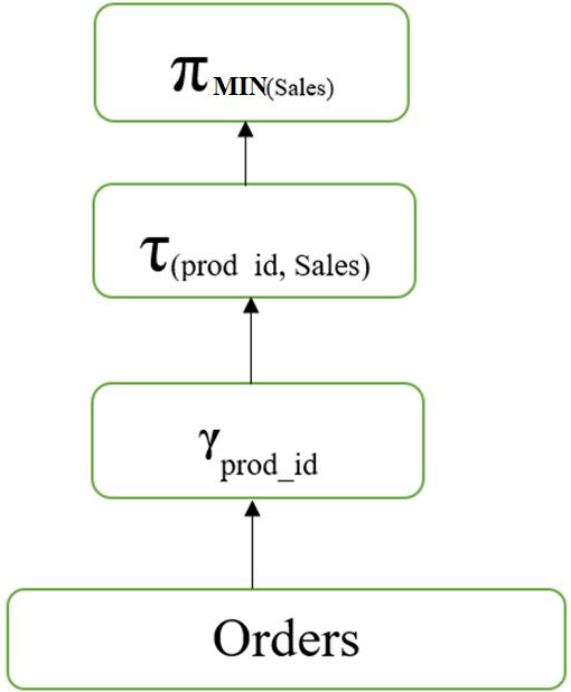
Hive Query:

```
SELECT MIN(SALES) FROM (SELECT prod_id, COUNT(*) SALES FROM ORDERS GROUP BY
prod_id) t;
```

Oracle Explain Plan

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
Plan hash value: 1143457762

------------------------------------------------------------------------------
---------

| Id  | Operation              | Name          | Rows  | Bytes | Cost (%CPU)| T
ime     |

------------------------------------------------------------------------------
---------


PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |               |     1 |    13 |     3  (34)| 0
0:00:01 |

|   1 |  SORT AGGREGATE        |               |     1 |    13 |            |
        |

|   2 |   VIEW                 |               |    50 |   650 |     3  (34)| 0
0:00:01 |

|   3 |    HASH GROUP BY       |               |    50 |   650 |     3  (34)| 0
0:00:01 |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------

|   4 |     INDEX FAST FULL SCAN| SYS_C00868561 |    50 |   650 |     2   (0)| 0
0:00:01 |

------------------------------------------------------------------------------
---------


Note
-----
   - dynamic sampling used for this statement (level=2)

15 rows selected.
```

Proposed Query Plan



Oracle Output

```
SQL> SELECT MIN(SALES) FROM (SELECT prod_id, COUNT(*) SALES FROM ORDERS GROUP BY prod_id) t;

MIN(SALES)
----------
         1
```

Time to execute on Oracle: 00:00:10.15 seconds

Hive Output

bigdata01@linux11114:~

```
Launching Job 2 out of 2
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_1605809401632_0091, Tracking URL = http://name1.hadoop.dc.engr.scu.edu:8088/proxy/application_1605809401632_0091/
Kill Command = /DCNFS/applications/cdh/5.16/app/CDH-5.16.2-1.cdh5.16.2.p0.8/lib/hadoop/bin/hadoop job  -kill job_1605809401632_0091
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-11-30 14:04:48,071 Stage-2 map = 0%,  reduce = 0%
2020-11-30 14:04:52,309 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.1 sec
2020-11-30 14:04:57,606 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.82 sec
MapReduce Total cumulative CPU time: 2 seconds 820 msec
Ended Job = job_1605809401632_0091
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.81 sec   HDFS Read: 7618 HDFS Write: 114 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.82 sec   HDFS Read: 4653 HDFS Write: 2 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 630 msec
OK
1
Time taken: 32.947 seconds, Fetched: 1 row(s)
hive>
```

Time to execute on Hive: 32.947 seconds

4. LIMIT

The below query gives the first 3 payment entries:

Oracle query:

```
SELECT  * from payments Where ROWNUM <= 3;
```

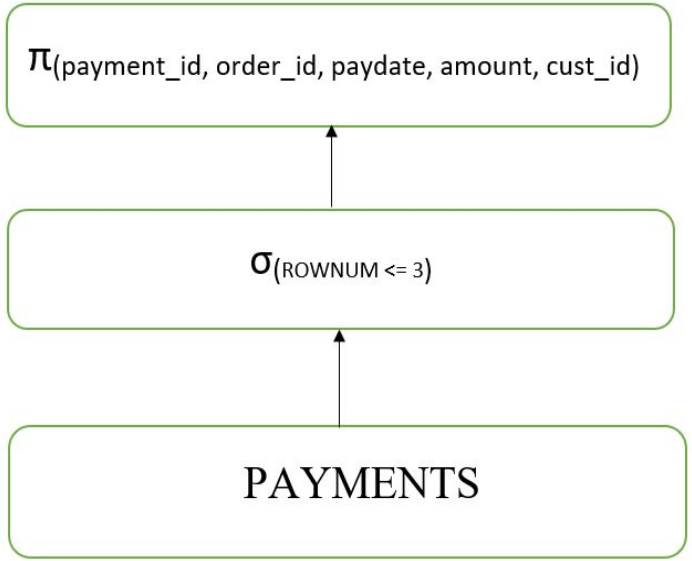
Hive query:

```
SELECT  * from payments limit 3;
```

Oracle Explain Plan

```
Plan hash value: 1799139040

-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |     3 |    63 |     2   (0)| 00:00:01 |
|*  1 |  COUNT STOPKEY     |          |       |       |            |          |
|   2 |   TABLE ACCESS FULL| PAYMENTS |     3 |    63 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(ROWNUM<=3)

14 rows selected.

SQL>
```

Proposed Query Plan



Oracle Output

```
SQL> SELECT  * from payments Where ROWNUM <= 3;

PAYMENT_ID   ORDER_ID PAYDATE       AMOUNT    CUST_ID
---------- ---------- --------- ---------- ----------
       101          1 12-OCT-20         10          3
       102          2 25-JAN-05         30          5
       103          3 31-DEC-20          5          6
```

Time taken to execute on Oracle: 00:00:09.91 seconds

Hive Output

```
hive> select * from payments limit 3;
OK
101     1       12-10-2020      10.0    3
102     2       25-01-05        30.0    5
103     3       31-12-20        5.0     6
Time taken: 0.774 seconds, Fetched: 3 row(s)
hive>
```

Time to execute on Hive: 0.774seconds

5. GROUP BY

The below Query gives the sales of each product.

Oracle Query:

```
SELECT prod_id, COUNT(*) AS SALES FROM Orders GROUP BY prod_id;
```

Hive Query:

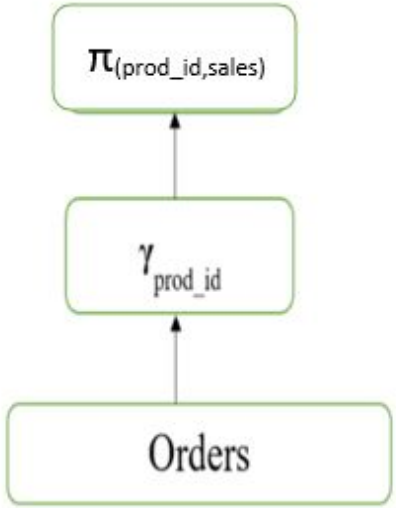
```
SELECT prod_id, COUNT(*) AS SALES FROM Orders GROUP BY prod_id;
```

Oracle Explain Plan

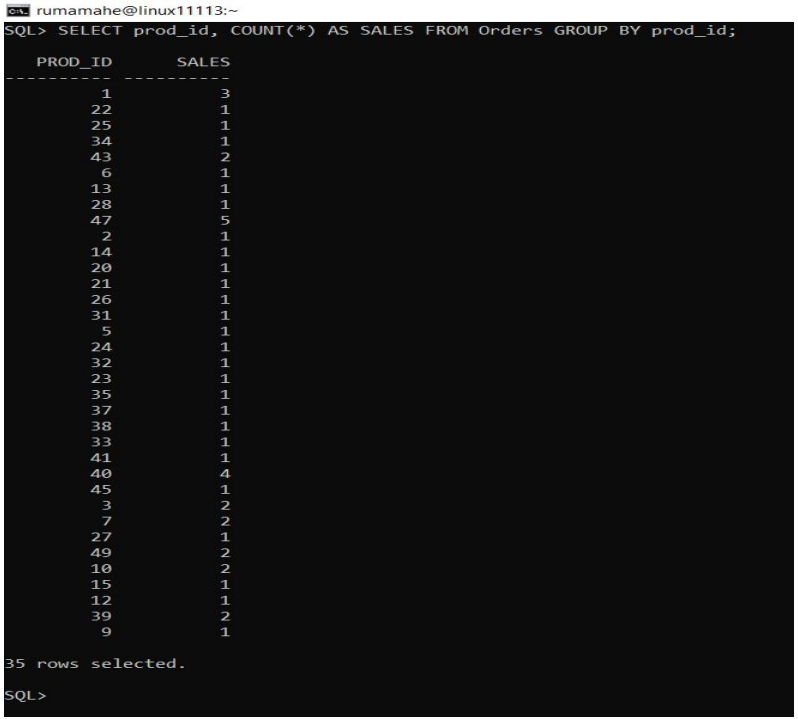
rumamahe@linux11113:~

```
SQL> select plan_table_output from table(dbms_xplan.display('plan_table',null,'typical'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Plan hash value: 3139884282

-------------------------------------------------------------------------------
| Id  | Operation         | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |               |    35 |   105 |     2  (50)| 00:00:01 |
|   1 |  HASH GROUP BY    |               |    35 |   105 |     2  (50)| 00:00:01 |
|   2 |   INDEX FULL SCAN | SYS_C00868471 |    50 |   150 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------

9 rows selected.

SQL>
```

Proposed Query Plan

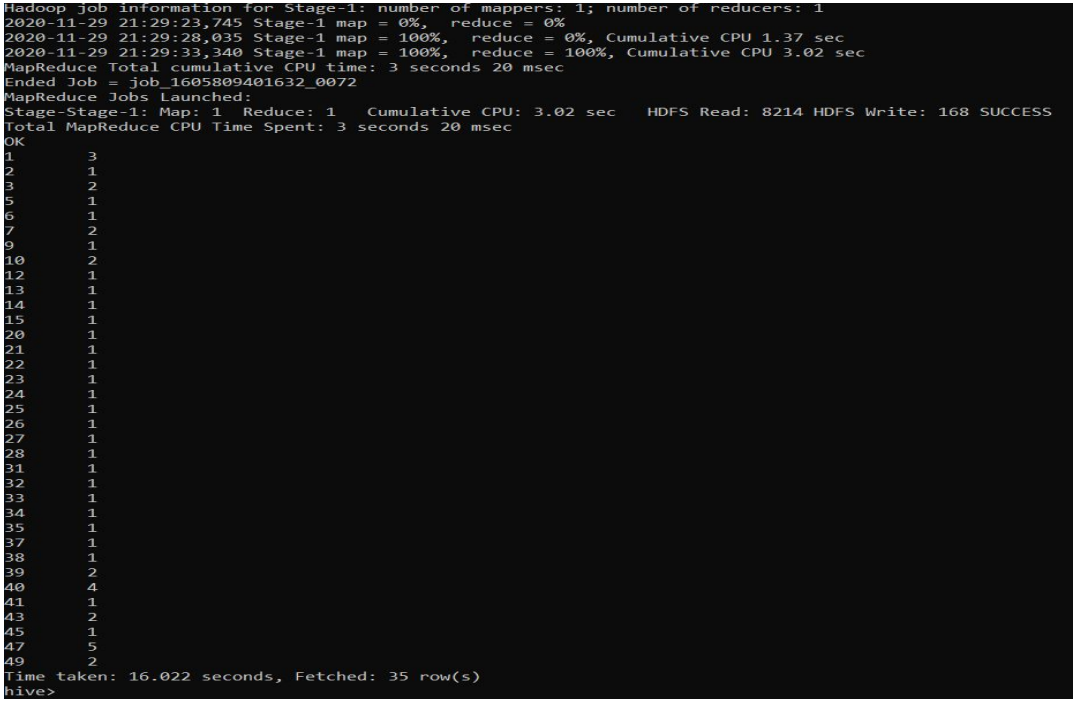


Oracle Output



Time to execute in oracle: 00:00:06.93

Hive Output



Time to execute on Hive:
16.022 seconds

6. NESTED QUERIES

This query gives us the top 3 best rated products.

Oracle Query:

```
SELECT * from (SELECT prod_id, ROUND(AVG(Rating))  rating FROM
Reviews group by prod_id order by rating desc) t Where ROWNUM <= 3;
```

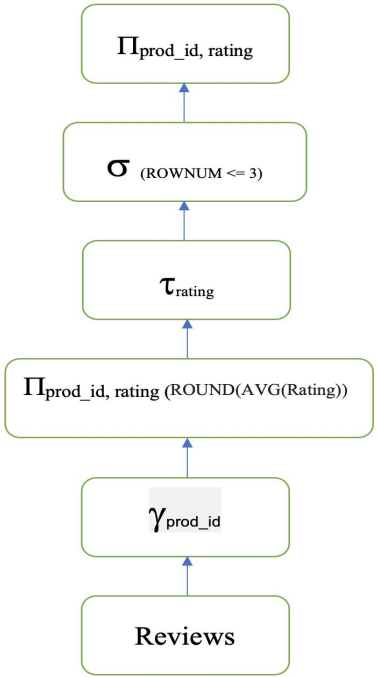
Hive Query:

```
SELECT * from (SELECT prod_id, ROUND(AVG(Rating))  rating FROM
reviews group by prod_id) t order by t.rating desc LIMIT 3;
```

Oracle Explain Plan

```
SQL> select plan_table_output from table(dbms_xplan.display('plan_table',null,'typical'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1915442491

--------------------------------------------------------------------------------
---
[
[| Id  | Operation               | Name    | Rows  | Bytes | Cost (%CPU)| Time
   |

--------------------------------------------------------------------------------
---
[

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |         |     3 |    78 |     5  (40)| 00:00:0
1 |

|*  1 |  COUNT STOPKEY          |         |       |       |            |
   |

|   2 |   VIEW                  |         |    25 |   650 |     5  (40)| 00:00:0
1 |

|*  3 |    SORT ORDER BY STOPKEY|         |    25 |   150 |     5  (40)| 00:00:0
1 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|   4 |     HASH GROUP BY       |         |    25 |   150 |     5  (40)| 00:00:0
1 |

|   5 |      TABLE ACCESS FULL  | REVIEWS |    50 |   300 |     3   (0)| 00:00:0
1 |

--------------------------------------------------------------------------------
---


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(ROWNUM<=3)
   3 - filter(ROWNUM<=3)

18 rows selected.
```

Proposed Query Plan



Oracle Output

```
SQL> SELECT * from (SELECT prod_id, ROUND(AVG(Rating))  rating FROM Reviews grou
p by prod_id order by rating desc) t Where ROWNUM <= 3;

   PROD_ID     RATING
---------- ----------
         1          5
         5          5
        22          5
```

Time to execute on Oracle: 00:00:16.25 seconds

Hive Output

```
  set mapreduce.job.reduces=<number>
Starting Job = job_1605809401632_0056, Tracking URL = http://name1.hadoop.dc.engr.scu.edu:8088/proxy/application_1605809401632_0056/
Kill Command = /DCNFS/applications/cdh/5.16/app/CDH-5.16.2-1.cdh5.16.2.p0.8/lib/hadoop/bin/hadoop job  -kill job_1605809401632_0056
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-11-29 15:49:43,688 Stage-2 map = 0%,  reduce = 0%
2020-11-29 15:49:48,986 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.03 sec
2020-11-29 15:49:53,223 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.83 sec
MapReduce Total cumulative CPU time: 2 seconds 830 msec
Ended Job = job_1605809401632_0056
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.88 sec   HDFS Read: 8025 HDFS Write: 746 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.83 sec   HDFS Read: 5683 HDFS Write: 19 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 710 msec
OK
22      5.0
5       5.0
1       5.0
Time taken: 33.845 seconds, Fetched: 3 row(s)
```

Time to execute on Hive: 33.845 seconds

7. ORDER BY

This query gives us sales of each product ordered by decreasing order of sales.

Oracle Query:

```
SELECT prod_id, Count(*)  sales FROM orders group by prod_id order
by sales desc;
```

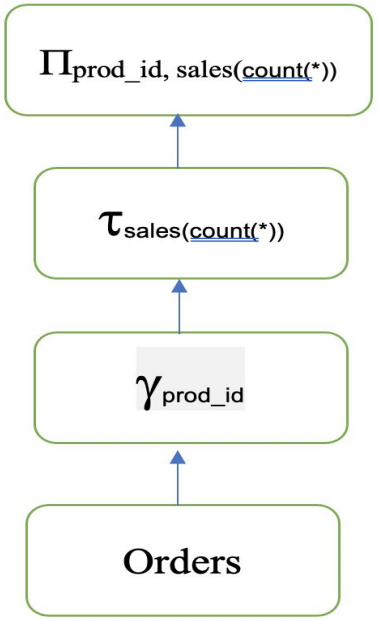
Hive Query:

```
SELECT prod_id, Count(*)  sales FROM orders group by prod_id order
by sales desc;
```

Oracle Explain Plan

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1245999929

--------------------------------------------------------------------------------
---
| Id  | Operation         | Name          | Rows  | Bytes | Cost (%CPU)| Time
   |

--------------------------------------------------------------------------------
---


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |               |    35 |   105 |     3  (67)| 00:00:0
1 |
[
|   1 |  SORT ORDER BY    |               |    35 |   105 |     3  (67)| 00:00:0
1 |

|   2 |   HASH GROUP BY   |               |    35 |   105 |     3  (67)| 00:00:0
[1 |

|   3 |    INDEX FULL SCAN| SYS_C00868471 |    50 |   150 |     1   (0)| 00:00:0
1 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
---


10 rows selected.
```

Proposed Query Plan



Oracle Output

```
SQL> SELECT prod_id, Count(*)  sales FROM orders group by prod_id order by sales desc;

   PROD_ID       SALES
---------- ----------
        47          5
        40          4
         1          3
         7          2
         3          2
        43          2
        10          2
        39          2
        49          2
        21          1
        26          1

   PROD_ID       SALES
---------- ----------
        31          1
         5          1
        24          1
        32          1
        23          1
        35          1
        37          1
        38          1
        33          1
        41          1
        45          1

   PROD_ID       SALES
---------- ----------
        27          1
        15          1
        12          1
        20          1
        14          1
         2          1
        28          1
        13          1
         6          1
        34          1
        25          1

   PROD_ID       SALES
---------- ----------
         9          1
        22          1

35 rows selected.
```

Time to execute on Oracle: 00:00:12.71 seconds

Hive Output

```
Kill Command = /DCNFS/applications/cdh/5.16/app/CDH-5.16.2-1.cdh5.16.2.p0.8/lib/hadoop/bin/ha
doop job  -kill job_1605809401632_0050
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-11-29 12:49:26,075 Stage-2 map = 0%,  reduce = 0%
2020-11-29 12:49:31,376 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.16 sec
2020-11-29 12:49:36,674 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.95 sec
MapReduce Total cumulative CPU time: 2 seconds 950 msec
Ended Job = job_1605809401632_0050
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.77 sec   HDFS Read: 7268 HDFS Write: 761
 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.95 sec   HDFS Read: 5696 HDFS Write: 168
 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 720 msec
OK
```

```
Time taken: 35.28 seconds, Fetched: 35 row(s)
hive>
```

Time to execute on Hive: 35.28 seconds

8. COMPLEX JOIN- Three way join

This query gives us all the products that have been sold in every state.

Oracle Query:

```
SELECT p.prod_id,  p.name, c.state from products p
Inner join orders ON orders.prod_id = p.prod_id
Inner join customers c on c.cust_id = orders.cust_id
Order by p.prod_id;
```

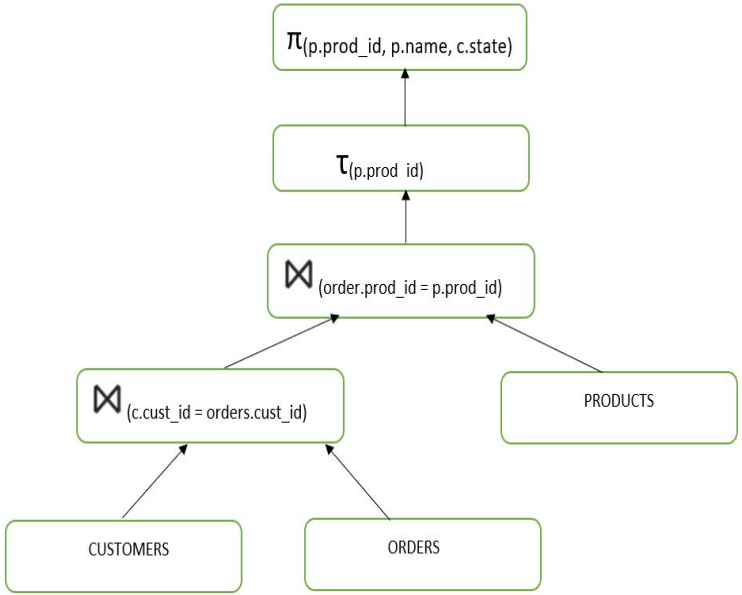
Hive Query:

```
SELECT p.prod_id,  p.name, c.state from products p
Inner join orders ON orders.prod_id = p.prod_id
Inner join customers c on c.cust_id = orders.cust_id
Order by p.prod_id;
```

Oracle Explain Plan

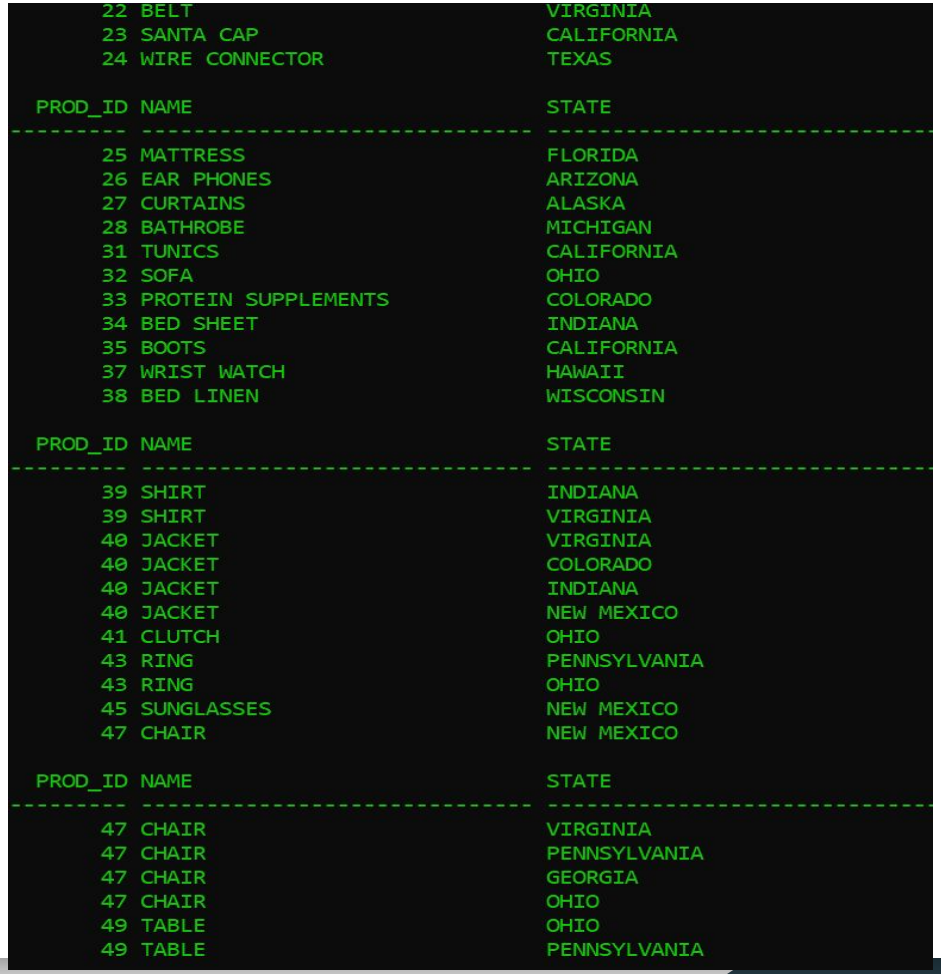
```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
Plan hash value: 2154960647

-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |    50 |  1550 |    11  (28)| 00:00:01 |
|   1 |  SORT ORDER BY                |               |    50 |  1550 |    11  (28)| 00:00:01 |
|*  2 |   HASH JOIN                   |               |    50 |  1550 |    10  (20)| 00:00:01 |
|   3 |    MERGE JOIN                 |               |    50 |   950 |     6  (17)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID| PRODUCTS      |    50 |   650 |     2   (0)| 00:00:01 |
|   5 |      INDEX FULL SCAN          | SYS_C00868470 |    50 |       |     1   (0)| 00:00:01 |
|*  6 |     SORT JOIN                 |               |    50 |   300 |     4  (25)| 00:00:01 |
|   7 |      TABLE ACCESS FULL        | ORDERS        |    50 |   300 |     3   (0)| 00:00:01 |
|   8 |    TABLE ACCESS FULL          | CUSTOMERS     |    50 |   600 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("C"."CUST_ID"="ORDERS"."CUST_ID")
   6 - access("ORDERS"."PROD_ID"="P"."PROD_ID")
       filter("ORDERS"."PROD_ID"="P"."PROD_ID")

22 rows selected.

SQL>
```

Proposed Query Plan



Oracle Output

```
SQL> SELECT p.prod_id,  p.name, c.state from products p
Inner join orders ON orders.prod_  2  id = p.prod_id
Inner join  3   customers c on c.cust_id = orders.cust_id
Order by  4   p.prod_id;

   PROD_ID NAME                           STATE
---------- ------------------------------ ------------------------------
         1 HOT WHEELS                     OHIO
         1 HOT WHEELS                     FLORIDA
         1 HOT WHEELS                     NORTH CAROLINA
         2 CLAY                           CALIFORNIA
         3 PAINT BRUSH                    FLORIDA
         3 PAINT BRUSH                    CALIFORNIA
         5 WATER COLORS                   CALIFORNIA
         6 CRAYONS                        CALIFORNIA
         7 BEYBLADE                       COLORADO
         7 BEYBLADE                       ALASKA
         9 JINGLE BELLS                   NEW MEXICO

   PROD_ID NAME                           STATE
---------- ------------------------------ ------------------------------
        10 ELECTRIC LIGHTS                ALASKA
        10 ELECTRIC LIGHTS                NEW MEXICO
        12 FOOTBALL                       TEXAS
        13 LANTERNS                       OHIO
        14 CANDLES                        ARIZONA
        15 SKATEBOARD                     MICHIGAN
        20 LEGGINS                        NORTH CAROLINA
        21 PLANT POT                      NEW JERSEY
```



PROD_ID	NAME	STATE
22	BELT	VIRGINIA
23	SANTA CAP	CALIFORNIA
24	WIRE CONNECTOR	TEXAS

PROD_ID	NAME	STATE
25	MATTRESS	FLORIDA
26	EAR PHONES	ARIZONA
27	CURTAINS	ALASKA
28	BATHROBE	MICHIGAN
31	TUNICS	CALIFORNIA
32	SOFA	OHIO
33	PROTEIN SUPPLEMENTS	COLORADO
34	BED SHEET	INDIANA
35	BOOTS	CALIFORNIA
37	WRIST WATCH	HAWAII
38	BED LINEN	WISCONSIN

PROD_ID	NAME	STATE
39	SHIRT	INDIANA
39	SHIRT	VIRGINIA
40	JACKET	VIRGINIA
40	JACKET	COLORADO
40	JACKET	INDIANA
40	JACKET	NEW MEXICO
41	CLUTCH	OHIO
43	RING	PENNSYLVANIA
43	RING	OHIO
45	SUNGLASSES	NEW MEXICO
47	CHAIR	NEW MEXICO

PROD_ID	NAME	STATE
47	CHAIR	VIRGINIA
47	CHAIR	PENNSYLVANIA
47	CHAIR	GEORGIA
47	CHAIR	OHIO
49	TABLE	OHIO
49	TABLE	PENNSYLVANIA

Time Taken In Oracle:00:00:27.50

Hive Output

```
Kill Command = /DCNFS/applications/cdh/5.16/app/CDH-5.16.2-1.cdh5.16.2.p0.8/lib/hadoop/bin/hadoop job  -kill job_1605809401632_0093
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2020-11-30 14:15:06,071 Stage-3 map = 0%,  reduce = 0%
2020-11-30 14:15:11,311 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 2.15 sec
2020-11-30 14:15:15,541 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 4.13 sec
MapReduce Total cumulative CPU time: 4 seconds 130 msec
Ended Job = job_1605809401632_0093
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 4.13 sec   HDFS Read: 15029 HDFS Write: 1038 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 130 msec
OK
1       HOT WHEELS      OHIO
1       HOT WHEELS      FLORIDA
1       HOT WHEELS      NORTH CAROLINA
2       CLAY    CALIFORNIA
3       PAINT BRUSH     FLORIDA
3       PAINT BRUSH     CALIFORNIA
5       WATER COLORS    CALIFORNIA
6       CRAYONS CALIFORNIA
7       BEYBLADE        ALASKA
7       BEYBLADE        COLORADO
9       JINGLE BELLS    NEW MEXICO
10      ELECTRIC LIGHTS NEW MEXICO
10      ELECTRIC LIGHTS ALASKA
12      FOOTBALL        TEXAS
13      LANTERNS        OHIO
14      CANDLES ARIZONA
15      SKATEBOARD      MICHIGAN
20      LEGGINS NORTH CAROLINA
21      PLANT POT       NEW JERSEY
22      BELT    VIRGINIA
23      SANTA CAP       CALIFORNIA
24      WIRE CONNECTOR  TEXAS
25      MATTRESS        FLORIDA
26      EAR PHONES      ARIZONA
27      CURTAINS        ALASKA
28      BATHROBE        MICHIGAN
31      TUNICS  CALIFORNIA
```

Time to execute on Hive: 58.843seconds

Theta Join does not work in Hive!!

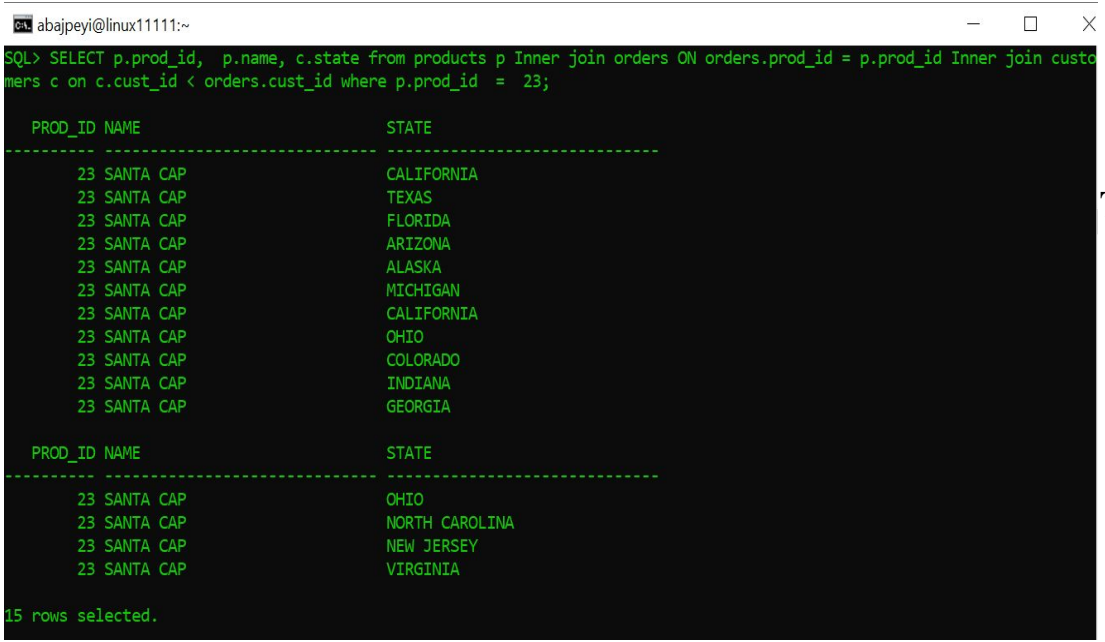
Oracle:

SELECT p.prod_id, p.name, c.state from products p Inner join orders ON orders.prod_id = p.prod_id Inner join customers c on **c.cust_id < orders.cust_id** where p.prod_id = 23 ;

Hive:

SELECT p.prod_id, p.name, c.state from products p Inner join orders ON orders.prod_id = p.prod_id Inner join customers c on **c.cust_id < orders.cust_id** where p.prod_id = 23 ;

Oracle Output



Time taken on Oracle: 00:00:15.66

Hive Output

```
hive> SELECT p.prod_id,  p.name, c.state from products p Inner join orders ON orders.prod_id = p.prod_id Inner join customers c on c.cust_id < orders.cust_id where p.prod_id = 23;
FAILED: SemanticException [Error 10017]: Line 1:125 Both left and right aliases encountered in JOIN 'cust_id'
hive>
```

Theta Join Does Not execute in Hive

9. COMPLEX JOIN- Four way join

Query: To track orders and payments by customer with id = 1

Oracle Query:

```
select c.name, o.order_id, p.prod_id, p.name, pay.payment_id from
products p inner join orders o on p.prod_id = o.prod_id
inner join customers c on c.cust_id = o.cust_id
inner join payments pay on pay.cust_id = c.cust_id and o.order_id = pay.order_id
where c.cust_id = 1 order by (p.prod_id);
```

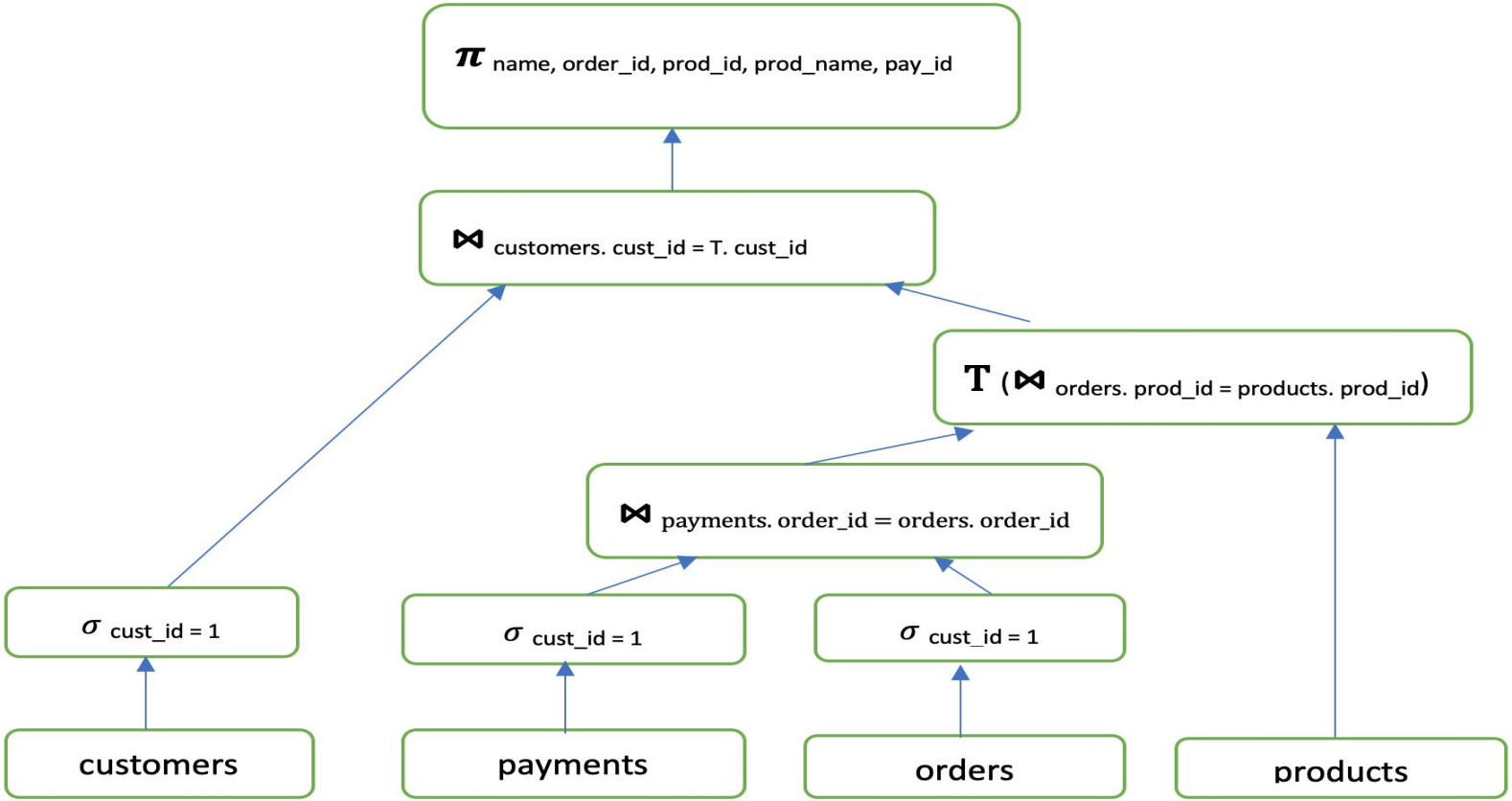
Hive Query:

```
select c.name, o.order_id, p.prod_id, p.name, pay.payment_id from
products p inner join orders o on p.prod_id = o.prod_id
inner join customers c on c.cust_id = o.cust_id
inner join payments pay on pay.cust_id = c.cust_id and o.order_id =  pay.order_id
where c.cust_id = 1 order by (p.prod_id);
```

Explain Oracle Plan

```
PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
Plan hash value: 940326620

-------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                |     1 |    48 |     8  (13)| 00:00:01 |
|   1 |  SORT ORDER BY                     |                |     1 |    48 |     8  (13)| 00:00:01 |
|   2 |   NESTED LOOPS                     |                |       |       |            |          |
|   3 |    NESTED LOOPS                    |                |     1 |    48 |     7   (0)| 00:00:01 |
|   4 |     NESTED LOOPS                   |                |     1 |    35 |     6   (0)| 00:00:01 |
|   5 |      NESTED LOOPS                  |                |     1 |    26 |     4   (0)| 00:00:01 |
|   6 |       TABLE ACCESS BY INDEX ROWID  | CUSTOMERS      |     1 |    16 |     1   (0)| 00:00:01 |
|*  7 |        INDEX UNIQUE SCAN           | SYS_C00868469  |     1 |       |     0   (0)| 00:00:01 |
|*  8 |       TABLE ACCESS FULL            | PAYMENTS       |     1 |    10 |     3   (0)| 00:00:01 |
|*  9 |      TABLE ACCESS BY INDEX ROWID   | ORDERS         |     1 |     9 |     2   (0)| 00:00:01 |
|* 10 |       INDEX RANGE SCAN             | SYS_C00868471  |     1 |       |     1   (0)| 00:00:01 |
|* 11 |     INDEX UNIQUE SCAN              | SYS_C00868470  |     1 |       |     0   (0)| 00:00:01 |
|  12 |    TABLE ACCESS BY INDEX ROWID     | PRODUCTS       |     1 |    13 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   7 - access("C"."CUST_ID"=1)
   8 - filter("PAY"."CUST_ID"=1)
   9 - filter("O"."CUST_ID"=1)
  10 - access("O"."ORDER_ID"="PAY"."ORDER_ID")
  11 - access("P"."PROD_ID"="O"."PROD_ID")
```

Proposed Query Plan



Oracle Output

```
NAME                           ORDER_ID    PROD_ID NAME                           PAYMENT_ID
------------------------------ ---------- ---------- ------------------------------ ----------
ALEX ROSS                              10          3 PAINT BRUSH                           110
ALEX ROSS                              10          5 WATER COLORS                          110
ALEX ROSS                              10          6 CRAYONS                               110

SQL> timing stop
Elapsed: 00:00:08.68
```

Hive Output

```
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 1
2020-11-30 10:57:47,403 Stage-4 map = 0%,  reduce = 0%
2020-11-30 10:57:52,717 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 2.49 sec
2020-11-30 10:57:58,029 Stage-4 map = 100%,  reduce = 100%, Cumulative CPU 4.12 sec
MapReduce Total cumulative CPU time: 4 seconds 120 msec
Ended Job = job_1605809401632_0086
MapReduce Jobs Launched:
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 4.12 sec   HDFS Read: 18239 HDFS Write: 90 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 120 msec
OK
ALEX ROSS       10      3       PAINT BRUSH     110
ALEX ROSS       10      5       WATER COLORS    110
ALEX ROSS       10      6       CRAYONS 110
Time taken: 63.934 seconds, Fetched: 3 row(s)
```

Elapsed Time:
Time taken by Oracle: 00:00:08.68 sec
Time taken by Hive: 63.934 sec

10. SELF JOIN

The below Query gives the list of 10 customers belonging to the same state.

Oracle:

SELECT t.ID,t.CustomerName_from_c1,t.CustomerName_from_c2,t.State from (select c1.cust_id ID,c1.name CustomerName_from_c1, c2.name CustomerName_from_c2, c1.state State from customers c1 JOIN customers c2 on c1.state = c2.state and c1.name <>c2.name and c1.cust_id<=10 order by c1.cust_id) t where rownum < = 10;

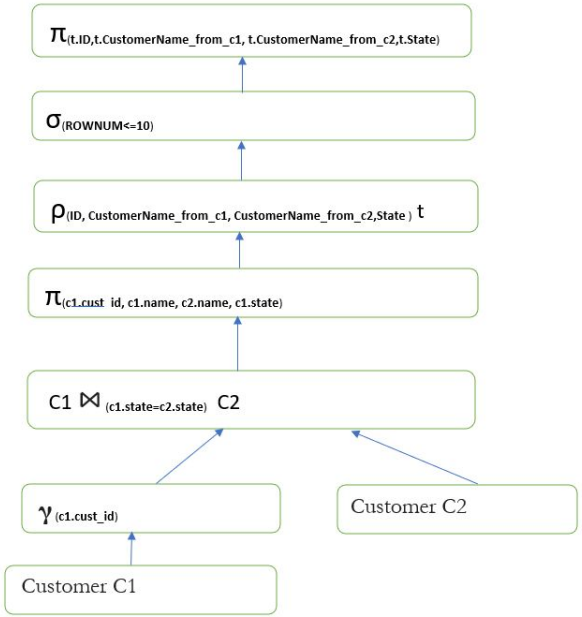
Hive:

select c1.cust_id,c1.name, c2.name, c1.state from customers c1 JOIN customers c2 on c1.state = c2.state where c1.name <>c2.name order by c1.cust_id limit 10;

Oracle Explain Plan

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------
Plan hash value: 1674776304

-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |    10 |   640 |     7  (29)| 00:00:01 |
|*  1 |  COUNT STOPKEY                |               |       |       |            |          |
|   2 |   VIEW                        |               |    21 |  1344 |     7  (29)| 00:00:01 |
|*  3 |    SORT ORDER BY STOPKEY      |               |    21 |   987 |     7  (29)| 00:00:01 |
|*  4 |     HASH JOIN                 |               |    21 |   987 |     6  (17)| 00:00:01 |
|   5 |      TABLE ACCESS BY INDEX ROWID| CUSTOMERS     |    10 |   250 |     2   (0)| 00:00:01 |
|*  6 |       INDEX RANGE SCAN        | SYS_C00868469 |    10 |       |     1   (0)| 00:00:01 |
|   7 |      TABLE ACCESS FULL        | CUSTOMERS     |    50 |  1100 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(ROWNUM<=10)
   3 - filter(ROWNUM<=10)
   4 - access("C1"."STATE"="C2"."STATE")
       filter("C1"."NAME"<>"C2"."NAME")
   6 - access("C1"."CUST_ID"<=10)

23 rows selected.

SQL>
```

Proposed Query Plan



Oracle Output

ID	CUSTOMERNAME_FROM_C1	CUSTOMERNAME_FROM_C2	STATE
1	ALEX ROSS	PETER MILLER	CALIFORNIA
1	ALEX ROSS	NANCY SHEPHERD	CALIFORNIA
1	ALEX ROSS	CHRIS DIAZ	CALIFORNIA
1	ALEX ROSS	GROVER PARKER	CALIFORNIA
1	ALEX ROSS	SHANE ROSS	CALIFORNIA
3	SHANE WARNE	CARL WARNE	FLORIDA
4	ADAM DIAZ	DREW DIAZ	ARIZONA
5	JIM BAKER	ESTHER BAKER	ALASKA
6	ALEX PARKER	FRANK BAKER	MICHIGAN
7	PETER MILLER	ALEX ROSS	CALIFORNIA

10 rows selected.

Time taken to execute in oracle: 00:00:08.07

Hive Output

```
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-11-29 21:51:16,593 Stage-2 map = 0%,  reduce = 0%
2020-11-29 21:51:20,876 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 2.6 sec
2020-11-29 21:51:26,153 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 4.69 sec
MapReduce Total cumulative CPU time: 4 seconds 690 msec
Ended Job = job_1605809401632_0081
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.69 sec   HDFS Read: 13499 HDFS Write: 349 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 690 msec
OK
1       ALEX ROSS       GROVER PARKER   CALIFORNIA
1       ALEX ROSS       SHANE ROSS      CALIFORNIA
1       ALEX ROSS       NANCY SHEPHERD  CALIFORNIA
1       ALEX ROSS       PETER MILLER    CALIFORNIA
1       ALEX ROSS       CHRIS DIAZ      CALIFORNIA
3       SHANE WARNE     CARL WARNE      FLORIDA
4       ADAM DIAZ       DREW DIAZ       ARIZONA
5       JIM BAKER       ESTHER BAKER    ALASKA
6       ALEX PARKER     FRANK BAKER     MICHIGAN
7       PETER MILLER    GROVER PARKER   CALIFORNIA
Time taken: 56.229 seconds, Fetched: 10 row(s)
hive>
```

Time to execute on Hive:
56.229 seconds

Conclusion

- Oracle SQL is more efficient for most of our queries when compared to Hive.
- Hadoop / Hive does not enforce any constraints on table/data while writing into tables but it does while reading, hence it is called ‘schema on read’. Because of this property, Hive takes less time to load the data but more time to query as it verifies the data against the schema during query execution.
- Oracle is highly efficient to work on small datasets and row level DML operations than Hive / Hadoop.
- Hive works extremely well with large data sets (E.g. 100 TB) with batch jobs that have high latency. That is “select * from <table>” will execute quickly in Hive.
- Hive has a slower performance for aggregate, joins, count type of queries.

Conclusion

Oracle	Hive
1. Explicit BEGIN, COMMIT and ROLLBACK model is supported for transactions.	All operations are auto committed. No explicit support of transactions as in Oracle is available.
2. Joins on operators like <,> are also supported.	Only equality joins are supported in Hive.
3. If the data being loaded doesn't conform to schema, then load is terminated. This design is called 'schema on write'.	Doesn't verify data when it is loaded, rather when it is retrieved. This is called 'schema on read'.
4. Supports referential integrity and foreign keys.	Does not support it.
5. An executing query can be cancelled.	Does not support cancelling of an executing query.
6. Supports the MINUS set operator	Does not support the MINUS set operator

Thank You!!
Q/A?