CREATING TABLES WITH BUCKETS

CREATING TABLES WITH BUCKETS

WE USE PARTITIONING TO SPLIT THE DATA AND OPTIMIZE FREQUENT QUERIES

CREATING TABLES WITH BUCKETS

BUCKETING IS ANOTHER TECHNIQUE
TO SPLIT THE DATA

CREATING TABLES WITH BUCKETS

BUCKETS CAN APPLY TO **BOTH** PARTITIONED AND NON-PARTITIONED TABLES

CREATING TABLES WITH BUCKETS

WHEN DO WE USE BUCKETING VS PARTITIONING?

PARTITIONS SHOULD BE CREATED WITH CARE

CREATING TABLES WITH BUCKETS

WHEN DO WE USE BUCKETING VS PARTITIONING?

THE NUMBER OF PARTITIONS SHOULD NOT BE VERY HIGH

ELSE THE NAME NODE THAT STORES THE METADATA OF DIRECTORIES WILL BE OVERWHELMED

CREATING TABLES WITH BUCKETS

WHEN DO WE USE BUCKETING VS PARTITIONING?

IN FACT, HIVE RESTRICTS THE MAX NUMBER OF PARTITIONS

TO AVOID THIS VERY SCENARIO

CREATING TABLES WITH BUCKETS

WHEN DO WE USE BUCKETING VS PARTITIONING?

IN FACT, HIVE RESTRICTS THE MAX NUMBER OF PARTITIONS

TO AVOID THIS VERY SCENARIO

WHEN DO WE USE BUCKETING VS PARTITIONING?

LET'S SAY YOU HAVE A TABLE WITH USER IDS

THIS TABLE IS QUERIED OFTEN BASED ON SPECIFIC USER IDS

WHEN DO WE USE BUCKETING VS PARTITIONING?

IF YOU CREATE PARTITIONS BASED ON USER IDS

HIVE WILL NOT ALLOW IT IF THAT LEADS TO TOO MANY PARTITIONS

WHEN DO WE USE BUCKETING VS PARTITIONING?

WHAT IF YOU COULD BREAK THE DATA INTO BUCKETS

SO THAT YOU ALWAYS KNOW WHICH
BUCKET A SPECIFIC USER ID WILL BELONG TO

WHEN DO WE USE BUCKETING VS PARTITIONING?

THEN YOU ONLY NEED TO LOOK AT 1 BUCKET

TO FIND THE RECORDS
CORRESPONDING TO 1 USER

WHEN DO WE USE BUCKETING VS PARTITIONING?

THIS IS EXACTLY WHERE
BUCKETING COMES INTO PLAY

HIVE CAN ORGANIZE TABLES FURTHER INTO BUCKETS

SPECIFY THE NUMBER OF BUCKETS LIKE THIS

```
clustered by (COLUMN_NAME) INTO N Buckets
```

N IS THE NUMBER OF BUCKETS!

```
clustered by (COLUMN_NAME) INTO N Buckets
```

HIVE DETERMINES WHICH BUCKET EACH RECORD BELONGS TO BY USING THE COLUMN VALUE

E.G. FOR AN INTEGER COLUMN

FIND A HASH OF THE COLUMN VALUE

USE THE MODULO OPERATOR TO FIND A BUCKET

```
clustered by (COLUMN_NAME) INTO N Buckets
```

ALL ROWS WITH THE SAME VALUE FOR THE BUCKETING COLUMN

WILL BELONG TO 1 BUCKET

```
clustered by (COLUMN_NAME) INTO N Buckets
```

THIS IS THE STUDENTS TABLE

StudentID	FirstName	LastName	Gender	Email
1	Janani	Ravi	F	janani@loonycorn.com
2	Swetha	Kolalapudi	F	swetha@loonycorn.com
3	Navdeep	Singh	M	navdeep@loonycorn.com
4	Anu	Radha	F	anuradha@gmail.com
5	Vitthal	Srinivasan	M	vitthal@loonycorn.com
6	Jitendra	Kedia	M	jitendra@loonycorn.com

SPLIT INTO 3 BUCKETS BASED ON THE STUDENTID COLUMN

SPLIT INTO 3 BUCKETS BASED ON THE STUDENTID COLUMN

Stude	FirstN	LastName	Gend	Email
1	Janani	Ravi	F	janani@loonycorn.com
2	Sweth	Kolalapudi	F	swetha@loonycorn.com
3	Navde	Singh	M	navdeep@loonycorn.com
4	Anu	Radha	F	anuradha@gmail.com
5	Vitthal	Srinivasan	M	vitthal@loonycorn.com
6	Jitendr	Kedia	M	jitendra@loonycorn.com

THE 3 BUCKETS ARE BUCKET 0, BUCKET 1 AND BUCKET 2

CONSIDER ONE ROW AT A TIME TO SEE WHICH BUCKET IT FALLS INTO

StudentID	FirstName	LastName	Gender	Email
1	Janani	Ravi	F	janani@loonycorn.com

1 modulo 3 = BUCKET 1

SPLIT INTO 3 BUCKETS BASED ON THE STUDENTID COLUMN

Stude	FirstN	LastName	Gend	Email
1	Janani	Ravi	F	janani@loonycorn.com
2	Sweth	Kolalapudi	F	swetha@loonycorn.com
3	Navde	Singh	M	navdeep@loonycorn.com
4	Anu	Radha	F	anuradha@gmail.com
5	Vitthal	Srinivasan	M	vitthal@loonycorn.com
6	Jitendr	Kedia	M	jitendra@loonycorn.com

THE 3 BUCKETS ARE BUCKET 0, BUCKET 1 AND BUCKET 2

StudentID	FirstName	LastName	Gender	Email
2	Swetha	Kolalapudi	F	swetha@loonycorn.com

2 modulo 3 = BUCKET 2

SPLIT INTO 3 BUCKETS BASED ON THE STUDENTID COLUMN

Stude	FirstN	LastName	Gend	Email
1	Janani	Ravi	F	janani@loonycorn.com
2	Sweth	Kolalapudi	F	swetha@loonycorn.com
3	Navde	Singh	M	navdeep@loonycorn.com
4	Anu	Radha	F	anuradha@gmail.com
5	Vitthal	Srinivasan	M	vitthal@loonycorn.com
6	Jitendr	Kedia	M	jitendra@loonycorn.com

THE 3 BUCKETS ARE BUCKET 0, BUCKET 1 AND BUCKET 2

StudentID	FirstName	LastName	Gender	Email
3	Navdeep	Singh	M	navdeep@loonycorn.com

3 modulo 3 = BUCKET 0

SPLIT INTO 3 BUCKETS BASED ON THE STUDENTID COLUMN

Stude	FirstN	LastName	Gend	Email
1	Janani	Ravi	F	janani@loonycorn.com
2	Sweth	Kolalapudi	F	swetha@loonycorn.com
3	Navde	Singh	M	navdeep@loonycorn.com
4	Anu	Radha	F	anuradha@gmail.com
5	Vitthal	Srinivasan	M	vitthal@loonycorn.com
6	Jitendr	Kedia	M	jitendra@loonycorn.com

THE 3 BUCKETS ARE BUCKET 0, BUCKET 1 AND BUCKET 2

StudentID	FirstName	LastName	Gender	Email
4	Anu	Radha	F	anuradha@gmail.com

4 modulo 3 = BUCKET 1

SPLIT INTO 3 BUCKETS BASED ON THE STUDENTID COLUMN

Stude	FirstN	LastName	Gend	Email
1	Janani	Ravi	F	janani@loonycorn.com
2	Sweth	Kolalapudi	F	swetha@loonycorn.com
3	Navde	Singh	M	navdeep@loonycorn.com
4	Anu	Radha	F	anuradha@gmail.com
5	Vitthal	Srinivasan	M	vitthal@loonycorn.com
6	Jitendr	Kedia	M	jitendra@loonycorn.com

THE 3 BUCKETS ARE BUCKET 0, BUCKET 1 AND BUCKET 2

StudentID	FirstName	LastName	Gender	Email
5	Vitthal	Srinivasan	M	vitthal@loonycorn.com

5 modulo 3 = BUCKET 2

SPLIT INTO 3 BUCKETS BASED ON THE STUDENTID COLUMN

Stude	FirstN	LastName	Gend	Email
1	Janani	Ravi	F	janani@loonycorn.com
2	Sweth	Kolalapudi	F	swetha@loonycorn.com
3	Navde	Singh	M	navdeep@loonycorn.com
4	Anu	Radha	F	anuradha@gmail.com
5	Vitthal	Srinivasan	M	vitthal@loonycorn.com
6	Jitendr	Kedia	M	jitendra@loonycorn.com

THE 3 BUCKETS ARE BUCKET 0, BUCKET 1 AND BUCKET 2

StudentID	FirstName	LastName	Gender	Email
6	Jitendra	Kedia	M	jitendra@loonycorn.com

6 modulo 3 = BUCKET 0

StudentID	FirstName	LastName	Gender	Email
1	Janani	Ravi	F	janani@loonycorn.com
2	Swetha	Kolalapudi	F	swetha@loonycorn.com
3	Navdeep	Singh	M	navdeep@loonycorn.com
4	Anu	Radha	F	anuradha@gmail.com
5	Vitthal	Srinivasan	M	vitthal@loonycorn.com
6	Jitendra	Kedia	M	jitendra@loonycorn.com

BUCKET 0

StudentID	FirstName	LastName	Gender	Email
3	Swetha	Kolalapudi	F	swetha@loonycorn.com
6	Jitendra	Kedia	M	jitendra@loonycorn.com

BUCKET 1

StudentID	FirstName	LastName	Gender	Email
1	Janani	Ravi	F	janani@loonycorn.com
4	Anu	Radha	F	anuradha@gmail.com

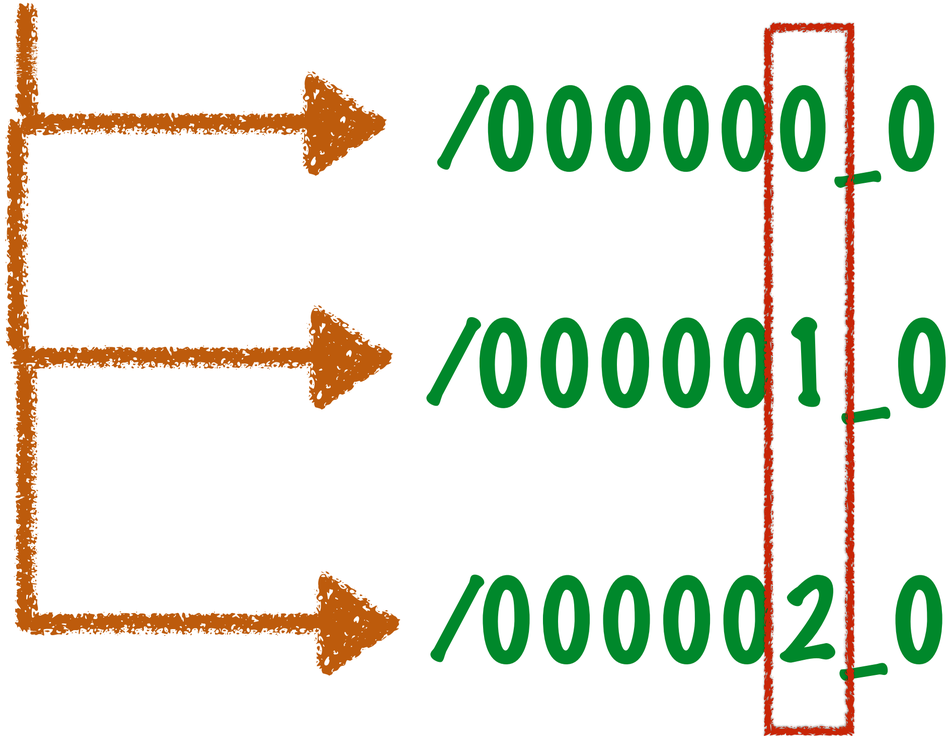
BUCKET 2

StudentID	FirstName	LastName	Gender	Email
2	Swetha	Kolalapudi	F	swetha@loonycorn.com
5	Vitthal	Srinivasan	M	vitthal@loonycorn.com

/USER/HIVE/WAREHOUSE

/STUDENTS-TABLE

DATA IS STORED IN THESE 3 BUCKETS



EACH BUCKET HAS A CORRESPONDING FILE

ADVANTAGES OF BUCKETING

BUCKETING OFFERS MORE EFFICIENT SAMPLING OF DATA THAN NON-BUCKETED TABLES

SAMPLING IS USEFUL TO QUERY A FRACTION OF THE DATA WHEN RUNNING TESTS AND FOR DEBUGGING

THIS IS VERY USEFUL WHEN DATA SETS ARE HUGE!

ADVANTAGES OF BUCKETING

BUCKETING MAKES MAP-SIDE JOINS MUCH FASTER

JOINS HAVE A LEFT TABLE AND A RIGHT TABLE

ADVANTAGES OF BUCKETING

BUCKETING MAKES MAP-SIDE JOINS MUCH FASTER

IN A MAP-SIDE JOIN A MAPPER WHICH IS PROCESSING A LEFT TABLE

KNOWS EXACTLY WHICH BUCKET TO TO LOOK IN TO FIND THE ROW IN THE RIGHT TABLE

INSTEAD OF LOADING THE ENTIRE RIGHT TABLE

ADVANTAGES OF BUCKETING

BUCKETING MAKES MAP-SIDE JOINS MUCH FASTER

INSTEAD OF LOADING THE ENTIRE RIGHT TABLE

ONLY A SMALL FRACTION OF THE RIGHT TABLE NEEDS TO BE LOADED

ONLY THE BUCKET WHICH HOLDS THE ROWS WHICH ARE PART OF THE JOIN

ADVANTAGES OF BUCKETING

BUCKETING ALLOWS ROWS IN EACH BUCKET TO BE SORTED BY ONE OR MORE COLUMNS

ADVANTAGES OF BUCKETING

BUCKETING ALLOWS ROWS IN EACH BUCKET TO BE SORTED BY ONE OR MORE COLUMNS

THIS MAKES MAP-JOINS EVEN MORE EFFICIENT

THE JOIN OF EACH BUCKET IS AN EFFICIENT MERGE-SORT

ADVANTAGES OF BUCKETING

BUCKETING, LIKE PARTITIONING IMPROVES QUERY PERFORMANCE

RUNNING QUERIES ON SMALLER DATA SETS MAKES QUERIES MORE EFFICIENT

WHY DO WE BUCKET THE TABLE WHEN WE ALREADY HAVE PARTITIONING?

THE NUMBER OF BUCKETS IS FIXED
AND NOT DEPENDENT ON THE
VALUE OF THE COLUMN

IN ORDER TO GET SPLITS OF THE DATA
WHERE THE NUMBER OF SPLITS IS LIMITED
WE USE BUCKETS RATHER THAN PARTITIONS

WHY DO WE BUCKET THE TABLE WHEN WE ALREADY HAVE PARTITIONING?

ONE HAS TO BE CAREFUL WHILE PARTITIONING. IF YOU DON'T PARTITION CORRECTLY ON THE RIGHT COLUMN, YOU MAY END UP WITH MILLIONS OF DIRECTORIES IN YOUR FILE SYSTEM

HOW DO WE BUCKET TABLES?

HOW DO WE BUCKET TABLES?

REVIEWS

REVIEWID	MovieID	Review	RatingStars
1	1	Amazing	5
2	1	Genre-Defining	5
3	1	Classic	5
4	1	Overrated	1
5	1	OK, Not Great	3
6	1	Two Thumbs Up	5
7	3	Crossover Hit	5
8	3	Love It	4
9	3	Nailbiting!	5
10	1	Cinematic Masterpiece	5
11	5	So Bad Its Good	0
12	2	Not As Good As Original	3
13	2	Overrated	2
14	2	Too Morbid	3
15	4	Sad, Thought-Provoking	4
16	4	Cinema At Its Best	5

THIS IS A TABLE WHICH HOLDS MOVIE REVIEWS, SOMETHING LIKE WHAT MIGHT BE USED AT IMDB

HOW DO WE BUCKET TABLES?

THIS IS A TABLE WHICH HOLDS MOVIE REVIEWS, SOMETHING LIKE WHAT MIGHT BE USED AT IMDB

WE WANT 4 BUCKETS BASED ON REVIEW ID

REVIEWS

ReviewWID	MovieID	Review	RatingStars
1	1	Amazing	5
2	1	Genre-Defining	5
3	1	Classic	5
4	1	Overrated	1
5	1	OK, Not Great	3
6	1	Two Thumbs Up	5
7	3	Crossover Hit	5
8	3	Love It	4

HOW DO WE BUCKET TABLES?

BY ADDING

```
Clustered by(column name) into N buckets
```

TO THE CREATE TABLE COMMAND

HOW DO WE BUCKET TABLES?

CREATE TABLE COMMAND FOR THE REVIEWS TABLE

```
create table Reviews
(
MovieID INT,
ReviewID BIGINT,
Review VARCHAR(100),
Rating TINYINT
)
```

REVIEWS

ReviewWID	MovieID	Review	RatingStars
1	1	Amazing	5
2	1	Genre-Defining	5
3	1	Classic	5
4	1	Overrated	1
5	1	OK, Not Great	3

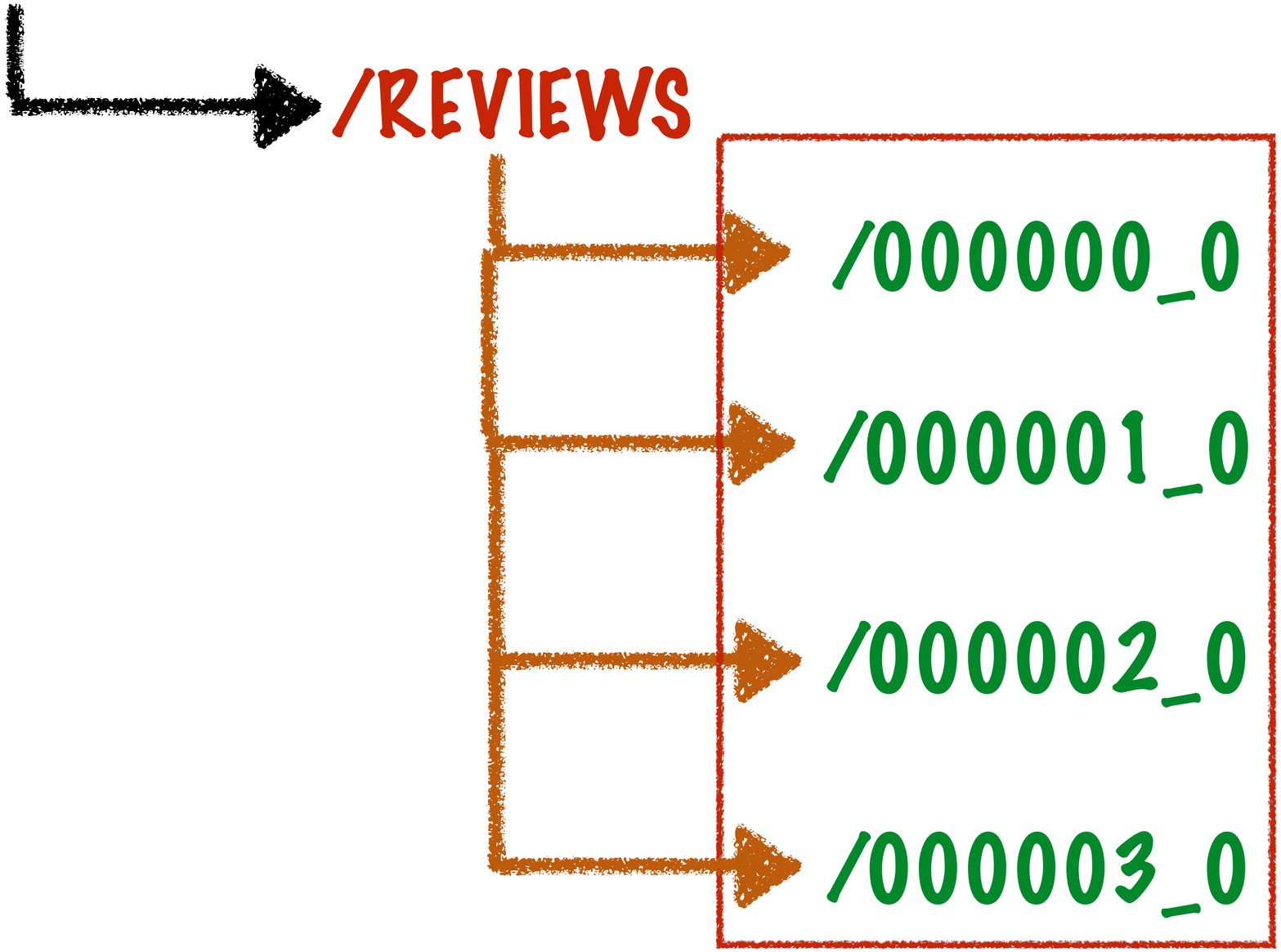
WE WANT TO BUCKET IT ON REVIEWID IN 4 BUCKETS

BY ADDING

```
Clustered by(column name) into N buckets
```

```
create table Reviews
(
MovieID INT,
ReviewID BIGINT,
Review VARCHAR(100),
Rating TINYINT
)
clustered by (ReviewID) INTO 4 Buckets;
```

/USER/HIVE/WAREHOUSE



DATA IS STORED IN THESE 4 FILES(BUCKETS)

SETTING UP A TABLE WITH BOTH PARTITIONING AND BUCKETING

SETTING UP A TABLE WITH BOTH PARTITIONING AND BUCKETING

CONSIDER THE SALES DATA OF AN E-COMMERCE FRUIT SELLER

StoreID	ProductID	OrderDate	Revenue
1	1	January 18,2016	8,236.33
1	3	January 18,2016	7,455.67
1	4	January 18,2016	5,316.89
1	2	January 18,2016	2,433.76
2	1	January 18,2016	9,456.01
2	3	January 18,2016	3,644.33
2	4	January 18,2016	8,988.64
2	2	January 18,2016	1,621.58
1	1	January 17,2016	2342.33
1	3	January 17,2016	6345.10
1	4	January 17,2016	5673.01
1	2	January 17,2016	4543.98
2	1	January 17,2016	8902.65
2	3	January 17,2016	9114.67
2	4	January 17,2016	5102.05
2	2	January 17,2016	1299.45

SETTING UP A TABLE WITH BOTH PARTITIONING AND BUCKETING

CONSIDER THE SALES DATA OF AN E-COMMERCE FRUIT SELLER

WE WANT TO STORE ITS DATA IN SUCH A WAY THAT IT IS

PARTITIONED ON STOREID

BUCKETED ON PRODUCTID INTO 4 BUCKETS

StoreID	ProductID	OrderDate	Revenue
1	1	January 18,2016	8,236.33
1	3	January 18,2016	7,455.67
1	4	January 18,2016	5,316.89

SETTING UP A TABLE WITH BOTH PARTITIONING AND BUCKETING

PARTITIONED ON STOREID

BUCKETED ON PRODUCTID INTO 4 BUCKETS

StoreID	ProductID	OrderDate	Revenue

STEP 1: WRITE THE CREATE TABLE COMMAND FOR THIS TABLE

HOW TO PARTITION TABLES?

BY ADDING

```
partitioned by(Partition_Column_Name column_data_type)
```

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
```

SETTING UP A TABLE WITH BOTH PARTITIONING AND BUCKETING

PARTITIONED ON STOREID

BUCKETED ON PRODUCTID INTO 4 BUCKETS

StoreID	ProductID	OrderDate	Revenue

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
```

SETTING UP A TABLE WITH BOTH PARTITIONING AND BUCKETING

PARTITIONED ON STOREID

BUCKETED ON PRODUCTID INTO 4 BUCKETS

StoreID	ProductID	OrderDate	Revenue

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
```

SETTING UP A TABLE WITH BOTH PARTITIONING AND BUCKETING

PARTITIONED ON STOREID

BUCKETED ON PRODUCTID INTO 4 BUCKETS

StoreID	ProductID	OrderDate	Revenue

NOW TO BUCKET THIS TABLE

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
```

HOW TO BUCKET TABLES?

BY ADDING

```
Clustered by(column name) into N buckets


CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
```

SETTING UP A TABLE WITH BOTH PARTITIONING AND BUCKETING

PARTITIONED ON STOREID

BUCKETED ON PRODUCTID INTO 4 BUCKETS

StoreID	ProductID	OrderDate	Revenue

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
Clustered by(PRODUCTID) into 4 buckets;
```

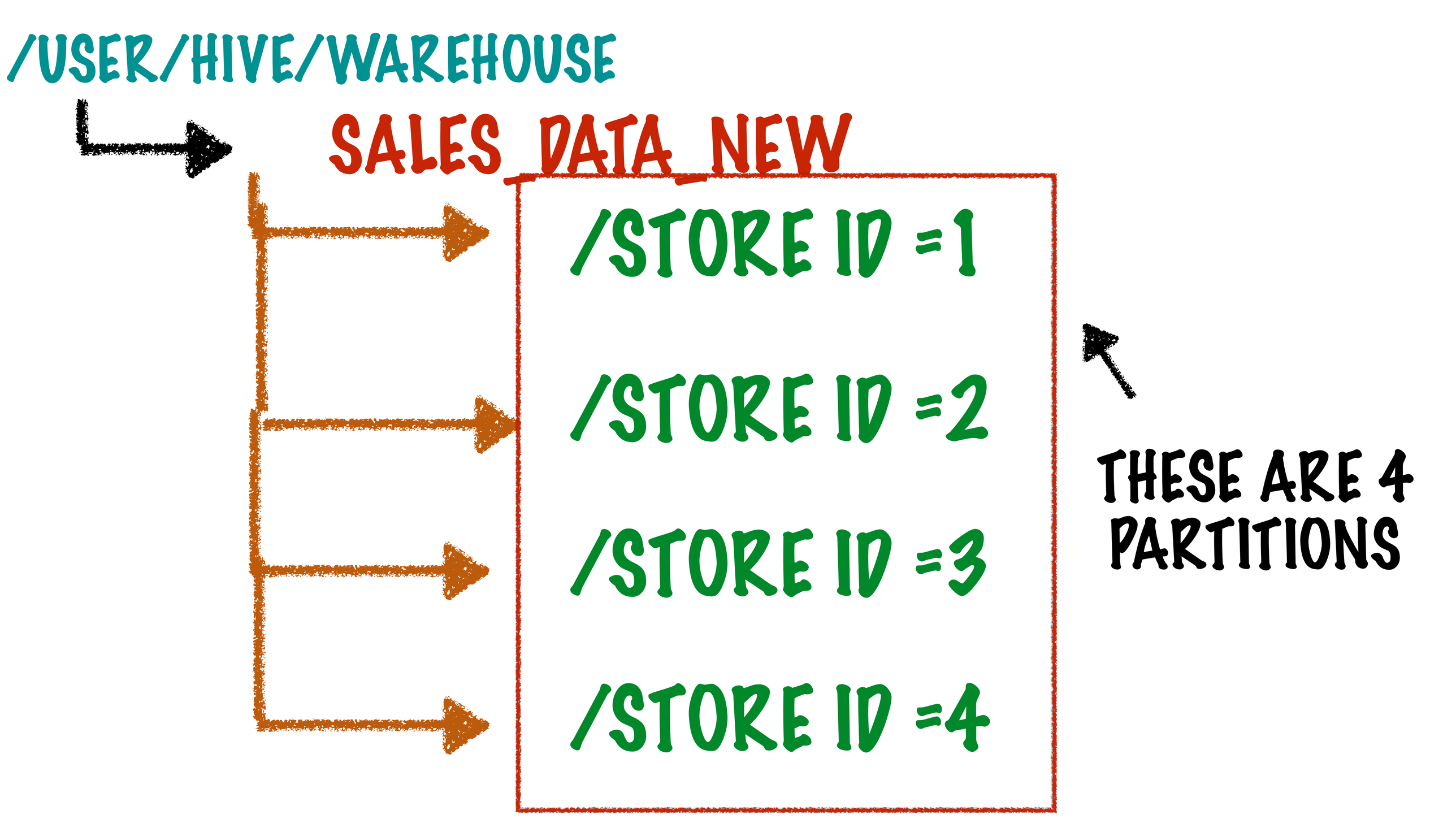
SETTING UP A TABLE WITH BOTH PARTITIONING AND BUCKETING

PARTITIONED ON STOREID
BUCKETED ON PRODUCTID INTO 4 BUCKETS

StoreID	ProductID	OrderDate	Revenue

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)

Clustered by(PRODUCTID) into 4 buckets;
```


/USER/HIVE/WAREHOUSE
SALES_DATA_NEW
/STORE ID =1
/STORE ID =2
/STORE ID =3
/STORE ID =4
THESE ARE 4 PARTITIONS

/USER/HIVE/WAREHOUSE

SALES_DATA_NEW

/STORE ID =1

/000000_0

/000000_1

/000000_2

/000000_3

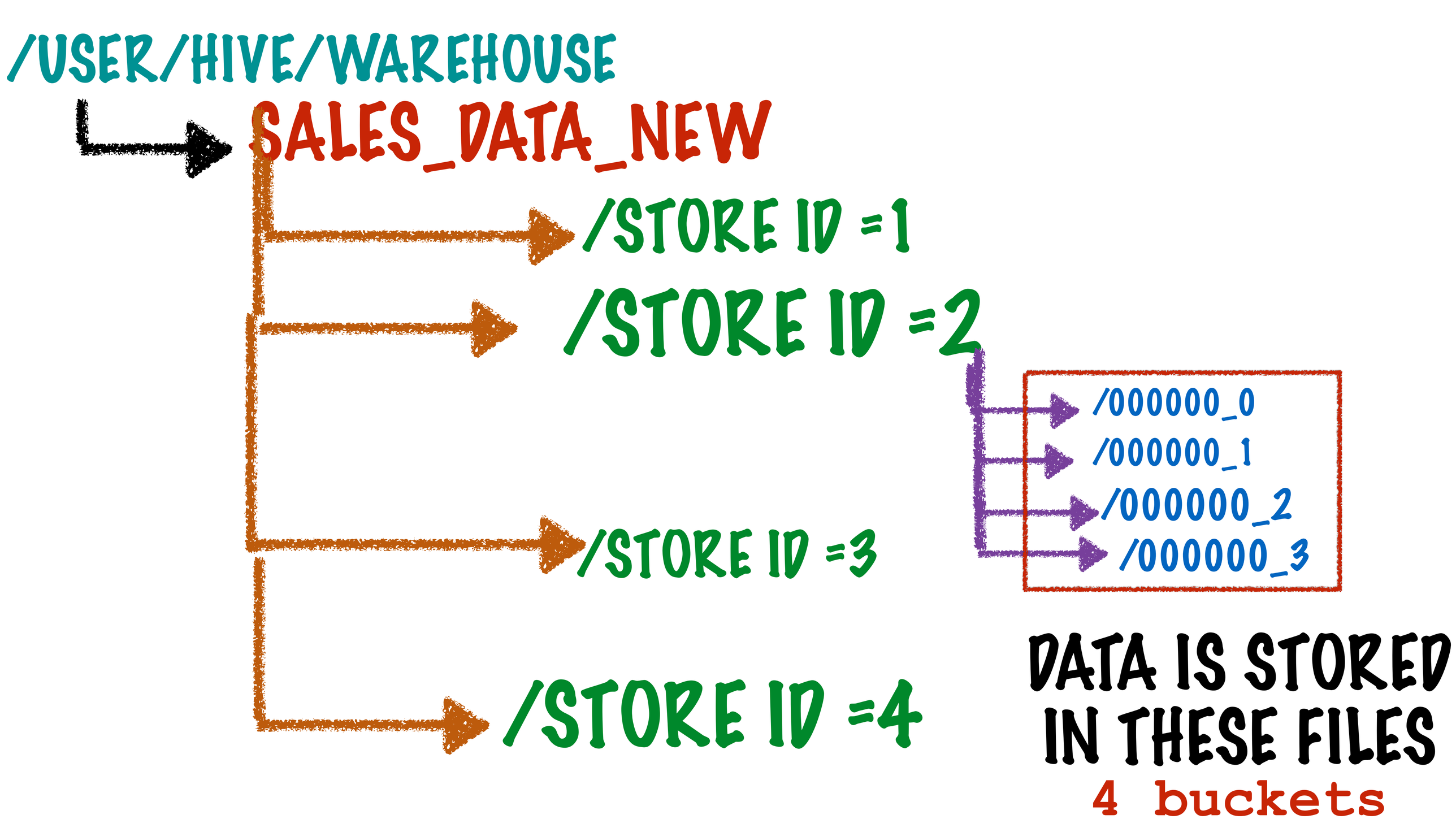
/STORE ID =2

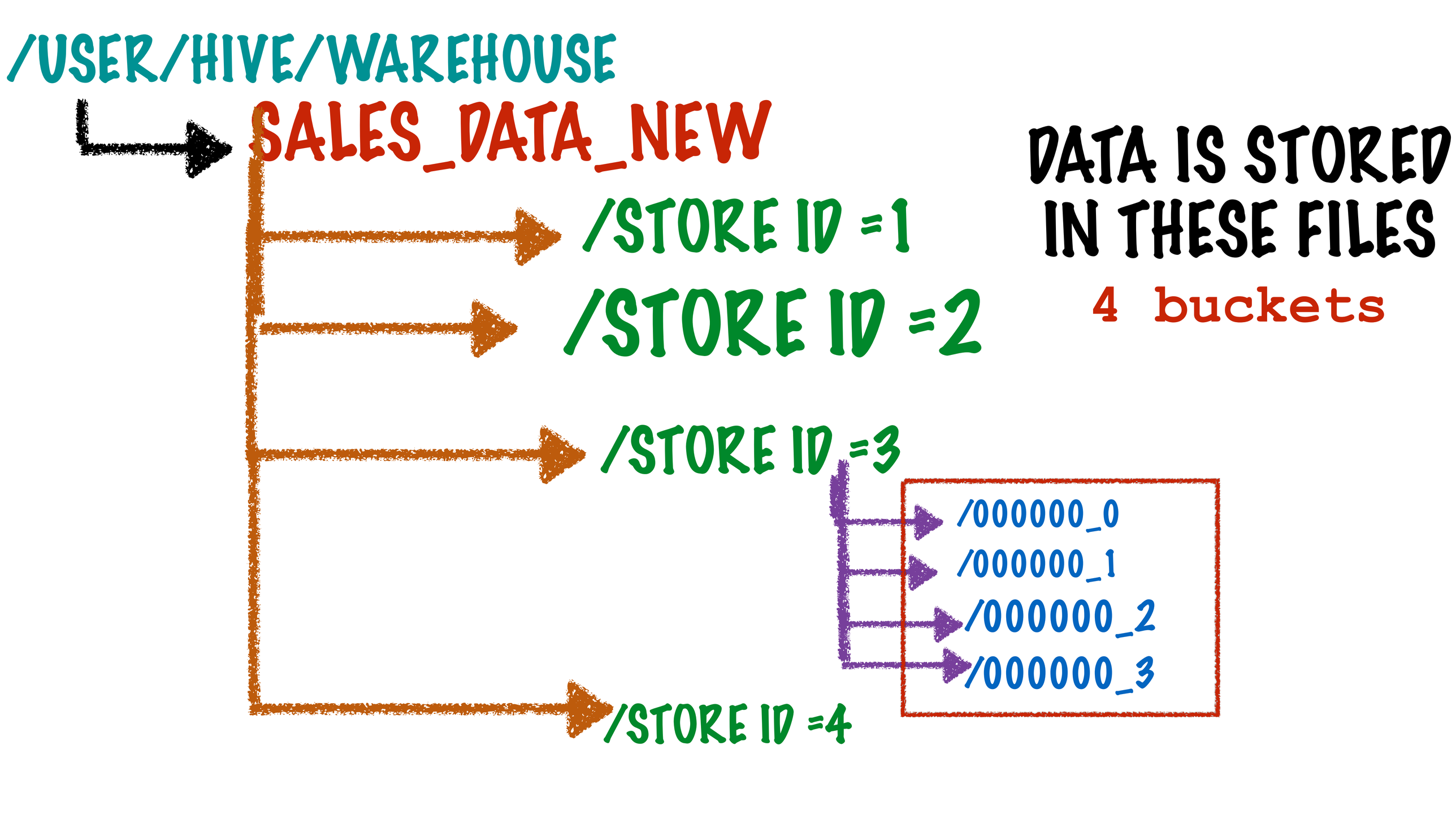
/STORE ID =3

/STORE ID =4

DATA IS STORED IN THESE FILES

4 buckets


/USER/HIVE/WAREHOUSE
SALES_DATA_NEW
/STORE ID =1
/STORE ID =2
/000000_0
/000000_1
/000000_2
/000000_3
/STORE ID =3
/STORE ID =4
DATA IS STORED IN THESE FILES
4 buckets


/USER/HIVE/WAREHOUSE
SALES_DATA_NEW
/STORE ID =1
/STORE ID =2
/STORE ID =3
/000000_0
/000000_1
/000000_2
/000000_3
/STORE ID =4
DATA IS STORED IN THESE FILES
4 buckets

/USER/HIVE/WAREHOUSE

SALES_DATA_NEW

/STORE ID =1

/STORE ID =2

/STORE ID =3

/STORE ID =4

/000000_0

/000000_1

/000000_2

/000000_3

DATA IS STORED IN THESE FILES

4 buckets

CREATE A TABLE WITH BUCKETS THAT HAVE SORTED RECORDS

CREATE A TABLE WITH BUCKETS
THAT HAVE SORTED RECORDS

BUCKETING ALLOWS THE RECORDS IN
EACH BUCKET TO BE SORTED BY ONE
OR MORE COLUMNS USING

SORTED BY

ALONG WITH CLUSTERED BY

CREATE A TABLE WITH BUCKETS THAT HAVE SORTED RECORDS

LET'S WORK WITH THE SAME SALES DATA TABLE, FOR AN E-COMMERCE FRUIT SELLER

StoreID	ProductID	OrderDate	Revenue
1	1	January 18,2016	8,236.33
1	3	January 18,2016	7,455.67
1	4	January 18,2016	5,316.89
1	2	January 18,2016	2,433.76
2	1	January 18,2016	9,456.01
2	3	January 18,2016	3,644.33
2	4	January 18,2016	8,988.64
2	2	January 18,2016	1,621.58
1	1	January 17,2016	2342.33
1	3	January 17,2016	6345.10
1	4	January 17,2016	5673.01
1	2	January 17,2016	4543.98
2	1	January 17,2016	8902.65
2	3	January 17,2016	9114.67
2	4	January 17,2016	5102.05
2	2	January 17,2016	1299.45

THE DATA CURRENTLY IS

PARTITIONED ON STOREID

BUCKETED ON PRODUCTID INTO 4 BUCKETS

StoreID	ProductID	OrderDate	Revenue

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)

Clustered by(PRODUCTID) into 4 buckets;
```

CREATE A TABLE WITH BUCKETS THAT HAVE SORTED RECORDS

WE WANT TO SORT EACH BUCKET BY 'ORDER DATE'

StoreID	ProductID	OrderDate	Revenue
1	1	January 18,2016	8,236.33
1	3	January 18,2016	7,455.67
1	4	January 18,2016	5,316.89
1	2	January 18,2016	2,433.76
2	1	January 18,2016	9,456.01
2	3	January 18,2016	3,644.33
2	4	January 18,2016	8,988.64
2	2	January 18,2016	1,621.58

CREATE A TABLE WITH BUCKETS THAT HAVE SORTED RECORDS

WE WANT TO SORT EACH BUCKET BY 'ORDER DATE'

SORTED BY ORDERDATE

StoreID	ProductID	OrderDate	Revenue
1	1	January 18,2016	8,236.33
1	3	January 18,2016	7,455.67
1	4	January 18,2016	5,316.89
1	2	January 18,2016	2,433.76
2	1	January 18,2016	9,456.01
2	3	January 18,2016	3,644.33
2	4	January 18,2016	8,988.64

CREATE A TABLE WITH BUCKETS THAT HAVE SORTED RECORDS

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
Clustered by(PRODUCTID)
SORTED BY (ORDERDATE) into 4 buckets;
```

StoreID	ProductID	OrderDate	Revenue
1	1	January 18,2016	8,236.33
1	3	January 18,2016	7,455.67

CREATE A TABLE WITH BUCKETS THAT HAVE SORTED RECORDS

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
Clustered by(PRODUCTID)
SORTED BY (ORDERDATE) into 4 buckets;
```

PARTITIONED BY STOREID

StoreID	ProductID	OrderDate	Revenue
1	1	January 18,2016	8,236.33

CREATE A TABLE WITH BUCKETS THAT HAVE SORTED RECORDS

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
Clustered by(PRODUCTID)
SORTED BY (ORDERDATE) into 4 buckets;
```

CLUSTERED BY PRODUCTID

StoreID	ProductID	OrderDate	Revenue
1	1	January 18,2016	8,236.33

CREATE A TABLE WITH BUCKETS THAT HAVE SORTED RECORDS

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
Clustered by(PRODUCTID)
SORTED BY (ORDERDATE) into 4 buckets;
```

SORTED BY ORDERDATE

StoreID	ProductID	OrderDate	Revenue
1	1	January 18,2016	8,236.33

CREATE A TABLE WITH BUCKETS THAT HAVE SORTED RECORDS

BUCKETS ARE SORTED BY ORDERDATE

ProductID	OrderDate	Revenue
2	January 17,2016	8,236.33
2	January 18,2016	7,455.67
1	January 18,2016	5,316.89
1	January 17,2016	2,433.76

THE PARTITION WITH STOREID =1

ProductID	OrderDate	Revenue
2	January 17,2016	8,236.33
2	January 18,2016	7,455.67

THIS IS BUCKET =2

ProductID	OrderDate	Revenue
1	January 17,2016	2,433.76
1	January 18,2016	5,316.89

THIS IS BUCKET =1

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS?

TO POPULATE THE BUCKETED TABLE ENTER THE FOLLOWING SET COMMAND

```
SET hive.enforce.bucketing = true;
```

THIS IS CALLED DYNAMIC BUCKETING

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS?

```
SET hive.enforce.bucketing = true;
```

THIS IS CALLED DYNAMIC BUCKETING

THIS IS NEEDED ONLY IF YOU ARE USING AN OLDER VERSION OF HIVE. IGNORE THIS SETTING FOR ANY HIVE VERSION AFTER 2.X

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS?

```
SET hive.enforce.bucketing = true;
```

THIS IS CALLED DYNAMIC BUCKETING

IT SETS THE NUMBER OF REDUCE TASKS
TO BE EQUAL TO THE NUMBER OF BUCKETS

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS?

LET'S ADD DATA TO THE REVIEWS TABLE

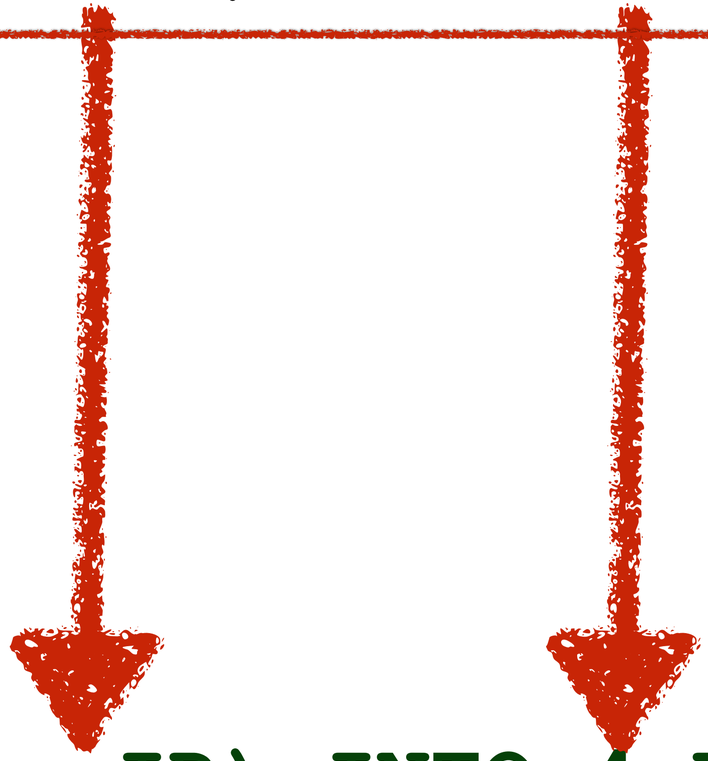
REVIEWS

REVIEWID	MovieID	Review	RatingStars
1	1	Amazing	5
2	1	Genre-Defining	5
3	1	Classic	5
4	1	Overrated	1
5	1	OK, Not Great	3
6	1	Two Thumbs Up	5
7	3	Crossover Hit	5
8	3	Love It	4
9	3	Nailbiting!	5
10	1	Cinematic Masterpiece	5
11	5	So Bad Its Good	0
12	2	Not As Good As Original	3
13	2	Overrated	2
14	2	Too Morbid	3
15	4	Sad, Thought-Provoking	4
16	4	Cinema At Its Best	5

WE BUCKETED IT ON REVIEWID INTO 4 BUCKETS

BY ADDING

```
Clustered by(column name) into N buckets
```



```
create table Reviews
(
ReviewID BIGINT,
MovieID INT,
Review VARCHAR(100),
Rating TINYINT
)
clustered by (ReviewID) INTO 4 Buckets;
```

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS?

```
insert into Reviews
values
(1,1,'Amazing',5),
(2,1,'Genre-Defining',5);
```

THE COMMAND IS THE SAME!

REVIEWS

REVIEWID	MovieID	Review	RatingStars
1	1	Amazing	5

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS?

```
insert into Reviews
values
(1,1,'Amazing',5),
(1,2,'Bad',5);
```

THE COMMAND IS THE SAME!

REVIEWS

REVIEWID	MovieID	Review	RatingStars
1	1	Amazing	5

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS AND PARTITIONS?

LET'S INSERT DATA INTO A TABLE THAT WE'RE FAMILIAR WITH

SALES DATA OF AN E-COMMERCE FRUIT SELLER

StoreID	ProductID	OrderDate	Revenue
1	1	January 18,2016	8,236.33
1	3	January 18,2016	7,455.67
1	4	January 18,2016	5,316.89
1	2	January 18,2016	2,433.76
2	1	January 18,2016	9,456.01
2	3	January 18,2016	3,644.33
2	4	January 18,2016	8,988.64
2	2	January 18,2016	1,621.58
1	1	January 17,2016	2342.33
1	3	January 17,2016	6345.10

THIS TABLE IS PARTITIONED AND BUCKETED

PARTITIONED ON STOREID

BUCKETED ON PRODUCTID INTO 4 BUCKETS

StoreID	ProductID	OrderDate	Revenue

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
Clustered by(PRODUCTID) into 4 buckets;
```

THIS TABLE IS PARTITIONED AND BUCKETED

PARTITIONED ON STOREID

BUCKETED ON PRODUCTID INTO 4 BUCKETS

StoreID	ProductID	OrderDate	Revenue

```
CREATE TABLE Sales_Data_NEW
(
PRODUCTID INT
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
)partitioned by (STOREID INT)
Clustered by(PRODUCTID) into 4 buckets;
```

THIS IS THE DESTINATION TABLE

THIS TABLE HAS NO PARTITIONS AND NO BUCKETS

StoreID	ProductID	OrderDate	Revenue

```
CREATE TABLE Sales_Data
(
STOREID INT,
PRODUCTID INT,
ORDERDATE DATE,
REVENUE DECIMAL(10,2)
);
```

THIS IS THE SOURCE TABLE

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS AND PARTITIONS?

TO INSERT DATA USE THE SAME COMMAND AS WHEN WE INSERTED DATA INTO A PARTITIONED TABLE

ENABLE DYNAMIC PARTITIONING

```
SET hive.exec.dynamic.partition = true;
SET hive.exec.dynamic.partition.mode = nonstrict;
```

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS AND PARTITIONS?

```
insert into Sales_Data_NEW
partition(STOREID)
SELECT PRODUCTID,ORDERDATE,REVENUE,STOREID
FROM Sales_Data;
```

THERE IS NO CHANGE IN INSERT COMMAND DUE TO BUCKETING!

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS AND PARTITIONS?

INDICATE WHAT THE PARTITION IS

```
insert into Sales_Data_NEW
partition(STOREID)
SELECT PRODUCTID,ORDERDATE,REVENUE,STOREID
FROM Sales_Data;
```

THERE IS NO CHANGE IN INSERT COMMAND DUE TO BUCKETING!

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS AND PARTITIONS?

```
insert into Sales_Data_NEW
partition(STOREID)
SELECT PRODUCTID,ORDERDATE,REVENUE,STOREID
FROM Sales_Data;
```

THERE IS NO CHANGE IN INSERT COMMAND DUE TO BUCKETING!

HOW DO WE PUT STUFF INTO TABLES WITH BUCKETS AND PARTITIONS?

THERE IS NO CHANGE IN INSERT COMMAND DUE TO BUCKETING!

WHEN YOU INSERT IN A TABLE THAT HAS SORTED COLUMNS, THERE IS ONCE AGAIN NO CHANGE IN THE INSERT COMMAND

HOW TO CHECK BUCKETED TABLES ON HDFS?

```
hadoop fs -ls  /user/hive/warehouse/reviews
```

TO SEE BUCKETS OF THE REVIEWS TABLE

```
16/05/20 21:52:04 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 4 items
-rwxrwxr-x   1 navdeepsingh supergroup         92 2016-05-18 12:10 /user/hive/warehouse/reviews/000000_0
-rwxrwxr-x   1 navdeepsingh supergroup         69 2016-05-18 12:10 /user/hive/warehouse/reviews/000001_0
-rwxrwxr-x   1 navdeepsingh supergroup         88 2016-05-18 12:10 /user/hive/warehouse/reviews/000002_0
-rwxrwxr-x   1 navdeepsingh supergroup         87 2016-05-18 12:10 /user/hive/warehouse/reviews/000003_0
```

HOW TO CHECK BUCKETED TABLES ON HDFS?

```
hadoop fs -ls
/user/hive/warehouse/Sales_Data_NEW/STOREID=1
```

TO SEE BUCKETS OF THE SALES_DATA_NEW TABLE

```
Navdeeps-MacBook-Pro:hadoop-2.7.2 navdeepsingh$ hadoop fs -ls /user/hive/warehouse/sales_data_new/storeid=1
16/05/23 06:30:50 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 4 items
-rwxr-xr-x   1 navdeepsingh supergroup         42 2016-05-23 06:26 /user/hive/warehouse/sales_data_new/storeid=1/000000_0
-rwxr-xr-x   1 navdeepsingh supergroup         42 2016-05-23 06:26 /user/hive/warehouse/sales_data_new/storeid=1/000001_0
-rwxr-xr-x   1 navdeepsingh supergroup         42 2016-05-23 06:26 /user/hive/warehouse/sales_data_new/storeid=1/000002_0
-rwxr-xr-x   1 navdeepsingh supergroup         42 2016-05-23 06:26 /user/hive/warehouse/sales_data_new/storeid=1/000003_0
Navdeeps-MacBook-Pro:hadoop-2.7.2 navdeepsingh$ hadoop fs -ls /user/hive/warehouse/sales_data_new/storeid=2
16/05/23 06:31:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 4 items
-rwxr-xr-x   1 navdeepsingh supergroup         42 2016-05-23 06:26 /user/hive/warehouse/sales_data_new/storeid=2/000000_0
-rwxr-xr-x   1 navdeepsingh supergroup         42 2016-05-23 06:26 /user/hive/warehouse/sales_data_new/storeid=2/000001_0
-rwxr-xr-x   1 navdeepsingh supergroup         42 2016-05-23 06:26 /user/hive/warehouse/sales_data_new/storeid=2/000002_0
-rwxr-xr-x   1 navdeepsingh supergroup         42 2016-05-23 06:26 /user/hive/warehouse/sales_data_new/storeid=2/000003_0
```

SAMPLING DATA FROM BUCKETED TABLES

SAMPLING DATA FROM BUCKETED TABLES

BY USING THE COMMAND

```
select * from table_name tablesample(bucket x out of y on column)
```

```
select * from reviews tablesample(bucket 4 out of 4 on ReviewId);
```

THE COLUMN ON WHICH BUCKETS HAVE BEEN CREATED

SAMPLING DATA FROM BUCKETED TABLES

BY USING THE COMMAND

```
select * from table_name tablesample(bucket x out of y on column)
```

```
select * from reviews tablesample(bucket 4 out of 4 on ReviewId);
```

THE TOTAL NUMBER OF BUCKETS

SAMPLING DATA FROM BUCKETED TABLES

BY USING THE COMMAND

```
select * from table_name tablesample(bucket x out of y on column)
```

```
select * from reviews tablesample(bucket 4 out of 4 on ReviewId);
```

THE BUCKET NUMBER FROM WHICH THE ROWS SHOULD BE SELECTED

SAMPLING DATA FROM BUCKETED TABLES

```
select * from reviews tablesample(bucket 4 out of 4 on ReviewId);
```

IT WILL OUTPUT ALL THE ROWS OF BUCKET 4

SAMPLING DATA FROM BUCKETED TABLES

```
select * from reviews tablesample(bucket 4 out of 4 on ReviewId);
```

```
hive> select * from reviews;
OK
4       16      Cinema At Its Best      5
2       12      Not Best As Good as Original    3
3       8       Love It 4
1       4       Overrated       1
3       9       Nailbiting!     5
1       5       OK, Not Great   3
1       1       Amazing 5
2       13      Overrated       2
1       10      Cinematic Masterpiece   5
1       2       Genre-Defining  5
1       6       Two Thumbs Up   5
2       14      Too Morbid      3
1       3       Classic 5
5       11      So Bad Its Good 0
3       7       Crossover Hit   5
4       15      Sad, Thought-Provoking  4
```

ORIGINAL TABLE

```
hive> select * from reviews tablesample(bucket 4 out of 4 on ReviewId);
OK
1       3       Classic 5
5       11      So Bad Its Good 0
3       7       Crossover Hit   5
4       15      Sad, Thought-Provoking  4
```

BUCKET 4

SAMPLING DATA FROM BUCKETED TABLES

```
select * from reviews tablesample(bucket 1 out of 2 on ReviewId);
```

WHAT IF WE SPECIFY THE NUMBER OF BUCKETS AS DIFFERENT FROM THE ORIGINAL NUMBER ?

REVIEWS HAD 4 BUCKETS, BUT WE TOLD TABLESAMPLE THAT THE TOTAL NUMBER OF BUCKETS IS 2

SAMPLING DATA FROM BUCKETED TABLES

```
select * from reviews tablesample(bucket 1 out of 2 on ReviewId);
```

NEW BUCKETS ARE CREATED , EACH MADE UP OF 2 OF THE ORIGINAL BUCKETS

THE OUTPUT WILL HAVE ALL THE ROWS FROM BUCKET 1 AND BUCKET 3

SAMPLING DATA FROM BUCKETED TABLES

```
select * from reviews tablesample(bucket 1 out of 8 on ReviewId);
```

REVIEWS HAD 4 BUCKETS, BUT WE TOLD TABLESAMPLE THAT THE TOTAL NUMBER OF BUCKETS IS 8

SAMPLING DATA FROM BUCKETED TABLES

```
select * from reviews tablesample(bucket 1 out of 8 on ReviewId);
```

NEW BUCKETS ARE CREATED , EACH MADE UP OF HALF OF THE ORIGINAL BUCKETS

THE OUTPUT WILL HAVE THE FIRST HALF OF THE ROWS FROM BUCKET 1

SAMPLING DATA FROM BUCKETED TABLES

BY USING THE SAME COMMAND

```
select * from reviews tablesample(bucket 1 out of 4 on MovieId);
```

WHAT IF WE SAMPLE ON MOVIE ID WHEN WE ORIGINALLY BUCKETED ON REVIEW ID

SAMPLING DATA FROM BUCKETED TABLES

BY USING THE SAME COMMAND

```
select * from reviews tablesample(bucket 1 out of 4 on MovieId);
```

BUCKETS WILL BE CREATED AT THE TIME OF QUERY ON MOVIEID

TABLESAMPLE COMMAND WILL SCAN THE ENTIRE TABLE TO CREATE THE BUCKETS

SAMPLING DATA FROM BUCKETED TABLES

BY USING THE SAME COMMAND

```
select * from reviews tablesample(bucket 1 out of 4 on MovieId);
```

4 BUCKETS WILL BE CREATED AND THE ROWS FROM BUCKET 1 WILL BE THE OUTPUT

SAMPLING DATA FROM NON-BUCKETED TABLES

SAMPLING DATA FROM NON-BUCKETED TABLES

THE SAME TABLESAMPLE COMMAND WORKS

THE COMMAND SAMPLES THE ENTIRE TABLE AND FETCHES A FEW ROWS

IN CASE OF BUCKETED TABLES, TABLESAMPLE SCANS ONLY THE REQUIRED HASH-PORTIONS OF THE DATA

SAMPLING DATA FROM NON-BUCKETED TABLES

THE SAME TABLESAMPLE COMMAND WORKS

THE COMMAND SCANS THE ENTIRE TABLE AND FILTERS OUT ROWS

IN CASE OF BUCKETED TABLES, TABLESAMPLE SCANS ONLY THE REQUIRED HASH-PORTIONS OF THE DATA

THIS IS INEFFICIENT AS COMPARED TO SAMPLING WHEN BUCKETS ARE ENABLED

BLOCK SAMPLING

BY USING THE COMMAND

```
select * from sales_data_new tablesample(4 PERCENT);
```

HIVE WILL SAMPLE 4% OF THE DATA IN THE TABLE

BLOCK SAMPLING

```
select * from sales_data_new tablesample(4 PERCENT);
```

```
hive> SELECT * FROM SALES_DATA_NEW;
OK
4       2016-01-17      5673.01 1
4       2016-01-18      5316.89 1
1       2016-01-17      2342.33 1
1       2016-01-18      8236.33 1
2       2016-01-17      4543.98 1
2       2016-01-18      2433.76 1
3       2016-01-17      6345.10 1
3       2016-01-18      7455.67 1
4       2016-01-17      5102.05 2
4       2016-01-18      8988.64 2
1       2016-01-17      8902.65 2
1       2016-01-18      9456.01 2
2       2016-01-17      1299.45 2
2       2016-01-18      1621.58 2
3       2016-01-17      9114.67 2
3       2016-01-18      3644.33 2
Time taken: 0.113 seconds, Fetched: 16 row(s)
hive> SELECT * FROM SALES_DATA_NEW TABLESAMPLE(4 PERCENT);
OK
4       2016-01-17      5673.01 1
4       2016-01-17      5102.05 2
Time taken: 0.142 seconds, Fetched: 2 row(s)
hive> SELECT * FROM SALES_DATA_NEW TABLESAMPLE(20 PERCENT);
OK
4       2016-01-17      5673.01 1
4       2016-01-18      5316.89 1
4       2016-01-17      5102.05 2
4       2016-01-18      8988.64 2
```

ROW COUNT SAMPLING

BY USING THE COMMAND

```
select * from sales_data_new tablesample(2 ROWS);
```

HIVE ALSO SUPPORTS SAMPLING A FIXED NUMBER OF ROWS

ROW COUNT SAMPLING

```
select * from sales_data_new tablesample(2 ROWS);
```

```
hive> SELECT * FROM SALES_DATA_NEW;
OK
4       2016-01-17      5673.01 1
4       2016-01-18      5316.89 1
1       2016-01-17      2342.33 1
1       2016-01-18      8236.33 1
2       2016-01-17      4543.98 1
2       2016-01-18      2433.76 1
3       2016-01-17      6345.10 1
3       2016-01-18      7455.67 1
4       2016-01-17      5102.05 2
4       2016-01-18      8988.64 2
1       2016-01-17      8902.65 2
1       2016-01-18      9456.01 2
2       2016-01-17      1299.45 2
2       2016-01-18      1621.58 2
3       2016-01-17      9114.67 2
3       2016-01-18      3644.33 2
Time taken: 0.099 seconds, Fetched: 16 row(s)
hive> SELECT * FROM SALES_DATA_NEW TABLESAMPLE(2 ROWS);
OK
4       2016-01-17      5673.01 1
4       2016-01-18      5316.89 1
```

HOW TO SAMPLE FROM BUCKETED TABLES?

LIMIT AND TABLESAMPLE COMMANDS ARE SIMILAR IN RESULTS

```
select * from SALES_DATA_NEW limit 4;
```

```
select * from SALES_DATA_NEW
tablesample(bucket 1 out of 4 on ProductID);
```

HOW TO SAMPLE FROM BUCKETED TABLES?

```
select * from SALES_DATA_NEW limit 4;
```

RETURNS 4 ROWS AFTER EXECUTING THE QUERY

```
select * from SALES_DATA_NEW
tablesample(bucket 1 out of 4 on ProductID);
```

RETURNS THE ROWS FROM A BUCKET WHICH CONTAINS SOME PRODUCTIDS

LIMIT AND TABLESAMPLE COMMANDS ARE SIMILAR IN RESULTS

BUT THERE ARE DIFFERENCES BETWEEN THE COMMANDS

LIMIT EXECUTES THE QUERY ON THE ENTIRE TABLE AND THEN RETURNS A LIMITED RESULT

TABLESAMPLE EXECUTES THE QUERY ON A LIMITED PORTION OF THE TABLE

LIMIT AND TABLESAMPLE COMMANDS ARE SIMILAR IN RESULTS

BUT THERE ARE DIFFERENCES BETWEEN THE COMMANDS

```
hive> SELECT * FROM SALES_DATA_NEW TABLESAMPLE(20 PERCENT);
OK
4       2016-01-17      5673.01 1
4       2016-01-18      5316.89 1
4       2016-01-17      5102.05 2
4       2016-01-18      8988.64 2
Time taken: 0.037 seconds, Fetched: 4 row(s)
hive> SELECT * FROM SALES_DATA_NEW LIMIT 4;
OK
4       2016-01-17      5673.01 1
4       2016-01-18      5316.89 1
1       2016-01-17      2342.33 1
1       2016-01-18      8236.33 1
Time taken: 0.123 seconds, Fetched: 4 row(s)
hive> SELECT * FROM SALES_DATA_NEW;
OK
4       2016-01-17      5673.01 1
4       2016-01-18      5316.89 1
1       2016-01-17      2342.33 1
1       2016-01-18      8236.33 1
2       2016-01-17      4543.98 1
2       2016-01-18      2433.76 1
3       2016-01-17      6345.10 1
3       2016-01-18      7455.67 1
4       2016-01-17      5102.05 2
4       2016-01-18      8988.64 2
1       2016-01-17      8902.65 2
1       2016-01-18      9456.01 2
2       2016-01-17      1299.45 2
2       2016-01-18      1621.58 2
3       2016-01-17      9114.67 2
3       2016-01-18      3644.33 2
```

LIMIT RETURNS THE FIRST 4 ROWS HERE