Bucketing Columns for Faster Joins

Overview

Understand how Hive data can be split using bucketing

Know the differences between partitioning and bucketing

Understand the advantages of using bucketed tables

Learn how to implement buckets in Hive

Learn how to sample data from Hive tables

Fast Lookup of Records

Fast Lookup

Requires us to know where exactly records are stored

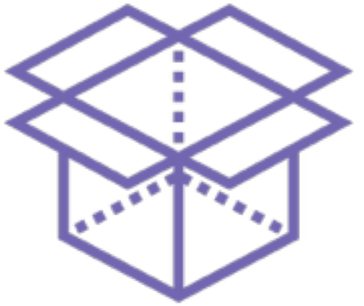
Hash Tables

Data structures which allow you to lookup specific values very quickly

Hash Tables

Have a fixed number of categories or buckets

Hash Tables

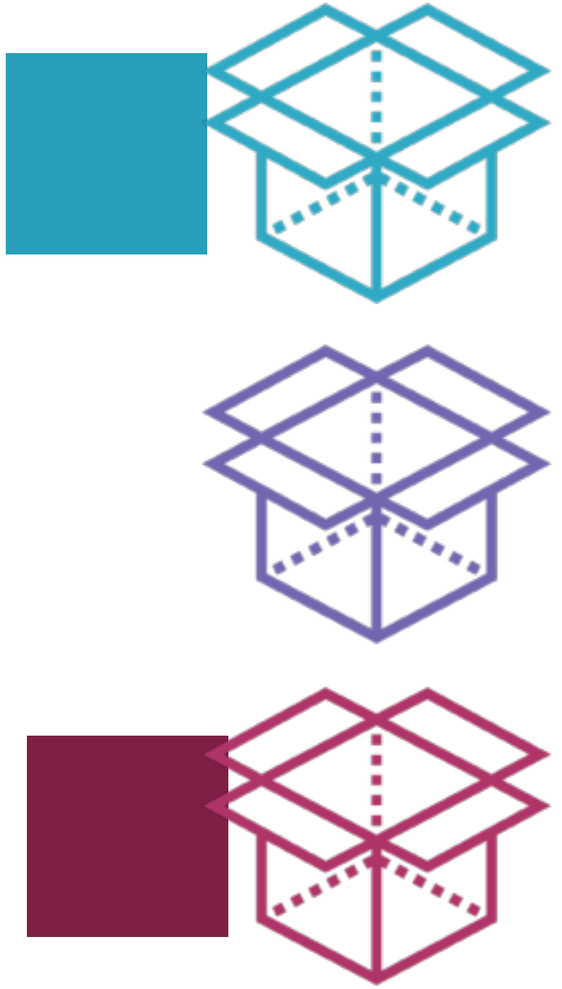


Accessing a bucket is fast

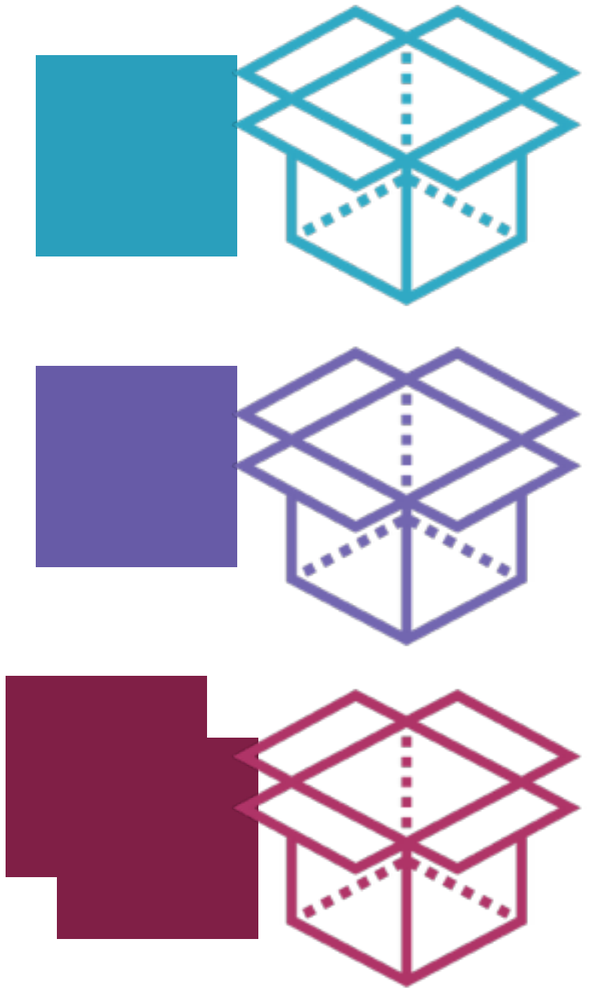
Hash Tables

A hash function determines which bucket each value belongs to

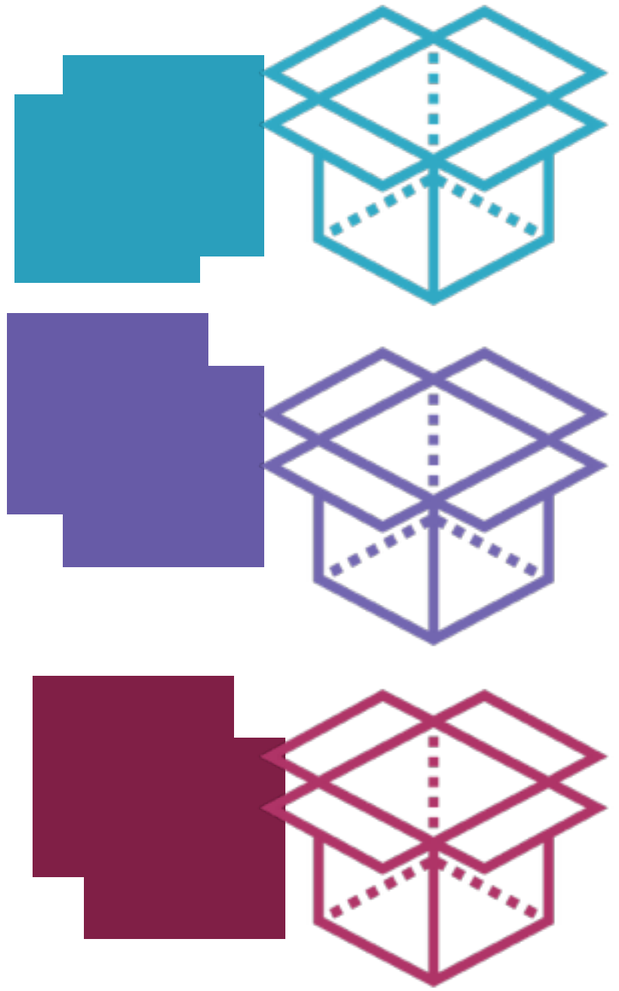
Hash Tables



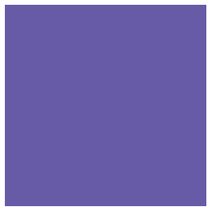
Hash Tables

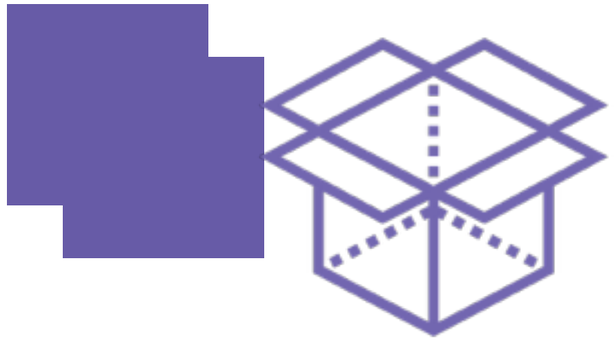


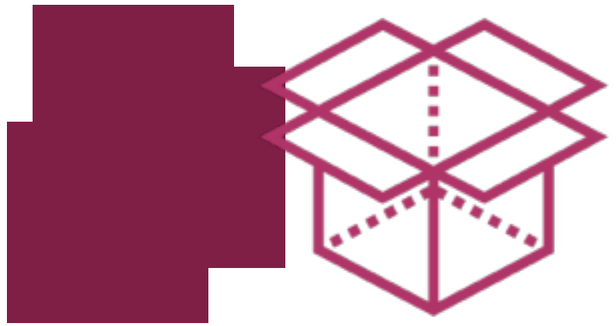
Hash Tables



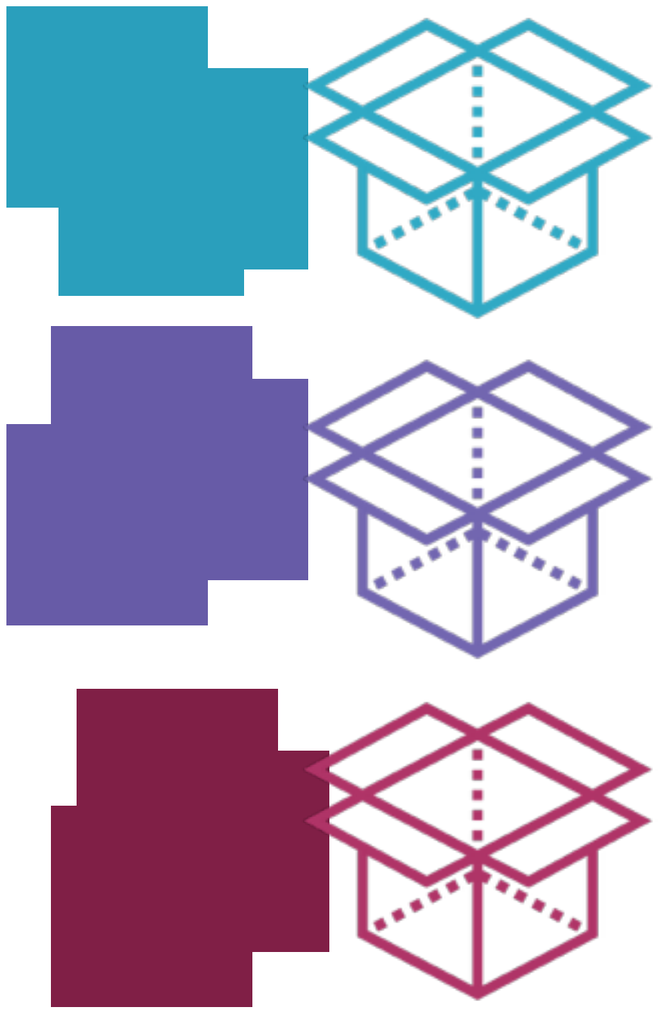
Hash Tables







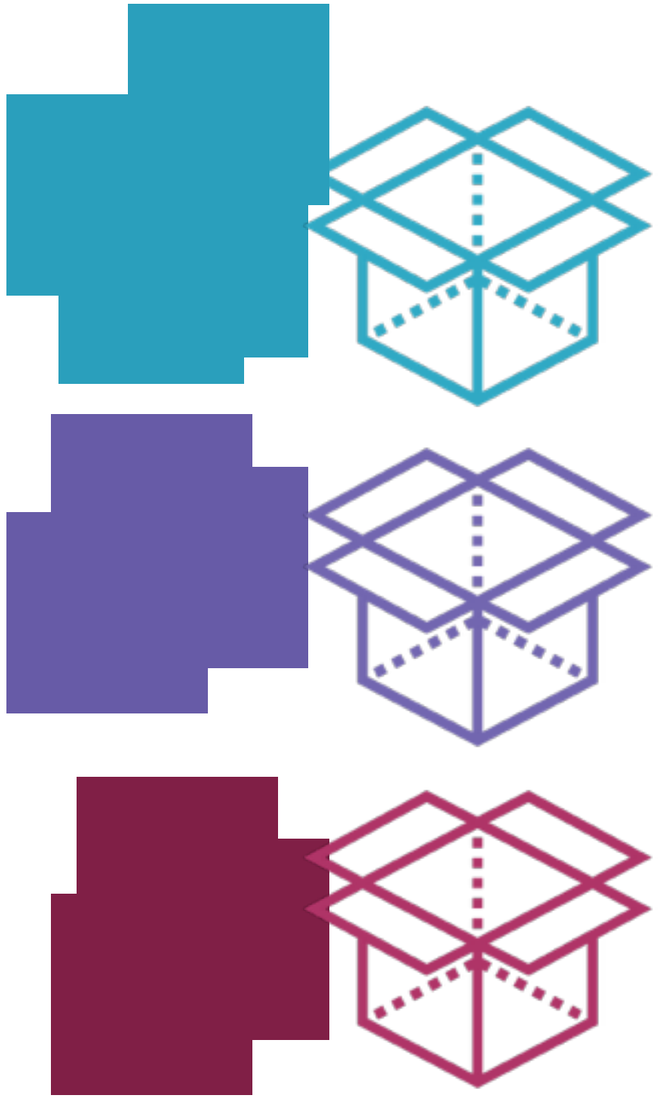
Hash Tables



Hash Tables

For any new value we know immediately which bucket it belongs to

Hash Tables



For any new value we know immediately which bucket it belongs to

Hash Tables

Each value is hashed so it falls in one of these buckets

Hash Tables

A value can only belong to one bucket and always belongs to the same bucket

Hashing ~ Bucketing

Bucketing is the logical equivalent of hashing in Hive

Hashing ~ Bucketing

Buckets are files on the storage system which store those records which map to it

Hashing ~ Bucketing

Partitions are directories, buckets are files under those directories

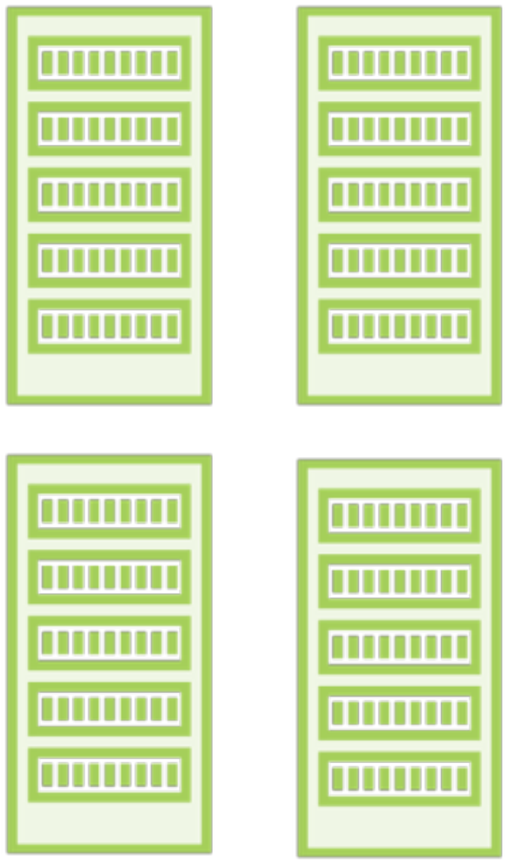
Bucketing Tables in Hive

Bucketing

Split data into smaller subsets

Separate records into manageable parts based on a column value

Bucketing



Bucketing

Records are split across multiple machines in the cluster

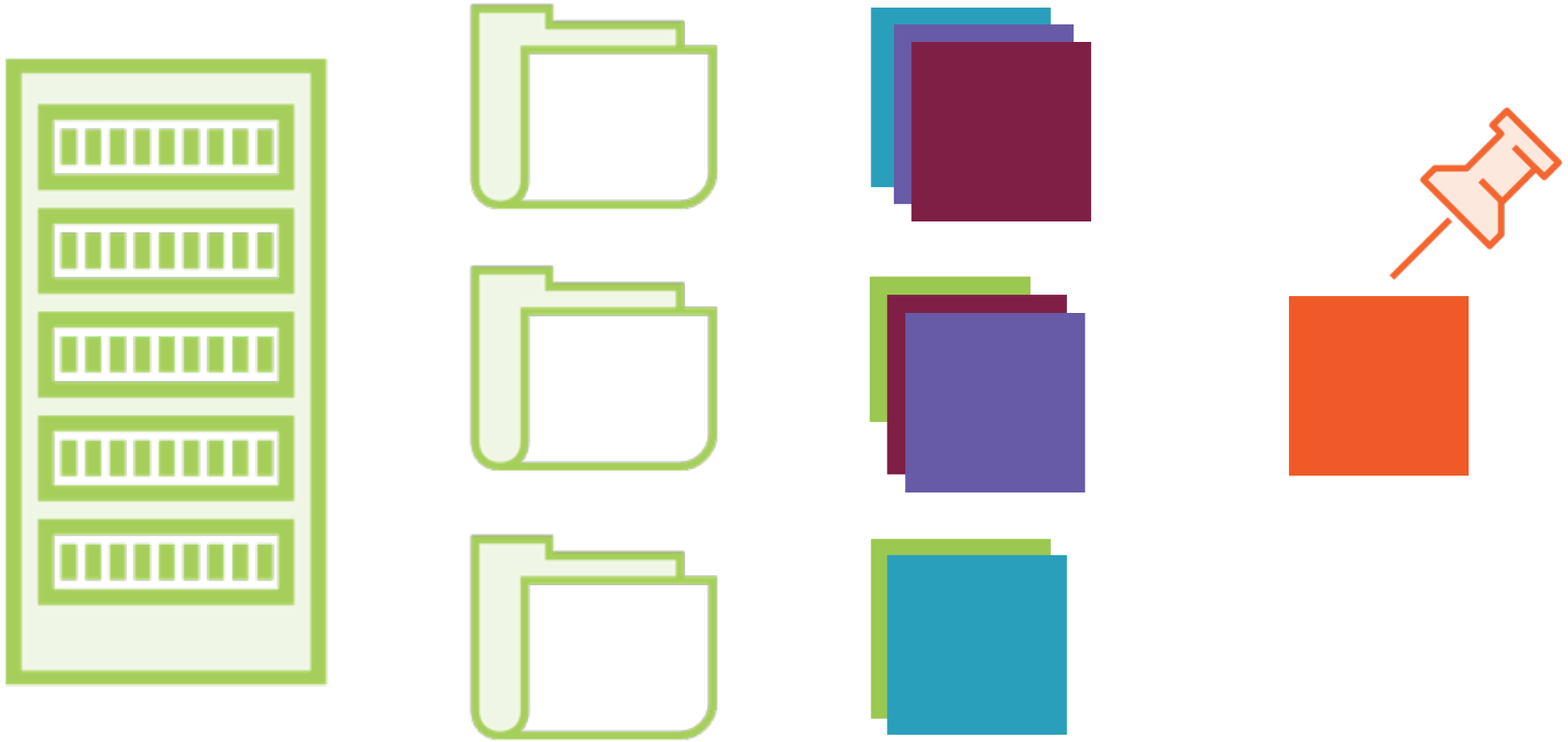
Bucketing

Each machine has directories for databases, tables and partitions

Bucketing

And files which store the actual data in each directory

Bucketing



Knowing which file holds the record can make lookup and join operations very fast

Bucketing

A hash function determines which bucket each value belongs to

Bucketing

A common hash function for integer column values is the modulo (%) operator

Bucketing

Consider an e-commerce site which sells millions of products

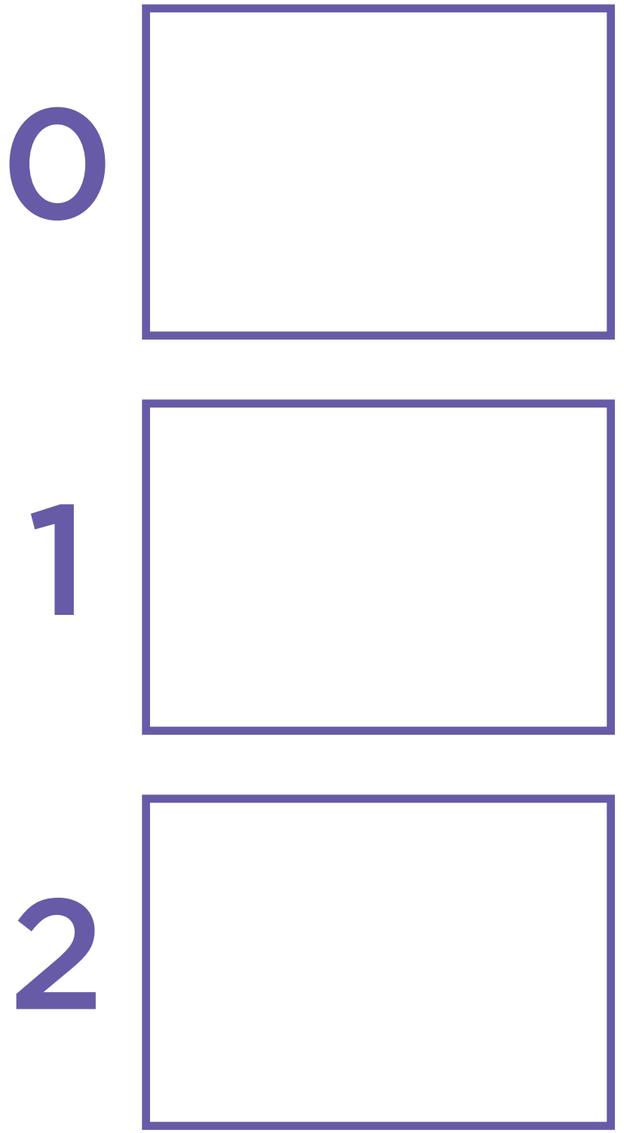
Bucketing

ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books
8	Belt	20	Fashion

Products

Bucketing

ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books
8	Belt	20	Fashion

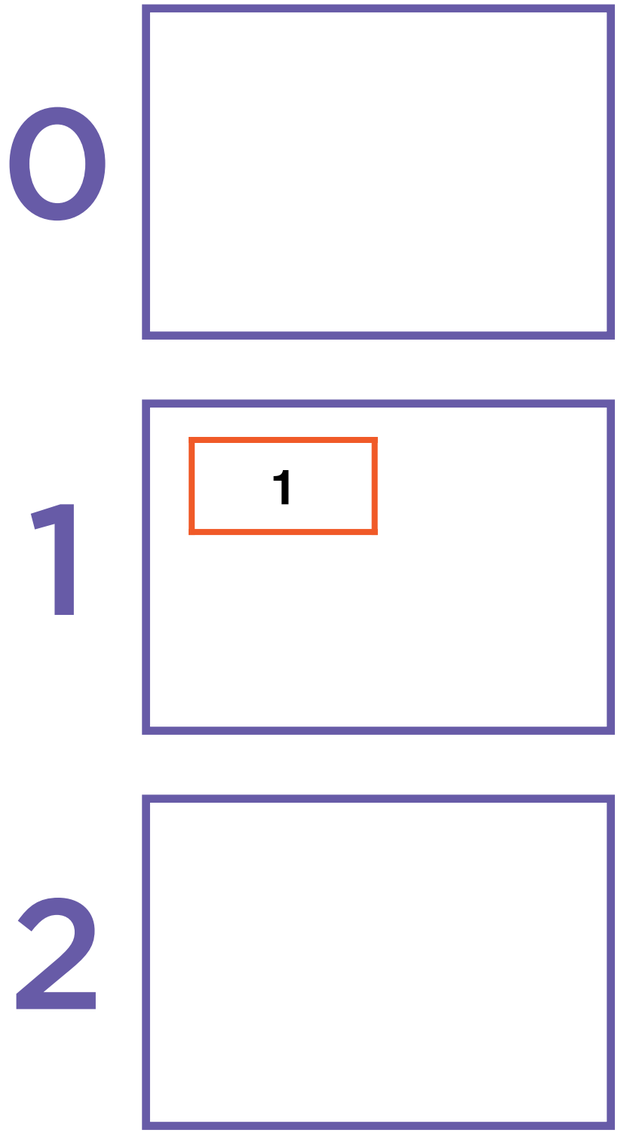


bucketed on the ID column

Number of buckets = 3

Bucketing

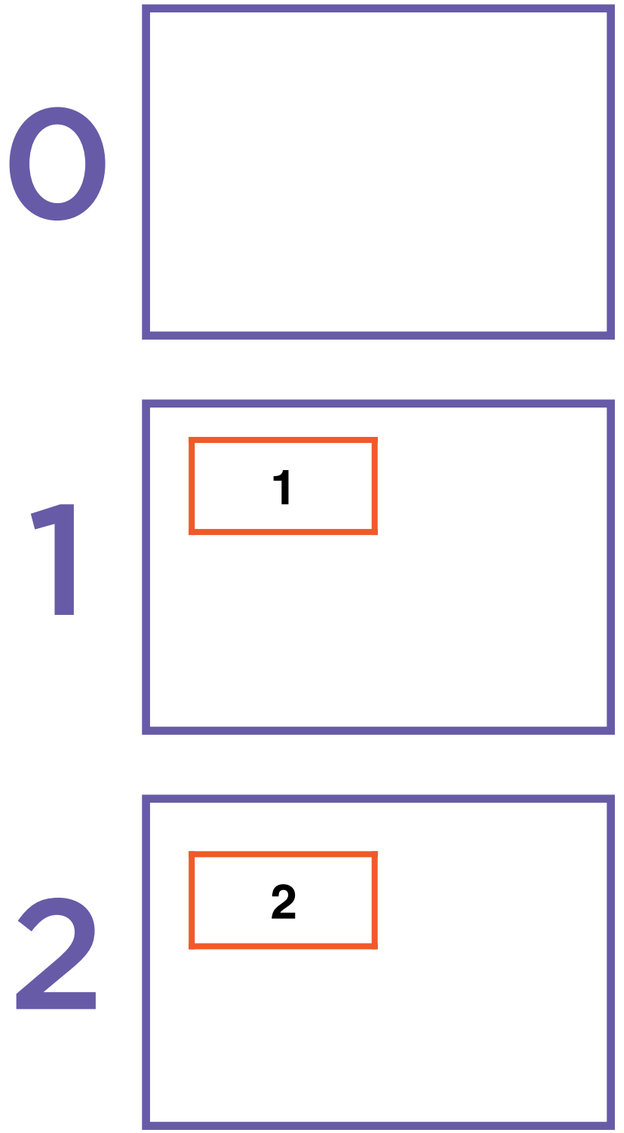
ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books
8	Belt	20	Fashion



1 % 3 = 1

Bucketing

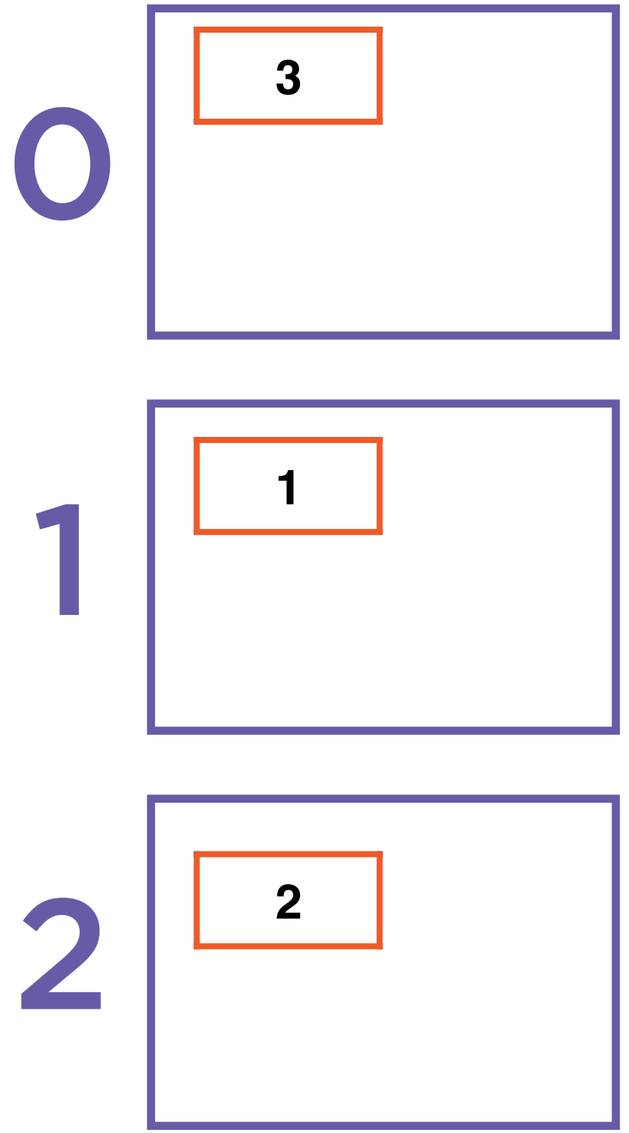
ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books
8	Belt	20	Fashion



2 % 3 = 2

Bucketing

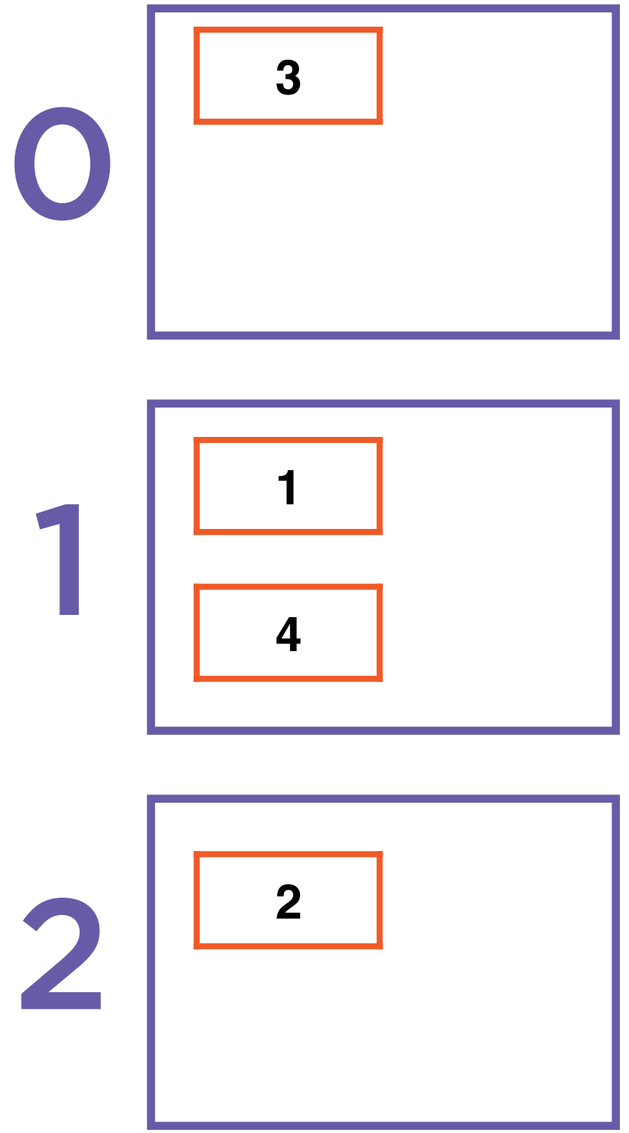
ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books
8	Belt	20	Fashion



3 % 3 = 0

Bucketing

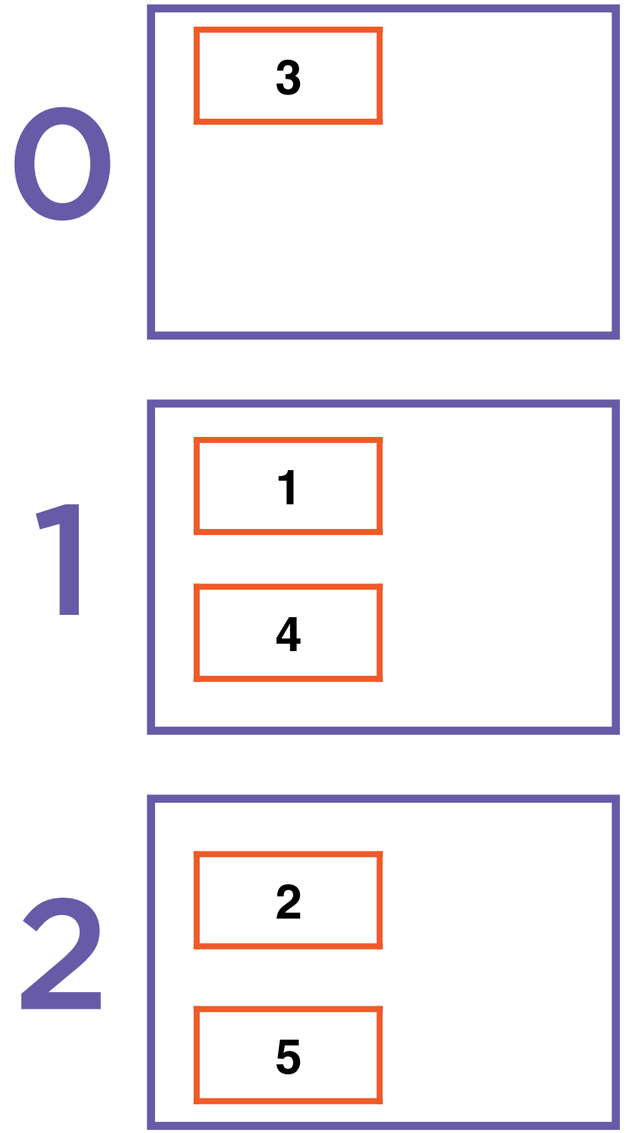
ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books
8	Belt	20	Fashion



4 % 3 = 1

Bucketing

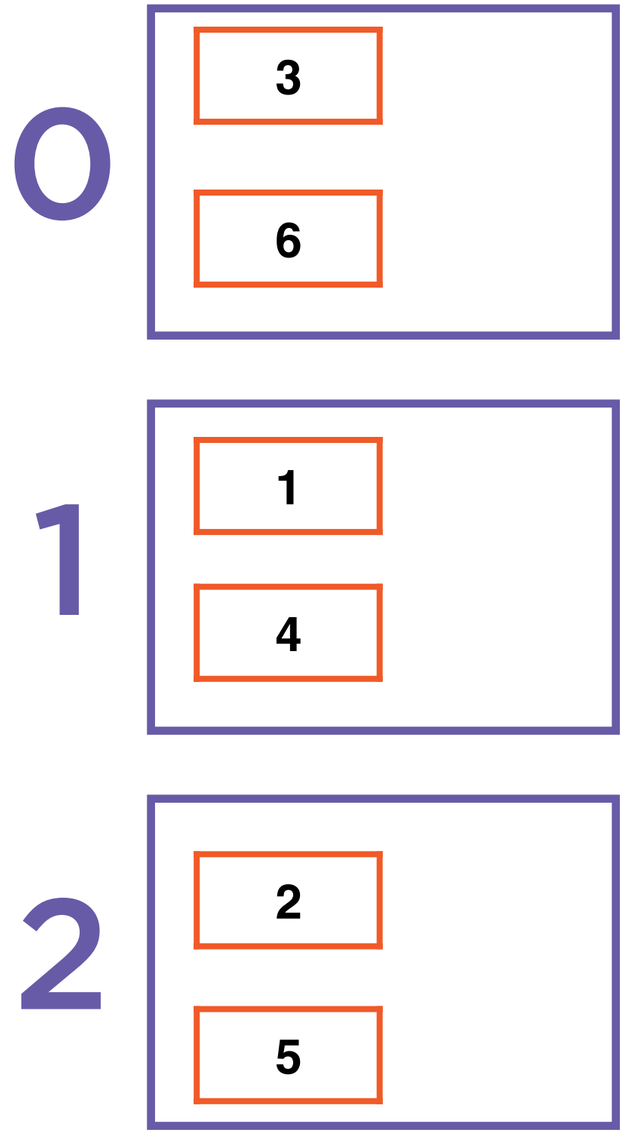
ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books
8	Belt	20	Fashion



5 % 3 = 2

Bucketing

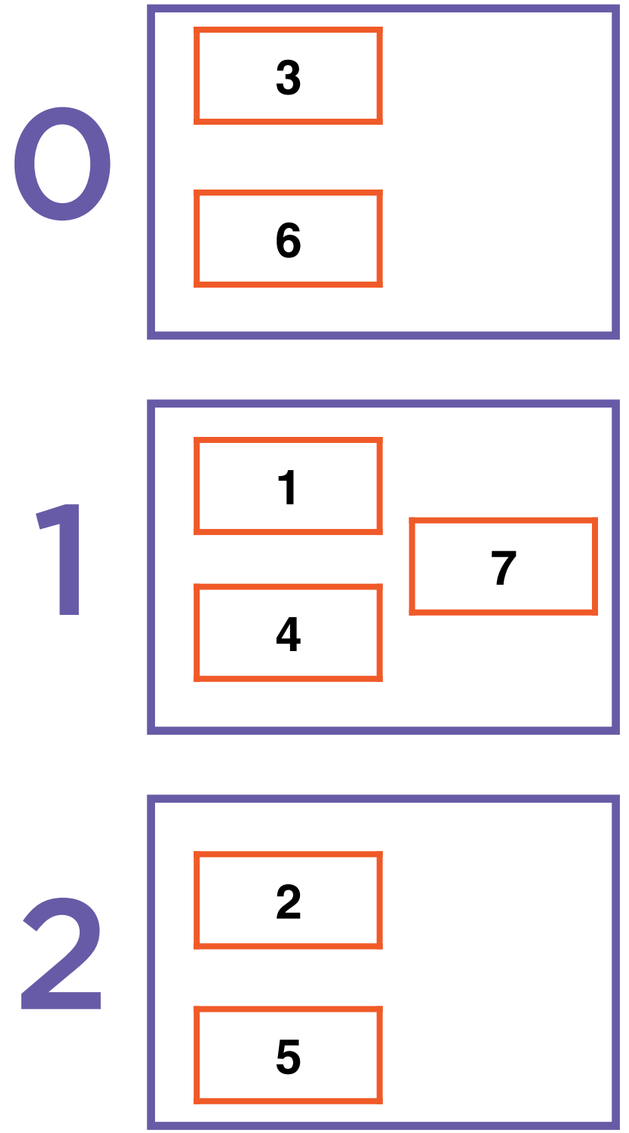
ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books
8	Belt	20	Fashion



6 % 3 = 0

Bucketing

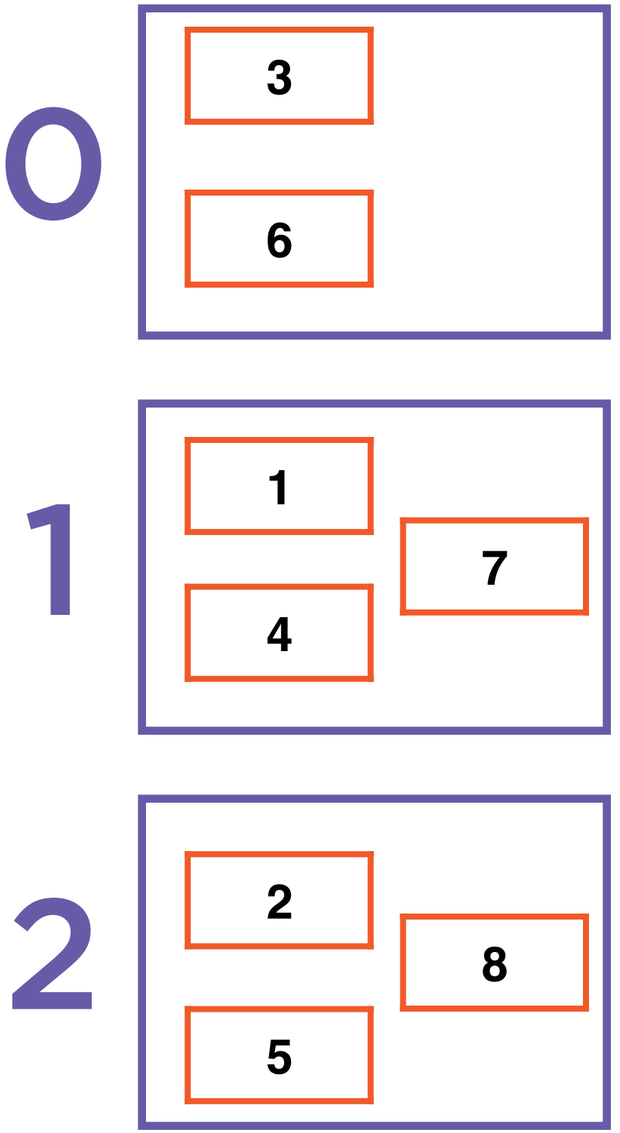
ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books
8	Belt	20	Fashion



7 % 3 = 1

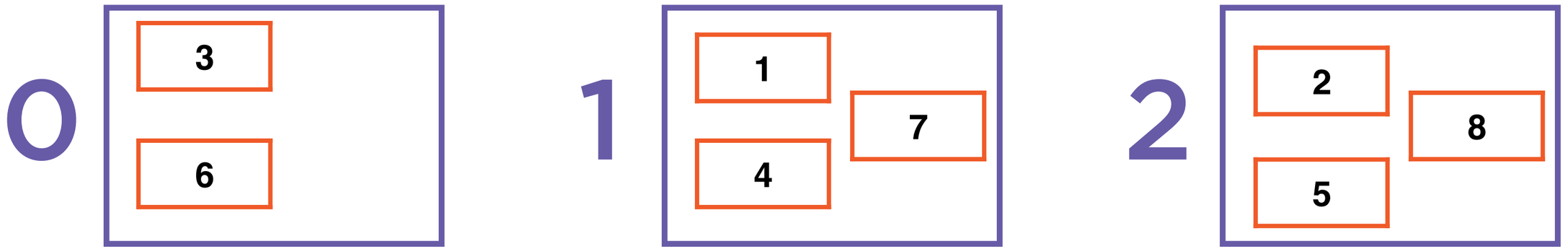
Bucketing

ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books
8	Belt	20	Fashion



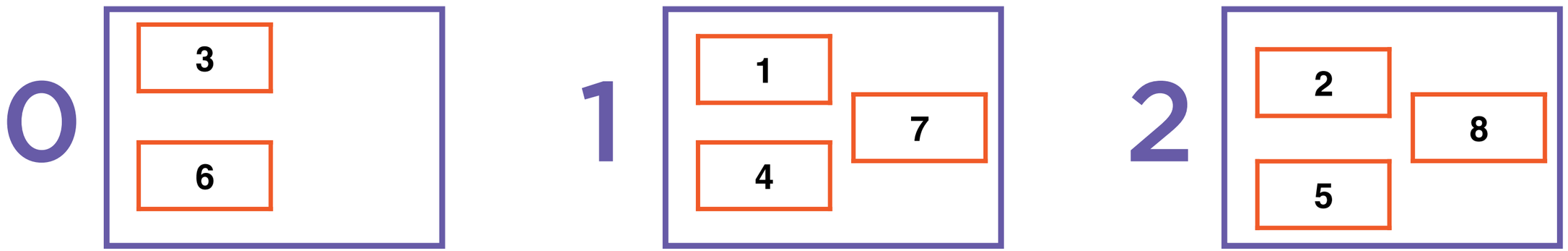
8 % 3 = 2

Bucketing



Each of these buckets are files on HDFS

Bucketing



Knowing which file each record is stored in makes certain operations very fast

Demo

Bucketing a non-partitioned table

Loading data into and querying a bucketed table

Layout of a bucketed table

Bucketing vs. Partitioning

Partitioning

ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books

Partitioning on a column such as product id can result in millions of directories

Partitioning

ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books

Hive restricts the maximum number of partitions allowed to avoid overwhelming the NameNode

Partitioning

ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books

Query optimization on this column requires bucketing

Partitioning

ID	Name	Cost	Category
1	iPhone	599	Mobiles
2	Doll	35	Toys
3	Shoes	33	Footwear
4	Jeans	69	Fashion
5	Skates	123	Sports
6	Make Up	99	Fashion
7	Book	24	Books

Fixed number of buckets and an easy way to know which bucket holds a record

Bucketing vs. Partitioning

Bucketing	Partitioning
Fixed number of buckets	Unknown number of partitions
Based on a hash value of a column	Based on actual column values
Each bucket stored as files under a directory	Each partition stored as a directory
Buckets are almost the same size	Partitions can vary a lot in size
Optimizes queries such as lookup, joins, sampling	Optimizes queries which retrieve or scan data in the logical group

Partitioned and Bucketed Tables

Hive tables can be both partitioned and bucketed

Partitioned and Bucketed Tables

Each partitioned directory holds a file for each bucket

Demo

Bucketing a partitioned table

Loading data into and querying a partitioned, bucketed table

Layout of a partitioned, bucketed table

Advantages of Bucketing

Advantages of Bucketing

Faster query responses

Faster join operations

- Map side joins
- Sorted records in buckets

More efficient sampling to test and debug operations

Advantages of Bucketing

Faster query responses

Faster join operations

- Map side joins
- Sorted records in buckets

More efficient sampling to test and debug operations

Faster Query Responses

Given a record, it is possible to figure out where exactly the record is stored

Advantages of Bucketing

Faster query responses

Faster join operations

- Map side joins
- Sorted records in buckets

More efficient sampling to test and debug operations

Faster Join Operations

Joins combine values from 2 or more tables on the same column values

Faster Join Operations

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
7	iPhone	599
8	Doll	35
3	Shoes	33
1	Jeans	69
6	Skates	123
5	Make Up	99
4	Book	24
2	Belt	20

Join Orders and Products to get the names of the products users have bought

Faster Join Operations

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
7	iPhone	599
8	Doll	35
3	Shoes	33
1	Jeans	69
6	Skates	123
5	Make Up	99
4	Book	24
2	Belt	20

Product ID is the join column

Faster Join Operations

Orders

ID	Product ID	Quantity	Amount
o1	**4**	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
7	iPhone	599
8	Doll	35
3	Shoes	33
1	Jeans	69
6	Skates	123
5	Make Up	99
4	**Book**	**24**
2	Belt	20

Need to scan the entire dataset to find the corresponding row

Faster Join Operations

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
6	Skates	123
3	Shoes	33

ID	Name	Cost
7	iPhone	599
1	Jeans	69
4	Book	24

ID	Name	Cost
2	Belt	20
8	Doll	35
5	Make Up	99

Bucket the Products table on the ID column

Faster Join Operations

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
6	Skates	123
3	Shoes	33

ID	Name	Cost
7	iPhone	599
1	Jeans	69
4	**Book**	**24**

ID	Name	Cost
2	Belt	20
8	Doll	35
5	Make Up	99

Scan a much smaller dataset to access each row

Faster Join Operations

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
6	Skates	123
3	Shoes	33

ID	Name	Cost
7	iPhone	599
1	Jeans	69
4	Book	24

ID	Name	Cost
2	Belt	20
8	Doll	35
5	Make Up	99

Faster joins

Advantages of Bucketing

Faster query responses

Faster join operations

- Map side joins
- Sorted records in buckets

More efficient sampling to test and debug operations

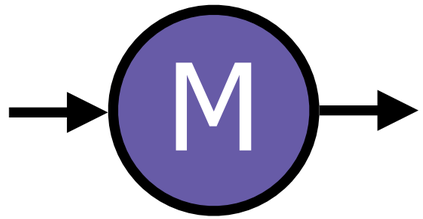
Joins as Map-only Operations

MapReduce operations have 2 phases of processing

Joins as Map-only Operations

Certain queries can be structured to have no reduce phase

Faster Map-only Joins



Map-only joins load one or more of the join tables in memory

Faster Map-only Joins

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
6	Skates	123
3	Shoes	33

ID	Name	Cost
7	iPhone	599
1	Jeans	69
4	Book	24

ID	Name	Cost
2	Belt	20
8	Doll	35
5	Make Up	99

Mapper knows the exact bucket where the row is present

Faster Map-only Joins

Orders

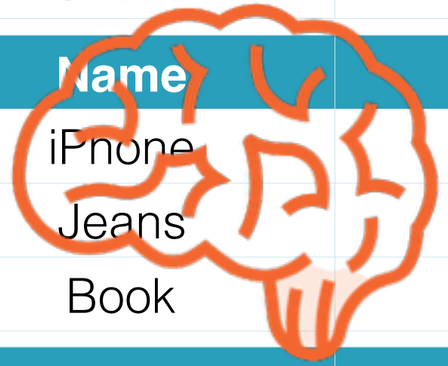
ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
6	Skates	123
3	Shoes	33

ID	Name	Cost
7	iPhone	599
1	Jeans	69
4	Book	24

ID	Name	Cost
2	Belt	20
8	Doll	35
5	Make Up	99



Load only one bucket in memory

Advantages of Bucketing

Faster query responses

Faster join operations

- Map side joins
- Sorted records in buckets

More efficient sampling to test and debug operations

Sorted Records for Faster Joins

Records in each bucket can be sorted

Sorted Records for Faster Joins

Orders

ID	Product ID	Quantity	Amount
o1	4	1	599
o2	7	1	35
o3	8	1	33
o4	5	2	69
o5	1	1	123
o6	6	1	99
o7	2	2	24
o8	3	2	20

Products

ID	Name	Cost
6	Skates	123
3	Shoes	33

ID	Name	Cost
7	iPhone	599
1	Jeans	69
4	Book	24

ID	Name	Cost
2	Belt	20
8	Doll	35
5	Make Up	99

Bucket both tables on the product ID column

Sorted Records for Faster Joins

Orders

ID	Product ID	Quantity	Amount
o6	6	1	99
o8	3	2	20

ID	Product ID	Quantity	Amount
o5	1	1	123
o2	7	1	35
o1	4	1	599

ID	Product ID	Quantity	Amount
o3	8	1	33
o7	2	2	24
o4	5	2	69

Products

ID	Name	Cost
6	Skates	123
3	Shoes	33

ID	Name	Cost
7	iPhone	599
1	Jeans	69
4	Book	24

ID	Name	Cost
2	Belt	20
8	Doll	35
5	Make Up	99

Sort records in each bucket on product ID

Sorted Records for Faster Joins

Orders

ID	Product ID	Quantity	Amount
o8	3	2	20
o6	6	1	99

ID	Product ID	Quantity	Amount
o5	1	1	123
o1	4	1	599
o2	7	1	35

ID	Product ID	Quantity	Amount
o7	2	2	24
o4	5	2	69
o3	8	1	33

Products

ID	Name	Cost
3	Shoes	33
6	Skates	123

ID	Name	Cost
1	Jeans	69
4	Book	24
7	iPhone	599

ID	Name	Cost
2	Belt	20
5	Make Up	99
8	Doll	35

Join operations work like merge-sort

Sorted Records for Faster Joins

Orders

ID	Product ID	Quantity	Amount
o8	3	2	20
o6	6	1	99

ID	Product ID	Quantity	Amount
o5	1	1	123
o1	4	1	599
o2	7	1	35

ID	Product ID	Quantity	Amount
o7	2	2	24
o4	5	2	69
o3	8	1	33

Products

ID	Name	Cost
3	Shoes	33
6	Skates	123

ID	Name	Cost
1	Jeans	69
4	Book	24
7	iPhone	599

ID	Name	Cost
2	Belt	20
5	Make Up	99
8	Doll	35

Faster joins

Demo

Bucketed tables with sorted records

Advantages of Bucketing

Faster query responses

Faster join operations

- Map side joins
- Sorted records in buckets

More efficient sampling to test and debug operations

Sampling Hive Table Data

Sampling

Get a smaller, representative subset of data from the table

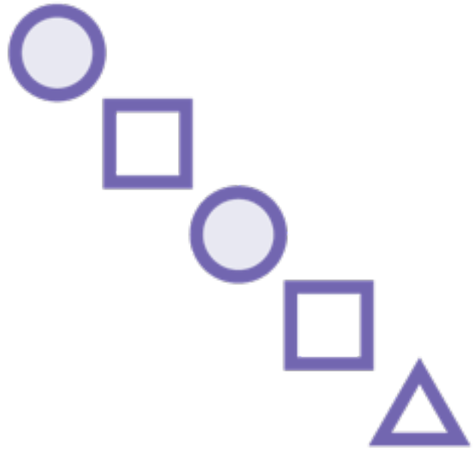
Sampling

Used for testing, aggregations, debugging

Sampling

Limit

Block sampling



Row count sampling

Bucket sampling

Limit

Executes the query on the entire table

Returns only a limited set of results to the user

The first N rows are retrieved

Not a random, representative subset

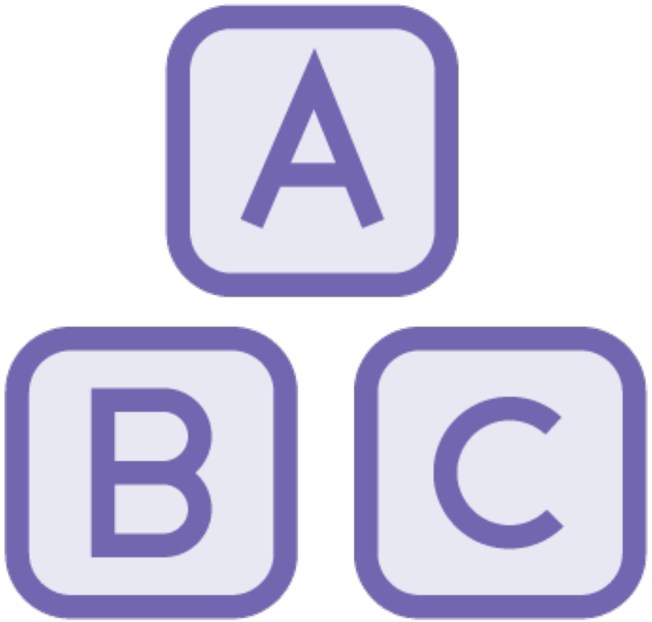
```
select * from products limit 4;
```

Using the Limit Keyword

Run a select on the entire table but only return 4 records to the user

Can be run on all tables, bucketed or non-bucketed

Block Sampling



Samples the entire table

Returns only a few rows to the user

A percentage of the dataset retrieved

Inefficient way to sample data

```
select * from products_no_buckets tablesample(10 percent);
```

Block Sampling, Non-bucketed Tables

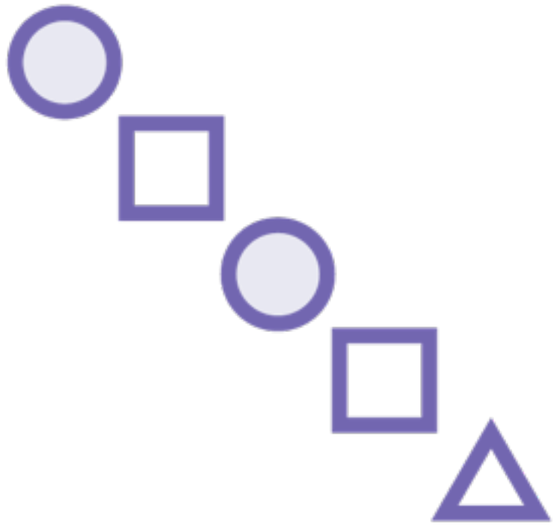
Sample the entire table and return roughly 10% of the rows

```
select * from products tablesample(10 percent);
```

Block Sampling, Bucketed Tables

Sample the entire table and return roughly 10% of the rows

Works on both non-buckets and bucketed tables



Row Count Sampling

Samples the entire table

Returns only a few rows to the user

A fixed number of rows retrieved

Inefficient way to sample data

```
select * from products_no_buckets tablesample(3 rows);
```

Row Count Sampling, Non-bucketed Tables

Sample the entire table and return exactly 3 rows

```
select * from products tablesample(3 rows);
```

Row Count Sampling, Bucketed Tables

Sample the entire table and return exactly 3 rows

Works on both non-buckets and bucketed tables

Bucket Sampling

Samples only a few buckets in a table

Returns only a few rows to the user

A limited number of rows retrieved based on bucket sizes

Much more efficient way to sample data

Bucket Sampling

Samples only a few buckets in a table

Returns only a few rows to the user

A limited number of rows retrieved based on bucket sizes

Works only on bucketed tables

Demo

Sample data in Hive tables using bucket sampling

Summary

Understood how bucketing works and the advantages of bucketing in Hive

Understood the differences between partitioning and bucketing

Learnt how to implement buckets in Hive

Learnt how to sample data from Hive tables