Postgres Runs:

Thumb Rule-1Runs:

Creating table, adding values from CSVs, Creating Primary keys with cluster indexing:

```
advdb=# CREATE TABLE trades_uniform (
    stock_symbol VARCHAR(10),
    time BIGINT,
    quantity INT,
    price NUMERIC(10, 2)
);
CREATE TABLE
advdb=# CREATE TABLE trades_fractal (
    stock_symbol VARCHAR(10),
    time BIGINT,
    quantity INT,
    price NUMERIC(10, 2)
);
CREATE TABLE
advdb=# COPY trades_uniform (stock_symbol, time, quantity, price)
FROM '/home/sc10670/ADB/advDB-HW2/Q2/uniform_trades.csv'
DELIMITER ','
CSV HEADER;
COPY 10000000
advdb=# COPY trades_fractal (stock_symbol, time, quantity, price)
FROM '/home/sc10670/ADB/advDB-HW2/Q2/fractal_trades.csv'
DELIMITER ','
CSV HEADER;
COPY 10000000
advdb=# ALTER TABLE trades_uniform ADD PRIMARY KEY (stock_symbol, time);
ALTER TABLE
advdb=# ALTER TABLE trades_fractal ADD PRIMARY KEY (stock_symbol, time);
ALTER TABLE
advdb=# CLUSTER trades_uniform USING trades_uniform_pkey;
CLUSTER
advdb=# CLUSTER trades_fractal USING trades_fractal_pkey;
CLUSTER
```

Running Query on Uniform Dataset with Clustered Indexing:

```
advdb=# EXPLAIN ANALYZE
SELECT stock_symbol, time, quantity, price,
       SUM(quantity) AS total_quantity,
       AVG(price) AS avg_price
FROM trades_uniform
WHERE time BETWEEN 100000 AND 800000
  AND price > 100
  AND quantity BETWEEN 500 AND 8000
GROUP BY stock_symbol, time, quantity, price
ORDER BY total_quantity DESC, avg_price ASC;
                                                                                                                  QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------
 Sort  (cost=316714.13..317923.64 rows=483803 width=63) (actual time=10732.557..12411.067 rows=470037 loops=1)
   Sort Key: (sum(quantity)) DESC, (avg(price))
   Sort Method: external merge  Disk: 25368kB
   ->  Finalize GroupAggregate  (cost=190177.70..252840.31 rows=483803 width=63) (actual time=1702.504..8926.413 rows
=470037 loops=1)
         Group Key: stock_symbol, "time"
         ->  Gather Merge  (cost=190177.70..241753.15 rows=403170 width=63) (actual time=1702.430..5439.377 rows=4700
37 loops=1)
               Workers Planned: 2
               Workers Launched: 2
               ->  Partial GroupAggregate  (cost=189177.68..194217.30 rows=201585 width=63) (actual time=1665.395..32
97.630 rows=156679 loops=3)
                     Group Key: stock_symbol, "time"
                     ->  Sort  (cost=189177.68..189681.64 rows=201585 width=23) (actual time=1665.329..2183.321 rows=
156679 loops=3)
                           Sort Key: stock_symbol, "time"
                           Sort Method: external merge  Disk: 5536kB
                           Worker 0:  Sort Method: external merge  Disk: 5288kB
                           Worker 1:  Sort Method: external merge  Disk: 5312kB
                           ->  Parallel Seq Scan on trades_uniform  (cost=0.00..167280.00 rows=201585 width=23) (actu
al time=0.435..1191.903 rows=156679 loops=3)
                                 Filter: (("time" >= 100000) AND ("time" <= 800000) AND (price > '100'::numeric) AND
(quantity >= 500) AND (quantity <= 8000))
                                 Rows Removed by Filter: 3176654
 Planning Time: 10.149 ms
 Execution Time: 13547.375 ms
(20 rows)
```

Running Query on Fractal Dataset with Clustered Indexing:

```
                                                                                      QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=309575.74..310728.91 rows=461267 width=63) (actual time=10715.727..12529.905 rows=468152 loops=1)
   Sort Key: (sum(quantity)) DESC, (avg(price))
   Sort Method: external merge  Disk: 25264kB
   ->  Finalize GroupAggregate  (cost=189091.76..248835.49 rows=461267 width=63) (actual time=1703.825..8901.576 rows
=468152 loops=1)
         Group Key: stock_symbol, "time"
         ->  Gather Merge  (cost=189091.76..238264.78 rows=384390 width=63) (actual time=1703.760..5446.401 rows=4681
52 loops=1)
               Workers Planned: 2
               Workers Launched: 2
               ->  Partial GroupAggregate  (cost=188091.74..192896.61 rows=192195 width=63) (actual time=1689.259..33
21.099 rows=156051 loops=3)
                     Group Key: stock_symbol, "time"
                     ->  Sort  (cost=188091.74..188572.22 rows=192195 width=23) (actual time=1689.166..2210.687 rows=
156051 loops=3)
                           Sort Key: stock_symbol, "time"
                           Sort Method: external merge  Disk: 5600kB
                           Worker 0:  Sort Method: external merge  Disk: 5376kB
                           Worker 1:  Sort Method: external merge  Disk: 5104kB
                           ->  Parallel Seq Scan on trades_fractal  (cost=0.00..167280.00 rows=192195 width=23) (actu
al time=0.343..1221.364 rows=156051 loops=3)
                                 Filter: (("time" >= 100000) AND ("time" <= 800000) AND (price > '100'::numeric) AND
(quantity >= 500) AND (quantity <= 8000))
                                 Rows Removed by Filter: 3177283
 Planning Time: 3.432 ms
 Execution Time: 13503.324 ms
(20 rows)
```

Running the Query to get output:

```
advdb=# SELECT stock_symbol, time, quantity, price,
       SUM(quantity) AS total_quantity,
       AVG(price) AS avg_price
FROM trades_uniform
WHERE time BETWEEN 100000 AND 800000
  AND price > 100
  AND quantity BETWEEN 500 AND 8000
GROUP BY stock_symbol, time, quantity, price
ORDER BY total_quantity DESC, avg_price ASC;
```

stock_symbol	time	quantity	price	total_quantity	avg_price
s44195	796008	8000	111.00	8000	111.0000000000000000
s68886	579948	8000	119.00	8000	119.0000000000000000
s48222	237046	8000	124.00	8000	124.0000000000000000
s39386	396811	8000	129.00	8000	129.0000000000000000
s26148	112051	8000	133.00	8000	133.0000000000000000
s66240	313299	8000	154.00	8000	154.0000000000000000
s58062	474786	8000	172.00	8000	172.0000000000000000
s6459	464132	8000	173.00	8000	173.0000000000000000
s55302	458803	8000	173.00	8000	173.0000000000000000
s46889	656298	8000	177.00	8000	177.0000000000000000
s12420	643729	8000	180.00	8000	180.0000000000000000
s33561	220169	8000	185.00	8000	185.0000000000000000
s27213	437222	8000	189.00	8000	189.0000000000000000
s29541	481724	8000	209.00	8000	209.0000000000000000
s4534	756678	8000	209.00	8000	209.0000000000000000
s20057	731377	8000	212.00	8000	212.0000000000000000
s30353	145200	8000	219.00	8000	219.0000000000000000
s239	418904	8000	225.00	8000	225.0000000000000000
s13282	491977	8000	227.00	8000	227.0000000000000000
s42102	794912	8000	230.00	8000	230.0000000000000000
s52457	272736	8000	232.00	8000	232.0000000000000000
s2567	679497	8000	239.00	8000	239.0000000000000000
s463	484194	8000	251.00	8000	251.0000000000000000
s64078	212335	8000	256.00	8000	256.0000000000000000
s26076	442859	8000	257.00	8000	257.0000000000000000
s50418	319715	8000	263.00	8000	263.0000000000000000
s41638	662383	8000	265.00	8000	265.0000000000000000
s36124	504909	8000	273.00	8000	273.0000000000000000
s3316	214929	8000	286.00	8000	286.0000000000000000
s40940	251435	8000	287.00	8000	287.0000000000000000
s65400	202232	8000	288.00	8000	288.0000000000000000
s63333	617459	8000	290.00	8000	290.0000000000000000
s69073	545721	8000	300.00	8000	300.0000000000000000
s47609	742565	8000	304.00	8000	304.0000000000000000
s16326	214891	8000	315.00	8000	315.0000000000000000
s21264	122999	8000	318.00	8000	318.0000000000000000
s1800	164939	8000	320.00	8000	320.0000000000000000
s114	375481	8000	321.00	8000	321.0000000000000000
s47332	770992	8000	328.00	8000	328.0000000000000000
s20305	349253	8000	329.00	8000	329.0000000000000000
s2366	478673	8000	332.00	8000	332.0000000000000000
s5540	740409	8000	341.00	8000	341.0000000000000000
s39698	776697	8000	347.00	8000	347.0000000000000000
s61161	583331	8000	354.00	8000	354.0000000000000000
s62226	425751	8000	356.00	8000	356.0000000000000000
s57635	132206	8000	364.00	8000	364.0000000000000000
s35364	175141	8000	366.00	8000	366.0000000000000000
s39848	136702	8000	368.00	8000	368.0000000000000000
s51210	518091	8000	386.00	8000	386.0000000000000000
s60139	324381	8000	389.00	8000	389.0000000000000000
s34759	525507	8000	391.00	8000	391.0000000000000000
s32393	730299	8000	394.00	8000	394.0000000000000000
s7037	190898	8000	410.00	8000	410.0000000000000000
s68692	742004	8000	412.00	8000	412.0000000000000000
s53666	249227	8000	412.00	8000	412.0000000000000000
s56293	190225	8000	418.00	8000	418.0000000000000000
s57431	216729	8000	428.00	8000	428.0000000000000000
s30814	729853	8000	438.00	8000	438.0000000000000000
s61722	195124	8000	444.00	8000	444.0000000000000000
s2476	493622	8000	454.00	8000	454.0000000000000000

Run 2:

Creating Non Clustered Indexing for the second run::

```
advdb=# ALTER TABLE trades_uniform DROP CONSTRAINT trades_uniform_pkey;
ALTER TABLE
advdb=# ALTER TABLE trades_fractal DROP CONSTRAINT trades_fractal_pkey;
ALTER TABLE
advdb=# CREATE INDEX idx_uniform_nonclustered ON trades_uniform (stock_symbol, time);
CREATE INDEX
advdb=# CREATE INDEX idx_fractal_nonclustered ON trades_fractal (stock_symbol, time);
CREATE INDEX
advdb=#
```

Run for fractal Dataset:

```
advdb=# EXPLAIN ANALYZE
SELECT stock_symbol, time, quantity, price,
       SUM(quantity) AS total_quantity,
       AVG(price) AS avg_price
FROM trades_fractal
WHERE time BETWEEN 100000 AND 800000
  AND price > 100
  AND quantity BETWEEN 500 AND 8000
GROUP BY stock_symbol, time, quantity, price
ORDER BY total_quantity DESC, avg_price ASC;
                                                                                QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=316478.58..317662.91 rows=473730 width=63) (actual time=11366.936..13050.872 rows=468152 loops=1)
   Sort Key: (sum(quantity)) DESC, (avg(price))
   Sort Method: external merge  Disk: 25264kB
   ->  Finalize GroupAggregate  (cost=189690.46..254009.26 rows=473730 width=63) (actual time=2355.243..9558.719 rows=468152 loops=1)
         Group Key: stock_symbol, "time", quantity, price
         ->  Gather Merge  (cost=189690.46..241179.05 rows=394776 width=63) (actual time=2355.186..5963.823 rows=468152 loops=1)
               Workers Planned: 2
               Workers Launched: 2
               ->  Partial GroupAggregate  (cost=188690.44..194612.08 rows=197388 width=63) (actual time=2240.829..3951.350 rows=156051 loops=3)
                     Group Key: stock_symbol, "time", quantity, price
                     ->  Sort  (cost=188690.44..189183.91 rows=197388 width=23) (actual time=2240.769..2829.924 rows=156051 loops=3)
                           Sort Key: stock_symbol, "time", quantity, price
                           Sort Method: external merge  Disk: 5376kB
                           Worker 0:  Sort Method: external merge  Disk: 5376kB
                           Worker 1:  Sort Method: external merge  Disk: 5328kB
                           ->  Parallel Seq Scan on trades_fractal  (cost=0.00..167280.00 rows=197388 width=23) (actual time=7.677..1144.793 rows=156051 loops=3)
                                 Filter: (("time" >= 100000) AND ("time" <= 800000) AND (price > '100'::numeric) AND (quantity >= 500) AND (quantity <= 8000))
                                 Rows Removed by Filter: 3177283
 Planning Time: 57.844 ms
 Execution Time: 14237.571 ms
(20 rows)
```

Run for Uniform Dataset:

```
advdb=# EXPLAIN ANALYZE
SELECT stock_symbol, time, quantity, price,
       SUM(quantity) AS total_quantity,
       AVG(price) AS avg_price
FROM trades_uniform
WHERE time BETWEEN 100000 AND 800000
  AND price > 100
  AND quantity BETWEEN 500 AND 8000
GROUP BY stock_symbol, time, quantity, price
ORDER BY total_quantity DESC, avg_price ASC;
                                                                            QUERY PLAN

-------------------------------------------------------------------------------------------------
------------------------------------------------------------------
 Sort  (cost=317110.75..318299.96 rows=475685 width=63) (actual time=11644.086..13282.170 rows=47
0037 loops=1)
   Sort Key: (sum(quantity)) DESC, (avg(price))
   Sort Method: external merge  Disk: 25368kB
   ->  Finalize GroupAggregate  (cost=189785.44..254369.49 rows=475685 width=63) (actual time=240
2.413..9827.377 rows=470037 loops=1)
         Group Key: stock_symbol, "time", quantity, price
         ->  Gather Merge  (cost=189785.44..241486.36 rows=396404 width=63) (actual time=2402.370
..6207.556 rows=470037 loops=1)
               Workers Planned: 2
               Workers Launched: 2
               ->  Partial GroupAggregate  (cost=188785.42..194731.48 rows=198202 width=63) (actu
al time=2248.428..3961.938 rows=156679 loops=3)
                     Group Key: stock_symbol, "time", quantity, price
                     ->  Sort  (cost=188785.42..189280.92 rows=198202 width=23) (actual time=2248
.363..2835.504 rows=156679 loops=3)
                           Sort Key: stock_symbol, "time", quantity, price
                           Sort Method: external merge  Disk: 5872kB
                           Worker 0:  Sort Method: external merge  Disk: 5256kB
                           Worker 1:  Sort Method: external merge  Disk: 5016kB
                           ->  Parallel Seq Scan on trades_uniform  (cost=0.00..167280.00 rows=19
8202 width=23) (actual time=12.770..1136.336 rows=156679 loops=3)
                                 Filter: (("time" >= 100000) AND ("time" <= 800000) AND (price >
'100'::numeric) AND (quantity >= 500) AND (quantity <= 8000))
                                 Rows Removed by Filter: 3176654
 Planning Time: 19.289 ms
 Execution Time: 14270.868 ms
(20 rows)
```

Thumb Rule #2 Runs:

Creating tables for Skewed and Uniform Data:

```
[sc10670@crunchy1 Q2]$ psql -p 34567 -h /home/sc10670/pgsql/data/run
psql (13.16)
Type "help" for help.

sc10670=# CREATE TABLE uniform_data (
    id BIGINT PRIMARY KEY,
    user_id BIGINT NOT NULL
);
CREATE TABLE
sc10670=# CREATE TABLE skewed_data (
    id BIGINT PRIMARY KEY,
    user_id BIGINT NOT NULL
);
CREATE TABLE
sc10670=# COPY uniform_data (id, user_id)
FROM '/home/sc10670/ADB/advDB-HW2/Q2/uniform_data_thumb2.csv'
DELIMITER ','
CSV HEADER;
COPY 10000000
sc10670=# COPY skewed_data (id, user_id)
FROM '/home/sc10670/ADB/advDB-HW2/Q2/skewed_data_thumb2.csv'
DELIMITER ','
CSV HEADER;
COPY 10000000
sc10670=#
```

Running query on Uniform Data without Indexing:

```
sc10670=# EXPLAIN ANALYZE
SELECT user_id, COUNT(*) AS frequency
FROM uniform_data
WHERE user_id BETWEEN 100000 AND 200000
GROUP BY user_id
ORDER BY frequency DESC;
                                                              QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------
--------
 Sort  (cost=345080.72..346634.58 rows=621543 width=16) (actual time=8314.985..8535.101 rows=99997 loops=1)
   Sort Key: (count(*)) DESC
   Sort Method: external merge  Disk: 2552kB
   ->  HashAggregate  (cost=260582.01..274648.71 rows=621543 width=16) (actual time=7567.132..8051.232 rows=99997 loops=1)
         Group Key: user_id
         Planned Partitions: 16  Batches: 17  Memory Usage: 4241kB  Disk Usage: 30896kB
         ->  Seq Scan on uniform_data  (cost=0.00..204052.90 rows=1004962 width=8) (actual time=91.263..4612.659 rows=999158 l
oops=1)
               Filter: ((user_id >= 100000) AND (user_id <= 200000))
               Rows Removed by Filter: 9000842
 Planning Time: 19.286 ms
 Execution Time: 9148.476 ms
(11 rows)

sc10670=# EXPLAIN ANALYZE
SELECT user_id, COUNT(*) AS frequency
FROM skewed_data
WHERE user_id BETWEEN 1 AND 1000
GROUP BY user_id
ORDER BY frequency DESC;
```

Running query on Skewed Data without Indexing:

```
sc10670=# EXPLAIN ANALYZE
SELECT user_id, COUNT(*) AS frequency
FROM skewed_data
WHERE user_id BETWEEN 1 AND 1000
GROUP BY user_id
ORDER BY frequency DESC;
                                                                          QUERY PLAN

-----------------------------------------------------------------------------------------------------------------------------
---------------------------------
 Sort  (cost=138733.93..138736.31 rows=951 width=16) (actual time=19351.391..22198.332 rows=1000 loops=1)
   Sort Key: (count(*)) DESC
   Sort Method: quicksort  Memory: 71kB
   ->  Finalize GroupAggregate  (cost=138445.95..138686.89 rows=951 width=16) (actual time=19326.888..22193.614 rows=1000 loop
s=1)
         Group Key: user_id
         ->  Gather Merge  (cost=138445.95..138667.87 rows=1902 width=16) (actual time=19326.848..22183.699 rows=3000 loops=1)
               Workers Planned: 2
               Workers Launched: 2
               ->  Sort  (cost=137445.93..137448.30 rows=951 width=16) (actual time=19294.207..19296.723 rows=1000 loops=3)
                     Sort Key: user_id
                     Sort Method: quicksort  Memory: 71kB
                     Worker 0:  Sort Method: quicksort  Memory: 71kB
                     Worker 1:  Sort Method: quicksort  Memory: 71kB
                     ->  Partial HashAggregate  (cost=137389.37..137398.88 rows=951 width=16) (actual time=19288.305..19291.09
6 rows=1000 loops=3)
                           Group Key: user_id
                           Batches: 1  Memory Usage: 193kB
                           Worker 0:  Batches: 1  Memory Usage: 193kB
                           Worker 1:  Batches: 1  Memory Usage: 193kB
                           ->  Parallel Seq Scan on skewed_data  (cost=0.00..116556.09 rows=4166656 width=8) (actual time=0.36
5..9621.322 rows=3333333 loops=3)
                                 Filter: ((user_id >= 1) AND (user_id <= 1000))
 Planning Time: 1.474 ms
 Execution Time: 22200.418 ms
(22 rows)
```

Running query on Uniform Data with Indexing:

```
sc10670=# CREATE INDEX idx_user_id ON uniform_data(user_id);
CREATE INDEX
sc10670=# CREATE INDEX idx_user_id2 ON skewed_data(user_id);
CREATE INDEX
sc10670=# EXPLAIN ANALYZE
SELECT user_id, COUNT(*) AS frequency
FROM uniform_data
WHERE user_id BETWEEN 100000 AND 200000
GROUP BY user_id
ORDER BY frequency DESC;
                                                                          QUERY PLAN

-----------------------------------------------------------------------------------------------------------------------------
-------------------------------
 Sort  (cost=106332.73..107886.60 rows=621547 width=16) (actual time=5229.736..5448.095 rows=99997 loops=1)
   Sort Key: (count(*)) DESC
   Sort Method: external merge  Disk: 2552kB
   ->  GroupAggregate  (cost=0.43..35900.31 rows=621547 width=16) (actual time=0.978..4964.843 rows=99997 loops=1)
         Group Key: user_id
         ->  Index Only Scan using idx_user_id on uniform_data  (cost=0.43..24659.96 rows=1004976 width=8) (actual time=0.928.
.2431.084 rows=999158 loops=1)
               Index Cond: ((user_id >= 100000) AND (user_id <= 200000))
               Heap Fetches: 0
 Planning Time: 2.438 ms
 Execution Time: 5648.564 ms
(10 rows)

sc10670=# EXPLAIN ANALYZE
SELECT user_id, COUNT(*) AS frequency
FROM skewed_data
WHERE user_id BETWEEN 1 AND 1000
GROUP BY user_id
ORDER BY frequency DESC;
```

Running query on Skewed Data with Indexing:

```
sc10670=# EXPLAIN ANALYZE
SELECT user_id, COUNT(*) AS frequency
FROM skewed_data
WHERE user_id BETWEEN 1 AND 1000
GROUP BY user_id
ORDER BY frequency DESC;
                                                                          QUERY PLAN

-----------------------------------------------------------------------------------------------------------------------------
---------------------------------
 Sort  (cost=138732.89..138735.27 rows=951 width=16) (actual time=19936.806..19939.001 rows=1000 loops=1)
   Sort Key: (count(*)) DESC
   Sort Method: quicksort  Memory: 71kB
   ->  Finalize GroupAggregate  (cost=138444.91..138685.85 rows=951 width=16) (actual time=19912.477..19934.356 rows=1000 loop
s=1)
         Group Key: user_id
         ->  Gather Merge  (cost=138444.91..138666.83 rows=1902 width=16) (actual time=19912.433..19924.414 rows=3000 loops=1)
               Workers Planned: 2
               Workers Launched: 2
               ->  Sort  (cost=137444.89..137447.27 rows=951 width=16) (actual time=19900.879..19903.455 rows=1000 loops=3)
                     Sort Key: user_id
                     Sort Method: quicksort  Memory: 71kB
                     Worker 0:  Sort Method: quicksort  Memory: 71kB
                     Worker 1:  Sort Method: quicksort  Memory: 71kB
                     ->  Partial HashAggregate  (cost=137388.33..137397.85 rows=951 width=16) (actual time=19894.984..19897.77
2 rows=1000 loops=3)
                           Group Key: user_id
                           Batches: 1  Memory Usage: 193kB
                           Worker 0:  Batches: 1  Memory Usage: 193kB
                           Worker 1:  Batches: 1  Memory Usage: 193kB
                           ->  Parallel Seq Scan on skewed_data  (cost=0.00..116555.00 rows=4166667 width=8) (actual time=72.7
80..10166.368 rows=3333333 loops=3)
                                 Filter: ((user_id >= 1) AND (user_id <= 1000))
 Planning Time: 1.903 ms
 Execution Time: 19941.123 ms
(22 rows)
```