Question One answers

Heap Scan:

- There are B pages to be retrieve from disk (the heap file)
- Each page has expected cost of D
- Therefore, the expected cost of scan is BD

Heap Equality:

- The expected location of the record that satisfy the equality condition is in the middle of the heap file
- There are B data pages for the heap file
- Each page has expected cost of D
- Therefore, the expected cost of scan is $0.5BD$

Heap Range:

- Since heap file is unordered, which means it requires to scan the whole heap file
- There are B data pages
- Each page has expected cost of D
- Therefore, the expected cost of scan is BD

Heap Insert:

- For heap file, we insert the record at the end of the last page
- Retrieve the whole disk page to memory has expected cost of D
- Insert the modified disk page back to memory has expected cost of D
- Therefore, the expected cost of insert is $2D$

Heap Delete:

- Since heap file is unordered, and the expected location of the record to be deleted is in the middle of the heap file, thus the expected cost would be $0.5BD$
- After delete the record in the disk page, the cost of put the disk page back to disk is D
- Therefore, the expected cost of delete is $0.5BD + D = Search + D$

Sorted Scan:

- There are B data pages to be retrieve from disk
- Each page has expected cost of D
- Therefore, the expected cost of scan is BD

Sorted Equality:

- The sorted file has records in sorted order
- There are B data pages total, using binary search, we only need to retrieve $\log_2 B$ pages to find the desired record, due to the sorted order of records
- Each page has expected cost of D
- Therefore, the expected cost of equality search is $D \log_2 B$

Sorted Range:

- Since the sorted file has records in sorted order
- There are B pages total, using binary search, we only need to retrieve $\log_2 B$ pages to find the first record that satisfy the lower bound of the range condition
- Since it is range search, records that satisfy the range condition may span on multiple pages
- Each page has expected cost of D
- Therefore, the expected cost of range search is $D(\log_2 B + \#\ of\ pages\ with\ match\ records)$

Sorted Insert:

- Since the sorted file has records in sorted order
- Use equality search to find the correct position to be inserted is $Dlog_2 B$ (proven from above) to maintain the sorted property
- The expected location to insert the record is in the middle of the file, thus it cost $0.5BD$ to retrieve half the number of pages of a sorted file into memory
- After pushes down the position of each record in the retrieved pages, we put the pages back onto the disk, which cost another $0.5BD$
- Therefore, the expected cost of insert operation is

$$Dlog_2 B + 2(0.5BD) = Dlog_2 B + BD = Search + BD$$

Sorted Delete:

- Since the sorted file has records in sorted order
- Use equality search to find the correct record to be deleted is $Dlog_2 B$ (proven from above)
- The expected location of the record to be deleted is in the middle of the file, thus it cost $0.5BD$ to retrieve half the number of pages of a sorted file into memory
- After pushes up the position of each record in the retrieved pages, we put the pages back onto the disk, which cost another $0.5BD$
- Therefore, the expected cost of delete operation is

$$Dlog_2 B + 2(0.5BD) = Dlog_2 B + BD = Search + BD$$

Clustered Scan:

- Since each data page only has 67% full, and there are B data pages, thus, we need $\frac{1}{0.67}B = 1.5B$ data pages to store all the records
- Each page has expected cost of D
- Therefore, the expected cost of Scan operation is $1.5BD$

Clustered Equality:

- Since each data page only has 67% full, and there are Bdata pages, thus, we need $\frac{1}{0.67}B = 1.5B$ data pages to store all the records
- Assume each interior page of a B-tree has F children, then, $the\ number\ of\ tree\ levels\ = number\ of\ interior\ pages\ to\ read\ =\ \log_F 1.5B$ to get the leaf page that has the desired record in it

- Each page has expected cost of D
- Therefore, the expected cost of Equality operation is $D(\log_F 1.5B)$

Clustered Range:

- Since each data page only has 67% full, and there are B data pages, thus, we need $\frac{1}{0.67}B = 1.5B$ data pages to store all the records
- Assume each interior page of a B-tree has F children, then, $the\ number\ of\ tree\ levels = number\ of\ interior\ pages\ to\ read = \log_F 1.5B$ to get the leaf page that has the first record that satisfy the lower bound of the range condition
- Since it is range search, records that satisfy the range condition may span on multiple pages
- Leaf pages are linked together
- Therefore, the expected cost of the range operation is

$$D(\log_F 1.5B + \#\ pages\ with\ match\ records)$$

Clustered Insert:

- the cost of finding the correct leaf page to insert the new record is $D(\log_F 1.5B)$ to maintain the sorted property (equality search)
- the cost of retrieve the whole leaf page is D
- the cost of put the modified leaf page (with a new record) back onto disk is D
- Therefore, the expected cost of insert operation is $D(\log_F 1.5B) + 2\mathrm{D} = Search + 2D$

Clustered delete:

- the cost of finding the correct leaf page that has the record to be deleted is $D(\log_F 1.5B)$
- the cost of retrieve the whole leaf page is D
- the cost of put the modified leaf page (without a record) back onto disk is D
- Therefore, the expected cost of insert operation is $D(\log_F 1.5B) + 2\mathrm{D} = Search + 2D$

Unclustered Tree index scan:

- Since each data page only has 67% full, and there are B data pages, thus, we need $\frac{1}{0.67}B = 1.5B$ data pages to store all the records
- Since data entry size is 10% size of record, then we have $(0.1)(1.5B) = 0.15B$ pages to store all the data entries
- To retrieve the pages with matching record for each entry, which is BR
- Each page cost D to retrieve
- Therefore, the expected cost of scan operation is $BD(R + 0.15)$

Unclustered Tree index Equality:

- Since each data page only has 67% full, and there are B pages, thus, we need $\frac{1}{0.67}B = 1.5B$ data pages to store all the records
- Since data entry size is 10% size of record, then we have $(0.1)(1.5B) = 0.15B$ pages to store all the data entries

- Assume on average the number of children each interior index page has is F, then we have $\log_F 0.15B$ index pages to retrieve to get to the location of the leaf page that contains the desired data entry, since each page cost D, we have cost of $Dlog_F B$
- Using the data entry and retrieve the data page in the heap file that contains the desired record, which cost another D
- Therefore, the expected cost of equality search is $D(1 + \log_F 0.15B)$

Unclustered Tree index Range:

- Since each data page only has 67% full, and there are B data pages, thus, we need $\frac{1}{0.67}B = 1.5B$ data pages to store all the records
- Since data entry size is 10% size of record, then we have $(0.1)(1.5B) = 0.15B$ pages to store all the data entries
- The number of tree levels = number of index pages to retrieve $= \log_F 0.15B$ pages; each page cost D, then we have cost of $Dlog_F 0.15B$ to traverse from the root index page to the leaf page that contains the desired data entry
- Assume in the worst case, each desired record that satisfy the range condition is in different data page, then we have cost of $D(\#\ of\ pages\ with\ match\ records)$
- Therefore, the expected cost of the range operation is
  $D(log_F 0.15B + \#\ of\ pages\ with\ match\ records)$

Unclustered Tree index Insert:

- From above, we know that the cost of equality search in the unclustered index B-tree for the correct location is $D(1 + \log_F 0.15B)$, the cost includes retrieving the desired data page
- The cost of put back the modified data page (with new record) onto the disk is D
- The cost of put back the modified index page (with new entry for the new record) onto the disk is D
- Therefore, the expected cost of the insert operation is $D(1 + \log_F 0.15B) + 2D = Search + 2D$

Unclustered Tree index delete:

- From above, we know that the cost of equality search in the unclustered index B-tree for the correct location is $D(1 + \log_F 0.15B)$, the cost includes retrieving the desired data page
- The cost of put back the modified data page (without a record) onto the disk is D
- The cost of put back the modified index page (without an entry) onto the disk is D
- Therefore, the expected cost of the insert operation is $D(1 + \log_F 0.15B) + 2D = Search + 2D$

Unclustered Hash index scan:

- Since each data page in a hash index only 80% full, which implies that we need $\frac{1}{0.8} = 1.25$ data pages to store all the records
- Since data entry size occupies 10% of the data records, which implies that we have $(0.1)(1.25) = 0.125$ pages to store all the data entries
- Each page has cost of D, then we have cost of $0.125BD$ to get all the pages that contains all the data entries
- To retrieve the record from the heap file and assume each record locates on different data page, we have a cost of DBR

- Therefore, the expected cost of the scan operation is $BD(R + 0.125)$

Unclustered Hash index equality:
- Assume the cost of execute the hash function is negligible
- The hash function points directly to the correct bucket and assume no overflow pages in that bucket
- The cost of retrieve the index page for the bucket is D to read in memory
- The cost of retrieve the actual data page for the heap file is D to read in memory
- Therefore, the expected cost of equality search is $2D$

Unclustered Hash index range:
- Since the hash index offers no help with range search
- Requires scanning the whole heap file, which includes all the data pages
- There are B data pages for the heap file
- Each page has expected cost of D
- Therefore, the expected cost of range search is BD

Unclustered Hash index Insert:
- Assume the cost of execute the hash function is negligible
- The cost of locate the correct bucket, retrieve the index page, and retrieve the data page is $2D$, proven by above
- The cost of put back the modified index page (with a new entry) onto the disk is D
- The cost of put back the modified data page (with a new record) onto the disk is D
- Therefore, the expected cost of insert is $2D + 2D = Search + 2D$

Unclustered Hash index delete:
- Assume the cost of execute the hash function is negligible
- The cost of locate the correct bucket, retrieve the index page, and retrieve the data page is $2D$, proven by above
- The cost of put back the modified index page (without an entry) onto the disk is D
- The cost of put back the modified data page (without a record) onto the disk is D
- Therefore, the expected cost of delete is $2D + 2D = Search + 2D$

Question 2.5 and 2.6 Answers (answers for 2.4 is at the end)

Database	Without any index with page size of 4KB			
Query operation	Query and print the employee id and full name of anybody whose last name is "Rowe"			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	27699	0	0	0.00611ms

Comparison:
Not consistent, the following are the reasoning:
- SQLite creates a table B-tree with key as rowid and value as the record in the leaf pages, it is a clustered B-tree

- the reserved space for each page is 0 bytes (got from the database file header)
- SQLite does not use heap file to store actual data (it uses table B-Tree)

Therefore, the result above got from traverse all the pages of a table B-Tree (interior and leaf), instead of just data pages $1.5B$ obtained in Q1

Database	Without any index with page size of 4KB			
Query operation	Query and print the full name of employee #181162			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	27675	0	0	0.00683

Comparison:

Not consistent, the following are the reasoning:

- the reserved space for each page is 0 bytes
- the sorted property of rowid key does not help speed up the search with an equality condition of employee id

Therefore, the result here comes from a scan operation until it finds a record with employee #181162 instead of $Dlog_F(1.5B)$.

Database	Without any index with page size of 4KB			
Query operation	Query and print the employee id and full name of all employees with "Emp ID" between #171800 and #171899			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	27699	0	0	0.00583

Comparison:

Not consistent, the following are the reasoning:

- The reserved space for each page is 0 bytes
- Sorted property of the rowid in each cell of each page does not speed up the search of a record with equality condition, employee id

Therefore, the result comes from a scan operation that traverse all the pages of a table B-tree instead of $D(\log_F 1.5B + \#\ pages\ with\ match\ records)$

Database	Without any index but with page size of 16KB bytes			
Query operation	Query and print the employee id and full name of anybody whose last name is "Rowe"			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	6723	0	0	0.0154

Comparison:

Not consistent, the following are the reasoning:

- SQLite creates a table B-tree with key as rowid and value as the record in the leaf pages, it is a clustered B-tree

- the reserved space for each page is 0 bytes (got from the database file header)
- SQLite does not use heap file to store actual data (it uses table B-Tree)

Therefore, the result above got from traverse all the pages of a table B-Tree (interior and leaf), instead of just data pages $1.5B$ obtained in Q1

Database	Without any index but with page size of 16KB bytes			
Query operation	Query and print the full name of employee #181162			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	6382	0	0	0.0162

Comparison:

Not consistent, the following are the reasoning:

- the reserved space for each page is 0 bytes
- the sorted property of rowid key does not help speed up the search with an equality condition of employee id

Therefore, the result here comes from a scan operation until it finds a record with employee #181162 instead of $Dlog_F(1.5B)$.

Database	Without any index but with page size of 16KB bytes			
Query operation	Query and print the employee id and full name of all employees with "Emp ID" between #171800 and #171899			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	6723	0	0	0.0160

Comparison:

Not consistent, the following are the reasoning:

- The reserved space for each page is 0 bytes
- Sorted property of the rowid in each cell of each page does not speed up the search of employee id

Therefore, the result comes from a scan operation that traverse all the pages of a table B-tree instead of $D(\log_F 1.5B + \#\ pages\ with\ match\ records)$

Database	With primary index on "Emp ID" column (Unclustered Index) with page size of 4KB			
Query operation	Query and print the employee id and full name of anybody whose last name is "Rowe"			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	27699	0	0	0.0071

Comparison:

Not consistent, the following are the reasoning:

- The indexed column is “Emp ID” but not last name, thus the index is not useful for this type of query

- Table B-tree sorts the cells by rowid, instead of last name, thus cannot use the key to speed up the search
- Table B-tree has all the records in the leaf pages
- The reserved space for each page is 0 bytes
- SQLite does not use heap file to store actual data (it uses table B-Tree)

Therefore, the result above got from traverse all the pages of a table B-Tree (interior and leaf), instead of just data pages B obtained in Q1

Database	With primary index on "Emp ID" column (Unclustered Index) with page size of 4KB			
Query operation	Query and print the full name of employee #181162			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	3	2	1	0.016

Comparison:

Not consistent, the following are the reasoning:

- each index page contains a list of data entry as (Emp_ID, rowid) inside each cell, sorted by Emp_ID in each page
- each page does not have reserved space
- the records in cell, in each page of index B-Tree, not necessary 10% of the actual record data
- SQLite does not use heap file to store data (it uses table B-Tree)

Therefore, the result comes from traverse the index B-Tree to find #181162 and get the corresponding rowid and use the rowid to traverse the table B-Tree (use rowid to decides with child page to go) until it gets to a leaf page that contains the record, instead of $(1 + \log_F 0.15B)$ page reads obtained in Q1

Database	With primary index on "Emp ID" column (Unclustered Index) with page size of 4KB			
Query operation	Query and print the employee id and full name of all employees with "Emp ID" between #171800 and #171899			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	114	3	4	0.0141

Comparison:

Not consistent, the following are the reasoning:

- each index page contains a list of data entry as (Emp_ID, rowid) inside each cell, sorted by Emp_ID in each page
- the records in cell, in each page of index B-Tree, not necessary 10% of the actual record data
- each page does not have reserved space
- The leaf pages of an index B-Tree are not linked together
- SQLite does not use heap file to store data (it uses table B-Tree)

Therefore, SQLite traverse the index B-Tree once and repeatedly traverse the table B-Tree K times, K is number of records that satisfy the range condition, instead of the followings

$$\log_F(0.15B) + \#\ page\ with\ match\ records$$

Database	With primary index on "Emp ID" column but defined as clustered with page size of 4KB			
Query operation	Query and print the employee id and full name of anybody whose last name is "Rowe"			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	0	2437	29516	0.0194

Comparison:

Not consistent, the following are the reasoning:

- SQLite creates an index B-tree with key as Emp_ID and value as the rest of columns of record
- both interior and leaf index pages contain record data.
- the reserved space for each page is 0 bytes (got from the database file header)
- SQLite does not use heap file to store actual data (it uses index B-Tree in this case)
- Clustered B-tree is an index B-tree in SQLite

Therefore, the result comes from a scan operation of the index B-tree instead of the heap file scan in Q1.

Database	With primary index on "Emp ID" column but defined as clustered with page size of 4KB			
Query operation	Query and print the full name of employee #181162			
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	0	4	1	0.0104

Comparison:

Not consistent, the following are the reasoning:

- SQLite creates an index B-tree with key as Emp_ID and value as the rest of columns of record; both interior and leaf index pages contain record data.
- the reserved space for each page is 0 bytes (got from the database file header)
- SQLite does not use heap file to store actual data (it uses index B-Tree)
- Clustered B-tree is an index B-tree in SQLite
- Record locates in both interior and leaf pages

Therefore, the result comes from traversal of an index B-Tree and find the record in either an interior page or a leaf page (which child page to go is decided by the Emp_ID); instead of $\log_F 1.5B$ in Q1, in which case it needs to traverse all the way down to the leaf page that contains the desired record.

Database	With primary index on "Emp ID" column but defined as clustered with page size of 4KB
Query operation	Query and print the employee id and full name of all employees with "Emp ID" between #171800 and #171899

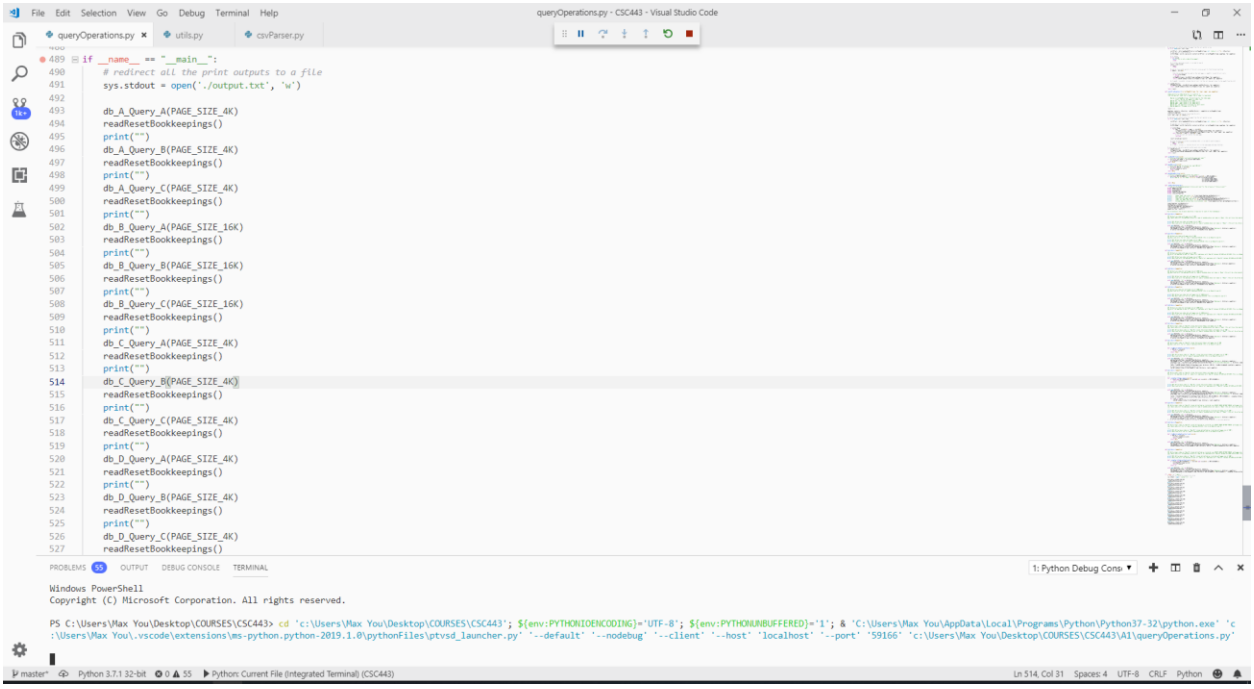
Header page read counts	Data page read counts	index internal page read counts	index leaf page read counts	Average page accessing time in ms
1	0	16	19	0.0112

Comparison:
the result here needs to read number of pages with match records, which is same as the one in Q1. However, the number of pages read by SQLite may be smaller than the one in Q1 since records also locate in the interior index page, whereas the cost in Q1 assumes that all data locates in the leaf pages only.

- SQLite creates an index B-tree with key as Emp_ID and value as the rest of columns of record; both interior and leaf index pages contain record data.
- the reserved space for each page is 0 bytes (got from the database file header)
- SQLite does not use heap file to store actual data (it uses index B-Tree)
- Clustered B-tree is an index B-tree in SQLite
- Record locates in both interior and leaf pages
- Leaf pages are not linked together by pointers

Therefore, the result comes from traverse the index B-tree until see an Emp_ID that goes beyond the upper bound of range condition, instead of $\log_F 0.15B + \#\ pgs\ with\ match\ records$

Screenshot of running my program (note that I redirect all the std_out output to a file called ouput.txt at line 491):



Answers for Question 2.4

The following are all the output results from the **output.txt (also inside the A1.zip file):**

DB: Without any index with page size of 4KB

Query and print the employee id and full name of anybody whose last name is "Rowe"; (this will be a Scan operation)

- Emp ID: 398361, Full Name: Sherman W Rowe
- Emp ID: 403530, Full Name: Aleta B Rowe
- Emp ID: 783799, Full Name: Enoch K Rowe
- Emp ID: 176587, Full Name: Kiera S Rowe
- Emp ID: 339175, Full Name: Glenn U Rowe
- Emp ID: 189582, Full Name: Zenaida J Rowe
- Emp ID: 622253, Full Name: Rhett M Rowe
- Emp ID: 771556, Full Name: Shaina E Rowe
- Emp ID: 959340, Full Name: Jacquelyne W Rowe
- Emp ID: 167422, Full Name: Alphonse Z Rowe
- Emp ID: 313265, Full Name: Flo S Rowe
- Emp ID: 631156, Full Name: Shirleen Y Rowe
- Emp ID: 689557, Full Name: Daria Y Rowe
- Emp ID: 657000, Full Name: Angel Y Rowe
- Emp ID: 199223, Full Name: Kimiko T Rowe
- Emp ID: 703373, Full Name: Beatrice H Rowe
- Emp ID: 144809, Full Name: Tommy W Rowe
- Emp ID: 808221, Full Name: Lane O Rowe
- Emp ID: 667295, Full Name: Edmund W Rowe
- Emp ID: 926265, Full Name: Ramonita M Rowe
- Emp ID: 142327, Full Name: Simonne L Rowe
- Emp ID: 540314, Full Name: Chantay R Rowe

DB: Without any index with page size of 4KB

Query and print the full name of employee #181162 (this is an Equality search)

- Full Name: Cristobal C Foy

DB: Without any index with page size of 4KB

Query and print the employee id and full name of all employees with "Emp ID" between #171800 and #171899 (This is a Range search)

- Emp ID: 171885, Full Name: Dudley R Kelemen
- Emp ID: 171816, Full Name: Corie P Alcorn
- Emp ID: 171844, Full Name: Mika O Bradshaw
- Emp ID: 171834, Full Name: Otis C Mcrae
- Emp ID: 171886, Full Name: Rosalee Q Rutz
- Emp ID: 171852, Full Name: Shemika Y Cutts
- Emp ID: 171832, Full Name: Deshawn Q Tavarez

- Emp ID: 171845, Full Name: Sanora Y Hardwick
- Emp ID: 171814, Full Name: Junita N Schoenberg
- Emp ID: 171887, Full Name: Isis N Dusek
- Emp ID: 171815, Full Name: Elenor U Kivett
- Emp ID: 171867, Full Name: Hobert G Saleh
- Emp ID: 171851, Full Name: Ashely D Flavin
- Emp ID: 171869, Full Name: Karl V Hooley
- Emp ID: 171859, Full Name: Katelyn K Moir
- Emp ID: 171896, Full Name: Joel P Berrier
- Emp ID: 171839, Full Name: Valentin Q Glassman
- Emp ID: 171892, Full Name: Porter G Sapien
- Emp ID: 171889, Full Name: Collene O Spino
- Emp ID: 171875, Full Name: Aubrey D Bounds
- Emp ID: 171858, Full Name: Mason C Burchette
- Emp ID: 171856, Full Name: Emmitt F Olmos
- Emp ID: 171826, Full Name: Winnie Q Mass
- Emp ID: 171868, Full Name: Dana E Goosby
- Emp ID: 171843, Full Name: Apryl Y Tavera
- Emp ID: 171865, Full Name: Maximo A Failla
- Emp ID: 171812, Full Name: Alise P Earnhardt
- Emp ID: 171860, Full Name: Warren M Hicklin
- Emp ID: 171879, Full Name: Darell V Motes
- Emp ID: 171849, Full Name: Bobby F Reep
- Emp ID: 171855, Full Name: Vanessa L Philpott
- Emp ID: 171874, Full Name: Elba X Finnegan
- Emp ID: 171823, Full Name: Ivana G Pool
- Emp ID: 171818, Full Name: Barney U Kirschbaum
- Emp ID: 171881, Full Name: Anamaria B Buzzell
- Emp ID: 171847, Full Name: Gwyn F Goodrum
- Emp ID: 171841, Full Name: Deon I Hallam
- Emp ID: 171895, Full Name: Lyle D Weedman

DB: Without any index but with page size of 16KB bytes
Query and print the employee id and full name of anybody whose last name is "Rowe" (this will be a Scan operation)

- Emp ID: 398361, Full Name: Sherman W Rowe
- Emp ID: 403530, Full Name: Aleta B Rowe
- Emp ID: 783799, Full Name: Enoch K Rowe
- Emp ID: 176587, Full Name: Kiera S Rowe
- Emp ID: 339175, Full Name: Glenn U Rowe
- Emp ID: 189582, Full Name: Zenaida J Rowe
- Emp ID: 622253, Full Name: Rhett M Rowe
- Emp ID: 771556, Full Name: Shaina E Rowe

- Emp ID: 959340, Full Name: Jacquelyne W Rowe
- Emp ID: 167422, Full Name: Alphonse Z Rowe
- Emp ID: 313265, Full Name: Flo S Rowe
- Emp ID: 631156, Full Name: Shirleen Y Rowe
- Emp ID: 689557, Full Name: Daria Y Rowe
- Emp ID: 657000, Full Name: Angel Y Rowe
- Emp ID: 199223, Full Name: Kimiko T Rowe
- Emp ID: 703373, Full Name: Beatrice H Rowe
- Emp ID: 144809, Full Name: Tommy W Rowe
- Emp ID: 808221, Full Name: Lane O Rowe
- Emp ID: 667295, Full Name: Edmund W Rowe
- Emp ID: 926265, Full Name: Ramonita M Rowe
- Emp ID: 142327, Full Name: Simonne L Rowe
- Emp ID: 540314, Full Name: Chantay R Rowe

DB: Without any index but with page size of 16KB bytes
Query and print the full name of employee #181162 (this is an Equality search)

- Full Name: Cristobal C Foy

DB: Without any index but with page size of 16KB byte
Query and print the employee id and full name of all employees with "Emp ID" between #171800 and #171899 (This is a Range search)

- Emp ID: 171885, Full Name: Dudley R Kelemen
- Emp ID: 171816, Full Name: Corie P Alcorn
- Emp ID: 171844, Full Name: Mika O Bradshaw
- Emp ID: 171834, Full Name: Otis C Mcrae
- Emp ID: 171886, Full Name: Rosalee Q Rutz
- Emp ID: 171852, Full Name: Shemika Y Cutts
- Emp ID: 171832, Full Name: Deshawn Q Tavarez
- Emp ID: 171845, Full Name: Sanora Y Hardwick
- Emp ID: 171814, Full Name: Junita N Schoenberg
- Emp ID: 171887, Full Name: Isis N Dusek
- Emp ID: 171815, Full Name: Elenor U Kivett
- Emp ID: 171867, Full Name: Hobert G Saleh
- Emp ID: 171851, Full Name: Ashely D Flavin
- Emp ID: 171869, Full Name: Karl V Hooley
- Emp ID: 171859, Full Name: Katelyn K Moir
- Emp ID: 171896, Full Name: Joel P Berrier
- Emp ID: 171839, Full Name: Valentin Q Glassman
- Emp ID: 171892, Full Name: Porter G Sapien
- Emp ID: 171889, Full Name: Collene O Spino
- Emp ID: 171875, Full Name: Aubrey D Bounds
- Emp ID: 171858, Full Name: Mason C Burchette

- Emp ID: 171856, Full Name: Emmitt F Olmos
- Emp ID: 171826, Full Name: Winnie Q Mass
- Emp ID: 171868, Full Name: Dana E Goosby
- Emp ID: 171843, Full Name: Apryl Y Tavera
- Emp ID: 171865, Full Name: Maximo A Failla
- Emp ID: 171812, Full Name: Alise P Earnhardt
- Emp ID: 171860, Full Name: Warren M Hicklin
- Emp ID: 171879, Full Name: Darell V Motes
- Emp ID: 171849, Full Name: Bobby F Reep
- Emp ID: 171855, Full Name: Vanessa L Philpott
- Emp ID: 171874, Full Name: Elba X Finnegan
- Emp ID: 171823, Full Name: Ivana G Pool
- Emp ID: 171818, Full Name: Barney U Kirschbaum
- Emp ID: 171881, Full Name: Anamaria B Buzzell
- Emp ID: 171847, Full Name: Gwyn F Goodrum
- Emp ID: 171841, Full Name: Deon I Hallam
- Emp ID: 171895, Full Name: Lyle D Weedman

DB: With primary index on "Emp ID" column (Unclusterd Index) with page size of 4KB
Query and print the employee id and full name of anybody whose last name is "Rowe" (this will be a Scan operation)

- Emp ID: 398361, Full Name: Sherman W Rowe
- Emp ID: 403530, Full Name: Aleta B Rowe
- Emp ID: 783799, Full Name: Enoch K Rowe
- Emp ID: 176587, Full Name: Kiera S Rowe
- Emp ID: 339175, Full Name: Glenn U Rowe
- Emp ID: 189582, Full Name: Zenaida J Rowe
- Emp ID: 622253, Full Name: Rhett M Rowe
- Emp ID: 771556, Full Name: Shaina E Rowe
- Emp ID: 959340, Full Name: Jacquelyne W Rowe
- Emp ID: 167422, Full Name: Alphonse Z Rowe
- Emp ID: 313265, Full Name: Flo S Rowe
- Emp ID: 631156, Full Name: Shirleen Y Rowe
- Emp ID: 689557, Full Name: Daria Y Rowe
- Emp ID: 657000, Full Name: Angel Y Rowe
- Emp ID: 199223, Full Name: Kimiko T Rowe
- Emp ID: 703373, Full Name: Beatrice H Rowe
- Emp ID: 144809, Full Name: Tommy W Rowe
- Emp ID: 808221, Full Name: Lane O Rowe
- Emp ID: 667295, Full Name: Edmund W Rowe
- Emp ID: 926265, Full Name: Ramonita M Rowe
- Emp ID: 142327, Full Name: Simonne L Rowe
- Emp ID: 540314, Full Name: Chantay R Rowe

DB: With primary index on "Emp ID" column (Unclusterd Index) with page size of 4KB
Query and print the full name of employee #181162 (this is an Equality search)

- Full Name: Cristobal C Foy

DB: With primary index on "Emp ID" column (Unclusterd Index) with page size of 4KB
Query and print the employee id and full name of all employees with "Emp ID" between #171800 and #171899 (This is a Range search)

- Full Name: Alise P Earnhardt
- Full Name: Junita N Schoenberg
- Full Name: Elenor U Kivett
- Full Name: Corie P Alcorn
- Full Name: Barney U Kirschbaum
- Full Name: Ivana G Pool
- Full Name: Winnie Q Mass
- Full Name: Deshawn Q Tavarez
- Full Name: Otis C Mcrae
- Full Name: Valentin Q Glassman
- Full Name: Deon I Hallam
- Full Name: Apryl Y Tavera
- Full Name: Mika O Bradshaw
- Full Name: Sanora Y Hardwick
- Full Name: Gwyn F Goodrum
- Full Name: Bobby F Reep
- Full Name: Ashely D Flavin
- Full Name: Shemika Y Cutts
- Full Name: Vanessa L Philpott
- Full Name: Emmitt F Olmos
- Full Name: Mason C Burchette
- Full Name: Katelyn K Moir
- Full Name: Warren M Hicklin
- Full Name: Maximo A Failla
- Full Name: Hobert G Saleh
- Full Name: Dana E Goosby
- Full Name: Karl V Hooley
- Full Name: Elba X Finnegan
- Full Name: Aubrey D Bounds
- Full Name: Darell V Motes
- Full Name: Anamaria B Buzzell
- Full Name: Dudley R Kelemen
- Full Name: Rosalee Q Rutz
- Full Name: Isis N Dusek
- Full Name: Collene O Spino

- Full Name: Porter G Sapien
- Full Name: Lyle D Weedman
- Full Name: Joel P Berrier

DB: With primary index on "Emp ID" column but defined as clustered with page size of 4KB
Query and print the employee id and full name of anybody whose last name is "Rowe" (this will be a Scan operation)

- Emp ID: 142327, Full Name: Simonne L Rowe
- Emp ID: 144809, Full Name: Tommy W Rowe
- Emp ID: 167422, Full Name: Alphonse Z Rowe
- Emp ID: 176587, Full Name: Kiera S Rowe
- Emp ID: 189582, Full Name: Zenaida J Rowe
- Emp ID: 199223, Full Name: Kimiko T Rowe
- Emp ID: 313265, Full Name: Flo S Rowe
- Emp ID: 339175, Full Name: Glenn U Rowe
- Emp ID: 398361, Full Name: Sherman W Rowe
- Emp ID: 403530, Full Name: Aleta B Rowe
- Emp ID: 540314, Full Name: Chantay R Rowe
- Emp ID: 622253, Full Name: Rhett M Rowe
- Emp ID: 631156, Full Name: Shirleen Y Rowe
- Emp ID: 657000, Full Name: Angel Y Rowe
- Emp ID: 667295, Full Name: Edmund W Rowe
- Emp ID: 689557, Full Name: Daria Y Rowe
- Emp ID: 703373, Full Name: Beatrice H Rowe
- Emp ID: 771556, Full Name: Shaina E Rowe
- Emp ID: 783799, Full Name: Enoch K Rowe
- Emp ID: 808221, Full Name: Lane O Rowe
- Emp ID: 926265, Full Name: Ramonita M Rowe
- Emp ID: 959340, Full Name: Jacquelyne W Rowe

DB: With primary index on "Emp ID" column but defined as clustered with page size of 4KB
Query and print the full name of employee #181162 (this is an Equality search)

- Full Name: Cristobal C Foy

DB: With primary index on "Emp ID" column but defined as clustered with page size of 4KB
Query and print the employee id and full name of all employees with "Emp ID" between #171800 and #171899 (This is a Range search)

- Emp ID: 171812, Full Name: Alise P Earnhardt
- Emp ID: 171814, Full Name: Junita N Schoenberg
- Emp ID: 171815, Full Name: Elenor U Kivett
- Emp ID: 171816, Full Name: Corie P Alcorn
- Emp ID: 171818, Full Name: Barney U Kirschbaum
- Emp ID: 171823, Full Name: Ivana G Pool

- Emp ID: 171826, Full Name: Winnie Q Mass
- Emp ID: 171832, Full Name: Deshawn Q Tavarez
- Emp ID: 171834, Full Name: Otis C Mcrae
- Emp ID: 171839, Full Name: Valentin Q Glassman
- Emp ID: 171841, Full Name: Deon I Hallam
- Emp ID: 171843, Full Name: Apryl Y Tavera
- Emp ID: 171844, Full Name: Mika O Bradshaw
- Emp ID: 171845, Full Name: Sanora Y Hardwick
- Emp ID: 171847, Full Name: Gwyn F Goodrum
- Emp ID: 171849, Full Name: Bobby F Reep
- Emp ID: 171851, Full Name: Ashely D Flavin
- Emp ID: 171852, Full Name: Shemika Y Cutts
- Emp ID: 171855, Full Name: Vanessa L Philpott
- Emp ID: 171856, Full Name: Emmitt F Olmos
- Emp ID: 171858, Full Name: Mason C Burchette
- Emp ID: 171859, Full Name: Katelyn K Moir
- Emp ID: 171860, Full Name: Warren M Hicklin
- Emp ID: 171865, Full Name: Maximo A Failla
- Emp ID: 171867, Full Name: Hobert G Saleh
- Emp ID: 171868, Full Name: Dana E Goosby
- Emp ID: 171869, Full Name: Karl V Hooley
- Emp ID: 171874, Full Name: Elba X Finnegan
- Emp ID: 171875, Full Name: Aubrey D Bounds
- Emp ID: 171879, Full Name: Darell V Motes
- Emp ID: 171881, Full Name: Anamaria B Buzzell
- Emp ID: 171885, Full Name: Dudley R Kelemen
- Emp ID: 171886, Full Name: Rosalee Q Rutz
- Emp ID: 171887, Full Name: Isis N Dusek
- Emp ID: 171889, Full Name: Collene O Spino
- Emp ID: 171892, Full Name: Porter G Sapien
- Emp ID: 171895, Full Name: Lyle D Weedman
- Emp ID: 171896, Full Name: Joel P Berrier