Project Report - Implementing a Heap File

Arpan Banerjee

UFID: 9359-9083

[arpanbanerjee@ufl.edu](mailto:arpanbanerjee@ufl.edu)

Krutantak Patil

UFID: 5615-6343

Krutantakb.patil@ufl.edu

***1) Steps to compile and run - code, tests and gtests***

1. **make main** – Command to build the main program.
2. **./main** – Command to run the main executable.
3. **make test.out** – Command to build the tests.
4. **./test.out** – Command to run the tests.
5. **make DBFileTest.out** – Command to build the gtests.
6. **./ DBFileTest.out** – Command to run the gtests.

***2) DBFile implementation***

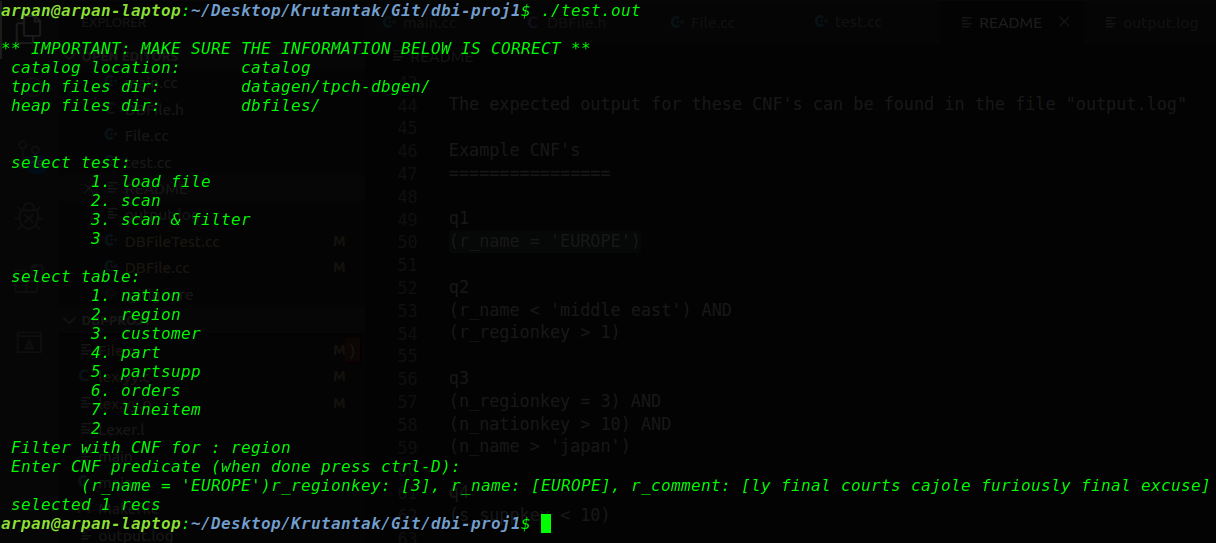
**Data members**

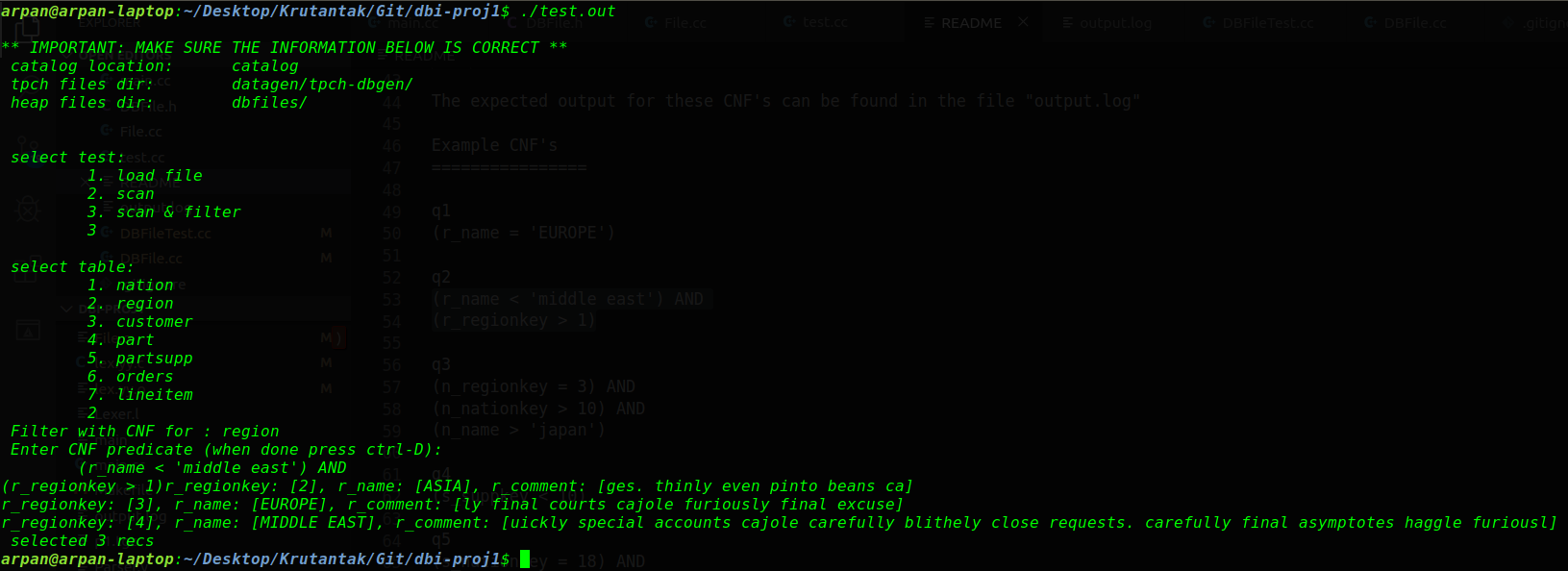
1. char \*filename – Used to copy filename in Create and Open functions.
2. File \*heapfile – Used to store pointer to the heapfile object used to store records.
3. Page \*deltapage – Pointer to page holding recent record additions to the file.
4. Page \*readpage – Pointer to page for reading records from the file.
5. ComparisonEngine comp – Used to apply and check CNF with a record
6. int readPageNumber – Stores page number of current page being read.

**Member functions**

1. void MoveFirst () – This function resets the *readPageNumber* variable and empties the *readPage*.
2. [private function] void AddDeltaPageToFile() – This function adds the *deltapage* to the *heapfile* and clears it out for further additions.
3. void Add (Record &record) – This function adds a record to the *deltapage*. If the *deltapage* is full, it calls *AddDeltaPageToFile()* and adds the record to the fresh *deltapage*.
4. int GetNext (Record &fetchme) ) – This function gets the next record from the file by reading from the *readpage* or incrementing the *readPage* if it runs out. If we reach the end of the file and *deltaPage* contains records, then it is added to the file and read.
5. int GetNext (Record &fetchme, CNF &cnf, Record &literal)) – This function repeatedly calls *GetNext* and checks the record against the CNF.
6. int Create (const char \* f\_path, fType f\_Type, void \*startup) ) – This function creates a DBFile of the type heap. Returns 1 on success.
7. int Open (const char \*f\_path) ) – This function opens an existing DBFile. Returns 1 on success and 0 on failure or when File not found.
8. int Close ()) – This function calls *AddDeltaPageToFile()* and then closes the DBFile.
9. void Load (Schema &file\_schema, const char \*recPath) ) – This function loads data from a given file path. It repeatedly calls *SuckNextRecord()* and then *Add()* to add it to the DBFile.

***3) Test.out results for 1GB data***

1. **Query 1 :** (r\_name = 'EUROPE')
2. **Query 2 :** (r\_name < 'middle east') AND (r\_regionkey > 1)

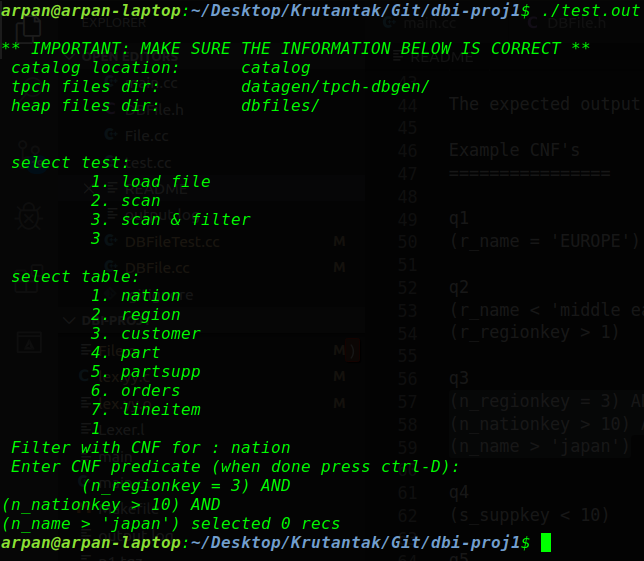


1. **Query 3 :**

(n\_regionkey = 3) AND

(n\_nationkey > 10) AND

(n\_name > 'japan')



1. **Query 11 :**

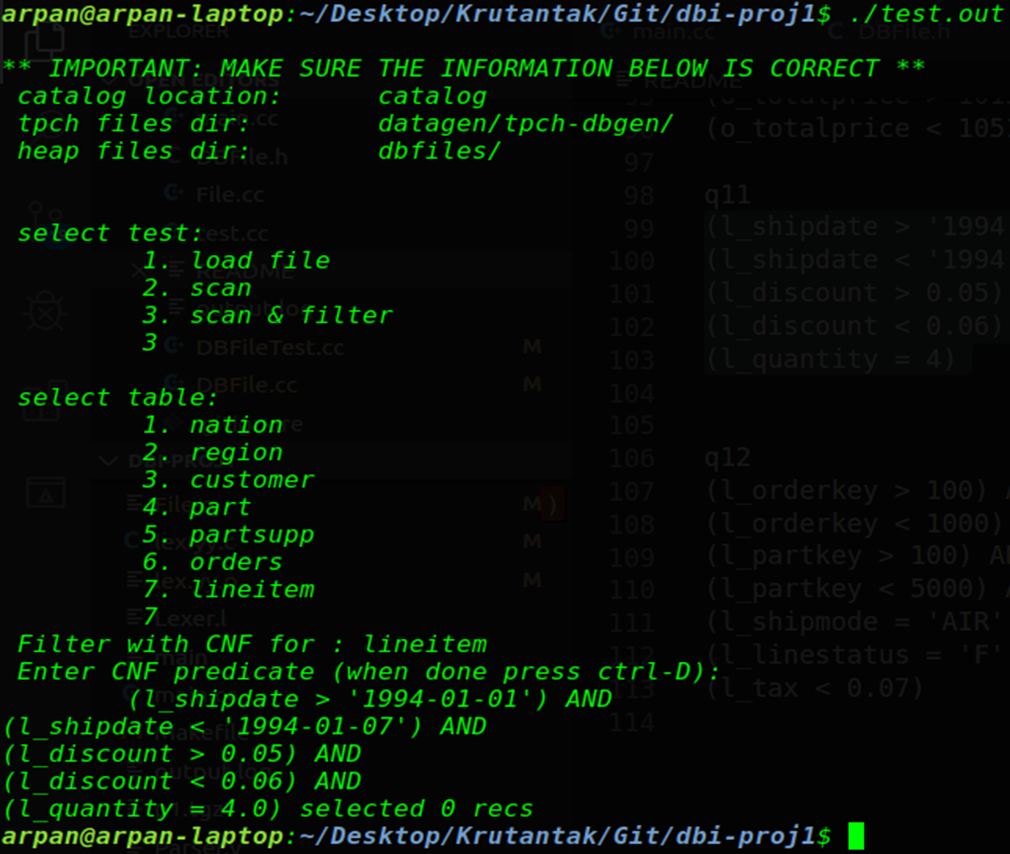
(l\_shipdate > '1994-01-01') AND

(l\_shipdate < '1994-01-07') AND

(l\_discount > 0.05) AND

(l\_discount < 0.06) AND

(l\_quantity = 4)



1. **Query 12 :**

(l\_orderkey > 100) AND

(l\_orderkey < 1000) AND

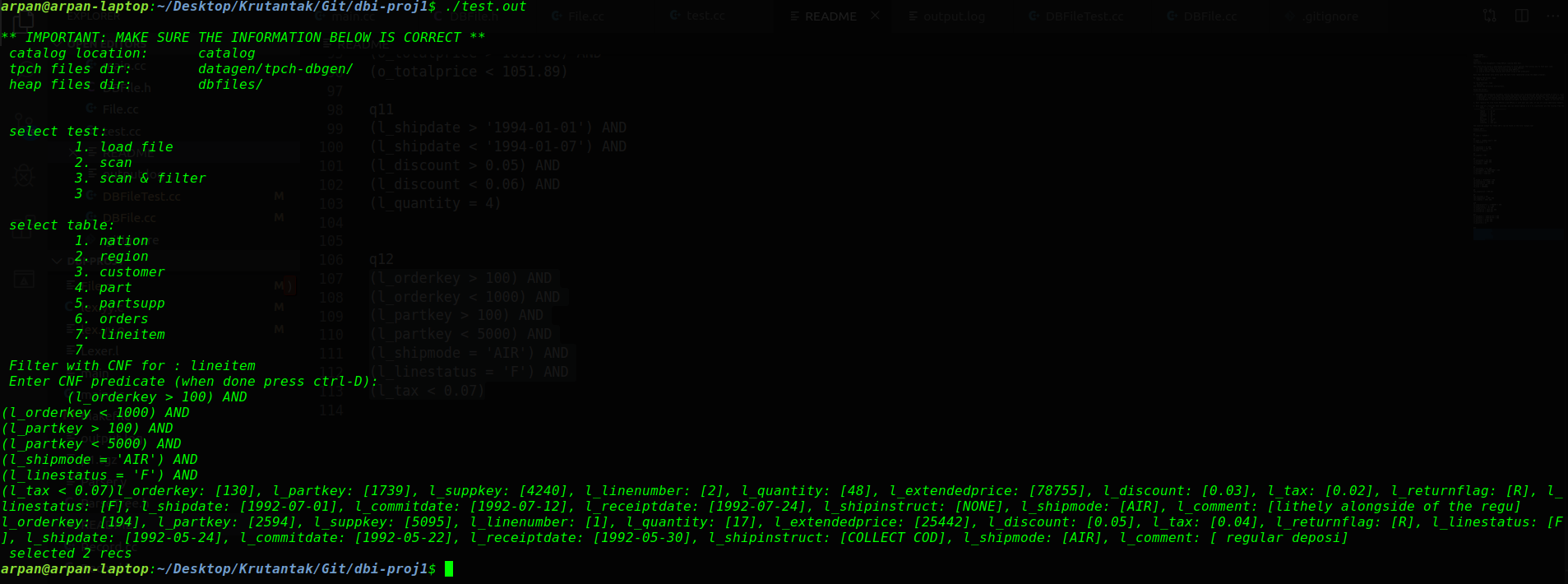
(l\_partkey > 100) AND

(l\_partkey < 5000) AND

(l\_shipmode = 'AIR') AND

(l\_linestatus = 'F') AND

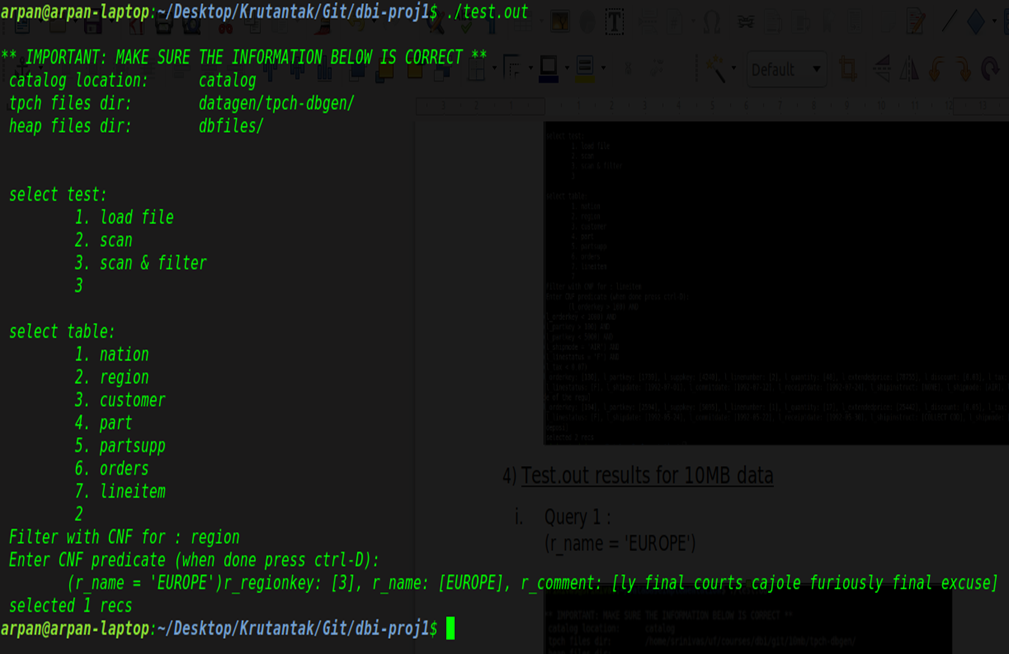
(l\_tax < 0.07)



***4) Test.out results for 10MB data***

1. **Query 1 :**

(r\_name = 'EUROPE')



1. **Query 2 :**

(r\_name < 'middle east') AND

(r\_regionkey > 1)

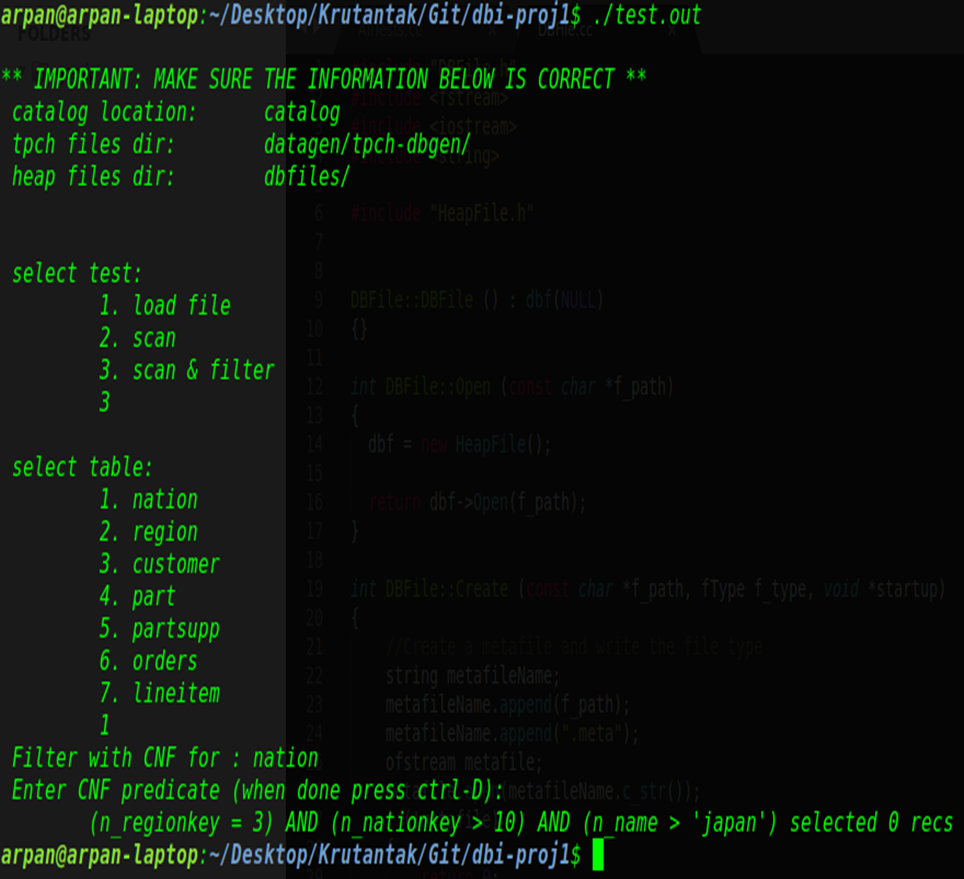


1. **Query 3 :**

(n\_regionkey = 3) AND

(n\_nationkey > 10) AND

(n\_name > 'japan')



1. **Query 11 :**

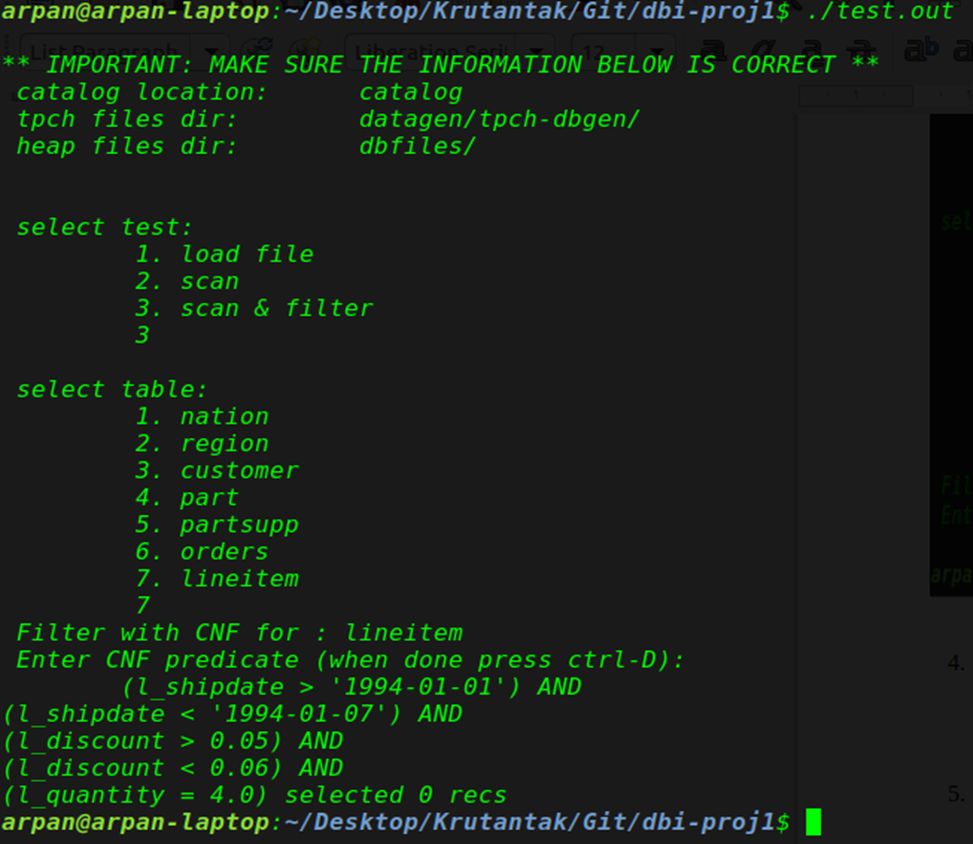
(l\_shipdate > '1994-01-01') AND

(l\_shipdate < '1994-01-07') AND

(l\_discount > 0.05) AND

(l\_discount < 0.06) AND

(l\_quantity = 4.0)



1. **Query 12 :**

(l\_orderkey > 100) AND

(l\_orderkey < 1000) AND

(l\_partkey > 100) AND

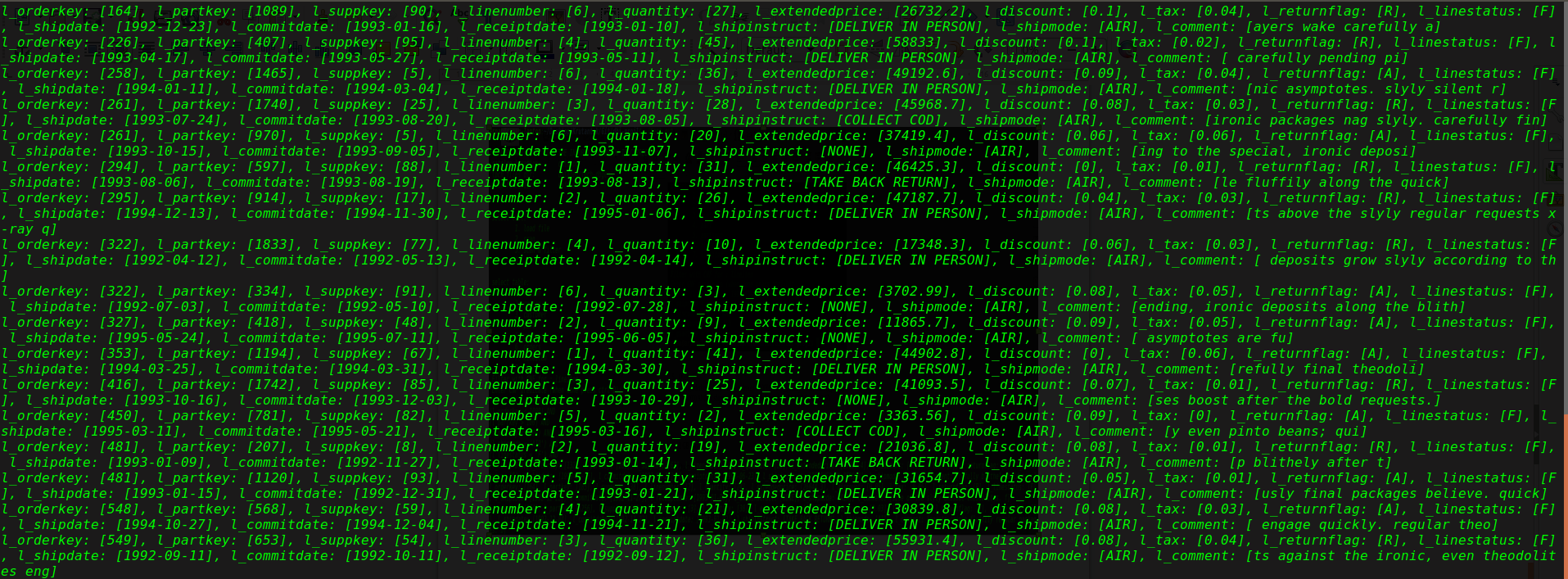
(l\_partkey < 5000) AND

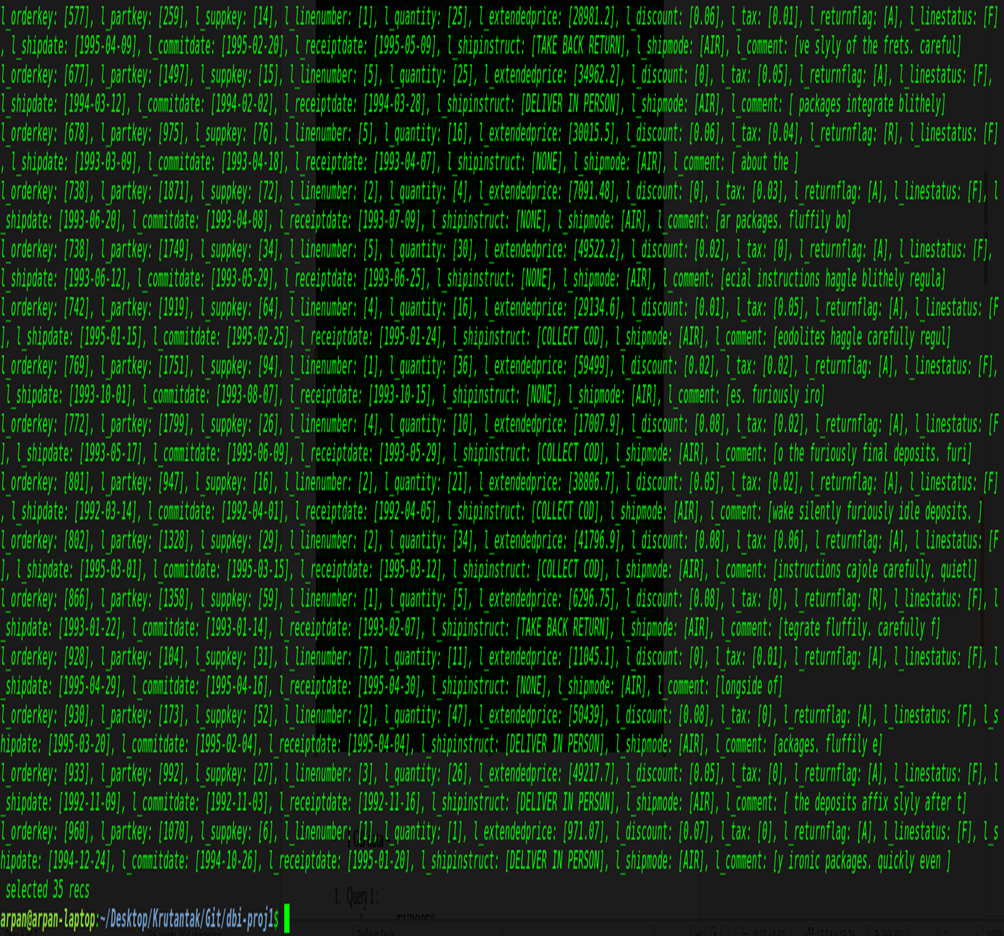
(l\_shipmode = 'AIR') AND

(l\_linestatus = 'F') AND

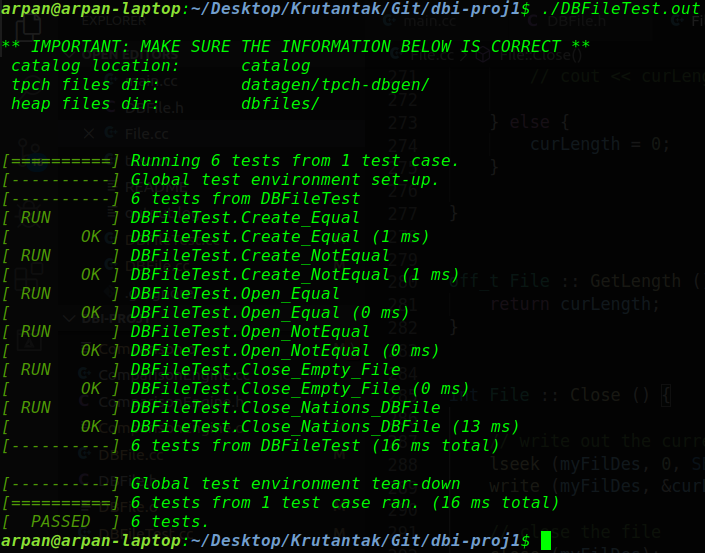
(l\_tax < 0.07)







***5) gTest results:***

1. **TEST(DBFileTest, Create\_Equal)** –
   1. Check return value is 1 on successful creation of file.
2. **TEST(DBFileTest, Create\_NotEqual)** –
   1. Check program terminates with error message “BAD! Open did not work for …” on passing empty file path function argument.
   2. Check return value is 0 on passing wrong file type function argument.
3. **TEST(DBFileTest, Open\_Equal)** –
   1. Check return value is 1 on successfully opening file at the specified file path.
4. **TEST(DBFileTest, Open\_NotEqual)** –
   1. Check return value is 0 on passing incorrect file name.
5. **TEST(DBFileTest, Close\_Emty\_File)** –
   1. Check return value is 0 on closing files that are not loaded or opened.
6. **TEST(DBFileTest, Close\_ Nations\_DBFile)** –
   1. Check return type is not 0 on closing non empty opened files.