API Documentation

Database class:

Database()

* database constructor that creates the empty database
* No inputs are needed for this method and returns no values

Database(const Database& d)

* database copy constructor that initializes the database using a database object
* input for this method is an Object of type Database
* Returns no values

~Database()

* Database destructor
* No inputs are needed and returns no values

void add\_table(Table object, std::string name)

* adds the table to the database
* Requires 2 inputs, first is a Table object second is the name of the table in the type of a string
* Returns no values

void drop\_table(std::string name)

* drops the table from the database
* input is the name of the table to be dropped from the database as a string
* returns no values

vector<std::string> list\_table()

* gets the list of tables from the database
* takes in no inputs
* returns a vector of type string

vector<Table> get\_table()

* gets all of the tables that are currently in the database
* takes in no input and returns a vector of type Table

Table query\_command(std::string select, std::string from, std::string where)

* Query function that can help with selection and finding of the databases
* returns table with queried clauses

Table Class:

int \_size

* gives size of table

std::deque<Record>records

* keeps track of the records in the table

typedef std::vector<std::pair<std::string, Record>>Table\_column

* keeps the records and

names associated with the records. Type def

std::vector<Table\_column> Table\_vec

* Keeps track of the rows of the tables

Table\_column column\_list

* the actual columns

std::vector<std::string> keys

* keeps track of keys

typedef std::deque<Record>::const\_iterator Table\_iterator

* keeps track of the position in thetable

Table\_iterator begin() const

* function that returns the beginning of the table
* no input is needed
* returns a Table\_iterator

Table\_iterator end() const

* function that returns the end of the table
* no input is needed
* returns a Table\_iterator

Table()

* table constructor initializes the Table
* Requires no input and has no output

Table(const Table& t)

* table copy constructor that initialized a table using a Table object
* Takes in one input that is of Table type
* Returns no values

Table(int size)

* initializes a table with a specified size
* Takes in one input of type int which serves at the size of the desired table
* Returns no values

Table(const Table\_column& columns\_list)

* initializes a table with certain columns
* takes in an input of type Table\_column
* returns no values

Table(std::string, std::string, std::string)

* constructor that creates a table with 3 attributes
* takes in 3 inputs of type string each input being the attribute name
* returns no values

void add\_attribute(std::string table\_attribute)

* adds an attribute to the table
* takes in 1 input of type string which is the name of the attribute
* returns no values

void delete\_attribute(std::string table\_attribute)

* removes the table attribute given the name of the attribute
* takes in one input of type string
* no values are returned

std::vector<std::string> get\_attribute\_in\_order()

* gets attributes of the table in order
* requires no inputs and returns a vector of string

int get\_size()

* gets size of table
* requires no input and returns a value of type int

void specify\_key(std::string table\_key)

* finds key in the table given the table key
* Takes in 1 input of type string
* Returns no values

Table cross\_join(Table a, Table b)

* joins two tables and creates one Table
* Takes in two inputs of type Table
* Returns a value of type Table

Table natural\_join(Table a, Table b)

* joins two tables based on key and returns one Table
* Takes in two inputs of type table
* Returns a value of type Table

int compute\_count(std::string table\_attribute)

* Computes a count on non-null entries in table on an attribute from the table
* Takes in one input of type string which is the name of the attribute to do the count
* Returns a value of type int

int compute\_min(std::string table\_attribute)

* finds minimum record on an attribute from the table
* takes in one input of type string which is the name of the attribute to find the min
* Returns a value of type int

int compute\_max(std::string table\_attribute)

* finds maximum record on an attribute from the table
* takes in one input of type string which is the name of the attribute to find the max
* returns a value of type int

Table add(Record r)

* adds record to table and returns table
* Takes in a input of type Record
* Returns an object of type table

Record Class:

Record()

* record constructor; initializes a record
* Takes in no input and returns no values

Record(int size)

* Creates a record with a specified size
* Takes in an input of type int; this is the size of the record to be created
* Returns no values

Record(const Record& r)

* record copy constructor; initialized a record using a record object
* requires one input of type Record
* returns no values

int size()

* returns the size of the record container
* takes in no inputs
* returns a value of type int

~Record()

* Destructor
* Takes in no input and returns no values

std::string get(unsigned int index)

* get a string at an index from the record
* takes in one input of type int; this is the index of the record
* returns a value of type string which is the value of the record

void set(int index, std::string name)

* used to set a a record given the index and the name of the record to create
* takes in 2 inputs, first is a value of type int, second is a value of type string
* returns no values