W4111_HW2_Programming

October 24, 2021

COMS W4111-002 (Fall 2021)Introduction to Databases

Homework 2: Programming Implement a Simple Database Engine15 Points

0.1 This assignment is due October 22, 11:59 pm EDT

Note: Please replace the information below with your last name, first name and UNI.

<span style="font-size: 20pt; line-height: 1.2"; > Suri_Chandan, CS4090

0.1.1 Submission

1. File > Print Preview > Save as PDF…
2. Upload .pdf and .ipynb to GradeScope

This assignment is due October 22, 11:59 pm EDT

0.1.2 Collaboration

- You may use any information you get in TA or Prof. Ferguson's office hours, from lectures or from recitations.
- You may use information that you find on the web.
- You are NOT allowed to collaborate with other students outside of office hours.

0.2 Part 1: Written & SQL

0.2.1 Written

Please keep your answers brief.

1. Codd's Fourth Rule states that: The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data. In two sentences please explain this rule and why it is so important.
2. Give 3 examples of what would be stored in a databsae catalog
3. What is the SQL database catalog called?

0.2.2 Solution 1. 2. & 3.

1. In the case of SQL, this metadata (data about the data in the database) should also be stored as tables, rows and columns following a similar structure as that of the regular data. Also, this

metadata should obey all the characteristics of the databases including access privileges if needed. Moreover, the most important thing about this metadata is that it should be accessible by using the same query language that we use to access the database. Lastly, this is a very important rule because the metadata is really important to get the information about the complete database, and without this rule any database management system cannot be considerred to be relational as the metadata should be stored in the same form as regular data and should be accessible by the same query language, keeping it uniform for the users to access any kind of data when working with the databases.

2. Some of the things that could be stored in the database catalog are as follows: a. Information about the relations (tables) such as names of the tables, information about different attributes in each relation/table, names and definitions of views, and integrity constraints associated with each of those tables. b. Authorization details: Users, roles and accounting information, inlcuding passwords and information like who can access what from the database. c. Physical file organization information: How the tables are stored, and what is the physical location of a table/relation.

3. The (SQL Server) SQL database catalog is also known as "System Catalog". In MySQL, it's also known as "Information Schema".

4. What is the overall goal of indicies in SQL?

5. What are the differences between a primary key and a unique index?

0.2.3 Solution 4. & 5.

4. An index on an attribute of a relation is a data structure that allows the database system to find some tuples in a relation efficiently, without scanning through all the tuples in a relation. Precisely, as many queries reference only a small portion of the records in the table, it is very inefficient to read every record in order to find a particular one; using the indices, we don't need to go over all the records to find some matching tuples according to some attributes and we can get them really fast and in an optimized way. Moreover, in the case of primary keys in mySQL, an index is created for the primary and foreign keys in a table which helps the users access the records according to those keys really fast and efficiently.

5. a. Unique Index: It refers to an index where all the rows must be unique and there can be multiple unique indices per table or a relation. On the other hand, a primary key is always just one attribute of the table that must be unique. b.While, unique indices are not used to uniquely identify any row/s in a table and can have NULL values, on the other hand, primary key (or index) related to a table is used to uniquely identify any row/s in a table and thus, is not allowed to have any NULL values. Thus, the attributes or columns as a part of the primary key should be non-nullable. c. Primary key by itself doesn't enable fast searching and optimized query run, although primary index does that job. In the case of unique index, as it's an index, it enables the fast searching for finding a record or a set of records based on that attribute (unique, can be set of columns too). Reference: https://stackoverflow.com/questions/707874/differences-between-index-primary-unique-fulltext-in-mysql

6. Which SELECT statement is more efficient? Why?

- SELECT playerID,birthState,nameLast,nameFirst FROM people where birthCountry = 'USA' and nameFirst = 'John' and playerID in (select playerID from collegeplaying where schoolID = 'Fordham');

- SELECT playerID,birthState,nameLast,nameFirst FROM people NATURAL JOIN collegeplaying where birthCountry = 'USA' and nameFirst = 'John' and schoolID = 'Fordham' group by playerID,birthState,nameLast,nameFirst;

HINT: SQL uses a query optimizer so you can't just run both of these and see which one performs faster.

The SELECT statement with NATURAL JOIN would be less efficient as compared to the one with a sub-query. Primarily, this is because of how a subquery within a query is executed and how the natural join works. For the above query with a sub-query, the sub-query runs first over the table 'collegeplaying' to find all the playerIDs with a schoolID as "Fordham". Now, the data returned here is matched with the playerIDs from the table 'people' to give us the final result. Basically, in this case, we run over the data for 2 tables separately to find the result and considering the playerID column, if index is there for the attribute, it will all the more faster. On the other hand, for the SELECT statement with NATURAL JOIN, we first need to combine all the records of the 2 tables 'people' and 'collegeplaying' according to the playerID column and return the result after grouping by the columns which need to be a part of the result. If we don't take indexing and other optimizations into consideration, firstly, we need to join the 2 tables across some columns which would take up a lot of time, and then we need to iterate over the complete set of rows to find the matching rows from the joined table. This would take a lot more time! Assumption: I am taking into consideration that no indexing and optimization are applicable for both the queries above. In that case, the NATURAL JOIN should take more time in comparison to the the one with a sub-query! Thus, the SELECT statement with a natural join is less efficient in comparison to the one with a sub-query.

7. The create.sql file provided in the zip folder makes a schema and some tables that mimics metadata tables. Note there is the sytax "ON DELETE CASCADE" after the foreign key creation. What does this mean? Why do we want to specify CASCADE for the metadata tables? What does "ON DELETE RESTRICT" mean and when would we generally want to use this?

The "ON DELETE CASCADE" precisely means that if some particular record corresponding to the parent table is deleted, which contains the column that was referenced by another table with a foreign key constraint along with the CASCADE statement mentioned above, then the record corresponding to that key will also be deleted from the table in which that foreign key constraint has been added and so on and so forth. It goes recursively to all the tables where that constraint on that column has been added. For instance, in the csvcolumns table, a foreign key constraint has been added to the table name column in the csvolumns table and references the table name column in the csvtables along with the "ON DELETE CASCADE". In this case, if an entry named 'table1' was added to the csvtables and the same was present in csvcolumns table too, if we delete the entry from the csvtables, the corresponding entry or entries in the csvcolumns having table name as 'table1' will also be deleted. As we also have a constraint added on that column in the 'csvindexes' table that references 'csvcolumns' table, the entry or entries in the csvindexes having table name as 'table1' will also be deleted. Also, if all the entries having table name 'table1' was deleted from the csvcolumns table, that wouldn't delete any entry from the csvtables column as the "ON DELETE CASCADE" was added to the csvcolumns table as it works in a top-down fashion. Thus, any entry or entries having table name as 'table1' will be deleted from the csvindexes table though as we had added the foreign key constraint with "ON DELETE CASCADE" in the csvindexes table too. If we do not specify cascading deletes, the default behavior of the database server prevents us from deleting data in a table if other tables reference it. So, that's what the "ON DELETE CASCADE"

means. Coming to "ON DELETE RESTRICT" after the foreign key creation, that precisely means that we can't delete a given parent row if a child row exists that references the value for that parent row. If the parent row has no referencing child rows, then we can delete that parent row. The child row here corresponds to the row in the table which has the foreign key constraint added on column/s in that table that references the columns from another table which is the one having the parent row in there. Furthemore, in this case, "ON DELETE RESTRICT" would thus be added to the table having the child rows referencing the column/s in the parent table. One more thing is that this is the normal behaviour for a foreign key even if you don't add the "ON DELETE RESTRICT" with the foreign key constraint. So, it's kind of a superfluous syntax and is not very much needed. If you want to explicitly state such a behaviour for a foreign key constraint, then we might add it but generally it's not needed with database engines like mySQL.

0.2.4 SQL

[10]:
```
%load_ext sql
```

[11]:
```
%sql mysql+pymysql://root:dbuserdbuser@localhost/lahmansbaseballdb
```

[11]:
```
'Connected: root@lahmansbaseballdb'
```

1. Initials

- Find the `initials, firstName, lastName`, for every player from the people table.
- You need to return 10 rows.
- Sort by the nameFirst, nameLast ascending.
- Note: Even for those players with two last names, just return the first letter of their first last name

Answer:

[12]:
```
%%sql
SELECT CONCAT(SUBSTRING(nameFirst, 1, 1), SUBSTRING(nameLast, 1, 1)) as␣
 ↪initials,
    nameFirst as firstName, nameLast as lastName from people
    WHERE nameFirst is not NULL AND nameLast is not NULL
    ORDER BY firstName, lastName ASC LIMIT 10;
```

```
 * mysql+pymysql://root:***@localhost/lahmansbaseballdb
10 rows affected.
```

[12]:
```
[('AA', 'A. J.', 'Achter'),
 ('AB', 'A. J.', 'Burnett'),
 ('AC', 'A. J.', 'Cole'),
 ('AE', 'A. J.', 'Ellis'),
 ('AG', 'A. J.', 'Griffin'),
 ('AH', 'A. J.', 'Hinch'),
 ('AJ', 'A. J.', 'Jimenez'),
```

```
 ('AM', 'A. J.', 'Minter'),
 ('AM', 'A. J.', 'Morris'),
 ('AM', 'A. J.', 'Murray')]
```

0.3 Question 1a): Games Per Player using GROUP BY

- Find the `yearID, lgID, games_per_player`, for every year and league from the appearances table.
- Use a function to round down the games_per_player
- You need to return 10 rows.
- You must use `group by` in this query.

Answer:

[13]:
```
%%sql
SELECT yearID, lgID, ROUND(SUM(G_all)/COUNT(playerID), 2) as games_per_player
from appearances
GROUP BY yearID, lgID
LIMIT 10;
```

```
 * mysql+pymysql://root:***@localhost/lahmansbaseballdb
10 rows affected.
```

[13]:
```
[(1871, 'NA', Decimal('19.97')),
 (1872, 'NA', Decimal('20.61')),
 (1873, 'NA', Decimal('28.83')),
 (1874, 'NA', Decimal('34.14')),
 (1875, 'NA', Decimal('28.74')),
 (1876, 'NL', Decimal('37.87')),
 (1877, 'NL', Decimal('33.82')),
 (1878, 'NL', Decimal('41.49')),
 (1879, 'NL', Decimal('45.63')),
 (1880, 'NL', Decimal('45.61'))]
```

0.4 Part 2: CSVCatalog Tests

Once you have tested everything successfuly in python, execute your tests one more time in jupyter notebook to show the expected output. You will need to restart your kernel after saving your python files so that jupyter will use the most recent version of your work.

You may need to drop tables before executing your tests one last time so you don't run into integrity errors

[7]:
```
import unit_test_catalog as cat # This notebook should be in the same directory␣
↪as your project
```

[8]:
```
cat.create_table_test()
```

```
Adding test_table_1 with file name test_path_file_1.csv…
Running save core definition
Q =  insert into csvtables values(%s, %s)
Q =  select * from csvtables where table_name = 'test_table_1'
Q =  select * from csvcolumns WHERE table_name = 'test_table_1'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_1'
Table =  in the index if statement
{
  "table_name": "test_table_1",
  "file_name": "test_path_file_1.csv",
  "columns": [],
  "indexes": []
}
Adding test_table_2 with file name test_path_file_2.csv…
Running save core definition
Q =  insert into csvtables values(%s, %s)
Q =  select * from csvtables where table_name = 'test_table_2'
Q =  select * from csvcolumns WHERE table_name = 'test_table_2'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_2'
Table =  in the index if statement
{
  "table_name": "test_table_2",
  "file_name": "test_path_file_2.csv",
  "columns": [],
  "indexes": []
}
Adding test_table_3 with file name test_path_file_3.csv…
Running save core definition
Q =  insert into csvtables values(%s, %s)
Q =  select * from csvtables where table_name = 'test_table_3'
Q =  select * from csvcolumns WHERE table_name = 'test_table_3'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_3'
Table =  in the index if statement
{
  "table_name": "test_table_3",
  "file_name": "test_path_file_3.csv",
  "columns": [],
  "indexes": []
}
Adding test_table_4 with file name test_path_file_4.csv…
Running save core definition
Q =  insert into csvtables values(%s, %s)
Q =  select * from csvtables where table_name = 'test_table_4'
Q =  select * from csvcolumns WHERE table_name = 'test_table_4'
Q =  select column_name, index_name, type, index_order from csvindexes where
```

```
table_name = 'test_table_4'
Table =  in the index if statement
{
  "table_name": "test_table_4",
  "file_name": "test_path_file_4.csv",
  "columns": [],
  "indexes": []
}
Adding test_table_5 with file name test_path_file_5.csv…
Running save core definition
Q =  insert into csvtables values(%s, %s)
Q =  select * from csvtables where table_name = 'test_table_5'
Q =  select * from csvcolumns WHERE table_name = 'test_table_5'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_5'
Table =  in the index if statement
{
  "table_name": "test_table_5",
  "file_name": "test_path_file_5.csv",
  "columns": [],
  "indexes": []
}
--------------------------------------------------------------------------------
---
```

[9]:
```
cat.drop_table_test()
```

```
Dropping test_table_4 …
Q =  DELETE FROM csvtables WHERE table_name = 'test_table_4'
Table 'test_table_4' was dropped
Dropping test_table_5 …
Q =  DELETE FROM csvtables WHERE table_name = 'test_table_5'
Table 'test_table_5' was dropped
--------------------------------------------------------------------------------
---
```

[10]:
```
cat.add_column_test()
```

```
Q =  select * from csvtables where table_name = 'test_table_1'
Q =  select * from csvcolumns WHERE table_name = 'test_table_1'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_1'
Q =  insert into csvcolumns values(%s, %s, %s, %s)
in the index if statement
For table {
  "table_name": "test_table_1",
  "file_name": "test_path_file_1.csv",
  "columns": [
    {
```

```
      "column_name": "t1_tc_1",
      "column_type": "text",
      "not_null": true
    }
  ],
  "indexes": []
}, Column Definition Added:
 {
  "column_name": "t1_tc_1",
  "column_type": "text",
  "not_null": true
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
in the index if statement
For table {
  "table_name": "test_table_1",
  "file_name": "test_path_file_1.csv",
  "columns": [
    {
      "column_name": "t1_tc_1",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "t1_tc_2",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": []
}, Column Definition Added:
 {
  "column_name": "t1_tc_2",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
in the index if statement
For table {
  "table_name": "test_table_1",
  "file_name": "test_path_file_1.csv",
  "columns": [
    {
      "column_name": "t1_tc_1",
      "column_type": "text",
      "not_null": true
    },
    {
```

```
      "column_name": "t1_tc_2",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "t1_tc_3",
      "column_type": "number",
      "not_null": false
    }
  ],
  "indexes": []
}, Column Definition Added:
 {
  "column_name": "t1_tc_3",
  "column_type": "number",
  "not_null": false
}
Q =  select * from csvtables where table_name = 'test_table_2'
Q =  select * from csvcolumns WHERE table_name = 'test_table_2'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_2'
Q =  insert into csvcolumns values(%s, %s, %s, %s)
in the index if statement
For table {
  "table_name": "test_table_2",
  "file_name": "test_path_file_2.csv",
  "columns": [
    {
      "column_name": "t2_tc_1",
      "column_type": "text",
      "not_null": true
    }
  ],
  "indexes": []
}, Column Definition Added:
 {
  "column_name": "t2_tc_1",
  "column_type": "text",
  "not_null": true
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
in the index if statement
For table {
  "table_name": "test_table_2",
  "file_name": "test_path_file_2.csv",
  "columns": [
    {
      "column_name": "t2_tc_1",
```

```
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "t2_tc_2",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": []
}, Column Definition Added:
 {
  "column_name": "t2_tc_2",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
in the index if statement
For table {
  "table_name": "test_table_2",
  "file_name": "test_path_file_2.csv",
  "columns": [
    {
      "column_name": "t2_tc_1",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "t2_tc_2",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "t2_tc_3",
      "column_type": "number",
      "not_null": true
    }
  ],
  "indexes": []
}, Column Definition Added:
 {
  "column_name": "t2_tc_3",
  "column_type": "number",
  "not_null": true
}
Q =  select * from csvtables where table_name = 'test_table_3'
Q =  select * from csvcolumns WHERE table_name = 'test_table_3'
Q =  select column_name, index_name, type, index_order from csvindexes where
```

```
table_name = 'test_table_3'
Q =  insert into csvcolumns values(%s, %s, %s, %s)
in the index if statement
For table {
  "table_name": "test_table_3",
  "file_name": "test_path_file_3.csv",
  "columns": [
    {
      "column_name": "t3_tc_1",
      "column_type": "number",
      "not_null": true
    }
  ],
  "indexes": []
}, Column Definition Added:
 {
  "column_name": "t3_tc_1",
  "column_type": "number",
  "not_null": true
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
in the index if statement
For table {
  "table_name": "test_table_3",
  "file_name": "test_path_file_3.csv",
  "columns": [
    {
      "column_name": "t3_tc_1",
      "column_type": "number",
      "not_null": true
    },
    {
      "column_name": "t3_tc_2",
      "column_type": "text",
      "not_null": true
    }
  ],
  "indexes": []
}, Column Definition Added:
 {
  "column_name": "t3_tc_2",
  "column_type": "text",
  "not_null": true
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
in the index if statement
For table {
  "table_name": "test_table_3",
```

```
  "file_name": "test_path_file_3.csv",
  "columns": [
    {
      "column_name": "t3_tc_1",
      "column_type": "number",
      "not_null": true
    },
    {
      "column_name": "t3_tc_2",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "t3_tc_3",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": []
}, Column Definition Added:
 {
  "column_name": "t3_tc_3",
  "column_type": "text",
  "not_null": false
}
--------------------------------------------------------------------------------
---
```

[11]:
```
cat.column_name_failure_test()  # This will throw an error
```

```
issue!!
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/p9/cs27515x0b7g56k1qf9hh5580000gn/T/ipykernel_2667/1013433648.py i
 ↪<module>
----> 1 cat.column_name_failure_test()  # This will throw an error

~/Assignments/Introduction to DBs/Assignment2/unit_test_catalog.py in
 ↪column_name_failure_test()
     85         db="CSVCatalog")
     86
---> 87     col = CSVCatalog.ColumnDefinition(None, "text", False)
     88     t = cat.get_table("test_table")
     89     t.add_column_definition(col)

~/Assignments/Introduction to DBs/Assignment2/CSVCatalog.py in __init__(self,
 ↪column_name, column_type, not_null)
```

```
     47         if column_name == None:
     48             print("issue!!")
---> 49             raise ValueError('You must have a column name!!')
     50         else:
     51             self.column_name = column_name

ValueError: You must have a column name!!
```

[12]:
```
cat.column_type_failure_test()  # This will throw an error
```

```
Issue!
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/p9/cs27515x0b7g56k1qf9hh5580000gn/T/ipykernel_2667/2600232317.py i␣
 ↪<module>
----> 1 cat.column_type_failure_test()  # This will throw an error

~/Assignments/Introduction to DBs/Assignment2/unit_test_catalog.py in␣
 ↪column_type_failure_test()
    103         db="CSVCatalog")
    104
--> 105     col = CSVCatalog.ColumnDefinition("bird", "canary", False)
    106     t = cat.get_table("test_table")
    107     t.add_column_definition(col)

~/Assignments/Introduction to DBs/Assignment2/CSVCatalog.py in __init__(self,␣
 ↪column_name, column_type, not_null)
     55         else:
     56             print("Issue!")
---> 57             raise ValueError('That column type is not accepted. Please␣
 ↪try again.')
     58
     59         if type(not_null)==type(True):

ValueError: That column type is not accepted. Please try again.
```

[13]:
```
cat.column_not_null_failure_test()  # This will throw an error
```

```
issue!
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/p9/cs27515x0b7g56k1qf9hh5580000gn/T/ipykernel_2667/559455694.py in␣
 ↪<module>
----> 1 cat.column_not_null_failure_test()  # This will throw an error
```

```
~/Assignments/Introduction to DBs/Assignment2/unit_test_catalog.py in␣
↪column_not_null_failure_test()
    121         db="CSVCatalog")
    122
--> 123     col = CSVCatalog.ColumnDefinition("name", "text", "happy")
    124     t = cat.get_table("test_table")
    125     t.add_column_definition(col)

~/Assignments/Introduction to DBs/Assignment2/CSVCatalog.py in __init__(self,␣
↪column_name, column_type, not_null)
     61         else:
     62             print("issue!")
---> 63             raise ValueError('The not_null column must be either True o␣
↪False! Please try again.')
     64
     65

ValueError: The not_null column must be either True or False! Please try again.
```

[14]:
```
cat.add_index_test()
```

```
Q =  select * from csvtables where table_name = 'test_table_1'
Q =  select * from csvcolumns WHERE table_name = 'test_table_1'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_1'
Index 't1_PK_index' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Index t1_PK_index for test_table_1 has been added!
For Table test_table_1, Index Definition Added:
 {
  "index_name": "t1_PK_index",
  "type": "PRIMARY",
  "columns": [
    "t1_tc_1"
  ]
}
Q =  select * from csvtables where table_name = 'test_table_2'
Q =  select * from csvcolumns WHERE table_name = 'test_table_2'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_2'
Index 't2_PK_index' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Index t2_PK_index for test_table_2 has been added!
For Table test_table_2, Index Definition Added:
```

```
 {
  "index_name": "t2_PK_index",
  "type": "PRIMARY",
  "columns": [
    "t2_tc_1"
  ]
}
Index 't2_index' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Index t2_index for test_table_2 has been added!
For Table test_table_2, Index Definition Added:
 {
  "index_name": "t2_index",
  "type": "INDEX",
  "columns": [
    "t2_tc_2"
  ]
}
Index 't2_unique_index' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Index t2_unique_index for test_table_2 has been added!
For Table test_table_2, Index Definition Added:
 {
  "index_name": "t2_unique_index",
  "type": "UNIQUE",
  "columns": [
    "t2_tc_3"
  ]
}
Q =  select * from csvtables where table_name = 'test_table_3'
Q =  select * from csvcolumns WHERE table_name = 'test_table_3'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_3'
Index 't3_PK_index' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Index t3_PK_index for test_table_3 has been added!
For Table test_table_3, Index Definition Added:
 {
  "index_name": "t3_PK_index",
  "type": "PRIMARY",
  "columns": [
    "t3_tc_1",
    "t3_tc_2"
```

```
  ]
}
Index 't3_index' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Index t3_index for test_table_3 has been added!
For Table test_table_3, Index Definition Added:
 {
  "index_name": "t3_index",
  "type": "INDEX",
  "columns": [
    "t3_tc_3"
  ]
}
All the indexes were added successfully!!!
--------------------------------------------------------------------------------
---
```

[15]:
```
cat.col_drop_test()
```

```
Q =  select * from csvtables where table_name = 'test_table_1'
Q =  select * from csvcolumns WHERE table_name = 'test_table_1'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_1'
Column 'invalid_column' not found
Q =  select * from csvtables where table_name = 'test_table_2'
Q =  select * from csvcolumns WHERE table_name = 'test_table_2'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_2'
Q =  DELETE FROM csvcolumns WHERE table_name = 'test_table_2' and column_name =
't2_tc_3'
Column 't2_tc_3' has been dropped!
Q =  select * from csvtables where table_name = 'test_table_3'
Q =  select * from csvcolumns WHERE table_name = 'test_table_3'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_3'
Q =  DELETE FROM csvcolumns WHERE table_name = 'test_table_3' and column_name =
't3_tc_3'
Column 't3_tc_3' has been dropped!
Intended Columns have been dropped!!!
--------------------------------------------------------------------------------
---
```

[16]:
```
cat.index_drop_test()
```

```
Q =  select * from csvtables where table_name = 'test_table_1'
Q =  select * from csvcolumns WHERE table_name = 'test_table_1'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_1'
```

```
Index 'invalid_index' not found
Q =  select * from csvtables where table_name = 'test_table_2'
Q =  select * from csvcolumns WHERE table_name = 'test_table_2'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_2'
Q =  DELETE FROM csvindexes WHERE table_name = 'test_table_2' and index_name =
't2_index'
Index 't2_index' has been dropped!
Q =  select * from csvtables where table_name = 'test_table_3'
Q =  select * from csvcolumns WHERE table_name = 'test_table_3'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_3'
Q =  DELETE FROM csvindexes WHERE table_name = 'test_table_3' and index_name =
't3_PK_index'
Index 't3_PK_index' has been dropped!
Intended Indexes have been dropped!!!
------------------------------------------------------------------------------
---
```

[17]:
```
cat.describe_table_test()
```

```
Q =  select * from csvtables where table_name = 'test_table_1'
Q =  select * from csvcolumns WHERE table_name = 'test_table_1'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_1'
in the index if statement
DESCRIBE test_table_1 =
 {
  "table_name": "test_table_1",
  "file_name": "test_path_file_1.csv",
  "columns": [
    {
      "column_name": "t1_tc_1",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "t1_tc_2",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "t1_tc_3",
      "column_type": "number",
      "not_null": false
    }
  ],
  "indexes": [
```

```
    {
      "index_name": "t1_PK_index",
      "type": "PRIMARY",
      "columns": [
        "t1_tc_1"
      ]
    }
  ]
}
Q =  select * from csvtables where table_name = 'test_table_2'
Q =  select * from csvcolumns WHERE table_name = 'test_table_2'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_2'
in the index if statement
DESCRIBE test_table_2 =
 {
  "table_name": "test_table_2",
  "file_name": "test_path_file_2.csv",
  "columns": [
    {
      "column_name": "t2_tc_1",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "t2_tc_2",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": [
    {
      "index_name": "t2_PK_index",
      "type": "PRIMARY",
      "columns": [
        "t2_tc_1"
      ]
    }
  ]
}
Q =  select * from csvtables where table_name = 'test_table_3'
Q =  select * from csvcolumns WHERE table_name = 'test_table_3'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'test_table_3'
in the index if statement
DESCRIBE test_table_3 =
 {
  "table_name": "test_table_3",
```

```
  "file_name": "test_path_file_3.csv",
  "columns": [
    {
      "column_name": "t3_tc_1",
      "column_type": "number",
      "not_null": true
    },
    {
      "column_name": "t3_tc_2",
      "column_type": "text",
      "not_null": true
    }
  ],
  "indexes": []
}
--------------------------------------------------------------------------------
---
```

0.5 Part 3: CSVTable Tests

In the event that the data sent is too large, jupyter notebook will throw a warning and not print any output. This will happen when you try to retrieve an entire table. Don't worry about getting the output if this happens.

Additonally, the table formatting will get messed up if the columns makes the output too wide. In your tests make sure you project fields so that your outputs are legible.

[1]:
```
import unit_test_csv_table as tab
```

[16]:
```
# Drop the tables if you already made them when testing
tab.drop_tables_for_prep()
```

```
Q =  DELETE FROM csvtables WHERE table_name = 'people'
Table 'people' was dropped
Q =  DELETE FROM csvtables WHERE table_name = 'batting'
Table 'batting' was dropped
Q =  DELETE FROM csvtables WHERE table_name = 'appearances'
Table 'appearances' was dropped
--------------------------------------------------------------------------------
---
```

[17]:
```
tab.create_lahman_tables()
```

```
Running save core definition
Q =  insert into csvtables values(%s, %s)
Running save core definition
Q =  insert into csvtables values(%s, %s)
Running save core definition
Q =  insert into csvtables values(%s, %s)
```

```
--------------------------------------------------------------------------------
---
```

[18]:
```
tab.update_people_columns()
```

```
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "playerID",
  "column_type": "text",
  "not_null": true
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "birthYear",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "birthMonth",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "birthDay",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "birthCountry",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "birthState",
```

```
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "birthCity",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "deathYear",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "deathMonth",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "deathDay",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "deathCountry",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "deathState",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
```

```
  "column_name": "deathCity",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "nameFirst",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "nameLast",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "nameGiven",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "weight",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "height",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "bats",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
```

```
 {
  "column_name": "throws",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "debut",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "finalGame",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "retroID",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table people, Column Definition Added:
 {
  "column_name": "bbrefID",
  "column_type": "text",
  "not_null": false
}
--------------------------------------------------------------------------------
---
```

[19]:
```
tab.update_appearances_columns()
```

```
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "yearID",
  "column_type": "text",
  "not_null": true
```

```
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "teamID",
  "column_type": "text",
  "not_null": true
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "lgID",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "playerID",
  "column_type": "text",
  "not_null": true
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_all",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "GS",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_batting",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_defense",
  "column_type": "text",
```

```
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_p",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_c",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_1b",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_2b",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_3b",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_ss",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_lf",
```

```
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_cf",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_rf",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_of",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_dh",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_ph",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table appearances, Column Definition Added:
 {
  "column_name": "G_pr",
  "column_type": "text",
  "not_null": false
}
--------------------------------------------------------------------------------
---
```

[20]:
```
tab.update_batting_columns()
```

```
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "playerID",
  "column_type": "text",
  "not_null": true
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "yearID",
  "column_type": "text",
  "not_null": true
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "stint",
  "column_type": "text",
  "not_null": true
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "teamID",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "lgID",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "G",
  "column_type": "text",
  "not_null": false
}
```

```
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "AB",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "R",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "H",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "2B",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "3B",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "HR",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "RBI",
  "column_type": "text",
  "not_null": false
```

```
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "SB",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "CS",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "BB",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "SO",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "IBB",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "HBP",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "SH",
  "column_type": "text",
```

```
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "SF",
  "column_type": "text",
  "not_null": false
}
Q =  insert into csvcolumns values(%s, %s, %s, %s)
For table batting, Column Definition Added:
 {
  "column_name": "GIDP",
  "column_type": "text",
  "not_null": false
}
--------------------------------------------------------------------------------
---
```

[21]:
```
tab.add_index_definitions()
```

```
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Index 'PK People' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Index PK People for people has been added!
For Table people, Index Definition Added:
 {
  "index_name": "PK People",
  "type": "PRIMARY",
  "columns": [
    "playerID"
  ]
}
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Index 'PK Batting' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
```

```
Index PK Batting for batting has been added!
For Table batting, Index Definition Added:
 {
  "index_name": "PK Batting",
  "type": "PRIMARY",
  "columns": [
    "playerID",
    "yearID",
    "stint"
  ]
}
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Index 'PK Appearances' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Index PK Appearances for appearances has been added!
For Table appearances, Index Definition Added:
 {
  "index_name": "PK Appearances",
  "type": "PRIMARY",
  "columns": [
    "playerID",
    "yearID",
    "teamID"
  ]
}
--------------------------------------------------------------------------------
---
```

[22]:
```
tab.test_load_info()
```

```
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
People.csv
--------------------------------------------------------------------------------
---
```

[23]:
```
tab.test_get_col_names()
```

```
Q =  select * from csvtables where table_name = 'people'
```

```
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
['bats', 'bbrefID', 'birthCity', 'birthCountry', 'birthDay', 'birthMonth',
'birthState', 'birthYear', 'deathCity', 'deathCountry', 'deathDay',
'deathMonth', 'deathState', 'deathYear', 'debut', 'finalGame', 'height',
'nameFirst', 'nameGiven', 'nameLast', 'playerID', 'retroID', 'throws', 'weight']
--------------------------------------------------------------------------------
---
```

[24]:
```
tab.add_other_indexes()
```

```
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Index 'last_first' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Index last_first for people has been added!
For Table people, Index Definition Added:
 {
  "index_name": "last_first",
  "type": "INDEX",
  "columns": [
    "nameLast",
    "nameFirst"
  ]
}
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Index 'TeamID' not found
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Index TeamID for batting has been added!
For Table batting, Index Definition Added:
 {
  "index_name": "TeamID",
  "type": "INDEX",
  "columns": [
    "teamID"
  ]
}
--------------------------------------------------------------------------------
```

---

[25]:
```
# This should throw an error
# Make sure it works properly when you run it in pycharm though!
tab.load_test()
# This would be missing some rows in the PDF as I had to delete some pages as it was really long PDF!!!
```

```
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where table_name = 'batting'
+------+------+------+------+------+-----+--------+-----+-------+------+-------+--------+------------+-----+-------+------+--------+-----+-------+------+-------+
|   2B |   3B |   AB |   BB |   CS |   G | GIDP   |   H | HBP   |   HR | IBB   | lgID   | playerID   |   R |   RBI |   SB | SF   | SH   |   SO |   stint | teamID | yearID |
+======+======+======+======+======+=====+========+=====+=======+======+=======+========+============+=====+=======+======+======+======+======+=========+======+========+
|    0 |    0 |    0 |    0 |    0 |  11 | 0      |   0 | 0     |    0 | 0     | NL     | aardsda01  |   0 |     0 |    0 | 0    | 0    |    0 |       1 | SFN    |   2004 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+--------+------------+-----+-------+------+------+------+------+---------+--------+--------+
|    0 |    0 |    2 |    0 |    0 |  45 | 0      |   0 | 0     |    0 | 0     | NL     | aardsda01  |   0 |     0 |    0 | 0    | 1    |    0 |       1 | CHN    |   2006 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+--------+------------+-----+-------+------+------+------+------+---------+--------+--------+
|    0 |    0 |    0 |    0 |    0 |  25 | 0      |   0 | 0     |    0 | 0     | AL     | aardsda01  |   0 |     0 |    0 | 0    | 0    |    0 |       1 | CHA    |   2007 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+--------+------------+-----+-------+------+------+------+------+---------+--------+--------+
|    0 |    0 |    1 |    0 |    0 |  47 | 0      |   0 | 0     |    0 | 0     | AL     | aardsda01  |   0 |     0 |    0 | 0    | 0    |    1 |       1 | BOS    |   2008 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+--------+------------+-----+-------+------+------+------+------+---------+--------+--------+
|    0 |    0 |    0 |    0 |    0 |  73 | 0      |   0 | 0     |    0 | 0     | AL     | aardsda01  |   0 |     0 |    0 | 0    | 0    |    0 |       1 | SEA    |   2009 |
```

```
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    0 |    0 |    0 |    0 |    0 |  53 | 0      |   0 | 0     |    0 | 0     |
AL     | aardsda01  |   0 |     0 |    0 | 0    | 0    |    0 |       1 | SEA
|     2010 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    0 |    0 |    0 |    0 |    0 |   1 | 0      |   0 | 0     |    0 | 0     |
AL     | aardsda01  |   0 |     0 |    0 | 0    | 0    |    0 |       1 | NYA
|     2012 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    0 |    0 |    0 |    0 |    0 |  43 | 0      |   0 | 0     |    0 | 0     |
NL     | aardsda01  |   0 |     0 |    0 | 0    | 0    |    0 |       1 | NYN
|     2013 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    0 |    0 |    1 |    0 |    0 |  33 | 0      |   0 | 0     |    0 | 0     |
NL     | aardsda01  |   0 |     0 |    0 | 0    | 0    |    1 |       1 | ATL
|     2015 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|   27 |    6 |  468 |   28 |    2 | 122 | 13     | 131 | 3     |   13 |       |
NL     | aaronha01  |  58 |    69 |    2 | 4    | 6    |   39 |       1 | ML1
|     1954 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|   37 |    9 |  602 |   49 |    1 | 153 | 20     | 189 | 3     |   27 | 5     |
NL     | aaronha01  | 105 |   106 |    3 | 4    | 7    |   61 |       1 | ML1
|     1955 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|   34 |   14 |  609 |   37 |    4 | 153 | 21     | 200 | 2     |   26 | 6     |
NL     | aaronha01  | 106 |    92 |    2 | 7    | 5    |   54 |       1 | ML1
|     1956 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|   27 |    6 |  615 |   57 |    1 | 151 | 13     | 198 | 0     |   44 | 15    |
NL     | aaronha01  | 118 |   132 |    1 | 3    | 0    |   58 |       1 | ML1
|     1957 |
```

```
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    9 |    0 |  177 |    6 |    4 |  88 | 4      |  46 | 2     |    5 | 1     |
NL     | beanbi01   |  19 |    32 |    2 | 5    | 2    |   29 |       1 | SDN
|     1993 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    5 |    1 |  135 |    7 |    1 |  84 | 4      |  29 | 0     |    0 | 1     |
NL     | beanbi01   |   7 |    14 |    0 | 3    | 1    |   25 |       1 | SDN
|     1994 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    0 |    0 |    7 |    1 |    0 |   4 | 0      |   0 | 0     |    0 | 0     |
NL     | beanbi01   |   1 |     0 |    0 | 0    | 0    |    4 |       1 | SDN
|     1995 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    0 |    0 |    0 |    0 |    0 |   1 | 0      |   0 | 0     |    0 | 0     |
AL     | beanco01   |   0 |     0 |    0 | 0    | 0    |    0 |       1 | NYA
|     2005 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    0 |    0 |    0 |    0 |    0 |   2 | 0      |   0 | 0     |    0 | 0     |
AL     | beanco01   |   0 |     0 |    0 | 0    | 0    |    0 |       1 | NYA
|     2006 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    0 |    0 |    0 |    0 |    0 |   3 | 0      |   0 | 0     |    0 | 0     |
AL     | beanco01   |   0 |     0 |    0 | 0    | 0    |    0 |       1 | NYA
|     2007 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    0 |    0 |   10 |    0 |    1 |   5 | 0      |   1 | 0     |    0 | 0     |
NL     | beanebi01  |   0 |     0 |    0 | 0    | 0    |    2 |       1 | NYN
|     1984 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    1 |    0 |    8 |    0 |    0 |   8 | 0      |   2 | 0     |    0 | 0     |
NL     | beanebi01  |   0 |     1 |    0 | 0    | 0    |    3 |       1 | NYN
|     1985 |
```

```
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    6 |    0 |  183 |   11 |    3 |  80 | 6      |  39 | 0     |    3 | 0     |
AL     | beanebi01  |  20 |    15 |    2 | 0    | 0    |   54 |       1 | MIN
|     1986 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    2 |    0 |   15 |    0 |    0 |  12 | 0      |   4 | 0     |    0 | 0     |
AL     | beanebi01  |   1 |     1 |    0 | 0    | 0    |    6 |       1 | MIN
|     1987 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    0 |    0 |    6 |    0 |    0 |   6 | 0      |   1 | 0     |    0 | 0     |
AL     | beanebi01  |   1 |     1 |    0 | 0    | 0    |    2 |       1 | DET
|     1988 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    5 |    0 |   79 |    0 |    1 |  37 | 2      |  19 | 0     |    0 | 0     |
AL     | beanebi01  |   8 |    11 |    3 | 1    | 2    |   13 |       1 | OAK
|     1989 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    2 |    1 |  176 |    5 |    0 |  48 |        |  39 | 1     |    0 |       |
NL     | beanjo01   |  13 |     5 |    9 |     | 7    |    0 |       1 | NY1
|     1902 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
--------------------------------------------------------------------------------
---
```

[26]:
```
# Might throw an error depending on table size
# Make sure it works properly when you run it in pycharm though!
tab.dumb_join_test()
```

```
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
```

```
Processed 1000 left rows.
Processed 2000 left rows.
Processed 3000 left rows.
Processed 4000 left rows.
Processed 5000 left rows.
Processed 5162 left rows.
+-----------+---------+----------+------+-----+---------+-------------+
| playerID  |  yearID | teamID   |   AB |   H |   G_all |   G_batting |
+===========+=========+==========+======+=====+=========+=============+
| baxtemi01 |    2011 | NYN      |   34 |   8 |      22 |          22 |
+-----------+---------+----------+------+-----+---------+-------------+
| baxtemi01 |    2012 | NYN      |  179 |  47 |      89 |          89 |
+-----------+---------+----------+------+-----+---------+-------------+
| baxtemi01 |    2013 | NYN      |  132 |  25 |      74 |          74 |
+-----------+---------+----------+------+-----+---------+-------------+
Time in searching and retrieval: 9.63268518447876
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Processed 1000 left rows.
Processed 2000 left rows.
Processed 3000 left rows.
Processed 4000 left rows.
Processed 5000 left rows.
Processed 6000 left rows.
Processed 7000 left rows.
Processed 8000 left rows.
Processed 9000 left rows.
Processed 10000 left rows.
Processed 11000 left rows.
Processed 12000 left rows.
Processed 13000 left rows.
Processed 14000 left rows.
Processed 15000 left rows.
Processed 16000 left rows.
Processed 17000 left rows.
Processed 18000 left rows.
Processed 19000 left rows.
Processed 19369 left rows.
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
```

```
| playerID   |   yearID | teamID   | nameLast   | nameFirst   |   height |
weight |   G_all |   G_batting |
+============+==========+==========+============+=============+==========+======
====+=========+=============+
| baxtemi01  |     2010 | SDN      | Baxter     | Mike        |       72 |
205 |       9 |           9 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2011 | NYN      | Baxter     | Mike        |       72 |
205 |      22 |          22 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2012 | NYN      | Baxter     | Mike        |       72 |
205 |      89 |          89 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2013 | NYN      | Baxter     | Mike        |       72 |
205 |      74 |          74 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2014 | LAN      | Baxter     | Mike        |       72 |
205 |       4 |           4 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2015 | CHN      | Baxter     | Mike        |       72 |
205 |      34 |          34 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
Time in searching and retrieval: 34.555185079574585
------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Processed 1000 left rows.
Processed 2000 left rows.
Processed 3000 left rows.
Processed 4000 left rows.
Processed 5000 left rows.
Processed 6000 left rows.
Processed 7000 left rows.
Processed 8000 left rows.
Processed 9000 left rows.
```

```
Processed 10000 left rows.
Processed 11000 left rows.
Processed 12000 left rows.
Processed 13000 left rows.
Processed 14000 left rows.
Processed 15000 left rows.
Processed 16000 left rows.
Processed 17000 left rows.
Processed 18000 left rows.
Processed 19000 left rows.
Processed 19369 left rows.
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
| playerID   |   yearID | teamID   | nameLast   | nameFirst   |   height |
weight |   R |   AB |
+============+==========+==========+============+=============+==========+======
====+=====+======+
| baxtemi01  |     2011 | NYN      | Baxter     | Mike        |       72 |
205 |   6 |   34 |
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
| baxtemi01  |     2012 | NYN      | Baxter     | Mike        |       72 |
205 |  26 |  179 |
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
| baxtemi01  |     2013 | NYN      | Baxter     | Mike        |       72 |
205 |  14 |  132 |
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
Time in searching and retrieval: 34.14841890335083
--------------------------------------------------------------------------------
---
```

[27]:
```
tab.get_access_path_test()
```

```
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Test Case -1: Batting Table: Template: {'teamID': 'NYN', 'playerID':
'baxtemi01', 'yearID': '2012'}
Index: TeamID
Count: 124
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
```

```
table_name = 'batting'
Test Case -2: Batting Table: Template: {'teamID': 'NYN', 'playerID':
'baxtemi01', 'yearID': '2012', 'stint': '1'}
Index: PK Batting
Count: 5162
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Test Case -3: Appearances Table: Template: {'teamID': 'NYN', 'playerID':
'baxtemi01', 'yearID': '2012', 'lgID': 'NA'}
Index: PK Appearances
Count: 5257
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Test Case -4: (No Best Index will be Found!) Appearances Table: Template:
{'teamID': 'NYN', 'playerID': 'baxtemi01', 'lgID': 'NA'}
Index: None
Count: None
--------------------------------------------------------------------------------
---
```

[28]:
```
tab.sub_where_template_test()
```

```
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Test for Table batting and template: {'playerID': 'baxtemi01', 'teamID': 'NYN',
'yearID': '2012', 'G': '89', 'AB': '179', 'R': '26', 'ABC': '12345'}
Sub Template: {'playerID': 'baxtemi01', 'teamID': 'NYN', 'yearID': '2012', 'G':
'89', 'AB': '179', 'R': '26'}
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Test for Table people and template: {'playerID': 'aguilje01', 'nameFirst':
'Chris', 'nameLast': 'Aguila', 'throws': 'R', 'random_col_1': '1234',
'random_col_2': '5678', 'weight': '200'}
Sub Template: {'playerID': 'aguilje01', 'nameFirst': 'Chris', 'nameLast':
```

```
'Aguila', 'throws': 'R', 'weight': '200'}
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Test for Table appearances and template: {'playerID': 'addybo01', 'teamID':
'RC1', 'yearID': '1871', 'lgID': 'NA', 'random_col_1': '1234', 'random_col_2':
'5678', 'GS': '25'}
Sub Template: {'playerID': 'addybo01', 'teamID': 'RC1', 'yearID': '1871',
'lgID': 'NA', 'GS': '25'}
--------------------------------------------------------------------------------
---
```

[29]:
```
tab.test_find_by_template_index()
```

```
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Test for Table batting, template: {'playerID': 'baxtemi01', 'stint': '1',
'yearID': '2012', 'lgID': 'NL', 'AB': '179'} with index PK Batting and
projection fields ['playerID', 'yearID', 'stint', 'lgID', 'AB', 'G', 'R', 'H',
'2B', '3B']
Matching Rows:
[{'playerID': 'baxtemi01', 'yearID': '2012', 'stint': '1', 'lgID': 'NL', 'AB':
'179', 'G': '89', 'R': '26', 'H': '47', '2B': '14', '3B': '2'}]
--------------------------------------------------------------------------------
---
Test for Table batting, template: {'teamID': 'RC1', 'G': '25'} with index TeamID
and projection fields ['playerID', 'yearID', 'stint', 'lgID', 'AB', 'G', 'R',
'H', '2B', '3B']
Matching Rows:
[{'playerID': 'addybo01', 'yearID': '1871', 'stint': '1', 'lgID': 'NA', 'AB':
'118', 'G': '25', 'R': '30', 'H': '32', '2B': '6', '3B': '0'}, {'playerID':
'ansonca01', 'yearID': '1871', 'stint': '1', 'lgID': 'NA', 'AB': '120', 'G':
'25', 'R': '29', 'H': '39', '2B': '11', '3B': '3'}]
--------------------------------------------------------------------------------
---
When No Indexing Happens: Test for Table batting, template: {'teamID': 'RC1',
'G': '25', 'stint': '1'} with index PK Batting and projection fields
['playerID', 'yearID', 'stint', 'lgID', 'AB', 'G', 'R', 'H', '2B', '3B']
Rows can't be indexed with this set of template!!!
Matching Rows:
[{'playerID': 'addybo01', 'yearID': '1871', 'stint': '1', 'lgID': 'NA', 'AB':
'118', 'G': '25', 'R': '30', 'H': '32', '2B': '6', '3B': '0'}, {'playerID':
'ansonca01', 'yearID': '1871', 'stint': '1', 'lgID': 'NA', 'AB': '120', 'G':
```

```
'25', 'R': '29', 'H': '39', '2B': '11', '3B': '3'}]
--------------------------------------------------------------------------------
---
```

[30]:
```
tab.smart_join_test()
```

```
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Processed 3 left rows.
+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| baxtemi01  |     2011 | NYN      |   34 |   8 |      22 |          22 |
+------------+----------+----------+------+-----+---------+-------------+
| baxtemi01  |     2012 | NYN      |  179 |  47 |      89 |          89 |
+------------+----------+----------+------+-----+---------+-------------+
| baxtemi01  |     2013 | NYN      |  132 |  25 |      74 |          74 |
+------------+----------+----------+------+-----+---------+-------------+
Time in searching and retrieval: 0.4516479969024658
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Processed 1 left rows.
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| playerID   |   yearID | teamID   | nameLast   | nameFirst   |   height |
weight |   G_all |   G_batting |
+============+==========+==========+============+=============+==========+======
====+=========+=============+
| baxtemi01  |     2010 | SDN      | Baxter     | Mike        |       72 |
205 |       9 |           9 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2011 | NYN      | Baxter     | Mike        |       72 |
205 |      22 |          22 |
```

```
+------------+---------+---------+------------+-------------+---------+------
----+---------+-------------+
| baxtemi01  |     2012 | NYN      | Baxter     | Mike        |       72 |
205 |      89 |          89 |
+------------+---------+---------+------------+-------------+---------+------
----+---------+-------------+
| baxtemi01  |     2013 | NYN      | Baxter     | Mike        |       72 |
205 |      74 |          74 |
+------------+---------+---------+------------+-------------+---------+------
----+---------+-------------+
| baxtemi01  |     2014 | LAN      | Baxter     | Mike        |       72 |
205 |       4 |           4 |
+------------+---------+---------+------------+-------------+---------+------
----+---------+-------------+
| baxtemi01  |     2015 | CHN      | Baxter     | Mike        |       72 |
205 |      34 |          34 |
+------------+---------+---------+------------+-------------+---------+------
----+---------+-------------+
Time in searching and retrieval: 1.1431481838226318
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Processed 1 left rows.
+------------+---------+---------+------------+-------------+---------+------
----+-----+------+
| playerID   |   yearID | teamID   | nameLast   | nameFirst   |   height |
weight |   R |   AB |
+============+=========+=========+============+=============+==========+======
====+=====+======+
| baxtemi01  |     2011 | NYN      | Baxter     | Mike        |       72 |
205 |   6 |   34 |
+------------+---------+---------+------------+-------------+---------+------
----+-----+------+
| baxtemi01  |     2012 | NYN      | Baxter     | Mike        |       72 |
205 |  26 |  179 |
+------------+---------+---------+------------+-------------+---------+------
----+-----+------+
| baxtemi01  |     2013 | NYN      | Baxter     | Mike        |       72 |
205 |  14 |  132 |
+------------+---------+---------+------------+-------------+---------+------
----+-----+------+
```

```
Time in searching and retrieval: 1.16861891746521
--------------------------------------------------------------------------------
---
```

[31]:
```
# Compare the time it takes to do the dumb join and the smart join below
#This is a timer that will track how long it takes to execute your cell.

# Times will vary based on how long it takes to query your AWS Server, but you␣
 ↪should see a notable improvement using smart_join()

#----Your Code Here----
%time tab.dumb_join_test()
```

```
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Processed 1000 left rows.
Processed 2000 left rows.
Processed 3000 left rows.
Processed 4000 left rows.
Processed 5000 left rows.
Processed 5162 left rows.
+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| baxtemi01  |     2011 | NYN      |   34 |   8 |      22 |          22 |
+------------+----------+----------+------+-----+---------+-------------+
| baxtemi01  |     2012 | NYN      |  179 |  47 |      89 |          89 |
+------------+----------+----------+------+-----+---------+-------------+
| baxtemi01  |     2013 | NYN      |  132 |  25 |      74 |          74 |
+------------+----------+----------+------+-----+---------+-------------+
Time in searching and retrieval: 9.726093769073486
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Processed 1000 left rows.
```

```
Processed 2000 left rows.
Processed 3000 left rows.
Processed 4000 left rows.
Processed 5000 left rows.
Processed 6000 left rows.
Processed 7000 left rows.
Processed 8000 left rows.
Processed 9000 left rows.
Processed 10000 left rows.
Processed 11000 left rows.
Processed 12000 left rows.
Processed 13000 left rows.
Processed 14000 left rows.
Processed 15000 left rows.
Processed 16000 left rows.
Processed 17000 left rows.
Processed 18000 left rows.
Processed 19000 left rows.
Processed 19369 left rows.
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| playerID   |   yearID | teamID   | nameLast   | nameFirst   |   height |
weight |   G_all |   G_batting |
+============+==========+==========+============+=============+==========+======
====+=========+=============+
| baxtemi01  |     2010 | SDN      | Baxter     | Mike        |       72 |
205 |       9 |           9 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2011 | NYN      | Baxter     | Mike        |       72 |
205 |      22 |          22 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2012 | NYN      | Baxter     | Mike        |       72 |
205 |      89 |          89 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2013 | NYN      | Baxter     | Mike        |       72 |
205 |      74 |          74 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2014 | LAN      | Baxter     | Mike        |       72 |
205 |       4 |           4 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2015 | CHN      | Baxter     | Mike        |       72 |
205 |      34 |          34 |
+------------+----------+----------+------------+-------------+----------+------
```

```
----+---------+-------------+
Time in searching and retrieval: 34.58417510986328
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Processed 1000 left rows.
Processed 2000 left rows.
Processed 3000 left rows.
Processed 4000 left rows.
Processed 5000 left rows.
Processed 6000 left rows.
Processed 7000 left rows.
Processed 8000 left rows.
Processed 9000 left rows.
Processed 10000 left rows.
Processed 11000 left rows.
Processed 12000 left rows.
Processed 13000 left rows.
Processed 14000 left rows.
Processed 15000 left rows.
Processed 16000 left rows.
Processed 17000 left rows.
Processed 18000 left rows.
Processed 19000 left rows.
Processed 19369 left rows.
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
| playerID   |   yearID | teamID   | nameLast   | nameFirst   |   height |
weight |   R |   AB |
+============+==========+==========+============+=============+==========+======
====+=====+======+
| baxtemi01  |     2011 | NYN      | Baxter     | Mike        |       72 |
205 |   6 |   34 |
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
| baxtemi01  |     2012 | NYN      | Baxter     | Mike        |       72 |
205 |  26 |  179 |
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
| baxtemi01  |     2013 | NYN      | Baxter     | Mike        |       72 |
205 |  14 |  132 |
```

```
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
Time in searching and retrieval: 34.15868806838989
--------------------------------------------------------------------------------
---
CPU times: user 1min 18s, sys: 135 ms, total: 1min 18s
Wall time: 1min 18s
```

[32]:
```
# Compare the time it takes to do the dumb join and the smart join below
#This is a timer that will track how long it takes to execute your cell.

# Times will vary based on how long it takes to query your AWS Server, but you␣
 ↪should see a notable improvement using smart_join()

#----Your Code Here----
%time tab.smart_join_test()
```

```
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Processed 3 left rows.
+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| baxtemi01  |     2011 | NYN      |   34 |   8 |      22 |          22 |
+------------+----------+----------+------+-----+---------+-------------+
| baxtemi01  |     2012 | NYN      |  179 |  47 |      89 |          89 |
+------------+----------+----------+------+-----+---------+-------------+
| baxtemi01  |     2013 | NYN      |  132 |  25 |      74 |          74 |
+------------+----------+----------+------+-----+---------+-------------+
Time in searching and retrieval: 0.46011877059936523
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Q =  select * from csvtables where table_name = 'appearances'
Q =  select * from csvcolumns WHERE table_name = 'appearances'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'appearances'
Processed 1 left rows.
+------------+----------+----------+------------+-------------+----------+------
```

```
----+---------+-------------+
| playerID   |   yearID | teamID   | nameLast   | nameFirst   |   height |
weight |   G_all |   G_batting |
+============+==========+==========+============+=============+==========+======
====+=========+=============+
| baxtemi01  |     2010 | SDN      | Baxter     | Mike        |       72 |
205 |       9 |           9 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2011 | NYN      | Baxter     | Mike        |       72 |
205 |      22 |          22 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2012 | NYN      | Baxter     | Mike        |       72 |
205 |      89 |          89 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2013 | NYN      | Baxter     | Mike        |       72 |
205 |      74 |          74 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2014 | LAN      | Baxter     | Mike        |       72 |
205 |       4 |           4 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
| baxtemi01  |     2015 | CHN      | Baxter     | Mike        |       72 |
205 |      34 |          34 |
+------------+----------+----------+------------+-------------+----------+------
----+---------+-------------+
Time in searching and retrieval: 1.1450910568237305
--------------------------------------------------------------------------------
---
Q =  select * from csvtables where table_name = 'people'
Q =  select * from csvcolumns WHERE table_name = 'people'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'people'
Q =  select * from csvtables where table_name = 'batting'
Q =  select * from csvcolumns WHERE table_name = 'batting'
Q =  select column_name, index_name, type, index_order from csvindexes where
table_name = 'batting'
Processed 1 left rows.
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
| playerID   |   yearID | teamID   | nameLast   | nameFirst   |   height |
weight |   R |   AB |
+============+==========+==========+============+=============+==========+======
====+=====+======+
| baxtemi01  |     2011 | NYN      | Baxter     | Mike        |       72 |
```

```
205 |   6 |   34 |
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
| baxtemi01  |     2012 | NYN      | Baxter     | Mike        |       72 |
205 |  26 |  179 |
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
| baxtemi01  |     2013 | NYN      | Baxter     | Mike        |       72 |
205 |  14 |  132 |
+------------+----------+----------+------------+-------------+----------+------
----+-----+------+
Time in searching and retrieval: 1.171389102935791
--------------------------------------------------------------------------------
---
CPU times: user 2.76 s, sys: 15.7 ms, total: 2.78 s
Wall time: 2.79 s
```

As we can see above, there is a huge difference in the time taken by the dumb join and the smart join. While running 3 dumb join queries takes about 78 secs, running smart join queries (the same 3 of them), takes only about 2.79 secs which is a huge difference. Smart Join having the optimizations seems really important for a good DB Engine.