W4111_HW2_Programming

February 14, 2021

1 COMS W4111 Introduction to Databases

2 Homework 2: Implement a Simple Database Engine (Programming)

2.0.1 Shuoting Kao

2.0.2 sk4920

2.1 Part 1: Written & SQL

2.1.1 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.

Answer: Metadata defines a way that users access data and metadata itself, and the columns in the catalog's table offer information about columns of the tables. Refer to https://www.tutorialspoint.com/dbms/dbms_codds_rules.htm

2. Give 3 examples of what would be stored in a database catalog

Answer: Table definition, Index definition, and columns definition.

3. What is the SQL database catalog called?

Answer: Database catalog is the collection of metadata used to define and locate data. Refer to https://www.alation.com/blog/what-is-a-data-catalog/

A Data Catalog is a collection of metadata, combined with data management and search tools, that helps analysts and other data users to find the data that they need, serves as an inventory of available data, and provides information to evaluate fitness data for intended uses.

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

Answer: Indicies play crucial roles in database system since it improves operation efficiency by locating specific values, rather than searching a whole table. Refer to Textbook, "4.6 Index Definition in SQL"

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

Answer: The former one would not accept a NULL value, but the latter would accpet a NULL value once.

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.

Answer: The first one would be faster since it does not join a whole table. Besides, the where clause reduces the size of table. "playerID in (select playerID from collegeplaying where schoolID = 'Fordham')"

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?

Answer: "ON DELETE CASCADE" helps users deleting the data automatically from the descendant table as they delete data from the parent table. Without this option, users have to delete data from relevant tables manually to maintain foreign key constraints.

"ON DELETE RESTRICT" prevents users from deleting a row in the parent table alone, which also presents in the descendant table.

Refer to https://www.mysqltutorial.org/mysql-on-delete-cascade/

2.1.2 SQL

[1]:
```
%load_ext sql
%sql mysql+pymysql://admin:orphanage73@database-1.cie2eqwscgmp.us-east-2.rds.
↪amazonaws.com/lahmansbaseballdb
%sql use lahmansbaseballdb;
```

```
 * mysql+pymysql://admin:***@database-1.cie2eqwscgmp.us-
east-2.rds.amazonaws.com/lahmansbaseballdb
0 rows affected.
```

[1]:
```
[]
```

1. Initials

- Find the `initials`, `firstName`, `lastName`, for every player.
- You need to return 10 rows.

- Note: Even for those players with two last names, just return the first letter of their first last name

Answer: %sql select concat(left(nameFirst,1),left(nameLast,1)) as initials, nameFirst, nameLast from people limit 10;

[2]:
```
%%sql
select
concat(
    left(nameFirst,1),
    left(nameLast,1)
    ) as initials,
nameFirst,
nameLast
from people limit 10;
```

```
 * mysql+pymysql://admin:***@database-1.cie2eqwscgmp.us-
east-2.rds.amazonaws.com/lahmansbaseballdb
10 rows affected.
```

[2]:
```
[('DA', 'David', 'Aardsma'),
 ('HA', 'Hank', 'Aaron'),
 ('TA', 'Tommie', 'Aaron'),
 ('DA', 'Don', 'Aase'),
 ('AA', 'Andy', 'Abad'),
 ('FA', 'Fernando', 'Abad'),
 ('JA', 'John', 'Abadie'),
 ('EA', 'Ed', 'Abbaticchio'),
 ('BA', 'Bert', 'Abbey'),
 ('CA', 'Charlie', 'Abbey')]
```

2. Games Per Player

- Find the `yearID, lgID, games_per_player`, for every year and league.
- Use a function to round down to the nearest integer the average games_per_player
- You need to return 10 rows.
- YOU MAY NOT USE `GROUP BY`! A statement using `group by` will receive 0.

Answer:

[48]:
```
%%sql
select
       games.yearID,
       games.lgID,
       floor(games.tot_games/players.tot_players) as games_per_player
from
    (
    select
    distinct yearID,
    lgID,
    sum(G_all)
        over (partition by yearID,lgID) as tot_games from appearances
    ) as games

inner join

    (
    select
    distinct distinct_players.yearID,
    distinct_players.lgID,
    count(distinct_players.playerID)
        over (partition by distinct_players.yearID, distinct_players.lgID) as␣
 ↪tot_players
            from (select distinct playerID, yearID, lgID  from appearances) as␣
 ↪distinct_players
    ) as players

on games.yearID = players.yearID
and games.lgID = players.lgID
order by games.yearID desc
limit 10;
```

```
 * mysql+pymysql://admin:***@database-1.cie2eqwscgmp.us-
east-2.rds.amazonaws.com/lahmansbaseballdb
10 rows affected.
```

```
[48]: [(2019, 'AL', Decimal('46')),
       (2019, 'NL', Decimal('50')),
       (2018, 'AL', Decimal('48')),
       (2018, 'NL', Decimal('49')),
       (2017, 'AL', Decimal('48')),
       (2017, 'NL', Decimal('51')),
       (2016, 'AL', Decimal('49')),
       (2016, 'NL', Decimal('49')),
       (2015, 'AL', Decimal('49')),
       (2015, 'NL', Decimal('49'))]
```

2.2 Part 2: CSVCatalog Tests

Once you have tested everything successfully 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

[2]:
```
import unit_test_catalog as cat # This notebook should be in the same directory␣
↪as your project
```

[4]:
```
cat.create_table_test()
```

```
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)
```

[3]:
```
cat.drop_table_test()
```

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

[5]:
```
cat.add_column_test()
```

```
Running load core definition appearances
Q =  select * from csvtables where table_name = %s
load core definition table/filename appearances NewAppearances.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
```

```
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Running load core definition people
Q =  select * from csvtables where table_name = %s
load core definition table/filename people NewPeople.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
```

```
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Table =  in the index if statement
{
  "table_name": "batting",
  "path": "NewBatting.csv",
  "columns": [
    {
      "column_name": "playerID",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "yearID",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "stint",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "teamID",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "lgID",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "G",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "AB",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "R",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "H",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "2B",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "3B",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "HR",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "RBI",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "SB",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "CS",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "BB",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "SO",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "IBB",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "HBP",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "SH",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "SF",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "GIDP",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": []
}
```

[6]:
```
cat.column_name_failure_test()  # This will throw an error
```

```
issue!!
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-6-342044887ceb> in <module>
----> 1 cat.column_name_failure_test()  # This will throw an error

~\HW2\unit_test_catalog.py in column_name_failure_test()
    123 def column_name_failure_test():
    124     cat = CSVCatalog.CSVCatalog()
--> 125     col = CSVCatalog.ColumnDefinition(None, "text", False)
    126     t = cat.get_table("people")
    127     t.add_column_definition(col)

~\HW2\CSVCatalog.py in __init__(self, column_name, column_type, not_null)
     51         if column_name == None:
     52             print("issue!!")
---> 53             raise ValueError('You must have a column name!!')
     54         else:
     55             self.column_name = column_name

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

[7]:
```
cat.column_type_failure_test()  # This will throw an error
```

```
Issue!
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-7-eebf587b1ffc> in <module>
----> 1 cat.column_type_failure_test()  # This will throw an error

~\HW2\unit_test_catalog.py in column_type_failure_test()
    133 def column_type_failure_test():
    134     cat = CSVCatalog.CSVCatalog()
--> 135     col = CSVCatalog.ColumnDefinition("bird", "canary", False)
    136     t = cat.get_table("people")
    137     t.add_column_definition(col)

~\HW2\CSVCatalog.py in __init__(self, column_name, column_type, not_null)
     59         else:
     60             print("Issue!")
---> 61             raise ValueError('That column type is not accepted. Please␣
 ↪try again.')
     62 
     63         if type(not_null) == type(True):

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

[8]:
```
cat.column_not_null_failure_test()  # This will throw an error
```

```
issue!
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-8-9b5701466b82> in <module>
----> 1 cat.column_not_null_failure_test()  # This will throw an error
```

```
~\HW2\unit_test_catalog.py in column_not_null_failure_test()
    143 def column_not_null_failure_test():
    144     cat = CSVCatalog.CSVCatalog()
--> 145     col = CSVCatalog.ColumnDefinition("name", "text", "happy")
    146     t = cat.get_table("people")
    147     t.add_column_definition(col)

~\HW2\CSVCatalog.py in __init__(self, column_name, column_type, not_null)
     65         else:
     66             print("issue!")
---> 67             raise ValueError('The not_null column must be either True o
 ↪False! Please try again.')
     68
     69     def __str__(self):

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

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

```
Running load core definition people
Q =  select * from csvtables where table_name = %s
load core definition table/filename people NewPeople.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
Running load core definition appearances
Q =  select * from csvtables where table_name = %s
load core definition table/filename appearances NewAppearances.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
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)
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)
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
```

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

```
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'stint', 'type': 'PRIMARY', 'index_name':
'playerID_stint_yearID', 'index_order': '1'}, {'table_name': 'batting',
'column_name': 'yearID', 'type': 'PRIMARY', 'index_name':
'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
Q =  DELETE FROM csvcolumns WHERE table_name = 'batting' and column_name = '2B'
sqlColumn '2B' has been dropped! None
Column '2B' has been dropped!
in the index if statement
{
  "table_name": "batting",
  "path": "NewBatting.csv",
  "columns": [
    {
      "column_name": "3B",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "AB",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "BB",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "CS",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G",
    "column_type": "text",
    "not_null": true
  },
  {
    "column_name": "GIDP",
    "column_type": "text",
    "not_null": false
  },
  {
    "column_name": "H",
    "column_type": "text",
    "not_null": false
  },
  {
    "column_name": "HBP",
    "column_type": "text",
    "not_null": false
  },
  {
    "column_name": "HR",
    "column_type": "text",
    "not_null": false
  },
  {
    "column_name": "IBB",
    "column_type": "text",
    "not_null": false
  },
  {
    "column_name": "lgID",
    "column_type": "text",
    "not_null": true
  },
  {
    "column_name": "playerID",
    "column_type": "text",
    "not_null": true
  },
  {
    "column_name": "R",
    "column_type": "text",
    "not_null": false
  },
  {
    "column_name": "RBI",
    "column_type": "text",
    "not_null": false
  },
    {
      "column_name": "SB",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "SF",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "SH",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "SO",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "stint",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "teamID",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "yearID",
      "column_type": "text",
      "not_null": true
    }
  ],
  "indexes": [
    {
      "index_name": "playerID_stint_yearID",
      "type": "PRIMARY",
      "columns": [
        "playerID",
        "stint",
        "yearID"
      ]
    }
  ]
}
```

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

```
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'stint', 'type': 'PRIMARY', 'index_name':
'playerID_stint_yearID', 'index_order': '1'}, {'table_name': 'batting',
'column_name': 'yearID', 'type': 'PRIMARY', 'index_name':
'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
Q =  DELETE FROM csvindexes WHERE table_name = 'batting' and index_name =
'playerID_stint_yearID'
in the index if statement
{
  "table_name": "batting",
  "path": "NewBatting.csv",
  "columns": [
    {
      "column_name": "3B",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "AB",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "BB",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "CS",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "GIDP",
      "column_type": "text",
  "not_null": false
},
{
  "column_name": "H",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "HBP",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "HR",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "IBB",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "lgID",
  "column_type": "text",
  "not_null": true
},
{
  "column_name": "playerID",
  "column_type": "text",
  "not_null": true
},
{
  "column_name": "R",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "RBI",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "SB",
  "column_type": "text",
  "not_null": false
},
{
        "column_name": "SF",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "SH",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "SO",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "stint",
        "column_type": "text",
        "not_null": true
      },
      {
        "column_name": "teamID",
        "column_type": "text",
        "not_null": true
      },
      {
        "column_name": "yearID",
        "column_type": "text",
        "not_null": true
      }
    ],
    "indexes": []
  }
```

[12]:
```
cat.describe_table_test()
```

```
Running load core definition people
Q =  select * from csvtables where table_name = %s
load core definition table/filename people NewPeople.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'people', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID', 'index_order': '0'}] <class 'list'>
in the index if statement
DESCRIBE People =
 {
  "table_name": "people",
  "path": "NewPeople.csv",
  "columns": [
    {
      "column_name": "bats",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "bbrefID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "birthCity",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "birthCountry",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "birthDay",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "birthMonth",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "birthState",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "birthYear",
      "column_type": "text",
      "not_null": true
    },
    {
      "column_name": "deathCity",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "deathCountry",
      "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "deathDay",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "deathMonth",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "deathState",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "deathYear",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "debut",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "finalGame",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "height",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "nameFirst",
        "column_type": "text",
        "not_null": false
      },
      {
        "column_name": "nameGiven",
        "column_type": "text",
        "not_null": false
      },
      {
          "column_name": "nameLast",
          "column_type": "text",
          "not_null": false
        },
        {
          "column_name": "playerID",
          "column_type": "text",
          "not_null": true
        },
        {
          "column_name": "retroID",
          "column_type": "text",
          "not_null": false
        },
        {
          "column_name": "throws",
          "column_type": "text",
          "not_null": false
        },
        {
          "column_name": "weight",
          "column_type": "text",
          "not_null": false
        }
      ],
      "indexes": [
        {
          "index_name": "playerID",
          "type": "PRIMARY",
          "columns": [
            "playerID"
          ]
        }
      ]
    }
```

2.3 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.

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

```
[14]: # 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
```

```
[15]: 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)
```

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

```
Running load core definition people
Q =  select * from csvtables where table_name = %s
load core definition table/filename people NewPeople.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
```

```
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
```

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

```
Running load core definition appearances
Q =  select * from csvtables where table_name = %s
load core definition table/filename appearances NewAppearances.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
```

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

```
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
```

```
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
```

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

```
Running load core definition people
Q =  select * from csvtables where table_name = %s
load core definition table/filename people NewPeople.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
Running load core definition appearances
Q =  select * from csvtables where table_name = %s
load core definition table/filename appearances NewAppearances.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
() <class 'tuple'>
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)
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)
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
```

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

```
Running load core definition people
Q =  select * from csvtables where table_name = %s
load core definition table/filename people NewPeople.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'people', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID', 'index_order': '0'}] <class 'list'>
NewPeople.csv
```

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

```
Running load core definition people
Q =  select * from csvtables where table_name = %s
load core definition table/filename people NewPeople.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'people', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID', 'index_order': '0'}] <class 'list'>
['bats', 'bbrefID', 'birthCity', 'birthCountry', 'birthDay', 'birthMonth',
'birthState', 'birthYear', 'deathCity', 'deathCountry', 'deathDay',
'deathMonth', 'deathState', 'deathYear', 'debut', 'finalGame', 'height',
'nameFirst', 'nameGiven', 'nameLast', 'playerID', 'retroID', 'throws', 'weight']
```

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

```
Running load core definition people
Q =  select * from csvtables where table_name = %s
load core definition table/filename people NewPeople.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'people', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID', 'index_order': '0'}] <class 'list'>
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'stint', 'type': 'PRIMARY', 'index_name':
'playerID_stint_yearID', 'index_order': '1'}, {'table_name': 'batting',
'column_name': 'yearID', 'type': 'PRIMARY', 'index_name':
'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
Running load core definition appearances
Q =  select * from csvtables where table_name = %s
```

```
load core definition table/filename appearances NewAppearances.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'appearances', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'yearID_teamID_playerID', 'index_order': '0'}, {'table_name':
'appearances', 'column_name': 'teamID', 'type': 'PRIMARY', 'index_name':
'yearID_teamID_playerID', 'index_order': '1'}, {'table_name': 'appearances',
'column_name': 'playerID', 'type': 'PRIMARY', 'index_name':
'yearID_teamID_playerID', 'index_order': '2'}] <class 'list'>
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)
```

[23]:
```
# This should throw an error
# Make sure it works properly when you run it in pycharm though!
tab.load_test()
```

```
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'yearID', 'type': 'INDEX', 'index_name': 'yearID',
'index_order': '0'}, {'table_name': 'batting', 'column_name': 'stint', 'type':
'PRIMARY', 'index_name': 'playerID_stint_yearID', 'index_order': '1'},
{'table_name': 'batting', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+-----------+-----+-------+------+------+------+------+--------+------
----+----------+
|   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 |
+------+------+------+------+------+-----+--------+-----+-------+-----+-------+
--------+-----------+-----+-------+------+------+------+------+---------+------
----+----------+
```

[24]:
```
# Might throw an error depending on table size
# Make sure it works properly when you run it in pycharm though!
tab.dumb_join_test()
```

```
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'yearID', 'type': 'INDEX', 'index_name': 'yearID',
'index_order': '0'}, {'table_name': 'batting', 'column_name': 'stint', 'type':
'PRIMARY', 'index_name': 'playerID_stint_yearID', 'index_order': '1'},
{'table_name': 'batting', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
Running load core definition appearances
Q =  select * from csvtables where table_name = %s
load core definition table/filename appearances NewAppearances.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'appearances', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'yearID_teamID_playerID', 'index_order': '0'}, {'table_name':
'appearances', 'column_name': 'teamID', 'type': 'PRIMARY', 'index_name':
'yearID_teamID_playerID', 'index_order': '1'}, {'table_name': 'appearances',
'column_name': 'playerID', 'type': 'PRIMARY', 'index_name':
'yearID_teamID_playerID', 'index_order': '2'}] <class 'list'>
Processed 10 left rows.
Processed 20 left rows.
Processed 30 left rows.
Processed 40 left rows.
Processed 50 left rows.
Processed 60 left rows.
Processed 70 left rows.
Processed 80 left rows.
Processed 90 left rows.
Processed 100 left rows.
Processed 110 left rows.
Processed 120 left rows.
Processed 130 left rows.
Processed 140 left rows.
Processed 150 left rows.
Processed 160 left rows.
Processed 170 left rows.
Processed 180 left rows.
Processed 190 left rows.
Processed 200 left rows.
Processed 210 left rows.
Processed 220 left rows.
Processed 230 left rows.
Processed 240 left rows.
Processed 250 left rows.
Processed 260 left rows.
Processed 270 left rows.
Processed 280 left rows.
Processed 290 left rows.
Processed 300 left rows.
Processed 310 left rows.
Processed 320 left rows.
Processed 330 left rows.
Processed 340 left rows.
Processed 350 left rows.
Processed 360 left rows.
Processed 370 left rows.
Processed 380 left rows.
Processed 390 left rows.
Processed 400 left rows.
Processed 410 left rows.
Processed 420 left rows.
Processed 430 left rows.
Processed 440 left rows.
Processed 450 left rows.
Processed 460 left rows.
Processed 470 left rows.
Processed 480 left rows.
Processed 490 left rows.
Processed 500 left rows.
Processed 510 left rows.
Processed 520 left rows.
Processed 530 left rows.
Processed 540 left rows.
Processed 550 left rows.
Processed 560 left rows.
Processed 570 left rows.
Processed 580 left rows.
Processed 590 left rows.
Processed 600 left rows.
Processed 610 left rows.
Processed 620 left rows.
Processed 630 left rows.
Processed 640 left rows.
Processed 650 left rows.
Processed 660 left rows.
Processed 670 left rows.
Processed 680 left rows.
Processed 690 left rows.
Processed 700 left rows.
Processed 710 left rows.
Processed 720 left rows.
Processed 730 left rows.
Processed 740 left rows.
Processed 750 left rows.
Processed 760 left rows.
Processed 770 left rows.
Processed 780 left rows.
Processed 790 left rows.
Processed 800 left rows.
Processed 810 left rows.
Processed 820 left rows.
Processed 830 left rows.
Processed 840 left rows.
Processed 850 left rows.
Processed 860 left rows.
Processed 870 left rows.
Processed 880 left rows.
Processed 890 left rows.
Processed 900 left rows.
Processed 910 left rows.
Processed 920 left rows.
Processed 930 left rows.
Processed 940 left rows.
Processed 950 left rows.
Processed 960 left rows.
Processed 970 left rows.
Processed 980 left rows.
Processed 990 left rows.
Processed 1000 left rows.
Processed 1010 left rows.
Processed 1020 left rows.
Processed 1030 left rows.
Processed 1040 left rows.
Processed 1050 left rows.
Processed 1060 left rows.
Processed 1070 left rows.
Processed 1080 left rows.
Processed 1090 left rows.
Processed 1100 left rows.
Processed 1110 left rows.
Processed 1120 left rows.
Processed 1130 left rows.
Processed 1140 left rows.
Processed 1150 left rows.
Processed 1160 left rows.
Processed 1170 left rows.
Processed 1180 left rows.
Processed 1190 left rows.
Processed 1200 left rows.
Processed 1210 left rows.
Processed 1220 left rows.
Processed 1230 left rows.
Processed 1240 left rows.
Processed 1250 left rows.
Processed 1260 left rows.
Processed 1270 left rows.
Processed 1280 left rows.
Processed 1290 left rows.
Processed 1300 left rows.
Processed 1310 left rows.
Processed 1320 left rows.
Processed 1330 left rows.
Processed 1340 left rows.
Processed 1350 left rows.
Processed 1360 left rows.
Processed 1370 left rows.
Processed 1380 left rows.
Processed 1390 left rows.
Processed 1400 left rows.
Processed 1410 left rows.
Processed 1420 left rows.
Processed 1430 left rows.
Processed 1440 left rows.
Processed 1450 left rows.
Processed 1460 left rows.
Processed 1470 left rows.
Processed 1480 left rows.
Processed 1490 left rows.
Processed 1500 left rows.
Processed 1510 left rows.
Processed 1520 left rows.
Processed 1530 left rows.
Processed 1540 left rows.
Processed 1550 left rows.
Processed 1560 left rows.
Processed 1570 left rows.
Processed 1580 left rows.
Processed 1590 left rows.
Processed 1600 left rows.
Processed 1610 left rows.
Processed 1620 left rows.
Processed 1630 left rows.
Processed 1640 left rows.
Processed 1650 left rows.
Processed 1660 left rows.
Processed 1670 left rows.
Processed 1680 left rows.
Processed 1690 left rows.
Processed 1700 left rows.
Processed 1710 left rows.
Processed 1720 left rows.
Processed 1730 left rows.
Processed 1740 left rows.
Processed 1750 left rows.
Processed 1760 left rows.
Processed 1770 left rows.
Processed 1780 left rows.
Processed 1790 left rows.
Processed 1800 left rows.
Processed 1810 left rows.
Processed 1820 left rows.
Processed 1830 left rows.
Processed 1840 left rows.
Processed 1850 left rows.
Processed 1860 left rows.
Processed 1870 left rows.
Processed 1880 left rows.
Processed 1890 left rows.
Processed 1900 left rows.
Processed 1910 left rows.
Processed 1920 left rows.
Processed 1930 left rows.
Processed 1940 left rows.
Processed 1950 left rows.
Processed 1960 left rows.
Processed 1970 left rows.
Processed 1980 left rows.
Processed 1990 left rows.
Processed 2000 left rows.
Processed 2010 left rows.
Processed 2020 left rows.
Processed 2030 left rows.
Processed 2040 left rows.
Processed 2050 left rows.
Processed 2060 left rows.
Processed 2070 left rows.
Processed 2080 left rows.
Processed 2090 left rows.
Processed 2100 left rows.
Processed 2110 left rows.
Processed 2120 left rows.
Processed 2130 left rows.
Processed 2140 left rows.
Processed 2150 left rows.
Processed 2160 left rows.
Processed 2170 left rows.
Processed 2180 left rows.
Processed 2190 left rows.
Processed 2200 left rows.
Processed 2210 left rows.
Processed 2220 left rows.
Processed 2230 left rows.
Processed 2240 left rows.
Processed 2250 left rows.
Processed 2260 left rows.
Processed 2270 left rows.
Processed 2280 left rows.
Processed 2290 left rows.
Processed 2300 left rows.
Processed 2310 left rows.
Processed 2320 left rows.
Processed 2330 left rows.
Processed 2340 left rows.
Processed 2350 left rows.
Processed 2360 left rows.
Processed 2370 left rows.
Processed 2380 left rows.
Processed 2390 left rows.
Processed 2400 left rows.
Processed 2410 left rows.
Processed 2420 left rows.
Processed 2430 left rows.
Processed 2440 left rows.
Processed 2450 left rows.
Processed 2460 left rows.
Processed 2470 left rows.
Processed 2480 left rows.
Processed 2490 left rows.
Processed 2500 left rows.
Processed 2510 left rows.
Processed 2520 left rows.
Processed 2530 left rows.
Processed 2540 left rows.
Processed 2550 left rows.
Processed 2560 left rows.
Processed 2570 left rows.
Processed 2580 left rows.
Processed 2590 left rows.
Processed 2600 left rows.
Processed 2610 left rows.
Processed 2620 left rows.
Processed 2630 left rows.
Processed 2640 left rows.
Processed 2650 left rows.
Processed 2660 left rows.
Processed 2670 left rows.
Processed 2680 left rows.
Processed 2690 left rows.
Processed 2700 left rows.
Processed 2710 left rows.
Processed 2720 left rows.
Processed 2730 left rows.
Processed 2740 left rows.
Processed 2750 left rows.
Processed 2760 left rows.
Processed 2770 left rows.
Processed 2780 left rows.
Processed 2790 left rows.
Processed 2800 left rows.
Processed 2810 left rows.
Processed 2820 left rows.
Processed 2830 left rows.
Processed 2840 left rows.
Processed 2850 left rows.
Processed 2860 left rows.
Processed 2870 left rows.
Processed 2880 left rows.
Processed 2890 left rows.
Processed 2900 left rows.
Processed 2910 left rows.
Processed 2920 left rows.
Processed 2930 left rows.
Processed 2940 left rows.
Processed 2950 left rows.
Processed 2960 left rows.
Processed 2970 left rows.
Processed 2980 left rows.
Processed 2990 left rows.
Processed 3000 left rows.
Processed 3010 left rows.
Processed 3020 left rows.
Processed 3030 left rows.
Processed 3040 left rows.
Processed 3050 left rows.
Processed 3060 left rows.
Processed 3070 left rows.
Processed 3080 left rows.
Processed 3090 left rows.
Processed 3100 left rows.
Processed 3110 left rows.
Processed 3120 left rows.
Processed 3130 left rows.
Processed 3140 left rows.
Processed 3150 left rows.
Processed 3160 left rows.
Processed 3170 left rows.
Processed 3180 left rows.
Processed 3190 left rows.
Processed 3200 left rows.
Processed 3210 left rows.
Processed 3220 left rows.
Processed 3230 left rows.
Processed 3240 left rows.
Processed 3250 left rows.
Processed 3260 left rows.
Processed 3270 left rows.
Processed 3280 left rows.
Processed 3290 left rows.
Processed 3300 left rows.
Processed 3310 left rows.
Processed 3320 left rows.
Processed 3330 left rows.
Processed 3340 left rows.
Processed 3350 left rows.
Processed 3360 left rows.
Processed 3370 left rows.
Processed 3380 left rows.
Processed 3390 left rows.
Processed 3400 left rows.
Processed 3410 left rows.
Processed 3420 left rows.
Processed 3430 left rows.
Processed 3440 left rows.
Processed 3450 left rows.
Processed 3460 left rows.
Processed 3470 left rows.
Processed 3480 left rows.
Processed 3490 left rows.
Processed 3500 left rows.
Processed 3510 left rows.
Processed 3520 left rows.
Processed 3530 left rows.
Processed 3540 left rows.
Processed 3550 left rows.
Processed 3560 left rows.
Processed 3570 left rows.
Processed 3580 left rows.
Processed 3590 left rows.
Processed 3600 left rows.
Processed 3610 left rows.
Processed 3620 left rows.
Processed 3630 left rows.
Processed 3640 left rows.
Processed 3650 left rows.
Processed 3660 left rows.
Processed 3670 left rows.
Processed 3680 left rows.
Processed 3690 left rows.
Processed 3700 left rows.
Processed 3710 left rows.
Processed 3720 left rows.
Processed 3730 left rows.
Processed 3740 left rows.
Processed 3750 left rows.
Processed 3760 left rows.
Processed 3770 left rows.
Processed 3780 left rows.
Processed 3790 left rows.
Processed 3800 left rows.
Processed 3810 left rows.
Processed 3820 left rows.
Processed 3830 left rows.
Processed 3840 left rows.
Processed 3850 left rows.
Processed 3860 left rows.
Processed 3870 left rows.
Processed 3880 left rows.
Processed 3890 left rows.
Processed 3900 left rows.
Processed 3910 left rows.
Processed 3920 left rows.
Processed 3930 left rows.
Processed 3940 left rows.
Processed 3950 left rows.
Processed 3960 left rows.
Processed 3970 left rows.
Processed 3980 left rows.
Processed 3990 left rows.
Processed 4000 left rows.
Processed 4010 left rows.
Processed 4020 left rows.
Processed 4030 left rows.
Processed 4040 left rows.
Processed 4050 left rows.
Processed 4060 left rows.
Processed 4070 left rows.
Processed 4080 left rows.
Processed 4090 left rows.
Processed 4100 left rows.
Processed 4110 left rows.
Processed 4120 left rows.
Processed 4130 left rows.
Processed 4140 left rows.
Processed 4150 left rows.
Processed 4160 left rows.
Processed 4170 left rows.
Processed 4180 left rows.
Processed 4190 left rows.
Processed 4200 left rows.
Processed 4210 left rows.
Processed 4220 left rows.
Processed 4230 left rows.
Processed 4240 left rows.
Processed 4250 left rows.
Processed 4260 left rows.
Processed 4270 left rows.
Processed 4280 left rows.
Processed 4290 left rows.
Processed 4300 left rows.
Processed 4310 left rows.
Processed 4320 left rows.
Processed 4330 left rows.
Processed 4340 left rows.
Processed 4350 left rows.
Processed 4360 left rows.
Processed 4370 left rows.
Processed 4380 left rows.
Processed 4390 left rows.
Processed 4400 left rows.
Processed 4410 left rows.
Processed 4420 left rows.
Processed 4430 left rows.
Processed 4440 left rows.
Processed 4450 left rows.
Processed 4460 left rows.
Processed 4470 left rows.
Processed 4480 left rows.
Processed 4490 left rows.
Processed 4500 left rows.
Processed 4510 left rows.
Processed 4520 left rows.
Processed 4530 left rows.
Processed 4540 left rows.
Processed 4550 left rows.
Processed 4560 left rows.
Processed 4570 left rows.
Processed 4580 left rows.
Processed 4590 left rows.
Processed 4600 left rows.
Processed 4610 left rows.
Processed 4620 left rows.
Processed 4630 left rows.
Processed 4640 left rows.
Processed 4650 left rows.
Processed 4660 left rows.
Processed 4670 left rows.
Processed 4680 left rows.
Processed 4690 left rows.
Processed 4700 left rows.
Processed 4710 left rows.
Processed 4720 left rows.
Processed 4730 left rows.
Processed 4740 left rows.
Processed 4750 left rows.
Processed 4760 left rows.
Processed 4770 left rows.
Processed 4780 left rows.
Processed 4790 left rows.
Processed 4800 left rows.
Processed 4810 left rows.
Processed 4820 left rows.
Processed 4830 left rows.
Processed 4840 left rows.
Processed 4850 left rows.
Processed 4860 left rows.
Processed 4870 left rows.
Processed 4880 left rows.
Processed 4890 left rows.
Processed 4900 left rows.
Processed 4910 left rows.
Processed 4920 left rows.
Processed 4930 left rows.
Processed 4940 left rows.
Processed 4950 left rows.
Processed 4960 left rows.
Processed 4970 left rows.
Processed 4980 left rows.
Processed 4990 left rows.
Processed 5000 left rows.
Processed 5010 left rows.
Processed 5020 left rows.
Processed 5030 left rows.
Processed 5040 left rows.
Processed 5050 left rows.
Processed 5060 left rows.
Processed 5070 left rows.
Processed 5080 left rows.
Processed 5090 left rows.
Processed 5100 left rows.
Processed 5110 left rows.
Processed 5120 left rows.
Processed 5130 left rows.
Processed 5140 left rows.
Processed 5150 left rows.
Processed 5160 left rows.
+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| alcanis01  |     2000 | BOS      |   45 |  13 |      21 |          21 |
+------------+----------+----------+------+-----+---------+-------------+
| alexama02  |     2000 | BOS      |  194 |  41 |     101 |         101 |
+------------+----------+----------+------+-----+---------+-------------+
| arrojro01  |     2000 | BOS      |   28 |   3 |      13 |           0 |
+------------+----------+----------+------+-----+---------+-------------+
| arrojro01  |     2000 | BOS      |    0 |   0 |      13 |           0 |
+------------+----------+----------+------+-----+---------+-------------+
Duration for dumb_join =  14.285259485244751
```

[25]:
```
tab.get_access_path_test()
```

```
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'yearID', 'type': 'INDEX', 'index_name': 'yearID',
'index_order': '0'}, {'table_name': 'batting', 'column_name': 'stint', 'type':
'PRIMARY', 'index_name': 'playerID_stint_yearID', 'index_order': '1'},
{'table_name': 'batting', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
```

```
playerID_stint_yearID
5162
```

[26]:
```
tab.sub_where_template_test()
```

```
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'yearID', 'type': 'INDEX', 'index_name': 'yearID',
'index_order': '0'}, {'table_name': 'batting', 'column_name': 'stint', 'type':
'PRIMARY', 'index_name': 'playerID_stint_yearID', 'index_order': '1'},
{'table_name': 'batting', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
[{'2B': '27', '3B': '3', 'AB': '376', 'BB': '55', 'CS': '5', 'G': '101', 'GIDP':
'', 'H': '140', 'HBP': '1', 'HR': '8', 'IBB': '', 'lgID': 'AL', 'playerID':
'alexada01', 'R': '58', 'RBI': '56', 'SB': '4', 'SF': '', 'SH': '0', 'SO': '19',
'stint': '2', 'teamID': 'BOS', 'yearID': '1932'}, {'2B': '1', '3B': '0', 'AB':
'51', 'BB': '0', 'CS': '0', 'G': '27', 'GIDP': '', 'H': '7', 'HBP': '0', 'HR':
'0', 'IBB': '', 'lgID': 'AL', 'playerID': 'andreiv01', 'R': '5', 'RBI': '1',
'SB': '0', 'SF': '', 'SH': '2', 'SO': '13', 'stint': '2', 'teamID': 'BOS',
'yearID': '1932'}, {'2B': '1', '3B': '0', 'AB': '17', 'BB': '0', 'CS': '0', 'G':
'11', 'GIDP': '', 'H': '3', 'HBP': '0', 'HR': '0', 'IBB': '', 'lgID': 'AL',
'playerID': 'applepe01', 'R': '3', 'RBI': '1', 'SB': '0', 'SF': '', 'SH': '0',
'SO': '5', 'stint': '2', 'teamID': 'BOS', 'yearID': '1932'}]
```

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

```
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'yearID', 'type': 'INDEX', 'index_name': 'yearID',
'index_order': '0'}, {'table_name': 'batting', 'column_name': 'stint', 'type':
'PRIMARY', 'index_name': 'playerID_stint_yearID', 'index_order': '1'},
{'table_name': 'batting', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
[{'2B': '4', '3B': '1', 'AB': '108', 'BB': '7', 'CS': '0', 'G': '43', 'GIDP':
'1', 'H': '20', 'HBP': '1', 'HR': '0', 'IBB': '0', 'lgID': 'AL', 'playerID':
'bakerdo01', 'R': '15', 'RBI': '12', 'SB': '3', 'SF': '0', 'SH': '2', 'SO':
'22', 'stint': '1', 'teamID': 'DET', 'yearID': '1984'}]
```

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

```
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'yearID', 'type': 'INDEX', 'index_name': 'yearID',
'index_order': '0'}, {'table_name': 'batting', 'column_name': 'stint', 'type':
'PRIMARY', 'index_name': 'playerID_stint_yearID', 'index_order': '1'},
{'table_name': 'batting', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
Running load core definition appearances
Q =  select * from csvtables where table_name = %s
load core definition table/filename appearances NewAppearances.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'appearances', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'yearID_teamID_playerID', 'index_order': '0'}, {'table_name':
'appearances', 'column_name': 'teamID', 'type': 'PRIMARY', 'index_name':
'yearID_teamID_playerID', 'index_order': '1'}, {'table_name': 'appearances',
'column_name': 'playerID', 'type': 'PRIMARY', 'index_name':
'yearID_teamID_playerID', 'index_order': '2'}] <class 'list'>
+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| alcanis01  |     2000 | BOS      |   45 |  13 |      21 |          21 |
+------------+----------+----------+------+-----+---------+-------------+
| alexama02  |     2000 | BOS      |  194 |  41 |     101 |         101 |
+------------+----------+----------+------+-----+---------+-------------+
| arrojro01  |     2000 | BOS      |   28 |   3 |      13 |           0 |
+------------+----------+----------+------+-----+---------+-------------+
| arrojro01  |     2000 | BOS      |    0 |   0 |      13 |           0 |
+------------+----------+----------+------+-----+---------+-------------+
Duration for smart_join =  0.0030014514923095703 s
```

[31]:
```
# Compare the time it takes to do the dumb join and the smart join below
import time
start_time = time.time() #This is a timer that will track how long it takes to␣
 ↪execute your cell.
tab.dumb_join_test()
end_time = time.time()
print("Duration for whole dumb_join_test = ", end_time - start_time, "s")
```

```
start_time = time.time() #This is a timer that will track how long it takes to␣
↪execute your cell.
tab.smart_join_test()
end_time = time.time()
print("Duration for whole smart_join_test = ", end_time - start_time, "s")
```

```
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'yearID', 'type': 'INDEX', 'index_name': 'yearID',
'index_order': '0'}, {'table_name': 'batting', 'column_name': 'stint', 'type':
'PRIMARY', 'index_name': 'playerID_stint_yearID', 'index_order': '1'},
{'table_name': 'batting', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
Running load core definition appearances
Q =  select * from csvtables where table_name = %s
load core definition table/filename appearances NewAppearances.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'appearances', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'yearID_teamID_playerID', 'index_order': '0'}, {'table_name':
'appearances', 'column_name': 'teamID', 'type': 'PRIMARY', 'index_name':
'yearID_teamID_playerID', 'index_order': '1'}, {'table_name': 'appearances',
'column_name': 'playerID', 'type': 'PRIMARY', 'index_name':
'yearID_teamID_playerID', 'index_order': '2'}] <class 'list'>
Processed 10 left rows.
Processed 20 left rows.
Processed 30 left rows.
Processed 40 left rows.
Processed 50 left rows.
Processed 60 left rows.
Processed 70 left rows.
Processed 80 left rows.
Processed 90 left rows.
Processed 100 left rows.
Processed 110 left rows.
Processed 120 left rows.
Processed 130 left rows.
Processed 140 left rows.
Processed 150 left rows.
Processed 160 left rows.
Processed 170 left rows.
Processed 180 left rows.
Processed 190 left rows.
Processed 200 left rows.
Processed 210 left rows.
Processed 220 left rows.
Processed 230 left rows.
Processed 240 left rows.
Processed 250 left rows.
Processed 260 left rows.
Processed 270 left rows.
Processed 280 left rows.
Processed 290 left rows.
Processed 300 left rows.
Processed 310 left rows.
Processed 320 left rows.
Processed 330 left rows.
Processed 340 left rows.
Processed 350 left rows.
Processed 360 left rows.
Processed 370 left rows.
Processed 380 left rows.
Processed 390 left rows.
Processed 400 left rows.
Processed 410 left rows.
Processed 420 left rows.
Processed 430 left rows.
Processed 440 left rows.
Processed 450 left rows.
Processed 460 left rows.
Processed 470 left rows.
Processed 480 left rows.
Processed 490 left rows.
Processed 500 left rows.
Processed 510 left rows.
Processed 520 left rows.
Processed 530 left rows.
Processed 540 left rows.
Processed 550 left rows.
Processed 560 left rows.
Processed 570 left rows.
Processed 580 left rows.
Processed 590 left rows.
Processed 600 left rows.
Processed 610 left rows.
Processed 620 left rows.
Processed 630 left rows.
Processed 640 left rows.
Processed 650 left rows.
Processed 660 left rows.
Processed 670 left rows.
Processed 680 left rows.
Processed 690 left rows.
Processed 700 left rows.
Processed 710 left rows.
Processed 720 left rows.
Processed 730 left rows.
Processed 740 left rows.
Processed 750 left rows.
Processed 760 left rows.
Processed 770 left rows.
Processed 780 left rows.
Processed 790 left rows.
Processed 800 left rows.
Processed 810 left rows.
Processed 820 left rows.
Processed 830 left rows.
Processed 840 left rows.
Processed 850 left rows.
Processed 860 left rows.
Processed 870 left rows.
Processed 880 left rows.
Processed 890 left rows.
Processed 900 left rows.
Processed 910 left rows.
Processed 920 left rows.
Processed 930 left rows.
Processed 940 left rows.
Processed 950 left rows.
Processed 960 left rows.
Processed 970 left rows.
Processed 980 left rows.
Processed 990 left rows.
Processed 1000 left rows.
Processed 1010 left rows.
Processed 1020 left rows.
Processed 1030 left rows.
Processed 1040 left rows.
Processed 1050 left rows.
Processed 1060 left rows.
Processed 1070 left rows.
Processed 1080 left rows.
Processed 1090 left rows.
Processed 1100 left rows.
Processed 1110 left rows.
Processed 1120 left rows.
Processed 1130 left rows.
Processed 1140 left rows.
Processed 1150 left rows.
Processed 1160 left rows.
Processed 1170 left rows.
Processed 1180 left rows.
Processed 1190 left rows.
Processed 1200 left rows.
Processed 1210 left rows.
Processed 1220 left rows.
Processed 1230 left rows.
Processed 1240 left rows.
Processed 1250 left rows.
Processed 1260 left rows.
Processed 1270 left rows.
Processed 1280 left rows.
Processed 1290 left rows.
Processed 1300 left rows.
Processed 1310 left rows.
Processed 1320 left rows.
Processed 1330 left rows.
Processed 1340 left rows.
Processed 1350 left rows.
Processed 1360 left rows.
Processed 1370 left rows.
Processed 1380 left rows.
Processed 1390 left rows.
Processed 1400 left rows.
Processed 1410 left rows.
Processed 1420 left rows.
Processed 1430 left rows.
Processed 1440 left rows.
Processed 1450 left rows.
Processed 1460 left rows.
Processed 1470 left rows.
Processed 1480 left rows.
Processed 1490 left rows.
Processed 1500 left rows.
Processed 1510 left rows.
Processed 1520 left rows.
Processed 1530 left rows.
Processed 1540 left rows.
Processed 1550 left rows.
Processed 1560 left rows.
Processed 1570 left rows.
Processed 1580 left rows.
Processed 1590 left rows.
Processed 1600 left rows.
Processed 1610 left rows.
Processed 1620 left rows.
Processed 1630 left rows.
Processed 1640 left rows.
Processed 1650 left rows.
Processed 1660 left rows.
Processed 1670 left rows.
Processed 1680 left rows.
Processed 1690 left rows.
Processed 1700 left rows.
Processed 1710 left rows.
Processed 1720 left rows.
Processed 1730 left rows.
Processed 1740 left rows.
Processed 1750 left rows.
Processed 1760 left rows.
Processed 1770 left rows.
Processed 1780 left rows.
Processed 1790 left rows.
Processed 1800 left rows.
Processed 1810 left rows.
Processed 1820 left rows.
Processed 1830 left rows.
Processed 1840 left rows.
Processed 1850 left rows.
Processed 1860 left rows.
Processed 1870 left rows.
Processed 1880 left rows.
Processed 1890 left rows.
Processed 1900 left rows.
Processed 1910 left rows.
Processed 1920 left rows.
Processed 1930 left rows.
Processed 1940 left rows.
Processed 1950 left rows.
Processed 1960 left rows.
Processed 1970 left rows.
Processed 1980 left rows.
Processed 1990 left rows.
Processed 2000 left rows.
Processed 2010 left rows.
Processed 2020 left rows.
Processed 2030 left rows.
Processed 2040 left rows.
Processed 2050 left rows.
Processed 2060 left rows.
Processed 2070 left rows.
Processed 2080 left rows.
Processed 2090 left rows.
Processed 2100 left rows.
Processed 2110 left rows.
Processed 2120 left rows.
Processed 2130 left rows.
Processed 2140 left rows.
Processed 2150 left rows.
Processed 2160 left rows.
Processed 2170 left rows.
Processed 2180 left rows.
Processed 2190 left rows.
Processed 2200 left rows.
Processed 2210 left rows.
Processed 2220 left rows.
Processed 2230 left rows.
Processed 2240 left rows.
Processed 2250 left rows.
Processed 2260 left rows.
Processed 2270 left rows.
Processed 2280 left rows.
Processed 2290 left rows.
Processed 2300 left rows.
Processed 2310 left rows.
Processed 2320 left rows.
Processed 2330 left rows.
Processed 2340 left rows.
Processed 2350 left rows.
Processed 2360 left rows.
Processed 2370 left rows.
Processed 2380 left rows.
Processed 2390 left rows.
Processed 2400 left rows.
Processed 2410 left rows.
Processed 2420 left rows.
Processed 2430 left rows.
Processed 2440 left rows.
Processed 2450 left rows.
Processed 2460 left rows.
Processed 2470 left rows.
Processed 2480 left rows.
Processed 2490 left rows.
Processed 2500 left rows.
Processed 2510 left rows.
Processed 2520 left rows.
Processed 2530 left rows.
Processed 2540 left rows.
Processed 2550 left rows.
Processed 2560 left rows.
Processed 2570 left rows.
Processed 2580 left rows.
Processed 2590 left rows.
Processed 2600 left rows.
Processed 2610 left rows.
Processed 2620 left rows.
Processed 2630 left rows.
Processed 2640 left rows.
Processed 2650 left rows.
Processed 2660 left rows.
Processed 2670 left rows.
Processed 2680 left rows.
Processed 2690 left rows.
Processed 2700 left rows.
Processed 2710 left rows.
Processed 2720 left rows.
Processed 2730 left rows.
Processed 2740 left rows.
Processed 2750 left rows.
Processed 2760 left rows.
Processed 2770 left rows.
Processed 2780 left rows.
Processed 2790 left rows.
Processed 2800 left rows.
Processed 2810 left rows.
Processed 2820 left rows.
Processed 2830 left rows.
Processed 2840 left rows.
Processed 2850 left rows.
Processed 2860 left rows.
Processed 2870 left rows.
Processed 2880 left rows.
Processed 2890 left rows.
Processed 2900 left rows.
Processed 2910 left rows.
Processed 2920 left rows.
Processed 2930 left rows.
Processed 2940 left rows.
Processed 2950 left rows.
Processed 2960 left rows.
Processed 2970 left rows.
Processed 2980 left rows.
Processed 2990 left rows.
Processed 3000 left rows.
Processed 3010 left rows.
Processed 3020 left rows.
Processed 3030 left rows.
Processed 3040 left rows.
Processed 3050 left rows.
Processed 3060 left rows.
Processed 3070 left rows.
Processed 3080 left rows.
Processed 3090 left rows.
Processed 3100 left rows.
Processed 3110 left rows.
Processed 3120 left rows.
Processed 3130 left rows.
Processed 3140 left rows.
Processed 3150 left rows.
Processed 3160 left rows.
Processed 3170 left rows.
Processed 3180 left rows.
Processed 3190 left rows.
Processed 3200 left rows.
Processed 3210 left rows.
Processed 3220 left rows.
Processed 3230 left rows.
Processed 3240 left rows.
Processed 3250 left rows.
Processed 3260 left rows.
Processed 3270 left rows.
Processed 3280 left rows.
Processed 3290 left rows.
Processed 3300 left rows.
Processed 3310 left rows.
Processed 3320 left rows.
Processed 3330 left rows.
Processed 3340 left rows.
Processed 3350 left rows.
Processed 3360 left rows.
Processed 3370 left rows.
Processed 3380 left rows.
Processed 3390 left rows.
Processed 3400 left rows.
Processed 3410 left rows.
Processed 3420 left rows.
Processed 3430 left rows.
Processed 3440 left rows.
Processed 3450 left rows.
Processed 3460 left rows.
Processed 3470 left rows.
Processed 3480 left rows.
Processed 3490 left rows.
Processed 3500 left rows.
Processed 3510 left rows.
Processed 3520 left rows.
Processed 3530 left rows.
Processed 3540 left rows.
Processed 3550 left rows.
Processed 3560 left rows.
Processed 3570 left rows.
Processed 3580 left rows.
Processed 3590 left rows.
Processed 3600 left rows.
Processed 3610 left rows.
Processed 3620 left rows.
Processed 3630 left rows.
Processed 3640 left rows.
Processed 3650 left rows.
Processed 3660 left rows.
Processed 3670 left rows.
Processed 3680 left rows.
Processed 3690 left rows.
Processed 3700 left rows.
Processed 3710 left rows.
Processed 3720 left rows.
Processed 3730 left rows.
Processed 3740 left rows.
Processed 3750 left rows.
Processed 3760 left rows.
Processed 3770 left rows.
Processed 3780 left rows.
Processed 3790 left rows.
Processed 3800 left rows.
Processed 3810 left rows.
Processed 3820 left rows.
Processed 3830 left rows.
Processed 3840 left rows.
Processed 3850 left rows.
Processed 3860 left rows.
Processed 3870 left rows.
Processed 3880 left rows.
Processed 3890 left rows.
Processed 3900 left rows.
Processed 3910 left rows.
Processed 3920 left rows.
Processed 3930 left rows.
Processed 3940 left rows.
Processed 3950 left rows.
Processed 3960 left rows.
Processed 3970 left rows.
Processed 3980 left rows.
Processed 3990 left rows.
Processed 4000 left rows.
Processed 4010 left rows.
Processed 4020 left rows.
Processed 4030 left rows.
Processed 4040 left rows.
Processed 4050 left rows.
Processed 4060 left rows.
Processed 4070 left rows.
Processed 4080 left rows.
Processed 4090 left rows.
Processed 4100 left rows.
Processed 4110 left rows.
Processed 4120 left rows.
Processed 4130 left rows.
Processed 4140 left rows.
Processed 4150 left rows.
Processed 4160 left rows.
Processed 4170 left rows.
Processed 4180 left rows.
Processed 4190 left rows.
Processed 4200 left rows.
Processed 4210 left rows.
Processed 4220 left rows.
Processed 4230 left rows.
Processed 4240 left rows.
Processed 4250 left rows.
Processed 4260 left rows.
Processed 4270 left rows.
Processed 4280 left rows.
Processed 4290 left rows.
Processed 4300 left rows.
Processed 4310 left rows.
Processed 4320 left rows.
Processed 4330 left rows.
Processed 4340 left rows.
Processed 4350 left rows.
Processed 4360 left rows.
Processed 4370 left rows.
Processed 4380 left rows.
Processed 4390 left rows.
Processed 4400 left rows.
Processed 4410 left rows.
Processed 4420 left rows.
Processed 4430 left rows.
Processed 4440 left rows.
Processed 4450 left rows.
Processed 4460 left rows.
Processed 4470 left rows.
Processed 4480 left rows.
Processed 4490 left rows.
Processed 4500 left rows.
Processed 4510 left rows.
Processed 4520 left rows.
Processed 4530 left rows.
Processed 4540 left rows.
Processed 4550 left rows.
Processed 4560 left rows.
Processed 4570 left rows.
Processed 4580 left rows.
Processed 4590 left rows.
Processed 4600 left rows.
Processed 4610 left rows.
Processed 4620 left rows.
Processed 4630 left rows.
Processed 4640 left rows.
Processed 4650 left rows.
Processed 4660 left rows.
Processed 4670 left rows.
Processed 4680 left rows.
Processed 4690 left rows.
Processed 4700 left rows.
Processed 4710 left rows.
Processed 4720 left rows.
Processed 4730 left rows.
Processed 4740 left rows.
Processed 4750 left rows.
Processed 4760 left rows.
Processed 4770 left rows.
Processed 4780 left rows.
Processed 4790 left rows.
Processed 4800 left rows.
Processed 4810 left rows.
Processed 4820 left rows.
Processed 4830 left rows.
Processed 4840 left rows.
Processed 4850 left rows.
Processed 4860 left rows.
Processed 4870 left rows.
Processed 4880 left rows.
Processed 4890 left rows.
Processed 4900 left rows.
Processed 4910 left rows.
Processed 4920 left rows.
Processed 4930 left rows.
Processed 4940 left rows.
Processed 4950 left rows.
Processed 4960 left rows.
Processed 4970 left rows.
Processed 4980 left rows.
Processed 4990 left rows.
Processed 5000 left rows.
Processed 5010 left rows.
Processed 5020 left rows.
Processed 5030 left rows.
Processed 5040 left rows.
Processed 5050 left rows.
Processed 5060 left rows.
Processed 5070 left rows.
Processed 5080 left rows.
Processed 5090 left rows.
Processed 5100 left rows.
Processed 5110 left rows.
Processed 5120 left rows.
Processed 5130 left rows.
Processed 5140 left rows.
Processed 5150 left rows.
Processed 5160 left rows.
+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| alcanis01  |     2000 | BOS      |   45 |  13 |      21 |          21 |
+------------+----------+----------+------+-----+---------+-------------+
| alexama02  |     2000 | BOS      |  194 |  41 |     101 |         101 |
+------------+----------+----------+------+-----+---------+-------------+
| arrojro01  |     2000 | BOS      |   28 |   3 |      13 |           0 |
+------------+----------+----------+------+-----+---------+-------------+
| arrojro01  |     2000 | BOS      |    0 |   0 |      13 |           0 |
+------------+----------+----------+------+-----+---------+-------------+
Duration for dumb_join =  13.242977619171143
Duration for whole dumb_join_test =  18.293416023254395 s
Running load core definition batting
Q =  select * from csvtables where table_name = %s
load core definition table/filename batting NewBatting.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'batting', 'column_name': 'playerID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '0'}, {'table_name':
'batting', 'column_name': 'yearID', 'type': 'INDEX', 'index_name': 'yearID',
'index_order': '0'}, {'table_name': 'batting', 'column_name': 'stint', 'type':
'PRIMARY', 'index_name': 'playerID_stint_yearID', 'index_order': '1'},
{'table_name': 'batting', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'playerID_stint_yearID', 'index_order': '2'}] <class 'list'>
Running load core definition appearances
Q =  select * from csvtables where table_name = %s
load core definition table/filename appearances NewAppearances.csv
Q =  select * from csvcolumns where table_name = %s
Q =  select * from csvindexes where table_name = %s order by index_order asc
[{'table_name': 'appearances', 'column_name': 'yearID', 'type': 'PRIMARY',
'index_name': 'yearID_teamID_playerID', 'index_order': '0'}, {'table_name':
'appearances', 'column_name': 'teamID', 'type': 'PRIMARY', 'index_name':
'yearID_teamID_playerID', 'index_order': '1'}, {'table_name': 'appearances',
'column_name': 'playerID', 'type': 'PRIMARY', 'index_name':
'yearID_teamID_playerID', 'index_order': '2'}] <class 'list'>
+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| alcanis01  |     2000 | BOS      |   45 |  13 |      21 |          21 |
+------------+----------+----------+------+-----+---------+-------------+
| alexama02  |     2000 | BOS      |  194 |  41 |     101 |         101 |
+------------+----------+----------+------+-----+---------+-------------+
| arrojro01  |     2000 | BOS      |   28 |   3 |      13 |           0 |
+------------+----------+----------+------+-----+---------+-------------+
| arrojro01  |     2000 | BOS      |    0 |   0 |      13 |           0 |
+------------+----------+----------+------+-----+---------+-------------+
Duration for smart_join =  0.003036022186279297 s
Duration for whole smart_join_test =  4.930052042007446 s
```

```
[ ]:
```