W4111_HW2_Programming-Submission

October 19, 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.

1 Sidharta_AdityaKelvianto_aks2266

1.0.1 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

1.0.2 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.

1.1 Part 1: Written & SQL

1.1.1 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.

The metadata that is being stored should act similar to all of the tables that we have in the database. This allows the right people to be able to access information about the tables in the database in the same manner as querying the tables.

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

indexes, users, user groups

3. What is the SQL database catalog called?

System catalog

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

It aims to find rows with the associated rows that are associated with the corresponding indices efficiently

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

All Primary Key is a Unique Index, while not all Unique Index is a primary key. Primary Key is not nullable, while Unique Index is nullable. There can me multiple Unique Indices, while there can only be one Primary key.

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 first one is more efficient because The first one only rely on the indices of the second table to filter the rows that needs to be included in the first table, while the second one combined both of the tables first (which takes a lot of time), and then perform the subsequent filtering and duplicate removal (by using group by)

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?

ON DELETE CASCADE is used to make sure that the parent table (reference table) delete a row, then the row on the child table (referencing table) with the corresponding values will be deleted as well. In this case, whenever specific table_name and column_name in `csvtables` are deleted, rows in `csvcolumns` and `csvindexes` that have the corresponding `table_name` and `column_name` foreign key will be deleted.

ON DELETE RESTICT is used when we don't allow the user to delete a row that are referenced in other table. We use this to prevent accidental deletion, when the corresponding information is referenced in other places

1.1.2 SQL

[1]:
```
%reload_ext sql
%sql mysql+pymysql://root:password@127.0.0.1/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:

[2]:
```
%%sql
SELECT CONCAT(IFNULL(SUBSTRING(nameFirst, 1, 1), ""), SUBSTRING(nameLast, 1,␣
↪1)) as Initials
FROM people
ORDER BY nameFirst, nameLast
LIMIT 10
```

```
 * mysql+pymysql://root:***@127.0.0.1/lahmansbaseballdb
10 rows affected.
```

[2]:
```
[('B',),
 ('B',),
 ('C',),
 ('E',),
 ('E',),
 ('F',),
 ('G',),
 ('H',),
 ('H',),
 ('H',)]
```

1.2 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:

[3]:
```
%%sql

SELECT yearID, lgID, FLOOR(SUM(G_all) / COUNT(DISTINCT playerID)) AS␣
 ↪games_per_player
FROM appearances
GROUP BY yearID, lgID
LIMIT 10
```

```
 * mysql+pymysql://root:***@127.0.0.1/lahmansbaseballdb
10 rows affected.
```

[3]:
```
[(1871, 'NA', Decimal('19')),
 (1872, 'NA', Decimal('22')),
 (1873, 'NA', Decimal('29')),
 (1874, 'NA', Decimal('34')),
 (1875, 'NA', Decimal('33')),
 (1876, 'NL', Decimal('38')),
 (1877, 'NL', Decimal('35')),
 (1878, 'NL', Decimal('43')),
 (1879, 'NL', Decimal('48')),
 (1880, 'NL', Decimal('48'))]
```

1.3 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

[4]:
```
%load_ext autoreload
%autoreload 2
```

[5]:
```
import unit_test_catalog as cat # This notebook should be in the same directory␣
 ↪as your project
```

[6]:
```
cat.drop_table_test()
```

```
Q =  DELETE FROM csvtables WHERE table_name = 'test_table'
Table 'test_table' was dropped
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
```

```
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

Table =  in the index if statement
{
  "table_name": "test_table",
  "file_name": null,
  "columns": [],
  "indexes": []
}
```

[7]:
```
cat.create_table_test()
```

```
Running save core definition
Q =  insert into csvtables values(%s, %s)
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

Table =  in the index if statement
{
  "table_name": "test_table",
  "file_name": "file_path_test.woo",
  "columns": [],
  "indexes": []
}
```

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

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s
```

```
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Table =  in the index if statement
{
  "table_name": "test_table",
  "file_name": "file_path_test.woo",
  "columns": [
    {
      "column_name": "foo",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": []
}
```

[9]:
```
cat.column_name_failure_test()  # This will throw an error
```

```
issue!!
```

```
---------------------------------------------
ValueError    Traceback (most recent call last)
<ipython-input-9-342044887ceb> in <module>
----> 1 cat.column_name_failure_test()  # This will throw an error

~/columbia/database/hw2/W4111_HW2_Programming/unit_test_catalog.py in␣
 ↪column_name_failure_test()
     57         dbpw="adit2511",
     58         db="CSVCatalog")
---> 59     col = CSVCatalog.ColumnDefinition(None, "text", False)
     60     t = cat.get_table("test_table")
     61     t.add_column_definition(col)

~/columbia/database/hw2/W4111_HW2_Programming/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!!
```

[10]:
```
cat.column_type_failure_test()  # This will throw an error
```

```
Issue!
```

```
---------------------------------------------
ValueError    Traceback (most recent call last)
<ipython-input-10-eebf587b1ffc> in <module>
----> 1 cat.column_type_failure_test()  # This will throw an error

~/columbia/database/hw2/W4111_HW2_Programming/unit_test_catalog.py in␣
 ↪column_type_failure_test()
     72         dbpw="adit2511",
     73         db="CSVCatalog")
---> 74     col = CSVCatalog.ColumnDefinition("bird", "canary", False)
     75     t = cat.get_table("test_table")
     76     t.add_column_definition(col)

~/columbia/database/hw2/W4111_HW2_Programming/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.
```

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

```
issue!
---------------------------------------------
ValueError    Traceback (most recent call last)
<ipython-input-11-9b5701466b82> in <module>
----> 1 cat.column_not_null_failure_test()  # This will throw an error

~/columbia/database/hw2/W4111_HW2_Programming/unit_test_catalog.py in␣
 ↪column_not_null_failure_test()
     87         dbpw="adit2511",
     88         db="CSVCatalog")
---> 89     col = CSVCatalog.ColumnDefinition("name", "text", "happy")
     90     t = cat.get_table("test_table")
     91     t.add_column_definition(col)

~/columbia/database/hw2/W4111_HW2_Programming/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.
```

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

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

Q =  insert into csvcolumns values(%s, %s, %s, %s)
insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('test_table', 'bar', 'UNIQUE', 'barunique', '0')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Table =  in the index if statement
{
  "table_name": "test_table",
  "file_name": "file_path_test.woo",
  "columns": [
    {
      "column_name": "foo",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "bar",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": [
    {
      "index_name": "barunique",
      "type": "UNIQUE",
      "columns": [
```

```
        "bar"
      ]
    }
  ]
}
```

[13]:
```
cat.col_drop_test()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

Q =  DELETE FROM csvcolumns WHERE table_name = %s and column_name = %s
Column 'foo' has been dropped!
Table =  in the index if statement
{
  "table_name": "test_table",
  "file_name": "file_path_test.woo",
  "columns": [
    {
      "column_name": "bar",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": [
    {
      "index_name": "barunique",
      "type": "UNIQUE",
      "columns": [
        "bar"
      ]
    }
  ]
}
```

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

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
```

```
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

Q =
        select column_name, type, index_order
        from csvindexes
        where table_name = %s
        and index_name = %s

Table =  in the index if statement
{
  "table_name": "test_table",
  "file_name": "file_path_test.woo",
  "columns": [
    {
      "column_name": "bar",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": []
}
```

[18]:
```
cat.describe_table_test()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

in the index if statement
DESCRIBE People =
 {
  "table_name": "test_table",
  "file_name": "file_path_test.woo",
```

```
  "columns": [
    {
      "column_name": "bar",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": []
}
```

1.4 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.

[25]:
```
import unit_test_csv_table as tab
```

[39]:
```
# 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
```

[40]:
```
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)
```

[41]:
```
tab.update_people_columns()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
```

```
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %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)
Q =  insert into csvcolumns values(%s, %s, %s, %s)
Table =  in the index if statement
{
  "table_name": "people",
  "file_name": "People.csv",
  "columns": [
    {
      "column_name": "playerID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "birthYear",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "birthMonth",
      "column_type": "text",
      "not_null": false
    },
```

```
{
  "column_name": "birthDay",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "birthCountry",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "birthState",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "birthCity",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "deathYear",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "deathMonth",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "deathDay",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "deathCountry",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "deathState",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "deathCity",
  "column_type": "text",
```

```
  "not_null": false
},
{
  "column_name": "nameFirst",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "nameLast",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "nameGiven",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "weight",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "height",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "bats",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "throws",
  "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": "retroID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "bbrefID",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": []
}
```

[42]:
```
tab.update_appearances_columns()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %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": "appearances",
  "file_name": "Appearances.csv",
  "columns": [
    {
      "column_name": "yearID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "teamID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "lgID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "playerID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_all",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "GS",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_batting",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_defense",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_p",
      "column_type": "text",
```

```
  "not_null": false
},
{
  "column_name": "G_c",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_1b",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_2b",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_3b",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_ss",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_lf",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_cf",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_rf",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_of",
  "column_type": "text",
  "not_null": false
},
{
```

```
      "column_name": "G_dh",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_ph",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_pr",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": []
}
```

[43]:
```
tab.update_batting_columns()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %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",
  "file_name": "Batting.csv",
  "columns": [
    {
      "column_name": "playerID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "yearID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "stint",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "teamID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "lgID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G",
      "column_type": "text",
      "not_null": false
    },
    {
      "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": []
}
```

[44]:
```
tab.add_index_definitions()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('batting', 'playerID', 'PRIMARY', 'battingprimary',
'0')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
```

```
insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('batting', 'yearID', 'PRIMARY', 'battingprimary',
'1')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('batting', 'stint', 'PRIMARY', 'battingprimary',
'2')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Table =  in the index if statement
{
  "table_name": "batting",
  "file_name": "Batting.csv",
  "columns": [
    {
      "column_name": "2B",
      "column_type": "text",
      "not_null": false
    },
    {
      "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": false
    },
    {
      "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": false
},
{
  "column_name": "playerID",
  "column_type": "text",
  "not_null": false
},
{
  "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": false
    },
    {
      "column_name": "teamID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "yearID",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": [
    {
      "index_name": "battingprimary",
      "type": "PRIMARY",
      "columns": [
        "playerID",
        "yearID",
        "stint"
      ]
    }
  ]
}
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s
```

```
Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('appearances', 'yearID', 'PRIMARY',
'appearancesprimary', '0')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('appearances', 'teamID', 'PRIMARY',
'appearancesprimary', '1')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('appearances', 'playerID', 'PRIMARY',
'appearancesprimary', '2')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Table =  in the index if statement
{
  "table_name": "appearances",
  "file_name": "Appearances.csv",
  "columns": [
    {
      "column_name": "G_1b",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_2b",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_3b",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_all",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_batting",
```

```
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_c",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_cf",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_defense",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_dh",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_lf",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_of",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_p",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_ph",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "G_pr",
  "column_type": "text",
  "not_null": false
},
```

```
    {
      "column_name": "G_rf",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "G_ss",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "GS",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "lgID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "playerID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "teamID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "yearID",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": [
    {
      "index_name": "appearancesprimary",
      "type": "PRIMARY",
      "columns": [
        "yearID",
        "teamID",
        "playerID"
      ]
    }
  ]
}
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('people', 'playerID', 'PRIMARY', 'peopleprimary',
'0')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Table =  in the index if statement
{
  "table_name": "people",
  "file_name": "People.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": false
    },
    {
      "column_name": "birthDay",
      "column_type": "text",
      "not_null": false
    },
    {
```

```
  "column_name": "birthMonth",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "birthState",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "birthYear",
  "column_type": "text",
  "not_null": false
},
{
  "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": false
    },
    {
      "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": "peopleprimary",
      "type": "PRIMARY",
      "columns": [
        "playerID"
      ]
    }
  ]
}
```

[45]:
```
tab.test_load_info()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

People.csv
```

[46]:
```
tab.test_get_col_names()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

['bats', 'bbrefID', 'birthCity', 'birthCountry', 'birthDay', 'birthMonth',
'birthState', 'birthYear', 'deathCity', 'deathCountry', 'deathDay',
'deathMonth', 'deathState', 'deathYear', 'debut', 'finalGame', 'height',
'nameFirst', 'nameGiven', 'nameLast', 'playerID', 'retroID', 'throws', 'weight']
```

[47]:
```
tab.add_other_indexes()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
```

```
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('batting', 'teamID', 'INDEX', 'battingindex', '0')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Table =  in the index if statement
{
  "table_name": "batting",
  "file_name": "Batting.csv",
  "columns": [
    {
      "column_name": "2B",
      "column_type": "text",
      "not_null": false
    },
    {
      "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": false
```

```
},
{
  "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": false
},
{
  "column_name": "playerID",
  "column_type": "text",
  "not_null": false
},
{
  "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": false
    },
    {
      "column_name": "teamID",
      "column_type": "text",
      "not_null": false
    },
    {
      "column_name": "yearID",
      "column_type": "text",
      "not_null": false
    }
  ],
  "indexes": [
    {
      "index_name": "battingprimary",
      "type": "PRIMARY",
      "columns": [
        "playerID",
        "yearID",
        "stint"
      ]
    },
    {
      "index_name": "battingindex",
      "type": "INDEX",
      "columns": [
```

```
        "teamID"
      ]
    }
  ]
}
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('people', 'nameLast', 'INDEX', 'peopleindex', '0')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
insert into csvindexes (table_name, column_name, type, index_name, index_order)
values(%s, %s, %s, %s, %s) ('people', 'nameFirst', 'INDEX', 'peopleindex', '1')
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Table =  in the index if statement
{
  "table_name": "people",
  "file_name": "People.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": false
},
{
  "column_name": "birthDay",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "birthMonth",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "birthState",
  "column_type": "text",
  "not_null": false
},
{
  "column_name": "birthYear",
  "column_type": "text",
  "not_null": false
},
{
  "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": false
},
{
  "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": "peopleprimary",
      "type": "PRIMARY",
      "columns": [
        "playerID"
      ]
    },
    {
      "index_name": "peopleindex",
      "type": "INDEX",
      "columns": [
        "nameLast",
        "nameFirst"
      ]
    }
  ]
}
```

[54]:
```
# This should throw an error
# Make sure it works properly when you run it in pycharm though!
tab.load_test()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|   2B |   3B |   AB |   BB |   CS |   G | GIDP   |   H | HBP   |   HR | IBB   |
lgID   | playerID   |   R |   RBI |   SB | SF   | SH   |   SO |   stint | teamID
|   yearID |
```

0	0	4	0	0	1		0		0	
NA	abercda01	0	0	0			0	1	TRO	
1871										
6	0	118	4	1	25		32		0	
NA	addybo01	30	13	8			0	1	RC1	
1871										
4	5	137	2	1	29		40		0	
NA	allisar01	28	19	3			5	1	CL1	
1871										
10	2	133	0	1	27		44		2	
NA	allisdo01	28	27	1			2	1	WS3	
1871										
11	3	120	2	2	25		39		0	
NA	ansonca01	29	16	6			1	1	RC1	
1871										
2	1	49	0	1	12		11		0	
NA	armstbo01	9	5	0			1	1	FW1	
1871										
0	0	4	1	0	1		1		0	
NA	barkeal01	0	2	0			0	1	RC1	
1871										
10	9	157	13	6	31		63		0	
NA	barnero01	66	34	11			1	1	BS1	
1871										

```
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
|    1 |    0 |    5 |    0 |    0 |   1 |        |   1 |       |    0 |       |
NA     | barrebi01  |   1 |     1 |    0 |      |      |    0 |       1 | FW1
|     1871 |
+------+------+------+------+------+-----+--------+-----+-------+------+-------+
--------+------------+-----+-------+------+------+------+------+---------+------
----+----------+
```

[66]:
```
# Might throw an error depending on table size
# Make sure it works properly when you run it in pycharm though!
tab.dumb_join_test()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| barrebi01  |     1871 | FW1      |    5 |   1 |       1 |           1 |
+------------+----------+----------+------+-----+---------+-------------+
```

[83]:
```
tab.get_access_path_test()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
```

```
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

('battingprimary', 'abercda01_1871_1')
1
```

[84]:
```
tab.sub_where_template_test()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

{'playerID': 'adi'}
```

[85]:
```
tab.test_find_by_template_index()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

[{'2B': '0', '3B': '0', 'AB': '4', 'BB': '0', 'CS': '0', 'G': '1', 'GIDP': '',
'H': '0', 'HBP': '', 'HR': '0', 'IBB': '', 'lgID': 'NA', 'playerID':
'abercda01', 'R': '0', 'RBI': '0', 'SB': '0', 'SF': '', 'SH': '', 'SO': '0',
'stint': '1', 'teamID': 'TRO', 'yearID': '1871'}]
```

[86]:
```
tab.smart_join_test()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| barrebi01  |     1871 | FW1      |    5 |   1 |       1 |           1 |
+------------+----------+----------+------+-----+---------+-------------+
```

[94]:
```
# 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()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
```

```
        from csvindexes
        where table_name = %s

Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| barrebi01  |     1871 | FW1      |    5 |   1 |       1 |           1 |
+------------+----------+----------+------+-----+---------+-------------+
CPU times: user 72.9 ms, sys: 4 µs, total: 73 ms
Wall time: 72.6 ms
```

[96]:
```
# 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()
```

```
Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

Loading core definition
Q =  select path from csvtables where table_name = %s
Q =
        select column_name, type, not_null
```

```
        from csvcolumns
        where table_name = %s

Q =
        select column_name, type, index_name, index_order
        from csvindexes
        where table_name = %s

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.
+------------+----------+----------+------+-----+---------+-------------+
| playerID   |   yearID | teamID   |   AB |   H |   G_all |   G_batting |
+============+==========+==========+======+=====+=========+=============+
| barrebi01  |     1873 | BL1      |    4 |   1 |       1 |           1 |
+------------+----------+----------+------+-----+---------+-------------+
| barrebi01  |     1872 | WS3      |    4 |   0 |       1 |           1 |
+------------+----------+----------+------+-----+---------+-------------+
| barrebi01  |     1871 | FW1      |    5 |   1 |       1 |           1 |
+------------+----------+----------+------+-----+---------+-------------+
CPU times: user 703 ms, sys: 16 ms, total: 719 ms
Wall time: 705 ms
```

[ ]: