W4111_HW2_Programming

October 22, 2021

COMS W4111-002 (Fall 2021)Introduction to Databases

Homework 2: Programming Implement a Simple Database Engine15 Points

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

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

<span style="font-size: 20pt; line-height: 1.2"; > Yi_Yang, yy3089

0.1.1 Submission

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

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

0.1.2 Collaboration

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

0.2 Part 1: Written & SQL

0.2.1 Written

Please keep your answers brief.

1. Codd's Fourth Rule states that: The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data. In two sentences please explain this rule and why it is so important.

Metadata stored in the data dictionary should obey all the characteristics of a database and it should have correct up to date data. We should be able to access these metadata by using same query language that we use to access the database.

https://www.tutorialcup.com/dbms/codds-rule.htm#Active_Online_Catalog

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

base tables, views, indexes

3. What is the SQL database catalog called?

INFORMATION_SCHEMA

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

An index contains keys built from one or more columns in the table or view. These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently. Clustered indexes sort and store the data rows in the table or view based on their key values.

https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-ver15

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

Primary key will not accept NULL values whereas Unique key can accept NULL values. A table can have only primary key whereas there can be multiple unique key on a table. A Clustered index is automatically created when a primary key is defined whereas Unique key generates the non-clustered index.

https://www.geeksforgeeks.org/difference-between-primary-key-and-unique-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.

I think the second one is better because in the first query we have to go through all two tables, while in the second query we only have to go through one joined table.

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?

CASCADE means that the child data is either deleted or updated when the parent data is deleted or updated. ON DELETE CASCADE means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted.

0.2.2 SQL

[1]:
```
%load_ext sql
%sql mysql+pymysql://root:dbuserdbuser@localhost/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(substr(nameFirst,1,1),'_'),␣
 ↪ifnull(substr(nameLast,1,1),'_')) as initials,
ifnull(nameFirst, '____') as firstName,
nameLast as lastName
from lahmansbaseballdb.people
order by firstName, lastName
limit 10
```

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

[2]:
```
[('_B', '____', 'Boland'),
 ('_B', '____', 'Booth'),
 ('_C', '____', 'Carroll'),
 ('_E', '____', 'Edwards'),
 ('_E', '____', 'Evans'),
 ('_F', '____', 'Franklin'),
 ('_G', '____', 'Gavern'),
 ('_H', '____', 'Harrison'),
 ('_H', '____', 'Hellings'),
 ('_H', '____', 'Higby')]
```

[ ]:
```

```

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

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

Answer:

[3]:
```
%%sql
SELECT yearID, lgID, round(sum(G_all)/count(distinct playerID)) as␣
↪games_per_player FROM lahmansbaseballdb.appearances
group by yearID, lgID
limit 10
```

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

[3]:
```
[(1871, 'NA', Decimal('20')),
 (1872, 'NA', Decimal('22')),
 (1873, 'NA', Decimal('30')),
 (1874, 'NA', Decimal('35')),
 (1875, 'NA', Decimal('33')),
 (1876, 'NL', Decimal('39')),
 (1877, 'NL', Decimal('35')),
 (1878, 'NL', Decimal('43')),
 (1879, 'NL', Decimal('49')),
 (1880, 'NL', Decimal('48'))]
```

[ ]:
```
```

0.4 Part 2: CSVCatalog Tests

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

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

[4]:
```
import unit_test_catalog as cat # This notebook should be in the same directory␣
↪as your project
```

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

```
Running save core definition
Q =  insert into csvtables values(%s, %s)
Running load core definition
Q =  select * from csvtables where table_name = 'test_table'
Running load columns
Q =  select * from csvcolumns where table_name = 'test_table'
Running load indexes
Q =  select * from csvindexes where table_name = 'test_table' group by
index_name, table_name, type, column_name, index_order order by index_order
Table =  {
  "table_name": "test_table",
  "file_name": "./Appearances.csv"
```

```
}
```

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

```
Q =  DELETE FROM csvtables WHERE table_name = 'test_table'
Table 'test_table' was dropped
Drop test_table
```

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

```
Running save core definition
Q =  insert into csvtables values(%s, %s)
Running load core definition
Q =  select * from csvtables where table_name = 'test_table'
Running load columns
Q =  select * from csvcolumns where table_name = 'test_table'
Running load indexes
Q =  select * from csvindexes where table_name = 'test_table' group by
index_name, table_name, type, column_name, index_order order by index_order
Table =  {
  "table_name": "test_table",
  "file_name": "./Appearances.csv"
}
```

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'test_table'
Running load columns
Q =  select * from csvcolumns where table_name = 'test_table'
Running load indexes
Q =  select * from csvindexes where table_name = 'test_table' group by
index_name, table_name, type, column_name, index_order order by index_order
Q =  insert into csvcolumns values(%s, %s, %s, %s)
```

[9]:
```
try:
    cat.column_name_failure_test()  # This will throw an error
except ValueError as e:
    print(e)
```

```
issue!!
You must have a column name!!
```

[10]:
```
try:
    cat.column_type_failure_test()  # This will throw an error
except ValueError as e:
    print(e)
```

```
Issue!
That column type is not accepted. Please try again.
```

[11]:
```
try:
    cat.column_not_null_failure_test()  # This will throw an error
except ValueError as e:
    print(e)
```

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

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'test_table'
Running load columns
Q =  select * from csvcolumns where table_name = 'test_table'
Running load indexes
Q =  select * from csvindexes where table_name = 'test_table' group by
index_name, table_name, type, column_name, index_order order by index_order
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
```

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'test_table'
Running load columns
Q =  select * from csvcolumns where table_name = 'test_table'
Running load indexes
Q =  select * from csvindexes where table_name = 'test_table' group by
index_name, table_name, type, column_name, index_order order by index_order
Q =  delete from csvcolumns where table_name = 'test_table' and column_name =
'test_column'
Column 'test_column' has been dropped!
```

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'test_table'
Running load columns
Q =  select * from csvcolumns where table_name = 'test_table'
Running load indexes
Q =  select * from csvindexes where table_name = 'test_table' group by
index_name, table_name, type, column_name, index_order order by index_order
Running drop index
Q =  delete from csvindexes where table_name = 'test_table' and index_name =
'test_index'
```

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'test_table'
Running load columns
Q =  select * from csvcolumns where table_name = 'test_table'
Running load indexes
Q =  select * from csvindexes where table_name = 'test_table' group by
index_name, table_name, type, column_name, index_order order by index_order
DESCRIBE test_table =
 {
  "table_name": "test_table",
  "file_name": "./Appearances.csv"
}
```

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

```
Q =  DELETE FROM csvtables WHERE table_name = 'test_table'
Table 'test_table' was dropped
Drop test_table
```

0.5 Part 3: CSVTable Tests

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

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

[17]:
```
import unit_test_csv_table as tab
```

[18]:
```
# 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
```

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

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

[20]:

```
try:
    tab.update_people_columns()
except Exception as e:
    print(e)
```

```
Running load core definition
Q =  select * from csvtables where table_name = 'people'
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
DataTableException: code: -200 , message: Updated not implemented
```

[21]:

```
try:
    tab.update_appearances_columns()
except Exception as e:
    print(e)
```

```
Running load core definition
Q =  select * from csvtables where table_name = 'appearances'
Running load columns
Q =  select * from csvcolumns where table_name = 'appearances'
Running load indexes
Q =  select * from csvindexes where table_name = 'appearances' group by
index_name, table_name, type, column_name, index_order order by index_order
DataTableException: code: -200 , message: Updated not implemented
```

[22]:
```
try:
    tab.update_batting_columns()
except Exception as e:
    print(e)
```

```
Running load core definition
Q =  select * from csvtables where table_name = 'batting'
Running load columns
Q =  select * from csvcolumns where table_name = 'batting'
Running load indexes
Q =  select * from csvindexes where table_name = 'batting' group by index_name,
table_name, type, column_name, index_order order by index_order
DataTableException: code: -200 , message: Updated not implemented
```

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'people'
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Running load core definition
Q =  select * from csvtables where table_name = 'batting'
Running load columns
Q =  select * from csvcolumns where table_name = 'batting'
Running load indexes
Q =  select * from csvindexes where table_name = 'batting' group by index_name,
table_name, type, column_name, index_order order by index_order
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
Running load core definition
Q =  select * from csvtables where table_name = 'appearances'
Running load columns
Q =  select * from csvcolumns where table_name = 'appearances'
Running load indexes
Q =  select * from csvindexes where table_name = 'appearances' group by
index_name, table_name, type, column_name, index_order order by index_order
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
```

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'people'
```

```
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
./People.csv
```

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'people'
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
['bats', 'bbrefID', 'birthCity', 'birthCountry', 'birthDay', 'birthMonth',
'birthState', 'birthYear', 'deathCity', 'deathCountry', 'deathDay',
'deathMonth', 'deathState', 'deathYear', 'debut', 'finalGame', 'height',
'nameFirst', 'nameGiven', 'nameLast', 'playerID', 'retroID', 'throws', 'weight']
```

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'people'
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
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)
Running load core definition
Q =  select * from csvtables where table_name = 'appearances'
Running load columns
Q =  select * from csvcolumns where table_name = 'appearances'
Running load indexes
Q =  select * from csvindexes where table_name = 'appearances' group by
index_name, table_name, type, column_name, index_order order by index_order
Q =  insert into csvindexes (table_name, column_name, type, index_name,
index_order)  values(%s, %s, %s, %s, %s)
```

[27]:
```
# This should throw an error
# Make sure it works properly when you run it in pycharm though!
tab.load_test()
```

```
Running load core definition
Q =  select * from csvtables where table_name = 'batting'
Running load columns
Q =  select * from csvcolumns where table_name = 'batting'
Running load indexes
Q =  select * from csvindexes where table_name = 'batting' group by index_name,
table_name, type, column_name, index_order order by index_order

IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

[28]:
```
# 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
Q =  select * from csvtables where table_name = 'people'
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
Running load core definition
Q =  select * from csvtables where table_name = 'appearances'
Running load columns
Q =  select * from csvcolumns where table_name = 'appearances'
Running load indexes
Q =  select * from csvindexes where table_name = 'appearances' group by
index_name, table_name, type, column_name, index_order order by index_order
Processed 1000 left rows.
Processed 2000 left rows.
Processed 3000 left rows.
Processed 4000 left rows.
Processed 5000 left rows.
Processed 6000 left rows.
Processed 7000 left rows.
Processed 8000 left rows.
Processed 9000 left rows.
Processed 10000 left rows.
Processed 11000 left rows.
```

```
Processed 12000 left rows.
Processed 13000 left rows.
Processed 14000 left rows.
Processed 15000 left rows.
Processed 16000 left rows.
Processed 17000 left rows.
Processed 18000 left rows.
Processed 19000 left rows.
+------------+----------+----------+-------------+------------+---------+
| playerID   |   yearID | teamID   | nameFirst   | nameLast   |   G_all |
+============+==========+==========+=============+============+=========+
| baxtemi01  |     2010 | SDN      | Mike        | Baxter     |       9 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2011 | NYN      | Mike        | Baxter     |      22 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2012 | NYN      | Mike        | Baxter     |      89 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2013 | NYN      | Mike        | Baxter     |      74 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2014 | LAN      | Mike        | Baxter     |       4 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2015 | CHN      | Mike        | Baxter     |      34 |
+------------+----------+----------+-------------+------------+---------+
```

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'batting'
Running load columns
Q =  select * from csvcolumns where table_name = 'batting'
Running load indexes
Q =  select * from csvindexes where table_name = 'batting' group by index_name,
table_name, type, column_name, index_order order by index_order
primary_index
18915
```

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

```
Running load core definition
Q =  select * from csvtables where table_name = 'people'
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
{'nameFirst': 'Hank', 'nameLast': 'Aaron'}
```

[31]:
```
tab.test_find_by_template_index()
```

```
Running load core definition
Q =  select * from csvtables where table_name = 'people'
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
[
  {
    "bats": "R",
    "bbrefID": "aaronha01",
    "birthCity": "Mobile",
    "birthCountry": "USA",
    "birthDay": "5",
    "birthMonth": "2",
    "birthState": "AL",
    "birthYear": "1934",
    "deathCity": "",
    "deathCountry": "",
    "deathDay": "",
    "deathMonth": "",
    "deathState": "",
    "deathYear": "",
    "debut": "1954-04-13",
    "finalGame": "1976-10-03",
    "height": "72",
    "nameFirst": "Hank",
    "nameGiven": "Henry Louis",
    "nameLast": "Aaron",
    "playerID": "aaronha01",
    "retroID": "aaroh101",
    "throws": "R",
    "weight": "180"
  }
]
```

[32]:
```
tab.smart_join_test()
```

```
Running load core definition
Q =  select * from csvtables where table_name = 'people'
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
Running load core definition
Q =  select * from csvtables where table_name = 'appearances'
Running load columns
```

```
Q =  select * from csvcolumns where table_name = 'appearances'
Running load indexes
Q =  select * from csvindexes where table_name = 'appearances' group by
index_name, table_name, type, column_name, index_order order by index_order
+------------+----------+----------+-------------+------------+---------+
| playerID   |   yearID | teamID   | nameFirst   | nameLast   |   G_all |
+============+==========+==========+=============+============+=========+
| baxtemi01  |     2010 | SDN      | Mike        | Baxter     |       9 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2011 | NYN      | Mike        | Baxter     |      22 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2012 | NYN      | Mike        | Baxter     |      89 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2013 | NYN      | Mike        | Baxter     |      74 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2014 | LAN      | Mike        | Baxter     |       4 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2015 | CHN      | Mike        | Baxter     |      34 |
+------------+----------+----------+-------------+------------+---------+
```

[33]:
```
# Compare the time it takes to do the dumb join and the smart join below
%time #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----
```

```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 4.77 µs
```

[34]:
```
%%time
tab.dumb_join_test()
```

```
Running load core definition
Q =  select * from csvtables where table_name = 'people'
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
Running load core definition
Q =  select * from csvtables where table_name = 'appearances'
Running load columns
Q =  select * from csvcolumns where table_name = 'appearances'
Running load indexes
Q =  select * from csvindexes where table_name = 'appearances' group by
index_name, table_name, type, column_name, index_order order by index_order
```

```
Processed 1000 left rows.
Processed 2000 left rows.
Processed 3000 left rows.
Processed 4000 left rows.
Processed 5000 left rows.
Processed 6000 left rows.
Processed 7000 left rows.
Processed 8000 left rows.
Processed 9000 left rows.
Processed 10000 left rows.
Processed 11000 left rows.
Processed 12000 left rows.
Processed 13000 left rows.
Processed 14000 left rows.
Processed 15000 left rows.
Processed 16000 left rows.
Processed 17000 left rows.
Processed 18000 left rows.
Processed 19000 left rows.
+------------+----------+----------+-------------+------------+---------+
| playerID   |   yearID | teamID   | nameFirst   | nameLast   |   G_all |
+============+==========+==========+=============+============+=========+
| baxtemi01  |     2010 | SDN      | Mike        | Baxter     |       9 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2011 | NYN      | Mike        | Baxter     |      22 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2012 | NYN      | Mike        | Baxter     |      89 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2013 | NYN      | Mike        | Baxter     |      74 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2014 | LAN      | Mike        | Baxter     |       4 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2015 | CHN      | Mike        | Baxter     |      34 |
+------------+----------+----------+-------------+------------+---------+
CPU times: user 21min 23s, sys: 4.95 s, total: 21min 28s
Wall time: 21min 35s
```

[35]:
```
%%time
tab.smart_join_test()
```

```
Running load core definition
Q =  select * from csvtables where table_name = 'people'
Running load columns
Q =  select * from csvcolumns where table_name = 'people'
Running load indexes
Q =  select * from csvindexes where table_name = 'people' group by index_name,
table_name, type, column_name, index_order order by index_order
Running load core definition
```

```
Q =  select * from csvtables where table_name = 'appearances'
Running load columns
Q =  select * from csvcolumns where table_name = 'appearances'
Running load indexes
Q =  select * from csvindexes where table_name = 'appearances' group by
index_name, table_name, type, column_name, index_order order by index_order
+------------+----------+----------+-------------+------------+---------+
| playerID   |   yearID | teamID   | nameFirst   | nameLast   |   G_all |
+============+==========+==========+=============+============+=========+
| baxtemi01  |     2010 | SDN      | Mike        | Baxter     |       9 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2011 | NYN      | Mike        | Baxter     |      22 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2012 | NYN      | Mike        | Baxter     |      89 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2013 | NYN      | Mike        | Baxter     |      74 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2014 | LAN      | Mike        | Baxter     |       4 |
+------------+----------+----------+-------------+------------+---------+
| baxtemi01  |     2015 | CHN      | Mike        | Baxter     |      34 |
+------------+----------+----------+-------------+------------+---------+
CPU times: user 57.6 s, sys: 354 ms, total: 58 s
Wall time: 58.1 s
```