4111_s21_hw1_programming

September 29, 2021

COMS W4111-002 (Fall 2021)Introduction to Databases

Homework 1: Programming - 10 Points

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

1 Suri_Chandan, CS4090

1.1 Introduction

1.1.1 Objectives

This homework has you practice and build skill with:

- PART A: (1 point) Understanding relational databases
- PART B: (1 point) Understanding relational algebra
- PART C: (1 point) Cleaning data
- PART D: (1 point) Performing simple SQL queries to analyze the data.
- PART E: (6 points) CSVDataTable.py

Note: The motivation for PART E may not be clear. The motivation will become clearer as the semester proceeds. The purpose of PART E is to get you started on programming in Python and manipulating data.

1.1.2 Submission

1. File > Print Preview > Download as PDF
2. Upload .pdf and .ipynb to GradeScope
3. Upload CSVDataTable.py and CSVDataTable_Tests.py

This assignment is due September 24, 11:59 pm ET

1.1.3 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.

2 Part A: Written

1. What is a database management system?

"Database Management System" is defined as a software package or a complete system that can do the following tasks:

"Define" the data and it's fields/files' relationships in a database.

"Manipulate" the data or the structure of the data like fields names in a database.

"Retrieve" the data in many ways the user might want to from the database.

"Manage" the database.

2. What is a primary key and why is it important?

   A primary key is a special type of column (or a set of columns) in a relational database table that is used to uniquely identify a row/record in a table. A primary key serves as a unique identifier as such it must contain a unique value for each row/record in the table. This is very useful when you would have a lot of data and want to quickly identify or retrieve some specific row/record in a database as using that primary key, being unique, you can uniquely get that record from the databse.

3. Please explain the differences between SQL, MySQL Server and DataGrip?

   SQL is the query language designed and used for defining, managing, manipulating and retireving data from the database or a relational database management system. However, MySQL is the relational database that uses SQL for querying and do all the operations SQL can dp on that database. So, MySQL is essentially the database that stores the existing data in a database in an organized manner as defined by the SQL. There is also a workbench that comes with MySQL although some professional developers use some other IDE for that. So, DataGrip is basically a cross-platform database IDE for professional SQL developers who use it for working with SQL and the databases.

4. What are 4 different types of DBMS table relationships, give a brief explanaition for each?

   The 4 different types of DBMS table relationships are as follows:

   One to One (1:1) : Let's say if you have two tables that follow this relationship, then, both the tables can only have one corresponding record/row on either side of the relationship. Precisely, when each record or row of one table is related to only one record of the another table, then, it's known as a 1:1 relationship. This is generally enforced by business rules and that's when you would see this kind of a relationship. This is not naturally occuring as essentially you could combine those 2 tables into one without breaking any of the normalization rules, so, it's generally when some business rules enforce you to do so and keep both the tables separate. For example, such a relationship would be between a table of an employee (employee_id, employee_name, employee_job_grade, employee_team_name, employee_designation) at a company and the salaries (salary_in_usd, bonus_in_usd) of those employees. This could be done this way, so, that the employee data would be available to any other employee in the company but that employee's salary data shouldn't be public. You could also write rules for this while querying the database but sometimes, it's enforced by company's business rules and for some security reasons.

   One to Many : If you have two tables in your database where each row/record in the 1st table can be related to many rows/records in the 2nd table but not the other way around, then that relationship between those tables is known as "One to Many" relationship. For example, let's say the 1st table is Parents (father_name, mother_name) table and you have

a 2nd table as Childrens (child_name) table, then basically a set of parents can have multiple children but a child can only have a single parent set as the child's parents (taking this as an assumption!).

Many to One : This is essentially the opposite case of the 2nd kind of relationship defined above. So, if you have two tables where each row/record in the 1st table is related to one row/record in the 2nd table but each record in the 2nd table can be related to multiple rows in the 1st table, then, that's known as the "Many to One" relationship. For example, the 1st table could be the students (student_id, student_name, student_gpa) table and the 2nd table could be the departments (department_name, department_code) table of a school/college, then, a department can have multiple students related to it but a student will be related to a single department only of that school/college.

Many to Many: This kind of relationship exists when many rows of the 1st table can be related to one or more than one record in the 2nd table and vice-versa. For example, if the 1st table is the Professors (professor_id, professor_name, professor_office_address, professor_email_id) table and the 2nd table is the Students (student_id, student_name, student_email_id) table, then a professor can teach multiple students and multiple students can be taught by multiple professors, so, that's a many to many relationship between the two tables.

5. What is an ER model?

   An ER model (entity-relationship model) describes the interrelated entities of interest in a specific domain of knowledge. Basically, it's composed of entity types (which classifies the things of interest) and specifies the relationships that can exist between those entities (instances of those instance types). Generally, it's shown as an ER diagram that diagramatically defines those entities and their relationships with each other.

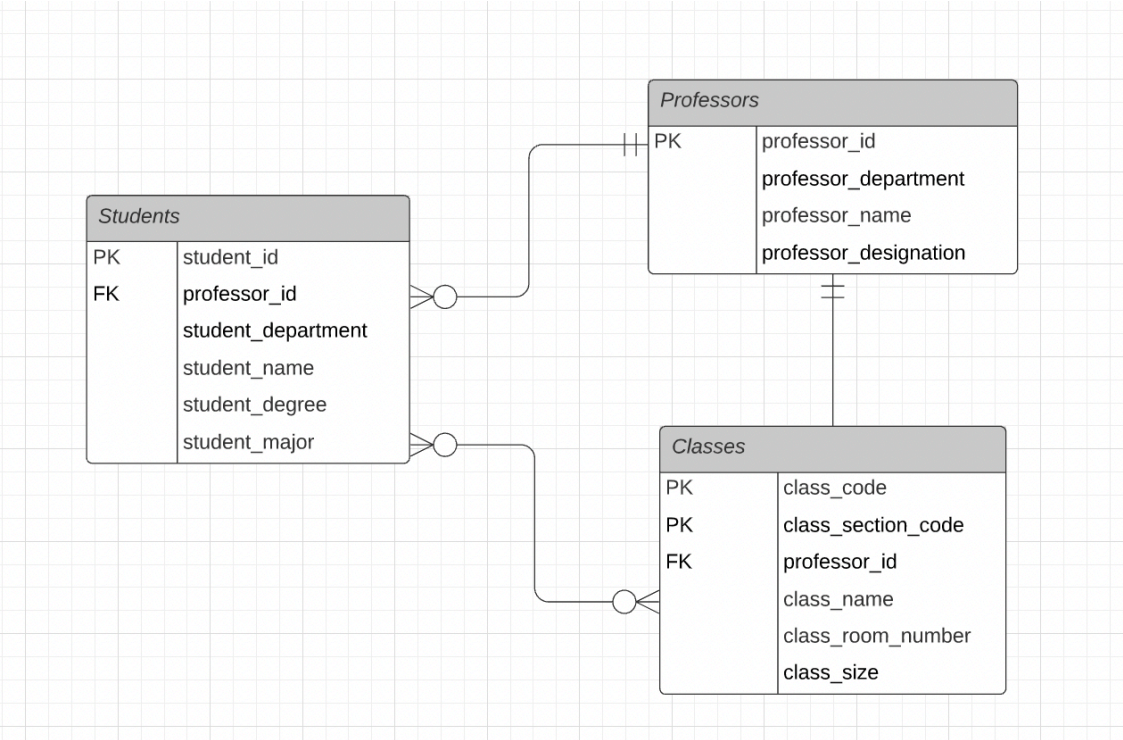
6. Using Lucidchart draw an example of a logical ER model using Crow's Foot notation for Columbia classes. The entity types are:
   - Students, Professors, and Classes.
   - The relationships are:
     - A Class has exactly one Professor.
     - A Student has exactly one professor who is an *advisor.*
     - A Professor may advise 0, 1 or many Students.
     - A Class has 0, 1 or many enrolled students.
     - A Student enrolls in 0, 1 or many Classes.

- You can define what you think are common attributes for each of the entity types. Do not define more than 5 or 6 attributes per entity type.
- In this example, explicitly show an example of a primary-key, foreign key, one-to-many relationship, and many-to-many relationship.

Notes: - If you have not already done so, please register for a free account at Lucidchart.com. You can choose the option at the bottom of the left pane to add the ER diagram shapes. - You can take a screen capture of you diagram and save in the zip directory that that contains you Jupyter notebook. Edit the following cell and replace "Boromir.jpg" with the name of the file containing your screenshot.

[2]:
```
from IPython.display import Image
Image("ER_Diagram_HW_1.png")
```

[2]:



3 Part B: Relational Algebra

You will use the online relational calculator, choose the "Karlsruhe University of Applied Sciences" dataset.

An anti-join is a form of join with reverse logic. Instead of returning rows when there is a match (according to the join predicate) between the left and right side, an anti-join returns those rows from the left side of the predicate for which there is no match on the right.

The Anti-Join Symbol is .

Consider the following relational algebra expression and result.

/* (1) Set X = The set of classrooms in buildings Taylor or Watson. */

```
X =  building='Watson'  building='Taylor' (classroom)
```

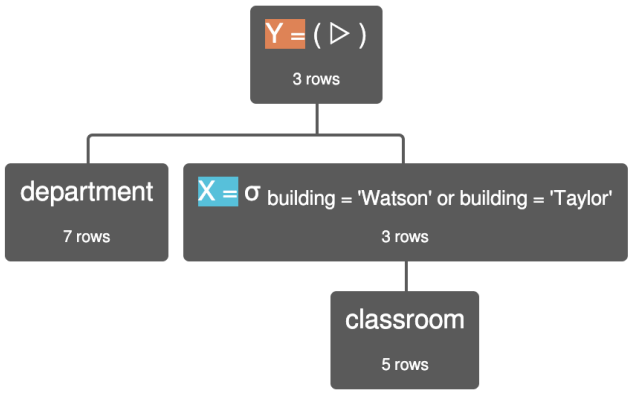
/* (2) Set Y = The Anti-Join of department and X */

```
Y = (department  X)
```

/* (3) Display the rows in Y. */

```
Y
```

[1]:
```
from IPython.display import Image
Image("ra.png")
```
[1]:

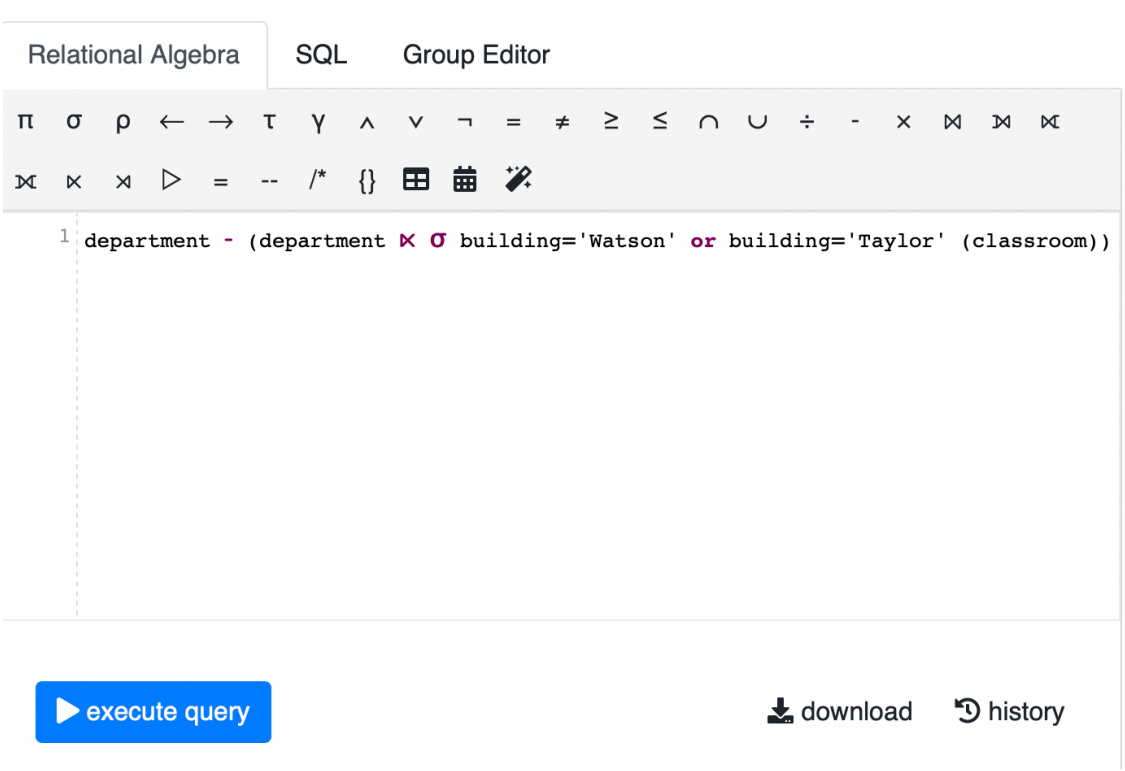


(department ▷ σ building = 'Watson' or building = 'Taylor' (classroom))

department.dept_name	department.building	department.budget
'Finance'	'Painter'	120000
'History'	'Painter'	50000
'Music'	'Packard'	80000

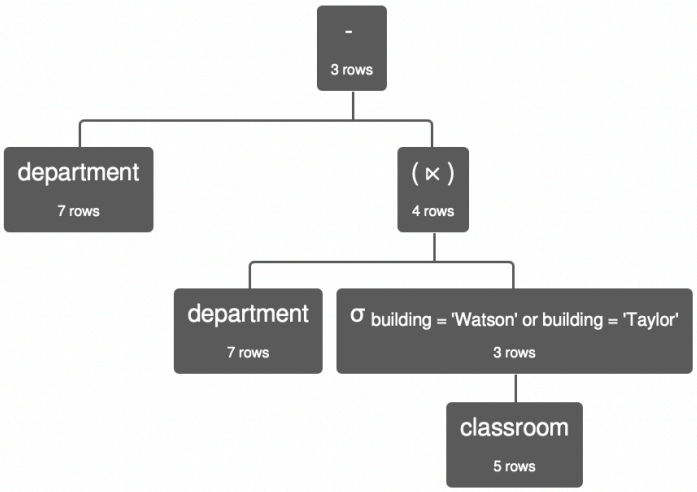
1. Find an alternate expression to (2) that computes the correct answer given X. Display the execution of your query below.

[4]:
```
from IPython.display import Image
Image("Anti_Join_Alternative2_Query.png")
```
[4]:



[5]:
```
Image("Anti_Join_Alternative2_Query_Solution.png")
```

[5]:



department - (department ∝ σ building = 'Watson' or building = 'Taylor' (classroom))

department.dept_name	department.building	department.budget
'Finance'	'Painter'	120000
'History'	'Painter'	50000
'Music'	'Packard'	80000

4 Part C: Data Clean Up

4.1 Please note: You MUST make a new schema using the lahmansdb_to_clean.sql file provided in the data folder.

Use thelahmansdb_to_clean.sql file to make a new schema containing the raw data. The lahman database you created in Homework 0 has already been cleaned with all the constraints and will be used for Part D. Knowing how to clean data and add integrity constraints is very important which is why you go through the steps in part C.

TLDR: If you use the HW0 lahman schema for this part you will get a lot of errors and recieve a lot of deductions.

[10]:
```
# You will need to follow instructions from HW 0 to make a new schema, import␣
↪the data.
# Connect to the unclean schema below by setting the database host, user ID and␣
↪password.
```

```
%load_ext sql
%sql mysql+pymysql://root:dbuserdbuser@localhost/lahmansdb_to_clean
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

Data cleanup: For each table we want you to clean, we have provided a list of changes you have to make. You can reference the cleaned lahman db for inspiration and guidance, but know that there are different ways to clean the data and you will be graded for your choice rationalization. You should make these changes through DataGrip's workbench's table editor and/or using SQL queries. In this part you will clean two tables: People and Batting.

4.1.1 You must have:

- A brief explanation of why you think the change we requested is important.
- What change you made to the table.
- Any queries you used to make the changes, either the ones you wrote or the Alter statements provided by DataGrip's editor.
- Executed the test statements we provided
- The cleaned table's new create statement (after you finish all the changes)

4.1.2 Overview of Changes:

People Table

0. Primary Key (Explanation is given, but you still must add the key to your table yourself)
1. Empty strings to NULLs
2. Column typing
3. isDead column
4. deathDate and birthDate column

Batting Table

1. Empty strings to NULLs
2. Column typing
3. Primary Key
4. Foreign Key

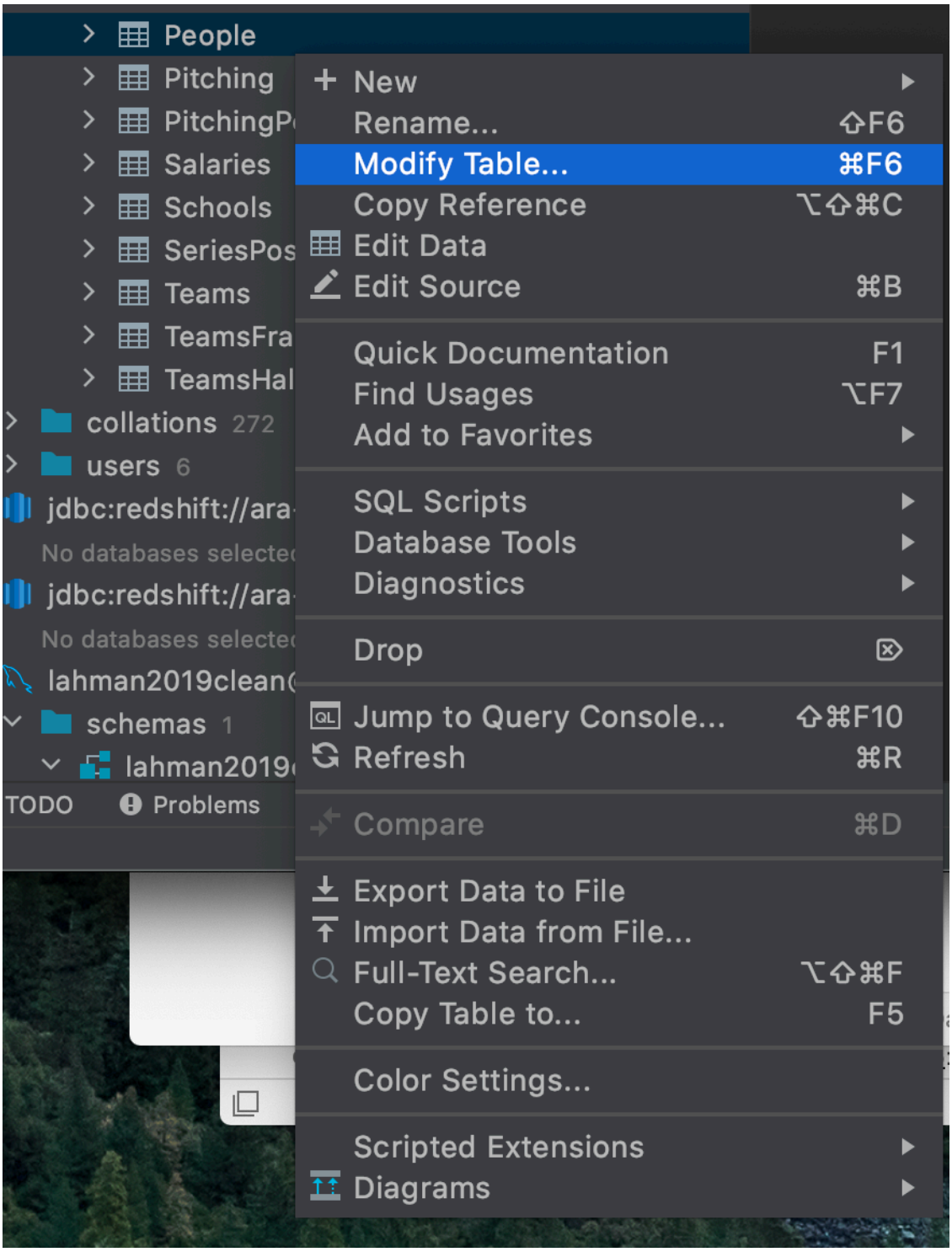
4.1.3 How to make the changes:

Using the Table Editor:

When you hit apply, a popup will open displaying the ALTER statments sql generates. Copy the sql provided first and paste it into this notebook. Then you can apply the changes. This means that you are NOT executing the ALTER statements through your notebook.

1. Right click on the table > Modify Table…

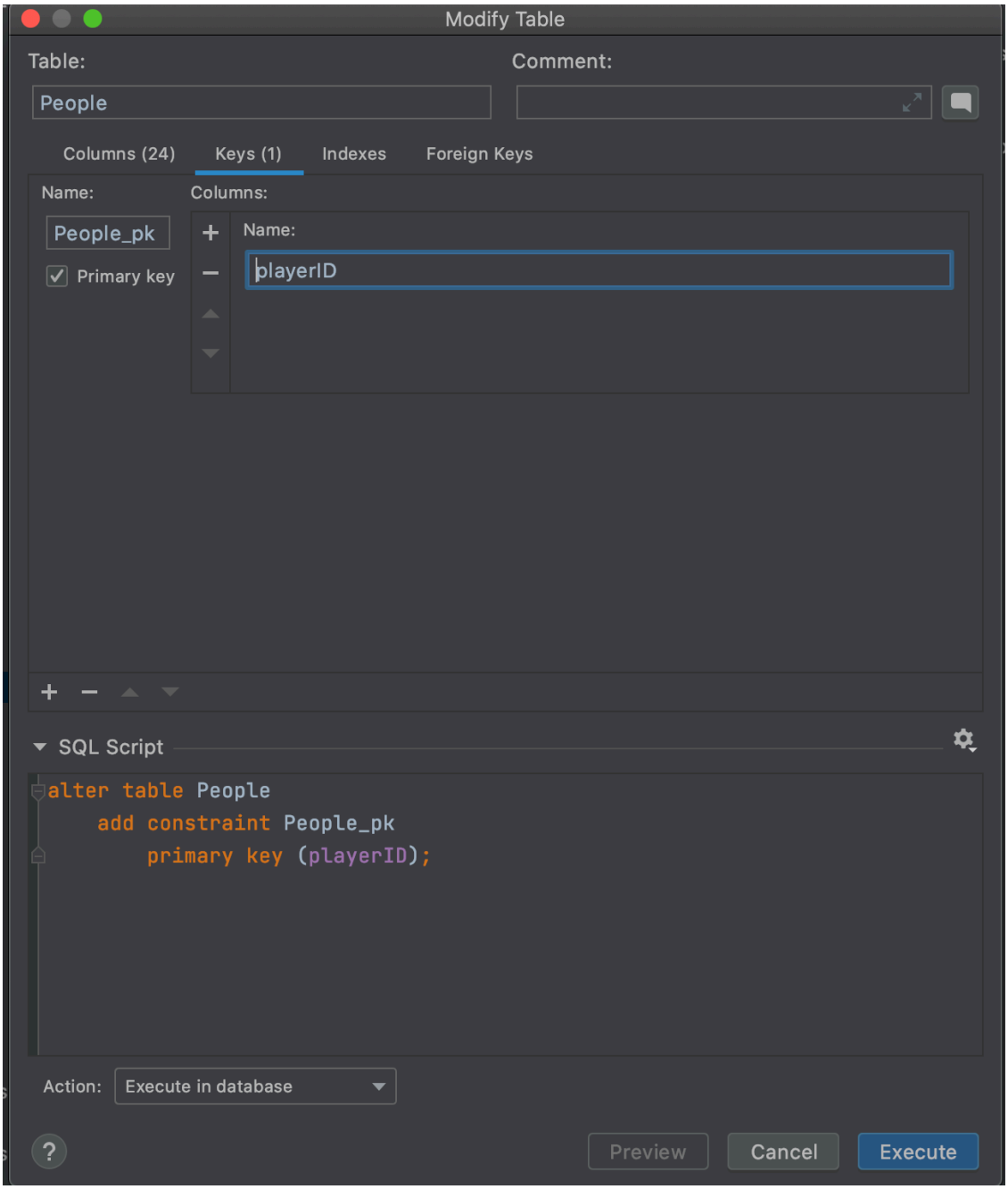
[2]:
```
Image("modify.png")
```
[2]:



2. Keys > press the + button > input the parameters > Execute OR Keys > press the + button > input the parameters > copy and paste the script generated under "SQL Script" and paste into your notebook > Run the cell in jupyter notebook

[3]:
```
Image("pk.png")
```

[3]:



Using sql queries:

Copy paste any queries that you write manually into the notebook as well!

4.2 People Table

4.2.1 0) EXAMPLE: Add a Primary Key

(Solutions are given but make sure you still do this step in workbench!)

Explanation We want to add a Primary Key because we want to be able to uniquely identify rows within our data. A primary key is also an index, which allows us to locate data faster.

Change I added a Primary Key on the playerID column and made the datatype VARCHAR(15)

Note: This is for demonstration purposes only. playerID **is not** a primary key for fielding.

SQL

```
ALTER TABLE `lahmansdb_to_clean`.`people`
CHANGE COLUMN `playerID` `playerID` VARCHAR(15) NOT NULL ,
ADD PRIMARY KEY (`playerID`);
```

Tests

[11]:
```
%sql SHOW KEYS FROM people WHERE Key_name = 'PRIMARY'
```

```
 * mysql+pymysql://root:***@localhost/lahmansdb_to_clean
1 rows affected.
```

[11]:
```
[('people', 0, 'PRIMARY', 1, 'playerID', 'A', 19502, None, None, '', 'BTREE',
'', '', 'YES', None)]
```

4.2.2 1) Convert all empty strings to NULL

Explanation Empty strings in a database can mean many things but NULL is a reference variable and that shows that the value is missing or has not been entered in the database. However, when you find an empty string for some of the features/columns in the database, it might not tell you exactly what that empty value means. To remove the disambiguation, we change the empty strings to NULL value. Furthermore, it's easier to check for NULL values and work with them accordingly than some random text in there, be it be an empty string.

Change I have replaced all the empty strings across all the columns in all the rows of the database with a NULL reference variable. So, for any of the columns in the database, there isn't any empty string but in it's place, there is a NULL value. For example, "deathDay", "deathMonth" and "deathYear" before would have some empty strings, which now are replaced by NULL values in there suggesting that the person would still be alive and thus, the data is not entered.

SQL

```
UPDATE people
SET
    birthYear = CASE birthYear WHEN '' THEN NULL ELSE birthYear END,
    birthMonth = CASE birthMonth WHEN '' THEN NULL ELSE birthMonth END,
    birthDay = CASE birthDay WHEN '' THEN NULL ELSE birthDay END,
    birthCountry = CASE birthCountry WHEN '' THEN NULL ELSE birthCountry END,
```

```
birthState = CASE birthState WHEN '' THEN NULL ELSE birthState END,
birthCity = CASE birthCity WHEN '' THEN NULL ELSE birthCity END,
deathYear = CASE deathYear WHEN '' THEN NULL ELSE deathYear END,
deathMonth = CASE deathMonth WHEN '' THEN NULL ELSE deathMonth END,
deathDay = CASE deathDay WHEN '' THEN NULL ELSE deathDay END,
deathCountry = CASE deathCountry WHEN '' THEN NULL ELSE deathCountry END,
deathState = CASE deathState WHEN '' THEN NULL ELSE deathState END,
deathCity = CASE deathCity WHEN '' THEN NULL ELSE deathCity END,
nameFirst = CASE nameFirst WHEN '' THEN NULL ELSE nameFirst END,
nameLast = CASE nameLast WHEN '' THEN NULL ELSE nameLast END,
nameGiven = CASE nameGiven WHEN '' THEN NULL ELSE nameGiven END,
weight = CASE weight WHEN '' THEN NULL ELSE weight END,
height = CASE height WHEN '' THEN NULL ELSE height END,
bats = CASE bats WHEN '' THEN NULL ELSE bats END,
throws = CASE throws WHEN '' THEN NULL ELSE throws END,
debut = CASE debut WHEN '' THEN NULL ELSE debut END,
finalGame = CASE finalGame WHEN '' THEN NULL ELSE finalGame END,
retroID = CASE retroID WHEN '' THEN NULL ELSE retroID END,
bbrefID = CASE bbrefID WHEN '' THEN NULL ELSE bbrefID END;
```

Tests

[16]:
```
%sql SELECT * FROM people WHERE birthState = ""
```

```
 * mysql+pymysql://root:***@localhost/lahmansdb_to_clean
0 rows affected.
```

[16]: []

4.2.3 2) Change column datatypes to appropriate values (ENUM, INT, VARCHAR, DATETIME, ETC)

Explanation The datatypes are really important part of the database schema and tells us the domain of the values that can be entered in that column in the database. This also prevents users from entering an invalid data in the field with a datatype that doesn't match the data entered by the user. For example, for "weight" and "height" fields, some user might enter some decimal values or might try to enter the value in words, but that query wouldn't go through if we have defined those fields as "SMALLINT". Even so, the user wouldn't be able to add some nonsensical value such as 100000 as the weight or height values are generally quite less and the datatypes help define us the bounds or the domain of those fields. This is really important to maintain a clean database and consistent database. This consistency can also help when you need to perform some kind of operations on the field, such as for an int field, when the sorting is done, you would get the sorted integer values in numerically ascending order but for a varchar column, it's sorted according to the ascii values of the characters in the string.

Change I have modified the datatypes for the columns in the table 'people'. - For 'birthYear' the datatype is changed to 'SMALLINT' specifying the domain of the values for the birthYear to be integers and less than or equal to 65535 (unsigned integers). - For 'birthMonth' and 'birthDay' the datatype is changed to TINYINT as the month values and day values are always integers

having values from 1-12 and 1-31 respectively. - For 'birthCountry', 'birthState' and 'birthCity', the datatype is changed to a variable string having at most 100 characters. Mostly, the characters needed for these fields would be much less than 100 characters, but just to be safe, I have taken it to be 100 so, that no problems arise when we are adding more data! - For 'deathYear' the datatype is changed to 'SMALLINT' specifying the domain of the values for the birthYear to be integers and less than or equal to 65535 (unsigned integers). - For 'deathMonth' and 'deathDay' the datatype is changed to TINYINT as the month values and day values are always integers having values from 1-12 and 1-31 respectively. - For 'deathCountry', 'deathState' and 'deathCity', the datatype is changed to a variable string having at most 100 characters. Mostly, the characters needed for these fields would be much less than 100 characters, but just to be safe, I have taken it to be 100 so, that no problems arise when we are adding more data! - For 'nameFirst', 'nameLast' and 'nameGiven', the datatype is changed to a variable string having at most 100 characters. Mostly, the characters needed for these fields would be much less than 100 characters, but just to be safe, I have taken it to be 100 so, that no problems arise when we are adding more data! Maybe someone can have long names too! - For 'weight' and 'height' fields, the datatype is changed to SMALLINT to reflect the fact that these values need to be integers and always less than 65535 (unsigned). I couldn't take TINYINT as sometimes, it can be greater than 255 and we need to cover those cases too if and when more data is added to the database. - For 'bats' and 'throws', the datatype changed to an ENUM with some specific enumerations possible. For these fields, there are only 3 enumerations possible (other than NULL!), so, we specify those in the datatype itself so, no invalid value can be entered to the database and we know precisely what all values can be entered for these fields. - For 'debut' and 'finalGame', the datatype is changed to DATETIME to specify the fact that these are dates (maybe time could be added too!) and thus, specifying the type of value could help maintain the consistency of the data. - For 'retroID' and 'bbrefID', the datatype is changed to variable strings (VARHCAR) to reflect the fact that these are variable strings and can have at most 50 characters. Even though the number characters is less than 50 in the current database but it's good to leave some room considering the scalability and maintainability of the database.

I know that for some of the columns in there, 100 would be a lot for the number of characters that could be added for that column but still it's good to leave some room considering the maintainability aspect of the database. However, I also know that we can change the schema of the database anytime using ALTER command, but, it's good that we don't use it a lot.

SQL

```
ALTER TABLE people
    MODIFY COLUMN birthYear SMALLINT,
    MODIFY COLUMN birthMonth TINYINT,
    MODIFY COLUMN birthDay TINYINT,
    MODIFY COLUMN birthCountry VARCHAR(100),
    MODIFY COLUMN birthState VARCHAR(100),
    MODIFY COLUMN birthCity VARCHAR(100),
    MODIFY COLUMN deathYear SMALLINT,
    MODIFY COLUMN deathMonth TINYINT,
    MODIFY COLUMN deathDay TINYINT,
    MODIFY COLUMN deathCountry VARCHAR(100),
    MODIFY COLUMN deathState VARCHAR(100),
    MODIFY COLUMN deathCity VARCHAR(100),
```

```
MODIFY COLUMN nameFirst VARCHAR(100),
MODIFY COLUMN nameLast VARCHAR(100),
MODIFY COLUMN nameGiven VARCHAR(100),
MODIFY COLUMN weight SMALLINT,
MODIFY COLUMN height SMALLINT,
MODIFY COLUMN bats ENUM('L', 'R', 'B'),
MODIFY COLUMN throws ENUM('L', 'R', 'S'),
MODIFY COLUMN debut DATETIME,
MODIFY COLUMN finalGame DATETIME,
MODIFY COLUMN retroID VARCHAR(50),
MODIFY COLUMN bbrefID VARCHAR(50)
```

4.2.4 3) Add an isDead Column that is either 'Y' or 'N'

- Some things to think of: What data type should this column be? How do you know if the player is dead or not? Maybe you do not know if the player is dead.
- You do not need to make guesses about life spans, etc. Just apply a simple rule.

'Y' means the player is dead

'N' means the player is alive

Explanation These type of boolean or ENUM fields can be very useful for data analysis purposes so, that you know some of the facts about the database stored. This could easily let us know whether the person is dead or not without computing the same from the deathYear, deathMonth and deathDay at the runtime. Such type of fields provide us with an intuition about the database in an effective manner and can be quite useful to store in the database for analytical purposes.

Change I have changed the database by adding an 'isDead" column to it which is specified to be of ENUM type that can hold only 2 values other than NULL default value, as 'Y' - YES and 'N' - NO. Then, I update the table 'people' to calculate the values in the 'isDead' column based on some of the other columns in the database like deathYear, deathMonth and deathDay to correctly reflect the state of the person being dead or not. So, I take the assumption that if no values are present for either of the 3 fields mentioned above, then the person is not dead and that's why it's a NULL value in there. Following that assumption, if deathYear is present and is less than the current year, then for sure the person is dead but if deathYear is the current Year, then deathMonth and deathDay should be less than the current date to state the person to be Dead thus, the value in that case will be 'Y' for the "isDead" column otherwise it will be 'N'.

SQL

```
ALTER TABLE people ADD COLUMN isDead ENUM('Y', 'N') DEFAULT NULL;
UPDATE people
SET people.isDead = (CASE
                        WHEN deathYear IS NULL OR deathMonth IS NULL
                        OR deathDay IS NULL
                        THEN 'N'
                        WHEN deathYear < 2021 OR (deathYear = 2021
                             AND deathMonth <= 9 AND deathDay <= 29)
```

```
                    THEN 'Y'
                    END);
```

Tests

[18]:
```
%sql SELECT * FROM people WHERE isDead = "N" limit 10
```

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

[18]:
```
[('aardsda01', 1981, 12, 27, 'USA', 'CO', 'Denver', None, None, None, None,
None, None, 'David', 'Aardsma', 'David Allan', 215, 75, 'R', 'R',
datetime.datetime(2004, 4, 6, 0, 0), datetime.datetime(2015, 8, 23, 0, 0),
'aardd001', 'aardsda01', 'N'),
 ('aaronha01', 1934, 2, 5, 'USA', 'AL', 'Mobile', None, None, None, None, None,
None, 'Hank', 'Aaron', 'Henry Louis', 180, 72, 'R', 'R', datetime.datetime(1954,
4, 13, 0, 0), datetime.datetime(1976, 10, 3, 0, 0), 'aaroh101', 'aaronha01',
'N'),
 ('aasedo01', 1954, 9, 8, 'USA', 'CA', 'Orange', None, None, None, None, None,
None, 'Don', 'Aase', 'Donald William', 190, 75, 'R', 'R',
datetime.datetime(1977, 7, 26, 0, 0), datetime.datetime(1990, 10, 3, 0, 0),
'aased001', 'aasedo01', 'N'),
 ('abadan01', 1972, 8, 25, 'USA', 'FL', 'Palm Beach', None, None, None, None,
None, None, 'Andy', 'Abad', 'Fausto Andres', 184, 73, 'L', 'L',
datetime.datetime(2001, 9, 10, 0, 0), datetime.datetime(2006, 4, 13, 0, 0),
'abada001', 'abadan01', 'N'),
 ('abadfe01', 1985, 12, 17, 'D.R.', 'La Romana', 'La Romana', None, None, None,
None, None, None, 'Fernando', 'Abad', 'Fernando Antonio', 235, 74, 'L', 'L',
datetime.datetime(2010, 7, 28, 0, 0), datetime.datetime(2019, 9, 28, 0, 0),
'abadf001', 'abadfe01', 'N'),
 ('abbotgl01', 1951, 2, 16, 'USA', 'AR', 'Little Rock', None, None, None, None,
None, None, 'Glenn', 'Abbott', 'William Glenn', 200, 78, 'R', 'R',
datetime.datetime(1973, 7, 29, 0, 0), datetime.datetime(1984, 8, 8, 0, 0),
'abbog001', 'abbotgl01', 'N'),
 ('abbotje01', 1972, 8, 17, 'USA', 'GA', 'Atlanta', None, None, None, None,
None, None, 'Jeff', 'Abbott', 'Jeffrey William', 190, 74, 'R', 'L',
datetime.datetime(1997, 6, 10, 0, 0), datetime.datetime(2001, 9, 29, 0, 0),
'abboj002', 'abbotje01', 'N'),
 ('abbotji01', 1967, 9, 19, 'USA', 'MI', 'Flint', None, None, None, None, None,
None, 'Jim', 'Abbott', 'James Anthony', 200, 75, 'L', 'L',
datetime.datetime(1989, 4, 8, 0, 0), datetime.datetime(1999, 7, 21, 0, 0),
'abboj001', 'abbotji01', 'N'),
 ('abbotku01', 1969, 6, 2, 'USA', 'OH', 'Zanesville', None, None, None, None,
None, None, 'Kurt', 'Abbott', 'Kurt Thomas', 180, 71, 'R', 'R',
datetime.datetime(1993, 9, 7, 0, 0), datetime.datetime(2001, 4, 13, 0, 0),
'abbok002', 'abbotku01', 'N'),
 ('abbotky01', 1968, 2, 18, 'USA', 'MA', 'Newburyport', None, None, None, None,
None, None, 'Kyle', 'Abbott', 'Lawrence Kyle', 200, 76, 'L', 'L',
datetime.datetime(1991, 9, 10, 0, 0), datetime.datetime(1996, 8, 24, 0, 0),
```

```
 'abbok001', 'abbotky01', 'N')]
```

4.2.5 4) Add a deathDate and birthDate column

Some things to think of: What do you do if you are missing information? What datatype should this column be?

You have to create this column from other columns in the table.

Explanation Generally, full dates can enable us to perform various type of queries involving operations on dates quite easily, rather than spanning multiple columns which would mean a more complicated query to do a task on the dates for the data present in the database. For example, if you want to compute the number of days between deathDate of the persons in the database and the current date, then you just need to subtract those 2 in your query and you would get the answer but when the date is broken up into multiple columns, a more involved query would be needed to subtract the date from the current date to get the number of days. Moreover, storing data in a more precise way can also reduce memory overheads and also reduce the complexity of the data. Thus, it's very important to store the "deathDate" and "birthDate" as a datetime in columns specified as such.

Change Firstly, I add 2 extra columns to the database known as "birthDate" and "deathDate" as "DateTime" columns meaning that these columns specify some kind of a date and time. Secondly, when those 2 columns have been added to the 'people' table, I compute the birthdate as a concatenated version of the other columns present in the table, "birthYear", "birthMonth" and "birthDay". Similarly, the deathDate is computed as the concatenated version of the other columns present in the table, "deathYear", "deathMonth" and "deathDay". This gives us the datetime type columns as "year-month-day".

SQL

```
ALTER TABLE people
    ADD COLUMN birthDate DATETIME,
    ADD COLUMN deathDate DATETIME;
UPDATE people
    SET people.birthDate = CONCAT(birthYear,'-',birthMonth,'-',birthDay);
UPDATE people
    SET people.deathDate = CONCAT(deathYear,'-',deathMonth,'-',deathDay);
```

Tests

[19]:
```
%sql SELECT deathDate FROM people WHERE deathDate >= '2005-01-01' ORDER BY␣
↪deathDate ASC LIMIT 10;
```

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

[19]:
```
[(datetime.datetime(2005, 1, 4, 0, 0),),
 (datetime.datetime(2005, 1, 7, 0, 0),),
 (datetime.datetime(2005, 1, 9, 0, 0),),
 (datetime.datetime(2005, 1, 10, 0, 0),),
```

```
 (datetime.datetime(2005, 1, 21, 0, 0),),
 (datetime.datetime(2005, 1, 22, 0, 0),),
 (datetime.datetime(2005, 1, 31, 0, 0),),
 (datetime.datetime(2005, 2, 4, 0, 0),),
 (datetime.datetime(2005, 2, 8, 0, 0),),
 (datetime.datetime(2005, 2, 11, 0, 0),)]
```

[20]:
```
%sql SELECT birthDate FROM people WHERE birthDate <= '1965-01-01' ORDER BY␣
↪birthDate ASC LIMIT 10;
```

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

[20]:
```
[(datetime.datetime(1820, 4, 17, 0, 0),),
 (datetime.datetime(1824, 10, 5, 0, 0),),
 (datetime.datetime(1832, 9, 17, 0, 0),),
 (datetime.datetime(1832, 10, 23, 0, 0),),
 (datetime.datetime(1835, 1, 10, 0, 0),),
 (datetime.datetime(1836, 2, 29, 0, 0),),
 (datetime.datetime(1837, 12, 26, 0, 0),),
 (datetime.datetime(1838, 3, 10, 0, 0),),
 (datetime.datetime(1838, 7, 16, 0, 0),),
 (datetime.datetime(1838, 8, 27, 0, 0),)]
```

4.2.6 Final CREATE Statement

To find the create statement:

- Right click on the table name in workbench
- Select 'Copy to Clipboard'
- Select 'Create Statement'

The create statement will now be copied into your clipboard and can be pasted into the cell below.

```
create table if not exists lahmansdb_to_clean.people
(
    playerID     varchar(15)          not null
        primary key,
    birthYear    smallint             null,
    birthMonth   tinyint              null,
    birthDay     tinyint              null,
    birthCountry varchar(100)         null,
    birthState   varchar(100)         null,
    birthCity    varchar(100)         null,
    deathYear    smallint             null,
    deathMonth   tinyint              null,
    deathDay     tinyint              null,
    deathCountry varchar(100)         null,
    deathState   varchar(100)         null,
    deathCity    varchar(100)         null,
```

```
    nameFirst    varchar(100)          null,
    nameLast     varchar(100)          null,
    nameGiven    varchar(100)          null,
    weight       smallint              null,
    height       smallint              null,
    bats         enum ('L', 'R', 'B') null,
    throws       enum ('L', 'R', 'S') null,
    debut        datetime              null,
    finalGame    datetime              null,
    retroID      varchar(50)           null,
    bbrefID      varchar(50)           null,
    isDead       enum ('Y', 'N')      null,
    birthDate    datetime              null,
    deathDate    datetime              null
);
```

4.3 Batting Table

4.3.1 1) Convert all empty strings to NULL

Explanation Empty strings in a database can mean many things but NULL is a reference variable and that shows that the value is missing or has not been entered in the database. However, when you find an empty string for some of the features/columns in the database, it might not tell you exactly what that empty value means. To remove the disambiguation, we change the empty strings to NULL value. Furthermore, it's easier to check for NULL values and work with them accordingly than some random text in there, be it be an empty string.

Change I have replaced all the empty strings across all the columns in all the rows of the database with a NULL reference variable. So, for any of the columns in the database, there isn't any empty string but in it's place, there is a NULL value. For example, "RBI", "SB" and "CS" before would have some empty strings, which now are replaced by NULL values in the table 'batting'.

SQL

```
UPDATE batting
SET
    playerID = CASE playerID WHEN '' THEN NULL ELSE playerID END,
    yearID = CASE yearID WHEN '' THEN NULL ELSE yearID END,
    stint = CASE stint WHEN '' THEN NULL ELSE stint END,
    teamID = CASE teamID WHEN '' THEN NULL ELSE teamID END,
    lgID = CASE lgID WHEN '' THEN NULL ELSE lgID END,
    G = CASE G WHEN '' THEN NULL ELSE G END,
    AB = CASE AB WHEN '' THEN NULL ELSE ab END,
    R = CASE R WHEN '' THEN NULL ELSE R END,
    H = CASE H WHEN '' THEN NULL ELSE H END,
    2B = CASE 2B WHEN '' THEN NULL ELSE 2B END,
    3B = CASE 3B WHEN '' THEN NULL ELSE 3B END,
    HR = CASE HR WHEN '' THEN NULL ELSE HR END,
```

```
    RBI = CASE RBI WHEN '' THEN NULL ELSE RBI END,
    SB = CASE SB WHEN '' THEN NULL ELSE SB END,
    CS = CASE CS WHEN '' THEN NULL ELSE CS END,
    BB = CASE BB WHEN '' THEN NULL ELSE BB END,
    SO = CASE SO WHEN '' THEN NULL ELSE SO END,
    IBB = CASE IBB WHEN '' THEN NULL ELSE IBB END,
    HBP = CASE HBP WHEN '' THEN NULL ELSE HBP END,
    SH = CASE SH WHEN '' THEN NULL ELSE SH END,
    SF = CASE SF WHEN '' THEN NULL ELSE SF END,
    GIDP = CASE GIDP WHEN '' THEN NULL ELSE GIDP END;
```

Tests

[29]:
```
%sql SELECT count(*) FROM batting where RBI is NULL;
```

```
 * mysql+pymysql://root:***@localhost/lahmansdb_to_clean
1 rows affected.
```

[29]:
```
[(756,)]
```

4.3.2 2) Change column datatypes to appropriate values (ENUM, INT, VARCHAR, DATETIME, ETC)

Explanation The datatypes are really important part of the database schema and tells us the domain of the values that can be entered in that column in the database. This also prevents users from entering an invalid data in the field with a datatype that doesn't match the data entered by the user. For example, for "yearID" field, some user might enter some decimal values (by mistake!) or might try to enter the value in words, but that query wouldn't go through if we have defined those fields as "SMALLINT". Even so, the user wouldn't be able to add some nonsensical value such as 10000 as the yearID values are generally quite less and the datatypes help define us the bounds or the domain of those fields. This is really important to maintain a clean database and consistent database. This consistency can also help when you need to perform some kind of operations on the field, such as for an int field, when the sorting is done, you would get the sorted integer values in numerically ascending order but for a varchar column, it's sorted according to the ascii values of the characters in the string.

Change I have modified the datatypes for the columns in the table 'batting'. - I have changed the datatype for 'playerID' column to a variable string that can have at most 15 characters so, that the values can have variable length and are strings that can have numbers too. This also enables us to have consistent values in the table for playerID as nonsensical values shouldn't be entered in the table for the column and sets the bounds for the playerID field. - I have changed the datatype for 'stint' to 'TINYINT' as the values in this fields are generally very small integers less than 255. Actually, the values in this column in this table is very less even compared to 255 but, it's good to leave some room for the data that would be added to the table in the future. This also puts a domain for the values into place thus, making the data consistent in the table for this column. - I have changed the datatype for 'teamID' and 'lgID' columns to variable strings (VARHCAR) having at most the length of 20 characters, thus setting the domain for the values being entered into these fields. - I have changed the datatype for all the other columns to 'SMALLINT' thus, specifying the domain of the values to be less than or equal to 65535. Some of these fields could

also have been set to the datatype 'TINYINT' but I have left some room for the fields for the data that would be entered in the future taking into prespective the scalability and consistency of the data in these fields.

SQL

```
ALTER TABLE batting
    MODIFY COLUMN playerID VARCHAR(15),
    MODIFY COLUMN yearID SMALLINT,
    MODIFY COLUMN stint TINYINT,
    MODIFY COLUMN teamID VARCHAR(20),
    MODIFY COLUMN lgID VARCHAR(20),
    MODIFY COLUMN G SMALLINT,
    MODIFY COLUMN AB SMALLINT,
    MODIFY COLUMN R SMALLINT,
    MODIFY COLUMN H SMALLINT,
    MODIFY COLUMN 2B SMALLINT,
    MODIFY COLUMN 3B SMALLINT,
    MODIFY COLUMN HR SMALLINT,
    MODIFY COLUMN RBI SMALLINT,
    MODIFY COLUMN SB SMALLINT,
    MODIFY COLUMN CS SMALLINT,
    MODIFY COLUMN BB SMALLINT,
    MODIFY COLUMN SO SMALLINT,
    MODIFY COLUMN IBB SMALLINT,
    MODIFY COLUMN HBP SMALLINT,
    MODIFY COLUMN SH SMALLINT,
    MODIFY COLUMN SF SMALLINT,
    MODIFY COLUMN GIDP SMALLINT;
```

4.3.3 3) Add a Primary Key

Two options for the Primary Key:

- Composite Key: playerID, yearID, stint
- Covering Key (Index): playerID, yearID, stint, teamID

Choice I have chosen the first option above by taking the 'Composite Key' to be the primary key that is composed of 3 fields, 'playerID', 'yearID' and 'stint. Furthermore, I have specified these fields to be non-nullable which is a prerequisite for the primary key and it's components and have also specified the datatypes to be variable string having upto 15 characters, small integer having values less than or equal to 65535, and tiny integer having values less than or equal to 255 for the fields 'yearID', and 'stint' respectively.

Explanation We need some primary key for the table 'batting' so, that we can uniquely identify each row/record in the table by that key. As primary key is also an index which is beneficial for faster searching of a particular record in the table, it's really significant to be added to the table.

SQL

```
ALTER TABLE batting
CHANGE COLUMN playerID playerID VARCHAR(15) NOT NULL,
CHANGE COLUMN yearID yearID SMALLINT NOT NULL,
CHANGE COLUMN stint stint TINYINT NOT NULL,
ADD PRIMARY KEY (playerID, yearID, stint);
```

Test

[24]:
```
%sql SHOW KEYS FROM batting WHERE Key_name = 'PRIMARY' and Column_name =␣
↪'playerID'
```

```
 * mysql+pymysql://root:***@localhost/lahmansdb_to_clean
1 rows affected.
```

[24]:
```
[('batting', 0, 'PRIMARY', 1, 'playerID', 'A', 20175, None, None, '', 'BTREE',
'', '', 'YES', None)]
```

4.3.4 4) Add a foreign key on playerID between the People and Batting Tables

Note: Two people in the batting table do not exist in the people table. How should you handle this issue?

Explanation Adding a foreign key can serve various purposes and is really important for indexing in the table 'batting' and connecting it to the table 'people'. It's very important to prevent actions that would destroy links between the tables. It also enforces control over the data that can be stored in the foreign key table.

Change I have added a foreign key in the table 'batting' as 'playerID' that is linked (references) to the primary key in the other table 'people' thus, establishing logical relationship between the two tables.

SQL

```
ALTER TABLE batting
ADD FOREIGN KEY (playerID) REFERENCES people(playerID);
```

Tests

[25]:
```
%sql Select playerID from batting where playerID not in (select playerID from␣
↪people);
```

```
 * mysql+pymysql://root:***@localhost/lahmansdb_to_clean
0 rows affected.
```

[25]:
```
[]
```

4.3.5 Final CREATE Statement

To find the create statement:

- Right click on the table name in workbench

- Select 'Copy to Clipboard'
- Select 'Create Statement'

The create statement will now be copied into your clipboard and can be pasted into the cell below.

```
create table if not exists lahmansdb_to_clean.batting
(
    playerID varchar(15) not null,
    yearID   smallint    not null,
    stint    tinyint     not null,
    teamID   varchar(20) null,
    lgID     varchar(20) null,
    G        smallint    null,
    AB       smallint    null,
    R        smallint    null,
    H        smallint    null,
    `2B`     smallint    null,
    `3B`     smallint    null,
    HR       smallint    null,
    RBI      smallint    null,
    SB       smallint    null,
    CS       smallint    null,
    BB       smallint    null,
    SO       smallint    null,
    IBB      smallint    null,
    HBP      smallint    null,
    SH       smallint    null,
    SF       smallint    null,
    GIDP     smallint    null,
    primary key (playerID, yearID, stint),
    constraint batting_ibfk_1
        foreign key (playerID) references lahmansdb_to_clean.people (playerID)
);
```

5 Part D: SQL Queries

NOTE: You must use the CLEAN lahman schema provided in HW0 for the queries below to ensure your answers are consistent with the solutions.

[3]:
```
%reload_ext sql
%sql mysql+pymysql://root:dbuserdbuser@localhost/lahmansbaseballdb
```

5.1 Question 0

What is the average salary in baseball history?

```
SELECT AVG(salary) from salaries
```

Average Salary = 2085634.053125473

5.2 Question 1

Select the players with a first name of Sam who were born in the United States and attended college.

Include their first name, last name, playerID, school ID, yearID and birth state. Limit 10

Hint: Use a Join between People and CollegePlaying

```
SELECT nameFirst, nameLast, people.playerID, schoolID, yearID, birthState
FROM people JOIN collegeplaying c on people.playerID = c.playerID
WHERE nameFirst = 'Sam' and birthCountry = 'USA' and c.schoolID is not NULL
LIMIT 10;
```

[12]:
```
%%sql SELECT nameFirst, nameLast, people.playerID, schoolID, yearID, birthState
FROM people JOIN collegeplaying c on people.playerID = c.playerID
WHERE nameFirst = 'Sam' and birthCountry = 'USA' and c.schoolID is not NULL
LIMIT 10;
```

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

[12]:
```
[('Sam', 'Barnes', 'barnesa01', 'auburn', 1918, 'AL'),
 ('Sam', 'Barnes', 'barnesa01', 'auburn', 1919, 'AL'),
 ('Sam', 'Barnes', 'barnesa01', 'auburn', 1920, 'AL'),
 ('Sam', 'Barnes', 'barnesa01', 'auburn', 1921, 'AL'),
 ('Sam', 'Bowens', 'bowensa01', 'tennst', 1956, 'NC'),
 ('Sam', 'Bowens', 'bowensa01', 'tennst', 1957, 'NC'),
 ('Sam', 'Bowens', 'bowensa01', 'tennst', 1958, 'NC'),
 ('Sam', 'Bowen', 'bowensa02', 'gacoast', 1971, 'GA'),
 ('Sam', 'Bowen', 'bowensa02', 'gacoast', 1972, 'GA'),
 ('Sam', 'Brown', 'brownsa01', 'grovecity', 1899, 'PA')]
```

5.3 Question 2

Update all entries with full_name Columbia University to 'Columbia University in the City of New York' in the Schools table. Then select the row.

```
UPDATE schools
SET name_full = 'Columbia University in the City of New York'
WHERE name_full = 'Columbia University'
```

```
SELECT * from schools
WHERE name_full = 'Columbia University in the City of New York';
```

[43]:
```
%%sql SELECT * from schools
WHERE name_full = 'Columbia University in the City of New York';
```

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

[43]:
```
[('columbia', 'Columbia University in the City of New York', 'New York', 'NY',
'USA')]
```

6 Part E: CSVDataTable

6.1 i. Conceptual Questions

The purpose of this homework is to teach you the behaviour of SQL Databases by asking you to implement functions that will model the behaviour of a real database with CSVDataTable. You will mimic a SQL Database using CSV files.

Read through the scaffolding code provided in the CSVDataTable folder first to understand and answer the following conceptual questions.

1. Given this SQL statement:

   ```
   SELECT nameFirst, nameLast FROM people WHERE playerID = collied01
   ```

   If you run find_by_primary_key() on this statement, what are key_fields and field_list?

   key_fields will include just one field 'playerID' as collied01 and field_list will include ['nameFirst', 'nameLast'] fields from the table 'people'.

2. What should be checked when you are trying to INSERT a new row into a table with a PK?

   As the Primary Key (PK) uniquely identifies a row in the table, we should first check whether the new row that we are trying to enter has a primary key not already present in the table and also that the new row contains a primary key which is not NULL as the primary key column is non-nullable. If any of these cases turn out to be true, such that if the primary key in the new row is already present or is NULL, we don't try to add it to the table.

3. What should be checked when you are trying to UPDATE a row in a table with a PK?

   When updating a row in a table which has a primary key, we should check whether the primary key or any of it's components (for a composite key) is/are not being changed to something. We need to raise an exception and give an error for this case. If we need to update the primary key, then, generally, a new record is added making that update and copying over the data and then deleting the row that you don't want. This also helps keep the foreign key relationships intact.

6.2 ii. Coding

You are responsible for implementing and testing two classes in Python: CSVDataTable, BaseDataTable. The python files and data can be found in the assignment under Courseworks.

We have already given you **find_by_template(self, template, field_list=None, limit=None, offset=None, order_by=None)** Use this as a jumping off point for the rest of your functions.

Methods to complete:

CSVDataTable.py - find_by_primary_key(self, key_fields, field_list=None) - delete_by_key(self, key_fields) - delete_by_template(self, template) - update_by_key(self, key_fields, new_values) - update_by_template(self, template, new_values) - insert(self, new_record) CSV_table_tests.py

- You must test all methods. You will have to write these tests yourself. - You must test your methods on the People and Batting table.

If you do not include tests and tests outputs 50% of this section's points will be deducted at the start

6.3 iii. Testing

Please copy the text from the output of your tests and paste it below:

find_by_primary_key(): Known Record {'playerID': 'aardsda01', 'birthYear': '1981', 'birthMonth': '12', 'birthDay': '27', 'birthCountry': 'USA', 'birthState': 'CO', 'birthCity': 'Denver', 'deathYear': '', 'deathMonth': '', 'deathDay': '', 'deathCountry': '', 'deathState': '', 'deathCity': '', 'nameFirst': 'David', 'nameLast': 'Aardsma', 'nameGiven': 'David Allan', 'weight': '215', 'height': '75', 'bats': 'R', 'throws': 'R', 'debut': '2004-04-06', 'finalGame': '2015-08-23', 'retroID': 'aardd001', 'bbrefID': 'aardsda01'}

find_by_primary_key(): Unknown Record None

find_by_primary_key(): Known Record with Field List Projected {'playerID': 'aardsda01', 'nameFirst': 'David', 'nameLast': 'Aardsma', 'nameGiven': 'David Allan', 'debut': '2004-04-06', 'finalGame': '2015-08-23'}

find_by_primary_key(): Unknown Record with Field List Projected None

find_by_template(): Known Template [{'playerID': 'aardsda01', 'birthYear': '1981', 'birthMonth': '12', 'birthDay': '27', 'birthCountry': 'USA', 'birthState': 'CO', 'birthCity': 'Denver', 'deathYear': '', 'deathMonth': '', 'deathDay': '', 'deathCountry': '', 'deathState': '', 'deathCity': '', 'nameFirst': 'David', 'nameLast': 'Aardsma', 'nameGiven': 'David Allan', 'weight': '215', 'height': '75', 'bats': 'R', 'throws': 'R', 'debut': '2004-04-06', 'finalGame': '2015-08-23', 'retroID': 'aardd001', 'bbrefID': 'aardsda01'}]

find_by_template(): Known Template - Multiple Records [{'playerID': 'woodyma01', 'birthYear': '1978', 'birthMonth': '12', 'birthDay': '19', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '', 'deathMonth': '', 'deathDay': '', 'deathCountry': '', 'deathState': '', 'deathCity': '', 'nameFirst': 'Mark', 'nameLast': 'Woodyard', 'nameGiven': 'Mark Anthony', 'weight': '195', 'height': '74', 'bats': 'R', 'throws': 'R', 'debut': '2005-09-17', 'finalGame': '2005-10-01', 'retroID': 'woodm003', 'bbrefID': 'woodyma01'}, {'playerID': 'veltmpa01', 'birthYear': '1906', 'birthMonth': '3', 'birthDay': '24', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '1980', 'deathMonth': '10', 'deathDay': '1', 'deathCountry': 'USA', 'deathState': 'TX', 'deathCity': 'San Antonio', 'nameFirst': 'Pat', 'nameLast': 'Veltman', 'nameGiven': 'Arthur Patrick', 'weight': '175', 'height': '72', 'bats': 'R', 'throws': 'R', 'debut': '1926-04-17', 'finalGame': '1934-09-30', 'retroID': 'veltp101', 'bbrefID': 'veltmpa01'}, {'playerID': 'townsle01', 'birthYear': '1891', 'birthMonth': '1', 'birthDay': '15', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '1976', 'deathMonth': '12', 'deathDay': '3', 'deathCountry': 'USA', 'deathState': 'AL', 'deathCity': 'Mobile', 'nameFirst': 'Leo', 'nameLast': 'Townsend', 'nameGiven': 'Leo Alphonse', 'weight': '160', 'height': '70', 'bats': 'L', 'throws': 'L', 'debut': '1920-09-08', 'finalGame': '1921-05-27', 'retroID': 'townl101', 'bbrefID': 'townsle01'}, {'playerID': 'toeneha01', 'birthYear': '1917', 'birthMonth': '10', 'birthDay': '8', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '2004', 'deathMonth': '6', 'deathDay': '28', 'deathCountry': 'USA', 'deathState': 'FL', 'deathCity': 'Tampa', 'nameFirst': 'Hal', 'nameLast': 'Toenes', 'nameGiven':

‘William Harrel’, ‘weight’: ‘175’, ‘height’: ‘71’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1947-09-17’, ‘finalGame’: ‘1947-09-27’, ‘retroID’: ‘toenh101’, ‘bbrefID’: ‘toeneha01’}, {‘playerID’: ‘taylowa01’, ‘birthYear’: ‘1965’, ‘birthMonth’: ‘10’, ‘birthDay’: ‘19’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Wade’, ‘nameLast’: ‘Taylor’, ‘nameGiven’: ‘Wade Eric’, ‘weight’: ‘185’, ‘height’: ‘73’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1991-06-02’, ‘finalGame’: ‘1991-09-26’, ‘retroID’: ‘taylw001’, ‘bbrefID’: ‘taylowa01’}, {‘playerID’: ‘sparkst02’, ‘birthYear’: ‘1975’, ‘birthMonth’: ‘3’, ‘birthDay’: ‘28’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Steve’, ‘nameLast’: ‘Sparks’, ‘nameGiven’: ‘Stephen Lanier’, ‘weight’: ‘210’, ‘height’: ‘76’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘2000-07-19’, ‘finalGame’: ‘2000-08-02’, ‘retroID’: ‘spars002’, ‘bbrefID’: ‘sparkst02’}, {‘playerID’: ‘smithoz01’, ‘birthYear’: ‘1954’, ‘birthMonth’: ‘12’, ‘birthDay’: ‘26’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Ozzie’, ‘nameLast’: ‘Smith’, ‘nameGiven’: ‘Osborne Earl’, ‘weight’: ‘150’, ‘height’: ‘71’, ‘bats’: ‘B’, ‘throws’: ‘R’, ‘debut’: ‘1978-04-07’, ‘finalGame’: ‘1996-09-29’, ‘retroID’: ‘smito001’, ‘bbrefID’: ‘smithoz01’}, {‘playerID’: ‘sextoji01’, ‘birthYear’: ‘1951’, ‘birthMonth’: ‘12’, ‘birthDay’: ‘15’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Jimmy’, ‘nameLast’: ‘Sexton’, ‘nameGiven’: ‘Jimmy Dale’, ‘weight’: ‘175’, ‘height’: ‘70’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1977-09-02’, ‘finalGame’: ‘1983-10-02’, ‘retroID’: ‘sextj101’, ‘bbrefID’: ‘sextoji01’}, {‘playerID’: ‘rollire01’, ‘birthYear’: ‘1904’, ‘birthMonth’: ‘3’, ‘birthDay’: ‘31’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ‘1964’, ‘deathMonth’: ‘12’, ‘deathDay’: ‘31’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘AL’, ‘deathCity’: ‘Mobile’, ‘nameFirst’: ‘Red’, ‘nameLast’: ‘Rollings’, ‘nameGiven’: ‘William Russell’, ‘weight’: ‘167’, ‘height’: ‘71’, ‘bats’: ‘L’, ‘throws’: ‘R’, ‘debut’: ‘1927-04-17’, ‘finalGame’: ‘1930-09-11’, ‘retroID’: ‘rollr103’, ‘bbrefID’: ‘rollire01’}, {‘playerID’: ‘randasa01’, ‘birthYear’: ‘1960’, ‘birthMonth’: ‘8’, ‘birthDay’: ‘19’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Sap’, ‘nameLast’: ‘Randall’, ‘nameGiven’: ‘James Odell’, ‘weight’: ‘195’, ‘height’: ‘71’, ‘bats’: ‘B’, ‘throws’: ‘R’, ‘debut’: ‘1988-08-02’, ‘finalGame’: ‘1988-08-06’, ‘retroID’: ‘randj001’, ‘bbrefID’: ‘randasa01’}, {‘playerID’: ‘pierrju01’, ‘birthYear’: ‘1977’, ‘birthMonth’: ‘8’, ‘birthDay’: ‘14’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Juan’, ‘nameLast’: ‘Pierre’, ‘nameGiven’: “Juan D’Vaughn”, ‘weight’: ‘180’, ‘height’: ‘70’, ‘bats’: ‘L’, ‘throws’: ‘L’, ‘debut’: ‘2000-08-07’, ‘finalGame’: ‘2013-09-29’, ‘retroID’: ‘pierj002’, ‘bbrefID’: ‘pierrju01’}, {‘playerID’: ‘peavyja01’, ‘birthYear’: ‘1981’, ‘birthMonth’: ‘5’, ‘birthDay’: ‘31’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Jake’, ‘nameLast’: ‘Peavy’, ‘nameGiven’: ‘Jacob Edward’, ‘weight’: ‘195’, ‘height’: ‘73’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘2002-06-22’, ‘finalGame’: ‘2016-09-21’, ‘retroID’: ‘peavj001’, ‘bbrefID’: ‘peavyja01’}, {‘playerID’: ‘pattejo03’, ‘birthYear’: ‘1992’, ‘birthMonth’: ‘2’, ‘birthDay’: ‘12’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Jordan’, ‘nameLast’: ‘Patterson’, ‘nameGiven’: ‘Jordan Andrew’, ‘weight’: ‘215’, ‘height’: ‘76’, ‘bats’: ‘L’, ‘throws’: ‘L’, ‘debut’: ‘2016-09-08’, ‘finalGame’: ‘2016-10-02’, ‘retroID’: ‘pattj005’, ‘bbrefID’: ‘pattejo03’}, {‘playerID’: ‘paigesa01’, ‘birthYear’: ‘1906’, ‘birthMonth’: ‘7’, ‘birthDay’: ‘7’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ‘1982’, ‘deathMonth’: ‘6’, ‘deathDay’: ‘8’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘MO’, ‘deathCity’: ‘Kansas City’, ‘name-

First': 'Satchel', 'nameLast': 'Paige', 'nameGiven': 'Leroy Robert', 'weight': '180', 'height': '75', 'bats': 'R', 'throws': 'R', 'debut': '1948-07-09', 'finalGame': '1965-09-25', 'retroID': 'paigs101', 'bbrefID': 'paigesa01'}, {'playerID': 'otisam01', 'birthYear': '1947', 'birthMonth': '4', 'birthDay': '26', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '', 'deathMonth': '', 'deathDay': '', 'deathCountry': '', 'deathState': '', 'deathCity': '', 'nameFirst': 'Amos', 'nameLast': 'Otis', 'nameGiven': 'Amos Joseph', 'weight': '165', 'height': '71', 'bats': 'R', 'throws': 'R', 'debut': '1967-09-06', 'finalGame': '1984-08-05', 'retroID': 'otisa001', 'bbrefID': 'otisam01'}, {'playerID': 'milnepe01', 'birthYear': '1925', 'birthMonth': '4', 'birthDay': '10', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '1999', 'deathMonth': '4', 'deathDay': '11', 'deathCountry': 'USA', 'deathState': 'AL', 'deathCity': 'Mobile', 'nameFirst': 'Pete', 'nameLast': 'Milne', 'nameGiven': 'William James', 'weight': '180', 'height': '73', 'bats': 'L', 'throws': 'R', 'debut': '1948-09-15', 'finalGame': '1950-05-14', 'retroID': 'milnp101', 'bbrefID': 'milnepe01'}, {'playerID': 'merchan01', 'birthYear': '1950', 'birthMonth': '8', 'birthDay': '30', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '', 'deathMonth': '', 'deathDay': '', 'deathCountry': '', 'deathState': '', 'deathCity': '', 'nameFirst': 'Andy', 'nameLast': 'Merchant', 'nameGiven': 'James Anderson', 'weight': '185', 'height': '71', 'bats': 'L', 'throws': 'R', 'debut': '1975-09-28', 'finalGame': '1976-06-10', 'retroID': 'merca101', 'bbrefID': 'merchan01'}, {'playerID': 'mcgilra01', 'birthYear': '1953', 'birthMonth': '10', 'birthDay': '29', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '', 'deathMonth': '', 'deathDay': '', 'deathCountry': '', 'deathState': '', 'deathCity': '', 'nameFirst': 'Randy', 'nameLast': 'McGilberry', 'nameGiven': 'Randall Kent', 'weight': '195', 'height': '73', 'bats': 'B', 'throws': 'R', 'debut': '1977-09-06', 'finalGame': '1978-10-01', 'retroID': 'mcgir101', 'bbrefID': 'mcgilra01'}, {'playerID': 'mccovwi01', 'birthYear': '1938', 'birthMonth': '1', 'birthDay': '10', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '2018', 'deathMonth': '10', 'deathDay': '31', 'deathCountry': 'USA', 'deathState': 'CA', 'deathCity': 'Palo Alto', 'nameFirst': 'Willie', 'nameLast': 'McCovey', 'nameGiven': 'Willie Lee', 'weight': '198', 'height': '76', 'bats': 'L', 'throws': 'L', 'debut': '1959-07-30', 'finalGame': '1980-07-06', 'retroID': 'mccow101', 'bbrefID': 'mccovwi01'}, {'playerID': 'masonji01', 'birthYear': '1950', 'birthMonth': '8', 'birthDay': '14', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '', 'deathMonth': '', 'deathDay': '', 'deathCountry': '', 'deathState': '', 'deathCity': '', 'nameFirst': 'Jim', 'nameLast': 'Mason', 'nameGiven': 'James Percy', 'weight': '185', 'height': '74', 'bats': 'L', 'throws': 'R', 'debut': '1971-09-26', 'finalGame': '1979-09-24', 'retroID': 'masoj101', 'bbrefID': 'masonji01'}, {'playerID': 'kellyhe01', 'birthYear': '1892', 'birthMonth': '6', 'birthDay': '4', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '1973', 'deathMonth': '5', 'deathDay': '18', 'deathCountry': 'USA', 'deathState': 'CA', 'deathCity': 'Torrance', 'nameFirst': 'Herb', 'nameLast': 'Kelly', 'nameGiven': 'Herbert Barrett', 'weight': '160', 'height': '69', 'bats': 'L', 'throws': 'L', 'debut': '1914-09-21', 'finalGame': '1915-09-06', 'retroID': 'kellh102', 'bbrefID': 'kellyhe01'}, {'playerID': 'jumonge01', 'birthYear': '1917', 'birthMonth': '5', 'birthDay': '16', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '1996', 'deathMonth': '12', 'deathDay': '12', 'deathCountry': 'USA', 'deathState': 'AL', 'deathCity': 'Mobile', 'nameFirst': 'George', 'nameLast': 'Jumonville', 'nameGiven': 'George Benedict', 'weight': '175', 'height': '72', 'bats': 'R', 'throws': 'R', 'debut': '1940-09-13', 'finalGame': '1941-05-20', 'retroID': 'jumog101', 'bbrefID': 'jumonge01'}, {'playerID': 'howelpa01', 'birthYear': '1968', 'birthMonth': '8', 'birthDay': '31', 'birthCountry': 'USA', 'birthState': 'AL', 'birthCity': 'Mobile', 'deathYear': '', 'deathMonth': '', 'deathDay': '', 'deathCountry': '', 'deathState': '', 'deathCity': '', 'nameFirst': 'Pat', 'nameLast': 'Howell', 'nameGiven': "Patrick O'Neal", 'weight': '155', 'height': '71', 'bats': 'B', 'throws': 'R', 'debut': '1992-07-10', 'finalGame': '1992-10-04', 'retroID': 'howep001', 'bbrefID': 'howelpa01'}, {'playerID': 'hoodde01', 'birthYear': '1990', 'birthMonth': '4', 'birthDay': '3', 'birthCountry': 'USA', 'birthState': 'AL',

‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Destin’, ‘nameLast’: ‘Hood’, ‘nameGiven’: ‘Destin Dwane’, ‘weight’: ‘205’, ‘height’: ‘74’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘2016-09-02’, ‘finalGame’: ‘2016-10-02’, ‘retroID’: ‘hoodd001’, ‘bbrefID’: ‘hoodde01’}, {‘playerID’: ‘henlebo01’, ‘birthYear’: ‘1973’, ‘birthMonth’: ‘1’, ‘birthDay’: ‘30’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Bob’, ‘nameLast’: ‘Henley’, ‘nameGiven’: ‘Robert Clifton’, ‘weight’: ‘190’, ‘height’: ‘74’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1998-07-19’, ‘finalGame’: ‘1998-09-26’, ‘retroID’: ‘henlb001’, ‘bbrefID’: ‘henlebo01’}, {‘playerID’: ‘hazewdr01’, ‘birthYear’: ‘1959’, ‘birthMonth’: ‘9’, ‘birthDay’: ‘2’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ‘2013’, ‘deathMonth’: ‘7’, ‘deathDay’: ‘28’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘CA’, ‘deathCity’: ‘Sacramento’, ‘nameFirst’: ‘Drungo’, ‘nameLast’: ‘Hazewood’, ‘nameGiven’: ‘Drungo LaRue’, ‘weight’: ‘210’, ‘height’: ‘75’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1980-09-19’, ‘finalGame’: ‘1980-10-04’, ‘retroID’: ‘hazed101’, ‘bbrefID’: ‘hazewdr01’}, {‘playerID’: ‘fritzch01’, ‘birthYear’: ‘1882’, ‘birthMonth’: ‘6’, ‘birthDay’: ‘13’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ‘1943’, ‘deathMonth’: ‘7’, ‘deathDay’: ‘30’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘AL’, ‘deathCity’: ‘Mobile’, ‘nameFirst’: ‘Charlie’, ‘nameLast’: ‘Fritz’, ‘nameGiven’: ‘Charles Cornelius’, ‘weight’: ’‘, ’height’: ’‘, ’bats’: ’‘, ’throws’: ‘L’, ‘debut’: ‘1907-10-05’, ‘finalGame’: ‘1907-10-05’, ‘retroID’: ‘fritc101’, ‘bbrefID’: ‘fritzch01’}, {‘playerID’: ‘frankch01’, ‘birthYear’: ‘1870’, ‘birthMonth’: ‘5’, ‘birthDay’: ‘30’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ‘1922’, ‘deathMonth’: ‘5’, ‘deathDay’: ‘24’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘TN’, ‘deathCity’: ‘Memphis’, ‘nameFirst’: ‘Charlie’, ‘nameLast’: ‘Frank’, ‘nameGiven’: ‘Charles’, ‘weight’: ‘170’, ‘height’: ‘70’, ‘bats’: ‘L’, ‘throws’: ‘L’, ‘debut’: ‘1893-08-18’, ‘finalGame’: ‘1894-07-25’, ‘retroID’: ‘franc101’, ‘bbrefID’: ‘frankch01’}, {‘playerID’: ‘ervinph01’, ‘birthYear’: ‘1992’, ‘birthMonth’: ‘7’, ‘birthDay’: ‘15’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Phil’, ‘nameLast’: ‘Ervin’, ‘nameGiven’: ‘Phillip S.’, ‘weight’: ‘207’, ‘height’: ‘70’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘2017-04-22’, ‘finalGame’: ‘2018-09-30’, ‘retroID’: ‘ervip001’, ‘bbrefID’: ‘ervinph01’}, {‘playerID’: ‘duncaco01’, ‘birthYear’: ‘1974’, ‘birthMonth’: ‘10’, ‘birthDay’: ‘9’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Courtney’, ‘nameLast’: ‘Duncan’, ‘nameGiven’: ‘Courtney Demond’, ‘weight’: ‘185’, ‘height’: ‘72’, ‘bats’: ‘L’, ‘throws’: ‘R’, ‘debut’: ‘2001-04-02’, ‘finalGame’: ‘2002-06-01’, ‘retroID’: ‘duncc001’, ‘bbrefID’: ‘duncaco01’}, {‘playerID’: ‘duffech01’, ‘birthYear’: ‘1866’, ‘birthMonth’: ‘1’, ‘birthDay’: ‘27’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ‘1894’, ‘deathMonth’: ‘12’, ‘deathDay’: ‘24’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘AL’, ‘deathCity’: ‘Mobile’, ‘nameFirst’: ‘Charlie’, ‘nameLast’: ‘Duffee’, ‘nameGiven’: ‘Charles Edward’, ‘weight’: ‘151’, ‘height’: ‘65’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1889-04-17’, ‘finalGame’: ‘1893-04-30’, ‘retroID’: ‘duffc101’, ‘bbrefID’: ‘duffech01’}, {‘playerID’: ‘davisto03’, ‘birthYear’: ‘1973’, ‘birthMonth’: ‘5’, ‘birthDay’: ‘21’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Tommy’, ‘nameLast’: ‘Davis’, ‘nameGiven’: ‘Thomas James’, ‘weight’: ‘195’, ‘height’: ‘73’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1999-05-14’, ‘finalGame’: ‘1999-05-30’, ‘retroID’: ‘davit003’, ‘bbrefID’: ‘davisto03’}, {‘playerID’: ‘bradfbu01’, ‘birthYear’: ‘1944’, ‘birthMonth’: ‘7’, ‘birthDay’: ‘25’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Buddy’, ‘nameLast’: ‘Bradford’, ‘nameGiven’: ‘Charles William’, ‘weight’: ‘170’, ‘height’: ‘71’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1966-09-09’, ‘finalGame’: ‘1976-07-24’, ‘retroID’: ‘bradb105’, ‘bbrefID’: ‘bradfbu01’}, {‘playerID’:

‘bollija01’, ‘birthYear’: ‘1917’, ‘birthMonth’: ‘2’, ‘birthDay’: ‘20’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ‘1998’, ‘deathMonth’: ‘4’, ‘deathDay’: ‘13’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘FL’, ‘deathCity’: ‘Panama City’, ‘nameFirst’: ‘Jack’, ‘nameLast’: ‘Bolling’, ‘nameGiven’: ‘John Edward’, ‘weight’: ‘168’, ‘height’: ‘71’, ‘bats’: ‘L’, ‘throws’: ‘L’, ‘debut’: ‘1939-06-10’, ‘finalGame’: ‘1944-09-20’, ‘retroID’: ‘bollj101’, ‘bbrefID’: ‘bollija01’}, {‘playerID’: ‘bollifr01’, ‘birthYear’: ‘1931’, ‘birthMonth’: ‘11’, ‘birthDay’: ‘16’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Frank’, ‘nameLast’: ‘Bolling’, ‘nameGiven’: ‘Frank Elmore’, ‘weight’: ‘175’, ‘height’: ‘73’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1954-04-13’, ‘finalGame’: ‘1966-09-15’, ‘retroID’: ‘bollf101’, ‘bbrefID’: ‘bollifr01’}, {‘playerID’: ‘adamste01’, ‘birthYear’: ‘1973’, ‘birthMonth’: ‘3’, ‘birthDay’: ‘6’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Terry’, ‘nameLast’: ‘Adams’, ‘nameGiven’: ‘Terry Wayne’, ‘weight’: ‘180’, ‘height’: ‘75’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1995-08-10’, ‘finalGame’: ‘2005-05-23’, ‘retroID’: ‘adamt001’, ‘bbrefID’: ‘adamste01’}, {‘playerID’: ‘adairbi99’, ‘birthYear’: ‘1913’, ‘birthMonth’: ‘2’, ‘birthDay’: ‘10’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ‘2002’, ‘deathMonth’: ‘6’, ‘deathDay’: ‘17’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘AL’, ‘deathCity’: ‘Bay Minette’, ‘nameFirst’: ‘Bill’, ‘nameLast’: ‘Adair’, ‘nameGiven’: ‘Marion Danne’, ‘weight’: ‘168’, ‘height’: ‘72’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ’‘, ’finalGame’: ’‘, ’retroID’: ‘adaib801’, ‘bbrefID’: ‘adairbi99’}, {‘playerID’: ‘aaronto01’, ‘birthYear’: ‘1939’, ‘birthMonth’: ‘8’, ‘birthDay’: ‘5’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ‘1984’, ‘deathMonth’: ‘8’, ‘deathDay’: ‘16’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘GA’, ‘deathCity’: ‘Atlanta’, ‘nameFirst’: ‘Tommie’, ‘nameLast’: ‘Aaron’, ‘nameGiven’: ‘Tommie Lee’, ‘weight’: ‘190’, ‘height’: ‘75’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1962-04-10’, ‘finalGame’: ‘1971-09-26’, ‘retroID’: ‘aarot101’, ‘bbrefID’: ‘aaronto01’}, {‘playerID’: ‘aaronha01’, ‘birthYear’: ‘1934’, ‘birthMonth’: ‘2’, ‘birthDay’: ‘5’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘AL’, ‘birthCity’: ‘Mobile’, ‘deathYear’: ’‘, ’deathMonth’: ’‘, ’deathDay’: ’‘, ’deathCountry’: ’‘, ’deathState’: ’‘, ’deathCity’: ’‘, ’nameFirst’: ‘Hank’, ‘nameLast’: ‘Aaron’, ‘nameGiven’: ‘Henry Louis’, ‘weight’: ‘180’, ‘height’: ‘72’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1954-04-13’, ‘finalGame’: ‘1976-10-03’, ‘retroID’: ‘aaroh101’, ‘bbrefID’: ‘aaronha01’}]

find_by_template(): Unknown Record with Empty Data Possibility []

find_by_template(): Known Template - Field List Projected [{‘playerID’: ‘aardsda01’, ‘nameFirst’: ‘David’, ‘nameLast’: ‘Aardsma’, ‘nameGiven’: ‘David Allan’, ‘debut’: ‘2004-04-06’, ‘finalGame’: ‘2015-08-23’}]

find_by_template(): Known Template - Multiple Records - Field List Projected [{‘playerID’: ‘woodyma01’, ‘nameFirst’: ‘Mark’, ‘nameLast’: ‘Woodyard’, ‘nameGiven’: ‘Mark Anthony’, ‘debut’: ‘2005-09-17’, ‘finalGame’: ‘2005-10-01’}, {‘playerID’: ‘veltmpa01’, ‘nameFirst’: ‘Pat’, ‘nameLast’: ‘Veltman’, ‘nameGiven’: ‘Arthur Patrick’, ‘debut’: ‘1926-04-17’, ‘finalGame’: ‘1934-09-30’}, {‘playerID’: ‘townsle01’, ‘nameFirst’: ‘Leo’, ‘nameLast’: ‘Townsend’, ‘nameGiven’: ‘Leo Alphonse’, ‘debut’: ‘1920-09-08’, ‘finalGame’: ‘1921-05-27’}, {‘playerID’: ‘toeneha01’, ‘nameFirst’: ‘Hal’, ‘nameLast’: ‘Toenes’, ‘nameGiven’: ‘William Harrel’, ‘debut’: ‘1947-09-17’, ‘finalGame’: ‘1947-09-27’}, {‘playerID’: ‘taylowa01’, ‘nameFirst’: ‘Wade’, ‘nameLast’: ‘Taylor’, ‘nameGiven’: ‘Wade Eric’, ‘debut’: ‘1991-06-02’, ‘finalGame’: ‘1991-09-26’}, {‘playerID’: ‘sparkst02’, ‘nameFirst’: ‘Steve’, ‘nameLast’: ‘Sparks’, ‘nameGiven’: ‘Stephen Lanier’, ‘debut’: ‘2000-07-19’, ‘finalGame’: ‘2000-08-02’}, {‘playerID’: ‘smithoz01’, ‘nameFirst’: ‘Ozzie’, ‘nameLast’: ‘Smith’, ‘nameGiven’: ‘Osborne Earl’, ‘debut’: ‘1978-04-07’, ‘finalGame’: ‘1996-09-29’}, {‘playerID’: ‘sextoji01’, ‘name-

First': 'Jimmy', 'nameLast': 'Sexton', 'nameGiven': 'Jimmy Dale', 'debut': '1977-09-02', 'finalGame': '1983-10-02'}, {'playerID': 'rollire01', 'nameFirst': 'Red', 'nameLast': 'Rollings', 'nameGiven': 'William Russell', 'debut': '1927-04-17', 'finalGame': '1930-09-11'}, {'playerID': 'randasa01', 'nameFirst': 'Sap', 'nameLast': 'Randall', 'nameGiven': 'James Odell', 'debut': '1988-08-02', 'finalGame': '1988-08-06'}, {'playerID': 'pierrju01', 'nameFirst': 'Juan', 'nameLast': 'Pierre', 'nameGiven': "Juan D'Vaughn", 'debut': '2000-08-07', 'finalGame': '2013-09-29'}, {'playerID': 'peavyja01', 'nameFirst': 'Jake', 'nameLast': 'Peavy', 'nameGiven': 'Jacob Edward', 'debut': '2002-06-22', 'finalGame': '2016-09-21'}, {'playerID': 'pattejo03', 'nameFirst': 'Jordan', 'nameLast': 'Patterson', 'nameGiven': 'Jordan Andrew', 'debut': '2016-09-08', 'finalGame': '2016-10-02'}, {'playerID': 'paigesa01', 'nameFirst': 'Satchel', 'nameLast': 'Paige', 'nameGiven': 'Leroy Robert', 'debut': '1948-07-09', 'finalGame': '1965-09-25'}, {'playerID': 'otisam01', 'nameFirst': 'Amos', 'nameLast': 'Otis', 'nameGiven': 'Amos Joseph', 'debut': '1967-09-06', 'finalGame': '1984-08-05'}, {'playerID': 'milnepe01', 'nameFirst': 'Pete', 'nameLast': 'Milne', 'nameGiven': 'William James', 'debut': '1948-09-15', 'finalGame': '1950-05-14'}, {'playerID': 'merchan01', 'nameFirst': 'Andy', 'nameLast': 'Merchant', 'nameGiven': 'James Anderson', 'debut': '1975-09-28', 'finalGame': '1976-06-10'}, {'playerID': 'mcgilra01', 'nameFirst': 'Randy', 'nameLast': 'McGilberry', 'nameGiven': 'Randall Kent', 'debut': '1977-09-06', 'finalGame': '1978-10-01'}, {'playerID': 'mccovwi01', 'nameFirst': 'Willie', 'nameLast': 'McCovey', 'nameGiven': 'Willie Lee', 'debut': '1959-07-30', 'finalGame': '1980-07-06'}, {'playerID': 'masonji01', 'nameFirst': 'Jim', 'nameLast': 'Mason', 'nameGiven': 'James Percy', 'debut': '1971-09-26', 'finalGame': '1979-09-24'}, {'playerID': 'kellyhe01', 'nameFirst': 'Herb', 'nameLast': 'Kelly', 'nameGiven': 'Herbert Barrett', 'debut': '1914-09-21', 'finalGame': '1915-09-06'}, {'playerID': 'jumonge01', 'nameFirst': 'George', 'nameLast': 'Jumonville', 'nameGiven': 'George Benedict', 'debut': '1940-09-13', 'finalGame': '1941-05-20'}, {'playerID': 'howelpa01', 'nameFirst': 'Pat', 'nameLast': 'Howell', 'nameGiven': "Patrick O'Neal", 'debut': '1992-07-10', 'finalGame': '1992-10-04'}, {'playerID': 'hoodde01', 'nameFirst': 'Destin', 'nameLast': 'Hood', 'nameGiven': 'Destin Dwane', 'debut': '2016-09-02', 'finalGame': '2016-10-02'}, {'playerID': 'henlebo01', 'nameFirst': 'Bob', 'nameLast': 'Henley', 'nameGiven': 'Robert Clifton', 'debut': '1998-07-19', 'finalGame': '1998-09-26'}, {'playerID': 'hazewdr01', 'nameFirst': 'Drungo', 'nameLast': 'Hazewood', 'nameGiven': 'Drungo LaRue', 'debut': '1980-09-19', 'finalGame': '1980-10-04'}, {'playerID': 'fritzch01', 'nameFirst': 'Charlie', 'nameLast': 'Fritz', 'nameGiven': 'Charles Cornelius', 'debut': '1907-10-05', 'finalGame': '1907-10-05'}, {'playerID': 'frankch01', 'nameFirst': 'Charlie', 'nameLast': 'Frank', 'nameGiven': 'Charles', 'debut': '1893-08-18', 'finalGame': '1894-07-25'}, {'playerID': 'ervinph01', 'nameFirst': 'Phil', 'nameLast': 'Ervin', 'nameGiven': 'Phillip S.', 'debut': '2017-04-22', 'finalGame': '2018-09-30'}, {'playerID': 'duncaco01', 'nameFirst': 'Courtney', 'nameLast': 'Duncan', 'nameGiven': 'Courtney Demond', 'debut': '2001-04-02', 'finalGame': '2002-06-01'}, {'playerID': 'duffech01', 'nameFirst': 'Charlie', 'nameLast': 'Duffee', 'nameGiven': 'Charles Edward', 'debut': '1889-04-17', 'finalGame': '1893-04-30'}, {'playerID': 'davisto03', 'nameFirst': 'Tommy', 'nameLast': 'Davis', 'nameGiven': 'Thomas James', 'debut': '1999-05-14', 'finalGame': '1999-05-30'}, {'playerID': 'bradfbu01', 'nameFirst': 'Buddy', 'nameLast': 'Bradford', 'nameGiven': 'Charles William', 'debut': '1966-09-09', 'finalGame': '1976-07-24'}, {'playerID': 'bollija01', 'nameFirst': 'Jack', 'nameLast': 'Bolling', 'nameGiven': 'John Edward', 'debut': '1939-06-10', 'finalGame': '1944-09-20'}, {'playerID': 'bollifr01', 'nameFirst': 'Frank', 'nameLast': 'Bolling', 'nameGiven': 'Frank Elmore', 'debut': '1954-04-13', 'finalGame': '1966-09-15'}, {'playerID': 'adamste01', 'nameFirst': 'Terry', 'nameLast': 'Adams', 'nameGiven': 'Terry Wayne', 'debut': '1995-08-10', 'finalGame': '2005-05-23'}, {'playerID': 'adairbi99', 'nameFirst': 'Bill', 'nameLast': 'Adair', 'nameGiven': 'Marion Danne', 'debut': '', 'finalGame': ''}, {'playerID': 'aaronto01', 'nameFirst': 'Tommie', 'nameLast': 'Aaron', 'nameGiven': 'Tommie Lee', 'debut': '1962-04-10', 'finalGame': '1971-09-26'}, {'playerID': 'aaronha01',

‘nameFirst’: ‘Hank’, ‘nameLast’: ‘Aaron’, ‘nameGiven’: ‘Henry Louis’, ‘debut’: ‘1954-04-13’, ‘finalGame’: ‘1976-10-03’}]

find_by_template(): Unknown Record with Empty Data Possibility - Field List Projected []

delete_by_key(): Known Record 1

delete_by_key(): UnKnown Record 0

delete_by_template(): Known Record 1

delete_by_template(): Known Records - Multiple Entries Matching 119

delete_by_template(): UnKnown Record 0

delete_by_template(): UnKnown Key entered 0

update_by_key(): Update Known Record’s Columns - Columns don’t contain the primary key components Record BEFORE updation: {‘playerID’: ‘akersje01’, ‘birthYear’: ‘1887’, ‘birthMonth’: ‘11’, ‘birthDay’: ‘1’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘IN’, ‘birthCity’: ‘Shelbyville’, ‘deathYear’: ‘1979’, ‘deathMonth’: ‘5’, ‘deathDay’: ‘15’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘FL’, ‘deathCity’: ‘Bay Pines’, ‘nameFirst’: ‘Jerry’, ‘nameLast’: ‘Akers’, ‘nameGiven’: ‘Albert Earl’, ‘weight’: ‘175’, ‘height’: ‘71’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1912-05-04’, ‘finalGame’: ‘1912-05-25’, ‘retroID’: ‘akerj101’, ‘bbrefID’: ‘akersje01’} 1 Record AFTER updation: {‘playerID’: ‘akersje01’, ‘birthYear’: ‘1891’, ‘birthMonth’: ‘2’, ‘birthDay’: ‘21’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘IN’, ‘birthCity’: ‘Shelbyville’, ‘deathYear’: ‘1979’, ‘deathMonth’: ‘5’, ‘deathDay’: ‘15’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘FL’, ‘deathCity’: ‘Bay Pines’, ‘nameFirst’: ‘Jerry’, ‘nameLast’: ‘Akers’, ‘nameGiven’: ‘Albert Earl’, ‘weight’: ‘175’, ‘height’: ‘71’, ‘bats’: ‘R’, ‘throws’: ‘R’, ‘debut’: ‘1912-05-04’, ‘finalGame’: ‘1912-05-25’, ‘retroID’: ‘akerj101’, ‘bbrefID’: ‘akersje01’}

update_by_key(): Update UnKnown Record’s Columns Record BEFORE updation: None 0 Record AFTER updation: None

update_by_key(): Update Known Record’s Columns - Columns contain the primary key components Record BEFORE updation: {‘playerID’: ‘albieoz01’, ‘birthYear’: ‘1997’, ‘birthMonth’: ‘1’, ‘birthDay’: ‘7’, ‘birthCountry’: ‘Curacao’, ‘birthState’: ‘‘, ’birthCity’: ‘Willemstad’, ‘deathYear’: ‘‘, ’deathMonth’: ‘‘, ’deathDay’: ‘‘, ’deathCountry’: ‘‘, ’deathState’: ‘‘, ’deathCity’: ‘‘, ’nameFirst’: ‘Ozzie’, ‘nameLast’: ‘Albies’, ‘nameGiven’: ‘Ozhaino Jurdy Jiandro’, ‘weight’: ‘165’, ‘height’: ‘68’, ‘bats’: ‘B’, ‘throws’: ‘R’, ‘debut’: ‘2017-08-01’, ‘finalGame’: ‘2018-09-30’, ‘retroID’: ‘albio001’, ‘bbrefID’: ‘albieoz01’} 0 Record AFTER updation: {‘playerID’: ‘albieoz01’, ‘birthYear’: ‘1997’, ‘birthMonth’: ‘1’, ‘birthDay’: ‘7’, ‘birthCountry’: ‘Curacao’, ‘birthState’: ‘‘, ’birthCity’: ‘Willemstad’, ‘deathYear’: ‘‘, ’deathMonth’: ‘‘, ’deathDay’: ‘‘, ’deathCountry’: ‘‘, ’deathState’: ‘‘, ’deathCity’: ‘‘, ’nameFirst’: ‘Ozzie’, ‘nameLast’: ‘Albies’, ‘nameGiven’: ‘Ozhaino Jurdy Jiandro’, ‘weight’: ‘165’, ‘height’: ‘68’, ‘bats’: ‘B’, ‘throws’: ‘R’, ‘debut’: ‘2017-08-01’, ‘finalGame’: ‘2018-09-30’, ‘retroID’: ‘albio001’, ‘bbrefID’: ‘albieoz01’} Duplicate Record Found: {‘playerID’: ‘alburvi01’, ‘birthYear’: ‘1947’, ‘birthMonth’: ‘5’, ‘birthDay’: ‘12’, ‘birthCountry’: ‘USA’, ‘birthState’: ‘FL’, ‘birthCity’: ‘Key West’, ‘deathYear’: ‘2017’, ‘deathMonth’: ‘4’, ‘deathDay’: ‘18’, ‘deathCountry’: ‘USA’, ‘deathState’: ‘FL’, ‘deathCity’: ‘Tampa’, ‘nameFirst’: ‘Vic’, ‘nameLast’: ‘Albury’, ‘nameGiven’: ‘Victor’, ‘weight’: ‘190’, ‘height’: ‘72’, ‘bats’: ‘L’, ‘throws’: ‘L’, ‘debut’: ‘1973-08-07’, ‘finalGame’: ‘1976-09-26’, ‘retroID’: ‘albuv101’, ‘bbrefID’: ‘alburvi01’}

update_by_template(): Update Known Records’ Columns - Columns don’t contain the primary key components Records BEFORE updation: [{‘playerID’: ‘akersje01’, ‘birthYear’: ‘1891’, ‘birth-

Month': '2', 'birthDay': '21', 'birthCountry': 'USA', 'birthState': 'IN', 'birthCity': 'Shelbyville', 'deathYear': '1979', 'deathMonth': '5', 'deathDay': '15', 'deathCountry': 'USA', 'deathState': 'FL', 'deathCity': 'Bay Pines', 'nameFirst': 'Jerry', 'nameLast': 'Akers', 'nameGiven': 'Albert Earl', 'weight': '175', 'height': '71', 'bats': 'R', 'throws': 'R', 'debut': '1912-05-04', 'finalGame': '1912-05-25', 'retroID': 'akerj101', 'bbrefID': 'akersje01'}] 1 Records AFTER updation: [{'playerID': 'akersje01', 'birthYear': '1891', 'birthMonth': '2', 'birthDay': '21', 'birthCountry': 'USA', 'birthState': 'IN', 'birthCity': 'Shelbyville', 'deathYear': '1994', 'deathMonth': '7', 'deathDay': '14', 'deathCountry': 'USA', 'deathState': 'FL', 'deathCity': 'Bay Pines', 'nameFirst': 'Jerry', 'nameLast': 'Akers', 'nameGiven': 'Albert Earl', 'weight': '175', 'height': '71', 'bats': 'R', 'throws': 'R', 'debut': '1912-05-04', 'finalGame': '1912-05-25', 'retroID': 'akerj101', 'bbrefID': 'akersje01'}]

update_by_template(): Update UnKnown Records' Columns Records BEFORE updation: [] 0 Records AFTER updation: []

update_by_template(): Update Known Record's Columns - Columns contain the primary key components Records BEFORE updation: [{'playerID': 'akersje01', 'birthYear': '1891', 'birthMonth': '2', 'birthDay': '21', 'birthCountry': 'USA', 'birthState': 'IN', 'birthCity': 'Shelbyville', 'deathYear': '1994', 'deathMonth': '7', 'deathDay': '14', 'deathCountry': 'USA', 'deathState': 'FL', 'deathCity': 'Bay Pines', 'nameFirst': 'Jerry', 'nameLast': 'Akers', 'nameGiven': 'Albert Earl', 'weight': '175', 'height': '71', 'bats': 'R', 'throws': 'R', 'debut': '1912-05-04', 'finalGame': '1912-05-25', 'retroID': 'akerj101', 'bbrefID': 'akersje01'}] 0 Records AFTER updation: [{'playerID': 'akersje01', 'birthYear': '1891', 'birthMonth': '2', 'birthDay': '21', 'birthCountry': 'USA', 'birthState': 'IN', 'birthCity': 'Shelbyville', 'deathYear': '1994', 'deathMonth': '7', 'deathDay': '14', 'deathCountry': 'USA', 'deathState': 'FL', 'deathCity': 'Bay Pines', 'nameFirst': 'Jerry', 'nameLast': 'Akers', 'nameGiven': 'Albert Earl', 'weight': '175', 'height': '71', 'bats': 'R', 'throws': 'R', 'debut': '1912-05-04', 'finalGame': '1912-05-25', 'retroID': 'akerj101', 'bbrefID': 'akersje01'}]

insert(): No Duplication of Primary Key None AFTER Insertion: [{'playerID': 'chansu4090', 'birthYear': '1996', 'birthMonth': '2', 'birthDay': '21', 'birthCountry': 'India', 'birthState': 'UP', 'birthCity': 'Lucknow', 'deathYear': '', 'deathMonth': '', 'deathDay': '', 'deathCountry': '', 'deathState': '', 'deathCity': '', 'nameFirst': 'Chandan', 'nameLast': 'Suri', 'nameGiven': 'Chandan', 'weight': '180', 'height': '72', 'bats': 'R', 'throws': 'R', 'debut': '2010-05-24', 'finalGame': '2020-02-26', 'retroID': 'chansuri49', 'bbrefID': 'chansu007'}]

insert(): Duplication of Primary Key New record cannot be added as primary key already present in the database! None AFTER Insertion: [] Record with Duplicate PK Present: {'playerID': 'chansu4090', 'birthYear': '1996', 'birthMonth': '2', 'birthDay': '21', 'birthCountry': 'India', 'birthState': 'UP', 'birthCity': 'Lucknow', 'deathYear': '', 'deathMonth': '', 'deathDay': '', 'deathCountry': '', 'deathState': '', 'deathCity': '', 'nameFirst': 'Chandan', 'nameLast': 'Suri', 'nameGiven': 'Chandan', 'weight': '180', 'height': '72', 'bats': 'R', 'throws': 'R', 'debut': '2010-05-24', 'finalGame': '2020-02-26', 'retroID': 'chansuri49', 'bbrefID': 'chansu007'}

insert(): Primary Key Not Present as part of Insertion and No Default Set New record cannot be added as missing a primary key entry or the value entered for primary key or a component of primary key is NULL! None AFTER Insertion: []

insert(): Primary Key Value set to NULL New record cannot be added as missing a primary key entry or the value entered for primary key or a component of primary key is NULL! None AFTER Insertion: []

find_by_primary_key(): Known Record {'playerID': 'abercda01', 'yearID': '1871', 'stint': '1',

‘teamID’: ‘TRO’, ‘lgID’: ‘NA’, ‘G’: ‘1’, ‘AB’: ‘4’, ‘R’: ‘0’, ‘H’: ‘0’, ‘2B’: ‘0’, ‘3B’: ‘0’, ‘HR’: ‘0’, ‘RBI’: ‘0’, ‘SB’: ‘0’, ‘CS’: ‘0’, ‘BB’: ‘0’, ‘SO’: ‘0’, ‘IBB’: ’‘, ’HBP’: ’‘, ’SH’: ’‘, ’SF’: ’‘, ’GIDP’: ‘0’}

find_by_primary_key(): Unknown Record - Partially Not Matching None

find_by_primary_key(): Unknown Record - Completely Not Matching None

find_by_primary_key(): Known Record with Field List Projected {‘playerID’: ‘goldswa01’, ‘yearID’: ‘1871’, ‘stint’: ‘1’, ‘teamID’: ‘FW1’, ‘lgID’: ‘NA’, ‘G’: ‘19’, ‘AB’: ‘88’, ‘R’: ‘8’, ‘H’: ‘18’}

find_by_primary_key(): Unknown Record - Partially Not Matching with Field List Projected None

find_by_primary_key(): Unknown Record - Completely Not Matching with Field List Projected None

find_by_template(): Known Template [{‘playerID’: ‘malaych01’, ‘yearID’: ‘1905’, ‘stint’: ‘1’, ‘teamID’: ‘BRO’, ‘lgID’: ‘NL’, ‘G’: ‘102’, ‘AB’: ‘349’, ‘R’: ‘33’, ‘H’: ‘88’, ‘2B’: ‘7’, ‘3B’: ‘2’, ‘HR’: ‘1’, ‘RBI’: ‘31’, ‘SB’: ‘13’, ‘CS’: ’‘, ’BB’: ‘22’, ‘SO’: ‘45’, ‘IBB’: ’‘, ’HBP’: ‘2’, ‘SH’: ‘14’, ‘SF’: ’‘, ’GIDP’: ’‘}, {’playerID’: ‘robinwi01’, ‘yearID’: ‘1888’, ‘stint’: ‘1’, ‘teamID’: ‘PH4’, ‘lgID’: ‘AA’, ‘G’: ‘66’, ‘AB’: ‘254’, ‘R’: ‘32’, ‘H’: ‘62’, ‘2B’: ‘7’, ‘3B’: ‘2’, ‘HR’: ‘1’, ‘RBI’: ‘31’, ‘SB’: ‘11’, ‘CS’: ’‘, ’BB’: ‘9’, ‘SO’: ‘26’, ‘IBB’: ’‘, ’HBP’: ‘0’, ‘SH’: ’‘, ’SF’: ’‘, ’GIDP’: ’‘}, {’playerID’: ‘ganzech01’, ‘yearID’: ‘1886’, ‘stint’: ‘2’, ‘teamID’: ‘DTN’, ‘lgID’: ‘NL’, ‘G’: ‘57’, ‘AB’: ‘213’, ‘R’: ‘28’, ‘H’: ‘58’, ‘2B’: ‘7’, ‘3B’: ‘2’, ‘HR’: ‘1’, ‘RBI’: ‘31’, ‘SB’: ‘5’, ‘CS’: ’‘, ’BB’: ‘7’, ‘SO’: ‘22’, ‘IBB’: ’‘, ’HBP’: ’‘, ’SH’: ’‘, ’SF’: ’‘, ’GIDP’: ’‘}, {’playerID’: ‘treacfr01’, ‘yearID’: ‘1873’, ‘stint’: ‘1’, ‘teamID’: ‘PH2’, ‘lgID’: ‘NA’, ‘G’: ‘51’, ‘AB’: ‘243’, ‘R’: ‘49’, ‘H’: ‘62’, ‘2B’: ‘7’, ‘3B’: ‘2’, ‘HR’: ‘1’, ‘RBI’: ‘31’, ‘SB’: ‘4’, ‘CS’: ‘3’, ‘BB’: ‘5’, ‘SO’: ‘7’, ‘IBB’: ’‘, ’HBP’: ’‘, ’SH’: ’‘, ’SF’: ’‘, ’GIDP’: ‘2’}]

find_by_template(): Known Template with subpart of Composite PK [{‘playerID’: ‘ganzech01’, ‘yearID’: ‘1886’, ‘stint’: ‘2’, ‘teamID’: ‘DTN’, ‘lgID’: ‘NL’, ‘G’: ‘57’, ‘AB’: ‘213’, ‘R’: ‘28’, ‘H’: ‘58’, ‘2B’: ‘7’, ‘3B’: ‘2’, ‘HR’: ‘1’, ‘RBI’: ‘31’, ‘SB’: ‘5’, ‘CS’: ’‘, ’BB’: ‘7’, ‘SO’: ‘22’, ‘IBB’: ’‘, ’HBP’: ’‘, ’SH’: ’‘, ’SF’: ’‘, ’GIDP’: ’ ’}]

find_by_template(): Unknown Record with Empty Data Possibility []

find_by_template(): Known Template - Field List Projected [{‘playerID’: ‘harrile01’, ‘yearID’: ‘1994’, ‘teamID’: ‘CIN’, ‘lgID’: ‘NL’, ‘G’: ‘66’, ‘AB’: ‘100’, ‘R’: ‘13’, ‘H’: ‘31’}]

find_by_template(): Unknown Record with Empty Data Possibility - Field List Projected []

delete_by_key(): Known Record 1

delete_by_key(): Unknown Record - Partially Matching Keys 0

delete_by_key(): Unknown Record - Not Matching 0

delete_by_template(): Known Record 13

delete_by_template(): UnKnown Record 0

delete_by_template(): UnKnown Key entered 0

update_by_key(): Update Known Record’s Columns - Columns don’t contain the primary key components Record BEFORE updation: {‘playerID’: ‘stanlbo01’, ‘yearID’: ‘1984’, ‘stint’: ‘1’, ‘teamID’: ‘BOS’, ‘lgID’: ‘AL’, ‘G’: ‘57’, ‘AB’: ‘0’, ‘R’: ‘0’, ‘H’: ‘0’, ‘2B’: ‘0’, ‘3B’: ‘0’, ‘HR’: ‘0’, ‘RBI’: ‘0’, ‘SB’: ‘0’, ‘CS’: ‘0’, ‘BB’: ‘0’, ‘SO’: ‘0’, ‘IBB’: ‘0’, ‘HBP’: ‘0’, ‘SH’: ‘0’, ‘SF’: ‘0’, ‘GIDP’:

‘0’} 1 Record AFTER updation: {‘playerID’: ‘stanlbo01’, ‘yearID’: ‘1984’, ‘stint’: ‘1’, ‘teamID’: ‘BOS’, ‘lgID’: ‘NL’, ‘G’: ‘28’, ‘AB’: ‘49’, ‘R’: ‘22’, ‘H’: ‘44’, ‘2B’: ‘0’, ‘3B’: ‘0’, ‘HR’: ‘0’, ‘RBI’: ‘0’, ‘SB’: ‘0’, ‘CS’: ‘0’, ‘BB’: ‘0’, ‘SO’: ‘0’, ‘IBB’: ‘0’, ‘HBP’: ‘0’, ‘SH’: ‘0’, ‘SF’: ‘0’, ‘GIDP’: ‘0’}

update_by_key(): Update UnKnown Record’s Columns - Partially Matching PKs - Test 1 Record BEFORE updation: None 0 Record AFTER updation: None

update_by_key(): Update UnKnown Record’s Columns - Partially Matching PKs - Test 2 Record BEFORE updation: None 0 Record AFTER updation: None

update_by_key(): Update Known Record’s Columns - Columns contain the primary key components - Test 1 Record BEFORE updation: {‘playerID’: ‘gomezch02’, ‘yearID’: ‘1996’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘48’, ‘AB’: ‘128’, ‘R’: ‘21’, ‘H’: ‘31’, ‘2B’: ‘5’, ‘3B’: ‘0’, ‘HR’: ‘1’, ‘RBI’: ‘16’, ‘SB’: ‘1’, ‘CS’: ‘1’, ‘BB’: ‘18’, ‘SO’: ‘20’, ‘IBB’: ‘0’, ‘HBP’: ‘1’, ‘SH’: ‘3’, ‘SF’: ‘0’, ‘GIDP’: ‘5’} 0 Record AFTER updation: None

update_by_key(): Update Known Record’s Columns - Columns contain the primary key components - Test 2 Record BEFORE updation: {‘playerID’: ‘orourjo01’, ‘yearID’: ‘1879’, ‘stint’: ‘1’, ‘teamID’: ‘BSN’, ‘lgID’: ‘NL’, ‘G’: ‘72’, ‘AB’: ‘317’, ‘R’: ‘69’, ‘H’: ‘108’, ‘2B’: ‘17’, ‘3B’: ‘11’, ‘HR’: ‘6’, ‘RBI’: ‘62’, ‘SB’: ‘’, ’CS’: ‘’, ’BB’: ‘8’, ‘SO’: ‘32’, ‘IBB’: ‘’, ’HBP’: ‘’, ’SH’: ‘’, ’SF’: ‘’, ’GIDP’: ‘’} 0 Record AFTER updation: {’playerID’: ‘orourjo01’, ‘yearID’: ‘1879’, ‘stint’: ‘1’, ‘teamID’: ‘BSN’, ‘lgID’: ‘NL’, ‘G’: ‘72’, ‘AB’: ‘317’, ‘R’: ‘69’, ‘H’: ‘108’, ‘2B’: ‘17’, ‘3B’: ‘11’, ‘HR’: ‘6’, ‘RBI’: ‘62’, ‘SB’: ’’, ’CS’: ’’, ’BB’: ‘8’, ‘SO’: ‘32’, ‘IBB’: ’’, ’HBP’: ’’, ’SH’: ’’, ’SF’: ’’, ’GIDP’: ’’} Duplicate Record Found: {’playerID’: ‘quinnjo01’, ‘yearID’: ‘1881’, ‘stint’: ‘2’, ‘teamID’: ‘WOR’, ‘lgID’: ‘NL’, ‘G’: ‘2’, ‘AB’: ‘7’, ‘R’: ‘0’, ‘H’: ‘1’, ‘2B’: ‘0’, ‘3B’: ‘0’, ‘HR’: ‘0’, ‘RBI’: ‘1’, ‘SB’: ’’, ’CS’: ’’, ’BB’: ‘1’, ‘SO’: ‘2’, ‘IBB’: ’’, ’HBP’: ’’, ’SH’: ’’, ’SF’: ’’, ’GIDP’: ’ ’}

update_by_template(): Update Known Records’ Columns - Columns don’t contain the primary key components Records BEFORE updation: [{‘playerID’: ‘gutiece01’, ‘yearID’: ‘1971’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘38’, ‘AB’: ‘37’, ‘R’: ‘8’, ‘H’: ‘7’, ‘2B’: ‘0’, ‘3B’: ‘0’, ‘HR’: ‘0’, ‘RBI’: ‘4’, ‘SB’: ‘0’, ‘CS’: ‘0’, ‘BB’: ‘0’, ‘SO’: ‘3’, ‘IBB’: ‘0’, ‘HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ‘0’, ‘GIDP’: ‘0’}, {‘playerID’: ‘wilsoea01’, ‘yearID’: ‘1968’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘40’, ‘AB’: ‘88’, ‘R’: ‘9’, ‘H’: ‘20’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘7’, ‘RBI’: ‘17’, ‘SB’: ‘0’, ‘CS’: ‘0’, ‘BB’: ‘2’, ‘SO’: ‘35’, ‘IBB’: ‘0’, ‘HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ‘0’, ‘GIDP’: ‘1’}, {‘playerID’: ‘maniocl01’, ‘yearID’: ‘1921’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘12’, ‘AB’: ‘10’, ‘R’: ‘0’, ‘H’: ‘2’, ‘2B’: ‘0’, ‘3B’: ‘0’, ‘HR’: ‘0’, ‘RBI’: ‘2’, ‘SB’: ‘0’, ‘CS’: ‘0’, ‘BB’: ‘2’, ‘SO’: ‘2’, ‘IBB’: ’’, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’’, ’GIDP’: ’’}, {’playerID’: ‘mitchwi01’, ‘yearID’: ‘1916’, ‘stint’: ‘2’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘23’, ‘AB’: ‘36’, ‘R’: ‘3’, ‘H’: ‘9’, ‘2B’: ‘0’, ‘3B’: ‘0’, ‘HR’: ‘0’, ‘RBI’: ‘3’, ‘SB’: ‘0’, ‘CS’: ’’, ’BB’: ‘5’, ‘SO’: ‘6’, ‘IBB’: ’’, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’’, ’GIDP’: ’’}, {’playerID’: ‘schalbi01’, ‘yearID’: ‘1911’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘40’, ‘AB’: ‘60’, ‘R’: ‘8’, ‘H’: ‘8’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘1’, ‘RBI’: ‘7’, ‘SB’: ‘1’, ‘CS’: ’’, ’BB’: ‘4’, ‘SO’: ’’, ’IBB’: ’’, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’’, ’GIDP’: ’ ’}] 5 Records AFTER updation: [{‘playerID’: ‘gutiece01’, ‘yearID’: ‘1971’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘38’, ‘AB’: ‘37’, ‘R’: ‘8’, ‘H’: ‘7’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘4’, ‘SB’: ‘1’, ‘CS’: ‘0’, ‘BB’: ‘0’, ‘SO’: ‘3’, ‘IBB’: ‘0’, ‘HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ‘0’, ‘GIDP’: ‘0’}, {‘playerID’: ‘wilsoea01’, ‘yearID’: ‘1968’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘40’, ‘AB’: ‘88’, ‘R’: ‘9’, ‘H’: ‘20’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘17’, ‘SB’: ‘1’, ‘CS’: ‘0’, ‘BB’: ‘2’, ‘SO’: ‘35’, ‘IBB’: ‘0’, ‘HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ‘0’, ‘GIDP’: ‘1’}, {‘playerID’: ‘maniocl01’, ‘yearID’: ‘1921’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘12’, ‘AB’: ‘10’, ‘R’: ‘0’, ‘H’: ‘2’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘2’, ‘SB’: ‘1’, ‘CS’: ‘0’, ‘BB’: ‘2’, ‘SO’: ‘2’, ‘IBB’: ’’, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’’, ’GIDP’: ’’}, {’playerID’: ‘mitchwi01’, ‘yearID’: ‘1916’, ‘stint’: ‘2’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’,

‘G’: ‘23’, ‘AB’: ‘36’, ‘R’: ‘3’, ‘H’: ‘9’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘3’, ‘SB’: ‘1’, ‘CS’: ’‘, ’BB’: ‘5’, ‘SO’: ‘6’, ‘IBB’: ’‘, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’‘, ’GIDP’: ’‘}, {’playerID’: ‘schalbi01’, ‘yearID’: ‘1911’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘40’, ‘AB’: ‘60’, ‘R’: ‘8’, ‘H’: ‘8’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘7’, ‘SB’: ‘1’, ‘CS’: ’‘, ’BB’: ‘4’, ‘SO’: ’‘, ’IBB’: ’‘, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’‘, ’GIDP’: ’ ’}]

update_by_template(): Update UnKnown Records’ Columns Records BEFORE updation: [] 0 Records AFTER updation: []

update_by_template(): Update Known Record’s Columns - Columns contain the primary key components Records BEFORE updation: [{‘playerID’: ‘gutiece01’, ‘yearID’: ‘1971’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘38’, ‘AB’: ‘37’, ‘R’: ‘8’, ‘H’: ‘7’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘4’, ‘SB’: ‘1’, ‘CS’: ‘0’, ‘BB’: ‘0’, ‘SO’: ‘3’, ‘IBB’: ‘0’, ‘HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ‘0’, ‘GIDP’: ‘0’}, {‘playerID’: ‘wilsoea01’, ‘yearID’: ‘1968’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘40’, ‘AB’: ‘88’, ‘R’: ‘9’, ‘H’: ‘20’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘17’, ‘SB’: ‘1’, ‘CS’: ‘0’, ‘BB’: ‘2’, ‘SO’: ‘35’, ‘IBB’: ‘0’, ‘HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ‘0’, ‘GIDP’: ‘1’}, {‘playerID’: ‘maniocl01’, ‘yearID’: ‘1921’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘12’, ‘AB’: ‘10’, ‘R’: ‘0’, ‘H’: ‘2’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘2’, ‘SB’: ‘1’, ‘CS’: ‘0’, ‘BB’: ‘2’, ‘SO’: ‘2’, ‘IBB’: ’‘, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’‘, ’GIDP’: ’‘}, {’playerID’: ‘mitchwi01’, ‘yearID’: ‘1916’, ‘stint’: ‘2’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘23’, ‘AB’: ‘36’, ‘R’: ‘3’, ‘H’: ‘9’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘3’, ‘SB’: ‘1’, ‘CS’: ’‘, ’BB’: ‘5’, ‘SO’: ‘6’, ‘IBB’: ’‘, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’‘, ’GIDP’: ’‘}, {’playerID’: ‘schalbi01’, ‘yearID’: ‘1911’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘40’, ‘AB’: ‘60’, ‘R’: ‘8’, ‘H’: ‘8’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘7’, ‘SB’: ‘1’, ‘CS’: ’‘, ’BB’: ‘4’, ‘SO’: ’‘, ’IBB’: ’‘, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’‘, ’GIDP’: ’ ’}] 0 Records AFTER updation: [{‘playerID’: ‘gutiece01’, ‘yearID’: ‘1971’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘38’, ‘AB’: ‘37’, ‘R’: ‘8’, ‘H’: ‘7’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘4’, ‘SB’: ‘1’, ‘CS’: ‘0’, ‘BB’: ‘0’, ‘SO’: ‘3’, ‘IBB’: ‘0’, ‘HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ‘0’, ‘GIDP’: ‘0’}, {‘playerID’: ‘wilsoea01’, ‘yearID’: ‘1968’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘40’, ‘AB’: ‘88’, ‘R’: ‘9’, ‘H’: ‘20’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘17’, ‘SB’: ‘1’, ‘CS’: ‘0’, ‘BB’: ‘2’, ‘SO’: ‘35’, ‘IBB’: ‘0’, ‘HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ‘0’, ‘GIDP’: ‘1’}, {‘playerID’: ‘maniocl01’, ‘yearID’: ‘1921’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘12’, ‘AB’: ‘10’, ‘R’: ‘0’, ‘H’: ‘2’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘2’, ‘SB’: ‘1’, ‘CS’: ‘0’, ‘BB’: ‘2’, ‘SO’: ‘2’, ‘IBB’: ’‘, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’‘, ’GIDP’: ’‘}, {’playerID’: ‘mitchwi01’, ‘yearID’: ‘1916’, ‘stint’: ‘2’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘23’, ‘AB’: ‘36’, ‘R’: ‘3’, ‘H’: ‘9’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘3’, ‘SB’: ‘1’, ‘CS’: ’‘, ’BB’: ‘5’, ‘SO’: ‘6’, ‘IBB’: ’‘, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’‘, ’GIDP’: ’‘}, {’playerID’: ‘schalbi01’, ‘yearID’: ‘1911’, ‘stint’: ‘1’, ‘teamID’: ‘DET’, ‘lgID’: ‘AL’, ‘G’: ‘40’, ‘AB’: ‘60’, ‘R’: ‘8’, ‘H’: ‘8’, ‘2B’: ‘0’, ‘3B’: ‘1’, ‘HR’: ‘5’, ‘RBI’: ‘7’, ‘SB’: ‘1’, ‘CS’: ’‘, ’BB’: ‘4’, ‘SO’: ’‘, ’IBB’: ’‘, ’HBP’: ‘1’, ‘SH’: ‘1’, ‘SF’: ’‘, ’GIDP’: ’ ’}]

insert(): No Duplication of Primary Key None AFTER Insertion: [{‘playerID’: ‘chansu4090’, ‘yearID’: ‘1996’, ‘stint’: ‘2’, ‘teamID’: ‘CS1’, ‘lgID’: ‘AL’, ‘G’: ‘99’, ‘AB’: ‘121’, ‘R’: ‘32’, ‘H’: ‘57’, ‘2B’: ‘6’, ‘3B’: ‘7’, ‘HR’: ‘2’, ‘RBI’: ‘321’, ‘SB’: ‘4’, ‘CS’: ‘3’, ‘BB’: ‘21’, ‘SO’: ‘3’, ‘IBB’: ‘0’, ‘HBP’: ’‘, ’SH’: ’‘, ’SF’: ’‘, ’GIDP’: ‘2’}]

insert(): Duplication of Primary Key New record cannot be added as primary key already present in the database! None AFTER Insertion: [] Record with Duplicate PK Present: {‘playerID’: ‘chansu4090’, ‘yearID’: ‘1996’, ‘stint’: ‘2’, ‘teamID’: ‘CS1’, ‘lgID’: ‘AL’, ‘G’: ‘99’, ‘AB’: ‘121’, ‘R’: ‘32’, ‘H’: ‘57’, ‘2B’: ‘6’, ‘3B’: ‘7’, ‘HR’: ‘2’, ‘RBI’: ‘321’, ‘SB’: ‘4’, ‘CS’: ‘3’, ‘BB’: ‘21’, ‘SO’: ‘3’, ‘IBB’: ‘0’, ‘HBP’: ’‘, ’SH’: ’‘, ’SF’: ’‘, ’GIDP’: ‘2’}

insert(): Primary Key Not Present as part of Insertion and No Default Set - Test 1 New record

cannot be added as missing a primary key entry or the value entered for primary key or a component of primary key is NULL! None AFTER Insertion: []

insert(): Primary Key Not Present as part of Insertion and No Default Set - Test 2 New record cannot be added as missing a primary key entry or the value entered for primary key or a component of primary key is NULL! None AFTER Insertion: []

insert(): Primary Key Not Present as part of Insertion and No Default Set - Test 3 New record cannot be added as missing a primary key entry or the value entered for primary key or a component of primary key is NULL! None AFTER Insertion: []

insert(): Primary Key Value set to NULL - Test 1 New record cannot be added as missing a primary key entry or the value entered for primary key or a component of primary key is NULL! None AFTER Insertion: []

insert(): Primary Key Value set to NULL - Test 2 New record cannot be added as missing a primary key entry or the value entered for primary key or a component of primary key is NULL! None AFTER Insertion: []

insert(): Primary Key Value set to NULL - Test 3 New record cannot be added as missing a primary key entry or the value entered for primary key or a component of primary key is NULL! None AFTER Insertion: []