

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.

Gupta Tushar, tg2749

I am unable to run magic commands (%%sql) in the notebook, hence have used pandas(read_sql_query) function to show the results of the tests. I have taken permission from Prof. Ferguson for the same. Please let me know in case of any questions

Introduction

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.

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

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.

Part A: Written

1. What is a database management system?

- DBMS software functions as an interface between the end user and the database. It provides tools for managing data, and performing various operations like visualization, manipulation addition/deletion etc. It also stores data in a structured format by defining relationships to ensure it can be managed in a convenient and logical manner.

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

- A column which is defined as primary key becomes the unique identifier for each row in the table. It is important as it helps to search and identify each record since it cannot contain a duplicate. Also, a primary key cannot have a NULL value

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

- SQL is the query language we use to fetch, alter and perform different operations on a database.
- SQL server is the platform which hosts the database so that we can use a 3rd party software to connect to it and do operations on the database
- Datagrip is a tool/editor which provides us a easy and convenient method to connect to a database hosted on a server and perform operations using support SQL language

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

- One to One: One record of a table relates to only one record of the other table
- One to Many : One record of a table is related to more than 1 record of other table
- Many to One : This would be reverse of the above relationship. Where many records of the first table relate to one record in the second table.
- Many to Many: This relationship ensures many record in the 1st table relate to more than record in table

5. What is an ER model?

- AN Entity relationship model is used to define entities, relationships and attributes used to build database system. It provides a graphical approach to make designing a database schema easier and more convenient. An Entity is a thing or object in real world that can independently exist in an environment. They can have attributes that define their properties. Relationships define how they interact with one another.

6. Using Lucidchart draw an example of a logical ER model using Crow's Foot notation for Columbia classes.

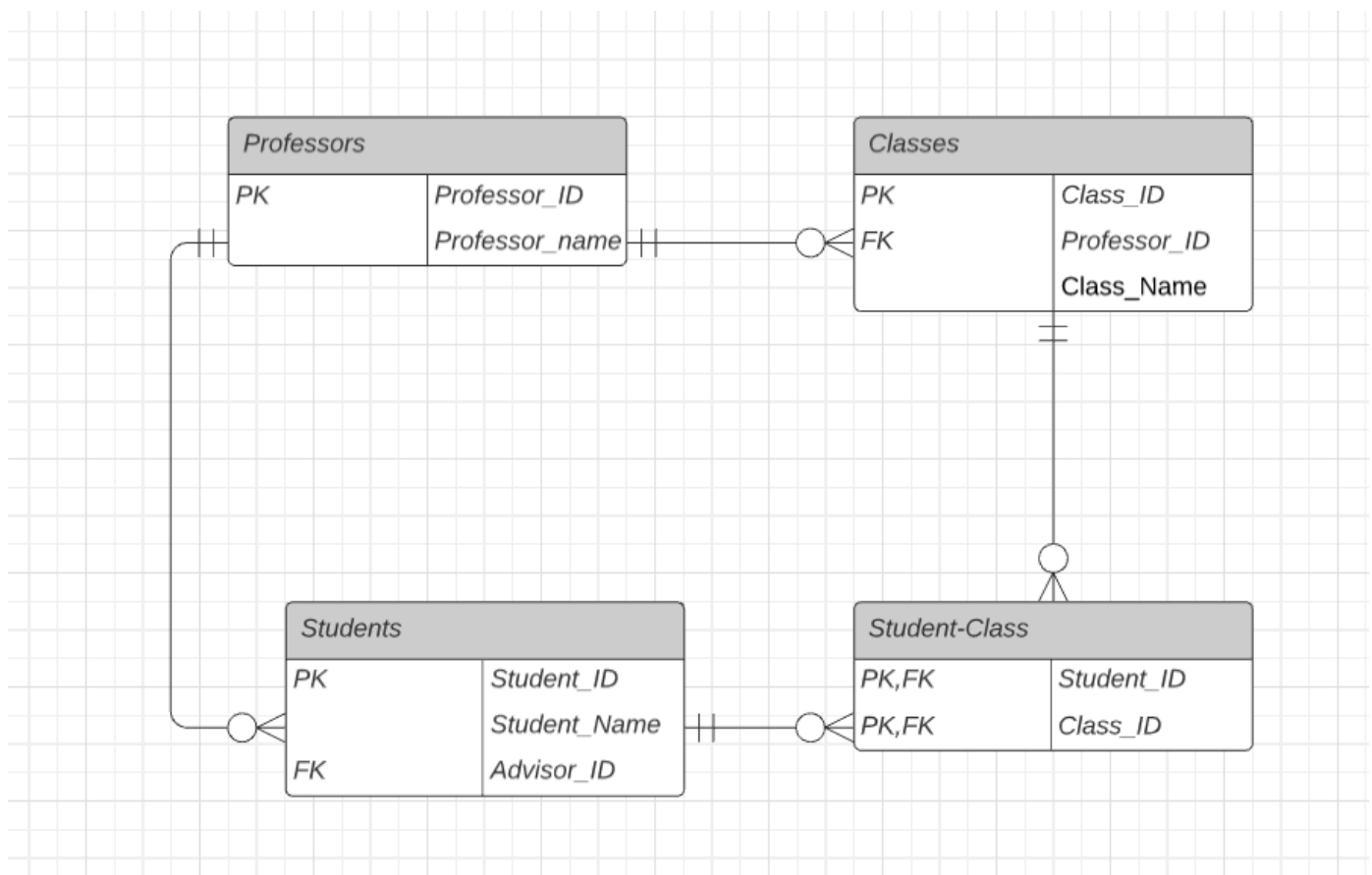
The entity types are:

- 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.



- Primary Key : Each student ID is unique hence added as a Primary Key
- Foreign Key : Advisor_ID relates to the Professor_ID and is used as a foreign key. It is used to identify professors who advise students
- One-to-Many : A professor can advise multiple students, which can be described as one-to-many relationships
- Many-to-Many : A student can take many classes and each class has many students which depicts a many-to-many relationship

Part B: Relational Algebra

You will use [the online relational calculator \(https://dbis-uibk.github.io/relax/landing\)](https://dbis-uibk.github.io/relax/landing), 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 \triangleright .

Consider the following relational algebra expression and result.

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

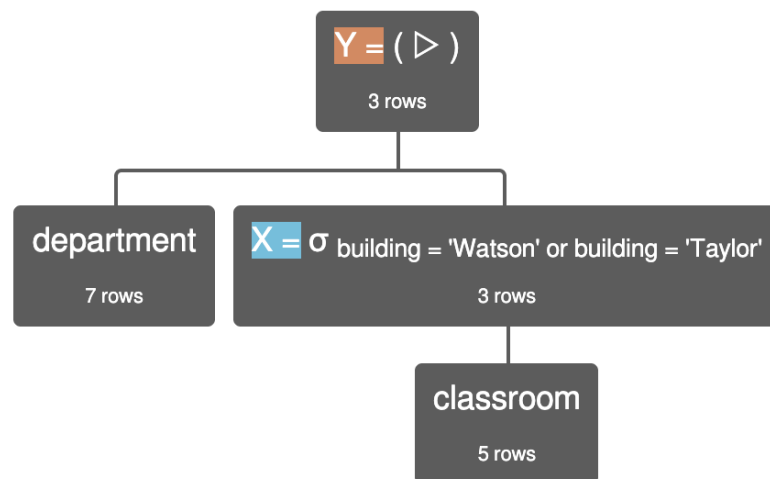
$X = \sigma \text{ building} = \text{'Watson'} \vee \text{ building} = \text{'Taylor'} \text{ (classroom)}$

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

$Y = (\text{department} \triangleright X)$

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

Y



$(\text{department} \triangleright \sigma \text{ building} = \text{'Watson'} \text{ or building} = \text{'Taylor'} (\text{classroom}))$

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

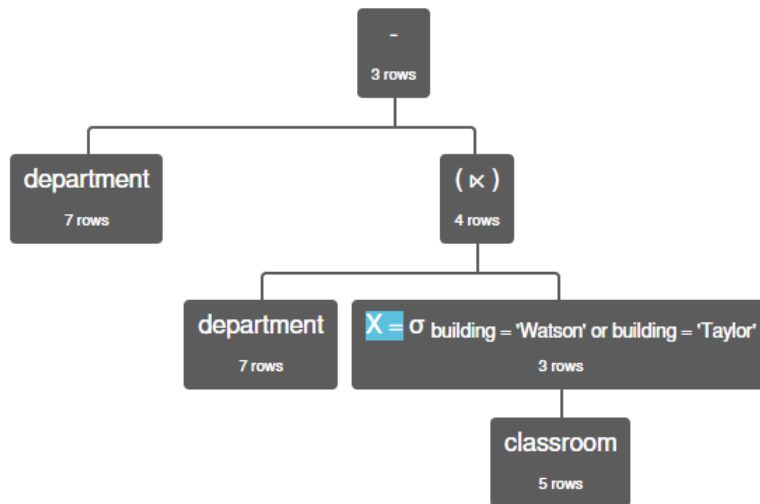
1. Find an alternate expression to (2) that computes the correct answer given X. Display the execution of your query below.

$Y = \text{department} - (\text{department} \bowtie X)$

In [4]:

```
from IPython.display import Image
Image("b.png")
```

Out[4]:



$\text{department} - (\text{department} \bowtie \sigma_{\text{building} = \text{'Watson'} \text{ or } \text{building} = \text{'Taylor'}} (\text{classroom}))$

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

Part C: Data Clean Up

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

Use the lahmansdb_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 receive a lot of deductions.

In [8]:

```
# 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:admin123@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.

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)

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

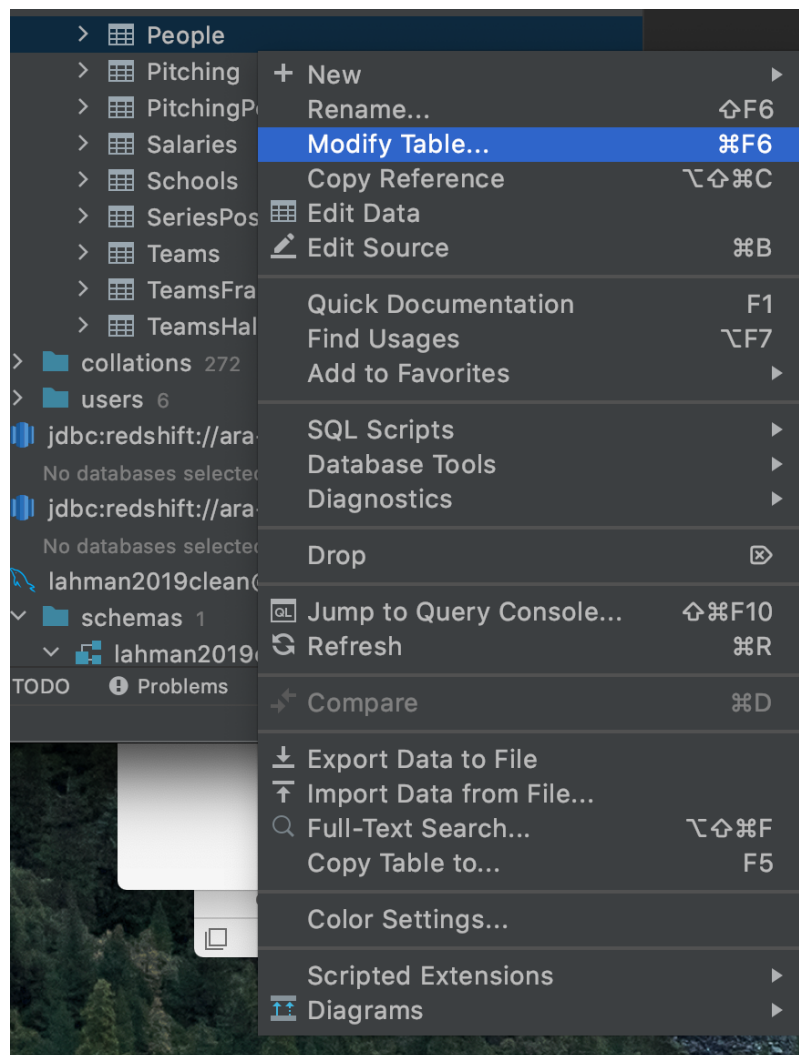
1. Empty strings to NULLs
2. Column typing
3. Primary Key
4. Foreign Key

How to make the changes:

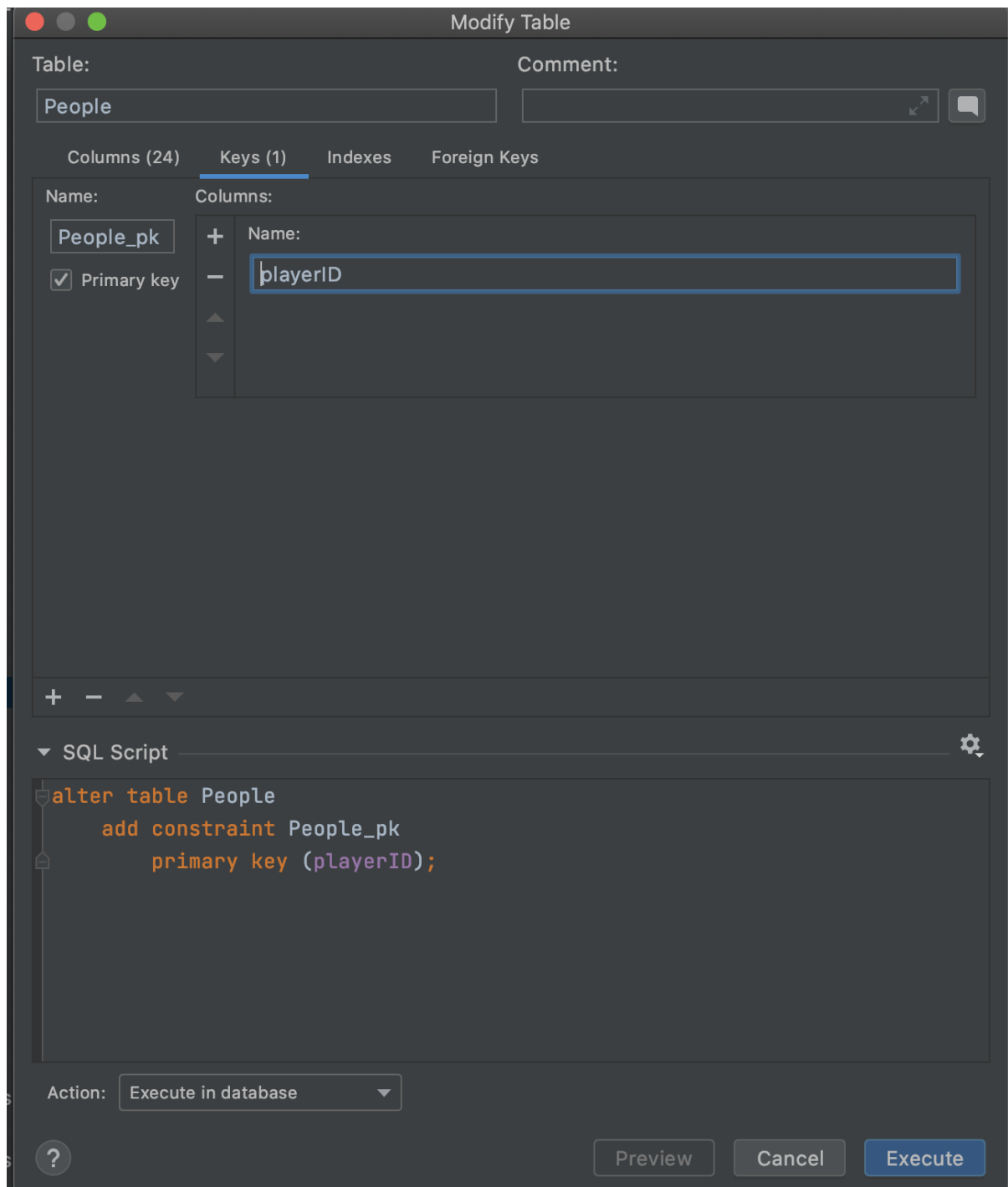
Using the Table Editor:

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. 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



Using sql queries:

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

People Table

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 a player within the data. A primary key is cannot be NULL and provides a easy way to search for information due to uniqueness.

Change

I added a Primary Key constraint 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

In [5]:

```
import pymysql
import pandas as pd
conn=pymysql.connect(host='localhost',port=int(3306),user='root',passwd='admin123', db = 'l
```

In [6]:

```
#!/sql SHOW KEYS FROM people WHERE Key_name = 'PRIMARY'

pd.read_sql_query("SHOW KEYS FROM people WHERE Key_name = 'PRIMARY';",conn)
```

Out[6]:

	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_pa
0	people	0	PRIMARY	1	playerID	A	19999	Non

1) Convert all empty strings to NULL

Explanation

Empty values are converted to NULL as NULL values are more efficiently stored and empty strings ('') take up 2 CHAR space. Also, it provides convience while converting between different datatypes. For eg: converting the string column with '' to INT will raise errors. Also, most functions raise error on NULL values which makes it easier to identify invalid data points whereas empty strings indicate a string is present however it is empty which can sometimes pass as a valid record.

Change

I used the UPDATE statement to SET each row as NULL whenever the value is an empty string

SQL

```
~~~~~sql
UPDATE lahmansdb_to_clean.people SET birthYear = NULL WHERE birthYear = '';
UPDATE lahmansdb_to_clean.people SET birthMonth = NULL WHERE birthMonth = '';
UPDATE lahmansdb_to_clean.people SET birthDay = NULL WHERE birthDay = '';
UPDATE lahmansdb_to_clean.people SET birthCountry = NULL WHERE birthCountry = '';
UPDATE lahmansdb_to_clean.people SET birthState = NULL WHERE birthState = '';
UPDATE lahmansdb_to_clean.people SET birthCity = NULL WHERE birthCity = '';
UPDATE lahmansdb_to_clean.people SET deathYear = NULL WHERE deathYear = '';
UPDATE lahmansdb_to_clean.people SET deathMonth = NULL WHERE deathMonth = '';
UPDATE lahmansdb_to_clean.people SET deathDay = NULL WHERE deathDay = '';
UPDATE lahmansdb_to_clean.people SET deathCountry = NULL WHERE deathCountry = '';
UPDATE lahmansdb_to_clean.people SET deathState = NULL WHERE deathState = '';
UPDATE lahmansdb_to_clean.people SET deathCity = NULL WHERE deathCity = '';
UPDATE lahmansdb_to_clean.people SET nameFirst = NULL WHERE nameFirst = '';
UPDATE lahmansdb_to_clean.people SET nameLast = NULL WHERE nameLast = '';
UPDATE lahmansdb_to_clean.people SET nameGiven = NULL WHERE nameGiven = '';
UPDATE lahmansdb_to_clean.people SET weight = NULL WHERE weight = '';
UPDATE lahmansdb_to_clean.people SET height = NULL WHERE height = '';
UPDATE lahmansdb_to_clean.people SET bats = NULL WHERE bats = '';
UPDATE lahmansdb_to_clean.people SET throws = NULL WHERE throws = '';
UPDATE lahmansdb_to_clean.people SET debut = NULL WHERE debut = '';
UPDATE lahmansdb_to_clean.people SET finalGame = NULL WHERE finalGame = '';
UPDATE lahmansdb_to_clean.people SET retroID = NULL WHERE retroID = '';
UPDATE lahmansdb_to_clean.people SET bbrefID = NULL WHERE bbrefID = '';
~~~~~
```

Tests

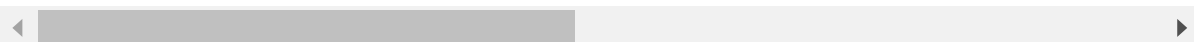
In [7]:

```
##sql SELECT * FROM people WHERE birthState = ""
conn=pymysql.connect(host='localhost',port=int(3306),user='root',passwd='admin123', db = 'l
pd.read_sql_query("SELECT * FROM people WHERE birthState = '' ;", conn)
```

Out[7]:

playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	death
----------	-----------	------------	----------	--------------	------------	-----------	-----------	-------

0 rows × 26 columns



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

Explanation

Different datatypes are present in SQL to ensure we can impose the right restrictions on different columns in our database. For eg: Day, Month, Year cannot be strings and will always be numbers / NULL values. Thus converting them to INT maintains data sanity. Also, IDs can be alphanumeric for which a VARCHAR datatype makes sense.

Change

Used the alter table statement to pick each column and assign them a suitable datatype.

SQL

```
alter table people modify birthYear INT null;
alter table people modify birthMonth INT null;
alter table people modify birthDate INT null;

alter table people modify birthCountry VARCHAR(40) null;

alter table people modify birthState VARCHAR(40) null;

alter table people modify birthCity VARCHAR(40) null;

alter table people modify deathYear INT null;

alter table people modify deathMonth INT null;

alter table people modify deathCountry VARCHAR(40) null;

alter table people modify deathState VARCHAR(40) null;

alter table people modify deathCity VARCHAR(40) null;

alter table people modify nameFirst VARCHAR(40) null;

alter table people modify nameLast VARCHAR(15) null;

alter table people modify nameGiven VARCHAR(500) null;

alter table people modify weight INT null;

alter table people modify height INT null;

alter table people modify bats VARCHAR(15) null;

alter table people modify throws VARCHAR(10) null;

alter table people modify debut VARCHAR(10) null;

alter table people modify finalGame VARCHAR(10) null;
```

```
alter table people modify retroID VARCHAR(10) null;
```

```
alter table people modify bbrefID VARCHAR(10) null;
```

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

I use the simple logic when deathDay exists, it means the person is dead and thus and 'N' when deathDate does not.

Change

We first create a column named 'isDead' using the Alter table command. Then, UPDATE command is used to set the value using conditions from CASE statement.

SQL

```
ALTER TABLE people ADD COLUMN isDead VARCHAR(5) NULL;
```

```
UPDATE people
SET people.isDead = CASE
    WHEN deathDay IS NOT NULL THEN 'Y'
    ELSE 'N'
END
WHERE people.isDead IS NULL;
```

Tests

In [8]:

```
conn=pymysql.connect(host='localhost',port=int(3306),user='root',passwd='admin123', db = 'l
pd.read_sql_query("SELECT * FROM people WHERE isDead = 'N' limit 10 ;", conn)
#%sql SELECT * FROM people WHERE isDead = "N" limit 10
```

Out[8]:

rthState	birthCity	deathYear	deathMonth	deathDay	...	weight	height	bats	throws	debut
CO	Denver	None	None	None	...	215	75	R	R	2004-04-06
AL	Mobile	None	None	None	...	180	72	R	R	1954-04-13
CA	Orange	None	None	None	...	190	75	R	R	1977-07-26
FL	Palm Beach	None	None	None	...	184	73	L	L	2001-09-10
La Romana	La Romana	None	None	None	...	235	74	L	L	2010-07-28
AR	Little Rock	None	None	None	...	200	78	R	R	1973-07-29
GA	Atlanta	None	None	None	...	190	74	R	L	1997-06-10
MI	Flint	None	None	None	...	200	75	L	L	1989-04-08
OH	Zanesville	None	None	None	...	180	71	R	R	1993-09-07
MA	Newburyport	None	None	None	...	200	76	L	L	1991-09-10

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

A birthDate and deathDate column can be created using the individual day, month and year columns in the appropriate format. A single column makes it easier for data visualization and using it for different queries

Change

birthDate and deathDate are created by using the STR_TO_DATE function which concatenates the day, month and year column in the given appropriate format.

SQL

```
ALTER TABLE people ADD COLUMN deathDate DATETIME NULL;
ALTER TABLE people ADD COLUMN birthDate DATETIME NULL;
```

```
UPDATE people
SET people.birthDate = STR_TO_DATE(CONCAT(birthYear, '-', LPAD(birthMonth,2,0), '-',
birthDay),"%Y-%m-%d")
WHERE people.birthDate IS NULL;
```

```
UPDATE people
SET people.deathDate = STR_TO_DATE(CONCAT(deathYear, '-', LPAD(deathMonth,2,0), '-',
deathDay),"%Y-%m-%d")
WHERE people.deathDate IS NULL;
```

Tests

In [13]:

```
##reload_ext sql
##sql select birthDate FROM people WHERE birthDate >= '2000-01-01' ORDER BY birthDate ASC L
conn=pymysql.connect(host='localhost',port=int(3306),user='root',passwd='admin123', db = 'l
pd.read_sql_query("SELECT deathDate FROM people WHERE deathDate >= '2000-01-01' ORDER BY de
```

Out[13]:

	deathDate
0	2000-01-01
1	2000-01-04
2	2000-01-11
3	2000-01-19
4	2000-01-19
5	2000-01-20
6	2000-01-26
7	2000-01-26
8	2000-01-28
9	2000-02-03

In [10]:

```
pd.read_sql_query("SELECT birthDate FROM people WHERE birthDate <= '1965-01-01' ORDER BY bi
#%sql SELECT birthDate FROM people WHERE birthDate <= '1965-01-01' ORDER BY birthDate ASC L
```

Out[10]:

	birthDate
0	1820-04-17
1	1824-10-05
2	1832-09-17
3	1832-10-23
4	1835-01-10
5	1836-02-29
6	1837-12-26
7	1838-03-10
8	1838-07-16
9	1838-08-27

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 people
(
    playerID      varchar(15) not null
        primary key,
    birthYear     int          null,
    birthMonth    int          null,
    birthDay      int          null,
    birthCountry  varchar(15)  null,
    birthState    varchar(40)  null,
    birthCity     varchar(40)  null,
    deathYear     int          null,
    deathMonth    int          null,
    deathDay      int          null,
    deathCountry  varchar(15)  null,
    deathState    varchar(40)  null,
    deathCity     varchar(40)  null,
```

```
nameFirst    varchar(40) null,  
nameLast     varchar(15) null,  
nameGiven    varchar(500) null,  
weight       int null,  
height       int null,  
bats         varchar(15) null,  
throws       varchar(10) null,  
debut        varchar(10) null,  
finalGame    varchar(10) null,  
retroID      varchar(10) null,  
bbrefID      varchar(10) null,  
isDead       varchar(5) null,  
birthDate    datetime null,  
deathDate    datetime null  
);
```

Batting Table

1) Convert all empty strings to NULL

Explanation

Empty values are converted to NULL as NULL values are more efficiently stored and empty strings ("") take up 2 CHAR space. Also, it provides convenience while converting between different datatypes. For eg: converting the string column with "" to INT will raise errors. Also, most functions raise error on NULL values which makes it easier to identify invalid data points whereas empty strings indicate a string is present however it is empty which can sometimes pass as a valid record.

Change

I set the the value of a column as NULL when the row contains a NULL value using the UPDATE statement

SQL


```

UPDATE lahmansdb_to_clean.batting SET playerID=NULL WHERE playerID='';
UPDATE lahmansdb_to_clean.batting SET yearID=NULL WHERE yearID='';
UPDATE lahmansdb_to_clean.batting SET stint =NULL WHERE stint ='' ;
UPDATE lahmansdb_to_clean.batting SET teamID=NULL WHERE teamID='';
UPDATE lahmansdb_to_clean.batting SET lgID=NULL WHERE lgID='' ;
UPDATE lahmansdb_to_clean.batting SET G =NULL WHERE G ='';
UPDATE lahmansdb_to_clean.batting SET AB=NULL WHERE AB='';
UPDATE lahmansdb_to_clean.batting SET R=NULL WHERE R='';
UPDATE lahmansdb_to_clean.batting SET H=NULL WHERE H='';
UPDATE lahmansdb_to_clean.batting SET `2B`=NULL WHERE `2B`='';
UPDATE lahmansdb_to_clean.batting SET `3B`=NULL WHERE `3B`='';
UPDATE lahmansdb_to_clean.batting SET HR=NULL WHERE HR='';
UPDATE lahmansdb_to_clean.batting SET RBI=NULL WHERE RBI='';
UPDATE lahmansdb_to_clean.batting SET SB=NULL WHERE SB='';
UPDATE lahmansdb_to_clean.batting SET CS=NULL WHERE CS='';
UPDATE lahmansdb_to_clean.batting SET BB=NULL WHERE BB='';
UPDATE lahmansdb_to_clean.batting SET SO=NULL WHERE SO='';
UPDATE lahmansdb_to_clean.batting SET IBB=NULL WHERE IBB='';
UPDATE lahmansdb_to_clean.batting SET HBP=NULL WHERE HBP='';
UPDATE lahmansdb_to_clean.batting SET SH=NULL WHERE SH='';
UPDATE lahmansdb_to_clean.batting SET SF=NULL WHERE SF='';
UPDATE lahmansdb_to_clean.batting SET GIDP =NULL WHERE GIDP ='';

```

Tests

In [15]:

```

conn=pymysql.connect(host='localhost',port=int(3306),user='root',passwd='admin123', db = 'l
pd.read_sql_query("SELECT count(*) FROM lahmansdb_to_clean.batting where RBI is NULL;", con
#%sql SELECT count(*) FROM Lahmansdb_to_clean.batting where RBI is NULL;

```

Out[15]:

	count(*)
0	756

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

Explanation

Put your answer in this cell

Change

I use the alter table statement to change the datatype of the column. For eg: yearID contains only integer values hence it is converted to small int. Similarly we observe, playerID contains alphanumeric values so it uses VARCHAR. Other columns like G, AB etc. only contain small integer values hence small int is given.

SQL

```
alter table batting modify playerID VARCHAR(9) not null;

alter table batting modify yearID smallint null;

alter table batting modify stint smallint null;

alter table batting modify teamID VARCHAR(3) null;

alter table batting modify lgID char(2) null;

alter table batting modify G smallint null;

alter table batting modify AB smallint null;

alter table batting modify R smallint null;

alter table batting modify H smallint null;

alter table batting modify `2B` smallint null;

alter table batting modify `3B` smallint null;

alter table batting modify HR smallint null;

alter table batting modify RBI smallint null;

alter table batting modify SB smallint null;

alter table batting modify CS smallint null;

alter table batting modify BB smallint null;

alter table batting modify SO smallint null;

alter table batting modify IBB smallint null;

alter table batting modify HBP smallint null;

alter table batting modify SH smallint null;

alter table batting modify SF smallint null;

alter table batting modify GIDP smallint null;
```

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 use a composite key of (playerID, yearID and stint) to make a Primary Key.

Explanation

It is used because it is sufficient to query information in many scenarios. In a case we frequently use team information as well along with the above 3 values, we can add it to the key. The 3 columns can help to uniquely identify a player information with reasonable granularity.

SQL

```
alter table batting
  add constraint batting_pk
    primary key (playerID, yearID, stint);
```

Test

In [18]:

```
conn=pymysql.connect(host='localhost',port=int(3306),user='root',passwd='admin123', db = 'baseball')
pd.read_sql_query("SHOW KEYS FROM batting WHERE Key_name = 'PRIMARY';", conn)
#%sql SHOW KEYS FROM batting WHERE Key_name = 'PRIMARY' and Column_name = 'playerID'
```

Out[18]:

	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part
0	batting	0	PRIMARY	1	playerID	A	20328	None
1	batting	0	PRIMARY	2	yearID	A	98762	None
2	batting	0	PRIMARY	3	stint	A	107474	None

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

When adding a foreign key, we should ensure that entries in the child and parent table match. We should delete these values before creating a foreign relationship.

Change

I use the alter table expression to add foreign key constraint to playerID in the batting table related to playerID in people.

SQL

```
alter table batting
  add constraint batting_people_playerID_fk
    foreign key (playerID) references people (playerID);
```

Tests

In [19]:

```
conn=pymysql.connect(host='localhost',port=int(3306),user='root',passwd='admin123', db = '1
pd.read_sql_query("Select playerID from batting where playerID not in (select playerID from
#%sql Select playerID from batting where playerID not in (select playerID from people);
```

Out[19]:

playerID

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 batting
(
  playerID varchar(9) not null,
  yearID    smallint    not null,
  stint     smallint    not null,
  teamID    varchar(3) null,
  lgID      char(2)     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      text        null,
        IBB     text        null,
        HBP     text        null,
        SH      text        null,
        SF      text        null,
        GIDP    text        null,
        primary key (playerID, yearID, stint),
        constraint batting_people_playerID_fk
            foreign key (playerID) references people (playerID)
    );

```

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.

In [29]:

```

%reload_ext sql
%sql mysql+pymysql://root:admin123@localhost/lahmansbaseballdb

```

Question 0

What is the average salary in baseball history?

In [25]:

```
pd.read_sql_query("select FLOOR(AVG(salary)) AS average FROM lahmansbaseballdb.salaries;",
```

Out[25]:

	average
0	2085634.0

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

In [29]:

```
conn=pymysql.connect(host='localhost',port=int(3306),user='root',passwd='admin123', db = 'l
pd.read_sql_query("select P1.playerID, P2.playerID, P2.schoolID, P2.yearID, nameFirst, name
```

Out[29]:

	playerID	playerID	schoolID	yearID	nameFirst	nameLast	birthState
0	barnesa01	barnesa01	auburn	1918	Sam	Barnes	AL
1	barnesa01	barnesa01	auburn	1919	Sam	Barnes	AL
2	barnesa01	barnesa01	auburn	1920	Sam	Barnes	AL
3	barnesa01	barnesa01	auburn	1921	Sam	Barnes	AL
4	bowensa01	bowensa01	tennst	1956	Sam	Bowens	NC
5	bowensa01	bowensa01	tennst	1957	Sam	Bowens	NC
6	bowensa01	bowensa01	tennst	1958	Sam	Bowens	NC
7	bowensa02	bowensa02	gacoast	1971	Sam	Bowen	GA
8	bowensa02	bowensa02	gacoast	1972	Sam	Bowen	GA
9	brownsa01	brownsa01	grovecity	1899	Sam	Brown	PA

In [3]:

%%sql

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

Out[3]:

nameFirst	nameLast	playerID	schoolID	yearID	birthState
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

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';
```

In [30]:

```
conn=pymysql.connect(host='localhost',port=int(3306),user='root',passwd='admin123', db = 'l
pd.read_sql_query("select * FROM schools where name_full = 'Columbia University in the City
```

Out[30]:

	schoolID	name_full	city	state	country
0	columbia	Columbia University in the City of New York	New York	NY	USA

In [6]:

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

Out[6]:

	schoolID	name_full	city	state	country
	columbia	Columbia University in the City of New York	New York	NY	USA

Part E: CSVDataTable

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 is 'playerID' and fieldlist is ['collied01']

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

When inserting a new record the following should be checked:

- The PK specified should not already exist in the table

- The value of the PK to be inserted should not be empty or NULL

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

- If the primary key is modified, the new value should not already exist in the table
- The column names given to be modified should all exist in the table

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

iii. Testing

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

```
===== test session starts ===== platform win32
-- Python 3.7.3, pytest-4.3.1, py-1.8.0, pluggy-0.9.0 -- C:\Users\tushar\Anaconda3\python.exe cachedir:
.pytest_cache rootdir: C:\PersonalFiles\Courses\IntroToDB\HW1\W4111_HW1 F21-2\W4111_HW1
F21\4111_s21_hw1_programming_CSVDataTable\tests, inifile: plugins: remotedata-0.3.1, openfiles-0.3.2,
doctestplus-0.3.0, arraydiff-0.3, anyio-3.3.0 collecting ... collected 1 item
```

```
csv_table_tests.py::tests_people PASSED [100%]
```

```
find_by_primary_key(): Known Record
```

```
OrderedDict([('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
```

```
True
```


find_by_template(): Known Template

```
[OrderedDict([('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')])]
```

delete_by_key(): Known record

Before DELETE operation 19619

Num Rows deleted 1

After DELETE operation 19618

delete_by_key(): Unknown record

Before DELETE operation 19618

Num Rows deleted 0

After DELETE operation 19618

delete_by_template(): Known Template

Num rows deleted 1

update_by_key(): Known record

Before UPDATE operation

```
[OrderedDict([('playerID', 'abercda01'), ('birthYear', '1850'), ('birthMonth', '1'), ('birthDay', '2'), ('birthCountry', 'USA'), ('birthState', 'OK'), ('birthCity', 'Fort Towson'), ('deathYear', '1939'), ('deathMonth', '11'), ('deathDay', '11'), ('deathCountry', 'USA'), ('deathState', 'PA'), ('deathCity', 'Philadelphia'), ('nameFirst', 'Frank'), ('nameLast', 'Abercrombie'), ('nameGiven', 'Francis Patterson'), ('weight', ''), ('height', ''), ('bats', ''), ('throws', ''), ('debut', '1871-10-21'), ('finalGame', '1871-10-21'), ('retroID', 'aberd101'), ('bbrefID', 'abercda01')])]
```

Num Rows updated 1

update_by_key(): Unknown record

Searched key does not exist

Num Rows updated 0

update_by_template(): Known record

Before UPDATE operation

```
[OrderedDict([('playerID', 'abbotje01'), ('birthYear', '1972'), ('birthMonth', '8'), ('birthDay', '17'), ('birthCountry', 'USA'), ('birthState', 'GA'), ('birthCity', 'Atlanta'), ('deathYear', ''), ('deathMonth', ''), ('deathDay', ''), ('deathCountry', ''), ('deathState', ''), ('deathCity', ''), ('nameFirst', 'Jeff'), ('nameLast', 'Abbott'), ('nameGiven', 'Jeffrey William'), ('weight', '190'), ('height', '74'), ('bats', 'R'), ('throws', 'L'), ('debut', '1997-06-10'), ('finalGame', '2001-09-29'), ('retroID', 'abboj002'), ('bbrefID', 'abbotje01')])]
```

Num Rows updated 1

After UPDATE operation

```
OrderedDict([('playerID', 'abbotje01'), ('birthYear', '1985'), ('birthMonth', '6'), ('birthDay', '15'), ('birthCountry', 'USA'), ('birthState', 'GA'), ('birthCity', 'Atlanta'), ('deathYear', ''), ('deathMonth', ''), ('deathDay', ''), ('deathCountry', ''), ('deathState', ''), ('deathCity', ''), ('nameFirst', 'Jeff'), ('nameLast', 'Abbott'), ('nameGiven', 'Jeffrey William'), ('weight', '190'), ('height', '74'), ('bats', 'R'), ('throws', 'L'), ('debut', '1997-06-10'), ('finalGame', '2001-09-29'), ('retroID', 'abboj002'), ('bbrefID', 'abbotje01')]))
```

update_by_template(): Wrong field name in new_values()

Before UPDATE operation

```
[OrderedDict([('playerID', 'allenet01'), ('birthYear', '1904'), ('birthMonth', '1'), ('birthDay', '1'), ('birthCountry', 'USA'), ('birthState', 'OH'), ('birthCity', 'Cincinnati'), ('deathYear', '1993'), ('deathMonth', '9'), ('deathDay', '15'), ('deathCountry', 'USA'), ('deathState', 'CA'), ('deathCity', 'Crescent City'), ('nameFirst', 'Ethan'), ('nameLast', 'Allen'), ('nameGiven', 'Ethan Nathan'), ('weight', '180'), ('height', '73'), ('bats', 'R'), ('throws', 'R'), ('debut', '1926-06-13'), ('finalGame', '1938-06-18'), ('retroID', 'allee101'), ('bbrefID', 'allenet01')]))]
```

Error: Field to be modified does not exist

Num Rows updated 0

update_by_key(): Modified value already exists in Primary Key

Before UPDATE operation

```
[OrderedDict([('playerID', 'abercda01'), ('birthYear', '1980'), ('birthMonth', '2'), ('birthDay', '14'), ('birthCountry', 'USA'), ('birthState', 'OK'), ('birthCity', 'Fort Towson'), ('deathYear', '1939'), ('deathMonth', '11'), ('deathDay', '11'), ('deathCountry', 'USA'), ('deathState', 'PA'), ('deathCity', 'Philadelphia'), ('nameFirst', 'Frank'), ('nameLast', 'Abercrombie'), ('nameGiven', 'Francis Patterson'), ('weight', ''), ('height', ''), ('bats', ''), ('throws', ''), ('debut', '1871-10-21'), ('finalGame', '1871-10-21'), ('retroID', 'aberd101'), ('bbrefID', 'abercda01')]))]
```

Error: Value key is modified to already exists in table

Num Rows updated 0

insert(): Unique Primary key

Before INSERT operation

Num Rows 19617

After INSERT operation

Num Rows 19618

insert(): Duplicate Primary key

Before INSERT operation

Num Rows 19618

Duplicate primary key found

After INSERT operation

Num Rows 19618

insert(): No Primary key specified

Before INSERT operation

Num Rows 19618

Specify primary key to insert new record

After INSERT operation

Num Rows 19618

```
===== test session starts ===== platform win32
-- Python 3.7.3, pytest-4.3.1, py-1.8.0, pluggy-0.9.0 -- C:\Users\tushar\Anaconda3\python.exe cachedir:
.pytest_cache rootdir: C:\PersonalFiles\Courses\IntroToDB\HW1\W4111_HW1 F21-2\W4111_HW1
F21\4111_s21_hw1_programming_CSVDDataTable\tests, inifile: plugins: remotedata-0.3.1, openfiles-0.3.2,
doctestplus-0.3.0, arraydiff-0.3, anyio-3.3.0 collecting ... collected 1 item
```

csv_table_tests.py::tests_batting PASSED [100%]

find_by_primary_key(): Known Record

```
OrderedDict([('playerID', 'aardsda01'), ('yearID', '2004'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'),
('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')])
```

find_by_primary_key(): Unknown Record

Records found None

find_by_template(): Known Template

```
[OrderedDict([('playerID', 'runzlda01'), ('yearID', '2009'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'),
('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')]), OrderedDict([('playerID', 'niekrla01'), ('yearID', '2007'),
('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB', '17'), ('R', '0'), ('H', '3'), ('2B', '0'), ('3B', '0'), ('HR', '0'),
('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '5'), ('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '1')]),
OrderedDict([('playerID', 'torcato01'), ('yearID', '2005'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB',
'11'), ('R', '1'), ('H', '3'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '2'), ('IBB',
'0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '2')]), OrderedDict([('playerID', 'aardsda01'), ('yearID', '2004'),
('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('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')]),
OrderedDict([('playerID', 'castial01'), ('yearID', '2003'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB',
'15'), ('R', '2'), ('H', '3'), ('2B', '1'), ('3B', '0'), ('HR', '1'), ('RBI', '4'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '5'), ('IBB',
'0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0')]), OrderedDict([('playerID', 'murraca01'), ('yearID', '2002'),
('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB', '12'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'),
('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '2'), ('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0')]),
OrderedDict([('playerID', 'brohatr01'), ('yearID', '2002'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('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')]), OrderedDict([('playerID', 'schmija01'), ('yearID', '2001'), ('stint',
'2'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB', '26'), ('R', '3'), ('H', '4'), ('2B', '0'), ('3B', '0'), ('HR', '1'), ('RBI',
'3'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '12'), ('IBB', '0'), ('HBP', '0'), ('SH', '1'), ('SF', '0'), ('GIDP', '0')]),
OrderedDict([('playerID', 'meloju01'), ('yearID', '2000'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB',
'13'), ('R', '0'), ('H', '1'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '1'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '5'), ('IBB',
'0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0')]), OrderedDict([('playerID', 'hernali01'), ('yearID', '1999'), ('stint',
'2'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB', '18'), ('R', '1'), ('H', '4'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI',
'1'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '5'), ('IBB', '0'), ('HBP', '1'), ('SH', '6'), ('SF', '0'), ('GIDP', '0')]),
OrderedDict([('playerID', 'foulkke01'), ('yearID', '1997'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB',
'13'), ('R', '0'), ('H', '2'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '4'), ('IBB',
```

localhost:8888/notebooks/4111 s21 hw1 programming.ipynb

```
(('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '2'), ('SO', '2'), ('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0'))], OrderedDict([('playerID', 'hendeke01'), ('yearID', '1966'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB', '29'), ('R', '4'), ('H', '9'), ('2B', '1'), ('3B', '1'), ('HR', '1'), ('RBI', '1'), ('SB', '0'), ('CS', '0'), ('BB', '2'), ('SO', '3'), ('IBB', '0'), ('HBP', '1'), ('SH', '0'), ('SF', '0'), ('GIDP', '0'))], OrderedDict([('playerID', 'staneal01'), ('yearID', '1963'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB', '1'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '1'), ('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0'))], OrderedDict([('playerID', 'maricju01'), ('yearID', '1960'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB', '31'), ('R', '3'), ('H', '4'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '1'), ('SB', '0'), ('CS', '1'), ('BB', '3'), ('SO', '7'), ('IBB', '0'), ('HBP', '1'), ('SH', '1'), ('SF', '0'), ('GIDP', '1'))], OrderedDict([('playerID', 'byerlbu01'), ('yearID', '1959'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('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'))], OrderedDict([('playerID', 'jonesgo01'), ('yearID', '1958'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'NL'), ('G', '11'), ('AB', '7'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '2'), ('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0')]))]
```

delete_by_key(): Known record

Before DELETE operation 105861

Num Rows deleted 1

After DELETE operation 105860

delete_by_key(): Unknown record

Before DELETE operation 105860

Num Rows deleted 0

After DELETE operation 105860

delete_by_template(): Known Template

Num rows deleted 36

update_by_key(): Known record

Before UPDATE operation

```
OrderedDict([('playerID', 'aardsda01'), ('yearID', '2006'), ('stint', '1'), ('teamID', 'CHN'), ('lgID', 'NL'), ('G', '45'), ('AB', '2'), ('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', '1'), ('SF', '0'), ('GIDP', '0')]))]
```

Num Rows updated 1

After Update operation

```
OrderedDict([('playerID', 'aardsda01'), ('yearID', '2006'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'AL'), ('G', '20'), ('AB', '2'), ('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', '1'), ('SF', '0'), ('GIDP', '0')]))]
```

update_by_key(): Unknown record

Searched key does not exist

Num Rows updated 0

update_by_template(): Known record

[illegible]

```

OrderedDict([('playerID', 'bradfbu01'), ('yearID', '1975'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'),
('AB', '58'), ('R', '8'), ('H', '9'), ('2B', '3'), ('3B', '1'), ('HR', '2'), ('RBI', '15'), ('SB', '3'), ('CS', '2'), ('BB', '8'), ('SO', '22'),
('IBB', '0'), ('HBP', '3'), ('SH', '1'), ('SF', '4'), ('GIDP', '1')]), OrderedDict([('playerID', 'carlocci01'), ('yearID', '1969'),
('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '10'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'),
('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '4'), ('IBB', '0'), ('HBP', '0'), ('SH', '2'), ('SF', '0'), ('GIDP', '0')]),
OrderedDict([('playerID', 'mcmahdo02'), ('yearID', '1968'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'),
('AB', '3'), ('R', '0'), ('H', '1'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '1'),
('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0')]), OrderedDict([('playerID', 'johnto01'), ('yearID', '1968'),
('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '62'), ('R', '3'), ('H', '12'), ('2B', '1'), ('3B', '0'), ('HR', '1'),
('RBI', '9'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '15'), ('IBB', '0'), ('HBP', '0'), ('SH', '5'), ('SF', '0'), ('GIDP', '0')]),
OrderedDict([('playerID', 'joycemi01'), ('yearID', '1962'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'),
('AB', '7'), ('R', '1'), ('H', '3'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '2'),
('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0')]), OrderedDict([('playerID', 'larsed01'), ('yearID', '1961'),
('stint', '2'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '25'), ('R', '2'), ('H', '8'), ('2B', '0'), ('3B', '0'), ('HR', '1'),
('RBI', '4'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '5'), ('IBB', '0'), ('HBP', '0'), ('SH', '1'), ('SF', '0'), ('GIDP', '0')]),
OrderedDict([('playerID', 'byrdha01'), ('yearID', '1955'), ('stint', '2'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB',
'30'), ('R', '2'), ('H', '2'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '7'), ('IBB',
'0'), ('HBP', '0'), ('SH', '3'), ('SF', '0'), ('GIDP', '0')]), OrderedDict([('playerID', 'alomalu01'), ('yearID', '1952'), ('stint',
'1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '7'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'),
('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '2'), ('IBB', ''), ('HBP', '0'), ('SH', '0'), ('SF', ''), ('GIDP', '0')]),
OrderedDict([('playerID', 'alomalu01'), ('yearID', '1951'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'),
('AB', '20'), ('R', '2'), ('H', '7'), ('2B', '1'), ('3B', '0'), ('HR', '0'), ('RBI', '2'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '5'),
('IBB', ''), ('HBP', '0'), ('SH', '1'), ('SF', ''), ('GIDP', '0')]), OrderedDict([('playerID', 'gillebo01'), ('yearID', '1948'),
('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '16'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'),
('RBI', '1'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '10'), ('IBB', ''), ('HBP', '0'), ('SH', '1'), ('SF', ''), ('GIDP', '0')]),
OrderedDict([('playerID', 'caldwea01'), ('yearID', '1948'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'),
('AB', '5'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '1'), ('SB', '0'), ('CS', '0'), ('BB', '2'), ('SO', '3'),
('IBB', ''), ('HBP', '0'), ('SH', '0'), ('SF', ''), ('GIDP', '0')]), OrderedDict([('playerID', 'groveor01'), ('yearID', '1947'),
('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '48'), ('R', '2'), ('H', '7'), ('2B', '2'), ('3B', '0'), ('HR', '0'),
('RBI', '1'), ('SB', '0'), ('CS', '0'), ('BB', '5'), ('SO', '16'), ('IBB', ''), ('HBP', '0'), ('SH', '1'), ('SF', ''), ('GIDP', '1')]),
OrderedDict([('playerID', 'gillebo01'), ('yearID', '1947'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB',
'33'), ('R', '2'), ('H', '2'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '3'), ('SO', '12'), ('IBB',
''), ('HBP', '0'), ('SH', '0'), ('SF', ''), ('GIDP', '2')]), OrderedDict([('playerID', 'hamnera01'), ('yearID', '1946'), ('stint',
'1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '18'), ('R', '1'), ('H', '3'), ('2B', '1'), ('3B', '0'), ('HR', '0'), ('RBI',
'2'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '7'), ('IBB', ''), ('HBP', '0'), ('SH', '2'), ('SF', ''), ('GIDP', '1')]),
OrderedDict([('playerID', 'smithed04'), ('yearID', '1943'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'),
('AB', '69'), ('R', '4'), ('H', '11'), ('2B', '1'), ('3B', '0'), ('HR', '1'), ('RBI', '8'), ('SB', '0'), ('CS', '0'), ('BB', '2'), ('SO', '9'),
('IBB', ''), ('HBP', '0'), ('SH', '4'), ('SF', ''), ('GIDP', '2')]), OrderedDict([('playerID', 'knottja01'), ('yearID', '1940'),
('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '57'), ('R', '2'), ('H', '5'), ('2B', '1'), ('3B', '0'), ('HR', '0'),
('RBI', '4'), ('SB', '0'), ('CS', '0'), ('BB', '5'), ('SO', '17'), ('IBB', ''), ('HBP', '0'), ('SH', '4'), ('SF', ''), ('GIDP', '0')]),
OrderedDict([('playerID', 'applepe01'), ('yearID', '1940'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'),
('AB', '17'), ('R', '2'), ('H', '3'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '5'),
('IBB', ''), ('HBP', '0'), ('SH', '1'), ('SF', ''), ('GIDP', '0')]), OrderedDict([('playerID', 'knottja01'), ('yearID', '1939'),
('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '53'), ('R', '3'), ('H', '8'), ('2B', '0'), ('3B', '0'), ('HR', '0'),
('RBI', '1'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '18'), ('IBB', ''), ('HBP', '1'), ('SH', '5'), ('SF', ''), ('GIDP', '2')]),
OrderedDict([('playerID', 'dietrbi01'), ('yearID', '1939'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB',
'37'), ('R', '5'), ('H', '8'), ('2B', '3'), ('3B', '0'), ('HR', '1'), ('RBI', '2'), ('SB', '0'), ('CS', '0'), ('BB', '4'), ('SO', '12'), ('IBB',
''), ('HBP', '0'), ('SH', '2'), ('SF', ''), ('GIDP', '0')]), OrderedDict([('playerID', 'sheame01'), ('yearID', '1937'), ('stint',
'1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '71'), ('R', '7'), ('H', '15'), ('2B', '1'), ('3B', '0'), ('HR', '0'), ('RBI',
'5'), ('SB', '1'), ('CS', '0'), ('BB', '15'), ('SO', '10'), ('IBB', ''), ('HBP', '0'), ('SH', '0'), ('SF', ''), ('GIDP', '')]),
OrderedDict([('playerID', 'riddljo02'), ('yearID', '1930'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB',
'58'), ('R', '7'), ('H', '14'), ('2B', '3'), ('3B', '1'), ('HR', '0'), ('RBI', '4'), ('SB', '0'), ('CS', '0'), ('BB', '3'), ('SO', '6'), ('IBB',

```

```
"", ('HBP', '1'), ('SH', '2'), ('SF', ''), ('GIDP', ''))], OrderedDict([('playerID', 'walshed02'), ('yearID', '1929'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '43'), ('R', '5'), ('H', '10'), ('2B', '2'), ('3B', '1'), ('HR', '0'), ('RBI', '6'), ('SB', '0'), ('CS', '0'), ('BB', '3'), ('SO', '9'), ('IBB', ''), ('HBP', '1'), ('SH', '1'), ('SF', ''), ('GIDP', ''))], OrderedDict([('playerID', 'jacobel01'), ('yearID', '1927'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '20'), ('R', '4'), ('H', '3'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '2'), ('SB', '0'), ('CS', '1'), ('BB', '1'), ('SO', '5'), ('IBB', ''), ('HBP', '0'), ('SH', '4'), ('SF', ''), ('GIDP', ''))], OrderedDict([('playerID', 'blankte01'), ('yearID', '1924'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '46'), ('R', '10'), ('H', '15'), ('2B', '2'), ('3B', '1'), ('HR', '1'), ('RBI', '5'), ('SB', '0'), ('CS', '0'), ('BB', '4'), ('SO', '6'), ('IBB', ''), ('HBP', '1'), ('SH', '3'), ('SF', ''), ('GIDP', ''))], OrderedDict([('playerID', 'faberre01'), ('yearID', '1919'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '54'), ('R', '8'), ('H', '10'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '4'), ('SB', '0'), ('CS', ''), ('BB', '6'), ('SO', '20'), ('IBB', ''), ('HBP', '0'), ('SH', '3'), ('SF', ''), ('GIDP', ''))], OrderedDict([('playerID', 'olmstfr01'), ('yearID', '1911'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '37'), ('R', '3'), ('H', '7'), ('2B', '1'), ('3B', '1'), ('HR', '0'), ('RBI', '4'), ('SB', '0'), ('CS', ''), ('BB', '1'), ('SO', ''), ('IBB', ''), ('HBP', '0'), ('SH', '1'), ('SF', ''), ('GIDP', ''))], OrderedDict([('playerID', 'owenfr01'), ('yearID', '1908'), ('stint', '1'), ('teamID', 'CHA'), ('lgID', 'AL'), ('G', '25'), ('AB', '50'), ('R', '3'), ('H', '9'), ('2B', '3'), ('3B', '0'), ('HR', '0'), ('RBI', '4'), ('SB', '1'), ('CS', ''), ('BB', '3'), ('SO', '15'), ('IBB', ''), ('HBP', '0'), ('SH', '1'), ('SF', ''), ('GIDP', ''))])]
```

Num Rows updated 47

After UPDATE operation

```
[OrderedDict([('playerID', 'putnaza01'), ('yearID', '2016'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('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'))], OrderedDict([('playerID', 'beckch02'), ('yearID', '2016'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('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'))], OrderedDict([('playerID', 'quintjo01'), ('yearID', '2012'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB', '3'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '3'), ('IBB', '0'), ('HBP', '0'), ('SH', '1'), ('SF', '0'), ('GIDP', '0'))], OrderedDict([('playerID', 'willira01'), ('yearID', '2009'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('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'))], OrderedDict([('playerID', 'aardsda01'), ('yearID', '2007'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('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'))], OrderedDict([('playerID', 'macdomi01'), ('yearID', '2006'), ('stint', '2'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('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'))], OrderedDict([('playerID', 'snydejo02'), ('yearID', '1999'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('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'))], OrderedDict([('playerID', 'castito02'), ('yearID', '1998'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('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'))], OrderedDict([('playerID', 'levinal01'), ('yearID', '1997'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('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'))], OrderedDict([('playerID', 'mcdowja01'), ('yearID', '1994'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('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'))], OrderedDict([('playerID', 'thigpbo01'), ('yearID', '1993'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('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'))], OrderedDict([('playerID', 'fiskca01'), ('yearID', '1993'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB', '53'), ('R', '2'), ('H', '10'), ('2B', '0'), ('3B', '0'), ('HR', '1'), ('RBI', '4'), ('SB', '0'), ('CS', '1'), ('BB', '2'), ('SO', '11'), ('IBB', '0'), ('HBP', '1'), ('SH', '1'), ('SF', '1'), ('GIDP', '0'))], OrderedDict([('playerID', 'kinger01'), ('yearID', '1990'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB', '0'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '0'), ('IBB',
```


33/36

```

'2'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '7'), ('IBB', ''), ('HBP', '0'), ('SH', '2'), ('SF', ''), ('GIDP', '1'))],
OrderedDict([('playerID', 'smithed04'), ('yearID', '1943'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'),
('AB', '69'), ('R', '4'), ('H', '11'), ('2B', '1'), ('3B', '0'), ('HR', '1'), ('RBI', '8'), ('SB', '0'), ('CS', '0'), ('BB', '2'), ('SO', '9'),
('IBB', ''), ('HBP', '0'), ('SH', '4'), ('SF', ''), ('GIDP', '2'))], OrderedDict([('playerID', 'knottja01'), ('yearID', '1940'),
('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB', '57'), ('R', '2'), ('H', '5'), ('2B', '1'), ('3B', '0'), ('HR', '0'),
('RBI', '4'), ('SB', '0'), ('CS', '0'), ('BB', '5'), ('SO', '17'), ('IBB', ''), ('HBP', '0'), ('SH', '4'), ('SF', ''), ('GIDP', '0'))],
OrderedDict([('playerID', 'applepe01'), ('yearID', '1940'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'),
('AB', '17'), ('R', '2'), ('H', '3'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '5'),
('IBB', ''), ('HBP', '0'), ('SH', '1'), ('SF', ''), ('GIDP', '0'))], OrderedDict([('playerID', 'knottja01'), ('yearID', '1939'),
('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB', '53'), ('R', '3'), ('H', '8'), ('2B', '0'), ('3B', '0'), ('HR', '0'),
('RBI', '1'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '18'), ('IBB', ''), ('HBP', '1'), ('SH', '5'), ('SF', ''), ('GIDP', '2'))],
OrderedDict([('playerID', 'dietrbi01'), ('yearID', '1939'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB',
'37'), ('R', '5'), ('H', '8'), ('2B', '3'), ('3B', '0'), ('HR', '1'), ('RBI', '2'), ('SB', '0'), ('CS', '0'), ('BB', '4'), ('SO', '12'), ('IBB',
''), ('HBP', '0'), ('SH', '2'), ('SF', ''), ('GIDP', '0'))], OrderedDict([('playerID', 'sheame01'), ('yearID', '1937'), ('stint',
'1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB', '71'), ('R', '7'), ('H', '15'), ('2B', '1'), ('3B', '0'), ('HR', '0'), ('RBI',
'5'), ('SB', '1'), ('CS', '0'), ('BB', '15'), ('SO', '10'), ('IBB', ''), ('HBP', '0'), ('SH', '0'), ('SF', ''), ('GIDP', '0'))],
OrderedDict([('playerID', 'riddljo02'), ('yearID', '1930'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB',
'58'), ('R', '7'), ('H', '14'), ('2B', '3'), ('3B', '1'), ('HR', '0'), ('RBI', '4'), ('SB', '0'), ('CS', '0'), ('BB', '3'), ('SO', '6'), ('IBB',
''), ('HBP', '1'), ('SH', '2'), ('SF', ''), ('GIDP', '0'))], OrderedDict([('playerID', 'walshed02'), ('yearID', '1929'), ('stint',
'1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB', '43'), ('R', '5'), ('H', '10'), ('2B', '2'), ('3B', '1'), ('HR', '0'), ('RBI',
'6'), ('SB', '0'), ('CS', '0'), ('BB', '3'), ('SO', '9'), ('IBB', ''), ('HBP', '1'), ('SH', '1'), ('SF', ''), ('GIDP', '0'))],
OrderedDict([('playerID', 'jacobel01'), ('yearID', '1927'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB',
'20'), ('R', '4'), ('H', '3'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '2'), ('SB', '0'), ('CS', '1'), ('BB', '1'), ('SO', '5'), ('IBB',
''), ('HBP', '0'), ('SH', '4'), ('SF', ''), ('GIDP', '0'))], OrderedDict([('playerID', 'blankte01'), ('yearID', '1924'), ('stint', '1'),
('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB', '46'), ('R', '10'), ('H', '15'), ('2B', '2'), ('3B', '1'), ('HR', '1'), ('RBI', '5'),
('SB', '0'), ('CS', '0'), ('BB', '4'), ('SO', '6'), ('IBB', ''), ('HBP', '1'), ('SH', '3'), ('SF', ''), ('GIDP', '0'))],
OrderedDict([('playerID', 'faberre01'), ('yearID', '1919'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB',
'54'), ('R', '8'), ('H', '10'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '4'), ('SB', '0'), ('CS', '0'), ('BB', '6'), ('SO', '20'), ('IBB',
''), ('HBP', '0'), ('SH', '3'), ('SF', ''), ('GIDP', '0'))], OrderedDict([('playerID', 'olmstfr01'), ('yearID', '1911'), ('stint', '1'),
('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB', '37'), ('R', '3'), ('H', '7'), ('2B', '1'), ('3B', '1'), ('HR', '0'), ('RBI', '4'),
('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '0'), ('IBB', ''), ('HBP', '0'), ('SH', '1'), ('SF', ''), ('GIDP', '0'))],
OrderedDict([('playerID', 'owenfr01'), ('yearID', '1908'), ('stint', '1'), ('teamID', 'NLA'), ('lgID', 'FL'), ('G', '36'), ('AB',
'50'), ('R', '3'), ('H', '9'), ('2B', '3'), ('3B', '0'), ('HR', '0'), ('RBI', '4'), ('SB', '1'), ('CS', '0'), ('BB', '3'), ('SO', '15'), ('IBB',
''), ('HBP', '0'), ('SH', '1'), ('SF', ''), ('GIDP', '0'))]]

```

update_by_template(): Wrong field name in new_values()

Before UPDATE operation

```

[OrderedDict([('playerID', 'johnske05'), ('yearID', '2016'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'),
('AB', '121'), ('R', '8'), ('H', '26'), ('2B', '6'), ('3B', '0'), ('HR', '1'), ('RBI', '10'), ('SB', '1'), ('CS', '0'), ('BB', '10'), ('SO',
'25'), ('IBB', '1'), ('HBP', '0'), ('SH', '0'), ('SF', '1'), ('GIDP', '4'))], OrderedDict([('playerID', 'johnsji04'), ('yearID',
'2015'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('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'))],
OrderedDict([('playerID', 'simmoan01'), ('yearID', '2012'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'),
('AB', '166'), ('R', '17'), ('H', '48'), ('2B', '8'), ('3B', '2'), ('HR', '3'), ('RBI', '19'), ('SB', '1'), ('CS', '0'), ('BB', '12'), ('SO',
'21'), ('IBB', '1'), ('HBP', '1'), ('SH', '0'), ('SF', '3'), ('GIDP', '5'))], OrderedDict([('playerID', 'wickmbo01'), ('yearID',
'2007'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('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'))],
OrderedDict([('playerID', 'sanchre01'), ('yearID', '2001'), ('stint', '2'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('AB',
'154'), ('R', '10'), ('H', '35'), ('2B', '4'), ('3B', '1'), ('HR', '0'), ('RBI', '9'), ('SB', '2'), ('CS', '0'), ('BB', '4'), ('SO', '15'),
('IBB', '1'), ('HBP', '0'), ('SH', '4'), ('SF', '1'), ('GIDP', '9'))], OrderedDict([('playerID', 'sprinru01'), ('yearID', '1999'),
('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('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'))],
OrderedDict([('playerID', 'stantmi02'), ('yearID', '1994'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('AB',
'3'), ('R', '0'), ('H', '2'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '1'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '0'), ('IBB',
'0'), ('HBP', '0'), ('SH', '1'), ('SF', '0'), ('GIDP', '0'))], OrderedDict([('playerID', 'bedrost01'), ('yearID', '1993'), ('stint',
'1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('AB', '2'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'),
('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '2'), ('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0'))],
OrderedDict([('playerID', 'heathmi02'), ('yearID', '1991'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('AB',
'139'), ('R', '4'), ('H', '29'), ('2B', '3'), ('3B', '1'), ('HR', '1'), ('RBI', '12'), ('SB', '0'), ('CS', '0'), ('BB', '7'), ('SO', '26'),
('IBB', '5'), ('HBP', '1'), ('SH', '2'), ('SF', '1'), ('GIDP', '4'))], OrderedDict([('playerID', 'berenju01'), ('yearID', '1991'),
('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('AB', '5'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'),
('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '3'), ('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0'))],
OrderedDict([('playerID', 'assenpa01'), ('yearID', '1989'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'),
('AB', '2'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '1'),
('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '1'))], OrderedDict([('playerID', 'roeniga01'), ('yearID', '1988'),
('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('AB', '114'), ('R', '11'), ('H', '26'), ('2B', '5'), ('3B', '0'), ('HR',
'1'), ('RBI', '7'), ('SB', '0'), ('CS', '0'), ('BB', '8'), ('SO', '15'), ('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP',
'4'))], OrderedDict([('playerID', 'garbege01'), ('yearID', '1987'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'),
('AB', '4'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'), ('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '1'), ('SO', '2'),
('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '0'))], OrderedDict([('playerID', 'watsobo01'), ('yearID', '1984'),
('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('AB', '85'), ('R', '4'), ('H', '18'), ('2B', '4'), ('3B', '0'), ('HR', '2'),
('RBI', '12'), ('SB', '0'), ('CS', '0'), ('BB', '9'), ('SO', '12'), ('IBB', '2'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '2'))],
OrderedDict([('playerID', 'casanpa01'), ('yearID', '1972'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'),
('AB', '136'), ('R', '8'), ('H', '28'), ('2B', '3'), ('3B', '0'), ('HR', '2'), ('RBI', '10'), ('SB', '0'), ('CS', '1'), ('BB', '4'), ('SO',
'28'), ('IBB', '2'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '7'))], OrderedDict([('playerID', 'upshace01'), ('yearID',
'1971'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'), ('AB', '15'), ('R', '0'), ('H', '0'), ('2B', '0'), ('3B', '0'),
('HR', '0'), ('RBI', '0'), ('SB', '0'), ('CS', '0'), ('BB', '0'), ('SO', '5'), ('IBB', '0'), ('HBP', '0'), ('SH', '2'), ('SF', '0'), ('GIDP',
'1'))], OrderedDict([('playerID', 'aaronto01'), ('yearID', '1969'), ('stint', '1'), ('teamID', 'ATL'), ('lgID', 'NL'), ('G', '49'),
('AB', '60'), ('R', '13'), ('H', '15'), ('2B', '2'), ('3B', '0'), ('HR', '1'), ('RBI', '5'), ('SB', '0'), ('CS', '1'), ('BB', '6'), ('SO', '6'),
('IBB', '0'), ('HBP', '0'), ('SH', '0'), ('SF', '0'), ('GIDP', '1'))]]
```

Error: Field to be modified does not exist

Num Rows updated 0

update_by_key(): Modified value already exists in Primary Key field

Before UPDATE operation

```
OrderedDict([('playerID', 'aardsda01'), ('yearID', '2006'), ('stint', '1'), ('teamID', 'SFN'), ('lgID', 'AL'), ('G', '20'),
('AB', '2'), ('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', '1'), ('SF', '0'), ('GIDP', '0'))]
```

Error: Value key is modified to already exists in table

Num Rows updated 0

insert(): Unique Primary key

Before INSERT operation

Num Rows 105824

After INSERT operation

Num Rows 105825

insert(): Duplicate Primary key

Before INSERT operation

Num Rows 105825

Duplicate primary key found

After INSERT operation

Num Rows 105825

insert(): No Primary key specified

Before INSERT operation

Num Rows 105825

Specify primary key to insert new record

After INSERT operation

Num Rows 105825

```
DEBUG:root:CSVDataTable.init: data = { "table_name": "Batting", "connect_info": { "directory":  
"C:\\PersonalFiles\\Courses\\IntroToDB\\HW1\\W4111_HW1 F21-2\\W4111_HW1  
F21\\4111_s21_hw1_programming_CSVDataTable\\Data\\Baseball", "file_name": "Batting.csv" }, "key_columns": [  
"playerID", "yearID", "stint" ], "debug": true } DEBUG:root:CSVDataTable._load: Loaded 105861 rows
```