COMS W4111-002 (Fall 2021)
Introduction to Databases

Homework 1: Non-Programming - 10 Points

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

LastName_FirstName, UNI

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: (2 points) Loading data
- PART D: (2 points) Cleaning data using SQL and pandas
- PART E: (4 points) Performing SQL queries to analyze the data

Submission

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

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

Collaboration

- You may use any information found in TA or Prof. Ferg's office hours
- You are not allowed to collaborate outside of office hours
- You are NOT allowed to collaborate with other students outside of office hours.

Part A: Written

1. What is a database management system?

   Your answer here

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

   Your answer here

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

Your answer here

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

   Your answer here

1. What is an ER model?

   Your answer here

1. 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.

Use the following line to upload a photo of your Luicd Chart

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

   Your screenshot here

Part C: Loading Data

In this section we are going to work through loading data from CSV files to pandas DataFrames to SQL tables. You only need to write two lines of code for this entire section, and they can be found within the pandas documentation

Step 1: CSV to Pandas DataFrame

Loading data from files in various formats into Python data structures and databases for processing and analysis is extremely common. Complete the following code fragment. Only one line of code is required here. This fragment should load data from a CSV file into a Pandas data frame.Hint: use the file path

In [69]:

```
csv_files = ['flavors', 'generic-food']
dfs = {}
for f in csv_files:
    fn = "./Data/" + f + ".csv"

    ###### Your Code Goes Here #####



    ##### End Code #####
    dfs[f] = df

    print("Loaded file " + f)
```

```
Loaded file flavors
Loaded file generic-food
```

Tests

The next cells print the first 5 rows in each dataframe to help you confirm that you loaded the data. Your execution should produce the same answer as the cells below. Remember, your execution will overwrite the ones in this notebook. So, you should make a copy of the correct answers for comparison.

In [70]:

```
for k,v in dfs.items():
    print ("\n\n************* Data for dataframe = ", k, "*************\n")
    print (v.head(5))
```

```
************* Data for dataframe =  flavors *************

          FOOD    FLAVOUR
0       celery  vegetable
1         corn  vegetable
2     cucumber  vegetable
3  horseradish  vegetable
4    vegetable  vegetable


************* Data for dataframe =  generic-food *************

        FOOD NAME                 SCIENTIFIC NAME             GROUP  \
0        Angelica                Angelica keiskei  Herbs and Spices
1   Savoy cabbage  Brassica oleracea var. sabauda        Vegetables
2   Silver linden                  Tilia argentea  Herbs and Spices
3            Kiwi             Actinidia chinensis            Fruits
4  Allium (Onion)                           Allium        Vegetables

                 SUB GROUP
0                    Herbs
1                 Cabbages
2                    Herbs
3          Tropical fruits
4  Onion-family vegetables
```

Part 2: Pandas DataFrame to SQL

The following cell is an implementation template for a function that will create a table for a data frame and load the data into the table. Answer

- *You must create a connection.*

In [73]:

```
# The admin ID you chose when creating the instance.
rds_user = ""

# The host name that you copied from AWS console into DataGrip
rds_host = ""

# This is standard. Do not worry about it.
rds_port = 3306

# The password for your admin ID.
rds_password = ""
```

In [74]:

```
from sqlalchemy import create_engine


db_data = 'mysql+pymysql://' + rds_user + ':' + rds_password + '@' + rds_host + ':'+ str(r
engine = create_engine(db_data)
engine
```

Out[74]:

```
Engine(mysql+pymysql://admin:***@ara-project.ckkqqktwkcji.us-east-1.rds.amazonaws.com:330
6/)
```

- *This creates a new schema for your CSV files*

In [53]:

```
schema = "Foods"
sql = "create schema " + schema
cursor = conn.cursor()
cnt = cursor.execute(sql)
res = cursor.fetchall()
res
```

Out[53]:

```
()
```

- *You need to read the DataFrames you made to SQL, you should be able to do this in one line*

In [55]:

```
#HINT: you will need the "engine" that was created and the name of the schema you are conn
for table_name, data_frame in dfs.items():

    ##### Your Code Goes Here #####




    ##### End Your Code #####
    print("Created and loaded table = ", table_name)
```

```
Created and loaded table =  flavors
Created and loaded table =  generic-food
```

<u>Tests</u>

- *The following cell displays the first five rows from the* `generic-foods` *table.*
- *You should also check your SQL server on DataGrip and see if you can see your new schema and tables!*

In [62]:

```
sql = "select * from Foods.`generic-food` limit 5"
cursor = conn.cursor()
cnt = cursor.execute(sql)
res = cursor.fetchall()
res
```

Out[62]:

```
[{'index': 0,
  'FOOD NAME': 'Angelica',
  'SCIENTIFIC NAME': 'Angelica keiskei',
  'GROUP': 'Herbs and Spices',
  'SUB GROUP': 'Herbs'},
 {'index': 1,
  'FOOD NAME': 'Savoy cabbage',
  'SCIENTIFIC NAME': 'Brassica oleracea var. sabauda',
  'GROUP': 'Vegetables',
  'SUB GROUP': 'Cabbages'},
 {'index': 2,
  'FOOD NAME': 'Silver linden',
  'SCIENTIFIC NAME': 'Tilia argentea',
  'GROUP': 'Herbs and Spices',
  'SUB GROUP': 'Herbs'},
 {'index': 3,
  'FOOD NAME': 'Kiwi',
  'SCIENTIFIC NAME': 'Actinidia chinensis',
  'GROUP': 'Fruits',
  'SUB GROUP': 'Tropical fruits'},
 {'index': 4,
  'FOOD NAME': 'Allium (Onion)',
  'SCIENTIFIC NAME': 'Allium',
```

```
 'GROUP': 'Vegetables',
 'SUB GROUP': 'Onion-family vegetables'}]
```

Part D: Data Clean Up

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 E. Knowing how to clean data and add integrity constraints is very important which is why you go through the steps in part D.

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

In [63]:

```
# 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://dbuser:dbuserdbuser@localhost/lahmansdb_to_clean
```

You must also create a new engine to connect to lahmansdb_to_clean database using sqlalchemy. All of the code you need for this is found in Part C

In [ ]:

```
# Specify parameters
```

In [ ]:

```
# Make your new engine here
```

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 the DataGrip table editor, using sql queries, or pandas. In this part you will clean two tables: People, Batting and Teams. We specify with each question whether to use SQL or pandas.

You must have:

- *A brief explanation of why we asked you to make each change*
- *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 SQL workbench's table 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

1. *Primary Key (Explanation is given, but you still must add the key to your table yourself)*
2. *Empty strings to NULLs - SQL*

3. *Column typing - SQL*
4. *isDead column - Pandas*
5. *deathDate and birthDate column - Pandas*

Batting Table

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

Teams Table

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

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...*

1. *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*

1. *To paste the "Create Table Statement": Right click on the table > SQL Scripts > Generate DDL to Clipboard*

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 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)

SQL

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

Tests

In [2]:

```
%sql SHOW KEYS FROM people WHERE Key_name = 'PRIMARY'
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
1 rows affected.
```

Out[2]:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Inde
people	0	PRIMARY	1	playerID	A	19539	None	None		

1) SQL: Convert all empty strings to NULL

Explanation

Put your answer in this cell

Change

Put your answer in this cell

SQL

```
Put your answer in this cell
```

Tests

In [5]:

```
%sql SELECT * FROM people WHERE birthState = ""
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
0 rows affected.
```

Out[5]:

playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	deathMonth	deathDay	de

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

Explanation

Put your answer in this cell

Change

Put your answer in this cell

SQL

```
Put your answer in this cell
```

3) Pandas: 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?

'Y' means the player is dead

'N' means the player is alive

Explanation

Put your answer in this cell

Change

Put your answer in this cell

Pandas

In [ ]:

```
# Read in SQL table
df = pandas.read_sql("select * from people;", engine)

    ##### Your Code Goes Here #####


    ##### End Your Code #####

# Send updated dataframcedata back to sql
df.to_sql("people", engine, if_exists="replace")
```

Tests

In [6]:

```
%sql SELECT * FROM people WHERE isDead = "N" limit 10
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
10 rows affected.
```

Out[6]:

playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	deathMonth	deathDa
aardsda01	1981	12	27	USA	CO	Denver	None	None	Non
aaronha01	1934	2	5	USA	AL	Mobile	None	None	Non
aasedo01	1954	9	8	USA	CA	Orange	None	None	Non
abadan01	1972	8	25	USA	FL	Palm Beach	None	None	Non
abadfe01	1985	12	17	D.R.	La Romana	La Romana	None	None	Non
abbotgl01	1951	2	16	USA	AR	Little Rock	None	None	Non
abbotje01	1972	8	17	USA	GA	Atlanta	None	None	Non
abbotji01	1967	9	19	USA	MI	Flint	None	None	Non
abbotku01	1969	6	2	USA	OH	Zanesville	None	None	Non
abbotky01	1968	2	18	USA	MA	Newburyport	None	None	Non

4) Pandas: 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?

Explanation

Put your answer in this cell

Change

Put your answer in this cell

Pandas

In [ ]:

```
# Read in SQL table
df = pandas.read_sql("select * from people;", engine)

    ##### Your Code Goes Here #####


    ##### End Your Code #####

# Send updated dataframcedata back to sql
df.to_sql("people", engine, if_exists="replace")
```

Tests

In [9]:

```
%sql SELECT deathDate FROM people WHERE deathDate >= '2005-01-01' ORDER BY deathDate ASC L
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
10 rows affected.
```

Out[9]:

deathDate
2005-01-04 00:00:00
2005-01-07 00:00:00
2005-01-09 00:00:00
2005-01-10 00:00:00
2005-01-21 00:00:00
2005-01-22 00:00:00
2005-01-31 00:00:00
2005-02-04 00:00:00
2005-02-08 00:00:00
2005-02-11 00:00:00

In [10]:

```
%sql SELECT birthDate FROM people WHERE birthDate <= '1965-01-01' ORDER BY birthDate ASC L
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
10 rows affected.
```

Out[10]:

birthDate
1820-04-17 00:00:00
1824-10-05 00:00:00

1832-02-25 00:00:00
1832-09-17 00:00:00
1832-10-23 00:00:00
1835-01-10 00:00:00
1836-02-29 00:00:00
1837-12-26 00:00:00
1838-03-10 00:00:00
1838-07-16 00:00:00

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.

```
Put your answer in this cell
```

Batting Table

1) SQL: Convert all empty strings to NULL

Explanation

Put your answer in this cell

Change

Put your answer in this cell

SQL

```
Put your answer in this cell
```

Tests

In [11]:
```
%sql SELECT count(*) FROM lahman2019clean.batting where RBI is NULL;
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
1 rows affected.
```

Out[11]:

count(*)
756

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

Explanation

Put your answer in this cell

Change

Put your answer in this cell

SQL

```
Put your answer in this cell
```

3) SQL: Add a Primary Key

Two options for the Primary Key:

- *Composite Key: playerID, yearID, stint*
- *Covering Key: playerID, yearID, stint, teamID*

Choice

Put your answer in this cell

Explanation

Put your answer in this cell

SQL

```
Put your answer in this cell
```

Test

In [33]:

```
%sql SHOW KEYS FROM batting WHERE Key_name = 'PRIMARY' and Column_name = 'playerID'
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
1 rows affected.
```

Out[33]:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Inde
batting	0	PRIMARY	1	playerID	A	20242	None	None		

4) SQL: 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

Put your answer in this cell

Change

Put your answer in this cell

SQL

```
Put your answer in this cell
```

Tests

In [12]:

```
%sql Select playerID from batting where playerID not in (select playerID from people);
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
0 rows affected.
```

Out[12]:

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.

```
Put your answer in this cell
```

Teams Table

1) SQL: Convert all empty strings to NULL

Explanation

Put your answer in this cell

Change

Put your answer in this cell

SQL

```
Put your answer in this cell
```

Tests

In [20]:

```
%sql select * from teams where divID is NULL limit 10;
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
10 rows affected.
```

Out[20]:

yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	DivWin	WCWin	LgWin	WSWin	R	AB
1884	UA	ALT	ALT	None	10	25		6	19			N		90	899
1966	NL	ATL	ATL	None	5	163	82	85	77			N	N	782	5617
1967	NL	ATL	ATL	None	7	162	81	77	85			N	N	631	5450
1968	NL	ATL	ATL	None	5	163	81	81	81			N	N	514	5552
1954	AL	BAL	BAL	None	7	154	77	54	100			N	N	483	5206
1955	AL	BAL	BAL	None	7	156	79	57	97			N	N	540	5257
1956	AL	BAL	BAL	None	6	154	77	69	85			N	N	571	5090

1957	*AL*	*BAL*	*BAL*	*None*	*5*	*154*	*77*	*76*	*76*		*N*	*N*	*597*	*5264*
1958	*AL*	*BAL*	*BAL*	*None*	*6*	*154*	*78*	*74*	*79*		*N*	*N*	*521*	*5111*
1959	*AL*	*BAL*	*BAL*	*None*	*6*	*155*	*78*	*74*	*80*		*N*	*N*	*551*	*5208*

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

Explanation

Put your answer in this cell

Change

Put your answer in this cell

SQL

```
Put your answer in this cell
```

3) SQL: Add a Primary Key

Explanation

Put your answer in this cell

Change

Put your answer in this cell

SQL

```
Put your answer in this cell
```

Tests

In [37]:

```
%sql SHOW KEYS FROM teams WHERE Key_name = 'PRIMARY' and column_name = 'teamID'
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
1 rows affected.
```

Out[37]:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Inde
teams	0	PRIMARY	1	teamID	A	143	None	None		I

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.

```
Put your answer in this cell
```

4) SQL: Add a foreign key on teamID between the Team and Batting Tables

Explanation

Put your answer in this cell

Change

Put your answer in this cell

SQL

```
Put your answer in this cell
```

Tests

Write your own sql statement to test the creation of the foreign key.

In [ ]:

In [ ]:

In [ ]:

Part E: SQL Queries

NOTE: For these queries, you must connect to and use the CLEAN lahman database that we provided in hw0. This will ensure your solutions are consistent with the answers.

In [2]:

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

In [ ]:

Question 0

What is the average salary in baseball history?

In [ ]:

```
%%sql #PUT YOUR ANSWER IN THIS CELL
```

In [11]:

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

Out[11]:

avg(salary)
2085634.053125473

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 [7]:
```
%%sql #PUT YOUR ANSWER IN THIS CELL
```

```
 * mysql+pymysql://dbuser:***@localhost/lahman2019clean
2 rows affected.
```

Out[7]:
```
[]
```

In [6]:
```
%%sql
```

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

Out[6]:

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.

In [ ]:
```
%%sql #PUT YOUR ANSWER IN THIS CELL
```

In [18]:
```
%sql
```

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

Out[18]:
```
[]
```

In [19]:
```
%sql select * from schools where schoolID="columbia"
```

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

Out[19]:

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

Question 3

Find the playerID, awardID, yearID, lgID, death_date, and G_all for Ted Williams (willite01) in 1939.

You will need to use the appearances, awardsplayers and people tables

In [ ]:
```
%%sql #PUT YOUR ANSWER IN THIS CELL
```

In [7]:
```
%%sql
```

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

Out[7]:

playerID	awardID	yearID	lgID	death_date	G_all
willite01	Baseball Magazine All-Star	1939	AL	2002-07-05	149
willite01	Baseball Magazine All-Star	1939	ML	2002-07-05	149
willite01	TSN All-Star	1939	ML	2002-07-05	149

Question 4

Find the highest salaries for players born after 1970. Order alphabetically. Limit 10.

Answer:

In [ ]:
```
%%sql #PUT YOUR ANSWER IN THIS CELL
```

Expected output

In [14]:
```
%%sql
```

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

Out[14]:

salary	playerID	nameFirst	nameLast
421000.0	ellisaj01	A. J.	Ellis
490000.0	ellisaj01	A. J.	Ellis
2000000.0	ellisaj01	A. J.	Ellis
3550000.0	ellisaj01	A. J.	Ellis
4250000.0	ellisaj01	A. J.	Ellis
4500000.0	ellisaj01	A. J.	Ellis
170000.0	hinchaj01	A. J.	Hinch
230000.0	hinchaj01	A. J.	Hinch
260000.0	hinchaj01	A. J.	Hinch
260000.0	hinchaj01	A. J.	Hinch

Question 5

Find the personal best salary for players who were born after 1980. Order by largest to smallest salary. Limit 10.

In [ ]:
```
%%sql #PUT YOUR ANSWER IN THIS CELL
```

In [20]:

```
%%sql
```

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

Out[20]:

pb_salary	playerID	nameFirst	nameLast
33000000.0	kershcl01	Clayton	Kershaw
31799030.0	greinza01	Zack	Greinke
30000000.0	priceda01	David	Price
28000000.0	verlaju01	Justin	Verlander
28000000.0	cabremi01	Miguel	Cabrera
27328046.0	cespeyo01	Yoenis	Cespedes
25857143.0	hernafe02	Felix	Hernandez
25000000.0	lestejo01	Jon	Lester
24000000.0	canoro01	Robinson	Cano
24000000.0	fieldpr01	Prince	Fielder