**Milestone #2: Mid Project**

**NFL Play statistics Dataset**

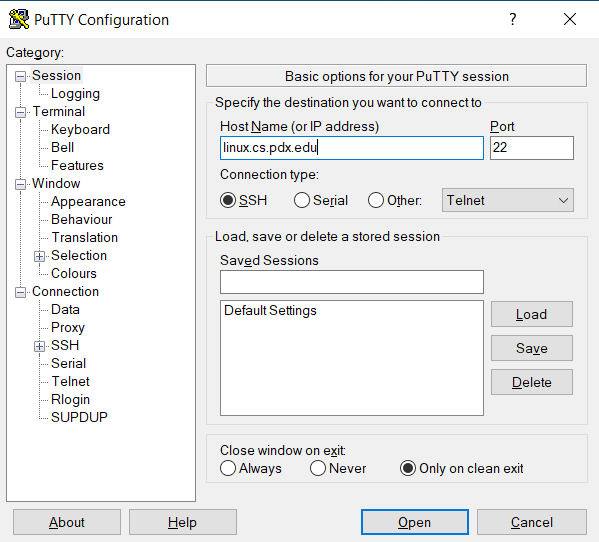
Submitted by - ***Chitradevi Maruthavanan***

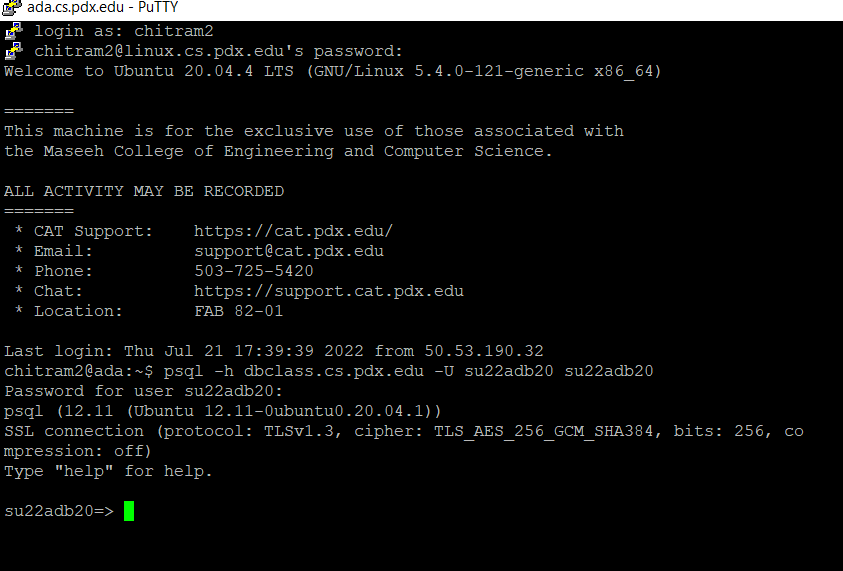
**Reasons to change the proposal from milestone #1:**

The COVID Vaccine dataset at <https://github.com/nychealth/covid-vaccine-data> had issues in relating one table with other. I could not establish any foreign key for the tables. So, I chose the new dataset on NFL Play statistics from <https://www.kaggle.com/datasets/toddsteussie/nfl-play-statistics-dataset-2004-to-present?resource=download>

**Postgres installation Steps:**

To show that I have successfully installed Postgres on my virtual machine. The screenshot is below:





**Source of data:**

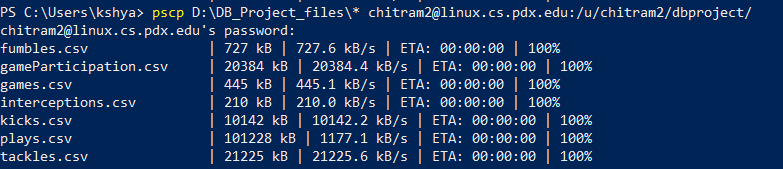
I am going to use the NFL (National Football League) Play statistics dataset. This NFL dataset provides play-by-play data from the 2004 to 2019 seasons. The dataset is at <https://www.kaggle.com/datasets/toddsteussie/nfl-play-statistics-dataset-2004-to-present?resource=download>

The dataset contains the following CSV files which contains a lot of data as elaborated below

1. plays.csv – Contains important data related to the game such as play type, possession team, non-possession team, field position, net yards etc.
2. games.csv – Contains game related data such as game time, season, weeks etc
3. kicks.csv – Contains kicks related data for the games such as Kick type, Kick outcome etc
4. interceptions.csv – Contains data related to interception position, interception yards etc
5. fumbles.csv – Contains information such as fumble type, fumble position and fumble turn around.
6. tackles.csv – Contains data such as tackle type, tackle position and tackle yards
7. gameParticipation.csv- Contains information such as game Participant name, college detail, participant home city details

These CSV files are transferred to the pdx linux machines using pscp as below.

*pscp D:\DB\_Project\_files\\* chitram2@linux.cs.pdx.edu:/u/chitram2/dbproject*



Next step is to create and populate the tables in the database using below steps.

The **data types** used in my tables are

1. Text
2. Int
3. Float
4. Date

**Table Creation:**

1. **Games** table

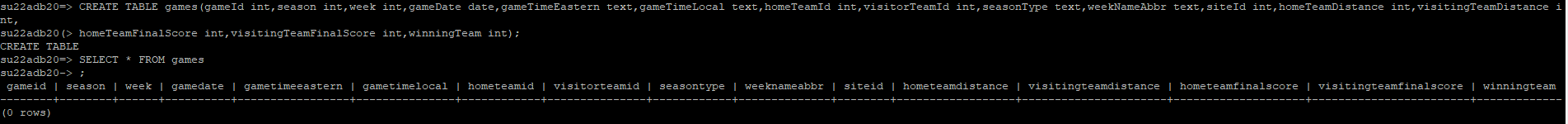
**Data Preprocessing:**

I use the data as it is. No Preprocessing was done.

**CREATE Command for table**

CREATE TABLE games(gameId int,season int,week int,gameDate date,gameTimeEastern text,gameTimeLocal text,homeTeamId int,visitorTeamId int,seasonType text,weekNameAbbr text,siteId int,homeTeamDistance int,visitingTeamDistance int,

homeTeamFinalScore int,visitingTeamFinalScore int,winningTeam int);

****

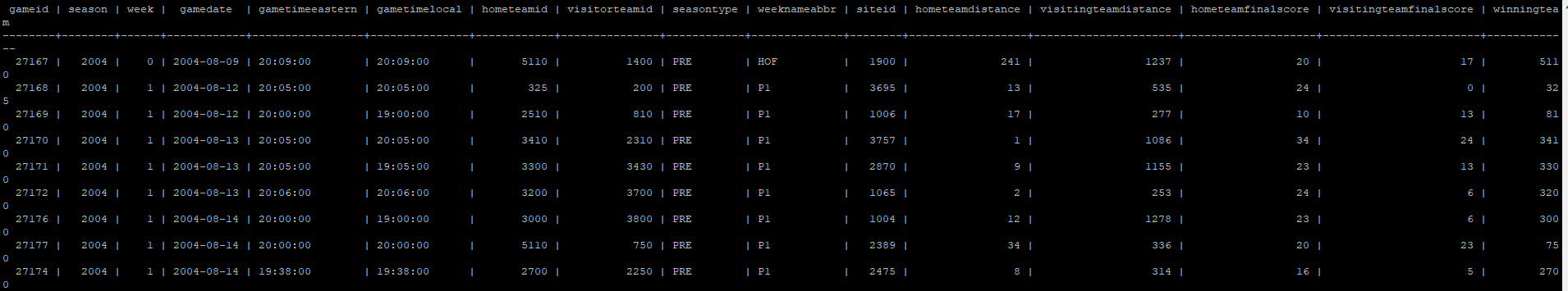
**Copy data from CSV to table:**

\COPY games from games.csv with csv header

****

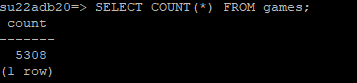
**Screenshot of the populated table:**

SELECT \* FROM games;



**Cardinality of the table:**

SELECT COUNT(\*) FROM games;

****

**Primary key:**

ALTER TABLE games add constraint pkey\_games primary key(gameid);

****

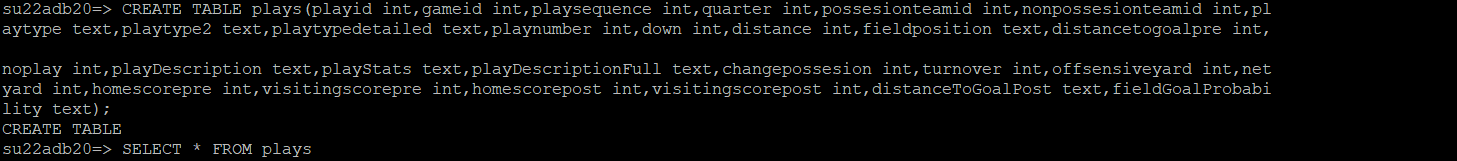
1. **Plays** table

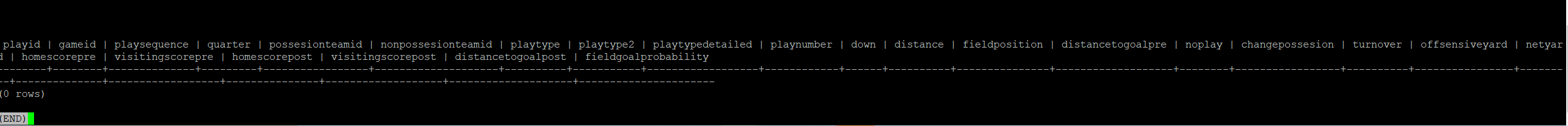
**Data Preprocessing:**

I removed few columns – gameclock, safety, firstdown, efficientplay, evpre, evpost, evplay, downconversion, huddle and formation because data is not sufficiently populated.

**CREATE Command for table**

CREATE TABLE plays(playid int, gameid int, playsequence int, quarter int, possesionteamid int, nonpossesionteamid int, playtype text, playtype2 text, playtypedetailed text, playnumber int ,down int, distance int, fieldposition text, distancetogoalpre int, noplay int, playDescription text, playStats text, playDescriptionFull text, changepossesion int, turnover int, offsensiveyard int, net yard int, homescorepre int, visitingscorepre int ,homescorepost int, visitingscorepost int, distanceToGoalPost text, fieldGoalProbabi lity text);





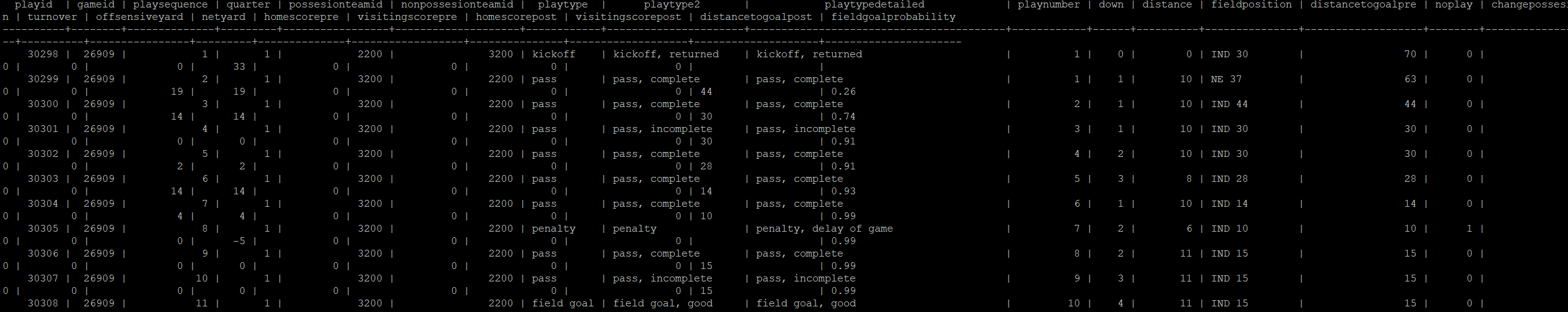
**Copy data from CSV to table:**

\COPY plays from plays.csv with csv header



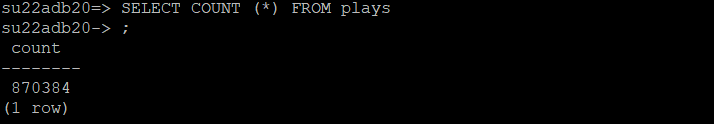
**Screenshot of the populated table:**

SELECT \* FROM plays;



**Cardinality of the table:**

SELECT COUNT(\*) FROM plays;

****

**Primary key:**

ALTER TABLE plays add constraint pkey\_plays primary key(playId);



**Foreign key:**

ALTER TABLE plays add constraint fk\_gameid foreign key(gameid) references games(gameid);

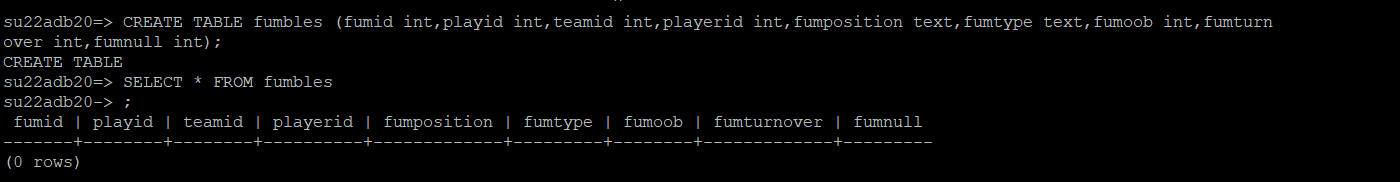
****

1. **Fumbles** table

**Data Preprocessing:**

I use the data as it is. No Preprocessing was done.

**CREATE Command for table**

CREATE TABLE **fumbles** (fumid int, playid int, teamid int, playerid int, fumposition text, fumtype text, fumoob int, fumturnover float, fumnull int);

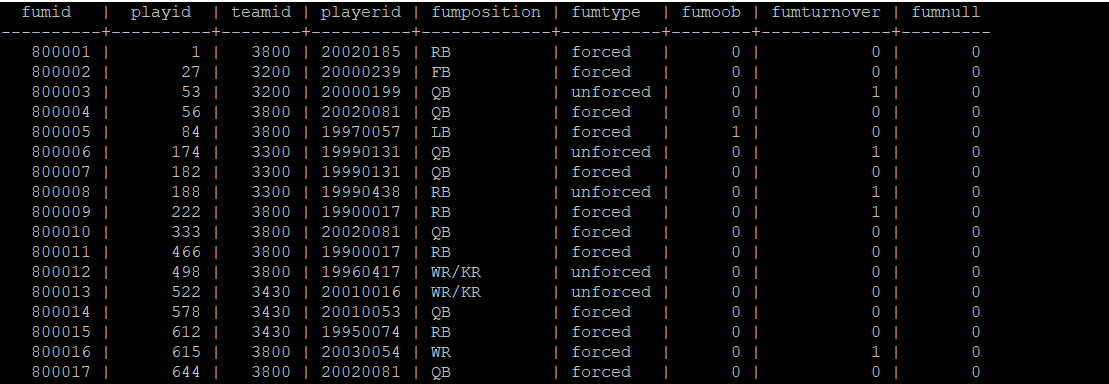
**Copy data from CSV to table:**

\COPY fumbles from fumbles.csv with csv header



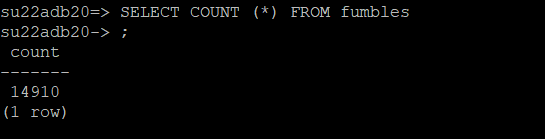
**Screenshot of the populated table:**

SELECT \* FROM fumbles;

****

**Cardinality of the table:**

SELECT COUNT( \*) FROM fumbles;



**Primary Key:**

ALTER TABLE fumbles add constraint pkey\_fumbles primary key(fumid);



**Foreign key:**

ALTER TABLE fumbles add constraint fk\_plays foreign key(playid) references plays(playid);

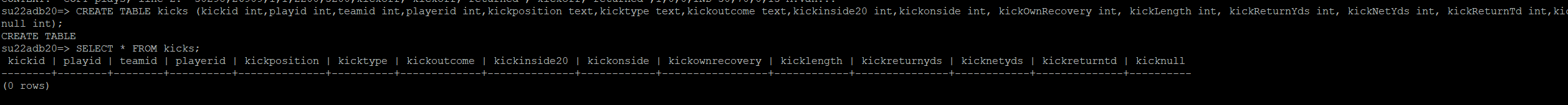


1. **Kicks** table

**Data Preprocessing:**

I use the data as it is. No Preprocessing was done.

**CREATE Command for table**

CREATE TABLE kicks (kickid int, playid int, teamid int, playerid int, kickposition text, kicktype text, kickoutcome text, kickinside20 int, kickonside int, kickOwnRecovery int, kickLength int, kickReturnYds int, kickNetYds int, kickReturnTd int, kicknull int);

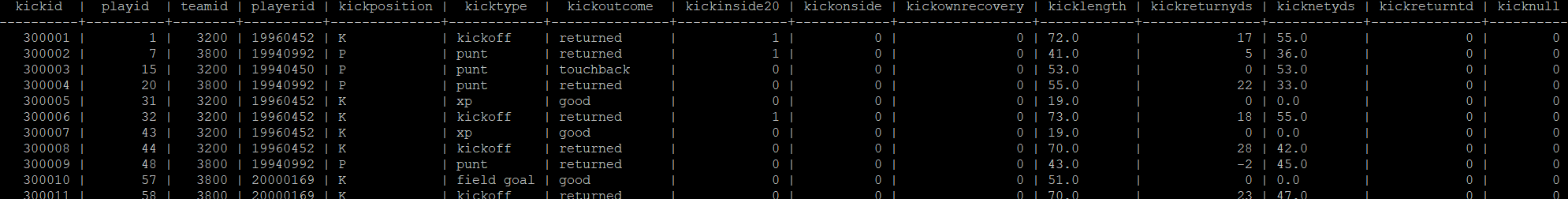
**Copy data from CSV to table:**

\COPY kicks from kicks.csv with csv header



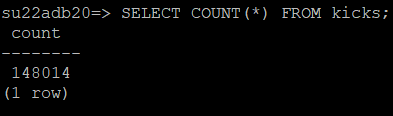
**Screenshot of the populated table:**

SELECT \* FROM kicks;



**Cardinality of the table:**

SELECT COUNT( \*) FROM kicks;



**Primary Key:**

ALTER TABLE kicks add constraint pkey\_kicks primary key(kickid);



**Foreign key:**

ALTER TABLE kicks add constraint fk\_players foreign key(playid) references plays(playid);



1. **Interception** table

**Data Preprocessing:**

Used pandas library in python to remove duplicate rows in the table.

**Code:**

>>>import pandas as pd

>>> file\_name = r"D:\DB\_Project\_files\interceptions\_original.csv"

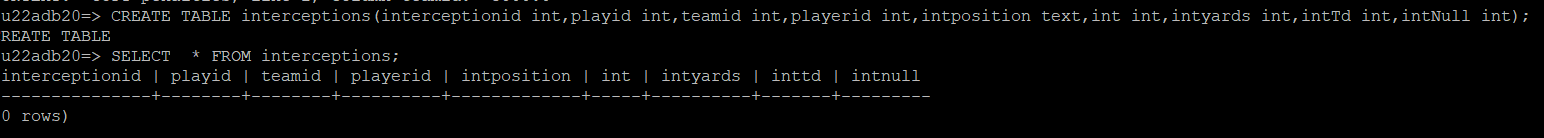
>>> file\_name\_output = r"D:\DB\_Project\_files\interceptions.csv"

>>> df = pd.read\_csv(file\_name)

>>> df.drop\_duplicates(subset=None, inplace=True)

>>> df.to\_csv(file\_name\_output, index=False)

**CREATE Command for table**

CREATE TABLE **interceptions** (interceptionid int, playid int, teamid int, playerid int, intposition text, int int, intyards int, intTd int, intNull int);

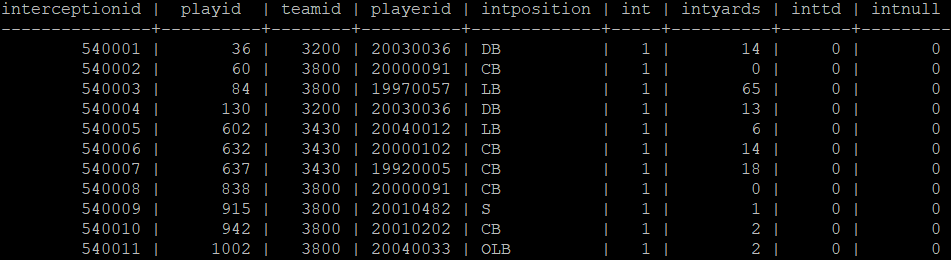
**Copy data from CSV to table:**

\COPY interceptions from interceptions.csv with csv header

****

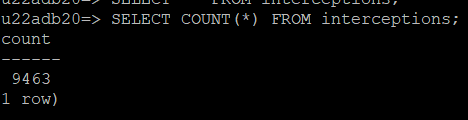
**Screenshot of the populated table:**

SELECT \* FROM interception;



**Cardinality of the table:**

SELECT COUNT(\*) FROM interception;



**Primary key:**

ALTER TABLE interceptions add constraint pkey\_interceptions primary key(interceptionid);

****

**Foreign key:**

ALTER TABLE interceptions add constraint fk\_playid foreign key(playid) references plays(playid);

****

1. **Tackles** table

**Data Preprocessing:**

Used pandas library in python to remove duplicate rows in the table.

**Code:**

>>>import pandas as pd

>>> file\_name = r"D:\DB\_Project\_files\tackles\_original.csv"

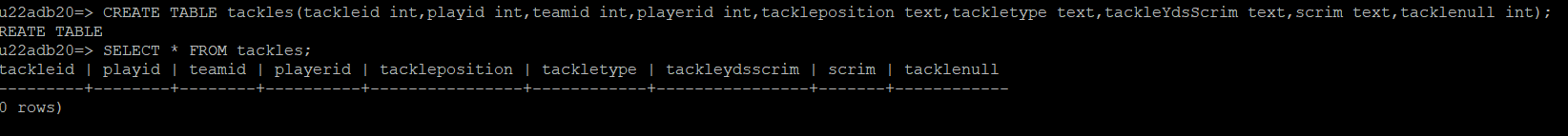
>>> file\_name\_output = r"D:\DB\_Project\_files\tackles.csv

>>> df = pd.read\_csv(file\_name)

>>> df.drop\_duplicates(subset=None, inplace=True)

>>> df.to\_csv(file\_name\_output, index=False)

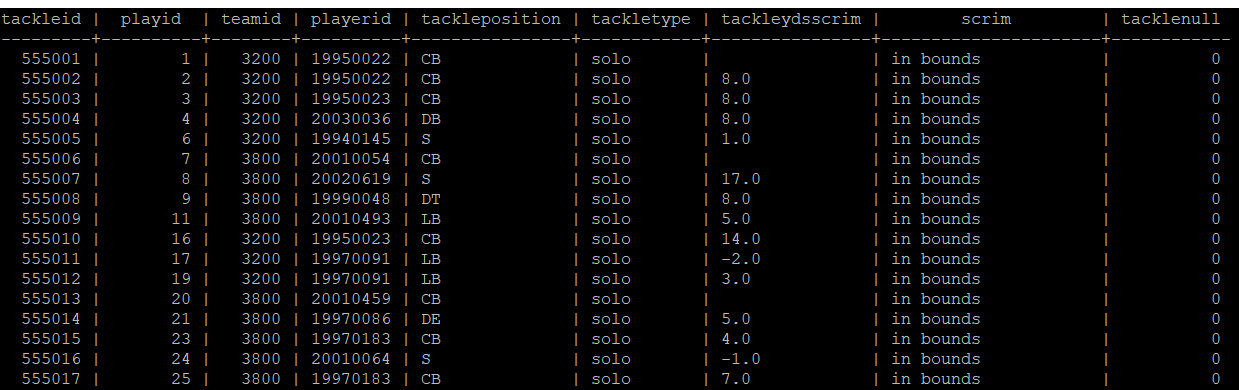
**CREATE Command for table**

CREATE TABLE **tackles** (tackleid int, playid int, teamid int, playerid int, tackleposition text, tackletype text, tackleYdsScrim text, scrim text, tacklenull int);  **Copy data from CSV to table:**

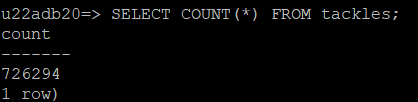
\COPY tackles from tackles.csv with csv header

**Screenshot of the populated table:**

SELECT \* FROM tackles;

**Cardinality of the table:**

SELECT COUNT(\*) FROM tackles;



**Primary Key:**

ALTER TABLE tackles add constraint pkey\_tackles primary key(tackleid);

****

1. **Game Participation** table

**Data Preprocessing:**

Used pandas library in python to remove duplicate rows in the table.

**Code:**

>>>import pandas as pd

>>> file\_name = r"D:\DB\_Project\_files\gameParticipation\_original.csv"

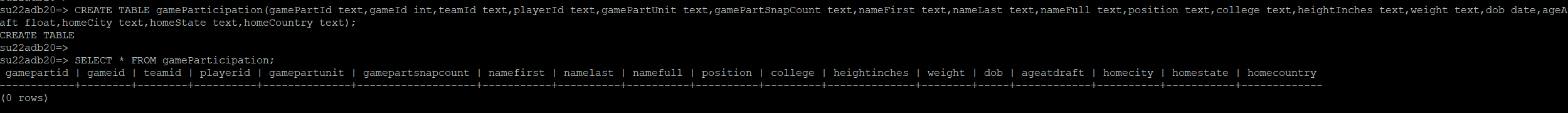
>>> file\_name\_output = r"D:\DB\_Project\_files\gameParticipation.csv

>>> df = pd.read\_csv(file\_name)

>>> df.drop\_duplicates(subset=None, inplace=True)

>>> df.to\_csv(file\_name\_output, index=False)

**CREATE Command for table**

CREATE TABLE gameParticipation(gamePartId text,gameId int,teamId text,playerId text,gamePartUnit text,gamePartSnapCount text,nameFirst text,nameLast text,nameFull text,position text,college text,heightInches text,weight text,dob date,ageAtDraft float,homeCity text,homeState text,homeCountry text);

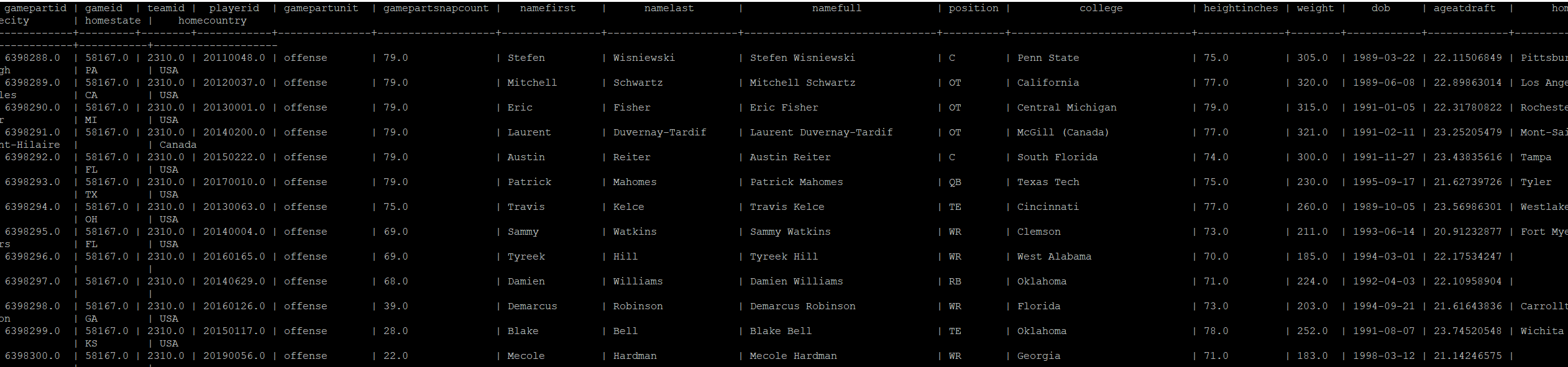
**Copy data from CSV to table:**

\COPY gameParticipation from gameParticipation.csv with csv header



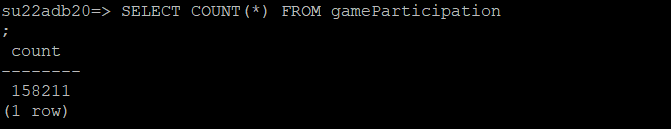
**Screenshot of the populated table:**

SELECT \* FROM gameParticipation;

****

**Cardinality of the table:**

SELECT COUNT(\*) FROM tackles;



**Primary Key:**

ALTER TABLE gameParticipation add constraint pkey\_gamePartId primary key(gamePartId);

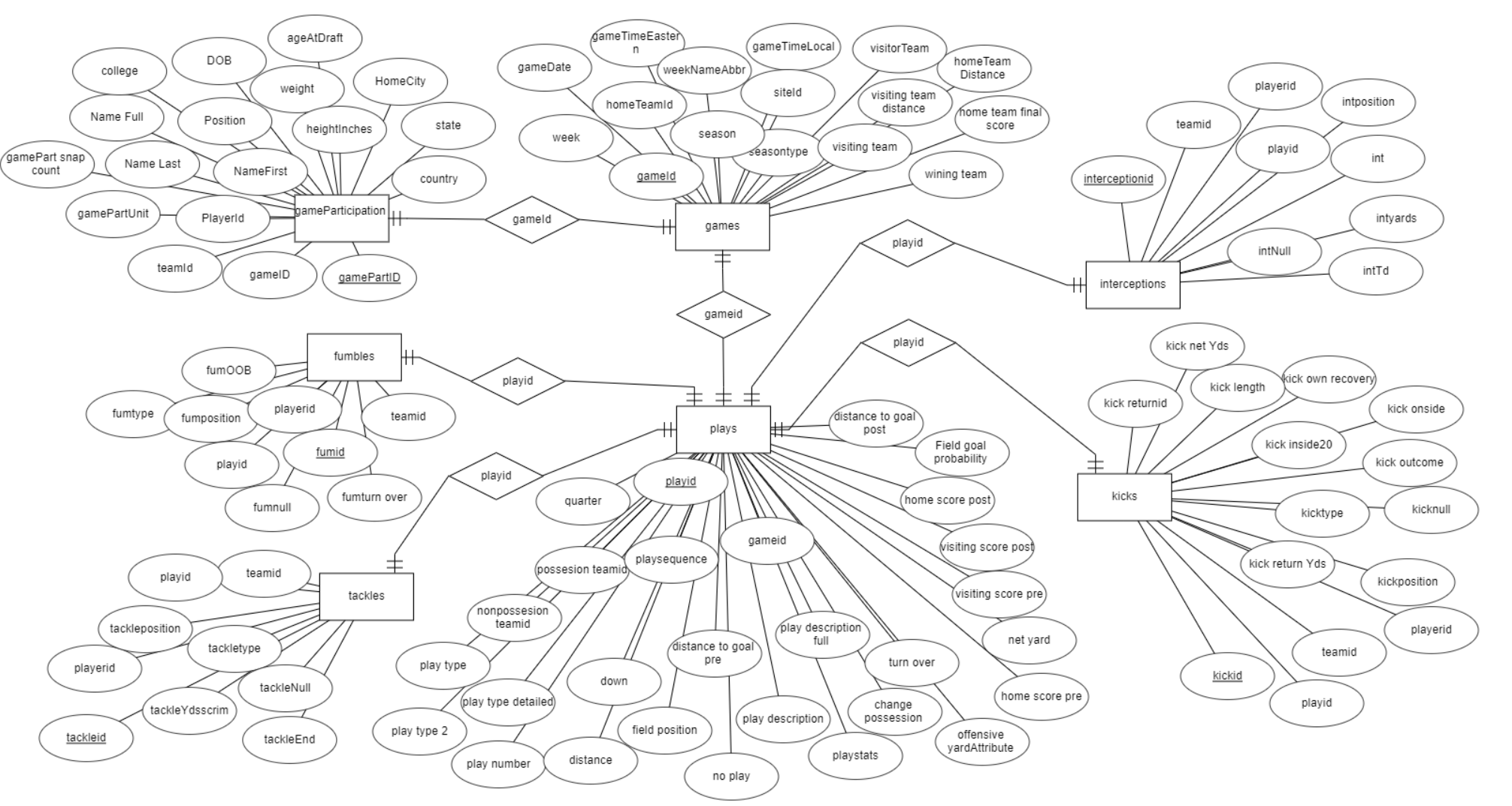


**Foreign Key:**

ALTER TABLE gameParticipation add constraint fk\_gameParticipationid foreign key(gameid) references games(gameid);

****

**ER-Diagram:**



Explain select distinct gp.college from gameparticipation gp INNER JOIN games g on gp.gameid = g.gameid INNER join plays p on p.gameid = g. g

ameid where P.fieldposition = 'SF 25';

277 rows

Explain select distinct gp.college from gameparticipation gp where gp.gameid IN (select p.gameid from plays p where p.fieldposition = 'SF 25' );

select distinct gp.college from gameparticipation gp where gp.college IN (Select p.fieldposition from plays p,games g where p.gameid = g.gameid and p.fieldposition = 'SF 25');

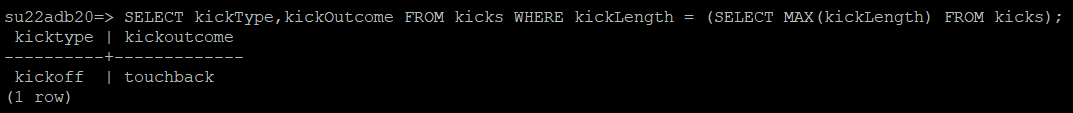
**Questions**

**Q1) Finding out the type and outcome of the kick with the longest length among all kicks.**

Answer: The table that is used to find the answer is kicks.

The query type is Subquery with aggregate function which gives the below result:

SELECT kickType, kickOutcome FROM kicks WHERE kickLength = (SELECT MAX (kickLength) FROM kicks);



Total number of rows returned = 1 row

**Q2) Find the name of the players who play the type two point and belong to possession team ID 1540 with home score pre greater than 29.**

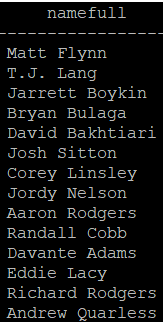
Answer: The tables that are used to find the answer is gameParticipation, games and plays.

The query type is Inner Join query with more than 2 tables and LIMIT clause which gives the below result:

SELECT nameFull FROM gameParticipation gp

INNER JOIN games g ON g.gameId = gp.gameId

INNER JOIN plays p ON g.gameId = p.gameId

WHERE p.possesionteamid = 1540 and p.playtype = 'two-point' and p.homescorepre > 29;

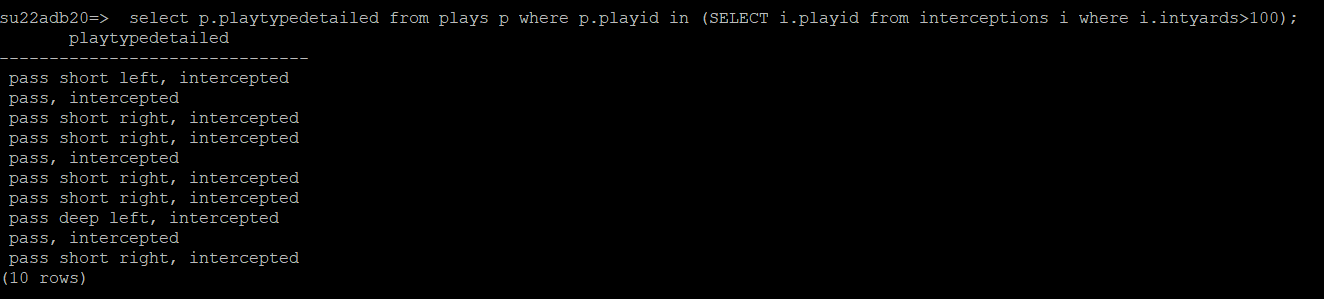
Total number of rows returned = 150 rows

**Q3) Return the detailed play type for plays whose interception yards is greater than 100.**

Answer: The tables that are used to find the answer is plays and interceptions.

The query type is Subquery with IN which gives the below result:

SELECT p.playtypedetailed FROM plays p WHERE p.playid IN (SELECT i.playid FROM interceptions i WHERE i.intyards>100);



Total number of rows returned = 10 rows

**Q4) Provide the list of players’ name and college attended for players whose date of birth is not populated**

Answer: The table that is used to find the answer is gameParticipation

The query type is DISTINCT and NULL value which gives the below result:

SELECT DISTINCT(nameFull),college FROM gameParticipation WHERE dob IS NULL;



Total number of rows returned = 172 rows

**Q5) Find the count of Plays where forced fumbles happened along with tackles.**

Answer: The table that is used to find the answer is fumbles, plays and tackles

The query type is GROUP BY, COUNT and HAVING which gives the below result:

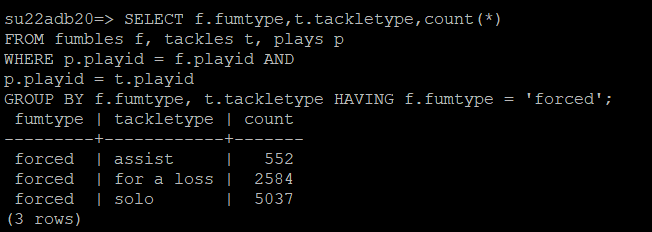
SELECT f.fumtype,t.tackletype,count(\*)

FROM fumbles f, tackles t, plays p

WHERE p.playid = f.playid AND

p.playid = t.playid

GROUP BY f.fumtype, t.tackletype HAVING f.fumtype = 'forced';



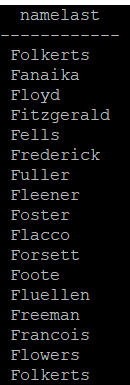
Total number of rows returned = 3 rows

**Q6) Find the players’ last names that begins with letter ‘F’ whose week name abbreviation is ‘WC’.**

Answer: The table that is used to find the answer is gameParticipation and games

The query type is JOIN and LIKE clause which gives the below result:

SELECT gp.nameLast FROM gameParticipation gp JOIN games g ON gp.gameid = g.gameid WHERE g.weeknameabbr = 'WC' AND gp.nameLast LIKE 'F%';

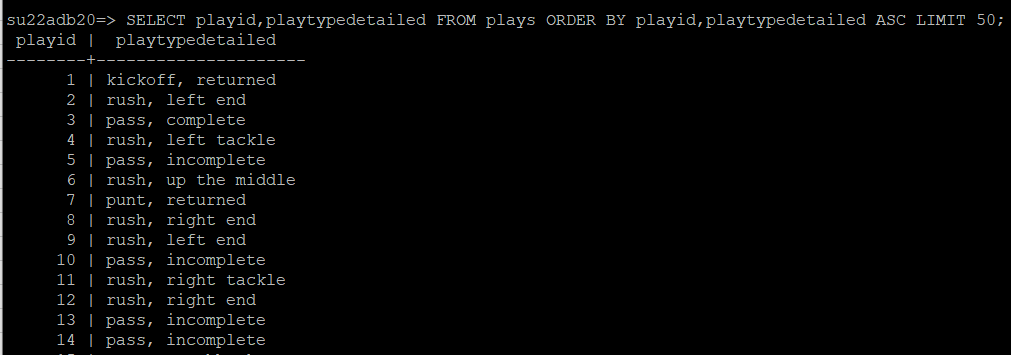


Total number of rows returned = 63 rows

**Q7) List the first 50 plays and the detailed type of the plays**

Answer: The table that is used to find the answer is plays

The query type is ORDER BY which gives the below result:

SELECT playid,playtypedetailed FROM plays ORDER BY playid,playtypedetailed ASC LIMIT 50;

Total number of rows returned = 50 rows

**Q8) Write an SQL view definition that displays each fumble type and, calculate and display the average of visiting score post for plays belonging to each fumble type.**

Answer:

The table that is used to find the answer is plays and fumbles

The query type is View which gives the below result:

CREATE VIEW FumtypeAvgVisitingscorepostView AS

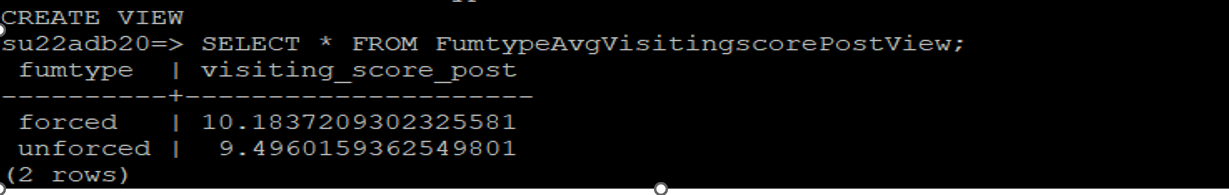
SELECT f.fumtype, AVG(p.visitingscorepost) AS visiting\_score\_post

FROM plays p,fumbles f

where f.playid = p.playid

GROUP BY f.fumtype;

SELECT \* FROM FumtypeAvgVisitingscorePostView;



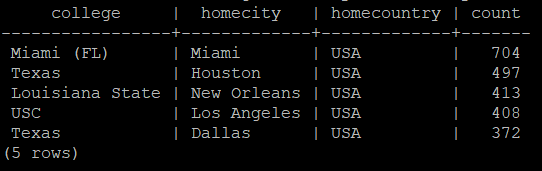
Total number of rows returned = 2 rows

**Q9) List the top five colleges attended by the NFL players and their home city and home country.**

Answer:

The table that is used to find the answer is plays and fumbles

The query type is GROUP BY and NOT NULL which gives the below result:

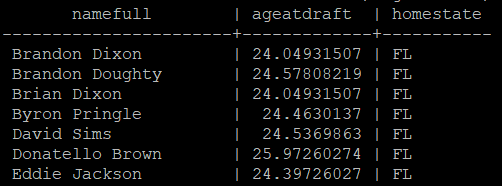
SELECT college,homecity,homecountry,count(college) FROM gameParticipation GROUP BY college, homecity, homecountry HAVING homecountry is NOT NULL ORDER BY COUNT DESC LIMIT 5; 

Total number of rows returned = 5 rows

**Q10) List the full names, age at drafting and home state for the players who were drafted at age 24 and above and belong to the state of Florida.**

Answer:

The table that is used to find the answer is gameParticipation

SELECT DISTINCT nameFull, ageAtDraft, homeState FROM gameParticipation WHERE ageAtDraft>24 AND homeState = 'FL';

Total number of rows returned = 27 rows

**c) Optimized query plan:**

select distinct gp.college from gameparticipation gp where gp.gameid IN (select p.gameid from plays p where p.fieldposition = 'SF 25' and p.playtype = ‘field goal’ );

Query tree 1:

∏gp.college (on-the-fly)

⋈gp.gameid = p.gameid (Hashjoin)

σp.playtype = ‘field goal’ σp.fieldposition = ‘SF 25’(on-the  fly)

**gameparticipant** (scan) plays **(scan)**

## M- # of pages in outer table(playtype = ‘field goal’)

## N- # of pages in inner table (field position = ‘SF 25’)

Query tree 2:

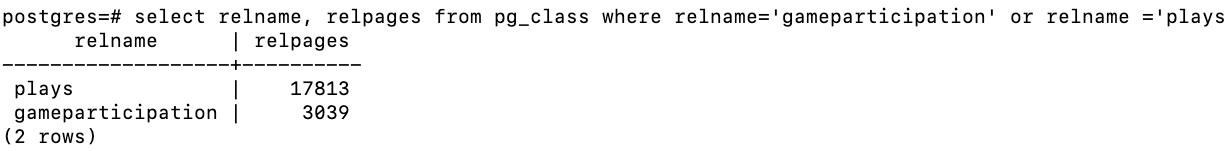
∏gp.college. (on-the-fly)

σp.fieldposition = ‘SF 25’ ^ p.playtype = ‘field goal’ (on-the fly)

⋈gp.gameid = p.gameid**.** (page-oriented nested loop join)

gameparticipant plays

## (scan) (scan)



M= # of pages in outer table (game participant) = 17813

N = # of pages in inner table(plays) = 3039

Cost of page oriented nested loop join is : M+M\*N

17813+17813\*3039

108,267,414

**c) Optimized query plan:**

select nameFull, MIN(ageatdraft)AS ageatdraft from gameParticipation GROUP BY nameFull ORDER BY MIN(ageatdraft) ASC LIMIT 1;

Query tree 1:

∏nameFull,MIN(ageatdraft) (on-the-fly)

GROUP BY

nameFull

**gameparticipant** (scan) plays **(scan)**

## M- # of pages in outer table(playtype = ‘field goal’)

## N- # of pages in inner table (field position = ‘SF 25’)