1.Analyzing the NFL games

In [29]:

```
import pandas as pd
import matplotlib.pyplot as plt
import datetime

%matplotlib inline
```

In [30]:

```
# Next, importing the CSV file called games.csv which contains information abc
```

In [31]:

```
# Reading in the CSV file as a DataFrame
games_df = pd.read_csv("C:/Users/fai/Downloads/games.csv")
```

In [32]:

```
# Looking at the first five rows
games_df.head()
```

Out[32]:

	gameId	gameDate	gameTimeEastern	homeTeamAbbr	visitorTeamAbbr	week
0	2018090600	09/06/2018	20:20:00	PHI	ATL	1
1	2018090901	09/09/2018	13:00:00	CLE	PIT	1
2	2018090902	09/09/2018	13:00:00	IND	CIN	1
3	2018090903	09/09/2018	13:00:00	MIA	TEN	1
4	2018090900	09/09/2018	13:00:00	BAL	BUF	1

In [33]:

```
# Let us look at the shape of the DataFrame to determine how many games were p
```

In [34]:

```
# Viewing the shape of the DataFrame
games_df.shape
```

Out[34]:

```
(253, 6)
```

In [35]:

```
#Before we begin our analysis, let us convert the date and time columns to Par

#This will help to standarize such data across the multiple datasets that we w
```

In [36]:

```
# Converting to datetime.date values
games_df['gameDate'] = pd.to_datetime(games_df['gameDate']).dt.date

# Converting to datetime.time values
games_df['gameTimeEastern'] = pd.to_datetime(games_df['gameTimeEastern']).dt.t

# Looking at the first five rows
games_df.head()
```

Out[36]:

	gameId	gameDate	gameTimeEastern	homeTeamAbbr	visitorTeamAbbr	week
0	2018090600	2018-09-06	20:20:00	PHI	ATL	1
1	2018090901	2018-09-09	13:00:00	CLE	PIT	1
2	2018090902	2018-09-09	13:00:00	IND	CIN	1
3	2018090903	2018-09-09	13:00:00	MIA	TEN	1
4	2018090900	2018-09-09	13:00:00	BAL	BUF	1

In [37]:

```
#Now, let us understand how the games are distributed in accordance to the dat

#Starting the analysis by looking at the distribution of games in relation to 
```

In [38]:

```
# Checking the frequency of games in relation to game dates
# games_df['gameDate'].value_counts().reset_index()

games_df['gameDate'].value_counts().reset_index()
```

Out[38]:

	index	gameDate
0	2018-12-30	16
1	2018-12-02	14
2	2018-09-16	14
3	2018-09-23	14
4	2018-12-09	14
5	2018-09-30	13
6	2018-12-23	13
7	2018-10-14	13
8	2018-10-07	13
9	2018-11-11	12
10	2018-10-28	12
11	2018-10-21	12
12	2018-12-16	12
13	2018-11-25	11
14	2018-11-18	11
15	2018-11-04	11
16	2018-09-09	10
17	2018-11-22	3
18	2018-12-15	2
19	2018-12-22	2
20	2018-09-10	2
21	2018-10-22	1
22	2018-10-25	1
23	2018-12-24	1
24	2018-09-13	1
25	2018-12-17	1
26	2018-09-17	1
27	2018-09-20	1
28	2018-12-13	1
29	2018-12-10	1
30	2018-09-24	1
31	2018-12-06	1
32	2018-12-03	1
33	2018-09-27	1
34	2018-11-29	1
35	2018-11-26	1

	index	gameDate
36	2018-10-01	1
37	2018-10-04	1
38	2018-11-19	1
39	2018-10-08	1
40	2018-11-15	1
41	2018-11-12	1
42	2018-10-11	1
43	2018-11-08	1
44	2018-11-05	1
45	2018-10-15	1
46	2018-11-01	1
47	2018-10-29	1
48	2018-10-18	1
49	2018-09-06	1

In [39]:
```
# There were a total of 50 different game dates.
```

In [40]:
```
# Checking the frequency of games in relation to game dates
date_dist = games_df['gameDate'].value_counts().reset_index()

# Renaming the columns
date_dist.columns = ['date', 'frequency']

# Looking at the first five rows
date_dist.head()
```

Out[40]:

	date	frequency
0	2018-12-30	16
1	2018-12-02	14
2	2018-09-16	14
3	2018-09-23	14
4	2018-12-09	14

In [41]:
```
#Next, sorting the data based on the date and setting the index as the date.
```

In [42]:

```
# Sorting the DataFrame based on the date values
sorted_date_dist = date_dist.sort_values('date').set_index('date')

# Looking at the first five rows
sorted_date_dist.head()
```

Out[42]:

	frequency
date	
2018-09-06	1
2018-09-09	10
2018-09-10	2
2018-09-13	1
2018-09-16	14

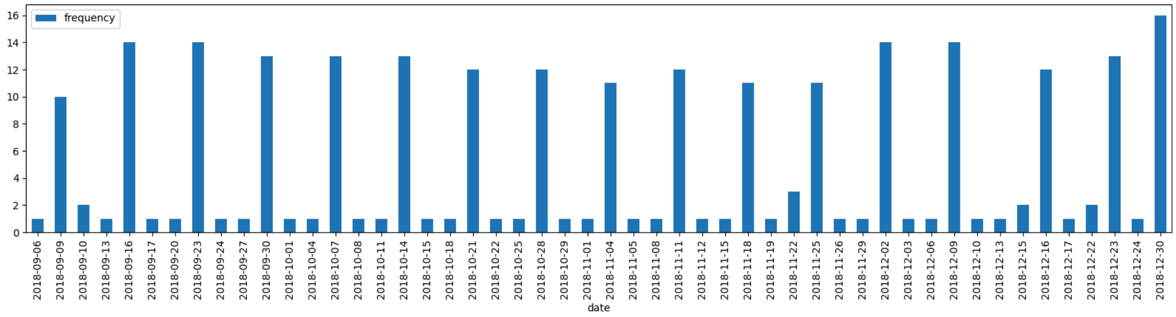
Let us plot the distribution using a bar plot.

In [43]:

```
# Plotting a bar plot
sorted_date_dist.plot(kind='bar', figsize=(20,4))
```

Out[43]:

```
<Axes: xlabel='date'>
```

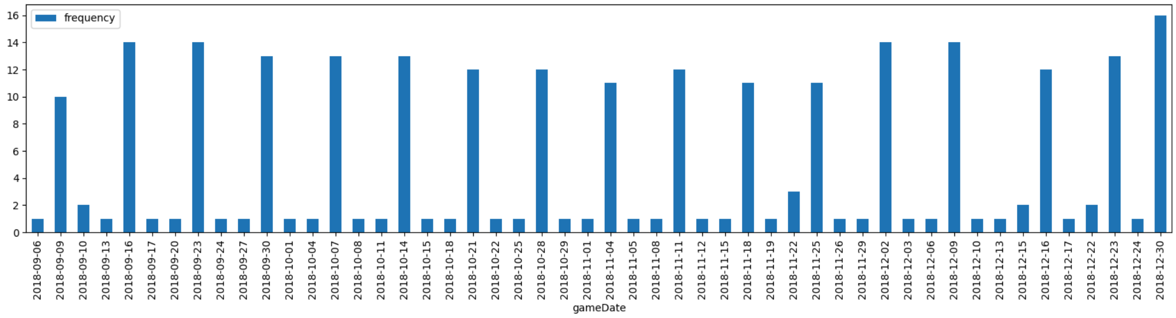


In [44]:

```
#We can do the same analysis for the time, day and week as well. So, let us co
```

In [45]:
```
def find_dist(df, col_name):

    # Checking the frequency of games in relation to the column values
    dist = df[col_name].value_counts().reset_index()

    # Renaming the columns
    dist.columns = [col_name, 'frequency']

    # Sorting the DataFrame based on the column values
    sorted_dist = dist.sort_values(col_name, ascending=True).set_index(col_nar

    # Plotting a bar plot
    sorted_dist.plot(kind='bar', figsize=(20,4))

    # Return a boolean indicating the function was successfully executed
    return True

# Visualizing the frequency distribution of games in relation to the date
find_dist(games_df, 'gameDate')
```

Out[45]:
```
True
```



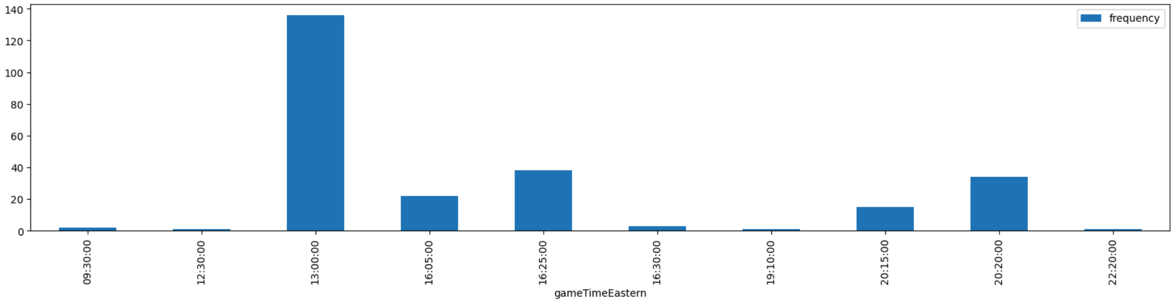
In [46]:
```
# Let us visualize the frequency distribution of games in relation to time anc
```

In [47]:
```
# Looking at the first five rows
games_df.head()
```

Out[47]:
```
        gameId    gameDate  gameTimeEastern  homeTeamAbbr  visitorTeamAbbr  week
0   2018090600  2018-09-06         20:20:00           PHI              ATL     1
1   2018090901  2018-09-09         13:00:00           CLE              PIT     1
2   2018090902  2018-09-09         13:00:00           IND              CIN     1
3   2018090903  2018-09-09         13:00:00           MIA              TEN     1
4   2018090900  2018-09-09         13:00:00           BAL              BUF     1
```

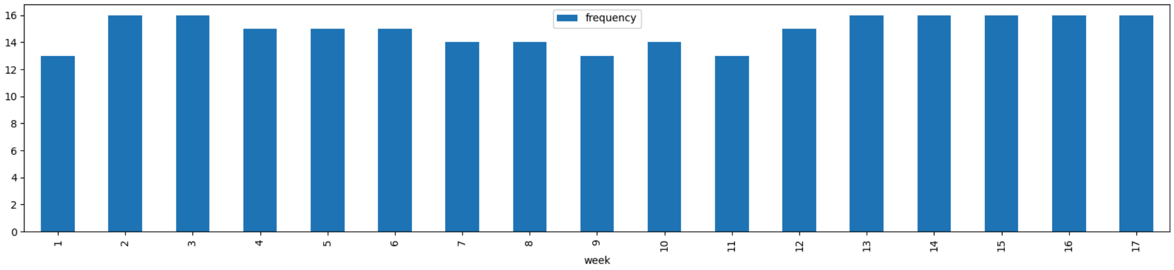
In [48]:
```
# Visualizing frequency distribution of games in relation to the time
find_dist(games_df, 'gameTimeEastern')
```

Out[48]:
```
True
```



In [49]:
```
# Visualizing frequency distribution of games in relation to the week
find_dist(games_df, 'week')
```

Out[49]:
```
True
```



Finally, let us look at how the games are distributed in relation to the game days. For this, we will have to convert the dates to which day they fall in the week.

In [50]:
```
# Looking at the first five rows
games_df.head()
```

Out[50]:

	gameId	gameDate	gameTimeEastern	homeTeamAbbr	visitorTeamAbbr	week
0	2018090600	2018-09-06	20:20:00	PHI	ATL	1
1	2018090901	2018-09-09	13:00:00	CLE	PIT	1
2	2018090902	2018-09-09	13:00:00	IND	CIN	1
3	2018090903	2018-09-09	13:00:00	MIA	TEN	1
4	2018090900	2018-09-09	13:00:00	BAL	BUF	1

In [51]:

```
# Creating a column containing the day of the week information extracted from
games_df['gameDay'] = games_df['gameDate'].apply(lambda x: x.strftime('%A'))

# Looking at the first five rows
games_df.head()
```

Out[51]:

	gameId	gameDate	gameTimeEastern	homeTeamAbbr	visitorTeamAbbr	week	gameDay
0	2018090600	2018-09-06	20:20:00	PHI	ATL	1	Thursday
1	2018090901	2018-09-09	13:00:00	CLE	PIT	1	Sunday
2	2018090902	2018-09-09	13:00:00	IND	CIN	1	Sunday
3	2018090903	2018-09-09	13:00:00	MIA	TEN	1	Sunday
4	2018090900	2018-09-09	13:00:00	BAL	BUF	1	Sunday

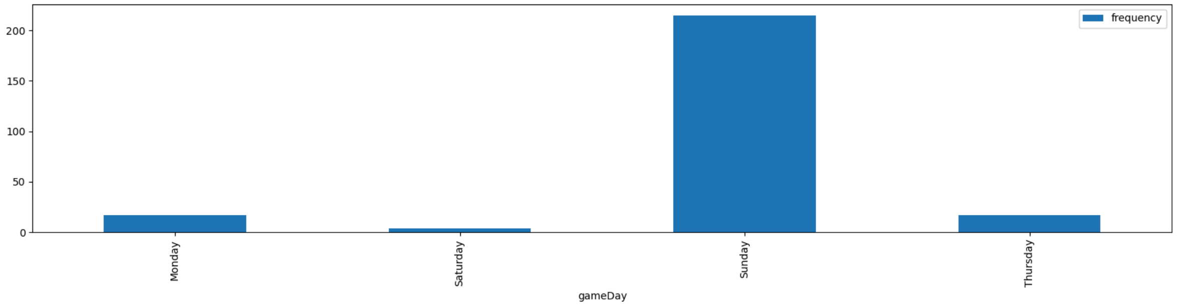
Visualizing the game distribution in relation to the game day.

In [52]:

```
# Visualizing frequency distribution of games in relation to the day of the we
find_dist(games_df, 'gameDay')
```

Out[52]:

```
True
```



2.Knowing the NFL players

In [53]:

```
import seaborn as sns
import datetime
```

Next, importing the CSV file called players.csv which contains information about the NFL players.

In [56]:

```
# Reading in the CSV file as a DataFrame
players_df = pd.read_csv("C:/Users/fai/Downloads/players.csv")
```

In [57]:

```
# Looking at the first five rows
players_df.head()
```

Out[57]:

	nflId	height	weight	birthDate	collegeName	position	displayName
0	2539334	72	190	1990-09-10	Washington	CB	Desmond Trufant
1	2539653	70	186	1988-11-01	Southeastern Louisiana	CB	Robert Alford
2	2543850	69	186	1991-12-18	Purdue	SS	Ricardo Allen
3	2555162	73	227	1994-11-04	Louisiana State	MLB	Deion Jones
4	2555255	75	232	1993-07-01	Minnesota	OLB	De'Vondre Campbell

Let us also view the shape of the DataFrame to know how many players are present in the dataset.

In [59]:

```
# Viewing the shape of the DataFrame
players_df.shape
```

Out[59]:

```
(1303, 7)
```

Before we begin, let us convert the date columns to Pandas datetime values.

In [61]:
```
# Converting to datetime.date values
players_df['birthDate'] = pd.to_datetime(players_df['birthDate']).dt.date

# Extracting the year
players_df['birthYear'] = pd.to_datetime(players_df['birthDate']).dt.year

# Looking at the first five rows
players_df.head()
```

Out[61]:

	nflId	height	weight	birthDate	collegeName	position	displayName	birthYear
0	2539334	72	190	1990-09-10	Washington	CB	Desmond Trufant	1990
1	2539653	70	186	1988-11-01	Southeastern Louisiana	CB	Robert Alford	1988
2	2543850	69	186	1991-12-18	Purdue	SS	Ricardo Allen	1991
3	2555162	73	227	1994-11-04	Louisiana State	MLB	Deion Jones	1994
4	2555255	75	232	1993-07-01	Minnesota	OLB	De'Vondre Campbell	1993

Let us start our analysis by finding the age distribution of the NFL players. For this, we will have to find the age of the players in respect to the year 2018.

In [63]:
```
# Finding the age of the players
players_df['age'] = 2018 - players_df['birthYear']

# Looking at the first five rows
players_df.head()
```

Out[63]:

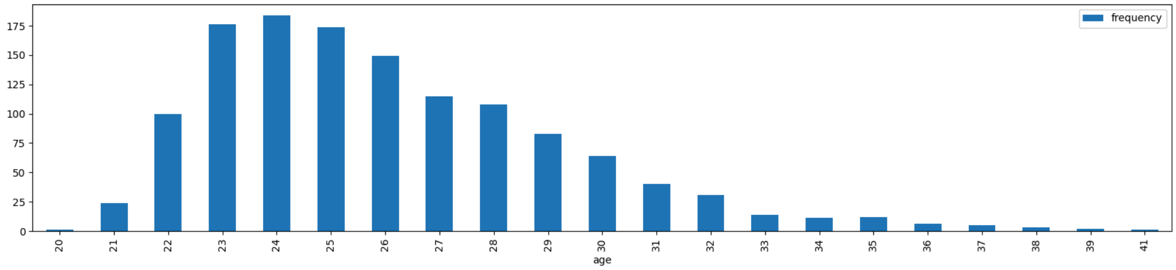
	nflId	height	weight	birthDate	collegeName	position	displayName	birthYear	age
0	2539334	72	190	1990-09-10	Washington	CB	Desmond Trufant	1990	28
1	2539653	70	186	1988-11-01	Southeastern Louisiana	CB	Robert Alford	1988	30
2	2543850	69	186	1991-12-18	Purdue	SS	Ricardo Allen	1991	27
3	2555162	73	227	1994-11-04	Louisiana State	MLB	Deion Jones	1994	24
4	2555255	75	232	1993-07-01	Minnesota	OLB	De'Vondre Campbell	1993	25

Since, we have the function we made in the previous section, we can use it to find the age distribution of the players easily.

In [65]:
```
def find_dist(df, col_name):

    # Checking the frequency of games in relation to the column values
    dist = df[col_name].value_counts().reset_index()

    # Renaming the columns
    dist.columns = [col_name, 'frequency']

    # Sorting the DataFrame based on the column values
    sorted_dist = dist.sort_values(col_name, ascending=True).set_index(col_nar

    # Plotting a bar plot
    sorted_dist.plot(kind='bar', figsize=(20,4))

    # Return a boolean indicating the function was successfully executed
    return True
```

In [66]:
```
# Visualizing frequency distribution of players in relation to their age
find_dist(players_df, 'age')
```

Out[66]:
```
True
```



Next, let us also see how the players are distributed amongst different team positions.

In [67]:

```
# Looking at the first five rows
players_df.head()
```

Out[67]:

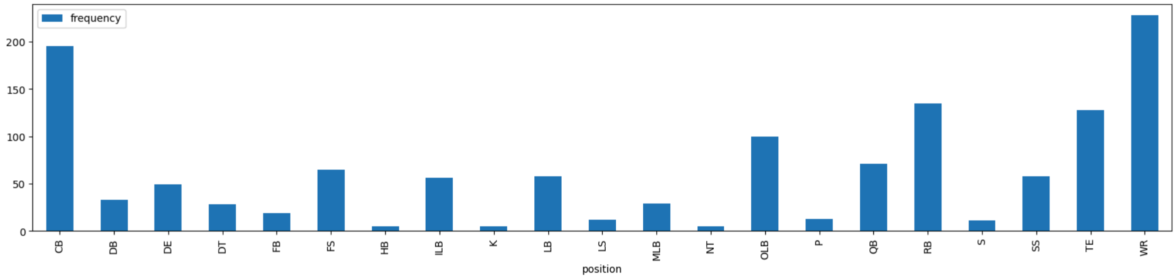
	nflId	height	weight	birthDate	collegeName	position	displayName	birthYear	age
0	2539334	72	190	1990-09-10	Washington	CB	Desmond Trufant	1990	28
1	2539653	70	186	1988-11-01	Southeastern Louisiana	CB	Robert Alford	1988	30
2	2543850	69	186	1991-12-18	Purdue	SS	Ricardo Allen	1991	27
3	2555162	73	227	1994-11-04	Louisiana State	MLB	Deion Jones	1994	24
4	2555255	75	232	1993-07-01	Minnesota	OLB	De'Vondre Campbell	1993	25

In [68]:

```
# Visualizing frequency distribution of players in relation to their positions
find_dist(players_df, 'position')
```

Out[68]:

```
True
```



Now, let us look at how the age distribution of players in the CB (Cornerback) and WR (Wide Receiver) positions.

For this, we can select the data points for either of the positions and then, find their age distribution.

In [69]:
```
# Selecting position = CB
players_df.query('position == "CB"')
```

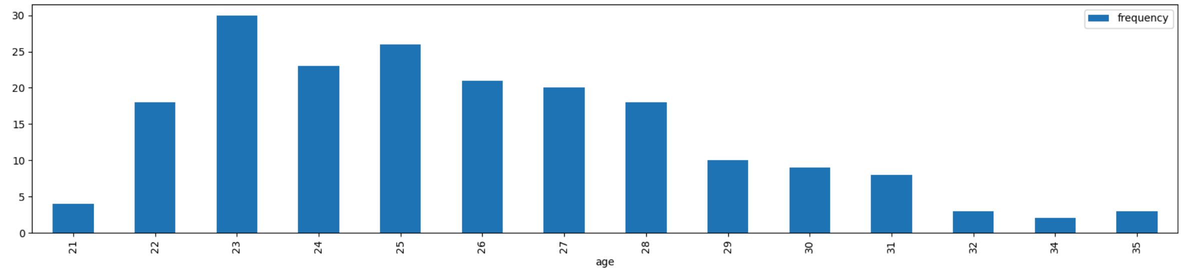
Out[69]:

	nflId	height	weight	birthDate	collegeName	position	displayName	birthYear	age
0	2539334	72	190	1990-09-10	Washington	CB	Desmond Trufant	1990	28
1	2539653	70	186	1988-11-01	Southeastern Louisiana	CB	Robert Alford	1988	30
6	2556445	70	211	1992-10-20	Florida	CB	Brian Poole	1992	26
28	2552689	71	193	1994-01-02	Florida State	CB	Ronald Darby	1994	24
29	2555383	72	191	1994-04-06	Louisiana State	CB	Jalen Mills	1994	24
...	...	...	...	...	...	...	...	...	...
1228	2561316	70	185	1996-02-02	Utah State	CB	Jalen Davis	1996	22
1241	2556371	72	200	1992-09-09	Texas A&M	CB	Brandon Williams	1992	26
1248	2558858	5-11	195	1994-06-25	Auburn	CB	Joshua Holsey	1994	24
1278	2558819	6-2	188	1994-12-04	Mississippi	CB	Derrick Jones	1994	24
1301	2561469	69	187	1993-03-15	Alabama-Birmingham	CB	Darious Williams	1993	25

195 rows × 9 columns

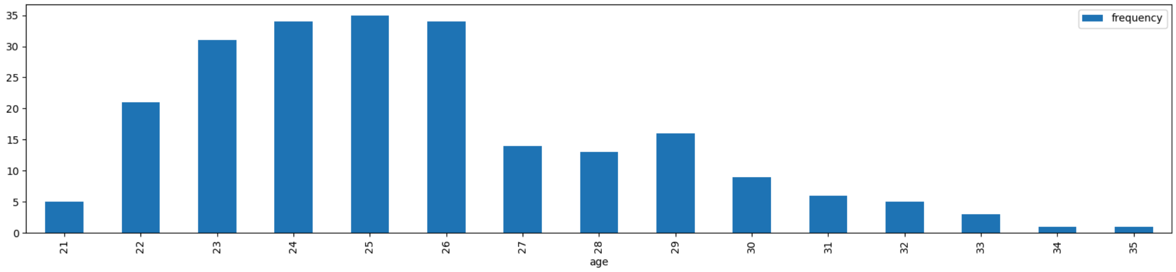
In [70]:
```
# Visualizing frequency distribution of players in relation to the CB positio
find_dist(players_df.query('position == "CB"'), 'age')
```

Out[70]:
```
True
```



In [71]:
```
# Visualizing frequency distribution of players in relation to the WR positior
find_dist(players_df.query('position == "WR"'), 'age')
```

Out[71]:
```
True
```



Now, let us look at the actual height and weight distribution of the players. However, their is some inconsistency in the data in the height column.

```
In [72]: # Looking at the first twenty rows
         players_df.head(20)
```

Out[72]:

	nflId	height	weight	birthDate	collegeName	position	displayName	birthYear	age
0	2539334	72	190	1990-09-10	Washington	CB	Desmond Trufant	1990	28
1	2539653	70	186	1988-11-01	Southeastern Louisiana	CB	Robert Alford	1988	30
2	2543850	69	186	1991-12-18	Purdue	SS	Ricardo Allen	1991	27
3	2555162	73	227	1994-11-04	Louisiana State	MLB	Deion Jones	1994	24
4	2555255	75	232	1993-07-01	Minnesota	OLB	De'Vondre Campbell	1993	25
5	2555543	73	216	1995-07-26	Florida	FS	Keanu Neal	1995	23
6	2556445	70	211	1992-10-20	Florida	CB	Brian Poole	1992	26
7	2507763	6-0	200	1986-08-01	Mississippi	WR	Mike Wallace	1986	32
8	2532842	78	243	1989-01-20	Arizona	QB	Nick Foles	1989	29
9	2540158	77	250	1990-11-10	Stanford	TE	Zach Ertz	1990	28
10	2552582	6-0	223	1993-06-15	Boise State	RB	Jay Ajayi	1993	25
11	2552600	72	198	1993-05-24	Southern California	WR	Nelson Agholor	1993	25
12	2553502	68	190	1993-04-10	Sacramento State	WR	DeAndre Carter	1993	25
13	2552301	75	246	1992-07-08	Clemson	DE	Vic Beasley	1992	26
14	2506467	66	190	1983-06-20	Kansas State	RB	Darren Sproles	1983	35
15	2557967	73	230	1994-08-09	Louisiana State	OLB	Duke Riley	1994	24
16	2558184	71	190	1993-06-05	San Diego State	SS	Damontae Kazee	1993	25
17	2560995	77	256	1995-01-03	South Dakota State	TE	Dallas Goedert	1995	23
18	2559150	70	220	1994-11-02	Wisconsin	RB	Corey Clement	1994	24
19	310	76	217	1985-05-17	Boston College	QB	Matt Ryan	1985	33

Let us fix it by converting all datapoints to inches.

```
In [73]: # Fixing the inconsistency by converting all data to inches
         players_df['height'] = players_df['height'].apply(lambda x: int(x[0])*12 + int

         # Looking at the first twenty rows
         players_df.head(20)
```

Out[73]:

	nflId	height	weight	birthDate	collegeName	position	displayName	birthYear	age
0	2539334	72	190	1990-09-10	Washington	CB	Desmond Trufant	1990	28
1	2539653	70	186	1988-11-01	Southeastern Louisiana	CB	Robert Alford	1988	30
2	2543850	69	186	1991-12-18	Purdue	SS	Ricardo Allen	1991	27
3	2555162	73	227	1994-11-04	Louisiana State	MLB	Deion Jones	1994	24
4	2555255	75	232	1993-07-01	Minnesota	OLB	De'Vondre Campbell	1993	25
5	2555543	73	216	1995-07-26	Florida	FS	Keanu Neal	1995	23
6	2556445	70	211	1992-10-20	Florida	CB	Brian Poole	1992	26
7	2507763	72	200	1986-08-01	Mississippi	WR	Mike Wallace	1986	32
8	2532842	78	243	1989-01-20	Arizona	QB	Nick Foles	1989	29
9	2540158	77	250	1990-11-10	Stanford	TE	Zach Ertz	1990	28
10	2552582	72	223	1993-06-15	Boise State	RB	Jay Ajayi	1993	25
11	2552600	72	198	1993-05-24	Southern California	WR	Nelson Agholor	1993	25
12	2553502	68	190	1993-04-10	Sacramento State	WR	DeAndre Carter	1993	25
13	2552301	75	246	1992-07-08	Clemson	DE	Vic Beasley	1992	26
14	2506467	66	190	1983-06-20	Kansas State	RB	Darren Sproles	1983	35
15	2557967	73	230	1994-08-09	Louisiana State	OLB	Duke Riley	1994	24
16	2558184	71	190	1993-06-05	San Diego State	SS	Damontae Kazee	1993	25
17	2560995	77	256	1995-01-03	South Dakota State	TE	Dallas Goedert	1995	23
18	2559150	70	220	1994-11-02	Wisconsin	RB	Corey Clement	1994	24
19	310	76	217	1985-05-17	Boston College	QB	Matt Ryan	1985	33

Now, instead of looking at the height and weight distribution of players seperately, let us look at them together by making a joint plot.

In [77]:

```
# Extracting the height values
players_df['height'].values
```

Out[77]:

```
array([72, 70, 69, ..., 78, 69, 74], dtype=int64)
```

In [78]:

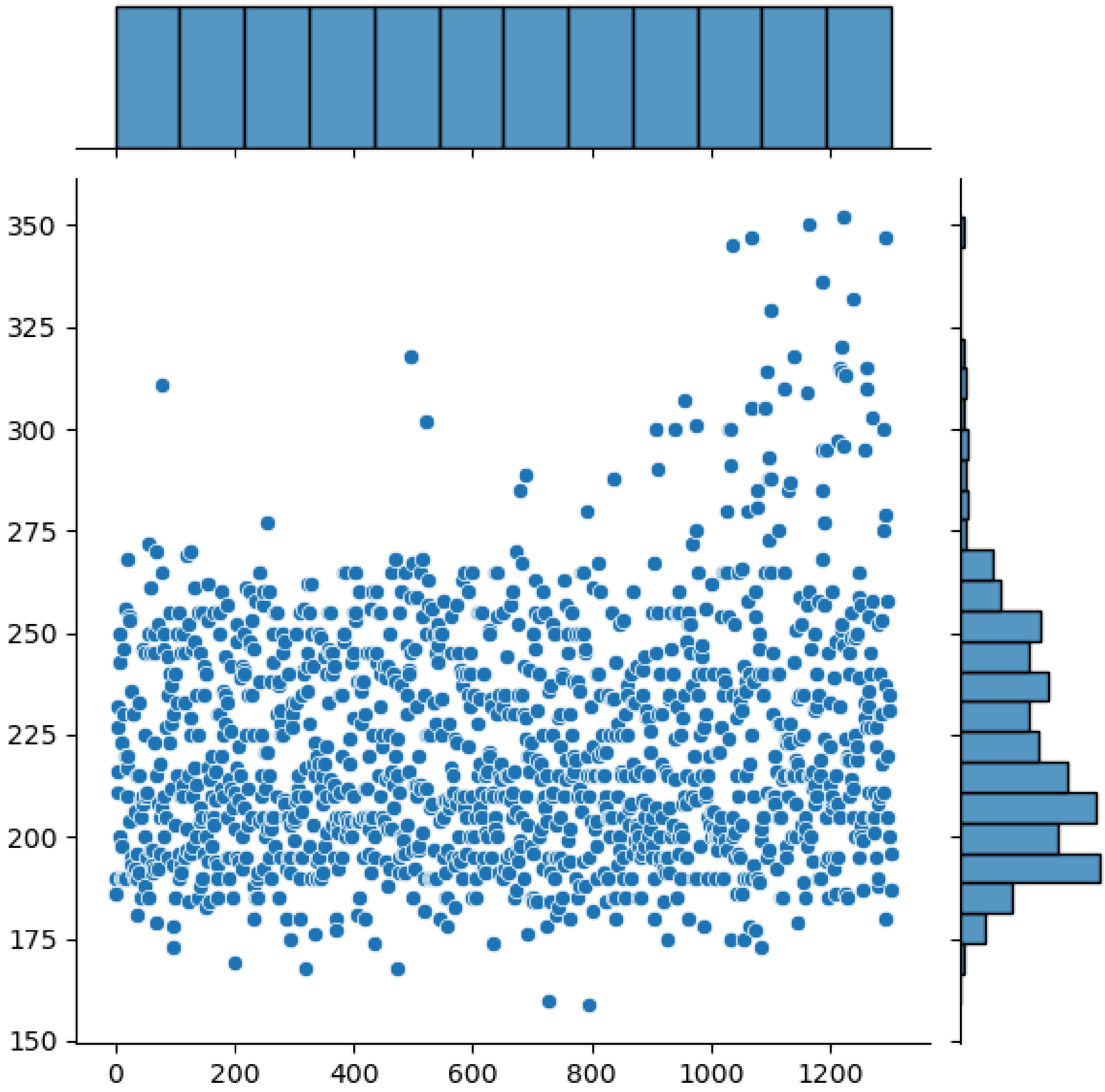
```
# Assigning the height and weight values
height = players_df['height'].values
weight = players_df['weight'].values
```

In [80]:

```
# Plotting a joint plot
sns.jointplot(weight)
```

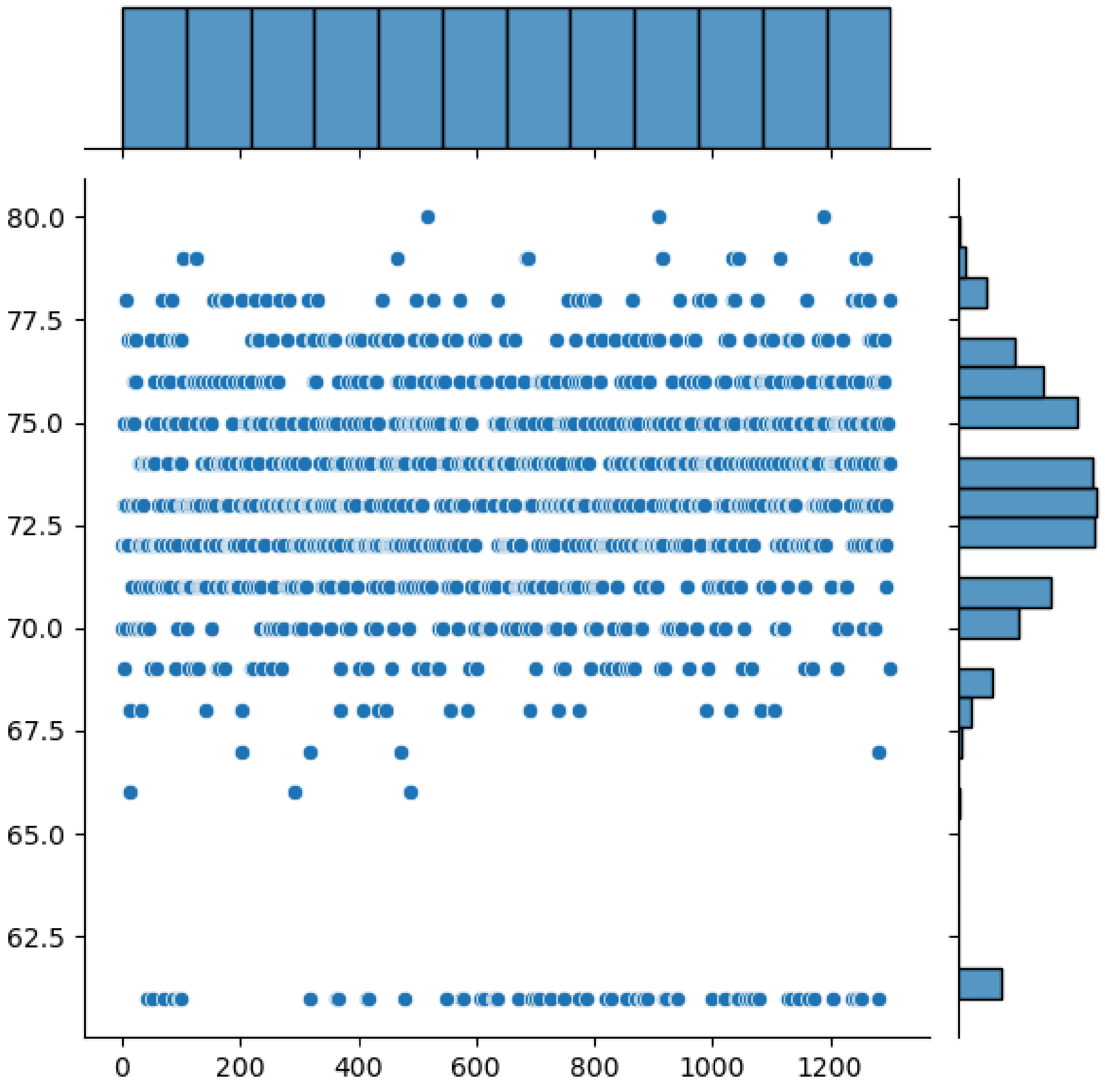
Out[80]:

```
<seaborn.axisgrid.JointGrid at 0x23594281190>
```



In [81]:
```
# Plotting a joint plot
sns.jointplot(height)
```

Out[81]:
```
<seaborn.axisgrid.JointGrid at 0x2359431bb10>
```



3.Understanding the NFL plays

In [82]:
```
# Reading in the CSV file as a DataFrame
plays_df = pd.read_csv("C:/Users/fai/Downloads/plays.csv")
```

In [83]:

```
# Looking at the first five rows
plays_df.head()
```

Out[83]:

	gameId	playId	playDescription	quarter	down	yardsToGo	possessionTeam	playTy
0	2018090600	75	(15:00) M.Ryan pass short right to J.Jones pus...	1	1	15	ATL	play_type_p
1	2018090600	146	(13:10) M.Ryan pass incomplete short right to ...	1	1	10	ATL	play_type_p
2	2018090600	168	(13:05) (Shotgun) M.Ryan pass incomplete short...	1	2	10	ATL	play_type_p
3	2018090600	190	(13:01) (Shotgun) M.Ryan pass deep left to J.J...	1	3	10	ATL	play_type_p
4	2018090600	256	(10:59) (Shotgun) M.Ryan pass incomplete short...	1	3	1	ATL	play_type_p

5 rows × 27 columns

In [84]:

```
plays_df.shape
```

Out[84]:

```
(19239, 27)
```

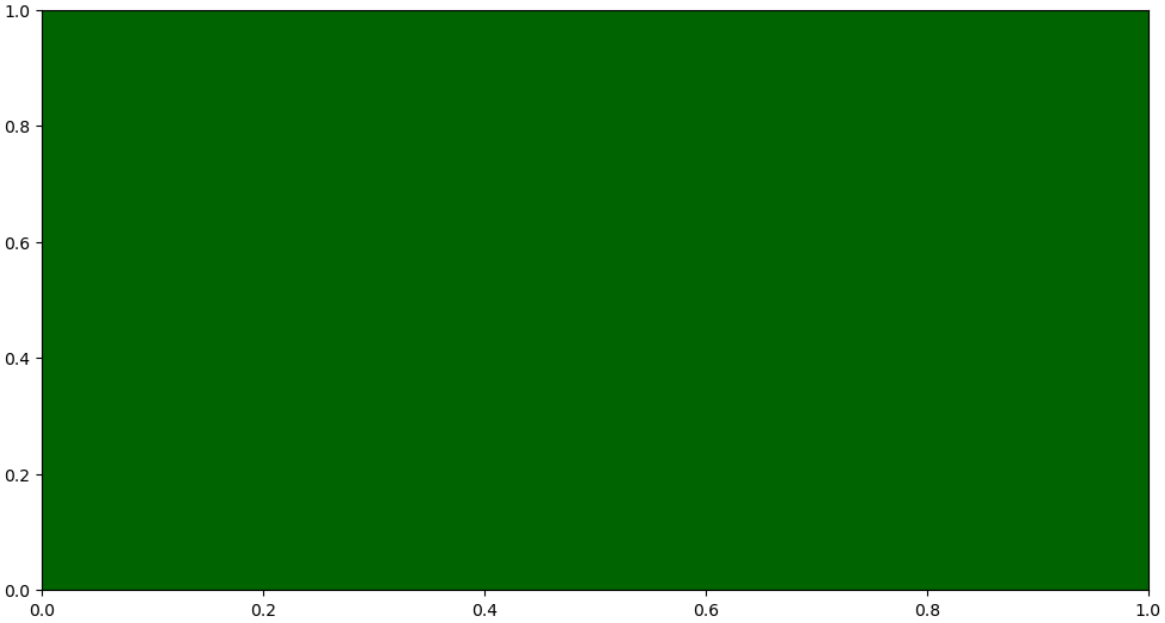
4. Visualizing the American Football Field

In [85]:

```
import matplotlib.patches as patches
```

```
In [86]: # Create a rectangle defined via an anchor point *xy* and its *width* and *hei
         rect = patches.Rectangle((0, 0), 120, 53.3, facecolor='darkgreen', zorder=0)

         # Creating a subplot to plot our field on
         fig, ax = plt.subplots(1, figsize=(12, 6.33))

         # Adding the rectangle to the plot
         ax.add_patch(rect)
```

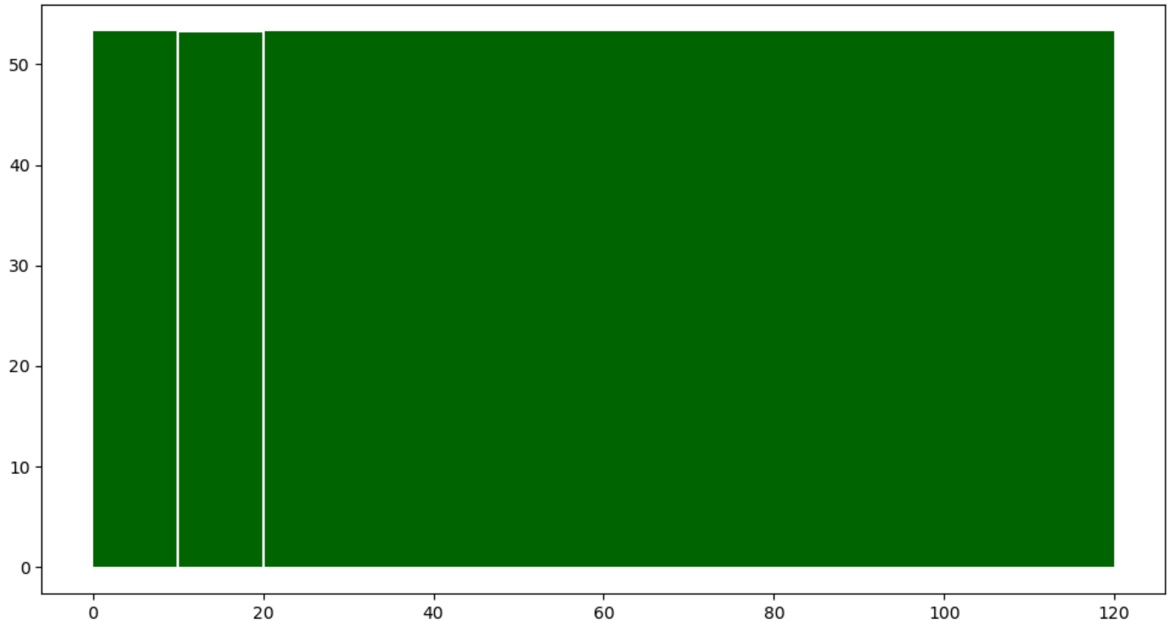
```
Out[86]: <matplotlib.patches.Rectangle at 0x23594e0a4d0>
```



Let us add a line plot to create some lines on the field by using the plot() method.

In [87]:
```
# Create a rectangle defined via an anchor point *xy* and its *width* and *hei
rect = patches.Rectangle((0, 0), 120, 53.3, facecolor='darkgreen', zorder=0)

# Creating a subplot to plot our field on
fig, ax = plt.subplots(1, figsize=(12, 6.33))

# Adding the rectangle to the plot
ax.add_patch(rect)

# Plotting a line plot for marking the field lines
plt.plot([10, 10, 20, 20],
         [0, 53.3, 53.3, 0],
         color='white', zorder=0)
```

Out[87]:
```
[<matplotlib.lines.Line2D at 0x2359533a950>]
```



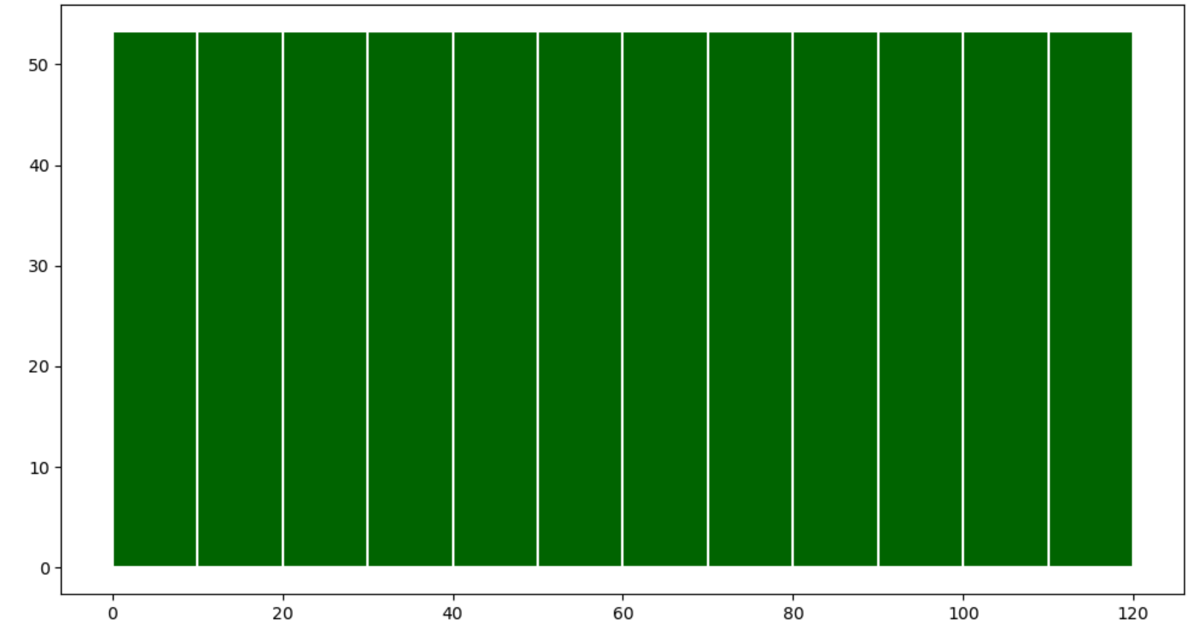
Now let us create all the lines on the field.

In [90]:

```
# Create a rectangle defined via an anchor point *xy* and its *width* and *hei
rect = patches.Rectangle((0, 0), 120, 53.3, facecolor='darkgreen', zorder=0)

# Creating a subplot to plot our field on
fig, ax = plt.subplots(1, figsize=(12, 6.33))

# Adding the rectangle to the plot
ax.add_patch(rect)

# Plotting a line plot for marking the field lines
plt.plot([10, 10, 20, 20, 30, 30, 40, 40, 50, 50, 60, 60, 70, 70, 80,
          80, 90, 90, 100, 100, 110, 110, 120, 0, 0, 120, 120],
         [0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3,
          0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 53.3, 0, 0, 53.3],
         color='white', zorder = 0)
```

Out[90]:

```
[<matplotlib.lines.Line2D at 0x235953e6590>]
```



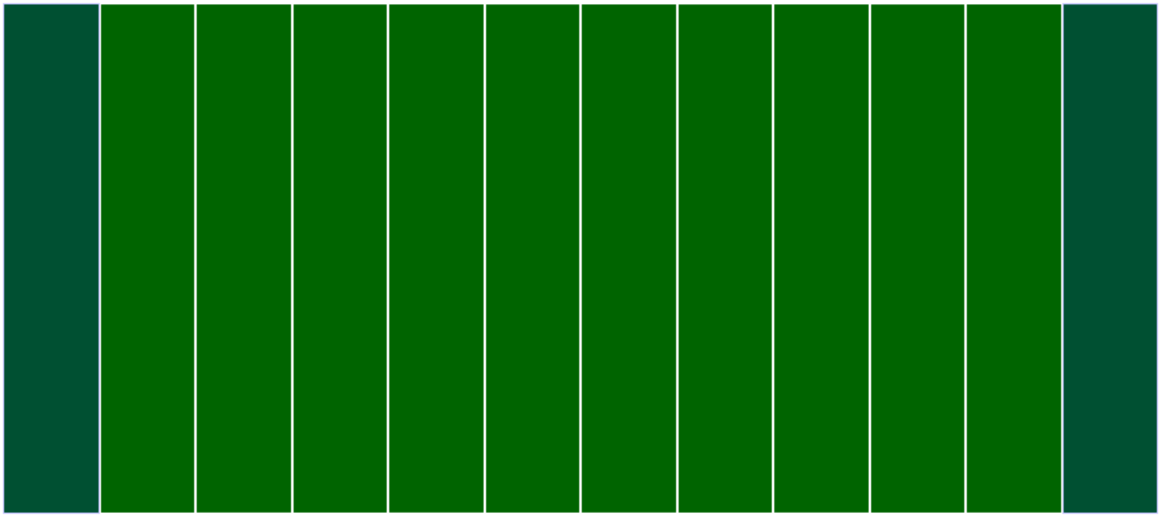
Now, let us add the endzones onto the plot.

In [91]:

```
# Create a rectangle defined via an anchor point *xy* and its *width* and *hei
rect = patches.Rectangle((0, 0), 120, 53.3, facecolor='darkgreen', zorder=0)

# Creating a subplot to plot our field on
fig, ax = plt.subplots(1, figsize=(12, 6.33))

# Adding the rectangle to the plot
ax.add_patch(rect)

# Plotting a line plot for marking the field lines
plt.plot([10, 10, 20, 20, 30, 30, 40, 40, 50, 50, 60, 60, 70, 70, 80,
          80, 90, 90, 100, 100, 110, 110, 120, 0, 0, 120, 120],
         [0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3,
          0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 53.3, 0, 0, 53.3],
         color='white', zorder = 0)

# Creating the left end-zone
left_end_zone = patches.Rectangle((0, 0), 10, 53.3, facecolor='blue', alpha=0.

# Creating the right end-zone
right_end_zone = patches.Rectangle((110, 0), 120, 53.3, facecolor='blue', alph
# Adding the patches to the subplot
ax.add_patch(left_end_zone)
ax.add_patch(right_end_zone)

# Setting the limits of x-axis from 0 to 120
plt.xlim(0, 120)

# Setting the limits of y-axis from -5 to 58.3
plt.ylim(-5, 58.3)

# Removing the axis values from the plot
plt.axis('off')
```

Out[91]:

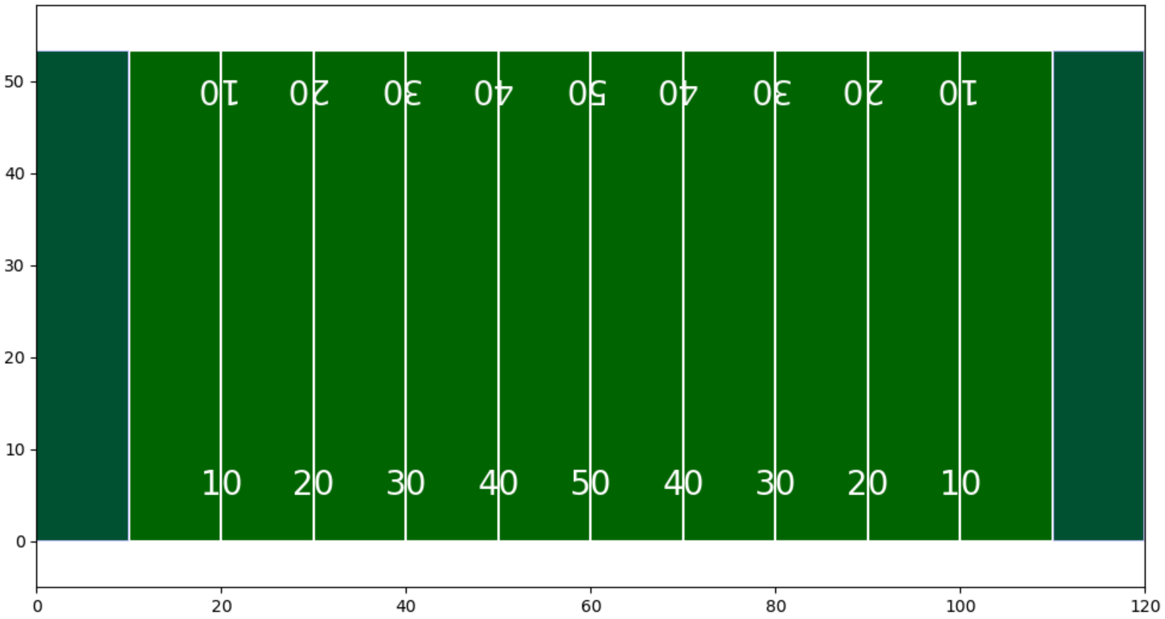
```
(0.0, 120.0, -5.0, 58.3)
```



It is time for us to plot the numbers on the field.

In [92]:

```
# Create a rectangle defined via an anchor point *xy* and its *width* and *hei
rect = patches.Rectangle((0, 0), 120, 53.3, facecolor='darkgreen', zorder=0)

# Creating a subplot to plot our field on
fig, ax = plt.subplots(1, figsize=(12, 6.33))

# Adding the rectangle to the plot
ax.add_patch(rect)

# Plotting a line plot for marking the field lines
plt.plot([10, 10, 20, 20, 30, 30, 40, 40, 50, 50, 60, 60, 70, 70, 80,
          80, 90, 90, 100, 100, 110, 110, 120, 0, 0, 120, 120],
         [0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3,
          0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 53.3, 0, 0, 53.3],
         color='white', zorder = 0)

# Creating the left end-zone
left_end_zone = patches.Rectangle((0, 0), 10, 53.3, facecolor='blue', alpha=0.

# Creating the right end-zone
right_end_zone = patches.Rectangle((110, 0), 120, 53.3, facecolor='blue', alph
# Adding the patches to the subplot
ax.add_patch(left_end_zone)
ax.add_patch(right_end_zone)

# Setting the limits of x-axis from 0 to 120
plt.xlim(0, 120)

# Setting the limits of y-axis from -5 to 58.3
plt.ylim(-5, 58.3)

# Removing the axis values from the plot
# plt.axis('off')

# Plotting the numbers starting from x = 20 and ending at x = 110
# with a step of 10
for x in range(20, 110, 10):

    # Intializing another variable named 'number'
    number = x

    # If x exceeds 50, subtract it from 120
    if x > 50:
        number = 120 - x

    # Plotting the text at the bottom
    plt.text(x, 5, str(number - 10),
             horizontalalignment='center',
             fontsize=20,
             color='white')

    # Plotting the text at the top
    plt.text(x - 0.95, 53.3 - 5, str(number - 10),
             horizontalalignment='center',
             fontsize=20,
             color='white',
             rotation=180)
```

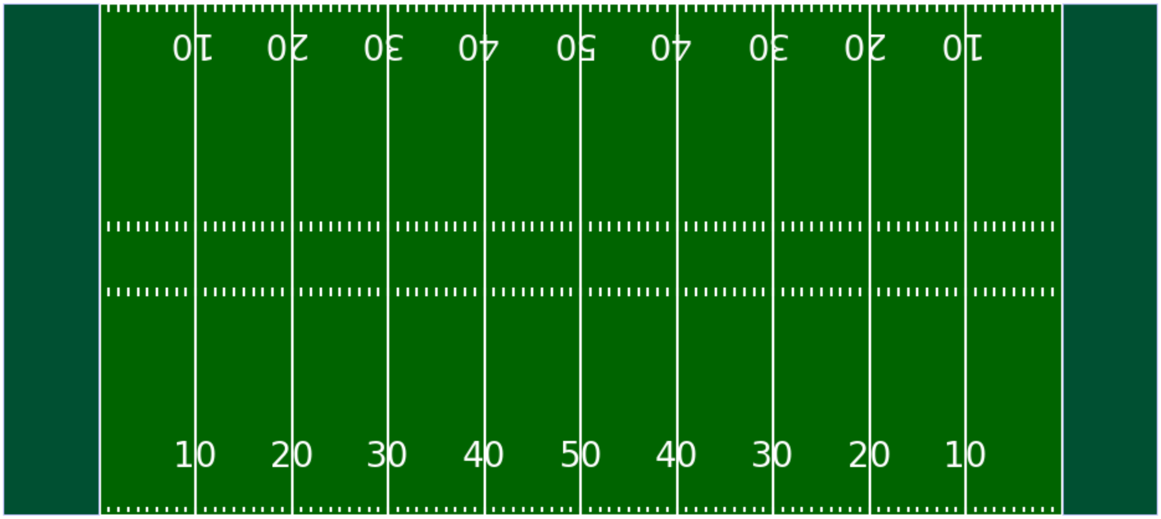

10
20
30
40
50
40
30
20
10
0
10
20
30
40
50
0
20
40
60
80
100
120

Let us finally create the gound markings and complete the plot.

In [93]:

```
# Create a rectangle defined via an anchor point *xy* and its *width* and *hei
rect = patches.Rectangle((0, 0), 120, 53.3, facecolor='darkgreen', zorder=0)

# Creating a subplot to plot our field on
fig, ax = plt.subplots(1, figsize=(12, 6.33))

# Adding the rectangle to the plot
ax.add_patch(rect)

# Plotting a line plot for marking the field lines
plt.plot([10, 10, 20, 20, 30, 30, 40, 40, 50, 50, 60, 60, 70, 70, 80,
          80, 90, 90, 100, 100, 110, 110, 120, 0, 0, 120, 120],
         [0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3
          0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 53.3, 0, 0, 53.3],
         color='white', zorder = 0)

# Creating the left end-zone
left_end_zone = patches.Rectangle((0, 0), 10, 53.3, facecolor='blue', alpha=0

# Creating the right end-zone
right_end_zone = patches.Rectangle((110, 0), 120, 53.3, facecolor='blue', alph
# Adding the patches to the subplot
ax.add_patch(left_end_zone)
ax.add_patch(right_end_zone)

# Setting the limits of x-axis from 0 to 120
plt.xlim(0, 120)

# Setting the limits of y-axis from -5 to 58.3
plt.ylim(-5, 58.3)

# Removing the axis values from the plot
plt.axis('off')

# Plotting the numbers starting from x = 20 and ending at x = 110
# with a step of 10
for x in range(20, 110, 10):

    # Intializing another variable named 'number'
    number = x
    # If x exceeds 50, subtract it from 120
    if x > 50:
        number = 120 - x

    # Plotting the text at the bottom
    plt.text(x, 5, str(number - 10),
             horizontalalignment='center',
             fontsize=20,
             color='white')

    # Plotting the text at the top
    plt.text(x - 0.95, 53.3 - 5, str(number - 10),
             horizontalalignment='center',
             fontsize=20,
             color='white',
             rotation=180)
```

```
# Making ground markings
for x in range(11, 110):
        ax.plot([x, x], [0.4, 0.7], color='white', zorder = 0)
        ax.plot([x, x], [53.0, 52.5], color='white', zorder = 0)
        ax.plot([x, x], [22.91, 23.57], color='white', zorder = 0)
        ax.plot([x, x], [29.73, 30.39], color='white', zorder = 0)
```

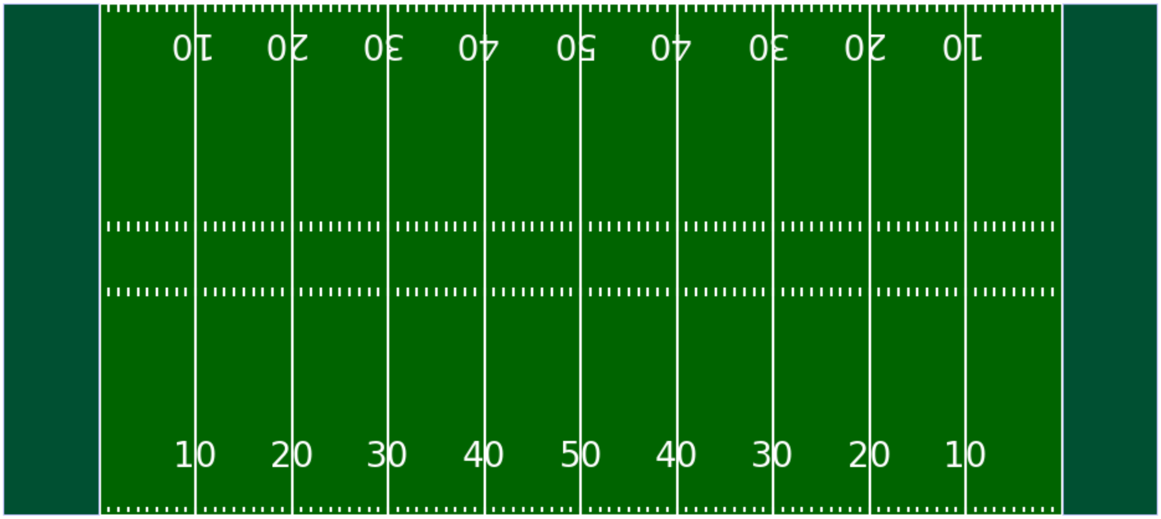


Wrapping the entire code in a function for easy plotting

In [99]:
```
def create_football_field():

    # Create a rectangle defined via an anchor point *xy* and its *width* and
    rect = patches.Rectangle((0, 0), 120, 53.3, facecolor='darkgreen', zorder=

    # Creating a subplot to plot our field on
    fig, ax = plt.subplots(1, figsize=(12, 6.33))

    # Adding the rectangle to the plot
    ax.add_patch(rect)

    # Plotting a line plot for marking the field lines
    plt.plot([10, 10, 20, 20, 30, 30, 40, 40, 50, 50, 60, 60, 70, 70, 80,
              80, 90, 90, 100, 100, 110, 110, 120, 0, 0, 120, 120],
             [0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 5
              0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 53.3, 0, 0, 53.3],
             color='white', zorder = 0)

    # Creating the left end-zone
    left_end_zone = patches.Rectangle((0, 0), 10, 53.3, facecolor='blue', alph
    # Creating the right end-zone
    right_end_zone = patches.Rectangle((110, 0), 120, 53.3, facecolor='blue',

    # Adding the patches to the subplot
    ax.add_patch(left_end_zone)
    ax.add_patch(right_end_zone)

    # Setting the limits of x-axis from 0 to 120
    plt.xlim(0, 120)

    # Setting the limits of y-axis from -5 to 58.3
    plt.ylim(-5, 58.3)

    # Removing the axis values from the plot
    plt.axis('off')

    # Plotting the numbers starting from x = 20 and ending at x = 110
    # with a step of 10
    for x in range(20, 110, 10):
       # Intializing another variable named 'number'
        number = x

        # If x exceeds 50, subtract it from 120
        if x > 50:
            number = 120 - x

        # Plotting the text at the bottom
        plt.text(x, 5, str(number - 10),
                 horizontalalignment='center',
                 fontsize=20,
                 color='white')

        # Plotting the text at the top
        plt.text(x - 0.95, 53.3 - 5, str(number - 10),
                 horizontalalignment='center',
                 fontsize=20,
                 color='white',
                 rotation=180)

    # Making ground markings
    for x in range(11, 110):
            ax.plot([x, x], [0.4, 0.7], color='white', zorder = 0)
            ax.plot([x, x], [53.0, 52.5], color='white', zorder = 0)
            ax.plot([x, x], [22.91, 23.57], color='white', zorder = 0)
            ax.plot([x, x], [29.73, 30.39], color='white', zorder = 0)

    # Returning the figure and axis
    return fig, ax
```

In [100]:

```
# Calling the plotting function
fig, ax = create_football_field()

# Plotting the figure
plt.show()
```



5. Adding Players onto the Field

In [106]:

```
# Reading the data as a Pandas DataFrame
df = pd.read_csv("C:/Users/fai/Downloads/week_data.csv")
```

In [107]:

```
# Looking at the first five rows of the DataFrame
df.head()
```

Out[107]:

	time	x	y	s	a	dis	o	dir	event	nflId	displayNan
0	2018-11-16T01:24:15.799Z	77.97	18.61	0.00	0.00	0.00	109.88	289.98	None	497236.0	Jimr Graha
1	2018-11-16T01:24:15.799Z	79.41	23.71	0.00	0.00	0.00	90.31	159.68	None	2506363.0	Aar Rodge
2	2018-11-16T01:24:15.799Z	85.05	22.71	0.00	0.00	0.00	288.53	141.92	None	2532966.0	Bob Wagn
3	2018-11-16T01:24:15.799Z	84.81	17.84	0.01	0.01	0.01	283.13	295.48	None	2539243.0	Bradl McDouga
4	2018-11-16T01:24:15.799Z	85.35	27.05	0.00	0.00	0.00	251.12	350.19	None	2540140.0	Barkevio Min

In [108]:

```
# Looking at the shape of the DataFrame
df.shape
```

Out[108]:

```
(932240, 19)
```

Since the time is in an improper format for analysis, let us convert it to datetime.

In [109]:

```
# Converting to Time values
df['time'] = pd.to_datetime(df['time']).dt.time

# Looking at the first five rows of the DataFrame
df.head()
```

Out[109]:

	time	x	y	s	a	dis	o	dir	event	nflId	displayName
0	01:24:15.799000	77.97	18.61	0.00	0.00	0.00	109.88	289.98	None	497236.0	Jimmy Grahan
1	01:24:15.799000	79.41	23.71	0.00	0.00	0.00	90.31	159.68	None	2506363.0	Aaror Rodgers
2	01:24:15.799000	85.05	22.71	0.00	0.00	0.00	288.53	141.92	None	2532966.0	Bobby Wagne
3	01:24:15.799000	84.81	17.84	0.01	0.01	0.01	283.13	295.48	None	2539243.0	Bradley McDougal
4	01:24:15.799000	85.35	27.05	0.00	0.00	0.00	251.12	350.19	None	2540140.0	Barkevious Ming

We would want to analyze each game by the passage of time, so let us sort the values to be ascending.

In [110]:

```
# Sorting the values of the DataFrame by time in an ascending order
df = df.sort_values(by='time', ascending=True).reset_index(drop=True)

# Looking at the first five rows of the DataFrame
df.head()
```

Out[110]:

	time	x	y	s	a	dis	o	dir	event	nflId	displayName
0	00:00:38.500000	86.80	44.97	0.01	0.01	0.01	265.98	318.85	None	2552600.0	Nelsor Agholo
1	00:00:38.500000	80.39	37.16	0.08	0.23	0.01	89.72	339.97	None	2552484.0	P.J. Williams
2	00:00:38.500000	86.34	36.31	0.00	0.00	0.00	285.81	269.36	None	497326.0	Golden Tate
3	00:00:38.500000	85.10	23.74	0.00	0.00	0.00	NaN	NaN	None	NaN	Footbal
4	00:00:38.500000	86.82	7.17	0.03	0.02	0.01	263.02	288.96	None	2560949.0	Josh Adams

Let us select a specific gameId and playID to visualize the player positions within a specific game and play.

In [111]:

```
# Selecting the data for the given game and play based on their Id
sel_df = df.query('gameId == 2018111900 and playId == 5577')

# Looking at the shape of the DataFrame
print(f'The shape of the DataFrame is: {sel_df.shape}')

# Looking at the DataFrame
sel_df
```

```
The shape of the DataFrame is: (1770, 19)
```

Out[111]:

	time	x	y	s	a	dis	o	dir	event	nflId	display
282518	04:57:48	84.26	23.74	0.00	0.00	0.00	NaN	NaN	None	NaN	F
282519	04:57:48	83.26	29.86	0.00	0.00	0.00	64.10	79.64	None	2558830.0	S E
282520	04:57:48	63.54	18.25	0.07	0.76	0.01	67.33	3.53	None	2558183.0	Jo
282521	04:57:48	88.42	23.54	0.11	0.04	0.01	272.66	325.11	None	2558125.0	Ma
282522	04:57:48	79.76	26.36	0.01	0.01	0.00	105.22	4.91	None	2556593.0	Cory L
...	...	...	...	...	...	...	...	...	...	...	
284283	04:57:59.700000	60.25	29.51	6.40	2.21	0.65	314.31	294.95	None	2540204.0	S
284284	04:57:59.700000	63.23	45.92	3.81	1.20	0.37	257.27	262.77	None	1037374.0	Sam S
284285	04:57:59.700000	52.42	49.74	3.74	1.48	0.38	12.75	290.79	None	2553536.0	T
284286	04:57:59.700000	84.26	28.11	1.73	0.54	0.17	260.29	249.13	None	2558830.0	S E
284287	04:57:59.700000	42.19	41.24	2.05	4.00	0.23	NaN	NaN	None	NaN	F

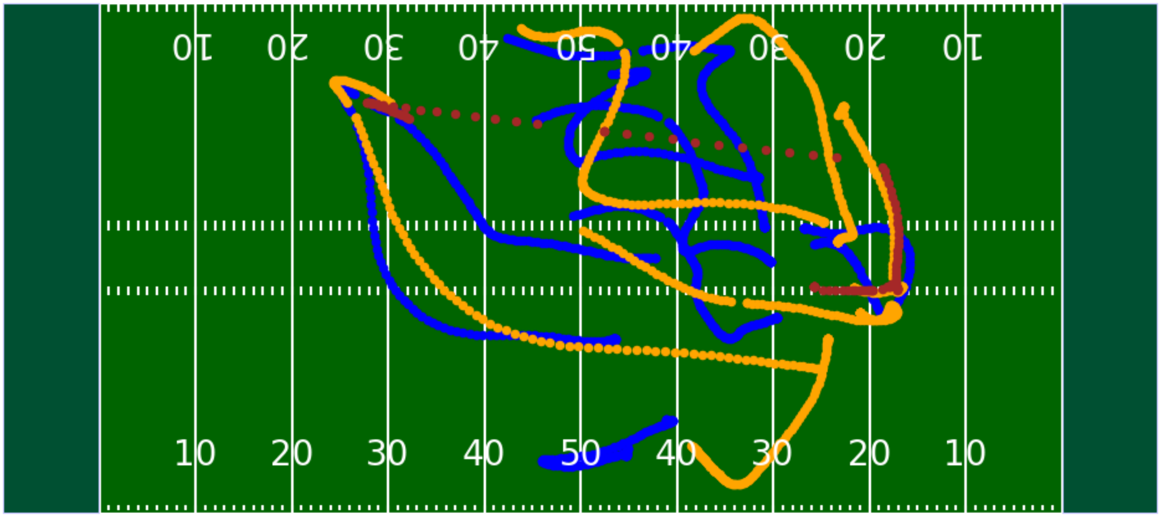
1770 rows × 19 columns

Now, let us seperate out the teams as well as the football in the data for plotting.

In [112]:

```
# Selecting the home and away team
home_team = sel_df.query('team == "home"')
away_team = sel_df.query('team == "away"')

# Selecting the football
football = sel_df.query('team == "football"')
```

In [113]:
```
# Creating the football field
fig, ax = create_football_field()

# Plotitng the home team
home_team.plot(x='x', y='y', kind='scatter', ax=ax, color='blue', s=20, zorder

# Plotting the away team
away_team.plot(x='x', y='y', kind='scatter', ax=ax, color='orange', s=20, zord

# Plotting the football
football.plot(x='x', y='y', kind='scatter', ax=ax, color='brown', s=20, zorder

# Displaying the plot
plt.show()
```



We can also visualize a specific event by just selecting the event.

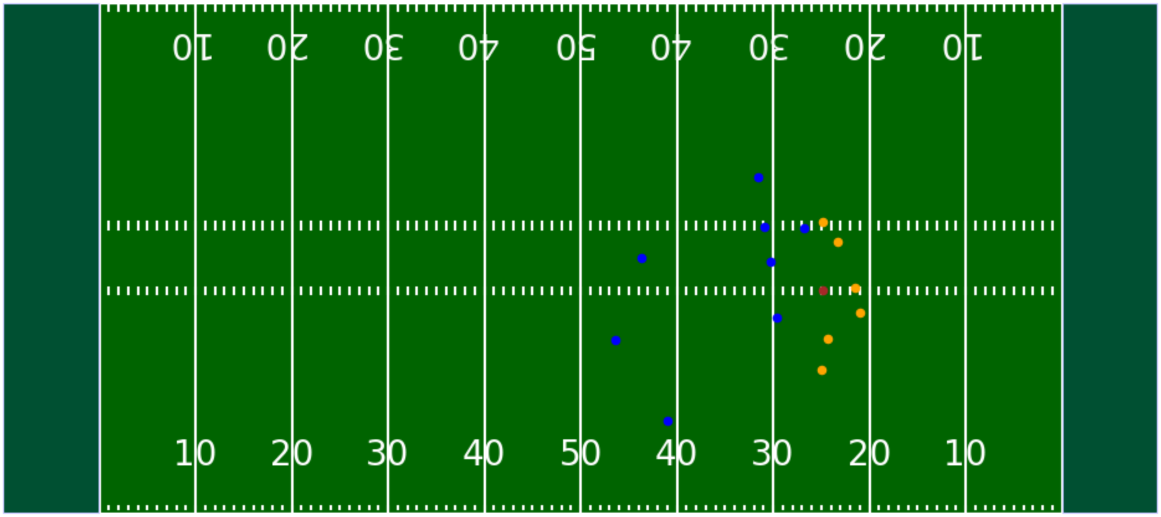
In [114]:
```
sel_df['event'].unique()
```

Out[114]:
```
array(['None', 'ball_snap', 'first_contact', 'pass_forward',
       'pass_outcome_interception', 'tackle'], dtype=object)
```

Plotting the data for the event of ball_snap, that is, when the quarterback first receives the football

In [115]:

```
# Creating the football field
fig, ax = create_football_field()

# Plotitng the home team
home_team.query('event == "ball_snap"').plot(x='x', y='y', kind='scatter', ax=

# Plotting the away team
away_team.query('event == "ball_snap"').plot(x='x', y='y', kind='scatter', ax=

# Plotting the football
football.query('event == "ball_snap"').plot(x='x', y='y', kind='scatter', ax=a

# Displaying the plot
plt.show()
```



In this way, we can visualize any game, play and event on the football field.