This project compares the goals of Cristiano Ronaldo and Lionel Messi.

We will collect their goal data from club and international matches.

Using EDA, we will explore and visualize their performance.

The aim is to understand their goal-scoring patterns and achievements.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:
```
#Load the cristiano ronaldo vs Lionel Messi dataset from Google Drive
df = pd.read_csv('/content/drive/MyDrive/Harsha_Dataset/cristiano_vs_mess
```

In [4]:
```
#Display the basic information about the DataFrame including column names
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1300 entries, 0 to 1299
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   player    1300 non-null   object
 1   comp      846 non-null    object
 2   round     846 non-null    object
 3   date      846 non-null    object
 4   venue     846 non-null    object
 5   opp       846 non-null    object
 6   pos       793 non-null    object
 7   min       1300 non-null   object
 8   type      1281 non-null   object
 9   assisted  874 non-null    object
dtypes: object(10)
memory usage: 101.7+ KB
```

In [5]:
```
#Display the first 3 rows of the dataset
df.head(3)
```

Out[5]:

	player	comp	round	date	venue	opp	pos	min	type	assisted
0	ronaldo	Liga NOS	6	10/7/2002	H	Moreirense	RW	34'	NaN	NaN
1	ronaldo	Liga NOS	6	10/7/2002	H	Moreirense	NaN	90'	NaN	NaN
2	ronaldo	Liga NOS	8	10/26/2002	A	Boavista	NaN	88'	NaN	Carlos Martins

Fill Nan goal Values

If palyer score more than one goal on game only min and type filled

In [6]:
```
#Display the column names of the DataFrame
df.columns
```

Out[6]:
```
Index(['player', 'comp', 'round', 'date', 'venue', 'opp', 'pos', 'min',
'type',
       'assisted'],
      dtype='object')
```

In [7]:
```
#Fill missing values in the 'comp' (competition) column using forward  fi
df['comp']=df['comp'].ffill()

#Fill missing values in the 'date' column using forward fill method
df['date']=df['date'].ffill()

#Fill missing values in the 'round' column using forward fill method
df['round']=df['round'].ffill()

#Fill missing values in the 'venue' column using forward fill method
df['venue']=df['venue'].ffill()

#Fill missing values in the 'opp'(opponent) column using forward fill met
df['opp']=df['opp'].ffill()

#check  the update structure of the DataFrame to ensure missing values ar
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1300 entries, 0 to 1299
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   player    1300 non-null   object
 1   comp      1300 non-null   object
 2   round     1300 non-null   object
 3   date      1300 non-null   object
 4   venue     1300 non-null   object
 5   opp       1300 non-null   object
 6   pos       793 non-null    object
 7   min       1300 non-null   object
 8   type      1281 non-null   object
 9   assisted  874 non-null    object
dtypes: object(10)
memory usage: 101.7+ KB
```

In [8]:
```
#Remove single quotes (') from the 'min'  column values
df['min']=df['min'].apply(lambda x:x.replace("'",''))

# Remove plus signs (+) from the 'min' column values
df['min']=df['min'].apply(lambda x:x.replace("+",''))

#Display all unique values in the 'min' column after cleaning
df['min'].unique()
```

Out[8]:
```
array(['34', '90', '88', '67', '13', '80', '74', '89', '60', '4', '44',
       '87', '9', '8', '54', '58', '451', '21', '76', '63', '12', '68',
       '14', '38', '45', '59', '23', '19', '73', '82', '10', '39', '84',
       '85', '47', '50', '77', '49', '28', '5', '62', '51', '41', '35',
       '903', '22', '81', '48', '70', '902', '69', '56', '79', '16', '5
3',
       '3', '24', '32', '26', '25', '30', '43', '65', '11', '6', '57',
       '61', '17', '27', '1', '75', '2', '64', '71', '18', '78', '29',
       '36', '15', '55', '86', '901', '72', '20', '42', '103', '46', '5
2',
       '904', '37', '66', '906', '31', '40', '120', '83', '7', '33', '9
7',
       '104', '105', '109', '907', '457', '110', '452', '454'],
      dtype=object)
```

Manipulate

In [9]:
```
#shows basic information about the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1300 entries, 0 to 1299
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   player    1300 non-null   object
 1   comp      1300 non-null   object
 2   round     1300 non-null   object
 3   date      1300 non-null   object
 4   venue     1300 non-null   object
 5   opp       1300 non-null   object
 6   pos       793 non-null    object
 7   min       1300 non-null   object
 8   type      1281 non-null   object
 9   assisted  874 non-null    object
dtypes: object(10)
memory usage: 101.7+ KB
```

In [10]:
```
df['min'] = pd.to_numeric(df['min']) #This line converts the values in th
df['time_class'] = df['min'].apply(lambda x: 'first_half' if x <= 45 else

#It checks the value in the 'min' column for each row and gives it a labe
df.head(10)
```

Out[10]:

	player	comp	round	date	venue	opp	pos	min	type	ass
0	ronaldo	Liga NOS	6	10/7/2002	H	Moreirense	RW	34	NaN	
1	ronaldo	Liga NOS	6	10/7/2002	H	Moreirense	NaN	90	NaN	
2	ronaldo	Liga NOS	8	10/26/2002	A	Boavista	NaN	88	NaN	M
3	ronaldo	Taça de Portugal Placard	Fourth Round	11/24/2002	H	Estarreja	NaN	67	Left-footed shot	
4	ronaldo	Taça de Portugal Placard	Fifth Round	12/18/2002	H	Oliv. Hospital	NaN	13	NaN	
5	ronaldo	Premier League	11	11/1/2003	H	Portsmouth	RW	80	Right-footed shot	
6	ronaldo	FA Cup	Fifth Round	2/14/2004	H	Man City	RW	74	Tap-in	
7	ronaldo	Premier League	29	3/20/2004	H	Spurs	NaN	89	Right-footed shot	
8	ronaldo	Premier League	32	4/10/2004	A	Birmingham	NaN	60	Header	
9	ronaldo	Premier League	38	5/15/2004	A	Aston Villa	NaN	4	Right-footed shot	

In [11]:
```
df['assist'] = df['assisted'].fillna(0)  #This line takes the 'assisted'
df['solo'] = df['assist'].apply(lambda x: 'solo' if x == 0 else 'assisted
df.head()
```

Out[11]:

	player	comp	round	date	venue	opp	pos	min	type	assis
0	ronaldo	Liga NOS	6	10/7/2002	H	Moreirense	RW	34	NaN	
1	ronaldo	Liga NOS	6	10/7/2002	H	Moreirense	NaN	90	NaN	
2	ronaldo	Liga NOS	8	10/26/2002	A	Boavista	NaN	88	NaN	Ca Mai
3	ronaldo	Taça de Portugal Placard	Fourth Round	11/24/2002	H	Estarreja	NaN	67	Left-footed shot	C Pr
4	ronaldo	Taça de Portugal Placard	Fifth Round	12/18/2002	H	Oliv. Hospital	NaN	13	NaN	

In [12]:
```
from datetime import date #This imports the date class from Python's buil
df['date'] = pd.to_datetime(df['date']) #This line converts the 'date' co
L = ['year', 'month', 'day', 'dayofweek', 'dayofyear', 'weekofyear', 'qua
```

In [13]:
```
# Assuming 'df' already exists and has a 'dayofweek' column from a dateti
# Example: make sure 'df["date"]' is already converted to datetime
df['dayofweek'] = pd.to_datetime(df['date']).dt.dayofweek  # Monday=0, Su

# Step 1: Convert to numeric (if needed — usually it's already numeric fr
df['dayofweek'] = pd.to_numeric(df['dayofweek'])

# Step 2: Add 1 to shift from 0–6 (Mon–Sun) to 1–7
df['dayofweek'] = df['dayofweek'].apply(lambda x: x + 1)

# Step 3: Show unique values to confirm it's working
print(df['dayofweek'].unique())

# Step 4: Add 'goal' column with value 1 in all rows
df['goal'] = 1
```

```
[1 6 7 3 2 4 5]
```

In [14]:
```
df.head()
```

Out[14]:

	player	comp	round	date	venue	opp	pos	min	type	assisted
0	ronaldo	Liga NOS	6	2002-10-07	H	Moreirense	RW	34	NaN	NaN
1	ronaldo	Liga NOS	6	2002-10-07	H	Moreirense	NaN	90	NaN	NaN
2	ronaldo	Liga NOS	8	2002-10-26	A	Boavista	NaN	88	NaN	Carlos Martins
3	ronaldo	Taça de Portugal Placard	Fourth Round	2002-11-24	H	Estarreja	NaN	67	Left-footed shot	César Prates
4	ronaldo	Taça de Portugal Placard	Fifth Round	2002-12-18	H	Oliv. Hospital	NaN	13	NaN	NaN

GOALS

In [15]:
```
#show how many time each player appears in the dataset
df['player'].value_counts()
```

Out[15]:

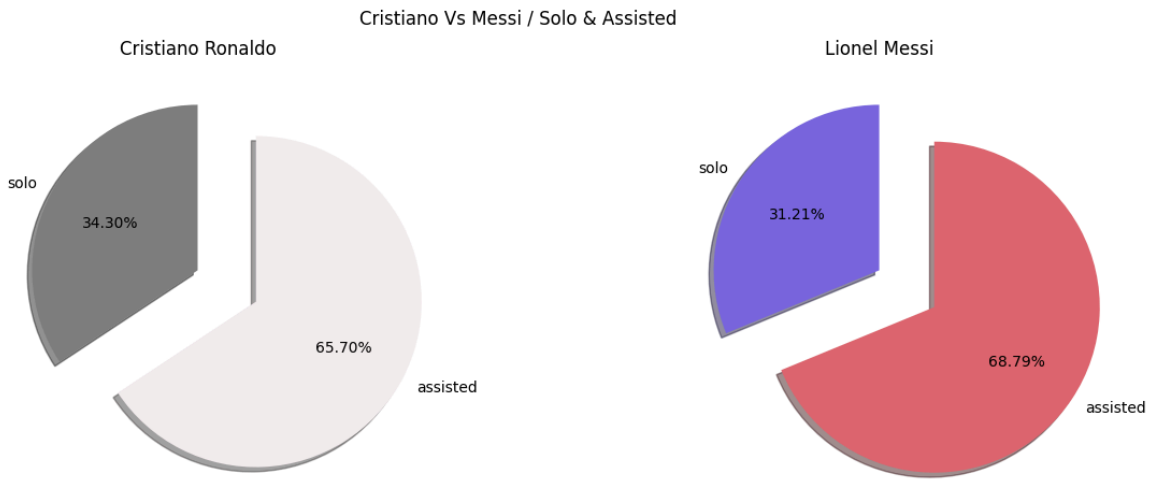
player	count
ronaldo	656
messi	644

dtype: int64

In [16]:
```
# Filter only Cristiano Ronaldo's data
df_ronaldo = df.loc[df['player'] == 'ronaldo']

# Filter only Lionel Messi's data
df_messi = df.loc[df['player'] == 'messi']

# === Ronaldo's solo and assisted goals ===

# Get all solo goals (not assisted) by Ronaldo
ronaldo_solo = df_ronaldo[df_ronaldo['solo'] == 'solo']

# Get all assisted goals by Ronaldo
ronaldo_assisted = df_ronaldo[df_ronaldo['solo'] == 'assisted']

# Count of solo and assisted goals for Ronaldo
slices = [len(ronaldo_solo), len(ronaldo_assisted)]

# Labels for the pie chart
labels = ['solo', 'assisted']

# === Messi's solo and assisted goals ===

# Get all solo goals by Messi
```

```
messi_solo = df_messi[df_messi['solo'] == 'solo']

# Get all assisted goals by Messi
messi_assisted = df_messi[df_messi['solo'] == 'assisted']

# Count of solo and assisted goals for Messi
slices1 = [len(messi_solo), len(messi_assisted)]

# Labels for Messi's chart
labels1 = ['solo', 'assisted']

# === Plotting the Pie Charts ===

# Create 1 row, 2 column subplots (side-by-side)
fig, axes = plt.subplots(1, 2, figsize=(15, 5), sharey=False)

# Main title for the figure
fig.suptitle('Cristiano Vs Messi / Solo & Assisted')

# Pie chart for Ronaldo
axes[0].pie(
    slices,                 # Values to show (solo, assisted)
    labels=labels,          # Labels on the chart
    startangle=90,          # Start drawing from the top
    shadow=1,               # Adds a shadow effect
    explode=(0, 0.4),       # Pull out the 'assisted' slice for emphasis
    autopct='%1.2f%%',      # Show percentage with 2 decimals
    colors=['#808080', '#F2EBED']  # Custom colors
)
axes[0].set_title('Cristiano Ronaldo')  # Sub-title for Ronaldo's chart

# Pie chart for Messi
axes[1].pie(
    slices1,                # Values for Messi
    labels=labels1,         # Labels: solo, assisted
    startangle=90,
    shadow=1,
    explode=(0, 0.4),       # Emphasize assisted goals
    autopct='%1.2f%%',
    colors=['#7868DF', '#DF6870']
)
axes[1].set_title('Lionel Messi')  # Sub-title for Messi's chart
```

Out[16]: `Text(0.5, 1.0, 'Lionel Messi')`



In [17]:
```
# Step 1: Count number of goals on each date and convert to DataFrame
r_goal = df_ronaldo['date'].value_counts().sort_values(ascending=False).r

# Step 2: Rename columns for clarity
r_goal.columns = ['date', 'goal_count']

# Step 3: Add nicknames based on how many goals were scored
r_goal['nick'] = r_goal['goal_count'].apply(
    lambda x: 'hatrick' if x == 3 else (
        'haul' if x == 4 else (
            'glut' if x == 5 else (
                'brace' if x == 2 else 'single goal'
            )
        )
    )
)

# Show the top rows
r_goal.head()
```

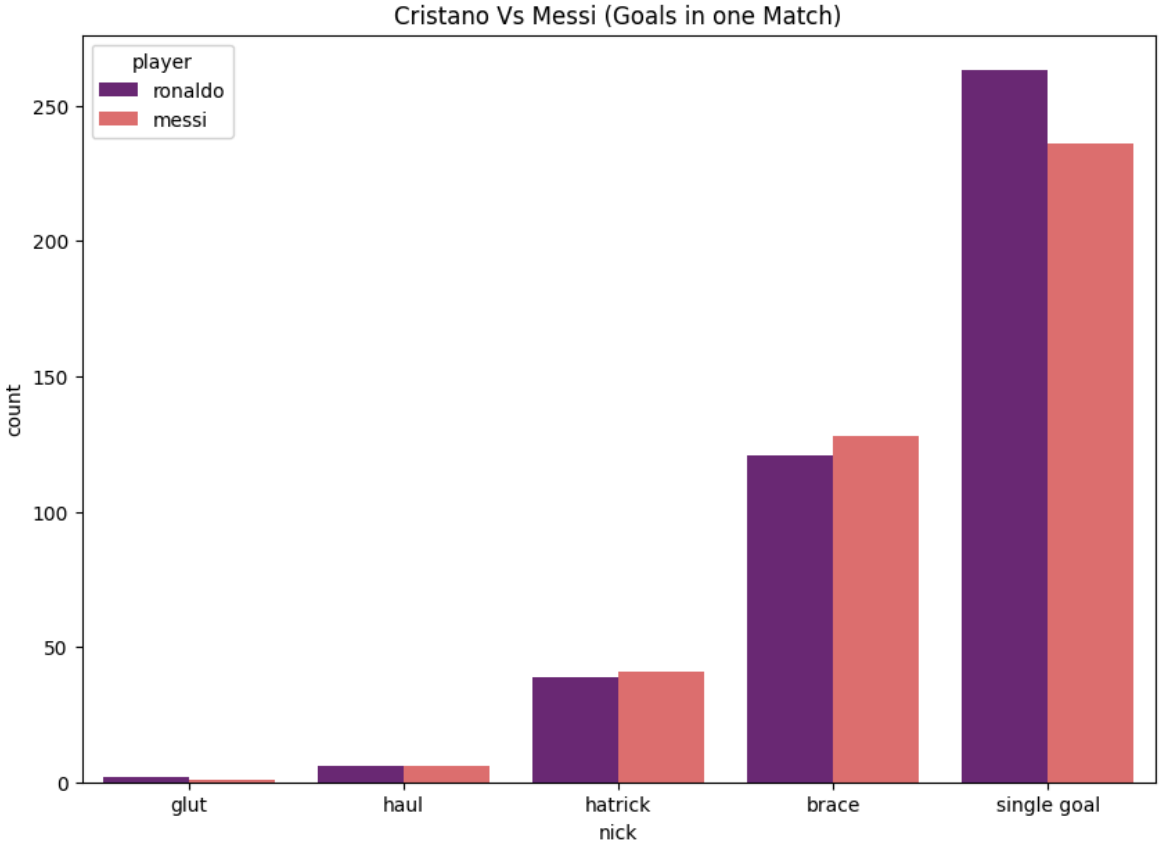
Out[17]:

	date	goal_count	nick
0	2015-09-12	5	glut
1	2015-04-05	5	glut
2	2015-12-08	4	haul
3	2016-03-05	4	haul
4	2011-05-07	4	haul

In [18]:
```
# Step 1: Count goals by date for Messi
m_goal = pd.DataFrame(df_messi['date'].value_counts().sort_values(ascendi
m_goal.columns = ['date', 'goal_count']  # Rename columns to be clear

# Step 2: Label each row with a nickname based on the number of goals
m_goal['nick'] = m_goal['goal_count'].apply(
    lambda x: 'hatrick' if x == 3 else (
        'haul' if x == 4 else (
            'glut' if x == 5 else (
                'brace' if x == 2 else 'single goal'
            )
        )
    )
)

# Step 3: Add player names
m_goal['player'] = 'messi'
r_goal['player'] = 'ronaldo'  # Assuming r_goal is structured similarly

# Step 4: Combine both players' data
all_goal = pd.concat([r_goal, m_goal], ignore_index=True)

# Optional: Rename column if it represents goal count
# Only do this if the column 'date' actually holds goal numbers, not actu
# But in this fixed version, 'date' is really a date, so you probably sho
# So this line is likely NOT needed anymore:
# all_goal.rename(columns={'date': 'goal_count'}, inplace=True)
```

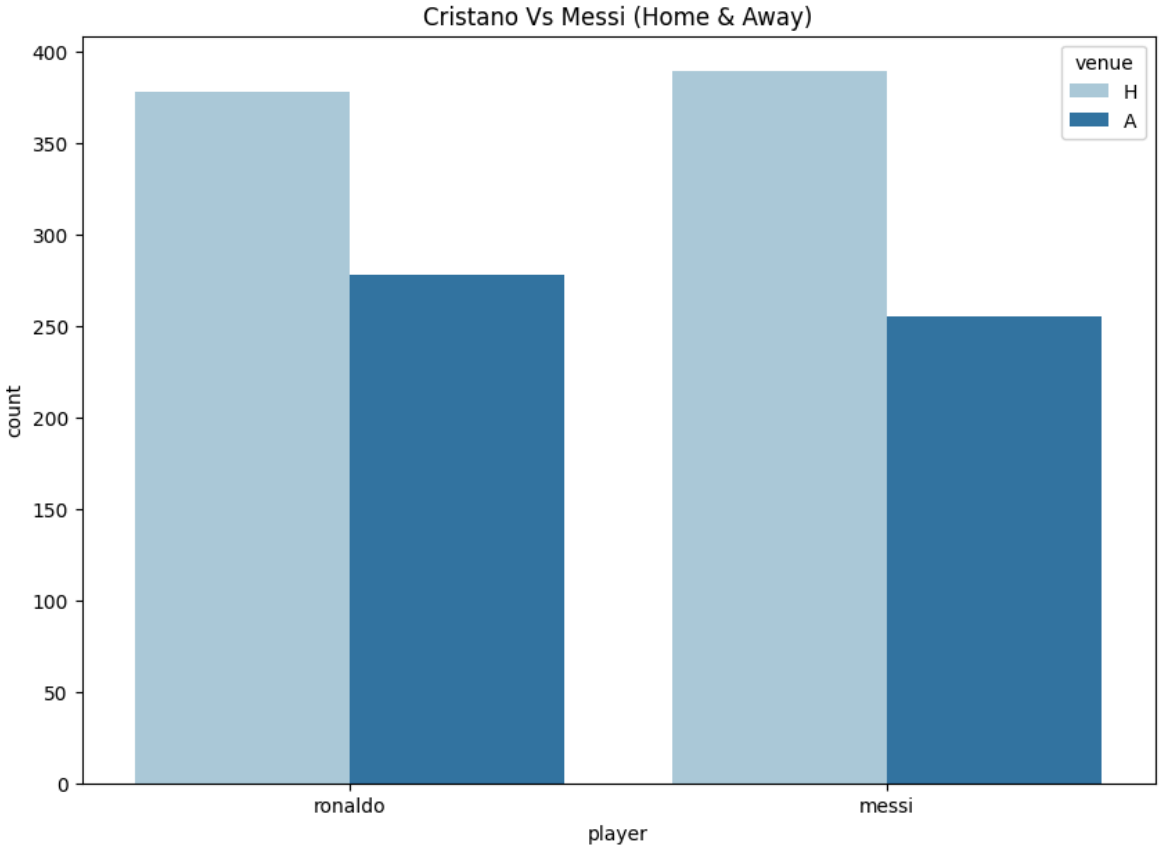
In [19]:
```
fig = px.box(all_goal, x="player", y='goal_count')
fig.show()
```

In [20]:
```
#create a count plot showing the number of goals by each player(cristiano
# nick reperesents individual matches , and the hue differentiates betwee
plt.figure(figsize=(10,7))
sns.countplot(data=all_goal, x='nick', hue='player', palette='magma').set
```



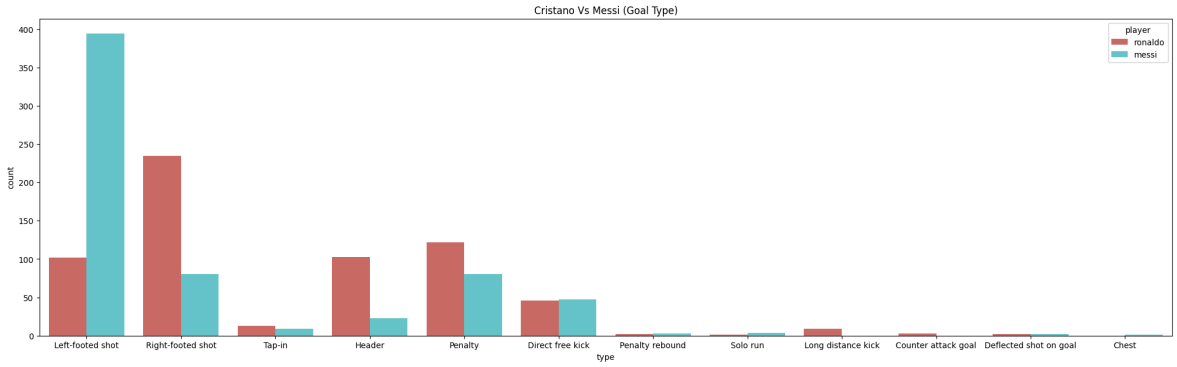
Home & Away Goals

In [21]:
```
#create a count plot to compare cristano ronaldo and Messi's matches play
#The hue represents the venue type  (home or away) for each player
plt.figure(figsize=(10,7))
sns.countplot(data=df,x='player',hue='venue',palette="Paired").set_title(
```

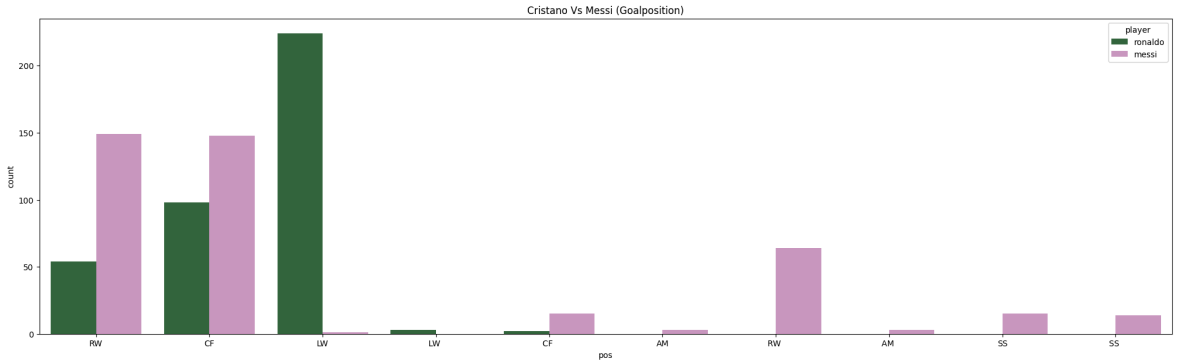


Goal Type

In [22]:
```
#create a count plot to compare the type of goals scored by Ronaldo and M
#The hue differentiates between the two players
plt.figure(figsize=(25,7))
sns.countplot(data=df,x='type',hue='player',palette="hls").set_title('Cri
```



In [23]:
```
#shows how many goals each player scored from different positions on the
plt.figure(figsize=(25,7))
sns.countplot(data=df,x='pos',hue='player',palette="cubehelix").set_title
```

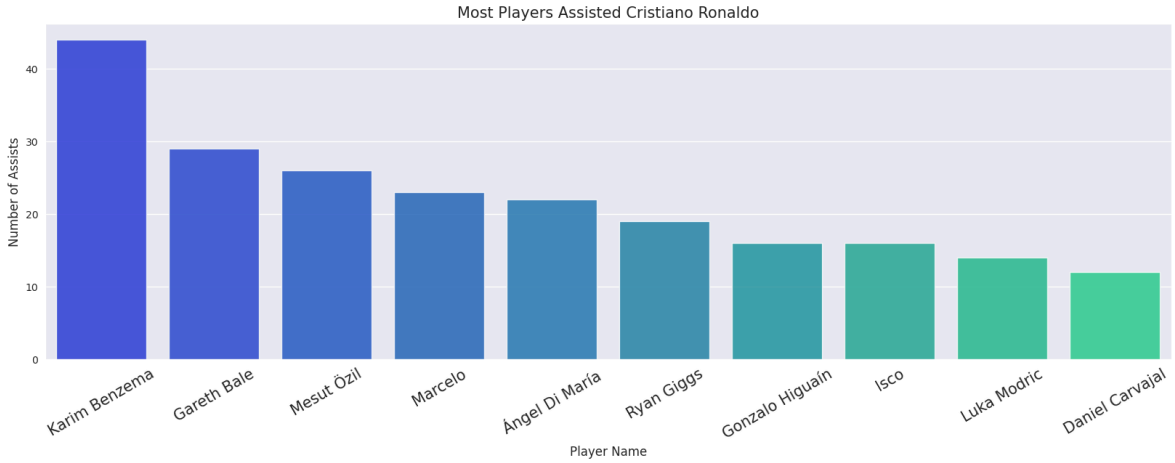


Best Friends to Cristiano and Messi¶

In [24]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Count how many times each player assisted Ronaldo and get the top 10
r_assist = df_ronaldo['assisted'].value_counts()
r_assist = r_assist[:10]

# Set the style of the plot
sns.set_style("darkgrid")

# Create a figure with specified size
plt.figure(figsize=(20,6))

# Create a bar plot to visualize the top 10 players who assisted Ronaldo
r_assist_vis = sns.barplot(x=r_assist.index, y=r_assist.values, alpha=0.8

# Set the title and labels
plt.title('Most Players Assisted Cristiano Ronaldo', fontsize=15)
plt.ylabel('Number of Assists', fontsize=12)
plt.xlabel('Player Name', fontsize=12)

# Rotate x-axis labels for better readability
r_assist_vis.set_xticklabels(r_assist.index, rotation=30, fontsize=15)
```

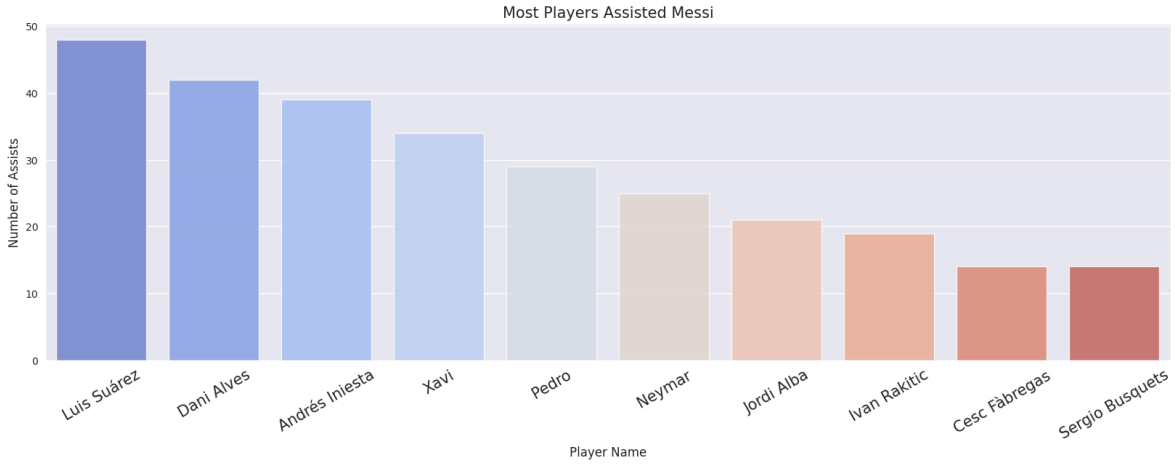
```
# Show the plot
plt.show()
```

```
<ipython-input-24-4598ccdb1dce>:15: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.


<ipython-input-24-4598ccdb1dce>:23: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. af
ter set_ticks() or using a FixedLocator.
```



In [25]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Count how many times each player assisted Messi and get the top 10
m_assist = df_messi['assisted'].value_counts()
m_assist = m_assist[:10]

# Set the style of the plot
sns.set_style("darkgrid")

# Create a figure with specified size
plt.figure(figsize=(20,6))

# Create a bar plot to visualize the top 10 players who assisted Messi
m_assist_vis = sns.barplot(x=m_assist.index, y=m_assist.values, alpha=0.8

# Set the title and labels
plt.title('Most Players Assisted Messi', fontsize=15)
plt.ylabel('Number of Assists', fontsize=12)
plt.xlabel('Player Name', fontsize=12)

# Rotate x-axis labels for better readability
m_assist_vis.set_xticklabels(m_assist.index, rotation=30, fontsize=15)

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

```
<ipython-input-25-9056c773d4d8>:15: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.


<ipython-input-25-9056c773d4d8>:23: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. af
ter set_ticks() or using a FixedLocator.
```

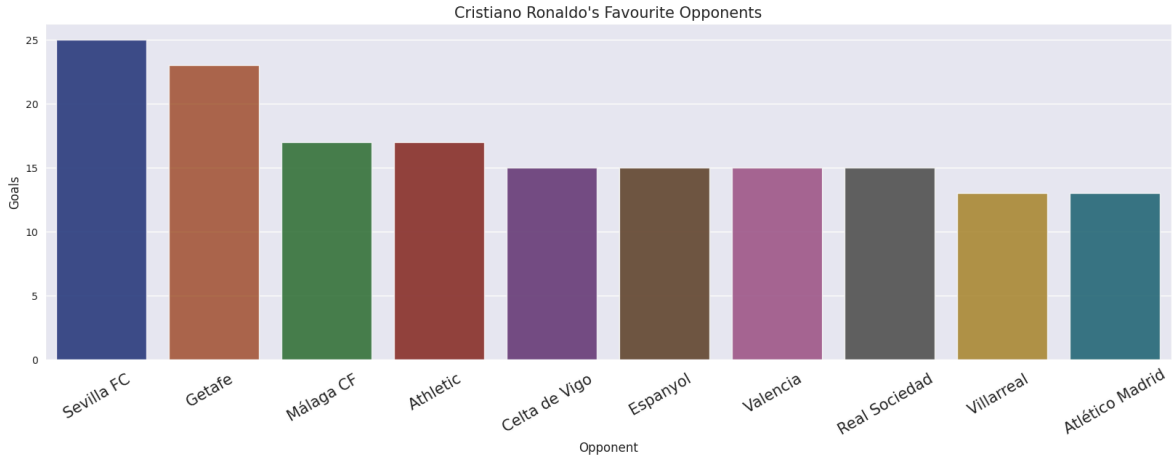


Favourite Opponent

In [26]:

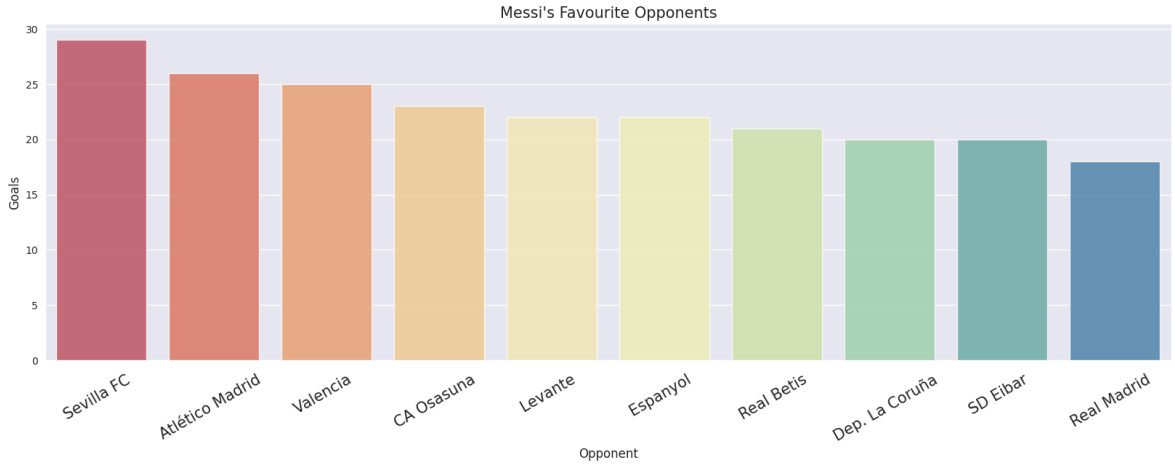
```
import seaborn as sns
import matplotlib.pyplot as plt

# Count the number of times each opponent has faced Ronaldo and get the t
r_op = df_ronaldo['opp'].value_counts()
r_op = r_op[:10]

# Set the style of the plot
sns.set_style("darkgrid")

# Create a figure with specified size
plt.figure(figsize=(20,6))

# Create a bar plot to visualize Ronaldo's top 10 favorite opponents
r_op_vis = sns.barplot(x=r_op.index, y=r_op.values, alpha=0.8, palette="d

# Set the title and labels
plt.title('Cristiano Ronaldo\'s Favourite Opponents', fontsize=15)
plt.ylabel('Goals', fontsize=12)
plt.xlabel('Opponent', fontsize=12)

# Rotate x-axis labels for better readability
r_op_vis.set_xticklabels(r_op.index, rotation=30, fontsize=15)

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

```
<ipython-input-26-c285e64283b0>:15: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.


<ipython-input-26-c285e64283b0>:23: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. af
ter set_ticks() or using a FixedLocator.
```



In [27]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Count the number of times each opponent has faced Messi and get the top
m_op = df_messi['opp'].value_counts()
m_op = m_op[:10]

# Set the style of the plot
sns.set_style("darkgrid")

# Create a figure with specified size
plt.figure(figsize=(20,6))

# Create a bar plot to visualize Messi's top 10 favourite opponents
m_op_vis = sns.barplot(x=m_op.index, y=m_op.values, alpha=0.8, palette="S

# Set the title and labels
plt.title('Messi\'s Favourite Opponents', fontsize=15)
plt.ylabel('Goals', fontsize=12)
plt.xlabel('Opponent', fontsize=12)

# Rotate x-axis labels for better readability
m_op_vis.set_xticklabels(m_op.index, rotation=30, fontsize=15)

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

```
<ipython-input-27-5d490737abbc>:15: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.


<ipython-input-27-5d490737abbc>:23: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. af
ter set_ticks() or using a FixedLocator.
```



Scoring Time

In [28]:
```
#Group Ronaldo's goals by minute and count how many times he scored in ea
min_ronaldo1=df_ronaldo.groupby(['min']).size().to_frame('count').reset_i

#sort the grouped data in descending order and select the top 10 minutes
min_ronaldo=min_ronaldo1.sort_values(by='count', ascending=False)[:10]

#Reset the index after sorting
min_ronaldo=min_ronaldo.reset_index()

#Rename the 'min' column to 'Ronaldo_min'  for clarity
min_ronaldo=min_ronaldo.rename(columns={'min':('Ronald_min')})

#Drop the extra 'index' column created during reset
min_ronaldo=min_ronaldo.drop(columns=['index'])
#=====
#=====
#Group Messi's goals by minute and count how many times  he scored in eac
min_messi=df_messi.groupby(['min']).size().to_frame('count').reset_index(

#sort the grouped data in desecending order and select the top 10 minutes
min_messi=min_messi.sort_values(by='count', ascending=False)[:10]

#Reset the index after sorting
min_messi=min_messi.reset_index()

#Rename the 'min' column to 'messi_min' for clarity
min_messi=min_messi.rename(columns={'min':('messi_min')})
```

```
#Drop the  extra 'index' column created during reset
min_messi=min_messi.drop(columns='index')
#====

#Display the final DataFrame  showing Ronaldo's top 10 goal-scoring minut
min_ronaldo
```

Out[28]:

	Ronald_min	count
0	90	18
1	23	14
2	45	14
3	76	13
4	70	13
5	89	13
6	82	12
7	26	11
8	49	10
9	59	10

In [29]:
```
#show Messi's  top 10  goal-scoring minutes
min_messi
```

Out[29]:

	messi_min	count
0	55	13
1	78	12
2	45	12
3	87	12
4	86	11
5	63	11
6	82	11
7	75	11
8	90	11
9	16	10

In [30]:
```
#Filter ronaldo's goals to include only those scored before 90th minute
min_cr7=df_ronaldo[df_ronaldo['min']<90]

#Extract the 'min' column values (goals minutes)   as Numpy for Ronaldo
min_values=min_cr7['min'].values
#========
#Filter  messi's goals to include only those scored before 90th minute
min_messi=df_messi[df_messi['min']<90]
```

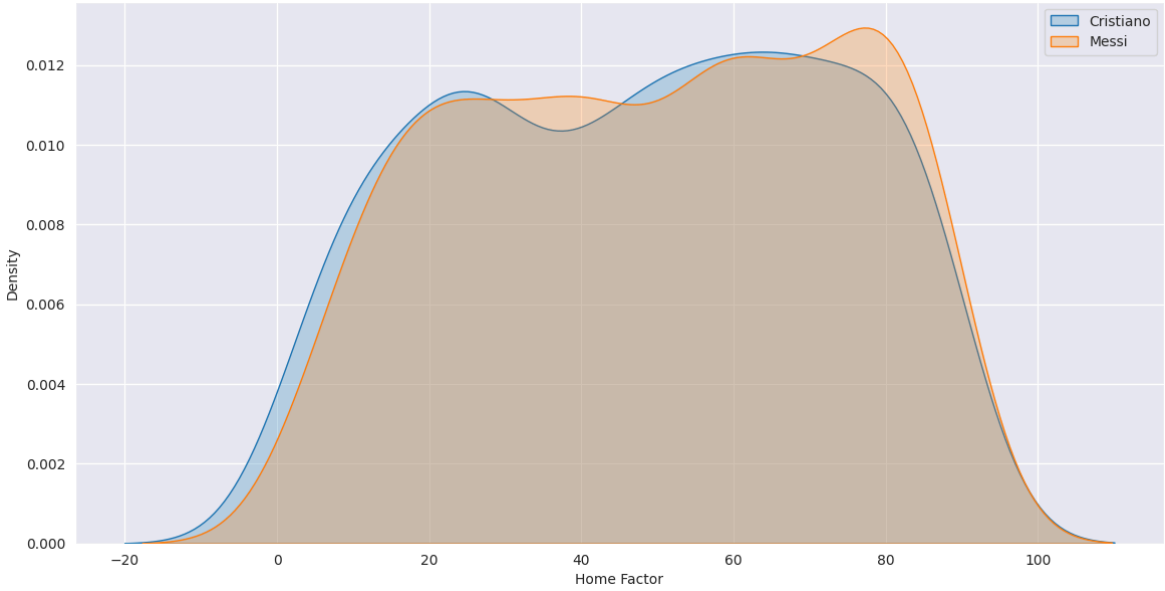
```
# Extract the 'min' column values (goal minutes) as a NumPy array for Mes
min_values_messi=min_messi['min'].values
min_values_messi
#===
#figure,axes = plt.subplots(1,2,figsize=(10,5))
# Create a figure for the histogram with a specific size
plt.figure(figsize=(15,5))

# Plot the distribution of Ronaldo's goals across minutes using a histogr
plt.hist(min_values,histtype='bar',bins=45,density=True,label='Ronaldo',a

# Plot the distribution of Messi's goals across minutes using a histogram
plt.hist(min_values_messi,bins=45,histtype='bar',density=True,label='Mess

# Add a legend to distinguish between Ronaldo and Messi
plt.legend(loc='upper left')
```

Out[30]: <matplotlib.legend.Legend at 0x78e8a0981f10>

In [31]:
```
# Create a new figure for the KDE plot with a defined size
plt.figure(figsize=(14,7))

# Plot the Kernel Density Estimate (KDE) for Ronaldo's goal minutes
sns.kdeplot(min_values, shade = True)

# Plot the KDE for Messi's goal minutes
sns.kdeplot(min_values_messi, shade = True)

# Add a legend to label the two lines as Cristiano and Messi
plt.legend(['Cristiano','Messi'])

# Label the x-axis (you can change 'Home Factor' to something like 'Minut
plt.xlabel('Home Factor')
```

```
<ipython-input-31-25ce9fc5980c>:5: FutureWarning:


`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.


<ipython-input-31-25ce9fc5980c>:8: FutureWarning:


`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.
```

Out[31]: Text(0.5, 0, 'Home Factor')



In [32]:
```
# Create a pivot table to summarize total goals scored by each player
# across different time classes (e.g., 'First Half', 'Second Half').
# Rows represent players, columns represent time classes,
# and values are the sum of goals. The index is reset to keep 'player' as
df_stack = pd.pivot_table(df, values='goal', index=['player'],
                          columns=['time_class'], aggfunc=np.sum).reset_index()
df_stack
```

```
<ipython-input-32-30ca59033d2d>:5: FutureWarning:

The provided callable <function sum at 0x78e8fd9b42c0> is currently using
DataFrameGroupBy.sum. In a future version of pandas, the provided callable
will be used directly. To keep current behavior pass the string "sum" inst
ead.
```

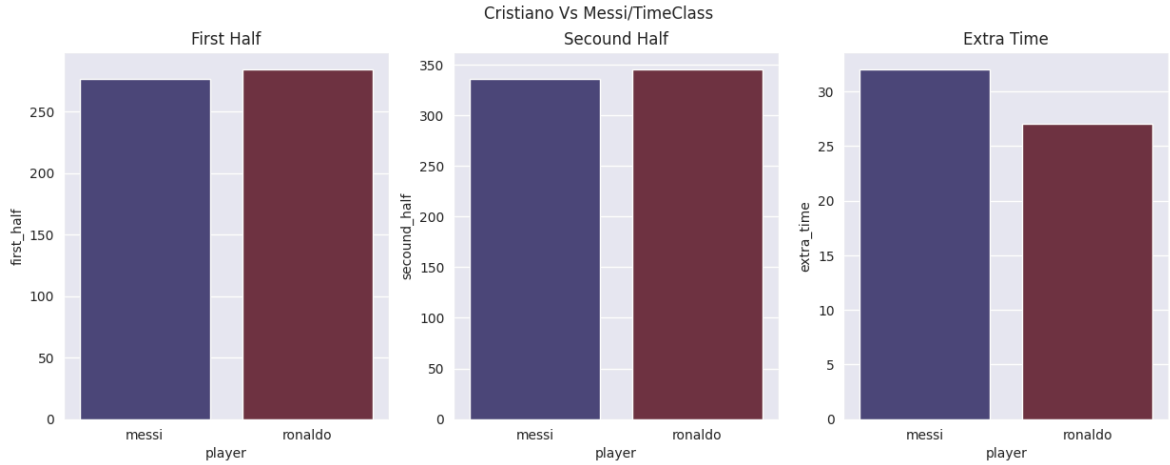
Out[32]:

	time_class	player	extra_time	first_half	secound_half
0	messi	32	276	336
1	ronaldo	27	284	345

In [33]:
```
# Set the seaborn style to 'darkgrid' for better visual appearance
sns.set_style("darkgrid")

# Create a figure with 3 subplots (side by side), sharing the y-axis is t
fig, axes = plt.subplots(1, 3, figsize=(15, 5), sharey=False)

# Set a common title for the entire figure
fig.suptitle('Cristiano Vs Messi/TimeClass')

# Plot bar chart for First Half goals
sns.barplot(ax=axes[0], x=df_stack.player, y=df_stack.first_half,palette=
axes[0].set_title('First Half')

# Plot bar chart for Second Half goals
sns.barplot(ax=axes[1], x=df_stack.player, y=df_stack.secound_half,palett
axes[1].set_title('Secound Half')

# Plot bar chart for Extra Time goals
sns.barplot(ax=axes[2], x=df_stack.player, y=df_stack.extra_time,palette=
axes[2].set_title('Extra Time')
```

```
<ipython-input-33-842c7b0b1789>:11: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.


<ipython-input-33-842c7b0b1789>:15: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.


<ipython-input-33-842c7b0b1789>:19: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.
```

Out[33]:
```
Text(0.5, 1.0, 'Extra Time')
```



In [34]:
```
# Display all the unique values in the 'comp' (competition) column
df['comp'].unique()
```

Out[34]:
```
array(['Liga NOS', 'Taça de Portugal Placard', 'Premier League', 'FA Cup',
       'Champions League Qualifying', 'EFL Cup', 'Champions League',
       'Club World Cup', 'LaLiga', 'Copa del Rey', 'Supercopa',
       'UEFA Super Cup', 'Serie A', 'Supercoppa Italiana', 'Coppa Italia'],
      dtype=object)
```

UEFA

In [35]:
```
# Filter the dataframe to include only rows where the 'comp' column is 'C
df_champ=df.loc[df['comp']=='Champions League']
#====
# Count the number of occurrences of each player in the filtered datafram
df_champ['player'].value_counts()
```

Out[35]:

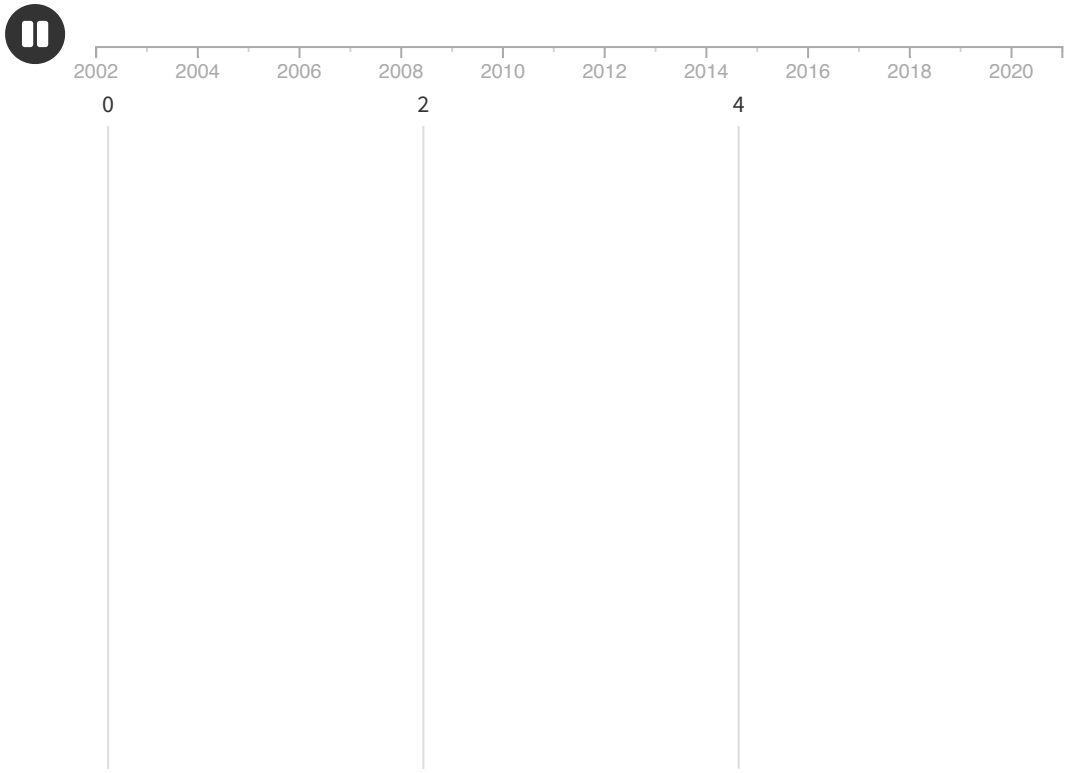
player	count
ronaldo	134
messi	118

dtype: int64

Goals Race for each season - 2020

In [36]:
```
from IPython.core.display import HTML
HTML('''<div class="flourish-embed flourish-bar-chart-race" data-src="vis
```

Out[36]:

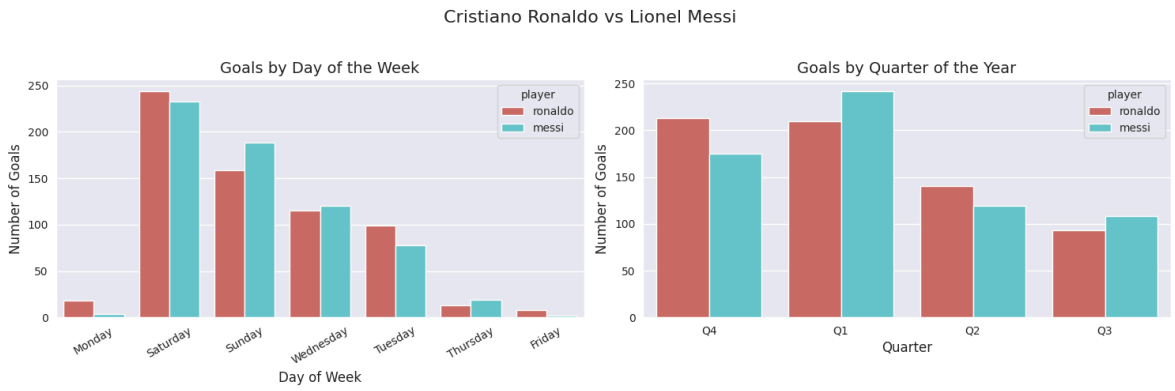


A Flourish bar chart race

In [37]:
```
# Ensure the 'date' column is in datetime format
df['date'] = pd.to_datetime(df['date'])

# Add 'dayofweek' (name of the day) column
df['dayofweek'] = df['date'].dt.day_name()

# Add 'quarter' (Q1, Q2, Q3, Q4) column
df['quarter'] = df['date'].dt.quarter.apply(lambda x: f'Q{x}')
```

In [38]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Create 2 side-by-side subplots
fig, axes = plt.subplots(1, 2, figsize=(15, 5), sharey=False)

# Set a main title for the entire figure
fig.suptitle('Cristiano Ronaldo vs Lionel Messi', fontsize=16)

# Plot 1: Goals by Day of the Week
```

```
sns.countplot(ax=axes[0], data=df, x='dayofweek', hue='player', palette="
axes[0].set_title('Goals by Day of the Week', fontsize=14)
axes[0].set_xlabel('Day of Week', fontsize=12)
axes[0].set_ylabel('Number of Goals', fontsize=12)
axes[0].tick_params(axis='x', rotation=30)

# Plot 2: Goals by Quarter
sns.countplot(ax=axes[1], data=df, x='quarter', hue='player', palette="hl
axes[1].set_title('Goals by Quarter of the Year', fontsize=14)
axes[1].set_xlabel('Quarter', fontsize=12)
axes[1].set_ylabel('Number of Goals', fontsize=12)

# Adjust layout so titles don't overlap
plt.tight_layout(rect=[0, 0, 1, 0.95])

# Show the plots
plt.show()
```



Conclusion

•Ronaldo Score more goals than Messi •Ronaldo solo goal percentage is 3% higher than Messi •Ronaldo score single goals more than Messi but Messi score more brace and whatricks •Cristiano scored more in away matches but Messi scored more in home •Cristiano Ronaldo Score more headers, long-distance, counter-attack goal and penalties more than Messi •Lionel Messi Scored many goals in many positions in the squad contrary to Cristiano •Ronaldo Scored more in UEFA •Messi and Cristiano Scored more against the same team which is Sevilla •Messi Scored more in the last minutes and extra times •Messi started the season better than Ronaldo but Ronaldo end it better

The Tale of Two Titans

The football world has been blessed with two extraordinary talents: Cristiano Ronaldo and Lionel Messi. Their rivalry has defined a generation, captivating fans with their mesmerizing skills and goal-scoring brilliance.

Ronaldo: The Goal Machine

Cristiano Ronaldo, a force of nature, has established himself as a prolific goal scorer. His hunger for goals is insatiable, as evident from his higher overall tally compared to Messi. This Portuguese powerhouse is a master of solo plays, often finding the back of the net without relying on assists. He's a threat from anywhere on the pitch, known for his powerful long-range shots, lightning-fast counter-attacks, and aerial dominance in headers. And when it comes to penalties, he rarely misses, converting pressure into goals with ice-cold precision.

Messi: The Magician

Lionel Messi, the Argentine maestro, is a magician with the ball at his feet. He weaves through defenses like a phantom, creating scoring opportunities with his unparalleled dribbling and vision. Messi is more likely to score in a variety of ways and positions, highlighting his adaptability and tactical intelligence. While Ronaldo might score more solo goals, Messi shines in teamwork, often racking up assists for his teammates. He has a knack for scoring in crucial moments, delivering when it matters most, especially in the closing stages of games and during extra time.

Contrasting Styles, Shared Stage

While both excel at home and away, Ronaldo shows a slight preference for away games, while Messi feels most comfortable on his home turf. They share a common nemesis in Sevilla, a team against which both players have consistently found success.

Looking at their performance throughout a season, we see different trends. Messi starts strong, exploding with goals early on. However, Ronaldo, with his relentless drive, typically finishes the season on a high note, accumulating goals as the season progresses.

A Legacy of Greatness

Despite their contrasting approaches, Ronaldo and Messi have pushed each other to become legends. Their rivalry has elevated the game, captivating audiences with every match. Whether you admire Ronaldo's power and directness or Messi's finesse and playmaking, there's no denying that these two titans have left an indelible mark on football history. Their story is one of individual brilliance, a tale of two contrasting styles that have dominated the world stage.

Key Insights from the Data

Ronaldo has scored more total goals. Ronaldo scores more solo goals, while Messi has more assists. Ronaldo excels in headers, long-distance shots, counter-attacks, and penalties. Messi is more versatile, scoring from various positions on the field. Ronaldo has an edge in away goals; Messi thrives at home. Both players often find the net against Sevilla. Messi scores more in the final minutes and extra time. Messi starts seasons strong, while Ronaldo finishes stronger. I hope this story, along with the simplified data insights, helps you present your analysis in a clear and engaging way! Let me know if you'd like any further adjustments or insights added.