

PROJECT: TEAM DATABASE

Data Analysis of National Hockey League (NHL)

TEAM#11

EVAN HAHN

DYLAN BOLT

JONGBAE YOON

NOAH LYFORD

SANTHOSH VELAYUDHAN NAIR

Table of Contents

1. Project Scenario	3
1.1. Organization Description	3
1.2. Tables, Relationships, Themes, Business Logic	3
1.3. Relevant Columns, Transformations	4
1.4. Normalization	5
2. ER MODEL	7
3. Business Questions (SELECT Statements)	8
4. Visualizations	27
5. Stored Functions, Store Procedures, Triggers, and Indices	45
6. References	47

1. Project Scenario

1.1. Organization Description

Analytics and data analysis has always been a part of sports, but in recent years the use of data to improve player and team performance has become a major business. As the business has grown the demand for more technical analysis has grown as well. These sorts of data analyses are most prevalent in baseball and basketball, but other sports like soccer, football and hockey are entering the data revolution as well. Our goal in creating our database is to replicate something that resembles what a National Hockey League franchise might utilize to analyze their own and other team's players.

Our database compiles information about the National Hockey League from the 2010 season through the 2019/2020 season. The information includes player and team statistics, as well as characteristic data. While the database is not as complex as what a real NHL franchise might use, it is a valuable tool for answering difficult questions that would be impossible without a sophisticated database management system.

In our imagined business scenario, we are operating a single NHL franchise but need to utilize a database of the entire league in order to gain information about our opponents and their players. We can identify who the most valuable players in the league are and trade for them or recognize when a player is undervalued and sign that player to a contract. Similarly, we can identify when one of our own players is underperforming and might need to be removed from the team.

1.2. Tables, Relationships, Themes, Business Logic

We have 10 tables in this database. City, state, game, game_teams, game_goalies, game_skater_status, position, player, team, player. This project is

mainly focusing on the NHL teams' performance to better explain how the teams played in home and away conditions, the outcome of each game, how many goals scored, how many saves made etc. in various conditions. This also explains the performance of various players with various parameters.

The city and state tables are small, containing the city and state information of teams. We used one to many relationships to connect various tables. There is a many to many relationship between the team and game tables using a bridge table, game_scratches. Game_skater_stats and game_goalies tables hold mostly game information. Game_skater_stats has almost a million observations which give every performance data required for the analysis of any individual or team based performance.

1.3. Relevant Columns, Transformations

Two of the most important tables are game_skater_stats and game tables, since these two tables have data related to goals. In any sport, winning is the single most important thing and not unlike the other sports, one team must score more than the other team to win in hockey. As most of our data analysis involves goals (e.g. goals, shots, home_goals, and away_goals, etc), these two tables are two of the most important tables in our database.

Likewise, the columns involving the goal data are important in our project. With the goal data, we are able to find out which teams and players score the most and combining with other columns, we can figure out information like efficiency and productivity. With the home_goals and away_goals data, we are also able to find out which teams and players have similar performance regardless of where they play.

After we downloaded the data from Kaggle, we cleaned some of the data and transformed some data types. First, we determined what data to use and removed columns not necessary to our business model, so we could optimize the data. Then, we made some transformations from its original form. For example, we transformed date type to be imported into MYSQL, and Boolean type from true/false to 0/1 to be imported and indicate win / lose.

1.4. Normalization

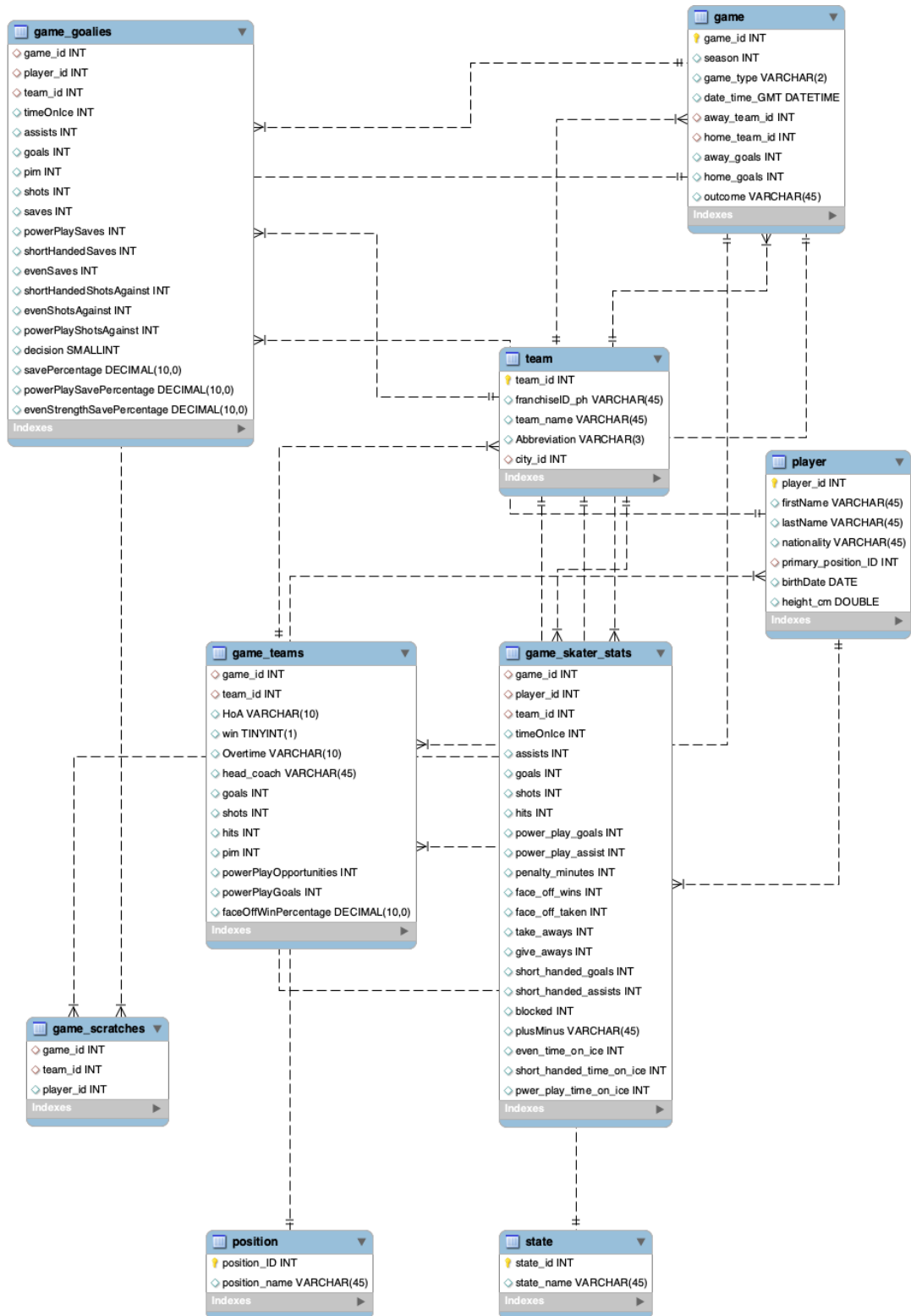
We started this project by importing data in .csv format from Kaggle, but the data was organized in a way where it was not normalized. For one, we downloaded a giant csv that had to be put in 1NF. We accomplished this by splitting the columns with multiple values into multiple columns with each of the values listed separately. Now we have Normal Form 1.

Next, we looked onward to 2NF. In order to get our data to the second Normal Form, it has to be in the first Normal Form and all primary keys must consist of one column. For example, if we had a table that included teams and their cities, then the tuple (team,city) would have to be unique. To solve this, we split the tables up into multiple tables each with a primary key that makes more sense. We split the team table into team and city to solve this issue.

Lastly, we had to get our data into 3NF. Third Normal Form means that the data is already in Second Normal Form and has no functional dependencies. Functional dependencies would cause issues in our database if we update one column's information, but forget to update the columns that are functionally dependent on this table. In order to solve this problem, we assigned IDs to players, teams, cities, and

games. We then split our tables up even more so that there would be no functional dependencies.

2. ER MODEL



3. Business Questions (SELECT Statements)

- Evan Hahn

```
# -Average Shots on Goal per season

select season, sum(shots) as numShots, count(distinct gss.game_id) as
`Game Count`, (sum(shots)/count(distinct gss.game_id)) as `Avg Shots
Per Game`

from game_skater_stats as gss

join game as g on gss.game_id = g.game_id

group by g.season

order by (sum(shots)/count(distinct gss.game_id)) desc;
```

This query gets the average shots on goal per season; this highlights which season had the most goals and which had the fewest. We can see that as time went on, teams were more and more aggressive with getting shots on goal., but we can also see that the average shots per game stayed mostly the same, due to the `Game Count` column having drastically different counts, but the averages still staying mildly close to each other. We can not tell if the lower averages for the earlier seasons are due to lack of data, or if teams were actually less aggressive in those seasons.

```
#Average Power play goals per season.

Select season, sum(power_play_goals) as Ppgoals , count(distinct
gss.game_id) as `Game Count`, (SUM(power_play_goals)/count(distinct
gss.game_id)) as `Avg Power Play Goals Per Game`

from game_skater_stats as gss

join game as g on gss.game_id = g.game_id

group by g.season;
```


This query builds slightly off the last one, with looking at Power Play goals instead of shots. As seen before, there were more games in the later season, but the number of Average Power Play Goals stays relatively the same. Meaning that no matter how many games are played, you are likely to see the same amount of power play goals in any season in any given game.

```
DELIMITER &&

create function get_winner(
    gameid integer
)
returns integer
deterministic
begin
    declare winner integer;

    select (case
        when outcome like 'home win%' then home_team_id
        when outcome like 'away win%' then away_team_id
        else null
    end) as winTeam
    into winner
    from game
    where game_id = gameid;

    return winner;
end; &&

#Average penalty_minutes per win/loss
# winners 8.7575
select avg(penMins)
from (select sum(penalty_minutes) as penMins
```

```

from game as g
join game_skater_stats using(game_id)
where (get_winner(game_id)) = team_id
group by game_id) as winnerTable;

#Losers 9.099 mins
select avg(penMins)
from (select sum(penalty_minutes) as penMins
from game as g
join game_skater_stats using(game_id)
where (get_winner(game_id)) != team_id
group by game_id) as loserTable;

```

These two queries work in tandem with each other and the associated stored function 'get_winner'. The first one gets the average amount of time a player is in a penalty box per game for only the winning teams, and the other is the same for losing teams. Winning teams spend on average 8.75 minutes and losing teams spend 9.09 minutes, meaning that the longer you have a player in the penalty box, the more likely you are to lose.

```

# Most winning team
select c.city_name, t.team_name, count(team) as wins
from (select (case
              when outcome like 'home win%' then home_team_id
              when outcome like 'away win%' then
away_team_id
              else null
            end) as team
from game) as games

```

```

join team as t on games.team = t.team_id
join city as c on t.city_id = c.city_id
group by team
order by wins desc;

```

This query looks a lot more complicated than it actually is. This query just calculates all wins by any given team and adds them up, leaving us with the most winning teams in our dataset. We can see a very large spread of wins across all teams, but this is largely due to the fact that many new teams have been brought into the NHL during the seasons our dataset contains. For instance, the Las Vegas Golden Knights first season was 2017-2018, the last season our database records.

```

#Average penalty time per game per team per season
Select season, c.city_name, t.team_name, avg(total_pen_mins) as
'Average Penalty Minutes'
from (
    select season, gss.game_id, team_id, sum(penalty_minutes) as
total_pen_mins from game_skater_stats as gss
    join game as g on gss.game_id = g.game_id
    group by game_id, team_id
) as penMinGame
join team as t on penMinGame.team_id = t.team_id
join city as c on c.city_id = t.city_id
group by season, t.team_id
order by t.team_id, season;

```

This query calculates every team's average time spent in the penalty box per game for every season they played in. The team who spent the most time in the penalty

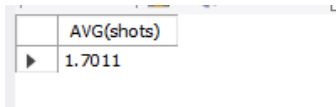
box was The Winnipeg Jets during the 2013-2014 season, with an average of 17.25 minutes per game.

– **Dylan Bolt**

#Average skater Shots on goal across league?

#Simple query finding the average shots for skaters across the league. This is useful for setting a benchmark for new recruits.

```
SELECT AVG(shots)
FROM game_skater_stats;
```



AVG(shots)
1.7011

#Average Shots on goal per team?

#Used a join and group by to find average shots per team. This is helpful for a specific team to see where they rank among all teams.

```
SELECT t.team_name, AVG(s.shots)
FROM game_skater_stats AS s
JOIN team AS t
USING(team_id)
GROUP BY t.team_name;
```

	team_name	AVG(s.shots)
▶	Devils	1.5103
	Islanders	1.7310
	Rangers	1.6830
	Flyers	1.7277
	Penguins	1.8270
	Bruins	1.8111
	Sabres	1.6204
	Canadiens	1.7049
	Senators	1.6798
	Maple Leafs	1.7429
	Hurricanes	1.7399
	Panthers	1.7551
	Lightning	1.6889
	Capitals	1.6745
	Blackhawks	1.7995
	Red Wings	1.6506
	Predators	1.7509
	Blues	1.6920
	Flames	1.6688

#Least winning team?

#Used join, group by, and order by to find the worst team in our database at winning games. Could be used to direct funding or advertisements for the league.

```
SELECT t.team_name, SUM(gt.win)
FROM team AS t
JOIN game_teams AS gt
USING(team_id)
GROUP BY t.team_name
ORDER BY SUM(gt.win)
LIMIT 1;
```

	team_name	SUM(gt.win)
▶	Thrashers	34

#Average shots per game for skaters for most winning team?

#Utilized a subquery to find average shots for most winning team. Very useful as a goal for other teams.

```
SELECT t.team_name, AVG(gt.shots)
FROM game_teams AS gt
JOIN team AS t
WHERE t.team_name = (
    SELECT t.team_name
    FROM team AS t
    JOIN game_teams AS gt
    USING(team_id)
    GROUP BY t.team_name
    ORDER BY SUM(gt.win) DESC
    LIMIT 1
);
```

	team_name	AVG(gt.shots)
▶	Lightning	30.6267

#Most home wins by a head coach?

#Used where, group by, and order by. Could be used by teams to identify the coach

with the most wins for possible recruitment.

```
SELECT HoA, head_coach, SUM(win) as wins
FROM game_teams
WHERE HoA = "home"
GROUP BY head_coach
ORDER BY wins DESC
LIMIT 1;
```

	HoA	head_coach	wins
▶	home	Barry Trotz	318

- JONGBAE YOON

1. Top 10 most efficient players (i.e. goal / shots) in the year 2017

```
select s.player_id, CONCAT(p.firstName, p.lastName) AS player_Name,
      (goals / shots) AS efficiency
FROM game_skater_stats as s
LEFT JOIN game as g
      USING(game_id)
LEFT JOIN player as p
      USING(player_id)
WHERE YEAR(date_time_GMT) = 2017
ORDER BY efficiency DESC
LIMIT 10;
```

	player_id	player_Name	efficiency
▶	8469665	JohnnyOduya	1.0000
	8476923	DamonSeverson	1.0000
	8476207	BrianGibbons	1.0000
	8476459	MikaZibanejad	1.0000
	8479337	AlexDeBrincat	1.0000
	8479337	AlexDeBrincat	1.0000
	8476466	SvenBaertschi	1.0000
	8476440	MarkusGranlund	1.0000
	8475179	DmitryKulikov	1.0000
	8476460	MarkScheifele	1.0000

This query shows top 10 most efficient players in the year 2017. I defined the efficiency as goal scored divided by shots attempts.

2. players who played more than 120,000 hours in the year 2017 *TABLEAU

```
select distinct(s.player_id) as id, CONCAT(p.firstName, p.lastName)
AS Full_Name, sum(timeOnIce) as playTime
FROM game_skater_stats as s
LEFT JOIN game as g
      USING(game_id)
LEFT JOIN player as p
      USING(player_id)
```



```

WHERE YEAR(date_time_GMT) = 2017

group by id, Full_Name

HAVING sum(timeOnIce) > 120000

ORDER BY playTime DESC;

```

	id	Full_Name	playTime
▶	8474563	DrewDoughty	138540
▢	8470600	RyanSuter	137280
	8475171	OliverEkman-Larsson	125439
▢	8474565	AlexPietrangelo	124110
	8477495	SethJones	123851
▢	8470613	BrentBurns	122868
	8475218	MattiasEkholm	120929
▢	8471274	AlexGoligoski	120689
	8475167	VictorHedman	120365

This query shows players id and name who played more than 120,000 minutes in the year.

3. TOP 10 scorers in the league in 2017

```

select player_id, concat(firstname, lastName) AS fullName, sum(goals)
from game_skater_stats
join player
using (player_id)
join game
using (game_id)
where year(date_time_GMT) = 2017
group by player_id, concat(firstname, lastName)
order by sum(goals) desc
LIMIT 10;

```

	player_id	fullName	sum(goals)
▶	8471214	AlexOvechkin	197
▢	8475765	VladimirTarasenko	156
	8471675	SidneyCrosby	150
▢	8475794	TylerSeguin	148
	8474141	PatrickKane	147
▢	8473994	JamieBenn	145
	8475166	JohnTavares	142
▢	8473419	BradMarchand	141
	8476453	NikitaKucherov	140
▢	8471215	EvgeniMalkin	136

This query shows the top 10 scorers in the league in 2017. I joined three tables and used limit to indicate top 10 players.

4. Number of players by height (cm) *TABLEAU

```
select height_cm, count(player_id)
FROM player
group by height_cm;
```

height_cm	count(player_i...
190.5	185
185.42	266
182.88	218
195.58	38
205.74	1
187.96	244
193.04	106
180.34	154
177.8	83
170.18	5
175.26	40
198.12	29
200.66	4
162.56	1
203.2	3
172.72	15
167.64	1
165.1	1

This query shows the number of players in each height range. This has normal distribution with approximate mean 185cm (6'1").

5. Average goals by position in each year

```
select subq_1.position_name as position, YEAR(subq_1.time) as year,
sum(subq_1.goals) as goals, count(numplayer) as 'number of players',
sum(subq_1.goals)/count(numplayer) as average
from (
select po.position_name, gm.date_time_GMT as time, sum(gs.goals) as
goals
from game_skater_stats as gs
join game as gm
using (game_id)
join player as pl
using (player_id)
```

```

join position as po
on pl.primary_position_ID = po.position_ID
group by po.position_name, time
) as subq_1
join(
select po.position_name, count(player_id) as numPlayer
from player as pl
join position as po
on pl.primary_position_ID = po.position_ID
group by po.position_name
) subq_2
using (position_name)
group by position_name, YEAR(subq_1.time)
order by position, year;

```

	position	year	goals	number of players	average
►	Center	2013	418	209	2.0000
◄	Center	2014	1160	417	2.7818
◄	Center	2015	2337	743	3.1454
◄	Center	2016	2506	739	3.3911
◄	Center	2017	2680	768	3.4896
◄	Center	2018	1633	424	3.8514
◄	Defense	2013	173	209	0.8278
◄	Defense	2014	571	417	1.3693
◄	Defense	2015	1053	743	1.4172
◄	Defense	2016	1014	739	1.3721
◄	Defense	2017	1097	768	1.4284
◄	Defense	2018	638	424	1.5047
◄	Left Wing	2013	255	209	1.2201
◄	Left Wing	2014	856	417	2.0528
◄	Left Wing	2015	1656	743	2.2288
◄	Left Wing	2016	1623	739	2.1962
◄	Left Wing	2017	1844	768	2.4010
◄	Left Wing	2018	977	424	2.3042
◄	Right Wing	2013	267	209	1.2775
◄	Right Wing	2014	805	417	1.9305
◄	Right Wing	2015	1426	743	1.9192
◄	Right Wing	2016	1427	739	1.9310
◄	Right Wing	2017	1605	768	2.0898
◄	Right Wing	2018	744	424	1.7547

This shows the average goals by position in each year. I used two subqueries, one in the from clause and the other in the join clause.

– Noah Lyford

1. Show total goals by position. This query shows the total number of goals scored by position from our entire database. Obviously, the way to win games is by scoring more goals than the other team. This query is useful because it allows us to see what positions are responsible for the most goals. Now that we know which positions provide the most goals, we have a better understanding of where we should be allocating salary.

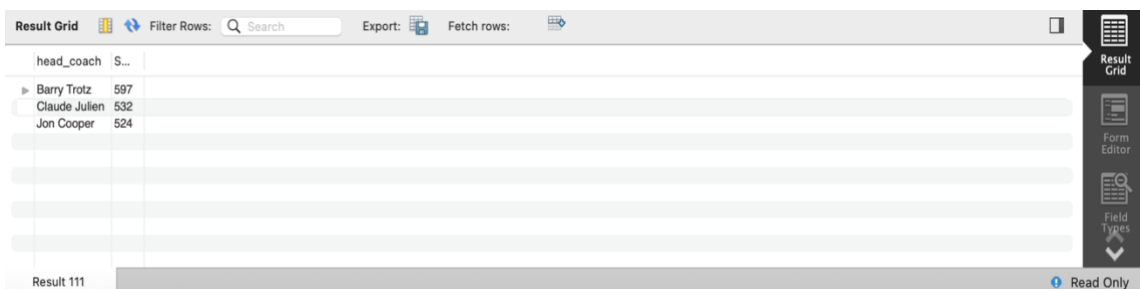


The screenshot shows a database interface with a 'Result Grid' header. Below the header, there is a table with two columns: 'position_name' and 'Total Goals'. The table contains four rows of data. To the right of the table, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom right, there is a 'Read Only' status indicator.

position_name	Total Goals
Center	10734
Left Wing	7211
Right Wing	6274
Defense	4546

2. Show the top 3 head coaches in terms of wins

This query shows which coach has the most wins in our database. While players are the most important team asset that contributes to winning games, coaches are very important as well. The best way to judge a coach is by his/her number of wins, from this query we have an idea who the best coaches in the league are.



The screenshot shows a database interface with a 'Result Grid' header. Below the header, there is a table with two columns: 'head_coach' and 'Wins'. The table contains three rows of data. To the right of the table, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom right, there is a 'Read Only' status indicator.

head_coach	Wins
Barry Trotz	597
Claude Julien	532
Jon Cooper	524

3. Show total wins from 2018/2019 season.

This query shows us the team_id, team_name and the number of wins from the most recent season. This will give us a better sense of which teams are good in the league and who we will be competing against to make it to the playoffs. The win total includes wins in the playoffs as well.

team_id	team_name	wins
6	Bruins	64.0000
14	Lightning	62.0000
19	Blues	61.0000
28	Sharks	56.0000
12	Hurricanes	54.0000
29	Blue Jackets	53.0000
2	Islanders	52.0000
15	Capitals	51.0000
20	Flames	51.0000
25	Stars	50.0000
52	Jets	49.0000

4. Show average saves per game, top 5

This query utilizes the goalies states table and the player table and shows us the goalies in our database that have the highest average saves. Having a good goalie is crucial for winning games and having a goalie who saves a high number of shots per game would be really valuable for our team. We also now know when we are facing a really good goalie and could change our strategy against him.

Name	averageSaves
Malcolm Subban	36.0000
J-F Berube	33.0000
Semyon Varlamov	32.2778
Robin Lehner	31.9524
James Reimer	31.8000

5. Show top 3 scorers from each season

This query utilizes a subquery, and window function to rank the top 3 three goal scorers from each season. This type of query is useful for when we are playing against these players. We need to be able to identify when we are going against a prolific goal

scorer and could change our defense to stop him. We also could use this list to target players for potential trades.

season	Name	seasongoals	ranking	
20172018	Alex Ovechkin	49	1	
20172018	Patrik Laine	44	2	
20172018	William Karlsson	43	3	
20162017	Sidney Crosby	44	1	
20162017	Auston Matthews	40	2	
20162017	Nikita Kucherov	40	2	
20152016	Alex Ovechkin	50	1	
20152016	Patrick Kane	46	2	
20152016	Jamie Benn	41	3	
20142015	Alex Ovechkin	53	1	
20142015	Steven Stamkos	43	2	

Result 114

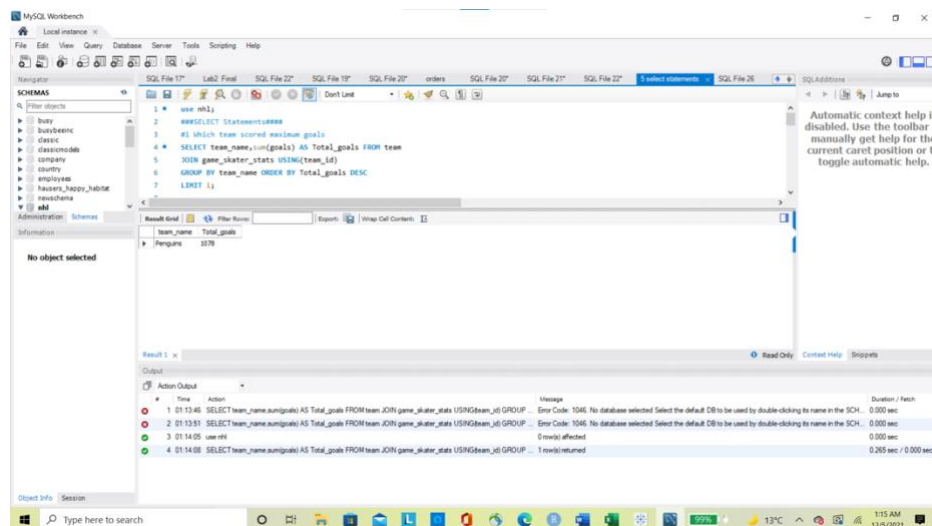
Read Only

– **Santhosh Velayudhan Nair**

Which team scored maximum goals

This statement is used to find which team scored maximum goals. Used JOIN and grouping functions.

```
SELECT team_name,sum(goals) AS Total_goals FROM team
JOIN game_skater_stats USING(team_id)
GROUP BY team_name ORDER BY Total_goals DESC
LIMIT 1;
```

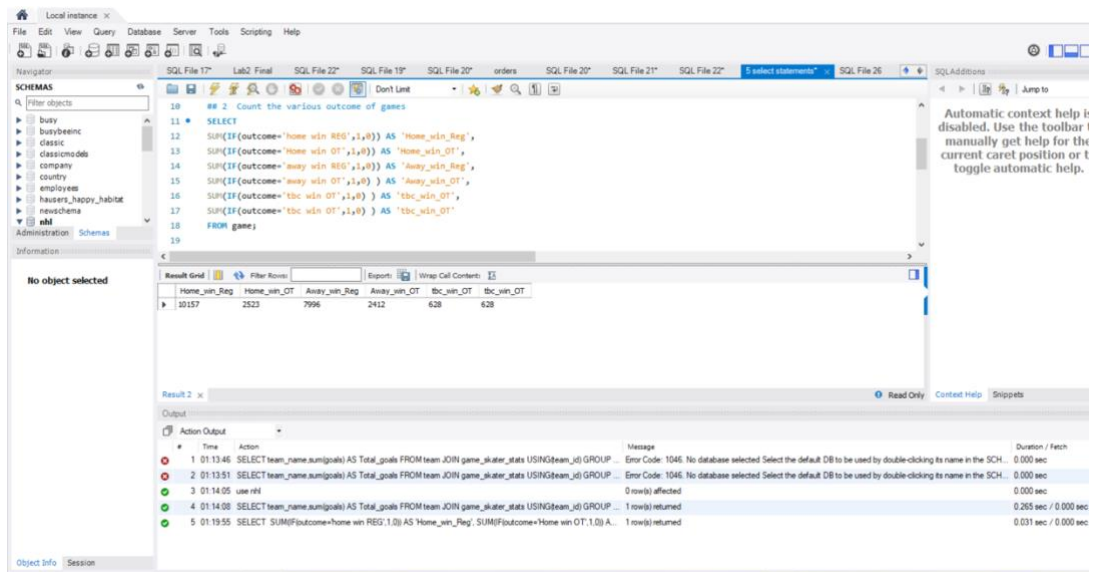


This statement shows how the win for all the teams are distributed among home regular, home OT, away regular etc. Used 6 IF statements and calculated field to count each category. Game table is used. This query is same as the visualization 2

```
SELECT
SUM(IF(outcome='home win REG',1,0)) AS 'Home_win_Reg',
SUM(IF(outcome='Home win OT',1,0)) AS 'Home_win_OT',
SUM(IF(outcome='away win REG',1,0)) AS 'Away_win_Reg',
SUM(IF(outcome='away win OT',1,0) ) AS 'Away_win_OT',
SUM(IF(outcome='tbc win OT',1,0) ) AS 'tbc_win_OT',
```

```
SUM(IF(outcome='tbc win OT',1,0) ) AS 'tbc_win_OT'

FROM game;
```



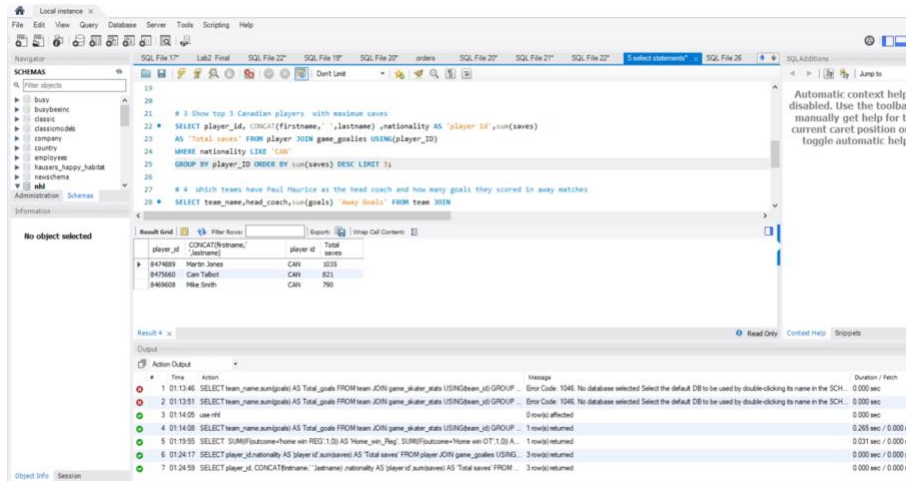
Show top 3 players from Canada with maximum saves

This select statement who are the top three Canadian players who made maximum saves. Here game_goalies and player tables are used with CONCAT, JOIN, GROUP and ORDER and calculated fields. Same is done in Tableau visualization 3

```
SELECT player_id,nationality AS 'player id',sum(saves)
AS 'Total saves' FROM player JOIN game_goalies USING(player_ID)
WHERE nationality LIKE 'CAN'
```



```
GROUP BY player_ID ORDER BY sum(saves) DESC LIMIT 3;
```

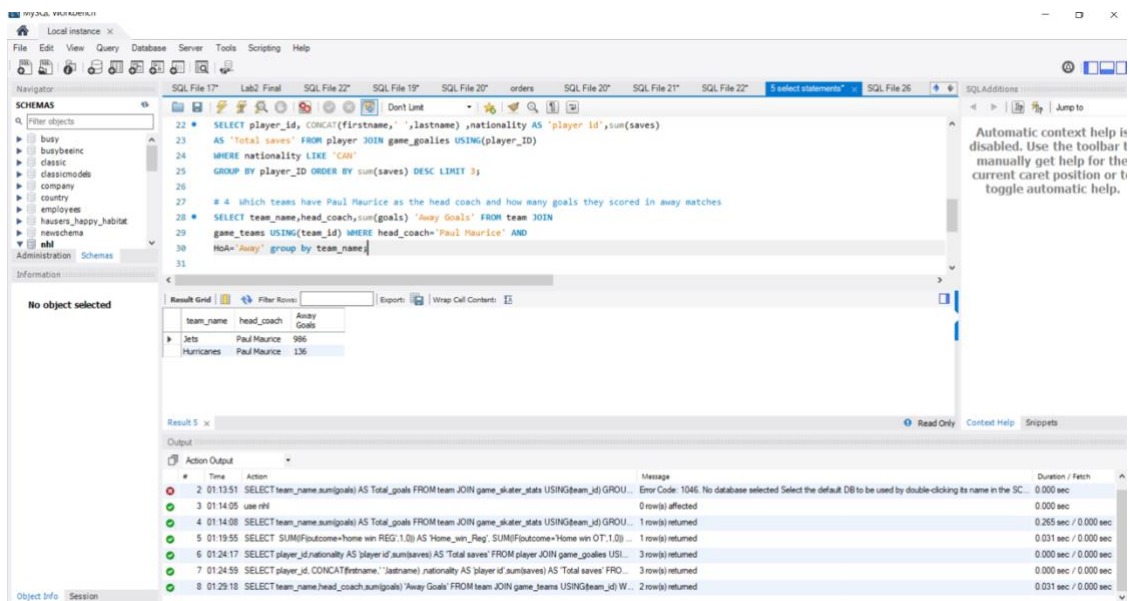


Which teams have Paul Maurice as the head coach and how many goals they scored in away matches. This shows which teams have Paul Morris worked as the head coach and how many goals these teams scored in away matches. This has JOIN, WHERE and GROUP BY.

```

SELECT team_name, head_coach, sum(goals) 'Away Goals' FROM team JOIN
game_teams USING(team_id) WHERE head_coach='Paul Maurice' AND
HoA='Away' group by team_name;

```

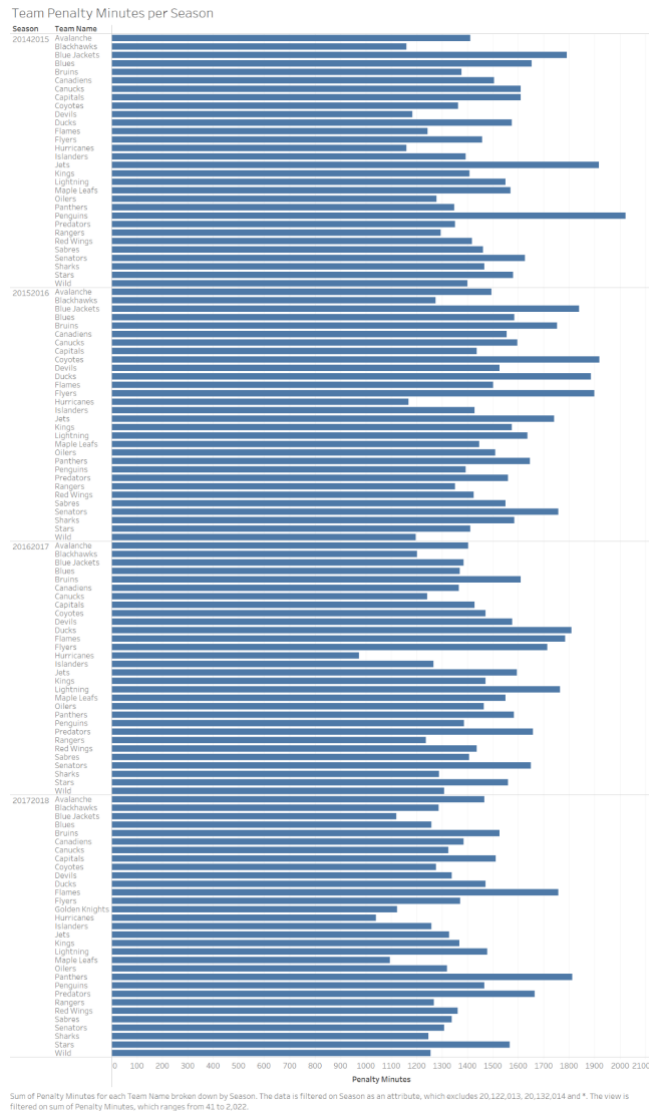


```
SELECT count(player_id) FROM player
WHERE height_cm >
(SELECT AVG(height_cm) FROM player WHERE nationality='CAN')
AND nationality= 'USA';
```



4. Visualizations

– Evan Hahn



This chart shows the average team penalty minutes by season. I dropped the first 2 seasons in the dataset because they were substantially lower than the rest, meaning that either the dataset was incomplete or missing. As you can see, certain teams tend to be on the higher end of this list more frequently than others do.

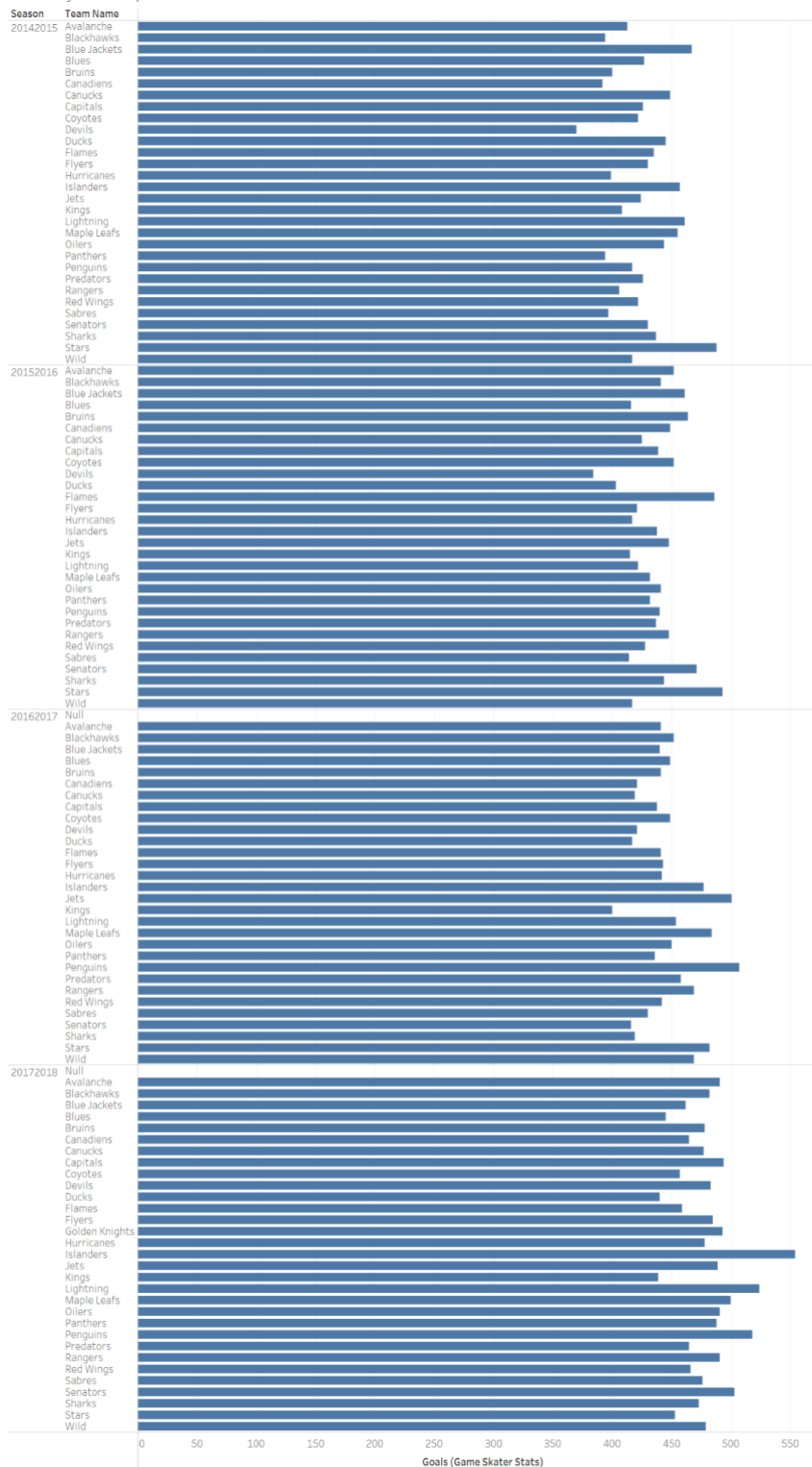
This shows the teams that score the most in our dataset. Unfortunately for some of the newer teams, it is hard to catch up to the mainstay teams like the Lightning, Islanders, Maple Leafs and Penguins.

Most Scoring Teams



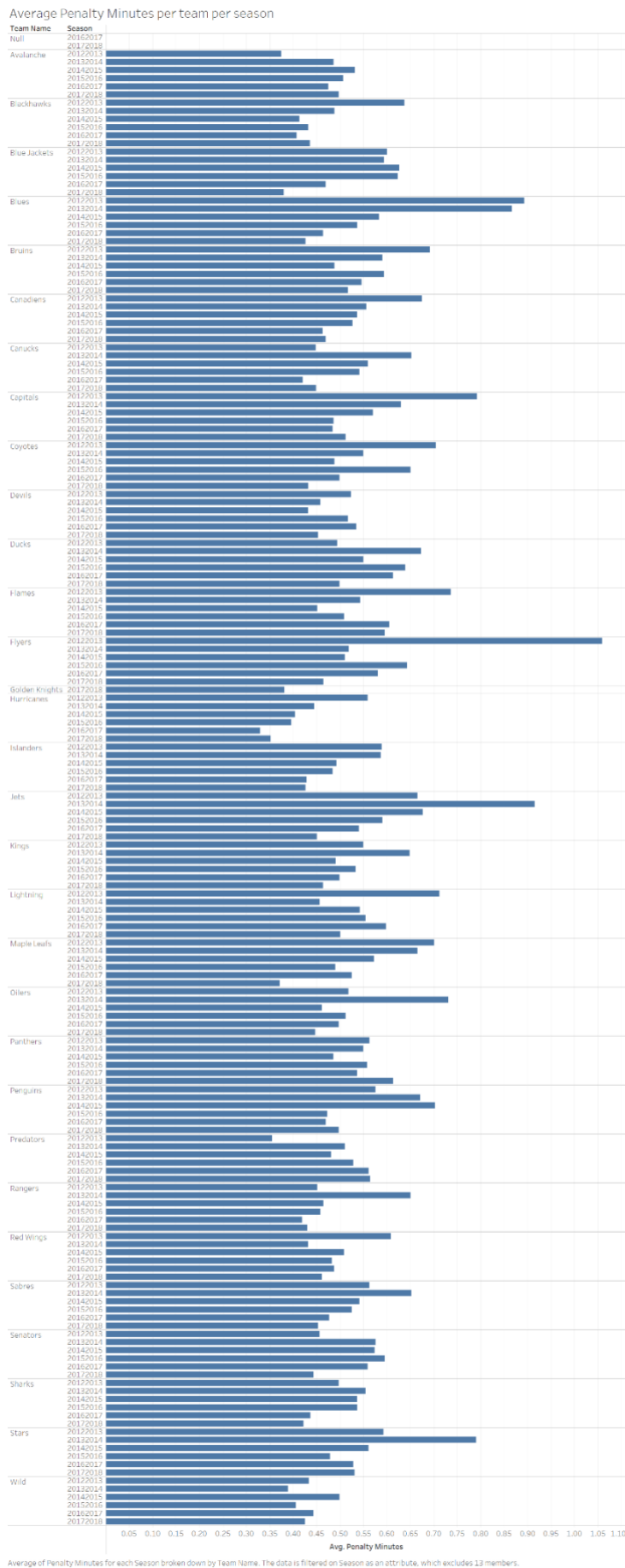
Team Name. Size shows details about sum of Goals (Game Skater Stats). The data is filtered on sum of Goals (Game Skater Stats), which keeps non-Null values only. The view is filtered on Team Name, which excludes Golden Knights.

Goals by season per Team



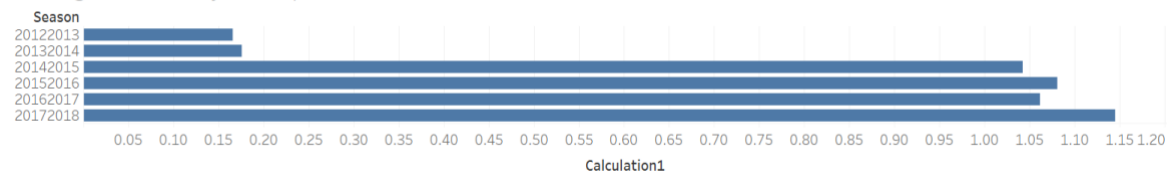
Sum of Goals (Game Skater Stats) for each Team Name broken down by Season. The data is filtered on Season as an attribute, which excludes 15 members.

This Graph shows the number of goals scored by each team every season. I removed the 2012-13 and the 2013-14 seasons as they had far fewer data points and skewed the graph in an unreadable way. As you can see there is an increasing amount of goals scored on average, possibly due to the higher aggression levels from teams as time goes on.



This graph shows that average penalty minutes per season per team. It can tell us who the most penalized teams were every season and also provides us information on who is actually just a heavily penalized team. You can also see a few spikes as teams get overly aggressive throughout their careers in this dataset.

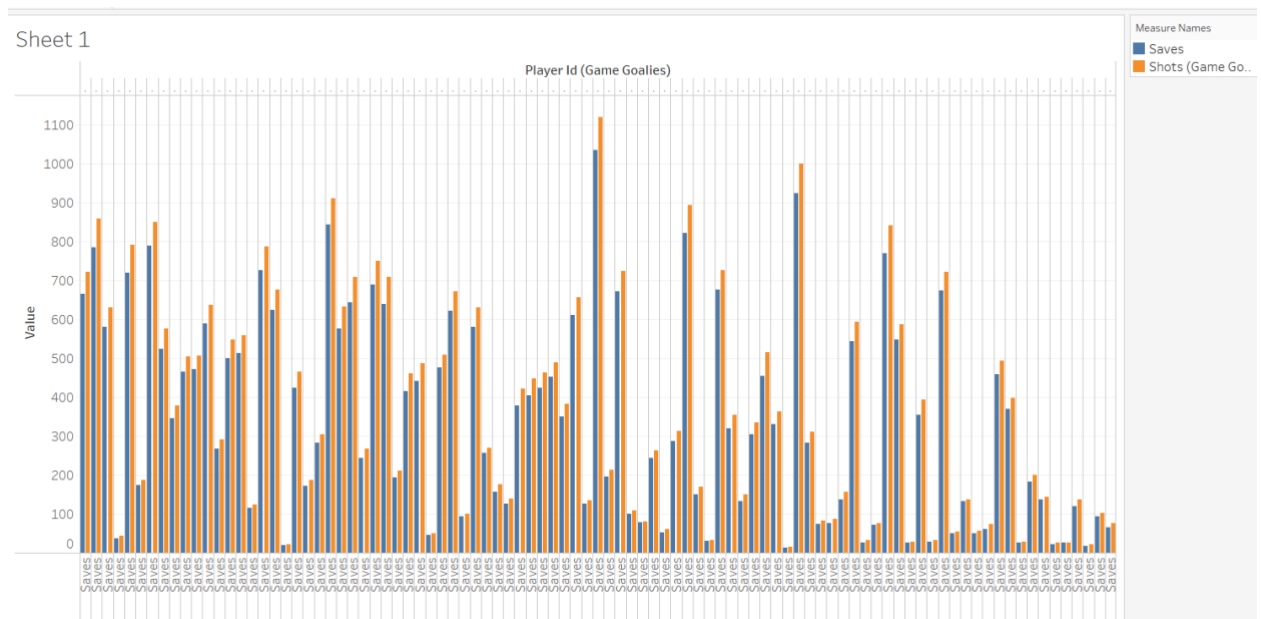
Average Power Play Goals per season



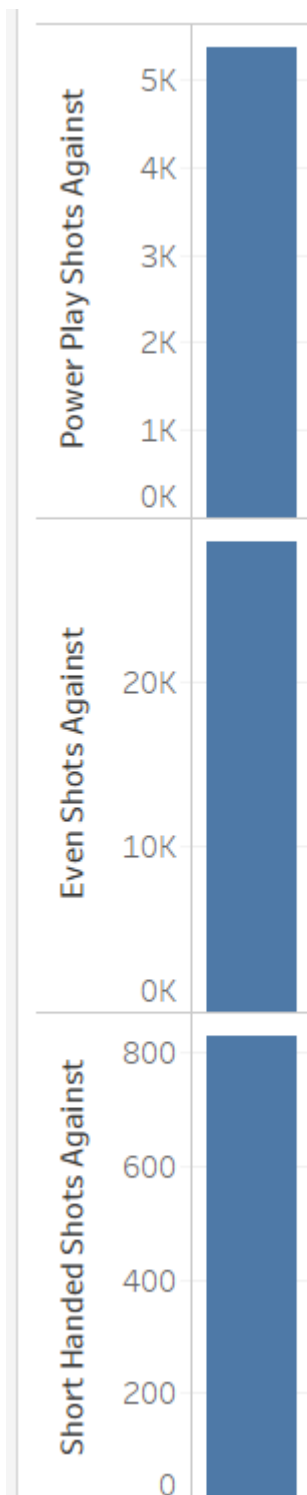
Calculation1 for each Season. The data is filtered on Season as an attribute, which excludes 13 members.

This visualization shows the average power play goals per game per season. The early seasons of our dataset (12-13 and 13-14) have significantly lower values, most likely due to incomplete data, but you can see that there is a positive trend in average goals, steadily increasing from a little over 1 per game in the 14-15 season to almost 1.15 per game in the 17-18 season. It would be interesting to see if that trend continues through to the present season or not.

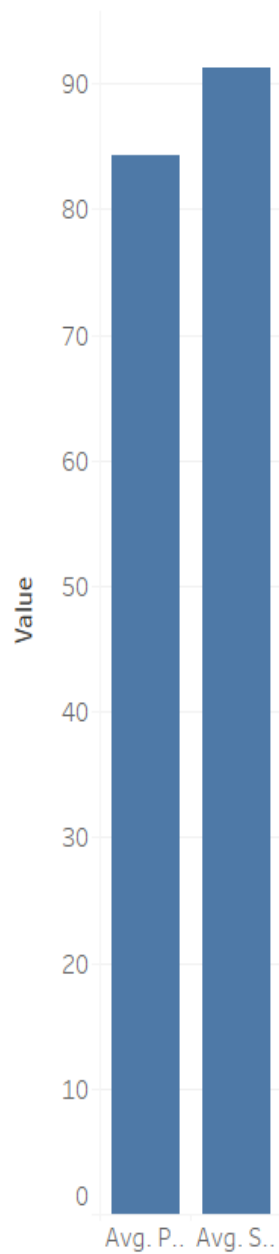
– **Dylan Bolt**



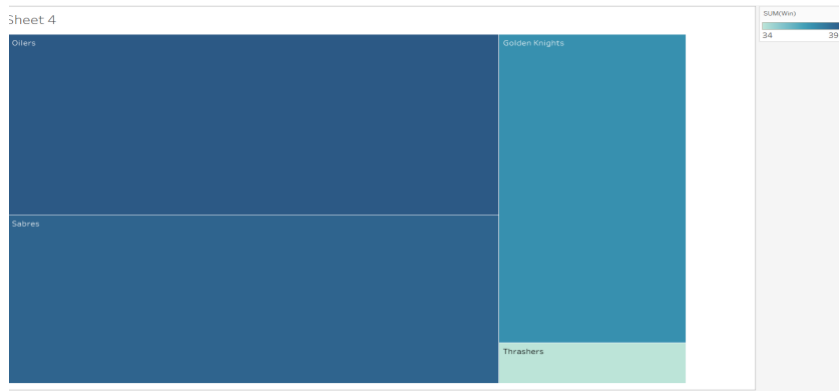
I joined game_teams with game_goalies, game_skater_stats, and team to form this visualization. I wanted to display the sum of shots vs the sum of saves for each goalie. I used a bar plot.



I joined game_teams with game_goalies, game_skater_stats, and team to form this visualization. I wanted to compare all of the different types of shots by the amount that they occurred. Again I used a bar chart.



I joined game_teams with game_goalies, game_skater_stats, and team to form this visualization. I wanted to compare power save percentage and regular save percentage. I used a bar chart.

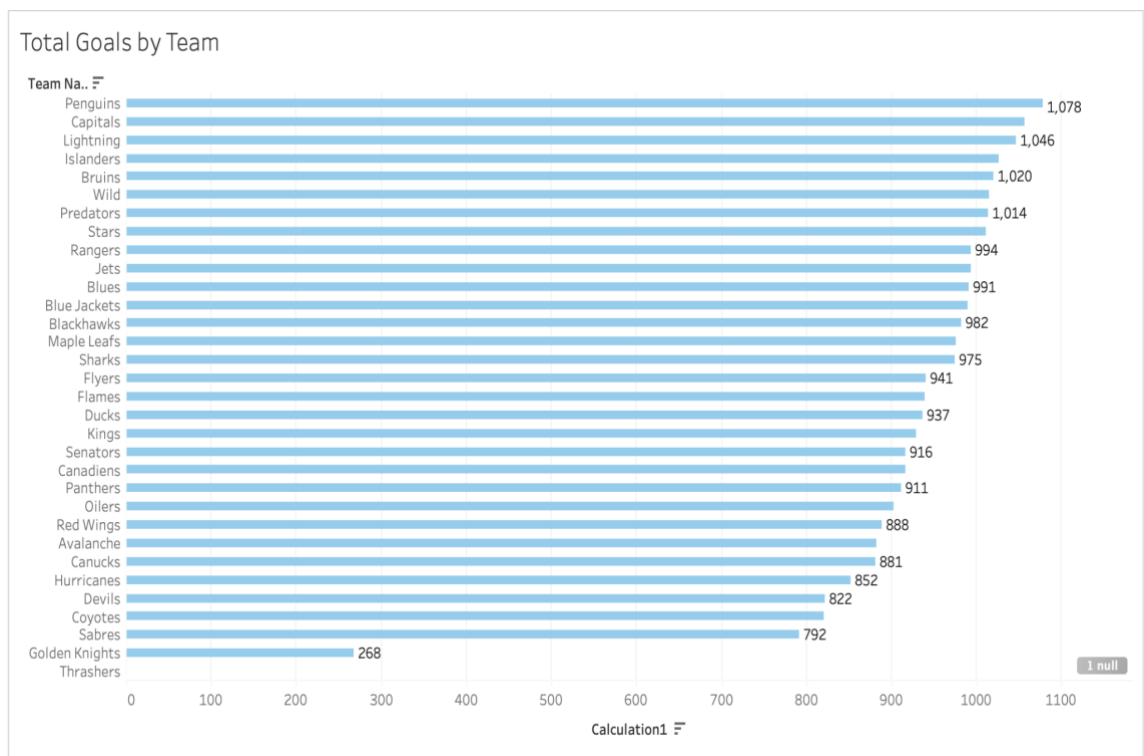


I joined game_teams with game_goalies, game_skater_stats, and team to form this visualization. I wanted to see the sum of wins for the top 5 highest winning teams. I used a heat map.

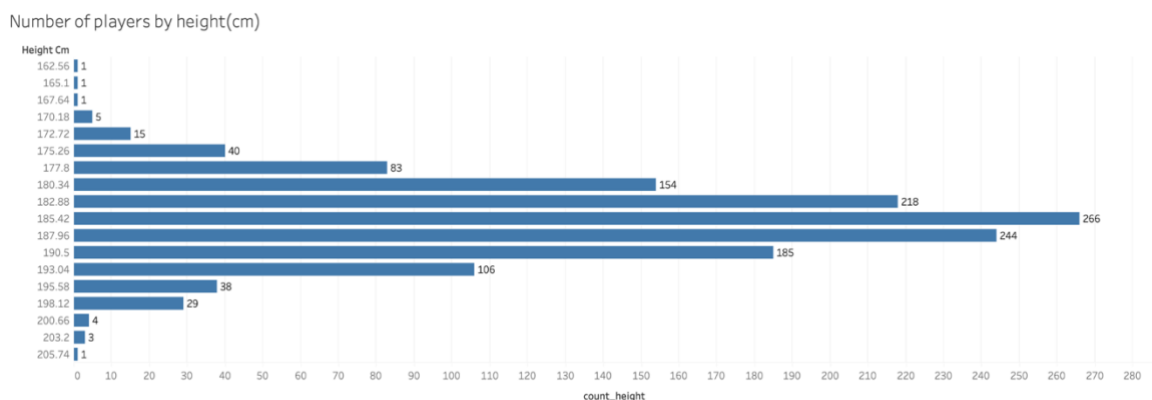
Team Na..	
Blues	30.422
Bruins	32.421
Capitals	30.008
Lightning	30.468
Penguins	32.370

I joined game_teams with game_goalies, game_skater_stats, and team to form this visualization. I wanted to see the average shots of the top 5 most winning teams. I used a text table.

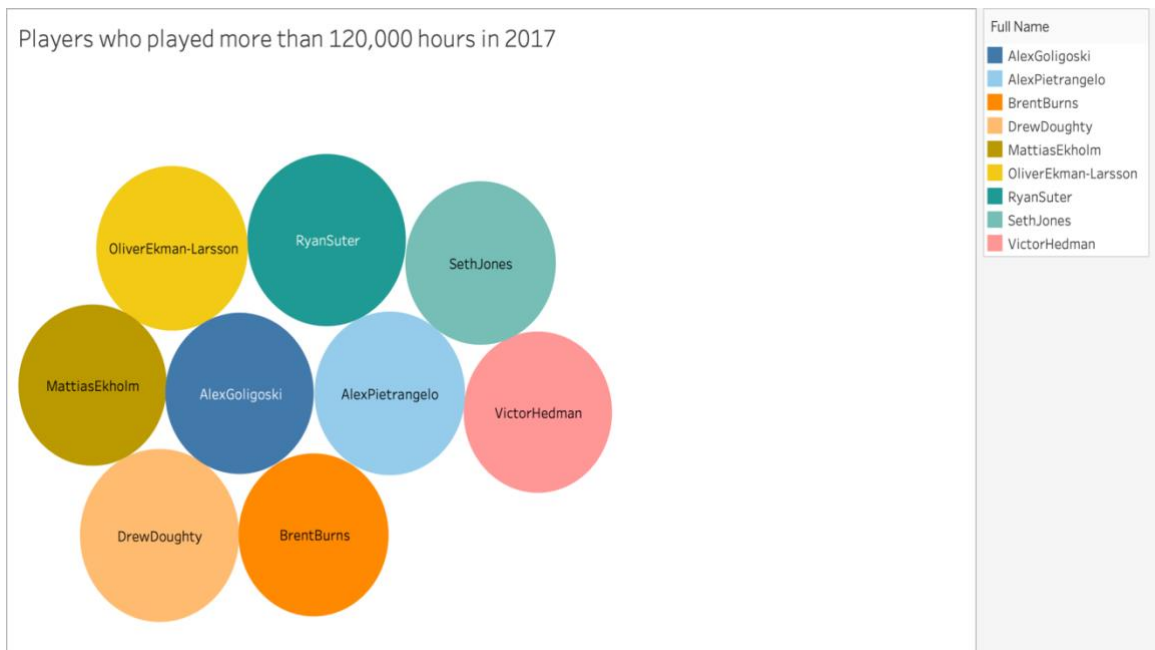
– **Jongbae Yoon**



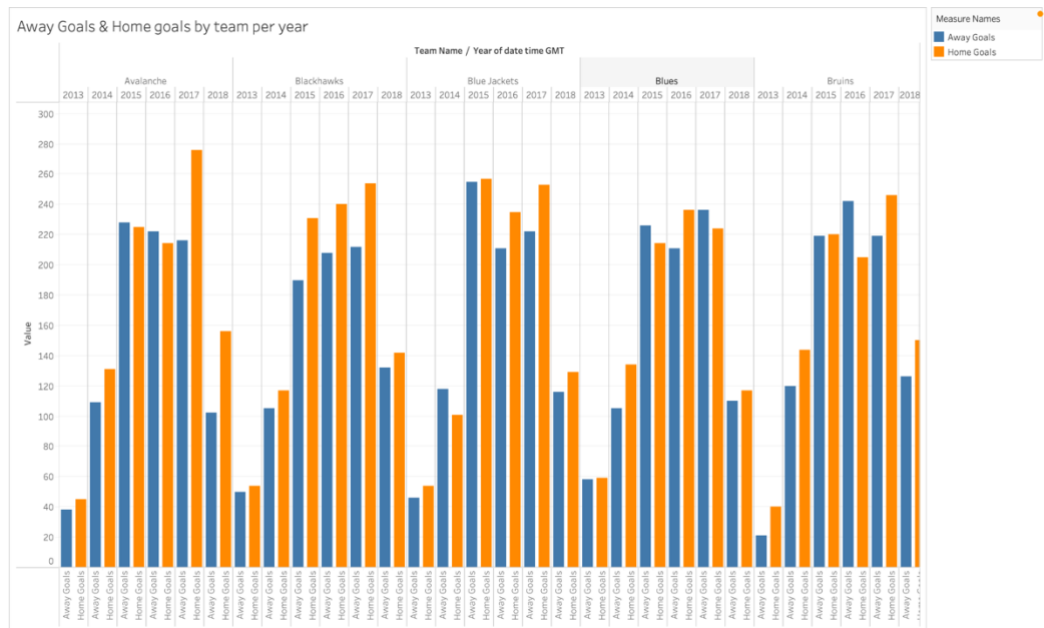
This visualization shows the total goals by team. I wanted to show the total goals by each team because it shows which team is more aggressive and this means their games are more fun to watch. I joined game_skater_stats table and teams table using team_di, and calculated the sum of the goals and order by it. I used bar chart



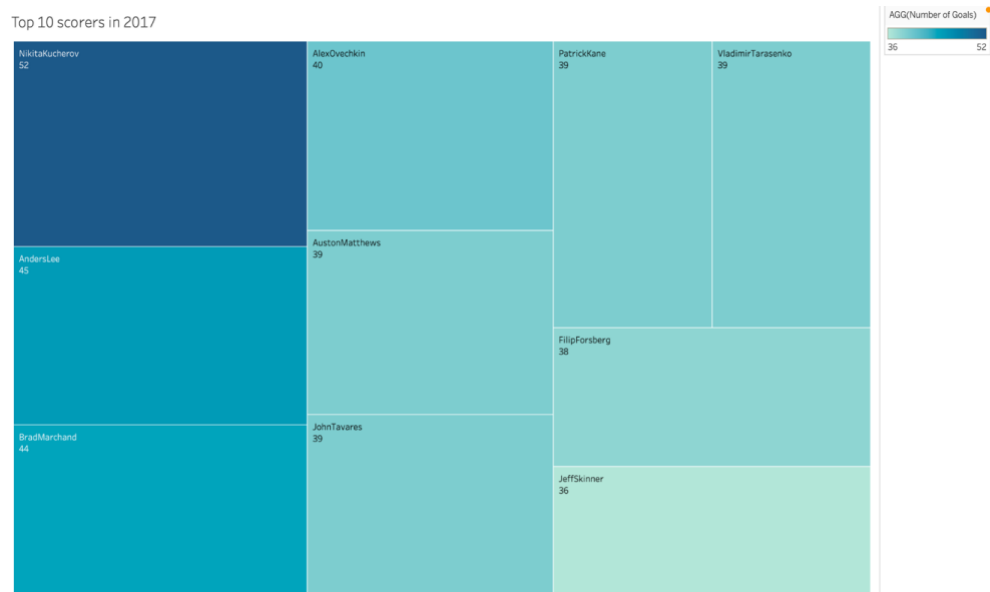
This visualization shows the distribution of players' height. It has normal distribution, and the average height is approximately 185.42 cm, which is 6'1'' . I used player table and bar chart.



This visualization shows the players who played more than 120,000 minutes in 2017. This data shows the most durable players in the league. I joined game_skater_stats, game, player tables using player_id, and used bubble chart.



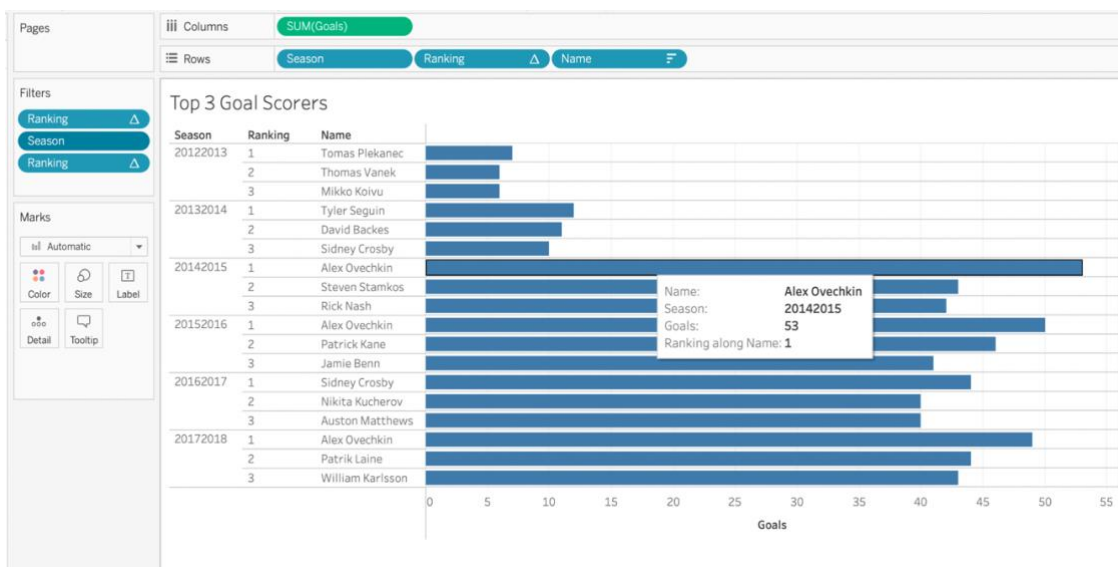
This visualization shows the home goals and away goals of each team per year. This shows that most of them score more at home games than away games. I joined team table and game table and used bar chart.



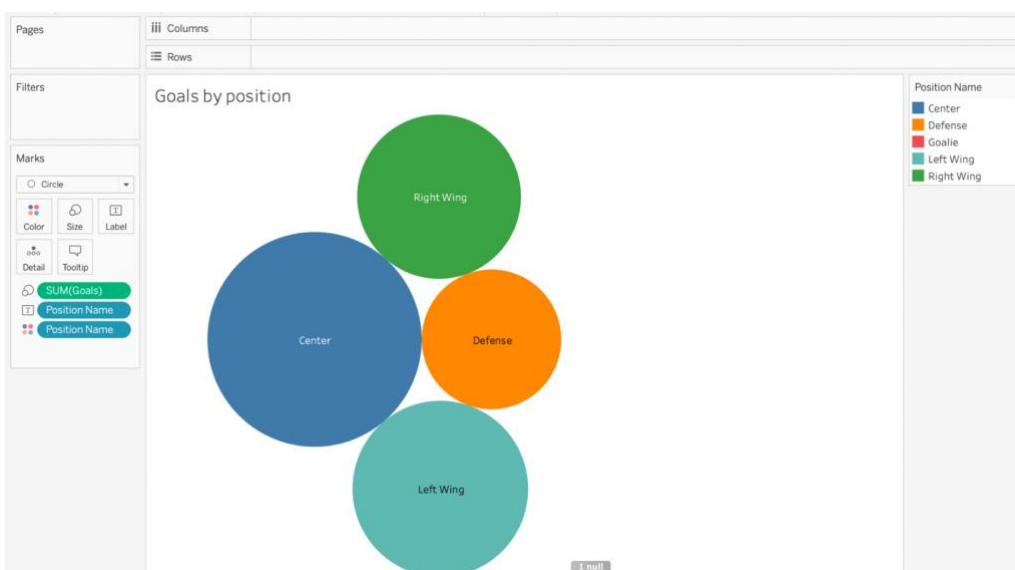
This visualization shows the top 10 scorers in the year of 2017. Top scorers always get the attention from the fans and the media and have a great potential to be superstars. I joined player and game_skaters_stat tables.

- Noah Lyford

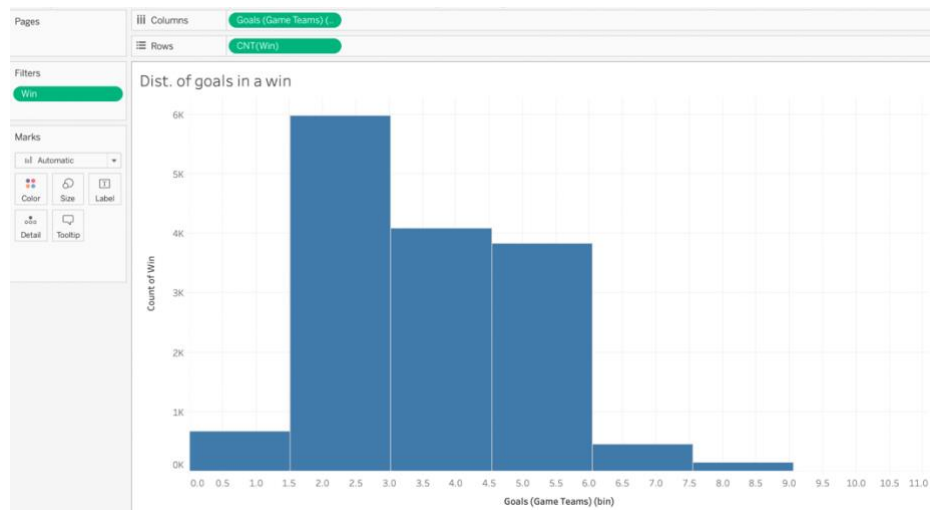
1. This visualization is from one of my select statements. It shows the top 3 scorers in the league for each of the seasons in our database. To create the visualization, I joined the game_skater_stats table, with the game table and the player table. I summed the goals by player for each year, and then created a ranking field and filtered by the ranking. The final result shows who the top scorers were for each season, ordered by ranking.



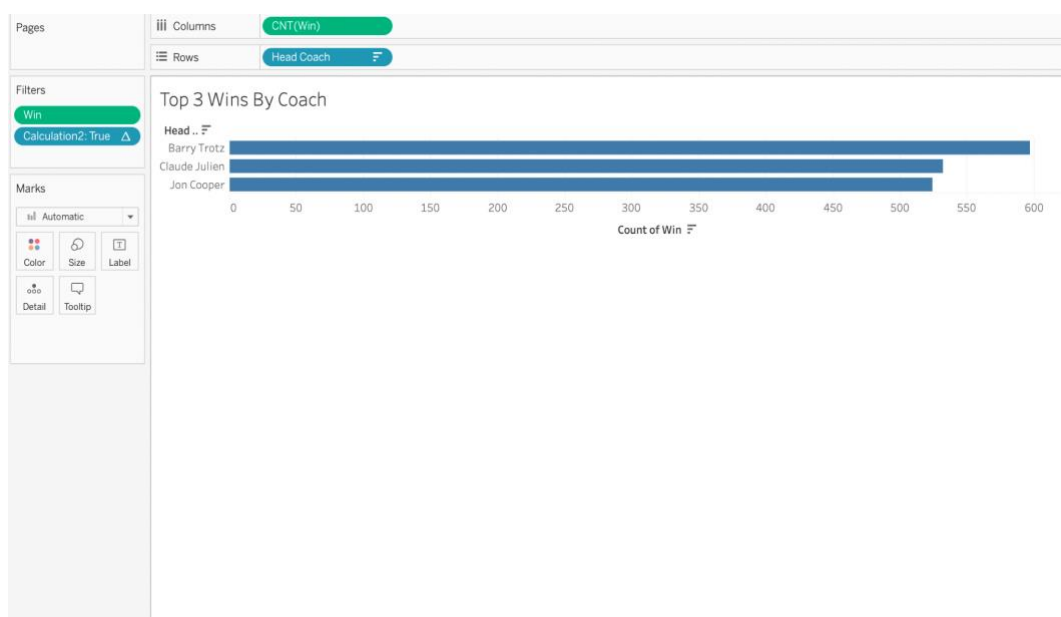
2. My second visualization is also from one of my select statements. It shows the total goals by position in our database and represents them in a bubble chart. I mentioned above this is useful for seeing what position is the most valuable. To create the chart I needed to join the game_skater_stats with the player and the position tables.



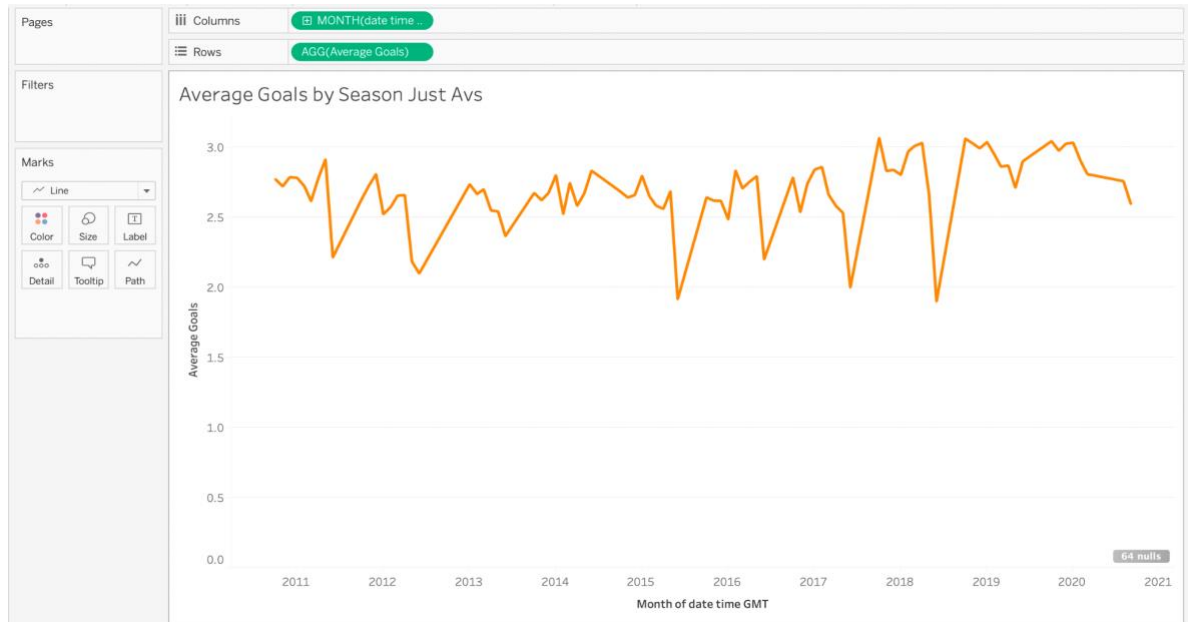
3. This visualization shows how goals are distributed when a team wins. Basically, it's a visualization of how likely our team is to win if we scored x amount of goals. This was created by breaking up the goals scored into bins, and then filtering for a win.



4. This was another one of my select statement queries. The result shows the top 3 total wins by coach. The resulting visualization was created by utilizing the game_teams table, and a count of the win column separated by the coach. Then I created an index for the ranking, which required the coach to be in the top three, that was set to true and filtered for on the left side.

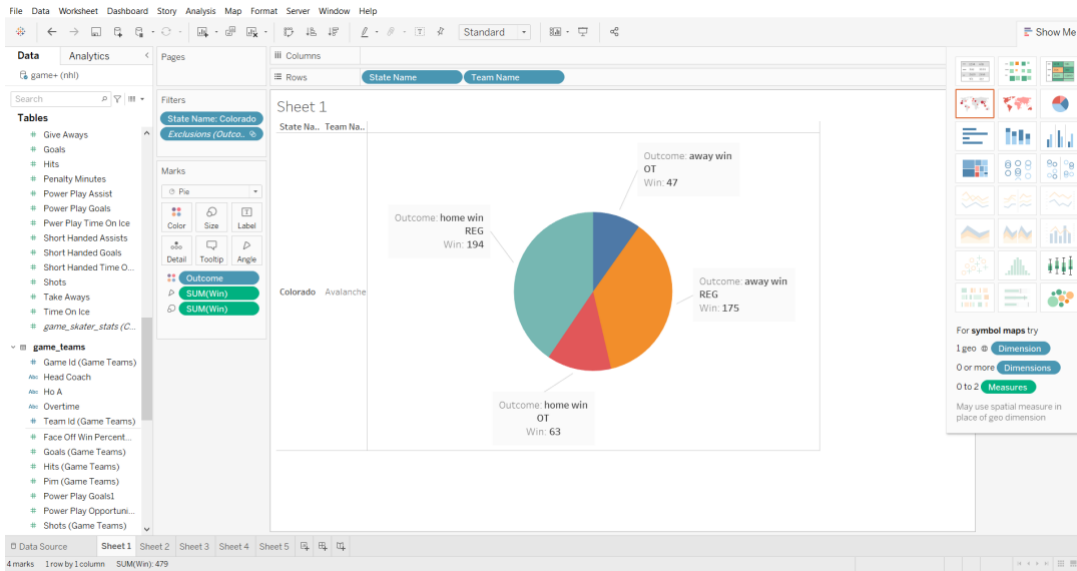


5. My final visualization shows the average goals per month per year for just the Avalanche. This was created by joining the game_teams table, with the game table and the team table. This gives us a sense of how our team has been performing over the last few seasons.

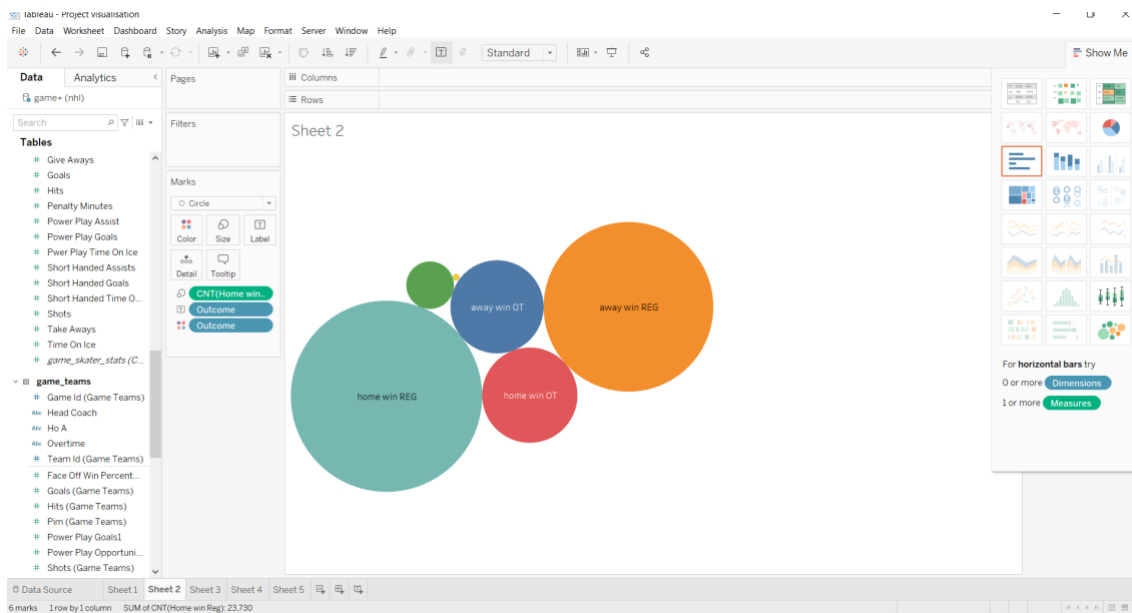


– Santhosh Velayudhan Nair

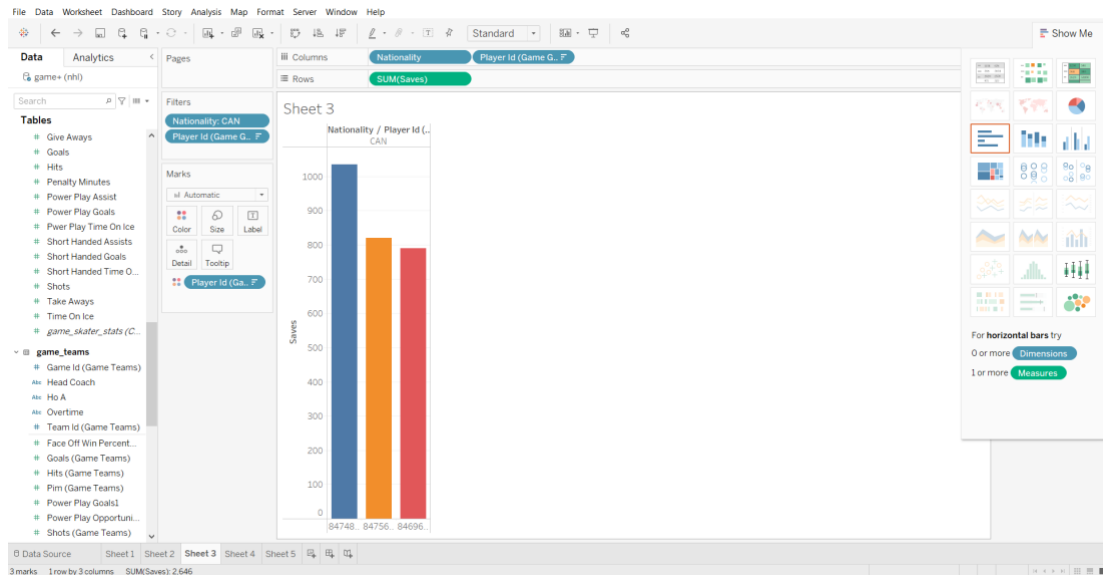
1. This visualization shows how the outcome is distributed for the Colorado team. This uses two tables 'Team' and 'state'. This shows the team's ability to win away compared to the home wins. **Pie chart** is used with proper labelling.



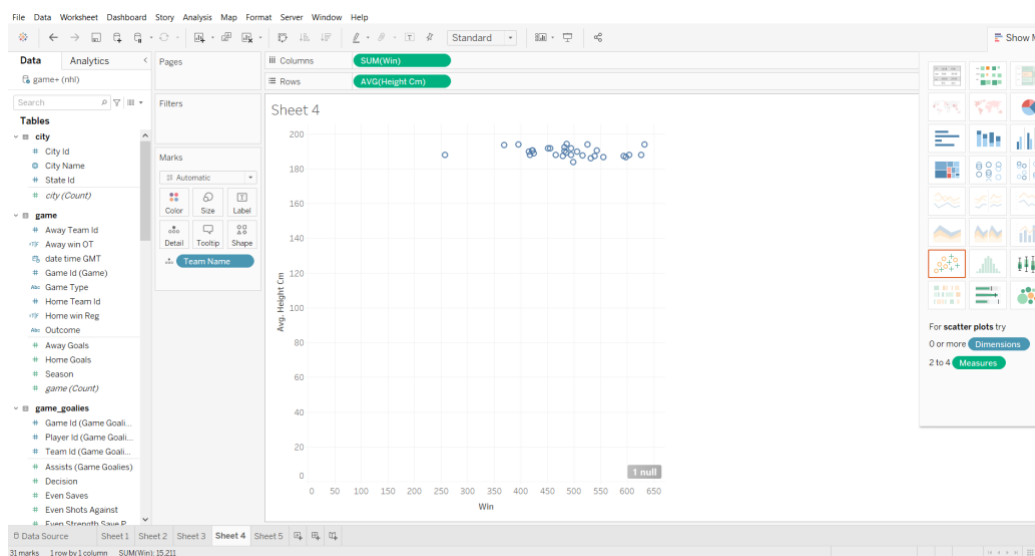
2. This visualization explains how the win pattern is changed for all teams. This represents the different categories through a **bubble chart**. Game table is used. This is a good analysis of how teams perform at home and away. Same question is done through SQL code 2



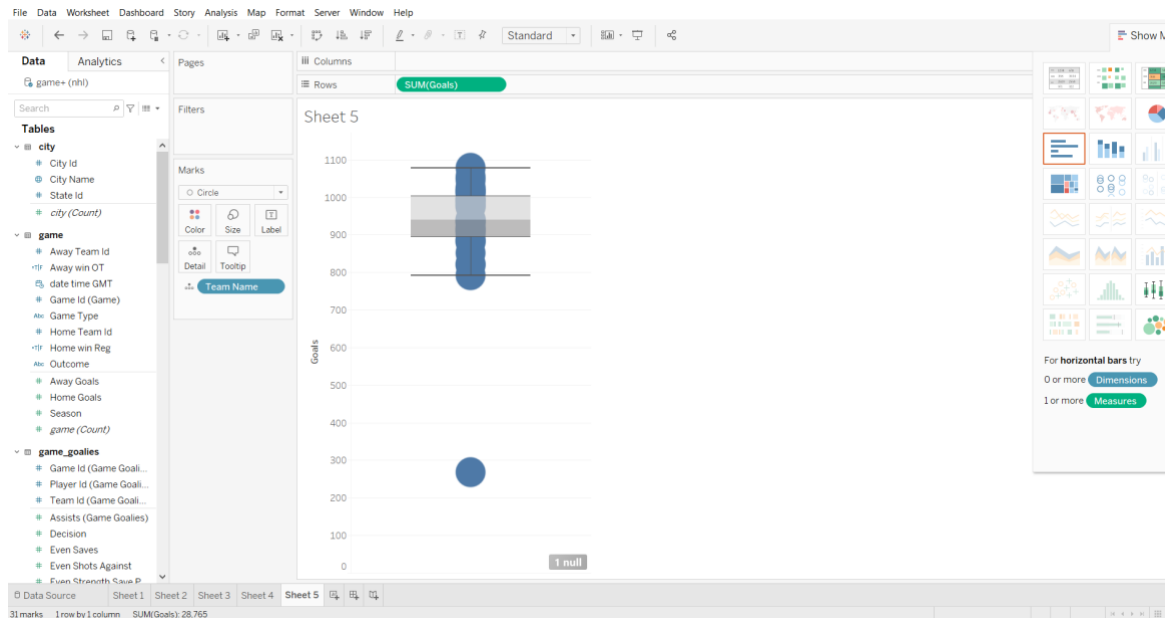
3. This shows who are the top three players of the Canadian teams who made maximum saves. Filter is used in nationality column. Bar chart is used to compare the performance. Player and game tables are used. This shows which goalies performed best among Canadian teams. **Bar** chart is used .Same in SQL code 3



4. This visualization looks for any relationship between the average height of the players of a team and its capacity to win. It is evident that winning is not related to height. **Scattered plot** is used. Game and player tables are used. Height is not a factor influencing winning ability.



5. This shows how the total goal scored is distributed using a boxplot. The median, other IQRs and outliers are displayed. Team and game tables are used. **Box-plot is used.** This shows the distribution of goals for different teams. Golden Knights is an outlier with much fewer goals scored.



5. Stored Functions, Store Procedures, Triggers, and Indices

```
# 1. Stored Functions and applications
# 1-1. Stored Functions
Delimiter $$
CREATE FUNCTION goal_shot_ratios (
shots INT,
goals INT
)
RETURNS DEC(5,2)
DETERMINISTIC
BEGIN
DECLARE ratios DEC(5,2);
SET ratios=goals/shots;
RETURN ratios;
END $$

delimiter ;

DELIMITER &&
create function get_winner(
    gameid integer
)
returns integer
deterministic
begin
    declare winner integer;
    select (case
        when outcome like 'home win%' then home_team_id
        when outcome like 'away win%' then away_team_id
        else null
    end) as winTeam
    into winner
    from game
    where game_id = gameid;
    return winner;
end &&

delimiter ;

DELIMITER $$

CREATE FUNCTION full_name_function(
first VARCHAR(50),
last VARCHAR(50)
)
RETURNS VARCHAR(100)
deterministic
BEGIN ### FUNCTION BODY
    DECLARE full_name VARCHAR(100);
    SET full_name = CONCAT(first, ' ', last);
    RETURN (full_name);
END $$

DELIMITER ;

# 1-2. Applications
#calculates goal to shot ratio in each game.
SELECT game_id,player_id, goal_shot_ratios(shots,goals) AS 'ratios'
```

```

FROM game_skater_stats
group by game_id
order by ratios DESC;

#Average penalty_minutes per win/loss
# winners 8.7575
select avg(penMins)
from (select sum(penalty_minutes) as penMins
from game as g
join game_skater_stats using(game_id)
where (get_winner(game_id)) = team_id
group by game_id) as winnerTable;

#Losers 9.099 mins
select avg(penMins)
from (select sum(penalty_minutes) as penMins
from game as g
join game_skater_stats using(game_id)
where (get_winner(game_id)) != team_id
group by game_id) as loserTable;

# 2. Store Procedure and application
# 2-1. Stored Procedure
DELIMITER //

CREATE PROCEDURE GetPlayersUSA()
BEGIN
    SELECT * FROM player
        WHERE nationality = 'USA';
END //

DELIMITER ;

# 2-2. Application

CALL GetPlayersUSA();

# 3. Triggers
# 3-1. Triggers

DELIMITER //

CREATE TRIGGER insert_firstname
BEFORE INSERT
ON player
FOR EACH ROW
BEGIN
SET NEW.firstName = CONCAT(UPPER(SUBSTRING(NEW.firstname,1,1)),
                            LOWER(SUBSTRING(NEW.firstname
FROM
2))));

END//

DELIMITER ;

DELIMITER $$

CREATE TRIGGER insert_lastname

```

```

BEFORE INSERT
ON player
FOR EACH ROW
BEGIN
SET NEW.lastName = CONCAT(UPPER(SUBSTRING(NEW.lastname,1,1)),
                           LOWER(SUBSTRING(NEW.lastname FROM 2)));

END$$

DELIMITER ;

select * from player;
# 3-2. Application

INSERT INTO player (player_id, firstName, lastName, nationality,
primary_position_ID, birthDate, height_cm)
VALUES ('9999999', 'FAIRY', 'gandhi', 'USA', '1', '2000-01-01', '170');
SELECT * from player;

# 4. INDEX

CREATE INDEX player_firstName_index
ON player (firstName);

CREATE INDEX player_lastName_index
ON Player (lastName);

CREATE INDEX game_id_index
ON game (game_id);

```

6. References

1. Kaggle NHL Game Data (<https://www.kaggle.com/martinellis/nhl-game-data>)
2. NHL Official Homepage (<https://www.nhl.com>)