# MIE1628: BIG DATA SCIENCE
# ASSIGNMENT 1: HDFS, YARN and HIVE SQL

**Student Name: Yuchao Wu**
**Student Number: 1000651984**

**Q1. What are the components of HDFS and YARN? Give one-line explanations (15 marks)**

HDFS components:

- NameNode is the master node for processing metadata information for data blocks.
- DataNode stores the data for processing and use by the NameNode.
- Secondary NameNode creates checkpoint of metadata in case of Name node failures.

YARN components:

- ResourceManager acts as a scheduler to allocate respective NodeManagers accordingly depending on processing requests.
- NodeManager executes tasks on each single Data Node.

**Q2. How many files all together will be saved in HDFS when we want to store a files of 1920MB. Let's assume that default partition size is 128Mb. (8 marks)**

A file of size 1920 MB will be divided into 15 blocks (1920MB/128 MB) where the 15 blocks will be of size 128 MB. It is assumed that the default replication factor is used, thus each block will be replicated three times. Therefore, there will be in total 45 (15*3) blocks.

**Q3 Write the commands for the following HDFS tasks: - (12 marks)**

- **Which command is used to copy a file from HDFS to local file system?**

```
hdfs dfs -copyToLocal <HDFS file path> <Local system directory path>
```

- **Which command is used to move files within HDFS?**

```
hdfs dfs -mv <before move file path> <after move file path>
```

- **Which command is used to print the contents of the file?**

```
hdfs dfs -cat <file path>
```

- **Which command is used to move file from local to HDFS?**

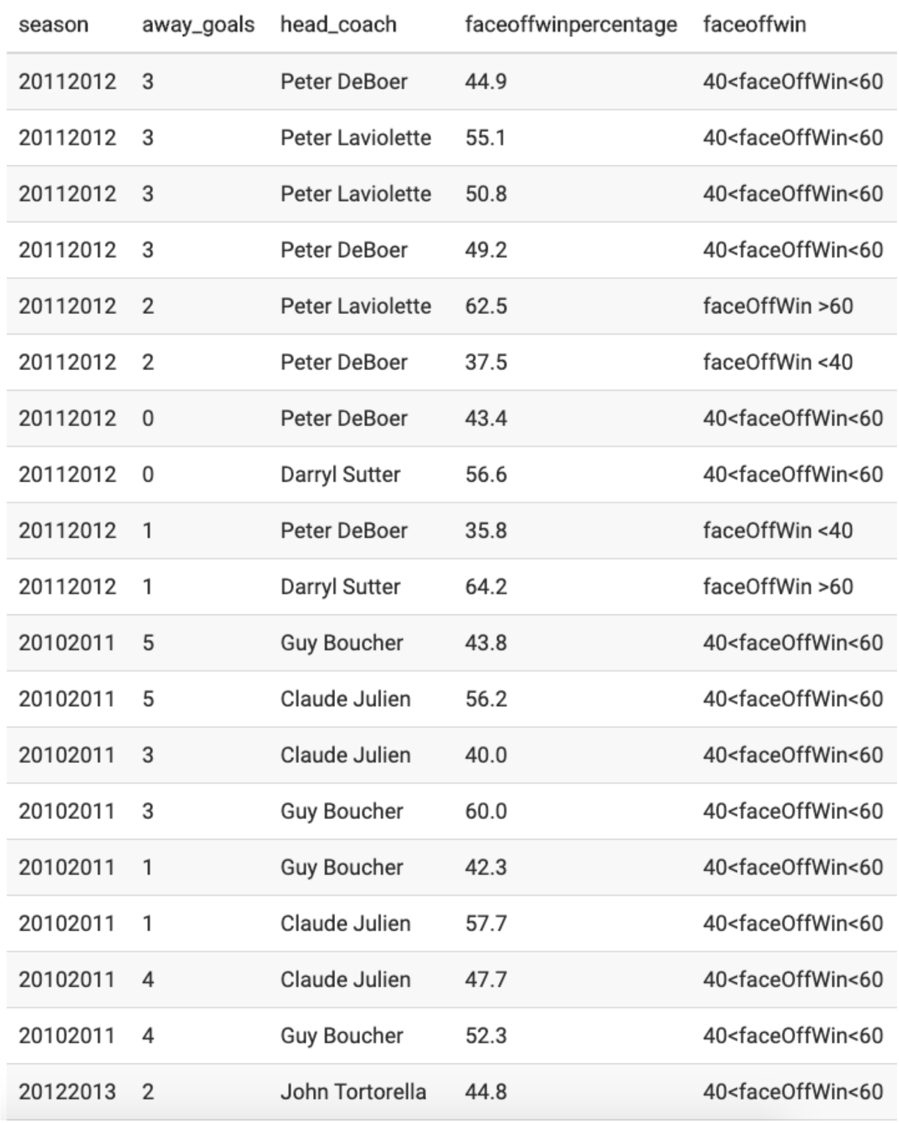
```
hdfs dfs -copyFromLocal <Local system directory path> <HDFS file path>
```

**Q4. Display the table with season, head_coach, faceOffWinPercentage, away_goals and using CASE statement, categorize the faceOffWins by following:**

- **faceOffWinPercentage < 40, ‘faceOffWin < 40’**
- **faceOffWinPercentage > 60, ‘faceOffWins > 60’**
- **elsewhere, ‘40<faceOffWins<60’**

**only where the home_goals > 2**

```
SELECT a.season AS season, a.away_goals AS away_goals,
b.head_coach AS head_coach, b.faceOffWinPercentage AS faceOffWinPercentage,
        CASE
                WHEN b.faceOffWinPercentage<40 THEN 'faceOffWin <40'
                WHEN b.faceOffWinPercentage>60 THEN 'faceOffWin >60'
                ELSE '40<faceOffWin<60'
        END AS faceOffWin
FROM MIE_1628_Assignment1.game AS a
INNER JOIN
MIE_1628_Assignment1.game_teams_stats AS b
ON a.game_id=b.game_id
where a.home_goals>2;
```

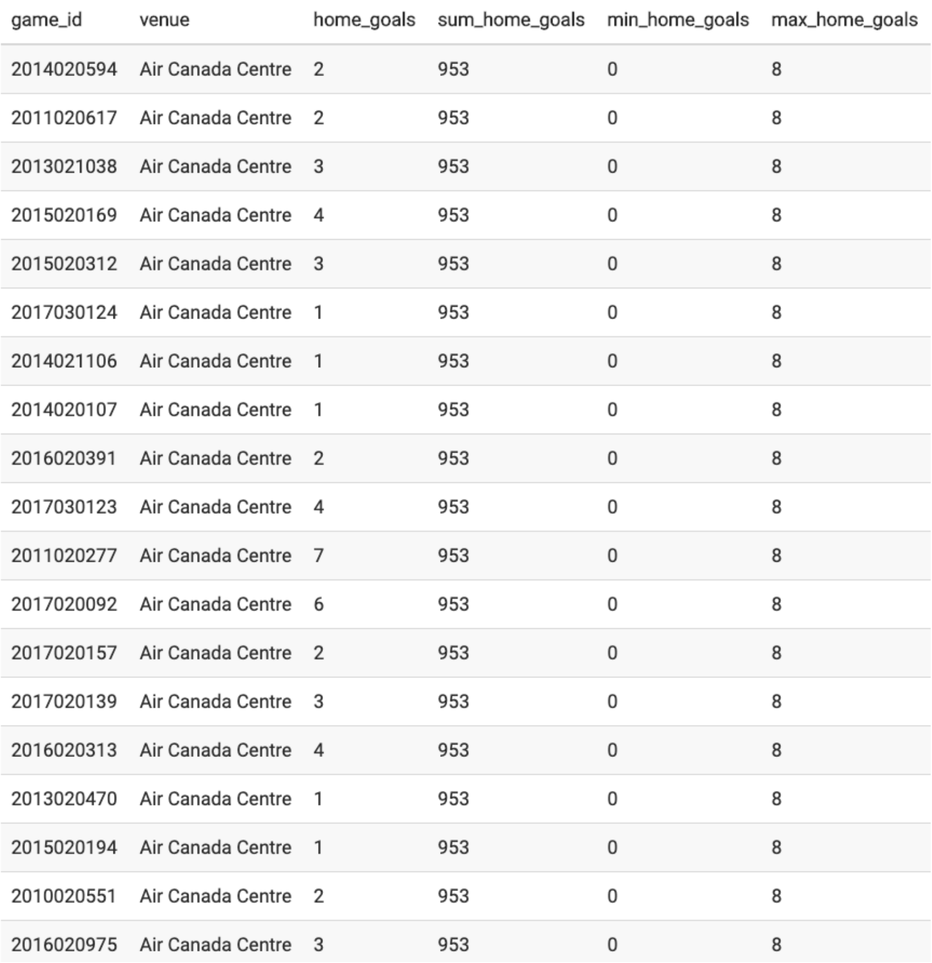


| season | away_goals | head_coach | faceoffwinpercentage | faceoffwin |
|---|---|---|---|---|
| 20112012 | 3 | Peter DeBoer | 44.9 | 40<faceOffWin<60 |
| 20112012 | 3 | Peter Laviolette | 55.1 | 40<faceOffWin<60 |
| 20112012 | 3 | Peter Laviolette | 50.8 | 40<faceOffWin<60 |
| 20112012 | 3 | Peter DeBoer | 49.2 | 40<faceOffWin<60 |
| 20112012 | 2 | Peter Laviolette | 62.5 | faceOffWin >60 |
| 20112012 | 2 | Peter DeBoer | 37.5 | faceOffWin <40 |
| 20112012 | 0 | Peter DeBoer | 43.4 | 40<faceOffWin<60 |
| 20112012 | 0 | Darryl Sutter | 56.6 | 40<faceOffWin<60 |
| 20112012 | 1 | Peter DeBoer | 35.8 | faceOffWin <40 |
| 20112012 | 1 | Darryl Sutter | 64.2 | faceOffWin >60 |
| 20102011 | 5 | Guy Boucher | 43.8 | 40<faceOffWin<60 |
| 20102011 | 5 | Claude Julien | 56.2 | 40<faceOffWin<60 |
| 20102011 | 3 | Claude Julien | 40.0 | 40<faceOffWin<60 |
| 20102011 | 3 | Guy Boucher | 60.0 | 40<faceOffWin<60 |
| 20102011 | 1 | Guy Boucher | 42.3 | 40<faceOffWin<60 |
| 20102011 | 1 | Claude Julien | 57.7 | 40<faceOffWin<60 |
| 20102011 | 4 | Claude Julien | 47.7 | 40<faceOffWin<60 |
| 20102011 | 4 | Guy Boucher | 52.3 | 40<faceOffWin<60 |
| 20122013 | 2 | John Tortorella | 44.8 | 40<faceOffWin<60 |

**Q5. Display the table showing venue, game_id, home_goals, sum, min and max of home_goals, for specific venues using OVER() function with partition window. Try getting the same results using GROUP BY function. Are the results same? Hint: try removing game_id from the query. What do you see now? Explain**

**Using Over() Clause**

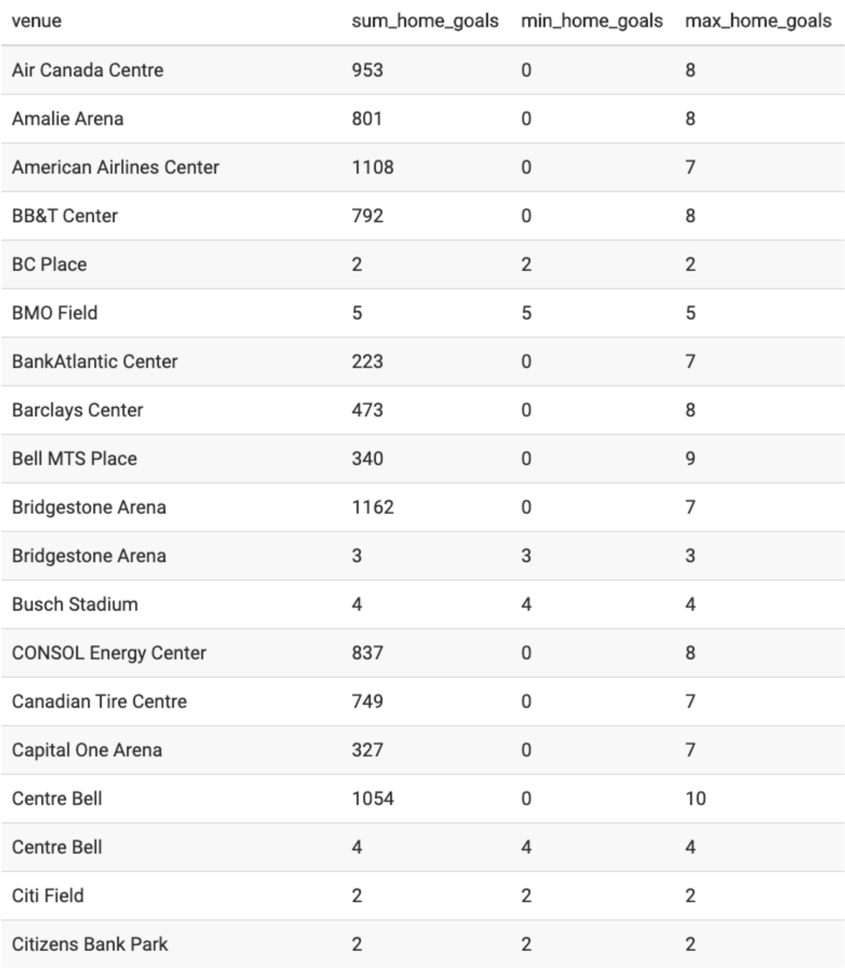
```
SELECT game_id, venue, home_goals,
sum(home_goals) over (PARTITION BY venue) as sum_home_goals,
min(home_goals) over (PARTITION BY venue) as min_home_goals,
max(home_goals) over (PARTITION BY venue) as max_home_goals
FROM MIE_1628_Assignment1.game;
```



| game_id | venue | home_goals | sum_home_goals | min_home_goals | max_home_goals |
|---|---|---|---|---|---|
| 2014020594 | Air Canada Centre | 2 | 953 | 0 | 8 |
| 2011020617 | Air Canada Centre | 2 | 953 | 0 | 8 |
| 2013021038 | Air Canada Centre | 3 | 953 | 0 | 8 |
| 2015020169 | Air Canada Centre | 4 | 953 | 0 | 8 |
| 2015020312 | Air Canada Centre | 3 | 953 | 0 | 8 |
| 2017030124 | Air Canada Centre | 1 | 953 | 0 | 8 |
| 2014021106 | Air Canada Centre | 1 | 953 | 0 | 8 |
| 2014020107 | Air Canada Centre | 1 | 953 | 0 | 8 |
| 2016020391 | Air Canada Centre | 2 | 953 | 0 | 8 |
| 2017030123 | Air Canada Centre | 4 | 953 | 0 | 8 |
| 2011020277 | Air Canada Centre | 7 | 953 | 0 | 8 |
| 2017020092 | Air Canada Centre | 6 | 953 | 0 | 8 |
| 2017020157 | Air Canada Centre | 2 | 953 | 0 | 8 |
| 2017020139 | Air Canada Centre | 3 | 953 | 0 | 8 |
| 2016020313 | Air Canada Centre | 4 | 953 | 0 | 8 |
| 2013020470 | Air Canada Centre | 1 | 953 | 0 | 8 |
| 2015020194 | Air Canada Centre | 1 | 953 | 0 | 8 |
| 2010020551 | Air Canada Centre | 2 | 953 | 0 | 8 |
| 2016020975 | Air Canada Centre | 3 | 953 | 0 | 8 |

**Using Group by Clause:**

```
SELECT venue,
sum(home_goals) as sum_home_goals,
min(home_goals) as min_home_goals,
max(home_goals) as max_home_goals
FROM MIE_1628_Assignment1.game
GROUP BY venue;
```



| venue | sum_home_goals | min_home_goals | max_home_goals |
|---|---|---|---|
| Air Canada Centre | 953 | 0 | 8 |
| Amalie Arena | 801 | 0 | 8 |
| American Airlines Center | 1108 | 0 | 7 |
| BB&T Center | 792 | 0 | 8 |
| BC Place | 2 | 2 | 2 |
| BMO Field | 5 | 5 | 5 |
| BankAtlantic Center | 223 | 0 | 7 |
| Barclays Center | 473 | 0 | 8 |
| Bell MTS Place | 340 | 0 | 9 |
| Bridgestone Arena | 1162 | 0 | 7 |
| Bridgestone Arena | 3 | 3 | 3 |
| Busch Stadium | 4 | 4 | 4 |
| CONSOL Energy Center | 837 | 0 | 8 |
| Canadian Tire Centre | 749 | 0 | 7 |
| Capital One Arena | 327 | 0 | 7 |
| Centre Bell | 1054 | 0 | 10 |
| Centre Bell | 4 | 4 | 4 |
| Citi Field | 2 | 2 | 2 |
| Citizens Bank Park | 2 | 2 | 2 |

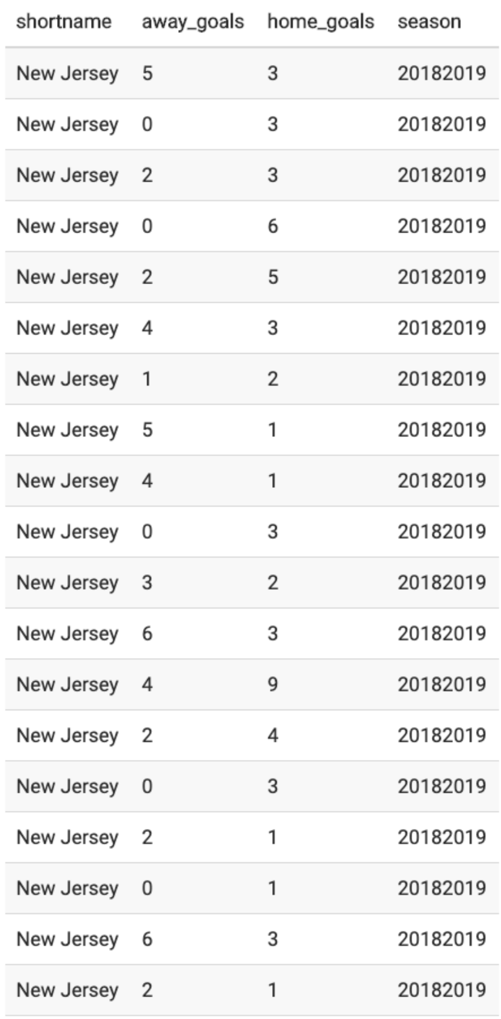
When using OVER() function, the resulting table has the same number of records with the original table. Basically, the function applies to the three added columns which summarizes min, max and sum of home_goals by venue. For game_ids with the same venue, games will have the same venue level min, max and sum of home_goals.

However, when using GROUP BY function, the summarization is done on the original table and the original table gets transformed to a new table. Therefore, game_id has to be left out of the query because each venue can be pointed to multiple game_ids, causing the function to output different results with using OVER() function. Home_goals has to be left out the query too, because without any transformation, it cannot be aggregated by venue. As a result, the number of records is reduced to be the number of different venues.

**Q6. Create a table that has four columns: home team's short name, away goals, home goals and season only for team id = 1. Order records by season starting with most recent season.**

```
SELECT a.shortName AS shortName, b.away_goals AS away_goals,
b.home_goals AS home_goals, b.season AS season
```

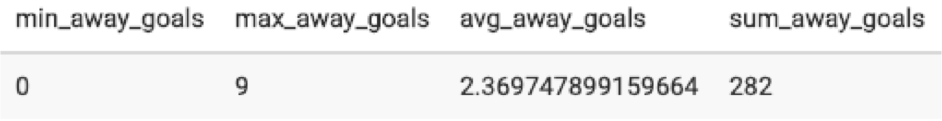
```
FROM MIE_1628_Assignment1.team_info AS a
INNER JOIN MIE_1628_Assignment1.game_teams_stats as c
ON a.team_id = c.team_id
INNER JOIN MIE_1628_Assignment1.game AS b
ON c.game_id = b.game_id
WHERE a.team_id =1
ORDER BY season DESC;
```



| shortname | away_goals | home_goals | season |
|---|---|---|---|
| New Jersey | 5 | 3 | 20182019 |
| New Jersey | 0 | 3 | 20182019 |
| New Jersey | 2 | 3 | 20182019 |
| New Jersey | 0 | 6 | 20182019 |
| New Jersey | 2 | 5 | 20182019 |
| New Jersey | 4 | 3 | 20182019 |
| New Jersey | 1 | 2 | 20182019 |
| New Jersey | 5 | 1 | 20182019 |
| New Jersey | 4 | 1 | 20182019 |
| New Jersey | 0 | 3 | 20182019 |
| New Jersey | 3 | 2 | 20182019 |
| New Jersey | 6 | 3 | 20182019 |
| New Jersey | 4 | 9 | 20182019 |
| New Jersey | 2 | 4 | 20182019 |
| New Jersey | 0 | 3 | 20182019 |
| New Jersey | 2 | 1 | 20182019 |
| New Jersey | 0 | 1 | 20182019 |
| New Jersey | 6 | 3 | 20182019 |
| New Jersey | 2 | 1 | 20182019 |

**Q7. Calculate the minimum, maximum, average and sum of all the goals played by teams away from home in TD Garden having faceOffWinPercentage > 50**

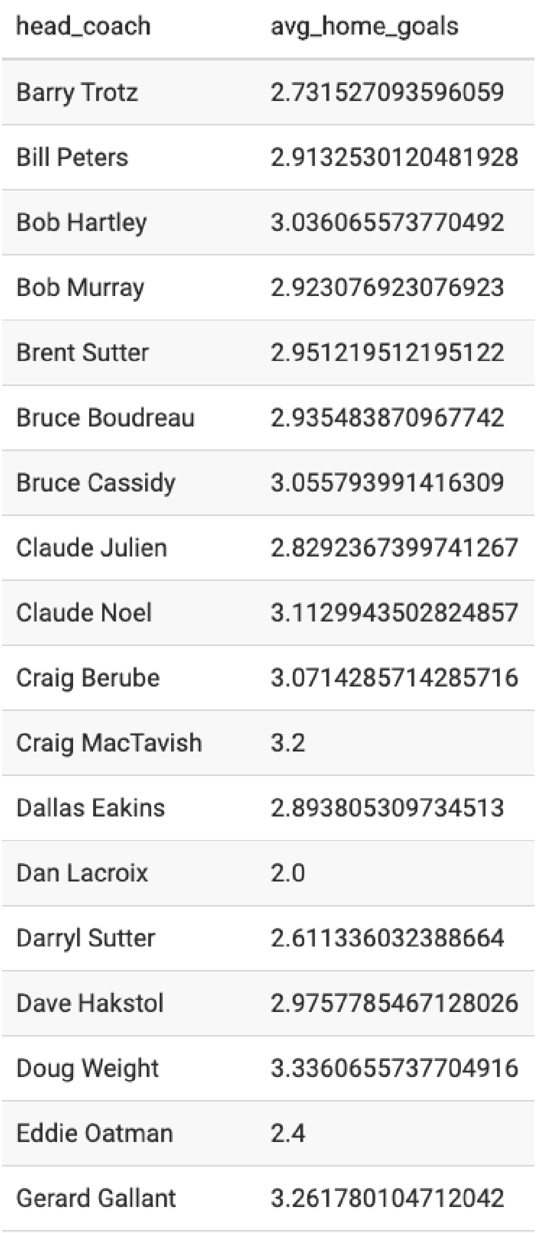
```
SELECT
min(away_goals) AS min_away_goals,
max(away_goals) AS max_away_goals,
avg(away_goals) AS avg_away_goals,
sum(away_goals) AS sum_away_goals
FROM (SELECT
        a.away_team_id AS away_team_id,
        a.away_goals AS away_goals,
        a.venue AS venue FROM MIE_1628_Assignment1.game AS a
        INNER JOIN MIE_1628_Assignment1.game_teams_stats AS b
        ON a.game_id = b.game_id
        WHERE b.faceOffWinPercentage>50
        AND a.venue = 'TD Garden'
        AND b.HoA = 'away') AS c;
```



| min_away_goals | max_away_goals | avg_away_goals | sum_away_goals |
|---|---|---|---|
| 0 | 9 | 2.369747899159664 | 282 |

**Q8. Show the average home goals per coaches for all the even team id numbers. (15 marks)**

```
SELECT head_coach, avg(home_goals) AS avg_home_goals
FROM
  (SELECT
   a.home_goals AS home_goals,
   b.head_coach AS head_coach,
   b.team_id AS team_id
  FROM MIE_1628_Assignment1.game AS a
  INNER JOIN MIE_1628_Assignment1.game_teams_stats AS b
  ON a.game_id = b.game_id) AS c
WHERE team_id % 2 = 0
GROUP BY head_coach;
```



| head_coach | avg_home_goals |
|---|---|
| Barry Trotz | 2.731527093596059 |
| Bill Peters | 2.9132530120481928 |
| Bob Hartley | 3.036065573770492 |
| Bob Murray | 2.923076923076923 |
| Brent Sutter | 2.951219512195122 |
| Bruce Boudreau | 2.935483870967742 |
| Bruce Cassidy | 3.055793991416309 |
| Claude Julien | 2.8292367399741267 |
| Claude Noel | 3.1129943502824857 |
| Craig Berube | 3.0714285714285716 |
| Craig MacTavish | 3.2 |
| Dallas Eakins | 2.893805309734513 |
| Dan Lacroix | 2.0 |
| Darryl Sutter | 2.611336032388664 |
| Dave Hakstol | 2.9757785467128026 |
| Doug Weight | 3.3360655737704916 |
| Eddie Oatman | 2.4 |
| Gerard Gallant | 3.261780104712042 |