Name- Sarvesh Kumar Sharma **Subject – Big Data Analytics Lab**

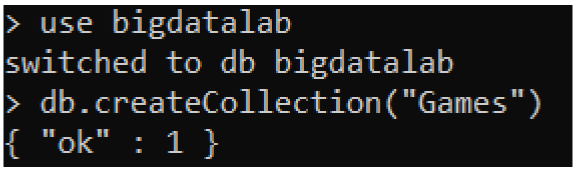
Section – H **Subject Code- BCSE0158**

Roll No- 29 **University Roll No- 181500625**

Q1. Answer the following using MongoDB:

a. Create a collection called 'games'.

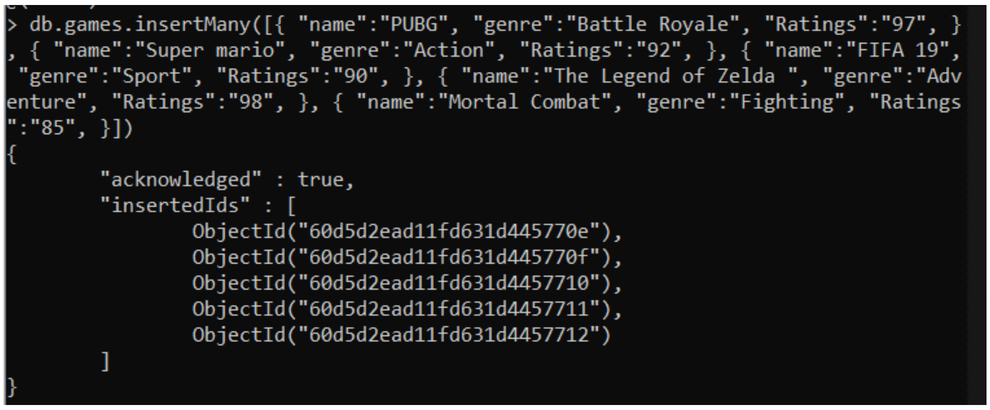
db.createCollection("Games")



```
> use bigdatalab
switched to db bigdatalab
> db.createCollection("Games")
{ "ok" : 1 }
```

b. Add 5 games to the database. Give each document the following properties: name, genre, rating (out of 100)

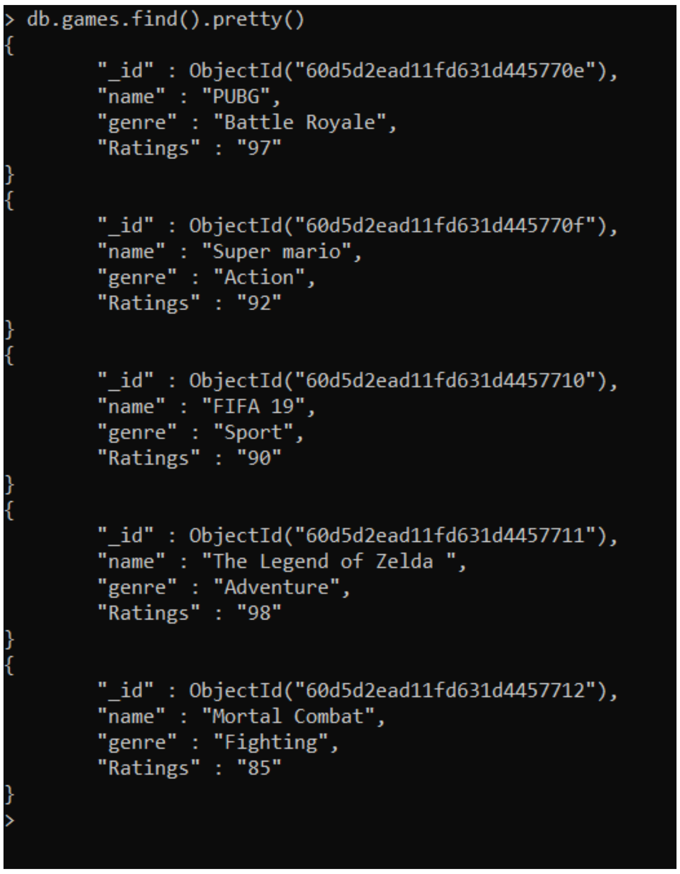
```
db.games.insertMany([

{ "name":"PUBG", "genre":"Battle Royale", "Ratings":"97", },

{ "name":"Super mario", "genre":"Action", "Ratings":"92", },

{ "name":"FIFA 19", "genre":"Sport", "Ratings":"90", },

{ "name":"The Legend of Zelda ", "genre":"Adventure", "Ratings":"98", },

{ "name":"Mortal Combat", "genre":"Fighting", "Ratings":"85", }]

)
```



```
> db.games.insertMany([{ "name":"PUBG", "genre":"Battle Royale", "Ratings":"97", }
, { "name":"Super mario", "genre":"Action", "Ratings":"92", }, { "name":"FIFA 19",
 "genre":"Sport", "Ratings":"90", }, { "name":"The Legend of Zelda ", "genre":"Adv
enture", "Ratings":"98", }, { "name":"Mortal Combat", "genre":"Fighting", "Ratings
":"85", }])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("60d5d2ead11fd631d445770e"),
                ObjectId("60d5d2ead11fd631d445770f"),
                ObjectId("60d5d2ead11fd631d4457710"),
                ObjectId("60d5d2ead11fd631d4457711"),
                ObjectId("60d5d2ead11fd631d4457712")
        ]
}
```

c. Write a query that returns all the games

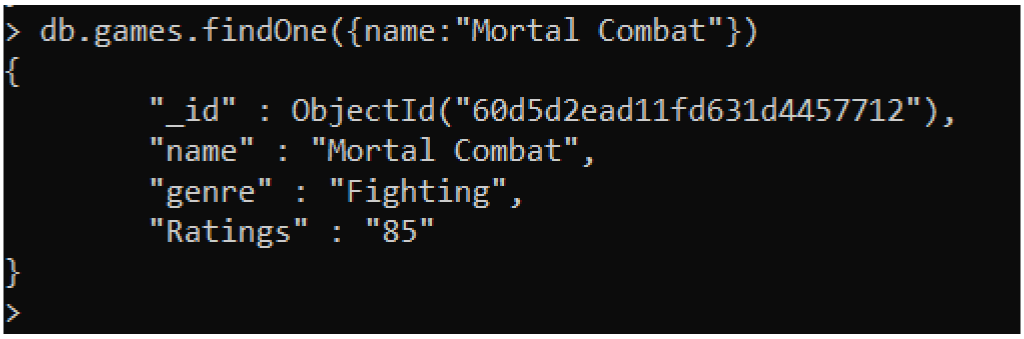
```
> db.games.find().pretty()
```



```
> db.games.find().pretty()
{
        "_id" : ObjectId("60d5d2ead11fd631d445770e"),
        "name" : "PUBG",
        "genre" : "Battle Royale",
        "Ratings" : "97"
}
{
        "_id" : ObjectId("60d5d2ead11fd631d445770f"),
        "name" : "Super mario",
        "genre" : "Action",
        "Ratings" : "92"
}
{
        "_id" : ObjectId("60d5d2ead11fd631d4457710"),
        "name" : "FIFA 19",
        "genre" : "Sport",
        "Ratings" : "90"
}
{
        "_id" : ObjectId("60d5d2ead11fd631d4457711"),
        "name" : "The Legend of Zelda ",
        "genre" : "Adventure",
        "Ratings" : "98"
}
{
        "_id" : ObjectId("60d5d2ead11fd631d4457712"),
        "name" : "Mortal Combat",
        "genre" : "Fighting",
        "Ratings" : "85"
}
>
```

d. Write a query to find one of your games by name without using limit()

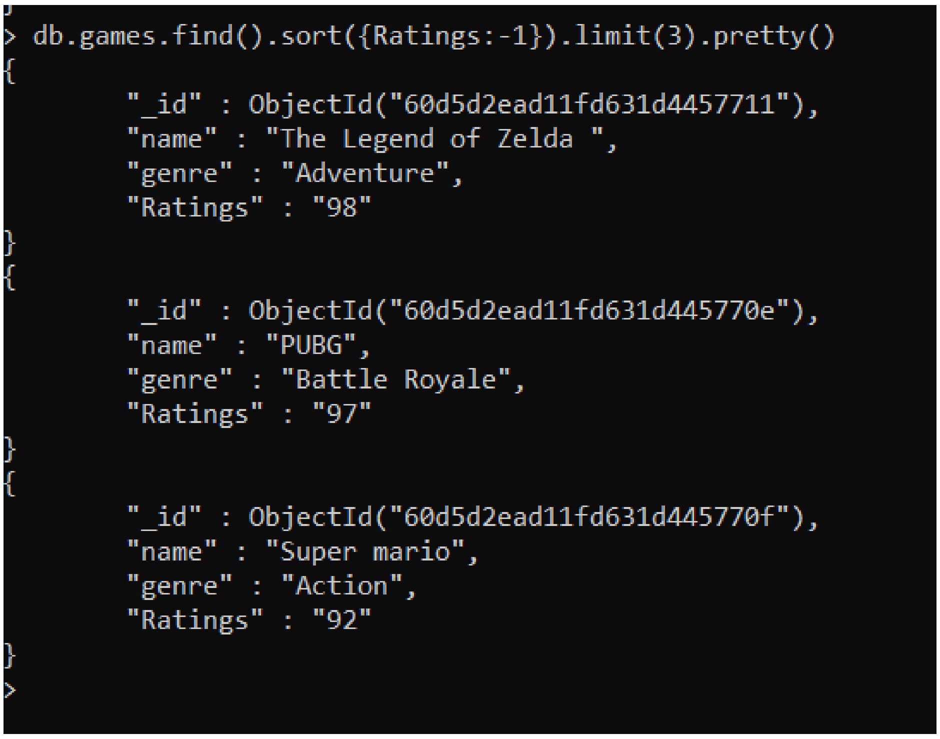
db.games.findOne({name:"Mortal Combat"})



```
> db.games.findOne({name:"Mortal Combat"})
{
        "_id" : ObjectId("60d5d2ead11fd631d4457712"),
        "name" : "Mortal Combat",
        "genre" : "Fighting",
        "Ratings" : "85"
}
>
```

e. Write a query that returns the 3 highest rated games.

> db.games.find().sort({Ratings:-1}).limit(3).pretty()



```
> db.games.find().sort({Ratings:-1}).limit(3).pretty()
{
        "_id" : ObjectId("60d5d2ead11fd631d4457711"),
        "name" : "The Legend of Zelda ",
        "genre" : "Adventure",
        "Ratings" : "98"
}
{
        "_id" : ObjectId("60d5d2ead11fd631d445770e"),
        "name" : "PUBG",
        "genre" : "Battle Royale",
        "Ratings" : "97"
}
{
        "_id" : ObjectId("60d5d2ead11fd631d445770f"),
        "name" : "Super mario",
        "genre" : "Action",
        "Ratings" : "92"
}
>
```

Q3. Answer any *two* of the following:

a. **Difference between internal and external table in Hive**

Internal Table	**External Table**
Hive moves table data to warehouse directory	Hive does not move table data to warehouse directory
Dropping deletes table data and metadata	Dropping deletes table's metadata
Support TRUNCATE	No TRUNCATE support
Support ACID transaction	No ACID transaction support
Query Result Caching works	Query Result Caching does not works

b. **Pig Philosophy**

What does it mean to be a pig?

The Apache Pig Project has some founding principles that help pig developers decide how the system should grow over time. those principles are mentioned below.

Pigs Eat Anything

Pig can operate on data whether it has metadata or not. It can operate on data that is relational, nested, or unstructured. And it can easily be extended to operate on data beyond files, including key/value stores, databases, etc.

Pigs Live Anywhere

Pig is intended to be a language for parallel data processing. It is not tied to one particular parallel framework. It has been implemented first on Hadoop, but we do not intend that to be only on Hadoop.

Pigs Are Domestic Animals

Pig is designed to be easily controlled and modified by its users.

Pig allows integration of user code where ever possible, so it currently supports user defined field transformation functions, user defined aggregates, and user defined conditionals. These functions can be written in Java or scripting languages that can compile down to Java (e.g. Jython). Pig supports user provided load and store functions. It supports external executables via its stream command and Map Reduce jars via its mapreduce command. It allows users to provide a custom partitioner for their jobs in some circumstances and to set the level of reduce parallelism for their jobs. command. It allows users to set the level of reduce parallelism for their jobs and in some circumstances to provide a custom partitioner.

Pig has an optimizer that rearranges some operations in Pig Latin scripts to give better performance, combines Map Reduce jobs together, etc. However, users can easily turn this optimizer off to prevent it from making changes that do not make sense in their situation.

Pigs Fly

Pig processes data quickly. We want to consistently improve performance, and not implement features in ways that weigh pig down so it can't fly.

c. What is ObjectID in MongoDB and how replication works in MongoDB

An **ObjectID** is a 12-byte Field of BSON type. The first 4 bytes representing the Unix Timestamp of the document. The next 3 bytes are the machine Id on which the **MongoDB** server is running. The next 2 bytes are of process id. The last Field is 3 bytes used for increment the **objectid**.

With MongoDB, replication is achieved through a replica set. Writer operations are sent to the primary server (node), which applies the operations across secondary servers, replicating the data.

If the primary server fails (through a crash or system failure), one of the secondary servers takes over and becomes the new primary node via election. If that server comes back online, it becomes a secondary once it fully recovers, aiding the new primary node.

d. Define Big Data

Bigdata are high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and processes optimization.

Exploratory data Analysis on Iris Data

Work done by -

- **Name- Sarvesh Kumar Sharma**
- **Section - H**
- **Roll no- 29**
- **University Roll no - 181500625**

In [1]:

```
#loading iris Data

data(iris)
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

In [2]:

```
# Dimension of dataset
dim(iris)
```

150 5

In [3]:

```
# variables names

names(iris)
```

'Sepal.Length' 'Sepal.Width' 'Petal.Length' 'Petal.Width' 'Species'

In [4]:

```
#structure of data

str(iris)
```

```
'data.frame': 150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

In [5]:

```
#Summery of the data

summary(iris)
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
       Species  
 setosa    :50  
 versicolor:50  
 virginica :50  
```

In [6]:

```
# group mean
aggregate(.~Species, iris, mean)
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

In [7]:

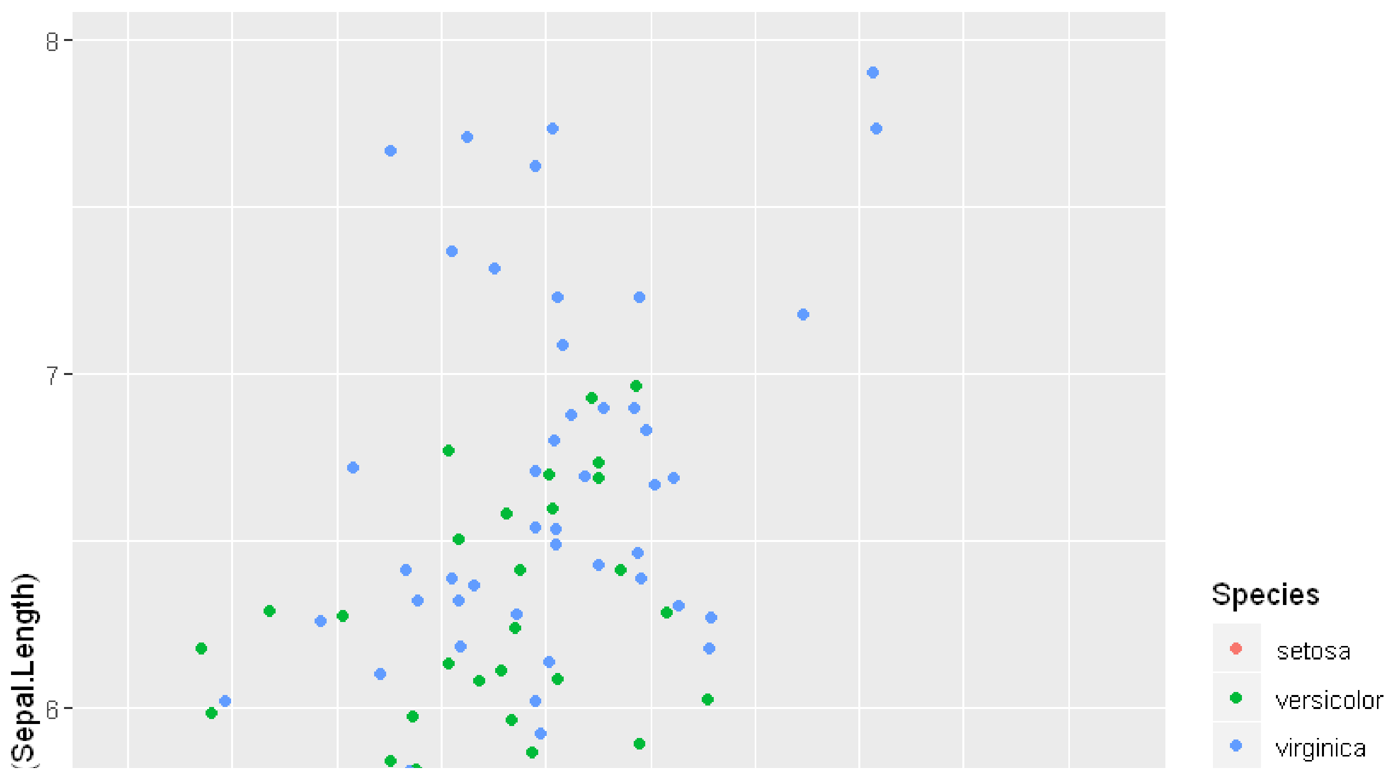
```
# find missing values

sum(is.na(iris))
```

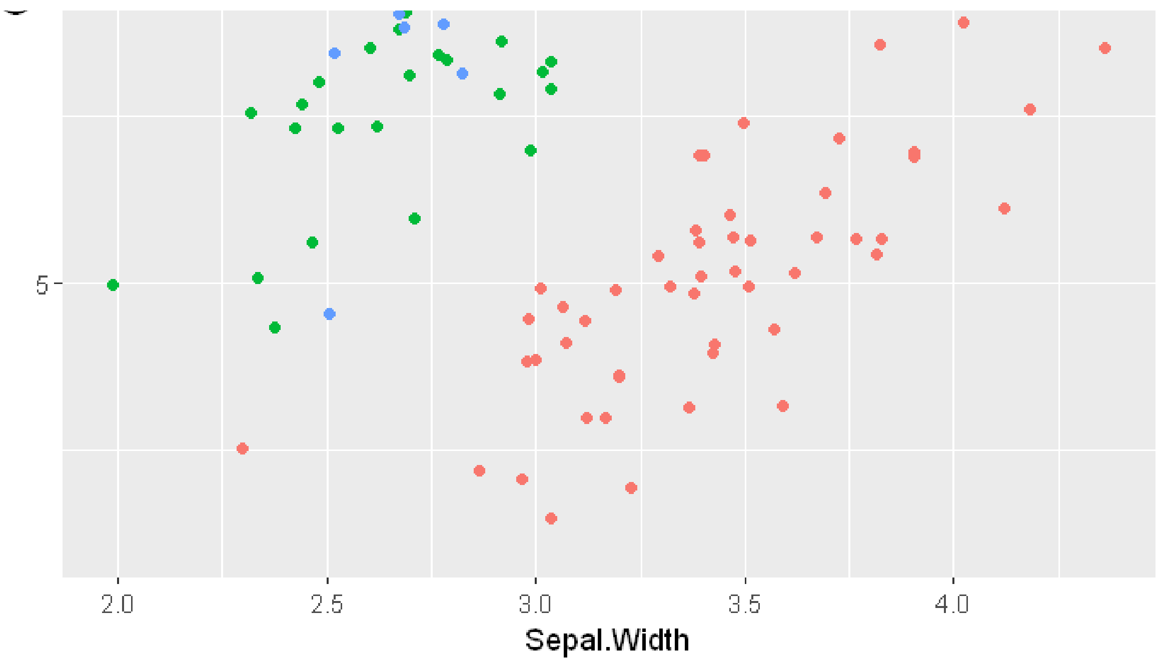
0

In [8]:

```
# graph between sepal length and sepal length

library(ggplot2)
ggplot(iris,aes(Sepal.Width,(Sepal.Length),color=Species))+
  geom_point(position="jitter")
```

```
Registered S3 methods overwritten by 'ggplot2':
  method         from 
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
```





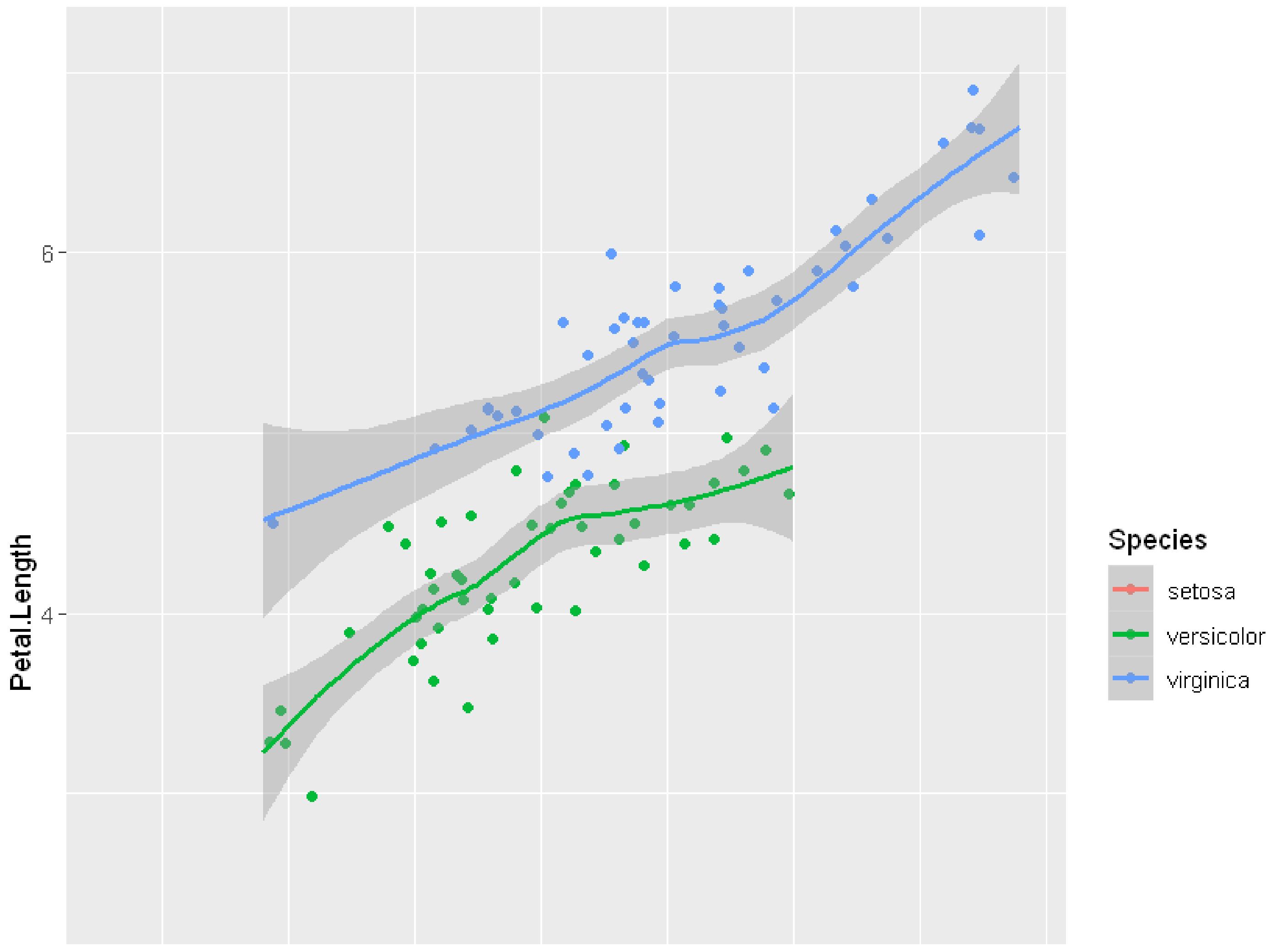
By the two plots I can conclude that out of all the three, virginica must be the biggest flower and setosa must be the smallest as the median petal length is 5.55 and 1.5 for each respectively. Versicolor comes close to virginica with a median petal length of 4.35.

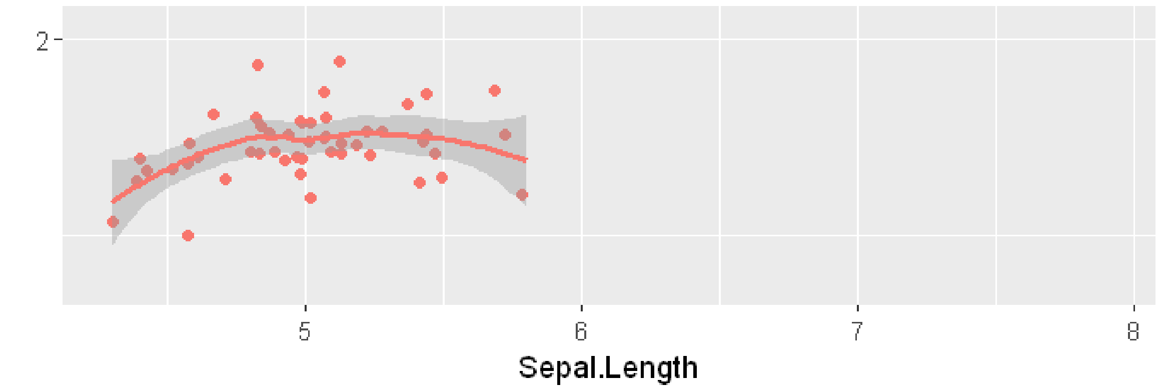
as the virginica has the highest petal length, relatively its sepal length is also big. with a median of 5.8

```
In [9]:
```

```r
#sepal width and petal width for each species
ggplot(iris, aes(Sepal.Length,Petal.Length , color=Species))+
  geom_point(position = "jitter")+
  geom_smooth()
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



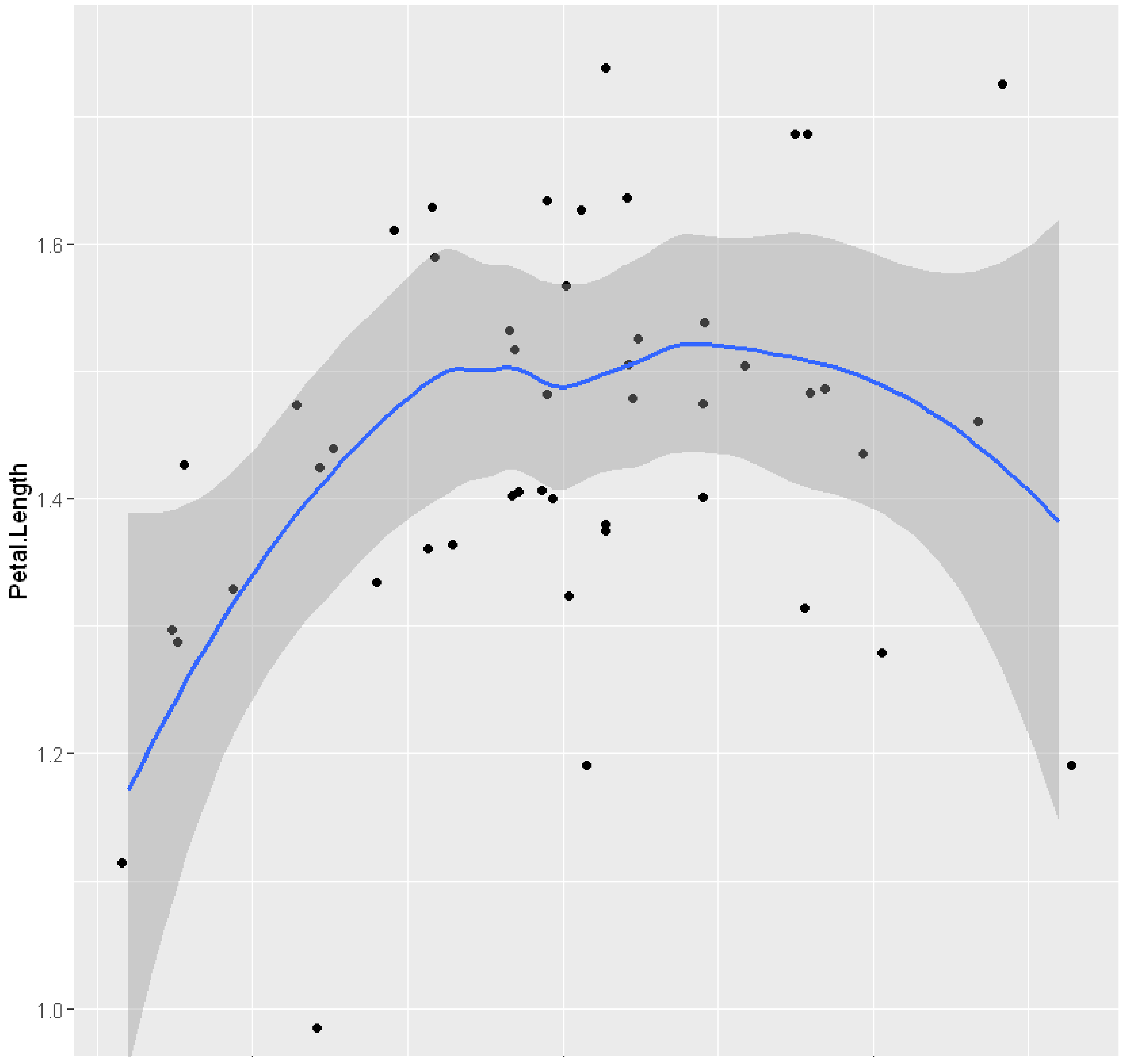


his shows that for the versicolor and virginica, the petal length has kind of a linear relationship with the sepal length. But for setosa, the smoother line is a bit curved, which means that it should be straight but appears curved due to outliers. We can inspect the setosa species seperately to know more about its distribution.

In [10]:

```
#plot for sepal length and petal length of setosa species
ggplot(subset(iris, iris$Species=="setosa"),aes(Sepal.Length,Petal.Length))+
  geom_point(position="jitter")+
  coord_cartesian(ylim=c(1,1.75))+
  geom_smooth()
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

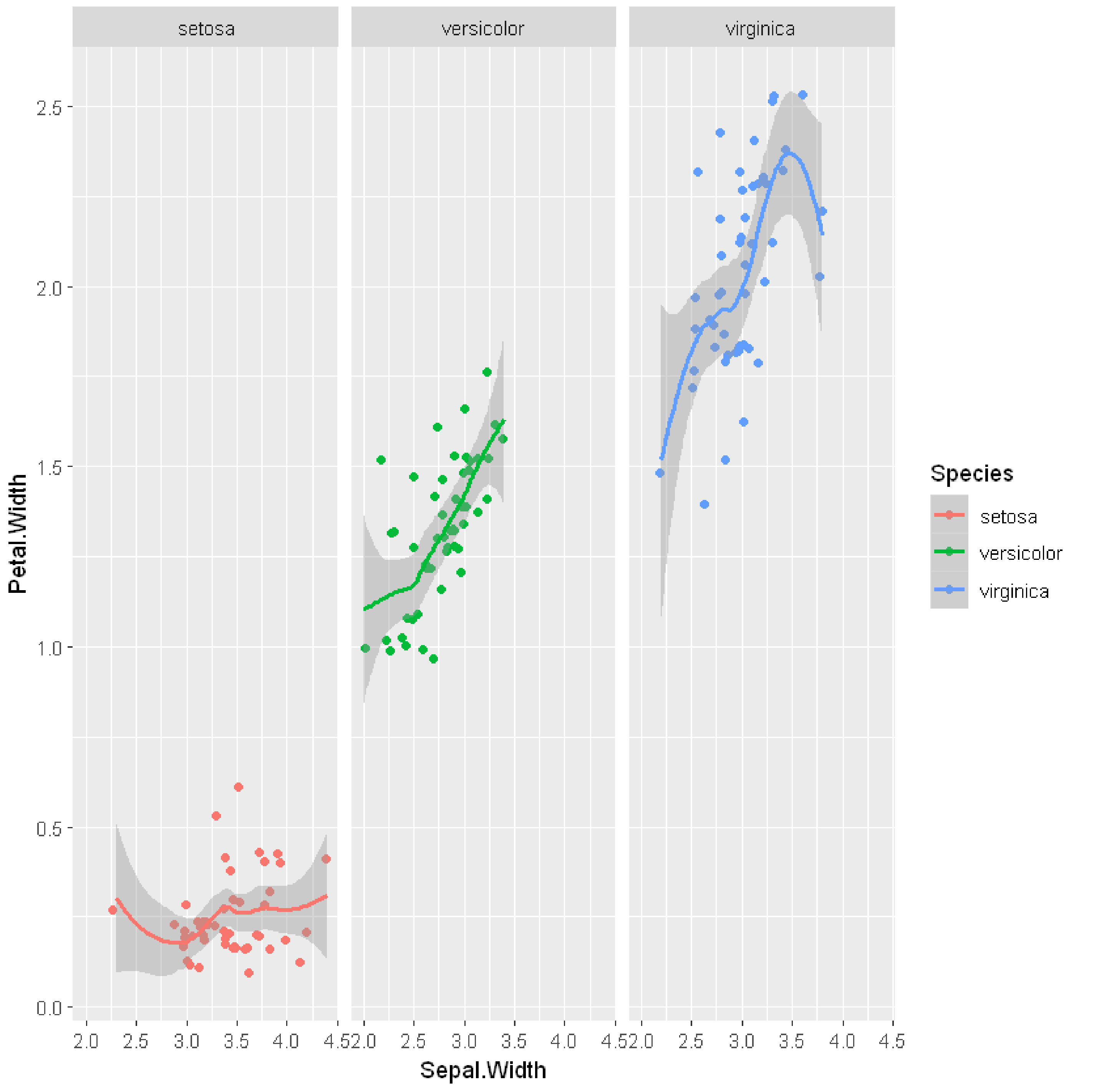


we see that the plot is quite non-linear. There are quite notable outliers in the data for the species setosa. we can see a flower with a sepal length of 4.6 and a petal length of 1 while there is another flower with a petal length of 1.55 and sepal length ofaround 4.55. throughout the plot the data points are spread out, down and up and accross. hence when tried to smooth, the line becomes a bit curved due to these outliers. Another instance of this non linearity is shown at 5.75 where the petal length is around 1.2. to its left there is another point with a value of around 5.7 with a petal length of 1.7.

In [11]:

```
#plot for sepal width and petal width faceted by species
ggplot(iris,aes(Sepal.Width,Petal.Width, color=Species))+
  geom_point(position="jitter")+
  geom_smooth()+
  facet_wrap(~Species)
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Here we see that the Sepal width doesnt seem to have a correlation with petal length in the setosa species with the help of the smoothed line. For other species there is a linear relationship for petal width and sepal width

the help of the smoothed line. For other species there is a linear relationship for petal width and sepal width.

From all the plots above, we can see that although the 3 species of flowers are from the same family, the setosa is linearly seperable, ie a line can be drawn to seperate setosa from the other 2 species on each of the graphs. So we can say that virginica and versicolor are quite identical as there are a few points from both these species that fall a bit close to each other, but the setosa is comparitively different from the other 2 species as all its points lie away from the group of versicolor and virginica. Also in all the graphs there was a linear trend for virginica and versicolor, whereas, setosa showed a non linearity or if adjusted for outliers, we can say that the setosa doesnt show a trend with respect to its petal and sepal dimensions.

```
In [ ]:
```