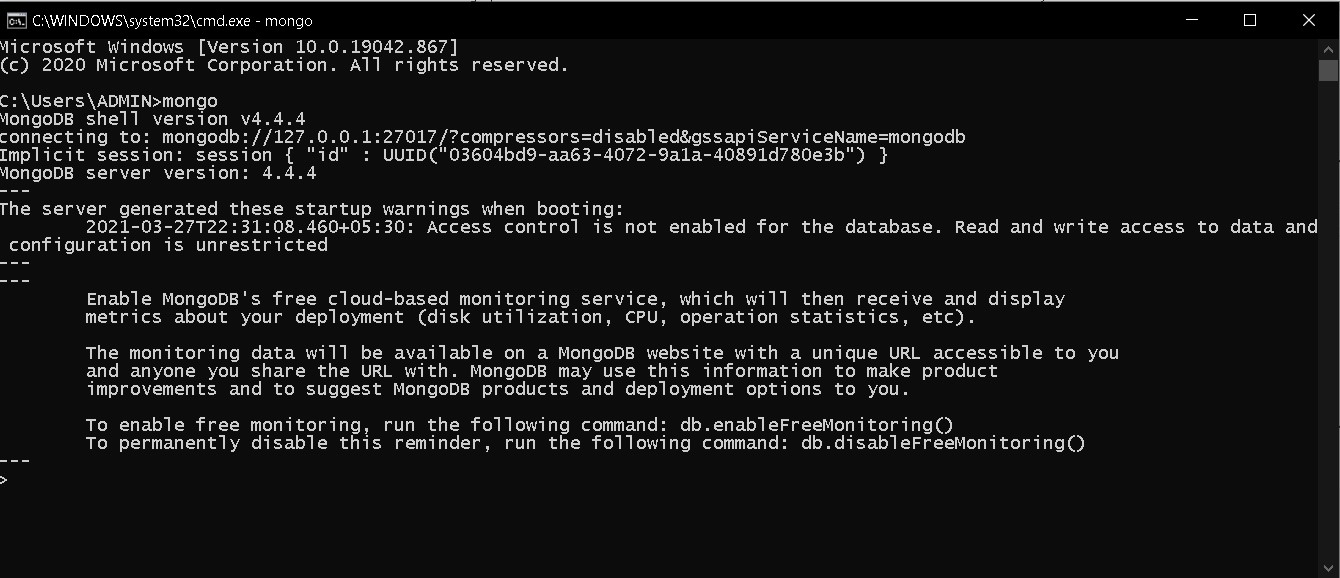
**Practical No. 6**

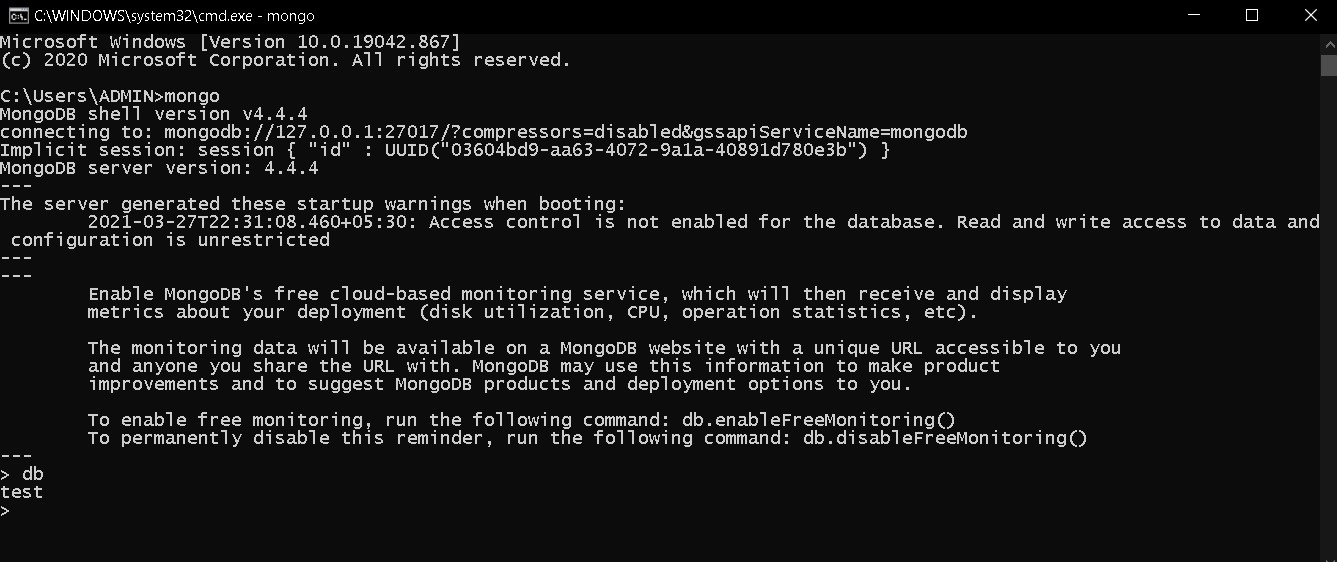
**Aim:-** Practical of Mongo DB.

1. To start mongodb we need to run the following command. This command will start the server.

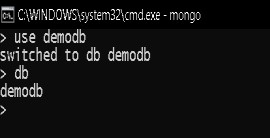
Now run **mongo** command in cmd this will start the mongodb shell.



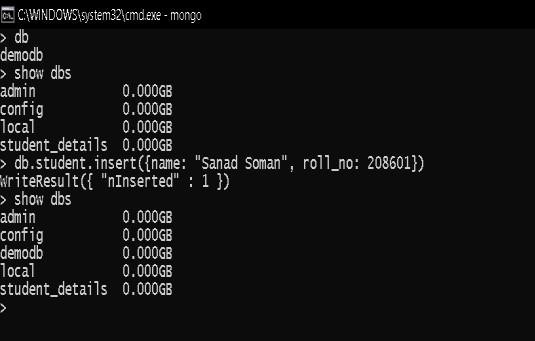
1. To display name of current database we use command “**db**”.



1. To create database or use a particular database we can use command “**use**”.



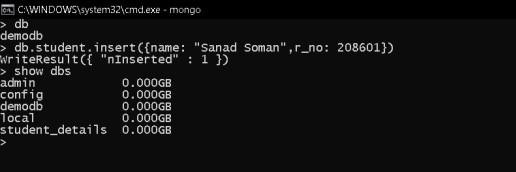
1. Now to see the created database we use command “**show dbs**” but at first it will not show us the created database as soon as we create the collection within database we can see our database by using “**show dbs** “ command.



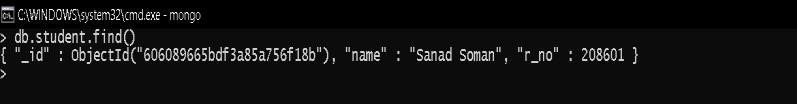
1. To drop database we can use command “**db.dropDatabase()**”.



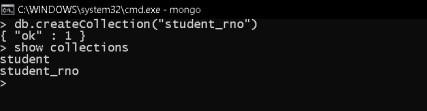
1. To create collection we can use the command “**db.collection\_name.insert({key\_value pairs})**”.



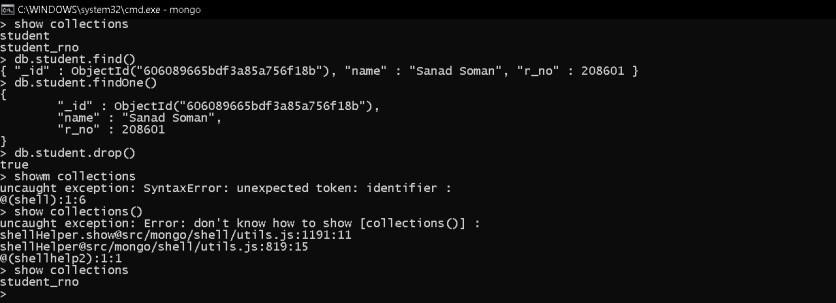
1. To see the records within collection we can use following command:”**db.collection\_name.find()**”.



1. To create collections with options before inserting data use command “**db.createCollection(“collection\_name”)**”.

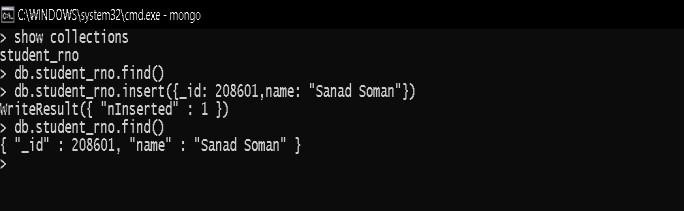


1. To drop a collection we use command : “**db.collection\_name.drop()**”.

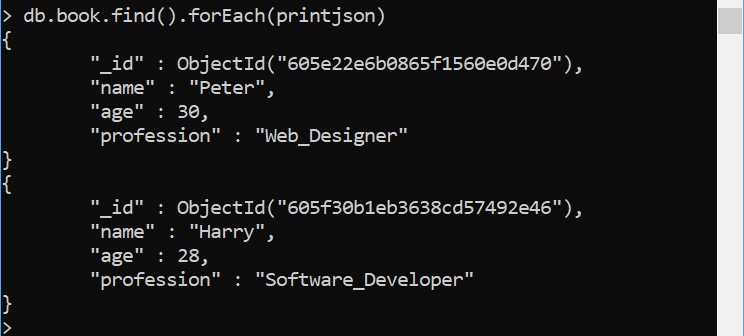


1. To insert values into collection we use command:

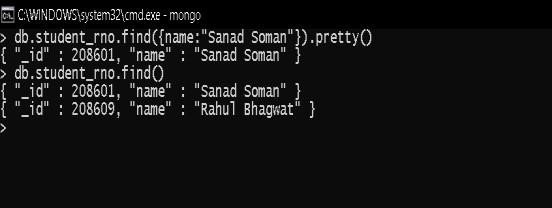
“**db.collection\_name.insert({key\_value pairs})**”.



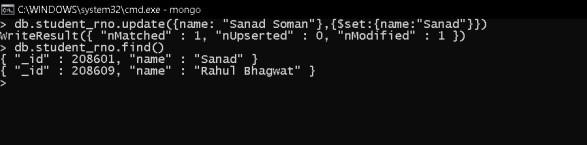
1. To display data in json format.



1. To fetch specific data based on criteria we use command: “**db.collection\_name.find({keyvaluepair}).pretty()**”.



1. To update we use command:”db.collection\_name.update({key\_value condition},{$set:keyvalue pair})”.



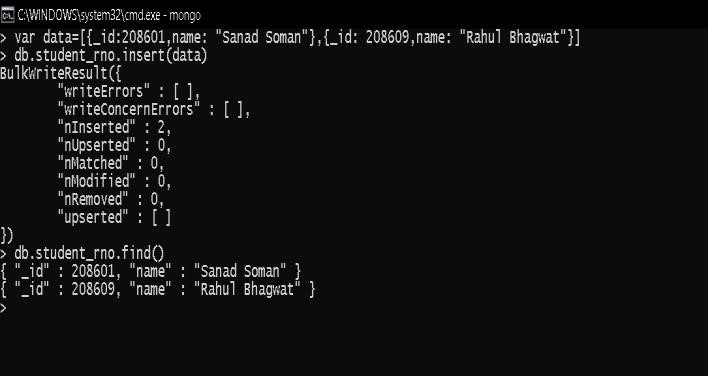
1. We can remove a record from a collection using command :”**db.collection\_name.remove({key-value pair condition”})**”.



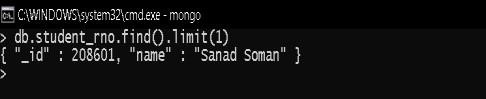
1. To remove all the records use command: ”**db.collection\_name.remove({})**”.



1. To insert multiple records in collection we must create an array of records and pass that array as parameter to command:”**db.collection\_name.insert(array\_name)**”.

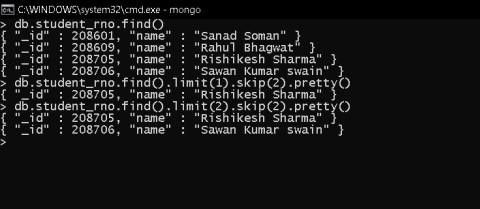


1. To limit records we need to use command **“db.collection\_name.find().limit(num\_records)”.**

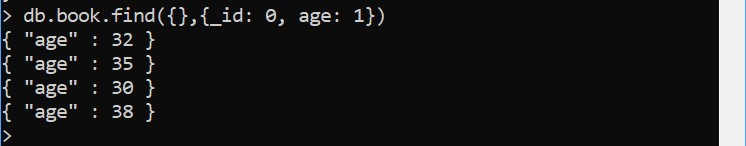


1. To skip records use command:

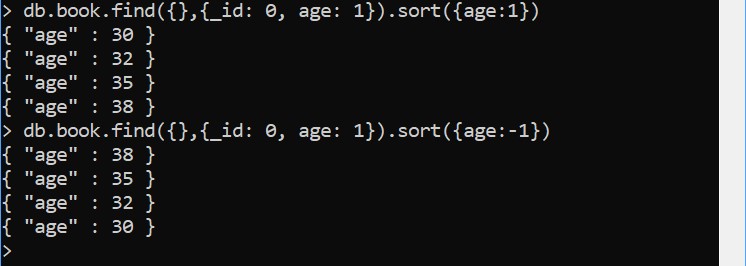
“**db.collection\_name.find().limit(num\_records).skip(num\_records).pretty()**”.



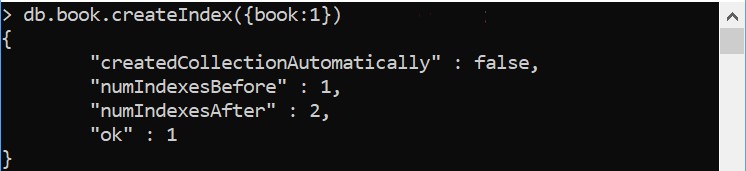
1. To fetch a specific field.



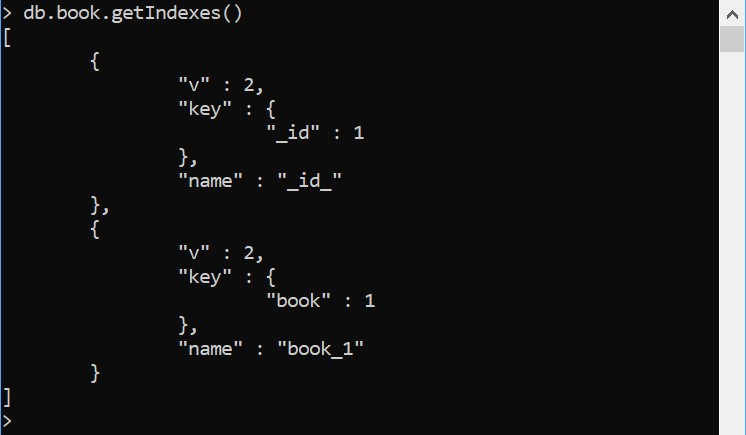
1. To sort the data: 1 will sort it in ascending order and -1 will sort it in descending order.



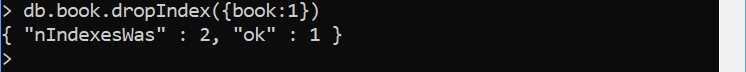
1. To create the index:



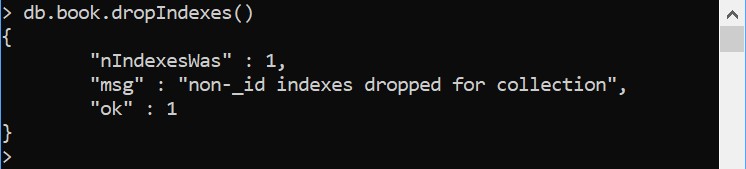
1. To find the index:



1. To drop the index:



1. To drop all the indexes:

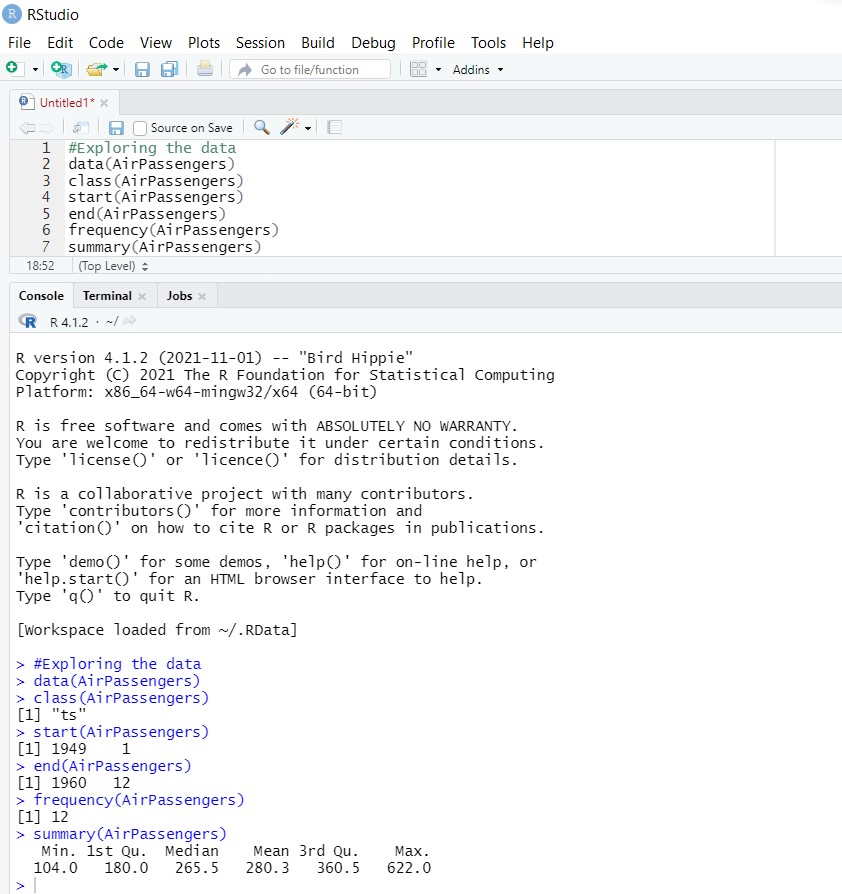


**Practical No: 7**

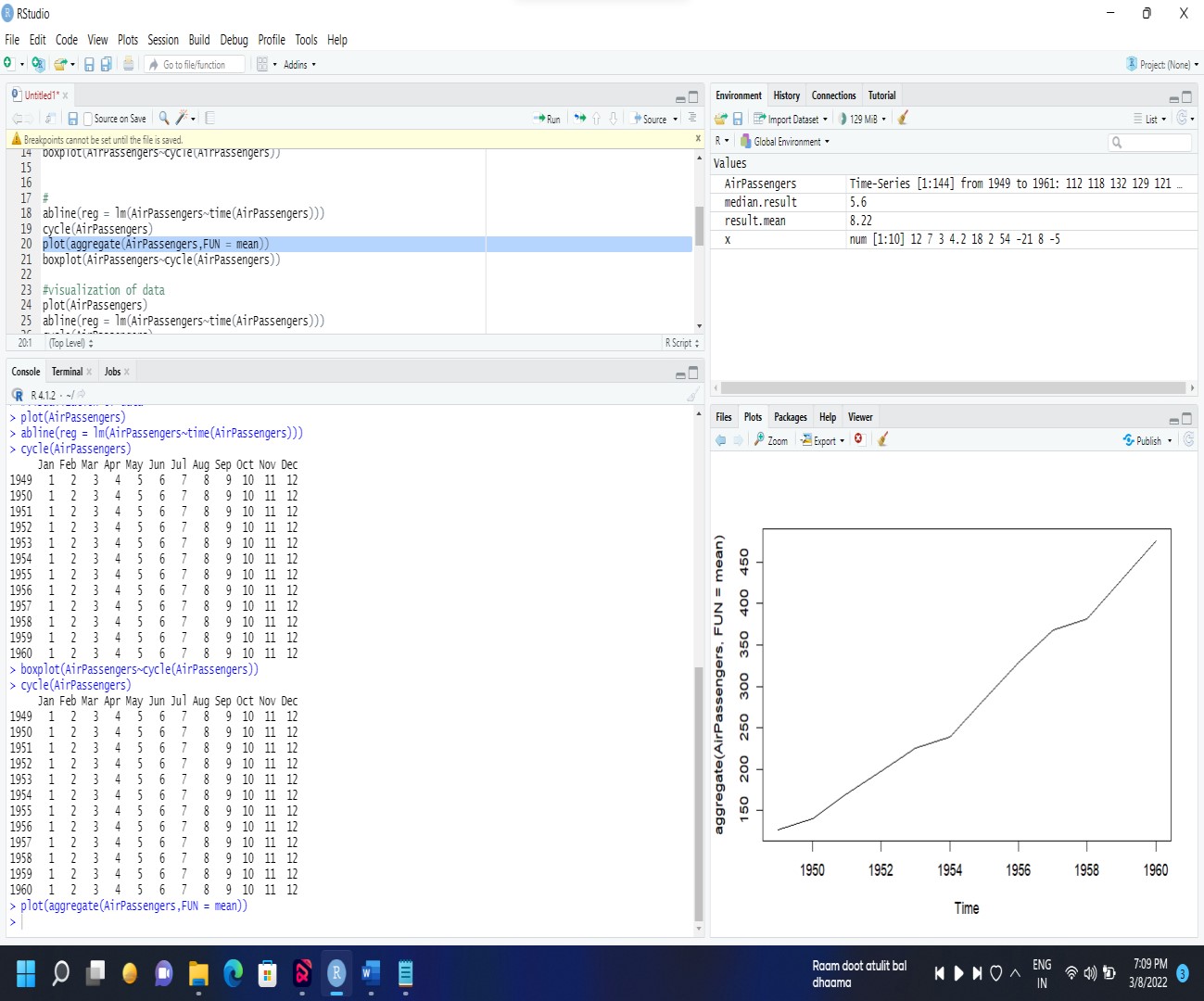
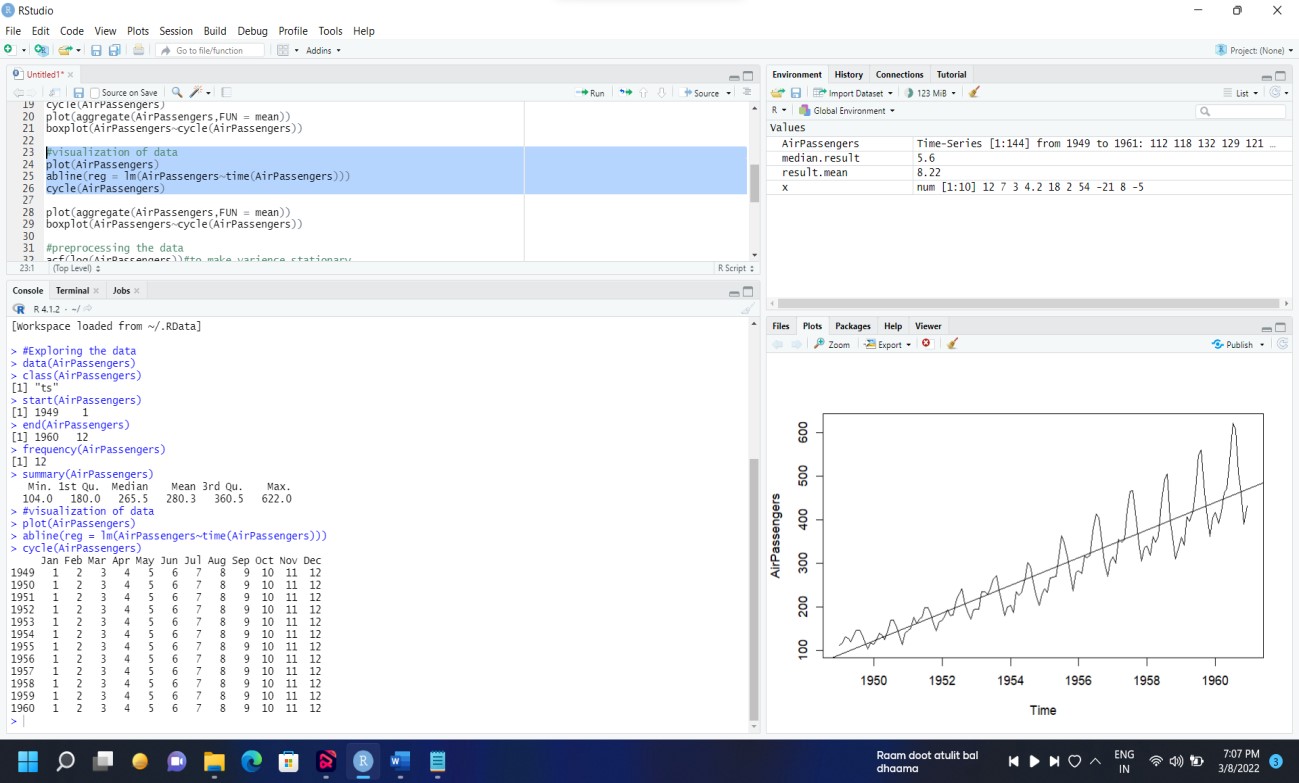
**Aim:** Practical of Time Series.

1) The AirPassenger dataset in R provides monthly totals of US airline passengers, from 1949 to

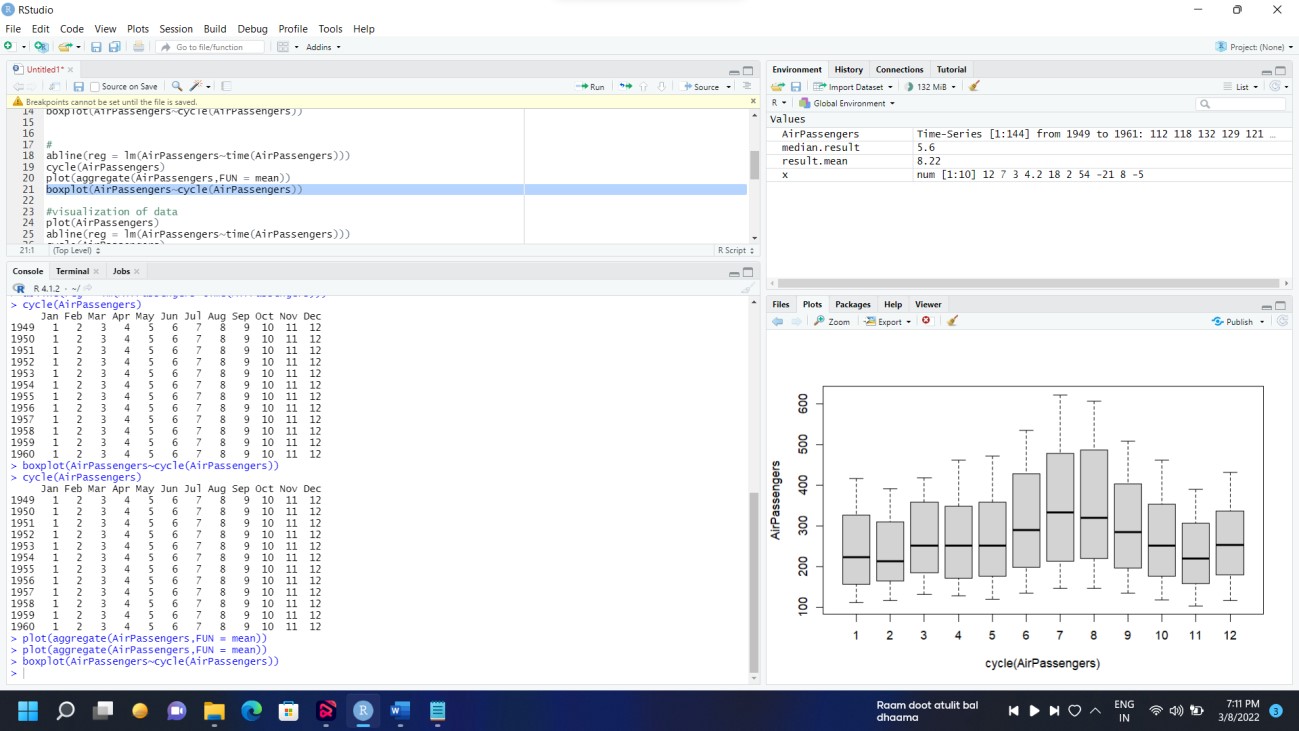
1960. This dataset is already of a time series class therefore no further class or date manipulation is required.



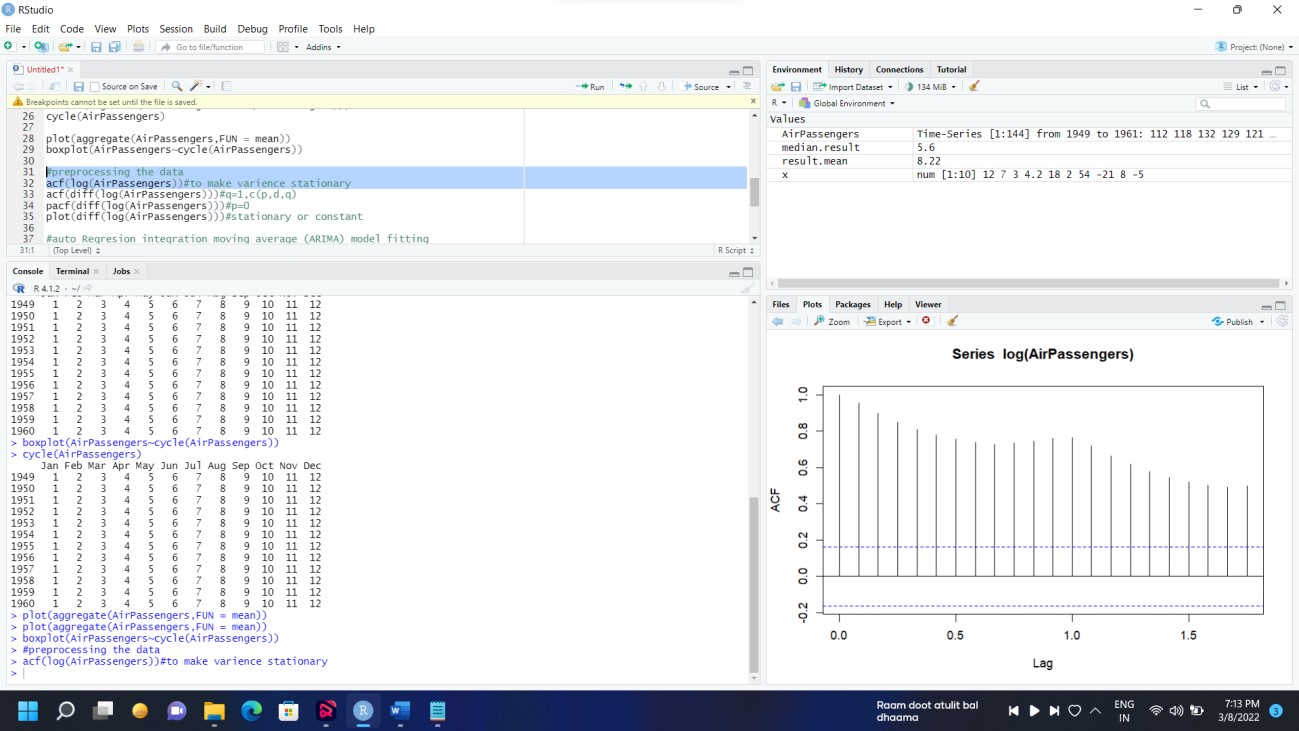
1. From the following plot we can say that the passenger numbers increase over time with each year in linear form.



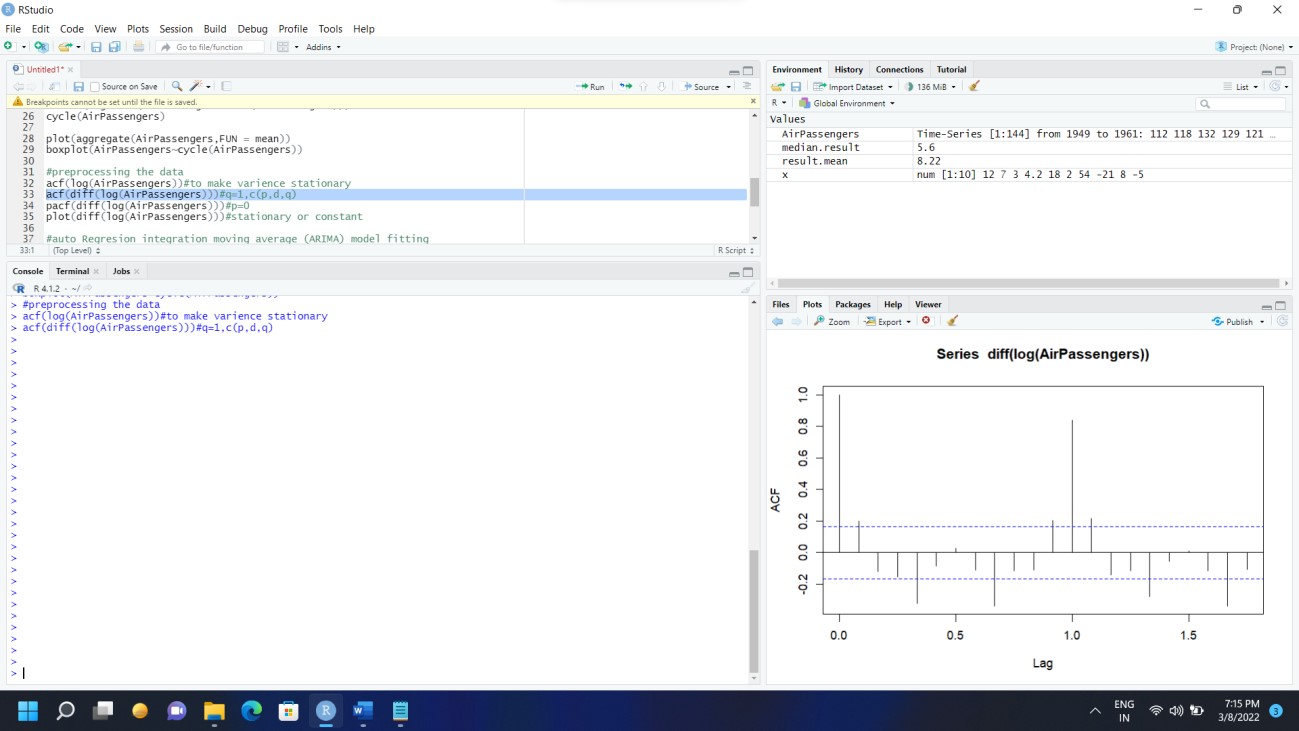
1. In the boxplot there are more passengers travelling in months 7 and 8 with higher means and higher variances than the other months.



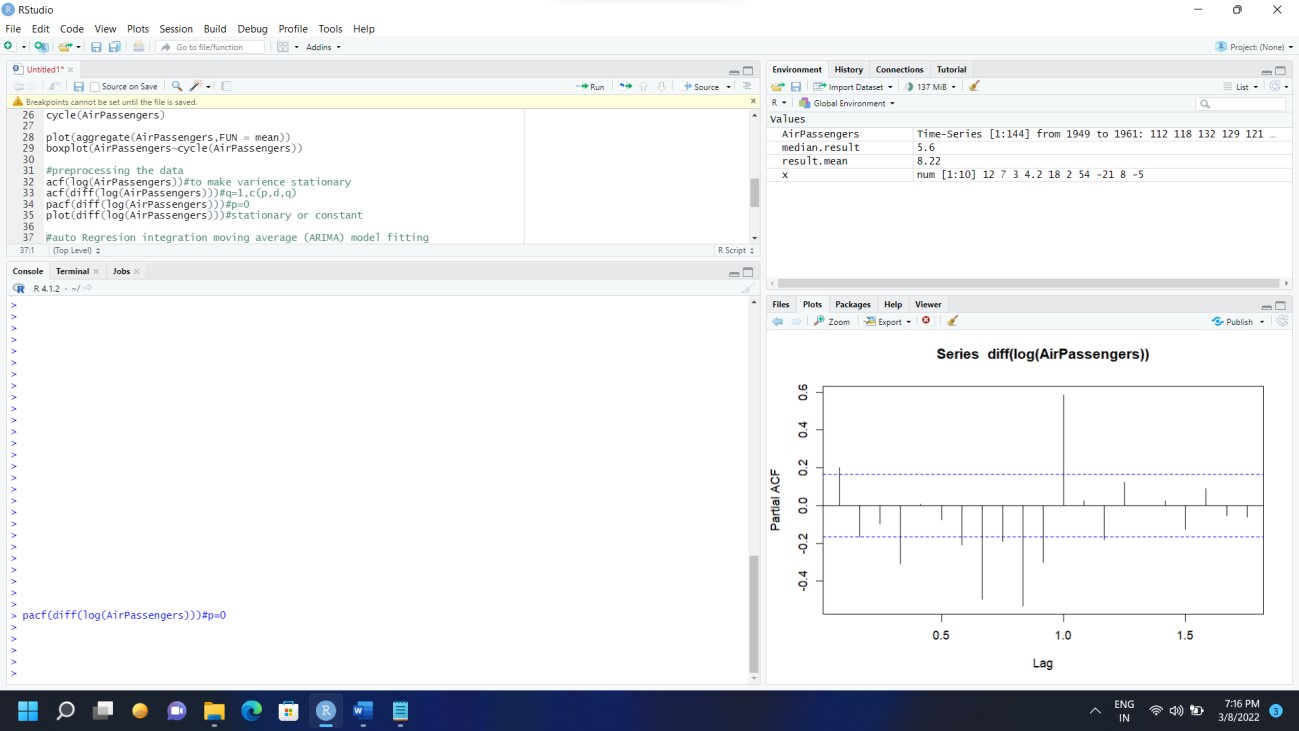
1. To make the variance stationary we will use the Autocorrelation Function (acf).



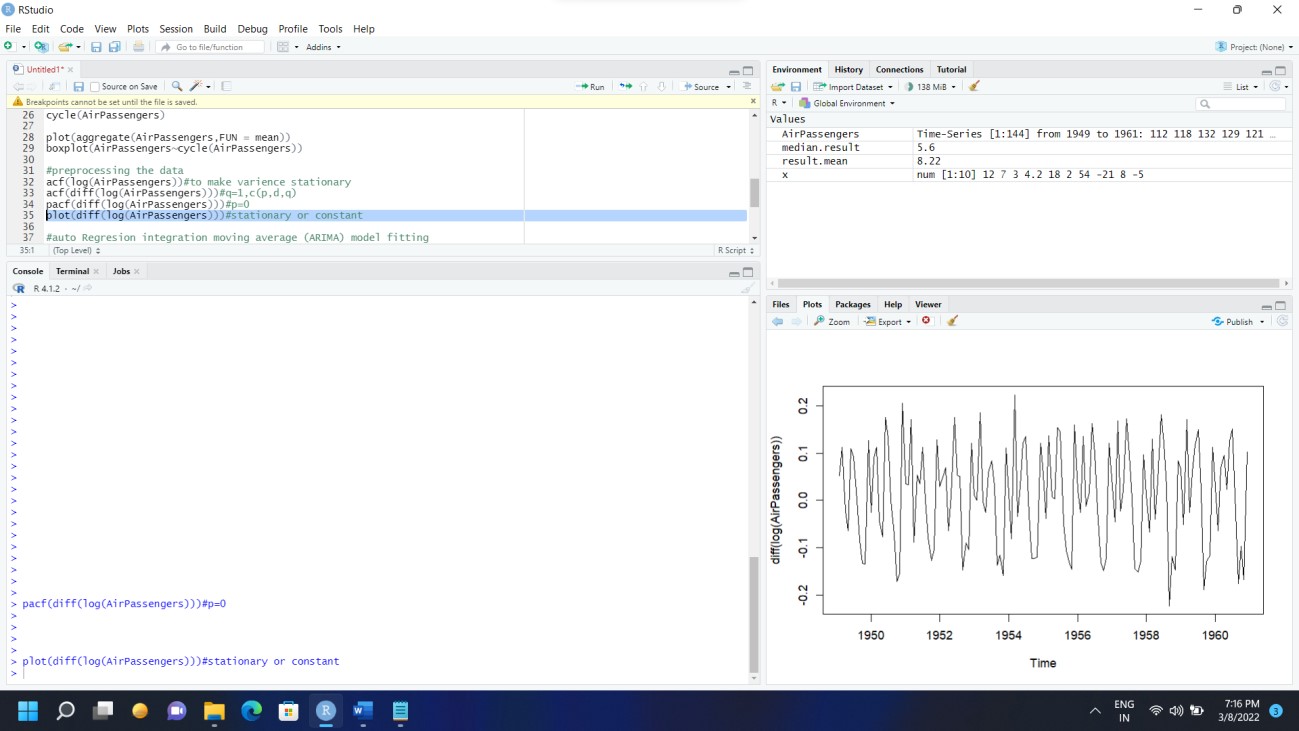
1. We can see that the acf of the residuals is centered around 0.



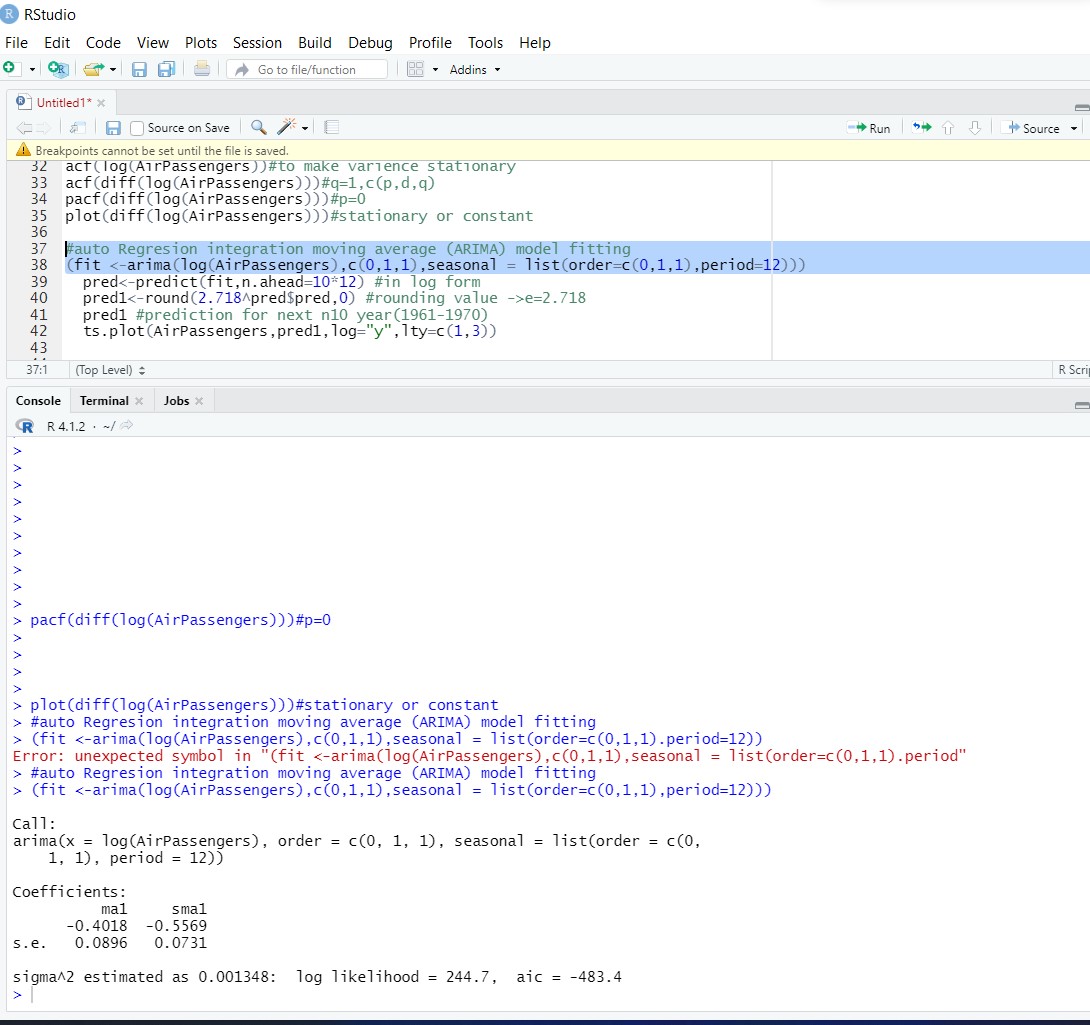
1. Partial acf(Autocorrelation Function).



1. From the following plot we can say that the Means and variance are stationary.



1. ARIMA Model fitting.



1. In the following data we can see the prediction value for the next 10 year. Which is from Year 1961 to 1970. In the graph the dotted lines represent that value is increasing in linear form for the next 10 year as well.

