```
# Load the dataset into R
data <- read.csv("/content/dataset.csv")
head(data)
```

A data.frame: 6 × 10

	X	open	high	low	close	volume	marketCap	timestamp	crypto_name	date
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<chr>	<chr>
1	0	112.90000	118.80000	107.14300	115.91000	0	1288693176	2013-05-05T23:59:59.999Z	Bitcoin	2013-05-05
2	1	3.49313	3.69246	3.34606	3.59089	0	62298185	2013-05-05T23:59:59.999Z	Litecoin	2013-05-05
3	2	115.98000	124.66300	106.64000	112.30000	0	1249023060	2013-05-06T23:59:59.999Z	Bitcoin	2013-05-06
4	3	3.59422	3.78102	3.11602	3.37125	0	58594361	2013-05-06T23:59:59.999Z	Litecoin	2013-05-06
5	4	112.25000	113.44400	97.70000	111.50000	0	1240593600	2013-05-07T23:59:59.999Z	Bitcoin	2013-05-07
6	5	3.37087	3.40672	2.93979	3.33274	0	58051265	2013-05-07T23:59:59.999Z	Litecoin	2013-05-07

Converting all anonymous attributes to suitable data types

```
# Load the dataset into R
data <- read.csv("/content/dataset.csv")

# Convert date to date type
data$date <- as.Date(data$date)

# Convert open, high, low, close, volume attributes to numeric
data$open <- as.numeric(data$open)
data$high <- as.numeric(data$high)
data$low <- as.numeric(data$low)
data$close <- as.numeric(data$close)
data$volume <- as.numeric(data$volume)



# View the structure of the dataset
str(data)

# View the first few rows of the dataset
head(data)
```

```
'data.frame':   24414 obs. of  10 variables:
 $ X          : int  0 1 2 3 4 5 6 7 8 9 ...
 $ open       : num  112.9 3.49 115.98 3.59 112.25 ...
 $ high       : num  118.8 3.69 124.66 3.78 113.44 ...
 $ low        : num  107.14 3.35 106.64 3.12 97.7 ...
 $ close      : num  115.91 3.59 112.3 3.37 111.5 ...
 $ volume     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ marketCap  : num  1.29e+09 6.23e+07 1.25e+09 5.86e+07 1.24e+09 ...
 $ timestamp  : chr  "2013-05-05T23:59:59.999Z" "2013-05-05T23:59:59.999Z" "2013-0
 $ crypto_name: chr  "Bitcoin" "Litecoin" "Bitcoin" "Litecoin" ...
 $ date       : Date, format: "2013-05-05" "2013-05-05" ...
```

A data.frame: 6 × 10

	X	open	high	low	close	volume	marketCap	time
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
1	0	112.90000	118.80000	107.14300	115.91000	0	1288693176	20 05T23:59:59
2	1	3.49313	3.69246	3.34606	3.59089	0	62298185	20 05T23:59:59
3	2	115.98000	124.66300	106.64000	112.30000	0	1249023060	20 06T23:59:59
4	3	3.59422	3.78102	3.11602	3.37125	0	58594361	20 06T23:59:59

Finding missing and sentinel values and do the necessary preprocessing

```
# Check for missing values in the dataset
missing_data <- is.na(data)

# Summarize the number of missing values in each column
colSums(missing_data)

# Handle missing values

#  Impute missing values with mean/median/mode
# Impute missing values in the open column with mean
data$open[is.na(data$open)] <- mean(data$open, na.rm = TRUE)
# Impute missing values in the high column with median
data$high[is.na(data$high)] <- median(data$high, na.rm = TRUE)
# Impute missing values in the low column with mode
mode_low <- names(which.max(table(data$low)))
data$low[is.na(data$low)] <- mode_low
```

```
X:         0 open:         0 high:         0 low:         1 close:         1 volume:         1 marketCap:
      1 timestamp:         0 crypto_name:         0 date:         1
```

Sentinel values are special values that are used to represent missing or unknown data. They are used as a placeholder for missing data, and are often used in situations where it is not possible or practical to represent missing data with the standard missing value indicator (NA). Sentinel values can be any value, but they are often chosen to be a value that is unlikely to occur in the dataset, such as -1, 999, or -999. The idea behind sentinel values is that they allow missing data to be distinguished from actual data, so that the data can be properly handled during data analysis and modeling.

```
# Find sentinel values
sentinel_values <- data == -1
sum(sentinel_values)

# Replace sentinel values with NAs
data[sentinel_values] <- NA

# Remove rows with NAs
data <- data[complete.cases(data),]
```

```
<NA>
```

```
# Summarize the numerical variables
summary(data[,c("open","high","low","close","volume")])
```

```
      open                high               low                close         
 Min.   :    0.000   Min.   :    0.000   Length:24413       Min.   :    0.000  
 1st Qu.:    0.049   1st Qu.:    0.052   Class :character   1st Qu.:    0.049  
 Median :    1.002   Median :    1.013   Mode  :character   Median :    1.002  
 Mean   :  283.360   Mean   :  292.712                      Mean   :  283.633  
 3rd Qu.:   17.762   3rd Qu.:   18.708                      3rd Qu.:   17.810  
 Max.   :19475.801   Max.   :20089.000                      Max.   :19497.400  
     volume         
 Min.   :0.000e+00  
 1st Qu.:7.476e+05  
 Median :9.452e+06  
 Mean   :4.268e+08  
 3rd Qu.:1.282e+08  
 Max.   :2.656e+10  
```

Transform the skinny dataset to a wide shaped dataset using reshape function and check correctness using melt function

```
# Convert the dataset from skinny to wide format
wide_data1 <- reshape(data, idvar = "date", timevar = "crypto_name", direction = "wide")
head(wide_data1)
```

	date	X.Bitcoin	open.Bitcoin	high.Bitcoin	low.Bitcoin	close.Bitcoin	\
	<date>	<int>	<dbl>	<dbl>	<chr>	<dbl>	
1	2013-05-05	0	112.900	118.800	107.1429977417	115.910	
3	2013-05-06	2	115.980	124.663	106.6399993896	112.300	

```
install.packages("MASS")
install.packages("reshape2")
install.packages("reshape")

library(MASS)
library(reshape2)
library(reshape)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘plyr’, ‘Rcpp’


Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)


Attaching package: ‘reshape’


The following objects are masked from ‘package:reshape2’:

    colsplit, melt, recast
```

```
# Check the correctness of reshape function using melt function
melt_data1 <- melt(wide_data1, id.vars = "date",timevar = "crypto_name")
head(melt_data1,100)
# Verify that the number of rows in the original dataset and the melted dataset match
if(nrow(data) == nrow(melt_data1)){
  print("valid")
}
```

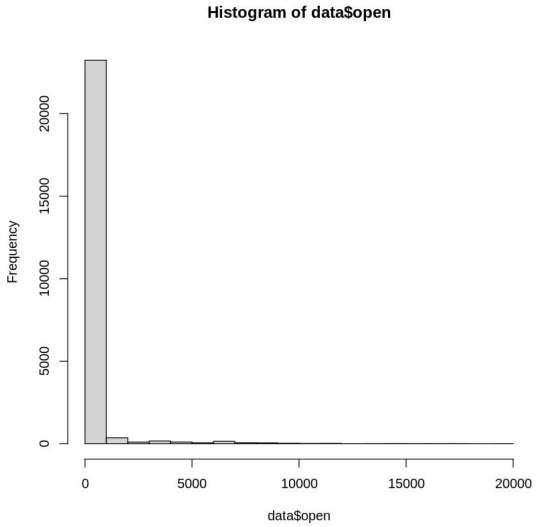
A data.frame: 100 × 3

	date	variable	value
	<date>	<fct>	<chr>
1	2013-05-05	X.Bitcoin	0
2	2013-05-06	X.Bitcoin	2
3	2013-05-07	X.Bitcoin	4
4	2013-05-08	X.Bitcoin	7
5	2013-05-09	X.Bitcoin	9
6	2013-05-10	X.Bitcoin	10
7	2013-05-11	X.Bitcoin	12
8	2013-05-12	X.Bitcoin	15
9	2013-05-13	X.Bitcoin	17
10	2013-05-14	X.Bitcoin	18
11	2013-05-15	X.Bitcoin	20
12	2013-05-16	X.Bitcoin	22

```
# Generate histograms for the numerical variables
hist(data$open)
hist(data$high)
hist(data$low)
hist(data$close)
hist(data$volume)

# Generate boxplots for the numerical variables
boxplot(data$open)
boxplot(data$high)
boxplot(data$low)
boxplot(data$close)
boxplot(data$volume)

# Generate scatter plots for the numerical variables
plot(data$open ~ data$high)
plot(data$low ~ data$close)
plot(data$volume ~ data$close)
```




```
Error in hist.default(data$low): 'x' must be numeric
Traceback:

1. hist(data$low)
2. hist.default(data$low)
3. stop("'x' must be numeric")
```

SEARCH STACK OVERFLOW