Record buffer ($0) and Use of NR (Record number) :

```
awk '{print NR}' EMP.dat

awk '{print NR,$0}' EMP.dat

awk '{print NR","$0}' EMP.dat

awk '{print NR"--SUBBU--"$0}' EMP.dat

awk 'NR==1' EMP.dat

awk 'NR==1{print}' EMP.dat

awk 'NR==1{print $0}' EMP.dat

awk 'NR==1{}' EMP.dat

awk 'NR<=3' EMP.dat

awk 'NR<=3{print NR,$0}' EMP.dat
```

Use of field buffer ($1 , $2 ....) , NF is the last field :

```
awk 'NR<=3{print $1}' EMP.dat

echo "Suubu Sama" |awk '{print $1}'

awk -F"|" 'NR<=3{print $1}' EMP.dat

awk -F"|" '{print $1}' EMP.dat

awk -F"|" '{print $1,NF}' EMP.dat

awk -F"|" '{print $1,$NF}' EMP.dat
```

Difference between NR and $NR :

```
awk -F"|" '{print $NR}' EMP.dat

awk -F"|" '{print NR}' EMP.dat
```

Condition on a specific field : (NF - Last filed , (NF-1) - Last but one filed ..)

```
awk -F"|" '$(NF-2) >=3000 {print}' EMP.dat

awk -F"|" '$(NF-2) >=3000 {print $2"-"$1"-"$(NF-2)}' EMP.dat

awk -F"|" '$(NF-2) >=3000 {print $2"-"$1"-"$NF-2}' EMP.dat

awk -F"|" '$(NF-2) >=3000 {print $2"-"$1"-"$(NF-2)}' EMP.dat
```

Search records based on Pattern or regular expression :

```
awk '/SCOTT/' EMP.dat

awk '/MANAGER/' EMP1.dat

awk '$3~/MANAGER$/' EMP1.dat

awk '$3~/MANAGER/' EMP1.dat

awk -F"|" '$3~/MANAGER$/' EMP1.dat

awk -F"|" '/SCOTT/' EMP1.dat

awk -F"|" '$2~/SCOTT/' EMP1.dat

awk -F"|" '$2~/SCOTT/{print $3}' EMP1.dat
```

awk can process the command output :

```
df -h|awk 'NR>1 && /%/ {print $NF,$(NF-1)}'

echo "A S D F" |awk 'BEGIN{OFS=":";}{print $0}'
```

```
echo "A S D F" |awk 'BEGIN{OFS=":";}{print $1,$2}'

echo "A S D F" |awk 'BEGIN{OFS=":";}{print $1,$2,$3}'

echo "A S D F" |awk 'BEGIN{OFS=":";}{print $0}'

echo "A S D F" |awk 'BEGIN{OFS=":";}{print $0,"Hello"}'
```

we can specify multiple filed separators :

```
awk -F["|,"] '{print $1}' *

awk -F["|,"] '{print FNR,$1}' *
```

BEGIN , PROCESSING and END block :

```
awk 'BEGIN{FS="|";} {print $2} END{print "END of EMPLOYEE RECORD"}' EMP.csv

awk 'BEGIN{FS="|";} {print $2} END{print "END of EMPLOYEE RECORD"}' EMP.dat

awk 'BEGIN{FS="|";print "EMPLOYEE REPORT"} {print $2} END{print "END of EMPLOYEE RECORD"}'
EMP.dat

awk 'BEGIN{FS="|";print "EMPLOYEE REPORT"} {print $2}' EMP.dat

awk 'BEGIN{FS="|";print "EMPLOYEE REPORT"}END{print "END of EMPLOYEE RECORD"}' EMP.dat

awk 'END{print "END of EMPLOYEE RECORD"}' EMP.dat

awk 'BEGIN{FS="|";} END{print $0}' EMP.dat

awk 'END{print $0}' EMP.dat
```

Arithmatic opertions :

```
awk 'BEGIN{print 4+3}'
```

```
awk 'BEGIN{print 100/3}'

awk 'BEGIN{print 12%4}'

awk 'BEGIN{print 0.0909090*7}'

awk 'BEGIN{A=34;print ++A;}'

awk 'BEGIN{A=34;print ++A;}'

awk 'BEGIN{A=34;B=100;print A/B;}'

awk '{A=34;B=100;print A/B;}'
```

Example :

```
awk 'BEGIN{FS="|" ;print;print "ENAME JOINING-DT SAL BONUS";} $3~/MANAGER/{print $2,$5,$(NF-2),$(NF-2)*3} END{print "END-OF-BONUS_REPORT"}' EMP.dat

awk -f Bonus.awk EMP.dat
```

Use of OFS :

```
echo "A B C D" |awk 'BEGIN{OFS=":";}{print $2,$3}'

echo "A B C D" |awk '{print $2,$3}'

echo "A B C D" |awk 'BEGIN{OFS=":";}{print $2,$3}'

echo "A B C D" |awk 'BEGIN{OFS=":";}{print $0}'

echo "A B C D" |awk 'BEGIN{OFS=":";}{print $0,"Hello"}'

echo "A B C D" |awk 'BEGIN{OFS=":";}{print $4,$2,$1}'
```

Use of RS :

```
awk 'BEGIN{RS="|";}NR==2' Order

awk 'BEGIN{RS="|";ORS=","}' Order

awk 'BEGIN{RS="|";ORS=","}{print $0}' Order
```

Difference between NR and FNR :

```
awk '{print NR,$0}' EMP*

awk '{print FNR,$0}' EMP*

awk 'FNR==2 && FNR==3{print FNR,$0}' EMP*

awk 'FNR==2 || FNR==3{print FNR,$0}' EMP*

awk 'FNR==2 || FNR==3{print FNR,$0}' EMP*

awk -F[",|"] 'FNR==2 || FNR==3{print FNR,$0}' EMP*

awk -F[",|"] 'FNR==2 || FNR==3{print FNR,$2}' EMP*

awk -F[,\|] 'FNR==2 || FNR==3{print FNR,$2}' EMP*
```

Use of FILENAME :

```
awk 'FNR==2 || FNR==3{print FILENAME,FNR,$0}' EMP*
```

AWK Script & use of ARGC and ARGV :

```
$ cat ./AWK/Sum.awk

#!/bin/awk -f
```

```
BEGIN{
print "Argument Count :",ARGC ;
print "Program Name:",ARGV[0] ;
print "Argument-1:",ARGV[1] ;
print "Argument-2:" , ARGV[2];
print ARGV[1]"+"ARGV[2]"="ARGV[1]+ARGV[2];
}
```

```
awk -f Sum.awk 10 20
```

AWK length function :

```
echo "Hello" |awk 'BEGIN{print length($0)}'
echo "Hello" |awk '{print length($0)}'
echo "ABCD 123 xy" |awk '{print length($2)}'
echo "ABCD 123 xy" |awk '{print length($3)}'
echo "ABCD 123 xy" |awk '{print length($1)}'
echo "ABCD 123 xy" |awk '{print length($0)}'
awk 'BEGIN{FS="|";} length($2)==6 {print $2}' EMP.dat
```

AWK index Function :

```
echo "AbcA:cA" |awk '{print index($0,"c")}'
```

```
echo "AbcA:cA" |awk '{print index($0,":")}'

echo "AbcA:cA" |awk '{print index($0,"x")}'

echo "here we go subbusamsubbu" |awk '{print index($0,"subbu")}'

echo "here we go subbusamsubbu" |awk '{print index($0,"subhasis")}'

awk 'BEGIN{FS="|";} index($2,"C") !=0 {print $2}' EMP.dat

awk 'BEGIN{FS="|";} index($2,"C") !=0 || index($2,"R") !=0 {print $2}' EMP.dat

ps -eaf |awk 'index($5,":") !=0 {print $1,$2,$5}'

ps -eaf |awk 'index($5,":") ==0 {print $1}' |sort|uniq -c|sort -n|tail -1 |awk '{print $2}'
```

AWK substr function :

```
echo "Suubu Sama" |awk '{print substr($1,1,3)}'

echo "Suubu Sama" |awk '{print substr($1,5,3)}'

echo "Suubu Sama" |awk '{print substr($0,5,3)}'

echo "Suubu Sama" |awk '{print substr($0,5)}'

awk 'BEGIN{FS="|";} {print substr($2,1,1),substr($1,1,1)}' EMP.dat

awk 'BEGIN{FS="|";} {print substr($2,1,1)substr($1,1,1)}' EMP.dat
```

AWK toupper , tolower function :

```
echo "Hello Subbu" |awk '{print toupper($0)}'

echo "Hello Subbu" |awk '{print tolower($0)}'
```

Use of substr and toupper , tolower function :

```
awk -F"|" '{print toupper(substr($2,1,1))tolower(substr($2,2))}'

awk -F"|" '{print toupper(substr($2,1,1))tolower(substr($2,2))}' EMP.dat
```

Use of systime and system function :

```
awk 'BEGIN{print systime()}'

awk 'BEGIN{system("date")}'

awk 'BEGIN{system("date;ps")}'
```

AWK IGNORECASE example :

```
awk '/[Ss][cC][oO][Tt][Tt]/' EMP.dat

awk 'BEGIN{IGNORECASE=1;} /SCOTT/' EMP.dat

awk 'BEGIN{IGNORECASE=87878787;} /SCOTT/' EMP.dat

awk 'BEGIN{IGNORECASE=0;} /SCOTT/' EMP.dat

awk '/[Ss][cC][oO][Tt]*/' EMP.dat
```

AWK split function :

```
awk '{split($0,EMPREC,"|");print EMPREC[2]}' EMP.dat

awk '{split($0,EMPREC,"|");print EMPREC[5]}' EMP.dat

awk 'BEGIN{FS=",";} {split($2,PH,"|");print $1,PH[1]} ' Data
```

Use of if - else block in AWK :

```
$ cat ./AWK/Bonus_report.awk
#!/bin/awk -f

BEGIN{

FS="|" ;
print "--------------------------------";
print system("date");
print "--------------------------------";
printf "%-15s%-15s%-15s%-10s\n","ENAME","JOIN-DT","SALARY","FLAG" ;
}

{
      if($3=="MANAGER")
      {
            printf "%-15s%-15s%-15s%-10s\n",$2,$5,$(NF-2),"NO" ;
      }
      else
      {
            printf "%-15s%-15s%-15s%-10s\n",$2,$5,$(NF-2),"YES" ;
      }
}
```

```
END{
print "-------END-----------" ;
}

awk -f Bonus_report.awk EMP.dat
```

Use of while loop in AWK :

```
$ cat While.awk
#!/bin/awk -f

BEGIN{

i=0;
while ( i<=10)
{
      print i"*"i"="i*i ;
      i++;
}

}

awk -f While.awk
```

Use of For loop :

```
$ cat For.awk
#!/bin/awk -f

BEGIN{
    for (i=1;i<=10;i++)
    {
        for(j=1;j<=10;j++)
        {
            printf "%-4s",i*j;
        }
        printf "\n";
    }
}
```

```
awk -f Fold.awk
```

```
$ cat ./AWK/For-1.awk
#!/bin/awk -f
BEGIN{
    COMP="CENTURYLINK";
    for(i=1;i<=length(COMP) ;i++)
    {
```

```
        print substr(COMP,i,1);
    }
}
```

```
chmod +x Fold.awk
./Fold.awk
```

Use of getline :

```
awk '/SMITH/{print}' EMP.dat
awk '/SMITH/{print;getline;print}' EMP.dat
awk '/SMITH/{print;getline;print;getline;print}' EMP.dat
```

AWK next :

```
$cat Report.awk

#!/bin/awk -f

BEGIN{
FS="|";
}

{
```

```
        if ($3 == "PRESIDENT" )
        {
                print "Bonus will not calculated for :",$2 ,$3;
                next;
        }
        else
        {
                print $2,$(NF-2),$(NF-2)*3;
        }
        print NR,"-- Control is here ...."

}
```

```
awk -f Report.awk EMP.dat
```

Use of continue and break :

```
$ cat Continue-Break.awk
#!/bin/awk -f

BEGIN{
        i=1;
        while (i<=10)
        {
```

```
        if (i==5)
        {
            i++;
            continue;
        }
        else
        {
            print i;
        }
    i++ ;
    }

i=1;
    while (i<=10)
    {
        if (i==5)
        {
            break;
        }
        else
        {
            print i;
        }
    i++ ;
    }
```

}

awk -f Continue-Break.awk

AWK associate array :

ps -eaf |awk '{PROCESS[$1]++;}'

ps -eaf |awk '{PROCESS[$1]++;} END{ for (user in PROCESS){print user} }'

ps -eaf |awk '{PROCESS[$1]++;} END{ for (user in PROCESS){print user,PROCESS[user]} }'

ps -eaf |awk 'NR>1{PROCESS[$1]++;} END{ for (user in PROCESS){print user,PROCESS[user]} }'

ps -eaf |awk 'NR>1{PROCESS[$1]++;} END{ for (user in PROCESS){printf "%-20s%-10s\n",user,PROCESS[user]} }'

$ cat AWK/order

subbu1,1234,Iphone

Rakesh,3456,tab

Mihir,432,xx

subbu,1234,Iphone

Rakesh1,3456,tab

awk 'BEGIN{FS="," ;}{ORDER[$2]++;}END{for(ord in ORDER){if(ORDER[ord]>1){print ord;}}}'

awk 'BEGIN{FS="," ;}{ORDER[$2]++;}END{for(ord in ORDER){if(ORDER[ord]>1){print ord;}}}' order

awk 'BEGIN{FS=","} {ORDERS[$2]++ ;} END { for(order in ORDERS) {if (ORDERS[order] >1) {print order} } }
' order

```
$ cat Dup-order.awk
#!/bin/awk

BEGIN{
FS=",";
}

{
    ORDERS[$2]++ ;
}

END {
for(order in ORDERS)
{
    if (ORDERS[order] >1)
    {
        print order;
    }
}
}
```

```
awk -f Dup-order.awk order
```

User defined function :

```
$ cat Multiplication_Table.awk
#!/bin/awk -f
## User Defined function
function usage()
{
	print "Usage: <Programname> <number> | <Programname> <number1> <number2>" ;
	exit;
}

BEGIN{
if (ARGC <2)
{
	usage()
}
if(ARGC==2)
{
	START_NUM=ARGV[1];
	END_NUM=START_NUM;
}
if(ARGC==3)
{
	if ( ARGV[1] < ARGV[2] )
```

```
        {
        START_NUM=ARGV[1];
        END_NUM=ARGV[2];
        }
        else
        {
                print "Second number should be higher than first number";
                usage();
        }
}

for(i=START_NUM;i<=END_NUM;i++)
{
        for(j=1;j<=10;j++)
        {
        printf "%-5s",i*j;
        }
        printf "\n";
}

}
```

```
awk -f Multiplication_Table.awk 1
awk -f Multiplication_Table.awk
awk -f Multiplication_Table.awk 2 5
```

Use of ENVIRON :

awk 'BEGIN{print ENVIRON["comp"]}'

awk 'BEGIN{print ENVIRON["comp"]}'

Use of Shell variable :

$ comp=ctl

$ awk 'BEGIN{print "'$comp'"}'

ctl

Use of option -v :

$comp1=savis

awk -v comp=$comp1 'BEGIN{print comp}'