Understanding ROLLBACK
======================

```
SQL> SELECT * FROM table2 ORDER BY col1;

      COL1 COL2                 C
---------- -------------------- -
         1 Deepak               M
         2 Adnan                M
         3 Sony                 F
         4 Harshan              M
         5 Yamini               F
         6 Ranjani              F
         7 Pavan                M
         8 Tanmayee             F

8 rows selected.

SQL> -- #1 -> Insertion
SQL> INSERT INTO table2 VALUES( 9, 'Sandeep', 'M');

1 row created.

SQL> -- #2 -> Updating
SQL> UPDATE table2
  2  SET col2 = 'Sony Joy'
  3  WHERE col1 = 3;

1 row updated.

SQL> -- #3 -> Deleting a row
SQL> DELETE FROM table2 WHERE col2 = 'Yamini';

1 row deleted.

SQL> SELECT * FROM table2 ORDER BY col1;

      COL1 COL2                 C
---------- -------------------- -
         1 Deepak               M
         2 Adnan                M
         3 Sony Joy             F
         4 Harshan              M
         6 Ranjani              F
         7 Pavan                M
         8 Tanmayee             F
         9 Sandeep              M

8 rows selected.
```

OBSERVE - Now the THREE DML operations (i.e. Insert, Update & Delete) are in place.
NOTE-2: Sandeep's record has been inserted. Sony name has been update to Sony Joy
         and Yamini's record has been deleted.

If we use ROLLBACK the entire DML operations will be undone.

```
SQL> ROLLBACK;

Rollback complete.

SQL> SELECT * FROM table2 ORDER BY col1;

      COL1 COL2                 C
---------- -------------------- -
         1 Deepak               M
         2 Adnan                M
         3 Sony                 F
         4 Harshan              M
         5 Yamini               F
         6 Ranjani              F
         7 Pavan                M
         8 Tanmayee             F

8 rows selected.

OBSERVE - That the insertion of 'Sandeep' record is not there, the
          updation of 'Sony' name to 'Sony Joy' is no longer there.
          Moreover, 'Yamini' record which was delete has undone. i.e
        we have got it back.

SQL>

Understanding SAVEPOINT
=======================
SQL> SELECT * FROM table2 ORDER BY col1;

      COL1 COL2                 C
---------- -------------------- -
         1 Deepak               M
         2 Adnan                M
         3 Sony                 F
         4 Harshan              M
         5 Yamini               F
         6 Ranjani              F
         7 Pavan                M
         8 Tanmayee             F

8 rows selected.

SQL> SAVEPOINT  A;

Savepoint created.

SQL> -- #1 -> Inserting
SQL> INSERT INTO table2 VALUES(9, 'Sandeep', 'M')
  2  ;

1 row created.

SQL> SAVEPOINT  B;

Savepoint created.

SQL> -- #2 -> Updating
```

```
SQL> UPDATE table2 SET col2 = 'Sony Joy' WHERE col1 = 3;

1 row updated.

SQL> SAVEPOINT  C;

Savepoint created.

SQL> -- #3 -> Deleting
SQL> DELETE FROM table2 WHERE col2 = 'Yamini';

1 row deleted.

SQL> SELECT * FROM table2 ORDER BY col1;

      COL1 COL2                 C
---------- -------------------- -
         1 Deepak               M
         2 Adnan                M
         3 Sony Joy             F
         4 Harshan              M
         6 Ranjani              F
         7 Pavan                M
         8 Tanmayee             F
         9 Sandeep              M

8 rows selected.
```

Now, if we ROLLBACK TO C; Only the last DML operation (i.e. DELETE) will be undone. Observe.

```
SQL> ROLLBACK TO C;

Rollback complete.

SQL> SELECT * FROM table2 ORDER BY col1;

      COL1 COL2                 C
---------- -------------------- -
         1 Deepak               M
         2 Adnan                M
         3 Sony Joy             F
         4 Harshan              M
         5 Yamini               F
         6 Ranjani              F
         7 Pavan                M
         8 Tanmayee             F
         9 Sandeep              M

9 rows selected.

SQL>
```

However, if we had ROLLBACK TO B; then not only the DELETE dml statement but also the UPDATE dml statement would have been undone.

```
##########################################################################
########################
DAY-2 :: SQL
Lesson-02 :: Operators & Single Row Functions
==========================================================================
======================
SQL> -- Understanding ARITHMETIC Operators
SQL>
SQL> SELECT 10 + 5 FROM dual;

      10+5
----------
        15

SQL> SELECT 10 - 5 FROM dual;

      10-5
----------
         5

SQL> SELECT 10 * 5 FROM dual;

      10*5
----------
        50

SQL> SELECT 10 * 5 "Multiplication" FROM dual;

Multiplication
--------------
            50

SQL> SELECT 10 / 5 "Division" FROM dual;

  Division
----------
         2

What is DUAL?
~~~~~~~~~~~~
DUAL is a table automatically created by Oracle Database along with the
data dictionary.
DUAL is in the schema of the user SYS but is accessible by the name DUAL
to all users.
It has one column, DUMMY , defined to be VARCHAR2(1) , and contains one
row with a value X


SQL> desc DUAL
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
-------
 DUMMY                                              VARCHAR2(1)


Example : Finding the USER name and System Date

SQL> SELECT user FROM dual;
```

```
USER
------------------------------
SCOTT

SQL> SELECT sysdate FROM dual;

SYSDATE
---------
23-SEP-21

Comparision Operators
=====================
 =          Equal to
 !=  <>  ^= Not Equal to
 >          Greater than
 >=         Greater than or Equal to
 <          Lesser than
 <=         Lesser than or Equal to

Logical Opertors
================
AND
OR
NOT


Other Comparision Operators
===========================
BETWEEN          Allows user to express a range
IN          Is similar to OR logical Operator
LIKE        Used for searching patterns
            REMEMBER  the use of _ (Underscore) and % (Percentage)
characters
IS NULL          To check for NULL values

Example for BETWEEN operator
~~~~~~~~~~~~~~~~~~~~~~~~~~~
SQL> SELECT ename, sal FROM emp
  2  WHERE sal >= 2000 AND sal <= 3000;

ENAME             SAL
---------- ----------
JONES            2975
BLAKE            2850
CLARK            2450
SCOTT            3000
FORD             3000

SQL> SELECT ename, sal FROM emp
  2  WHERE sal BETWEEN 2000 AND 3000;

ENAME             SAL
---------- ----------
JONES            2975
BLAKE            2850
CLARK            2450
SCOTT            3000
```

```
FORD            3000

SQL> -- Observe the use of NOT BETWEEN below

SQL> SELECT ename, sal FROM emp
  2  WHERE sal NOT BETWEEN 2000 AND 3000;

ENAME             SAL
---------- ----------
SMITH              800
ALLEN             1600
WARD              1250
MARTIN            1250
KING              5000
TURNER            1500
ADAMS             1100
JAMES              950
MILLER            1300

Example for IN operator
~~~~~~~~~~~~~~~~~~~~~~
SQL> SELECT ename, job FROM emp
  2  WHERE job = 'MANAGER' OR job = 'SALESMAN';

ENAME      JOB
---------- ---------
ALLEN      SALESMAN
WARD       SALESMAN
JONES      MANAGER
MARTIN     SALESMAN
BLAKE      MANAGER
CLARK      MANAGER
TURNER     SALESMAN

7 rows selected.

SQL> SELECT ename, job FROM emp
  2  WHERE job IN ('MANAGER', 'SALESMAN');

ENAME      JOB
---------- ---------
ALLEN      SALESMAN
WARD       SALESMAN
JONES      MANAGER
MARTIN     SALESMAN
BLAKE      MANAGER
CLARK      MANAGER
TURNER     SALESMAN

7 rows selected.

SQL> -- Observe the use of NOT along with IN operator below
SQL> SELECT ename, job FROM emp
  2  WHERE job NOT IN ('MANAGER', 'SALESMAN');

ENAME      JOB
---------- ---------
SMITH      CLERK
```

```
SCOTT      ANALYST
KING       PRESIDENT
ADAMS      CLERK
JAMES      CLERK
FORD       ANALYST
MILLER     CLERK

Example for LIKE operator
~~~~~~~~~~~~~~~~~~~~~~~~
Used for PATTERN matching.
To HARNESS the full power of pattern matching, use WILDCARDs

 _ (Underscore)  Is a substitute for a SINGLE character
 % (Percentage) Is a substitute for ZERO or MORE characters

[1] Getting a list of all employees who have 'LL' in their name
SQL> SELECT ename FROM emp
  2  WHERE ename LIKE '%LL%';

ENAME
----------
ALLEN
MILLER

[2] Getting a lise of employees who have 'A' as the second letter
    in their name.

SQL> SELECT ename FROM emp
  2  WHERE ename LIKE '_A%';

ENAME
----------
WARD
MARTIN
JAMES

Example for IS NULL operator
~~~~~~~~~~~~~~~~~~~~~~~~~~~
To search for NULL value in a query, we should use the IS NULL
operator.

SQL> SELECT ename, sal, comm FROM emp
  2  WHERE comm = NULL;

no rows selected

SQL> SELECT ename, sal, comm FROM emp
  2  WHERE comm IS NULL;

ENAME             SAL       COMM
---------- ---------- ----------
SMITH             800
JONES            2975
BLAKE            2850
CLARK            2450
SCOTT            3000
KING             5000
ADAMS            1100
```

```
JAMES             950
FORD             3000
MILLER           1300

10 rows selected.

SQL> SELECT ename, sal, comm FROM emp
  2  WHERE comm IS NOT NULL;

ENAME              SAL       COMM
---------- ---------- ----------
ALLEN            1600        300
WARD             1250        500
MARTIN           1250       1400
TURNER           1500          0

SQL>

Use BETWEEN instead of IN (Guideline)
~~~~~~~~~~~~~~~~~~~~~~~~~
SQL> SELECT ename, deptno  FROM emp
  2  WHERE deptno IN (10, 11, 15, 18, 20, 21, 25, 28, 30);

SQL> SELECT ename, deptno  FROM emp
  2  WHERE deptno BETWEEN 10 AND 30;

Both the above queries will give you the same output.
However, the one with BETWEEN is more efficient than the
one which uses the IN operator.

================================================================================
====================
Oracle SQL Functions
====================
[1] Single Row Functions
      Return one result per row
[2] Group Functions

Single Row Functions - can further be classified as follows:
(a) Numeric Functions
(b) String/Character Functions
(c) Date Functions
(d) Conversion Functions
(e) Miscelleanous Functions

Numeric Functions
=================
SQL> SELECT trunc(12.345, 2) FROM dual;

TRUNC(12.345,2)
---------------
          12.34

SQL> SELECT round(12.345, 2) FROM dual;

ROUND(12.345,2)
---------------
          12.35
```

```
SQL> SELECT ceil(13.456) FROM dual;

CEIL(13.456)
------------
          14

SQL> SELECT floor(13.456) FROM dual;

FLOOR(13.456)
-------------
           13

SQL> SELECT abs(13) FROM dual;

   ABS(13)
----------
        13

SQL> SELECT abs(-43) FROM dual;

  ABS(-43)
----------
        43

SQL> SELECT power(2, 5) FROM dual;

POWER(2,5)
----------
        32

SQL> SELECT power(10, 3) FROM dual;

POWER(10,3)
-----------
       1000

SQL> SELECT power(10, 3.5) FROM dual;

POWER(10,3.5)
-------------
   3162.27766

SQL> SELECT sqrt(25) FROM dual;

  SQRT(25)
----------
         5

SQL> SELECT sign(25) FROM dual;

  SIGN(25)
----------
         1

SQL> SELECT sign(-25) FROM dual;

 SIGN(-25)
```

```
----------
        -1

SQL> SELECT sign(0) FROM dual;

   SIGN(0)
----------
         0

SQL> SELECT mod(10, 3) FROM dual;

 MOD(10,3)
----------
         1
```

NOTE: There is no operator %, however we have the MOD() function

Character Functions
~~~~~~~~~~~~~~~~~~
Case Conversion Functions
~~~~~~~~~~~~~~~~~~~~~~~~
```
SQL> SELECT upper('harSHan') FROM dual;

UPPER('
-------
HARSHAN

SQL> SELECT lower('harSHan') FROM dual;

LOWER('
-------
harshan

SQL> SELECT initcap('harSHan') FROM dual;

INITCAP
-------
Harshan

SQL> SELECT initcap('soNY joY') FROM dual;

INITCAP(
--------
Sony Joy

SQL> SELECT concat(concat(first_name,  ' '),last_name) "Full Name" FROM
employees;

Full Name
----------------------------------------------
Ellen Abel
Sundar Ande
Mozhe Atkinson
David Austin
Hermann Baer
Shelli Baida
Amit Banda
```

```
NOTE: The concat() function takes TWO arguments ONLY.

SQL> SELECT substr('Computer', 4, 3) FROM dual;

SUB
---
put

SQL> SELECT substr('Capgemini', 4, 3) FROM dual;

SUB
---
gem

SQL> SELECT substr('Capgemini', 6, 4) FROM dual;

SUBS
----
mini

SQL> SELECT first_name, substr(first_name, 1, 3) "First 3 Letters" FROM
employees;

FIRST_NAME           Fir
-------------------- ---
Ellen                Ell
Sundar               Sun
Mozhe                Moz
David                Dav
Hermann              Her
Shelli               She
Amit                 Ami

SQL> SELECT length('Computer') FROM dual;

LENGTH('COMPUTER')
------------------
                 8

SQL> SELECT length('   Computer    ') FROM dual;

LENGTH('COMPUTER')
------------------
                15

SQL> SELECT length(rtrim(ltrim('   Computer    '))) FROM dual;

LENGTH(RTRIM(LTRIM('COMPUTER')))
--------------------------------
                               8

SQL> SELECT replace('Computer', 'put', 'XYZ') FROM dual;

REPLACE(
--------
ComXYZer

SQL> SELECT lpad('Hello', 10, '*') FROM dual;
```

```
LPAD('HELL
----------
*****Hello

SQL> SELECT rpad('Hello', 10, '*') FROM dual;

RPAD('HELL
----------
Hello*****

Date Functions
==============
SQL> SELECT sysdate FROM dual;

SYSDATE
---------
23-SEP-21


SQL> SELECT current_date FROM dual;

CURRENT_D
---------
23-SEP-21

SQL> SELECT sysdate, add_months(sysdate, 1) FROM dual;

SYSDATE   ADD_MONTH
--------- ---------
23-SEP-21 23-OCT-21

SQL> SELECT sysdate, add_months(sysdate, 3) FROM dual;

SYSDATE   ADD_MONTH
--------- ---------
23-SEP-21 23-DEC-21

SQL> SELECT months_between('15-AUG-47', sysdate) FROM dual;

MONTHS_BETWEEN('15-AUG-47',SYSDATE)
-----------------------------------
                          310.726118

SQL> SELECT floor( months_between('15-AUG-47', sysdate) ) FROM dual;

FLOOR(MONTHS_BETWEEN('15-AUG-47',SYSDATE))
------------------------------------------
                                       310

SQL> SELECT last_day( sysdate ) FROM dual;

LAST_DAY(
---------
30-SEP-21

SQL> SELECT last_day( '15-AUG-47' ) FROM dual;
```

```
LAST_DAY(
---------
31-AUG-47

SQL> SELECT next_day( sysdate, 'Monday' ) FROM dual;  dual;

NEXT_DAY(
---------
27-SEP-21

SQL> SELECT next_day( sysdate, 'Wednesday' ) FROM dual;

NEXT_DAY(
---------
29-SEP-21

SQL> SELECT current_timestamp FROM dual;

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
--
23-SEP-21 11.51.20.291000 AM +05:30

SQL> SELECT extract(day from current_timestamp) FROM dual;

EXTRACT(DAYFROMCURRENT_TIMESTAMP)
---------------------------------
                               23

SQL> SELECT extract(month from current_timestamp) FROM dual;

EXTRACT(MONTHFROMCURRENT_TIMESTAMP)
-----------------------------------
                                  9

SQL> SELECT extract(year from current_timestamp) FROM dual;

EXTRACT(YEARFROMCURRENT_TIMESTAMP)
----------------------------------
                              2021
SQL> select first_name, hire_date, extract(month from hire_date) from
employees
  2  where extract(month from hire_date) = 6;

FIRST_NAME           HIRE_DATE EXTRACT(MONTHFROMHIRE_DATE)
-------------------- --------- ---------------------------
Steven               17-JUN-03                           6
David                25-JUN-05                           6
Jason                14-JUN-04                           6
Martha               21-JUN-07                           6
Julia                24-JUN-06                           6

Number of days since INDIA got INDEPENDENCE
SQL> SELECT ABS(to_date('15-AUG-1947') - sysdate) FROM dual;

ABS(TO_DATE('15-AUG-1947')-SYSDATE)
-----------------------------------
                         27068.4996
```

```
Date Arithmetics
================
SQL> SELECT sysdate FROM dual;

SYSDATE
---------
23-SEP-21

SQL> SELECT sysdate + 1 "Tomorrow" FROM dual;

Tomorrow
---------
24-SEP-21

SQL> SELECT sysdate - 1 "Yesterday" FROM dual;

Yesterday
---------
22-SEP-21

Conversion Functions
====================
SQL> SELECT TO_CHAR( sysdate, 'DD Month YYYY') FROM dual;

TO_CHAR(SYSDATE,'
-----------------
23 September 2021

NOTE: Refer to FORMAT ELEMENTS like DD, MM, Month, Year etc.

SQL> SELECT TO_CHAR( sysdate, 'DD Month Year') FROM dual;

TO_CHAR(SYSDATE,'DDMONTHYEAR')
------------------------------------------------------
23 September Twenty Twenty-One

SQL> SELECT TO_CHAR( sysdate, 'Day, DD Month Year') FROM dual;

TO_CHAR(SYSDATE,'DAY,DDMONTHYEAR')
-----------------------------------------------------------------
Thursday , 23 September Twenty Twenty-One

SQL> SELECT TO_CHAR( 17575, '$99,999.00')  FROM dual;

TO_CHAR(175
-----------
 $17,575.00


Miscellaneous Functions
=======================
Recap - E F Codd rules
      Rule for systematic handling of NULL values.

SQL> SELECT ename, sal, comm, sal + comm "Net Pay"
  2  FROM emp;
```

```
ENAME             SAL       COMM    Net Pay
---------- ---------- ---------- ----------
SMITH              800
ALLEN             1600        300       1900
WARD              1250        500       1750
JONES             2975
MARTIN            1250       1400       2650


SQL> -- If any one of the operand in an Arithmetic Expression is NULL
SQL> -- the result is a NULL
SQL>

Using NVL() to handle NULL values

SQL> -- Handling NULL value with NVL() function
SQL> SELECT ename, sal, comm, sal + NVL(comm, 0) "Net Pay"
  2  FROM emp;

ENAME             SAL       COMM    Net Pay
---------- ---------- ---------- ----------
SMITH              800                 800
ALLEN             1600        300       1900
WARD              1250        500       1750
JONES             2975                 2975
MARTIN            1250       1400       2650

SQL> select col1 "Sl. No.", NVL(col2, '--No Name--') "Name", col3
"Gender" FROM table2;

   Sl. No. Name                 G
---------- -------------------- -
         1 Deepak               M
         5 Yamini               F
         2 --No Name--          M
         3 Sony Joy             F
         4 Harshan              M
         6 --No Name--          F
         7 Pavan                M
         8 Tanmayee             F
         9 Sandeep              M


Example for NVL2() function
SQL> SELECT ename, sal, comm, sal + NVL2(comm, comm, 0) "Net Pay"  FROM
emp;

ENAME             SAL       COMM    Net Pay
---------- ---------- ---------- ----------
SMITH              800                 800
ALLEN             1600        300       1900
WARD              1250        500       1750
JONES             2975                 2975
MARTIN            1250       1400       2650
BLAKE             2850                 2850

SQL> select col1 "Sl. No.", NVL2(col2, col2, '--No Name--') "Name", col3
"Gender" FROM table2;
```

```
   Sl. No. Name                 G
---------- -------------------- -
         1 Deepak               M
         5 Yamini               F
         2 --No Name--          M
         3 Sony Joy             F
         4 Harshan              M
         6 --No Name--          F
         7 Pavan                M
         8 Tanmayee             F
         9 Sandeep              M

Example on NULLIF() Function
SQL> SELECT NULLIF('Apple', 'Apple') FROM dual;

NULLI
-----
           <-- Has returned NULL (which is not displayed)
           <-- Because the 1st and 2nd argument are identifcal/equal
SQL>
SQL> SELECT NULLIF('Apple', 'Banana') FROM dual;

NULLI
-----
Apple      <-- Returned 'Apple' as the arguments are different.

Example on COALESCE() function
SQL> SELECT COALESCE( null, null, 'Capgemini', null) FROM dual;

COALESCE(
---------
Capgemini  <-- The first two arguments are NULL, thus returned Capgemini

SQL> SELECT COALESCE( 'Infosys', 'Wipro', 'Capgemini', null) FROM dual;

COALESC
-------
Infosys          <-- The first argument is non-null value.

Example on CASE function
  1  select col1 "Sl. No.", NVL2(col2, col2, '--No Name--') "Name",
  2  CASE col3
  3     WHEN 'M' THEN 'Male'
  4     WHEN 'F' THEN 'Female'
  5     ELSE 'Invalid Gender'
  6* END "Gender" FROM table2
SQL> /

   Sl. No. Name                 Gender
---------- -------------------- ------------
         1 Deepak               Male
         5 Yamini               Female
         2 --No Name--          Male
         3 Sony Joy             Female

  1  SELECT ename, job, sal,
  2  CASE job
```

```
  3     WHEN 'CLERK'    THEN sal * 0.10
  4     WHEN 'SALESMAN' THEN sal * 0.12
  5     WHEN 'MANAGER'  THEN sal * 0.15
  6     ELSE sal * 0.18
  7* END "Bonus"  FROM emp
SQL> /

ENAME      JOB               SAL      Bonus
---------- --------- ---------- ----------
SMITH      CLERK             800         80
ALLEN      SALESMAN         1600        192
WARD       SALESMAN         1250        150
JONES      MANAGER          2975     446.25
MARTIN     SALESMAN         1250        150
BLAKE      MANAGER          2850      427.5
CLARK      MANAGER          2450      367.5
SCOTT      ANALYST          3000        540
KING       PRESIDENT        5000        900
TURNER     SALESMAN         1500        180
ADAMS      CLERK            1100        110
JAMES      CLERK             950         95
FORD       ANALYST          3000        540
MILLER     CLERK            1300        130

Example DECODE() functions
select col1 "Sl. No.", NVL2(col2, col2, '--No Name--') "Name",
DECODE(col3, 'M', 'Male', 'F', 'Female', 'Invalid Gender') "Gender" FROM
table2;

   Sl. No. Name                 Gender
---------- -------------------- ------------
         1 Deepak               Male
         5 Yamini               Female
         2 --No Name--          Male
         3 Sony Joy             Female
         4 Harshan              Male
         6 --No Name--          Female
         7 Pavan                Male
         8 Tanmayee             Female
         9 Sandeep              Male


  1  SELECT ename, job, sal,
  2  DECODE(job, 'CLERK', sal * 0.10,
  3              'SALESMAN', sal * 0.12,
  4              'MANAGER', sal * 0.15,
  5*        sal * 0.18 ) "Bonus"  FROM emp
SQL> /

ENAME      JOB               SAL      Bonus
---------- --------- ---------- ----------
SMITH      CLERK             800         80
ALLEN      SALESMAN         1600        192
WARD       SALESMAN         1250        150
JONES      MANAGER          2975     446.25
MARTIN     SALESMAN         1250        150
BLAKE      MANAGER          2850      427.5
CLARK      MANAGER          2450      367.5
```

```
SCOTT      ANALYST         3000        540
KING       PRESIDENT       5000        900
TURNER     SALESMAN        1500        180
ADAMS      CLERK           1100        110
```

Guidelines
~~~~~~~~~~
Avoid using a SUBSTR() function in the WHERE clause.
As it does a complete TABLE scan instead of using an INDEX (if exists)
Instead use the LIKE operator.

Example:

```
SQL> select * from table2;

      COL1 COL2                 COL3
---------- -------------------- ------------
         1 Deepak               M
         5 Yamini               F
         2                      M
         3 Sony Joy             F
         4 Harshan              M
         6                      F
         7 Pavan                M
         8 Tanmayee             F
         9 Sandeep              M

9 rows selected.

SQL> select * from table2 where substr(col2, 1, 1) = 'S';

      COL1 COL2                 COL3
---------- -------------------- ------------
         3 Sony Joy             F
         9 Sandeep              M

SQL> select * from table2 where col2 LIKE 'S%';

      COL1 COL2                 COL3
---------- -------------------- ------------
         3 Sony Joy             F
         9 Sandeep              M
```
~~~~~~~~~~