INTRODUCTION TO SQL

SQL (Structured Query Language) is a database computer language designed for the retrieval and management of data in relational database management systems (RDBMS), database schema creation and modification, and database object access control management.

SQL is a programming language for querying and modifying data and managing databases. SQL was standardized first by the ANSI and (later) by the ISO. Most database management systems implement a majority of one of these standards and add their proprietary extensions. SQL allows the retrieval, insertion, updating, and deletion of data.

A database management system also includes management and administrative functions. Most -- if not all -- implementations also include a Command-line Interface (SQL/CLI) that allows for the entry and execution of the language commands, as opposed to only providing an API intended for access from a GUI.

The first version of SQL was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s. This version, initially called SEQUEL, was designed to manipulate and retrieve data stored in IBM's original relational database product, System R. IBM patented their version of SQL in 1985, while the SQL language was not formally standardized until 1986, by the American National Standards Institute (ANSI) as SQL-86. Subsequent versions of the SQL standard have been released by ANSI and as International Organization for Standardization (ISO) standards.

Originally designed as a declarative query and data manipulation language, variations of SQL have been created by SQL database management system (DBMS) vendors that add procedural constructs, control-of-flow statements, user-defined data types, and various other language extensions. With the release of the SQL:1999 standard, many such extensions were formally adopted as part of the SQL language via the SQL Persistent Stored Modules (SQL/PSM) portion of the standard.
Common criticisms of SQL include a perceived lack of cross-platform portability between vendors, inappropriate handling of missing data (see Null (SQL)), and unnecessarily complex and occasionally ambiguous language grammar and semantics.

FEATURES OF SQL:

SQL is both an easy-to-understand language and a comprehensive tool for managing data. Some of the major features of SQL are

- Vendor independence
- Portability across computer systems
- SQL standards
- IBM endorsement and commitment (DB2)
- Microsoft commitment (SQL Server , ODBC, and ADO)

- Relational foundation
- High-level, English-like structure
- Interactive, ad hoc queries
- Programmatic database access
- Multiple views of data
- Complete database language
- Dynamic data definition
- Client/server architecture
- Enterprise application support
- Extensibility and object technology
- Internet database access
- Java integration (JDBC)
- Industry infrastructure

SQL COMMANDS

SQL Consisting of DDL,DML,DCL,TCL COMMANDS.

DDL

Data Definition Language (DDL) statements are used to define the database structure or schema.
DDL Commands: Create , Alter ,Drop , Rename, Truncate

CREATE - to create objects in the database
ALTER - alters the structure of the database
DROP - delete objects from the database
TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
RENAME - rename an object

DML

Data Manipulation Language (DML) statements are used for managing data within schema objects

DML Commands: Insert ,Update, Delete, Select

INSERT - insert data into a table
UPDATE - updates existing data within a table
DELETE - deletes all records from a table, the space for the records remain
SELECT - retrieve data from the a database

DCL

Data Control Language (DCL) statements is used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it.

DCL Commands: Grant, Revoke

GRANT - gives user's access privileges to database
REVOKE - withdraw access privileges given with the GRANT command

TCL

Transaction Control (TCL) statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

TCL Commands: Commit, Rollback, Save point

COMMIT - save work done
SAVEPOINT - identify a point in a transaction to which you can later roll back
ROLLBACK - restore database to original since the last COMMIT

SYNTAX'S OF COMMANDS

CREATE TABLE

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....
);
```

ALTER A TABLE

To add a column in a table

```
ALTER TABLE table_name
ADD column_name datatype;
```

To delete a column in a table

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

DROP TABLE

```
DROP TABLE table_name;
```

TRUNCATE TABLE

```
TRUNCATE TABLE table_name;
```

INSERT

```
INSERT INTO table_name
VALUES (value1, value2, value3,...);
```

(O R)

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...);
```

UPDATE

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value;
```

DELETE

```
DELETE FROM table_name
WHERE some_column=some_value;
```

SELECT

```
SELECT column_name(s)
FROM table_name;
```

CREATE,INSERT , UPDATE, DELETE, RENAME, TRUNCATE, ON TABLES

SQL>CREATE TABLE STUDENT(SNO NUMBER(5),SNAME VARCHAR2(15),DOJ DATE);

OUTPUT:-TABLE CREATED

```
SQL> INSERT INTO STUDENT100VALUES(&SNO,'&SNAME','&DOJ');
```

OUTPUT:-

```
Enter value for sno: 501
Enter value for sname: ABI
Enter value for doj: 12-OCT-07
old   1: INSERT INTO STUDENT100 VALUES(&SNO,'&SNAME','&DOJ')
new   1: INSERT INTO STUDENT100 VALUES(501,'ABI','12-OCT-07')

1 row created.

SQL> /

Enter value for sno: 502
Enter value for sname: ASHOK
Enter value for doj: 03-OCT-07
old   1: INSERT INTO STUDENT100 VALUES(&SNO,'&SNAME','&DOJ')
new   1: INSERT INTO STUDENT100 VALUES(502,'ASHOK','03-OCT-07')

1 row created.

SQL> /

Enter value for sno: 503
Enter value for sname: BHAVYA
Enter value for doj: 10-OCT-07
old   1: INSERT INTO STUDENT100 VALUES(&SNO,'&SNAME','&DOJ')
new   1: INSERT INTO STUDENT100 VALUES(503,'BHAVYA','10-OCT-07')

1 row created.
SQL> /

Enter value for sno: 504
Enter value for sname: AKASH
Enter value for doj: 05-OCT-07
old   1: INSERT INTO STUDENT100 VALUES(&SNO,'&SNAME','&DOJ')
new   1: INSERT INTO STUDENT100 VALUES(504,'AKASH','05-OCT-07')

1 row created.
```

SQL> /

Enter value for sno: 505
Enter value for sname: NIKHIL
Enter value for doj: 08-OCT-07
old 1: INSERT INTO STUDENT100 VALUES(&SNO,'&SNAME','&DOJ')
new 1: INSERT INTO STUDENT100 VALUES(505,'NIKHIL','08-OCT-07')

1 row created.

SQL> RENAME STUDENT TO CSESTUDENT;

OUTPUT:-Table renamed.

SQL>SELECT * FROM CSESTUDENT;

OUTPUT:-

SNO	SNAME	DOJ
501	ABI	12-OCT-07
502	ASHOK	03-OCT-07
503	BHAVYA	10-OCT-07
504	AKASH	05-OCT-07
505	NIKHIL	08-OCT-07

SQL> UPDATE CSESTUDENT SET SNAME='ABILASH' WHERE SNAME='ABI';

OUTPUT:-1 row updated.

SQL> ALTER TABLE CSESTUDENT ADD BRANCH VARCHAR2(6);

OUTPUT:-Table altered.

SQL> UPDATE CSESTUDENT SET BRANCH='CSE' WHERE SNO=501;

OUTPUT:-1 row updated.

SQL> UPDATE CSESTUDENT SET BRANCH='CSE' WHERE SNO=502;

OUTPUT:-1 row updated.

SQL> UPDATE CSESTUDENT SET BRANCH='CSE' WHERE SNO=503;

OUTPUT:-1 row updated.

SQL> UPDATE CSESTUDENT SET BRANCH='CSE' WHERE SNO=504;

OUTPUT:-1 row updated.

SQL> UPDATE CSESTUDENT SET BRANCH='CSE' WHERE SNO=505;

OUTPUT:-1 row updated.

SQL> SELECT * FROM CSESTUDENT;

SNO	SNAME	DOJ	BRANCH
501	ABILASH	12-OCT-07	CSE
502	ASHOK	03-OCT-07	CSE
503	BHAVYA	10-OCT-07	CSE
504	AKASH	05-OCT-07	CSE
505	NIKHIL	08-OCT-07	CSE

SQL> DELETE FROM CSESTUDENT WHERE SNO=501;

OUTPUT:-1 row deleted.

SQL> SELECT * FROM CSESTUDENT;

OUTPUT:-

SNO	SNAME	DOJ	BRANCH
502	ASHOK	03-OCT-07	CSE
503	BHAVYA	10-OCT-07	CSE
504	AKASH	05-OCT-07	CSE
505	NIKHIL	08-OCT-07	CSE
502	ASHOK	03-OCT-07	CSE

SQL> DROP TABLE CSESTUDENT;

OUTPUT:-Table dropped.

CREATING TABLES WITH CONSTRAINTS

(NOT NULL)

SQL> CREATE TABLE STUD(ROLLNO NUMBER(6) NOT NULL,NAME VARCHAR2(10),BRANCH VARCHAR2(6));

Table created.

SQL> DESC STUD;

Name	Null?	Type
ROLLNO	NOT NULL	NUMBER(6)
NAME		VARCHAR2(10)
BRANCH		VARCHAR2(6)

SQL> INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH');
Enter value for rollno: 501
Enter value for name: ABHILASH
Enter value for branch: CSE
old 1: INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH')
new 1: INSERT INTO STUD VALUES(501,'ABHILASH','CSE')

1 row created.

SQL> /
Enter value for rollno: 502
Enter value for name: ABI
Enter value for branch: CSE
old 1: INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH')
new 1: INSERT INTO STUD VALUES(502,'ABI','CSE')

1 row created.

SQL> SELECT * FROM STUD;

ROLLNO	NAME	BRANCH
501	ABHILASH	CSE

502 ABI CSE

```
SQL> INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH');
Enter value for rollno:
Enter value for name: BHAVYA
Enter value for branch: CSE
old   1: INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH')
new   1: INSERT INTO STUD VALUES(,'BHAVYA','CSE')
INSERT INTO STUD VALUES(,'BHAVYA','CSE')
                        *
ERROR:CANNOT INSERT NULL INTO("SCOTT",'STUD',ROLLNO)
```

(UNIQUE)

```
SQL> CREATE TABLE STUD(ROLLNO NUMBER(6) UNIQUE ,NAME
VARCHAR2(10),BRANCH VARCHAR2(6));

Table created.

SQL> INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH');
Enter value for rollno: 501
Enter value for name: abhilash
Enter value for branch: cse
old   1: INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH')
new   1: INSERT INTO STUD VALUES(501,'abhilash','cse')

1 row created.

SQL> /
Enter value for rollno: 502
Enter value for name: ABI
Enter value for branch: CSE
old   1: INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH')
new   1: INSERT INTO STUD VALUES(502,'ABI','CSE')

1 row created.


SQL> /
Enter value for rollno: 502
Enter value for name: BHAVYA
```

```
Enter value for branch: CSE
old   1: INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH')
new   1: INSERT INTO STUD VALUES(502,'BHAVYA','CSE')
INSERT INTO STUD VALUES(502,'BHAVYA','CSE')
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.SYS_C001290) violated
```

(PRIMARY KEY)

```
SQL> CREATE TABLE STUD(ROLLNO NUMBER(6) PRIMARY KEY ,NAME
VARCHAR2(10),BRANCH VARCHAR2(6));

Table created.

SQL> INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH');
Enter value for rollno: 501
Enter value for name: abhilash
Enter value for branch: cse
old   1: INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH')
new   1: INSERT INTO STUD VALUES(501,'abhilash','cse')

1 row created.

SQL> /
Enter value for rollno: 502
Enter value for name: ABI
Enter value for branch: CSE
old   1: INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH')
new   1: INSERT INTO STUD VALUES(502,'ABI','CSE')

1 row created.



SQL> /
Enter value for rollno: 502
Enter value for name: BHAVYA
Enter value for branch: CSE
old   1: INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH')
new   1: INSERT INTO STUD VALUES(502,'BHAVYA','CSE')
INSERT INTO STUD VALUES(502,'BHAVYA','CSE')
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.SYS_C001290) violated
```

```
SQL> INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH');
Enter value for rollno:
Enter value for name: BHAVYA
Enter value for branch: CSE
old   1: INSERT INTO STUD VALUES(&ROLLNO,'&NAME','&BRANCH')
new   1: INSERT INTO STUD VALUES(,'BHAVYA','CSE')
INSERT INTO STUD VALUES(,'BHAVYA','CSE')
                        *
ERROR:CANNOT INSERT NULL INTO("SCOTT",'STUD',ROLLNO)
```

```
SQL> SELECT * FROM STUD;

    ROLLNO NAME       BRANCH
---------- ---------- ------
       501 ABHILASH   CSE
       502 ABI        CSE
```

(CHECK)

```
SQL>  create table stud(rno number(5),name varchar2(10),sal number(10) constraint no_ck check(sal between 10000 and 30000));

Table created.

SQL> insert into stud values(&rno,'&name',&sal);
Enter value for rno: 567
Enter value for name: sachin
Enter value for sal: 29000
old   1: insert into stud values(&rno,'&name',&sal)
new   1: insert into stud values(567,'sachin',29000)

1 row created.

SQL> /
Enter value for rno: 565
Enter value for name: rohit
Enter value for sal: 35000
old   1: insert into stud values(&rno,'&name',&sal)
new   1: insert into stud values(565,'rohit',35000)
insert into stud values(565,'rohit',35000)
```

*
ERROR at line 1:
ORA-02290: check constraint (SCOTT.NO_CK) violated

(FOREIGN KEY)

SOL>create table adm(stuid number(6) constraint stuid_pk primary key,sname varchar2(15),per number(5));

Table created.

SQL> insert into adm values(&stuid,'&sname',&per);
Enter value for stuid: 1
Enter value for sname: abi
Enter value for per: 80
old 1: insert into adm values(&stuid,'&sname',&per)
new 1: insert into adm values(1,'abi',80)

1 row created.

SQL> /
Enter value for stuid: 2
Enter value for sname: rohit
Enter value for per: 89
old 1: insert into adm values(&stuid,'&sname',&per)
new 1: insert into adm values(2,'rohit',89)

1 row created.

SQL> /
Enter value for stuid: 3
Enter value for sname: sachin
Enter value for per: 99
old 1: insert into adm values(&stuid,'&sname',&per)
new 1: insert into adm values(3,'sachin',99)

1 row created.

SQL> /
Enter value for stuid: 4
Enter value for sname: naveen
Enter value for per: 70
old 1: insert into adm values(&stuid,'&sname',&per)
new 1: insert into adm values(4,'naveen',70)

1 row created.

SQL> select * from adm;

STUID	SNAME	PER
1	abi	80
2	rohit	89
3	sachin	99
4	naveen	70

SQL> create table course(stuid number(6) constraint sid_fk references adm(stuid),branch varchar2(5),sec varchar2(10));

Table created.

SQL> insert into course values(&stuid,'&branch','&sec');
Enter value for stuid: 1
Enter value for branch: cse
Enter value for sec: a
old 1: insert into course values(&stuid,'&branch','&sec')
new 1: insert into course values(1,'cse','a')

1 row created.

SQL> /
Enter value for stuid: 5
Enter value for branch: cse
Enter value for sec: b
old 1: insert into course values(&stuid,'&branch','&sec')
new 1: insert into course values(5,'cse','b')
insert into course values(5,'cse','b')
*
ERROR at line 1:
ORA-02291: integrity constraint (SCOTT.SID_FK) violated - parent key not found

SQL> delete from adm where stuid=1;
*
ERROR at line 1:
ORA-02292: integrity constraint (SCOTT.SID_FK) violated - child record found

SQL> delete from course where stuid=1;

1 row deleted.

SQL> delete from adm where stuid=1;

1 row deleted.

SQL>select * from adm;

```
STUID SNAME     PER
----- -------------- - ----------
    2      rohit  89
    3      sachin          99
    4      naveen          70
```

CREATING AND DROPING OF VIEWS

SQL> select * from emp2;

OUTPUT:-

```
ENAME              STREET             CITY
------------------- ------------------- --------------------
coyote              toon               hollywood
rabbit             tunnel             carrot ville
smith              revolver            death valley
williams            sea view           sea attle
```

SQL> create view emp22 as select * from emp2;

OUTPUT:-

View created.

SQL> select * from emp22;

OUTPUT:-

```
ENAME              STREET             CITY
------------------- ------------------- --------------------
coyote              toon               hollywood
rabbit             tunnel             carrot ville
smith              revolver            death valley
williams            sea view           sea attle
```

SQL> update emp22 set city='hyd' where ename='coyote';

OUTPUT:-

1 row updated.

SQL> select * from emp2;

OUTPUT:-

ENAME	STREET	CITY
coyote	toon	hyd
rabbit	tunnel	carrot ville
smith	revolver	death valley
williams	sea view	sea attle

SQL> select * from emp22;

OUTPUT:-

ENAME	STREET	CITY
coyote	toon	hyd
rabbit	tunnel	carrot ville
smith	revolver	death valley
williams	sea view	sea attle

SQL> drop table emp2;

OUTPUT:-
Table dropped.

SQL> select * from emp22;

OUTPUT:-

```
select * from emp22
          *
ERROR at line 1:
ORA-04063: view "SCOTT.EMP22" has errors
```

DCL AND TCL COMMANDS

CREATING A USER

SQL>CONNECT SYSTEM/MANAGER;

SQL>CREATE USER "USERNAME" IDENTIFIED BY "PASSWORD"

SQL>GRANT DBA TO "USERNAME"

SQL>CONNECT "USERNAME"/"PASSWORD";

EXAMPLE

CREATING A USER

SQL>CONNECT SYSTEM/MANAGER;

SQL>CREATE USER CSE2 IDENTIFIED BY CSECSE;

SQL>GRANT DBA TO CSE2;

SQL>CONNECT CSE2/CSECSE;

SQL>REVOKE DBA FROM CSE2;

IMPLEMENTING COMMINT, SAVEPOINT, ROLLBACK

SQL> select * from emp1;

OUTPUT:-

EID	ENAME	ESAL	EBRANCH
1	scotty	6000	a
2	wayne	10000	b
3	david	7000	c
4	max	8000	b
5	mark	7000	a
6	jeff	12000	a
7	matt	10000	b
8	steve	7000	c

8 rows selected.

SQL> savepoint s1;

OUTPUT:-
Savepoint created.

SQL> update emp1 set esal=6500 where ename='scotty';

OUTPUT:-
1 row updated.

SQL> savepoint s2;

OUTPUT:-
Savepoint created.

SQL> delete from emp1 where ename='wayne';

OUTPUT:-
1 row deleted.

SQL> savepoint s3;

OUTPUT:-
Savepoint created.

SQL> insert into emp1 values(9,'bruce',11000,'c');

OUTPUT:-
1 row created.

SQL> select * from emp1;

OUTPUT:-

EID	ENAME	ESAL	EBRANCH
1	scotty	6500	a
3	david	7000	c
4	max	8000	b
5	mark	7000	a
6	jeff	12000	a
7	matt	10000	b
8	steve	7000	c

```
      9    bruce          11000          c

8 rows selected.
```

SQL> rollback to s3;

OUTPUT:-
Rollback complete.

SQL> select * from emp1;

OUTPUT:-

```
 EID    ENAME          ESAL         EBRANCH
----------    ----------     ----------   ----------------
       1       scotty         6500              a
       3       david          7000              c
       4       max            8000              b
       5       mark           7000              a
       6       jeff          12000          a
       7       matt          10000          b
       8      steve           7000          c
```

SQL> rollback to s2;

OUTPUT:-
Rollback complete.

SQL> select * from emp1;

OUTPUT:-

```
  EID    ENAME          ESAL     EBRANCH
  ---------- ----------      ----------    -----------
       1      scotty           6500  a
       2     wayne            10000  b
       3     david             7000  c
       4     max               8000  b
       5    mark               7000  a
       6    jeff              12000  a
       7    matt              10000  b
       8    steve              7000  c
```

SQL> rollback to s1;

OUTPUT:-
Rollback complete.

SQL> select * from emp1;

OUTPUT:-

EID	ENAME	ESAL	EBRANCH
1	scotty	6000	a
2	wayne	10000	b
3	david	7000	c
4	max	8000	b
5	mark	7000	a
6	jeff	12000	a
7	matt	10000	b
8	steve	7000	c

SQL> savepoint s4;

OUTPUT:-
Savepoint created.

SQL> delete from emp1 where eid=1;

OUTPUT:-
1 row deleted.

SQL> select * from emp1;

OUTPUT:-

EID	ENAME	ESAL	EBRANCH
2	wayne	10000	b
3	david	7000	c
4	max	8000	b

```
      5   mark              7000    a
      6   jeff             12000    a
      7   matt             10000    b
      8   steve             7000    c
7 rows selected.

SQL> commit;

OUTPUT:-
  Commit complete.

SQL> rollback to s4;

OUTPUT:-
*
ERROR at line 1:
ORA-01086 :savepoint  's4'  never established


SQL> select * from emp1;

OUTPUT:-

      EID   ENAME          ESAL     EBRANCH
 ---------- ----------      ----------    ----------
      2     wayne            10000  b
      3     david             7000  c
      4     max               8000  b
      5     mark              7000  a
      6     jeff             12000  a
      7     matt             10000  b
      8     steve             7000  c
7 rows selected.
```

DRL-DATA RETRIEVAL LANGUAGE

IMPLEMENTING SELECT COMMANDS

SQL> select * from emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20

```
     7934 MILLER     CLERK          7782 23-JAN-82       1300            10


14 rows selected.


SQL> select empno,ename,sal from emp;

    EMPNO ENAME          SAL
---------- ---------- ----------
     7369 SMITH           800
     7499 ALLEN          1600
     7521 WARD           1250
     7566 JONES          2975
     7654 MARTIN         1250
     7698 BLAKE          2850
     7782 CLARK          2450
     7788 SCOTT          3000
     7839 KING           5000
     7844 TURNER         1500
     7876 ADAMS          1100

    EMPNO ENAME          SAL
---------- ---------- ----------
     7900 JAMES           950
     7902 FORD           3000
     7934 MILLER         1300

14 rows selected.

SQL> select empno,ename,sal from emp where sal between 2500 and 5000;

    EMPNO ENAME          SAL
---------- --------- ---------
     7566 JONES          2975
     7698 BLAKE          2850
     7788 SCOTT          3000
     7839 KING           5000
     7902 FORD           3000

SQL>select ename,job,sal,deptno from emp where sal not between 1500 and 5000;

ENAME      JOB           SAL     DEPTNO
---------- --------- ---------- ----------
```

```
SMITH      CLERK          800        20
WARD       SALESMAN       1250       30
MARTIN     SALESMAN       1250       30
ADAMS      CLERK          1100       20
JAMES      CLERK          950        30
MILLER     CLERK          1300       10

6 rows selected.
```

```
SQL> select empno,ename,sal from emp where sal in (800,5000);

     EMPNO ENAME           SAL
---------- ---------- ----------
      7369 SMITH            800
      7839 KING            5000
```

```
SQL> select empno,ename,sal from emp where sal not in(800,1250,3000,5000);

     EMPNO ENAME           SAL
---------- ---------- ----------
      7499 ALLEN           1600
      7566 JONES           2975
      7698 BLAKE           2850
      7782 CLARK           2450
      7844 TURNER          1500
      7876 ADAMS           1100
      7900 JAMES           950
      7934 MILLER          1300

8 rows selected.
```

```
SQL> select empno,ename,sal from emp where comm is null;

     EMPNO ENAME           SAL
---------- ---------- ----------
      7369 SMITH            800
      7566 JONES           2975
      7698 BLAKE           2850
      7782 CLARK           2450
      7788 SCOTT           3000
      7839 KING            5000
      7876 ADAMS           1100
```

```
      7900 JAMES           950
      7902 FORD           3000
      7934 MILLER          1300

10 rows selected.
```

SQL> select empno,ename,sal from emp where comm is not null;

```
    EMPNO ENAME           SAL
---------- ---------- ----------
      7499 ALLEN          1600
      7521 WARD           1250
      7654 MARTIN          1250
      7844 TURNER          1500
```

SQL> select empno,ename,job,sal from emp where ename like'S%';

```
    EMPNO ENAME      JOB           SAL
---------- ---------- --------- ----------
      7369 SMITH      CLERK          800
      7788 SCOTT      ANALYST        3000
```

SQL> select empno,ename,job,sal from emp where job not like'S%';

```
    EMPNO ENAME      JOB           SAL
---------- ---------- --------- ----------
      7369 SMITH      CLERK          800
      7566 JONES      MANAGER        2975
      7698 BLAKE      MANAGER        2850
      7782 CLARK      MANAGER        2450
      7788 SCOTT      ANALYST        3000
      7839 KING       PRESIDENT      5000
      7876 ADAMS      CLERK          1100
      7900 JAMES      CLERK          950
      7902 FORD       ANALYST        3000
      7934 MILLER     CLERK          1300

10 rows selected.
```

SQL> select ename,job,sal from emp where sal>2500;

```
ENAME      JOB           SAL
---------- --------- ----------
JONES      MANAGER        2975
BLAKE      MANAGER        2850
SCOTT      ANALYST        3000
KING       PRESIDENT      5000
FORD       ANALYST        3000
```

SQL> select ename,job,sal from emp where sal<2500;

```
ENAME      JOB           SAL
---------- --------- ----------
SMITH      CLERK          800
ALLEN      SALESMAN       1600
WARD       SALESMAN       1250
MARTIN     SALESMAN       1250
CLARK      MANAGER        2450
TURNER     SALESMAN       1500
ADAMS      CLERK          1100
JAMES      CLERK          950
MILLER     CLERK          1300
```

9 rows selected.

SQL> select empno,ename,job,sal from emp order by sal;

```
    EMPNO ENAME      JOB           SAL
---------- ---------- --------- ----------
      7369 SMITH      CLERK          800
      7900 JAMES      CLERK          950
      7876 ADAMS      CLERK          1100
      7521 WARD       SALESMAN       1250
      7654 MARTIN     SALESMAN       1250
      7934 MILLER     CLERK          1300
      7844 TURNER     SALESMAN       1500
      7499 ALLEN      SALESMAN       1600
      7782 CLARK      MANAGER        2450
      7698 BLAKE      MANAGER        2850
      7566 JONES      MANAGER        2975

    EMPNO ENAME      JOB           SAL
---------- ---------- --------- ----------
      7788 SCOTT      ANALYST        3000
      7902 FORD       ANALYST        3000
```

```
     7839 KING       PRESIDENT       5000

14 rows selected.
```

SQL> select empno,ename,job,sal from emp order by sal desc;

```
    EMPNO ENAME      JOB            SAL
---------- ---------- --------- ----------
     7839 KING       PRESIDENT       5000
     7788 SCOTT      ANALYST         3000
     7902 FORD       ANALYST         3000
     7566 JONES      MANAGER         2975
     7698 BLAKE      MANAGER         2850
     7782 CLARK      MANAGER         2450
     7499 ALLEN      SALESMAN        1600
     7844 TURNER     SALESMAN        1500
     7934 MILLER     CLERK           1300
     7521 WARD       SALESMAN        1250
     7654 MARTIN     SALESMAN        1250

    EMPNO ENAME      JOB            SAL
---------- ---------- --------- ----------
     7876 ADAMS      CLERK           1100
     7900 JAMES      CLERK           950
     7369 SMITH      CLERK           800

14 rows selected.
```

SUBQUERIES (APPLYING IN, ALL, ANY, EXISTS, NOT EXISTS,UNION,INTERSECT,MINUS)

SQL>select * from emp;

OUTPUT:-

```
    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
DEPTNO
---------- ---------- --------- --------- --------- --------- --------
---
     7369 SMITH      CLERK           7902 17-DEC-80         800       20

     7499 ALLEN      SALESMAN        7698 20-FEB-81        1600        300
     30

     7521 WARD       SALESMAN        7698 22-FEB-81        1250        500
     30
```

7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

SQL>select * from dept;

OUTPUT:-

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SQL> select * from emp where sal in(3000,5000);

OUTPUT:-

```
    EMPNO ENAME      JOB              MGR HIREDATE           SAL       COMM
   DEPTNO
---------- ---------- --------- ---------- --------- ---------- ----------
--          ------------

     7788 SCOTT      ANALYST         7566 19-APR-87                    3000
20
     7839 KING       PRESIDENT            17-NOV-81                    5000
10
     7902 FORD       ANALYST         7566 03-DEC-81                    3000
20
```

SQL> select empno,ename from emp where sal in(select max(sal) from emp);

OUTPUT:-

```
 EMPNO ENAME
---------- ----------
     7839 KING
```

SQL> select empno,ename from emp where sal in(select max(sal) from emp group by deptno);

OUTPUT:-

```
 EMPNO ENAME
---------- ----------
     7698 BLAKE
     7788 SCOTT
     7902 FORD
     7839 KING
```

SQL> select empno,ename,job,sal from emp where sal>all(select avg(sal) from emp group by deptno);

OUTPUT:-

```
EMPNO ENAME      JOB              SAL
---------- ---------- --------- ----------
     7566 JONES      MANAGER         2975
     7788 SCOTT      ANALYST         3000
     7839 KING       PRESIDENT       5000
     7902 FORD       ANALYST         3000
```

SQL> select empno,ename,job,sal from emp where sal<all(select avg(sal) from emp group by deptno);

OUTPUT:-

EMPNO	ENAME	JOB	SAL
7369	SMITH	CLERK	800
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7844	TURNER	SALESMAN	1500
7876	ADAMS	CLERK	1100
7900	JAMES	CLERK	950
7934	MILLER	CLERK	1300

7 rows selected.

SQL> select ename,job,sal from emp where sal>any(select sal from emp where job='CLERK');

OUTPUT:-

ENAME	JOB	SAL
ALLEN	SALESMAN	1600
WARD	SALESMAN	1250
JONES	MANAGER	2975
MARTIN	SALESMAN	1250
BLAKE	MANAGER	2850
CLARK	MANAGER	2450
SCOTT	ANALYST	3000
KING	PRESIDENT	5000
TURNER	SALESMAN	1500
ADAMS	CLERK	1100
JAMES	CLERK	950

ENAME	JOB	SAL
FORD	ANALYST	3000
MILLER	CLERK	1300

13 rows selected.

SQL> select ename,job,sal from emp where sal<any(select sal from emp where job='CLERK');

OUTPUT:-

ENAME	JOB	SAL

---------- --------- ----------
SMITH CLERK 800
WARD SALESMAN 1250
MARTIN SALESMAN 1250
ADAMS CLERK 1100
JAMES CLERK 950

SQL> select ename,job,sal from emp where sal=any(select sal from emp where job='CLERK');

OUTPUT:-

ENAME JOB SAL
---------- --------- ----------
SMITH CLERK 800
JAMES CLERK 950
ADAMS CLERK 1100
MILLER CLERK 1300

SQL> select deptno,dname from dept d where exists(select * from emp e where d.deptno=e.deptno);

OUTPUT:-

DEPTNO DNAME
---------- --------------
10 ACCOUNTING
20 RESEARCH
30 SALES

SQL> select deptno,dname from dept d where not exists(select * from emp e where d.deptno=e.deptno);

OUTPUT:-

DEPTNO DNAME
---------- --------------
40 OPERATIONS

SQL> create table s1(sid number(5),sname varchar2(10),rating number(5),age number(10));

Table created.

SQL> insert into s1 values(&sid,'&sname',&rating,&age);

```
Enter value for sid: 22
Enter value for sname: dustin
Enter value for rating: 7
Enter value for age: 45
old   1: insert into s1 values(&sid,'&sname',&rating,&age)
new   1: insert into s1 values(22,'dustin',7,45)

1 row created.

SQL> /
Enter value for sid: 31
Enter value for sname: lubber
Enter value for rating: 8
Enter value for age: 56
old   1: insert into s1 values(&sid,'&sname',&rating,&age)
new   1: insert into s1 values(31,'lubber',8,56)

1 row created.

SQL> /
Enter value for sid: 58
Enter value for sname: rusty
Enter value for rating: 10
Enter value for age: 35
old   1: insert into s1 values(&sid,'&sname',&rating,&age)
new   1: insert into s1 values(58,'rusty',10,35)

1 row created.
SQL>  create table s2(sid number(5),sname varchar2(10),rating number(5),age number(10));

Table created.

SQL> insert into s2 values(&sid,'&sname',&rating,&age);
Enter value for sid: 28
Enter value for sname: yuppy
Enter value for rating: 9
Enter value for age: 25
old   1: insert into s2 values(&sid,'&sname',&rating,&age)
new   1: insert into s2 values(28,'yuppy',9,25)

1 row created.

SQL> /
Enter value for sid: 31
Enter value for sname: lubber
```

```
Enter value for rating: 8
Enter value for age: 55
old   1: insert into s2 values(&sid,'&sname',&rating,&age)
new   1: insert into s2 values(31,'lubber',8,55)

1 row created.

SQL> /
Enter value for sid: 44
Enter value for sname: guppy
Enter value for rating: 5
Enter value for age: 35
old   1: insert into s2 values(&sid,'&sname',&rating,&age)
new   1: insert into s2 values(44,'guppy',5,35)

1 row created.

SQL> /
Enter value for sid: 58
Enter value for sname: rusty
Enter value for rating: 10
Enter value for age: 35
old   1: insert into s2 values(&sid,'&sname',&rating,&age)
new   1: insert into s2 values(58,'rusty',10,35)

1 row created.

SQL> select * from s2;

       SID SNAME        RATING       AGE
---------- ---------- ---------- ----------
        28 yuppy             9         25
        31 lubber            8         55
        44 guppy             5         35
        58 rusty            10         35

SQL> select sname from s1 union select sname from s2;

SNAME
----------
dustin
guppy
lubber
rusty
yuppy
```

SQL> select sname from s1 union select sname from s2;

```
SNAME
----------
dustin
lubber
rusty
yuppy
lubber
guppy
rusty

7 rows selected.
```

SQL> select sid from s1 intersect select sid from s2;

```
       SID
----------
        31
        58
```

SQL> select age from s1 minus select age from s2;

```
       AGE
----------
        45
        56
```

FUNCTIONS:

AGGREGATE FUNCTIONS

SQL> SELECT * FROM EMP;

output:-

```
       ENO ENAME      JOB             SAL
---------- ---------- ---------- ----------
      1001 STEVE      SALESMAN        1500
      1002 ADAM       CLERK           1000
      1003 EVE        MANAGER         5220
      1004 JAMES      DIRECTOR        7000
```

SQL> SELECT COUNT(*)FROM EMP;

output:-

```
  COUNT(*)
----------
         4
```

SQL> SELECT SUM(SAL) FROM EMP;

output:-

```
  SUM(SAL)
----------
     14720
```

SQL> SELECT AVG(SAL) FROM EMP;

output:-

```
  AVG(SAL)
----------
      3680
```

SQL> SELECT MAX(SAL) FROM EMP;

output:-

```
  MAX(SAL)
----------
      7000
```

SQL> SELECT MIN(SAL) FROM EMP;

output:-

```
  MIN(SAL)
----------
      1000
```

SQL>select * from emp;

OUTPUT:-

```
     EMPNO ENAME      JOB              SAL     DEPTNO
---------- ---------- --------- ---------- ----------
      7369 SMITH      CLERK             800         20
      7499 ALLEN      SALESMAN         1600         30
      7521 WARD       SALESMAN         1250         30
      7566 JONES      MANAGER          2975         20
      7654 MARTIN     SALESMAN         1250         30
      7698 BLAKE      MANAGER          2850         30
      7782 CLARK      MANAGER          2450         10
      7788 SCOTT      ANALYST          3000         20
      7839 KING       PRESIDENT        5000         10
      7844 TURNER     SALESMAN         1500         30
      7876 ADAMS      CLERK            1100         20
      7900 JAMES      CLERK             950         30
      7902 FORD       ANALYST          3000         20
      7934 MILLER     CLERK            1300         10
```

14 rows selected.

APPLYING BY GROUP BY ---

SQL> select deptno,max(sal) from emp group by deptno;

OUTPUT:-

```
DEPTNO          MAX(SAL)
----------      ----------
       10           5000
```

20 3000
30 2850

SQL> select deptno,min(sal) from emp group by deptno;

OUTPUT:-

DEPTNO	MIN(SAL)
10	1300
20	800
30	950

SQL> select deptno,max(sal) from emp group by deptno having max(sal)<3000;

OUTPUT:-

DEPTNO	MAX(SAL)
30	2850

SQL>select deptno,min(sal) from emp group by deptno having min(sal)>1000;

OUTPUT:-

DEPTNO	MIN(SAL)
10	1300

SQL> select ename,count(*) from emp group by deptno;

OUTPUT:-

DEPTNO	COUNT(*)
10	3
20	5
30	6

3 rows selected.

CONVERSION FUNCTIONS(TO_CHAR)

SQL>select to_char(65,'RN')from dual;

OUTPUT:LXV

SQL>select to_char(65,'rn')from dual;
OUTPUT:lxv

SQL>select to_char(58,'s9999')from dual;
OUTPUT:+58

SQL>select to_char(-100,'s9999')from dual;
OUTPUT:-100

SQL> select to_char(41,'XXXX')from dual;
OUTPUT:29

SQL> select to_char(10,'XXXX')from dual;
OUTPUT:A

SQL> select to_char(sysdate,'day')from dual;
OUTPUT:MONDAY

SQL> SELECT TO_CHAR(SYSDATE,'MONTH')FROM DUAL;
OUTPUT:JANUARY

SQL> SELECT TO_CHAR(SYSDATE,'YEAR')FROM DUAL;
OUTPUT:TWO THOUSAND NINE

SQL> select to_char(123456,'9g99g999')from dual;
OUTPUT:1,23,456

SQL> select to_char(1234,'l9999')from dual;
OUTPUT:$1234

SQL> select to_char(123456,'9g99g999d999')from dual;
OUTPUT:1,23,456.000

SELECT TO_CHAR(2234,'L9999','NLS_CURRENCY=RS')FROM DUAL;
OUTPUT:RS2234

STRING FUNCTIONS

SQL>SELECT CONCAT('ORACLE','CORPORATION')FROM DUAL;

OUTPUT:-ORACLECORPORATION

SQL>SELECT LPAD('ORACLE',15,'*')FROM DUAL;

OUTPUT:-*********ORACLE

SQL>SELECT RPAD('ORACLE',15,'*')FROM DUAL;

OUTPUT:-ORACLE*********

SQL>SELECT LTRIM('SSMITHSS','S')FROM DUAL;

OUTPUT:-MITHSS

SQL>SELECT RTRIM('SSMITHSS','S')FROM DUAL;

OUTPUT:-SSMITH

SQL>SELECT LOWER('DBMS')FROM DUAL;

OUTPUT:-dbms

SQL>SELECT UPPER('dbms')FROM DUAL;

OUTPUT:-DBMS

SQL>SELECT INITCAP('ORACLE','CORPORATION')FROM DUAL;

OUTPUT:-Oracle Corporation

SQL>SELECT LENGTH('DATABASE')FROM DUAL;

OUTPUT:-8

SQL>SELECT SUBSTR('ABCDEFGHIJ'3,4)FROM DUAL;

OUTPUT:-CDEF

SQL>SELECT INSTR('CORPORATE FLOOR','OR',3,2)FROM DUAL;

OUTPUT:-14

DATE FUNCTIONS

SQL>SELECT SYSDATE FROM DUAL;

OUTPUT:-29-DEC-08

SQL>SELECT NEXT_DAY(SYSDATE,’WED’)FROM DUAL;

OUTPUT:-05-JAN-09

SQL>SELECT ADD_MONTHS(SYSDATE,2)FROM DUAL;

OUTPUT:-28-FEB-09

SQL>SELECT LAST_DAY(SYSDATE)FROM DUAL;

OUTPUT:-31-DEC-08

SQL>SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE)FROM EMP;

OUTPUT:-

SQL>SELECT LEAST('10-JAN-07','12-OCT-07')FROM DUAL;

OUTPUT:-10-JAN-07

SQL>SELECT GREATEST('10-JAN-07','12-OCT-07')FROM DUAL;

OUTPUT:-10-JAN-07

SQL>SELECT TRUNC(SYSDATE,'DAY')FROM DUAL;

OUTPUT:-28-DEC-08

SQL>SELECT TRUNC(SYSDATE,'MONTH')FROM DUAL;

OUTPUT:-01-DEC-08
SQL>SELECT TRUNC(SYSDATE,'YEAR')FROM DUAL;

OUTPUT:-01-JAN-08

SQL>SELECT ROUND(SYSDATE,'DAY')FROM DUAL;

OUTPUT:-28-DEC-08

SQL>SELECT ROUND(SYSDATE,'MONTH')FROM DUAL;

OUTPUT:-01-JAN-09

SQL>SELECT ROUND(SYSDATE,'YEAR')FROM DUAL;

OUTPUT:-01-JAN-09

NUMBER FUNCTIONS

SQL> select round(12.36), round(14.63) from dual;

OUTPUT:-

```
ROUND(12.36)     ROUND(14.63)
------------     ------------
        12               15
```

SQL> select floor(12.87), floor(11.23) from dual;

OUTPUT:-

```
FLOOR(12.87)       FLOOR(11.23)
  ------------       ------------
        12                 11
```

SQL> select ceil(16.23), ceil(12.78) from dual;

OUTPUT:-

```
CEIL(16.23)       CEIL(12.78)
-----------       -----------
       17                13
```

SQL> select trunc(56.63) from dual;

OUTPUT:-

```
TRUNC(56.63)
------------
          56
```

SQL> select mod(11,4) from dual;

OUTPUT:-

```
 MOD(11,4)
----------
         3
```

SQL> select power(2,3) from dual;

OUTPUT:-

```
POWER(2,3)
----------
         8
```

SQL> select sign(0),sign(34),sign(-56) from dual;

OUTPUT:-

```
  SIGN(0)   SIGN(34)      SIGN(-56)
 ----------  ----------    ----------
         0           1             -1
```

SQL> select abs(12),abs(-89) from dual;

OUTPUT:-

```
   ABS(12)    ABS(-89)
  ---------  ---------
        12          89
```

SQL> select sqrt(25) from dual;

OUTPUT:-

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

INTRODUCTION TO PL/SQL

PL/SQL stands for PROCEDURAL Language Extensions to SQL.
PL/SQL extends SQL by adding programming structures and subroutines available in any high level language.
PL/SQL can be used for both server-side and Client side Development.
PL/SQL has syntax and rules that determine how programming statements work together.
PL/SQL is not a stand alone Programming Language.
PL/SQL is a part of the ORACLE RDBMS and hence can reside in two environments,the CLIENT and the SERVER.
Any MODULE that is developed using PL/SQL can be moved easily between SERVER SIDE and CLIENT SIDE applications.
Either in CLIENT/SERVER environments any PL/SQL Block or the PL/SQL Engine processes Subroutine.
PL/SQL Engine is a special component that processes and executes any PL/SQL statements and sends any SQL statement to the SQL statement processor.
The SQL statement processes are always located on the ORACLE SERVER.

As per the necessity the PL/SQL Engine can be located either at

SERVER
CLIENT

When PL/SQL Engine is locted upon the SERVER,the whole PL/SQL block is passed to the PL/SQL Engine on the ORACLE SERVER.
When the PL/SQL Engine is located upon the CLIENT,the PL/SQL processing is done on the CLIENT SIDE.All SQL statenents that are embedded within the PL/SQL block,are sent to the ORACLE SERVER for further processing.

If the PL/SQL block does not contain any SQL statements,the entire block is executed on the CLIENT SIDE.

PL/SQL BLOCK

```
DECLARE
--Declarations of memory variables,constants,cursors etc.,in PL/SQL
BEGIN
--SQL executable statements
--PL/SQL executable statements
EXCEPTION
/*SQL or PL/SQL code to handle errors that may arise during the execution of the code block between BEGIN and EXCEPTION section
END;
```

SYNTAX's of CONTROL STATEMENTS in PL/SQL

1. BRANCHING
2. SELECTION
3. LOOPING

BRANCHING STATEMENTS

1.Simple IF
2.ELSIF
3.ELSE IF

SIMPLE IF

```
IF condition THEN
   statement1;
   statement2;
END IF;
```

IF-THEN-ELSE STATEMENT

```
IF condition THEN
   statement1;
ELSE
   statement2;
END IF;
```

ELSIF STATEMENTS

```
IF condition1 THEN
  statement1;
ELSIF condition2 THEN
  statement2;
ELSIF condition3 THEN
  statement3;
ELSE
```

```
  statementn;
END IF;
```

NESTED IF

```
IF condition THEN
   statement1;
ELSE
  IF condition THEN
     statement2;
  ELSE
     statement3;
  END IF;
END IF;
ELSE
  statement3;
END IF;
```

SELECTION IN PL/SQL

SIMPLE CASE

```
CASE SELECTOR
    WHEN Expr1 THEN statement1;
    WHEN Expr2 THEN statement2;
          :
          :
          :
ELSE
     statementn;

END CASE;
```

SEARCHED CASE

```
CASE
  WHEN  searchcondition1 THEN statement1;
  WHEN  searchcondition2 THEN statement2;
        :
        :
        :
ELSE
        statementn;
END CASE;
```

ITERATIONS IN PL/SQL

SIMPLE LOOP

```
LOOP
  statement1;
EXIT [ WHEN Condition];
END LOOP;
```

WHILE LOOP

```
WHILE condition LOOP
        statement1;
        statement2;
END LOOP;
```

FOR LOOP

```
FOR counter IN [REVERSE]
   LowerBound..UpperBound
LOOP
        statement1;
        statement2;
END LOOP;
```

WRITE A PL/SQL PROGRAM TO SWAP TWO NUMBERS WITH OUT TAKING THIRD VARIABLE

```
declare
a number(10);
b number(10);
begin
a:=&a;
b:=&b;
dbms_output.put_line('THE PREV VALUES OF A AND B WERE');
dbms_output.put_line(a);
dbms_output.put_line(b);
a:=a+b;
b:=a-b;
a:=a-b;
dbms_output.put_line('THE VALUES OF A AND B ARE');
dbms_output.put_line(a);
dbms_output.put_line(b);
end;
```

OUTPUT:

```
SQL> @ SWAPPING.SQL
 17  /
Enter value for a: 5
old   5: a:=&a;
new   5: a:=5;
Enter value for b: 3
old   6: b:=&b;
new   6: b:=3;
THE PREV VALUES OF A AND B WERE
5
3
THE VALUES OF A AND B ARE
3
5

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO SWAP TWO NUMBERS BY TAKING THIRD VARIABLE

```
declare
a number(10);
b number(10);
c number(10);
begin
dbms_output.put_line('THE PREV VALUES OF A AND B WERE');
dbms_output.put_line(a);
dbms_output.put_line(b);
a:=&a;
b:=&b;
c:=a;
a:=b;
b:=c;
dbms_output.put_line('THE VALUES OF A AND B ARE');
dbms_output.put_line(a);
dbms_output.put_line(b);
end;
```

OUTPUT:

```
SQL> @ SWAPPING2.SQL
 19  /
Enter value for a: 5
old   6: a:=&a;
new   6: a:=5;
Enter value for b: 3
old   7: b:=&b;
new   7: b:=3;
THE PREV VALUES OF A AND B WERE
5
3
THE VALUES OF A AND B ARE
3
5

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO FIND THE LARGEST OF TWO NUMBERS

```
declare
a number;
b number;
begin
a:=&a;
b:=&b;
if a=b then
dbms_output.put_line('BOTH ARE EQUAL');
elsif   a>b   then
dbms_output.put_line('A IS GREATER');
else
dbms_output.put_line('B IS GREATER');
end if;
end;
```

OUTPUT:

```
SQL> @ GREATESTOF2.sql
 13  /
Enter value for a: 5
old   5: a:=&a;
new   5: a:=5;
Enter value for b: 2
old   6: b:=&b;
new   6: b:=2;
A IS GREATER

PL/SQL procedure successfully completed.
```


WRITE A PL/SQL PROGRAM TO FIND THE LARGEST OF THREE NUMBERS

```
declare
a number;
b number;
c number;
begin
a:=&a;
b:=&b;
c:=&c;
if a=b and b=c and c=a then
dbms_output.put_line('ALL ARE EQUAL');
elsif a>b and a>c then
dbms_output.put_line('A IS GREATER');
elsif b>c then
dbms_output.put_line('B IS GREATER');
else
dbms_output.put_line('C IS GREATER');
end if;
end;
```

OUTPUT:

```
SQL> @ GREATESTOF3.sql
 17  /
Enter value for a: 8
old   6: a:=&a;
new   6: a:=8;
Enter value for b: 9
old   7: b:=&b;
new   7: b:=9;
Enter value for c: 7
old   8: c:=&c;
new   8: c:=7;
B IS GREATER

PL/SQL procedure successfully completed.
```


WRITE A PL/SQL PROGRAM TO FIND THE TOTAL AND AVERAGE OF 6 SUBJECTS AND DISPLAY THE GRADE

```
declare
java number(10);
dbms number(10);
co number(10);
se number(10);
es number(10);
ppl number(10);
total number(10);
avgs number(10);
per number(10);
begin
dbms_output.put_line('ENTER THE MARKS');
java:=&java;
dbms:=&dbms;
co:=&co;
se:=&se;
es:=&es;
ppl:=&ppl;
total:=(java+dbms+co+se+es+ppl);
per:=(total/600)*100;
if java<40 or dbms<40 or co<40 or se<40 or es<40 or ppl<40 then
dbms_output.put_line('FAIL');
if per>75 then
dbms_output.put_line('GRADE A');
elsif per>65 and per<75 then
dbms_output.put_line('GRADE B');
elsif per>55 and per<65 then
dbms_output.put_line('GRADE C');
else
dbms_output.put_line('INVALID INPUT');
end if;
dbms_output.put_line('PERCENTAGE IS '||per);
dbms_output.put_line('TOTAL IS '||total);
end;
```

OUTPUT:

```
SQL> @ GRADE.sql
 31  /
Enter value for java: 80
old  12: java:=&java;
new  12: java:=80;
Enter value for dbms: 70
```

```
old  13: dbms:=&dbms;
new  13: dbms:=70;
Enter value for co: 89
old  14: co:=&co;
new  14: co:=89;
Enter value for se: 72
old  15: se:=&se;
new  15: se:=72;
Enter value for es: 76
old  16: es:=&es;
new  16: es:=76;
Enter value for ppl: 71
old  17: ppl:=&ppl;
new  17: ppl:=71;
GRADE A
PERCENTAGE IS 76
TOTAL IS 458
PL/SQL procedure successfully completed.
```


WRITE A PL/SQL PROGRAM TO CHECK WHETHER THE GIVEN NUMBER IS AN ARMSTRONG NUMBER OR NOT

```
declare
a number;
t number;
arm number;
d number;
begin
a:=&a;
t:=a;
arm:=0;
while t>0
loop
d:=mod(t,10);
arm:=arm+power(d,3);
t:=trunc(t/10);
end loop;
if arm=a then
dbms_output.put_line('given no is an armstrong no'|| a);
else
dbms_output.put_line('given no is not an armstrong no');
end if;
end;
```

OUTPUT:

```
SQL> @ ARMSTRONGNUM.sql
Enter value for a: 407
old   7: a:=&a;
new   7: a:=407;
given no is an armstrong no407

PL/SQL procedure successfully completed.

SQL> /
Enter value for a: 406
old   7: a:=&a;
new   7: a:=406;
given no is not an armstrong no

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO FIND THE SUM OF DIGITS IN A GIVEN NUMBER

```
declare
```

```
a number;
d number:=0;
sum1 number:=0;
begin
a:=&a;
while a>0
loop
d:=mod(a,10);
sum1:=sum1+d;
a:=trunc(a/10);
end loop;
dbms_output.put_line('sum is'|| sum1);
end;
```

OUTPUT:

```
SQL> @ SUMOFDIGITS.sql
 16  /
Enter value for a: 564
old   7: a:=&a;
new   7: a:=564;
sum is15

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO DISPLAY THE NUMBER IN REVERSE ORDER

```
declare
a number;
rev number;
d number;
begin
a:=&a;
rev:=0;
while a>0
loop
d:=mod(a,10);
rev:=(rev*10)+d;
a:=trunc(a/10);
end loop;
dbms_output.put_line('no is'|| rev);
end;
```

OUTPUT:

```
SQL> @ REVERSE2.sql
 16  /
Enter value for a: 536
old   6: a:=&a;
new   6: a:=536;
no is635

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO DISPLAY NUMBER IN REVERSE ORDER USING STRING FUNCTION

```
declare
gn varchar2(5):=4567;
sl number(2);
rv varchar2(5);
begin
sl:=length(gn);
for i in reverse 1..sl
loop
rv:=rv||substr(gn,i,1);
end loop;
dbms_output.put_line('given no r is'||gn);
dbms_output.put_line('given no in reverse order is'||rv);
end;
```

OUTPUT:

```
SQL> @ REVERSE.sql
 14  /
given no r is4567
given no in reverse order is7654

PL/SQL procedure successfully completed.
```


WRITE A PL/SQL PROGRAM TO CHECK WHETHER THE GIVEN NUMBER IS PRIME OR NOT

```
declare
```

```
a number;
c number:=0;
i number;
begin
a:=&a;
for i in 1..a
loop
if mod(a,i)=0 then
c:=c+1;
end if;
end loop;
if c=2 then
dbms_output.put_line(a ||'is a prime number');
else
dbms_output.put_line(a ||'is not a prime number');
end if;
end;
```

OUTPUT:

```
SQL> @ PRIME.SQL
 19  /
Enter value for a: 11
old   6: a:=&a;
new   6: a:=11;
11is a prime number

PL/SQL procedure successfully completed.
```


WRITE A PL/SQL PROGRAM TO FIND THE FACTORIAL OF A GIVEN NUMBER

```
declare
n number;
```

```
f number:=1;
begin
n:=&n;
for i in 1..n
loop
f:=f*i;
end loop;
dbms_output.put_line('the factorial is'|| f);
end;
```

OUTPUT:

```
SQL> @ FACTORIAL.sql
 12  /
Enter value for n: 5
old   5: n:=&n;
new   5: n:=5;
the factorial is120

PL/SQL procedure successfully completed.
```


WRITE A PL/SQL PROGRAM TO GENERATE FIBONACCI SERIES

```
declare
a number;
```

```
b number;
c number;
n number;
i number;
begin
n:=&n;
a:=0;
b:=1;
dbms_output.put_line(a);
dbms_output.put_line(b);
for i in 1..n-2
loop
c:=a+b;
dbms_output.put_line(c);
a:=b;
b:=c;
end loop;
end;
```

OUTPUT:

```
SQL> @ FIBONACCI.sql
 21  /
Enter value for n: 5
old   8: n:=&n;
new   8: n:=5;
0
1
1
2
3

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL CODE BLOCK TO CALCULATE THE AREA OF A CIRCLE FOR A VALUE OF RADIUS VARYING FROM 3 TO 7.
STORE THE RADIUS AND THE CORRESPONDING VALUES OF CALCULATED AREA IN AN EMPTY TABLE NAMED AREAS ,CONSISTING OF TWO COLUMNS RADIUS & AREA

TABLE NAME:AREAS

RADIUS AREA

```
SQL> create table areas(radius number(10),area number(6,2));

Table created.
--PROGRAM
declare
pi constant number(4,2):=3.14;
radius number(5):=3;
area number(6,2);
begin
while radius<7
loop
area:=pi*power(radius,2);
insert into areas values(radius,area);
radius:=radius+1;
end loop;
end;
```

OUTPUT:

```
SQL> @ AREAOFCIRCLE.SQL
 13  /
PL/SQL procedure successfully completed.
SQL> SELECT * FROM AREAS;
   RADIUS      AREA
    ----------  ----------
       3         28.26
       4         50.24
       5          78.5
       6        113.04
```

WRITE A PL/SQL CODE BLOCK THAT WILL ACCEPT AN ACCOUNT NUMBER FROM THE USER,CHECK IF THE USERS BALANCE IS LESS THAN MINIMUM BALANCE,ONLY THEN DEDUCT RS.100/- FROM THE BALANCE.THIS PROCESS IS FIRED ON THE ACCT TABLE.

SQL> create table acct(name varchar2(10),cur_bal number(10),acctno number(6,2));

SQL> insert into stud values('&sname',&rollno,&marks);

SQL> select * from acct;

ACCTNO	NAME	CUR_BAL
777	sirius	10000
765	john	1000
855	sam	500
353	peter	800

--PROGRAM

```
declare
mano number(5);
mcb number(6,2);
minibal constant number(7,2):=1000.00;
fine number(6,2):=100.00;
begin
mano:=&mano;
select cur_bal into mcb from acct where acctno=mano;
if mcb<minibal then
update acct set cur_bal=cur_bal-fine where acctno=mano;
end if;
end;
```

OUTPUT:

```
SQL> @ BANKACC.sql
 13  /
Enter value for mano: 855
old   7: mano:=&mano;
new   7: mano:=855;

PL/SQL procedure successfully completed.
```

SQL> select * from acct;

ACCTNO	NAME	CUR_BAL
777	sirius	10000

765	john	1000
855	sam	400
353	peter	800

PL/SQL PROGRAMS- FUNCTIONS

CUBE

```
create or replace function cube(n number) return number
as
```

```
c number;
begin
c:=n*n*n;
return c;
end;
```

output:-

```
SQL> @cub
  /

Function created.

SQL> select cube(6)from dual;

   CUBE(6)
   ------
     216
```


FACTORIAL

```
create or replace function fact(n number)return number
as

fac number:=1;
begin
```

```
for i in 1..n
loop
fac:=fac*i;
end loop;
return fac;
end;
```

output:-

```
SQL> @func.sql;
 12  /

Function created.

SQL> select fact(5) from dual;

  FACT(5)
----------
       120
```


IN- PROCEDURE

```
create or replace procedure inpro(dno number,depname  varchar2,city  varchar2)
as
begin
insert into dept (deptno,dname,loc)values(dno,depname,city);
end;
```

OUTPUT:-

```
SQL> select * from dept;

    DEPTNO DNAME         LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON


SQL> @pro1
   /

Procedure created.


SQL> exec inpro(50,'MARKETING','HYDERABAD');

PL/SQL procedure successfully completed.

SQL> select * from dept;

    DEPTNO DNAME         LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 MARKETING      HYDERABAD
```

OUT-PROCEDURE

```
create or replace procedure outpro(dno number,depname out varchar2,city out varchar2)
as
begin
select dname,loc into depname,city from dept where deptno=dno;
end;
```

OUTPUT:

```
SQL> @proc2
  /

Procedure created.

SQL> declare
  2  x varchar2(20);
  3  y varchar2(30);
  4  ch number;
  5  begin
  6  ch:=&ch;
  7  outpro(ch,x,y);
  8   dbms_output.put_line(x || ' ' || y);
  9  end;
 10  /
Enter value for ch: 10
old   6: ch:=&ch;
new   6: ch:=10;
ACCOUNTING NEW YORK

PL/SQL procedure successfully completed.
```


BEFORE TRIGGER

```
create or replace trigger tday before insert or delete or update  on emp
declare
we varchar2(10);
begin
we:=to_char(sysdate,'dy');
if we='sat' or we='sun' then
```

```
raise_application_error(-20015,'its a weekend');
end if;
end;
```

output:-

```
SQL> @trig1
     /

Trigger created.

SQL> delete from emp where empno=7902;

ERROR at line 1:
ORA-20015: its a weekend
ORA-06512: at "SCOTT.TDAY", line 6
ORA-04088: error during execution of trigger 'SCOTT.TDAY'
```


AFTER TRIGGER

```
create or replace trigger ttime after insert or delete or update on emp1 for each row
declare
tt varchar2(5);
begin
tt:=to_char(sysdate,'hh24');
if tt not between 10 and 17  then
```

```
raise_application_error(-20010,'not working hours');
end if;
end;
```

output:-

```
SQL> @trig2;
 10  /

Trigger created.

SQL> update emp1 set empno=7777 where empno=7902;

       *
ERROR at line 1:
ORA-20010: not working hours
ORA-06512: at "SCOTT.TTIME", line 6
ORA-04088: error during execution of trigger 'SCOTT.TTIME'
```