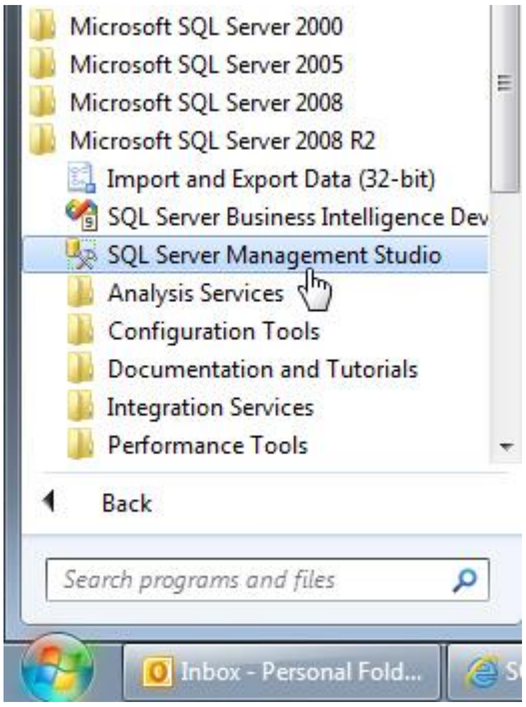
SR.	PRACTICALS
1	1. WHAT IS SQL? SQL(Structured Query Language) is a standard language for accessing and manipulating databases. SQL is an ANSI and ISO standard & not case sensitive. 2. INTRODUCTION TO EDITOR (SQL SERVER MANAGEMENT STUDIO). SQL Server Management Studio is used to create, delete and maintain databases, write and execute queries and stored procedures, and a host of other tasks.

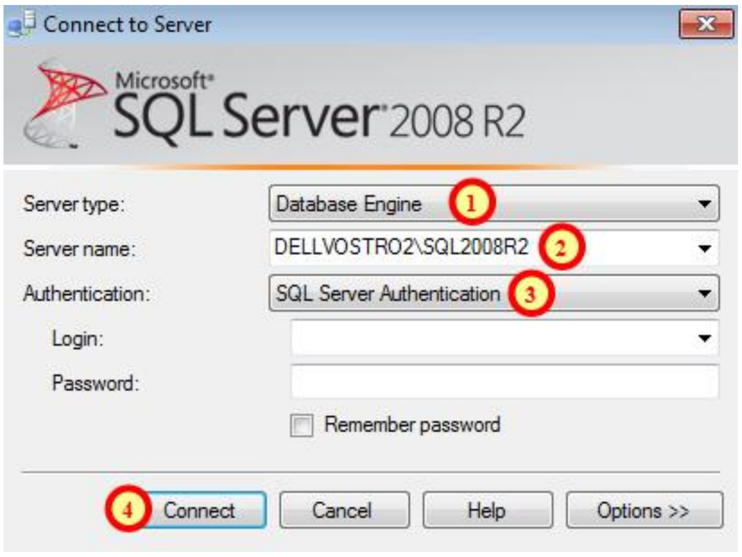
Opening Management Studio

After Installing Microsoft SQL Server on your computer you will have access to the Management Studio application from the Start menu.


Microsoft SQL Server 2000
Microsoft SQL Server 2005
Microsoft SQL Server 2008
Microsoft SQL Server 2008 R2
Import and Export Data (32-bit)
SQL Server Business Intelligence Dev
SQL Server Management Studio
Analysis Services
Configuration Tools
Documentation and Tutorials
Integration Services
Performance Tools
Back
Search programs and files
Inbox - Personal Fold...

Connecting to a Server

When you choose to open Management Studio, before you can start working with the application, you are asked to connect to a server. You don't have to do this at this stage but it makes sense to do so and the diagram below explains how:



Server type: Database Engine

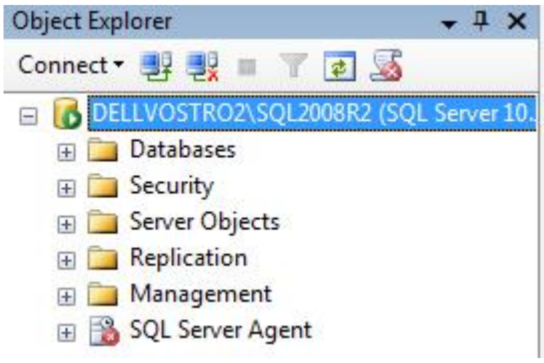
Server name: DELLVOSTRO2\SQL2008R2

Authentication: SQL Server Authentication

This dialog box allows you to connect to a server. The options shown are described below.

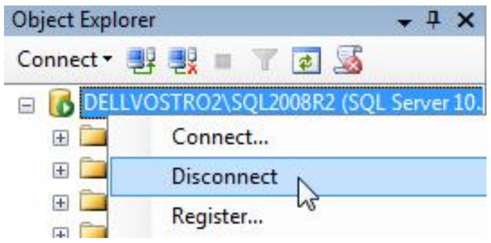
i. Choose the type of server to which you want to connect.
ii. Type in or select the name of the server from this drop down list.
iii. Choose the type of authentication you would like to use. You may be able to use your Windows credentials by selecting Windows Authentication from this list. Alternatively, you'll need to select the option shown and type in a login name and password in the appropriate boxes.
iv. Click Connect to connect to the selected server.

Once you have followed the steps above you will see the server you selected appear at the left hand side of the screen.
The server you selected will appear in the **Object Explorer** window at the left of the screen.

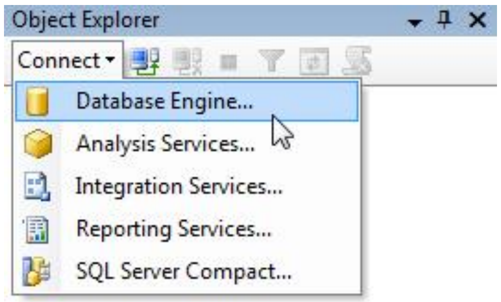


Disconnecting from and Reconnecting to a Server

You can disconnect from a server by right-clicking on the server name and choosing the option shown in the diagram below:

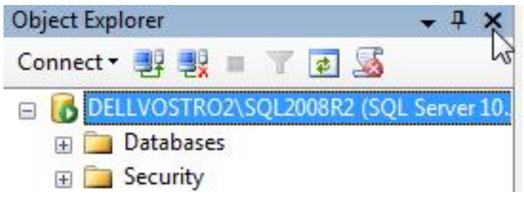


You can **reconnect** to a server using the **Connect** tool at the top of the **Object Explorer** window, as shown below:

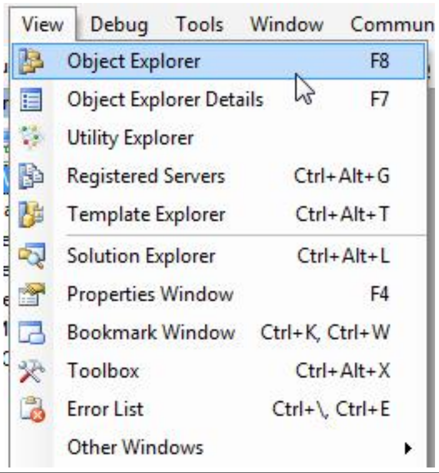


Choose the type of server to which you want to connect
When you choose an option from the list you'll be presented with the same dialog box you see when you open Management Studio for the first time

Opening and Closing Windows
You can close any windows in Management Studio by clicking the cross in the top right hand corner of the window.



To open a window that is closed you can use the View menu at the top of the screen.


View
Debug
Tools
Window
Commun
Object Explorer F8
Object Explorer Details F7
Utility Explorer
Registered Servers Ctrl+Alt+G
Template Explorer Ctrl+Alt+T
Solution Explorer Ctrl+Alt+L
Properties Window F4
Bookmark Window Ctrl+K, Ctrl+W
Toolbox Ctrl+Alt+X
Error List Ctrl+\, Ctrl+E
Other Windows

3. COMPONENTS OF SQL
   DDL, DML, DCL, DQL, TCL

I. **DDL:** DDL is stands for**Data Definition Language**, which deals with database schemas and descriptions, of how the data should reside in the database. DDL statements create, modify, and remove database objects such as tables, indexes, and users.

Examples of DDL commands:
CREATE – Create is used to create the database or its objects
DROP – Drop is used to delete objects from the database.
ALTER– Alter is used to alter the structure of the database.
TRUNCATE– Truncate is used to remove all records from a table but leaves its structure for future data.
COMMENT – Comment is used to add comments to the data dictionary.
RENAME – Rename is used to rename an object existing in the database.

II. **DML:** DML is stands for**Data Manipulation Language** which deals with data manipulation and it is used to store, modify, retrieve, delete and update data in a database.

Examples of DML:
SELECT – Select is used to retrieve data from the database.
INSERT – Insert is used to insert data into a table.
UPDATE – Update is used to update existing data within a table.
DELETE – Delete is used to delete records from a database table.

III. **DCL:** DCL is stands for**Data Control Language** which is mostly concerned with rights, permissions and other controls of the database system.

Examples of DCL commands:
GRANT-gives user's access privileges to database.
REVOKE-withdraw user's access privileges given by using the GRANT command.

IV. **DQL:** DQL is stands for**Data Query Language** which is are used to retrieve data from the database.

Examples of DQL commands:
SELECT – Select is used to retrieve data from the database table.

V. **TCL:** TCL is stands for**Transaction Control Language** which deals with a transaction within a database.

Examples of TCL commands:
COMMIT– commits a Transaction.
ROLLBACK– rollbacks a transaction in case of any error occurs.
SAVEPOINT–sets a savepoint within a transaction.
SET TRANSACTION–specify characteristics for the transaction.

2

1. INTRODUCTION TO DATABASE, TABLE, COLUMN, FIELD, ATTRIBUTE, ROW, RECORD, TUPLE.

I. **Database**
A database is a data structure that stores organized information.
DB is a collection of information organized in such a way that a computer program can quickly select desired pieces of data.

II. **Table**
A table is a collection of related data entries and it consists of columns and rows.

III. **Column/Field/Attribute**
A field is a character or group of characters that have specific meaning.
It is also called as Data Item.
Example: Empid, Name, Salary are fields of Employee table.

IV. **Row/Record/Tuple**
A record is a collection of logically related fields.
Example: Collection of fields (Empid, Name and Salary) forms a record for Employee table.

2. INTRODUCTION TO VARIOUS DATA TYPES
INT, CHAR, VARCHAR, DATETIME, BIT, DECIMAL

Data type	**Description**	**Storage**
Int	Allows **whole numbers** between -2,147,483,648 and 2,147,483,647	4 bytes
Char(size)	Holds a **fixed length** string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters.	Defined width
Varchar(size)	Holds a **variable length** string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. NOTE: If you put a greater value than 255 it will be converted to a TEXT type.	2 bytes + number of chars
Datetime	A **date and time** combination. From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	8 bytes
Bit	Integer that can be **0, 1, or NULL**	
Decimal(p,s)	**Fixed precision and scale numbers**. Allows numbers from -10^38 +1 to 10^38 –1. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0.	5-17 bytes

3 **CREATE FOLLOWING TABLES USING QUERY ACCORDING TO THE DEFINITION.**

```
CREATE TABLE DEPOSIT
(
      ACTNO INT,
      CNAME VARCHAR(50),
      BNAME VARCHAR(50),
      AMOUNT DECIMAL(8,2),
      ADATE DATETIME
);

CREATE TABLE BRANCH
(
      BNAME VARCHAR(50),
      CITY VARCHAR(50)
);

CREATE TABLE CUSTOMERS
(
      CNAME VARCHAR(50),
      CITY VARCHAR(50)
);

CREATE TABLE BORROW
(
      LOANNO INT,
      CNAME VARCHAR(50),
      BNAME VARCHAR(50),
      AMOUNT DECIMAL(8,2)
);
```

INSERT THE DATA INTO TABLES USING QUERY AS SHOWN BELOW.

DEPOSIT

```
INSERT INTO DEPOSIT VALUES(101,'ANIL','VRCE',1000,'1/MAR/95')
INSERT INTO DEPOSIT VALUES(102,'SUNIL','AJNI',5000,'4/JAN/96')
INSERT INTO DEPOSIT VALUES(103,'MEHUL','KAROLBAGH',3500,'17/NOV/95')
INSERT INTO DEPOSIT VALUES(104,'MADHURI','CHANDI',1200,'17/DEC/95')
INSERT INTO DEPOSIT VALUES(105,'PRMOD','M.G.ROAD',3000,'27/MAR/96')
INSERT INTO DEPOSIT VALUES(106,'SANDIP','ANDHERI',2000,'31/MAR/96')
INSERT INTO DEPOSIT VALUES(107,'SHIVANI','VIRAR',1000,'5/SEP/95')
INSERT INTO DEPOSIT VALUES(108,'KRANTI','NEHRU PLACE',5000,'2/JUL/95')
INSERT INTO DEPOSIT VALUES(109,'MINU','POWAI',7000,'10/AUG/95')
```

BRANCH

```
INSERT INTO BRANCH VALUES('VRCE','NAGPUR')
INSERT INTO BRANCH VALUES('AJNI','NAGPUR')
INSERT INTO BRANCH VALUES('KAROLBAGH','DELHI')
INSERT INTO BRANCH VALUES('CHANDI','DELHI')
INSERT INTO BRANCH VALUES('DHARAMPETH','NAGPUR')
INSERT INTO BRANCH VALUES('M.G.ROAD','BANGLORE')
INSERT INTO BRANCH VALUES('ANDHERI','BOMBAY')
INSERT INTO BRANCH VALUES('VIRAR','BOMBAY')
INSERT INTO BRANCH VALUES('NEHRU PLACE','DELHI')
INSERT INTO BRANCH VALUES('POWAI','BOMBAY')
```

CUSTOMERS

```
INSERT INTO CUSTOMERS VALUES('ANIL','CULCUTTA')
INSERT INTO CUSTOMERS VALUES('SUNIL','DELHI')
INSERT INTO CUSTOMERS VALUES('MEHUL','BARODA')
INSERT INTO CUSTOMERS VALUES('MANDAR','PATNA')
INSERT INTO CUSTOMERS VALUES('MADHURI','NAGPUR')
INSERT INTO CUSTOMERS VALUES('PRMOD','NAGPUR')
INSERT INTO CUSTOMERS VALUES('SANDIP','SURAT')
INSERT INTO CUSTOMERS VALUES('SHIVANI','BOMBAY')
INSERT INTO CUSTOMERS VALUES('KRANTI','BOMBAY')
INSERT INTO CUSTOMERS VALUES('NAREN','BOMBAY')
```

BORROW

```
INSERT INTO BORROW VALUES(201,'ANIL','VRCE',1000)
INSERT INTO BORROW VALUES(206,'MEHUL','AJNI',5000)
INSERT INTO BORROW VALUES(311,'SUNIL','DHARAMPETH',3000)
INSERT INTO BORROW VALUES(321,'MADHURI','ANDHERI',2000)
INSERT INTO BORROW VALUES(375,'PRMOD','VIRAR',8000)
INSERT INTO BORROW VALUES(481,'KRANTI','NEHRU PLACE',3000)
```

FROM THE ABOVE GIVEN TABLES PERFORM THE FOLLOWING QUERIES (SELECT OPERATION):

1. RETRIEVE ALL DATA FROM TABLE DEPOSIT.
```
SELECT*FROM DEPOSIT
```
2. RETRIEVE ALL DATA FROM TABLE BORROW.
```
SELECT*FROM BORROW
```
3. RETRIEVE ALL DATA FROM TABLE CUSTOMERS.
```
SELECT*FROM CUSTOMERS
```
4. DISPLAY ACCOUNT NO, CUSTOMER NAME & AMOUNT FROM DEPOSIT.
```
SELECT ACTNO,CNAME,AMOUNT FROM DEPOSIT
```
5. DISPLAY LOAN NO, AMOUNT FROM BORROW.
```
SELECT LOANNO,AMOUNT FROM BORROW
```
6. DISPLAY LOAN DETAILS OF ALL CUSTOMERS WHO BELONGS TO ‘ANDHERI’ BRANCH.
```
SELECT*FROM BORROW WHERE BNAME ='ANDHERI'
```
7. GIVE ACCOUNT NO AND AMOUNT OF DEPOSITOR, WHOSE ACCOUNT NO IS EQUALS TO 106.
```
SELECT ACTNO,AMOUNT FROM DEPOSIT WHERE ACTNO=106
```
8. GIVE NAME OF BORROWERS HAVING AMOUNT GREATER THAN 5000.
```
SELECT CNAME FROM BORROW WHERE AMOUNT>5000
```
9. GIVE NAME OF CUSTOMERS WHO OPENED ACCOUNT AFTER DATE '1-12-96'.
```
SELECT CNAME FROM DEPOSIT WHERE ADATE>'1/DEC/96'
```

10. DISPLAY NAME OF CUSTOMERS WHOSE ACCOUNT NO IS LESS THAN 105.
```
SELECT CNAME FROM DEPOSIT WHERE ACTNO<105
```

11. DISPLAY NAME OF CUSTOMER WHO BELONGS TO EITHER 'NAGPUR' OR 'DELHI'. (**OR & IN**)
```
SELECT CNAME FROM CUSTOMERS WHERE CITY='NAGPUR'OR CITY='DELHI'
SELECT CNAME FROM CUSTOMERS WHERE CITY IN('NAGPUR','DELHI')
```

12. DISPLAY NAME OF CUSTOMERS WITH BRANCH WHOSE AMOUNT IS GREATER THAN 4000 AND ACCOUNT NO IS LESS THAN 105.
```
SELECT CNAME,BNAME FROM DEPOSIT WHERE AMOUNT>4000 AND ACTNO<105
```

13. FIND ALL BORROWERS WHOSE AMOUNT IS GREATER THAN EQUALS TO 3000 & LESS EQUALS TOTHAN 8000. (**AND & BETWEEN**)
```
SELECT*FROM BORROW WHERE AMOUNT>=3000 AND AMOUNT<=8000
SELECT*FROM BORROW WHERE AMOUNT BETWEEN 3000 AND 8000
```

14. FIND ALL DEPOSITORS WHO DO NOT BELONGS TO 'ANDHERI' BRANCH.
```
SELECT*FROM DEPOSIT WHERE BNAME<>'ANDHERI'
```

15. DISPLAY ACCOUNT NO, CUSTOMER NAME & AMOUNT OF SUCH CUSTOMERS WHO BELONGS TO 'AJNI', 'KAROLBAGH' OR 'M.G.ROAD' AND ACCOUNT NO IS LESS THAN 104.
```
SELECT ACTNO,CNAME,AMOUNT FROM DEPOSIT WHERE BNAME
IN('AJNI','KAROLBAGH','M.G.ROAD')AND ACTNO<104
```

16. DISPLAY ALL THE DETAILS OF FIRST FIVE CUSTOMERS.
```
SELECT TOP 5 *FROM CUSTOMERS
```

17. DISPLAY ALL THE DETAILS OF FIRST THREE DEPOSITORS WHO'S AMOUNT IS GREATER THAN 1000.
```
SELEC TTOP 3 *FROM DEPOSIT WHERE AMOUNT>1000
```

18. DISPLAY LOAN NO, CUSTOMER NAME OF FIRST FIVE BORROWERS WHO'S BRANCH NAME DOES NOT BELONGS TO 'ANDHERI'.
```
SELECT TOP 5 LOANNO, CNAME FROM BORROW WHERE BNAME!='ANDHERI'
```

19. RETRIEVE ALL UNIQUE CITIES USING DISTINCT. (USE **CUSTOMERS TABLE)**
```
SELECT DISTINCT CITY FROM CUSTOMERS
```

20. RETRIEVE ALL UNIQUE BRANCHES USING DISTINCT. (USE **BRANCH TABLE)**
```
SELECT DISTINCT BNAME FROM BRANCH
```

21. RETRIEVE ALL THE RECORDS OF CUSTOMER TABLE AS PER THEIR CITY NAME IN ASCENDING ORDER.
```
SELECT*FROM CUSTOMERS ORDER BY CITY;
```

22. RETRIEVE ALL THE RECORDS OF DEPOSIT TABLE AS PER THEIR AMOUNT COLUMN IN DESCENDING ORDER.
```
SELECT*FROM DEPOSIT ORDER BY AMOUNT DESC;
```

FROM THE ABOVE GIVEN TABLES PERFORM THE FOLLOWING QUERIES (UPDATE OPERATION):

1. UPDATE DEPOSIT AMOUNT OF ALL CUSTOMERS FROM 3000 TO 5000.
```
UPDATE DEPOSIT SET AMOUNT=5000 WHERE AMOUNT=3000
```
2. CHANGE BRANCH NAME OF ANIL FROM VRCE TO C.G.ROAD. (USE **BORROW TABLE**)
```
UPDATE BORROW SET BNAME='C.G.ROAD'WHERE CNAME='ANIL'
```
3. UPDATE ACCOUNT NO OF SANDIP TO 111 & AMOUNT TO 5000.
```
UPDATE DEPOSIT SET ACTNO=111, AMOUNT=5000 WHERE CNAME='SANDIP'
```
4. GIVE 10% INCREMENT IN LOAN AMOUNT.
```
UPDATE BORROW SET AMOUNT=AMOUNT+(AMOUNT*10/100)
```
5. UPDATE DEPOSIT AMOUNT OF ALL DEPOSITORS TO 5000 WHOSE ACCOUNT NO BETWEEN 103 & 107.
```
UPDATE DEPOSIT SET AMOUNT=5000 WHERE ACTNO BETWEEN 103 AND 107
```
6. UPDATE AMOUNT OF LOAN NO 321 TO NULL.
```
UPDATE BORROW SET AMOUNT=NULL WHERE LOANNO=321
```
7. DISPLAY THE NAME OF BORROWERS WHOSE AMOUNT IS NULL.
```
SELECT*FROM BORROW WHERE AMOUNT ISNULL
```

4 **CREATE FOLLOWING TABLE USING QUERY ACCORDING TO THE DEFINITION.**

EMPLOYEE
```
CREATE TABLE EMPLOYEE
(
      EMPNO INT,
      EMPNAME VARCHAR(25),
      JOININGDATE DATETIME,
      SALARY DECIMAL(8,2),
      CITY VARCHAR(20)
)
```

INSERT THE FOLLOWING RECORDS IN THE EMPLOYEE TABLE.
```
INSERT INTO EMPLOYEE VALUES(101,'KEYUR','5/JAN/02',12000,'RAJKOT')
INSERT INTO EMPLOYEE VALUES(102,'HARDIK','15/FEB/04',14000,'AHMEDABAD')
INSERT INTO EMPLOYEE VALUES(103,'KAJAL','14/MAR/06',15000,'BARODA')
INSERT INTO EMPLOYEE VALUES(104,'BHOOMI','23/JUN/05',12500,'AHMEDABAD')
INSERT INTO EMPLOYEE VALUES(102,'HARMIT','15/FEB/04',14000,'RAJKOT')
```

FROM THE ABOVE GIVEN TABLES PERFORM THE FOLLOWING QUERIES (DELETE OPERATION):

1. DELETE ALL THE RECORDS OF EMPLOYEE TABLE HAVING SALARY GREATER THAN AND EQUALS TO 14000.
```
DELETE FROM EMPLOYEE WHERE SALARY>=14000
```
2. DELETE ALL THE EMPLOYEES WHO BELONGS TO ‘RAJKOT’ CITY.
```
DELETE FROM EMPLOYEE WHERE CITY='RAJKOT'
```

3. DELETE ALL THE EMPLOYEES WHO JOINED AFTER 1-1-2007.
```
DELETE FROM EMPLOYEE WHERE JOININGDATE>'1/JAN/2007'
```

4. DELETE ALL THE RECORDS OF EMPLOYEE TABLE. (USE **TRUNCATE**)
```
TRUNCATE TABLE EMPLOYEE
```

5. REMOVE EMPLOYEE TABLE. (USE **DROP**)
```
DROP TABLE EMPLOYEE
```

5 **CREATE FOLLOWING TABLE USING QUERY ACCORDING TO THE DEFINITION.**

STUDENT
```
CREATE TABLE STUDENT
(
        ENROLLMENTNO VARCHAR(20),
        NAME VARCHAR(25),
        CPI DECIMAL(5,2),
        BIRTHDATE DATETIME
);
```

FROM THE ABOVE GIVEN TABLES PERFORM THE FOLLOWING QUERIES (ALTER OPERATION):

1. ADD TWO MORE COLUMNS CITY VARCHAR (20) AND BACKLOG INT.
```
ALTER TABLE STUDENT ADD CITY VARCHAR(20), BACKLOG INT
```

2. CHANGE THE SIZE OF NAME COLUMN OF STUDENT FROM VARCHAR (25) TO VARCHAR (35).
```
ALTER TABLE STUDENT ALTER COLUMN NAME VARCHAR(35)
```

3. CHANGE THE DATA TYPE DECIMAL TO INT IN CPI COLUMN.
```
ALTER TABLE STUDENT ALTER COLUMN CPI INT
```

4. RENAME COLUMN ENROLLMENT NO TO ENO.
```
SP_RENAME'STUDENT.ENROLLMENTNO','ENO','COLUMN'
```

5. DELETE COLUMN CITY FROM THE STUDENT TABLE.
```
ALTER TABLE STUDENT DROP COLUMN CITY
```

6. CHANGE NAME OF TABLE STUDENT TO STUDENT_MASTER.
```
SP_RENAME'STUDENT','STUDENT_MASTER'
```

7. REMOVE THE TABLE STUDENT_MASTER.
```
DROP TABLE STUDENT_MASTER
```

6

CREATE FOLLOWING TABLE USING QUERY ACCORDING TO THE DEFINITION.

STUDENT

```
CREATE TABLE STUDENT
(
	STUID INT,
	FIRSTNAME VARCHAR(25),
	LASTNAME VARCHAR(25),
	WEBSITE VARCHAR(50),
	CITY VARCHAR(25)
);
```

INSERT THE FOLLOWING RECORDS IN THE EMPLOYEE TABLE.

```
INSERT INTO STUDENT VALUES(1011,'KEYUR','PATEL','TECHONTHENET.COM','RAJKOT')
INSERT INTO
STUDENTVALUES(1022,'HARDIK','SHAH','DIGMINECRAFT.COM','AHMEDABAD')
INSERT INTO STUDENT
VALUES(1033,'KAJAL','TRIVEDI','BIGACTIVITIES.COM','BARODA')
INSERT INTO STUDENT
VALUES(1044,'BHOOMI','GAJERA','CHECKYOURMATH.COM','AHMEDABAD')
INSERT INTO STUDENT VALUES(1055,'HARMIT','MITEL',NULL,'RAJKOT')
INSERT INTO STUDENT VALUES(1066,'ASHOK','JANI',NULL,'BARODA')
```

FROM THE ABOVE GIVEN TABLES PERFORM THE FOLLOWING QUERIES (LIKE OPERATION):

1. DISPLAY THE NAME OF STUDENTS WHO'S NAME STARTS WITH 'K'.
```
SELECT*FROM STUDENT WHERE FIRSTNAME LIKE'K%'
```

2. DISPLAY THE NAME OF STUDENTS WHO'S NAME CONSISTS OF FIVE CHARACTERS.
```
SELECT*FROM STUDENT WHERE FIRSTNAME LIKE'_____'
```

3. RETRIEVE THE FIRST NAME & LAST NAME OF STUDENTS WHOSE CITY NAME ENDS WITH A & CONTAINS SIX CHARACTERS.
```
SELECT FIRSTNAME,LASTNAME FROM STUDENT WHERE CITY LIKE'_____A'
```

4. DISPLAY ALL THE STUDENTS WHO'S LAST NAME ENDS WITH 'TEL'.
```
SELECT*FROM STUDENT WHERE LASTNAME LIKE'%TEL'
```

5. DISPLAY ALL THE STUDENTS WHOSE FIRST NAME STARTS WITH 'HA' & ENDS WITH'T'.
```
SELECT*FROM STUDENT WHERE FIRSTNAME LIKE'HA%T'
```

6. DISPLAY ALL THE STUDENTS WHOSE FIRST NAME STARTS WITH 'K' AND THIRD CHARACTER IS 'Y'.
```
SELECT*FROM STUDENT WHERE FIRSTNAME LIKE'K_Y%'
```

7. DISPLAY THE NAME OF STUDENTS HAVING NO WEBSITE AND NAME CONSISTS OF FIVE CHARACTERS.
```
SELECT*FROM STUDENT WHERE WEBSITE ISNULLAND FIRSTNAME LIKE'_____'
```

8. DISPLAY ALL THE STUDENTS WHO'S LAST NAME CONSIST OF 'JER'.
```
SELECT*FROM STUDENT WHERE LASTNAME LIKE'%JER%'
```

9. DISPLAY ALL THE STUDENTS WHOSE CITY NAME STARTS WITH EITHER ‘R’ OR ‘B’.
```
SELECT*FROM STUDENT WHERE CITY LIKE'R%'OR CITY LIKE'B%'
```

10. DISPLAY ALL THE NAME STUDENTS HAVING WEBSITES.
```
SELECT FIRSTNAME, LASTNAME FROM STUDENT WHERE WEBSITE ISNOTNULL
```

11. DISPLAY ALL THE STUDENTS WHOSE NAME STARTS FROM ALPHABET A TO H.
```
SELECT*FROM STUDENT WHERE FIRSTNAME LIKE'[A-H]%'
```

12. DISPLAY ALL THE STUDENTS WHOSE NAME’S SECOND CHARACTER IS VOWEL.
```
SELECT*FROM STUDENT WHERE FIRSTNAME LIKE'_[AEIOU]%'
```

7

FUNCTIONS

MATH FUNCTIONS.

1. DISPLAY THE RESULT OF 5 MULTIPLY BY 30.
```
SELECT 5*30
```

2. FIND OUT THE ABSOLUTE VALUE OF -25, 25, -50 AND 50.
```
SELECT ABS(-25),ABS(25),ABS(-50),ABS(50)
```

3. FIND SMALLEST INTEGER VALUE THAT IS GREATER THAN OR EQUAL TO 25.2, 25.7 AND -25.2.
```
SELECT CEILING(25.2),CEILING(25.7),CEILING(-25.2)
```

4. FIND LARGEST INTEGER VALUE THAT IS SMALLER THAN OR EQUAL TO 25.2, 25.7 AND -25.2.
```
SELECT FLOOR(25.2),FLOOR(25.7),FLOOR(-25.2)
```

5. FIND OUT REMAINDER OF 5 DIVIDED 2 AND 5 DIVIDED BY 3.
```
SELECT 5%2, 5%3
```

6. FIND OUT VALUE OF 3 RAISED TO 2ND POWER AND 4 RAISED 3RD POWER.
```
SELECT POWER(3,2),POWER(4,3)
```
7. FIND OUT THE SQUARE ROOT OF 25, 30 AND 50.
```
SELECTSQRT(25),SQRT(30),SQRT(50)
```

8. FIND OUT THE SQUARE OF 5, 15, AND 25.
```
SELECT SQUARE(5),SQUARE(25),SQUARE(25)
```

9. FIND OUT THE VALUE OF PI.
```
SELECT PI()
```

10. FIND OUT ROUND VALUE OF 157.732 FOR 2, 0 AND -2 DECIMAL POINTS.
```
SELECT ROUND(157.732,2),ROUND(157.732,0),ROUND(157.732,-2)
```

11. FIND OUT EXPONENTIAL VALUE OF 2 AND 3.
```
SELECT EXP(2),EXP(3)
```

12. FIND OUT LOGARITHM HAVING BASE E OF 10 AND 2.
```
SELECT LOG(10),LOG(2)
```

13. FIND OUT LOGARITHM HAVING BASE B HAVING VALUE 10 OF 5 AND 100.
```
SELECT LOG10(5),LOG10(100)
```

14. FIND SINE, COSINE AND TANGENT OF 3.1415.
```
SELECT SIN(3.1415),COS(3.1415),TAN(3.1415)
```

15. FIND SIGN OF -25, 0 AND 25.
```
SELECT SIGN(-25),SIGN(0),SIGN(25)
```

16. GENERATE RANDOM NUMBER USING FUNCTION.
```
SELECT RAND(),RAND(),RAND(9),RAND(3),RAND(5)
```

STRING FUNCTIONS

1. FIND THE LENGTH OF FOLLOWING. (I) NULL (II) ‘ HELLO ’ (III) BLANK
```
SELECT LEN(NULL),LEN('  HELLO    '),LEN('')
```

2. DISPLAY YOUR NAME IN LOWER & UPPER CASE.
```
SELECT LOWER('WRITEYOURNAME')AS'SMALL
LETTER',UPPER('WRITEYOURNAME')AS'CAPITAL LETTER'
```

3. DISPLAY FIRST THREE CHARACTERS OF YOUR NAME.
```
SELECT SUBSTRING('WRITEYOURNAME',1,3)
```

4. DISPLAY 3RD TO 10TH CHARACTER OF YOUR NAME.
```
SELECT SUBSTRING('WRITEYOURNAME',3,10)
```

5. WRITE A QUERY TO CONVERT ‘ABC123EFG’ TO ‘ABCXYZEFG’ & ‘ABCABCABC’ TO ‘AB5AB5AB5’ USING REPLACE.
```
SELECT REPLACE('ABC123EFG',123,'XYZ'),REPLACE('ABCABCABC','C',5)
```

6. WRITE A QUERY TO DISPLAY ASCII CODE FOR ‘a’,’A’,’z’,’Z’, 0, 9.
```
SELECT ASCII('a'),ASCII('A'),ASCII('z'),ASCII('Z'),ASCII(0),ASCII(9)
```

7. WRITE A QUERY TO DISPLAY CHARACTER BASED ON NUMBER 97, 65,122,90,48,57.
```
SELECT CHAR(97),CHAR(65),CHAR(122),CHAR(90),CHAR(48),CHAR(57)
```

8. WRITE A QUERY TO REMOVE SPACES FROM LEFT OF A GIVEN STRING ‘ HELLO WORLD ‘.
```
SELECT LTRIM('  HELLO WORLD            ') "LTRIM"
```

9. WRITE A QUERY TO REMOVE SPACES FROM RIGHT OF A GIVEN STRING ‘ HELLO WORLD ‘.
```
SELECT RTRIM('  HELLO WORLD            ') "RTRIM"
```

10. WRITE A QUERY TO DISPLAY FIRST 4 & LAST 5 CHARACTERS OF 'SQL SERVER'.
```
SELECT LEFT('SQL SERVER',4),RIGHT('SQL SERVER',5)
```

11. WRITE A QUERY TO CONVERT A STRING '1234.56' TO NUMBER (USE CAST AND CONVERT FUNCTION).
```
SELECT CAST('1234.56'ASFLOAT) "CAST",CONVERT(FLOAT,'1234.56') "CONVERT"
```

12. WRITE A QUERY TO CONVERT A FLOAT 10.58 TO INTEGER (USE CAST AND CONVERT FUNCTION).
```
SELECT CAST(10.58 ASINT) "CAST",CONVERT(INT,10.58) "CONVERT"
```

13. PUT 10 SPACE BEFORE YOUR NAME USING FUNCTION.
```
SELECT SPACE(10)+'WRITEYOURNAME'
```

14. COMBINE TWO STRINGS USING + SIGN AS WELL AS CONCAT ().
```
SELECT('HELLO'+' WORLD')AS "USING +" ,CONCAT('HELLO',' WORLD')AS "USING
FUNCTION"
```

15. FIND REVERSE OF "DARSHAN".
```
SELECT REVERSE('DARSHAN')AS REVERSE
```

16. REPEAT YOUR NAME 3 TIMES.
```
SELECT REPLICATE('NAME',3)
```

DATE FUNCTIONS

1. WRITE A QUERY TO DISPLAY THE CURRENT DATE& TIME. LABEL THE COLUMN TODAY_DATE.
```
SELECT GETDATE()AS TODAY_DATE
```

2. WRITE A QUERY TO FIND NEW DATE AFTER 365 DAY WITH REFERENCE TO TODAY.
```
SELECT GETDATE()+365
```
3. DISPLAY THE CURRENT DATE IN A FORMAT THAT APPEARS AS MAY 5 1994 12:00AM.
```
SELECT CONVERT(VARCHAR,GETDATE())
```

4. DISPLAY THE CURRENT DATE IN A FORMAT THAT APPEARS AS 03 JAN 1995.
```
SELECT CONVERT(VARCHAR,GETDATE(),106)
```

5. DISPLAY THE CURRENT DATE IN A FORMAT THAT APPEARS AS JAN 04, 96.
```
SELECT CONVERT(VARCHAR,GETDATE(),7)
```

6. WRITE A QUERY TO FIND OUT TOTAL NUMBER OF MONTHS BETWEEN 31-DEC-08 AND 31-MAR-09.
```
SELECT DATEDIFF(MONTH,'31/DEC/08','31/MAR/09')
```

7. WRITE A QUERY TO FIND OUT TOTAL NUMBER OFYEARS BETWEEN 25-JAN-12 AND 14-SEP-10.
```
SELECT DATEDIFF(YEAR,'25/JAN/12','14/SEP/10')
```

8. WRITE A QUERY TO FIND OUT TOTAL NUMBER OF HOURS BETWEEN 25-JAN-12 7:00 AND 26-JAN-12 10:30.

```
SELECT DATEDIFF(HOUR,'25/JAN/12 7:00','26/JAN/12 10:30')
```

9. WRITE A QUERY TO EXTRACT DAY, MONTH, YEAR FROM GIVEN DATE 12-MAY-16.

```
SELECT DAY('12/MAY/16')AS DAY,MONTH('12/MAY/16')AS
MONTH,YEAR('12/MAY/16')AS YEAR
```

10. WRITE A QUERY THAT ADDS 5 YEARS TO CURRENT DATE.

```
SELECT DATEADD(YEAR,5,GETDATE())
```

11. WRITE A QUERY TO SUBTRACT 2 MONTHS FROM CURRENT DATE.

```
SELECT DATEADD(MONTH,-2,GETDATE())
```

12. EXTRACT MONTH FROM CURRENT DATE USING DATENAME () AND DATEPART () FUNCTION.

```
SELECT DATENAME(MONTH,GETDATE()),DATEPART(MONTH,GETDATE())
```

13. WRITE A QUERY TO FIND OUT LAST DATE OF CURRENT MONTH.

```
SELECT EOMONTH(GETDATE())
```

14. CALCULATE YOUR AGE IN YEARS AND MONTHS.

```
SELECT DATEDIFF(YEAR,'1/1/90',GETDATE())AS'AGE IN
YEARS',DATEDIFF(MONTH,'1/1/90',GETDATE())AS'AGE IN MONTHS'
```

8 **PERFORM THE FOLLOWING QUERIES USING SET OPERATORS.**

CREATE TWO DIFFERENT TABLES AS PER FOLLOWING SCHEMA.

```
CREATE TABLE COMPUTER
(
      ROLLNO INT,
      NAME VARCHAR(20)
);

CREATE TABLE ELECTRICAL
(
      ROLLNO INT,
      NAME VARCHAR(20)
);

INSERT INTO COMPUTER VALUES(101,'AJAY')
INSERT INTO COMPUTER VALUES(109,'HARESH')
INSERT INTO COMPUTER VALUES(115,'MANISH')

INSERT INTO ELECTRICAL VALUES(101,'AJAY')
INSERT INTO ELECTRICAL VALUES(107,'MAHESH')
INSERT INTO ELECTRICAL VALUES(115,'MANISH')
```

1. DISPLAY NAME OF STUDENTS WHO IS EITHER IN COMPUTER OR IN ELECTRICAL.

```
SELECT NAME FROM COMPUTER
UNION
SELECT NAME FROM ELECTRICAL
```

2. DISPLAY NAME OF STUDENTS WHO IS EITHER IN COMPUTER OR IN ELECTRICAL INCLUDING DUPLICATE DATA.

```
SELECT NAME FROM COMPUTER
UNION ALL
SELECT NAME FROM ELECTRICAL
```

3. DISPLAY NAME OF STUDENTS WHO IS IN BOTH COMPUTER AND ELECTRICAL.

```
SELECT NAME FROM COMPUTER
INTERSECT
SELECT NAME FROM ELECTRICAL
```

4. DISPLAY NAME OF STUDENTS WHO ARE IN COMPUTER BUT NOT IN ELECTRICAL.

```
SELECT NAME FROM COMPUTER
EXCEPT
SELECT NAME FROM ELECTRICAL
```

5. DISPLAY NAME OF STUDENTS WHO ARE IN ELECTRICAL BUT NOT IN COMPUTER.

```
SELECT NAME FROM ELECTRICAL
EXCEPT
SELECT NAME FROM COMPUTER
```

6. DISPLAY ALL THE DETAILS OF STUDENTS WHO IS EITHER IN COMPUTER OR IN ELECTRICAL.

```
SELECT*FROM COMPUTER
UNION
SELECT*FROM ELECTRICAL
```

7. DISPLAY ALL THE DETAILS OF STUDENTS WHO IS IN BOTH COMPUTER AND ELECTRICAL.

```
SELECT*FROM COMPUTER
INTERSECT
SELECT*FROM ELECTRICAL
```

9 CREATE TABLE AS PER FOLLOWING.

```
CREATE TABLE CRICKET
(
NAME VARCHAR(20),
CITY VARCHAR(20),
AGE INT
);

INSERT INTO CRICKET VALUES('SACHIN TENDULKAR','MUMBAI',30)
INSERT INTO CRICKET VALUES('RAHUL DRAVID','BOMBAY',35)
INSERT INTO CRICKET VALUES('M. S. DHONI','JHARKHAND',31)
INSERT INTO CRICKET VALUES('SURESH RAINA','GUJARAT',30)
```

1. CREATE TABLE WORLDCUP FROM CRICKET WITH ALL THE COLUMNS.

```
SELECT*INTO WORLDCUP FROM CRICKET
```

2. CREATE TABLE T20 FROM CRICKET WITH FIRST TWO COLUMNS.

```
SELECT NAME,CITYINTO T20 FROM CRICKET
```

3. CREATE TABLE IPL FROM CRICKET WITH NO DATA
```
SELECT*INTO IPL FROM CRICKET WHERE 1=2
```

4. INSERT THE DATA INTO IPL FROM CRICKET WHOSE SECOND CHARACTER SHOULD BE ‘A’ AND STRING SHOULD HAVE ATLEAST 7 CHARACTERS IN CRICKET NAME FIELD.
```
INSERT INTO IPL SELECT*FROM CRICKET WHERENAMELIKE'_A_____%'
```

5. DELETE ALL THE ROWS FROM IPL.
```
DELETE FROM IPL
```

6. DELETE THE DETAIL OF CRICKETER WHOSE CITY IS JHARKHAND.
```
DELETE FROM CRICKET WHERE CITY=‘JHARKHAND’
```

7. RENAME THE TABLE IPL TO IPL2018.
```
SP_RENAME'IPL','IPL2018'
```

8. DESTROY TABLE T20 WITH ALL THE DATA.
```
DROP TABLE T20
```

10 **CREATE THE EMPLOYEE TABLE AND INSERT FOLLOWING RECORDS.**

EMPLOYEE
```
CREATE TABLE EMPLOYEE
(
      EID INT,
      ENAME VARCHAR(10),
      DEPARTMENT VARCHAR(10),
      SALARY INT,
      JOININGDATE DATETIME,
      CITY VARCHAR(10)
);

INSERT INTO EMPLOYEE VALUES(101,'RAHUL','ADMIN',56000,'1-JAN-90','RAJKOT')
INSERT INTO EMPLOYEE VALUES(102,'HARDIK','IT',18000,'25-SEP-90','AHMEDABAD')
INSERT INTO EMPLOYEE VALUES(103,'BHAVIN','HR',25000,'14-MAY-91','BARODA')
INSERT INTO EMPLOYEE VALUES(104,'BHOOMI','ADMIN',39000,'8-FEB-91','RAJKOT')
INSERT INTO EMPLOYEE VALUES(105,'ROHIT','IT',17000,'23-JUL-90','JAMNAGAR')
INSERT INTO EMPLOYEE VALUES(106,'PRIYA','IT',9000,'18-OCT-90','AHMEDABAD')
INSERT INTO EMPLOYEE VALUES(107,'NEHA','HR',34000,'25-DEC-91','RAJKOT')
```

1. DISPLAY THE HIGHEST, LOWEST, TOTAL, AND AVERAGE SALARY OF ALL EMPLOYEES. LABEL THE COLUMNS MAXIMUM, MINIMUM, TOTAL_SAL AND AVERAGE_SAL, RESPECTIVELY.
```
SELECT MAX(SALARY)AS MAXIMUM,MIN(SALARY)AS MINIMUM, SUM(SALARY)AS
TOTAL_SAL,AVG(SALARY)AS AVERAGE_SAL FROM EMPLOYEE
```

2. FIND TOTAL NUMBER OF EMPLOYEES OF EMPLOYEE TABLE.
```
SELECT COUNT(ENAME) "TOTAL NO OF EMPLOYEE" FROM EMPLOYEE
```

3. GIVE MAXIMUM SALARY FROM IT DEPARTMENT.
```
SELECT MAX(SALARY)AS MAXIMUM FROM EMPLOYEE WHERE DEPARTMENT='IT'
```

4. COUNT TOTAL NUMBER OF CITIES OF EMPLOYEE WITHOUT DUPLICATION.

```
SELECT COUNT(DISTINCT CITY) "TOTAL CITY" FROM EMPLOYEE
```

5. DISPLAY CITY WITH THE TOTAL NUMBER OF EMPLOYEES BELONGING TO EACH CITY.

```
SELECT CITY,COUNT(ENAME)FROM EMPLOYEE GROUPBY CITY
```

6. DISPLAY CITY HAVING MORE THAN ONE EMPLOYEES.

```
SELECT CITY,COUNT(ENAME)FROM EMPLOYEE GROUPBY CITY HAVINGCOUNT(ENAME)>1
```

7. GIVE TOTAL SALARY OF EACH DEPARTMENT OF EMPLOYEE TABLE.

```
SELECT DEPARTMENT,SUM(SALARY)FROM EMPLOYEE GROUPBY DEPARTMENT
```

8. GIVE AVERAGE SALARY OF EACH DEPARTMENT OF EMPLOYEE TABLE WITHOUT DISPLAYING THE RESPECTIVE DEPARTMENT NAME.

```
SELECT AVG(SALARY)FROM EMPLOYEE GROUPBY DEPARTMENT
```

9. GIVE MINIMUM SALARY OF EMPLOYEE WHO BELONGS TO AHMEDABAD.

```
SELECT MIN(SALARY)FROM EMPLOYEE WHERE CITY='AHMEDABAD'
```

10. LIST THE DEPARTMENTS HAVING TOTAL SALARIES MORE THAN 50000AND LOCATED IN CITY RAJKOT.

```
SELECT DEPARTMENT,SUM(SALARY)FROM EMPLOYEE  WHERE CITY='RAJKOT'GROUPBY
DEPARTMENT HAVINGSUM(SALARY)>50000
```

11. COUNT THE NUMBER OF EMPLOYEES LIVING IN RAJKOT.

```
SELECT COUNT(ENAME)FROM EMPLOYEE WHERE CITY='RAJKOT'
```

12. DISPLAY THE DIFFERENCE BETWEEN THE HIGHEST AND LOWEST SALARIES. LABEL THE COLUMN DIFFERENCE.

```
SELECT MAX(SALARY)-MIN(SALARY) "DIFFERENCE" FROM EMPLOYEE
```

13. DISPLAY THE TOTAL NUMBER OF EMPLOYEES HIRED BEFORE 1ST JANUARY, 1991.

```
SELECTCOUNT(ENAME)FROM EMPLOYEE WHERE JOININGDATE<'1/JAN/91'
```

14. DISPLAY TOTAL SALARY OF EACH DEPARTMENT WITH TOTAL SALARY EXCEEDING 35000 AND SORT THE LIST BY TOTAL SALARY.

```
SELECT DEPARTMENT,SUM(SALARY)FROM EMPLOYEE GROUPBY DEPARTMENT HAVING
SUM(SALARY)>35000 ORDERBYSUM(SALARY)
```

15. LIST OUT DEPARTMENT NAMES IN WHICH MORE THAN TWO EMPLOYEES.

```
SELECT DEPARTMENT FROM EMPLOYEE GROUP BY DEPARTMENT HAVING COUNT(ENAME)>2
```

11 **JOINS**

1. CREATE 3 DIFFERENT TABLES AS PER FOLLOWING SCHEMA.

```
CREATE TABLE STUDENT
(
      RNO INT,
        NAME VARCHAR(20),
        BRANCH VARCHAR(10)
);

CREATE TABLE RESULT
(
        RNO INT,
        SPI FLOAT
);

CREATETABLE EMPLOYEE
(
        EMPNO VARCHAR(10),
        NAME VARCHAR(10),
        MANAGERNO VARCHAR(10)
);

INSERT INTO STUDENT VALUES(101,'RAJU','CE')
INSERT INTO STUDENT VALUES(102,'AMIT','CE')
INSERT INTO STUDENT VALUES(103,'SANJAY','ME')
INSERT INTO STUDENT VALUES(104,'NEHA','EC')
INSERT INTO STUDENT VALUES(105,'MEERA','EE')
INSERT INTO STUDENT VALUES(106,'MAHESH','ME')

INSERT INTO RESULT VALUES(101,8.8)
INSERT INTO RESULT VALUES(102,9.2)
INSERT INTO RESULT VALUES(103,7.6)
INSERT INTO RESULT VALUES(104,8.2)
INSERT INTO RESULT VALUES(105,7.0)
INSERT INTO RESULT VALUES(107,8.9)

INSERT INTO EMPLOYEE VALUES('E01','TARUN', NULL)
INSERT INTO EMPLOYEE VALUES('E02','ROHAN','E02')
INSERT INTO EMPLOYEE VALUES('E03','PRIYA','E01')
INSERT INTO EMPLOYEE VALUES('E04','MILAN', 'E03')
INSERT INTO EMPLOYEE VALUES('E05','JAY', 'E01')
INSERT INTO EMPLOYEE VALUES('E06','ANJANA','E04')
```

2. COMBINE INFORMATION FROM STUDENT AND RESULT TABLE USING CROSS JOIN OR CARTESIAN PRODUCT.

```
SELECT S.RNO,NAME,BRANCH,SPI FROM STUDENT S, RESULT R
```

3. DISPLAY RNO, NAME, BRANCH AND SPI OF ALL STUDENTS.

```
SELECT S.RNO,NAME,BRANCH,SPI
FROM STUDENT S
FULL OUTER JOIN RESULT R
ON S.RNO=R.RNO
```

4. DISPLAY RNO, NAME, BRANCH AND SPI OF CE BRANCH’S STUDENT ONLY.

```
SELECT S.RNO,NAME,BRANCH,SPI
FROM STUDENT S
FULL OUTER JOIN RESULT R
ON S.RNO=R.RNO
WHERE S.BRANCH='CE'
```

5. DISPLAY RNO, NAME, BRANCH AND SPI OF OTHER THAN EC BRANCH'S STUDENT ONLY.

```
SELECT S.RNO,NAME,BRANCH,SPI
FROM STUDENT S
FULL OUTERJ OIN RESULT R
ON S.RNO=R.RNO
WHERE S.BRANCH!='EC'
```

6. DISPLAY AVERAGE RESULT OF EACH BRANCH.

```
SELECT BRANCH,AVG(SPI)
FROM STUDENT S, RESULT R
WHERE S.RNO=R.RNO
GROUP BY S.BRANCH
```

7. DISPLAY AVERAGE RESULT OF EACH BRANCH AND SORT THEM IN ASCENDING ORDER BY SPI.

```
SELECT BRANCH,AVG(SPI)
FROM STUDENT S, RESULT R
WHERE S.RNO=R.RNO
GROUP BY S.BRANCH
ORDER BY AVG(SPI)
```

8. DISPLAY AVERAGE RESULT OF CE AND ME BRANCH.

```
SELECT BRANCH,AVG(SPI)
FROM STUDENT S, RESULT R
WHERE S.RNO=R.RNO AND BRANCH IN('CE','ME')
GROUP BY S.BRANCH
```

9. PERFORM THE LEFT OUTER JOIN ON STUDENT AND RESULT TABLES.

```
SELECT S.RNO,NAME,BRANCH,SPI
FROM STUDENT S
LEFT OUTER JOIN RESULT R
ON S.RNO=R.RNO
```

10. PERFORM THE RIGHT OUTER JOIN ON STUDENT AND RESULT TABLES.

```
SELECT S.RNO,NAME,BRANCH,SPI
FROM STUDENT S
RIGHTOUTERJOIN RESULT R
ON S.RNO=R.RNO
```

11. PERFORM THE FULL OUTER JOIN ON STUDENT AND RESULT TABLES.

```
SELECT S.RNO,NAME,BRANCH,SPI
FROM STUDENT S
FULLOUTERJOIN RESULT R
ON S.RNO=R.RNO
```

12. RETRIEVE THE NAMES OF EMPLOYEE AND THE NAMES OF THEIR RESPECTIVE MANAGERS FROM THE EMLOYEE TABLE.

```
SELECT EMP.NAME,MNGR.NAME
FROM EMPLOYEE EMP,EMPLOYEE MNGR
WHERE EMP.MANAGERNO=MNGR.EMPLOYEENO
```

12

```
CREATE TABLE CITY
(
       CITYID INT NOT NULL PRIMARY KEY,
       NAME VARCHAR(50)NOT NULL UNIQUE,
       PINCODE INT NOT NULL,
       REMARKS VARCHAR(50)NULL
);

CREATE TABLE VILLAGE
(
       VID INT NOT NULL PRIMARY KEY,
       NAME VARCHAR(50)NOT NULL,
       CITYID INT NOT NULL REFERENCES CITY(CITYID)
)

INSERT INTO CITY VALUES (1,'RAJKOT',360005,'GOOD')
INSERT INTO CITY VALUES (2,'SURAT',335009,'VERY GOOD')
INSERT INTO CITY VALUES (3,'BARODA',390001,'AWESOME')
INSERT INTO CITY VALUES (4,'JAMNAGAR',361003,'SMART')
INSERT INTO CITY VALUES (5,'JUNAGADH',362229,'HISTORIC')
INSERT INTO CITY VALUES (6,'MORVI',363641,'CERAMIC CITY')

INSERT INTOVILLAGE VALUES (101,'RAIYA',1)
INSERT INTOVILLAGE VALUES (102,'MADHAPAR',1)
INSERT INTOVILLAGE VALUES (103,'DODKA',3)
INSERT INTOVILLAGE VALUES (104,'FALLA',4)
INSERT INTOVILLAGE VALUES (105,'BHESAN',5)
INSERT INTOVILLAGE VALUES (106,'DHORAJI',5)
```

1. DISPLAY ALL THE VILLAGES OF RAJKOT CITY.

```
SELECT VILLAGE.[NAME]
FROM CITY
INNER JOIN VILLAGE
ON CITY.CITYID = VILLAGE.CITYID
WHERE CITY.[NAME] ='RAJKOT'
```

2. DISPLAY CITY ALONG WITH THEIR VILLAGES & PIN CODE.

```
SELECT CITY.[NAME],VILLAGE.[NAME],PINCODE
FROM CITY
INNER JOIN VILLAGE
ON CITY.CITYID = VILLAGE.CITYID
```

3. DISPLAY THE CITY HAVING MORE THAN ONE VILLAGE.

```
SELECT CITY.[NAME],COUNT(VILLAGE.VID)AS TOTAL
FROM CITY
INNER JOIN VILLAGE
ON CITY.CITYID = VILLAGE.CITYID
GROUP BY CITY.[NAME]
HAVING COUNT(VILLAGE.VID)>1
```

4. DISPLAY THE CITY HAVING NO VILLAGE.

```
SELECT CITY.[NAME],COUNT(VILLAGE.VID)AS TOTAL
FROM CITY
LEFTOUTERJOIN VILLAGE
ON CITY.CITYID = VILLAGE.CITYID
GROUP BY CITY.[NAME]
HAVING COUNT(VILLAGE.VID)< 1
```

5. COUNT THE TOTAL NUMBER OF VILLAGE IN EACH CITY.

```
SELECT CITY.[NAME],COUNT(VILLAGE.VID)AS TOTALV
FROM CITY
INNER JOIN VILLAGE
ON CITY.CITYID = VILLAGE.CITYID
GROUPBY CITY.[NAME]
```

6. COUNT THE NUMBER OF CITIES HAVING MORE THAN ONE VILLAGE.

```
SELECTCOUNT(*)
FROM
(
SELECT CITY.[NAME],COUNT(VILLAGE.VID)AS TOTAL
FROM CITY
LEFT OUTER JOIN VILLAGE
ON CITY.CITYID = VILLAGE.CITYID
GROUP BY CITY.[NAME]
)AS T
WHERE TOTAL > 1
```

13

```
CREATE TABLE STUDENT
  (
        RNO INT PRIMARY KEY,
        NAME VARCHAR(50),
        BRANCH VARCHAR(20) DEFAULT 'General',
        SPI DECIMAL(4,2) CHECK(SPI BETWEEN 0 AND 10),
        BKLOG INT CHECK (BKLOG>=0)
)
```

14

```
CREATE TABLE STUDENT
(
        RNO INT,
        NAME VARCHAR(50),
        CITY VARCHAR(50),
        DID INT
);

CREATE TABLE ACADEMIC
(
        RNO INT,
        SPI DECIMAL(4,2),
        BKLOG INT
);

CREATE TABLE DEPARTMENT
(
        DID INT,
        DNAME VARCHAR(50)
);

INSERT INTO STUDENT VALUES(101,'RAJU','RAJKOT',10);
INSERT INTO STUDENT VALUES(102,'AMIT','AHMEDABAD',20);
INSERT INTO STUDENT VALUES(103,'SANJAY','BARODA',40);
INSERT INTO STUDENT VALUES(104,'NEHA','RAJKOT',20);
INSERT INTO STUDENT VALUES(105,'MEERA','AHMEDABAD',30);
INSERT INTO STUDENT VALUES(106,'MAHESH','BARODA',10);
INSERT INTO ACADEMIC VALUES(101,8.8,0);
INSERT INTO ACADEMIC VALUES(102,9.2,2);
INSERT INTO ACADEMIC VALUES(103,7.6,1);
INSERT INTO ACADEMIC VALUES(104,8.2,4);
INSERT INTO ACADEMIC VALUES(105,7.0,2);
INSERT INTO ACADEMIC VALUES(106,8.9,3);

INSERT INTO DEPARTMENT VALUES(10,'COMPUTER');
INSERT INTO DEPARTMENT VALUES(20,'ELECTRICAL');
INSERT INTO DEPARTMENT VALUES(30,'MECHANICAL');
INSERT INTO DEPARTMENT VALUES(40,'CIVIL');
```

1. DISPLAY DETAILS OF STUDENTS WHO ARE FROM COMPUTER DEPARTMENT.

```
SELECT RNO,NAME,CITY
FROM STUDENT
WHERE DID=(SELECT DID
         FROM DEPARTMENT
         WHERE DNAME='COMPUTER');
```

2. DISPLAY NAME OF STUDENTS WHOSE SPI IS MORE THAN 8.

```
SELECT NAME
FROM STUDENT
WHERE RNO IN(SELECT RNO
               FROM ACADEMIC
               WHERE SPI>8);
```

3. DISPLAY DETAILS OF STUDENTS OF COMPUTER DEPARTMENT WHO BELONGS TO RAJKOT CITY.

```
SELECT RNO,NAME,CITY
FROM STUDENT
WHERE CITY='RAJKOT'AND DID=(SELECT DID
                           FROM DEPARTMENT
                           WHERE DNAME='COMPUTER');
```

4. FIND TOTAL NUMBER OF STUDENTS OF ELECTRICAL DEPARTMENT.

```
SELECT COUNT(*)
FROM STUDENT
WHERE DID=(SELECT DID
         FROM DEPARTMENT
         WHERE DNAME='ELECTRICAL');
```

5. DISPLAY NAME OF STUDENT WHO IS HAVING MAXIMUM SPI.

```
SELECT NAME
FROM STUDENT
WHERE RNO=(SELECT RNO
         FROM ACADEMIC
         WHERE SPI=(SELECT MAX(SPI)FROM ACADEMIC)
         );
```

6. DISPLAY DETAILS OF STUDENTS HAVING MORE THAN 1 BACKLOG.

```
SELECT*
FROM STUDENT
WHERE RNO IN(SELECT RNO
                FROM ACADEMIC
                WHERE BKLOG>1);
```

7. DISPLAY NAME OF STUDENT WHO IS HAVING SECOND HIGHEST SPI.

```
SELECT NAME
FROM STUDENT
WHERE RNO=(SELECT RNO
          FROM ACADEMIC
          WHERE SPI=(SELECT MAX(SPI)
                      FROM ACADEMIC
                      WHERE SPI<(SELECT MAX(SPI)
                                  FROM ACADEMIC)
                      )
          );
OR

SELECT NAME
FROM STUDENT
WHERE RNO=(SELECT RNO
          FROM ACADEMIC
          WHERE SPI=(SELECT TOP 1 SPI
                      FROM (SELECT DISTINCT TOP 2 SPI FROM ACADEMIC ORDER
                      BY SPI DESC)AS TEMP
                      ORDER BY SPI ASC)
                      );
```

8. DISPLAY NAME OF STUDENTS WHO ARE EITHER FROM COMPUTER DEPARTMENT OR FROM MECHANICAL DEPARTMENT.

```
SELECT NAME
FROM STUDENT
WHERE DID IN(SELECT DID
                FROM DEPARTMENT
                WHERE DNAME='COMPUTER'OR DNAME='MECHANICAL');
```

9. DISPLAY NAME OF STUDENTS WHO ARE IN SAME DEPARTMENT AS 102 STUDYING IN.

```
SELECT NAME
FROM STUDENT
WHERE DID=(SELECT DID
          FROM STUDENT
          WHERE RNO=102);
```

10. DISPLAY NAME OF STUDENTS WHOSE SPI IS MORE THAN 9 AND WHO IS FROM ELECTRICAL DEPARTMENT.

```
SELECT NAME
FROM STUDENT
WHERE RNO IN(SELECT RNO FROM ACADEMIC WHERE SPI>9)
      AND
```

```
          DID=(SELECT DID FROM DEPARTMENT WHERE DNAME='ELECTRICAL');
```

15

```
CREATE TABLE STUDENT
(
      RNO INT PRIMARY KEY,
      NAME VARCHAR(20),
      BRANCH VARCHAR(10),
      SPI FLOAT,
      BKLOG INT
)

INSERT INTO STUDENT VALUES(101,'RAJU','CE',8.80,0)
INSERT INTO STUDENT VALUES(102,'AMIT','CE',2.20,3)
INSERT INTO STUDENT VALUES(103,'SANJAY','ME',1.50,6)
INSERT INTO STUDENT VALUES(104,'NEHA','EC',7.65,1)
INSERT INTO STUDENT VALUES(105,'MEERA','EE',5.52,2)
INSERT INTO STUDENT VALUES(106,'MAHESH','EC',4.50,3)
```

1. CREATE A VIEW PERSONAL WITH ALL COLUMNS.

```
--SELECT * FROM PERSONAL
CREATE VIEW PERSONAL
AS
SELECT RNO,NAME,BRANCH,SPI,BKLOG
FROM STUDENT
```

2. CREATE A VIEW STUDENT_DETAILS HAVING COLUMNS NAME, BRANCH & SPI.

```
--SELECT * FROM STUDENT_DETAILS
CREATE VIEW STUDENT_DETAILS
AS
SELECT NAME,BRANCH,SPI
FROM STUDENT
```

3. CREATE A VIEW ACADEMIC HAVING COLUMNS RNO, NAME, BRANCH.

```
--SELECT * FROM ACADEMIC ORDER BY NAME
CREATE VIEW ACADEMIC
AS
SELECT RNO,[NAME],BRANCH
FROM STUDENT
```

4. CREATE A VIEW STUDENT_DATA HAVING ALL COLUMNS BUT STUDENTS WHOSE BKLOG MORE THAN 2.

```
--SELECT * FROM STUDENT_DATA
CREAT EVIEW STUDENT_DATA
AS
SELECT RNO,NAME, BRANCH, SPI, BKLOG
FROM   STUDENT
WHERE BKLOG > 2
```

5. CREATE A VIEW STUDENT_PATTERN HAVING RNO, NAME & BRANCH COLUMNS IN WHICH NAME CONSISTS OF FOUR LETTERS.

```
--SELECT * FROM STUDENT_PATTERN
```

```sql
CREATE VIEW STUDENT_PATTERN
AS
SELECT RNO,NAME, BRANCH
FROM   STUDENT
WHERE NAME LIKE'____'
```

6. INSERT A NEW RECORD TO ACADEMIC VIEW. (107, MEET, ME).

```sql
INSERT INTO ACADEMIC VALUES(107,'MEET','ME')
```

7. UPDATE THE BRANCH OF AMIT FROM CE TO ME IN STUDENT_DETAILS VIEW.

```sql
UPDATE STUDENT_DETAILS SET BRANCH='ME'WHERE NAME='AMIT'
```

8. DELETE A STUDENT WHOSE ROLL NUMBER IS 104 FROM ACADEMIC VIEW.

```sql
DELETE FROM ACADEMIC WHERE RNO=104
```

16

```sql
CREATE TABLE Student
(
      RNo INT NOT NULL,
      Name VARCHAR(50)NOT NULL,
      Branch VARCHAR(50)NOT NULL
);

CREATETABLE Result
(
      RNo INT NOT NULL,
      SPI Decimal(5,2) NOT NULL
);

INSERT INTO Student VALUES (101,'RAJU','CE')
INSERT INTO Student VALUES (102,'AMIT','CE')
INSERT INTO Student VALUES (103,'SANJAY','ME')
INSERT INTO Student VALUES (104,'NEHA','EC')
INSERT INTO Student VALUES (105,'MEERA','EE')
INSERT INTO Student VALUES (106,'MAHESH','ME')

INSERT INTO Result VALUES (101,8.8)
INSERT INTO Result VALUES (102,9.2)
INSERT INTO Result VALUES (103,7.6)
INSERT INTO Result VALUES (104,8.2)
INSERT INTO Result VALUES (105,7.0)
INSERT INTO Result VALUES (105,8.9)
```

Create Following Procedures from above tables.

1. CREATE A STORED PROCEDURE TO DISPLAY ALL THE RECORDS. (RNO, NAME, BRANCH, SPI)

```sql
--EXEC SELECTALL
CREATE PROCEDURE SELECTALL
AS
SELECT S.RNO, S.NAME, S.BRANCH, R.SPI
FROM STUDENT S
INNER JOIN RESULT R
ON S.RNO = R. RNO
```

2. CREATE A STORED PROCEDURE TO GET A ROLL NUMBER FROM USER AND DISPLAY ALL THE DETAILS OF IT.

```
--EXEC SELECTPK 102
CREATE PROCEDURE SELECTPK
@RNO INT
AS
SELECT S.RNO, S.NAME, S.BRANCH, R.SPI
FROM STUDENT S
INNER JOIN RESULT R
ON S.RNO = R. RNO
WHERE S.RNO = @RNO
```

3. CREATE A STORED PROCEDURE TO INSERT A RECORD IN STUDENT TABLE. (107, RAJ, EE).

```
--EXEC STUDENTINSERT 107,'RAJ','EE'
CREATE PROCEDURE STUDENTINSERT
@RNO     INT,
@NAME    VARCHAR(50),
@BRANCH  VARCHAR(20)
AS
INSERT INTO STUDENT
VALUES
(
@RNO,
@NAME,
@BRANCH
)
```

4. CREATE A STORED PROCEDURE TO UPDATE BRANCH OF ROLL NUMBER 105 TO EC IN STUDENT TABLE.

```
--EXEC STUDENTUPDATE 105,'EC'
CREATE PROCEDURE STUDENTUPDATE
@RNO     INT,
@BRANCH  VARCHAR(20)
AS
UPDATE STUDENT
SET
BRANCH = @BRANCH
WHERE RNO = @RNO
```

5. CREATE A STORED PROCEDURE TO DELETE A RECORD IN STUDENT TABLE WHOSE ROLL NUMBER IS 103.

```
--EXEC STUDENTDELETE 103
CREATE PROCEDURE STUDENTDELETE
@RNO INT
AS
DELETEFROM STUDENT
WHERE RNO = @RNO
```

6. USE FOLLOWING COMMANDS ON ABOVE STORED PROCEDURE. SP_HELP, SP_HELPTEXT, SP_DEPENDS

```
SP_HELPDATABASE_OBJECTNAME
SP_HELPTEXTDATABASE_OBJECTNAME
SP_DEPENDSDATABASE_OBJECTNAME
```

17

1. WRITE A FUNCTION TO PRINT NUMBER FROM 1 TO 10. (USING WHILE LOOP)

```
DECLARE @V1 INT
SET @V1=1
WHILE (@V1<11)
BEGIN
PRINT @V1
SET @V1=@V1+1
END
```

2. WRITE A FUNCTION TO CHECK WHERE GIVEN NUMBER IS ODD OR EVEN.

```
DECLARE @V1 INT
SET @V1=15
BEGIN
IF @V1%2=0
PRINT'EVEN';
ELSE
PRINT'ODD';
END
```

3. WRITE A FUNCTION TO PRINT ODD NUMBERS BETWEEN 1 AND 10.

```
DECLARE @V1 INT
SET @V1=1
WHILE (@V1<11)
BEGIN
IF @V1%2!=0
PRINT @V1;
SET @V1=@V1+1
END
```

4. WRITE A FUNCTION TO PRINT SUM OF NUMBERS FROM 1 TO 50.

```
DECLARE @V INT,@SUM INT
SET @V=0
SET @SUM=0
WHILE(@V<=50)
BEGIN
      SET @SUM=@SUM+@V
      SET @V=@V+1
END
PRINT @SUM
```

5. WRITE A FUNCTION TO PRINT SUM OF EVEN NUMBERS BETWEEN 1 TO 20.

```
DECLARE @V INT,@SUM INT
SET @V=0
SET @SUM=0
WHILE(@V<=20)
BEGIN
      IF(@V%2=0)
      SET @SUM=@SUM+@V
      SET @V=@V+1
END
```

```
PRINT @SUM
```

6. WRITE A FUNCTION TO CHECK WEATHER GIVEN NUMBER IS PRIME OR NOT.

```
CREATE PROCEDURE PRIME
@N INT
AS
DECLARE @I INT,@FLAG INT
SET @I=2
SET @FLAG=1

WHILE(@I<=@N/2)
BEGIN
      IF(@N%@I=0)
      BEGIN
            SET @FLAG=0
            BREAK
      END
      SET @I=@I+1
END

IF(@FLAG=1)
      PRINT'PRIME'
ELSE
      PRINT'NOT PRIME'

--EXEC PRIME 45
```

7. WRITE A FUNCTION TO INSERTING EVEN NUMBERS INTO EVEN TABLE & ODD NUMBERS INTO ODD TABLE BETWEEN 1 TO 50.

```
DECLARE @V1 INT
SET @V1=1
WHILE (@V1 <= 50)
BEGIN
IF @V1%2=0
      BEGIN
      INSERTINTO EVEN(NO)VALUES (@V1);
      END
ELSE
      BEGIN
      INSERTINTO ODD(NO)VALUES (@V1);
      END
SET @V1 = @V1 + 1
END
```

18

1. CREATE A TRIGGER ON STUDENT TABLE FOR INSERT , UPDATE AND DELETE STATEMENT TO DISPLAY A MESSAGE “RECORD IS AFFECTED”.

```
CREATE TRIGGER STUDENT_MSG
ON STUDENT
AFTER INSERT,UPDATE,DELETE
AS
BEGIN
      PRINT'RECORD AFFECTED'
END
```

2. CREATE A TRIGGER ON STUDENT TABLE FOR INSERT STATEMENT TO INSERT DESCRIPTION (RECORD WITH RNO=[101] IS INSERTED ON [CURRENT DATE]) IN AUDIT TABLE.

```
CREATE TRIGGER TR_STUDENT_FORINSERT
ON STUDENT
FOR INSERT
AS
BEGIN
      DECLARE @ID INT
      SELECT @ID= RNO FROM INSERTED

      INSERT INTO AUDIT VALUES(@ID,'RECORD WITH RNO='+CAST(@ID AS
     VARCHAR(10))+' IS INSERTED ON '+CAST(GETDATE()AS VARCHAR(50)))
END
```

3. CREATE A TRIGGER ON STUDENT TABLE FOR UPDATE STATEMENT TO INSERT DESCRIPTION (RECORD WITH RNO=[101] IS UPDATED ON [CURRENT DATE]) IN AUDIT TABLE.

```
CREATE TRIGGER TR_STUDENT_FORUPDATE
ON STUDENT
FO RUPDATE
AS
BEGIN
      DECLARE @ID INT
      SELECT @ID= RNO FROM INSERTED

      INSERT INTO AUDIT VALUES(@ID,'RECORD WITH RNO='+CAST(@ID AS
     VARCHAR(10))+' IS UPDATED ON'+CAST(GETDATE()AS VARCHAR(50)))
END
```

4. CREATE A TRIGGER ON STUDENT TABLE FOR DELETE STATEMENT TO INSERT DESCRIPTION (RECORD WITH RNO=[101] IS DELETED ON [CURRENT DATE]) IN AUDIT TABLE.

```
CREATE TRIGGER TR_STUDENT_FORDELETE
ON STUDENT
FOR DELETE
AS
BEGIN
      DECLARE @ID INT
      SELECT @ID= RNO FROMDELETED

      INSERT INTO AUDIT VALUES(@ID,'RECORD WITH RNO='+CAST(@ID AS
VARCHAR
       (10))+'IS DELETEDON'+CAST(GETDATE()AS VARCHAR(50)))
END
```

5. CREATE A TRIGGER ON RESULT TABLE FOR INSERT STATEMENT TO UPDATE TOTAL MARKS AUTOMATICALLY. HERE TOTAL MARKS IS SUM OF OF SUB1,SUB2 AND SUB3.

```
CREATE TRIGGER TR_TOTALMARKS
ON RESULT
FOR INSERT
AS
BEGIN
      DECLARE @S1 INT, @S2 INT, @S3 INT, @TOTAL INT
```

```
        SELECT @S1= SUB1 FROM INSERTED
        SELECT @S2= SUB2 FROM INSERTED
        SELECT @S3= SUB3 FROM INSERTED
        SET @TOTAL= @S1+@S2+@S3

        UPDATE RESULT
        SET TOTAL=@TOTAL
        WHERE SUB1=@S1 AND SUB2=@S2 AND SUB3=@S3
END
```

19

1. CREATE A CURSOR TO DECREASE THE SALARY OF ALL EMPLOYEES BY 500. DISPLAY ERROR MESSAGE IF SALARY BECOMES NEGATIVE AND DOES NOT DECREASE SALARY.

```
DECLARE
    @EID INT, @SALARY DECIMAL(8,2);

DECLARE CURSOR_EMP CURSOR
FOR SELECT EID, SALARY FROM EMPLOYEE;

OPEN CURSOR_EMP;
FETCHNEXTFROM CURSOR_EMP INTO
    @EID, @SALARY;

WHILE@@FETCH_STATUS= 0
BEGIN
        SET @SALARY=@SALARY-500
        IF (@SALARY<0)
              PRINT'SALARY MUST BE GREATER THAN 0'
        ELSE
              UPDATE EMPLOYEE
              SET SALARY=@SALARY
              WHERE EID=@EID
FETCHNEXTFROM CURSOR_EMP INTO
        @EID, @SALARY;
END;

CLOSE CURSOR_EMP;
DEALLOCATE CURSOR_EMP;
```

2. CREATE A CURSOR TO INSERT DETAILS OF STUDENTS OF COMPUTER BRANCH INTO NEWSTUDENT TABLE.

```
DECLARE
   @RNO INT, @NAME VARCHAR(50),@BRANCH VARCHAR(50),@SPIDECIMAL(4,2);

DECLARE CURSOR_STUDENT CURSOR
FORSELECT RNO, NAME, BRANCH, SPI
FROM STUDENT;

OPEN CURSOR_STUDENT;
FETCHNEXTFROM CURSOR_STUDENT INTO
    @RNO, @NAME, @BRANCH, @SPI;

WHILE@@FETCH_STATUS= 0
BEGIN
        IF (@BRANCH='CE')
              INSERTINTO NEWSTUDENT VALUES (@RNO, @NAME, @BRANCH, @SPI)
```

```
FETCHNEXTFROM CURSOR_STUDENT INTO
        @RNO, @NAME, @BRANCH, @SPI;
END;

CLOSE CURSOR_STUDENT;
DEALLOCATE CURSOR_STUDENT;
```

3. CREATE A CURSOR TO DECREASE SPI OF ALL STUDENTS BY 7. DISPLAY ERROR MESSAGE IF SPI BECOMES NEGATIVE AND DOES NOT DECREASE SPI.

```
DECLARE
    @RNO INT, @SPI DECIMAL(8,2);
DECLARE CURSOR_STUDENT CURSOR
FORSELECT RNO, SPI
FROM STUDENT;
OPEN CURSOR_STUDENT;
FETCHNEXTFROM CURSOR_STUDENT INTO
    @RNO, @SPI;
WHILE@@FETCH_STATUS= 0
BEGIN
      SET @SPI=@SPI-7
      IF (@SPI<0)
            PRINT'SPI MUST BE GREATER THAN 0'
      ELSE
            UPDATE STUDENT
            SET SPI=@SPI
            WHERE RNO=@RNO
FETCHNEXTFROM CURSOR_STUDENT INTO
        @RNO, @SPI;
END;
CLOSE CURSOR_STUDENT;
DEALLOCATE CURSOR_STUDENT;
```