**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| ***S.no*** | ***Name of program*** | ***Page number*** | ***Signature and remarks*** |
| ***1.*** | *Write the queries for Data Definition Language in Relational Database Management Systems*. | 2 - 8 |  |
| ***2.*** | *Write the queries for Data Manipulation Language in Relational Database Management Systems.* | 9 - 12 |  |
| ***3.*** | *Write the queries for Data Control Language in Relational Database Management Systems.* | 13 |  |
| ***4.*** | *Write the queries for Logical operations.* | 14 - 15 |  |
| ***5.*** | *Write the queries for Arithmetic operations.* | 16 - 17 |  |
| ***6.*** | *Write the queries using character, date, number and group functions.* | 18 - 20 |  |
| ***7.*** | *Write the queries for non-equi join, and view formation.* | 21 - 23 |  |
| ***8.*** | *Write the queries for sub queries and nested queries.* | 24 - 25 |  |
| ***9.*** | *Write the queries for roll back, commit, savepoint.* | 26 - 28 |  |
| ***10.*** | *Write the queries for equi-join and different view formation.* | 29 - 32 |  |
| ***11.*** | *Write the queries for procedure of triggering any table.* | 33 - 34 |  |
| ***12.*** | *Write the queries to make procedure to find area of circle and insert it into a table.* | 35 - 36 |  |
| ***13.*** | *Write the queries to make procedure to find for computing the amount of telephone bill.* | 37 |  |
| ***14.*** | *Write the queries to create a trigger for before insertion or update.* | 38 - 39 |  |
| ***15.*** | *Write the queries to demonstrate Referential Integrity (Primary and Foreign key).* | 40 - 42 |  |

**Practical – 1**

***Aim –*** *Write the queries for Data Definition Language in Relational Database Management Systems.*

**Data Definition Language:**DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database.

**Examples of DDL commands:**

* ***CREATE*** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
* **DROP** – is used to delete objects from the database.
* [**ALTER**](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)- is used to alter the structure of the database.
* [**TRUNCATE**](https://www.geeksforgeeks.org/sql-drop-truncate/)– is used to remove all records from a table, including all spaces allocated for the records are removed.
* [**COMMENT**](https://www.geeksforgeeks.org/sql-comments/) – is used to add comments to the data dictionary.
* [**RENAME**](https://www.geeksforgeeks.org/sql-alter-rename/)– is used to rename an object existing in the database.

***A. CREATE Command****:* There are two **CREATE** statements available in SQL:

1. **CREATE DATABASE**
2. **CREATE TABLE**

***CREATE DATABASE***

A **Database** is defined as a structured set of data. So, in SQL the very first step to store the data in a well-structured manner is to create a database. The **CREATE DATABASE** statement is used to create a new database in SQL.

**Syntax**:

CREATE DATABASE database\_name;

**database\_name**: name of the database.

***CREATE TABLE***

Now to store the data we need a table to do that. The CREATE TABLE statement is used to create a table in SQL. We know that a table comprises of rows and columns. So while creating tables we have to provide all the information to SQL about the names of the columns, type of data to be stored in columns, size of the data etc. Let us now dive into details on how to use CREATE TABLE statement to create tables in SQL.

**Syntax**:

CREATE TABLE table\_name

(

column1 data\_type(size),

column2 data\_type(size),

column3 data\_type(size),

....

);

**table\_name**: name of the table.

**column1** name of the first column.

**data\_type**: Type of data we want to store in the particular column.

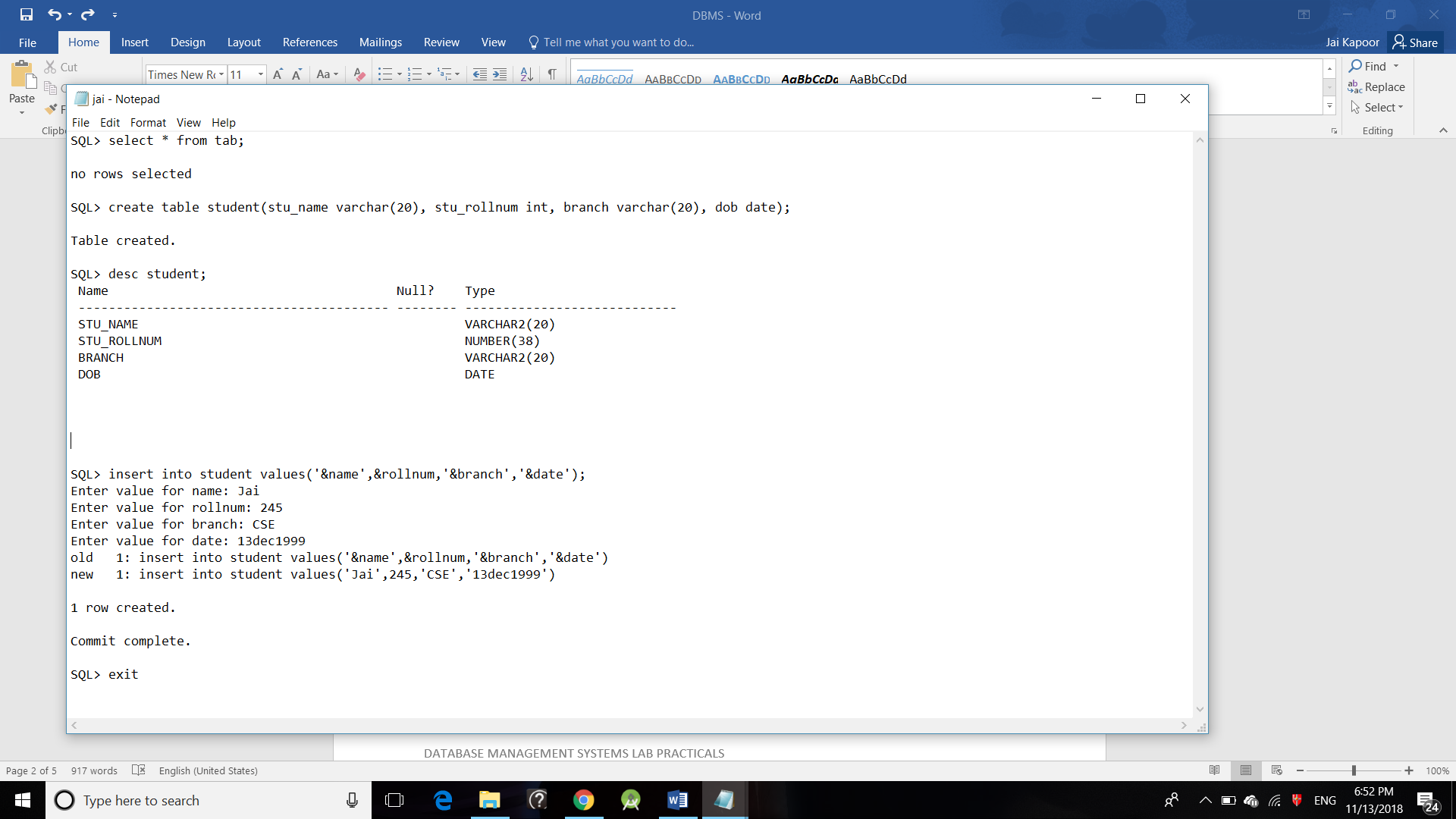
For example, **int** for integer data.

**size**: Size of the data we can store in a particular column. For example, if for

a column we specify the data\_type as int and size as 10 then this column can store an integer

number of maximum 10 digits.

***Implementation in ORACLE SQL+ :***



***B. DROP Command:***

DROP is used to delete a whole database or just a table. The DROP statement destroys the objects like an existing database, table, index or view. A DROP statement in SQL removes a component from a relational database management system (RDBMS).

**Syntax:**

**DROP object object\_name**

**Examples:**

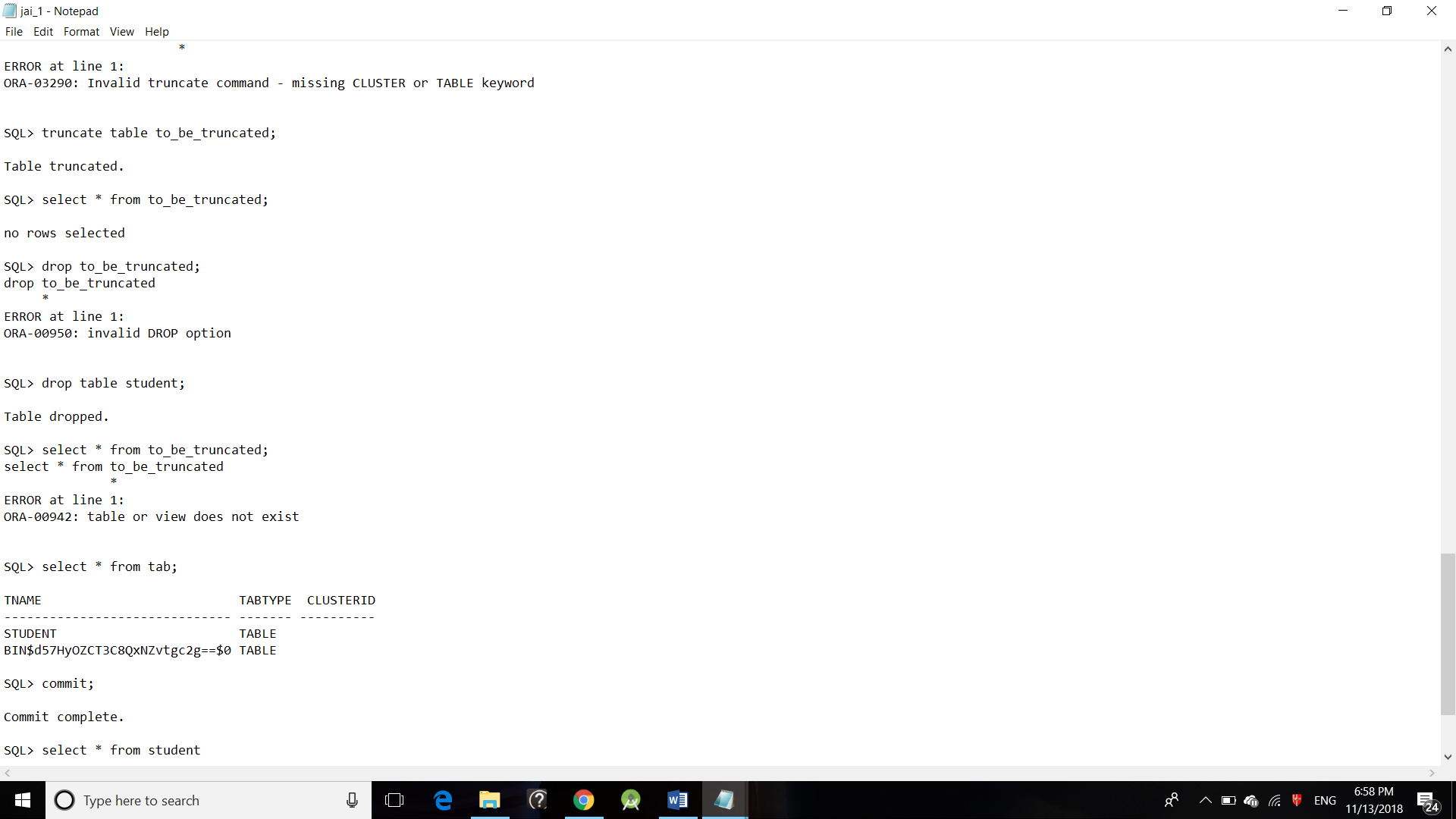
**DROP TABLE table\_name;**

**table\_name**: Name of the table to be deleted.

**DROP DATABASE database\_name;**

**database\_name**: Name of the database to be deleted.

***Implementation in ORACLE SQL+ :***



***C. TRUNCATE Command:***

TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extents of a table for deallocation (empty for reuse). The result of this operation quickly removes all data from a table, typically bypassing a number of integrity enforcing mechanisms. It was officially introduced in the SQL:2008 standard.

The TRUNCATE TABLE mytable statement is logically (though not physically) equivalent to the DELETE FROM mytable statement (without a WHERE clause).

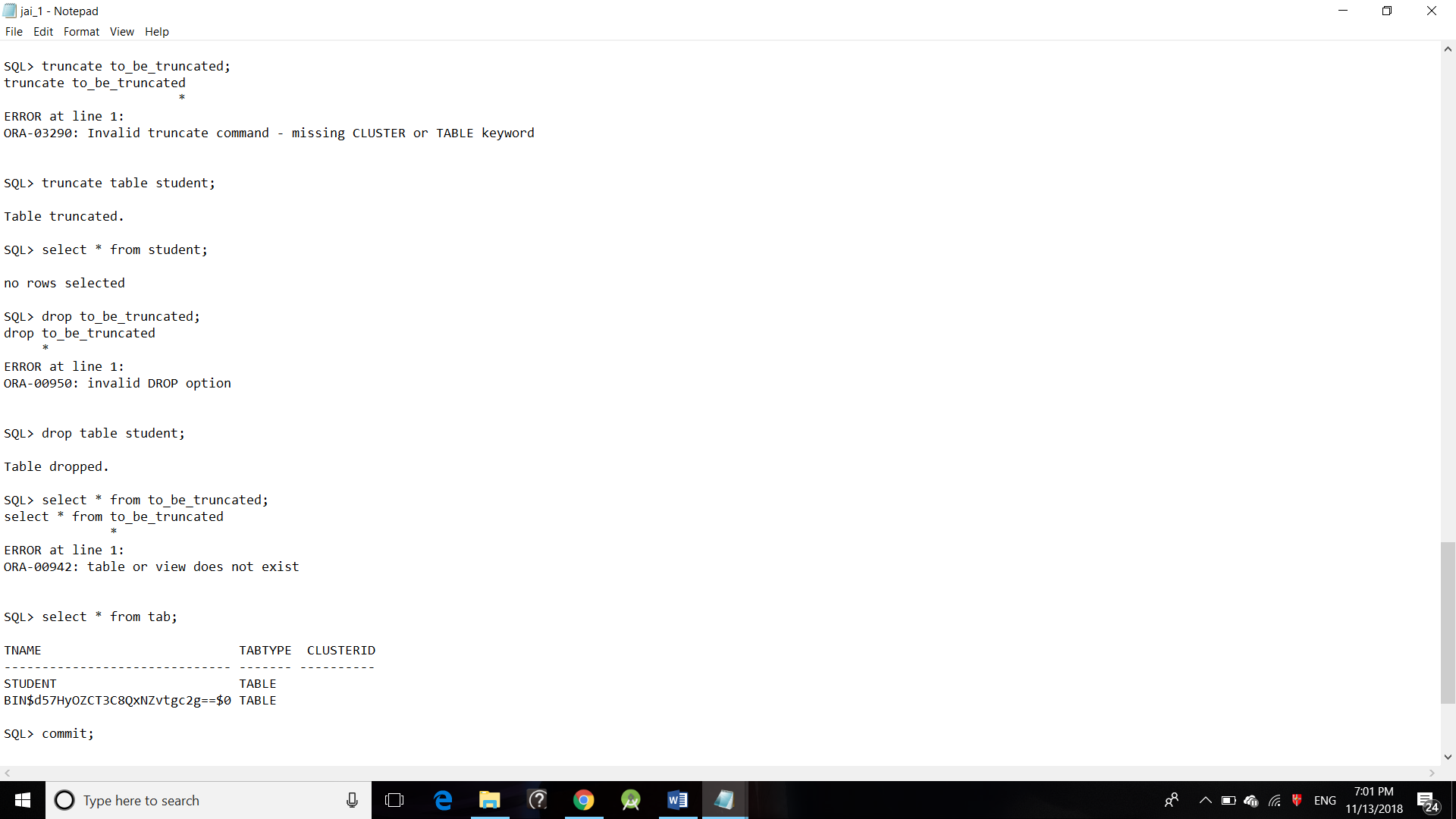
**Syntax:**

**TRUNCATE TABLE table\_name;**

**table\_name**: Name of the table to be truncated.

**DATABASE name - student\_data**

***Implementation in ORACLE SQL+ :***



***D. ALTER Command:***

ALTER TABLE is used to add, delete/drop or modify columns in the existing table. It is also used to add and drop various constraints on the existing table.

**ALTER TABLE – ADD**

ADD is used to add columns into the existing table. Sometimes we may require to add additional information, in that case we do not require to create the whole database again, **ADD** comes to our rescue.

**Syntax:**

**ALTER TABLE table\_name**

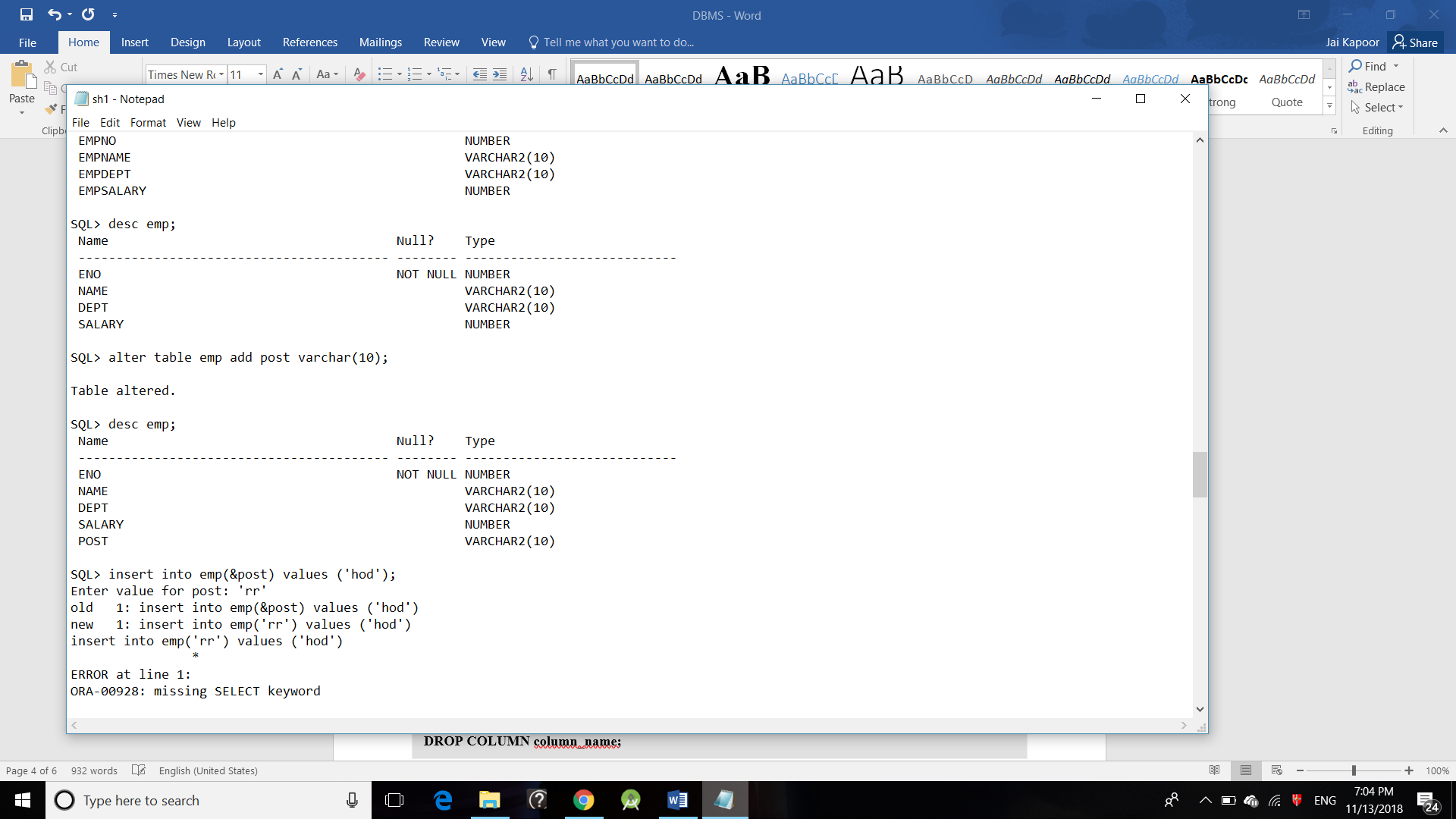
**ADD (Columnname\_1 datatype,**

**Columnname\_2 datatype,**

**…**

**Columnname\_n datatype);**

***Implementation in ORACLE SQL+ :***



**ALTER TABLE – DROP**

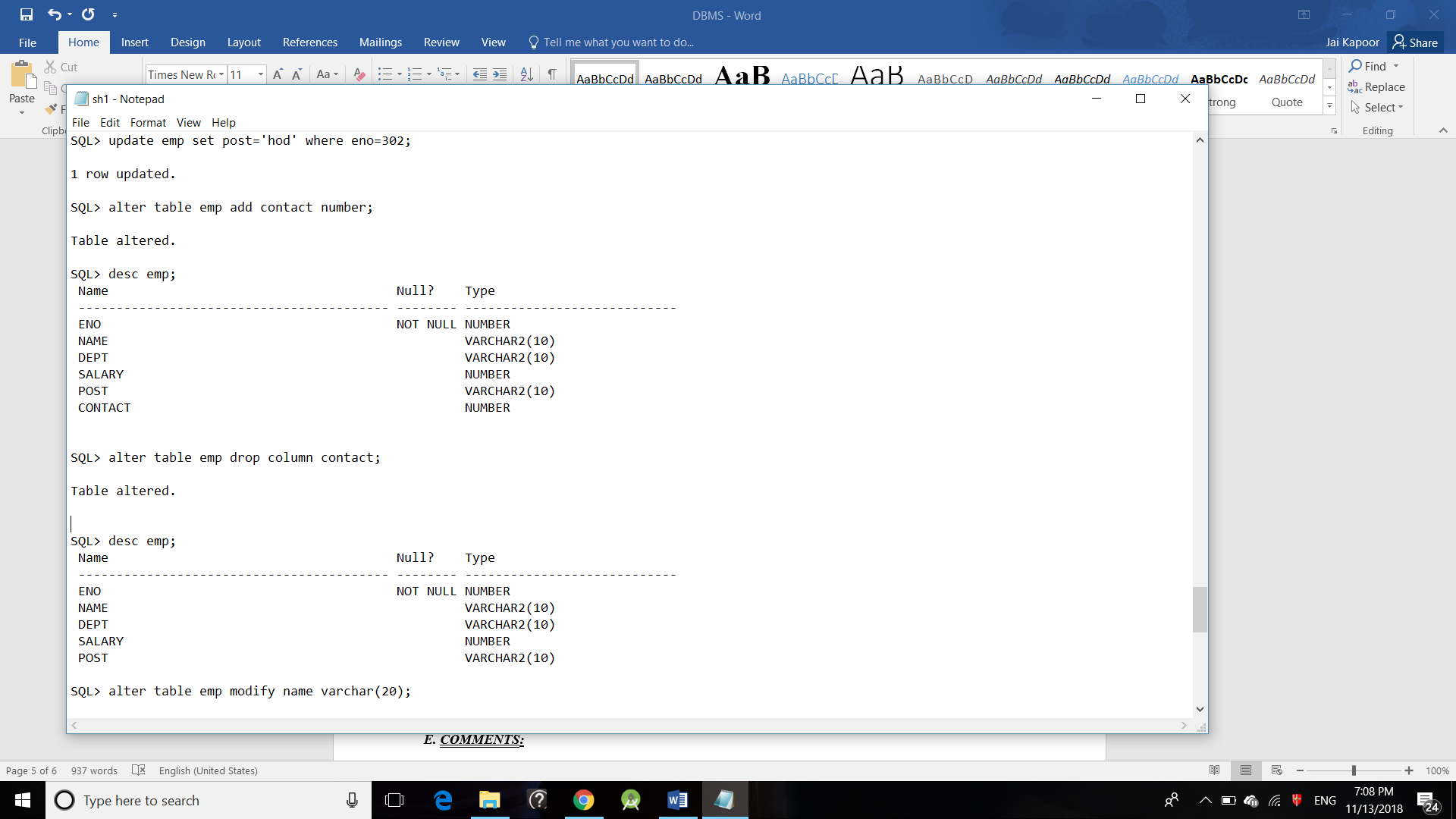
DROP COLUMN is used to drop column in a table. Deleting the unwanted columns from the table.

**Syntax:**

**ALTER TABLE table\_name**

**DROP COLUMN column\_name;**

***Implementation in ORACLE SQL+ :***



**ALTER TABLE-MODIFY**

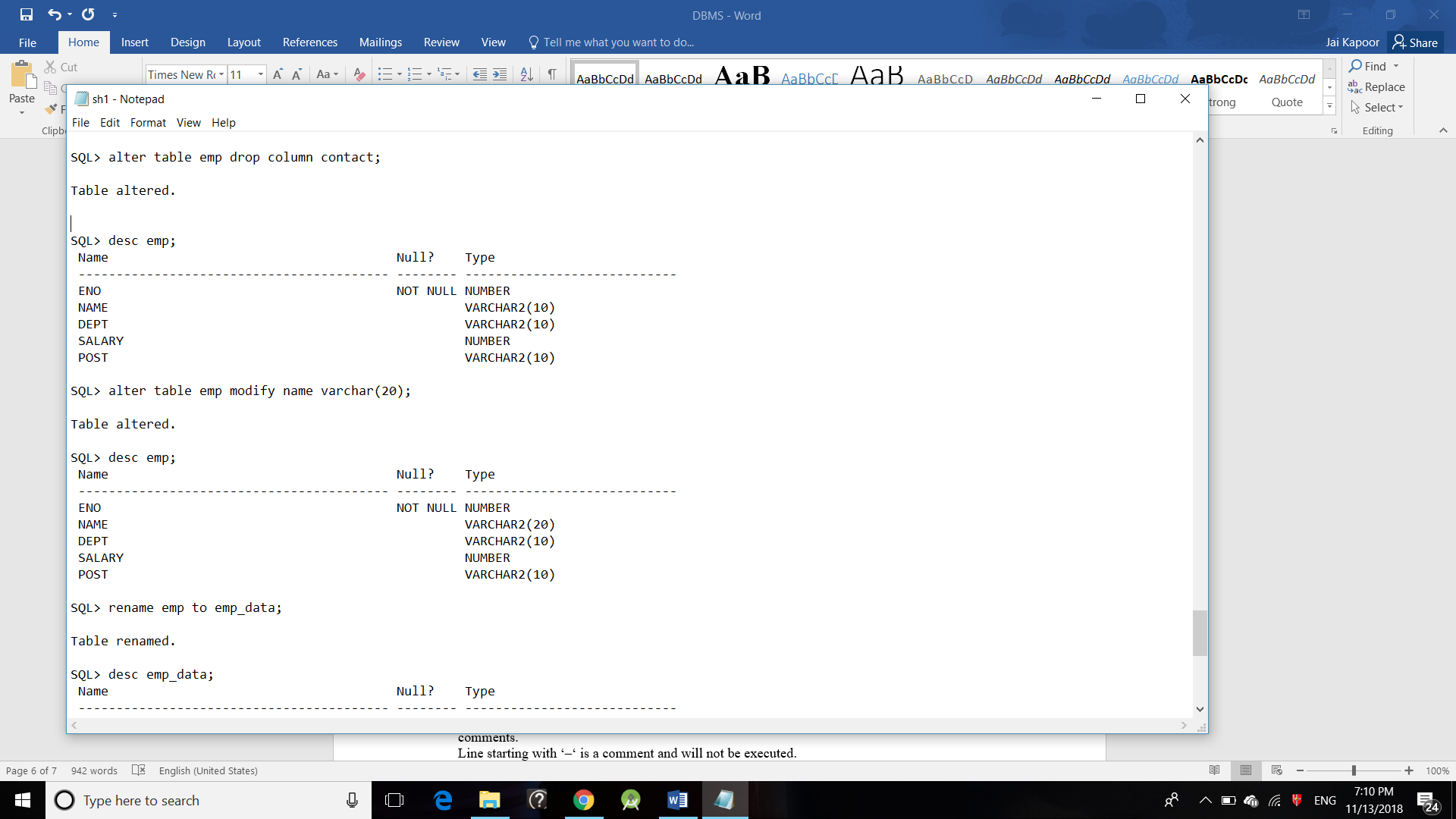
It is used to modify the existing columns in a table. Multiple columns can also be modified at once.

**Syntax:**

**ALTER TABLE table\_name**

**MODIFY column\_name column\_type;**

***Implementation in ORACLE SQL+ :***



# *E. COMMENTS:*

As is any programming languages comments matter a lot in SQL also.

Comments can be written in the following three formats:

1. Single line comments.
2. Multi line comments
3. In line comments

* **Single line comments:** Comments starting and ending in a single line are considered as single line comments.  
  Line starting with ‘–‘ is a comment and will not be executed.  
  Syntax:
* --single line comment
* --another comment
* SELECT \* FROM Customers;
* **Multi line comments:**Comments starting in one line and ending in different line are considered as multi line comments. Line starting with ‘/\*’ is considered as starting point of comment and are terminated when ‘\*/’ is encountered.  
  Syntax:

/\* multi line comment

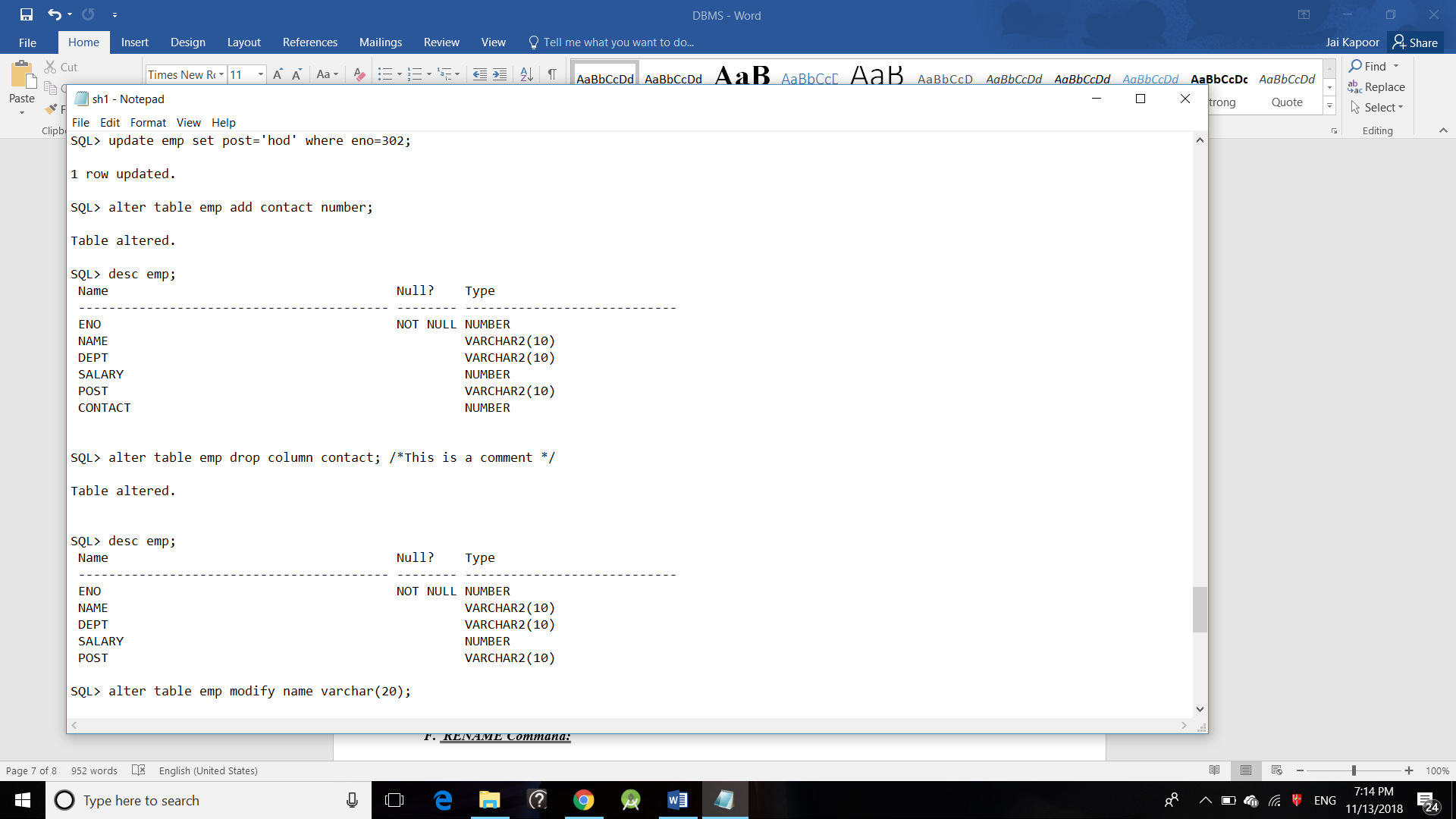
another comment \*/

SELECT \* FROM Customers;

* **In line comments:**In line comments are an extension of multi-line comments, comments can be stated in between the statements and are enclosed in between ‘/\*’ and ‘\*/’.  
  Syntax:

SELECT \* FROM /\* Customers; \*/

***Implementation in ORACLE SQL+ :***



***F. RENAME Command:***

Sometimes we may want to rename our table to give it a more relevant name. For this purpose, we can use **ALTER TABLE** to rename the name of table.

**Syntax:**

**ALTER TABLE table\_name**

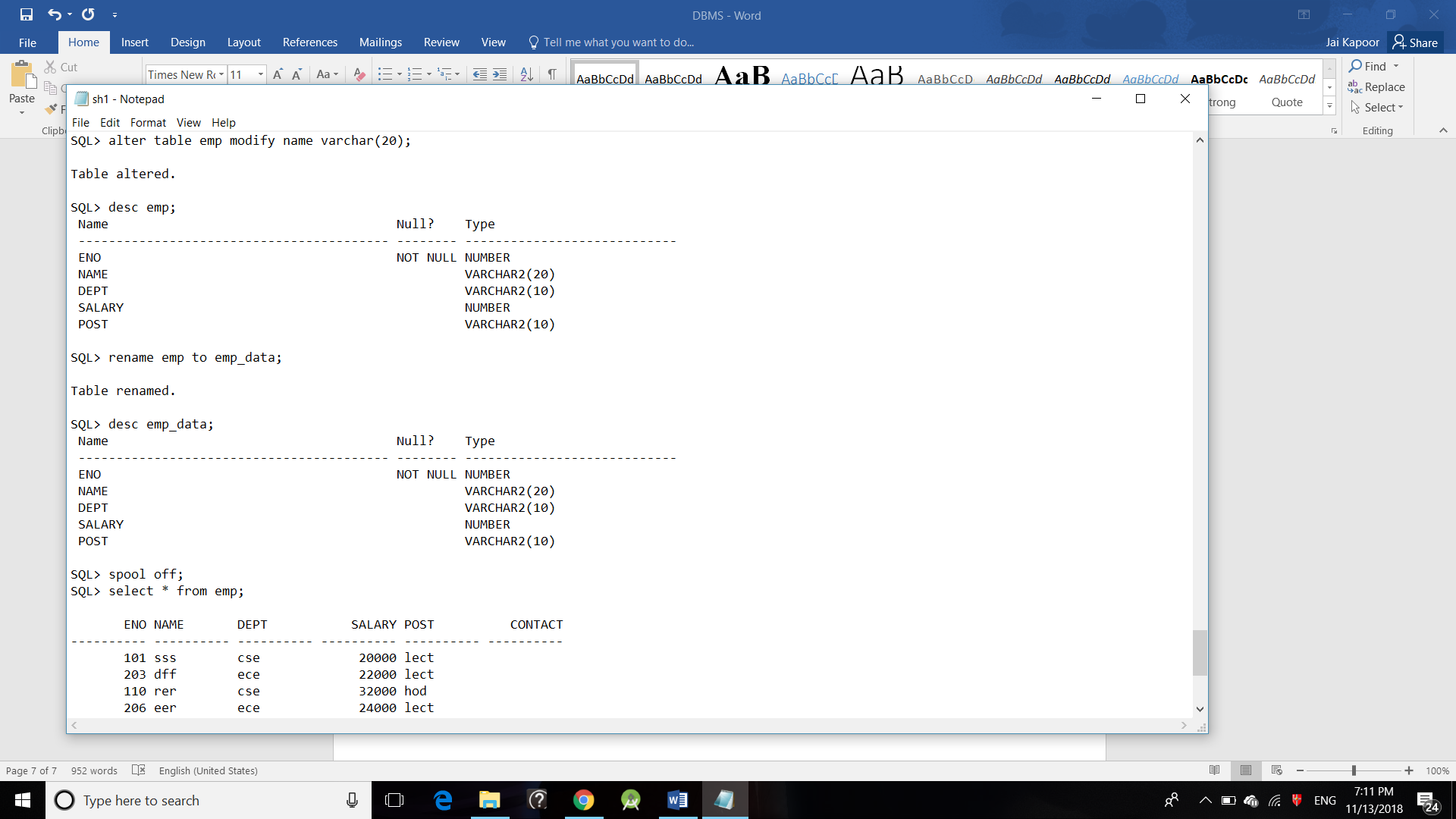
**RENAME TO new\_table\_name;**

Columns can be also be given new name with the use of **ALTER TABLE**.  
**Syntax:**

**ALTER TABLE table\_name**

**RENAME COLUMN old\_name TO new\_name;**

***Implementation in ORACLE SQL+ :***



**Practical – 2**

***Aim –*** *Write the queries for Data Manipulation Language in Relational Database Management Systems.*

***Data Manipulation Language(DML)*:**The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

**Examples of DML:**

* [**SELECT**](https://www.geeksforgeeks.org/sql-select-clause/) – is used to retrieve data from the a database.
* [**INSERT**](https://www.geeksforgeeks.org/sql-insert-statement/) – is used to insert data into a table.
* [**UPDATE**](https://www.geeksforgeeks.org/sql-update-statement/) – is used to update existing data within a table.
* [**DELETE**](https://www.geeksforgeeks.org/sql-delete-statement/) – is used to delete records from a database table.

1. ***SELECT Command:*** The SELECT Statement in SQL is used to retrieve or fetch data from a database. We can fetch either the entire table or according to some specified rules. The data returned is stored in a result table. This result table is also called result-set. With the SELECT clause of a SELECT command statement, we specify the columns that we want to be displayed in the query result and, optionally, which column headings we prefer to see above the result table.

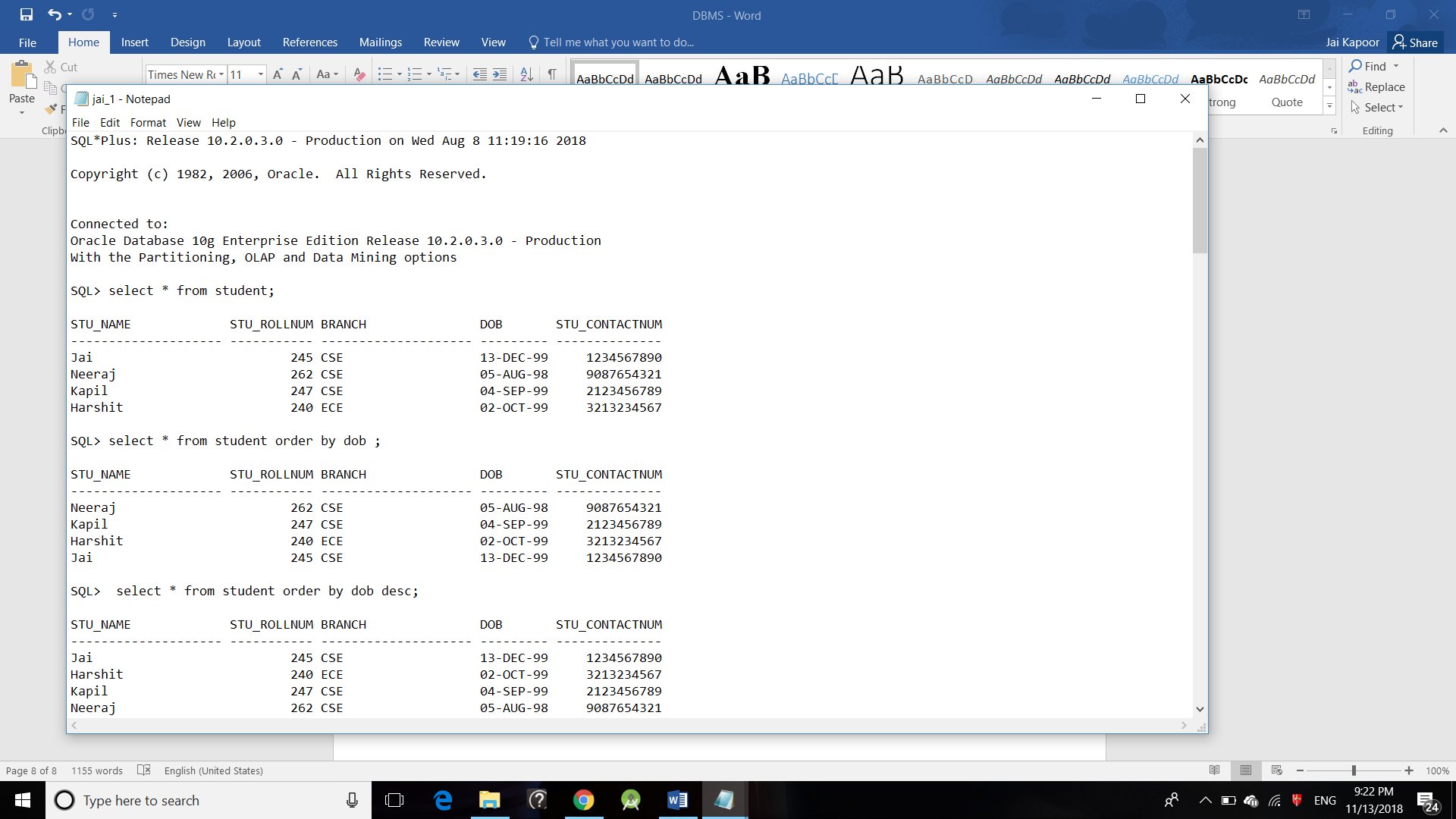
**Syntax:**

**SELECT column1,column2 FROM table\_name**

**column1 , column2**: names of the fields of the table

**table\_name:** from where we want to fetch

***Implementation in ORACLE SQL+ :***



1. ***INSERT Command:*** The INSERT INTO statement of SQL is used to insert a new row in a table. There are two ways of using INSERT INTO statement for inserting rows:
   1. **Only values:** First method is to specify only the value of data to be inserted without the column names.  
      **Syntax:**

**INSERT INTO table\_name VALUES (value1, value2, value3,...);**

**table\_name**: name of the table.

**value1, value2,..** : value of first column, second column,... for the new record

* 1. **Column names and values both:** In the second method we will specify both the columns which we want to fill and their corresponding values as shown below:  
     **Syntax:**

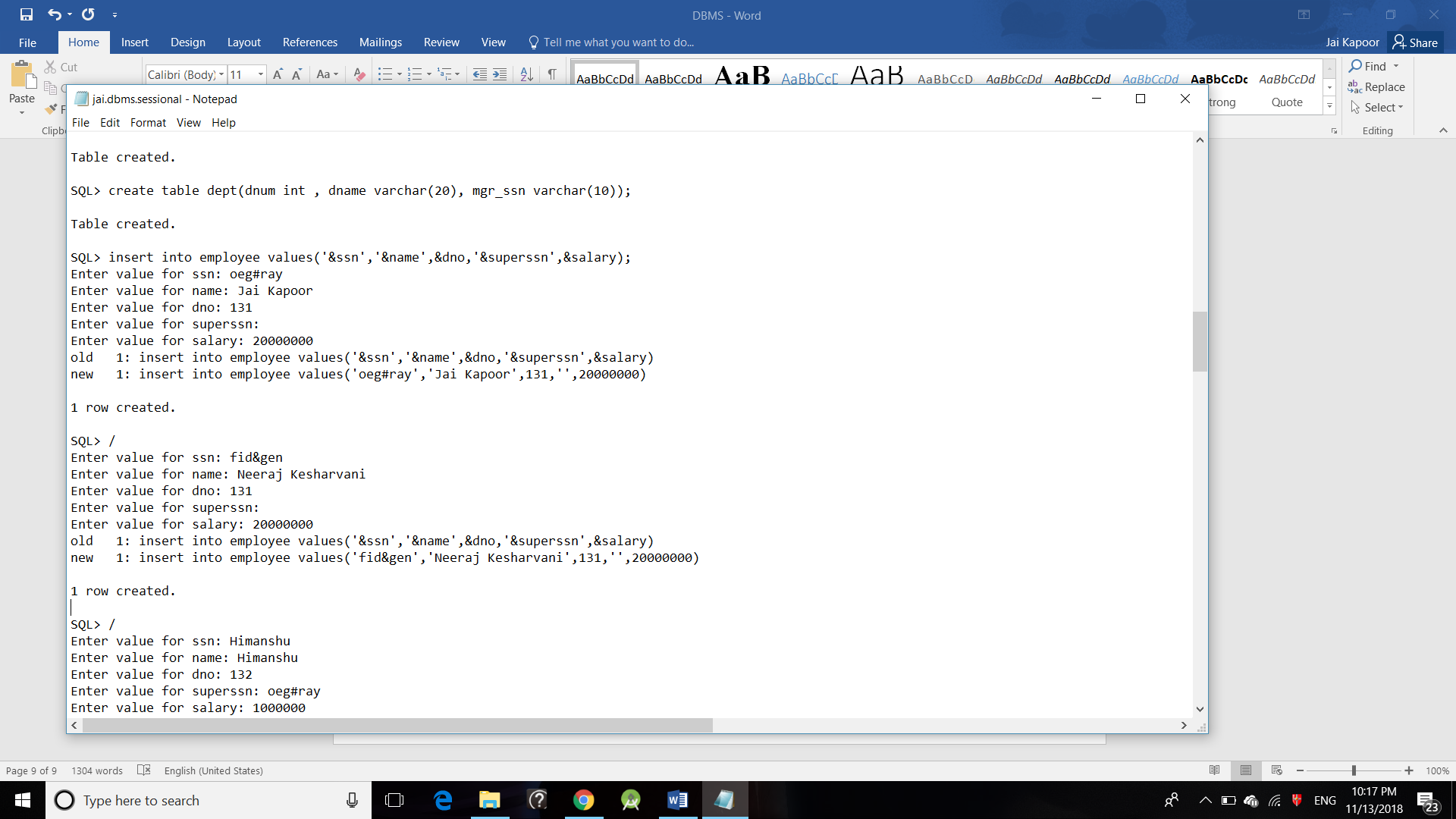
**INSERT INTO table\_name (column1, column2, column3,..) VALUES ( value1, value2, value3,..);**

**table\_name**: name of the table.

**column1**: name of first column, second column ...

**value1, value2, value3** : value of first column, second column,... for the new record

***Implementation in ORACLE SQL+ :***



1. ***UPDATE Command:*** The UPDATE statement in SQL is used to update the data of an existing table in database. We can update single columns as well as multiple columns using UPDATE statement as per our requirement.

**Syntax:**

**UPDATE table\_name SET column1 = value1, column2 = value2,...**

**WHERE condition;**

**table\_name:** name of the table

**column1**: name of first , second, third column....

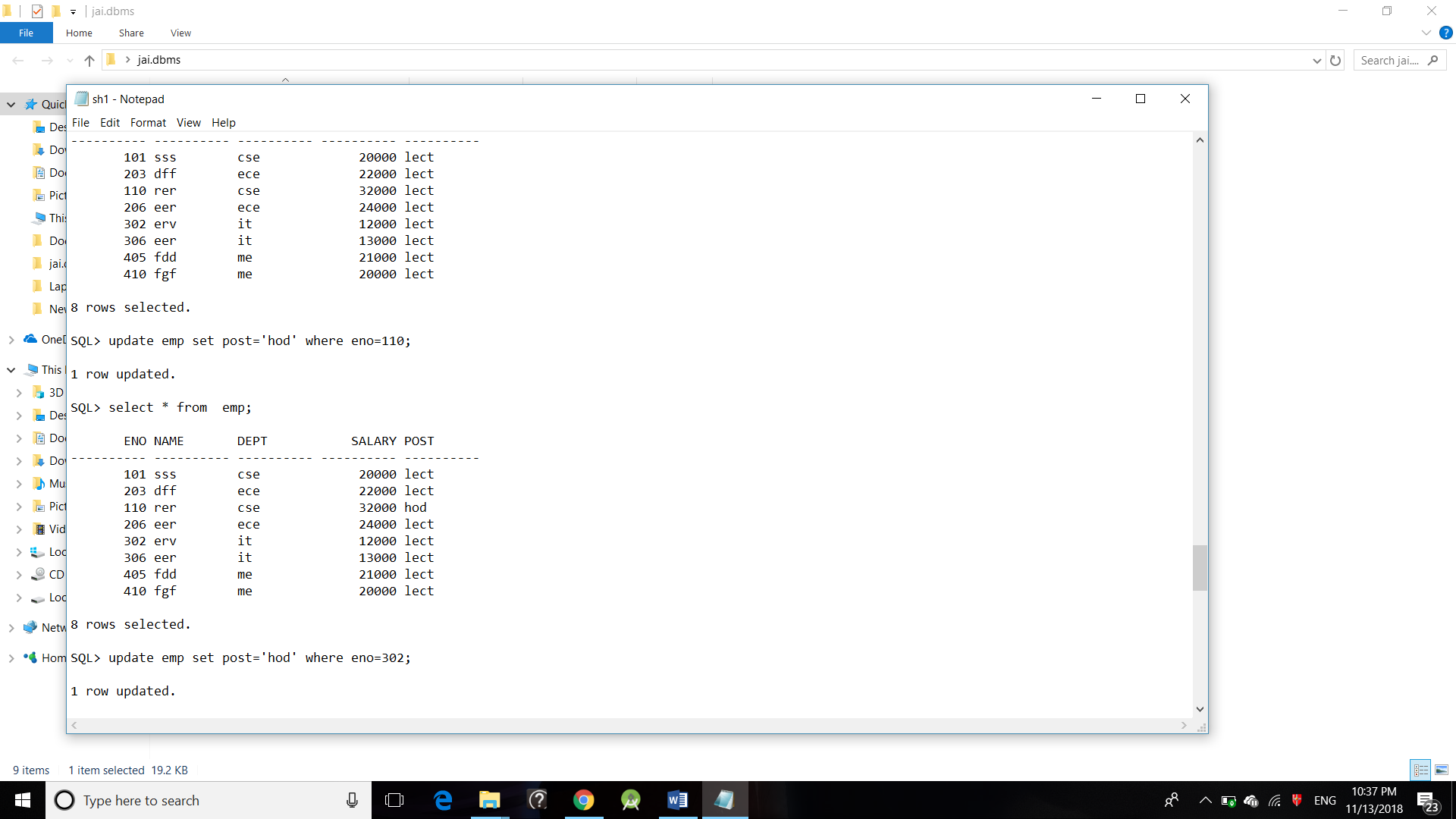
**value1**: new value for first, second, third column....

**condition**: condition to select the rows for which the

values of columns needs to be updated.

**NOTE:** In the above query the**SET**statement is used to set new values to the particular column and the **WHERE** clause is used to select the rows for which the columns are needed to be updated. If we have not used the WHERE clause then the columns in **all** the rows will be updated. So the WHERE clause is used to choose the particular rows.

***Implementation in ORACLE SQL+ :***



1. ***DELETE Command:*** The DELETE Statement in SQL is used to delete existing records from a table. We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.

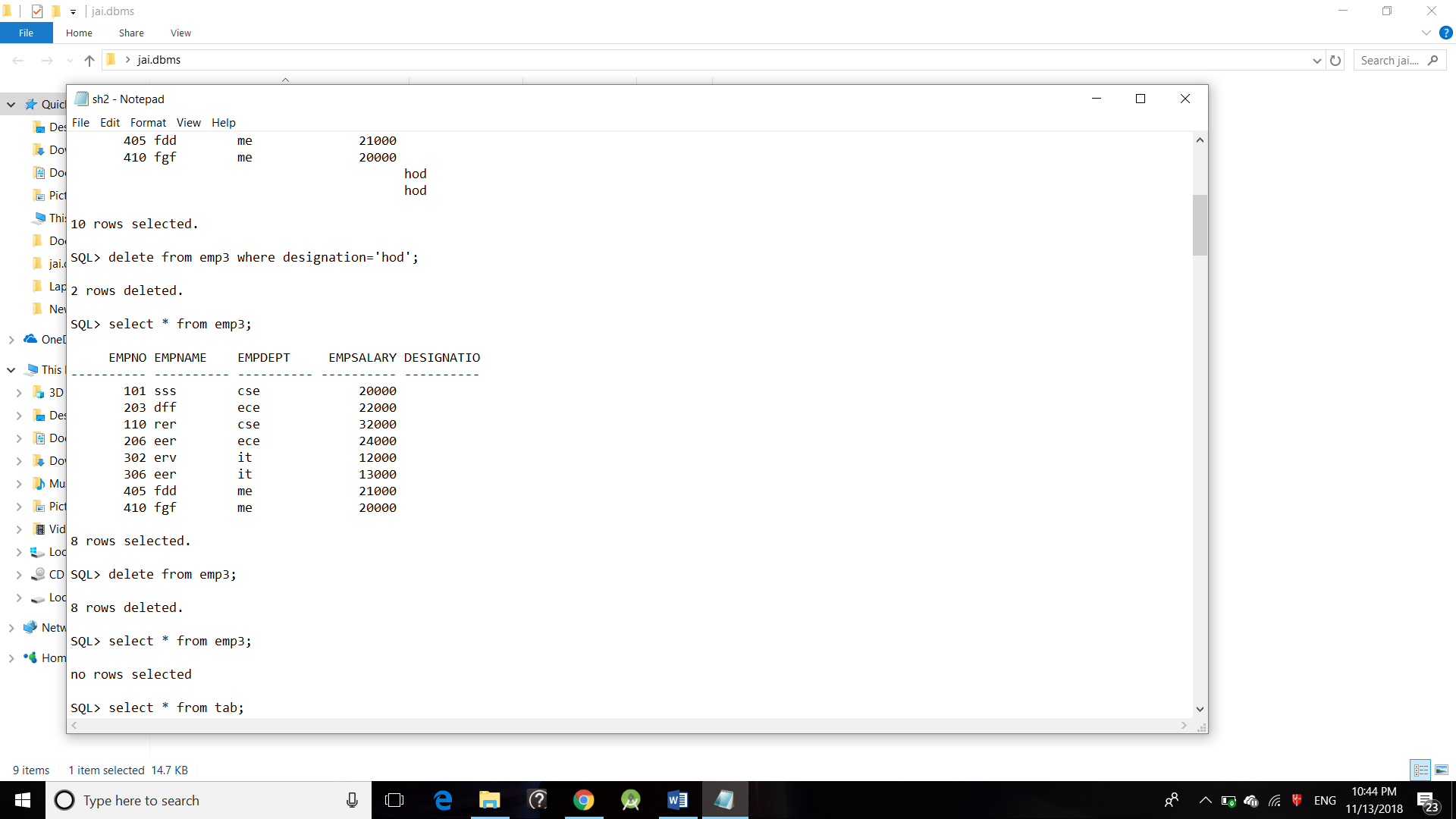
**Syntax:**

DELETE FROM table\_name WHERE some\_condition;

**table\_name**: name of the table

**some\_condition**: condition to choose particular record.

***Implementation in ORACLE SQL+ :***



**Practical – 3**

***Aim –*** *Write the queries for Data Control Language in Relational Database Management Systems.*

***Data Control language (DCL):*** DCL includes commands such as GRANT and REVOKE which mainly deals with the 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.

1. **GRANT command:**

In order to do anything within an Oracle database you must be given the appropriate privileges. Oracle operates a closed system in that you cannot perform any action at all unless you have been authorized to do so. This includes logging onto the database, creating tables, views, indexes and synonyms, manipulating data (i.e. select, insert, update and delete) in tables created by other users, etc.

The SQL command to grant a privilege on a table is:

**GRANT SELECT, INSERT, UPDATE, DELETE ON tablename TO username;**

1. **REVOKE command:**

The SQL command to revoke a privilege on a table is:

REVOKE SELECT, INSERT, UPDATE, DELETE ON tablename FROM username;

For example:

**REVOKE SELECT ON employee FROM hn23;**

**REVOKE SELECT, UPDATE, DELETE FROM hn44;**

**Practical – 4**

***Aim –*** *Write the queries for Logical operations.*

SQL> select \* from stud;

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

121722324 Yamato CSE 89

121722322 Venom CSE 96

SQL> select \* from stud where name like 'y%';

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

121722324 Yamato CSE 89

SQL> select \* from stud where name not like 'y%';

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

121722322 Venom CSE 96

SQL> select \* from stud where name like '\_j%';

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

121722322 Venom CSE 96

SQL> select name||branch from stud;

NAME||BRANCH

--------------------

YamatoCSE

VenomCSE

SQL> select \* from stud where name like 'y%' or marks>90;

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

121722324 Yamato CSE 89

121722322 Venom CSE 96

SQL> select \* from stud where rollno!=121722322;

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

121722324 Yamato CSE 89

SQL> select name ||' '||branch from stud;

NAME||''||BRANCH

---------------------

Yamato CSE

Venom CSE

SQL>spool off;

*.*

**Practical – 5**

***Aim –*** *Write the queries for Arithmetic operations.*

SQL> select \* from stud;

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

121722324 Yamato cse 89

121722322 Venom cse 96

SQL> select name,marks,markns+20 from stud;

NAME MARKS MARKS+20

---------- ---------- ----------

Yamato 89 109

Venom 96 116

SQL> select name,marks,marks-44 from stud;

NAME MARKS MARKS-44

---------- ---------- ----------

Yamato 89 45

Venom 96 52

SQL> select name,marks,marks/5 from stud;

NAME MARKS MARKS/5

---------- ---------- ----------

Yamato 89 17.8

Venom 96 19.2

SQL> create table m (a number(10),b number(20));

Table created.

SQL> insert into m values (12,4);

1 row created.

SQL> insert into m values (67,32);

1 row created.

SQL> insert into m values (21,7);

1 row created.

SQL>insert into m values (97,23);

1 row created.

SQL> select \* from m;

A B

---------- ----------

12 4

67 32

21 7

97 23

SQL> select a,b,mod(a,b) rem from m;

A B REM

---------- ---------- ----------

12 4 0

67 32 3

21 7 0

97 23 5

SQL>spool off;

**Practical – 6**

***Aim –*** *Write the queries using character, date, number and group functions.*

SQL> select \* from employee;

EMPNO ENAME DESIGNATIO SALARY DEPTNO

---------- ---------- ---------- ---------- ----------

310 viraj radaur 20000 48

325 rana karnal 3000 47

SQL> select concat(ename,salary) from employee;

CONCAT(ENAME,SALARY)

--------------------------------------------------

viraj20000

rana3000

SQL> select substr(ename,1,3) from employee;

SUB

---

pav

par

SQL> select instr(ename,'v') from employee;

INSTR(ENAME,'V')

----------------

3

0

SQL> select rpad(ename,10,'\_') from employee;

RPAD(ENAME

----------

viraj\_\_\_

rana\_\_\_\_\_

SQL> select lpad(ename,10,'\_') from employee;

LPAD(ENAME

----------

\_\_\_viraj

\_\_\_\_\_rana

SQL> select rtrim('avn') as ename from employee;

ENA

---

avn

avn

SQL> select upper(ename) from employee;

UPPER(ENAME)

----------

VIRAJ

RANA

SQL> select abs(salary) from employee;

ABS(SALARY)

-----------

20000

3000

SQL> select ceil(salary) from employee;

CEIL(SALARY)

------------

20000

3000

SQL> select mod(salary,deptno) from employee;

MOD(SALARY,DEPTNO)

------------------

32

39

SQL> select sign(salary) from employee;

SIGN(SALARY)

------------

1

1

SQL> select sqrt(salary) from employee;

SQRT(SALARY)

------------

141.421356

54.7722558

SQL> select add\_months('13-oct-16',3) from employee;

ADD\_MONTH

---------

13-JAN-17

13-JAN-17

SQL> select count(\*) from employee;

COUNT(\*)

----------

2

SQL> select distinct empno from employee;

EMPNO

----------

325

310

SQL> select max(salary) from employee;

MAX(SALARY)

-----------

20000

SQL> select sum(salary) from employee;

SUM(SALARY)

-----------

23000

**Practical – 7**

***Aim –*** *Write the queries for non-equi join, and view formation.*

SQL> select \* from stud;

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

121 yash cse 89

111 ajay cse 96

123 vijay it 43

SQL> create table mks (minmks number(10) ,maxmks number(10));

Table created.

SQL> insert into mks values(&minmks,&maxmks);

Enter value for minmks: 21

Enter value for maxmks: 99

old 1: insert into mks values(&minmks,&maxmks)

new 1: insert into mks values(21,99)

1 row created.

SQL> /

Enter value for minmks: 90

Enter value for maxmks: 100

old 1: insert into mks values(&minmks,&maxmks)

new 1: insert into mks values(90,100)

1 row created.

SQL> /

Enter value for minmks: 21

Enter value for maxmks: 33

old 1: insert into mks values(&minmks,&maxmks)

new 1: insert into mks values(21,33)

1 row created.

SQL> select \* from mks;

MINMKS MAXMKS

---------- ----------

21 99

90 100

21 33

SQL> create view v34 as select stud.name,mks.maxmks from stud,mks where stud.marks between mks.minmks and mks.maxmks;

View created.

SQL> select \* from v34;

NAME MAXMKS

---------- ----------

ajay 100

ajay 99

yash 99

vijay 99

SQL> spool off;

**Practical – 8**

***Aim –*** *Write the queries for sub queries and nested queries.*

SQL> select \* from stud;

NAME ROLLNO MOBILE MARKS

---------- ---------- ---------- ----------

yogesh 324 987766576 95

vishal 322 859656345 76

sagar 291 986756453 77

sushant 310 858578698 87

SQL> select \* from stud where rollno in (select rollno from stud where marks>85);

NAME ROLLNO MOBILE MARKS

---------- ---------- ---------- ----------

yogesh 324 987766576 95

sushant 310 58578698 87

SQL> create table stud\_bck (name varchar(10), rollno number(10), mobile number(10), marks number(5));

Table created.

SQL> insert into stud\_bck select \* from stud where rollno in (select rollno from stud);

4 rows created.

SQL> select \* from stud\_bck;

NAME ROLLNO MOBILE MARKS

---------- ---------- ---------- ----------

yogesh 324 987766576 95

vishal 322 859656345 76

sagar 291 986756453 77

sushant 310 858578698 87

SQL> update stud set marks=marks-5 where rollno in(select rollno from stud\_bck where rollno>239);

3 rows updated.

SQL> select \* from stud\_bck;

NAME ROLLNO MOBILE MARKS

---------- ---------- ---------- ----------

yogesh 324 987766576 95

vishal 322 859656345 76

sagar 291 986756453 77

sushant 310 858578698 87

SQL> select \* from stud;

NAME ROLLNO MOBILE MARKS

---------- ---------- ---------- ----------

yogesh 324 987766576 90

vishal 322 859656345 76

sagar 291 986756453 72

sushant 310 858578698 82

SQL> delete from stud where rollno in (select rollno from stud\_bck where rollno>=324);

3 rows deleted.

SQL> select \* from stud;

NAME ROLLNO MOBILE MARKS

---------- ---------- ---------- ----------

vishal 322 59656345 76

SQL> select \* from stud\_bck;

NAME ROLLNO MOBILE MARKS

---------- ---------- ---------- ----------

yogesh 324 987766576 95

vishal 322 859656345 76

sagar 291 986756453 77

sushant 310 858578698 87

**Practical – 9**

***Aim –*** *Write the queries for roll back, commit, savepoint.*

SQL> create table com (name varchar(10), roll number(10));

Table created.

SQL> insert into com values ('&name',&roll);

Enter value for name: sushant

Enter value for roll: 1217310

old 1: insert into com values('&name',&roll)

new 1: insert into com values('sushant',1217310)

1 row created.

SQL> /

Enter value for name: sagar

Enter value for roll: 1217291

old 1: insert into com values('&name',&roll)

new 1: insert into com values('sagar',1217291)

1 row created.

SQL> insert into com values ('&name',&roll);

Enter value for name: yogesh

Enter value for roll: 1217324

old 1: insert into com values('&name',&roll)

new 1: insert into com values('yogesh',1217324)

1 row created.

SQL> select \* from com;

NAME ROLL

---------- ----------

sushant 1

sagar 1217291

b 1217324

SQL> savepoint a1;

Savepoint created.

SQL> insert into com values(‘ss’,1234);

1 row created.

SQL> select \* from com;

NAME ROLL

---------- ----------

sushant 1

sagar 1217291

yogesh 1217324

ss 1234

SQL> rollback to a1;

Rollback complete.

SQL> select \* from com;

NAME ROLL

---------- ----------

sushant 1

sagar 1217291

yogesh 1217324

SQL> commit ;

Commit complete.

**Practical – 10**

***Aim –*** *Write the queries for equi-join and different view formation.*

SQL> select \* from stud;

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

1217324 yogesh cse 89

1217310 sushant cse 96

SQL> insert into stud values (1217291,'sagar','it',43);

1 row created.

SQL> select \* from stud;

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

1217324 yogesh cse 89

1217310 sushant cse 96

1217291 sagar it 43

SQL> create table emp1 (empid number(10),dept varchar(10));

Table created.

SQL> insert into emp1 values (&empid,'&dept');

Enter value for empid: 1217324

Enter value for dept: abc

old 1: insert into emp1 values(&empid,'&dept')

new 1: insert into emp1 values(1217324,'abc')

1 row created.

SQL> /

Enter value for empid: 1217310

Enter value for dept: hhh

old 1: insert into emp1 values(&empid,'&dept')

new 1: insert into emp1 values(1217310,'hhh')

1 row created.

SQL> /

Enter value for empid: 1217291

Enter value for dept: njh

old 1: insert into emp1 values(&empid,'&dept')

new 1: insert into emp1 values(1217291,'njh')

1 row created.

SQL> select \* from emp1;

EMPID DEPT

---------- ----------

1217 abc

1218 hhh

1219 njh

SQL> select stud.rollno,stud.name,stud.marks,emp1.dept from stud,emp1 where stud.rollno=emp1.empid;

ROLLNO NAME MARKS DEPT

---------- ---------- ---------- ----------

1217324 yogesh 89 abc

1217310 sushant 96 hhh

1217291 sagar 43 njh

SQL> create view V as select name,branch from stud;

View created.

SQL> select \* from V;

NAME BRANCH

---------- ----------

yogesh cse

sushant cse

sagar it

SQL> create view v22 as select stud.rollno,stud.name,emp1.dept from stud,emp1 where branch='cse';

View created.

SQL> select \* from v22;

ROLLNO NAME DEPT

---------- ---------- ----------

1217324 yogesh abc

1217324 yogesh hhh

1217324 yogesh njh

1217310 sushant abc

1217310 sushant hhh

1217310 sushant njh

6 rows selected.

SQL> create view v23 as select stud.name,emp1.dept from stud,emp1 where stud.rollno=emp1.empid;

View created.

SQL> select \* from v23;

NAME DEPT

---------- ----------

yogesh abc

sushant hhh

sagar njh

SQL> spool off;

**Practical – 11**

***Aim –*** *Write the queries for procedure of triggering any table.*

SQL> create table q1 (name varchar(10),salary number(10));

Table created.

SQL> create table q2 (name varchar(10),salary number(10));

Table created.

SQL> create trigger q3 after update or delete on q1 for each row

2 declare

3 oper varchar(10);

4 name varchar(10);

5 salary number(10);

6 begin

7 if updating then oper:='update';

8 end if;

9 if deleting then oper:='delete';

10 end if;

11 name:=:old.name;

12 salary:=:old.salary;

13 insert into q2 values(name,salary);

14 end;

15 /

Trigger created.

SQL> insert into q1 values('yogesh',3450);

1 row created.

SQL> insert into q1 values('sushant',1230);

1 row created.

SQL> select \* from q1;

NAME SALARY

---------- ----------

yogesh 3450

sushant 1230

SQL> update q1 set salary=8000 where name='yogesh';

1 row updated.

SQL> select \* from q2;

NAME SALARY

---------- ----------

yogesh 3450

SQL> select \* from q1;

NAME SALARY

---------- ----------

yogesh 8000

sushant 1230

**Practical – 12**

***Aim –*** *Write the queries to make procedure to find area of circle and insert it into a table.*

SQL> create table ar(radius number(10),area number(12));

Table created.

SQL> create or replace procedure pro(radius in number) is

2 r number(10);

3 area number(10);

4 begin

5 r:=radius;

6 area:=3.14\*r\*r;

7 insert into ar values(radius,area);

8 end;

9 /

Procedure created.

SQL> execute pro(7);

PL/SQL procedure successfully completed.

SQL> select \* from ar;

RADIUS AREA

---------- ----------

7 154

SQL> execute pro(3);

PL/SQL procedure successfully completed.

SQL> select \* from ar;

RADIUS AREA

---------- ----------

7 154

3 28

SQL> spool off;

**Practical – 13**

***Aim –*** *Write the queries to make procedure to find for computing the amount of telephone bill.*

SQL> create table tbill(name varchar(10),phone\_no number(10),units number(10),bill number(10,2));

Table created.

SQL> create or replace procedure tbil(name in varchar,phone\_no in number, units in number) is

2 unit number(10);

3 bill number(10,2);

4 begin

5 bill:=205;

6 unit:=units-105;

7 if unit>50 and unit<=500 then

8 bill:=bill+ (unit\*0.8);

9 end if;

10 if unit>500 then

11 bill:=bill+ (unit\*1.2);

12 end if;

13 insert into tbill values(name,phone\_no,units,bill);

14 end;

15 /

Procedure created.

SQL> execute tbil('sushant',264778,400);

PL/SQL procedure successfully completed.

SQL> select \* from tbill;

NAME PHONENO UNITS BILL

---------- ---------- ---------- ----------

sushant 264778 400 559

SQL> spool off;

**Practical – 14**

***Aim –*** *Write the queries to create a trigger for before insertion or update.*

SQL> create table students1(name varchar2(20),roll\_no varchar2(9) not null primary key, balance varchar2(10));

Table created.

SQL> insert into students1 values('sushant','1217310','250');

1 row created.

SQL> insert into students1 values('yogesh','1217324','195');

1 row created.

SQL> insert into students1 values('sagar','1217291','400');

1 row created.

SQL> select \* from students1;

NAME ROLL\_NO BALANCE

-------------------- --------- ----------

sushant 1217310 250

yogesh 1217324 195

sagar 1217291 400

SQL> create or replace trigger tri before insert or update on students1 for each row

2 begin

3 if :new.balance<=0 then

4 raise\_application\_error(-20000,'salary can not be less than zero');

5 end if;

6 end;

7 /

Trigger created.

SQL> insert into students1 values

2 ('rohit','1207938','-256');

insert into students1 values

\*

ERROR at line 1:

ORA-20000: salary can not be less than zero

ORA-06512: at "SCOTT.TRI", line 3

ORA-04088: error during execution of trigger 'SCOTT.TRI'

SQL> commit;

Commit complete.

**Practical – 15**

***Aim –*** *Write the queries to demonstrate Referential Integrity (Primary and Foreign key).*

SQL> create table stud (rollno number(10),name varchar(10),branch varchar(10),marks number(20),

constraint RN primary key(rollno),constraint MKS check (marks>27));

Table created.

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

Enter value for rollno: 1217324

Enter value for name: yogesh

Enter value for branch: cse

Enter value for marks: 97

old 1: insert into stud values(&rollno,'&name','&branch',&marks)

new 1: insert into stud values(1217324 ,'yogesh','cse',97)

1 row created.

SQL> /

Enter value for rollno: 1217310

Enter value for name: sushant

Enter value for branch: cse

Enter value for marks: 89

1 row created.

SQL> select \* from stud;

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

1217324 yogesh cse 97

1217310 sushant cse 89

SQL> insert into stud values ('1217291','sagar','it',77);

insert into stud values('1217291','sagar','it',77)

\*

ERROR at line 1:

ORA-00001: unique constraint (ROCK.RN) violated

SQL> insert into stud values ('1217291','sagar','cse',9);

1 row created.

SQL> select \* from stud;

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

1217324 yogesh cse 97

1217310 sushant cse 89

1217291 sagar cse 9

SQL> alter table stud drop constraint RN;

Table altered.

SQL> insert into stud values ('1217324','deepak','cse',67);

1 row created.

SQL> select \* from stud;

ROLLNO NAME BRANCH MARKS

---------- ---------- ---------- ----------

1217324 yogesh cse 97

1217310 sushant cse 89

1217291 sagar cse 9

SQL> spool off;