**Ex.No: 1 SQL DATA DEFINITION LANGUAGE COMMANDS**

**Date:**

**AIM**:

To write a program for SQL Data Definition Language Commands on sample exercise.

**Data Definition Language [DDL]:**

* Used to define database structure or pattern.
* Used to create schema, tables, indexes, constraints
* Create skeleton of the database.
* Used to store information of metadata like no. of tables and schemas, their names, indexes, columns in each table, constraints

Some of the commands are:

1. Create
2. Alter
3. Drop
4. Truncate
5. Rename

**CREATE Command:**

* Used to create objects in database.

*Syntax:*

Basic syntax of CREATE TABLE statement is as follows:

CREATE TABLE table\_name(

column1 datatype,

column2 datatype,

column3 datatype,

.....

columnN datatype,

);

**ALTER Command:**

The SQL **ALTER TABLE** command is used to add, delete or modify columns in an existing table. We would also use ALTER TABLE command to add and drop various constraints on a an existing table.

*Syntax*

The basic syntax of **ALTER TABLE** to add a new column in an existing table is as follows:

ALTER TABLE table\_name ADD column\_name datatype;

The basic syntax of ALTER TABLE to **DROP COLUMN** in an existing table is as follows:

ALTER TABLE table\_name DROP COLUMN column\_name;

The basic syntax of ALTER TABLE to change the **DATA TYPE** of a column in a table is as follows:

ALTER TABLE table\_name MODIFY COLUMN column\_name datatype;

**DROP Command:**

The DROP TABLE statement is used to drop an existing table in a database.

*Syntax*

DROP TABLE table\_name;

**TRUNCATECommand:**

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

*Syntax*

TRUNCATE TABLE table\_name;

**RENAME Command:**

* Used to rename the table.

SQL offers two ways to rename tables.

The first one uses the ALTER TABLE syntax:

ALTER TABLE old\_table\_name RENAME new\_table\_name;

The second way is to use RENAME TABLE:

RENAME TABLE old\_table\_name TO new\_table\_name;

RENAME TABLE offers more flexibility. It allows renaming multiple tables in one statement. This can be useful when replacing a table with a new pre-populated version:

RENAME TABLE old\_table\_name1 TO new\_table\_name1, old\_table\_name2 TO new\_table\_name2;

**Result:**

Thus the query for SQL Data Definition Language Commands are written, executed and the outputs are verified successfully.

**Ex.No: 2 SQL DATA MANIPULATION LANGUAGE COMMANDS**

**Date:**

**AIM**:

To write a program for SQL Data Manipulation Language Commands on sample exercise.

**Data Manipulation Language:**

DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects.

Some of the commands are:

1. Insert

2. Select

3. Update

4. Delete

**Syntax :**

**INSERT:** This is used to add one or more rows to a table. The values are separated by commas andthe data types char and date are enclosed in apostrophes. The values must br entered in the same order as they are defined.

**Inserting a single row into a table:**

insert into <table name> values(fieldvalue-1,fieldvalue-2,…,fieldvalue-n);

**Inserting more than one record using a single insert command:**

insert into <table name> values(&fieldname-1,&fieldname-2,…&fieldname-n);

**Skipping the fields while inserting:**

insert into <tablename(coln names to which datas to b inserted)> values (list of values);

Other way is to give null while passing the values.

insert into <table name>(select(att\_list) from <existing table name>);

**UPDATE** - It is used to alter the column values in a table. A single column may be updated or more than one column could be updated.

update <table name> set(fieldname-1 = value, fieldname-2 = value,…,fieldname-n = value)

[WHERE <condition/expression>];

**DELETE** - After inserting row in a table we can also delete them if required. The delete command consists of a from clause followed by an optional where clause.

delete from <table name> [where <condition/expression>];

**Result:**

Thus the query for SQL Data Manipulation Language Commands on sample exercise was executed and verified successfully.

**Ex.No: 3**

**Date:**

**SQL DATA CONTROL LANGUAGE COMMANDS AND TRANSACTION CONTROL COMMANDS**

**AIM**:

To write a program for SQL Data Control Language Commands and Transaction Control Commands on sample exercise.

**Data Control Language [DCL]:**

* DCL is used to retrieve stored or saved data.
* DCL execution is transactional. It also has rollback parameters.

**Some commands are:**

* GRANT
* REVOKE

**Grant:**

It is used to give user access privileges to a database.

**Syntax:**

GRANT SELECT, UPDATE ON TABLE NAME TO SOME\_USER, ANOTHER\_USER;

**Revoke:**

It is used to take back permissions from the user.

**Syntax:**

REVOKE SELECT, UPDATE ON TABLE NAME FROM USER1, USER2;

**Transaction Control Language[TCL]:**

* TCL is used to run changes made by DML statement.
* TCL can be grouped into a logical transaction.

**Some commands are:**

* COMMIT
* ROLLBACK
* SAVEPOINT
* SET TRANSACTION

**Commit:**

Commit command is used to save all the transactions to the database.

**Syntax:**

commit;

**Rollback:**

Rollback command is to rollback a transaction in case of any error occurs.

**Syntax:**

rollback;

**Savepoint:**

Savepoint command is to Set a savepoint within a transaction. If transaction happens in big data, then for checking and rollup can't do it with all the data, to rollback the small part of the data we use savepoint query.

**Syntax:**

SAVEPOINT savepoint\_name

**Set Transaction:**

Set command is to Specify the characteristics of the transaction.

**Syntax:**

SET TRANSACTION Access NAME transaction\_name

**Result:**

Thus the query for SQL Data Control Language Commands and Transaction Control Commands on sample exercise was executed and verified successfully.

**Ex:No:4 INBUILT FUNCTIONS IN SQL**

**Date:**

**AIM:**

To implement query for inbuilt functions using SQL.

**SQL Functions:**

Functions in SQL Server are the database objects that contain a **set of SQL statements to perform a specific task**. The main purpose of functions is to replicate the common task easily. We can build functions one time and can use them in multiple locations based on our needs. SQL Server does not allow to use of the functions for inserting, deleting, or updating records in the database tables.

SQL functions are categorized into the following two categories:

1. Aggregate Functions
2. Scalar Functions

## AGGREGATE SQL FUNCTIONS

The Aggregate Functions in SQL perform calculations on a group of values and then return a single value. Following are a few of the most commonly used Aggregate Functions:

|  |  |
| --- | --- |
| **Function** | **Description** |
| SUM() | Used to return the sum of a group of values. |
| COUNT() | Returns the number of rows either based on a condition, or without a condition. |
| AVG() | Used to calculate the average value of a numeric column. |
| MIN() | This function returns the minimum value of a column. |
| MAX() | Returns a maximum value of a column. |
| FIRST() | Used to return the first value of the column. |
| LAST() | This function returns the last value of the column. |

## SCALAR SQL FUNCTIONS

The Scalar Functions in SQL are used to return a single value from the given input value.  Following are a few of the most commonly used Aggregate Functions:

|  |  |
| --- | --- |
| **Function** | **Description** |
| LCASE() | Used to convert string column values to lowercase |
| UCASE() | This function is used to convert a string column values to Uppercase. |
| LEN() | Returns the length of the text values in the column. |
| MID() | Extracts substrings in SQL from column values having String data type. |
| ROUND() | Rounds off a numeric value to the nearest integer. |
| NOW() | This function is used to return the current system date and time. |
| FORMAT() | Used to format how a field must be displayed. |

**CHARACTER/STRING FUNCTION:**

SQL> select upper('welcome') from dual;

SQL> select upper('hai') from dual;

SQL> select lower('HAI') from dual;

SQL> select initcap(‘hello world') from dual;

SQL> select ltrim('hello world',’hell’) from dual;

SQL> select rtrim(''hello world',’ld’')from dual;

SQL> select concat('SRM',' University')from dual;

SQL> select length('Welcome’) from dual;

SQL> select replace('SRM University', 'University','IST')from dual;

SQL> select lpad('SRM University',20,'\*')from dual;

SQL> select rpad('SRM University',15,'$')from dual;

SQL> select substr('Welcome to SRM University', 4,7)from dual;

SQL> select replace('COMPUTER','O','AB')from dual;

SQL> select replace('University','city’,'Inter')from dual;

SQL> select translate('cold','ld','ol')from dual;

**DATE & TIME FUNCTION**

SQL> select sysdate from dual;

SQL> select round(sysdate)from dual;

SQL> select add\_months(sysdate,3)from dual;

SQL> select last\_day(sysdate)from dual;

SQL> select sysdate+20 from dual;

SQL> select next\_day(sysdate,'tuesday')from dual;

**NUMERIC FUNCTION**

SQL> select round(15.6789)from dual;

SQL> select ceil(23.20)from dual;

SQL> select floor(34.56)from dual;

SQL> select trunc(15.56743)from dual;

SQL> select sign(-345)from dual;

SQL> select abs(-70)from dual;

**MATH FUNCTION:**

SQL> select power(10,12) from dual;

SQL> select power(5,6) from dual;

SQL> select mod(11,5) from dual;

SQL> select exp(10) from dual;

SQL> select sqrt(225) from dual;

**RESULT:**

Thus the implementation of inbuilt functions using SQL was executed and verified successfully.

**Ex.No:5 CONSTRUCTING AN ER MODEL FOR THE APPLICATION**

**Date:**

**AIM:**

To construct an Entity Relation Diagram for (Application\_Name) using RDBMS.

**ENTITY RELATIONSHIP DIAGRAM:**

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes.

In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

**COMPONENTS OF ER DIAGRAM:**

An ER diagram has three main components:-

**1. Entity:**

An entity is an object or component of data. An entity is represented as a rectangle in an ER diagram.

It has 2 types:

- Weak Entity (represented by a double rectangle)

- Strong Entity (represented by a single rectangle).

**2. Attributes:**

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

- Key attribute

- Composite attribute

- Multivalued attribute

- Derived attribute

**3. Relationships:**

A relationship is represented by a diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:

- One to One

- One to Many

- Many to One

- Many to Many

**KEY CONSTRAINTS:**

**PRIMARY KEY:**

Primary key uniquely identifies each record in a table. A Primary Key must contain unique value and it must not contain null value. Usually the primary Key is used to index the data inside the table.

**FOREIGN KEY:**

Foreign Key is used to relate two tables. The relationship between the two tables matches the Primary Key in one of the tables with a Foreign Key in the second table. This is also called a referencing key.

**RESULT:**

Thus the Entity Relationship Diagram for (Application\_Name) has been constructed and verified successfully.

**Ex.No: 6 Nested Queries**

**Date:**

**AIM:**

To implement Nested Queries commands on sample exercise using SQL.

**Nested Query(Sub Query):**

Sub Query can have more than one level of nesting in one single query. A SQL nested query is a SELECT query that is nested inside a SELECT, UPDATE, INSERT, or DELETE SQL query.

1. Select Command Is Used To Select Records From The Table.

2. Where Command Is Used To Identify Particular Elements.

3. Having Command Is Used To Identify Particular Elements.

4. Min (Sal) Command Is Used To Find Minimum Salary.

**SYNTAX FOR CREATING A TABLE:**

SQL> Create <Obj.Type><Obj.Name> (Column Name.1 <Datatype> (Size), Column Name.1 <Datatype> (Size) ............................................);

**SYNTAX FOR INSERT RECORDS IN TO A TABLE:**

SQL>Insert Into <Table Name> Values< Val1, ‘Val2’,... );

**SYNTAX FOR SELECT RECORDS FROM THE TABLE:**

SQL> Select \* From <Table Name>;

**SYNTAX FOR NESTED QUERY:**

SQL>Select "Column\_Name1"From "Table\_Name1"Where "Column\_Name2" [Comparison Operator](Select"Column\_Name3"From "Table\_Name2"Where [Condition])

**NESTED QUERY STATEMENT:**

SQL> Select <Column\_Name> From Frorm<Table \_1> Where<Column\_Name><Relational \_Operation> ‘Value’(Select (Aggrecate Function) From <Table\_1> Where <ColumnName> = ‘Value’(Select<Column\_Name> From <Table\_2> Where <Column\_Name=‘Value’));

**RESULT:**

Thus the Nested Query commands on a sample exercise using SQL was executed and verified successfully.

**EX.NO:7 JOIN QUERIES**

**Date:**

**AIM:**

To implement join query commands using SQL.

**JOIN QUERIES:**

* SQL joins are used to query data from two or more tables, based on a relationship between certain columns in these tables.

**Different Types of SQL JOINs:**

1. Inner Join
2. Self Join
3. Outer Join
4. Left Outer Join
5. Right Outer Join
6. Full Outer Join
7. Cross Join

**INNER JOIN:**

Returns records that have matching values in both tables

**Syntax:**

SELECT column\_name(s)

FROM table1

INNER JOIN table2

ON table1.column\_name = table2.column\_name;

**SELF JOIN:**

A self join is a regular join, but the table is joined with itself.

**Syntax:**

SELECT column\_name(s)

FROM table1 T1, table1 T2

WHERE condition;

**LEFT OUTER JOIN:**

 Returns all records from the left table, and the matched records from the right table.

**Syntax:**

SELECT column\_name(s)

FROM table1

LEFT JOIN table2

ON table1.column\_name = table2.column\_name;

**RIGHT OUTER JOIN:**

Returns all records from the right table, and the matched records from the left table

**Syntax:**

SELECT column\_name(s)

FROM table1

RIGHT JOIN table2

ON table1.column\_name = table2.column\_name;

**FULL OUTER JOIN:**

Returns all records when there is a match in either left or right table

**Syntax:**

SELECT column\_name(s)

FROM table1

FULL OUTER JOIN table2

ON table1.column\_name = table2.column\_name

WHERE condition;

**CROSS JOIN:**

Combines all of the possibilities of two or more tables and returns a result that includes every row from all contributing tables. It's also known as CARTESIAN JOIN.

**Syntax:**

SELECT column\_lists

FROM table1

CROSS JOIN table2;

**RESULT:**

Thus the implementation of join queries using SQL was executed and verified successfully.

**Ex.No:8 SET OPERATORS & VIEWS**

**AIM:**

To implement Set Operators & Views using SQL

**SET Operators:**

SET operators are special type of operators which are used to combine the result of two queries.

Operators covered under SET operators are:

* UNION
* UNION ALL
* INTERSECT
* MINUS(EXCEPT)

**Union:**

UNION will be used to combine the result of two select statements. Duplicate rows will be eliminated from the results obtained after performing the UNION operation.

**Syntax**:

SELECT \*FROM (table\_name1) UNION SELECT \*FROM (table\_name2);

**Union All:**

This operator combines all the records from both the queries. Duplicate rows will be not be eliminated from the results obtained after performing the UNION ALL operation.

**Syntax:**

SELECT \*FROM (table\_name1) UNION ALL SELECT \*FROM (table\_name2);

**Intersect:**

It is used to combine two SELECT statements, but it only returns the records which are common from both SELECT statements.

**Synatx:**

SELECT \*FROM (table\_name1) INTERSECT SELECT \*FROM (table\_name2);

**Minus(Except):**

It displays the rows which are present in the first query but absent in the second query with no

duplicates.

**Synatx:**

SELECT \*FROM(table\_name1) MINUS SELECT \*FROM (table\_name2);

**VIEWS:**

1. CREATE VIEW command is used to define a view.

**Syntax:**

CREATE VIEW (view\_name) AS SELECT (column1, column2.....) FROM (table\_name) WHERE (condition);

2. INSERT command is used to insert a new row into the view.

**Syntax:**

INSERT INTO (Table\_Name) VALUES (value1, value2, value 3, .... Value N);

3. DELETE command is used to delete a row from the view.

**Syntax:**

DELETE FROM (table\_name) WHERE (some\_condition);

4. UPDATE command is used to change a value in a tuple without changing all values in the tuple.

**Syntax:**

UPDATE (table\_name)  SET (column1 = value1, column2 = value2, ...)  WHERE (condition);

5. DROP command is used to drop the viewtable.

**Syntax:**

DROP VIEW (view\_name);

**RESULT:**

Thus the implementation of set operators and view commands was executed and verified successfully.

**Ex.No: 9 PL/SQL CONDITIONAL AND ITERATIVE STATEMENTS**

**Date:**

**AIM:**

To write a program on PL/SQL Conditional and Iterative Statements.

**Conditional and Iterative Statements:**

PL/SQL supports the programming language features like conditional statements and iterative statements.

**PL/SQL If Statement:**

**Syntax:**

(IF-THEN statement):

IF condition

THEN

Statement: {It is executed when condition is true}

END IF;

**PL/SQL If-then-Else Statement:**

IF condition

THEN

{...statements to execute when condition is TRUE...}

ELSE

{...statements to execute when condition is FALSE...}

END IF;

**PL/SQL while loop:**

It is used when a set of statements has to be executed as long as a condition is true, the While loop is used. The condition is decided at the beginning of each iteration and continues until the condition becomes false.

**Syntax:**

WHILE <condition>

LOOP statements;

END LOOP;

**PL/SQL for loop:**

It is used when we want to execute a set of statements for a predetermined number of times. The loop is iterated between the start and end integer values. The counter is always incremented by 1 and once the counter reaches the value of end integer, the loop ends.

**Syntax of for loop:**

FOR counter IN initial\_value .. final\_value LOOP

LOOP statements;

END LOOP;

initial\_value : Start integer value

final\_value : End integer value

**Result:**

Thus a program on PL/SQL Conditional and Iterative Statements was executed and verified successfully.

**Ex.No: 10 PL/SQL PROCEDURES**

**Date:**

**AIM:**

To write a program on PL/SQL Procedures on sample exercises.

**PL/SQL PROCEDURES**

The PL/SQL stored procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

The procedure contains a header and a body.

**Header:** The header contains the name of the procedure and the parameters or variables passed to the procedure.

**Body:** The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

**Syntax for creating procedure:**

CREATE [OR REPLACE] PROCEDURE procedure\_name [ (parameter [,parameter]) ]

IS

[declaration\_section]

BEGIN

executable\_section

[EXCEPTION

exception\_section]

END [procedure\_name];

**Result:**

Thus a program on PL/SQL Procedures was verified and executed successfully.

**Ex.No: 11 PL/SQL FUNCTIONS**

**Date:**

**AIM:**

To write a program on PL/SQL Functions

**PL/SQL FUNCTIONS:**

* The PL/SQL Function is very similar to PL/SQL Procedure.
* The main difference between procedure and a function is, a function must always return a value, and on the other hand a procedure may or may not return a value.

**Syntax to create a function:**

CREATE [OR REPLACE] FUNCTION function\_name [parameters]

[(parameter\_name [IN | OUT | IN OUT] type [, ...])]

RETURN return\_datatype

{IS | AS}

BEGIN

< function\_body >

END [function\_name];

**Result:**

Thus a program on PL/SQL Functions was verified and executed successfully.

**Ex.No: 12 PL/SQL CURSORS**

**Date:**

**AIM:**

To write program on PL/SQL Cursors.

**PL/SQL Cursors:**

A cursor is used to refer to a program to fetch and process the rows returned by the SQL statement, one at a time.

**PL/SQL Implicit Cursors:**

* The implicit cursors are automatically generated by Oracle while an SQL statement is executed, if we don't use an explicit cursor for the statement.
* These are created by default to process the statements when DML statements like INSERT, UPDATE, DELETE etc. are executed.
* Some of them are: %FOUND, %NOTFOUND, %ROWCOUNT and %ISOPEN.

**PL/SQL Explicit Cursors:**

• The Explicit cursors are defined by the programmers to gain more control over the context area.

• These cursors should be defined in the declaration section of the PL/SQL block. It is created on a SELECT statement which returns more than one row.

**Syntax to create an explicit cursor:**

CURSOR cursor\_name IS select\_statement;;

**Result:**

Thus a program on PL/SQL Cursorswas verified and executed successfully.

**Ex.No: 13 PL/SQL EXCEPTION HANDLING**

**Date:**

**AIM:**

To write a program on PL/SQL Exception Handling.

**PL/SQL EXCEPTION HANDLING:**

* An error occurs during the program execution is called Exception in PL/SQL.
* PL/SQL facilitates programmers to catch such conditions using exception block in the program and an appropriate action is taken against the error condition.

There are two types of exceptions:

• System-defined Exceptions.

• User-defined Exceptions.

**Syntax for exception handling:**

DECLARE

<declarations section>

BEGIN

<executable command(s)>

EXCEPTION

<exception handling goes here >

WHEN exception1 THEN

exception1-handling-statements

WHEN exception2 THEN

exception2-handling-statements

WHEN exception3 THEN

exception3-handling-statements

........

WHEN others THEN

exception3-handling-statements

END;

**Result:**

Thus a program on PL/SQL Exception Handling was verified and executed successfully.

**Ex.No: 14 PL/SQL TRIGGER**

**Date:**

**AIM:**

To write a program on PL/SQL Trigger.

**PL/SQL TRIGGER:**

* Trigger is invoked by Oracle engine automatically whenever a specified event occurs.
* Trigger is stored into database and invoked repeatedly, when specific condition match.
* Triggers are stored programs, which are automatically executed or fired when some event occurs.

**Syntax for creating trigger:**

CREATE [OR REPLACE ] TRIGGER trigger\_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col\_name]

ON table\_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

DECLARE

Declaration-statements

BEGIN

Executable-statements

EXCEPTION

Exception-handling-statements

END;

**Result:**

Thus a program on PL/SQL Trigger was verified and executed successfully.