**Practical No: 1**

**SQL Statements- 1**

**1) Writing basic SQL statements**

**a**) **SELECT Query**

The SELECT statement is used to select data from a database.

**SYNTAX:**

**To view any particular column from table -**

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

**To view the entire table –**

SELECT \* FROM *table\_name*;

**b) WHERE Clause**

The WHERE clause is used to filter records.

It is used to extract only those records that fulfil a specified condition.

**SYNTAX:**

SELECT *column1*,*column2, ...*  
FROM *table\_name*  
WHERE *condition*;

**2) Restricting and sorting data**

**a) DISTINCT Clause -**

Used to remove duplicate records from the table and fetch only the unique records. The DISTINCT clause is only used with the SELECT statement.

**SYNTAX:**

**SELECT** **DISTINCT** expressions

**FROM** tables

[**WHERE** conditions];

**b)** **ORDER BY Clause**

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

**SYNTAX:**

SELECT *column1*,*column2, ...*  
FROM *table\_name*  
ORDER BY *column1, column2, ...*ASC|DESC;

**3) Single row functions**

**a) Case – conversion function**

The case-conversion functions perform lettercase conversion on alphabetic characters

The case-conversion functions are **UPPER**, **LOWER**, and **INITCAP**.

**SYNTAX:**

Select UPPER(columnname1), LOWER(columnname2), INITCAP(columnname3) from tablename;

**b) CONCAT -** The CONCAT() function adds two or more strings together.

**SYNTAX:** SELECT CONCAT(*string1*, *string2*, *....*, *string\_n*) FROM TABLENAME;

**c) LENGTH -** The LEN() function returns the length of a string.

**SYNTAX:** SELECT LENGTH(*string*) FROM TABLENAME;

**d) TRIM -** The TRIM() function removes the space character OR other specified characters from the start or end of a string.

**SYNTAX:** Select columnname, TRIM(LEADING ‘X’ FROM COLUMNAME) FROM TABLENAME;

**Practical No: 2**

**SQL Statements- 2**

**1) Displaying data from multiple tables**

**Using JOINS**

A) Inner joins:

The INNER JOIN keyword selects 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*;

B) Left outer join:

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

**SYNTAX:**

SELECT *column\_name(s)*  
FROM *table1*  
LEFT JOIN *table2*ON *table1.column\_name*=*table2.column\_name*;

C) Right outer join:

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

**SYNTAX:**

SELECT *column\_name(s)*  
FROM *table1*  
RIGHT JOIN *table2*ON *table1.column\_name*=*table2.column\_name*;

**2) Aggregating data using group functions**

A) Count ():

The COUNT() function returns the number of records returned by a select query.

**SYNTAX:**

SELECT *count(column\_name) from table\_name;*

B) Sum ():

The SUM() function calculates the sum of a set of values.

**SYNTAX:**

SELECT *sum(column\_name) from table\_name;*

C) Avg():

The AVG() function calculates the average of a set of values.

**SYNTAX:**

SELECT *avg(column\_name) from table\_name;*

**3) Sub-Queries**

A sub query is a query nested within another query such as SELECT, INSERT, UPDATE or DELETE.

**A) Min():**

The MIN() function in MySQL is used to return the minimum value in a set of values from the table.

**B Max():**

The MAX() function in MySQL is used to return the maximum value in a set of values from the table.

**Practical No: 3**

**Data manipulation Language - DML**

The Data Manipulation Language (DML) is used for accessing and manipulating data in a database.

It allows users to access, insert, update, and delete data from the database.

**(a) Using INSERT STATEMENT**

1) Inserting single row of data

**Syntax:**

insert into tablename (column\_name1, column\_name2)

values (value1,value2) ;

2) Inserting data into table from another exisiting table

**Syntax:**

Insert into new\_table select \* from old\_table**;**

**(b) Delete statement**

The DELETE statement is used to delete existing records in a table.

**Syntax:**

DELETE FROM *table\_name*WHERE *condition*;

**(c)Update statement**

The UPDATE statement is used to modify the existing records in a table.

**Syntax:**

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

**Practical No: 4**

**Creating and managing tables**

1. **Creating new table from existing table**

**Syntax:**

*create table table\_name*

*as*

*select \* from exisiting\_table\_name;*

1. **Creating new table with specific fields i.e. columns from existing table**

**Syntax:**

*Create table table\_name*

*As*

*Select column\_name1, column\_name2 from existing table-name;*

1. **Creating new table with specific records from existing table**

**Syntax:**

*create table table\_name*

*as*

*select \* from exisiting\_table\_name;*

*where condition;*

1. **Modifying table**

ALTER TABLE query is used to modify structure of a table which already exist in database

1. **Adding extra column in existing table**

**Syntax:** *ALTER TABLE table\_name ADD column\_name datatype;*

1. **Deleting column in existing table**

**Syntax:** *ALTER TABLE table\_name drop column\_name;*

1. **Modifying data-type of column in existing table**

**Syntax***:*

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

1. **Renaming table**

**Syntax***:*

*Rename table current\_table\_name to new\_table\_name;*

1. **Delete table**

**Syntax***: Drop table table\_name;*

**Practical No : 5**

**Aim : Creating and managing other database objects**

**1)** **Creating Views**

**a) Creating view having all records and field from existing table**

**Syntax :**

***Create or replace view view\_name1***

***As***

***Select \* from tablename;***

**b) Creating new view with specific fields i.e. columns from existing table**

**Syntax :**

***Create or replace view view\_name1***

***As***

***Select column\_name1,column\_name2 from tablename;***

**c) Creating new view with specific records from existing table**

**Syntax:**

***Create or replace view view\_name1***

***As***

***Select \* from tablename***

***Where condition;***

**2)** **Creating Index**

**a)** **Create index with specific records**

**Syntax:**

**Create index index\_name1 on tablename (column\_name);**

**Show index:**

**Syntax-**

***show index from table\_name;***

**b)** **Drop index**

**Syntax:**

***Drop index index\_name1;***

**Practical No: 6**

**Aim:** Using SET operators, DATE/TIME Functions

1. SET operators

Different SET operators are as follows

1. Union

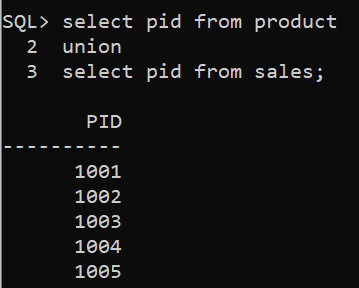
The union operator returns all distinct rows selected by query

Syntax:

*Select column\_name from table\_name1*

*Union*

*Select column\_name from table\_name2;*



1. Union All

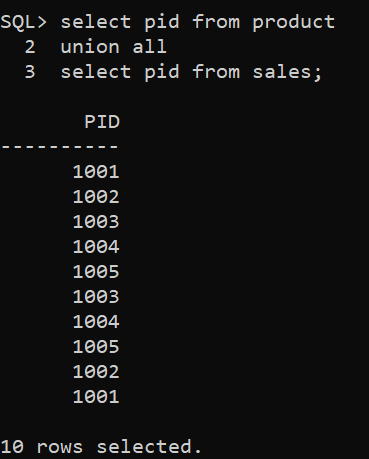
The union all operator returns all rows selected by either query including duplicates

Syntax:

*Select column\_name from table\_name1*

*Union all*

*Select column\_name from table\_name2;*



1. Intersect

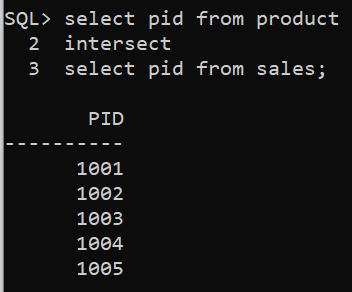
The intersect operator returns only these rows which are common to both the queries

Syntax:

*Select column\_name from table\_name1*

*Intersect*

*Select column\_name from table\_name2;*



1. Minus

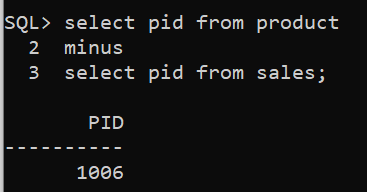
Minus operator displays the rows which are present in the first query but absent in the second query, with no duplicates and data is arranged in ascending order by default

Syntax:

*Select column\_name from table\_name1*

*minus*

*Select column\_name from table\_name2;*

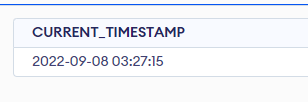


1. Date-Time Functions

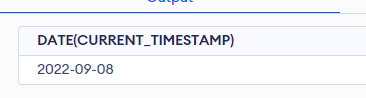
Syntax

a)

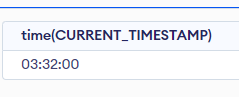
SELECT CURRENT\_TIMESTAMP ;



b) SELECT DATE(CURRENT\_TIMESTAMP);



c) SELECT TIME(CURRENT\_TIMESTAMP);



**PRACTICAL NO - 7**

**AIM: Basics of PL/SQL**

**The following illustrates the basic syntax of creating PL/SQL:**

**Syntax:**

[declaration statements]

BEGIN

[execution statements]

EXCEPTION

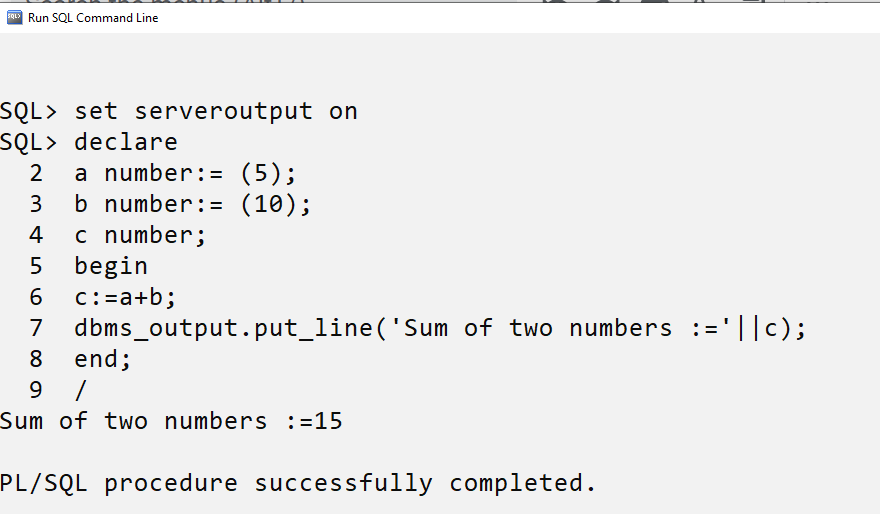
[exception handler]

END;

**(A) Declaring Variables Question**

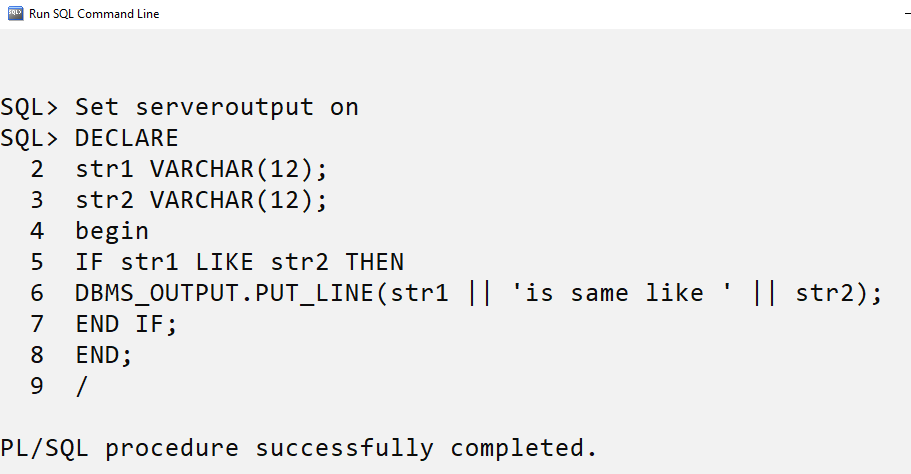
Displaying sum of two numbers

CODE:

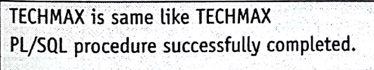


**(B) Writing Control Structures**

Using if Statement

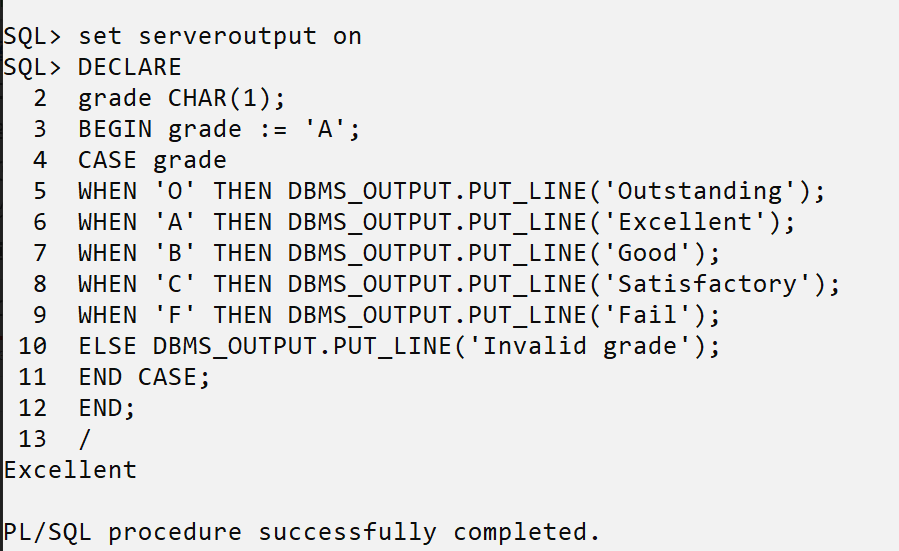
****

**Output:**



**(C) Using CASE WHEN statement**

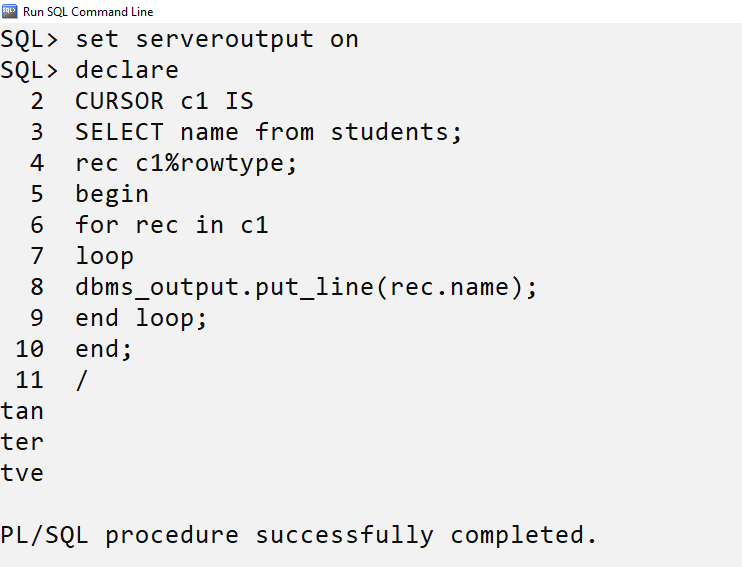
**Code:**

****

**PRACTICAL NO: 8**

AIM: Cursors in PL/SQL

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.



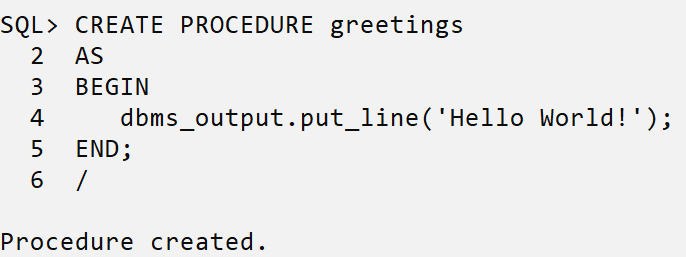
**PRACTICAL NO : 9**

AIM: Procedures in PL/SQL

**Procedure**

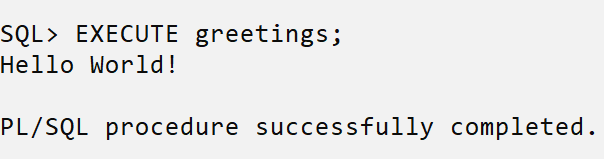
1. Creating procedure

Use the CREATE PROCEDURE statement to create a standalone stored procedure or a call specification.



1. **Executing procedure**

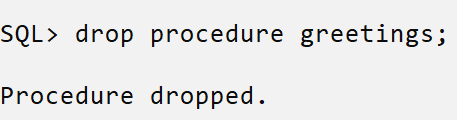
A **procedure** is a group of PL/SQL statements that you can call by name.



**c)Drop Procedure**

A standalone procedure is deleted with the DROP PROCEDURE statement.

Syntax: DROP PROCEDURE procedure-name;



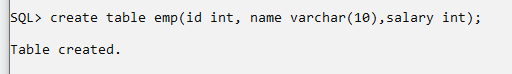
**PRACTICAL NO: 10**

AIM: Triggers in PL/SQL

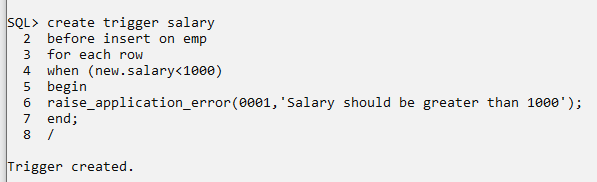
A PL/SQL trigger is a named database object that encapsulates and defines a set of actions that are to be performed in response to an insert, update, or delete operation against a table. Triggers are created using the PL/SQL CREATE TRIGGER statement.

Create trigger to insert salary above 1000

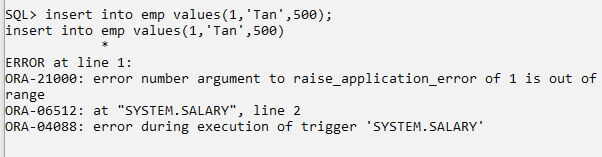
Step 1: Create table



Step 2: Create trigger



Step 3: Execute trigger



Step 4: Drop trigger

