Expt. No. 1

Date :

Page No. :

Create tables and specify the Queries in SQL :-

* Creating a table :-

```
CREATE TABLE Table1 (
        Name  varchar(30),
        Age   int(30),
        Branch  varchar(30)
    );
```

* Insertion of Data :-

```
INSERT INTO Table1 (Name, Age, Branch)
        VALUES ('Peter', '21', 'CSE');
```

* Retrieving of Data from the table :-

```
SELECT * FROM Table1
```

Retrieving of Selected data :-

```
SELECT Name FROM Table1
```

Create Table :-

Table created.

Insertion and Retrieving :-

Name	Age	Branch
Peter	21	CSE

HAWA MAHAL

Expt. No. 1

Date :

Page No. :

Create tables and specify the Queries in SQL :-

* Creating a table :-

```
CREATE TABLE Table1 (
        Name varchar(30),
        Age int(30),
        Branch varchar(30)
    );
```

* Insertion of Data :-

```
INSERT INTO Table1 (Name, Age, Branch)
        VALUES ('Peter', '21', 'CSE');
```

* Retrieving of Data from the table :-

```
SELECT * FROM Table1
```

Retrieving of Selected data :-

```
SELECT Name FROM Table1
```

Create Table :-

Table created.

Insertion and Retrieving :-

Name	Age	Branch
Peter	21	CSE

Expt. No. 2

Date :

Page No. :

Manipulate the operations on the table :-

Update Operation :- To change a value in the table without changing all values.

For the table ; (Persons)

Name	City	Age
Tony	Berlin	21
Steve	Rio	21
Harry	London	24

* Using Update Operation :-

```
UPDATE Persons
  SET Name = 'Tony',
      City = 'Paris'
```

Deletion Operation :- To delete a row or all of them.

In the same table shown above,

* DELETE FROM Persons WHERE
Name = 'Harry'

Update :-

Name	City	Age
Tony	Paris	21
Steve	Rio	21
Harry	London	24

Delete :-

Name	City	Age
Tony	Berlin	21
Steve	Rio	21

* Deleting all rows :-

~~$t~~ DELETE FROM Persons ;

Logical Operators :-

i) AND - all must be included
ii) OR - any of may be included
iii) NOT - none could be included

Sorting of data :-

* SELECT * FROM Persons
ORDER BY Name ASC

Expt. No. 3

Date :

Page No. :

Implement the restrictions on the table :-

* **NOT NULL** :- Constraint to enforce a column to NOT accept NULL values.

```
CREATE TABLE Persons2(
    ID int NOT NULL,
    LastName varchar(50) NOT NULL,
    FirstName varchar(50) NOT NULL
);
```

* **Primary Key**:- It uniquely indentifies each record in a table.

```
CREATE TABLE Persons2(
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(30) NOT NULL,
    FirstName varchar(30)
);
```

* **Unique**:- It ensures that all values in a column are different.

```
CREATE TABLE Persons2( ID int NOT NULL UNIQUE,
    LastName varchar(30) NOT NULL, FirstName varchar(30));
```

Teacher's Signature : .....

DEFAULT :- Used to set a default value for a column.

* CREATE TABLE Persons2(
    ID int NOT NULL,
    LastName varchar(30) NOT NULL,
    City varchar(20) DEFAULT 'Sandness');

FOREIGN KEY :- Used to prevent actions that would destroy links between tables.

* CREATE TABLE Orders (
    OrderID int NOT NULL,
    Order Number int NOT NULL,
    Person ID int,
    PRIMARY KEY (Order ID),
    FOREIGN KEY (Person ID)
    REFERENCES Persons (Person ID)
);

HAWA MAHAL

Expt. No. 4

Date :

Page No. :

Implement the Structure of the table :-

Add Column :- [taking the reference table mentioned in experiment 2]

* ALTER TABLE Persons
  ADD Branch varchar(30);

Drop Column :-

* ALTER TABLE Persons
  DROP COLUMN Age;

* ### Modify Column :-

ALTER TABLE Persons
ALTER COLUMN Name(100);

* ### Add and Drop Primary Key :-

ALTER TABLE Persons
ADD PRIMARY KEY(Name)

ALTER TABLE Persons
DROP PRIMARY KEY

Expt. No. 5

Date :

Page No. :

Implement Concept of Joins :-

JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Types of Joins :-

i) INNER JOIN :- Returns record that have matching values in both tables.

ii) LEFT (OUTER) JOIN :- Returns all records from the left table, and the matched records from the right table.

iii) RIGHT (OUTER) JOIN :- Returns all records from the right table, and the matched records from the left table.

iv) FULL (OUTER) JOIN :- Returns all records when there is a match in either left or right table.

For the table, 'Orders'

OrderID	Customer ID	Order Date
242	1	2021-12-15
323	37	2021-12-16
741	43	2021-12-17

& the table, 'Customers'

Customer ID	Customer Name	Contact Name	Country
1	Ana	Ana Trcuf	~~Bost~~ USA
2	Bill	Bill Stark	UK
3	Larry	Larry Clinton	Germany

* Using JOIN :-

```
SELECT Orders.OrderID,
Customers.CustomerName,
Orders.OrderDate
FROM Orders
INNER JOIN Customers ON
Orders.CustomerID = Customers.CustomerID;
```

JOIN :-

OrderID	CustomerName	OrderDate
242	Ana Truf	2021-12-15
573	Antonio Smith	2021-12-16
242	Peter Parker	2021-12-14
149	Will Parker	2021-12-12
563	Kate Thompson	2021-12-17

Expt. No. 6

Date :

Page No. :

Implement the concept of grouping of data :-

GROUP BY statement groups rows that have the same values into summary rows.

For the table,

Customer ID	Customer Name	Country
1	Tony	USA
2	Peter	UK
3	Steve	USA
4	Chris	USA
5	Bruce	USA

* Grouping :-

```
SELECT COUNT(CustomerID), Country
  FROM Customers
  GROUP BY Country;
```

Grouping:

Count (Customer ID)	Country
4	USA
1	UK

HAWA MAHAL

Expt. No. 7

Date :

Page No. :

Implement the concept of SubQuestionaries :-

Union Clause :- It merges the output of two or more Questionaries into a single set of rows and columns.

For the table 'Customers'

Customer ID	CName	City
1	Peter	New York
2	Tony	London
3	Bruce	Paris

& table 'Suppliers'

Supplier ID	SName	City
1	Harry	Berlin
2	Ron	Moscow
3	Tom	Rio

* UNION :-

```
SELECT City FROM Customers
UNION
 SELECT City FROM Suppliers
ORDER BY City;
```

Teacher's Signature : .....

Union:

City
Berlin
London
Moscow
New York
Paris
Rio

Intersect Clause:- It returns only those rows which are common to both SELECT statements.

For the table 'Customers'

ID	Name	Address	Age	Salary
1	Harsh	Delhi	20	30000
2	Pratik	Mumbai	21	40000
3	Akash	Madras	35	50000

& table 'Orders'

OID	Date	Customer.ID	Amount
102	2021-10-08	3	3000
100	2021-10-08	3	1500
101	2021-11-20	2	1560

* **INTERSECT:-**

```
SELECT ID, NAME, Amount, Date
    FROM Customers
    LEFT JOIN Orders
    ON Customers.ID = Orders.Customer.ID
INTERSECT
    SELECT ID, NAME, Amount, Date
    FROM Customers
    RIGHT JOIN Orders
    ON Customers.ID = Orders.Customer.ID;
```

Teacher's Signature : .....

Intersect :

ID	Name	Amount	Date
3	Akash	3000	2021-10-08
3	Akash	1500	2021-10-08
2	Pratik	1560	2021-11-20

MINUS Clause:- It returns only those rows which are unique in only first SELECT query and not those rows which are common to both first & second SELECT queries.

Table 1 ;

Name	Address	Age	Grade
Harsh	Delhi	20	A
Partik	Mumbai	21	A
Gaurav	Jaipur	21	B

Table 2 ;

Name	Age	Grade
Akash	20	A
Vaibhav	21	B
Gaurav	21	B

* **MINUS :-**

```
SELECT Name, Age, Grade
 FROM Table 1
 MINUS
SELECT Name, Age, Grade
 FROM Table2
```

Minus 1-

Name	Age	Grade
Harsh	20	A
Pratik	21	A

Expt. No. 8

Date :

Page No. :

To implement the concept of Indexes and Views:-

Index :- An index is an ordered list of content of a column or group of columns in a table.

* **Creating an Index :-**

```
CREATE INDEX index_name
ON tablename (column name);
```

* **Composite Index :-**

```
CREATE INDEX inden_name
ON tablename(columnname);
```

* **UniQuestion Inden :-**

```
CREATE UNIQUESTION INDEX indenfilename
ON tablename (columnname);
```

* **Dropping Inden :-**

```
DROP INDEX indenfilename
```

Teacher's Signature : .....

Views :- A view is a virtual table based on the result-set of an SQL statement.

* Creating a View :-

```
CREATE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = 'Brazil';
```

* Updating a View :-

```
CREATE OR REPLACE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName, City
FROM Customers
WHERE Country = 'Brazil';
```

* Dropping a View :-

```
DROP VIEW view_name; [Brazil Customers];
```

Expt. No. 9

Date :

Page No. :

Implement the basics of PL/SQL :-

PL/SQL bridges the gap between database technology and procedural programming languages. Via this, we can insert, delete, update and retrieve table data as well as used use procedural techniques.

Block Structure :-

* DECLARE
  Declarations of memory variables used later
  BEGIN
  SQL executable statements for manipulating table data
  EXCEPTIONS
  SQL and/or PL.SQL code to handle errors.
  END;

Any programing tool requires a method through which messages can be displayed to the user.

Setting the server output on :-

* ## Conditional Control :-

  IF <Condition> THEN
  <Action>

ELSEIF <condition>
<Action>
ELSE
<Action>
END IF;

* **WHILE LOOP:-**

WHILE <condition>
LOOP
<Action>
END LOOP;

* **FOR LOOP:-**

FOR variable IN [REVERSE] start-end
LOOP
<Action>
END LOOP;

* **GOTO:-** Allows to change the flow of control within a PL/SQL Block.

Expt. No. 10

Date :

Page No. :

Implement the concept of Cursor & Trigger :-

Cursor :- Oracle DBA uses a work area for its internal processing called as a cursor.

* **Declaring a Cursor :-**

Cusor < Customername > in SQL Statement

* **Opening a Cursor :-**

Open Cursorname ;

* **Closing a Cursor :-**

Close < cursorname >

* **Fetching a record from the cursor :-**

Fetch cursorname into variable 1, variable 2 —

Trigger :- Triggers are procedures that are stored in the database and are implicitly executed when the contents of a table are changed.

* **Creating a Trigger :-**

Create or replace Trigger <Triggername> {Before, After} {Delete ~~, Insert~~, Insert, Update}
On <Tablename> For Each row when Condition
Declare
<Variable declarations>;
<Constant Declarations>;
Begin
<PL/SQL> Subprogram Body;
Exception
Exception PL/SQL block;
END;

* **Deleting a Trigger :-**

DROP Trigger <Triggername>;