# INDEX

## 1.1 DDL Commands with Constraints

## NOT NULL Constraint

**Method 1: Not Null Constraint in-line without name**

```
CREATE TABLE Student
(
ID int NOT NULL,
NAME varchar(30) NOT NULL,
ADDRESS varchar(50)
);
```

---

**Method 2: Not Null Constraint in-line with name**

**Constraint name: Stu_NotNull**

```
CREATE TABLE Student
(
ID int constraint Stu_NotNull NOT NULL,
NAME varchar(30) NOT NULL,
ADDRESS varchar(50)
);
```

---

**Method 3: Adding a NOT NULL Constraint after the Creation of a Student Table**

```
ALTER TABLE Student MODIFY ID int NOT NULL
```

**OUTPUT:**

Run SQL Command Line

```
SQL> CREATE TABLE Student
  2  (
  3  ID int constraint Stu_NotNull NOT NULL,
  4  NAME varchar(30) NOT NULL,
  5  ADDRESS varchar(50)
  6  );

Table created.

SQL>
```

## 1.2 DDL Commands with Constraints

**Unique Constraint:**

**Method 1: Unique Constraint in-line without name**

**Unique Constraint Name :uq_pname**

```
Create table person
(
pno   int,
pname varchar(20),
mobile number(12) unique
);
```

---

**Method 2: Unique Constraint in-line with name**

**Unique Constraint Name: uq_mobile**

```
Create table person1
(
pno   int,
pname varchar(20),
mobile number(12) constraint uq_mobile  unique
);
```

---

**Method 3: Unique Constraint out-of-line with name**

```
Create table person
(
pno   int,
pname varchar(20),
mobile number(12),
constraint uq_mobile unique (mobile)
);
```

---

**Method 4: Adding a UNIQUE Constraint after the Creation of a person Table**

**ALTER TABLE** person **ADD CONSTRAINT** uq_mobile **UNIQUE(**mobile**)**

**OUTPUT:**

Run SQL Command Line

```
SQL> Create table person1
  2  (
  3  pno   int,
  4  pname varchar(20),
  5  mobile number(12) constraint uq_mobile  unique
  6  );

Table created.

SQL> _
```

## 1.3 DDL Commands with Constraints

**Primary Key Constraint:**

**Method 1: Primary Key Constraint in-line without name**

```
CREATE TABLE Student
(
ID int primary key
NAME varchar(30),
ADDRESS varchar(50),
);
```

---

**Method 2: Primary Key Constraint in-line with name**

**Primary Key Constraint Name: pk_ID**

```
CREATE TABLE Student
(
ID int constraint pk_ID primary key,
NAME varchar(30),
ADDRESS varchar(50),
);
```

---

**Method 3: Primary Key Constraint out-of-line with name**

**Primary Key Constraint Name: pk_ID**

```
CREATE TABLE Student
(
ID int,
NAME varchar(30),
ADDRESS varchar(50),
constraint pk_ID primary key(ID)
);
```

---

**Method 4: Adding a PRIMARY KEY Constraint after the Creation of a Student Table**

```
ALTER TABLE Student ADD CONSTRAINT pk_ID PRIMARY KEY(ID)
```

**OUTPUT:**

Run SQL Command Line

```
SQL> CREATE TABLE Student
  2  (
  3  ID int,
  4  NAME varchar(30),
  5  ADDRESS varchar(50),
  6  constraint pk_ID primary key(ID)
  7  );

Table created.

SQL> _
```

## 1.4 DDL Commands with Constraints

**Composite Primary Key Constraint:**

**Method 1: Composite Primary Key Using CREATE TABLE**

```
CREATE TABLE customer1
  (
    cust_id  NUMBER(3),
    cust_name VARCHAR(30),
    phone_no NUMBER(10),
    CONSTRAINT cust_pk PRIMARY KEY ( cust_id, phone_no)
  );
```

---

**Method 2: Composite Primary Key Using ALTER TABLE**

```
ALTER TABLE customer
ADD CONSTRAINT cust_cid_pk PRIMARY KEY (cust_id,phone_no);
```

**OUTPUT:**

Run SQL Command Line

```
SQL> CREATE TABLE customer1
  2   (
  3    cust_id  NUMBER(3),
  4    cust_name VARCHAR(30),
  5    phone_no NUMBER(10),
  6    CONSTRAINT cust_pk PRIMARY KEY ( cust_id, phone_no)
  7   );

Table created.

SQL>
```

## 1.5 DDL Commands with Constraints

### Foreign Key Constraint:

### Parent Table

```
CREATE TABLE dept1
    (
    deptno  NUMBER(2) CONSTRAINT pkdept PRIMARY KEY,
    dname   VARCHAR2(9),
    loc     VARCHAR2(10) );
```

---

### Method 1: Foreign Key Constraint in-line without name

```
CREATE TABLE emp1
   (
    empno      NUMBER(4),
    ename      VARCHAR2(10),
    job        VARCHAR2(9),
    mgr        NUMBER(4),
    hiredate   DATE,
    sal        NUMBER(7,2),
    comm       NUMBER(7,2),
    deptno     Number(2) REFERENCES dept(deptno) );
```

---

### Method 2: Foreign Key Constraint Out-of-line with name

```
CREATE TABLE emp
  (
   empno     NUMBER(4),
   ename     VARCHAR2(10),
   job       VARCHAR2(9),
   mgr       NUMBER(4),
   hiredate  DATE,
   sal       NUMBER(7,2),
   comm      NUMBER(7,2),
   deptno    NUMBER(2),
   CONSTRAINT fk_deptno FOREIGN KEY(deptno)
                    REFERENCES dept(deptno));
```

**Method 3: Foreign Key Constraint in-line with name & ON DELETE CASCADE**

```
CREATE TABLE emp
  (
   empno    NUMBER(4),
   ename    VARCHAR2(10),
   job      VARCHAR2(9),
   mgr      NUMBER(4),
   hiredate DATE,
   sal      NUMBER(7,2),
   comm     NUMBER(7,2),
   deptno   NUMBER(2)   CONSTRAINT fk_deptno
                  REFERENCES dept(deptno) ON DELETE CASCADE
);
```

---

**Method 4: Foreign Key Constraint with ALTER TABLE**

**ALTER TABLE** emp ADD **CONSTRAINT** fk_deptno **FOREIGN KEY**(deptno) **REFERENCES** dept(deptno) **ON DELETE CASCADE**;

**OUTPUT**

Run SQL Command Line

```
SQL> CREATE TABLE dept1
  2       (
  3       deptno  NUMBER(2) CONSTRAINT pkdept PRIMARY KEY,
  4       dname   VARCHAR2(9),
  5       loc     VARCHAR2(10) );

Table created.

SQL> CREATE TABLE emp1
  2      (
  3       empno      NUMBER(4),
  4       ename      VARCHAR2(10),
  5       job        VARCHAR2(9),
  6       mgr        NUMBER(4),
  7       hiredate   DATE,
  8       sal        NUMBER(7,2),
  9       comm       NUMBER(7,2),
 10       deptno     Number(2) REFERENCES dept(deptno) );

Table created.

SQL>
```

## 1.6 DDL Commands with Constraints

**Check Constraint:**

**Method 1: Check Constraint in-line without name**

```
CREATE TABLE dept
(
deptno  NUMBER   CHECK (deptno BETWEEN 10 AND 99),
dname VARCHAR2(9)  CHECK (dname = UPPER(dname)),
loc VARCHAR2(10)  CHECK (loc IN ('DALLAS','BOSTON',
                                 'NEW YORK','CHICAGO'))
);
```

---

**Method 2: Check Constraint in-line with name**

```
CREATE TABLE dept01
(
deptno  NUMBER  CONSTRAINT checkdeptno
CHECK (deptno BETWEEN 10 AND 99),
dname VARCHAR2(9)  CONSTRAINT checkdname
CHECK (dname = UPPER(dname)),
loc VARCHAR2(10)  CONSTRAINT checkloc
CHECK (loc IN ('DALLAS','BOSTON',
'NEW YORK','CHICAGO'))
);
```

**OUTPUT**

Run SQL Command Line

```
SQL> CREATE TABLE dept01
  2  (
  3  deptno  NUMBER  CONSTRAINT checkdeptno
  4  CHECK (deptno BETWEEN 10 AND 99),
  5  dname VARCHAR2(9)  CONSTRAINT checkdname
  6  CHECK (dname = UPPER(dname)),
  7  loc VARCHAR2(10)  CONSTRAINT checkloc
  8  CHECK (loc IN ('DALLAS','BOSTON',
  9  'NEW YORK','CHICAGO'))
 10  );

Table created.

SQL>
```

## 1.7 DDL Commands with Constraints

**Default Constraint:**

**Method 1: Default Constraint in CREATE TABLE**

```
CREATE TABLE Sales1 (
    Sale_Id int NOT NULL UNIQUE,
    Sale_Amount int NOT NULL,
    Vendor_Name varchar(255) DEFAULT 'Unknown Vendor',
    Sale_Date date,
    Profit int
);
```

**Method 2: Default Constraint in ALTER TABLE**

```
ALTER TABLE Sales
MODIFY Vendor_Name DEFAULT 'Unknown Vendor';
```

**OUTPUT:**

Run SQL Command Line

```
SQL> CREATE TABLE Sales1 (
  2       Sale_Id int NOT NULL UNIQUE,
  3       Sale_Amount int NOT NULL,
  4       Vendor_Name varchar(255) DEFAULT 'Unknown Vendor',
  5       Sale_Date date,
  6       Profit int
  7  );

Table created.

SQL> _
```

## 2.1 DML COMMANDS

**INSERT COMMAND**

The Employee table is created to execute the DML statements in Oracle.

```
CREATE  TABLE  Employee1
(
Id INT,
 Name  CHAR(20),
Salary NUMBER(8, 2)
);
```

**Case 1:**

**INSERT INTO EMPLOYEE1 VALUES (1, ‘Anurag’, 50000);**

**Case 2:**

**INSERT INTO EMPLOYEE1 (Id, Salary, Name) VALUES (2, 35000, ‘Mohanty’);**

**Case 3:**

**INSERT INTO EMPLOYEE1 (Id, Name) VALUES (3, ‘Sambit’);**

**Case 4:**

**INSERT INTO EMPLOYEE1 (Id, Salary, Name) VALUES (NULL, NULL, NULL);**

## OUTPUT

Run SQL Command Line

```
SQL> CREATE  TABLE  Employee1
  2  (
  3  Id INT,
  4   Name  CHAR(20),
  5  Salary NUMBER(8, 2)
  6  );

Table created.

SQL> INSERT INTO EMPLOYEE1 VALUES (1, 'Anurag', 50000);

1 row created.

SQL> INSERT INTO EMPLOYEE1 (Id, Salary, Name) VALUES (2, 35000, 'Mohanty');

1 row created.

SQL> INSERT INTO EMPLOYEE1 (Id, Name) VALUES (3, 'Sambit');

1 row created.

SQL> INSERT INTO EMPLOYEE1 (Id, Salary, Name) VALUES (NULL, NULL, NULL);

1 row created.

SQL>
```

Run SQL Command Line

```
SQL> select * from employee1;

        ID NAME                     SALARY
---------- -------------------- ----------
         1 Anurag                    50000
         2 Mohanty                   35000
         3 Sambit


SQL> _
```

## 2.2 DML COMMANDS

## UPDATE COMMAND

The Employee table is created to execute the DML statements in Oracle.

```
CREATE  TABLE  Employee
(
Id INT,
 Name  CHAR(100),
Salary NUMBER(8, 2)
);
```

**Case 1:**

To update all the rows in the table.

**UPDATE EMPLOYEE1 SET Salary =500000;**

**Case 2:**

**To update rows based on the where condition.**

**UPDATE EMPLOYEE1 SET Name=’Test1′, Salary=55000 WHERE Id=3;**

**UPDATE EMPLOYEE1 SET Salary=85000 WHERE Name=’Anurag’;**

**OUTPUT:**

Run SQL Command Line

```
SQL> select * from employee1;

        ID NAME                    SALARY
---------- -------------------- ----------
         1 Anurag                    50000
         2 Mohanty                   35000
         3 Sambit


SQL> UPDATE EMPLOYEE1 SET Salary =500000;

4 rows updated.

SQL> UPDATE EMPLOYEE1 SET Name='Test1', Salary=55000 WHERE Id=3;

1 row updated.

SQL> UPDATE EMPLOYEE1 SET Salary=85000 WHERE Name='Anurag';

1 row updated.

SQL>
```

Run SQL Command Line

```
SQL> select * from employee1;

        ID NAME                    SALARY
---------- -------------------- ----------
         1 Anurag                    85000
         2 Mohanty                  500000
         3 Test1                     55000
                                    500000

SQL>
```

## 2.3 DML COMMANDS

### DELETE COMMAND

The Employee table is created to execute the DML statements in Oracle.

```
CREATE  TABLE  Employee1
(
Id INT,
 Name  CHAR(100),
Salary NUMBER(8, 2)
);
```

**Case 1:**

To delete all the rows from the table.

**DELETE FROM EMPLOYEE1;**

**Case 2:**

**To delete rows based on the where condition.**

**DELETE FROM EMPLOYEE1 WHERE Id=5;**

**OUTPUT:**

Run SQL Command Line

```
SQL> select * from employee1;

        ID NAME                    SALARY
---------- -------------------- ----------
         1 Anurag                    85000
         2 Mohanty                  500000
         3 Test1                     55000
                                    500000

SQL> DELETE FROM EMPLOYEE1 WHERE Id=3;

1 row deleted.

SQL> select * from employee1;

        ID NAME                    SALARY
---------- -------------------- ----------
         1 Anurag                    85000
         2 Mohanty                  500000
                                    500000

SQL> _
```

### 3.1 SQL QUERIES

SELECT * FROM Student1;

Run SQL Command Line

```
SQL> select * from student1;

    ROLLNO SNAME             AGE COURSE
---------- ---------- ---------- --------------------
         6 Anu                 23 CS
         4 Chetan              20 BCA
         5 Nihal               19 BBA
         8 Arpit               21 CS

SQL> _
```

SELECT customer_id, name, credit_limit FROM customers;

Run SQL Command Line

```
SQL> SELECT    customer_id,    name,    credit_limit FROM    customers;

CUSTOMER_ID NAME                                              CREDIT_LIMIT
----------- ------------------------------------------------ ------------
        177 United Continental Holdings                              5000
        180 INTL FCStone                                             5000
        184 Publix Super Markets                                     1200
        187 ConocoPhillips                                           2400
        190 3M                                                       1200
        192 Exelon                                                    500
        208 Tesoro                                                    500
        207 Northwestern Mutual                                      3600
        200 Enterprise Products Partners                             2400
        204 Rite Aid                                                 3600
        212 Qualcomm                                                  500
```

SELECT name, address, credit_limit FROM customers ORDER BY name ASC;

Run SQL Command Line

```
SQL> SELECT    name,   address, credit_limit  FROM    customers   ORDER BY    name ASC;

NAME
-------------------------------------------------
ADDRESS                                           CREDIT_LIMIT
------------------------------------------------- ------------
3M
Via Frenzy 6903, Roma,                                    1200

ADP
Langstr 14, Zuerich, ZH                                    700

AECOM
2102 E Kimberly Rd, Davenport, IA                          500
```

SELECT DISTINCT first_name FROM contacts ORDER BY first_name;

Run SQL Command Line

```
SQL> SELECT  DISTINCT  first_name FROM  contacts ORDER BY  first_name;

FIRST_NAME
--------------------------------------------------------------------------------
Aaron
Adah
Adam
Adrienne
Agustina
Al
Aleshia
Alessandra
Alexandra
Alvaro
Alysa
```

SELECT product_name, list_price FROM products WHERE list_price > 500;

Run SQL Command Line

```
SQL> SELECT    product_name,    list_price FROM    products WHERE    list_price > 500;

PRODUCT_NAME
--------------------------------------------------------------------------------
LIST_PRICE
----------
Intel Xeon E5-2699 V3 (OEM/Tray)
   3410.46

Intel Xeon E5-2697 V3
   2774.98

Intel Xeon E5-2698 V3 (OEM/Tray)
   2660.72
```

SELECT product_name, list_price FROM products WHERE list_price **BETWEEN** 650 **AND** 680 ORDER BY list_price;

Run SQL Command Line

```
SQL> SELECT    product_name,    list_price FROM    products WHERE    list_price BETWEEN 650   AND 680 ORDER BY    list_price;

PRODUCT_NAME
--------------------------------------------------------------------------------
LIST_PRICE
----------
Kingston
     653.5

Corsair Dominator Platinum
    659.99

Intel Core i7-3930K
      660
```

SELECT order_id, SUM( unit_price * quantity ) order_value FROM order_items

GROUP BY order_id HAVING SUM( unit_price * quantity ) > 1000000

ORDER BY order_value DESC;

Run SQL Command Line

```
SQL> SELECT    order_id,    SUM( unit_price * quantity ) order_value FROM    order_items
  2  GROUP BY     order_id  HAVING  SUM( unit_price * quantity ) > 1000000
  3  ORDER BY     order_value DESC;

  ORDER_ID ORDER_VALUE
---------- -----------
        70  1278962.17
        46  1269323.77
        78  1198331.59
         1  1143716.87
        68  1088670.12
        27  1084871.49
        32  1081679.88
        92  1050939.97
        59  1043144.72

9 rows selected.

SQL>
```

### 3.2 SQL SUBQUERIES

SELECT product_id, product_name, list_price FROM products

WHERE list_price = (SELECT MAX( list_price ) FROM products);

Run SQL Command Line

```
SQL> SELECT    product_id,    product_name,    list_price  FROM    products
  2  WHERE    list_price =    (SELECT  MAX( list_price )  FROM   products);

PRODUCT_ID
----------
PRODUCT_NAME
--------------------------------------------------------------------------------
LIST_PRICE
----------
        50
Intel SSDPECME040T401
   8867.99


SQL> _
```

SELECT product_name, list_price, category_id,
ROUND ( ( SELECT AVG( list_price ) FROM products p1
WHERE p1. category_id = p2.category_id ), 2) avg_list_price
FROM products p2 ORDER BY product_name;

Run SQL Command Line

```
SQL> SELECT    product_name,    list_price,  category_id,
  2   ROUND  ( ( SELECT  AVG( list_price )  FROM   products p1
  3         WHERE  p1. category_id = p2.category_id ), 2) avg_list_price
  4  FROM    products p2  ORDER BY    product_name;

PRODUCT_NAME
--------------------------------------------------------------------------------
LIST_PRICE CATEGORY_ID AVG_LIST_PRICE
---------- ----------- --------------
ADATA ASU800SS-128GT-C
     52.65           5         635.22

ADATA ASU800SS-512GT-C
    136.69           5         635.22

AMD 100-5056062
   1499.99           2         1406.1
```

## 3.3 AGGREGATE FUNCTIONS

SELECT MIN( list_price ) FROM products;

Run SQL Command Line

```
SQL> SELECT   MIN( list_price ) FROM    products;

MIN(LIST_PRICE)
---------------
          15.55

SQL>
```

SELECT MAX( list_price ) FROM products;

Run SQL Command Line

```
SQL> SELECT    MAX( list_price ) FROM    products;

MAX(LIST_PRICE)
---------------
        8867.99

SQL>
```

SELECT COUNT(*) FROM products;

Run SQL Command Line

```
SQL> SELECT    COUNT(*) FROM    products;

  COUNT(*)
----------
       288

SQL>
```

SELECT ROUND(AVG( standard_cost ), 2) avg_std_cost FROM products;

Run SQL Command Line

```
SQL> SELECT    ROUND(AVG( standard_cost ), 2) avg_std_cost  FROM    products;

AVG_STD_COST
------------
      727.62

SQL>
```

SELECT SUM( quantity ) FROM order_items;

Run SQL Command Line

```
SQL> SELECT    SUM( quantity ) FROM    order_items;

SUM(QUANTITY)
-------------
        59606

SQL>
```

**4.1 Zero_Divide Exception**

# PROCEDURE:

```
DECLARE

   a int:=10;
   b int:=0;
   answer int;

 BEGIN
   answer:=a/b;
   dbms_output.put_line('the result after division is'||answer);

EXCEPTION

   WHEN zero_divide THEN
      dbms_output.put_line('dividing by zero please check the values again');
      dbms_output.put_line('the value of a is '||a);
      dbms_output.put_line('the value of b is '||b);

END;
```

## OUTPUT:

Run SQL Command Line

```
SQL> conn
Enter user-name: system
Enter password:
Connected.
SQL> set serveroutput on
SQL> DECLARE
  2
  3     a int:=10;
  4     b int:=0;
  5     answer int;
  6
  7   BEGIN
  8     answer:=a/b;
  9     dbms_output.put_line('the result after division is'||answer);
 10
 11  EXCEPTION
 12
 13     WHEN zero_divide THEN
 14        dbms_output.put_line('dividing by zero please check the values again');
 15        dbms_output.put_line('the value of a is '||a);
 16        dbms_output.put_line('the value of b is '||b);
 17
 18  END;
 19  /
dividing by zero please check the values again
the value of a is 10
the value of b is 0

PL/SQL procedure successfully completed.

SQL>
```

## 4.2 NO_DATA_FOUND Exception

**PROCEDURE:**

```
DECLARE

   l_name customers.NAME%TYPE;
   l_customer_id customers.customer_id%TYPE := &customer_id;

BEGIN

   SELECT NAME INTO l_name
   FROM customers
   WHERE customer_id = l_customer_id;

   dbms_output.put_line('customer name is ' || l_name);

   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         dbms_output.put_line('Customer ' || l_customer_id ||  ' does not exist');
END;
```

**OUTPUT:**

Run SQL Command Line

```
SQL> DECLARE
  2
  3      l_name customers.NAME%TYPE;
  4      l_customer_id customers.customer_id%TYPE := &customer_id;
  5
  6  BEGIN
  7
  8      SELECT NAME INTO l_name
  9      FROM customers
 10      WHERE customer_id = l_customer_id;
 11
 12      dbms_output.put_line('customer name is ' || l_name);
 13
 14      EXCEPTION
 15          WHEN NO_DATA_FOUND THEN
 16              dbms_output.put_line('Customer ' || l_customer_id ||  ' does not exist');

 17  END;
 18  /
Enter value for customer_id: 0
old   4:     l_customer_id customers.customer_id%TYPE := &customer_id;
new   4:     l_customer_id customers.customer_id%TYPE := 0;
Customer 0 does not exist

PL/SQL procedure successfully completed.

SQL>
```

## 4.3 TOO_MANY_ROWS Exception

**PROCEDURE:**

```
DECLARE

   l_name customers.NAME%TYPE;
   l_customer_id customers.customer_id%TYPE := &customer_id;

BEGIN

   SELECT NAME INTO l_name  FROM customers
   WHERE customer_id > l_customer_id;

   dbms_output.put_line('Customer name is ' || l_name);

EXCEPTION

      WHEN NO_DATA_FOUND THEN
         dbms_output.put_line('Customer ' || l_customer_id ||  ' does not exist');
      WHEN TOO_MANY_ROWS THEN
         dbms_output.put_line('The database returns more than one customer');

END;
```

**OUTPUT:**

Run SQL Command Line

```
SQL> DECLARE
  2      l_name customers.NAME%TYPE;
  3      l_customer_id customers.customer_id%TYPE := &customer_id;
  4  BEGIN
  5      -- get the customer
  6      SELECT NAME INTO l_name
  7      FROM customers
  8      WHERE customer_id > l_customer_id;
  9
 10      -- show the customer name
 11      dbms_output.put_line('Customer name is ' || l_name);
 12      EXCEPTION
 13          WHEN NO_DATA_FOUND THEN
 14              dbms_output.put_line('Customer ' || l_customer_id ||  ' does not exist');

 15          WHEN TOO_MANY_ROWS THEN
 16              dbms_output.put_line('The database returns more than one customer');
 17  END;
 18  /
Enter value for customer_id: 10
old   3:      l_customer_id customers.customer_id%TYPE := &customer_id;
new   3:      l_customer_id customers.customer_id%TYPE := 10;
The database returns more than one customer

PL/SQL procedure successfully completed.

SQL>
```

**5.1 PLSQL CURSORS – IMPLICIT CURSORS**

**PROCEDURE:**

```
DECLARE
  total_rows number(2);

BEGIN
  UPDATE customerinfo
  SET salary = salary + 100;

  IF sql%notfound THEN
    dbms_output.put_line('no customers selected');

  ELSIF sql%found THEN
    total_rows := sql%rowcount;
    dbms_output.put_line( total_rows || ' customers selected ');

  END IF;

END;
```

**OUTPUT:**

Run SQL Command Line

```
SQL> set serveroutput on
SQL> DECLARE
  2     total_rows number(2);
  3  BEGIN
  4     UPDATE customerinfo
  5     SET salary = salary + 100;
  6     IF sql%notfound THEN
  7        dbms_output.put_line('no customers selected');
  8     ELSIF sql%found THEN
  9        total_rows := sql%rowcount;
 10        dbms_output.put_line( total_rows || ' customers selected ');
 11     END IF;
 12  END;
 13  /
Old salary: 8300
New salary: 8400
Salary difference: 100
1 customers selected

PL/SQL procedure successfully completed.

SQL> select * from customerinfo;

        ID NAME                       AGE ADDRESS             SALARY
---------- -------------------- ---------- --------------- ----------
         7 Kriti                         22 HP                    8400

SQL> _
```

## 5.2 PLSQL CURSORS – EXPLICIT CURSORS

**PROCEDURE:**

```
DECLARE

   CURSOR customer_cur is
      SELECT id, name, address
      FROM customerinfo;

   customer_rec customer_cur%rowtype;

BEGIN
   OPEN customer_cur;

  LOOP

      FETCH customer_cur into customer_rec;
      EXIT WHEN customer_cur%notfound;

      DBMS_OUTPUT.put_line(customer_rec.id || ' ' || customer_rec.name);

   END LOOP;

END;
```

**OUTPUT:**

Run SQL Command Line

```
SQL> set serveroutput on
SQL> DECLARE
  2
  3     CURSOR customer_cur is
  4        SELECT id, name, address
  5        FROM customerinfo;
  6
  7     customer_rec customer_cur%rowtype;
  8
  9  BEGIN
 10     OPEN customer_cur;
 11
 12    LOOP
 13
 14        FETCH customer_cur into customer_rec;
 15        EXIT WHEN customer_cur%notfound;
 16
 17        DBMS_OUTPUT.put_line(customer_rec.id || ' ' || customer_rec.name);
 18
 19     END LOOP;
 20
 21  END;
 22  /
7 Kriti
1 Ramesh
2 Khilan
3 Koushik

PL/SQL procedure successfully completed.

SQL>
```

**6. PLSQL - TRIGGERS**

**PROCEDURE:**

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customerinfo
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
   sal_diff number;
BEGIN
   sal_diff := :NEW.salary  - :OLD.salary;
   dbms_output.put_line('Old salary: ' || :OLD.salary);
   dbms_output.put_line('New salary: ' || :NEW.salary);
   dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

**OUTPUT:**

Run SQL Command Line

```
SQL> CREATE OR REPLACE TRIGGER display_salary_changes
  2  BEFORE DELETE OR INSERT OR UPDATE ON customerinfo
  3  FOR EACH ROW
  4  WHEN (NEW.ID > 0)
  5  DECLARE
  6     sal_diff number;
  7  BEGIN
  8     sal_diff := :NEW.salary  - :OLD.salary;
  9     dbms_output.put_line('Old salary: ' || :OLD.salary);
 10     dbms_output.put_line('New salary: ' || :NEW.salary);
 11     dbms_output.put_line('Salary difference: ' || sal_diff);
 12  END;
 13
 14  /

Trigger created.

SQL> INSERT INTO CUSTOMERINFO(ID,NAME,AGE,ADDRESS,SALARY)
  2  VALUES (8, 'Kriti', 22, 'HP', 7500.00 );
Old salary:
New salary: 7500
Salary difference:

1 row created.

SQL> UPDATE customerinfo SET salary = salary + 500  WHERE id = 7
  2  ;
Old salary: 8400
New salary: 8900
Salary difference: 500

1 row updated.

SQL>
```

**7. PLSQL PACKAGES**

**PROCEDURE:**

**PL/SQL code for Package specification:**

```
CREATE OR REPLACE PACKAGE pkg_student IS
        PROCEDURE  updateRecord(sno student1.rollno%type);
        FUNCTION deleteRecord(snm student1.sname%type)
                RETURN boolean;
END pkg_student;
```

**PL/SQL code for Package Body:**

```
CREATE OR REPLACE PACKAGE BODY pkg_student IS
        PROCEDURE updateRecord(sno student1.rollno%type) IS
                BEGIN
                        Update student1 set age=23 where rollno=sno;
                        IF  SQL%FOUND THEN
                                dbms_output.put_line('RECORD UPDATED');
                        ELSE
                                dbms_output.put_line('RECORD NOT FOUND');
                        END IF;
                END updateRecord;

        FUNCTION deleteRecord(snm student1.sname%type) RETURN boolean IS
                BEGIN
                        Delete from student1 where sname=snm;
                        RETURN SQL%FOUND;
                END deleteRecord;
END pkg_student;
```

**PL/SQL code for calling the Procedure and Function used in Package.**

```
DECLARE
        sno student1.rollno%type;
        s_age student1.age%type;
        snm student1.sname%type;
BEGIN
        sno := &sno;
        snm := &snm;
        pkg_student.updateRecord(sno);
        IF pkg_student.deleteRecord(snm) THEN
                dbms_output.put_line('RECORD DELETED');
        ELSE
                dbms_output.put_line('RECORD NOT FOUND');
        END IF;
END;
```

**OUTPUT:**

## Package Specification Creation

Run SQL Command Line

```
SQL> CREATE OR REPLACE PACKAGE pkg_student IS
  2  PROCEDURE  updateRecord(sno student1.rollno%type);
  3  FUNCTION deleteRecord(snm student1.sname%type)
  4  RETURN boolean;
  5  END pkg_student;
  6
  7  /

Package created.
```

## Package Body Creation

Run SQL Command Line

```
SQL> CREATE OR REPLACE PACKAGE BODY pkg_student IS
  2  PROCEDURE updateRecord(sno student1.rollno%type) IS
  3  BEGIN
  4  Update student1 set age=23 where rollno=sno;
  5  IF  SQL%FOUND THEN
  6  dbms_output.put_line('RECORD UPDATED');
  7  ELSE
  8  dbms_output.put_line('RECORD NOT FOUND');
  9  END IF;
 10  END updateRecord;
 11
 12  FUNCTION deleteRecord(snm student1.sname%type) RETURN boolean IS
 13  BEGIN
 14  Delete from student1 where sname=snm;
 15  RETURN SQL%FOUND;
 16  END deleteRecord;
 17  END pkg_student;
 18  /

Package body created.

SQL>
```

**Student1 table before updating & Calling the Package**

Run SQL Command Line

```
SQL> select * from student1;

    ROLLNO SNAME              AGE COURSE
---------- ---------- ---------- --------------------
         6 Anu                 18 CS
         4 Chetan              20 BCA
         5 Nihal               19 BBA
         7 Kamal               20 CS
         8 Arpit               21 CS

SQL> DECLARE
  2  sno student1.rollno%type;
  3  s_age student1.age%type;
  4  snm student1.sname%type;
  5  BEGIN
  6  sno := &sno;
  7  snm := &snm;
  8  pkg_student.updateRecord(sno);
  9  IF pkg_student.deleteRecord(snm) THEN
 10  dbms_output.put_line('RECORD DELETED');
 11  ELSE
 12  dbms_output.put_line('RECORD NOT FOUND');
 13  END IF;
 14  END;
 15  /
Enter value for sno: 6
old   6: sno := &sno;
new   6: sno := 6;
Enter value for snm: 'Kamal'
old   7: snm := &snm;
new   7: snm := 'Kamal';
RECORD UPDATED
RECORD DELETED

PL/SQL procedure successfully completed.

SQL>
```

**Updated Student1 table after calling pkg_student package**

Run SQL Command Line

```
SQL> select * from student1;

    ROLLNO SNAME             AGE COURSE
---------- ---------- ---------- --------------------
         6 Anu                 23 CS
         4 Chetan              20 BCA
         5 Nihal               19 BBA
         8 Arpit               21 CS

SQL>
```

**8. Student Marksheet Processing**

**mystud.html**

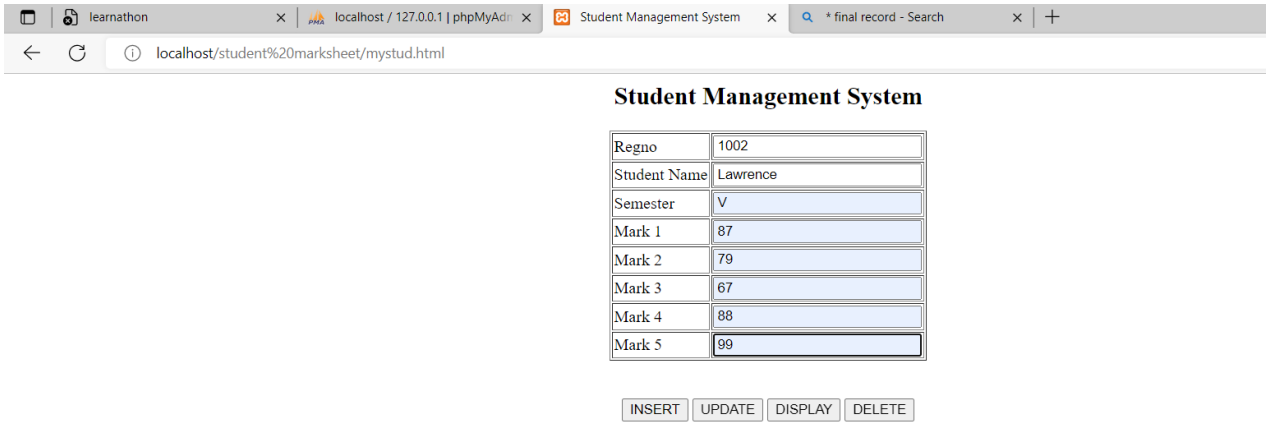
```
<html>
<head>
<title> Student Management System </title>
<center> <h2> Student Management System </h2></center>
<form name="std" method="post" action="mystud.php">
<center><table border="1" font-size=13 >
<tr>
<td> Regno </td><td> <input type="text" name="regno" size="25"></td></tr>
<tr>
<td> Student Name</td><td> <input type="text" name="studname"
size="25"></td></tr>
<tr>
<td> Semester</td><td> <input type="text" name="sem" size="25"></td></tr>
<tr>
<td> Mark 1</td><td> <input type="text" name="mark1" size="25"></td></tr>
<tr>
<td> Mark 2 </td><td><input type="text" name="mark2" size="25"></td></tr>
<tr>
<td> Mark 3</td><td> <input type="text" name="mark3" size="25"></td></tr>
<tr>
<td> Mark 4</td><td> <input type="text" name="mark4" size="25"></td></tr>
<tr>
<td> Mark 5</td><td><input type="text" name="mark5" size="25"></td></tr>
</table>
<br><br>
</center>
<center>
<input type="submit" name="btn_submit" value="INSERT">
<input type="submit" name="btn_submit" value="UPDATE">
<input type="submit" name="btn_submit" value="DISPLAY">
<input type="submit" name="btn_submit" value="DELETE">
</center>
</form>
</body>
</html>
```

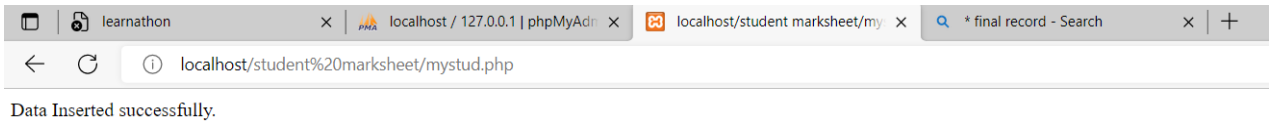
**mystud.php**

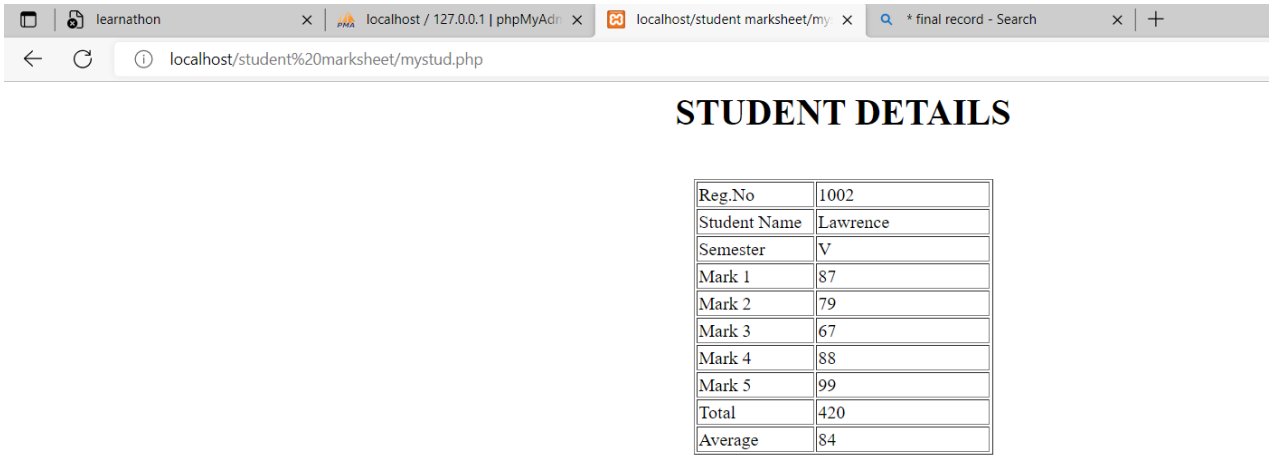
```
<?php
$regno = $_POST['regno'];
$studentname = $_POST['studname'];
$semester = $_POST['sem'];
$mark1 = $_POST['mark1'];
$mark2 = $_POST['mark2'];
$mark3 = $_POST['mark3'];
$mark4 = $_POST['mark4'];
$mark5 = $_POST['mark5'];

$btntype=$_POST['btn_submit'];
//database connection
$conn = new mysqli("localhost","root","","studentdb");
if($conn->connect_error)
die('connection failed : '.$conn->connect_error);

if ($btntype=="INSERT")
{
$tot=$mark1+$mark2+$mark3+$mark4+$mark5;
$avg=$tot/5;

$sql = "insert into stu
values($regno,'$studentname','$semester',$mark1,$mark2,$mark3,$mark4,$mark5,$
tot,$avg)";

if($conn->query($sql))
echo "Data Inserted successfully.";
}
elseif ($btntype=="UPDATE")
{
$tot=$mark1+$mark2+$mark3+$mark4+$mark5;
$avg=$tot/5;

$sql = "UPDATE stu SET
m1='$mark1',m2='$mark2',m3='$mark3',m4='$mark4',m5='$mark5',tot='$tot',avg='
$avg' WHERE regno='$regno'";
if($conn->query($sql))
echo "Data updated successfully.";
}
```

```
elseif ($btntype=="DELETE")
{
$sql = "delete from stu where regno='$regno'";
if($conn->query($sql))
echo "Data deleted successfully.";
}
elseif ($btntype=="DISPLAY")
{
$sql = "select * from stu where regno='$regno'";
if($result = $conn->query($sql))
{
$row = $result->fetch_assoc();
if ($result->num_rows > 0)
{
echo "<Center><h1>STUDENT DETAILS</h1><br><font font-size=20><table
border=1>";
echo "<tr><td width=100>","Reg.No","</td><td width=150>",$row["regno"],
"</td></tr>";
echo "<tr><td>","Student Name ","</td><td>",$row["stuname"] ,"</td></tr>" ;
echo "<tr><td>","Semester","</td><td>",$row["sem"],"</td></tr>" ;
echo "<tr><td>","Mark 1","</td><td>",$row["m1"], "</td></tr>";
echo "<tr><td>","Mark 2","</td><td>",$row["m2"],"</td></tr>" ;
echo "<tr><td>","Mark 3","</td><td>",$row["m3"],"</td></tr>" ;
echo "<tr><td>","Mark 4","</td><td>",$row["m4"],"</td></tr>" ;
echo "<tr><td>","Mark 5","</td><td>",$row["m5"],"</td></tr>" ;
echo "<tr><td>","Total","</td><td>",$row["tot"],"</td></tr>" ;
echo "<tr><td>","Average","</td><td>",$row["avg"],"</td></tr>" ;
echo "</table></h3></center>";
}
else{
echo "<Center><h1>STUDENT DETAILS NOT AVAILABLE</h1></Center>";
}}}?>
```

# OUTPUT


learnathon
localhost / 127.0.0.1 | phpMyAdm
Student Management System
* final record - Search
localhost/student%20marksheet/mystud.html
Student Management System
Regno 1002
Student Name Lawrence
Semester V
Mark 1 87
Mark 2 79
Mark 3 67
Mark 4 88
Mark 5 99
INSERT
UPDATE
DISPLAY
DELETE


learnathon
localhost / 127.0.0.1 | phpMyAdm
localhost/student marksheet/my
* final record - Search
localhost/student%20marksheet/mystud.php
Data Inserted successfully.


learnathon
localhost / 127.0.0.1 | phpMyAdm
localhost/student marksheet/my
* final record - Search
localhost/student%20marksheet/mystud.php
STUDENT DETAILS
Reg.No 1002
Student Name Lawrence
Semester V
Mark 1 87
Mark 2 79
Mark 3 67
Mark 4 88
Mark 5 99
Total 420
Average 84

## 9. EMPLOYEE PAYROLL PROCESS

**payroll.html**

```
<html>
<head>
<title> EMPLOYEE PAYROLL PROCESS </title>
<center> <h2> EMPLOYEE PAYROLL PROCESS </h2></center>
<form  method="post" action="Payroll.php">
<center><table border="1" >
<tr>
<td> Empno </td><td> <input type="text" name="empno"
size="25"></td></tr>
<tr>
<td> Employee Name</td><td> <input type="text" name="ename"
size="25"></td></tr>
<tr>
<td> Designation</td><td> <input type="text" name="des"
size="25"></td></tr>
<tr>
<td> Basic Pay</td><td> <input type="text" name="bp"
size="25"></td></tr>
<tr>
<td> DA </td><td><input type="text" name="da" size="25"></td></tr>
<tr>
<td> HRA </td><td> <input type="text" name="hra" size="25"></td></tr>
<tr>
<td> PF </td><td> <input type="text" name="pf" size="25"></td></tr>
</table><br><br></center>
<center>
<input type="submit" name="btn_submit" value="INSERT">
<input type="submit" name="btn_submit" value="UPDATE">
<input type="submit" name="btn_submit" value="DISPLAY">
<input type="submit" name="btn_submit" value="DISPLAY ALL">
<input type="submit" name="btn_submit" value="DELETE">
</center>
</form>
</body>
</html>
```

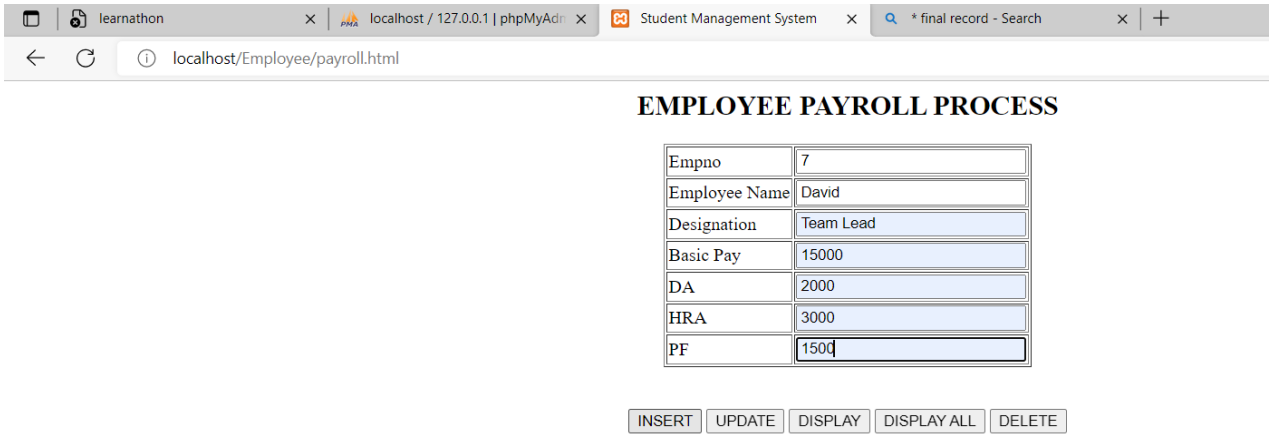
**payroll.php**

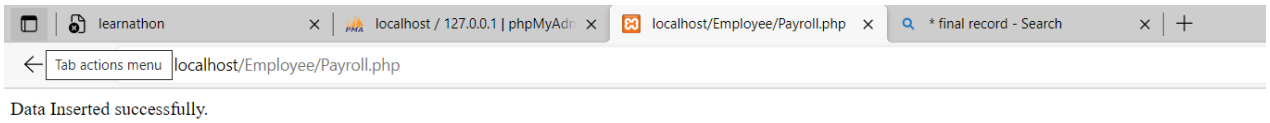
```
<?php
$eno = $_POST['empno'];
$ename = $_POST['ename'];
$desig = $_POST['des'];
$bp = $_POST['bp'];
$da = $_POST['da'];
$hra = $_POST['hra'];
$pf = $_POST['pf'];
$btntype=$_POST['btn_submit'];

//database connection
$conn = new mysqli("localhost","root","","studentdb");
if($conn->connect_error)
die('connection failed : '.$conn->connect_error);

if ($btntype=="INSERT")
{
$gp = $bp+$da+$hra;
$np = $gp-$pf;

$sql = "insert into emp
values($eno,'$ename','$desig',$bp,$da,$hra,$pf,$gp,$np)";

if($conn->query($sql))
echo "Data Inserted successfully.";
}
elseif ($btntype=="UPDATE")
{
$gp = $bp+$da+$hra;
$np = $gp-$pf;

$sql = "UPDATE emp SET
eno='$eno',ename='$ename',desig='$desig',bp='$bp',da='$da',hra='$hra',pf='$
pf',gp='$gp',np='$np'  WHERE eno='$eno'";

if($conn->query($sql))
echo "Data updated successfully.";
}
elseif ($btntype=="DELETE")
{
$sql = "delete from emp where eno='$eno'";
if($conn->query($sql))
```

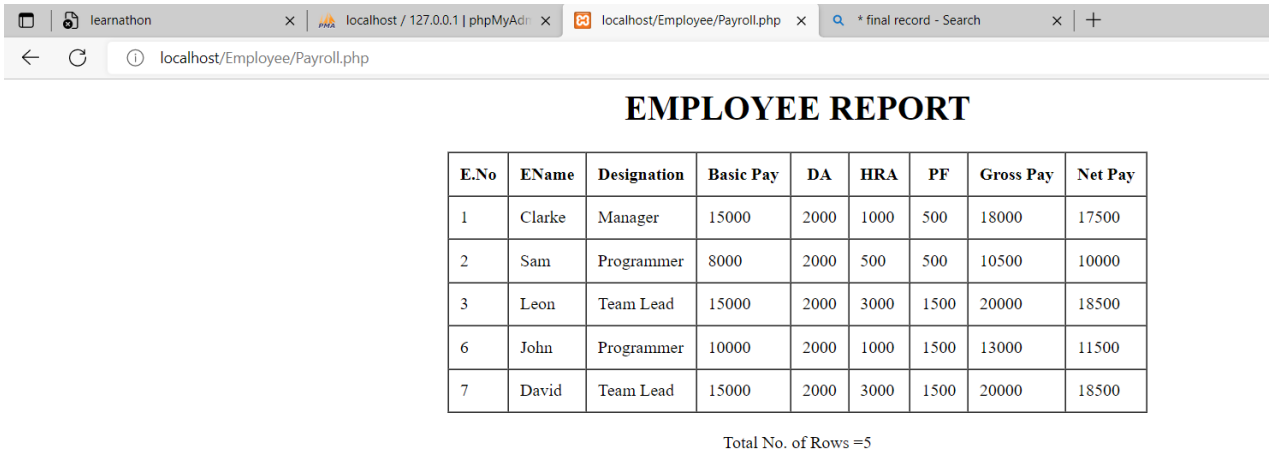
```
echo "Data deleted successfully.";
}
elseif ($btntype=="DISPLAY")
{
$sql = "select * from emp where eno='$eno'";
if($result = $conn->query($sql))
{
$row = $result->fetch_assoc();
if ($result->num_rows > 0)
{
echo "<Center><h1>EMPLOYEE DETAILS</h1><br><font font-
size=20><table border=1>";
echo "<tr><td width=100>","Emp.No","</td><td width=150>",$row["eno"],
"</td></tr>";
echo "<tr><td>","Employee Name ","</td><td>",$row["ename"]
,"</td></tr>" ;
echo "<tr><td>","Designation","</td><td>",$row["desig"],"</td></tr>" ;
echo "<tr><td>","Basic Pay","</td><td>",$row["bp"], "</td></tr>";
echo "<tr><td>","Dearness Allowance","</td><td>",$row["da"],"</td></tr>"
;
echo "<tr><td>","HRA","</td><td>",$row["hra"],"</td></tr>" ;
echo "<tr><td>","PF","</td><td>",$row["pf"],"</td></tr>" ;
echo "<tr><td>","Gross Pay","</td><td>",$row["gp"],"</td></tr>" ;
echo "<tr><td>","Net Pay","</td><td>",$row["np"],"</td></tr>" ;
echo "</table></h3></center>";
}
else
{
echo "<Center><h1>EMPLOYEE DETAILS NOT
AVAILABLE</h1></Center>";
}
}
}
elseif($btntype=="DISPLAY ALL")
{
$query = "SELECT * FROM emp";
$result = mysqli_query($conn, $query);
echo "<Center><h1>EMPLOYEE REPORT</h1></Center>";
echo "<center><table border =1 cellspacing=0 cellpadding=10>";
echo "<tr><th>E.No</th><th>EName</th><th>Designation</th><th>Basic
Pay</th>";
echo"<th>DA</th><th>HRA</th><th>PF</th><th>Gross Pay</th><th>Net
Pay</th></tr>";
$sn=0;
if ($result->num_rows  > 0) {
```

```
while($row = $result->fetch_assoc()) {
echo "<tr><td>", $row['eno'], "</td>";
echo "<td>",$row['ename'], "</td>";
echo "<td>", $row['desig'],"</td>";
echo "<td>", $row['bp'],"</td>";
echo "<td>", $row['da'],"</td>";
echo "<td>", $row['hra'],"</td>";
echo "<td>", $row['pf'],"</td>";
echo "<td>", $row['gp'],"</td>";
echo "<td>", $row['np'],"</td><tr></center>";
$sn++;
}
echo "</table>";
}
else {
echo "<tr><td>","No data found","</td></tr>";}
echo "<br>Total No. of Rows =", $sn;
}
?>
```

# OUTPUT:


learnathon
localhost / 127.0.0.1 | phpMyAdm
Student Management System
* final record - Search
localhost/Employee/payroll.html
EMPLOYEE PAYROLL PROCESS
Empno
7
Employee Name
David
Designation
Team Lead
Basic Pay
15000
DA
2000
HRA
3000
PF
1500
INSERT
UPDATE
DISPLAY
DISPLAY ALL
DELETE


learnathon
localhost / 127.0.0.1 | phpMyAdm
localhost/Employee/Payroll.php
* final record - Search
Tab actions menu
localhost/Employee/Payroll.php
Data Inserted successfully.


learnathon
localhost / 127.0.0.1 | phpMyAdm
localhost/Employee/Payroll.php
* final record - Search
localhost/Employee/Payroll.php
EMPLOYEE REPORT
E.No | EName | Designation | Basic Pay | DA | HRA | PF | Gross Pay | Net Pay
1 | Clarke | Manager | 15000 | 2000 | 1000 | 500 | 18000 | 17500
2 | Sam | Programmer | 8000 | 2000 | 500 | 500 | 10500 | 10000
3 | Leon | Team Lead | 15000 | 2000 | 3000 | 1500 | 20000 | 18500
6 | John | Programmer | 10000 | 2000 | 1000 | 1500 | 13000 | 11500
7 | David | Team Lead | 15000 | 2000 | 3000 | 1500 | 20000 | 18500
Total No. of Rows =5

## 10. Library Management System

## library.html

```
<html>
<head>
<title> Library Management System </title>
<center> <h2> Library Management System </h2></center>
<form  method="post" action="library.php">
<center><table border="1" >
<tr>
<td> Book Id </td><td> <input type="text" name="bookid" size="25"></td></tr>
<tr>
<td> Book Name</td><td> <input type="text" name="bname"
size="25"></td></tr>
<tr>
<td> Author</td><td> <input type="text" name="author" size="25"></td></tr>
<tr>
<td> Publisher</td><td> <input type="text" name="pub" size="25"></td></tr>
<tr>
<td> Price </td><td><input type="text" name="price" size="25"></td></tr>


</table>
<br><br>
</center>
<center>
<input type="submit" name="btn_submit" value="INSERT">
<input type="submit" name="btn_submit" value="UPDATE">
<input type="submit" name="btn_submit" value="DISPLAY">
<input type="submit" name="btn_submit" value="DISPLAY ALL">
<input type="submit" name="btn_submit" value="DELETE">
</center>
</form>
</body>
</html>
```

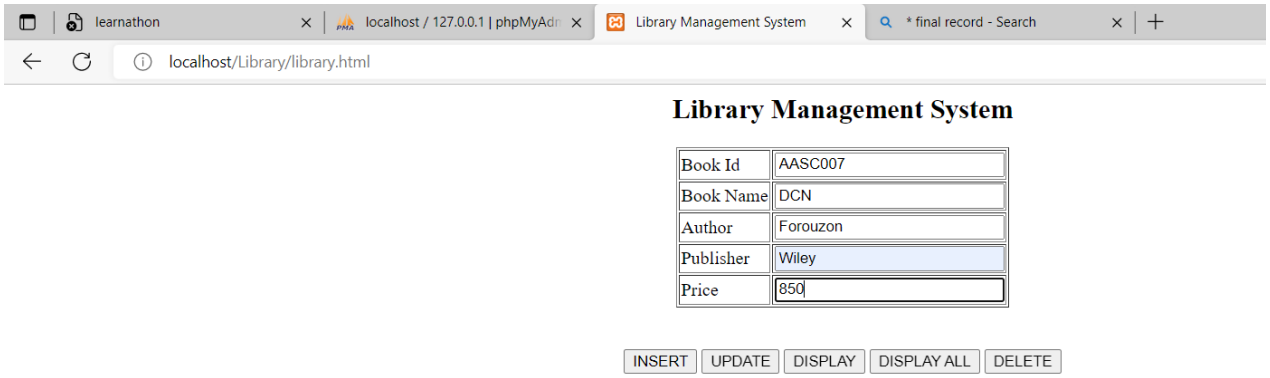
## library.php

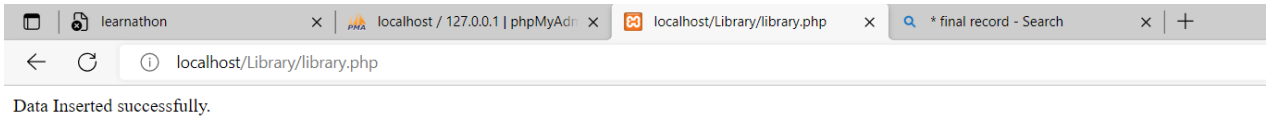
```
<?php
$bookid = $_POST['bookid'];
$bname = $_POST['bname'];
$author = $_POST['author'];
$pub= $_POST['pub'];
$price = $_POST['price'];



$btntype=$_POST['btn_submit'];
//database connection
$conn = new mysqli("localhost","root","","studentdb");
if($conn->connect_error)
die('connection failed : '.$conn->connect_error);

if ($btntype=="INSERT")
{


$sql = "insert into library values('$bookid','$bname','$author','$pub',$price)";

if($conn->query($sql))
echo "Data Inserted successfully.";
}
elseif ($btntype=="UPDATE")
{


$sql = "UPDATE library SET
bname='$bname',author='$author',publisher='$pub',price='$price' where bookid
='$bookid'";;

if($conn->query($sql))
echo "Data updated successfully.";
}
elseif ($btntype=="DELETE")
{

$sql = "delete from library where bookid='$bookid'";

if($conn->query($sql))
echo "Data deleted successfully.";
```

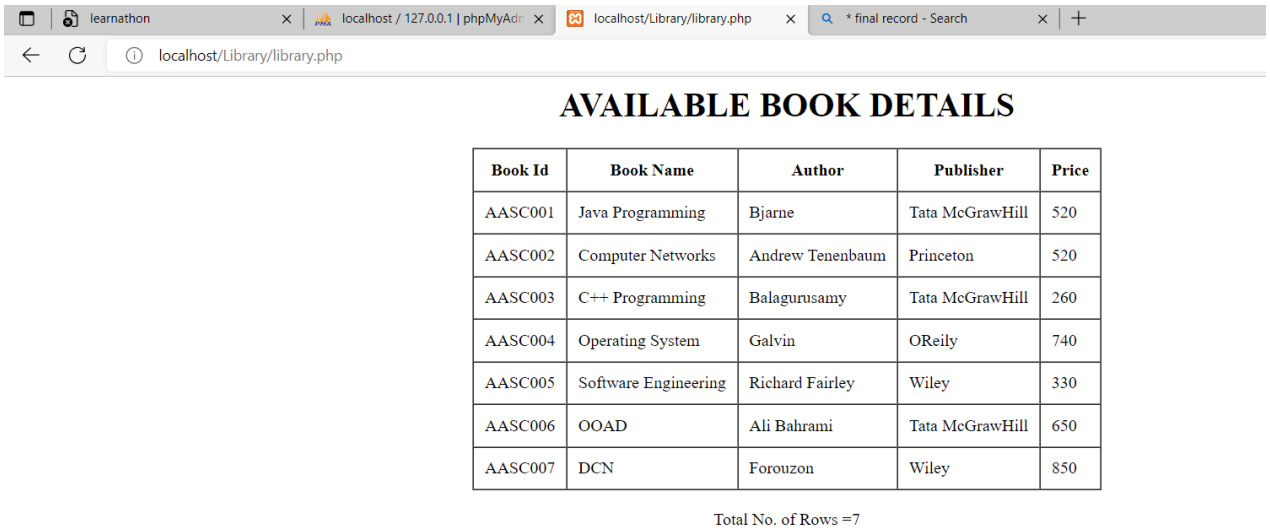
```
}

elseif ($btntype=="DISPLAY")
{
$sql = "select * from library where bookid='$bookid'";
if($result = $conn->query($sql))

{

$row = $result->fetch_assoc();
if ($result->num_rows > 0)
{

echo "<Center><h1>BOOK DETAILS<br><font font-size=20><table border=1>";
echo "<tr><td width=100>","Book Id","</td><td width=150>",$row["bookid"],
"</td></tr>";
echo "<tr><td>","Book Name ","</td><td>",$row["bname"] ,"</td></tr>" ;
echo "<tr><td>","Author Name","</td><td>",$row["author"],"</td></tr>" ;
echo "<tr><td>","Publisher","</td><td>",$row["publisher"], "</td></tr>";
echo "<tr><td>","Price","</td><td>",$row["price"],"</td></tr>" ;
echo "</table></center>";
}
else
{
echo "<Center><h1>BOOK DETAILS NOT AVAILABLE</h1></Center>";
}

}


}
elseif($btntype=="DISPLAY ALL")
{
$query = "SELECT * FROM library";
$result = mysqli_query($conn, $query);
echo "<Center><h1>AVAILABLE BOOK DETAILS</h1></Center>";
echo "<center><table border =1 cellspacing=0 cellpadding=10>";
echo "<tr><th>Book Id</th><th>Book
Name</th><th>Author</th><th>Publisher</th>";
echo"<th>Price</th></tr>";
$sn=0;
if ($result->num_rows  > 0) {

while($row = $result->fetch_assoc()) {
```

```
echo "<tr><td>", $row['bookid'], "</td>";
echo "<td>",$row['bname'], "</td>";
echo "<td>", $row['author'],"</td>";
echo "<td>", $row['publisher'],"</td>";
echo "<td>", $row['price'],"</td>";

$sn++;
}
echo "</table>";
}
else {
echo "<tr><td>","No data found","</td></tr>";}
echo "<br>Total No. of Rows =", $sn;
}
?>
```

# OUTPUT




| Book Id | Book Name | Author | Publisher | Price |
|---|---|---|---|---|
| AASC001 | Java Programming | Bjarne | Tata McGrawHill | 520 |
| AASC002 | Computer Networks | Andrew Tenenbaum | Princeton | 520 |
| AASC003 | C++ Programming | Balagurusamy | Tata McGrawHill | 260 |
| AASC004 | Operating System | Galvin | OReily | 740 |
| AASC005 | Software Engineering | Richard Fairley | Wiley | 330 |
| AASC006 | OOAD | Ali Bahrami | Tata McGrawHill | 650 |
| AASC007 | DCN | Forouzon | Wiley | 850 |