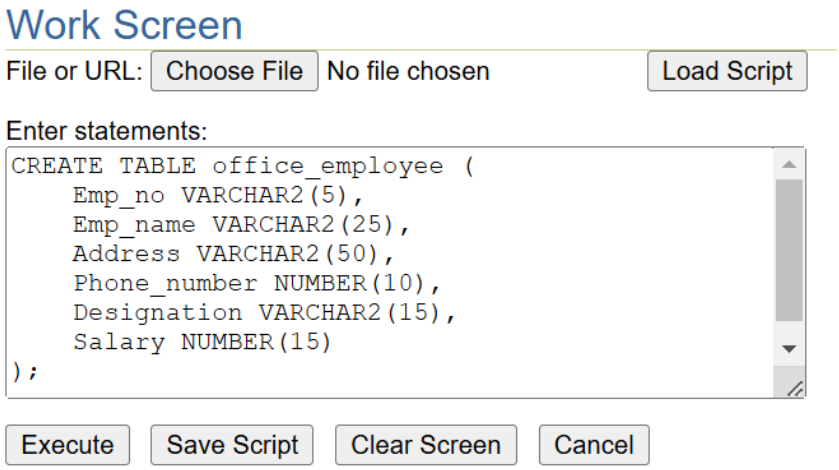
ADBMS (PL/SQL Practicals)

Practical 1 :- Write a SQL Query to create a table”employee”:

1. Display the structure of table.
2. Add qualification field at the end of employee table.
3. Modify the size of the name field 25 to 30.
4. Display the employee name whose salary is greater than 20,000.
5. Display the employee details whose name starts with —A‖.

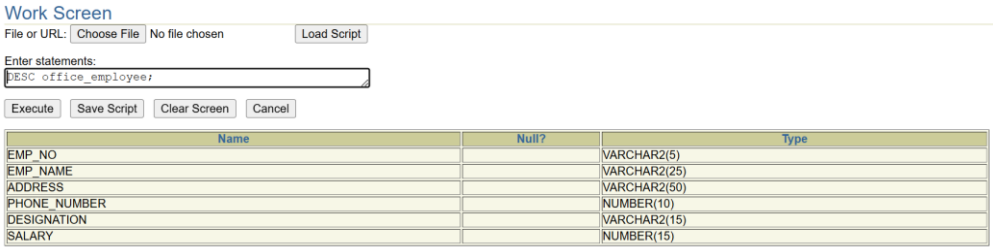
Queries & Output :-

```
CREATE TABLE office_employee (
   Emp_no VARCHAR2(5),
   Emp_name VARCHAR2(25),
   Address VARCHAR2(50),
   Phone_number NUMBER(10),
   Designation VARCHAR2(15),
   Salary NUMBER(15)
);
```



1. Display the structure of table.

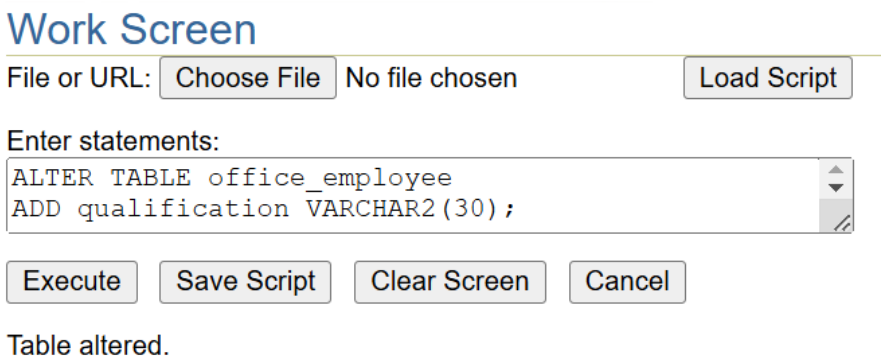
DESC office_employee;



Name	Null?	Type
EMP_NO		VARCHAR2(5)
EMP_NAME		VARCHAR2(25)
ADDRESS		VARCHAR2(50)
PHONE_NUMBER		NUMBER(10)
DESIGNATION		VARCHAR2(15)
SALARY		NUMBER(15)

2. Add qualification field at the end of employee table.

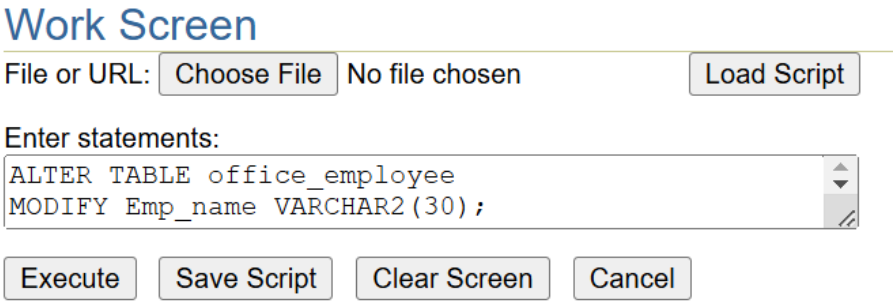
```
ALTER TABLE office_employee
ADD qualification VARCHAR2(30);
```

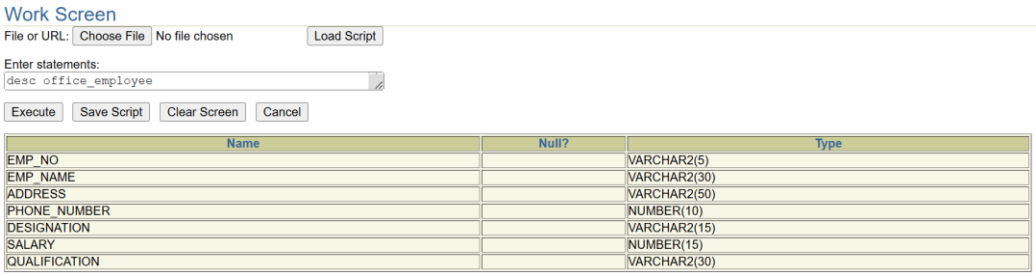


3. Modify the size of the name field 25 to 30.

ALTER TABLE office_employee

MODIFY Emp_name VARCHAR2(30);





INSERT INTO office_employee (Emp_no, Emp_name, Address, Phone_number, Designation, Salary, Qualification)

VALUES ('E001', 'Alice Johnson', '123 Main St, Cityville', 1234567890, 'Manager', 30000, 'MBA');

INSERT INTO office_employee (Emp_no, Emp_name, Address, Phone_number, Designation, Salary, Qualification)

VALUES ('E002', 'Bob Adams', '456 Oak St, Townsville', 9876543210, 'Developer', 25000, 'B.Tech');

INSERT INTO office_employee (Emp_no, Emp_name, Address, Phone_number, Designation, Salary, Qualification)

VALUES ('E003', 'Anna Smith', '789 Pine St, Villageville', 1122334455, 'Designer', 22000, 'B.Des');

INSERT INTO office_employee (Emp_no, Emp_name, Address, Phone_number, Designation, Salary, Qualification)

VALUES ('E004', 'David Brown', '101 Elm St, Citytown', 2233445566, 'Analyst', 18000, 'M.Sc');

INSERT INTO office_employee (Emp_no, Emp_name, Address, Phone_number, Designation, Salary, Qualification)

VALUES ('E005', 'Charlie Davis', '202 Maple St, Hamlet', 3344556677, 'Sales', 15000, 'BBA');

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
INSERT INTO office_employee (Emp_no, Emp_name, Address, Phone_number, Designation, Salary, Qualification)
VALUES ('E001', 'Alice Johnson', '123 Main St, Cityville', 1234567890, 'Manager', 30000, 'MBA');
INSERT INTO office_employee (Emp_no, Emp_name, Address, Phone_number, Designation, Salary, Qualification)
VALUES ('E002', 'Bob Adams', '456 Oak St, Townsville', 9876543210, 'Developer', 25000, 'B.Tech');
INSERT INTO office_employee (Emp_no, Emp_name, Address, Phone_number, Designation, Salary, Qualification)
VALUES ('E003', 'Anna Smith', '789 Pine St, Villageville', 1122334455, 'Designer', 22000, 'B.Des');
INSERT INTO office_employee (Emp_no, Emp_name, Address, Phone_number, Designation, Salary, Qualification)
VALUES ('E004', 'David Brown', '101 Elm St, Citytown', 2233445566, 'Analyst', 18000, 'M.Sc');
INSERT INTO office_employee (Emp_no, Emp_name, Address, Phone_number, Designation, Salary, Qualification)
VALUES ('E005', 'Charlie Davis', '202 Maple St, Hamlet', 3344556677, 'Sales', 15000, 'BBA');
```

Execute Save Script Clear Screen Cancel

1 row created.

4. Display the employee name whose salary is greater than 20,000.

SELECT Emp_name

FROM office_employee

WHERE Salary > 20000;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT Emp_name
FROM office_employee
WHERE Salary > 20000;
```

Execute Save Script Clear Screen Cancel

1 row created.

5. Display the employee details whose name starts with —A‖.

SELECT *

FROM office_employee

WHERE Emp_name LIKE 'A%';

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT *
FROM office_employee
WHERE Emp_name LIKE 'A%';
```

Execute Save Script Clear Screen Cancel

EMP_N	EMP_NAME	ADDRESS	PHONE_NUMBER	DESIGNATION	SALARY	QUALIFICATION
E001	Alice Johnson	123 Main St, Cityville	1234567890	Manager	30000	MBA
E003	Anna Smith	789 Pine St, Villageville	1122334455	Designer	22000	B.Des

Practical 2 :- Write a SQL Query to create a table “student”:

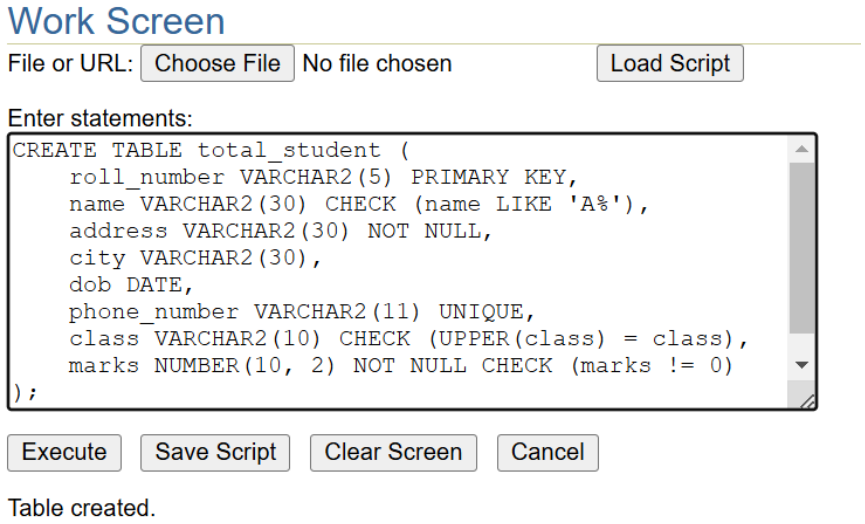
1. Display the structure of database and insert 10 records.
2. Display student information for all student in city Pune and Nagpur.
3. Display student information where marks greater than 80 and less than 90.
4. Display student name where first two character of student name _An‘.
5. Change student name to Ashish where student roll number A001.

CREATE TABLE total_student (

roll_number VARCHAR2(5) PRIMARY KEY,

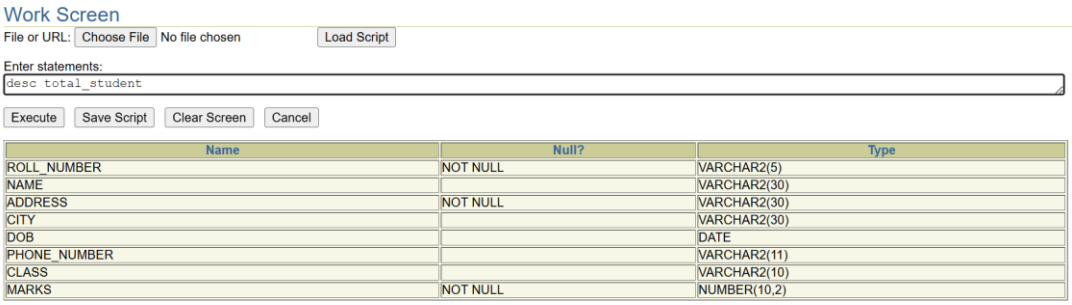
name VARCHAR2(30) CHECK (name LIKE 'A%'),

```
    address VARCHAR2(30) NOT NULL,
    city VARCHAR2(30),
    dob DATE,
    phone_number VARCHAR2(11) UNIQUE,
    class VARCHAR2(10) CHECK (UPPER(class) = class),
    marks NUMBER(10, 2) NOT NULL CHECK (marks != 0)
);
```



1. Display the structure of database and insert 10 records.

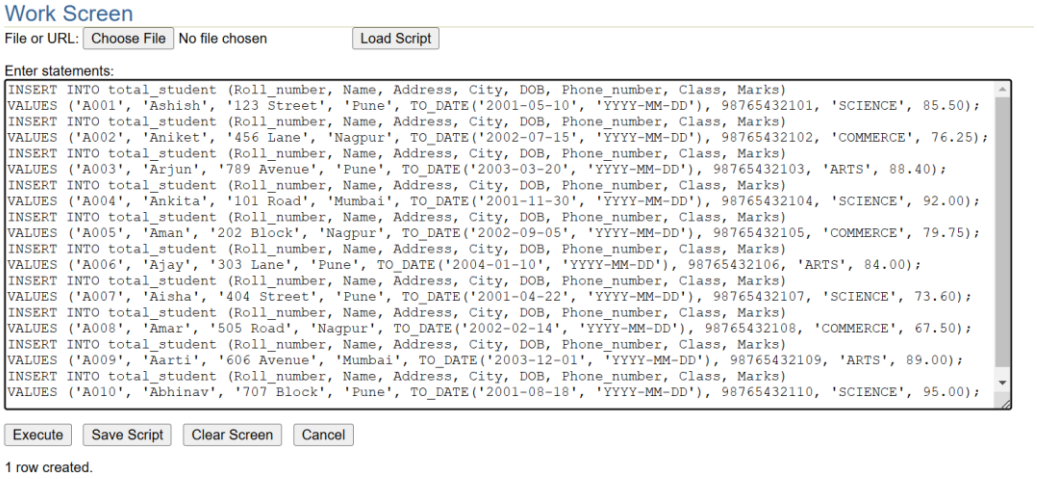
desc total_student



Name	Null?	Type
ROLL_NUMBER	NOT NULL	VARCHAR2(5)
NAME		VARCHAR2(30)
ADDRESS	NOT NULL	VARCHAR2(30)
CITY		VARCHAR2(30)
DOB		DATE
PHONE_NUMBER		VARCHAR2(11)
CLASS		VARCHAR2(10)
MARKS	NOT NULL	NUMBER(10,2)

```
INSERT INTO total_student (Roll_number, Name, Address, City, DOB, Phone_number, Class, Marks)
VALUES ('A001', 'Ashish', '123 Street', 'Pune', TO_DATE('2001-05-10', 'YYYY-MM-DD'), 98765432101,
'SCIENCE', 85.50);
INSERT INTO total_student (Roll_number, Name, Address, City, DOB, Phone_number, Class, Marks)
VALUES ('A002', 'Aniket', '456 Lane', 'Nagpur', TO_DATE('2002-07-15', 'YYYY-MM-DD'), 98765432102,
'COMMERCE', 76.25);
INSERT INTO total_student (Roll_number, Name, Address, City, DOB, Phone_number, Class, Marks)
VALUES ('A003', 'Arjun', '789 Avenue', 'Pune', TO_DATE('2003-03-20', 'YYYY-MM-DD'), 98765432103,
'ARTS', 88.40);
INSERT INTO total_student (Roll_number, Name, Address, City, DOB, Phone_number, Class, Marks)
VALUES ('A004', 'Ankita', '101 Road', 'Mumbai', TO_DATE('2001-11-30', 'YYYY-MM-DD'), 98765432104,
'SCIENCE', 92.00);
INSERT INTO total_student (Roll_number, Name, Address, City, DOB, Phone_number, Class, Marks)
VALUES ('A005', 'Aman', '202 Block', 'Nagpur', TO_DATE('2002-09-05', 'YYYY-MM-DD'), 98765432105,
'COMMERCE', 79.75);
INSERT INTO total_student (Roll_number, Name, Address, City, DOB, Phone_number, Class, Marks)
```

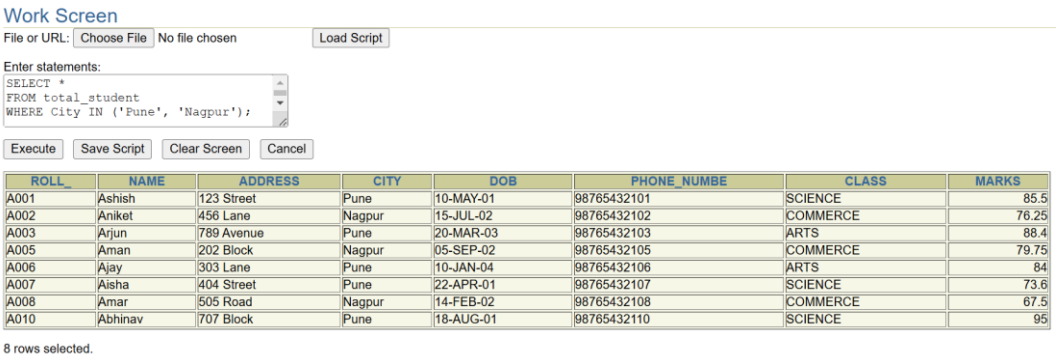
VALUES ('A006', 'Ajay', '303 Lane', 'Pune', TO_DATE('2004-01-10', 'YYYY-MM-DD'), 98765432106, 'ARTS', 84.00);
INSERT INTO total_student (Roll_number, Name, Address, City, DOB, Phone_number, Class, Marks)
VALUES ('A007', 'Aisha', '404 Street', 'Pune', TO_DATE('2001-04-22', 'YYYY-MM-DD'), 98765432107, 'SCIENCE', 73.60);
INSERT INTO total_student (Roll_number, Name, Address, City, DOB, Phone_number, Class, Marks)
VALUES ('A008', 'Amar', '505 Road', 'Nagpur', TO_DATE('2002-02-14', 'YYYY-MM-DD'), 98765432108, 'COMMERCE', 67.50);
INSERT INTO total_student (Roll_number, Name, Address, City, DOB, Phone_number, Class, Marks)
VALUES ('A009', 'Aarti', '606 Avenue', 'Mumbai', TO_DATE('2003-12-01', 'YYYY-MM-DD'), 98765432109, 'ARTS', 89.00);
INSERT INTO total_student (Roll_number, Name, Address, City, DOB, Phone_number, Class, Marks)
VALUES ('A010', 'Abhinav', '707 Block', 'Pune', TO_DATE('2001-08-18', 'YYYY-MM-DD'), 98765432110, 'SCIENCE', 95.00);



2. Display student information for all student in city Pune and Nagpur.

SELECT *FROM total_student

WHERE City IN ('Pune', 'Nagpur');



ROLL_	NAME	ADDRESS	CITY	DOB	PHONE_NUMBE	CLASS	MARKS
A001	Ashish	123 Street	Pune	10-MAY-01	98765432101	SCIENCE	85.5
A002	Aniket	456 Lane	Nagpur	15-JUL-02	98765432102	COMMERCE	76.25
A003	Arjun	789 Avenue	Pune	20-MAR-03	98765432103	ARTS	88.4
A005	Aman	202 Block	Nagpur	05-SEP-02	98765432105	COMMERCE	79.75
A006	Ajay	303 Lane	Pune	10-JAN-04	98765432106	ARTS	84
A007	Aisha	404 Street	Pune	22-APR-01	98765432107	SCIENCE	73.6
A008	Amar	505 Road	Nagpur	14-FEB-02	98765432108	COMMERCE	67.5
A010	Abhinav	707 Block	Pune	18-AUG-01	98765432110	SCIENCE	95

3. Display student information where marks greater than 80 and less than 90.

SELECT *

FROM total_student

WHERE Marks > 80 AND Marks < 90;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT *
FROM total_student
WHERE Marks > 80 AND Marks < 90;
```

Execute Save Script Clear Screen Cancel

ROLL_	NAME	ADDRESS	CITY	DOB	PHONE_NUMBE	CLASS	MARKS
A001	Ashish	123 Street	Pune	10-MAY-01	9876543210	SCIENCE	85.5
A003	Arjun	789 Avenue	Pune	20-MAR-03	9876543210	ARTS	88.4
A006	Ajay	303 Lane	Pune	10-JAN-04	9876543210	ARTS	84
A009	Aarti	606 Avenue	Mumbai	01-DEC-03	9876543210	ARTS	89

4.Display student name where first two character of student name _An‘.

SELECT Name

FROM total_student

WHERE Name LIKE 'An%';

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT Name
FROM total_student
WHERE Name LIKE 'An%';
```

Execute Save Script Clear Screen Cancel

NAME
Aniket
Ankita

5. Change student name to Ashish where student roll number A001.

UPDATE total_student

SET Name = 'Ashish'

WHERE Roll_number = 'A001';

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
UPDATE total_student
SET Name = 'Ashish'
WHERE Roll_number = 'A001';
```

Execute Save Script Clear Screen Cancel

1 row updated.

Practical 3 Write a SQL Query to create a table “sales_details”: Field Name Datatype Size

S_id varchar2 8
P_id varchar2 8
P_name varchar2 15
Price number 10
Qty number 8

1. Drop foreign key constraint on column p_no in table sales_details.
2. Add foreign key constraint on column sale_no in table sales_details.
3. Modify the column qty to include not null constraint.
4. Insert 10 records in sale_details.
5. Display p_id and total of quantity qty for each product.
6. Display p_id and total of price for all the products.

Solution:

```
CREATE TABLE sales_detailss0 (
   S_id VARCHAR2(8),
   P_id VARCHAR2(8),
   P_name VARCHAR2(15),
   Price NUMBER(10),
   Qty NUMBER(8),
   PRIMARY KEY (S_id, P_id)
);
CREATE TABLE saless0 (
   Sale_no VARCHAR2(8) PRIMARY KEY,
   Sale_date DATE
);
CREATE TABLE productss0 (
   P_id VARCHAR2(8) PRIMARY KEY,
   P_name VARCHAR2(15),
   Price NUMBER(10)
);
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
CREATE TABLE sales_details01 (
    S_id VARCHAR2(8),
    P_id VARCHAR2(8),
    P_name VARCHAR2(15),
    Price NUMBER(10),
    Qty NUMBER(8),
    PRIMARY KEY (S_id, P_id)
);


CREATE TABLE sales01 (
    Sale_no VARCHAR2(8) PRIMARY KEY,
    Sale_date DATE
);

CREATE TABLE products01 (
    P_id VARCHAR2(8) PRIMARY KEY,
    P_name VARCHAR2(15),
    Price NUMBER(10)
);
```

Execute Save Script Clear Screen Cancel

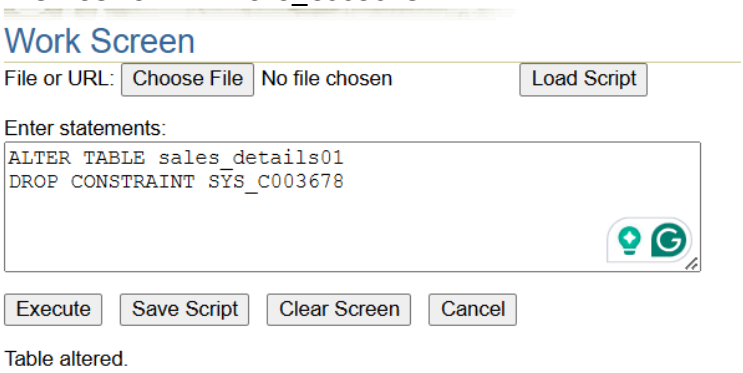
Table created.

Table created.

Table created.

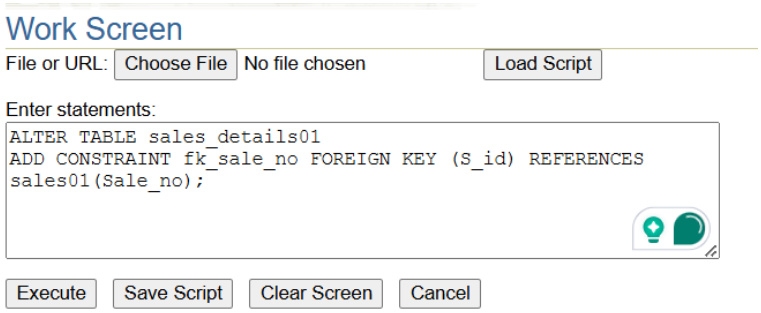
1. Drop foreign key constraint on column p_no in table sales_details.

```
ALTER TABLE sales_details01
DROP CONSTRAINT SYS_C003678
```

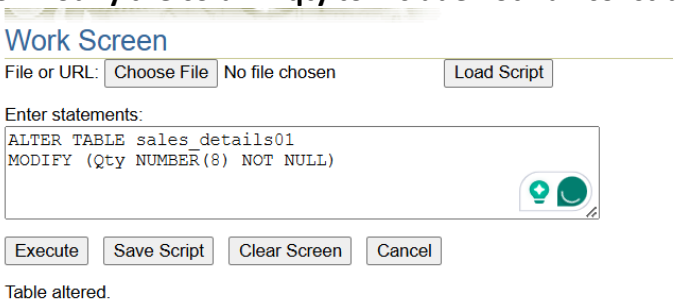


2. Add foreign key constraint on column sale_no in table sales_details.

ALTER TABLE sales_details01 ADD CONSTRAINT fk_sale_no FOREIGN KEY (S_id) REFERENCES sales01(Sale_no);



3. Modify the column qty to include not null constraint.



4. Insert 10 records in sale_details.

```
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S001', 'P001', 'Product 1', 100, 10);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S001', 'P002', 'Product 2', 150, 5);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S002', 'P001', 'Product 1', 100, 7);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S003', 'P003', 'Product 3', 200, 3);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S004', 'P004', 'Product 4', 250, 8);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S005', 'P005', 'Product 5', 300, 6);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S006', 'P002', 'Product 2', 150, 4);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S007', 'P003', 'Product 3', 200, 2);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S008', 'P004', 'Product 4', 250, 9);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S009', 'P001', 'Product 1', 100, 12);
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S001', 'P001', 'Product 1', 100, 10);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S001', 'P002', 'Product 2', 150, 5);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S002', 'P001', 'Product 1', 100, 7);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S003', 'P003', 'Product 3', 200, 3);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S004', 'P004', 'Product 4', 250, 8);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S005', 'P005', 'Product 5', 300, 6);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S006', 'P002', 'Product 2', 150, 4);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S007', 'P003', 'Product 3', 200, 2);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S008', 'P004', 'Product 4', 250, 9);
INSERT INTO sales_details01 (S_id, P_id, P_name, Price, Qty) VALUES ('S009', 'P001', 'Product 1', 100, 12);
```

Execute Save Script Clear Screen Cancel

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

5. Display p_id and total of quantity qty for each product.

SELECT P_id, SUM(Qty) AS total_qty FROM sales_details01 GROUP BY P_id;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT P_id, SUM(Qty) AS total_qty
FROM sales_details01
GROUP BY P_id;
```

Execute Save Script Clear Screen Cancel

P_ID	TOTAL_QTY
P001	29
P002	9
P003	5
P004	17
P005	6

6. Display p_id and total of price for all the products.

SELECT P_id, SUM(Price * Qty) AS total_price FROM sales_details01 GROUP BY P_id;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT P_id, SUM(Price * Qty) AS total_price
FROM sales_details01
GROUP BY P_id;
```

Execute Save Script Clear Screen Cancel

P_ID	TOTAL_PRICE
P001	2900
P002	1350
P003	1000
P004	4250
P005	1800

Practical 4 :-Write a SQL Query to create a table “customer”:

FieldName Datatype Size
Cust_no varchar2 10 Cust_name
usertype Address varchar2 10

Salary number 10

1. Modify address field with not null.
2. Add city field as it must keep city name Mumbai, Delhi and Kolkata.
3. Add salary field where salary greater than 20,000.
4. Display the structure of table customer.
5. Insert 10 records into the table customer.
6. Display all the customer details who lives in Mumbai and Kolkata.
7. Display all the customer records whose salary>20,000 and salary<30,000.
8. Modify the address field where customer number is _C001‘.

Solution:

CREATE TABLE customer_data (

cust_no VARCHAR2(10),

cust_name VARCHAR2(50),

address VARCHAR2(10),

salary NUMBER(10)

);

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
CREATE TABLE customer_data (
    cust_no VARCHAR2(10),
    cust_name VARCHAR2(50),
    address VARCHAR2(10),
    salary NUMBER(10)
);
```

Execute Save Script Clear Screen Cancel

Table created.

1 Modify address field with not null.

ALTER TABLE customer_data

MODIFY address VARCHAR2(10) NOT NULL;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
ALTER TABLE customer_data
MODIFY address VARCHAR2(10) NOT NULL;
```

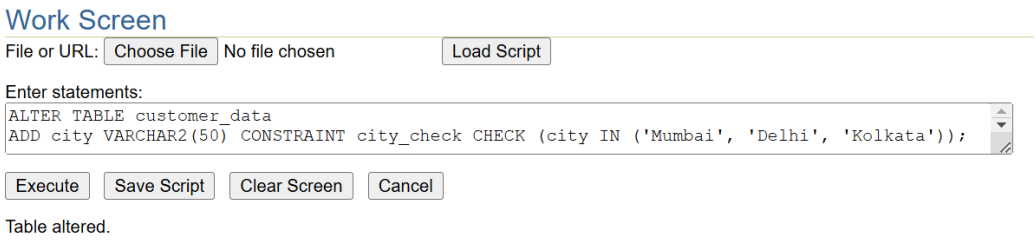
Execute Save Script Clear Screen Cancel

Table altered.

2. Add city field as it must keep city name Mumbai, Delhi and Kolkata.

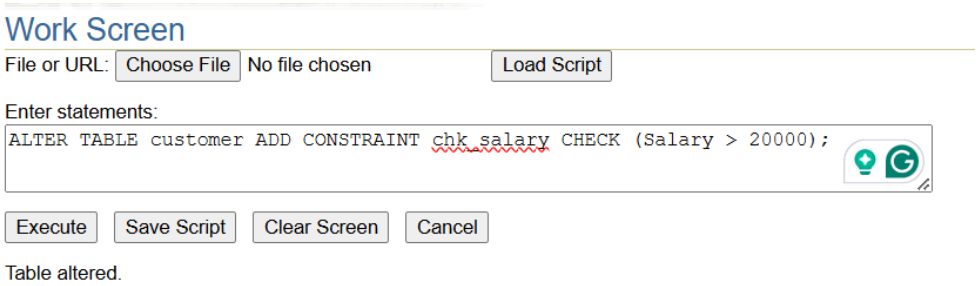
ALTER TABLE customer_data

ADD city VARCHAR2(50) CONSTRAINT city_check CHECK (city IN ('Mumbai', 'Delhi', 'Kolkata'));



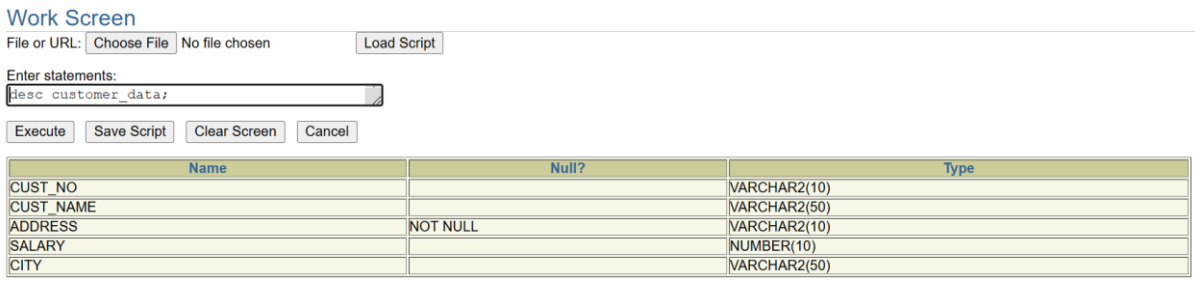
3. Add salary field where salary greater than 20,000.

ALTER TABLE customer ADD CONSTRAINT chk_salary CHECK (Salary > 20000);



4. Display the structure of table customer.

desc customer_data;



5. Add salary field where salary greater than 20,000.

INSERT INTO customer_data (cust_no, cust_name, address, salary, city) VALUES ('C001', 'John', 'Area1', 25000, 'Mumbai');

INSERT INTO customer_data (cust_no, cust_name, address, salary, city) VALUES ('C002', 'Mike', 'Area2', 30000, 'Delhi');

INSERT INTO customer_data (cust_no, cust_name, address, salary, city) VALUES ('C003', 'Sara', 'Area3', 15000, 'Kolkata');

INSERT INTO customer_data (cust_no, cust_name, address, salary, city) VALUES ('C004', 'Nina', 'Area4', 22000, 'Mumbai');

INSERT INTO customer_data (cust_no, cust_name, address, salary, city) VALUES ('C005', 'Paul', 'Area5', 18000, 'Delhi');

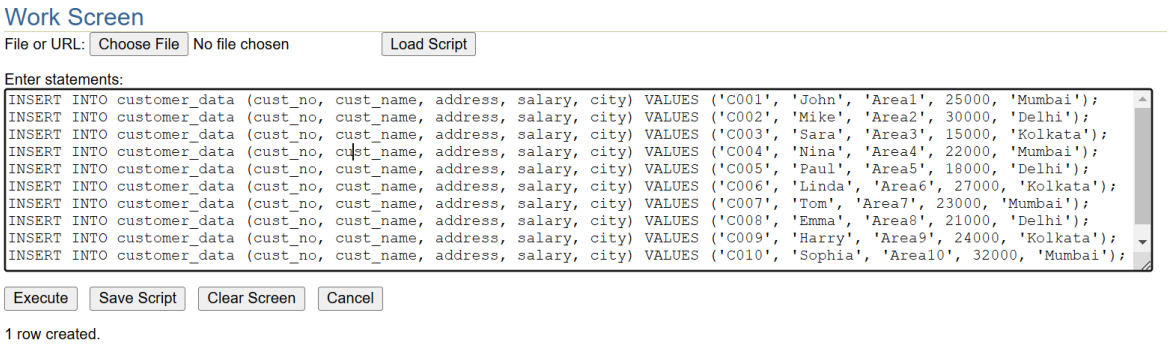
INSERT INTO customer_data (cust_no, cust_name, address, salary, city) VALUES ('C006', 'Linda', 'Area6', 27000, 'Kolkata');

INSERT INTO customer_data (cust_no, cust_name, address, salary, city) VALUES ('C007', 'Tom', 'Area7', 23000, 'Mumbai');

INSERT INTO customer_data (cust_no, cust_name, address, salary, city) VALUES ('C008', 'Emma', 'Area8', 21000, 'Delhi');

INSERT INTO customer_data (cust_no, cust_name, address, salary, city) VALUES ('C009', 'Harry', 'Area9', 24000, 'Kolkata');

INSERT INTO customer_data (cust_no, cust_name, address, salary, city) VALUES ('C010', 'Sophia', 'Area10', 32000, 'Mumbai');

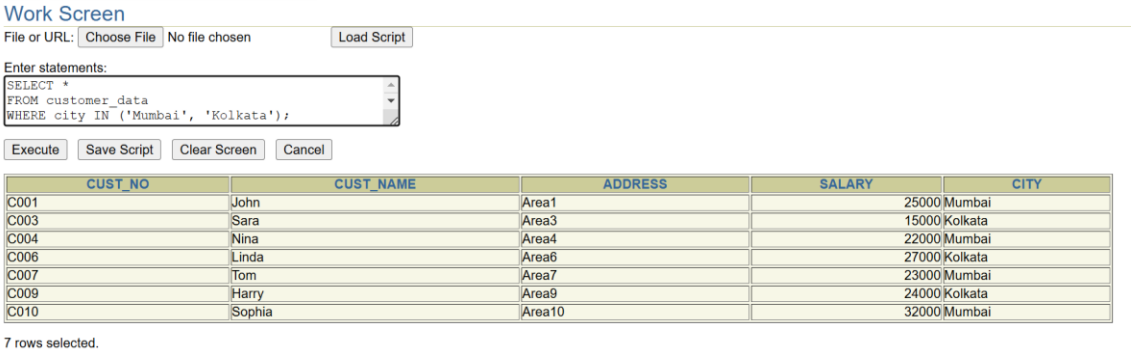


6. Display all the customer details who lives in Mumbai and Kolkata.

SELECT *

FROM customer_data

WHERE city IN ('Mumbai', 'Kolkata');

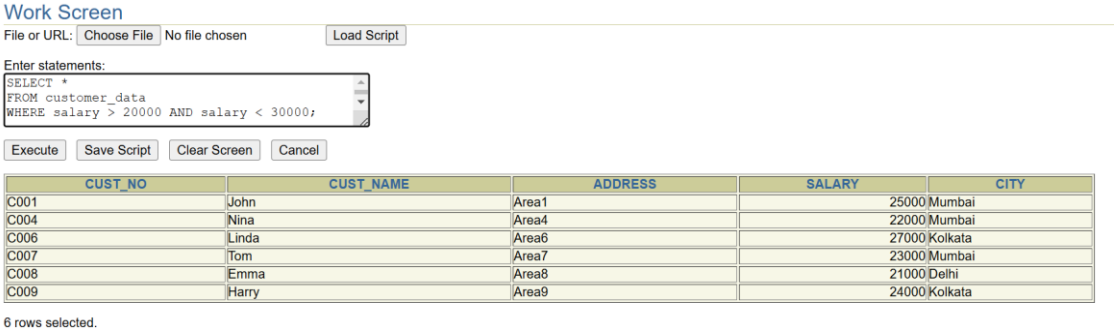


7. Display all the customer records whose salary>20,000 and salary<30,000.

SELECT *

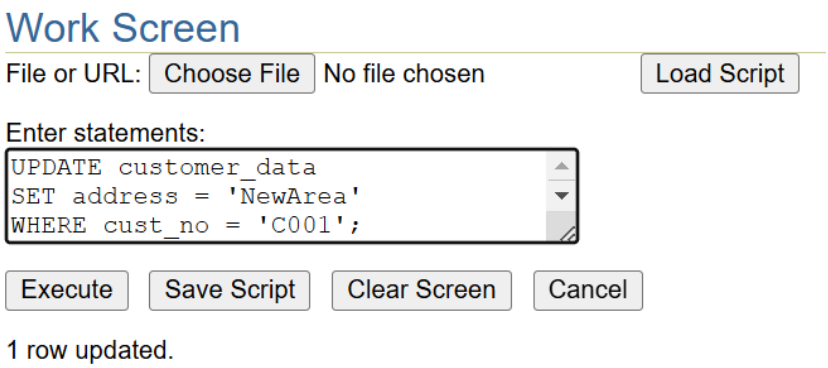
FROM customer_data

WHERE salary > 20000 AND salary < 30000;



8. Modify the address field where customer number is _C001‘.

```
UPDATE customer_data
SET address = 'NewArea'
WHERE cust_no = 'C001';
```



Practical 5 :- Write a SQL query to create c_master with fields c_no, name, address, city, state and pin_code:

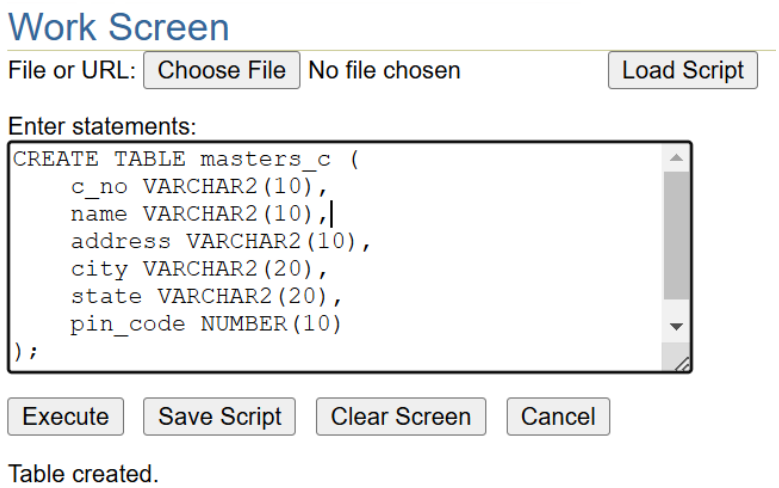
Field Name Datatype Size

C_no varchar2 10
Name varchar2 10
Address varchar2 10
State varchar2 20
City varchar2 20
Pin_code number 10

1. Create sequence which will generate number from 1..999 in ascending order, with an interval of 1 and in cyclic order.
2. Insert 10 records.
3. Create index on c_master which column name c_no and state.
4. Create view on c_master .
5. Select columns c_no, city which belongs to Nagpur and Mumbai.

Solution

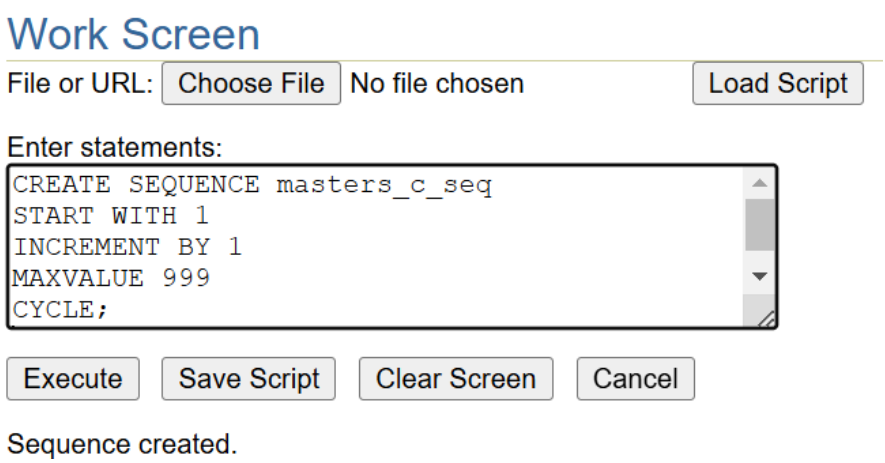
```
CREATE TABLE masters_c (

   c_no VARCHAR2(10),

   name VARCHAR2(10),

   address VARCHAR2(10),

   city VARCHAR2(20),

   state VARCHAR2(20),

   pin_code NUMBER(10)

);
```



1. **Create sequence which will generate number from 1..999 in ascending order, with an interval of 1 and in cyclic order.**

```
CREATE SEQUENCE masters_c_seq

START WITH 1

INCREMENT BY 1

MAXVALUE 999
```

CYCLE;



2. **Insert 10 records.**

INSERT INTO masters_c(c_no, name, address, city, state, pin_code)

VALUES (masters_c_seq.NEXTVAL, 'John', 'Addr1', 'Nagpur', 'Maharashtra', 440001);

INSERT INTO masters_c (c_no, name, address, city, state, pin_code)

VALUES (masters_c_seq.NEXTVAL, 'Alice', 'Addr2', 'Mumbai', 'Maharashtra', 400001);

INSERT INTO masters_c (c_no, name, address, city, state, pin_code)

VALUES (masters_c_seq.NEXTVAL, 'Sam', 'Addr3', 'Delhi', 'Delhi', 110001);

INSERT INTO masters_c (c_no, name, address, city, state, pin_code)

VALUES (masters_c_seq.NEXTVAL, 'Ravi', 'Addr4', 'Nagpur', 'Maharashtra', 440002);

INSERT INTO masters_c (c_no, name, address, city, state, pin_code)

VALUES (masters_c_seq.NEXTVAL, 'Nina', 'Addr5', 'Bangalore', 'Karnataka', 560001);

INSERT INTO masters_c (c_no, name, address, city, state, pin_code)

VALUES (masters_c_seq.NEXTVAL, 'Steve', 'Addr6', 'Hyderabad', 'Telangana', 500001);

INSERT INTO masters_c (c_no, name, address, city, state, pin_code)

VALUES (masters_c_seq.NEXTVAL, 'Priya', 'Addr7', 'Chennai', 'Tamil Nadu', 600001);

INSERT INTO masters_c (c_no, name, address, city, state, pin_code)

VALUES (masters_c_seq.NEXTVAL, 'Anil', 'Addr8', 'Mumbai', 'Maharashtra', 400002);

INSERT INTO masters_c (c_no, name, address, city, state, pin_code)

VALUES (masters_c_seq.NEXTVAL, 'Ramesh', 'Addr9', 'Kolkata', 'West Bengal', 700001);

INSERT INTO masters_c (c_no, name, address, city, state, pin_code)

VALUES (masters_c_seq.NEXTVAL, 'Sita', 'Addr10', 'Nagpur', 'Maharashtra', 440003);

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
INSERT INTO masters_c(c_no, name, address, city, state, pin_code)
VALUES (masters_c_seq.NEXTVAL, 'John', 'Addr1', 'Nagpur', 'Maharashtra', 440001);
INSERT INTO masters_c (c_no, name, address, city, state, pin_code)
VALUES (masters_c_seq.NEXTVAL, 'Alice', 'Addr2', 'Mumbai', 'Maharashtra', 400001);
INSERT INTO masters_c (c_no, name, address, city, state, pin_code)
VALUES (masters_c_seq.NEXTVAL, 'Sam', 'Addr3', 'Delhi', 'Delhi', 110001);
INSERT INTO masters_c (c_no, name, address, city, state, pin_code)
VALUES (masters_c_seq.NEXTVAL, 'Ravi', 'Addr4', 'Nagpur', 'Maharashtra', 440002);
INSERT INTO masters_c (c_no, name, address, city, state, pin_code)
VALUES (masters_c_seq.NEXTVAL, 'Nina', 'Addr5', 'Bangalore', 'Karnataka', 560001);
INSERT INTO masters_c (c_no, name, address, city, state, pin_code)
VALUES (masters_c_seq.NEXTVAL, 'Steve', 'Addr6', 'Hyderabad', 'Telangana', 500001);
INSERT INTO masters_c (c_no, name, address, city, state, pin_code)
VALUES (masters_c_seq.NEXTVAL, 'Priya', 'Addr7', 'Chennai', 'Tamil Nadu', 600001);
INSERT INTO masters_c (c_no, name, address, city, state, pin_code)
VALUES (masters_c_seq.NEXTVAL, 'Anil', 'Addr8', 'Mumbai', 'Maharashtra', 400002);
INSERT INTO masters_c (c_no, name, address, city, state, pin_code)
VALUES (masters_c_seq.NEXTVAL, 'Ramesh', 'Addr9', 'Kolkata', 'West Bengal', 700001);
INSERT INTO masters_c (c_no, name, address, city, state, pin_code)
VALUES (masters_c_seq.NEXTVAL, 'Sita', 'Addr10', 'Nagpur', 'Maharashtra', 440003);
```

Execute Save Script Clear Screen Cancel

1 row created.

3. Create index on c_master which column name c_no and state.

CREATE INDEX idx_c_master_cno_state

ON masters_c(c_no, state);

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
CREATE INDEX idx_c_master_cno_state
ON masters_c(c_no, state);
```

Execute Save Script Clear Screen Cancel

Index created.

4. Create view on c_master .

CREATE VIEW v_masters_c AS

SELECT c_no, name, address, city, state, pin_code

FROM masters_c;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
CREATE VIEW v_masters_c AS
SELECT c_no, name, address, city, state, pin_code
FROM masters_c;
```

Execute Save Script Clear Screen Cancel

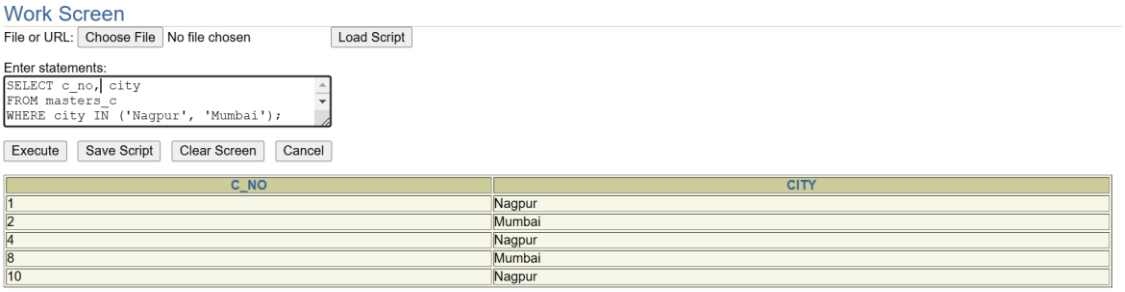
View created.

5. Select columns c_no, city which belongs to Nagpur and Mumbai.

SELECT c_no, city

FROM masters_c

WHERE city IN ('Nagpur', 'Mumbai');



C_NO	CITY
1	Nagpur
2	Mumbai
4	Nagpur
8	Mumbai
10	Nagpur

Practical 6 :- Write a SQL query to create a syntax seq_order which generating numbers from 1…9999 in ascending will number with an interval of 1 in cyclic order.

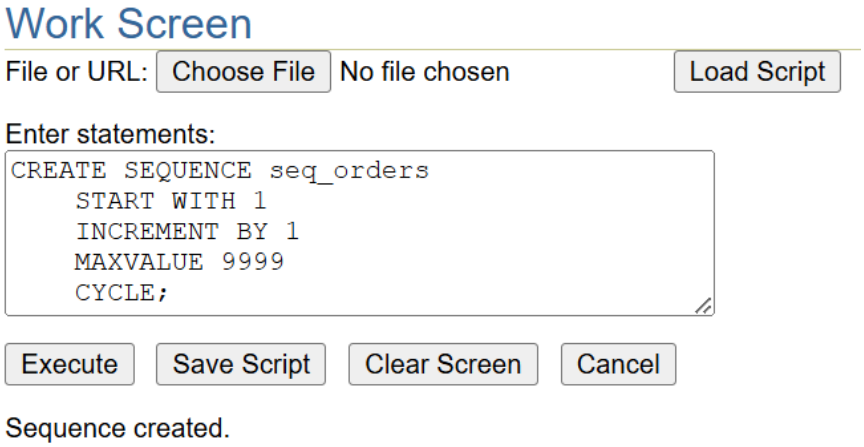
Field Name Datatype Size

P_no varchar2 10
P_name varchar2 20
Qty varchar2 10
P_rate varchar2 10

1. Display next value of sequence seq_order.
2. Display current value of sequence seq_order.
3. Insert values in sal_order table must be generated using sal_order sequence.
4. Display all records of sal_order table.
5. Change a cache memory of 50 seq_order sequence having interval 2.
6. Drop sequence.

Solution:

```
CREATE SEQUENCE seq_orders

   START WITH 1

   INCREMENT BY 1

   MAXVALUE 9999

   CYCLE;
```



1. **Display next value of sequence seq_order.**

```
CREATE TABLE sal_orders (

   P_no VARCHAR2(10),

   P_name VARCHAR2(20),

   Qty VARCHAR2(10),

   P_rate VARCHAR2(10)

);
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
CREATE TABLE sal_orders (
    P_no VARCHAR2(10),
    P_name VARCHAR2(20),
    Qty VARCHAR2(10),
    P_rate VARCHAR2(10)
);
```

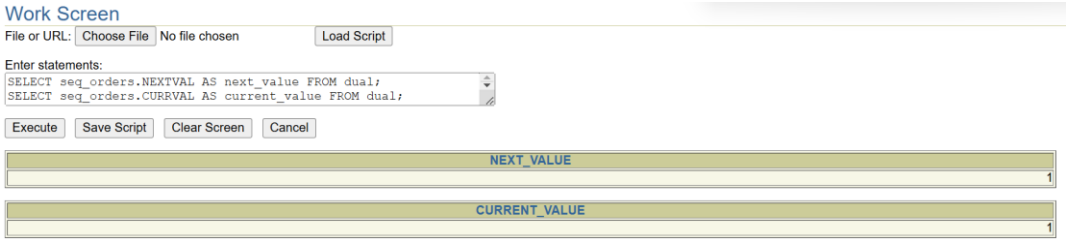
Execute Save Script Clear Screen Cancel

Table created.

2. **Display current value of sequence seq_order.**

SELECT seq_orders.NEXTVAL AS next_value FROM dual;

SELECT seq_orders.CURRVAL AS current_value FROM dual;



3. **Insert values in sal_order table must be generated using sal_order sequence.**

INSERT INTO sal_orders (P_no, P_name, Qty, P_rate)

VALUES (TO_CHAR(seq_orders.NEXTVAL), 'Product A', '50', '100');

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
INSERT INTO sal_orders (P_no, P_name, Qty, P_rate)
VALUES (TO_CHAR(seq_orders.NEXTVAL), 'Product A', '50', '100');
```

Execute Save Script Clear Screen Cancel

1 row created.

4. **Display all records of sal_order table.**

DROP SEQUENCE seq_orders;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
DROP SEQUENCE seq_orders;
```

Execute Save Script Clear Screen Cancel

Sequence dropped.

5. **Change a cache memory of 50 seq_order sequence having interval 2.**

```
CREATE SEQUENCE seq_orders
   START WITH 1
   INCREMENT BY 2
   MAXVALUE 9999
   CYCLE
   CACHE 50;
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
CREATE SEQUENCE seq_orders
    START WITH 1
    INCREMENT BY 2
    MAXVALUE 9999
    CYCLE
    CACHE 50;
```

Execute Save Script Clear Screen Cancel

Sequence created.

6. **Drop sequence.**

```
DROP SEQUENCE seq_orders;
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
DROP SEQUENCE seq_orders;
```

Execute Save Script Clear Screen Cancel

Sequence dropped.

Practical 7 :- Write a SQL query to illustrate numeric function.

1. Sqrt 2. Ceil 3. Power 4. Floor 5. Round
6. Mod 7. Abs 8. Exp 9. Greatest 10. Least

Solution

```
SELECT
    SQRT(16) AS sqrt_result,
    CEIL(5.3) AS ceil_result,
    POWER(2, 3) AS power_result,
    FLOOR(5.9) AS floor_result,
    ROUND(5.678, 2) AS round_result,
    MOD(17, 5) AS mod_result,
    ABS(-42) AS abs_result,
    EXP(2) AS exp_result,
    GREATEST(1, 10, 3, 7) AS greatest_result,
    LEAST(1, 10, 3, 7) AS least_result
FROM dual;
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT
    SQRT(16) AS sqrt_result,
    CEIL(5.3) AS ceil_result,
    POWER(2, 3) AS power_result,
    FLOOR(5.9) AS floor_result,
    ROUND(5.678, 2) AS round_result,
    MOD(17, 5) AS mod_result,
    ABS(-42) AS abs_result,
    EXP(2) AS exp_result,
    GREATEST(1, 10, 3, 7) AS greatest_result,
    LEAST(1, 10, 3, 7) AS least_result
FROM dual;
```

Execute Save Script Clear Screen Cancel

SQRT_RESULT	CEIL_RESULT	POWER_RESULT	FLOOR_RESULT	ROUND_RESULT	MOD_RESULT	ABS_RESULT	EXP_RESULT	GREATEST_RESULT	LEAST_RESULT
4	6	8	5	5.68	2	42	7.3890561	10	1

Practical 10 :- Write a SQL query for join, inner join, outer join, self join and Cartesian join.

Solution:

Table employes

CREATE TABLE employes (

employee_id NUMBER PRIMARY KEY,

first_name VARCHAR2(50),

last_name VARCHAR2(50),

department_id NUMBER

);

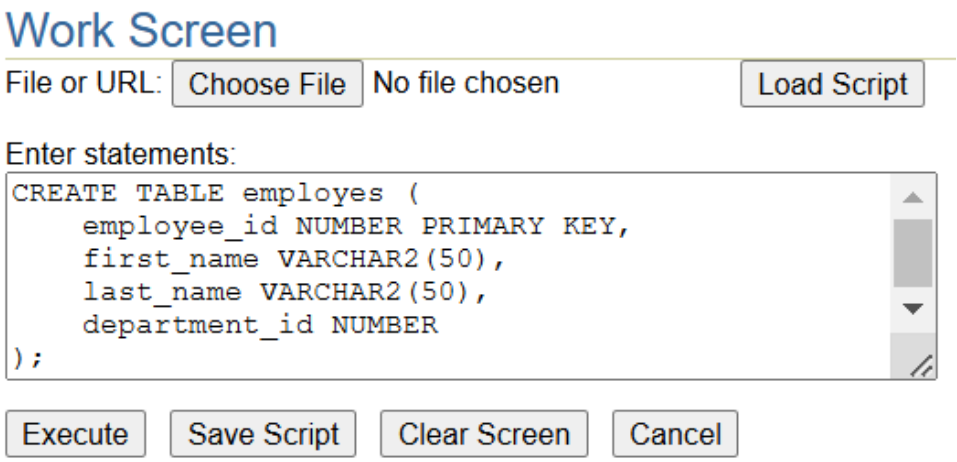


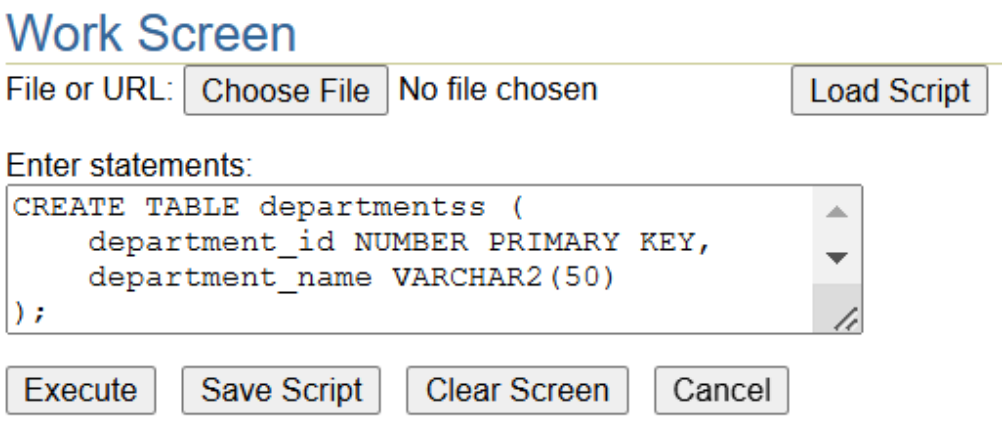
Table departmentss

CREATE TABLE departmentss (

department_id NUMBER PRIMARY KEY,

department_name VARCHAR2(50)

);



INSERT INTO employes (employee_id, first_name, last_name, department_id)

VALUES (1, 'John', 'Doe', 10);

INSERT INTO employes (employee_id, first_name, last_name, department_id)

VALUES (2, 'Jane', 'Smith', 20);

INSERT INTO employes (employee_id, first_name, last_name, department_id)

VALUES (3, 'Bob', 'Johnson', 10);

INSERT INTO employes (employee_id, first_name, last_name, department_id)

VALUES (4, 'Alice', 'Brown', NULL);

INSERT INTO departmentss (department_id, department_name)

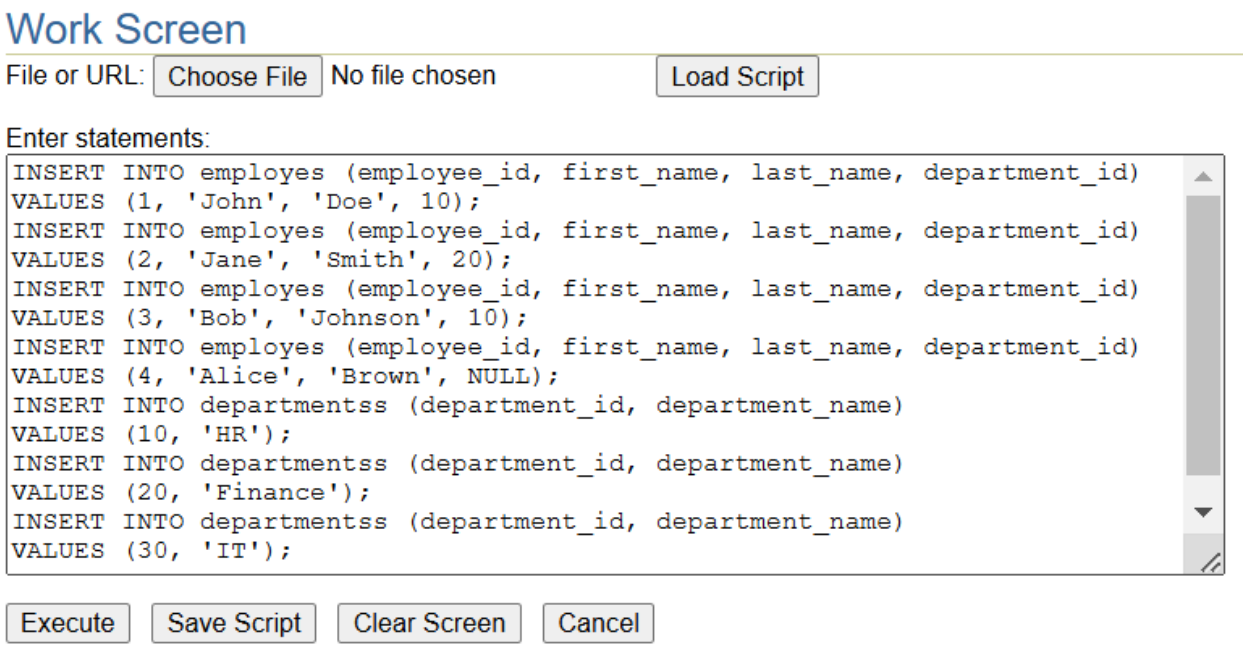
VALUES (10, 'HR');

INSERT INTO departmentss (department_id, department_name)

VALUES (20, 'Finance');

INSERT INTO departmentss (department_id, department_name)

VALUES (30, 'IT');



Join

SELECT e1.employee_id, e1.first_name, e1.last_name, e2.employee_id AS colleague_id, e2.first_name AS colleague_first_name

FROM employes e1

JOIN employes e2

ON e1.department_id = e2.department_id

AND e1.employee_id != e2.employee_id;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT e1.employee_id, e1.first_name, e1.last_name, e2.employee_id AS
colleague_id, e2.first_name AS colleague_first_name
FROM employes e1
JOIN employes e2
  ON e1.department_id = e2.department_id
  AND e1.employee_id != e2.employee_id;
```

Execute Save Script Clear Screen Cancel

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	COLLEAGUE_ID	COLLEAGUE_FIRST_NAME
3	Bob	Johnson	1	John
1	John	Doe	3	Bob

INNER JOIN

SELECT e.employee_id, e.first_name, e.last_name, d.department_name

FROM employes e

INNER JOIN departmentss d

ON e.department_id = d.department_id;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT e.employee_id, e.first_name, e.last_name, d.department_name
FROM employes e
INNER JOIN departmentss d
  ON e.department_id = d.department_id;
```

Execute Save Script Clear Screen Cancel

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
1	John	Doe	HR
2	Jane	Smith	Finance
3	Bob	Johnson	HR

LEFT JOIN

SELECT e.employee_id, e.first_name, e.last_name, d.department_name

FROM employes e

LEFT OUTER JOIN departmentss d

ON e.department_id = d.department_id;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT e.employee_id, e.first_name, e.last_name, d.department_name
FROM employes e
LEFT OUTER JOIN departmentss d
  ON e.department_id = d.department_id;
```

Execute Save Script Clear Screen Cancel

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
3	Bob	Johnson	HR
1	John	Doe	HR
2	Jane	Smith	Finance
4	Alice	Brown	

RIGHT JOIN

SELECT e.employee_id, e.first_name, e.last_name, d.department_name

FROM employes e

RIGHT OUTER JOIN departmentss d

ON e.department_id = d.department_id;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT e.employee_id, e.first_name, e.last_name, d.department_name
FROM employes e
RIGHT OUTER JOIN departmentss d
  ON e.department_id = d.department_id;
```

Execute Save Script Clear Screen Cancel

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
1	John	Doe	HR
2	Jane	Smith	Finance
3	Bob	Johnson	HR
			IT

FULL OUTER JOIN

SELECT e.employee_id, e.first_name, e.last_name, d.department_name

FROM employes e

FULL OUTER JOIN departmentss d

ON e.department_id = d.department_id;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT e.employee_id, e.first_name, e.last_name, d.department_name
FROM employes e
FULL OUTER JOIN departmentss d
  ON e.department_id = d.department_id;
```

Execute Save Script Clear Screen Cancel

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
3	Bob	Johnson	HR
1	John	Doe	HR
2	Jane	Smith	Finance
4	Alice	Brown	
			IT

CROSS JOIN

SELECT e.employee_id, e.first_name, d.department_name

FROM employes e

CROSS JOIN departmentss d;

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SELECT e.employee_id, e.first_name, d.department_name
FROM employes e
CROSS JOIN departmentss d;
```

Execute Save Script Clear Screen Cancel

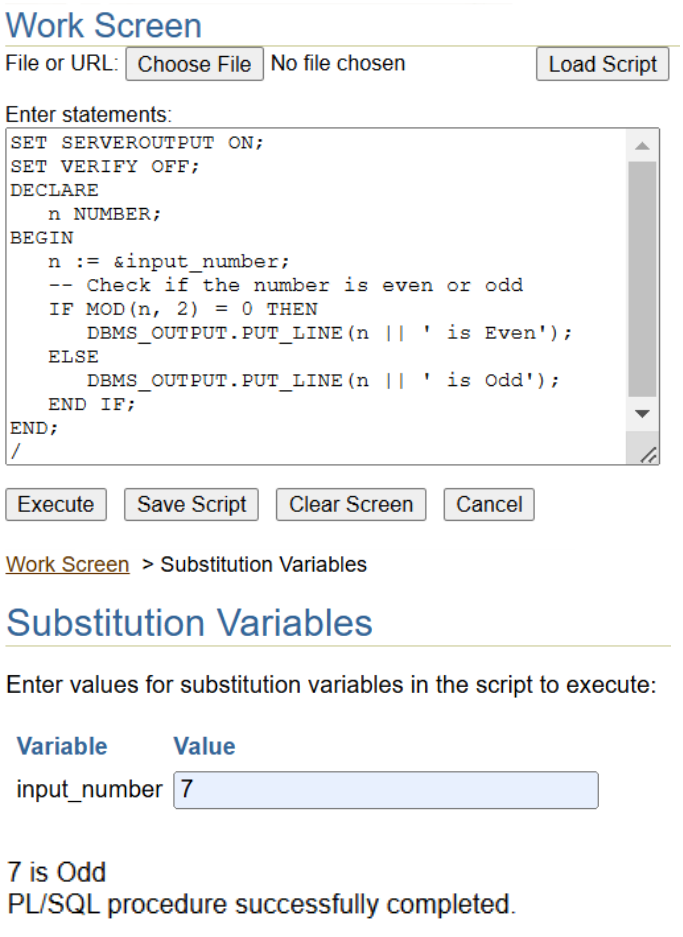
EMPLOYEE_ID	FIRST_NAME	DEPARTMENT_NAME
1	John	HR
2	Jane	HR
3	Bob	HR
4	Alice	HR
1	John	Finance
2	Jane	Finance
3	Bob	Finance
4	Alice	Finance
1	John	IT
2	Jane	IT
3	Bob	IT
4	Alice	IT

12 rows selected.

Practical 11 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to check given number is odd or even.

Solution:

```
SET SERVEROUTPUT ON;
SET VERIFY OFF;
DECLARE
  n NUMBER;
BEGIN
  n := &input_number;
  IF MOD(n, 2) = 0 THEN
    DBMS_OUTPUT.PUT_LINE(n || ' is Even');
  ELSE
    DBMS_OUTPUT.PUT_LINE(n || ' is Odd');
  END IF;
END;
/
```



Practical 12 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to check number is reverse or not.

Solution:

```
SET SERVEROUTPUT ON;
DECLARE
  n NUMBER;
  reverse_num NUMBER := 0;
  temp NUMBER;
  digit NUMBER;
BEGIN
  n := &input_number;
  temp := n;
  WHILE temp > 0 LOOP
    digit := MOD(temp, 10);
    reverse_num := reverse_num * 10 + digit;
    temp := FLOOR(temp / 10);
  END LOOP;
  IF reverse_num = n THEN
    DBMS_OUTPUT.PUT_LINE(n || ' is the same as its reverse: ' || reverse_num);
  ELSE
    DBMS_OUTPUT.PUT_LINE(n || ' is NOT the same as its reverse: ' || reverse_num);
  END IF;
END;
/
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SET SERVEROUTPUT ON;
DECLARE
   n NUMBER;
   reverse_num NUMBER := 0;
   temp NUMBER;
   digit NUMBER;
BEGIN
   n := &input_number;
   temp := n;
   WHILE temp > 0 LOOP
      digit := MOD(temp, 10);
      reverse_num := reverse_num * 10 + digit;
      temp := FLOOR(temp / 10);
   END LOOP;
   IF reverse_num = n THEN
      DBMS_OUTPUT.PUT_LINE(n || ' is the same as its reverse: ' || reverse_num);
   ELSE
      DBMS_OUTPUT.PUT_LINE(n || ' is NOT the same as its reverse: ' || reverse_num);
   END IF;
END;
/
```

Work Screen > Substitution Variables

Substitution Variables

Enter values for substitution variables in the script to execute:

Variable	Value
input_number	1551

1551 is the same as its reverse: 1551
PL/SQL procedure successfully completed.

Practical 13 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to check number is palindrome or not.

Solution:

```
SET SERVEROUTPUT ON;
DECLARE
  n NUMBER;
  reverse_num NUMBER := 0;
  temp NUMBER;
  digit NUMBER;
BEGIN
  n := &input_number;
  temp := n;
  WHILE temp > 0 LOOP
    digit := MOD(temp, 10);
    reverse_num := reverse_num * 10 + digit;
    temp := FLOOR(temp / 10);
  END LOOP;
  IF reverse_num = n THEN
    DBMS_OUTPUT.PUT_LINE(n || ' is a Palindrome.');
  ELSE
    DBMS_OUTPUT.PUT_LINE(n || ' is NOT a Palindrome.');
  END IF;
END;
/
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SET SERVEROUTPUT ON;
DECLARE
   n NUMBER;
   reverse_num NUMBER := 0;
   temp NUMBER;
   digit NUMBER;
BEGIN
   n := &input_number;
   temp := n;
   WHILE temp > 0 LOOP
      digit := MOD(temp, 10);
      reverse_num := reverse_num * 10 + digit;
      temp := FLOOR(temp / 10);
   END LOOP;
   IF reverse_num = n THEN
      DBMS_OUTPUT.PUT_LINE(n || ' is a Palindrome.');
   ELSE
      DBMS_OUTPUT.PUT_LINE(n || ' is NOT a Palindrome.');
   END IF;
END;
/
```

Execute Save Script Clear Screen Cancel

Work Screen > Substitution Variables

Substitution Variables

Enter values for substitution variables in the script to execute:

Variable	Value
input_number	664466

664466 is a Palindrome.
PL/SQL procedure successfully completed.

Practical 14 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to find the number is Armstrong or not.

Solution:

```
SET VERIFY OFF
DECLARE
   n NUMBER;
   s NUMBER := 0;
   r NUMBER;
   len NUMBER;
   m NUMBER;
BEGIN
   n := &n;
   m := n;
   len := LENGTH(TO_CHAR(n));
   WHILE n > 0 LOOP
      r := MOD(n, 10);
      s := s + POWER(r, len);
      n := TRUNC(n / 10);
   END LOOP;
   IF m = s THEN
      DBMS_OUTPUT.PUT_LINE(m || ' is an Armstrong number.');  -- Print if Armstrong
   ELSE
      DBMS_OUTPUT.PUT_LINE(m || ' is NOT an Armstrong number.');  -- Print if not Armstrong
   END IF;
END;
/
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SET VERIFY OFF
DECLARE
    n NUMBER;
    s NUMBER := 0;
    r NUMBER;
    len NUMBER;
    m NUMBER;
BEGIN
    n := &n;
    m := n;
    len := LENGTH(TO_CHAR(n));
    WHILE n > 0 LOOP
        r := MOD(n, 10);
        s := s + POWER(r, len);
        n := TRUNC(n / 10);
    END LOOP;
    IF m = s THEN
        DBMS_OUTPUT.PUT_LINE(m || ' is an Armstrong number.');  -- Print if Armstrong
    ELSE
        DBMS_OUTPUT.PUT_LINE(m || ' is NOT an Armstrong number.');  -- Print if not Armstrong
    END IF;
END;
/
```

Execute Save Script Clear Screen Cancel

Work Screen > Substitution Variables

Substitution Variables

Enter values for substitution variables in the script to execute:

Variable	Value
n	153

153 is an Armstrong number.
PL/SQL procedure successfully completed.

Practical 16 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to find the number is prime or not.

```
SET VERIFY OFF;
DECLARE
   n NUMBER;
   is_prime BOOLEAN := TRUE;
   i NUMBER;
BEGIN
   n := &n;
   IF n <= 1 THEN
      DBMS_OUTPUT.PUT_LINE(n || ' is not a prime number.');
      RETURN;
   END IF;
   FOR i IN 2 .. FLOOR(SQRT(n)) LOOP
      IF MOD(n, i) = 0 THEN
         is_prime := FALSE;
         EXIT;
      END IF;
   END LOOP;
   IF is_prime THEN
      DBMS_OUTPUT.PUT_LINE(n || ' is a prime number.');
   ELSE
      DBMS_OUTPUT.PUT_LINE(n || ' is not a prime number.');
   END IF;
END;
/
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SET VERIFY OFF;
DECLARE
    n NUMBER;
    is_prime BOOLEAN := TRUE;
    i NUMBER;
BEGIN
    n := &n;
    IF n <= 1 THEN
        DBMS_OUTPUT.PUT_LINE(n || ' is not a prime number.');
        RETURN;
    END IF;
    FOR i IN 2 .. FLOOR(SQRT(n)) LOOP
        IF MOD(n, i) = 0 THEN
            is_prime := FALSE;
            EXIT;
        END IF;
    END LOOP;
    IF is_prime THEN
        DBMS_OUTPUT.PUT_LINE(n || ' is a prime number.');
    ELSE
        DBMS_OUTPUT.PUT_LINE(n || ' is not a prime number.');
    END IF;
END;
/
```

Execute Save Script Clear Screen Cancel

Substitution Variables

Enter values for substitution variables in the script to execute:

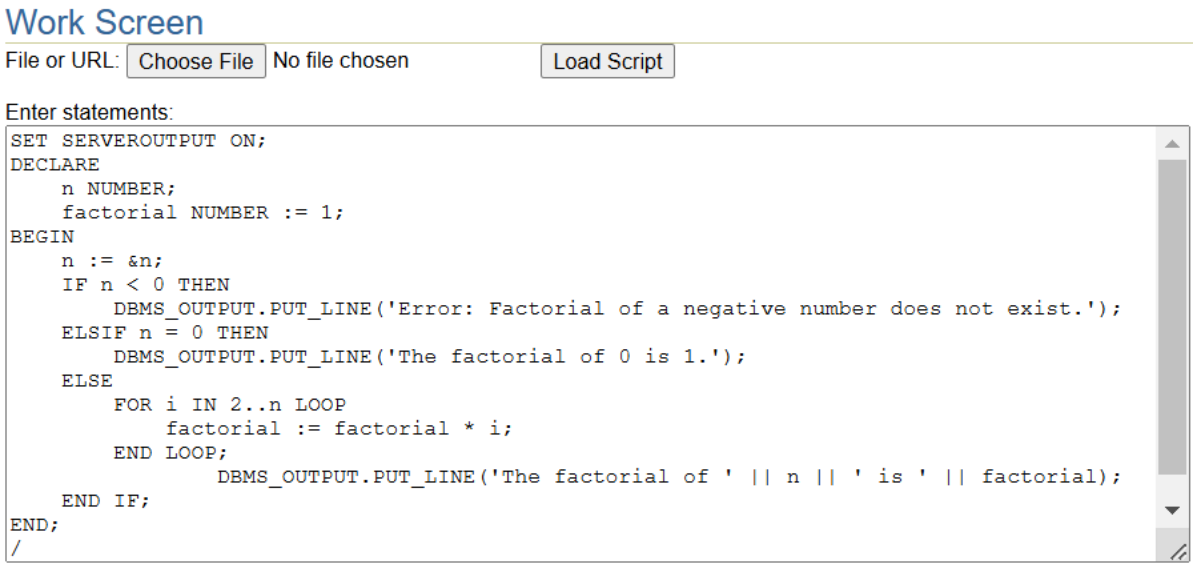
Variable	Value
n	29

29 is a prime number.
PL/SQL procedure successfully completed.

Practical 17 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to calculate factorial of a given number.

Solution:

```
SET SERVEROUTPUT ON;
DECLARE
   n NUMBER;
   factorial NUMBER := 1;
BEGIN
   n := &n;
   IF n < 0 THEN
      DBMS_OUTPUT.PUT_LINE('Error: Factorial of a negative number does not exist.');
   ELSIF n = 0 THEN
      DBMS_OUTPUT.PUT_LINE('The factorial of 0 is 1.');
   ELSE
      FOR i IN 2..n LOOP
         factorial := factorial * i;
      END LOOP;
           DBMS_OUTPUT.PUT_LINE('The factorial of ' || n || ' is ' || factorial);
   END IF;
END;
/
```



Substitution Variables

Enter values for substitution variables in the script to execute:

Variable	Value
n	5

The factorial of 5 is 120
PL/SQL procedure successfully completed.

Practical 18 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to generate Fibonacci series.

Solution:

```
SET VERIFY OFF;
DECLARE
   n NUMBER;
   a NUMBER := 0;
   b NUMBER := 1;
   next_term NUMBER;
   i NUMBER;
BEGIN
   n := &n;
   IF n <= 0 THEN
      DBMS_OUTPUT.PUT_LINE('Number of terms must be greater than 0.');
      RETURN;
   END IF;
   DBMS_OUTPUT.PUT_LINE('Fibonacci Series:');
   IF n >= 1 THEN
      DBMS_OUTPUT.PUT_LINE(a);
   END IF;
   IF n >= 2 THEN
      DBMS_OUTPUT.PUT_LINE(b);
   END IF;
   FOR i IN 3 .. n LOOP
      next_term := a + b;
      DBMS_OUTPUT.PUT_LINE(next_term);
      a := b;
      b := next_term;
   END LOOP;
END;
/
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SET VERIFY OFF;
DECLARE
    n NUMBER;
    a NUMBER := 0;
    b NUMBER := 1;
    next_term NUMBER;
    i NUMBER;
BEGIN
    n := &n;
    IF n <= 0 THEN
        DBMS_OUTPUT.PUT_LINE('Number of terms must be greater than 0.');
        RETURN;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Fibonacci Series:');
    IF n >= 1 THEN
        DBMS_OUTPUT.PUT_LINE(a);
    END IF;
    IF n >= 2 THEN
        DBMS_OUTPUT.PUT_LINE(b);
    END IF;
    FOR i IN 3 .. n LOOP
        next_term := a + b;
        DBMS_OUTPUT.PUT_LINE(next_term);
        a := b;
        b := next_term;
    END LOOP;
END;
/
```

Work Screen > Substitution Variables

Substitution Variables

Enter values for substitution variables in the script to execute:

Variable	Value
n	10

Fibonacci Series:
0
1
1
2
3
5
8
13
21
34
PL/SQL procedure successfully completed.

Practical 19 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to insert a new element in a given position in the array.

Solution:

```
SET VERIFY OFF;
DECLARE
   TYPE number_list IS TABLE OF NUMBER;
   arr number_list := number_list(10, 20, 30, 40, 50);
   new_element NUMBER;
   pos NUMBER;
   i NUMBER;
BEGIN
   new_element := &new_element;
   pos := &pos;
   IF pos < 1 OR pos > arr.COUNT + 1 THEN
      DBMS_OUTPUT.PUT_LINE('Error: Invalid position.');
      RETURN;
   END IF;
   DBMS_OUTPUT.PUT_LINE('Original Array:');
   FOR i IN 1 .. arr.COUNT LOOP
      DBMS_OUTPUT.PUT_LINE(arr(i));
   END LOOP;
   arr.EXTEND;
   FOR i IN REVERSE pos .. arr.COUNT - 1 LOOP
      arr(i + 1) := arr(i);
   END LOOP;
   arr(pos) := new_element;
   DBMS_OUTPUT.PUT_LINE('Array After Insertion:');
   FOR i IN 1 .. arr.COUNT LOOP
      DBMS_OUTPUT.PUT_LINE(arr(i));
   END LOOP;
END;
```

/

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SET VERIFY OFF;
DECLARE
    TYPE number_list IS TABLE OF NUMBER;
    arr number_list := number_list(10, 20, 30, 40, 50);
    new_element NUMBER;
    pos NUMBER;
    i NUMBER;
BEGIN
    new_element := &new_element;
    pos := &pos;
    IF pos < 1 OR pos > arr.COUNT + 1 THEN
        DBMS_OUTPUT.PUT_LINE('Error: Invalid position.');
        RETURN;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Original Array:');
    FOR i IN 1 .. arr.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(arr(i));
    END LOOP;
    arr.EXTEND;
    FOR i IN REVERSE pos .. arr.COUNT - 1 LOOP
        arr(i + 1) := arr(i);
    END LOOP;
    arr(pos) := new_element;
    DBMS_OUTPUT.PUT_LINE('Array After Insertion:');
    FOR i IN 1 .. arr.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(arr(i));
    END LOOP;
END;
/
```

Execute Save Script Clear Screen Cancel

Substitution Variables

Enter values for substitution variables in the script to execute:

Variable	Value
new_element	25
pos	3

Original Array:
10
20
30
40
50
Array After Insertion:
10
20
25
30
40
50
PL/SQL procedure successfully completed.

Practical 20: Write an algorithm, draw a flowchart and develop a PL/SQL program to delete the duplicate element from the array.

Solution:

```
SET SERVEROUTPUT ON
SET VERIFY OFF
DECLARE
   TYPE num_array IS TABLE OF NUMBER;
   original_array num_array := num_array(&no);
   unique_array num_array := num_array();
   v_element NUMBER;
   is_duplicate BOOLEAN;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Original Array:');
   FOR i IN 1 .. original_array.COUNT LOOP
      DBMS_OUTPUT.PUT_LINE(original_array(i));
   END LOOP;
   FOR i IN 1 .. original_array.COUNT LOOP
      v_element := original_array(i);
      is_duplicate := FALSE;
      FOR j IN 1 .. unique_array.COUNT LOOP
         IF unique_array(j) = v_element THEN
            is_duplicate := TRUE;
            EXIT;
         END IF;
      END LOOP;
      IF NOT is_duplicate THEN
         unique_array.EXTEND;
         unique_array(unique_array.COUNT) := v_element;
      END IF;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Array After Removing Duplicates:');
```

```
    FOR i IN 1 .. unique_array.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(unique_array(i));
    END LOOP;
END;
/
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SET SERVEROUTPUT ON
SET VERIFY OFF
DECLARE
    TYPE num_array IS TABLE OF NUMBER;
    original_array num_array := num_array(&no);
    unique_array num_array := num_array();
    v_element NUMBER;
    is_duplicate BOOLEAN;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Original Array:');
    FOR i IN 1 .. original_array.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(original_array(i));
    END LOOP;
    FOR i IN 1 .. original_array.COUNT LOOP
        v_element := original_array(i);
        is_duplicate := FALSE;
        FOR j IN 1 .. unique_array.COUNT LOOP
            IF unique_array(j) = v_element THEN
                is_duplicate := TRUE;
                EXIT;
            END IF;
        END LOOP;
        IF NOT is_duplicate THEN
            unique_array.EXTEND;
            unique_array(unique_array.COUNT) := v_element;
        END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Array After Removing Duplicates:');
    FOR i IN 1 .. unique_array.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(unique_array(i));
    END LOOP;
END;
/
```

Execute Save Script Clear Screen Cancel

```
Original Array:
101
202
101
303
404
202
Array After Removing Duplicates:
101
202
303
404
PL/SQL procedure successfully completed.
```

Practical 21 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to sort the data in ascending order

Solution:

```
SET SERVEROUTPUT ON;
DECLARE
   TYPE number_list IS TABLE OF NUMBER;
   numbers number_list := number_list(10, 25, 40, 17, 5, 42);
   temp NUMBER;
   swapped BOOLEAN := TRUE;
   i NUMBER;
BEGIN

   DBMS_OUTPUT.PUT_LINE('Original Data:');
   FOR i IN 1 .. numbers.COUNT LOOP
      DBMS_OUTPUT.PUT_LINE(numbers(i));
   END LOOP;

   WHILE swapped LOOP
      swapped := FALSE;
      FOR i IN 1 .. numbers.COUNT - 1 LOOP
         IF numbers(i) > numbers(i + 1) THEN
            -- Swap adjacent elements
            temp := numbers(i);
            numbers(i) := numbers(i + 1);
            numbers(i + 1) := temp;
            swapped := TRUE;
         END IF;
      END LOOP;
   END LOOP;

   -- Display the sorted data
   DBMS_OUTPUT.PUT_LINE('Sorted Data in Ascending Order:');
```

```
    FOR i IN 1 .. numbers.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(numbers(i));
    END LOOP;
END;
/
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
SET SERVEROUTPUT ON;
DECLARE
    TYPE number_list IS TABLE OF NUMBER;
    numbers number_list := number_list(10, 25, 40, 17, 5, 42);
    temp NUMBER;
    swapped BOOLEAN := TRUE;
    i NUMBER;
BEGIN

    DBMS_OUTPUT.PUT_LINE('Original Data:');
    FOR i IN 1 .. numbers.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(numbers(i));
    END LOOP;

    WHILE swapped LOOP
        swapped := FALSE;
        FOR i IN 1 .. numbers.COUNT - 1 LOOP
            IF numbers(i) > numbers(i + 1) THEN
                -- Swap adjacent elements
                temp := numbers(i);
                numbers(i) := numbers(i + 1);
                numbers(i + 1) := temp;
                swapped := TRUE;
            END IF;
        END LOOP;
    END LOOP;

    -- Display the sorted data
    DBMS_OUTPUT.PUT_LINE('Sorted Data in Ascending Order:');
    FOR i IN 1 .. numbers.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(numbers(i));
    END LOOP;
END;
```

```
Original Data:
10
25
40
17
5
42
Sorted Data in Ascending Order:
5
10
17
25
40
42
PL/SQL procedure successfully completed.
```

Practical 22 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to find reverse of a string.

Solution:

```
DECLARE
   original_string VARCHAR2(100);
   reversed_string VARCHAR2(100) := '';
   i INTEGER;
BEGIN
   original_string := '&Enter_String';
   FOR i IN REVERSE 1..LENGTH(original_string) LOOP
      reversed_string := reversed_string || SUBSTR(original_string, i, 1);
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Reversed String: ' || reversed_string);
END;
/
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
DECLARE
    original_string VARCHAR2(100);
    reversed_string VARCHAR2(100) := '';
    i INTEGER;
BEGIN
    original_string := '&Enter_String';
    FOR i IN REVERSE 1..LENGTH(original_string) LOOP
        reversed_string := reversed_string || SUBSTR(original_string, i, 1);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Reversed String: ' || reversed_string);
END;
/
```

Work Screen > Substitution Variables

Substitution Variables

Enter values for substitution variables in the script to execute:

Variable	Value
enter_string	Un known

Reversed String: nwonk nU
PL/SQL procedure successfully completed.

Practical 24 :- Write an algorithm, draw a flowchart and develop a PL/SQL program to find palindrome of a string.

Solution:

```
DECLARE
   input_string VARCHAR2(4000);
   char_count INTEGER := 0;
   space_count INTEGER := 0;
   word_count INTEGER := 0;
   i INTEGER;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Enter a string:');
   input_string := '&Enter_String';
   FOR i IN 1..LENGTH(input_string) LOOP
      IF SUBSTR(input_string, i, 1) = ' ' THEN
         space_count := space_count + 1;
         IF i < LENGTH(input_string) AND SUBSTR(input_string, i + 1, 1) <> ' ' THEN
            word_count := word_count + 1;
         END IF;
      ELSE
         char_count := char_count + 1;
      END IF;
   END LOOP;
   IF LENGTH(input_string) > 0 AND SUBSTR(input_string, 1, 1) <> ' ' THEN
      word_count := word_count + 1;
   END IF;
   DBMS_OUTPUT.PUT_LINE('Total Characters (excluding spaces): ' || char_count);
   DBMS_OUTPUT.PUT_LINE('Total Spaces: ' || space_count);
   DBMS_OUTPUT.PUT_LINE('Total Words: ' || word_count);
END;
/
```

Work Screen

File or URL: Choose File No file chosen Load Script

Enter statements:

```
DECLARE
    input_string VARCHAR2(4000);
    char_count INTEGER := 0;
    space_count INTEGER := 0;
    word_count INTEGER := 0;
    i INTEGER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Enter a string:');
    input_string := '&Enter_String';
    FOR i IN 1..LENGTH(input_string) LOOP
        IF SUBSTR(input_string, i, 1) = ' ' THEN
            space_count := space_count + 1;
            IF i < LENGTH(input_string) AND SUBSTR(input_string, i + 1, 1) <> ' ' THEN
                word_count := word_count + 1;
            END IF;
        ELSE
            char_count := char_count + 1;
        END IF;
    END LOOP;
    IF LENGTH(input_string) > 0 AND SUBSTR(input_string, 1, 1) <> ' ' THEN
        word_count := word_count + 1;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Total Characters (excluding spaces): ' || char_count);
    DBMS_OUTPUT.PUT_LINE('Total Spaces: ' || space_count);
    DBMS_OUTPUT.PUT_LINE('Total Words: ' || word_count);
END;
/
```

Execute Save Script Clear Screen Cancel

Substitution Variables

Enter values for substitution variables in the script to execute:

Variable	Value
enter_string	Un known

Enter a string:
Total Characters (excluding spaces): 7
Total Spaces: 1
Total Words: 2
PL/SQL procedure successfully completed.