

## Lab 10: ADMS Assignment on Range, List and ODBMS

**Aim: To understand Implementation of Data partitioning through Range and List partitioning and ODBMS**

**Range:**

1. Create a table **Customer** with following schema & partition the **cid** column by using range partitioning:

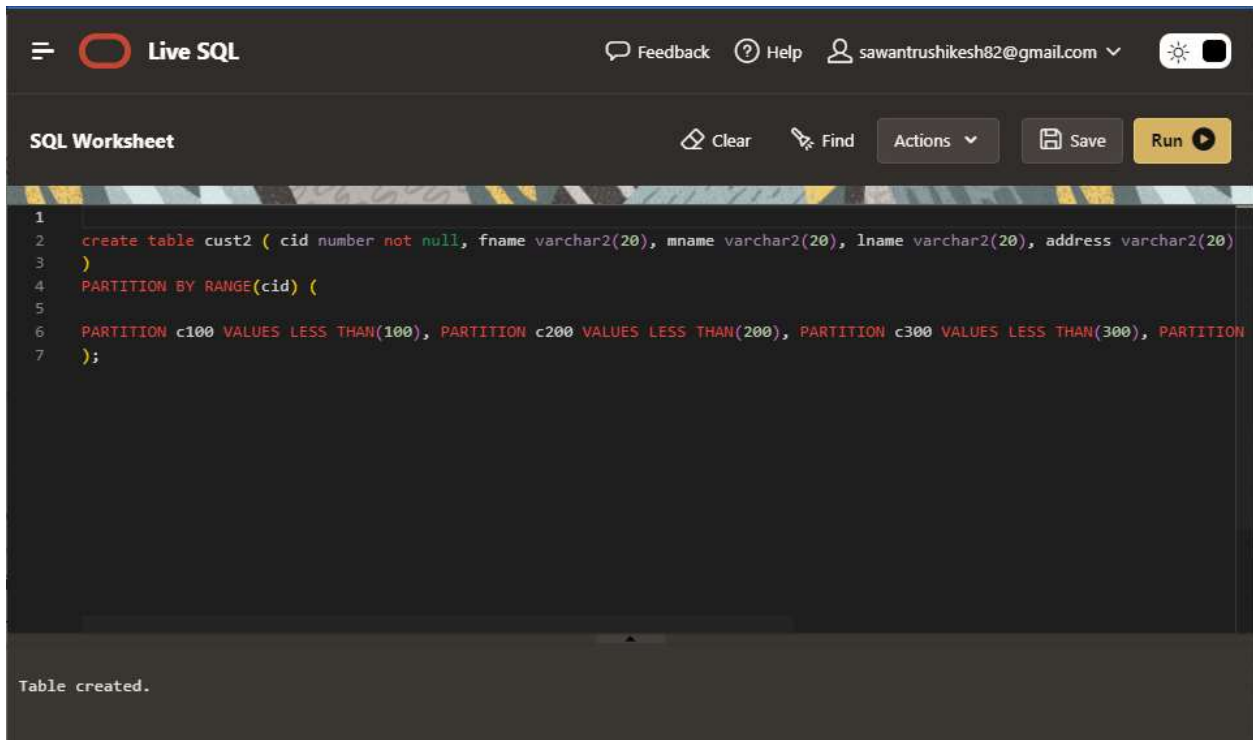
```
cid number not null
fname varchar2(20)
mname
varchar2(20) lname
varchar2(20)
address
varchar2(20)
```

- Make 3 partitions which will contain the values from 1 to 99, 100 to 199, and 200 onwards.
- Insert the appropriate records into the table (Also insert some ids with value more than 400)
- Retrieve the values from the table
- Retrieve the values from individual partition
  - Retrieve the partition details from Customer table.

query:

```
create table cust2 ( cid number not null, fname varchar2(20), mname varchar2(20), lname  
varchar2(20), address varchar2(20)  
)  
PARTITION BY RANGE(cid) (
```

```
PARTITION c100 VALUES LESS THAN(100), PARTITION c200 VALUES LESS THAN(200),  
PARTITION c300 VALUES LESS THAN(300), PARTITION c400 VALUES LESS THAN(400),  
PARTITION cother VALUES LESS THAN (MAXVALUE)  
);
```



The screenshot shows the 'Live SQL' web application interface. At the top, there is a navigation bar with a menu icon, the 'Live SQL' logo, and links for 'Feedback', 'Help', and a user profile 'sawantushikesh82@gmail.com'. Below this is a toolbar with 'Clear', 'Find', 'Actions', 'Save', and a 'Run' button. The main area is a 'SQL Worksheet' with a dark background and a light blue horizontal bar. The SQL query from the previous blocks is pasted into the editor, with line numbers 1 through 7 on the left. The query is:   
1 create table cust2 ( cid number not null, fname varchar2(20), mname varchar2(20), lname varchar2(20), address varchar2(20)  
2 )  
3 PARTITION BY RANGE(cid) (  
4 PARTITION c100 VALUES LESS THAN(100), PARTITION c200 VALUES LESS THAN(200), PARTITION c300 VALUES LESS THAN(300), PARTITION  
5 );  
6  
7  
At the bottom of the interface, a status message reads 'Table created.'

query:

insert into cust2 values (93, 'Rushikesh', 'Nitn', 'Sawant','Mumbai');

insert into cust2 values (90, 'Rutvik', 'R', 'Redkar', 'Mumbai');

insert into cust2 values (104, 'Ashvin', 'A', 'Shetty', 'Mumbai');

```
insert into cust2 values(93,'Rushikesh','Nitn','Sawant','Mumbai');  
insert into cust2 values(90,'Rutvik','R','Redkar','Mumbai');  
insert into cust2 values(104,'Ashvin','A','Shetty','Mumbai');
```

query:

select \* from cust2;

```
1 select * from cust2;  
2
```

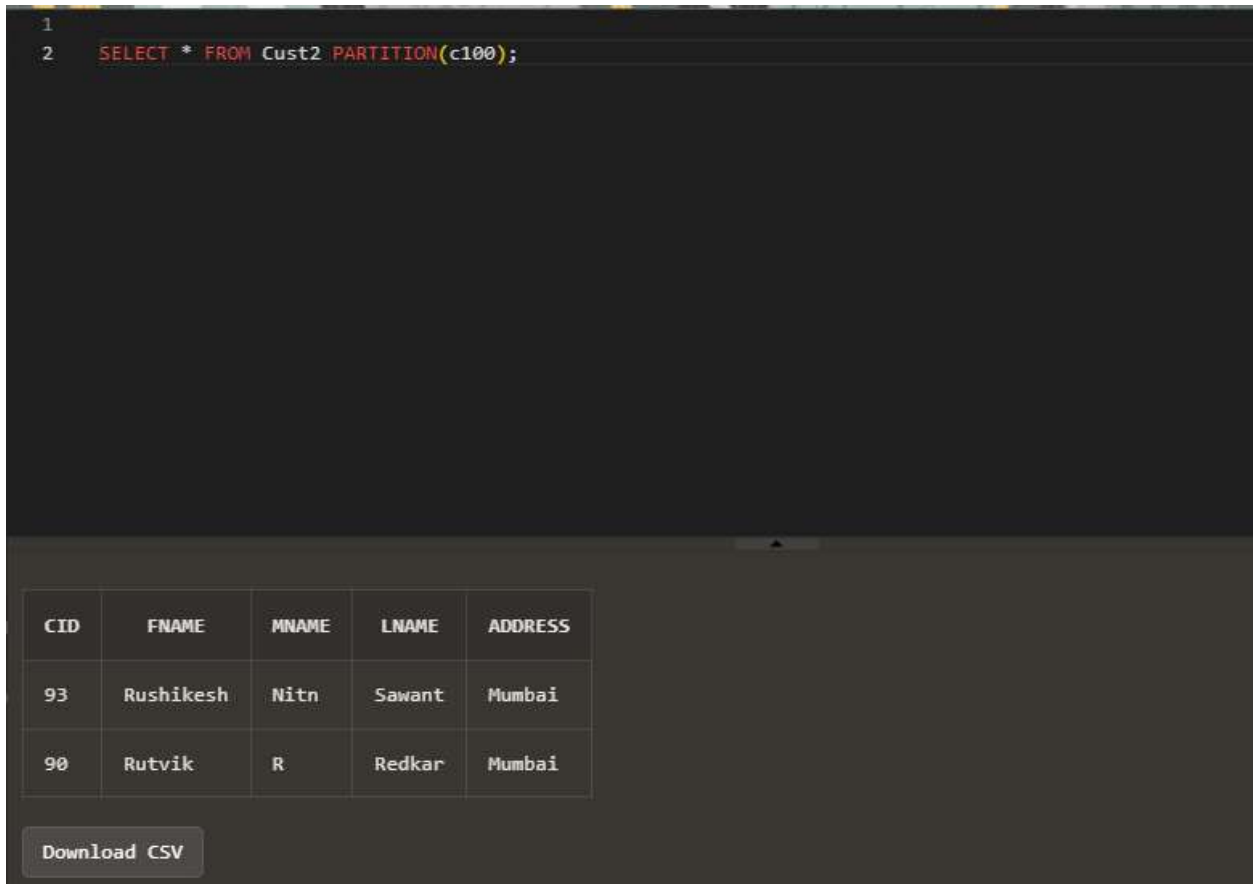
CID	FNAME	MNAME	LNAME	ADDRESS
93	Rushikesh	Nitn	Sawant	Mumbai
90	Rutvik	R	Redkar	Mumbai
104	Ashvin	A	Shetty	Mumbai

Download CSV

3 rows selected.

query:

```
SELECT * FROM Cust2 PARTITION(c100);
```



The screenshot shows a SQL query execution interface. At the top, a query is entered: `SELECT * FROM Cust2 PARTITION(c100);`. Below the query, the results are displayed in a table with 5 columns: CID, FNAME, MNAME, LNAME, and ADDRESS. The table contains two rows of data. Below the table, there is a button labeled "Download CSV".

CID	FNAME	MNAME	LNAME	ADDRESS
93	Rushikesh	Nitn	Sawant	Mumbai
90	Rutvik	R	Redkar	Mumbai

Download CSV

query:

```
SELECT * FROM Cust2 PARTITION(c200);
```



The screenshot shows a SQL query execution interface. At the top, a query is entered: `SELECT * FROM Cust2 PARTITION(c200);`. Below the query, the results are displayed in a table with 5 columns: CID, FNAME, MNAME, LNAME, and ADDRESS. The table contains one row of data. Below the table, there is a button labeled "Download CSV".

CID	FNAME	MNAME	LNAME	ADDRESS
104	Ashvin	A	Shetty	Mumbai

Download CSV

query:

SELECT \* FROM Cust2 PARTITION(c300);

```
1
2 SELECT * FROM Cust2 PARTITION(c300);
```

CID	FNAME	MNAME	LNAME	ADDRESS
269	Abhijeet	B	sharma	Thane

Download CSV

query:

SELECT \* FROM Cust2 PARTITION(c400);

```
1
2 SELECT * FROM Cust2 PARTITION(c400);
```

no data found

SELECT \* FROM Cust2 PARTITION(cother);

```
1
2 SELECT * FROM Cust2 PARTITION(cother);
3
```

CID	FNAME	MNAME	LNAME	ADDRESS
450	Abhijeet	A	sharma	Thane

Download CSV

SELECT TABLE\_NAME,PARTITION\_NAME FROM USER\_TAB\_PARTITIONS

```
1
2 SELECT TABLE_NAME,PARTITION_NAME FROM USER_TAB_PARTITIONS
3
```

TABLE_NAME	PARTITION_NAME
CUST2	C100
CUST2	C200
CUST2	C300
CUST2	C400
CUST2	COTHER

Download CSV

2. Create a table with following schema

Table name: Purchase

transid number not null

cust\_id number

inv\_date date

cust\_name

varchar2(30)

- Partition the table according to inv\_date such that it has 4 partitions having: Data of 2008 & previous years,

Data of 2009

Data of 2010

Data of 2011 & onwards.

- Insert the appropriate records into the table
- Retrieve the values from the table
- Split the last partition so that we have a separate partition for 2011; check the entries of system table.
- Retrieve the values from individual partitions
- Find the number of transactions done in the year 2009

query:

```
create table Purchase ( transid number not null, cust_id number, inv_date date , cust_name  
varchar2(30)
```

```
)
```

```
PARTITION BY RANGE(inv_date) (PARTITION sales2008 VALUES LESS
```

```
THAN(TO_DATE('31/12/2008','DD/MM/YYYY')), PARTITION sales2009 VALUES LESS
```

```
THAN(TO_DATE('31/12/2009','DD/MM/YYYY')), PARTITION sales2010 VALUES LESS
```

```
THAN(TO_DATE('31/12/2010','DD/MM/YYYY')), PARTITION sales2011 VALUES LESS THAN
```

```
(Maxvalue)
```

```
);
```

```
1  
2 create table Purchase ( transid number not null, cust_id number, inv_date date , cust_name varchar2(30)  
3 )  
4 PARTITION BY RANGE(inv_date) (PARTITION sales2008 VALUES LESS  
5 THAN(TO_DATE('31/12/2008','DD/MM/YYYY')), PARTITION sales2009 VALUES LESS THAN(TO_DATE('31/12/2009','DD/MM/YYYY')), PARTITION  
6 );  
7  
8
```

Table created.



query:

insert into Purchase values(1,1,TO\_DATE('23/02/2007','DD/MM/YYYY'),'Rushikesh');

insert into Purchase values(2,2,TO\_DATE('23/12/2008','DD/MM/YYYY'),'Ashvin');

insert into Purchase values(3,3,TO\_DATE('13/01/2009','DD/MM/YYYY'),'Rutvik');

insert into Purchase values(4,4,TO\_DATE('13/01/2010','DD/MM/YYYY'),'Abijeet');

insert into Purchase values(5,5,TO\_DATE('13/11/2011','DD/MM/YYYY'),'Muskan');

```
1
2  insert into Purchase values(1,1,TO_DATE('23/02/2007','DD/MM/YYYY'),'Rushikesh');
3  insert into Purchase values(2,2,TO_DATE('23/12/2008','DD/MM/YYYY'),'Ashvin');
4  insert into Purchase values(3,3,TO_DATE('13/01/2009','DD/MM/YYYY'),'Rutvik');
5  insert into Purchase values(4,4,TO_DATE('13/01/2010','DD/MM/YYYY'),'Abijeet');
6  insert into Purchase values(5,5,TO_DATE('13/11/2011','DD/MM/YYYY'),'Muskan');
7
8
```

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

query:

```
SELECT * FROM Purchase PARTITION(sales2008);
```

```
1
2  SELECT * FROM Purchase PARTITION(sales2008);
3
4
```

TRANSID	CUST_ID	INV_DATE	CUST_NAME
1	1	23-FEB-07	Rushikesh
2	2	23-DEC-08	Ashvin

[Download CSV](#)

query:

SELECT \* FROM Purchase PARTITION(sales2010);

```
1
2  SELECT * FROM Purchase PARTITION(sales2010);
3
4
```

TRANSID	CUST_ID	INV_DATE	CUST_NAME
4	4	13-JAN-10	Abijeet

query:

```
SELECT * FROM Purchase PARTITION(sales2011);
```



The screenshot shows a SQL query execution interface. The query is displayed in a text area with line numbers 1 through 4. The results are shown in a table below the query.

TRANSID	CUST_ID	INV_DATE	CUST_NAME
5	5	13-NOV-11	Muskan

query:

```
SELECT count(*) as Total_Transaction_2009 FROM Purchase PARTITION(sales2009);
```

```
1  
2  SELECT count(*) as Total_Transaction_2009 FROM Purchase PARTITION(sales2009);  
3
```

TOTAL_TRANSACTION_2009
------------------------

1
---

**List:**

1. Create a table Bookshelf with following schema & partition the Category column by using list partitioning:

**Table name: Bookshelf**

Title varchar2(60) not null Publisher  
varchar2(40) not null Category  
varchar2(30)

Rating number not null;

- Divide the data into 4 partitions using list partitioning on column Category with values 'TECHNOLOGY','QUANTITATIVE', 'LOGICAL', 'MYTHOLOGY'.
- Insert the appropriate records into the table
- retrieve the values from the table and from individual partitions  
Retrieve the partition details from system table

query:

```
CREATE TABLE Bookshelf (Title varchar2(60) Not Null,  
Publisher VARCHAR2(40) Not Null, Category VARCHAR2(30),  
Rating NUMBER(10) Not Null  
)  
PARTITION BY LIST(Category) (  
PARTITION cat_tech VALUES('Technology'), PARTITION cat_quant VALUES ('Quantitative'),  
PARTITION cat_log VALUES('Logical'), PARTITION cat_myth VALUES('Mythology'),  
PARTITION cat_other VALUES(Default)  
);
```

```
1  
2 CREATE TABLE Bookshelf (Title varchar2(60) Not Null,  
3 Publisher VARCHAR2(40) Not Null, Category VARCHAR2(30),  
4 Rating NUMBER(10) Not Null  
5 )  
6 PARTITION BY LIST(Category) (  
7 PARTITION cat_tech VALUES('Technology'), PARTITION cat_quant VALUES ('Quantitative'), PARTITION cat_log VALUES('Logical'),  
8 ) ;  
9  
10
```

Table created.

```
1 insert into Bookshelf values('NLP','Def','Technology',7);  
2 insert into Bookshelf values('Seeta','lmn','Mythology',9);  
3 insert into Bookshelf values('Stats','hgf','Logical',4);  
4 insert into Bookshelf values('jkl','pqr','Quantitative',8);  
5 insert into Bookshelf values('opy','rts','Quantitative',3);  
6 insert into Bookshelf values('srt','pol','other',3);  
7  
8
```

1 row(s) inserted.  
1 row(s) inserted.  
1 row(s) inserted.  
1 row(s) inserted.  
1 row(s) inserted.  
1 row(s) inserted.

```
SELECT * FROM Bookshelf PARTITION(cat_tech);
```



The screenshot shows a SQL query execution interface. The query is `SELECT * FROM Bookshelf PARTITION(cat_tech);`. The result is displayed in a table with the following data:

TITLE	PUBLISHER	CATEGORY	RATING
NLP	Def	Technology	7

```
SELECT * FROM Bookshelf PARTITION(cat_log);
```



The screenshot shows a SQL query execution interface. The query is `SELECT * FROM Bookshelf PARTITION(cat_log);`. The result is displayed in a table with the following data:

TITLE	PUBLISHER	CATEGORY	RATING
Stats	hgf	Logical	4



```
SELECT * FROM Bookshelf PARTITION(cat_quant);
```

```
1  SELECT * FROM Bookshelf PARTITION(cat_quant);
2
3
```

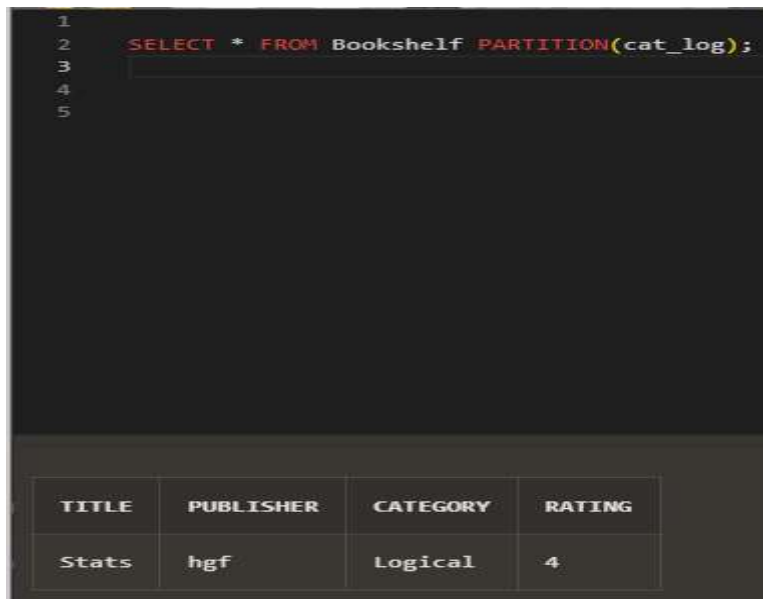
TITLE	PUBLISHER	CATEGORY	RATING
jkl	pqr	Quantitative	8
opq	rst	Quantitative	3

```
SELECT * FROM Bookshelf PARTITION(cat_tech);
```

```
1  SELECT * FROM Bookshelf PARTITION(cat_tech);
2
3
```

TITLE	PUBLISHER	CATEGORY	RATING
NLP	Def	Technology	7


SELECT \* FROM Bookshelf PARTITION(cat\_log);



The screenshot shows a SQL query execution interface. The query is `SELECT * FROM Bookshelf PARTITION(cat_log);`. The result set is displayed in a table with four columns: TITLE, PUBLISHER, CATEGORY, and RATING. The data row shows 'Stats' as the title, 'hgf' as the publisher, 'Logical' as the category, and '4' as the rating.

TITLE	PUBLISHER	CATEGORY	RATING
Stats	hgf	Logical	4

SELECT \* FROM Bookshelf PARTITION(cat\_quant);



The screenshot shows a SQL query execution interface. The query is `SELECT * FROM Bookshelf PARTITION(cat_quant);`. The result set is displayed in a table with four columns: TITLE, PUBLISHER, CATEGORY, and RATING. The data rows show 'jkl' and 'opy' as titles, 'pqr' and 'rts' as publishers, 'Quantitative' as the category, and '8' and '3' as ratings.

TITLE	PUBLISHER	CATEGORY	RATING
jkl	pqr	Quantitative	8
opy	rts	Quantitative	3

SELECT \* FROM Bookshelf PARTITION(cat\_myth);

```
1
2  SELECT * FROM Bookshelf PARTITION(cat_myth);
3
4
5
6
```

TITLE	PUBLISHER	CATEGORY	RATING
Seeta	lmn	Mythology	9

SELECT \* FROM Bookshelf PARTITION(cat\_other);

```
1
2  SELECT * FROM Bookshelf PARTITION(cat_other);
3
4
5
6
```

TITLE	PUBLISHER	CATEGORY	RATING
srt	pol	other	3

SELECT TABLE\_NAME,PARTITION\_NAME FROM USER\_TAB\_PARTITIONS

```
1
2  SELECT TABLE_NAME,PARTITION_NAME FROM USER_TAB_PARTITIONS
3
4
5
6
```

TABLE_NAME	PARTITION_NAME
BOOKSHELF	CAT_LOG
BOOKSHELF	CAT_MYTH
BOOKSHELF	CAT_OTHER
BOOKSHELF	CAT_QUANT
BOOKSHELF	CAT_TECH
CUST2	C100
CUST2	C200

## Implementation of ORDBMS using ADT (Abstract Data Types)

1. Create type **Address** having the specified columns (address1, address2, state, city, pincode).  
Create **Customer** table having the specified columns (Customer\_id, Customer\_name and Address type).

- Insert records into customer table.
- Display the details customer.
- display the description/structure of the customer table
- List the customers from Mumbai
- Count the number of customers' state wise

create type type\_address As object:

```
(  
address1 varchar(30), address2 varchar(30), state varchar(20), city varchar(20), pincode  
number(10)  
);
```

```
1  create type type_address As object  
2  
3  (  
4  address1 varchar(30), address2 varchar(30), state varchar(20), city varchar(20),  
5  );  
6  
7  
8  
9  
10
```

type created.

```
1 create table Customer (  
2 Customer_id number(5) primary key, Customer_name varchar2(30), Address type_address  
3 );  
4  
5  
6  
7  
8  
9
```

Table created.

```
1  select * from Customer;  
2  
3  
4  
5  
6  
7
```

CUSTOMER_ID	CUSTOMER_NAME	ADDRESS
3	Rutvik	[unsupported data type]
4	Muskan	[unsupported data type]
1	Rushikesh	[unsupported data type]
2	Ashwin	[unsupported data type]

```
1 desc Customer;
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

TABLE CUSTOMER

Column	Null?	Type
CUSTOMER_ID	NOT NULL	NUMBER(5,0)
CUSTOMER_NAME	-	VARCHAR2(30)
ADDRESS	-	TYPE_ADDRESS



```
1  Select * from Customer c where c.address.city = 'Mumbai';
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

CUSTOMER_ID	CUSTOMER_NAME	ADDRESS
1	Rushikesh	[unsupported data type]
2	Ashwin	[unsupported data type]

```
1  Select c.address.state, count(*) as No_of_customer from Customer c Group by c.address.state;  
2  
3  
4  
5  
6  
7
```

ADDRESS.STATE	NO_OF_CUSTOMER
Maharashtra	2
Gujarat	1
Madhya Pradesh	1

Download CSV

**Conclusion: Successfully implemented Data partitioning through Range and List partitioning and ODBMS**