A. Hash Partition

Q1. Create table Book details with the attribute b_id, title, author, price. Partition this table into 4 partitions

using hash partitioning method.

1. Display the contents of the table.

2. Display the contents of each partition

3. Rename the partition p1 to part1.

4. Display the partition names of table book_details.

create table e4(b_id number,title varchar(20),author varchar(20),price number(20)) partition by hash(b_id) partitions 4;

insert into e4 values(1, 'Chandalika','Rabindra Tagore', 145);

insert into e4 values(1, 'Time Machine','H.G Wells', 745);

insert into e4 values(3, 'Mein Kamph',' Adolf Hitler', 745);

insert into e4 values(4, ' MCA','BVIMIT', 111);

select * from e4;

select table_name, partition_name from user_tab_partitions where table_name='E4';

alter table e4 rename partition p4 to book4;

select table_name, partition_name from user_tab_partitions where table_name='E4';

Q2. Create a table student_details with the attributes Roll_no, names, marks using hash partitioning with 3

partitions.

1. Display the content of the partitions.

2. Delete one partition.

3. Display the name of existing partitions.

create table student_details(Roll_no number, name varchar(25),marks number) partition by hash(Roll_no) partitions 3;

insert into student_details values(31, 'Abhishek', 69);

insert into student_details values(18, 'Darshan', 88);

insert into student_details values(19, 'Suraj', 96);

select * from student_detail;

select table_name, partition_name from user_tab_partitions where table_name='STUDENT_DETAIL';

2)

A. Range Partition

Q1. Create table student with attributes stud_id, name, marks with range partitioning and the

partitioning attribute is marks.

1. Display contents of the table.

2. Display the details of the students who failed.

3. Display the details of the students of "second class".

4. Display the details of the students of "First class".

5. Display the name of partitions.

6. Display the details of students who passed with distinctions.

7. Display the number of students who failed.

8. Display the details of the student who scored highest marks

9. Split the partition fail to f1 with marks less than 42 and f2 to marks less than 55.

10. Merge f1, f2 into a new partition pp1;

11. Drop the partition dist_class.

12. Add a partition p_new for storing the marks less than 100.

create table stud_31(stud_id number,stud_name varchar(25),stud_marks number)

partition by range(stud_marks)(partition fail values less than(40),

partition second_class values less than(60),

partition first_class values less than (75),

```
partition dist_class values less than(100));


insert into stud_31 values(1,'Abhishek',78);

insert into stud_31 values(2,'Tanay',30);

insert into stud_31 values(3,'Omkar',89);

insert into stud_31 values(4,'Rahul',70);

insert into stud_31 values(5,'Yogesh',30);

insert into stud_31 values(6,'Akash',55);

insert into stud_31 values(7,'Parvin',25);

insert into stud_31 values(8,'Raj',23);


select * from stud_31;

select * from stud_31 partition (fail);

select * from stud_31 partition (second_class);

select * from stud_31 partition (first_class);

select * from stud_31 partition(dist_class);

select partition_name from user_tab_partitions where table_name='STUD_31';

select count(*) AS FAIL from stud_31 partition (fail);

select * from stud_31 where stud_marks = (select max(stud_marks) from stud_31);

ALTER TABLE stud_31 SPLIT PARTITION fail AT (30) INTO (PARTITION
f1,PARTITION f2);

select*from stud_31 partition (f1);

select*from stud_31 partition (f2);

ALTER TABLE stud_31 MERGE PARTITIONS f1, f2 INTO PARTITION pp1;

select * from stud_31 partition(pp1);

ALTER TABLE stud_31 DROP PARTITION dist_class;
```

select*from student04 partition (dist_class);

ALTER TABLE stud_31 ADD PARTITION p_new VALUES LESS THAN (100);

insert into stud_31 values(9,'Ravi',95);

select*from stud_31 partition (p_new);

Q2. Create a table purchase with attributes p_id, p_name and p_amt using range partitioning create the following six partitions -

P1- amount less than 1000,

P2- amount less than 2000,

P3- amount less than 3000,

P4- amount less than 4000,

P5- amount less than 5000,

P6- amount less than 10000

CREATE TABLE purchase04 (p_id INT,p_name CHAR(20),p_amt INT)PARTITION BY RANGE

(p_amt) (PARTITION p1 VALUES LESS THAN (1000),PARTITION p2 VALUES LESS THAN

(2000),PARTITION p3 VALUES LESS THAN (3000),PARTITION p4 VALUES LESS THAN

(4000),PARTITION p5 VALUES LESS THAN (5000),PARTITION p6 VALUES LESS THAN (10000));

INSERT INTO purchase04 VALUES (1, 'Purchase A', 500);

INSERT INTO purchase04 VALUES (2, 'Purchase B', 750);

INSERT INTO purchase04 VALUES (3, 'Purchase C', 1200);

INSERT INTO purchase04 VALUES (4, 'Purchase D', 2500);

INSERT INTO purchase04 VALUES (5, 'Purchase E', 3500);

INSERT INTO purchase04 VALUES (6, 'Purchase F', 4500);

INSERT INTO purchase04 VALUES (7, 'Purchase G', 9500);

INSERT INTO purchase04 VALUES (8, 'Purchase H', 800);

INSERT INTO purchase04 VALUES (9, 'Purchase I', 400);

SELECT *FROM purchase04 WHERE p_amt = (SELECT MAX(p_amt)FROM purchase04 WHERE p_amt <3000);

ALTER TABLE purchase04 SPLIT PARTITION p1 AT (500) INTO (PARTITION pp1,PARTITION pp2);

SELECT * FROM purchase04 PARTITION (pp1);

SELECT * FROM purchase04 PARTITION (pp2);

ALTER TABLE purchase04 MERGE PARTITIONS pp1, pp2 INTO PARTITION new_p1;

rename purchase04 to purchase31;

select * from purchase31 partition(new_p1);

Q3. Create a table tax details with attributes dept_no, name, tax_amt, state with three partitions p1, p2 and

p3 using the partition attribute tax_amt(range partition) partition p1 for tax < 5000, partition p2 for tax <

10000, p3 for tax < 20000.

1. Display the partition wise data.

2. Display the details if the tax amount is greater than 1000

3. Display the department having maximum tax amount

4. Display the state and department having minimum tax amount

5. Drop existing partition p3

6. Create a new partition p4 to store all the values greater than 10000

7. Split the partition p2 to s1 and s2 at 8000

8. Merge the partitions p1 and s1 into p11.

9. Rename the partition p11 to p1_new.

```
CREATE TABLE tax_details31 ( dept_no INT,name CHAR(30),tax_amt INT,state

CHAR(50))PARTITION BY RANGE (tax_amt) (PARTITION p1 VALUES LESS THAN
(5000),

PARTITION p2 VALUES LESS THAN (10000), PARTITION p3 VALUES LESS THAN
(20000));



INSERT INTO tax_details31 VALUES (1, 'Saloni', 3000, 'Maharashtra');

INSERT INTO tax_details31 VALUES (2, 'Priyanka', 7000, 'Punjab');

INSERT INTO tax_details31 VALUES (3, 'Sidhhika', 12000, 'Rajasthan');

INSERT INTO tax_details31 VALUES (4, 'Abhishek', 18000, 'Tamil Nadu');

INSERT INTO tax_details31 VALUES (5, 'Satyam', 900, 'Gujarat');

INSERT INTO tax_details31 VALUES (6, 'Divya', 800, 'Gujarat');

INSERT INTO tax_details31 VALUES (7, 'Anushka', 13000, 'Madhya Pardesh');

INSERT INTO tax_details31 VALUES (8, 'yash', 950, 'Delhi');

INSERT INTO tax_details31 VALUES (9, 'Anjali', 11000, 'Punjab');

INSERT INTO tax_details31 VALUES (10, 'Soham', 18000, 'Madhya Pardesh');



SELECT * FROM tax_details31 PARTITION(p1);

SELECT * FROM tax_details31 PARTITION(p2);

SELECT * FROM tax_details31 PARTITION(p3);

SELECT * FROM tax_details31 WHERE tax_amt > 1000;

SELECT dept_no, name, tax_amt FROM tax_details31 WHERE tax_amt = (SELECT
MAX(tax_amt) FROM tax_details31);

SELECT dept_no, state, tax_amt FROM tax_details31 WHERE tax_amt = (SELECT
MIN(tax_amt) FROM tax_details31);

ALTER TABLE tax_details31 DROP PARTITION p3;
```

```
SELECT PARTITION_NAME FROM USER_TAB_PARTITIONS WHERE
TABLE_NAME = 'TAX_DETAILS31';

ALTER TABLE tax_details31 ADD PARTITION p4 VALUES LESS THAN
(MAXVALUE);

INSERT INTO tax_details31 VALUES (11, 'Raj', 21000, 'Kashmir');

SELECT * FROM tax_details31 PARTITION(p4);

ALTER TABLE tax_details31 SPLIT PARTITION p2 AT (8000) INTO (PARTITION
s1,PARTITION s2);

INSERT INTO tax_details31 VALUES (12, 'Ravi', 9000, 'Haryana');

SELECT * FROM tax_details31 PARTITION(s1);

SELECT * FROM tax_details31 PARTITION(s2);

ALTER TABLE tax_details31 MERGE PARTITIONS p1, s1 INTO PARTITION p11;

ALTER TABLE tax_details31 RENAME PARTITION p11 TO p1_new;

Select * from tax_details31 partition(p1_new);
```

C) List Partition

Q1. Create a table to store customer details custid, cname, state with 4 different partitions for 4 different

regions north, south, east and west using the list partition.

1. Display data from all the partitions.

2. Split the partition south into s1 with Kerala and tamilnadu and s2 with the remaining data.

3. Display the contents of new partition.

4. Merge the partition back.

5. Modify an existing partition east to add Assam and Manipur.

6. Add new partition Central.

7. Truncate the partition west.

```
create table custom_detail(custid number, cname varchar2(25), state varchar2(35)) partition
by list(state)

(partition east values('west bengal','bihar','sikkim'),partition west
values('goa','mumbai','gujrat'), partition

north values('kashmir','punjab','uttarakhand'), partition south
values('kerala','tamilnadu','telangana'));



INSERT INTO custom_detail VALUES (1, 'Abhishek', 'punjab');

INSERT INTO custom_detail VALUES (2, 'Suraj', 'west bengal');

INSERT INTO custom_detail VALUES (3, 'Darshan', 'gujrat');

INSERT INTO custom_detail VALUES (4, 'Saloni', 'tamilnadu');

INSERT INTO custom_detail VALUES (5, 'Kaustubh', 'bihar');

INSERT INTO custom_detail VALUES (6, 'Siddika', 'kashmir');

INSERT INTO custom_detail VALUES (7, 'Yash', 'kerala');

INSERT INTO custom_detail VALUES (8, 'Tanay', 'mumbai');

select * from custom_detail;

select table_name, partition_name from user_tab_partitions where
table_name='CUSTOM_DETAIL';

ALTER TABLE custom_detail SPLIT ('kerala','tamilnadu')INTO(PARTITION
s1,PARTITION s2);

PARTITION south VALUES select * from custom_detail partition(s1);

select * from custom_detail partition(s2);

ALTER TABLE custom_detail MERGE PARTITIONS s1, s2 INTO PARTITION south;

ALTER TABLE custom_detail MODIFY PARTITION east ADD VALUES ('Assam',
'Manipur');

INSERT INTO custom_detail VALUES (11, 'Raju', 'Assam');

INSERT INTO custom_detail VALUES (12, 'Aditi', 'Manipur');
```

ALTER TABLE custom_detail ADD PARTITION central VALUES ('Madhya Pradesh', 'Chhattisgarh');

INSERT INTO custom_detail VALUES (13, 'Reena', 'Madhya Pradesh');

INSERT INTO custom_detail VALUES (14, 'Satyam', 'Chhattisgarh');

select * from custom_detail partition(central);

SELECT PARTITION_NAME FROM USER_TAB_PARTITIONS WHERE TABLE_NAME = 'CUSTOM_DETAIL';

ALTER TABLE custom_detail TRUNCATE PARTITION west;

select * from custom_detail partition(west);

4)

A. Abstract Data Type

Q.1) Create a table customer with attributes cid, name, address and price.

Create an abstract data type Name_Type for the attribute name with fname, lname.

Create an ADT Address_Type for the attribute address with street, city, pincode

1. Display the first name of all the customers.

2. Display the name of all the customers.

3. Display all the details of customer whose first name starts with 'p'.

4. Display the details of customers where city is 'Mumbai'

CREATE TYPE name_type31 AS object (fname VARCHAR(20), lname VARCHAR(20));

/

CREATE TYPE address_type31 AS object (street VARCHAR(20), city VARCHAR(20),pincode VARCHAR(6));

/

create table customers (cid number, name name_Type31, address address_Type31 , price number);

```
insert into customers values(1,
name_Type31('Abhishek','Mhamane'),address_Type31('Manorma
nagar','Mumbai',400604),15000);

insert into customers values(2,
name_Type31('Saloni','Basare'),address_Type31('Juinode','juinagar',400706),15000);

insert into customers values(3,
name_Type31('Pravin','Jadhav'),address_Type31('Manpada','Thane',400706),10000);

insert into customers values(4,
name_Type31('Darshan','Hodge'),address_Type31('Imagica','Panvel',400225),12000);

select * from customers;

SELECT c.name.fname AS First_Name FROM customer c;

SELECT c.name.fname || ' ' || c.name.lname AS Full_Name FROM customer c;

SELECT * FROM customer c WHERE c.name.fname LIKE 'P%';

SELECT * FROM customer c WHERE c.address.city = 'Mumbai';
```

Q2. Create a table with following details using Abstact datatype:

name_type

- **Fname**
- **Lname**

address _type

- **Street**
- **City**
- **Pin code**

Author_type

- **Name**
- **Address**

publisher_type

• Name

• Address

Create the table BOOK with following attributes

• Book id

• Book title

• Price

• Author

• Publisher

1. Display all the books published by “TMH” .

2. Display the first name of all publishers.

3.Display first name of all authors.

4. Display all books details written by author with fname 'Rahul’

5. Display all the information from BOOK table where price in between 250 and 400 where the

Author is from ‘Mumbai’ and ‘Delhi’

6. Display the number of books published by each author.

7. Display the name of author who wrote only one book.

```
CREATE TYPE name_type31 AS OBJECT(fname VARCHAR2(50) lname
VARCHAR2(50));

/



CREATE TYPE address_type31 AS OBJECT ( street VARCHAR2(100)city
VARCHAR(50), pincode VARCHAR(10));

/
```

```
CREATE TYPE author_type31 AS OBJECT( name name_type31, address address_type31);

/



CREATE TYPE publisher_type31 AS OBJECT( name name_type31,address
address_type31);

/



CREATE TABLE BOOK31 (book_id NUMBER ,book_title VARCHAR2(30),price
NUMBER,author AUTHOR_TYPE31,publisher PUBLISHER_TYPE31);



INSERT INTO book31 VALUES (1, 'Intro to Programming', 250,
author_type31(name_type31('Ankit', 'Joshi'), address_type31('Green Park', 'Delhi', 110016)),

publisher_type31(name_type31('TMH', 'Press'), address_type31('West End', 'Delhi',
110008)));

INSERT INTO book31 VALUES (2, 'Database Systems', 400,
author_type31(name_type31('Maya', 'Patil'), address_type31('Navi Rd', 'Mumbai', 400705)),

publisher_type31(name_type31('Pearson', 'Education'), address_type31('Marine Lines',
'Mumbai',400020)));

INSERT INTO book31 VALUES (3, 'Artificial', 520, author_type31(name_type31('Ankit',
'Joshi'), address_type31('Whitefield', 'Mumbai', 450066)),

publisher_type31(name_type31('TMH', 'Publishers'), address_type31('MG Road', 'Delhi',
420001)));

INSERT INTO book31 VALUES (4, 'Machine Learning Handbook',
475,author_type31(name_type31('Rajesh', 'Singh'), address_type31('Sector 22', 'Gurgaon',
122015)),

publisher_type31(name_type31('Reilly', 'Media'), address_type31('Cyber Hub', 'Gurgaon',
122002)));

INSERT INTO book31 VALUES (5, 'Cloud Computing Basics',
360,author_type31(name_type31('Simran', 'Kaur'), address_type31('Civil Lines', 'Delhi',
412006)),

publisher_type31(name_type31('TMH', 'Publishers'), address_type31('Station Rd', 'Mumbai',
412001)));
```

INSERT INTO book31 VALUES (6, 'Cybersecurity Fundamentals', 290, author_type31(name_type31('Ankit', 'Joshi'), address_type31('Sector 45', 'Noida', 201301)),

publisher_type31(name_type31('TMH', 'Publishers'), address_type31('Knowledge Park', 'Noida', 201310)));

INSERT INTO book31 VALUES (7, 'Big Data Analysis', 230, author_type31(name_type31('Rahul', 'Mishra'), address_type31('Connaught Place', 'Delhi', 110001)),

publisher_type31(name_type31('Springer', 'Nature'), address_type31('North Campus', 'Mumbai',410007)));

SELECT * FROM book31 b WHERE b.publisher.name.fname = 'TMH';

SELECT DISTINCT b.publisher.name.fname AS "Publisher First Name" FROM BOOK31 b;

SELECT b.publisher.name.fname AS "Publisher_First_Name" FROM BOOK31 b;

SELECT * FROM BOOK31 b WHERE b.author.name.fname = 'Rahul';

SELECT * FROM BOOK31 b WHERE price BETWEEN 250 AND 400 AND b.author.address.city IN ('Mumbai', 'Delhi');

SELECT b.author.name.fname || ' ' || b.author.name.lname AS "Author Name",

COUNT(*) AS "Number of Books" FROM BOOK31 b GROUP BY b.author.name.fname, b.author.name.lname;

SELECT b.author.name.fname || ' ' || b.author.name.lname AS "Author Name" FROM BOOK31 b GROUP BY b.author.name.fname, b.author.name.lname

HAVING COUNT(*) = 1;

5)

A. Object Table

1. Create an object table Person_Table with the attributes (id,pname,designation ,sal,location)

2. Insert 5 records.

3. Display the managers from Mumbai location

CREATE OR REPLACE TYPE Person_Type31 AS OBJECT (id NUMBER,pname VARCHAR2(20),designation VARCHAR2(20),sal NUMBER, location VARCHAR2(20));

```
/

CREATE TABLE Person_Table31 OF Person_Type31;

INSERT INTO Person_Table31 VALUES (Person_Type31(1, 'Abhishek', 'Manager', 60000,
'Mumbai'));

INSERT INTO Person_Table31 VALUES (Person_Type31(2, 'Saloni', 'Assistant Manager',
50000, 'Pune'));

INSERT INTO Person_Table31 VALUES (Person_Type31(3, 'Darshan', 'Manager', 65000,
'Mumbai'));

INSERT INTO Person_Table31 VALUES (Person_Type31(4, 'Suraj', 'Developer', 40000,
'Bangalore'));

INSERT INTO Person_Table31 VALUES (Person_Type31(5, 'Parvin', 'Manager', 62000,
'Mumbai'));

Select* from person_table31;

SELECT * FROM Person_Table31 WHERE designation = 'Manager' AND location =
'Mumbai';
```

6) Inheritance

Q1. Create a type person_type with attributes person_id, p_name, p_address. Create a

typestudent under person_type with the attributes dept_name and major subjects. Create a

typeemp_type under person_type with attributes emp_id and manager_name. Create a

type part_time_student _type under student with attributes no. of hours. Create a table

personas object table of person_type. (Attached Image 1)

Person Type

```
create type person_typeB31 as object

(p_id number, p_name varchar2(15), p_address varchar2(20))NOT FINAL;

/
```

Student Type

```
create type student_typeB31 UNDER person_typeB31

(dept_name varchar2(15), major_subject varchar2(20)) not final;

/
```

Employee Type

```
create type employee_typeB31 UNDER person_typeB31

(emp_id number, manager_name varchar2(15));

/
```

Part Time Student Type

```
create type parttime_student_typeB31 UNDER student_typeB31

(no_of_hours number);

/
```

Person Table

```
create table personB31 of person_typeB31;
```

Insert Query

```
insert into personB31 values(person_typeB31(1, 'Abhishek', 'Thane'));

insert into personB31 values(person_type31(2, 'Darshan', 'Panvel'));

insert into personB31 values(person_typeB31(3, 'Saloni', 'Juinagar'));

insert into personB31 values(person_typeB31(4, 'Suraj', 'Thane'));

insert into personB31 values(person_typeB31(5, 'Raj', 'Panvel'));

insert into personB31 values(student_typeB31(101, 'Abhishek', 'Thane', 'MCA', 'Java'));

insert into personB31 values(student_typeB31(102, 'Darshan', 'Panvel', 'MCA', 'ADBMS'));

insert into personB31 values(student_typeB31(103, 'Saloni', 'Juinagar', 'PHD', 'C++'));

insert into personB31 values(student_typeB31(104, 'Suraj', 'Thane', 'BSC', 'IT'));

insert into personB31 values(student_typeB31(105, 'Raj', 'Panvel', 'Mtech', 'Python'));
```

insert into personB31 values(employee_typeB31(101,'Abhishek', 'Thane', 101, ' Abhishek Mha'));

insert into personB31 values(employee_typeB31(102,'Darshan', 'Panvel', 102, 'Darshan Hod'));

insert into personB31 values(employee_typeB31(103,'Saloni', 'Juinage', 103, 'Saloni Basare'));

insert into personB31 values(employee_typeB31(104,'Suraj', 'Thane', 104, 'Suraj Jadhav'));

insert into personB31 values(employee_typeB31(105,'Raj', 'Panvel', 105, 'Raj Patil'));

1. Display all the details of the table

select * from personB31;

2. Display the details of the students.

select value(p) from PersonB31 p where value(p) is of (student_typeB31);

3. Display all the major subjects of student

select TREAT(value(p) AS student_typeB31).major_subject from PersonB31 p where TREAT(value(p) AS student_typeB31) IS NOT NULL;

4. Display the name of the manager of employee with p_id=101

select TREAT(value(p) AS student_typeB31).major_subject from PersonB31 p where TREAT(value(p) AS student_typeB31) IS NOT NULL;

Q2. Create a type shape_type with attribute shape+id, length.

Create a type under shape_type square_type,

CREATE TABLE Square (shape_id NUMBER PRIMARY KEY area_square NUMBER, FOREIGN KEY (shape_id) REFERENCES Shape(shape_id));

rectangle_type,

CREATE TABLE Rectangle C shape_id NUMBER PRIMARY KEY, Length NUMBER, breadth NUMBER area_rectangle NUMBER, FOREIGN KEY (shape_id) REFERENCES Shape(shape_id));

triangle_type.

CREATE TABLE Triangle (shape_id NUMBER PRIMARY KEY, Length NUMBER, base NUMBER, area_triangle NUMBER FOREIGN KEY (shape_id) REFERENCES Shape(shape_id));

Square(area_square()) as member function and

rectangle(breadth,area_rectangle()), triangle(base,area_triangle()).

1. Display all the length values of rectangle between 0-10.

select length from rectangle where length between 0 and 10;

2. Display the area of triangle having shape id 107

select area_Triangle from Triangle where shape_id=107;

3. Display all info of rectangle where area>3

select * from rectangle where area_rectangle >3 ;

4. Display the length and breadth of all rectangle

select length,breadth from rectangle ;

A. Rollup, Partial Rollup, Cube, Partial Cube

Q1. Create a table Sales with the attribute dept_id, deptname, year_of_sales,region and

profit.

CREATE TABLE Sales (dept id NUMBER, deptname VARCHAR2(50), year of sales NUMBER, region VARCHAR2(50),profit NUMBER(10, 2));

select * from sales;

Perform the rollup operation on this table.

1. Display year wise total profit.

SELECT year_of_sales, SUM(profit) AS total_profit FROM Sales GROUP BY ROLLUP(year_of_sales);

2. Display year wise total profit of each region.

SELECT year_of_sales, region, SUM(profit) AS total_profit FROM Sales GROUP BY ROLLUP(year_of_sales, region);

3. Display year wise, region wise and department wise total profit for the department "IT".

SELECT year_of_sales, region, deptname, SUM(profit) AS total_profit FROM
Sales WHERE deptname = 'IT' GROUP BY ROLLUP(year_of_sales, region, deptname);

4. Display year wise total profit of each department.

SELECT year_of_sales, deptname, SUM(profit) AS total_profit FROM Sales GROUP BY ROLLUP(year_of_sales, deptname);

5. Display region wise total profit of each department.

SELECT region,deptname,SUM(profit) AS total_profit FROM Sales GROUP BY ROLLUP(region, deptname);

6. Display region wise total profit if total profit >28

SELECT region, SUM(profit) AS total_profit FROM Sales GROUP BYregion HAVING SUM(profit) > 28;

7. Display region wise total profit.

SELECT region, SUM(profit) AS total_profit FROM Sales GROUP BY ROLLUP(region);

8. Display department wise total profit.

SELECT deptname, SUM(profit) AS total_profit FROM Sales GROUP BY ROLLUP(deptname);

Q2. Apply partial rollup on same table. Display region wise total profit of each department

by partially rolling the year.

SELECT year_of_sales, region, deptname, SUM(profit) AS total_profit FROM Sales GROUP BY year_of_sales, ROLLUP(region, deptname);

Q3. Implement Cube operation on the same table.

1. Display year, region and dept wise total profit using cube function.

2. Display region and dept wise total profit using year_of_sales as partial cube dimension

2) B. Rank and Dense Rank

Q1. Create a table student with attribute roll_num, name, subject, marks.

1. Display content of table.

CREATETABLEstudentB (roll_num INT, name VARCHAR(100), subject VARCHAR(100), marks INT);

INSERT INTO studentB (roll_num, name, subject, marks) VALUES

(1, 'Suraj', 'Math', 85),

(2, 'Darshan', 'Math', 90),

(3, 'Abhishek', 'Math', 85),

(4, 'Omkar', 'Math', 95),

(5, 'Amy', 'Math', 90),

(6, 'John', 'Science', 78),

(7, 'Sara', 'Science', 82),

(8, 'Tina', 'Science', 78),

(9, 'Peter', 'Science', 88),

(10, 'Alex', 'Math', 90),

(11, 'Maya', 'Science', 82),

(12, 'James', 'Science', 78),

(13, 'David', 'Science', 88);

select * from studentB ;

2. Assign sequence order for the student for the same subject based on their marks.

SELECT roll_num, name, subject, marks RANK() OVER(PARTITION BYsubject ORDER BY marks) AS rank FROM studentB;

3. Assign sequential order for the student for the same subject based on their marks in descending order.

4. Assign sequential order using dense rank function.

SELECT roll_num, name, subject, marks, RANK() OVER(PARTITION BYsubject ORDER BY marks DESC) AS rank_desc FROM studentB;

3)

C. FIRST AND LAST

Q1.) 1. Display the lowest marks of each subject.

SELECT subject, FIRST_VALUE(marks) OVER (PARTITION BY subject ORDER BY marks ASC) AS first_marks FROM studentB;

1. Display the highest marks of each subject

SELECT subject,

LAST_VALUE(marks) OVER (PARTITION BY subject ORDER BY marks DESC ROWS)

ASlast_marks

FROMstudentB;

4)

D. LEADANDLAG

Q1. Create a table employee with attribute empid, name, deptid,deptname, salary and joining date.

CREATETABLEemployee (

empid INT PRIMARY KEY,

name VARCHAR(24),

deptid INT,

deptname VARCHAR(24),

salary DECIMAL(10, 2),

joining_date DATE

);

```
INSERT INTO employee (empid, name, deptid, deptname, salary, joining_date) VALUES
(1, 'Suraj', 101, 'HR', 50000.00, TO_DATE('2023-01-10', 'YYYY-MM-DD'));
INSERT INTO employee (empid, name, deptid, deptname, salary, joining_date) VALUES
(2, 'Darshan', 102, 'IT', 60000.00, TO_DATE('2022-06-15', 'YYYY-MM-DD'));
INSERT INTO employee (empid, name, deptid, deptname, salary, joining_date) VALUES
(3, 'Omkar', 101, 'HR', 55000.00, TO_DATE('2021-11-22', 'YYYY-MM-DD'));
INSERT INTO employee (empid, name, deptid, deptname, salary, joining_date) VALUES
(4, 'Abhishek', 103, 'Finance', 65000.00, TO_DATE('2020-09-30', 'YYYY-MM-DD'));
INSERT INTO employee (empid, name, deptid, deptname, salary, joining_date) VALUES
(5, 'Ravi', 104, 'Marketing', 48000.00, TO_DATE('2023-02-25', 'YYYY-MM-DD'));
INSERT INTO employee (empid, name, deptid, deptname, salary, joining_date) VALUES
(6, 'Priya', 102, 'IT', 70000.00, TO_DATE('2021-08-10', 'YYYY-MM-DD'));
INSERT INTO employee (empid, name, deptid, deptname, salary, joining_date) VALUES
(7, 'Vikas', 103, 'Finance', 60000.00, TO_DATE('2020-12-05', 'YYYY-MM-DD'));
INSERT INTO employee (empid, name, deptid, deptname, salary, joining_date) VALUES
(8, 'Nisha', 101, 'HR', 52000.00, TO_DATE('2023-04-18', 'YYYY-MM-DD'));
INSERT INTO employee (empid, name, deptid, deptname, salary, joining_date) VALUES
(9, 'Amit', 104, 'Marketing', 51000.00, TO_DATE('2022-05-12', 'YYYY-MM-DD'));
INSERT INTO employee (empid, name, deptid, deptname, salary, joining_date) VALUES
(10, 'Pooja', 102, 'IT', 68000.00, TO_DATE('2021-09-10', 'YYYY-MM-DD'));
```

1. Display all the details of table.

```
select * from employee ;
```

2. Display the joining details of the entire employee joined just after the joining date of each employee in sales department.

```
SELECT empid, name, deptname, salary, joining_date,
```

LEAD(empid) OVER (PARTITION BY deptname ORDER BY joining_date) AS next_empid,

LEAD(name) OVER (PARTITION BY deptname ORDER BY joining_date) AS next_name,

LEAD(joining_date) OVER (PARTITION BY deptname ORDER BY joining_date) AS

next_joining_date

FROMemployee

WHEREdeptname = 'Sales';

3. Display joining date of all employee joined just before the joining date of employee

SELECT empid, name, deptname, salary, joining_date,

LAG(joining_date) OVER (PARTITION BY deptname ORDER BY joining_date) AS

prev_joining_date

FROM employee

ORDER BYdeptname, joining_date;

4. For each employee in employee table display the salary of the employee joined just before.

SELECT empid, name, deptname, salary, joining_date,

LAG(salary) OVER (PARTITION BY deptname ORDER BY joining_date) AS prev_salary

FROMemployee

ORDERBYdeptname, joining_date;

E. Windowing functions

Q1). Create a table employee with attribute emp_no , emp_name , dept_name and salary.

CREATETABLEemployeebb (

emp_no INT PRIMARY KEY,

emp_name VARCHAR(24),

```
dept_name VARCHAR(24),

salary DECIMAL(10, 2)

);
```

Insert data

```
INSERT INTO employeebb (emp_no, emp_name, dept_name, salary)

VALUES

(1, 'Suraj', 'HR', 50000.00),

(2, 'Darshan', 'IT', 60000.00),

(3, 'Abhishek', 'Finance', 65000.00),

(4, 'Omkar', 'Marketing', 48000.00),

(5, 'John', 'Sales', 45000.00),

(6, 'Alice', 'Sales', 47000.00),

(7, 'Bob', 'HR', 55000.00),

(8, 'Sara', 'IT', 51000.00),

(9, 'Eve', 'Marketing', 53000.00),

(10, 'Charlie', 'Finance', 58000.00);
```

Display emp_no,emp_name,dept_name,salary and dept wise sum of salary of current and previous

two records.

```
SELECT emp_no,

emp_name,

dept_name,

salary,

SUM(salary) OVER (PARTITION BY dept_name ORDER BY emp_no

ROWSBETWEEN2PRECEDINGANDCURRENTROW)ASdept_salary_sum
```

```
FROMemployeebb

ORDERBYdept_name, emp_no;
```

Display emp_no,emp_name,dept_name,salary and sum of salary for 3 earlier row and 1 next row

dept wise.

```
SELECT emp_no,

emp_name,

dept_name,

salary,

SUM(salary) OVER (PARTITION BY dept_name ORDER BY emp_no

ROWSBETWEEN3PRECEDINGAND1FOLLOWING)ASdept_salary_sum

FROMemployeebb

ORDERBYdept_name, emp_no;
```

Display emp_no.emp_name,dept_name,salary and sum of salary 3 preceding row and 1

preceding row dept wise

```
SELECT emp_no,

emp_name,

dept_name,

salary,

SUM(salary) OVER (PARTITION BY dept_name ORDER BY emp_no

ROWS BETWEEN 3 PRECEDING AND 1 PRECEDING)AS dept_salary_sum

FROMemployeebb

ORDERBYdept_name, emp_no;
```

Display emp_no.emp_name,dept_name,salary and sum of salary 1 following and 3

following row dept wise.

```
SELECT emp_no,
emp_name,
dept_name,
salary,
SUM(salary) OVER (PARTITION BY dept_name ORDER BY emp_no
ROWSBETWEEN1FOLLOWINGAND3FOLLOWING)ASdept_salary_sum
FROMemployeebb
ORDERBYdept_name, emp_no;
```

R Preprocessing

Q1. Implementation of Data preprocessing techniques in R

Naming and renaming variables

```
data = read.table(file="adbms.csv", sep = ",")
data[1:9,]
names(data)


data = read.table(file="adbms.csv", sep = ",", header = T)
data[1:8,]
names(data)
```

# Adding headers

```
data = read.csv(file="adbms.csv",col.names=c("EID","NAME"))
data[1:8,]
names(data)
```

# Renaming Headers

```
names(data) <- c("EID_renamed","NAME_renamed")

data[1:8, ]

names(data)
```

adding a new variable.

# Add new column to data frame

```
data <- read.csv(file = "adbms1.csv", col.names = c("ID", "NAME",
"MATHS","SCIENCE"))

data1$TOTMKS <- data1$MATHS + data1$SCIENCE

data1$MEANMKS <- (data1$MATHS + data1$SCIENCE)/2

data1
```

Dealing with categorical data.

```
data1$RESULT[data1$MATHS<50|data1$SCIENCE<50]<-"FAIL"

data1$RESULT[data1$MATHS>=50|data1$SCIENCE>=50]<-"PASS"

data1

data1[which.min(data1$MATHS), ]

data1[which.max(data1$SCIENCE), ]

mean(data1$MATHS)

median(data1$MATHS)

mean(data1$SCIENCE)

median(data1$SCIENCE)
```

Dealing with missing data.

```
NA+4

V=c(1,2,NA,3,4,5,6,7,8)

V

sum(V)
sum(V,na.rm=T)

is.na(V)

naVals=is.na(V)

dataNA=read.csv(file="adbms1.csv",na.strings = "")

dataNA

dim(dataNA)
```

Data reduction using subsetting

```
library(caTools)

set.seed(155)

split=sample.split(iris,SplitRatio = 0.75)

split

train=subset(iris,split==T)

head(train)

test=subset(iris,split==F)

head(test)

dim(train)

dim(test)
```

Data Mining

Q1. Implementation of Linear Regression in R

#Linear Regression

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

plot(x,y,col="blue",main="X-Y plot for Regression",pch=8,xlab="Height",ylab="Weight")
```

Output :

```
reg = lm(y~x)

print(summary(reg))
```

Output :

```
plot(x,y,col="blue",main="Height & Weight

Regression",pch=10,xlab="Height",ylab="Weight")

abline(reg,col="red")
```

Output :

```
height=data.frame(x=165)

predicted_weight=predict(reg,newdata = height)

predicted_weight
```

Output :

```
height=data.frame(x=c(165,170))

predicted_weight=predict(reg,newdata = height)

predicted_weight
```

Output :

Classification

Q2. Implementation and analysis of Classification algorithms:

Naive Bayesian, K-Nearest Neighbor, ID3, C4.5

a. Naive Bayes

```
Library(e1071)

train=iris[1:120,]

train

test=iris[121:150,]

test

nbmodel=naiveBayes(Species~.,data=train)

pred=predict(nbmodel,test)

pred

table(pred,iris[121:150,5])

accuracy=sum(diag(cm))/length(test$Species)

accuracy

accuracy*100
```

b. KNN

```
install.packages("class")

library(class, lib.loc = "C:/Program Files/R/R-4.4.2/library")

iris

train=iris[1:120,-5]

train

test=iris[121:150,-5]

test

pred=knn(train,test,iris[1:120,5],k=100)

pred
```

Output :

```
table(pred, iris[121:150,5])
```

Output :

```
model = knn(train,test,iris[1:120,5],k=6)

summary(model)
```

Output:

```
cm = table(iris[121:150,5],model)

accuracy = sum(diag(cm))/length(iris[121:150,5])

sprintf("Accuracy: %.2f%%", accuracy*100)
```

Output:

c. ID3/C4.5/J48

```
library(RWeka)

train = iris[1:120,]

test = iris[121:150,]

fit <- J48(Species~., train)

pred=predict(fit,test)

pred

cm=table(test$Species,pred)

cm

accuracy=sum(diag(cm))/length(test$Species)

accuracy*100
```

Q3. Implementation and analysis of Apriori Algorithm using Market Basket Analysis.

```
df_groceries <- read.csv("Groceries_dataset.csv",header=T)

df_groceries
```

```
df_sorted <- df_groceries[order(df_groceries$Member_number),]

df_sorted$Member_number <- as.numeric(df_sorted$Member_number)

library(plyr)

df_itemList <- ddply(df_groceries,c("Member_number","Date"),

function(df1)paste(df1$itemDescription,collapse = ","))

df_itemList


df_itemList$Member_number <- NULL

df_itemList$Date <- NULL

colnames(df_itemList) <- c("ItemList")

df_itemList

write.csv(df_itemList,"Grocery_ItemList1.csv", row.names = TRUE)

txn = read.transactions(file="Grocery_ItemList1.csv", rm.duplicates= TRUE,

format="basket",sep=",",cols=1);

txn

basket_rules <- apriori(txn,parameter = list(sup = 0.01, conf = 0.01));

print(basket_rules)

inspect(basket_rules)

plot(basket_rules)

#Graph to display top 5 items

itemFrequencyPlot(txn, topN = 5)
```

Clustering

Q4. Implementation and analysis of clustering algorithms: K Means and Agglomerative

K means

```
iris

irisdata = iris[,-5]

irisdata

head(irisdata)

plot(irisdata)

set.seed(789)


clust = kmeans(irisdata, centers = 3, iter.max = 6)

clust

clust$cluster

iris$Species


cm = table(iris$Species,clust$cluster)

cm

plotcluster(irisdata,clust$cluster)

acc= sum(diag(cm))/length(iris[121:150,5])

sprintf("Accuracy: %.2f%%",acc*100)


data.point=read.csv("C:\\Users\\Student\\Downloads\\seeds_dataset1.csv",header=T)

data.point

distMat <- dist(data.point,method = "euclidean")

distMat

Clust1 <- hclust(distMat,method="single")

Clust1
```

```
plot(Clust1)

dend <- as.dendrogram(Clust1)

plot(dend)
```