**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)