ADVANCED DATABASE MANAGEMENT SYSTEM

LAB ASSIGNMENT FILE

SUBMITTED TO:
MS VIMMI KOCHHER
AMITY SCHOOL OF
ENGINEERING & TECHNOLOGY

SUBMITTED BY:
SHRUTI KUMARI
M.TECH (DS)
A50568424006

INDEX

Date – 11/09/2024

Practical No : 01

DDL Commands

The DDL Commands in Structured Query Language are used to create and modify the schema of the database and its objects. DDL consist of Commands to commands like CREATE, ALTER, TRUNCATE and DROP.

DDL Commands :

1. **CREATE** :
   This command is used to create a new table in SQL. The user has to give information
   like table name, column names, and their datatypes.

   Syntax :
   CREATE TABLE table_name
   (
   column_1 datatype,
   column_2 datatype,
   column_3 datatype,
   ....
   );

1. **ALTER** :
   This command is used to add, delete or change columns in the existing table. The user needs to know the existing table name and can do add, delete or modify tasks easily.

   Syntax :
   a. ALTER TABLE table_name
      ADD column_name datatype;
   b. ALTER TABLE table_name
      DROP Column_name;
   c. ALTER TABLE table_name
      MODIFY (column_name column_datatype(size));

2. **TRUNCATE :**
   This command is used to remove all rows from the table, but the structure of the table still exists.

   Syntax :
   TRUNCATE TABLE table_name;

3. **DROP :**
   This command is used to remove an existing table along with its structure from the Database.
   Syntax :
   DROP TABLE table_name;

4. **RENAME :**
   It is possible to change name of table with or without data in it using simple RENAME command. We can rename any table object at any point of time.

   Syntax :
   RENAME TABLE <Table Name> To <New_Table_Name>;

CODE OF DDL COMMANDS

1. CREATE TABLE Student (
   Id int primary key,
   Firstname varchar(50),
   Lastname varchar(50),
   Age int,
   Gender char(6),
   Address varchar(50),
   Email varchar(50),
   Phone varchar(10),
   Couse varchar(50)
)

ID	FIRSTNAME	LASTNAME	AGE	GENDER	ADDRESS	EMAIL	PHONE	COURSE
1	Shruti	Kumari	24	F	xyz	abc@gmail.com	897851	M.Tech
2	Nidhi	Upadhayay	25	F	pqr	bcd@gmail.com	256398	MA
3	Sadaf	Khalil	23	F	lmp	efg@gmail.com	845632	MSC
4	Ankita	Singh	26	F	stu	hij@gmail.com	784512	MCA
5	Ravi	Thakur	30	M	rsn	klm@gmail.com	956478	MLLB
6	Adarsh	Kumar	21	M	cde	nmo@gmail.com	145236	MPharm

2. ALTER TABLE Student
   DROP COLUMN Email;

ID	FIRSTNAME	LASTNAME	AGE	GENDER	ADDRESS	PHONE	COURSE
1	Shruti	Kumari	24	F	xyz	897851	M.Tech
2	Nidhi	Upadhayay	25	F	pqr	256398	MA
3	Sadaf	Khalil	23	F	lmp	845632	MSC
4	Ankita	Singh	26	F	stu	784512	MCA
5	Ravi	Thakur	30	M	rsn	956478	MLLB
6	Adarsh	Kumar	21	M	cde	145236	MPharm

3. TRUNCATE TABLE Student;
4. DROP TABLE employee;
5. RENAME COLUMN Id TO Std_id;

STU_ID	FIRSTNAME	LASTNAME	AGE	GENDER	ADDRESS	PHONE	COURSE
1	Shruti	Kumari	24	F	xyz	897851	M.Tech
2	Nidhi	Upadhayay	25	F	pqr	256398	MA
3	Sadaf	Khalil	23	F	lmp	845632	MSC
4	Ankita	Singh	26	F	stu	784512	MCA
5	Ravi	Thakur	30	M	rsn	956478	MLLB
6	Adarsh	Kumar	21	M	cde	145236	MPharm

DATE: 11/09/2024

PRACTICAL NO : 02

DML COMMANDS: DML is an abbreviation of Data Manipulation Language. The DML commands in Structured Query Language change the data present in the SQL database. We can easily access, store, modify, update and delete the existing records from the database using DML commands.

The four main DML commands in SQL are followings:

1. **SELECT :** The SELECT command shows the records of the specified table. It also shows the particular record of a particular column by using the WHERE clause.
   Syntax :
   SELECT column_Name_1, column_Name_2, ….., column_Name_N F ROM Name_of_table

2. **INSERT :** INSERT is another most important data manipulation command in Structured Query Language, which allows users to insert data in database tables.
   Syntax :
   INSERT INTO TABLE_NAME (column_Name1 , column_Name2 , colum n_Name3 , ... column_NameN) VALUES (value_1, value_2, value_3, ... v alue_N) ;

3. **UPDATE :** UPDATE is another most important data manipulation command in Structured Query Language, which allows users to update or modify the existing data in database tables.
   Syntax :
   UPDATE Table_name SET [column_name1= value_1, ….., column_nameN = value_N] WHERE CONDITION;

4. **DELETE** : DELETE is a DML command which allows SQL users to remove single or multiple existing records from the database tables.
   Syntax :
   DELETE FROM Table_Name WHERE condition;

Code for DML commands :

1. Select * from employee;

EID	ENAME	DEPARTMENT	DEPTNO	SALARY	HIREDATE
1	SHRUTI	IT	101	50000	2022-05-30
2	SADAF	HR	102	40000	2023-01-15
3	NIDHI	PRODUCTION	103	30000	2023-06-01
4	ANIMESH	ACCOUNTANT	104	60000	2021-05-20
5	ZEESHAN	SALES	105	20000	2023-06-21

2. INSERT INTO EMPLOYEES(EID , ENAME, DEPARTMENT,DEPTNO,SALARY,HIREDATE) VALUES
(1,'SHRUTI','IT',101,50000,'2022-05-30'),
(2,'SADAF','HR',102,40000,'2023-01-15'),
(3,'NIDHI','PRODUCTION',103,30000,'2023-06-01'),
(4,'ANIMESH','ACCOUNTANT',104,60000,'2021-05-20'),
(5,'ZEESHAN','SALES',105,20000,'2023-06-21');

3. UPDATE EMPLOYEES SET DEPARTMENT = ‘HR’ WHERE DEPTNO = 101;

EID	ENAME	DEPARTMENT	DEPTNO	SALARY	HIREDATE
1	SHRUTI	HR	101	50000	2022-05-30
2	SADAF	HR	102	40000	2023-01-15
3	NIDHI	PRODUCTION	103	30000	2023-06-01
4	ANIMESH	ACCOUNTANT	104	60000	2021-05-20
5	ZEESHAN	SALES	105	20000	2023-06-21

4. DELETE FROM EMPLOYEES WHERE DEPTNO = 104;

EID	ENAME	DEPARTMENT	DEPTNO	SALARY	HIREDATE
1	SHRUTI	HR	101	50000	2022-05-30
2	SADAF	HR	102	40000	2023-01-15
3	NIDHI	PRODUCTION	103	30000	2023-06-01
5	ZEESHAN	SALES	105	20000	2023-06-21

DATE : 11/09/2024

PRACTICAL NO : 03

AIM - Using Relational, Logical Operators.

1. **Create Table Employee1.**
   CREATE TABLE EMPLOYEES1(
   EMPNO INT,
   ENAME VARCHAR(50),
   JOB VARCHAR(50),
   MANAGER_NAME VARCHAR(50),
   HIREDATE DATE,
   SALARY INT,
   COMMISSION INT,
   DEPARTMENT VARCHAR(50)
   );

2. **Insert values into employee1 table.**
   INSERT INTO EMPLOYEES1 (EMPNO, ENAME, JOB,
   MANAGER_NAME, HIREDATE, SALARY, COMMISSION,
   DEPARTMENT) VALUES
   (1, 'John', 'salesman', 'Bob', '2022-01-01', 4500, 10, 'sales'),
   (2, 'Mike', 'analyst', 'Sam', '2022-01-15', 4200, 20, 'analyst'),
   (3, 'Sam', 'manager', null, '2022-03-01', 4800, 30, 'manager'),
   (4, 'Bob', 'salesman', 'Sam', '2022-04-01', 4000, 40, 'sales'),
   (5, 'Sara', 'analyst', 'Bob', '2022-05-01', 4100, 50, 'analyst'),
   (6, 'David', 'salesman', 'Sara', '2022-06-01', 4300, 60, 'sales'),
   (7, 'Emily', 'analyst', 'David', '2022-07-01', 4400, 70, 'analyst'),
   (8, 'Kate', 'salesman', 'Emily', '2022-08-01', 4500, 80, 'sales'),
   (9, 'Olivia', 'analyst', 'Kate', '2022-09-01', 4600, 90, 'analyst'),
   (10, 'Sophia', 'salesman', 'Olivia', '2022-10-01', 4700, 100, 'sales');

3. **Create department table and insert values in it.**
   CREATE TABLE DEPARTMENT(
   DEPTNO INT,
   DEPTNAME VARCHAR(50),
   DEPT_LOCATION VARCHAR(50));

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME,
DEPT_LOCATION)
VALUES  (10, 'sales', 'New York'),
(20, 'analyst', 'Chicago'),
(30, 'manager', 'Los Angeles'),
(40, 'sales', 'New York'),
(50, 'analyst', 'Chicago'),
(60, 'manager', 'Los Angeles'),
(70, 'sales', 'New York'),
(80, 'analyst', 'Chicago'),
(90, 'manager', 'Los Angeles'),
(100, 'sales', 'New York');
```

4. **List all the information about all employees from emp table.**

```
SELECT * FROM EMPLOYEES1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMPNO	ENAME	JOB	MANAGER_NAME	HIREDATE	SALARY	COMMISSION	DEPARTMENT
1	John	salesman	Bob	2022-01-01	4500	10	sales
2	Mike	analyst	Sam	2022-01-15	4200	20	analyst
3	Sam	manager	NULL	2022-03-01	4800	30	manager
4	Bob	salesman	Sam	2022-04-01	4000	40	sales
5	Sara	analyst	Bob	2022-05-01	4100	50	analyst
6	David	salesman	Sara	2022-06-01	4300	60	sales
7	Emily	analyst	David	2022-07-01	4400	70	analyst
8	Kate	salesman	Emily	2022-08-01	4500	80	sales
9	Olivia	analyst	Kate	2022-09-01	4600	90	analyst
10	Sophia	salesman	Olivia	2022-10-01	4700	100	sales

5. **List all the information about all departments from department table.**
   SELECT * FROM DEPARTMENT;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DEPTNO	DEPTNAME	DEPT_LOCATION
10	sales	New York
20	analyst	Chicago
30	manager	Los Angeles
40	sales	New York
50	analyst	Chicago
60	manager	Los Angeles
70	sales	New York
80	analyst	Chicago
90	manager	Los Angeles
100	sales	New York

6. **List all employees names along with their salary from employee table.**
   SELECT ENAME, SALARY FROM EMPLOYEES1;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME	SALARY
John	4500
Mike	4200
Sam	4800
Bob	4000
Sara	4100
David	4300
Emily	4400
Kate	4500
Olivia	4600
Sophia	4700

7. **List all department name, employee number, manager name from employee table.**
   SELECT DEPARTMENT,EMPNO,MANAGER_NAME FROM EMPLOYEES1;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DEPARTMENT	EMPNO	MANAGER_NAME
sales	1	Bob
analyst	2	Sam
manager	3	NULL
sales	4	Sam
analyst	5	Bob
sales	6	Sara
analyst	7	David
sales	8	Emily
analyst	9	Kate
sales	10	Olivia

8. **List dept names and location from department table.**
   SELECT DEPTNAME, DEPT_LOCATION from DEPARTMENT;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DEPTNAME	DEPT_LOCATION
sales	New York
analyst	Chicago
manager	Los Angeles
sales	New York
analyst	Chicago
manager	Los Angeles
sales	New York
analyst	Chicago
manager	Los Angeles
sales	New York

9. **List the employees belong to department number=50.**
   SELECT * FROM DEPARTMENT WHERE DEPTNO = 50;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DEPTNO	DEPTNAME	DEPT_LOCATION
50	analyst	Chicago

10.List employee names and salary whose salary is greater than 1000.

SELECT ENAME, SALARY FROM EMPLOYEES1 WHERE SALARY>1000**;**

Result Grid | Filter Rows:

ENAME	SALARY
John	4500
Mike	4200
Sam	4800
Bob	4000
Sara	4100
David	4300
Emily	4400
Kate	4500
Olivia	4600
Sophia	4700

11.List names of clerk working in department 30.

SELECT ENAME FROM EMPLOYEES1 WHERE DEPARTMENT= (SELECT DEPTNAME FROM DEPARTMENT WHERE DEPTNO=30);

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
Sam

12. List names of employees who are either analyst or salesman.

SELECT ENAME FROM EMPLOYEES1 WHERE JOB = 'analyst' OR JOB = 'salesman';

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
John
Mike
Bob
Sara
David
Emily
Kate
Olivia
Sophia

13.List employee names and emp no who are manager.

SELECT EMPNO, ENAME FROM EMPLOYEES1 WHERE JOB='MANAGER';

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMPNO	ENAME
3	Sam

14. List the details of employees who have joined before end of September'22.

SELECT * FROM EMPLOYEES1 WHERE HIREDATE < '2022-09-30';

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMPNO	ENAME	JOB	MANAGER_NAME	HIREDATE	SALARY	COMMISSION	DEPARTMENT
1	John	salesman	Bob	2022-01-01	4500	10	sales
2	Mike	analyst	Sam	2022-01-15	4200	20	analyst
3	Sam	manager	NULL	2022-03-01	4800	30	manager
4	Bob	salesman	Sam	2022-04-01	4000	40	sales
5	Sara	analyst	Bob	2022-05-01	4100	4000	analyst
6	David	salesman	Sara	2022-06-01	4300	60	sales
7	Emily	analyst	David	2022-07-01	4400	70	analyst
8	Kate	salesman	Emily	2022-08-01	4500	80	sales
9	Olivia	analyst	Kate	2022-09-01	4600	90	analyst

15. List names of employees who have not managers.

SELECT ENAME FROM EMPLOYEES1 WHERE MANAGER_NAME IS NULL;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
Sam

16. List the names of employees whose employee number is 2,4,6.

SELECT ENAME FROM Employee
WHERE ENO IN (2, 4, 6);

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
Mike
Bob
David

17. List the names and salary of employees whose salary is between 4000 and 4500.

SELECT ENAME, SALARY FROM EMPLOYEES1 WHERE SALARY BETWEEN 4000 AND 4500;

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

ENAME	SALARY
John	4500
Mike	4200
Bob	4000
Sara	4100
David	4300
Emily	4400
Kate	4500

18. List the names of employees joined before 30 june'22 or after 31dec'22.

```
SELECT ENAME, HIREDATE FROM EMPLOYEES1 WHERE
HIREDATE < '2022-06-30'
or hIREDATE > '2022-12-31';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME	HIREDATE
John	2022-01-01
Mike	2022-01-15
Sam	2022-03-01
Bob	2022-04-01
Sara	2022-05-01
David	2022-06-01

DATE: 11/09/2024

PRACTICAL NO : 04

AIM : Using Relational, Logical Operators.

1. CREATE TABLE EMPLOYEES(
   EID INT,
   ENAME VARCHAR(50),
   DEPARTMENT VARCHAR(50),
   DEPTNO INT,
   SALARY INT,
   HIREDATE DATE);

2. INSERT INTO EMPLOYEES(EID , ENAME,
   DEPARTMENT,DEPTNO,SALARY,HIREDATE) VALUES
   (1,'SHRUTI','IT',101,50000,'2022-05-30'),
   (2,'SADAF','HR',102,40000,'2023-01-15'),
   (3,'NIDHI','PRODUCTION',103,30000,'2023-06-01'),
   (4,'ANIMESH','ACCOUNTANT',104,60000,'2021-05-20'),
   (5,'ZEESHAN','SALES',105,20000,'2023-06-21'),
   (6,'HIMANSHU','IT',101,60000,'2020-04-12'),
   (7,'ANKITA','HR',102,25000,'2022-08-30'),
   (8,'AMAN','SALES',105,35000,'2019-08-13'),
   (9,'SAMEER','PRODUCTION',103,45000,'2021-01-14'),
   (10,'ROHIT','ACCOUNTANT',104,25000,'2020-04-10'),
   (11,'LOVELY','IT',101,55000,'2020-05-30'),
   (12,'SAUMYA','SALES',105,56000,'2022-10-30'),
   (13,'ADITI','ACCOUNTANT',104,35000,'2023-02-28'),
   (14,'DIPTI','HR',102,52000,'2024-03-30'),
   (15,'HARSH','IT',101,42000,'2020-09-02');

3. **Display all records.**
   SELECT * FROM EMPLOYEES;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EID	ENAME	DEPARTMENT	DEPTNO	SALARY	HIREDATE
1	SHRUTI	IT	101	50000	2022-05-30
2	SADAF	HR	102	40000	2023-01-15
3	NIDHI	PRODUCTION	103	30000	2023-06-01
4	ANIMESH	ACCOUNTANT	104	60000	2021-05-20
5	ZEESHAN	SALES	105	20000	2023-06-21
6	HIMANSHU	IT	101	60000	2020-04-12
7	ANKITA	HR	102	25000	2022-08-30
8	AMAN	SALES	105	35000	2019-08-13
9	SAMEER	PRODUCTION	103	45000	2021-01-14
10	ROHIT	ACCOUNTANT	104	25000	2020-04-10
11	LOVELY	IT	101	55000	2020-05-30
12	SAUMYA	SALES	105	56000	2022-10-30
13	ADITI	ACCOUNTANT	104	35000	2023-02-28
14	DIPTI	HR	102	52000	2024-03-30
15	HARSH	IT	101	42000	2020-09-02

4. **Count total number of employee in the table.**
   SELECT COUNT(*) FROM EMPLOYEES;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNT(*)
15

5. **List max salary of employee whose dept is ‘Sales’.**
   SELECT MAX(SALARY) FROM EMPLOYEES WHERE DEPARTMENT='SALES';

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MAX(SALARY)
56000

6. **List min salary and name of employee in all of the department.**
   SELECT ENAME, SALARY FROM EMPLOYEES
   WHERE SALARY = (SELECT MIN(SALARY) FROM EMPLOYEES);

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME	SALARY
ZEESHAN	20000

7. **List avg sal and number of employee working in dept no – 101.**
   SELECT AVG(SALARY), COUNT(*) FROM EMPLOYEES WHERE DEPTNO = 101;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVG(SALARY)	COUNT(*)
51750.0000	4

8. **Total salary of all employees.**
   SELECT SUM(SALARY) AS TOTAL_SALARY FROM EMPLOYEES;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TOTAL_SALARY
630000

9. **List ename, salary ,dept and deptno in ascending order of deptno & descending order of salary.**

SELECT ENAME, SALARY, DEPARTMENT,DEPTNO
FROM EMPLOYEES
ORDER BY DEPTNO,SALARY DESC;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME	SALARY	DEPARTMENT	DEPTNO
HIMANSHU	60000	IT	101
LOVELY	55000	IT	101
SHRUTI	50000	IT	101
HARSH	42000	IT	101
DIPTI	52000	HR	102
SADAF	40000	HR	102
ANKITA	25000	HR	102
SAMEER	45000	PRODUCTION	103
NIDHI	30000	PRODUCTION	103
ANIMESH	60000	ACCOUNTANT	104
ADITI	35000	ACCOUNTANT	104
ROHIT	25000	ACCOUNTANT	104
SAUMYA	56000	SALES	105
AMAN	35000	SALES	105
ZEESHAN	20000	SALES	105

10. **List the name of employees who are working in organization for more than 2 months.**

SELECT ENAME FROM EMPLOYEES WHERE
DATEDIFF(NOW(), HIREDATE) > 60;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
SHRUTI
SADAF
NIDHI
ANIMESH
ZEESHAN
HIMANSHU
ANKITA
AMAN
SAMEER
ROHIT
LOVELY
SAUMYA
ADITI
DIPTI
HARSH

11.List ename & hire date in descending order of hire date.

SELECT ENAME, HIREDATE FROM
EMPLOYEES ORDER BY HIREDATE DESC;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME	HIREDATE
DIPTI	2024-03-30
ZEESHAN	2023-06-21
NIDHI	2023-06-01
ADITI	2023-02-28
SADAF	2023-01-15
SAUMYA	2022-10-30
ANKITA	2022-08-30
SHRUTI	2022-05-30
ANIMESH	2021-05-20
SAMEER	2021-01-14
HARSH	2020-09-02
LOVELY	2020-05-30
HIMANSHU	2020-04-12
ROHIT	2020-04-10
AMAN	2019-08-13

DATE: 23/10/2024

Practical No: 05

Aim: Based on null, distinct, like, derived attribute, order by clause.

1. **List the different jobs available in the employee table.**
   SELECT DISTINCT JOB FROM EMPLOYEES1;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

JOB
SALESMAN
ANALYST
MANAGER

2. **List the employee names that are not eligible for commission.**
   SELECT ENAME FROM EMPLOYEES1 WHERE COMMISSION IS NULL;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
BOB
EMILY

3. **List the name of employees and their designation who does not report to anyone.**
   SELECT ENAME,JOB FROM EMPLOYEES1 WHERE MANAGER_NAME IS NULL;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME	JOB
SAMS	MANAGER
DAVID	SALESMAN
OLIVIA	ANALYST

4. **List the employee who are not assign to any department.**
   SELECT ENAME FROM EMPLOYEES1 WHERE DEPARTMENT IS NULL;

Result Grid | Filter Rows: | Export: | Wrap

ENAME
MIKE
DAVID

5. **List the employees name and their designation that is eligible for commission.**
   SELECT ENAME, JOB FROM EMPLOYEES1 WHERE COMMISSION IS NOT NULL;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME	JOB
JOHN	SALESMAN
MIKE	ANALYST
SAMS	MANAGER
SARAS	ANALYST
DAVID	SALESMAN
KATE	SALESMAN
OLIVIA	ANALYST
SOPHAIA	SALESMAN

6. **List the details where salaries is greater than 2000 and commission is null.**
   SELECT * FROM EMPLOYEES1 WHERE SALARY>2000 AND COMMISSION IS NULL;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMPNO	ENAME	JOB	MANAGER_NAME	HIREDATE	SALARY	PF	HRA	DA	GROSS_SALARY	COMMISSION	DEPARTMENT
4	BOB	SALESMAN	SAM	2022-04-01	4000	400	2000	1200	7600	NULL	SALES
7	EMILY	ANALYST	DAVID	2022-07-01	4400	440	2200	1320	8360	NULL	ANALYST

7. **List the name of employees whose name is start with ‘ S ’.**
   SELECT ENAME FROM EMPLOYEES1 WHERE ENAME LIKE 'S%';

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
SAMS
SARAS
SOPHAIA

8. **List the employees name whose name end with ‘ S ‘.**
   SELECT ENAME FROM EMPLOYEES1 WHERE ENAME LIKE '%S';

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
SAMS
SARAS

9. **List the employees whose name has 5 characters.**
   SELECT ENAME FROM EMPLOYEES1 WHERE LENGTH(ENAME)=5;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
SARAS
DAVID
EMILY

10. List the name of the employees who is having ‘ I ’ as second character.

SELECT ENAME FROM EMPLOYEES1 WHERE ENAME LIKE '_I%';

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
MIKE

11. List the name of employee who are 2 years old in organization.

SELECT ENAME FROM EMPLOYEES1 WHERE YEAR(CURDATE()) - YEAR(HIREDATE) = 2;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME
JOHN
MIKE
SAMS
BOB
SARAS
DAVID
EMILY
KATE
OLIVIA
SOPHAIA

12. List the name of employee, salary, and pf amount of all employees where PF is 10% of salary.

SELECT ENAME, SALARY, PF FROM EMPLOYEES1;

Result Grid | Filter Rows: | Export: | Wrap Cell Conter

ENAME	SALARY	PF
JOHN	4500	450
MIKE	4200	420
SAMS	4800	480
BOB	4000	400
SARAS	4100	410
DAVID	4300	430
EMILY	4400	440
KATE	4500	450
OLIVIA	4600	460
SOPHAIA	4700	470

13. List the employees name, number and salary in ascending order by salary.

SELECT EMPNO, ENAME, SALARY FROM EMPLOYEES1 ORDER BY SALARY;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMPNO	ENAME	SALARY
4	BOB	4000
5	SARAS	4100
2	MIKE	4200
6	DAVID	4300
7	EMILY	4400
1	JOHN	4500
8	KATE	4500
9	OLIVIA	4600
10	SOPHAIA	4700
3	SAMS	4800

14. List the name of employees and hire date in descending order of hire date.

SELECT ENAME, HIREDATE FROM EMPLOYEES1 ORDER BY HIREDATE DESC;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME	HIREDATE
SOPHAIA	2022-11-03
OLIVIA	2022-11-01
KATE	2022-08-01
EMILY	2022-07-01
DAVID	2022-06-01
SARAS	2022-05-01
BOB	2022-04-01
SAMS	2022-03-01
MIKE	2022-01-15
JOHN	2022-01-01

15. List the employees name, salary, PF(10%), HRA(50%) , DA(30%) and gross salary(PF + HRA + DA + salary) order the result on the basis of gross.

SELECT * FROM EMPLOYEES1 ORDER BY SALARY;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMPNO	ENAME	JOB	MANAGER_NAME	HIREDATE	SALARY	PF	HRA	DA	GROSS_SALARY	COMMISSION	DEPARTMENT
4	BOB	SALESMAN	SAM	2022-04-01	4000	400	2000	1200	7600	NULL	SALES
5	SARAS	ANALYST	BOB	2022-05-01	4100	410	2050	1230	7790	50	ANALYST
2	MIKE	ANALYST	SAM	2022-01-15	4200	420	2100	1260	7980	20	NULL
6	DAVID	SALESMAN	NULL	2022-06-01	4300	430	2150	1290	8170	60	NULL
7	EMILY	ANALYST	DAVID	2022-07-01	4400	440	2200	1320	8360	NULL	ANALYST
1	JOHN	SALESMAN	BOB	2022-01-01	4500	450	2250	1350	8550	10	SALES
8	KATE	SALESMAN	EMILY	2022-08-01	4500	450	2250	1350	8550	80	SALES
9	OLIVIA	ANALYST	NULL	2022-11-01	4600	460	2300	1380	8740	90	ANALYST
10	SOPHAIA	SALESMAN	OLIVIA	2022-11-03	4700	470	2350	1410	8930	100	SALES
3	SAMS	MANAGER	NULL	2022-03-01	4800	480	2400	1440	9120	30	MANAGER

DATE: 13/11/2024

Practical No: 06

AIM – Based on null distinct, null, like, derived attribute, order by clause.

1. **List the department number and number of employees in each department.**
   SELECT DEPARTMENT, DEPTNO, COUNT(DEPARTMENT)
   FROM EMPLOYEE
   GROUP BY DEPTNO, DEPARTMENT;

Result Grid | Filter Rows: | Export:

DEPARTMENT	DEPTNO	COUNT(DEPARTMENT)
HR	10	3
IT	20	4
Finance	30	3
Design	40	2
Operations	50	2
Customer Service	60	1

2. **List the department number, total salary payable to each department.**
   SELECT DEPTNO, SUM(GROSS_SALARY) AS TOTAL_SALARY
   FROM EMPLOYEE
   GROUP BY DEPTNO;

Result Grid | Filter Rows:

DEPTNO	TOTAL_SALARY
10	240000
20	237000
30	201000
40	107000
50	201000
60	48000

3. **List the jobs & no of emp in each job result must be in descending order of no of employees.**

```
SELECT JOB, COUNT(ENO) AS NO_OF_EMPLOYEES
FROM EMPLOYEE
GROUP BY JOB
ORDER BY NO_OF_EMPLOYEES DESC;
```

Result Grid | Filter Rows:

JOB	NO_OF_EMPLOYEES
Manager	4
Developer	3
Analyst	2
Designer	1
HR Specialist	1
Accountant	1
Assistant	1
Support Staff	1
Technician	1

4. **List the sum of all salary, maximum salary & average salary of employees job wise.**

```
SELECT JOB,
SUM(SALARY) AS TOTAL_SALARY,
MAX(SALARY) AS MAX_SALARY,
AVG(SALARY) AS AVG_SALARY
FROM EMPLOYEE
GROUP BY JOB;
```

Result Grid | Filter Rows: | Export: | Wrap C

JOB	TOTAL_SALARY	MAX_SALARY	AVG_SALARY
Manager	277000	75000	69250.0000
Developer	139000	48000	46333.3333
Analyst	108000	55000	54000.0000
Designer	40000	40000	40000.0000
HR Specialist	42000	42000	42000.0000
Accountant	47000	47000	47000.0000
Assistant	35000	35000	35000.0000
Support Staff	32000	32000	32000.0000
Technician	42000	42000	42000.0000

5. List the average salary from each job excluding manager.

```
SELECT JOB, AVG(SALARY) AS AVG_SALARY
FROM EMPLOYEE
WHERE JOB != 'Manager'
GROUP BY JOB;
```

Result Grid | Filter Rows:

JOB	AVG_SALARY
Developer	46333.3333
Analyst	54000.0000
Designer	40000.0000
HR Specialist	42000.0000
Accountant	47000.0000
Assistant	35000.0000
Support Staff	32000.0000
Technician	42000.0000

6. **List the total salary max & min salary, avg salary of employees jobwise for department number 20.**

```
SELECT JOB,
SUM(SALARY) AS TOTAL_SALARY,
MAX(SALARY) AS MAX_SALARY,
MIN(SALARY) AS MIN_SALARY,
AVG(SALARY) AS AVG_SALARY
FROM EMPLOYEE
WHERE DEPTNO = 20
GROUP BY JOB;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

JOB	TOTAL_SALARY	MAX_SALARY	MIN_SALARY	AVG_SALARY
Developer	139000	48000	45000	46333.3333
Technician	42000	42000	42000	42000.0000

7. **List the average salary of all the departments employing more than 4 people.**

```
SELECT DEPTNO,
AVG(SALARY) AS AVG_SALARY
FROM EMPLOYEE
GROUP BY DEPTNO
HAVING COUNT(ENO) > 3;
```

Result Grid | Filter Rows:

DEPTNO	AVG_SALARY
20	45250.0000

;

8. **List average monthly salary for each job type within department number outer group by department number in a group by job.**

```
SELECT DEPTNO, JOB,
AVG(SALARY / 12) AS AVG_MONTHLY_SALARY
FROM EMPLOYEE
GROUP BY DEPTNO, JOB
ORDER BY DEPTNO, JOB;
```

Result Grid | Filter Rows: | Export: | W

DEPTNO	JOB	AVG_MONTHLY_SALARY
10	HR Specialist	3500.00000000
10	Manager	5500.00000000
20	Developer	3861.11110000
20	Technician	3500.00000000
30	Accountant	3916.66670000
30	Analyst	4500.00000000
40	Assistant	2916.66670000
40	Designer	3333.33330000
50	Manager	6041.66665000
60	Support Staff	2666.66670000

9. **List job of employees where salary >=2500.**

```
SELECT DISTINCT JOB
FROM EMPLOYEE
WHERE SALARY >= 2500;
```

Result Grid | Filter Rows:

JOB
Manager
Developer
Analyst
Designer
HR Specialist
Accountant
Assistant
Support Staff
Technician

Date: 13/11/2024

Practical No: 07

1. **Create a branch table having columns branch name, branch city, assets.**

```
CREATE TABLE BRANCH(
BRANCH_NAME varchar(50) ,
BRANCH_CITY varchar(50) ,
ASSETS int);
```

Result Grid | Filter Rows: | Export:

BRANCH_NAME	BRANCH_CITY	ASSETS
Mumbai Central	Mumbai	50000000
Delhi Connaught	Delhi	40000000
Bangalore Whitefield	Bangalore	30000000
Chennai Mount Road	Chennai	20000000
Hyderabad Banjara Hills	Hyderabad	15000000
Kolkata Park Street	Kolkata	25000000
Pune Koregaon Park	Pune	18000000
Ahmedabad SG Highway	Ahmedabad	22000000
Lucknow Hazratganj	Lucknow	19000000
Jaipur MI Road	Jaipur	17000000

2. **Create a customer table having columns customer name, street, city, assets.**

```
CREATE TABLE CUSTOMER(
CUSTOMER_NAME varchar(50) ,
CUSTOMER_STREET varchar(50),
CUSTOMER_CITY varchar(50) ,
ASSETS int);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

CUSTOMER_NAME	CUSTOMER_STREET	CUSTOMER_CITY	ASSETS
Amit Sharma	15/2, MG Road	Mumbai	3000000
Neha Gupta	8/4, Main Delhi	Delhi	5000000
Rahul Kumar	22, Whitefield Road	Bangalore	3500000
Priya Rao	3/1, Anna Nagar	Chennai	4500000
Rajesh Reddy	2, Banjara Hills	Hyderabad	4000000
Sita Mishra	45, Park Street	Kolkata	2000000
Vikram Deshpande	12, Koregaon Park	Pune	2500000
Pooja Patel	60, SG Highway	Ahmedabad	5500000
Vivek Srivastava	13/2, Hazratganj	Lucknow	3000000
Anjali Singh	23, MI Road	Jaipur	3200000

3. **Create a table account having columns account no, branch name, balance.**

```
CREATE TABLE ACCOUNT(
ACCOUNT_NUMBER int ,
BRANCH_NAME varchar(50) ,
BALANCE int);
```

Result Grid | Filter Rows: | Export: | W

ACCOUNT_NUMBER	BRANCH_NAME	BALANCE
1001	Mumbai Central	1000000
1002	Delhi Connaught	1500000
1003	Bangalore Whitefield	2000000
1004	Chennai Mount Road	2500000
1005	Hyderabad Banjara Hills	3000000
1006	Kolkata Park Street	1200000
1007	Pune Koregaon Park	2200000
1008	Ahmedabad SG Highway	1700000
1009	Lucknow Hazratganj	800000
1010	Jaipur MI Road	1300000

4. **Create a table depositor having columns depositor, customer name, account name.**

```
CREATE TABLE DEPOSITOR(
CUSTOMER_NAME varchar(50) ,
ACCOUNT_NAME varchar(50));
```

Result Grid | Filter Rows:

CUSTOMER_NAME	ACCOUNT_NAME
Amit Sharma	1001
Neha Gupta	1002
Rahul Kumar	1003
Priya Rao	1004
Rajesh Reddy	1005
Sita Mishra	1006
Vikram Deshpande	1007
Pooja Patel	1008
Vivek Srivastava	1009
Anjali Singh	1010

5. **Create a table loan having columns loan no, branch name, amount.**

```
CREATE TABLE LOAN(
LOAN_NUMBER int ,
BRANCH_NAME varchar(50) ,
AMOUNT int);
```

Result Grid | Filter Rows: | Export:

LOAN_NUMBER	BRANCH_NAME	AMOUNT
2001	Mumbai Central	5000000
2002	Delhi Connaught	4000000
2003	Bangalore Whitefield	3500000
2004	Chennai Mount Road	6000000
2005	Hyderabad Banjara Hills	2500000
2006	Kolkata Park Street	1000000
2007	Pune Koregaon Park	3000000
2008	Ahmedabad SG Highway	4500000
2009	Lucknow Hazratganj	2000000
2010	Jaipur MI Road	3500000

6. **Create a table borrower having customer name, loan number.**

```
CREATE TABLE BORROWER(
CUSTOMER_NAME varchar(50) ,
LOAN_NUMBER int);
```

Result Grid | Filter Rows:

CUSTOMER_NAME	LOAN_NUMBER
Amit Sharma	2001
Neha Gupta	2002
Rahul Kumar	2003
Priya Rao	2004
Rajesh Reddy	2005
Sita Mishra	2006
Vikram Deshpande	2007
Pooja Patel	2008
Vivek Srivastava	2009
Anjali Singh	2010

DATE: 13/11/2024

Practical No: 08

1. **Find the name of all branches in loan relation.**

```
SELECT BRANCH_NAME
FROM LOAN;
```

Result Grid | Filter Rows:

BRANCH_NAME
Mumbai Central
Delhi Connaught
Bangalore Whitefield
Chennai Mount Road
Hyderabad Banjara Hills
Kolkata Park Street
Pune Koregaon Park
Ahmedabad SG Highway
Lucknow Hazratganj
Jaipur MI Road

2. **Find the names of all branches in the loan relation and remove duplicates.**

```
SELECT DISTINCT BRANCH_NAME
FROM LOAN;
```

Result Grid | Filter Rows:

BRANCH_NAME
Mumbai Central
Delhi Connaught
Bangalore Whitefield
Chennai Mount Road
Hyderabad Banjara Hills
Kolkata Park Street
Pune Koregaon Park
Ahmedabad SG Highway
Lucknow Hazratganj
Jaipur MI Road

3. **Find all loan number for loans made by Delhi Connaught branch with loan amount >120000.**

```
SELECT LOAN_NUMBER
FROM LOAN
WHERE BRANCH_NAME = 'Delhi Connaught' AND AMOUNT > 120000;
```

Result Grid | Filter Rows:

LOAN_NUMBER
2002

4. **Find the cartesian product borrower * loan.**

```
SELECT *
FROM BORROWER
CROSS JOIN LOAN;
```

CUSTOMER_NAME	LOAN_NUMBER	LOAN_NUMBER	BRANCH_NAME	AMOUNT
Anjali Singh	2010	2001	Mumbai Central	5000000
Vivek Srivastava	2009	2001	Mumbai Central	5000000
Pooja Patel	2008	2001	Mumbai Central	5000000
Vikram Deshpande	2007	2001	Mumbai Central	5000000
Sita Mishra	2006	2001	Mumbai Central	5000000
Rajesh Reddy	2005	2001	Mumbai Central	5000000
Priya Rao	2004	2001	Mumbai Central	5000000
Rahul Kumar	2003	2001	Mumbai Central	5000000
Neha Gupta	2002	2001	Mumbai Central	5000000
Amit Sharma	2001	2001	Mumbai Central	5000000
Anjali Singh	2010	2002	Delhi Connaught	4000000
Vivek Srivastava	2009	2002	Delhi Connaught	4000000
Pooja Patel	2008	2002	Delhi Connaught	4000000
Vikram Deshpande	2007	2002	Delhi Connaught	4000000
Sita Mishra	2006	2002	Delhi Connaught	4000000
Rajesh Reddy	2005	2002	Delhi Connaught	4000000
Priya Rao	2004	2002	Delhi Connaught	4000000
Rahul Kumar	2003	2002	Delhi Connaught	4000000
Neha Gupta	2002	2002	Delhi Connaught	4000000
Amit Sharma	2001	2002	Delhi Connaught	4000000

CUSTOMER_NAME	LOAN_NUMBER	LOAN_NUMBER	BRANCH_NAME	AMOUNT
Anjali Singh	2010	2005	Hyderabad Ba...	2500000
Vivek Srivastava	2009	2005	Hyderabad Ba...	2500000
Pooja Patel	2008	2005	Hyderabad Ba...	2500000
Vikram Deshpande	2007	2005	Hyderabad Ba...	2500000
Sita Mishra	2006	2005	Hyderabad Ba...	2500000
Rajesh Reddy	2005	2005	Hyderabad Ba...	2500000
Priya Rao	2004	2005	Hyderabad Ba...	2500000
Rahul Kumar	2003	2005	Hyderabad Ba...	2500000
Neha Gupta	2002	2005	Hyderabad Ba...	2500000
Amit Sharma	2001	2005	Hyderabad Ba...	2500000
Anjali Singh	2010	2006	Kolkata Park St...	1000000
Vivek Srivastava	2009	2006	Kolkata Park St...	1000000
Pooja Patel	2008	2006	Kolkata Park St...	1000000
Vikram Deshpande	2007	2006	Kolkata Park St...	1000000
Sita Mishra	2006	2006	Kolkata Park St...	1000000
Rajesh Reddy	2005	2006	Kolkata Park St...	1000000
Priya Rao	2004	2006	Kolkata Park St...	1000000
Rahul Kumar	2003	2006	Kolkata Park St...	1000000
Neha Gupta	2002	2006	Kolkata Park St...	1000000
Amit Sharma	2001	2006	Kolkata Park St...	1000000

CUSTOMER_NAME	LOAN_NUMBER	LOAN_NUMBER	BRANCH_NAME	AMOUNT
Anjali Singh	2010	2007	Pune Koregao...	3000000
Vivek Srivastava	2009	2007	Pune Koregao...	3000000
Pooja Patel	2008	2007	Pune Koregao...	3000000
Vikram Deshpande	2007	2007	Pune Koregao...	3000000
Sita Mishra	2006	2007	Pune Koregao...	3000000
Rajesh Reddy	2005	2007	Pune Koregao...	3000000
Priya Rao	2004	2007	Pune Koregao...	3000000
Rahul Kumar	2003	2007	Pune Koregao...	3000000
Neha Gupta	2002	2007	Pune Koregao...	3000000
Amit Sharma	2001	2007	Pune Koregao...	3000000
Anjali Singh	2010	2008	Ahmedabad S...	4500000
Vivek Srivastava	2009	2008	Ahmedabad S...	4500000
Pooja Patel	2008	2008	Ahmedabad S...	4500000
Vikram Deshpande	2007	2008	Ahmedabad S...	4500000
Sita Mishra	2006	2008	Ahmedabad S...	4500000
Rajesh Reddy	2005	2008	Ahmedabad S...	4500000
Priya Rao	2004	2008	Ahmedabad S...	4500000
Rahul Kumar	2003	2008	Ahmedabad S...	4500000
Neha Gupta	2002	2008	Ahmedabad S...	4500000
Amit Sharma	2001	2008	Ahmedabad S...	4500000

CUSTOMER_NAME	LOAN_NUMBER	LOAN_NUMBER	BRANCH_NAME	AMOUNT
Anjali Singh	2010	2009	Lucknow Hazra...	2000000
Vivek Srivastava	2009	2009	Lucknow Hazra...	2000000
Pooja Patel	2008	2009	Lucknow Hazra...	2000000
Vikram Deshpande	2007	2009	Lucknow Hazra...	2000000
Sita Mishra	2006	2009	Lucknow Hazra...	2000000
Rajesh Reddy	2005	2009	Lucknow Hazra...	2000000
Priya Rao	2004	2009	Lucknow Hazra...	2000000
Rahul Kumar	2003	2009	Lucknow Hazra...	2000000
Neha Gupta	2002	2009	Lucknow Hazra...	2000000
Amit Sharma	2001	2009	Lucknow Hazra...	2000000
Anjali Singh	2010	2010	Jaipur MI Road	3500000
Vivek Srivastava	2009	2010	Jaipur MI Road	3500000
Pooja Patel	2008	2010	Jaipur MI Road	3500000
Vikram Deshpande	2007	2010	Jaipur MI Road	3500000
Sita Mishra	2006	2010	Jaipur MI Road	3500000
Rajesh Reddy	2005	2010	Jaipur MI Road	3500000
Priya Rao	2004	2010	Jaipur MI Road	3500000
Rahul Kumar	2003	2010	Jaipur MI Road	3500000
Neha Gupta	2002	2010	Jaipur MI Road	3500000
Amit Sharma	2001	2010	Jaipur MI Road	3500000

5. **Find the name, loan no and loan amount of all customers having a loan at Delhi branch.**

```
SELECT BORROWER.CUSTOMER_NAME, LOAN.LOAN_NUMBER,
LOAN.AMOUNT
FROM BORROWER
JOIN LOAN ON BORROWER.LOAN_NUMBER =
LOAN.LOAN_NUMBER
WHERE LOAN.BRANCH_NAME = 'Delhi Connaught';
```

Result Grid | Filter Rows: | Expor

CUSTOMER_NAME	LOAN_NUMBER	AMOUNT
Neha Gupta	2002	4000000

6. **Find the name, loan no, loan amount of all customers and rename the column name loan no as loan id.**

```
SELECT BORROWER.CUSTOMER_NAME,
LOAN.LOAN_NUMBER AS "LOAN ID",
LOAN.AMOUNT
FROM BORROWER
JOIN LOAN ON BORROWER.LOAN_NUMBER =
LOAN.LOAN_NUMBER;
```

	CUSTOMER_NAME	LOAN ID	AMOUNT
▶	Amit Sharma	2001	5000000
	Neha Gupta	2002	4000000
	Rahul Kumar	2003	3500000
	Priya Rao	2004	6000000
	Rajesh Reddy	2005	2500000
	Sita Mishra	2006	1000000
	Vikram Deshpande	2007	3000000
	Pooja Patel	2008	4500000
	Vivek Srivastava	2009	2000000
	Anjali Singh	2010	3500000

7. **Find the customer names and their loan number for all the customers having loan at some branch.**

```
SELECT CUSTOMER_NAME, LOAN_NUMBER
FROM BORROWER
NATURAL JOIN LOAN;
```

Result Grid | Filter Rows:

CUSTOMER_NAME	LOAN_NUMBER
Amit Sharma	2001
Neha Gupta	2002
Rahul Kumar	2003
Priya Rao	2004
Rajesh Reddy	2005
Sita Mishra	2006
Vikram Deshpande	2007
Pooja Patel	2008
Vivek Srivastava	2009
Anjali Singh	2010

8. **Find the names of all branches that have greater assests than some branch located in Jaipur.**

```
SELECT BRANCH_NAME
FROM BRANCH
WHERE ASSETS > (
    SELECT MAX(ASSETS)
    FROM BRANCH
    WHERE BRANCH_CITY = 'Jaipur'
);
```

BRANCH_NAME
Mumbai Central
Delhi Connaught
Bangalore Whitefield
Chennai Mount Road
Kolkata Park Street
Pune Koregaon Park
Ahmedabad SG Highway
Lucknow Hazratganj

9. **Find the names of all customers whose street includes the string "Main".**

```
SELECT CUSTOMER_NAME
FROM CUSTOMER
WHERE CUSTOMER_STREET LIKE '%Main%';
```

Result Grid | Filter Rows:

CUSTOMER_NAME
Neha Gupta

10. **List the names of all the customer having loan in Delhi branch.**

```
SELECT BORROWER.CUSTOMER_NAME
FROM BORROWER
JOIN LOAN ON BORROWER.LOAN_NUMBER =
LOAN.LOAN_NUMBER
WHERE LOAN.BRANCH_NAME = 'Delhi main branch';
```

Result Grid | Filter Rows:

	CUSTOMER_NAME
▶	Neha Gupta

11.Find the all customers who have a loan and account or both.

```
SELECT CUSTOMER_NAME
FROM BORROWER
UNION
SELECT CUSTOMER_NAME
FROM DEPOSITOR;
```

Result Grid | Filter Rows:

	CUSTOMER_NAME
▶	Amit Sharma
	Neha Gupta
	Rahul Kumar
	Priya Rao
	Rajesh Reddy
	Sita Mishra
	Vikram Deshpande
	Pooja Patel
	Vivek Srivastava
	Anjali Singh

12. Find all customers who have both loan and account.

```
SELECT DISTINCT BORROWER.CUSTOMER_NAME
FROM BORROWER
JOIN DEPOSITOR ON BORROWER.CUSTOMER_NAME =
DEPOSITOR.CUSTOMER_NAME;
```

Result Grid | Filter Rows: | Ex

CUSTOMER_NAME
Amit Sharma
Neha Gupta
Rahul Kumar
Priya Rao
Rajesh Reddy
Sita Mishra
Vikram Deshpande
Pooja Patel
Vivek Srivastava
Anjali Singh

13. Find all customers who have an account but no loan.

```
SELECT DISTINCT DEPOSITOR.CUSTOMER_NAME
FROM DEPOSITOR
LEFT JOIN BORROWER ON DEPOSITOR.CUSTOMER_NAME =
BORROWER.CUSTOMER_NAME
WHERE BORROWER.CUSTOMER_NAME IS NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

CUSTOMER_NAME

14.Find the average account balance at Delhi branch.

```
SELECT AVG(BALANCE) AS Average_Balance
FROM ACCOUNT
WHERE BRANCH_NAME = 'Delhi Connaught';
```

Result Grid | Filter Rows:

	Average_Balance
▶	1500000.0000

15.Find the number of tuple in the customer relation.

```
SELECT COUNT(*) AS NumberOfTuples
FROM CUSTOMER;
```

Result Grid | Filter Rows:

	NumberOfTuples
▶	10

16.Find the number of depositors in the bank.

```
SELECT COUNT(*) AS NumberOfDepositors
FROM DEPOSITOR;
```

Result Grid | Filter Rows:

	NumberOfDepositors
▶	10

17. Find the number of depositors in each branch.

```
SELECT A.BRANCH_NAME, COUNT(B.CUSTOMER_NAME) AS
NumberOfDepositors
FROM DEPOSITOR B
JOIN ACCOUNT A ON B.ACCOUNT_NAME =
A.ACCOUNT_NUMBER
GROUP BY A.BRANCH_NAME;
```

Result Grid | Filter Rows: | Export:

BRANCH_NAME	NumberOfDepositors
Mumbai Central	1
Delhi Connaught	1
Bangalore Whitefield	1
Chennai Mount Road	1
Hyderabad Banjara Hills	1
Kolkata Park Street	1
Pune Koregaon Park	1
Ahmedabad SG Highway	1
Lucknow Hazratganj	1
Jaipur MI Road	1

18. Find the names of all branches where the average account balance is more than 120000.

```
SELECT BRANCH_NAME
FROM ACCOUNT
GROUP BY BRANCH_NAME
HAVING AVG(BALANCE) > 120000;
```

Result Grid | Filter Rows:

BRANCH_NAME
Mumbai Central
Delhi Connaught
Bangalore Whitefield
Chennai Mount Road
Hyderabad Banjara Hills
Kolkata Park Street
Pune Koregaon Park
Ahmedabad SG Highway
Lucknow Hazratganj
Jaipur MI Road

DATE: 12/11/2024

Case Study on Oracle Database XE (Express Edition)

Introduction to Oracle XE

Oracle Database XE (Express Edition) is a free, lightweight, and limited-functionality version of Oracle's enterprise database offering. It is designed for learning, prototyping, and small applications. Unlike Oracle's other editions (Standard, Enterprise), Oracle XE is a small-scale version, with certain limitations like database size, memory usage, and CPU usage. Despite these limitations, it offers a real Oracle Database experience, making it ideal for small developers, startups, and educational purposes.

Features of Oracle XE

1. **Free to Use**: Oracle XE is free for use in development, testing, and deployment (subject to usage limits).
2. **Ease of Use**: The database provides an intuitive web-based interface for managing database objects and running SQL queries.
3. **Small Footprint**: The database is designed to run efficiently on smaller hardware configurations.
4. **SQL, PL/SQL, and ADO.NET Support**: It supports various languages and APIs, making it highly versatile for different applications.
5. **Integrated Oracle Application Express (APEX)**: Oracle XE includes APEX, a web-based development framework, for building and deploying web applications with minimal coding.

Case Study Overview

Company: A small startup building a web application for inventory management.
Objective: The goal was to create an application that tracks inventory, manages orders, and generates reports. The team needed a database that was easy to set up, cost-effective, and scalable to meet future growth. **Challenges**

1. **Cost Constraints:** The startup had limited resources and couldn't afford a full-fledged Oracle Enterprise Edition**.**
2. **Database Performance:** The application needed to scale as the user base grew, which required a solid, efficient database backend.
3. **Time-to-Market:** The application needed to be developed quickly to meet customer demand and begin generating revenue.

4. **Development Simplicity:** The development team had a limited database experience and needed an easy-to-use solution**.**

Solution: Oracle XE

The company decided to use Oracle Database XE for several reasons:

1. **No Licensing Cost**: Oracle XE is free to use, which was a critical factor for the startup's budget.
2. **Scalability**: Even though Oracle XE has limitations in terms of database size (up to 12GB) and CPU usage, the team anticipated that their application wouldn't exceed these limits in the near future.
3. **Rapid Setup and Development**: Oracle XE's simple installation process, along with the included tools like Oracle SQL Developer and APEX, allowed the team to quickly build, test, and deploy the application.
4. **Performance**: Oracle XE is built on the same engine as Oracle's enterprise editions, providing a solid, scalable database solution for growing applications.

Implementation Process

1. **Database Design:**
   - The development team designed a relational database model using tables for products, inventory, customers, orders, and transactions.
   - They utilized foreign key constraints, indexes, and views to ensure efficient querying and data integrity.
2. **Application Development:**
   - The team used Oracle APEX to develop a web-based user interface for managing the inventory.
   - They implemented PL/SQL for business logic, such as order processing and inventory updates.
3. **Testing:**
   - The team conducted stress testing and performance analysis to ensure the application could handle the required workloads within the limitations of Oracle XE.
   - They set up automated backup processes and ensured that data integrity was maintained even under heavy use.
4. **Deployment:**
   - The application was deployed in a small cloud environment using Oracle Cloud Free Tier, which included Oracle XE as part of their infrastructure offerings.
   - The team also used Oracle APEX's cloud hosting features for easy deployment and scalability.

Results and Outcomes

1. **Cost Savings:** The use of Oracle XE eliminated licensing fees, which allowed the startup to reinvest their resources into other parts of the business, such as marketing and customer acquisition.
2. **Ease of Development:** The development team was able to build and launch the application quickly due to Oracle XE's intuitive tools and ease of use.
3. **Scalability:** The application was able to handle the startup's initial load, and they were able to seamlessly scale the database by migrating to an Oracle Standard Edition when their database size approached the 12GB limit.
4. **Business Continuity:** Oracle XE's reliability and security features helped ensure that the application could operate smoothly, with minimal downtime, even in the early stages of the business.

Challenges Encountered

1. **Database Size Limitation:** As the company's product offerings expanded and the number of orders increased, the team had to closely monitor the size of their database to ensure they didn't exceed the 12GB limit. This required careful planning for data archiving and purging.
2. **Resource Constraints:** Oracle XE is limited to using a single CPU core, which, while sufficient for a small-scale application, created performance bottlenecks as the load increased. The team had to optimize queries and database structures to mitigate this.
3. **Limited Enterprise Features:** While Oracle XE provided a solid foundation, some advanced features such as partitioning, advanced security, and real-time performance tuning were not available. As a result, the company had to consider migrating to Oracle's Enterprise Edition as they scaled.

Lessons Learned

1. **Proper Scaling Planning:** Even with the initial low-cost and low-resource requirements, the team learned the importance of planning for future scaling early in the project to avoid performance bottlenecks.
2. **Cloud Hosting Synergy:** Using Oracle Cloud's free tier alongside Oracle XE provided a perfect combination of cost savings and performance, enabling the company to focus on development rather than managing infrastructure.
3. **Monitoring and Optimization:** Regular performance monitoring, query optimization, and database housekeeping (like archiving old data) were crucial to maintaining optimal performance within Oracle XE's constraints.

Conclusion

Oracle Database XE proved to be an excellent choice for the startup, meeting its needs for a low-cost, reliable, and scalable database solution. While Oracle XE's limitations in terms of CPU usage and database size posed challenges, the startup was able to overcome them through careful planning and optimization. The free and easy-to-use nature of Oracle XE enabled rapid development and deployment, helping the company bring its product to market quickly without incurring significant costs. As the business grows, the team plans to migrate to a more robust Oracle edition, but for now, Oracle XE continues to serve as a valuable asset for their operations.