**PRACTICAL - 1**

**AIM:**

**Evaluation of Database (File System, DBMS, RDBMS, DDBMS).**

**File System:­­­­­­­­­­­**

* A File Management system is a DBMS that allows acceSs to single files or tables at a time. In a File System, data is directly stored in set of files. It contains flat files that have no relation to other files (when only one table is stored in single file, then this file is known as flat file).
* It helps you to organizes the data and allows easy retrieval of files when they are required. It mostly consists of different types of files like mp3, mp4, txt, doc, etc. that are grouped into directories.
* A file system enables you to handle the way of reading and writing data to the storage medium. It is directly installed into the computer with the Operating systems such as Windows and

**Features of a File system:**

Here are important elements of the file system:

* It helps you to store data in a group of files.
* Files data are dependent on each other.
* C/C++ and COBOL languages were used to design the files.
* Shared File System Support
* Fast File System Recovery.

**DBMS:**

* A Database Management System (DBMS) is software designed to store, retrieve, define, and manage data in a database.
* DBMS software primarily functions as an interface between the end user and the database, simultaneously managing the data, the database engine, and the database schema in order to facilitate the organization and manipulation of data.

**Features of DBMS:**

* A user-accessible catalog of data
* Transaction support
* Concurrency control with Recovery services
* Authorization services
* The value of data is the same at all places.
* Offers support for data communication
* Independent utility services
* Allows multiple users to share a file at the same time



* DBMS software primarily functions as an interface between the end user and the database, simultaneously managing the data, the database engine, and the database schema in order to facilitate the organization and manipulation of data.
* Though functions of DBMS vary greatly, general-purpose DBMS features and capabilities should include: a user accessible catalogue describing metadata, DBMS library management system, data abstraction and independence, data security, logging and auditing of activity, support for concurrency and transactions, support for authorization of access, access support from remote locations, DBMS data recovery support in the event of damage, and enforcement of constraints to ensure the data follows certain rules.

**RDBMS:**

* A relational database management system (RDBMS) refers to a collection of programs and capabilities that is designed to enable the user to create, update, and administer a [relational database](https://www.omnisci.com/technical-glossary/relational-database), which is characterized by its structuring of data into logically independent tables, normally including a Structured Query Language (SQL) application programming interface
* All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL and Microsoft Access are based on RDBMS.
* It is a DBMS in which the database is organized and accessed according to the relationships between data items. In a relational database, relationships between data items are expressed by means of tables. Interdependencies among these tables are expressed by data values rather than by pointers. This allows a high degree of data independence.

**DDBMS:**

* A distributed database management system (DDBMS) is a set of multiple, logically interrelated databases distributed over a network. They provide a mechanism that makes the distribution of data transparent to users.
* Databases in the collection are logically interrelated with each other. Often they represent a single logical database.
* Data is physically stored across multiple sites. Data in each site can be managed by a DBMS independent of the other sites.
* The processors in the sites are connected via a network. They do not have any multiprocessor configuration.

**DIFFERENCE**

**1.File System vs DBMS**

|  |  |
| --- | --- |
| **File System** | **DBMS** |
| File system is a software that manages and organizes the files in a storage medium within a computer**.** | DBMS is a software for managing the database. |
| It doesn’t provide backup and recovery of data if it is lost. | It provides backup and recovery of data even if it is lost. |
| Redundant data can be present in a it. | In it there is no redundant data. |
| There is less data consistency in file system | There is more data consistency because of the process of normalization. |
| There is no efficient query processing in file system. | Efficient query processing is there in DBMS. |
| There is no data independence. | In DBMS data independence exists. |
| File systems provide less security in comparison to DBMS. | DBMS has more security mechanisms as compared to file system. |

**2. DBMS VS RDBMS**

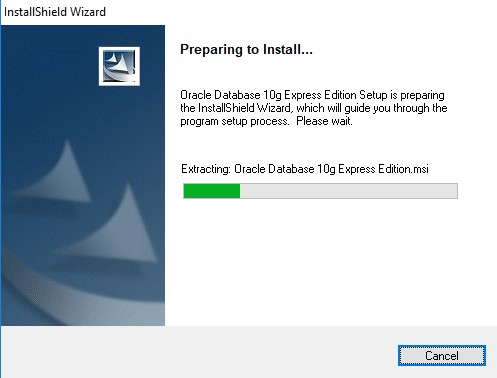
|  |  |
| --- | --- |
| **DBMS** | **RDBMS** |
| DBMS stores data as file. | RDBMS stores data in tabular form. |
| Data elements need to access individually. | Multiple data elements can be accessed at the same time. |
| DBMS does not support distributed database. | RDBMS supports distributed database. |
| Normalization is not present. | Normalization is present. |
| It deals with small quantity of data. | It deals with large amount of data. |
| It supports single user. | It supports multiple users. |
| The data in a DBMS is subject to low security levels with regards to data manipulation. | The data in a DBMS is subject to low security levels with regards to data manipulation. |
| Data fetching is slower for the large amount of data. | Data fetching is fast because of relational approach. |
| Examples: XML, Window Registry, etc | Examples: MySQL, PostgreSQL, SQL Server, Oracle, Microsoft Access etc. |

**PRACTICAL - 2**

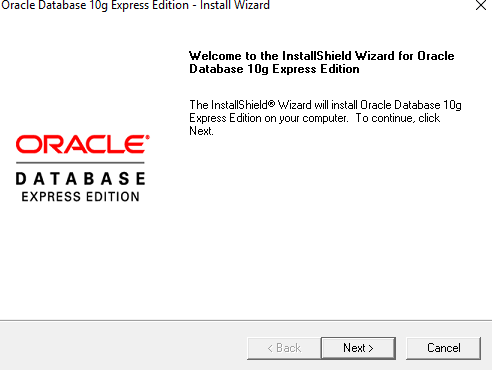
**AIM:**

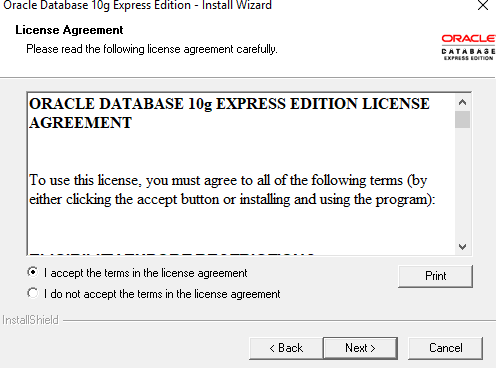
**Introduction to Oracle (step by step installation, introduction of sql, pl/sql).**

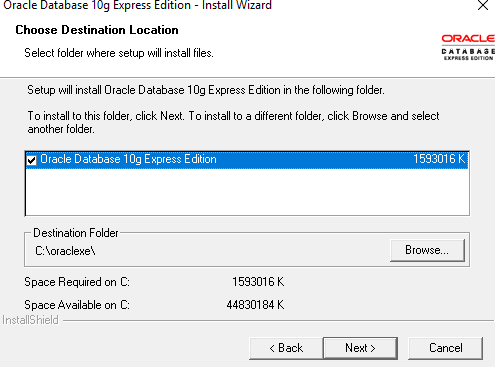
1. **Install it by double clicking .exe which you have downloaded**

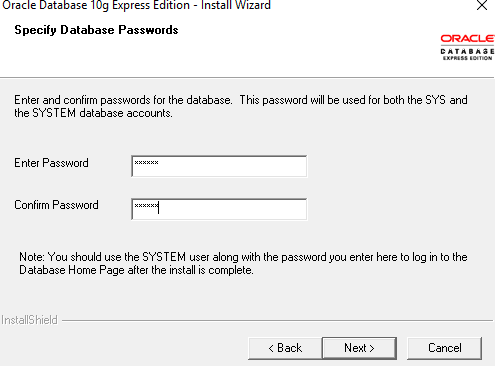
****

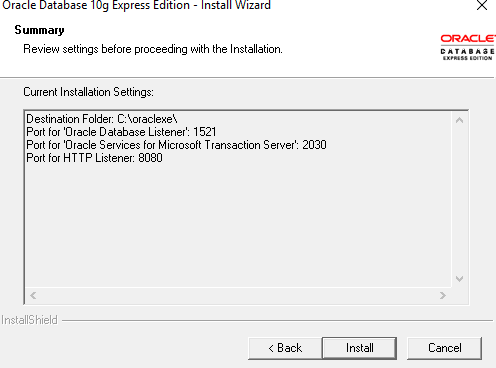
1. **Click on next button**

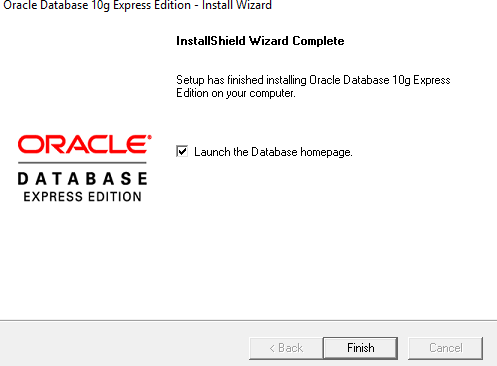
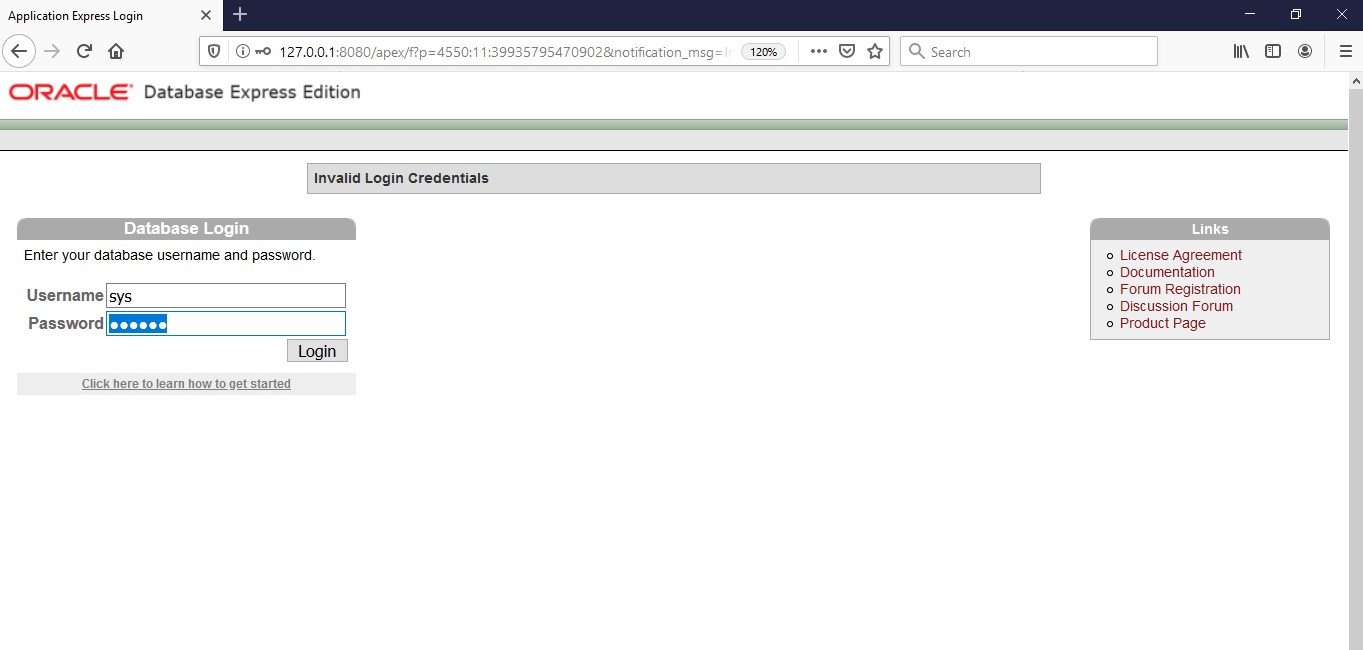
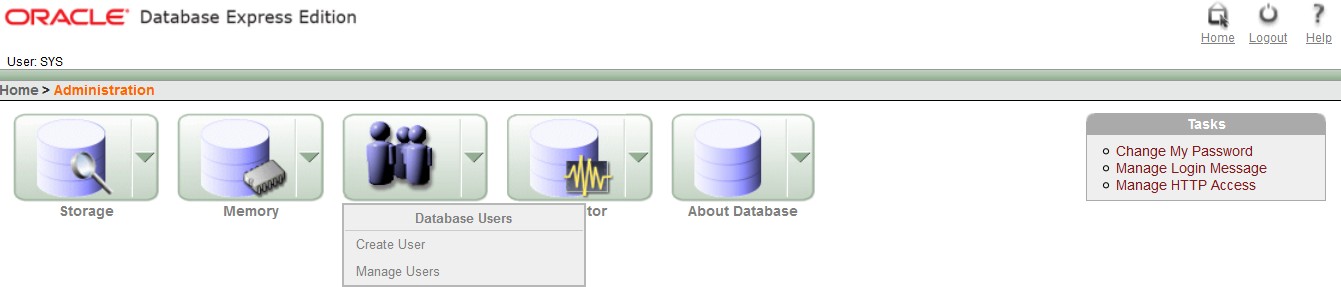
****

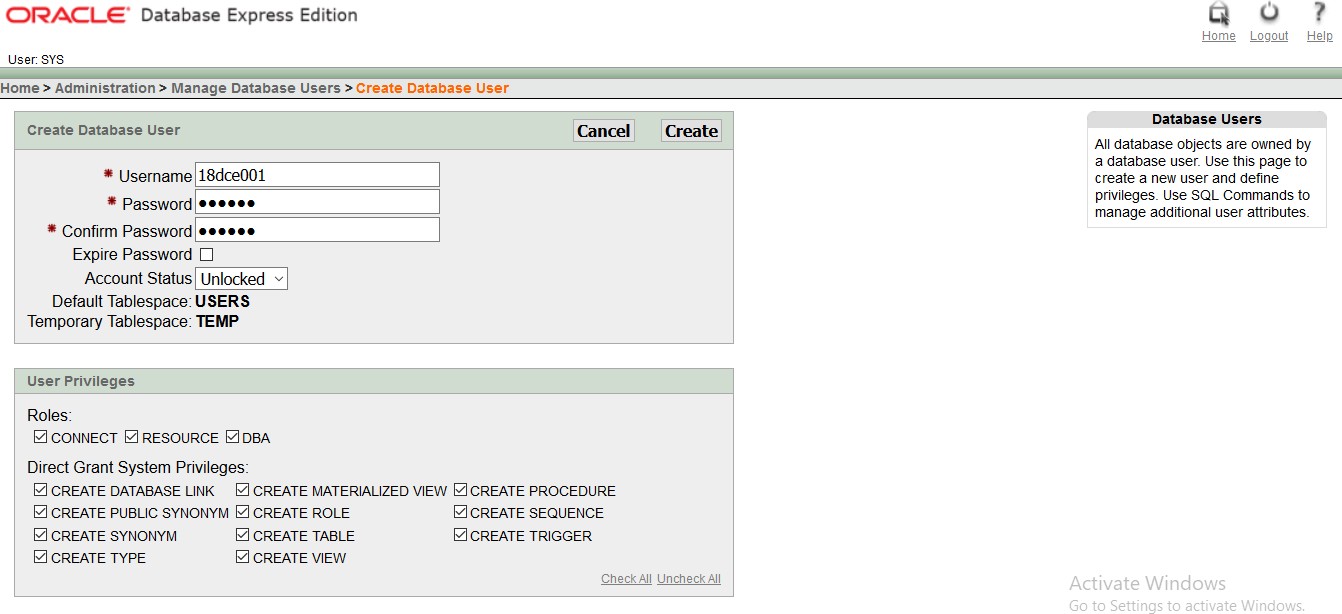
1. **Accept license agreement and click on next button  
     
   **

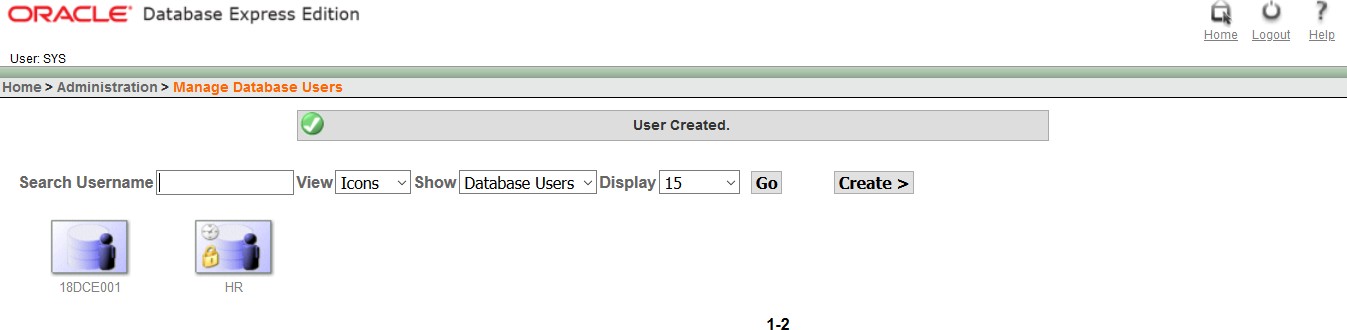
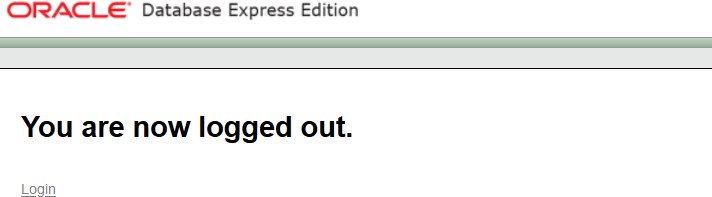
1. **Click on next button  
     
   **
2. **Enter password and confirm password for SYS and SYSTEM user. Please remember it because once installation will be over you have to enter it. To make it easy to**

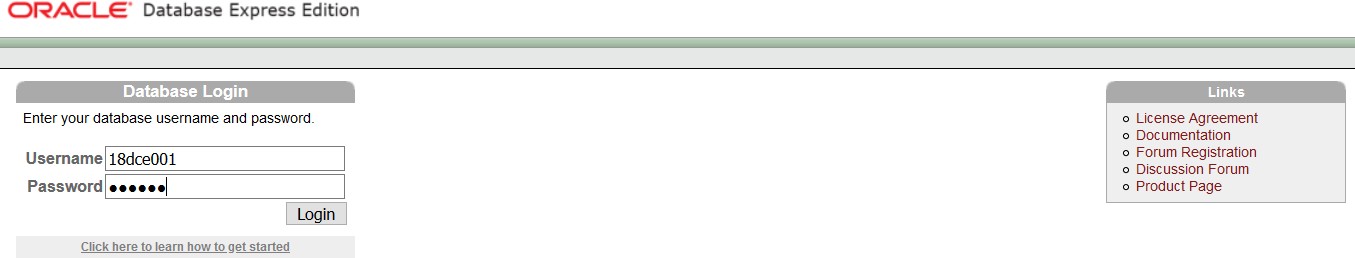
**remember give password as : “oracle”  
  
**

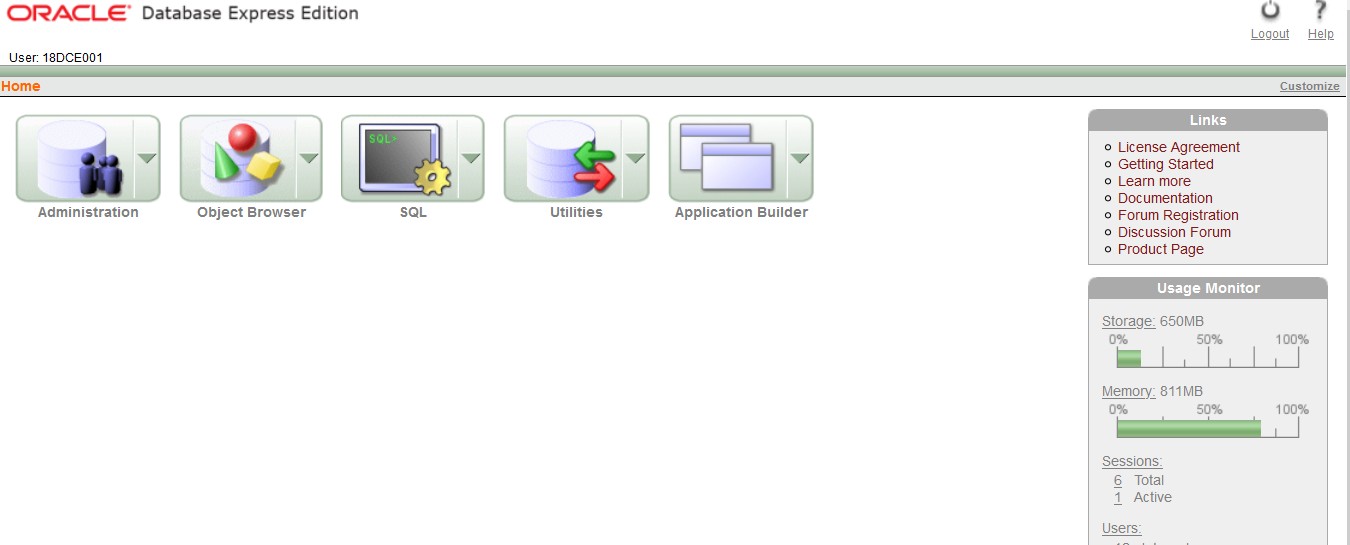
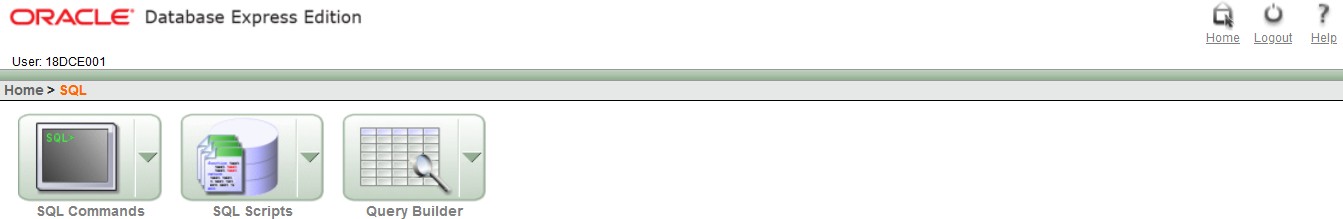
1. **Click on install button  
     
   **

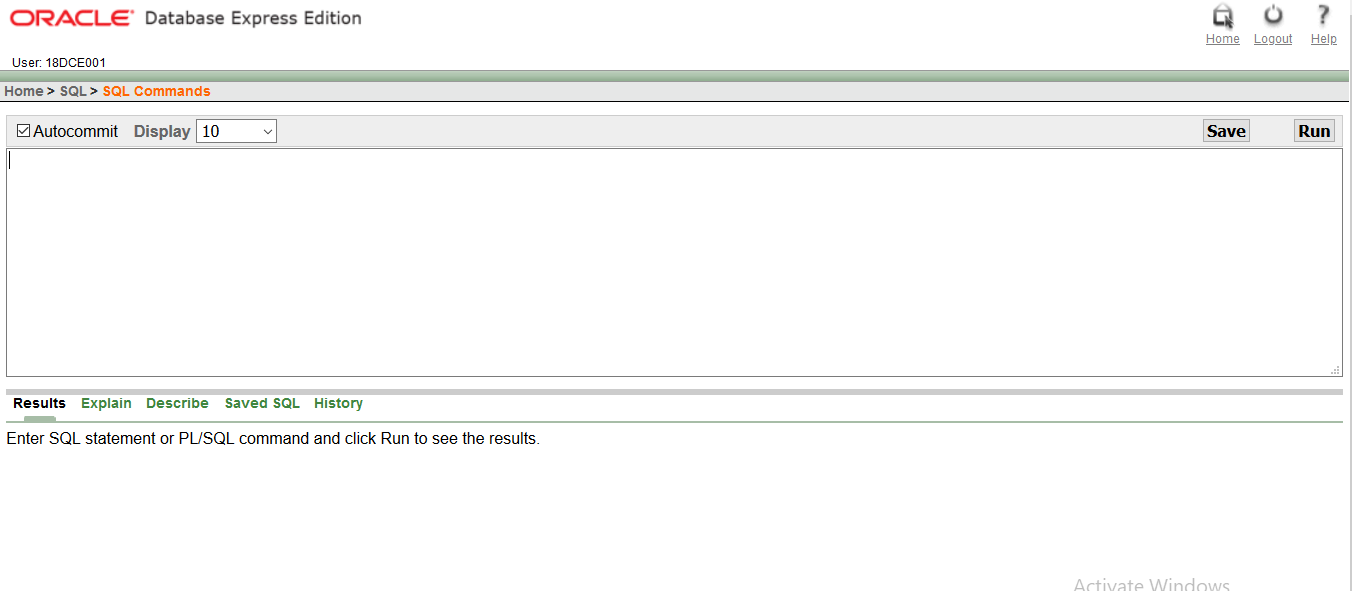
1. **Click on finish button.  
     
   **
2. **Enter username as SYS OR SYSTEM and enter your password (Entered in step: 6)  
     
   **
3. **Click on Administration  
     
   **
4. **Now click on “database user drop down button”. From that click on “create user**  
   
5. **Enter your college roll no in username and give password (NEW) and confirm password. Don’t check expire password, make account status unblocked if it is not. Give all privileges to your user. Finally click on “create” button.**



1. **This page will be shown to you. Now click on “logout” button.  
     
   **
2. **Click on login  
   **
3. **Enter username and password that you just created and click on “login” button**

****

1. **Click on SQL  
     
   **
2. **Home Screen  
     
   **
3. **Congratulation!!! Now you are ready to code SQL and PLSQL.**

****

1. **Thank You**

**INTRODUCTION**

**SQL** :

* SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.
* SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Also, they are using different dialects, such as −

1. MS SQL Server using T-SQL,
2. Oracle using PL/SQL,
3. MS Access version of SQL is called JET SQL (native format) etc.

**PLSQL :**

* PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements. All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

**PRACTICAL – 3**

**AIM:**

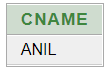
|  |
| --- |
| **To study DDL-create and DML-insert commands**.  **(i) Create tables according to the following definition.**   * **CREATE TABLE DEPOSIT (ACTNO VARCHAR2(5), CNAME VARCHAR2(18), BNAME VARCHAR2(18), AMOUNT NUMBER (8,2), ADATE DATE);** * **CREATE TABLE BRANCH (BNAME VARCHAR2(18), CITY VARCHAR2(18));** * **CREATE TABLE CUSTOMERS (CNAME VARCHAR2(19), CITY VARCHAR2(18));** * **CREATE TABLE BORROW (LOANNO VARCHAR2(5), CNAME VARCHAR2(18), BNAME VARCHAR2(18), AMOUNT NUMBER (8,2));**   **From the above given tables perform the following queries**.  **(1) Describe deposit, branch.**  DESC DEPOSIT1;  DESC BRANCH;    **(2) Describe borrow, customers.**  DESC BORROW;    DESC CUSTOMERS;    **(3) List all data from table DEPOSIT.**  SELECT \* FROM DEPOSIT1;  **(4) List all data from table BORROW.**  SELECT \* FROM BORROW;  **(5) List all data from table CUSTOMERS.**  SELECT \* FROM CUSTOMERS;  **(6) List all data from table BRANCH.**  SELECT \* FROM BRANCH;    **(7) Give account no and amount of depositors.**  SELECT ACTNO, AMOUNT FROM DEPOSIT1;    **(8) Give name of depositors having amount greater than 4000.**  SELECT CNAME FROM DEPOSIT1 WHERE AMOUNT>4000; |

|  |
| --- |
| **(9) Give name of customers who opened account after date '1-12-96'.** SELECT CNAME FROM DEPOSIT1 WHERE ADATE>'01-DEC-96'; |

**(10) Give name of city where branch karolbagh is located.**SELECT CITY FROM BRANCH WHERE BNAME='KAROLBAGH'; ****

**(11) Give account no and amount of customer having account opened between date 1-12-96 and 1-6- 96.**SELECT ACTNO, AMOUNT FROM DEPOSIT WHERE ADATE BETWEEN '01-DEC-96' AND '01-JUN-96';

****

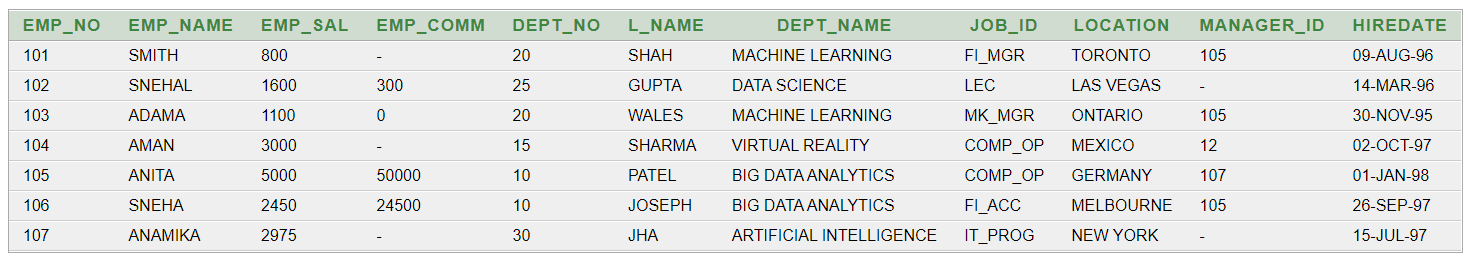
**(12) Give names of depositors having account at VRCE.**   
  
SELECT CNAME FROM DEPOSIT1 WHERE BNAME='VRCE'; ****

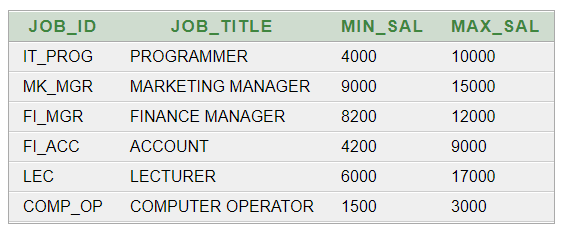
**PRACTICAL - 4**

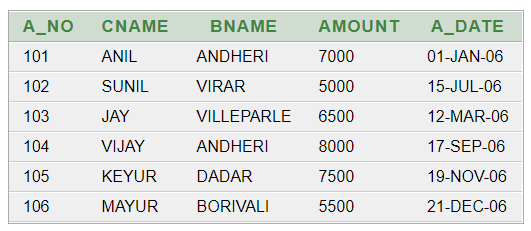
**AIM:**

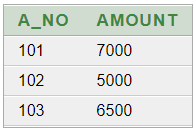
**Create the below given table and insert the data accordingly**.

**QUERIES:**

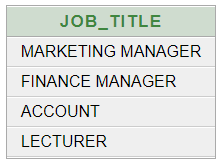
**(1) Retrieve all data from employee, jobs and deposit.**SELECT \* FROM EMPLOYEE; ****  
SELECT \* FROM JOB;



SELECT \* FROM DEPOSIT2;  
  


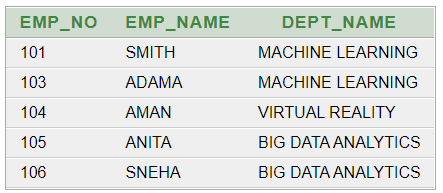
**(2) Give details of account no. and deposited rupees of customers having account opened between dates 01-01-06 and 25-07-06.**   
  
SELECT A\_NO, AMOUNT FROM DEPOSIT2 WHERE A\_DATE BETWEEN '01-JAN-06' AND '25-JUL-06';  
  


**(3) Display all jobs with minimum salary is greater than 4000.**   
  
SELECT JOB\_TITLE FROM JOB WHERE MIN\_SAL > 4000;

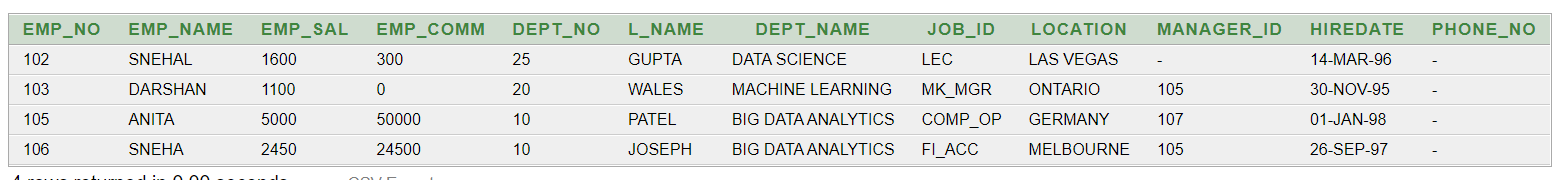


**(4) Display name and salary of employee whose department no is 20. Give alias name to name of employee.**   
  
SELECT EMP\_NAME AS NAME\_OF\_EMPLOYEE, EMP\_SAL FROM EMPLOYEE WHERE DEPT\_NO=20;  
  

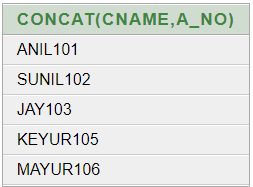

**(5) Display employee no, name and department details of those employee whose department lies in (10,20).**SELECT EMP\_NO, EMP\_NAME, DEPT\_NAME FROM EMPLOYEE WHERE DEPT\_NO BETWEEN '10' AND '20';

****

**(6) Display the non-null values of employees.**SELECT \* FROM EMPLOYEE WHERE EMP\_COMM IS NOT NULL;

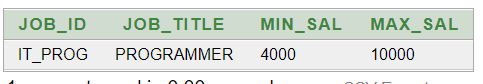
****

**(7) Display name of customer along with its account no (both column should be displayed as one) whose amount is not equal to 8000 Rs.**SELECT CONCAT(CNAME, A\_NO) FROM DEPOSIT2 WHERE AMOUNT != '8000';

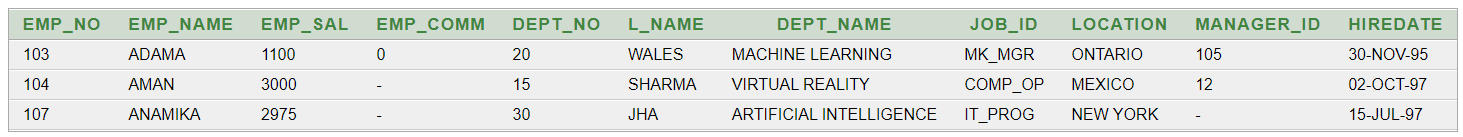
****

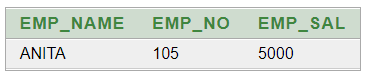
**(8)** **Display the content of job details with minimum salary either 2000 or 4000.**

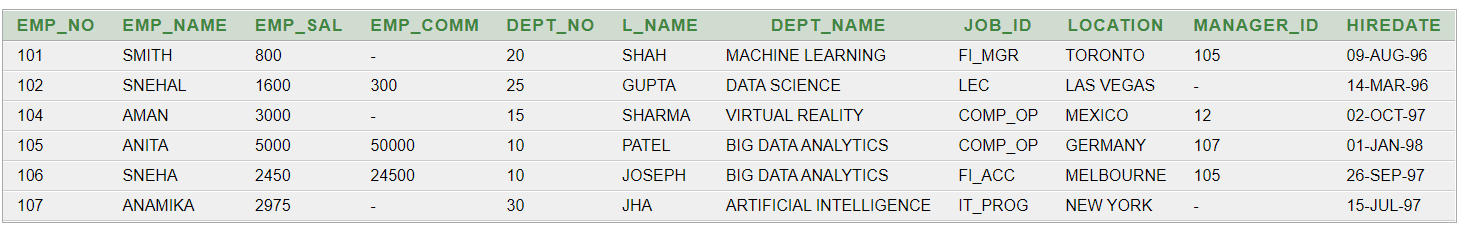
SELECT \* FROM JOB WHERE MIN\_SAL=2000 OR MIN\_SAL=4000;



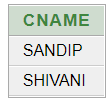
**To study various options of LIKE predicate**

**(1) Display all employee whose name start with ‘A’ and third character is ‘‘a’.**   
  
SELECT \* FROM EMPLOYEE WHERE EMP\_NAME LIKE 'A\_A%';  
  


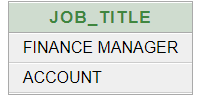
**(2) Display name, number and salary of those employees whose name is 5 characters long and first three characters are ‘Ani’.**   
  
SELECT EMP\_NAME, EMP\_NO, EMP\_SAL FROM EMPLOYEE WHERE EMP\_NAME LIKE 'ANI\_\_';  
  


**(3) Display all information of employee whose second character of name is either ‘M’ or ‘N’.**   
  
SELECT \* FROM EMPLOYEE WHERE EMP\_NAME LIKE '\_M%' OR EMP\_NAME LIKE '\_N%';  
  


**(4) Find the list of all customer name whose branch is in ‘andheri’ or ‘dadar’ or ‘virar’.**SELECT CNAME FROM DEPOSIT WHERE BNAME IN ('ANDHERI', 'DADAR', 'VIRAR')



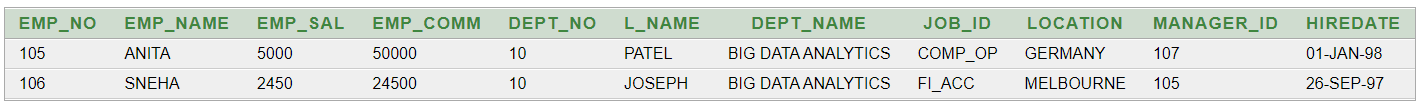
**(5) Display the job name whose first three character in job id field is ‘FI\_’.**SELECT JOB\_TITLE FROM JOB WHERE JOB\_ID LIKE 'FI\_%';

****

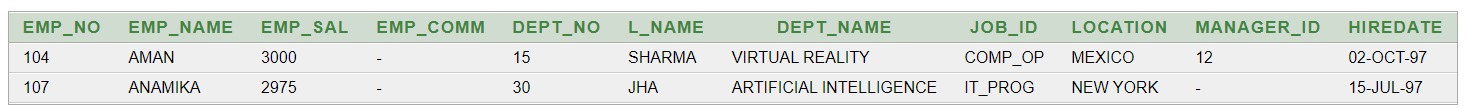
**(6) Display the title/name of job who’s last three character are ‘\_MGR’ and their maximum salary is greater than Rs 12000.**SELECT JOB\_TITLE FROM JOB WHERE JOB\_ID LIKE '%\_MGR' AND MAX\_SAL>12000;

****

**(7) Display the non-null values of employees and also employee name second character should be ‘n’ and string should be 5-character long.**   
  
SELECT \* FROM EMPLOYEE WHERE EMP\_COMM IS NOT NULL AND EMP\_NAME LIKE '\_N\_\_\_';

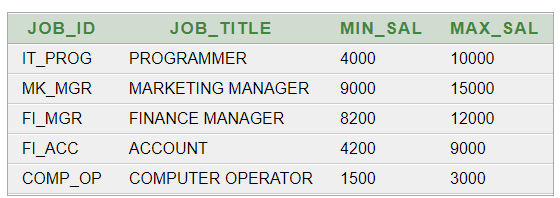


**(8) Display the null values of employee and also employee name’s third character should be ‘a’.**SELECT \* FROM EMPLOYEE WHERE EMP\_COMM IS NULL AND EMP\_NAME LIKE '\_\_A%';

****

**(9) What will be output if you are giving LIKE predicate as ‘%\\_%’ ESCAPE ‘\’**

SELECT \* FROM JOB WHERE JOB\_ID LIKE '%\\_%' ESCAPE '\';



**PRACTICAL - 5**

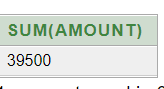
**AIM:**

**To Perform various data manipulation commands, aggregate functions and sorting concept on all created tables.**

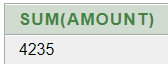
**QUERIES:**

**(1) List total deposit from deposit.**

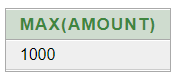
SELECT SUM(AMOUNT) FROM DEPOSIT2;



**(2) List total loan from karolbagh branch**   
  
SELECT SUM(AMOUNT) FROM DEPOSIT1 WHERE BNAME = 'KAROLBAGH';



**(3) Give maximum loan from branch vrce.**SELECT MAX(AMOUNT) FROM BORROW WHERE BNAME= 'VRCE';

****

**(4) Count total number of customers**

SELECT COUNT(CNAME) FROM CUSTOMERS;



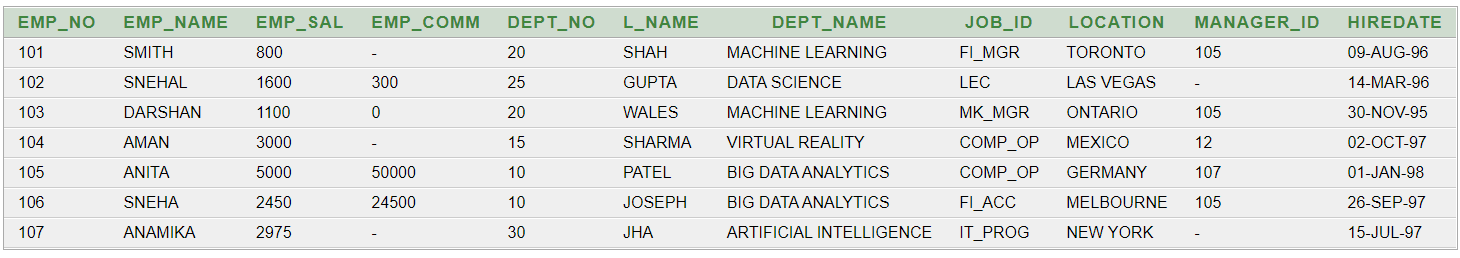
**(5) Count total number of customer’s cities.**SELECT COUNT(CITY) FROM CUSTOMERS;

****

**(6) Create table supplier from employee with all the columns.**CREATE TABLE SUPPLIER AS SELECT \* FROM EMPLOYEE;

SELECT \* FROM SUPPLIER;

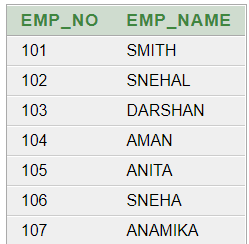




**(7) Create table sup1 from employee with first two columns.**  
CREATE TABLE SUP1 AS SELECT EMP\_NO, EMP\_NAME FROM EMPLOYEE;

SELECT \* FROM SUP1;





**(8) Create table sup2 from employee with no data**CREATE TABLE SUP2 AS SELECT \* FROM EMPLOYEE WHERE EMP\_NO = NULL;

SELECT \* FROM SUP2;

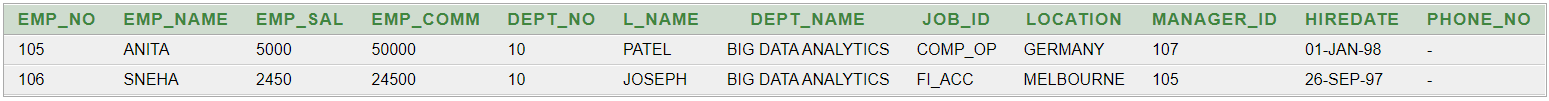


**(9) Insert the data into sup2 from employee whose second character should be ‘n’ and string should be 5 characters long in employee name field.**

INSERT INTO SUP2 SELECT \* FROM EMPLOYEE WHERE EMP\_NAME LIKE '\_N\_\_\_';

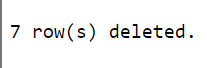
SELECT \* FROM SUP2;





**(10) Delete all the rows from sup1.**

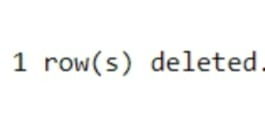
DELETE FROM SUP1;

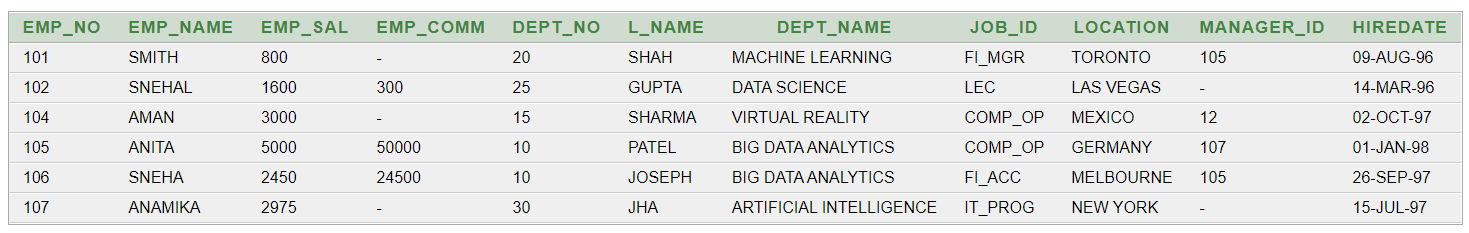
****

**(11) Delete the detail of supplier whose sup\_no is 103.**

DELETE FROM SUPPLIER WHERE EMP\_NO=103;

SELECT \* FROM SUPPLIER;





**(12) Rename the table sup2.**

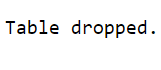
ALTER TABLE SUP2 RENAME TO GLOBAL;

SELECT \* FROM GLOBAL;



**(13) Destroy table sup1 with all the data.**

DROP TABLE SUP1

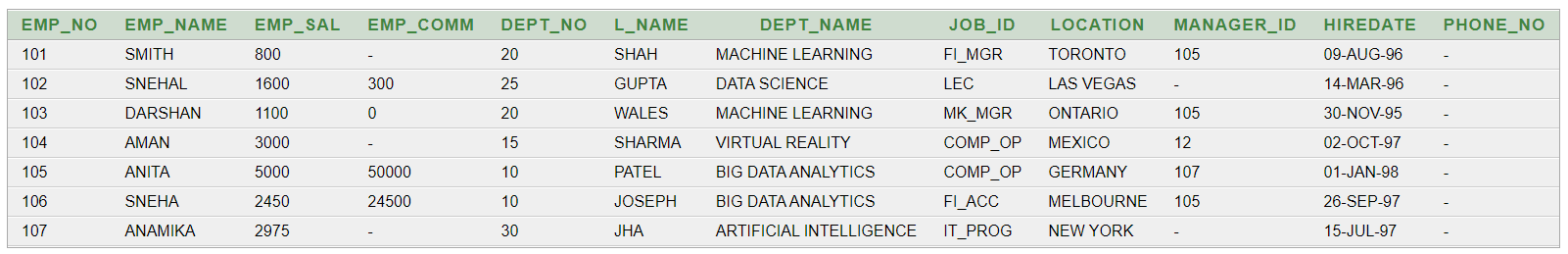


**(14) Update the value dept\_no to 10 where second character of emp. name is ‘m’.**

UPDATE EMPLOYEE SET DEPT\_NO=10 WHERE EMP\_NAME LIKE '\_m%';

SELECT \* FROM EMPLOYEE;

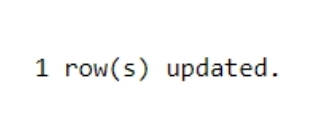


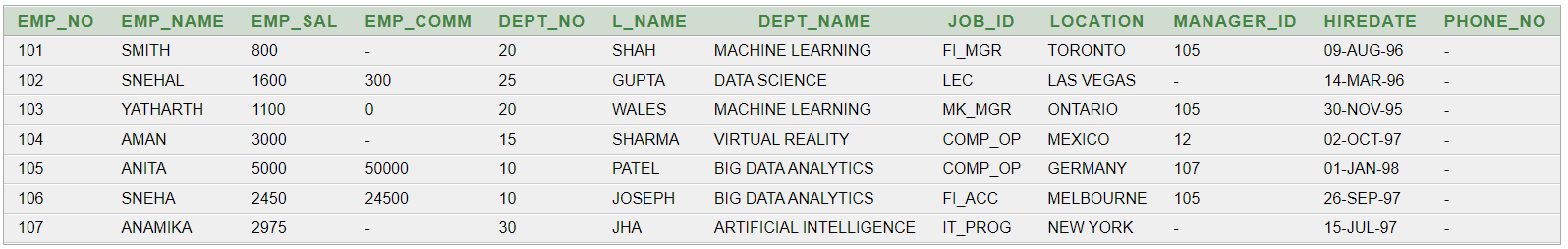


**(15) Update the value of employee name whose employee number is 103.**

UPDATE EMPLOYEE SET EMP\_NAME='YATHARTH' WHERE EMP\_NO=103;

SELECT \* FROM EMPLOYEE;

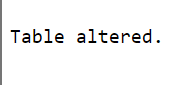


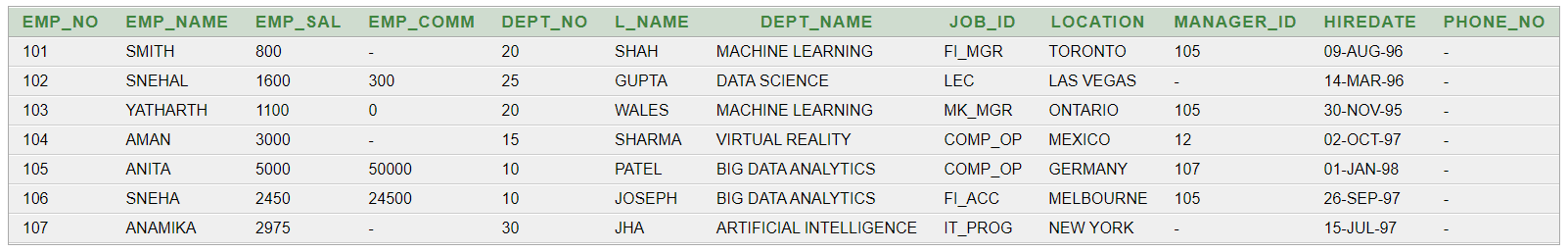


**(16) Add one column phone to employee with size of column is 10.**

ALTER TABLE EMPLOYEE ADD PHONE\_NO NUMBER(10,0);

SELECT \* FROM EMPLOYEE;



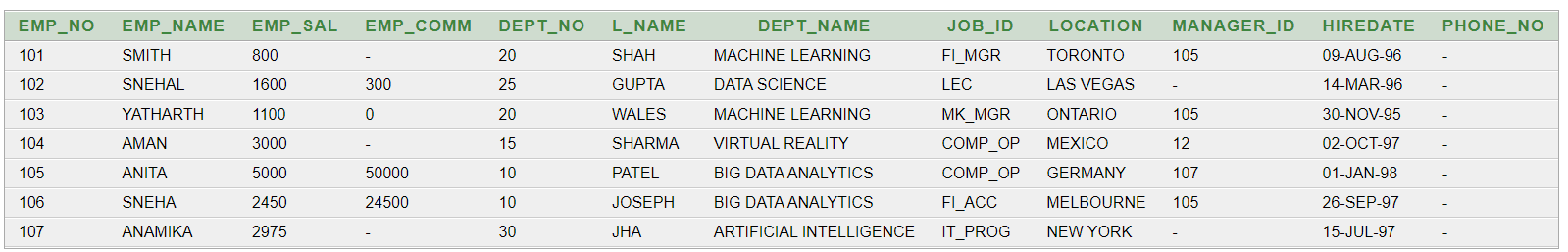


**(17) Modify the column emp\_name to hold maximum of 30 characters.**

ALTER TABLE EMPLOYEE MODIFY EMP\_NAME VARCHAR(30);

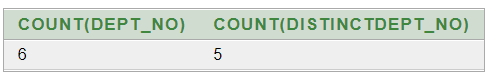
SELECT \* FROM EMPLOYEE;





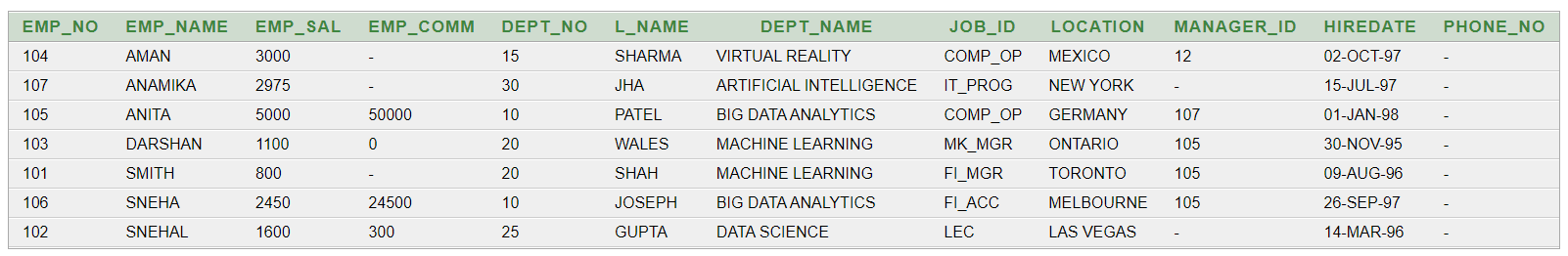
**(18) Count the total no as well as distinct rows in dept\_no column with a condition of salary greater than 1000 of employee**

SELECT COUNT(DEPT\_NO), COUNT(DISTINCT DEPT\_NO) FROM EMPLOYEE WHERE EMP\_SAL > 1000;



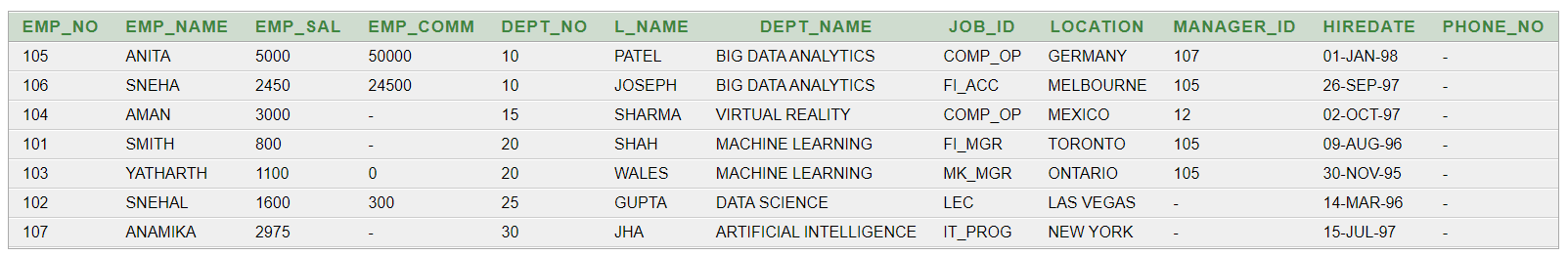
**(19) Display the detail of all employees in ascending order, descending order of their name and no.**

SELECT \* FROM EMPLOYEE ORDER BY EMP\_NAME ASC , EMP\_NO DESC;

****

**(20) Display the dept\_no in ascending order and accordingly display emp\_comm in descending order.**

SELECT \* FROM EMPLOYEE ORDER BY DEPT\_NO ASC, EMP\_COMM DESC;

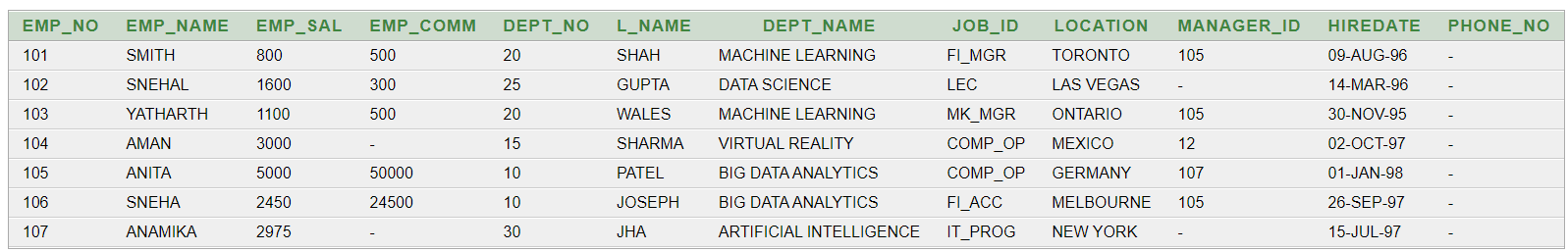
****

**(21) Update the value of emp\_comm to 500 where dept\_no is 20.**

UPDATE EMPLOYEE SET EMP\_COMM = 500 WHERE DEPT\_NO = 20;

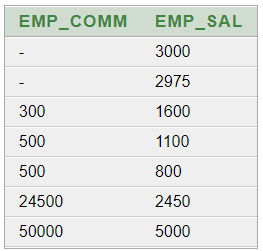
SELECT \* FROM EMPLOYEE;



****

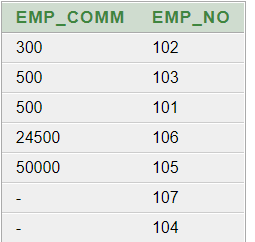
**(22) Display the emp\_comm in ascending order with null value first and accordingly sort employee salary in descending order.**

SELECT EMP\_COMM, EMP\_SAL FROM EMPLOYEE ORDER BY EMP\_COMM ASC NULLS FIRST, EMP\_SAL DESC;



**(23) Display the emp\_comm in ascending order with null value last and accordingly sort emp\_no in descending order.**

SELECT EMP\_COMM, EMP\_NO FROM EMPLOYEE ORDER BY EMP\_COMM ASC NULLS LAST, EMP\_NO DESC;



**PRACTICAL - 6**

**AIM:**

**To study Single-row functions.**

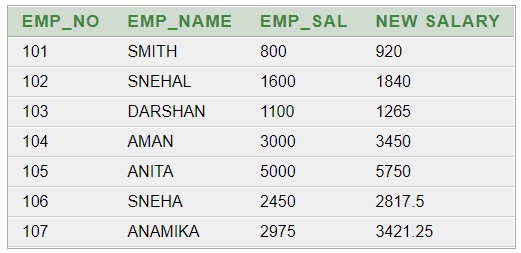
**QUERIES:**

**(1) Write a query to display the current date. Label the column Date**

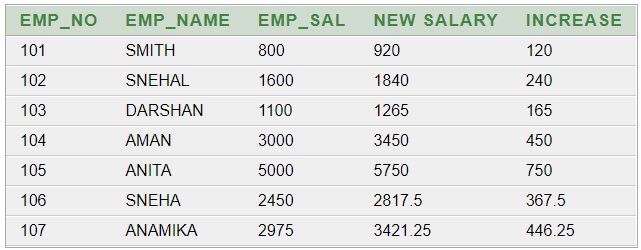
SELECT SYSDATE AS "DATE" FROM DUAL;

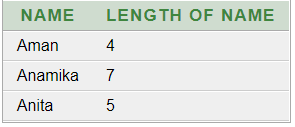


**(2) For each employee, display the employee number, salary, and salary increased by 15% and expressed as a whole number. Label the column New Salary**SELECT EMP\_NO, EMP\_NAME,EMP\_SAL, EMP\_SAL+(EMP\_SAL\*(15/100)) AS "NEW SALARY" FROM EMPLOYEE;

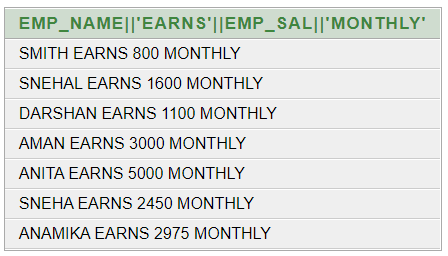
****

**(3) Modify your query no (2) to add a column that subtracts the old salary from the new salary. Label the column Increase**

SELECT EMP\_NO,EMP\_NAME,EMP\_SAL,EMP\_SAL+(EMP\_SAL\*15/100) AS "NEW SALARY", (EMP\_SAL+(EMP\_SAL\*(15/100))) - EMP\_SAL AS "INCREASE" FROM EMPLOYEE;  
  
****

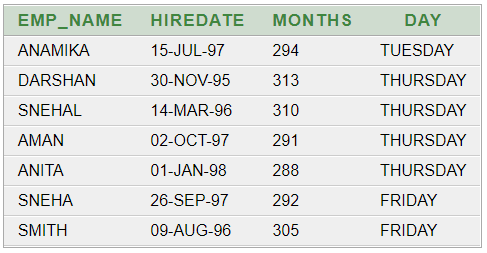
**(4) Write a query that displays the employee’s names with the first letter capitalized and all other letters lowercase, and the length of the names, for all employees whose name starts with J, A, or M. Give each column an appropriate label. Sort the results by the employees’ last names.**SELECT INITCAP(EMP\_NAME) AS "NAME", LENGTH(EMP\_NAME) AS "LENGTH OF NAME" FROM EMPLOYEE WHERE EMP\_NAME LIKE 'J%' OR EMP\_NAME LIKE 'A%' OR EMP\_NAME LIKE 'M%' ORDER BY EMP\_NAME; ****

**(5) Write a query that produces the following for each employee:**

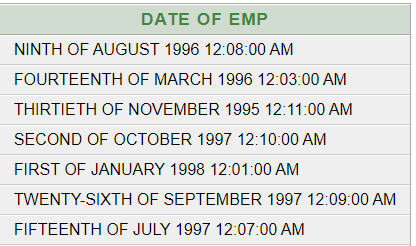
**<employee last name> earns <salary> monthly**SELECT EMP\_NAME || ' EARNS ' || EMP\_SAL || ' MONTHLY' FROM EMPLOYEE;  
 ****

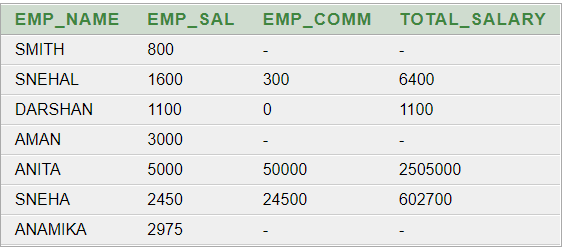
**(6) Display the name, date, number of months employed and day of the week on which the employee has started. Order the results by the day of the week starting with Monday.**

SELECT EMP\_NAME, HIREDATE, ROUND (MONTHS\_BETWEEN (SYSDATE, HIREDATE)) AS MONTHS, TO\_CHAR(HIREDATE, 'DAY') AS "DAY" FROM EMPLOYEE ORDER BY TO\_CHAR(HIREDATE, 'DAY') DESC;



**(7) Display the date of emp in a format that appears as Seventh of June 1994 12:00:00 AM.**

SELECT TO\_CHAR(HIREDATE, 'DDSPTH "OF" MONTH YYYY HH:MM:SS AM') "DATE OF EMP" FROM EMPLOYEE;  
  


**(8) Write a query to calculate the annual compensation of all employees (sal +comm.).**SELECT EMP\_NAME, EMP\_SAL, EMP\_COMM, (EMP\_SAL+ ((EMP\_SAL \* EMP\_COMM) / 100)) AS "TOTAL\_SALARY" FROM EMPLOYEE;  
  
****

**PRACTICAL - 7**

**AIM:**

**Displaying data from Multiple Tables (join)**

**QUERIES:**

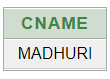
**(1) Give details of customers ANIL.**

SELECT \* FROM DEPOSIT1 JOIN BORROW ON DEPOSIT1.CNAME = BORROW.CNAME JOIN CUSTOMERS ON DEPOSIT1.CNAME = CUSTOMERS.CNAME WHERE DEPOSIT1.CNAME = 'ANIL';



**(2) Give name of customer who are borrowers and depositors and having living city nagpur**

SELECT BORROW.CNAME FROM BORROW JOIN DEPOSIT1 ON BORROW.CNAME = DEPOSIT1.CNAME JOIN CUSTOMERS ON BORROW.CNAME = CUSTOMERS.CNAME WHERE CITY = 'NAGPUR';



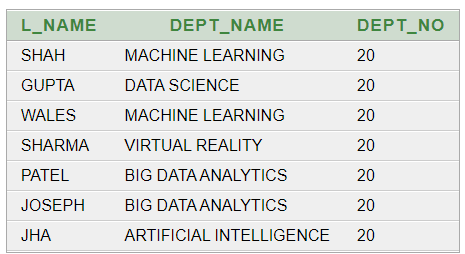
**(3) Give city as their city name of customers having same living branch.**

SELECT CITY FROM BRANCH JOIN DEPOSIT1 ON BRANCH.BNAME = DEPOSIT1.BNAME;



**(4) Write a query to display the last name, department number, and department name for all employees.**

SELECT L\_NAME, DEPT\_NAME,DEPT\_NO FROM EMPLOYEE;



**(5) Create a unique listing of all jobs that are in department 30. Include the location of the department in the output**

SELECT JOB\_TITLE, LOCATION FROM JOB JOIN EMPLOYEE ON JOB.JOB\_ID = EMPLOYEE.JOB\_ID WHERE DEPT\_NO = 30;



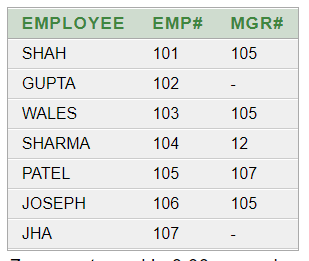
**(6) Write a query to display the employee name, department number, and department name for all employees who work in NEW YORK.**

SELECT EMP\_NAME,DEPT\_NO, DEPT\_NAME FROM EMPLOYEE WHERE LOCATION='NEW YORK';



**(7) Display the employee last name and employee number along with their manager’s last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively.**

SELECT L\_NAME AS "EMPLOYEE", EMP\_NO AS "EMP#", MANAGER\_ID AS "MGR#" FROM EMPLOYEE;



**(8) Create a query to display the name and hire date of any employee hired after employee “smith”.**

SELECT EMP\_NAME, HIREDATE FROM EMPLOYEE WHERE HIREDATE>(SELECT HIREDATE FROM EMPLOYEE WHERE EMP\_NAME='SMITH');



**PRACTICAL - 8**

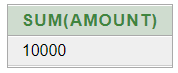
**AIM:**

**To apply the concept of Aggregating Data using Group functions.**

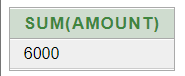
**QUERIES:**

**(1) List total deposit of customer having account date after 1-jan-96.**

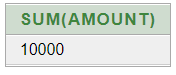
SELECT SUM(AMOUNT) FROM DEPOSIT1 WHERE ADATE > '1-JAN-96'

****

**(2) List total deposit of customers living in city Nagpur.**SELECT SUM(AMOUNT) FROM DEPOSIT1 NATURAL JOIN BRANCH WHERE CITY='NAGPUR';

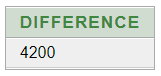
****

**(3) List maximum deposit of customers living in bombay.**SELECT SUM(AMOUNT) FROM DEPOSIT1 NATURAL JOIN BRANCH WHERE CITY='BOMBAY';

****

**(4) Display the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number.**SELECT MAX (EMP\_SAL) "MAXIMUM" ,MIN(EMP\_SAL) "MINIMUM",SUM(EMP\_SAL) "SUM", AVG (EMP\_SAL) "AVERAGE" FROM EMPLOYEE;

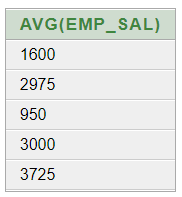
****

**(5) Write a query that displays the difference between the highest and lowest salaries. Label the column DIFFERENCE.**SELECT MAX(EMP\_SAL)-MIN(EMP\_SAL) "DIFFERENCE" FROM EMPLOYEE; ****

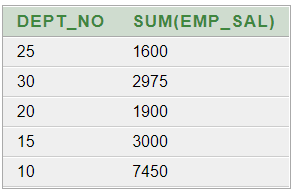
**(6) Create a query that will display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998**SELECT COUNT (EMP\_NO) FROM EMPLOYEE;

****

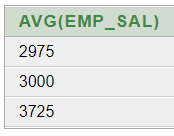
**(7) Find the average salaries for each department without displaying the respective department numbers.**SELECT AVG(EMP\_SAL) FROM EMPLOYEE GROUP BY DEPT\_NO;

****

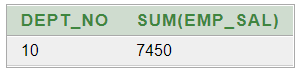
**(8) Write a query to display the total salary being paid to each job title, within each department.**SELECT DEPT\_NO,SUM(EMP\_SAL) FROM EMPLOYEE GROUP BY DEPT\_NO;

****

**(9) Find the average salaries > 2000 for each department without displaying the respective department numbers.**

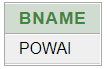
SELECT AVG(EMP\_SAL) FROM EMPLOYEE GROUP BY DEPT\_NO HAVING AVG(EMP\_SAL)> 2000; ****

**(10) Display the job and total salary for each job with a total salary amount exceeding 3000 and sorts the list by the total salary.**SELECT DEPT\_NO,SUM(EMP\_SAL) FROM EMPLOYEE GROUP BY DEPT\_NO HAVING SUM(EMP\_SAL) > 3000 ORDER BY SUM(EMP\_SAL);

****

**(11) List the branches having sum of deposit more than 5000 and located in city bombay.**

SELECT BNAME FROM DEPOSIT1 NATURAL JOIN BRANCH GROUP BY BNAME HAVING SUM(AMOUNT)>5000;



**PRACTICAL - 9**

**AIM:**

**To solve queries using the concept of sub query**

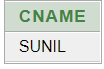
**QUERIES:**

**(1) Write a query to display the last name and hire date of any employee in the same department as smith. Exclude smith**

SELECT L\_NAME, HIREDATE FROM EMPLOYEE WHERE DEPT\_NAME=(SELECT DEPT\_NAME FROM EMPLOYEE WHERE EMP\_NAME='SMITH') AND EMP\_NAME<>'SMITH';

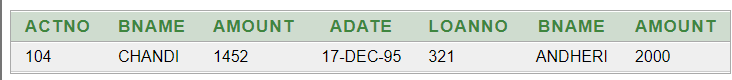
****

**(2) Give name of customers who are depositors having same branch city of mr. sunil.**SELECT CNAME FROM DEPOSIT1 WHERE BNAME=(SELECT BNAME FROM DEPOSIT1 WHERE CNAME='SUNIL');

****

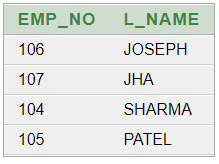
**(3) Give deposit details and loan details of customer in same city where pramod is living.**

SELECT DISTINCT D1.ACTNO,D1.BNAME , D1.AMOUNT , D1.ADATE , B1.LOANNO , B1.BNAME , B1.AMOUNT FROM DEPOSIT1 D1 , BORROW B1 , CUSTOMERS C1 WHERE C1.CNAME = D1.CNAME AND D1.CNAME=B1.CNAME AND C1.CITY IN (SELECT C2.CITY FROM CUSTOMERS C2 WHERE C2.CNAME='PRAMOD');



**(4) Create a query to display the employee numbers and last names of all employees who earn more than the average salary. Sort the results in ascending order of salary.**

SELECT EMP\_NO,L\_NAME FROM EMPLOYEE WHERE EMP\_SAL > (SELECT AVG(EMP\_SAL) FROM EMPLOYEE) ORDER BY EMP\_SAL;

****

**(5) Givenames of depositors having same living city as mr. anil and having deposit amount greater than 2000**

SELECT D.CNAME FROM DEPOSIT1 D,CUSTOMERS C WHERE C.CITY = ( SELECT C.CITY FROM CUSTOMERS C WHERE D.CNAME=C.CNAME AND D.CNAME='ANIL') AND D.AMOUNT > 2000 AND D.CNAME!= 'ANIL';



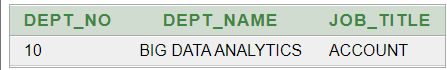
**(6) Display the last name and salary of every employee who reports to ford.**

SELECT L\_NAME, EMP\_SAL FROM EMPLOYEE WHERE EMP\_NAME = 'FORD'



**(7) Display the department number, name, and job for every employee in the Accounting department.**

SELECT E.DEPT\_NO, E.DEPT\_NAME, J.JOB\_TITLE FROM EMPLOYEE E, JOB J WHERE E.JOB\_ID=J.JOB\_ID AND J.JOB\_TITLE='ACCOUNT';



**(8) List the name of branch having highest number of depositors.**

SELECT BNAME FROM DEPOSIT1 HAVING COUNT(BNAME)=(SELECT MAX(COUNT(BNAME)) FROM DEPOSIT1 GROUP BY BNAME) GROUP BY BNAME;



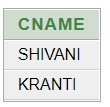
**(9) Give the name of cities where in which the maximum numbers of branches are located.**

SELECT CITY FROM BRANCH GROUP BY CITY HAVING COUNT(BNAME) = (SELECT MAX(COUNT(BNAME)) FROM BRANCH GROUP BY CITY)



**(10) Give name of customers living in same city where maximum depositors are located.**

SELECT CNAME FROM CUSTOMERS WHERE CNAME IN (SELECT CNAME FROM DEPOSIT1) AND CITY IN (SELECT CITY FROM CUSTOMERS GROUP BY CITY HAVING COUNT(CNAME) = (SELECT MAX(COUNT(CNAME)) FROM CUSTOMERS GROUP BY CITY))

****

**PRACTICAL - 10**

**AIM:**

**Manipulating Data**

**QUERIES:**

**(1) Give 10% interest to all depositors.**UPDATE DEPOSIT1 SET AMOUNT = AMOUNT + (AMOUNT\*10/100);

SELECT \* FROM DEPOSIT1;

****

**(2) Give 10% interest to all depositors having branch vrce**

UPDATE DEPOSIT1 SET AMOUNT = AMOUNT + (AMOUNT\*10/100) WHERE BNAME ='VRCE';

SELECT \* FROM DEPOSIT1; ****

**(3) Give 10% interest to all depositors living in nagpur and having branch city bombay.**

UPDATE DEPOSIT1 SET AMOUNT = AMOUNT + (AMOUNT\*10/100) WHERE CNAME IN (SELECT CNAME FROM CUSTOMERS WHERE CITY='NAGPUR') AND BNAME IN(SELECT BNAME FROM BRANCH WHERE CITY='BOMBAY');



**(4) Write a query which changes the department number of all employees with empno 7788’s job to employee 7844’current department number.**

UPDATE EMPLOYEE SET DEPT\_NO = (SELECT DEPT\_NO FROM EMPLOYEE WHERE EMP\_NO = 103) WHERE JOB\_ID = (SELECT JOB\_ID FROM EMPLOYEE WHERE EMP\_NO = 102)

****

**(5) Transfer 10 Rs from account of anil to sunil if both are having same branch.**

UPDATE DEPOSIT1 SET AMOUNT-10 WHERE CNAME = 'ANIL' AND BNAME IN(SELECT D1.BNAME FROM DEPOSIT1 D1 WHERE D1.CNAME = 'SUNIL')

UPDATE DEPOSIT1 SET AMOUNT-10 WHERE CNAME = 'SUNIL' AND BNAME IN(SELECT D1.BNAME FROM DEPOSIT1 D1 WHERE D1.CNAME = 'ANIL')

****

**(6) Give 100 Rs more to all depositors if they are maximum depositors in their respective branch.**

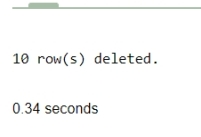
UPDATE DEPOSIT1 SET AMOUNT=AMOUNT+100 WHERE BNAME IN(SELECT BNAME FROM DEPOSIT1 GROUP BY BNAME HAVING COUNT(ACTNO) = (SELECT MAX(COUNT(ACTNO)) FROM DEPOSIT1 GROUP BY BNAME))



**(7) Delete depositors of branches having number of customers between 1 to 3.**

DELETE FROM DEPOSIT1 WHERE BNAME IN (SELECT BNAME FROM DEPOSIT1 GROUP BY BNAME HAVING COUNT(ACTNO) >= 1 AND

COUNT(ACTNO) <=3)



**(8) Delete deposit of vijay.**

DELETE FROM DEPOSIT2 WHERE CNAME='VIJAY';



**(9) Delete borrower of branches having average loan less than 1000.**

DELETE FROM BORROW WHERE CNAME IN (SELECT CNAME FROM BORROW B GROUP BY CNAME HAVING AVG(AMOUNT)< 1000);



**PRACTICAL - 11**

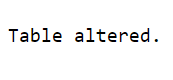
**AIM:**

**Add and Remove constraint**

**QUERIES:**

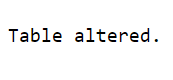
**(1) Add primary key constraint on job\_id in job table.**

ALTER TABLE JOB ADD CONSTRAINT PK\_JOB PRIMARY KEY(JOB.ID)

****

**(2)Add foreign key constraint on employee table referencing job table.**

ALTER TABLE EMPLOYEE ADD CONSTRAINT FK\_JOB\_ID FOREIGN KEY (JOB\_ID) REFERENCES JOB(JOB\_ID );

****

**(3)Add composite primary key on lock table (lock table does not exist, while creating table add composite key)**

CREATE TABLE LOCK\_T(

PROD\_ID INT NOT NULL,

NAME VARCHAR(45 ) ,

MANUFACTURE VARCHAR(45),

PRIMARY KEY(NAME. MANUFACTURE)

)

DESCRIBE LOCK\_T

**(4)Remove primary key constraint on job\_id**

ALTER TABLE JOB DROP PRIMARY KEY CASCADE

****

**(5)Remove foreign key constraint on employee table**

ALTER TABLE EMPLOYEE DROP CONSTRAINT FK\_JOB\_ID

**PRACTICAL - 12**

**AIM:**  **Data Dictionary and E-R Diagram**

Suppose that as the database administrator (DBA) in a hotel, you have to set up a database to capture all the following information that the hotel needs to maintain.

• The hotel offers three types of ROOMS, including single room, double room, and triple room. Every room is Identified by its unique number.

• Every employee at the hotel is either a receptionist, a cleaning staff, or a kitchen staff. Each RECEP-TIONIST is identified with her/his name, employee number and years of experience. Receptionists are responsible for ensuring the room is clean before the room is assigned to the guest. Thus, they assign a single CLEANING STAFF to clean each room every morning and/or whenever it is required. Note that the same room may need to be cleaned several times on the same day, before it gets reassigned. For each cleaning assignment, the date and the status need to be provided. The KITCHEN STAFF is characterized by their specific responsibilities, e.g. being a cook or a waiter. The cleaning staff and the kitchen staff are also uniquely identified by their employee number.

• Receptionists welcome GUESTS and upon presentation of their valid traveling documents, they allocate a unique room to each guest and specify one group of facilities which is accessible to the guest during his stay. Guests are uniquely identified with their passport number but other necessary information are also recorded about the guests, including: name, phone numbers, arrival date, departure date, and credit card number. Each FACILITY GROUP contains specific set of facilities, e.g. the bar or gym, in order to be used by the guests. The arrival and departure dates of a guest will in turn determine the occupation of a specific room.

• A guest can be accompanied with one person to have a double room or at most two people for a triple room. Each ACCOMPANYING person is identified by his/her name.

**12.1: Design Data Dictionary for above problem.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity** | **Attributes** | **DataType** | **Length** | **Constraint** |
| **Room** | Room No.  Room Type | Number  Varchar2 | 5  20 | PrimaryKey  Not Null |
| **Guest** | Name  Details  Passport No.  Phone No. | Varchar2  Varchar2  Number  Number | 20  30  15  10 | Not Null  -  PrimaryKey  Not Null |
| **Facility** | Name | Varchar2 | 20 | Not Null |
| **Employee** | Employee No.  Name | Number  Varchar2 | 5  20 | Primary Key  Not Null |
| **Receptionist** | Experience | Number | 2 | Not Null |
| **Cleaning staff** | - | - | - | - |
| **Kitchen Staff** | Responsibilities | Varchar | 25 | Not Null |
| **Cleaning** | Status  Date | Number  Date | 2  - | Primary Key  Not Null |

**12.2:** Considering the descriptions given above, draw an ER diagram for the database, representing entities, attributes, and relationships.



**PRACTICAL - 13**

**AIM:** To perform basic PL/SQL blocks

**QUERIES:** Write a PL-SQL block to find Sum and average of three numbers.

DECLARE

A NUMBER:= 2;

B NUMBER:= 4;

C NUMBER:= 6;

SUMOF NUMBER;

AVGOF NUMBER;

BEGIN

SUMOF:=A+B+C;

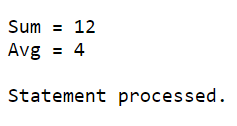
AVGOF:=(SUMOF)/3;

DBMS\_OUTPUT.PUT\_LINE('SUM = '||SUMOF);

DBMS\_OUTPUT.PUT\_LINE('AVG = '||AVGOF);

END;

**Output:**

****

**PRACTICAL - 14**

**AIM:** To perform the concept of loop

**QUERIES:** Find the factorial of a number in pl/sql using for, While and Simple   
LOOP.

DECLARE

NUM NUMBER := 5;

FACT NUMBER := 1;

TEMP NUMBER;

BEGIN

TEMP :=NUM;

WHILE( TEMP>0 )

LOOP

FACT := FACT\*TEMP;

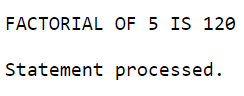
TEMP := TEMP-1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('FACTORIAL OF '|| NUM || ' IS ' || FACT);

END;

**OUTPUT:**



**PRACTICAL - 15**

**AIM:**

**To perform the concept of cursor**

**QUERIES:**

**(a) Display all the information of EMP table using %ROWTYPE.**

**(b) Create a PL/SQL block that does the following:**

**In a PL/SQL block, retrieve the name, salary, and MANAGER ID of the employees working in the particular department. Take Department Id from user.**

**If the salary of the employee is less than 1000 and if the manager ID is either 7902 or 7839, display the message <<last name>> Due for a raise. Otherwise, display the message <<last\_name>> Not due for a raise.**

DECLARE

D\_NO NUMBER :=:ENTER\_DEPT\_NO;

CURSOR C1 IS

SELECT \* FROM EMPLOYEE WHERE DEPT\_NO = D\_NO;

EMP\_RECORD C1%ROWTYPE;

BEGIN

OPEN C1;

LOOP

FETCH C1 INTO EMP\_RECORD;

EXIT WHEN C1%NOTFOUND;

IF(EMP\_RECORD.EMP\_SAL < 1000 AND (EMP\_RECORD.MANAGER\_ID = 105 OR EMP\_RECORD.MANAGER\_ID = 107)) THEN

DBMS\_OUTPUT.PUT\_LINE(EMP\_RECORD.L\_NAME || 'DUE FOR A RAISE ');

ELSE

DBMS\_OUTPUT.PUT\_LINE(EMP\_RECORD.L\_NAME || 'NOT DUE FOR A RAISE

');

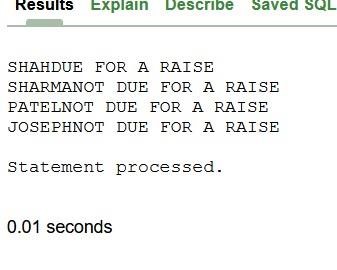
END IF;

END LOOP;

CLOSE C1;

END

**OUTPUT:**



**PRACTICAL - 16**

**AIM:**

**To perform the concept of trigger**

**QUERIES:**

**Write a PL/SQL block to update the salary where deptno is 10. Generate trigger that will store the original record in other table before updation take place.**

CREATE TABLE EMP\_NEW AS(SELECT \* FROM EMPLOYEE)

SELECT \* FROM EMP\_NEW;

CREATE OR REPLACE TRIGGER SALARY

BEFORE INSERT OR UPDATE OR DELETE ON EMP\_NEW

FOR EACH ROW

BEGIN

CASE

WHEN UPDATING THEN

DBMS\_OUTPUT.PUT\_LINE('UPDATING');

DBMS\_OUTPUT.PUT\_LINE('NEW SAL'||:NEW.EMP\_SAL);

WHEN INSERTING THEN

DBMS\_OUTPUT.PUT\_LINE('INSERTING');

WHEN DELETING THEN

DBMS\_OUTPUT.PUT\_LINE('DELETING');

END CASE;

END;

UPDATE EMP\_NEW SET EMP\_SAL=2000 WHERE DEPT\_NO=10;

**OUTPUT:**



**PRACTICAL - 17**

**AIM:** To solve queries using the concept of View.

**QUERIES:**

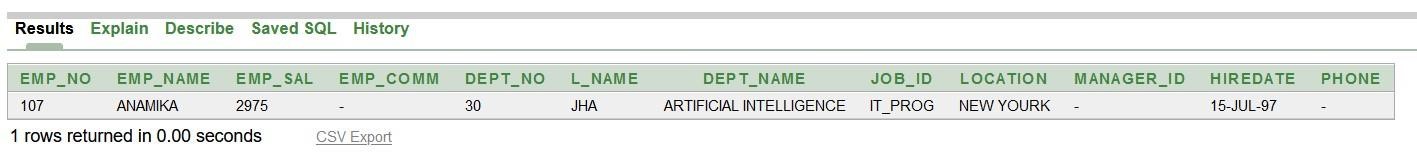
**(1) Write a query to create a view for those employee belongs to the location New York.**

CREATE VIEW EMP\_NEWYORK1

AS SELECT \* FROM EMPLOYEE

WHERE LOCATION = 'NEW YOURK'

SELECT \* FROM EMP\_NEWYORK1

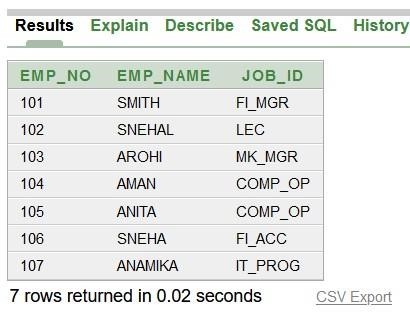


**(2) Write a query to create a view for all employee with columns emp\_id, emp\_name, and job\_id.**

CREATE VIEW EMP\_HALF

AS SELECT EMP\_NO, EMP\_NAME, JOB\_ID FROM EMPLOYEE

SELECT \* FROM EMP\_HALF



**(3)** **Write a query to find the salesmen of the location New York who having salary more than 3000.**

CREATE VIEW EMP\_Q3

AS SELECT \* FROM EMPLOYEE

WHERE LOCATION = 'NEW YOURK' AND EMP\_SAL > 3000

SELECT \* FROM EMP\_Q3



**(4)** **Write a query to create a view to getting a count of how many employee we have at each department**

CREATE VIEW EMP\_Q4

AS SELECT COUNT(EMP\_NO) AS "NUM\_OF\_EMP" FROM EMPLOYEE

SELECT \* FROM EMP\_Q4



**PRACTICAL - 18**

**AIM:**

**To perform the concept of function and procedure Write a PL/SQL block to update the salary of employee specified by empid. If record exist, then update the salary otherwise display appropriate message. Write a function as well as procedure for updating salary.**

**QUERIES:**

CREATE OR REPLACE FUNCTION SALARY(S1 NUMBER)

RETURN NUMBER IS

SAL NUMBER;

BEGIN

SAL := S1;

SAL := SAL +(SAL\*0.1);

RETURN SAL;

END;

DECLARE

E\_NO NUMBER :=:ENTER\_NO;

1. NUMBER;
2. NUMBER;

BEGIN

SELECT EMP\_SAL INTO A FROM EMPLOYEE WHERE EMP\_NO = E\_NO;

B := SALARY(A);

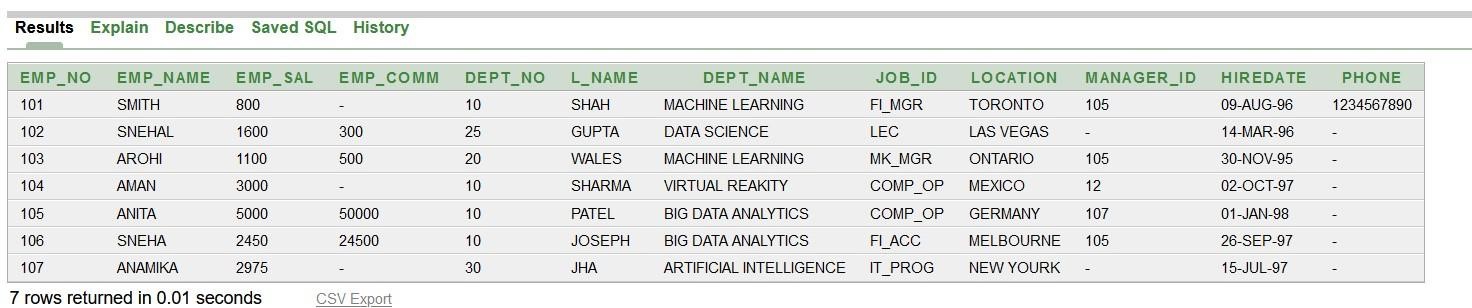
DBMS\_OUTPUT.PUT\_LINE('UPDATED SALARY IS: ' || B);

UPDATE EMPLOYEE SET EMP\_SAL = B WHERE EMP\_NO = E\_NO;

END;

SELECT \* FROM EMPLOYEE;

**OUTPUT:**



**PRACTICAL - 19**

**AIM:**

**To perform the concept of package**

**QUERIES:**

**Create and invoke a package that contains private and public constructs**

CREATE OR REPLACE PACKAGE PKG\_EMP IS

PROCEDURE UPDATE\_RECORD(D\_NO EMPLOYEE.DEPT\_NO%TYPE);

PROCEDURE DELETE\_RECORD(D\_NO EMPLOYEE.DEPT\_NO%TYPE);

END;

CREATE OR REPLACE PACKAGE BODY PKG\_EMP IS

PROCEDURE UPDATE\_RECORD(D\_NO EMPLOYEE.DEPT\_NO%TYPE) AS

D\_NAME VARCHAR2(30);

D\_NO1 NUMBER;

BEGIN

D\_NO1 := D\_NO;

UPDATE EMPLOYEE SET DEPT\_NAME = 'ML' WHERE DEPT\_NO = D\_NO1;

SELECT DEPT\_NAME INTO D\_NAME FROM EMPLOYEE WHERE DEPT\_NO = D\_NO1;

END;

PROCEDURE DELETE\_RECORD(D\_NO EMPLOYEE.DEPT\_NO%TYPE) AS

D\_NAME VARCHAR2(30);

D\_NO1 NUMBER;

BEGIN

D\_NO1 := D\_NO;

DELETE FROM EMPLOYEE WHERE D\_NO = 25;

END;

END;

DECLARE

D\_NO EMPLOYEE.DEPT\_NO%TYPE;

BEGIN

D\_NO := 25;

PKG\_EMP.UPDATE\_RECORD(D\_NO);

PKG\_EMP.DELETE\_RECORD(D\_NO);

END

**OUTPUT:**

