Practical -1

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

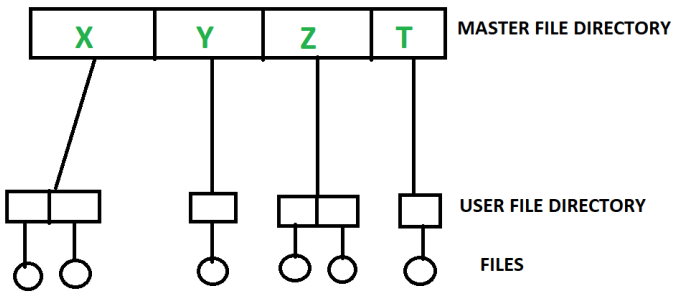
THEORY:

1. **File System:**

File system is basically a way of arranging the files in a storage medium like hard disk. File system organizes the files and helps in retrieval of files when they are required. File systems consists of different files which are grouped into directories. The directories further contain other folders and files. File system performs basic operations like management, file naming, giving access rules etc.

Example:

NTFS (New Technology File System), EXT(Extended File System).

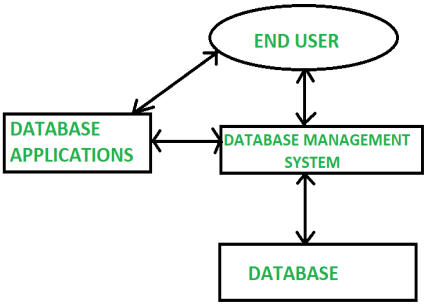


2. DBMS (Database Management System) :

Database Management System is basically a software that manages the collection of related data. It is used for storing data and retrieving the data effectively when it is needed. It also provides proper security measures for protecting the data from unauthorized access. In Database Management System the data can be fetched by SQL queries and relational algebra. It also provides mechanisms for data recovery and data backup.

Example:

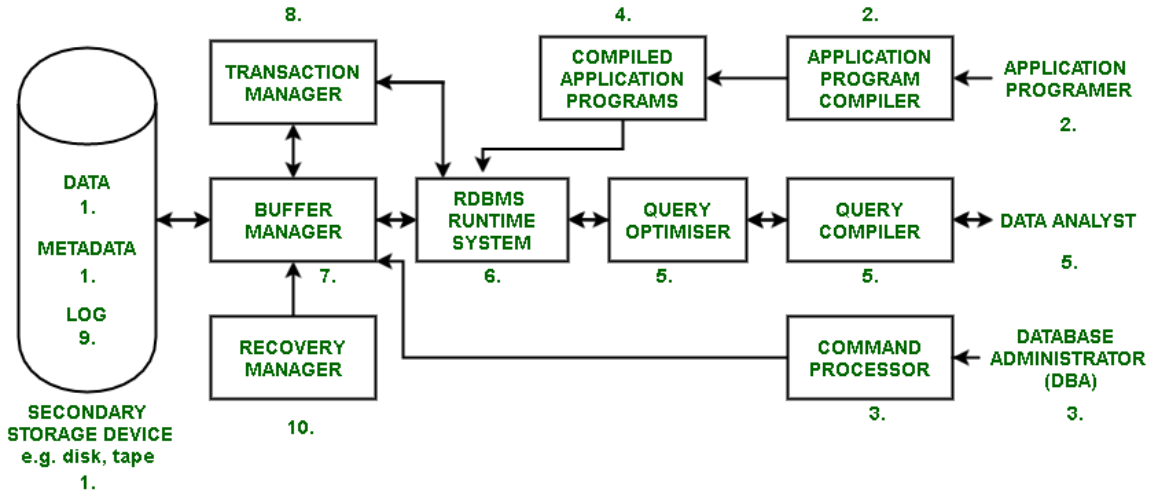
Oracle, MySQL, MS SQL server.



3. RDBMS:

RDBMS stands for Relational Database Management System and it implements SQL. In the real-world scenario, people use the Relational Database Management System to collect information and process it, to provide service. E.g. In a ticket processing system, details about us (e.g. age, gender) and our journey (e.g. source, destination), are collected, and the ticket is provided to us.

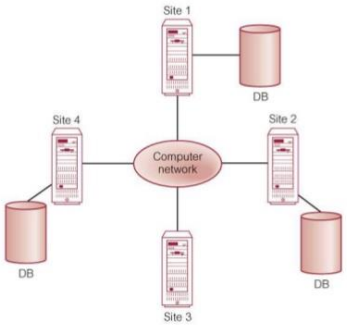
RDBMS Architecture:



4. DDBMS

A distributed database is basically a database that is not limited to one system, it is spread over different sites, i.e, on multiple computers or over a network of computers. A distributed database system is located on various sited that don't share physical components. This maybe required when a particular database needs to be accessed by various users globally. It needs to be managed such that for the users it looks like one single database.

Distributed DBMS



Practical -2

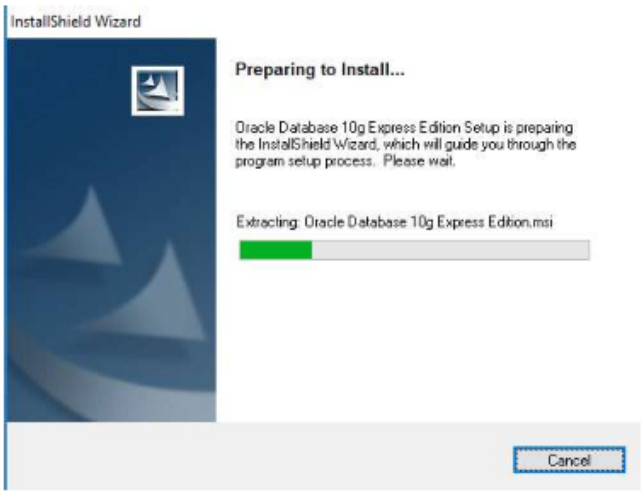
Aim: Introduction to Oracle (step by step installation, introduction of sql, plsql).

THEORY:

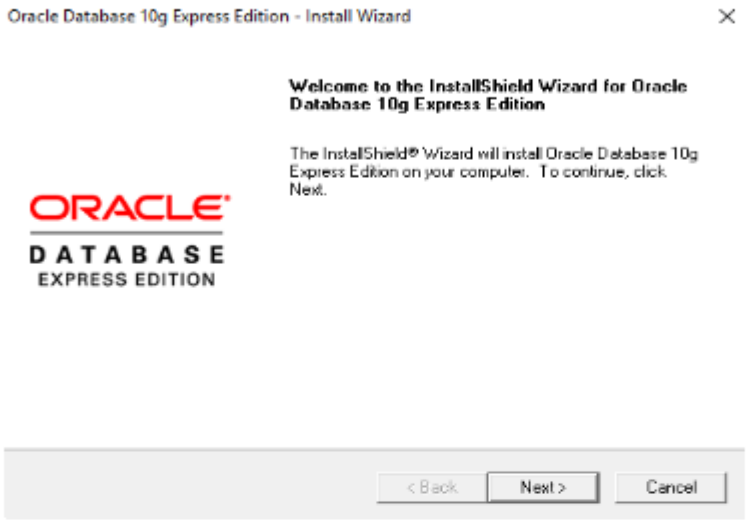
Guide to Install Oracle 10g

1) Download Oracle 10g from below link:
https://drive.google.com/file/d/1Y6ghDOEfVorTNrzWgF1UKaENHjeGgrHG/view

2) Install it by double clicking .exe which you have downloaded

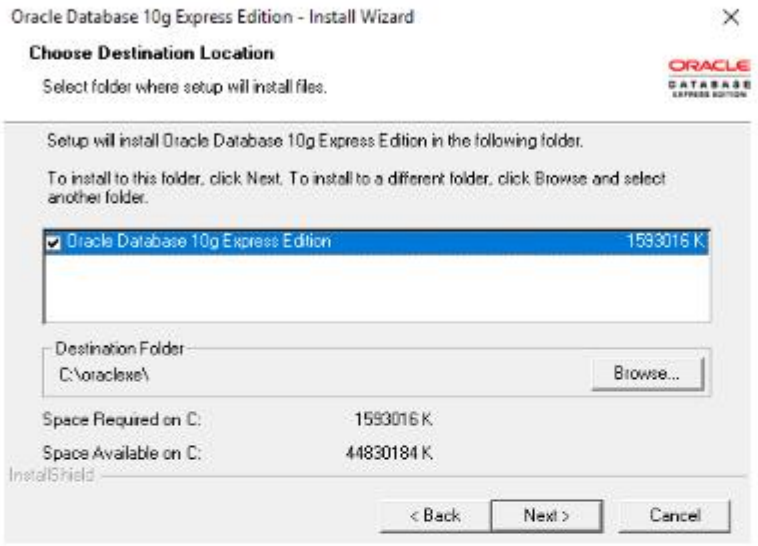


3) Click on Next button

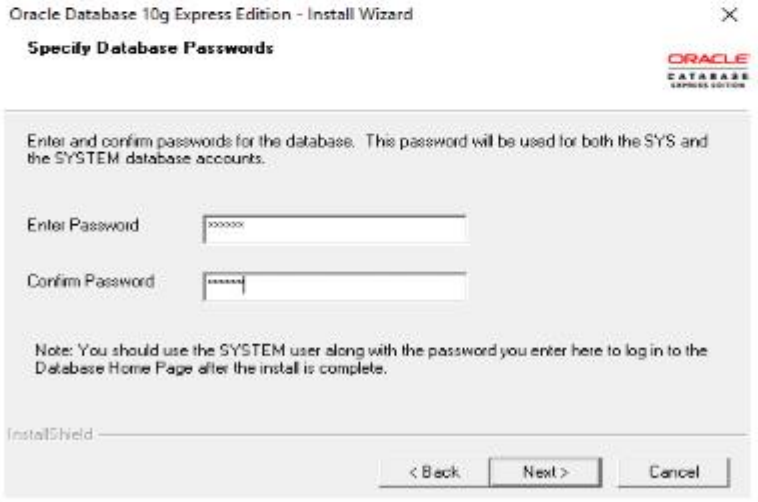


4) Accept license agreement and click on next button

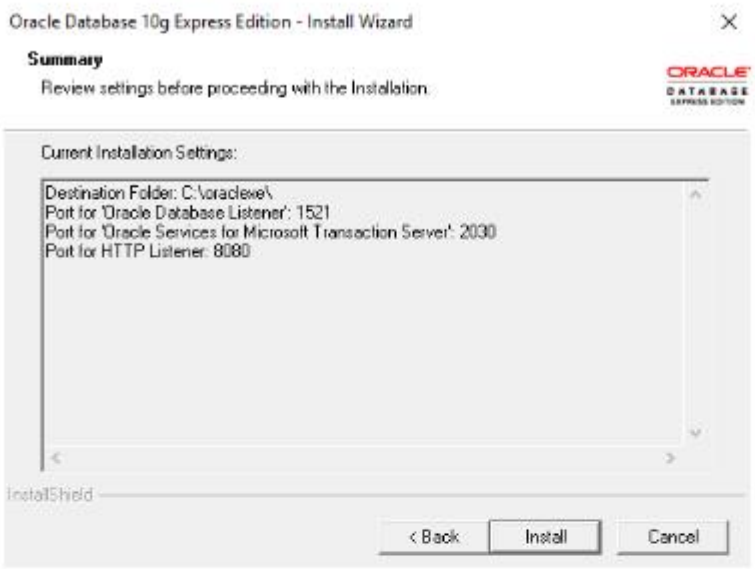
5) Click on next button



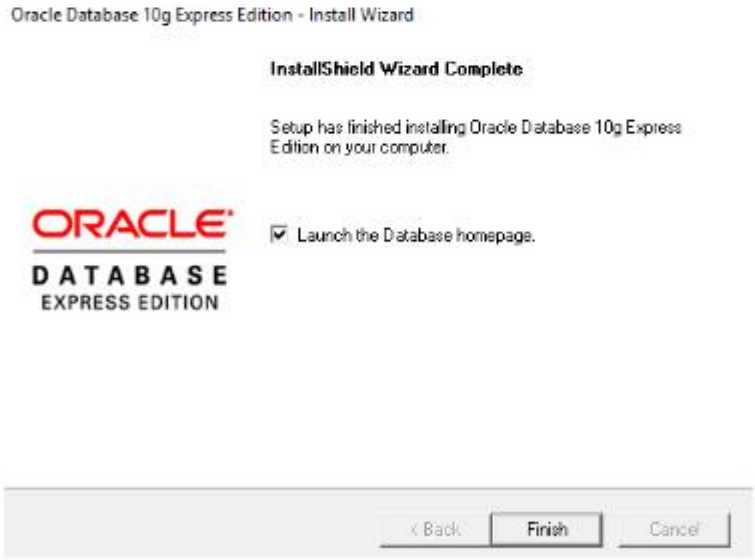
6) Enter password and confirm password for SYS and SYSTEM user. Please remember it because once installation will be over you have to enter it. To make it easy to remember give password as : "oracle"



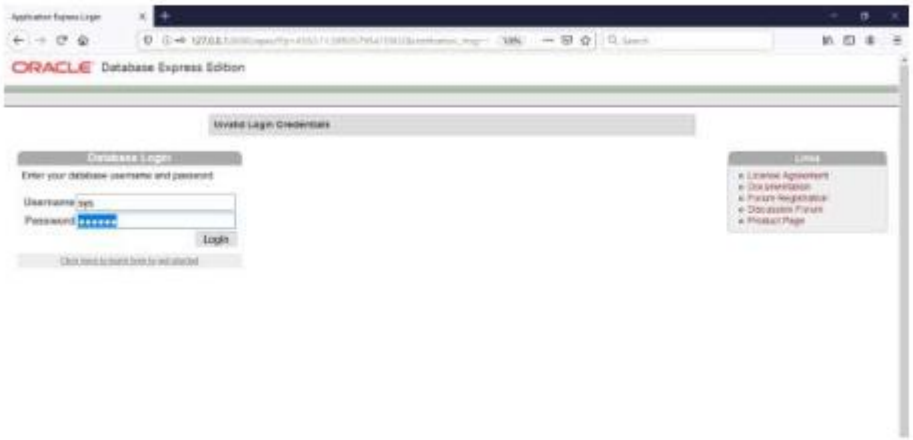
7) Click on install button



8) Click on finish button.



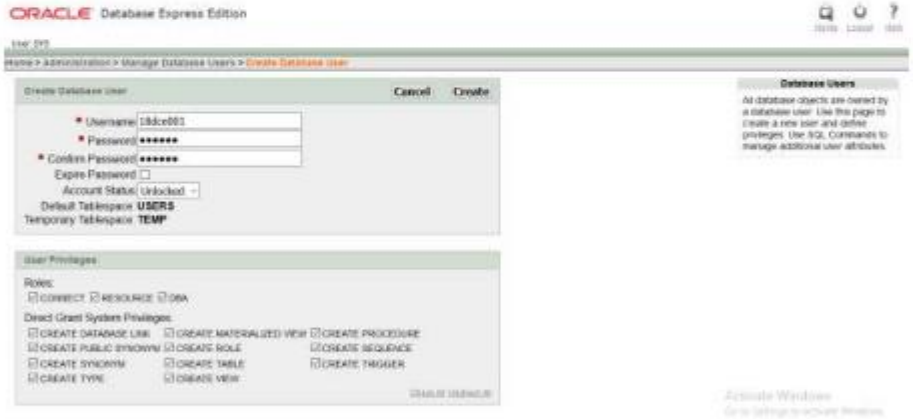
9) Enter username as SYS OR SYSTEM and enter your password (Entered in step: 6)



10) Click on Administration

11) Now click on “database user drop down button”. From that click on “create user”.

12) 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.



13) This page will be shown to you. Now click on "logout" button.

14) Click on login

15) Enter username and password that you just created and click on “login” button

16) Click on SQL

17) Click on SQL Commands

18) Congratulation!!! Now you are ready to code SQL and PLSQL.

19) Thank You!!!

Introduction

SQL:
Structured Query Language (SQL) is a standard Database language which is used to create, maintain and retrieve the relational database.

Introduction to PL/SQL:

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. PL/SQL stands for "Procedural language extensions to SQL." PL/SQL is a database-oriented programming language that extends SQL with procedural capabilities. It was developed by Oracle Corporation within the early 90's to boost the capabilities of SQL.

PL/SQL adds selective (i.e. if…then…else…) and iterative constructs (ie. loops) to SQL. PL/SQL is most helpful to put in writing triggers and keep procedures. Stored procedures square measure units of procedural code keep during a compiled type inside the info.

Conclusion: From this practical I learned how to install oracle & learned about sql & pl/sql.

Practical – 3

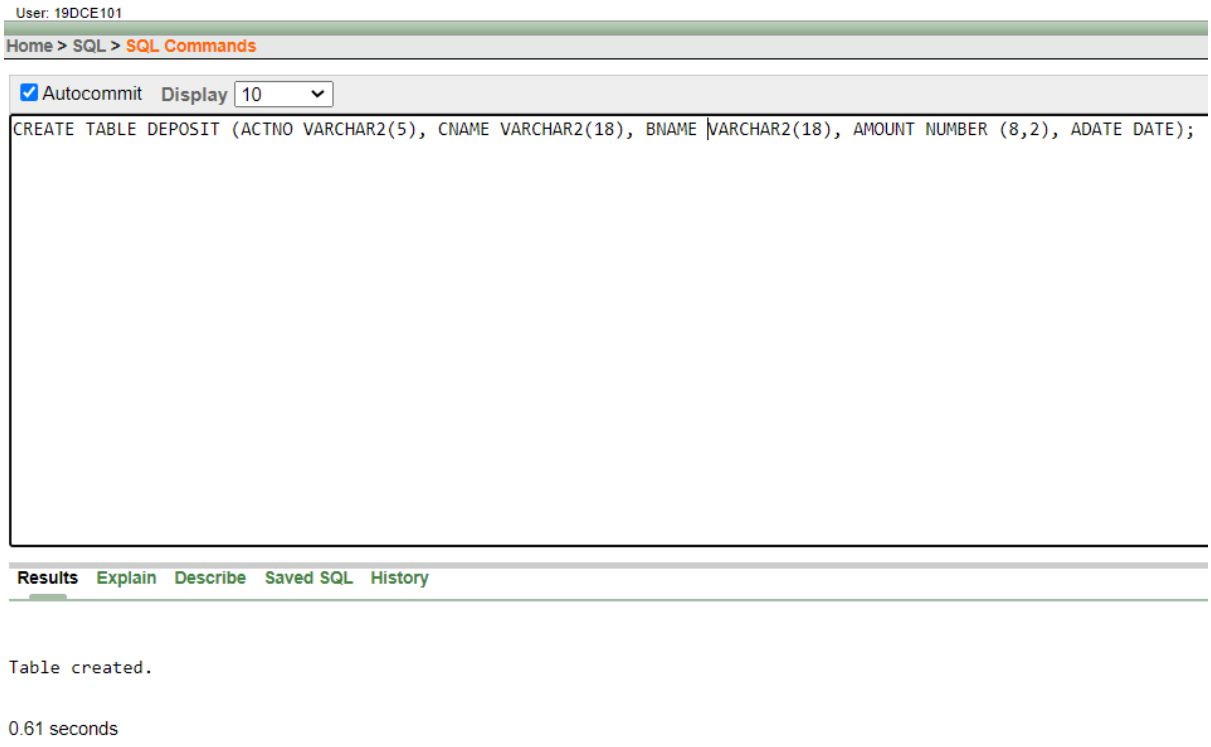
Aim: (i) To study DDL-CREATE and DML-INSERT Commands.

CREATE:

Syntax:

```
CREATE TABLE <TABLE_NAME> (
        FIELD_NAME <DATA_TYPE>,
);
```

Snapshot:

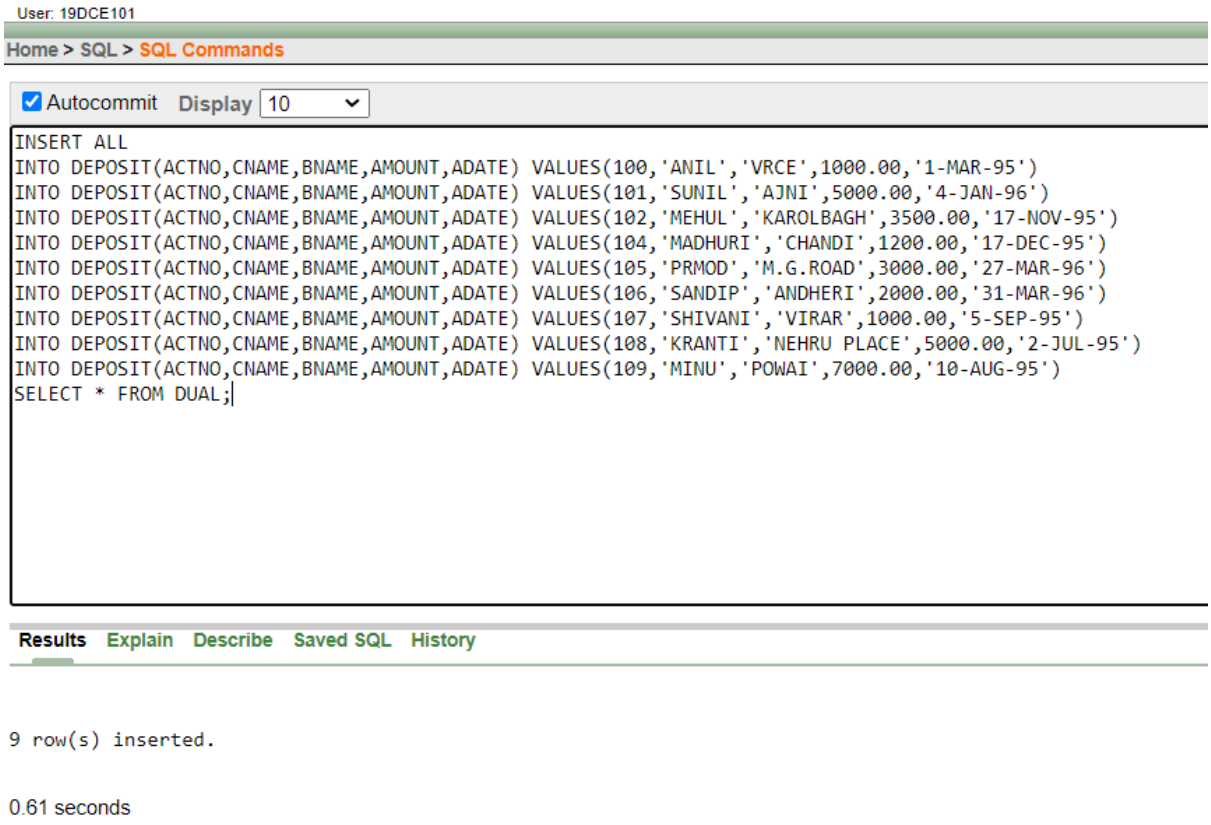


INSERT:

Syntax:

INSERT INTO <TABLE_NAME>
VALUES (‘val-1’, ‘val-2’,….);

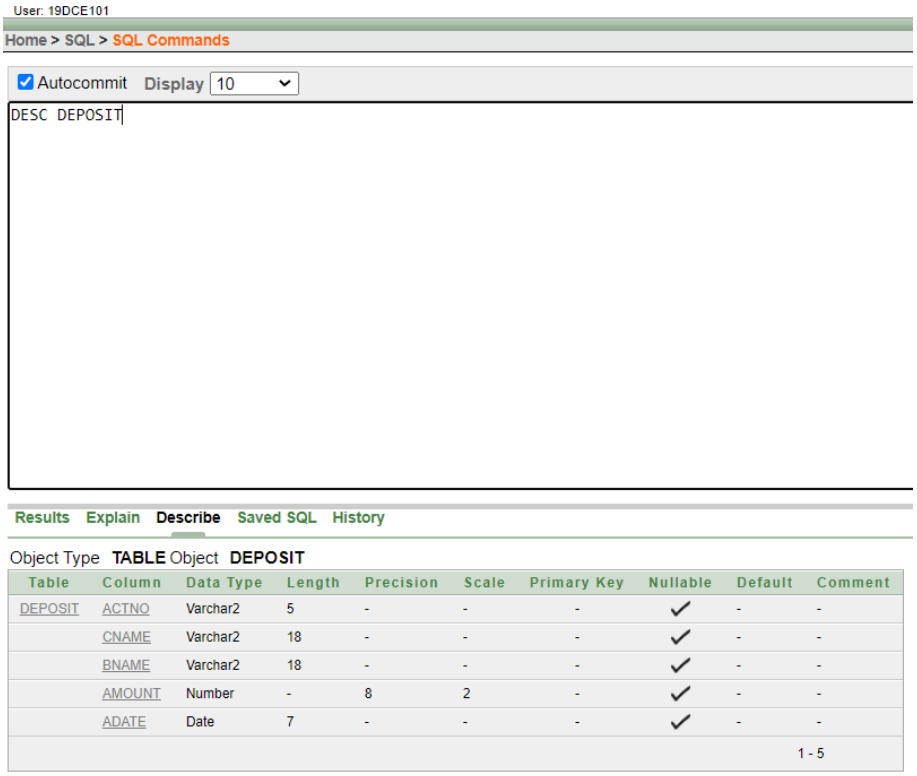
Snapshot:

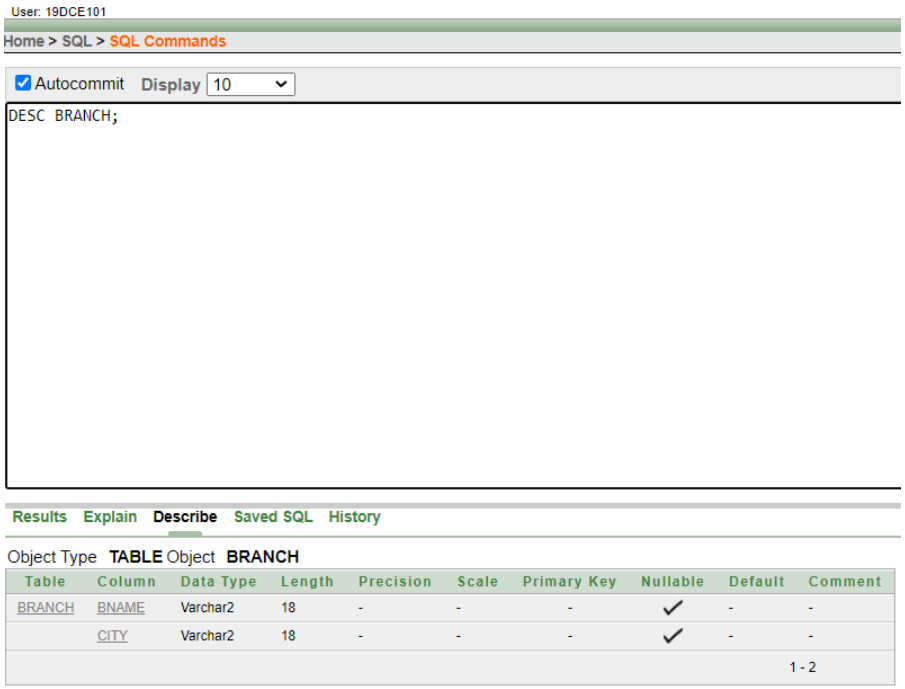


From the above given tables perform the following queries:

(1) Describe deposit, branch.

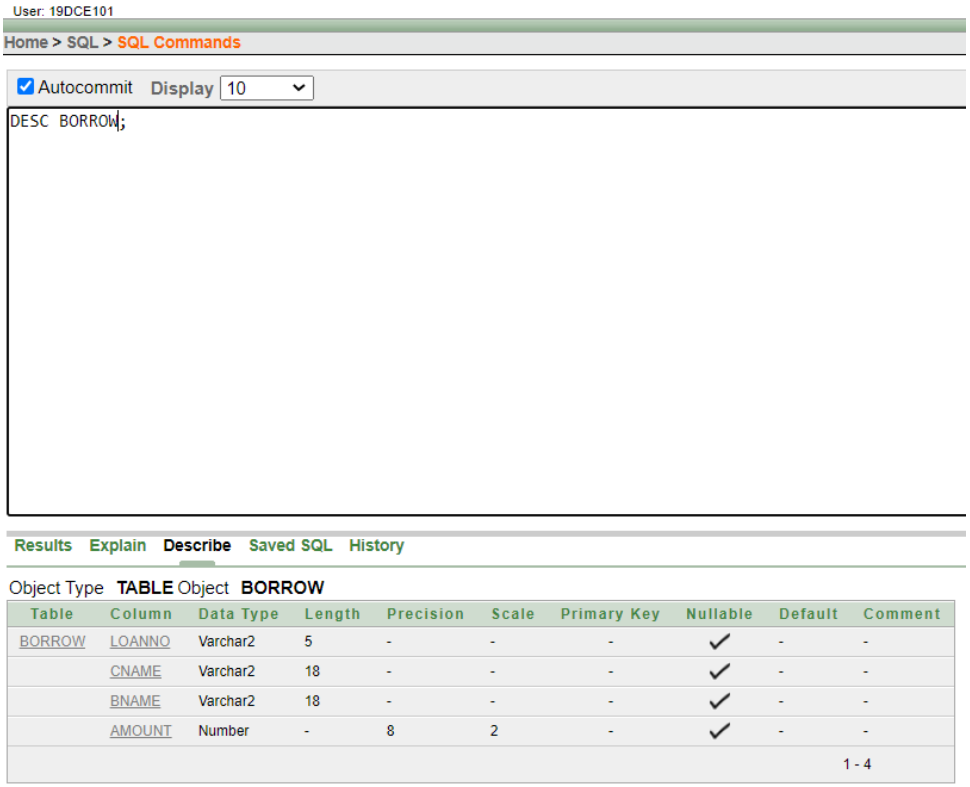
Snapshot:

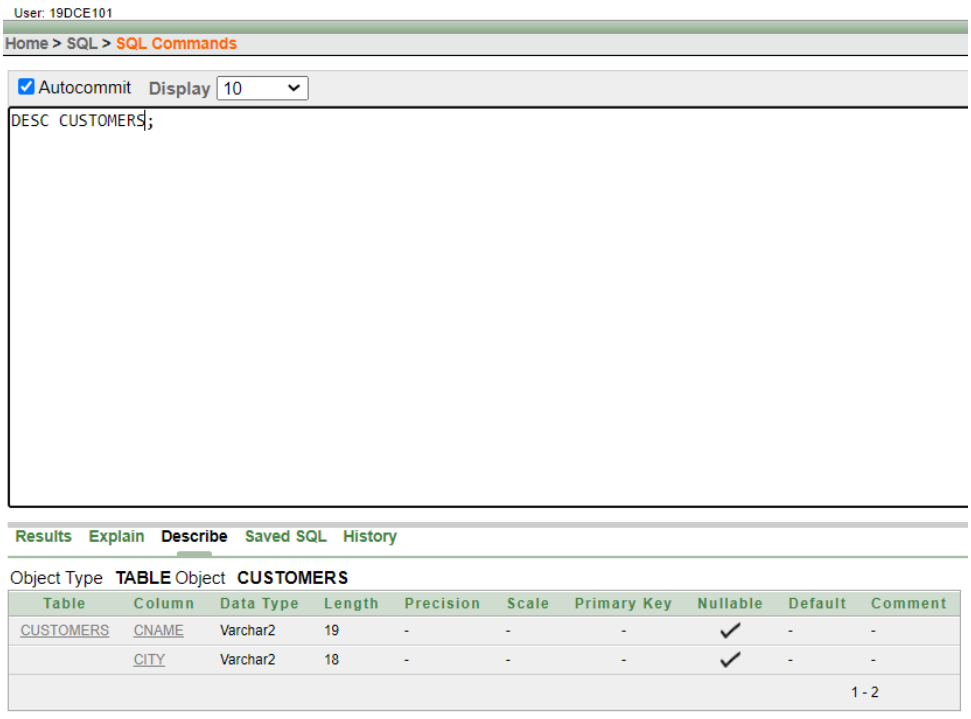




(2) Describe borrow, customers.

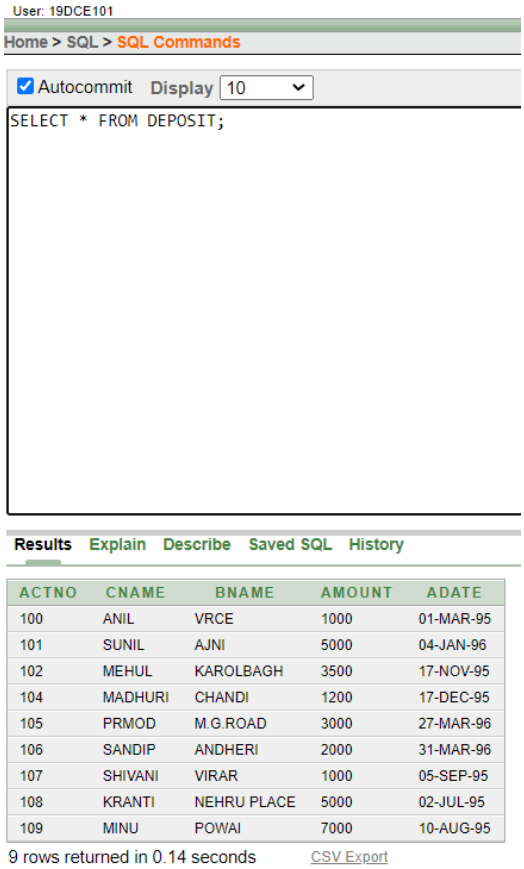
Snapshot:





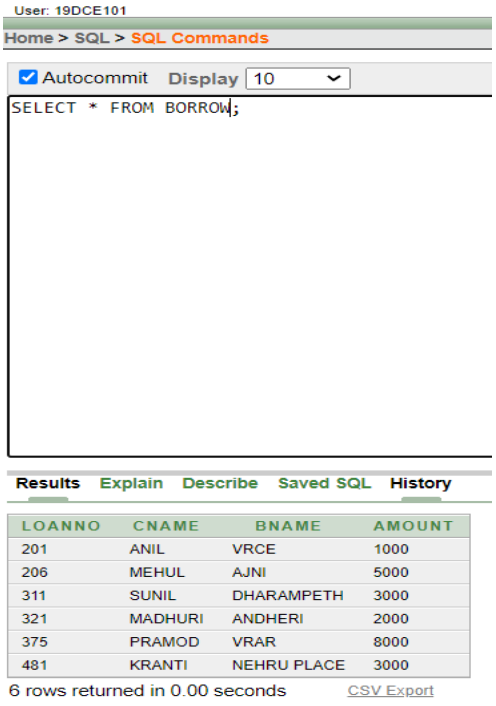
(3) List all data from table DEPOSIT.

Snapshot:



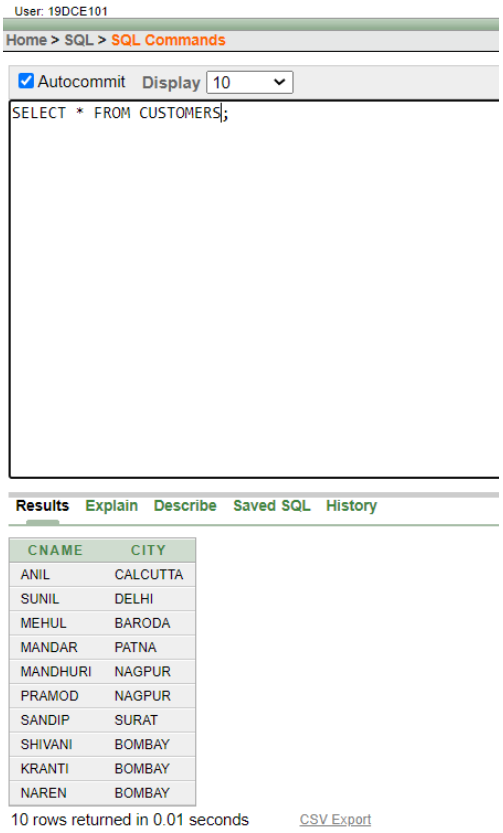
(4) List all data from table BORROW.

Snapshot:



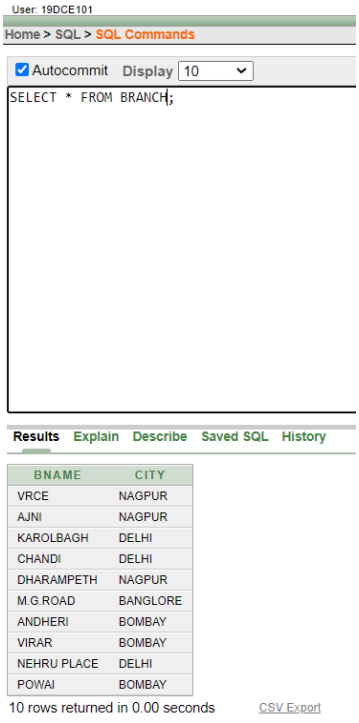
(5) List all data from table CUSTOMERS.

Snapshot:



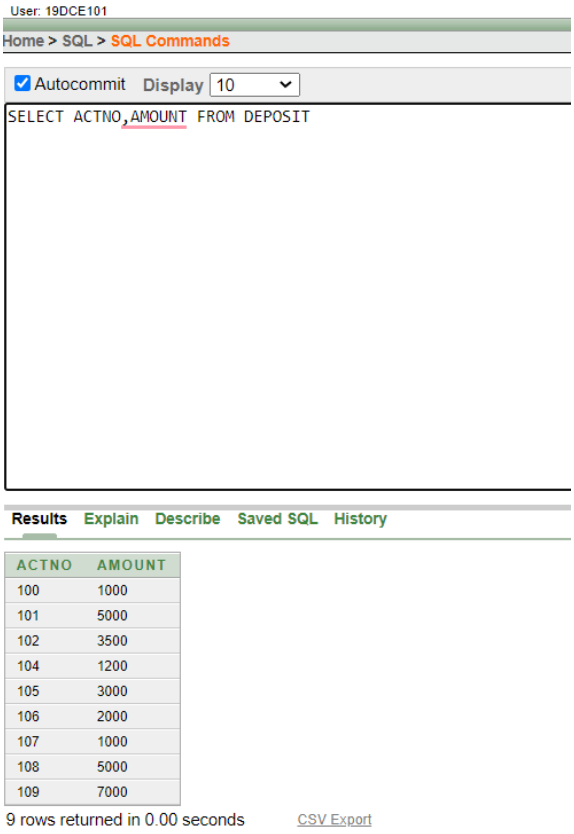
(6) List all data from table BRANCH.

Snapshot:



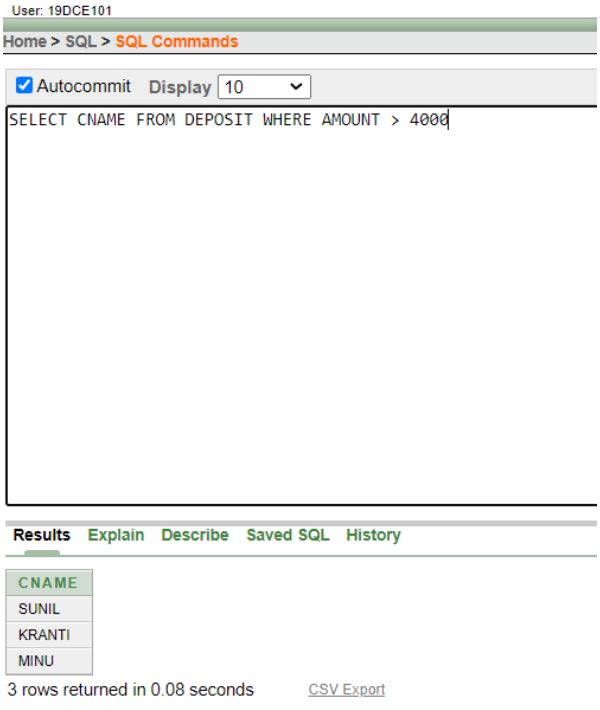
(7) Give account no and amount of depositors.

Snapshot:



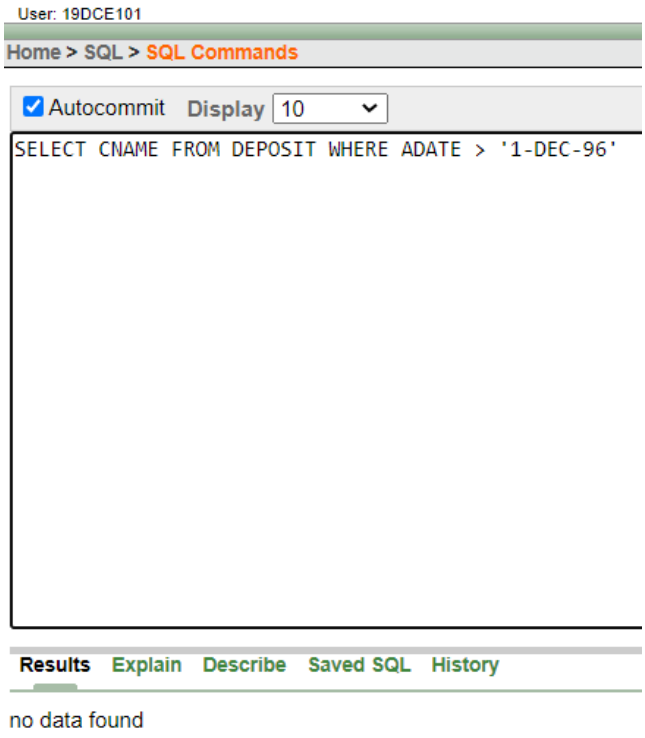
(8) Give name of depositors having amount greater than 4000.

Snapshot:



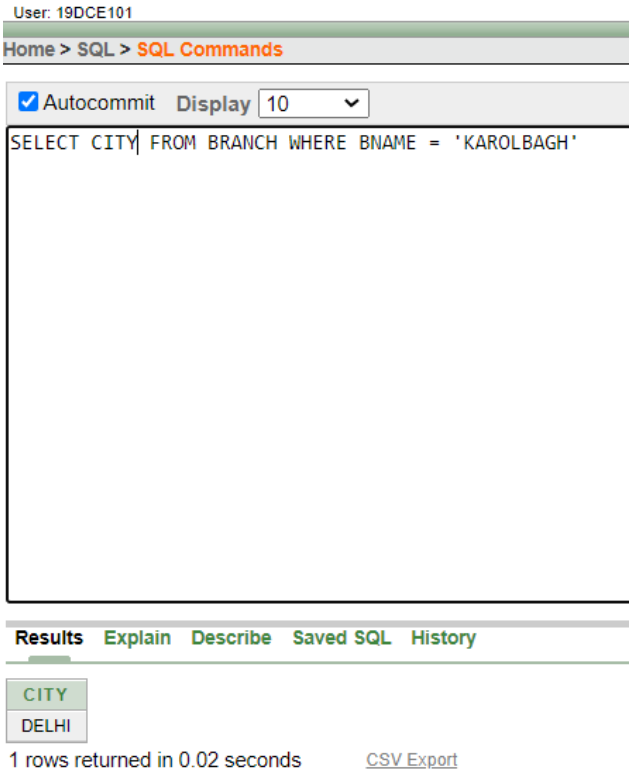
(9) Give name of customers who opened account after date '1-12-96'.

Snapshot:



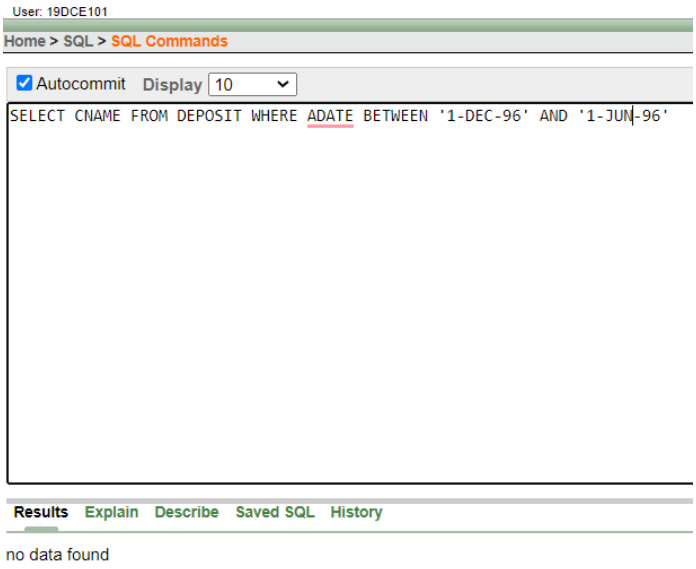
(10) Give name of city where branch karolbagh is located.

Snapshot:



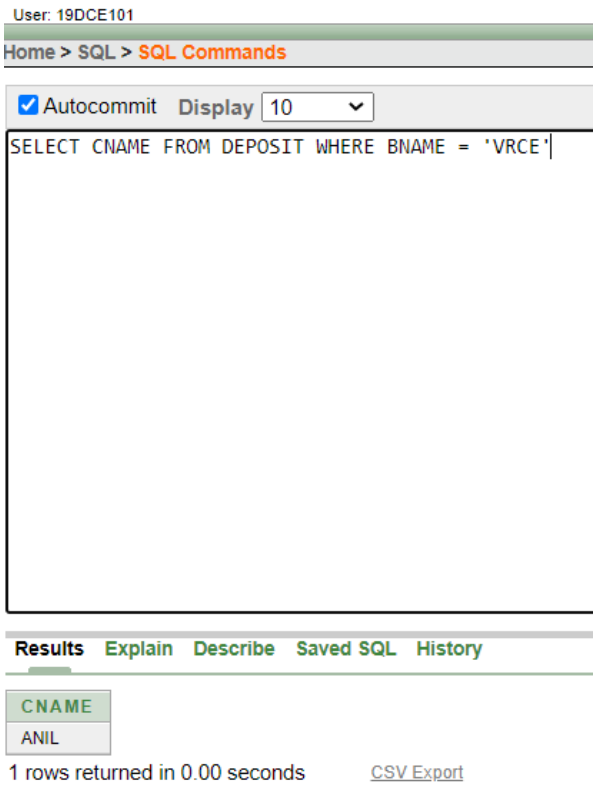
(11) Give account no and amount of customer having account opened between date 1-12-96 and 1-6-96.

Snapshot:



(12) Give names of depositors having account at VRCE.

Snapshot:



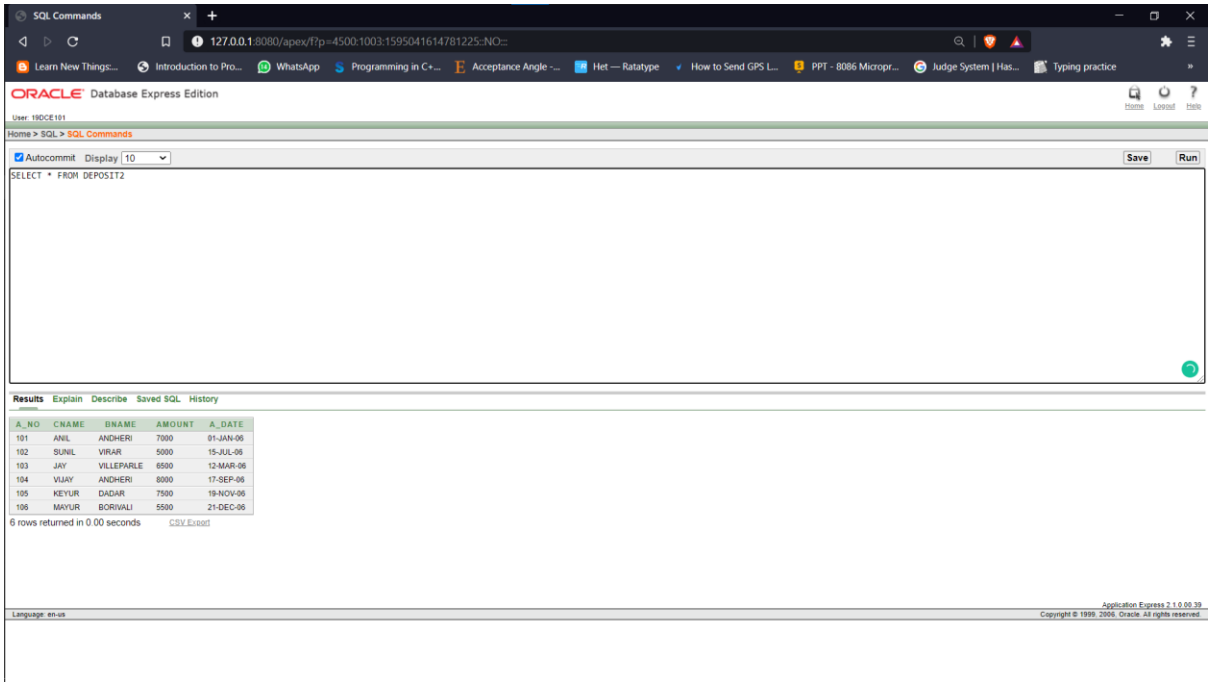
Conclusion: We Learned how to create tables, insert data into tables & read from the table as any required query.

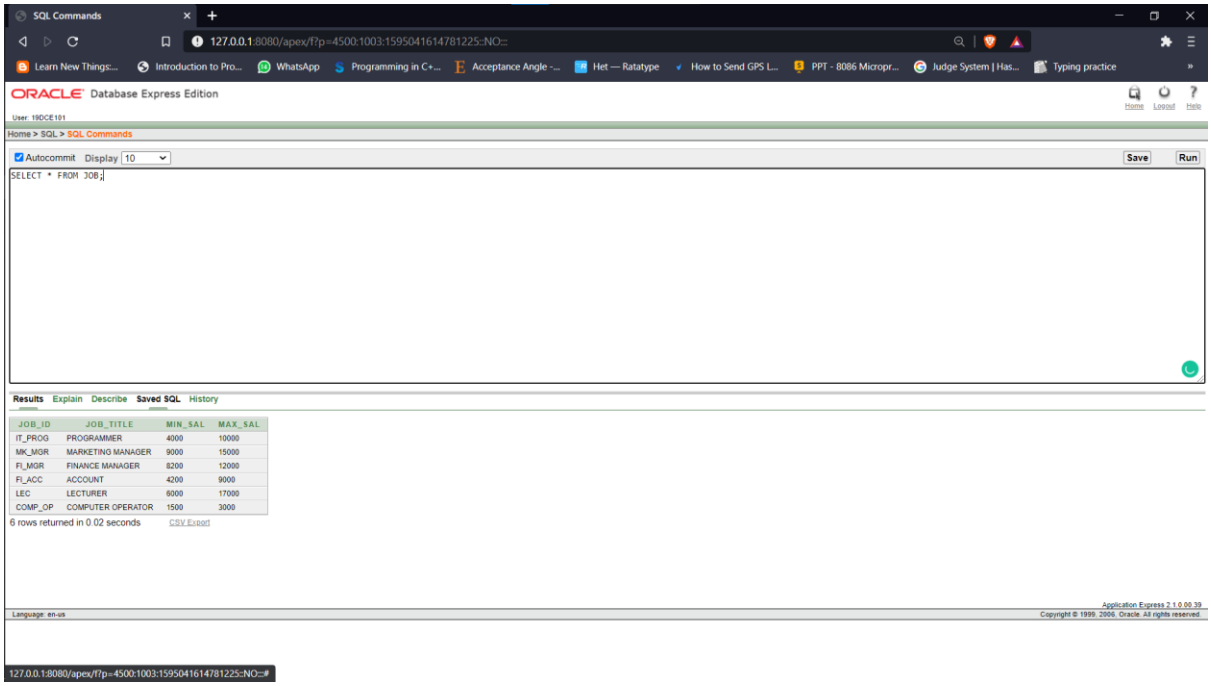
Practical -4

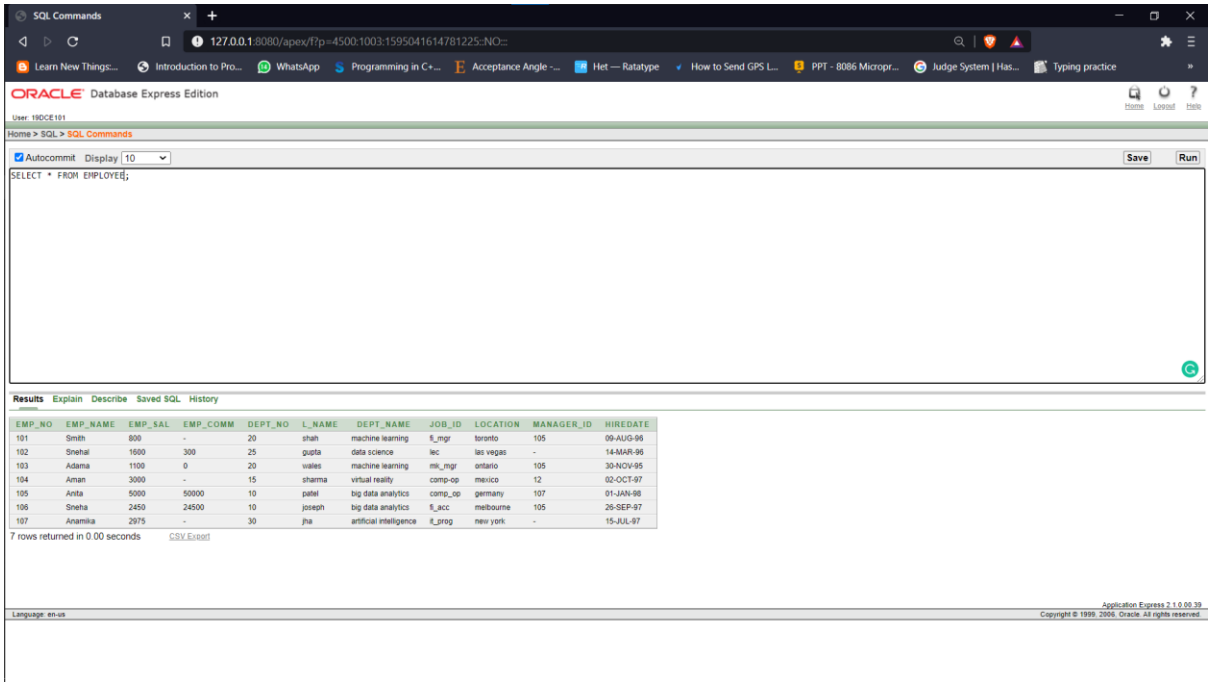
Perform following queries

(1) Retrieve all data from employee, jobs and deposit.

Snap-Shot:

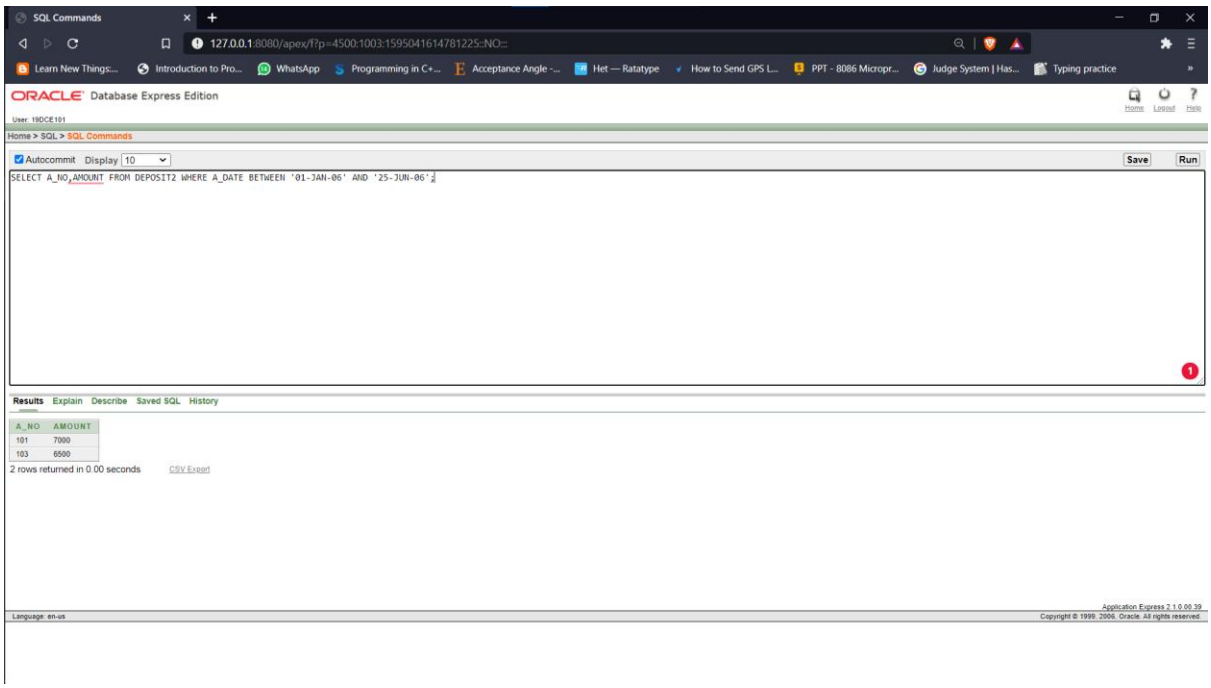






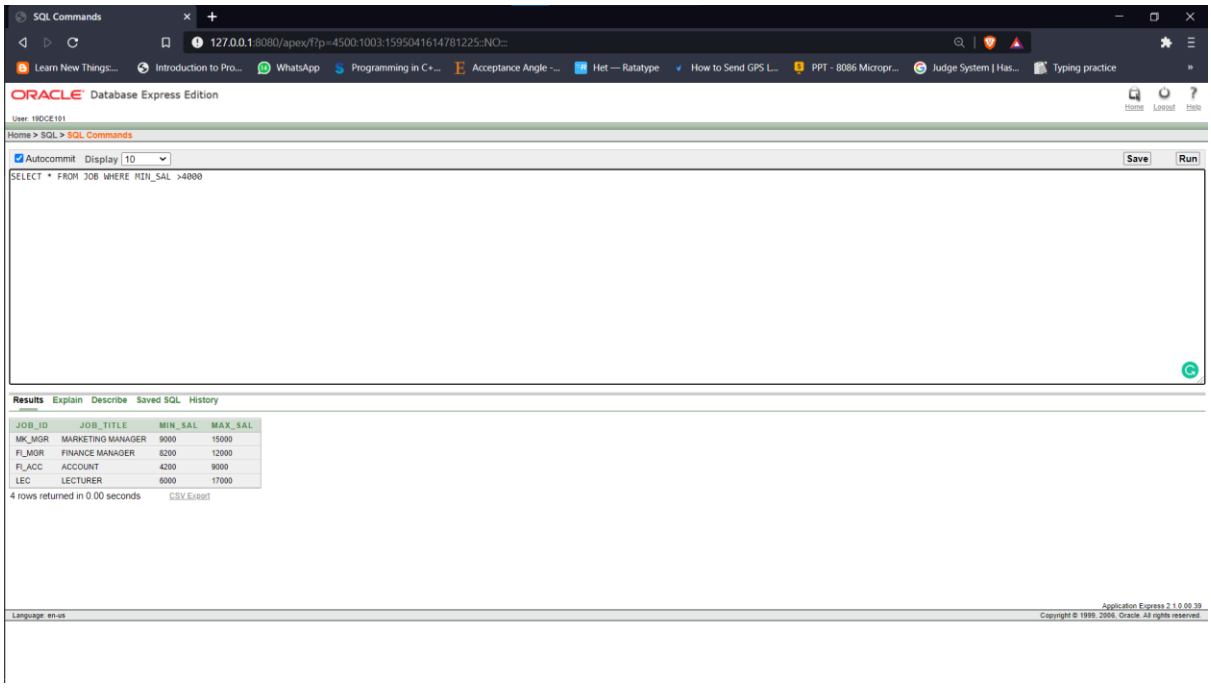
(2) Give details of account no. and deposited rupees of customers having account opened between dates 01-01-06 and 25-07-06.

Snap-Shot:



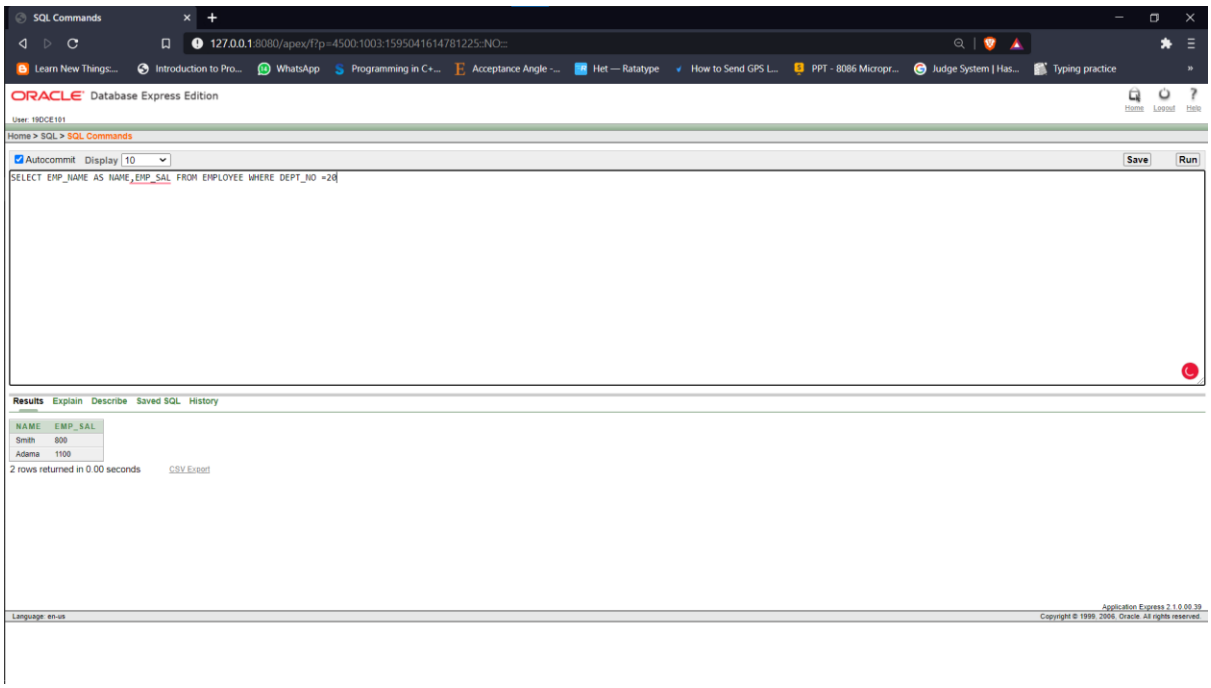
(3) Display all jobs with minimum salary is greater than 4000.

Snap-Shot:



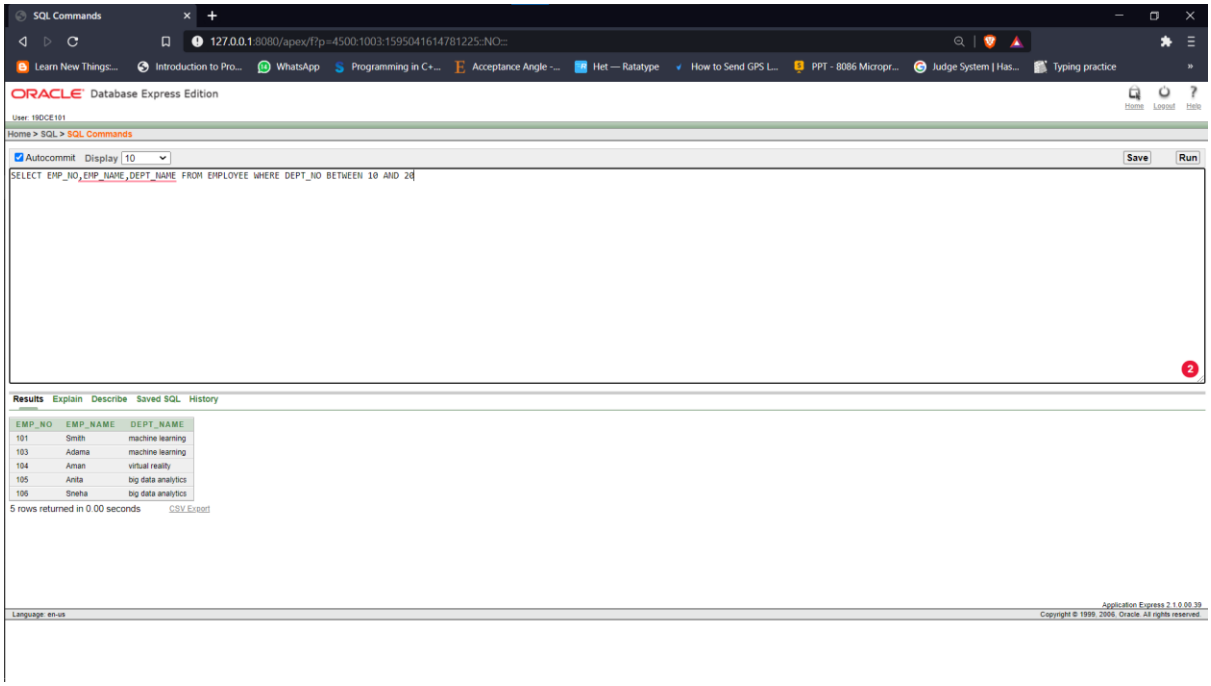
(4) Display name and salary of employee whose department no is 20. Give alias name to name of employee.

Snap-Shot:



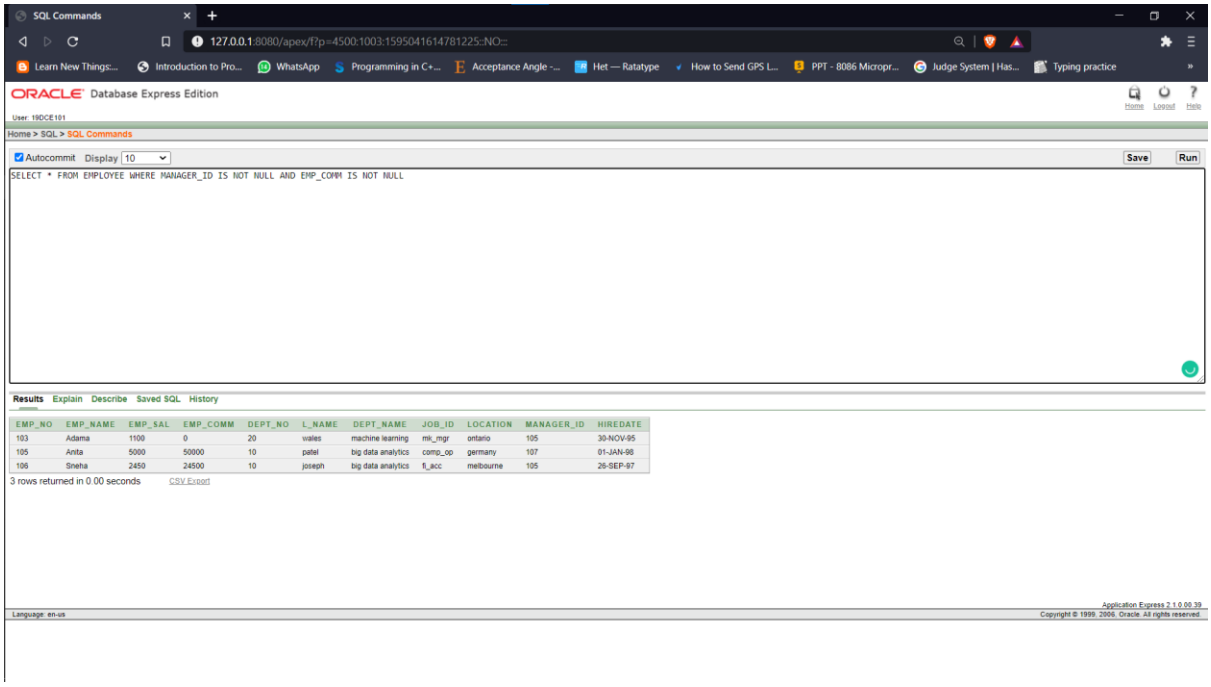
(5) Display employee no, name and department details of those employee whose department lies in (10,20).

Snap-Shot:



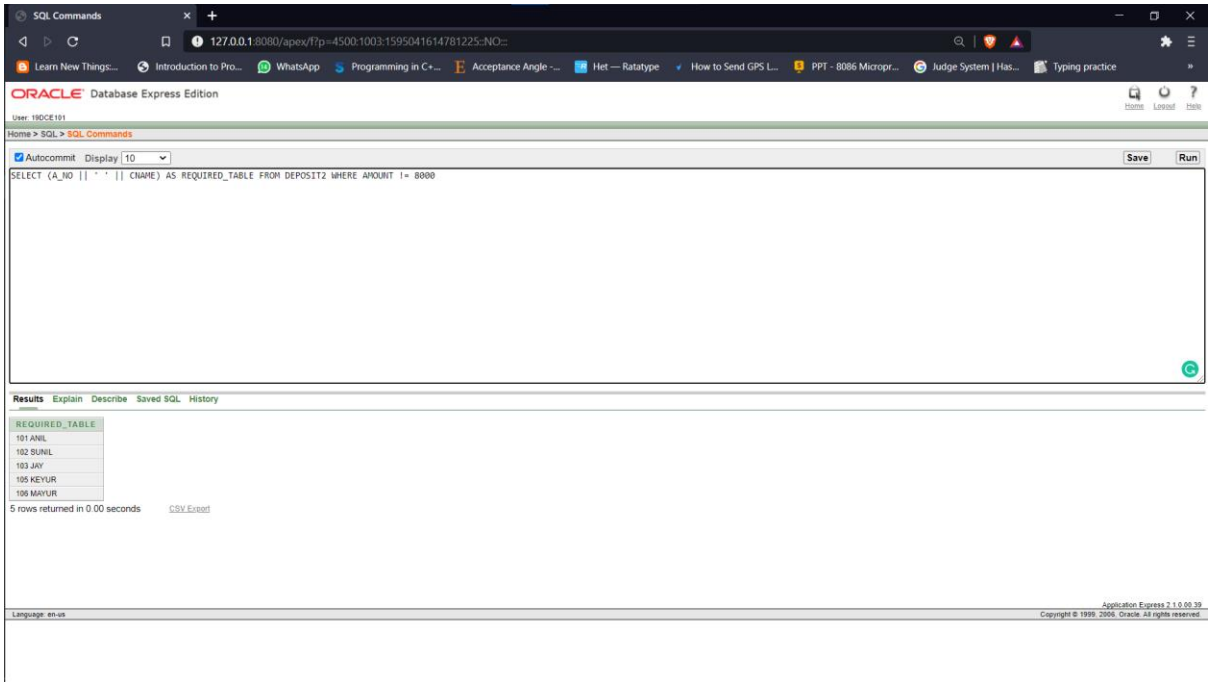
(6) Display the non-null values of employees.

Snap-Shot:



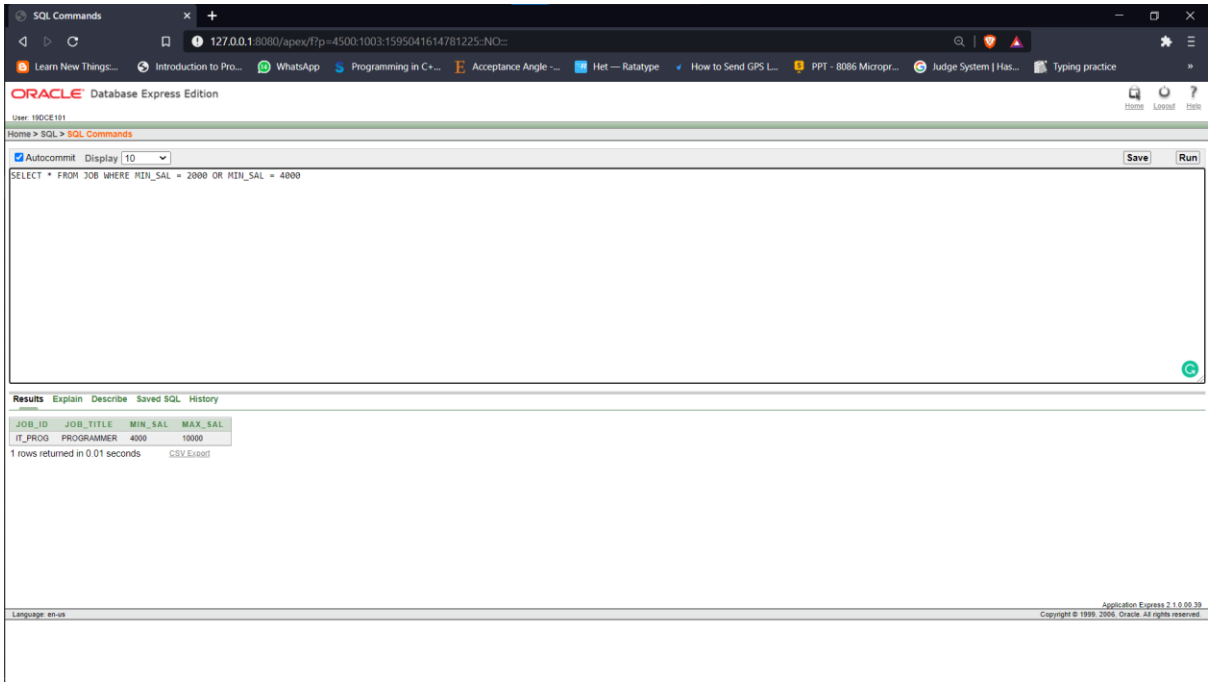
(7) Display name of customer along with its account no (both columns should be displayed as one) whose amount is not equal to 8000 Rs.

Snap-Shot:



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

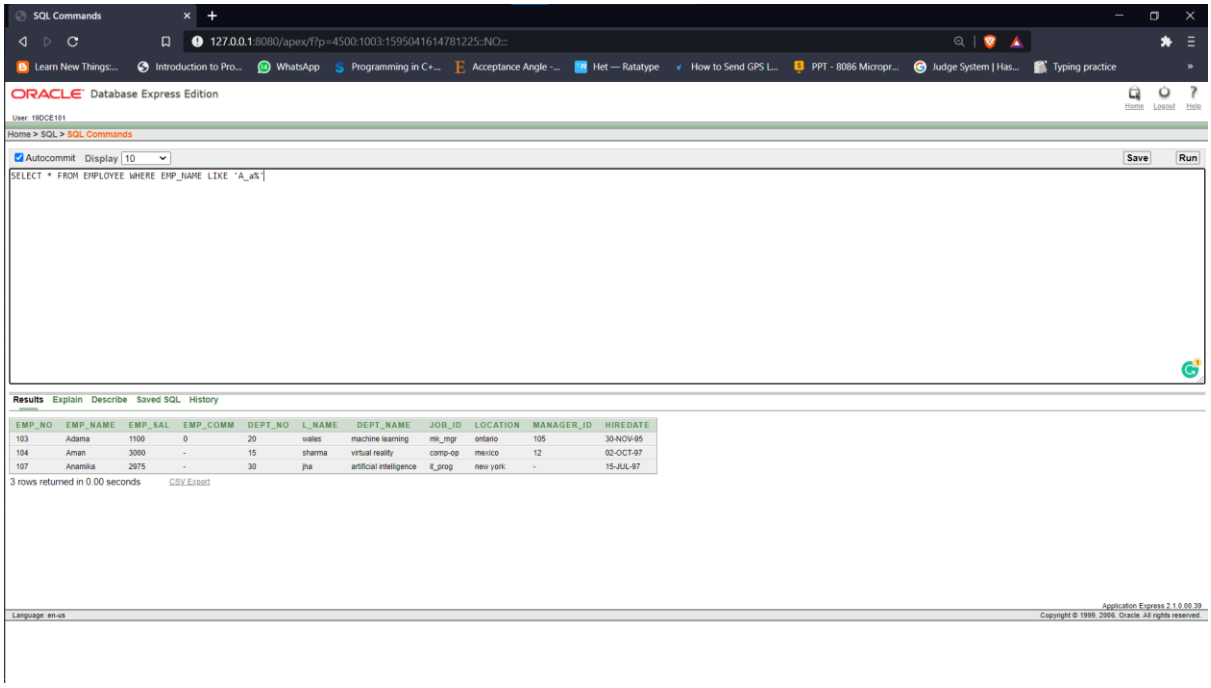
Snap-Shot:



To study various options of LIKE predicate

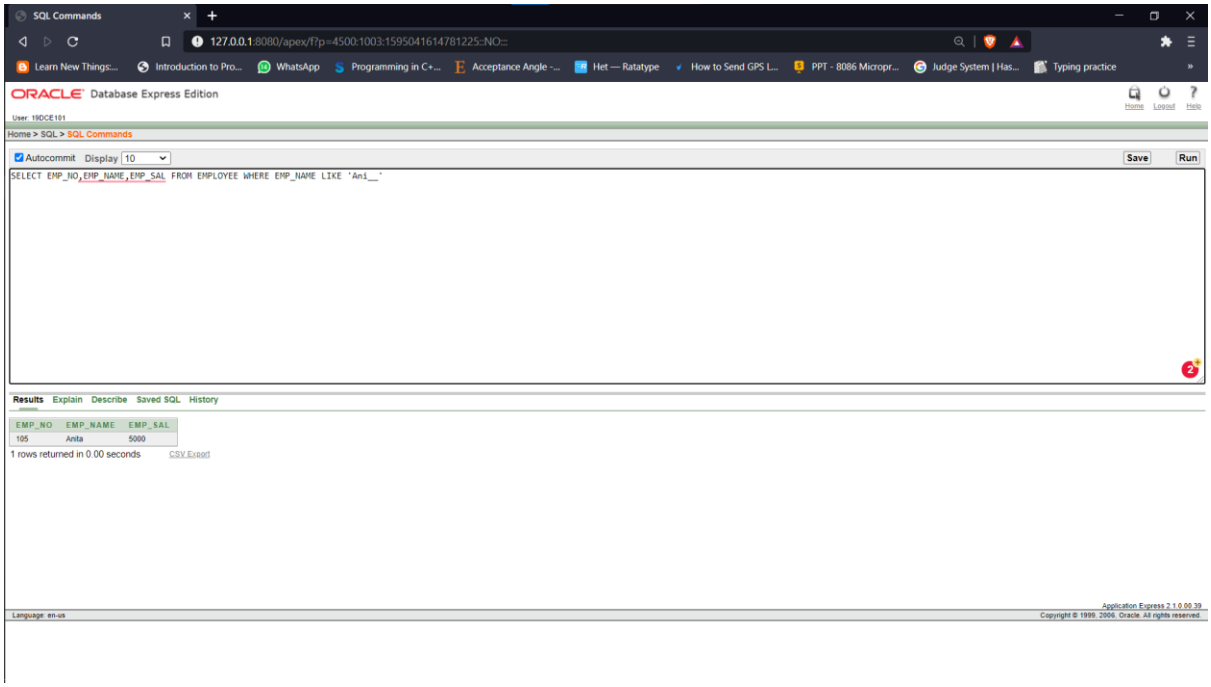
(1) Display all employee whose name start with ‘A’ and third character is ‘‘a’.

Snap-Shot:



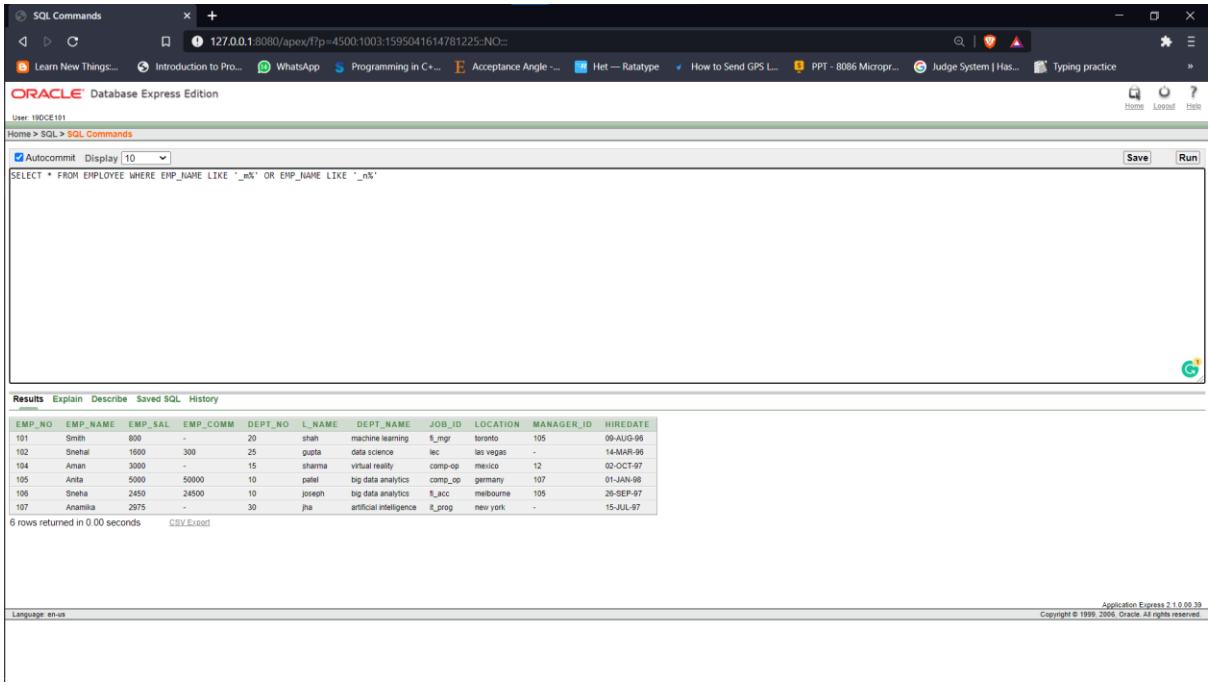
(2) Display name, number and salary of those employees whose name is 5 characters long and first three characters are ‘Ani’.

Snap-Shot:



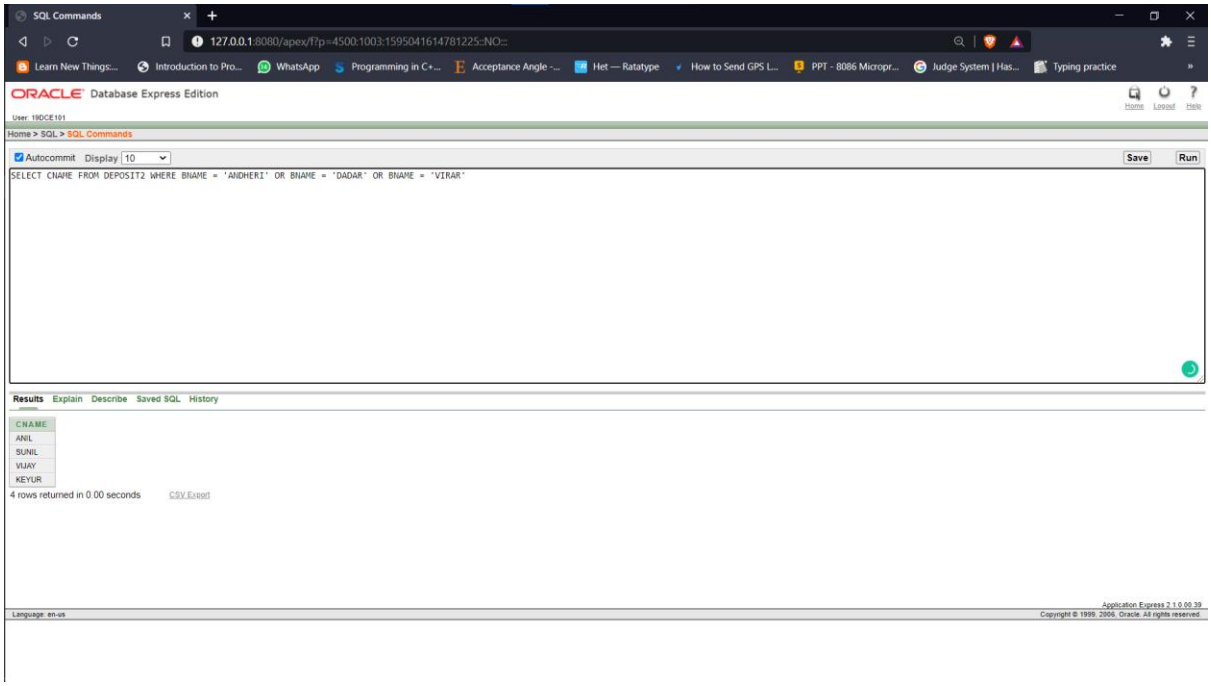
(3) Display all information of employee whose second character of name is either ‘M’ or ‘N’.

Snap-Shot:



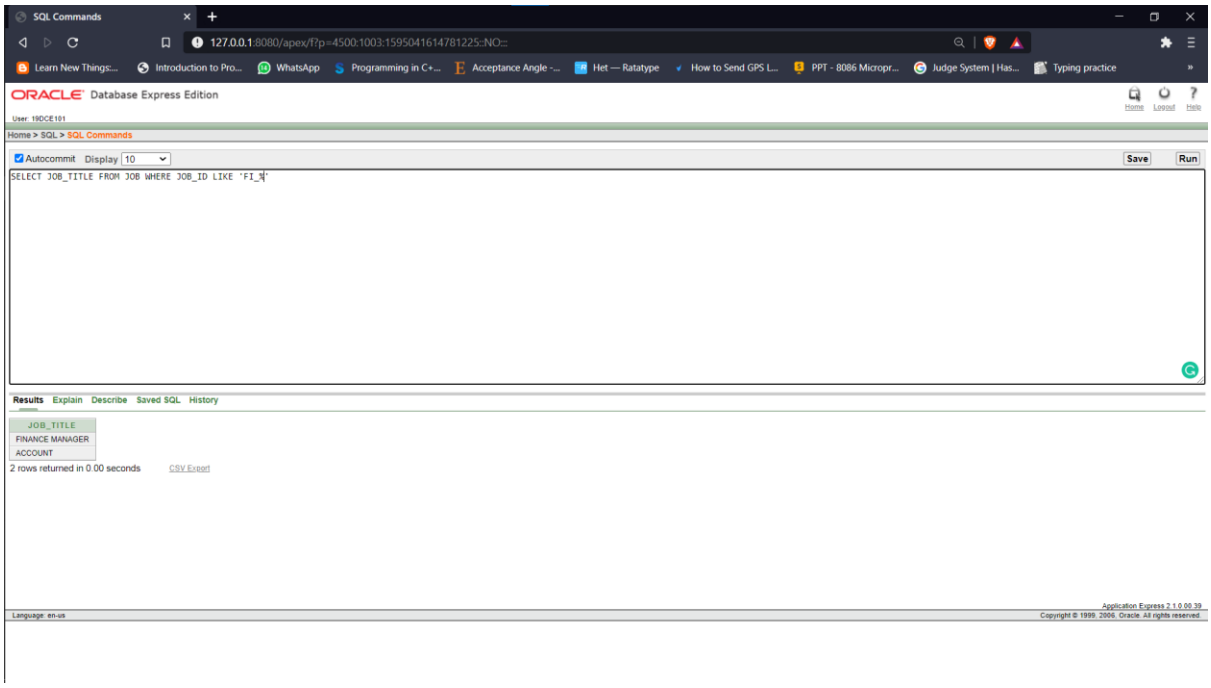
(4) Find the list of all customer name whose branch is in ‘andheri’ or ‘dadar’ or ‘virar’.

Snap-Shot:



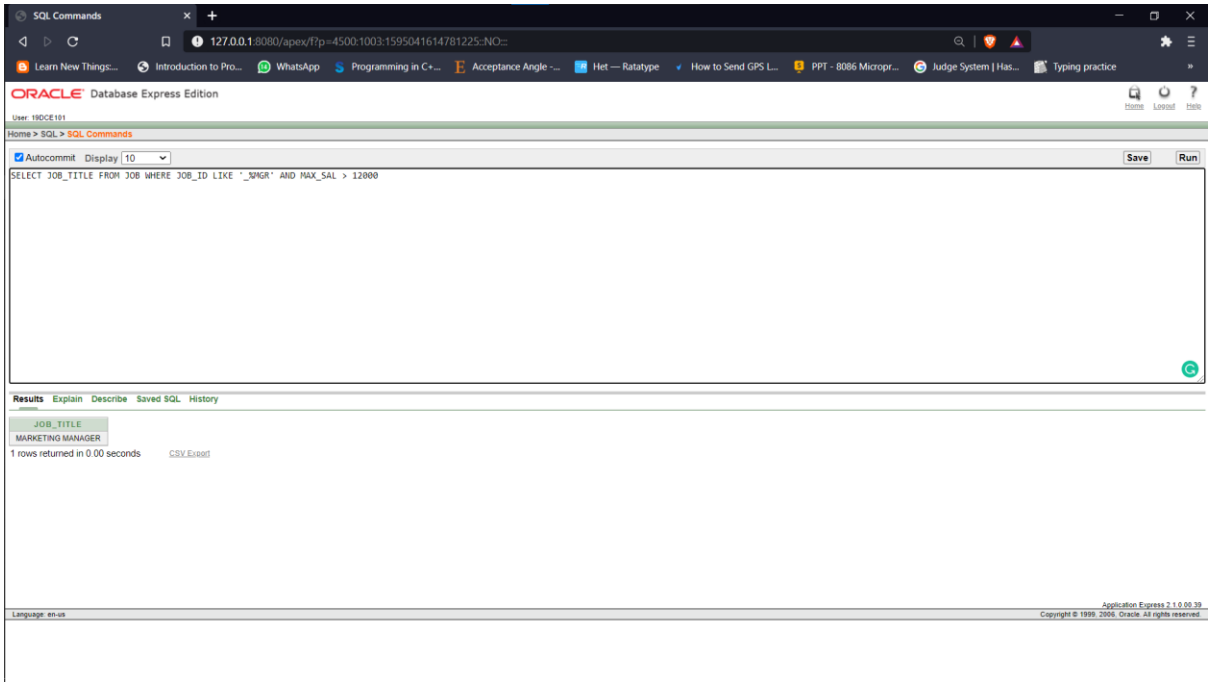
(5) Display the job name whose first three character in job id field is 'FI_'.

Snap-Shot:



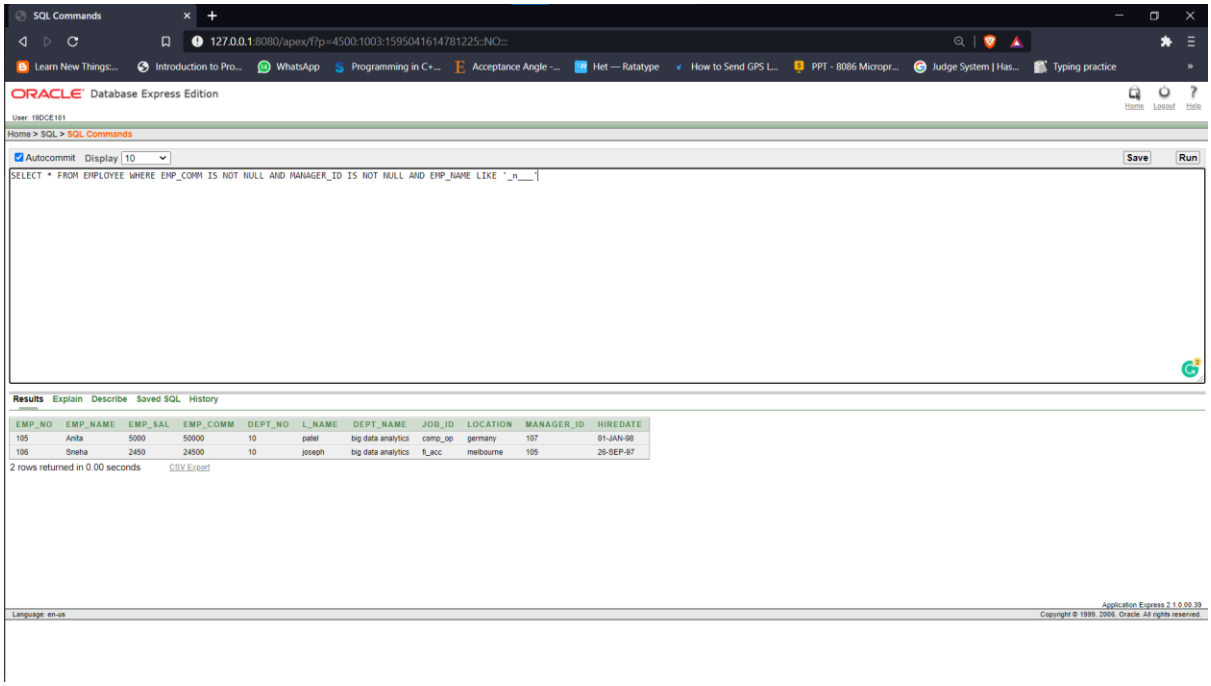
(6) Display the title/name of job who's last three character are '_MGR' and their maximum salary is greater than Rs 12000.

Snap-Shot:



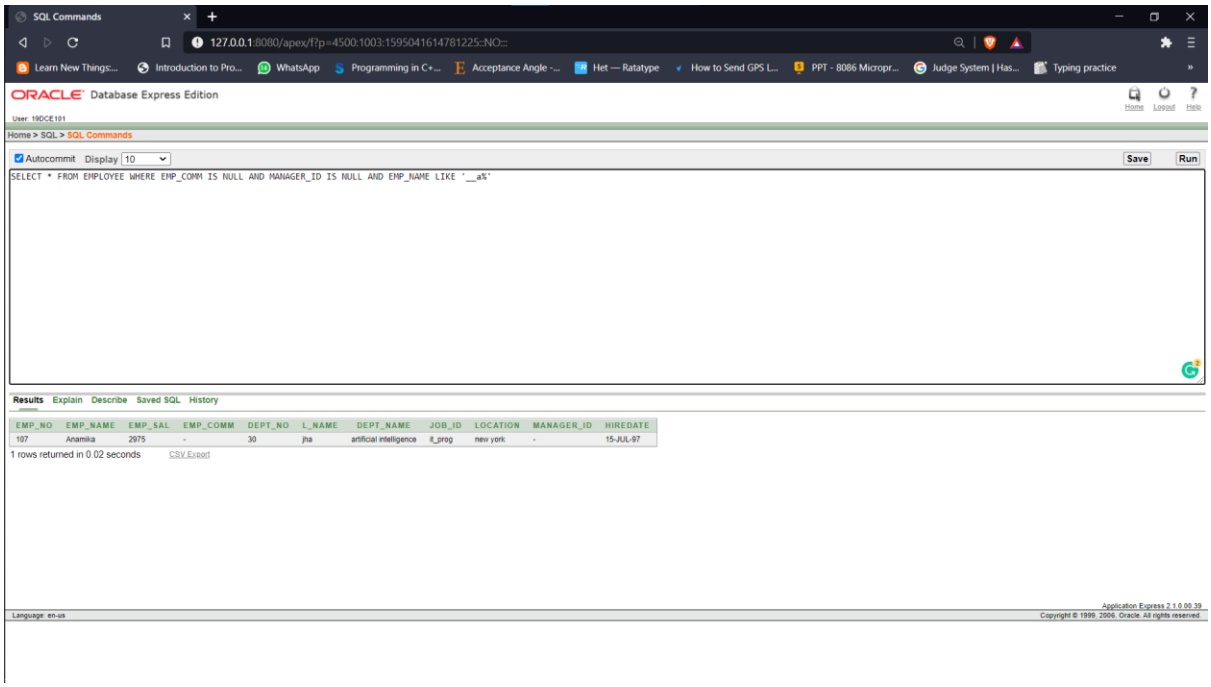
(7) Display the non-null values of employees and also employee name second character should be ‘n’ and string should be 5-character long.

Snap-Shot:



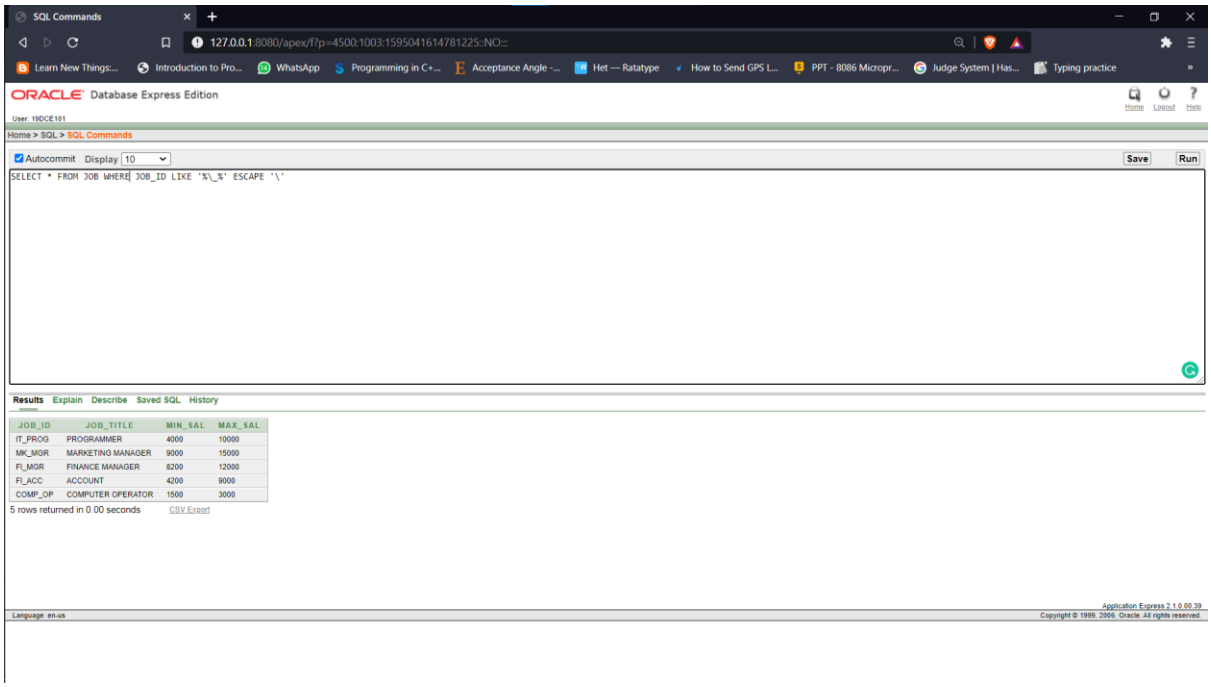
(8) Display the null values of employee and also employee name’s third character should be ‘a’.

Snap-Shot:



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

Snap-Shot:

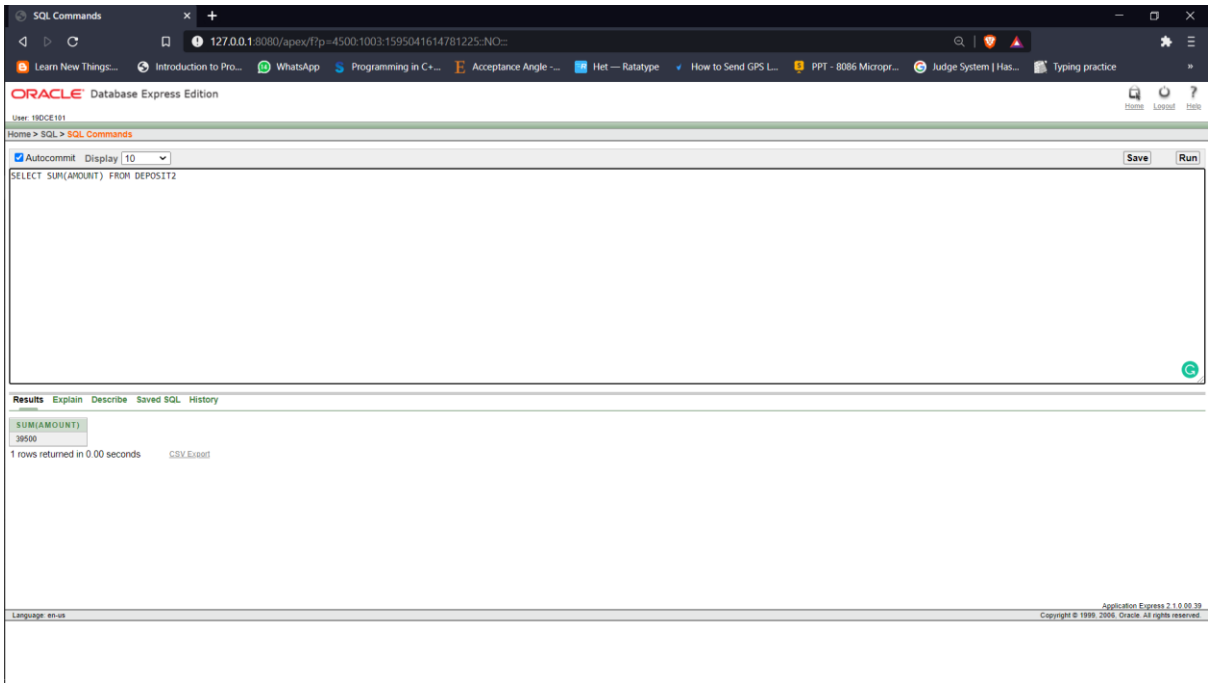


Conclusion: We Learned how to use Like Predicate & perform specific queries.

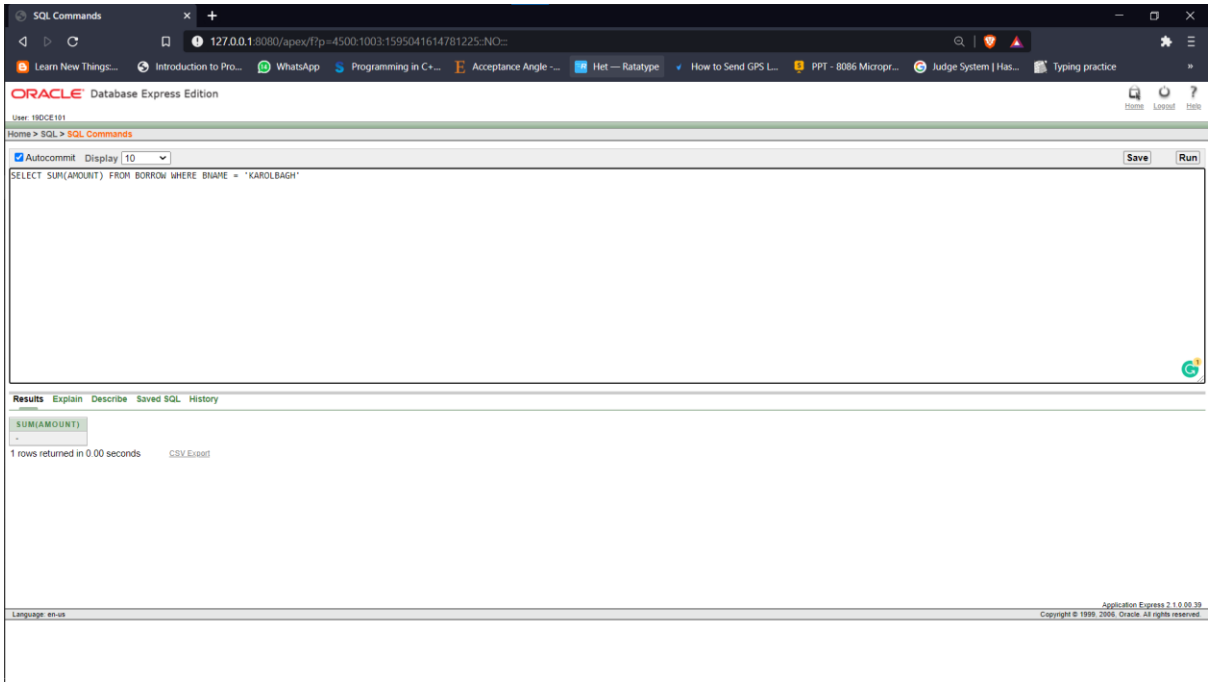
Practical -5

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

(1) List total deposit from deposit.
Snap-Shot:

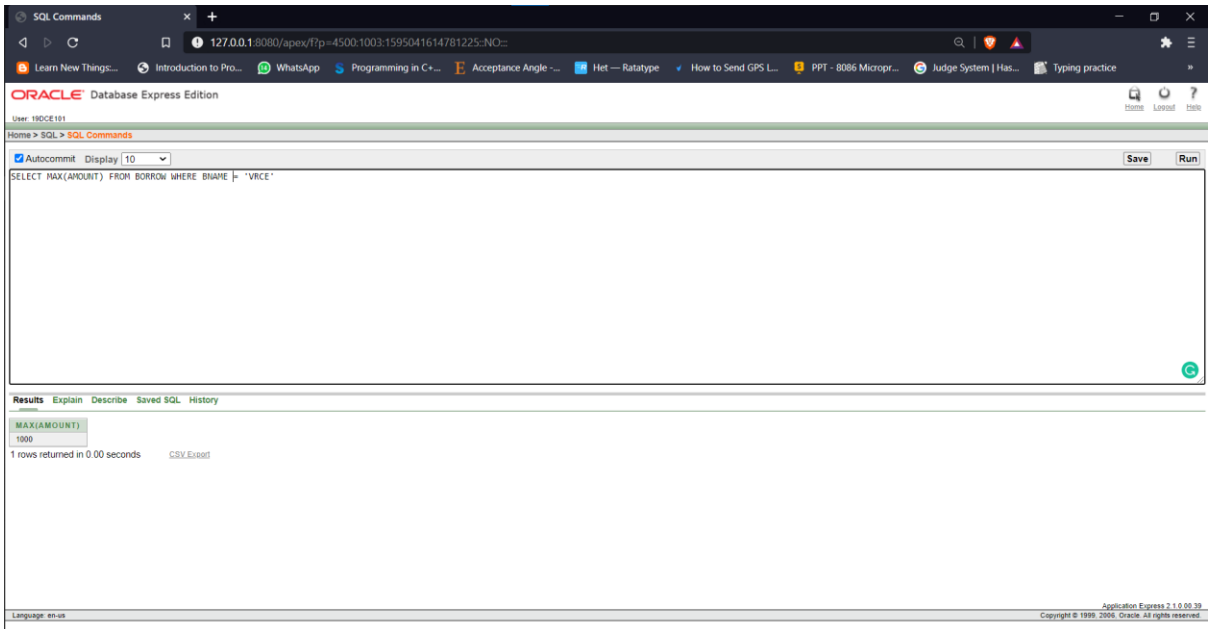


(2) List total loan from karolbagh branch
Snap-Shot:



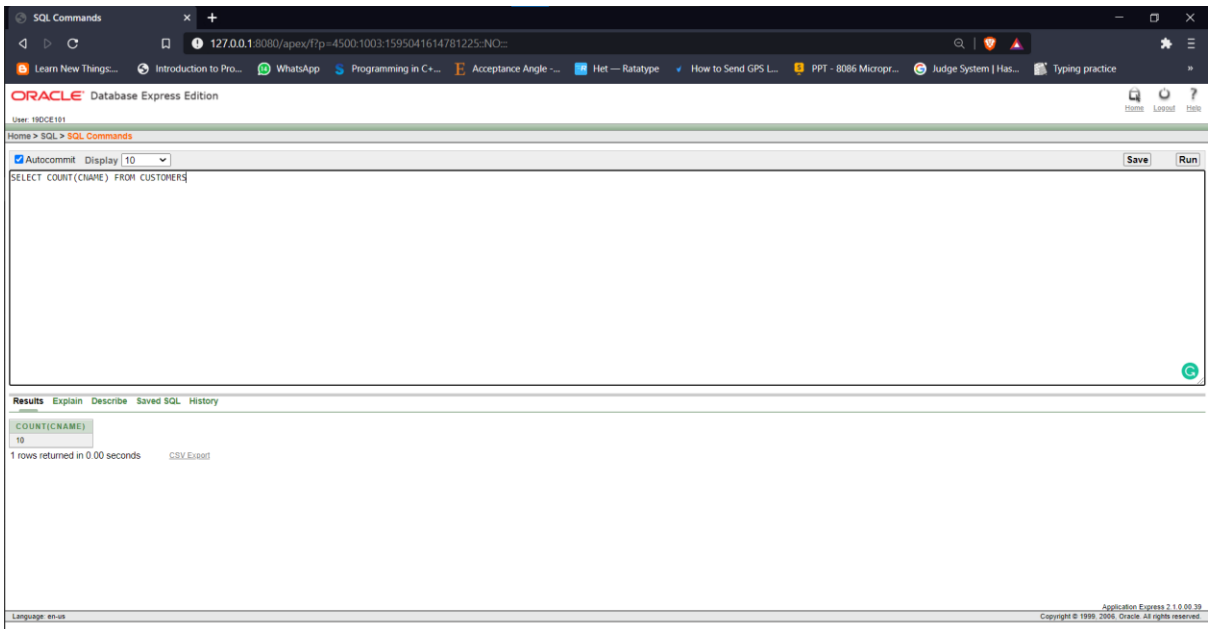
(3) Give maximum loan from branch vrce.

Snap-Shot:



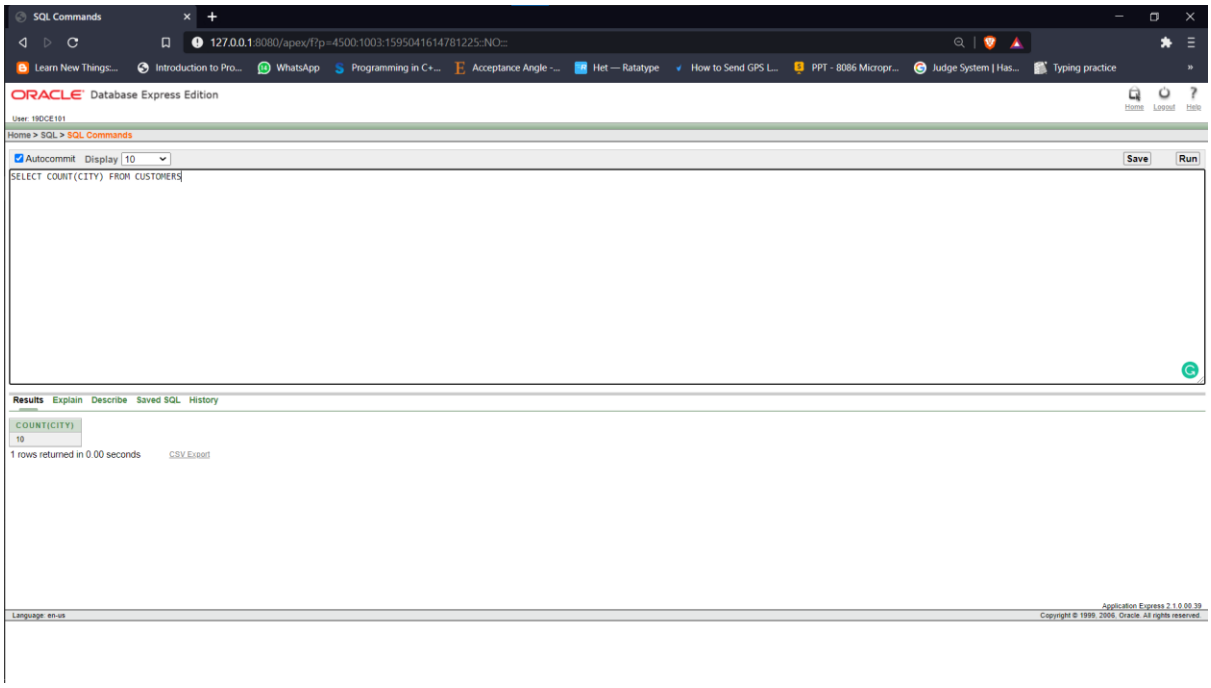
(4) Count total number of customers

Snap-Shot:



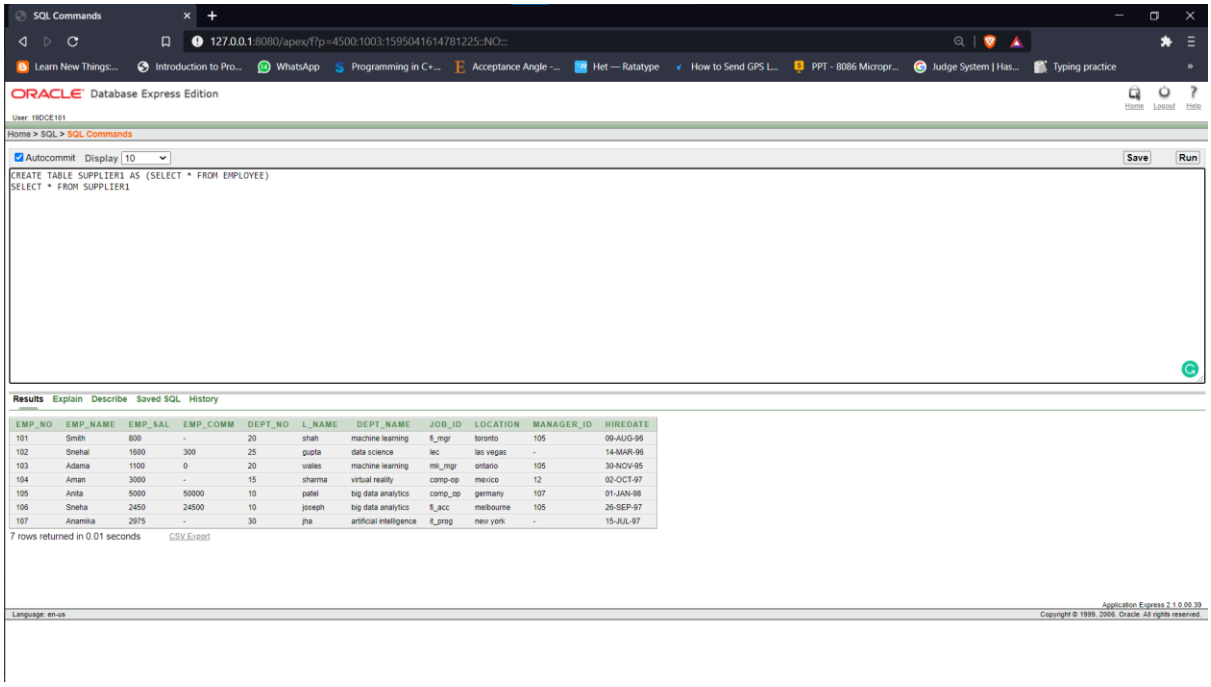
(5) Count total number of customer's cities.

Snap-Shot:



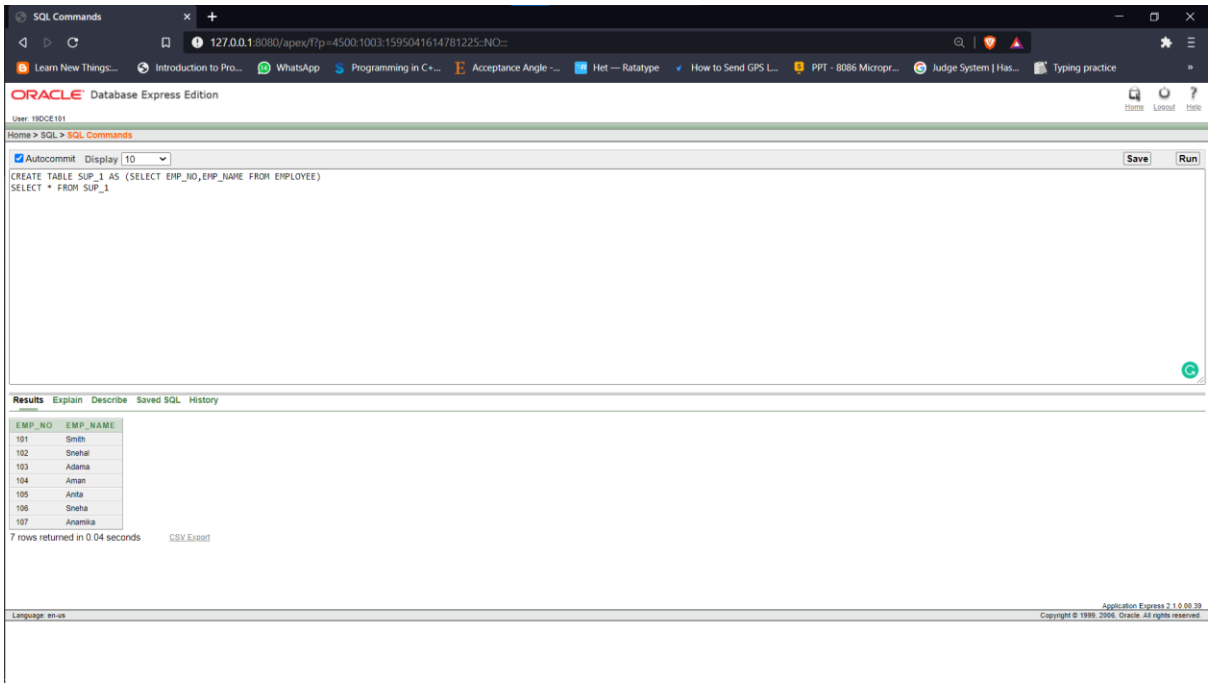
(6) Create table supplier from employee with all the columns.

Snap-Shot:



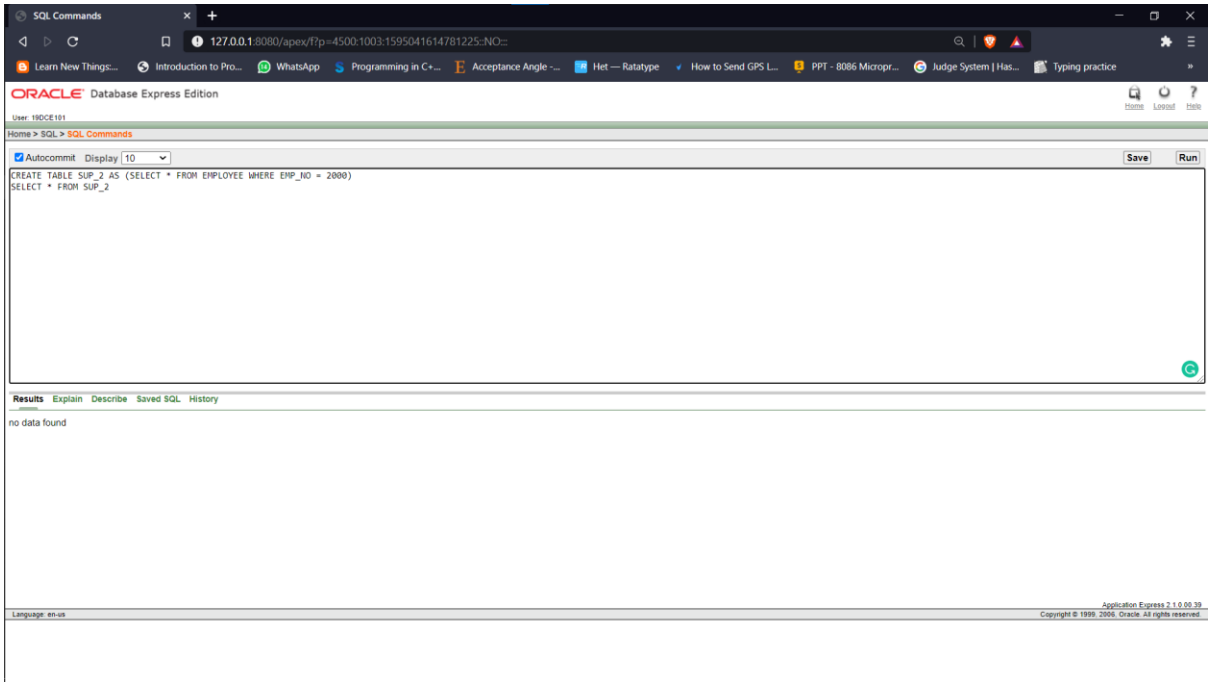
(7) Create table sup1 from employee with first two columns.

Snap-Shot:



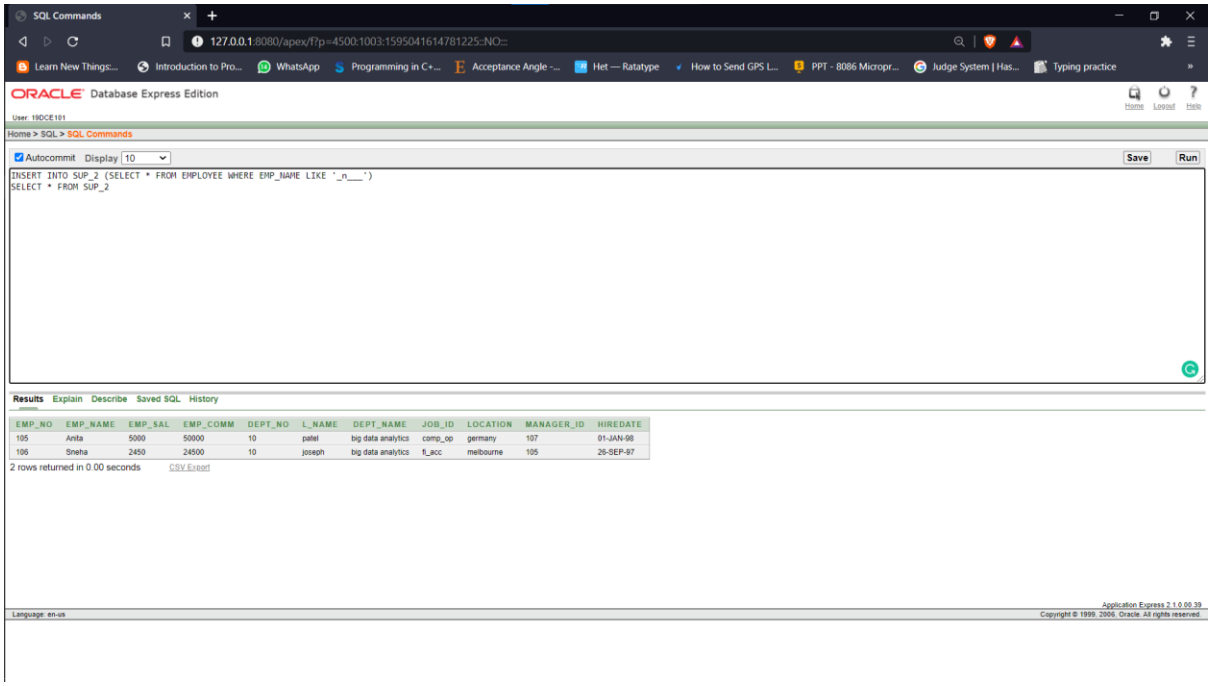
(8) Create table sup2 from employee with no data

Snap-Shot:



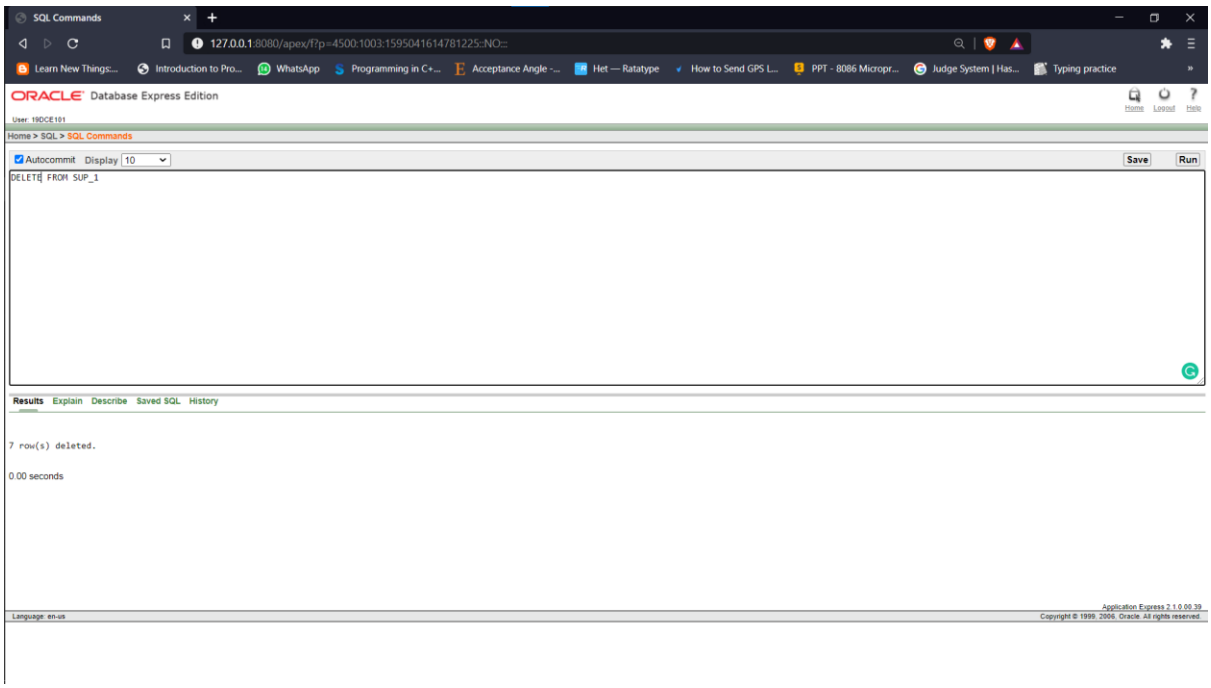
(9) Insert the data into sup2 from employee whose second character should be ‘n’ and string should be 5 characters long in employee name field.

Snap-Shot:



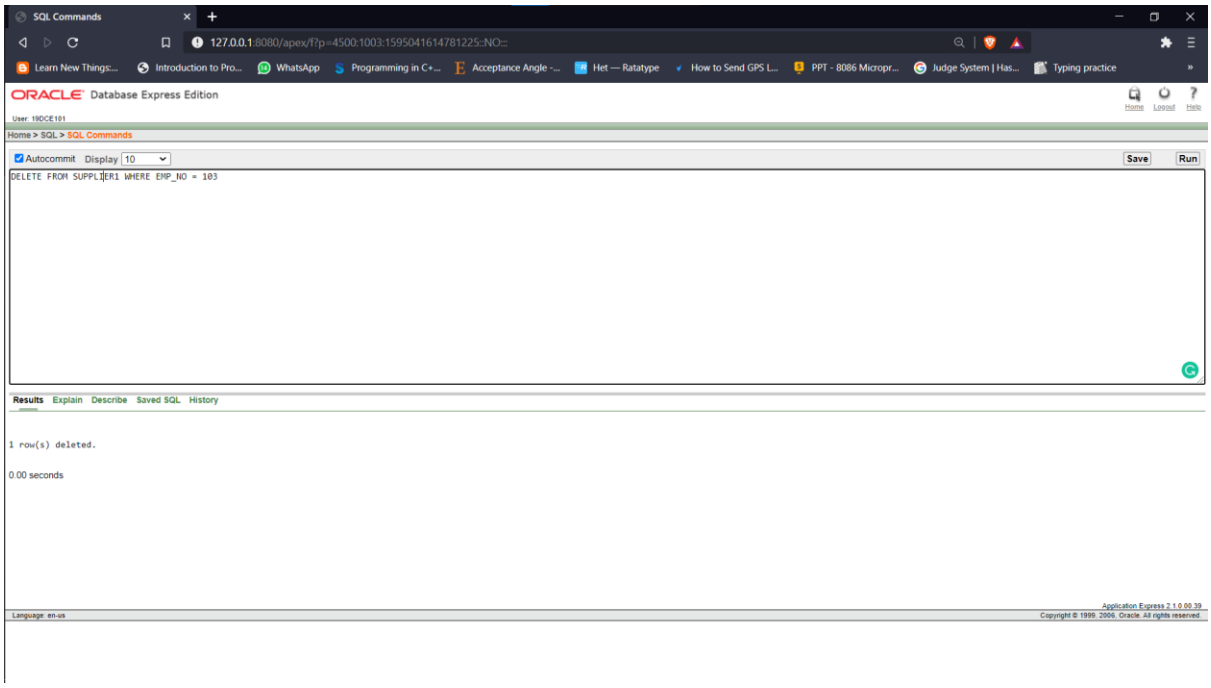
(10) Delete all the rows from sup1.

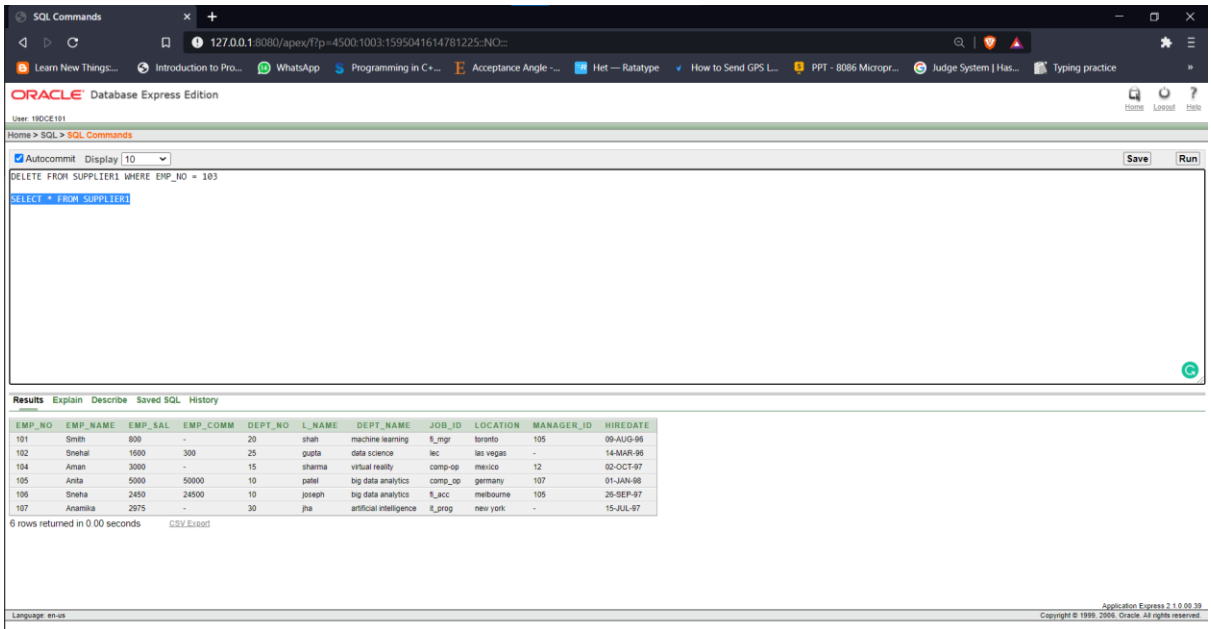
Snap-Shot:



(11) Delete the detail of supplier whose sup_no is 103.

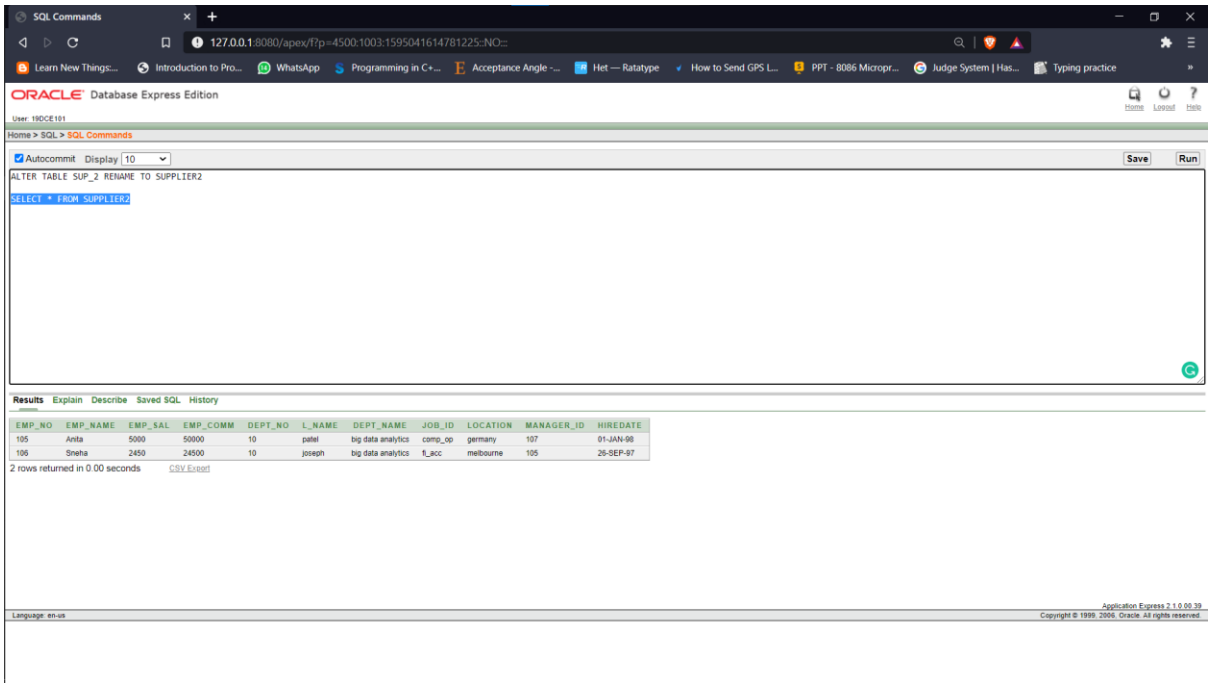
Snap-Shot:





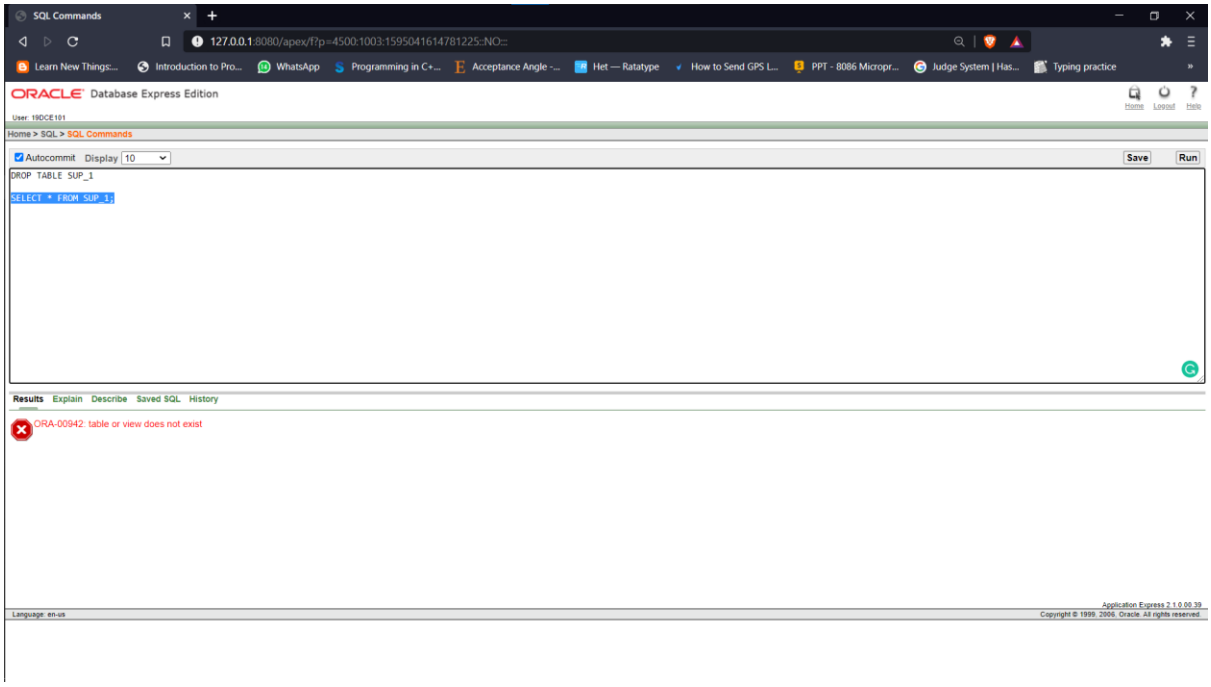
(12) Rename the table sup2.

Snap-Shot:



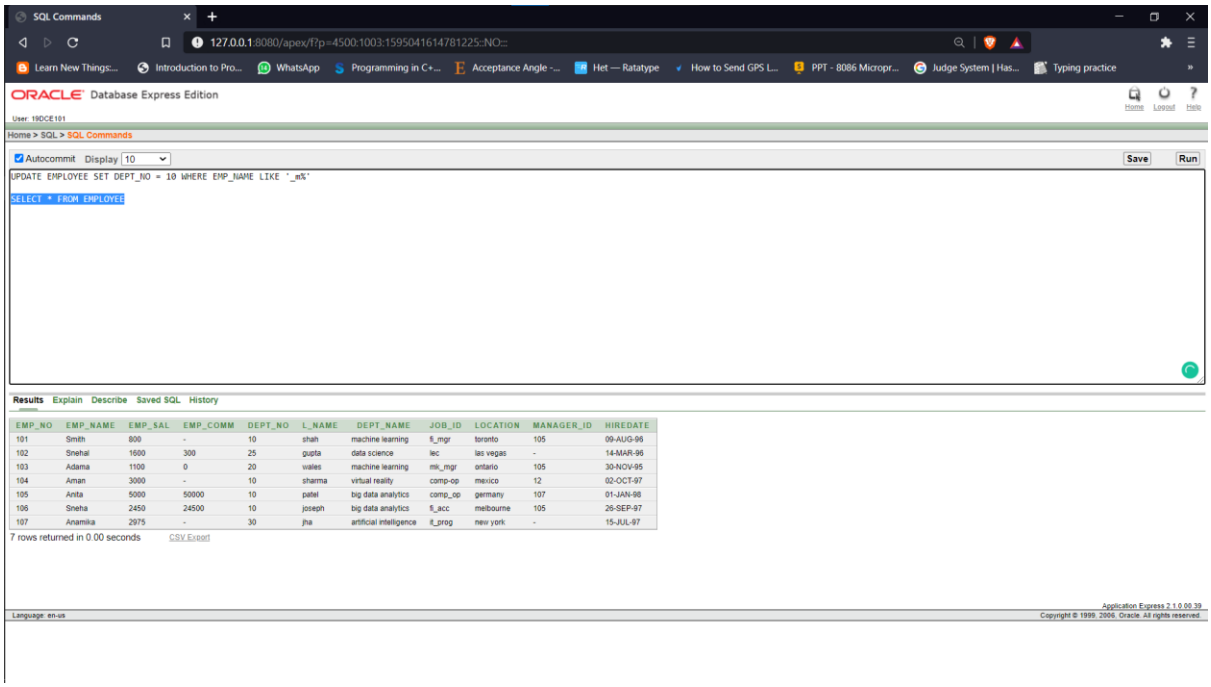
(13) Destroy table sup1 with all the data.

Snap-Shot:



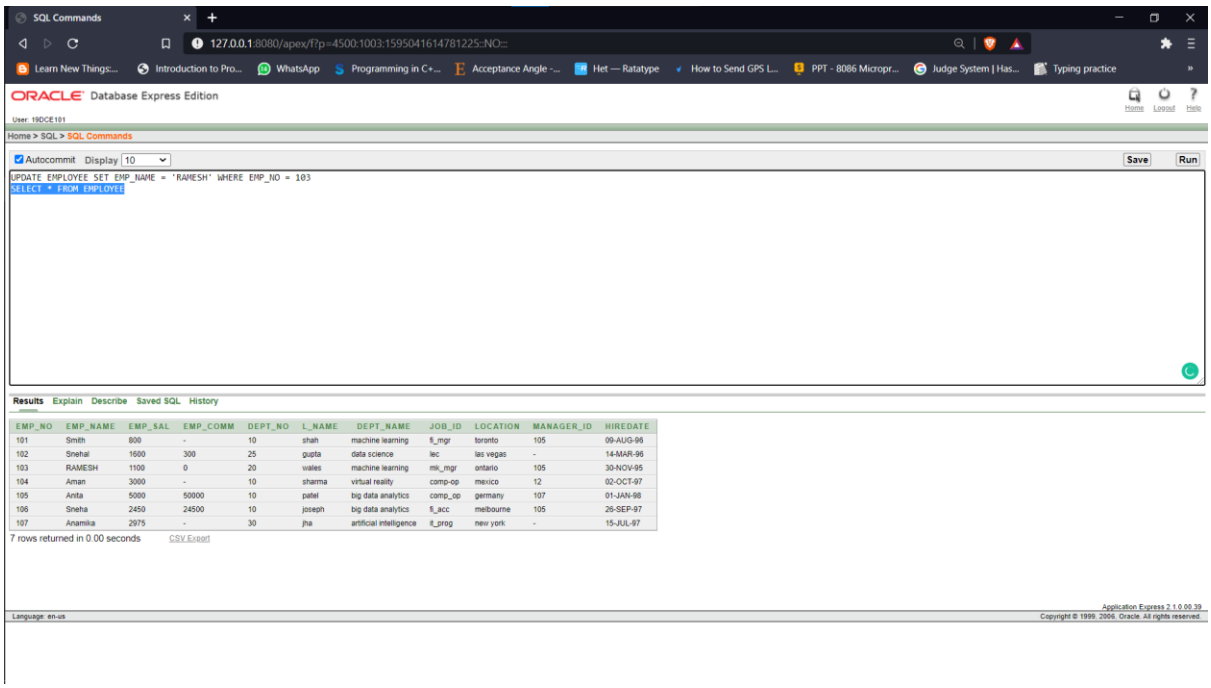
(14) Update the value dept_no to 10 where second character of emp. name is ‘m’.

Snap-Shot:



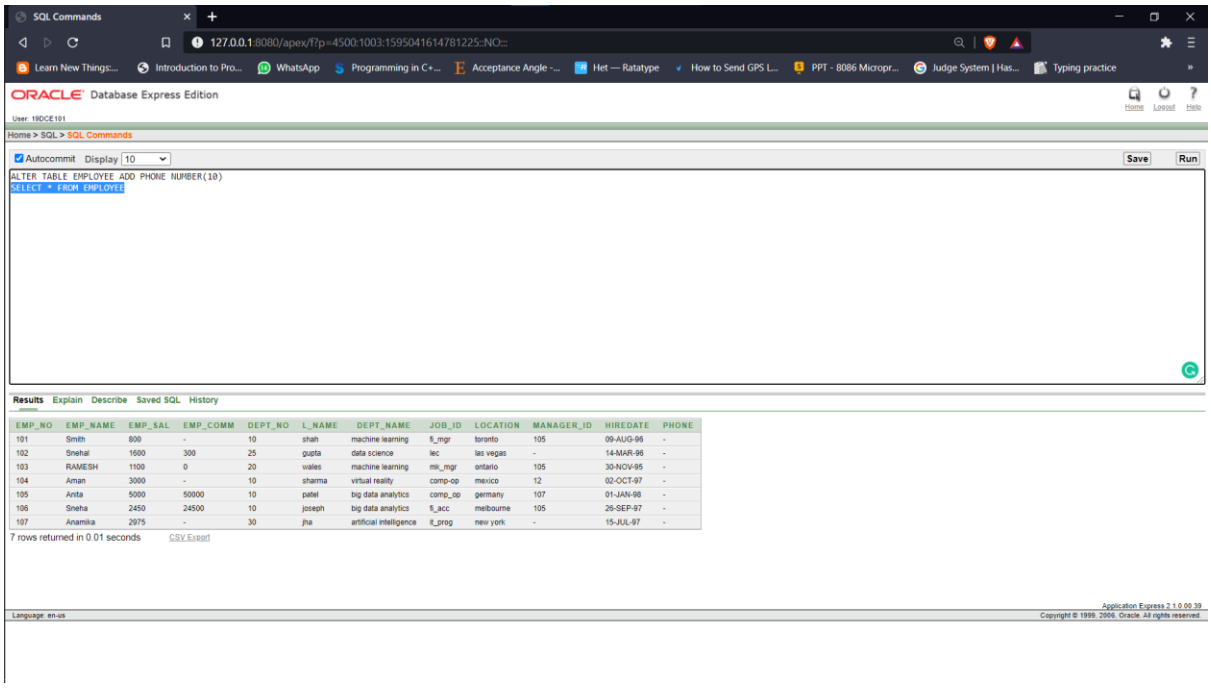
(15) Update the value of employee name whose employee number is 103.

Snap-Shot:



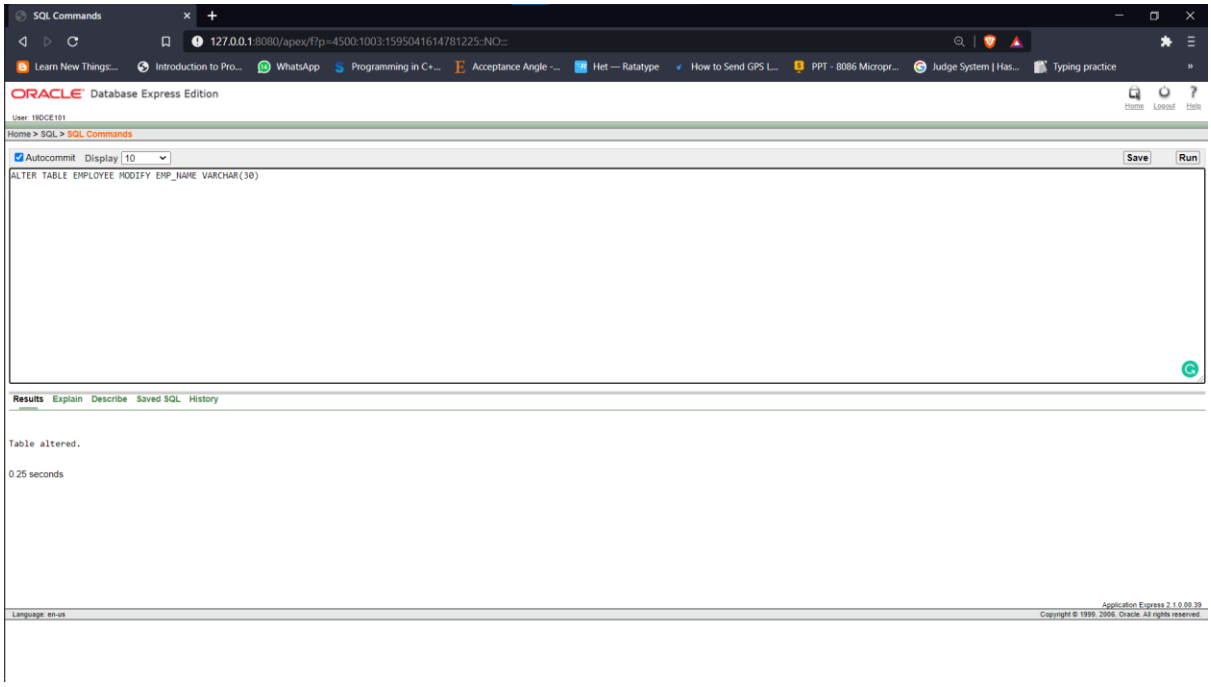
(16) Add one column phone to employee with size of column is 10.

Snap-Shot:



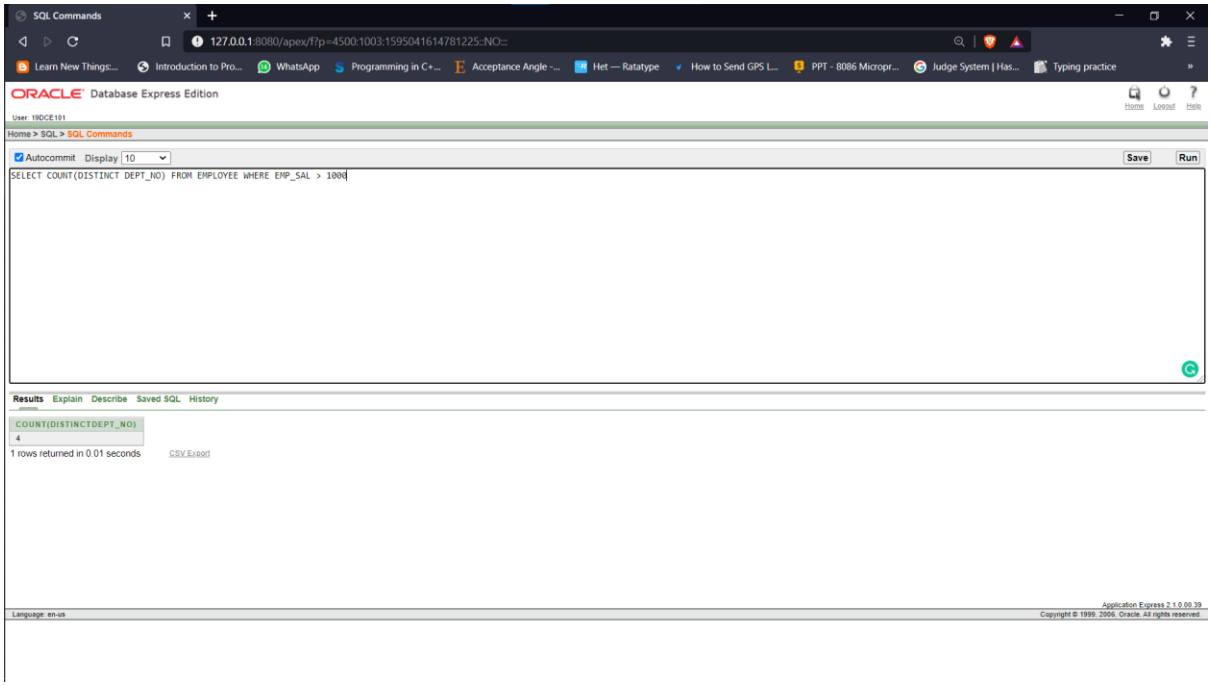
(17) Modify the column emp_name to hold maximum of 30 characters.

Snap-Shot:



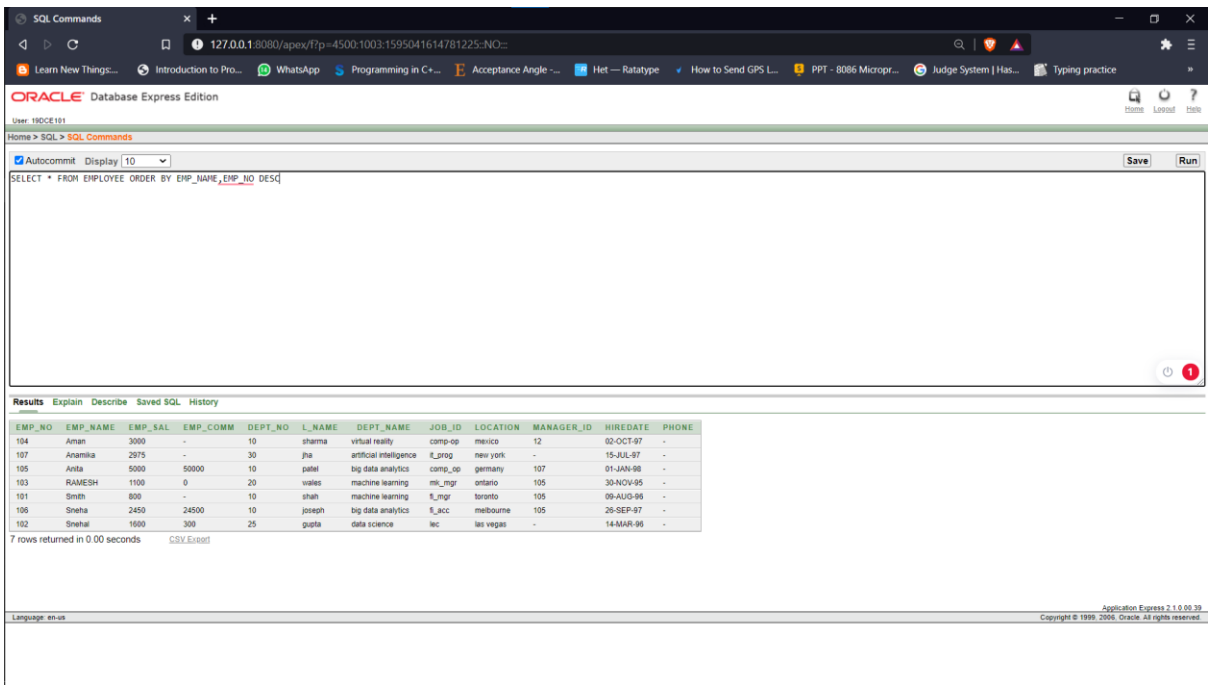
(18) Count the total no as well as distinct rows in dept_no column with a condition of salary greater than 1000 of employee

Snap-Shot:



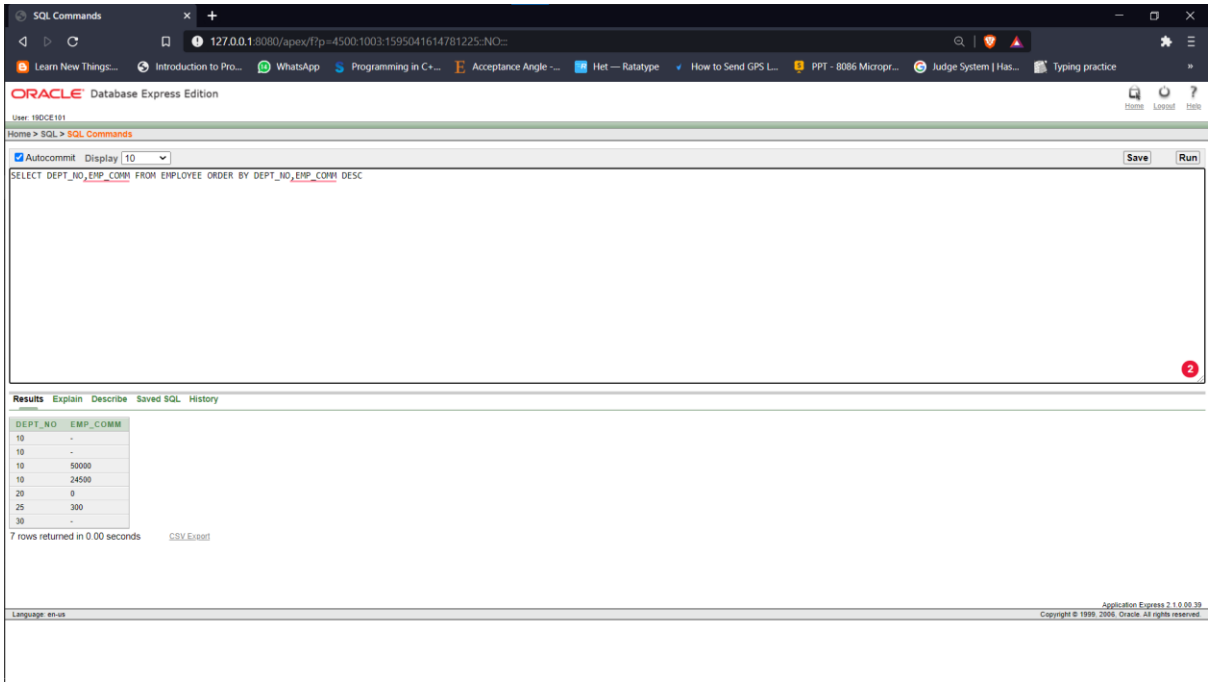
(19) Display the detail of all employees in ascending order, descending order of their name and no.

Snap-Shot:



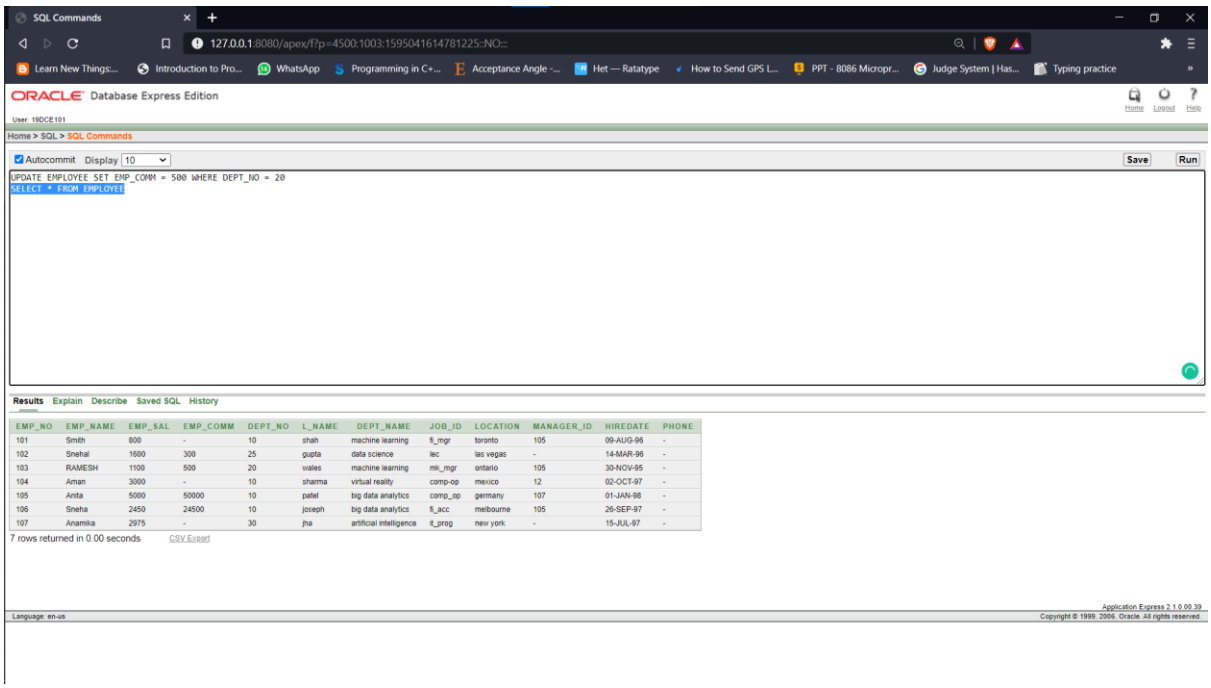
(20) Display the dept_no in ascending order and accordingly display emp_comm in descending order.

Snap-Shot:



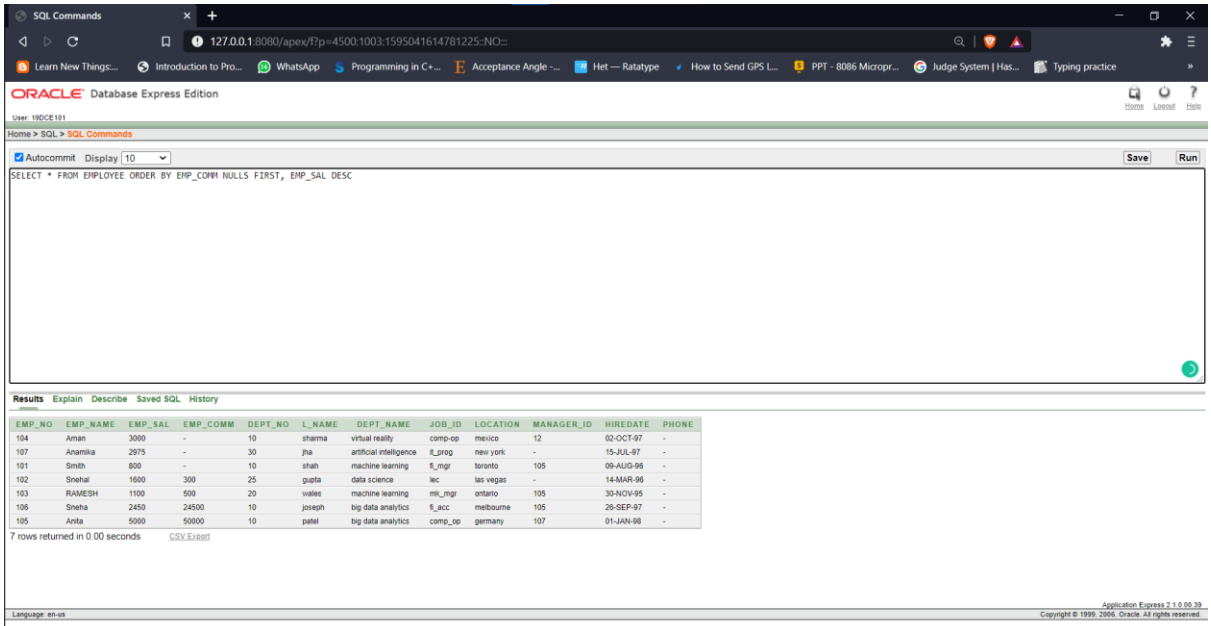
(21) Update the value of emp_comm to 500 where dept_no is 20.

Snap-Shot:



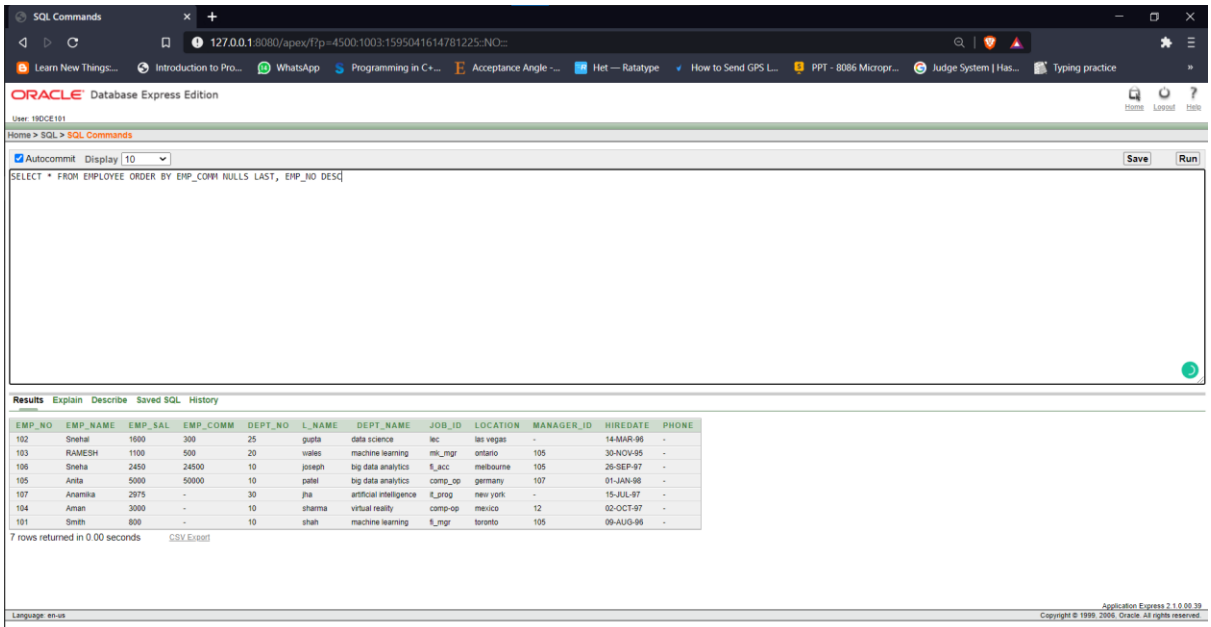
(22) Display the emp_comm in ascending order with null value first and accordingly sort employee salary in descending order.

Snap-Shot:



(23) Display the emp_comm in ascending order with null value last and accordingly sort emp_no in descending order.

Snap-Shot:



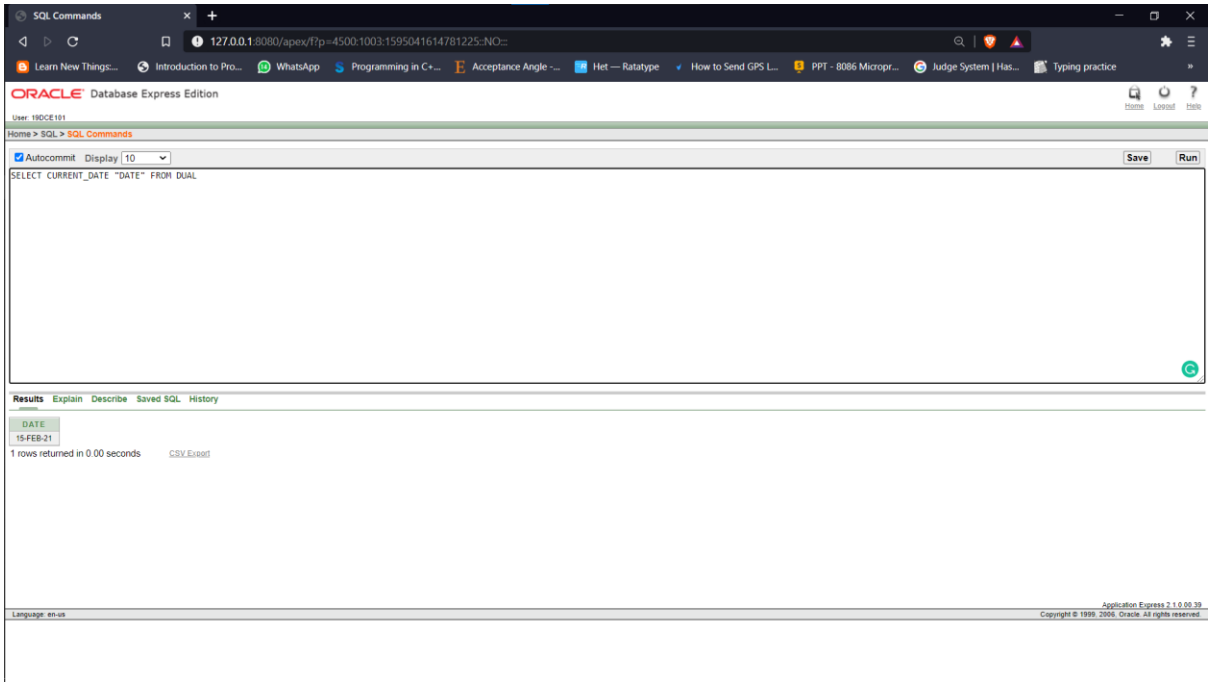
Conclusion: We Learned how to use aggregate functions, DML queries & Sorting.

Practical -6

To study Single-row functions.

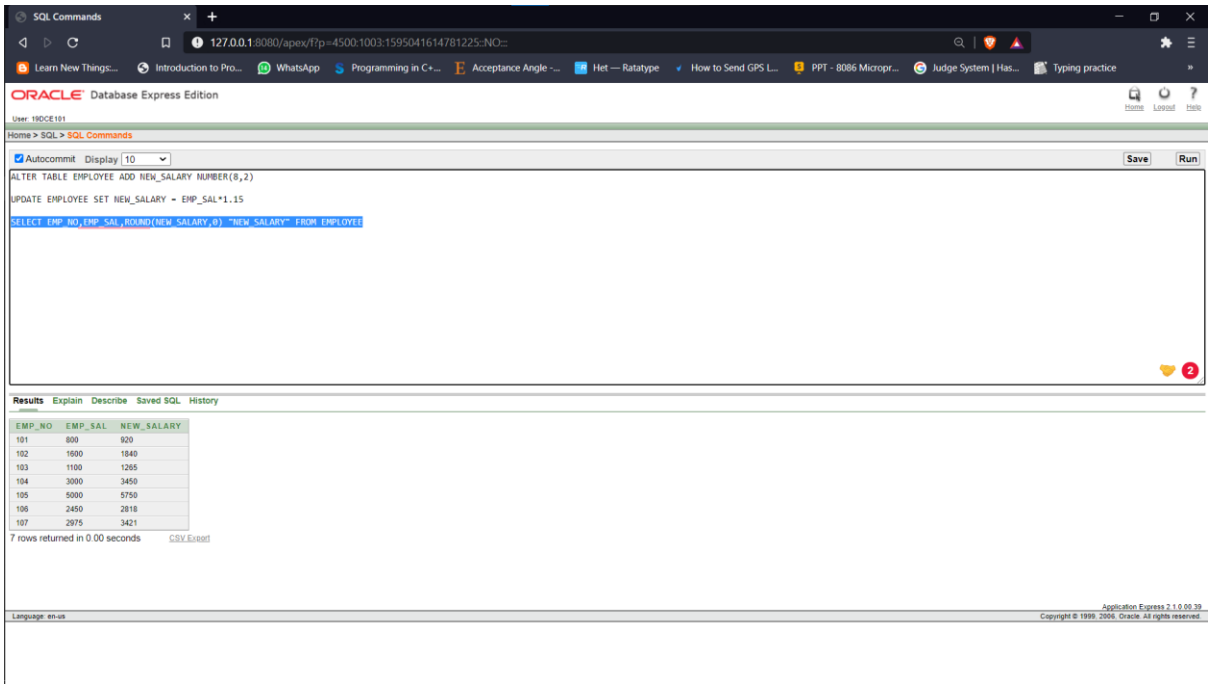
(1) Write a query to display the current date. Label the column Date.

Snap-Shot:



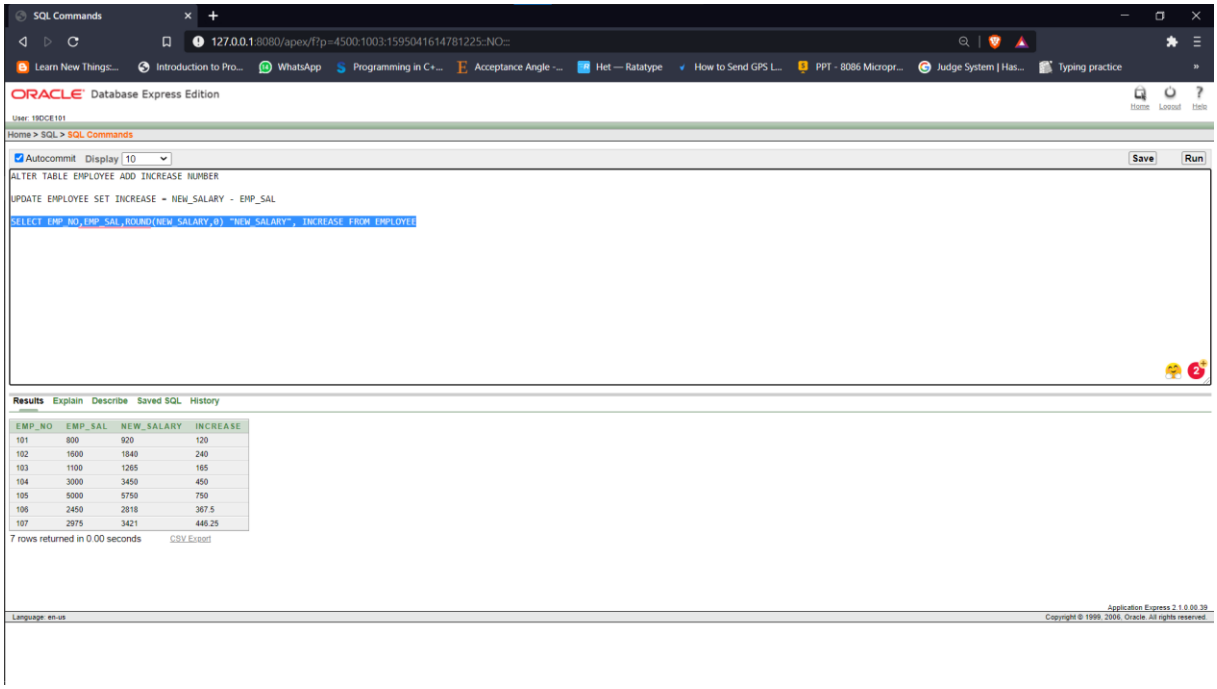
(2) For each employee, display the employee number, salary, and salary increased by 15% and expressed as a whole number. Label the column New Salary.

Snap-Shot:



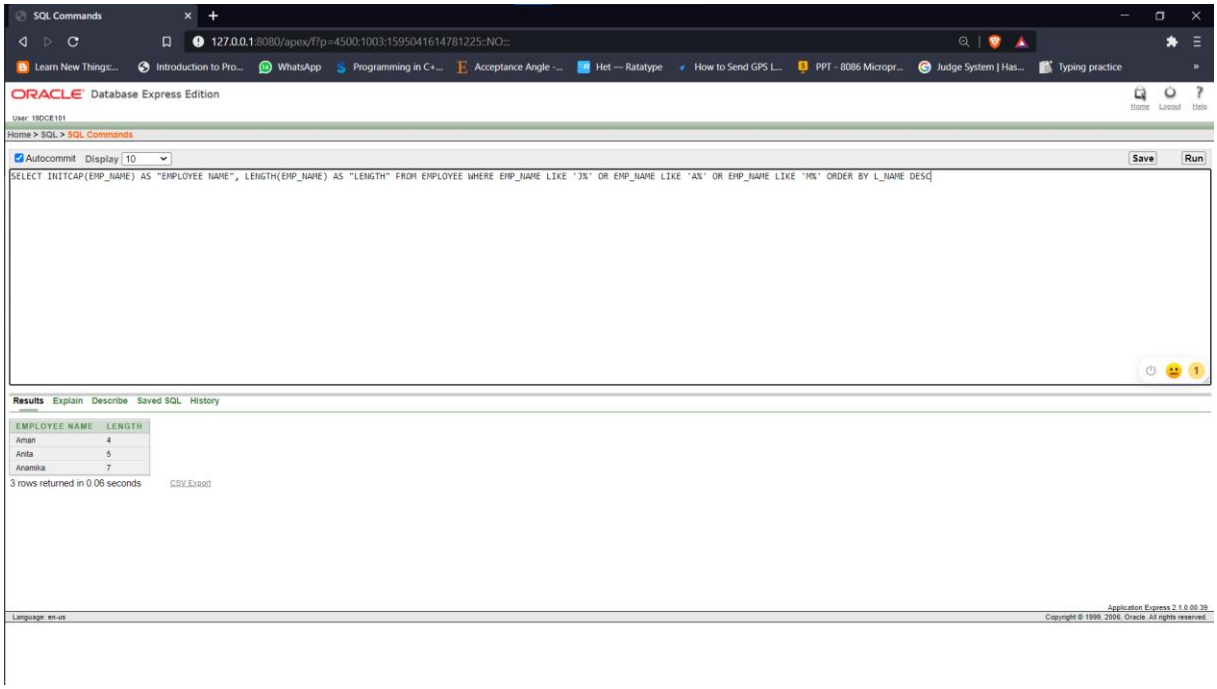
(3) Modify your query no (2) to add a column that subtracts the old salary from the new salary. Label the column Increase.

Snap-Shot:



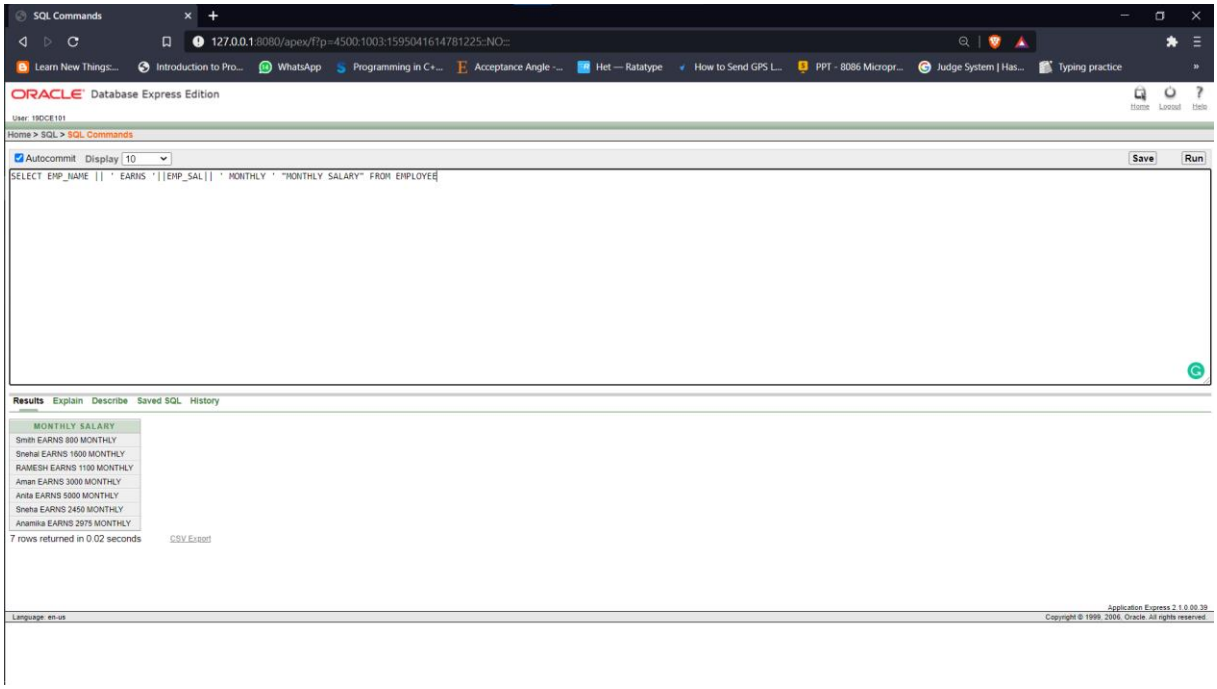
(4) Write a query that displays the employee's names with the first letter capitalized and all other letters lowercase, and the length of the names, for all employees whose name starts with J, A, or M. Give each column an appropriate label. Sort the results by the employees' last names.

Snap-Shot:



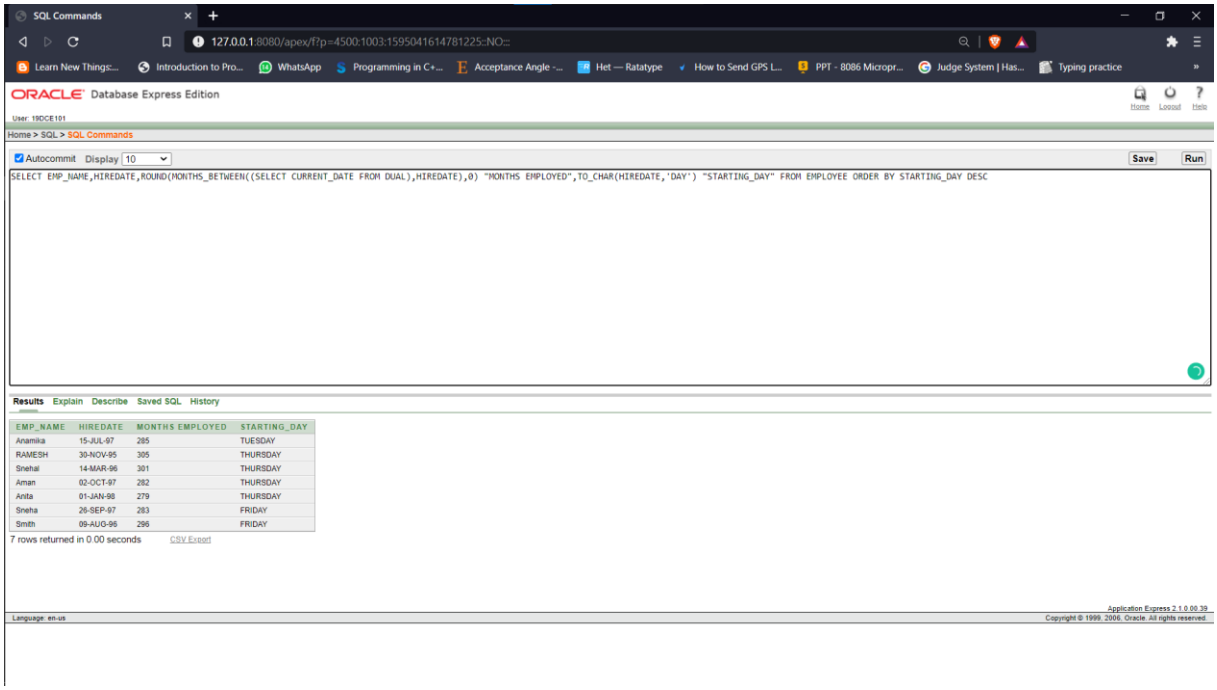
(5) Write a query that produces the following for each employee:
<employee last name> earns <salary> monthly

Snap-Shot:



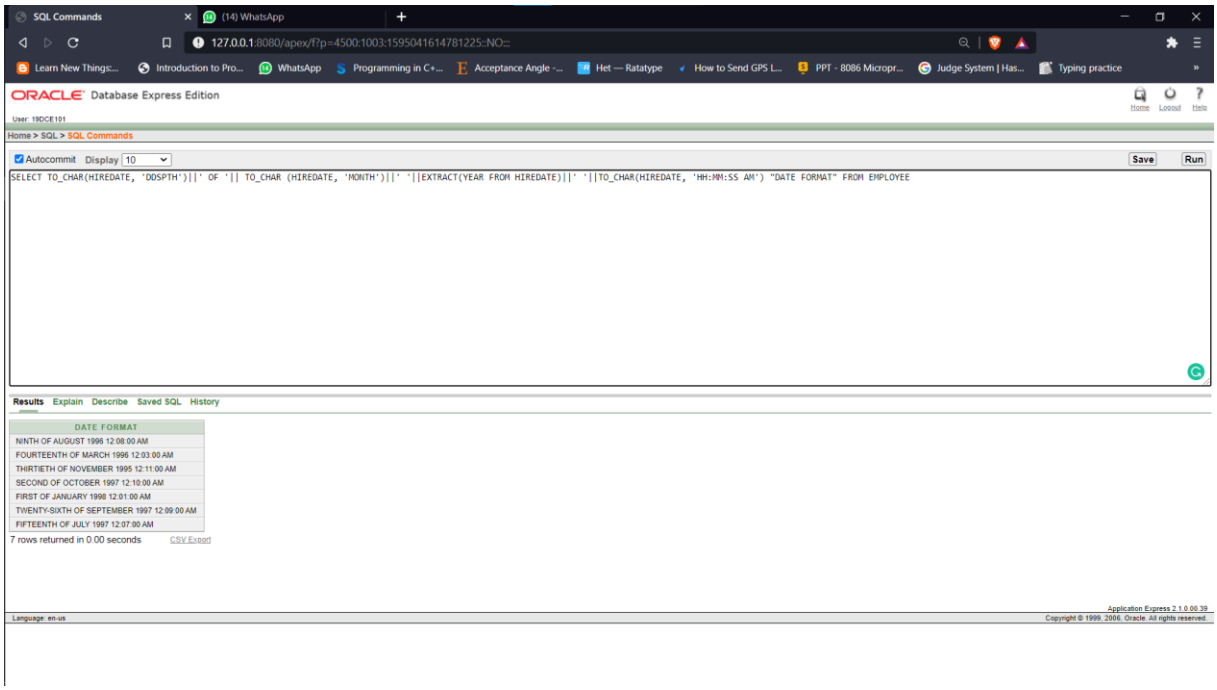
(6) Display the name, date, number of months employed and day of the week on which the employee has started. Order the results by the day of the week starting with Monday.

Snap-Shot:



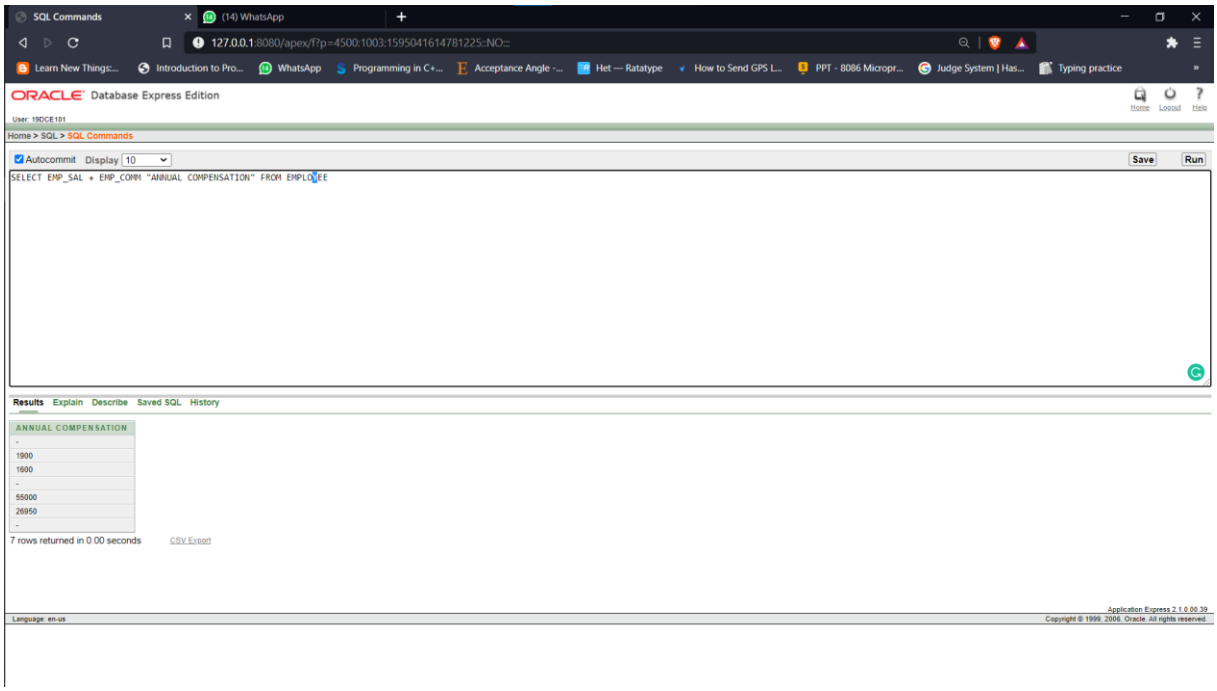
(7) Display the date of emp in a format that appears as Seventh of June 1994 12:00:00 AM.

Snap-Shot:



(8) Write a query to calculate the annual compensation of all employees (sal +comm.).

Snap-Shot:



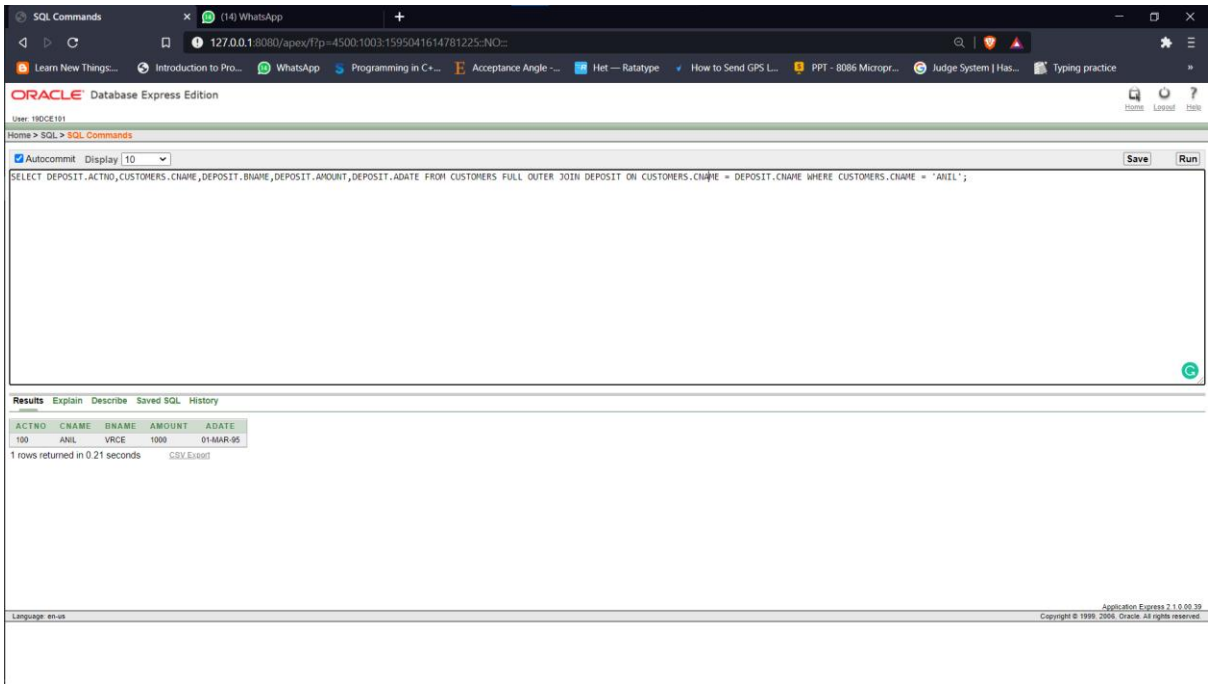
Conclusion: We Learned how to use Single-row functions**.**

Practical -7

Displaying data from Multiple Tables (join)

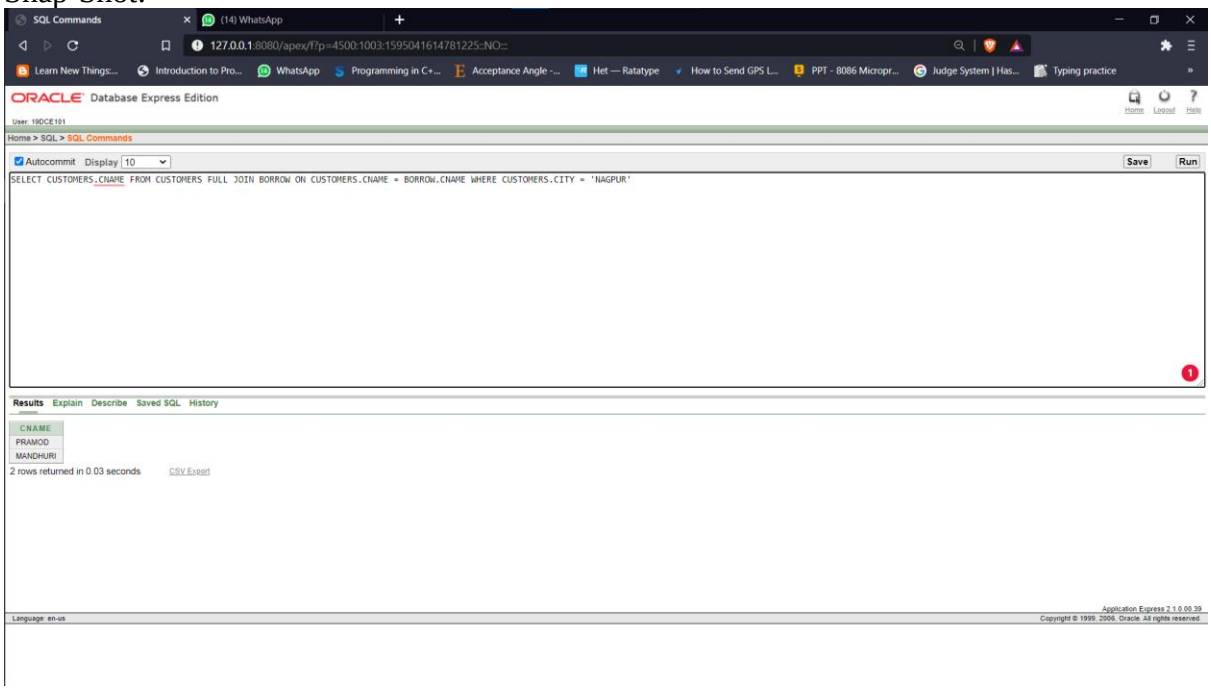
(1) Give details of customers ANIL.

Snap-Shot:



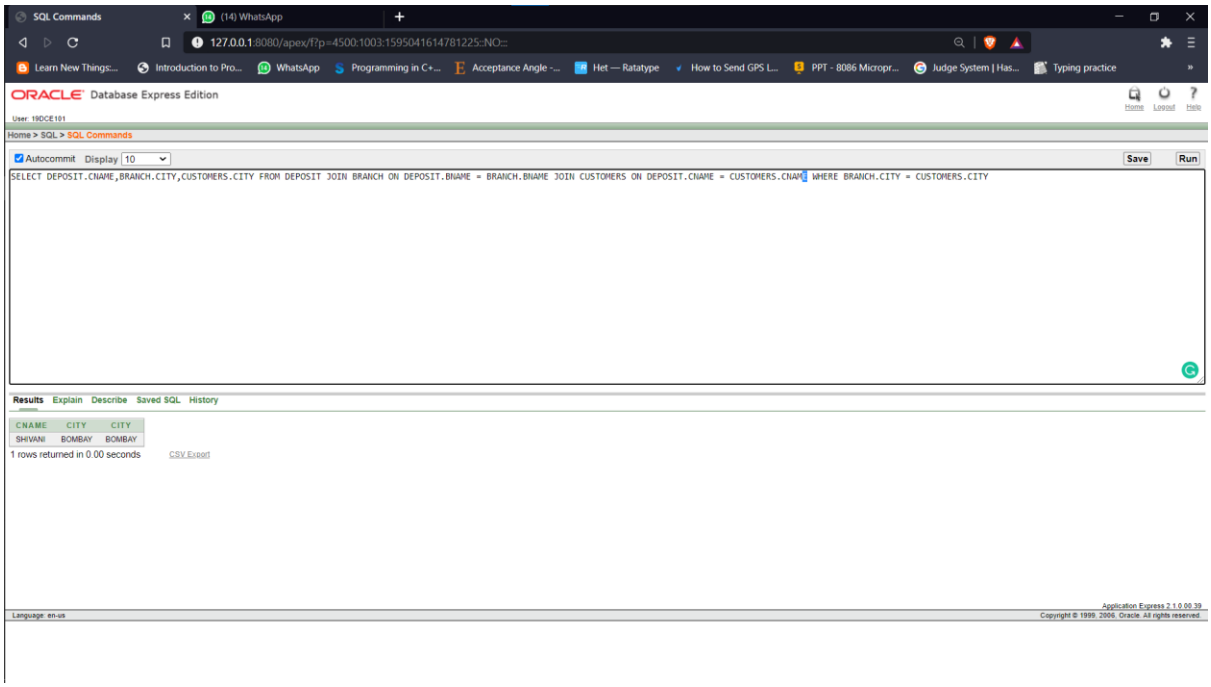
(2) Give name of customer who are borrowers and depositors and having living city nagpur

Snap-Shot:



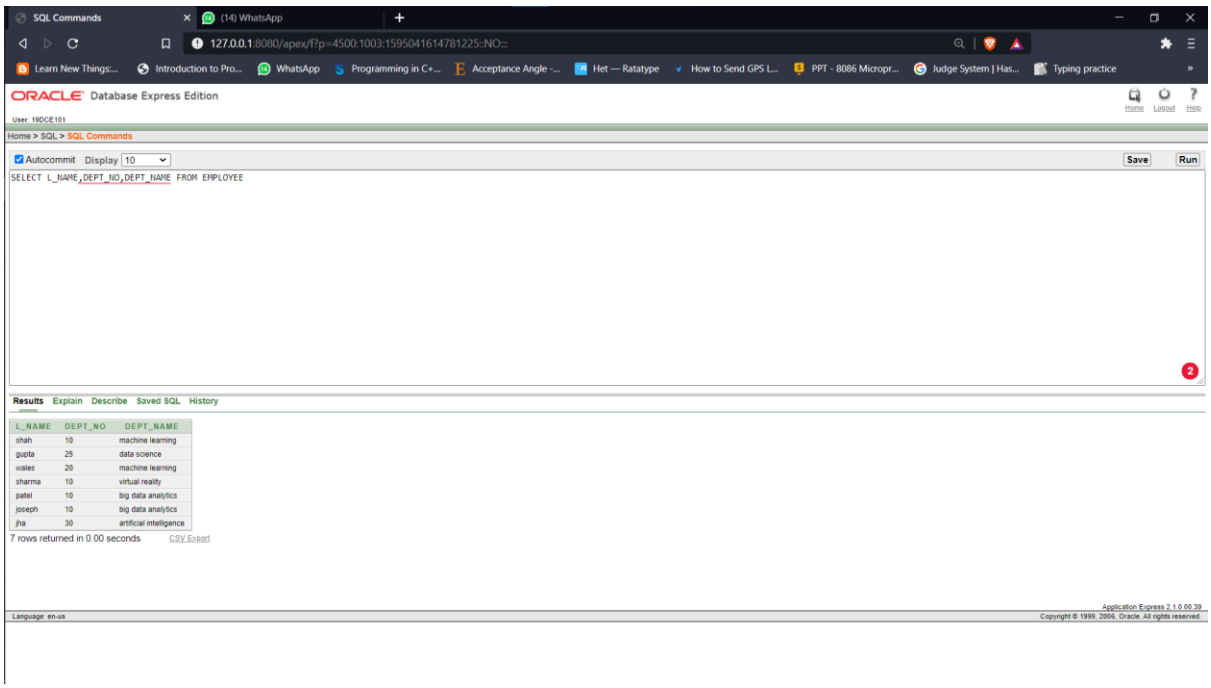
(3) Give city as their city name of customers having same living branch.

Snap-Shot:



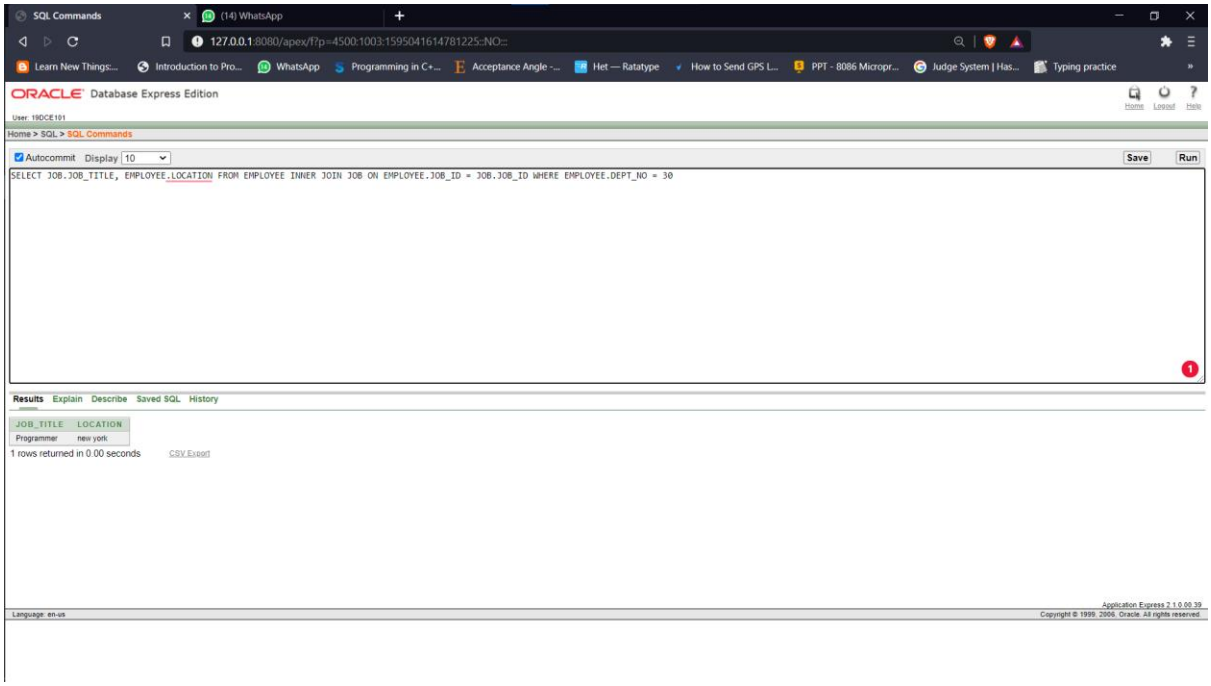
(4) Write a query to display the last name, department number, and department name for all employees.

Snap-Shot:



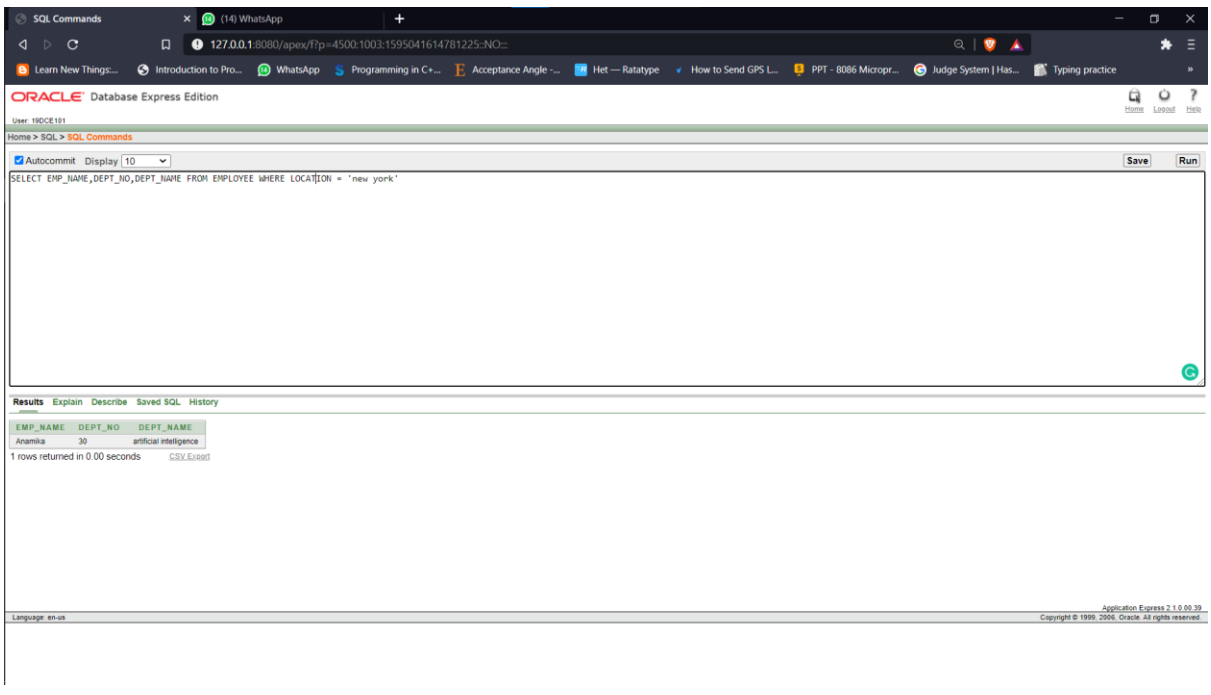
(5) Create a unique listing of all jobs that are in department 30. Include the location of the department in the output

Snap-Shot:



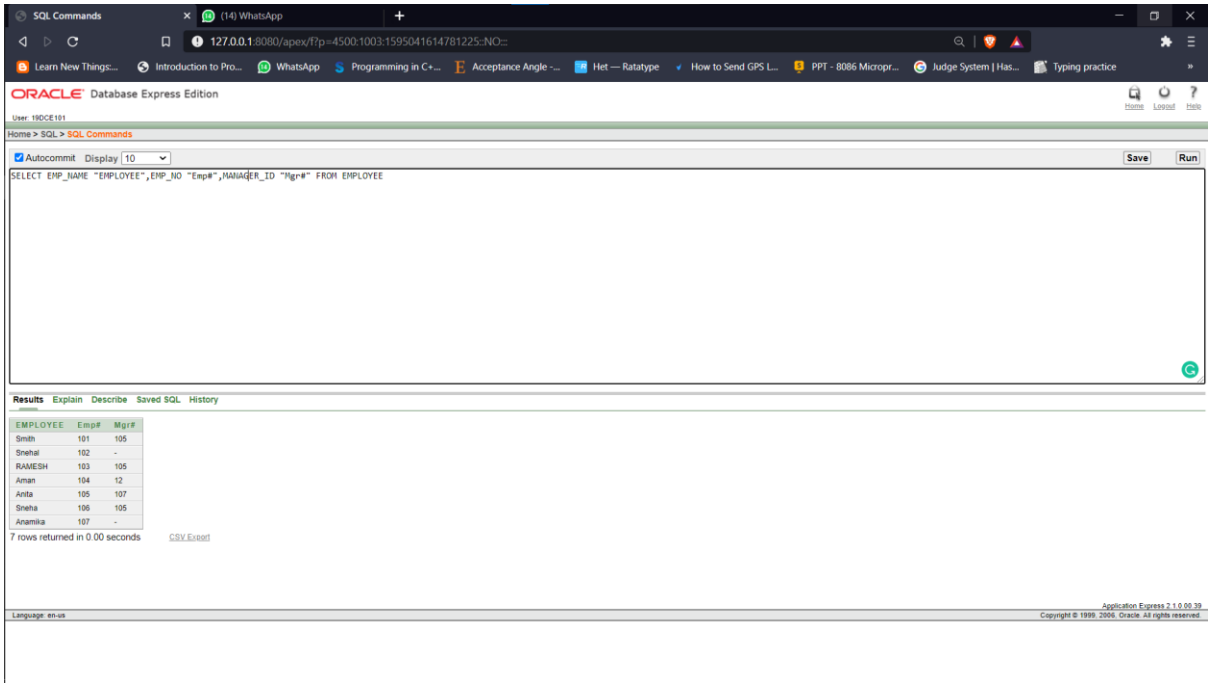
(6) Write a query to display the employee name, department number, and department name for all employees who work in NEW YORK.

Snap-Shot:



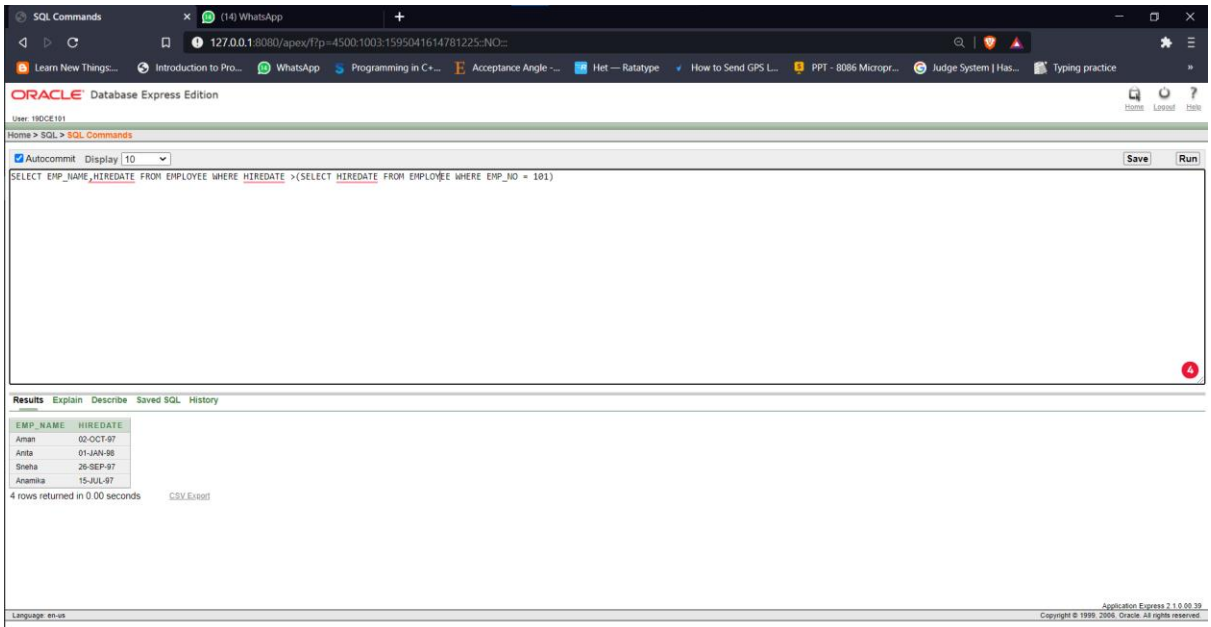
(7) Display the employee last name and employee number along with their manager's last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively.

Snap-Shot:



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

Snap-Shot:



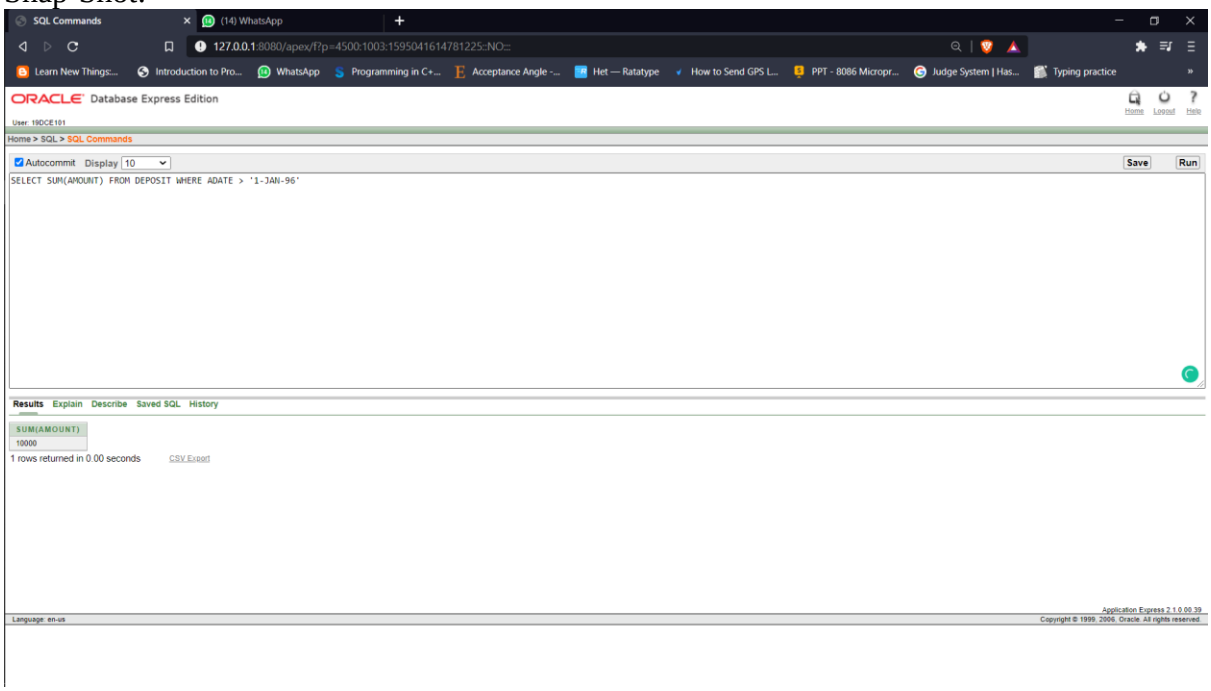
Conclusion: We Learned how to display data using join from multiple tables.

Practical -8

To apply the concept of Aggregating Data using Group functions.

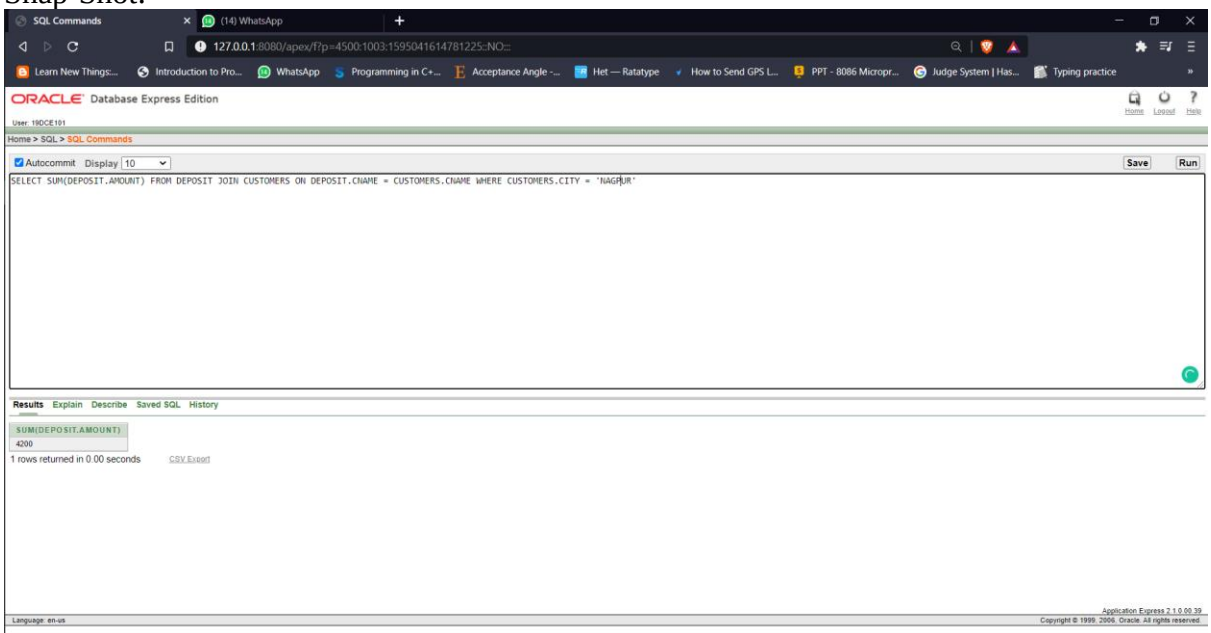
(1) List total deposit of customer having account date after 1-jan-96.

Snap-Shot:



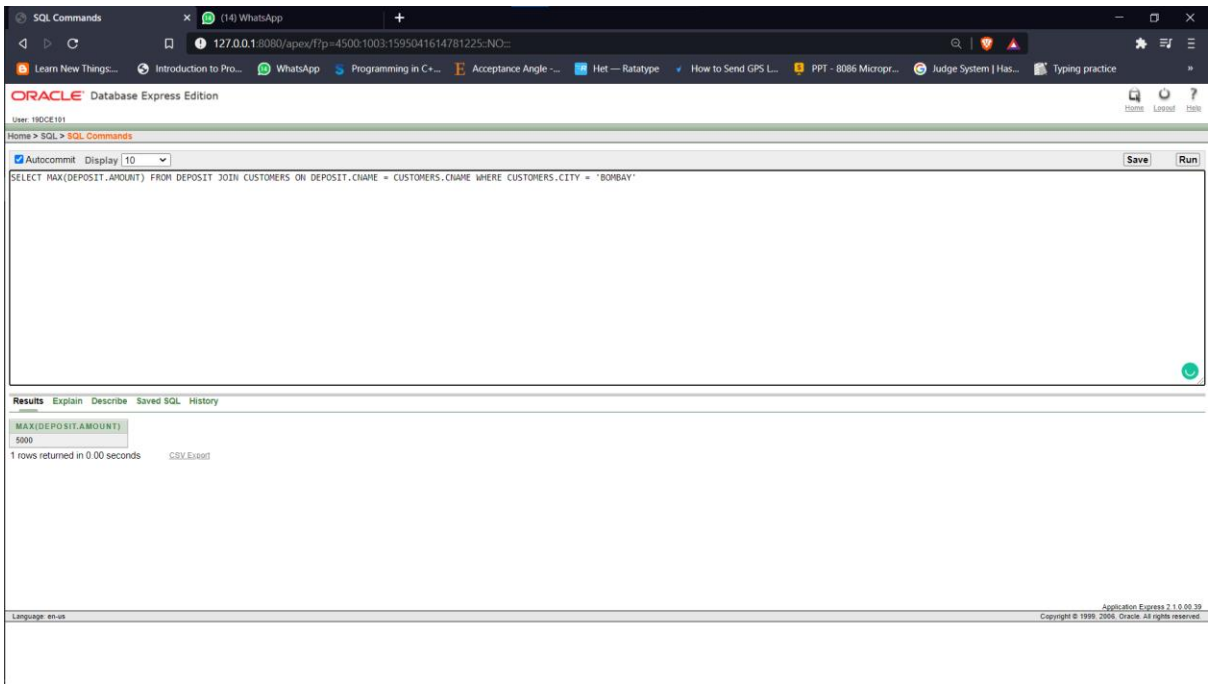
(2) List total deposit of customers living in city Nagpur.

Snap-Shot:



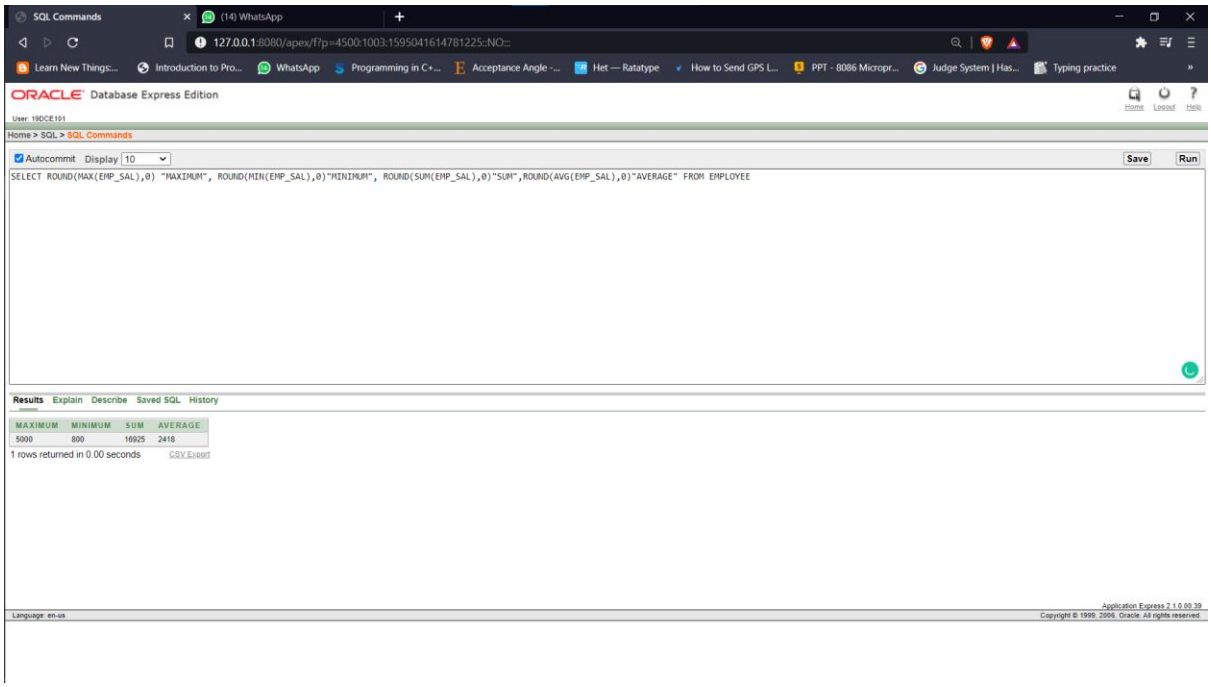
(3) List maximum deposit of customers living in Bombay.

Snap-Shot:



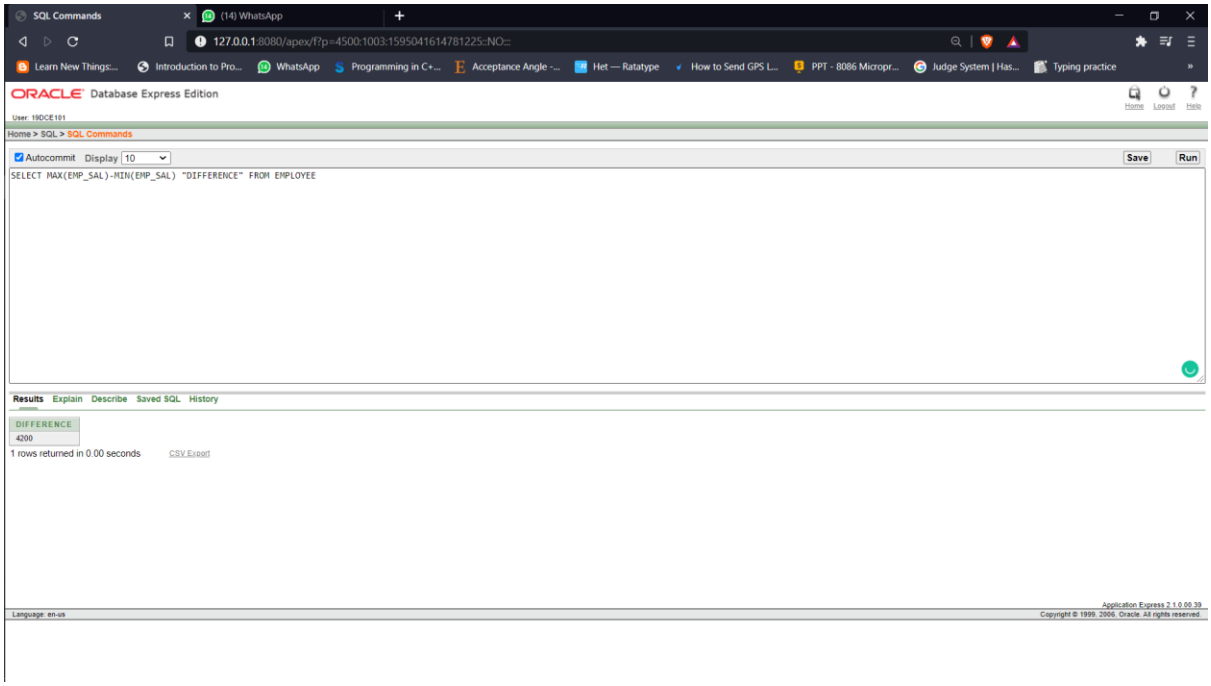
(4) Display the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number.

Snap-Shot:



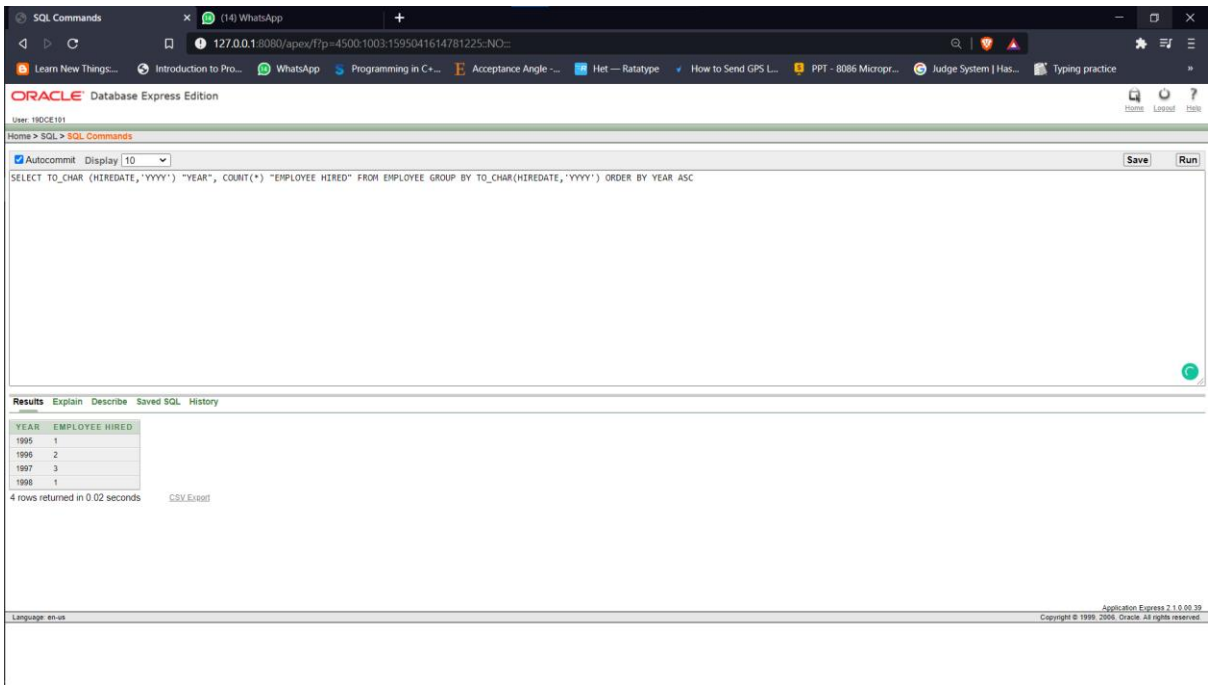
(5) Write a query that displays the difference between the highest and lowest salaries. Label the column DIFFERENCE.

Snap-Shot:



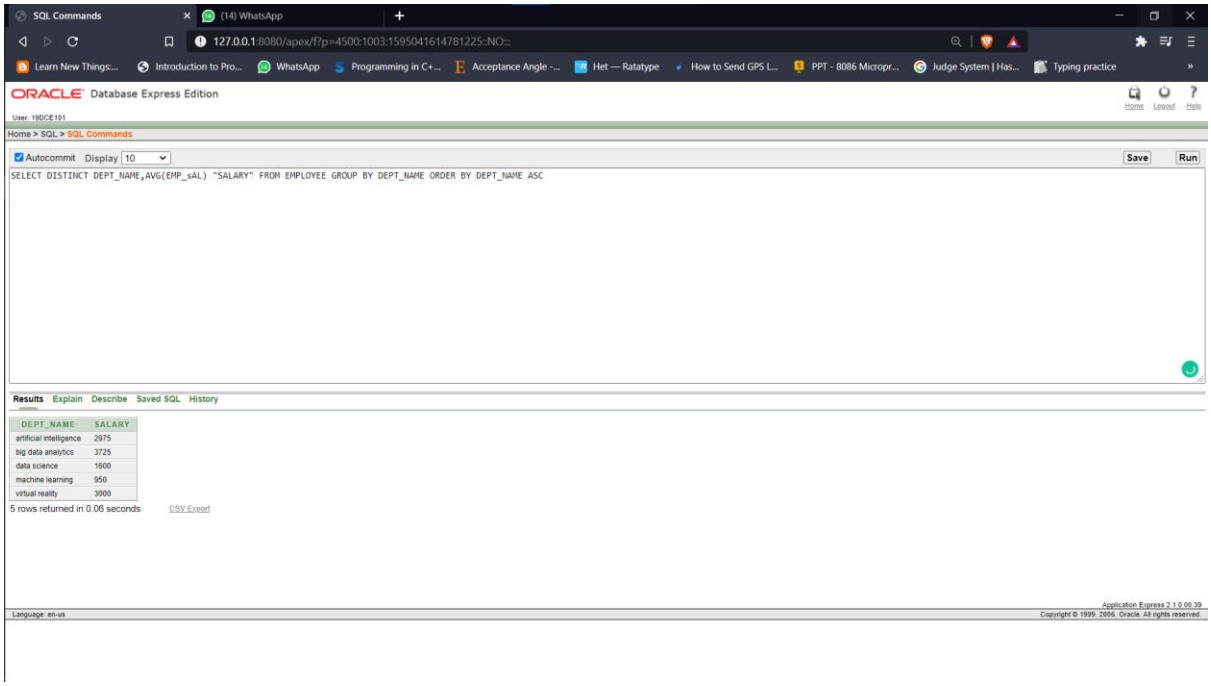
(6) Create a query that will display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998

Snap-Shot:



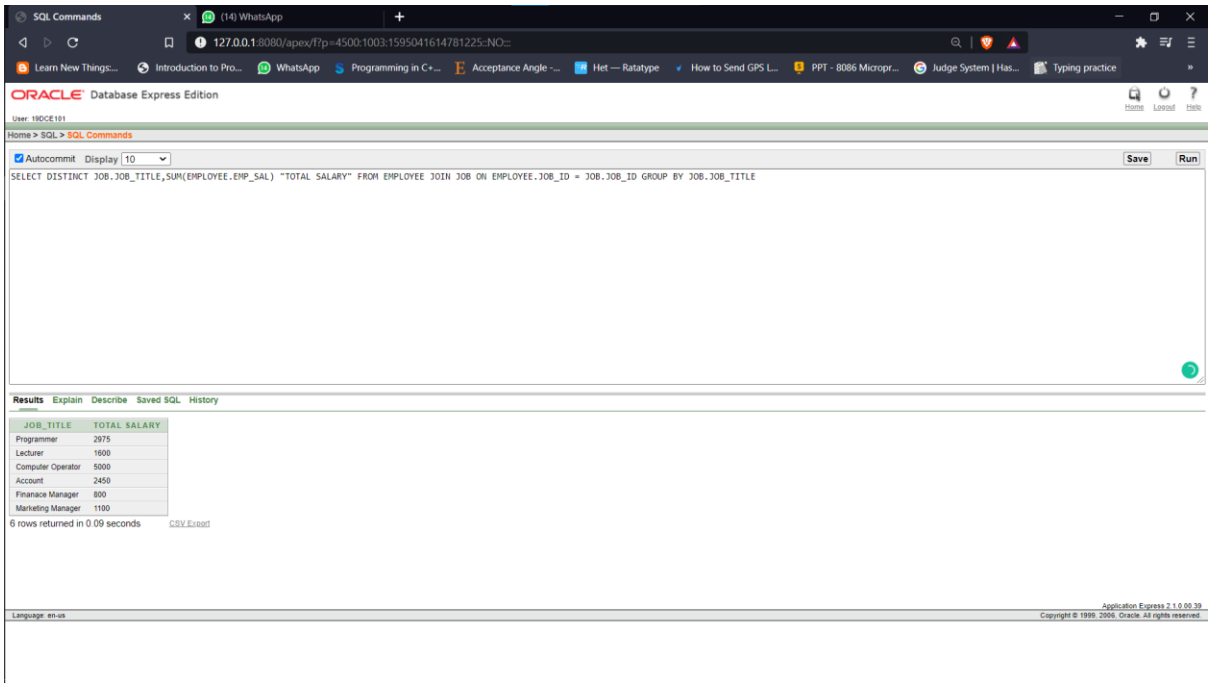
(7) Find the average salaries for each department without displaying the respective department numbers.

Snap-Shot:



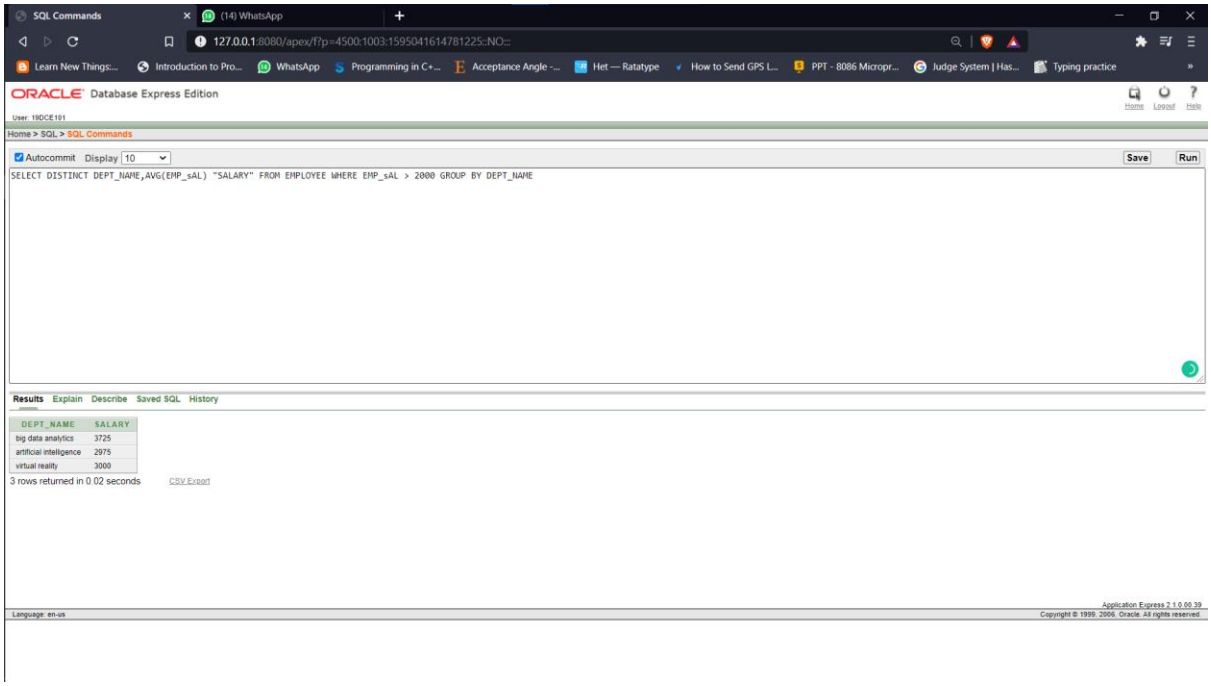
(8) Write a query to display the total salary being paid to each job title, within each department.

Snap-Shot:



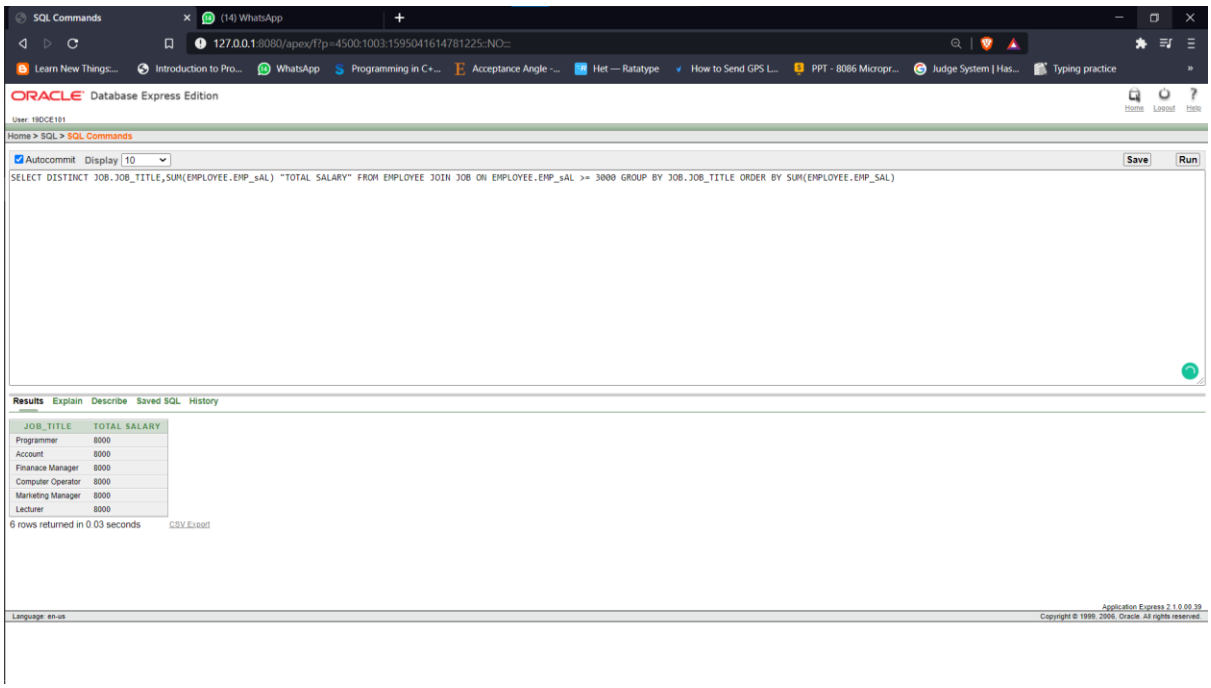
(9) Find the average salaries > 2000 for each department without displaying the respective department numbers.

Snap-Shot:



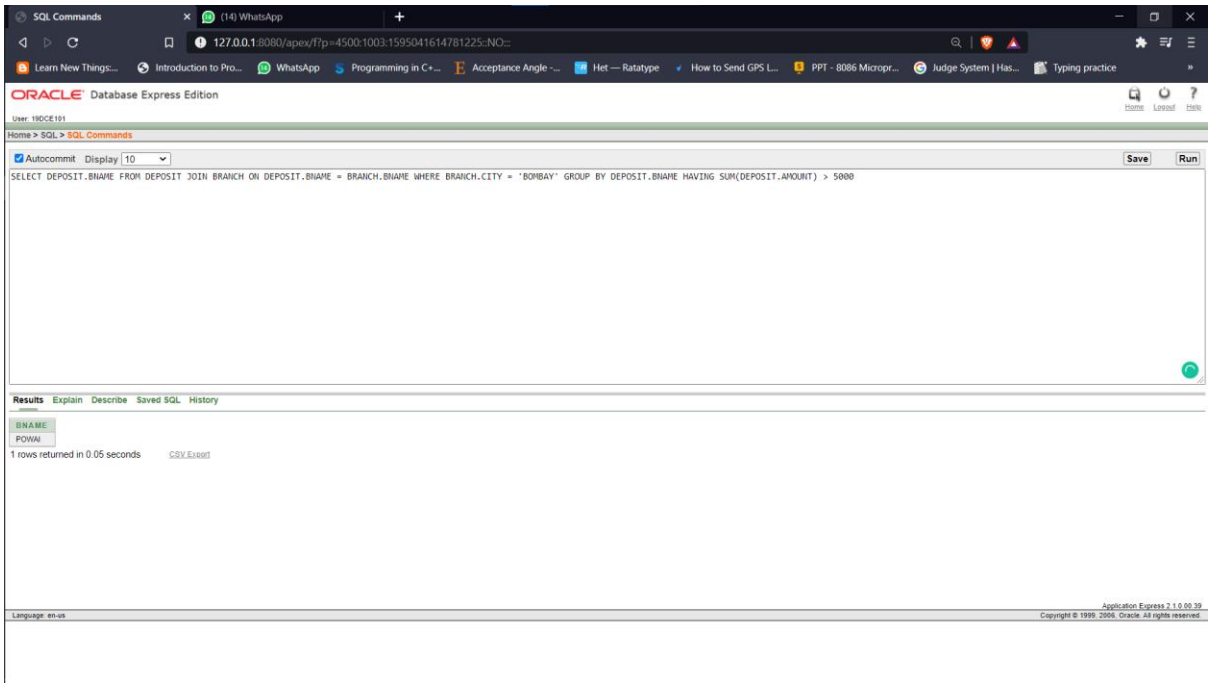
(10) Display the job and total salary for each job with a total salary amount exceeding 3000 and sorts the list by the total salary.

Snap-Shot:



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

Snap-Shot:



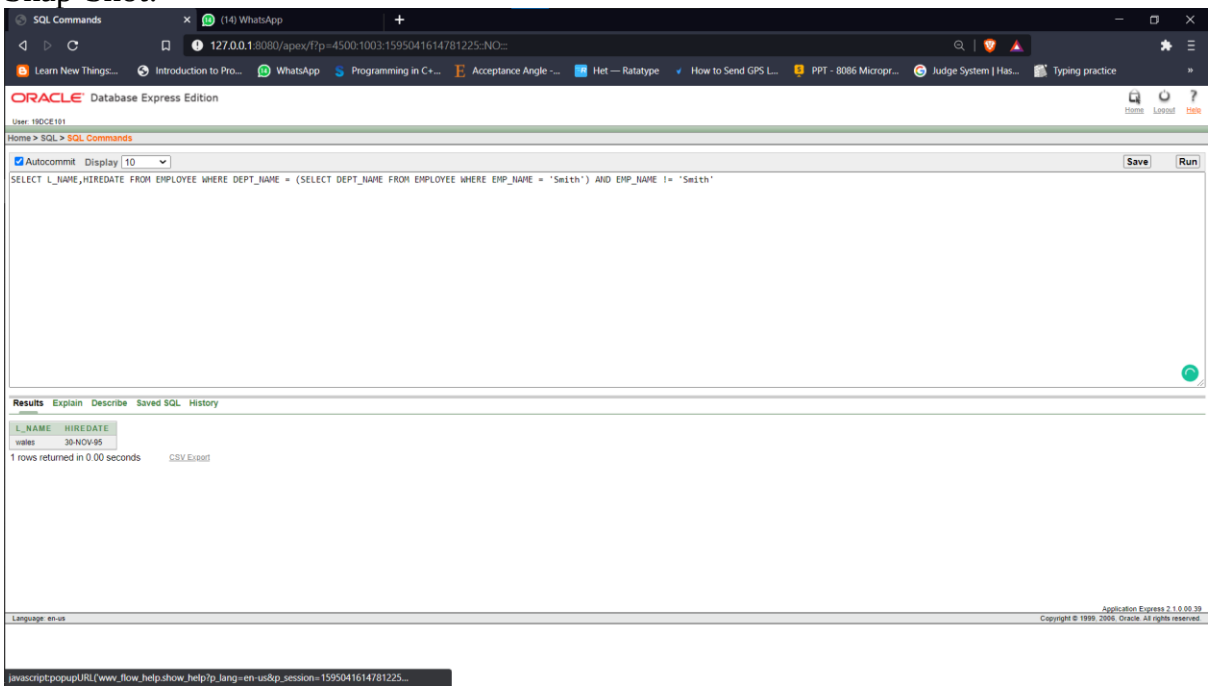
Conclusion: We Learned about Aggregating Data using Group functions.

Practical -9

To solve queries using the concept of sub query.

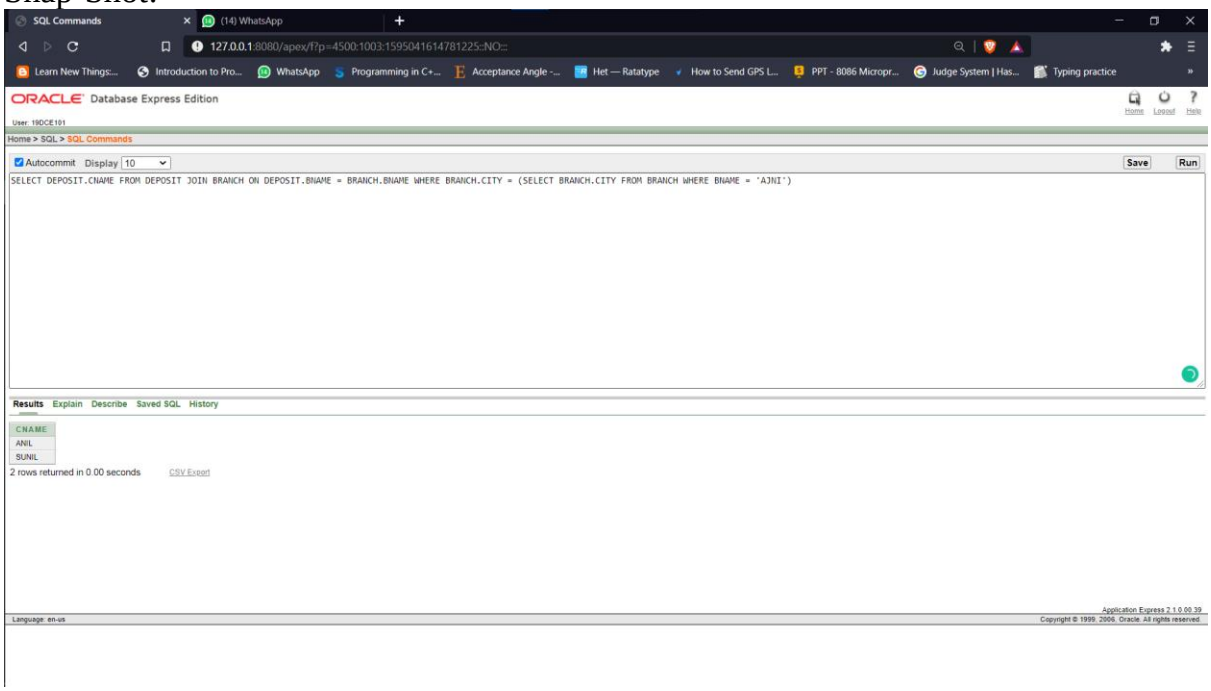
(1) Write a query to display the last name and hire date of any employee in the same department as smith. Exclude smith.

Snap-Shot:



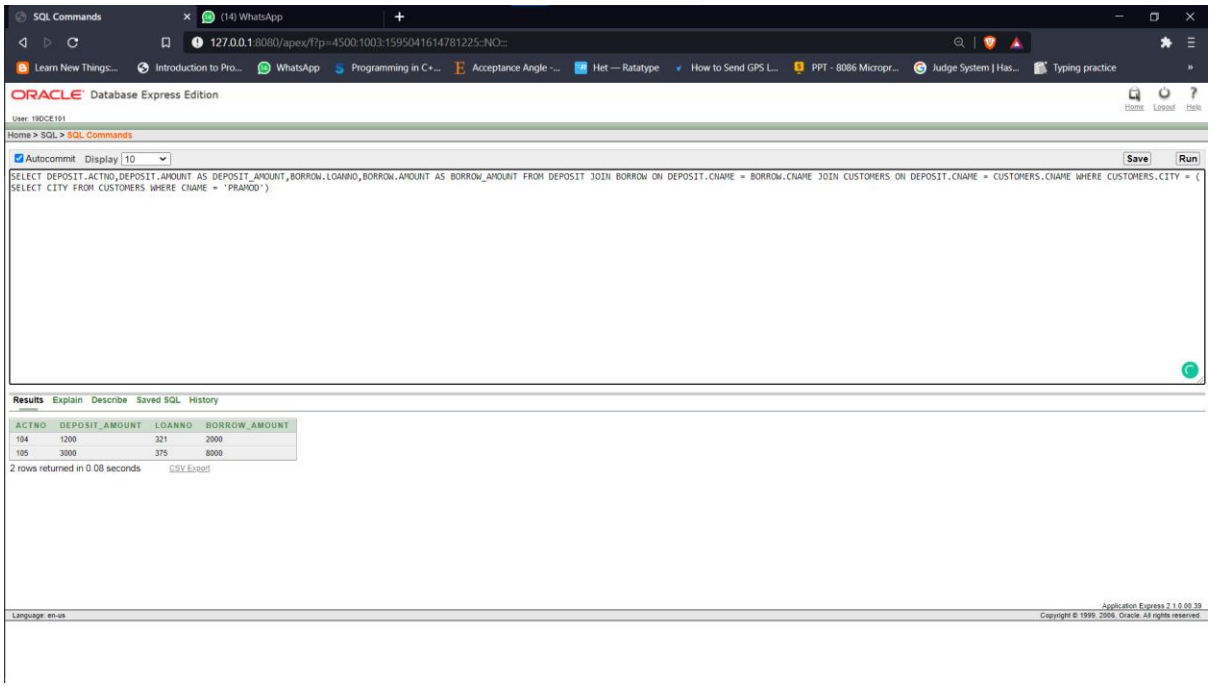
(2) Give name of customers who are depositors having same branch city of mr. sunil.

Snap-Shot:



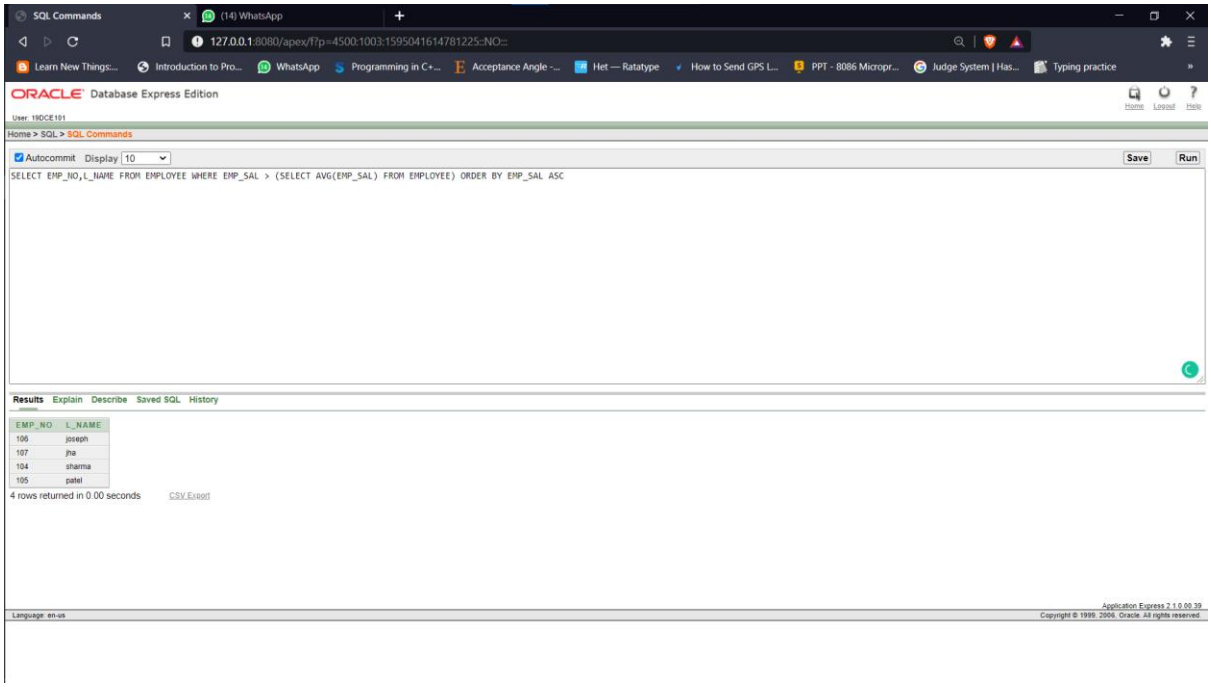
(3) Give deposit details and loan details of customer in same city where pramod is living.

Snap-Shot:



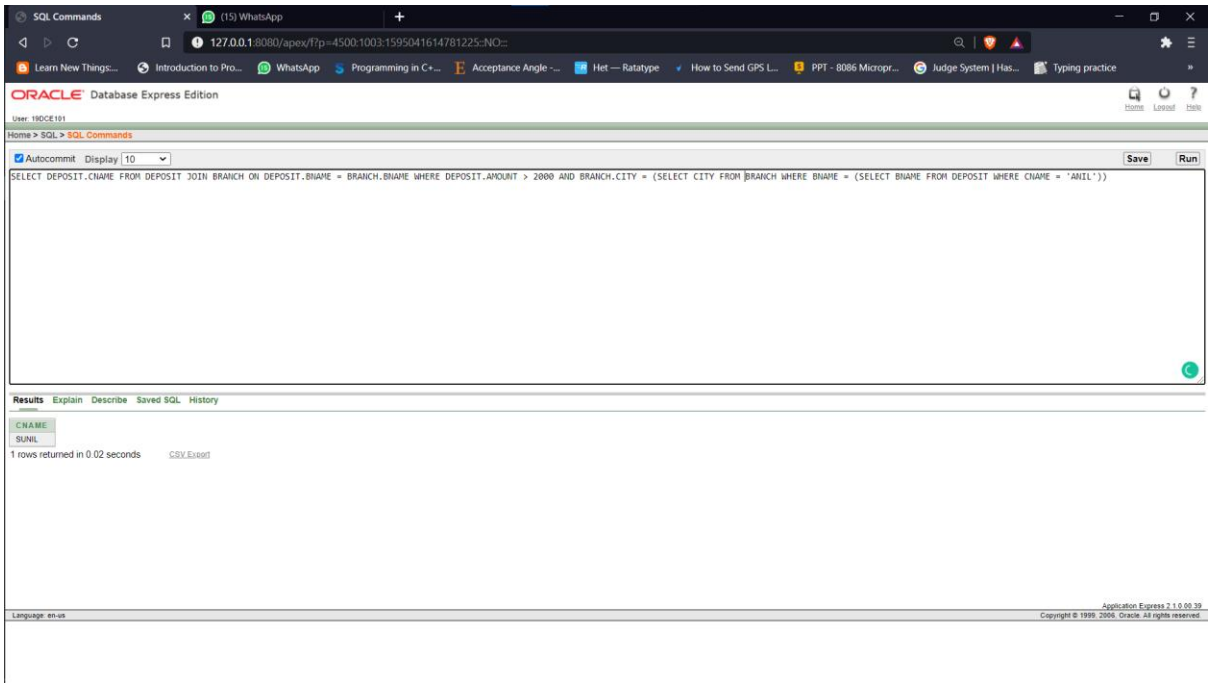
(4) Create a query to display the employee numbers and last names of all employees who earn more than the average salary. Sort the results in ascending order of salary.

Snap-Shot:



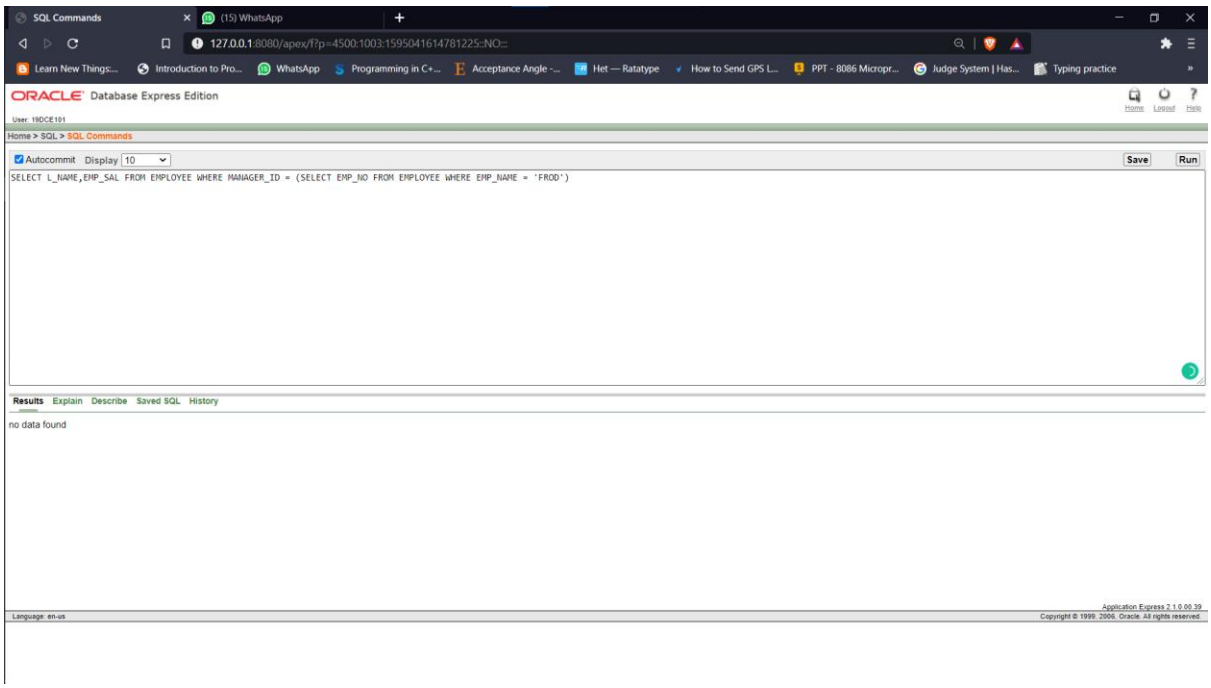
(5) Give names of depositors having same living city as mr. anil and having deposit amount greater than 2000

Snap-Shot:



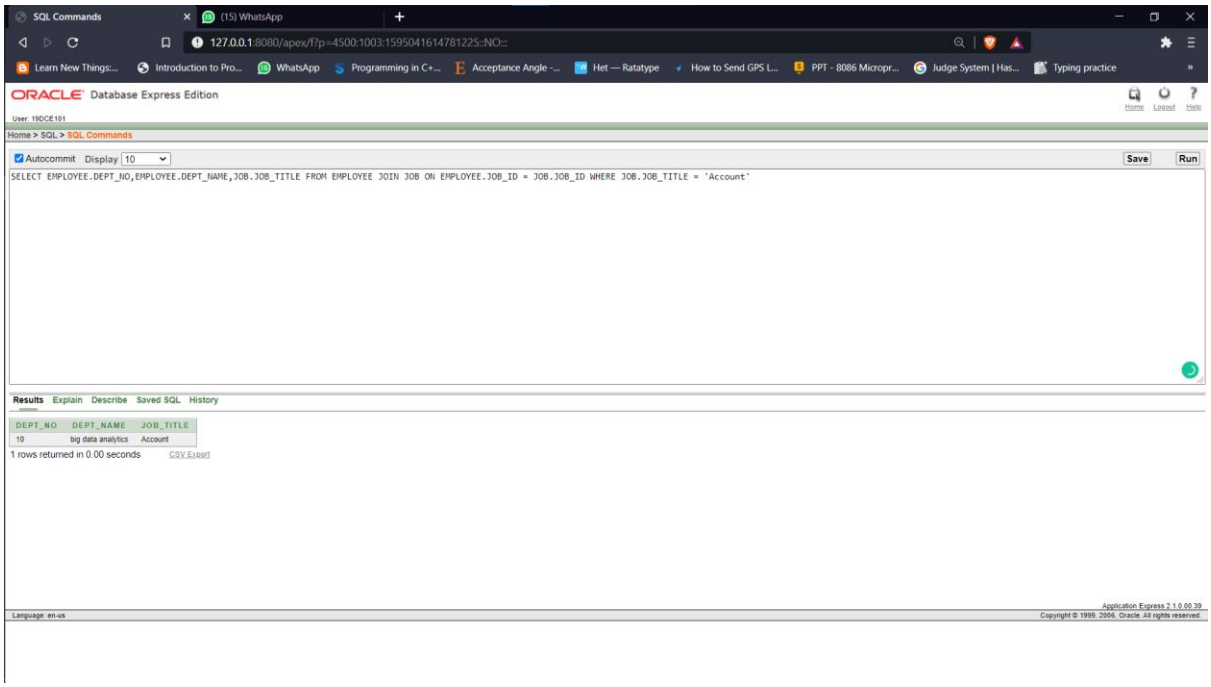
(6) Display the last name and salary of every employee who reports to ford.

Snap-Shot:



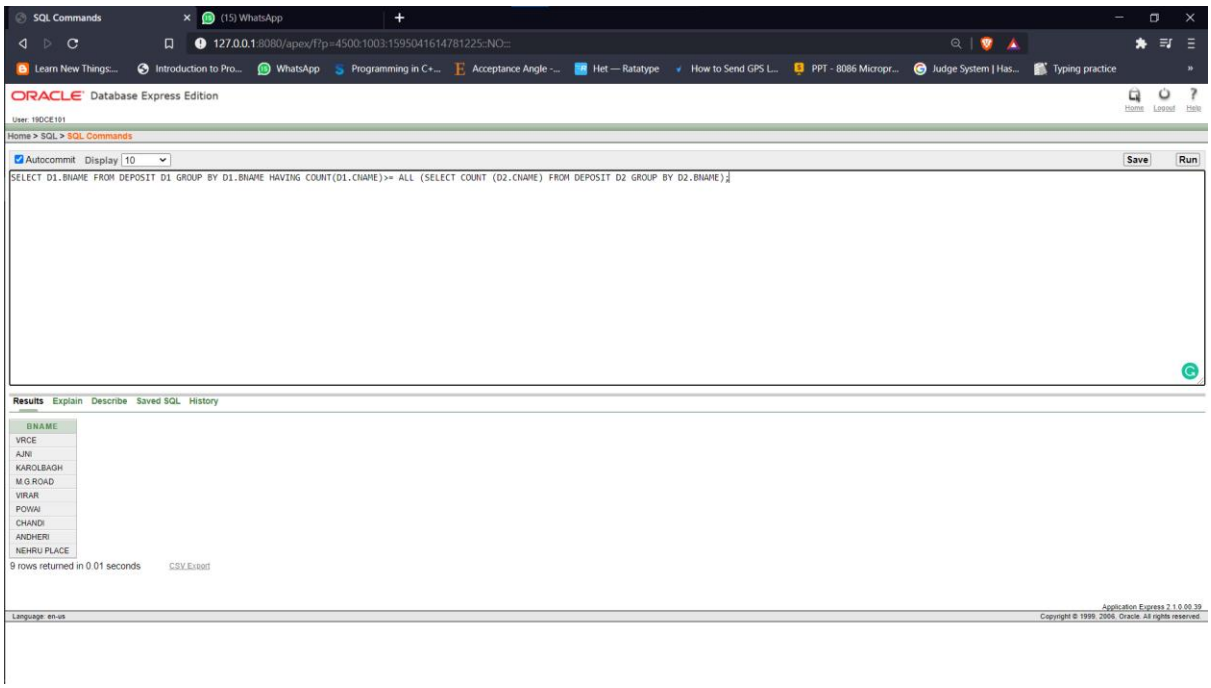
(7) Display the department number, name, and job for every employee in the Accounting department.

Snap-Shot:



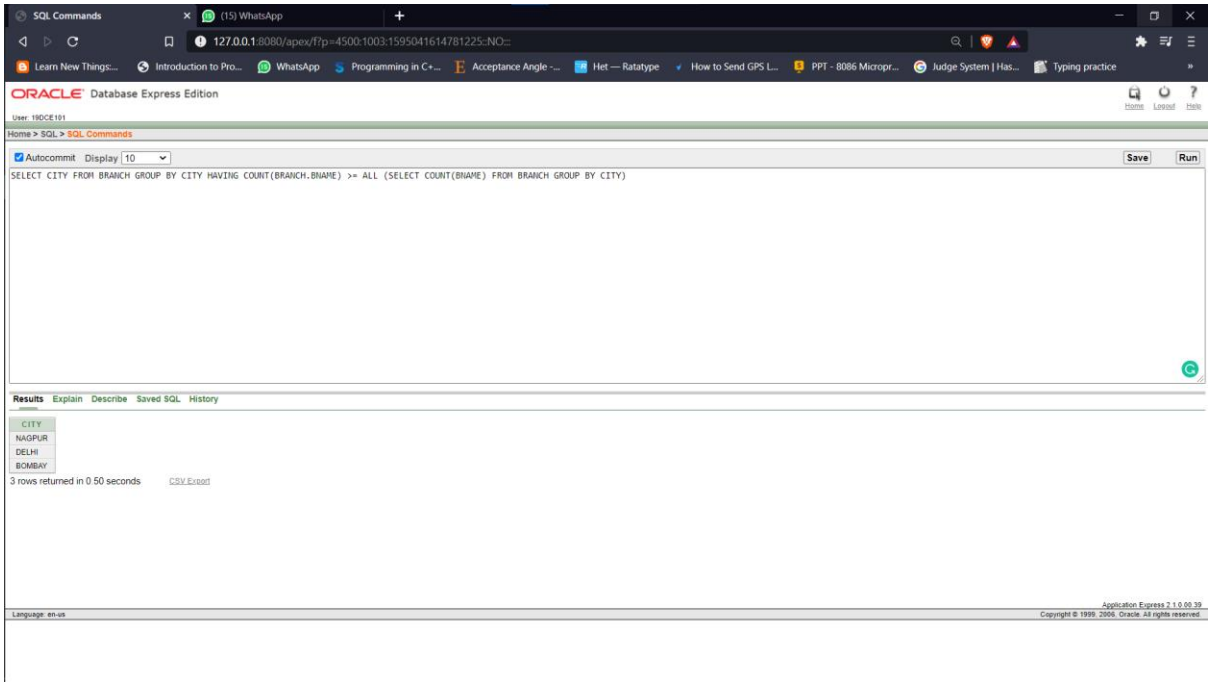
(8) List the name of branch having highest number of depositors.

Snap-Shot:



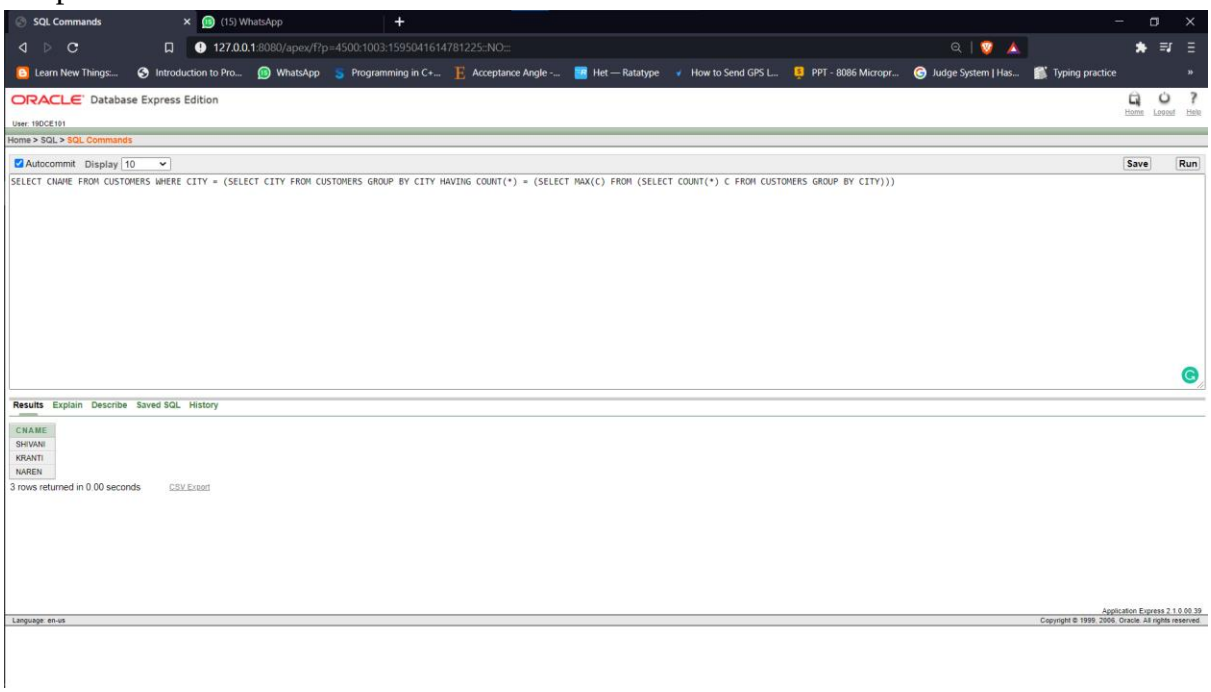
(9) Give the name of cities where in which the maximum numbers of branches are located.

Snap-Shot:



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

Snap-Shot:



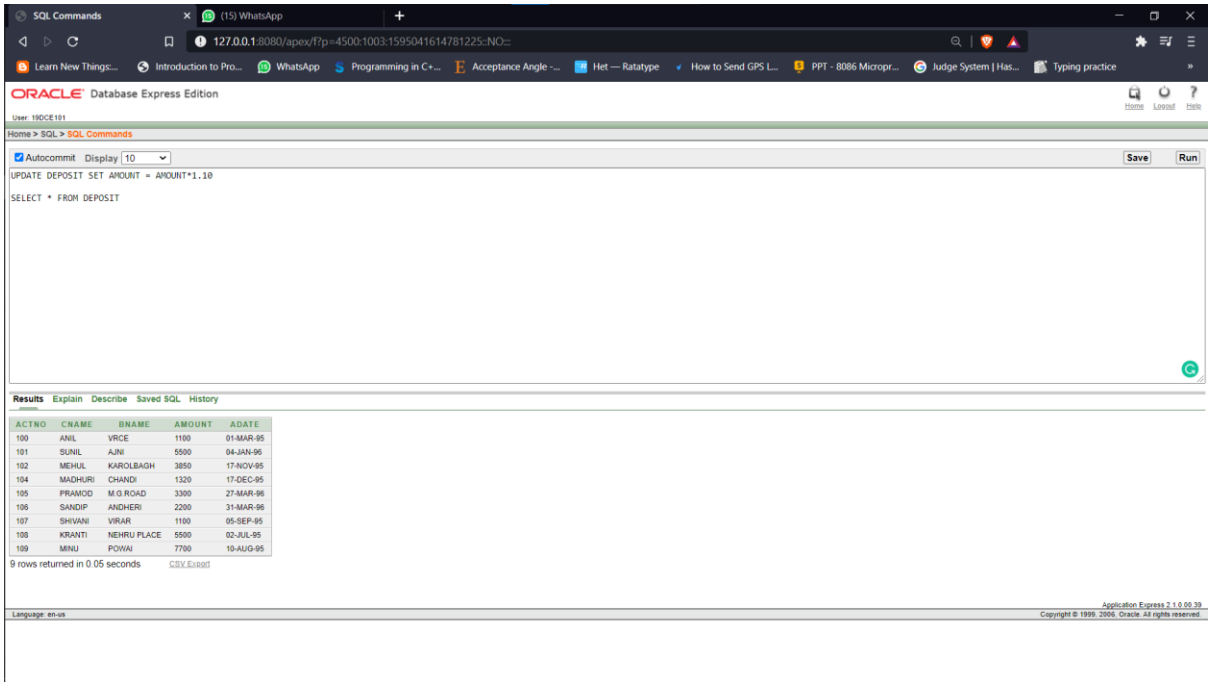
Conclusion: From This practical I learned how to perform sub-queries.

Practical -10

Manipulating Data

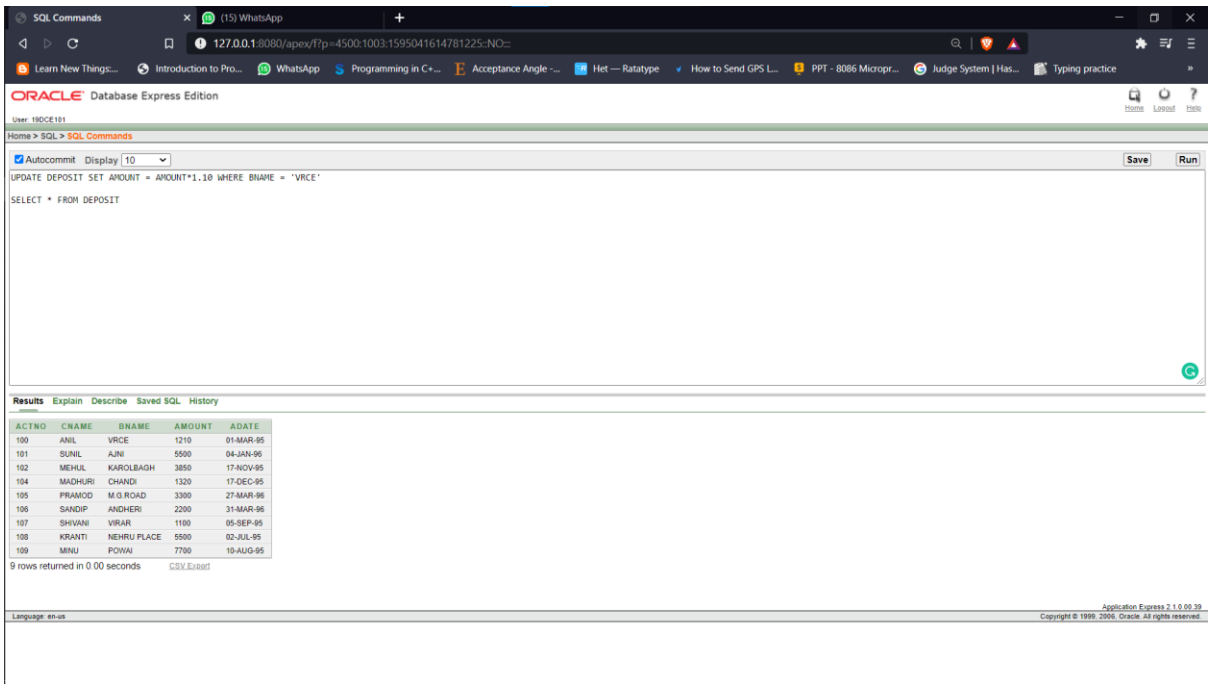
(1) Give 10% interest to all depositors.

Snap-Shot:



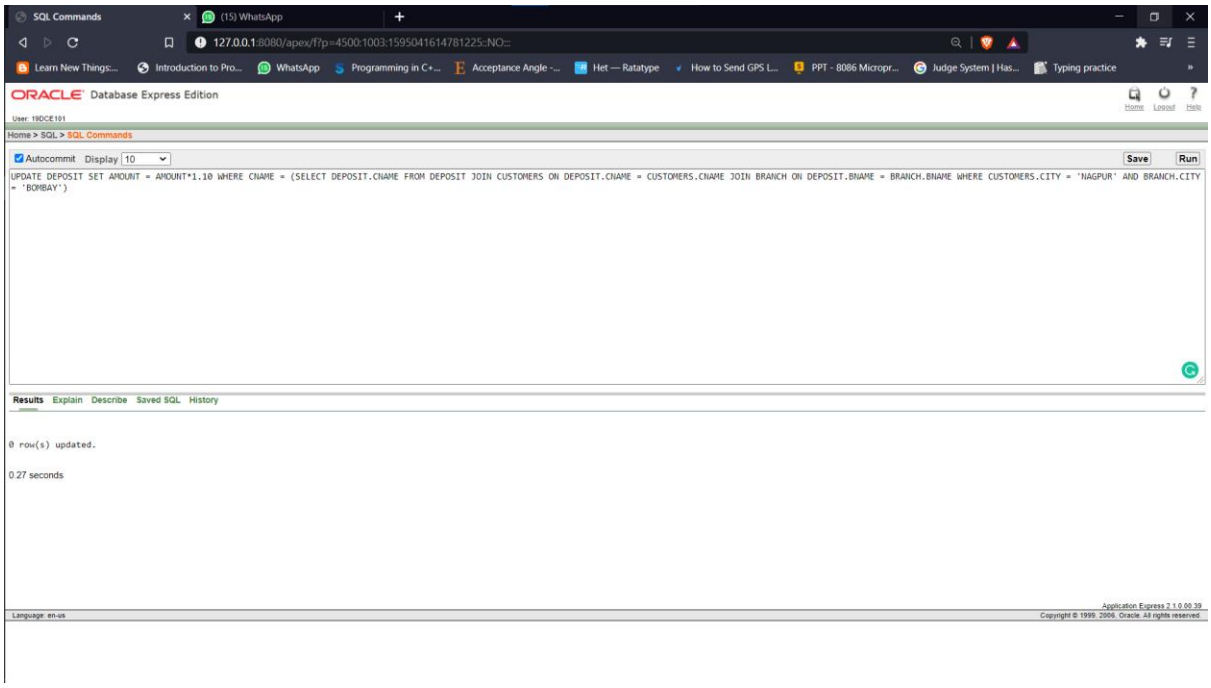
(2) Give 10% interest to all depositors having branch vrce.

Snap-Shot:



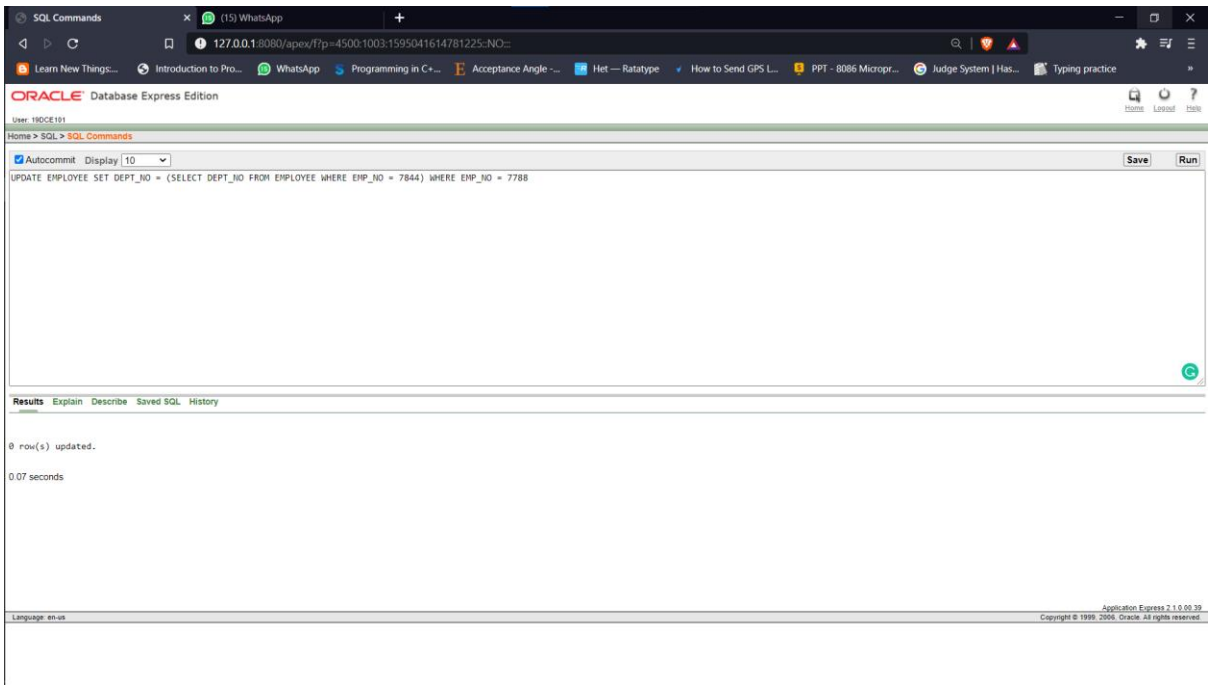
(3) Give 10% interest to all depositors living in Nagpur and having branch city Bombay.

Snap-Shot:



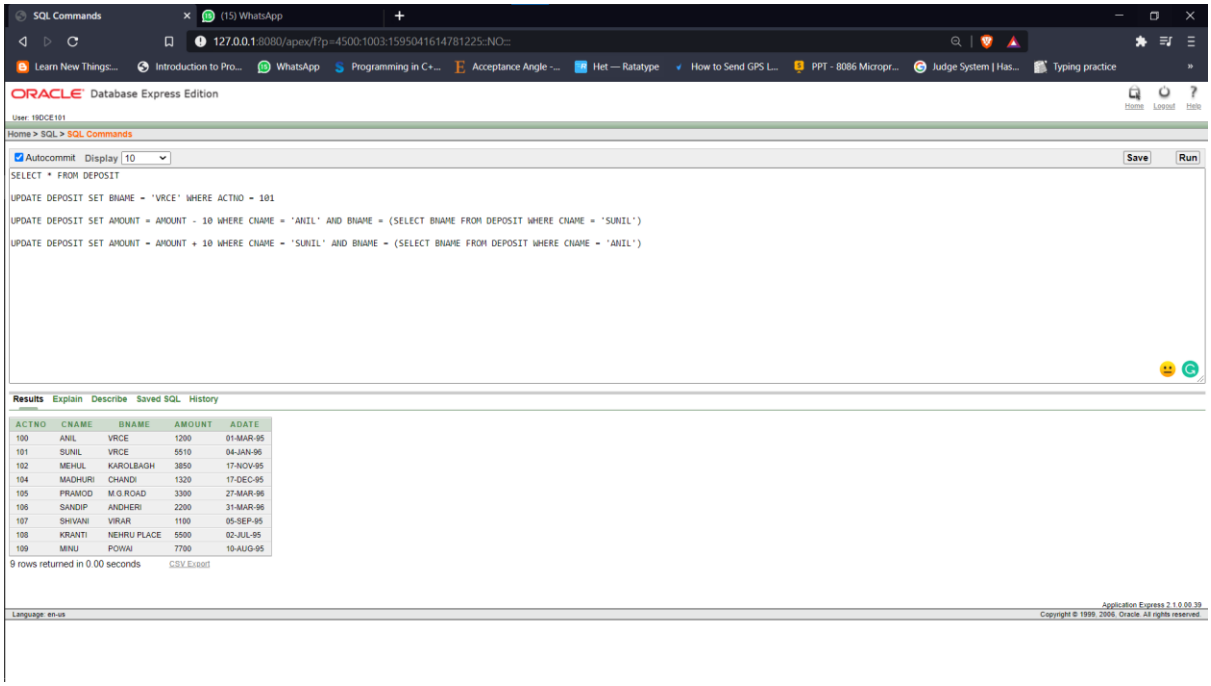
(4) Write a query which changes the department number of all employees with empno 7788's job to employee 7844'current department number.

Snap-Shot:



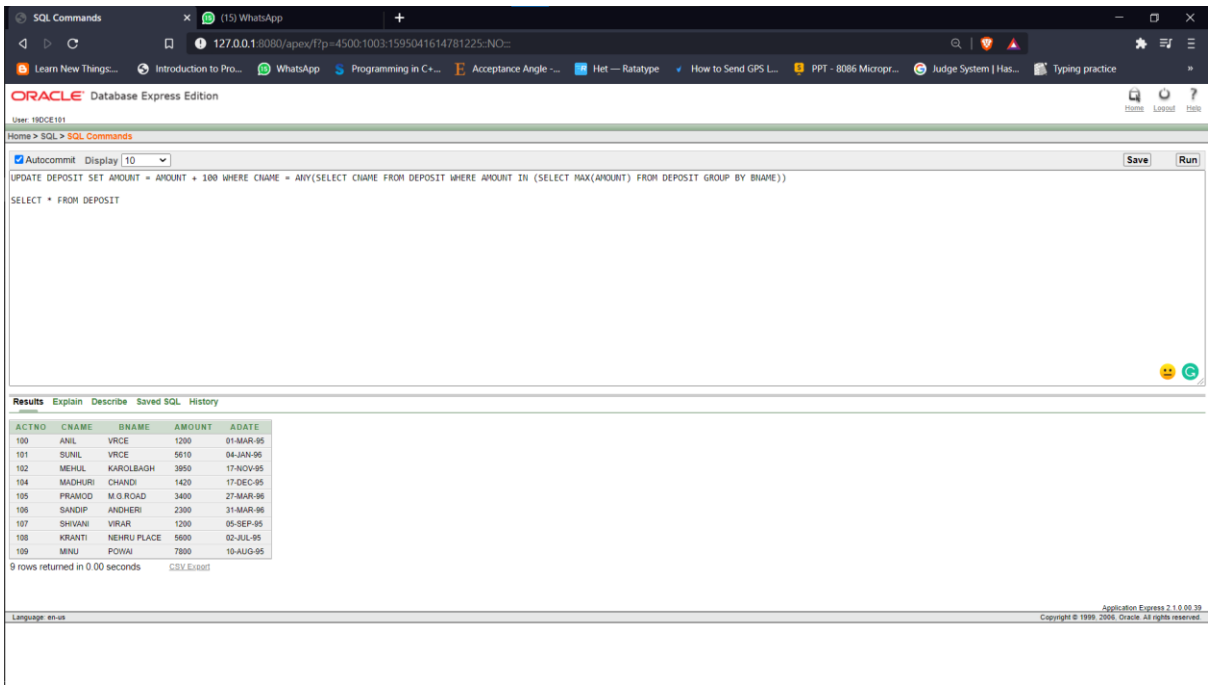
(5) Transfer 10 Rs from account of anil to Sunil if both are having same branch.

Snap-Shot:



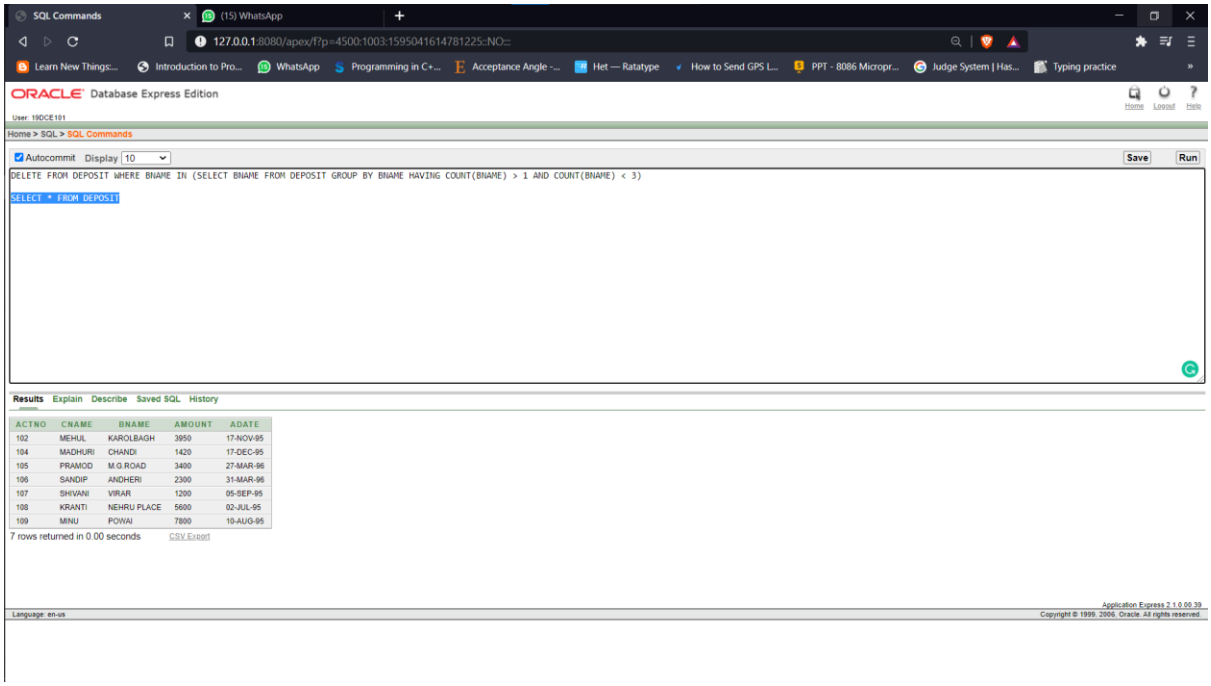
(6) Give 100 Rs more to all depositors if they are maximum depositors in their respective branch.

Snap-Shot:

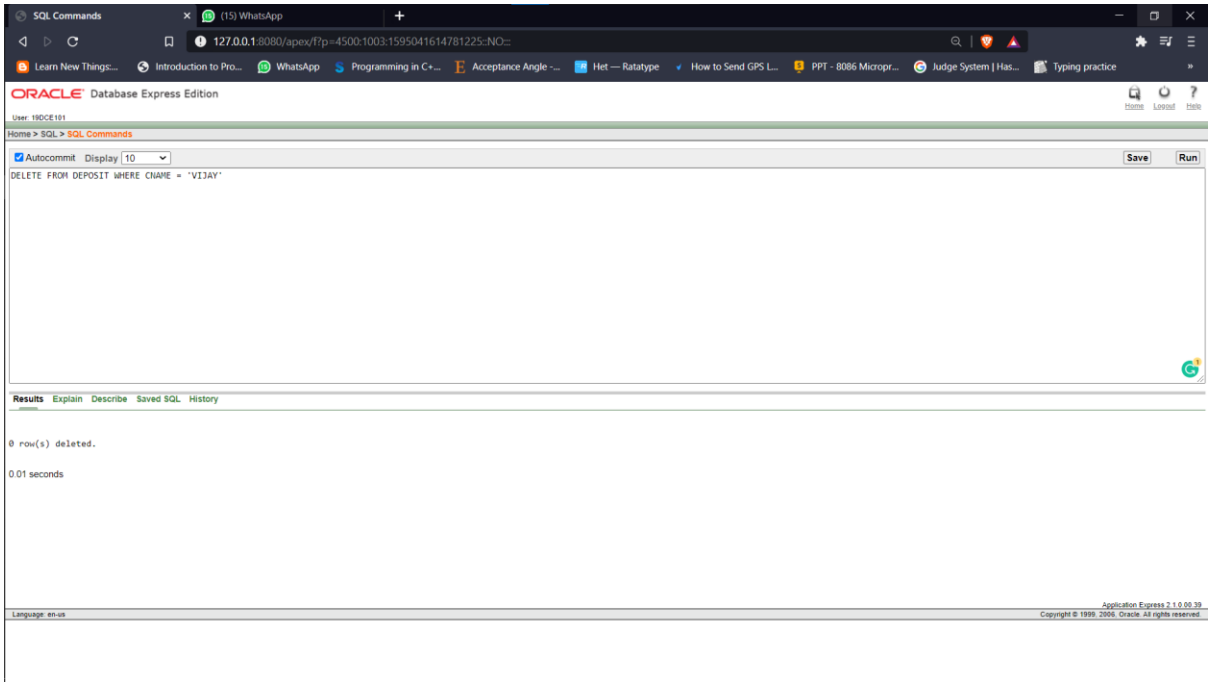


(7) Delete depositors of branches having number of customers between 1 to 3.

Snap-Shot:

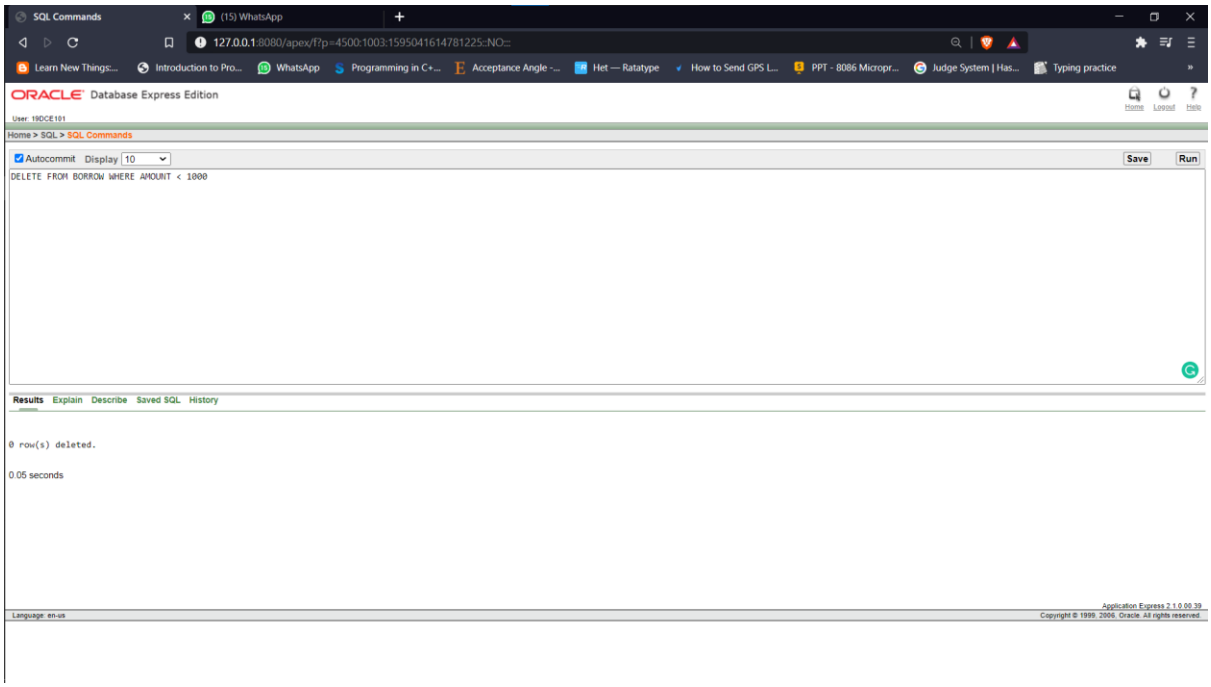


(8) Delete deposit of vijay.
Snap-Shot:



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

Snap-Shot:

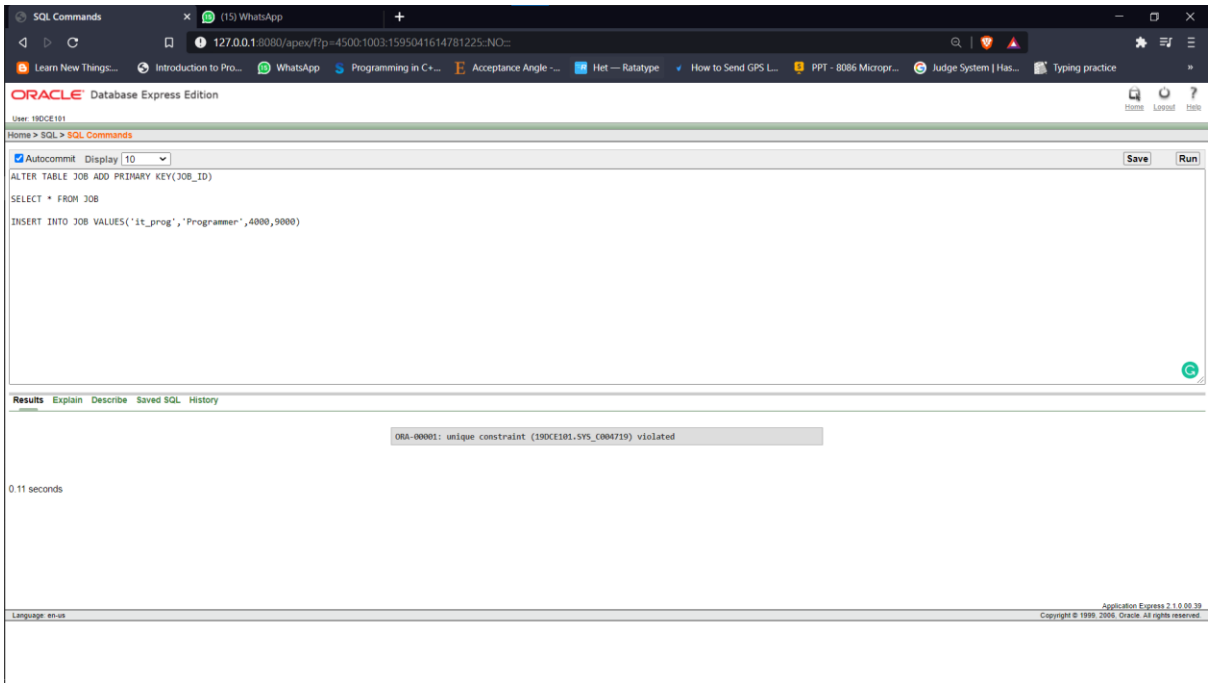


Conclusion: From this practical I learned how to manipulate data.

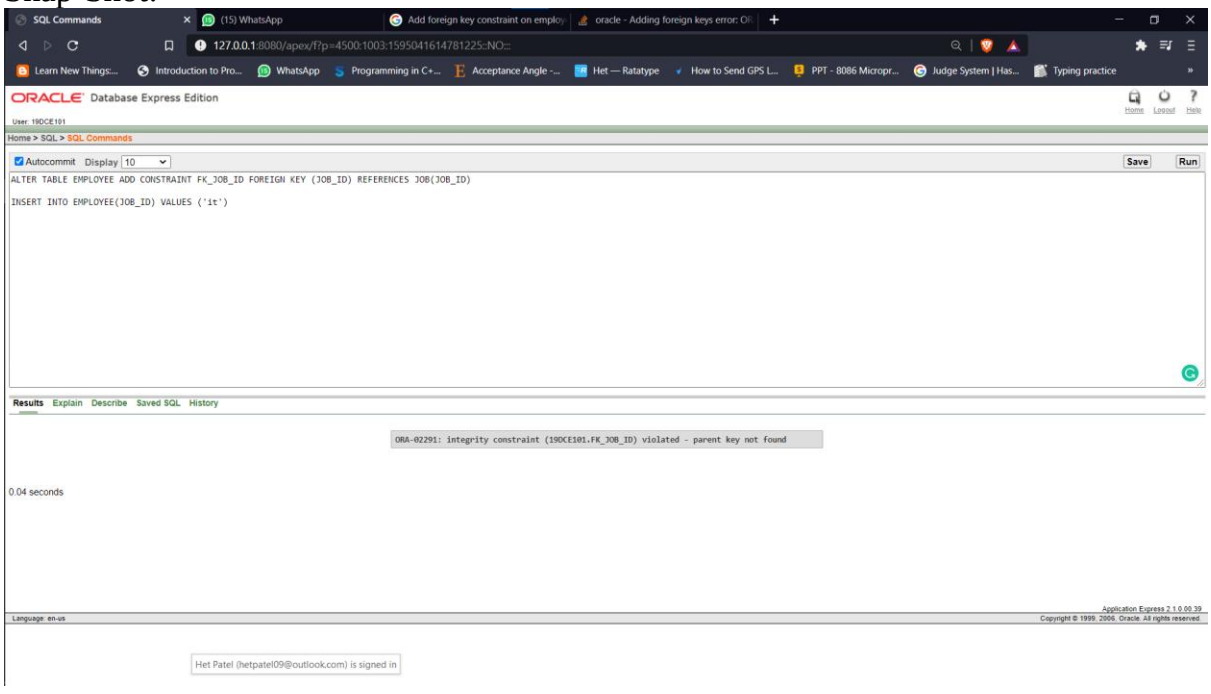
Practical -11

Add and Remove constraint

(1) Add primary key constraint on job_id in job table.
Snap-Shot:

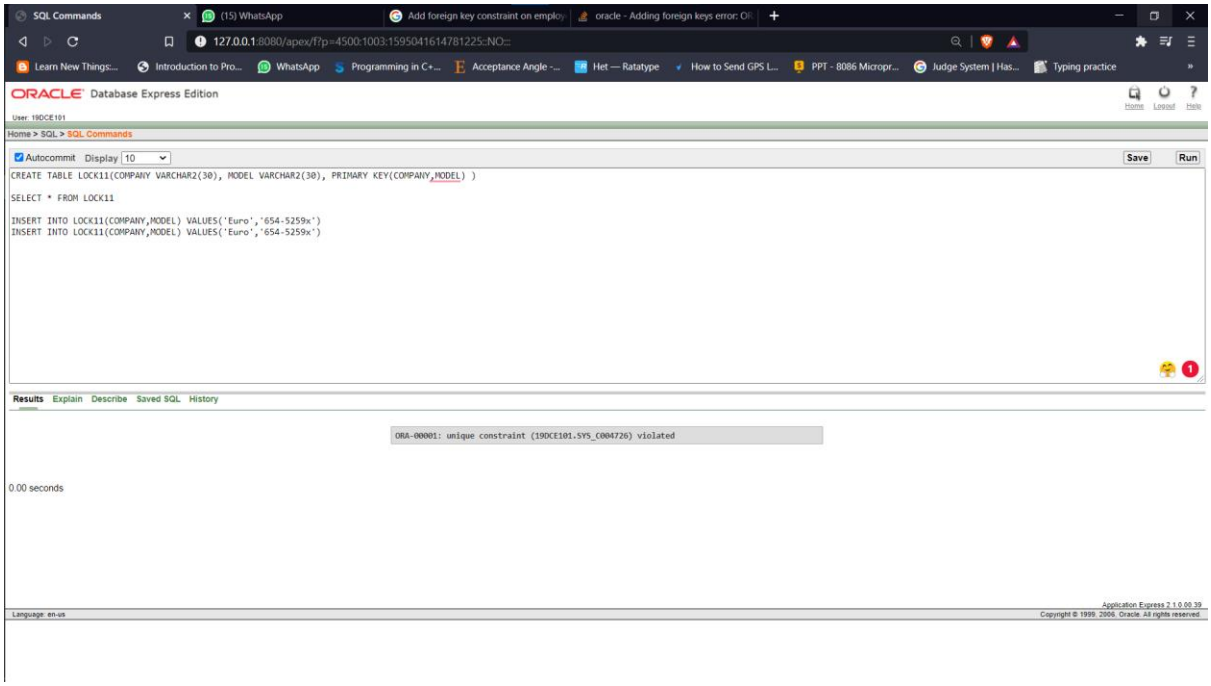


(2) Add foreign key constraint on employee table referencing job table.
Snap-Shot:



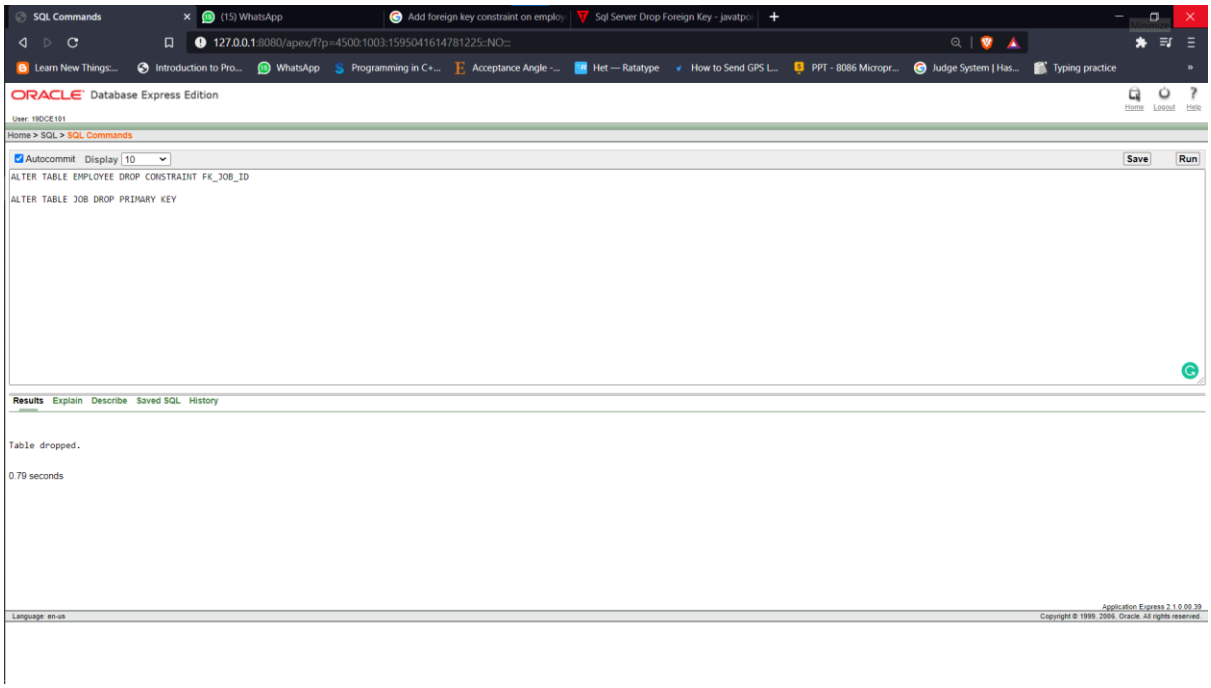
(3) Add composite primary key on lock table (lock table does not exist, while creating table add composite key)

Snap-Shot:



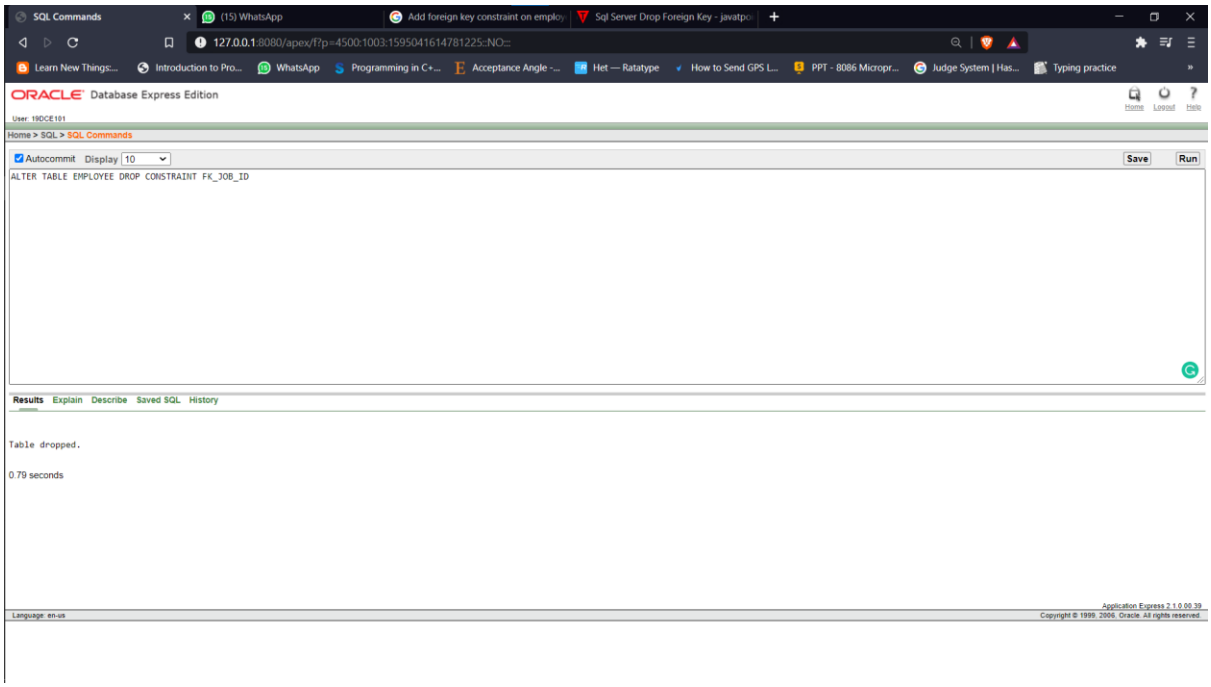
(4) Remove primary key constraint on job_id

Snap-Shot:



(5) Remove foreign key constraint on employee table

Snap-Shot:

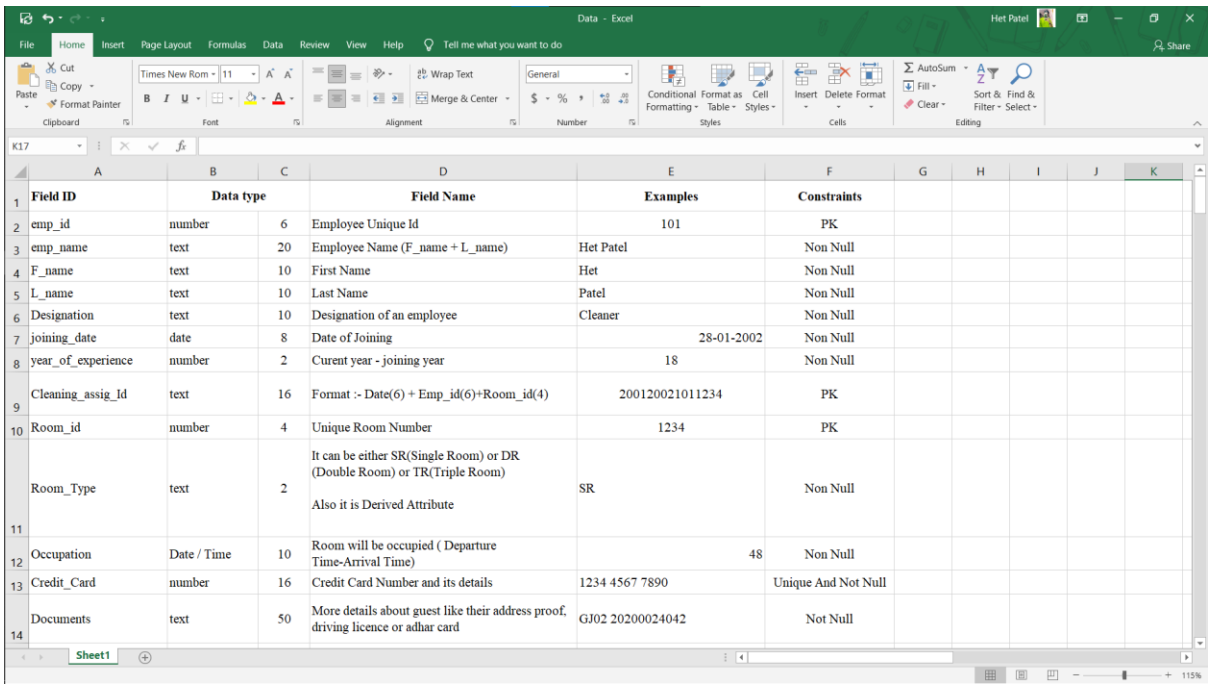


Conclusion: From this practical I learned how to remove and add constraints on a table.

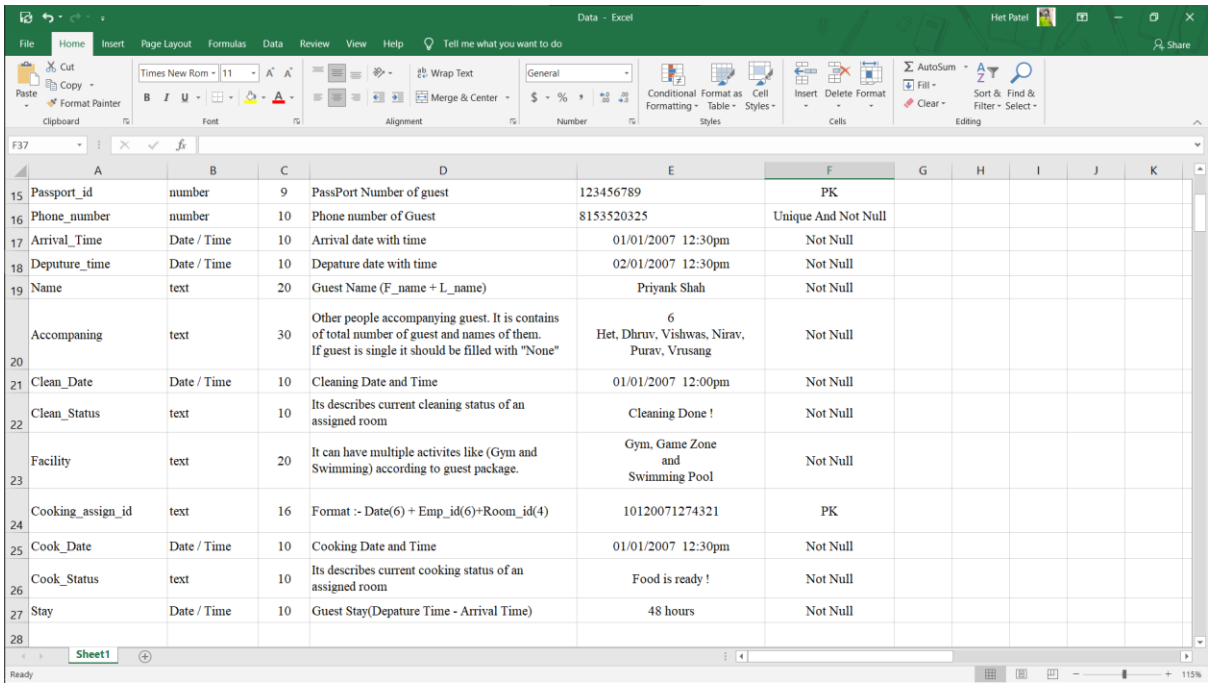
Practical -12

(12.1) Data Dictionary

Snap-shot:



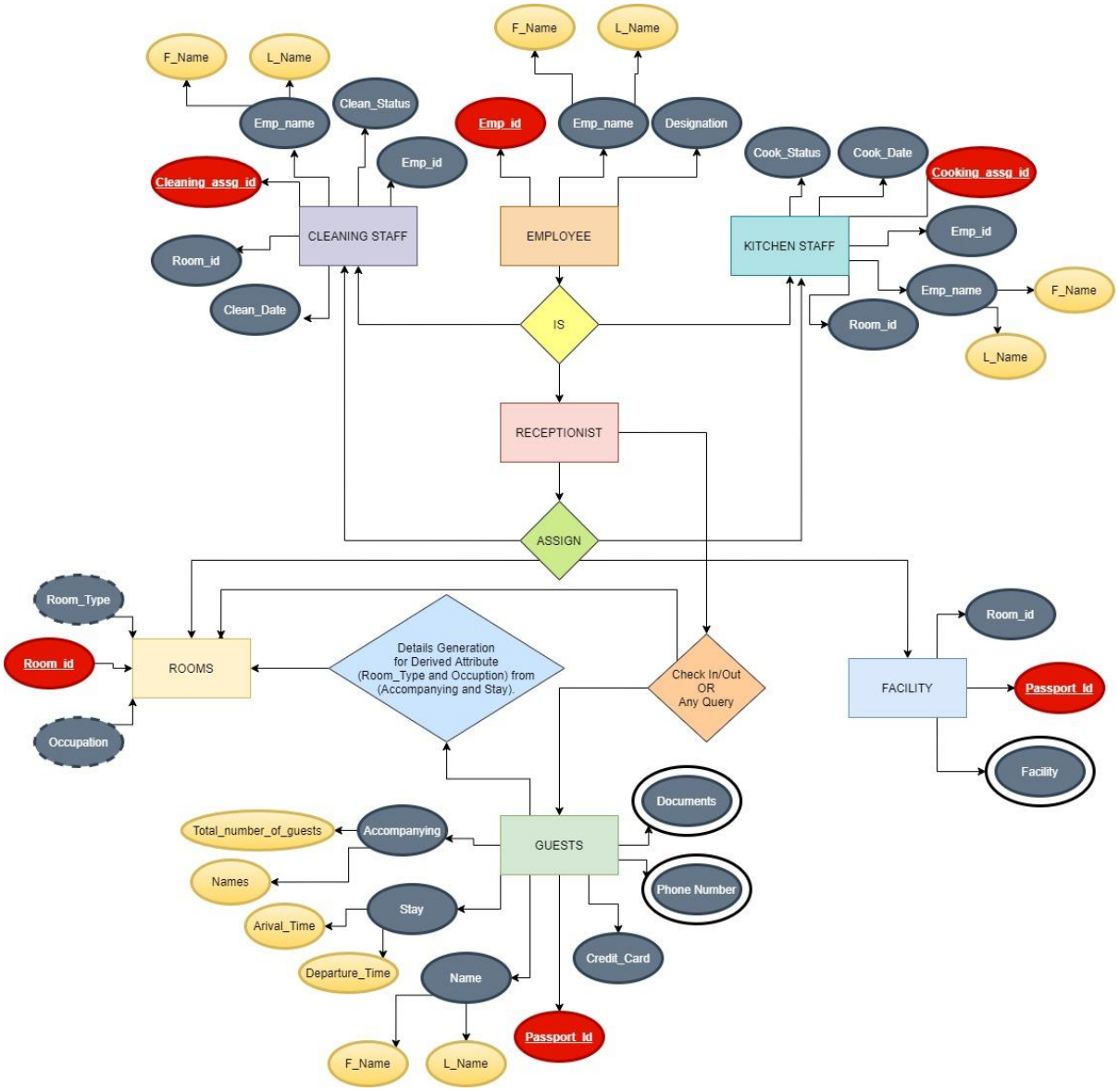
Field ID	Data type		Field Name	Examples	Constraints
emp_id	number	6	Employee Unique Id	101	PK
emp_name	text	20	Employee Name (F_name + L_name)	Het Patel	Non Null
F_name	text	10	First Name	Het	Non Null
L_name	text	10	Last Name	Patel	Non Null
Designation	text	10	Designation of an employee	Cleaner	Non Null
joining_date	date	8	Date of Joining	28-01-2002	Non Null
year_of_experience	number	2	Curent year - joining year	18	Non Null
Cleaning_assig_Id	text	16	Format :- Date(6) + Emp_id(6)+Room_id(4)	200120021011234	PK
Room_id	number	4	Unique Room Number	1234	PK
Room_Type	text	2	It can be either SR(Single Room) or DR (Double Room) or TR(Triple Room) Also it is Derived Attribute	SR	Non Null
Occupation	Date / Time	10	Room will be occupied (Departure Time-Arrival Time)	48	Non Null
Credit_Card	number	16	Credit Card Number and its details	1234 4567 7890	Unique And Not Null
Documents	text	50	More details about guest like their address proof, driving licence or adhar card	GJ02 20200024042	Not Null



Passport_id	number	9	PassPort Number of guest	123456789	PK
Phone_number	number	10	Phone number of Guest	8153520325	Unique And Not Null
Arrival_Time	Date / Time	10	Arrival date with time	01/01/2007 12:30pm	Not Null
Deputure_time	Date / Time	10	Depature date with time	02/01/2007 12:30pm	Not Null
Name	text	20	Guest Name (F_name + L_name)	Priyank Shah	Not Null
Accompaning	text	30	Other people accompanying guest. It is contains of total number of guest and names of them. If guest is single it should be filled with "None"	6 Het, Dhruv, Vishwas, Nirav, Purav, Vrusang	Not Null
Clean_Date	Date / Time	10	Cleaning Date and Time	01/01/2007 12:00pm	Not Null
Clean_Status	text	10	Its describes current cleaning status of an assigned room	Cleaning Done !	Not Null
Facility	text	20	It can have multiple activites like (Gym and Swimming) according to guest package.	Gym, Game Zone and Swimming Pool	Not Null
Cooking_assign_id	text	16	Format :- Date(6) + Emp_id(6)+Room_id(4)	10120071274321	PK
Cook_Date	Date / Time	10	Cooking Date and Time	01/01/2007 12:30pm	Not Null
Cook_Status	text	10	Its describes current cooking status of an assigned room	Food is ready !	Not Null
Stay	Date / Time	10	Guest Stay(Depature Time - Arrival Time)	48 hours	Not Null

(12.2) E-R Diagram.

Snap-shot:



Conclusion: From This Practical I Learned how to make an E-R Diagram.

Practical -13

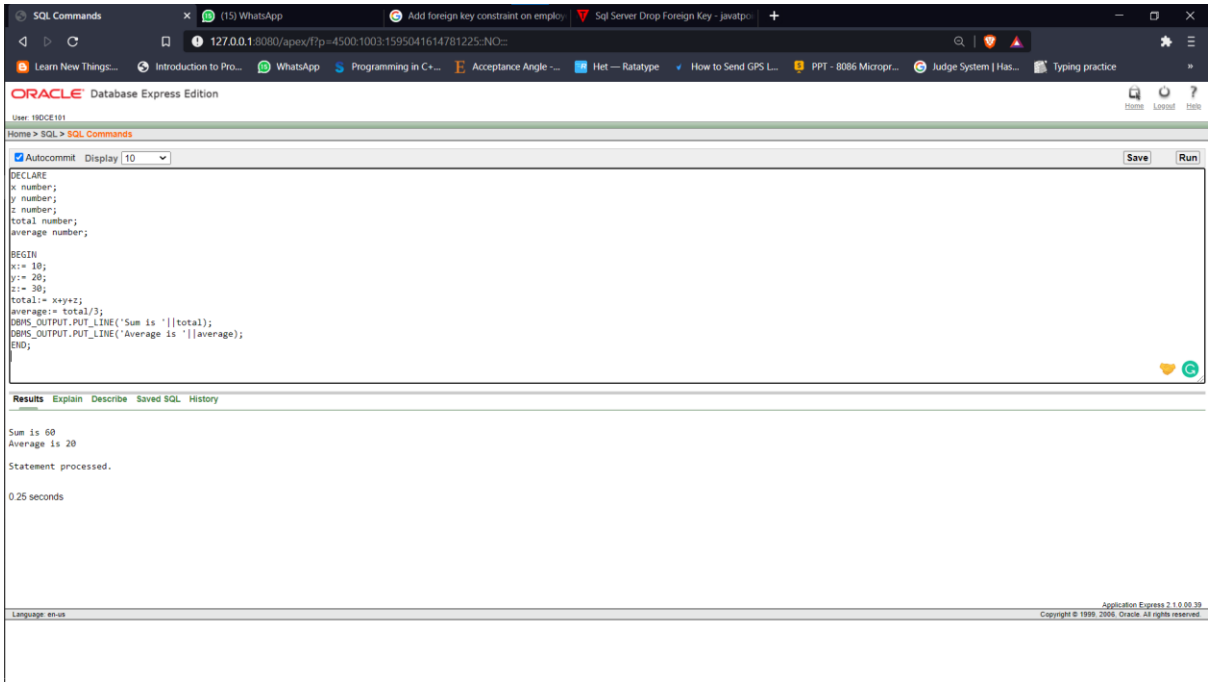
To perform basic PL/SQL blocks

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

Code:

```
DECLARE
x number;
y number;
z number;
total number;
average number;

BEGIN
x:= 10;
y:= 20;
z:= 30;
total:= x+y+z;
average:= total/3;
DBMS_OUTPUT.PUT_LINE('Sum is '||total);
DBMS_OUTPUT.PUT_LINE('Average is '||average);
END;
```

Snap-Shot:



Conclusion: From this practical I learned How to write code in PL/SQL.

Practical -14

To perform the concept of loop

Find the factorial of a number in pl/sql using for, While and Simple Loop.

Code:

Basic Loop

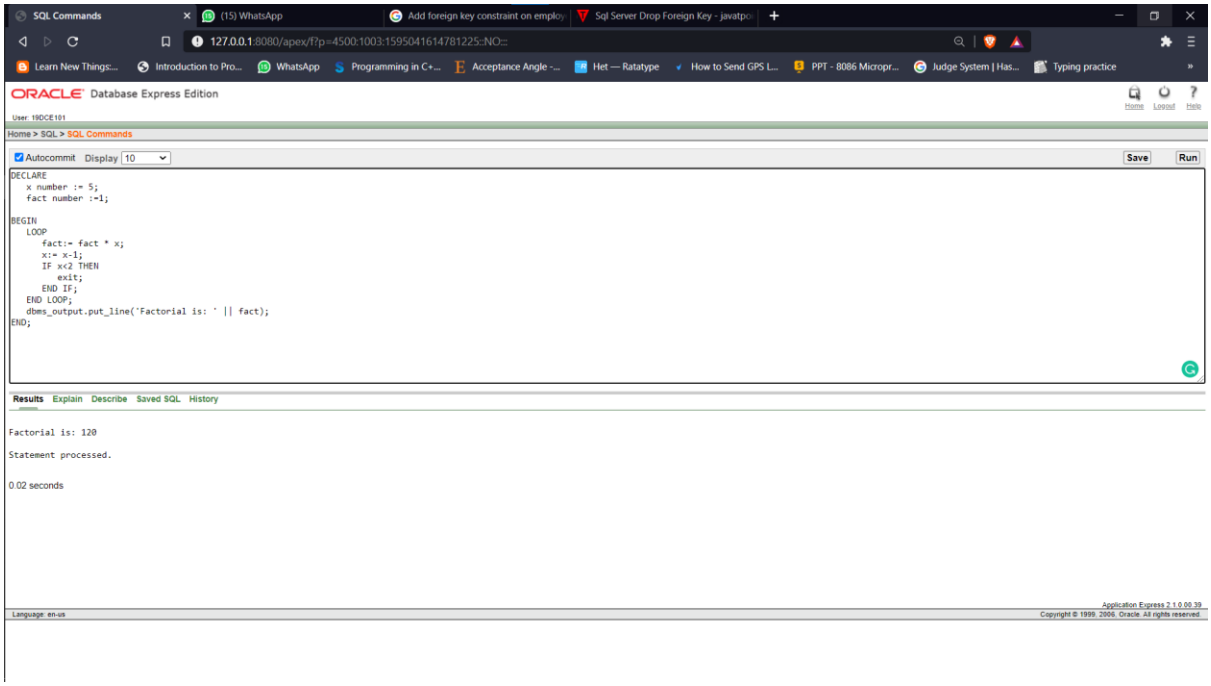
```
DECLARE
  x number := 5;
  fact number :=1;

BEGIN
  LOOP
    fact:= fact * x;
    x:= x-1;
    IF x<2 THEN
      exit;
    END IF;
  END LOOP;
  dbms_output.put_line('Factorial is: ' || fact);
END;
```

While Loop

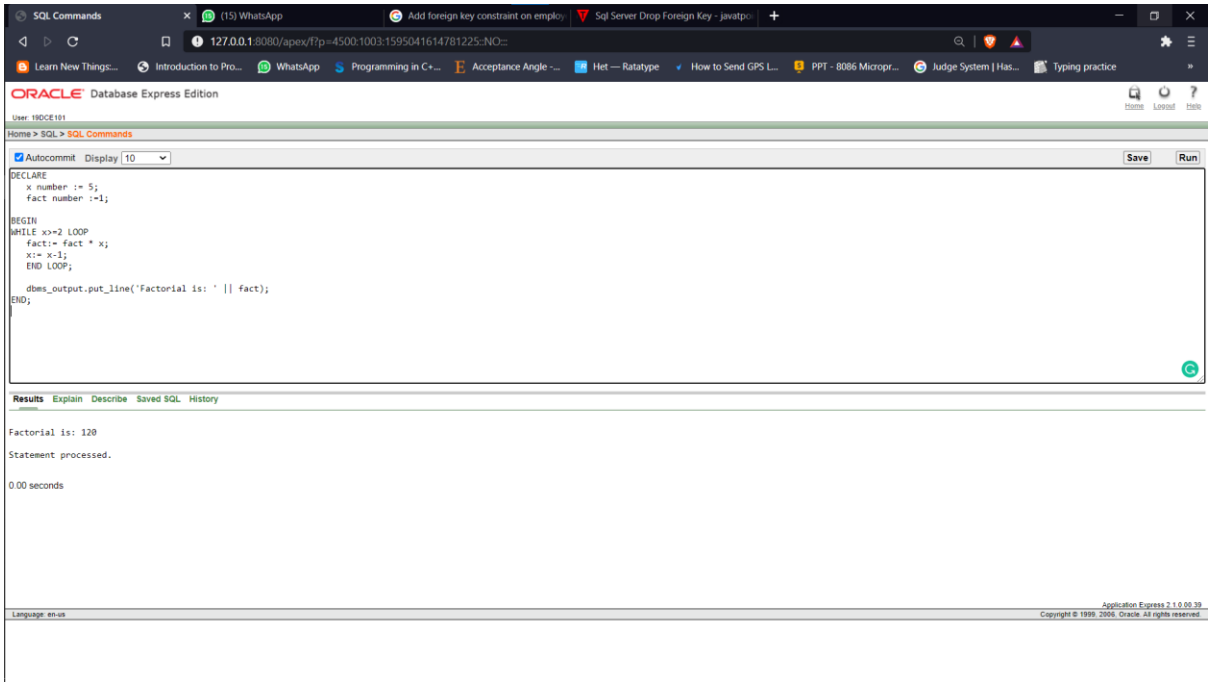
```
DECLARE
  x number := 5;
  fact number :=1;

BEGIN
WHILE x>=2 LOOP
  fact:= fact * x;
  x:= x-1;
  END LOOP;

  dbms_output.put_line('Factorial is: ' || fact);
END;
```

Snap-Shot:

Basic Loop



While Loop



Conclusion: From this practical I learned How to write simple loop & while loop in PL/SQL.

Practical -15

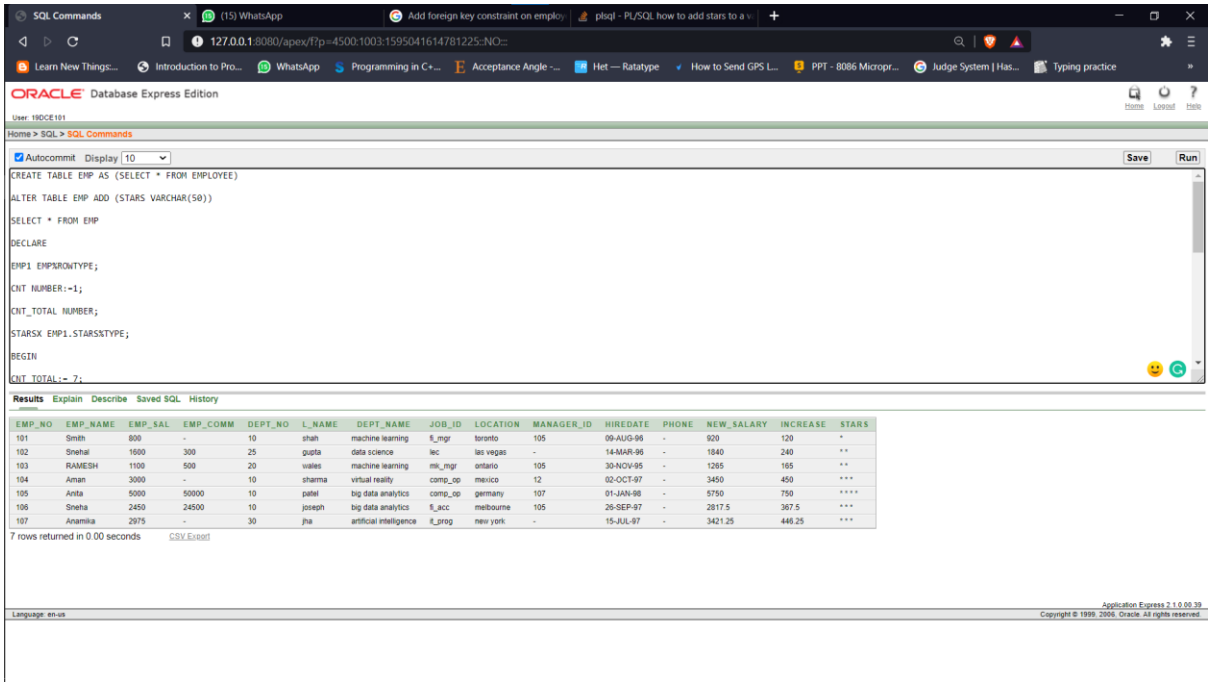
To understand the concept of "select into" and "% type" attribute.
Create an EMPLOYEES table that is a replica of the EMP table. Add a new column, STARS, of VARCHAR2 data type and length of 50 to the EMPLOYEES table for storing asterisk (*).
Create a PL/SQL block that rewards an employee by appending an asterisk in the STARS column for every Rs1000/- of the employee's salary. For example, if the employee has a salary amount of Rs8000/-, the string of asterisks should contain eight asterisks. If the employee has a salary amount of Rs12500/-, the string of asterisks should contain 13 asterisks.
Update the STARS column for the employee with the string of asterisks.

Code:

```
CREATE TABLE EMP AS (SELECT * FROM EMPLOYEE)
ALTER TABLE EMP ADD (STARS VARCHAR(50))
SELECT * FROM EMP

DECLARE

EMP1 EMP%ROWTYPE;

CNT NUMBER:=1;

CNT_TOTAL NUMBER;

STARSX EMP1.STARS%TYPE;

BEGIN

CNT_TOTAL:= 7;

WHILE CNT<=CNT_TOTAL LOOP

    SELECT * INTO EMP1 FROM EMP WHERE EMP_NO = 100+CNT;

     IF EMP1.EMP_SAL > 3000 THEN
      STARSX := '* * * *';

     ELSIF EMP1.EMP_SAL >2000 THEN
      STARSX := '* * *';
```

```
        ELSIF EMP1.EMP_SAL >1000 THEN
         STARSX := '* *';

        ELSE
         STARSX := '*';

       END IF;

    UPDATE EMP SET STARS = STARSX
    WHERE EMP_NO = EMP1.EMP_NO;

    CNT:=CNT+1;

END LOOP;

END;
```

Snap-Shot:



Conclusion: From this practical I learned how to update columns through PL/SQL.

Practical -16

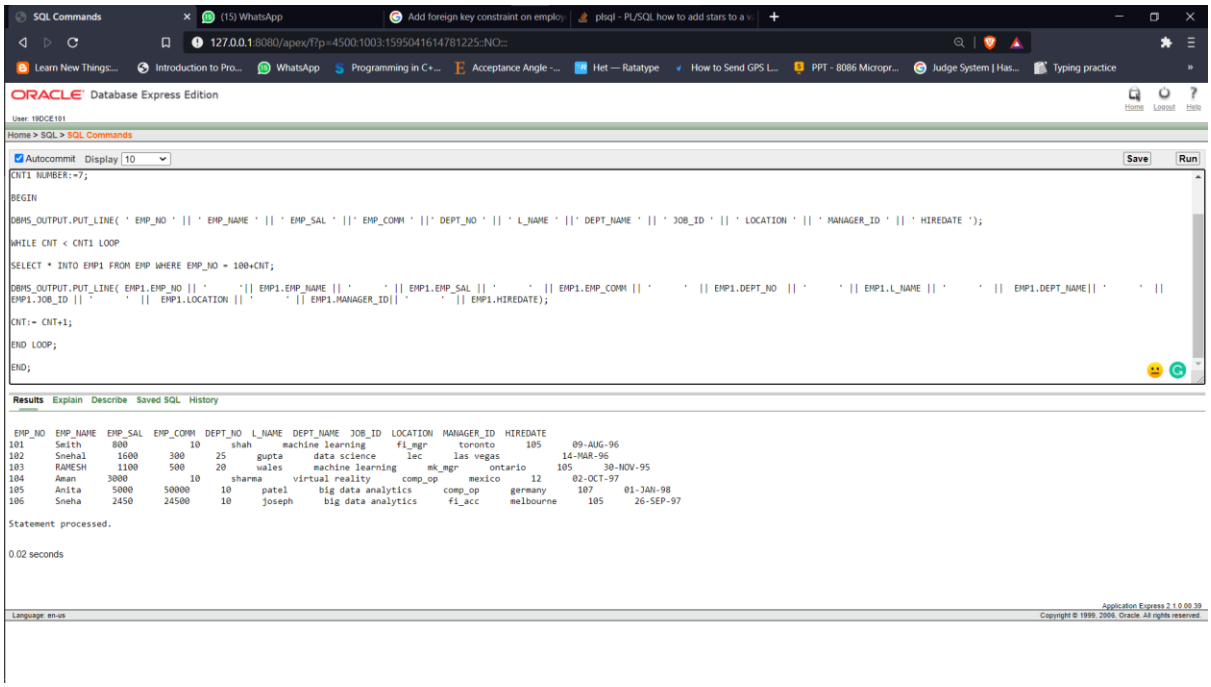
To perform the concept of cursor

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

Code:

```
DECLARE

EMP1 EMP%ROWTYPE;
CNT NUMBER:=1;
CNT1 NUMBER:=7;

BEGIN

DBMS_OUTPUT.PUT_LINE( ' EMP_NO ' || ' EMP_NAME ' || ' EMP_SAL ' ||'
EMP_COMM ' ||' DEPT_NO ' || ' L_NAME ' ||' DEPT_NAME ' || ' JOB_ID ' || ' LOCATION '
|| ' MANAGER_ID ' || ' HIREDATE ');

WHILE CNT < CNT1 LOOP

SELECT * INTO EMP1 FROM EMP WHERE EMP_NO = 100+CNT;

DBMS_OUTPUT.PUT_LINE( EMP1.EMP_NO || '      '|| EMP1.EMP_NAME || '      ' ||
EMP1.EMP_SAL || '      ' || EMP1.EMP_COMM || '      ' || EMP1.DEPT_NO  || '      ' ||
EMP1.L_NAME || '      ' ||  EMP1.DEPT_NAME|| '      ' ||  EMP1.JOB_ID || '      ' ||
EMP1.LOCATION || '      ' || EMP1.MANAGER_ID|| '      ' || EMP1.HIREDATE);

CNT:= CNT+1;

END LOOP;

END;
```

Snap-shot:



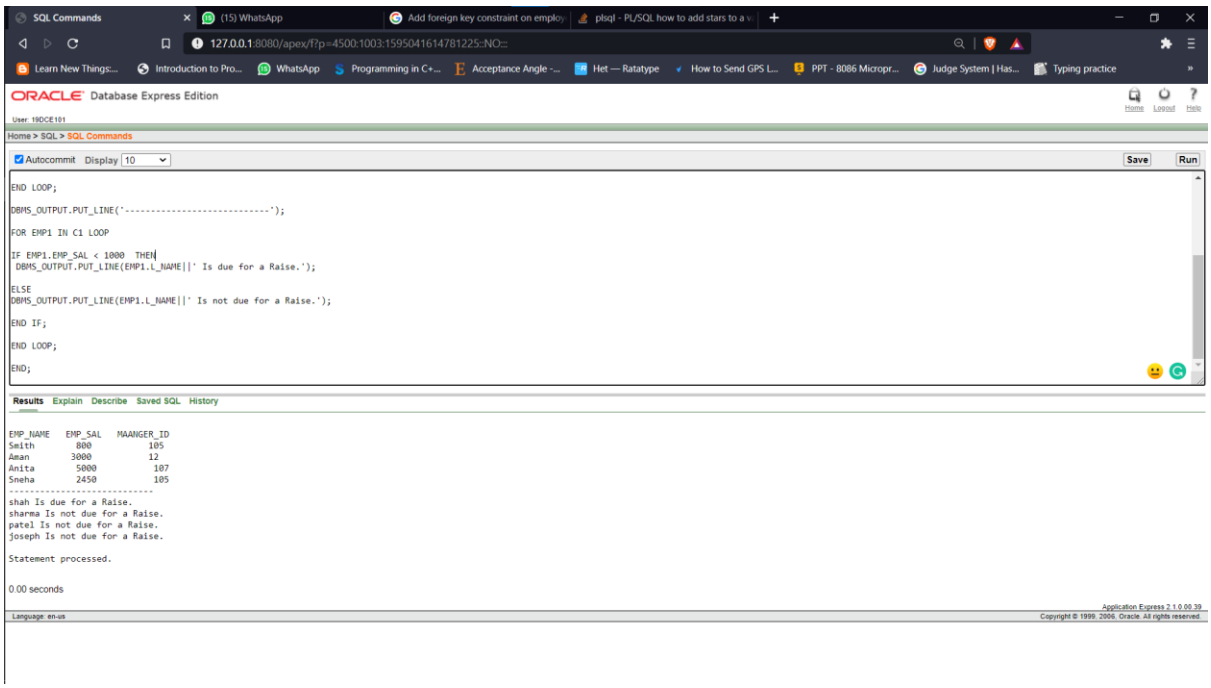
(b) Create a PL/SQL block that does the following:

Code:

```
DECLARE
EMP1 EMP%ROWTYPE;
X EMP.DEPT_NO%TYPE:= :X;

CURSOR C1 IS SELECT EMP_NAME,EMP_SAL,MANAGER_ID,L_NAME FROM EMP
WHERE DEPT_NO = X;

BEGIN

DBMS_OUTPUT.PUT_LINE( 'EMP_NAME   ' || 'EMP_SAL   ' || 'MAANGER_ID   ');

FOR EMP1 IN C1 LOOP

DBMS_OUTPUT.PUT_LINE( EMP1.EMP_NAME ||'        ' || EMP1.EMP_SAL || '          ' ||
EMP1.MANAGER_ID);

END LOOP;

DBMS_OUTPUT.PUT_LINE('----------------------------');
```

```
FOR EMP1 IN C1 LOOP

IF EMP1.EMP_SAL < 1000  THEN
 DBMS_OUTPUT.PUT_LINE(EMP1.L_NAME||' Is due for a Raise.');

ELSE
DBMS_OUTPUT.PUT_LINE(EMP1.L_NAME||' Is not due for a Raise.');

END IF;

END LOOP;

END;
```

Snap-shot:



Conclusion: From this Practical I learned how to perform the concept of cursor.

Practical -17

To perform the concept of trigger

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.

Code:

STEP 1 : CREATE BACKUP TABLE

```
CREATE TABLE SALARY(
        EMP_NO NUMBER,
        EMP_SAL     NUMBER
);
```

STEP 2: ADD TRIGGER

```
CREATE OR REPLACE TRIGGER COPY_SALARY
BEFORE UPDATE
ON EMPLOYEE4
FOR EACH ROW

DECLARE

EMP_SAL NUMBER;
EMP_NO NUMBER;

BEGIN
EMP_SAL := :OLD.EMP_SAL;
EMP_NO := :OLD.EMP_NO;
INSERT INTO SALARY
VALUES (EMP_NO,EMP_SAL);
END;
```

STEP 3: UPDATE VALUE IN EMP WHERE DEPT_NO = 10

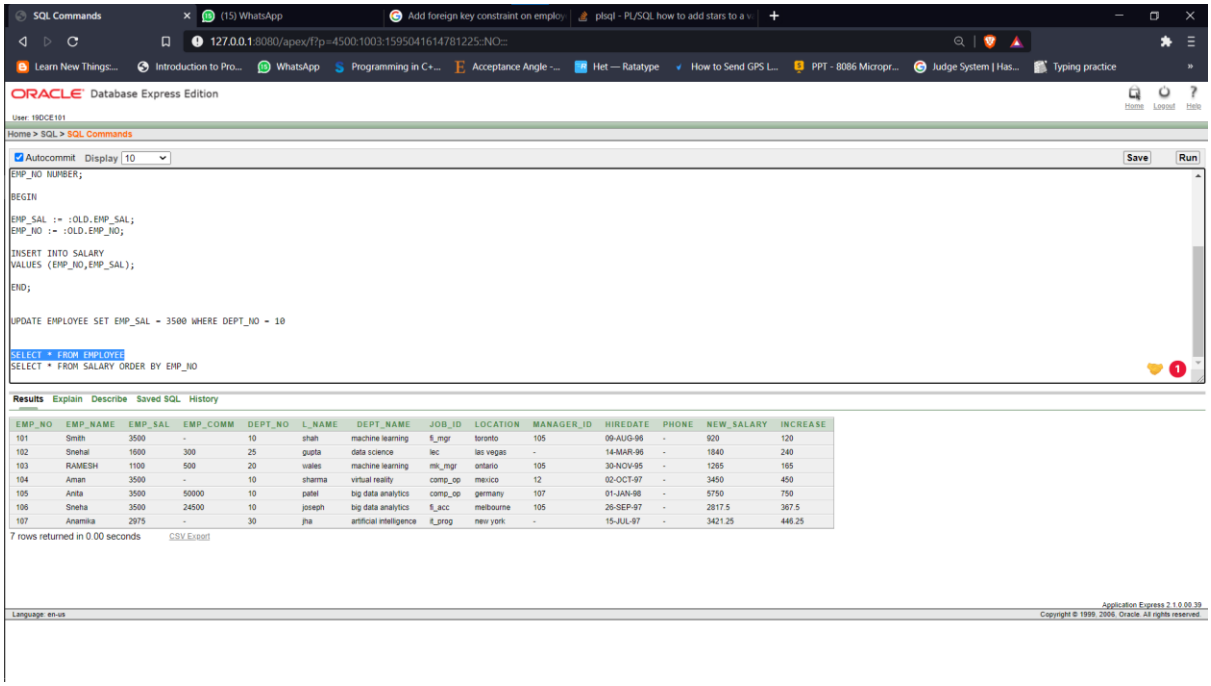
```
UPDATE EMPLOYEE SET EMP_SAL = 3500 WHERE DEPT_NO = 10
```

STEP 4: AFTER UPDATE CHECK VALUES IN BOTH TABLES

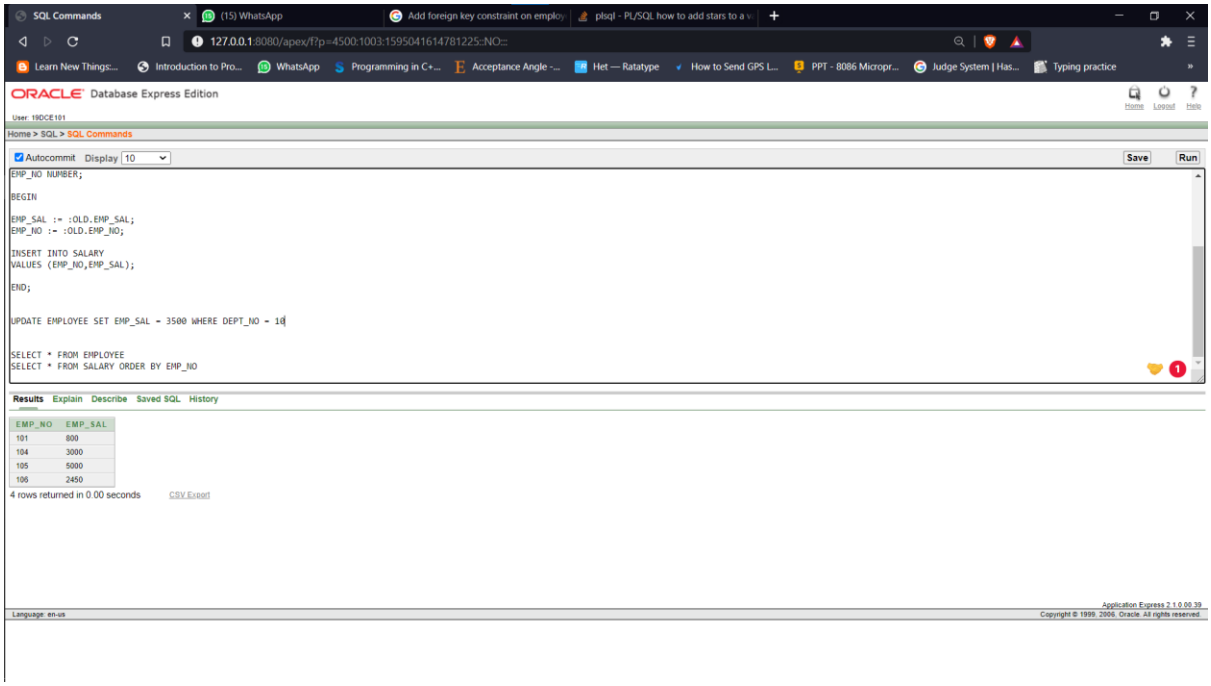
```
SELECT * FROM EMPLOYEE

SELECT * FROM SALARY ORDER BY EMP_NO
```

Snap-shots:

Employee Table After Command Execution:



Salary Table After Command Execution having old values of Emp_Sal:



Conclusion: From this Practical I learned how to use trigger to perform certain tasks.

Practical -18

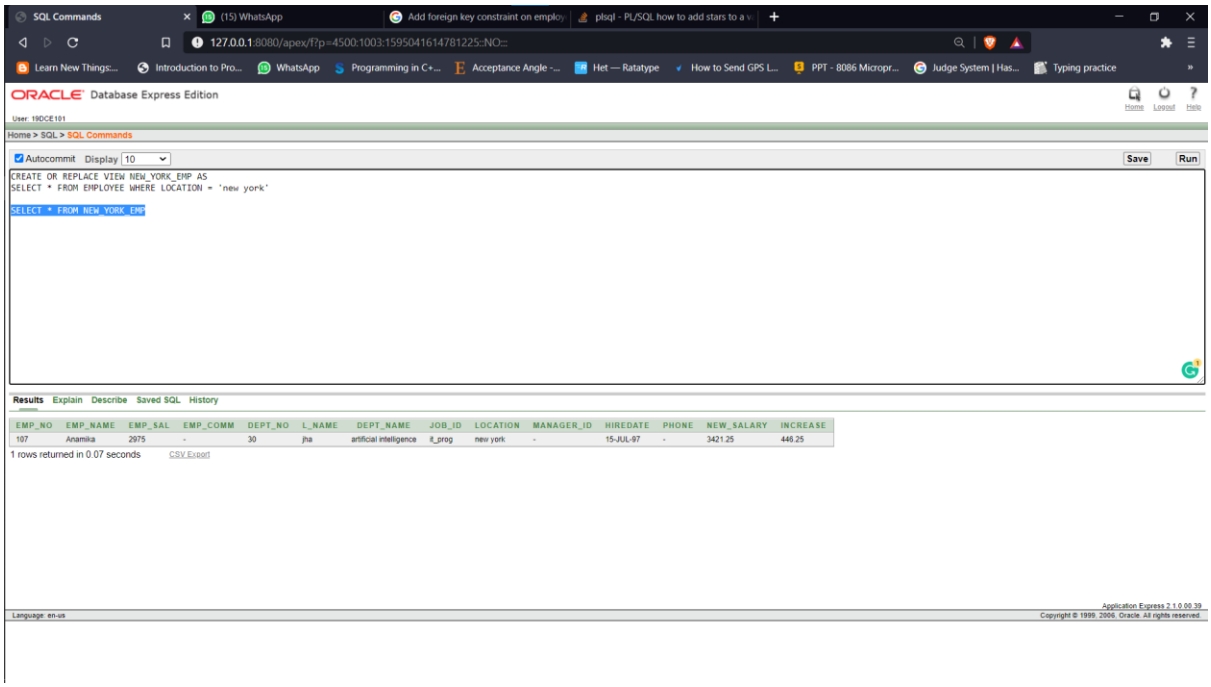
To solve queries using the concept of View.

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

Code:

```
--create view
CREATE OR REPLACE VIEW NEW_YORK_EMP AS
SELECT * FROM EMPLOYEE WHERE LOCATION = 'new york'
--display view
SELECT * FROM NEW_YORK_EMP;
```

Snap-shot:

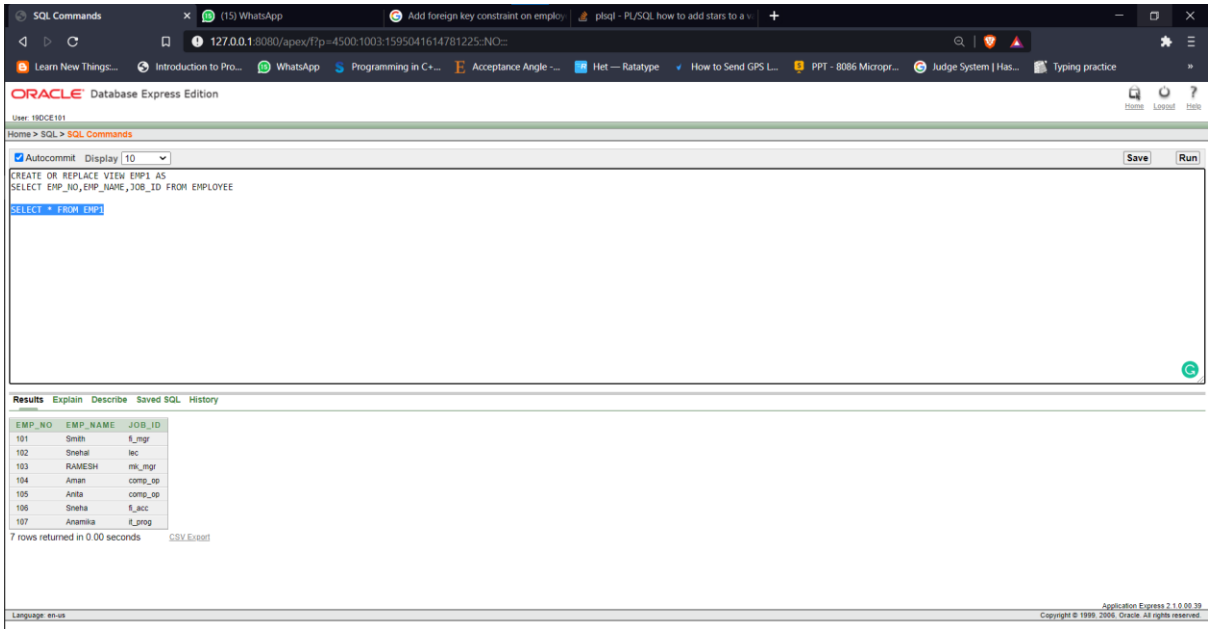


(2) Write a query to create a view for all employee with columns emp_id, emp_name, and job_id.

Code:

```
--create view
CREATE OR REPLACE VIEW EMP1 AS
SELECT EMP_NO,EMP_NAME,JOB_ID FROM EMPLOYEE
--display view
SELECT * FROM EMP1;
```

Snap-shot:

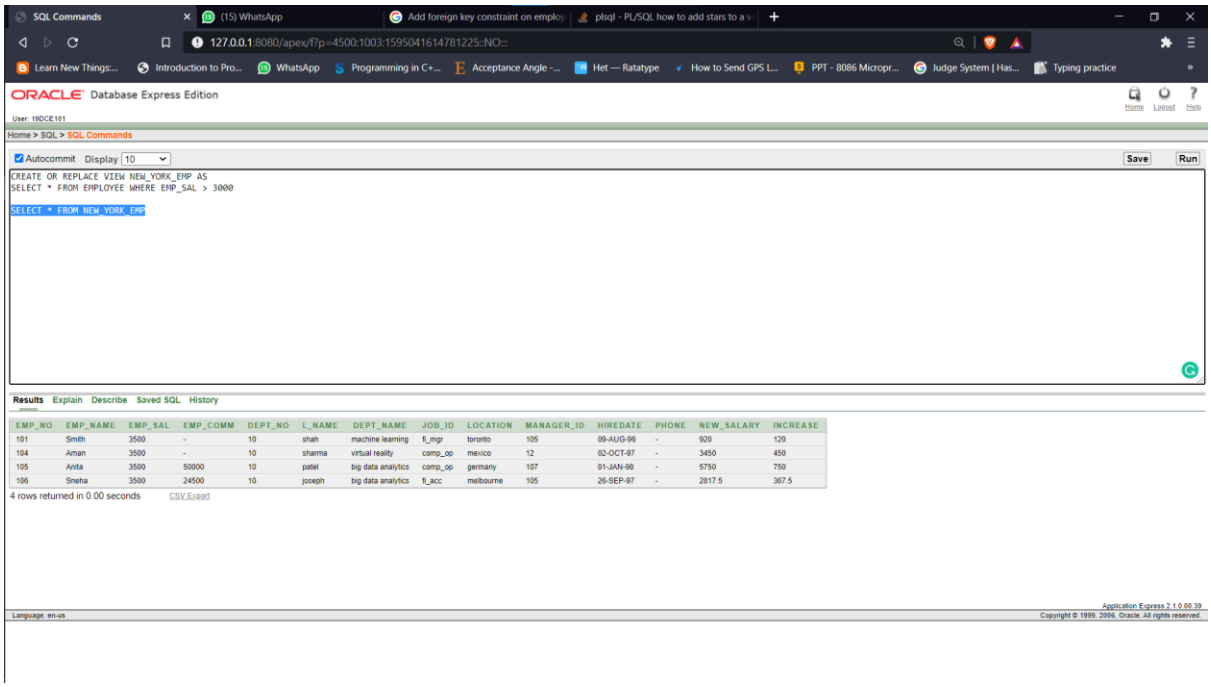


(3) Write a query to find the salesmen of the location New York who having salary more than 3000.

Code:

```
--create view
CREATE OR REPLACE VIEW NEW_YORK_EMP AS SELECT * FROM EMPLOYEE
WHERE EMP_SAL > 3000
--display view
SELECT * FROM NEW_YORK_EMP
```

Snap-shot

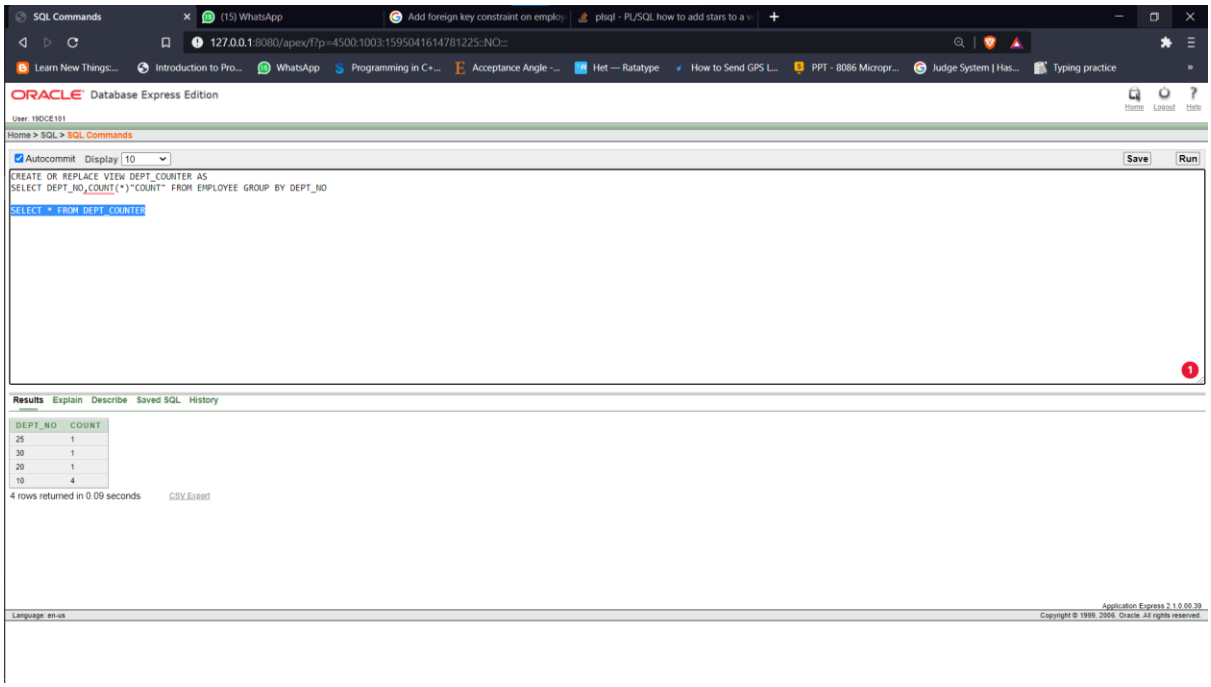


(4) Write a query to create a view to getting a count of how many employee we have at each department.

Code:

```
--create view
CREATE OR REPLACE VIEW DEPT_COUNTER AS
SELECT DEPT_NO,COUNT(*)"COUNT" FROM EMPLOYEE GROUP BY DEPT_NO;
--display view
SELECT * FROM DEPT_COUNTER;
```

Snap-shot:



Conclusion: From This Practical I Learned how to create a view in oracle PL/SQL.

Practical -19

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.

Code:

Using Procedure:

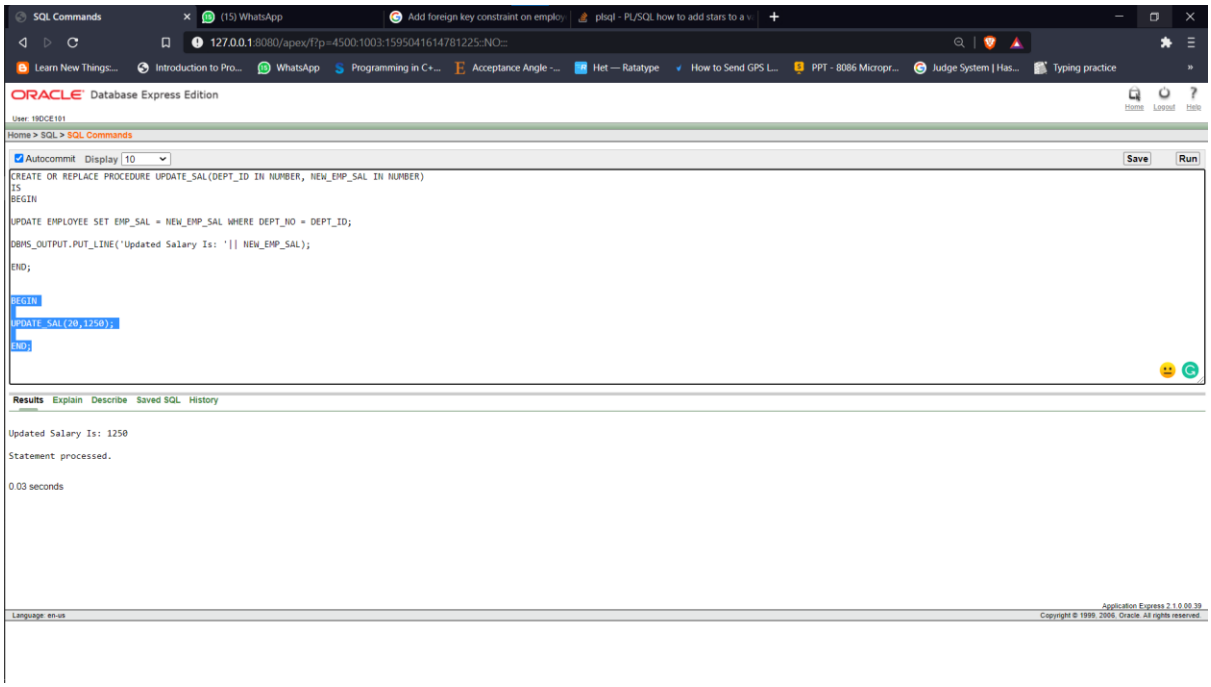
STEP 1: CREATE A PROCEDURE

```
CREATE OR REPLACE PROCEDURE UPDATE_SAL(DEPT_ID IN
NUMBER,NEW_EMP_SAL IN NUMBER)

IS

BEGIN

UPDATE EMPLOYEE SET EMP_SAL = NEW_EMP_SAL WHERE DEPT_NO =
DEPT_ID;

DBMS_OUTPUT.PUT_LINE('Updated Salary Is: '|| NEW_EMP_SAL);

END;
```

STEP 2: EXECUTE THE COMMAND

```
BEGIN
UPDATE_SAL(20,1250);
END;
```

Snap-shot:



Using Function:

STEP 1: CREATE A FUNCTION

CREATE OR REPLACE FUNCTION UPDATE_SALARY(DEPT_ID NUMBER,
EMP_NEW_SAL NUMBER)
RETURN NUMBER IS

BEGIN

UPDATE EMPLOYEE SET EMP_SAL = EMP_NEW_SAL WHERE DEPT_NO =
DEPT_ID;

RETURN EMP_NEW_SAL;

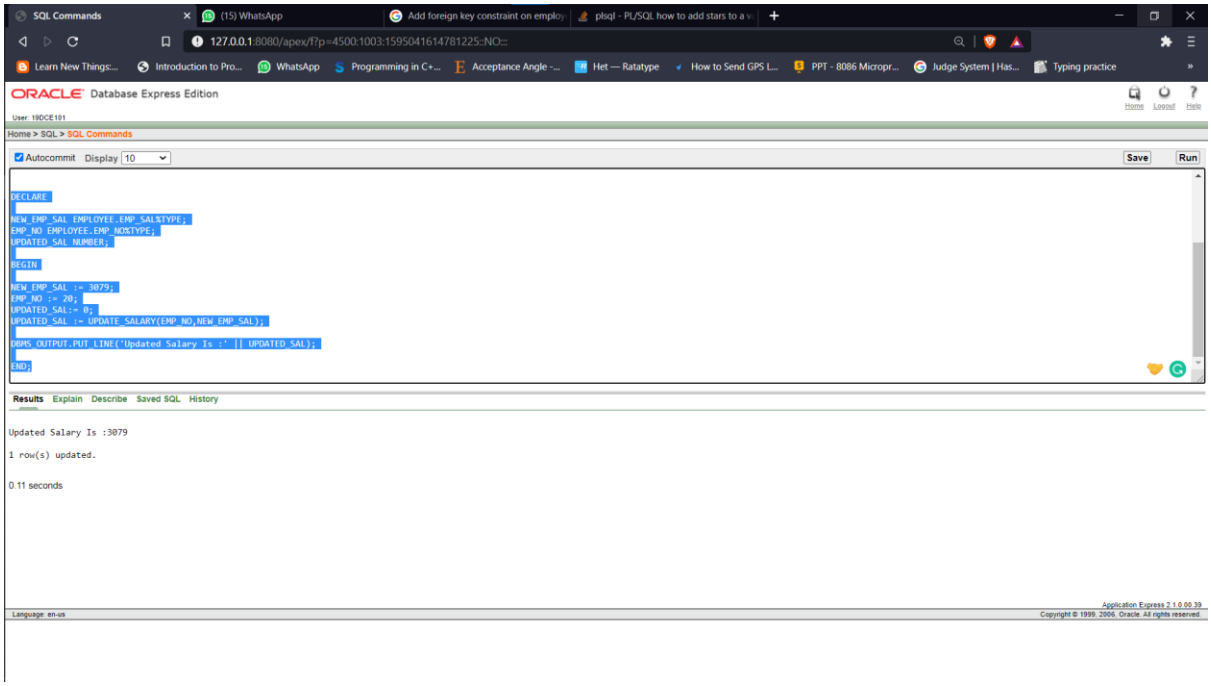
END;

STEP 2: USE THAT FUCNTION IN OTHER PL/SQL BLOCKS

DECLARE

```
NEW_EMP_SAL EMPLOYEE.EMP_SAL%TYPE;
EMP_NO EMPLOYEE.EMP_NO%TYPE;
UPDATED_SAL NUMBER;

BEGIN

NEW_EMP_SAL := 3079;
EMP_NO := 20;
UPDATED_SAL:= 0;
UPDATED_SAL := UPDATE_SALARY(EMP_NO,NEW_EMP_SAL);

DBMS_OUTPUT.PUT_LINE('Updated Salary Is :' || UPDATED_SAL);

END;
```

Snap-shot:



Conclusion: From this Practical I learned how to use Procedure & Functions which are pl/sql objects to perform tasks, making code reusable.

Practical -20

To perform the concept of exception handler

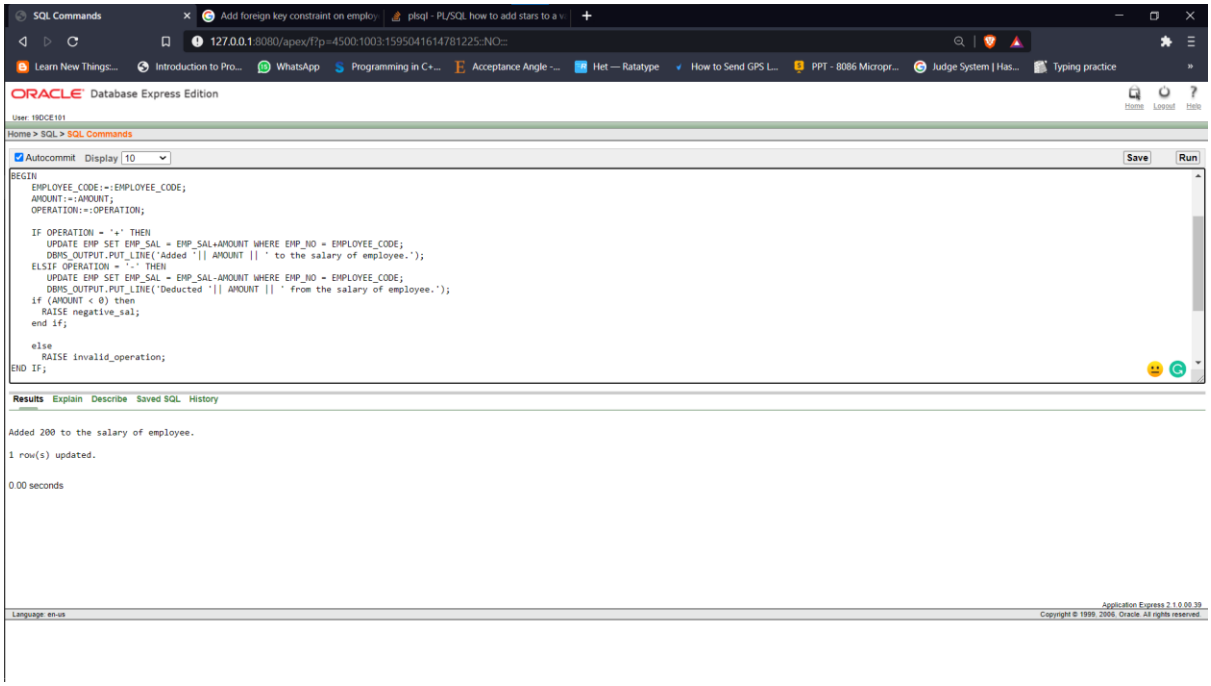
Write a PL/SQL block that will accept the employee code, amount and operation. Based on specified operation amount is added or deducted from salary of said employee. Use user defined exception handler for handling the exception.

Code:

```
DECLARE
   EMPLOYEE_CODE NUMBER;
   AMOUNT NUMBER;
   OPERATION VARCHAR(10);
   invalid_operation EXCEPTION;
   negative_sal EXCEPTION;

BEGIN
   EMPLOYEE_CODE:=:EMPLOYEE_CODE;
   AMOUNT:=:AMOUNT;
   OPERATION:=:OPERATION;

   IF OPERATION = '+' THEN
      UPDATE EMP SET EMP_SAL = EMP_SAL+AMOUNT WHERE EMP_NO =
EMPLOYEE_CODE;
      DBMS_OUTPUT.PUT_LINE('Added '|| AMOUNT || ' to the salary of employee.');
   ELSIF OPERATION = '-' THEN
      UPDATE EMP SET EMP_SAL = EMP_SAL-AMOUNT WHERE EMP_NO =
EMPLOYEE_CODE;
      DBMS_OUTPUT.PUT_LINE('Deducted '|| AMOUNT || ' from the salary of employee.');
   if (AMOUNT < 0) then
     RAISE negative_sal;
   end if;

   else
     RAISE invalid_operation;
END IF;

EXCEPTION
  when invalid_operation then
   dbms_output.put_line('Invalid Operation !!!');
  when NO_DATA_FOUND then
   dbms_output.put_line('No data Found !!!');
```

```
  when negative_sal then
    dbms_output.put_line('Salary cannot be negative !!!');
END;
/
```

Snap-shot:



Conclusion: From This Practical I Learned how to handle exceptions in oracle.

Practical -21

To perform the concept of package

Code:

STEP 1: CREATE A PACKAGE DECLARATION.

```
CREATE OR REPLACE PACKAGE getEmployeeData AS

-- public
FUNCTION getRow(Emp_id number) return EMPLOYEE4%ROWTYPE;

END getEmployeeData;
```

STEP 2: CREATE A PACKAGE BODY.

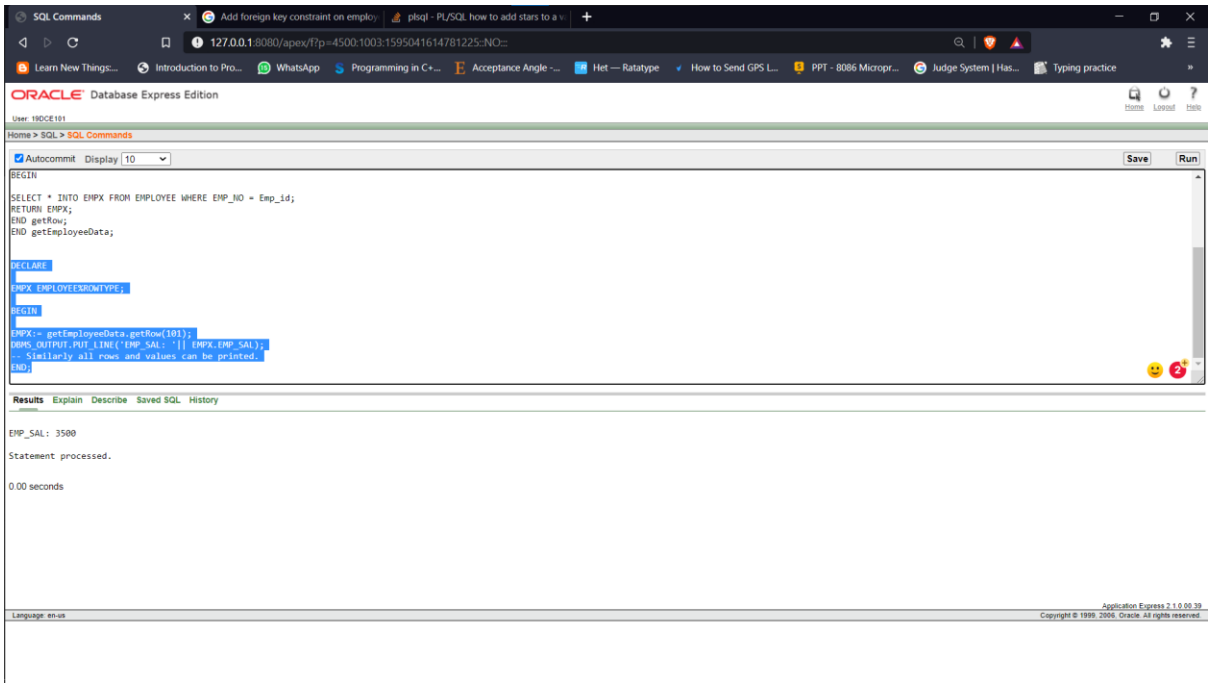
```
CREATE OR REPLACE PACKAGE BODY getEmployeeData AS

FUNCTION getRow(Emp_id number) return EMPLOYEE%ROWTYPE AS
-- THIS VARIABLE WILL BE PRIVATE.
EMPX EMPLOYEE%ROWTYPE;

BEGIN

SELECT * INTO EMPX FROM EMPLOYEE4 WHERE EMP_NO = Emp_id;

RETURN EMPX;

END getRow;

END getEmployeeData;
```

STEP 3: Use the Package In a PL/SQL Block.

```
DECLARE

EMPX EMPLOYEE%ROWTYPE;

BEGIN
```

```
EMPX:= getEmployeeData.getRow(101);

DBMS_OUTPUT.PUT_LINE('EMP_SAL: '|| EMPX.EMP_SAL);

-- Similarly all rows and values can be printed.

END;
```

Snap-shot:



Conclusion: From This Practical I Learned how to create & use a package in oracle.