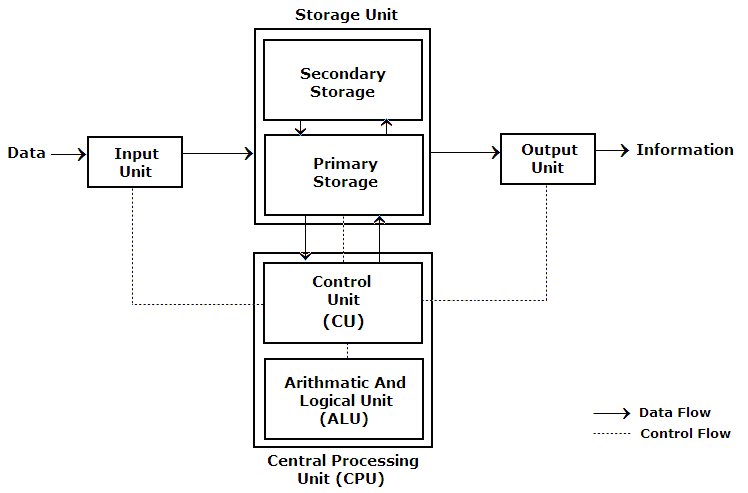
**PRACTICAL – 1**

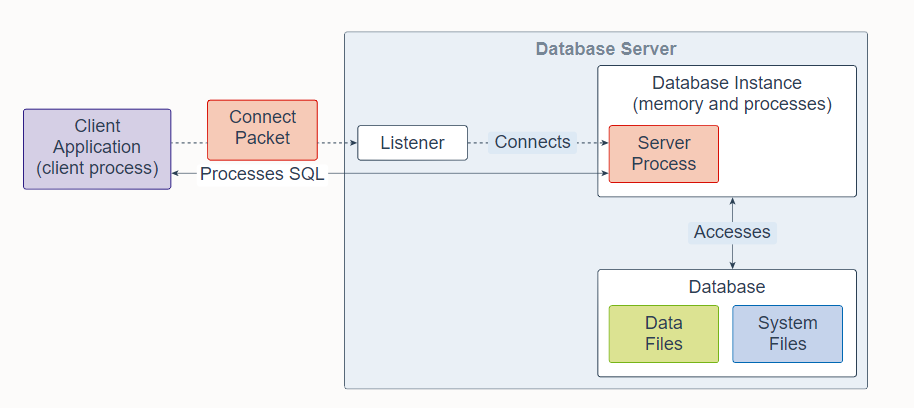
**INTRODUCTION TO ORACLE ARCHITECTURE**

**Computer Architecture:** Computer architecture is a set of rules and methods that describe the functionality, organization, and implementation of computer system Some definitions of architecture define it as describing the capabilities and programming model of a computer but not a particular implementation.



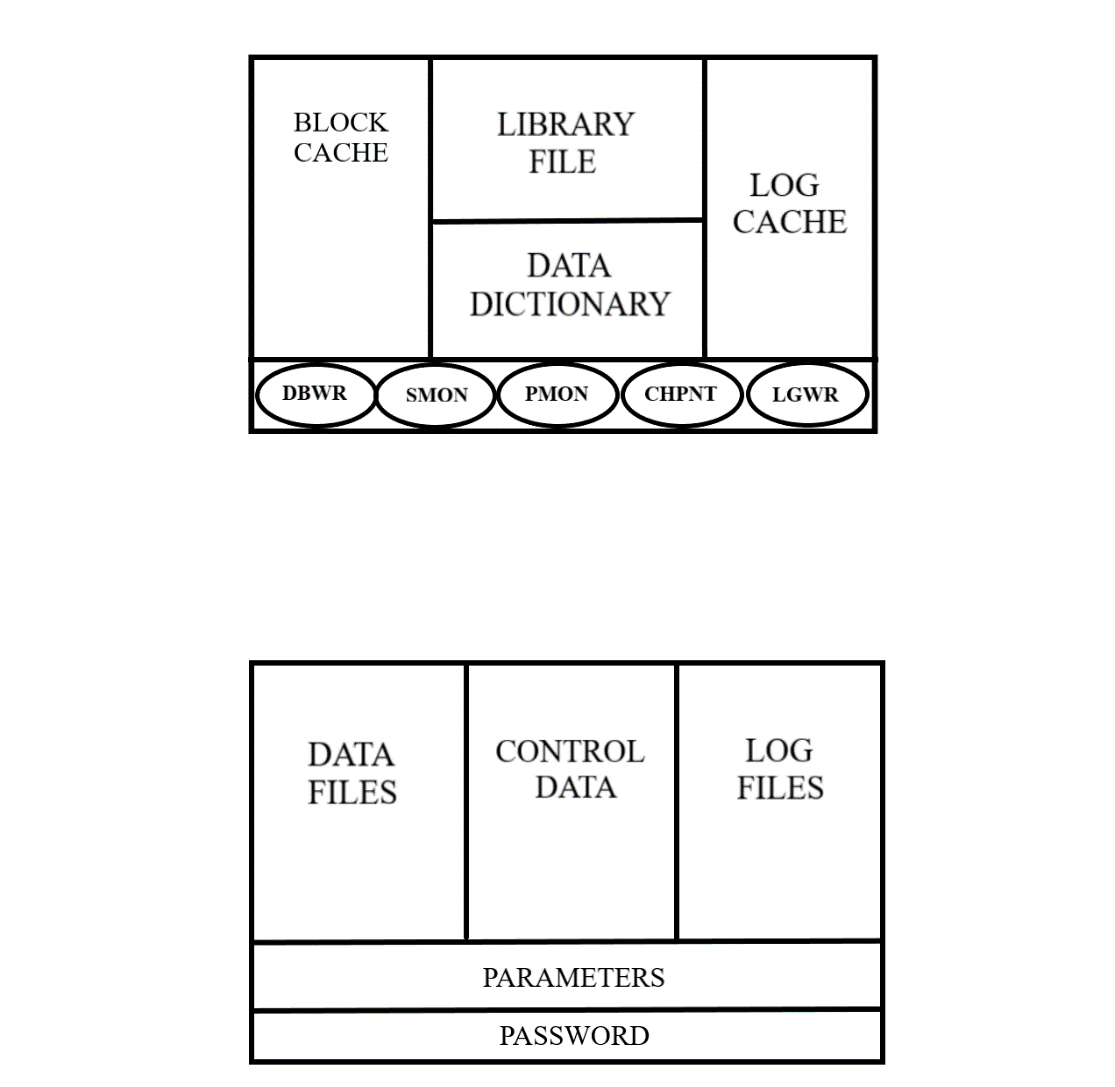
* **ORACLE ARCHITECTURE:**

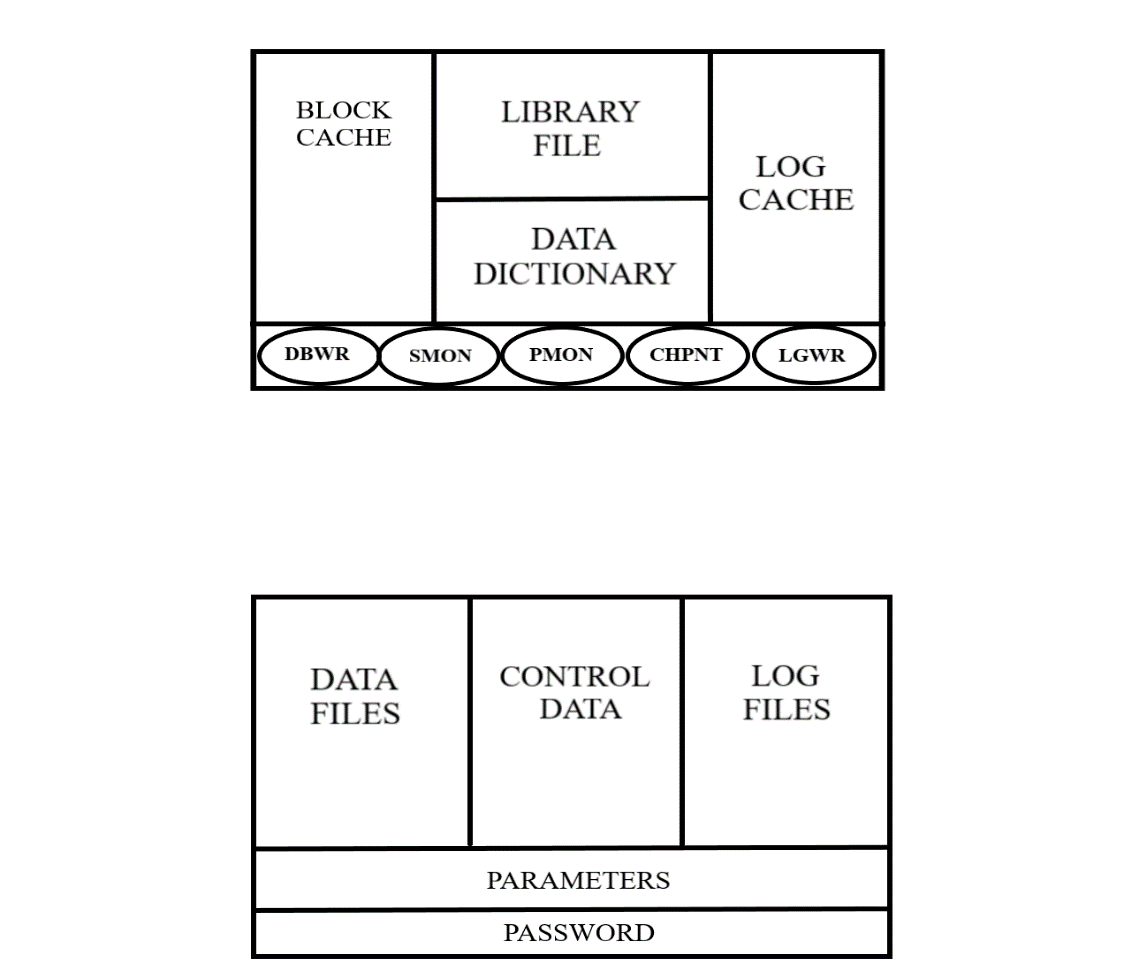
An Oracle Real Application Clusters (Oracle RAC) database architecture consist of multiple instances that run on separate server machines. All of them share the same database. The cluster of server machines appears as a single server on one end, and end users and applications on the other end.

An Oracle Database consists of at least one database instance and one database. The database instance handles memory and processes. The database consists of physical files called data files, and can be a non-container database or a multitenant container database. An Oracle Database also uses several database system files during its operation.

A single-instance database architecture consists of one database instance and one database. A one-to-one relationship exists between the database and the database instance. Multiple single-instance databases can be installed on the same server machine. There are separate database instances for each database. This configuration is useful to run different versions of Oracle Database on the same machine.

An Oracle Real Application Clusters (Oracle RAC) database architecture consists of multiple instances that run on separate server machines. All of them share the same database. The cluster of server machines appears as a single server on one end, and end users and applications on the other end. This configuration is designed for high availability, scalability, and high-end performance.





**COMPONENTS OF STORAGE IN ORACLE ARCHITECTURE:**

**Data File** : All tables and relationships between the tables is stored here.

**Control Data** : All the System files are stored.

**Log files** : All logs are stored here.

**Parameters :** Values of various parameters like block size are stored here.

**Password** : Password of all users are stored here.

**COMPONENTS OF MEMORY IN ORACLE ARCHITECTURE:**

**Block cache** : Data retrieved from files is stored here.

**Library files** : All commands and Syntax are stored here for verification.

**Data dictionary** : Meta data is stored here.

**Log cache** : Log of all operation is stored here before data is commited.

**Checkpoint**: It syncronises LGWR and DBWR.

**DBWR** (Database Writer) : It fetches data from data files & writes it to block cache.

**SMON** (System monitor) : It monitors the system.

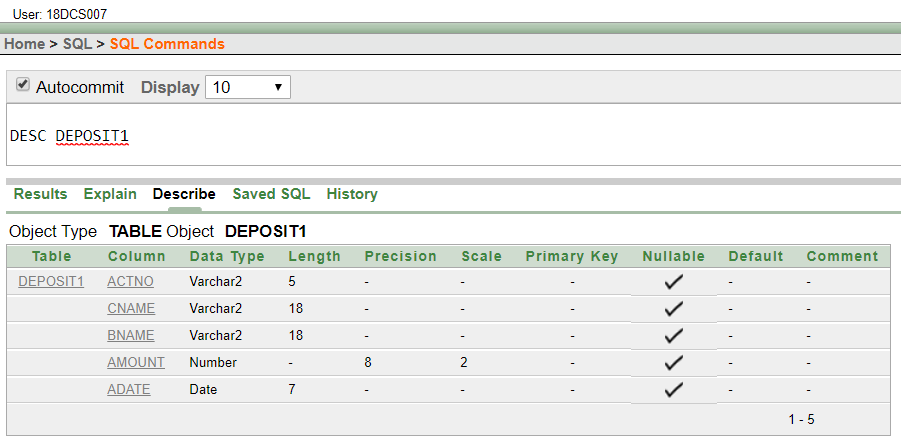
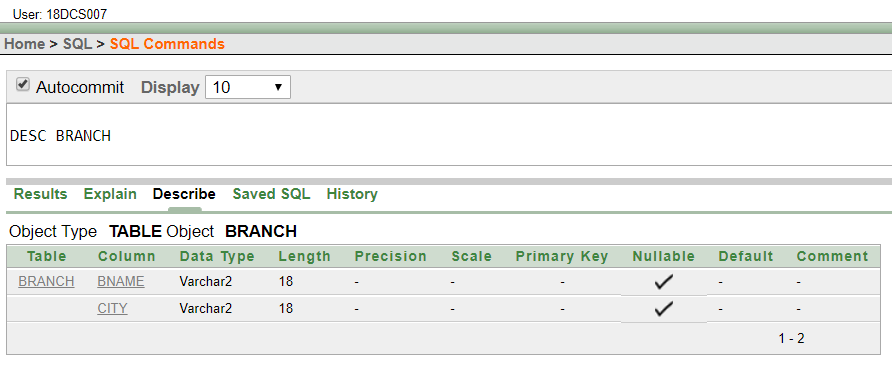
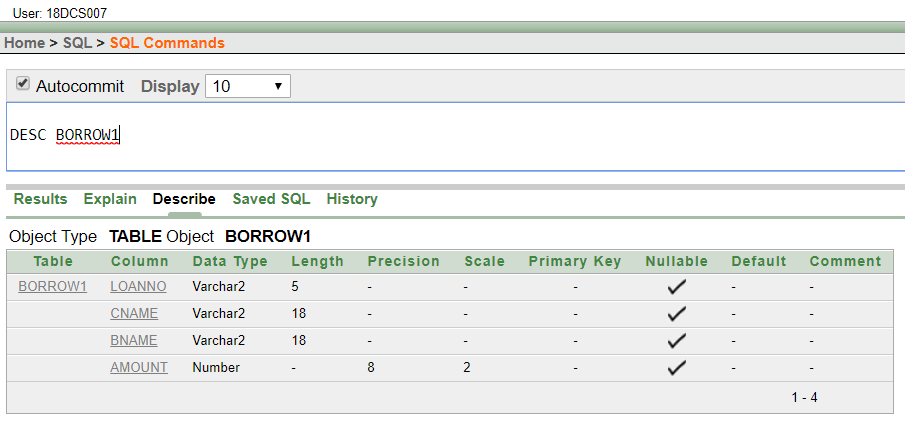
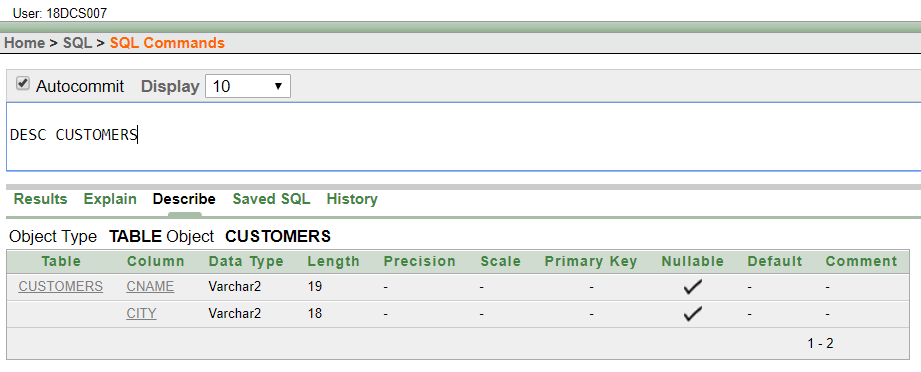
**PMON** (Process Monitor) : It monitors ongoing processes.

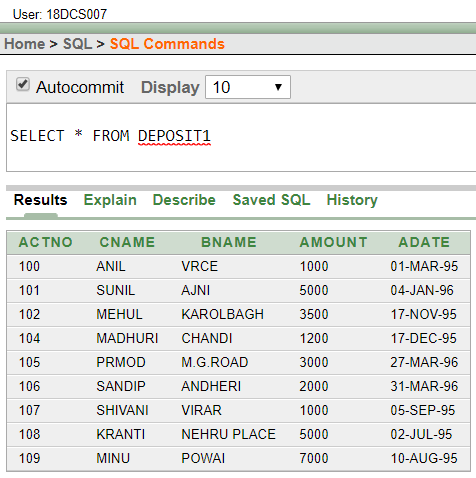
**LGWR** (Log Writer) : It fetches log from log cache & writes it to log files after data is committed.

**CONCLUSION**

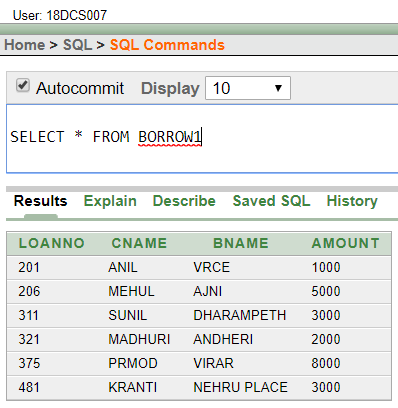
In this Practical I learnt about Oracle Architecture

**PRACTICAL – 2**

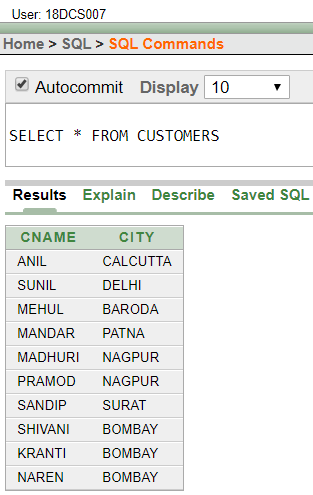
1. **DESCRIBE DEPOSIT1, BRANCH.**
2. **DESCRIBE BORROW1, CUSTOMERS.**
3. **LIST ALL DATA FROM TABLE DEPOSIT1.**



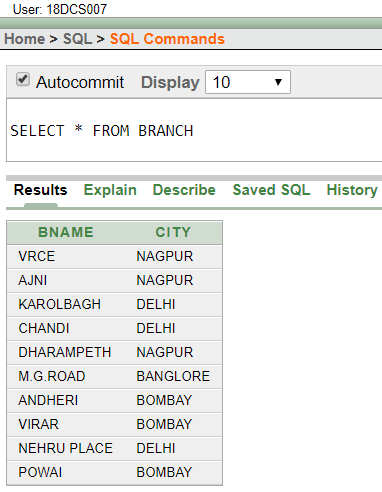
1. **LIST ALL DATA FROM TABLE BORROW1.**



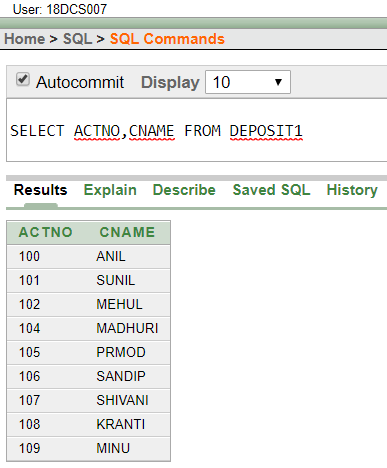
1. **LIST ALL DATA FROM TABLE CUSTOMERS.**



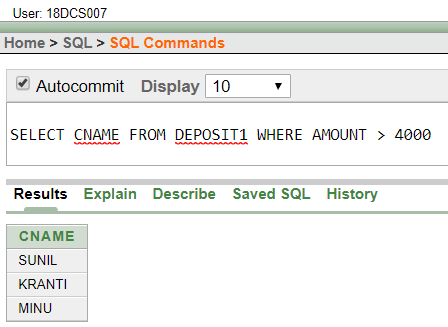
1. **LIST ALL DATA FROM TABLE BRANCH.**



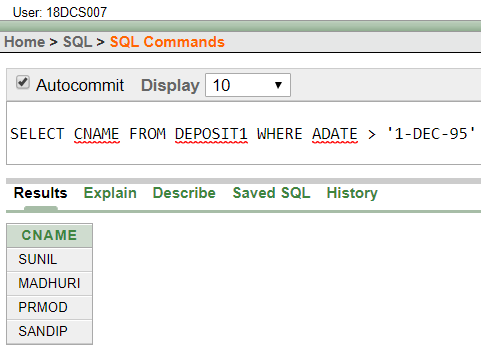
1. **GIVE ACCOUNT NO AND AMOUNT OF DEPOSITORS.**



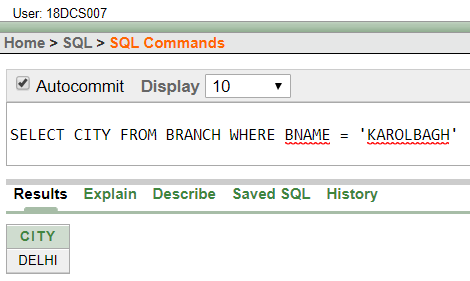
1. **GIVE NAME OF DEPOSITORS HAVING AMOUNT GREATER THAN 4000.**



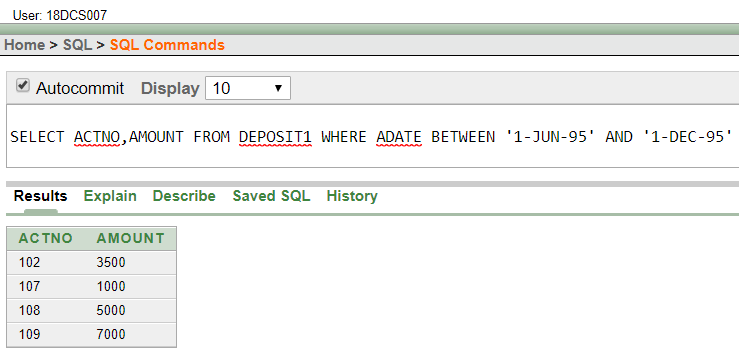
1. **GIVE NAME OF CUSTOMERS WHO OPENED ACCOUNT AFTER DATE '1-12-95'.**



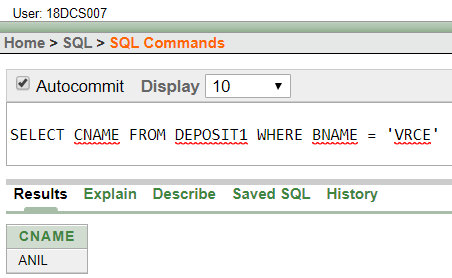
1. **GIVE NAME OF CITY WHERE BRANCH KAROLBAGH IS LOCATED.**



1. **GIVE ACCOUNT NO AND AMOUNT OF CUSTOMER HAVING ACCOUNT OPENED BETWEEN DATE 1-6-95 AND 1-12-95**



1. **GIVE NAMES OF DEPOSITORS HAVING ACCOUNT AT VRCE.**

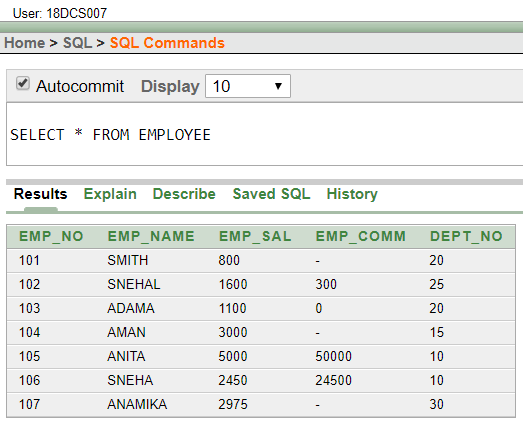


**CONCLUSION**

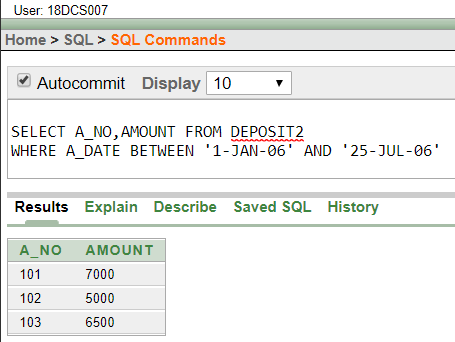
In this Practical I learnt DDL-create and DML-insert command and some Basic Queries

**PRACTICAL – 3**

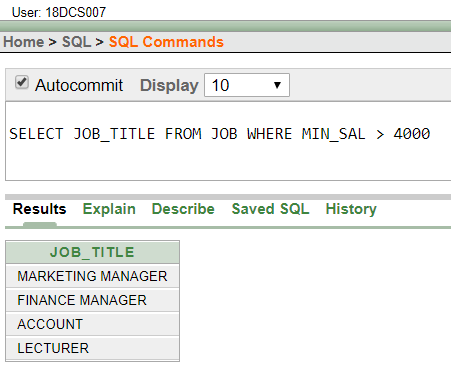
1. **RETRIEVE ALL DATA FROM EMPLOYEE, JOBS AND DEPOSIT.**



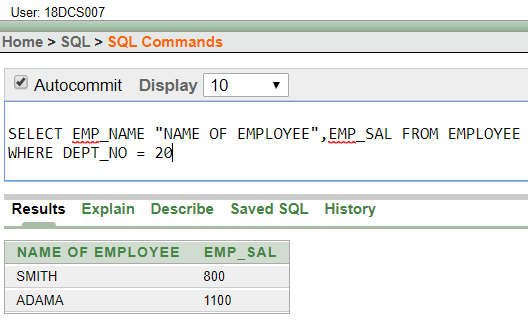
1. **GIVE DETAILS OF ACCOUNT NO. AND DEPOSITED RUPEES OF CUSTOMERS HAVING ACCOUNT OPENED BETWEEN DATES 01-01-06 AND 25-07-06.**



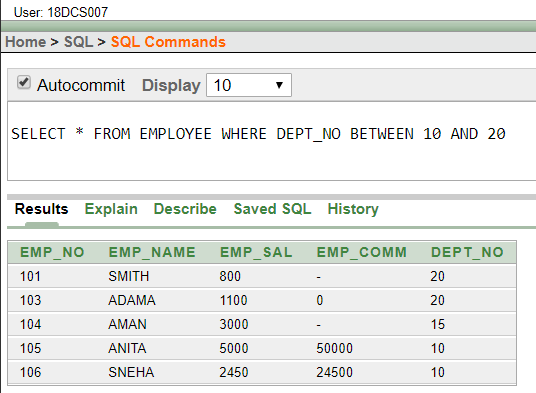
1. **DISPLAY ALL JOBS WITH MINIMUM SALARY IS GREATER THAN 4000.**



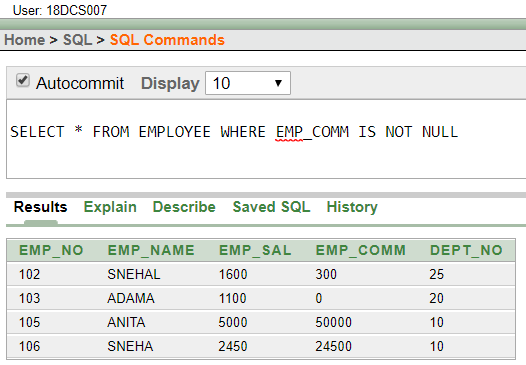
1. **DISPLAY NAME AND SALARY OF EMPLOYEE WHOSE DEPARTMENT NO IS 20. GIVE ALIAS NAME TO NAME OF EMPLOYEE.**



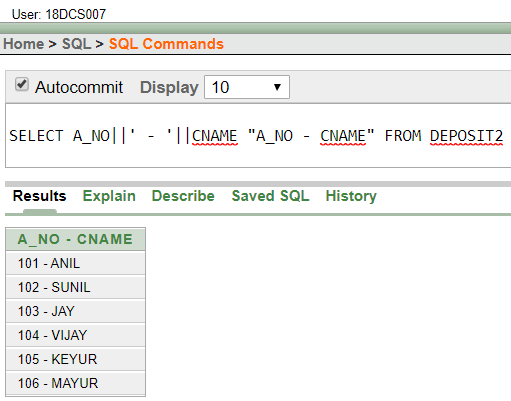
1. **DISPLAY EMPLOYEE NO, NAME AND DEPARTMENT DETAILS OF THOSE EMPLOYEE WHOSE DEPARTMENT LIES IN (10,20).**



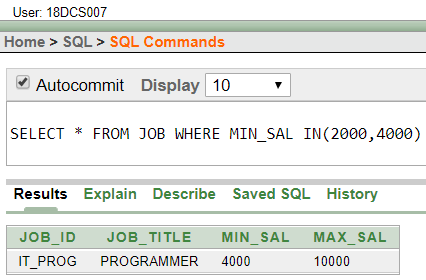
1. **DISPLAY THE NON-NULL VALUES OF EMPLOYEES.**



1. **DISPLAY NAME OF CUSTOMER ALONG WITH ITS ACCOUNT NO (BOTH COLUMNS SHOULD BE DISPLAYED AS ONE) WHOSE AMOUNT IS NOT EQUAL TO 8000 RS.**

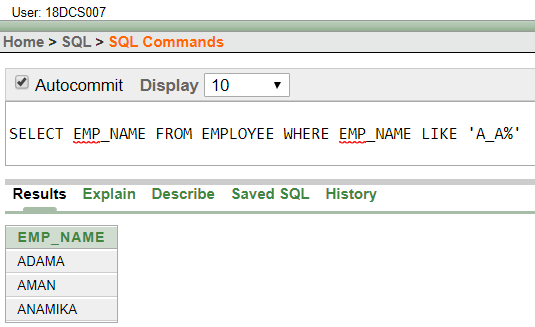


1. **DISPLAY THE CONTENT OF JOB DETAILS WITH MINIMUM SALARY EITHER 2000 OR 4000.**

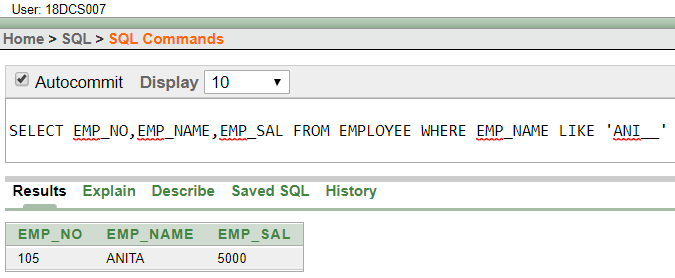


**To study various options of LIKE predicate**

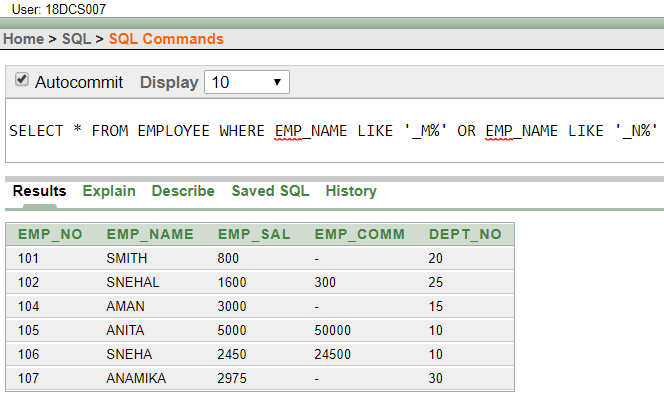
(1) DISPLAY ALL EMPLOYEE WHOSE NAME START WITH ‘A’ AND THIRD CHARACTER IS ‘‘A’.



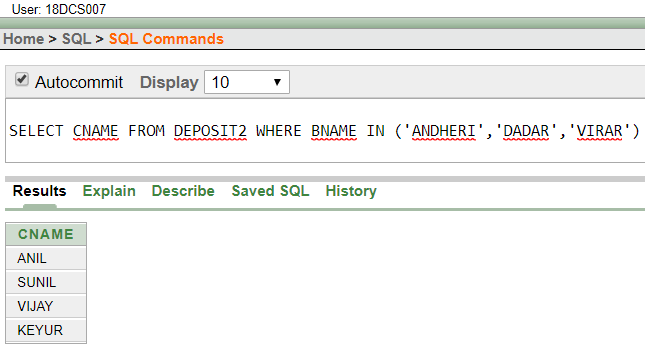
(2) DISPLAY NAME, NUMBER AND SALARY OF THOSE EMPLOYEES WHOSE NAME IS 5 CHARACTERS LONG AND FIRST THREE CHARACTERS ARE ‘ANI’.



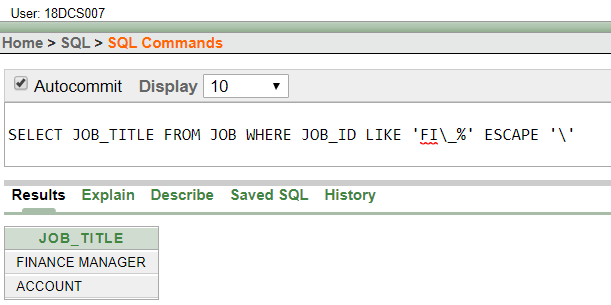
(3) DISPLAY ALL INFORMATION OF EMPLOYEE WHOSE SECOND CHARACTER OF NAME IS EITHER ‘M’ OR ‘N’.



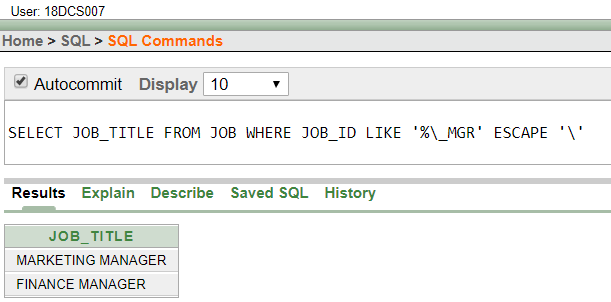
(4) FIND THE LIST OF ALL CUSTOMER NAME WHOSE BRANCH IS IN ‘ANDHERI’ OR ‘DADAR’ OR ‘VIRAR’.



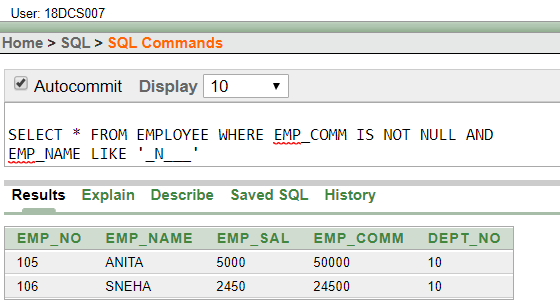
(5) DISPLAY THE JOB NAME WHOSE FIRST 3 CHAR. IN JOB ID FIELD IS ‘FI\_’.



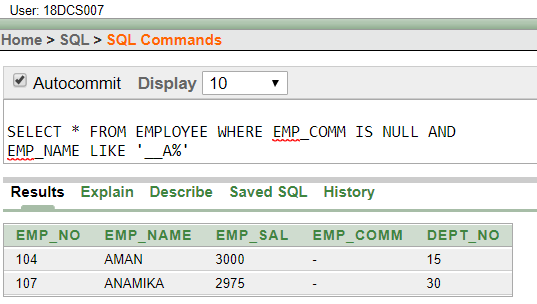
(6) DISPLAY THE TITLE/NAME OF JOB WHO’S LAST THREE CHARACTER ARE ‘\_MGR’ AND THEIR MAXIMUM SALARY IS GREATER THAN RS 12000.



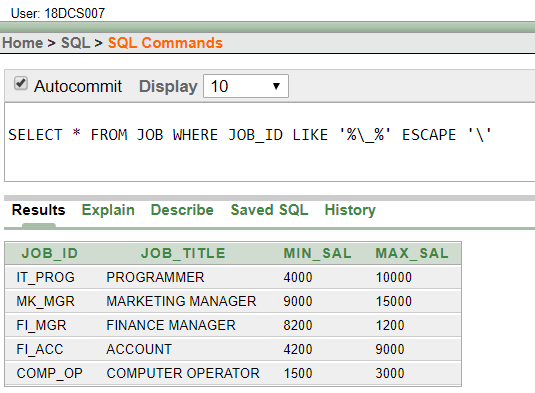
(7) DISPLAY THE NON-NULL VALUES OF EMPLOYEES AND ALSO EMPLOYEE NAME SECOND CHARACTER SHOULD BE ‘N’ AND STRING SHOULD BE 5-CHARACTER LONG.



(8) DISPLAY THE NULL VALUES OF EMPLOYEE AND ALSO EMPLOYEE NAME’S THIRD CHARACTER SHOULD BE ‘A’.



(9) WHAT WILL BE OUTPUT IF YOU ARE GIVING LIKE PREDICATE AS ‘%\\_%’ ESCAPE ‘\’

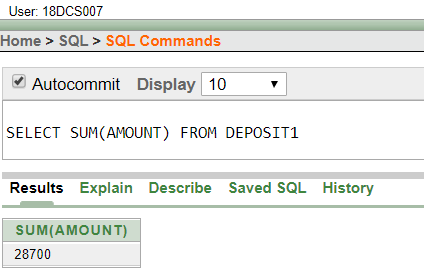


**CONCLUSION**

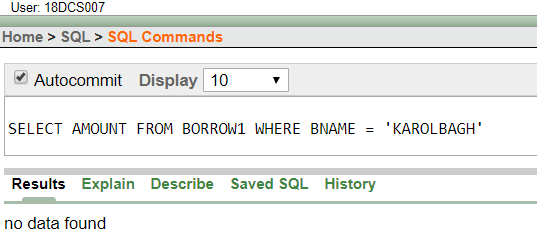
In this Practical I learnt about moderate queries and LIKE predicate

**PRACTICAL – 4**

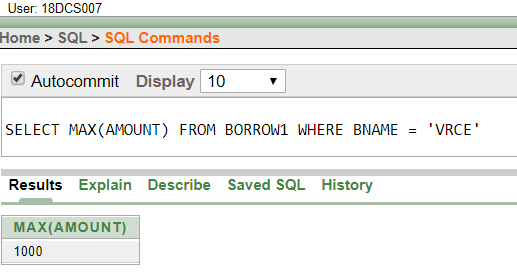
(1) LIST TOTAL DEPOSIT FROM DEPOSIT.



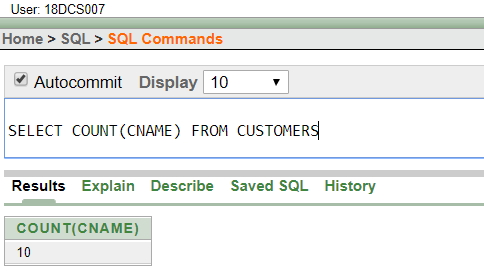
(2) LIST TOTAL LOAN FROM KAROLBAGH BRANCH



(3) GIVE MAXIMUM LOAN FROM BRANCH VRCE.



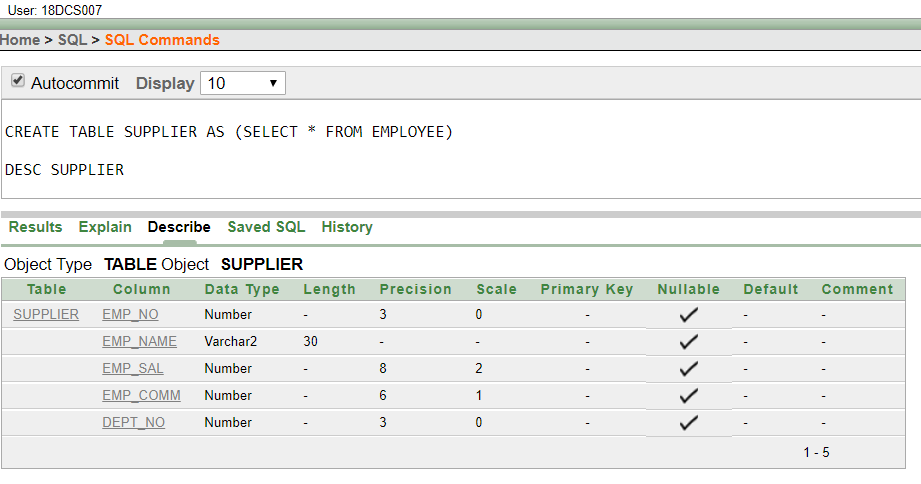
(4) COUNT TOTAL NUMBER OF CUSTOMERS

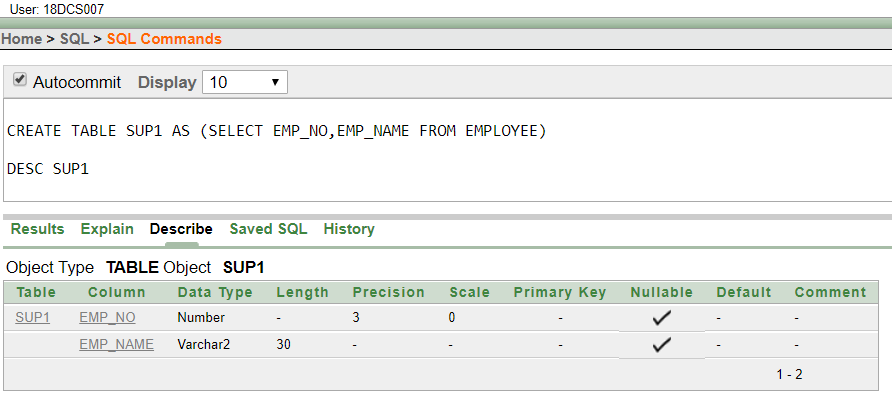


(5) COUNT TOTAL NUMBER OF CUSTOMER’S CITIES.

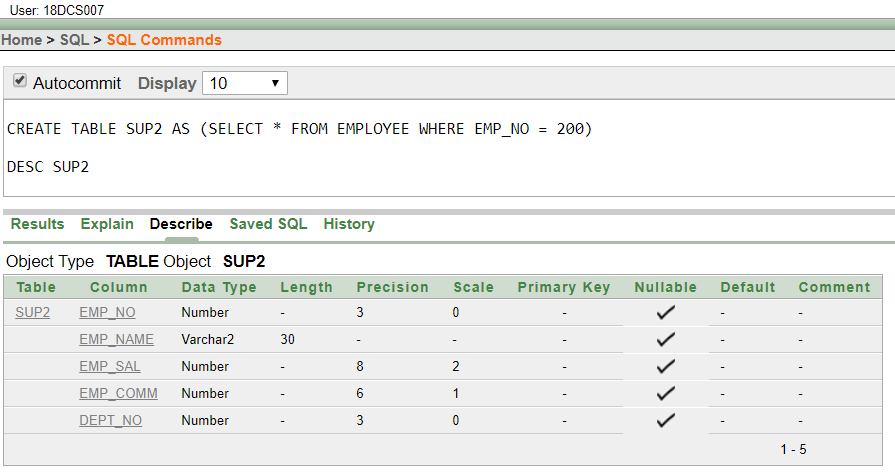


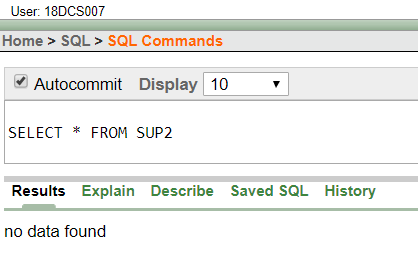
(6) CREATE TABLE SUPPLIER FROM EMPLOYEE WITH ALL THE COLUMNS.



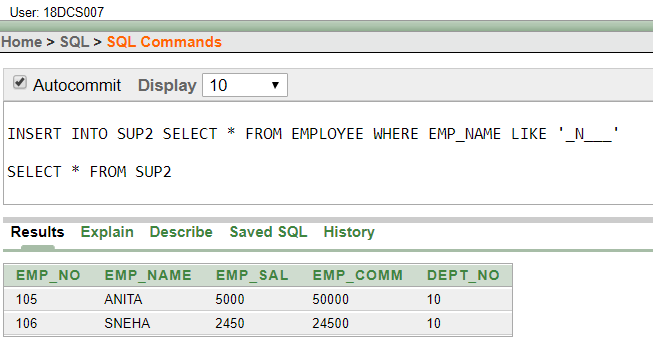
(7) CREATE TABLE SUP1 FROM EMPLOYEE WITH FIRST TWO COLUMNS.

(8) CREATE TABLE SUP2 FROM EMPLOYEE WITH NO DATA

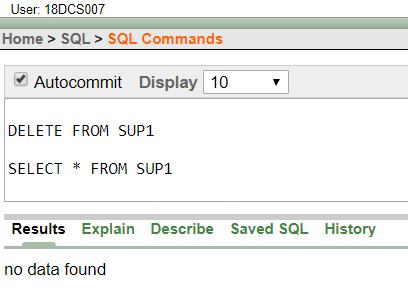




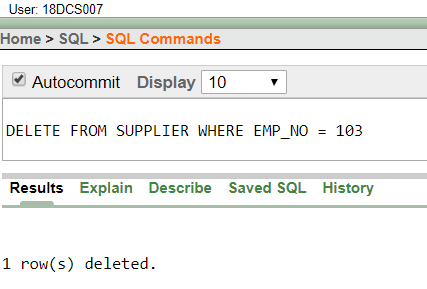
(9) INSERT THE DATA INTO SUP2 FROM EMPLOYEE WHOSE SECOND CHARACTER SHOULD BE ‘N’ AND STRING SHOULD BE 5 CHARACTERS LONG IN EMPLOYEE NAME FIELD.



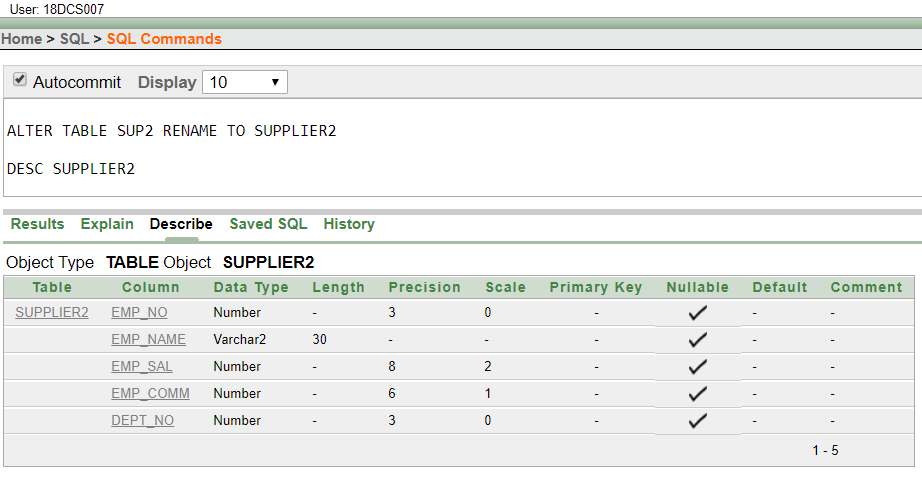
(10) DELETE ALL THE ROWS FROM SUP1.



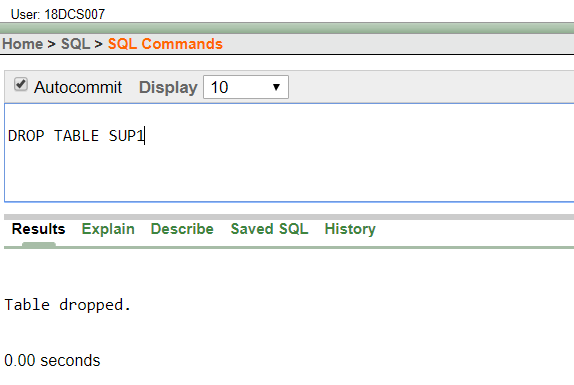
(11) DELETE THE DETAIL OF SUPPLIER WHOSE SUP\_NO IS 103.



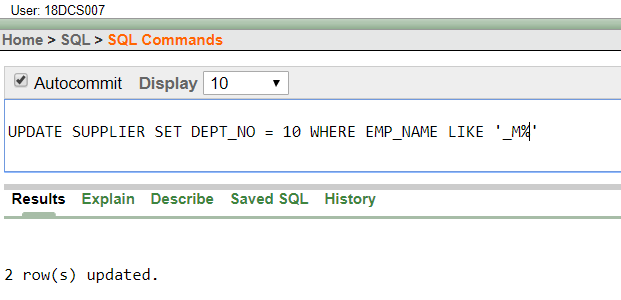
(12) RENAME THE TABLE SUP2.



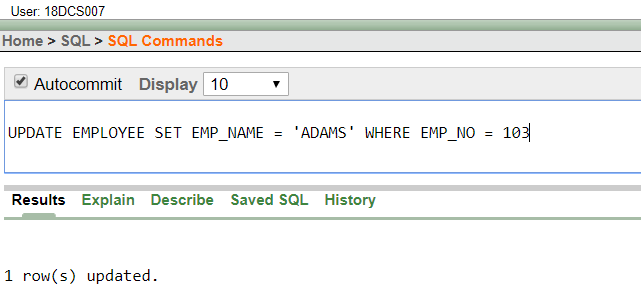
(13) DESTROY TABLE SUP1 WITH ALL THE DATA.



(14) UPDATE THE VALUE DEPT\_NO TO 10 WHERE SECOND CHARACTER OF EMP. NAME IS ‘M’.

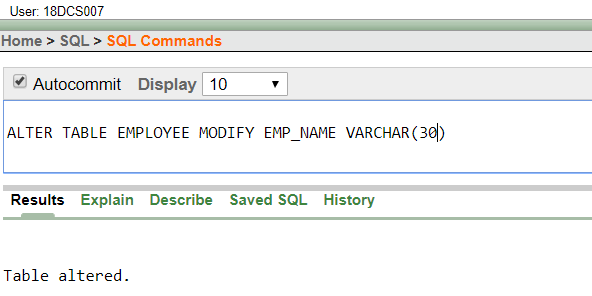


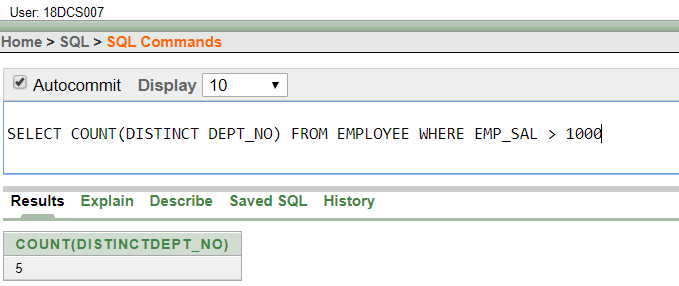
(15) UPDATE THE VALUE OF EMPLOYEE NAME WHOSE EMPLOYEE NUMBER IS 103.



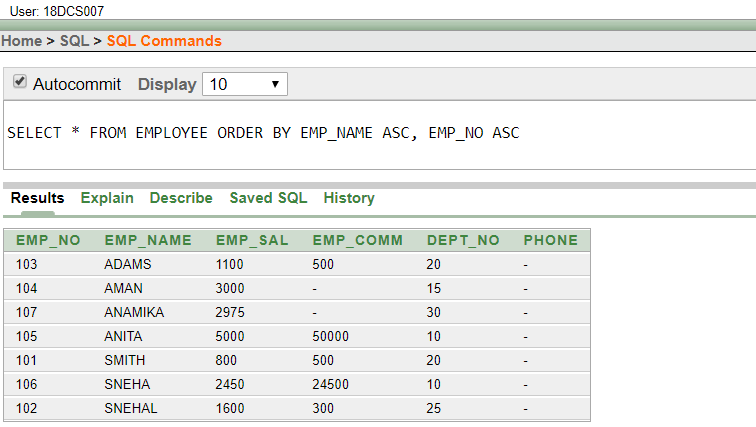
(16) ADD ONE COLUMN PHONE TO EMPLOYEE WITH SIZE OF COLUMN IS 10.

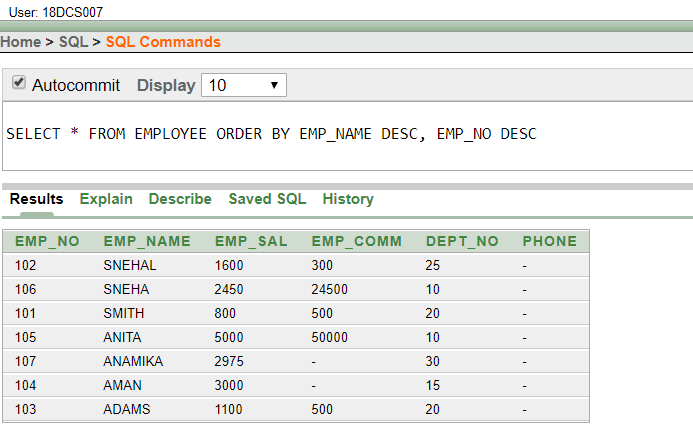


 (17) MODIFY COLUMN EMP\_NAME TO HOLD MAXIMUM OF 30 CHARACTERS.

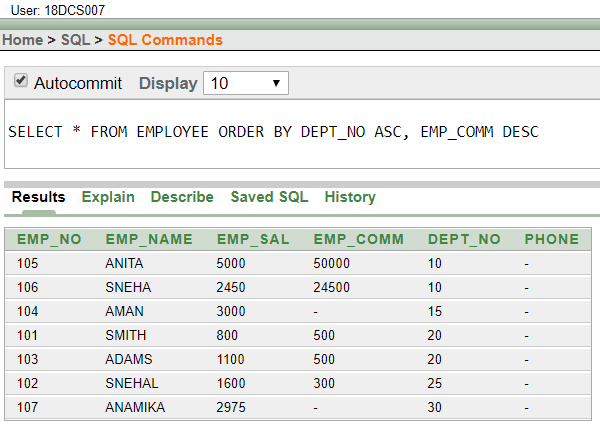
(18) COUNT THE TOTAL NO AS WELL AS DISTINCT ROWS IN DEPT\_NO COLUMN WITH A CONDITION OF SALARY GREATER THAN 1000 OF EMPLOYEE

(19) DISPLAY THE DETAIL OF ALL EMPLOYEES IN ASCENDING ORDER, DESCENDING ORDER OF THEIR NAME AND NO.

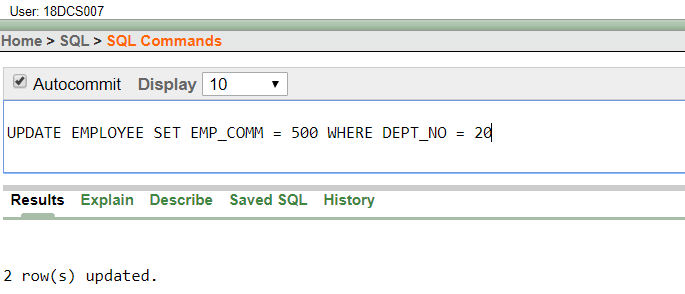




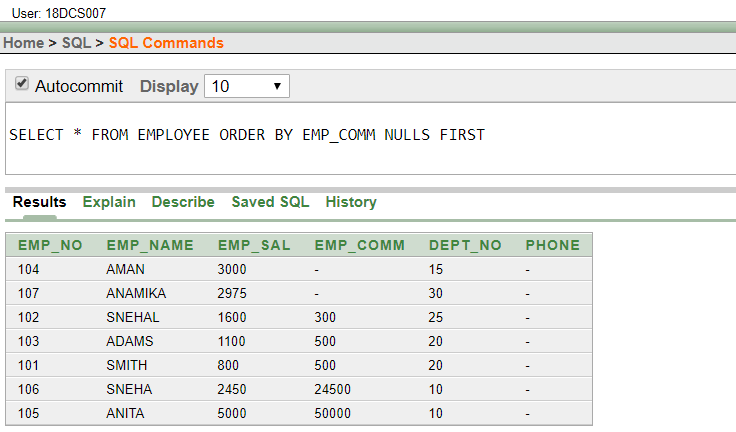
(20) DISPLAY THE DEPT\_NO IN ASCENDING ORDER AND ACCORDINGLY DISPLAY EMP\_COMM IN DESCENDING ORDER.

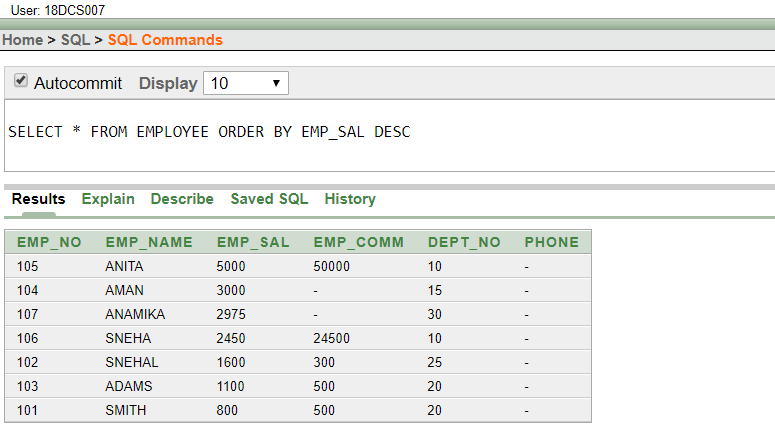


(21) UPDATE THE VALUE OF EMP\_COMM TO 500 WHERE DEPT\_NO IS 20.

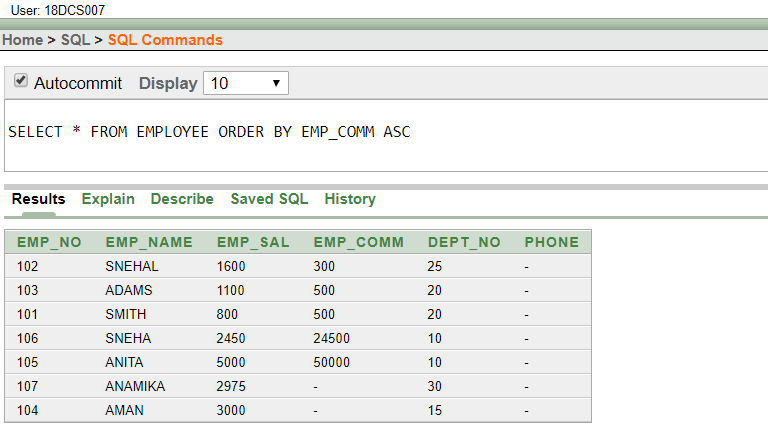


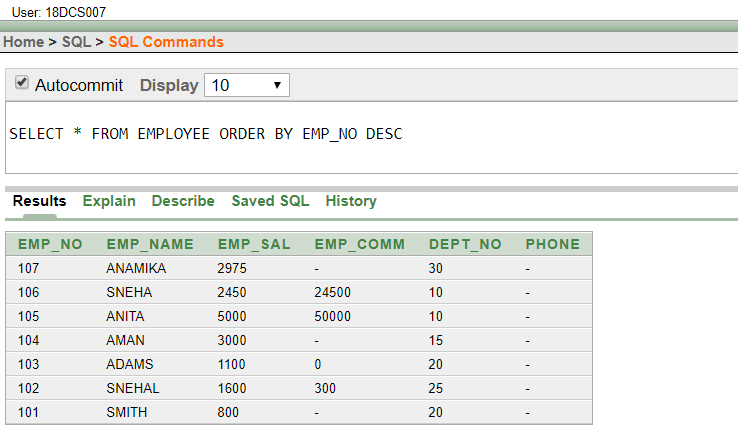
(22) DISPLAY THE EMP\_COMM IN ASCENDING ORDER WITH NULL VALUE FIRST AND ACCORDINGLY SORT EMPLOYEE SALARY IN DESCENDING ORDER.





(23) DISPLAY THE EMP\_COMM IN ASCENDING ORDER WITH NULL VALUE LAST AND ACCORDINGLY SORT EMP\_NO IN DESCENDING ORDER.





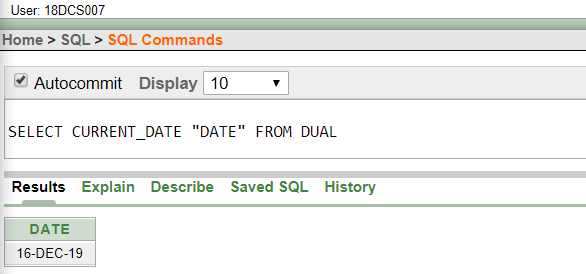
**CONCLUSION**

In this Practical I learnt to Perform various data manipulation commands, aggregate functions and sorting concept on all created tables

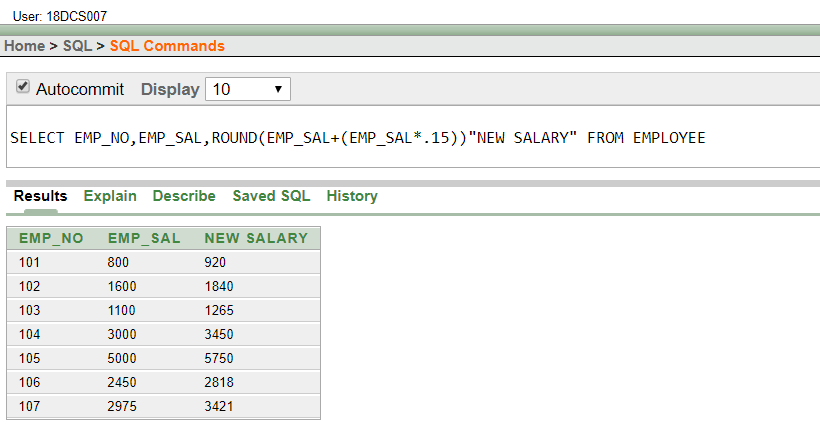
**PRACTICAL – 5**

**To Study Single-Row Functions.**

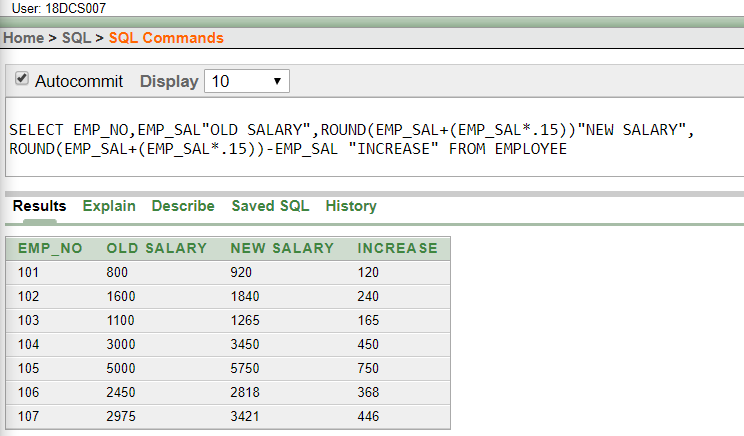
(1) WRITE A QUERY TO DISPLAY THE CURRENT DATE. LABEL THE COLUMN DATE



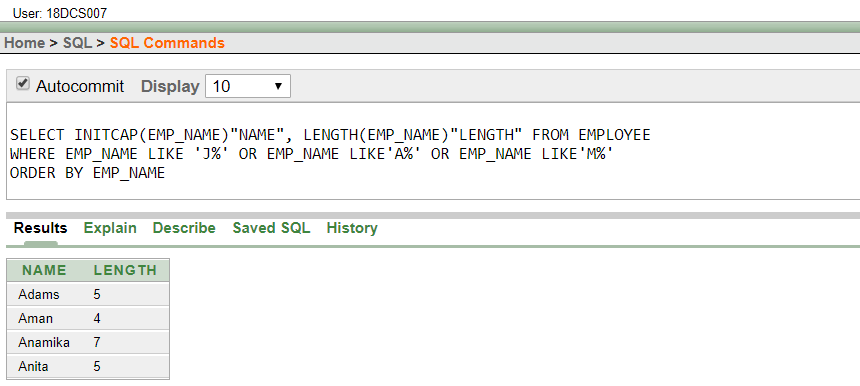
(2) FOR EACH EMPLOYEE, DISPLAY THE EMPLOYEE NUMBER, JOB, SALARY, AND SALARY INCREASED BY 15% AND EXPRESSED AS A WHOLE NUMBER. LABEL THE COLUMN NEW SALARY



(3) MODIFY YOUR QUERY NO (2) TO ADD A COLUMN THAT SUBTRACTS THE OLD SALARY FROM THE NEW SALARY. LABEL THE COLUMN INCREASE

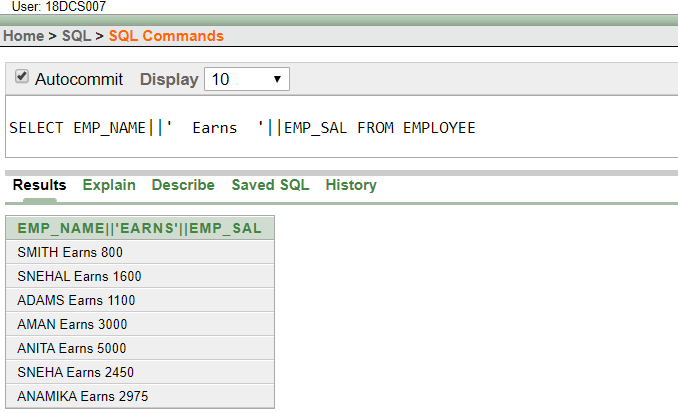


(4) WRITE A QUERY THAT DISPLAYS THE EMPLOYEE’S NAMES WITH THE FIRST LETTER CAPITALIZED AND ALL OTHER LETTERS LOWERCASE, AND THE LENGTH OF THE NAMES, FOR ALL EMPLOYEES WHOSE NAME STARTS WITH J, A, OR M. GIVE EACH COLUMN AN APPROPRIATE LABEL. SORT THE RESULTS BY THE EMPLOYEES NAMES.

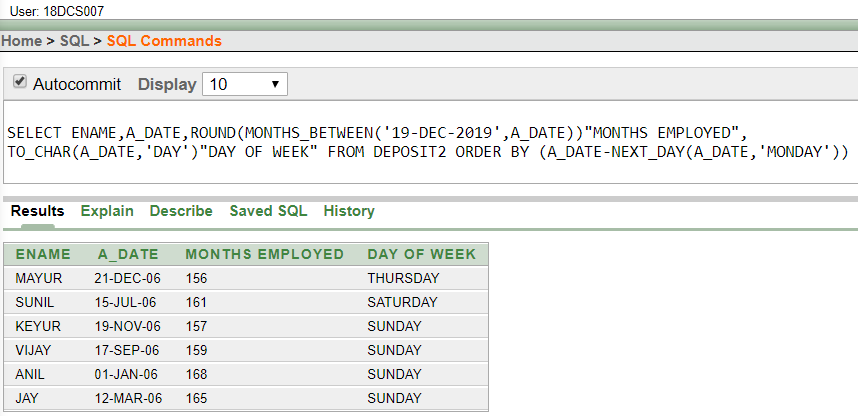


(5) WRITE A QUERY THAT PRODUCES THE FOLLOWING FOR EACH EMPLOYEE:

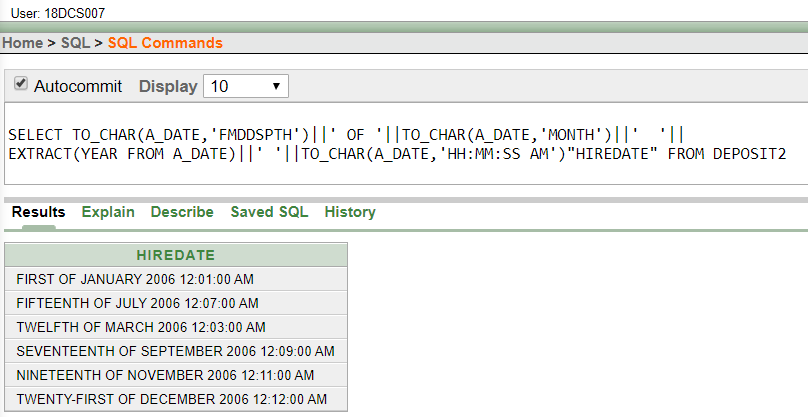
<EMPLOYEE LAST NAME> EARNS <SALARY> MONTHLY



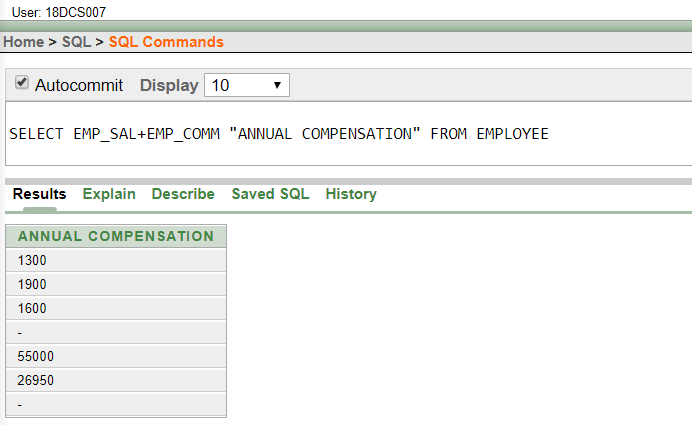
(6) DISPLAY THE NAME, HIRE DATE, NUMBER OF MONTHS EMPLOYED AND DAY OF THE WEEK ON WHICH THE EMPLOYEE HAS STARTED. ORDER THE RESULTS BY THE DAY OF THE WEEK STARTING WITH MONDAY.



(7) DISPLAY THE HIREDATE OF EMP IN A FORMAT THAT APPEARS AS SEVENTH OF JUNE 1994 12:00:00 AM.



1. WRITE A QUERY TO CALCULATE THE ANNUAL COMPENSATION OF ALL EMPLOYEES (SAL +COMM.).



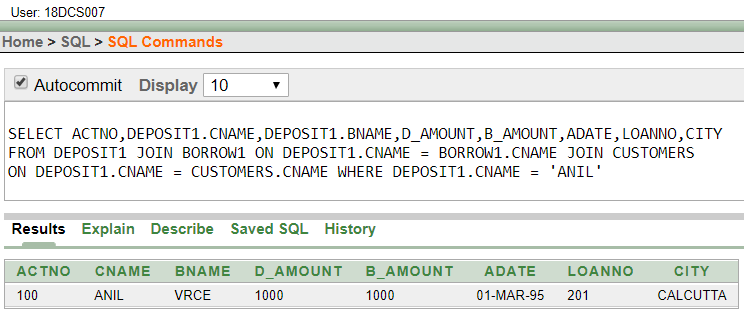
**CONCLUSION**

In this Practical I learnt Single Row Functions

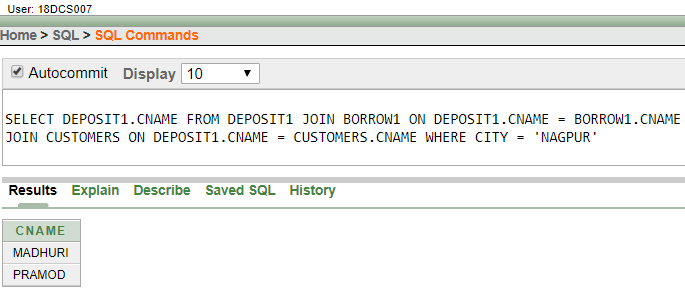
**PRACTICAL – 6**

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

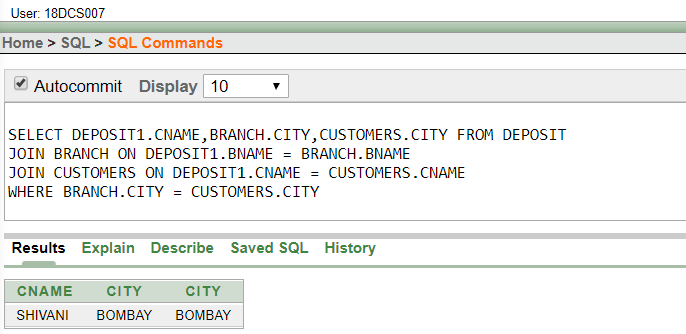
1. GIVE DETAILS OF CUSTOMERS ANIL



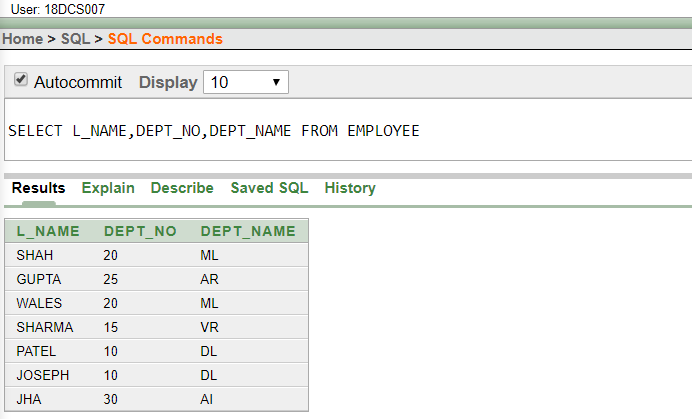
1. GIVE NAME OF CUSTOMER WHO ARE BORROWERS AND DEPOSITORS AND HAVING LIVING CITY NAGPUR



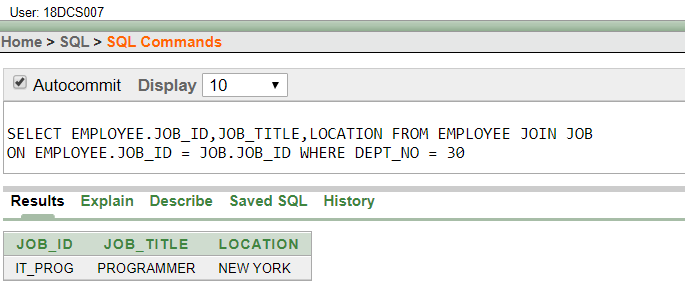
1. GIVE CITY AND NAME OF CUSTOMERS WITH SAME LIVING BRANCH.



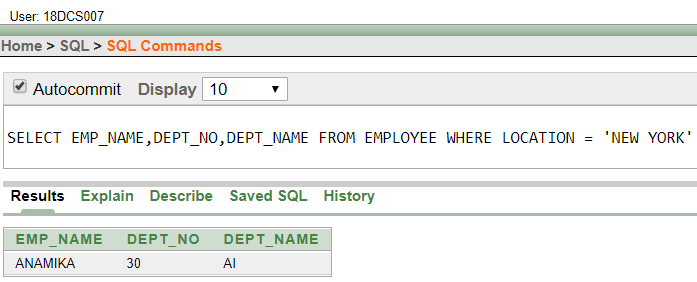
1. WRITE A QUERY TO DISPLAY THE LAST NAME, DEPARTMENT NUMBER, AND DEPARTMENT NAME FOR ALL EMPLOYEES.



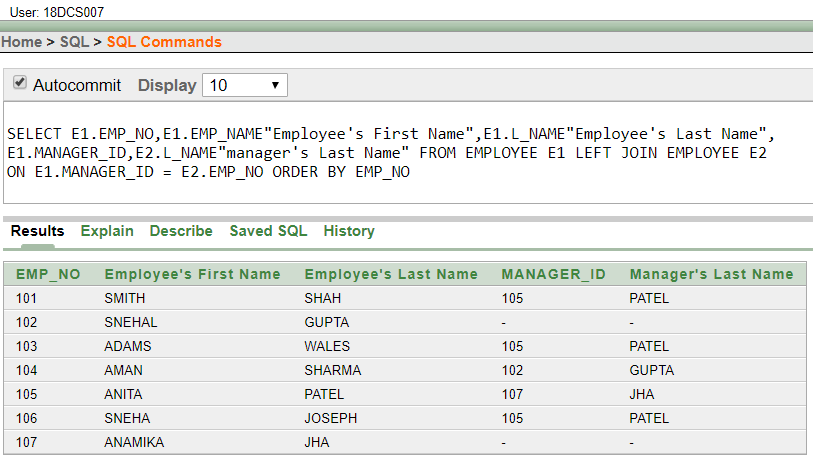
1. CREATE A UNIQUE LISTING OF ALL JOBS THAT ARE IN DEPARTMENT 30. INCLUDE THE LOCATION OF THE DEPARTMENT IN THE OUTPUT



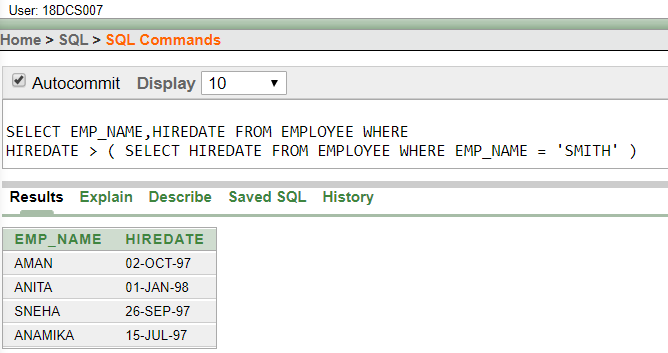
1. WRITE A QUERY TO DISPLAY THE EMPLOYEE NAME, DEPARTMENT NUMBER, AND DEPARTMENT NAME FOR ALL EMPLOYEES WHO WORK IN NEW YORK.



1. 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.



1. CREATE A QUERY TO DISPLAY THE NAME AND HIRE DATE OF ANY EMPLOYEE HIRED AFTER EMPLOYEE SMITH.



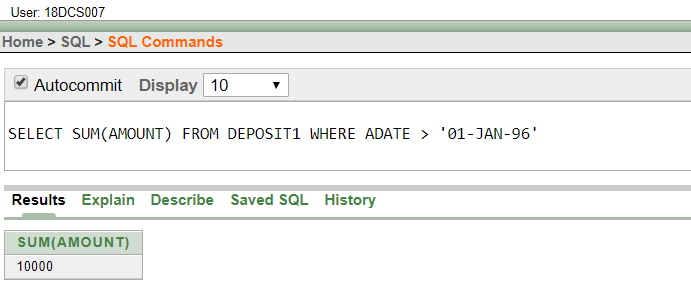
**CONCLUSION**

In this Practical I learnt to display data from Multiple Tables ( Join )

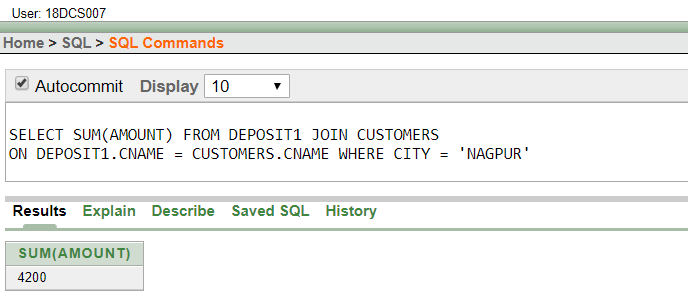
**PRACTICAL – 7**

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

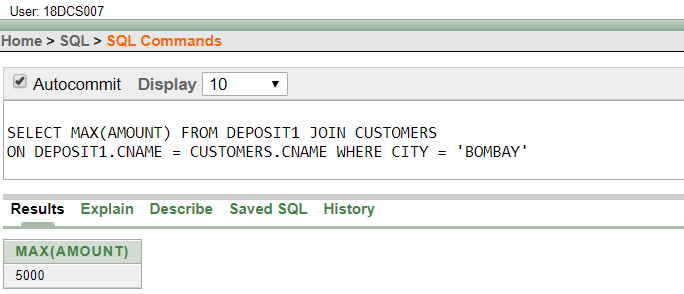
1. LIST TOTAL DEPOSIT OF CUSTOMER HAVING ACCT. DATE AFTER 1-JAN-96.



1. LIST TOTAL DEPOSIT OF CUSTOMERS LIVING IN CITY NAGPUR.

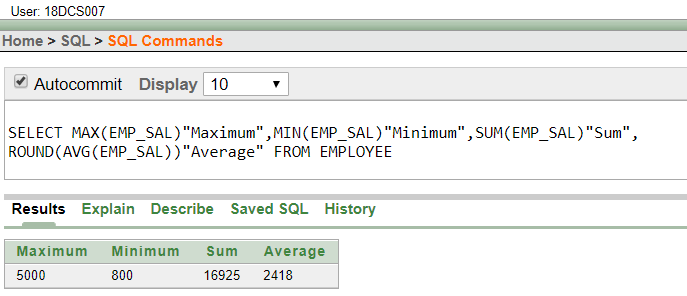


1. LIST MAXIMUM DEPOSIT OF CUSTOMERS LIVING IN BOMBAY.

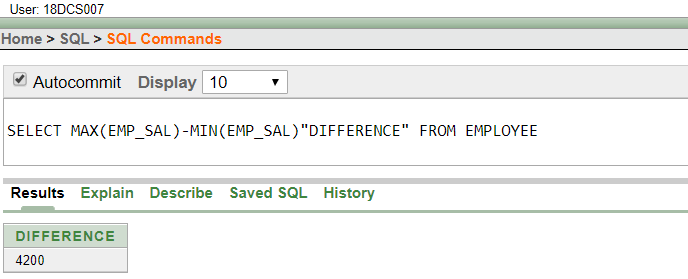


(4) DISPLAY THE HIGHEST, LOWEST, SUM, AND AVERAGE SALARY OF ALL

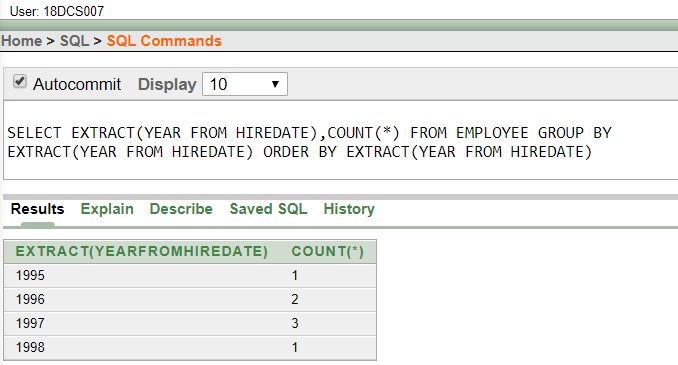
EMPLOYEES. LABEL THE COLUMNS MAXIMUM, MINIMUM, SUM, AND AVERAGE, RESPECTIVELY. ROUND YOUR RESULTS TO THE NEAREST WHOLE NUMBER.



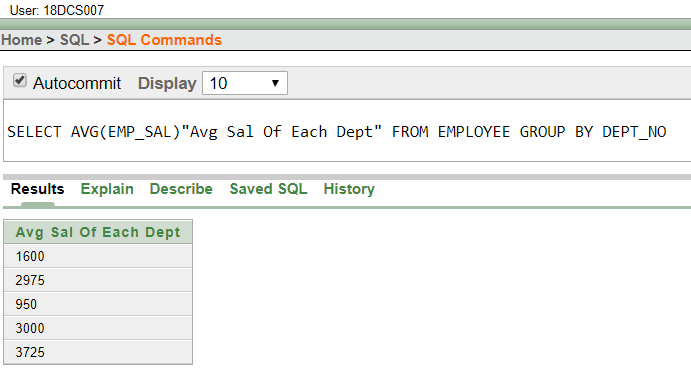
(5) WRITE A QUERY THAT DISPLAYS THE DIFFERENCE BETWEEN THE HIGHEST AND LOWEST SALARIES. LABEL THE COLUMN DIFFERENCE.



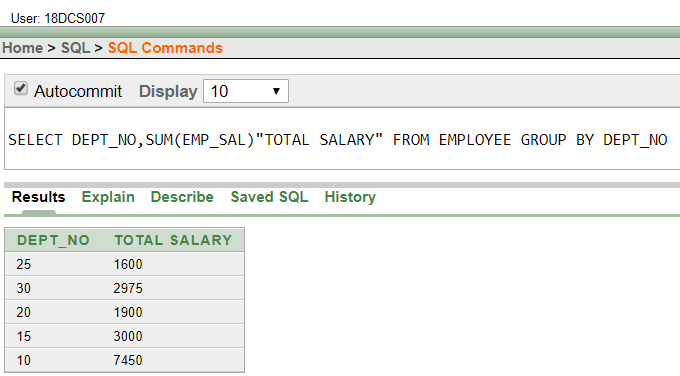
(6) CREATE A QUERY THAT WILL DISPLAY THE TOTAL NUMBER OF EMPLOYEES HIRED IN 1995, 1996, 1997, AND 1998.



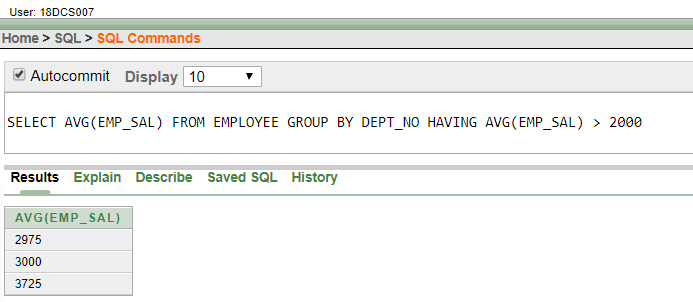
(7) FIND THE AVERAGE SALARIES FOR EACH DEPARTMENT WITHOUT DISPLAYING THE RESPECTIVE DEPARTMENT NUMBERS.



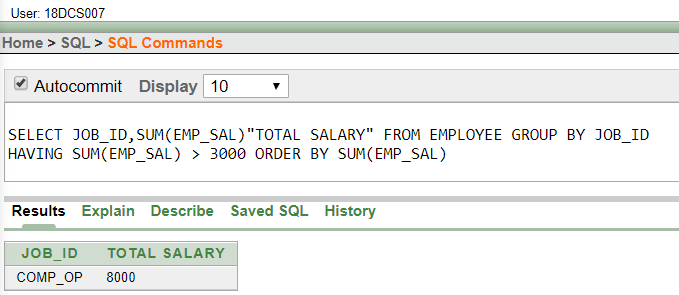
(8) WRITE A QUERY TO DISPLAY THE TOTAL SALARY BEING PAID TO EACH DEPARTMENT.



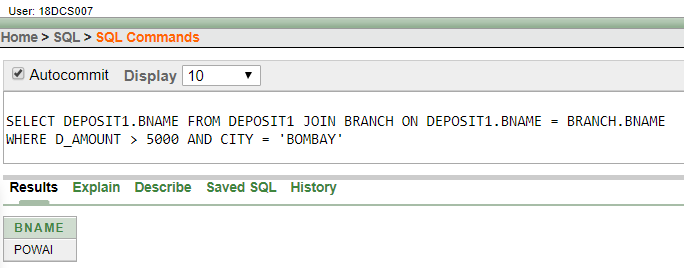
(9) FIND THE AVERAGE SALARIES > 2000 FOR EACH DEPARTMENT WITHOUT DISPLAYING THE RESPECTIVE DEPARTMENT NUMBERS.



(10) DISPLAY THE JOB AND TOTAL SALARY FOR EACH JOB WITH A TOTAL SALARY AMOUNT EXCEEDING 3000, IN WHICH EXCLUDES PRESIDENT AND SORTS THE LIST BY THE TOTAL SALARY.



(11) LIST THE BRANCHES HAVING SUM OF DEPOSIT MORE THAN 5000 AND LOCATED IN CITY BOMBAY.



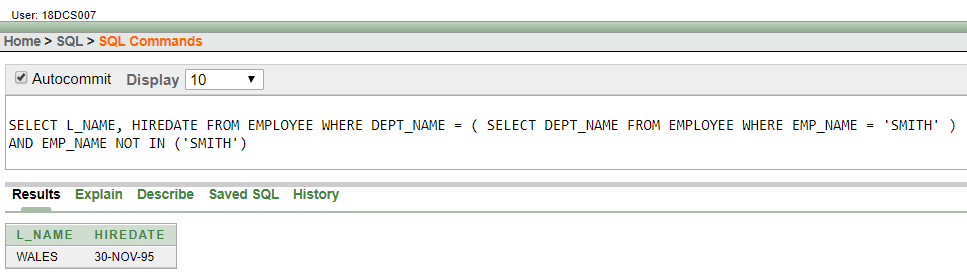
**CONCLUSION**

In this Practical I learnt to apply the concept of Aggregating Data using Group functions

**PRACTICAL – 8**

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

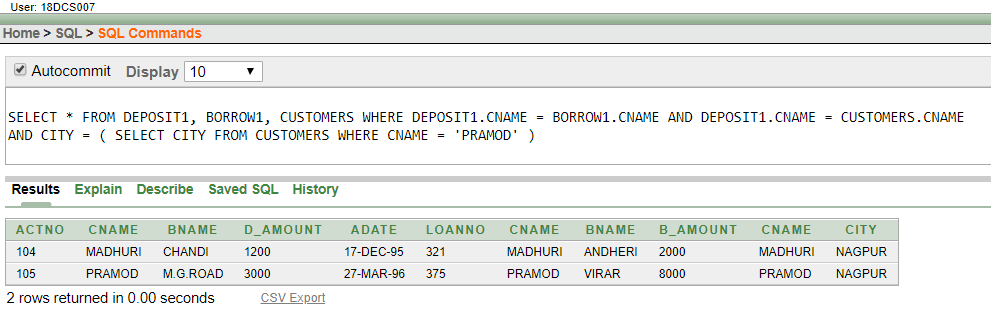
(1) WRITE A QUERY TO DISPLAY THE LAST NAME AND HIRE DATE OF ANY EMPLOYEE IN THE SAME DEPARTMENT AS SCOTT. EXCLUDE SCOTT



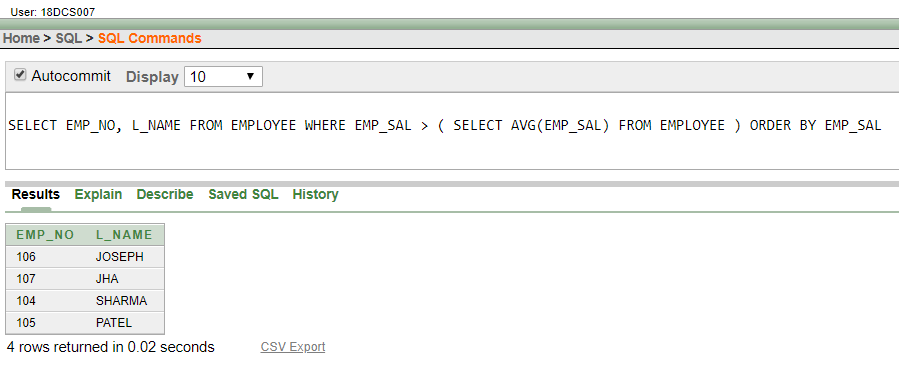
(2) GIVE NAME OF CUSTOMERS WHO ARE DEPOSITORS HAVING SAME BRANCH CITY OF MR. SUNIL.



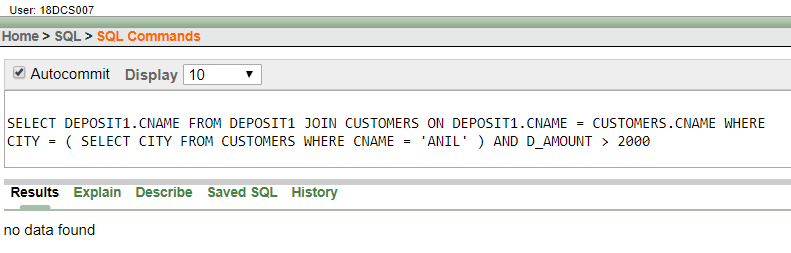
(3) GIVE DEPOSIT DETAILS AND LOAN DETAILS OF CUSTOMER IN SAME CITY WHERE PRAMOD IS LIVING.



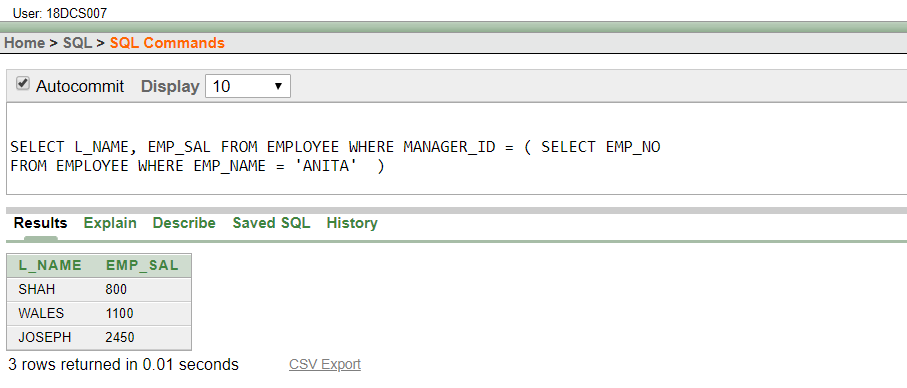
(4) CREATE A QUERY TO DISPLAY THE EMPLOYEE NUMBERS AND LAST NAMES OF ALL EMPLOYEES WHO EARN MORE THAN THE AVERAGE SALARY. SORT THE RESULTS IN ASCENDING ORDER OF SALARY.



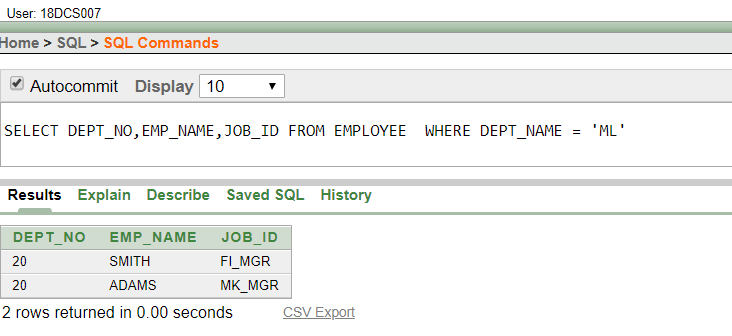
(5) GIVE NAMES OF DEPOSITORS HAVING SAME LIVING CITY AS MR. ANIL AND HAVING DEPOSIT AMOUNT GREATER THAN 2000



(6) DISPLAY THE LAST NAME AND SALARY OF EVERY EMPLOYEE WHO REPORTS TO FORD.



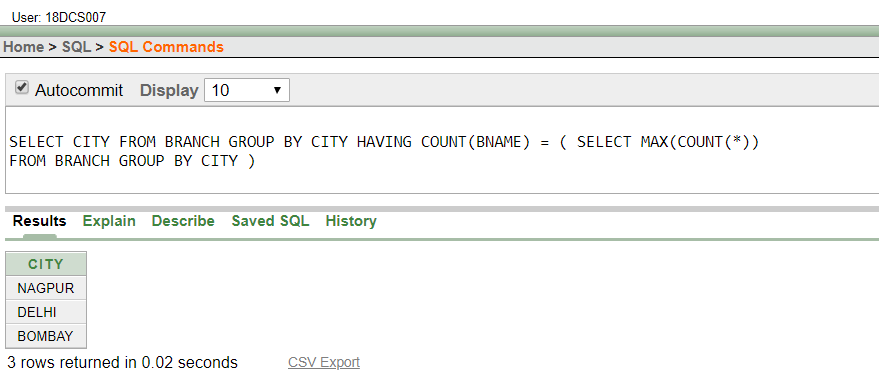
(7) DISPLAY THE DEPARTMENT NUMBER, NAME, AND JOB FOR EVERY EMPLOYEE IN THE ACCOUNTING DEPARTMENT.



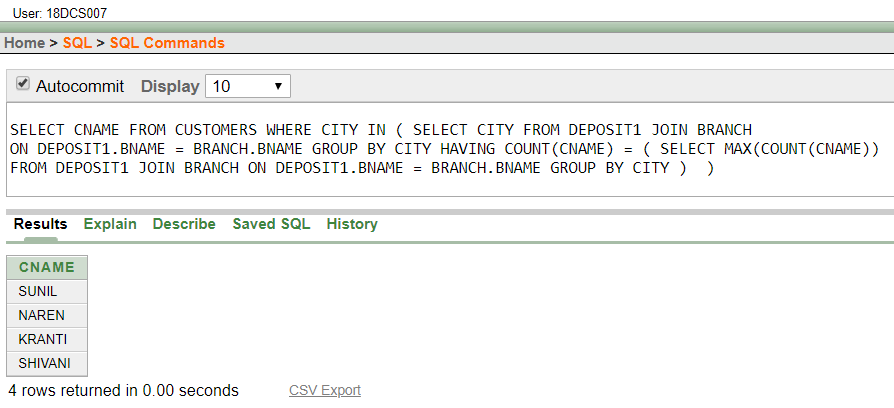
(8) LIST THE NAME OF BRANCH HAVING HIGHEST NUMBER OF DEPOSITORS.



(9) GIVE THE NAME OF CITIES WHERE IN WHICH THE MAXIMUM NUMBERS OF BRANCHES ARE LOCATED.



(10) GIVE NAME OF CUSTOMERS LIVING IN SAME CITY WHERE MAXIMUM DEPOSITORS ARE LOCATED.



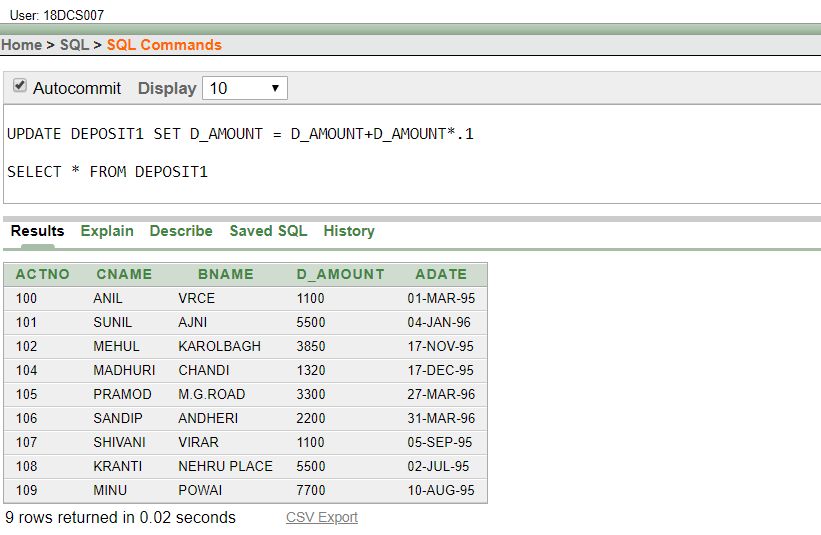
**CONCLUSION**

In this Practical I learnt to solve queries using the concept of sub query

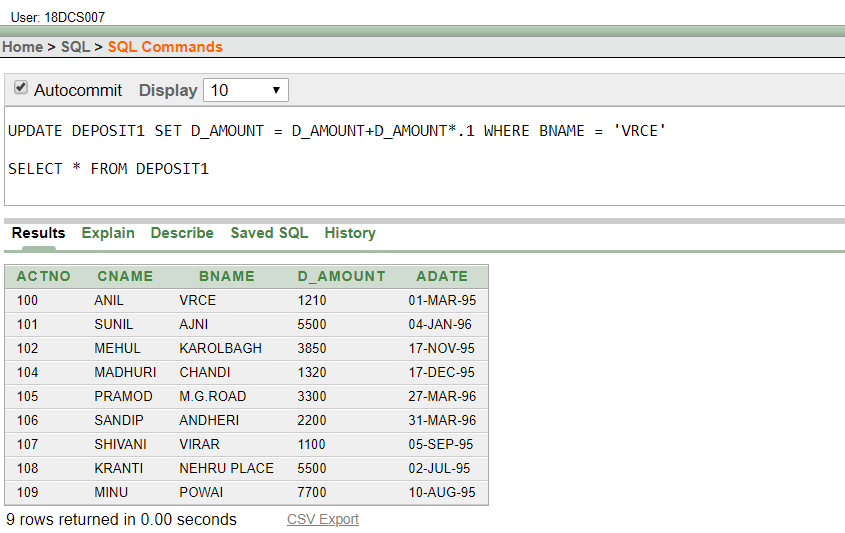
**PRACTICAL – 9**

**Manipulating Data**

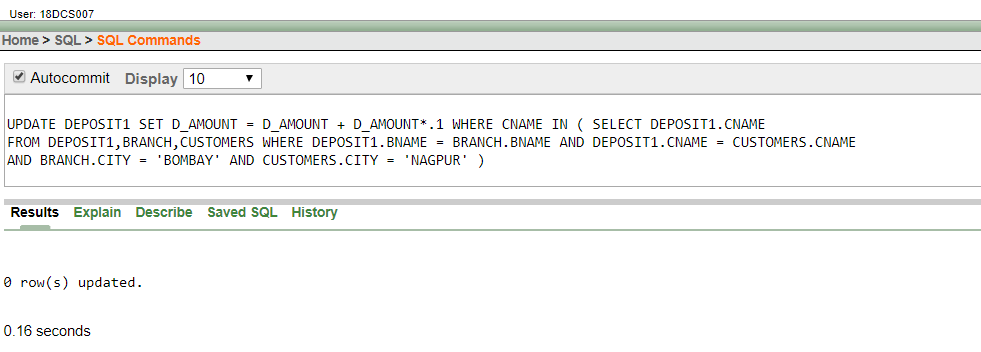
(1) GIVE 10% INTEREST TO ALL DEPOSITORS.



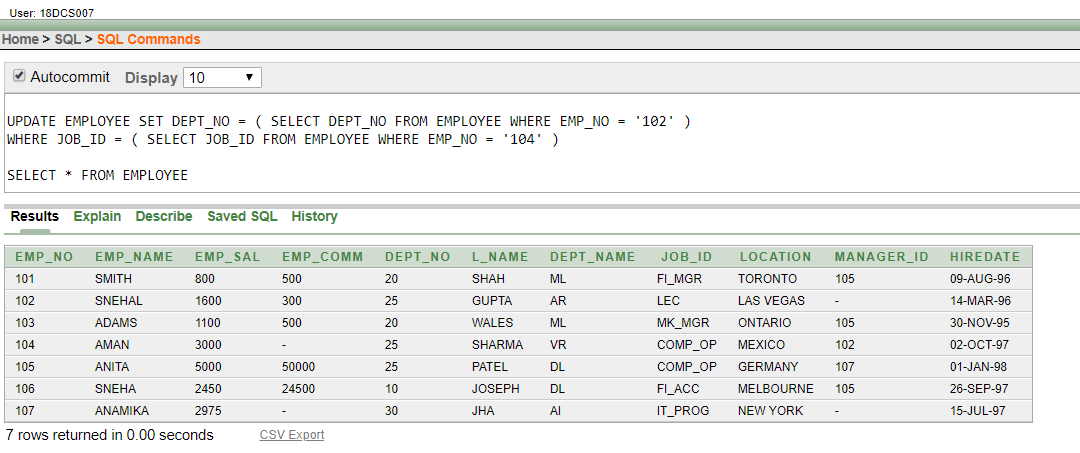
(2) GIVE 10% INTEREST TO ALL DEPOSITORS HAVING BRANCH VRCE



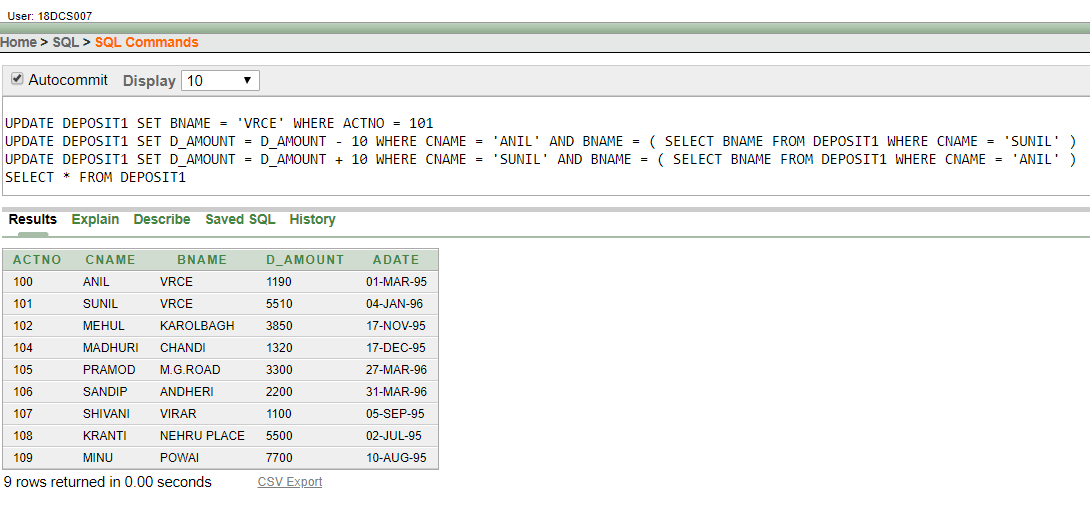
(3) GIVE 10% INTEREST TO ALL DEPOSITORS LIVING IN NAGPUR AND HAVING BRANCH CITY BOMBAY.



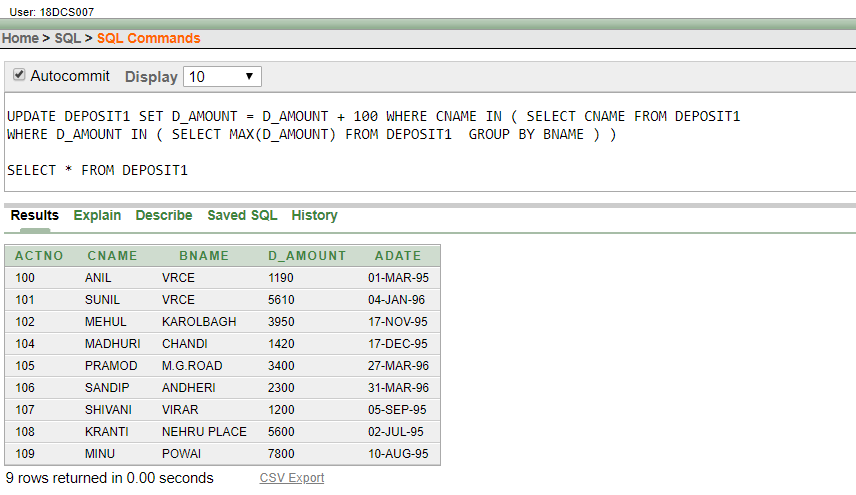
(4) WRITE A QUERY WHICH CHANGES THE DEPARTMENT NUMBER OF ALL EMPLOYEES WITH EMPNO 7788’S JOB TO EMPLOYEE 7844’CURRENT DEPARTMENT NUMBER.



(5) TRANSFER 10 RS FROM ACCOUNT OF ANIL TO SUNIL IF BOTH ARE HAVING SAME BRANCH.



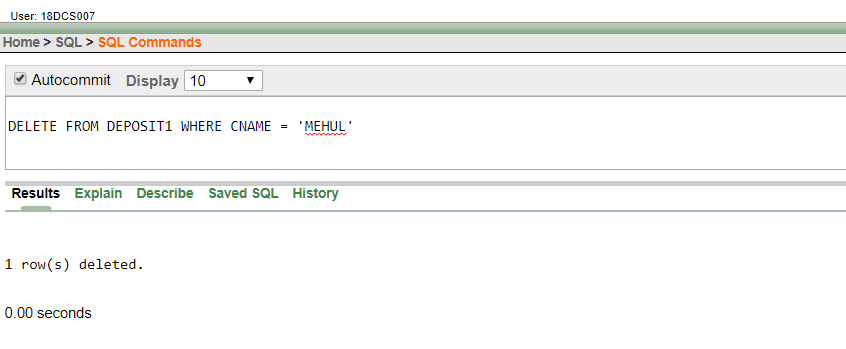
(6) GIVE 100 RS MORE TO ALL DEPOSITORS IF THEY ARE MAXIMUM DEPOSITORS IN THEIR RESPECTIVE BRANCH.



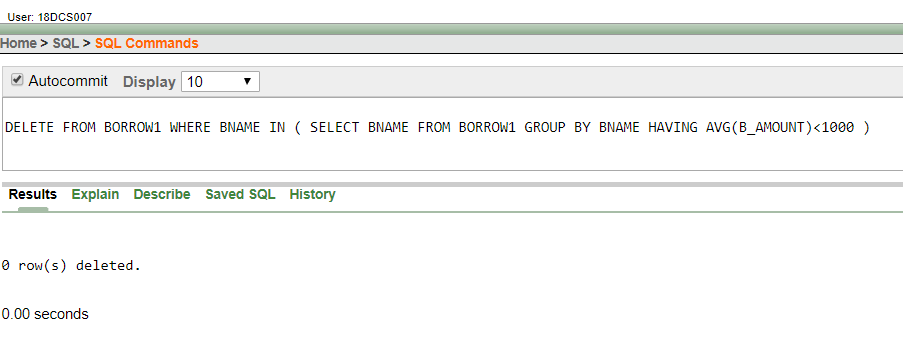
(7) DELETE DEPOSITORS OF BRANCHES HAVING NUMBER OF CUSTOMERS BETWEEN 1 TO 3.



(8) DELETE DEPOSIT OF VIJAY.



(9) DELETE BORROWER OF BRANCHES HAVING AVERAGE LOAN LESS THAN 1000.



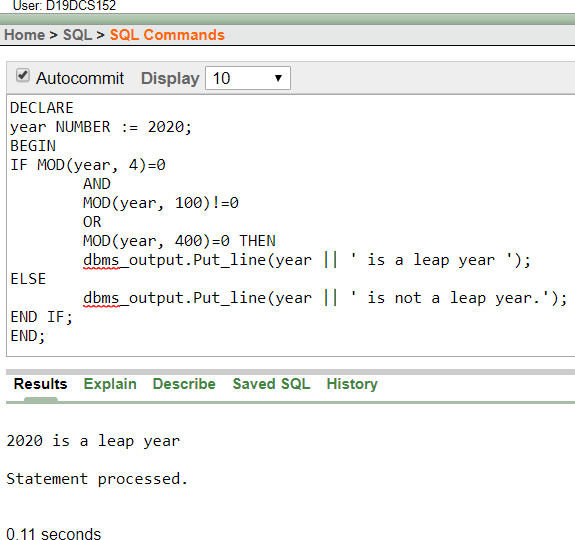
**CONCLUSION**

In this Practical I learnt how to Manipulate Data

**PRACTICAL – 10**

**To perform basic PL/SQL blocks**

WRITE A PL-SQL BLOCK FOR CHECKING WEATHER A GIVEN YEAR IS A LEAP YEAR OR NOT



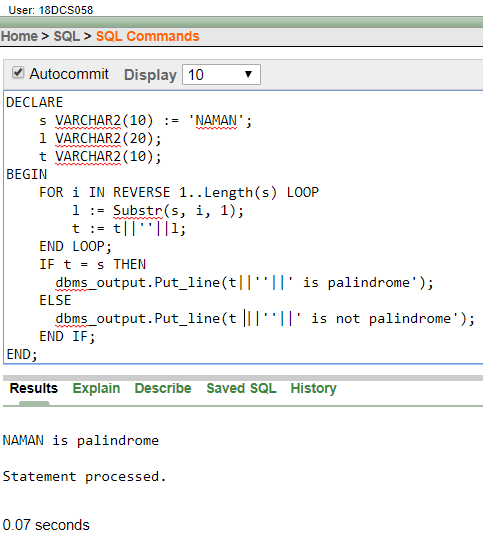
**CONCLUSION**

In this Practical I learnt to perform basic PL/SQL blocks

**PRACTICAL – 11**

**To perform the concept of loop**

FIND OUT WHETHER GIVEN STRING IS PALINDROME OR NOT USING FOR, WHILE AND SIMPLE LOOP.



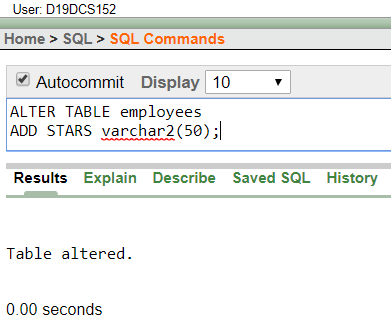
**CONCLUSION**

In this Practical I learnt to perform the concept of loop

**PRACTICAL – 12**

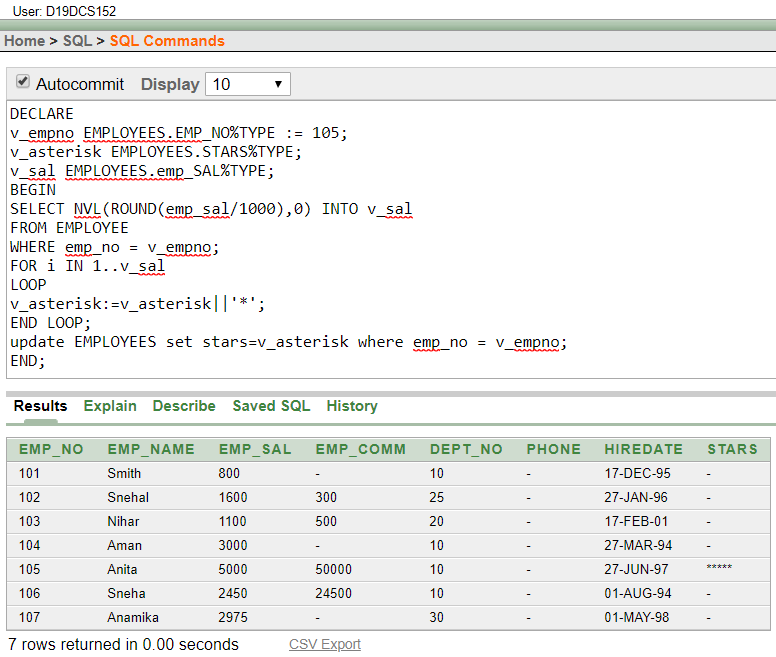
**To understand the concept of “select into” and “% type” attribute**

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



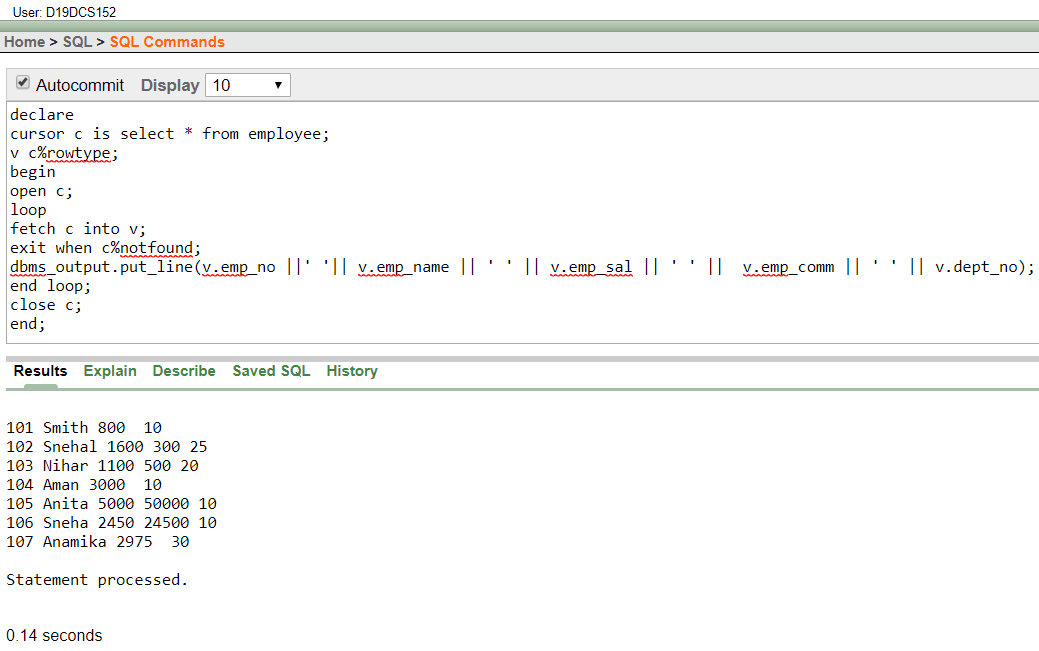
**CONCLUSION**

In this Practical I learnt to understand the concept of “select into” and “% type” attribute.

**PRACTICAL – 13**

**To perform the concept of cursor**

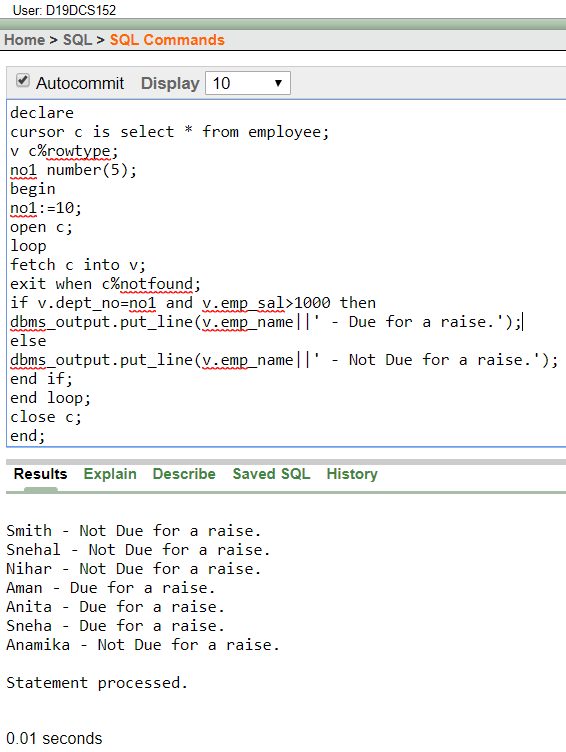
1. Display all the information of EMP table using %ROWTYPE.



(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.



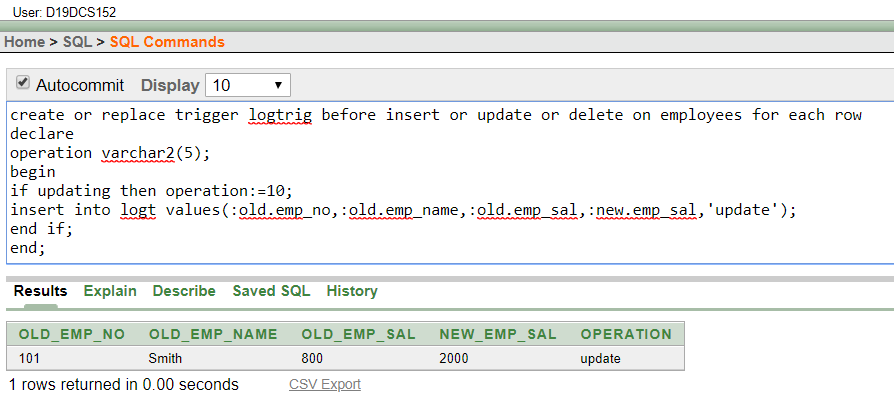
**CONCLUSION**

In this Practical I learnt to perform the concept of cursor

**PRACTICAL – 14**

**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 another table before updating take place



**CONCLUSION**

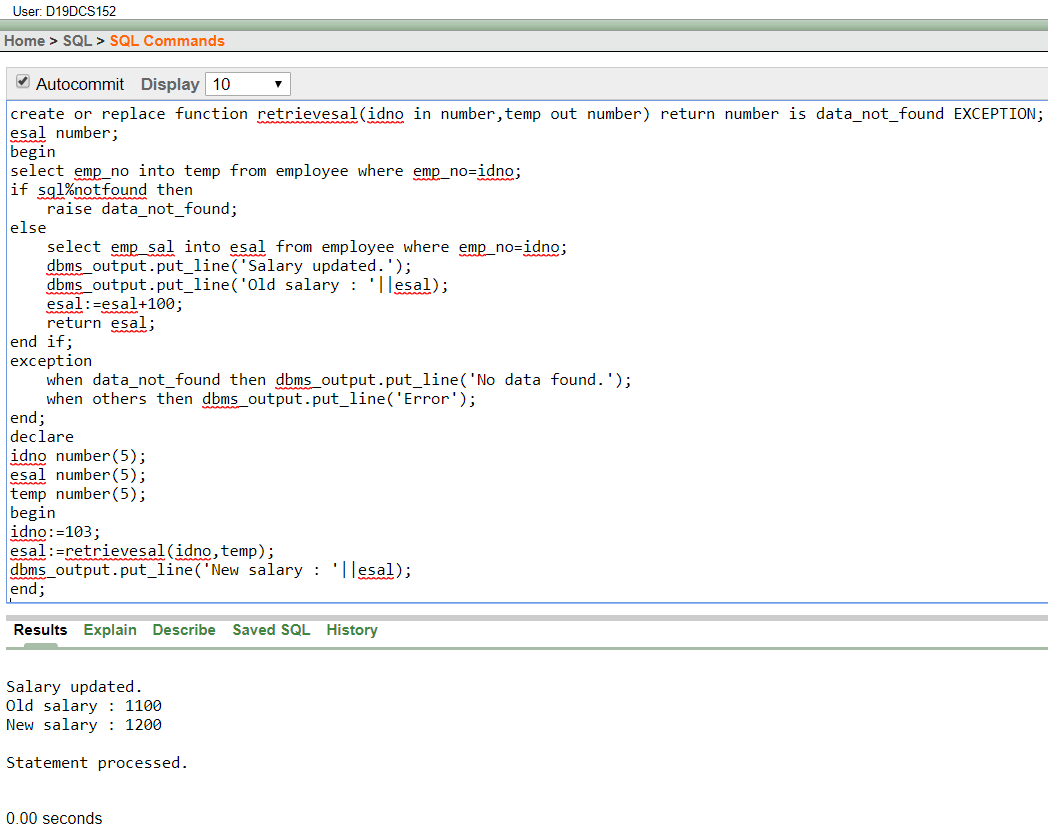
In this Practical I learnt to perform the concept of trigger

**PRACTICAL – 15**

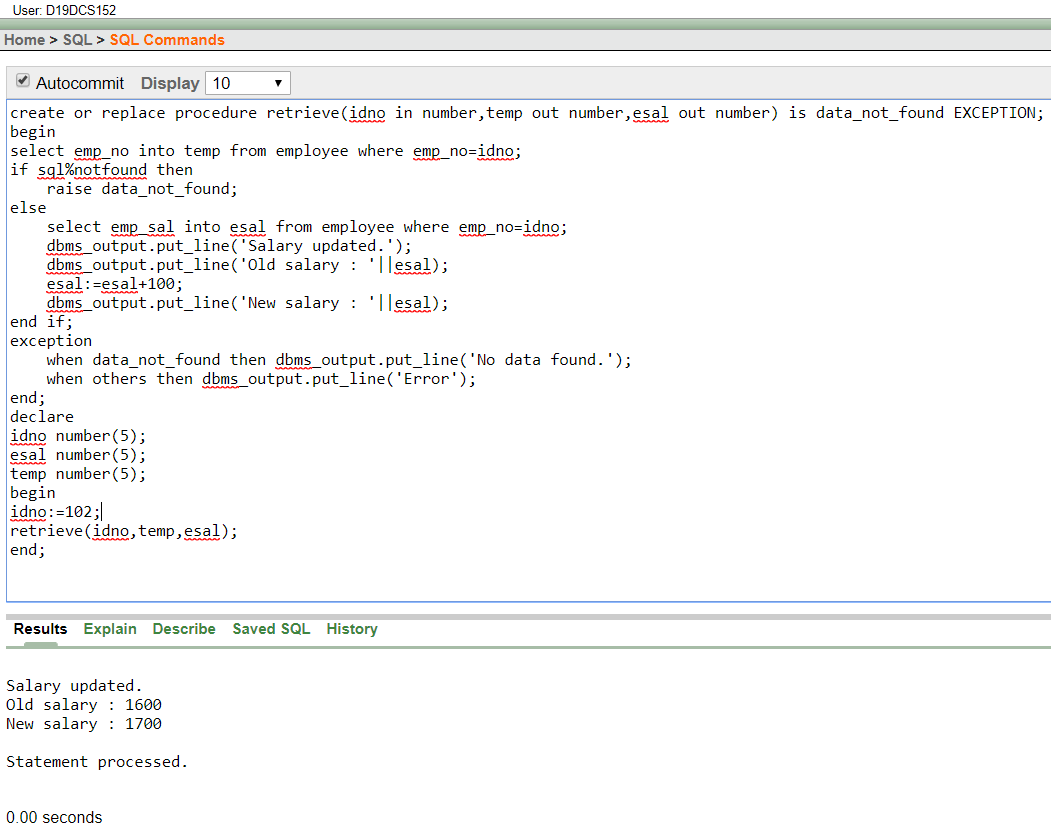
**To perform the concept of function and procedure**

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.

USING FUNCTION



USING PROCEDURE



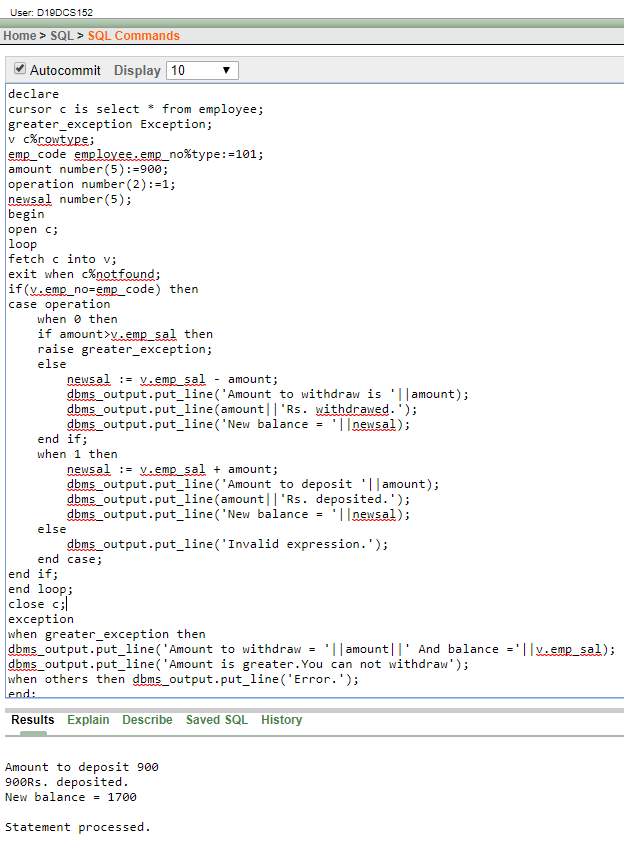
**CONCLUSION**

In this Practical I learnt to perform the concept of function and procedure

**PRACTICAL – 16**

**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.



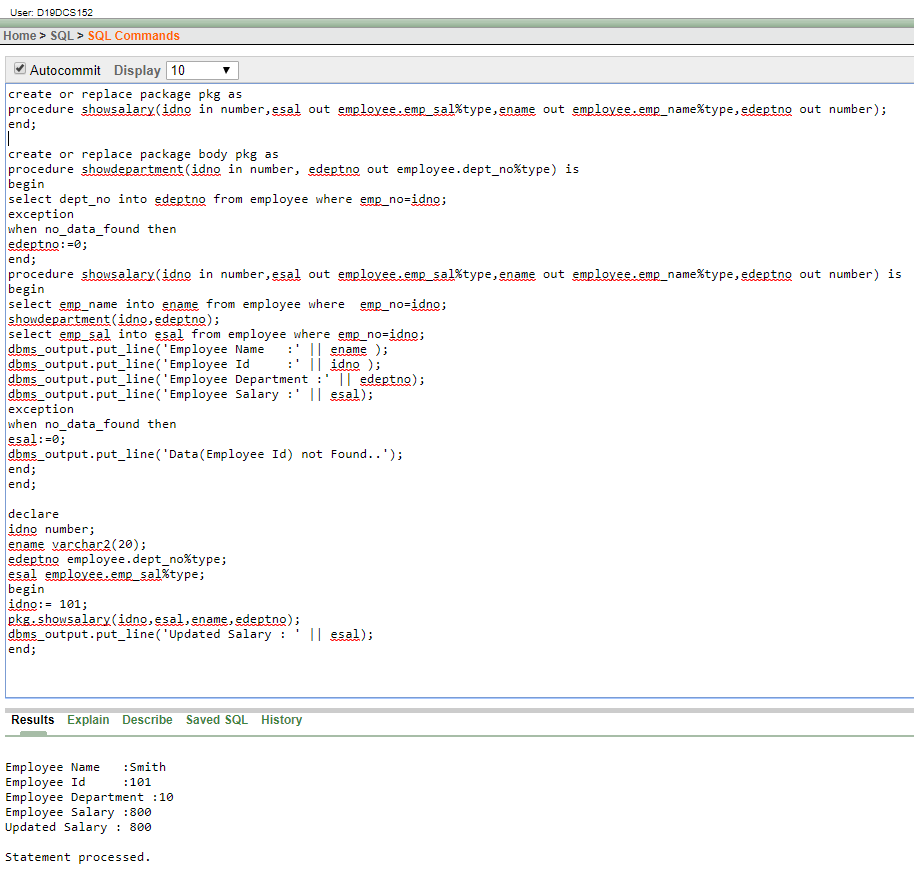
**CONCLUSION**

In this Practical I learnt to perform the concept of exception handler

**PRACTICAL – 17**

**To perform the concept of package**

CREATE AND INVOKE A PACKAGE THAT CONTAINS PRIVATE AND PUBLIC CONSTRUCTS.



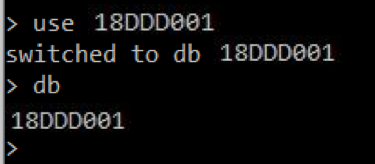
**CONCLUSION**

In this Practical I learnt to perform the concept of package

**PRACTICAL – 18**

**To create, insert values in MongoDB**

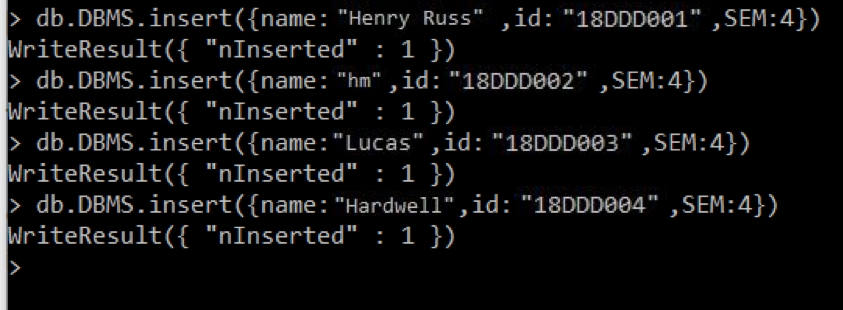
CREATE DATABASE



CREATE COLLECTION



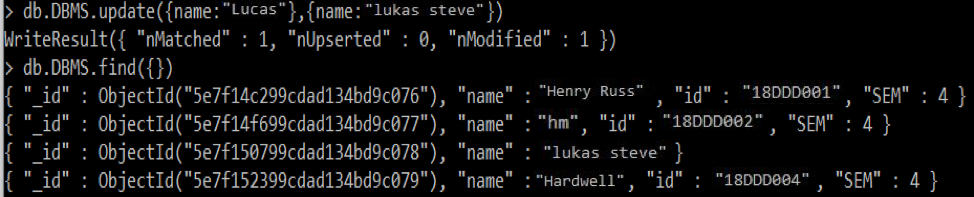
INSERT



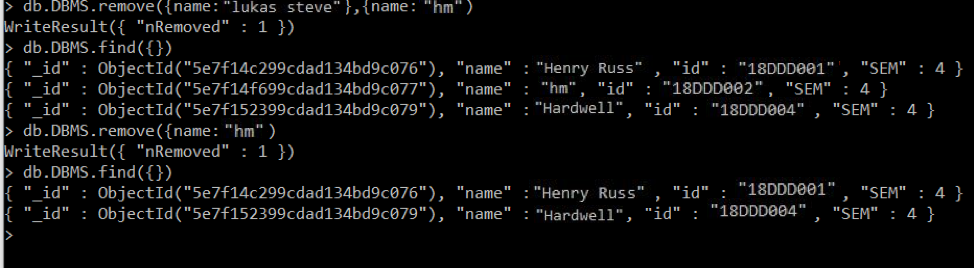
SELECT



UPDATE



DELETE



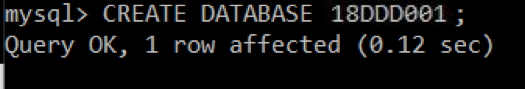
**CONCLUSION**

In this Practical I learnt to create, insert values in MongoDB

**PRACTICAL – 19**

**To create, modify, delete, execute and recompile a stored procedure in SQL Server/ MySQL**

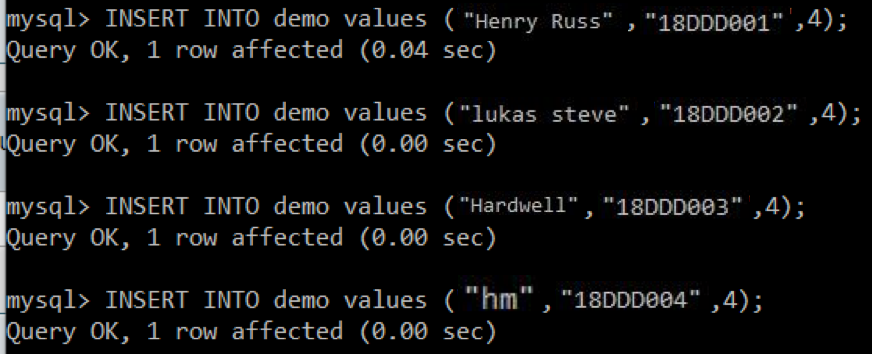
CREATE DATABASE



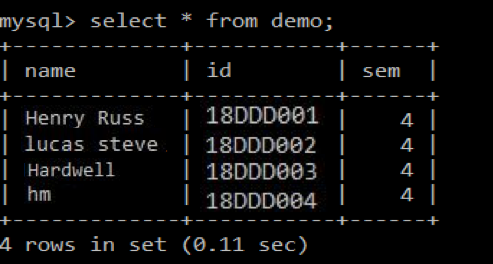
CREATE TABLE

page98image42770144

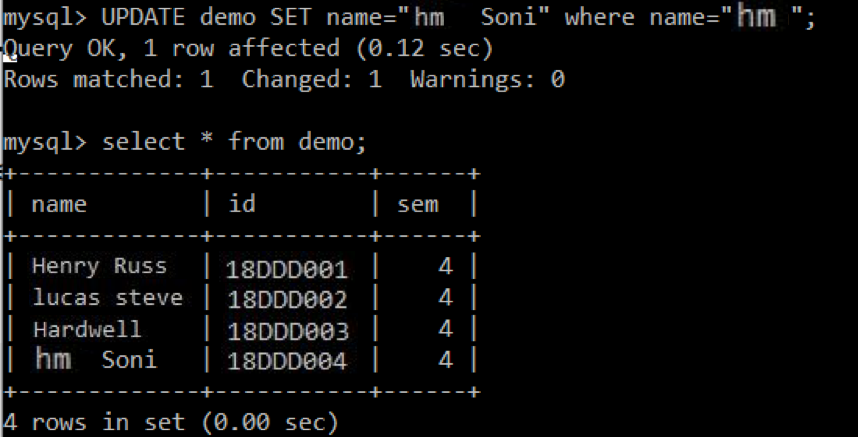
INSERT



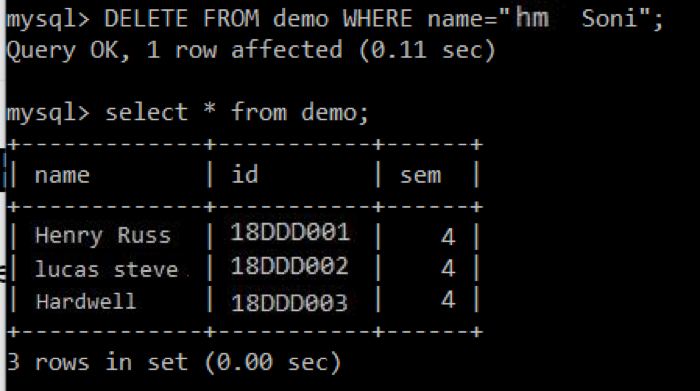
SELECT



UPDATE



DELETE



**CONCLUSION**

In this Practical I learnt to create, modify, delete, execute and recompile a stored procedure in SQL Server/ MySQL

**PRACTICAL – 20**

**Case study on Software used for NoSQL database.**

What is NoSQL ?

**NoSQL** is an approach to database design that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats. NoSQL, which stand for "not only SQL," is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built. NoSQL databases are especially useful for working with large sets of distributed data.

NoSQL encompasses a wide variety of different database technologies that were developed in response to the demands presented in building modern applications:

Developers are working with applications that create massive volumes of new, rapidly changing data types — structured, semi-structured, unstructured and polymorphic data.

Long gone is the twelve-to-eighteen month waterfall development cycle. Now small teams work in agile sprints, iterating quickly and pushing code every week or two, some even multiple times every day.

Applications that once served a finite audience are now delivered as services that must be always-on, accessible from many different devices and scaled globally to millions of users.

Organizations are now turning to scale-out architectures using open source software, commodity servers and cloud computing instead of large monolithic servers and storage infrastructure.

**How does NoSQL Database works ?**

Each of the systems labelled with the generic name works differently, but the basic idea is to offer better scalability and performance by using DB models that don't support all the functionality of a generic RDBMS, but still enough functionality to be useful. In a way it's like MySQL, which at one time lacked support for transactions but, exactly because of that, managed to outperform other DB systems. If you could write your app in a way that didn't require transactions, it was great.

**Types of NoSQL databases**

NoSQL databases typically fall into one of four categories:

* Key-value stores are the simplest. Every item in the database is stored as an attribute name (or "key") together with its value. Riak, Voldemort, and Redis are the most wellknown in this category.

* Wide-column stores store data together as columns instead of rows and are optimized for queries over large datasets. The most popular are Cassandra and HBase.

* Document databases pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents. MongoDB is the most popular of these databases.

* Graph databases are used to store information about networks, such as social connections. Examples are Neo4J and Hypergraph DB.

**Key-value store:**

* From an API perspective, key-value stores are the simplest NoSQL data stores to use. The client can either get the value for the key, assign a value for a key or delete a key from the data store.
* The value is a blob that the data store just stores, without caring or knowing what’s inside; it’s the responsibility of the application to understand what was stored.
* Since key-value stores always use primary-key access, they generally have great performance and can be easily scaled. The key-value database uses a hash table to store unique keys and pointers with respect to each data value it stores.
* There are no column type relations in the database; hence, its implementation is easy. Key-value databases give great performance and can be very easily scaled as per business needs.

**Document store:**

* It is similar to key-value databases in that there’s a key and a value. Data is stored as a value. Its associated key is the unique identifier for that value.
* The difference is that, in a document database, the value contains structured or semi-structured data. This structured/semi-structured value is referred to as a document and can be in XML, JSON or BSON format.

**Column store:**

* In column-oriented NoSQL databases, data is stored in cells grouped in columns of data rather than as rows of data. Columns are logically grouped into column families. Column families can contain a virtually unlimited number of columns that can be created at runtime or while defining the schema.
* Read and write is done using columns rather than rows. Column families are groups of similar data that is usually accessed together. As an example, we often access customers’ names and profile information at the same time, but not the information on their orders.

**Graph base:**

* Graph databases are basically built upon the Entity – Attribute – Value model. Entities are also known as nodes, which have properties. It is a very flexible way to describe how data relates to other data.
* Nodes store data about each entity in the database, relationships describe a relationship between nodes, and a property is simply the node on the opposite end of the relationship. Whereas a traditional database stores a description of each possible relationship in foreign key fields or junction tables, graph databases allow for virtually any relationship to be defined on-the-fly.

**Comparison between types of NoSQL database :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data model | Performance | Scalability | Flexibility | Complexity | Functionality |
| Key–  value store | high | high | high | none | variable (none) |
| Column-  oriented store | high | high | moderate | low | minimal |
| Document  -oriented store | high | variable (high) | high | low | variable (low) |
| Graph database | variable | variable | high | high | graph theory |

**CONCLUSION**

In this Practical I learnt Case study on Software used for NoSQL database.