**Oracle:**

Oracle is one of the powerful RDBMS products that provide efficient solutions for database applications. Oracle is the product of Oracle Corporation which was founded by LAWRENCE ELLISION in 1977. The first commercial product of oracle was delivered in 1970. The first version of oracle 2.0 was written in assembly language. Nowadays commonly used versions of oracle are ORACLE 8, 8i and 9i, Oracle 8 and onwards provide tremendous increase in performance, features and functionality.

**FEATURES OF ORACLE**

* Client/Server Architecture
* Large database and Space Management
* Concurrent Processing
* High transaction processing performance
* High Availability
* Many concurrent database users
* Controlled availability
* Openness industry standards
* Manageable security
* Database enforced integrity
* Distributed systems
* Portability
* Compatibility

**ORACLE SERVER TOOL**

Oracle is a company that produces most widely used server based multi-user RDBMS. Oracle server is a program installed on server hard-disk drive. This program must be loaded in RAM to that it can process the user requests. Oracle server takes care of following functions. Oracle server tools are also called as back end. Functions of server tool:

* Updates the data
* Retrieves the data sharing
* Manages the data sharing
* Accepts the query statements PL/SQL and SQL
* Enforce the transaction consistency

**ORACLE CLIENT TOOL**

Once Oracle engine is loaded into sever memory user would have to log in to engine in order to work done. Client tools are more useful in commercial application development. It provides facilities to work on database objects. These are more commonly used in commercial applications. Oracle client tools are also called front end.

**DDL – DATA DEFINATION LANGUAGE**

SQL sentences that are used to create these objects are called DDL’s or Data Definition Language. The SQL provides various commands for defining relation schemas, deleting relations, creating indexes and modify relation schemas. DDL is part of SQL which helps a user in defining the data structures into the database. Following are the various DDL commands are:

* Alter table, create table & drop table
* Create index & drop index
* Create view & drop view

**DML – DATA MANIPULATION LANGUAGE**

The SQL sentences used to manipulate data within these objects are called DML’s or Data Manipulation Language. It is language that enables users to access or manipulate data as organized by appropriate data model. By data manipulation we have

* Retrieval of information stored in database.
* Insertion of new information into database.
* Deletion of information from database.
* Modification of data stored in database.

Two types of DML are

* Procedural DML
* Non-procedural DML

Following are DML commands are

* Select
* Update
* Delete
* Insert

**DCL – DATA CONTROL LANGUAGE**

The SQL sentences, which are used to control the behavior of these objects, are called DCL’s or Data Control Language. It is language used to control data and access to the database. Following are some DCL commands are :

* Commit
* Rollback
* Save point
* Set transaction

**DATA TYPES OF SQL**

* **CHAR :** This data type is used to store character strings values of fixed length. The size in brackets determines the number of characters the cell can hold. The maximum number of characters (i.e. the size) this data type can hold is 255 characters. Syntax is CHAR(SIZE)

Example is CHAR (20)

* **VARCHAR** : This data type is used to store variable length alphanumeric data. The maximum this data type can hold is 4000 characters. One difference between this data type and the CHAR data type is ORACLE compares VARCHAR values using non-padded comparison semantics i.e. the inserted values will not be padded with spaces. Syntax is VARCHAR(SIZE)

Example is VARCHAR (20) OR VARCHAR2 (20)

* **NUMBER** : The NUMBER data type is used to store numbers (fixed or floating point). Numbers of virtually any magnitude maybe stored up to 38 digits of precision. Numbers as large as 9.99 \* 10 to the power of 124, i.e. followed by 125 zeros can be stored. The precision, (P), determines the maximum length of the data, whereas the scale, (S), determines the number of places to the right of the decimal. If scale is omitted then the default is zero. If precision is omitted values are stored with their original precision up to the maximum of 38 digits.

Syntax : NUMBER (P, S) Example is NUMBER (10, 2)

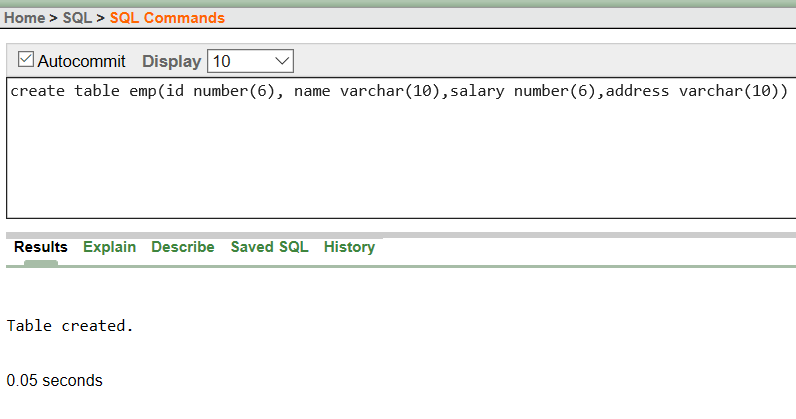
* **DATE** : This data type is used to represent data and time. The standard format id DD-MM-YY as in 13-JUL-85. To enter dates other than the standard format, use the appropriate functions. Date Time stores date in the 24-hour format. By default, the time in a date field is 12:00:00 am, if no time portion is specified. The default date for a date field is the first day of the current month.

Syntax : DATE

1. **Create table**

Relations can be described in the form of tables, and in SQL :

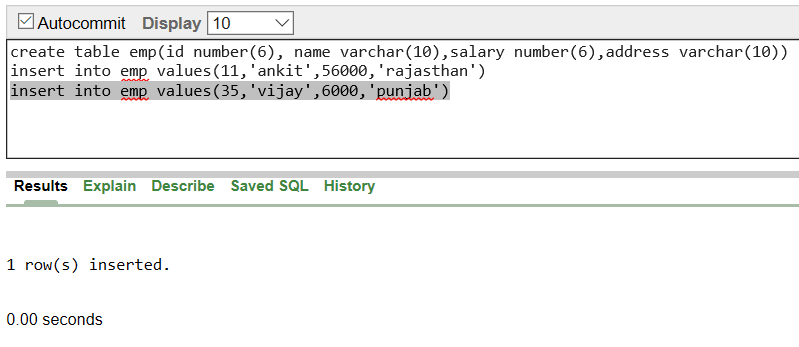
***create table table\_name( column\_name datatype, … );***



1. **Insert**

To enter the data, we can do manually by entering each data input with proper input values. Null can also be written if the value is unknown or if do not exist.

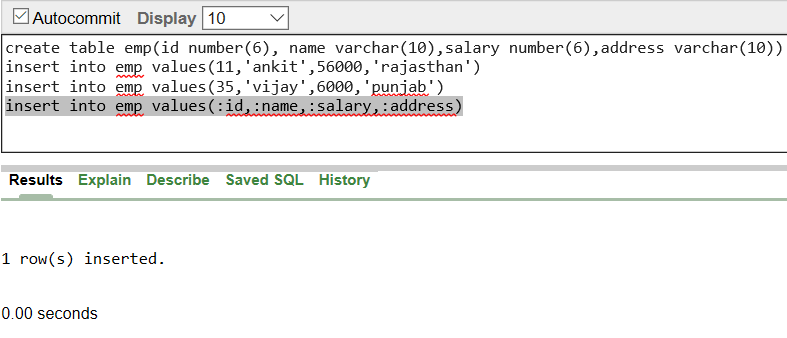
***insert into table\_name values( data…) ;***



1. **Insert with form means**

We can add the tuples by entering each and every input data according to the form means. One of the method is :

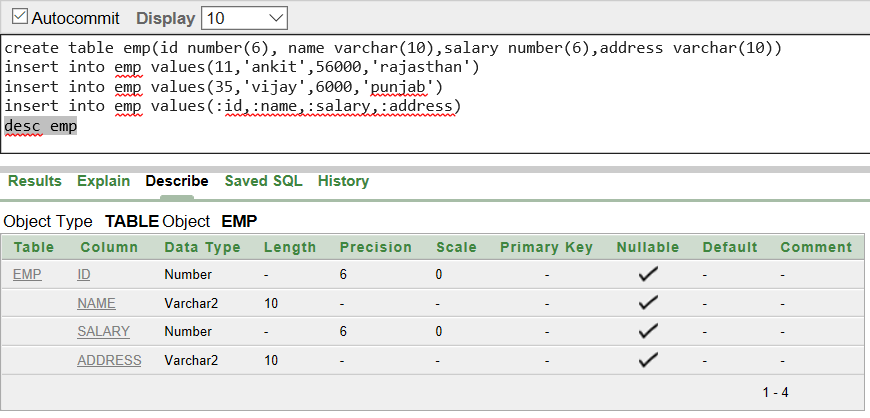
***insert into student values( :column\_name, …) ;***

****

1. **Describe the table**

We can describe the basic elements of the table i.e. column-properties, primary-key,etc.

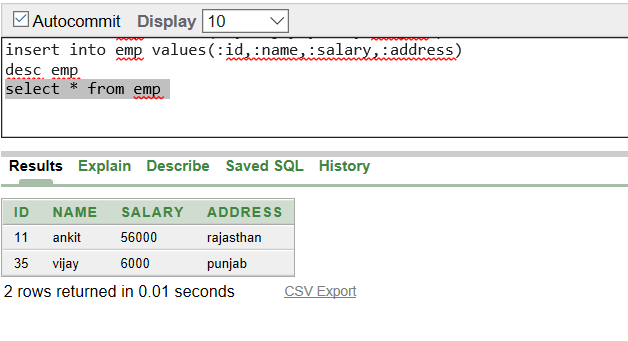
***desc table\_name;***



1. **Describe the table content/tuples.**

We can describe the tuples or, the data stored in these tables by:

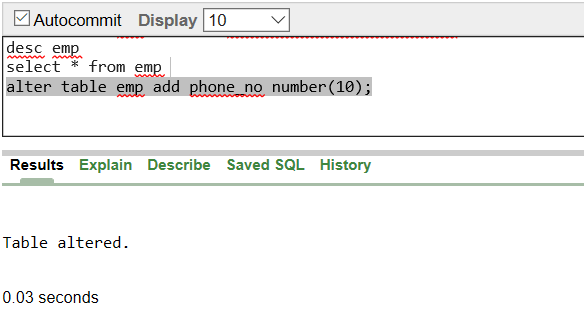
***Select \* from table\_name;***

****

1. **Add column**

Due to several modification, a table may need to add any column. We can do this by this command :

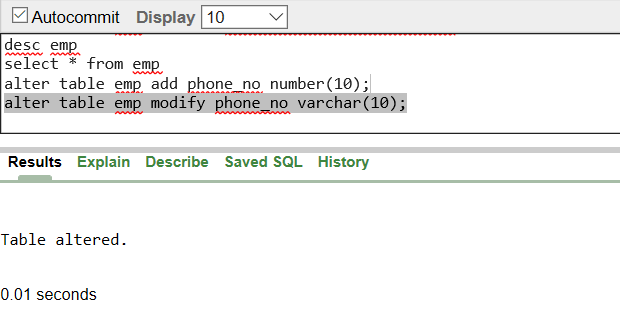
***Alter table table\_name add column\_name datatype;***

****

1. **Modify table column datatype**

The datatype of any column may need to change like from integer to floating numbers. So, this can also be done by altering the table as:

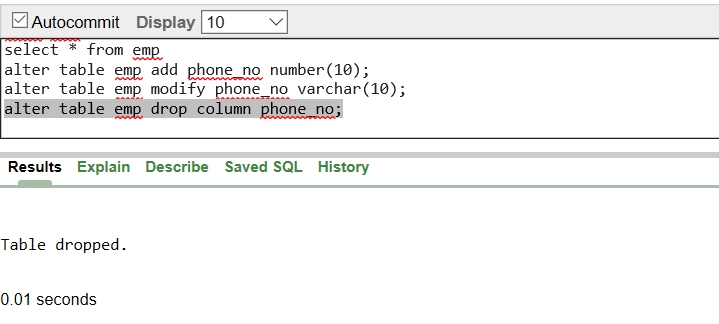
***alter table table\_name modify column\_name new\_datatype ;***

****

1. **Drop column**

Any column, that is no longer needed as the database can be dropped by altering the table as :

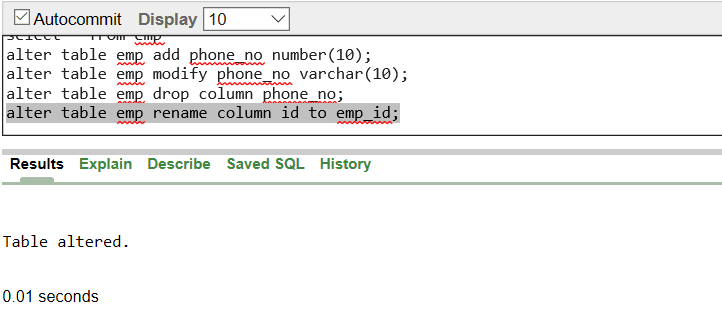
***alter table table\_name drop column column\_name;***

****

1. **Change of column name in table**

Column name can also be altered by using alter table command as :

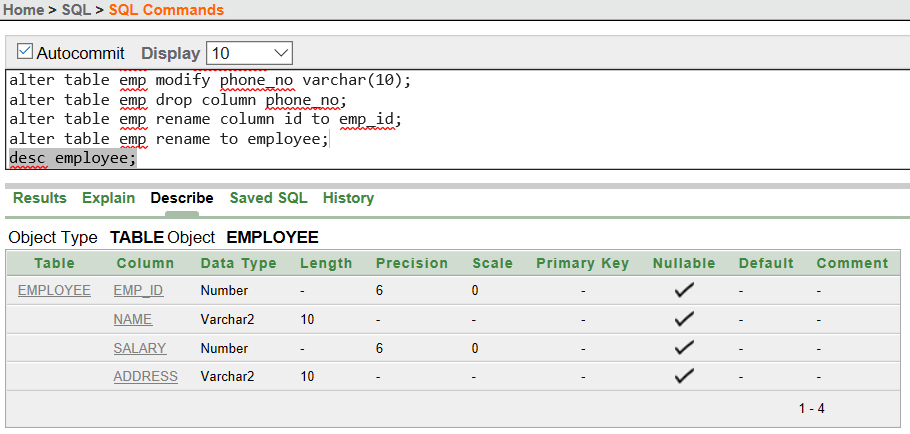
***alter table table\_name rename column column\_name to new\_column\_name ;***

****

1. **Change of table name**

Change of table\_name is also possible by altering the table as :

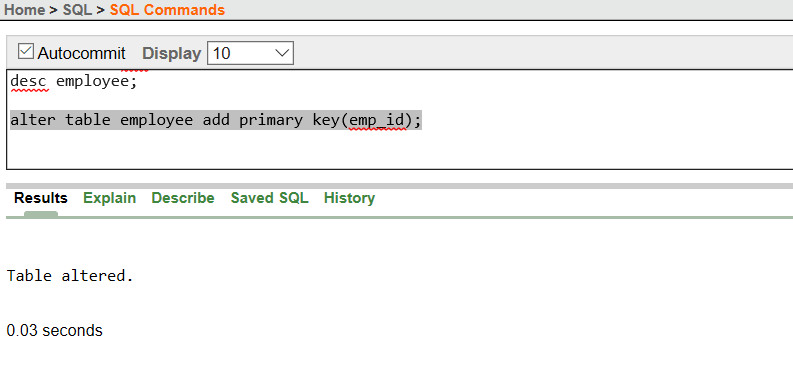
***alter table table\_name rename to new\_table\_name ;***

****

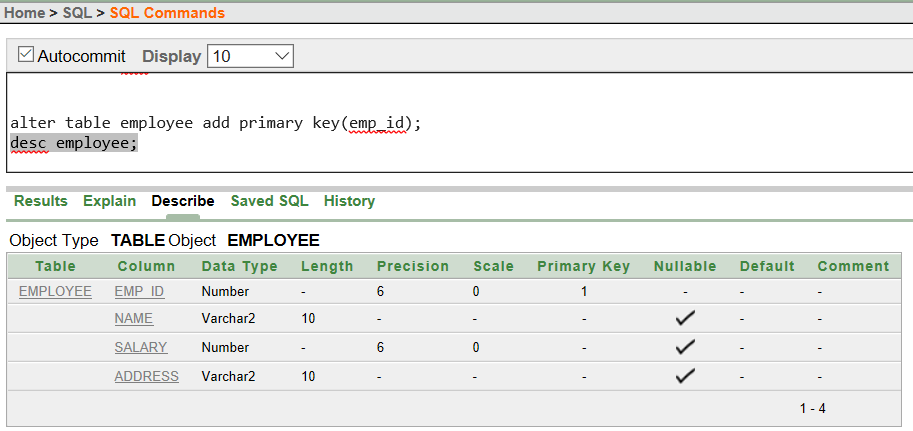
1. **Add a primary key**

A primary key is needed to uniquely identify the tuples. And hence, in the oracle, it can be done by altering the table as:

***alter table table\_name add primary key(column\_name);***

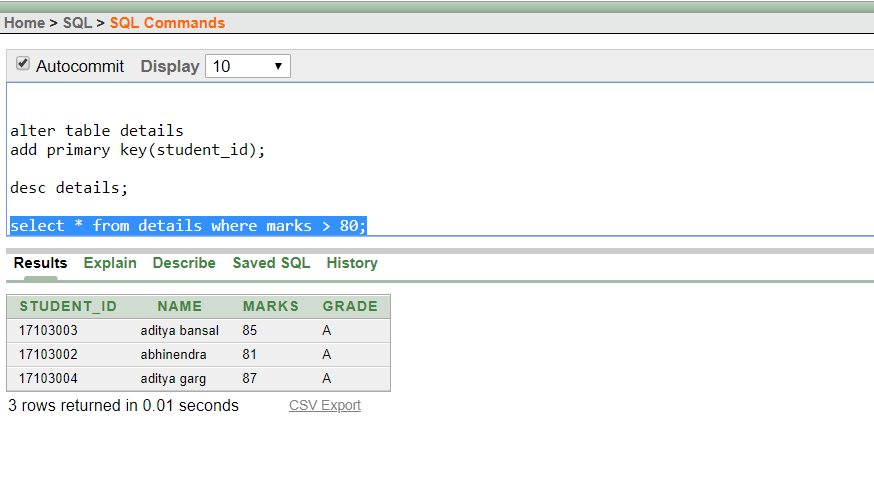
****

On describing the table, it can be seen as the primary key has turned 1 for the column the key has been enabled**.**

****

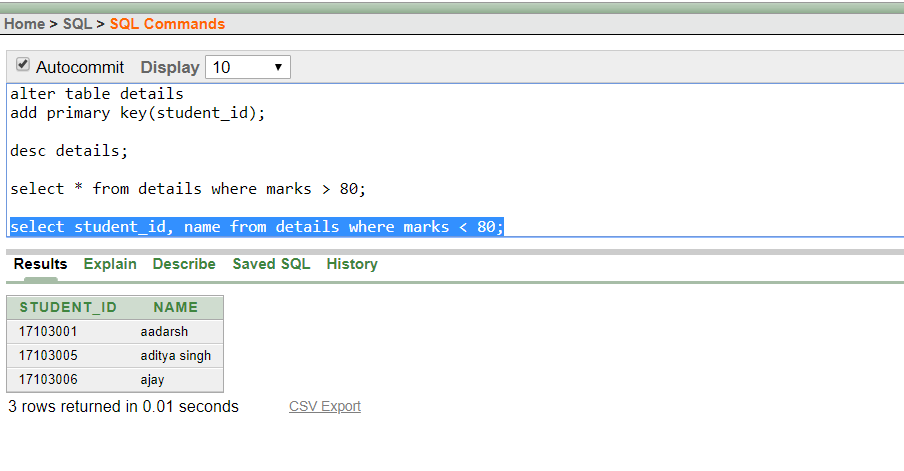
1. **Select a particular tuple (Use of where clause)**

Selecting all the tuples that manages a condition can be evaluated/expressed by where clause as :

****

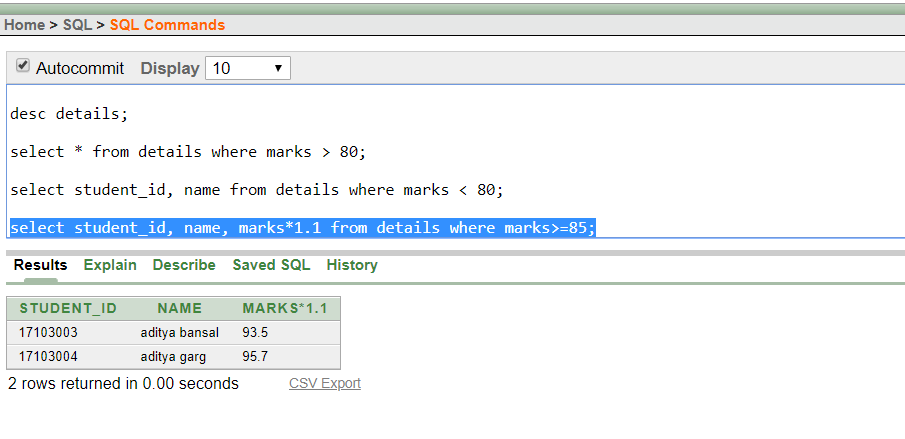
1. **Select a particular column (use of where clause)**

We can also select some unique column instead of the whole length tuple and can be shown as a list for a view perspective hiding all the irrelevant information, using select as :

****

1. **Arithmetic calculation on selection.**

Any arithmetic changes in the values can be just shown without actually performing it in the original database. It can be achieved by :

****

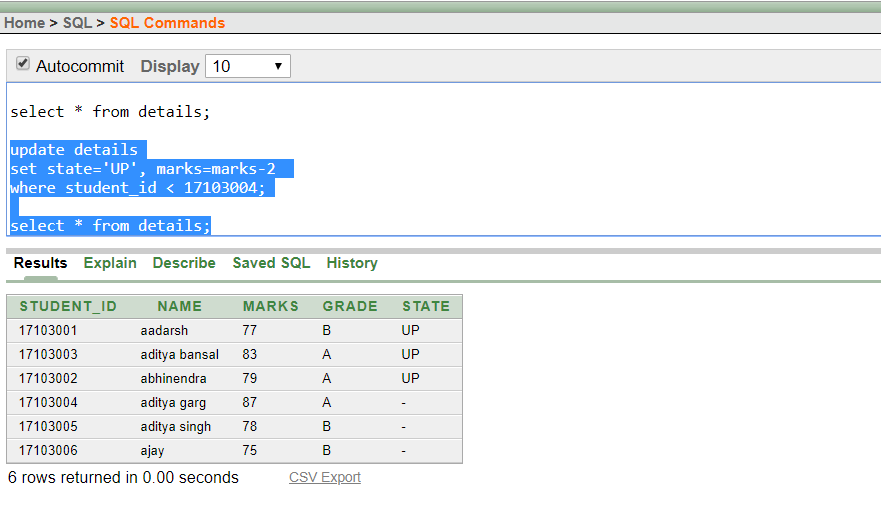
1. **Update the values (using update)**

Any value can be updated by using update and set keyword that is used as :

***Update table\_name***

***Set column\_name = “ … “***

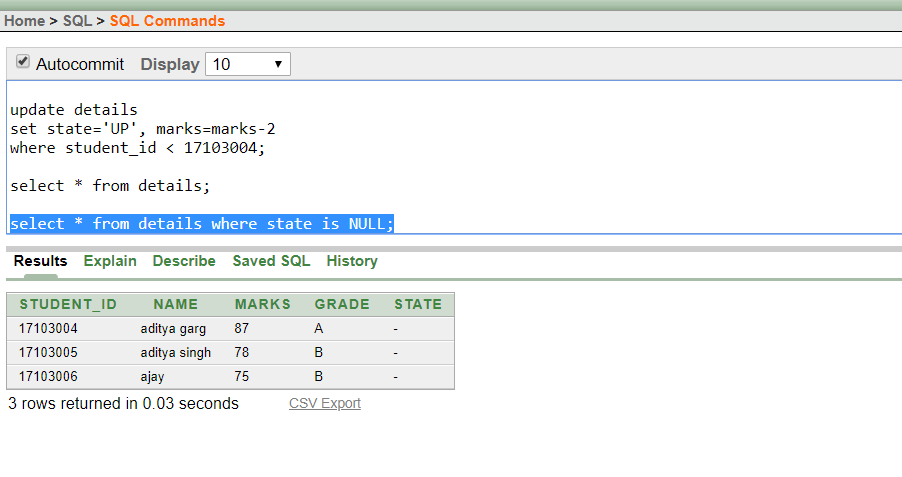
***From table\_name where “ condition “ ;***

****

1. **Select tuples having a particular column as NULL**

We can select any tuples having a particular column with a NULL value. It can be done by using is NULL clause.

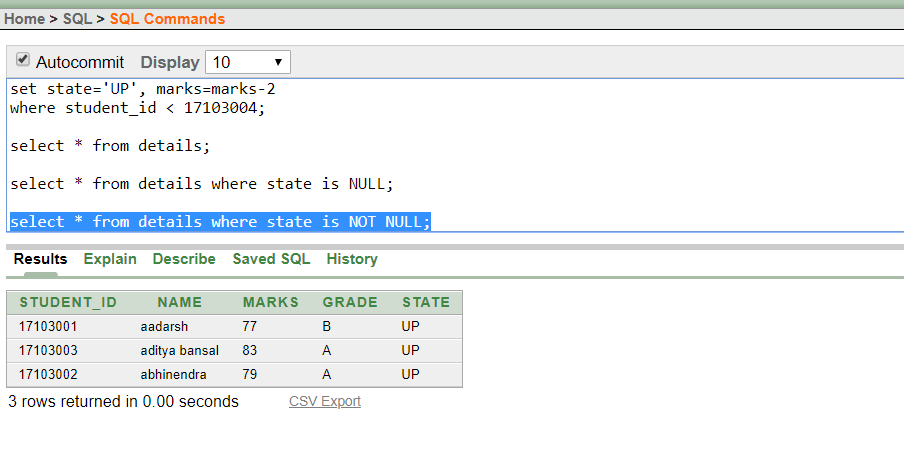
***Select \* from table\_name where column\_name is NULL;***

****

1. **Select tuples having a particular column as NOT NULL**

We can select any tuple having a particular column with the ‘is not NULL’ clause.

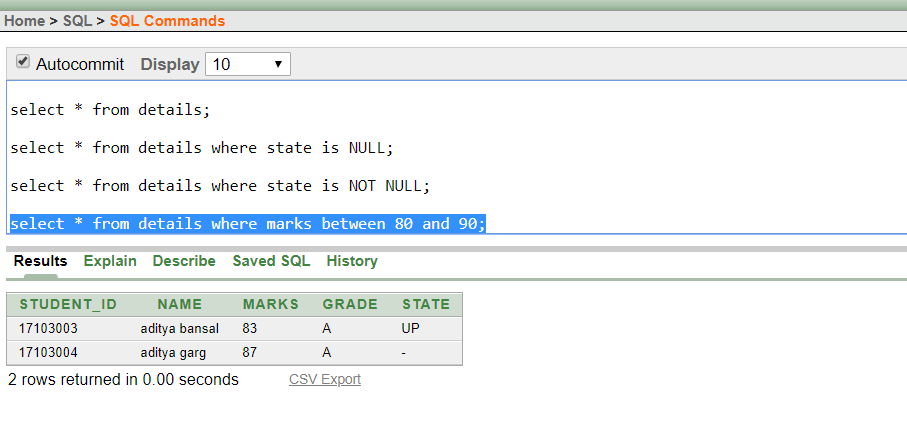
***Select \* from table\_name where column\_name is NOT NULL****;*

****

1. **Use of Between Clause**

To get the tuples having a particular value between the range, a between clause can be used which will show the values between the extremities.

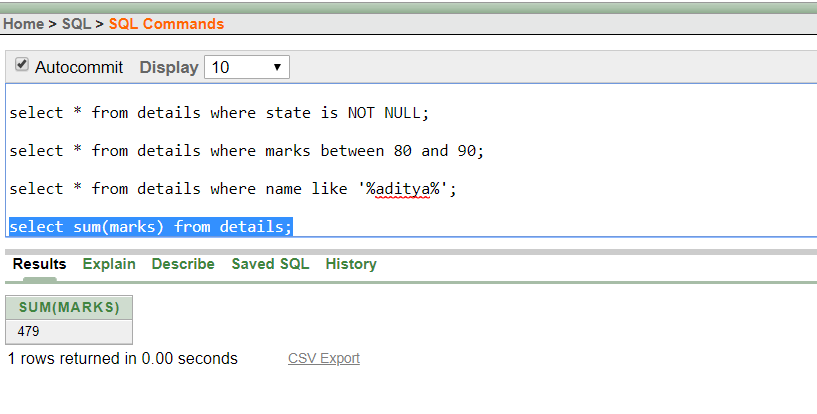
***Select \* from table\_name where column\_marks between min and max ;***

****

1. **Use of sum clause**

Sum function is used to find the sum of all the numbers in a pre-defined column.

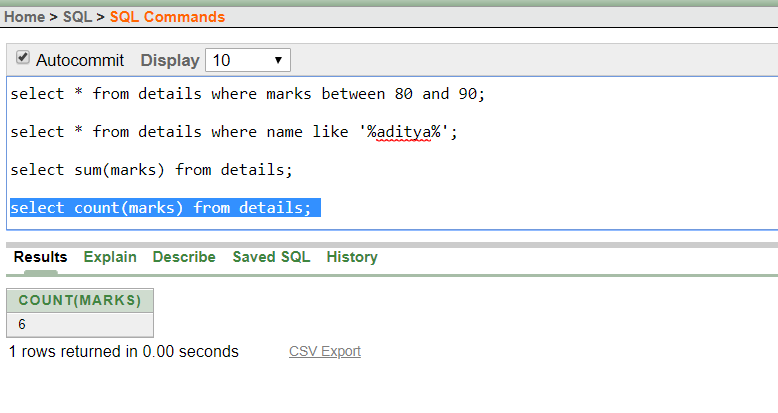
***Select sum(column\_name) from table\_name****;*

****

1. **Use of count clause**

To count the number of entries/tuples, we can use count function.

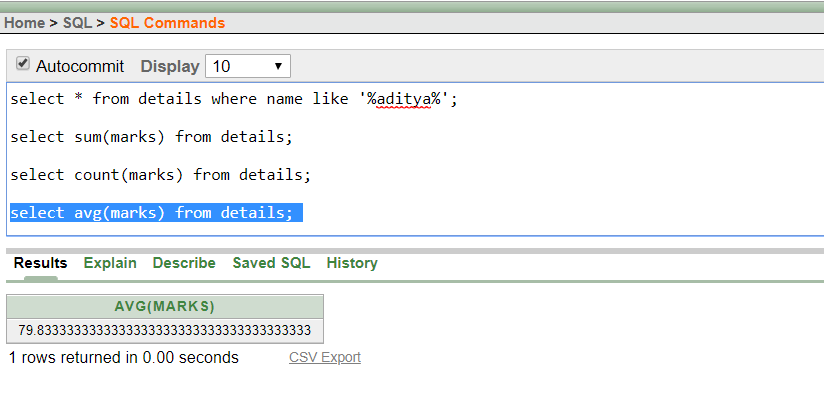
***Select count(column\_name) from table\_name;***

****

1. **Use of average clause**

The average of a particular column can be calculated as :

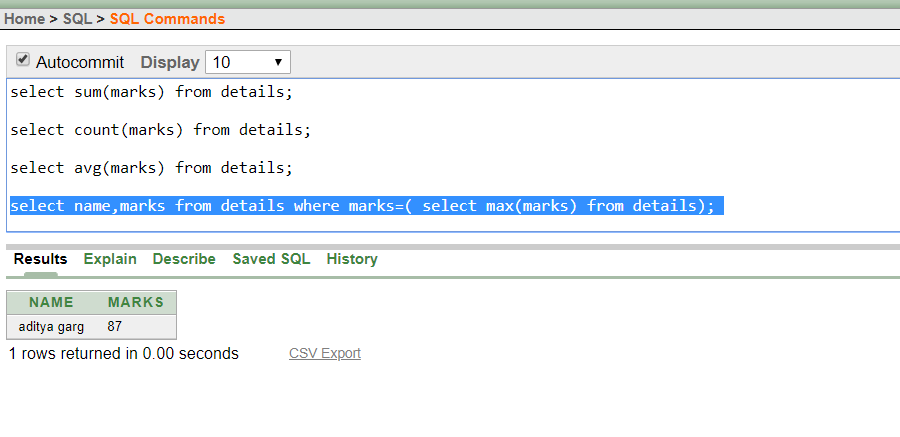
***Select avg(column\_name) from table\_name;***

****

1. **Use of Maximum clause**

The maximum of any particular number column can be given by this function as:

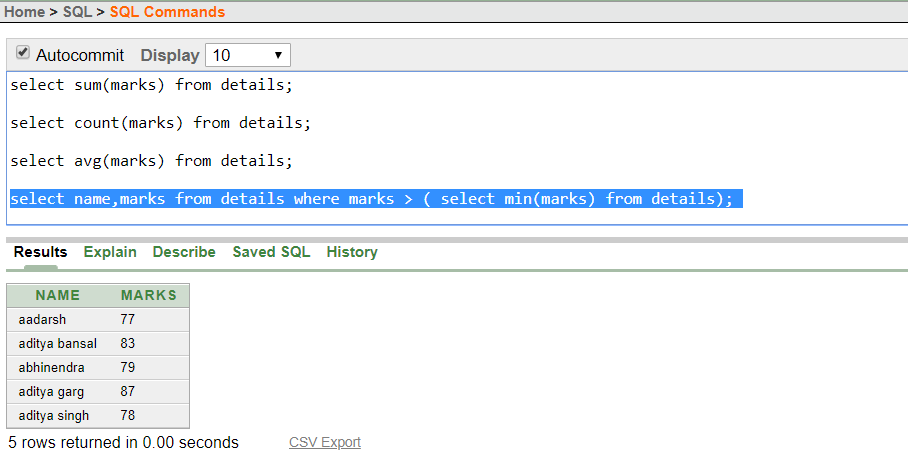
***Select max(column\_name) from table\_name;***

****

1. **Use of Minimum clause**

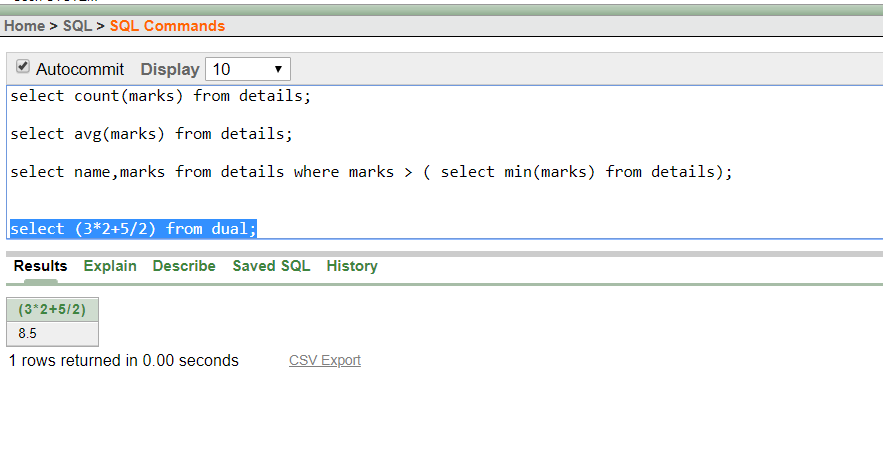
Similarly, minimum can be generated by using min clause.

***Select min(column\_name) from table\_name;***

****

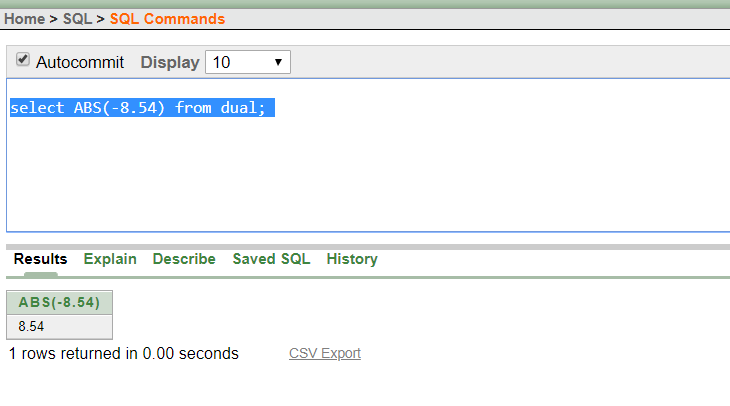
1. **Use of dual**

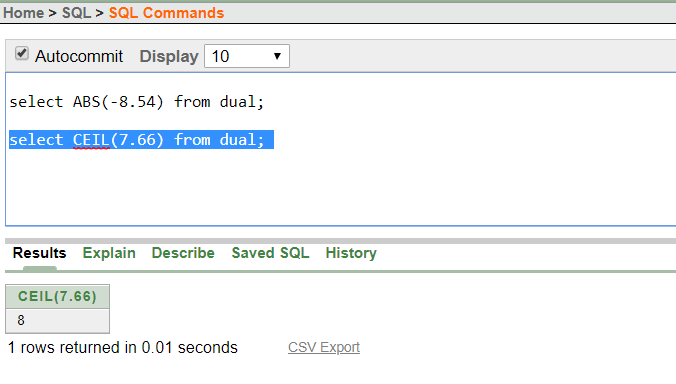
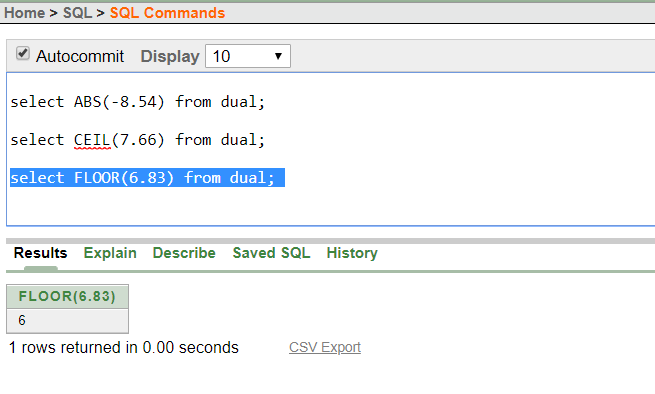
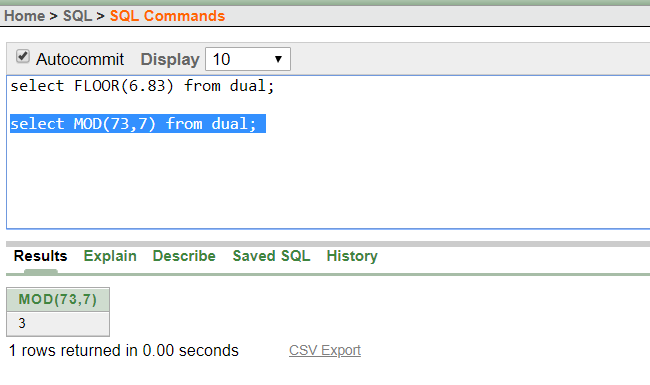
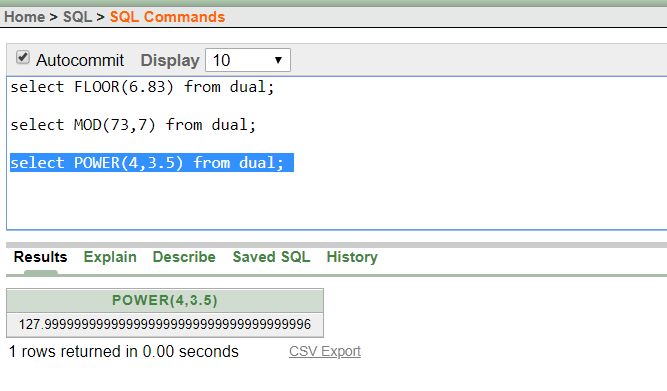
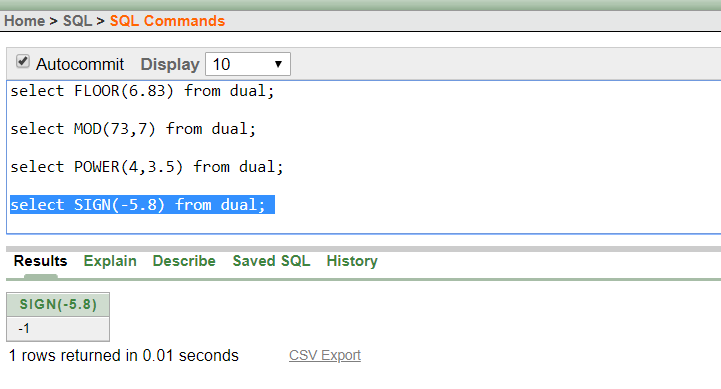
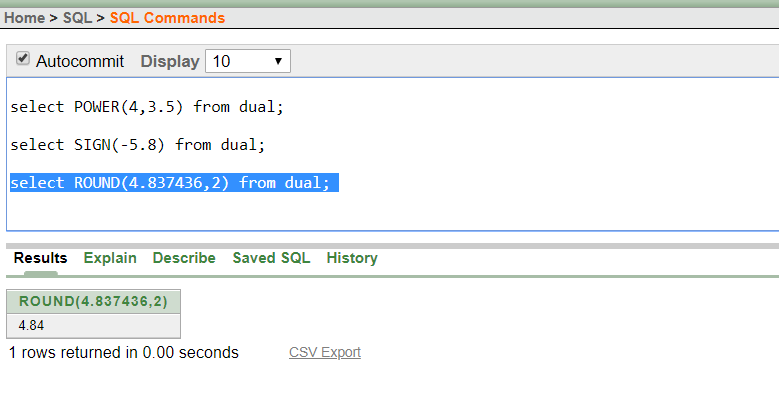
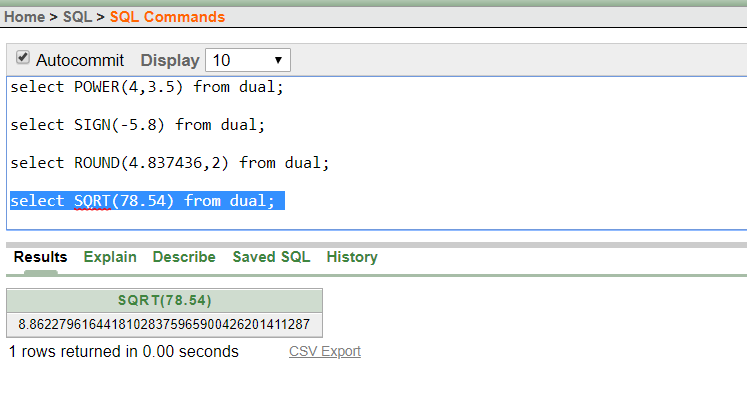
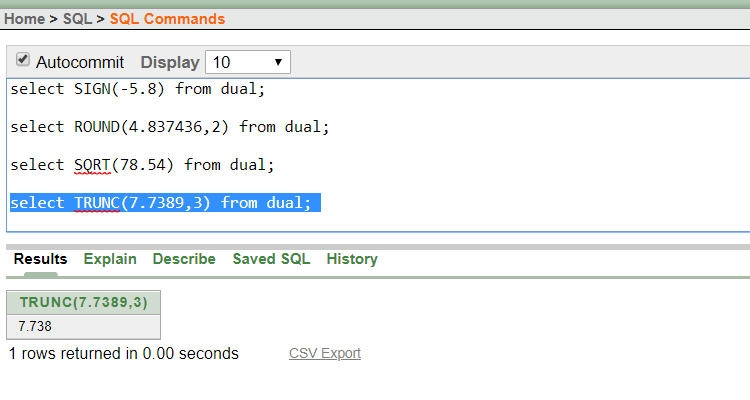
Dual is like a 1-row, 1-column table with a single record used for selecting when you’re not actually interested in the data, but instead want the result of some system function in a select statement.

****

1. **Use of number functions**

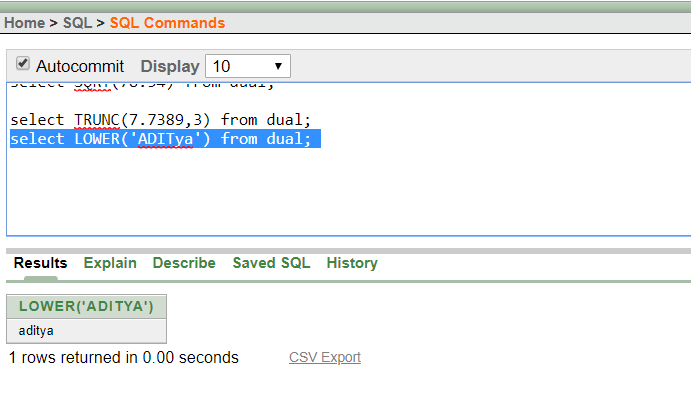
* **ABS –** Abstract function



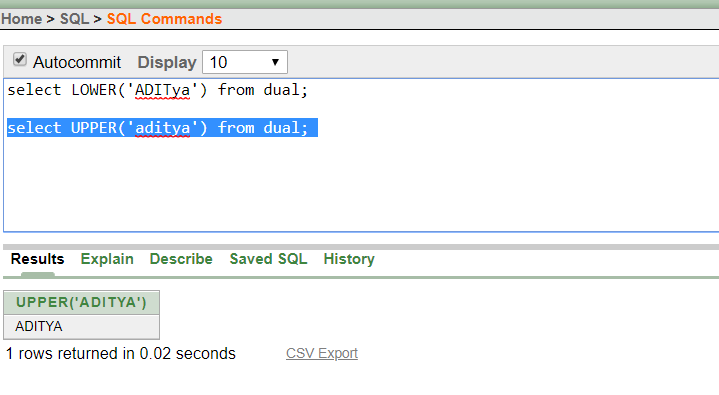
* **CEIL –** Ceiling function
* **FLOOR** – Floor function
* **MOD –** Modulus function
* **POWER –** power function
* **SIGN –** Sign function
* **ROUND –** Round function
* **SQRT –** Square Root function
* **TRUNC** – Truncating function

1. **Use of Character function**

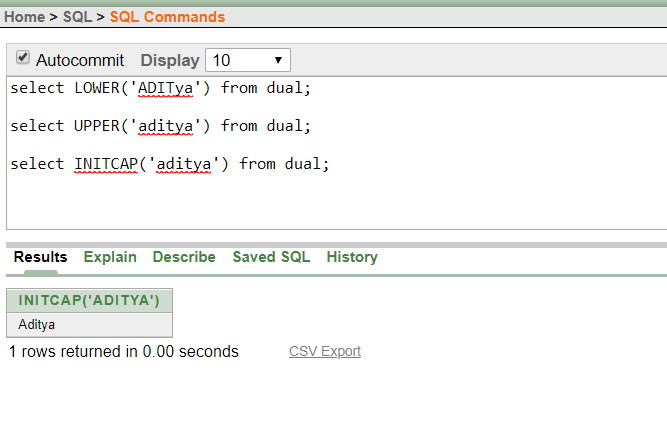
* **LOWER**



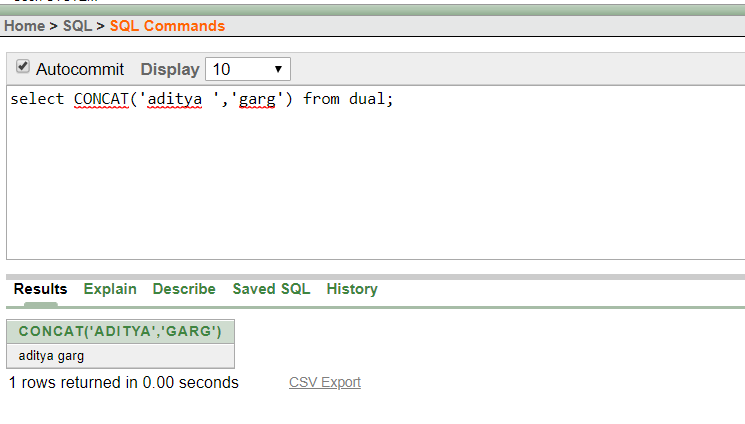
* **UPPER**



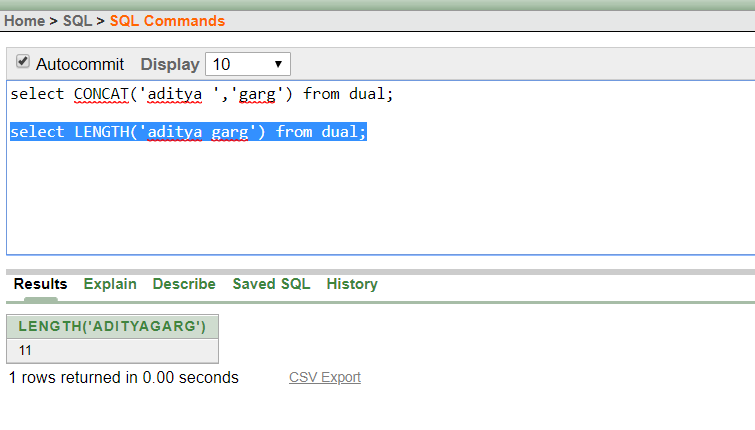
* **INITCAP**

****

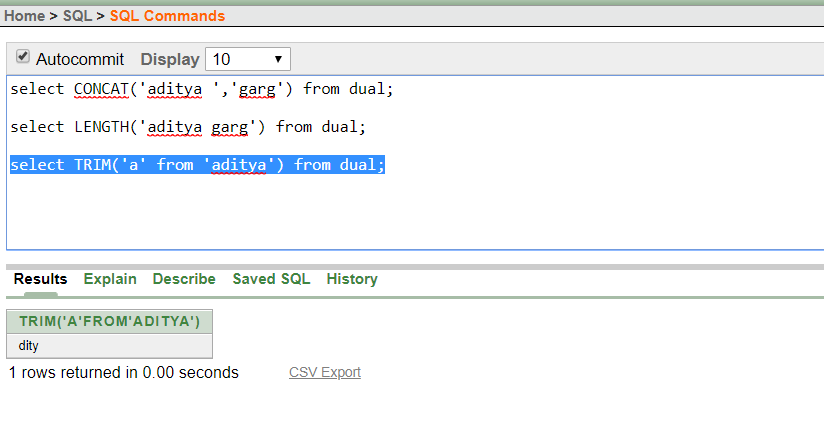
* **CONCAT**

****

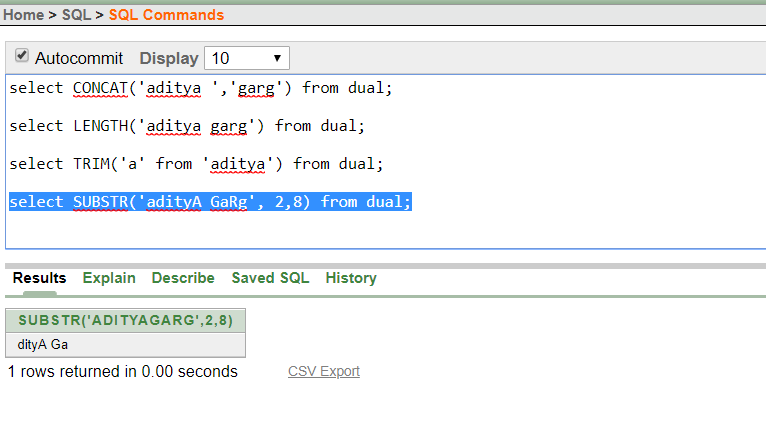
* **LENGTH**

****

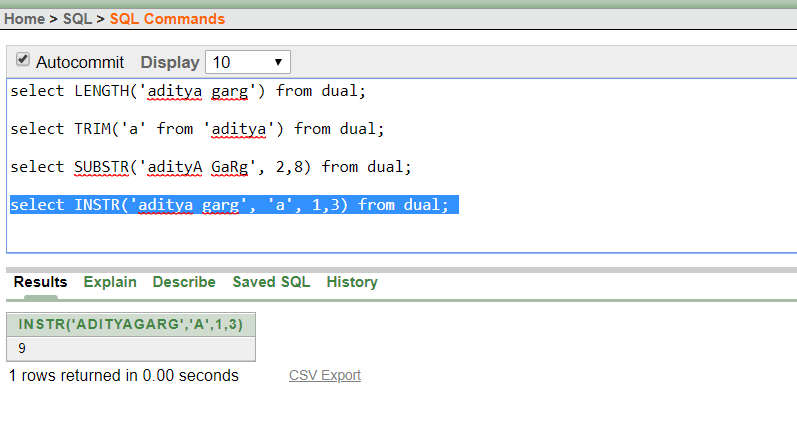
* **TRIM**

****

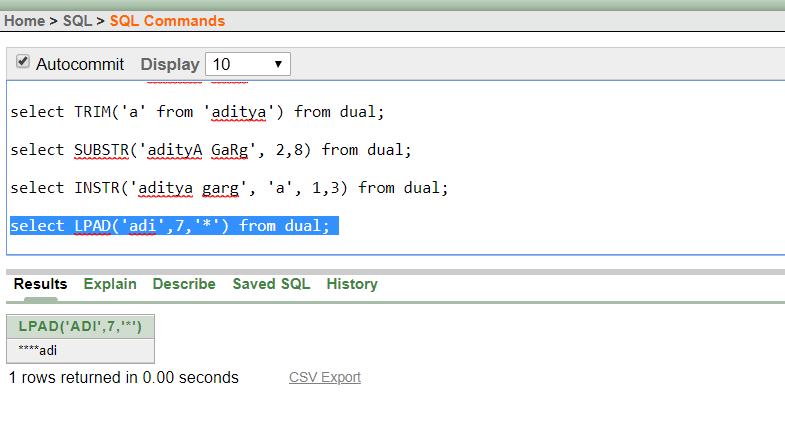
* **SUBSTR**

****

* **INSTR**

****

* **LPAD**

****

**JOINS**

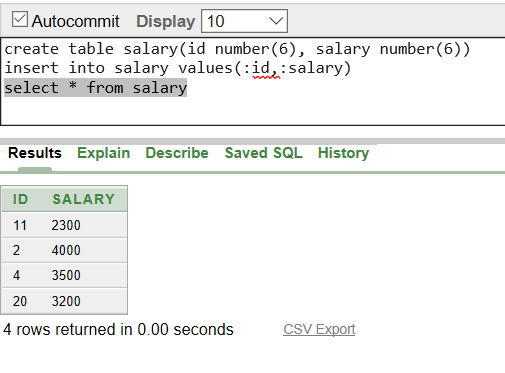
A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

There are different types of JOINS

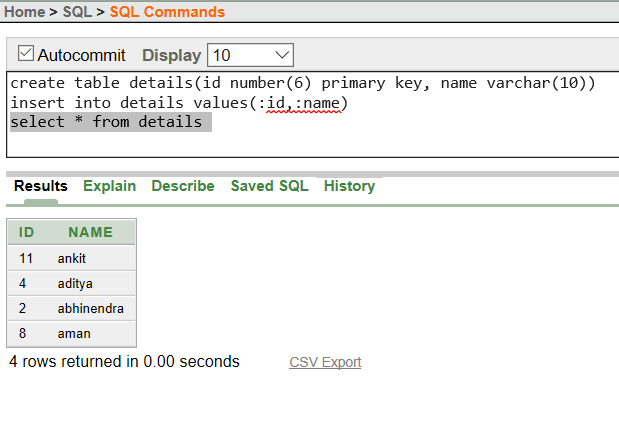
* Inner join/natural join
* Left outer join
* Right outer join
* Outer join

And for the JOIN, there will be need of 2 relations as table or query format

So, here is the tables



2nd table is

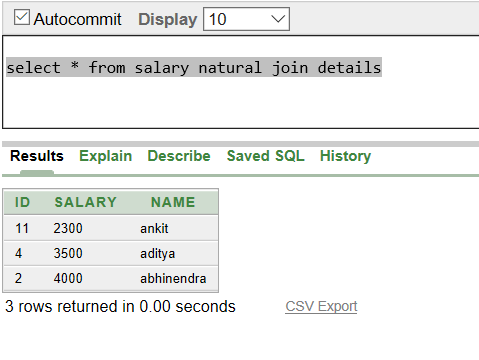


First table as SALARY describes relation between id and salary.

Second table as DETAILS describes relation between id and name.

1. **Natural join/ inner join:**

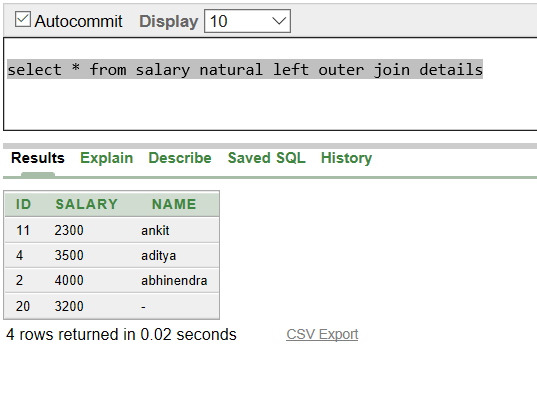
This join returns the records having common matching values.



1. **Left outer join:**

This join results in the all record of left table, with all/partial matched records of right table.

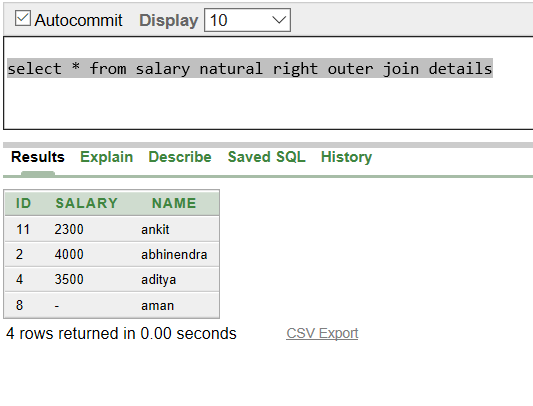
This may result some results as empty/NULL.



1. **Right outer join:**

This join results in the all record of right table, with all/partial matched records of left table.

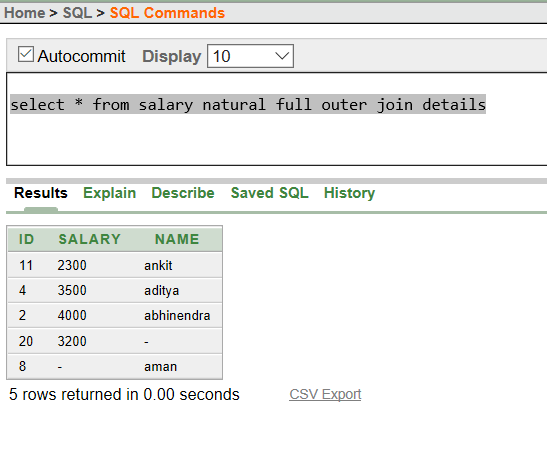
This may result some results as empty/NULL.



1. **Outer join:**

This results in each and every query to be displayed, and does not depend whether it matches with its together table.

This will be the largest of all the table resulted.



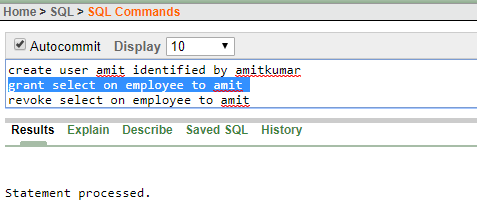
And here concludes the JOIN which can result another relation from given relations.

* **DCL (Data Control Language) commands:**

DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

Examples of DCL commands:

1. **GRANT**-gives user’s access privileges to database.



1. **REVOKE**-withdraw user’s access privileges given by using the GRANT command.

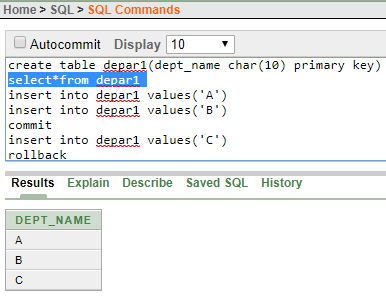
* **TCL (transaction Control Language) commands:**

TCL commands deals with the transaction within the database

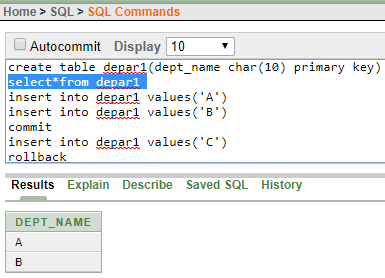
Examples of TCL commands:

1. **COMMIT**– commits a Transaction.
2. **ROLLBACK**– rollbacks a transaction in case of any error occurs.
3. **SAVEPOINT**–sets a savepoint within a transaction.

Before rollback:

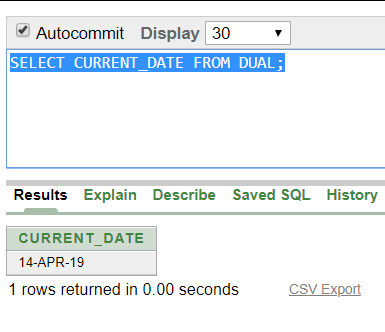
–

after rollback

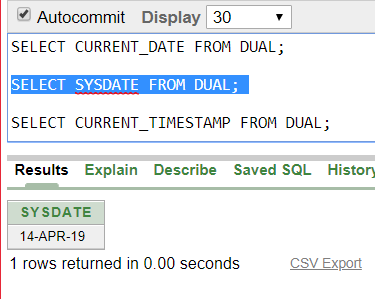


**TIME-DATE FUNCTIONS**

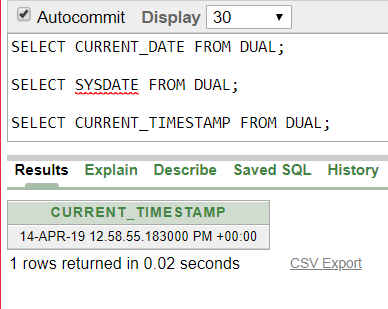
1. **CURRENT\_DATE** : Returns current date.



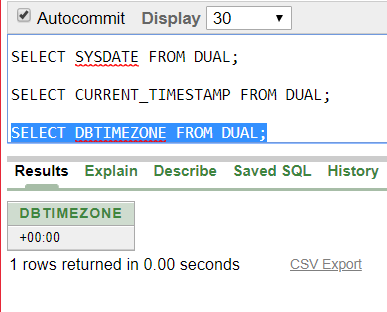
1. **SYSDATE** : Returns current date of the system.



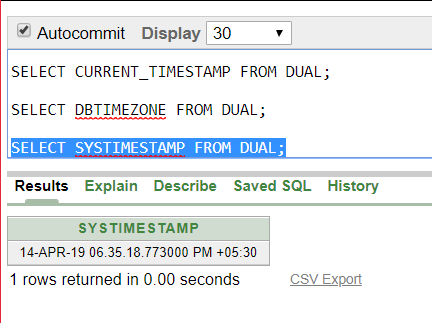
1. **CURRENT\_TIMESTAMP :** Returns current time-stamp



1. **DBTIMEZONE** : returns the current database time zone



1. **SYSTIMESTAMP** : Returns the time-stamp of the system.



**PL/SQL**

**PROGRAMMING IN SQL**

**PL/SQL** is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL. PL/SQL is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java.

PL/SQL has the following features −

* PL/SQL is tightly integrated with SQL.
* It offers extensive error checking.
* It offers numerous data types.
* It offers a variety of programming structures.
* It supports structured programming through functions and procedures.
* It supports object-oriented programming.
* It supports the development of web applications and server pages.

**SYNTAX:**

It is a block-structured language. Hence,

1. Declarations :

Starts with DECLARE. It is an optional section which defines all variables, cursors, subprograms, and all other elements to be used in the program.

1. Executable commands:

This section is enclosed between BEGIN and END and it is a mandatory section. Consists of all the executable PL/SQL statements of the program. It should have atleast one executable line of code, which may be just a NULL command to indicate that nothing has been executed.

1. Exception handling:

Starts with EXCEPTION. This optional section contains exception(s) that handle errors in the program.

**Note:** Identifiers are not case-sensitive**.**

**Data-types:**

1. Numeric
2. Character
3. Boolean
4. Date-time

Variable declaration 🡪 variable\_name datatype := DEFAULT initial-value ;

* **CONDITIONS:**

1. **IF-THEN** :

*IF condition THEN*

*S;*

*END IF;*

1. **IF-THEN-ELSE** :

*IF condition THEN*

*S1;*

*ELSE*

*S2;*

*END IF*;

* **CASE statement:**

*CASE selector*

*WHEN ‘value1’ THEN S1;*

*WHEN ‘value2’ THEN S2;*

*…*

*…*

*ELSE Sn;*

*END CASE*;

* **LOOPS:**

1. **BASIC LOOP :**

*LOOP*

*Sequence of statements;*

*END LOOP;*

1. **WHILE LOOP:**

*WHILE condition LOOP*

*Sequence of statements;*

*END LOOP;*

1. **FOR LOOP:**

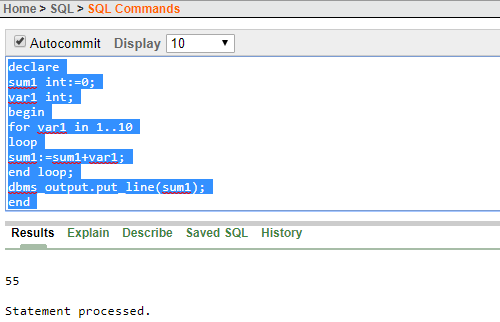
*FOR counter IN initial\_value . . final\_value LOOP*

*Sequence of statements;*

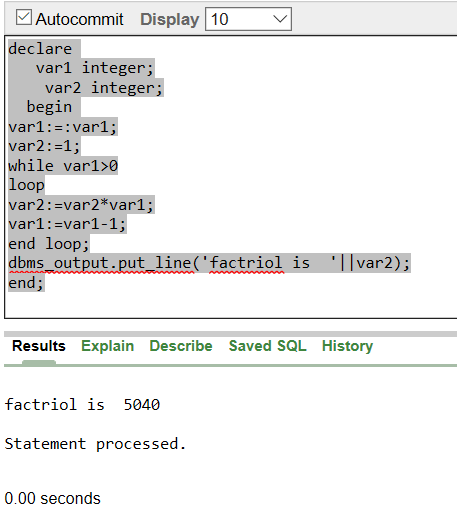
*END LOOP;*

**PROGRAMS:**

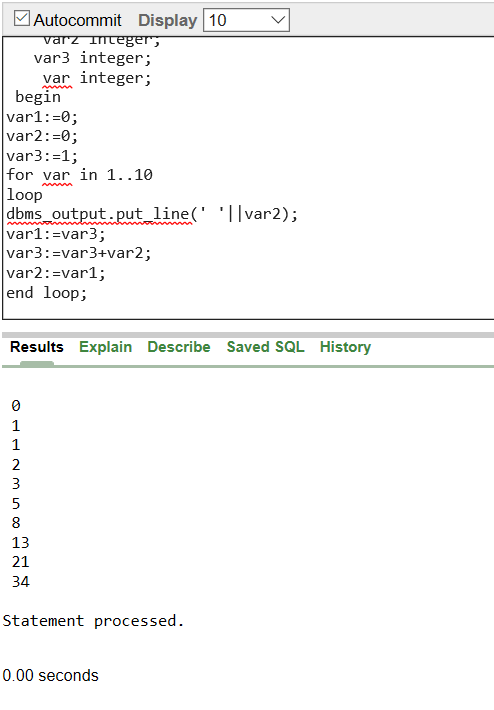
1. **SUM OF FIRST N NUMBERS**:



1. **FACTORIAL OF A NUMBER:**



1. **PRINT FIBONACCI SERIES:**

****

* **PROCEDURE and FUNCTION:**

A subprogram is a program unit/module that performs a specific task. These subprograms are combined to form larger programs. This is basically called the 'Modular design'. A subprogram can be invoked by another subprogram or program which is called the calling program.

PL/SQL provides two kind of subprograms-

1. FUNCTIONS: These returns a single value; used to compute and answer.
2. PROCEDURE: These do not return a value; used to perform a subtask.

PARTS OF SUBPROGRAM-

1. Declaration

Optional part. Contains declaration part of a subprogram. Does not start with DECLARE keyword.

1. Executable

Mandatory part and contains statements that perform the designated part.

1. Exception handling

It contains code that handle run-time errors.

SYNTAX FOR PROCEDURE:

***CREATE OR REPLACE PROCEDURE procedure\_name***

***[(parameter\_name[ IN | OUT | IN OUT] type[, , …] ) ]***

***{ IS | AS }***

***BEGIN***

***<procedure\_body>***

***END procedure\_name****;*

SYNTAX FOR FUNCTION:

***CREATE [OR REPLACE] FUNCTION function\_name***

***[(parameter\_name [ IN | OUT | IN OUT] type[, . . .])]***

***RETURN return\_datatype***

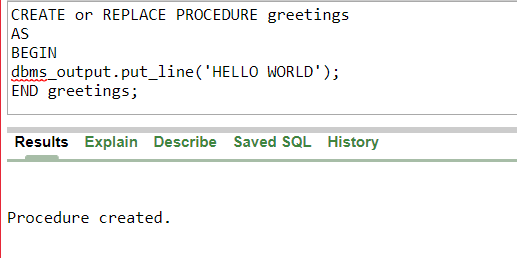
***{IS | AS}***

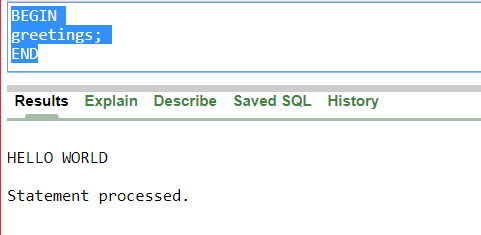
***BEGIN***

***< function\_body>***

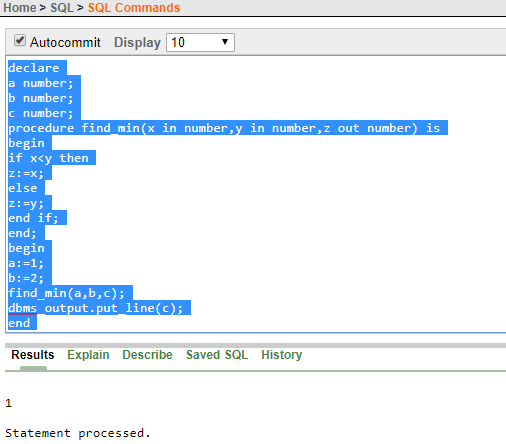
***END [ function\_name];***

**Explicit Procedure** **:**

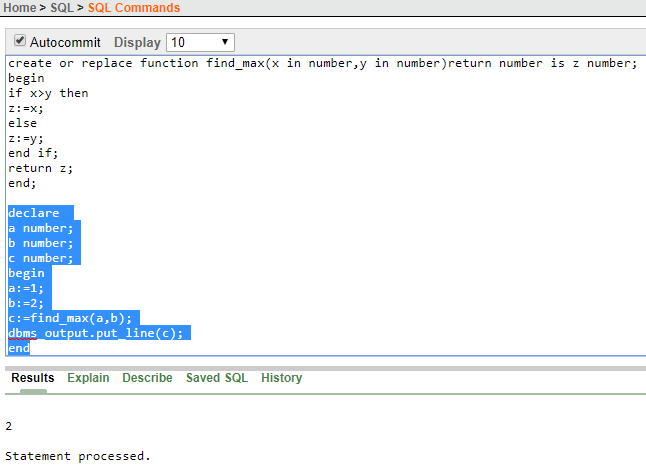




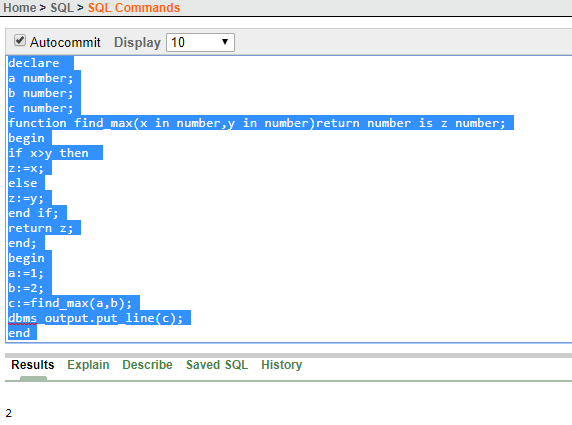
**Implicit Procedure**:



**Explicit Functions:**



**Implicit Functions:**

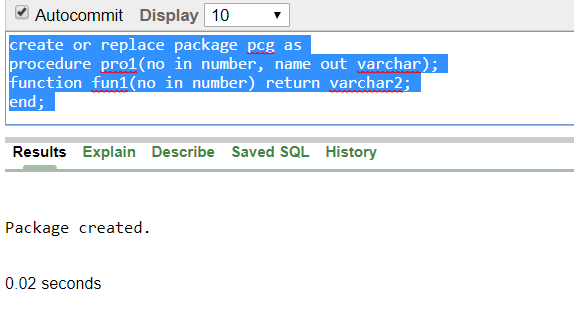


* **PACKAGES:**

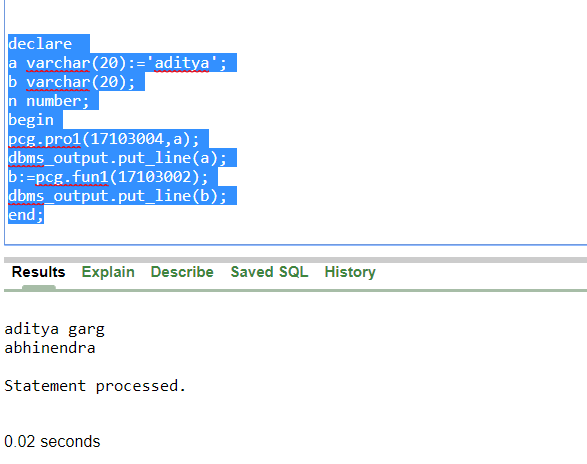
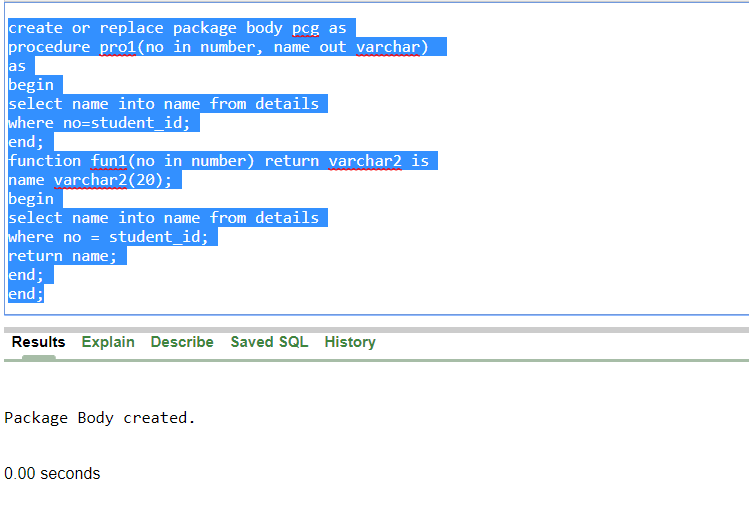
The specification is the interface to the package. It just DECLARES the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. In other words, it contains all information about the content of the package, but excludes the code for the subprograms.

The package body has the codes for various methods declared in the package specification and other private declarations, which are hidden from the code outside the package.

Creating package :



Creating package body and executing :



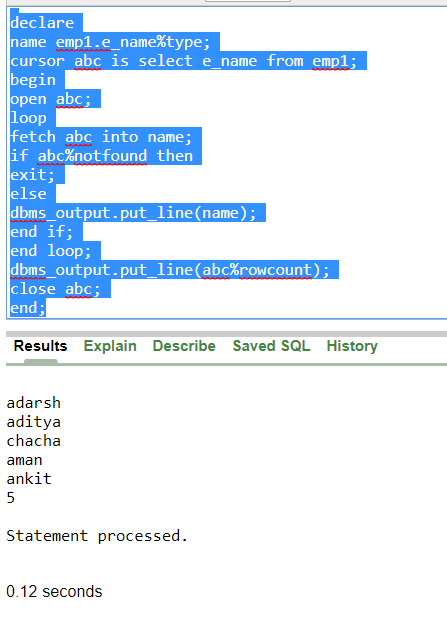
**CURSORS**

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

you can refer to the most recent implicit cursor as the SQL cursor, which always has attributes such as %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT.

* **%FOUND** : Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
* **%NOTFOUND** : The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.
* **%ISOPEN** : Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
* **%ROWCOUNT** : Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Example of a cursor :



**TRIGGERS**

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers can be defined on the table, view, schema, or database with which the event is associated.

SYNTAX:

***CREATE [OR REPLACE] TRIGGER trigger\_name***

***{BEFORE | AFTER| INSTEAD OF }***

***{INSERT [OR] | UPDATE [OR] | DELETE }***

***[OF col\_name]***

***ON table\_name***

***[REFERENCING OLD AS o NEW AS n]***

***[FOR EACH ROW]***

***WHEN (condition)***

***DECLARE***

***Declarative statements***

***BEGIN***

***Executable-statements***

***END;***

