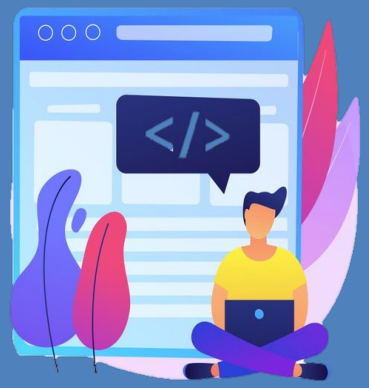
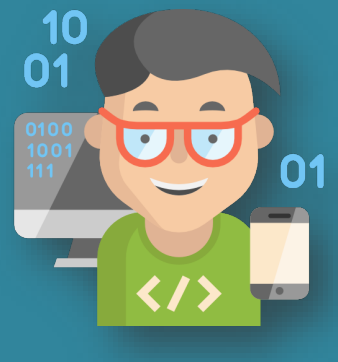




ORACLE SQL TUTORIAL

Agenda

1. Introduction
2. Creation of Table
3. Oracle setup and installation
4. Relational Model
5. Miscellaneous topics
6. Operators as keywords
7. Functions
8. Clauses
9. Nested Query
10. SQL Commands
11. Database Objects and SP

Introduction

Introduction

DATA

DATABASE

Introduction

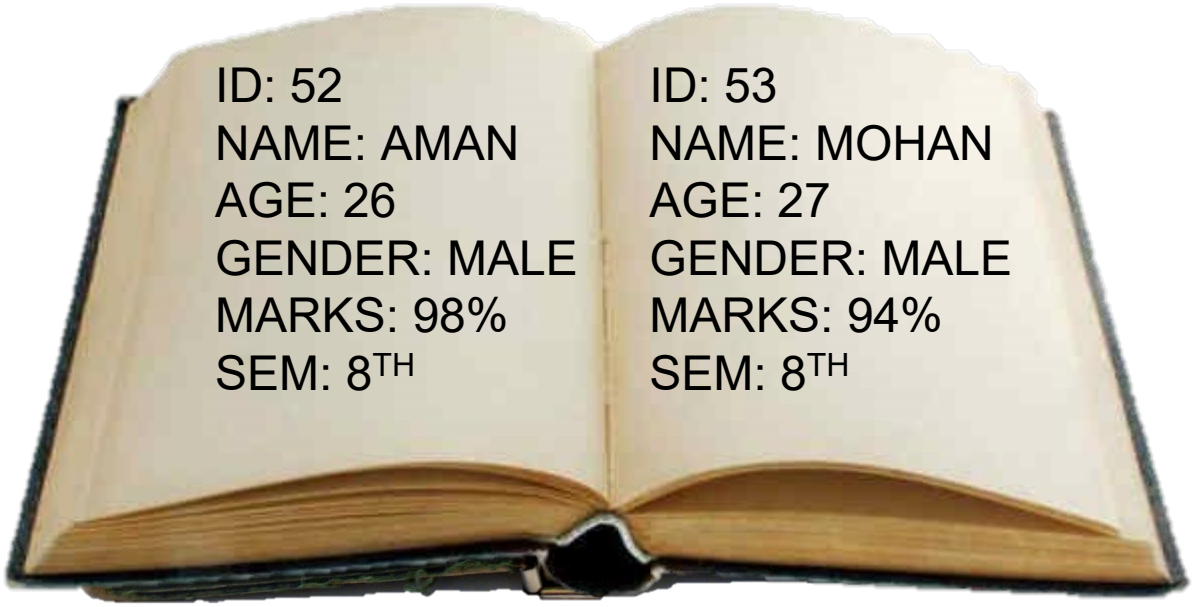
Managing refers to storing permanently and retrieving efficiently

1. How to store data(small volumes of facts) permanently and how to retrieve (get the stored facts) efficiently?

2. How to store database(large volumes of facts) permanently and how to retrieve(get the stored large volumes of facts) efficiently ?

Introduction

In the below LEDGER, one page has DATA(small volume of FACT) and all pages and many books put together forms DATABASE(large volume of FACTS)



Introduction

PROBLEMS associated in managing DATA or DATABASE in LEDGERS:-

1. Security challenges
2. Back up issues
3. Space issues
4. Human Resource issue
5. Costly
6. Inefficiency in retrieving facts

etc.

COMPUTERS came into existence in 1940s but it was very costly, so people did not think of using computers. But in late 1950s, Computer prices came low as compared to 1940s and people thought of using computers to manage facts and figures, But there were no means to do it.

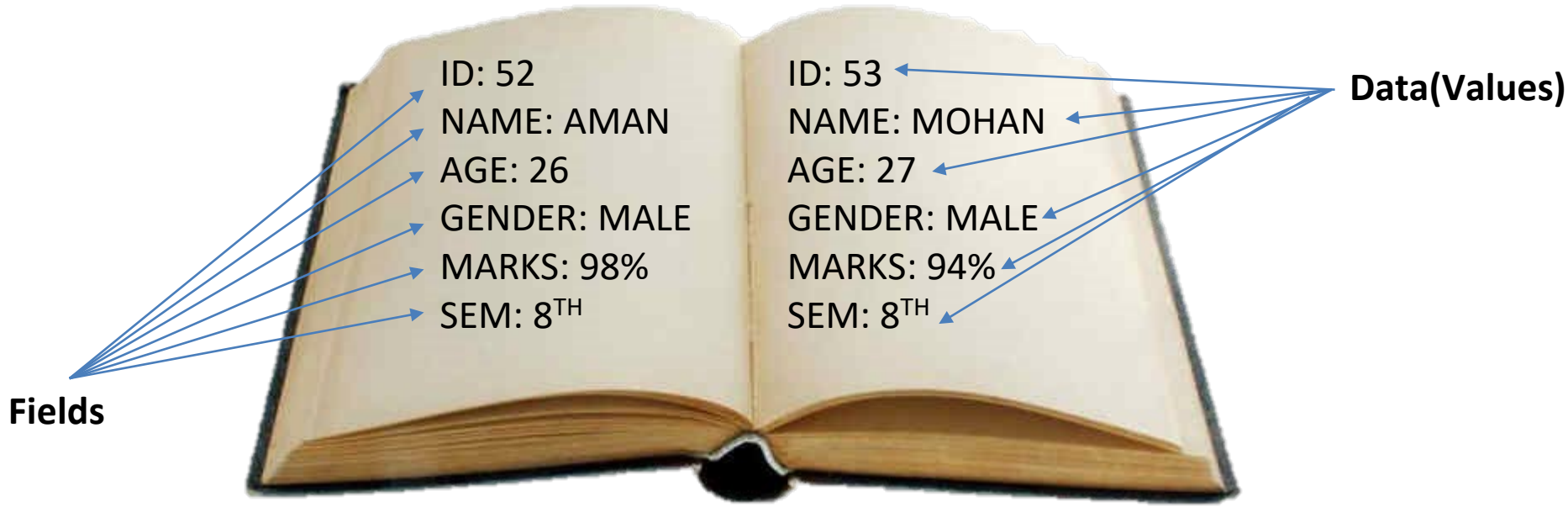
Introduction

- In 1960s, high level programming language developers suggested to make use of programming languages to manage(store and retrieve) the facts and figures.
- In programming languages there is a concept called datatypes, variables and arrays. One can make use of it and can store facts and figures.
- People were very happy because computers do not have disadvantages like LEDGERS, and now they have a way to store facts and figures in computers and they started to store facts & figures in computers using programming languages.
- But as the size of facts & figures increased, programming languages failed to manage it.
- Programming languages are successful only in managing (storing and retrieving) data (small volumes of facts and figures) but not database(large volumes of facts & figures).

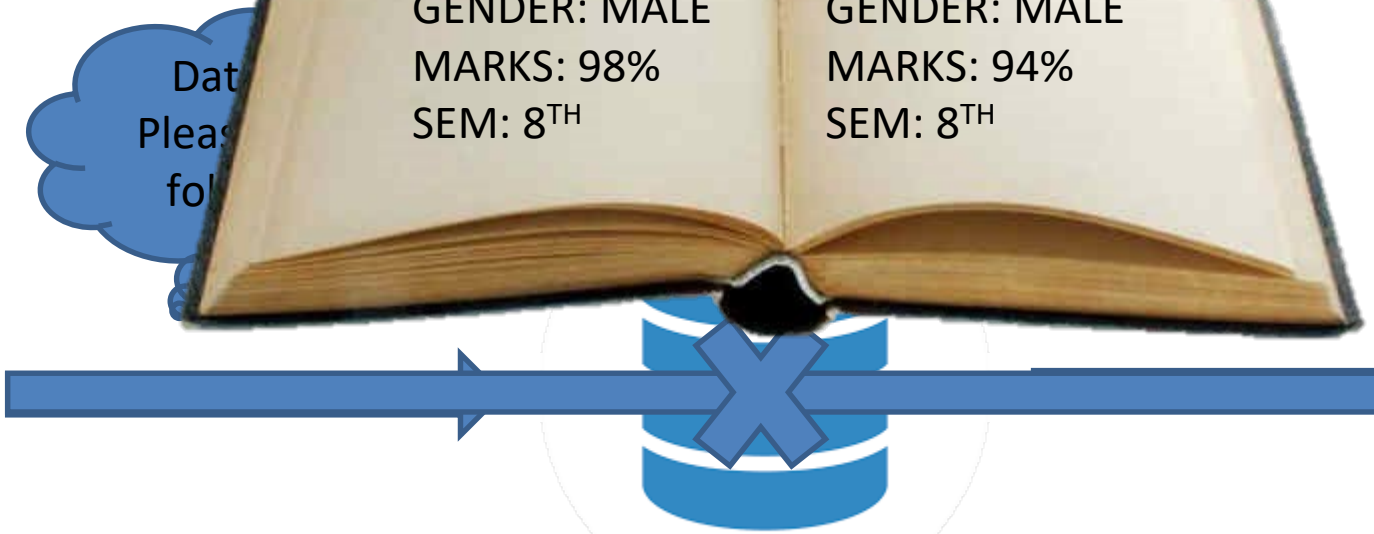
Introduction

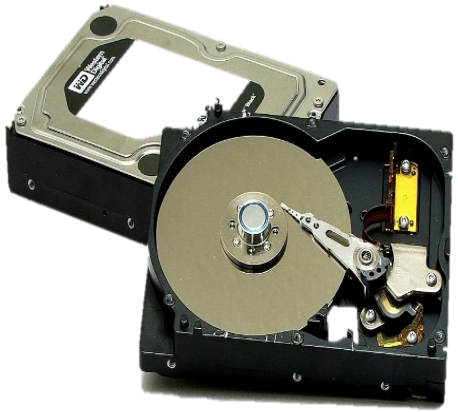
DBMS – It is a Software which is used to (manage) store data permanently and retrieve the large amount of data efficiently.

For Ex: **Oracle**, MySQL, DB2, Informix etc..



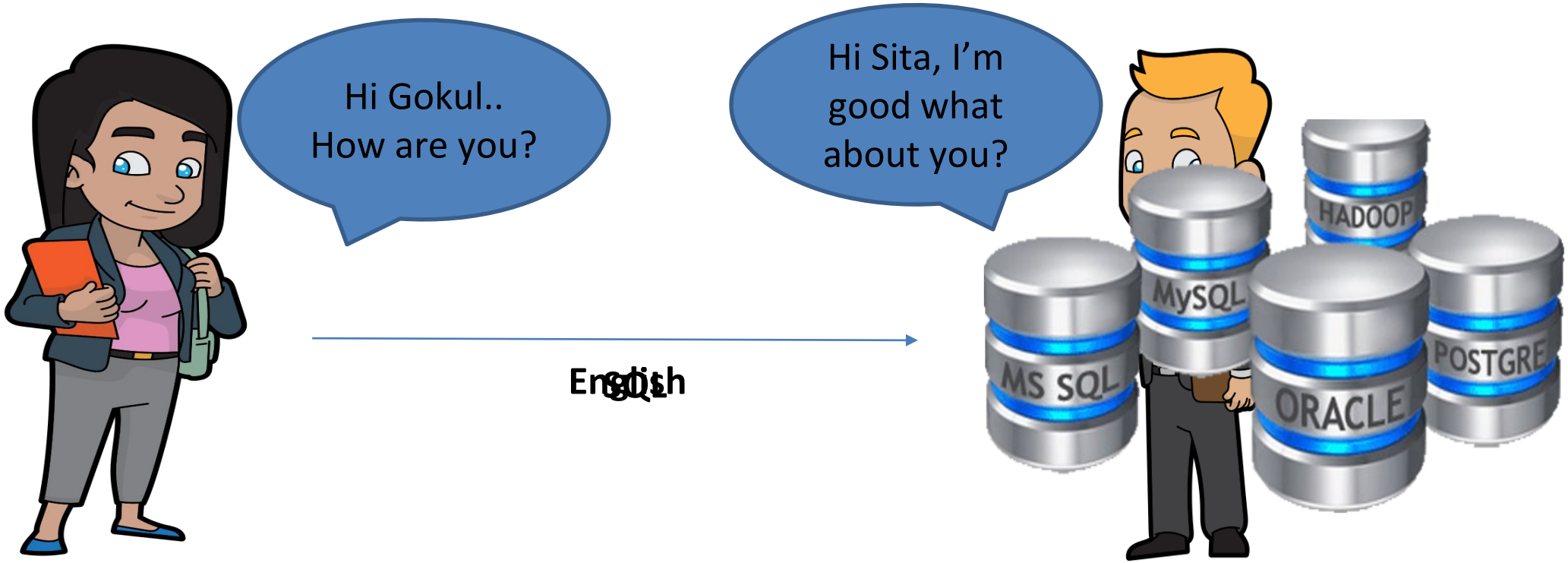
Now what you have is a Table, with column, field, table name, in which you have inserted the data, this or to store the data that is present in the le he guideline which you have to follow.

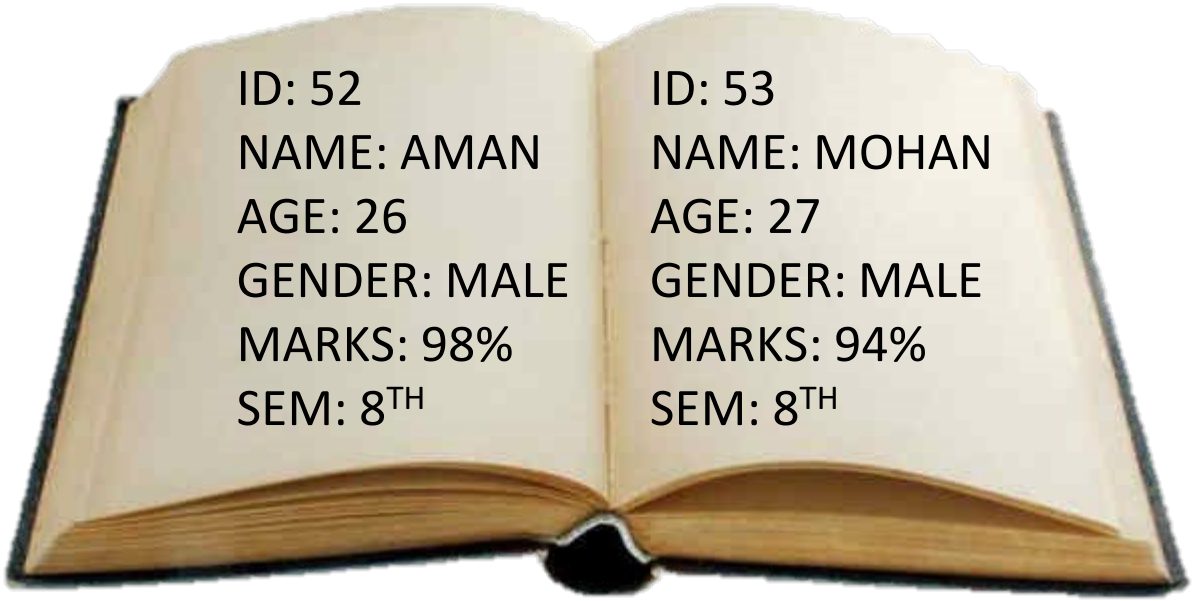




1. **Create the Table**
2. **The field of the ledger should be the columns of a Table**
3. **Give a name to the Table**
4. **Insert the data inside the Table**

SQL – It is a Language used to communicate with Database Management System.
We will be using **Oracle DBMS** to store our Data.





Using SQL

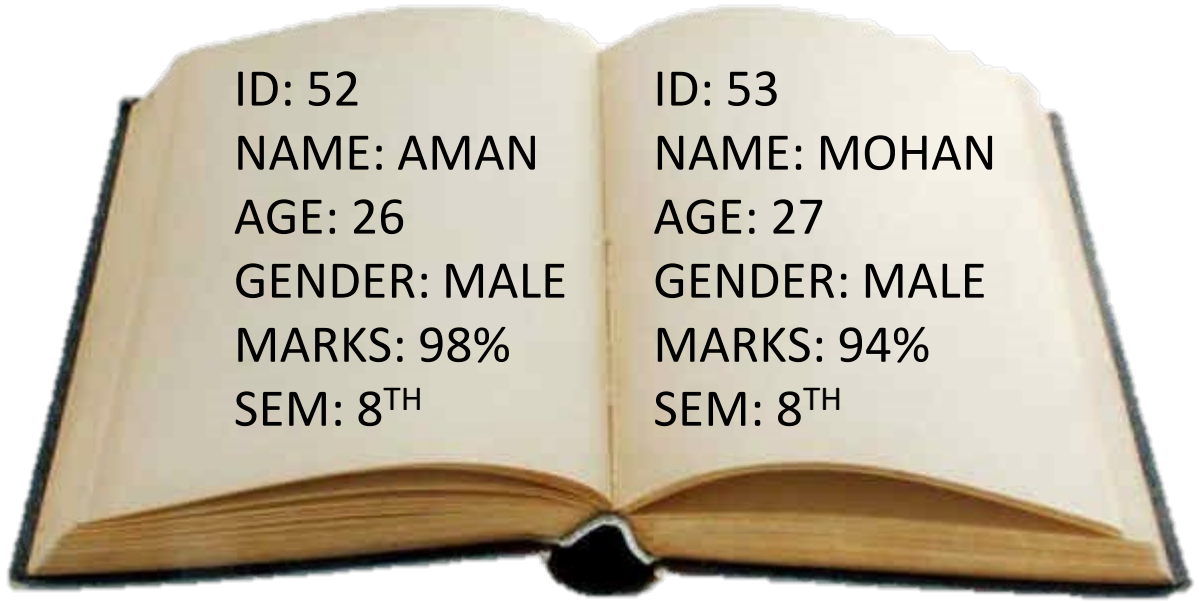
DBMS

ID	NAME	AGE	GENDER	MARKS	SEM
52	AMAN	26	MALE	98%	8TH
53	MOHAN	27	MALE	94%	8TH

Creation of Table

Creation of Table

1. How to insert data into the DBMS by creating the table




```
CREATE TABLE Student
(
ID number(10),
Name varchar2(30),
Age number,
Gender varchar2(6),
Marks number,
SEM number
);
```

ID	NAME	AGE	GENDER	MARKS	SEM

Creation of Table


```
CREATE TABLE Employee
(
EmployeeID number(10),
Name varchar2(30),
Age number,
Address varchar2(30),
Salary number,
Department number,
Experience number
);
```

EmployeeID	Name	Age	Address	Salary	Department	Experience

Creation of Table

2. Now, Let's see how to Insert the data into the table

```
INSERT INTO Student VALUES( 52, 'AMAN', 26, 'MALE',98, 8 );

INSERT INTO Student VALUES( 53, 'MOHAN', 27, 'MALE', 94, 8 );
```

ID	NAME	AGE	GENDER	MARKS	SEM
52	AMAN	26	MALE	98%	8TH
53	MOHAN	27	MALE	94%	8TH

Creation of Table

```
INSERT INTO STUDENTS (ID, NAME, AGE, GENDER, MARKS, SEM) VALUES (54, ABHIRAM, 22, MALE, 85, 7);
INSERT INTO STUDENTS (ID, NAME, AGE, GENDER, MARKS, SEM) VALUES (55, ALKA, 20, FEMALE, 78, 7);
INSERT INTO STUDENTS (ID, NAME, AGE, GENDER, MARKS, SEM) VALUES (56, DISHA, 21, FEMALE, 91, 8);
INSERT INTO STUDENTS (ID, NAME, AGE, GENDER, MARKS, SEM) VALUES (57, ESHA, 21, FEMALE, 89, 8);
```

ID	NAME	AGE	GENDER	MARKS	SEM
54	ABHIRAM	22	MALE	85	7
55	ALKA	20	FEMALE	78	7
56	DISHA	21	FEMALE	91	8
57	ESHA	21	FEMALE	89	8

Oracle setup and Installation

Relational Model

Relational Model

Roll Number		Student Name			Marks Obtained			
1	1	Rohit	Rohit		1	45	45	
2	2	Mohan	Mohan		2	78	78	
3	3	Sanjay	Sanjay		3	89	89	
4	4	Kiran	Kiran		4	56	56	

Relational Model

Country	Capital
Greece	Athens
Italy	Rome
USA	Washington
China	Beijing
Japan	Tokyo
Australia	Sydney
France	Paris

Country	Currency
Greece	Drachma
Italy	Lira
USA	Dollar
China	Renminbi (Yuan)
Japan	Yen
Australia	Australian Dollar
France	Francs

Relational Model

Term	Meaning
Relation	A table
Tuple	A row or a record in a relation
Attribute	A field or a column in a relation
Cardinality of a relation	The number of tuples in a relation
Degree of a relation	The number of attributes in a relation
Domain of an attribute	The set of all values that can be taken by the attribute
Primary Key of a relation	An attribute or a combination of attributes that uniquely defines each tuple in a relation
Foreign Key	An attribute or a combination of attributes in one relation R1 that indicates the relationship of R1 with another relation R2. The foreign key attributes in R1 must contain values matching with those of the values in R2

Miscellaneous topics

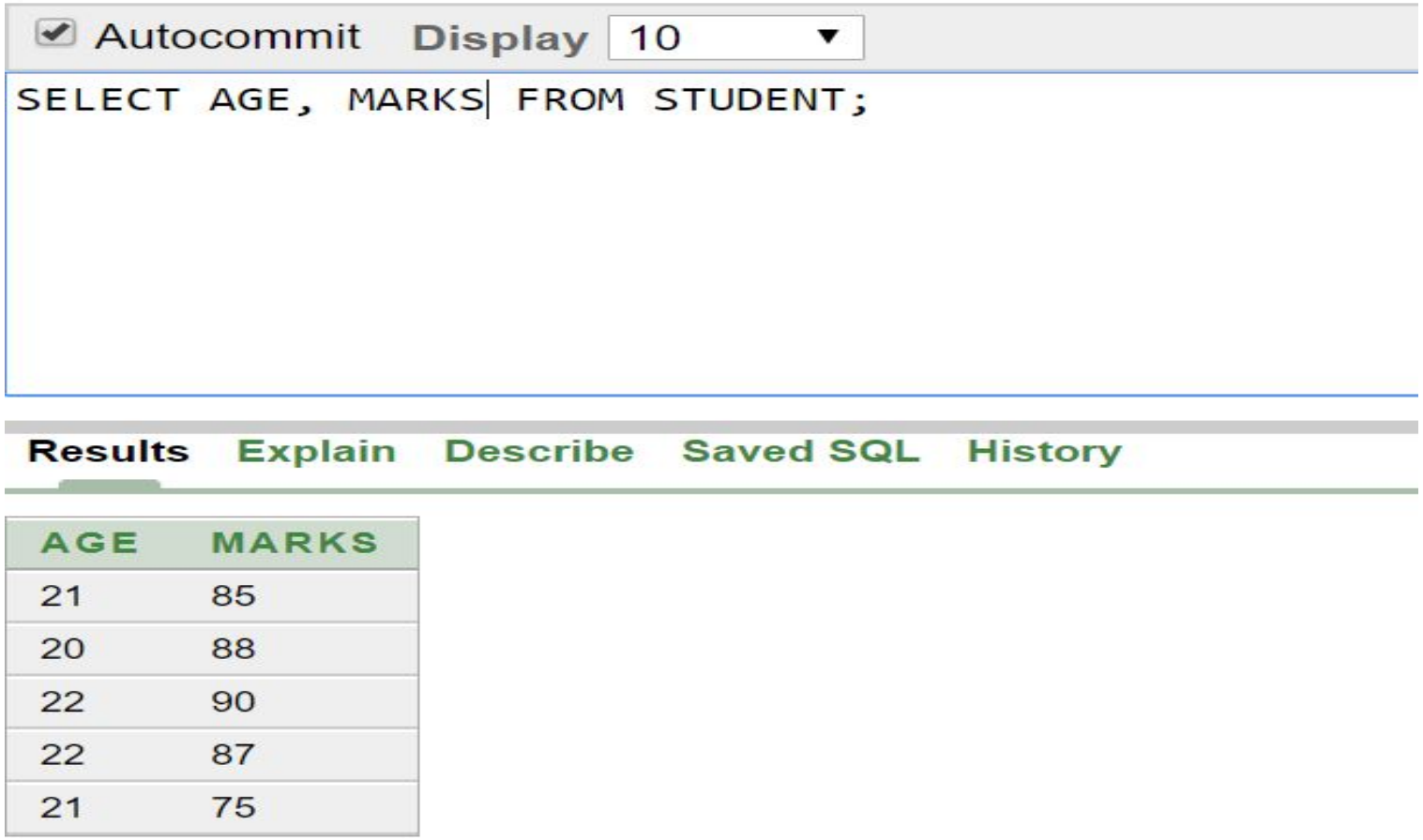
Projection

Projection means choosing which columns (or expressions) the query shall return.
Projection is a process of displaying the result using project queries.
Project queries are all those queries which is used to display(Project) particular columns to the user based on the requirement from the user.

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75

Projection

Q1. Write a query to display AGE and MARKS of all the students.



AGE	MARKS
21	85
20	88
22	90
22	87
21	75

Projection

Q2. Write a query to display NAME and USN of the student whose age is greater than 21.

Autocommit Display 10

```
SELECT NAME, USN FROM STUDENT WHERE AGE>21;
```

Results Explain Describe Saved SQL History

NAME	USN
KAVITHA	122
ARJUN	123

Selection

Selection means which rows are to be returned.
Selection is the process of displaying the result using select queries.
Select queries are those queries which displays particular rows to the user base on specific condition/s.

Q3. Write a query to display all the information of student named “KAVITHA”.

Autocommit Display 10

```
SELECT * FROM STUDENT WHERE NAME='KAVITHA';
```

Results Explain Describe Saved SQL History

USN	NAME	GENDER	AGE	MARKS
122	KAVITHA	FEMALE	22	90

Selection

Q4. Write a query to display NAME, GENDER and USN of student named “RAVI”.

Autocommit Display 10

```
SELECT NAME, GENDER, USN FROM STUDENT WHERE NAME='RAVI';
```

Results Explain Describe Saved SQL History

NAME	GENDER	USN
RAVI	MALE	121

Case Sensitivity in SQL

The **SQL** Keywords are **case-insensitive** (SELECT, FROM, WHERE, AS, ORDER BY, HAVING, GROUP BY, etc), but are usually written in all capitals.

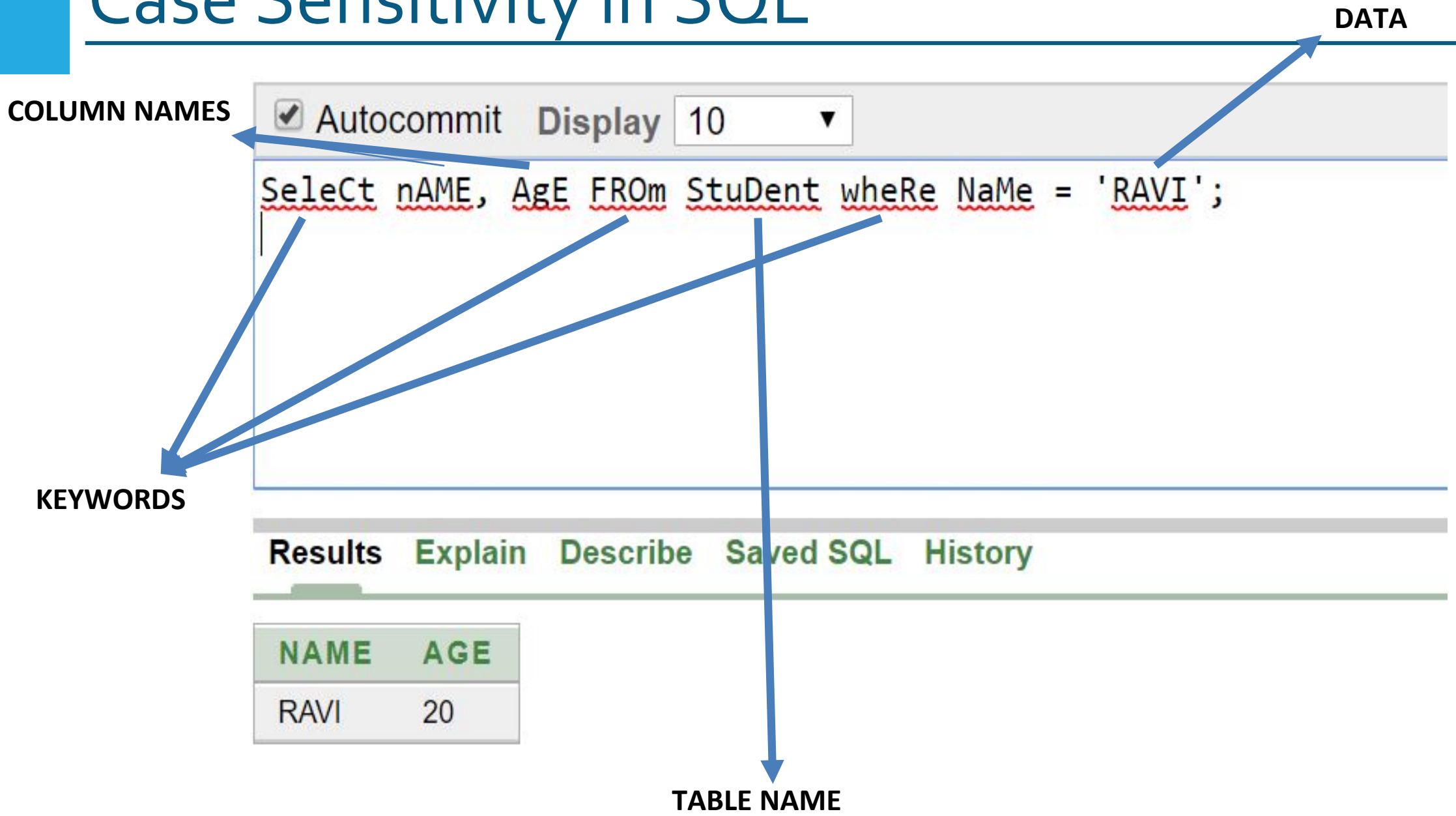
1.KEYWORDS IN SQL IS NOT CASE SENSITIVE.

2.TABLE NAMES IN SQL ARE NOT CASE SENSITIVE.

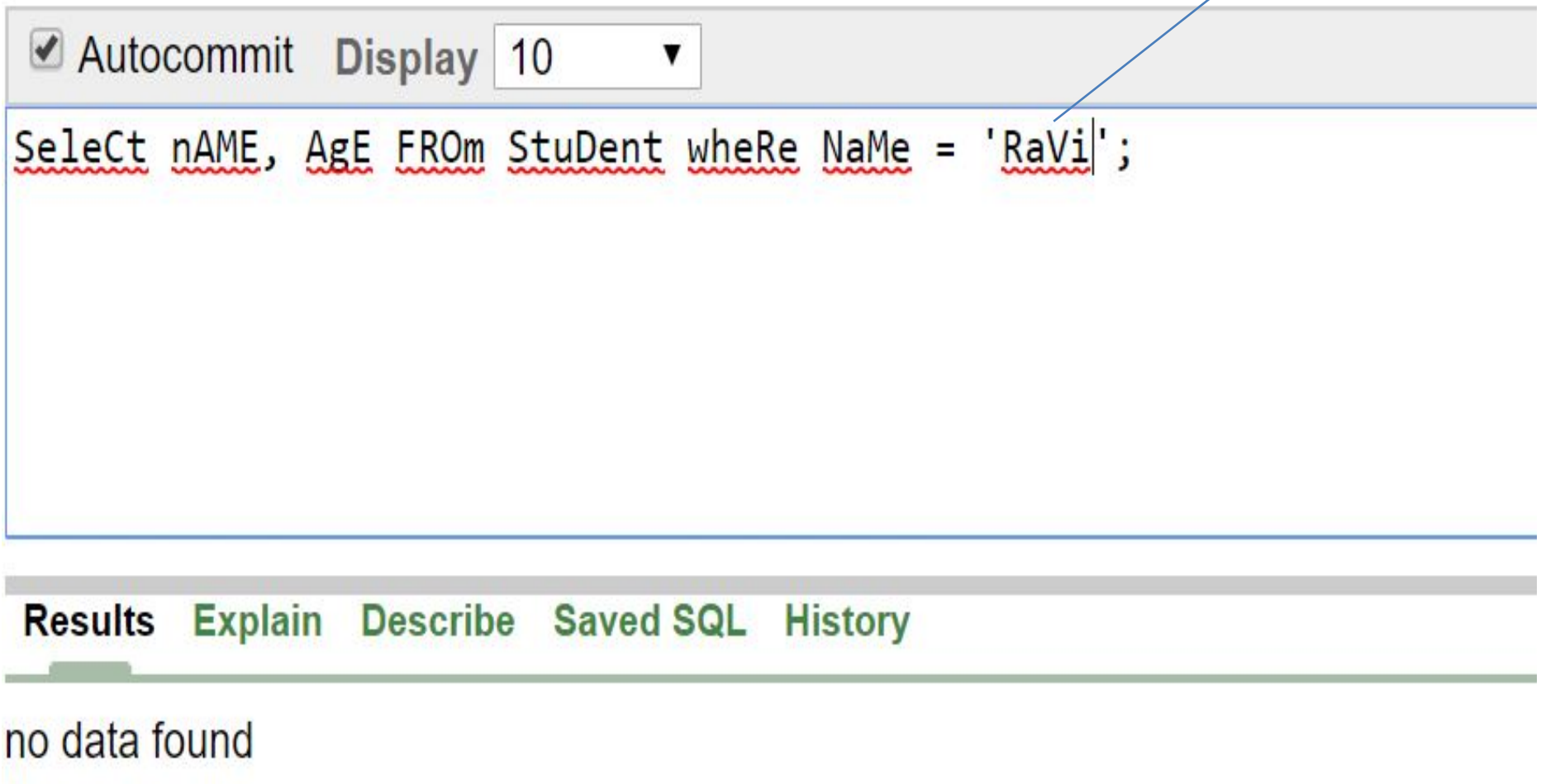
3.COLUMN NAMES IN SQL ARE NOT CASE SENSITIVE.

4.DATA IN SQL IS CASE SENSITIVE.

Case Sensitivity in SQL



Case Sensitivity in SQL



Datatypes

A data type specifies a particular type of data, such as integer, floating-point, Boolean etc.

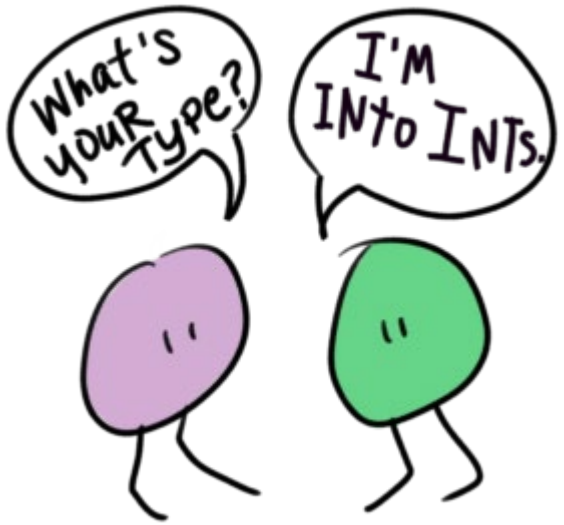
A data type also specifies the possible values for that type, the operations that can be performed on that type and the way the values of that type are stored.

Oracle provides the following built-in datatypes:

Character Datatypes
- CHAR Datatype
- VARCHAR2 and VARCHAR Datatypes
- LONG Datatype

NUMBER Datatype
- FLOAT Datatype

DATE Datatype



Datatypes

Datatype	Description	Column Length and Default
CHAR (*size*)	Fixed-length character data of length *size* bytes.	Fixed for every row in the table (with trailing blanks); maximum size is 2000 bytes per row, default size is 1 byte per row. Consider the character set (one-byte or multibyte) before setting *size.*
VARCHAR (*size*)	Variable-length character data. A maximum *size* must be specified.	Variable for each row, up to 2000 bytes per row. Consider the character set (one-byte or multibyte) before setting *size.* If we declare datatype as **VARCHAR** then it will occupy space for NULL values.
VARCHAR2 (*size*)	Variable-length character data. A maximum *size* must be specified.	Variable for each row, up to 4000 bytes per row. Consider the character set (one-byte or multibyte) before setting *size.* If we declare datatype as **VARCHAR2** then it will not occupy any space for NULL values.

Datatypes

Datatype	Description	Column Length and Default
NUMBER [(p [, s])]	Variable-length numeric data. Range of p : From 1 to 38. Range of s : from -84 to 127.	Variable for each row. The maximum space required for a given column is 21 bytes per row.
FLOAT [(p)]	A subtype of the NUMBER datatype having precision *p*. The precision *p* can range from 1 to 126 binary digits.	A FLOAT value is represented internally as NUMBER. A FLOAT value requires from 1 to 22 bytes.
DATE	Fixed-length date and time data, Valid date range : From January 1, 4712 BC, to December 31, 9999 AD. The default format is determined explicitly by the NLS_DATE_FORMAT parameter or implicitly by the NLS_TERRITORY parameter.	Fixed at 7 bytes for each row in the table. Default format is a string (such as DD-MON-YY) specified by NLS_DATE_FORMAT parameter.

Operators

An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations. These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

OPERATOR	NAME	PRECEDANCE
+	ADD	3
-	SUB	4
*	MUL	1
/	DIV	2

Column Alias

Shubham Agarwal

AKA

Monu

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75

REGISTRATION_NUMBER	NAME	GENDER	YEARS_OLD	SCORE
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75

Column Alias

Aliases are the other names given to columns in SQL.
Whenever aliases names have to be specified for columns then following syntax has to be used

columnname as aliasname
or
columnname “alias name”

NOTE : The alias names will never be reflected in the actual tables present on the hard disk of the computer.

Column Alias

Autocommit Display 10

```
SELECT USN AS REGISTRATION_NUMBER, NAME, GENDER, AGE AS YEARS_OLD, MARKS AS SCORE FROM STUDENT;
```

Results Explain Describe Saved SQL History

REGISTRATION_NUMBER	NAME	GENDER	YEARS_OLD	SCORE
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75

Distinct Keyword

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75

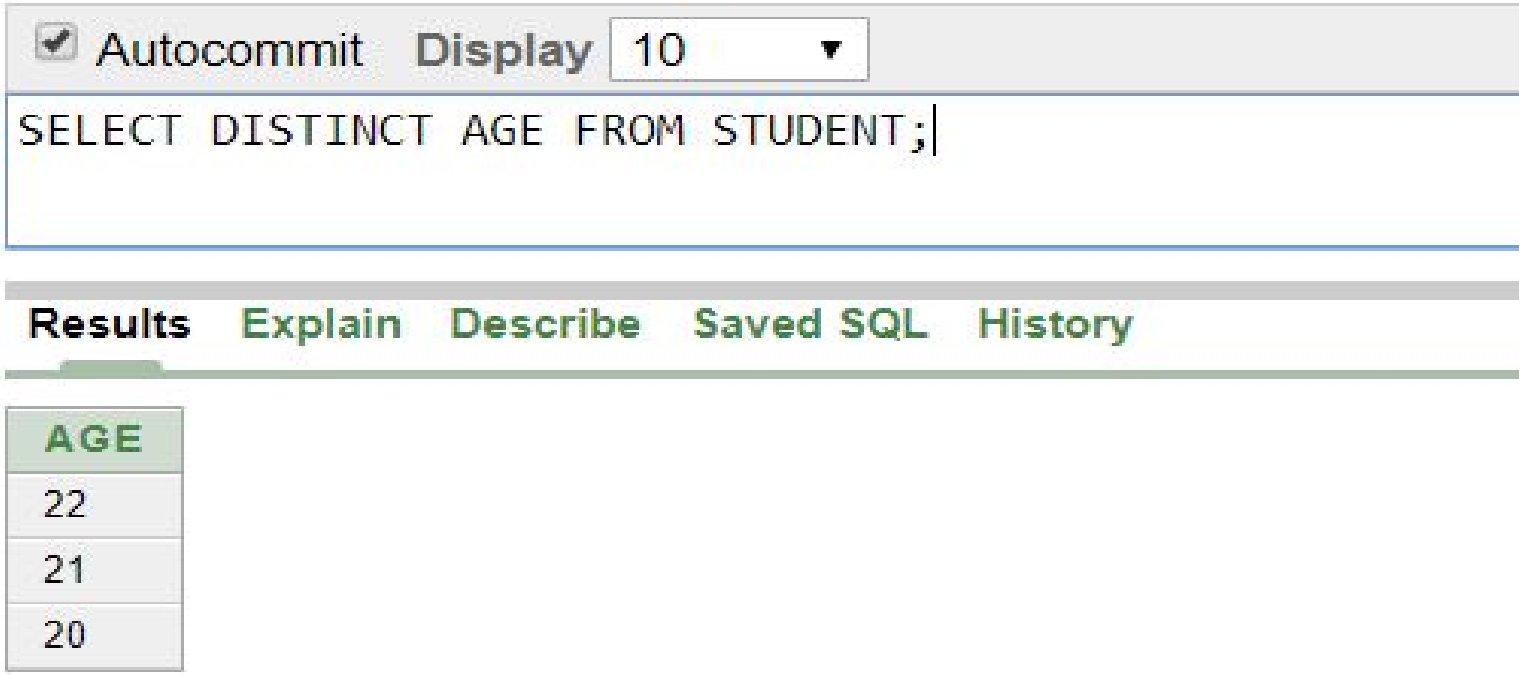
AGE
22
21
20

Distinct Keyword

There may be a situation when you need to retrieve unique records and not multiple duplicate records.

The SQL DISTINCT keyword is used after SELECT to eliminate duplicate records and fetch only unique records.

Note1:Distinct keyword is used to avoid displaying of repeated values



Concatenation Operator

Note1: The operator || is the concatenation operator in SQL.

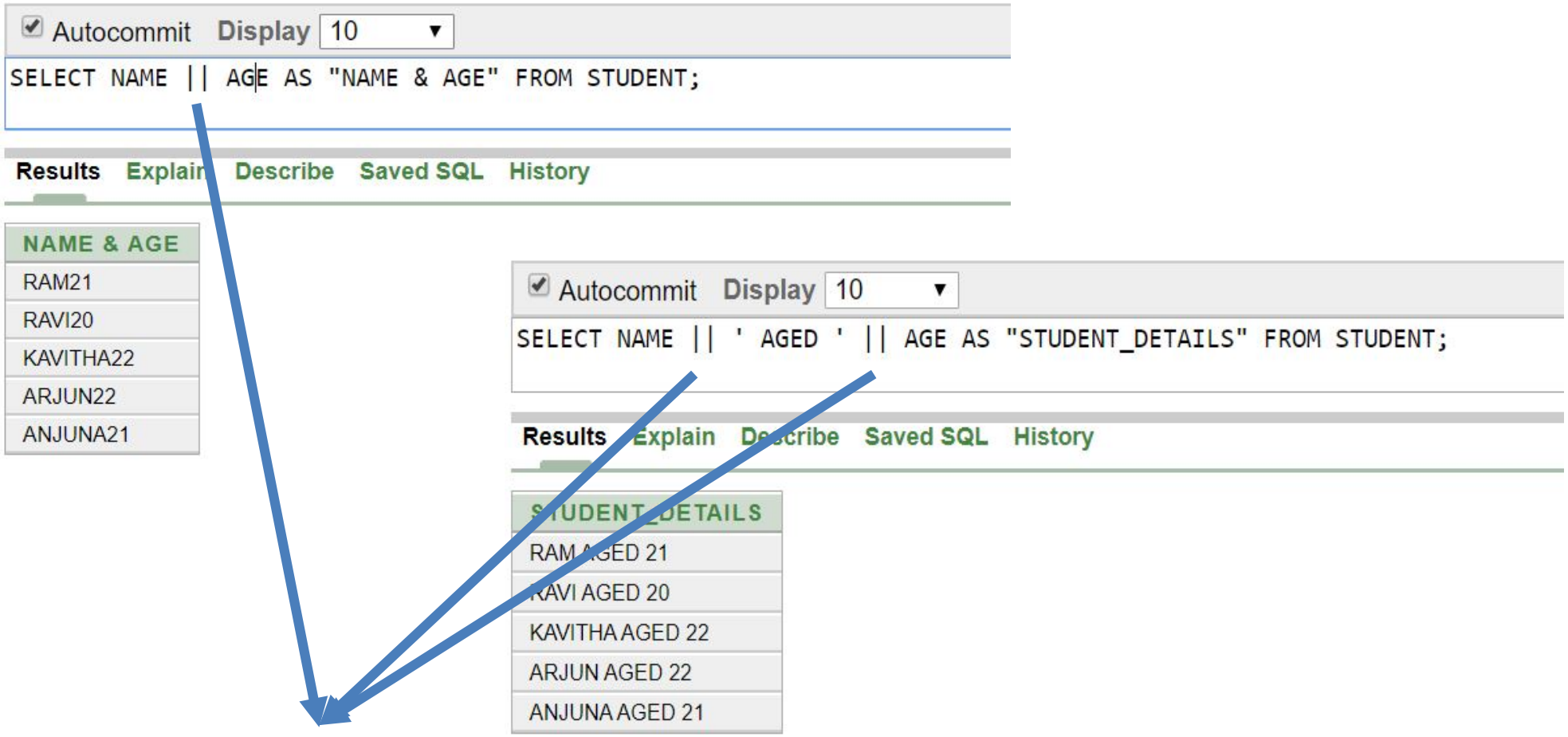
Note2: Concatenation operator is used to combine multiple data or multiple columns.

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75

NAME & AGE
RAM21
RAVI20
KAVITHA22
ARJUN22
ANJUNA21

NAME || AGE AS NAME & AGE

Concatenation Operator



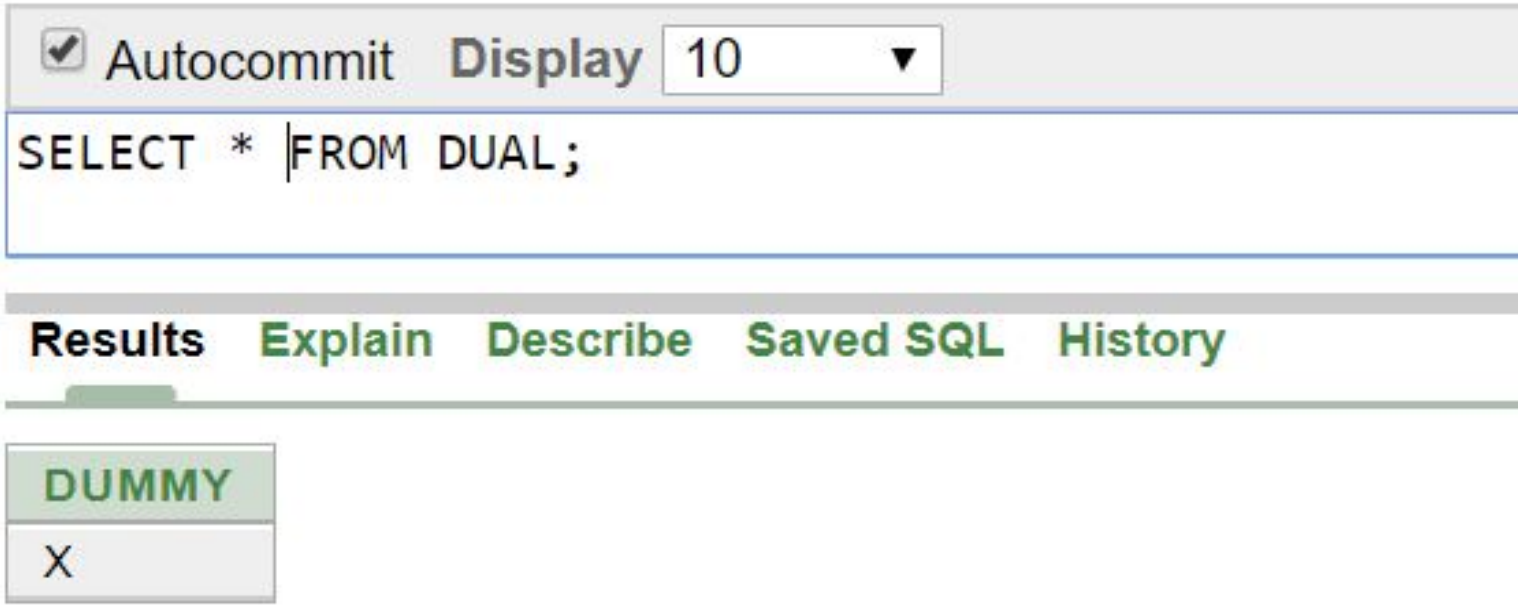
CONCATENATION

DUAL table

The DUAL table is a special one-row, one-column table present by default in Oracle and other database installations.

In Oracle, the table has a single VARCHAR2(1) column called DUMMY that has a value of 'X'.

if the query “select * from dual” is executed then following will be the output :



DUAL table

Want to see how DUAL table looks like or what is the structure ?

Autocommit Display 10

```
DESC DUAL;
```

Results Explain Describe Saved SQL History

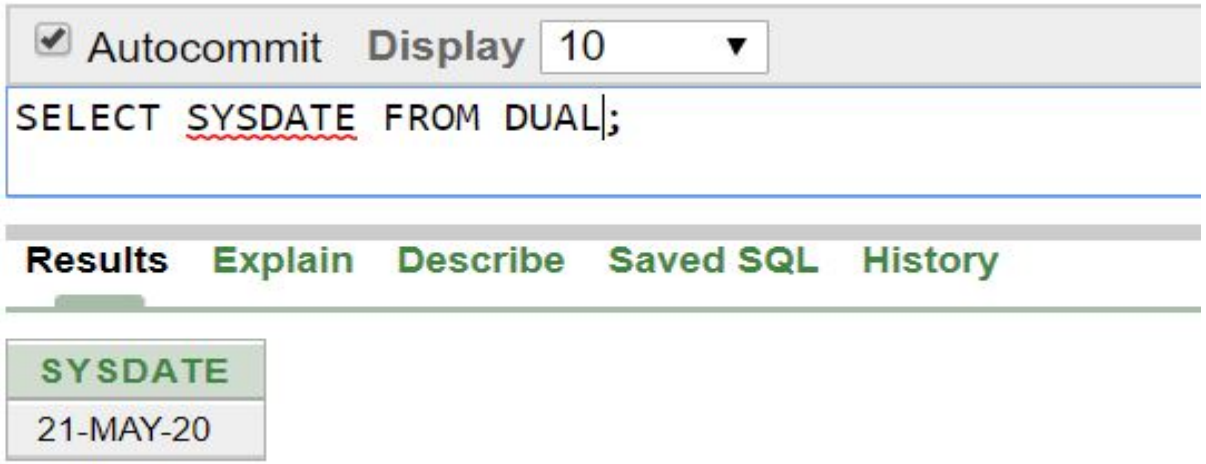
Object Type **TABLE** Object **DUAL**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DUAL	DUMMY	Varchar2	1	-	-	-	✓	-	-

1 - 1

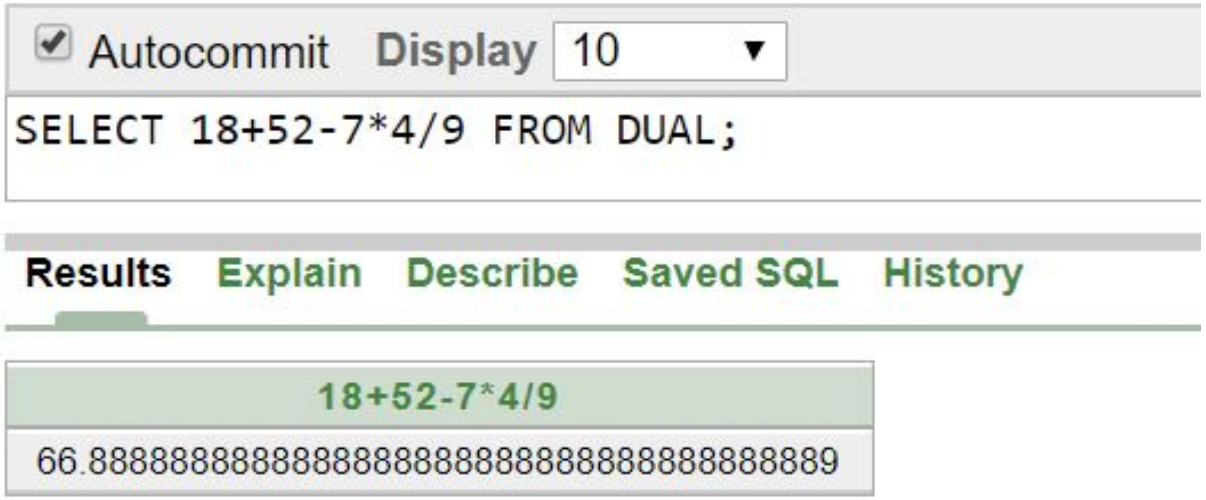
DUAL table

You can also check the system date from the DUAL table using the following statement



DUAL table

You can also check the arithmetic calculation from the DUAL table using the following statement



Operators as keywords

Relational Operators

A comparison (or **relational**) **operator** is a mathematical symbol which is used to compare two values.

Operator	Description
=	Equal to
>	Greater than
<	Lesser than
>=	Greater than or equal to
<=	Lesser than or equal to
!=	Not equal to
< >	Not equal to
^ =	Not equal to

Relational Operators

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75

Q1. Write a query to display USN, NAME and AGE of all the students whose age is equal to 22.

Q2. Write a query to display USN, NAME and AGE of all the students whose age is greater than 20.

Q3. Write a query to display USN, NAME and AGE of all the students whose age is lesser than 22.

Relational Operators

Q4. Write a query to display USN, NAME and AGE of all the students whose age is greater than and equal to 20.

Q5. Write a query to display USN, NAME and AGE of all the students whose age is lesser than or equal to 22.

Q6. Write a query to display USN, NAME and AGE of all the students whose age is not equal to 20.

Operator	Precedence
=	1
>	2
<	3
>=	4
<=	5
!=	6
< >	6
^ =	6

Operators as keywords

1. BETWEEN AND OPERATOR

When the condition has a range of values to be compared we should be using BETWEEN AND operator.

Q. Write a query to display all the information of the students whose MARKS is in between 80 and 90 from the STUDENT table.

Autocommit Display 10

```
SELECT * FROM student WHERE MARKS BETWEEN 80 AND 90;
```

Results Explain Describe Saved SQL History

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87

Operators as keywords

Autocommit Display 10

```
SELECT * FROM student WHERE MARKS BETWEEN 90 AND 80;
```

Results Explain Describe Saved SQL History

no data found

If we use higher value first and lower value after that, then "no data found" will be the output.

If we want to get the output for data which doesn't come in the range of given values.

Autocommit Display 10

```
SELECT * FROM student WHERE MARKS NOT BETWEEN 80 AND 90;
```

Results Explain Describe Saved SQL History

USN	NAME	GENDER	AGE	MARKS
124	ANJUNA	FEMALE	21	75

Operators as keywords

2. IN OPERATOR

When ever comparison has to be done with respect to a set of values we have to use the IN operator.

Q. Write a query to display all the information of the students whose AGE is 20 and 22 from the STUDENT table.

Autocommit Display 10

```
SELECT * FROM student WHERE AGE IN (20,22);
```

Results Explain Describe Saved SQL History

USN	NAME	GENDER	AGE	MARKS
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87

Operators as keywords

3. LIKE KEYWORD

The LIKE operator is used to search for a specified pattern in a column.

Note: Pattern matching in SQL

There are two pattern matching symbols in sql

1) % (modulus) it matches 0 or more characters
2) _ (underscore) it matches exactly one character

Operators as keywords

Q. Write a query to display all the details of the student whose names start with “Ra”.

Autocommit Display 10

```
SELECT * FROM student WHERE NAME LIKE 'RA%';
```

Results Explain Describe Saved SQL History

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88

Operators as keywords

Q24. Write a query to display all the details of the students whose names fourth character is “I” from the STUDENT table.

Autocommit Display 10

```
SELECT * FROM student WHERE NAME LIKE '___I%';
```

Results Explain Describe Saved SQL History

USN	NAME	GENDER	AGE	MARKS
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90

Operators as keywords

4. IS NULL OPERATOR

It is used to select only the records with NULL values in the column

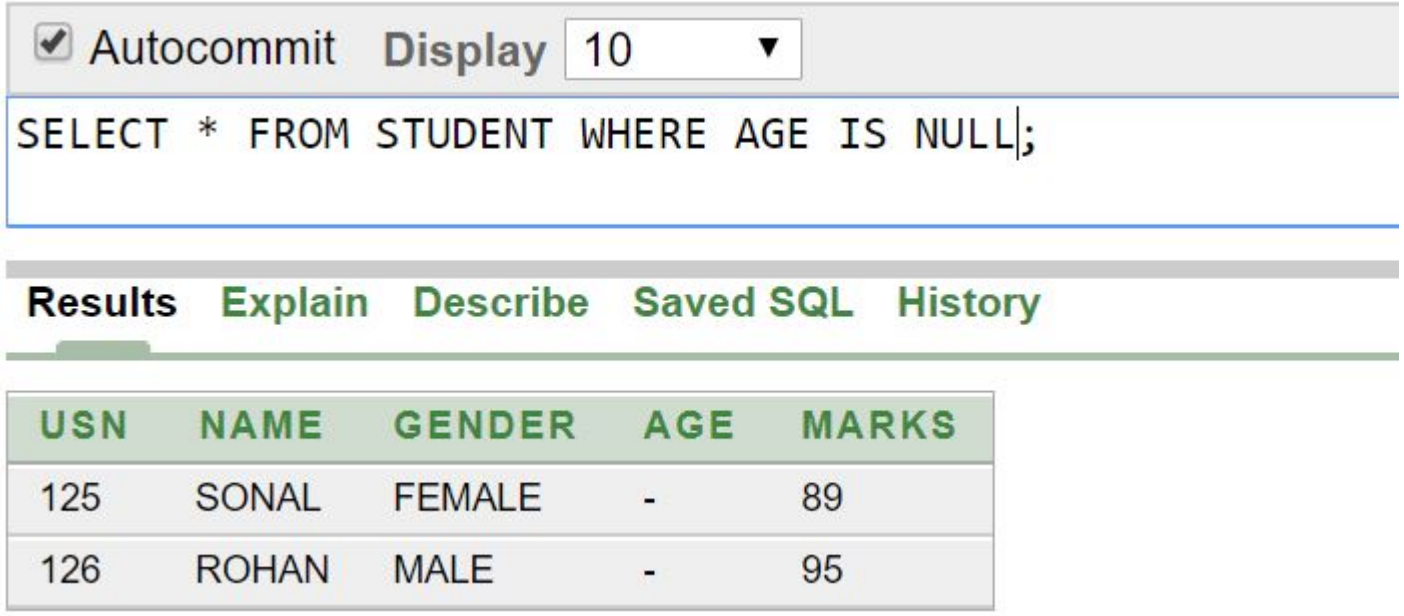
Q. Write a query to display all the details of the student whose age is null from student table.

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75
125	SONAL	FEMALE	-	89
126	ROHAN	MALE	-	95
127	MANISHA	FEMALE	25	85

UPDATED TABLE

Operators as keywords

Q. Write a query to display all the details of the student whose age is null from student table.



USN	NAME	GENDER	AGE	MARKS
125	SONAL	FEMALE	-	89
126	ROHAN	MALE	-	95

Operators as keywords

5. LOGICAL OPERATORS

OPERATORS	DESCRIPTION
AND	TRUE if all the conditions separated by AND is TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
NOT	Displays a record if the condition(s) is NOT TRUE

Q. Write a query to display all the details of the students whose marks is greater than 85 and their names should not start with “R” from STUDENT table.

Operators as keywords

Q. Write a query to display all the details of the students whose marks is greater than 85 and their names should not start with “R” from STUDENT table.

Autocommit Display 10

```
SELECT * FROM STUDENT WHERE MARKS>85 AND NAME NOT LIKE 'R%';
```

Results Explain Describe Saved SQL History

USN	NAME	GENDER	AGE	MARKS
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
125	SONAL	FEMALE	-	89

Functions

Functions

In SQL we have two types functions :

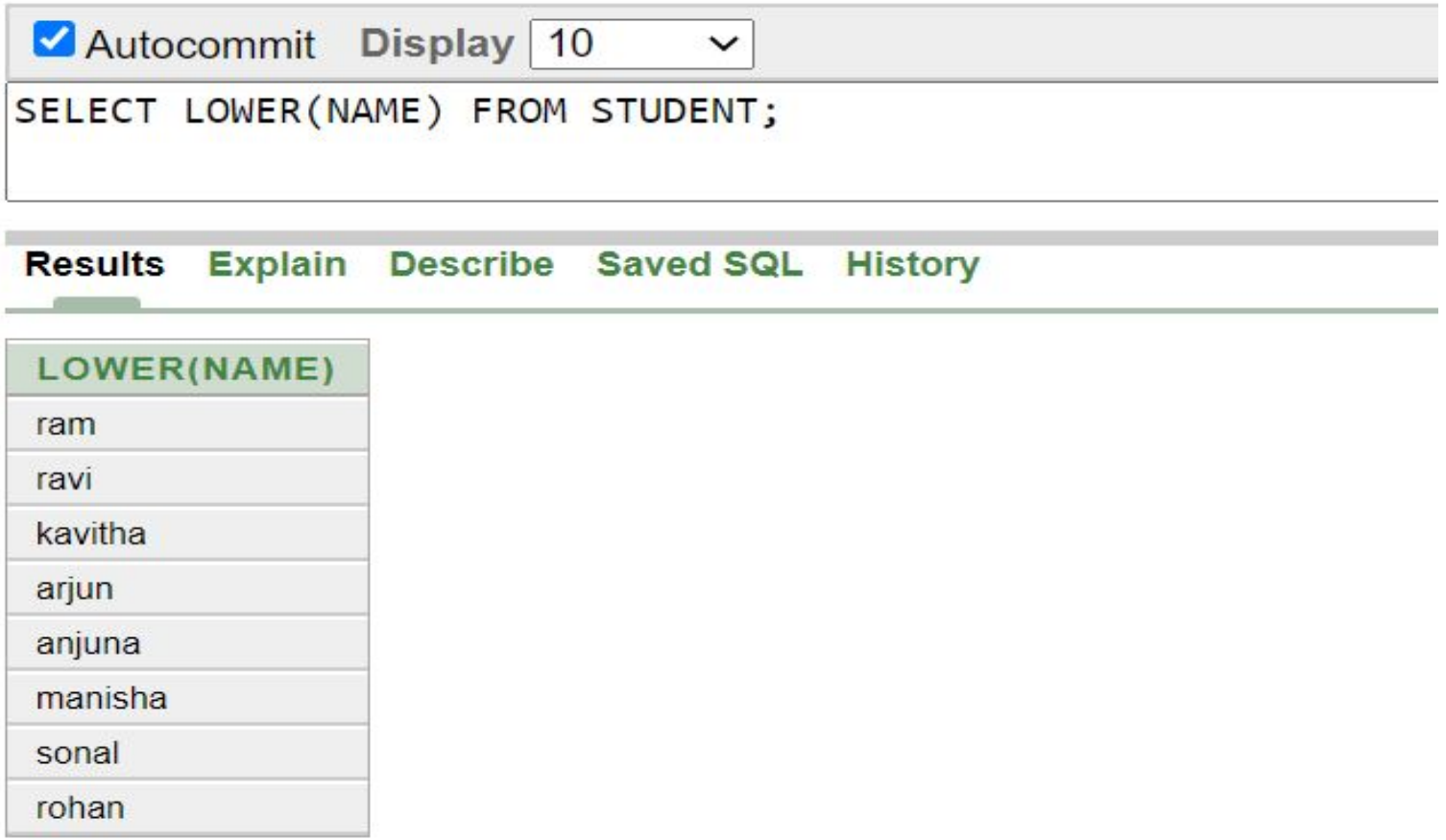
1) Single row functions
2) Multiple row functions

Single row functions are such functions which will accept a single row as input (or) it accept multiple row as input but produces one result per row.

Multiple row functions/group functions/aggregate functions : are such functions which will accept a single row or multiple row as input but produces one result per group.

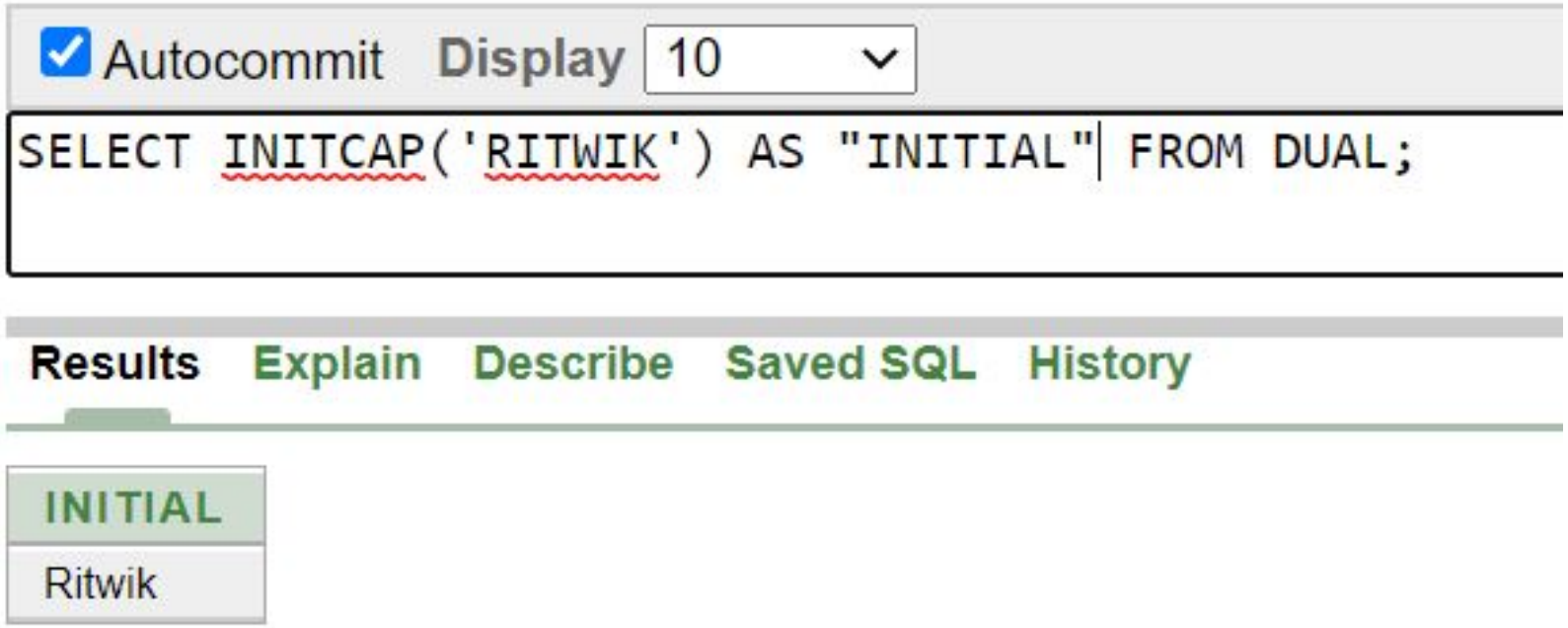
Functions

Q. Write a query to display NAME of all the students in lowercase letters from the table STUDENT.



Functions

Q. Write a query to display Initial letter capital of data ‘RITWIK’.



Functions

Q. Write a query to display all the names and gender of the students in lowercase where gender is ‘MALE’.

Autocommit Display 10

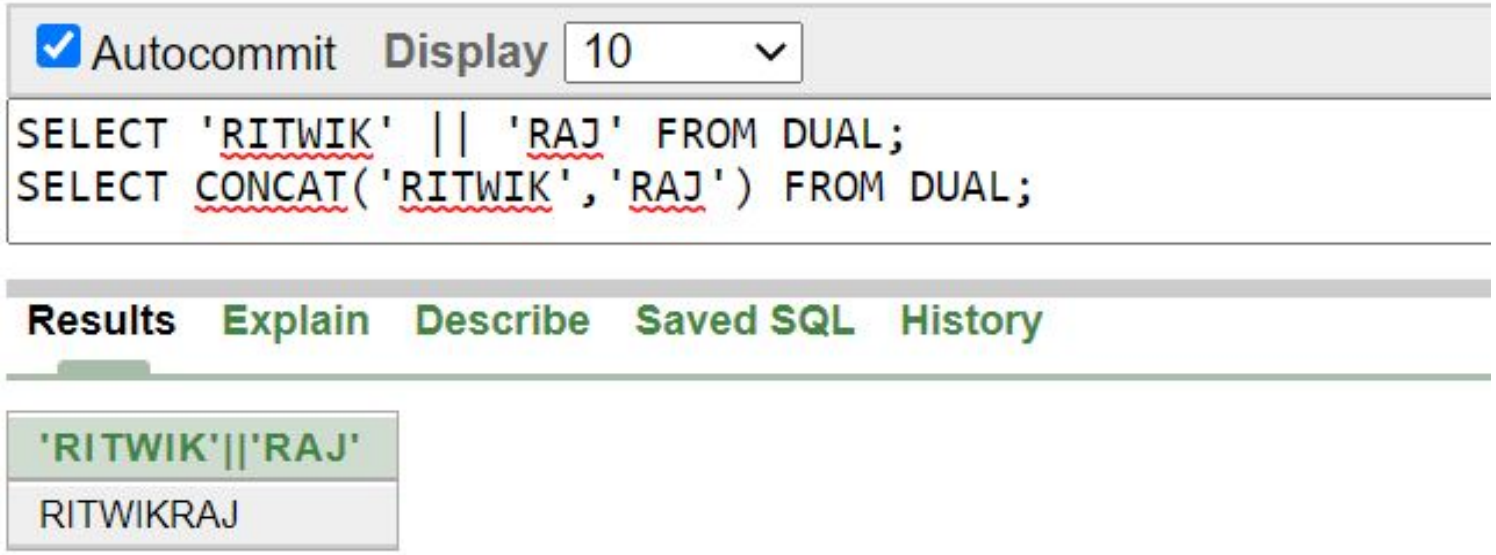
```
SELECT LOWER(NAME), LOWER(GENDER) FROM STUDENT WHERE GENDER = 'MALE';
```

Results Explain Describe Saved SQL History

LOWER(NAME)	LOWER(GENDER)
ram	male
ravi	male
arjun	male
rohan	male

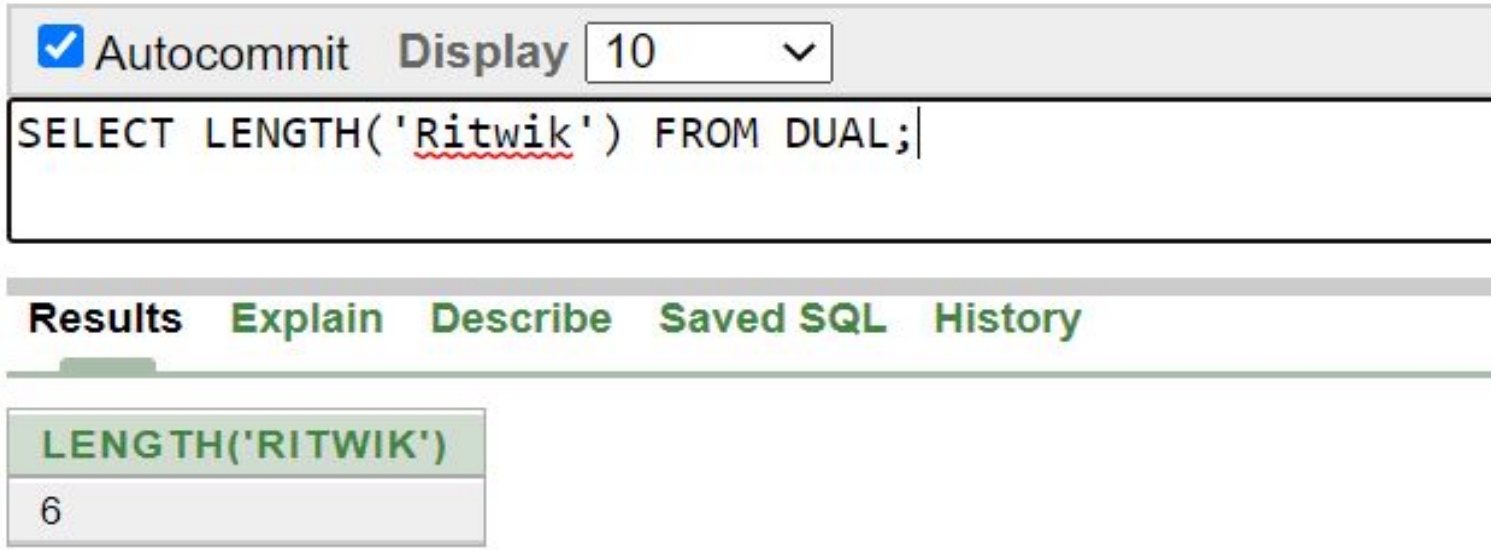
Functions

Q. Write a query to concatenate data ‘Ritwik’ & ‘Raj’.



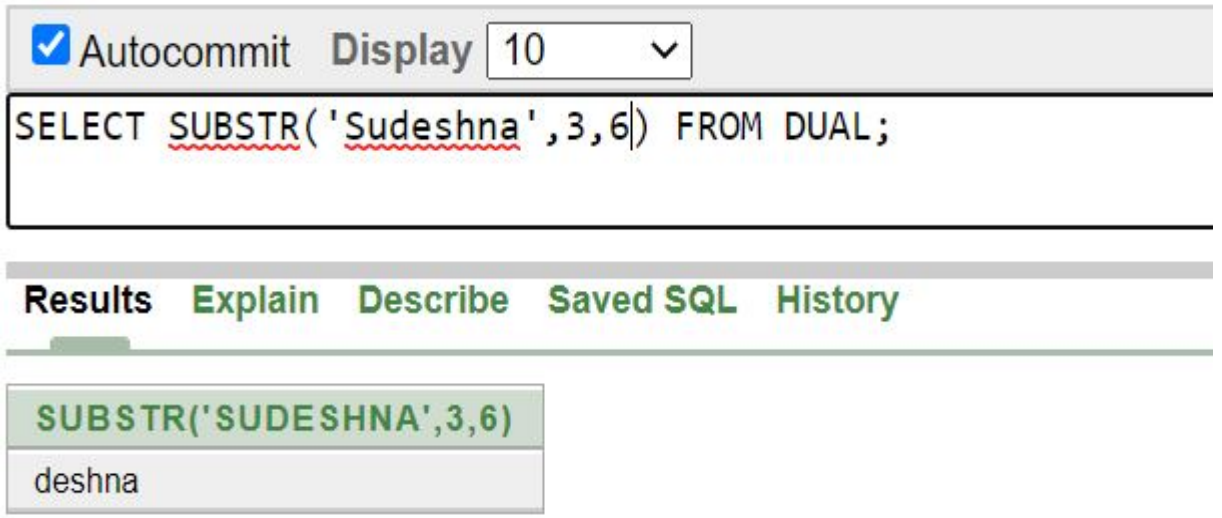
Functions

Q. Write a query to find the length of the string ‘Ritwik’.



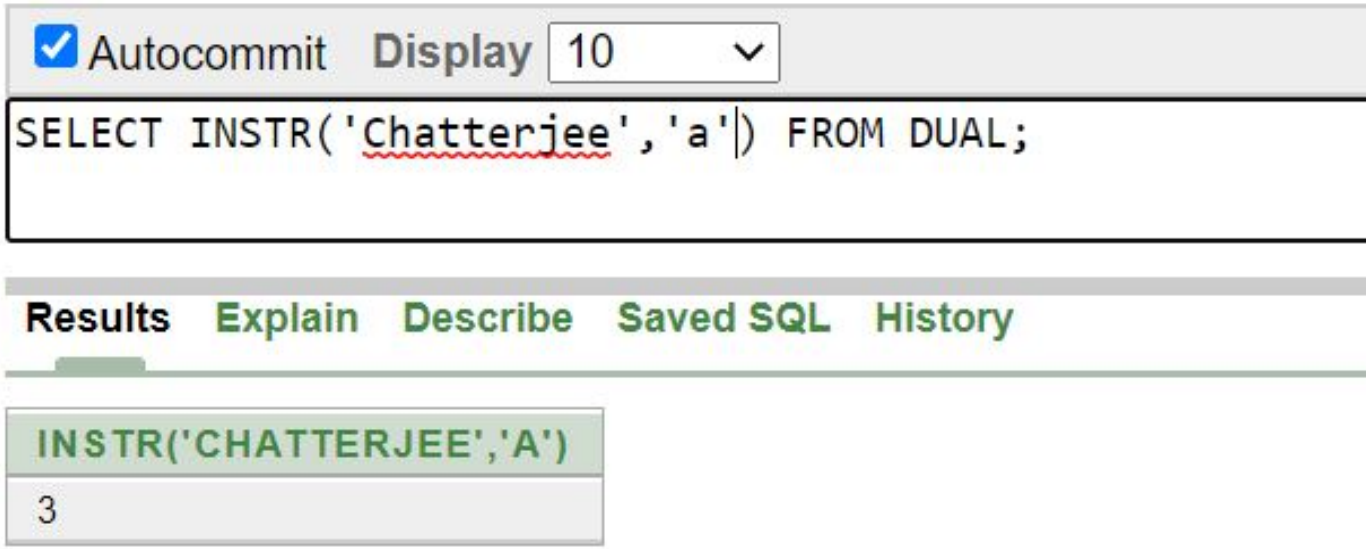
Functions

Q. Write a query to display the substring of the string ‘Sudeshna’ from 3rd position extract 6 characters.



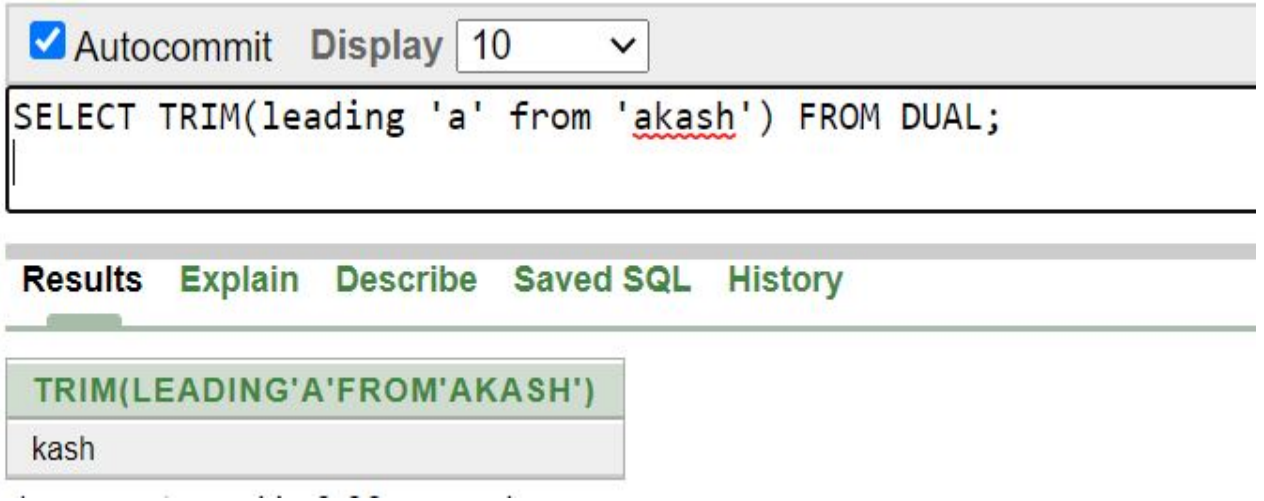
Functions

Q. Write a query to display the position of the character ‘a’ in the string ‘Chatterjee’.

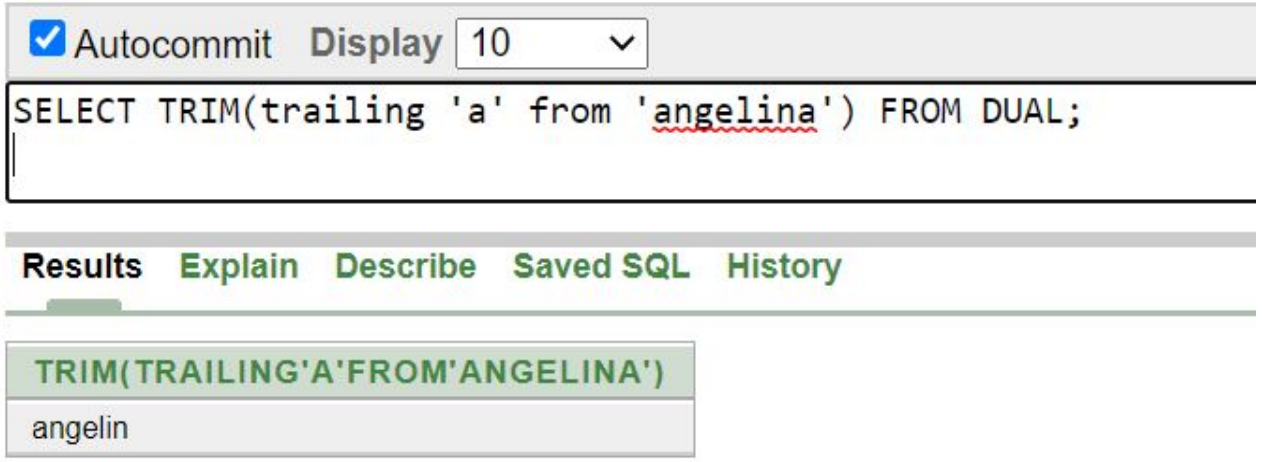


Functions

Q. Write a query to trim the leading ‘a’ in the string ‘akash’.

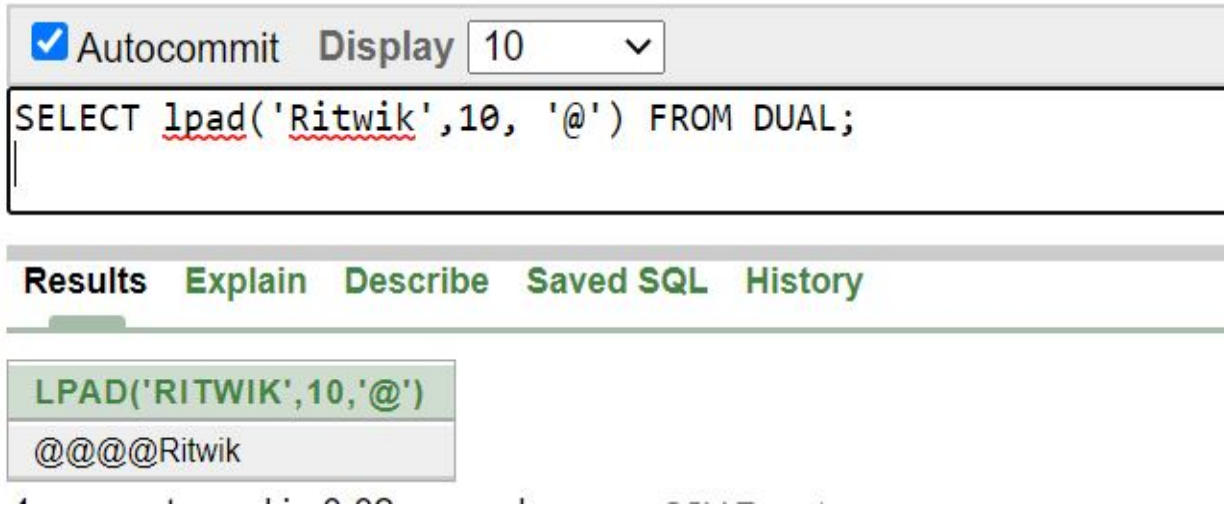


Q. Write a query to trim the trailing ‘a’ in the string ‘angelina’.

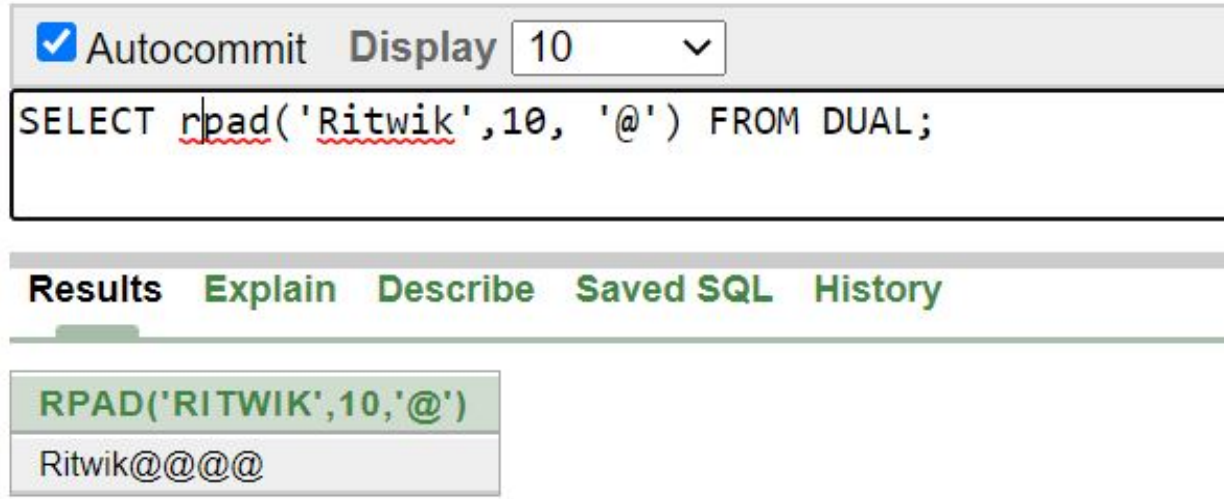


Functions

Q. Write a query to display the data ‘Ritwik’ in the format ‘@@@@Ritwik’.



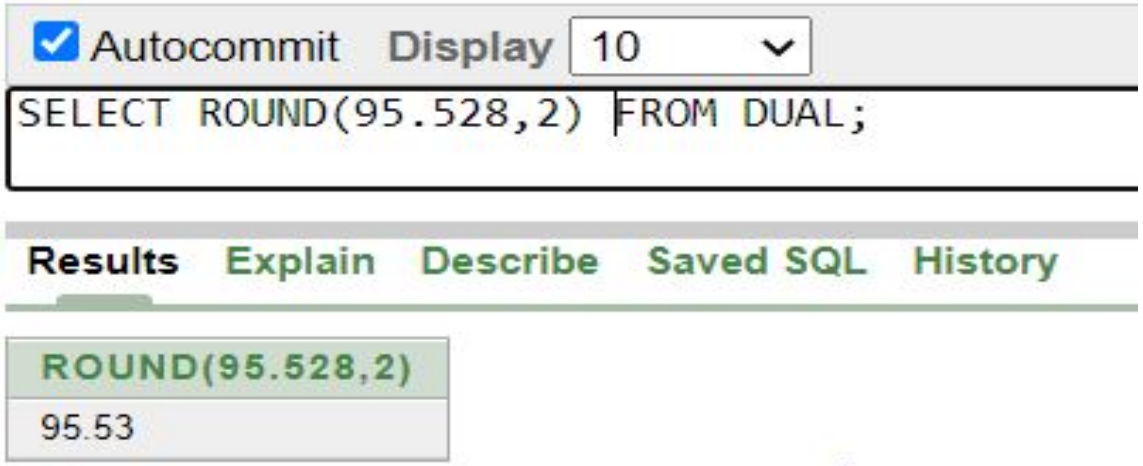
Q. Write a query to display the data ‘Ritwik’ in the format ‘Ritwik@@@@’.



Functions

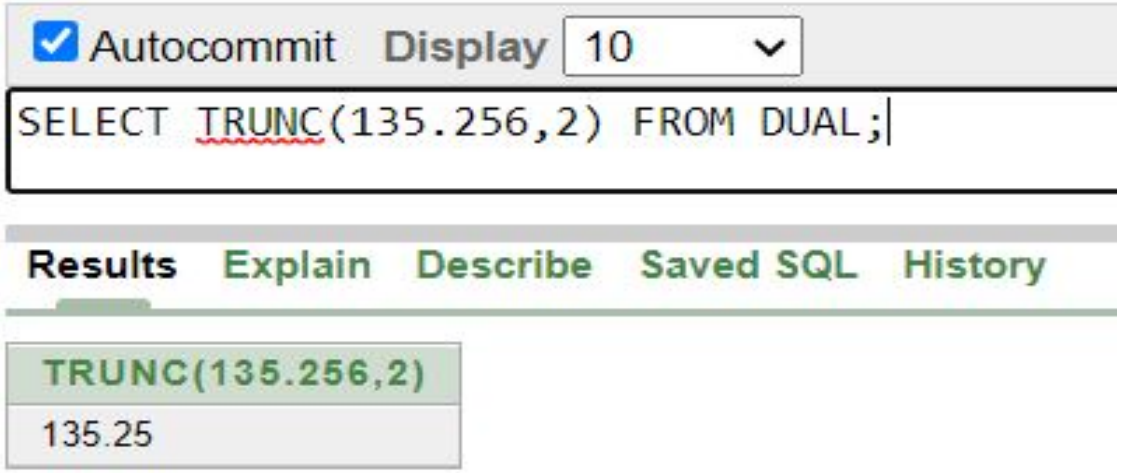
Q. What would be the output of the following query

SELECT ROUND(95.528,2) FROM DUAL;



Q. What would be the output of the following query

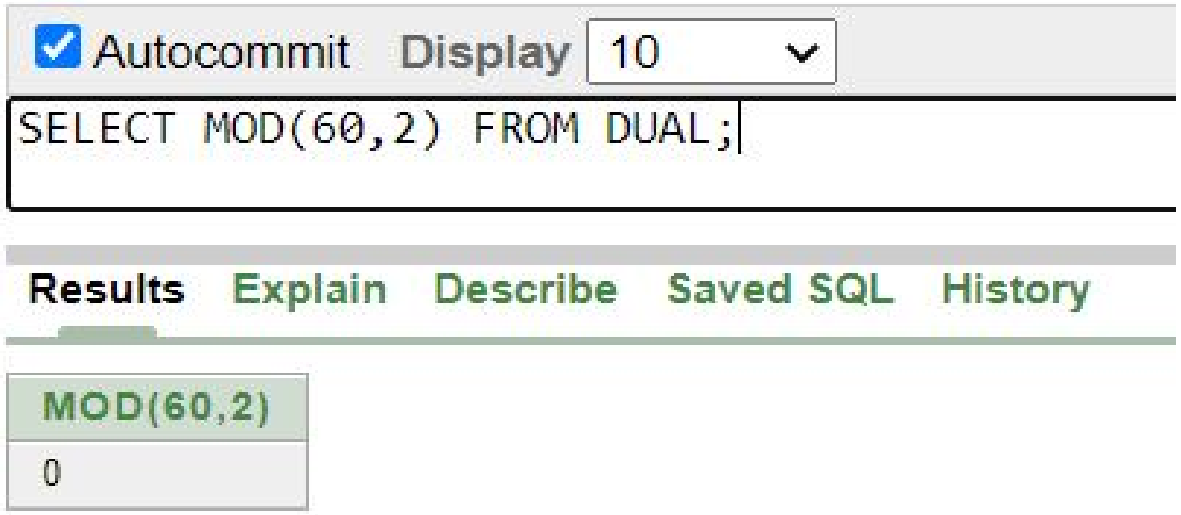
SELECT TRUNC(135.256,2) FROM DUAL;



Functions

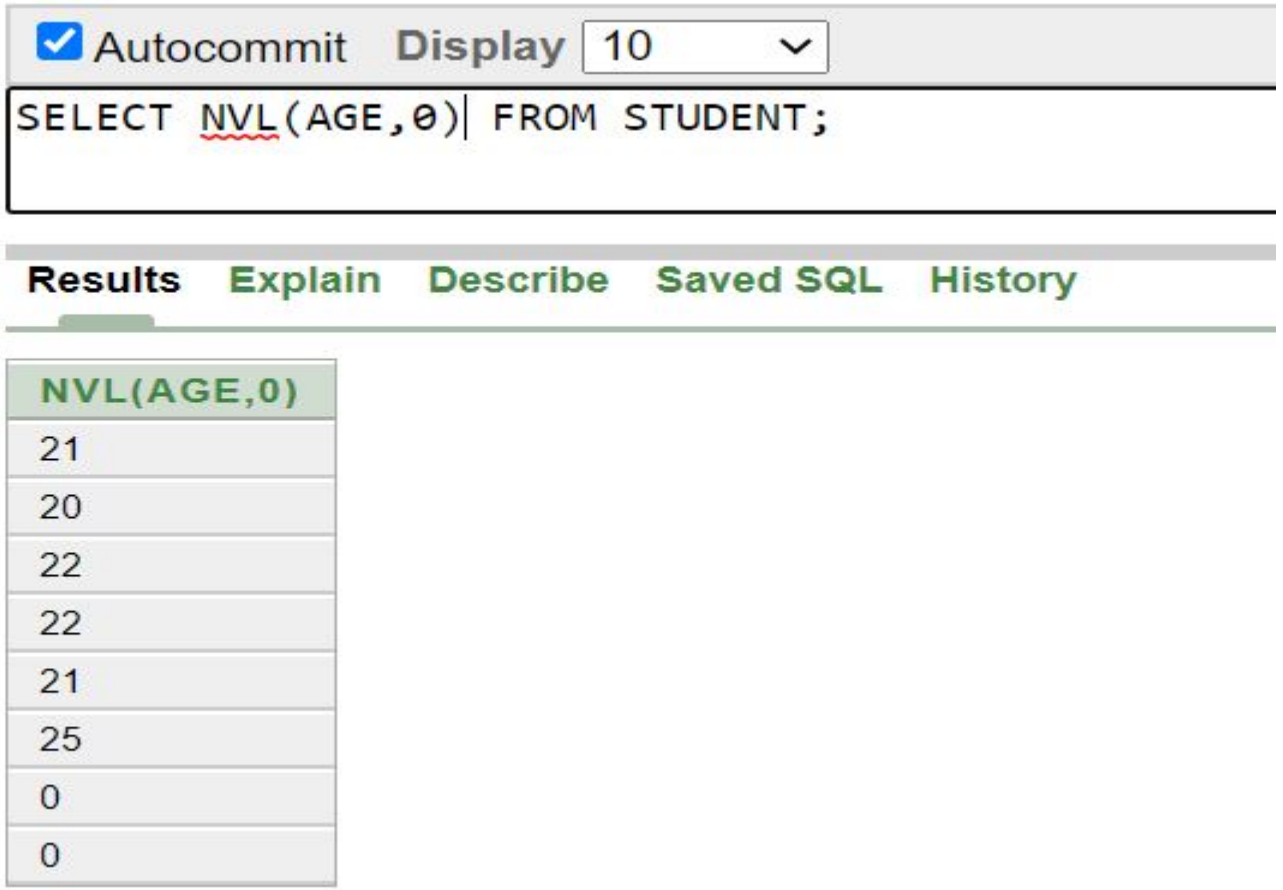
Q. What would be the output of the following query

SELECT MOD(60,2) FROM DUAL;

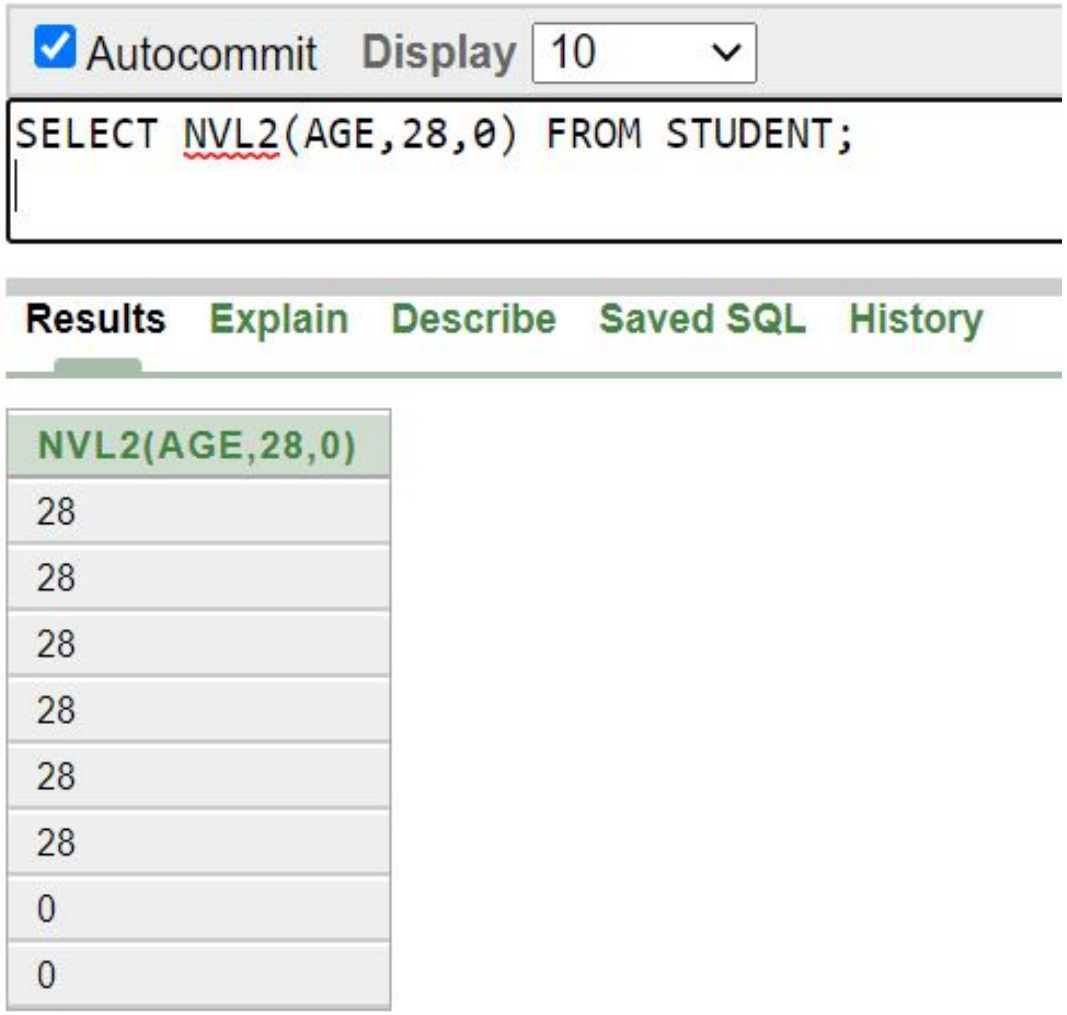


Functions

Q. Write a query to display the age. If age is null replace it with 0.



Functions

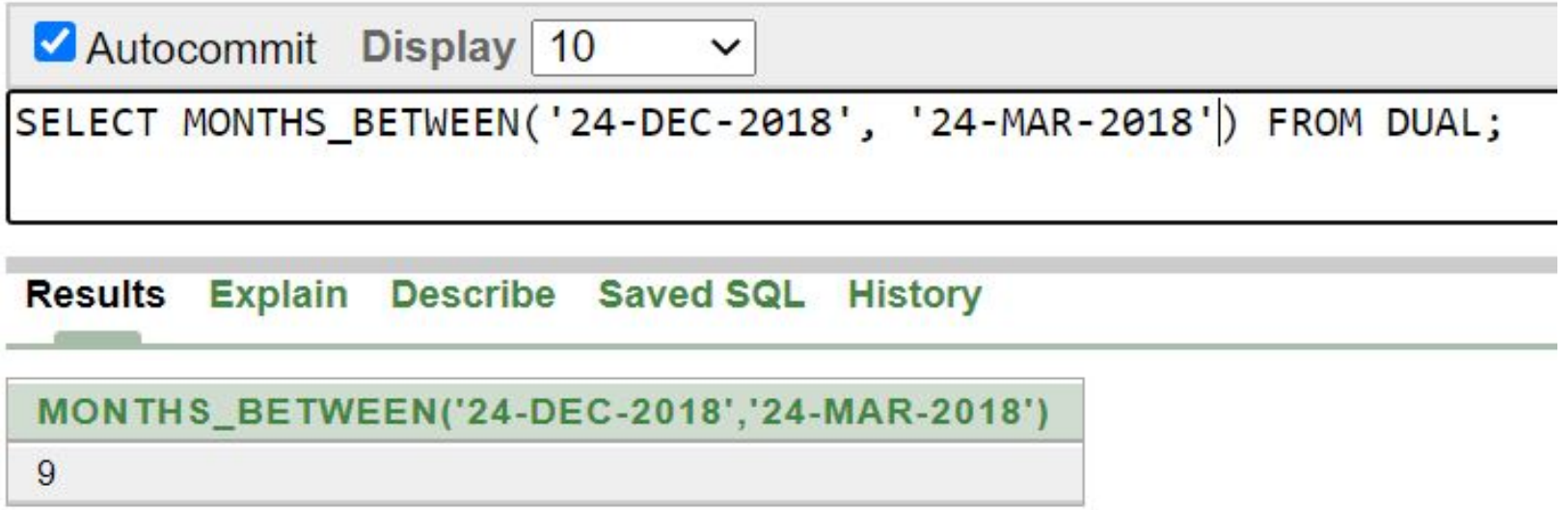


NVL2(AGE,28,0)
28
28
28
28
28
28
0
0

Q. Write a query to display the age. If age is null replace it with 0 else replace with 28.

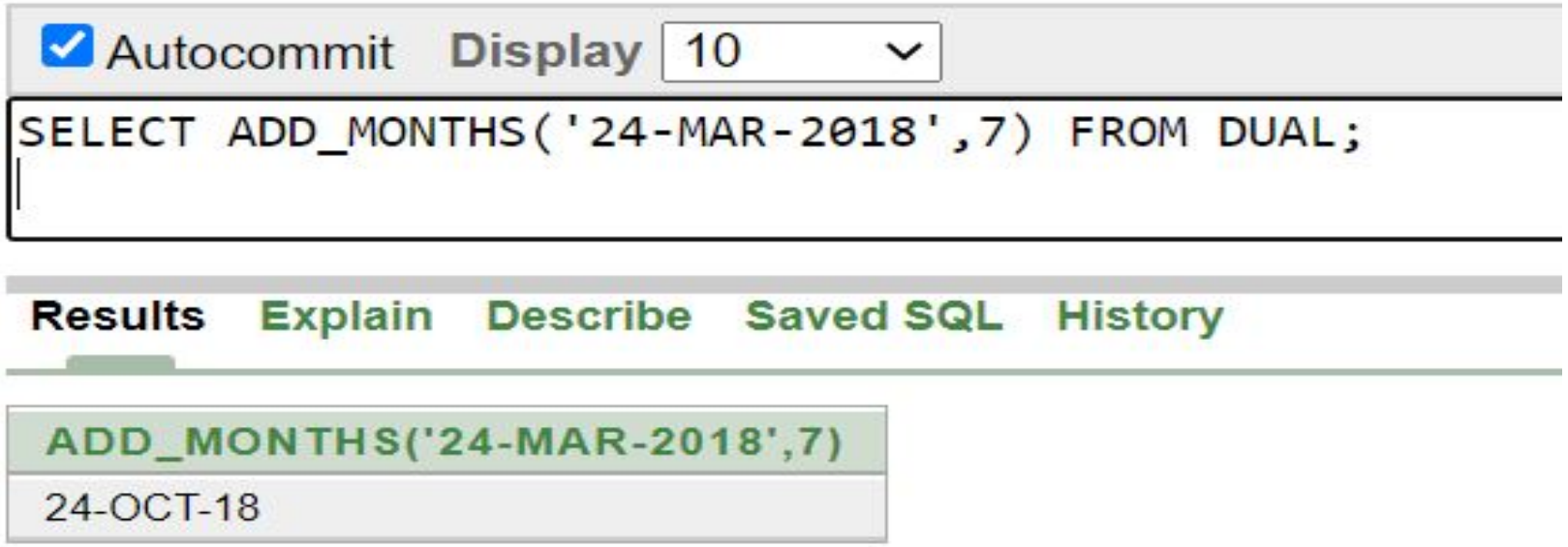
Functions

Q. Write a query to display the number of months between the dates 24-dec-2018 and 24-mar-2018.



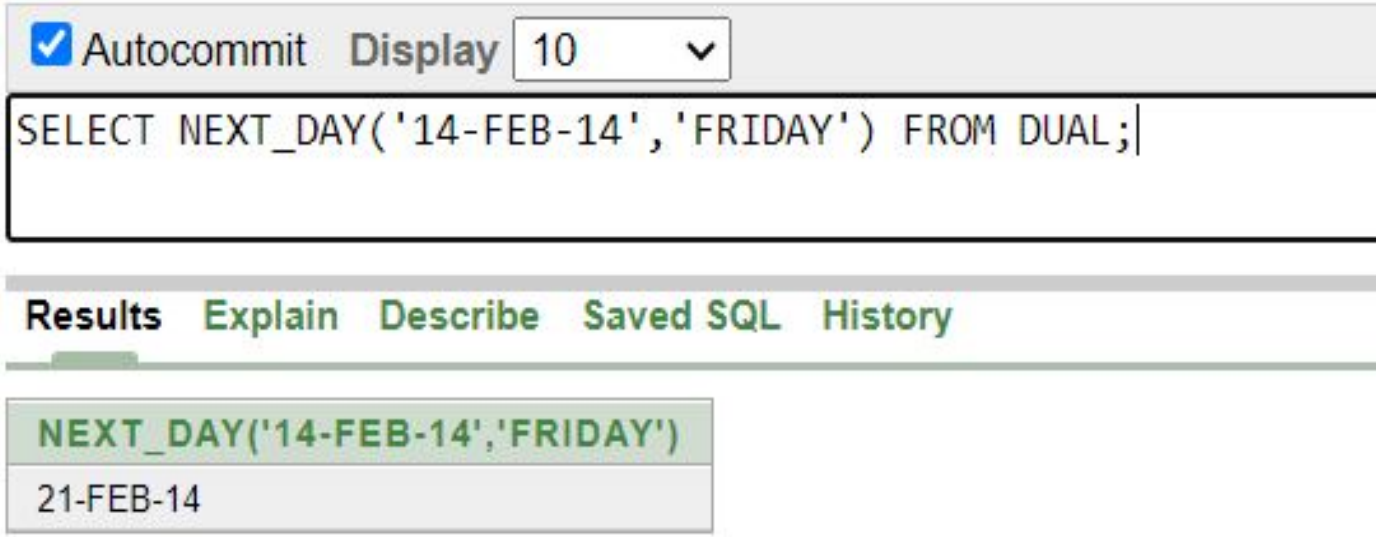
Functions

Q. Write a query in order to add 7 months for the date 24-mar-2018.



Functions

Q. Write a query to find the date of the next Friday after the date 14-feb-2014.



Functions

The Oracle NULLIF() function accepts two arguments. It returns a null value if the two arguments are equal.
In case the arguments are not equal, the NULLIF() function returns the first argument.

Autocommit Display 10

```
SELECT NULLIF(100,100) FROM DUAL;
```

Results Explain Describe Saved SQL History

NULLIF(100,100)
-

Autocommit Display 10

```
SELECT NULLIF(100,200) FROM DUAL;
```

Results Explain Describe Saved SQL History

NULLIF(100,200)
100

Functions

Q. Write a query to display the l_name, hire_date where the hire_date should be displayed in the format dd/mm

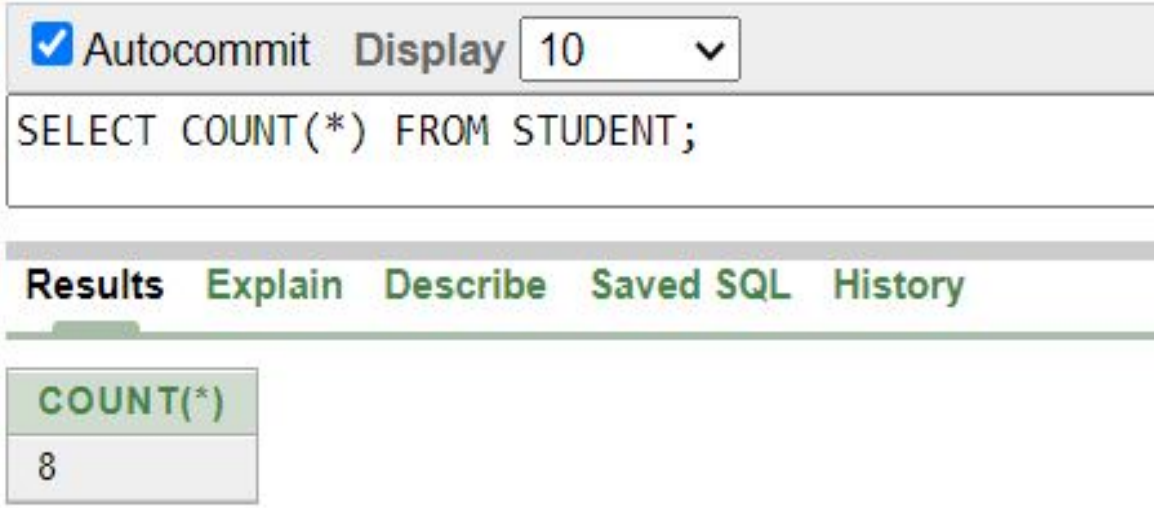
```
SELECT   L_NAME   TO_CHAR(HIRE_DATE,  'DD/MM')   FROM    EMP;
```

Q. Write a query to display the l_name and salary . the salary should be displayed in the format $99,999.99

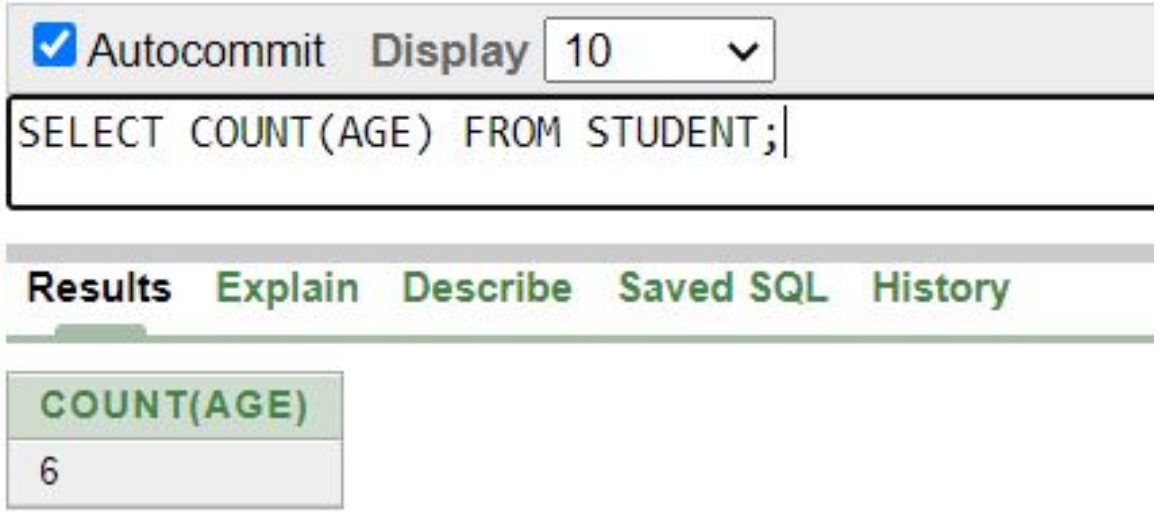
```
SELECT   L_NAME   TO_CHAR(SALARY,  '$99999.99')   FROM   EMP;
```

Functions

Q. Write a query to display count of all the rows present in STUDENT table

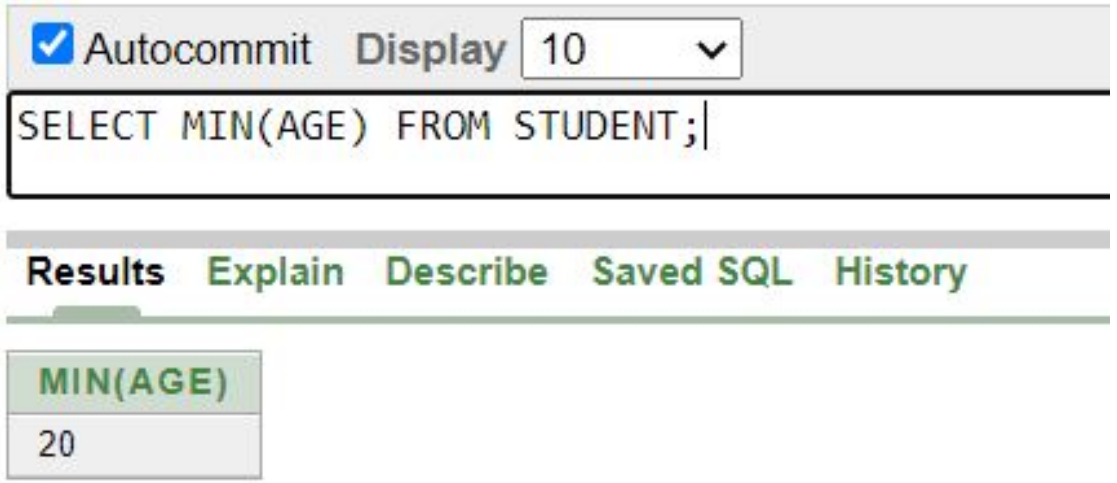


Q. Write a query to display count of DISTINCT AGE present in STUDENT table

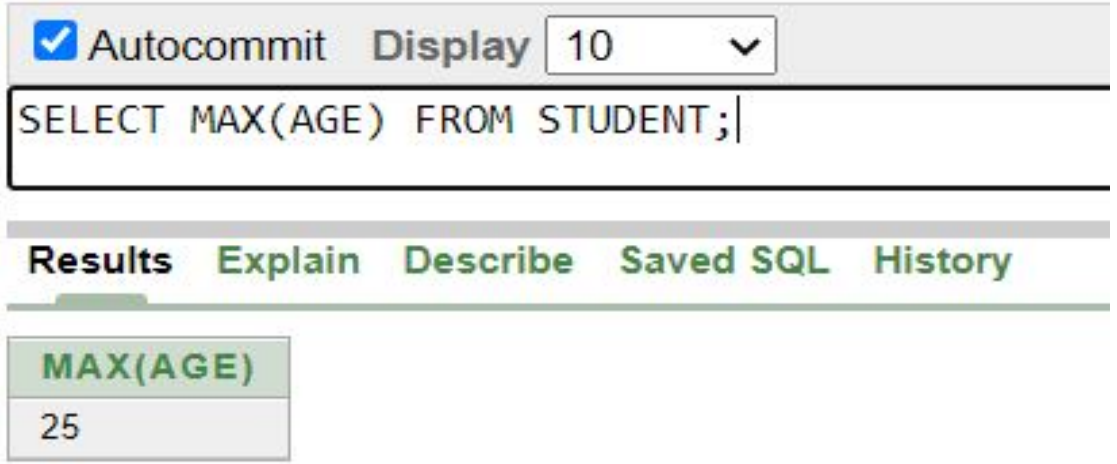


Functions

Q. Write a query to display the minimum of all age present in STUDENT table

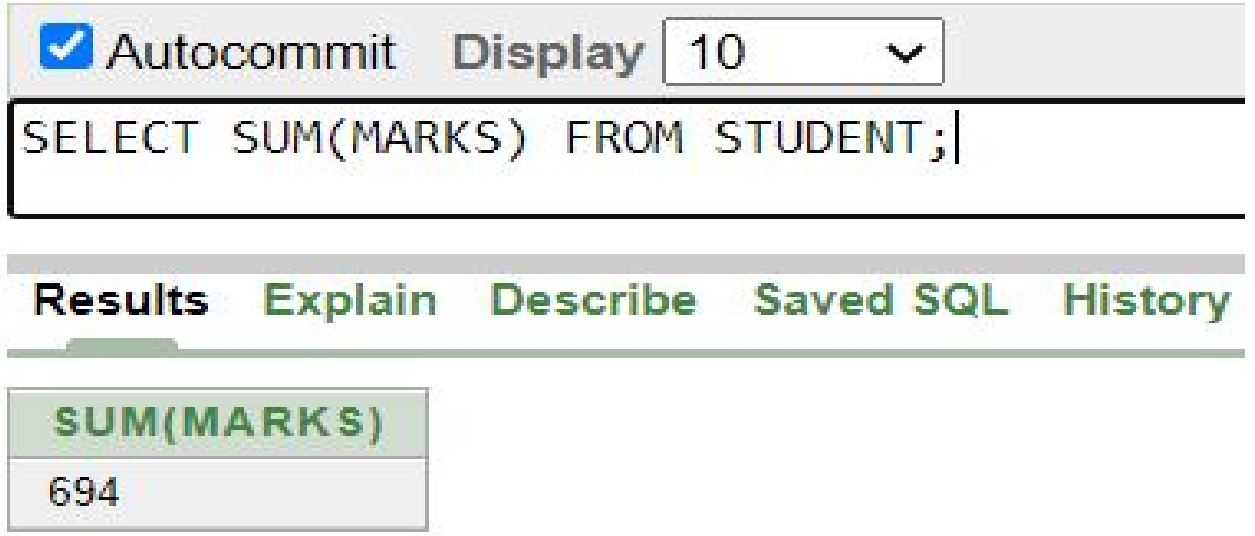


Q. Write a query to display the maximum of all age present in STUDENT table

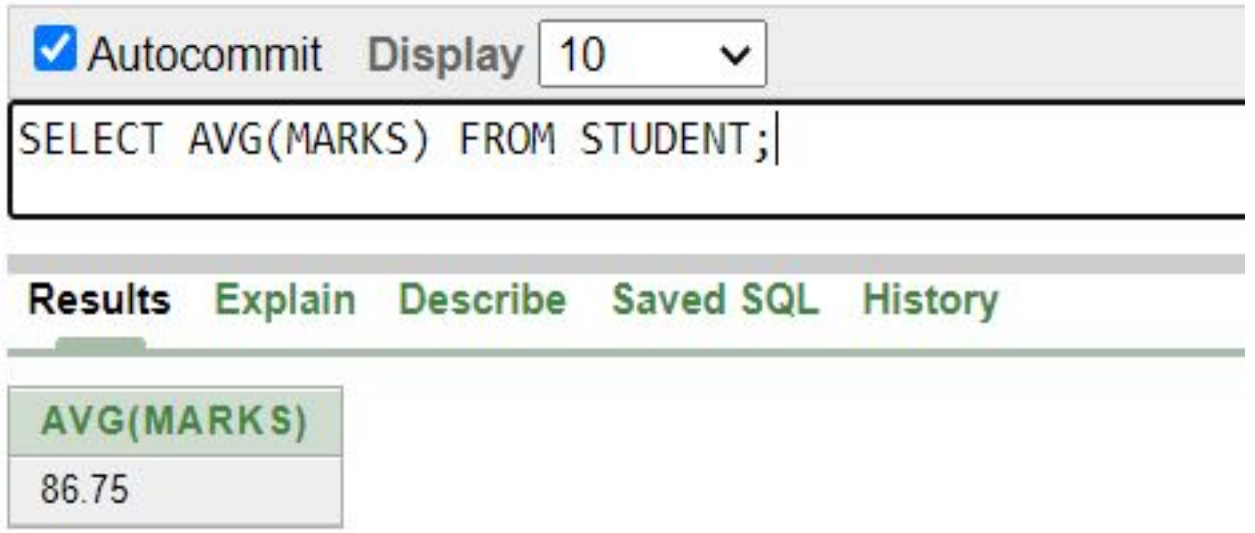


Functions

Q. Write a query to display the sum of all the marks present in STUDENT table



Q. Write a query to display the average of all the marks present in STUDENT table



Clauses

Clauses

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75
125	SONAL	FEMALE	-	89
126	ROHAN	MALE	-	95
127	MANISHA	FEMALE	25	85

Student Table

Q. Write a query to display all the details of all the students in the order of AGE from STUDENT table.

Order By

The output of the SQL query can be predicted but the order in which the rows get displayed cannot be predicted.

If we have to print the output in a specific order we should be using order by clause

Syntax:

order by column_name [asc/desc]

Autocommit Display 10

```
SELECT * FROM STUDENT ORDER BY AGE;
```

Results Explain Describe Saved SQL History

USN	NAME	GENDER	AGE	MARKS
121	RAVI	MALE	20	88
124	ANJUNA	FEMALE	21	75
120	RAM	MALE	21	85
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
127	MANISHA	FEMALE	25	85
125	SONAL	FEMALE	-	89
126	ROHAN	MALE	-	95

Group By

Q. Write a query to display AGE and the least marks scored by each UNIQUE aged student from the table STUDENT.

Autocommit Display 10

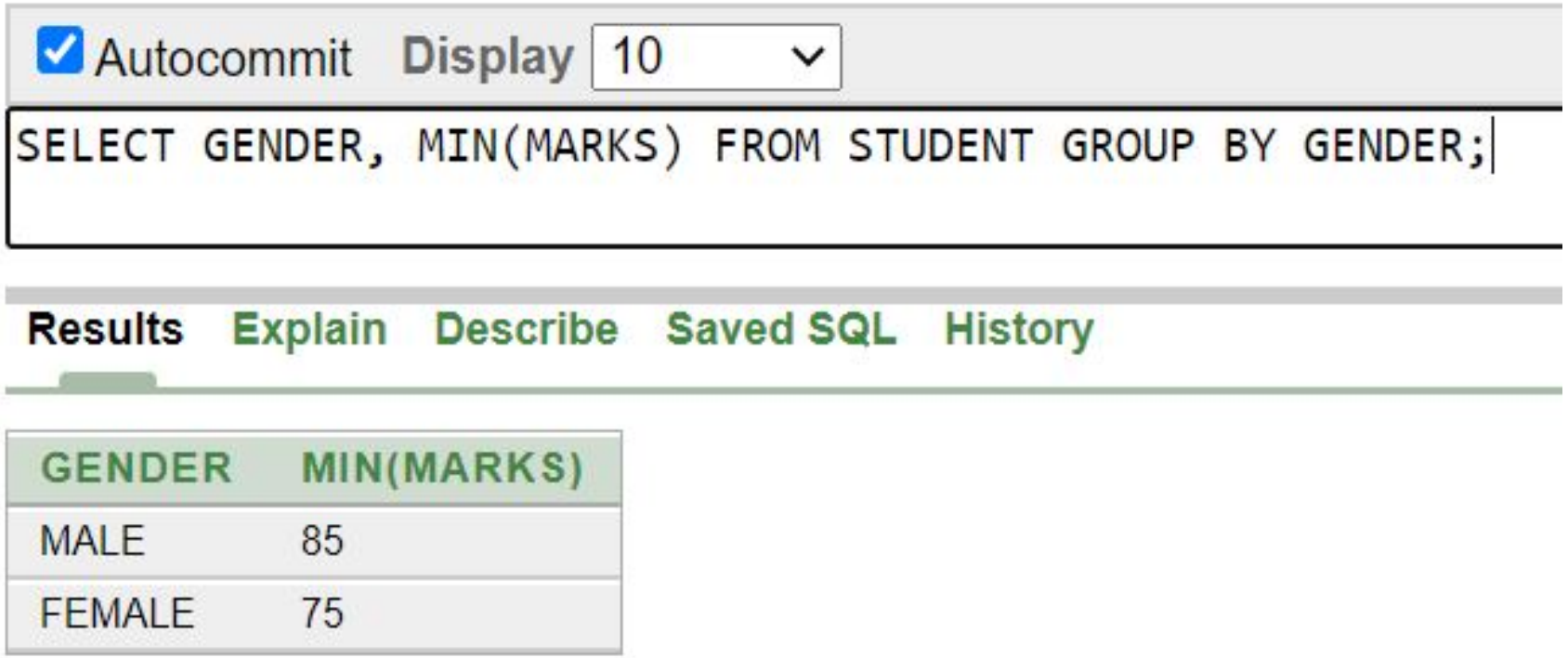
```
SELECT AGE, MIN(MARKS) FROM STUDENT GROUP BY AGE;
```

Results Explain Describe Saved SQL History

AGE	MIN(MARKS)
22	87
25	85
-	89
21	75
20	88

Group By

Q. Write a query to display GENDER and the least marks scored by each UNIQUE gender student from the table STUDENT.



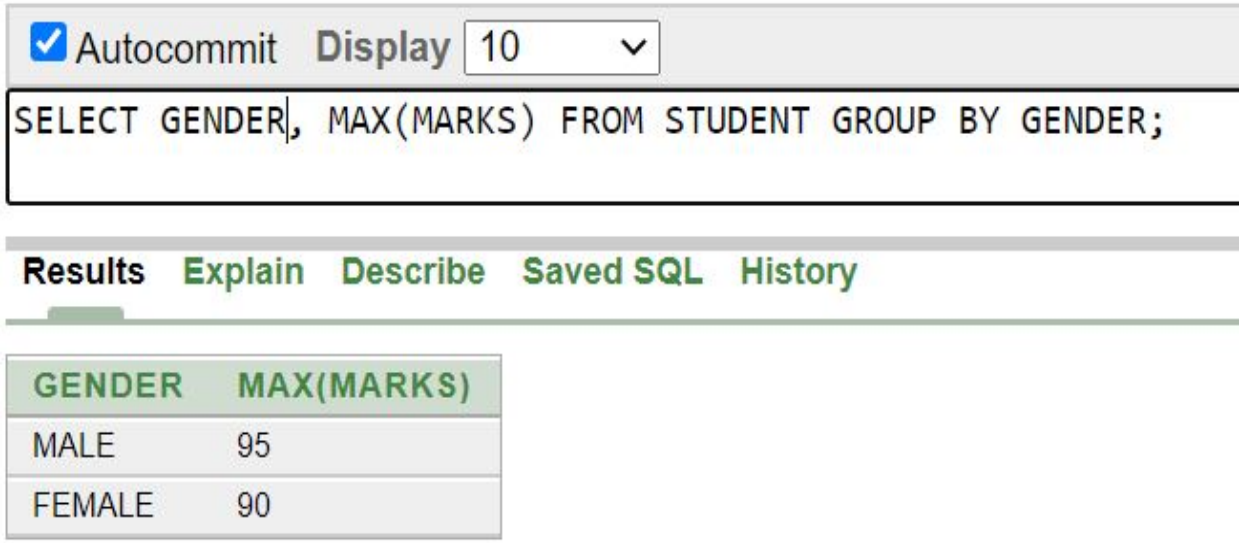
GENDER	MIN(MARKS)
MALE	85
FEMALE	75

Group By

The SQL GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.

The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.



GENDER	MAX(MARKS)
MALE	95
FEMALE	90

Group By

Q. Write a query to display AGE and count of the age for all the students whose age is equal to 21 from the table STUDENT.

Autocommit Display 10

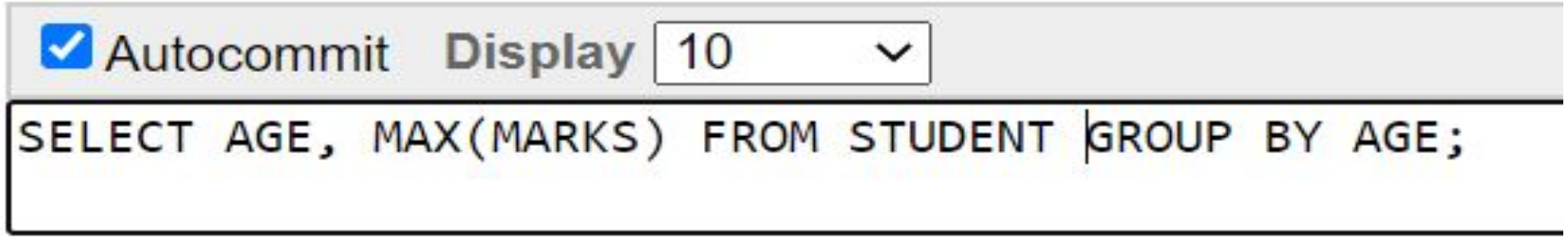
```
SELECT AGE, COUNT(AGE) FROM STUDENT WHERE AGE=21 GROUP BY AGE;
```

Results Explain Describe Saved SQL History

AGE	COUNT(AGE)
21	2

Group By

Q. Write a query to display AGE and the MAXIMUM marks scored by each UNIQUE aged student from the table STUDENT.

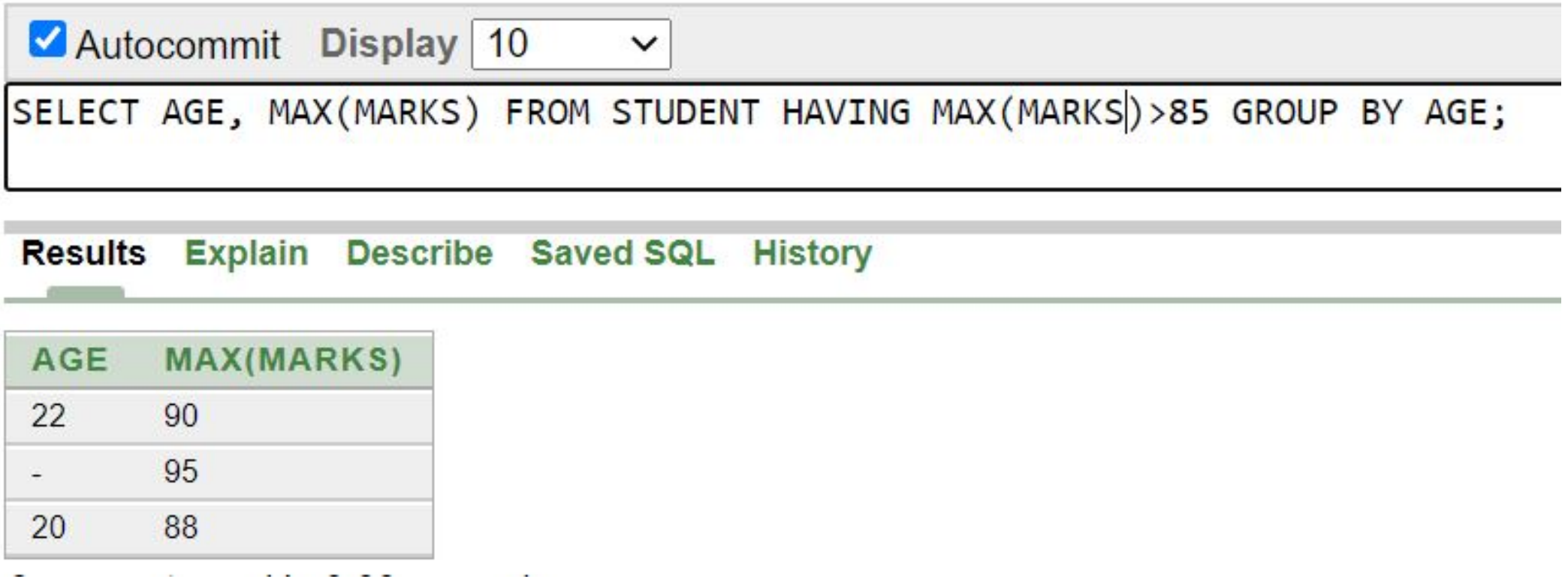


Results | Explain | Describe | Saved SQL | History

AGE	MAX(MARKS)
22	90
25	85
-	95
21	85
20	88

Group By

Q. Write a query to display AGE and the MAXIMUM marks scored by each UNIQUE aged student whose maximum marks is greater than 85 from the table STUDENT.



Having

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions

HAVING Syntax

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
HAVING *condition*
ORDER BY *column_name(s);*

Having

Q6. Write a query to display AGE and the MAXIMUM marks scored by all student whose age is greater than 19 and having maximum marks greater than 85 for each age, while displaying the data in ascending order with respect to age.

Autocommit Display 10

```
SELECT AGE, MAX(MARKS) FROM STUDENT WHERE AGE>19 GROUP BY AGE HAVING MAX(MARKS)>85 ORDER BY AGE;
```

Results Explain Describe Saved SQL History

AGE	MAX(MARKS)
20	88
22	90

Nested Query

Nested Query

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75
127	MANISHA	FEMALE	25	85
125	SONAL	FEMALE	-	89
126	ROHAN	MALE	-	95

Nested Query

Q. Write a query to display USN, NAME and GENDER of all students whose marks is more than RAVI's marks from the table STUDENT.

Autocommit Display 10 Save

```
SELECT USN, NAME, GENDER FROM STUDENT WHERE MARKS>(SELECT MARKS FROM STUDENT WHERE NAME='RAVI');
```

Results Explain Describe Saved SQL History

USN	NAME	GENDER
122	KAVITHA	FEMALE
125	SONAL	FEMALE
126	ROHAN	MALE

Nested Query

Autocommit Display 10 Save

```
SELECT USN, NAME, GENDER FROM STUDENT WHERE MARKS>(SELECT MARKS FROM STUDENT WHERE NAME='RAVI');
```

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75
127	MANISHA	FEMALE	25	85
125	SONAL	FEMALE	-	89
126	ROHAN	MALE	-	95

SELECT MARKS FROM STUDENT WHERE NAME= 'RAVI';

MARKS = 88

SELECT USN, NAME, GENDER FROM STUDENT WHERE MARKS > 88;

Results Explain Describe Saved SQL History

USN	NAME	GENDER
122	KAVITHA	FEMALE
125	SONAL	FEMALE
126	ROHAN	MALE

Nested Query

Q. Write a query to display the USN and NAME for all the student whose age is same as ARJUN's age from the table STUDENT.

Autocommit Display 10

```
SELECT USN, NAME FROM STUDENT WHERE AGE=(SELECT AGE FROM STUDENT WHERE NAME='ARJUN');
```

Results Explain Describe Saved SQL History

USN	NAME
122	KAVITHA
123	ARJUN

Nested Query

Autocommit Display 10

```
SELECT USN, NAME FROM STUDENT WHERE AGE=(SELECT AGE FROM STUDENT WHERE NAME='ARJUN');
```

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75
127	MANISHA	FEMALE	25	85
125	SONAL	FEMALE	-	89
126	ROHAN	MALE	-	95

SELECT AGE FROM STUDENT WHERE NAME= 'ARJUN';

AGE = 22

SELECT USN, NAME FROM STUDENT WHERE AGE = 22;

Results Explain Describe Saved SQL History

USN	NAME
122	KAVITHA
123	ARJUN

Nested Query

A Subquery is a query within another SQL query and embedded within the WHERE clause.

A Subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Properties of subqueries

1.Subqueries must be enclosed within parentheses.

2.A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.

3.An ORDER BY cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY can be used to perform the same function as the ORDER BY in a subquery.

4.Subqueries that return more than one row can only be used with multiple value operators, such as the IN operator.

Nested Query

Q. Write a query to display NAME, AGE whose age is equal to student named RAM and marks is greater than student named ANJUNA from the table STUDENT.

Autocommit Display 10 Save Run

```
SELECT NAME, AGE FROM STUDENT WHERE AGE=(SELECT AGE FROM STUDENT WHERE NAME='RAM')AND MARKS>(SELECT
MARKS FROM STUDENT WHERE NAME='ANJUNA');
```

Results Explain Describe Saved SQL History

NAME	AGE
RAM	21

Nested Query

Autocommit Display 10 Save Run

```
SELECT NAME, AGE FROM STUDENT WHERE AGE=(SELECT AGE FROM STUDENT WHERE NAME='RAM')AND MARKS>(SELECT
MARKS FROM STUDENT WHERE NAME='ANJUNA');
```

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75
127	MANISHA	FEMALE	25	85
125	SONAL	FEMALE	-	89
126	ROHAN	MALE	-	95

SELECT MARKS FROM STUDENT WHERE NAME= 'ANJUNA';

MARKS = 75

SELECT AGE FROM STUDENT WHERE NAME = 'RAM';

AGE = 21

SELECT NAME, AGE FROM STUDENT WHERE AGE = 21 AND MARKS > 75;

NAME	AGE
RAM	21

Nested Query

Autocommit Display 50

```
SELECT * FROM STU_DEPT;
```

Results Explain Describe Saved SQL History

DEPT_ID	USN	COURSE	CITY	PINCODE
21	125	Operating system	Bangalore	560076
22	123	Management system	Pune	426520
23	124	Engineer system	Mumbai	760548
24	122	Microprocessor	Bangalore	560076
25	121	Microcontroller	Bangalore	560076
26	126	Semiconductor	Bangalore	560076

```
CREATE TABLE STU_DEPT(
DEPT_ID NUMBER,
USN NUMBER,
COURSE VARCHAR2(30),
CITY VARCHAR2(30),
PINCODE NUMBER
);
```

```
INSERT INTO STU_DEPT VALUES (21,125,‘Operating system','Bangalore',560076);
INSERT INTO STU_DEPT VALUES (22,123,‘Management system',’Pune',426520);
INSERT INTO STU_DEPT VALUES (23,124,‘Engineer system',‘Mumbai',760548);
INSERT INTO STU_DEPT VALUES (24,122,‘Microprocessor','Bangalore',560076);
INSERT INTO STU_DEPT VALUES (25,121,‘Microcontroller','Bangalore',560076);
INSERT INTO STU_DEPT VALUES (26,126,'Semiconductor','Bangalore',560076);
```

Nested Query

Q. Write a query to display the DEPT_ID of all the students from STU-DEPT table whose USN in STUDENT table is equal to USN in STU-DEPT.

Autocommit Display 50

```
SELECT DEPT_ID FROM STU_DEPT WHERE USN IN (SELECT USN FROM STUDENT);
```

Results Explain Describe Saved SQL History

DEPT_ID
25
24
22
23
21
26

Nested Query

Autocommit Display 50

```
SELECT DEPT_ID FROM STU_DEPT WHERE USN IN (SELECT USN FROM STUDENT);
```

USN	NAME	GENDER	AGE	MARKS
120	RAM	MALE	21	85
121	RAVI	MALE	20	88
122	KAVITHA	FEMALE	22	90
123	ARJUN	MALE	22	87
124	ANJUNA	FEMALE	21	75
127	MANISHA	FEMALE	25	85
125	SONAL	FEMALE	-	89
126	ROHAN	MALE	-	95

DEPT_ID	USN	COURSE	CITY	PINCODE
21	125	Operating system	Bangalore	560076
22	123	Management system	Pune	426520
23	124	Engineer system	Mumbai	760548
24	122	Microprocessor	Bangalore	560076
25	121	Microcontroller	Bangalore	560076
26	126	Semiconductor	Bangalore	560076

SELECT USN FROM STUDENT;

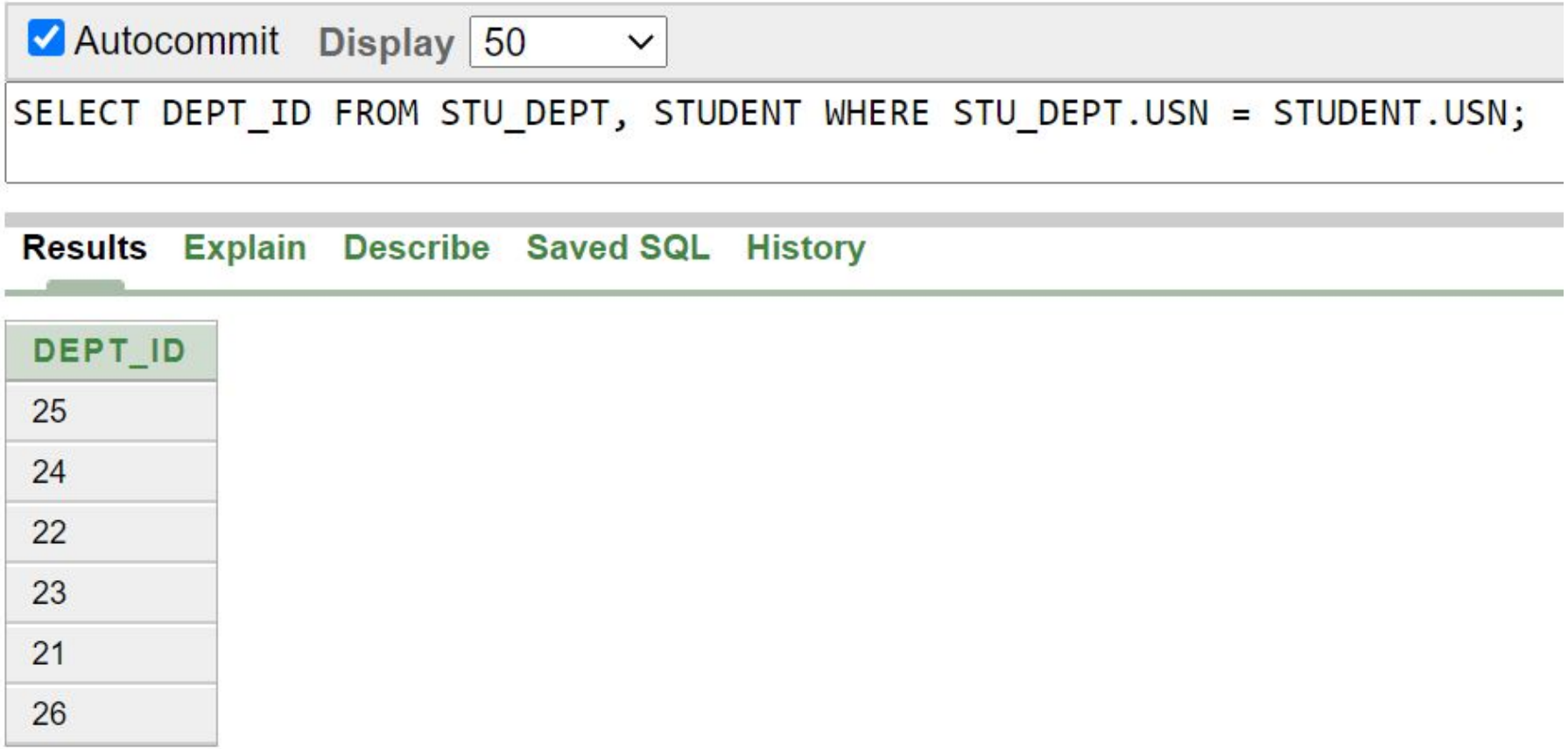
120, 121, 122, 123, 124, 127, 125, 126

SELECT DEPT_ID FROM STU_DEPT WHERE USN IN (120, 121, 122, 123, 124, 127, 125, 126);

25, 24, 22, 23, 21, 26

Nested Query

Q. Write a query to display the DEPT_ID of all the students from STU-DEPT table whose USN in STUDENT table is equal to USN in STU-DEPT.



Joins

Introduction

Consider Two Tables

EMP_ID	NAME	GENDER	AGE	SALARY
206	AMAN	MALE	24	350000
208	TANISHA	FEMALE	27	460000
203	HARPREET	FEMALE	29	890000
201	RAM	MALE	23	358000
205	VISHAL	MALE	25	560000
204	SHUBHAM	MALE	29	679000
207	ROHAN	MALE	28	700000
210	PRIYANKA	FEMALE	27	650000

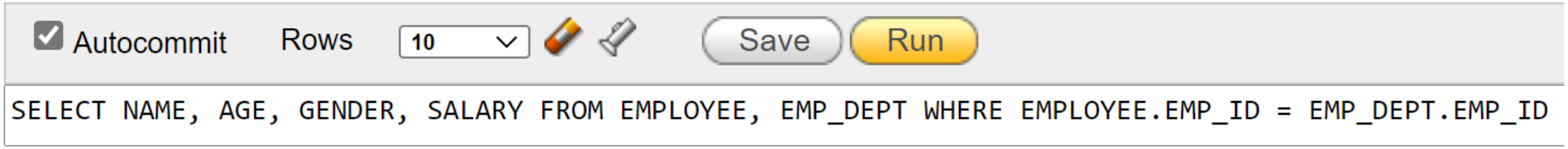
EMPLOYEE

DEPT_ID	EMP_ID	DEPARTMENT	CITY	PINCODE
21	205	HR	BANGALORE	560076
23	203	DEVELOPER	DELHI	879009
23	206	DEVELOPER	PUNE	476501
22	207	SALES	BANGALORE	568077
21	201	HR	PUNE	564487
22	210	SALES	MUMBAI	656009
24	209	ANALYST	AHEMDABAD	678879
25	200	ANALYST	PUNE	238977

EMP_DEPT

Question

Write a query to display NAME, GENDER, AGE and MARKS of all employees whose EMP_ID in the EMPLOYEE table is equal to the EMP_ID in the EMP_DEPT table.



NAME	AGE	GENDER	SALARY
VISHAL	25	MALE	560000
HARPREET	29	FEMALE	890000
AMAN	24	MALE	350000
ROHAN	28	MALE	700000
RAM	23	MALE	358000
PRIYANKA	27	FEMALE	650000

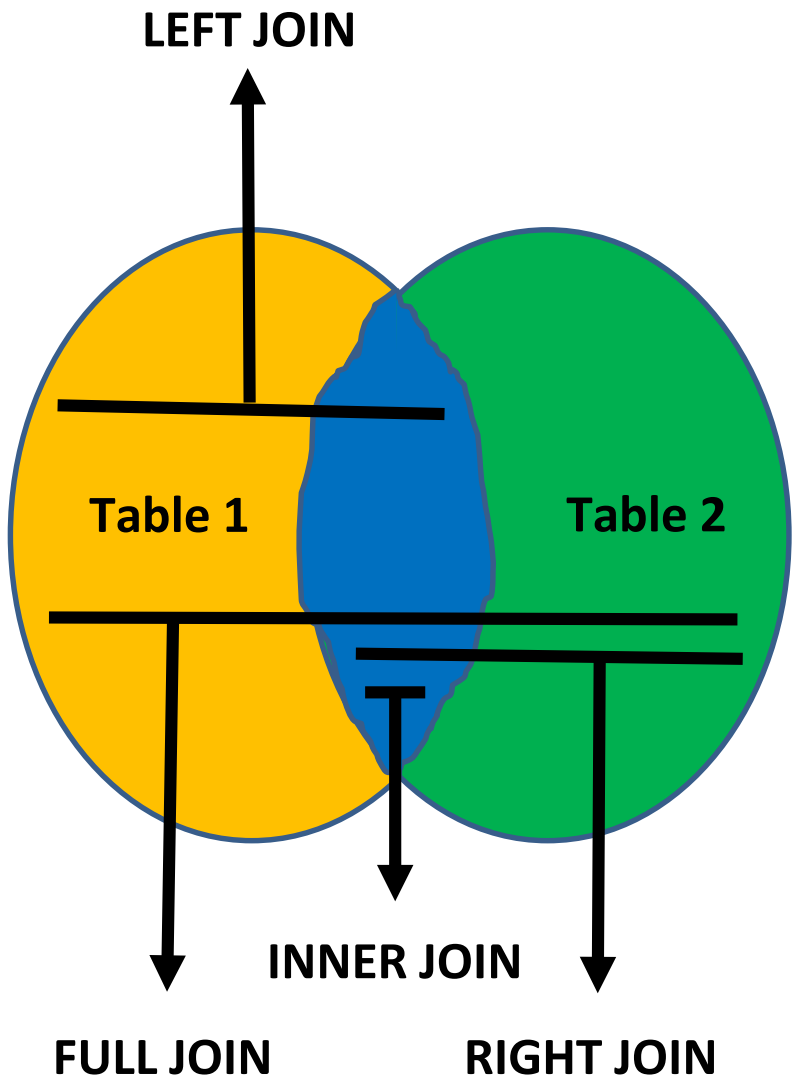
What is JOINS in SQL?

Joins

A JOIN is a means for combining fields from two tables (or more) by using values common to each.

Types of JOINS:-

1. Inner Join
2. Left Join
3. Right Join
4. Full Join
5. Natural Join
6. Cross Join



INNER JOIN: Returns all rows when there is at least one match in BOTH tables.

LEFT JOIN: Return all rows from the left table, and the matched rows from the right table.

RIGHT JOIN: Return all rows from the right table, and the matched rows from the left table.

FULL JOIN: Return all rows when there is a match in ONE of the tables.

INNER JOIN

Join, Inner Join

Write a query to display all the details from EMPLOYEE and EMP_DEPT where the value of EMP_ID in EMPLOYEE is equal to the value Of EMP_ID in EMP_DEPT.

```
SELECT * FROM EMPLOYEE E1 JOIN EMP_DEPT E2 ON E1.EMP_ID = E2.EMP_ID
```

Results Explain Describe Saved SQL History

EMP_ID	NAME	GENDER	AGE	SALARY	DEPT_ID	EMP_ID	DEPARTMENT	CITY	PINCODE
205	VISHAL	MALE	25	560000	21	205	HR	BANGALORE	560076
203	HARPREET	FEMALE	29	890000	23	203	DEVELOPER	DELHI	879009
206	AMAN	MALE	24	350000	23	206	DEVELOPER	PUNE	476501
207	ROHAN	MALE	28	700000	22	207	SALES	BANGALORE	568077
201	RAM	MALE	23	358000	21	201	HR	PUNE	564487
210	PRIYANKA	FEMALE	27	650000	22	210	SALES	MUMBAI	656009

LEFT JOIN

Left Join, Left Outer Join

Write a query to display all the details from EMPLOYEE and common details from EMP_DEPT by using left join keyword.

```
SELECT * FROM EMPLOYEE E1 LEFT OUTER JOIN EMP_DEPT E2 ON E1.EMP_ID = E2.EMP_ID
```

Results Explain Describe Saved SQL History

EMP_ID	NAME	GENDER	AGE	SALARY	DEPT_ID	EMP_ID	DEPARTMENT	CITY	PINCODE
205	VISHAL	MALE	25	560000	21	205	HR	BANGALORE	560076
203	HARPREET	FEMALE	29	890000	23	203	DEVELOPER	DELHI	879009
206	AMAN	MALE	24	350000	23	206	DEVELOPER	PUNE	476501
207	ROHAN	MALE	28	700000	22	207	SALES	BANGALORE	568077
201	RAM	MALE	23	358000	21	201	HR	PUNE	564487
210	PRIYANKA	FEMALE	27	650000	22	210	SALES	MUMBAI	656009
204	SHUBHAM	MALE	29	679000	-	-	-	-	-
208	TANISHA	FEMALE	27	460000	-	-	-	-	-

RIGHT JOIN

Right Join, Right Outer Join

Write a query to display common details from EMPLOYEE and all the details from EMP_DEPT by using right join keyword.

```
SELECT * FROM EMPLOYEE E1 RIGHT OUTER JOIN EMP_DEPT E2 ON E1.EMP_ID = E2.EMP_ID
```

Results | Explain | Describe | Saved SQL | History

EMP_ID	NAME	GENDER	AGE	SALARY	DEPT_ID	EMP_ID	DEPARTMENT	CITY	PINCODE
206	AMAN	MALE	24	350000	23	206	DEVELOPER	PUNE	476501
203	HARPREET	FEMALE	29	890000	23	203	DEVELOPER	DELHI	879009
201	RAM	MALE	23	358000	21	201	HR	PUNE	564487
205	VISHAL	MALE	25	560000	21	205	HR	BANGALORE	560076
207	ROHAN	MALE	28	700000	22	207	SALES	BANGALORE	568077
210	PRIYANKA	FEMALE	27	650000	22	210	SALES	MUMBAI	656009
-	-	-	-	-	25	200	ANALYST	PUNE	238977
-	-	-	-	-	24	209	ANALYST	AHEMDABAD	678879

FULL JOIN

Full Join, Full Outer Join

Write a Query to display all the details from EMPLOYEE and EMP_DEPT by using full join keyword.

```
SELECT * FROM EMPLOYEE E1 FULL OUTER JOIN EMP_DEPT E2 ON E1.EMP_ID = E2.EMP_ID
```

Results | Explain | Describe | Saved SQL | History

EMP_ID	NAME	GENDER	AGE	SALARY	DEPT_ID	EMP_ID	DEPARTMENT	CITY	PINCODE
205	VISHAL	MALE	25	560000	21	205	HR	BANGALORE	560076
203	HARPREET	FEMALE	29	890000	23	203	DEVELOPER	DELHI	879009
206	AMAN	MALE	24	350000	23	206	DEVELOPER	PUNE	476501
207	ROHAN	MALE	28	700000	22	207	SALES	BANGALORE	568077
201	RAM	MALE	23	358000	21	201	HR	PUNE	564487
210	PRIYANKA	FEMALE	27	650000	22	210	SALES	MUMBAI	656009
-	-	-	-	-	24	209	ANALYST	AHEMDABAD	678879
-	-	-	-	-	25	200	ANALYST	PUNE	238977
204	SHUBHAM	MALE	29	679000	-	-	-	-	-
208	TANISHA	FEMALE	27	460000	-	-	-	-	-

NATURAL JOIN

Natural Join

A NATURAL JOIN is a JOIN operation that creates an implicit join clause for you based on the common columns in the two tables being joined.

A NATURAL JOIN can be an INNER join, a LEFT join, or a RIGHT join. The default is INNER join.

Natural Join

Write a query to display all the details from EMPLOYEE and EMP_DEPT where the value of EMP_ID in EMPLOYEE is equal to the value of EMP_ID in EMP_DEPT by using natural join keyword.

```
SELECT * FROM EMPLOYEE NATURAL FULL JOIN EMP_DEPT
```

Results | Explain | Describe | Saved SQL | History

EMP_ID	NAME	GENDER	AGE	SALARY	DEPT_ID	DEPARTMENT	CITY	PINCODE
205	VISHAL	MALE	25	560000	21	HR	BANGALORE	560076
203	HARPREET	FEMALE	29	890000	23	DEVELOPER	DELHI	879009
206	AMAN	MALE	24	350000	23	DEVELOPER	PUNE	476501
207	ROHAN	MALE	28	700000	22	SALES	BANGALORE	568077
201	RAM	MALE	23	358000	21	HR	PUNE	564487
210	PRIYANKA	FEMALE	27	650000	22	SALES	MUMBAI	656009
209	-	-	-	-	24	ANALYST	AHEMDABAD	678879
200	-	-	-	-	25	ANALYST	PUNE	238977
204	SHUBHAM	MALE	29	679000	-	-	-	-
208	TANISHA	FEMALE	27	460000	-	-	-	-

CROSS JOIN

Cross Join

The CARTESIAN JOIN or CROSS JOIN returns the Cartesian product of the sets of records from the two or more joined tables.

EMP_ID	NAME	GENDER	AGE	SALARY
206	AMAN	MALE	24	350000
208	TANISHA	FEMALE	27	460000
203	HARPREET	FEMALE	29	890000
201	RAM	MALE	23	358000
205	VISHAL	MALE	25	560000
204	SHUBHAM	MALE	29	679000
207	ROHAN	MALE	28	700000
210	PRIYANKA	FEMALE	27	650000

EMPLOYEE

DEPT_ID	EMP_ID	DEPARTMENT	CITY	PINCODE
21	205	HR	BANGALORE	560076
23	203	DEVELOPER	DELHI	879009
23	206	DEVELOPER	PUNE	476501
22	207	SALES	BANGALORE	568077
21	201	HR	PUNE	564487
22	210	SALES	MUMBAI	656009
24	209	ANALYST	AHEMDABAD	678879
25	200	ANALYST	PUNE	238977

EMP_DEPT

Cross Join

EMP_ID	NAME	GENDER	AGE	SALARY	DEPT_ID	EMP_ID	DEPARTMENT	CITY	PINCODE
206	AMAN	MALE	24	350000	21	205	HR	BANGALORE	560076
206	AMAN	MALE	24	350000	23	203	DEVELOPER	DELHI	879009
206	AMAN	MALE	24	350000	23	206	DEVELOPER	PUNE	476501
206	AMAN	MALE	24	350000	22	207	SALES	BANGALORE	568077
206	AMAN	MALE	24	350000	21	201	HR	PUNE	564487
206	AMAN	MALE	24	350000	22	210	SALES	MUMBAI	656009
206	AMAN	MALE	24	350000	24	209	ANALYST	AHEMDABAD	678879
206	AMAN	MALE	24	350000	25	200	ANALYST	PUNE	238977
208	TANISHA	FEMALE	27	460000	21	205	HR	BANGALORE	560076
208	TANISHA	FEMALE	27	460000	23	203	DEVELOPER	DELHI	879009
208	TANISHA	FEMALE	27	460000	23	206	DEVELOPER	PUNE	476501
208	TANISHA	FEMALE	27	460000	22	207	SALES	BANGALORE	568077
208	TANISHA	FEMALE	27	460000	21	201	HR	PUNE	564487
208	TANISHA	FEMALE	27	460000	22	210	SALES	MUMBAI	656009
208	TANISHA	FEMALE	27	460000	24	209	ANALYST	AHEMDABAD	678879
208	TANISHA	FEMALE	27	460000	25	200	ANALYST	PUNE	238977
203	HARPREET	FEMALE	29	890000	21	205	HR	BANGALORE	560076
203	HARPREET	FEMALE	29	890000	23	203	DEVELOPER	DELHI	879009

SELF JOIN

Self Join

The SQL SELF JOIN is used to join a table to itself as if the table were two tables.

To join a table itself means that each row of the table is combined with itself and with every other row of the table.

Syntax:-

```
SELECT a.column_name, b.column_name...

FROM table1 a, table1 b

WHERE a.common_field = b.common_field;
```

EMP_ID	NAME	GENDER	AGE	SALARY
206	AMAN	MALE	24	350000
208	TANISHA	FEMALE	27	460000
203	HARPREET	FEMALE	29	890000
201	RAM	MALE	23	358000
205	VISHAL	MALE	25	560000
204	SHUBHAM	MALE	29	679000
207	ROHAN	MALE	28	700000
210	PRIYANKA	FEMALE	27	650000

Self Join

SELECT E1.EMP_ID, E2.NAME, E1.SALARY FROM EMPLOYEE E1, EMPLOYEE E2 WHERE E1.SALARY < E2.SALARY

EMP_ID	NAME	GENDER	AGE	SALARY
206	AMAN	MALE	24	350000
208	TANISHA	FEMALE	27	460000
203	HARPREET	FEMALE	29	890000
201	RAM	MALE	23	358000
205	VISHAL	MALE	25	560000
204	SHUBHAM	MALE	29	679000
207	ROHAN	MALE	28	700000
210	PRIYANKA	FEMALE	27	650000

EMPLOYEE E1

EMP_ID	NAME	SALARY
206	TANISHA	350000
206	HARPREET	350000
206	RAM	350000
206	VISHAL	350000
206	SHUBHAM	350000
206	ROHAN	350000
206	PRIYANKA	350000
208	HARPREET	460000
208	VISHAL	460000
208	SHUBHAM	460000
208	ROHAN	460000
208	PRIYANKA	460000

201	TANISHA	358000
201	HARPREET	358000
201	VISHAL	358000
201	SHUBHAM	358000
201	ROHAN	358000
201	PRIYANKA	358000
205	HARPREET	560000
205	SHUBHAM	560000
205	ROHAN	560000
205	PRIYANKA	560000
204	HARPREET	679000
204	ROHAN	679000
207	HARPREET	700000

EMPLOYEE E2

Select From Multiple Tables without Join

Select from multiple tables without Join

Write a query to display all the details of employees whose EMP_ID in the EMPLOYEE table is equal to the EMP_ID in the EMP_DEPT table without using Joins.

```
SELECT * FROM EMPLOYEE E1, EMP_DEPT E2 WHERE E1.EMP_ID = E2.EMP_ID
```

Results Explain Describe Saved SQL History

EMP_ID	NAME	GENDER	AGE	SALARY	DEPT_ID	EMP_ID	DEPARTMENT	CITY	PINCODE
205	VISHAL	MALE	25	560000	21	205	HR	BANGALORE	560076
203	HARPREET	FEMALE	29	890000	23	203	DEVELOPER	DELHI	879009
206	AMAN	MALE	24	350000	23	206	DEVELOPER	PUNE	476501
207	ROHAN	MALE	28	700000	22	207	SALES	BANGALORE	568077
201	RAM	MALE	23	358000	21	201	HR	PUNE	564487
210	PRIYANKA	FEMALE	27	650000	22	210	SALES	MUMBAI	656009

Join Multiple Tables with Conditions

Join Multiple Tables with Conditions

Write a query to display salary, employee id from EMPLOYEE table and department, city from EMP_DEPT table whose EMP_ID in the EMPLOYEE table is equal to the EMP_ID in the EMP_DEPT table and salary is greater than 400000.

```
SELECT E1.SALARY, E1.EMP_ID, E2.DEPARTMENT, E2.CITY FROM EMPLOYEE E1
INNER JOIN EMP_DEPT E2 ON E1.EMP_ID = E2.EMP_ID WHERE E1.SALARY>400000
```

Results Explain Describe Saved SQL History

SALARY	EMP_ID	DEPARTMENT	CITY
560000	205	HR	BANGALORE
890000	203	DEVELOPER	DELHI
700000	207	SALES	BANGALORE
650000	210	SALES	MUMBAI

How to Join 3 Tables in SQL?

How to Join 3 Tables in SQL?

EMP_ID	EMP_FIRSTNAME	EMP_LASTNAME
1	RAM	KUMAR
2	ROHAN	DAS
3	SAURABH	GOYAL

EMPLOYEE

EMP_ID	DEPT_ID
1	2
1	3
2	1
2	2
2	3
2	3
3	1

EMP_DEPT

EMP_ID	DEPT_NAME	DEPT_HEAD_ID
1	HR	1
2	DEVELOPER	2
3	SALES	1

DEPARTMENT

How to Join 3 Tables in SQL?

SELECT EMPLOYEE.EMP_FIRSTNAME, EMPLOYEE.EMP_LASTNAME
FROM EMPLOYEE

EMP_FIRSTNAME	EMP_LASTNAME
RAM	KUMAR
ROHAN	DAS
SAURABH	GOYAL

EMP_ID	DEPT_ID
1	2
1	3
2	1
2	2
2	3
2	3
3	1

EMP_DEPT

DEPARTMENT

EMP_ID	DEPT_NAME	DEPT_HEAD_ID
1	HR	1
2	DEVELOPER	2
3	SALES	1

EMP_ID	EMP_FIRSTNAME	EMP_LASTNAME
1	RAM	KUMAR
2	ROHAN	DAS
3	SAURABH	GOYAL

EMPLOYEE

How to Join 3 Tables in SQL?

```
SELECT EMPLOYEE.EMP_FIRSTNAME, EMPLOYEE.EMP_LASTNAME
FROM EMPLOYEE
JOIN EMP_DEPT
ON EMPLOYEE.EMP_ID = EMP_DEPT.EMP_ID
```

EMP_FIRSTNAME	EMP_LASTNAME
RAM	KUMAR
RAM	KUMAR
ROHAN	DAS
ROHAN	DAS
ROHAN	DAS
ROHAN	DAS
SAURABH	GOYAL

EMP_ID	DEPT_ID
1	2
1	3
2	1
2	2
2	3
2	3
3	1

EMP_DEPT

DEPARTMENT

EMP_ID	DEPT_NAME	DEPT_HEAD_ID
1	HR	1
2	DEVELOPER	2
3	SALES	1

EMP_ID	EMP_FIRSTNAME	EMP_LASTNAME
1	RAM	KUMAR
2	ROHAN	DAS
3	SAURABH	GOYAL

EMPLOYEE

How to Join 3 Tables in SQL?

```
SELECT EMPLOYEE.EMP_FIRSTNAME, EMPLOYEE.EMP_LASTNAME,
DEPARTMENT.DEPT_NAME
FROM EMPLOYEE
JOIN EMP_DEPT
ON EMPLOYEE.EMP_ID = EMP_DEPT.EMP_ID
JOIN DEPARTMENT
ON DEPARTMENT.EMP_ID = EMP_DEPT.EMP_ID
```

EMP_FIRSTNAME	EMP_LASTNAME	DEPT_NAME
RAM	KUMAR	HR
RAM	KUMAR	HR
ROHAN	DAS	DEVELOPER
ROHAN	DAS	DEVELOPER
ROHAN	DAS	DEVELOPER
ROHAN	DAS	DEVELOPER
SAURABH	GOYAL	SALES

EMP_ID	DEPT_ID
1	2
1	3
2	1
2	2
2	3
2	3
3	1

EMP_DEPT

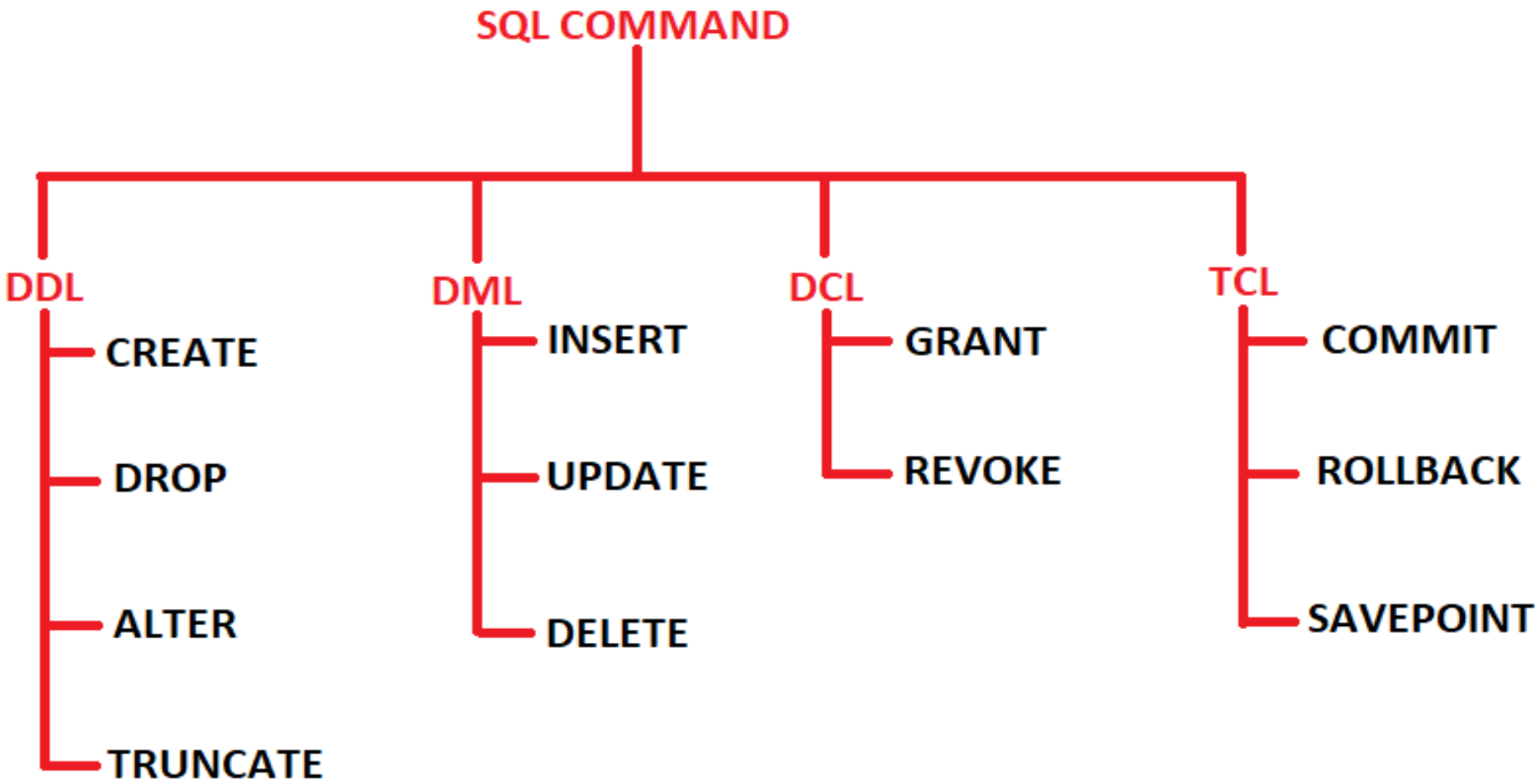
DEPARTMENT

EMP_ID	DEPT_NAME	DEPT_HEAD_ID
1	HR	1
2	DEVELOPER	2
3	SALES	1

EMP_ID	EMP_FIRSTNAME	EMP_LASTNAME
1	RAM	KUMAR
2	ROHAN	DAS
3	SAURABH	GOYAL

EMPLOYEE

SQL Commands


SQL COMMAND
DDL
CREATE
DROP
ALTER
TRUNCATE
DML
INSERT
UPDATE
DELETE
DCL
GRANT
REVOKE
TCL
COMMIT
ROLLBACK
SAVEPOINT

DDL

Data Definition – How to define Data.

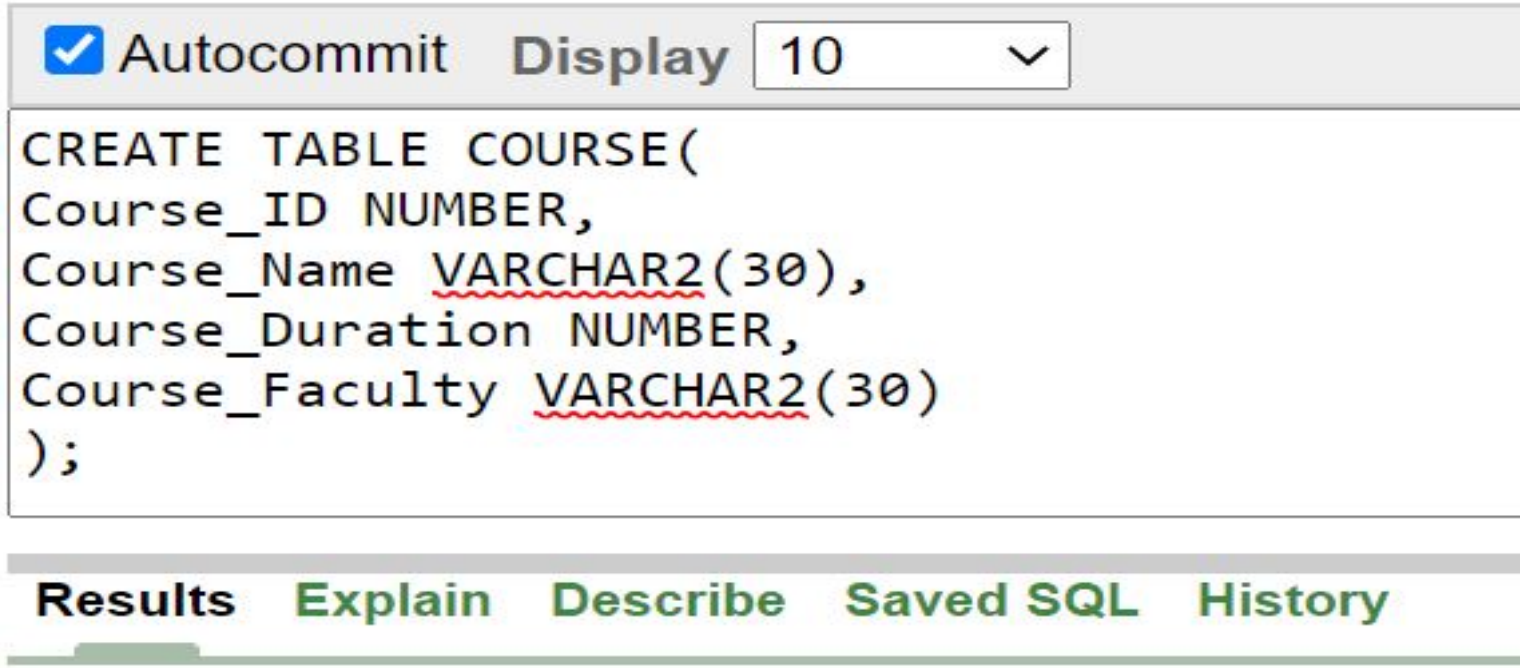
- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

CREATE: It is used to create a new table in the database.

Syntax:
CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

DDL

Q1. Write a query to create the table “COURSE” with columns such as “Course_ID”, “Course_Name”, “Course_Duration” and “Course_Faculty”.

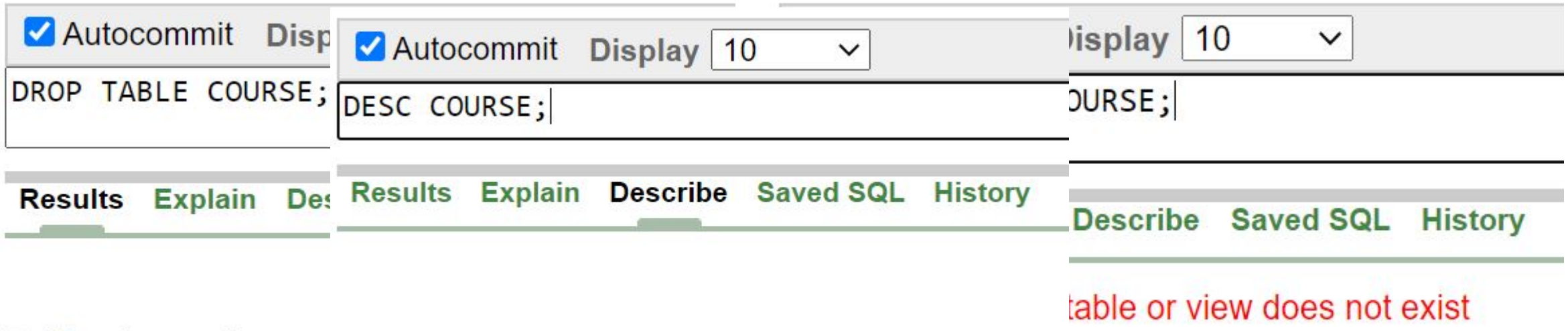


DDL

DROP: It is used to delete both the structure and record stored in the table.

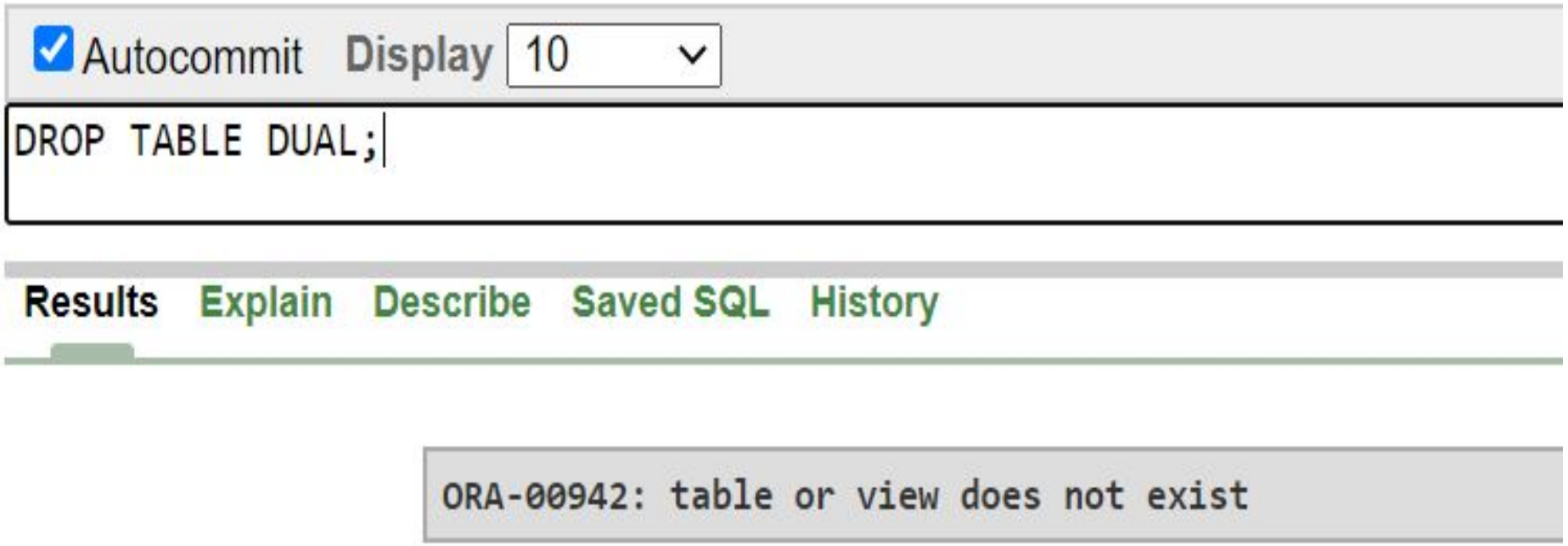
Syntax:
DROP TABLE table_name;

Q2. Write a query to delete the structure & record of the table “COURSE”.



DDL

Q3. What happens if we try to DROP dual table ?



DDL

ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

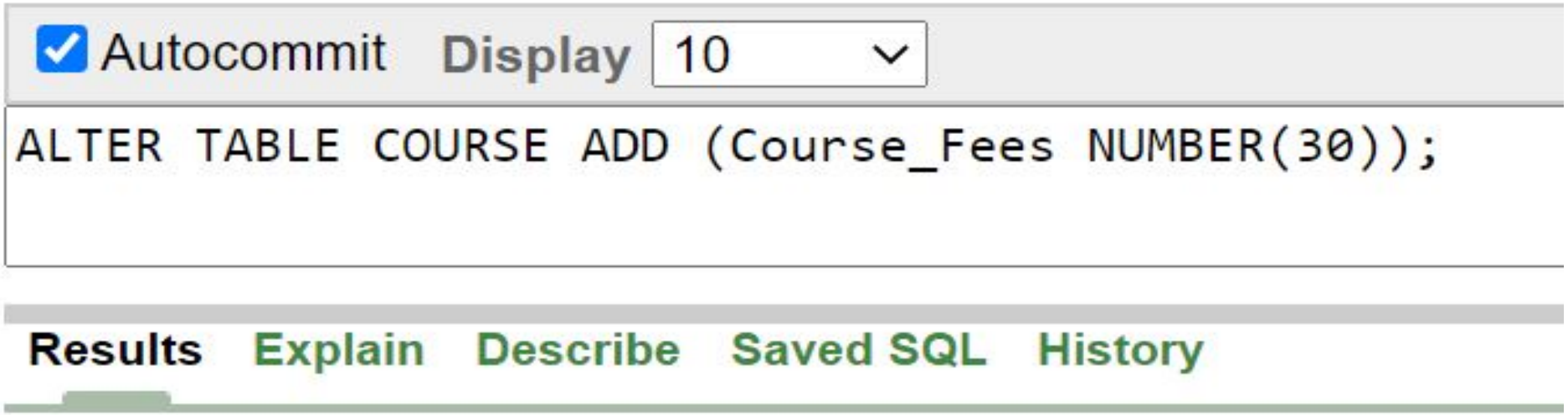
To add a new column in the table:
ALTER TABLE table_name ADD (column_name COLUMN-definition);

To modify existing column in the table:
ALTER TABLE table_name MODIFY(column_name COLUMN-definition);

To rename existing column name in the table:
ALTER TABLE table_name RENAME column old_column_name to new_column_name;

DDL

Q4. Write a query to add a new column in the table “COURSE” with columns such as “Course_Fees”.



DDL

Autocommit Display 10 Save Run

```
DESC COURSE;
```

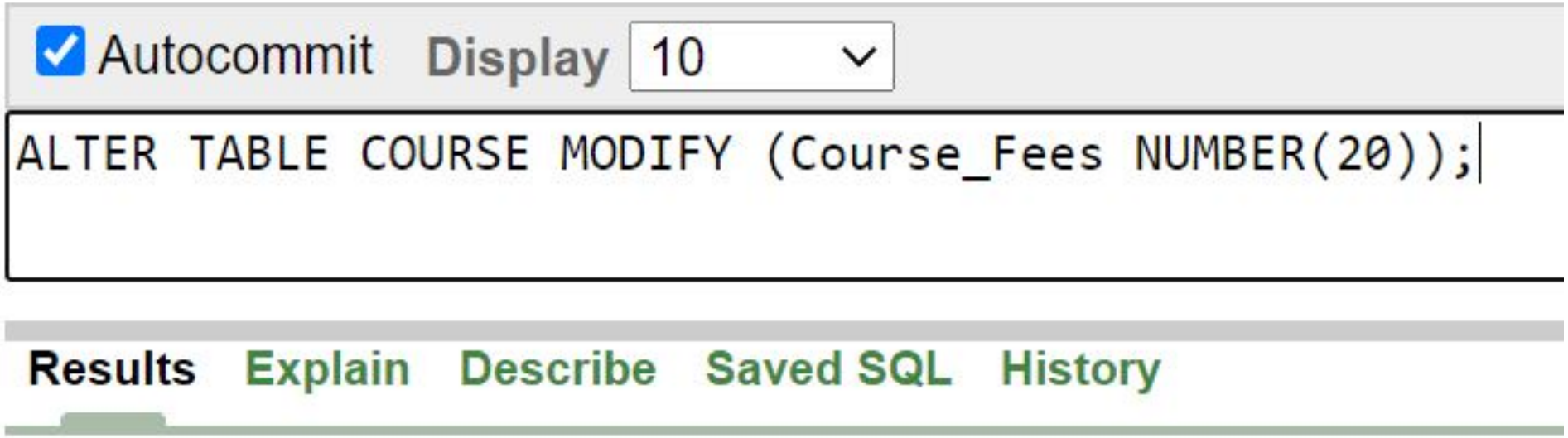
Results Explain Describe Saved SQL History

Object Type **TABLE** Object **COURSE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COURSE	COURSE_ID	Number	-	-	-	-	✓	-	-
	COURSE_NAME	Varchar2	30	-	-	-	✓	-	-
	COURSE_DURATION	Number	-	-	-	-	✓	-	-
	COURSE_FACULTY	Varchar2	30	-	-	-	✓	-	-
	COURSE_FEES	Number	-	30	0	-	✓	-	-

DDL

Q5. Write a query to modify the column “Course_Fees” in the table “COURSE” with size as 20.



DDL

Autocommit Display 10 Save Run

```
DESC COURSE;
```

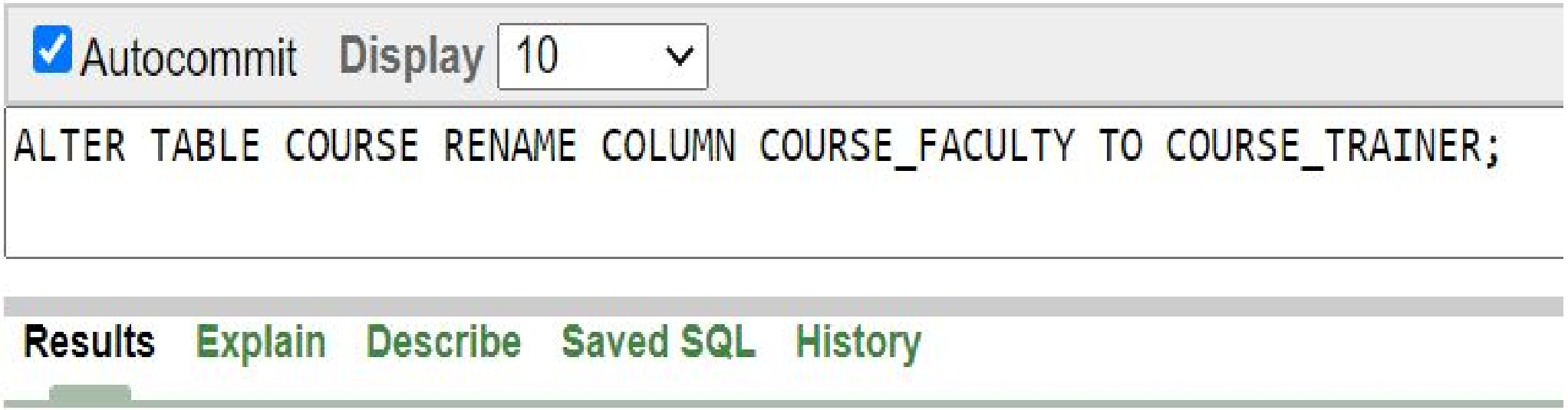
Results Explain Describe Saved SQL History

Object Type **TABLE** Object **COURSE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COURSE	COURSE_ID	Number	-	-	-	-	✓	-	-
	COURSE_NAME	Varchar2	30	-	-	-	✓	-	-
	COURSE_DURATION	Number	-	-	-	-	✓	-	-
	COURSE_FACULTY	Varchar2	30	-	-	-	✓	-	-
	COURSE_FEES	Number	-	20	0	-	✓	-	-

DDL

Q6. Write a query to rename the column name “Course_Faculty” in the table “COURSE” with “Course_Trainer”.



DDL

Autocommit Display 10

```
DESC COURSE;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **COURSE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COURSE	COURSE_ID	Number	-	-	-	-	✓	-	-
	COURSE_NAME	Varchar2	30	-	-	-	✓	-	-
	COURSE_DURATION	Number	-	-	-	-	✓	-	-
	COURSE_TRAINER	Varchar2	30	-	-	-	✓	-	-
	COURSE_FEES	Number	-	-	-	-	✓	-	-

1 - 5

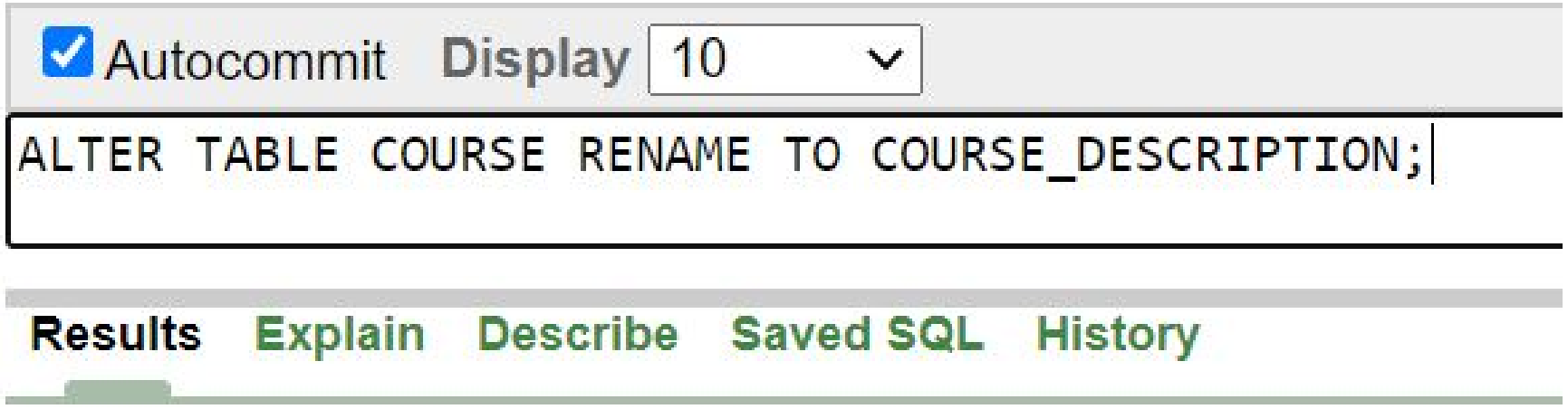
DDL

Q7. Write a query to rename the table name “Course” to “COURSE_DESCRIPTION”.

Syntax: ALTER TABLE old_table_name RENAME TO new_table_name;

OR

RENAME old_table_name TO new_table_name;



DDL

OR


Autocommit
Display
10
RENAME COURSE TO COURSE_DESCRIPTION;
Results
Explain
Describe
Saved SQL
History
Statement processed.

DDL

Autocommit Display 10

```
DESC COURSE_DESCRIPTION;
```

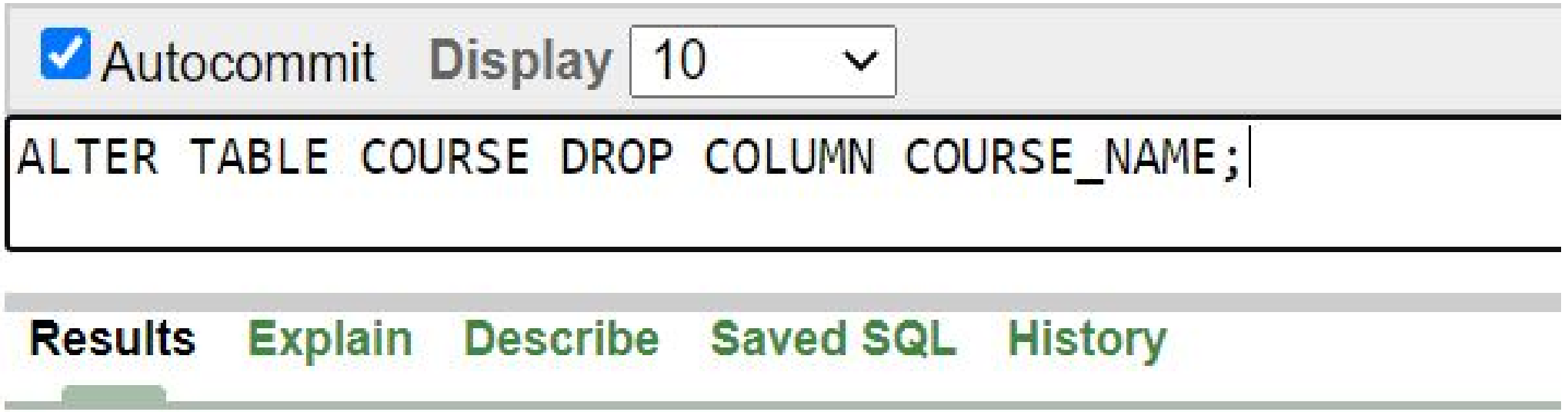
Results Explain Describe Saved SQL History

Object Type **TABLE** Object **COURSE_DESCRIPTION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
COURSE_DESCRIPTION	COURSE_ID	Number	-	-	-	-	✓
	COURSE_NAME	Varchar2	30	-	-	-	✓
	COURSE_DURATION	Number	-	-	-	-	✓
	COURSE_TRAINER	Varchar2	30	-	-	-	✓
	COURSE_FEES	Number	-	-	-	-	✓

DDL

Q8. Write a query to delete one of the column from the “COURSE” table?



DDL

Autocommit Display 10

```
DESC COURSE;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **COURSE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
COURSE	COURSE_ID	Number	-	-	-	-	✓
	COURSE_DURATION	Number	-	-	-	-	✓
	COURSE_TRAINER	Varchar2	30	-	-	-	✓
	COURSE_FEES	Number	-	-	-	-	✓

DDL

TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Syntax:
TRUNCATE TABLE table_name;

Q9. Write a query to delete all the rows, not the structure from the table "COURSE".

Save Run

```
DESC COURSE;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **COURSE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COURSE	COURSE_ID	Number	-	-	-	-	✓	-	-
	COURSE_NAME	Varchar2	30	-	-	-	✓	-	-
	COURSE_DURATION	Number	-	-	-	-	✓	-	-
	COURSE_FACULTY	Varchar2	30	-	-	-	✓	-	-
	COURSE_FEES	Number	-	20	0	-	✓	-	-

DML

Data Manipulation – How to manipulate(inserting, updating and deleting) Data.

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

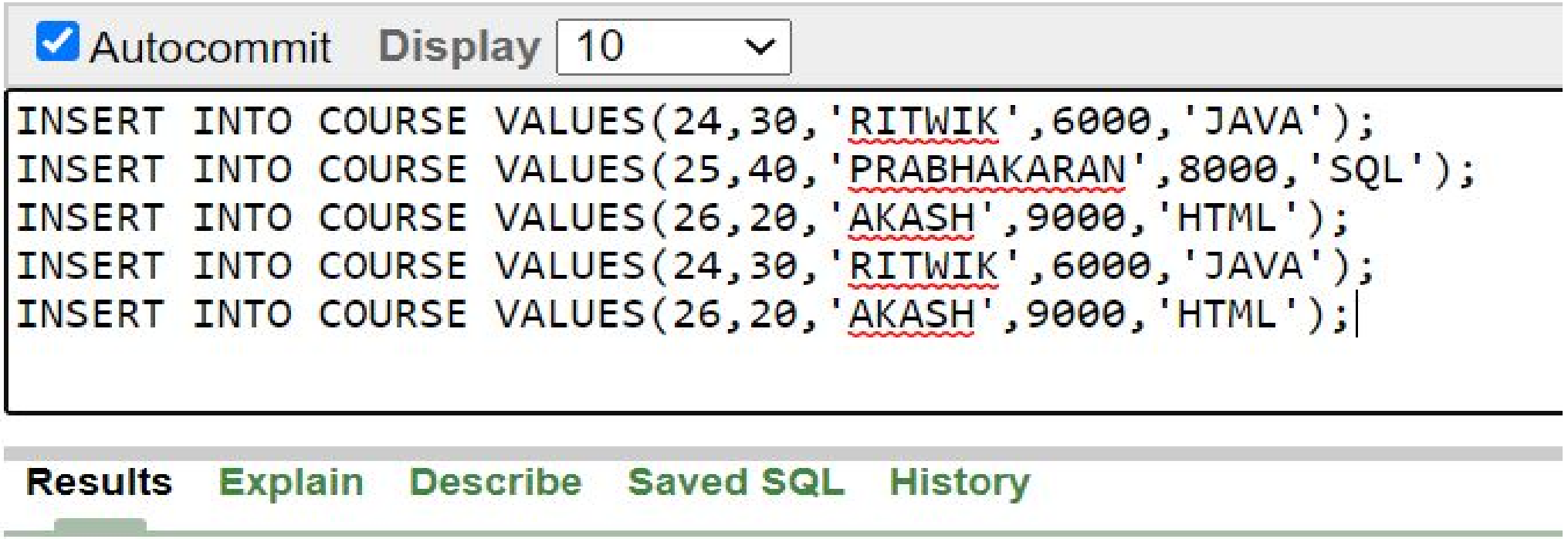
INSERT INTO TABLE_NAME (col1, col2, col3,.... col N) VALUES (value1, value2, value3, ... valueN);

Or

INSERT INTO TABLE_NAME VALUES (value1, value2, value3, ... valueN);

DML

Q10. Write a query to insert all the values to all the columns of the table “COURSE”.




```
1 row(s) inserted.
```

DML

Autocommit Display 10

```
SELECT * FROM COURSE;
```

Results Explain Describe Saved SQL History

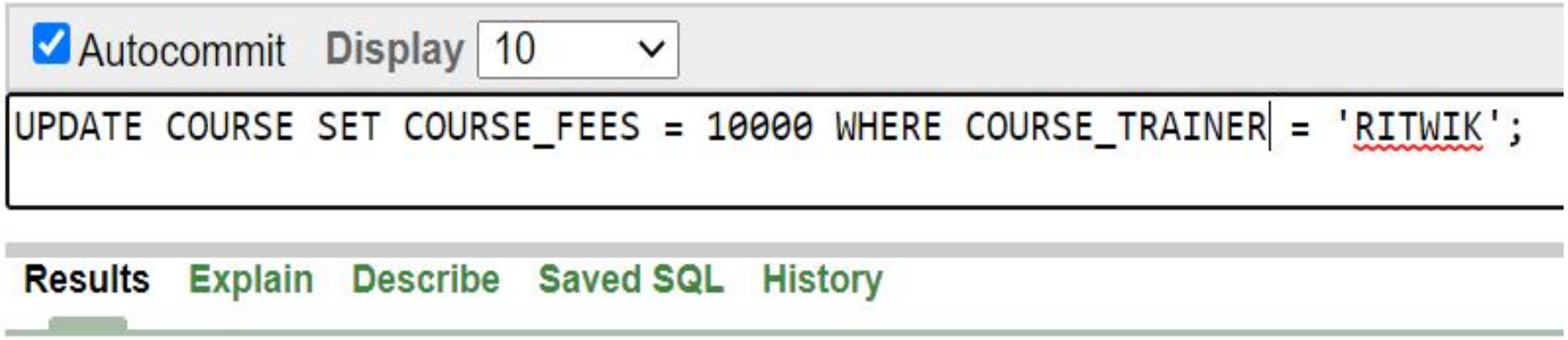
COURSE_ID	COURSE_DURATION	COURSE_TRAINER	COURSE_FEES	COURSE_NAME
24	30	RITWIK	6000	JAVA
25	40	PRABHAKARAN	8000	SQL
26	20	AKASH	9000	HTML
24	30	RITWIK	6000	JAVA
26	20	AKASH	9000	HTML

DML

UPDATE: This command is used to update or modify the value of a column in the table.

Syntax:
UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]

Q11. Write a query to update the value of course_fees from 6000 to 10000 where course_faculty is Ritwik in the table "COURSE".



DML

Autocommit Display 10

```
SELECT * FROM COURSE;
```

Results Explain Describe Saved SQL History

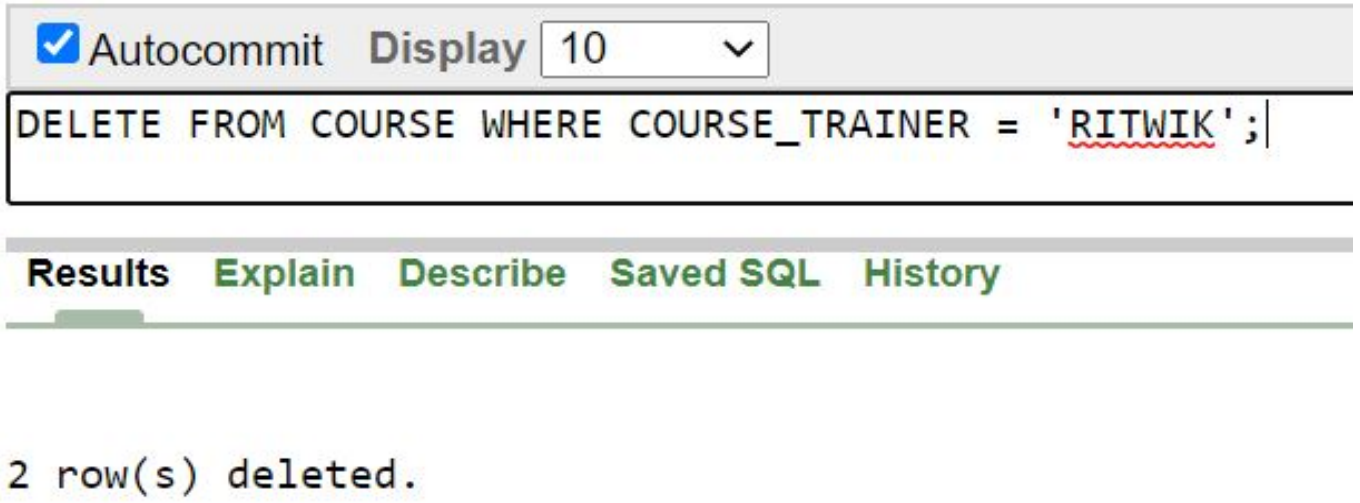
COURSE_ID	COURSE_DURATION	COURSE_TRAINER	COURSE_FEES	COURSE_NAME
24	30	RITWIK	10000	JAVA
25	40	PRABHAKARAN	8000	SQL
26	20	AKASH	9000	HTML
24	30	RITWIK	10000	JAVA
26	20	AKASH	9000	HTML

DML

DELETE: It is used to remove one or more row from a table.

Syntax:
DELETE FROM table_name [WHERE condition];

Q12. Write a query to delete all the rows from the table “COURSE” where the course is taken by “RITWIK” faculty.



DML

Autocommit Display 10

```
SELECT * FROM COURSE;
```

Results Explain Describe Saved SQL History

COURSE_ID	COURSE_DURATION	COURSE_TRAINER	COURSE_FEES	COURSE_NAME
25	40	PRABHAKARAN	8000	SQL
26	20	AKASH	9000	HTML
26	20	AKASH	9000	HTML

DML

Q13. What happens when no condition is specified in delete statement? Ex: DELETE FROM COURSE;

DML

Autocommit Display 10

```
SELECT * FROM COURS
```

Results Explain Des

no data found

Autocommit Display 10

```
DESC COURSE;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **COURSE**

Table	Column	Data Type	Length	Precision
COURSE	COURSE_ID	Number	-	-
	COURSE_DURATION	Number	-	-
	COURSE_TRAINER	Varchar2	30	-
	COURSE_FEES	Number	-	-
	COURSE_NAME	Varchar2	30	-

DCL

Data Definition – How to control Data.

- DCL commands are used to grant and take back authority from any database user. are used to enforce database security in a multiple database environment.
- Database Administrator's or owner’s of the database object can provide/remove privileges on a database object.

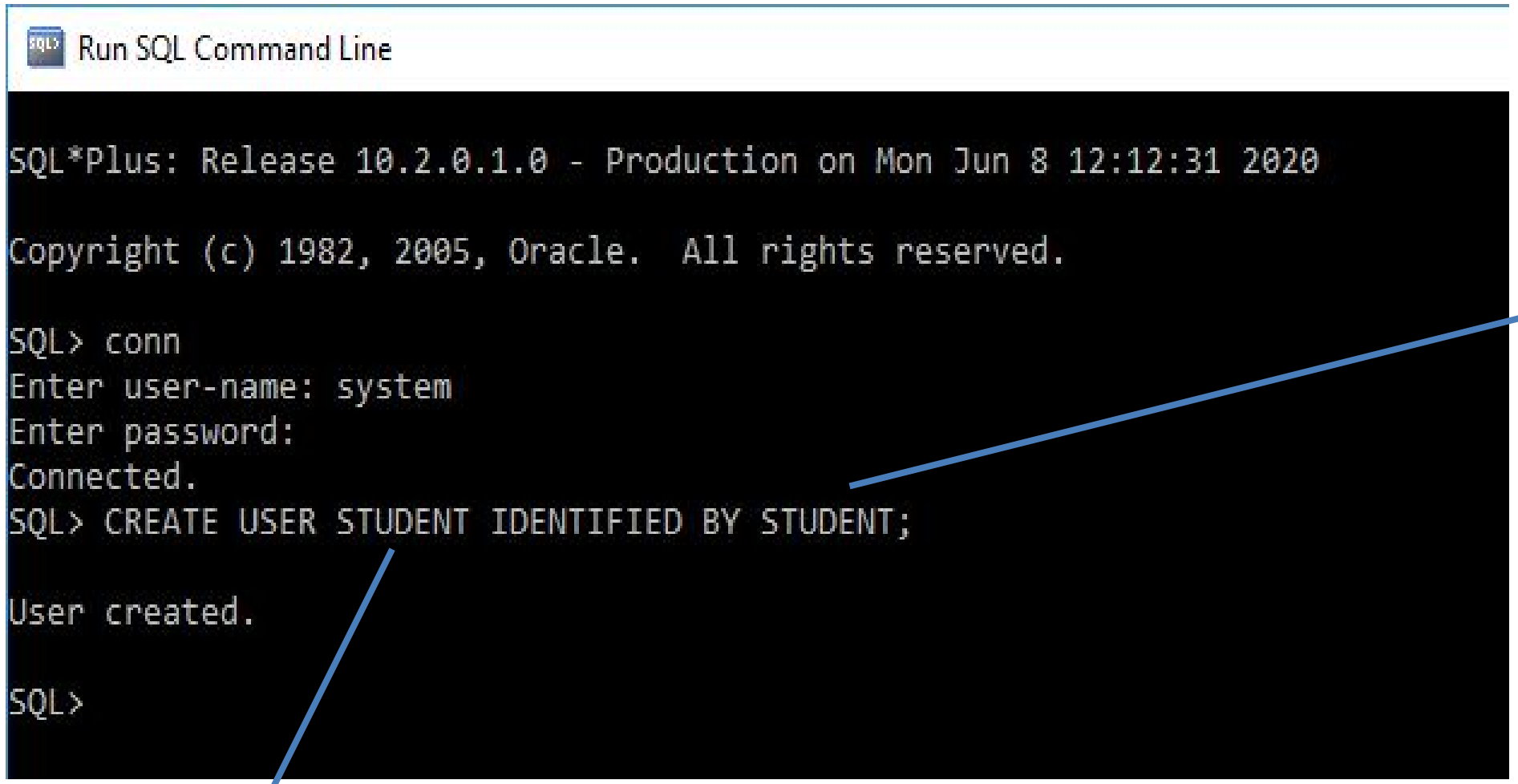
DCL

Grant: It is used to give user access privileges to a database.

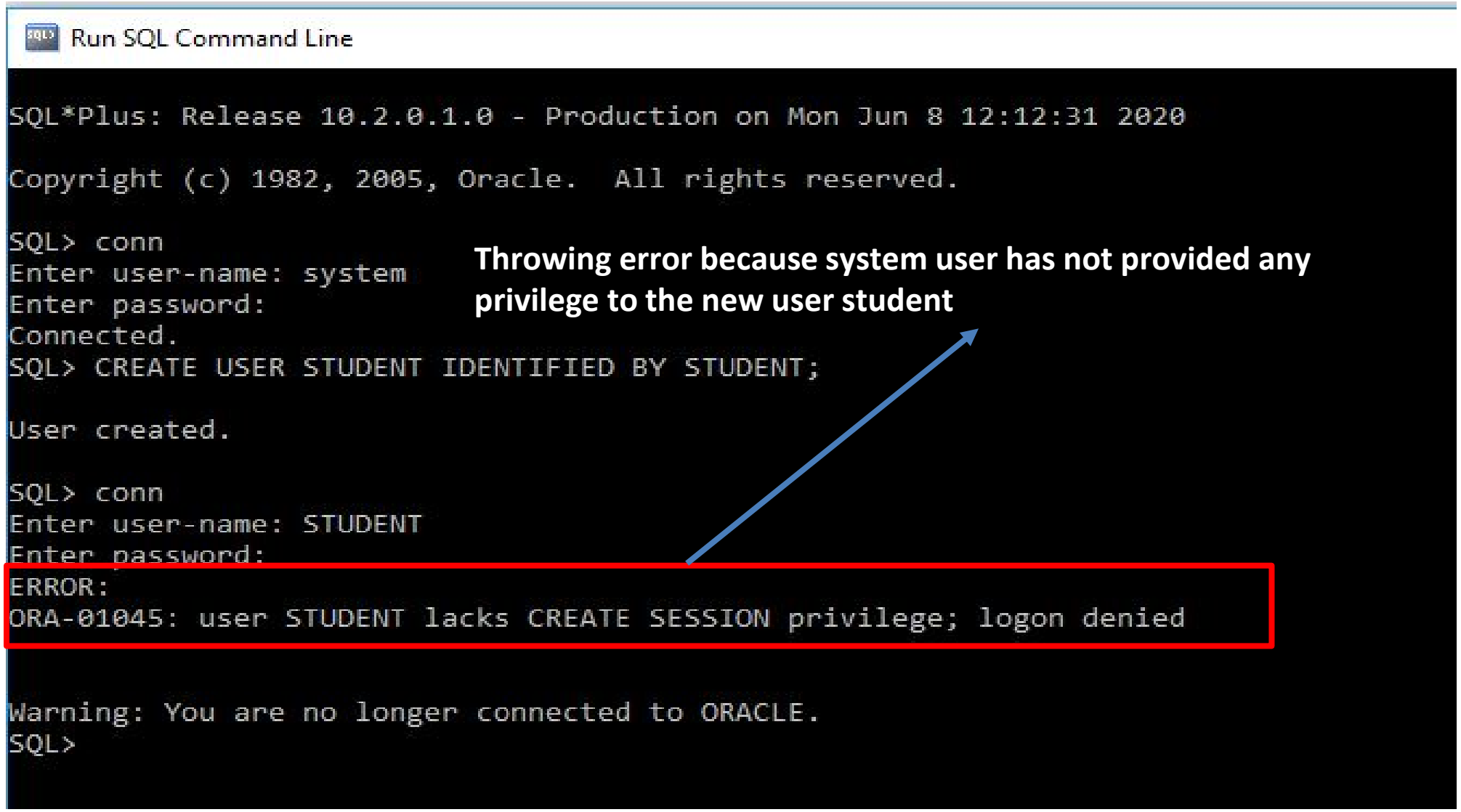
Syntax:
GRANT privilege_name ON object_name TO {user_name | PUBLIC | role_name}

Here,
privilege_name: is the access right or privilege granted to the user.
object_name: is the name of the database object like table.
user_name: is the name of the user to whom an access right is being granted. **Public** is used to grant rights to all the users.

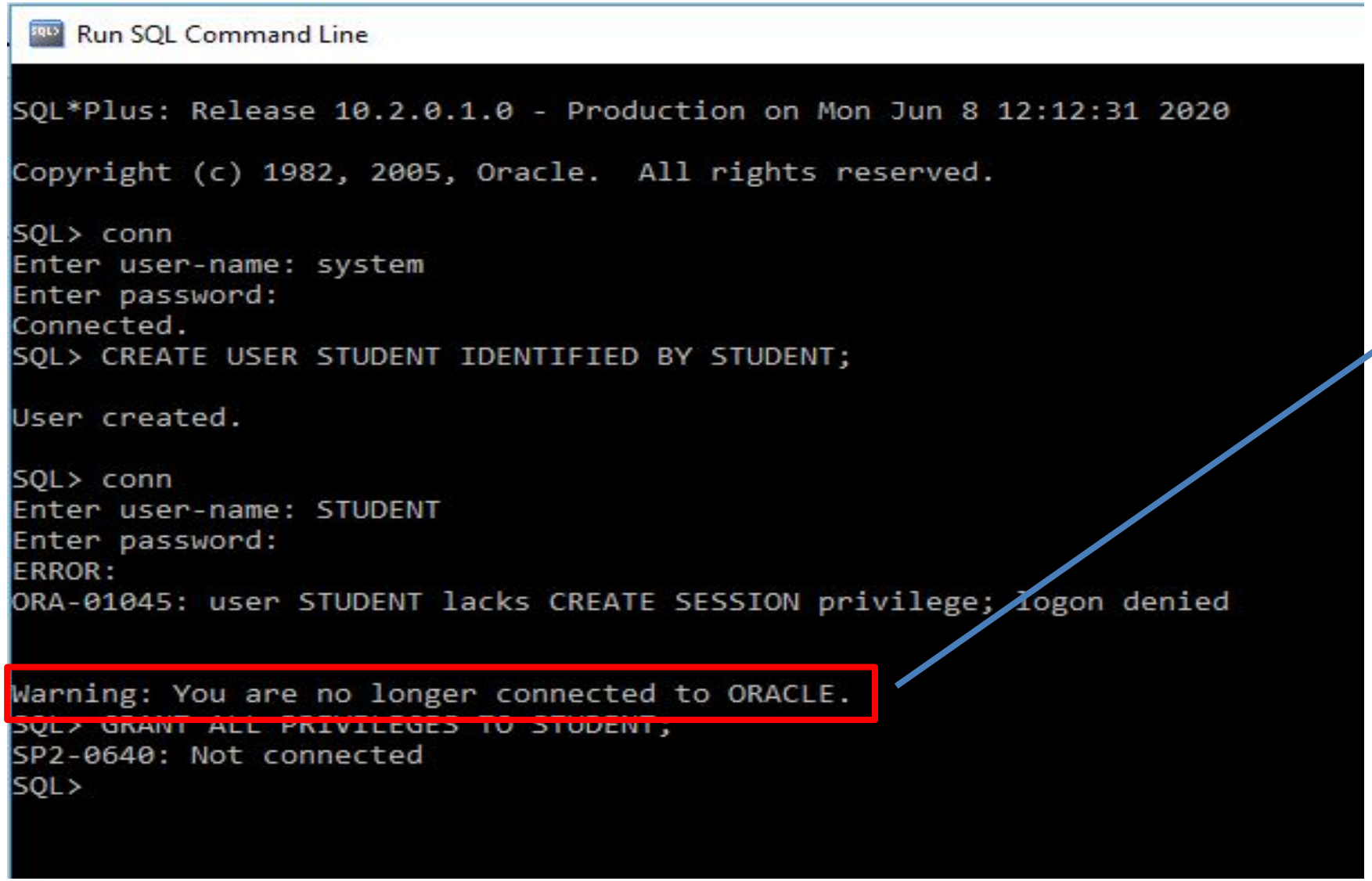
DCL


Run SQL Command Line
SQL*Plus: Release 10.2.0.1.0 - Production on Mon Jun 8 12:12:31 2020
Copyright (c) 1982, 2005, Oracle. All rights reserved.
SQL> conn
Enter user-name: system
Enter password:
Connected.
SQL> CREATE USER STUDENT IDENTIFIED BY STUDENT;
User created.
SQL>
New Password
New username

DCL

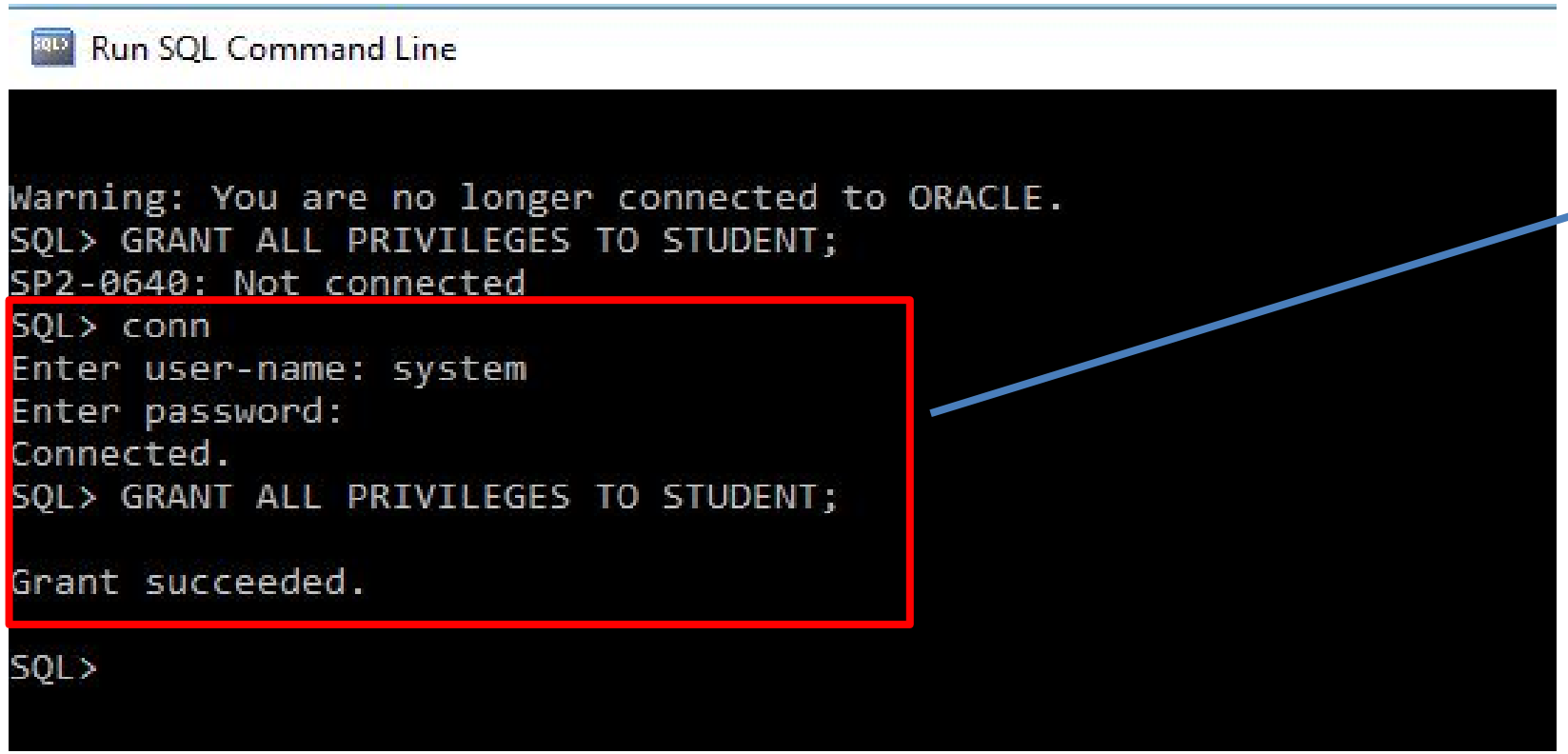

Run SQL Command Line
SQL*Plus: Release 10.2.0.1.0 - Production on Mon Jun 8 12:12:31 2020
Copyright (c) 1982, 2005, Oracle. All rights reserved.
SQL> conn
Enter user-name: system
Enter password:
Connected.
SQL> CREATE USER STUDENT IDENTIFIED BY STUDENT;
User created.
SQL> conn
Enter user-name: STUDENT
Enter password:
ERROR:
ORA-01045: user STUDENT lacks CREATE SESSION privilege; logon denied
Warning: You are no longer connected to ORACLE.
SQL>
Throwing error because system user has not provided any privilege to the new user student

DCL



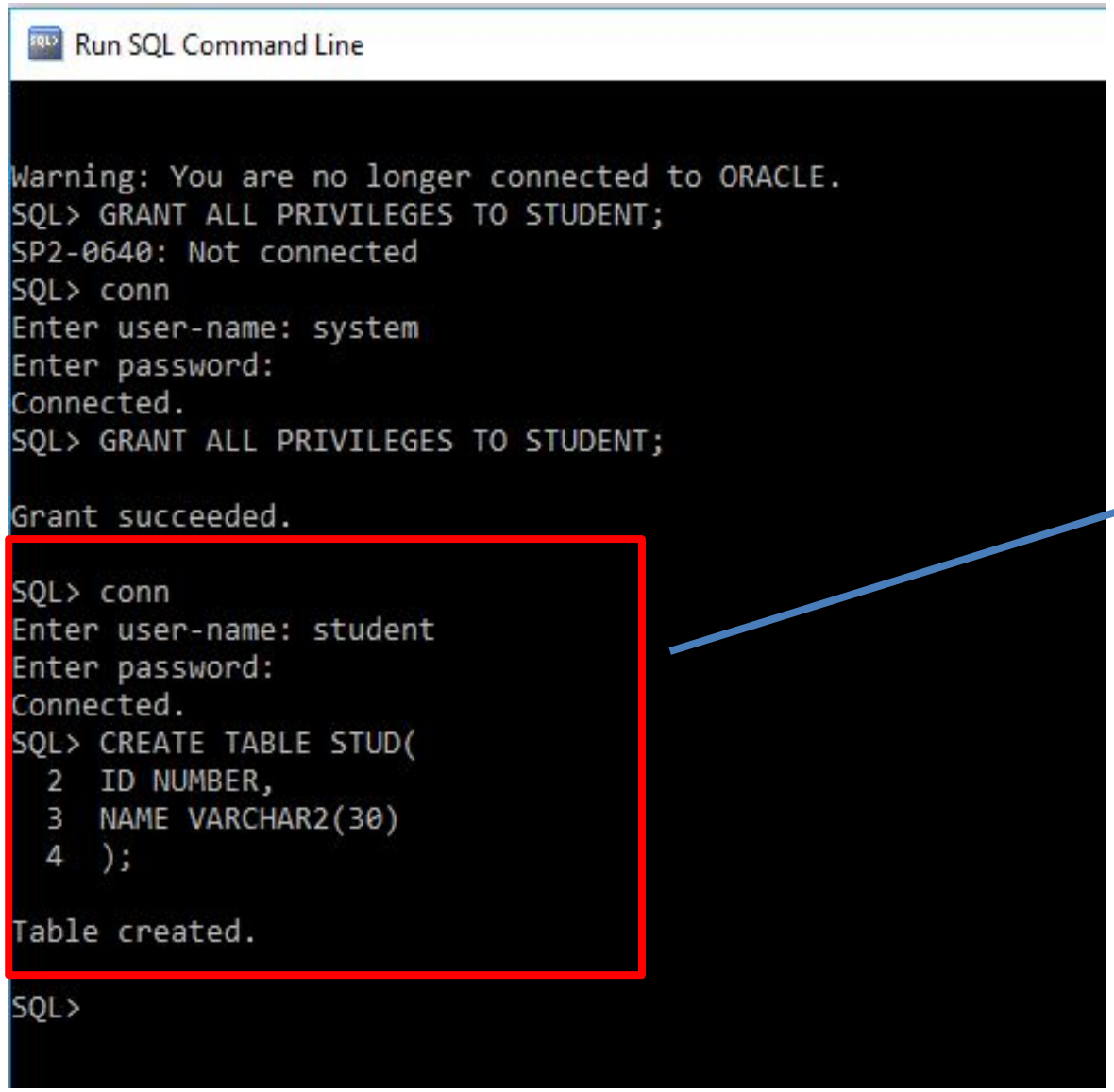
We have to reconnect to user **system** to grant Privileges from it.

DCL



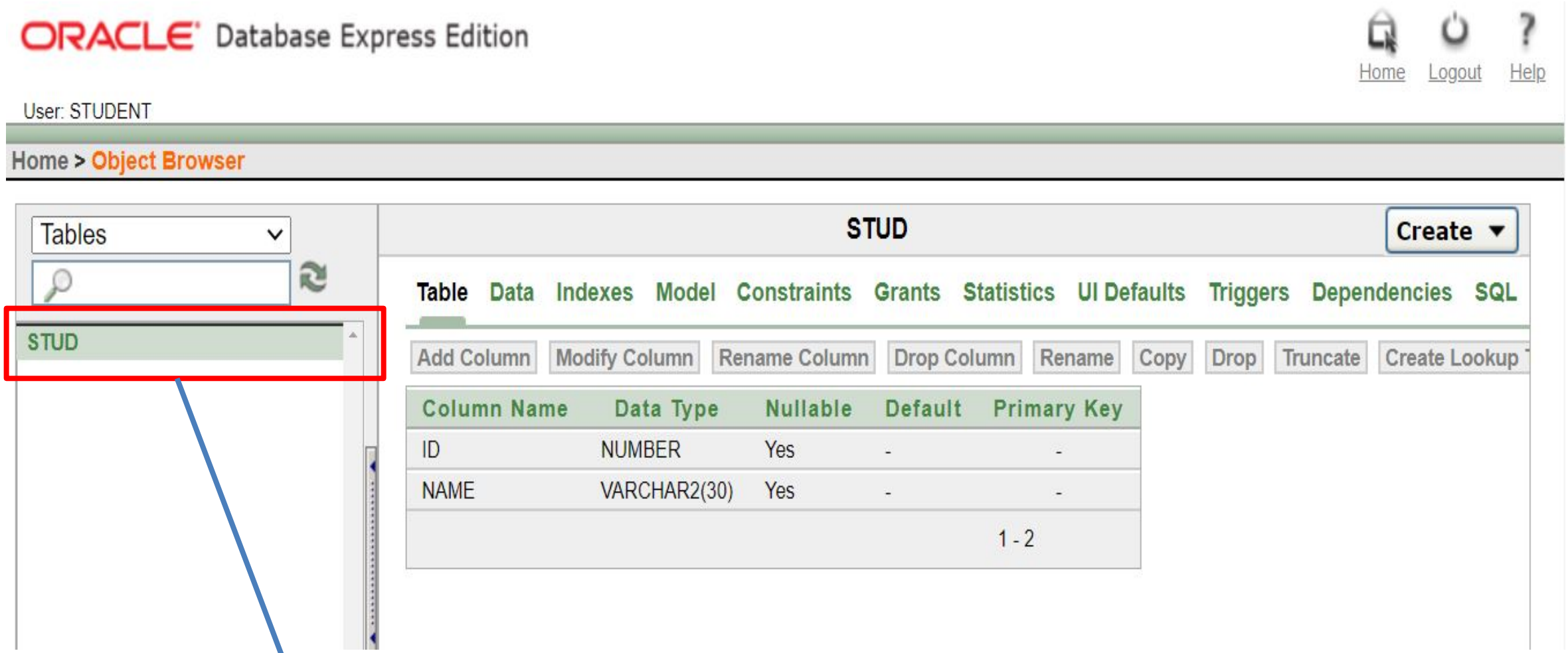
Using system user to grant all privileges to new user student

DCL



Now, Using create command to create a table named stud inside the new user student

DCL



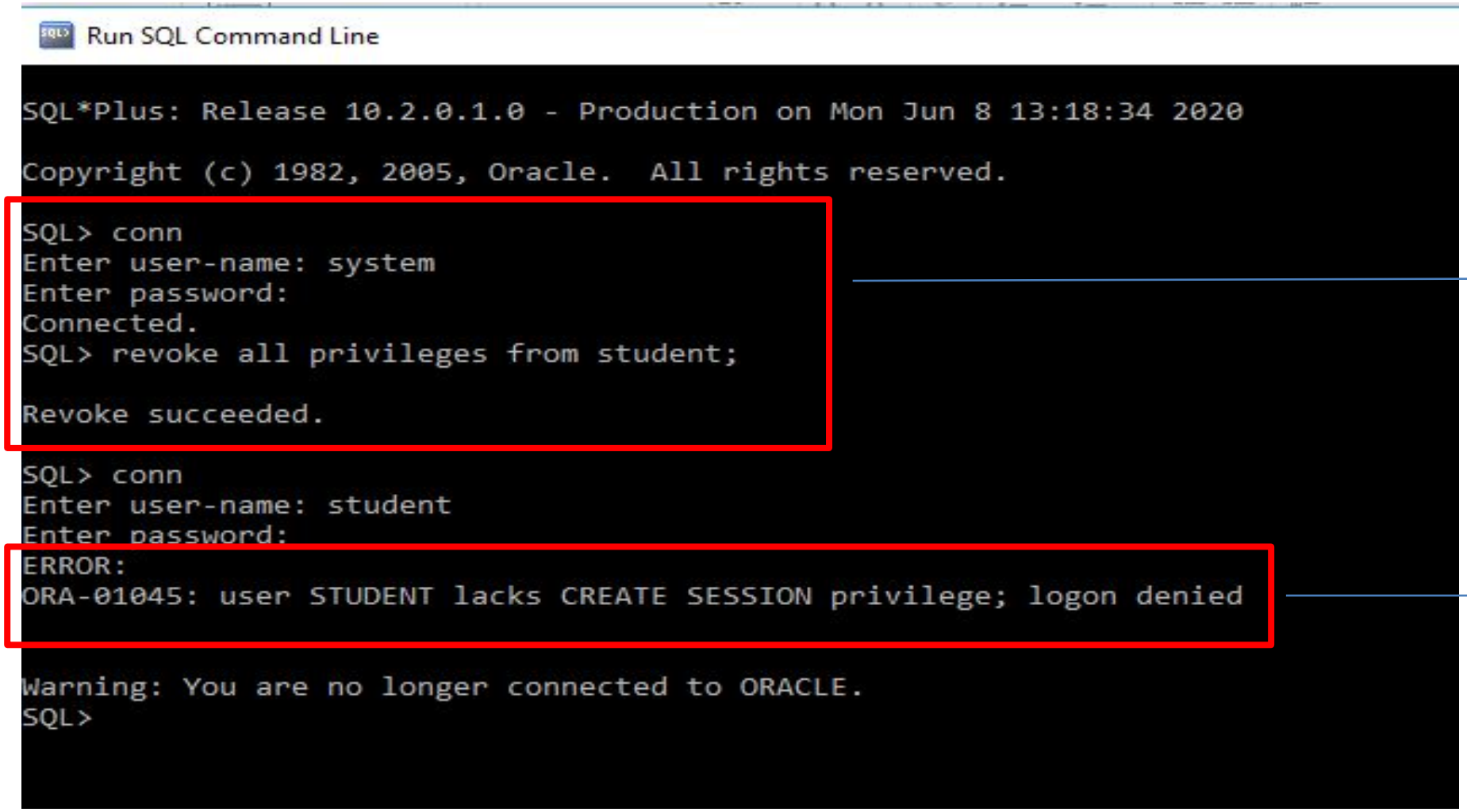
Checking through database home page

DCL

Revoke: It is used to take back permissions from the user.

The syntax for the REVOKE command is:
REVOKE privilege_name ON object_name FROM {User_name | PUBLIC | Role_name}

DCL



Taking back permissions from user student through user system.

When trying to access new user student, throws an error.

DCL

How To List Users in the Oracle Database :

- List all users that are visible to the current user:
  SELECT * FROM all_users;
  The ALL_USERS view lists of all users that is visible to the current user. However, this view doesn't describe the users.
  SELECT * FROM all_users ORDER BY created;

- List all users in the Oracle Database:
  SELECT * FROM dba_users;
  The DBA_USERS view describes all user in the Oracle database.
  SELECT * FROM DBA_USERS ORDER BY created DESC;

- Show the information of the current user:
  SELECT * FROM user_users;

Great Learning

Autocommit Display 100 Save Run

```
SELECT * FROM user_users;
```

Results Explain Describe Saved SQL History

USERNAME	USER_ID	ACCOUNT_STATUS	LOCK_DATE	EXPIRY_DATE	DEFAULT_TABLESPACE	TEMPORARY_TABLESPACE	CREA
SYSTEM	5	OPEN	-	-	SYSTEM	TEMP	07-FEE

Autocommit Display 10 Save Run

```
SELECT * FROM user_users;
```

Results Explain Describe Saved SQL History

USERNAME	USER_ID	ACCOUNT_STATUS	LOCK_DATE	EXPIRY_DATE	DEFAULT_TABLESPACE	TEMPORARY_TABLESPACE	CREA
STUDENT	39	OPEN	-	-	SYSTEM	TEMP	08-JUN

TCL

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

- TCL used to control transactional processing in a database.
- A transaction is logical unit of work that comprises one or more SQL statements

TCL

Commit: Commit c

Syntax:
COMMIT;

Autocommit Display 10

```
SELECT * FROM DEPT;
```

Results Explain Describe Saved SQL History

DEPT_ID	DEPT_NAME	MANAGER_ID	LOC_ID
20	Admin	300	2100
22	Shipping	302	2300
23	IT	303	2400
24	Sales	304	2500
25	Executive	305	2600
110	Acc	306	2700
190	Contracting	-	2800
26	Admin	307	2900

Autocommit Dis

```
DELETE FROM DEPT
```

Results Explain De

```
1 row(s) deleted.
```

```
ME = 'Marketing';
```

QL History

TCL

Rollback: Rollback database.

Syntax:
ROLLBACK;

Autocommit Display 10

```
SELECT * FROM DEPT;
```

Results Explain Describe Saved SQL History

Autocommit Display 10

```
DELETE FROM DEPT WHERE DEPT_NAME = 'Admin';
```

Results Explain Describe Saved SQL History

2 row(s) deleted.

Autocommit Display 10

```
DELETE FROM DEPT WHERE DEPT_NAME = 'Admin';
ROLLBACK;
```

Results Explain Describe Saved SQL History

Statement processed.

110	Acc	306	2700
190	Contracting	-	2800
26	Admin	307	2900

TCL

Autocommit Display 10

```
DELETE FROM DEPT WHERE DEPT_NAME = 'Admin';
COMMIT;
ROLLBACK;
```

Results Explain Describe

```
Statement processed.
```

Autocommit Display 10

```
SELECT * FROM DEPT;
```

Results Explain Describe Saved SQL History

DEPT_ID	DEPT_NAME	MANAGER_ID	LOC_ID
22	Shipping	302	2300
23	IT	303	2400
24	Sales	304	2500
25	Executive	305	2600
110	Acc	306	2700
190	Contracting	-	2800

TCL

SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction. It is to divide the transaction into smaller sections. It defines breakpoints for a transaction to allow partial rollback.

Syntax:
SAVEPOINT SAVEPOINT_NAME;

```
SQL> select * from student1;

        ID NAME                           GENDER
---------- ------------------------------ ------------------------------
       AGE PASSOUT_YEAR
---------- ------------
         1 RITWIK                         MALE
        26          2015

         2 ROHAN                          MALE
        28          2015

         3 ADITYA                         MALE
        25          2015
```

TCL

```
SQL> UPDATE STUDENT1 SET AGE=29 WHERE NAME = 'RITWIK';

1 row updated.

SQL> SAVEPOINT SP1;

Savepoint created.

SQL> UPDATE STUDENT1 SET AGE=30 WHERE NAME = 'ROHAN';

1 row updated.

SQL> SAVEPOINT SP2;

Savepoint created.
```

```
SQL> SELECT * FROM STUDENT1;

        ID NAME                           GENDER
---------- ------------------------------ ------------------------------
       AGE PASSOUT_YEAR
---------- ------------
         1 RITWIK                         MALE
        29         2015

         2 ROHAN                          MALE
        30         2015

         3 ADITYA                         MALE
        25         2015
```

TCL

```
SQL> ROLLBACK TO SAVEPOINT SP1;

Rollback complete.

SQL> SELECT * FROM STUDENT1;

        ID NAME                           GENDER
---------- ------------------------------ ------------------------------
       AGE PASSOUT_YEAR
---------- ------------
         1 RITWIK                         MALE
        29         2015

         2 ROHAN                          MALE
        28         2015

         3 ADITYA                         MALE
        25         2015
```

Database objects and SP

Thank You