SQL TUTORIAL

Submitted By
Maimoona Khilji
Registration no. 195300273
BS-Data Science
Semester - IV

What is SQL?

- SQL stands for Structured Query Language.
- A computer language for:
  - Storing data
  - manipulating data
  - retrieving data
- SQL is the standard language for Relational Database System.

Why SQL?

SQL is widely popular because it offers the following advantages:

- It is standard language.
- It is easy to understand.
- Queries are short.
- It allows user to describe, define, manipulate and drop the data.

Brief History of SQL

- 1970 – Dr. Edgar F. "Ted" Codd (Dr. EF. Codd) of IBM is known as the father of relational databases. He described a relational model for databases.
- 1974 – Structured Query Language appeared.
- 1986 – IBM developed the first prototype of relational database. The first relational database was released by Relational Software which later came to be known as Oracle.

SQL Statements

- A SQL statement is a computer program or instruction that consists of identifiers, parameters, variables, names, data types, and SQL reserved words.
- SQL **reserved words** have special meaning in SQL and should not be used for any other purpose. For example, SELECT and UPDATE are reserved words and should not be used as table names
- SQL statements are divided into the following categories:
    - DDL - Data Definition Language
    - DML - Data Manipulation Language
    - DRL - Data Retrieval Language
    - DCL - Data Control Language
    - TCL - Transaction Control Language

Data Definition Language (DDL)

Data Definition Language (DDL) Statements

- DDL or Data Definition Language actually consists of the SQL commands that can **be used to define the database schema**.
- It simply deals with descriptions of the database schema and is used to create and modify the **structure of database** objects in the database. It enable you to change the **structure of the database**.

Command	Description
CREATE	It is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
ALTER	It is used to alter the structure of the database. It modifies an existing database object, such as a table.
DROP	It is used to delete objects from the database like an entire table, a view of a table or other objects in the database.
TRUNCATE	It is used to remove all records from a table, including all spaces allocated for the records are removed.
COMMENT	It is used to add comments to the data dictionary.
RENAME	It is used to rename an object existing in the database.

Data Manipulation Language (DML)

Data Manipulation Language (DML) Statements

The SQL commands that deals with the manipulation of data present in the database belong to DML or **Data Manipulation Language** and this includes most of the SQL statements.

Command	Description
INSERT	Creates a record. It is used to insert data into a table.
UPDATE	Modifies records. It is used to update existing data within a table.
DELETE	It is used to delete records from a database table.

Data Retrieval Language (DRL)

Data Retrieval Language (DRL)

DRL or Data Retrieval Language actually consists of the SQL commands that can be used to retrieve the data from the tables of database schema

Command	Description
SELECT	Retrieves certain records from one or more tables.

Data Control Language (DCL)

Data Control Language (DCL) Statements

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

Command	Description
GRANT	It gives user’s access privileges to the database.
REVOKE	It withdraw user’s access privileges given by using the GRANT command.

Transaction Control Language (TCL)

Transaction Control Language (TCL) Statements

Transaction control statements manage the changes made by DML statements and group DML statements into transactions.

Command	Description
Commit	Make changes to a transaction permanent
Rollback	Undo the changes in a transaction, since the transaction started

SQL – Syntax

SQL - Syntax

- SQL is followed by a unique set of rules and guidelines called **Syntax**. This tutorial gives you a quick start with SQL by listing all the basic SQL Syntax.

- All the **SQL statements** start with any of the keywords like SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, CREATE, USE, SHOW and all the statements end with a semicolon (;).

- The most important point to be noted here is that SQL is case **insensitive**, which means **SELECT** and **select** have same meaning in SQL statements.

SQL- Data Definition Language

CREATE

- CREATE DATABASE
- CREATE TABLE
- CREATE USER

CREATE Database

- Syntax:

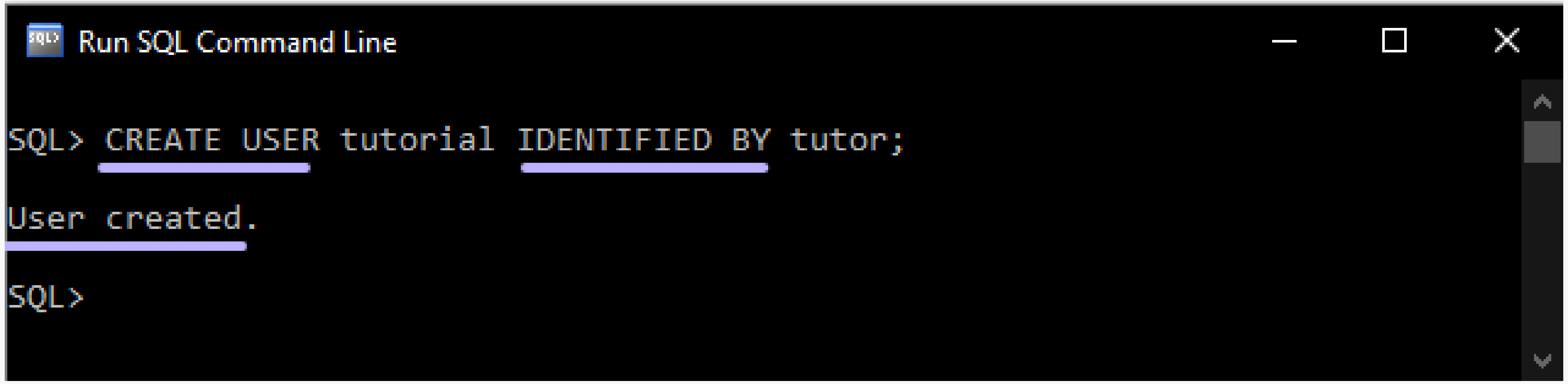
```
CREATE DATABASE  database_name;
```

CREATE USER

- Syntax:

  CREATE USER user_name **identified BY** password;

Note: For creating user, first connect to a System.



Run SQL Command Line

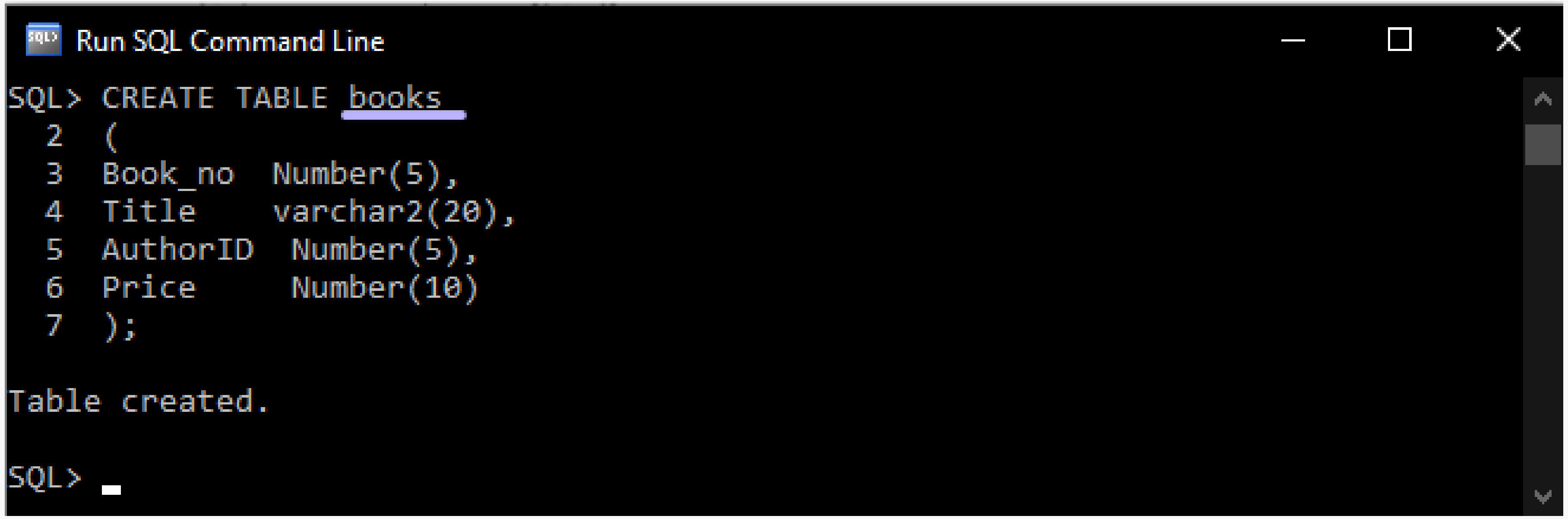
```
SQL> CREATE USER tutorial IDENTIFIED BY tutor;

User created.

SQL>
```

CREATE Table

- Syntax:

```
CREATE TABLE table_name
(
  column1 datatype,
  column2 datatype,
  .....
   columnN datatype
 );
```

Table = **Books**
columns are **Book_no, Title, AuthorID, Price.**


Run SQL Command Line

```
SQL> CREATE TABLE books
(
Book_no  Number(5),
Title    varchar2(20),
AuthorID  Number(5),
Price     Number(10)
);

Table created.

SQL>
```

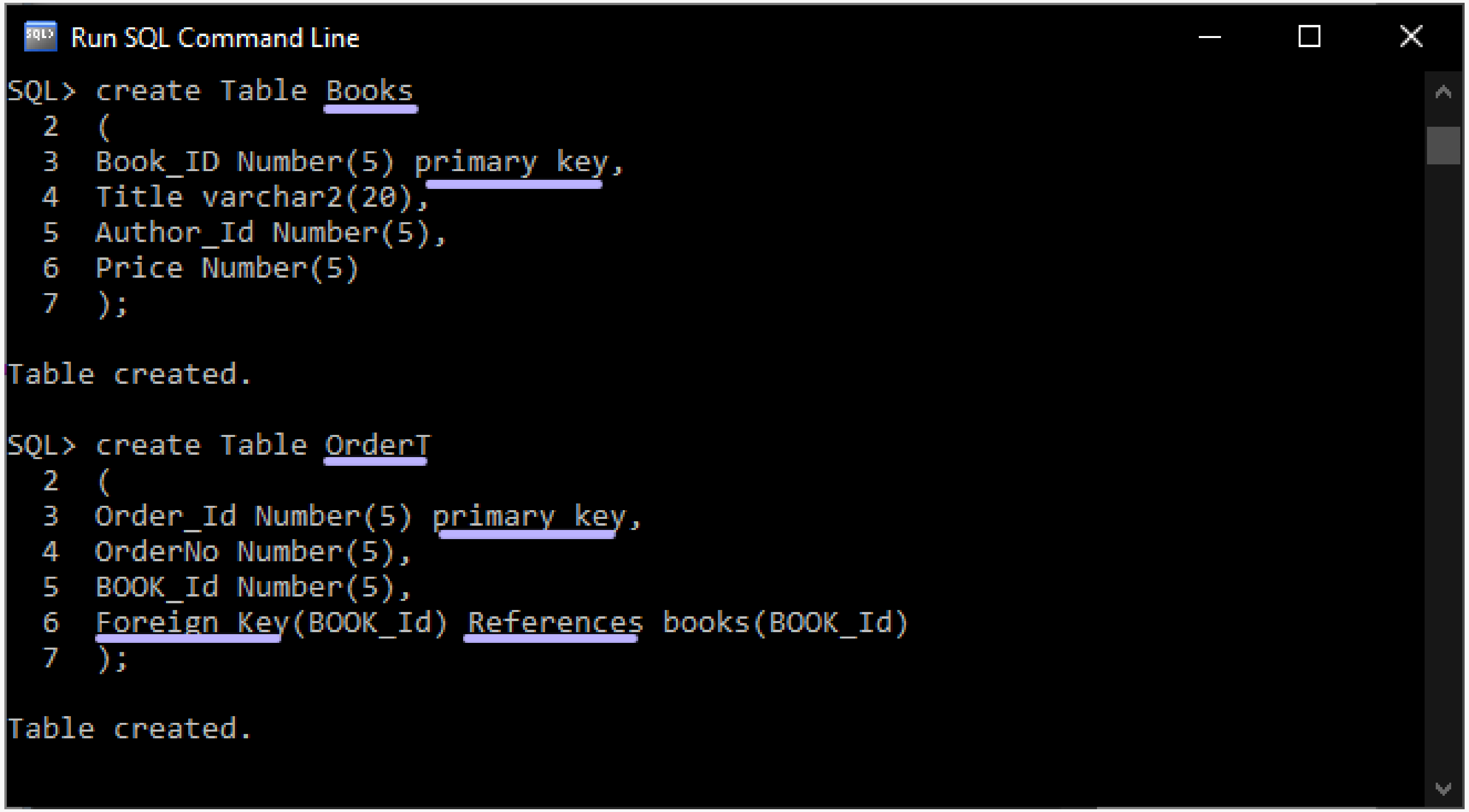
DDL Specifying Keys
Single and Multi - Column Keys

CREATE Table with constraints

- Single column keys can be defined at the column level instead of at the table level at the end of the field descriptions.
- Multi-Column keys still need to be defined separately at the table level
- Syntax:

```
CREATE TABLE table_name
(          column1 datatype  Primary Key,
           column2 datatype  UNIQUE,
           column3 datatype,
           column4 datatype,
           column5 datatype,
           Unique(column3, column4)
           FOREIGN KEY (column5) REFERENCES table2 (column)
 );
```

Table with Primary and Foreign Key



```
SQL> create Table Books
  2  (
  3  Book_ID Number(5) primary key,
  4  Title varchar2(20),
  5  Author_Id Number(5),
  6  Price Number(5)
  7  );

Table created.

SQL> create Table OrderT
  2  (
  3  Order_Id Number(5) primary key,
  4  OrderNo Number(5),
  5  BOOK_Id Number(5),
  6  Foreign Key(BOOK_Id) References books(BOOK_Id)
  7  );

Table created.
```

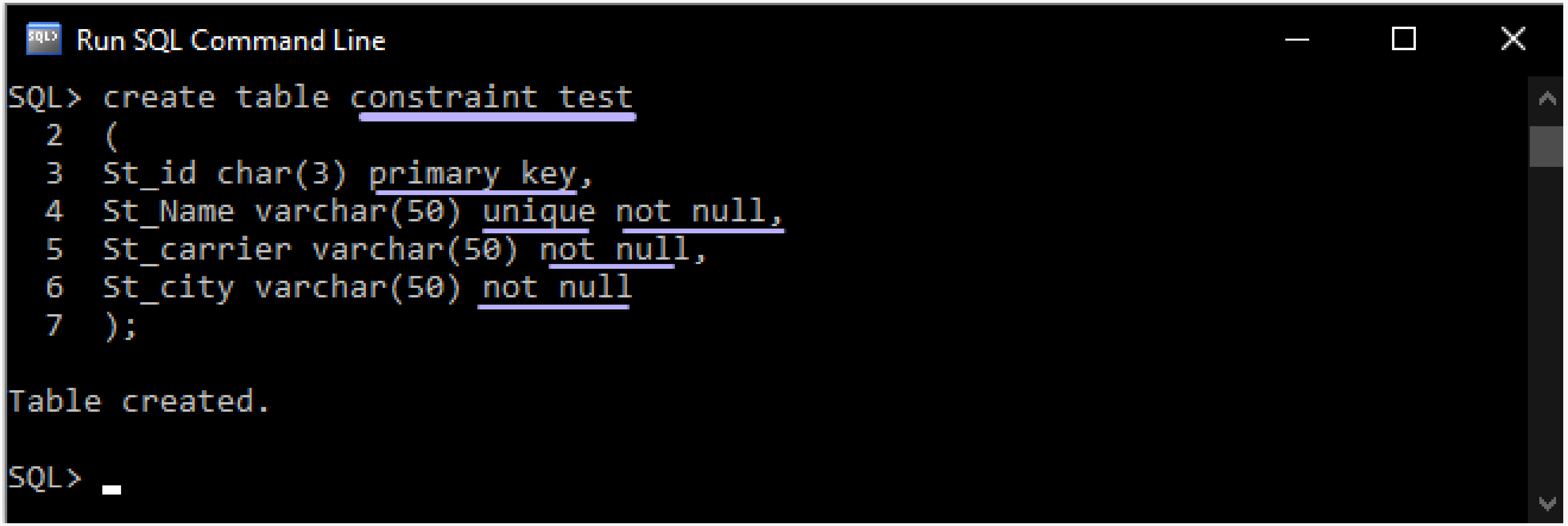
DDL Constraints
Disallowing Null Values

Disallowing Null Values:

- Null values entered into a column means that the data in not known.
- These can cause problems in Querying the database.
- Specifying Primary Key automatically prevents null being entered in columns which specify the primary key
- Not Null clause is used in preventing null values from being entered in a column. Syntax:

```
CREATE TABLE table_name
(
  column1 datatype  Primary Key,
  column2 datatype  NULL,
  column3 datatype  NOT NULL,
  column4 datatype,
   Unique(column3, column4)
 );
```

Null constraints



```
SQL> create table constraint test
  2  (
  3  St_id char(3) primary key,
  4  St_Name varchar(50) unique not null,
  5  St_carrier varchar(50) not null,
  6  St_city varchar(50) not null
  7  );

Table created.

SQL>
```

DDL
Constraints- Value Constraints

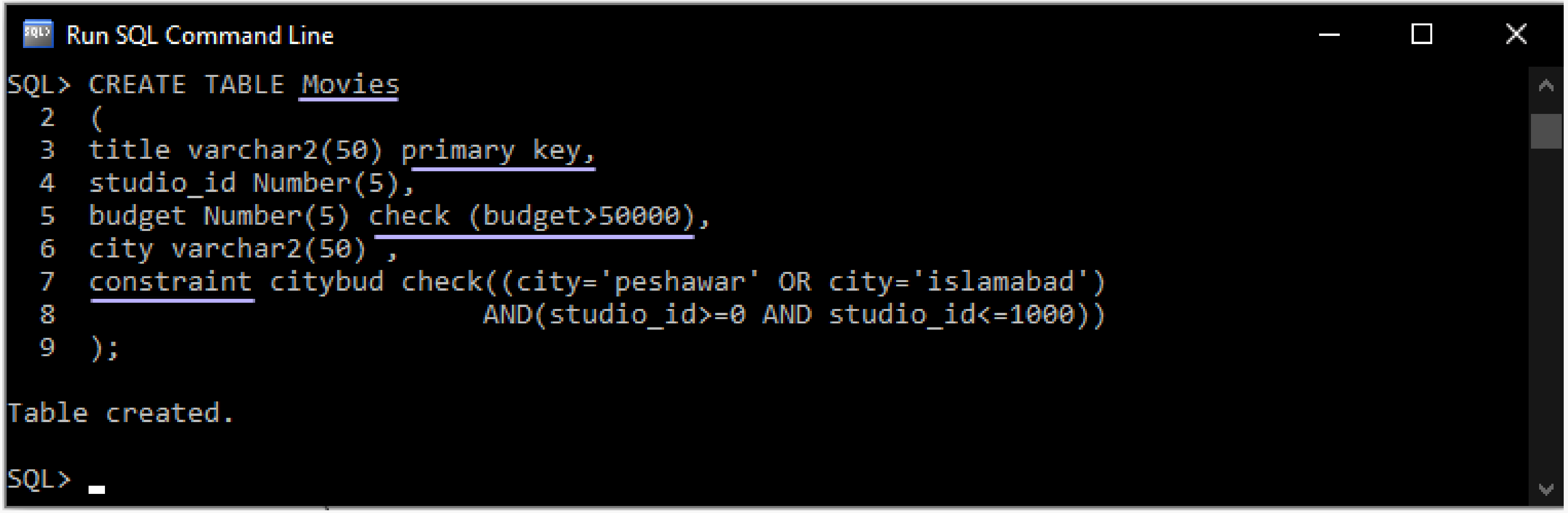
Value Constraints:

- Allows value inserted in the column to be checked condition in the column constraint.
- Check clause is used to create a constraint in SQL.
- Table level constraints can also be defined using the Constraint keyword

Syntax:

```
CREATE TABLE table_name
(
  column1 datatype  Primary Key check (column1 > 0),
  column2 datatype,
  column3 datatype,
  column4 datatype,

 CONSTRAINT constraint_name Check ( conditon1 and condition2) )
 );
```

Value Constraints



```
SQL> CREATE TABLE Movies
  2  (
  3  title varchar2(50) primary key,
  4  studio_id Number(5),
  5  budget Number(5) check (budget>50000),
  6  city varchar2(50) ,
  7  constraint citybud check((city='peshawar' OR city='islamabad')
  8                          AND(studio_id>=0 AND studio_id<=1000))
  9  );

Table created.

SQL> 
```

DDL
Constraints- Default Value

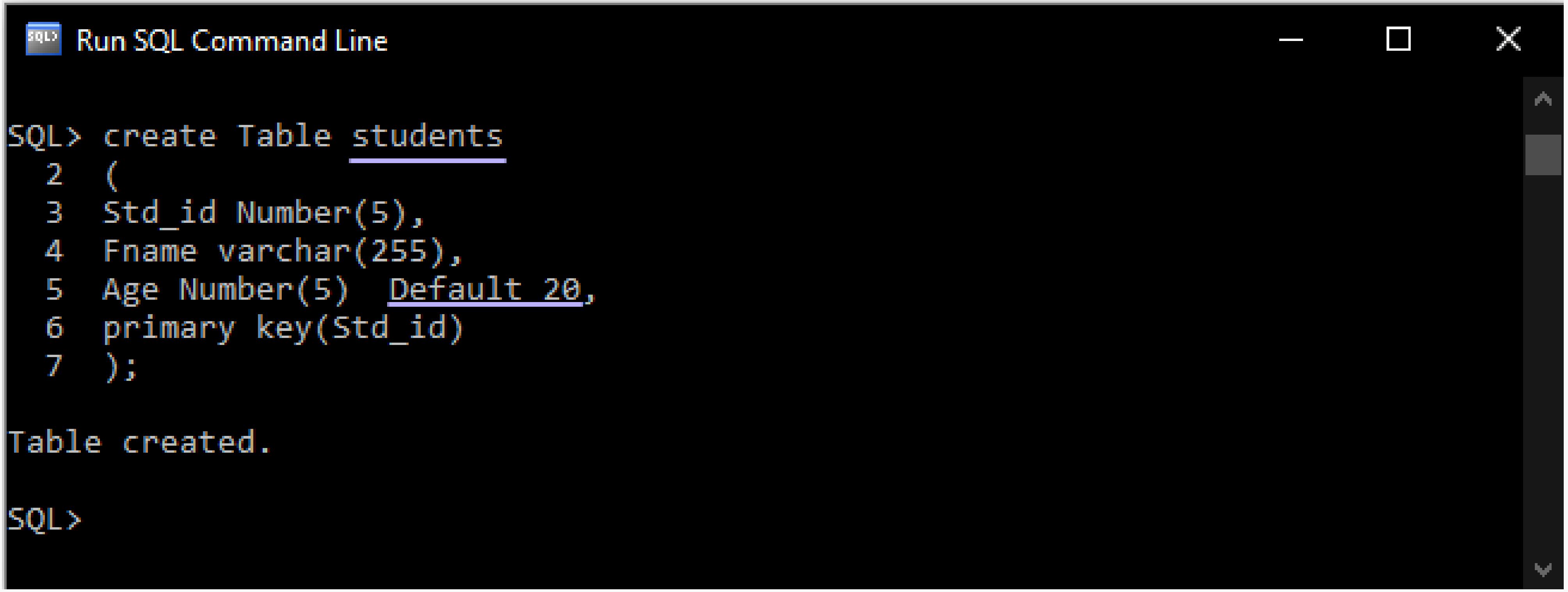
Default Value:

- A default value can be inserted in any column by using the Default keyword.

Syntax:

```
CREATE TABLE table_name
(
  column1 datatype,
  column2 datatype,
  column3 datatype,
  column4 datatype  default ‘value’

 );
```

Default Value



```
SQL> create Table students
  2  (
  3  Std_id Number(5),
  4  Fname varchar(255),
  5  Age Number(5)  Default 20,
  6  primary key(Std_id)
  7  );

Table created.

SQL>
```

Data Manipulation Language (DML)

INSERT

- It allows you to add new records to the Table
- If the columns are not specified, then data goes in the order specified in the table

Syntax:

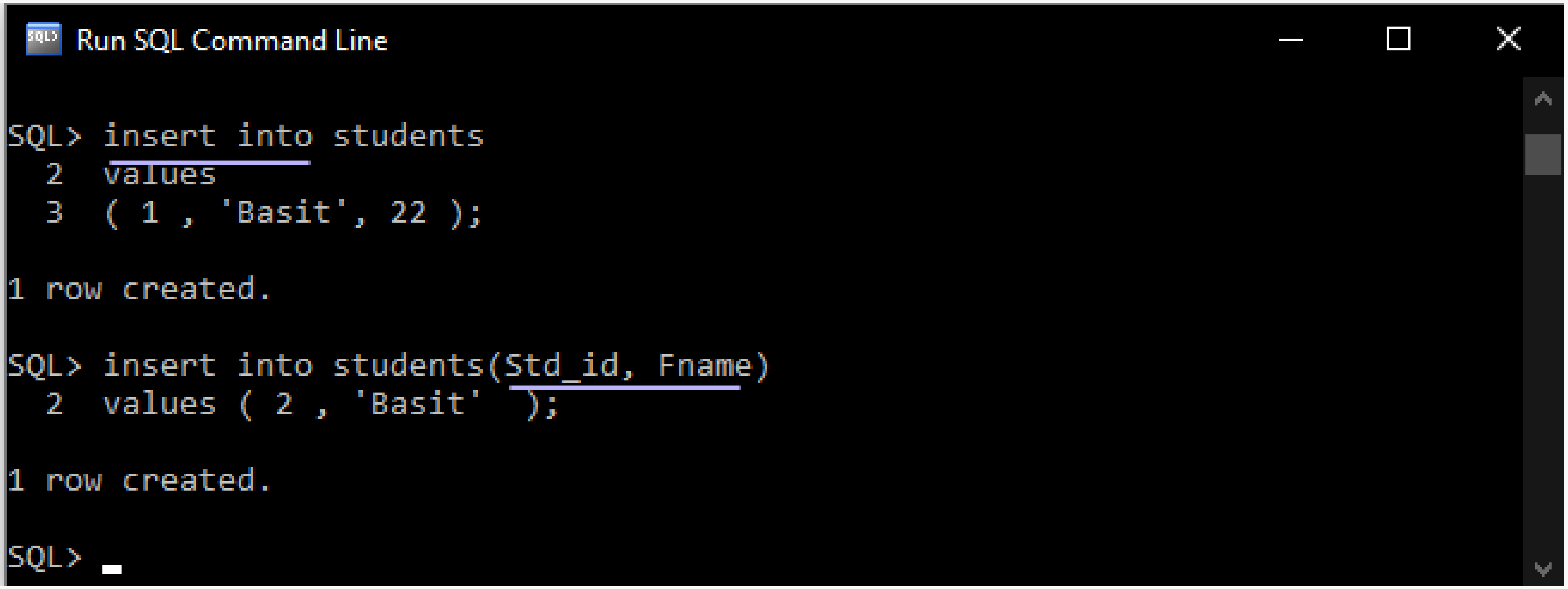
```
INSERT INTO table_name

VALUES ( [value_list]);
```

- If the columns are specified, then data goes in the order of columns specified in the column_list.

Syntax:

```
INSERT INTO table_name ( [column_list])
VALUES ( [value_list]);
```

INSERT



Run SQL Command Line

```
SQL> insert into students
  2  values
  3  ( 1 , 'Basit', 22 );

1 row created.

SQL> insert into students(Std_id, Fname)
  2  values ( 2 , 'Basit'  );

1 row created.

SQL>
```

INSERT INTO - SELECT

- Copy all columns from one table to another table:

Syntax:

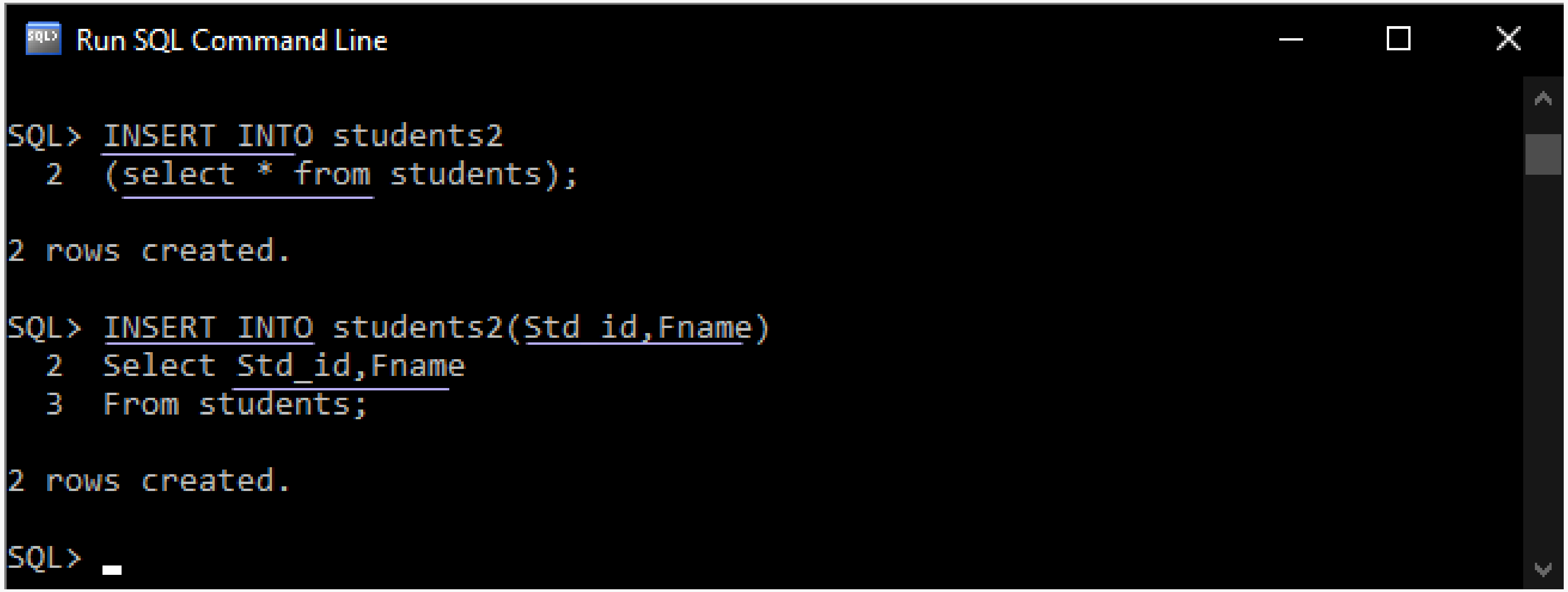
INSERT INTO *table2*
SELECT * **FROM** *table1*
WHERE *condition*;

- Copy only some columns from one table into another table:

Syntax:

INSERT INTO *table2* (*column1*, *column2*, *column3*, ...)
SELECT *column1*, *column2*, *column3*, ...
FROM *table1*
WHERE *condition*;

INSERT INTO - SELECT



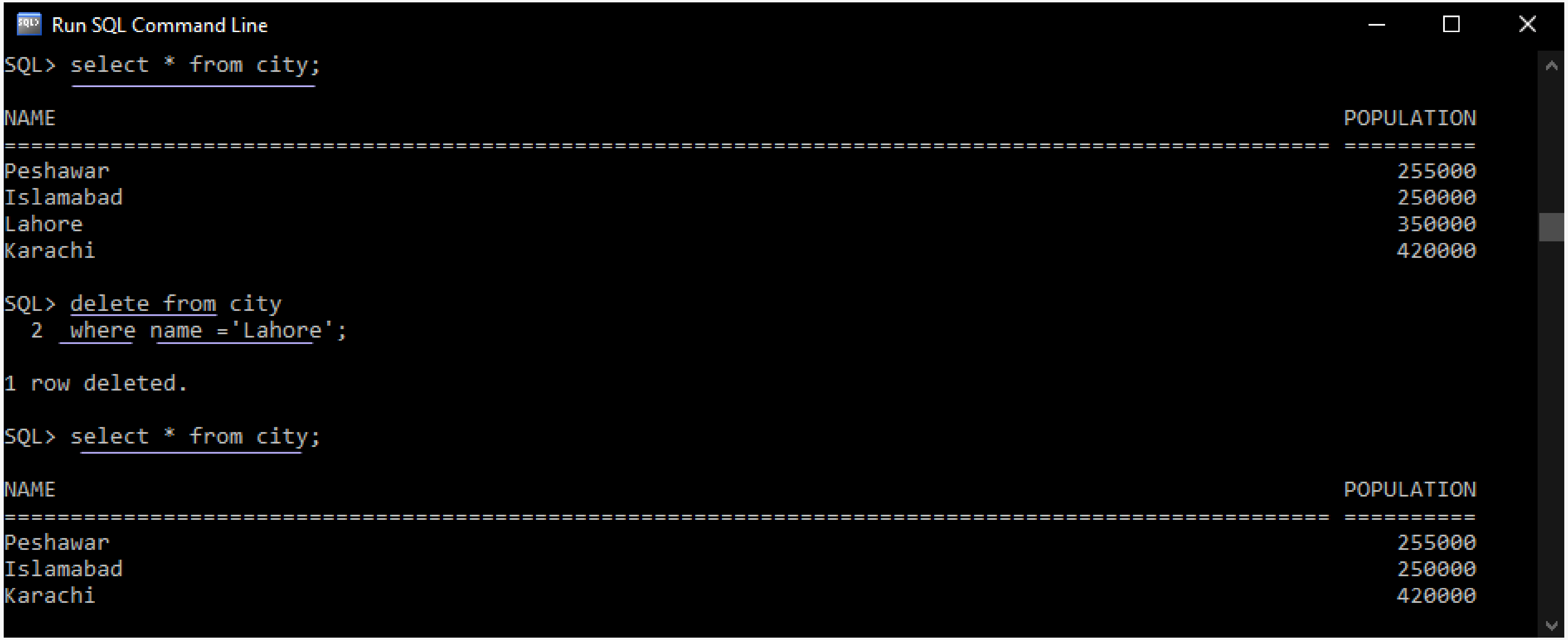
```
SQL> INSERT INTO students2
  2  (select * from students);

2 rows created.

SQL> INSERT INTO students2(Std_id,Fname)
  2  Select Std_id,Fname
  3  From students;

2 rows created.

SQL>
```

DELETE

- It is used to remove records from a table of the database. The **where** clause in the syntax is used to restrict the rows deleted from the table otherwise all the rows from the table are deleted.
- Syntax:

```
DELETE FROM table_name
WHERE {CONDITION};
```

DELETE


Run SQL Command Line

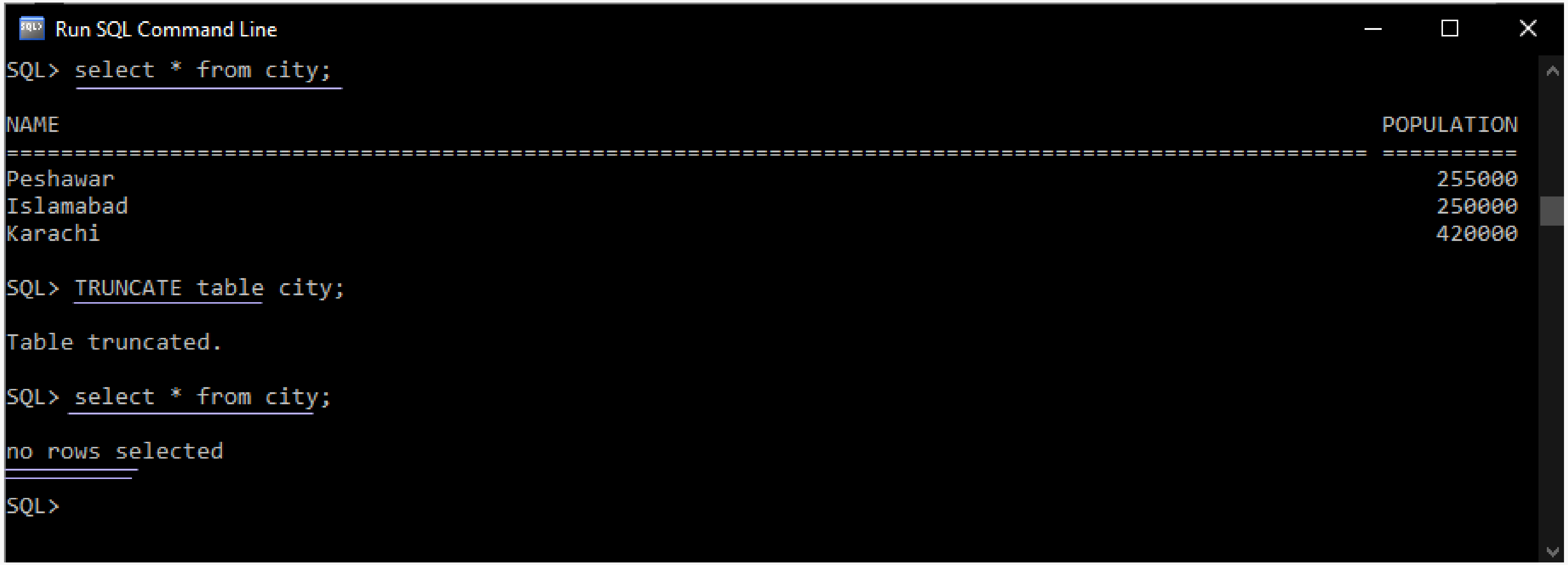
```
SQL> select * from city;

NAME                                                                                                   POPULATION
==================================================================================================== ==========
Peshawar                                                                                                   255000
Islamabad                                                                                                  250000
Lahore                                                                                                     350000
Karachi                                                                                                    420000

SQL> delete from city
  2  where name ='Lahore';

1 row deleted.

SQL> select * from city;

NAME                                                                                                   POPULATION
==================================================================================================== ==========
Peshawar                                                                                                   255000
Islamabad                                                                                                  250000
Karachi                                                                                                    420000
```

TRUNCATE

- It used to delete all the rows of a table. Delete can also be used to delete all the rows from the table.
- The difference is that **delete** performs a delete operation on each row in the table while the **Truncate** statement simply throws away all the rows at once and is much quicker.
- The note of caution is that truncate does not do integrity checks on the way which can lead to inconsistencies on the way. If there are dependencies requiring integrity checks we should use delete.
- Syntax:

TRUNCATE TABLE table_name;

TRUNCATE


Run SQL Command Line

```
SQL> select * from city;

NAME                                                                                                POPULATION
=================================================================================================== ==========
Peshawar                                                                                                255000
Islamabad                                                                                               250000
Karachi                                                                                                 420000

SQL> TRUNCATE table city;

Table truncated.

SQL> select * from city;

no rows selected

SQL>
```

UPDATE

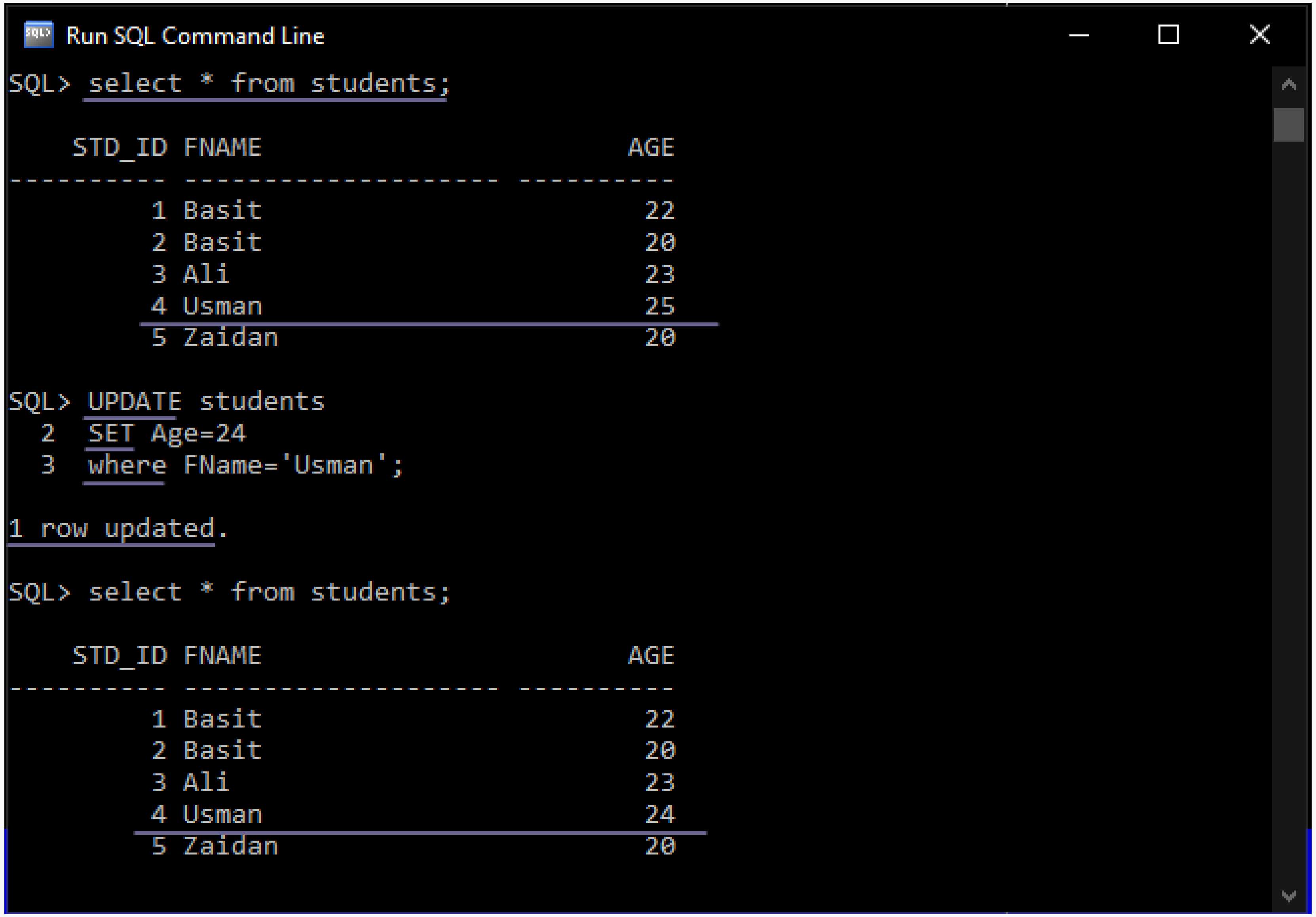
- It used to make changes to existing rows of the table.
- It has three parts.
  - ➢ First, you ,must specify which table is going to be updated.
  - ➢ The second part of the statement is the set clause, in which you should specify the columns that will be updated as well as the values that will be inserted.
  - ➢ Finally, the where clause is used to specify which rows will be updated.
- Syntax:

UPDATE table_name

SET column_name1 = value1, column_name2 = value2, …..

[**WHERE** Condition]

Update



```
SQL> select * from students;

    STD_ID FNAME                       AGE
---------- -------------------- ----------
         1 Basit                        22
         2 Basit                        20
         3 Ali                          23
         4 Usman                        25
         5 Zaidan                       20

SQL> UPDATE students
  2  SET Age=24
  3  where FName='Usman';

1 row updated.

SQL> select * from students;

    STD_ID FNAME                       AGE
---------- -------------------- ----------
         1 Basit                        22
         2 Basit                        20
         3 Ali                          23
         4 Usman                        24
         5 Zaidan                       20
```

DROP

- It is used to remove elements from a database, such as tables, indexes, users and databases.
- Drop command is used with a variety of keywords based on the need.
    - Drop Table
    - Drop User
    - Drop Database

DROP Database

- Syntax:

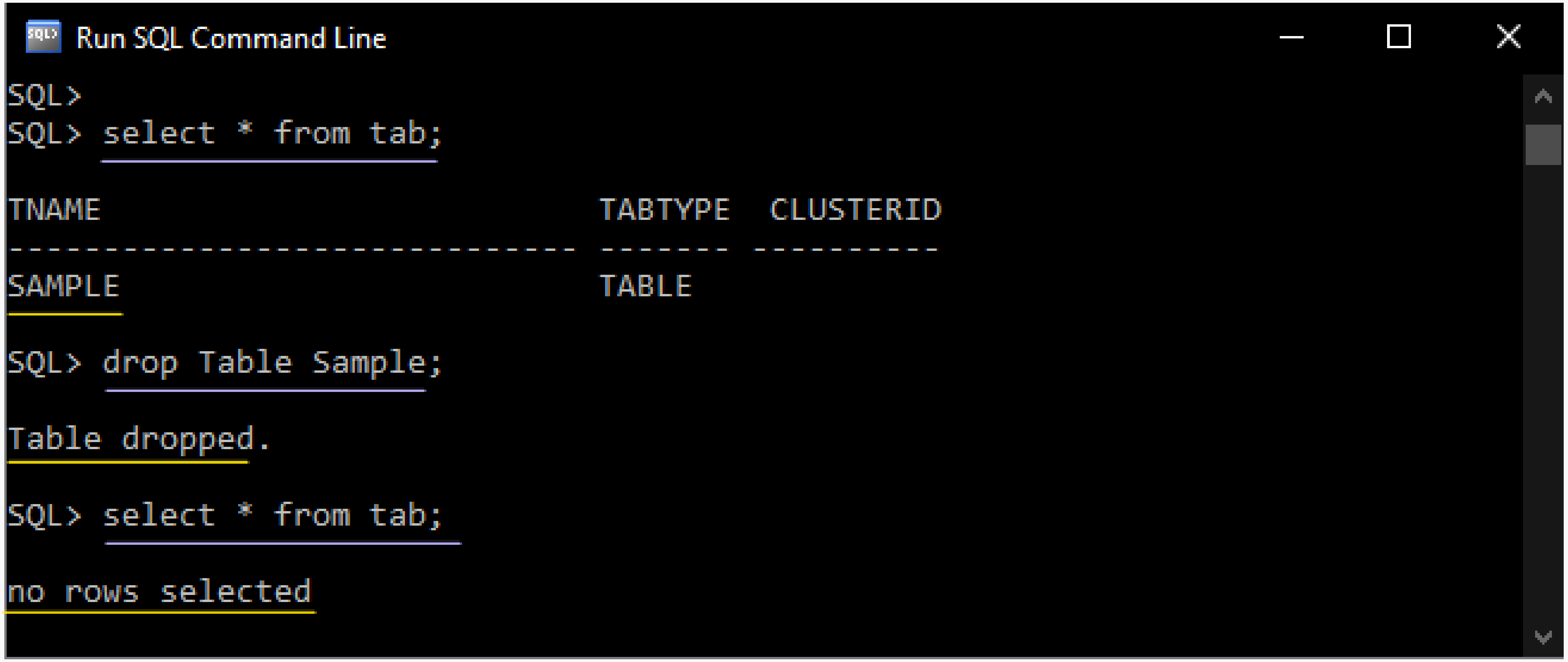
  DROP DATABASE database_name;

DROP Table

- Syntax:

```
DROP TABLE table_name;
```

Drop table



```
SQL>
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
SAMPLE                         TABLE

SQL> drop Table Sample;

Table dropped.

SQL> select * from tab;

no rows selected
```

DROP USER

- Syntax:

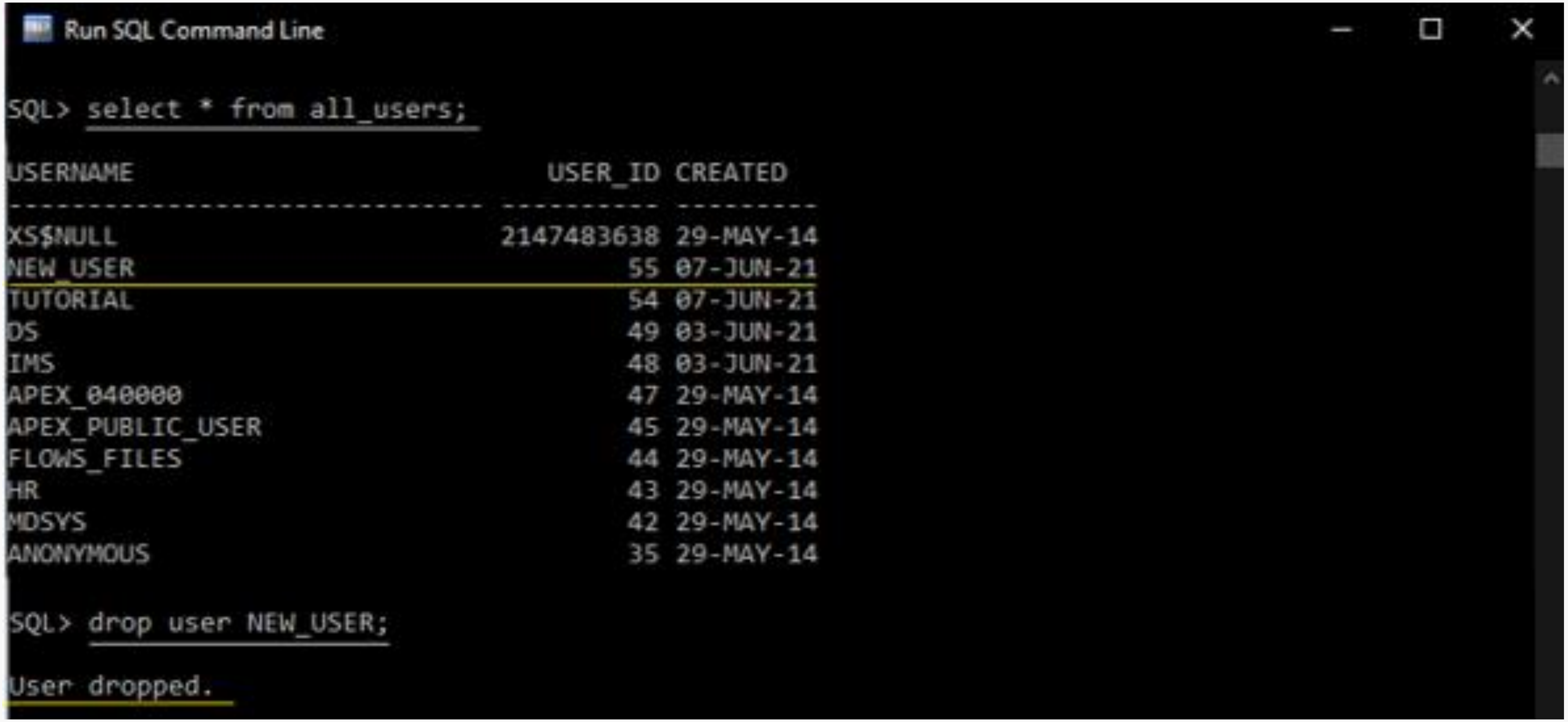
DROP USER user_name;

Note:

If schema contains object, then

DROP USER user_name **CASCADE**;

Drop User



```
SQL> select * from all_users;

USERNAME                          USER_ID CREATED
------------------------------ ---------- ---------
XS$NULL                        2147483638 29-MAY-14
NEW_USER                               55 07-JUN-21
TUTORIAL                               54 07-JUN-21
DS                                     49 03-JUN-21
IMS                                    48 03-JUN-21
APEX_040000                            47 29-MAY-14
APEX_PUBLIC_USER                       45 29-MAY-14
FLOWS_FILES                            44 29-MAY-14
HR                                     43 29-MAY-14
MDSYS                                  42 29-MAY-14
ANONYMOUS                              35 29-MAY-14

SQL> drop user NEW_USER;

User dropped.
```

ALTER

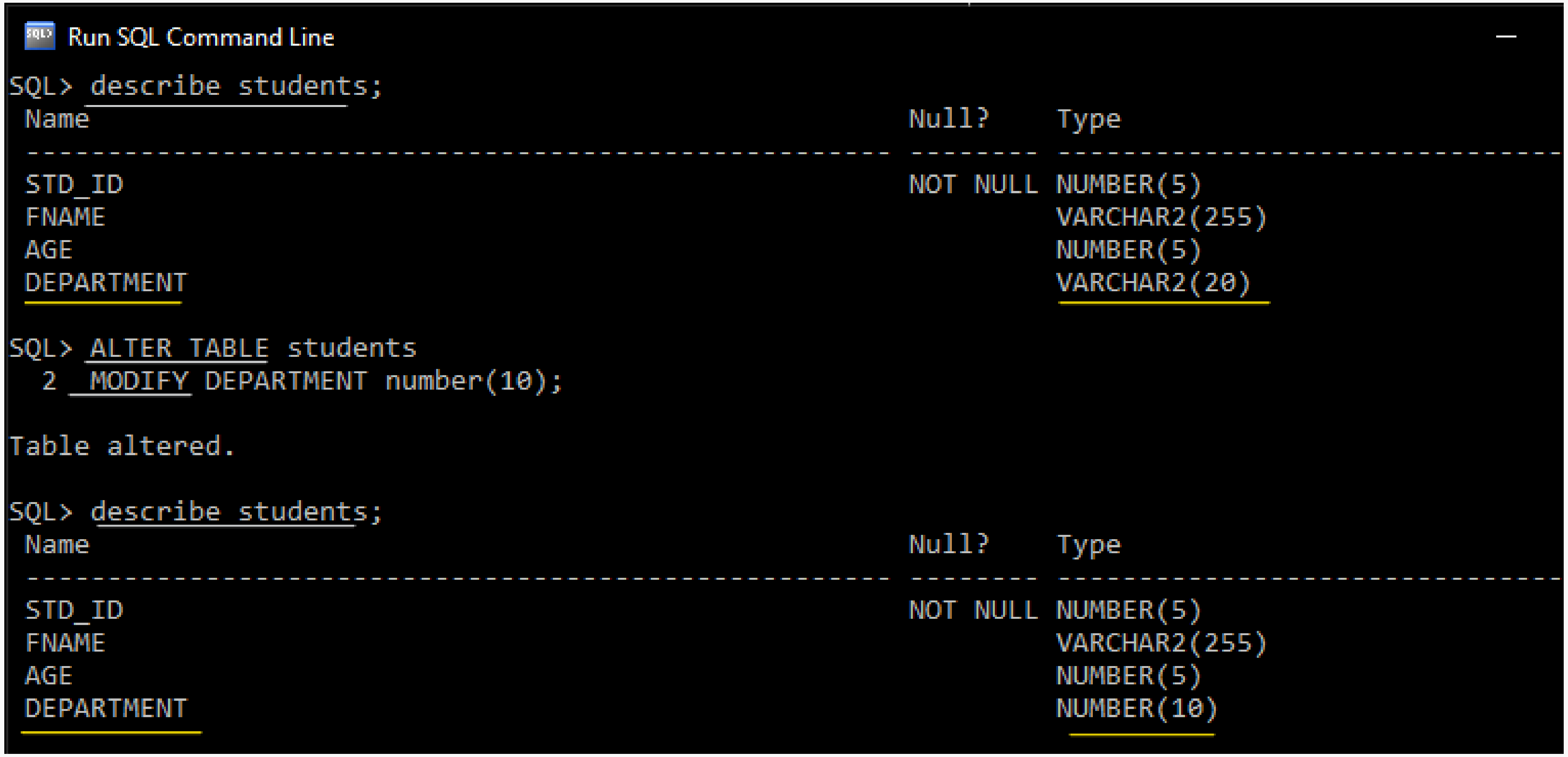
- It is used to make changes to the schema of the table.
    - Columns can be added
    - Columns can be removed
    - Datatype of the column can be changed.
    - Name of the table can be renamed.

ALTER – Modify Column

- Syntax:

```
ALTER TABLE table_name
MODIFY column_name {data_ype};
```

ALTER – Modify Column



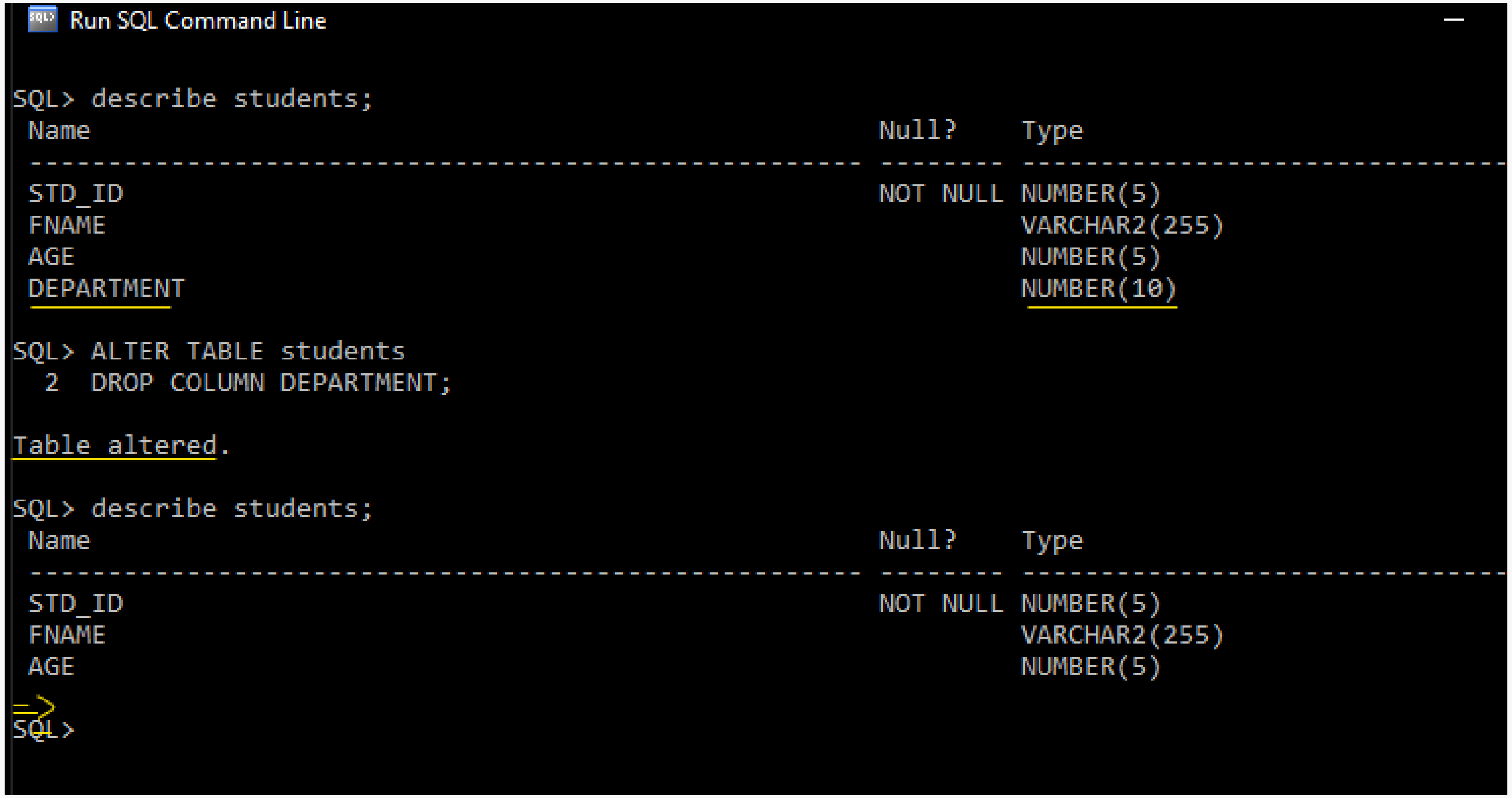
```
SQL> describe students;
 Name                                                   Null?    Type
 ------------------------------------------------------ -------- --------------------------------
 STD_ID                                                 NOT NULL NUMBER(5)
 FNAME                                                           VARCHAR2(255)
 AGE                                                             NUMBER(5)
 DEPARTMENT                                                      VARCHAR2(20)

SQL> ALTER TABLE students
  2  MODIFY DEPARTMENT number(10);

Table altered.

SQL> describe students;
 Name                                                   Null?    Type
 ------------------------------------------------------ -------- --------------------------------
 STD_ID                                                 NOT NULL NUMBER(5)
 FNAME                                                           VARCHAR2(255)
 AGE                                                             NUMBER(5)
 DEPARTMENT                                                      NUMBER(10)
```

ALTER – Drop Column

- Syntax:

```
ALTER TABLE table_name
DROP column_name;
```

ALTER – Drop Column



Run SQL Command Line

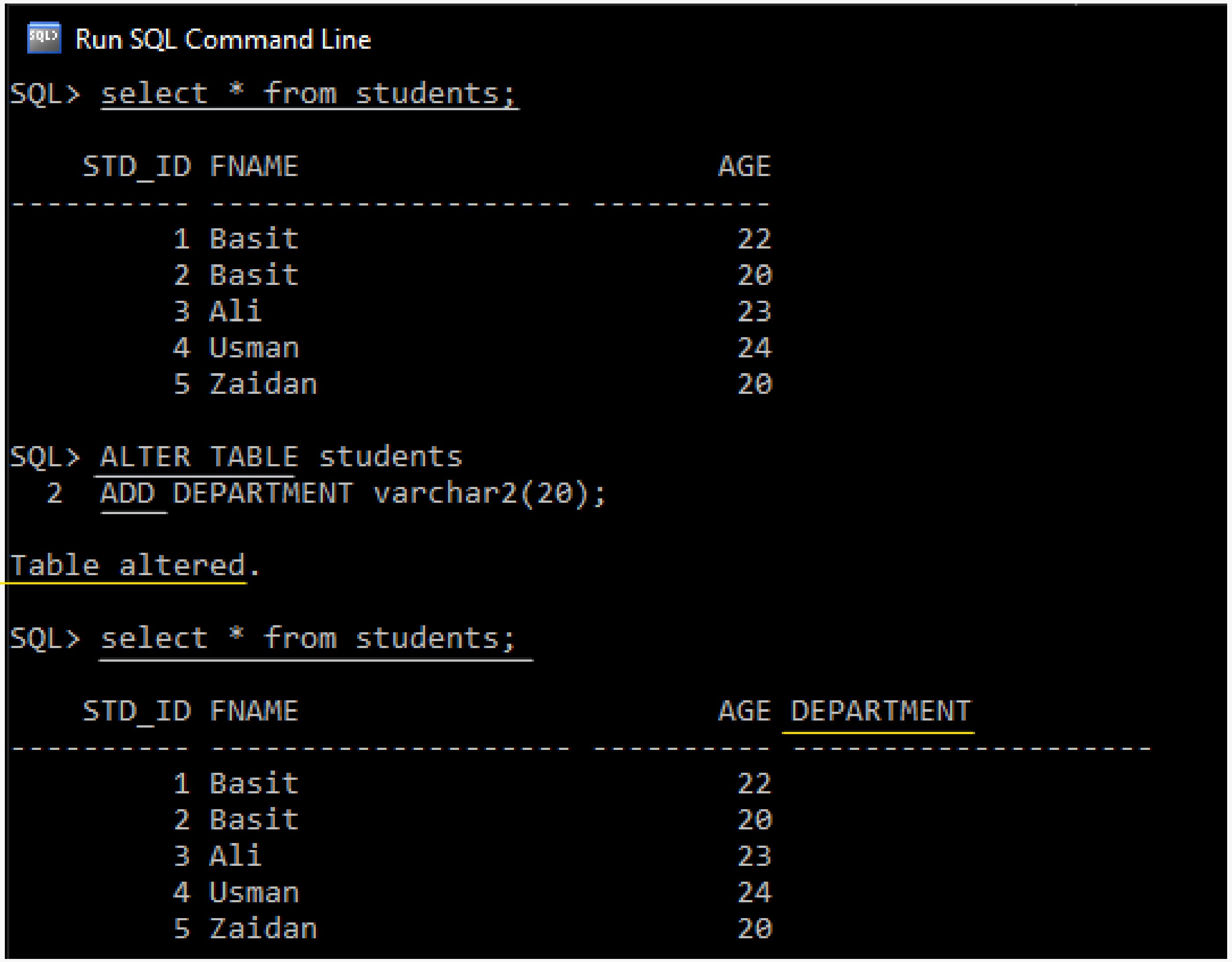
```
SQL> describe students;
 Name                                                      Null?    Type
 ----------------------------------------------------- -------- ------------------------------
 STD_ID                                                    NOT NULL NUMBER(5)
 FNAME                                                              VARCHAR2(255)
 AGE                                                                NUMBER(5)
 DEPARTMENT                                                         NUMBER(10)

SQL> ALTER TABLE students
  2  DROP COLUMN DEPARTMENT;

Table altered.

SQL> describe students;
 Name                                                      Null?    Type
 ----------------------------------------------------- -------- ------------------------------
 STD_ID                                                    NOT NULL NUMBER(5)
 FNAME                                                              VARCHAR2(255)
 AGE                                                                NUMBER(5)

SQL>
```

ALTER – Add Column

- Syntax:

```
ALTER TABLE table_name
ADD column_name {data_ype};
```

ALTER – Add Column


Run SQL Command Line

```
SQL> select * from students;

    STD_ID FNAME                       AGE
---------- -------------------- ----------
         1 Basit                        22
         2 Basit                        20
         3 Ali                          23
         4 Usman                        24
         5 Zaidan                       20

SQL> ALTER TABLE students
  2  ADD DEPARTMENT varchar2(20);

Table altered.

SQL> select * from students;

    STD_ID FNAME                       AGE DEPARTMENT
---------- -------------------- ---------- --------------------
         1 Basit                        22
         2 Basit                        20
         3 Ali                          23
         4 Usman                        24
         5 Zaidan                       20
```

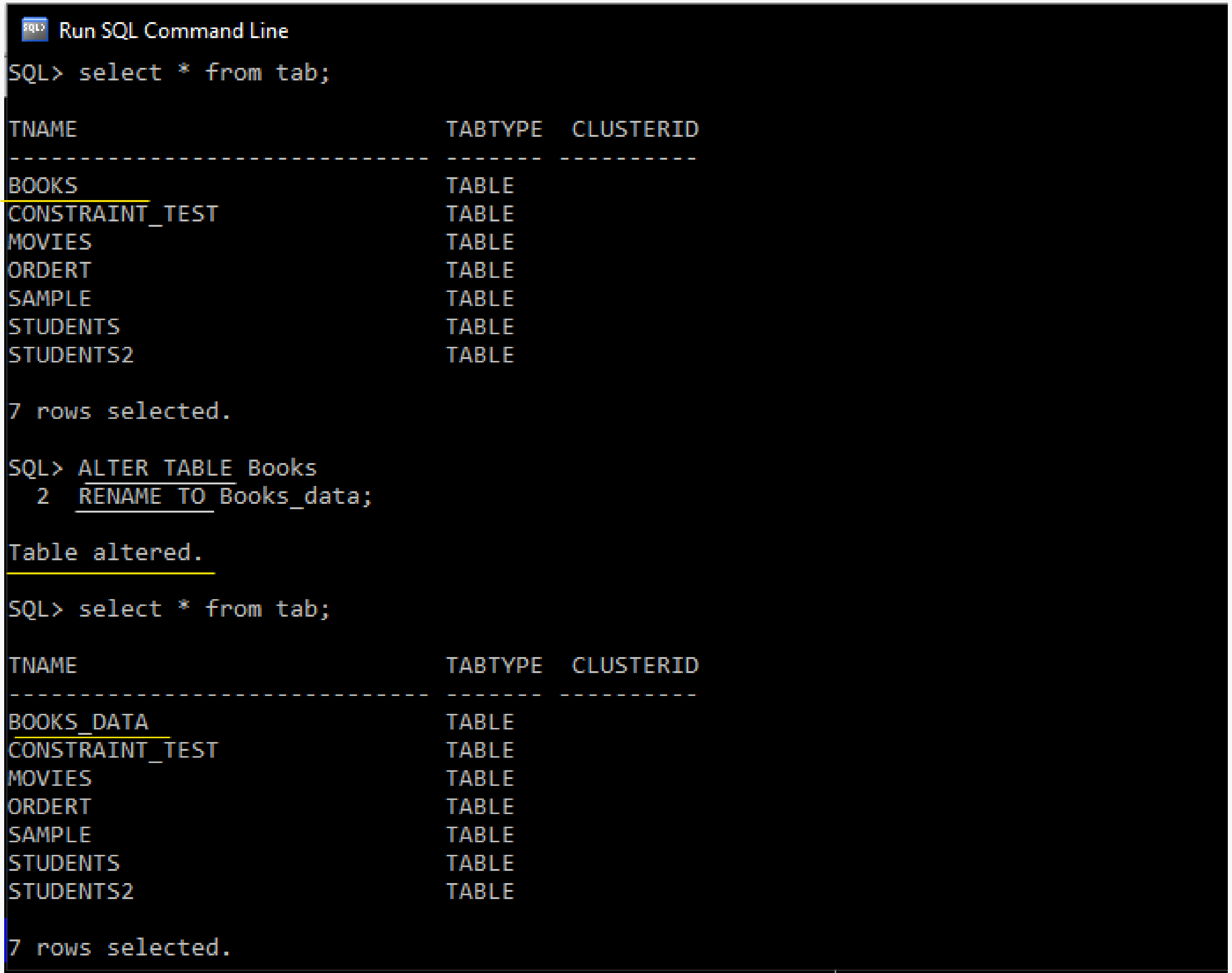
ALTER – Rename Table

- Syntax:

ALTER TABLE table_name

RENAME TO new_table_name;

ALTER – Rename Table



Run SQL Command Line

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
BOOKS                          TABLE
CONSTRAINT_TEST                TABLE
MOVIES                         TABLE
ORDERT                         TABLE
SAMPLE                         TABLE
STUDENTS                       TABLE
STUDENTS2                      TABLE

7 rows selected.

SQL> ALTER TABLE Books
  2  RENAME TO Books_data;

Table altered.

SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
BOOKS_DATA                     TABLE
CONSTRAINT_TEST                TABLE
MOVIES                         TABLE
ORDERT                         TABLE
SAMPLE                         TABLE
STUDENTS                       TABLE
STUDENTS2                      TABLE

7 rows selected.
```

SQL-Data Retrieval Language (DRL)

SELECT

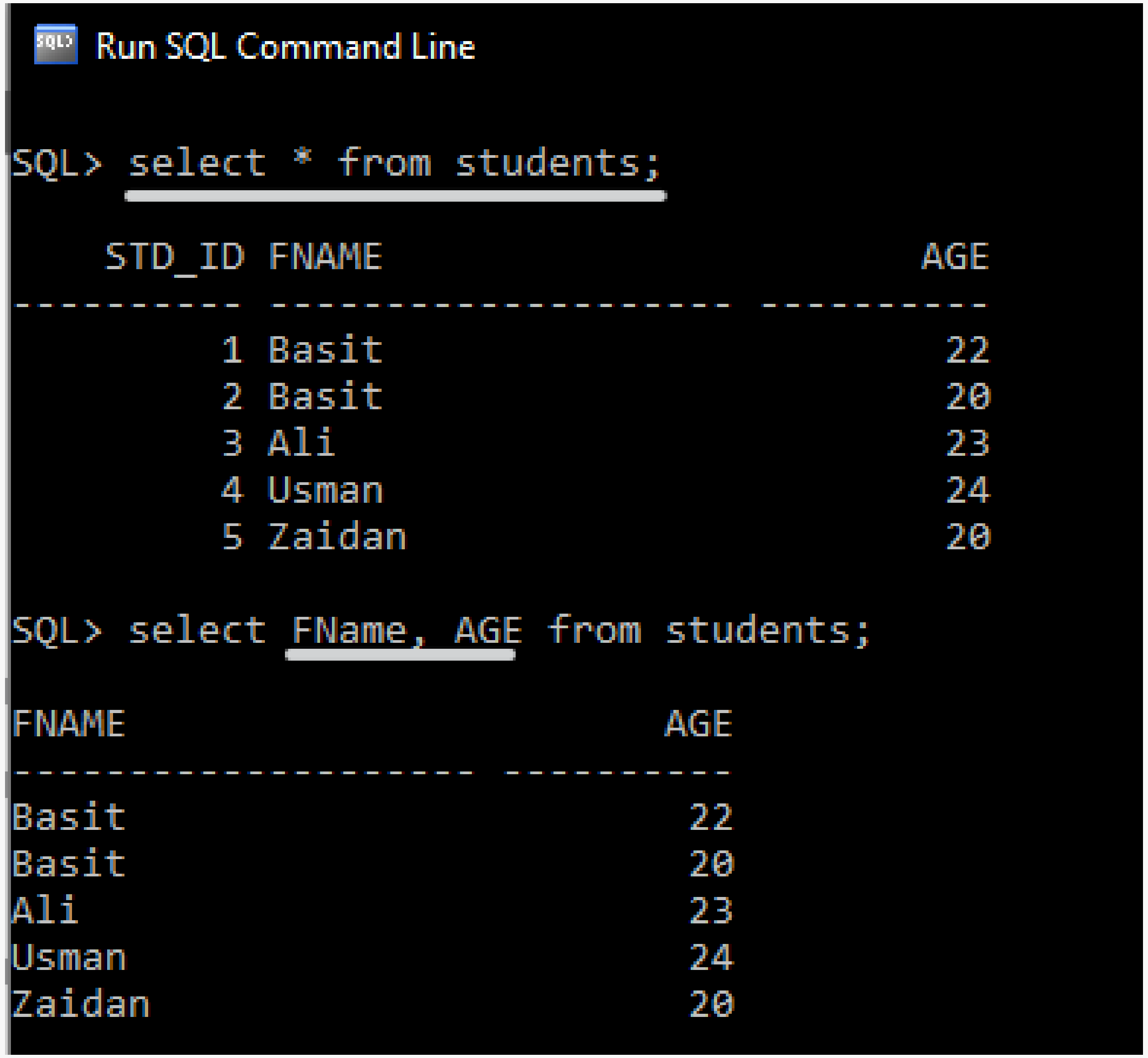
- SELECT identifies the columns to be displayed.
- FROM identifies the table containing those columns.

- Syntax:

  SELECT column1, column2,....,columnN

  FROM table_name;

Select- single and multiple columns


Run SQL Command Line

```
SQL> select * from students;

    STD_ID FNAME                                 AGE
---------- -------------------- ----------
         1 Basit                                  22
         2 Basit                                  20
         3 Ali                                    23
         4 Usman                                  24
         5 Zaidan                                 20

SQL> select FName, AGE from students;

FNAME                                 AGE
-------------------- ----------
Basit                                  22
Basit                                  20
Ali                                    23
Usman                                  24
Zaidan                                 20
```

Select Statement Using Arithmetic Operators

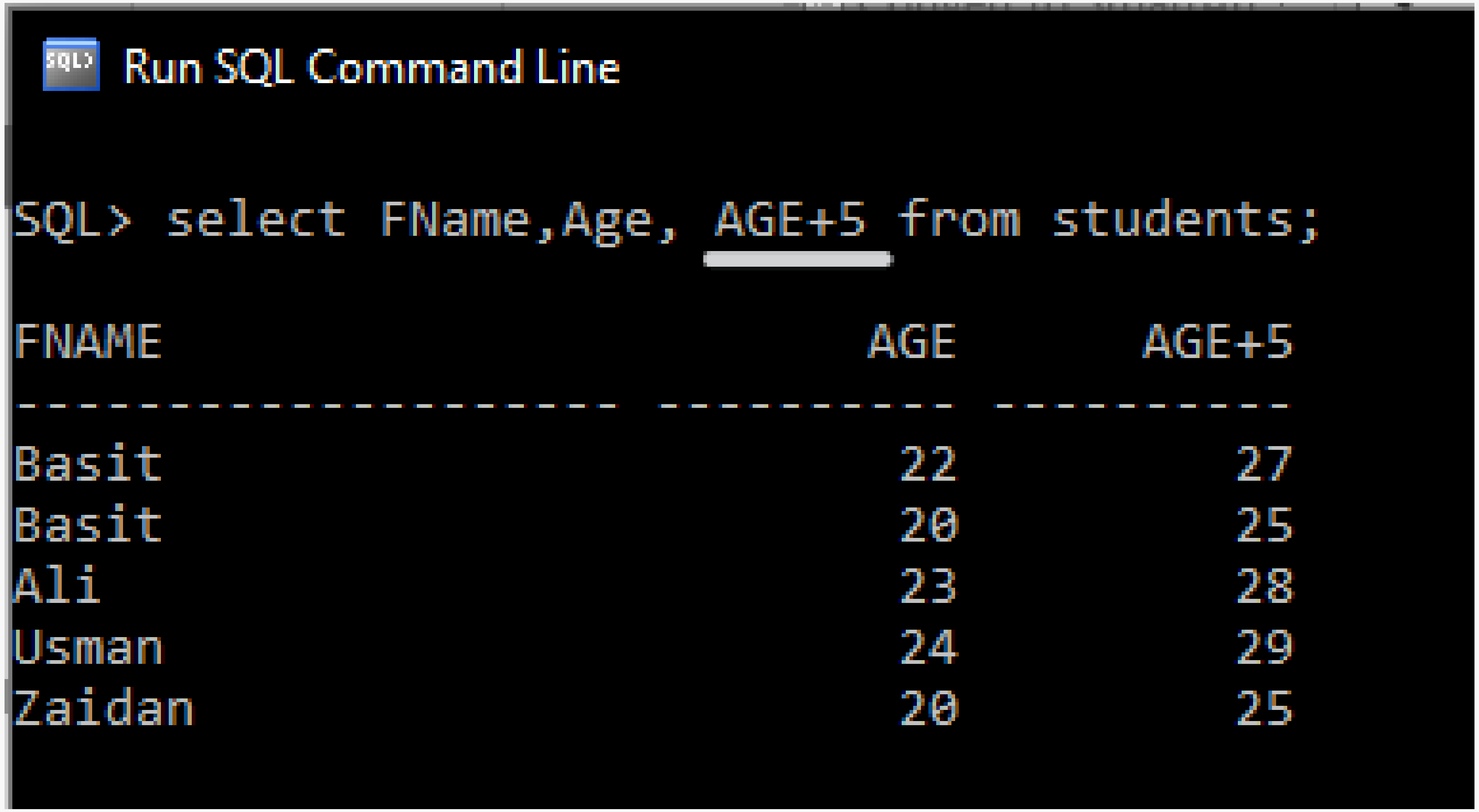
- Syntax:

SELECT column [operator]<expression>,…

FROM table_name;

Operator	Meaning	Operates on
+ (Add)	Addition	Numeric value
- (Subtract)	Subtraction	Numeric value
* (Multiply)	Multiplication	Numeric value
/ (Divide)	Division	Numeric value
% (Modulo)	Returns the integer remainder of a division. For example, 17 % 5 = 2 because the remainder of 17 divided by 5 is 2.	Numeric value

Select- using arithmetic operator


Run SQL Command Line

```
SQL> select FName,Age, AGE+5 from students;

FNAME                          AGE      AGE+5
-------------------- ---------- ----------
Basit                           22         27
Basit                           20         25
Ali                             23         28
Usman                           24         29
Zaidan                          20         25
```

Operator Precedence

Operator	Meaning	Precedence
( )	Addition	Highest
* / %	Subtraction	
+ -	Multiplication	
=	Division	Lowest

Note: Arithmetic expressions containing a null value evaluate to null.

Select Statement
Using Column Aliases

- SQL aliases are used to give a temporary name to column in a table.
- Aliases are often used to make column names more readable.
- An alias only exists for the duration of that query.
- An alias is created with the AS keyword.

Syntax:

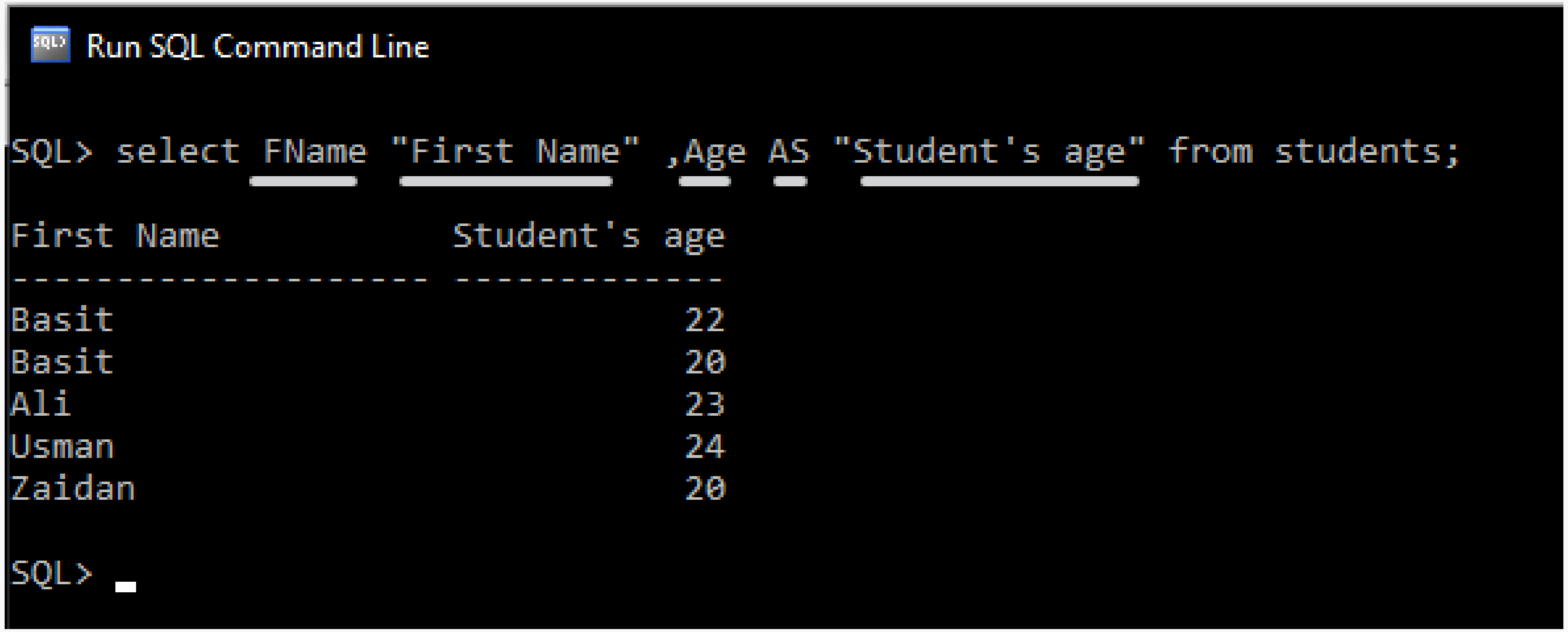
```
SELECT column_name AS alias_name
FROM table_name;
```

- An alias can be created without the AS keyword.

Syntax:

```
SELECT column_name  alias_name
FROM table_name;
```

Select- Using Column Aliases



```
SQL> select FName "First Name" ,Age AS "Student's age" from students;

First Name            Student's age
--------------------- -------------
Basit                            22
Basit                            20
Ali                              23
Usman                            24
Zaidan                           20

SQL>
```

Select Statement Concatenation Operator

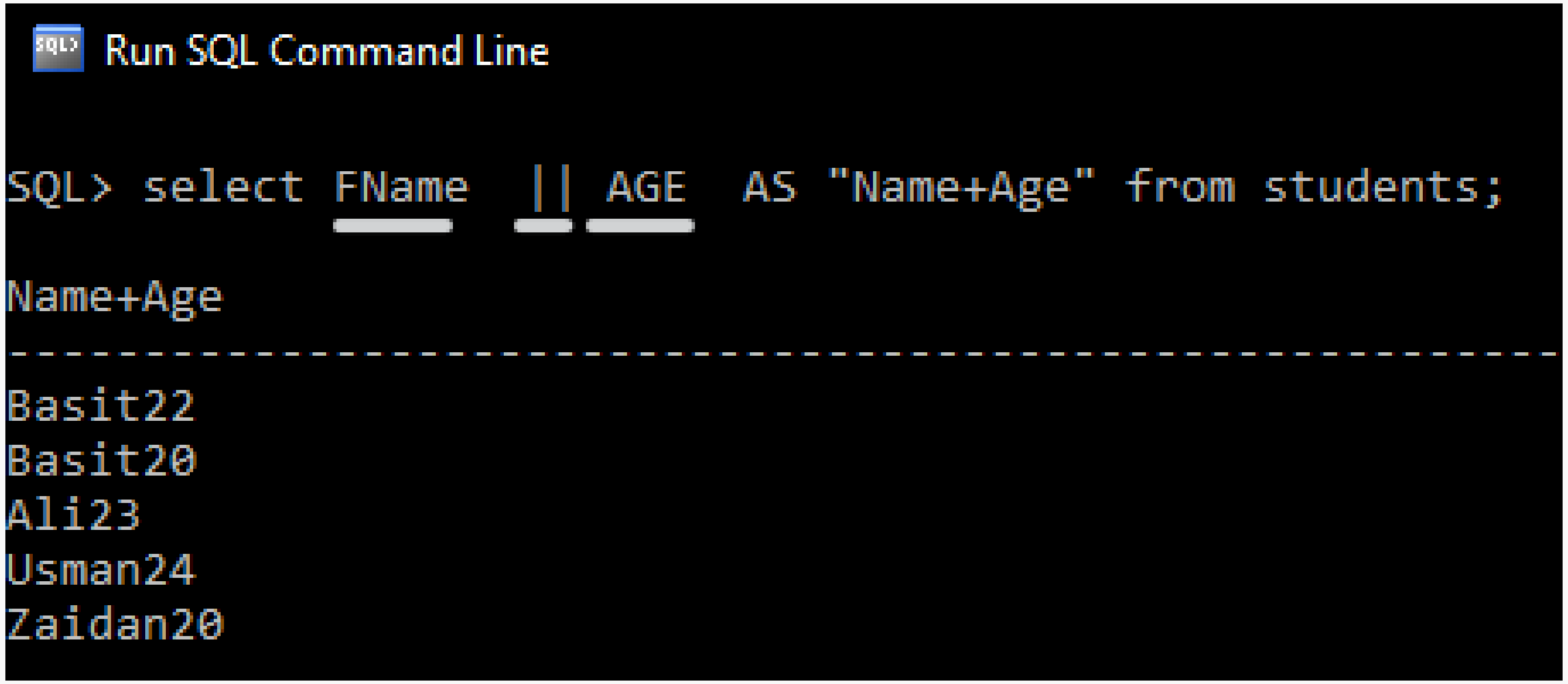
- It links columns or character strings to other columns
- It is represented by two vertical bars (||)
- It creates a resultant column that is a character expression.

Syntax:

SELECT column_name1 **| |** column_name2

FROM table_name;

Select - Concatenation Operator



```
SQL> select FName  || AGE  AS "Name+Age" from students;

Name+Age
-----------------------------------------------------------------
Basit22
Basit20
Ali23
Usman24
Zaidan20
```

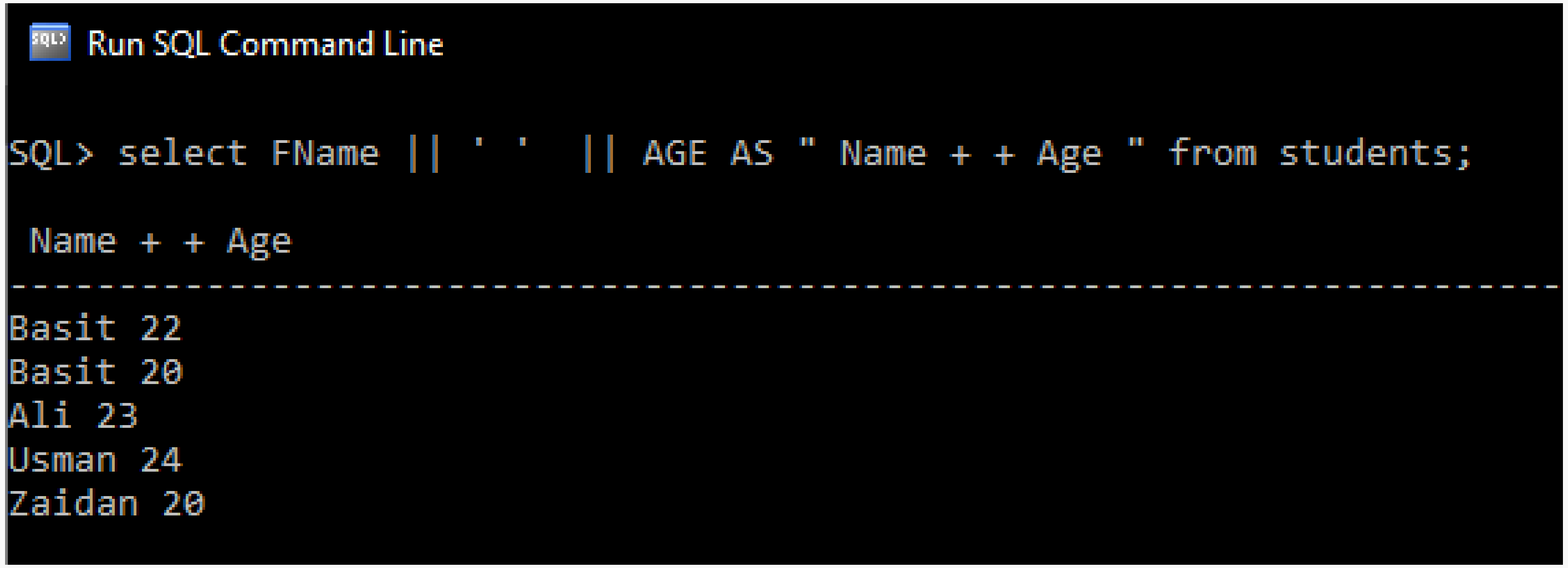
Select Statement
Literal Character Strings

- A literal is a character, a number, or a date that is included in the SELECT statement.
- Date and character literal values must be enclosed within single quotation marks.
- Each character string is output once for each row returned.

Syntax:

```
SELECT column_name1 || ‘String’ || column_name2
FROM table_name;
```

Select - Literal Character Strings


Run SQL Command Line

```
SQL> select FName || ' '  || AGE AS " Name + + Age " from students;

 Name + + Age
----------------------------------------------------------------------
Basit 22
Basit 20
Ali 23
Usman 24
Zaidan 20
```

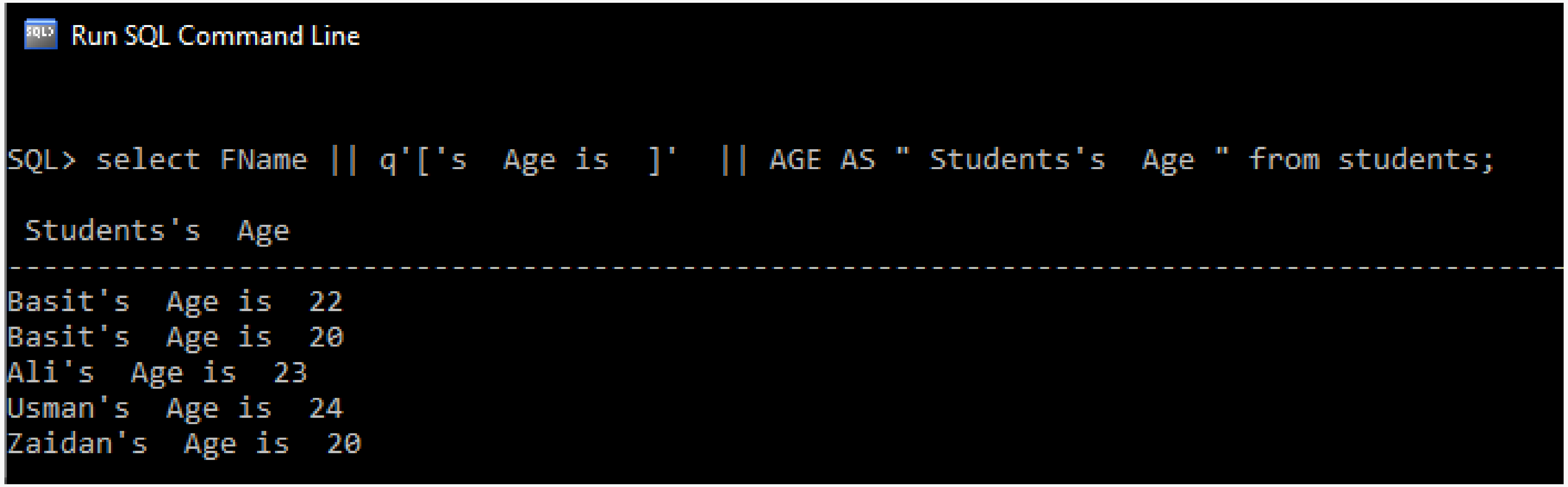
Select Statement
Alternative Quote (q) Operator

- Specify your own quotation mark delimiter.
- Select any delimiter.
- Increase readability and usability.

Syntax:

```
SELECT column_name1 || q‘ [ 's string ] ' || column_name2
FROM table_name;
```

Select - Alternative Quote (q) Operator



Run SQL Command Line

```
SQL> select FName || q'['s  Age is  ]'  || AGE AS " Students's  Age " from students;

 Students's  Age
------------------------------------------------------------------------------------
Basit's  Age is  22
Basit's  Age is  20
Ali's  Age is  23
Usman's  Age is  24
Zaidan's  Age is  20
```

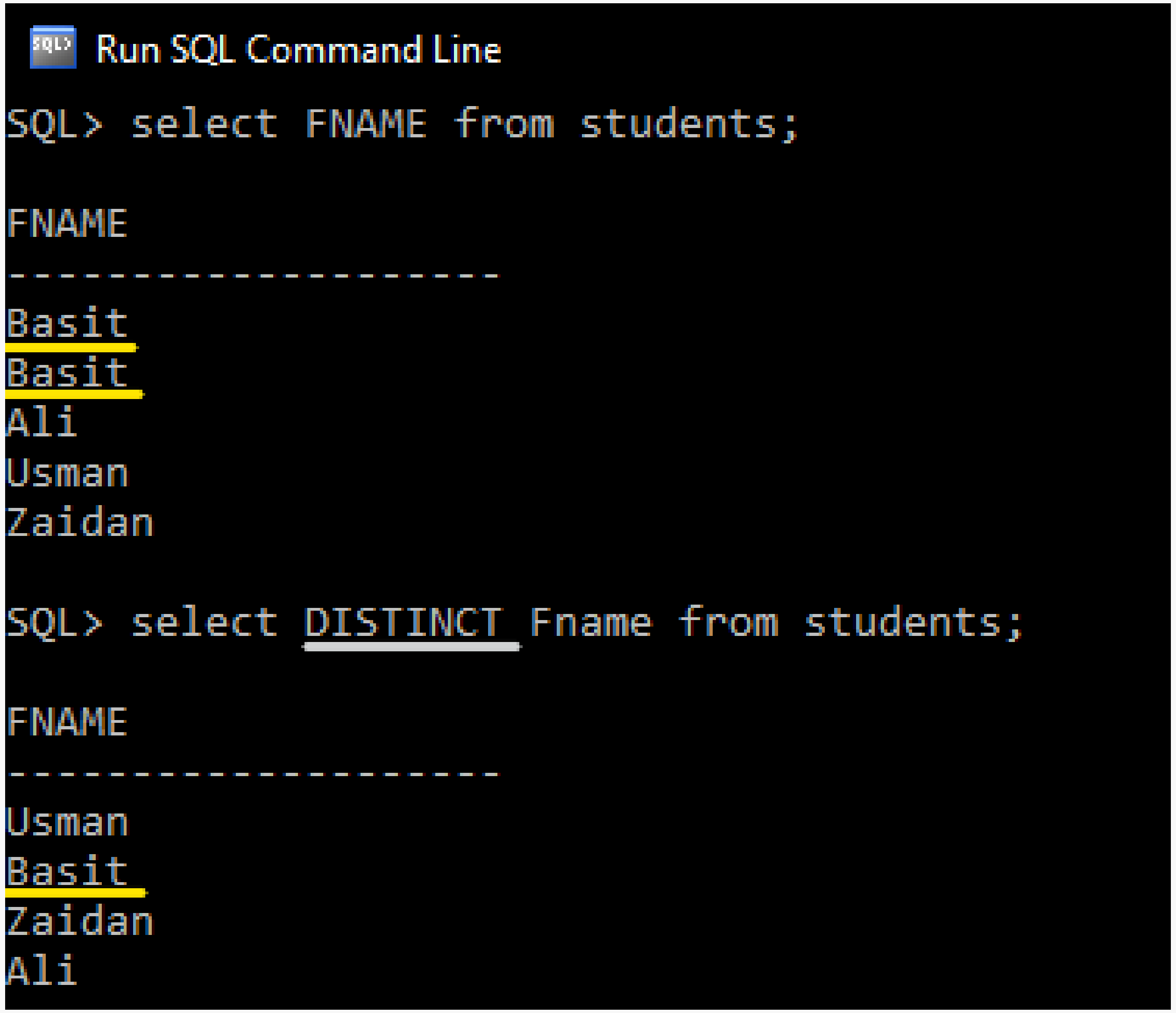
Select Statement Avoid Duplicate Rows

- The default display of queries is all rows, including duplicate rows.
- To avoid Duplicate rows, Use keyword **DISTINCT.**

Syntax:

```
SELECT  DISTINCT   column_name1
FROM table_name;
```

Select – Avoid Duplicate Rows



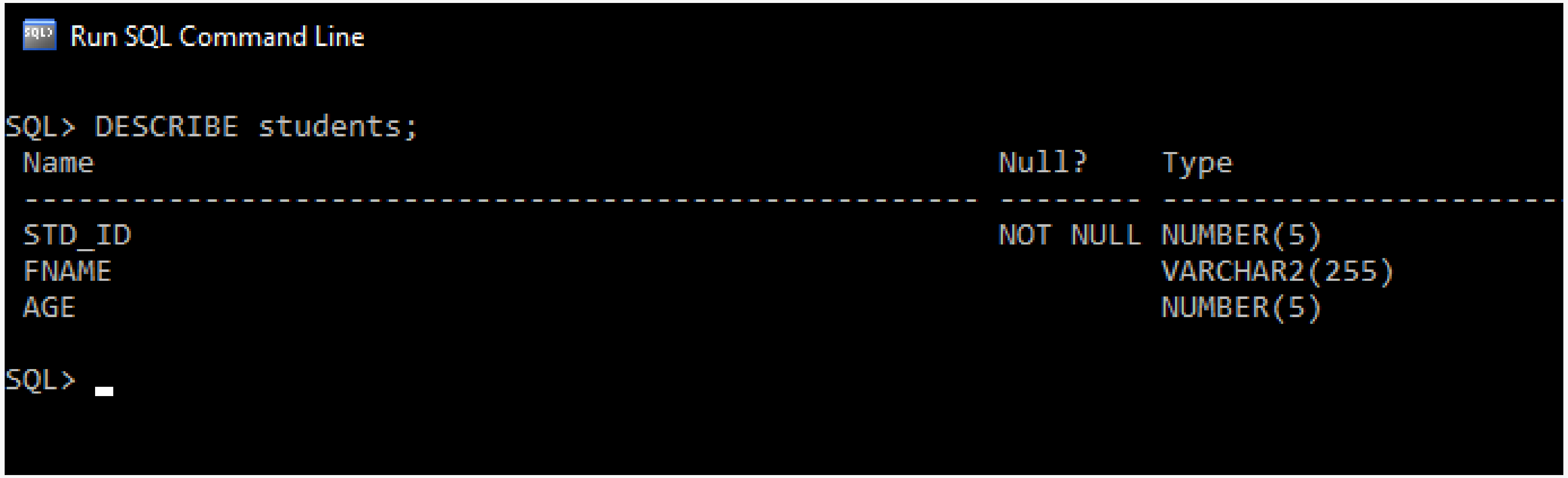
```
SQL> select FNAME from students;

FNAME
--------------------
Basit
Basit
Ali
Usman
Zaidan

SQL> select DISTINCT Fname from students;

FNAME
--------------------
Usman
Basit
Zaidan
Ali
```

Displaying the Table Structure

- Use the DESCRIBE command to display the structure of a table.

Syntax:

DESCRIBE table_name;


Run SQL Command Line

```
SQL> DESCRIBE students;
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ----------------------
 STD_ID                                                NOT NULL NUMBER(5)
 FNAME                                                          VARCHAR2(255)
 AGE                                                            NUMBER(5)

SQL> _
```

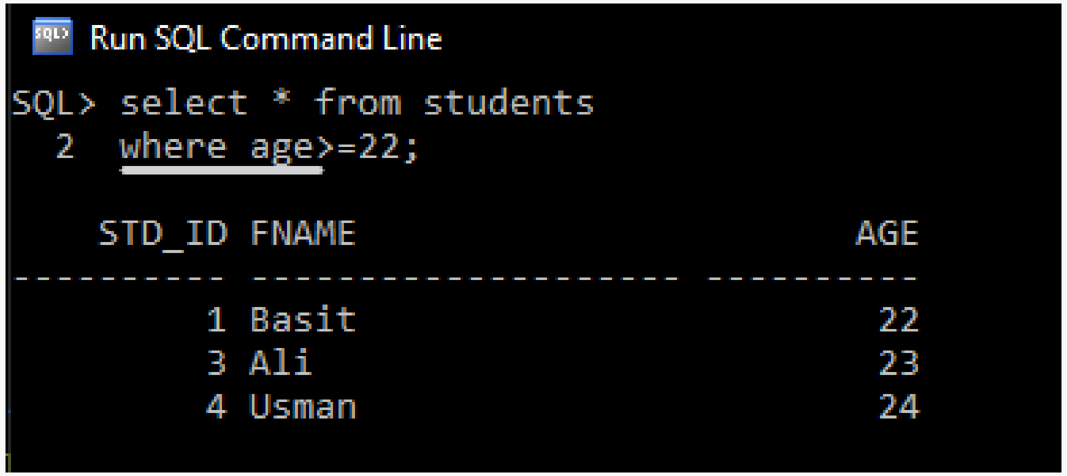
Restricting and Sorting Data

Using the WHERE Clause

- Use the WHERE clause to limit the rows that are selected.

Syntax:

SELECT *column1*, *column2, ...*
FROM *table_name*
WHERE *condition*;

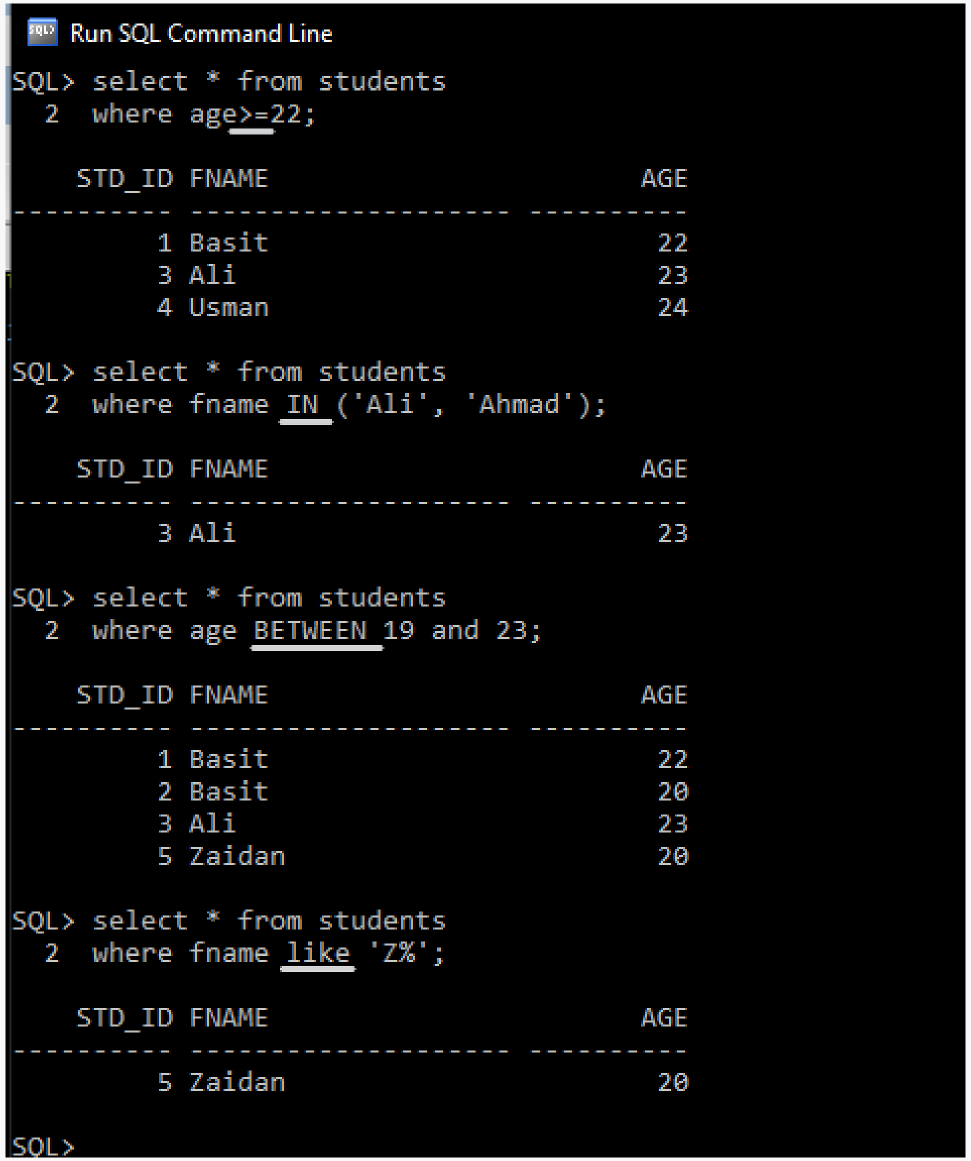


```
SQL> select * from students
  2  where age>=22;

    STD_ID FNAME                       AGE
---------- -------------------- ----------
         1 Basit                        22
         3 Ali                          23
         4 Usman                        24
```

Operators in WHERE clause

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. Note: In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

WHERE Clause



Run SQL Command Line

```
SQL> select * from students
  2  where age>=22;

    STD_ID FNAME                       AGE
---------- -------------------- ----------
         1 Basit                        22
         3 Ali                          23
         4 Usman                        24

SQL> select * from students
  2  where fname IN ('Ali', 'Ahmad');

    STD_ID FNAME                       AGE
---------- -------------------- ----------
         3 Ali                          23

SQL> select * from students
  2  where age BETWEEN 19 and 23;

    STD_ID FNAME                       AGE
---------- -------------------- ----------
         1 Basit                        22
         2 Basit                        20
         3 Ali                          23
         5 Zaidan                       20

SQL> select * from students
  2  where fname like 'Z%';

    STD_ID FNAME                       AGE
---------- -------------------- ----------
         5 Zaidan                       20

SQL>
```

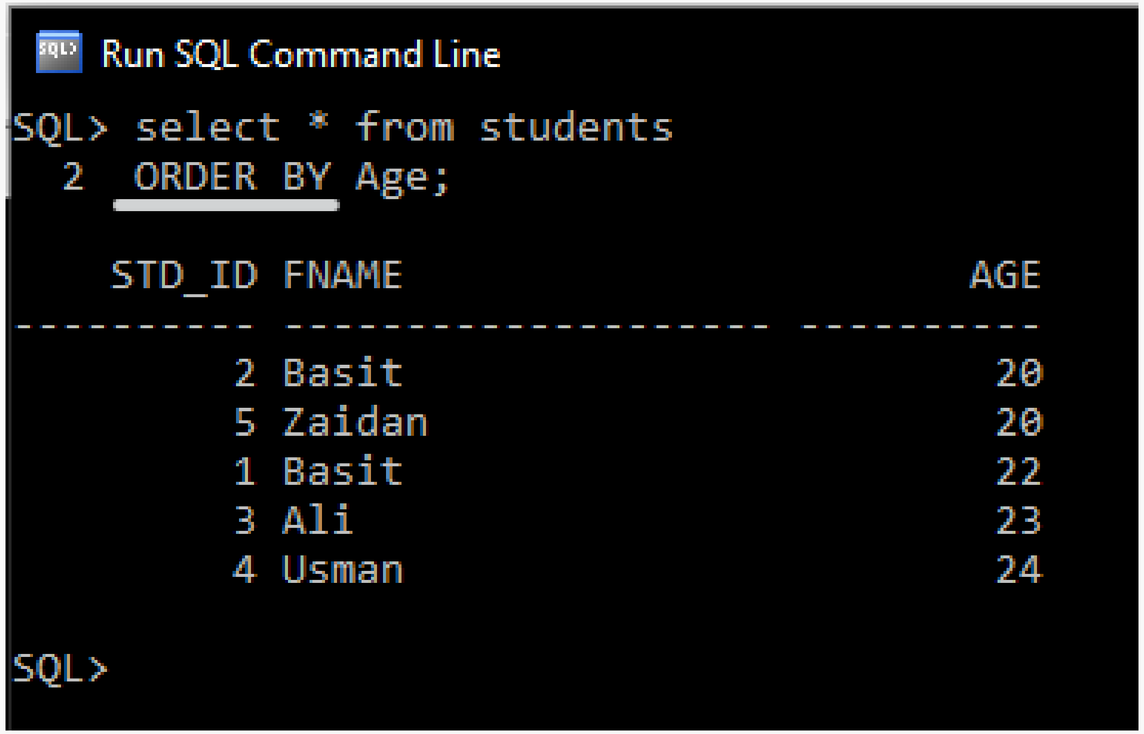
Using the ORDER BY Clause

- Sort retrieved rows with the ORDER BY clause:
    - **ASC:** Ascending order, default
    - **DESC:** Descending order
- The ORDER BY clause comes last in the SELECT statement.

Syntax:

```
SELECT    column1, column2, ...
FROM       table_name
ORDER BY     column;
```

ORDER BY



```
SQL> select * from students
  2  ORDER BY Age;

    STD_ID FNAME                                 AGE
---------- -------------------- ----------
         2 Basit                                  20
         5 Zaidan                                 20
         1 Basit                                  22
         3 Ali                                    23
         4 Usman                                  24

SQL>
```

Substitution Variables

- Use substitution variables to:
    - Temporarily store values with single-ampersand (&) and
- double-ampersand (&&) substitution
- Use substitution variables to supplement the following:
    - WHERE conditions
    - ORDER BY clauses
    - Column expressions
    - Table names

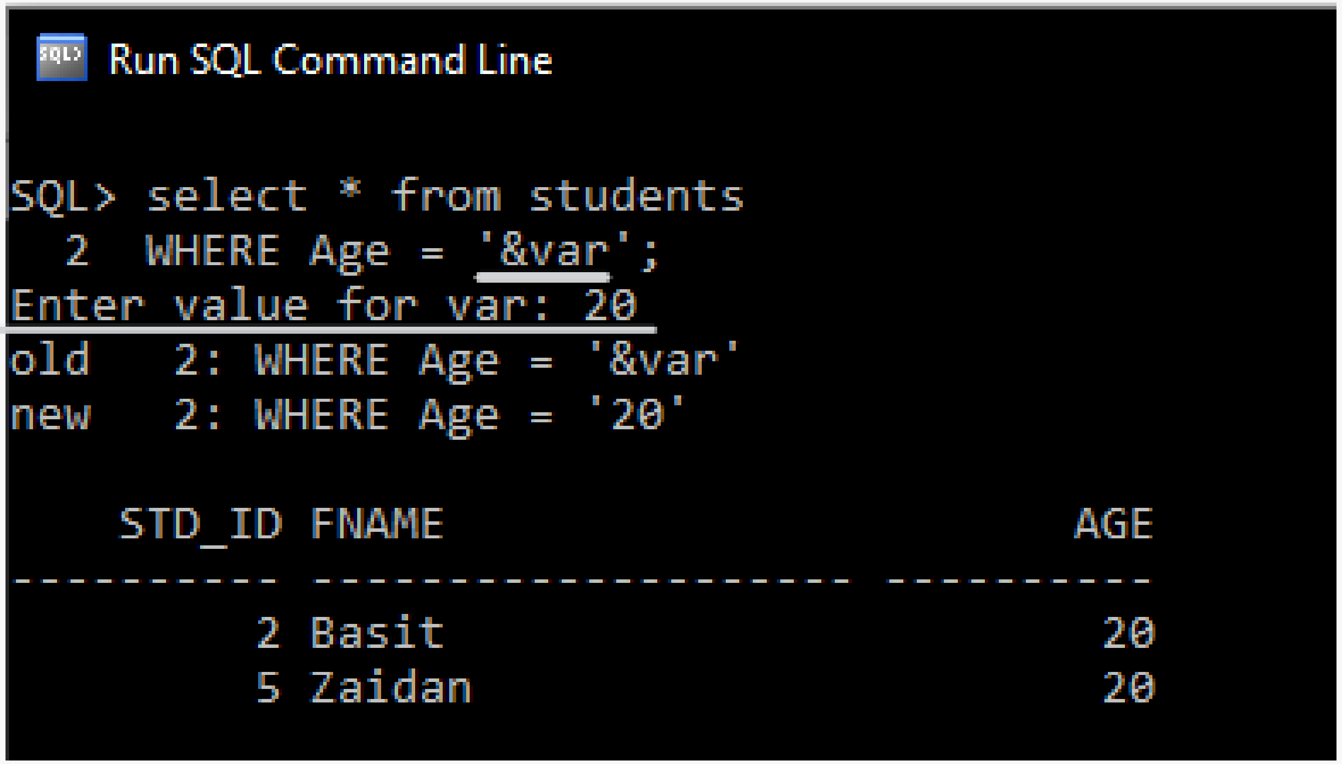
Using the Single-Ampersand Substitution Variable

- Use a variable prefixed with an ampersand (&) to prompt the user for a value:

Syntax:

```
SELECT    column1, column2, ...
FROM       table_name
Where     column= &variable;
```

Single-Ampersand Substitution Variable



```
SQL> select * from students
  2  WHERE Age = '&var';
Enter value for var: 20
old   2: WHERE Age = '&var'
new   2: WHERE Age = '20'

    STD_ID FNAME                                 AGE
---------- -------------------- ----------
         2 Basit                                  20
         5 Zaidan                                 20
```

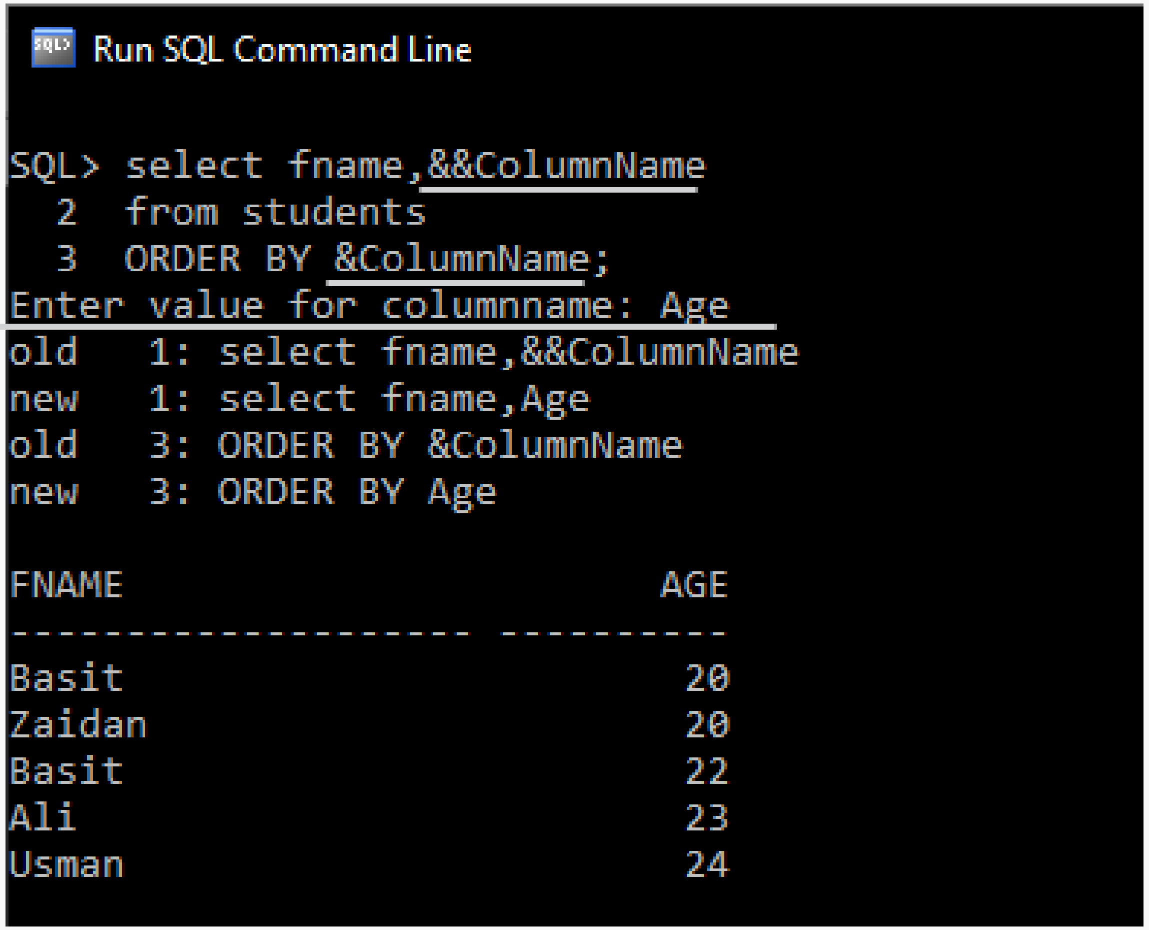
Using the Double-Ampersand Substitution Variable

- Use double ampersand (&&) if you want to reuse the variable value without prompting the user each time.

Syntax:

```
SELECT    column1, &&variable
FROM      table_name
ORDER BY      &variable;
```

Double-Ampersand Substitution Variable



```
SQL> select fname,&&ColumnName
  2  from students
  3  ORDER BY &ColumnName;
Enter value for columnname: Age
old   1: select fname,&&ColumnName
new   1: select fname,Age
old   3: ORDER BY &ColumnName
new   3: ORDER BY Age

FNAME                       AGE
-------------------- ----------
Basit                        20
Zaidan                       20
Basit                        22
Ali                          23
Usman                        24
```

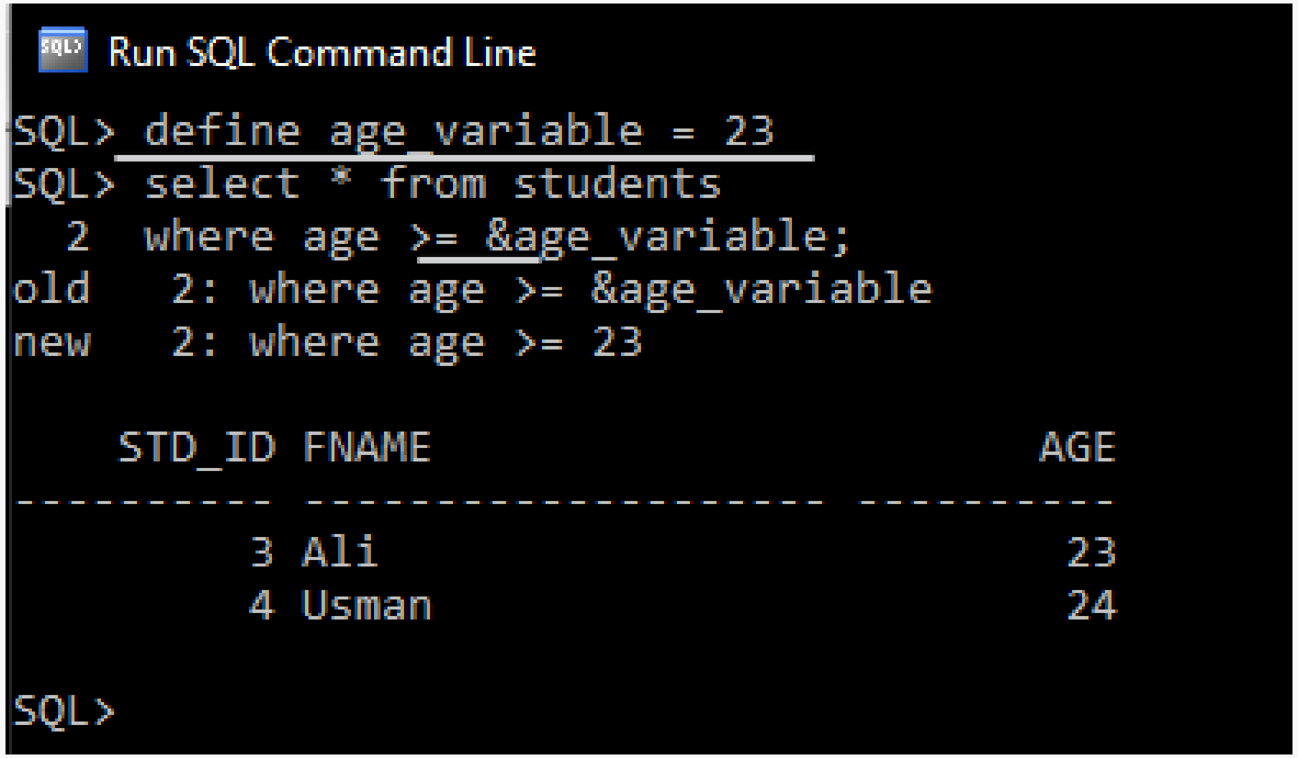
Using the DEFINE Command

- Use the DEFINE command to create and assign a value to a variable.
- Use the UNDEFINE command to remove a variable.
- Syntax:

```
DEFINE    variable =value
SELECT    column1, &&variable
FROM      table_name
Where     column= &variable;
```

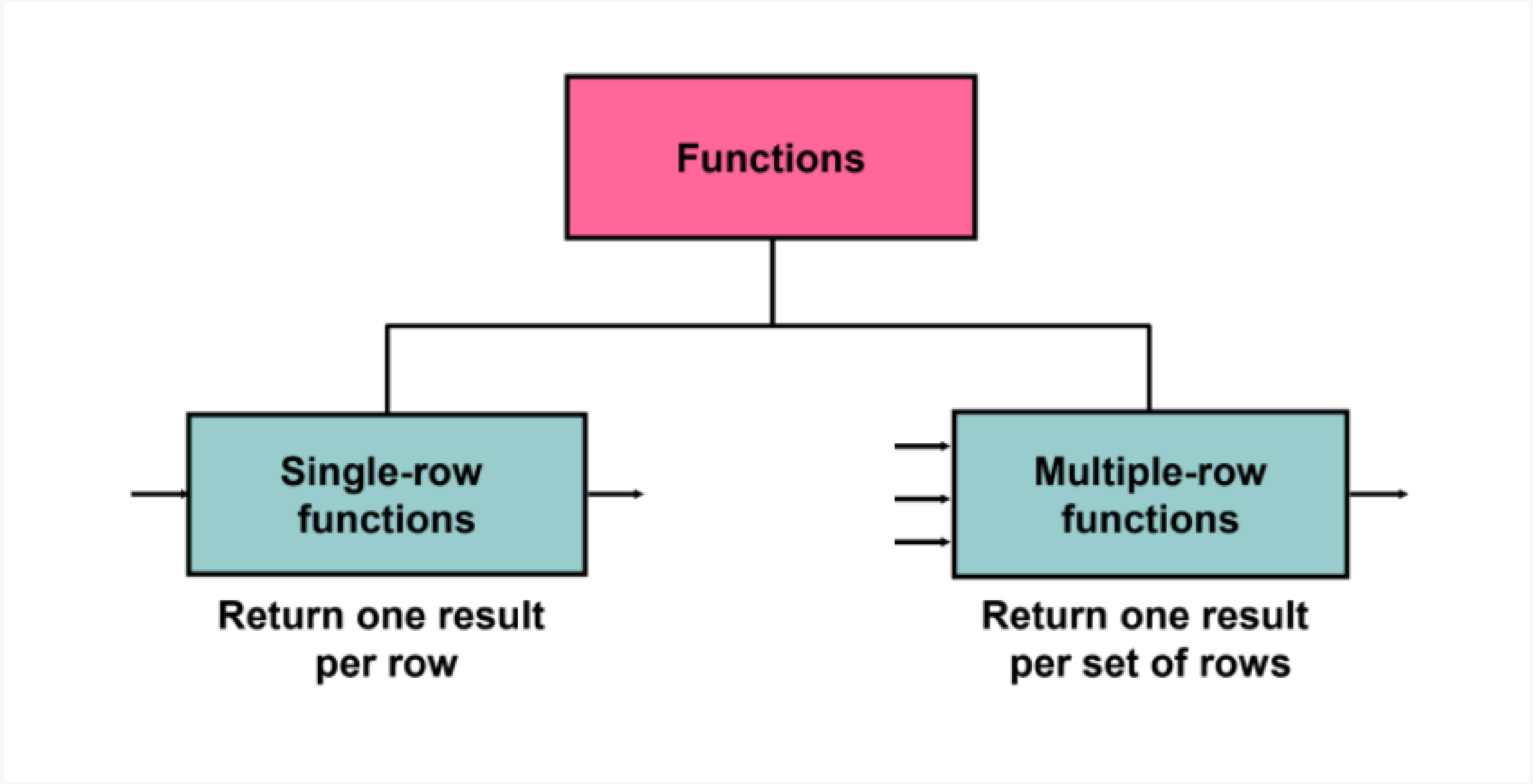
;

Define Command



```
SQL> define age_variable = 23
SQL> select * from students
  2  where age >= &age_variable;
old   2: where age >= &age_variable
new   2: where age >= 23

    STD_ID FNAME                                 AGE
---------- -------------------- ----------
         3 Ali                                    23
         4 Usman                                  24

SQL>
```

Functions in SQL

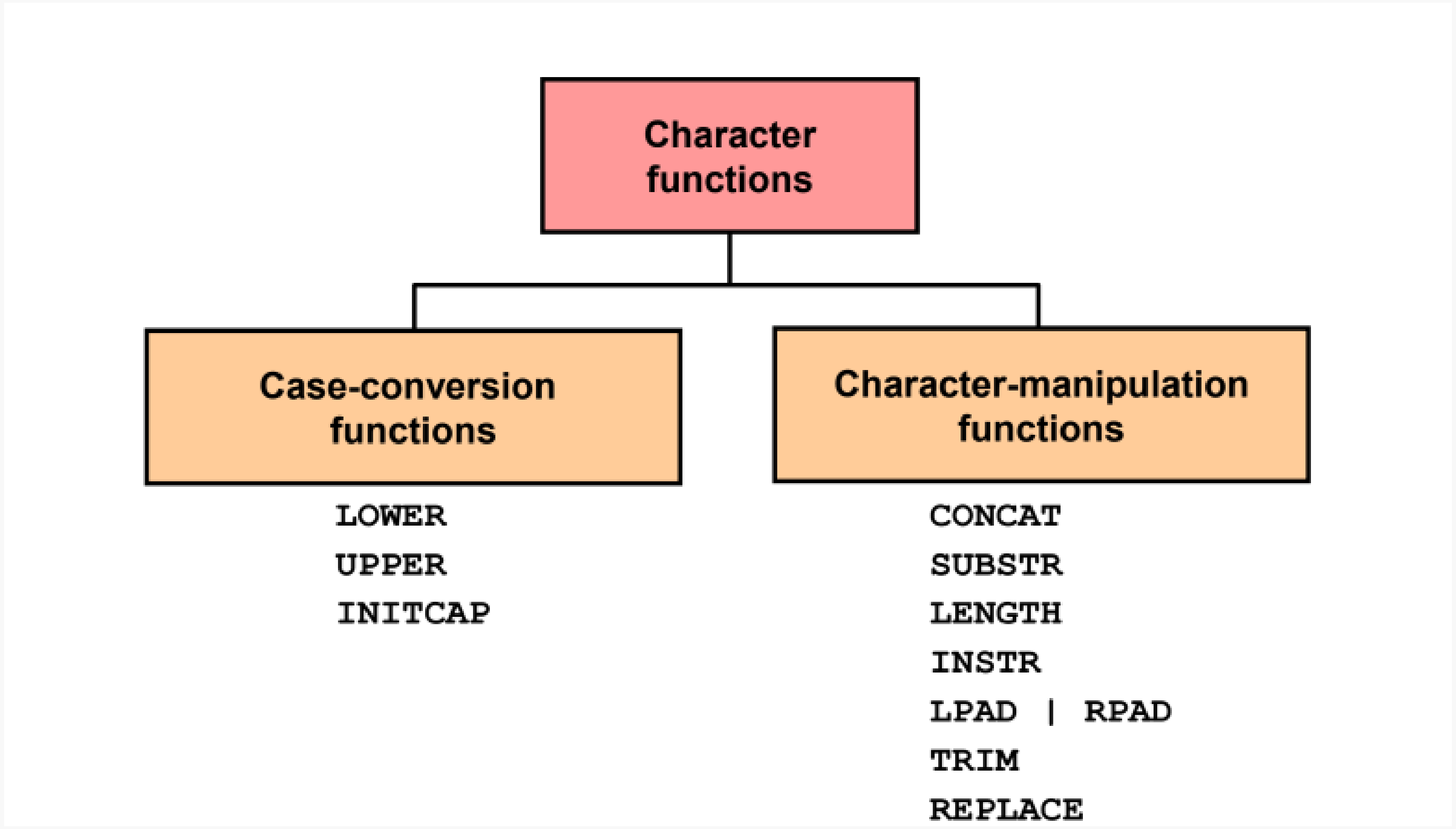


Single-row Functions

Single-row Functions

- Single row functions are those function that give one result per row.
- The single row functions are:
    - Character Functions
    - Number Functions
    - Date Functions

Character Functions



Case-Conversion Functions

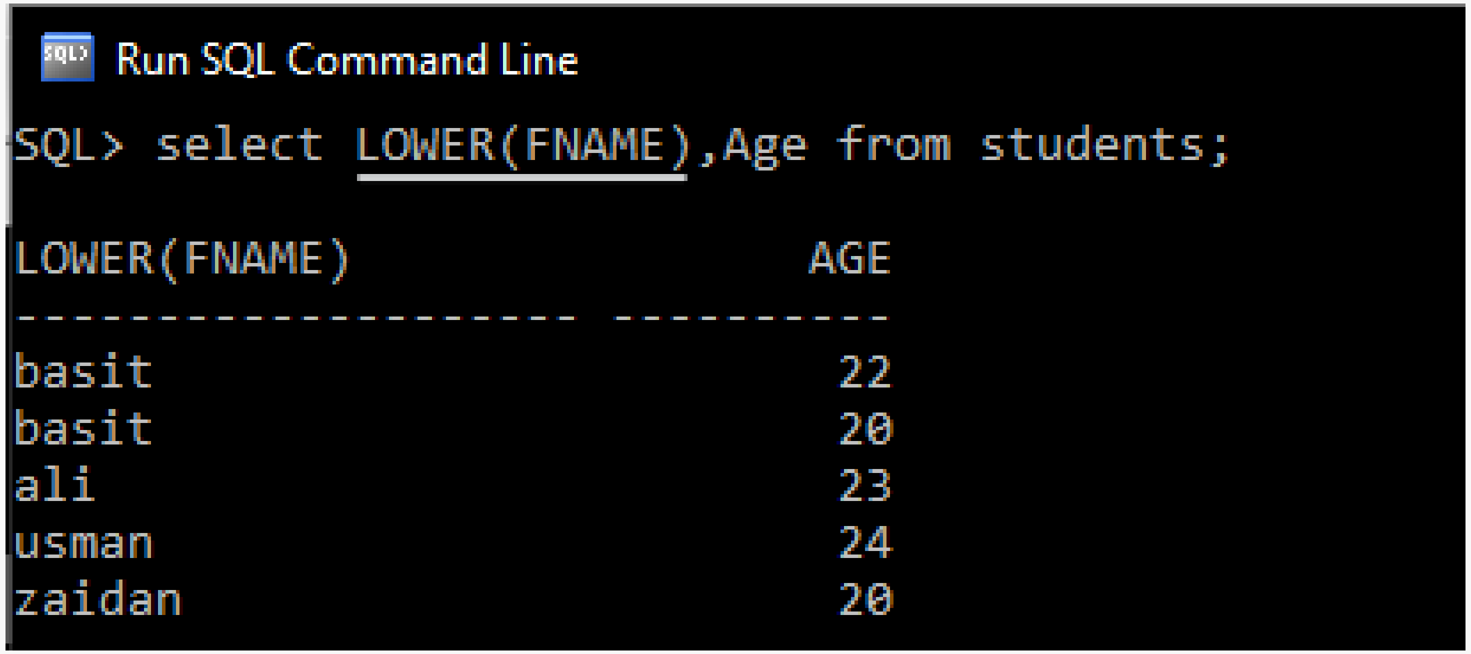
- These functions convert the case for character strings:

Function	Result
LOWER('SQL Course')	sql course
INITCAP('SQL Course')	Sql Course
UPPER('SQL Course')	SQL COURSE

Syntax:

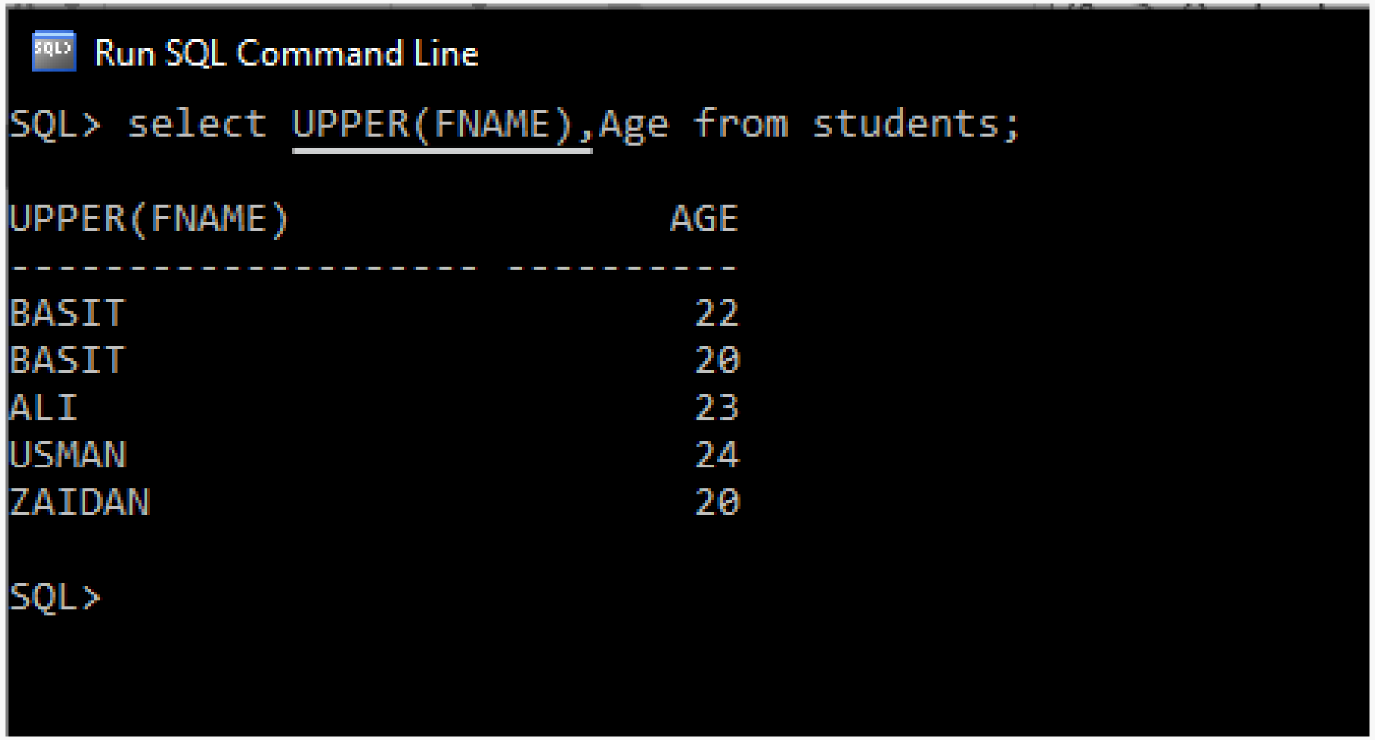
```
SELECT    column1, column2, ...
FROM       table_name
Where     Function(column) = value;
```

Case Conversion - LOWER



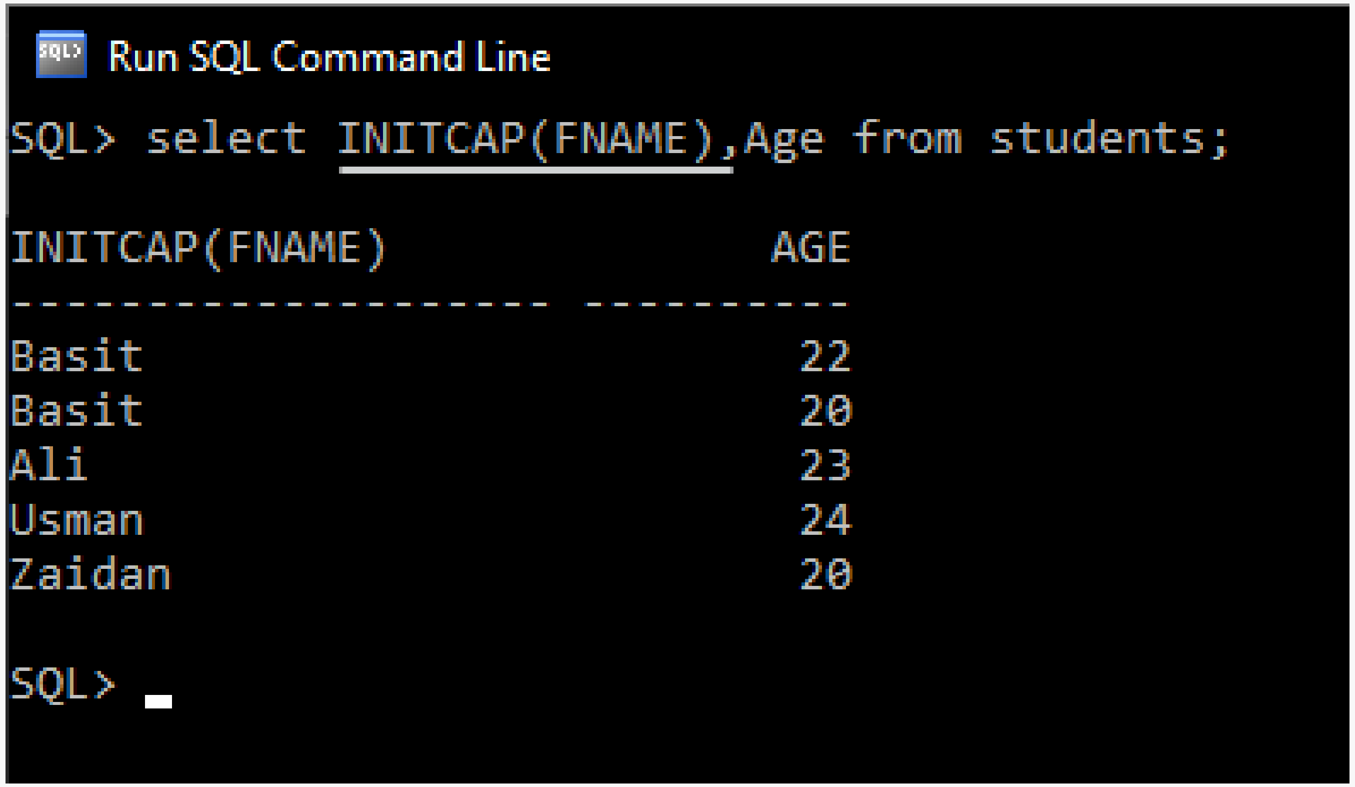
```
SQL> select LOWER(FNAME),Age from students;

LOWER(FNAME)                         AGE
-------------------- ----------
basit                                 22
basit                                 20
ali                                   23
usman                                 24
zaidan                                20
```

Case Conversion - UPPER


Run SQL Command Line

```
SQL> select UPPER(FNAME),Age from students;

UPPER(FNAME)                AGE
-------------------- ----------
BASIT                        22
BASIT                        20
ALI                          23
USMAN                        24
ZAIDAN                       20

SQL>
```

Case Conversion - INITCAP



```
SQL> select INITCAP(FNAME),Age from students;

INITCAP(FNAME)                  AGE
-------------------- ----------
Basit                            22
Basit                            20
Ali                              23
Usman                            24
Zaidan                           20

SQL> _
```

Character-Manipulation Functions

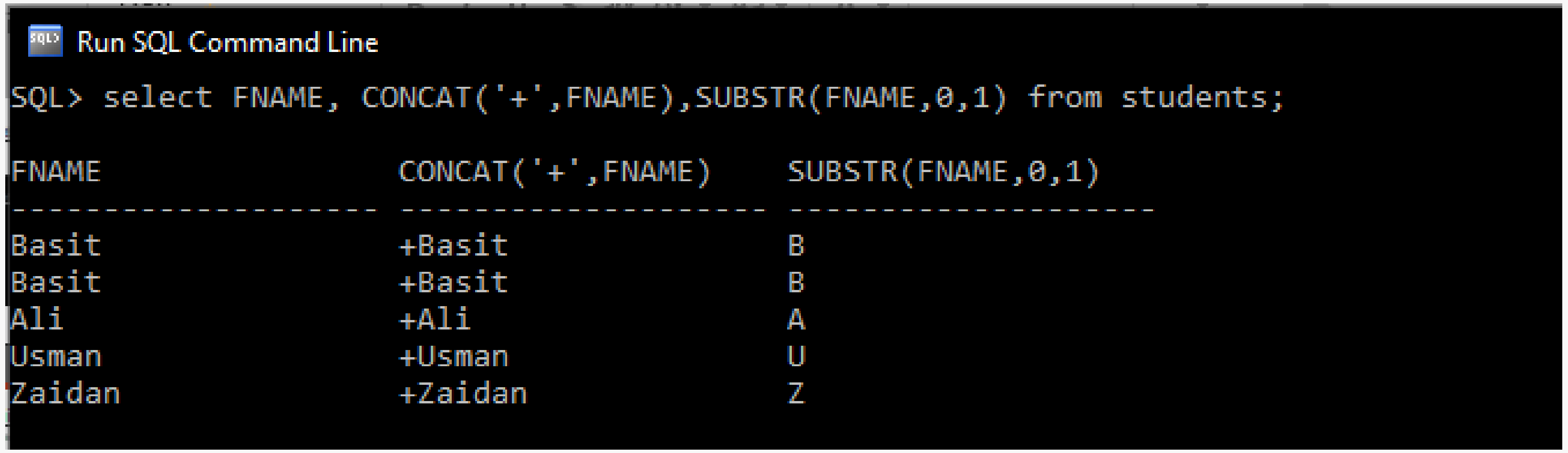
- These functions manipulate character strings:

Function	Result
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
REPLACE ('JACK and JUE','J','BL')	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld

Syntax:

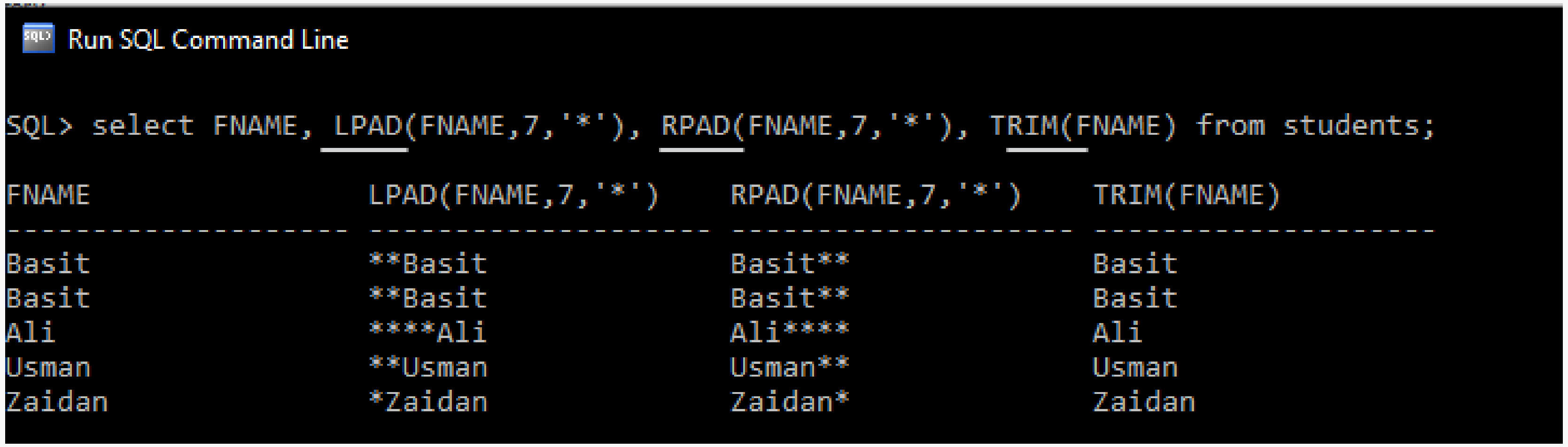
```
SELECT    column1, Function(column2) , ...
FROM      table_name
Where     Function(column)= value;
```

Character-Manipulation Functions



Run SQL Command Line

```
SQL> select FNAME, CONCAT('+',FNAME),SUBSTR(FNAME,0,1) from students;

FNAME                CONCAT('+',FNAME)    SUBSTR(FNAME,0,1)
-------------------- -------------------- --------------------
Basit                +Basit               B
Basit                +Basit               B
Ali                  +Ali                 A
Usman                +Usman               U
Zaidan               +Zaidan              Z
```



Run SQL Command Line

```
SQL> select FNAME, LPAD(FNAME,7,'*'), RPAD(FNAME,7,'*'), TRIM(FNAME) from students;

FNAME                LPAD(FNAME,7,'*')    RPAD(FNAME,7,'*')    TRIM(FNAME)
-------------------- -------------------- -------------------- --------------------
Basit                **Basit              Basit**              Basit
Basit                **Basit              Basit**              Basit
Ali                  ****Ali              Ali****              Ali
Usman                **Usman              Usman**              Usman
Zaidan               *Zaidan              Zaidan*              Zaidan
```

Number Functions

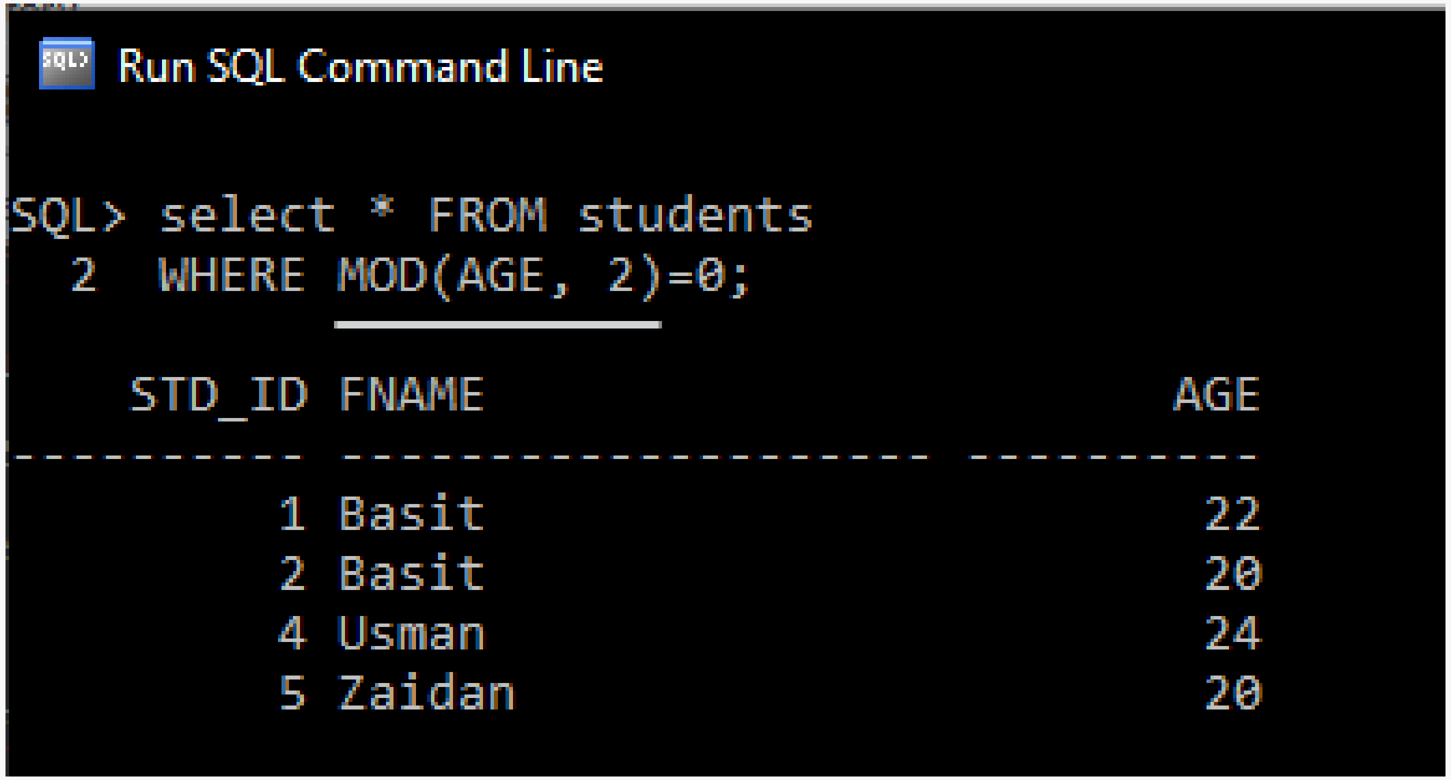
- **ROUND:** Rounds value to a specified decimal
- **TRUNC:** Truncates value to a specified decimal
- **MOD:** Returns remainder of division

Function	Result
ROUND(1.268, 2)	1.27
TRUNC(1.268, 2)	1.26
MOD(5, 2)	1

Syntax:

```
SELECT    column1, Function(column2) , ...
FROM       table_name
Where     Function(column)= value;
```

Number Functions-MOD



Run SQL Command Line

```
SQL> select * FROM students
  2  WHERE MOD(AGE, 2)=0;

    STD_ID FNAME                                 AGE
---------- -------------------- ----------
         1 Basit                                  22
         2 Basit                                  20
         4 Usman                                  24
         5 Zaidan                                 20
```

Date-Manipulation Functions

Function	Result	Use of function	Result
MONTHS_BETWEEN	Number of months between two dates	MONTHS_BETWEEN ('01-SEP-95‘ , '11-JAN-94‘)	19.6774194
ADD_MONTHS	Add calendar months to date	ADD_MONTHS ('31-JAN-96',1)	'29-FEB-96'
NEXT_DAY	Next day of the date specified	NEXT_DAY ('01-SEP-95','FRIDAY')	'08-SEP-95'
LAST_DAY	Last day of the month	LAST_DAY ('01-FEB-95')	'28-FEB-95'
ROUND	Round date	ROUND ('25-MAY-93 ‘ , 'YEAR')	'01-JAN-94'
TRUNC	Truncate date	TRUNC ('25-MAY-93 ‘ , 'YEAR')	'01-JAN-93'

Multiple-row Functions

Multiple-row Functions

- Multiple row functions are those function that give one result per row.
- Multiple row functions are **Group** functions.
- **Group functions** operate on sets of rows to give one result per group.

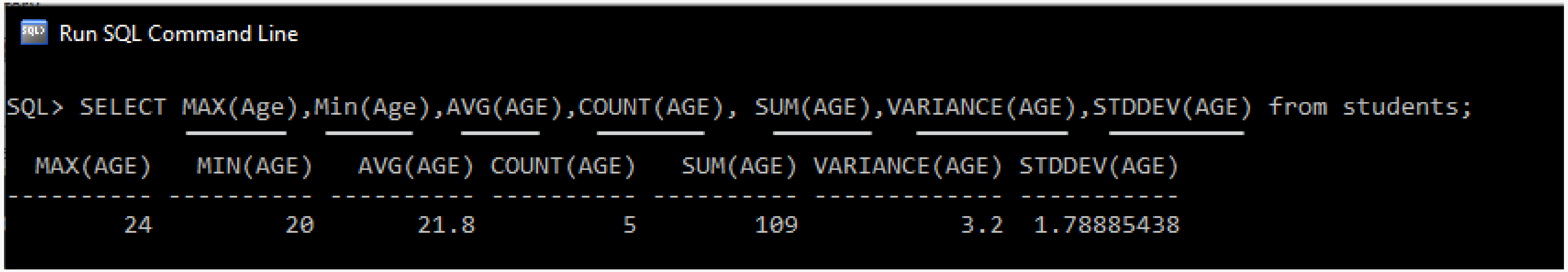
GROUP Functions

- Types of Group COUNT
    - MAX
    - MIN
    - STDDEV
    - SUM
    - VARIANCE

Syntax:

SELECT group_function (column), ...
FROM table
[WHERE condition]
[ORDER BY column];

GROUP Functions



```
SQL> SELECT MAX(Age),Min(Age),AVG(AGE),COUNT(AGE), SUM(AGE),VARIANCE(AGE),STDDEV(AGE) from students;

  MAX(AGE)   MIN(AGE)   AVG(AGE) COUNT(AGE)   SUM(AGE) VARIANCE(AGE) STDDEV(AGE)
---------- ---------- ---------- ---------- ---------- ------------- -----------
        24         20       21.8          5        109           3.2  1.78885438
```

Grouping rows

- Grouping rows:
  - GROUP BY clause
  - HAVING clause

GROUP BY

- Creating Groups of Data using **GROUP BY** Clause.
- You can divide rows in a table into smaller groups by using the GROUP BY clause.
- Syntax:

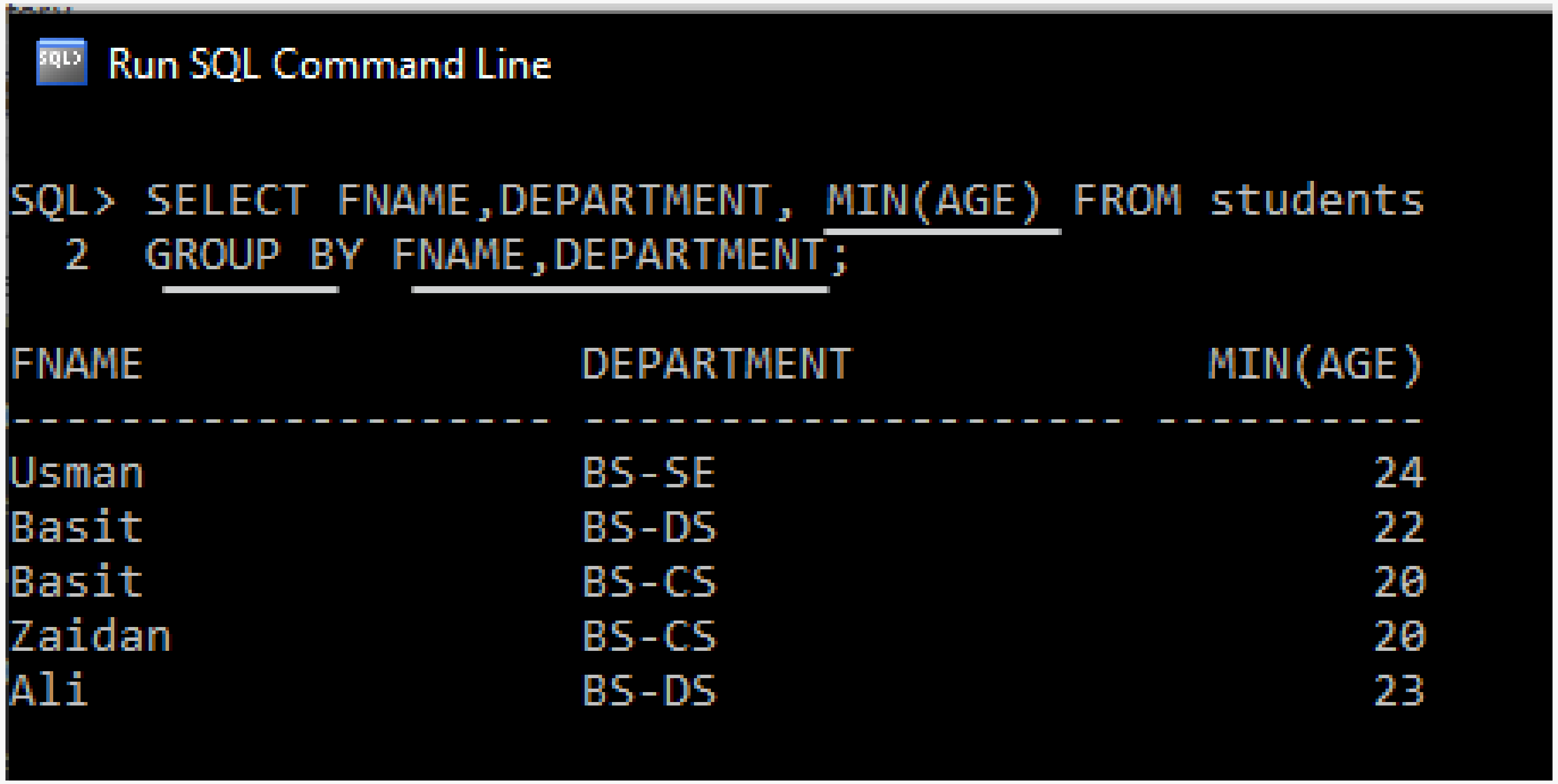
```
SELECT column, group_function(column)
FROM table
GROUP BY column;
```

Grouping by More than One Column

- Using the GROUP BY Clause on Multiple Columns
- Syntax:

```
SELECT column1, column2, group_function(column)
FROM table
GROUP BY column1, column2;
```

Grouping by More than One Column



```
SQL> SELECT FNAME,DEPARTMENT, MIN(AGE) FROM students
  2  GROUP BY FNAME,DEPARTMENT;

FNAME                DEPARTMENT             MIN(AGE)
-------------------- -------------------- ----------
Usman                BS-SE                        24
Basit                BS-DS                        22
Basit                BS-CS                        20
Zaidan               BS-CS                        20
Ali                  BS-DS                        23
```

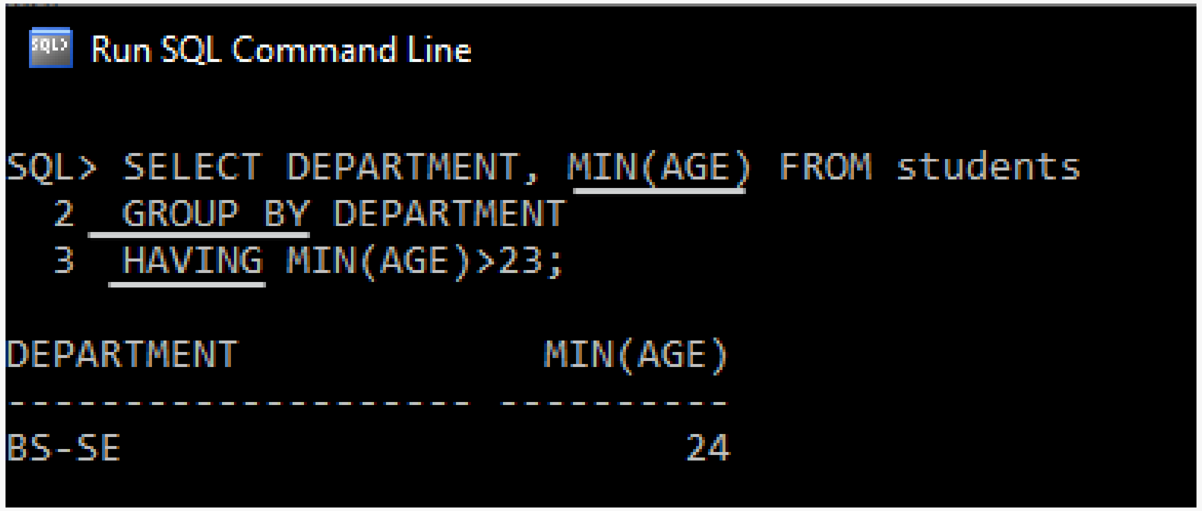
Having clause

- When you use the HAVING clause, the Oracle server restricts groups as follows:
  1. Rows are grouped.
  2. The group function is applied.
  3. Groups matching the HAVING clause are displayed.

Syntax:

```
SELECT column, group_function(column)
FROM table
GROUP BY column
Having  group_condition;
```

Having clause


Run SQL Command Line

```
SQL> SELECT DEPARTMENT, MIN(AGE) FROM students
  2  GROUP BY DEPARTMENT
  3  HAVING MIN(AGE)>23;

DEPARTMENT             MIN(AGE)
-------------------- ----------
BS-SE                        24
```

Displaying Data from Multiple Tables

Types of Joins

- Natural joins:
  - NATURAL JOIN clause
  - USING clause
  - ON clause
- Outer joins:
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN
- Cross joins

Difference between Natural Joins, Using, and ON Join

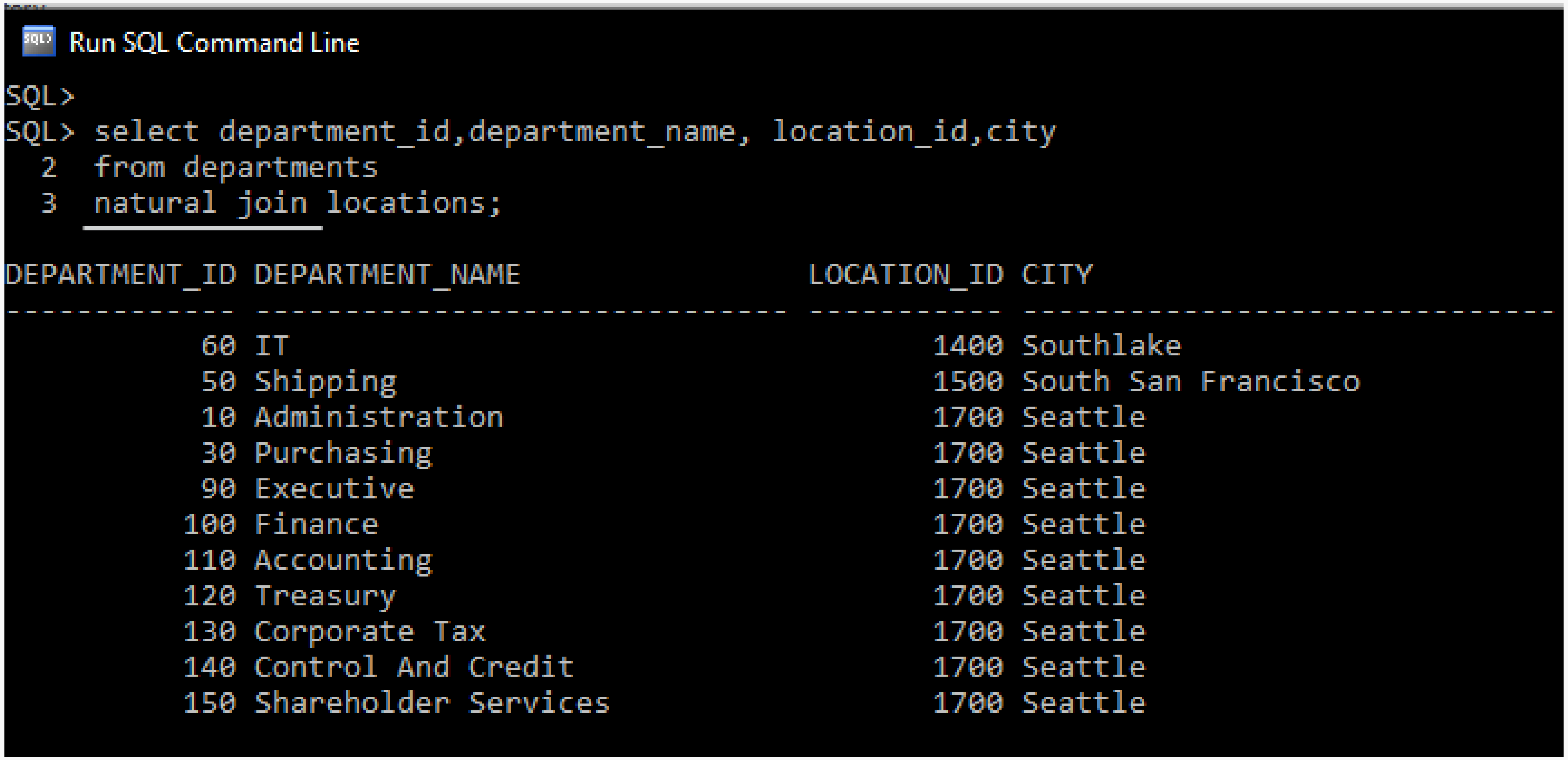
Natural Joins	Using Clause	ON Clause
It is used to join two tables on the basis of identical columns that have same names and same data types	It is used to join two tables on the basis of multiple columns with same name	It is used to join two tables on the basis of columns that have different names.

Natural Joins

- The NATURAL JOIN clause is based on all columns in the two tables that have the same name.
- It selects rows from the two tables that have identical column names and data types.
- If the columns having the same names have different data types, an error is returned.
- Syntax:

```
SELECT       Table1.column, Table2.column
FROM         Table1
NATURAL JOIN Table2 ;
```

Natural Joins


Run SQL Command Line

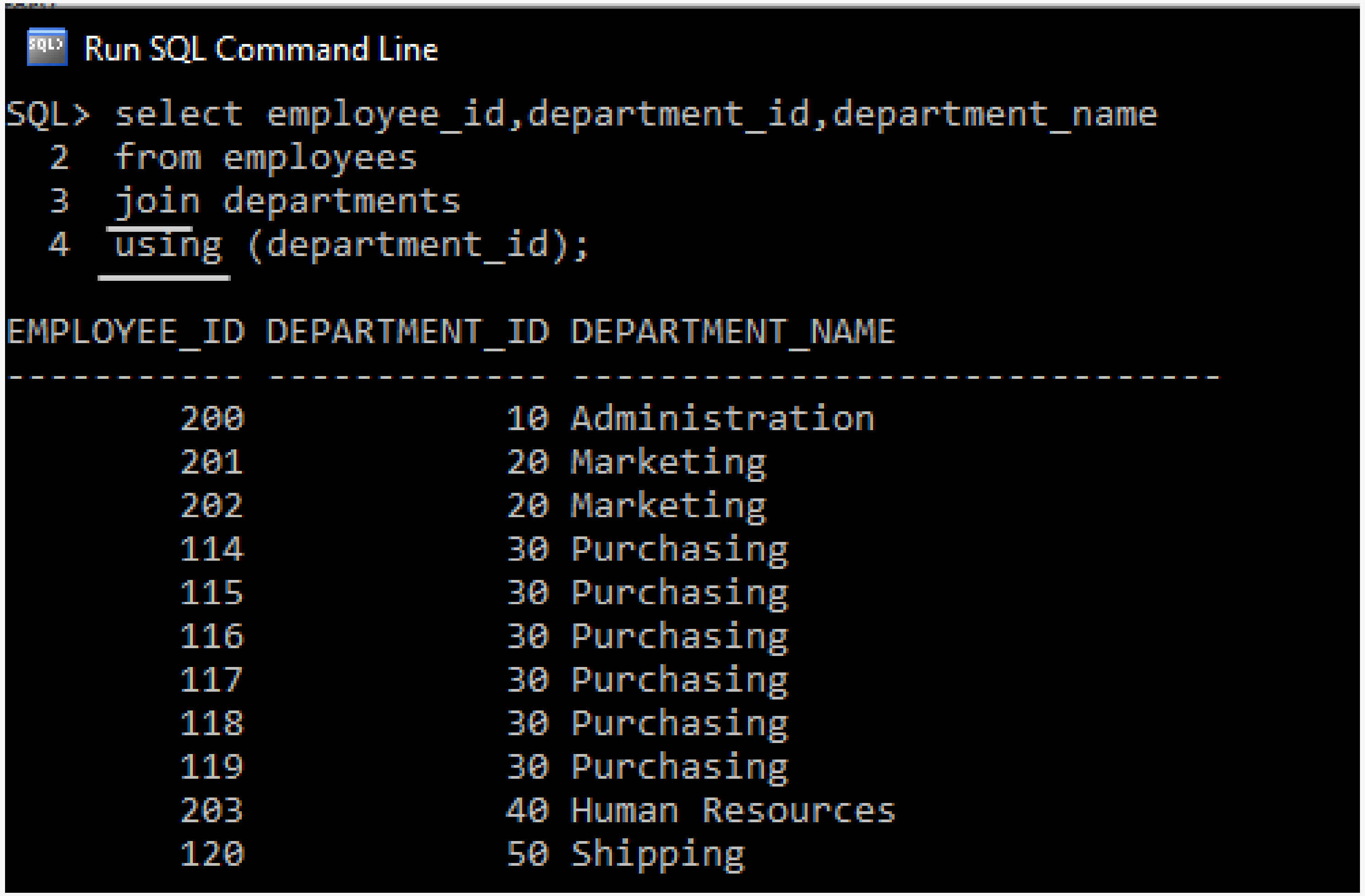
```
SQL>
SQL> select department_id,department_name, location_id,city
  2  from departments
  3  natural join locations;

DEPARTMENT_ID DEPARTMENT_NAME                LOCATION_ID CITY
------------- ------------------------------ ----------- ------------------------------
           60 IT                                    1400 Southlake
           50 Shipping                              1500 South San Francisco
           10 Administration                        1700 Seattle
           30 Purchasing                            1700 Seattle
           90 Executive                             1700 Seattle
          100 Finance                               1700 Seattle
          110 Accounting                            1700 Seattle
          120 Treasury                              1700 Seattle
          130 Corporate Tax                         1700 Seattle
          140 Control And Credit                    1700 Seattle
          150 Shareholder Services                  1700 Seattle
```

Creating Joins with the USING Clause

- If several columns have the same names but the data types do not match, natural join can be applied using the **USING** clause to specify the columns that should be used for an equijoin.
- Use the **USING** clause to match only one column when more than one column matches.
- The NATURAL JOIN and USING clauses are mutually exclusive.
- Syntax:

```
SELECT      Table1.column, Table2.column
FROM        Table1
JOIN Table2
USING  join_column;
```

Joins - USING Clause


Run SQL Command Line

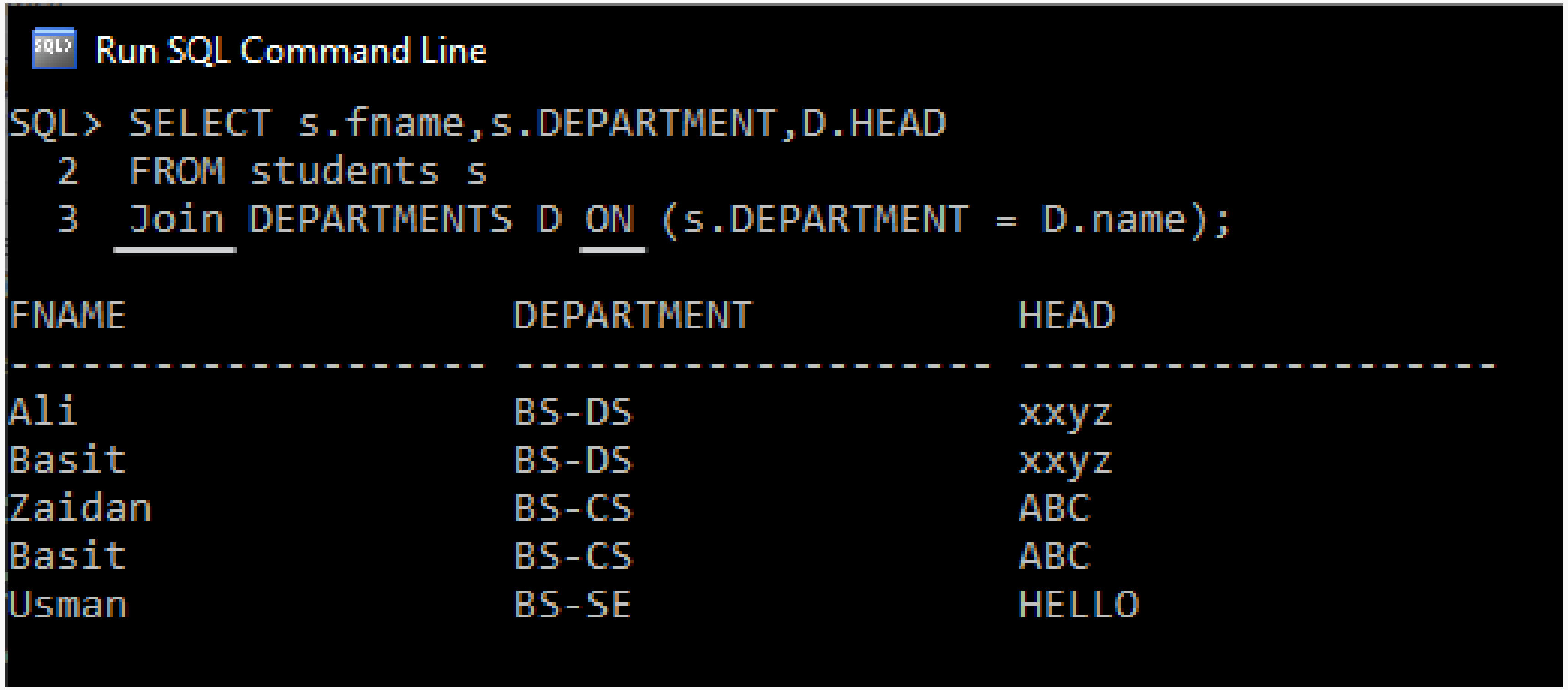
```
SQL> select employee_id,department_id,department_name
  2  from employees
  3  join departments
  4  using (department_id);

EMPLOYEE_ID DEPARTMENT_ID DEPARTMENT_NAME
----------- ------------- ------------------------------
        200            10 Administration
        201            20 Marketing
        202            20 Marketing
        114            30 Purchasing
        115            30 Purchasing
        116            30 Purchasing
        117            30 Purchasing
        118            30 Purchasing
        119            30 Purchasing
        203            40 Human Resources
        120            50 Shipping
```

Creating Joins with the ON Clause

- The join condition for the natural join is basically an equijoin of all columns with the same name.
- Use the ON clause to specify arbitrary conditions or specify columns to join.
- The join condition is separated from other search conditions.
- The ON clause makes code easy to understand.
- Syntax:

```
SELECT      Table1.column, Table2.column
FROM        Table1
JOIN Table2
ON  ( Table1.column_name = Table2.column_name ) ;
```

Joins with the ON Clause



```
SQL> SELECT s.fname,s.DEPARTMENT,D.HEAD
  2  FROM students s
  3  Join DEPARTMENTS D ON (s.DEPARTMENT = D.name);

FNAME                    DEPARTMENT               HEAD
-------------------- -------------------- --------------------
Ali                  BS-DS                xxyz
Basit                BS-DS                xxyz
Zaidan               BS-CS                ABC
Basit                BS-CS                ABC
Usman                BS-SE                HELLO
```

Applying Additional Conditions to a Join

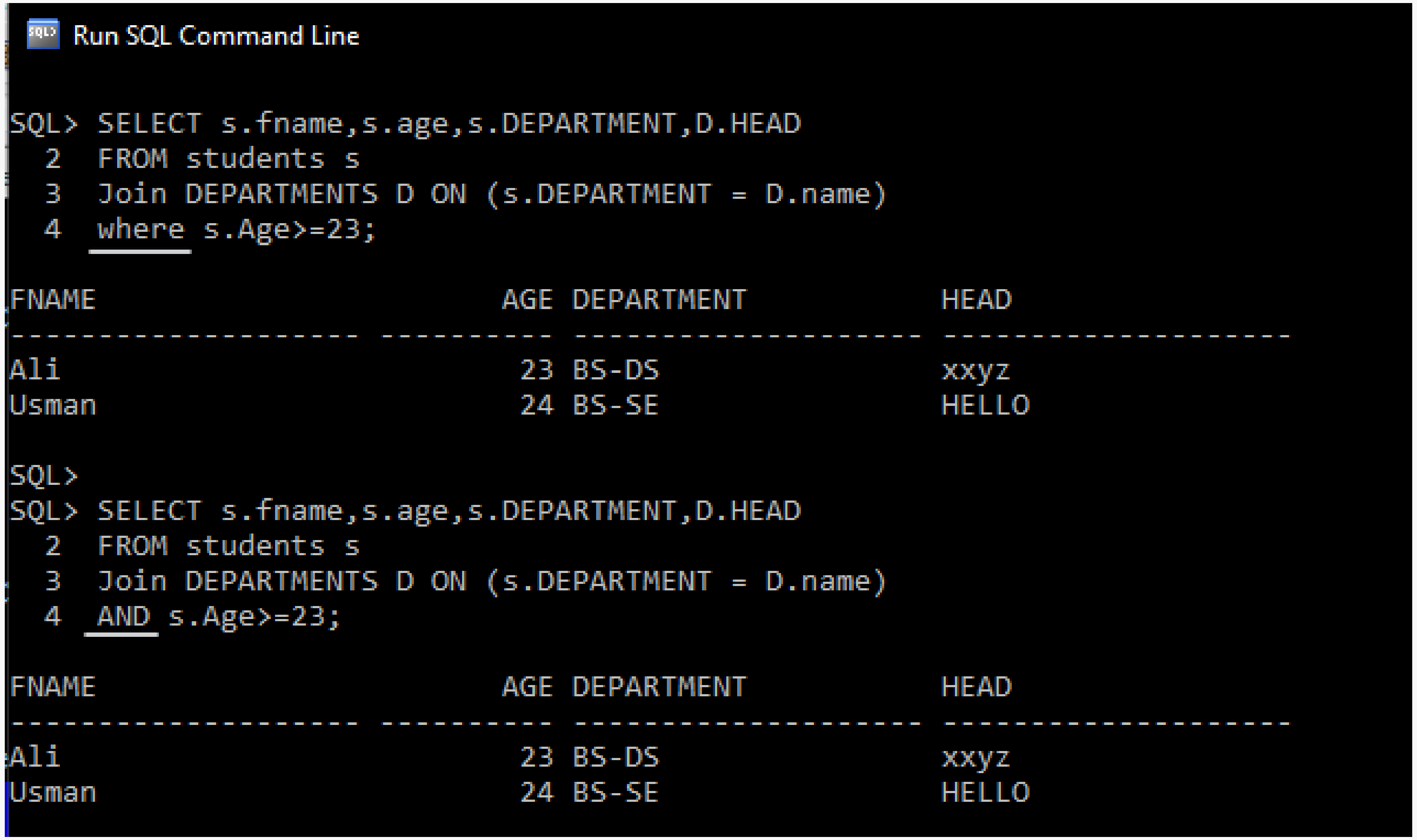
- Use the AND clause or the WHERE clause to apply additional conditions
- Syntax:

```
SELECT      Table1.column, Table2.column
FROM        Table1
JOIN Table2
ON  ( Table1.column_name = Table2.column_name )
AND [condition] ;
```

- Syntax:

```
SELECT      Table1.column, Table2.column
FROM        Table1
JOIN Table2
ON  ( Table1.column_name = Table2.column_name )
WHERE [condition] ;
```

Conditions to a Join



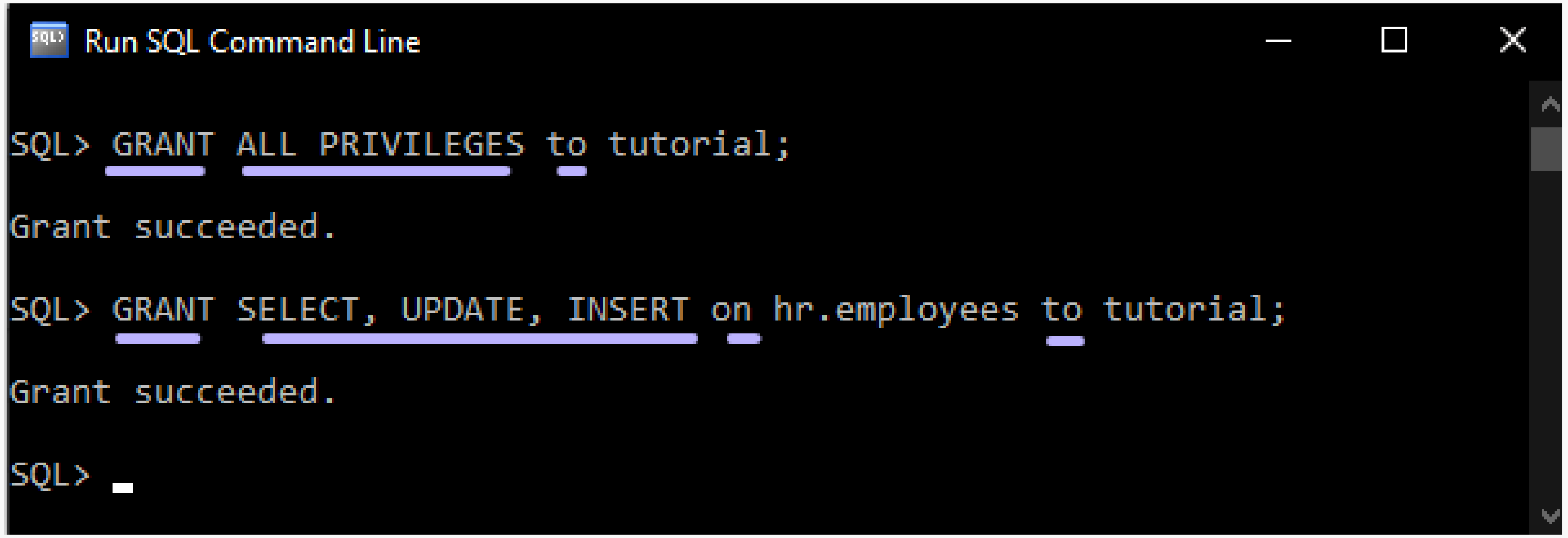
Run SQL Command Line

```
SQL> SELECT s.fname,s.age,s.DEPARTMENT,D.HEAD
  2  FROM students s
  3  Join DEPARTMENTS D ON (s.DEPARTMENT = D.name)
  4  where s.Age>=23;

FNAME                           AGE DEPARTMENT           HEAD
-------------------- ---------- -------------------- --------------------
Ali                              23 BS-DS                xxyz
Usman                            24 BS-SE                HELLO

SQL>
SQL> SELECT s.fname,s.age,s.DEPARTMENT,D.HEAD
  2  FROM students s
  3  Join DEPARTMENTS D ON (s.DEPARTMENT = D.name)
  4  AND s.Age>=23;

FNAME                           AGE DEPARTMENT           HEAD
-------------------- ---------- -------------------- --------------------
Ali                              23 BS-DS                xxyz
Usman                            24 BS-SE                HELLO
```

SQL-Data Control Language (DCL)

GRANT

- Syntax:
  - **GRANT** type_of_permission **TO** username;
  - **GRANT** type_of_permission **ON** database_name.table_name **TO** username;



Run SQL Command Line

```
SQL> GRANT ALL PRIVILEGES to tutorial;

Grant succeeded.

SQL> GRANT SELECT, UPDATE, INSERT on hr.employees to tutorial;

Grant succeeded.

SQL>
```

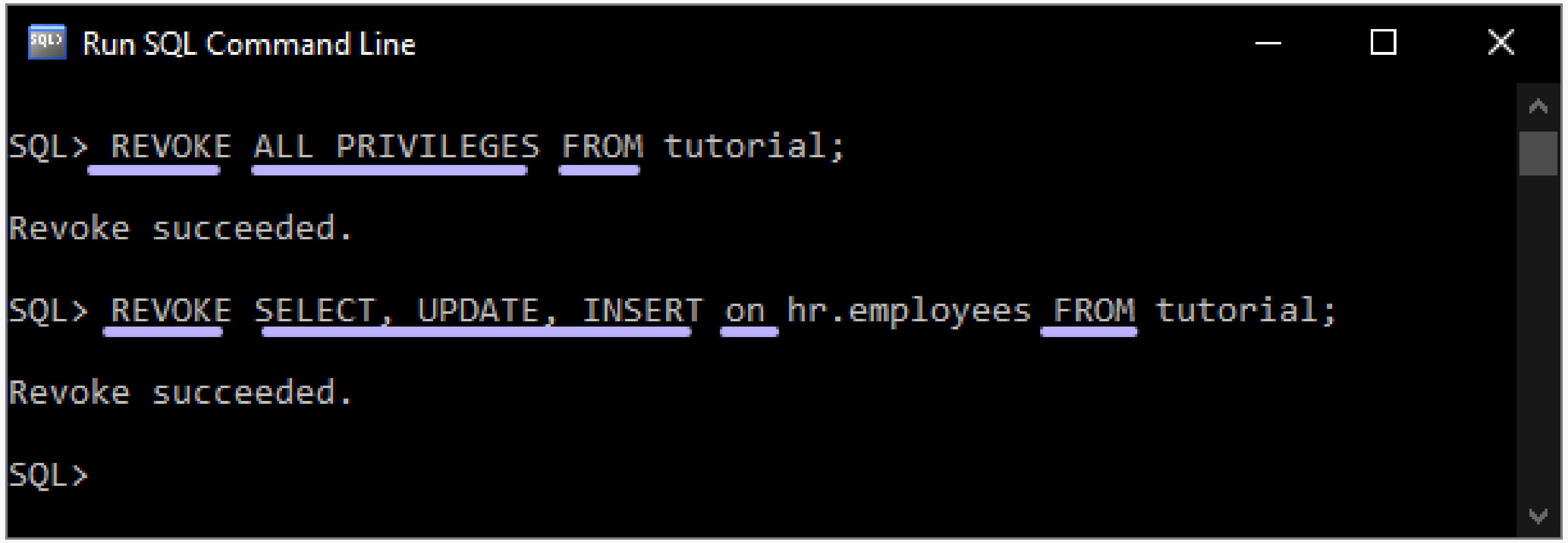
How To Grant Different User Permissions

Here is a short list of other common possible permissions that users can enjoy.

Permissions	Description
ALL PRIVILEGES	This would allow a user full access to a designated database (or if no database is selected, global access across the system)
CREATE	allows them to create new tables or databases
DROP	allows them to them to delete tables or databases
DELETE	allows them to delete rows from tables
INSERT	allows them to insert rows into tables
SELECT	allows them to use the SELECT command to read through databases
UPDATE	allow them to update table rows
GRANT OPTION-	allows them to grant or remove other users’ privileges

REVOKE

- Syntax:
  - **REVOKE** type_of_permission **FROM** username**;**
  - **REVOKE** type_of_permission **ON** database_name.table_name **FROM** username**;**



Run SQL Command Line

```
SQL> REVOKE ALL PRIVILEGES FROM tutorial;

Revoke succeeded.

SQL> REVOKE SELECT, UPDATE, INSERT on hr.employees FROM tutorial;

Revoke succeeded.

SQL>
```

THE END