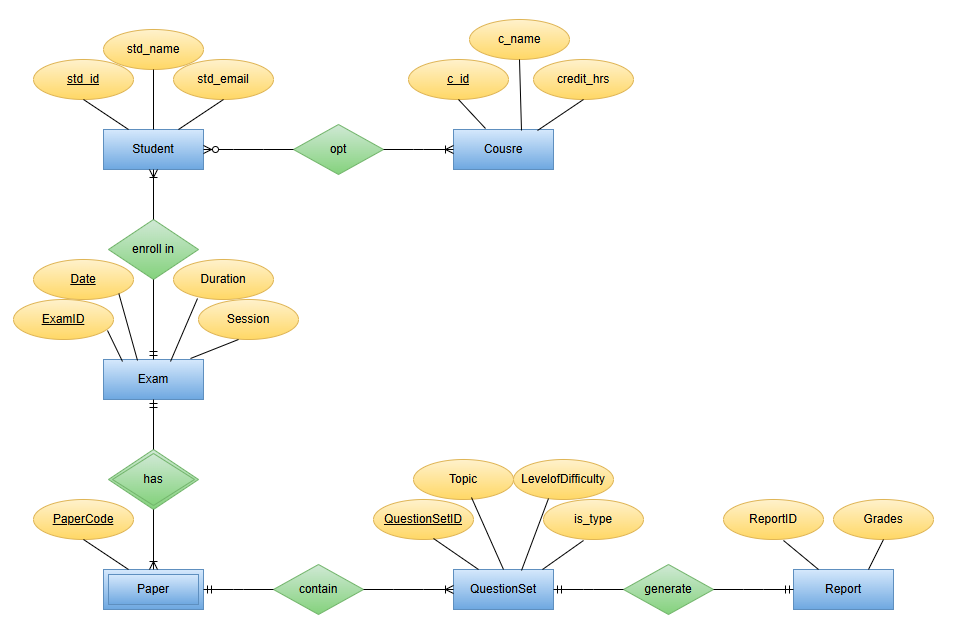
**Entity-Relationship Diagram:**



**Tables (In Normalized Form):**

**student** (std\_id, course\_id, std\_name, std\_email)

**course** (course\_id, c\_name, credit\_hrs)

**student\_course** (std\_id, course\_id)

**exam** (exam\_id, course\_id, date, duration, session)

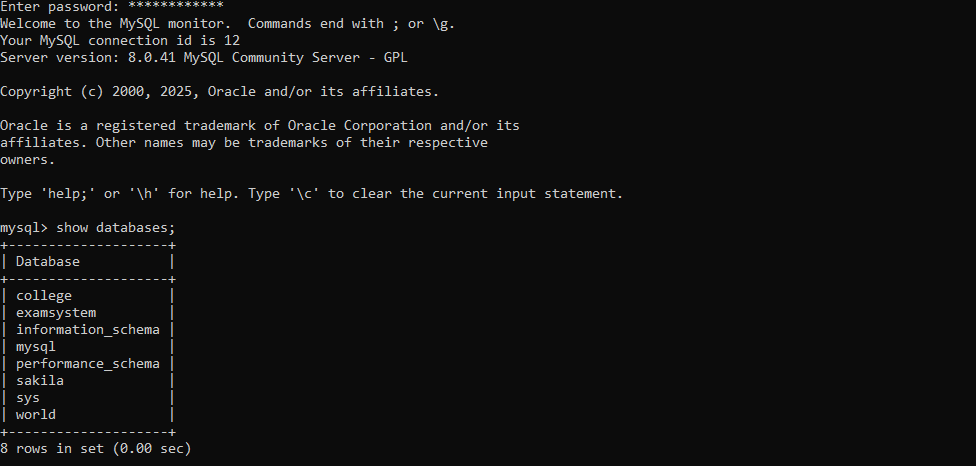
**paper** (papercode, exam\_id)

**questionset** (questionset\_id, papercode, exam\_id, Topic, LevelofDifficulty, is\_type)

**report** (report\_id, std\_id, questionset\_id, course\_id, grades)

**QUERIES:**

1. To view all databases:

Entered password and used **SHOW DATABASES;** -this **MySQL** statement displays all databases the user has access to.

1. To create new databases:

Here I am adding **create database online\_exam\_system;** -to create a new database

1. To use a databases:

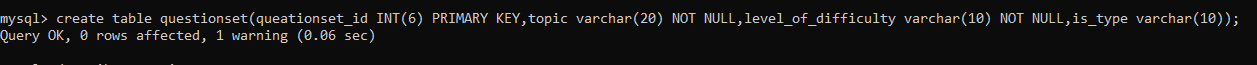
Now using **use online\_exam\_system** to select a specific database to work with.

1. To create table:

This statement is to create table in **MySQL** named **student** while adding attributes

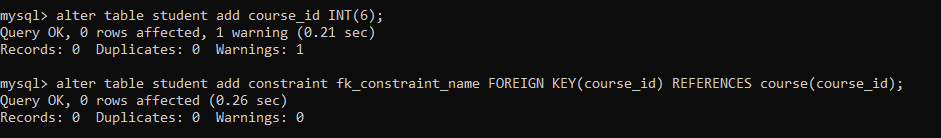
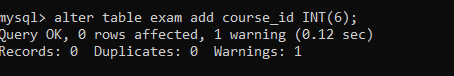
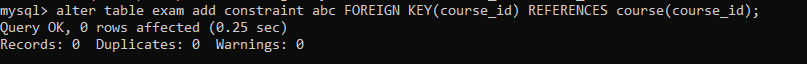
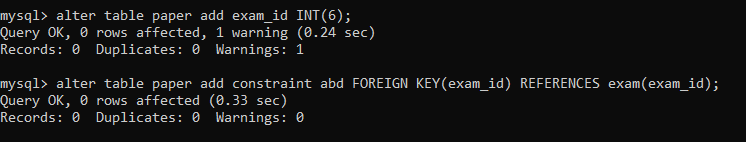
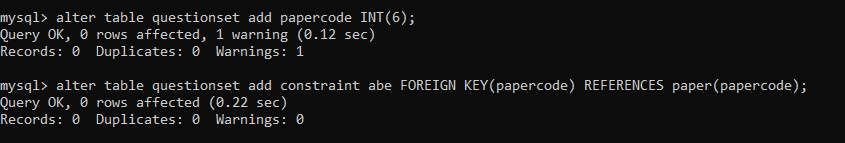
Student table:Course table:

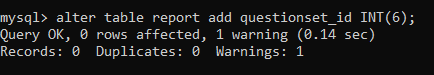


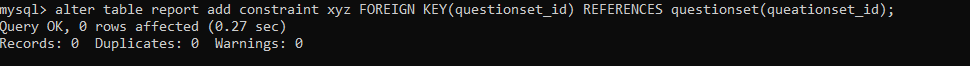
Exam table: Paper table: Questionset table: Report table:

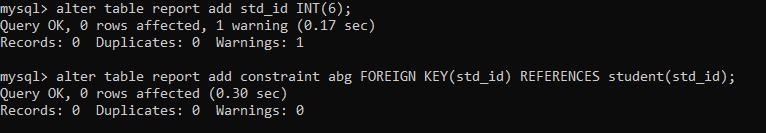
1. To add foreign key in a table:

Using alter table statement to **add** course id.

Here first making primary key of course table as foreign key in student.Here **alter**ing **table** exam to add new attribute.  Here **alter**ing **table** paper to add new attribute **exam\_id** with integer datatype. Here **alter**ing **table** **questionset** to add new attribute **papercode**. Here **alter**ing **table** report to add new attribute **question\_set.**



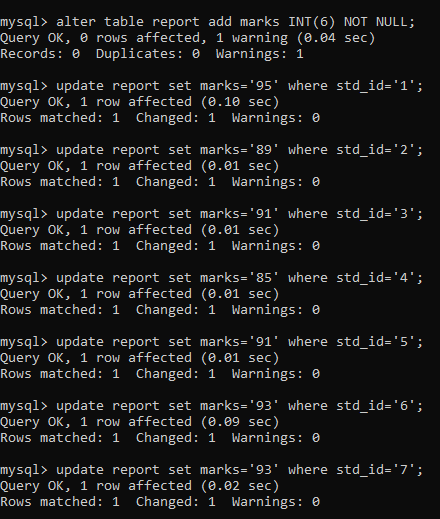
 Here **alter**ing **table** report to add new attribute named std\_id.

Here first making primary key of table student as foreign key in table report 

1. To add new attribute in an existing table:

Here **alter**ing **table** report to add new attribute named std\_id.

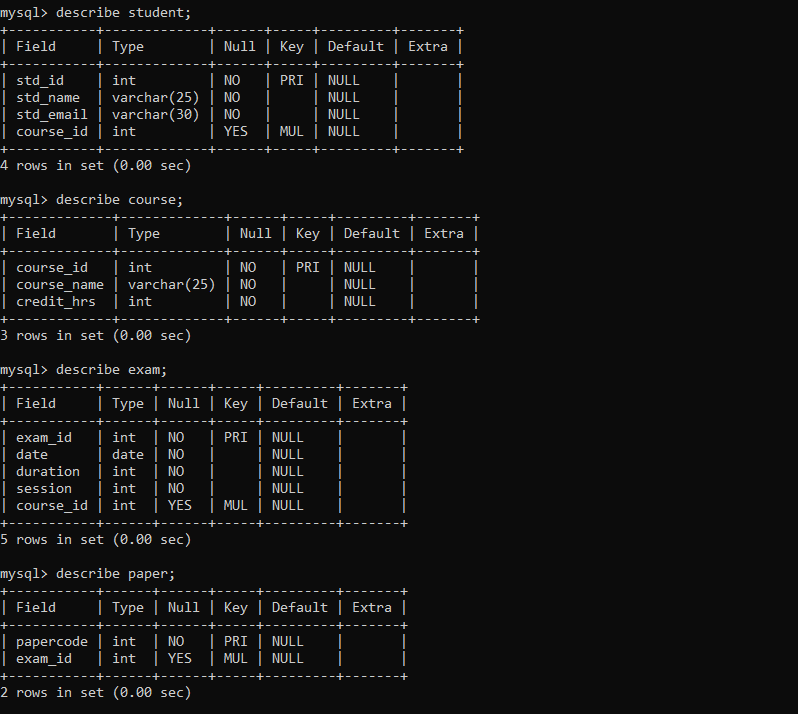
**Update** table REPORT to set marks. Update query is used to modify existing records in a table. WHERE clause is used to filter rows in SQL statement.

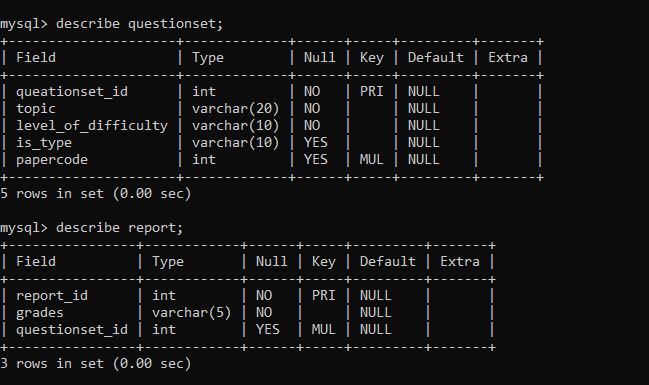


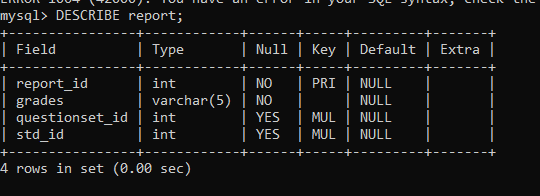
1. To view structure of a table:

**DESCRIBE** (or **DESC**) command is used to ***show the structure of table***.

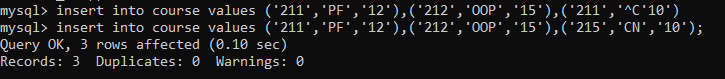
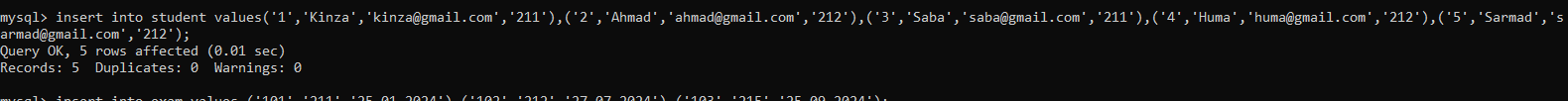
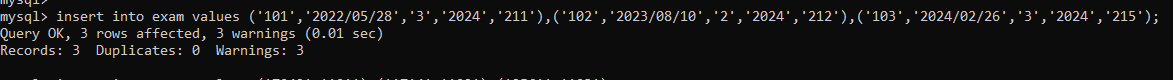
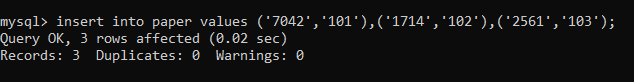
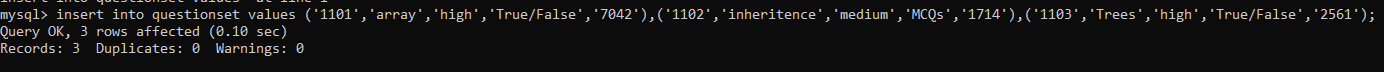
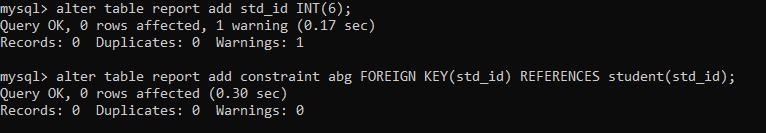
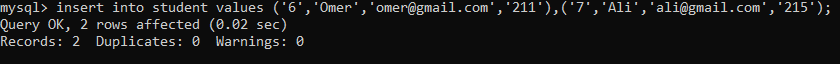
Writing describe student show all the entered data of the specific table.this command is showing the structure that includes its field, types, null, key , and extras

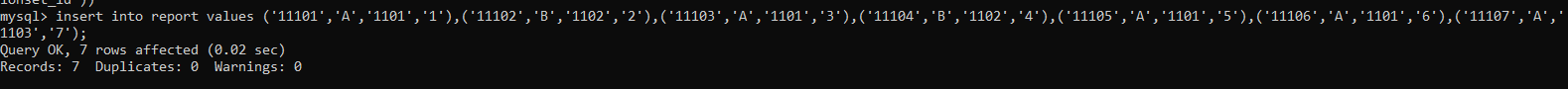
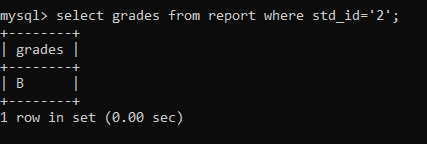
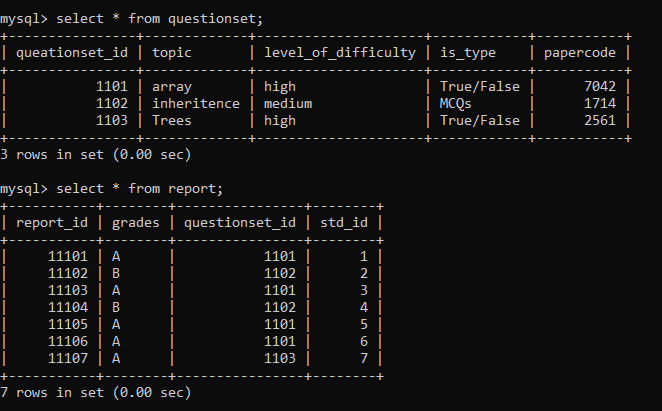


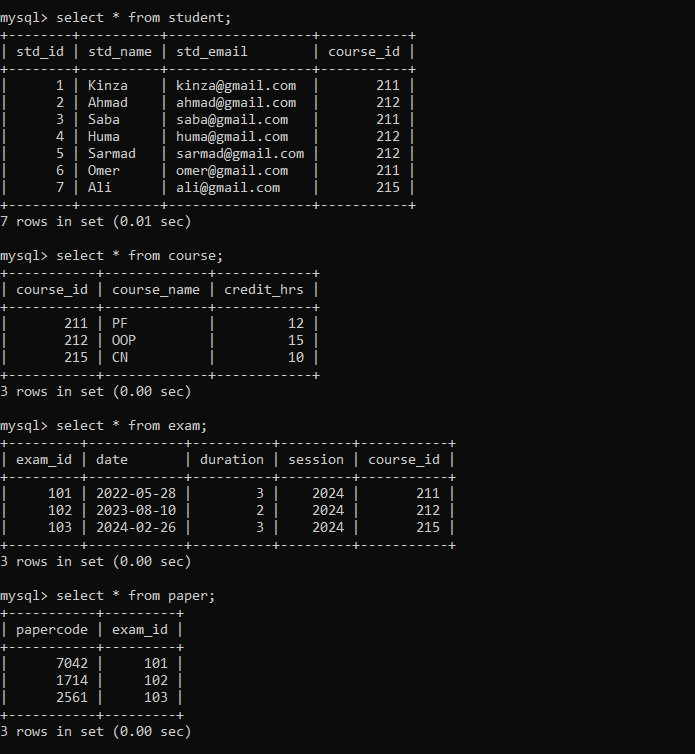




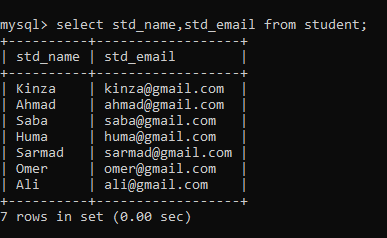
1. To insert values in a table:

To **insert** values into table course.to enter values in table the structure of table should must be prioritize otherwise due to datatype’s difference, errors occurred. To **insert** values into table student. To **insert** values into table exam. To **insert** values into table paper. To **insert** values into table questionset. To **insert** values into table report. To **insert** values into table student. To **insert** values into table report.

Here select statement is used to select a specific column named **grades** from report table WHERE clause is used to **filter rows in SQL** statement.so here using where we are referring to **2nd** value **“a** **specific** **data”** of column std\_id.**Select \*** is used to get a **view** of **all the entered data of a specific table** one by one here.

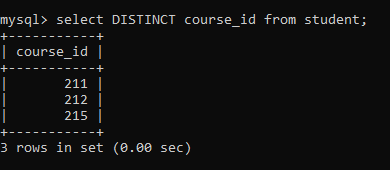


This statement is to view more than one attribute of a table:



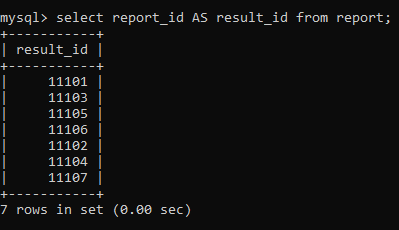
1. Use of DISTINCT clause:

Here using **DISTINCT** clause with select. the distinct keyword is used to remove duplictae values from the result

.

1. Use os ALIAS:

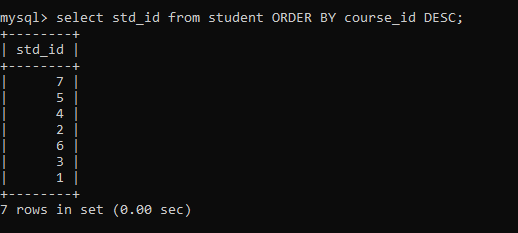
**AS** is used to **rename a coloumn** or **table** temorarily in the result set. this is called an **alias**.



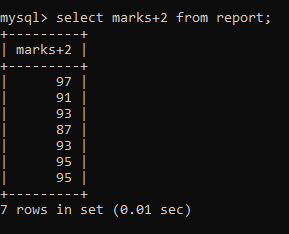
1. Use of ORDER BY clause:

Order by is used to sort the result set of a query by one or more columns.

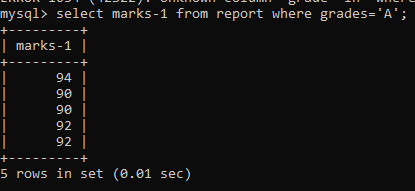
* ASC=Ascending order(default)
* DESC=Descending order



1. Arithmetic operators:

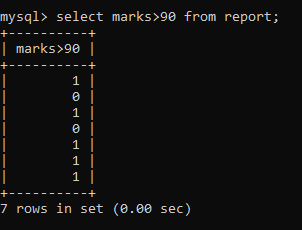
An Arithmetic operator are used in SQL to perform basic mathematical operations on numeric values. Here adding 2 in marks adds bonus marks to each student.

Using **subtraction Arithmetic operator** to subtract 1 from report when the grade of student is specifically “A”

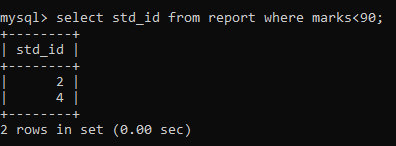
.

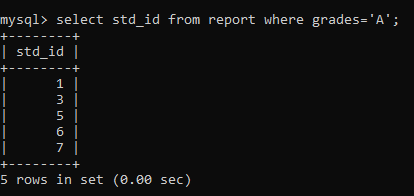
1. Relational operators:

Are used to **compare** two values. They return **either true or false** and are commonly used in **where** clauses. In this query marks of student that are more or greater than 90 are displayed from report table.

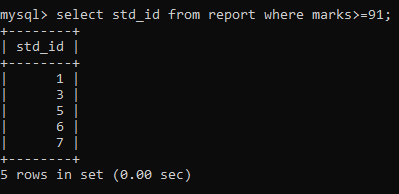
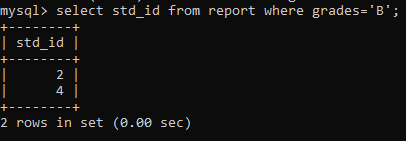
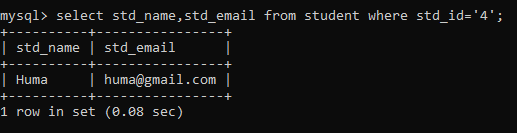


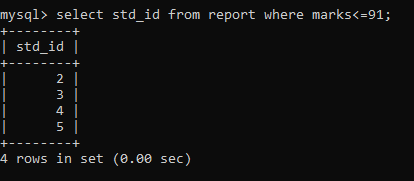
Selecting std\_id column with some conditions as we want to display marks that are less than 90 from report table in std\_id column

.



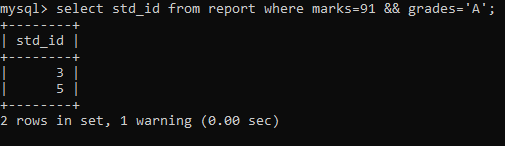
Selecting specific columns: name and email from student where std\_id=4.

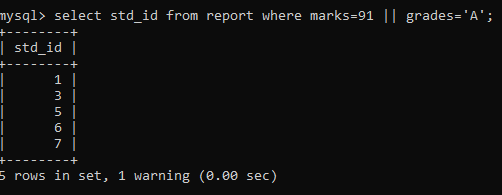


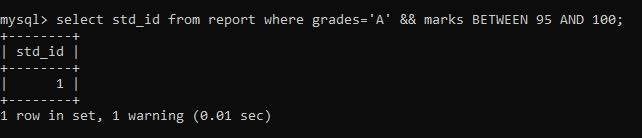


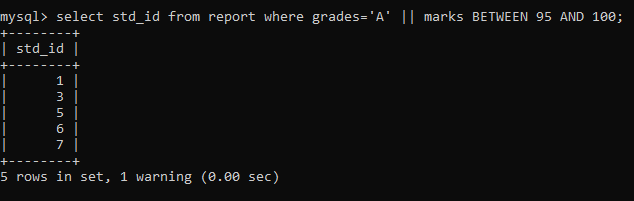
1. Logical operators**:**

These operators are used to **combine multiple conditions** in a **WHERE** clause. They return true or false based on the logic. here in this query we want to display std\_id from report table where marks of students are equal to 91 **and** grade =’A’.

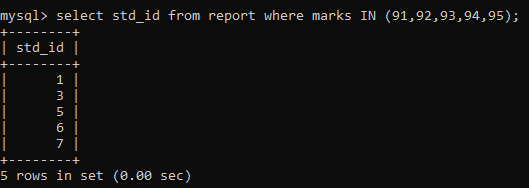


Here in this query we want to display std\_id from report table where marks of students are equal to 91 **OR** grade =’A’.

The **between… AND** operator is used to filter values within a specific range, including the start and end values. As we want to display values between **95 and 100** so we used **BETWEEN** & **AND** operator.

To display std\_id from report where grade=A while using **logical OR** using between & and to get result according to required result. 

1. Use of IN:

The **IN** operator is used to match a column’s value against a list of values. It helps simplify **multiple OR condition**. 

1. Use of LIKE:

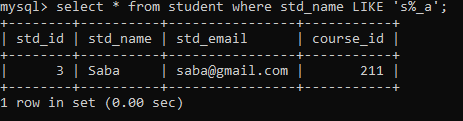
The **LIKE** operator is used in a where clause to search for a pattern in a column, especially in **VARCHAR** or **TEXT** format. Here the pattern can be broken down as:

**S->**The value must start from the letter s.

**%->** Followed by zero or more characters.

**\_->** Followed by exactly one characters.

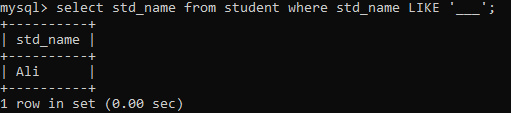
**A->** Ending with the letter ‘a’



**\_\_\_** (three underscores) means:

Exactly three characters, any characters in each position.

Each\_ represents one character only.



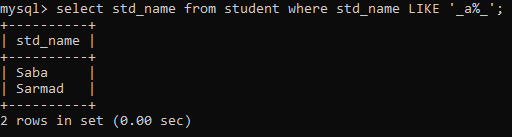
\_a%\_ means:

\_-> any single character.

a-> the second character must be ‘a’.

%-> Any number of characters (including 0) after that.

\_->the value must have at least one more character at the end.

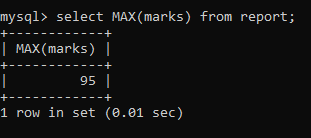


1. Aggregate functions:

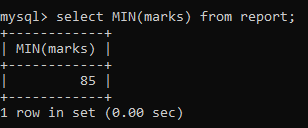
**Aggregate function**: perform calculations on multiple rows of a column and return a single value. They are often used with **GROUP BY**.

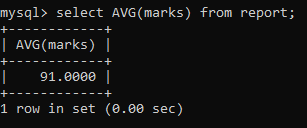
This statement is to **find** the **highest marks** from the **report** table.

**MAX (marks)** finds the **maximum** value in the **marks column**.



This statement is to **find** the **lowest marks** from the **report** table.

**MIN (marks)** find the **minimum** value in the **marks column**.

To calculate the **average marks** from a table named report.

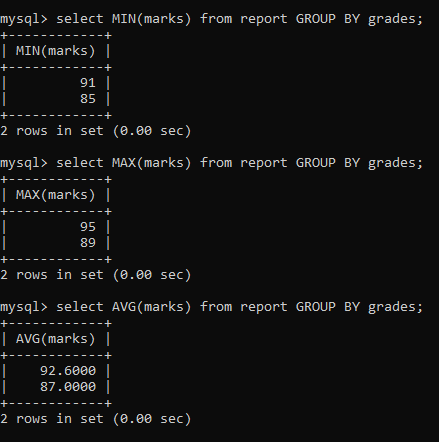
1. GROUP BY CLAUSE:

is used to group rows that have the same values in specified column, often used with aggregate functions like **SUM(), AVG(), COUNT().**

To find the minimum marks for each grade in the report table:

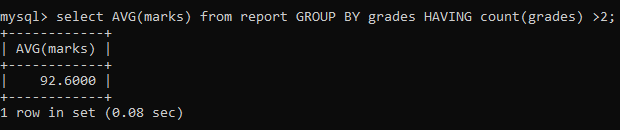
MIN (marks) finds the lowest mark in each group.

**GROUP BY** grade **groups the records** by the grade column.



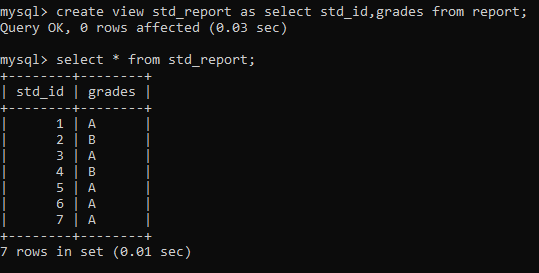
1. HAVING clause:

The **HAVING** clause is used with **GROUP BY** to **filter** **grouped** **results** after **aggregation** **(**unlike **WHERE**, which **filters rows after grouping).**



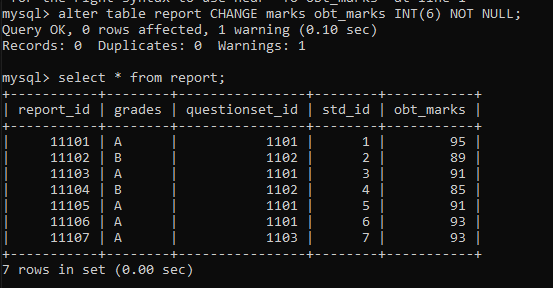
1. To create a view:

A **VIEW** is a **virtual** **table** based on the result of a **select query**. It simplifies **complex queries** and improves **readability** and **security**.



1. To rename a column of a table:

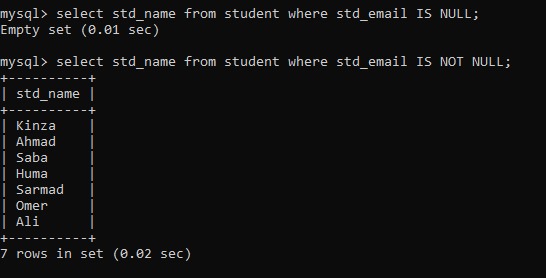
The **ALTER TABLE** statement is used to **rename** and **modify** a column name in the report table

.

1. IS NULL/ IS NOT NULL:

This statement is to check **whether an attribute is null or not null**.

SELECT **std\_name**->retrieves only the names from the student table WHERE **std\_email** is null implies that it filters rows where **std\_email** has no value.

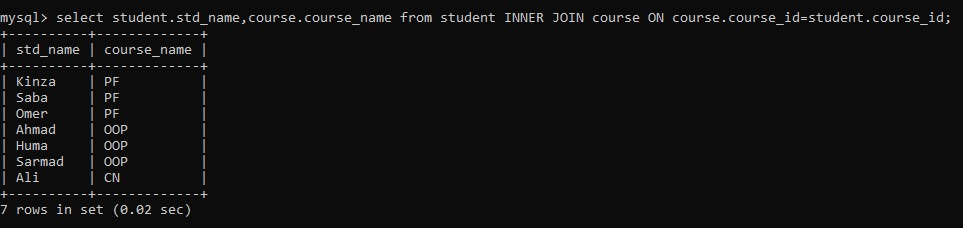


1. JOINS:

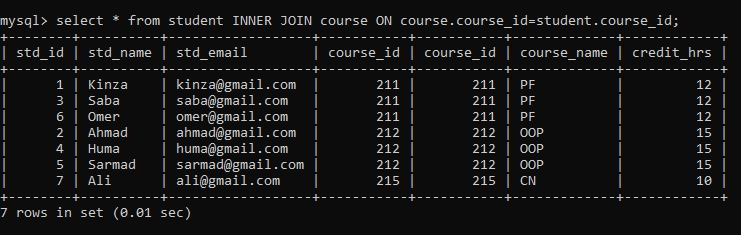
* An **INNER JOIN** returns **only the rows with matching values** in both tables.

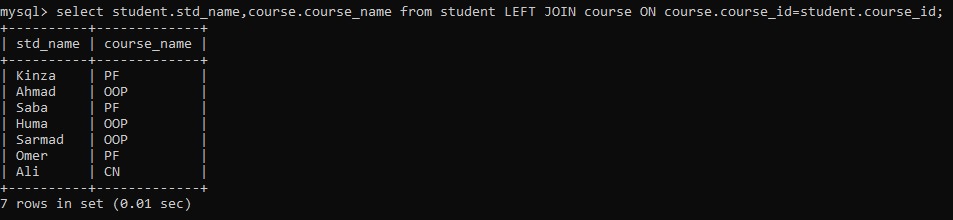
Select student.std\_name, course.course\_name->selects student names and their course names.

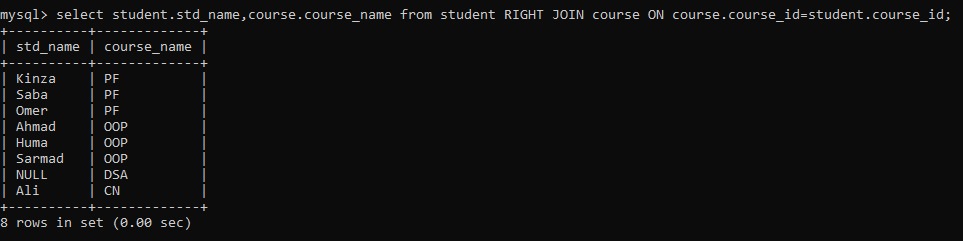
From student Inner Join course-> joins the student and course tables.

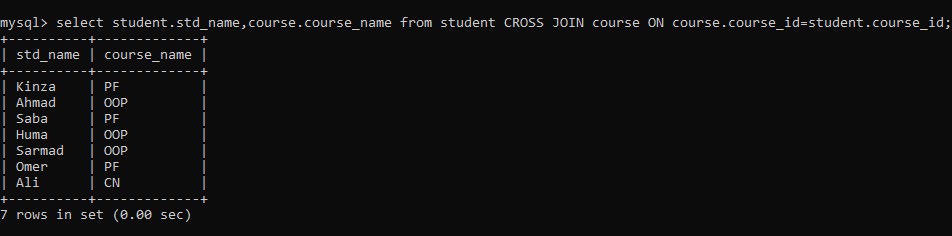
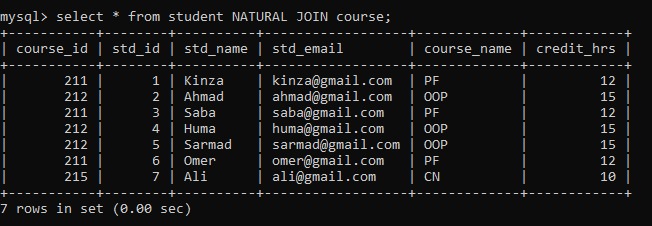
ON student.course\_id = course.course\_id-> matches records where the course IDs are the same.

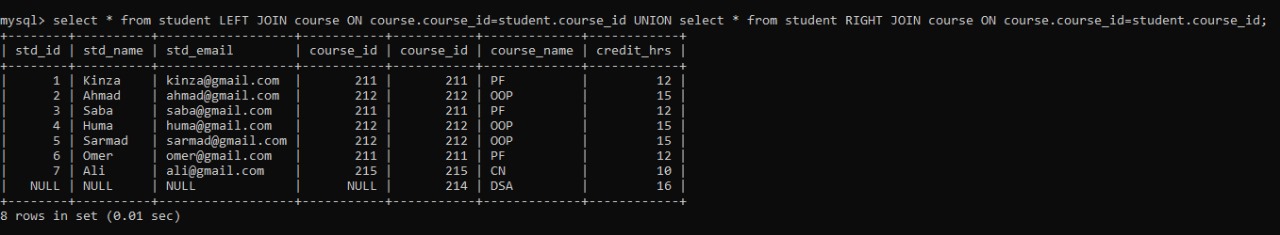
**Inner join** to select all columns as SELECT\* is used.



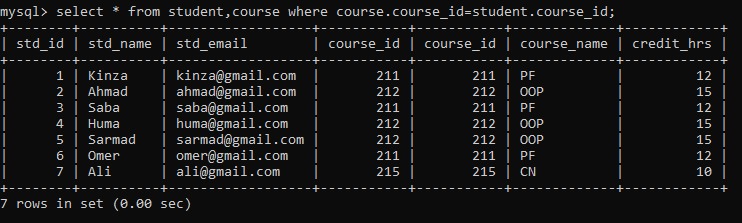
* An **LEFT JOIN** returns **all records from the left table, and the matching records** from the right table. **ON** clause is used with join operations to specify the conditions for joining two tables- usually using a common column.
* An **RIGHT JOIN** returns **all records from the right table, and the matching records** from the left table. . **ON** clause is used with join operations to specify the conditions for joining two tables- usually using a common column.



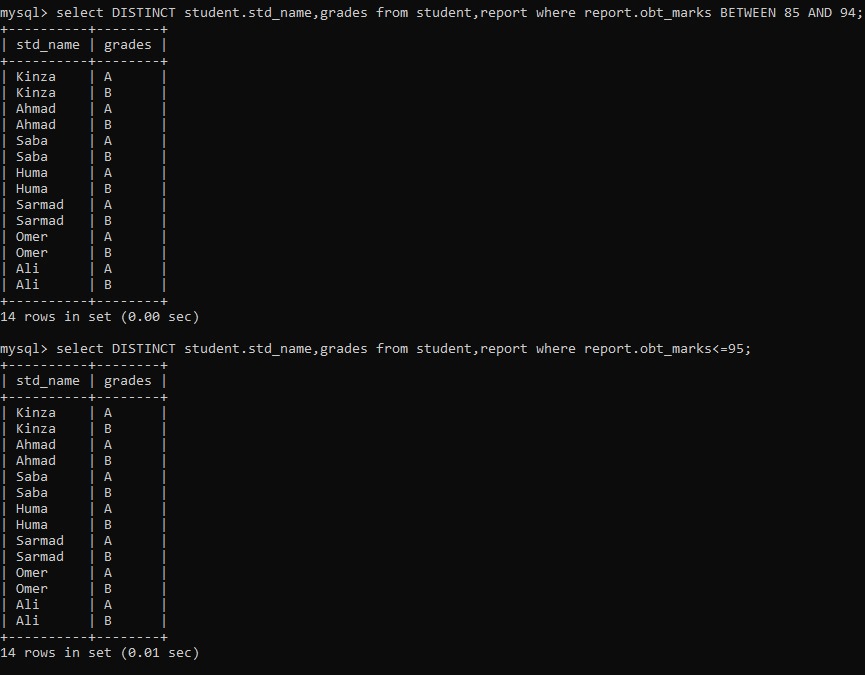
* A **CROSS JOIN** returns the **Cartesian product** of two tables – it pairs **every row** from the first table with **every row** from the second. . **ON** clause is used with join operations to specify the conditions for joining two tables- usually using a common column.
* A **NATURAL JOIN** automatically joins two tables based on columns with the same name and compatible data types- no need to specify the **ON** clause.
* **UNION** is used to **combine the result sets** of **two or more** select **queries** into a **single set result set, removing duplicate rows. FULL JOIN…**



* An **EQUI JOIN** is a type of JOIN that **compares common columns** of **two tables** using the **equality operator (=).**



**Distinct** is used in a **SELECT query** to **remove duplicate values** from the result set.



1. Create user in database:

This command **creates** a **new** **MySQL** user **named** ‘**tayyaba’** with the **password** ‘**12345’**.

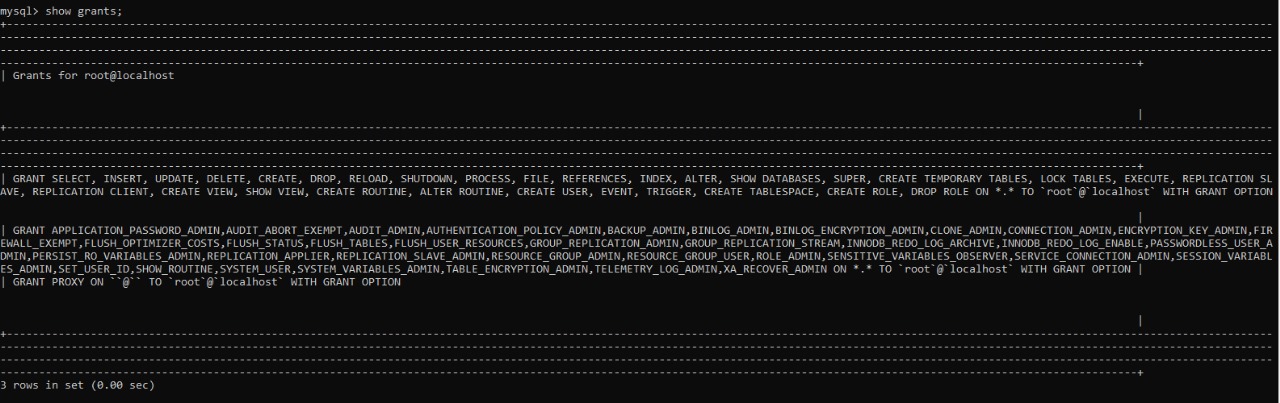


This statement is to **grant privileges** to **tayyaba**

.

To grant all **privileges** on the **online\_exam\_system** **database** to **user** **tayyaba**.

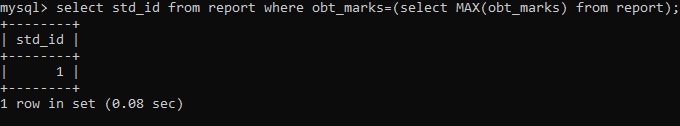
It displays all **privileges (grants)** of a specific **MySQL** user like **tayyaba**, use this command.



1. Subqueries:

The query retrieves the **std\_id** of the **student** who obtained the **highest marks** from the **report** **table.**

* Single row subquery:



* Multirow subquery:

