

NAME: Divyansh Kumar Singh

SAP ID: 590012175

BATCH: B1

DBMS LAB 2

**AIM: To Implement and Understand the Concept
Of Constraints**

Lab Objective: To understand the concept of data constraints that are enforced on data being stored in the table. Focus on Primary Key and the Foreign Key.

Name: Divyansh Kumar Singh

SAP ID: 590012175

Batch: B1

classmate

Date

19/8/25

Page

Lab - Q 8

Aim: To implement and understand the concept of constraints.

1. CREATE ⇒ For creating a database and tables namely "Products" and "Customers".

INT, VARCHAR(N), DECIMAL ⇒ Data Types.

NOT NULL ⇒ Constraint such that the field shouldn't be empty.

DESC ⇒ For displaying the structure of tables.

2. INSERT ⇒ For inserting values into table

VALUES ⇒ The rowwise values for the table is sequence to schema.

SELECT * FROM ⇒ For displaying the entire table.

3. ALTER ⇒ To modify Customers table.

ADD ⇒ To add a column Phone Number in table.

NOT NULL, DEFAULT ⇒ Constraints to make the field to not be empty and default number as "0000000000".

CHECK ⇒ Checking the length of phone number.

- Date _____
Page _____
4. Altering table by adding a column "unique_email" having constraint UNIQUE. Two fields with same email will show error.
5. ADD PRIMARY KEY (column) → To make the column as Primary Key.
6. Creating a new table "Orders" Table
FOREIGN KEY (customerID) References Customers
↓ ↓ ↓
To create a foreign key To which table & columns name it is referencing
7. BETWEEN 1 AND 100 → The value should be within the range (1 to 100)
8. DEFAULT → Making "Pending" as default value for order status if no value is given.
9. The index will speed up queries that filter or sort by Unit Price.
10. Joining Customers object c CustomerId and Orders object o. [JOIN]

classmate

Date _____

Page _____

11. Verify OrderQuantity > 50 for getting shipped.
using WHERE clause.
12. Delete fields from Products where unit price
is less than 10.
13. MODIFY the data type to FLOAT.
14. Avg(UnitPrice) will give average of values
in UnitPrice and AS will store it in
column AveragePrice. FROM table products.

Constraints in SQL

Constraints in SQL means we are applying certain conditions or restrictions on the database. This further means that before inserting data into the database, we are checking for some conditions. If the condition we have applied to the database holds true for the data which is to be inserted, then only the data will be inserted into the database tables.

Constraints in SQL can be categorized into two types:

1. **Column Level Constraint:** Column Level Constraint is used to apply a constraint on a single column.
2. **Table Level Constraint:** Table Level Constraint is used to apply a constraint on multiple columns.

Some of the real-life examples of constraints are as follows:

1. Every person has a unique email id. This is because while creating an email account for any user, the email providing services such as Gmail, Yahoo or any other email providing service will always check for the availability of the email id that the user wants for himself. If some other user already takes the email id that the user wants, then that id cannot be assigned to another user. This simply means that no two users can have the same email ids on the same email providing service. So, here the email id is the constraint on the database of email providing services.
2. Whenever we set a password for any system, there are certain constraints that are to be followed. These constraints may include the following:
 - There must be one uppercase character in the password.
 - Password must be of at least eight characters in length.
 - Password must contain at least one special symbol.

Constraints available in SQL are:

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK
6. DEFAULT
7. CREATE INDEX

Now let us try to understand the different constraints available in SQL in more detail with the help of examples. We will use MySQL database for writing all the queries.

1. NOT NULL

- NULL means empty, i.e., the value is not available.
- Whenever a table's column is declared as NOT NULL, then the value for that column cannot be empty for any of the table's records.

- There must exist a value in the column to which the NOT NULL constraint is applied.

NOTE: NULL does not mean zero. NULL means an empty column, not even zero.

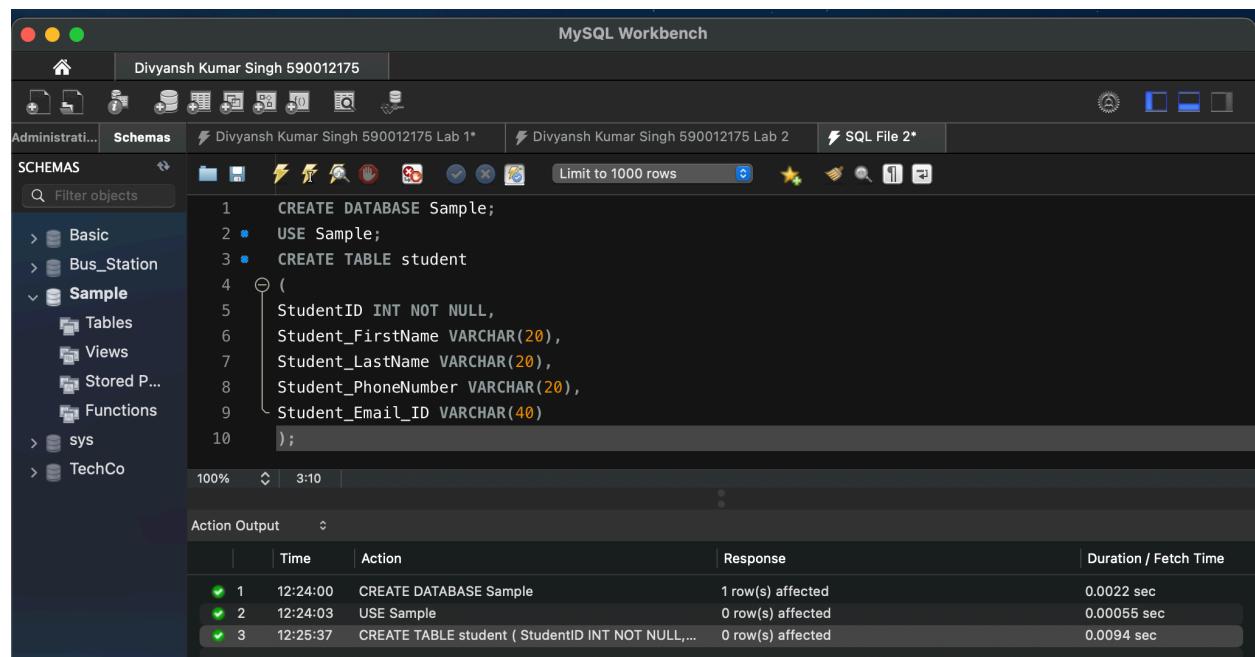
Syntax to apply the NOT NULL constraint during table creation:

1. **CREATE TABLE** TableName (ColumnName1 datatype NOT NULL, ColumnName2 datatype, ..., ColumnNameN datatype);

Example:

Create a student table and apply a NOT NULL constraint on one of the table's column while creating a table.

1. **CREATE TABLE** student(StudentID **INT** NOT NULL, Student_FirstName **VARCHAR(20)**, Student_LastName **VARCHAR(20)**, Student_PhoneNumber **VARCHAR(20)**, Student_Email_ID **VARCHAR(40)**);



The screenshot shows the MySQL Workbench interface. In the top navigation bar, the title is "MySQL Workbench" and the user is "Divyansh Kumar Singh 590012175". The left sidebar shows the "Schemas" section with "Sample" selected, containing "Tables", "Views", "Stored P...", and "Functions". The main SQL editor window contains the following code:

```

1 CREATE DATABASE Sample;
2 USE Sample;
3 CREATE TABLE student (
4     StudentID INT NOT NULL,
5     Student_FirstName VARCHAR(20),
6     Student_LastName VARCHAR(20),
7     Student_PhoneNumber VARCHAR(20),
8     Student_Email_ID VARCHAR(40)
9 );
10

```

The "Action Output" pane at the bottom shows the execution results:

Action	Time	Response	Duration / Fetch Time
CREATE DATABASE Sample	12:24:00	1 row(s) affected	0.0022 sec
USE Sample	12:24:03	0 row(s) affected	0.00055 sec
CREATE TABLE student (StudentID INT NOT NULL,...	12:25:37	0 row(s) affected	0.0094 sec

To verify that the not null constraint is applied to the table's column and the student table is created successfully, we will execute the following query:

1. mysql> **DESC** student;

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' tab, there is a tree view of databases: 'Basic', 'Bus_Station', 'Sample' (which is expanded to show 'Tables', 'Views', 'Stored P...', and 'Functions'), 'sys', and 'TechCo'. The main pane displays the following SQL code:

```

1 CREATE DATABASE Sample;
2 USE Sample;
3 CREATE TABLE student
4 (
5     StudentID INT NOT NULL,
6     Student_FirstName VARCHAR(20),
7     Student_LastName VARCHAR(20),
8     Student_PhoneNumber VARCHAR(20),
9     Student_Email_ID VARCHAR(40)
10 );
11
12 DESC student;
13

```

Below the code, the 'Result Grid' shows the structure of the 'student' table:

Field	Type	Null	Key	Default	Extra
StudentID	int	NO		NULL	
Student_FirstName	varchar(20)	YES		NULL	
Student_LastName	varchar(20)	YES		NULL	
Student_PhoneNumber	varchar(20)	YES		NULL	
Student_Email_ID	varchar(40)	YES		NULL	

The 'Action Output' section at the bottom lists the execution history:

Action	Time	Response	Duration / Fetch Time
CREATE DATABASE Sample	12:24:00	1 row(s) affected	0.0022 sec
USE Sample	12:24:03	0 row(s) affected	0.00055 sec
CREATE TABLE student (StudentID INT NOT NULL, Student_Firs...	12:25:37	0 row(s) affected	0.0094 sec
DESC student	12:28:45	5 row(s) returned	0.0026 sec / 0.00007...

Syntax to apply the NOT NULL constraint on an existing table's column:

1. **ALTER TABLE** TableName **CHANGE** Old_ColumnName New_ColumnName Datatype **NOT NULL;**

Example:

Consider we have an existing table student, without any constraints applied to it. Later, we decided to apply a NOT NULL constraint to one of the table's column. Then we will execute the following query:

1. mysql> **ALTER TABLE** student **CHANGE** StudentID StudentID **INT NOT NULL;**

```

CREATE DATABASE Sample;
USE Sample;
CREATE TABLE student (
    StudentID INT NOT NULL,
    Student_FirstName VARCHAR(20),
    Student_LastName VARCHAR(20),
    Student_PhoneNumber VARCHAR(20),
    Student_Email_ID VARCHAR(40)
);
DESC student;
ALTER TABLE student CHANGE StudentID StudentID INT NOT NULL;

```

Action Output

	Time	Action	Response	Duration / Fetch Time
1	12:24:00	CREATE DATABASE Sample	1 row(s) affected	0.0022 sec
2	12:24:03	USE Sample	0 row(s) affected	0.00055 sec
3	12:25:37	CREATE TABLE student (StudentID INT NOT NULL, Student_Firs...	0 row(s) affected	0.0094 sec
4	12:28:45	DESC student	5 row(s) returned	0.0026 sec / 0.00007...
5	12:31:03	ALTER TABLE student CHANGE StudentID StudentID INT NOT NU...	0 row(s) affected Records: 0 Duplicates:...	0.0081 sec

To verify that the not null constraint is applied to the student table's column, we will execute the following query:

2. mysql> **DESC student;**

```

DESC student;

```

Result Grid

Field	Type	Null	Key	Default	Extra
StudentID	int	NO			
Student_FirstName	varchar(20)	YES			
Student_LastName	varchar(20)	YES			
Student_PhoneNumber	varchar(20)	YES			
Student_Email_ID	varchar(40)	YES			

Action Output

	Time	Action	Response	Duration / Fetch Time
1	12:24:00	CREATE DATABASE Sample	1 row(s) affected	0.0022 sec
2	12:24:03	USE Sample	0 row(s) affected	0.00055 sec
3	12:25:37	CREATE TABLE student (StudentID INT NOT NULL, Student_Firs...	0 row(s) affected	0.0094 sec
4	12:28:45	DESC student	5 row(s) returned	0.0026 sec / 0.00007...
5	12:31:03	ALTER TABLE student CHANGE StudentID StudentID INT NOT NU...	0 row(s) affected Records: 0 Duplicates:...	0.0081 sec
6	12:32:13	DESC student	5 row(s) returned	0.0026 sec / 0.00001...

2. UNIQUE

- o Duplicate values are not allowed in the columns to which the UNIQUE constraint is applied.
- o The column with the unique constraint will always contain a unique value.
- o This constraint can be applied to one or more than one column of a table, which means more than one unique constraint can exist on a single table.
- o Using the UNIQUE constraint, you can also modify the already created tables.

Syntax to apply the UNIQUE constraint on a single column:

1. **CREATE TABLE** TableName (ColumnName1 datatype **UNIQUE**, ColumnName2 datatype ,..., ColumnNameN datatype);

Example:

Create a student table and apply a UNIQUE constraint on one of the table's column while creating a table.

1. mysql> **CREATE TABLE** student(StudentID **INT UNIQUE**, Student_FirstName **VARCHAR(20)**, Student_LastName **VARCHAR(20)**, Student_PhoneNumber **VARCHAR(20)**, Student_Email_ID **VARCHAR(40)**);

```
CREATE TABLE student
(
    StudentID INT UNIQUE,
    Student_FirstName VARCHAR(20),
    Student_LastName VARCHAR(20),
    Student_PhoneNumber VARCHAR(20),
    Student_Email_ID VARCHAR(40)
);
```

To verify that the unique constraint is applied to the table's column and the student table is created successfully, we will execute the following query:

2. mysql> **DESC** student;

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' tab, there is a 'Sample' schema expanded, showing 'Tables', 'Views', 'Stored Procs', and 'Functions'. In the main pane, the SQL editor contains the following code:

```
1 CREATE DATABASE Sample;
2 USE Sample;
3 CREATE TABLE student
4 (
5     StudentID INT UNIQUE,
6     Student_FirstName VARCHAR(20),
7     Student_LastName VARCHAR(20),
8     Student_PhoneNumber VARCHAR(20),
9     Student_Email_ID VARCHAR(40)
10 );
11
12 DESC student;
13
```

Below the SQL editor is a 'Result Grid' table showing the structure of the 'student' table:

Field	Type	Null	Key	Default	Extra
StudentID	INT	YES	UNI	NULL	
Student_FirstName	varchar(20)	YES		NULL	
Student_LastName	varchar(20)	YES		NULL	
Student_PhoneNumber	varchar(20)	YES		NULL	
Student_Email_ID	varchar(40)	YES		NULL	

At the bottom, the 'Action Output' section shows two log entries:

Time	Action	Response	Duration / Fetch Time
12:48:51	CREATE TABLE student [StudentID INT UNIQUE, Student_FirstN...	0 row(s) affected	0.0085 sec
12:48:55	DESC student	5 row(s) returned	0.0023 sec / 0.00001...

Syntax to apply the UNIQUE constraint on more than one column:

1. **CREATE TABLE** TableName (ColumnName1 datatype, ColumnName2 datatype,...., ColumnNameN datatype, **UNIQUE** (ColumnName1, ColumnName 2));

Example:

Create a student table and apply a UNIQUE constraint on more than one table's column while creating a table.

1. mysql> **CREATE TABLE** student(StudentID **INT**, Student_FirstName **VARCHAR**(20), Student_LastName **VARCHAR**(20), Student_PhoneNumber **VARCHAR**(20), Student_Email_ID **VARCHAR**(40), **UNIQUE**(StudentID, Student_PhoneNumber));

```
CREATE TABLE student
(
    StudentID INT,
    Student_FirstName VARCHAR(20),
    Student_LastName VARCHAR(20),
    Student_PhoneNumber VARCHAR(20),
    Student_Email_ID VARCHAR(40),
    UNIQUE(StudentID, Student_PhoneNumber)
);
```

To verify that the unique constraint is applied to more than one table's column and the student table is created successfully, we will execute the following query:

2. mysql> **DESC** student;

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' tab, there is a database named 'Sample'. Inside 'Sample', a table named 'student' is defined with the following structure:

```

CREATE TABLE student
(
    StudentID INT,
    Student_FirstName VARCHAR(20),
    Student_LastName VARCHAR(20),
    Student_PhoneNumber VARCHAR(20),
    Student_Email_ID VARCHAR(40),
    UNIQUE(StudentID, Student_PhoneNumber)
);

```

In the main SQL editor area, the command `DESC student;` is run, and the results are displayed in a table:

Field	Type	Null	Key	Default	Extra
StudentID	int	YES	MUL		
Student_FirstName	varchar(20)	YES			
Student_LastName	varchar(20)	YES			
Student_PhoneNumber	varchar(20)	YES			
Student_Email_ID	varchar(40)	YES			

Below the table, the status bar indicates "Result 5". At the bottom, the "Action Output" section shows two log entries:

Action	Time	Response	Duration / Fetch Time
CREATE TABLE student (StudentID INT, Student_FirstName VARCHAR(2...	13:09:53	0 row(s) affected	0.010 sec
DESC student	13:09:57	5 row(s) returned	0.0024 sec / 0.00001...

Syntax to apply the UNIQUE constraint on an existing table's column:

1. **ALTER TABLE TableName ADD UNIQUE (ColumnName);**

Example:

Consider we have an existing table student, without any constraints applied to it. Later, we decided to apply a UNIQUE constraint to one of the table's column. Then we will execute the following query:

2. mysql> **ALTER TABLE student ADD UNIQUE (StudentID);**

```
CREATE TABLE student
(
    StudentID INT,
    Student_FirstName VARCHAR(20),
    Student_LastName VARCHAR(20),
    Student_PhoneNumber VARCHAR(20),
    Student_Email_ID VARCHAR(40)
);

ALTER TABLE student ADD UNIQUE (StudentID);

DESC student;
```

To verify that the unique constraint is applied to the table's column and the student table is created successfully, we will execute the following query:

3. mysql> **DESC student;**

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the SQL code for creating the student table and adding a unique constraint to the StudentID column.
- Result Grid:** Displays the description of the student table, showing the columns and their properties. The StudentID column is identified as the primary key (Key: UNI).
- Action Output:** Shows the log of database actions with their timestamps, affected rows, and execution times.

```
USE Sample;
CREATE TABLE student
(
    StudentID INT,
    Student_FirstName VARCHAR(20),
    Student_LastName VARCHAR(20),
    Student_PhoneNumber VARCHAR(20),
    Student_Email_ID VARCHAR(40)
);
ALTER TABLE student ADD UNIQUE (StudentID);
DESC student;
```

Field	Type	Null	Key	Default	Extra
StudentID	int	YES	UNI	NULL	
Student_FirstName	varchar(20)	YES		NULL	
Student_LastName	varchar(20)	YES		NULL	
Student_PhoneNumber	varchar(20)	YES		NULL	
Student_Email_ID	varchar(40)	YES		NULL	

Time	Action	Response	Duration / Fetch Time
13:11:27	CREATE TABLE student (StudentID INT, Student_FirstName VARCHAR(2...	0 row(s) affected	0.0083 sec
13:11:39	DESC student	5 row(s) returned	0.0022 sec / 0.00001...
13:12:22	ALTER TABLE student ADD UNIQUE (StudentID)	0 row(s) affected Records: 0 Du...	0.0095 sec
13:12:26	DESC student	5 row(s) returned	0.0018 sec / 0.00001...

3. PRIMARY KEY

- o PRIMARY KEY Constraint is a combination of NOT NULL and Unique constraints.
- o NOT NULL constraint and a UNIQUE constraint together forms a PRIMARY constraint.
- o The column to which we have applied the primary constraint will always contain a unique value and will not allow null values.

Syntax of primary key constraint during table creation:

1. **CREATE TABLE** TableName (ColumnName1 datatype **PRIMARY KEY**, ColumnName2 datatype, ..., ColumnNameN datatype);

Example:

Create a student table and apply the PRIMARY KEY constraint while creating a table.

1. mysql> **CREATE TABLE** student(StudentID **INT PRIMARY KEY**, Student_FirstName **VARCHAR(20)**, Student_LastName **VARCHAR(20)**, Student_PhoneNumber **VARCHAR(20)**, Student_Email_ID **VARCHAR(40)**);

```
CREATE TABLE student
(
  StudentID INT PRIMARY KEY,
  Student_FirstName VARCHAR(20),
  Student_LastName VARCHAR(20),
  Student_PhoneNumber VARCHAR(20),
  Student_Email_ID VARCHAR(40)
);
```

To verify that the primary key constraint is applied to the table's column and the student table is created successfully, we will execute the following query:

2. mysql> **DESC** student;

The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'Schemas', there is a tree view with 'Sample' selected. The main pane displays the following SQL code:

```

1 CREATE DATABASE Sample;
2 USE Sample;
3 CREATE TABLE student
4 (
5   StudentID INT PRIMARY KEY,
6   Student_FirstName VARCHAR(20),
7   Student_LastName VARCHAR(20),
8   Student_PhoneNumber VARCHAR(20),
9   Student_Email_ID VARCHAR(40)
10 );
11
12 DESC student;
13
14:12

```

Below the code, the 'Result Grid' shows the structure of the 'student' table:

Field	Type	Null	Key	Default	Extra
StudentID	int	NO	PRI	NULL	
Student_FirstName	varchar(20)	YES		NULL	
Student_LastName	varchar(20)	YES		NULL	
Student_PhoneNumber	varchar(20)	YES		NULL	
Student_Email_ID	varchar(40)	YES		NULL	

The 'Action Output' section at the bottom shows two log entries:

Action	Time	Response	Duration / Fetch Time
CREATE TABLE student (StudentID INT PRIMARY KEY, Student_FirstName...	13:18:42	0 row(s) affected	0.0082 sec
DESC student	13:18:47	5 row(s) returned	0.0014 sec / 0.0000...

Syntax to apply the primary key constraint on an existing table's column:

1. **ALTER TABLE TableName ADD PRIMARY KEY (ColumnName);**

Example:

Consider we have an existing table student, without any constraints applied to it. Later, we decided to apply the PRIMARY KEY constraint to the table's column. Then we will execute the following query:

1. mysql> **ALTER TABLE student ADD PRIMARY KEY (StudentID);**

```

3 •  CREATE TABLE student
4   (
5     StudentID INT,
6     Student_FirstName VARCHAR(20),
7     Student_LastName VARCHAR(20),
8     Student_PhoneNumber VARCHAR(20),
9     Student_Email_ID VARCHAR(40)
10    );
11
12 •  ALTER TABLE student ADD PRIMARY KEY (StudentID);
13
14 •  DESC student;

```

To verify that the primary key constraint is applied to the student table's column, we will execute the following query:

2. mysql> **DESC student;**

The screenshot shows the MySQL Workbench interface. In the top-left corner, there are three colored dots (red, yellow, green) and the title "Divyansh Kumar Singh 590012175". Below the title, the "Schemas" tab is selected, showing a tree view with "Basic", "Bus_Station", "Sample" (which is expanded to show "Tables", "Views", "Stored P...", and "Functions"), "sys", and "TechCo". The main area contains the following SQL code:

```

3 •  CREATE TABLE student
4   (
5     StudentID INT,
6     Student_FirstName VARCHAR(20),
7     Student_LastName VARCHAR(20),
8     Student_PhoneNumber VARCHAR(20),
9     Student_Email_ID VARCHAR(40)
10    );
11
12 •  ALTER TABLE student ADD PRIMARY KEY (StudentID);
13
14 •  DESC student;
15

```

Below the code, the "Result Grid" pane displays the table structure:

Field	Type	Null	Key	Default	Extra
StudentID	int	NO	PRI	NULL	
Student_FirstName	varchar(20)	YES		NULL	
Student_LastName	varchar(20)	YES		NULL	
Student_PhoneNumber	varchar(20)	YES		NULL	
Student_Email_ID	varchar(40)	YES		NULL	

The "Result" section shows the output of the DESCRIBE command:

Result 9

Action Output

Time	Action	Response	Duration / Fetch Time
13:22:24	CREATE TABLE student (StudentID INT, Student_FirstName VARCHAR(2...	0 row(s) affected	0.0087 sec
13:22:28	ALTER TABLE student ADD PRIMARY KEY (StudentID)	0 row(s) affected Records: 0 Du...	0.013 sec
13:22:31	DESC student	5 row(s) returned	0.0020 sec / 0.00001...

4. FOREIGN KEY

- A foreign key is used for referential integrity.
- When we have two tables, and one table takes reference from another table, i.e., the same column is present in both the tables and that column acts as a primary key in one table. That particular column will act as a foreign key in another table.

Syntax to apply a foreign key constraint during table creation:

1. **CREATE TABLE** tablename(ColumnName1 Datatype(**SIZE**) **PRIMARY KEY**, ColumnNameN Datatype(**SIZE**), **FOREIGN KEY**(ColumnName) **REFERENCES** PARENT_TABLE_NAME(Primary_Key_ColumnName));

Example:

Create an employee table and apply the FOREIGN KEY constraint while creating a table.
To create a foreign key on any table, first, we need to create a primary key on a table.

1. mysql> **CREATE TABLE** employee (Emp_ID **INT NOT NULL PRIMARY KEY**, Emp_Name **VARCHAR** (40), Emp_Salary **VARCHAR** (40));

```
CREATE TABLE employee
(
    Emp_ID INT NOT NULL PRIMARY KEY,
    Emp_Name VARCHAR (40),
    Emp_Salary VARCHAR (40)
);
```

To verify that the primary key constraint is applied to the employee table's column, we will execute the following query:

2. mysql> **DESC** employee;

The screenshot shows the MySQL Workbench interface. In the Schemas pane, under the Sample database, the Tables section is selected. A SQL editor window displays the following code:

```

1 CREATE DATABASE Sample;
2 USE Sample;
3 CREATE TABLE employee
4 (
5     Emp_ID INT NOT NULL PRIMARY KEY,
6     Emp_Name VARCHAR (40),
7     Emp_Salary VARCHAR (40)
8 );
9
10 DESC employee;
11

```

Below the editor, the Result Grid shows the structure of the employee table:

Field	Type	Null	Key	Default	Extra
Emp_ID	int	NO	PRI		HULL
Emp_Name	varchar(40)	YES			HULL
Emp_Salary	varchar(40)	YES			HULL

The Action Output pane at the bottom shows the execution history:

Action	Time	Response	Duration / Fetch Time
CREATE TABLE employee (Emp_ID INT NOT NULL PRIMARY KEY, Emp_N...	16:24:26	0 row(s) affected	0.0084 sec
DESC employee	16:24:38	3 row(s) returned	0.0023 sec / 0.00001...

Now, we will write a query to apply a foreign key on the department table referring to the primary key of the employee table, i.e., Emp_ID.

3. mysql> **CREATE TABLE** department(Dept_ID **INT** NOT NULL **PRIMARY KEY**, Dept_Name **VARCHAR**(40), Emp_ID **INT** NOT NULL, **FOREIGN KEY**(Emp_ID) **REFERENCES** employee(Emp_ID));

```
12 • CREATE TABLE department
13 ((
14   Dept_ID INT NOT NULL PRIMARY KEY,
15   Dept_Name VARCHAR(40),
16   Emp_ID INT NOT NULL,
17   FOREIGN KEY(Emp_ID) REFERENCES employee(Emp_ID)
18 );
19
```

To verify that the foreign key constraint is applied to the department table's column, we will execute the following query:

4. mysql> **DESC** department;

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the title is "MySQL Workbench" and the user is "Divyansh Kumar Singh 590012175". The left sidebar shows the "Schemas" tree with "Sample" selected, containing "Tables", "Views", "Stored Procs", and "Functions". The main SQL editor window contains the following code:

```
11
12 • CREATE TABLE department
13 ((
14   Dept_ID INT NOT NULL PRIMARY KEY,
15   Dept_Name VARCHAR(40),
16   Emp_ID INT NOT NULL,
17   FOREIGN KEY(Emp_ID) REFERENCES employee(Emp_ID)
18 );
19
20 • DESC department;
21
```

Below the code, the "Result Grid" shows the table structure:

Field	Type	Null	Key	Default	Extra
Dept_ID	int	NO	PRI	NULL	
Dept_Name	varchar(40)	YES		NULL	
Emp_ID	int	NO	MUL	NULL	

The status bar at the bottom indicates "Result 11" and "Read Only". The "Action Output" section shows the execution history:

	Time	Action	Response	Duration / Fetch Time
1	16:24:26	CREATE TABLE employee (Emp_ID INT NOT NULL PRIMARY KEY, Emp_N...	0 row(s) affected	0.0084 sec
2	16:24:38	DESC employee	3 row(s) returned	0.0023 sec / 0.00001...
3	16:26:34	CREATE TABLE department (Dept_ID INT NOT NULL PRIMARY KEY, Dept...	0 row(s) affected	0.0095 sec
4	16:27:25	DESC department	3 row(s) returned	0.0028 sec / 0.00001...

Syntax to apply the foreign key constraint with constraint name:

1. **CREATE TABLE** tablename(Column Name1 Datatype **PRIMARY KEY**, Column NameN Datatype(**SIZE**), **CONSTRAINT** ConstraintName **FOREIGN KEY**(Column Name) **REFERENCES** PARENT_TABLE_NAME(Primary Key Column Name));

Example:

Create an employee table and apply the FOREIGN KEY constraint with a constraint name while creating a table.

To create a foreign key on any table, first, we need to create a primary key on a table.

1. mysql> **CREATE TABLE** employee (Emp_ID INT NOT NULL **PRIMARY KEY**, Emp_Name VARCHAR (40), Emp_Salary VARCHAR (40));

```

3 •   CREATE TABLE employee
4   (
5     Emp_ID INT NOT NULL PRIMARY KEY,
6     Emp_Name VARCHAR (40),
7     Emp_Salary VARCHAR (40)
8   );

```

To verify that the primary key constraint is applied to the student table's column, we will execute the following query:

2. mysql> **DESC** employee;

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' tab, the 'Sample' schema is selected. In the main SQL editor area, the following code is shown:

```

1 CREATE DATABASE Sample;
2 USE Sample;
3 CREATE TABLE employee
4   (
5     Emp_ID INT NOT NULL PRIMARY KEY,
6     Emp_Name VARCHAR (40),
7     Emp_Salary VARCHAR (40)
8   );
9
10 DESC employee;
11

```

Below the code, the 'Result Grid' shows the table structure:

Field	Type	Null	Key	Default	Extra
Emp_ID	int	NO	PRI	NULL	
Emp_Name	varchar(40)	YES		NULL	
Emp_Salary	varchar(40)	YES		NULL	

The 'Action Output' pane at the bottom shows the execution log:

Action	Time	Response	Duration / Fetch Time
CREATE TABLE employee (Emp_ID INT NOT NULL PRIMARY KEY, Emp_Name VARCHAR...	16:42:37	0 row(s) affected	0.0084 sec
DESC employee	16:42:44	3 row(s) returned	0.0021 sec / 0.00001...

Now, we will write a query to apply a foreign key with a constraint name on the department table referring to the primary key of the employee table, i.e., Emp_ID.

3. mysql> **CREATE TABLE** department(Dept_ID **INT NOT NULL PRIMARY KEY**, Dept_Name **VARCHAR(40)**, Emp_ID **INT NOT NULL**, **CONSTRAINT** emp_id_fk **FOREIGN KEY**(Emp_ID) **REFERENCES** employee(Emp_ID));

```

12 *   CREATE TABLE department
13   (
14     Dept_ID INT NOT NULL PRIMARY KEY,
15     Dept_Name VARCHAR(40),
16     Emp_ID INT NOT NULL,
17     CONSTRAINT emp_id_fk FOREIGN KEY(Emp_ID) REFERENCES employee(Emp_ID));
18

```

To verify that the foreign key constraint is applied to the department table's column, we will execute the following query:

4. mysql> **DESC** department;

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Basic, Bus_Station, Sample (Tables, Views, Stored P..., Functions), sys, TechCo.
- Current Schema:** Sample
- SQL Editor:**

```

10 *   DESC employee;
11
12 *   CREATE TABLE department
13   (
14     Dept_ID INT NOT NULL PRIMARY KEY,
15     Dept_Name VARCHAR(40),
16     Emp_ID INT NOT NULL,
17     CONSTRAINT emp_id_fk FOREIGN KEY(Emp_ID) REFERENCES employee(Emp_ID));
18
19 *   DESC department;
20

```
- Result Grid:**

Field	Type	Null	Key	Default	Extra
Dept_ID	int	NO	PRI		
Dept_Name	varchar(40)	YES			
Emp_ID	int	NO	MUL		

Result 13 | Read Only
- Action Output:**

	Time	Action	Response	Duration / Fetch Time
1	16:42:37	CREATE TABLE employee (Emp_ID INT NOT NULL PRIMARY KEY, Emp_Name VARCHAR...	0 row(s) affected	0.0084 sec
2	16:42:44	DESC employee	3 row(s) returned	0.0021 sec / 0.00001...
3	16:44:19	CREATE TABLE department (Dept_ID INT NOT NULL PRIMARY KEY, Dept_Name VARC...	0 row(s) affected	0.0098 sec
4	16:44:23	DESC department	3 row(s) returned	0.0029 sec / 0.00001...

Syntax to apply the foreign key constraint on an existing table's column:

1. **ALTER TABLE Parent_TableName ADD FOREIGN KEY (ColumnName) REFERENCE S Child_TableName (ColumnName);**

Example: Consider we have an existing table employee and department. Later, we decided to apply a FOREIGN KEY constraint to the department table's column. Then we will execute the following query:

1. mysql> **DESC employee;**

The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel (Schemas):** Shows the Sample schema expanded, displaying Tables, Views, Stored Procedures, and Functions.
- Central Editor Area:**
 - SQL History: A list of SQL statements including `CREATE DATABASE Sample;` and `USE Sample;` followed by the creation of the `employee` table.
 - Table Definition: The `employee` table definition is shown with columns: Emp_ID (INT, primary key), Emp_Name (VARCHAR(40)), and Emp_Salary (VARCHAR(40)).
 - Result Grid: A table showing the structure of the `employee` table with columns: Field, Type, Null, Key, Default, Extra.
 - Action Output: A log of actions with rows 1 and 2, both marked with a green checkmark.
- Bottom Status Bar:** Shows the current time as 15:10.

The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel (Schemas):** Shows the Sample schema expanded, displaying Tables, Views, Stored Procedures, and Functions.
- Central Editor Area:**
 - SQL History: A list of SQL statements including `CREATE TABLE department` with columns: Dept_ID (INT, primary key), Dept_Name (VARCHAR(40)), and Emp_ID (INT).
 - Table Definition: The `department` table definition is shown with columns: Dept_ID (INT, primary key), Dept_Name (VARCHAR(40)), and Emp_ID (INT).
 - Result Grid: A table showing the structure of the `department` table with columns: Field, Type, Null, Key, Default, Extra.
 - Action Output: A log of actions with rows 1 through 4, all marked with a green checkmark.
 - Bottom Status Bar:

2. mysql> **ALTER TABLE** department **ADD FOREIGN KEY** (Emp_ID) **REFERENCES** employee (Emp_ID);

```
21 • ALTER TABLE department ADD FOREIGN KEY (Emp_ID) REFERENCES employee (Emp_ID);
```

To verify that the foreign key constraint is applied to the department table's column, we will execute the following query:

3. mysql> **DESC** department;

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Sample (selected)
- Tables:** department
- SQL Editor:**

```

14 |     Dept_ID INT NOT NULL PRIMARY KEY,
15 |     Dept_Name VARCHAR(40),
16 |     Emp_ID INT NOT NULL
17 | );
18
19 • DESC department;
20
21 • ALTER TABLE department ADD FOREIGN KEY (Emp_ID) REFERENCES employee (Emp_ID);
22
23 • DESC department;
24
    
```
- Result Grid:**

Field	Type	Null	Key	Default	Extra
Dept_ID	int	NO	PRI	NULL	
Dept_Name	varchar(40)	YES		NULL	
Emp_ID	int	NO	MUL	NULL	
- Action Output:**

	Time	Action	Response	Duration / Fetch Time
✓ 1	16:46:48	CREATE TABLE employee (Emp_ID INT NOT NULL PRIMARY KEY, Emp_Name VARCHAR...	0 row(s) affected	0.0073 sec
✓ 2	16:46:51	DESC employee	3 row(s) returned	0.0024 sec / 0.00000...
✓ 3	16:47:45	CREATE TABLE department (Dept_ID INT NOT NULL PRIMARY KEY, Dept_Name VARC...	0 row(s) affected	0.0080 sec
✓ 4	16:47:49	DESC department	3 row(s) returned	0.0023 sec / 0.00001...
✓ 5	16:48:47	ALTER TABLE department ADD FOREIGN KEY (Emp_ID) REFERENCES employee (Emp_ID);	0 row(s) affected R...	0.019 sec
✓ 6	16:49:00	DESC department	3 row(s) returned	0.0019 sec / 0.00002...

5. CHECK

o Whenever a check constraint is applied to the table's column, and the user wants to insert the value in it, then the value will first be checked for certain conditions before inserting the value into that column.

o **For example:** if we have an age column in a table, then the user will insert any value of his choice. The user will also enter even a negative value or any other invalid value. But, if the user has applied check constraint on the age column with the condition age greater than 18. Then in such cases, even if a user tries to insert an invalid value such as zero or any other value less than 18, then the age column will not accept that value and will not allow the user to insert it due to the application of check constraint on the age column.

Syntax to apply check constraint on a single column:

1. **CREATE TABLE** TableName (ColumnName1 datatype **CHECK** (ColumnName1 Condition), ColumnName2 datatype, ..., ColumnNameN datatype);

Example:

Create a student table and apply CHECK constraint to check for the age less than or equal to 15 while creating a table.

1. mysql> **CREATE TABLE** student(StudentID **INT**, Student_FirstName **VARCHAR(20)**, Student_LastName **VARCHAR(20)**, Student_PhoneNumber **VARCHAR(20)**, Student_Email_ID **VARCHAR(40)**, Age **INT CHECK**(Age <= 15));

```
3 •   CREATE TABLE student
4   (
5     StudentID INT,
6     Student_FirstName VARCHAR(20),
7     Student_LastName VARCHAR(20),
8     Student_PhoneNumber VARCHAR(20),
9     Student_Email_ID VARCHAR(40),
10    Age INT CHECK( Age <= 15)
11  );
```

To verify that the check constraint is applied to the student table's column, we will execute the following query:

2. mysql> **DESC** student;

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the title is "Divyansh Kumar Singh 590012175". Below it, the "Schemas" tab is selected, showing a list of databases: Basic, Bus_Station, Sample, Tables, Views, Stored P..., Functions, sys, and TechCo. The "Tables" section is expanded, showing the "student" table. The SQL editor pane contains the following code:

```

3 * CREATE TABLE student
4 (
5     StudentID INT,
6     Student_FirstName VARCHAR(20),
7     Student_LastName VARCHAR(20),
8     Student_PhoneNumber VARCHAR(20),
9     Student_Email_ID VARCHAR(40),
10    Age INT CHECK( Age <= 15 )
11 );
13 * DESC student;

```

The "Result Grid" pane displays the structure of the "student" table:

Field	Type	Null	Key	Default	Extra
StudentID	int	YES			
Student_FirstName	varchar(20)	YES			
Student_LastName	varchar(20)	YES			
Student_PhoneNumber	varchar(20)	YES			
Student_Email_ID	varchar(40)	YES			
Age	int	YES			

The "Action Output" pane shows two log entries:

Time	Action	Response	Duration / Fetch Time
17:56:20	CREATE TABLE student (StudentID INT, Student_FirstName VARCHAR(20), Student_LastName VARCHAR(20), Student_PhoneNumber VARCHAR(20), Student_Email_ID VARCHAR(40), Age INT, Percentage INT, CHECK(Age <= 15 AND Percentage > 85));	0 row(s) affected	0.0071 sec
17:56:23	DESC student	6 row(s) returned	0.0025 sec / 0.00001...

Syntax to apply check constraint on multiple columns:

1. **CREATE TABLE** TableName (ColumnName1 datatype, ColumnName2 datatype **CHECK** (ColumnName1 Condition **AND** ColumnName2 Condition),..., ColumnNameN datatype);

Example:

Create a student table and apply CHECK constraint to check for the age less than or equal to 15 and a percentage greater than 85 while creating a table.

1. mysql> **CREATE TABLE** student(StudentID **INT**, Student_FirstName **VARCHAR**(20), Student_LastName **VARCHAR**(20), Student_PhoneNumber **VARCHAR**(20), Student_Email_ID **VARCHAR**(40), Age **INT**, Percentage **INT**, **CHECK**(Age <= 15 **AND** Percentage > 85));

The terminal window shows the following SQL command:

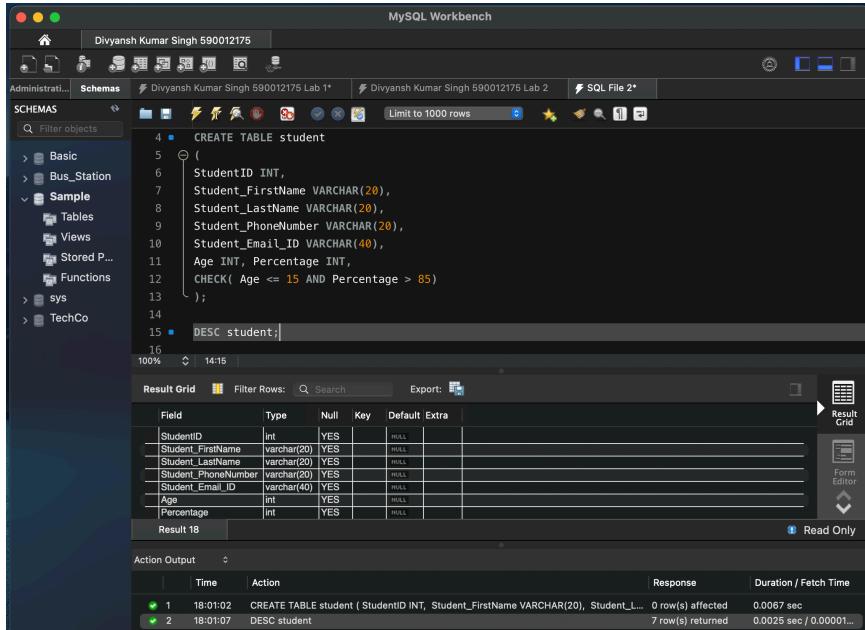
```

3 * CREATE TABLE student
4 (
5     StudentID INT,
6     Student_FirstName VARCHAR(20),
7     Student_LastName VARCHAR(20),
8     Student_PhoneNumber VARCHAR(20),
9     Student_Email_ID VARCHAR(40),
10    Age INT, Percentage INT,
11    CHECK( Age <= 15 AND Percentage > 85 )
12 );

```

To verify that the check constraint is applied to the age and percentage column, we will execute the following query:

2. mysql> **DESC student;**



The screenshot shows the MySQL Workbench interface. In the SQL editor, the following SQL code is present:

```

4 CREATE TABLE student
5 (
6     StudentID INT,
7     Student_FirstName VARCHAR(20),
8     Student_LastName VARCHAR(20),
9     Student_PhoneNumber VARCHAR(20),
10    Student_Email_ID VARCHAR(40),
11    Age INT, Percentage INT,
12    CHECK( Age <= 15 AND Percentage > 85)
13 );
14
15 DESC student;
16

```

The Result Grid shows the structure of the student table:

Field	Type	Null	Key	Default	Extra
StudentID	int	YES			
Student_FirstName	varchar(20)	YES			
Student_LastName	varchar(20)	YES			
Student_PhoneNumber	varchar(20)	YES			
Student_Email_ID	varchar(40)	YES			
Age	int	YES			
Percentage	int	YES			

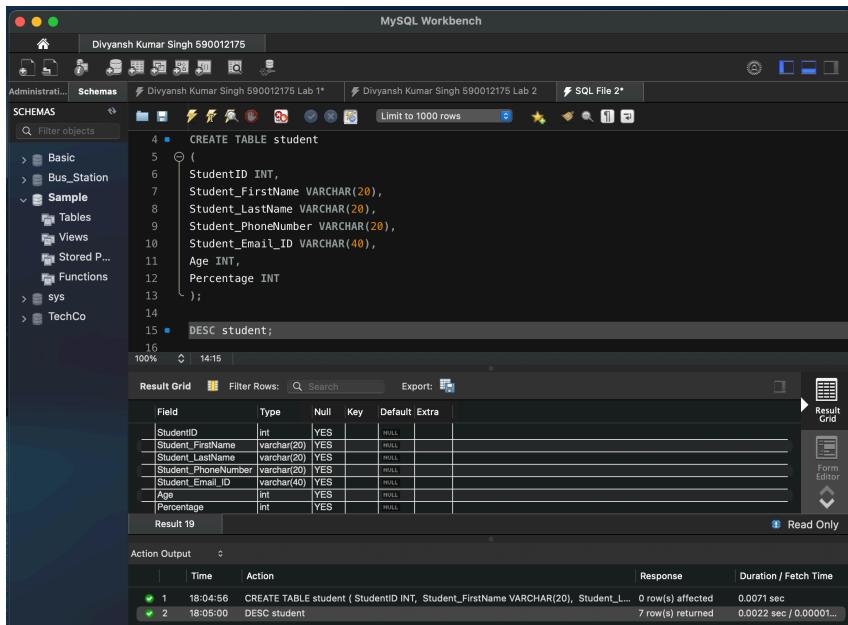
The Action Output pane shows the results of the DESCRIBE command:

Action	Time	Response	Duration / Fetch Time
CREATE TABLE student (StudentID INT, Student_FirstName VARCHAR(20), Student_L...	18:01:02	0 row(s) affected	0.0067 sec
DESC student	18:01:07	7 row(s) returned	0.0025 sec / 0.00001...

Syntax to apply check constraint on an existing table's column:

1. **ALTER TABLE TableName ADD CHECK (ColumnName Condition);**

Example: Consider we have an existing table student. Later, we decided to apply the CHECK constraint on the student table's column. Then we will execute the following query:



The screenshot shows the MySQL Workbench interface. In the SQL editor, the following SQL code is present:

```

4 CREATE TABLE student
5 (
6     StudentID INT,
7     Student_FirstName VARCHAR(20),
8     Student_LastName VARCHAR(20),
9     Student_PhoneNumber VARCHAR(20),
10    Student_Email_ID VARCHAR(40),
11    Age INT,
12    Percentage INT
13 );
14
15 DESC student;
16

```

The Result Grid shows the structure of the student table:

Field	Type	Null	Key	Default	Extra
StudentID	int	YES			
Student_FirstName	varchar(20)	YES			
Student_LastName	varchar(20)	YES			
Student_PhoneNumber	varchar(20)	YES			
Student_Email_ID	varchar(40)	YES			
Age	int	YES			
Percentage	int	YES			

The Action Output pane shows the results of the DESCRIBE command:

Action	Time	Response	Duration / Fetch Time
CREATE TABLE student (StudentID INT, Student_FirstName VARCHAR(20), Student_L...	18:04:56	0 row(s) affected	0.0071 sec
DESC student	18:05:00	7 row(s) returned	0.0022 sec / 0.00001...

- mysql> **ALTER TABLE** student **ADD CHECK** (Age <=15);

```
17 •      ALTER TABLE student ADD CHECK ( Age <=15 );
```

To verify that the check constraint is applied to the student table's column, we will execute the following query:

- mysql> **DESC** student;

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' tab, the 'Sample' schema is selected. In the main pane, the SQL editor contains the following code:

```

4 • CREATE TABLE student
5 • (
6     StudentID INT,
7     Student_FirstName VARCHAR(20),
8     Student_LastName VARCHAR(20),
9     Student_PhoneNumber VARCHAR(20),
10    Student_Email_ID VARCHAR(40),
11    Age INT,
12    Percentage INT
13 );
14
15 • DESC student;
16
17 • ALTER TABLE student ADD CHECK ( Age <=15 );
18
19 • DESC student;
20

```

Below the SQL editor is a 'Result Grid' showing the structure of the 'student' table:

Field	Type	Null	Key	Default	Extra
StudentID	int	YES		NULL	
Student_FirstName	varchar(20)	YES		NULL	
Student_LastName	varchar(20)	YES		NULL	
Student_PhoneNumber	varchar(20)	YES		NULL	
Student_Email_ID	varchar(40)	YES		NULL	
Age	int	YES		NULL	
Percentage	int	YES		NULL	

At the bottom, the 'Action Output' section shows the execution history:

Time	Action	Response	Duration / Fetch Time
18:04:56	CREATE TABLE student (StudentID INT, Student_FirstName VARCHAR(20), Student_L...	0 row(s) affected	0.0071 sec
18:05:00	DESC student	7 row(s) returned	0.0022 sec / 0.00001...
18:07:46	ALTER TABLE student ADD CHECK (Age <=15)	0 row(s) affected R...	0.019 sec
18:07:57	DESC student	7 row(s) returned	0.0021 sec / 0.00001...

6. DEFAULT

Whenever a default constraint is applied to the table's column, and the user has not specified the value to be inserted in it, then the default value which was specified while applying the default constraint will be inserted into that particular column.

Syntax to apply default constraint during table creation:

- CREATE TABLE** TableName (ColumnName1 datatype **DEFAULT** Value, ColumnName2 datatype, ..., ColumnNameN datatype);

Example:

Create a student table and apply the default constraint while creating a table.

1. mysql> **CREATE TABLE** student(StudentID **INT**, Student_FirstName **VARCHAR(20)**, Student_LastName **VARCHAR(20)**, Student_PhoneNumber **VARCHAR(20)**, Student_Email_ID **VARCHAR(40)** **DEFAULT** “anuja.k8@gmail.com”);

```
4 •      CREATE TABLE student
5   (
6     StudentID INT,
7     Student_FirstName VARCHAR(20),
8     Student_LastName VARCHAR(20),
9     Student_PhoneNumber VARCHAR(20),
10    Student_Email_ID VARCHAR(40) DEFAULT 'anuja.k8@gmail.com'
11 );
```

To verify that the default constraint is applied to the student table's column, we will execute the following query:

2. mysql> **DESC** student;

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The left sidebar shows the database structure with a tree view. Under the 'Sample' schema, there are tables, views, stored procedures, functions, and other objects.
- SQL Editor:** The main area contains two SQL queries:
 - Line 1: `CREATE DATABASE Sample;`
 - Line 2: `USE Sample;`
 - Line 4: `CREATE TABLE student`
 - Line 5: `(`
 - Line 6: `StudentID INT,`
 - Line 7: `Student_FirstName VARCHAR(20),`
 - Line 8: `Student_LastName VARCHAR(20),`
 - Line 9: `Student_PhoneNumber VARCHAR(20),`
 - Line 10: `Student_Email_ID VARCHAR(40) DEFAULT 'anuja.k8@gmail.com'`
 - Line 11: `);`
 - Line 13: `DESC student;`
 - Line 14: `;`
- Result Grid:** Below the SQL editor, the 'Result Grid' tab displays the description of the 'student' table with the following columns:

Field	Type	Null	Key	Default	Extra
StudentID	int	YES		NULL	
Student_FirstName	varchar(20)	YES		NULL	
Student_LastName	varchar(20)	YES		NULL	
Student_PhoneNumber	varchar(20)	YES		NULL	
Student_Email_ID	varchar(40)	YES		'anuja.k8@gmail.com'	
- Action Output:** At the bottom, the 'Action Output' section shows the execution history with two entries:

Time	Action	Response	Duration / Fetch Time
18:17:15	CREATE TABLE student (StudentID INT, Student_FirstName VARCHAR(20), Student_L...	0 row(s) affected	0.0068 sec
18:17:18	DESC student	5 row(s) returned	0.0040 sec / 0.00002...

Syntax to apply default constraint on an existing table's column:

1. **ALTER TABLE** TableName **ALTER** ColumnName **SET DEFAULT** Value;

Example:

Consider we have an existing table student. Later, we decided to apply the DEFAULT constraint on the student table's column. Then we will execute the following query:

1. mysql> **ALTER TABLE** student **ALTER** Student_Email_ID **SET DEFAULT** "anuja.k8@gmail.com";

```

4 • CREATE TABLE student
5   (
6     StudentID INT,
7     Student_FirstName VARCHAR(20),
8     Student_LastName VARCHAR(20),
9     Student_PhoneNumber VARCHAR(20),
10    Student_Email_ID VARCHAR(40)
11  );
12
13 • ALTER TABLE student ALTER Student_Email_ID SET DEFAULT 'anuja.k8@gmail.com';

```

To verify that the default constraint is applied to the student table's column, we will execute the following query:

1. mysql> **DESC** student;

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' tab, there is a tree view of databases: Basic, Bus_Station, Sample (Tables, Views, Stored P..., Functions), sys, and TechCo. The 'Sample' database is selected. The main area is titled 'Divyansh Kumar Singh 590012175'. The SQL editor contains the following code:

```

3
4 • CREATE TABLE student
5   (
6     StudentID INT,
7     Student_FirstName VARCHAR(20),
8     Student_LastName VARCHAR(20),
9     Student_PhoneNumber VARCHAR(20),
10    Student_Email_ID VARCHAR(40)
11  );
12
13 • ALTER TABLE student ALTER Student_Email_ID SET DEFAULT 'anuja.k8@gmail.com';
14
15 • DESC student;
16

```

The results grid shows the table structure:

Field	Type	Null	Key	Default	Extra
StudentID	int	YES		NULL	
Student_FirstName	varchar(20)	YES		NULL	
Student_LastName	varchar(20)	YES		NULL	
Student_PhoneNumber	varchar(20)	YES		NULL	
Student_Email_ID	varchar(40)	YES		'anuja.k8@gmail.com'	

The log at the bottom shows the execution of the commands:

Action	Time	Action	Response	Duration / Fetch Time
1	18:18:47	CREATE TABLE student (StudentID INT, Student_FirstName VARCHAR(20), Student_L...	0 row(s) affected	0.0071 sec
2	18:19:13	ALTER TABLE student ALTER Student_Email_ID SET DEFAULT 'anuja.k8@gmail.com'	0 row(s) affected R...	0.0075 sec
3	18:19:18	DESC student	5 row(s) returned	0.0018 sec / 0.00001...

7. CREATE INDEX

CREATE INDEX constraint is used to create an index on the table. Indexes are not visible to the user, but they help the user to speed up the searching speed or retrieval of data from the database.

Syntax to create an index on single column:

1. **CREATE INDEX** IndexName **ON** TableName (ColumnName 1);

Example:

Create an index on the student table and apply the default constraint while creating a table.

1. mysql> **CREATE INDEX** idx_StudentID **ON** student (StudentID);

```

4 •   CREATE TABLE student
5   (
6     StudentID INT,
7     Student_FirstName VARCHAR(20),
8     Student_LastName VARCHAR(20),
9     Student_PhoneNumber VARCHAR(20),
10    Student_Email_ID VARCHAR(40)
11  );
12
13 •   CREATE INDEX idx_StudentID ON student (StudentID);

```

To verify that the create index constraint is applied to the student table's column, we will execute the following query:

2. mysql> **DESC** student;

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' tab, there is a tree view of databases and tables. The 'Sample' database is selected, and within it, the 'Tables' node is expanded, showing the 'student' table. The main workspace displays the following SQL code:

```

3
4 •   CREATE TABLE student
5   (
6     StudentID INT,
7     Student_FirstName VARCHAR(20),
8     Student_LastName VARCHAR(20),
9     Student_PhoneNumber VARCHAR(20),
10    Student_Email_ID VARCHAR(40)
11  );
12
13 •   CREATE INDEX idx_StudentID ON student (StudentID);
14
15 •   DESC student;
16

```

Below the code, the 'Result Grid' shows the description of the 'student' table:

Field	Type	Null	Key	Default	Extra
StudentID	int	YES	MUL		HULL
Student_FirstName	varchar(20)	YES			HULL
Student_LastName	varchar(20)	YES			HULL
Student_PhoneNumber	varchar(20)	YES			HULL
Student_Email_ID	varchar(40)	YES			anuja.k8@gmail.com

The 'Action Output' section at the bottom shows two log entries:

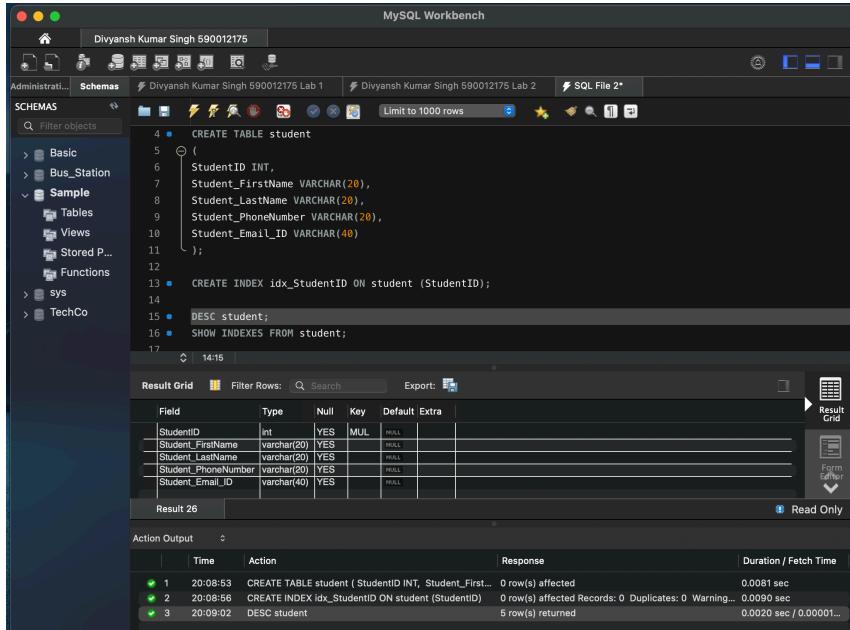
Action	Time	Response	Duration / Fetch Time
CREATE INDEX idx_StudentID ON student (StudentID)	19:14:53	0 row(s) affected R...	0.0060 sec
DESC student	19:20:42	5 row(s) returned	0.0018 sec / 0.00001...

Syntax to create an index on multiple columns:

1. **CREATE INDEX IndexName ON TableName (ColumnName 1, ColumnName 2, ColumnName N);**

Example:

1. mysql> **CREATE INDEX idx_Student ON student (StudentID, Student_PhoneNumber);**



The screenshot shows the MySQL Workbench interface with the following details:

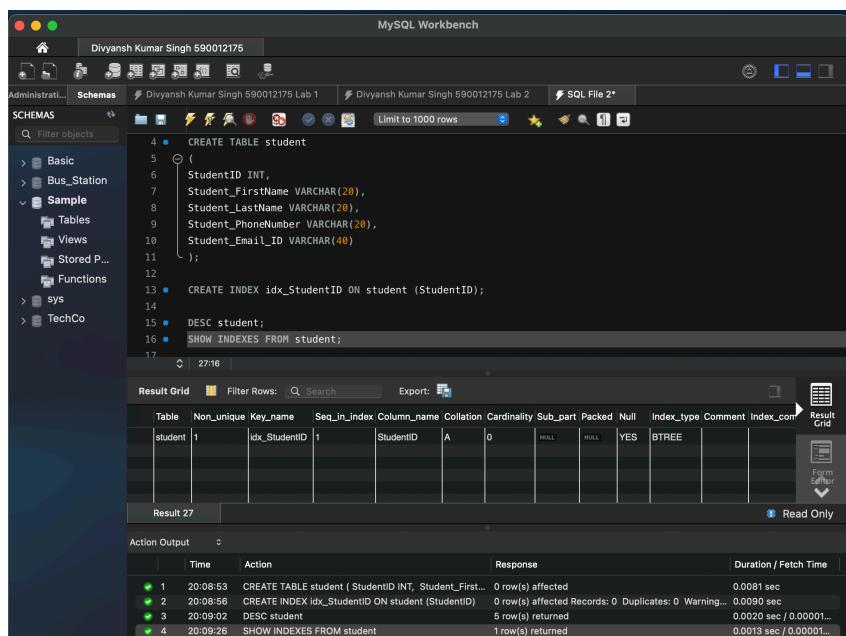
- SQL Editor:** Contains the following SQL code:


```

4 CREATE TABLE student
5 (
6   StudentID INT,
7   Student_FirstName VARCHAR(20),
8   Student_LastName VARCHAR(20),
9   Student_PhoneNumber VARCHAR(20),
10  Student_Email_ID VARCHAR(40)
11 );
12
13 CREATE INDEX idx_StudentID ON student (StudentID);
14
15 DESC student;
16 SHOW INDEXES FROM student;
17 
```
- Result Grid:** Displays the structure of the student table and the newly created index idx_StudentID.
- Action Output:** Shows the execution log with three entries corresponding to the SQL statements run.

To verify that the create index constraint is applied to the student table's column, we will execute the following query:

2. mysql> **DESC student;**



The screenshot shows the MySQL Workbench interface with the following details:

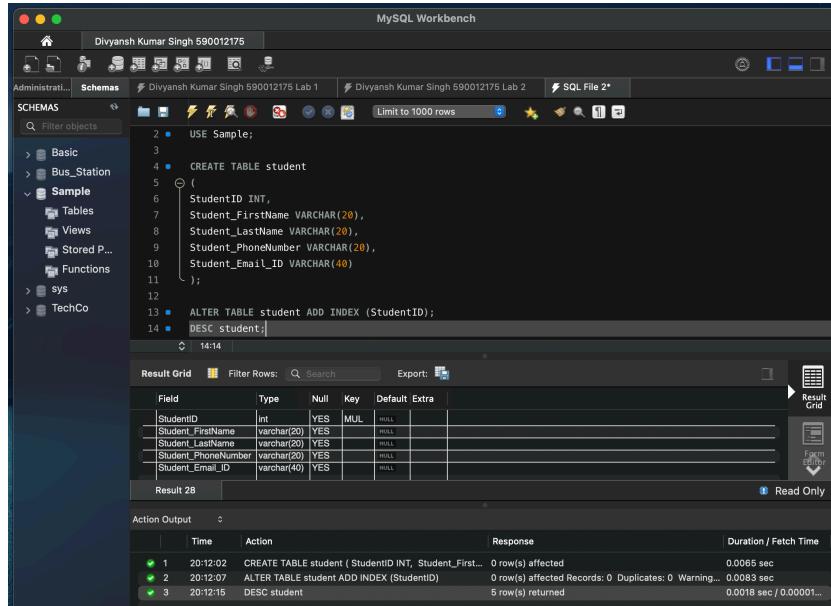
- SQL Editor:** Contains the same SQL code as the previous screenshot, followed by the DESCRIBE command.
- Result Grid:** Displays the description of the student table, including the newly created index idx_StudentID.
- Action Output:** Shows the execution log with four entries corresponding to the SQL statements run.

Syntax to create an index on an existing table:

1. **ALTER TABLE TableName ADD INDEX (ColumnName);**

Consider we have an existing table student. Later, we decided to apply the DEFAULT constraint on the student table's column. Then we will execute the following query:

1. mysql> **ALTER TABLE student ADD INDEX (StudentID);**



The screenshot shows the MySQL Workbench interface with the following details:

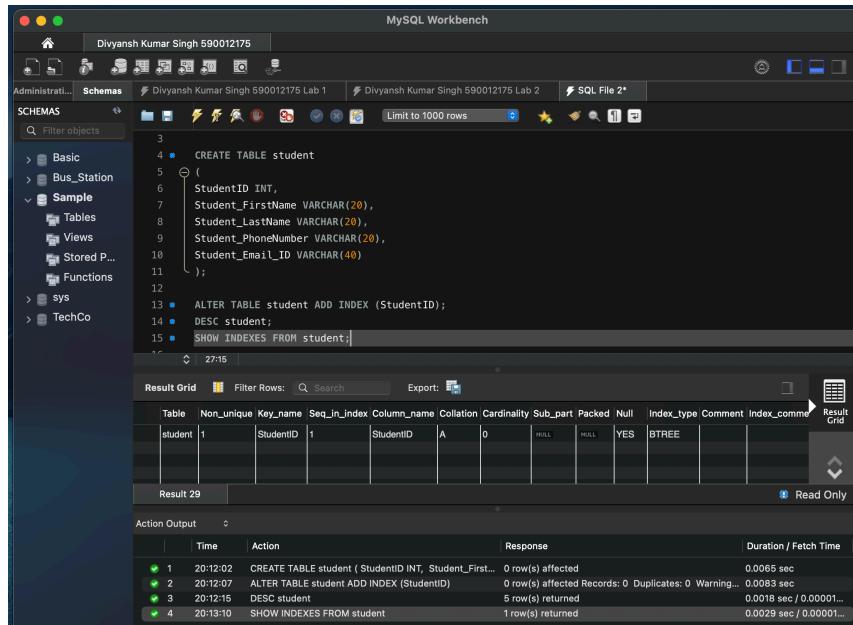
- SQL Editor:** Contains the following SQL code:


```

2 USE Sample;
3
4 CREATE TABLE student (
5   (
6     StudentID INT,
7     Student_FirstName VARCHAR(20),
8     Student_LastName VARCHAR(20),
9     Student_PhoneNumber VARCHAR(20),
10    Student_Email_ID VARCHAR(40)
11  );
12
13 ALTER TABLE student ADD INDEX (StudentID);
14 DESC student;
      
```
- Result Grid:** Shows the structure of the student table with columns: Field, Type, Null, Key, Default, Extra. The StudentID column is defined as INT, Null YES, and Key MUL.
- Action Output:** Displays the execution log with three entries corresponding to the CREATE TABLE, ALTER TABLE, and DESC statements.

To verify that the create index constraint is applied to the student table's column, we will execute the following query:

2. mysql> **DESC student;**



The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:


```

3
4 CREATE TABLE student (
5   (
6     StudentID INT,
7     Student_FirstName VARCHAR(20),
8     Student_LastName VARCHAR(20),
9     Student_PhoneNumber VARCHAR(20),
10    Student_Email_ID VARCHAR(40)
11  );
12
13 ALTER TABLE student ADD INDEX (StudentID);
14 DESC student;
15 SHOW INDEXES FROM student;
      
```
- Result Grid:** Shows the index information for the student table, indicating a unique index named 'student' on the 'StudentID' column.
- Action Output:** Displays the execution log with five entries corresponding to the CREATE TABLE, ALTER TABLE, DESC, and SHOW INDEXES statements.

LAB Performance Questions

You are a junior database administrator at a company called TechCo. The company manages information about its products, customers, and orders through a relational database. Your task is to perform various SQL queries to retrieve and manipulate data within this database.

1. Write an SQL script that includes DDL statements to create the following tables:

- `Products` with columns: `ProductID` (integer), `ProductName` (varchar), `UnitPrice` (decimal), `Manufacturer` (varchar).
- - `Customers` with columns: `CustomerID` (integer), `CustomerName` (varchar), `Email` (varchar).

Ensure appropriate constraints for data integrity.

The screenshot shows the MySQL Workbench interface with a dark theme. The left sidebar displays the 'Schemas' tree, which includes 'Basic', 'Bus_Station' (selected), 'sys', and 'TechCo'. The 'TechCo' schema contains 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The main pane shows an SQL editor with the following code:

```
1 CREATE DATABASE TechCo;
2
3 • USE TechCo;
4
5 -- Question 1
6 -- Create Products table
7 • ⊖ CREATE TABLE Products (
8     ProductID INT NOT NULL,
9     ProductName VARCHAR(50) NOT NULL,
10    UnitPrice DECIMAL(10,2) NOT NULL,
11    Manufacturer VARCHAR(50) NOT NULL
12 );
13 • DESC Products;
14
15 -- Create Customers table
16 • ⊖ CREATE TABLE Customers (
17     CustomerID INT NOT NULL,
18     CustomerName VARCHAR(50) NOT NULL,
19     Email VARCHAR(100) NOT NULL
20 );
21 • DESC Customers;
```

The 'Action Output' pane at the bottom shows the execution results:

Action	Time	Action	Response	Duration / Fetch Time
1	15:25:46	CREA...	1 row(s) affected	0.0018 sec
2	15:25:49	USE...	0 row(s) affected	0.00082 sec
3	15:25:55	CREA...	0 row(s) affected	0.0072 sec
4	15:25:59	CREA...	0 row(s) affected	0.0069 sec

The status bar at the bottom indicates 'Query Completed'.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The left sidebar shows the current schema is "TechCo".
- SQL Editor:** The main area contains the following SQL code:

```
1 CREATE DATABASE TechCo;
2
3 • USE TechCo;
4
5 -- Question 1
6 -- Create Products table
7 • CREATE TABLE Products (
8     ProductID INT NOT NULL,
9     ProductName VARCHAR(50) NOT NULL,
10    UnitPrice DECIMAL(10,2) NOT NULL,
11    Manufacturer VARCHAR(50) NOT NULL
12 );
13 • DESC Products;
14
```
- Result Grid:** Below the SQL editor, the result grid shows the structure of the "Products" table:

Field	Type	Null	Key	Default	Extra
ProductID	int	NO		NULL	
ProductName	varchar(50)	NO		NULL	
UnitPrice	decimal(10,2)	NO		NULL	
Manufacturer	varchar(50)	NO		NULL	

Result 29 | Read Only
- Action Output:** A log of actions and their durations:

Time	Action	Response	Duration / Fetch Time
15:25:46	CREA...	1 row(s) affected	0.0018 sec
15:25:49	USE...	0 row(s) affected	0.00082 sec
15:25:55	CREA...	0 row(s) affected	0.0072 sec
15:25:59	CREA...	0 row(s) affected	0.0069 sec
15:26:58	DESC...	4 row(s) returned	0.0019 sec / 0.0001...
- Status:** At the bottom left, it says "Query Completed".

```

10     UnitPrice DECIMAL(10,2) NOT NULL,
11     Manufacturer VARCHAR(50) NOT NULL
12   );
13 • DESC Products;
14
15  -- Create Customers table
16 • ⊖ CREATE TABLE Customers (
17   CustomerID INT NOT NULL,
18   CustomerName VARCHAR(50) NOT NULL,
19   Email VARCHAR(100) NOT NULL
20 );
21 • DESC Customers;
22 A
23 -- Question 2

```

Result Grid

Field	Type	Null	Key	Default	Extra
CustomerID	int	NO		NULL	
CustomerName	varchar(50)	NO		NULL	
Email	varchar(100)	NO		NULL	

Result 30 Read Only

Action Output

	Time	Action	Response	Duration / Fetch Time
✓ 1	15:25:46	CREA...	1 row(s) affected	0.0018 sec
✓ 2	15:25:49	USE...	0 row(s) affected	0.00082 sec
✓ 3	15:25:55	CREA...	0 row(s) affected	0.0072 sec
✓ 4	15:25:59	CREA...	0 row(s) affected	0.0069 sec
✓ 5	15:26:58	DES...	4 row(s) returned	0.0019 sec / 0.00001...
✓ 6	15:27:27	DES...	3 row(s) returned	0.0020 sec / 0.00000...

Query Completed

- **CREATE TABLE** → Creates a new table.
- **INTEGER** → Stores whole numbers.
- **VARCHAR(n)** → Stores variable-length text up to n characters.
- **DECIMAL** → Stores fixed-precision numbers (like prices).
- **PRIMARY KEY** → Uniquely identifies each row, cannot be NULL.

2. Insert at least three sample records into the 'Products' table created in Question 1. Ensure each record has a unique product ID.

Insert at least three sample records into the 'Customers' table created in Question 1. Each record should have a unique customer ID.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The left sidebar shows the database structure with schemas like Basic, Bus_Station, sys, and TechCo.
- SQL Editor:** The main area contains the following SQL code:

```
-- Question 2
-- Inserting records into Products
INSERT INTO Products (ProductID, ProductName, UnitPrice, Manufacturer)
VALUES
    (1, 'Laptop', 999.99, 'TechBrand'),
    (2, 'Smartphone', 699.99, 'MobileTech'),
    (3, 'Tablet', 349.99, 'GadgetCorp');

SELECT * FROM Products;

-- Inserting records into Customers
INSERT INTO Customers (CustomerID, CustomerName, Email)
VALUES
    (101, 'Divyansh', 'divyansh@email.com');
```
- Result Grid:** Below the editor, the results of the SELECT query are displayed in a grid:

ProductID	ProductName	UnitPrice	Manufacturer
1	Laptop	999.99	TechBrand
2	Smartphone	699.99	MobileTech
3	Tablet	349.99	GadgetCorp
- Action Output:** At the bottom, a table shows the execution history with columns: Time, Action, Response, and Duration / Fetch Time.

	Time	Action	Response	Duration / Fetch Time
1	15:25:46	CREA...	1 row(s) affected	0.0018 sec
2	15:25:49	USE...	0 row(s) affected	0.00082 sec
3	15:25:55	CREA...	0 row(s) affected	0.0072 sec
4	15:25:59	CREA...	0 row(s) affected	0.0069 sec
5	15:26:58	DES...	4 row(s) returned	0.0019 sec / 0.00001...
6	15:27:27	DES...	3 row(s) returned	0.0020 sec / 0.0000...
7	15:27:42	INSE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0023 sec
8	15:27:42	SELE...	3 row(s) returned	0.00044 sec / 0.000...
- Status:** The bottom status bar indicates "Query Completed".

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, which includes 'Basic', 'Bus_Station' (selected), 'sys', and 'TechCo'. The main area contains the following SQL code:

```
29      (3, 'Tablet', 349.99, 'GadgetCorp');
30
31 •  SELECT * FROM Products;
32
33 -- Inserting records into Customers
34 •  INSERT INTO Customers (CustomerID, CustomerName, Email)
35   VALUES
36     (101, 'Divyansh', 'divyansh@email.com'),
37     (102, 'Sauhard', 'sauhard@email.com'),
38     (103, 'Sarthak', 'Sarthak@email.com');
39
40 •  SELECT * FROM Customers;
41
42 -- Question 3
```

The 'Result Grid' tab is selected, showing the data inserted into the 'Customers' table:

CustomerID	CustomerName	Email
101	Divyansh	divyansh@email.com
102	Sauhard	sauhard@email.com
103	Sarthak	Sarthak@email.com

The 'Action Output' tab shows the execution log:

Action	Time	Response	Duration / Fetch Time
CREA...	15:25:46	1 row(s) affected	0.0018 sec
USE...	15:25:49	0 row(s) affected	0.00082 sec
CREA...	15:25:55	0 row(s) affected	0.0072 sec
CREA...	15:25:59	0 row(s) affected	0.0069 sec
DES...	15:26:58	4 row(s) returned	0.0019 sec / 0.00001...
DES...	15:27:27	3 row(s) returned	0.0020 sec / 0.0000...
INSE...	15:27:42	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0023 sec
SELE...	15:27:42	3 row(s) returned	0.00044 sec / 0.000...
INSE...	15:27:51	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0020 sec
SELE...	15:27:51	3 row(s) returned	0.00040 sec / 0.000...

At the bottom, a message says 'Query Completed'.

- **INSERT INTO** → Adds new rows to a table.
- **VALUES** → Specifies the data to insert into the columns.

3. Add a column named 'PhoneNumber' to the 'Customers' table. Apply the NOT NULL constraint to ensure that this information is always provided.

The screenshot shows the MySQL Workbench interface. In the left sidebar under 'Schemas', the 'TechCo' schema is selected. In the main SQL editor area, the following SQL code is written:

```

36      (101, 'Divyansh', 'divyansh@email.com'),
37      (102, 'Sauhard', 'sauhard@email.com'),
38      (103, 'Sarthak', 'Sarthak@email.com');
39
40 •  SELECT * FROM Customers;
41
42 -- Question 3
43 •  ALTER TABLE Customers
44 ADD PhoneNumber VARCHAR(15) NOT NULL DEFAULT '0000000000'
45 CHECK (LENGTH(PhoneNumber) = 10);
46
47 •  DESC Customers;
48
49 -- Question 4

```

The 'DESC Customers;' command is highlighted. Below the SQL editor, the 'Result Grid' shows the structure of the 'Customers' table:

Field	Type	Null	Key	Default	Extra
CustomerID	int	NO			
CustomerName	varchar(50)	NO			
Email	varchar(100)	NO			
PhoneNumber	varchar(15)	NO		0000000000	

The 'Action Output' section at the bottom lists 12 database operations with their times, actions, responses, and durations.

- **ALTER TABLE** → Modifies an existing table.
- **ADD COLUMN** → Introduces a new column.
- **NOT NULL** → Ensures the column must always have a value.

4. Implement a UNIQUE constraint on the 'Email' column in the 'Customers' table. Provide an example of how this constraint prevents data duplication.

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:

```
47 • DESC Customers;
48
49 -- Question 4
50 • ALTER TABLE Customers
51   ADD CONSTRAINT unique_email UNIQUE (Email);
52
53 • DESC Customers;
54 A
55 -- This will fail due to duplicate email:
56 • INSERT INTO Customers (CustomerID, CustomerName, Email, PhoneNumber)
57   VALUES (104, 'Test User', 'divyansh@email.com', '1234567890');
58
59 -- Question 5
60 • ALTER TABLE Products
```
- Result Grid:** Shows the structure of the 'Customers' table:

Field	Type	Null	Key	Default	Extra
CustomerID	int	NO		HULL	
CustomerName	varchar(50)	NO		HULL	
Email	varchar(100)	NO	PRI	HULL	
PhoneNumber	varchar(15)	NO		0000000000	
- Action Output:** Displays the execution history of the queries:

Time	Action	Response	Duration / Fetch Time
15:25:59	CREATE...	0 row(s) affected	0.0009 sec
15:26:58	DESC...	4 row(s) returned	0.0019 sec / 0.00001...
15:27:27	DESC...	3 row(s) returned	0.0020 sec / 0.00000...
15:27:42	INSE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0023 sec
15:27:42	SELE...	3 row(s) returned	0.00044 sec / 0.000...
15:27:51	INSE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0020 sec
15:27:51	SELE...	3 row(s) returned	0.00040 sec / 0.000...
15:28:08	ALTE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.017 sec
15:28:17	DESC...	4 row(s) returned	0.0017 sec / 0.00001...
15:31:08	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
15:31:15	DESC...	4 row(s) returned	0.0018 sec / 0.00001...
15:31:27	DESC...	4 row(s) returned	0.0011 sec / 0.00001...
- Status Bar:** Shows "Query Completed".

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree, which includes 'Basic', 'Bus_Station', 'Tables', 'Views', 'Stored Procedures', 'Functions', 'sys', and 'TechCo'. The 'TechCo' schema is expanded, showing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The main area contains a SQL editor with the following code:

```
47 • DESC Customers;
48
49 -- Question 4
50 • ALTER TABLE Customers
51   ADD CONSTRAINT unique_email UNIQUE (Email);
52
53 • DESC Customers;
54
55 -- This will fail due to duplicate email:
56 • INSERT INTO Customers (CustomerID, CustomerName, Email, PhoneNumber)
57   VALUES (104, 'Test User', 'divyansh@email.com', '1234567890');
58 A
59 -- Question 5
60 • ALTER TABLE Products
61   ADD PRIMARY KEY (ProductID);
62
63 • ALTER TABLE Customers
64   ADD PRIMARY KEY (CustomerID);
65
66 • DESC Products;
67 • DESC Customers;
```

The SQL editor has a status bar at the bottom showing '130%' zoom and '1:56' time. Below the editor is an 'Action Output' table showing the results of the executed statements:

	Time	Action	Response	Duration / Fetch Time
✓ 5	15:26:58	DESC...	4 row(s) returned	0.0019 sec / 0.00001...
✓ 6	15:27:27	DESC...	3 row(s) returned	0.0020 sec / 0.00000...
✓ 7	15:27:42	INSE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0023 sec
✓ 8	15:27:42	SELE...	3 row(s) returned	0.00044 sec / 0.000...
✓ 9	15:27:51	INSE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0020 sec
✓ 10	15:27:51	SELE...	3 row(s) returned	0.00040 sec / 0.000...
✓ 11	15:28:08	ALTE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.017 sec
✓ 12	15:28:17	DES...	4 row(s) returned	0.0017 sec / 0.00001...
✓ 13	15:31:08	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
✓ 14	15:31:15	DES...	4 row(s) returned	0.0018 sec / 0.00001...
✓ 15	15:31:27	DES...	4 row(s) returned	0.0011 sec / 0.00001...
✗ 16	15:31:41	INSE...	Error Code: 1062. Duplicate entry 'divyansh@email.com' for key 'customers.unique_email'	0.00075 sec

The status bar at the bottom also shows 'Query interrupted'.

- **UNIQUE** → Ensures no duplicate values exist in the column.

5. Set the 'ProductID' column as the primary key for the 'Products' table. Also, define 'CustomerID' as the primary key for the 'Customers' table.

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:

```
53 • DESC Customers;
54
55 -- This will fail due to duplicate email:
56 • INSERT INTO Customers (CustomerID, CustomerName, Email, PhoneNumber)
57 VALUES (104, 'Test User', 'divyansh@email.com', '1234567890');
58
59 -- Question 5
60 • ALTER TABLE Products
61 ADD PRIMARY KEY (ProductID);
62
63 • ALTER TABLE Customers
64 ADD PRIMARY KEY (CustomerID);
65
66 • DESC Products;
```
- Result Grid:** Displays the structure of the 'Products' table:

Field	Type	Null	Key	Default	Extra
ProductID	int	NO	PRI	NULL	
ProductName	varchar(50)	NO		NULL	
UnitPrice	decimal(10,2)	NO		NULL	
Manufacturer	varchar(50)	NO		NULL	

Result 36 | Read Only
- Action Output:** Shows the execution history with the following log entries:

Time	Action	Response	Duration / Fetch Time
15:27:42	SELE...	3 row(s) returned	0.00044 sec / 0.000...
15:27:51	INSE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0020 sec
15:27:51	SELE...	3 row(s) returned	0.00040 sec / 0.000...
15:28:08	ALTE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.017 sec
15:28:17	DES...	4 row(s) returned	0.0017 sec / 0.00001...
15:31:08	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
15:31:15	DES...	4 row(s) returned	0.0018 sec / 0.00001...
15:31:27	DES...	4 row(s) returned	0.0011 sec / 0.00001...
15:31:41	INSE...	Error Code: 1062. Duplicate entry 'divyansh@email.com' for key 'customers.unique_email'	0.00075 sec
15:33:18	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
15:33:23	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.019 sec
15:33:27	DES...	4 row(s) returned	0.0022 sec / 0.00001...
- Status Bar:** Shows "Query Completed".

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with several schemas listed under 'TechCo'. The main area contains a SQL editor with the following code:

```
55 -- This will fail due to duplicate email:
56 • INSERT INTO Customers (CustomerID, CustomerName, Email, PhoneNumber)
57 VALUES (104, 'Test User', 'divyansh@email.com', '1234567890');
58
59 -- Question 5
60 • ALTER TABLE Products
61 ADD PRIMARY KEY (ProductID);
62
63 • ALTER TABLE Customers
64 ADD PRIMARY KEY (CustomerID);
65
66 • DESC Products;
67 • DESC Customers;
68 A
```

The code includes comments and several statements: an insertion into the 'Customers' table that fails due to a duplicate email address; two 'ALTER TABLE' statements to add primary keys to the 'Products' and 'Customers' tables; and two 'DESC' statements to describe the 'Products' and 'Customers' tables.

Below the SQL editor is a 'Result Grid' showing the structure of the 'Customers' table:

Field	Type	Null	Key	Default	Extra
CustomerID	int	NO	PRI	HULL	
CustomerName	varchar(50)	NO		HULL	
Email	varchar(100)	NO	UNI	HULL	
PhoneNumber	varchar(15)	NO		0000000000	

The 'Result Grid' panel also includes a 'Form Editor' button and a 'Read Only' status indicator.

The 'Action Output' panel at the bottom lists the execution history with details like time, action, response, and duration:

Action	Time	Response	Duration / Fetch Time
INSE...	15:27:51	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0020 sec
SELE...	15:27:51	3 row(s) returned	0.00040 sec / 0.000...
ALTE...	15:28:08	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.017 sec
DES...	15:28:17	4 row(s) returned	0.0017 sec / 0.0001...
ALTE...	15:31:08	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
DES...	15:31:15	4 row(s) returned	0.0018 sec / 0.0001...
DES...	15:31:27	4 row(s) returned	0.0011 sec / 0.0001...
INSE...	15:31:41	Error Code: 1062. Duplicate entry 'divyansh@email.com' for key 'customers.unique_email'	0.00075 sec
SELE...	15:33:18	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
ALTE...	15:33:23	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.019 sec
DES...	15:33:27	4 row(s) returned	0.0022 sec / 0.0001...
DES...	15:33:35	4 row(s) returned	0.0022 sec / 0.0000...

The status bar at the bottom indicates 'Query Completed'.

- **PRIMARY KEY** → Combination of NOT NULL + UNIQUE, ensures entity integrity.

6. Establish a relationship between the 'Orders' table (containing columns 'OrderID' and 'CustomerID') and the 'Customers' table. Ensure referential integrity by using a FOREIGN KEY constraint.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "Sample".
- SQL Editor:** Contains the following SQL code:

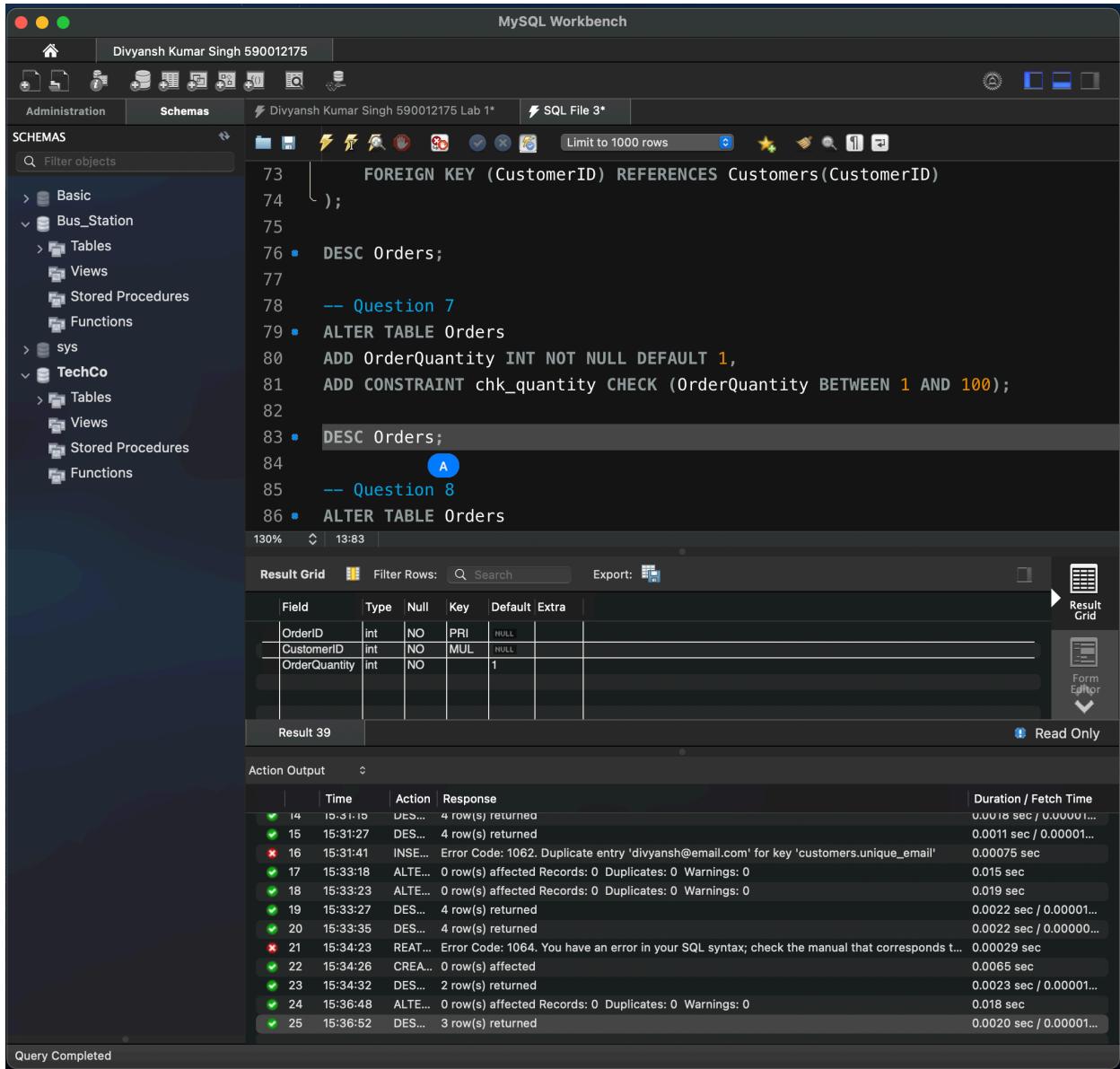
```
66 • DESC Products;
67 • DESC Customers;
68
69 -- Question 6
70 • CREATE TABLE Orders (
    OrderID INT NOT NULL PRIMARY KEY,
    CustomerID INT NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
76 • DESC Orders;
```
- Result Grid:** Shows the structure of the Orders table:

Field	Type	Null	Key	Default	Extra
OrderID	int	NO	PRI	NULL	
CustomerID	int	NO	MUL	NULL	
- Action Output:** Displays the execution results:

Time	Action	Response	Duration / Fetch Time
22:41:59	CREATE TABLE Orders (OrderID INT NOT NULL PRI...	0 row(s) affected	0.010 sec
22:41:59	DESC Orders	2 row(s) returned	0.0017 sec / 0.00000...

- **FOREIGN KEY** → Links a column to a primary key in another table.
- **REFERENCES** → Defines which table/column the foreign key points to.
- **Referential Integrity** → Ensures relationships between tables remain valid.

7. Add a column named 'OrderQuantity' to the 'Orders' table. Apply a CHECK constraint to limit the order quantity to a range between 1 and 100.



```

MySQL Workbench

Administration | Schemas | SQL File 3*
SCHEMAS
    Basic
    Bus_Station
        Tables
        Views
        Stored Procedures
        Functions
    sys
    TechCo
        Tables
        Views
        Stored Procedures
        Functions
130% 13:33

73     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
74 );
75
76 • DESC Orders;
77
78 --- Question 7
79 • ALTER TABLE Orders
80     ADD OrderQuantity INT NOT NULL DEFAULT 1,
81     ADD CONSTRAINT chk_quantity CHECK (OrderQuantity BETWEEN 1 AND 100);
82
83 • DESC Orders;
84 A
85 --- Question 8
86 • ALTER TABLE Orders

```

Result Grid

Field	Type	Null	Key	Default	Extra
OrderID	int	NO	PRI	NULL	
CustomerID	int	NO	MUL	NULL	
OrderQuantity	int	NO		1	

Result 39 Read Only

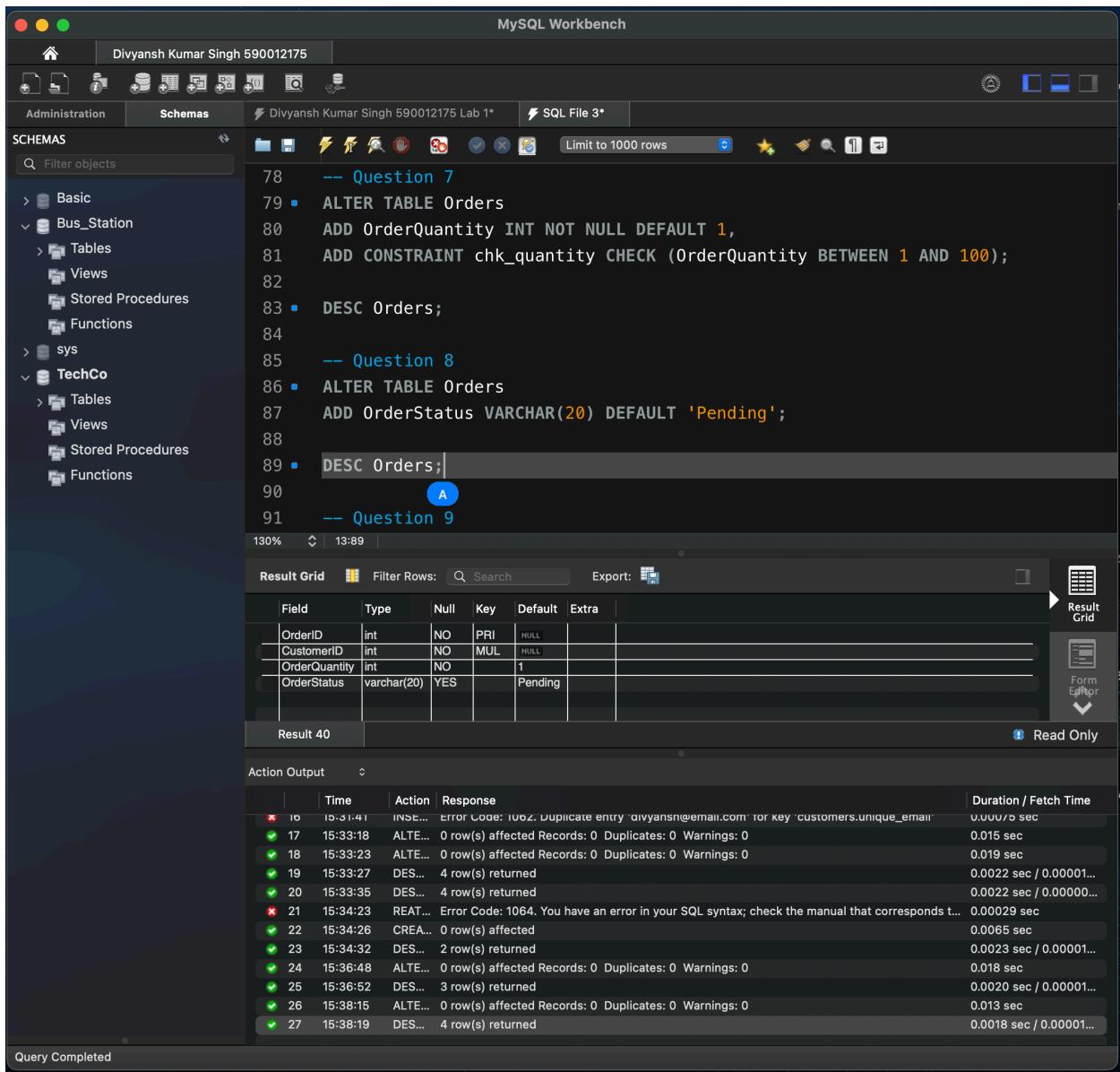
Action Output

Time	Action	Response	Duration / Fetch Time
14 15:31:10	DESC...	4 row(s) returned	0.0018 sec / 0.00001...
15 15:31:27	DESC...	4 row(s) returned	0.0011 sec / 0.00001...
16 15:31:41	INSE...	Error Code: 1062. Duplicate entry 'divyansh@email.com' for key 'customers.unique_email'	0.00075 sec
17 15:33:18	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
18 15:33:23	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.019 sec
19 15:33:27	DES...	4 row(s) returned	0.0022 sec / 0.00001...
20 15:33:35	DES...	4 row(s) returned	0.0022 sec / 0.00000...
21 15:34:23	REAT...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds t...	0.00029 sec
22 15:34:26	CREA...	0 row(s) affected	0.0065 sec
23 15:34:32	DES...	2 row(s) returned	0.0023 sec / 0.00001...
24 15:36:48	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.018 sec
25 15:36:52	DES...	3 row(s) returned	0.0020 sec / 0.00001...

Query Completed

- o **CHECK** → Restricts values in a column based on a condition (e.g., between 1–100).

8. Include a column named 'OrderStatus' in the 'Orders' table. Set a DEFAULT constraint so that if the status is not specified during insertion, it defaults to 'Pending.'



The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:


```

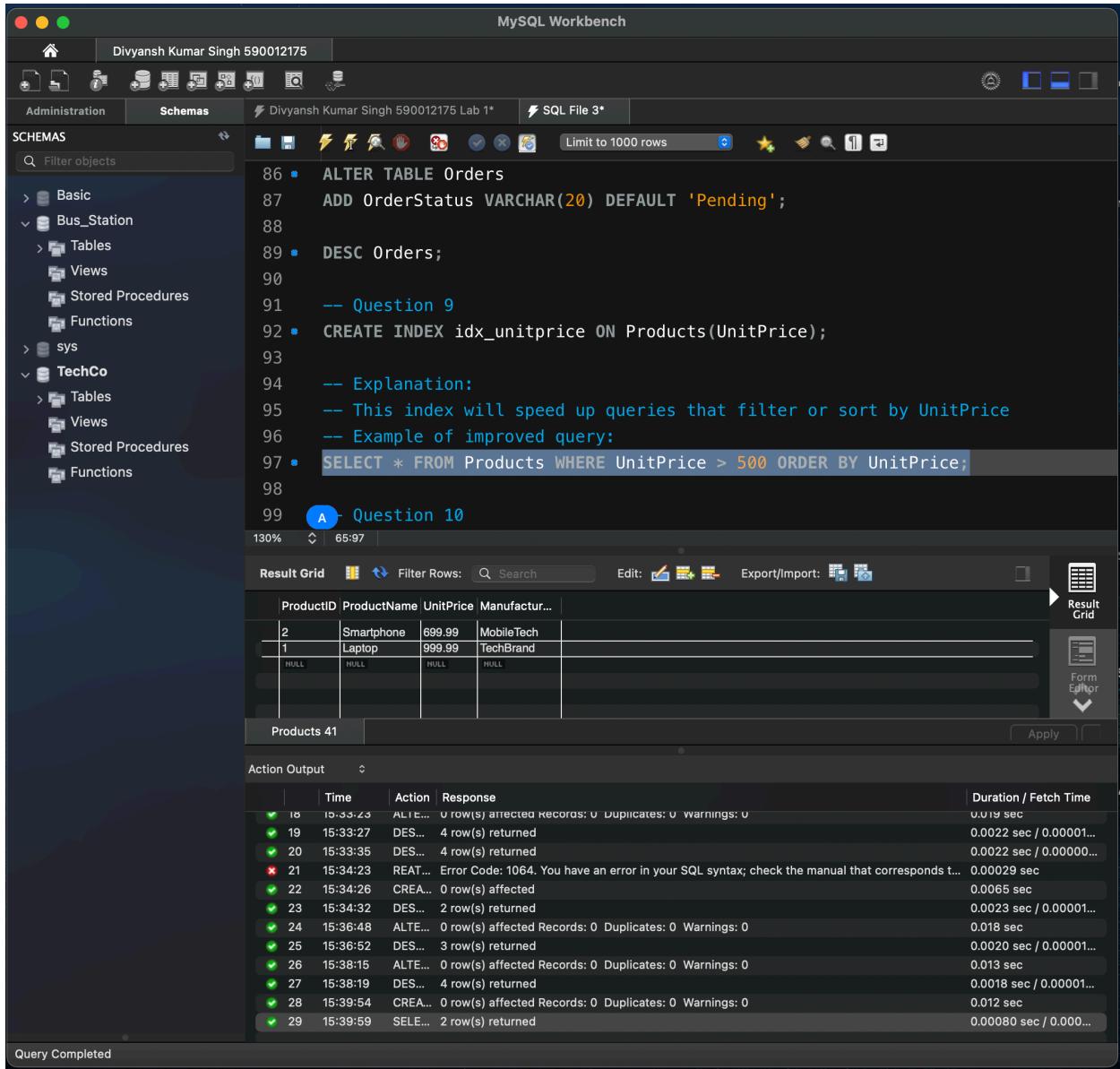
78 -- Question 7
79 • ALTER TABLE Orders
80 ADD OrderQuantity INT NOT NULL DEFAULT 1,
81 ADD CONSTRAINT chk_quantity CHECK (OrderQuantity BETWEEN 1 AND 100);
82
83 • DESC Orders;
84
85 -- Question 8
86 • ALTER TABLE Orders
87 ADD OrderStatus VARCHAR(20) DEFAULT 'Pending';
88
89 • DESC Orders;
90
91 -- Question 9
      
```
- Result Grid:** Shows the structure of the 'Orders' table with the following columns:

Field	Type	Null	Key	Default	Extra
OrderID	int	NO	PRI		HULL
CustomerID	int	NO	MUL		HULL
OrderQuantity	int	NO		1	
OrderStatus	varchar(20)	YES		Pending	
- Action Output:** Displays the execution log with the following entries:

Time	Action	Response	Duration / Fetch Time
15:31:41	INSE...	Error Code: 1062. Duplicate entry 'divyansh@gmail.com' for key 'customers.unique_email'	0.0000/0 sec
15:33:18	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
15:33:23	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.019 sec
15:33:27	DES...	4 row(s) returned	0.0022 sec / 0.00001...
15:33:35	DES...	4 row(s) returned	0.0022 sec / 0.00001...
15:34:23	REAT...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds t...	0.00029 sec
15:34:26	CREA...	0 row(s) affected	0.0065 sec
15:34:32	DES...	2 row(s) returned	0.0023 sec / 0.00001...
15:36:48	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.018 sec
15:36:52	DES...	3 row(s) returned	0.0020 sec / 0.00001...
15:38:15	ALTE...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.013 sec
15:38:19	DES...	4 row(s) returned	0.0018 sec / 0.00001...

- **DEFAULT** → Inserts a preset value when no value is provided (e.g., 'Pending').

9. Create an index on the 'UnitPrice' column in the 'Products' table. Explain how this index can improve query performance.



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The left sidebar shows the current schema is "Divyansh Kumar Singh 590012175".
- SQL Editor:** The main area contains the following SQL code:


```

86 • ALTER TABLE Orders
87     ADD OrderStatus VARCHAR(20) DEFAULT 'Pending';
88
89 • DESC Orders;
90
91 --- Question 9
92 • CREATE INDEX idx_unitprice ON Products(UnitPrice);
93
94 --- Explanation:
95 --- This index will speed up queries that filter or sort by UnitPrice
96 --- Example of improved query:
97 • SELECT * FROM Products WHERE UnitPrice > 500 ORDER BY UnitPrice;
98
99 A - Question 10
      
```
- Result Grid:** Below the SQL editor, the result grid shows the first two rows of the Products table:

ProductID	ProductName	UnitPrice	Manufactur...
2	Smartphone	699.99	MobileTech
1	Laptop	999.99	TechBrand
HULL	HULL	HULL	HULL
- Action Output:** At the bottom, a table lists the actions taken during the session:

	Time	Action	Response	Duration / Fetch Time
✓ 18	15:33:23	ALTER...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.019 sec
✓ 19	15:33:27	DES...	4 row(s) returned	0.0022 sec / 0.00001...
✓ 20	15:33:35	DES...	4 row(s) returned	0.0022 sec / 0.00000...
✗ 21	15:34:23	REAT...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds t...	0.00029 sec
✓ 22	15:34:26	CREA...	0 row(s) affected	0.0065 sec
✓ 23	15:34:32	DES...	2 row(s) returned	0.0023 sec / 0.00001...
✓ 24	15:36:48	ALTER...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.018 sec
✓ 25	15:36:52	DES...	3 row(s) returned	0.0020 sec / 0.00001...
✓ 26	15:38:15	ALTER...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.013 sec
✓ 27	15:38:19	DES...	4 row(s) returned	0.0018 sec / 0.00001...
✓ 28	15:39:54	CREA...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.012 sec
✓ 29	15:39:59	SELE...	2 row(s) returned	0.00080 sec / 0.000...

- **CREATE INDEX** → Improves search and query performance on a column.

10. Write an SQL query to retrieve the names and email addresses of all customers who placed orders.

Join the 'Customers' and 'Orders' tables to achieve this.

The screenshot shows the MySQL Workbench interface. In the center, there is a code editor window displaying the following SQL script:

```
-- Question 10
INSERT INTO Orders (OrderID, CustomerID, OrderQuantity)
VALUES
    (1001, 101, 2),
    (1002, 102, 1),
    (1003, 101, 5);

-- Now the query:
SELECT DISTINCT c.CustomerName, c.Email
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID;
```

The code editor has line numbers from 96 to 110. The status bar at the bottom indicates "100% 46:109". Below the code editor is a "Result Grid" table with two rows:

CustomerName	Email
Divyansh	divyansh@email.com
Sauhard	sauhard@email.com

At the bottom of the interface, there is an "Action Output" section showing the results of the executed queries:

Action	Time	Response	Duration / Fetch Time
1	22:50:50	INSERT INTO Orders (OrderID, CustomerID, OrderQuantity) VALUES (1001, 101, 2), (1002, 102, 1), (1003, 101, 5); 3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0031 sec
2	22:50:55	SELECT DISTINCT c.CustomerName, c.Email FROM Customers c JOIN Orders o ON c.CustomerID = o.CustomerID; 2 row(s) returned	0.0015 sec / 0.00001...

- **SELECT** → Retrieves data from tables.
- **JOIN** → Combines rows from multiple tables based on related columns.
- **ON** → Defines the join condition.

11. Modify the 'OrderStatus' of all orders with a quantity greater than 50 to 'Shipped.' Provide the SQL statement for this update.

The screenshot shows the MySQL Workbench interface with the following details:

- Title Bar:** MySQL Workbench, Divyansh Kumar Singh 590012175
- Schemas Tab:** Shows the current schema is 'Sample'. The left sidebar lists other schemas like 'Basic', 'Bus_Station', 'sys', and 'TechCo' with their respective tables, views, stored procedures, and functions.
- SQL Editor:** Contains the following SQL code:

```
108 FROM Customers c
109 JOIN Orders o ON c.CustomerID = o.CustomerID;
110
111 -- Question 11
112 -- First verify which orders will be updated
113 • SELECT OrderID, CustomerID, OrderStatus
114   FROM Orders
115 WHERE OrderQuantity > 50;
116
117 -- Then perform the update using the primary key
118 • UPDATE Orders o
119   JOIN (
120     SELECT OrderID
121       FROM Orders
122      WHERE OrderQuantity > 50
123    ) t ON o.OrderID = t.OrderID
124   SET o.OrderStatus = 'Shipped';
```
- Result Grid:** Displays the results of the SELECT query, showing three rows of data with OrderID 1005, 1006, and 1007, all with CustomerID 102 and OrderStatus 'Pending'.
- Action Output:** Shows the history of actions taken, including successful INSERTs, a failed UPDATE attempt (marked with a red X), and successful SELECTs. The log includes entries for each step of the process, such as inserting rows into the Orders table and selecting distinct customer information.

The screenshot shows the MySQL Workbench interface. The top bar displays the title "MySQL Workbench" and the user information "Divyansh Kumar Singh 590012175". The left sidebar shows the "Schemas" tree, which includes the "TechCo" schema where the "Orders" table is located. The main area contains the following SQL code:

```

110
111  -- Question 11
112  -- First verify which orders will be updated
113  •  SELECT OrderID, CustomerID, OrderStatus
114    FROM Orders
115    WHERE OrderQuantity > 50;
116
117  -- Then perform the update using the primary key
118  •  UPDATE Orders o
119  JOIN (
120      SELECT OrderID
121        FROM Orders
122       WHERE OrderQuantity > 50
123   ) t ON o.OrderID = t.OrderID
124   SET o.OrderStatus = 'Shipped';
125
126  •  SELECT * FROM Orders;

```

The "Result Grid" tab shows the results of the last query, displaying the following data:

OrderID	CustomerID	OrderQuantity	OrderStatus
1001	101	2	Pending
1002	102	1	Pending
1003	101	5	Pending
1004	101	2	Pending
1005	102	56	Shipped
1006	101	52	Shipped
NULL	NULL	NULL	NULL

The "Action History" tab at the bottom shows the execution details for each statement:

Action	Time	Response	Duration / Fetch Time
SELECT OrderID, CustomerID, OrderStatus FROM Orders WHERE...	22:51:41	0 row(s) returned	0.00038 sec / 0.000...
UPDATE Orders SET OrderStatus = 'Shipped' WHERE OrderID I...	22:52:05	Error Code: 1093. You can't specify target t...	0.00089 sec
UPDATE Orders o JOIN (SELECT OrderID FROM Orders...	22:52:54	0 row(s) affected Rows matched: 0 Chang...	0.00091 sec
SELECT OrderID, CustomerID, OrderStatus FROM Orders WHERE...	22:53:45	0 row(s) returned	0.00081 sec / 0.000...
INSERT INTO Orders (OrderID, CustomerID, OrderQuantity) VAL...	22:54:08	Error Code: 1062. Duplicate entry '1001' for...	0.00041 sec
INSERT INTO Orders (OrderID, CustomerID, OrderQuantity) VAL...	22:54:17	Error Code: 1062. Duplicate entry '1002' for...	0.0012 sec
INSERT INTO Orders (OrderID, CustomerID, OrderQuantity) VAL...	22:54:29	3 row(s) affected Records: 3 Duplicates: 0...	0.0021 sec
SELECT OrderID, CustomerID, OrderStatus FROM Orders WHERE...	22:54:40	2 row(s) returned	0.00061 sec / 0.000...
UPDATE Orders o JOIN (SELECT OrderID FROM Orders...	22:55:31	2 row(s) affected Rows matched: 2 Change...	0.0022 sec
SELECT * FROM Orders LIMIT 0, 1000	22:55:50	6 row(s) returned	0.00072 sec / 0.000...

- **UPDATE** → Modifies existing rows in a table.
- **SET** → Assigns new values to columns.
- **WHERE** → Applies a condition to restrict which rows are updated.

12. Delete all records from the 'Products' table where the 'UnitPrice' is less than \$10. Include the SQL statement for this deletion.

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:


```

122 |     SELECT OrderID
123 |         FROM Orders
124 |         WHERE OrderQuantity > 50
125 |     ;
126
127 |     -- Question 12
128 •   DELETE FROM Products
129 |         WHERE UnitPrice < 10;
130
131 |     -- Verify:
132 •   SELECT * FROM Products;
133
134 |     -- Question 13
135 •   ALTER TABLE Products
      
```
- Result Grid:** Displays the results of the `SELECT * FROM Products;` query, showing 4 rows of data:

ProductID	ProductName	UnitPrice	Manufactur...
1	Laptop	999.99	TechBrand
2	Smartphone	699.99	MobileTech
3	Tablet	349.99	GadgetCorp
HULL	HULL	HULL	HULL
- Action Output:** Shows the log of database operations with the following entries:

Action	Time	Response	Duration / Fetch Time
ALTER...	15:30:48	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.018 sec
DELETE...	15:36:52	3 row(s) returned	0.0020 sec / 0.0001...
ALTER...	15:38:15	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.013 sec
DELETE...	15:38:19	4 row(s) returned	0.0018 sec / 0.0001...
CREATE...	15:39:54	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.012 sec
SELECT...	15:39:59	2 row(s) returned	0.00080 sec / 0.000...
INSERT...	15:41:42	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.00093 sec
SELECT...	15:41:46	2 row(s) returned	0.0015 sec / 0.0001...
SELECT...	15:43:13	0 row(s) returned	0.00072 sec / 0.000...
UPDATE...	15:43:21	Error Code: 1093. You can't specify target table 'Orders' for update in FROM clause	0.00071 sec
DELETE...	15:44:55	0 row(s) affected	0.00075 sec
SELECT...	15:44:59	3 row(s) returned	0.00067 sec / 0.000...

- **DELETE FROM** → Removes rows from a table.
- **WHERE** → Restricts deletion to matching rows.

13. The 'UnitPrice' column in the 'Products' table needs to be changed to a new datatype, 'float'. Write an SQL statement using the ALTER TABLE statement to make this change.

The screenshot shows the MySQL Workbench interface with the following details:

- Title Bar:** MySQL Workbench, Divyansh Kumar Singh 590012175
- Schemas Tab:** Shows the current database schema structure.
- SQL Editor:** Contains the following SQL code:


```

125 );
126
127 -- Question 12
128 • DELETE FROM Products
129 WHERE UnitPrice < 10;
130
131 -- Verify:
132 • SELECT * FROM Products;
133
134 -- Question 13
135 • ALTER TABLE Products
136 MODIFY COLUMN UnitPrice FLOAT NOT NULL;
137
138 • DESC Products;
      
```
- Result Grid:** Displays the structure of the Products table with the following columns:

Field	Type	Null	Key	Default	Extra
ProductID	int	NO	PRI	NULL	
ProductName	varchar(50)	NO		NULL	
UnitPrice	float	NO	MUL	NULL	
Manufacturer	varchar(50)	NO		NULL	
- Action Output:** Shows the log of actions taken during the session, including the execution of the ALTER TABLE statement and its duration.
- Status Bar:** Query Completed

- **ALTER TABLE** → Changes table structure.
- **MODIFY / ALTER COLUMN** → Alters column definition.
- **FLOAT** → Data type for approximate decimal values.

14. Write an SQL query to calculate the average 'UnitPrice' of all products in the 'Products' table.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree, which includes 'Basic', 'Bus_Station' (selected), 'sys', and 'TechCo'. The main pane contains the following SQL code:

```
131 -- Verify:  
132 • SELECT * FROM Products;  
133  
134 -- Question 13  
135 • ALTER TABLE Products  
136 MODIFY COLUMN UnitPrice FLOAT NOT NULL;  
137  
138 • DESC Products;  
139  
140 -- Question 14  
141 • SELECT AVG(UnitPrice) AS AveragePrice  
142 FROM Products;  
143
```

The 'Result Grid' tab shows the output of the last query:

AveragePrice
683.3233235677084

The 'Action Output' tab shows the execution log:

Time	Action	Response	Duration / Fetch Time
27/15:38:19	DESC...	4 row(s) returned	0.0018 sec / 0.0001...
28 15:39:54	CREA...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.012 sec
29 15:39:59	SELE...	2 row(s) returned	0.00080 sec / 0.000...
30 15:41:42	INSE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.00093 sec
31 15:41:46	SELE...	2 row(s) returned	0.0015 sec / 0.00001...
32 15:43:13	SELE...	0 row(s) returned	0.00072 sec / 0.0000...
33 15:43:21	UPD...	Error Code: 1093. You can't specify target table 'Orders' for update in FROM clause	0.00071 sec
34 15:44:55	DELE...	0 row(s) affected	0.00075 sec
35 15:44:59	SELE...	3 row(s) returned	0.00067 sec / 0.0000...
36 15:45:39	ALTE...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.016 sec
37 15:45:50	DES...	4 row(s) returned	0.0019 sec / 0.00001...
38 15:46:43	SELE...	1 row(s) returned	0.00070 sec / 0.0000...

At the bottom, a message says 'Query Completed'.

- **SELECT AVG()** → Aggregate function to calculate the average of a column.
- **FROM** → Specifies the table to query.

CONCLUSION

This DBMS lab helped me understand the importance of SQL constraints in ensuring data accuracy and integrity. I practiced using constraints like **NOT NULL**, **UNIQUE**, **PRIMARY KEY**, **FOREIGN KEY**, **CHECK**, **DEFAULT**, and **INDEX** to manage and validate data effectively.

Along with this, I worked on **DDL and DML queries** to create, modify, and manipulate tables. Overall, the lab improved my practical knowledge of database design and strengthened my ability to build reliable and consistent databases.