AROR UNIVERSITY OF ART, ARCHITECTURE, DESIGN & HERITAGE, SUKKUR, SINDH

Department of Artificial Intelligence & Multimedia Gamming

CSC-207: Database Systems

Lab # 08: To Work with MySQL Workbench & DDL Queries

Objectives

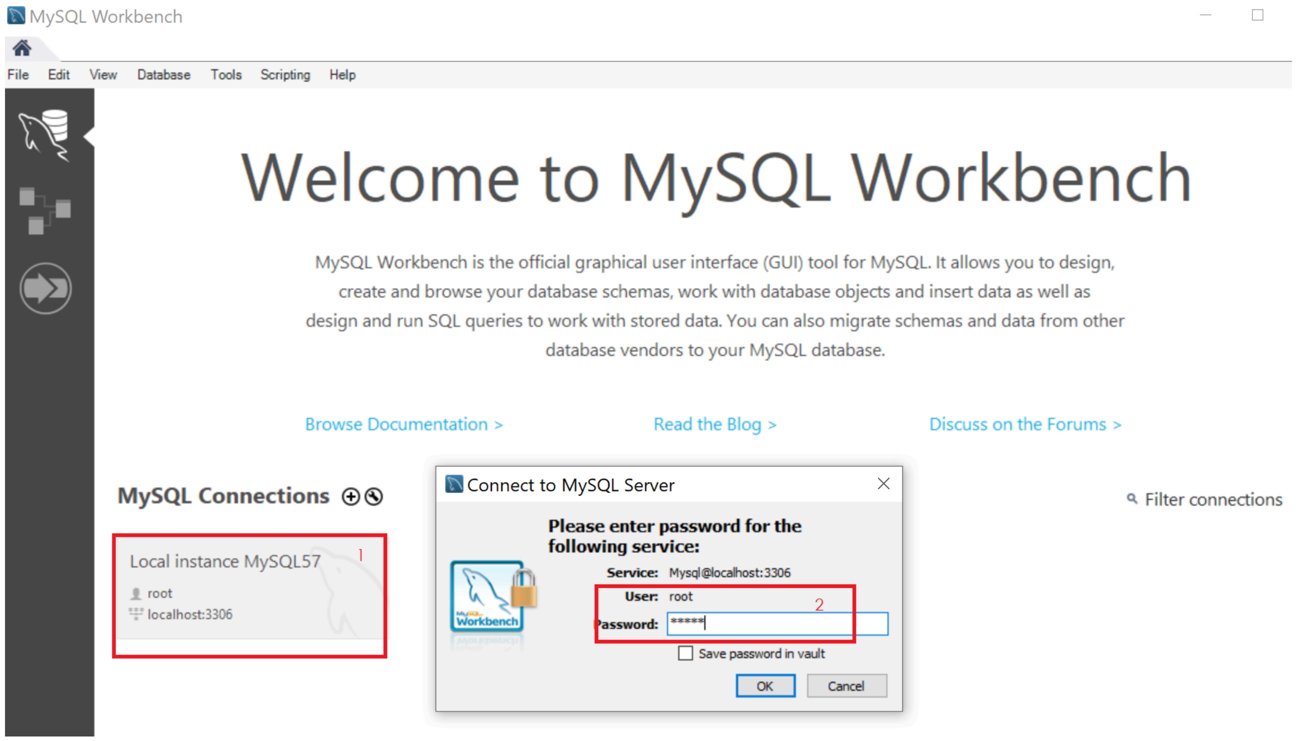
1. MySQL Workbench Introduction
2. MySQL Workbench Getting Started
3. DDL Queries in MySQL Workbench
4. Create Command
5. Alter Command
6. Drop Command
7. Truncate Command
8. Rename Command
9. Comments in SQL
10. Primary Key & Foreign Key Concept

MySQL Workbench Introduction

MySQL Workbench is a powerful tool for database architects, developers, and DBAs. It provides a graphical user interface to the MySQL server environment where users can manage databases, run queries, and set up servers.

MySQL Workbench Getting Started

- Launch MySQL



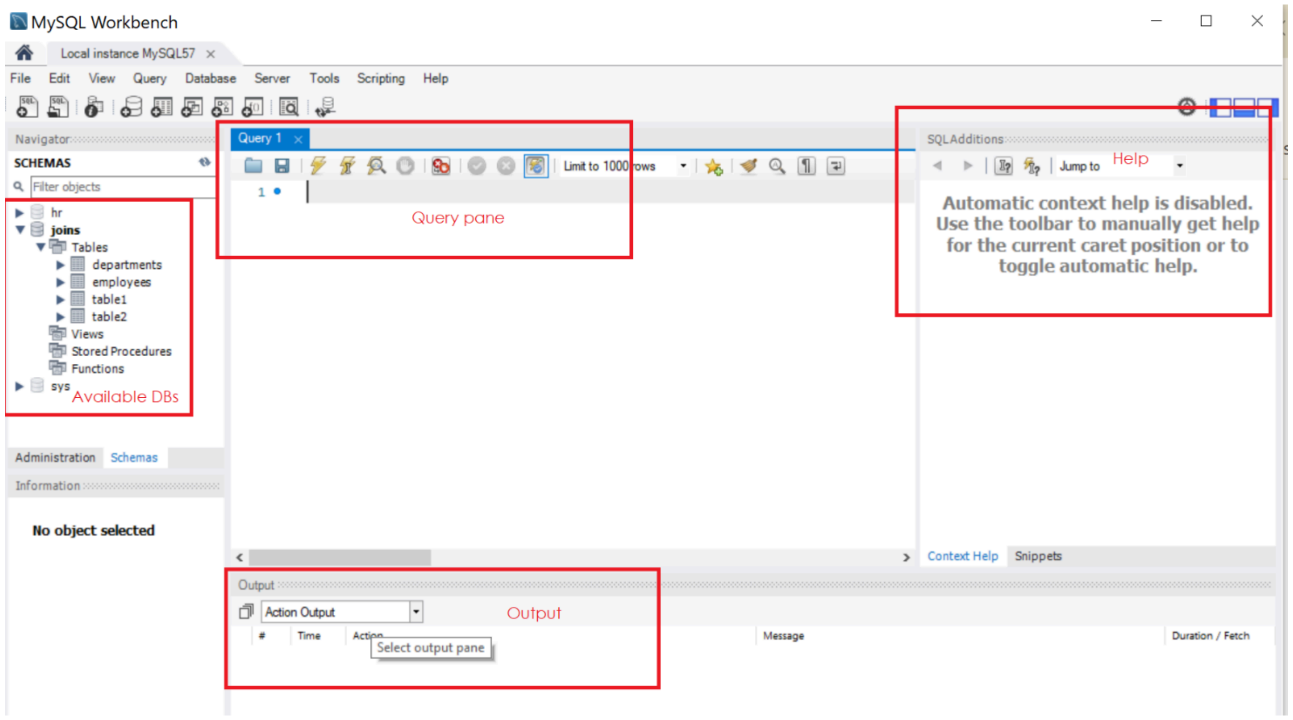
Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Please enter password for the following service:

Service: Mysql@localhost:3306
User: root
Password: *****
Save password in vault

- Explore the visual interface, you have a query pane to execute a query , output pane where output of query is displayed, is the left side you have panel for available databases under schemas. And in right most corner you have help.



No object selected

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

DDL Queries in MySQL Workbench

- DDL is short name of Data Definition Language, defines the database structure or database schema.
- DDL also defines additional properties of the data defined in the database, as the domain of the attributes.
- The Data Definition Language also provide the facility to specify some constraints that would maintain the data consistency.

Following are the common SQL commands:

CREATE - to create a database and its objects like (table, index, views, store procedure, function, and triggers).

ALTER - alters the structure of the existing database.

DROP - delete objects from the database.

TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed.

RENAME - rename an object.

Note:

DDL statements automatically commit the current transaction; they cannot be rolled back.

1. CREATE Command

This command is used to create database & its objects. There are two common CREATE statements available in SQL:

1. **CREATE DATABASE**
2. **CREATE TABLE**

1. CREATE DATABASE

A Database is defined as a structured set of data. So, in SQL the very first step to store the data in a well-structured manner is to create a database. The CREATE DATABASE statement is used to create a new database in SQL.

Syntax:

CREATE DATABASE database_name;

and then to use it follow command *USE database_name;*

2. CREATE TABLE

The CREATE TABLE statement is used to create a table in SQL. We know that a table comprises of rows and columns. So while creating tables we have to provide all the information to SQL about the names of the columns, type of data to be stored in columns, size of the data etc. Let us now dive into details on how to use CREATE TABLE statement to create tables in SQL.

Syntax:

CREATE TABLE table_name

(

column1 data_type(size),

column2 data_type(size),

column3 data_type(size),

. . .

);

Example:

CREATE TABLE patient

(

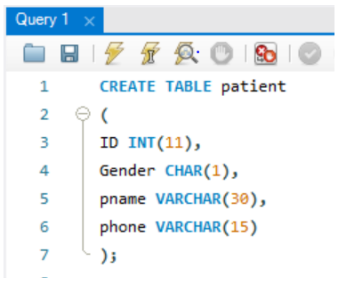
ID INT(11),

Gender CHAR(1),

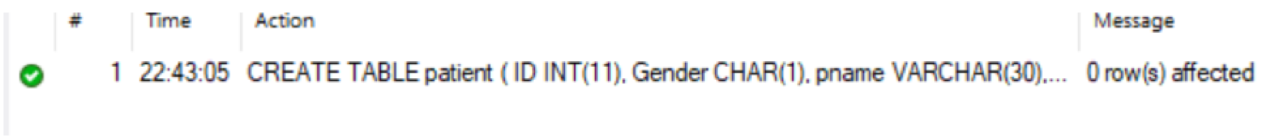
pname VARCHAR(30),

phone VARCHAR(15)

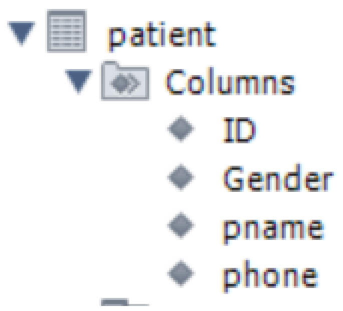
);



```
CREATE TABLE patient
(
ID INT(11),
Gender CHAR(1),
pname VARCHAR(30),
phone VARCHAR(15)
);
```



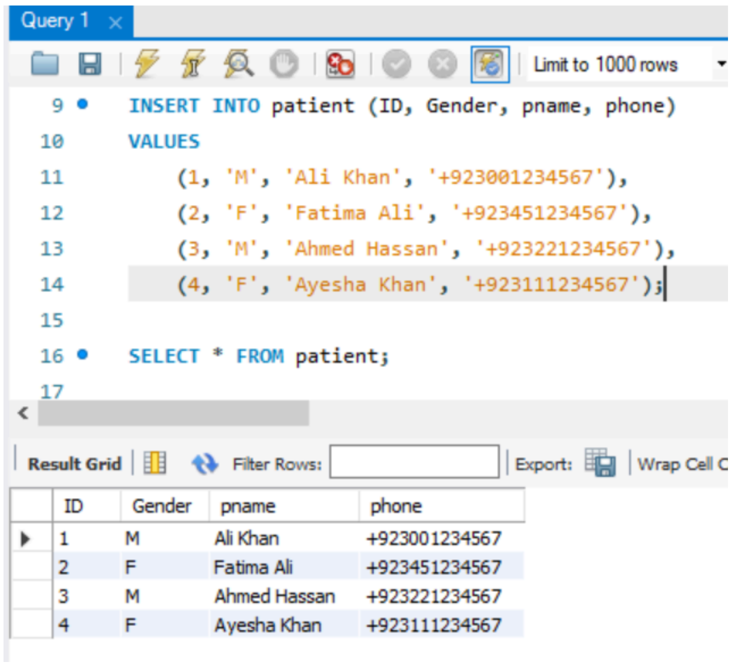
#	Time	Action	Message
1	22:43:05	CREATE TABLE patient (ID INT(11), Gender CHAR(1), pname VARCHAR(30),...	0 row(s) affected



Now add some records

```
INSERT INTO patient (ID, Gender, pname, phone)
VALUES
  (1, 'M', 'Ali Khan', '+923001234567'),
  (2, 'F', 'Fatima Ali', '+923451234567'),
  (3, 'M', 'Ahmed Hassan', '+923221234567'),
  (4, 'F', 'Ayesha Khan', '+923111234567');
```

```
SELECT * FROM patient;
```



```
INSERT INTO patient (ID, Gender, pname, phone)
VALUES
    (1, 'M', 'Ali Khan', '+923001234567'),
    (2, 'F', 'Fatima Ali', '+923451234567'),
    (3, 'M', 'Ahmed Hassan', '+923221234567'),
    (4, 'F', 'Ayesha Khan', '+923111234567');

SELECT * FROM patient;
```

ID	Gender	pname	phone
1	M	Ali Khan	+923001234567
2	F	Fatima Ali	+923451234567
3	M	Ahmed Hassan	+923221234567
4	F	Ayesha Khan	+923111234567

2. ALTER Command

`alter` command is used for altering the table structure, such as,

- ✔ to add a column to existing table
- ✔ to rename any existing column & Table
- ✔ to change datatype of any column or to modify its size.
- ✔ to drop a column from the table.

ALTER Command: Add a new Column

Using ALTER command we can add a column to any existing table. Following is the syntax,

ALTER TABLE table_name ADD(

column_name datatype);

Here is an Example for this,

ALTER TABLE student ADD(

address VARCHAR(200)

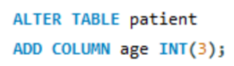
);

The above command will add a new column address to the table student, which will hold data of type varchar which is nothing but string, of length 200.

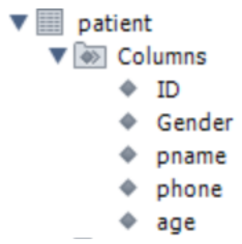
Example:

ALTER TABLE patient

ADD COLUMN age INT(3);



```
ALTER TABLE patient
ADD COLUMN age INT(3);
```



UPDATE patient

SET age = 25

WHERE ID = 1;

UPDATE patient

SET age = 30

WHERE ID = 2;

UPDATE patient

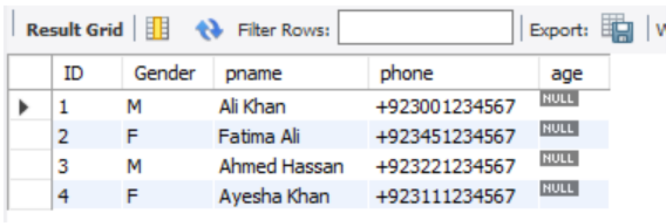
SET age = 40

WHERE ID = 3;

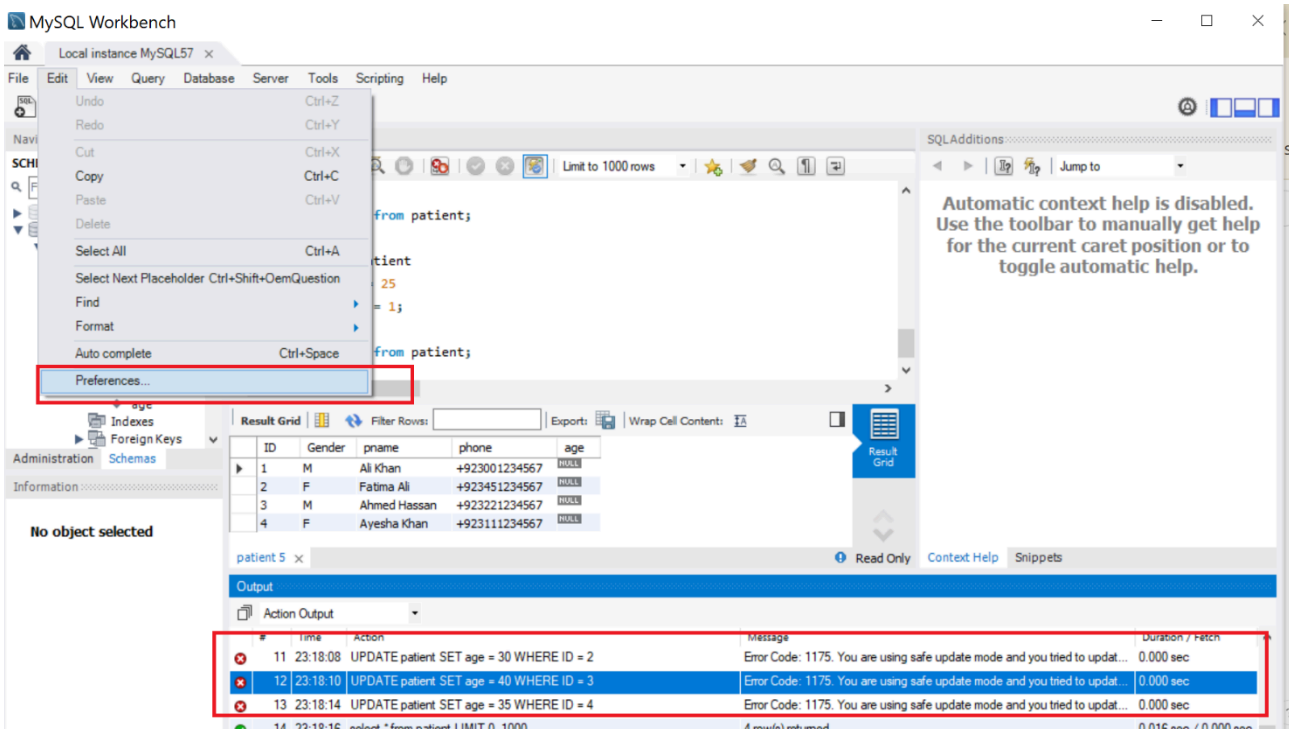
UPDATE patient

SET age = 35

WHERE ID = 4;



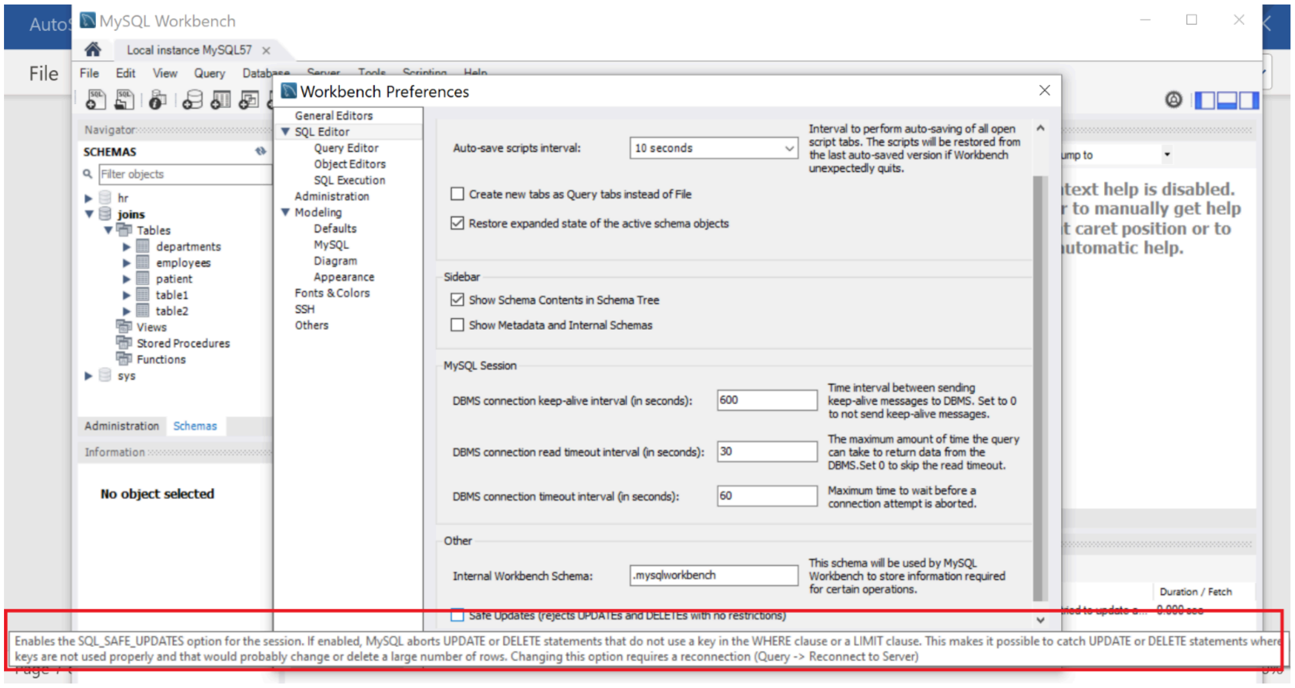
ID	Gender	pname	phone	age
1	M	Ali Khan	+923001234567	NULL
2	F	Fatima Ali	+923451234567	NULL
3	M	Ahmed Hassan	+923221234567	NULL
4	F	Ayesha Khan	+923111234567	NULL



```
from patient;
tient
25
= 1;
from patient;
```

ID	Gender	pname	phone	age
1	M	Ali Khan	+923001234567	NULL
2	F	Fatima Ali	+923451234567	NULL
3	M	Ahmed Hassan	+923221234567	NULL
4	F	Ayesha Khan	+923111234567	NULL

```
#   Time      Action                             Message                                                                 Duration / Fetch
11  23:18:08  UPDATE patient SET age = 30 WHERE ID = 2   Error Code: 1175. You are using safe update mode and you tried to updat...   0.000 sec
12  23:18:10  UPDATE patient SET age = 40 WHERE ID = 3   Error Code: 1175. You are using safe update mode and you tried to updat...   0.000 sec
13  23:18:14  UPDATE patient SET age = 35 WHERE ID = 4   Error Code: 1175. You are using safe update mode and you tried to updat...   0.000 sec
14  23:18:16  select * from patient LIMIT 0, 1000        4 row(s) returned                                                           0.016 sec / 0.000 sec
```



Auto-save scripts interval: 10 seconds — Interval to perform auto-saving of all open script tabs. The scripts will be restored from the last auto-saved version if Workbench unexpectedly quits.

☐ Create new tabs as Query tabs instead of File

☑ Restore expanded state of the active schema objects

Sidebar

☑ Show Schema Contents in Schema Tree

☐ Show Metadata and Internal Schemas

MySQL Session

DBMS connection keep-alive interval (in seconds): 600 — Time interval between sending keep-alive messages to DBMS. Set to 0 to not send keep-alive messages.

DBMS connection read timeout interval (in seconds): 30 — The maximum amount of time the query can take to return data from the DBMS.Set 0 to skip the read timeout.

DBMS connection timeout interval (in seconds): 60 — Maximum time to wait before a connection attempt is aborted.

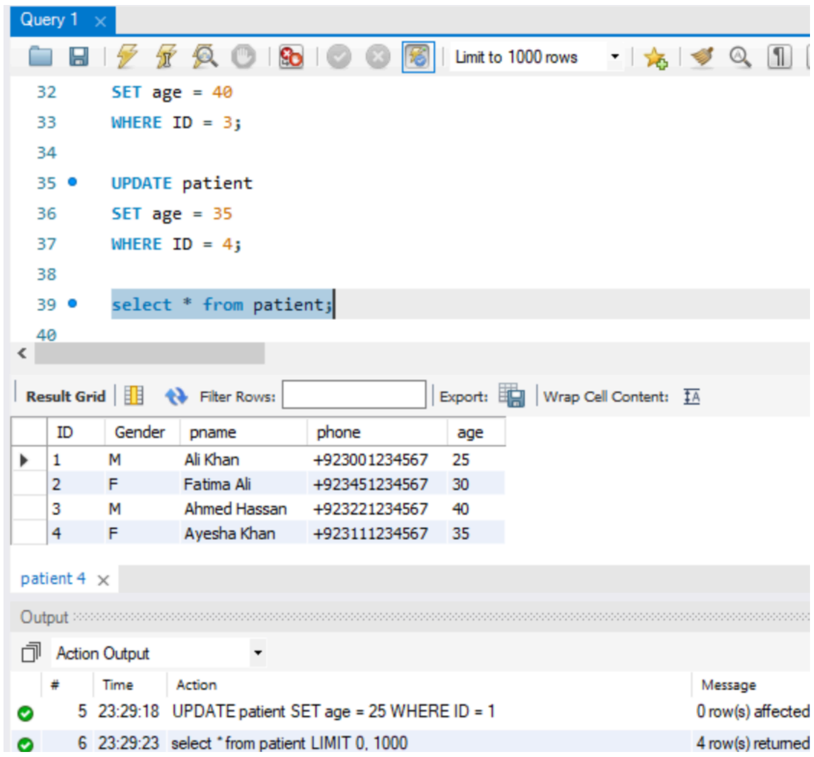
Other

Internal Workbench Schema: .mysqlworkbench — This schema will be used by MySQL Workbench to store information required for certain operations.

☐ Safe Updates (rejects UPDATEs and DELETEs with no restrictions)

Enables the SQL_SAFE_UPDATES option for the session. If enabled, MySQL aborts UPDATE or DELETE statements that do not use a key in the WHERE clause or a LIMIT clause. This makes it possible to catch UPDATE or DELETE statements where keys are not used properly and that would probably change or delete a large number of rows. Changing this option requires a reconnection (Query -> Reconnect to Server)

Restart and run query.



```
SET age = 40
WHERE ID = 3;

UPDATE patient
SET age = 35
WHERE ID = 4;

select * from patient;
```

ID	Gender	pname	phone	age
1	M	Ali Khan	+923001234567	25
2	F	Fatima Ali	+923451234567	30
3	M	Ahmed Hassan	+923221234567	40
4	F	Ayesha Khan	+923111234567	35

#	Time	Action	Message
5	23:29:18	UPDATE patient SET age = 25 WHERE ID = 1	0 row(s) affected
6	23:29:23	select * from patient LIMIT 0, 1000	4 row(s) returned

ALTER Command: Add multiple new Columns

Using ALTER command we can even add multiple new columns to any existing table. Following is the syntax,

```
ALTER TABLE table_name ADD(
  column_name1 datatype1,
  column-name2 datatype2,
  column-name3 datatype3);
```

Here is an Example for this,

```
ALTER TABLE student ADD(
  father_name VARCHAR(60),
  mother_name VARCHAR(60),
  dob DATE);
```

The above command will add three new columns to the student table

ALTER Command: Add Column with default value

ALTER command can add a new column to an existing table with a default value too. The default value is used when no value is inserted in the column. Following is the syntax, ALTER TABLE table_name ADD(

column-name1 datatype1 DEFAULT some_value

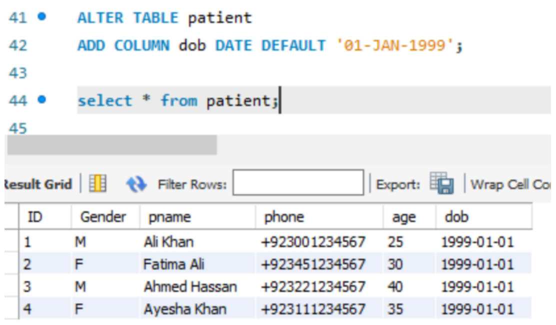
);

Here is an Example for this,

ALTER TABLE patient

ADD COLUMN dob DATE DEFAULT '01-JAN-1999';

The above command will add a new column with a preset default value to the table student.



```
ALTER TABLE patient
ADD COLUMN dob DATE DEFAULT '01-JAN-1999';

select * from patient;
```

ID	Gender	pname	phone	age	dob
1	M	Ali Khan	+923001234567	25	1999-01-01
2	F	Fatima Ali	+923451234567	30	1999-01-01
3	M	Ahmed Hassan	+923221234567	40	1999-01-01
4	F	Ayesha Khan	+923111234567	35	1999-01-01

ALTER Command: Modify an existing Column

ALTER command can also be used to modify data type of any existing column. Following is the syntax,

ALTER TABLE table_name modify(

column_name datatype

);

Here is an Example for this,

ALTER TABLE student MODIFY

address varchar(300);

Remember we added a new column address in the beginning? The above command will modify the address column of the student table, to now hold upto 300 characters.

ALTER Command: Modify multiple Columns

Using ALTER command we can even modify multiple columns to any existing table. Following is the syntax,

ALTER TABLE table_name

modify column_name datatype,

modify column_name2 datatype;

Here is an Example for this,

ALTER TABLE student

modify father_name VARCHAR(70),

modify mother_name VARCHAR(70);

The above command will modify the father_name & mother_name column of the student table, to now hold upto 70 characters.

ALTER Command: Rename a Column

Using ALTER command you can rename an existing column. Following is the syntax,

ALTER TABLE table_name CHANGE COLUMN

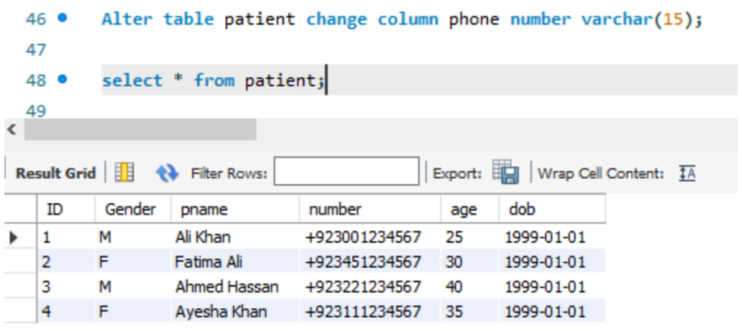
old_column_name new_column_name datatype;

Here is an example for this,

ALTER TABLE student CHANGE COLUMN

address location VARCHAR(20);

The above command will rename address column to location.



```
Alter table patient change column phone number varchar(15);

select * from patient;
```

ID	Gender	pname	number	age	dob
1	M	Ali Khan	+923001234567	25	1999-01-01
2	F	Fatima Ali	+923451234567	30	1999-01-01
3	M	Ahmed Hassan	+923221234567	40	1999-01-01
4	F	Ayesha Khan	+923111234567	35	1999-01-01

ALTER Command: Drop a Column

ALTER command can also be used to drop or remove columns. Following is the syntax,

ALTER TABLE table_name DROP

column_name;

Here is an example for this,

ALTER TABLE student DROP

address;

The above command will drop the address column from the table student.

ALTER Command: Drop Multiple Columns

ALTER command can also be used to drop or remove multiple columns. Following is the syntax,

```
ALTER TABLE table_name
  DROP column_name,
  DROP column_name2;
```

Here is an example for this,

```
ALTER TABLE student
  DROP address,
  DROP phone;
```

The above command will drop address & phone column from the table student.

3. DROP Command

Drop command is used to delete an existing SQL database or its objects.

DROP DATABASE command is used to drop or remove SQL database. Following is the syntax:

DROP DATABASE database_name;

Note: Be careful before dropping a database. Deleting a database will result in loss of complete information stored in the database!

DROP TABLE command is used to drop or remove table from database. Following is the syntax,

DROP TABLE table_name;

Drop column is already covered in alter command section.

4. TRUNCATE Command

The TRUNCATE TABLE command deletes the data inside a table, but not the table itself.

Following is the syntax,

TRUNCATE TABLE table_name;

5. RENAME Command

The rename command is used to change the name of an existing database object(like Table,Column) to a new name.

Renaming a table does not make it to lose any data is contained within it.

Following is the syntax,

RENAME TABLE current_name TO new_name;

You can also use command to rename a table name:

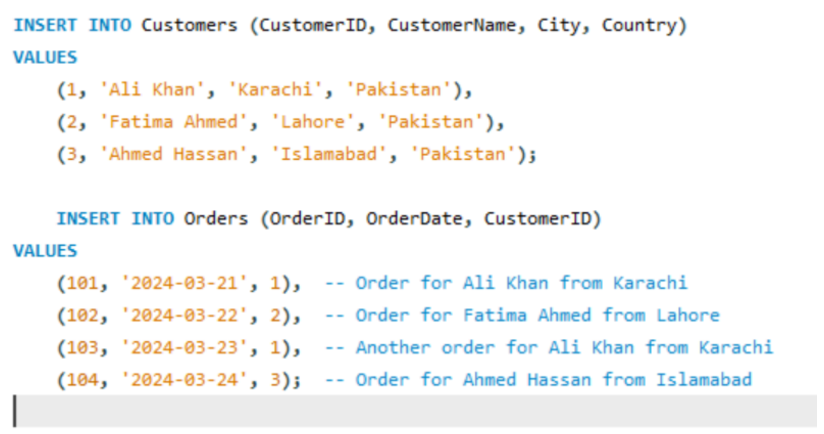
```
ALTER TABLE current_name RENAME new_name;
```

Comments in SQL

```
-- This is a single-line comment in SQL
/* This is a
  multi-line
  comment in SQL */
```

Primary Key & Foreign Key Concept

```
CREATE TABLE Customers (
   CustomerID INT PRIMARY KEY,
   CustomerName VARCHAR(50),
   City VARCHAR(50),
   Country VARCHAR(50)
);
CREATE TABLE Orders (
   OrderID INT PRIMARY KEY,
   OrderDate DATE,
   CustomerID INT,
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
INSERT INTO Customers (CustomerID, CustomerName, City, Country)
VALUES
   (1, 'Ali Khan', 'Karachi', 'Pakistan'),
   (2, 'Fatima Ahmed', 'Lahore', 'Pakistan'),
   (3, 'Ahmed Hassan', 'Islamabad', 'Pakistan');
INSERT INTO Orders (OrderID, OrderDate, CustomerID)
VALUES
   (101, '2024-03-21', 1),  -- Order for Ali Khan from Karachi
   (102, '2024-03-22', 2),  -- Order for Fatima Ahmed from Lahore
   (103, '2024-03-23', 1),  -- Another order for Ali Khan from Karachi
   (104, '2024-03-24', 3);  -- Order for Ahmed Hassan from Islamabad
```



```sql
INSERT INTO Customers (CustomerID, CustomerName, City, Country)
VALUES
    (1, 'Ali Khan', 'Karachi', 'Pakistan'),
    (2, 'Fatima Ahmed', 'Lahore', 'Pakistan'),
    (3, 'Ahmed Hassan', 'Islamabad', 'Pakistan');

    INSERT INTO Orders (OrderID, OrderDate, CustomerID)
VALUES
    (101, '2024-03-21', 1),  -- Order for Ali Khan from Karachi
    (102, '2024-03-22', 2),  -- Order for Fatima Ahmed from Lahore
    (103, '2024-03-23', 1),  -- Another order for Ali Khan from Karachi
    (104, '2024-03-24', 3);  -- Order for Ahmed Hassan from Islamabad
```

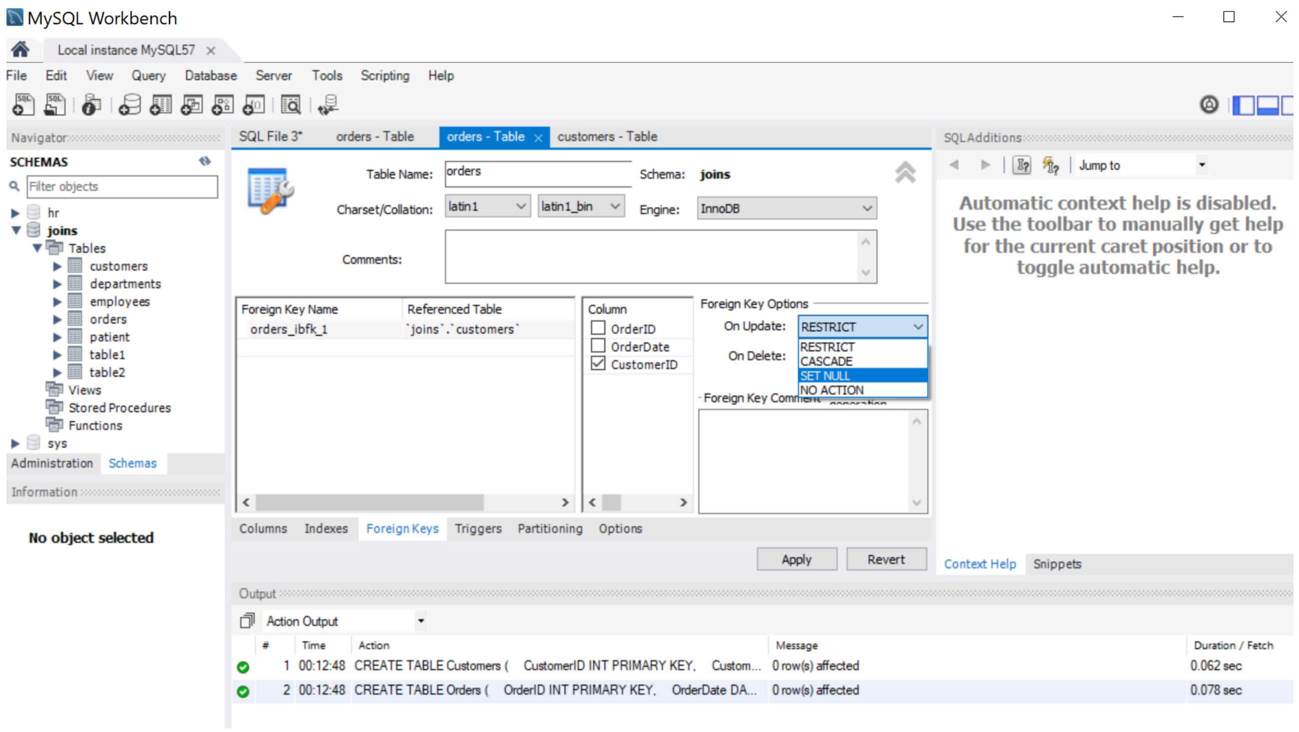

MySQL Workbench
Local instance MySQL57
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hr
joins
Tables
customers
departments
employees
orders
patient
table1
table2
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
SQL File 3* orders - Table orders - Table customers - Table
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets

Table Name: orders Schema: joins
Charset/Collation: latin1 latin1_bin Engine: InnoDB
Comments:

Foreign Key Name	Referenced Table
orders_ibfk_1	`joins`.`customers`

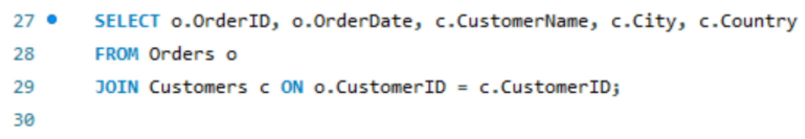
Column
☐ OrderID
☐ OrderDate
☑ CustomerID

Foreign Key Options
On Update: RESTRICT
On Delete: RESTRICT / CASCADE / SET NULL / NO ACTION
Foreign Key Comment

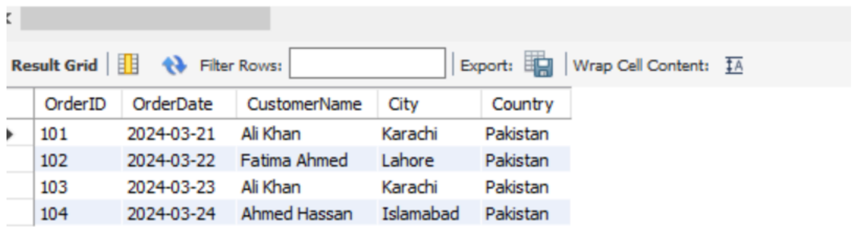
Output
Action Output

	#	Time	Action	Message	Duration / Fetch
✓	1	00:12:48	CREATE TABLE Customers (CustomerID INT PRIMARY KEY, Custom...	0 row(s) affected	0.062 sec
✓	2	00:12:48	CREATE TABLE Orders (OrderID INT PRIMARY KEY, OrderDate DA...	0 row(s) affected	0.078 sec

Retrieve orders along with customer information:



```sql
SELECT o.OrderID, o.OrderDate, c.CustomerName, c.City, c.Country
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID;
```



OrderID	OrderDate	CustomerName	City	Country
101	2024-03-21	Ali Khan	Karachi	Pakistan
102	2024-03-22	Fatima Ahmed	Lahore	Pakistan
103	2024-03-23	Ali Khan	Karachi	Pakistan
104	2024-03-24	Ahmed Hassan	Islamabad	Pakistan

Exercises (Class)

1. Add here all the tasks performed in lab.

Exercises (Weekly)

FACULTY:

FacultyID (integer, primary key)

FacultyName (25 characters)

COURSE:

CourseID (8 characters, primary key)

CourseName (15 characters)

CLASS:

ClassID (8 characters)

CourseID (8 characters foreign key)

SectionNo (integer)

Semester (10 characters)

STUDENT:

StudentID (integer, primary key)

StudentName (25 characters)

FacultyID (integer foreign key)

1. How would you add an attribute, CLASS, to the STUDENT table?

Using Employee table, solve the following queries

1. Create a replica of Employee table with all the records in it.
2. Add a column 'Address' in it.
3. Drop column 'Address' from it.
4. Add columns 'House No' character, 'Street No' numeric, 'Area' character,'City' character in it with the respective data types.
5. Change the data type of 'House No' from character to numeric.
6. Write a SQL statement to rename the table department to dept (with both methods).
7. Write a SQL statement to add a column regionId to the table locations.
8. Write a SQL statement to change the name of the column state_province to state in locations table, keeping the data type and size same.