* SQL- lab (Week 1 – Week 8)
* Assignments (week -05, week-07)
* Midterm Exam
* Final Exam
All together in one pdf file
(Birchwood University Note Files)

Student Name : Bishowjith Ghosh (Bidhan)

Week-01

In MySQL, a database is created using the `CREATE DATABASE` statement, and tables are created using the `CREATE TABLE` statement. The `USE` statement is used to switch to a specific database.

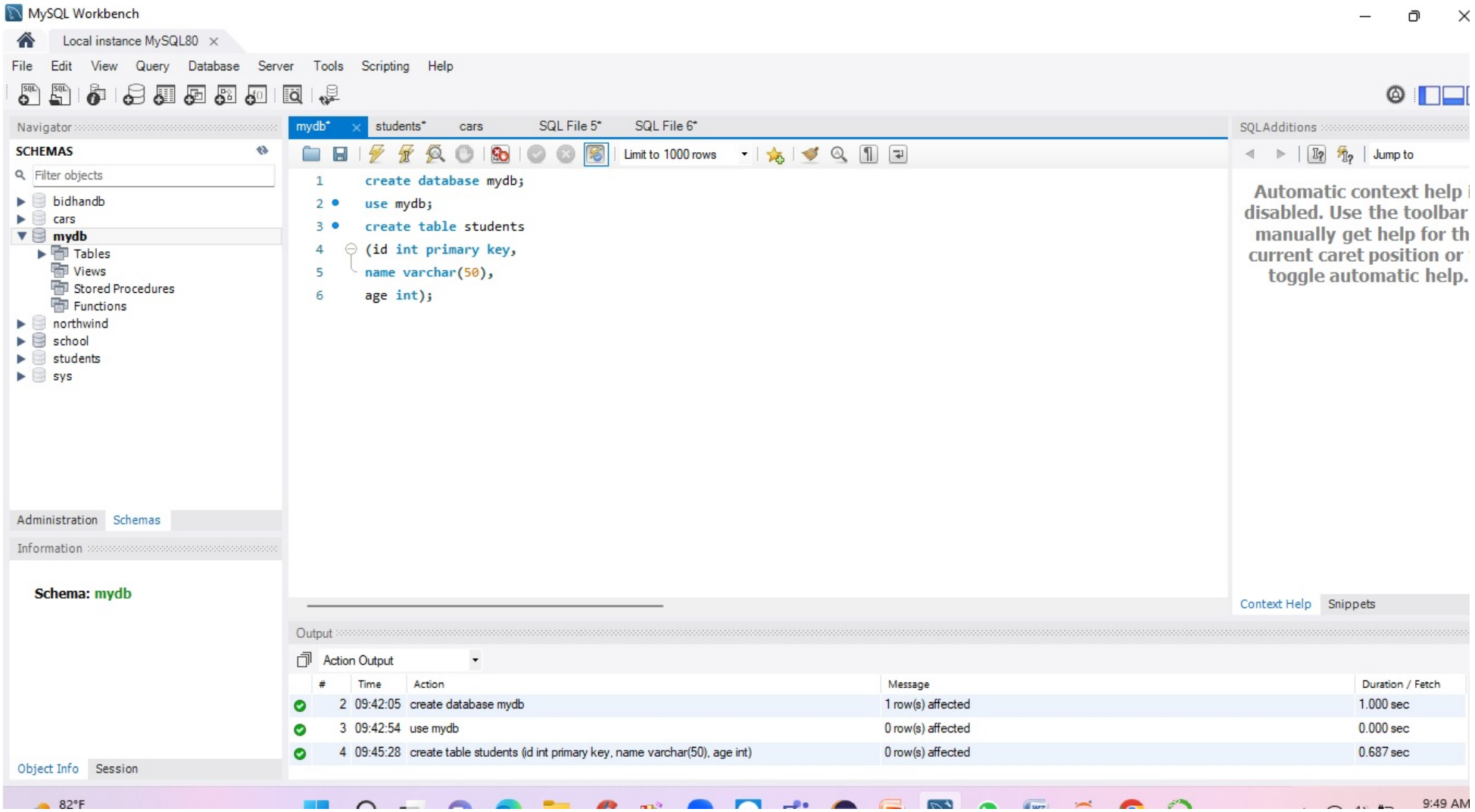
1. Create a database named "mydb":

```
CREATE DATABASE mydb;
```

2. Switch to the "mydb" database:

```
USE mydb;
```

3. Create a table named "students" with columns "id" (INT), "name" (VARCHAR), and "age" (INT):


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bidhandb
cars
mydb
Tables
Views
Stored Procedures
Functions
northwind
school
students
sys
Administration Schemas
Information
Schema: mydb
Object Info Session
mydb* students* cars SQL File 5* SQL File 6*
Limit to 1000 rows
1 create database mydb;
2 use mydb;
3 create table students
4 (id int primary key,
5 name varchar(50),
6 age int);
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
2 09:42:05 create database mydb 1 row(s) affected 1.000 sec
3 09:42:54 use mydb 0 row(s) affected 0.000 sec
4 09:45:28 create table students (id int primary key, name varchar(50), age int) 0 row(s) affected 0.687 sec
82°F
9:49 AM

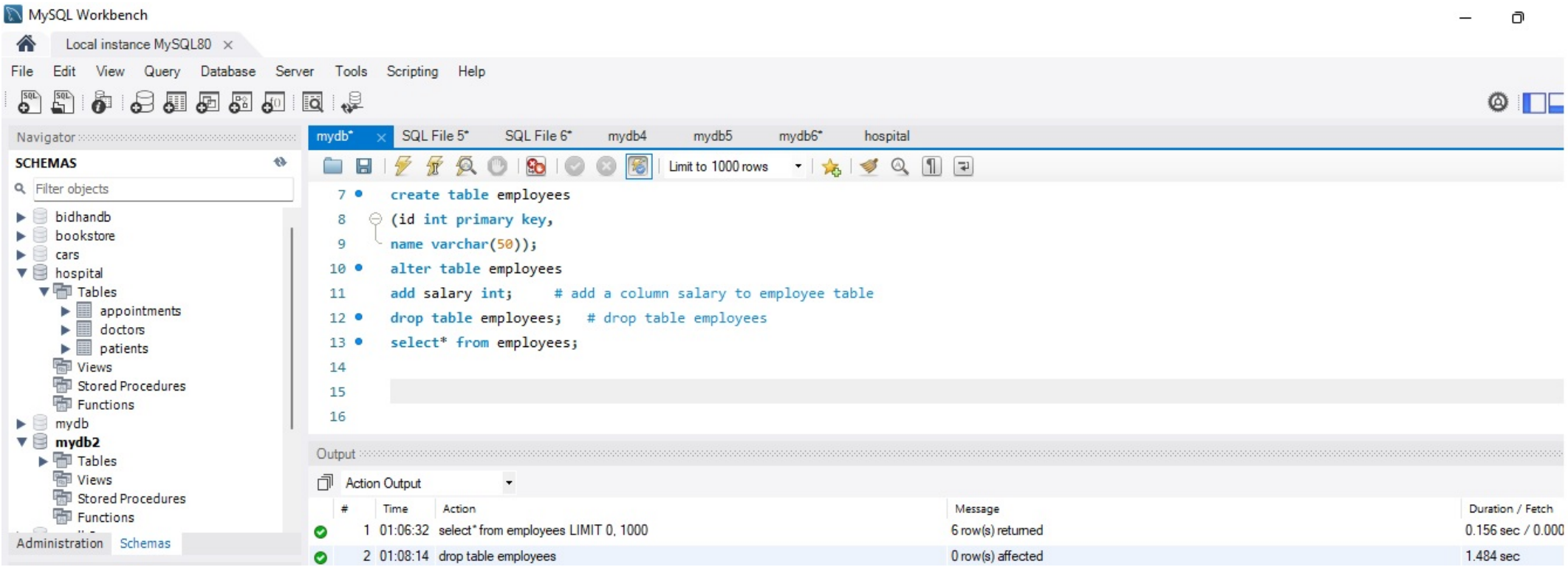
Week-02

DDL commands in MySQL are used to define and manage database objects such as tables, indexes, and constraints. The main DDL commands include CREATE, ALTER, and DROP.

1. Create a table named "employees" with columns "id" (INT) and "name" (VARCHAR):

2. Add a column "salary" (INT) to the "employees" table:

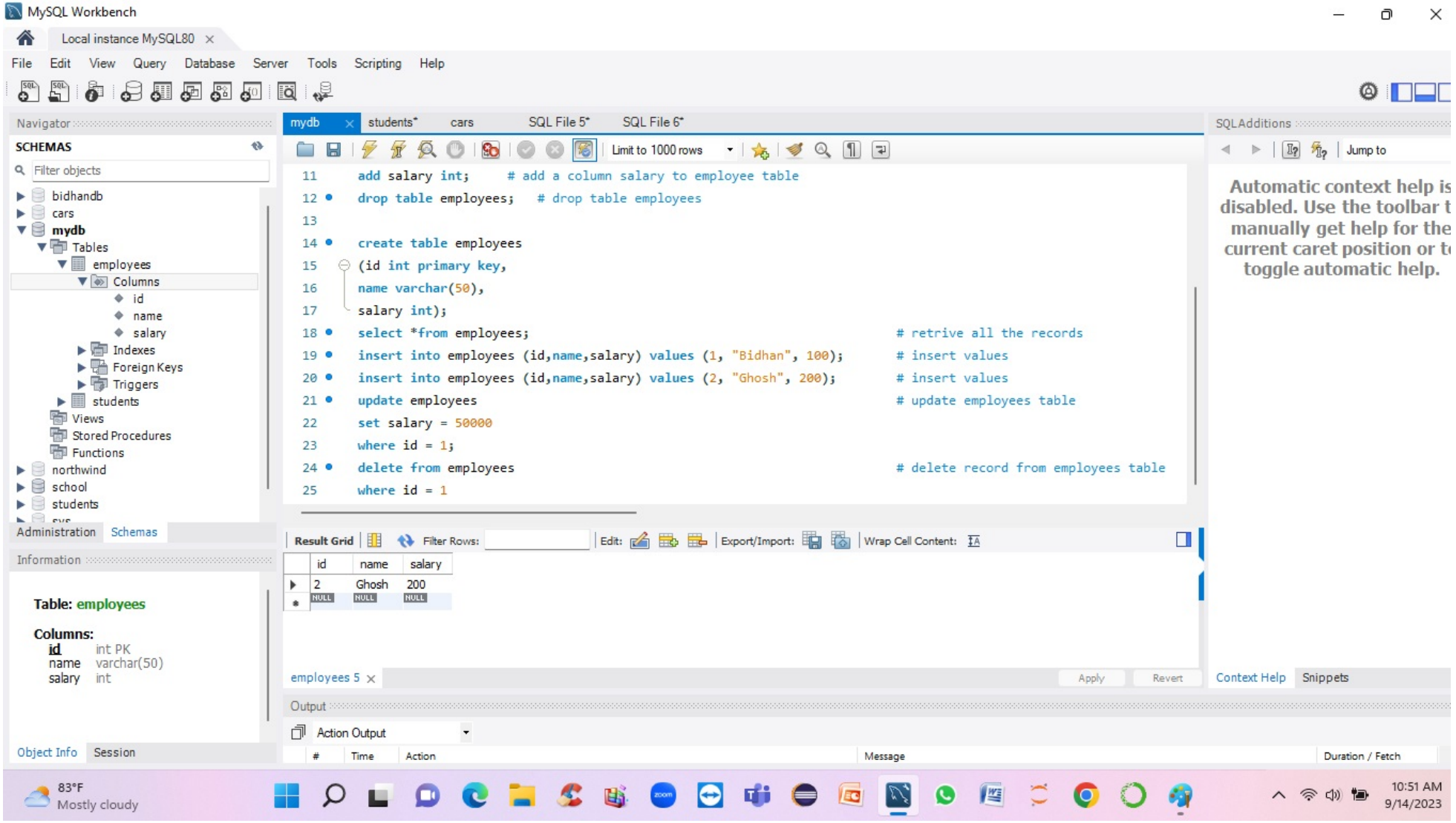
3. Drop the table "employees":


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bidhandb
bookstore
cars
hospital
Tables
appointments
doctors
patients
Views
Stored Procedures
Functions
mydb
mydb2
Tables
Views
Stored Procedures
Functions
Administration Schemas
mydb* SQL File 5* SQL File 6* mydb4 mydb5 mydb6* hospital
Limit to 1000 rows
7 create table employees
8 (id int primary key,
9 name varchar(50));
10 alter table employees
11 add salary int; # add a column salary to employee table
12 drop table employees; # drop table employees
13 select* from employees;
14
15
16
Output
Action Output
Time Action Message Duration / Fetch
1 01:06:32 select* from employees LIMIT 0, 1000 6 row(s) returned 0.156 sec / 0.000
2 01:08:14 drop table employees 0 row(s) affected 1.484 sec

Week-03

DML commands in MySQL are used to retrieve, insert, update, and delete data in database tables. The main DML commands include SELECT, INSERT, UPDATE, and DELETE.

1. Retrieve all records from the "employees" table:
2. Insert a new record into the "employees" table:
3. Update the salary of an employee in the "employees" table:
4. Delete a record from the "employees" table:


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bidhandb
cars
mydb
Tables
employees
Columns
id
name
salary
Indexes
Foreign Keys
Triggers
students
Views
Stored Procedures
Functions
northwind
school
students
Administration Schemas
Information
Table: employees
Columns:
id int PK
name varchar(50)
salary int
Object Info Session
mydb students* cars SQL File 5* SQL File 6*
Limit to 1000 rows
11 add salary int; # add a column salary to employee table
12 drop table employees; # drop table employees
13
14 create table employees
15 (id int primary key,
16 name varchar(50),
17 salary int);
18 select *from employees; # retrive all the records
19 insert into employees (id,name,salary) values (1, "Bidhan", 100); # insert values
20 insert into employees (id,name,salary) values (2, "Ghosh", 200); # insert values
21 update employees # update employees table
22 set salary = 50000
23 where id = 1;
24 delete from employees # delete record from employees table
25 where id = 1
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
id name salary
2 Ghosh 200
NULL NULL NULL
employees 5
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
83°F
Mostly cloudy
10:51 AM
9/14/2023

Week-04

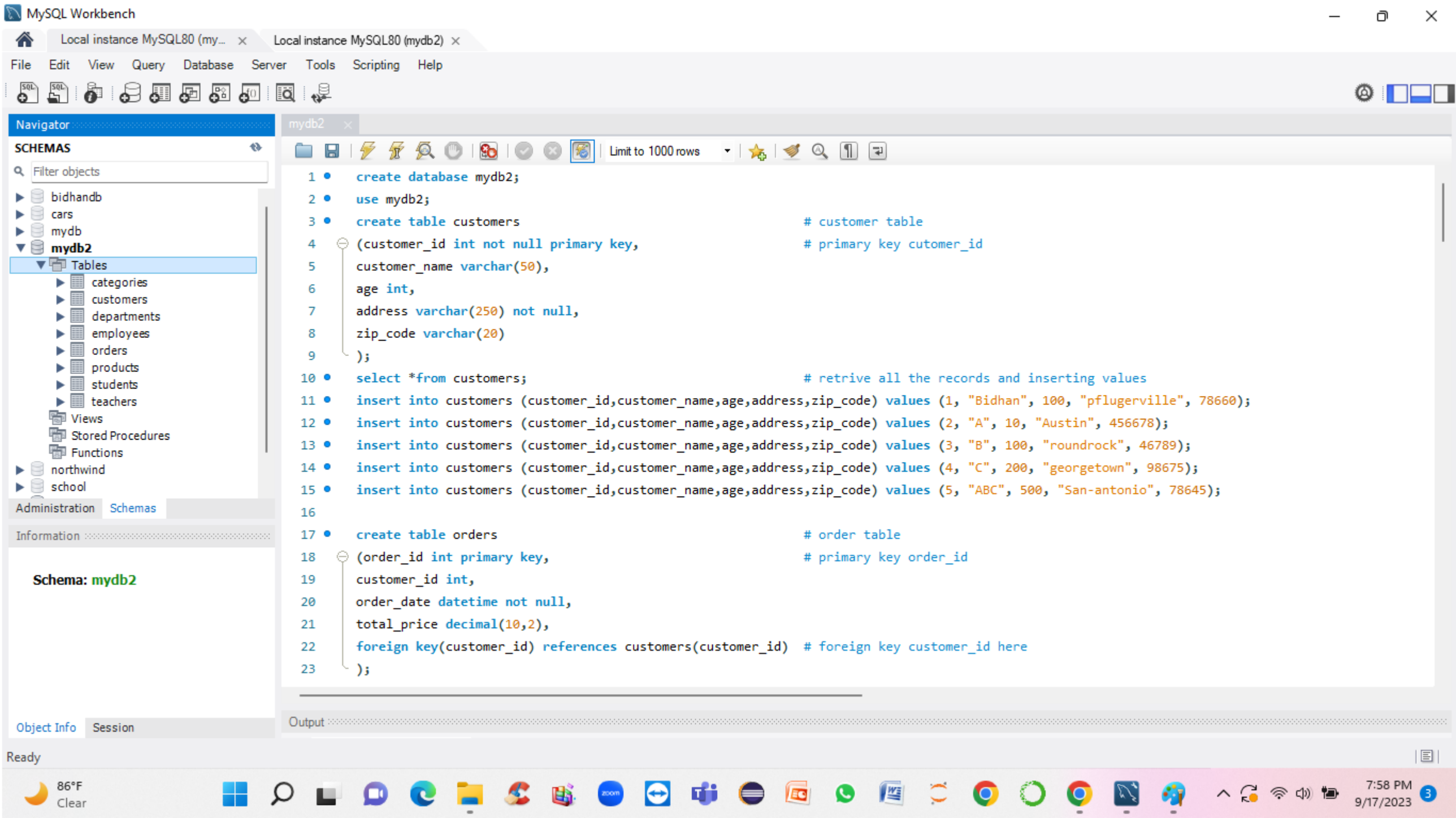
Joins in MySQL are used to combine rows from multiple tables based on a related column between them. The main types of joins include INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN.

1. Inner join between the "orders" and "customers" tables on the "customer_id" column:

2. Left join between the "products" and "categories" tables on the "category_id" column:

3. Right join between the "employees" and "departments" tables on the "department_id" column:

4. Full join between the "students" and "teachers" tables on the "class_id" column:


MySQL Workbench
Local instance MySQL80 (my...
Local instance MySQL80 (mydb2)
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bidhandb
cars
mydb
mydb2
Tables
categories
customers
departments
employees
orders
products
students
teachers
Views
Stored Procedures
Functions
northwind
school
Administration Schemas
Information
Schema: mydb2
Object Info Session
mydb2
Limit to 1000 rows
1 create database mydb2;
2 use mydb2;
3 create table customers # customer table
4 (customer_id int not null primary key, # primary key cutomer_id
5 customer_name varchar(50),
6 age int,
7 address varchar(250) not null,
8 zip_code varchar(20)
9);
10 select *from customers; # retrive all the records and inserting values
11 insert into customers (customer_id,customer_name,age,address,zip_code) values (1, "Bidhan", 100, "pflugerville", 78660);
12 insert into customers (customer_id,customer_name,age,address,zip_code) values (2, "A", 10, "Austin", 456678);
13 insert into customers (customer_id,customer_name,age,address,zip_code) values (3, "B", 100, "roundrock", 46789);
14 insert into customers (customer_id,customer_name,age,address,zip_code) values (4, "C", 200, "georgetown", 98675);
15 insert into customers (customer_id,customer_name,age,address,zip_code) values (5, "ABC", 500, "San-antonio", 78645);
16
17 create table orders # order table
18 (order_id int primary key, # primary key order_id
19 customer_id int,
20 order_date datetime not null,
21 total_price decimal(10,2),
22 foreign key(customer_id) references customers(customer_id) # foreign key customer_id here
23);
Output
Ready
86°F Clear
7:58 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - employees
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school

Administration | Schemas

Information

Schema: mydb2

Object Info | Session

mydb2

Limit to 1000 rows

```
29  INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES (101, 1, '2023-10-01', 50000);
30  INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES (102, 2, '2022-10-11', 340000);
31  INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES (103, 3, '2021-10-08', 40000);
32  INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES (104, 4, '2020-06-10', 870000);
33  INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES (105, 5, '2023-09-10', 27000);
34
35  -- Inner join between orders and customers tables on the customer_id column
36  SELECT orders.order_id, customers.customer_name
37  FROM orders
38  INNER JOIN customers ON orders.customer_id = customers.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_name
101	Bidhan
102	A
103	B
104	C
105	ABC

Result Grid | Form Editor

Result 1 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	20:00:22	SELECT orders.order_id, customers.customer_name FROM orders INNER JOIN custome...	5 row(s) returned	0.234 sec / 0.000 sec

Query Completed

86°F Clear

8:00 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - employees
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school

Administration | Schemas

Information

Schema: mydb2

Object Info | Session

mydb2*

Limit to 1000 rows

```
40 • create table categories                                  # categories table
41   (category_id int not null primary key,                   # primary key category_id
42   category_name varchar(255)
43   );
44
45 • INSERT INTO categories (category_id, category_name) VALUES (901, "Computer Accessories");
46 • INSERT INTO categories (category_id, category_name) VALUES (904, "Clothing");
47 • INSERT INTO categories (category_id, category_name) VALUES (907, "Computer Accessories");
48
49 • create table products                                    # products table
50   (product_id int not null primary key,                    # primary key product_id
51   product_name varchar(50),
52   category_id int,                                         # category id foreign key
53   price decimal(10,2)
54   );
55 • INSERT INTO products (product_id, product_name, category_id, price) VALUES (500, "Mouse", 901, 1000.58);
56 • INSERT INTO products (product_id, product_name, category_id, price) VALUES (501, "Ram", 902, 2000.98);
57 • INSERT INTO products (product_id, product_name, category_id, price) VALUES (502, "Shirt", 903, 200.58);
58 • INSERT INTO products (product_id, product_name, category_id, price) VALUES (503, "T-short", 904, 500.98);
59 • INSERT INTO products (product_id, product_name, category_id, price) VALUES (504, "Jeans", 905, 300.98);
60 • INSERT INTO products (product_id, product_name, category_id, price) VALUES (505, "SSD", 906, 3000.58);
61 • INSERT INTO products (product_id, product_name, category_id, price) VALUES (506, "Gaming Pc", 907,5000.98);
62 • INSERT INTO products (product_id, product_name, category_id, price) VALUES (507, "Laptop", 908, 7000.98);
```

Output

Query Completed

83°F Clear

8:05 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - employees
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school

Administration | Schemas

Information

Schema: mydb2

Object Info | Session

mydb2*

Limit to 1000 rows

```
61 • INSERT INTO products (product_id, product_name, category_id, price) VALUES (506, "Gaming Pc", 907,5000.98);
62 • INSERT INTO products (product_id, product_name, category_id, price) VALUES (507, "Laptop", 908, 7000.98);
63
64   -- left join between categories and products tables on the category_id column
65 • Select products.product_id, products.product_name, category_name
66   from products
67   left join categories on products.category_id = categories.category_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	product_name	category_name
500	Mouse	Computer Accessories
501	Ram	NULL
502	Shirt	NULL
503	T-short	Clothing
504	Jeans	NULL
505	SSD	NULL
506	Gaming Pc	Computer Accessories
507	Laptop	NULL

Result Grid | Form Editor | Field Types

Result 5 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
3	20:12:01	select employees.employee_name,departments.departments_name from employees righ...	4 row(s) returned	0.063 sec / 0.000 sec
4	20:21:07	SELECT teachers.teacher_name, students.student_name FROM teachers LEFT JOIN ...	5 row(s) returned	0.000 sec / 0.000 sec
5	20:27:25	Select products.product_id, products.product_name, category_name from products left ...	8 row(s) returned	0.000 sec / 0.000 sec

Query Completed

86°F Mostly clear

8:28 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - employees
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school

Administration | Schemas

Information

Schema: mydb2

Object Info | Session

mydb2*

Limit to 1000 rows

```
69 • create table employees                                    # employee table
70   (employee_id int primary key,                             # primary key employee_id
71   employee_name varchar(50),
72   department varchar(50),
73   department_id int,                                        # department_id foreign key
74   salary int);
75 • select *from employees;                                   # retrive all the records
76 • insert into employees (employee_id,employee_name,department,department_id,salary) values (601, "Bidhan","Data Science",1001, 1800);          # in
77 • insert into employees (employee_id,employee_name,department,department_id,salary) values (602, "Ghosh","Chemistry",1002, 2400);
78 • insert into employees (employee_id,employee_name,department,department_id,salary) values (603, "BG","Business",1003, 3900);
79 • insert into employees (employee_id,employee_name,department,department_id,salary) values (604, "ABC","Physics",1004, 4600);
80 • insert into employees (employee_id,employee_name,department,department_id,salary) values (605, "BCD","Mathmatics",1005, 5300);
81 • insert into employees (employee_id,employee_name,department,department_id,salary) values (606, "Austin", "AI",1006, 5100);
82 • insert into employees (employee_id,employee_name,department,department_id,salary) values (607, "Justin","Machine Learning",1007, 700);
83 • insert into employees (employee_id,employee_name,department,department_id,salary) values (608, "Bobby","Programming-R",1008, 3600);
84
85 • update employees                                            # update employees table
86   set salary = 250000
87   where employee_id = 603;
88 • delete from employees                                       # delete record from employees table
89   where employee_id = 607;
90 • delete from employees                                       # delete record from employees table
91   where employee_id = 608;
```

Output

Query Completed

83°F Clear

8:11 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - employees
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school

Administration | Schemas

Information

Schema: mydb2

Object Info | Session

mydb2*

Limit to 1000 rows

```
95  create table departments                                    # department table
96  (department_id int primary key,                             # primary key employee_id
97  departments_name varchar(50)
98  );
99
100 select *from departments;
101 insert into departments(department_id,departments_name) values ( 1001, "Masters");
102 insert into departments(department_id,departments_name) values ( 1003, "BCom" );
103 insert into departments(department_id,departments_name) values (1005, "BSc");
104 insert into departments(department_id,departments_name) values ( 1006, "Post Graduate Program" );
105
106 -- right join between "employees" and "departments" tables on the "departments" id column
107 select employees.employee_name,departments.departments_name
108 from employees
109 right join departments on employees.department_id = departments.department_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee_name	departments_name
Bidhan	Masters
BG	BCom
BCD	BSc
Austin	Post Graduate Program

Result Grid | Form Editor

Result 3 | Read Only

Output

Query Completed

83°F Clear

8:13 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - employees
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school

Administration | Schemas

Information

Schema: mydb2

Object Info | Session

mydb2*

Limit to 1000 rows

```
111  create table teachers                                  # table teachers
112  (teacher_id int primary key,
113  teacher_name varchar(25),
114  teacher_age int,
115  contact_no int,
116  gender char(10),
117  department varchar(25) );
118
119  insert into teachers(teacher_id, teacher_name, teacher_age, contact_no, gender, department) values(1, 'Ap' , 30, 6897, "Male", "Science") ;
120  insert into teachers(teacher_id, teacher_name, teacher_age, contact_no, gender, department) values(2, 'bc' , 30, 2563, "Male", "Science") ;
121  insert into teachers(teacher_id, teacher_name, teacher_age, contact_no, gender, department) values(3, 'Ab' , 33, 6372, "Female", "Science") ;
122  insert into teachers(teacher_id, teacher_name, teacher_age, contact_no, gender, department) values(4, 'Mr Piterson' , 32, 6233, "Male", "Science
123  insert into teachers(teacher_id, teacher_name, teacher_age, contact_no, gender, department) values(5, 'Ad' , 31, 6444, "Female", "Science") ;
124
125  select * from teachers;                                # showing all records of teachers table
126  select teacher_name, gender from teachers;             # showing teacher name and gender records only
127  select * from teachers limit 2;                        # showing first two records
128
129  update teachers set contact_no = 12345, department= "Commerce", teacher_name ="Rafiq" where teacher_id = 1;        # update teachers table
130  update teachers set contact_no = 75300, department= "Data Science", teacher_name ="Mr john" where teacher_id = 2;
131
132  select * from teachers;
133  select distinct contact_no from teachers;              # showing only contact number only one column
```

Output

Query Completed

83°F Clear

8:15 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - employees
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school

Administration | Schemas

Information

Schema: mydb2

Object Info | Session

mydb2*

Limit to 1000 rows

```
134 • select distinct teacher_name from teachers;                    # showing distinct column
135 • select * from teachers where teacher_name = "Mr Piterson";     # showing a particular person's record
136 • select contact_no as "Contact_information" from teachers;      # provide contact_no as contact_information as output
137
138 • select teacher_age, min(contact_no) as "ZZZZZZZZ" from teachers group by teacher_age;
139 • select * from teachers;
140
141 • select teacher_age, min(teacher_id) as "T-id" from teachers group by teacher_age;       # use of group by
142
143 • select * from teachers;
144 • alter table teachers add Email varchar(225);                   # add a new table to teachers table
145
146 • update teachers set Email = "john@gmail.com" where teacher_id = 1;  # inputing values to email column
147 • update teachers set Email= "bidhan@gmail.com" where teacher_id = 2;
148
149 • alter table teachers add class_id int;                         # here class_id is foreign key
150 • update teachers set class_id= 1 where teacher_id = 1;          # adding value for the new column class_id
151 • update teachers set class_id= 2 where teacher_id = 2;
152 • update teachers set class_id= 3 where teacher_id = 3;
153 • update teachers set class_id= 4 where teacher_id = 4;
154 • update teachers set class_id= 5 where teacher_id = 5;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch

Query Completed

83°F Clear

8:18 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - employees
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school

Administration | Schemas

Information

Schema: mydb2

Object Info | Session

mydb2*

Limit to 1000 rows

```
155
156 •  create table students
157   (student_id int primary key,
158   student_name varchar(25),
159   class_id int,
160   student_gender char(10)
161    );
162
163 •   Select* from students;
164
165 •   insert into students (student_id, student_name,class_id, student_gender) values (655,"Zack", 1, "Male");
166 •   insert into students (student_id,student_name,class_id, student_gender) values (658,"Rose", 2, "Female");
167 •   insert into students (student_id,student_name,class_id, student_gender) values (659,"Alex", 3, "Male");
168 •   insert into students (student_id,student_name,class_id, student_gender) values (660,"Smith", 4, "Male");
169 •   insert into students (student_id,student_name,class_id, student_gender) values (661, "Maria",5, "Female");
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	20:00:22	SELECT orders.order_id, customers.customer_name FROM orders INNER JOIN custome...	5 row(s) returned	0.234 sec / 0.000 sec
2	20:08:06	Select products.product_id, products.product_name, category_name from products left jo...	8 row(s) returned	0.157 sec / 0.000 sec
3	20:12:01	select employees.employee_name,departments.departments_name from employees right j...	4 row(s) returned	0.063 sec / 0.000 sec

Query Completed

83°F Clear

8:20 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - employees
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school

Administration | Schemas

Information

Schema: mydb2

Object Info | Session

mydb2*

Limit to 1000 rows

```
171  -- Full join between "teachers" and "students" tables on the "class_id" column
172
173  -- Perform a LEFT JOIN from teachers to students
174  SELECT teachers.teacher_name, students.student_name
175  FROM teachers
176  LEFT JOIN students ON teachers.class_id = students.class_id
177
178  UNION
179
180  -- Perform a RIGHT JOIN from teachers to students
181  SELECT teachers.teacher_name, students.student_name
182  FROM teachers
183  RIGHT JOIN students ON teachers.class_id = students.class_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

teacher_name	student_name
Rafiq	Zack
Mr john	Rose
Ab	Alex
Mr Piterson	Smith
Ad	Maria

Result Grid | Form Editor

Result 4 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
3	20:12:01	select employees.employee_name departments.department_name from employees righ	4 row(s) returned	0.063 sec / 0.000 sec

Query Completed

83°F Clear

8:22 PM 9/17/2023

Week-05

- The WHERE clause is used to filter rows based on a specific condition.
- The GROUP BY clause is used to group rows based on one or more columns.
- The HAVING clause is used to filter grouped rows based on a condition.

1. Retrieve customers from the "customers" table where the country is set to 'USA':

2. Group customers by country and count the number of customers in each country:

3. Retrieve countries and their counts from the "customers" table, only including countries with more than one customer:

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- mydb2
- **mydb3**
  - Tables
    - customers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school
- students
- sys

Administration | Schemas

Information

Schema: mydb3

Object Info | Session

mydb3*

Limit to 1000 rows

```sql
create database mydb3;
use mydb3;
create table customers
(customer_id int not null primary key,
customer_name varchar(250),
age int,
address varchar(250),
country varchar(250));
INSERT INTO customers (customer_id, customer_name, age, address, country)    # Insert data into the "customers" table
VALUES
    (1, 'John Peter', 35, '123 Main St', 'USA'),
    (2, 'Stive Smith', 28, '456 Elm St', 'USA'),
    (3, 'Boby Johnson', 42, '789 Oak St', 'Canada'),
    (4, 'Eva Martinez', 30, '101 Pine St', 'Mexico'),
    (5, 'Mia Kim', 25, '202 Cedar St', 'USA'),
    (6, 'David Lee', 38, '303 Birch St', 'Canada'),
    (7, 'Sophia Wilson', 29, '404 Maple St', 'USA'),
    (8, 'Liam Brown', 45, '505 Redwood St', 'Mexico'),
    (9, 'Olivia Davis', 27, '606 Spruce St', 'Canada');

select* from customers ;                                    # retrive all the data
select * from customers where country = 'USA' ;             # Retrieve customers from the customers table where the country is set to USA:
select country, COUNT(*) from customers group by country;   # Group customers by country and count the number of customers in each country:
select country, COUNT(*) FROM customers group by country having COUNT(*) > 1; # Retrieve countries and their counts with more than one customer:
```

SQL script saved to 'D:\DataScience\DataElement\mydb3.sql'

82°F Mostly clear

9:35 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- mydb2
- **mydb3**
  - Tables
    - customers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school
- students
- sys

Administration | Schemas

Information

Schema: mydb3

Object Info | Session

mydb3*

Limit to 1000 rows

```
20
21 • select* from customers ;                                  # retrive all the data
22 • select * from customers where country = 'USA' ;           # Retrieve customers from the customers table where the country is set to USA:
23 • select country, COUNT(*) from customers group by country;  # Group customers by country and count the number of customers in each country:
24 • select country, COUNT(*) FROM customers group by country having COUNT(*) > 1; # Retrieve countries and their counts with more than one customer:
25
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	customer_name	age	address	country
1	John Peter	35	123 Main St	USA
2	Stive Smith	28	456 Elm St	USA
5	Mia Kim	25	202 Cedar St	USA
7	Sophia Wilson	29	404 Maple St	USA
NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor

customers 11

Apply | Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
19	21:31:40	SELECT country, COUNT(*) FROM customers group by country having COUNT(*) > 1 ...	3 row(s) returned	0.032 sec / 0.000 sec
20	21:31:50	select country, COUNT(*) from customers group by country LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
21	21:32:01	SELECT country, COUNT(*) FROM customers group by country having COUNT(*) > 1 ...	3 row(s) returned	0.000 sec / 0.000 sec
22	21:35:56	select * from customers where country = 'USA' LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Query Completed

82°F Mostly clear

9:36 PM 9/17/2023

MySQL Workbench

Local instance MySQL80 (my... | Local instance MySQL80 (mydb2)

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- mydb2
- **mydb3**
  - Tables
    - customers
  - Views
  - Stored Procedures
  - Functions
- northwind
- school
- students
- sys

Administration | Schemas

Information

Schema: mydb3

Object Info | Session

mydb3*

Limit to 1000 rows

```
18        (8, 'Liam Brown', 45, '505 Redwood St', 'Mexico'),
19        (9, 'Olivia Davis', 27, '606 Spruce St', 'Canada');
20
21 •  select* from customers ;                                  # retrive all the data
22 •  select * from customers where country = 'USA' ;           # Retrieve customers from the customers table where the country is set to USA:
23 •  select country, COUNT(*) from customers group by country;   # Group customers by country and count the number of customers in each country:
24 •  select country, COUNT(*) FROM customers group by country having COUNT(*) > 1; # Retrieve countries and their counts with more than one customer:
25
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

country	COUNT(*)
USA	4
Canada	3
Mexico	2

Result Grid

Result 12

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
20	21:31:50	select country, COUNT(*) from customers group by country LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
21	21:32:01	SELECT country, COUNT(*) FROM customers group by country having COUNT(*) > 1 ...	3 row(s) returned	0.000 sec / 0.000 sec
22	21:35:56	select * from customers where country = 'USA' LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
23	21:37:20	select country, COUNT(*) FROM customers group by country having COUNT(*) > 1 LIM...	3 row(s) returned	0.000 sec / 0.000 sec

Query Completed

82°F Mostly clear

9:38 PM 9/17/2023

Week-06

- Views in MySQL are virtual tables derived from the result of a query. They provide an alternative way to represent data from one or more tables.

-Triggers in MySQL are stored programs that are automatically executed or fired in response to specific events or actions performed on a table.

1. Create a view named "product_prices" to display product names and their prices from the "products" table:

2. Create a trigger named "update_inventory" to update the inventory count of a product after an insert or update operation on the "orders" table:

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- bookstore
- cars
- hospital
  - Tables
  - Views
  - Stored Procedures
  - Functions
- librarydb
- mydb
- mydb2
- mydb3
- mydb4
- mydb5
- **mydb6**
  - Tables
  - Views

Administration Schemas

Information

No object selected

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db

Limit to 1000 rows

```
1  create database mydb6;
2  use mydb6;
3  CREATE TABLE products (
4      product_id INT PRIMARY KEY,
5      product_name VARCHAR(255),
6      price DECIMAL(10, 2),
7      inventory_count INT
8  );
9  INSERT INTO products (product_id, product_name, price, inventory_count)
10 VALUES
11     (1, 'Product A', 19.99, 100),
12     (2, 'Product B', 29.99, 75),
13     (3, 'Product C', 9.99, 200),
14     (4, 'Product D', 49.99, 50);
15 select * from products;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	product_name	price	inventory_count
1	Product A	19.99	90
2	Product B	29.99	75
3	Product C	9.99	200
4	Product D	49.99	50
NULL	NULL	NULL	NULL

Result Grid

Form Editor

products 11

Apply Revert

Output

Action Output

93°F Sunny

Search

2:23 PM 10/1/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- bookstore
- cars
- hospital
  - Tables
  - Views
  - Stored Procedures
  - Functions
- librarydb
- mydb
- mydb2
- mydb3
- mydb4
- mydb5
- **mydb6**
  - Tables
  - Views

Administration Schemas

Information

No object selected

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db

Limit to 1000 rows

```
16  # 1. Create a view named "product_prices" to display product names and their prices from the "products" table:
17  CREATE VIEW product_prices AS
18  SELECT product_name, price FROM products;
19
20  CREATE TABLE orders (
21      order_id INT PRIMARY KEY,
22      product_id INT,
23      quantity INT,
24      order_date DATE);
25  INSERT INTO orders (order_id, product_id, quantity, order_date)
26  VALUES
27      (1, 1, 5, '2023-09-23'),
28      (2, 2, 3, '2023-09-24'),
29      (3, 3, 2, '2023-09-25'),
30      (4, 4, 1, '2023-09-26');
31  select* from orders;
32
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_name	price
Product A	19.99
Product B	29.99
Product C	9.99
Product D	49.99

Result Grid

Form Editor

products 13

Read Only

Output

Show desktop

93°F Sunny

Search

2:27 PM 10/1/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- bookstore
- cars
- hospital
  - Tables
  - Views
  - Stored Procedures
  - Functions
- librarydb
- mydb
- mydb2
- mydb3
- mydb4
- mydb5
- **mydb6**
  - Tables
  - Views

Administration Schemas

Information

No object selected

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db

Limit to 1000 rows

```
16  # 1. Create a view named "product_prices" to display product names and their prices from the "products" table:
17  CREATE VIEW product_prices AS
18  SELECT product_name, price FROM products;
19
20  CREATE TABLE orders (
21      order_id INT PRIMARY KEY,
22      product_id INT,
23      quantity INT,
24      order_date DATE);
25  INSERT INTO orders (order_id, product_id, quantity, order_date)
26  VALUES
27      (1, 1, 5, '2023-09-23'),
28      (2, 2, 3, '2023-09-24'),
29      (3, 3, 2, '2023-09-25'),
30      (4, 4, 1, '2023-09-26');
31  select* from orders;
32
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

order_id	product_id	quantity	order_date
1	1	5	2023-09-23
2	2	3	2023-09-24
3	3	2	2023-09-25
4	4	1	2023-09-26
5	1	10	2023-09-27
NULL	NULL	NULL	NULL

orders 12

Result Grid Form Editor

Apply Revert

Output

93°F Sunny

Search

2:25 PM 10/1/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- bookstore
- cars
- hospital
  - Tables
  - Views
  - Stored Procedures
  - Functions
- librarydb
- mydb
- mydb2
- mydb3
- mydb4
- mydb5
- **mydb6**
  - Tables
  - Views

Administration Schemas

Information

No object selected

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db

Limit to 1000 rows

```
32  DELIMITER //
33  CREATE TRIGGER update_inventory
34  AFTER INSERT ON orders
35  FOR EACH ROW
36  BEGIN
37    -- Decrease inventory count when a new order is placed
38    UPDATE products
39    SET inventory_count = inventory_count - NEW.quantity
40    WHERE product_id = NEW.product_id;
41  END;
42  //
43  DELIMITER ;
44  INSERT INTO orders (order_id, product_id, quantity, order_date) VALUES (5, 1, 10, '2023-09-27');
45  SELECT inventory_count FROM products WHERE product_id = 1;
46  INSERT INTO orders (order_id, product_id, quantity, order_date) VALUES (6, 2, 11, '2023-09-27');
47  SELECT inventory_count FROM products WHERE product_id = 2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

inventory_count
64

Result Grid

Form Editor

products 15

Read Only

Output

USD/CAD +0.67%

Search

2:32 PM 10/1/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- bookstore
- cars
- hospital
  - Tables
  - Views
  - Stored Procedures
  - Functions
- librarydb
- mydb
- mydb2
- mydb3
- mydb4
- mydb5
- **mydb6**
  - Tables
  - Views

Administration Schemas

Information

No object selected

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db

Limit to 1000 rows

```
34  AFTER INSERT ON orders
35  FOR EACH ROW
36  BEGIN
37    -- Decrease inventory count when a new order is placed
38    UPDATE products
39    SET inventory_count = inventory_count - NEW.quantity
40    WHERE product_id = NEW.product_id;
41  END;
42  //
43  DELIMITER ;
44  INSERT INTO orders (order_id, product_id, quantity, order_date) VALUES (5, 1, 10, '2023-09-27');
45  SELECT inventory_count FROM products WHERE product_id = 1;
46  INSERT INTO orders (order_id, product_id, quantity, order_date) VALUES (6, 2, 11, '2023-09-27');
47  SELECT inventory_count FROM products WHERE product_id = 2;
48
49  select * from products;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	product_name	price	inventory_count
1	Product A	19.99	90
2	Product B	29.99	64
3	Product C	9.99	200
4	Product D	49.99	50
NULL	NULL	NULL	NULL

Result Grid | Form Editor

products 16 | Apply

Output

USD/CAD +0.67%

Search

2:33 PM 10/1/2023

Week-07

Aggregate functions in MySQL are used to perform calculations on a set of values and return a single value. They are particularly useful for generating summary statistics or performing calculations on grouped data. Some commonly used aggregate functions include COUNT, SUM, AVG, MIN, and MAX.

Example 1: Calculate the average age and the total number of students from the "students" table.

Example 2: Calculate the total sales and the highest sales amount from the "sales" table.

Example 3: Calculate the total number of employees in each department from the "employees" table.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- mydb2
- mydb3
- **mydb4**
  - Tables
  - Views
  - Stored Procedures
  - Functions
- northwind
- school
- students
- sys

Administration Schemas

Information

Table: students

Columns:

student_id	int PK
student_name	varchar(25)
class_id	int
Student_gender	char(10)

Object Info Session

mydb* SQL File 5* SQL File 6* mydb2* mydb4*

Limit to 1000 rows

```
1  create database mydb4;
2  use mydb4;
3  CREATE TABLE students (student_id INT PRIMARY KEY,
4      student_name VARCHAR(25),
5      class_id INT,
6      student_gender CHAR(10),
7      age INT);
8  INSERT INTO students (student_id, student_name, class_id, student_gender, age) VALUES (655, 'Zack', 1, 'Male', 20);
9  INSERT INTO students (student_id, student_name, class_id, student_gender, age) VALUES (658, 'Rose', 2, 'Female', 22);
10 INSERT INTO students (student_id, student_name, class_id, student_gender, age) VALUES (659, 'Alex', 3, 'Male', 21);
11 INSERT INTO students (student_id, student_name, class_id, student_gender, age) VALUES (660, 'Smith', 4, 'Male', 23);
12 INSERT INTO students (student_id, student_name, class_id, student_gender, age) VALUES (661, 'Maria', 5, 'Female', 20);
13 select avg(age) AS average_age, COUNT(*) AS total_students          # Calculate the average age and total number of students
14 FROM students;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

average_age	total_students
21.2000	5

Result Grid

Form Editor

Result 1

Read Only

Output

Action Output

SQL script saved to 'D:\DataScience\DataElement\mydb4.sql'

79°F Sunny

10:30 AM 9/18/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- mydb2
- mydb3
- **mydb4**
  - Tables
    - sales
    - students
  - Views
  - Stored Procedures
  - Functions
- northwind
- school
- students
- sys

Administration Schemas

Information

Schema: mydb4

Object Info Session

mydb* SQL File 5* SQL File 6* mydb2* mydb4*

Limit to 1000 rows

```
17  CREATE TABLE sales (                                    # Create the sales table
18      sale_id INT PRIMARY KEY,
19      sales_amount DECIMAL(10, 2)
20  );
21  INSERT INTO sales (sale_id, sales_amount) VALUES (01, 1000.50);     # Insert data into the sales table
22  INSERT INTO sales (sale_id, sales_amount) VALUES (02, 1500.25);
23  INSERT INTO sales (sale_id, sales_amount) VALUES (03, 2000.75);
24  INSERT INTO sales (sale_id, sales_amount) VALUES (04, 1800.00);
25  INSERT INTO sales (sale_id, sales_amount) VALUES (05, 1200.80);
26
27  SELECT SUM(sales_amount) AS total_sales, MAX(sales_amount) AS highest_sale      # Calculate the total sales and highest sales amount
28  FROM sales;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_sales	highest_sale
7502.30	2000.75

Result Grid

Form Editor

Result 2

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
16	10:37:52	INSERT INTO sales (sale_id, sales_amount) VALUES (04, 1800.00)	1 row(s) affected	0.297 sec

Query Completed

80°F High pollen

10:41 AM 9/18/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- mydb2
- mydb3
- **mydb4**
  - Tables
    - employees
    - sales
    - students
  - Views
  - Stored Procedures
  - Functions
- northwind
- school
- students
- sys

Administration Schemas

Information

Schema: mydb4

Object Info Session

mydb* SQL File 5* SQL File 6* mydb2* mydb4*

Limit to 1000 rows

```
31  CREATE TABLE employees (                          # Create the employees table
32      employee_id INT PRIMARY KEY,
33      employee_name VARCHAR(50),
34      department_id INT
35  );
36  INSERT INTO employees (employee_id, employee_name, department_id) VALUES (1, 'John', 1);        # Insert data into the employees table
37  INSERT INTO employees (employee_id, employee_name, department_id) VALUES (2, 'Sarah', 1);
38  INSERT INTO employees (employee_id, employee_name, department_id) VALUES (3, 'Michael', 2);
39  INSERT INTO employees (employee_id, employee_name, department_id) VALUES (4, 'Emily', 2);
40  INSERT INTO employees (employee_id, employee_name, department_id) VALUES (5, 'David', 3);
41
42  SELECT department_id, COUNT(*) AS total_employees   # Calculate the total number of employees in each department
43  FROM employees
44  GROUP BY department_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

department_id	total_employees
1	2
2	2
3	1

Result Grid

Result 4

Read Only

Output

Action Output

Query Completed

80°F Sunny

10:59 AM 9/18/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- mydb2
- mydb3
- mydb4
- **mydb5**
  - Tables
  - Views
  - Stored Procedures
  - Functions
- northwind
- school
- students
- sys

Administration Schemas

Information

Schema: mydb5

Object Info Session

mydb* SQL File 5* SQL File 6* mydb2* mydb4 SQL File 7*

Limit to 1000 rows

```
1  create database mydb5;
2  use mydb5;
3  CREATE TABLE employees (                          # Create the employees table
4      employee_id INT PRIMARY KEY,
5      employee_name VARCHAR(50),
6      job_title VARCHAR(50),
7      salary DECIMAL(10, 2));
8  INSERT INTO employees (employee_id, employee_name, job_title, salary) VALUES (1, 'John', 'Manager', 60000.00);
9  INSERT INTO employees (employee_id, employee_name, job_title, salary) VALUES (2, 'Sarah', 'Manager', 65000.00);
10 INSERT INTO employees (employee_id, employee_name, job_title, salary) VALUES (3, 'Michael', 'Analyst', 50000.00);
11 INSERT INTO employees (employee_id, employee_name, job_title, salary) VALUES (4, 'Emily', 'Analyst', 48000.00);
12 INSERT INTO employees (employee_id, employee_name, job_title, salary) VALUES (5, 'David', 'Clerk', 35000.00);
13
14 SELECT job_title, AVG(salary) AS average_salary, MIN(salary) AS minimum_salary   # Calculate the average & minimum salary for each job title
15 FROM employees
16 GROUP BY job_title;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

job_title	average_salary	minimum_salary
Manager	62500.000000	60000.00
Analyst	49000.000000	48000.00
Clerk	35000.000000	35000.00

Result 1

Read Only

Output

Query Completed

80°F Sunny

11:14 AM 9/18/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- mydb2
- mydb3
- mydb4
- **mydb5**
  - Tables
  - Views
  - Stored Procedures
  - Functions
- northwind
- school
- students
- sys

Administration Schemas

Information

Schema: mydb5

Object Info Session

mydb* SQL File 5* SQL File 6* mydb2* mydb4 mydb5*

Limit to 1000 rows

```
19  CREATE TABLE orders (                                   # Create the orders table
20      order_id INT PRIMARY KEY,
21      customer_id INT,
22      order_date DATE,
23      total_amount DECIMAL(10, 2)
24  );
25  INSERT INTO orders (order_id, customer_id, order_date, total_amount) VALUES (1, 101, '2023-09-15', 250.00);
26  INSERT INTO orders (order_id, customer_id, order_date, total_amount) VALUES (2, 102, '2023-09-16', 320.50);
27  INSERT INTO orders (order_id, customer_id, order_date, total_amount) VALUES (3, 103, '2023-09-17', 180.25);
28  INSERT INTO orders (order_id, customer_id, order_date, total_amount) VALUES (4, 104, '2023-09-18', 420.75);
29  INSERT INTO orders (order_id, customer_id, order_date, total_amount) VALUES (5, 105, '2023-09-19', 300.00);
30
31  SELECT SUM(total_amount) AS total_revenue, AVG(total_amount) AS average_order_value
32  FROM orders;                                            # Calculate the total revenue and average order value
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_revenue	average_order_value
1471.50	294.300000

Result Grid

Form Editor

Result 2

Read Only

Output

Action Output

Query Completed

85°F High pollen

11:20 AM 9/18/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- cars
- mydb
- mydb2
- mydb3
- mydb4
- **mydb5**
  - Tables
  - Views
  - Stored Procedures
  - Functions
- northwind
- school
- students
- sys

Administration Schemas

Information

Schema: mydb5

Object Info Session

mydb* SQL File 5* SQL File 6* mydb2* mydb4 mydb5*

Limit to 1000 rows

```
36  CREATE TABLE students (                          # Create the students table
37      student_id INT PRIMARY KEY,
38      student_name VARCHAR(50),
39      age INT
40  );
41  INSERT INTO students (student_id, student_name, age) VALUES (1, 'John', 20);
42  INSERT INTO students (student_id, student_name, age) VALUES (2, 'Sarah', 22);
43  INSERT INTO students (student_id, student_name, age) VALUES (3, 'Michael', 21);
44  INSERT INTO students (student_id, student_name, age) VALUES (4, 'Emily', 23);
45  INSERT INTO students (student_id, student_name, age) VALUES (5, 'David', 19);
46
47  SELECT MAX(age) AS highest_age, MIN(age) AS lowest_age          # Calculate the highest and lowest ages among the students
48  FROM students;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	highest_age	lowest_age
▶	23	19

Result Grid

Result 3

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
59	11:23:36	INSERT INTO students (student_id, student_name, age) VALUES (5, 'David', 19)	1 row(s) affected	0.032 sec

Query Completed

85°F Sunny

11:24 AM 9/18/2023

Week-08

Establishing a Database Connection using Python and MySQL

Python provides various libraries to connect and interact with MySQL databases. The `mysql.connector` module is one such library.

Example:

```
import mysql.connector

# Establish a connection
mydb = mysql.connector.connect(
  host="localhost",
  user="your_username",
  password="your_password",
  database="your_database"
)
# Create a cursor object
mycursor = mydb.cursor()
# Execute SQL queries
mycursor.execute("SELECT * FROM your_table")
result = mycursor.fetchall()
for row in result:
  print(row)

# Close the connection
mydb.close()
```

Note: Replace `localhost`, `your_username`, `your_password`, `your_database`, and `your_table` with the appropriate values for your MySQL setup

In [18]:
```
import mysql.connector

#connection parameters
host = "localhost"
user = "root"
password = "bidhan"
database = "University"

# Create a connection to MySQL server
conn = mysql.connector.connect(
    host=host,
    user=user,
    password=password,
    database=database
)

# Check if the connection was successful
if conn.is_connected():
    print("Connected to MySQL")
else:
    print("Connection to MySQL failed")

# Create a cursor object to interact with the database
cursor = conn.cursor()

cursor.execute("SELECT * FROM Students")
results = cursor.fetchall()

for row in results:
    print(row)

cursor.close()
conn.close()
```

```
Connected to MySQL
(1, 'Bidhan Ghosh', datetime.date(2000, 1, 1), 'bidhan@gmail.com')
(2, 'Jr. Zack Piterson', datetime.date(1998, 9, 9), 'zack@gmail.com')
(3, 'John Bell', datetime.date(1999, 7, 25), 'john@gmail.com')
(4, 'Steve johnson', datetime.date(1997, 5, 5), 'steve@gmail.com')
(5, 'Jackson Ganzales', datetime.date(1996, 6, 19), 'jackson@gmail.com')
```

In [46]:
```
conn = mysql.connector.connect(
    host=host,
    user=user,
    password=password,
    database=database
)

if conn.is_connected():
    cursor = conn.cursor()

    # Fetch student names. Emails and their emails
    cursor.execute("SELECT StudentName, Email, DOB FROM Students")
    Students_data = cursor.fetchall()

    for Students in Students_data:
        print(f"StudentName: {Students[0]}, Email: {Students[1]} , DOB: {Students[2

    cursor.close()
    conn.close()
else:
    print("Connection to MySQL failed")
```

```
StudentName: Bidhan Ghosh, Email: bidhan@gmail.com , DOB: 2000-01-01
StudentName: Jr. Zack Piterson, Email: zack@gmail.com , DOB: 1998-09-09
StudentName: John Bell, Email: john@gmail.com , DOB: 1999-07-25
StudentName: Steve johnson, Email: steve@gmail.com , DOB: 1997-05-05
StudentName: Jackson Ganzales, Email: jackson@gmail.com , DOB: 1996-06-19
```

In [50]:
```python
conn = mysql.connector.connect(
    host=host,
    user=user,
    password=password,
    database=database
)

if conn.is_connected():
    cursor = conn.cursor()

    # Fetch course names and their credits
    cursor.execute("SELECT CourseName, Credits FROM Courses")
    courses_data = cursor.fetchall()

    for course in courses_data:
        print(f"CourseName: {course[0]}, Credits: {course[1]}")

    cursor.close()
    conn.close()
else:
    print("Connection to MySQL failed")
```

```
CourseName: Programming Python, Credits: 4
CourseName: SQL, Credits: 4
CourseName: ML, Credits: 4
CourseName: AI, Credits: 4
CourseName: Data Visulization, Credits: 4
```

In [ ]:
```python

```

Assignment 1

Create a database named "Bookstore" and design the following tables:

Table: Books

Columns: BookID (int, primary key), Title (varchar), Author (varchar), Price (decimal)

Table: Customers

Columns: CustomerID (int, primary key), Name (varchar), Email (varchar), Phone (varchar)

Table: Orders

Columns: OrderID (int, primary key), CustomerID (int, foreign key), BookID (int, foreign key), Quantity (int), OrderDate (date)

1. Write the SQL statements to create the above tables and establish the necessary relationships between them.

2. Write a SQL query to retrieve the titles of all the books in the "Books" table whose price is greater than $50.

3. Write a SQL query to calculate the total quantity of books ordered by each customer. The result should display the customer's name and the total quantity ordered, sorted in descending order of the total quantity.

4. Write a SQL query to update the price of the book with BookID = 1001 to $29.99.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- **bookstore**
  - Tables
    - books
      - Columns
        - bookid
        - title
        - author
        - price
      - Indexes
      - Foreign Keys
      - Triggers
    - customers
    - orders
  - Views
  - Stored Procedures
  - Functions

Administration Schemas

Information

Table: customers

Columns:

customerID	int PK
customer_name	varchar(250)
email	varchar(250)
phone	varchar(250)

Object Info Session

mydb* SQL File 5* SQL File 6* mydb4 mydb5 mydb6*

Limit to 1000 rows

```
1                                                  # Assignment Week 05
2   create database bookstore;
3   use bookstore;
4   create table books(
5       bookid int primary key,
6       title varchar(250),
7       author varchar(250),
8       price decimal(10,2)
9    );
10  INSERT INTO books (bookid, title, author, price) VALUES (1, 'To Kill a Mockingbird', 'Harper Lee', 24.99);
11  INSERT INTO books (bookid, title, author, price) VALUES (2, '1984', 'George Orwell', 19.99);
12  INSERT INTO books (bookid, title, author, price) VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 18.50);
13  INSERT INTO books (bookid, title, author, price) VALUES (4, 'Pride and Prejudice', 'Jane Austen', 22.95);
14  INSERT INTO books (bookid, title, author, price) VALUES (5, 'Harry Potter and the Sorcerer''s Stone', 'J.K. Rowling', 29.99);
15  INSERT INTO books (bookid, title, author, price) VALUES (1001, 'Sample Book', 'Sample Author', 50.99);
16  select * from books;
17  # 2. Write a SQL query to retrieve the titles of all the books in the "Books" table whose price is greater than $50.
18  select title  from books where price>50.00;   # above 50
19  select title  from books where price>20.00;   # above 20.00
20  # 4.    Write a SQL query to update the price of the book with bookID = 1001 to $29.99.
21  update books set price = 29.99 where bookid = 1001 ;
22
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch

Query Completed

USD/CAD -0.33%

11:50 AM 9/20/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- **bookstore**
  - Tables
    - books
      - Columns
        - bookid
        - title
        - author
        - price
      - Indexes
      - Foreign Keys
      - Triggers
    - customers
    - orders
  - Views
  - Stored Procedures
  - Functions

Administration Schemas

Information

Table: customers

Columns:

customerID	int PK
customer_name	varchar(250)
email	varchar(250)
phone	varchar(250)

Object Info Session

mydb* SQL File 5* SQL File 6* mydb4 mydb5 mydb6*

Limit to 1000 rows

```
16 •  select * from books;
17    # 2. Write a SQL query to retrieve the titles of all the books in the "Books" table whose price is greater than $50.
18 •  select title  from books where price>50.00;   # above 50
19 •  select title  from books where price>20.00;   # above 20.00
20    # 4.    Write a SQL query to update the price of the book with bookID = 1001 to $29.99.
21 •  update books set price = 29.99 where bookid = 1001 ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title

Result Grid

Output

Action Output

#	Time	Action	Message	Duration / Fetch
80	11:32:30	select title from books where price>50.00 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
81	11:36:40	update books set price = 29.99 where bookid = 1001	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.109 sec
82	11:36:45	select * from books LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
83	11:38:38	select title from books where price>50.00 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
84	11:38:48	select title from books where price>50.00 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
85	11:38:48	select title from books where price>20.00 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
86	11:38:55	select title from books where price>50.00 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
87	11:46:55	select title from books where price>50.00 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
88	11:50:51	select title from books where price>50.00 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

Query Completed

88°F Partly sunny

11:51 AM 9/20/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- **bookstore**
  - Tables
    - books
      - Columns
        - bookid
        - title
        - author
        - price
      - Indexes
      - Foreign Keys
      - Triggers
    - customers
    - orders
  - Views
  - Stored Procedures
  - Functions

Administration Schemas

Information

Table: customers

Columns:

customerID	int PK
customer_name	varchar(250)
email	varchar(250)
phone	varchar(250)

Object Info Session

mydb* SQL File 5* SQL File 6* mydb4 mydb5 mydb6*

Limit to 1000 rows

```
10  INSERT INTO books (bookid, title, author, price) VALUES (1, 'To Kill a Mockingbird', 'Harper Lee', 24.99);
11  INSERT INTO books (bookid, title, author, price) VALUES (2, '1984', 'George Orwell', 19.99);
12  INSERT INTO books (bookid, title, author, price) VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 18.50);
13  INSERT INTO books (bookid, title, author, price) VALUES (4, 'Pride and Prejudice', 'Jane Austen', 22.95);
14  INSERT INTO books (bookid, title, author, price) VALUES (5, 'Harry Potter and the Sorcerer''s Stone', 'J.K. Rowling', 29.99);
15  INSERT INTO books (bookid, title, author, price) VALUES (1001, 'Sample Book', 'Sample Author', 50.99);
16  select * from books;
17  # 2. Write a SQL query to retrieve the titles of all the books in the "Books" table whose price is greater than $50.
18  select title  from books where price>50.00;   # above 50
19  select title  from books where price>20.00;   # above 20.00
20  # 4.    Write a SQL query to update the price of the book with bookID = 1001 to $29.99.
21  update books set price = 29.99 where bookid = 1001 ;
22
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

bookid	title	author	price
1	To Kill a Mockingbird	Harper Lee	24.99
2	1984	George Orwell	19.99
3	The Great Gatsby	F. Scott Fitzgerald	18.50
4	Pride and Prejudice	Jane Austen	22.95
5	Harry Potter and the Sorcerer's Stone	J.K. Rowling	29.99
1001	Sample Book	Sample Author	29.99
NULL	NULL	NULL	NULL

books 41

Result Grid Form Editor

Apply Revert

Output

Query Completed

88°F Partly sunny

11:55 AM 9/20/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- bookstore
- cars
- **hospital**
  - Tables
    - appointments
    - doctors
    - patients
  - Views
  - Stored Procedures
  - Functions
- mydb
- mydb2
- mydb3
- mydb4
- mydb5
- northwind

Administration Schemas

Information

No object selected

Object Info Session

mydb* SQL File 5* SQL File 6* mydb4 mydb5 mydb6* hospital*

Limit to 1000 rows

```
23  create table customers(
24      customerID int primary key,
25      customer_name varchar(250),
26      email varchar(250),
27      phone varchar(250));
28  INSERT INTO customers (customerID, customer_name, email, phone) VALUES (1, 'John Doe', 'johndoe@example.com', '+1 (123) 456-7890');
29  INSERT INTO customers (customerID, customer_name, email, phone) VALUES (2, 'Alice Smith', 'alice@example.com', '+1 (987) 654-3210');
30  INSERT INTO customers (customerID, customer_name, email, phone) VALUES (3, 'Bob Johnson', 'bob@example.com', '+1 (555) 123-4567');
31  INSERT INTO customers (customerID, customer_name, email, phone) VALUES (4, 'Emily Davis', 'emily@example.com', '+1 (333) 555-7777');
32  INSERT INTO customers (customerID, customer_name, email, phone) VALUES (5, 'Michael Wilson', 'michael@example.com', '+1 (777) 888-9999');
33  select *from customers;
34  create table orders (
35      orderID int primary key,
36      customerID int ,            # foreign key
37      bookID int,                 # foreign key
38      quantity int,
39      orderDate datetime,
40      FOREIGN KEY (customerID) REFERENCES customers(customerID),
41      FOREIGN KEY (bookID) REFERENCES books(bookID)
42      );
43  INSERT INTO orders (orderID, customerID, bookID, quantity, orderDate) VALUES (1, 1, 201, 3, '2023-09-20 09:30:00');
44  INSERT INTO orders (orderID, customerID, bookID, quantity, orderDate) VALUES (2, 2, 202, 2, '2023-09-21 14:45:00');
45  INSERT INTO orders (orderID, customerID, bookID, quantity, orderDate) VALUES (3, 3, 203, 1, '2023-09-22 11:15:00');
46  INSERT INTO orders (orderID, customerID, bookID, quantity, orderDate) VALUES (4, 4, 204, 4, '2023-09-23 16:00:00');
47  INSERT INTO orders (orderID, customerID, bookID, quantity, orderDate) VALUES (5, 5, 205, 2, '2023-09-24 10:30:00');
```

SQL script saved to 'D:\DataScience\DataElement\hospital.sql'

Search

9:54 AM
9/21/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- **bookstore**
  - Tables
    - books
      - Columns
        - bookid
        - title
        - author
        - price
      - Indexes
      - Foreign Keys
      - Triggers
    - customers
    - orders
  - Views
  - Stored Procedures
  - Functions

Administration Schemas

Information

Table: customers

Columns:

customerID	int PK
customer_name	varchar(250)
email	varchar(250)
phone	varchar(250)

Object Info Session

mydb* SQL File 5* SQL File 6* mydb4 mydb5 mydb6*

Limit to 1000 rows

```
46
47    # 3. Write a SQL query to calculate the total quantity of books ordered by each customer.
48    # The result should display the customer's name and the total quantity ordered, sorted in descending order of the total quantity.
49
50    SELECT customers.customer_name AS CustomerName, SUM(orders.quantity) AS TotalQuantity
51    FROM Customers
52    LEFT JOIN orders ON customers.customerID = orders.customerID
53    GROUP BY customers.customerID
54    ORDER BY TotalQuantity DESC;
55
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerName	TotalQuantity
Emily Davis	4
John Doe	3
Alice Smith	2
Michael Wilson	2
Bob Johnson	1

Result Grid Form Editor

Result 42 Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
90	11:54:20	select * from books LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
91	11:57:07	SELECT customers.customer_name AS CustomerName, SUM(orders.quantity) AS Total...	5 row(s) returned	0.000 sec / 0.000 sec

Query Completed

88°F Partly sunny

11:57 AM 9/20/2023

Assignment 2

Create a database named "Hospital" and design the following tables:

Table: Doctors

Columns: DoctorID (int, primary key), Name (varchar), Specialization (varchar), Experience (int)

Table: Patients

Columns: PatientID (int, primary key), Name (varchar), Age (int), Gender (varchar), Diagnosis (varchar)

Table: Appointments

Columns: AppointmentID (int, primary key), DoctorID (int, foreign key), PatientID (int, foreign key), AppointmentDate (date), StartTime (time), EndTime (time)

1. Write the SQL statements to create the above tables and establish the necessary relationships between them.

2. Write a SQL query to retrieve the names of all the doctors who specialize in Cardiology.

3. Write a SQL query to calculate the average age of all the patients.

4. Write a SQL query to delete all the appointments from the "Appointments" table for which the appointment date is in the past.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- bookstore
- cars
- **hospital**
  - Tables
  - Views
  - Stored Procedures
  - Functions
- mydb
- mydb2
  - Tables
  - Views
  - Stored Procedures
  - Functions
- mydb3
- mydb4
- mydb5

Administration Schemas

Information

Schema: bidhandb

Object Info Session

SQL File 9 | hospital*

Limit to 1000 rows

```
2  Create database hospital;
3  use hospital;
4  Create table doctors(
5  doctorID int primary key,
6  doctor_Name varchar(250),
7  specialization varchar(250),
8  experience int
9  );
10 INSERT INTO doctors (doctorID, doctor_Name, specialization, experience)
11 VALUES
12     (1, 'Dr. Smith', 'Cardiologist', 10),
13     (2, 'Dr. Johnson', 'Pediatrician', 8),
14     (3, 'Dr. Williams', 'Dermatologist', 15),
15     (4, 'Dr. Brown', 'Orthopedic Surgeon', 12);
16 select * from doctors;
17 # 2.    Write a SQL query to retrieve the names of all the doctors who specialize in Cardiology.
18 SELECT doctor_Name FROM doctors WHERE specialization = 'Cardiologist';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

doctor_Name
Dr. Smith

Result Grid

doctors 2

Read Only

Query Completed

96°F Sunny

6:51 PM 9/23/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- bookstore
- cars
- **hospital**
  - Tables
  - Views
  - Stored Procedures
  - Functions
- mydb
- mydb2
  - Tables
  - Views
  - Stored Procedures
  - Functions
- mydb3
- mydb4
- mydb5

Administration Schemas

Information

Schema: bidhandb

Object Info Session

SQL File 9 | hospital*

Limit to 1000 rows

```
20  Create table patients(
21   patientID int primary key,
22   patientName varchar(250),
23   age int,
24   gender varchar(250),
25   diagnosis varchar(250)
26   );
27  INSERT INTO patients (patientID, patientName, age, gender, diagnosis)
28  VALUES
29      (1, 'John Doe', 35, 'Male', 'Hypertension'),
30      (2, 'Jane Smith', 25, 'Female', 'Asthma'),
31      (3, 'Robert Johnson', 10, 'Male', 'Common Cold'),
32      (4, 'Emily Davis', 42, 'Female', 'Psoriasis');
33  select * from patients;
34  # 3.    Write a SQL query to calculate the average age of all the patients.
35  SELECT AVG(age) AS AverageAge FROM patients;
36
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AverageAge
28.0000

Result Grid

Result 3

Read Only

Query Completed

96°F Sunny

6:53 PM 9/23/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
- bookstore
- cars
- **hospital**
  - Tables
    - appointments
      - Columns
        - appointmentID
        - doctorID
        - patientID
        - appointmentDate
        - startTime
        - endTime
      - Indexes
      - Foreign Keys
      - Triggers
    - doctors

Administration Schemas

Information

No object selected

Object Info Session

SQL File 9 | hospital*

Limit to 1000 rows

```
37  Create table appointments(
38  appointmentID int primary key,
39  doctorID int,              # foreign key
40  patientID int,             # foreign key
41  appointmentDate date, startTime time,  endTime time,
42  FOREIGN KEY (doctorID) REFERENCES doctors(doctorID), FOREIGN KEY (patientID) REFERENCES patients(patientID));
43  INSERT INTO appointments (appointmentID, doctorID, patientID, appointmentDate, startTime, endTime)
44  VALUES
45      (1, 1, 1, '2023-09-23', '09:00:00', '09:30:00'),
46      (2, 2, 2, '2023-09-22', '14:30:00', '15:00:00'),
47      (3, 3, 3, '2023-09-25', '11:15:00', '11:45:00'),
48      (4, 4, 4, '2023-09-20', '10:00:00', '10:30:00');
49  select * from appointments;
50  DELETE FROM appointments # 4.Delete all the appointments from the "Appointments" table for which the appointment date is in the past.
51  WHERE appointmentDate < CURDATE() AND appointmentID IS NOT NULL;
52
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

appointmentID	doctorID	patientID	appointmentDate	startTime	endTime
1	1	1	2023-09-23	09:00:00	09:30:00
3	3	3	2023-09-25	11:15:00	11:45:00
NULL	NULL	NULL	NULL	NULL	NULL

Result Grid

appointments 13

Apply Revert

Output

Query Completed

94°F Sunset

7:26 PM 9/23/2023

BIRCHWOOD UNIVERSITY

Midterm Exam Question Paper

Course Name: Database Management System
Course Number: MDS510
Total Marks: 30
Note:

1. Each question carries 5 marks
2. Kindly upload solution on Moodle as .pdf file

Q1. Draw an ER diagram for a restaurant management system. Demonstrate a minimum of 4 entities, their relationships, cardinalities and specialization/generalization.

Q2. Describe the fundamental differences between Data Definition Language (DDL) and Data Manipulation Language (DML) commands. Provide examples of each type of command and explain their respective roles in database administration and data analysis.

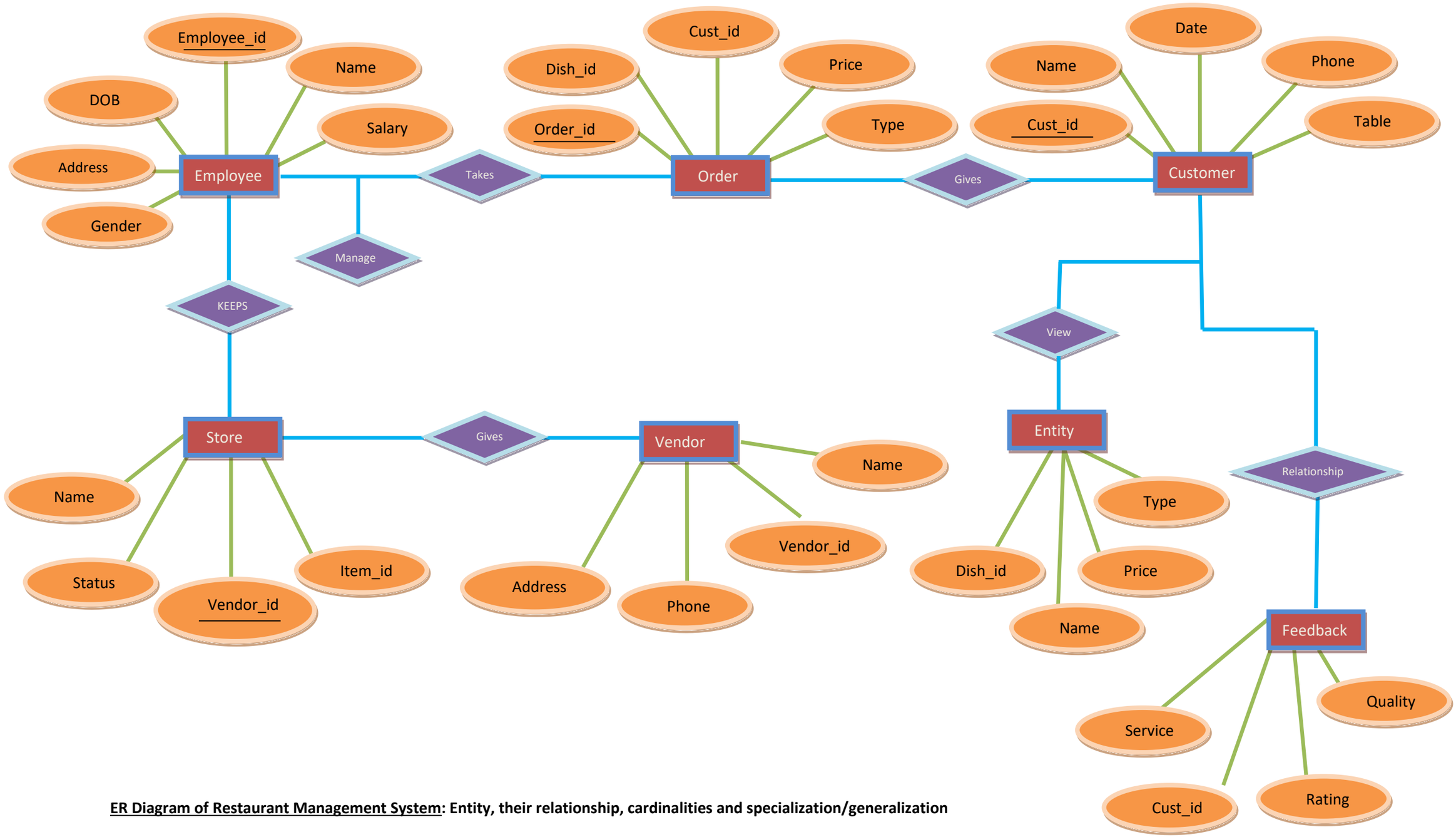
Q3. Consider two tables, "Orders" and "Customers," in a relational database. Write an SQL query to perform an inner join between these tables, retrieving information about customers who have placed orders.

Q4. Given a scenario where you have a MySQL database containing sales data, write an SQL query to calculate the total sales revenue for a specific product category within a specified time frame. Explain the logic behind your query.

Q5. Compare and contrast SQL and NoSQL databases in terms of their data models, use cases, and advantages and disadvantages for data science projects. Provide examples of scenarios where you would choose one type of database over the other, and discuss the key factors that influence your decision.

Q6. Consider a database containing information about employees in a company. The Employees table has columns employee_id, employee_name, salary, and department_id. Write an SQL query using the CASE statement to retrieve a list of employees with their names and a bonus amount based on the following criteria:

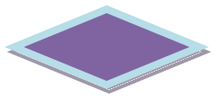
Employees in the "Sales" department with a salary less than $50,000 receive a bonus of $2,000. Employees in the "Marketing" department with a salary less than $55,000 receive a bonus of $1,500. All other employees receive a bonus of $1,000.



ER Diagram of Restaurant Management System: Entity, their relationship, cardinalities and specialization/generalization

Ellipses: represents attributes

Rectangles represent entity sets



Diamonds represent relationship sets

Lines link attributes to entity sets and entity sets to relationship set

Underline (_): indicates primary key attributes

Ellipses: Ellipses in the ER diagram represent attributes. These attributes are properties or characteristics of the entities (rectangles) in the diagram. Attributes are linked to their respective entity sets through lines.

Rectangles (Entity Sets): Rectangles represent entity sets, which are essentially tables in a database. Each rectangle represents a distinct entity or concept within the system, such as "Customer," "Employee," "Menu Item," or "Order."

Diamonds (Relationship Sets): Diamonds represent relationship sets between different entity sets. These relationships describe how the entities are connected or related to each other. For example, I might have a "Works_For" relationship between the "Employee" and "Restaurant" entity sets to indicate which employees work at which restaurants.

Lines: Lines are used to connect attributes to their respective entity sets and to connect entity sets to relationship sets. This visually shows how attributes are associated with entities and how entities are related to each other through relationships.

Underline (_): An underline is used to indicate primary key attributes within an entity set. A primary key is a unique identifier for each record (row) in a table (entity set). For example, in the "Customer" entity set, there might be a "Customer_ID" attribute underlined to indicate that it is the primary key.

Cardinalities: Cardinalities represent the numerical relationships between entities in a relationship set. They describe how many instances of one entity are related to how many instances of another entity through a particular relationship. Common cardinalities include "1" (one), "M" (many), "0" (zero), or "N" (variable number).

Specialization/Generalization: Specialization and generalization represent inheritance relationships between entity sets. This is used to model entities that share common attributes and behaviors but also have their unique attributes and behaviors. For example, i might have a general entity set called "Employee," and then I can specialize it into "Waiter," "Chef," and "Manager" entity sets, each with its own attributes and relationships.

In summary, an ER diagram is a visual representation of a database schema, showing entities, their attributes, relationships, cardinalities, and specialization/generalization relationships. It's a crucial step in database design to help developers and database administrators understand and plan the structure of the database system.

Q2. Describe the fundamental differences between Data Definition Language (DDL) and Data Manipulation Language (DML) commands. Provide examples of each type of command and explain their respective roles in database administration and data analysis.

……..

Data Definition Language (DDL) and Data Manipulation Language (DML) are two types of SQL (Structured Query Language) commands used in database management systems (DBMS). They serve distinct purposes and have fundamental differences:

Principle:

DDL (Data Definition Language): DDL commands are used for defining, managing, and modifying the structure and schema of the database. They are responsible for creating, altering, and dropping database objects such as tables, indexes, views, and constraints. DDL commands focus on the database's structure.

Examples of DDL commands:

CREATE TABLE: Creates a new table in the database.

ALTER TABLE: Modifies an existing table.

DROP TABLE: Deletes an existing table.

CREATE INDEX: Creates an index on a table to improve the performance of queries.

CREATE VIEW: Creates a virtual table that is based on one or more existing tables.

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Salary DECIMAL(10, 2)
);
ALTER TABLE Employees
ADD Email VARCHAR(100);
DROP TABLE Employees;
```

DML (Data Manipulation Language): DML commands are used for interacting with and manipulating the data within the database. They perform operations such as inserting, updating, deleting, and retrieving data from database tables. DML commands focus on the data within the database.

Examples of DML commands:

INSERT INTO: Inserts new data into a table.

UPDATE: Updates existing data in a table.

DELETE FROM: Deletes data from a table.

SELECT: Retrieves data from a table.

MERGE: Combines data from two tables into a single table.

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary)
VALUES (1, Michel, 'Doe', 90000);

UPDATE Employees
SET Salary = 65000
WHERE EmployeeID = 1;

SELECT FirstName, LastName
FROM Employees
WHERE Salary > 60000;
```

Database administration:

- A DBA might use the CREATE TABLE command to create a new table to store customer data.
- A DBA might use the ALTER TABLE command to add a new column to the customer table to store the customer's email address.
- A DBA might use the DROP TABLE command to delete the customer table if it is no longer needed.

Data analysis:

- A data analyst might use the INSERT INTO command to load a new dataset of customer data into the database.
- A data analyst might use the UPDATE command to update the customer's email address in the database.
- A data analyst might use the DELETE FROM command to delete a customer record from the database.
- A data analyst might use the SELECT command to generate a report of all customers who have purchased a product in the past year.
- A data analyst might use the MERGE command to combine two tables of customer data into a single table.

DDL and DML commands are essential for managing and analyzing data in a database. DDL commands are used to define the structure of the database, while DML commands are used to manipulate the data within the database. By understanding the differences between DDL and DML commands, database administrators and data analysts can effectively manage and analyze their data.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- cars
- hospital
- librarydb
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions

Administration Schemas

Information

Schema: mydb2

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db Establishing a Database Conne... mydb2*

Limit to 1000 rows

```
1   # Q3. Consider two tables, "Orders" and "Customers," in a relational database.
2   # Write an SQL query to perform an inner join between these tables, retrieving information about customers who have placed orders.
3
4   create database mydb2;
5   use mydb2;
6   create table customers                          # customer table
7   (customer_id int not null primary key,          # primary key cutomer_id
8   customer_name varchar(50),
9   age int,
10  address varchar(250) not null,
11  zip_code varchar(20)
12  );
13  select *from customers;                         # retrive all the records and inserting values
14  insert into customers (customer_id,customer_name,age,address,zip_code) values (1, "Bidhan", 100, "pflugerville", 78660);
15  insert into customers (customer_id,customer_name,age,address,zip_code) values (2, "A", 10, "Austin", 456678);
16  insert into customers (customer_id,customer_name,age,address,zip_code) values (3, "B", 100, "roundrock", 46789);
17  insert into customers (customer_id,customer_name,age,address,zip_code) values (4, "C", 200, "georgetown", 98675);
18  insert into customers (customer_id,customer_name,age,address,zip_code) values (5, "ABC", 500, "San-antonio", 78645);
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	01:04:49	use mydb2	0 row(s) affected	0.000 sec
2	01:05:00	SELECT orders.order_id, customers.customer_name FROM orders INNER JOIN custome...	5 row(s) returned	1.125 sec / 0.000 sec

Query Completed

Search

1:09 AM
10/3/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- cars
- hospital
- librarydb
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - orders
    - products
    - students
    - teachers
  - Views
  - Stored Procedures
  - Functions

Administration Schemas

Information

Schema: mydb2

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db Establishing a Database Conne... mydb2*

Limit to 1000 rows

```
17  create table orders                                  # order table
18  (order_id int primary key,                           # primary key order_id
19  customer_id int,
20  order_date datetime not null,
21  total_price decimal(10,2),
22  foreign key(customer_id) references customers(customer_id)  # foreign key customer_id here
23  );
24
25  -- Select all rows from the "orders" table
26  SELECT * FROM orders;
27
28  -- Insert data into the "orders" table
29  INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES (101, 1, '2023-10-01', 50000);
30  INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES (102, 2, '2022-10-11', 340000);
31  INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES (103, 3, '2021-10-08', 40000);
32  INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES (104, 4, '2020-06-10', 870000);
33  INSERT INTO orders (order_id, customer_id, order_date, total_price) VALUES (105, 5, '2023-09-10', 27000);
34
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	01:04:49	use mydb2	0 row(s) affected	0.000 sec
2	01:05:00	SELECT orders.order_id, customers.customer_name FROM orders INNER JOIN custome...	5 row(s) returned	1.125 sec / 0.000 sec

Query Completed

Search

1:11 AM
10/3/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bidhandb
  - Tables
  - Views
  - Stored Procedures
  - Functions
- bookstore
- cars
- hospital
- librarydb
- mydb
- **mydb2**
  - Tables
    - categories
    - customers
    - departments
    - orders
    - products
    - students

Administration Schemas

Information

Schema: mydb2

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db Establishing a Database Conne... mydb2*

Limit to 1000 rows

```
35  -- Inner join between orders and customers tables on the customer_id column
36  SELECT orders.order_id, customers.customer_name
37  FROM orders
38  INNER JOIN customers ON orders.customer_id = customers.customer_id;
39
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_name
101	Bidhan
102	A
103	B
104	C
105	ABC

Result 2

Result Grid Form Editor Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	01:04:49	use mydb2	0 row(s) affected	0.000 sec
2	01:05:00	SELECT orders.order_id, customers.customer_name FROM orders INNER JOIN custome...	5 row(s) returned	1.125 sec / 0.000 sec
3	01:13:31	SELECT orders.order_id, customers.customer_name FROM orders INNER JOIN custome...	5 row(s) returned	0.000 sec / 0.000 sec

Query Completed

Search

1:15 AM
10/3/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- Stored Procedures
- Functions
- librarydb
- **midterm_sql**
  - Tables
    - sales
      - Columns
        - sale_id
        - product_id
        - category
        - sale_date
        - revenue
      - Indexes
      - Foreign Keys
      - Triggers
  - Views
  - Stored Procedures
  - Functions

Administration Schemas

Information

No object selected

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db Establishing a Database Conne... mydb2* midterm_sql

Limit to 1000 rows

```
1   #Q4. Given a scenario where you have a MySQL database containing sales data,
2   # write an SQLquery to calculate the total sales revenue for a specific product category within a specified time frame.
3   # Explain the logic behind your query.
4   create database midterm_sql;
5   use midterm_sql;
6
7   CREATE TABLE sales (
8       sale_id INT AUTO_INCREMENT PRIMARY KEY,
9       product_id INT NOT NULL,
10      category VARCHAR(255) NOT NULL,
11      sale_date DATE NOT NULL,
12      revenue DECIMAL(10, 2) NOT NULL
13  );
14  INSERT INTO sales (product_id, category, sale_date, revenue) VALUES
15      (1, 'Electronics', '2023-01-05', 1500.00),
16      (2, 'Clothing', '2023-02-06', 750.50),
17      (3, 'Electronics', '2023-03-07', 2000.75),
18      (4, 'Furniture', '2023-04-08', 1200.25),
19      (5, 'Clothing', '2023-05-09', 900.90),
20      (6, 'Furniture', '2023-04-10', 1500.50);
21
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	01:49:40	create database midterm_sql	1 row(s) affected	0.860 sec

SQL script saved to 'D:\DataScience\DataElement\midterm_sql.sql'

77°F Mostly cloudy

Search

2:06 AM 10/4/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- Stored Procedures
- Functions
- librarydb
- **midterm_sql**
  - Tables
    - sales
      - Columns
        - sale_id
        - product_id
        - category
        - sale_date
        - revenue
      - Indexes
      - Foreign Keys
      - Triggers
  - Views
  - Stored Procedures
  - Functions

Administration Schemas

Information

No object selected

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db Establishing a Database Conne... mydb2* midterm_sql

Limit to 1000 rows

```
21
22  SELECT
23      category,
24      SUM(revenue) AS total_sales_revenue
25  FROM
26      sales
27  WHERE
28      category = 'Electronics'
29      AND sale_date BETWEEN '2023-01-01' AND '2023-03-31';
30
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	total_sales_revenue
Electronics	3500.75

Result Grid

Result 2

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
3	01:49:52	CREATE TABLE sales (sale_id INT AUTO_INCREMENT PRIMARY KEY, produc...	0 row(s) affected	1.938 sec
4	01:50:04	INSERT INTO sales (product_id, category, sale_date, revenue) VALUES (1, 'Electro...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.281 sec
5	01:55:22	SELECT category, SUM(revenue) AS total_sales_revenue FROM sales WHER...	1 row(s) returned	0.031 sec / 0.000 sec
6	02:08:24	SELECT category, SUM(revenue) AS total_sales_revenue FROM sales WHER...	1 row(s) returned	0.000 sec / 0.000 sec

Query Completed

77°F Mostly cloudy

Search

2:08 AM 10/4/2023

Explanation/logic of the query:

SELECT: This clause specifies the columns i want to include in the result set.
In this case, we want to retrieve the category and the sum of revenue.

FROM: Specifies the table from which i am retrieving data.

WHERE: This clause filters the rows from the sales table. I include conditions for two aspects:

category = 'Electronics': This condition ensures that only rows with the specified product category
(in this case, 'Electronics') are considered. sale_date BETWEEN '2023-01-01' AND '2023-03-31':
This condition filters the rows to include only those with sale
dates falling within the specified time frame ('2023-01-01' to '2023-03-31').

GROUP BY: Since we want to calculate the total sales revenue for a specific category, we don't use
a GROUP BY clause in this case because we are interested in a single category. However, if i want to
calculate the total sales revenue for multiple categories, i can include a GROUP BY clause for the category column.

SUM(revenue) AS total_sales_revenue: This part of the query calculates the sum of the revenue column for the
specified category and time frame. The result is given the alias total_sales_revenue, which makes it easy to
reference the calculated total sales revenue in the result set.

When i execute this query with the appropriate product category and time frame values, it returns the total
sales revenue for the specified product category within the given time frame.

Q5. Compare and contrast SQL and NoSQL databases in terms of their data models, use cases, and advantages and disadvantages for data science projects. Provide examples of scenarios where you would choose one type of database over the other, and discuss the key factors that influence your decision.

SQL (Structured Query Language) and NoSQL (Not Only SQL) databases are two different types of database management systems, each with its own data models, use cases, and advantages/disadvantages for data science projects. Let's compare and contrast them:

Data Models:

SQL: SQL databases are relational databases, which mean they use a structured and tabular format to store data. Data is organized into tables with predefined schemas, and relationships between tables are established using foreign keys. SQL databases support ACID (Atomicity, Consistency, Isolation, and Durability) transactions, ensuring data consistency and reliability.

NoSQL: NoSQL databases encompass various data models, including document-based, key-value, column-family, and graph databases. Data is stored in a more flexible, schema-less or schema-light manner, allowing for dynamic and unstructured data. NoSQL databases often sacrifice some ACID properties in favor of scalability and performance.

Use Cases:

SQL: Best suited for structured data with well-defined schemas, such as financial data, e-commerce transactions, and relational data.

Excellent for applications that require complex querying, reporting, and data integrity.

NoSQL: Ideal for handling unstructured or semi-structured data, such as JSON or XML documents, user-generated content, and sensor data.

Suitable for applications that require high scalability, fast read/write operations, and flexible data modeling, such as social media platforms and content management systems.

Advantages and disadvantages for data science projects

SQL databases

Advantages:

Well-established and mature technology

Strong query language (SQL)

ACID compliance for transactional data

Wide range of tools and libraries available

***Disadvantages*:**

Not as flexible as NoSQL databases

Can be difficult to scale to large datasets

Not as well-suited for unstructured data

NoSQL databases

Advantages:

More flexible than SQL databases

Easier to scale to large datasets

Well-suited for unstructured data

Disadvantages:

Not as mature as SQL databases

Query languages can be less powerful

ACID compliance may not be guaranteed

Fewer tools and libraries available

Overall

SQL databases are a good choice for data science projects that involve structured data and transactional applications.

NoSQL databases are a good choice for data science projects that involve unstructured data, high scalability, or real-time analytics.

Scenarios where I would choose a SQL database:

Storing and managing structured data: SQL databases are well-suited for storing and managing structured data, such as customer records, product catalogs, and financial transactions. This is because SQL databases use a relational data model, which is well-suited for organizing and querying structured data.

Transactional applications: SQL databases are also well-suited for transactional applications, where data needs to be updated and retrieved in a consistent manner. This is because SQL databases support ACID transactions, which guarantee that data is written to the database in a consistent manner.

Projects where I need a wide range of tools and libraries: SQL databases have a wide range of tools and libraries available, which can make them easier to use for data science projects.

Scenarios where I would choose a NoSQL database:

Storing and managing unstructured data: NoSQL databases are well-suited for storing and managing unstructured data, such as social media posts, sensor data, and clickstream data. This is because NoSQL databases use a variety of data models that are more flexible than the relational data model used by SQL databases.

Projects that require high scalability and availability: NoSQL databases are also well-suited for projects that require high scalability and availability. This is because NoSQL databases can be scaled horizontally by adding more servers.

Projects that require real-time analytics: NoSQL databases can be used to collect and analyze streaming data in real time. This makes them well-suited for projects that require real-time analytics, such as fraud detection and anomaly detection.

Key factors that influence my decision:

The type of data I need to store and manage: If I need to store and manage structured data, I will choose a SQL database. If I need to store and manage unstructured data, I will choose a NoSQL database.

The type of application I am developing: If I am developing a transactional application, I will choose a SQL database. If I am developing an application that requires high scalability and availability, or real-time analytics, I will choose a NoSQL database.

My existing skills and experience: If I am more familiar with SQL, I will choose a SQL database. If I am more familiar with NoSQL, I will choose a NoSQL database.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- Stored Procedures
- Functions
- librarydb
- midterm_sql
  - Tables
  - Views
  - Stored Procedures
  - Functions
- mydb
- mydb2
- mydb3
- mydb4
- mydb5
- mydb6
- northwind
- school
- students
- sys

Administration Schemas

Information

Schema: midterm_sql

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db Establishing a Database Conne... mydb2* midterm_sql*

Limit to 1000 rows

```
34  # Q6. Consider a database containing information about employees in a company. The Employees table has columns
35  # employee_id, employee_name, salary, and department_id. Write an SQL query using the CASE statement to retrieve
36  # a list of employees with their names and a bonus amount based on the following criteria:
37  # Employees in the "Sales" department with a salary less than $50,000 receive a bonus of $2,000.
38  # Employees in the "Marketing" department with a salary less than $55,000 receive a bonus of $1,500.
39  # All other employees receive a bonus of $1,000.
40
41  CREATE TABLE Employees (
42      employee_id INT PRIMARY KEY,
43      employee_name VARCHAR(255),
44      salary DECIMAL(10, 2),
45      department_id VARCHAR(255)
46  );
47  INSERT INTO Employees (employee_id, employee_name, salary, department_id)
48  VALUES
49      (1, 'John Doe', 45000.00, 'Sales'),
50      (2, 'Jane Smith', 60000.00, 'Marketing'),
51      (3, 'Mike Johnson', 55000.00, 'Sales'),
52      (4, 'Mary Brown', 48000.00, 'Marketing'),
53      (5, 'David Lee', 52000.00, 'IT'),
54      (6, 'Linda Wilson', 60000.00, 'HR'),
55      (7, 'Chris Evans', 42000.00, 'Sales');
```

Output

Action Output

Query Completed

70°F Heavy rain

Search

2:03 AM 10/5/2023

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- Stored Procedures
- Functions
- librarydb
- midterm_sql
  - Tables
  - Views
  - Stored Procedures
  - Functions
- mydb
- mydb2
- mydb3
- mydb4
- mydb5
- mydb6
- northwind
- school
- students
- sys

Administration Schemas

Information

Schema: midterm_sql

Object Info Session

view-trigger* school SQL File 4 mydb6* library-db Establishing a Database Conne... mydb2* midterm_sql*

Limit to 1000 rows

```
56
57  SELECT
58      employee_name,
59      CASE
60          WHEN department_id = 'Sales' AND salary < 50000 THEN 2000
61          WHEN department_id = 'Marketing' AND salary < 55000 THEN 1500
62          ELSE 1000
63      END AS bonus_amount
64  FROM Employees;
65
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee_name	bonus_amount
John Doe	2000
Jane Smith	1000
Mike Johnson	1000
Mary Brown	1500
David Lee	1000
Linda Wilson	1000
Chris Evans	2000

Result Grid Form Editor Field Types

Result 2 Read Only

Output

Action Output

Query Completed

70°F Heavy rain

Search

2:00 AM 10/5/2023

BIRCHWOOD UNIVERSITY

Final Exam Question Paper

Course Name: DBMS **Course Number:** MDS510

Total Marks: 30

Note:

1. Every question carries 5 marks.

2. Upload ER diagram, SQL scripts, output screenshot in one .pdf file on Moodle

Consider a database system for a university that needs to manage information about students, courses, and enrolments.

Q1. Draw an ER model that includes entities, relationships, attributes, and cardinalities to represent the following information:

Students (Attributes: StudentID, Name, DOB, Email)

Courses (Attributes: CourseID, CourseName, Credits)

Enrollments (Attributes: EnrollmentID, StudentID, CourseID, EnrollmentDate)

Q2. Based on your ER model, create the corresponding relational tables and insert values using MySQL Workbench. Include primary keys, foreign keys, and appropriate data types.

Insert at least 5 sample records into each of the tables (Students, Courses, Enrolments).

Update the name of a student in the Students table and a course name in the Courses table.

Delete a specific enrolment record from the Enrolments table.

Write a SQL query to retrieve the names of students who are enrolled in a specific course.

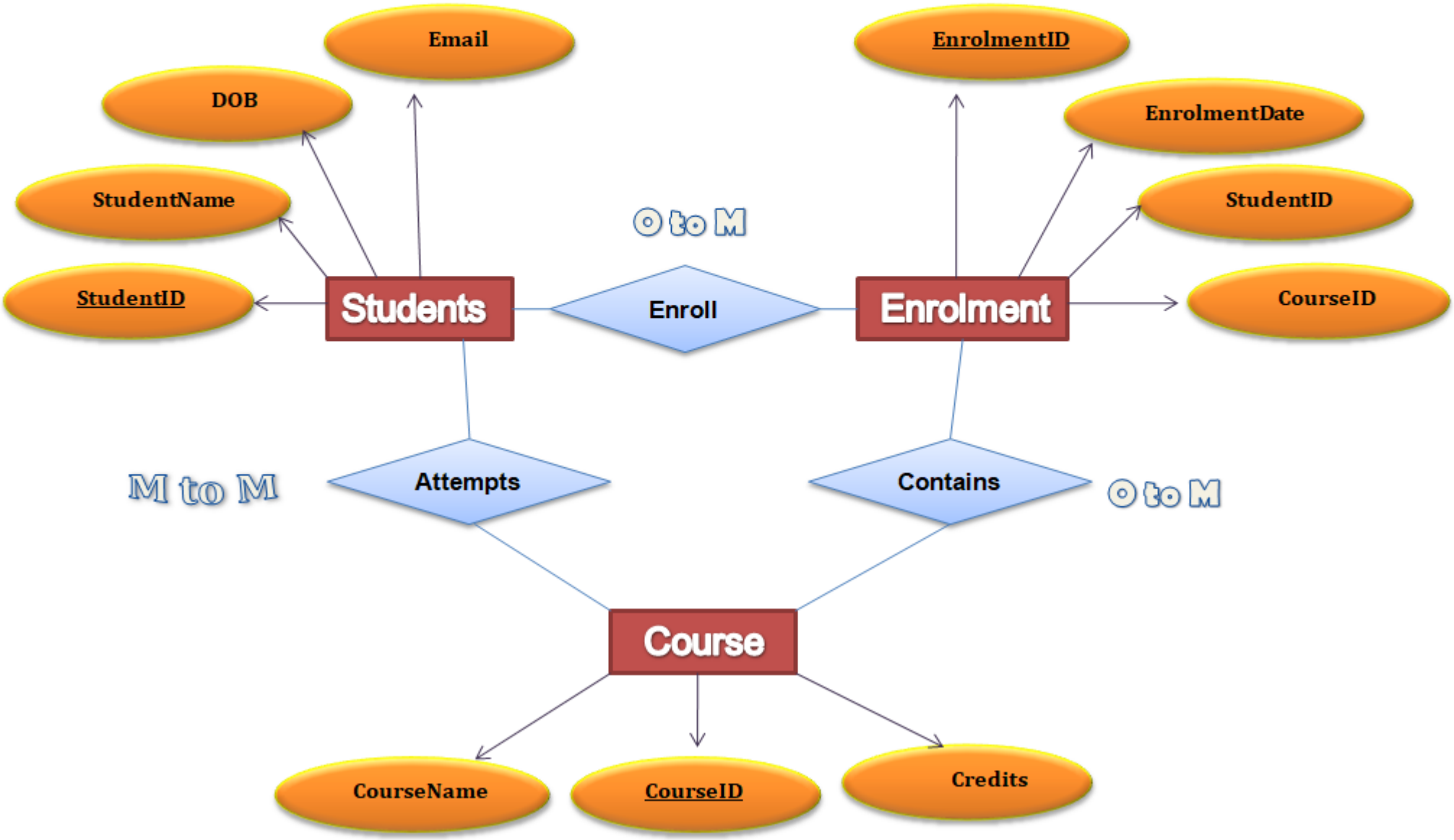
Q3. Write a SQL query to calculate the total number of credits taken by a specific student.

Q4. Write a SQL query to retrieve the courses with more than 3 credits.

Q5. Write a SQL query that joins the Students, Enrolments, and Courses tables to retrieve the names of students and the courses they are enrolled in.

Q6. Write a SQL query to create a report that displays the student names and a custom column "Status" indicating "Full-time" for students enrolled in more than 7 credits and "Part-time" for others.

Q1. Draw an ER model that includes entities, relationships, attributes, and cardinalities to represent the following information: Students (Attributes: StudentID, Name, DOB, Email) Courses (Attributes: CourseID, CourseName, Credits) Enrollments (Attributes: EnrollmentID, StudentID, CourseID, EnrollmentDate)



Entities: Student, Course, Enrollment

Attributes:
Student Entity Attributes:

StudentID (Primary Key),Name, DOB , Email

Course Entity Attributes: CourseID (Primary Key) CourseName , Credits

Enrollment Entity Attributes: EnrollmentID (Primary Key), StudentID (Foreign Key referencing Student), CourseID (Foreign Key referencing Course), EnrollmentDate

Relationships: " Attempts" Relationship between Student and Course (Many-to-Many):

Each student can enroll in multiple courses. Each course can have multiple students enrolled.

"Enroll" Relationship between Student and Enrollment (One-to-Many):

Each student can have multiple enrollments. Each enrollment corresponds to one student.

" Contains " Relationship between Course and Enrollment (One-to-Many):

Each course can have multiple enrollments. Each enrollment corresponds to one course.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- university
  - Tables
    - courses
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - enrolment
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - students
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
  - Views

Administration Schemas

Information

Schema: university

view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql*

Limit to 1000 rows

```
1  # Q2. Based on your ER model, create the corresponding relational tables and insert values using MySQL Workbench. Include primary keys, foreign keys, and appropriate
2  # data types. Insert at least 5 sample records into each of the tables (Students, Courses, Enrolments). Update the name of a student in the Students table and a course
3  # name in the Courses table.Delete a specific enrolment record from the Enrolments table.Write query to retrieve d names of students who r enrolled in a specific course
4  create database University;
5  use University;
6  create table Students
7  (StudentID int primary key,
8  StudentName varchar(25),
9  DOB DATE,
10 Email varchar(25));
11 INSERT INTO Students (StudentID, StudentName, DOB, Email)
12 VALUES (1, "Bidhan Ghosh", '2000-01-01', 'bidhan@gmail.com'),
13        (2, "Zack Piterson", '1998-09-09', 'zack@gmail.com'),
14        (3, "John Bell", '1999-07-25', 'john@gmail.com'),
15        (4, "Steve johnson", '1997-05-05', 'steve@gmail.com'),
16        (5, "Jackson Ganzales", '1996-06-19', 'jackson@gmail.com');
17 select * from Students;
```

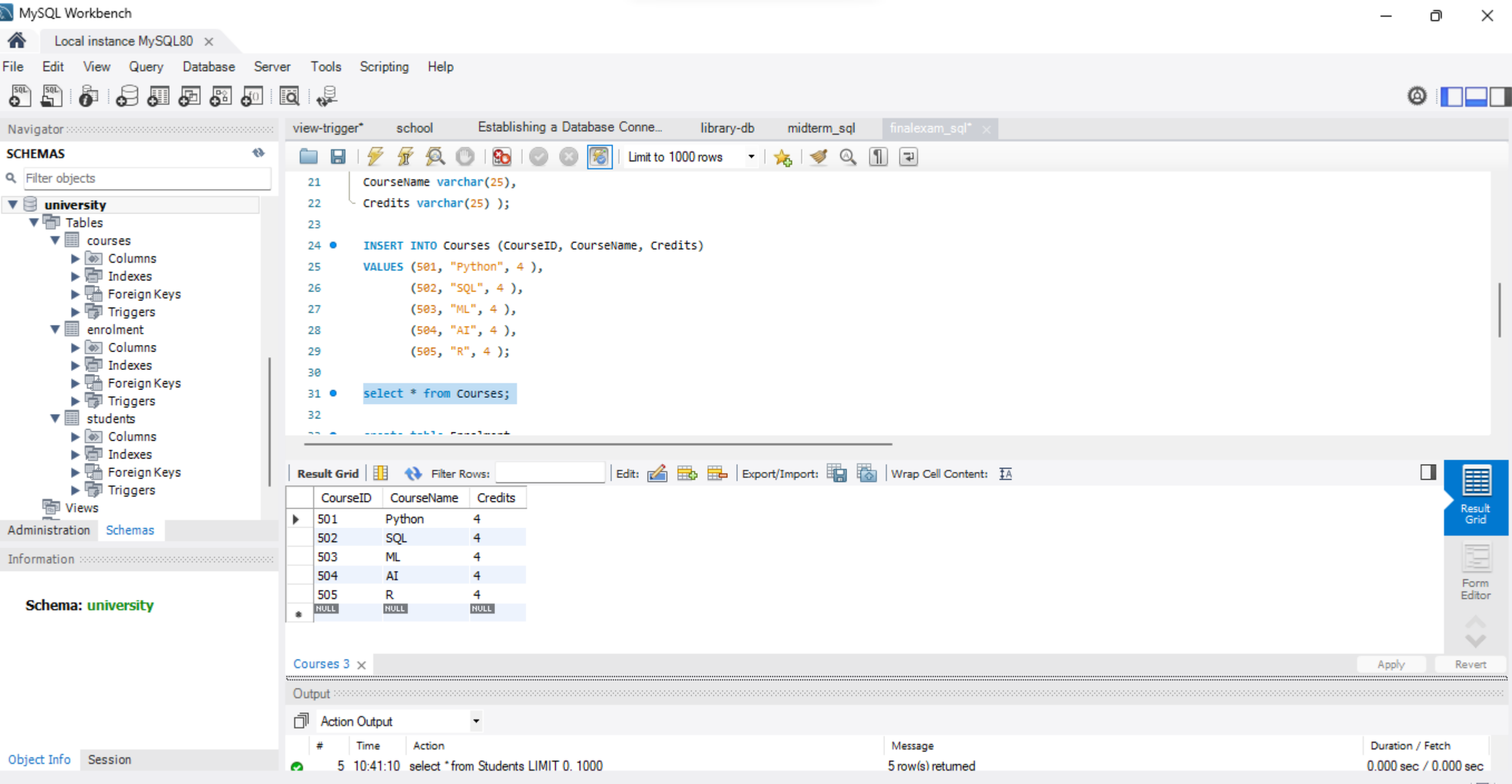
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

StudentID	StudentName	DOB	Email
1	Bidhan Ghosh	2000-01-01	bidhan@gmail.com
2	Zack Piterson	1998-09-09	zack@gmail.com
3	John Bell	1999-07-25	john@gmail.com
4	Steve johnson	1997-05-05	steve@gmail.com
5	Jackson Ganzales	1996-06-19	jackson@gmail.com

Result Grid

Students 2

Apply Revert


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
university
Tables
courses
Columns
Indexes
Foreign Keys
Triggers
enrolment
Columns
Indexes
Foreign Keys
Triggers
students
Columns
Indexes
Foreign Keys
Triggers
Views
Administration Schemas
Information
Schema: university
Object Info Session
view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql*
Limit to 1000 rows
21 CourseName varchar(25),
22 Credits varchar(25));
23
24 INSERT INTO Courses (CourseID, CourseName, Credits)
25 VALUES (501, "Python", 4),
26 (502, "SQL", 4),
27 (503, "ML", 4),
28 (504, "AI", 4),
29 (505, "R", 4);
30
31 select * from Courses;
32
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
CourseID CourseName Credits
501 Python 4
502 SQL 4
503 ML 4
504 AI 4
505 R 4
NULL NULL NULL
Result Grid
Form Editor
Courses 3
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
5 10:41:10 select * from Students LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- university
  - Tables
    - courses
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - enrolment
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - students
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
  - Views

Administration Schemas

Information

Schema: university

view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql*

Limit to 1000 rows

```
32
33 •  create table Enrolment
34    (EnrolmentID int primary key,
35    EnrolmentDate varchar(25),
36    StudentID int,
37    CourseID int,
38    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
39    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID));
40
41 •  INSERT INTO Enrolment (EnrolmentID, EnrolmentDate, StudentID, CourseID)
42    VALUES (7799, '2023-07-20', 1, 501 ),
43           (7791, '2023-08-27', 2, 502),
44           (7792, '2023-09-23', 3, 503 ),
45           (7793, '2023-06-21', 4, 504 ),
46           (7798, '2023-07-19', 5, 504 );
47 •  select * from Enrolment;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

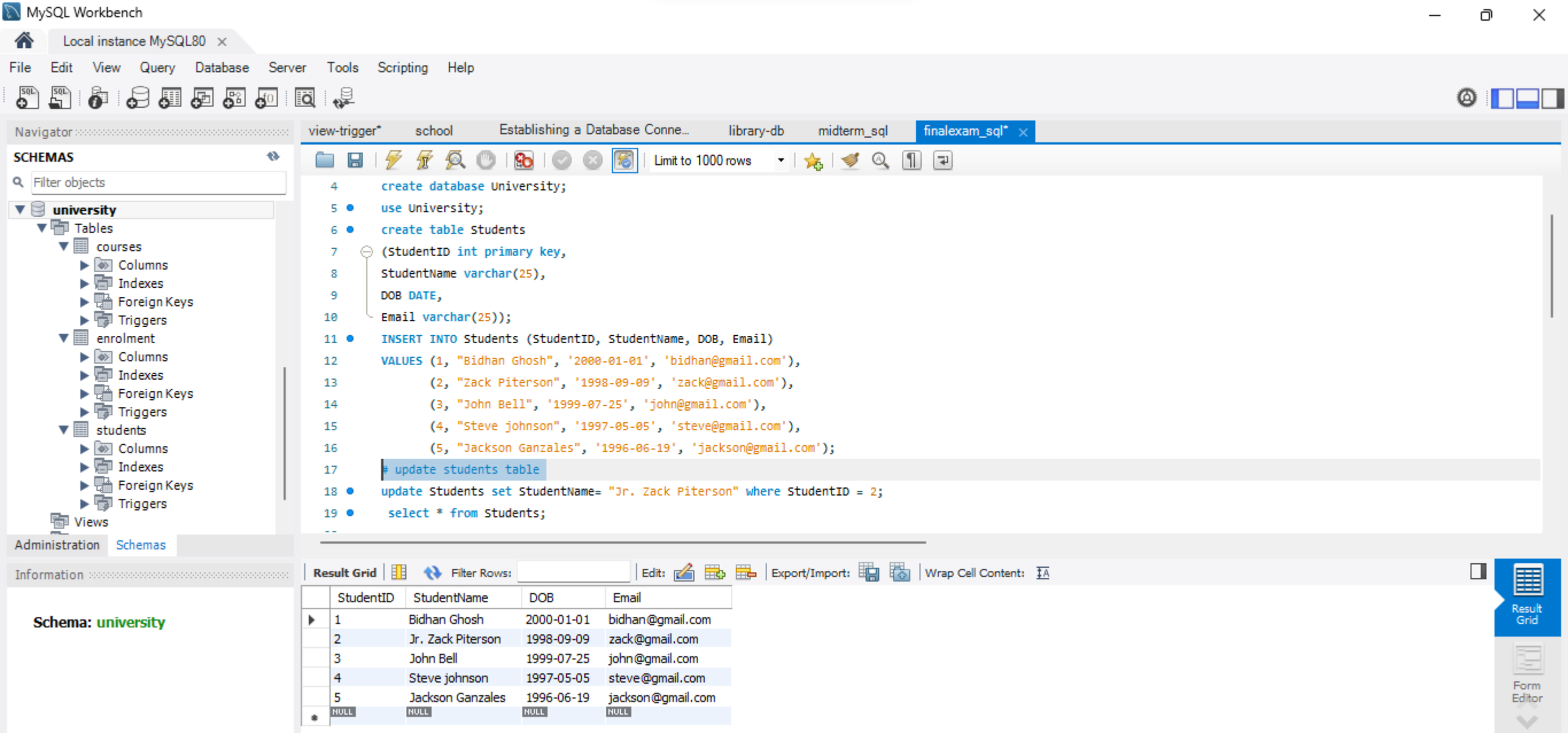
EnrolmentID	EnrolmentDate	StudentID	CourseID
7791	2023-08-27	2	502
7792	2023-09-23	3	503
7793	2023-06-21	4	504
7798	2023-07-19	5	504
7799	2023-07-20	1	501
NULL	NULL	NULL	NULL

Result Grid

Form Editor

Enrolment 4

Apply Revert


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
university
Tables
courses
Columns
Indexes
Foreign Keys
Triggers
enrolment
Columns
Indexes
Foreign Keys
Triggers
students
Columns
Indexes
Foreign Keys
Triggers
Views
Administration Schemas
Information
Schema: university
view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql*
Limit to 1000 rows
4 create database University;
5 use University;
6 create table Students
7 (StudentID int primary key,
8 StudentName varchar(25),
9 DOB DATE,
10 Email varchar(25));
11 INSERT INTO Students (StudentID, StudentName, DOB, Email)
12 VALUES (1, "Bidhan Ghosh", '2000-01-01', 'bidhan@gmail.com'),
13 (2, "Zack Piterson", '1998-09-09', 'zack@gmail.com'),
14 (3, "John Bell", '1999-07-25', 'john@gmail.com'),
15 (4, "Steve johnson", '1997-05-05', 'steve@gmail.com'),
16 (5, "Jackson Ganzales", '1996-06-19', 'jackson@gmail.com');
17 # update students table
18 update Students set StudentName= "Jr. Zack Piterson" where StudentID = 2;
19 select * from Students;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
StudentID StudentName DOB Email
1 Bidhan Ghosh 2000-01-01 bidhan@gmail.com
2 Jr. Zack Piterson 1998-09-09 zack@gmail.com
3 John Bell 1999-07-25 john@gmail.com
4 Steve johnson 1997-05-05 steve@gmail.com
5 Jackson Ganzales 1996-06-19 jackson@gmail.com
NULL NULL NULL NULL
Result Grid
Form Editor

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- university
  - Tables
    - courses
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - enrolment
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - students
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
  - Views

Administration Schemas

Information

Schema: university

view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql*

Limit to 1000 rows

```
21  create table Courses
22  (CourseID int primary key,
23  CourseName varchar(25),
24  Credits varchar(25) );
25
26  INSERT INTO Courses (CourseID, CourseName, Credits)
27  VALUES (501, "Python", 4 ),
28         (502, "SQL", 4 ),
29         (503, "ML", 4 ),
30         (504, "AI", 4 ),
31         (505, "R", 4 );
32  # update course name
33  update Courses set CourseName= "Data Visulization" where CourseID = 505;
34  update Courses set CourseName= "Programming Python" where CourseID = 501;
35  select * from Courses;
36
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

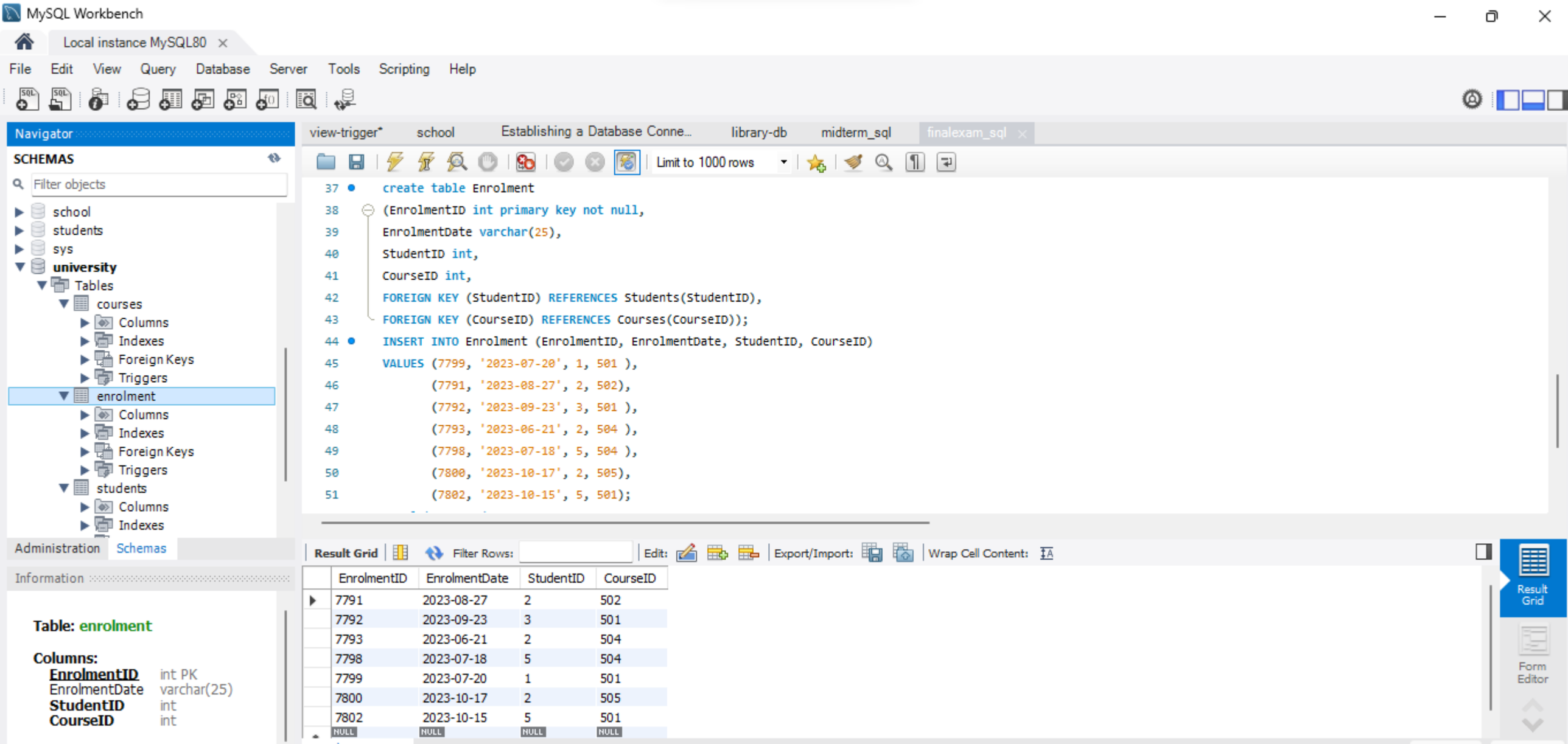
CourseID	CourseName	Credits
501	Programming Python	4
502	SQL	4
503	ML	4
504	AI	4
505	Data Visulization	4
NULL	NULL	NULL

Result Grid

Form Editor

Courses 6

Apply Revert


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
school
students
sys
university
Tables
courses
Columns
Indexes
Foreign Keys
Triggers
enrolment
Columns
Indexes
Foreign Keys
Triggers
students
Columns
Indexes
Administration Schemas
Information
Table: enrolment
Columns:
EnrolmentID int PK
EnrolmentDate varchar(25)
StudentID int
CourseID int
view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql
Limit to 1000 rows
37 create table Enrolment
38 (EnrolmentID int primary key not null,
39 EnrolmentDate varchar(25),
40 StudentID int,
41 CourseID int,
42 FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
43 FOREIGN KEY (CourseID) REFERENCES Courses(CourseID));
44 INSERT INTO Enrolment (EnrolmentID, EnrolmentDate, StudentID, CourseID)
45 VALUES (7799, '2023-07-20', 1, 501),
46 (7791, '2023-08-27', 2, 502),
47 (7792, '2023-09-23', 3, 501),
48 (7793, '2023-06-21', 2, 504),
49 (7798, '2023-07-18', 5, 504),
50 (7800, '2023-10-17', 2, 505),
51 (7802, '2023-10-15', 5, 501);
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
EnrolmentID EnrolmentDate StudentID CourseID
7791 2023-08-27 2 502
7792 2023-09-23 3 501
7793 2023-06-21 2 504
7798 2023-07-18 5 504
7799 2023-07-20 1 501
7800 2023-10-17 2 505
7802 2023-10-15 5 501
NULL NULL NULL NULL
Result Grid
Form Editor

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- school
- students
- sys
- **university**
  - Tables
    - courses
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - enrolment
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - students
      - Columns
      - Indexes

Administration Schemas

Information

Table: enrolment

Columns:

EnrolmentID	int PK
EnrolmentDate	varchar(25)
StudentID	int
CourseID	int

view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql

Limit to 1000 rows

```
41    CourseID int,
42    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
43    FOREIGN KEY (CourseID) REFERENCES Courses(CourseID));
44    INSERT INTO Enrolment (EnrolmentID, EnrolmentDate, StudentID, CourseID)
45    VALUES (7799, '2023-07-20', 1, 501 ),
46           (7791, '2023-08-27', 2, 502),
47           (7792, '2023-09-23', 3, 501 ),
48           (7793, '2023-06-21', 2, 504 ),
49           (7798, '2023-07-18', 5, 504 ),
50           (7800, '2023-10-17', 2, 505),
51           (7802, '2023-10-15', 5, 501);
52    # Delete record
53
54    DELETE FROM Enrolment WHERE EnrolmentID = 7791;
55    select * from Enrolment;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EnrolmentID	EnrolmentDate	StudentID	CourseID
7792	2023-09-23	3	501
7793	2023-06-21	2	504
7798	2023-07-18	5	504
7799	2023-07-20	1	501
7800	2023-10-17	2	505
7802	2023-10-15	5	501
NULL	NULL	NULL	NULL

Result Grid

Form Editor

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- school
- students
- sys
- **university**
  - Tables
    - courses
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - enrolment
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - students
      - Columns
      - Indexes

Administration Schemas

Information

Table: enrolment

Columns:

EnrolmentID	int PK
EnrolmentDate	varchar(25)
StudentID	int
CourseID	int

Object Info Session

view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql*

Limit to 1000 rows

```
52  # Delete record
53
54  DELETE FROM Enrolment WHERE EnrolmentID = 7791;
55
56  select * from Enrolment;
57  #  Write query to retrieve the names of students who are enrolled in a specific course
58
59  SELECT Students.StudentName
60  FROM Students
61  JOIN Enrolment ON Students.StudentID = Enrolment.StudentID
62  WHERE Enrolment.CourseID = 501;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

StudentName
John Bell
Bidhan Ghosh
Jackson Ganzales

Result Grid

Form Editor

Result 20

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
40	12:14:38	select * from Enrolment LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
41	12:15:40	SELECT Students.StudentName FROM Students JOIN Enrolment ON Students.Studen...	3 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- school
- students
- sys
- **university**
  - Tables
    - courses
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - enrolment
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - students
      - Columns
      - Indexes

Administration Schemas

Information

Table: enrolment

Columns:

EnrolmentID	int PK
EnrolmentDate	varchar(25)
StudentID	int
CourseID	int

Object Info Session

view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql*

Limit to 1000 rows

```
65
66      # Write a SQL query to calculate the total number of credits taken by a specific student.
67
68      SELECT Students.StudentName, SUM(Courses.Credits) AS TotalCredits
69      FROM Students
70      JOIN Enrolment ON Students.StudentID = Enrolment.StudentID
71      JOIN Courses ON Enrolment.CourseID = Courses.CourseID
72      WHERE Students.StudentID = 1;
73
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

StudentName	TotalCredits
Bidhan Ghosh	4

Result Grid

Result 21 Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
39	12:14:35	DELETE FROM Enrolment WHERE EnrolmentID = 7791	1 row(s) affected	0.062 sec
40	12:14:38	select * from Enrolment LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
41	12:15:40	SELECT Students.StudentName FROM Students JOIN Enrolment ON Students.Studen...	3 row(s) returned	0.000 sec / 0.000 sec
42	12:27:29	SELECT Students.StudentName, SUM(Courses.Credits) AS TotalCredits FROM Studen...	1 row(s) returned	0.109 sec / 0.000 sec

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- school
- students
- sys
- **university**
  - Tables
    - courses
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - enrolment
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - students
      - Columns
      - Indexes

Administration Schemas

Information

Table: enrolment

Columns:

EnrolmentID	int PK
EnrolmentDate	varchar(25)
StudentID	int
CourseID	int

Object Info Session

view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql

Limit to 1000 rows

```
66  # Write a SQL query to calculate the total number of credits taken by a specific student.
67
68  SELECT Students.StudentName, SUM(Courses.Credits) AS TotalCredits
69  FROM Students
70  JOIN Enrolment ON Students.StudentID = Enrolment.StudentID
71  JOIN Courses ON Enrolment.CourseID = Courses.CourseID
72  WHERE Students.StudentID = 1;
73
74  # Write a SQL query to retrieve the courses with more than 3 credits.
75
76  SELECT *
77  FROM Courses
78  WHERE Credits > 3;
79
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CourseID	CourseName	Credits
501	Programming Python	4
502	SQL	4
503	ML	4
504	AI	4
505	Data Visulization	4
NULL	NULL	NULL

Result Grid | Form Editor

Courses 22 | Apply | Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
40	12:14:38	select * from Enrolment LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

SQL script saved to 'D:\DataScience\DataElement\finalexam_sql.sql'

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- school
- students
- sys
- **university**
  - Tables
    - courses
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - enrolment
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - students
      - Columns
      - Indexes

Administration Schemas

Information

Table: enrolment

Columns:

EnrolmentID	int PK
EnrolmentDate	varchar(25)
StudentID	int
CourseID	int

Object Info Session

view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql*

Limit to 1000 rows

```
74  # Write a SQL query to retrieve the courses with more than 3 credits.
75
76  SELECT *
77  FROM Courses
78  WHERE Credits > 3;
79
80  # Write a SQL query that joins the Students, Enrolments, and Courses tables to retrieve the names of students and the courses they are enrolled in.
81
82  SELECT Students.StudentName, Courses.CourseName
83  FROM Students
84  JOIN Enrolment ON Students.StudentID = Enrolment.StudentID
85  JOIN Courses ON Enrolment.CourseID = Courses.CourseID;
86
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

StudentName	CourseName
Bidhan Ghosh	Programming Python
Jr. Zack Piterson	AI
Jr. Zack Piterson	Data Visulization
John Bell	Programming Python
Jackson Ganzales	AI
Jackson Ganzales	Programming Python

Result Grid | Form Editor | Read Only

Result 23

Output

Action Output

#	Time	Action	Message	Duration / Fetch
43	12:31:22	SELECT * FROM Courses WHERE Credits > 3 LIMIT 0, 1000	5 row(s) returned	0.047 sec / 0.000 sec

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- school
- students
- sys
- **university**
  - Tables
    - courses
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - enrolment
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - students
      - Columns
      - Indexes

Administration Schemas

Information

Table: enrolment

Columns:

EnrolmentID	int PK
EnrolmentDate	varchar(25)
StudentID	int
CourseID	int

Object Info Session

view-trigger* school Establishing a Database Conne... library-db midterm_sql finalexam_sql*

Limit to 1000 rows

```
87  # Write a SQL query to create a report that displays the student names and a custom column "Status" indicating "Full-time" for students enrolled in more than 7 credits
88  # and "Part time" for others.
89
90  SELECT StudentName,
91         CASE
92             WHEN (SELECT SUM(Credits) FROM Courses WHERE Courses.CourseID = Enrolment.CourseID) > 7 THEN 'Full-time'
93             ELSE 'Part-time'
94         END AS Status
95  FROM Students
96  JOIN Enrolment ON Students.StudentID = Enrolment.StudentID;
97
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

StudentName	Status
Bidhan Ghosh	Part-time
Jr. Zack Piterson	Part-time
Jr. Zack Piterson	Part-time
John Bell	Part-time
Jackson Ganzales	Part-time
Jackson Ganzales	Part-time

Result Grid | Form Editor

Result 24 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
44	12:34:11	SELECT Students.StudentName, Courses.CourseName FROM Students JOIN Enrolme...	6 row(s) returned	0.000 sec / 0.000 sec
45	12:39:24	SELECT StudentName, CASE WHEN (SELECT SUM(Credits) FROM Cours...	6 row(s) returned	0.015 sec / 0.000 sec