MY ASSIGNMENT : JAVA Foundation Hexaware

TASK - 1

Task 1.
Database Design:

1. **Create the database named "SISDB"**

CODE :

```
create database SISDB;

use SISDB;
```

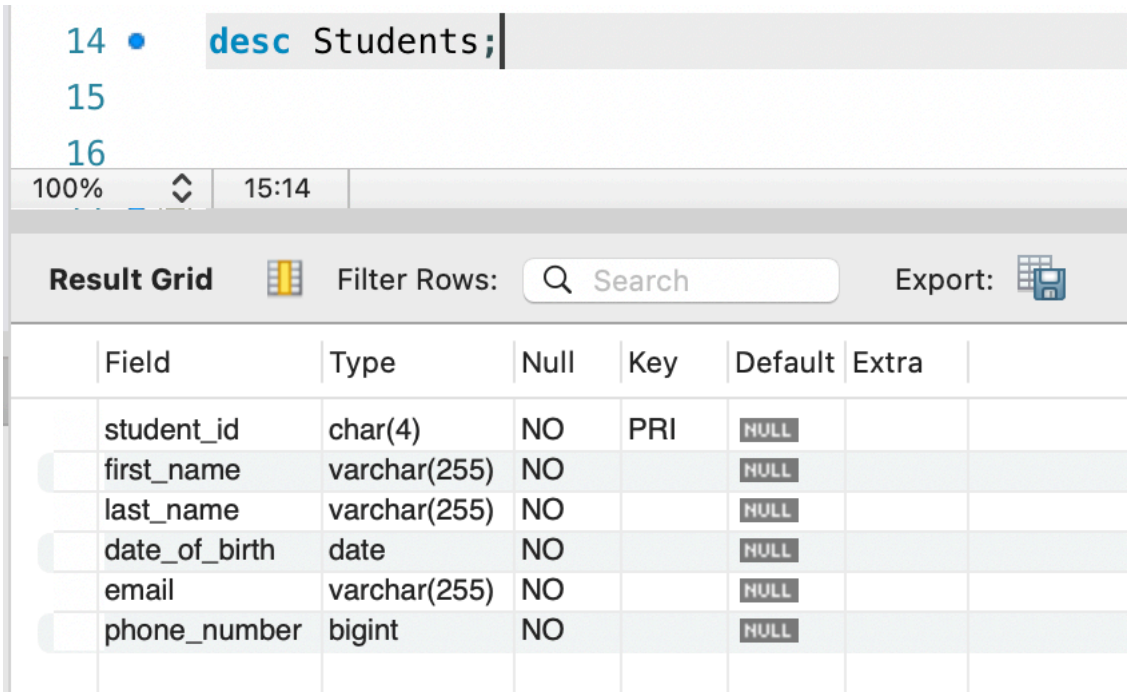
1. **Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.**

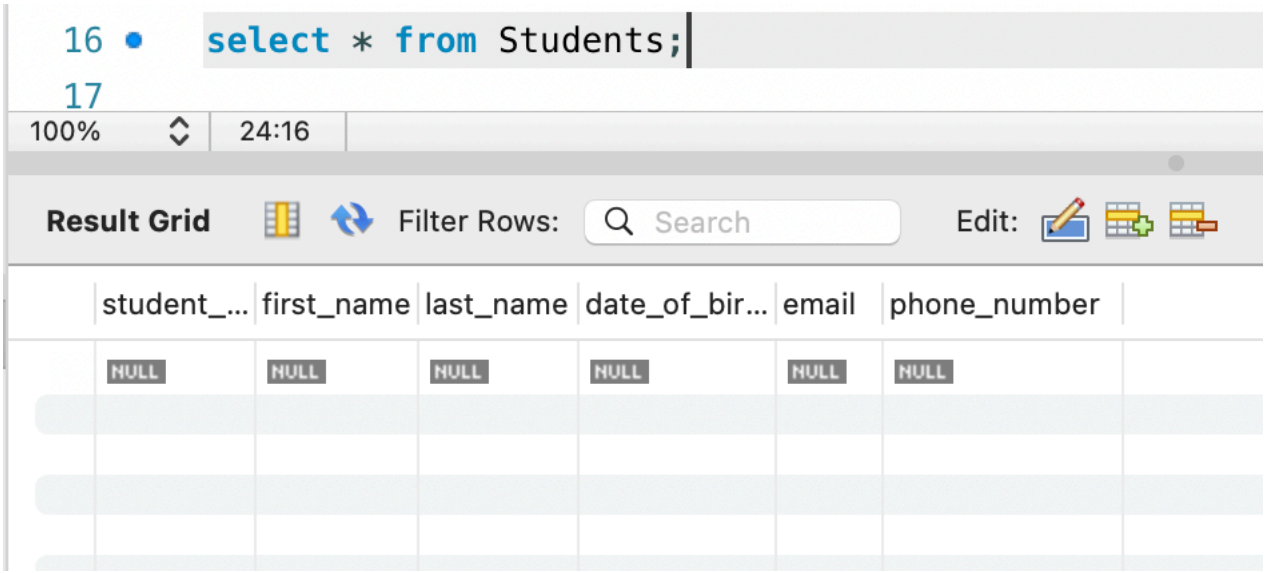
a. Students
b. Courses
c. Enrollments
d. Teacher
e. Payments

CODE :

Student

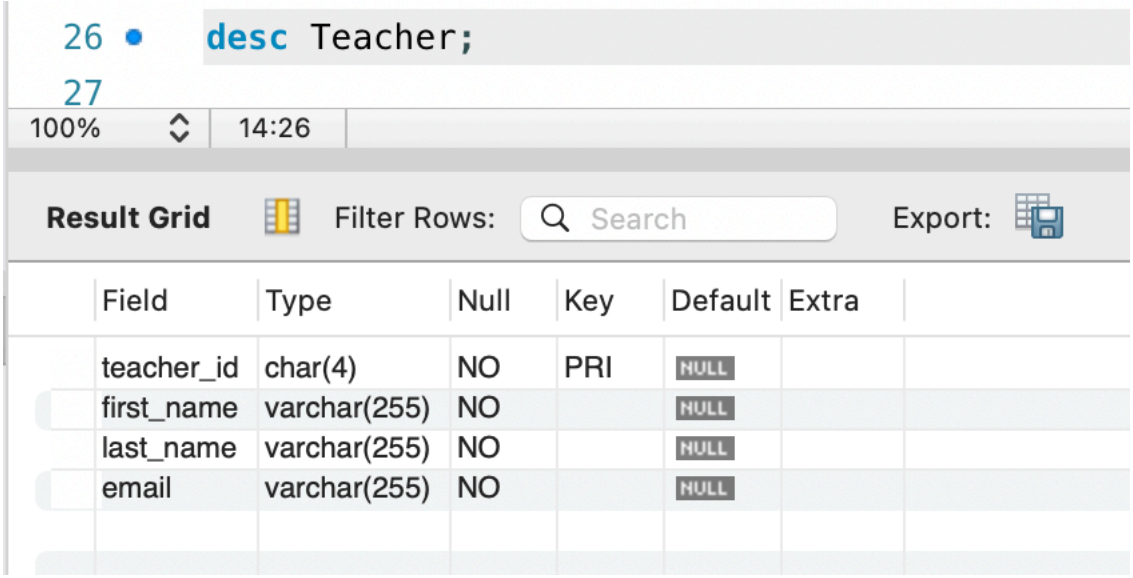
```
create table Students(
        student_id char(4) primary key,
    first_name varchar(255) not null,
    last_name varchar(255) not null,
    date_of_birth date not null,
    email varchar(255) not null,
    phone_number bigint not null
);
```

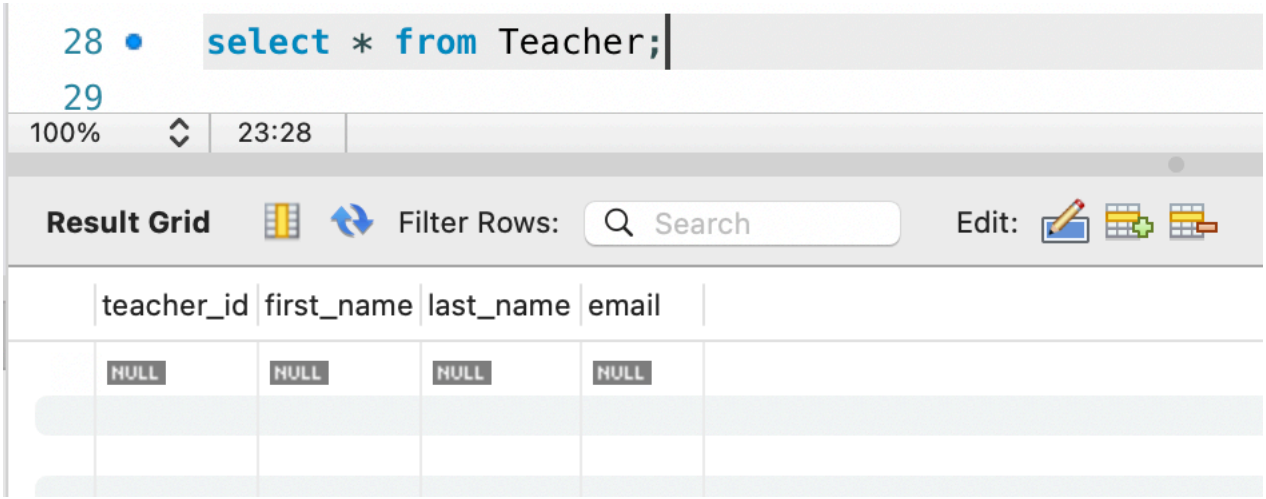




Teacher

```
create table Teacher(
    teacher_id char(4) primary key,
  first_name varchar(255) not null,
  last_name varchar(255) not null,
    email varchar(255) not null
);
```



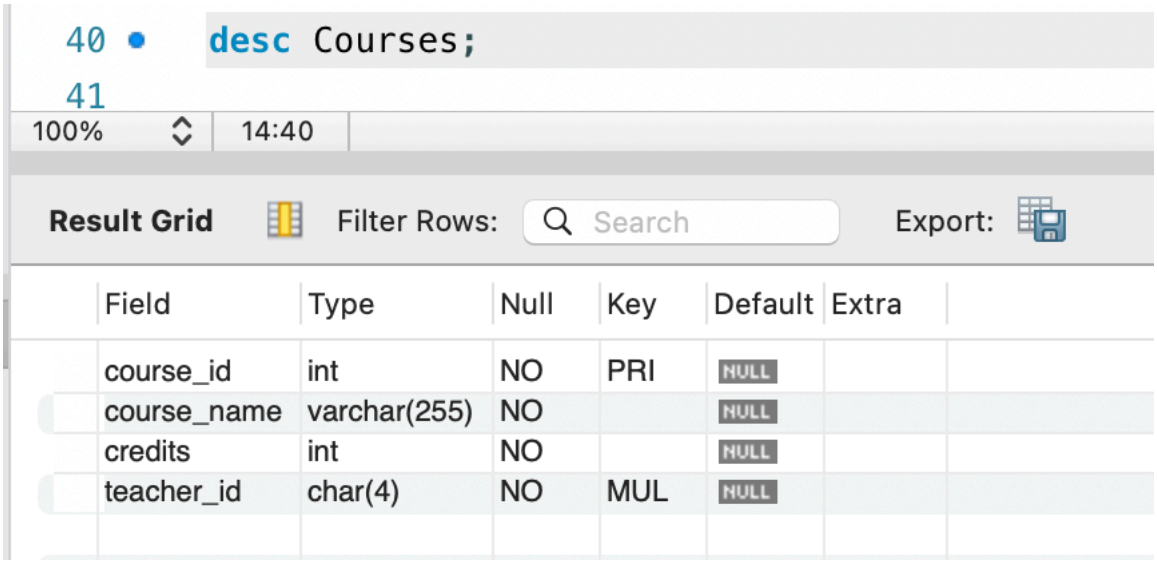


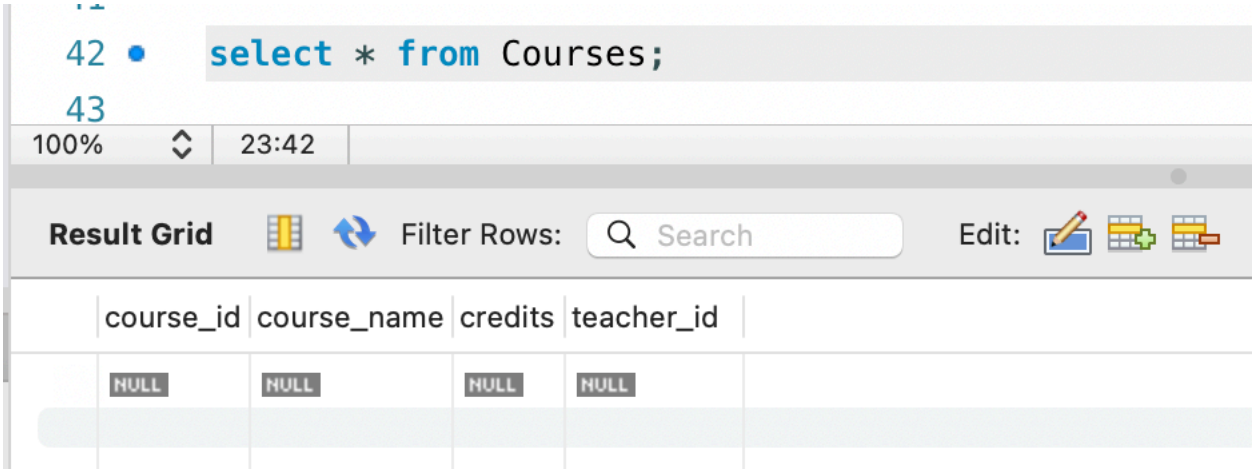
Courses

```
create table Courses(
        course_id  int not null primary key auto_increment,
    course_name varchar(255) not null,
    credits int not null,
    teacher_id char(4),
    constraint tid_fk foreign key(teacher_id)
    references Teacher(teacher_id)
    on delete set null on update cascade
) auto_increment = 101;

-- altering table to set auto increment for course from 101
```

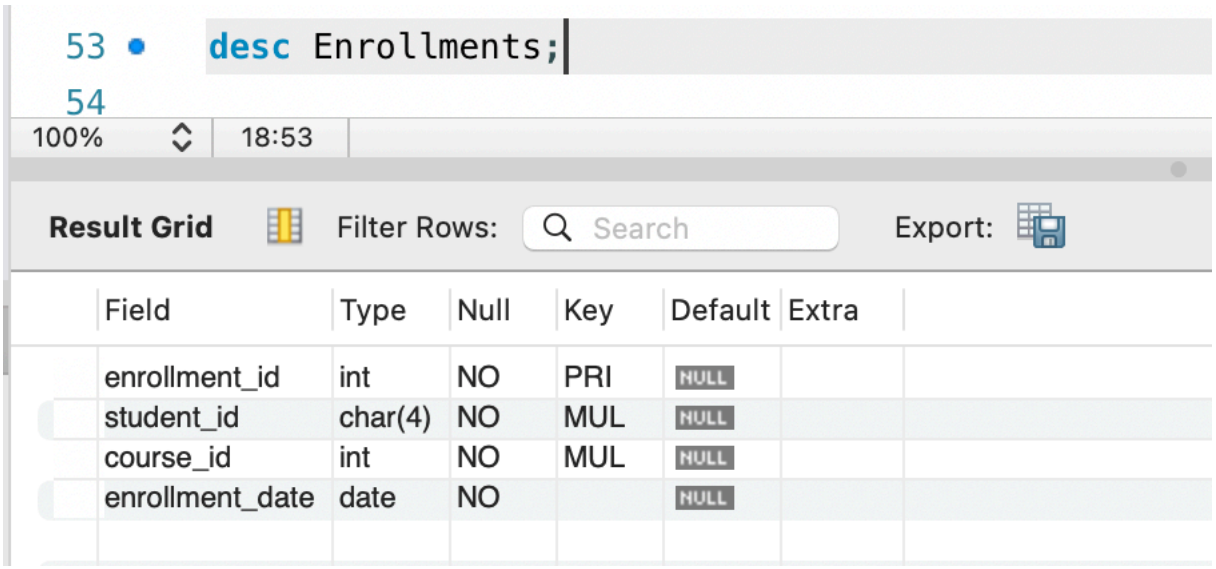
alter table Courses auto_increment = 101;

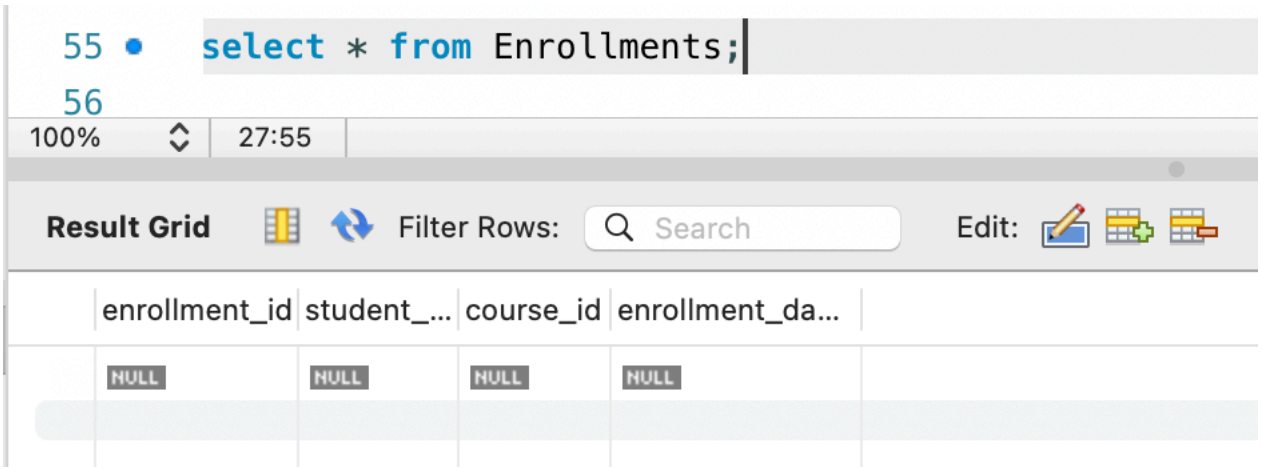




Enrollments

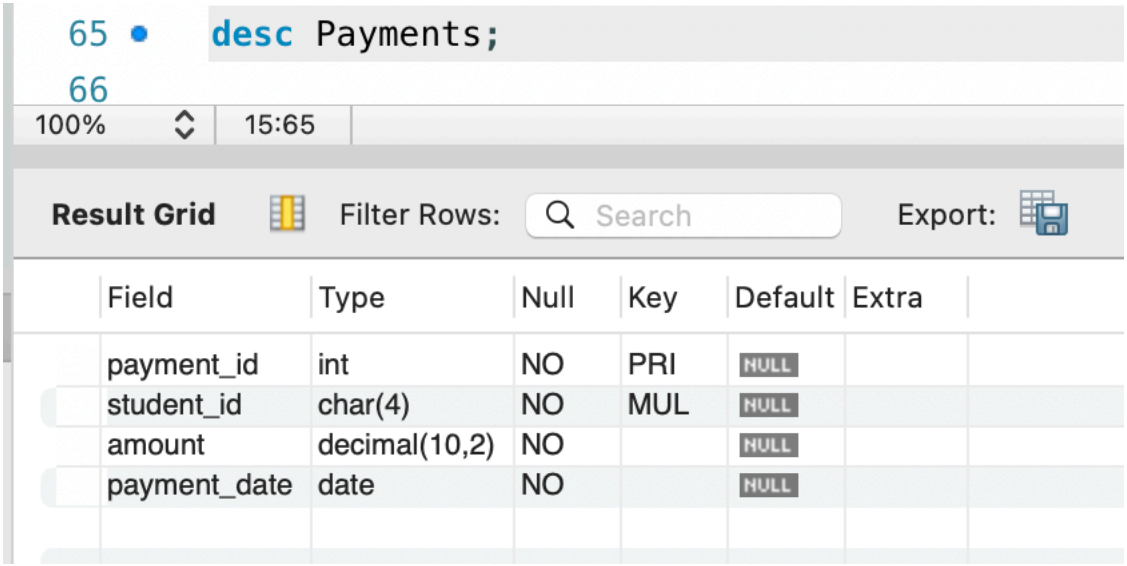
```
create table Enrollments(
    enrollment_id  int not null primary key auto_increment,
  student_id char(4) not null,
  course_id int not null,
  enrollment_date date not null,
    constraint sid_fk foreign key(student_id)
  references  Students(student_id) on delete cascade on update cascade,
  constraint cid_fk foreign key(course_id)
  references Courses(course_id) on delete cascade on update cascade
);
```

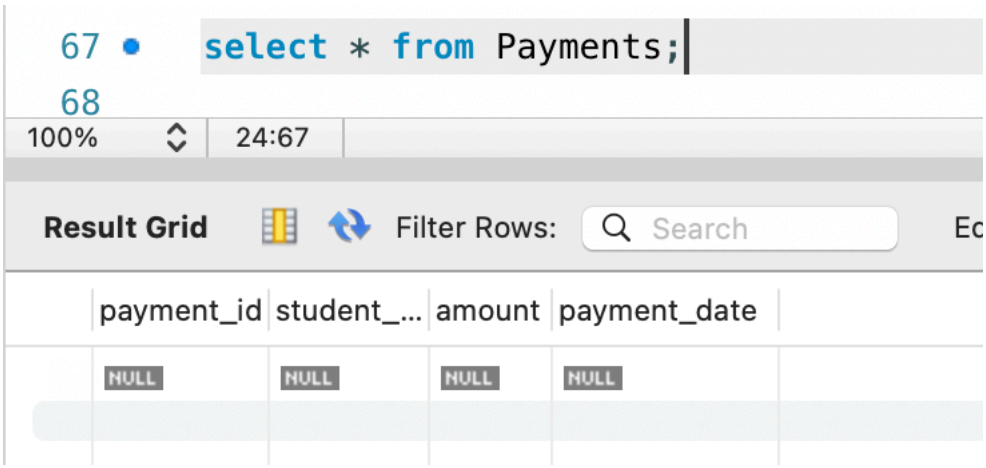




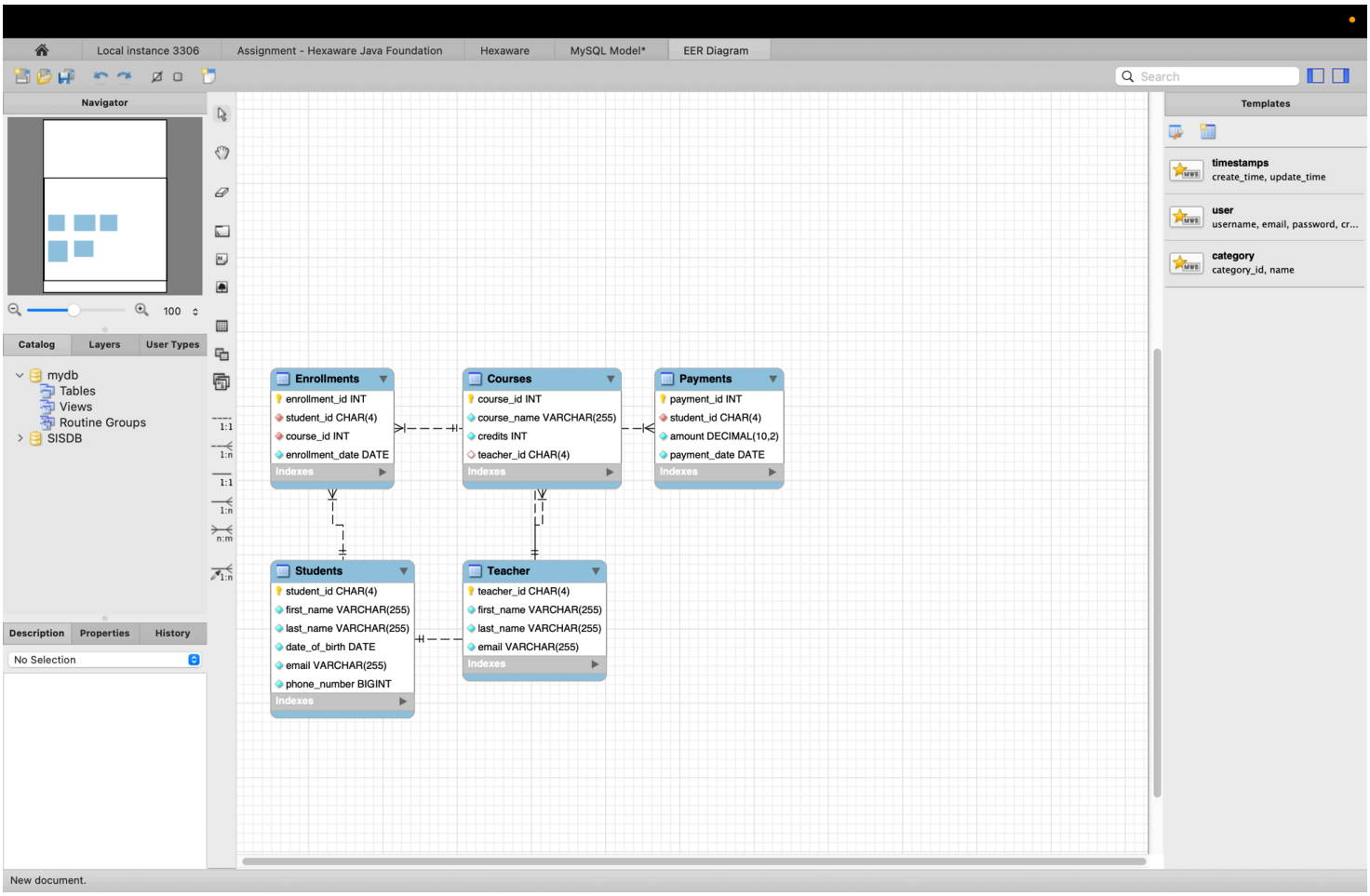
Payments

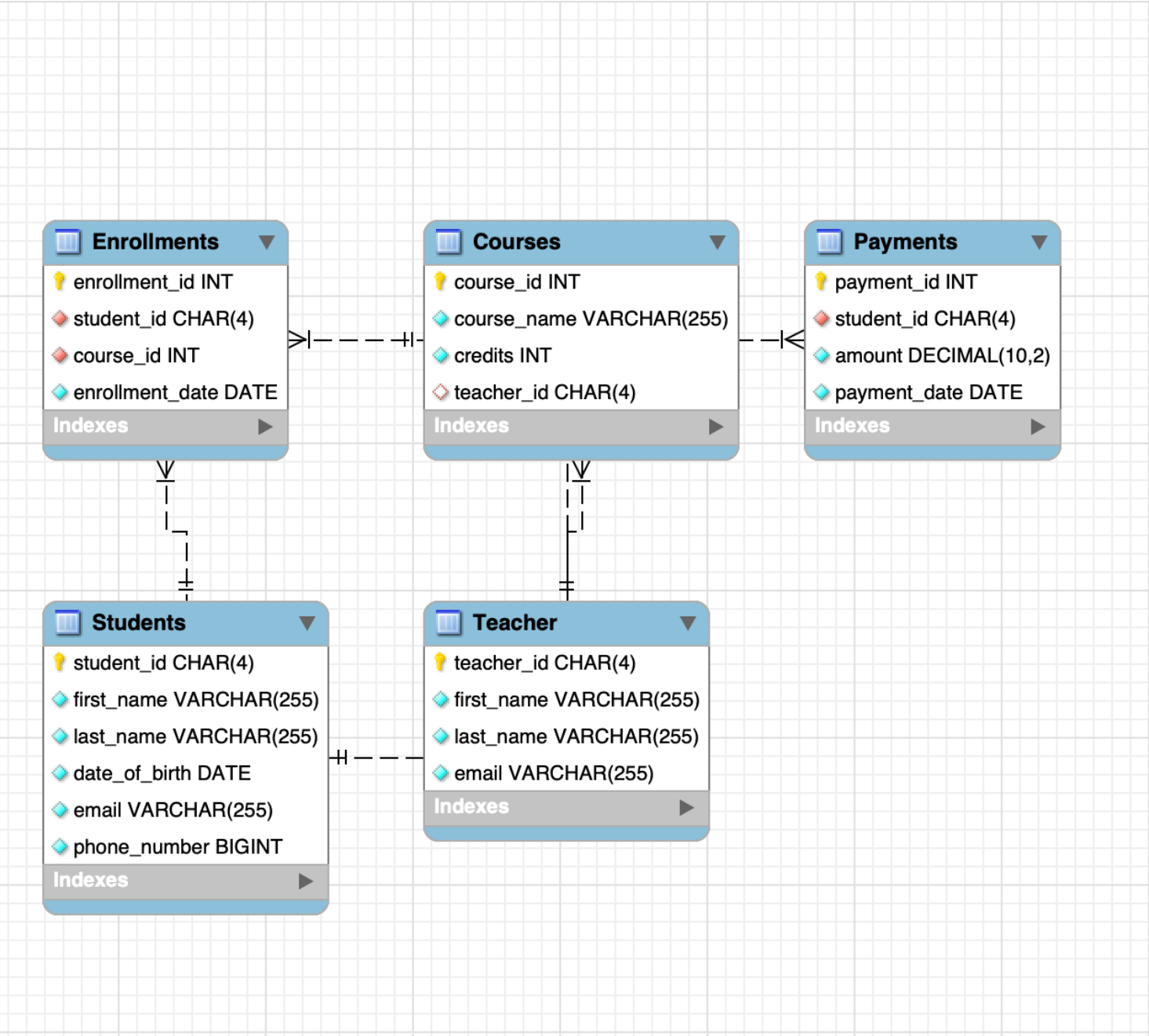
```
create table Payments(
        payment_id  int not null primary key auto_increment,
    student_id char(4) not null,
    amount decimal(10, 2) not null,
    payment_date date not null,
        constraint studentid_fk foreign key(student_id)
    references  Students(student_id) on delete cascade on update cascade
) auto_increment = 1001;
```





3. **Create an ERD (Entity Relationship Diagram) for the database.**


Local instance 3306
Assignment - Hexaware Java Foundation
Hexaware
MySQL Model*
EER Diagram
Search
Navigator
Templates
timestamps
create_time, update_time
user
username, email, password, cr...
category
category_id, name
Catalog
Layers
User Types
mydb
Tables
Views
Routine Groups
SISDB
Description
Properties
History
No Selection
Enrollments
enrollment_id INT
student_id CHAR(4)
course_id INT
enrollment_date DATE
Indexes
Courses
course_id INT
course_name VARCHAR(255)
credits INT
teacher_id CHAR(4)
Indexes
Payments
payment_id INT
student_id CHAR(4)
amount DECIMAL(10,2)
payment_date DATE
Indexes
Students
student_id CHAR(4)
first_name VARCHAR(255)
last_name VARCHAR(255)
date_of_birth DATE
email VARCHAR(255)
phone_number BIGINT
Indexes
Teacher
teacher_id CHAR(4)
first_name VARCHAR(255)
last_name VARCHAR(255)
email VARCHAR(255)
Indexes
New document.



4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

Primary Key Constraints

Table Name	Primary Key Column	Data Type
Students	student_id	CHAR(4)
Teacher	teacher_id	CHAR(4)
Courses	course_id	INT
Enrollments	enrollment_id	INT
Payments	payment_id	INT

Foreign Key Constraints

Table Name	Foreign Key Column	References	On Delete	On Update
Courses	teacher_id	Teacher(teacher_id)	SET NULL	CASCADE
Enrollments	student_id	Students(student_id)	CASCADE	CASCADE
Enrollments	course_id	Courses(course_id)	CASCADE	CASCADE
Payments	student_id	Students(student_id)	CASCADE	CASCADE

5. **Insert at least 10 sample records into each of the following tables.**

i. Students

ii. Courses

iii. Enrollments

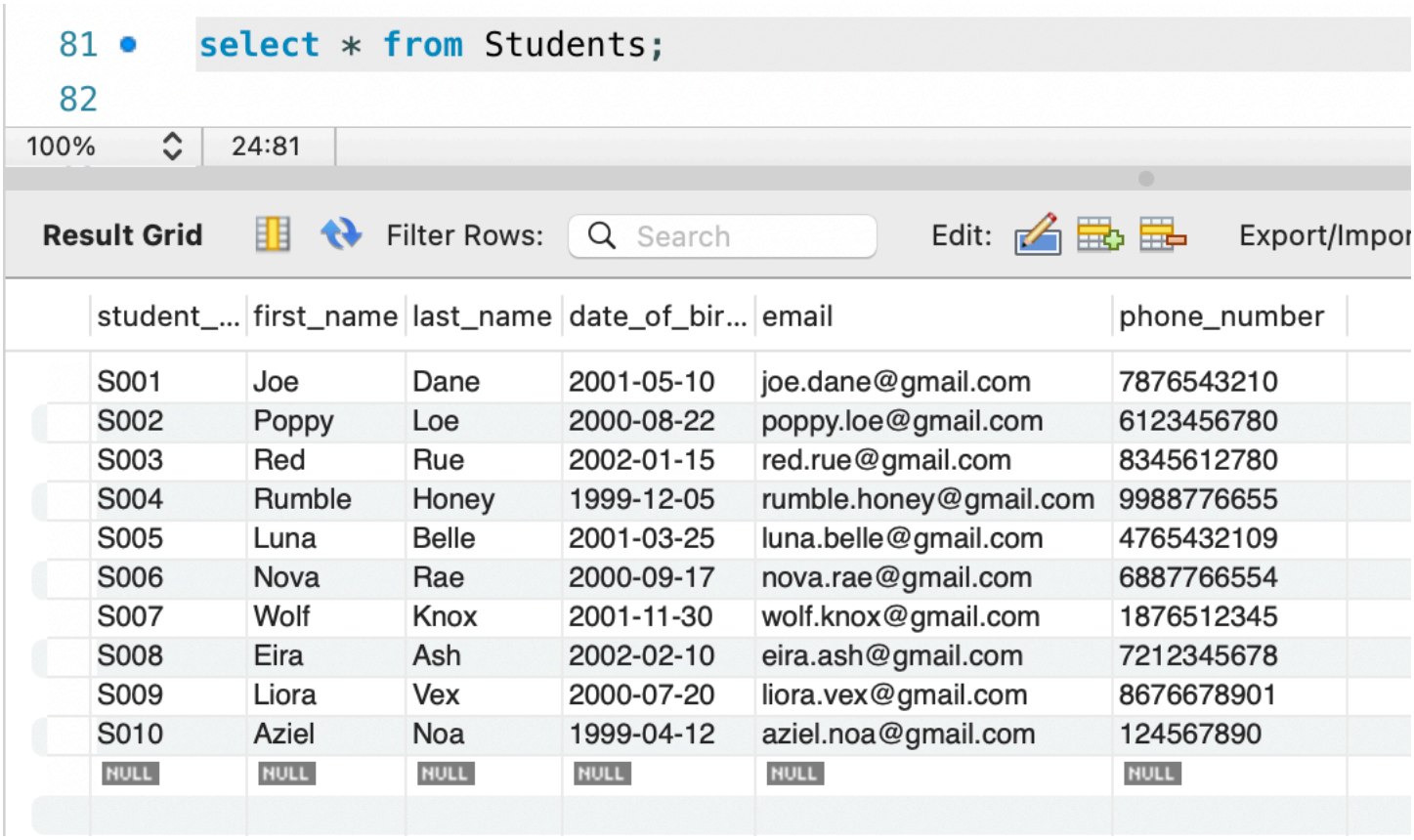
iv. Teacher

v. Payments

CODE :

Students

```
insert into Students
values('S001', 'Joe', 'Dane', '2001-05-10', 'joe.dane@gmail.com', 7876543210),
('S002', 'Poppy', 'Loe', '2000-08-22', 'poppy.loe@gmail.com', 6123456780),
('S003', 'Red', 'Rue', '2002-01-15', 'red.rue@gmail.com', 8345612780),
('S004', 'Rumble', 'Honey', '1999-12-05', 'rumble.honey@gmail.com', 9988776655),
('S005', 'Luna', 'Belle', '2001-03-25', 'luna.belle@gmail.com', 4765432109),
('S006', 'Nova', 'Rae', '2000-09-17', 'nova.rae@gmail.com', 6887766554),
('S007', 'Wolf', 'Knox', '2001-11-30', 'wolf.knox@gmail.com', 1876512345),
('S008', 'Eira', 'Ash', '2002-02-10', 'eira.ash@gmail.com', 7212345678),
('S009', 'Liora', 'Vex', '2000-07-20', 'liora.vex@gmail.com', 8676678901),
('S010', 'Aziel', 'Noa', '1999-04-12', 'aziel.noa@gmail.com', 0124567890);
```



```
81 • select * from Students;
82
```

100% 24:81

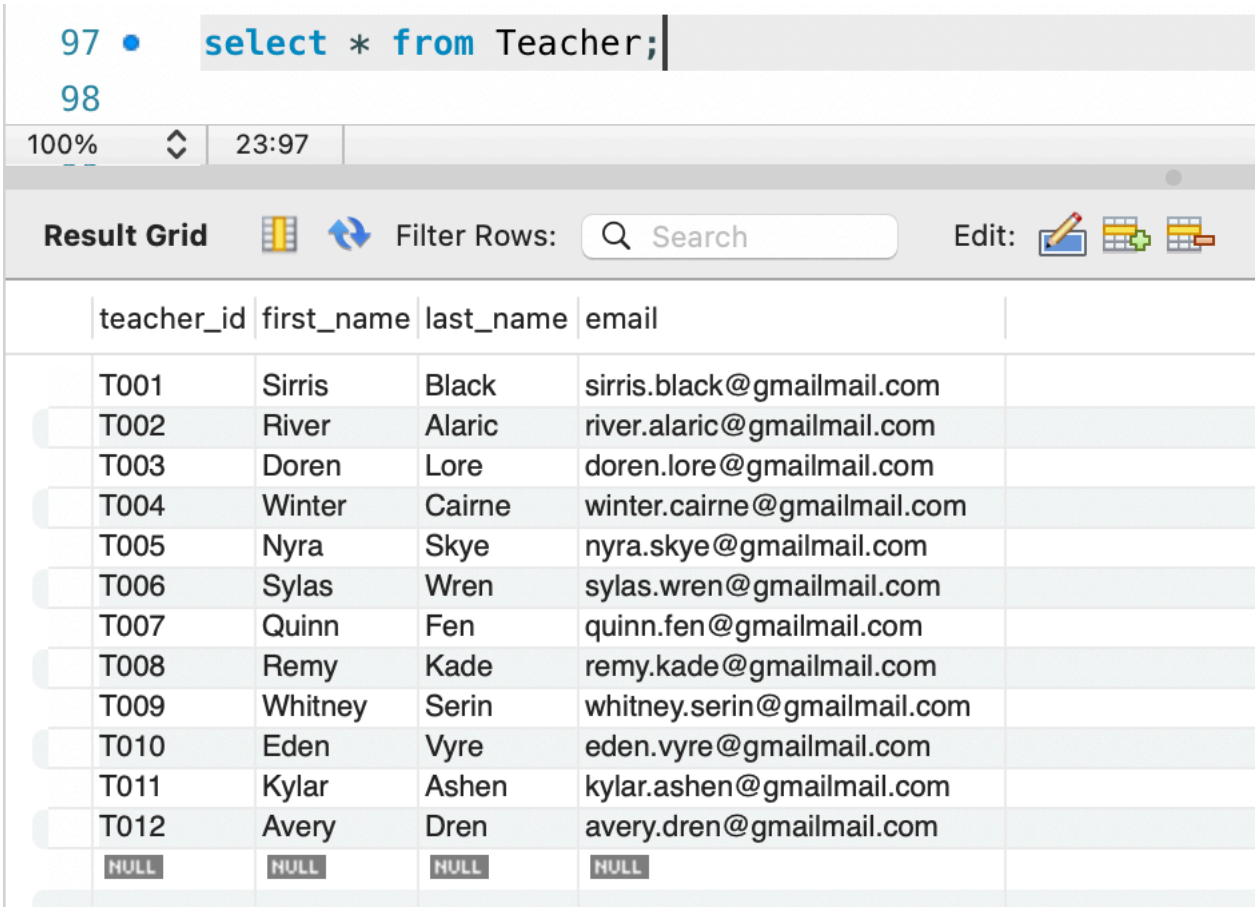
Result Grid Filter Rows: Search Edit: Export/Impor

student_...	first_name	last_name	date_of_bir...	email	phone_number
S001	Joe	Dane	2001-05-10	joe.dane@gmail.com	7876543210
S002	Poppy	Loe	2000-08-22	poppy.loe@gmail.com	6123456780
S003	Red	Rue	2002-01-15	red.rue@gmail.com	8345612780
S004	Rumble	Honey	1999-12-05	rumble.honey@gmail.com	9988776655
S005	Luna	Belle	2001-03-25	luna.belle@gmail.com	4765432109
S006	Nova	Rae	2000-09-17	nova.rae@gmail.com	6887766554
S007	Wolf	Knox	2001-11-30	wolf.knox@gmail.com	1876512345
S008	Eira	Ash	2002-02-10	eira.ash@gmail.com	7212345678
S009	Liora	Vex	2000-07-20	liora.vex@gmail.com	8676678901
S010	Aziel	Noa	1999-04-12	aziel.noa@gmail.com	124567890
NULL	NULL	NULL	NULL	NULL	NULL

Teacher

```
insert into Teacher values
('T001', 'Sirris', 'Black', 'sirris.black@gmailmail.com'),
('T002', 'River', 'Alaric', 'river.alaric@gmailmail.com'),
('T003', 'Doren', 'Lore', 'doren.lore@gmailmail.com'),
('T004', 'Winter', 'Cairne', 'winter.cairne@gmailmail.com'),
('T005', 'Nyra', 'Skye', 'nyra.skye@gmailmail.com'),
```

('T006', 'Sylas', 'Wren', 'sylas.wren@gmailmail.com'),
('T007', 'Quinn', 'Fen', 'quinn.fen@gmailmail.com'),
('T008', 'Remy', 'Kade', 'remy.kade@gmailmail.com'),
('T009', 'Whitney', 'Serin', 'whitney.serin@gmailmail.com'),
('T010', 'Eden', 'Vyre', 'eden.vyre@gmailmail.com'),
('T011', 'Kylar', 'Ashen', 'kylar.ashen@gmailmail.com'),
('T012', 'Avery', 'Dren', 'avery.dren@gmailmail.com');



teacher_id	first_name	last_name	email
T001	Sirris	Black	sirris.black@gmailmail.com
T002	River	Alaric	river.alaric@gmailmail.com
T003	Doren	Lore	doren.lore@gmailmail.com
T004	Winter	Cairne	winter.cairne@gmailmail.com
T005	Nyra	Skye	nyra.skye@gmailmail.com
T006	Sylas	Wren	sylas.wren@gmailmail.com
T007	Quinn	Fen	quinn.fen@gmailmail.com
T008	Remy	Kade	remy.kade@gmailmail.com
T009	Whitney	Serin	whitney.serin@gmailmail.com
T010	Eden	Vyre	eden.vyre@gmailmail.com
T011	Kylar	Ashen	kylar.ashen@gmailmail.com
T012	Avery	Dren	avery.dren@gmailmail.com
NULL	NULL	NULL	NULL

Courses

insert into Courses(course_name, credits, teacher_id)
values ('Introduction to Computer Science', 4, 'T001'),
('Modern English Literature', 3, 'T002'),
('Data Structures and Algorithms', 5, 'T001'),
('World History', 3, 'T003'),
('Fundamentals of Physics', 5, 'T004'),
('Creative Writing', 2, 'T002'),
('Calculus', 4, 'T005'),
('Principles of Economics', 3, 'T006'),

('Introduction to Psychology', 3, 'T007'),
('Environmental Science', 4, 'T008'),
('Art and Design Basics', 2, 'T009'),
('Sociology and Culture', 3, 'T010'),
('Linear Algebra', 5, 'T005'),
('Philosophy of Mind', 2, 'T011'),
('Web Development', 4, 'T012'),
('Film Studies and Analysis', 3, 'T003'),
('Music Theory', 2, 'T004'),
('Theatre and Performance', 3, 'T006'),
('Modern Political Thought', 3, 'T010'),
('Cultural Anthropology', 4, 'T011'),
('Ethics and Society', 1, 'T012');

```
122 •  select * from Courses;
123
```

100% 23:122

Result Grid Filter Rows: Search Edit:

course_id	course_name	credits	teacher_id
102	Modern English Literature	3	T002
103	Data Structures and Algorithms	5	T001
104	World History	3	T003
105	Fundamentals of Physics	5	T004
106	Creative Writing	2	T002
107	Calculus	4	T005
108	Principles of Economics	3	T006
109	Introduction to Psychology	3	T007
110	Environmental Science	4	T008
111	Art and Design Basics	2	T009
112	Sociology and Culture	3	T010
113	Linear Algebra	5	T005
114	Philosophy of Mind	2	T011
115	Web Development	4	T012
116	Film Studies and Analysis	3	T003
117	Music Theory	2	T004
118	Theatre and Performance	3	T006
119	Modern Political Thought	3	T010
120	Cultural Anthropology	4	T011
121	Ethics and Society	1	T012
NULL	NULL	NULL	NULL

Enrollments

```
insert into Enrollments(student_id, course_id, enrollment_date)
values('S004', 116, '2022-08-01'),
('S010', 121, '2022-08-01'),
('S001', 101, '2023-01-15'),
('S007', 120, '2023-08-01'),
('S002', 102, '2023-08-02'),
('S009', 118, '2023-08-02'),
('S006', 113, '2024-02-03'),
('S003', 105, '2024-08-01'),
('S008', 115, '2024-08-01'),
('S005', 112, '2024-08-03'),
('S004', 104, '2025-01-12'),
('S010', 114, '2025-01-12'),
('S001', 103, '2025-08-01'),
('S007', 110, '2025-08-02'),
('S002', 106, '2025-08-02'),
('S009', 102, '2025-08-04'),
('S006', 107, '2025-08-05'),
('S003', 113, '2025-08-05'),
('S008', 108, '2025-08-05'),
('S005', 109, '2025-08-06');
```

```
146 •  select * from Enrollments;
147
```

100% 27:146

Result Grid Filter Rows: Search Edit:

enrollment_id	student_...	course_id	enrollment_da...
1	S004	116	2022-08-01
2	S010	121	2022-08-01
3	S001	101	2023-01-15
4	S007	120	2023-08-01
5	S002	102	2023-08-02
6	S009	118	2023-08-02
7	S006	113	2024-02-03
8	S003	105	2024-08-01
9	S008	115	2024-08-01
10	S005	112	2024-08-03
11	S004	104	2025-01-12
12	S010	114	2025-01-12
13	S001	103	2025-08-01
14	S007	110	2025-08-02
15	S002	106	2025-08-02
16	S009	102	2025-08-04
17	S006	107	2025-08-05
18	S003	113	2025-08-05
19	S008	108	2025-08-05
20	S005	109	2025-08-06
NULL	NULL	NULL	NULL

Payments

insert into Payments (student_id, amount, payment_date)
values('S004', 11000.00, '2022-08-05'),
('S010', 9500.00, '2022-08-10'),
('S001', 12000.00, '2023-01-20'),
('S007', 11500.00, '2023-08-05'),
('S002', 9800.00, '2023-08-08'),
('S009', 10200.00, '2023-08-12'),
('S006', 11800.00, '2024-02-05'),
('S003', 9900.00, '2024-08-10'),
('S008', 10150.00, '2024-08-12'),
('S005', 9700.00, '2024-08-15'),
('S004', 11300.00, '2025-01-20'),

('S010', 9800.00, '2025-01-25'),
('S001', 12200.00, '2025-08-01'),
('S007', 11450.00, '2025-08-02'),
('S002', 9850.00, '2025-08-04'),
('S009', 10000.00, '2025-08-05'),
('S006', 11900.00, '2025-08-07'),
('S003', 9950.00, '2025-08-08'),
('S008', 10300.00, '2025-08-09'),
('S005', 9650.00, '2025-08-10');

```
177 •  select * from Payments;
178
```

100% 24:177

Result Grid Filter Rows: Search

payment_id	student_...	amount	payment_date
1001	S004	11000.00	2022-08-05
1002	S010	9500.00	2022-08-10
1003	S001	12000.00	2023-01-20
1004	S007	11500.00	2023-08-05
1005	S002	9800.00	2023-08-08
1006	S009	10200.00	2023-08-12
1007	S006	11800.00	2024-02-05
1008	S003	9900.00	2024-08-10
1009	S008	10150.00	2024-08-12
1010	S005	9700.00	2024-08-15
1011	S004	11300.00	2025-01-20
1012	S010	9800.00	2025-01-25
1013	S001	12200.00	2025-08-01
1014	S007	11450.00	2025-08-02
1015	S002	9850.00	2025-08-04
1016	S009	10000.00	2025-08-05
1017	S006	11900.00	2025-08-07
1018	S003	9950.00	2025-08-08
1019	S008	10300.00	2025-08-09
1020	S005	9650.00	2025-08-10
NULL	NULL	NULL	NULL

TASK - 2

Tasks 2.

Select, Where, Between, AND, LIKE:

1. Write an SQL query to insert a new student into the "Students" table with the following details:

a. First Name: John

b. Last Name: Doe

c. Date of Birth: 1995-08-15

d. Email: john.doe@example.com

e. Phone Number: 1234567890

CODE :

```
insert into Students
values('S011', 'John', 'Doe', '1995-08-15', 'john.doe@example.com', 1234567890);
```

```
175 •  select * from Students;
176
```

100% 24:175

Result Grid | Filter Rows: Search | Edit: | Export/Import:

student_...	first_name	last_name	date_of_bir...	email	phone_number
S001	Joe	Dane	2001-05-10	joe.dane@gmail.com	7876543210
S002	Poppy	Loe	2000-08-22	poppy.loe@gmail.com	6123456780
S003	Red	Rue	2002-01-15	red.rue@gmail.com	8345612780
S004	Rumble	Honey	1999-12-05	rumble.honey@gmail.com	9988776655
S005	Luna	Belle	2001-03-25	luna.belle@gmail.com	4765432109
S006	Nova	Rae	2000-09-17	nova.rae@gmail.com	6887766554
S007	Wolf	Knox	2001-11-30	wolf.knox@gmail.com	1876512345
S008	Eira	Ash	2002-02-10	eira.ash@gmail.com	7212345678
S009	Liora	Vex	2000-07-20	liora.vex@gmail.com	8676678901
S010	Aziel	Noa	1999-04-12	aziel.noa@gmail.com	124567890
S011	John	Doe	1995-08-15	john.doe@example.com	1234567890
NULL	NULL	NULL	NULL	NULL	NULL

2. Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.

CODE :

insert into Enrollments values(21, 'S011', 112, '2025-08-07');

```
179 •   select * from Enrollments;
180
```

100% 27:179

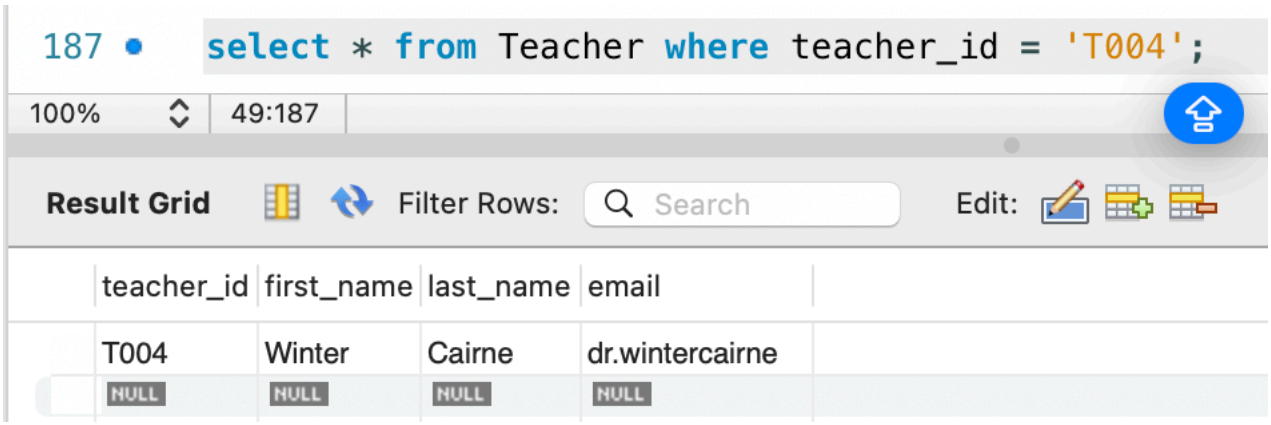
Result Grid Filter Rows: Search

enrollment_id	student_...	course_id	enrollment_da...
2	S010	121	2022-08-01
3	S001	101	2023-01-15
4	S007	120	2023-08-01
5	S002	102	2023-08-02
6	S009	118	2023-08-02
7	S006	113	2024-02-03
8	S003	105	2024-08-01
9	S008	115	2024-08-01
10	S005	112	2024-08-03
11	S004	104	2025-01-12
12	S010	114	2025-01-12
13	S001	103	2025-08-01
14	S007	110	2025-08-02
15	S002	106	2025-08-02
16	S009	102	2025-08-04
17	S006	107	2025-08-05
18	S003	113	2025-08-05
19	S008	108	2025-08-05
20	S005	109	2025-08-06
21	S011	112	2025-08-07
NULL	NULL	NULL	NULL

3. **Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.**

CODE :

update Teacher
set email = 'dr.wintercairne'
where teacher_id = 'T004';



4. **Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.**

CODE :

```
delete from Enrollments where student_id = 'S008' and course_id = 108;
```

Before

```
209 •   select * from Enrollments;
210
```

100% 27:209

Result Grid Filter Rows: Search

enrollment_id	student_...	course_id	enrollment_da...
1	S004	116	2022-08-01
2	S010	121	2022-08-01
3	S001	101	2023-01-15
4	S007	120	2023-08-01
5	S002	102	2023-08-02
6	S009	118	2023-08-02
7	S006	113	2024-02-03
8	S003	105	2024-08-01
9	S008	115	2024-08-01
10	S005	112	2024-08-03
11	S004	104	2025-01-12
12	S010	114	2025-01-12
13	S001	103	2025-08-01
14	S007	110	2025-08-02
15	S002	106	2025-08-02
16	S009	102	2025-08-04
17	S006	107	2025-08-05
18	S003	113	2025-08-05
19	S008	108	2025-08-05
20	S005	109	2025-08-06
21	S011	112	2025-08-07
NULL	NULL	NULL	NULL

After

```
197 •  select * from Enrollments;
198
```

100% 27:197

Result Grid Filter Rows: Search

enrollment_id	student_...	course_id	enrollment_da...
1	S004	116	2022-08-01
2	S010	121	2022-08-01
3	S001	101	2023-01-15
4	S007	120	2023-08-01
5	S002	102	2023-08-02
6	S009	118	2023-08-02
7	S006	113	2024-02-03
8	S003	105	2024-08-01
9	S008	115	2024-08-01
10	S005	112	2024-08-03
11	S004	104	2025-01-12
12	S010	114	2025-01-12
13	S001	103	2025-08-01
14	S007	110	2025-08-02
15	S002	106	2025-08-02
16	S009	102	2025-08-04
17	S006	107	2025-08-05
18	S003	113	2025-08-05
20	S005	109	2025-08-06
21	S011	112	2025-08-07
NULL	NULL	NULL	NULL

5. **Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.**

CODE :

```
update Courses
set teacher_id = 'T001'
where course_id = 115;
```

Before

```
122 •  select * from Courses;
123
```

100% 23:122

Result Grid Filter Rows: Search Edit:

course_id	course_name	credits	teacher_id
102	Modern English Literature	3	T002
103	Data Structures and Algorithms	5	T001
104	World History	3	T003
105	Fundamentals of Physics	5	T004
106	Creative Writing	2	T002
107	Calculus	4	T005
108	Principles of Economics	3	T006
109	Introduction to Psychology	3	T007
110	Environmental Science	4	T008
111	Art and Design Basics	2	T009
112	Sociology and Culture	3	T010
113	Linear Algebra	5	T005
114	Philosophy of Mind	2	T011
115	Web Development	4	T012
116	Film Studies and Analysis	3	T003
117	Music Theory	2	T004
118	Theatre and Performance	3	T006
119	Modern Political Thought	3	T010
120	Cultural Anthropology	4	T011
121	Ethics and Society	1	T012
NULL	NULL	NULL	NULL

After

```
205 •   select * from Courses;
206
```

100% 23:205

Result Grid Filter Rows: Search Edit:

course_id	course_name	credits	teacher_id
102	Modern English Literature	3	T002
103	Data Structures and Algorithms	5	T001
104	World History	3	T003
105	Fundamentals of Physics	5	T004
106	Creative Writing	2	T002
107	Calculus	4	T005
108	Principles of Economics	3	T006
109	Introduction to Psychology	3	T007
110	Environmental Science	4	T008
111	Art and Design Basics	2	T009
112	Sociology and Culture	3	T010
113	Linear Algebra	5	T005
114	Philosophy of Mind	2	T011
115	Web Development	4	T001
116	Film Studies and Analysis	3	T003
117	Music Theory	2	T004
118	Theatre and Performance	3	T006
119	Modern Political Thought	3	T010
120	Cultural Anthropology	4	T011
121	Ethics and Society	1	T012
NULL	NULL	NULL	NULL

6. **Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.**

CODE :

delete from Students where student_id = 'S009';

Before:

```
209 •   select * from Enrollments;
210
```

100% 27:209

Result Grid Filter Rows: Search

enrollment_id	student_...	course_id	enrollment_da...
1	S004	116	2022-08-01
2	S010	121	2022-08-01
3	S001	101	2023-01-15
4	S007	120	2023-08-01
5	S002	102	2023-08-02
6	S009	118	2023-08-02
7	S006	113	2024-02-03
8	S003	105	2024-08-01
9	S008	115	2024-08-01
10	S005	112	2024-08-03
11	S004	104	2025-01-12
12	S010	114	2025-01-12
13	S001	103	2025-08-01
14	S007	110	2025-08-02
15	S002	106	2025-08-02
16	S009	102	2025-08-04
17	S006	107	2025-08-05
18	S003	113	2025-08-05
19	S008	108	2025-08-05
20	S005	109	2025-08-06
21	S011	112	2025-08-07
NULL	NULL	NULL	NULL

After:

```
245 •  select * from Enrollments;
246
```

100% 27:245

Result Grid Filter Rows: Search

enrollment_id	student_...	course_id	enrollment_da...
1	S004	116	2022-08-01
2	S010	121	2022-08-01
3	S001	101	2023-01-15
4	S007	120	2023-08-01
5	S002	102	2023-08-02
7	S006	113	2024-02-03
8	S003	105	2024-08-01
9	S008	115	2024-08-01
10	S005	112	2024-08-03
11	S004	104	2025-01-12
12	S010	114	2025-01-12
13	S001	103	2025-08-01
14	S007	110	2025-08-02
15	S002	106	2025-08-02
17	S006	107	2025-08-05
18	S003	113	2025-08-05
20	S005	109	2025-08-06
21	S011	112	2025-08-07
NULL	NULL	NULL	NULL

```
245 •  select * from Enrollments where student_id ='S009';
246
```

100% 52:245

Result Grid Filter Rows: Search Edit: Export

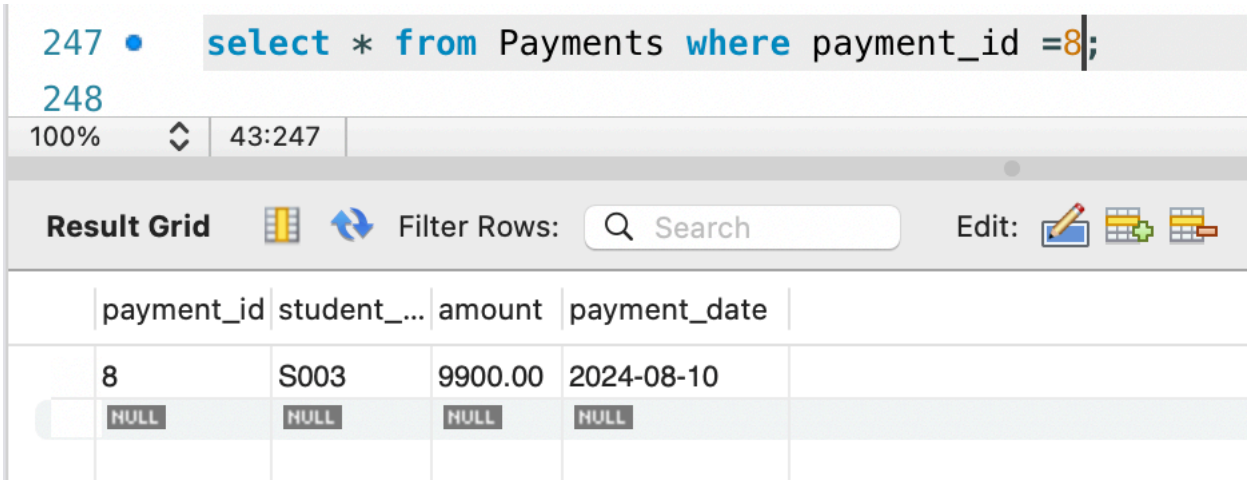
enrollment_id	student_...	course_id	enrollment_da...
NULL	NULL	NULL	NULL

7. **Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.**

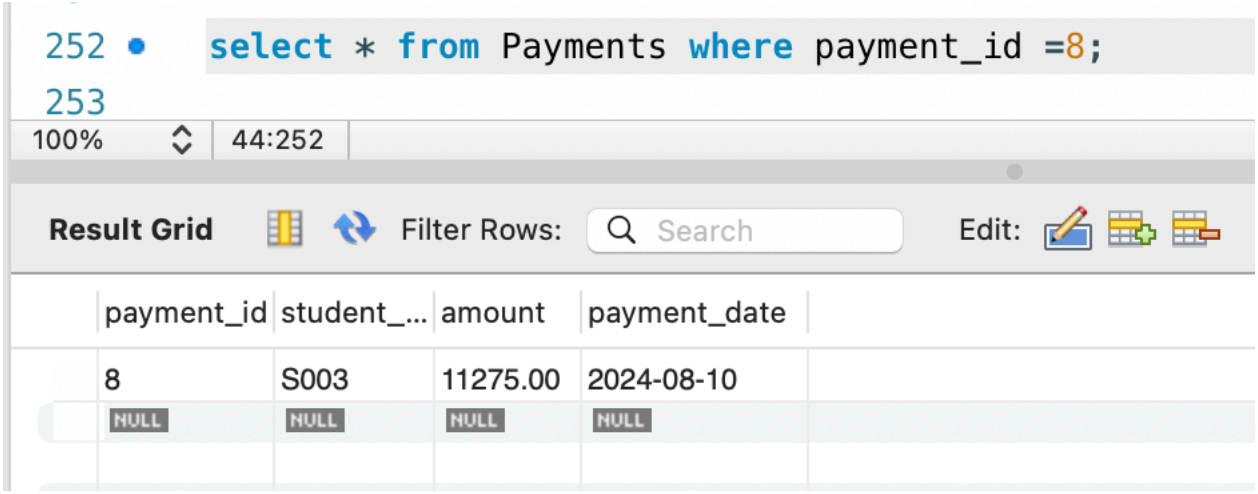
CODE :

```
update Payments
set amount = 11275 where payment_id = 8;
```

Before:



After:



TASK - 3

Task 3.

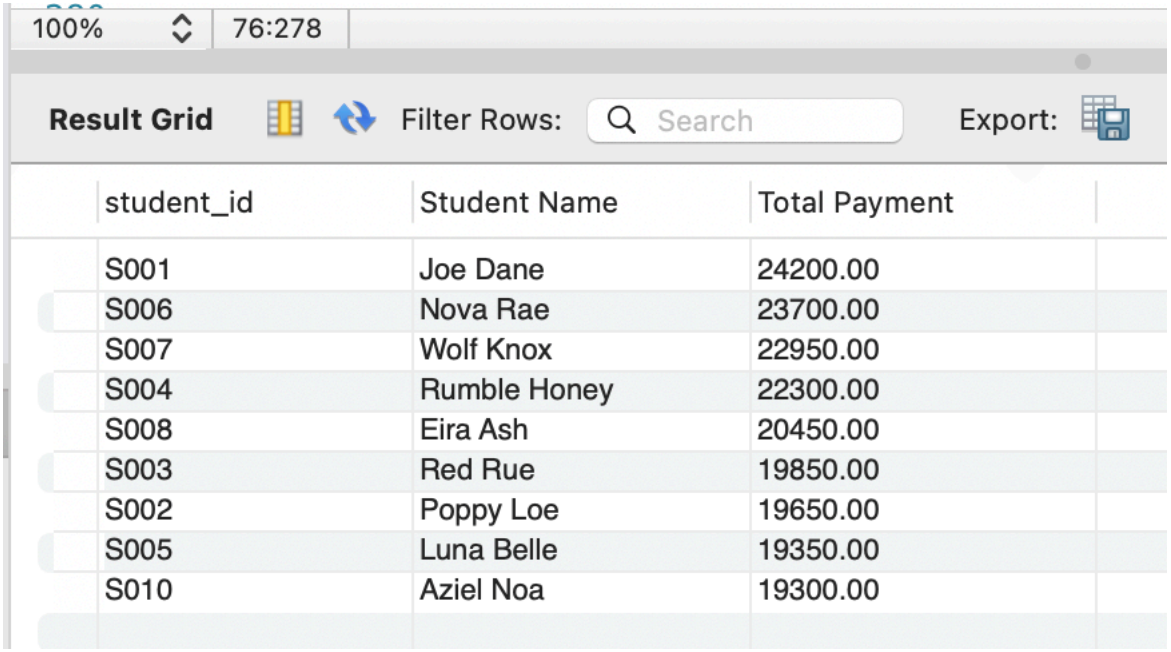
Aggregate functions, Having, Order By, GroupBy and Joins:

1. **Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.**

CODE :

For all students:

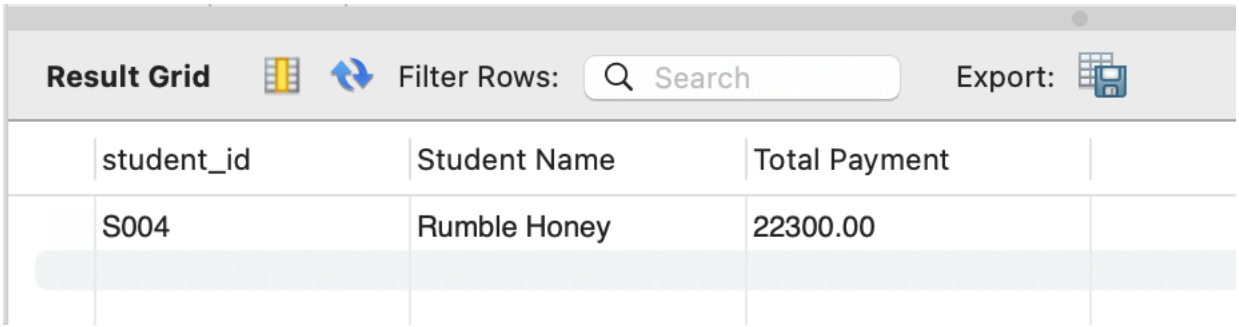
```
select Students.student_id, concat (Students.first_name," ",Students.last_name) as 'Student Name', sum(Payments.amount) 'Total Payment' from Students
join Payments
on Students.student_id = Payments.student_id  group by Students.student_id;
```



student_id	Student Name	Total Payment
S001	Joe Dane	24200.00
S006	Nova Rae	23700.00
S007	Wolf Knox	22950.00
S004	Rumble Honey	22300.00
S008	Eira Ash	20450.00
S003	Red Rue	19850.00
S002	Poppy Loe	19650.00
S005	Luna Belle	19350.00
S010	Aziel Noa	19300.00

For specific student:

```
select Students.student_id, concat (Students.first_name," ",Students.last_name) as 'Student Name', sum(Payments.amount) 'Total Payment' from Students
join Payments
on Students.student_id = Payments.student_id
where Students.student_id = 'S004'
group by Students.student_id;
```

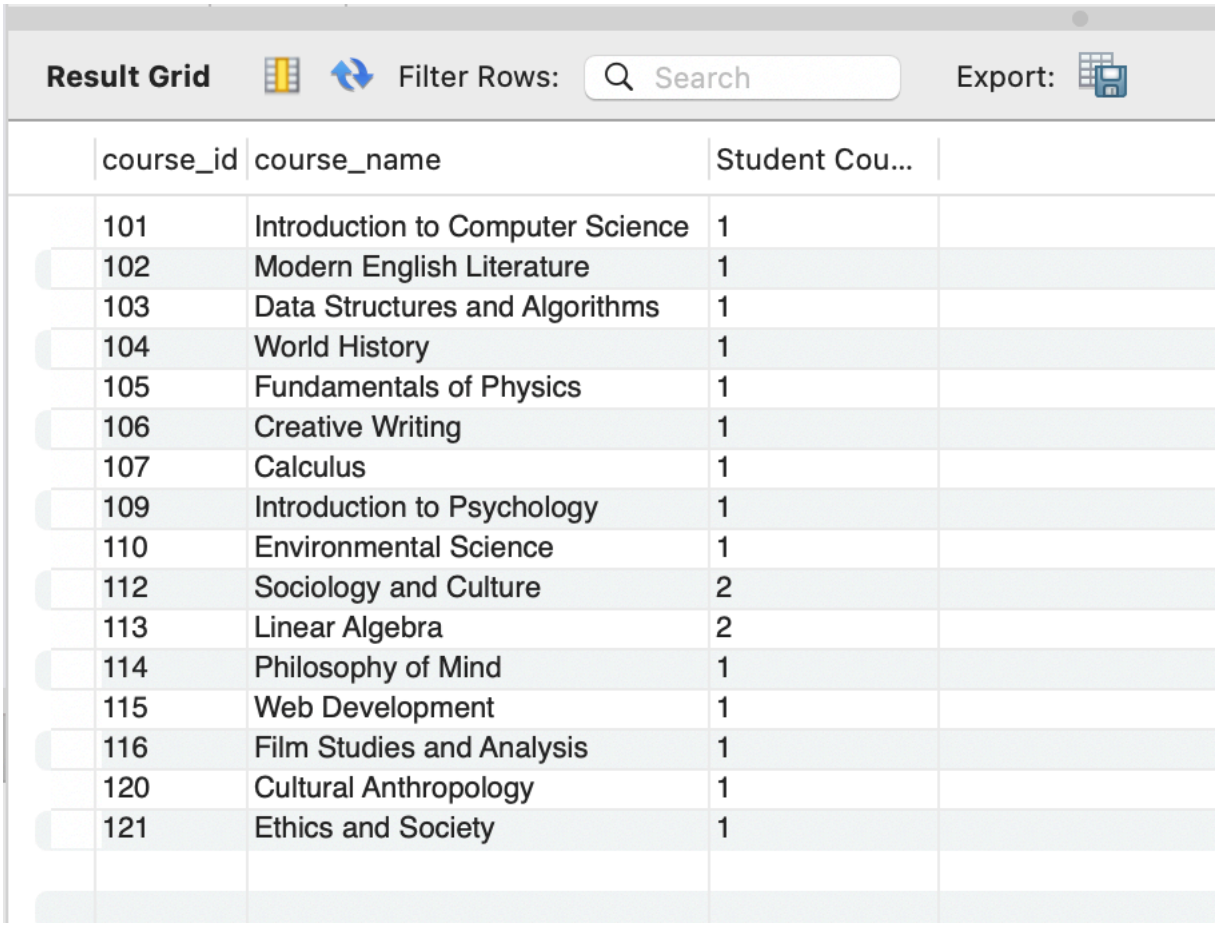


2. **Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.**

CODE :

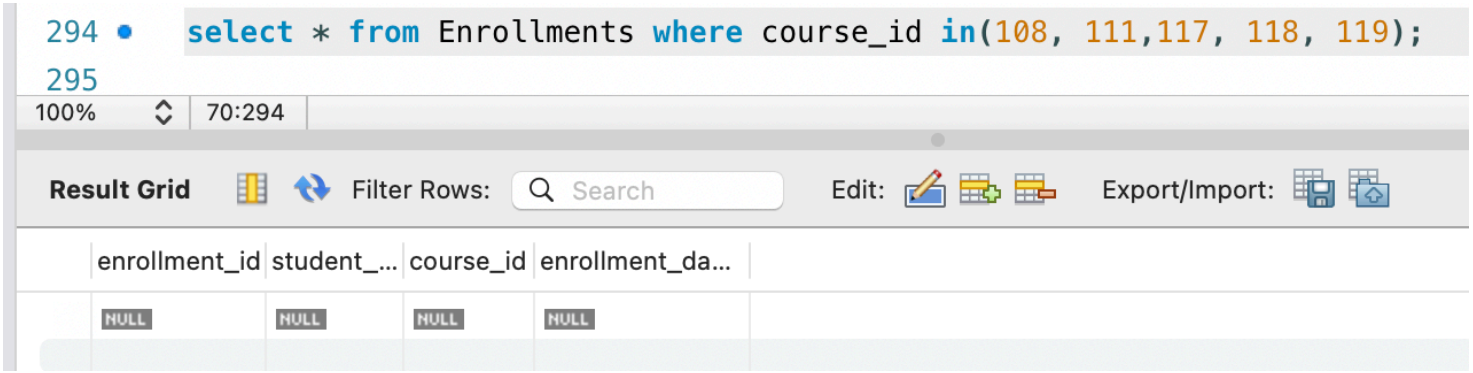
Join:

```
Select Courses.course_id, Courses.course_name, count(Enrollments.student_id) 'Student Count'
from Courses
join Enrollments
on Courses.course_id = Enrollments.course_id
group by Courses.course_id, Courses.course_name
order by Courses.course_id;
```



Result Grid Filter Rows: Search Export:

course_id	course_name	Student Cou...
101	Introduction to Computer Science	1
102	Modern English Literature	1
103	Data Structures and Algorithms	1
104	World History	1
105	Fundamentals of Physics	1
106	Creative Writing	1
107	Calculus	1
109	Introduction to Psychology	1
110	Environmental Science	1
112	Sociology and Culture	2
113	Linear Algebra	2
114	Philosophy of Mind	1
115	Web Development	1
116	Film Studies and Analysis	1
120	Cultural Anthropology	1
121	Ethics and Society	1



```
294 • select * from Enrollments where course_id in(108, 111,117, 118, 119);
295
```

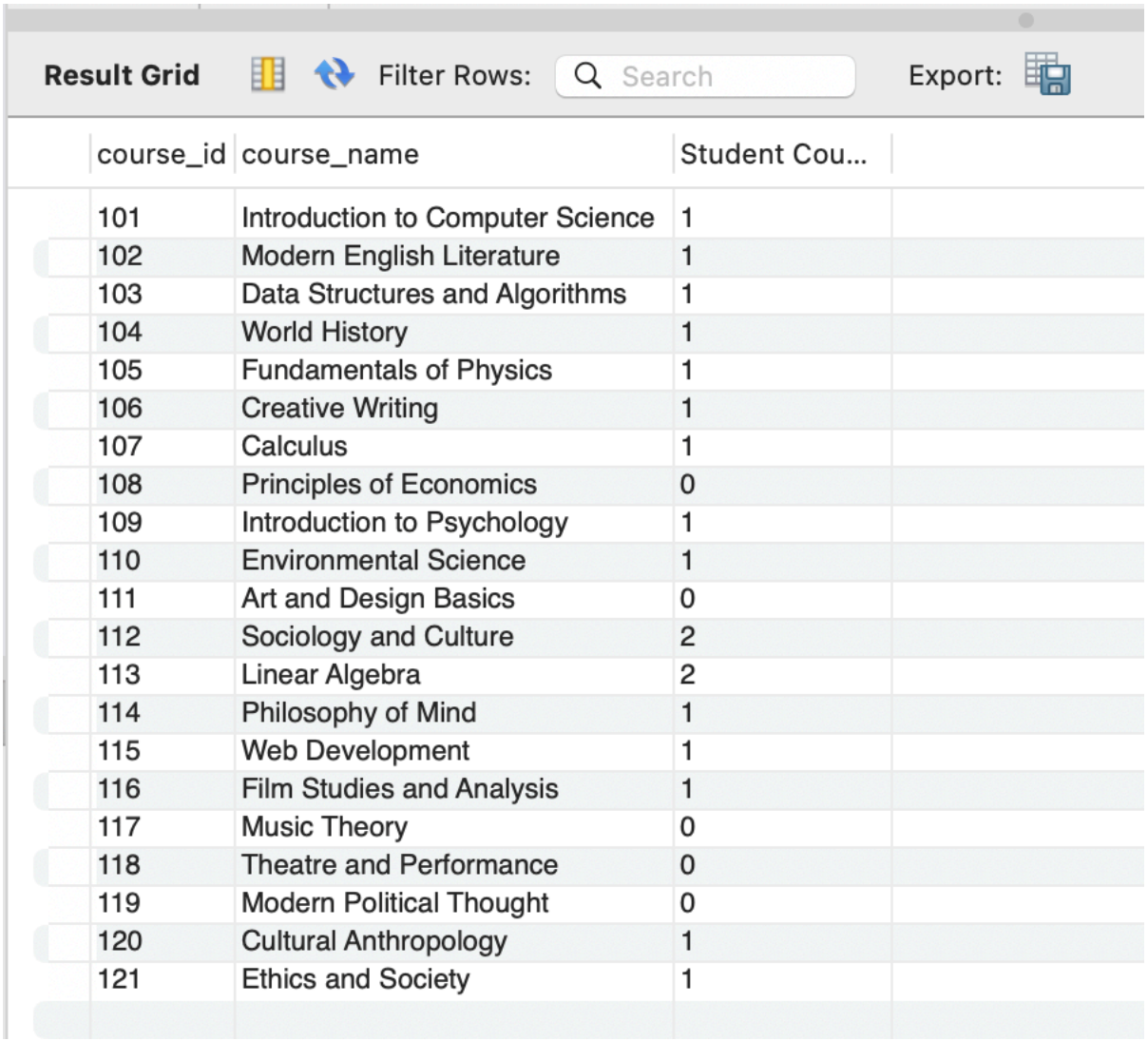
100% 70:294

Result Grid Filter Rows: Search Edit: Export/Import:

enrollment_id	student_...	course_id	enrollment_da...
NULL	NULL	NULL	NULL

Left Join:

Select Courses.course_id, Courses.course_name, count(Enrollments.student_id) 'Student Count'
from Courses
left join Enrollments
on Courses.course_id = Enrollments.course_id
group by Courses.course_id, Courses.course_name
order by Courses.course_id;



course_id	course_name	Student Cou...
101	Introduction to Computer Science	1
102	Modern English Literature	1
103	Data Structures and Algorithms	1
104	World History	1
105	Fundamentals of Physics	1
106	Creative Writing	1
107	Calculus	1
108	Principles of Economics	0
109	Introduction to Psychology	1
110	Environmental Science	1
111	Art and Design Basics	0
112	Sociology and Culture	2
113	Linear Algebra	2
114	Philosophy of Mind	1
115	Web Development	1
116	Film Studies and Analysis	1
117	Music Theory	0
118	Theatre and Performance	0
119	Modern Political Thought	0
120	Cultural Anthropology	1
121	Ethics and Society	1

3. **Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.student_id**

Inserted some extra students who have not enrolled in any courses yet :

```
273 •  insert into Students values
274    ('S012', "Camy", "Dove", "1999-07-15","camy.dove@gmail.com", 1524386790),
275    ('S013', "Lara", "Lin", "2003-08-25","lara.lin@gmail.com", 3774385770),
276    ('S014', "Daisy", "May", "2004-02-11","daisy.may@gmail.com", 5678382183);
277
278 •  select * from Students;
279
```

100% 24:278

Result Grid Filter Rows: Search Edit: Export/Import:

student_...	first_name	last_name	date_of_bir...	email	phone_number
S001	Joe	Dane	2001-05-10	joe.dane@gmail.com	7876543210
S002	Poppy	Loe	2000-08-22	poppy.loe@gmail.com	6123456780
S003	Red	Rue	2002-01-15	red.rue@gmail.com	8345612780
S004	Rumble	Honey	1999-12-05	rumble.honey@gmail.com	9988776655
S005	Luna	Belle	2001-03-25	luna.belle@gmail.com	4765432109
S006	Nova	Rae	2000-09-17	nova.rae@gmail.com	6887766554
S007	Wolf	Knox	2001-11-30	wolf.knox@gmail.com	1876512345
S008	Eira	Ash	2002-02-10	eira.ash@gmail.com	7212345678
S010	Aziel	Noa	1999-04-12	aziel.noa@gmail.com	124567890
S011	John	Doe	1995-08-15	john.doe@example.com	1234567890
S012	Camy	Dove	1999-07-15	camy.dove@gmail.com	1524386790
S013	Lara	Lin	2003-08-25	lara.lin@gmail.com	3774385770
S014	Daisy	May	2004-02-11	daisy.may@gmail.com	5678382183
NULL	NULL	NULL	NULL	NULL	NULL

CODE :

First option:

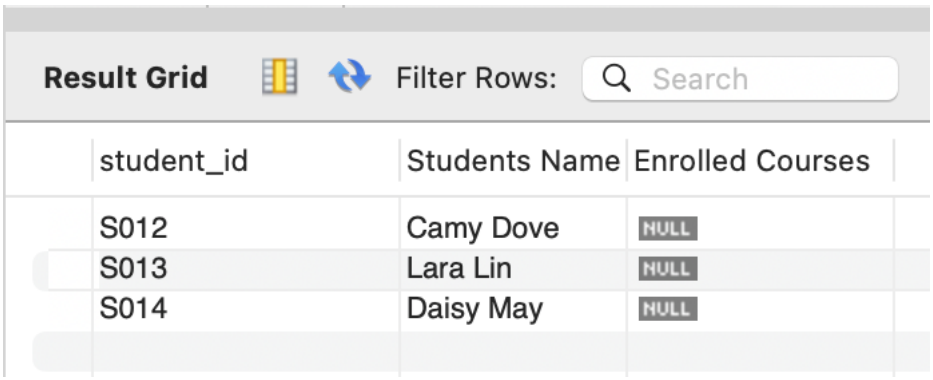
```
Select Students.student_id, concat (Students.first_name," ",Students.last_name) as 'Not
Enrolled Students List', Enrollments.course_id 'Enrolled Courses'  from Students
left join Enrollments
on Students.student_id = Enrollments.student_id
where Enrollments.course_id is null;
```

Result Grid Filter Rows: Search

Not Enrolled Students List
Camy Dove
Lara Lin
Daisy May

Second option:

```
Select Students.student_id, concat (Students.first_name," ",Students.last_name) as 'Students
Name', Enrollments.course_id 'Enrolled Courses'  from Students
left join Enrollments
on Students.student_id = Enrollments.student_id
where Enrollments.course_id is null;
```



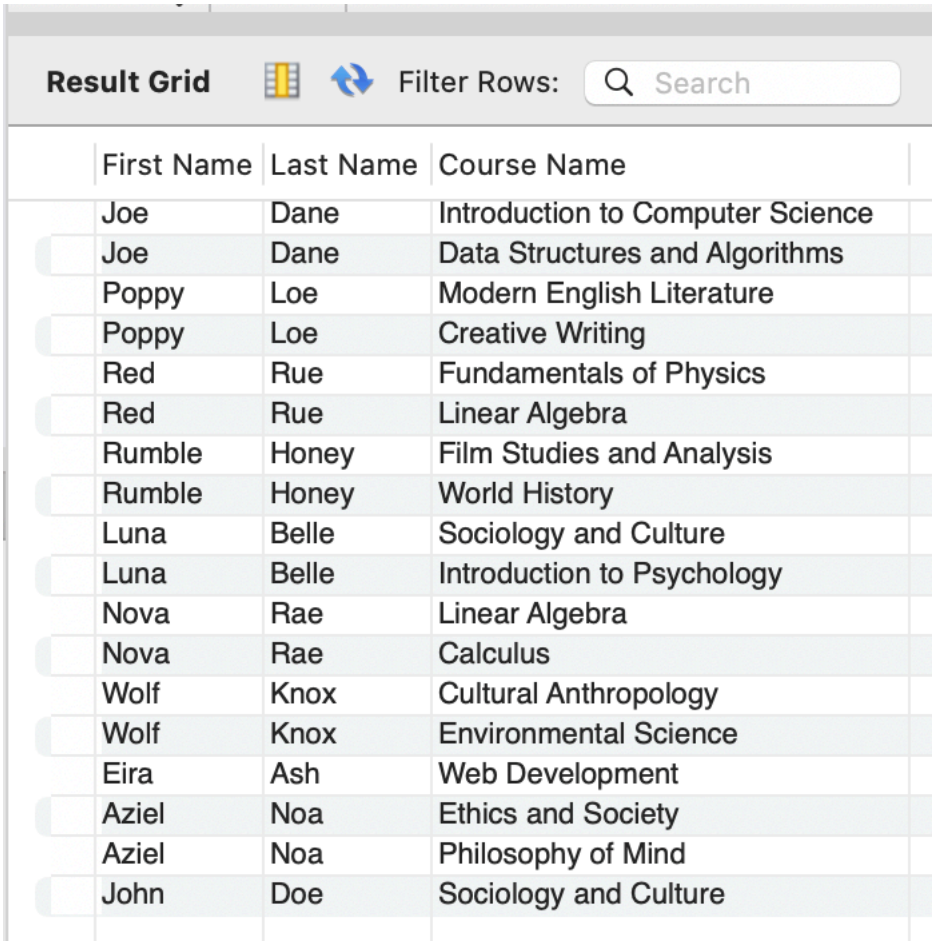
student_id	Students Name	Enrolled Courses
S012	Camy Dove	NULL
S013	Lara Lin	NULL
S014	Daisy May	NULL

4. **Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.**

CODE :

1st: (Ordered by id)

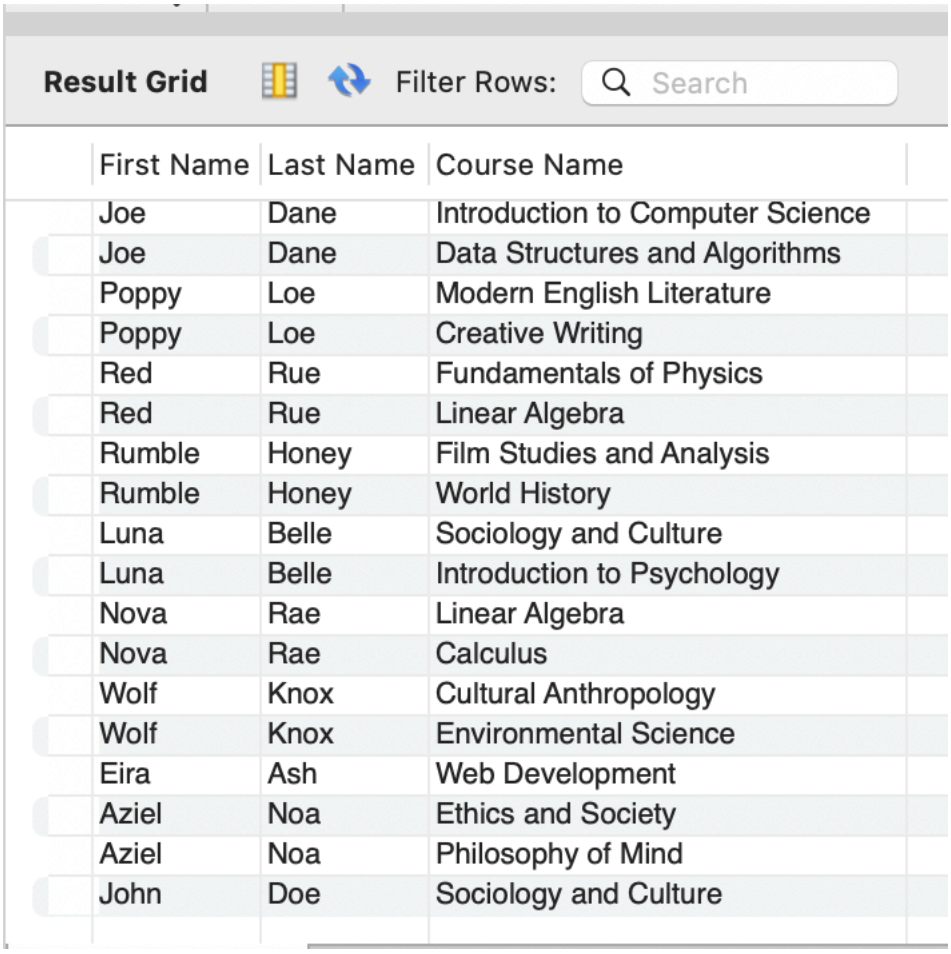
```
Select s.first_name 'First Name', s.last_name 'Last Name', c.course_name 'Course Name'
from Students as s
join Enrollments as e
on s.student_id = e.student_id
join Courses as c
on e.course_id = c.course_id
order by e.student_id;
```



First Name	Last Name	Course Name
Joe	Dane	Introduction to Computer Science
Joe	Dane	Data Structures and Algorithms
Poppy	Loe	Modern English Literature
Poppy	Loe	Creative Writing
Red	Rue	Fundamentals of Physics
Red	Rue	Linear Algebra
Rumble	Honey	Film Studies and Analysis
Rumble	Honey	World History
Luna	Belle	Sociology and Culture
Luna	Belle	Introduction to Psychology
Nova	Rae	Linear Algebra
Nova	Rae	Calculus
Wolf	Knox	Cultural Anthropology
Wolf	Knox	Environmental Science
Eira	Ash	Web Development
Aziel	Noa	Ethics and Society
Aziel	Noa	Philosophy of Mind
John	Doe	Sociology and Culture

2st: (Unordered by id)

```
Select s.first_name 'First Name', s.last_name 'Last Name', c.course_name 'Course Name'
from Students as s
join Enrollments as e
on s.student_id = e.student_id
join Courses as c
on e.course_id = c.course_id;
```



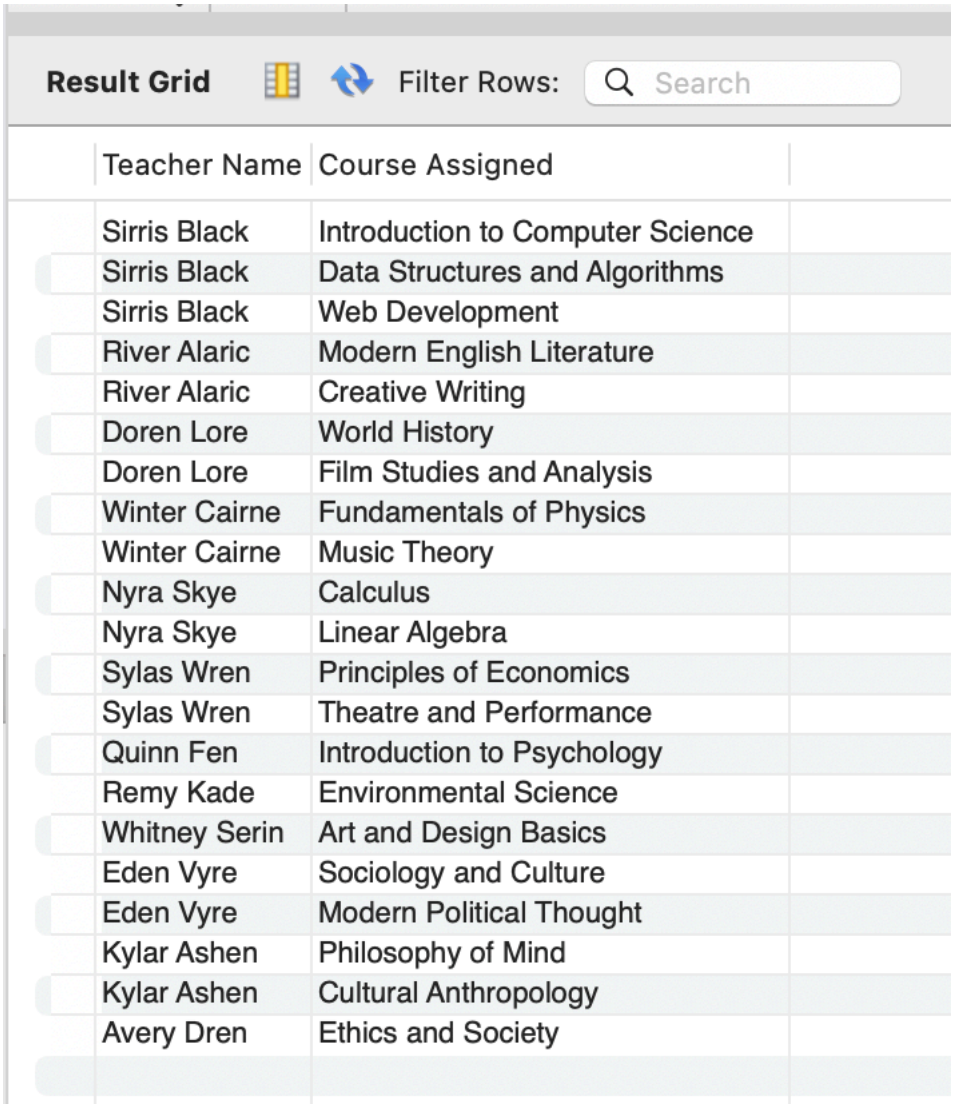
Result Grid Filter Rows: Search

First Name	Last Name	Course Name
Joe	Dane	Introduction to Computer Science
Joe	Dane	Data Structures and Algorithms
Poppy	Loe	Modern English Literature
Poppy	Loe	Creative Writing
Red	Rue	Fundamentals of Physics
Red	Rue	Linear Algebra
Rumble	Honey	Film Studies and Analysis
Rumble	Honey	World History
Luna	Belle	Sociology and Culture
Luna	Belle	Introduction to Psychology
Nova	Rae	Linear Algebra
Nova	Rae	Calculus
Wolf	Knox	Cultural Anthropology
Wolf	Knox	Environmental Science
Eira	Ash	Web Development
Aziel	Noa	Ethics and Society
Aziel	Noa	Philosophy of Mind
John	Doe	Sociology and Culture

5. **Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.**

CODE :

```
Select concat(t.first_name," ",t.last_name) 'Teacher Name', c.course_name 'Course Assigned'
from Teacher as t
join Courses as c
on t.teacher_id = c.teacher_id;
```



Teacher Name	Course Assigned
Sirris Black	Introduction to Computer Science
Sirris Black	Data Structures and Algorithms
Sirris Black	Web Development
River Alaric	Modern English Literature
River Alaric	Creative Writing
Doren Lore	World History
Doren Lore	Film Studies and Analysis
Winter Cairne	Fundamentals of Physics
Winter Cairne	Music Theory
Nyra Skye	Calculus
Nyra Skye	Linear Algebra
Sylas Wren	Principles of Economics
Sylas Wren	Theatre and Performance
Quinn Fen	Introduction to Psychology
Remy Kade	Environmental Science
Whitney Serin	Art and Design Basics
Eden Vyre	Sociology and Culture
Eden Vyre	Modern Political Thought
Kylar Ashen	Philosophy of Mind
Kylar Ashen	Cultural Anthropology
Avery Dren	Ethics and Society

Inserted some teachers who don’t handle any courses

```
326 •  insert into Teacher
327    values
328    ('T013', 'Gwen', 'Skye', 'gwen.skye@gmail.com'),
329    ('T014', 'Chrollo', 'Lucilfer', 'chrollo.lucilfer@gmail.com'),
330    ('T015', 'Killua', 'Zoldyck', 'killua.zoldyck@gmail.com');
331
332 •  Select * from Teacher;
333
```

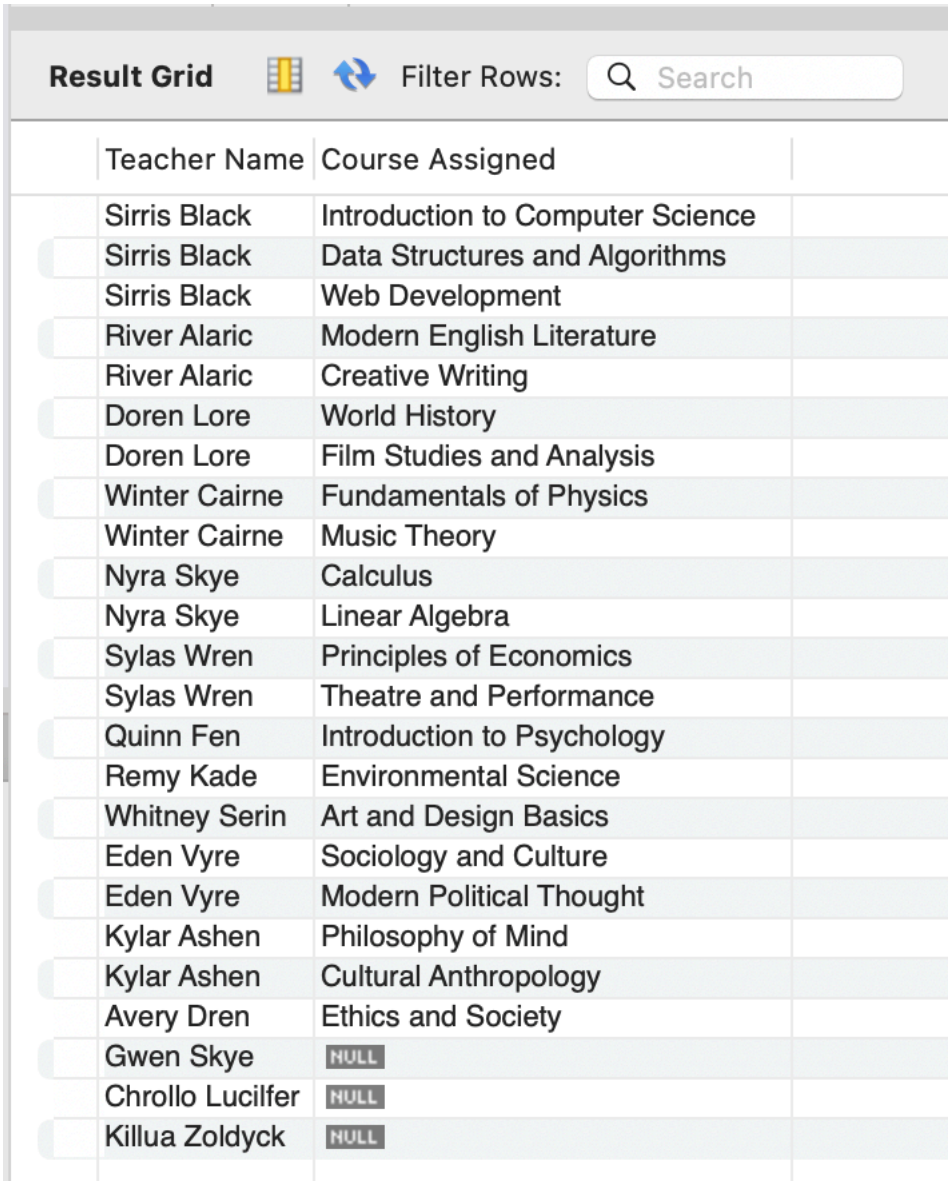
100% 23:332

Result Grid Filter Rows: Search Edit: Export/Import:

teacher_id	first_name	last_name	email
T001	Sirris	Black	sirris.black@gmailmail.com
T002	River	Alaric	river.alaric@gmailmail.com
T003	Doren	Lore	doren.lore@gmailmail.com
T004	Winter	Cairne	dr.wintercairne
T005	Nyra	Skye	nyra.skye@gmailmail.com
T006	Sylas	Wren	sylas.wren@gmailmail.com
T007	Quinn	Fen	quinn.fen@gmailmail.com
T008	Remy	Kade	remy.kade@gmailmail.com
T009	Whitney	Serin	whitney.serin@gmailmail.com
T010	Eden	Vyre	eden.vyre@gmailmail.com
T011	Kylar	Ashen	kylar.ashen@gmailmail.com
T012	Avery	Dren	avery.dren@gmailmail.com
T013	Gwen	Skye	gwen.skye@gmail.com
T014	Chrollo	Lucilfer	chrollo.lucilfer@gmail.com
T015	Killua	Zoldyck	killua.zoldyck@gmail.com
NULL	NULL	NULL	NULL

CODE : Now some teachers are not assigned to any course

```
Select concat(t.first_name," ",t.last_name) 'Teacher Name', c.course_name 'Course Assigned'
from Teacher as t
left join Courses as c
on t.teacher_id = c.teacher_id
order by t.teacher.id;
```



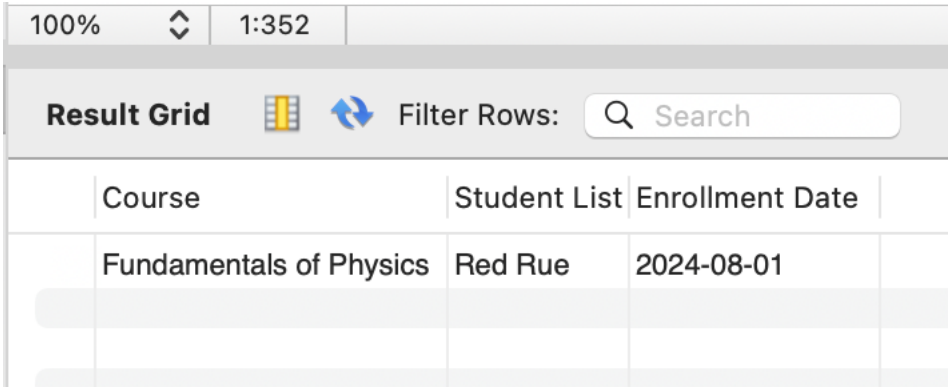
Result Grid | Filter Rows: Search

Teacher Name	Course Assigned
Sirris Black	Introduction to Computer Science
Sirris Black	Data Structures and Algorithms
Sirris Black	Web Development
River Alaric	Modern English Literature
River Alaric	Creative Writing
Doren Lore	World History
Doren Lore	Film Studies and Analysis
Winter Cairne	Fundamentals of Physics
Winter Cairne	Music Theory
Nyra Skye	Calculus
Nyra Skye	Linear Algebra
Sylas Wren	Principles of Economics
Sylas Wren	Theatre and Performance
Quinn Fen	Introduction to Psychology
Remy Kade	Environmental Science
Whitney Serin	Art and Design Basics
Eden Vyre	Sociology and Culture
Eden Vyre	Modern Political Thought
Kylar Ashen	Philosophy of Mind
Kylar Ashen	Cultural Anthropology
Avery Dren	Ethics and Society
Gwen Skye	NULL
Chrollo Lucilfer	NULL
Killua Zoldyck	NULL

6. **Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.**

CODE :

```
Select c.course_name 'Course', concat(s.first_name, " ", s.last_name) 'Student List' ,
e.enrollment_date  'Enrollment Date'
from Students as s
join Enrollments as e
on s.student_id = e.student_id
join Courses as c
on e.course_id = c.course_id
where c.course_id = 105;
```



Inserted some students into same course on enrollment to have multiple students in a course

```
350 •  insert into Enrollments
351    values ( null,'S005', 115, '2025-08-15'),
352    ( null,'S002', 115, '2025-08-15'),
353    ( null,'S010', 115, '2025-08-16'),
354    ( null,'S007', 115, '2025-08-17');
355
356 •  select * from Enrollments;
```

100% 35:354

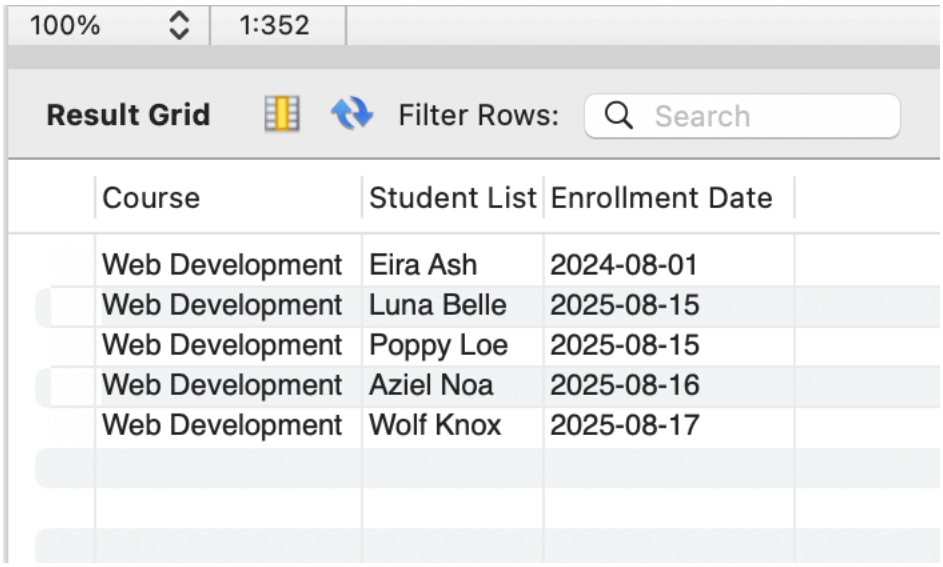
Result Grid Filter Rows: Search Edit:

enrollment_id	student_...	course_id	enrollment_da...
1	S004	116	2022-08-01
2	S010	121	2022-08-01
3	S001	101	2023-01-15
4	S007	120	2023-08-01
5	S002	102	2023-08-02
7	S006	113	2024-02-03
8	S003	105	2024-08-01
9	S008	115	2024-08-01
10	S005	112	2024-08-03
11	S004	104	2025-01-12
12	S010	114	2025-01-12
13	S001	103	2025-08-01
14	S007	110	2025-08-02
15	S002	106	2025-08-02
17	S006	107	2025-08-05
18	S003	113	2025-08-05
20	S005	109	2025-08-06
21	S011	112	2025-08-07
22	S005	115	2025-08-15
23	S002	115	2025-08-15
24	S010	115	2025-08-16
25	S007	115	2025-08-17
NULL	NULL	NULL	NULL

CODE :

```
Select c.course_name 'Course', concat(s.first_name, " ", s.last_name) 'Student List' ,
e.enrollment_date  'Enrollment Date'
from Students as s
```

```
join Enrollments as e
on s.student_id = e.student_id
join Courses as c
on e.course_id = c.course_id
where c.course_name = "Web Development";
```



Course	Student List	Enrollment Date
Web Development	Eira Ash	2024-08-01
Web Development	Luna Belle	2025-08-15
Web Development	Poppy Loe	2025-08-15
Web Development	Aziel Noa	2025-08-16
Web Development	Wolf Knox	2025-08-17

7. **Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.**

I have deleted student 7 to have some non paid student who registed in course.

```
delete from Payments
where student_id='S007';

select * from payments order by student_id;
```

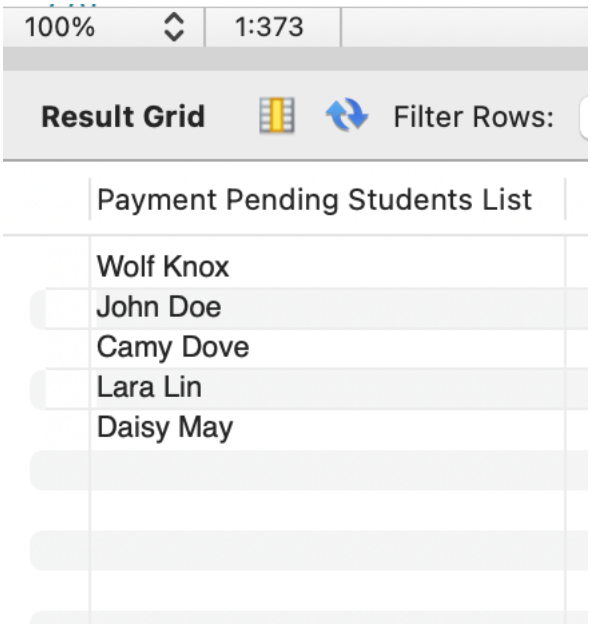
100% 44:371

Result Grid Filter Rows: Search Edit:

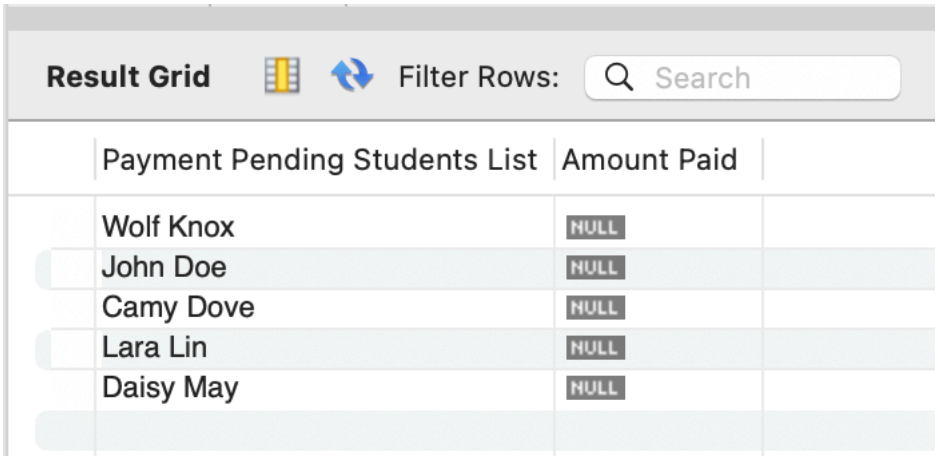
payment_id	student_...	amount	payment_date
1003	S001	12000.00	2023-01-20
1013	S001	12200.00	2025-08-01
1005	S002	9800.00	2023-08-08
1015	S002	9850.00	2025-08-04
1008	S003	9900.00	2024-08-10
1018	S003	9950.00	2025-08-08
1001	S004	11000.00	2022-08-05
1011	S004	11300.00	2025-01-20
1010	S005	9700.00	2024-08-15
1020	S005	9650.00	2025-08-10
1007	S006	11800.00	2024-02-05
1017	S006	11900.00	2025-08-07
1009	S008	10150.00	2024-08-12
1019	S008	10300.00	2025-08-09
1002	S010	9500.00	2022-08-10
1012	S010	9800.00	2025-01-25
NULL	NULL	NULL	NULL

CODE :

```
select concat (Students.first_name," ",Students.last_name) as 'Payment Pending Students List'
from Students
left join Payments
on Students.student_id = Payments.student_id
where Payments.student_id is null;
```




```
select concat (Students.first_name," ",Students.last_name) as 'Payment Pending Students List',
Payments.amount 'Amount Paid'
from Students
left join Payments
on Students.student_id = Payments.student_id
where Payments.student_id is null;
```

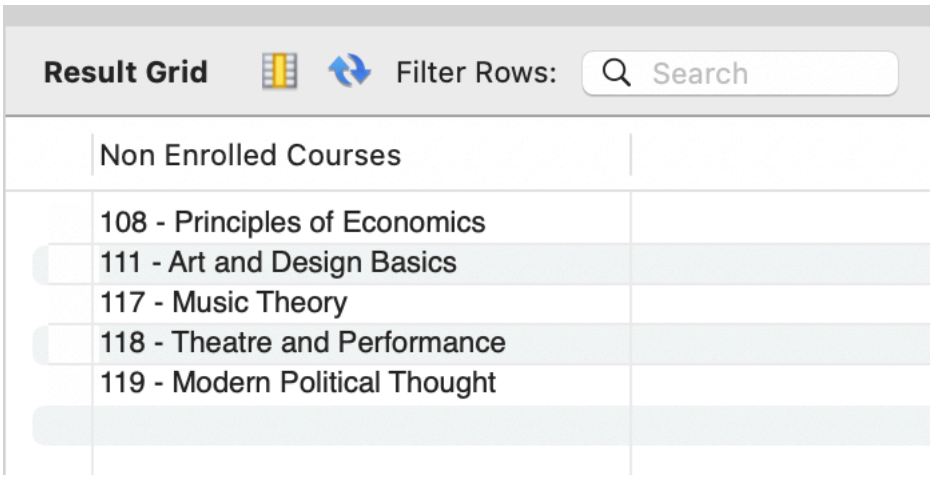


8. **Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.**

CODE :

```
Select concat(c.course_id," - ",c.course_name) 'Non Enrolled Courses'
```

```
from Courses c
left join Enrollments e
on c.course_id = e.course_id
where e.course_id is null;
```



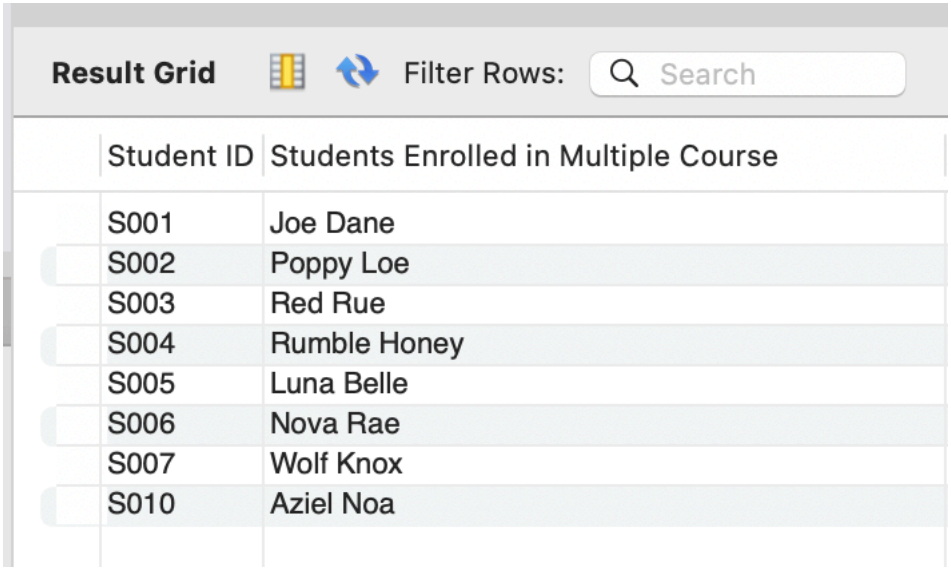
Result Grid Filter Rows: Search

Non Enrolled Courses
108 - Principles of Economics
111 - Art and Design Basics
117 - Music Theory
118 - Theatre and Performance
119 - Modern Political Thought

9. **Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.**

CODE :

```
select distinct e1.student_id 'Student ID', concat (s.first_name," ",s.last_name)  'Students Enrolled in Multiple Course' from Enrollments e1
join  Enrollments e2
on e1.student_id = e2.student_id and e1.course_id != e2.course_id
join Students s
on e2.student_id = s.student_id
order by e1.student_id;
```



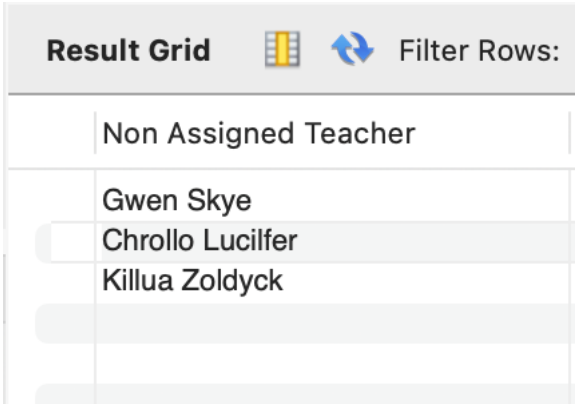
Result Grid Filter Rows: Search

Student ID	Students Enrolled in Multiple Course
S001	Joe Dane
S002	Poppy Loe
S003	Red Rue
S004	Rumble Honey
S005	Luna Belle
S006	Nova Rae
S007	Wolf Knox
S010	Aziel Noa

10. **Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.**

CODE :

```
Select concat(t.first_name," ",t.last_name) 'Non Assigned Teacher'
from Teacher as t
left join Courses as c
on t.teacher_id = c.teacher_id
where c.teacher_id is null;
```



Result Grid Filter Rows:

Non Assigned Teacher
Gwen Skye
Chrollo Lucilfer
Killua Zoldyck

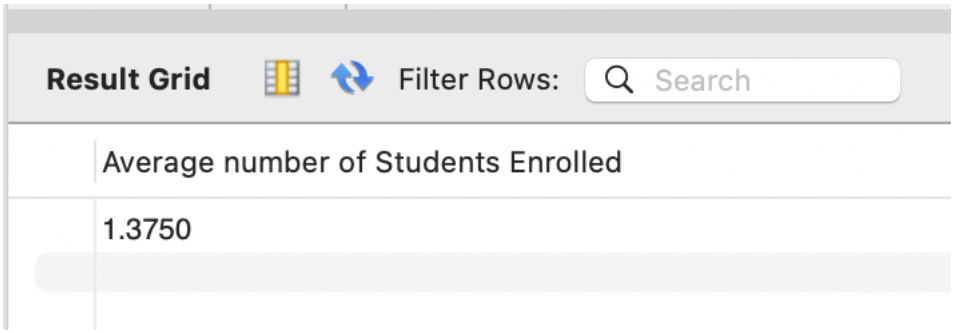
TASK 4

Task 4.
Subquery and its type:

1. Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.

CODE :

```
select avg(Student_Count_per_Course) 'Average number of Students Enrolled'
from (select Courses.course_id, count(Enrollments.student_id) Student_Count_per_Course
from Enrollments
right join Courses
on Enrollments.course_id = Courses.course_id
group by course_id) Enrollments_per_Course;
```



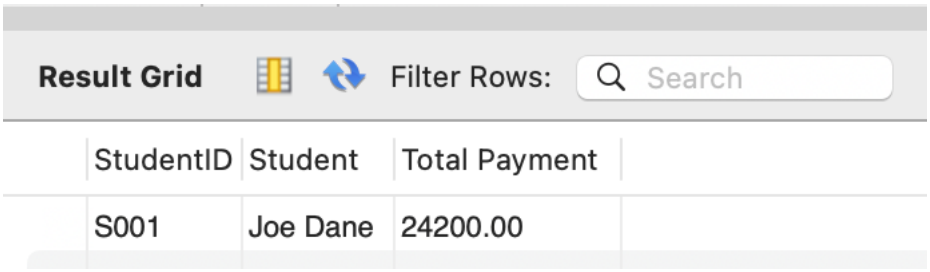
2. Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.

CODE :
Calculating the sum of all payments made by each Student and finding the max

```
select s.student_id StudentID,
        concat(s.first_name," ",s.last_name) 'Student',
    Student_Total_Individual_Payment.Total_Amount 'Total Payment'
from Students s
join (select student_id, sum(amount) Total_Amount
        from Payments
```

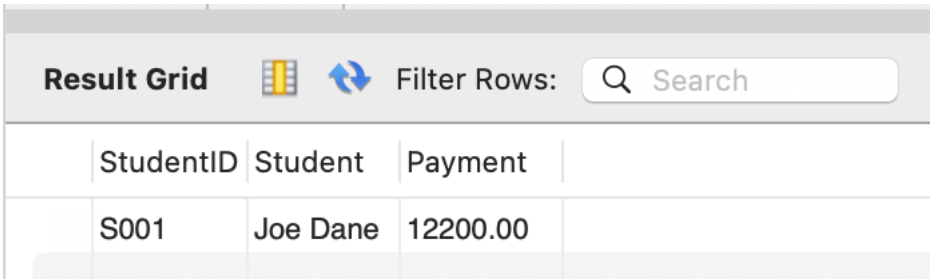
```
        group by student_id) Student_Total_Individual_Payment
on s.student_id = Student_Total_Individual_Payment.student_id

where Student_Total_Individual_Payment.Total_Amount = (select max(Total_Amount)
                                                       from (select student_id , sum(amount)
Total_Amount
                                                       from Payments
                                                       group by student_id)
Student_Total_Individual_Payment);
```



Not summing all payments from a Single Student

```
select s.student_id  StudentID, concat(s.first_name," ",s.last_name) 'Student', p.amount
Payment
from Students s
join Payments p
on s.student_id = p.student_id
where p.amount = (select max(amount) from Payments);
```

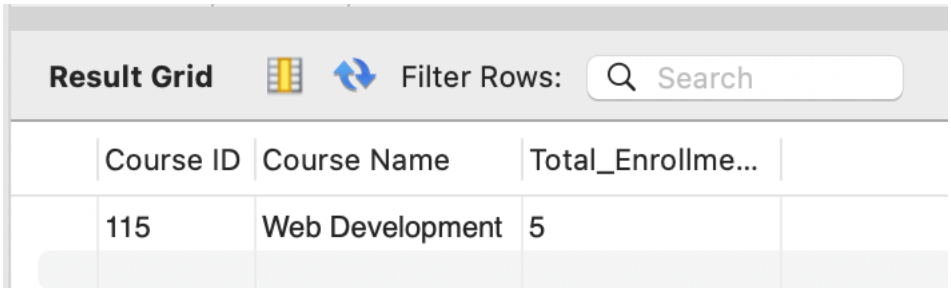


3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find

the course(s) with the maximum enrollment count.

CODE :

```
select c.course_id 'Course ID',
       c.course_name 'Course Name',
       Cousre_Enrollment.Course_count 'Total_Enrollment'
from Courses c
join (select course_id, count(*) Course_count
         from Enrollments
         group by course_id) Cousre_Enrollment
on c.course_id = Cousre_Enrollment.course_id
where Cousre_Enrollment.Course_count = (select max(Course_count)
                                                    from (select course_id, count(*) Course_count
                                                    from Enrollments
                                                    group by course_id) Cousre_Enrollment);
```

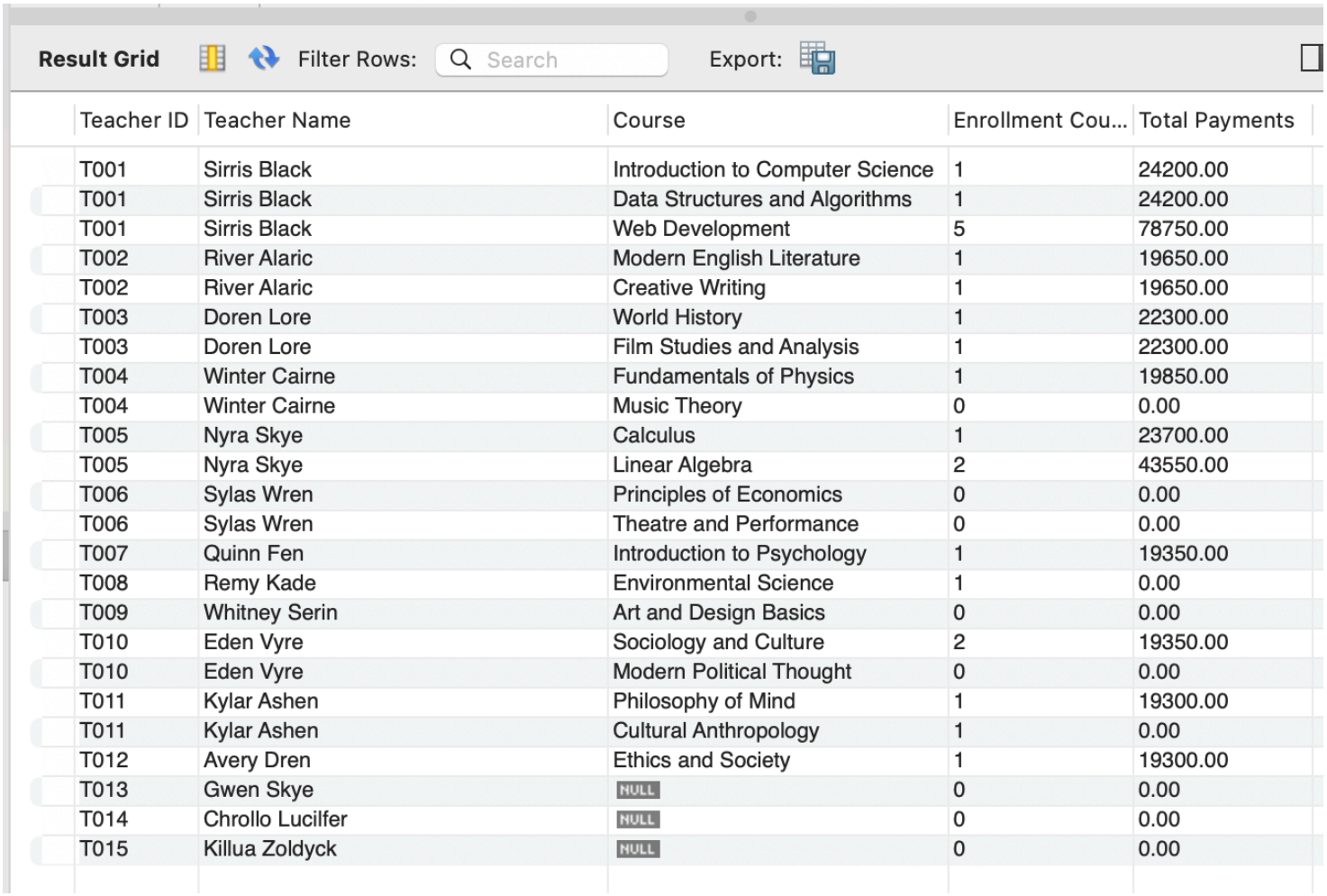


4. Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.

CODE : For each Course

```
select t.teacher_id 'Teacher ID',
       concat(t.first_name," ", t.last_name) 'Teacher Name',
       c.course_name 'Course',
       count(distinct e.enrollment_id) 'Enrollment Count',
       ifnull(SUM(p.amount),0) 'Total Payments'
from Teacher t
```

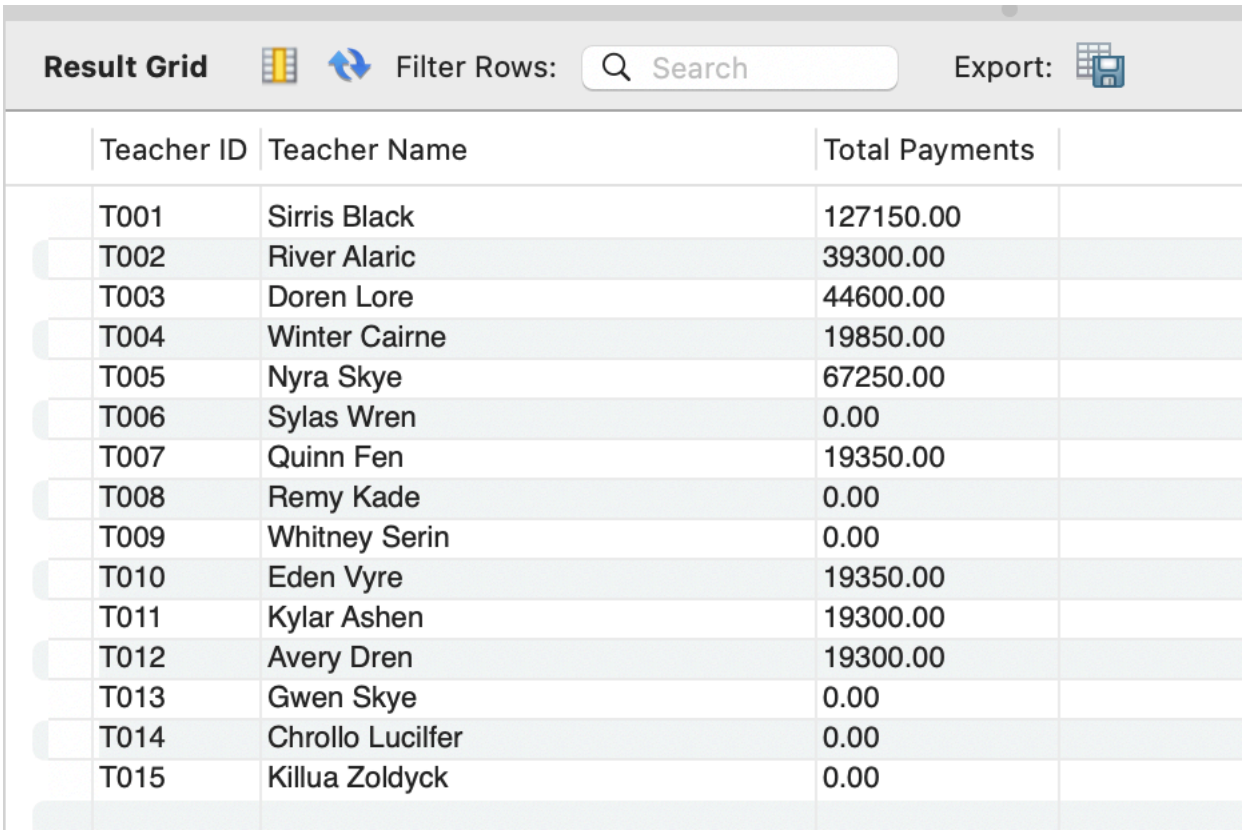
```
left join Courses c
        on  t.teacher_id = c.teacher_id
left join Enrollments e
        on c.course_id = e.course_id
left join Payments p
        on e.student_id = p.student_id
group by t.teacher_id, t.first_name, t.last_name, c.course_id, c.course_name
order by t.teacher_id, c.course_id;
```



Result Grid Filter Rows: Search Export:

Teacher ID	Teacher Name	Course	Enrollment Cou...	Total Payments
T001	Sirris Black	Introduction to Computer Science	1	24200.00
T001	Sirris Black	Data Structures and Algorithms	1	24200.00
T001	Sirris Black	Web Development	5	78750.00
T002	River Alaric	Modern English Literature	1	19650.00
T002	River Alaric	Creative Writing	1	19650.00
T003	Doren Lore	World History	1	22300.00
T003	Doren Lore	Film Studies and Analysis	1	22300.00
T004	Winter Cairne	Fundamentals of Physics	1	19850.00
T004	Winter Cairne	Music Theory	0	0.00
T005	Nyra Skye	Calculus	1	23700.00
T005	Nyra Skye	Linear Algebra	2	43550.00
T006	Sylas Wren	Principles of Economics	0	0.00
T006	Sylas Wren	Theatre and Performance	0	0.00
T007	Quinn Fen	Introduction to Psychology	1	19350.00
T008	Remy Kade	Environmental Science	1	0.00
T009	Whitney Serin	Art and Design Basics	0	0.00
T010	Eden Vyre	Sociology and Culture	2	19350.00
T010	Eden Vyre	Modern Political Thought	0	0.00
T011	Kylar Ashen	Philosophy of Mind	1	19300.00
T011	Kylar Ashen	Cultural Anthropology	1	0.00
T012	Avery Dren	Ethics and Society	1	19300.00
T013	Gwen Skye	NULL	0	0.00
T014	Chrollo Lucilfer	NULL	0	0.00
T015	Killua Zoldyck	NULL	0	0.00

Total Payment of each Teacher (regardless of course)

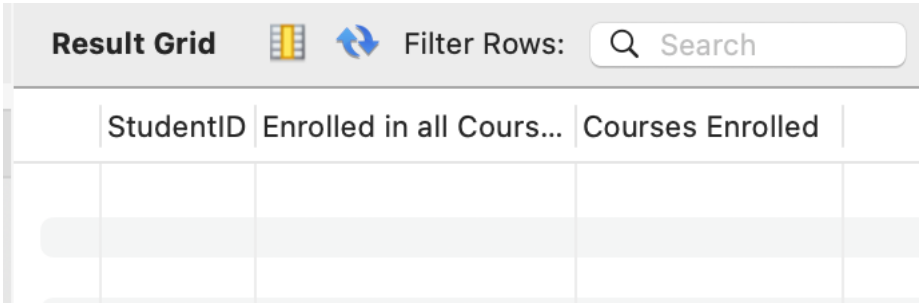

Result Grid Filter Rows: Search Export:

Teacher ID	Teacher Name	Total Payments
T001	Sirris Black	127150.00
T002	River Alaric	39300.00
T003	Doren Lore	44600.00
T004	Winter Cairne	19850.00
T005	Nyra Skye	67250.00
T006	Sylas Wren	0.00
T007	Quinn Fen	19350.00
T008	Remy Kade	0.00
T009	Whitney Serin	0.00
T010	Eden Vyre	19350.00
T011	Kylar Ashen	19300.00
T012	Avery Dren	19300.00
T013	Gwen Skye	0.00
T014	Chrollo Lucilfer	0.00
T015	Killua Zoldyck	0.00

5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.

CODE :

```
select s.student_id StudentID,
        concat(s.first_name," ",s.last_name) 'Enrolled in all Courses',
    count(distinct e.course_id) 'Courses Enrolled'
from Students s
join Enrollments e
on s.student_id = e.student_id
group by s.student_id
having count(distinct e.course_id) = (select count(*) from Courses);
```



6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.

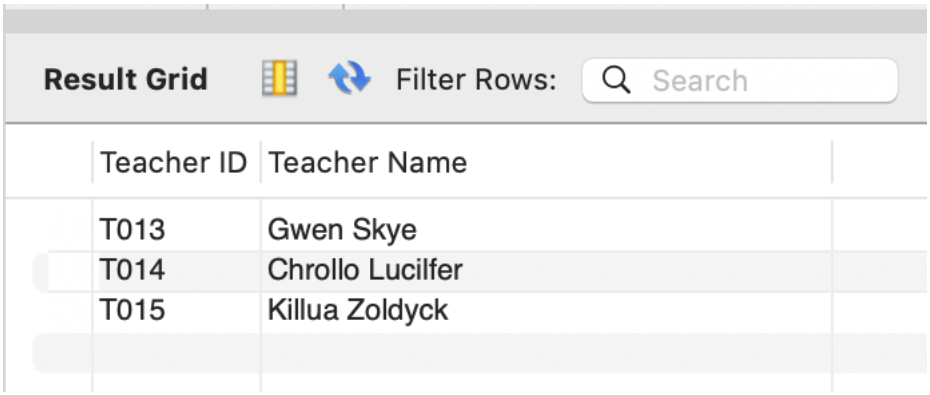
CODE :

```
Select teacher_id 'Teacher ID',
      concat(first_name," ", last_name) 'Teacher Name'
from Teacher
where teacher_id not in(Select distinct teacher_id
                                    from Courses);
```

Using Correlated Subquery

```
Select teacher_id 'Teacher ID',
      concat(first_name," ", last_name) 'Teacher Name'
from Teacher
where not exists (Select *
                            from Courses
                  where Teacher.teacher_id = Courses.teacher_id);
```

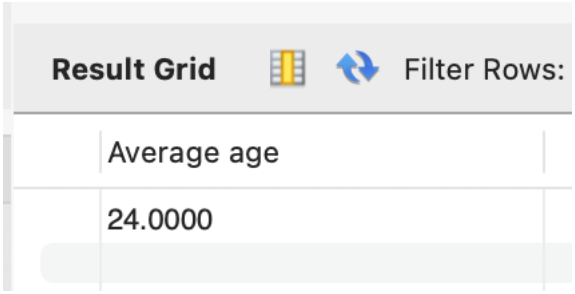
Result for both



7. Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.

CODE :

```
Select avg(Student_Ages.Age) 'Average age'
from ( select student_id, timestampdiff(year, date_of_birth, current_date()) Age
      from Students) Student_Ages;
```



Result Grid Filter Rows:

Average age
24.0000

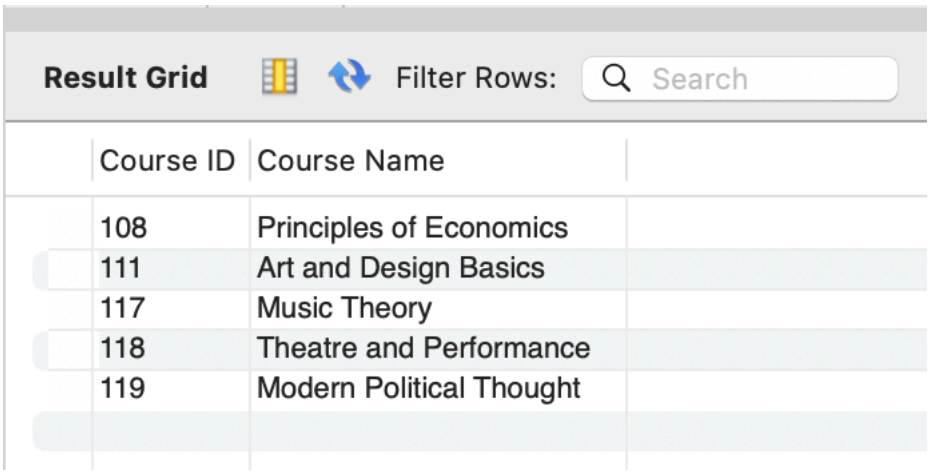
8. Identify courses with no enrollments. Use subqueries to find courses without enrollment records.

CODE :

```
Select course_id 'Course ID',
      course_name 'Course Name'
from Courses
where course_id not in(Select distinct course_id
                                    from Enrollments);
```

Using Correlated Subquery

```
Select course_id 'Course ID',
      course_name 'Course Name'
from Courses
where not exists (Select *
                           from Enrollments
            where Courses.course_id  = Enrollments.course_id );
```

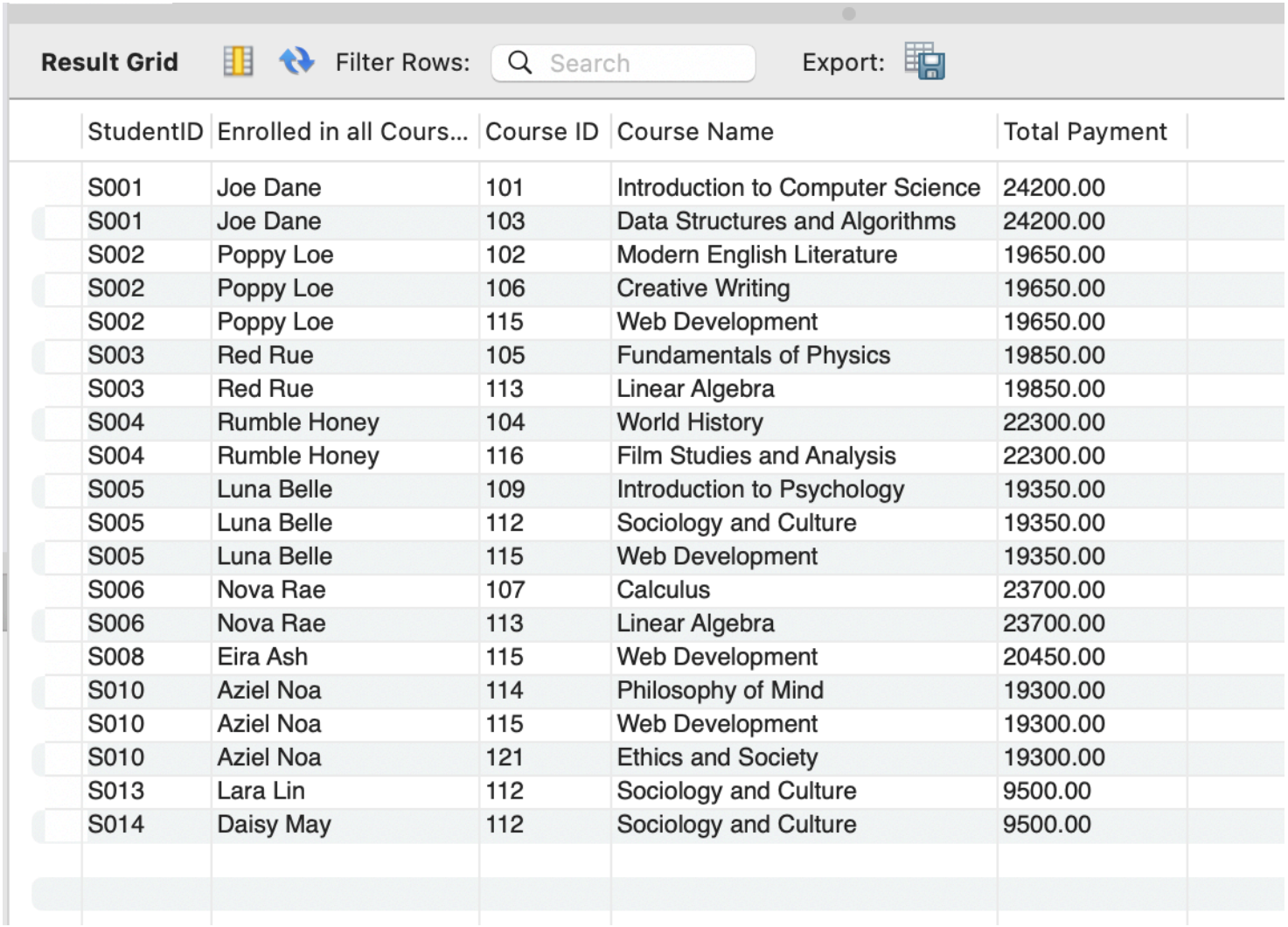


Course ID	Course Name
108	Principles of Economics
111	Art and Design Basics
117	Music Theory
118	Theatre and Performance
119	Modern Political Thought

9. Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.

CODE :

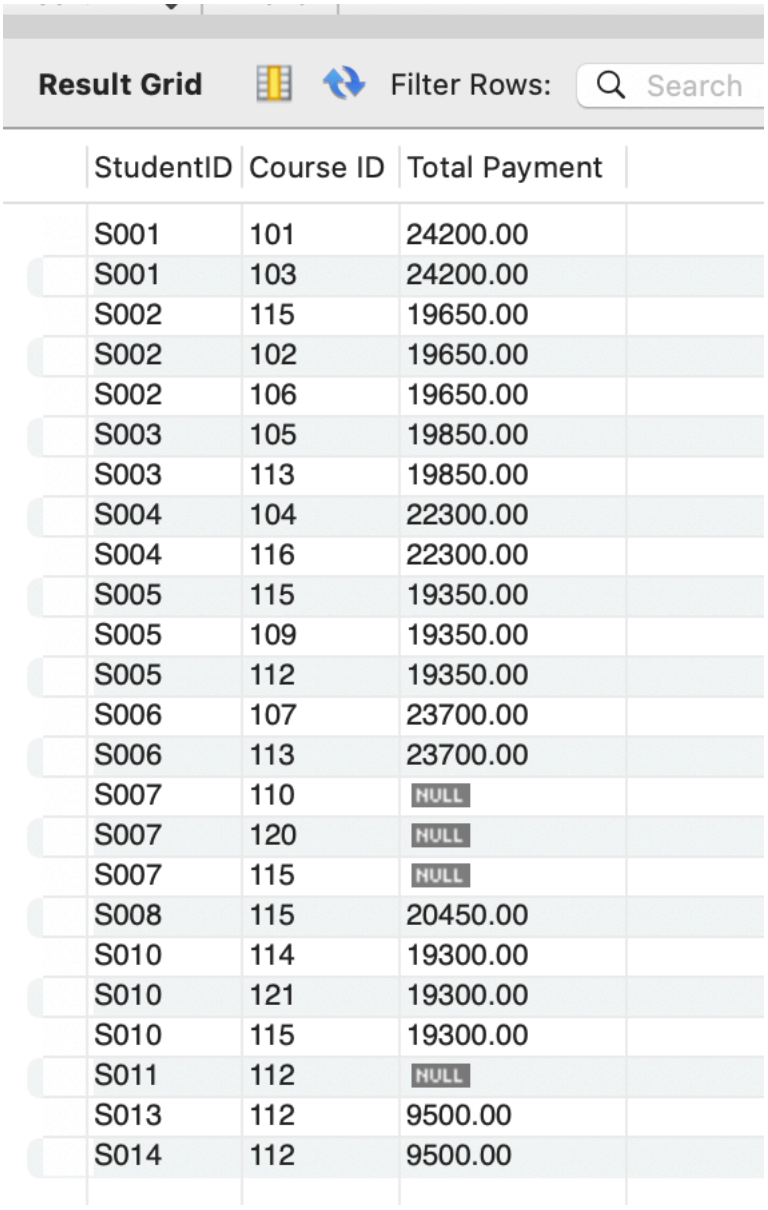
```
Select s.student_id StudentID,
          concat(s.first_name," ",s.last_name) 'Enrolled in all Courses',
       c.course_id 'Course ID',
       c.course_name 'Course Name',
       sum(p.amount) 'Total Payment'
from Students s
join Enrollments e
        on s.student_id =  e.student_id
join Courses c
        on e.course_id = c.course_id
join Payments p
        on s.student_id =  p.student_id
group by s.student_id, c.course_id
order by s.student_id;
```



Result Grid Filter Rows: Search Export:

StudentID	Enrolled in all Cours...	Course ID	Course Name	Total Payment
S001	Joe Dane	101	Introduction to Computer Science	24200.00
S001	Joe Dane	103	Data Structures and Algorithms	24200.00
S002	Poppy Loe	102	Modern English Literature	19650.00
S002	Poppy Loe	106	Creative Writing	19650.00
S002	Poppy Loe	115	Web Development	19650.00
S003	Red Rue	105	Fundamentals of Physics	19850.00
S003	Red Rue	113	Linear Algebra	19850.00
S004	Rumble Honey	104	World History	22300.00
S004	Rumble Honey	116	Film Studies and Analysis	22300.00
S005	Luna Belle	109	Introduction to Psychology	19350.00
S005	Luna Belle	112	Sociology and Culture	19350.00
S005	Luna Belle	115	Web Development	19350.00
S006	Nova Rae	107	Calculus	23700.00
S006	Nova Rae	113	Linear Algebra	23700.00
S008	Eira Ash	115	Web Development	20450.00
S010	Aziel Noa	114	Philosophy of Mind	19300.00
S010	Aziel Noa	115	Web Development	19300.00
S010	Aziel Noa	121	Ethics and Society	19300.00
S013	Lara Lin	112	Sociology and Culture	9500.00
S014	Daisy May	112	Sociology and Culture	9500.00

```
Select student_id StudentID,
       course_id 'Course ID',
       (select sum(amount)
        from Payments p
        where p.student_id = e.student_id) 'Total Payment'
from Enrollments e
order by student_id;
```



Result Grid Filter Rows: Search

StudentID	Course ID	Total Payment
S001	101	24200.00
S001	103	24200.00
S002	115	19650.00
S002	102	19650.00
S002	106	19650.00
S003	105	19850.00
S003	113	19850.00
S004	104	22300.00
S004	116	22300.00
S005	115	19350.00
S005	109	19350.00
S005	112	19350.00
S006	107	23700.00
S006	113	23700.00
S007	110	NULL
S007	120	NULL
S007	115	NULL
S008	115	20450.00
S010	114	19300.00
S010	121	19300.00
S010	115	19300.00
S011	112	NULL
S013	112	9500.00
S014	112	9500.00

10. Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one.

Enrolled some students to have total payment = 1 in count, to show variety in total amount payment by each student.

```
668 •   insert into Enrollments values (null, 'S013', 112, '2026-01-08'),
669                                    (null, 'S014', 112, '2026-01-08');
670
```

100% | 27:663 | 1 error found

Result Grid | Filter Rows: Search | Edit: | Export/Import:

enrollment_id	student_id	course_id	enrollment_da...
1	S004	116	2022-08-01
2	S010	121	2022-08-01
3	S001	101	2023-01-15
4	S007	120	2023-08-01
5	S002	102	2023-08-02
7	S006	113	2024-02-03
8	S003	105	2024-08-01
9	S008	115	2024-08-01
10	S005	112	2024-08-03
11	S004	104	2025-01-12
12	S010	114	2025-01-12
13	S001	103	2025-08-01
14	S007	110	2025-08-02
15	S002	106	2025-08-02
17	S006	107	2025-08-05
18	S003	113	2025-08-05
20	S005	109	2025-08-06
21	S011	112	2025-08-07
22	S005	115	2025-08-15
23	S002	115	2025-08-15
24	S010	115	2025-08-16
25	S007	115	2025-08-17
26	S013	112	2026-01-08
27	S014	112	2026-01-08
NULL	NULL	NULL	NULL

```
671 •   insert into Payments values (null, 'S013', 9500, '2026-01-08'),
672                                  (null, 'S014', 9500, '2026-01-19');
```

100% | 24:664 | 1 error found

Result Grid | Filter Rows: Search | Edit: | Export/Import:

payment_id	student_...	amount	payment_date
1001	S004	11000.00	2022-08-05
1002	S010	9500.00	2022-08-10
1003	S001	12000.00	2023-01-20
1005	S002	9800.00	2023-08-08
1007	S006	11800.00	2024-02-05
1008	S003	9900.00	2024-08-10
1009	S008	10150.00	2024-08-12
1010	S005	9700.00	2024-08-15
1011	S004	11300.00	2025-01-20
1012	S010	9800.00	2025-01-25
1013	S001	12200.00	2025-08-01
1015	S002	9850.00	2025-08-04
1017	S006	11900.00	2025-08-07
1018	S003	9950.00	2025-08-08
1019	S008	10300.00	2025-08-09
1020	S005	9650.00	2025-08-10
1021	S013	9500.00	2026-01-08
1022	S014	9500.00	2026-01-19
NULL	NULL	NULL	NULL

CODE :

Correlated Subquery

```
Select s.student_id StudentID,
      concat(s.first_name," ",s.last_name) 'Student Name'
from Students s
where exists (select p.student_id, count(amount) Payment_Count
from Payments p
where s.student_id = p.student_id
group by student_id
having Payment_Count > 1) ;
```

Result Grid Filter Rows:

StudentID	Student Name
S001	Joe Dane
S002	Poppy Loe
S003	Red Rue
S004	Rumble Honey
S005	Luna Belle
S006	Nova Rae
S008	Eira Ash
S010	Aziel Noa

Using join

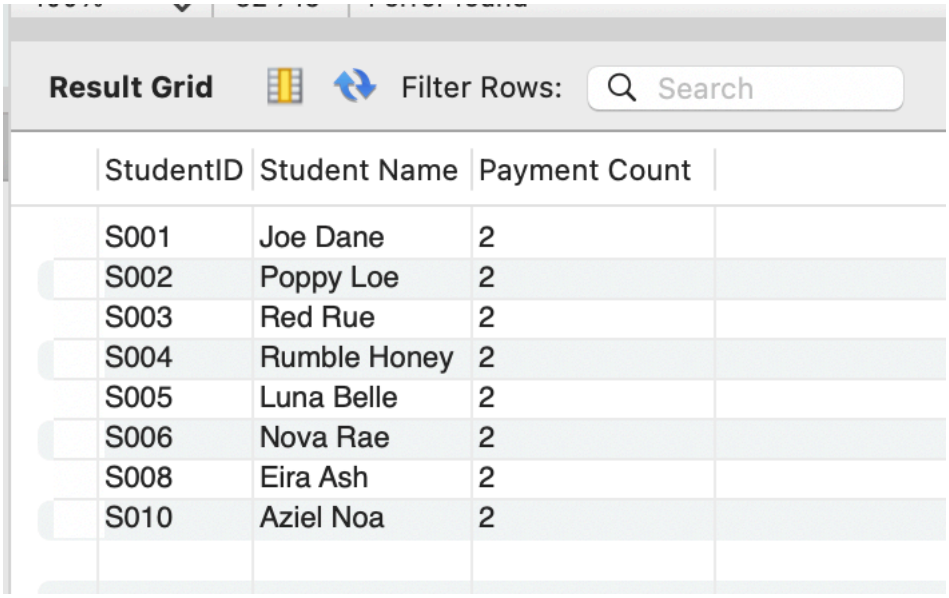
```
Select s.student_id StudentID,
    concat(s.first_name," ",s.last_name) 'Student Name',
    count(p.amount) 'Payment Count'
from Students s
join Payments p
on s.student_id = p.student_id
group by s.student_id
having count(p.amount) > 1
order by s.student_id;
```

Result Grid Filter Rows: Search

StudentID	Student Name	Payment Count
S001	Joe Dane	2
S002	Poppy Loe	2
S003	Red Rue	2
S004	Rumble Honey	2
S005	Luna Belle	2
S006	Nova Rae	2
S008	Eira Ash	2
S010	Aziel Noa	2

Using join and subquery

```
Select s.student_id StudentID,
       concat(s.first_name," ",s.last_name) 'Student Name',
       Student_Payment_Count.Payment_Count 'Payment Count'
from Students s
join (select student_id, count(amount) Payment_Count
          from Payments
          group by student_id
      having Payment_Count> 1) Student_Payment_Count
on     s.student_id = Student_Payment_Count.student_id;
```



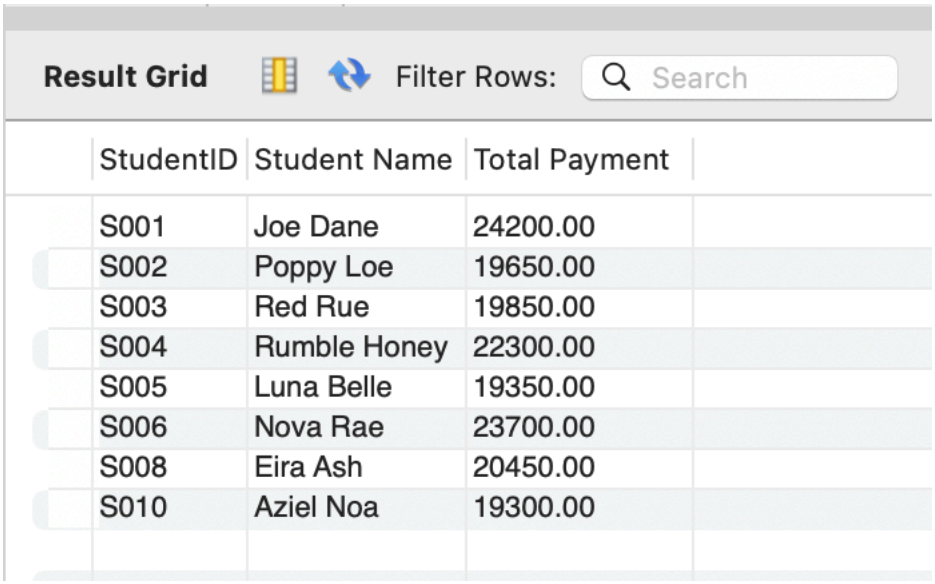
StudentID	Student Name	Payment Count
S001	Joe Dane	2
S002	Poppy Loe	2
S003	Red Rue	2
S004	Rumble Honey	2
S005	Luna Belle	2
S006	Nova Rae	2
S008	Eira Ash	2
S010	Aziel Noa	2

11. Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.

CODE :

```
Select s.student_id StudentID,
       concat(s.first_name," ",s.last_name) 'Student Name',
       sum(p.amount) 'Total Payment'
from Students s
```

```
join Payments p
on s.student_id = p.student_id
group by s.student_id
order by s.student_id;
```



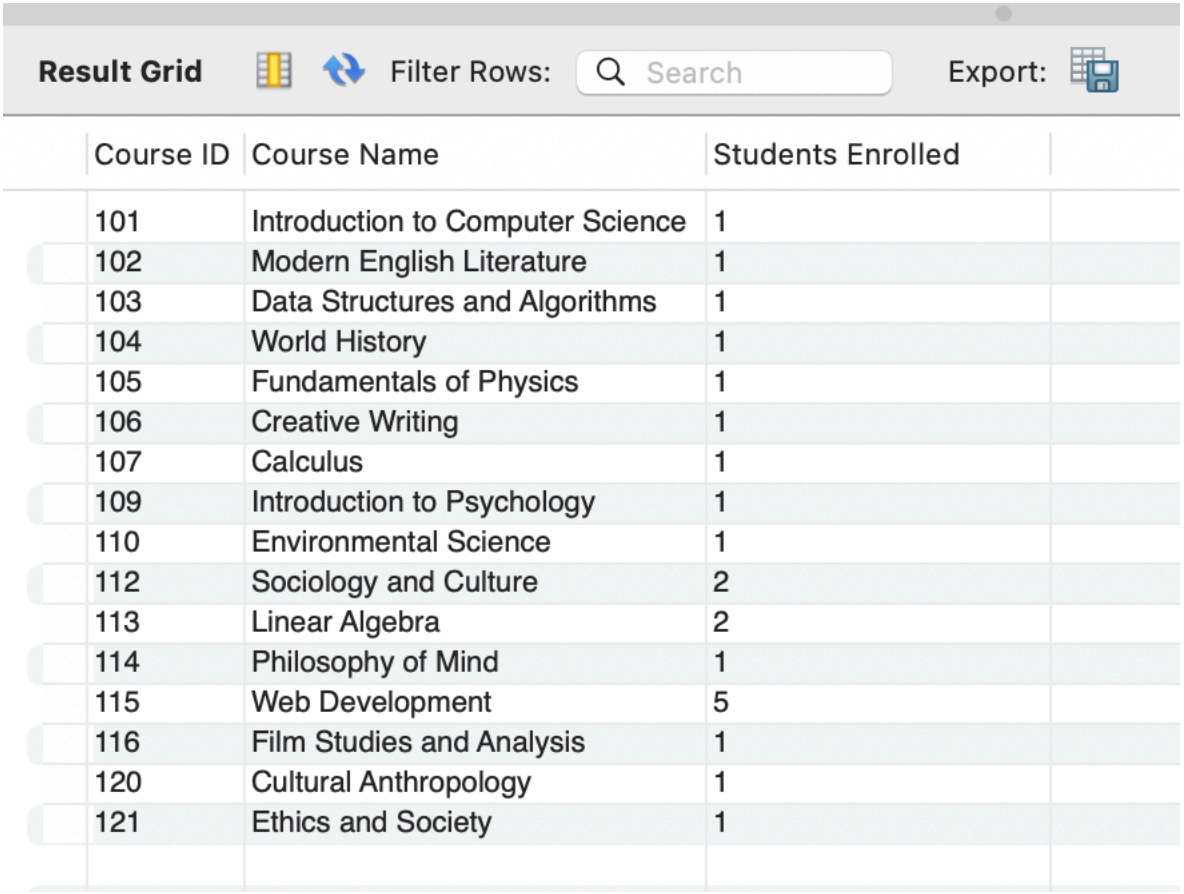
Result Grid Filter Rows: Search

StudentID	Student Name	Total Payment
S001	Joe Dane	24200.00
S002	Poppy Loe	19650.00
S003	Red Rue	19850.00
S004	Rumble Honey	22300.00
S005	Luna Belle	19350.00
S006	Nova Rae	23700.00
S008	Eira Ash	20450.00
S010	Aziel Noa	19300.00

12. Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.

CODE :

```
Select c.course_id 'Course ID',
	c.course_name 'Course Name',
	count(e.student_id) 'Students Enrolled'
from Courses c
join Enrollments e
on c.course_id = e.course_id
group by c.course_id, c.course_name
order by c.course_id;
```



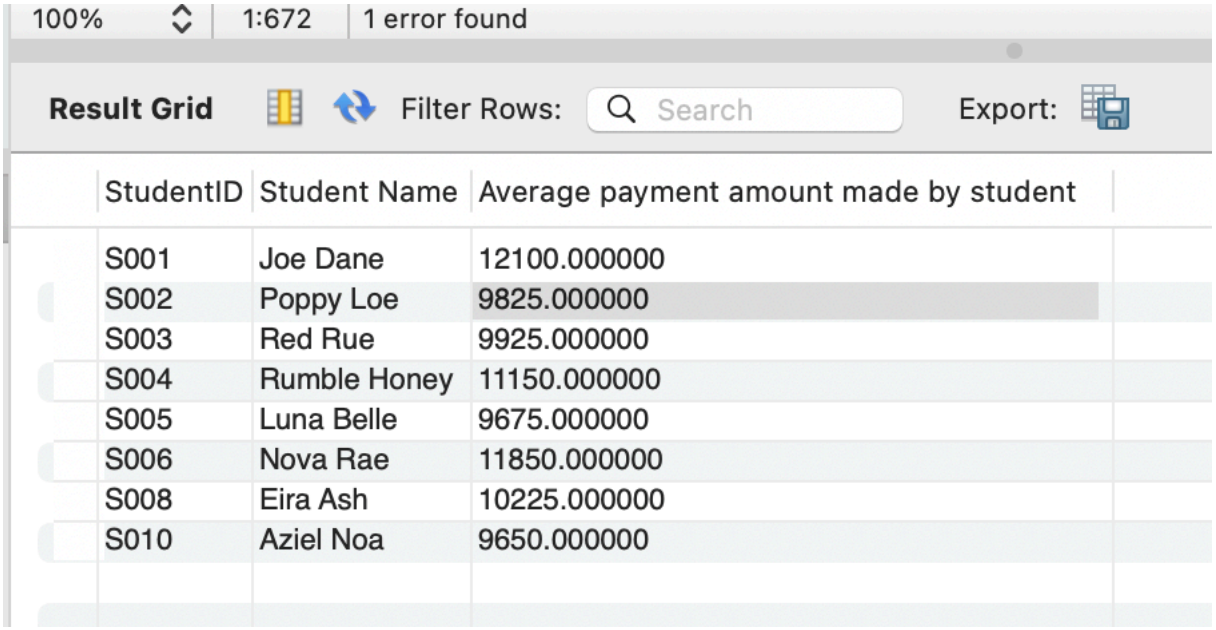
Result Grid Filter Rows: Search Export:

Course ID	Course Name	Students Enrolled
101	Introduction to Computer Science	1
102	Modern English Literature	1
103	Data Structures and Algorithms	1
104	World History	1
105	Fundamentals of Physics	1
106	Creative Writing	1
107	Calculus	1
109	Introduction to Psychology	1
110	Environmental Science	1
112	Sociology and Culture	2
113	Linear Algebra	2
114	Philosophy of Mind	1
115	Web Development	5
116	Film Studies and Analysis	1
120	Cultural Anthropology	1
121	Ethics and Society	1

13. Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.

CODE :

```
Select s.student_id StudentID,
    concat(s.first_name," ",s.last_name) 'Student Name',
    avg(p.amount) 'Average payment amount made by student'
from Students s
join Payments p
on s.student_id = p.student_id
group by s.student_id
order by s.student_id;
```



StudentID	Student Name	Average payment amount made by student
S001	Joe Dane	12100.000000
S002	Poppy Loe	9825.000000
S003	Red Rue	9925.000000
S004	Rumble Honey	11150.000000
S005	Luna Belle	9675.000000
S006	Nova Rae	11850.000000
S008	Eira Ash	10225.000000
S010	Aziel Noa	9650.000000