

DB Assignment

1. Create the tables below in the database. Use foreign keys and primary keys as required.
 - a. Create a table called as student with the following columns student_id, first_name, last_name ,birthdate , department_id ,address_id .

```
create table student(  
    student_id int primary key,  
    first_name varchar(100),  
    last_name varchar(100),  
    birthdate date,  
    department_id int,  
    address_id int,  
    foreign key(department_id) references department(department_id),  
    foreign key(address_id) references address(address_id));
```

- b. Create Address table with following columns address_id , street_address, city, State, postal_code

```
create table Address(  
    address_id int primary key,  
    street_address varchar(255),  
    city varchar(100),  
    state varchar(100),  
    postal_code varchar(20));
```

- c. Create department table department_id, department name. Make sure you are using the right data type against all the columns.

```
create table Department(  
    department_id int primary key,  
    department_name varchar(100));
```

2. Use Sample data from [sampledata.txt](#) to insert data into the database

```
load data infile 'C:/Java/TrainingAtBounteous/28-05-2025_Day5/sampledata.txt'  
into table student  
fields terminated by 'n' lines terminated by '\n';
```

3. Write a query to find the total number of students.

```
47 • select count(*) from student;
48
```

Result Grid | Filter Rows: | Export:

	count(*)
▶	5

4. Write a query to find which department john belongs to.

```
49 • SELECT s.first_name, s.last_name, d.department_name
50 FROM student s
51 JOIN department d ON s.department_id = d.department_id
52 WHERE s.first_name = 'John';
53
```

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

	first_name	last_name	department_name
▶	John	Doe	Computer Science

5. List All Departments with Their Number of Students (Including Departments with No Students)

```
55 • SELECT d.department_name, COUNT(s.student_id) AS student_count
56 FROM department d
57 LEFT JOIN student s ON d.department_id = s.department_id
58 GROUP BY d.department_name;
59
```

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

	department_name	student_count
▶	Computer Science	1
	Mathematics	1
	Physics	1
	Biology	1
	Engineering	1

6. Select all students with their department and address.

```
62 • SELECT s.first_name,s.last_name, d.department_name, a.street_address, a.city, a.state, a.postal_code
63 FROM student s
64 JOIN department d ON s.department_id = d.department_id
65 JOIN address a ON s.address_id = a.address_id;
```

Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content:							
	first_name	last_name	department_name	street_address	city	state	postal_code
▶	John	Doe	Computer Science	123 Maple St	Springfield	IL	62704
	Jane	Smith	Mathematics	456 Oak Ave	Riverside	CA	92501
	Robert	Brown	Physics	789 Pine Rd	Hilltown	TX	75001
	Linda	Johnson	Biology	321 Cedar Blvd	Lakeview	NY	10001
	Michael	Williams	Engineering	654 Birch Ln	Meadowbrook	FL	33101

7. Find all students who are in the 'Computer Science' department

```
68 • SELECT s.*
69 FROM student s
70 JOIN department d ON s.department_id = d.department_id
71 WHERE d.department_name = 'Computer Science';
```

Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content:						
	student_id	first_name	last_name	birthdate	department_id	address_id
▶	1	John	Doe	2000-05-15	101	1001

8. Update Jane's city name to New York.

```
update address set city='New York'
where address_id=(select address_id from student where first_name='Jane');
```

9. Delete a student from the student table.

```
DELETE FROM student
WHERE student_id = 5;
```

10. Select all students with their department and address in New York.

```
78 • SELECT s.*, d.department_name, a.city
79 FROM student s
80 JOIN department d ON s.department_id = d.department_id
81 JOIN address a ON s.address_id = a.address_id
82 WHERE a.city = 'New York';
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	student_id	first_name	last_name	birthdate	department_id	address_id	department_name	city
▶	2	Jane	Smith	1999-10-20	102	1002	Mathematics	New York

11. Count how many students are in each department

```
84 • select d.department_name as Department, count(*) as NumberOfStudents
85 from department d left join student s on
86 s.department_id=d.department_id group by d.department_name;
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Department	NumberOfStudents
▶	Computer Science	1
	Mathematics	1
	Physics	1
	Biology	1
	Engineering	1

12. Find students who live in 'Springfield'

```

89 • SELECT s.*
90 FROM student s
91 JOIN address a ON s.address_id = a.address_id
92 WHERE a.city = 'Springfield';
93

```

Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content:						
	student_id	first_name	last_name	birthdate	department_id	address_id
▶	1	John	Doe	2000-05-15	101	1001

13. Select students whose birthday falls in February

```

94 • SELECT *
95 FROM student
96 WHERE MONTH(birthdate) = 2;
97

```

Result Grid Filter Rows: <input type="text"/> Edit: Export/Import:						
	student_id	first_name	last_name	birthdate	department_id	address_id
*	NULL	NULL	NULL	NULL	NULL	NULL

14. Get the department and address details for a specific student, example john

```

100 • SELECT s.first_name, s.last_name, d.department_name, a.*
101 FROM student s
102 JOIN department d ON s.department_id = d.department_id
103 JOIN address a ON s.address_id = a.address_id
104 WHERE s.first_name = 'John';
105

```

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

	first_name	last_name	department_name	address_id	street_address	city	state	postal_code
•	John	Doe	Computer Science	1001	123 Maple St	Springfield	IL	62704

15. Find all students who are born within 1995 to 1998

```

107 • SELECT *
108 FROM student
109 WHERE birthdate BETWEEN '1995-01-01' AND '1998-12-31';
110

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

	student_id	first_name	last_name	birthdate	department_id	address_id
*	NULL	NULL	NULL	NULL	NULL	NULL

16. List all students and their corresponding department names, sorted by department

```

111 • select s.first_name,s.last_name,d.department_name
112 from student s inner join department d on s.department_id=d.department_id
113 order by d.department_name;
114

```

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

	first_name	last_name	department_name
▶	Linda	Johnson	Biology
	John	Doe	Computer Science
	Jane	Smith	Mathematics
	Robert	Brown	Physics

17. Find the number of students in each department who are living in 'Champaign'

```
115 • select d.department_name as Department, count(*) as NumberOfStudentsInChampaign
116 from department d join student s on
117 s.department_id=d.department_id join address a where a.city='Champaign' group by d.department_name;
118
```

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

Department	NumberOfStudentsInChampaign
------------	-----------------------------

18. Retrieve the names of students who live on 'Pine' street

```
121 • SELECT s.first_name, s.last_name
122 FROM student s
123 JOIN address a ON s.address_id = a.address_id
124 WHERE a.street_address = 'Pine';
125
```

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





first_name	last_name
------------	-----------

19. Update the department of a student with student_id = 6 to 'Mechanical Engineering'

```
update student set department_id=(select department_id from department
where department_name='Mechanical Engineering') where student_id=6;
```

20. Find the student(s) who live in the city 'Chicago' and are in the 'Mathematics' department

```
SELECT s.*  
FROM student s  
JOIN department d ON s.department_id = d.department_id  
JOIN address a ON s.address_id = a.address_id  
WHERE d.department_name = 'Mathematics' AND a.city = 'Chicago';
```

rid			Filter Rows: <input type="text"/>	Export: 	Wrap Cell Content: 
ent_id	first_name	last_name	birthdate	department_id	address_id

21. List all students who have an address in 'Urbana' or 'Peoria'

```
SELECT s.*  
FROM student s  
JOIN address a ON s.address_id = a.address_id  
WHERE a.city IN ('Urbana', 'Peoria');
```

22. Find the student with the highest student_id


```

142 • SELECT *
143 FROM student
144 ORDER BY student_id DESC
145 LIMIT 1;

```

Result Grid Filter Rows: <input type="text"/> Edit: Export/Import:						
	student_id	first_name	last_name	birthdate	department_id	address_id
▶	4	Linda	Johnson	2000-12-11	104	1004
✱	NULL	NULL	NULL	NULL	NULL	NULL

23. Find all students who are not in the 'Computer Science' department

```

SELECT s.*
FROM student s
JOIN department d ON s.department_id = d.department_id
WHERE d.department_name <> 'Computer Science';

```

Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content:					
student_id	first_name	last_name	birthdate	department_id	address_id
	Jane	Smith	1999-10-20	102	1002
	Robert	Brown	2001-07-08	103	1003
	Linda	Johnson	2000-12-11	104	1004

24. Count the total number of addresses in the 'Champaign' city

```

SELECT COUNT(*) AS address_count
FROM address
WHERE city = 'Champaign';

```

25. Find the name of the student who lives at '520 Pine Rd'

```

SELECT s.first_name, s.last_name
FROM student s
JOIN address a ON s.address_id = a.address_id
WHERE a.street_address = '520 Pine Rd';

```

26. Get the average age of students in the 'Electrical Engineering' department

```

SELECT AVG(YEAR(CURDATE()) - YEAR(birthdate)) AS average_age
FROM student s
JOIN department d ON s.department_id = d.department_id
WHERE d.department_name = 'Electrical Engineering';



```

27. List the students, their department, and the city where they live, but only for those in departments starting with 'M'

```

167 • SELECT s.first_name, s.last_name, d.department_name, a.city
168     FROM student s
169     JOIN department d ON s.department_id = d.department_id
170     JOIN address a ON s.address_id = a.address_id
171     WHERE d.department_name LIKE 'M%';
172

```

Result Grid				
Filter Rows: <input type="text"/>				
Export: 				
Wrap Cell Content: 				
	first_name	last_name	department_name	city
▶	Jane	Smith	Mathematics	New York

28. Delete a student from the 'Mechanical Engineering' department




```
DELETE FROM student
WHERE department_id = (
    SELECT department_id
    FROM department
    WHERE department_name = 'Mechanical Engineering'
    LIMIT 1)
LIMIT 1;
```

Click on “Open file” and select [order.sql](#) from your device and execute it.

1. Retrieve All Orders with Their Customer Details and Current Status

<div> <div>Result Grid</div> <div> <div>Filter Rows:</div> <div>Export:</div> <div>Wrap Cell Content:</div> </div> </div>								
	order_id	order_date	customer_id	first_name	last_name	email	phone_number	status_name
4	2025-02-18	1	John	Doe	john.doe@example.com	555-1234	Cancelled	
3	2025-02-17	3	Emily	Jones	emily.jones@example.com	555-8765	Shipped	
2	2025-02-16	2	Jane	Smith	jane.smith@example.com	555-5678	Pending	
1	2025-02-15	1	John	Doe	john.doe@example.com	555-1234	Shipped	

```
SELECT o.customer_id, SUM(oi.quantity * oi.price) AS total_order_value
FROM order_schema.orders o
JOIN order_schema.order_items oi ON o.order_id = oi.order_id
WHERE o.customer_id = 4
      AND o.order_date BETWEEN 2022-05-12 AND 2025-03-01
GROUP BY o.customer_id;
```

id |  Filter Rows: | Export:  | Wrap Cell Content: 

3. Find the Most Expensive Order by Customer

```
315 • SELECT o.customer_id, o.order_id, SUM(oi.quantity * oi.price) AS order_total
316 FROM order_schema.orders o
317 JOIN order_schema.order_items oi ON o.order_id = oi.order_id
318 GROUP BY o.customer_id, o.order_id
319 ORDER BY order_total DESC
320 LIMIT 1;
321
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
customer_id	order_id	order_total		
1	1	1499.98		

4. Find the Total Revenue for Each Product Based on Orders

```
324 • SELECT p.product_id, p.product_name, COALESCE(SUM(oi.quantity * oi.price), 0) AS total_revenue
325 FROM order_schema.product p
326 LEFT JOIN order_schema.order_items oi ON p.product_id = oi.product_id
327 GROUP BY p.product_id, p.product_name
328 ORDER BY total_revenue DESC;
329
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:	
product_id	product_name	total_revenue		
1	Laptop	999.99		
2	Smartphone	999.98		
3	Headphones	449.97		
4	Monitor	0.00		

5. Write a query to retrieve the order ID, customer ID, and the total amount of each order. If the total amount is null, display '0.00' instead.

```

SELECT o.order_id, o.customer_id,
       COALESCE(SUM(oi.quantity * oi.price), 0.00) AS total_amount
FROM order_schema.orders o
LEFT JOIN order_schema.order_items oi ON o.order_id = oi.order_id
GROUP BY o.order_id, o.customer_id;

```

Grid	Filter Rows:	Export:	Wrap Cell Content:
order_id	customer_id	total_amount	
1	1	1499.98	
1	1	149.99	
2	2	299.98	
3	3	499.99	

6. Retrieve the Order History of a Specific Customer Along with Product Details

```

SELECT o.order_id, o.order_date, s.status_name,
       p.product_id, p.product_name, oi.quantity, oi.price
FROM order_schema.orders o
JOIN order_schema.order_items oi ON o.order_id = oi.order_id
JOIN order_schema.product p ON oi.product_id = p.product_id
JOIN order_schema.status s ON o.status_id = s.status_id
WHERE o.customer_id = 1
ORDER BY o.order_date DESC;

```

Grid

Filter Rows:

Export:

Wrap Cell Content:

order_id	order_date	status_name	product_id	product_name	quantity	price
2025-02-18	Cancelled	3	Headphones	1	149.99	
2025-02-15	Shipped	1	Laptop	1	999.99	
2025-02-15	Shipped	2	Smartphone	1	499.99	

7. Get the Average Order Value Per Customer in the Last 30 Days.

```

SELECT o.customer_id, AVG(order_total) AS avg_order_value
FROM (
    SELECT o.order_id, o.customer_id, SUM(oi.quantity * oi.price) AS order_total
    FROM order_schema.orders o
    JOIN order_schema.order_items oi ON o.order_id = oi.order_id
    WHERE o.order_date >= CURRENT_DATE - INTERVAL '30 days'
    GROUP BY o.order_id, o.customer_id
) AS order_totals
GROUP BY o.customer_id;

```

8. Get the Top 5 Products with the Highest Number of Orders.

```

SELECT p.product_id, p.product_name, COUNT(DISTINCT oi.order_id) AS order_count
FROM order_schema.product p
JOIN order_schema.order_items oi ON p.product_id = oi.product_id
GROUP BY p.product_id, p.product_name
ORDER BY order_count DESC
LIMIT 5;

```

product_id	product_name	order_count
2	Smartphone	2
3	Headphones	2
1	Laptop	1

9. Get the Customers Who Have Not Placed Any Orders in the Last 60 Days

```

SELECT c.customer_id, c.first_name, c.last_name, c.email
FROM order_schema.customer c
LEFT JOIN order_schema.orders o ON c.customer_id = o.customer_id
AND o.order_date >= CURRENT_DATE - INTERVAL '60 days'
WHERE o.order_id IS NULL;


```

10. List the Orders with Products Ordered More Than Once, Sorted by Order Date


```

378 • SELECT o.order_id, o.order_date, p.product_name, oi.quantity
379 FROM order_schema.orders o
380 JOIN order_schema.order_items oi ON o.order_id = oi.order_id
381 JOIN order_schema.product p ON oi.product_id = p.product_id
382 WHERE oi.quantity > 1
383 ORDER BY o.order_date DESC;
384
385

```

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



	order_id	order_date	product_name	quantity
▶	2	2025-02-16	Headphones	2

11. Retrieve the Number of Orders and Total Revenue for Each Status

```

387 • SELECT s.status_name,
388         COUNT(DISTINCT o.order_id) AS order_count,
389         COALESCE(SUM(oi.quantity * oi.price), 0) AS total_revenue
390 FROM order_schema.status s
391 LEFT JOIN order_schema.orders o ON s.status_id = o.status_id
392 LEFT JOIN order_schema.order_items oi ON o.order_id = oi.order_id
393 GROUP BY s.status_name;
394
395

```

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

	status_name	order_count	total_revenue
▶	Cancelled	1	149.99
	Pending	1	299.98
	Shipped	2	1999.97

12. Customers Who Have Ordered More Than a Specific Product (e.g., "Laptop")


```

SELECT c.customer_id, c.first_name, c.last_name, SUM(oi.quantity) AS total_quantity
FROM order_schema.customer c
JOIN order_schema.orders o ON c.customer_id = o.customer_id
JOIN order_schema.order_items oi ON o.order_id = oi.order_id
JOIN order_schema.product p ON oi.product_id = p.product_id
WHERE p.product_name = 'Laptop'
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING SUM(oi.quantity) > 1;

```

Grid			Filter Rows:		Export:		Wrap Cell Content:	
customer_id	first_name	last_name	total_quantity					

13. Find the Products That Have Never Been Ordered

```

409 • SELECT p.product_id, p.product_name
410 FROM order_schema.product p
411 LEFT JOIN order_schema.order_items oi ON p.product_id = oi.product_id
412 WHERE oi.product_id IS NULL;
413
414

```

Result Grid			Filter Rows:		Export:		Wrap Cell Content:	
	product_id	product_name						
▶	4	Monitor						

14. Get the Total Quantity of Products Ordered in the Last 7 Days

```

SELECT COALESCE(SUM(oi.quantity), 0) AS total_quantity_ordered
FROM order_schema.order_items oi
JOIN order_schema.orders o ON oi.order_id = o.order_id
WHERE o.order_date >= CURRENT_DATE - INTERVAL '7 days';

```

15. Create a view named product_details that includes all columns from the product table.

```
CREATE VIEW order_schema.product_details AS
SELECT *
FROM order_schema.product;
```

16. Create a view named order_summary that includes the order_id, customer_id, order_date, total_amount, and status_name (from the status table) for each order.

```
CREATE VIEW order_schema.order_summary2 AS
SELECT o.order_id, o.customer_id, o.order_date,
       COALESCE(SUM(oi.quantity * oi.price), 0) AS total_amount,
       s.status_name
FROM order_schema.orders o
LEFT JOIN order_schema.order_items oi ON o.order_id = oi.order_id
JOIN order_schema.status s ON o.status_id = s.status_id
GROUP BY o.order_id, o.customer_id, o.order_date, s.status_name;
```