

JSPM UNIVERSITY PUNE

Faculty of Science and Technology

School of Computational Sciences



Lab Practical File

FY Master of Computer Application

**Academic year -2024-25
Semester-II**

**Course Name: Advance DataBase Management
System (ADBMS) Lab**

Course Code: 230GCAM19_02

Submit By:

Submitted To : Dr. Rachana Chavan



JSPM UNIVERSITY PUNE

Recognized by the UGC u/s 2 (f) of UGC Act 1956 and enacted by the
State Government of Maharashtra - JSPM University Act, 2022 (Mah. IV of 2023)

Date:18/4/2025

Certificate

This is to certify that the work entered in this journal is the work of Miss.

_____ of **FY Master of Computer**

Application (MCA) Division: **B**, Roll No.____, and PRN:-_____.

The student has satisfactorily completed the required number of practicals for the
Course **Advance DataBase Management System** of **Semester - II** for the
academic year 2024-2025 in the laboratory, as prescribed by the University.

External Examiner

Dr. Rachana Chavan

Course Coordinator

Dr. Khan Arshiya A.

Program Coordinator

Dr. R.S.Deshpande

Dean,FST



JSPM UNIVERSITY PUNE

Recognized by the UGC u/s 2 (f) of UGC Act 1956 and enacted by the
State Government of Maharashtra - JSPM University Act, 2022 (Mah. IV of 2023)

INDEX

Sr. No.	Experiment List	Date of Execution	Remark	Signature
1.	Creation of database and SQL Queries to retrieve information from database			
2.	Performing insertion, deletion, modification, altering and updating operations on the records based on conditions.			
3.	To Implement the Aggregate functions concept on the table			
4.	To Implement the structure of the table.			
5.	To implement the concept of Joins.			
6.	To implement the concept of grouping of Data.			
7.	To implement the concept of Sub-Queries.			
8.	To implement the concept of Indexes and views.			
9.	To apply the concept of security and privileges.			
10.	Create a database of Hotel Management System.			
11.	Create a database of university.			
12.	Create a database of E-Commerce business.			

Creation of database and SQL Queries to retrieve information from database

Create database CompanyDetails;

use CompanyDetails;

```
CREATE TABLE Salesman (  
    salesman_id INT PRIMARY KEY,  
    name VARCHAR(50),  
    city VARCHAR(50),  
    commission DECIMAL(4, 2)  
);
```

```
INSERT INTO Salesman (salesman_id, name, city, commission) VALUES  
(5001, 'James Hoog', 'New York', 0.15),  
(5002, 'Nail Knite', 'Paris', 0.13),  
(5005, 'Pit Alex', 'London', 0.11),  
(5006, 'Mc Lyon', 'Paris', 0.14),  
(5003, 'Lauson Hen', 'San Jose', 0.12),  
(5007, 'Paul Adam', 'Rome', 0.13);
```

```
CREATE TABLE Orders (  
    ord_no INT PRIMARY KEY,  
    purch_amt DECIMAL(10, 2),  
    ord_date DATE,  
    customer_id INT,  
    salesman_id INT  
);
```

```
INSERT INTO Orders (ord_no, purch_amt, ord_date, customer_id, salesman_id) VALUES  
(70001, 150.5, '2012-10-05', 3005, 5002),  
(70009, 270.65, '2012-09-10', 3001, 5005),  
(70002, 65.26, '2012-10-05', 3002, 5001),  
(70004, 110.5, '2012-08-17', 3009, 5003),  
(70007, 948.5, '2012-09-10', 3005, 5002),  
(70005, 2400.6, '2012-07-27', 3007, 5001),  
(70008, 5760, '2012-09-10', 3002, 5001),  
(70010, 1983.43, '2012-10-10', 3004, 5006),  
(70003, 2480.4, '2012-10-10', 3009, 5003),  
(70012, 250.45, '2012-06-27', 3008, 5002),  
(70011, 75.29, '2012-08-17', 3003, 5007),  
(70013, 3045.6, '2012-04-25', 3002, 5001);
```

```
CREATE TABLE Customer (  

```

```
customer_id INT PRIMARY KEY,  
cust_name VARCHAR(50),  
city VARCHAR(50),  
grade INT,  
salesman_id INT  
);
```

```
INSERT INTO Customer (customer_id, cust_name, city, grade, salesman_id) VALUES  
(3002, 'Nick Rimando', 'New York', 100, 5001),  
(3005, 'Graham Zusi', 'California', 200, 5002),  
(3001, 'Brad Guzan', 'London', 100, 5005),  
(3004, 'Fabian Johns', 'Paris', 300, 5006),  
(3007, 'Brad Davis', 'New York', 200, 5001),  
(3009, 'Geoff Camero', 'Berlin', 100, 5003),  
(3008, 'Julian Green', 'London', 300, 5002),  
(3003, 'Jozy Altidor', 'Moncow', 200, 5007);
```

Performing insertion, deletion, modification, altering and updating operations on the records based on conditions..

Select all data from the Salesman table.

Select * from salesman;

#	s_id	name	s_city	commisior
1	5001	James Hogg	New York	0.15
2	5002	Nail Knitr	Paris	0.13
3	5005	Pit Alex	London	0.11
4	5006	Mc Lyon	Paris	0.14
5	5003	Lauson Hen	San Jose	0.12
6	5007	Paul Adam	Rome	0.13

Select the names of salesmen in New York.

Select * from salesman where city = "New York" ;

#	s_id	name	s_city	commisior
1	5001	James Hogg	New York	0.15

Find all orders with a purchase amount greater than 500.

select * from orders where purch_amt > 500;

#	ord_no	purch_amt	ord_date	cust_id	salesman_id
1	70007	948.5	2012-08-17	3005	5002
2	70005	2400.6	2012-09-10	3007	5001

Select the name and commission of salesmen in Paris.

select name,commision from salesman where s_city="Paris";

#	name	commisior
1	Nail Knitr	0.13
2	Mc Lyon	0.14

Find customers with grades greater than or equal to 200.

select * from customer where grade >= 200;

#	cust_id	cust_name	city	grade	salesman_id
1	3005	Graham Zusi	California	200	5002
2	3004	Fabian Johns	Paris	300	5006
3	3007	Brad Davis	New York	200	5001
4	3008	Julian Green	London	300	5002
5	3003	Jozy Altidor	Moncow	200	5007

Select order numbers where the purchase amount is below 1000.

select * from customer where grade >= 200;

#	ord_no
1	70001
2	70009
3	70002
4	70004
5	70007
6	70011

Select all salesmen with commissions between 0.10 and 0.15.
select ord_no from orders where purch_amt < 1000;

#	s_id	name	s_city	commisior
1	5002	Nail Knitr	Paris	0.13
2	5005	Pit Alex	London	0.11
3	5006	Mc Lyon	Paris	0.14
4	5003	Lauson Hen	San Jose	0.12
5	5007	Paul Adam	Rome	0.13

List orders placed by customer ID 3002.
select * from orders where cust_id = 3002;

#	ord_no	purch_amt	ord_date	cust_id	salesman_ic
1	70002	65.26	2012-10-05	3002	5001

Select customers with IDs less than 3005.
select * from customer where cust_id < 3005;

#	cust_id	cust_name	city	grade	salesman_ic
1	3002	Nick Rimando	New York	100	5001
2	3001	Brad Guzan	London	100	5005
3	3004	Fabian Johns	Paris	300	5006
4	3003	Jozy Altidor	Moncow	200	5007

Select customers whose name starts with 'B'.
select * from customer where cust_name like "B%";

#	cust_id	cust_name	city	grade	salesman_ic
1	3001	Brad Guzan	London	100	5005
2	3007	Brad Davis	New York	200	5001

Delete all orders with purchase amounts below 100.
delete from orders where purch_amt < 100;

#	ord_no	purch_amt	ord_date	cust_id	salesman_ic
1	70001	150.5	2012-10-05	3005	5002
2	70009	270.65	2012-09-10	3001	5005
3	70004	110.5	2012-09-17	3009	5003
4	70007	948.5	2012-08-17	3005	5002
5	70005	2400.6	2012-09-10	3007	5001

Remove customers from the city 'Moncow'.
delete from customer where city ="Moncow";

#	cust_id	cust_name	city	grade	salesman_id
1	3002	Nick Rimando	New York	100	5001
2	3005	Graham Zusi	California	200	5002
3	3001	Brad Guzan	London	100	5005
4	3004	Fabian Johns	Paris	300	5006
5	3007	Brad Davis	New York	200	5001
6	3009	Geoff Camero	Berlin	100	5003
7	3008	Julian Green	London	300	5002

Delete orders placed before '2012-09-01'.
delete from orders where ord_date < "2012-09-01";

#	ord_no	purch_amt	ord_date	cust_id	salesman_id
1	70001	150.5	2012-10-05	3005	5002
2	70009	270.65	2012-09-10	3001	5005
3	70004	110.5	2012-09-17	3009	5003
4	70005	2400.6	2012-09-10	3007	5001

Delete salesmen with a commission less than 0.11.
delete from salesman where commision < 0.11;

#	s_id	name	s_city	commision
1	5001	James Hogg	New York	0.15
2	5002	Nail Knitr	Paris	0.13
3	5006	Mc Lyon	Paris	0.14
4	5003	Lauson Hen	San Jose	0.12
5	5007	Paul Adam	Rome	0.13

Remove customers with grades below 200.
delete from customer where grade < 200;

#	cust_id	cust_name	city	grade	salesman_id
1	3005	Graham Zusi	California	200	5002
2	3004	Fabian Johns	Paris	300	5006
3	3007	Brad Davis	New York	200	5001
4	3008	Julian Green	London	300	5002

To Implement the Aggregate functions concept on the table

Count the total number of salesmen.

```
select count(s_id) from salesman;
```

#	count(s_id)
1	6

Count the number of orders placed by each customer.

```
select cust_id, count(*) as order_count  
from orders  
group by cust_id;
```

#	cust_id	order_coun
1	3001	1
2	3002	1
3	3009	1
4	3005	2
5	3007	1
6	3003	1

Count the number of cities represented in the Salesman table.

```
select count(distinct s_city) as city_count  
from salesman;
```

#	city_coun
1	6

Count the total number of customers in each city.

```
select city, count(*) as customer_count  
from customer  
group by city;
```

#	city	customer_count
1	California	1
2	Paris	1
3	New York	1
4	London	2
5	Moscow	1

Count the number of orders placed by salesmen in 'Paris'

```
select salesman_id, count(*) as order_count
from orders
join salesman on orders.salesman_id = salesman.s_id
where salesman.s_city = 'Paris'
group by salesman_id;
```

Calculate the total purchase amount across all orders.

```
select sum(purch_amt) as total_purchase
from orders;
```

#	total_purchase
1	3999.385124206543

Calculate the total commission earned by all salesmen.

```
select sum(commission) as total_commission
from salesman;
```

#	total_commission
1	0.9400000050663948

Find the total purchase amount for each customer.

```
select cust_id, sum(purch_amt) as total_purchase
from orders
group by cust_id;
```

#	cust_id	total_purchase
1	3001	243.58499145507812
2	3002	165.260009765625
3	3009	110.5
4	3005	1004.1500244140625
5	3007	2400.60009765625
6	3003	75.29000091552734

Calculate the total purchase amount for orders placed on '2012-09-10'.

```
select sum(purch_amt) as total_purchase
from orders
where ord_date = '2012-09-10';
```

#	total_purchase
1	243.58499145507812

Find the total purchase amount for each salesman.
select salesman_id, sum(purch_amt) as total_purchase
from orders
group by salesman_id;

#	salesman_id	total_purchase
1	5005	243.58499145507812
2	5001	2565.860107421875
3	5003	110.5
4	5002	1004.1500244140625
5	5007	75.29000091552734

Find the average commission of salesmen
select avg(commission) as avg_commission
from salesman;

#	avg_commission
1	0.1566666675110658

Calculate the average purchase amount across all orders.
select avg(purch_amt) as avg_purchase
from orders;

#	avg_purchase
1	571.3407320295062

Find the average grade of customers grouped by their city.
select city, avg(grade) as avg_grade
from customer
group by city;

#	city	avg_grade
1	California	250.0000
2	Paris	300.0000
3	New York	200.0000
4	London	200.0000
5	Moscow	200.0000

Find the average purchase amount for orders placed by customers in 'New York'.
select avg(purch_amt) as avg_purchase
from orders
join customer on orders.cust_id = customer.cust_id
where customer.city = 'New York';

#	avg_purchase
1	2400.60009765625

Calculate the average purchase amount for each salesman.
select salesman_id, avg(purch_amt) as avg_purchase
from orders
group by salesman_id;

#	salesman_id	avg_purchase
1	5005	243.58499145507812
2	5001	1282.9300537109375
3	5003	110.5
4	5002	502.07501220703125
5	5007	75.29000091552734

Find the maximum purchase amount in the Orders table.

```
select max(purch_amt) as max_purchase
from orders;
```

#	max_purchase
1	2400.6

Find the highest grade assigned to a customer.

```
select max(grade) as max_grade
from customer;
```

#	max_grade
1	300

Find the salesman with the highest commission.

```
select name, commission
from salesman
where commission = (select max(commission) from salesman);
```

#	name	commisi
1	Pit Alex	0.33

Find the lowest purchase amount in the Orders table.

```
select min(purch_amt) as min_purchase
from orders;
```

#	min_purchase
1	75.29

Find the customer with the lowest grade.

```
select name, commission
from salesman
where commission = (select max(commision) from salesman);
```

#	cust_name	grad
1	Brad Guzan	150

Find the minimum commission among all salesmen.

```
select min(commission) as min_commission
from salesman;
```

#	min_commissio
1	0.1

SQL Query based on Join Concept

Retrieve employee names and their respective department names.

```
select e.name,d.dept_name from employees e join departments d on e.dept_id = d.dept_id;
```

#	name	dept_name
1	Charlie	IT
2	Alice	IT
3	Bob	HR
4	David	Finance
5	Jack	Sales
6	David	Finance

Retrieve employee names along with their salaries and department budgets

```
select e.name,e.salary,d.budget from employees e join departments d on e.dept_id = d.dept_id;
```

#	name	salary	budget
1	Charlie	65000	500000
2	Alice	60000	500000
3	Bob	55000	200000
4	David	70000	300000
5	Jack	67000	400000
6	David	70000	300000

List employees along with their job titles and department locations.

```
select e.job_title,d.location from employees e join departments d on e.dept_id = d.dept_id;
```

#	job_title	location
1	Data Analyst	New York
2	Software Eng	New York
3	HR Manager	Chicago
4	Accountant	Houston
5	Sales Exec	San Diego
6	Accountant	Houston

Retrieve the highest salary per department.

select max(e.salary) as salary , d.dept_name from employees e join departments d where e.dept_id = d.dept_id group by dept_name;

#	salary	dept_name
1	65000	IT
2	55000	HR
3	70000	Finance
4	67000	Sales

Retrieve all employees, even those without a department.

select name from employees;

#	name
1	Alice
2	Bob
3	Charlie
4	David
5	Eva
6	Jack

Retrieve employees with their department managers.

select e.name as name , d.manager as manager from employees e join departments d where e.dept_id = d.dept_id;

#	name	manager
1	Charlie	John Doe
2	Alice	John Doe
3	Bob	Sarah Lee
4	David	Mike Ross
5	Jack	Anna Kim
6	David	Mike Ross

Retrieve employees and include department emails if available.

select distinct e.name as name , d.email as email from employees e inner join departments d on e.dept_id = d.dept_id;

#	name	email
1	Charlie	it@company.com
2	Alice	it@company.com
3	Bob	hr@company.com
4	David	finance@company.com
5	Jack	salese@company.com

Retrieve all employees and show their department budget, even if they are not assigned to a Department.

```
select distinct e.name ,d.budget as budget from employees e left join departments d on e.dept_id = d.dept_id;
```

#	name	budget
1	Alice	500000
2	Bob	200000
3	Charlie	500000
4	David	300000
5	Eva	NULL
6	Jack	400000

List employees and their department's founding year.

```
select distinct e.name , d.founded_year from employees e inner join departments d on e.dept_id = d.dept_id;
```

#	name	founded_year
1	Charlie	2010
2	Alice	2010
3	Bob	2008
4	David	2012
5	Jack	2015

Retrieve all departments with employees (or NULL if no employees).

```
select distinct e.name, d.dept_name from employees e right join departments d on e.dept_id = d.dept_id;
```

#	name	dept_name
1	Charlie	IT
2	Alice	IT
3	Bob	HR
4	David	Finance
5	Jack	Sales
6	NULL	Marketing
7	NULL	Operations
8	NULL	R&D
9	NULL	Support
10	NULL	Legal
11	NULL	Logistics

Retrieve department managers along with employee names.

```
select distinct d.manager ,e.name from departments d inner join employees e on e.dept_id = d.dept_id;
```

#	manager	name
1	John Doe	Charlie
2	John Doe	Alice
3	Sarah Lee	Bob
4	Mike Ross	David
5	Anna Kim	Jack

Retrieve department phone numbers and employee names

`select distinct d.phone ,e.name from departments d left join employees e on e.dept_id = d.dept_id;`

#	phone	name
1	123-456-7890	Charlie
2	123-456-7890	Alice
3	234-567-8901	Bob
4	345-678-9012	David
5	456-789-0123	Jack
6	567-890-1234	NULL
7	678-901-2345	NULL
8	789-012-3456	NULL
9	890-123-4567	NULL
10	901-234-5678	NULL
11	012-345-6789	NULL

Retrieve department budgets along with employee salaries.

`select d.budget ,e.salary from departments d inner join employees e on e.dept_id = d.dept_id;`

#	budget	salary
1	500000	65000
2	500000	60000
3	200000	55000
4	300000	70000
5	400000	67000
6	300000	70000

List departments and their respective total employees, even if no employees exist.

`select d.dept_name ,count(e.emp_id) as total_emp from departments d left join employees e on e.dept_id = d.dept_id group by d.dept_name;`

#	dept_name	total_emp
1	IT	2
2	HR	1
3	Finance	2
4	Sales	1
5	Marketing	0
6	Operations	0
7	R&D	0
8	Support	0
9	Legal	0
10	Logistics	0

Retrieve all employees and departments, ensuring all data is included

`select e.name as name , d.dept_name as department from employees e left join departments d on e.dept_id = d.dept_id union`

select e.name as name , d.dept_name as department from employees e right join departments d on e.dept_id = d.dept_id ;

#	name	department
1	Alice	IT
2	Bob	HR
3	Charlie	IT
4	David	Finance
5	Eva	NULL
6	Jack	Sales
7	NULL	Marketing
8	NULL	Operations
9	NULL	R&D
10	NULL	Support
11	NULL	Legal
12	NULL	Logistics

Retrieve all departments with their budgets, ensuring no data is lost.
select dept_name , budget from departments;

#	dept_name	budget
1	IT	500000
2	HR	200000
3	Finance	300000
4	Sales	400000
5	Finance	300000
6	Marketing	350000
7	Operations	250000
8	R&D	600000
9	Support	150000
10	Legal	180000
11	Logistics	220000

List department emails and employees even if there is no relation
select d.email,e.name from departments d left join employees e on e.dept_id = d.dept_id;

#	email	name
1	it@company.com	Charlie
2	it@company.com	Alice
3	hr@company.com	Bob
4	finance@company.com	David
5	salese@company.com	Jack
6	finance@company.com	David
7	marketing@company.com	NULL
8	ops@company.com	NULL
9	rnd@company.com	NULL
10	support@company.com	NULL
11	legal@company.com	NULL
12	logistics@company.com	NULL

Find employees working in the same department as 'Alice'.

Retrieve employees with colleagues in the same department.
select e.name,e1.name as colleagues_with from employees e,employees e1 where e.emp_id <> e1.emp_id and e.dept_id = e1.dept_id;

#	name	colleagues_with
1	Charlie	Alice
2	Alice	Charlie

Find employees earning more than a specific colleague.

```
select e.name , e1.name as earning_more_than from employees e,employees e1 where e.salary > e1.salary order by e.name;
```

#	name	earning_more_than
1	Alice	Bob
2	Alice	Eva
3	Bob	Eva
4	Charlie	Alice
5	Charlie	Bob
6	Charlie	Eva
7	David	Alice
8	David	Bob
9	David	Charlie
10	David	Eva
11	David	Jack
12	Jack	Alice
13	Jack	Bob
14	Jack	Charlie
15	Jack	Eva

List employees with the same experience level.

```
select e.name,e1.name as same_experience_with, e.experience from employees e,employees e1 where e.emp_id <> e1.emp_id and e.experience = e1.experience;
```

#	name	same_experience_with	experience
1	Jack	David	9
2	David	Jack	9

Retrieve employees who joined the company on the same date.

```
select e.name,e1.name as name, e.joining_date from employees e,employees e1 where e.emp_id <> e1.emp_id and e.joining_date = e1.joining_date;
```

Retrieve employees working in departments with budgets greater than 300,000.

```
select e.name from employees e inner join departments d where e.dept_id = d.dept_id and d.budget > 300000;
```

#	name
1	Alice
2	Charlie
3	Jack

Retrieve departments with more than 20 employees

```
select dept_name,total_emp from departments where total_emp > 20;
```

#	dept_name	total_emp
1	IT	30
2	Sales	25
3	Marketing	22
4	R&D	35

Retrieve employees working in 'New York'.

```
select e.name , d.location from employees e inner join departments d on e.dept_id = d.dept_id and d.location = "New York";
```

#	name	location
1	Alice	New York
2	Charlie	New York

Retrieve the department having the maximum employees.

SQL Query Based on Left Join

List all employees along with their department names.

```
select e.name ,d.dept_name from employees e left join departments d on e.dept_id = d.dept_id;
```

#	name	dept_name
1	Alice	IT
2	Bob	HR
3	Charlie	IT
4	David	Finance
5	David	Finance
6	Eva	NULL
7	Jack	Sales

Get all employees and their respective manager names.

```
select e.name ,d.manager from employees e left join departments d on e.dept_id = d.dept_id;
```

#	name	manager
1	Alice	John Doe
2	Bob	Sarah Lee
3	Charlie	John Doe
4	David	Mike Ross
5	David	Mike Ross
6	Eva	NULL
7	Jack	Anna Kim

Show employees along with their department budgets.

```
select e.name ,d.budget from employees e left join departments d on e.dept_id = d.dept_id;
```

#	name	budget
1	Alice	500000
2	Bob	200000
3	Charlie	500000
4	David	300000
5	David	300000
6	Eva	NULL
7	Jack	400000

Get a list of employees along with the total employees in their departments.

```
select e.name ,d.total_emp from employees e left join departments d on e.dept_id = d.dept_id;
```

#	name	total_emp
1	Alice	30
2	Bob	15
3	Charlie	30
4	David	20
5	David	20
6	Eva	NULL
7	Jack	25

Show employees along with their department phone numbers.

`select e.name ,d.phone from employees e left join departments d on e.dept_id = d.dept_id;`

#	name	phone
1	Alice	123-456-7890
2	Bob	234-567-8901
3	Charlie	123-456-7890
4	David	345-678-9012
5	David	345-678-9012
6	Eva	NULL
7	Jack	456-789-0123

List all employees along with their department locations.

`select e.name ,d.location from employees e left join departments d on e.dept_id = d.dept_id;`

#	name	location
1	Alice	New York
2	Bob	Chicago
3	Charlie	New York
4	David	Houston
5	David	Houston
6	Eva	NULL
7	Jack	San Diego

Find employees and the founding year of their respective departments.

`select e.name ,d.founded_year from employees e left join departments d on e.dept_id = d.dept_id;`

#	name	founded_year
1	Alice	2010
2	Bob	2008
3	Charlie	2010
4	David	2012
5	David	2012
6	Eva	NULL
7	Jack	2015

List all employees with NULL department names.

Show all employees and their respective department emails.

`select e.name ,d.email from employees e left join departments d on e.dept_id = d.dept_id;`

#	name	email
1	Alice	it@company.com
2	Bob	hr@company.com
3	Charlie	it@company.com
4	David	finance@company.com
5	David	finance@company.com
6	Eva	NULL
7	Jack	salese@company.com

List employees and the number of employees in their department.

`select e.name ,count(e.emp_id) as total_emp from employees e left join departments d on e.dept_id = d.dept_id group by e.name;`

#	name	total_emp
1	Alice	1
2	Bob	1
3	Charlie	1
4	David	2
5	Eva	1
6	Jack	1

Find employees working in IT or Marketing departments.

select e.name ,d.dept_name from employees e left join departments d on e.dept_id = d.dept_id where d.dept_name = "IT" or d.dept_name = "Marketing";

#	name	dept_name
1	Alice	IT
2	Charlie	IT

Show employees whose department budget is greater than 300,000.

select e.name ,d.budget from employees e left join departments d on e.dept_id = d.dept_id where d.budget > 300000;

#	name	budget
1	Alice	500000
2	Charlie	500000
3	Jack	400000

Get employees and their managers for those who joined after 2018.

select e.name ,d.manager from employees e left join departments d on e.dept_id = d.dept_id where e.joining_date > "2018-01-01";

#	name	manager
1	Alice	John Doe
2	Bob	Sarah Lee
3	Charlie	John Doe
4	Eva	NULL

List all employees along with their department details.

select e.name ,d.* from employees e left join departments d on e.dept_id = d.dept_id;

#	name	dept_id	dept_name	manager	location	budget	total_emp	founded_year	phone	email	status
1	Alice	1	IT	John Doe	New York	500000	30	2010	123-456-7890	it@company.com	Active
2	Bob	2	HR	Sarah Lee	Chicago	200000	15	2008	234-567-8901	hr@company.com	Active
3	Charlie	1	IT	John Doe	New York	500000	30	2010	123-456-7890	it@company.com	Active
4	David	3	Finance	Mike Ross	Houston	300000	20	2012	345-678-9012	finance@company.com	Active
5	David	3	Finance	Mike Ross	Houston	300000	20	2012	345-678-9012	finance@company.com	Active
6	Eva	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
7	Jack	4	Sales	Anna Kim	San Diego	400000	25	2015	456-789-0123	salese@company.com	Active

SQL Query Based on Right Join

List all departments with employee details.

select d.dept_name , e.*from departments d right join employees e on e.dept_id = d.dept_id;

#	dept_name	emp_id	name	age	gender	dept_id	salary	job_title	experience	joining_date	location
1	IT	101	Alice	29	Female	1	60000	Software Eng	5	2019-05-21	New York
2	HR	102	Bob	32	Male	2	55000	HR Manager	7	2018-08-14	Chicago
3	IT	103	Charlie	27	Male	1	65000	Data Analyst	4	2020-01-10	Boston
4	Finance	104	David	35	Male	3	70000	Accountant	9	2015-06-30	Houston
5	Finance	104	David	35	Male	3	70000	Accountant	9	2015-06-30	Houston
6	HULL	105	Eva	30	Female	HULL	50000	Marketing Exec	6	2019-12-05	Seattle
7	Sales	110	Jack	34	Male	4	67000	Sales Exec	9	2015-03-25	San Jose

Get department names along with the total employees.

```
select d.dept_name , count(e.emp_id) from departments d right join employees e on e.dept_id = d.dept_id
group by d.dept_name;
```

#	dept_name	count(e.emp_id)
1	IT	2
2	HR	1
3	Finance	2
4	HULL	1
5	Sales	1

Show department managers and the employees under them.

```
select d.manager , e.name from departments d right join employees e on e.dept_id = d.dept_id;
```

#	manager	name
1	John Doe	Alice
2	Sarah Lee	Bob
3	John Doe	Charlie
4	Mike Ross	David
5	Mike Ross	David
6	HULL	Eva
7	Anna Kim	Jack

List department emails with employees assigned.

```
select d.email , e.name from departments d right join employees e on e.dept_id = d.dept_id;
```

#	email	name
1	it@company.com	Alice
2	hr@company.com	Bob
3	it@company.com	Charlie
4	finance@company.com	David
5	finance@company.com	David
6	HULL	Eva
7	salese@company.com	Jack

Get department budgets and associated employees.

```
select d.budget , e.name from departments d right join employees e on e.dept_id = d.dept_id;
```

#	budget	name
1	500000	Alice
2	200000	Bob
3	500000	Charlie
4	300000	David
5	300000	David
6	HULL	Eva
7	400000	Jack

Show department locations and their employees.

```
select d.location , e.name from departments d right join employees e on e.dept_id = d.dept_id;
```

#	location	name
1	New York	Alice
2	Chicago	Bob
3	New York	Charlie
4	Houston	David
5	Houston	David
6	NULL	Eva
7	San Diego	Jack

Find department status along with employees working there.

```
select d.status , e.name from departments d right join employees e on e.dept_id = d.dept_id;
```

#	status	name
1	Active	Alice
2	Active	Bob
3	Active	Charlie
4	Active	David
5	Active	David
6	NULL	Eva
7	Active	Jack

Show employees whose department budget is less than 300,000.

```
select d.budget , e.name from departments d right join employees e on e.dept_id = d.dept_id where d.budget < 300000;
```

#	budget	name
1	200000	Bob

List departments and employees who joined after 2020.

```
select d.dept_name , e.name ,e.joining_date from departments d right join employees e on e.dept_id = d.dept_id where e.joining_date > "2020-01-01";
```

#	dept_name	name	joining_date
1	IT	Charlie	2020-01-10

List all departments and their founding years along with employees assigned.

```
select d.dept_name , d.founded_year, e.name from departments d right join employees e on e.dept_id = d.dept_id;
```

#	dept_name	founded_year	name
1	IT	2010	Alice
2	HR	2008	Bob
3	IT	2010	Charlie
4	Finance	2012	David
5	Finance	2012	David
6	NULL	NULL	Eva
7	Sales	2015	Jack

Show department names with at least one employee assigned.

```
select d.dept_name,count(e.emp_id) from departments d right join employees e on e.dept_id = d.dept_id group by d.dept_name having count(e.emp_id) >= 1;
```

#	dept_name	count(e.emp_id)
1	IT	2
2	HR	1
3	Finance	2
4	NULL	1
5	Sales	1

Find departments that have no employees assigned.

```
select d.dept_name, count(e.emp_id) from departments d right join employees e on e.dept_id = d.dept_id
group by d.dept_name having count(e.emp_id) = 0;
```

Show all department names and their locations along with employees.

```
select d.dept_name , d.location, e.name from departments d right join employees e on e.dept_id =
d.dept_id;
```

#	dept_name	location	name
1	IT	New York	Alice
2	HR	Chicago	Bob
3	IT	New York	Charlie
4	Finance	Houston	David
5	Finance	Houston	David
6	NULL	NULL	Eva
7	Sales	San Diego	Jack

List department names and employees whose department was founded before 2012.

```
select d.dept_name , e.name, d.founded_year from departments d right join employees e on e.dept_id =
d.dept_id where d.founded_year < "2012-01-01";
```

#	dept_name	name	founded_year
1	IT	Alice	2010
2	HR	Bob	2008
3	IT	Charlie	2010

Get departments and the count of employees assigned to them.

```
select d.dept_name , count(e.emp_id) from departments d right join employees e on e.dept_id = d.dept_id
group by d.dept_name;
```

#	dept_name	count(e.emp_id)
1	IT	2
2	HR	1
3	Finance	2
4	NULL	1
5	Sales	1

To implement the concept of Indexes and views.

1. Creating an index on emp_name column

```
create index idx_emp_name on employees(emp_name);
```

2. Creating a composite index on emp_salary and emp_department

```
create index idx_emp_salary_dept on employees(salary, dept_name);
```

3. Creating a unique index on emp_id

```
create unique index idx_emp_id on employees(emp_id);
```

4. Creating a full-text index on emp_name for searching

```
create fulltext index idx_emp_name_fulltext on employees(emp_name);
```

5. Checking all indexes in the employees table

```
show indexes from employees;
```

#	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collator	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_commen	Visible	Expression
1	employees	0	idx_emp_id	1	emp_id	A	5			YES	BTREE			YES	
2	employees	1	idx_emp_name	1	emp_name	A	5			YES	BTREE			YES	
3	employees	1	idx_emp_salary_dept	1	salary	A	5			YES	BTREE			YES	

6. Dropping an index

```
drop index idx_emp_name on employees;
```

7. Using the indexed column in a query to improve performance

```
select * from employees where emp_name = 'Alice';
```

8. Creating an index on departments table for faster joins

```
create index idx_dept_id on department(dept_id);
```

9. Creating an index on projects table to speed up filtering

```
create index idx_start_date on project(start_date);
create index idx_end_date on project(end_date);
```

10. Checking query execution plan using EXPLAIN
`explain select * from employees where emp_name = 'Alice';`

11. Creating a view for high-salary employees
`create view high_salary_employees as`
`select emp_id, emp_name, salary`
`from employees`
`where salary > 50000; -- Adjust the salary threshold as needed`

12. Selecting data from the view
`select * from high_salary_employees;`

#	emp_id	emp_name	salary
1	1	Alice	60000
2	5	Emma	65000
3	2	Bob	75000
4	4	David	90000

13. Creating a view for employees who joined after 2020
`create view employees_joined_after_2020 as`
`select emp_id, emp_name, joining_date`
`from employees`
`where joining_date > '2020-01-01';`

14. Creating a view joining employees and departments
`create view employees_with_departments as`
`select e.emp_id, e.emp_name, e.salary, e.dept_name, d.manager_id`
`from employees e`
`join department d on e.dept_name = d.dept_name;`

15. Updating a view (requires an updateable view)
`update high_salary_employees`
`set salary = salary + 5000`
`where emp_name = 'Alice';`

16. Dropping a view
`drop view high_salary_employees;`

17. Creating a view to count employees per department
`create view employee_count_per_dept as`
`select dept_name, count(*) as employee_count`
`from employees`
`group by dept_name;`

18. Using a view in a query
`select * from employee_count_per_dept;`

19. Creating a view for active projects

```
create view active_projects as
select p_id, p_name, start_date, end_date, dept_id
from project
where end_date > current_date() or end_date is null;
```

To implement the concept of Sub-Queries

-- Basic Subquery Queries

-- Get employees who earn more than the average salary

select name from employees where salary > (select avg(salary) from employees);

#	name
1	Charlie
2	David
3	Jack

-- Get employees who work in the IT department

select name from employees where dept_id = (select dept_id from departments where dept_name = "IT");

#	name
1	Alice
2	Charlie

-- Find the highest salary among employees

select max(salary) from employees;

#	max(salary)
1	70000

-- Find employees who earn the lowest salary

select min(salary) from employees;

#	min(salary)
1	50000

-- List employees who work in New York

select name from employees where dept_id = (select dept_id from departments where location="New York");

#	name
1	Alice
2	Charlie

-- Find employees who earn more than the company's average salary
 select name from employees where salary > (select avg(salary) from employees);

#	name
1	Charlie
2	David
3	Jack

-- Get employees who work in the HR department
 select name from employees where dept_id = (select dept_id from departments where dept_name="HR");

#	name
1	Bob

-- List employees working in the city where the IT department is located
 select name from employees where location = (select location from departments where dept_name="IT");

#	name
1	Alice

-- Find employees who work in the department with the lowest budget
 select name from employees where dept_id = (select dept_id from departments where budget = (select min(budget) from departments));

-- Find employees who earn the second-highest salary in their department
 select name,salary from employees where salary = (select max(salary) from employees where salary < (select max(salary) from employees));

#	name	salary
1	Jack	67000

-- Subqueries with Multiple Conditions
 -- Get employees whose salary is greater than 60,000 and work in Finance
 select name,salary from employees where salary > 60000 and dept_id in (select dept_id from departments where dept_name ="Finance");

#	name	salary
1	David	70000

-- Get the names of employees who work in the department with the highest budget
 select name from employees where dept_id = (select dept_id from departments where budget = (select max(budget) from departments));

-- Find employees who joined after the first employee in the company
 select name,joining_date from employees where joining_date = (select min(joining_date) from employees where joining_date > (select min(joining_date) from employees));

#	name	joining_date
1	David	2015-06-30

-- List employees who work in departments founded after 2010
 select name from employees where dept_id in (select dept_id from departments where founded_year > "2010-01-01");

#	name
1	David
2	Jack

-- Find employees who have a salary greater than their department's average salary
 select name from employees where salary > (select avg(salary) from employees);

#	name
1	Charlie
2	David
3	Jack

-- Correlated Subqueries

-- Find employees who have the same salary as another employee

select e1.name from employees e1 join employees e2 on e1.salary = e2.salary and e1.emp_id != e2.emp_id;

-- Get employees who work in a department with more than 20 employees

select name from employees where dept_id in (select dept_id from departments where total_emp > 20);

#	name
1	Alice
2	Charlie
3	Jack

-- Get the youngest employee in each department

select e.name,e.age
 from employees e
 where age = (
 select min(age)
 from employees
 where dept_id = e.dept_id
);

#	name	age
1	Bob	32
2	Charlie	27
3	David	35
4	Jack	34

-- Find employees who have the highest salary in their department

select e.name,e.salary
 from employees e
 where salary = (
 select max(salary)
 from employees
 where dept_id = e.dept_id
);

#	name	salary
1	Bob	55000
2	Charlie	65000
3	David	70000
4	Jack	67000

-- Nested Subqueries

-- Get the second highest salary in the company

select max(salary)
 from employees
 where salary < (
 select max(salary)
 from employees

```
);
```

#	max(salary)
1	67000

```
-- Find the department with the second-highest budget
```

```
select dept_name, budget
```

```
from departments
```

```
where budget = (
```

```
    select max(budget)
```

```
    from departments
```

```
    where budget < (
```

```
        select max(budget)
```

```
        from departments
```

```
    )
```

```
);
```

#	dept_name	budget
1	IT	500000

```
select name from employees e where e.salary > (
```

```
    select min(e2.salary)
```

```
    from employees e2
```

```
    join departments d on e2.dept_id = d.dept_id
```

```
    where d.dept_name = 'HR'
```

```
);
```

#	name
1	Alice
2	Charlie
3	David
4	Jack

```
-- Advanced Level Subquery
```

```
-- Get employees who have the highest salary in their respective city
```

```
select e.name,e.salary,e.location
```

```
from employees e
```

```
join departments d on e.dept_id = d.dept_id
```

```
where e.salary = (
```

```
    select max(e2.salary)
```

```
    from employees e2
```

```
    join departments d2 on e2.dept_id = d2.dept_id
```

```
    where d2.location = d.location
```

```
);
```

#	name	salary	location
1	Charlie	65000	Boston
2	Bob	55000	Chicago
3	David	70000	Houston
4	Jack	67000	San Jose
5	David	70000	Houston

```
-- Find employees who are the only ones in their department
```

```
select e.name
```

```

from employees e
where (
  select count(*)
  from employees e2
  where e2.dept_id = e.dept_id
) = 1;

```

#	name
1	Bob
2	David
3	Jack

```

-- Get employees whose salary is greater than at least one employee in IT
select e.name,e.salary
from employees e
where e.salary in (
  select e2.salary
  from employees e2
  join departments d on e2.dept_id = d.dept_id
  where d.dept_name = 'IT'
);

```

#	name	salary
1	Alice	60000
2	Charlie	65000

To apply the concept of security and privileges.

```
CREATE DATABASE companyDB;
```

```
USE companyDB;
```

```

CREATE TABLE employees (
  id INT PRIMARY KEY AUTO_INCREMENT, name
  VARCHAR(100),
  position VARCHAR(100), salary
  DECIMAL(10,2)
);

```

```

CREATE TABLE departments (
  id INT PRIMARY KEY AUTO_INCREMENT,
  dept_name VARCHAR(100)
);

```

USERS :

```
CREATE USER 'alice'@'localhost' IDENTIFIED BY 'password1';
```

```
CREATE USER 'bob'@'localhost' IDENTIFIED BY 'password2';
```

```
CREATE USER 'charlie'@'localhost' IDENTIFIED BY 'password3';
```

```
CREATE USER 'dave'@'localhost' IDENTIFIED BY 'password4';
```

```
CREATE USER 'eve'@'localhost' IDENTIFIED BY 'password5';
```

Grant Queries :

1. Grant SELECT on employees to Alice

```
→ GRANT SELECT ON companyDB.employees TO 'alice'@'localhost';
```

-- 2. Grant SELECT, INSERT on employees to Bob

```
→ GRANT SELECT, INSERT ON companyDB.employees TO 'bob'@'localhost';
```

-- 3. Grant ALL PRIVILEGES on companyDB to Charlie

```
→ GRANT ALL PRIVILEGES ON companyDB.* TO 'charlie'@'localhost';
```

-- 4. Grant UPDATE on employees to Dave

```
→ GRANT UPDATE ON companyDB.employees TO 'dave'@'localhost';
```

-- 5. Grant DELETE on departments to Eve

```
→ GRANT DELETE ON companyDB.departments TO 'eve'@'localhost';
```

6. Grant SELECT on all tables to Alice

```
→ GRANT SELECT ON companyDB.* TO 'alice'@'localhost';
```

-- 7. Grant INSERT, UPDATE on departments to Bob

```
→ GRANT INSERT, UPDATE ON companyDB.departments TO 'bob'@'localhost';
```


-- 8. Grant DROP on database to Charlie
→ GRANT DROP ON *.* TO 'charlie'@'localhost';

-- 9. Grant CREATE on database to Dave
→ GRANT CREATE ON *.* TO 'dave'@'localhost';

-- 10. Grant EXECUTE on all procedures (if any) to Eve
→ GRANT EXECUTE ON companyDB.* TO 'eve'@'localhost';

Revoke Queries :

1. Revoke SELECT on employees from Alice
→ REVOKE SELECT ON companyDB.employees FROM 'alice'@'localhost';

2. Revoke INSERT from Bob
→ REVOKE INSERT ON companyDB.employees FROM 'bob'@'localhost';

3. Revoke ALL PRIVILEGES from Charlie
→ REVOKE ALL PRIVILEGES ON companyDB.* FROM 'charlie'@'localhost';

4. Revoke UPDATE on employees from Dave
→ REVOKE UPDATE ON companyDB.employees FROM 'dave'@'localhost';

5. Revoke DELETE on departments from Eve
→ REVOKE DELETE ON companyDB.departments FROM 'eve'@'localhost';

6. Revoke SELECT on all tables from Alice

→ REVOKE SELECT ON companyDB.* FROM 'alice'@'localhost';

7. Revoke UPDATE on departments from Bob

→ REVOKE UPDATE ON companyDB.departments FROM 'bob'@'localhost';

8. Revoke DROP on database from Charlie

→ REVOKE DROP ON *.* FROM 'charlie'@'localhost';

9. Revoke CREATE on database from Dave

→ REVOKE CREATE ON *.* FROM 'dave'@'localhost';

10. Revoke EXECUTE on all procedures from Eve

→ REVOKE EXECUTE ON companyDB.* FROM 'eve'@'localhost';

Create a database of Hotel Management System.

Create Database and Tables

```
mysql>
mysql>
mysql> CREATE DATABASE HotelManagementSystem;
Query OK, 1 row affected (0.02 sec)
```

```
mysql> CREATE TABLE Rooms (
->     RoomID INT PRIMARY KEY,
->     RoomNumber INT,
->     RoomType VARCHAR(255),
->     Rate DECIMAL(10, 2)
-> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> desc Rooms;
```

Field	Type	Null	Key	Default	Extra
RoomID	int	NO	PRI	NULL	
RoomNumber	int	YES		NULL	
RoomType	varchar(255)	YES		NULL	
Rate	decimal(10,2)	YES		NULL	

```
4 rows in set (0.00 sec)
```

```
mysql>
mysql> CREATE TABLE Guests (
->     GuestID INT PRIMARY KEY,
->     Name VARCHAR(255),
->     Email VARCHAR(255),
->     Phone VARCHAR(20)
-> );
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> desc Guests;
```

Field	Type	Null	Key	Default	Extra
GuestID	int	NO	PRI	NULL	
Name	varchar(255)	YES		NULL	
Email	varchar(255)	YES		NULL	
Phone	varchar(20)	YES		NULL	

```
4 rows in set (0.00 sec)
```

```
mysql>
mysql> CREATE TABLE Reservations (
->     ReservationID INT PRIMARY KEY,
->     GuestID INT,
->     RoomID INT,
->     CheckInDate DATE,
->     CheckOutDate DATE,
->     FOREIGN KEY (GuestID) REFERENCES Guests(GuestID),
->     FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID)
-> );
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> desc Reservations;
```

Field	Type	Null	Key	Default	Extra
ReservationID	int	NO	PRI	NULL	
GuestID	int	YES	MUL	NULL	
RoomID	int	YES	MUL	NULL	
CheckInDate	date	YES		NULL	
CheckOutDate	date	YES		NULL	

```
5 rows in set (0.00 sec)
```

```
mysql> select* from Reservations;
```

ReservationID	GuestID	RoomID	CheckInDate	CheckOutDate
1	1	1	2023-03-01	2023-03-04

```
1 row in set (0.00 sec)
```

```
mysql> select* from Guests;
```

GuestID	Name	Email	Phone
1	John Doe	johndoe2@example.com	123-456-7890

```
1 row in set (0.00 sec)
```

```
mysql> select* from Payments;
```

PaymentID	ReservationID	PaymentDate	Amount
1	1	2023-03-01	250.00

```
1 row in set (0.00 sec)
```

```
mysql>
mysql> CREATE TABLE Payments (
  ->   PaymentID INT PRIMARY KEY,
  ->   ReservationID INT,
  ->   PaymentDate DATE,
  ->   Amount DECIMAL(10, 2),
  ->   FOREIGN KEY (ReservationID) REFERENCES Reservations(ReservationID)
  -> );
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> desc Payments;
```

Field	Type	Null	Key	Default	Extra
PaymentID	int	NO	PRI	NULL	
ReservationID	int	YES	MUL	NULL	
PaymentDate	date	YES		NULL	
Amount	decimal(10,2)	YES		NULL	

```
4 rows in set (0.00 sec)
```

#Update Data

UPDATE Rooms

SET Rate = 120.00

WHERE RoomID = 1;

UPDATE Guests

SET Email = 'johndoe2@example.com'

WHERE GuestID = 1;

UPDATE Reservations

SET CheckOutDate = '2023-03-04'

WHERE ReservationID = 1;

UPDATE Payments

SET Amount = 250.00

WHERE PaymentID = 1;

Delete Data

DELETE FROM Payments

WHERE PaymentID = 2;

DELETE FROM Reservations

WHERE ReservationID = 2;

DELETE FROM Guests

WHERE GuestID = 2;

DELETE FROM Rooms
WHERE RoomID = 3;

```
mysql> select* from Rooms;
```

RoomID	RoomNumber	RoomType	Rate
1	101	Single	120.00
2	102	Double	150.00

```
2 rows in set (0.00 sec)
```

```
mysql> select* from Reservations;
```

ReservationID	GuestID	RoomID	CheckInDate	CheckOutDate
1	1	1	2023-03-01	2023-03-04

```
1 row in set (0.00 sec)
```

```
mysql> select* from Guests;
```

GuestID	Name	Email	Phone
1	John Doe	johndoe2@example.com	123-456-7890

```
1 row in set (0.00 sec)
```

```
mysql> select* from Payments;
```

PaymentID	ReservationID	PaymentDate	Amount
1	1	2023-03-01	250.00

```
1 row in set (0.00 sec)
```

Create a database of university.

CREATE DATABASE University;

USE University;

```
mysql>
mysql>
mysql> CREATE DATABASE University;
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> USE University;
Database changed
mysql>
```

CREATE TABLE Students (
 StudentID INT PRIMARY KEY,
 Name VARCHAR(255),
 Email VARCHAR(255),
 Department VARCHAR(255)
);

```
mysql>
mysql>
mysql> INSERT INTO Students (StudentID, Name, Email, Department)
-> VALUES
-> (1, 'John Doe', 'johndoe@example.com', 'Computer Science'),
-> (2, 'Jane Smith', 'janesmith@example.com', 'Mathematics'),
-> (3, 'Bob Johnson', 'bobjohnson@example.com', 'Engineering');
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> select* from Students;
```

StudentID	Name	Email	Department
1	John Doe	johndoe@example.com	Computer Science
2	Jane Smith	janesmith@example.com	Mathematics
3	Bob Johnson	bobjohnson@example.com	Engineering

3 rows in set (0.00 sec)

CREATE TABLE Faculty (
 FacultyID INT PRIMARY KEY,
 Name VARCHAR(255),
 Email VARCHAR(255),
 Department VARCHAR(255)
);

```
mysql>
mysql>
mysql> INSERT INTO Faculty (FacultyID, Name, Email, Department)
-> VALUES
-> (1, 'Dr. Smith', 'smith@example.com', 'Computer Science'),
-> (2, 'Dr. Johnson', 'johnson@example.com', 'Mathematics'),
-> (3, 'Dr. Williams', 'williams@example.com', 'Engineering');
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> select* from Faculty;
```

FacultyID	Name	Email	Department
1	Dr. Smith	smith@example.com	Computer Science
2	Dr. Johnson	johnson@example.com	Mathematics
3	Dr. Williams	williams@example.com	Engineering

3 rows in set (0.00 sec)

```
CREATE TABLE Courses (
  CourseID INT PRIMARY KEY,
  CourseName VARCHAR(255),
  Credits INT,
  Department VARCHAR(255)
);
```

```
mysql>
mysql>
mysql> INSERT INTO Courses (CourseID, CourseName, Credits, Department)
-> VALUES
-> (1, 'Introduction to Programming', 3, 'Computer Science'),
-> (2, 'Calculus', 4, 'Mathematics'),
-> (3, 'Thermodynamics', 3, 'Engineering');
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> select* from Courses;
```

CourseID	CourseName	Credits	Department
1	Introduction to Programming	3	Computer Science
2	Calculus	4	Mathematics
3	Thermodynamics	3	Engineering

```
3 rows in set (0.00 sec)
```

```
CREATE TABLE Enrollment (
  EnrollmentID INT PRIMARY KEY,
  StudentID INT,
  CourseID INT,
  Semester VARCHAR(255),
  Year INT,
  FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
  FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
```

```
mysql>
mysql>
mysql> INSERT INTO Enrollment (EnrollmentID, StudentID, CourseID, Semester, Year)
-> VALUES
-> (1, 1, 1, 'Fall', 2022),
-> (2, 2, 2, 'Spring', 2023),
-> (3, 3, 3, 'Fall', 2022);
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> select* from Enrollment;
```

EnrollmentID	StudentID	CourseID	Semester	Year
1	1	1	Fall	2022
2	2	2	Spring	2023
3	3	3	Fall	2022

```
3 rows in set (0.00 sec)
```

```
CREATE TABLE Grades (  
  GradeID INT PRIMARY KEY,  
  EnrollmentID INT,  
  Grade VARCHAR(10),  
  FOREIGN KEY (EnrollmentID) REFERENCES Enrollment(EnrollmentID)  
);
```

```
mysql>  
mysql>  
mysql> INSERT INTO Grades (GradeID, EnrollmentID, Grade)  
-> VALUES  
-> (1, 1, 'A'),  
-> (2, 2, 'B+'),  
-> (3, 3, 'A-');
```

```
Query OK, 3 rows affected (0.00 sec)  
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> select*from Grades;
```

GradeID	EnrollmentID	Grade
1	1	A
2	2	B+
3	3	A-

```
3 rows in set (0.00 sec)
```


Create a database of E-Commerce business.

Create Database and Tables

```
mysql>
mysql> CREATE DATABASE ECommerce;
Query OK, 1 row affected (0.01 sec)

mysql> USE ECommerce;
Database changed
```

```
CREATE TABLE Customers (
  CustomerID INT PRIMARY KEY,
  Name VARCHAR(255),
  Email VARCHAR(255),
  Phone VARCHAR(20),
  Address VARCHAR(255)
);
```

```
mysql>
mysql> CREATE TABLE Customers (
  -> CustomerID INT PRIMARY KEY,
  -> Name VARCHAR(255),
  -> Email VARCHAR(255),
  -> Phone VARCHAR(20),
  -> Address VARCHAR(255)
  -> );
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO Customers (CustomerID, Name, Email, Phone, Address)
  -> VALUES
  -> (1, 'John Doe', 'johndoe@example.com', '123-456-7890', '123 Main St'),
  -> (2, 'Jane Smith', 'janesmith@example.com', '987-654-3210', '456 Elm St');
Query OK, 2 rows affected (0.01 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> select* from Customers;
+-----+-----+-----+-----+-----+
| CustomerID | Name | Email | Phone | Address |
+-----+-----+-----+-----+-----+
| 1 | John Doe | johndoe@example.com | 123-456-7890 | 123 Main St |
| 2 | Jane Smith | janesmith@example.com | 987-654-3210 | 456 Elm St |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
CREATE TABLE Products (
  ProductID INT PRIMARY KEY,
  ProductName VARCHAR(255),
  Description TEXT,
  Price DECIMAL(10, 2),
  StockQuantity INT
);
```

```
mysql> CREATE TABLE Products (
  -> ProductID INT PRIMARY KEY,
  -> ProductName VARCHAR(255),
  -> Description TEXT,
  -> Price DECIMAL(10, 2),
  -> StockQuantity INT
  -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO Products (ProductID, ProductName, Description, Price, StockQuantity)
  -> VALUES
  -> (1, 'Apple Watch', 'A smartwatch from Apple', 299.99, 10),
  -> (2, 'Samsung TV', 'A 4K TV from Samsung', 999.99, 5),
  -> (3, 'Nike Shoes', 'Running shoes from Nike', 79.99, 20);
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select * from Products;
+-----+-----+-----+-----+-----+
| ProductID | ProductName | Description | Price | StockQuantity |
+-----+-----+-----+-----+-----+
| 1 | Apple Watch | A smartwatch from Apple | 299.99 | 10 |
| 2 | Samsung TV | A 4K TV from Samsung | 999.99 | 5 |
| 3 | Nike Shoes | Running shoes from Nike | 79.99 | 20 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
CREATE TABLE Orders (
  OrderID INT PRIMARY KEY,
  CustomerID INT,
  OrderDate DATE,
  Total DECIMAL(10, 2),
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

```
mysql>
mysql> CREATE TABLE Orders (
  ->   OrderID INT PRIMARY KEY,
  ->   CustomerID INT,
  ->   OrderDate DATE,
  ->   Total DECIMAL(10, 2),
  ->   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
  -> );
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> INSERT INTO Orders (OrderID, CustomerID, OrderDate, Total)
  -> VALUES
  -> (1, 1, '2022-01-01', 299.99),
  -> (2, 2, '2022-01-15', 999.99),
  -> (3, 1, '2022-02-01', 159.98);
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> select * from Orders;
```

OrderID	CustomerID	OrderDate	Total
1	1	2022-01-01	299.99
2	2	2022-01-15	999.99
3	1	2022-02-01	159.98

```
3 rows in set (0.00 sec)
```

```
CREATE TABLE OrderItems (
  OrderItemID INT PRIMARY KEY,
  OrderID INT,
  ProductID INT,
  Quantity INT,
  FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

```
mysql>
mysql>
mysql> CREATE TABLE OrderItems (
  ->   OrderItemID INT PRIMARY KEY,
  ->   OrderID INT,
  ->   ProductID INT,
  ->   Quantity INT,
  ->   FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
  ->   FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
  -> );
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> INSERT INTO OrderItems (OrderItemID, OrderID, ProductID, Quantity)
  -> VALUES
  -> (1, 1, 1, 1),
  -> (2, 2, 2, 1),
  -> (3, 3, 3, 2);
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> select * from OrderItems;
```

OrderItemID	OrderID	ProductID	Quantity
1	1	1	1
2	2	2	1
3	3	3	2

```
3 rows in set (0.00 sec)
```

```
CREATE TABLE Payments (  
    PaymentID INT PRIMARY KEY,  
    OrderID INT,  
    PaymentMethod VARCHAR(255),  
    PaymentDate DATE,  
    Amount DECIMAL(10, 2),  
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)  
);
```

```
mysql>  
mysql>  
mysql> CREATE TABLE Payments (  
->     PaymentID INT PRIMARY KEY,  
->     OrderID INT,  
->     PaymentMethod VARCHAR(255),  
->     PaymentDate DATE,  
->     Amount DECIMAL(10, 2),  
->     FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)  
-> );  
Query OK, 0 rows affected (0.05 sec)  
  
mysql> INSERT INTO Payments (PaymentID, OrderID, PaymentMethod, PaymentDate, Amount)  
-> VALUES  
-> (1, 1, 'Credit Card', '2022-01-01', 299.99),  
-> (2, 2, 'PayPal', '2022-01-15', 999.99),  
-> (3, 3, 'Credit Card', '2022-02-01', 159.98);  
Query OK, 3 rows affected (0.01 sec)  
Records: 3  Duplicates: 0  Warnings: 0  
  
mysql> select * from Payments;  
+-----+-----+-----+-----+-----+  
| PaymentID | OrderID | PaymentMethod | PaymentDate | Amount |  
+-----+-----+-----+-----+-----+  
| 1 | 1 | Credit Card | 2022-01-01 | 299.99 |  
| 2 | 2 | PayPal | 2022-01-15 | 999.99 |  
| 3 | 3 | Credit Card | 2022-02-01 | 159.98 |  
+-----+-----+-----+-----+-----+  
3 rows in set (0.00 sec)
```