PRACTICE

1. Select: Used to select data from database

```
SELECT * FROM customers;
```

customer_id	first_name	last_name	email	address_id
1	Mary	Smith	mary.smith@mailid.com	5
2	Patricia	Johnson	patricia.john@mailid.com	6
3	Madan	Mohan	madan.mohan@mailid.com	7
4	Barbara	Jones	barbara.jones@mailid.com	8
5	Elizabeth	Brown	elizabeth.brown@mailid.com	9
6	Jennifer	Davis	jennifer.davis@mailid.com	10
7	Maria	Miller	maria.miller@mailid.com	11
8	Susan	Wilson	susan.wilson@mailid.com	12
9	R	Madhav	r.madhav@mailid.com	13
10	Dorothy	Taylor	dorothy.taylor@mailid.com	14
11	Lisa	Anderson	lisa.anderson@mailid.com	15
12	Nancy	Thomas	nancy.thomas@mailid.com	16

2. From: specifies the table to select or delete data from

```
SELECT * FROM payments;
```

customer_id	amount	payment_mode	payment_date
1	60	Cash	2020-09-24
2	30	Credit Card	2020-04-27
3	90	Credit Card	2020-07-07
4	50	Debit Crad	2020-02-12
5	40	Mobile Payment	2020-11-20
6	40	Debit Crad	2021-06-28
7	10	Cash	2021-08-25
8	30	Mobile Payment	2021-06-17
9	80	Cash	2021-08-25
10	50	Mobile Payment	2021-11-03
11	70	Cash	2022-11-01
12	60	Netbanking	2022-09-11

3. Where: It is used to filter record

```
SELECT * FROM payments where payment_mode = 'Cash';
```

customer_id	amount	payment_mode	payment_date
1	60	Cash	2020-09-24
7	10	Cash	2021-08-25
9	80	Cash	2021-08-25
11	70	Cash	2022-11-01

4. Insert: It is used to insert new records into a table

```
INSERT INTO payments VALUES
(1,     60,     'Cash', '2020-09-24'),
(2,     30,     'Credit Card',  '2020-04-27'),
(3,     90,     'Credit Card',  '2020-07-07'),
(4,     50,     'Debit Crad',   '2020-02-12'),
(5,     40,     'Mobile Payment',       '2020-11-20'),
(6,     40,     'Debit Crad',   '2021-06-28'),
(7,     10,     'Cash', '2021-08-25'),
(8,     30,     'Mobile Payment',       '2021-06-17'),
(9,     80,     'Cash', '2021-08-25'),
(10, 50, 'Mobile Payment',      '2021-11-03'),
(11, 70, 'Cash', '2022-11-01'),
(12, 60, 'Netbanking', '2022-09-11'),
(13, 30, 'Netbanking', '2022-12-10'),
(14, 50, 'Credit Card', '2022-05-14'),
(15, 30, 'Credit Card', '2022-09-25');
```

customer_id	amount	payment_mode	payment_date
1	60	Cash	2020-09-24
2	30	Credit Card	2020-04-27
3	90	Credit Card	2020-07-07
4	50	Debit Crad	2020-02-12
5	40	Mobile Payment	2020-11-20
6	40	Debit Crad	2021-06-28
7	10	Cash	2021-08-25
8	30	Mobile Payment	2021-06-17
9	80	Cash	2021-08-25
10	50	Mobile Payment	2021-11-03
11	70	Cash	2022-11-01
12	60	Netbanking	2022-09-11
13	30	Netbanking	2022-12-10
14	50	Credit Card	2022-05-14
15	30	Credit Card	2022-09-25

5. Update: It is udes to modifies existing data in a table

```
UPDATE payments SET payment_mode = 'Paytm' WHERE
payment_mode = 'Mobile Payment';
```

customer_id	amount	payment_mode	payment_date
1	60	Cash	2020-09-24
2	30	Credit Card	2020-04-27
3	90	Credit Card	2020-07-07
4	50	Debit Crad	2020-02-12
5	40	Paytm	2020-11-20
6	40	Debit Crad	2021-06-28
7	10	Cash	2021-08-25
8	30	Paytm	2021-06-17
9	80	Cash	2021-08-25
10	50	Paytm	2021-11-03
11	70	Cash	2022-11-01
12	60	Netbanking	2022-09-11
13	30	Netbanking	2022-12-10
14	50	Credit Card	2022-05-14
15	30	Credit Card	2022-09-25

6. Delete: delete the data from table

```
DELETE FROM payments WHERE amount < 30;
```

customer_id	amount	payment_mode	payment_date
1	60	Cash	2020-09-24
2	30	Credit Card	2020-04-27
3	90	Credit Card	2020-07-07
4	50	Debit Crad	2020-02-12
5	40	Paytm	2020-11-20
6	40	Debit Crad	2021-06-28
8	30	Paytm	2021-06-17
9	80	Cash	2021-08-25
10	50	Paytm	2021-11-03
11	70	Cash	2022-11-01
12	60	Netbanking	2022-09-11
13	30	Netbanking	2022-12-10
14	50	Credit Card	2022-05-14
15	30	Credit Card	2022-09-25

7. Create: it is used to create a new table in the database;

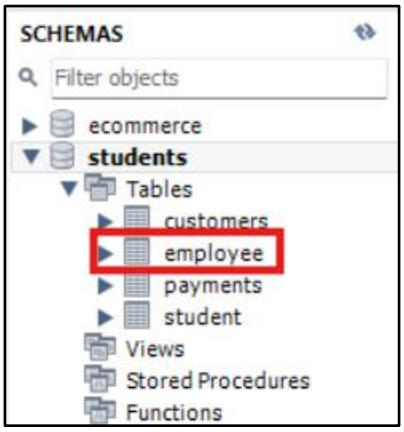
```
CREATE TABLE payments(customer_id int,  amount int, payment_mode varchar(20),
payment_date date);
```

Field	Type	Null	Key	Default	Extra
customer_id	int	YES		NULL	
amount	int	YES		NULL	
payment_mode	varchar(20)	YES		NULL	
payment_date	date	YES		NULL	

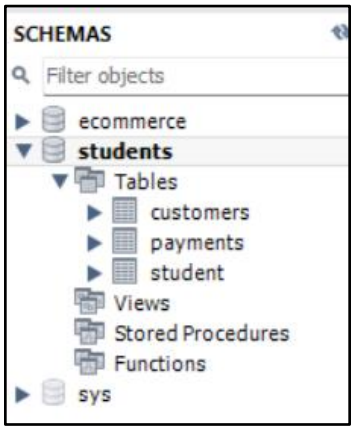
8. Drop: It is used to deletes a table

```
DROP TABLE employee;
```

Before



After command execute



9. Modify: Modifies the existing table

BEFORE: -- Create a table customers

```
CREATE TABLE customers(customer_id int,      first_name varchar(20),last_name varchar(20),
                       email varchar(25),      address_id text);
```

```
ALTER TABLE customers MODIFY email varchar (50);
```

Field	Type	Null	Key	Default	Extra
customer_id	int	YES		NULL	
first_name	varchar(20)	YES		NULL	
last_name	varchar(20)	YES		NULL	
email	varchar(50)	YES		NULL	
address_id	text	YES		NULL	

10. Inner Join: It returns records with matching value in both tables.

```
SELECT c.first_name, c.last_name, p.amount, p.payment_mode FROM customers as c
INNER JOIN  payments as p on c.customer_id = p.customer_id WHERE p.amount >= 50;
```

first_name	last_name	amount	payment_mode
Mary	Smith	60	Cash
Madan	Mohan	90	Credit Card
Barbara	Jones	50	Debit Crad
R	Madhav	80	Cash
Dorothy	Taylor	50	Paytm
Lisa	Anderson	70	Cash
Nancy	Thomas	60	Netbanking
Betty	White	50	Credit Card

11. Left Join: it returns all the records from the left table and matched records from the right table.

Fruits table:

Order_id	Fruit_name	Quantity
1001	Apple	20
1002	Mango	30
1004	Banana	12
1005	Kiwi	30
1007	Orange	27
1008	Dragon Fruit	10
1009	Guava	15
1011	Grapes	35
1012	PineApple	15

Orders Table

Order_id	Price	Payment_mode
1001	2000	Paytm
1002	2500	Cash
1003	3100	PhonePay
1004	1200	Cash
1005	3500	Paytm
1007	1500	Cash
1008	4000	PhonePay
1010	1000	Cash
1011	2500	Paytm
1012	2800	Cash

```
SELECT f.Fruit_name, o.Price, f.Quantity from fruits as f LEFT JOIN  orders as o on f.order_id = o.order_id;
```

Fruit_name	Price	Quantity
Apple	2000	20
Mango	2500	30
Banana	1200	12
Kiwi	3500	30
Orange	1500	27
Dragon Fruit	4000	10
Guava	NULL	15
Grapes	2500	35
PineApple	2800	15

12. Right Join: It returns all the records from the right table and the matched records from both the tables

```
SELECT orders.order_id, fruits.Fruit_name, orders.Price from fruits RIGHT JOIN orders on orders.order_id = fruits.order_id;
```

order_id	Fruit_name	Price
1001	Apple	2000
1002	Mango	2500
1003	NULL	3100
1004	Banana	1200
1005	Kiwi	3500
1007	Orange	1500
1008	Dragon Fruit	4000
1010	NULL	1000
1011	Grapes	2500
1012	PineApple	2800

13. Cross Join: It returns the cartesian product of the tables

```
SELECT * FROM fruits CROSS JOIN  orders where Fruit_name = 'Mango';
```

Order_id	Fruit_name	Quantity	Order_id	Price	Payment_mode
1002	Mango	30	1001	2000	Paytm
1002	Mango	30	1002	2500	Cash
1002	Mango	30	1003	3100	PhonePay
1002	Mango	30	1004	1200	Cash
1002	Mango	30	1005	3500	Paytm
1002	Mango	30	1007	1500	Cash
1002	Mango	30	1008	4000	PhonePay
1002	Mango	30	1010	1000	Cash
1002	Mango	30	1011	2500	Paytm
1002	Mango	30	1012	2800	Cash

14. Group By: It groups rows that have the same values into summary

```
SELECT count(*) as Total_Transaction, Payment_mode FROM orders group by Payment_mode;
```

Total_Transaction	Payment_mode
3	Paytm
5	Cash
2	PhonePay

15. Having: It is used to filter records that work on summarized group by result

```
SELECT COUNT(*), Payment_mode FROM orders group by Payment_mode having count(*) > 2;
```

COUNT(*)	Payment_mode
3	Paytm
5	Cash

16. Order By: Sort the result set in ascending or descending order

```
SELECT * FROM orders ORDER BY Price DESC;
```

	Order_id	Price	Payment_mode
▶	1008	4000	PhonePay
	1005	3500	Paytm
	1003	3100	PhonePay
	1012	2800	Cash
	1002	2500	Cash
	1011	2500	Paytm
	1001	2000	Paytm
	1007	1500	Cash
	1004	1200	Cash
	1010	1000	Cash

17. Distinct: Select only different values

```
SELECT DISTINCT Payment_mode from orders;
```

	Payment_mode
▶	Paytm
	Cash
	PhonePay

#18. Limit: Specifies the number of records to show

```
SELECT * FROM Fruits LIMIT 5;
```

	Order_id	Fruit_name	Quantity
▶	1001	Apple	20
	1002	Mango	30
	1004	Banana	12
	1005	Kiwi	30
	1007	Orange	27

19. Offset: Specifies the offset of the first row to return

```
SELECT * FROM Fruits LIMIT 6 OFFSET 2;
```

	Order_id	Fruit_name	Quantity
▶	1004	Banana	12
	1005	Kiwi	30
	1007	Orange	27
	1008	Dragon Fruit	10
	1009	Guava	15
	1011	Grapes	35

20. Union: Combines the result set of two or more select statement

```
SELECT student_name FROM student
UNION
SELECT stu_name FROM teacher;
```

student_name
Kash
Ash
Anil
Rohan
Dev
Ram
Shyam
Prashant

21. Union All: Combines the result set of two or more select statement including duplicates

```
SELECT student_name FROM student
UNION ALL
SELECT stu_name FROM teacher;
```

student_name
Kash
Ash
Anil
Rohan
Dev
Ram
Shyam
Prashant
Kash
Shyam
Rohan
Ash
Dev

22. In: Checks for values with in a set

```
SELECT * FROM orders where Payment_mode in ('Paytm', 'Cash');
```

Order_id	Price	Payment_mode
1001	2000	Paytm
1002	2500	Cash
1004	1200	Cash
1005	3500	Paytm
1007	1500	Cash
1010	1000	Cash
1011	2500	Paytm
1012	2800	Cash

23. Between: Selects values within a given range.

```
SELECT * FROM Fruits WHERE quantity BETWEEN 22 and 20;
```

Order_id	Fruit_name	Quantity
1001	Apple	20
1009	Guava	15
1012	PineApple	15

24. Like: Searches for a specified pattern in a columns

```
SELECT * FROM Fruits WHERE Fruit_name LIKE '%Apple';
```

Order_id	Fruit_name	Quantity
1001	Apple	20
1012	PineApple	15

25. Is Null: Tests for null values

```
SELECT * FROM employee WHERE age IS NULL;
```

26. Is Not Null: Tests for not null values

```
SELECT * FROM employee WHERE age IS NOT NULL;
```

emp_id	emp_name	age	salary
1001	Ravi	27	40000
1002	Anil	25	35000
1004	Bhumi	25	25000
1003	Kishan	26	40000
1005	Priya	23	20000

27. Case: Return value based on a condition

```
SELECT emp_name, age, CASE WHEN age >20 THEN 'ADULT' ELSE 'Younger' END
FROM employee;
```

emp_name	age	CASE WHEN age >20 THEN 'ADULT' ELSE 'Younger' END
Ravi	27	ADULT
Anil	20	Younger
Bhumi	20	Younger
Kishan	26	ADULT
Priya	23	ADULT

28. SubString: Extract character from string

```
SELECT SUBSTRING(emp_name,1, 3) FROM employee;
```

SUBSTRING(emp_name,1, 3)
Rav
Ani
Bhu
Kis
Pri

29. Length: Returns the length of the string

```
SELECT emp_name, LENGTH(emp_name) As length_of_name FROM employee;
```

emp_name	length_of_name
Ravi	4
Anil	4
Bhumi	5
Kishan	6
Priya	5

30. Trim: Remove spaces or specified characters from both ends of a string

```
SELECT TRIM(emp_name) as Remove_space FROM employee;
```

Remove_space
Ravi
Anil
Bhumi
Kishan
Priya

31. Upper: convert the string to uppercase

```
SELECT UPPER(emp_name) as First_name FROM employee;
```

First_name
RAVI
ANIL
BHUMI
KISHAN
PRIYA

32. LOWER: converts a string to lower

```
SELECT LOWER(emp_name) as First_name FROM employee;
```

First_name
ravi
anil
bhumi
kishan
priya

33. Replace: Replace occurrences of a specified string

```
SELECT REPLACE(emp_name, 'Anil', 'Anu') as emp_name FROM employee;
```

emp_name
Ravi
Anu
Bhumi
Kishan
Priya

34. Round: Round a number to a specified number of decimal places

```
SELECT ROUND(Price, 2) FROM orders;
```

35. Avg; Returns the average value of a numeric column

```
SELECT ROUND(AVG(Price),2) as Average_Price FROM orders;
```

Average_Price
2410.00

36. Count: Returns the number of rows that matches a specified column

```
SELECT Payment_mode, COUNT(*) as Total_rows FROM payments group by
Payment_mode;
```

	Payment_mode	Total_rows
▶	Cash	3
	Credit Card	4
	Debit Crad	2
	Paytm	3
	Netbanking	2

37. Sum: returns the total sum of a numeric column

```
SELECT SUM(Price) as Total_Price from orders where Payment_mode = 'Cash';
```

	Total_Price
▶	9000

38. Max: Return the maximum value in a set

```
SELECT MAX(Price) as Maximum_Salary FROM orders;
```

	Maximum_Salary
▶	4000

39. Min: Returns the minimum value in a set

```
SELECT MIN(Price) as Minimum_Price FROM orders;
```

	Minimum_Price
▶	1000

40. Get names of all students who scored more than class average.

```
SELECT Student_Name, Marks FROM student WHERE marks >
(SELECT AVG(marks) as Average from student) GROUP BY Student_Name, Marks;
```

	Student_Name	Marks
▶	Bhumika	93
	Dhruv	96
	Manish	92