

BASIC QUERY PART2

Continuing with the BASIC Query part 1 i.e. We have already created database, table and inserted the values in the table as shown in the below screenshot:

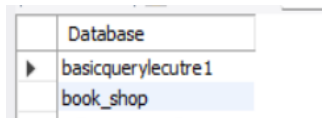


Figure 1-a

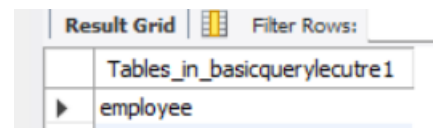
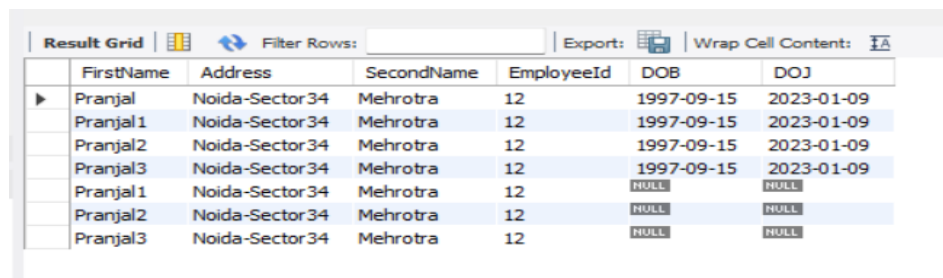


Figure 1-b



FirstName	Address	SecondName	EmployeeId	DOB	DOJ
Pranjal	Noida-Sector34	Mehrotra	12	1997-09-15	2023-01-09
Pranjal1	Noida-Sector34	Mehrotra	12	1997-09-15	2023-01-09
Pranjal2	Noida-Sector34	Mehrotra	12	1997-09-15	2023-01-09
Pranjal3	Noida-Sector34	Mehrotra	12	1997-09-15	2023-01-09
Pranjal1	Noida-Sector34	Mehrotra	12	NULL	NULL
Pranjal2	Noida-Sector34	Mehrotra	12	NULL	NULL
Pranjal3	Noida-Sector34	Mehrotra	12	NULL	NULL

Figure 1-c

As we can see,

Figure 1-a -----> represents database name(basicquylecture1)

Figure 1-b----->represents table(employee) in database(basicquylecture1)

Figure1-b ----->represents data in the table(employee)

INTEGRITY CONSTRAINTS IN SQL:

Integrity constraints are used so they keep the data consistent wrt to the data quality.

The major types of the integrity constraints used in MySQL are:

1.NOT NULL

2.UNIQUE

3.PRIMARY KEY

4.FOREIGN KEY

5.CHECK

6.CREATE INDEX

Creating a table in the same database applying the concept of constraints:

```
CREATE TABLE if not exists employee_table_with_constraints(  
    id int NOT NULL,  
    employeename varchar(50) NOT NULL,  
    salary double DEFAULT NULL,  
    hiring_date date DEFAULT '2022-01-14',  
    UNIQUE KEY (id),  
    CHECK (salary > 10000)  
);
```

```
• ○ CREATE TABLE if not exists employee_table_with_constraints(  
    id int NOT NULL,  
    employeename varchar(50) NOT NULL,  
    salary double DEFAULT NULL,  
    hiring_date date DEFAULT '2022-01-14',  
    UNIQUE KEY (id),  
    CHECK (salary > 10000)  
);  
show tables;
```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
id	employee_name	salary	hiring_date	
NULL	NULL	NULL	NULL	

Let's suppose we have created the above table with the constraints.

Case1: Inserting values that can lead to failure of integrity constraints.

Now here we are passing the id as null (as shown in the below screenshot) but in actual we have defined id cannot be null.

insert into

employee_table_with_constraints(id,employee_name,salary,hiring_date)values(
null,'Pranjal',12000,'2022-09-12');

```
show tables;
select * from employee_table_with_constraints;
insert into employee_table_with_constraints(id,employee_name,salary,hiring_date)values(null,'Pranjal',12000,'2022-09-12');
```

Output:

```
8 09:18:22 insert into employee_table_with_constraints(id,employee_name,salary,hiring_date)values(null,'Pranjal',12000,'2022-09-12'); Error Code: 1048. Column 'id' cannot be null
```

It is clearly giving the output as id column cannot have null values.

Case 2:

Passing the null value in employee_name column in the above table:


```
insert into employee_table_with_constraints(id,employee_name,salary,hiring_date)values(null,'Pranjal',12000,'2022-09-12');
insert into employee_table_with_constraints(id,employee_name,salary,hiring_date)values(3,null,12000,'2022-09-12');
```

Output:

Output				
Action Output				
#	Time	Action	Message	
1	10:11:40	insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(3,null,12000,'2022-09-12')	Error Code: 1048. Column 'employeeename' cannot be null	


Case 3:

Passing the value < 10000 in the salary column:



```
insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(null,'Pranjal',12000,'2022-09-12');
insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(3,null,12000,'2022-09-12');
insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(3,'Pranjal',500,'2022-09-12');
```

Output:



Output					Context Help	Snippets
Action Output						
#	Time	Action	Message			
1	10:11:40	insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(3,null,12000,'2022-09-12')	Error Code: 1048. Column 'employeeename' cannot be null			
2	10:15:05	insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(3,'Pranjal',500,'2022-09-12')	Error Code: 3819. Check constraint 'employee_table_with_constraints_chk_1' is violated.			

Now the output is like this:

Error Code: 3819. Check constraint 'employee_table_with_constraints_chk_1' is violated.

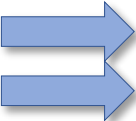
The problem with the above statement is that:

With the help of the above statement, the user is not able to know on which column name the constraint has failed.

In order to solve the above issue, we have to use the concept of ALIAS in SQL.

Case 4:


Now we want to check the integrity constraint of unique id



```
insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(3,'Pranjal',500,'2022-09-12');
insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(1,'Pranjal',15000,'2022-09-12');
insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(1,'Pranjal',15000,'2022-09-12');
```

Now we have inserted id as 1 in above table and again we are trying to insert the same value(1) in id .


Output:



```
3 10:18:04 insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(3,'Pranjal',500,'2022-09-12') Error Code: 3819. Check constraint 'employee_table_with_constraints_chk_1' is violated.
4 10:25:38 insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(1,'Pranjal',15000,'2022-09-12') 1 row(s) affected
5 10:25:50 insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(1,'Pranjal',15000,'2022-09-12') Error Code: 1062. Duplicate entry '1' for key 'employee_table_with_constraints.id'
```

Case 5: Now we want to check the integrity constraint of default which has been set in the hiring_date column


Note:



```
insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(1,'Pranjal',15000,'2022-09-12');
insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(1,'Pranjal',15000,'2022-09-12');
insert into employee_table_with_constraints(id,employeeename,salary,hiring_date)values(3,'Pranjal',15000,null);
select * from employee_table_with_constraints;
insert into employee_table_with_constraints(id,employeeename,salary)values(4,'Pranjal',15000);
```


If we have set default date in hiring_date column and we pass the query as shown above

i.e., passing the null value in hiring_date column it will take the null value not the default value as shown below:




Result Grid				
Filter Rows:				
	id	employee_name	salary	hiring_date
▶	1	Pranjal	15000	2022-09-12
	3	Pranjal	15000	NULL
	4	Pranjal	15000	2022-01-14
•	NULL	NULL	NULL	NULL

In order to check the default integrity constraint, we have to pass the value as shown below:



```
select * from employee_table_with_constraints;  
insert into employee_table_with_constraints(id,employee_name,salary)values(4,'Pranjal',15000);  
select * from employee_table_with_constraints;
```

Output:



Result Grid				
Filter Rows:				
	id	employee_name	salary	hiring_date
▶	1	Pranjal	15000	2022-09-12
	3	Pranjal	15000	NULL
	4	Pranjal	15000	2022-01-14
•	NULL	NULL	NULL	NULL

Here we have set default hiring date as 2022-01-14.

It has been inserted in the table column.
