

Assignment

March22/ DBT/002

Database Technologies

Diploma in Advance Computing (PG-DAC)

March 2022

Task 1.

1. Create new database temp2 and perform the following tasks.

```
mysql> create database temp2;
```

Query OK, 1 row affected (0.01 sec)

Task 2.

1. Create **COURSE** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
name	varchar(45)
duration	varchar(45)
summery	varchar(1024)

```
mysql> create table course(id int primary key, name varchar(45), duration varchar(45),  
summery varchar(1024));
```

Query OK, 0 rows affected (0.05 sec)

```
mysql> desc course;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int       | NO   | PRI | NULL    |      |
| name  | varchar(45) | YES  |     | NULL    |      |
| duration | varchar(45) | YES  |     | NULL    |      |
| summery | varchar(1024) | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
```

4 rows in set (0.01 sec)

2. Create **STUDENT** Relation with following columns.

Field Name	Datatype (size)
ID	Int primary key
namefirst	varchar(45)
namelast	varchar(45)
DOB	date
emailID	varchar(128)

```
mysql> create table student(id int primary key, firstname varchar(45),namelast varchar(45), DOB  
date, emailID varchar(128));
```

Query OK, 0 rows affected (0.04 sec)

```
mysql> desc student;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id    | int    | NO   | PRI | NULL    |      |
| firstname | varchar(45) | YES |    | NULL    |      |
| namelast | varchar(45) | YES |    | NULL    |      |
| DOB   | date   | YES |    | NULL    |      |
| emailID | varchar(128) | YES |    | NULL    |      |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

3. Create **STUDENT_PHONE** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
studentID	int foreign key(studentid) references student(id)
number	varchar(45)
isActive	bool

```
create table student_phone(id int primary key, studentID int, numbere varchar(45), isActive
bool, constraint FK_sid foreign key(studentID) references student(id));
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> desc student_phone;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id    | int    | NO   | PRI | NULL    |      |
| studentID | int    | YES | MUL | NULL    |      |
| numbere | varchar(45) | YES |    | NULL    |      |
| isActive | tinyint(1) | YES |    | NULL    |      |
+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec))
```

4. Create **STUDENT_ADDRESS** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
studentID	int unique not null foreign key(studentid) references student(id)
address	varchar(128)

```
create table student_address(id int primary key, studentID int, address varchar(128), foreign key
(studentid) references student(id));
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> desc student_address;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int    | NO   | PRI | NULL    |      |
| studentID | int    | YES  | MUL | NULL    |      |
| address | varchar(128) | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

5. Create **FACULTY** Relation with following columns.

Field Name	Datatype (size)
ID	Int primary key
namefirst	varchar(45)
namelast	varchar(45)
DOB	date
emailID	varchar(128)

```
create table faculty(id int primary key, namefirst varchar(45), namelast varchar(45), DOB date, emailID varchar(128));
```

Query OK, 0 rows affected (0.03 sec)

```
mysql> desc faculty;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int    | NO   | PRI | NULL    |      |
| namefirst | varchar(45) | YES  |     | NULL    |      |
| namelast | varchar(45) | YES  |     | NULL    |      |
| DOB    | date    | YES  |     | NULL    |      |
| emailID | varchar(128) | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

6. Create **FACULTY_PHONE** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
facultyID	int foreign key(facultyid) references faculty(id)
number	varchar(10)

```
mysql> create table faculty_phone(id int primary key, facultyID int, numbere varchar(45), foreign key(facultyID) references faculty(id));
```

Query OK, 0 rows affected (0.06 sec)

```
mysql> desc faculty_phone;
```

```
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    |      |
| facultyID | int  | YES  | MUL | NULL    |      |
| numbere | varchar(45) | YES  |      | NULL    |      |
+-----+-----+-----+-----+
```

3 rows in set (0.00 sec)

7. Create **FACULTY_ADDRESS** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
facultyID	int unique not null foreign key(facultyid) references faculty(id)
address	varchar(128)

```
create table faculty_address(id int primary key, facultyid int not null unique, address
varchar(128), foreign key(facultyid) references faculty(id));
```

Query OK, 0 rows affected (0.03 sec)

```
mysql> desc faculty_address;
```

```
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    |      |
| facultyid | int  | NO   | UNI | NULL    |      |
| address | varchar(128) | YES  |      | NULL    |      |
+-----+-----+-----+-----+
```

3 rows in set (0.01 sec)

8. Create **MODULES** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
name	varchar(128)
duration	int

```
mysql> create table modules(id int primary key, name varchar(128), duration int);
```

Query OK, 0 rows affected (0.02 sec)

```
mysql> desc modules;
```

```
+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| id    | int    | NO   | PRI | NULL    |       |
| name  | varchar(128) | YES |     | NULL    |       |
| duration | int    | YES  |     | NULL    |       |
+-----+-----+-----+-----+
```

3 rows in set (0.01 sec)

9. Create **COURSE_MODULES** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
courseID	int foreign key (courseid) references course(id)
moduleID	int foreign key (moduleid) references modules(id)

```
mysql> create table course_module(id int primary key, courseid int, moduleid int, foreign key (courseid) references course(id), foreign key(moduleid) references modules(id));
```

Query OK, 0 rows affected (0.03 sec)

```
mysql> desc course_module;
```

```
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    |       |
| courseid | int  | YES  | MUL | NULL    |       |
| moduleid | int  | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+
```

3 rows in set (0.01 sec)

10. Create **STUDENT_QUALIFICATIONS** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
studentID	int foreign key(studentid) references student(id)
name	varchar(128)
college	varchar(128)
university	varchar(128)
marks	varchar(45)
year	int

```
mysql> create table student_qualifications(id int primary key, studentid int, name
varchar(128),college varchar(128), university varchar(128), marks varchar(45), year int, foreign
key(studentid) references student(id));
```

Query OK, 0 rows affected (0.03 sec)

```
mysql> desc student_qualifications;
```

```
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    |      |
| studentid | int  | YES  | MUL | NULL    |      |
| name  | varchar(128) | YES  |      | NULL    |      |
| college | varchar(128) | YES  |      | NULL    |      |
| university | varchar(128) | YES  |      | NULL    |      |
| marks  | varchar(45) | YES  |      | NULL    |      |
| year  | int   | YES  |      | NULL    |      |
+-----+-----+-----+-----+
```

7 rows in set (0.01 sec)

11. Create **FACULTY_QUALIFICATIONS** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
facultyID	int foreign key (facultyid) references faculty(id)
name	varchar(128)

college	varchar(128)
university	varchar(128)
marks	varchar(45)
year	int

```
mysql> create table faculty_qualifications(id int primary key, facultyid int, name varchar(128),
college varchar(128), university varchar(128), marks varchar(45), year int, foreign key (facultyid)
references faculty(id));
```

Query OK, 0 rows affected (0.03 sec)

```
mysql> desc faculty_qualifications;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
facultyid	int	YES	MUL	NULL	
name	varchar(128)	YES		NULL	
college	varchar(128)	YES		NULL	
university	varchar(128)	YES		NULL	
marks	varchar(45)	YES		NULL	
year	int	YES		NULL	

7 rows in set (0.01 sec)

```
mysql>
```

12. Create **COURSE_BATCHES** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
name	varchar(45)
courseID	int foreign key (courseid) references course (id)
starton	date
endson	date
capacity	int

```
create table course_batches(id int primary key, name varchar(45), courseid int, starton date,
endson date, capacity int, foreign key (courseid) references course(id));
```

Query OK, 0 rows affected (0.03 sec)

```
mysql> desc course_batches;
```

```
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id    | int       | NO   | PRI | NULL    |       |
| name  | varchar(45) | YES  |     | NULL    |       |
| courseid | int       | YES  | MUL | NULL    |       |
| starton | date      | YES  |     | NULL    |       |
| endson  | date      | YES  |     | NULL    |       |
| capacity | int       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
```

6 rows in set (0.01 sec)

13. Create **BATCH_STUDENTS** Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
batchID	int foreign key (batchid) references course_batches (id)
studentID	int foreign key (studentid) references student (id)

```
create table batch_students(id int primary key, batchid int, studentid int, foreign key (batchid)
references course_batches(id), foreign key (studentid) references student(id));
```

Query OK, 0 rows affected (0.03 sec)

```
mysql> desc batch_students;
```

```
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    |       |
| batchid | int  | YES  | MUL | NULL    |       |
| studentid | int  | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+
```

3 rows in set (0.01 sec)

14. Create *STUDENT_CARDS* Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
studentID	int foreign key(studentid) references student(id)
name	varchar(45)
isActive	bool

create table student_cards(id int primary key, studentid int, name varchar(45), isactive bool, foreign key (studentid) references student(id));

Query OK, 0 rows affected (0.04 sec)

```
mysql> desc student_cards;
```

```
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    |      |
| studentid | int  | YES  | MUL | NULL    |      |
| name  | varchar(45) | YES  |     | NULL    |      |
| isactive | tinyint(1) | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+
```

4 rows in set (0.02 sec)

15. Create *STUDENT_ORDER* Relation with following columns.

Field Name	Datatype (size)
ID	int primary key
studentID	int foreign key(studentid) references student(id)
date	date
amount	int

mysql> create table student_order(id int primary key, studentid int, date date, amount int, foreign key(studentid) references student(id));

Query OK, 0 rows affected (0.04 sec)

```
mysql> desc student_order;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
studentid	int	YES	MUL	NULL	
date	date	YES		NULL	
amount	int	YES		NULL	

```
4 rows in set (0.01 sec)
```

Assignment

March22/ DBT/003

Database Technologies

Diploma in Advance Computing (PG-DAC)

March 2022

DML commands: Select data with WHERE clause.

USE *student_phone*, *student_address*, *faculty_phone*, *faculty_address*, *batch_students*, *course_batches*, *student_qualifications*, *faculty_qualifications*, *course_modules*, *modules*, *faculty*, *student*, *course*, *student_cards*, and *student_order* relation to solve the following queries.

1. List all courses.
<pre> select name from course; +-----+ name +-----+ PG-DAC DBDA Pre-DAC JAVA .NET DMC DSSD +-----+ 7 rows in set (0.00 sec) </pre>
2. List <i>namefirst</i> , <i>namelast</i> of all students.
<pre> select namefirst, namelast from student; +-----+-----+ namefirst namelast +-----+-----+ saleel bagde omkar rokde ulka joshi rahul patil ruhan bagde lala prasad sharmin bagde vrushali bagde vasant khande nitish patil neel save deep save nrupali save supriya karnik bandish karnik </pre>

```
| sangita | karnik |
| sangita | menon |
| rahul | shah |
| bhavin | patel |
| kaushal | patil |
| pankaj | gandhi |
| rajan | patel |
| bhavin | patel |
| mukesh | bhavsar |
| dilu | khande |
| sonam | khan |
| rohit | patil |
| raj | bubber |
```

```
+-----+-----+
```

28 rows in set (0.00 sec)

3. List *namefirst*, *namelast*, *DOB* and *emailID* of all students.

```
mysql> select namefirst, namelast, DOB, emailid from student;
```

```
+-----+-----+-----+-----+
```

```
| namefirst | namelast | DOB      | emailid      |
```

```
+-----+-----+-----+-----+
```

```
| saleel | bagde | 1984-06-12 | saleel.bagde@gmail.com |
| omkar | rokde | 1969-10-25 | omkar.rakde@gmail.com |
| ulka | joshi | 1970-10-25 | ulka.joshi@gmail.com |
| rahul | patil | 1982-10-31 | rahul.patil@gmail.com |
| ruhan | bagde | 1984-01-12 | ruhan.bagde@gmail.com |
| lala | prasad | 1980-12-01 | lala.prasad@gmail.com |
| sharmin | bagde | 1986-12-14 | sharmin.bagde@gmail.com |
| vrushali | bagde | 1984-12-29 | vrushali.bagde@gmail.com |
| vasant | khande | 1992-10-26 | vasant.khande@gmail.com |
| nitish | patil | 1990-10-26 | nitish.patil@gmail.com |
| neel | save | 1975-10-30 | neel.save@gmail.com |
| deep | save | 1986-11-30 | deep.save@gmail.com |
| nrupali | save | 1981-12-01 | nrupali.save@gmail.com |
| supriya | karnik | 1983-12-15 | supriya.karnik@gmail.com |
| bandish | karnik | 1987-12-30 | bandish.karnik@gmail.com |
| sangita | karnik | 1990-12-01 | sangita.karnik@gmail.com |
| sangita | menon | 1989-10-26 | sangita.menon@gmail.com |
| rahul | shah | 1982-06-12 | rahul.shah@gmail.com |
| bhavin | patel | 1983-11-13 | bhavin.patel@gmail.com |
| kaushal | patil | 1982-07-30 | kaushal.patil@gmail.com |
| pankaj | gandhi | 1982-07-30 | pankaj.gandhi@gmail.com |
| rajan | patel | 1982-07-30 | rajan.patel@gmail.com |
| bhavin | patel | 1982-07-30 | bhavin.patel@gmail.com |
| mukesh | bhavsar | 1982-07-30 | mukesh.bhavsar@gmail.com |
| dilu | khande | 1982-07-30 | dilu.khande@gmail.com |
| sonam | khan | 1972-05-13 | sonam.khan@gmail.com |
| rohit | patil | 1976-12-31 | rohit.patil@gmail.com |
| raj | bubber | 1982-02-28 | raj.bubber@gmail.com |
```

```
+-----+-----+-----+-----+
```

28 rows in set (0.00 sec)

4. Display student information of the *ID* is 15.

mysql> select * from student where id=15;

```
+-----+-----+-----+-----+-----+
| ID | namefirst | namelast | DOB      | emailID          |
+-----+-----+-----+-----+-----+
| 15 | bandish  | karnik   | 1987-12-30 | bandish.karnik@gmail.com |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

5. List *namefirst*, *namelast*, and *emailID* of student whose *student namefirst* is 'Nitish'.

mysql> select namefirst, namelast, emailid from student where namefirst = 'nitish';

```
+-----+-----+-----+
| namefirst | namelast | emailid          |
+-----+-----+-----+
| nitish    | patil    | nitish.patil@gmail.com |
+-----+-----+-----+
1 row in set (0.00 sec)
```

6. List all students having *ID* greater than equal to 12.

mysql> select * from student where id>=12;

```
+-----+-----+-----+-----+-----+
| ID | namefirst | namelast | DOB      | emailID          |
+-----+-----+-----+-----+-----+
| 12 | deep      | save     | 1986-11-30 | deep.save@gmail.com |
| 13 | nrupali   | save     | 1981-12-01 | nrupali.save@gmail.com |
| 14 | supriya   | karnik    | 1983-12-15 | supriya.karnik@gmail.com |
| 15 | bandish   | karnik    | 1987-12-30 | bandish.karnik@gmail.com |
| 16 | sangita   | karnik    | 1990-12-01 | sangita.karnik@gmail.com |
| 17 | sangita   | menon     | 1989-10-26 | sangita.menon@gmail.com |
| 18 | rahul     | shah      | 1982-06-12 | rahul.shah@gmail.com |
| 19 | bhavin    | patel     | 1983-11-13 | bhavin.patel@gmail.com |
| 20 | kaushal   | patil     | 1982-07-30 | kaushal.patil@gmail.com |
| 21 | pankaj    | gandhi    | 1982-07-30 | pankaj.gandhi@gmail.com |
| 22 | rajan     | patel     | 1982-07-30 | rajan.patel@gmail.com |
| 23 | bhavin    | patel     | 1982-07-30 | bhavin.patel@gmail.com |
| 24 | mukesh    | bhavsar   | 1982-07-30 | mukesh.bhavsar@gmail.com |
| 25 | dilu      | khande    | 1982-07-30 | dilu.khande@gmail.com |
| 26 | sonam     | khan      | 1972-05-13 | sonam.khan@gmail.com |
| 27 | rohit     | patil     | 1976-12-31 | rohit.patil@gmail.com |
| 28 | raj       | bubber    | 1982-02-28 | raj.bubber@gmail.com |
+-----+-----+-----+-----+-----+
17 rows in set (0.00 sec)
```

7. List all student details whose DOB is '1980-12-01'

```
mysql> select * from student where DOB = '1980-12-01';
```

```
+-----+-----+-----+-----+-----+
| ID | namefirst | namelast | DOB      | emailID      |
+-----+-----+-----+-----+-----+
| 6 | lala      | prasad   | 1980-12-01 | lala.prasad@gmail.com |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

8. Display the phone details where student ID is 5;

```
mysql> select * from student_phone where studentid = 5;
```

```
+-----+-----+-----+-----+
| ID | studentID | number    | isActive |
+-----+-----+-----+-----+
| 5 | 5 | 7032300001 | 1 |
| 25 | 5 | 7032300001 | 1 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

9. List student address whose student ID is 10.

```
mysql> select address from student_address where studentid = 10;
```

```
+-----+
| address |
+-----+
| 7710 Covington Rd , New State Road, New York, 37188 |
+-----+
1 row in set (0.00 sec)
```

10. List all faculty details.

```
select * from faculty;
```

```
+-----+-----+-----+-----+-----+
| ID | namefirst | namelast | DOB      | emailID      |
+-----+-----+-----+-----+-----+
| 1 | prachi    | gupta    | 1974-06-12 | prachi.gupta@gmail.com |
| 2 | ketan     | shukla   | 1972-10-25 | ketan.shukla@gmail.com |
| 3 | kiran     | dev      | 1971-10-25 | kiran.dev@gmail.com |
| 4 | parag     | patil    | 1972-10-31 | parag.patil@gmail.com |
+-----+-----+-----+-----+-----+
4 rows in set (0.03 sec)
```

11. List all phone number whose faculty ID is 2.

```
mysql> select number from faculty_phone where id = 2;
```

```
+-----+
| number |
+-----+
| 7032300039 |
+-----+
1 row in set (0.00 sec)
```

12. List all phone number whose student ID is 13.

```
mysql> select number from student_phone where studentid = 13;
```

```
+-----+
| number |
+-----+
| 7032300055 |
| 7132300055 |
| 7132300055 |
| 7132300055 |
+-----+
4 rows in set (0.00 sec)
```

13. List all modules.

```
mysql> select name from modules;
```

```
+-----+
| name |
+-----+
| Oracle |
| PHP |
| MySQL |
| Node |
| C++ |
| C |
| JAVA1 |
| JAVA2 |
| MongoDB |
| NET |
| Hive |
| Python |
| Aptitude |
| OOPs with C++ Programming |
| Data Structures |
| OS Concepts |
| iOS Programming |
+-----+
17 rows in set (0.00 sec)
```

14. List thecourse_modules whose courseID is 1.

```
mysql> select * from course_modules where id = 1;
```

```
+---+-----+-----+
| ID | courseID | moduleID |
+---+-----+-----+
| 1 | 1 | 1 |
+---+-----+-----+
1 row in set (0.00 sec)
```

15. Display all course_batches who's sitting capacity is 80.

```
mysql> select name from course_batches where capacity = 80;
```

```
+-----+
| name |
+-----+
```

Batch1
Batch6
Batch11
Batch16
Batch21
+-----+
5 rows in set (0.00 sec)

Assignment

Sept22/ DBT/004

Database Technologies

Diploma in Advance Computing (e-DAC)

September 2022

DML commands: Select data with WHERE, LIMIT, and ORDER BY clause.

USE *student_phone*, *student_address*, *faculty_phone*, *faculty_address*, *batch_students*, *course_batches*, *student_qualifications*, *faculty_qualifications*, *course_modules*, *modules*, *faculty*, *student*, *course*, *student_cards*, and *student_order* relation to solve the following queries.

1. List all student.
mysql> select * from student;
2. List <i>namefirst</i> , <i>namelast</i> of all students in ascending order of <i>namefirst</i> .
mysql> select namefirst, namelast from student order by namefirst;
3. List <i>namefirst</i> , <i>namelast</i> , <i>DOB</i> , and <i>emailID</i> for the first 5 students.
mysql> select namefirst, namelast, dob, emailid from student limit 5;
4. Display student information of the <i>studentID</i> is either 1, 2, 5 or 7.
mysql> select * from student where id in(1,2,5,7);
5. List <i>namefirst</i> , <i>namelast</i> , and <i>emailID</i> of student whose <i>studentID</i> is not 5, 10, 15, display first 7 rows only.
mysql> select namefirst, namelast, emailid from student where id not in(5,10,15) limit 7;
6. List first two faculty details only.
mysql> select * from faculty limit 2;
7. List all <i>student_phone</i> number in ascending order of phone number.
mysql> select * from student_phone order by number;
8. Display the <i>student_address</i> whose <i>studentID</i> is either 2, 4, 6 or 10 in descending order of <i>studentID</i> .

mysql> select * from student_address where studentid in (2, 4, 6, 10) order by studentid desc;
9. List all modules in ascending order of module names.
select * from modules order by name;
10. List first 10modulesafter arranging the module name in descending order.
mysql> select * from modules order by name desc;
11. List all student_qualification whose college is 'New York'.
mysql> select * from student_qualifications where college = "New York";
12. List all student_qualification whose have done "BE" from "Florida" college.
mysql> select * from student_qualifications where college ="florida" and name = "be";
13. List all student_qualifications whose passed the college in the year 2012 and have scored more than 67% marks.
mysql> select * from student_qualifications where marks>67 and year=2012;
14. List the qualification details for the faculty number 1, and 3.
mysql> select * from faculty_qualifications where id in (1,3);
15. Display the name, college, and university from the student_qualification who have passed in the year 2018.
mysql> select name, college, university from student_qualifications where year=2018;

Assignment

Sept22/ DBT/006

Database Technologies

Diploma in Advance Computing (e-DAC)

September 2022

String, Date, Math functions, and Date formats.

USE *student_phone*, *student_address*, *faculty_phone*, *faculty_address*, *batch_students*, *course_batches*, *student_qualifications*, *faculty_qualifications*, *course_modules*, *modules*, *faculty*, *student*, *course*, *student_cards*, and *student_order* relation to solve the following queries.

1. Get student <i>namefirst</i> with how many characters are there in their <i>namefirst</i> .
mysql> select namefirst, length(namefirst) characters_present from student ;
2. Get student details whose <i>namefirst</i> is having 4 characters only.
mysql> select namefirst from student where namefirst = length(namefirst)-4;
OR
mysql> select namefirst from student where namefirst = substr(namefirst,1,4);
3. Get the ASCII value of the 3 rd character of <i>namefirst</i> column.
mysql> select namefirst, ascii(substr(namefirst,3,1)) from student;
4. Get <i>namefirst</i> and <i>namelast</i> in lowercase.
mysql> select namefirst, lcase(namefirst), namelast, lcase(namelast) from student;
5. Get (<i>namefirst</i> , <i>namelast</i> , and <i>emailID</i>) all 7 letter <i>emailID</i> .
mysql> select namefirst, namelast, emailid, left(emailid,7) from student;
6. Get (<i>namefirst</i> , <i>namelast</i> and first 3 letters of <i>namefirst</i>) for all students.
mysql> select namefirst,namelast, substr(namefirst,1,3) from student;
7. Get(<i>namefirst</i> , <i>namelast</i> and last 3 letters of <i>namefirst</i>) for all student.
mysql> select namefirst, namelast, right(namefirst,3) from student;
8. Get all student (<i>phonenumber</i>) whose <i>phonenumber</i> starts with 70.
mysql> select * from student_phone where left(number,2)=70;
9. Get student details of first 5 students.
mysql> select *from student limit 5;
10. Get student details of last 5 students.

mysql> select * from student order by id desc limit 5;
11. Get student details in ascending order of <i>namefirst</i> .
mysql> select * from student order by namefirst;
12. Get student details in descending order of <i>namelast</i> .
mysql> select * from student order by namelast desc;
13. Get (<i>student id, namefirst, namelast, dob, and emailID</i>) for all students whose length of email id is more than 20 characters.
mysql> select namefirst, namelast, emailid, from student where length(emailid)>20;
14. Combine to display student <i>namefirst</i> and <i>namelast</i> .
mysql> select namefirst, namelast, concat(namefirst,' ',namelast) fullname from student;
15. Write a query to display the following output for all student. If (<i>namefirst, namelast or emailID</i>) is null then replace it with a blank space. eg. (Bhoopali Nanadikar and emailID is bhoopali.nanadikar@gmail.com)
mysql> select concat(namefirst,' ',namelast, 'and emailid is ', emailid), if(namefirst=null,' ',namefirst), if(namelast=null,' ',namelast), if(emailid=null,' ',emailid)from student; OR mysql> select concat(ifnull(namefirst,' '), ' ',ifnull(namelast,' '), 'and emailID is ',ifnull(emailid,' ')) from student;
16. Get student <i>namefirst</i> and <i>namelast</i> in upper case.
mysql> select ucase(namefirst) firstname,ucase(namelast) lastname from student;
17. Get student <i>firstname</i> and <i>lastname</i> in lower case.
mysql> select lcase(namefirst) firstname,lcase(namelast) lastname from student;
18. Get student <i>firstname</i> and <i>lastname</i> in reverse order.
mysql> select reverse(namefirst) firstname, reverse(namelast) lastname from student;
19. Get first 4 letters of student <i>namefirst</i> .
mysql> select namefirst, left(namefirst,4) from student;
20. Get second letter of student <i>namefirst</i> to second last letter of student <i>namefirst</i> .
mysql> select namefirst,substr(namefirst,2,(length(namefirst)-2)) from student;
21. Get ASCII character of student <i>namefirst</i> .
mysql> select namefirst, ascii(namefirst) from student;

22. Get first 5 letter of the students' <i>namefirst</i> .
mysql> select namefirst, left(namefirst,5) from student;
23. Print <i>phone number</i> of all student in the given format 7032300034****.
mysql> select concat(number,'****')from student_phone;
24. Get all student whose DOB is in the month of 'October'.
mysql> select * from student where month(dob) =10 ;
25. Get all student whose DOB is in the month of 'January' or 'December'.
mysql> select * from student where month(dob)= "1" or month(dob)="12" ;
26. Get all faculty who were born on 'Sunday'
mysql> select * from faculty where dayofweek(DOB)=1;
27. Print current date and time.
mysql> select now();
28. Extract month from the current date.
mysql> select extract(month from now());
29. Extract year from the current date.
mysql> select extract(year from now());
30. Get all student whose DOB is in the year 1984 in ascending order of <i>namefirst</i> .
mysql> select * from student where extract(year from dob)=1984 order by namefirst;
31. Get all student whose DOB is in the 4 quarter of a year.
mysql> select * from student where extract(quarter from dob)=4 ;
32. Get all student whose DOB is in the 43 rd week of a year.
mysql> select * from student where extract(week from dob)=43 ;
33. Get all student whose DOB is in between 10 and 19 day.
mysql> select * from student where (day(dob)=10 and day(dob)=19);
34. Generate the random number between 1 to 100
mysql> select round(rand()*100);
35. Display the 5 character of namefirst column from student table.

mysql> select namefirst, left(namefirst,5) from student;
36. Display all student in ascending order of their DOB, the ordering must be done on weekday name starting form 'Monday', 'Tuesday'
mysql> select * from student order by DOB,weekday(dob);
37. Display all student who's DOB comes in the 4 th quarter of the year.
mysql> select * from student where extract(quarter from dob)=4 ;
38. Display all student who were born on 'Sunday'.
mysql> select * from student where (dayofweek(dob))=1;
39. Display the DOB in the give format '12 th of June 1984'
mysql> select date_format(DOB,'%D of %b %Y') date from student;
40. Display all course_batches who ends on 'Sunday'.
mysql> select * from course_batches where (dayofweek(endson))=1;
41. Display student_phone number in the following format "7032*****" for all students.
mysql> select concat(left(number,4),'****')from student_phone;
42. Display student_phone number in the following format "7032****8765" for all students.
mysql> select concat(left(number,4),'****',right(number,4))from student_phone;
43. Display nameFirst and count how many 'A' char in appearing in their names.
mysql> select namefirst,count(namefirst) from student where namefirst like '%e%' group by namefirst;

Assignment

Sept22/ DBT/007
 Database Technologies
 Diploma in Advance Computing (e-DAC)
 September 2022

DML commands: Select data with WHERE, GROUP BY, HAVING, ORDER BY and LIMIT clause.

USE *student_phone, student_address, faculty_phone, faculty_address, batch_students, course_batches, student_qualifications, faculty_qualifications, course_modules, modules, faculty, student, course, student_cards, and student_order relation* to solve the following queries.

1. List all student.
mysql> select * from student;
2. List namefirst, namelast of all student.
mysql> select namefirst, namelast from student;
3. Display student information of the student whose student ID is 10.
mysql> select * from student where id = 10;
4. List of various faculties available from faculty table.
mysql> select * from faculty;
5. List all student having 'A' as second letter in their namefirst.
mysql> select * from student where substring(namefirst,2,1)='r';
6. List all student having letter 'A' in their namefirst.
mysql> select * from student where namefirst like '%a%';
7. Display the details of the student whose DoB is '1986-12-14'.
mysql> select * from student where extract(year from dob) = 1986 and extract(month from dob) = 12 and extract(day from dob) = 14;
8. List all student having 'R' as first letter in their namefirst.

mysql> select * from student where left(namefirst,1)='R';
9. Display the <i>namefirst</i> , <i>lastName</i> from student relation with Customized column headings.
mysql> select namefirst FIRST_NAME, namelast LAST_NAME from student;
10. Display all students in ascending order of their DOB.
mysql> select * from student order by dob;
11. Display two records of student whose name starts with the letter 'S'.
mysql> select * from student where left(namefirst,1)='s' limit 2;
12. Display the student detail whose DOB is '1986-12-14'.
mysql> select * from student where extract(year from dob) = 1986 and extract(month from dob) = 12 and extract(day from dob) = 14;
13. Display all modules whose module duration is 1 (use modules table).
mysql> select * from modules where duration = 1;
14. Display all batches whose sitting capacity is 80 students (use course_batches table).
mysql> select * from course_batches where capacity=80;
15. Display all student qualification who have done 'BE' and secured marks more than 70. (use student_qualifications table).
mysql> select * from student_qualifications where name='be' and marks>=70;
16. Display all student qualification who have done 'BE' and graduated in the year 2017. (use student_qualifications table).
mysql> select * from student_qualifications where name='be' and year=2017;
17. Display all student qualification who have done 'BE' and graduated in the year 2017 and scored marks more than 80. (use student_qualifications table).
mysql> select * from student_qualifications where name='be' and marks>=80 and year=2017;
18. Display faculty qualification who have done 'BE' from 'Harvard University'(use faculty_qualifications table)

mysql> select * from faculty_qualifications where name='be' and university="Harvard University";
19. Display all courses whose course duration is 6 months.(use course table)
mysql> select * from course where duration=6;
20. Display module details whose module duration is between 1 and 2, arrange the data in ascending order of module duration. (use module table)
mysql> select * from modules where duration between 1 and 2 order by duration;
21. Display all student with their voting rights, if the student is below 1980 then print the message “*The student can vote” else print “The student cannot vote”.
mysql> select namefirst, namelast, if((year(dob)<1980),"The student can vote","The student cannot vote") from student;
22. Display all distinct universities from student_qualifications table.
mysql> SELECT count(distinct university) FROM student_qualifications;
23. Display the second highest marks scored by any student in 'BE'.
mysql> Select distinct(marks) from student_qualifications where name = 'be' order by marks desc limit 1,1;
24. Display the second lowest marks scored by any student in 'BE'.
mysql> Select distinct(marks) from student_qualifications where name = 'be' order by marks limit 1,1;
25. Display last 7 student.
mysql> select * from student order by id desc limit 7;

Assignment

Sept22/ DBT/008
Database Technologies
Diploma in Advance Computing (e-DAC)
September 2022

Aggregate Functions.

USE *student_phone*, *student_address*, *faculty_phone*, *faculty_address*, *batch_students*, *course_batches*, *student_qualifications*, *faculty_qualifications*, *course_modules*, *modules*, *faculty*, *student*, *course*, *student_cards*, and *student_order* relation to solve the following queries.

1. Count total number of students.
<code>mysql> select count(*) from student;</code>
2. Count total number of students who are born in 1986.
<code>mysql> select count(*) from student where year(dob)=1986;</code>
3. Count total number of students whose namefirst starts with the letter 'B'.
<code>mysql> select count(*) from student where left(namefirst,1)='b';</code>
4. Count total number student who were born in 'July.
<code>mysql> select count(*) from student where month(dob)="7";</code>
5. Display studentID and count the student who are having more than two phones.
<code>mysql> select studentid, count(*)'number' from student_phone group by studentID having number>2;</code>
6. Count unique universities from student_qualifications table.
<code>mysql> select count(distinct university) from student_qualifications;</code>
7. Display the university name and the count of those students who have done 'BE'
<code>mysql> select university ,count(*) from student_qualifications where name="be" group by university;</code>
8. Count how many students has done 'BE'.
<code>mysql> select count(*) from student_qualifications where name= "BE";</code>
9. Count how many students has not done 'BE'.
<code>mysql> select count(*) from student_qualifications where name!="be";</code>
10. Find the maximum marks student got in 'BE'.
<code>mysql> select max(marks) from student_qualifications where name = "BE";</code>

11. Find the minimum marks student got in 'BE'.
<code>mysql> select min(marks) from student_qualifications where name = "BE";</code>
12. Count how many course_batches have started on '2016-02-01'.
<code>mysql> select count(*) from course_batches where starton='2016-02-01';</code>
13. Count the number of students who have more than 60% in 'BE'.
<code>mysql> select count(*) from student_qualifications where name = "be" and marks>60;</code>
14. Count the number of students who have more than 60% in 'BE' and done from 'Harvard university'.
<code>mysql> select count(*) from student_qualifications where marks>60 and name='be' and university='harvard university';</code>
15. Count number of courses.
<code>mysql> select distinct(count(name)) from course;</code>
16. Count how many distinct universities from student_qualifications table.
<code>mysql> SELECT university, count(distinct university) FROM student_qualifications group by university;</code>
17. Find the maximum marks any student has got in "BE".
<code>mysql> select max(marks) from student_qualifications where name='be';</code>

Assignment

Sept22/ DBT/009
 Database Technologies
 Diploma in Advance Computing (PG-DAC)
 September 2022

Joins

USE *student_phone*, *student_address*, *faculty_phone*, *faculty_address*, *batch_students*, *course_batches*, *student_qualifications*, *faculty_qualifications*, *course_modules*, *modules*, *faculty*, *student*, *course*, *student_cards*, and *student_order* relation to solve the following queries.

1. Display all student and with their address from student and student_address tables.
<code>mysql> select student.*, address from student, student_address where student.id = student_address.studentid;</code>
2. Display (namefirst, namelast, emailID, and student_qualification details) from student and student_qualification relations.
<code>mysql> select namefirst,namelast, emailid, student_qualifications.* from student, student_qualifications where student.id = student_qualifications.studentid;</code>
3. Display (namefirst, namelast, emailID, college, and university) who have studied in 'Yale University'. (Use student, and student_qualification relation)
<code>mysql> select namefirst,namelast, emailid, college, university from student inner join student_qualifications on student.id = student_qualifications.studentid and university = "Yale University";</code>
4. Display all student details his phone details and his qualification details. (Use student, student_phone and student_qualification relation)
<code>mysql> select * from student s join student_phone sp join student_qualifications sq on s.id=sp.studentid and s.id=sq.studentid;</code>
5. Display (studentID, namefirst, namelast, name, college, university, and marks) whose name is 'BE'. (Use student, and student_qualification relation)
<code>mysql> select studentID, namefirst, namelast, name, college, university, marks from student s inner join student_qualifications sq on s.id=sq.studentid and name="be";</code>
6. Display the module name and the duration of the module for the batch "Batch1".
<code>mysql> select m.name, duration, cb.name from modules m join course_batches cb join course_modules cm on m.id=cm.id and cm.courseid=cb.courseid and cb.name = 'batch1';</code>
7. Display student information along with his batch details who have joined in "Batch1".
<code>mysql> select * from student s join course_batches cb join batch_students bs on s.id=bs.studentid and bs.batchid=cb.courseid and cb.name='batch1';</code>

8. Display module names for “PG-DAC” course.
<code>mysql> select * from modules m join course c join course_modules cm on cm.courseID=m.id and cm.moduleID=c.id and c.name="pg-dac";</code>
9. Display <i>namefirst, namelast, and batch name</i> for all students.
<code>mysql> select namefirst, namelast, name from student s, course_batches cb, batch_students bs where s.id=bs.studentid and cb.id=bs.batchid;</code>
10. Display (<i>namefirst, namelast, phone number, and emailid</i>) whose student ID is 13.
<code>mysql> select namefirst, namelast, number, emailid from student s, student_phone sp where s.id=sp.studentid and s.id =13;</code>
11. Display (<i>namefirst and count the total number of phones a student is having</i>) for all student.
<code>mysql> select namefirst, count(number) from student s, student_phone sp where s.id=sp.studentid group by studentid;</code>
12. Get student's (<i>namefirst, namelast, DOB, address, name, college, university, marks, and year</i>).
<code>mysql> select namefirst, namelast, dob, address, name, college, university, marks, year from student s inner join student_address sa inner join student_qualifications sq on s.id=sa.studentid and s.id=sq.studentid;</code>
13. Get (<i>namefirst, namelast, emailID, phone number, and address</i>) whose faculty name is 'ketan'.
<code>mysql> select namefirst, namelast, emailid, number, address from faculty f join faculty_address fa join faculty_phone fp on f.id=fa.facultyid and f.id=fp.facultyid and namefirst='ketan';</code>
14. Get(<i>course name and batch name</i>)for all courses.
<code>mysql> select c.name, cb.name from course c inner join course_batches cb on c.id=cb.courseid;</code>
15. Get all student details who have taken admission in 'PG-DAC' course.
<code>mysql> select s.* , c.name from student s join batch_students bs join course_batches cb join course c on s.id=bs.studentid and cb.id=bs.batchid and c.id=cb.courseid and c.name='PG-DAC';</code>
16. Get all course details which had started on '2016-02-01'.
<code>mysql> select c.* from course c join course_batches cb on c.id=cb.courseid and starton="2016-02-01";</code>
17. Get all course name and module names which are taught in 'PG-DAC' course.
<code>mysql> select c.name, m.name from course c join modules m join course_modules cm on c.id=cm.courseid and m.id=cm.moduleid and c.name = "PG-DAC";</code>
18. Display how many modules are taught in each course.
<code>mysql> select c.name,count(m.name) from course c join modules m join course_modules cm on c.id=cm.courseid and m.id=cm.moduleid group by c.name;</code>
19. Display the student detail who are 'BE' graduate.

mysql> select s.*, name from student s inner join student_qualifications sq on s.id=sq.studentid and name="be";	
20. Display all distinct course detail, where module for every course is designed.	
mysql> select modules.name, course.name from modules inner join course_modules on modules.ID = course_modules.moduleID inner join course on course.ID = course_modules.courseID;	
21. Display studentID who have more than 2 phone numbers.	
mysql> select s.id, count(number) from student s, student_phone sp where s.id=sp.studentid group by sp.studentid having count(number)>2;	
22. Display the courses where 'JAVA1' is taught.	
mysql> select c.name, m.name from course c join modules m join course_modules cm on c.id=cm.courseid and cm.moduleid=m.id and m.name='java1';	
23. Display all student who have taken admission in 6 months course.	
mysql> select s.id, s.namefirst, c.name, c.duration from student s inner join course c inner join batch_students bs inner join course_batches cb on c.id=cb.courseid and bs.studentid=s.id and cb.id=bs.batchid and c.duration=6;	
24. Write a query to display the output in the following manner. 'saleel', 'Aadhaar, Driving Licence, PAN, Voter ID, Passport, Debit, Credit' Arrange the data is ascending order of <i>nameFirst</i> .	
mysql> select s.namefirst, group_concat(sc.name) from student s inner join student_cards sc on s.id=sc.studentid group by sc.studentid order by s.namefirst;	
25. Write a query to display the output in the following manner. 'ruhan', 'DBDA, PG-DAC, Pre-DAC'	
mysql> select s.namefirst, group_concat(c.name) from course c inner join course_batches cb inner join batch_students bs inner join student s on bs.studentid = s.id and c.id=cb.courseid and cb.id =bs.batchid group by s.namefirst;	

Assignment

March22/ DBT/011
Database Technologies
Diploma in Advance Computing (PG-DAC)
March 2022

Sub-queries with joins.

USE *student_phone*, *student_address*, *faculty_phone*, *faculty_address*, *batch_students*, *course_batches*, *student_qualifications*, *faculty_qualifications*, *course_modules*, *modules*, *faculty*, *student*, *course*, *student_cards*, and *student_order* relation to solve the following queries.

1. Display all student who have taken admission in more than 2 batches.
<code>select s.* from student s inner join batch_students bs on s.Id = bs.studentID and bs.studentID in (select studentid from batch_students bs where (select count(studentID) from batch_students) > 2) group by studentID;</code>
2. Display the student detail that have joined the same batch of the student 'saleel'.
<code>select s.* from student s, batch_students bs where s.ID = bs.studentID and batchID in (select batchID from batch_students bs, student where s.ID = bs.studentID and namefirst = 'saleel') group by s.ID;</code>
3. Display all courses where least number of students have taken the admission.
<code>select c.* from course c, batch_students bs, course_batches cb where cb.ID = bs.batchID and c.ID = cb.courseID and bs.studentID in (select count(studentID) from batch_students group by studentID having count(studentID) = min(studentID));</code>
4. Display student details who have not taken the admission.
<code>select s.* from student s where not exists (select bs.* from batch_students bs where s.ID = bs.studentID);</code>
5. Get all courses where no modules are defined in course_modules table.
<code>select c.* from course c where not exists (select m.* from modules m, course_modules cm where m.ID = cm.moduleID);</code>
6. Display course_batches details where student has taken the admission.
<code>select cb.* from course_batches cb where exists (select bs.* from batch_students bs where cb.ID = bs.batchID);</code>
7. Display all students whose marks of 'BE' is more than 'ULKA' marks in 'BE'.
<code>select s.* from student s, student_qualifications sq where s.ID = sq.studentID and marks > (select marks from student_qualifications sq, student s where s.ID = sq.studentID and namefirst = 'ulka' and name = 'BE') and name = 'BE';</code>
8. Display all students whose marks are more than 'saleel' marks in 10 th std.

```
select s.* from student s, student_qualifications sq where s.ID = sq.studentID and marks > (select marks from student_qualifications sq, student s where s.ID = sq.studentID and namefirst = 'saleel' and name = '10') and name = '10';
```

9. Display students whose DOB is as same as 'kaushal'

```
select * from student where DOB = (select DOB from student where namefirst = 'kaushal');
```

10. Display all student details who have three or more phone numbers.

```
select * from student where id = (select studentID from student_phone group by studentID having count(number) > 3);
```

11. Display marks for the studentID 1 and 7 who have done 'BE'. (Note: the marks must be displayed side by side).

```
+-----+-----+
| StudentId 1 | StudentId 7 |
+-----+-----+
| 68          | 97          |
+-----+-----+
1 row in set (0.00 sec)
```

```
select (select marks from student_qualifications where studentID = 1 and name = 'BE') "studentID 1",
(select marks from student_qualifications where studentID = 7 and name = 'BE') "studentID 7";
```

12. Display marks for the studentID 1 and 7 who have done 'BE' also fine out the difference of marks between them.

(Note: the marks and difference between the marks must be displayed side by side)

```
+-----+-----+-----+
| StudentId 1 | StudentId 7 | Marks Difference |
+-----+-----+-----+
| 68          | 97          | 29               |
+-----+-----+-----+
1 row in set (0.00 sec)
```

```
select (select marks from student_qualifications where studentID = 1 and name = 'BE') "studentID 1",
(select marks from student_qualifications where studentID = 7 and name = 'BE') "studentID 7",
abs((select marks from student_qualifications where studentID = 1 and name = 'BE') - (select marks from student_qualifications where studentID = 7 and name = 'BE')) "Marks Difference";
```

13. Display all student who are not joined any of the batch.

```
select * from student where not exists (select * from batch_students where student.ID = batch_students.studentID);
```

14. Display all course_batches details who are starting on the same day as 'Batch1'.

```
select * from course_batches where starton = (select starton from course_batches where name = 'Batch1');
```

15. Display all students whose 10th marks is more than student 'Neel's 10th marks.

select * from student s, student_qualifications sq where s.ID = sq.studentID and marks > (select marks from student_qualifications sq, student where s.ID = sq.studentID and namefirst = 'neel' and name = '10') and name = '10';
16. Get all student with their qualification details who have highest marks in 'BE'.
select s.namefirst, sq.* from student s, student_qualifications sq where s.ID = sq.studentID and
17. Get all student with their qualification details who have second highest marks in 'BE'.
select * from student s, student_qualifications sq where s.ID = sq.studentID and marks = (select max(marks) from student_qualifications where marks < (select max(marks) from student_qualifications where name = 'BE') and name = 'BE') and name = 'BE';
18. Display the student and student_qualification details who have scored the maximum marks in 'BE'
select * from student s, student_qualifications sq where s.ID = sq.studentID and marks = (select max(marks) from student_qualifications where name = 'BE') and name = 'BE';
19. Display the student details who have scored the maximum marks in 'BE'
select s.* from student s, student_qualifications sq where s.ID = sq.studentID and marks = (select max(marks) from student_qualifications where name = 'BE') and name = 'BE';
20. Display the student details who have scored the minimum marks in '10' std.
select s.* from student s, student_qualifications sq where s.ID = sq.studentID and marks = (select min(marks) from student_qualifications where name = '10') and name = '10';
21. Display all student and student_qualification details of those students who have scored marks more than 'RAJAN' in 'BE'.
select * from student s, student_qualifications sq where s.ID = sq.studentID and marks > (select marks from student_qualifications sq, student s where s.ID = sq.studentID and namefirst = 'rajan' and name = 'BE') and name = 'BE';
22. Display all student who have done 'BE' in the same year as of studentID 16.
select s.* from student s, student_qualifications sq where s.ID = sq.studentID and year = (select year from student_qualifications where studentID = 16 and name = 'BE') and name = 'BE';
23. Display all odd records.
select * from (select * from student order by ID) T1 where mod(ID,2) <> 0;
24. Calculate the sum of marks student wise of their qualifications (i.e. 10 th , 12 th and BE marks)
select studentID, sum(marks) from student s, student_qualifications sq where s.ID = sq.studentID and studentID in(select marks from student_qualifications where name = '10' and name = '12' and name = 'BE') group by studentID;
25. Display students' details who are not having 'Aadhaar' card.
select * from student where not exists (select studentID from student_cards sc, student s where s.ID = sc.studentID and name = 'Aadhar');

Assignment

March22/ DBT/012
Database Technologies
Diploma in Advance Computing (PG-DAC)
March 2022

Temporary tables and VIEWS

1. Write a query to create a view named StudentAddress for all students with their address details.
<code>mysql> CREATE or replace VIEW studentAddress as SELECT s.*,sa.address FROM student s, student_address sa WHERE s.id = sa.studentID;</code>
2. Write a query to create a view named StudentQualifications for all students with their qualification details.
<code>mysql> create or replace view studentqualifications as select s.*, sq.name, sq.college, sq.university, sq.marks, sq.year from student s inner join student_qualifications sq on s.id=sq.studentid;</code>
3. Write a query to create a view named <i>ModuleDuration</i> that display the module name and the duration of the module for the batch "Batch1".
<code>mysql> create or replace view moduleduration as select m.name, m.duration from modules m inner join course_modules cm inner join course_batches cb on m.id=cm.moduleid and cm.courseid=cb.courseid and cb.name="batch1";</code>
4. Write a query to create a view named <i>PGDACModules</i> which display module names that are taught in 'PG-DAC' course.
<code>mysql> create or replace view PGDACModule as select m.name from course c, modules m, course_modules cm where c.id=cm.courseid and m.id=cm.moduleid and c.name="pg-dac";</code>
5. Write a query to create a view named <i>Student_BE_2017</i> with columns (all student details with their student_qualifications.name, student_qualifications.college, student_qualifications.university, student_qualifications.marks, and student_qualifications.year) who have done 'BE' in 2017.
<code>mysql> create or replace view student_be_2017 as select s.*, sq.name, sq.college, sq.university, sq.marks, sq.year from student s inner join student_qualifications sq on s.id=sq.studentid and sq.name="BE" and sq.year="2017";</code>
6. Write a query to create a view named <i>StudentView</i> and add 2 records from the StudentView view.
<code>mysql> create or replace view studentview as select * from student;</code> <code>mysql> insert into StudentView values(29, 'xyz', 'Abc', '1998-07-21', 'abc@gmail.com');</code> Query OK, 1 row affected (0.00 sec) <code>mysql> insert into StudentView values(30, 'ABC', 'xyz', '1999-02-20', 'xyz@gmail.com');</code> Query OK, 1 row affected (0.00 sec)

7. Write a query to create a view named CourseJava that display the courses where 'JAVA1' is taught.
<code>mysql> create or replace view coursejava as select c.name from course c inner join modules m inner join course_modules cm on c.id=cm.courseid and cm.moduleid=m.id and m.name="java1";</code>
8. Write a query to create a view named Student_A that gets all the students whose name first starts with 'A'.
<code>mysql> create or replace view student_a as select * from student where left(namefirst,1)='a';</code>
9. Create temporary table named studentAddress as Student and his address (columns to be taken namefirst, namelast, DOB, emailID, and address) (hint: use AS)
<code>mysql> create temporary table studentaddress as select s.namefirst, s.namelast, s.DOB, s.emailid, sa.address from student s inner join student_address sa on s.id=sa.studentid ;</code>
10. Create temporary table named <i>temp_student</i> alike student relation. (hint: use LIKE)
<code>mysql> create temporary table temp_student like student;</code>

Assignment

March22/ DBT/125
Database Technologies
Diploma in Advance Computing (PG-DAC)
March 2022

Basic Programming

1. Write a basic PL/SQL programme to create two variables and store some default value and print them.

```
drop procedure if exists Q1;
delimiter $
create procedure Q1()
begin
    set @x=12, @y=15;
    select @x as 'X variable', @y as 'Yvariable';
end $
delimiter ;
```

=====

2. Write a simple procedure to print 'Hello World'

```
drop procedure if exists Q2;
delimiter $
create procedure Q2()
begin
    select "Hello World!" as "Message ";
end $
delimiter ;
```

=====

3. Write a simple procedure to print a table of a given number?

```
drop procedure if exists Q3;
delimiter $
create procedure Q3(x int)
begin
    declare i,c int;
    select x 'table of ';

    set i := 1;
```

```
lbl:loop

    set c := x * i;
    select c;
    set i:=i+1;

    if i >10 then
        leave lbl;
    end if;
end loop lbl;
```

```
end $
delimiter ;
```

=====

4. Write a procedure to print the maximum number of 3 inputted numbers.

```
drop procedure if exists Q4;
delimiter $
create procedure Q4(a int, b int, c int)
BEGIN
    if a>b && a>c then
        select a;
    elseif b>c then
        select b;
    else
        select c;
    end if;
end$
delimiter ;
```

Assignment

March22/ DBT/126

Database Technologies

Diploma in Advance Computing (PG-DAC)

March 2022

Procedure

1. Create a LOGIN table (username, password, and email). Write a procedure (named *addUser*) to pass the username, password, and email-ID through the procedure and store the data in the LOGIN table.

```
drop procedure if exists adduser;
delimiter $
create procedure adduser(name varchar(10), pass varchar(10), emailid varchar(20))
BEGIN
    insert into login values(name, pass, emailid);
end $
delimiter ;
```

2. Create a LOG table having following columns (id (auto_increment), curr_date, curr_time, and message). Write a procedure (named *checkUser*) to pass the email-ID as an input, check whether passed email-ID is available in LOGIN table or not available. If the email-ID is available then display the username and his password. If the email-ID is not available then, insert (curr_date, curr_time, and message) in LOG table.

```
drop procedure if exists checkuser;
delimiter $
create procedure checkuser(email varchar(50))
BEGIN
    DECLARE V1 VARCHAR(50);
    select emailid into v1 from login where emailid=email;

    if V1 is null then
        insert into LOG values(default, curdate(), curtime(), "Email not found");
    ELSE
        select username, password from login where emailid=email;
    end IF;
end $
delimiter ;
```

```
mysql> call checkuser("pqr@gmail.com");
+-----+-----+
| username | password |
+-----+-----+
| pqr      | pqr123   |
+-----+-----+
1 row in set (0.00 sec)
```

Query OK, 0 rows affected (0.01 sec)

mysql> call checkuser("gahs@gmail.com");

Query OK, 1 row affected (0.01 sec)

mysql> select * from log;

```
+-----+-----+-----+-----+
| id | Curr_date | Curr_time | message |
+-----+-----+-----+-----+
| 1 | 2022-03-29 | 23:14:35 | Email not found |
+-----+-----+-----+-----+
```

1 row in set (0.00 sec)

3. Write a procedure(named getQualification) that takes studentID as a parameter. If studentID is present in the student table, then print his student details along with STUDENT_QUALIFICATION details and if the studentID is not present display message "Student not found..." (Use: STUDENT, and STUDENT_QUALIFICATION tables)

```
drop procedure if exists getQualification;
delimiter $
create procedure getQualification(sid int)
begin
    declare gid int;
    select id into gid from student where student.id=sid;

    if gid is null then
        select "Student not found" as "message box";
    else
        select * from student join student_qualifications where
        student.id=student_qualifications.studentid and student.id=sid;
    end if;

end $
delimiter ;
```

OUTPUT:-

mysql> call getqualification(10);

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| ID | namefirst | namelast | DOB | emailID | ID | studentID | name | college |
university | marks | year |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| 10 | nitish | patil | 1990-10-26 | nitish.patil@gmail.com | 28 | 10 | 10 | Texas | Yale
University | 65 | 2012 |
| 10 | nitish | patil | 1990-10-26 | nitish.patil@gmail.com | 29 | 10 | 12 | Oregon |
University of Michigan | 76 | 2014 |
| 10 | nitish | patil | 1990-10-26 | nitish.patil@gmail.com | 30 | 10 | M.Com. | New Mexico |
California University | 61 | 2018 |
```



```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> call getqualification(40);
+-----+
| message box      |
+-----+
| Student not found |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.02 sec)

```

4. Write a procedure (named addStudent) that inserts a new student with his phone number and his address into the STUDENT, PHONE, and ADDRESS table.

```

drop procedure if exists addstudent;
delimiter $
create procedure addstudent(firstname varchar(45), lastname varchar(45), DOB date, emailid
varchar(128), phone varchar(11), address varchar(45))
BEGIN
    declare sid, pid, aid int;

    select max(student.id)+1 into sid from student;
    select max(student_phone.id)+1 into pid from student_phone;
    select max(student_address.id)+1 into aid from student_address;

    insert into student values(sid, firstname , lastname , DOB , emailid);
    insert into student_phone values(pid,sid, phone,1);
    insert into student_address values(aid,sid, address);
end $
delimiter ;

```

OUTPUT:-

```

mysql> source E:\DAC INFOWAY\DATA BASE
TECHNOLOGIES\ASSIGNNMENT\Assignment126.sql
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> call addstudent('Sairaj','Pawar','1999-02-20', 'pawarsairaj@gmail.com', '9589899525', 'Jalgaon');
Query OK, 1 row affected (0.01 sec)

mysql>

```

5. Write a procedure (named addQualification) that takes studentID, and qualification details as a parameter. If studentID is present in the STUDENT table, then insert the qualification in STUDENT_QUALIFICATION table and return a message "Record inserted" or else print 'Student not found'. (hint: using OUT parameter) (Use: STUDENT, and STUDENT_QUALIFICATION tables)

```
drop procedure if exists addQualification;
delimiter $
create procedure addQualification(sid int, qname varchar(20), qcollege varchar(20), quniversity
varchar(20), qmarks int, qyear int)
BEGIN
    declare aid, qid int;

    select id into aid from student where student.id=sid;
    select max(id) + 1 into qid from student_qualifications;

    if aid is NULL then
        select "Student not found" as "Message";
    else
        insert into student_qualifications values(qid, sid, qname, qcollege, quniversity,
qmarks, qyear);
        select "Record Inserted" as "Message";
    end if;

end $
delimiter ;
```

OUTPUT:-

```
mysql> source E:\DAC INFOWAY\DATA BASE
TECHNOLOGIES\ASSIGNNMENT\Assignment126.sql
Query OK, 0 rows affected (0.01 sec)
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> call addQualification(40,'ME','abc','xyz',78,2021);
+-----+
| Message |
+-----+
| Student not found |
+-----+
1 row in set (0.03 sec)
```

```
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> call addQualification(1,'ME','abc','xyz',78,2021);
+-----+
| Message |
+-----+
```

| Record Inserted |

+-----+

1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql>

Assignment

March22/ DBT/127

Database Technologies

Diploma in Advance Computing (PG-DAC)

March 2022

Function

1. Pass DEPTNO to the function (named sumSalary) and calculate the sum of salary.(Use: EMP table)
<pre> DROP FUNCTION IF EXISTS sumsalary; delimiter \$ CREATE FUNCTION sumsalary(v_ID INT) RETURNS INT DETERMINISTIC begin declare v_amount INT; SELECT sum(sal) INTO v_amount FROM emp where deptno = v_ID; return(v_amount); end \$ delimiter ; </pre>
2. Create a new table called STUDENT_NEW having following columns (studentID, namefirst, namelast, DOB, and emailID). Write a function names autoNumber to return auto generate studentID and return the new value (Use: STUDENT_NEW table).
<pre> drop function if exists autonumber; delimiter \$ create function autonumber() returns INT deterministic BEGIN declare num1 int; select ifnull(max(studentid)+1,0) into num1 from student_new; return(num1); end \$ delimiter ; </pre>
3. Write a function which will accept email-ID from the user, if the email-ID is present return his username and password or else `Return "Employee not exists". (Use: LOGIN table)
<pre> drop function if exists echeck; delimiter \$ create function echeck(cemailid varchar(45)) returns varchar(100) deterministic BEGIN declare eid varchar(100); select emailid into eid from login where emailid=cemailid; if eid is null then </pre>

```

        return(select "employee not exists" as "message box");
    else
        return (select concat("username: ",(select username from login where
        emailid=cemailid)," ", "Password:" ,(select password from login where emailid=cemailid)));
    end if;
end $
delimiter ;

```

OUTPUT:-

```

mysql> source E:\DAC INFOWAY\DATA BASE
TECHNOLOGIES\ASSIGNNMENT\Assignment127.sql
Query OK, 0 rows affected (0.01 sec)

```

```

Query OK, 0 rows affected (0.00 sec)

```

```

mysql> select echeck("abc@gmail.com");

```

```

+-----+
| echeck("abc@gmail.com") |
+-----+
| username: abc Password:abc123 |
+-----+
1 row in set (0.00 sec)

```

```

mysql> select echeck("dsa@gmail.com");

```

```

+-----+
| echeck("dsa@gmail.com") |
+-----+
| employee not exists |
+-----+
1 row in set (0.00 sec)

```

- Write a function which will accept studentID from the user and calculate the sum of (10th, 12th, and BE) marks.

drop function if exists tmarks;

delimiter \$

create function tmarks(sid int) returns INT

deterministic

BEGIN

**return (select sum(marks) from student_qualifications group by studentid
having studentID=sid);**

end \$

delimiter ;

```
mysql> source E:\DAC INFOWAY\DATA BASE  
TECHNOLOGIES\ASSIGNNMENT\Assignment127.sql  
Query OK, 0 rows affected (0.01 sec)
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select tmarks(6);
```

```
+-----+  
| tmarks(6) |  
+-----+  
|      219 |  
+-----+
```

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

Assignment

March22/ DBT/128

Database Technologies

Diploma in Advance Computing (PG-DAC)

March 2022

Trigger

1. Write a trigger (named insertStudent) that saves the message "Record inserted successfully" in LOG(current date, current time, and message columns) table as soon as you insert the record in STUDENT table.

```
drop trigger if exists insertStudent;
delimiter $
create trigger insertStudent after insert on student for each ROW
begin
    insert into log values(default, current_date(), current_time(), "record inserted");
end $
delimiter ;
```

```
mysql> source E:\DAC INFOWAY\DATA BASE
TECHNOLOGIES\ASSIGNNMENT\Assignment128.sql
Query OK, 0 rows affected (0.01 sec)
```

```
Query OK, 0 rows affected (0.01 sec)
```

2. Write a trigger (named insertDuplicate) on STUDENT table, that as when we INSERT a record in STUDENT table the same record should get duplicated (INSERTED) in STUDENT_LOG table. (Create STUDENT_LOG table, having the same structure as STUDENT table).

```
drop trigger if exists insertDuplicate;
delimiter $
create trigger insertDuplicate before insert on student for each row
begin
    insert into STUDENT_LOG values (new.id, new.namefirst, new.namelast, new.DOB,
new.emailID);
end $

delimiter ;
```

OUTPUT:-

```
mysql> source E:\DAC INFOWAY\DATA BASE
TECHNOLOGIES\ASSIGNNMENT\triggers128.sql
Query OK, 0 rows affected (0.02 sec)
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> insert into student values(30,"sairaj","pawar","1999-01-28","sairaj@gmail.com");
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from student_log;
+----+-----+-----+-----+-----+
| ID | namefirst | namelast | DOB      | emailID      |
+----+-----+-----+-----+-----+
| 30 | sairaj   | pawar   | 1999-01-28 | sairaj@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

3. Write a trigger(named updateStudent) on STUDENT table, that as soon as we UPDATE student emailID column data in STUDENT table, the update record should get inserted in STUDENT_LOG table.

```
drop trigger if exists updatestudent;
delimiter $
create trigger updatestudent after update on student for each ROW
BEGIN
    insert into student_log values(old.id, old.namefirst, old.namelast, old.DOB, new.emailID);
end $
delimiter ;
```

OUTPUT:-

```
mysql> source E:\DAC INFOWAY\DATA BASE
TECHNOLOGIES\ASSIGNNMENT\triggers128.sql
Query OK, 0 rows affected (0.02 sec)
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> insert into student values(30,"sairaj","pawar","1999-01-28","sairaj@gmail.com");
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from student_log;
+----+-----+-----+-----+-----+
| ID | namefirst | namelast | DOB      | emailID      |
+----+-----+-----+-----+-----+
| 30 | sairaj   | pawar   | 1999-01-28 | sairaj@gmail.com |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> update student_log set emailid="sairajpawar@gmail.com" where id = 30;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
mysql> select * from student_log;
+----+-----+-----+-----+-----+
| ID | namefirst | namelast | DOB      | emailID      |
+----+-----+-----+-----+-----+
| 30 | sairaj   | pawar   | 1999-01-28 | sairajpawar@gmail.com |
+----+-----+-----+-----+-----+
```


+-----+-----+-----+-----+-----+-----+	
1 row in set (0.00 sec)	
4.	Write a trigger (named deleteStudent) on STUDENT table, that as soon as we DELETE any record from STUDENT table, then that record should get inserted into STUDENT_LOG table.
<pre>drop trigger if exists deleteStudent; delimiter \$ create trigger deleteStudent before delete on student for each row begin insert into student_log values (old.id, old.namefirst, old.namelast, old.DOB, old.emailID); end \$ delimiter ;</pre>	
5.	Write a trigger (named insertValidation) on STUDENT table, that if today is Sunday then, no record should get inserted in STUDENT table.
<pre>drop trigger if exists insertValidation; delimiter \$ create trigger insertValidation before insert on student for each row begin declare a varchar(45); select dayname(current_date()) into a; if a ='SUNDAY' then signal sqlstate '42000' set message_text='rec cannot be inserted on sunday'; end if ; end \$ delimiter ;</pre>	