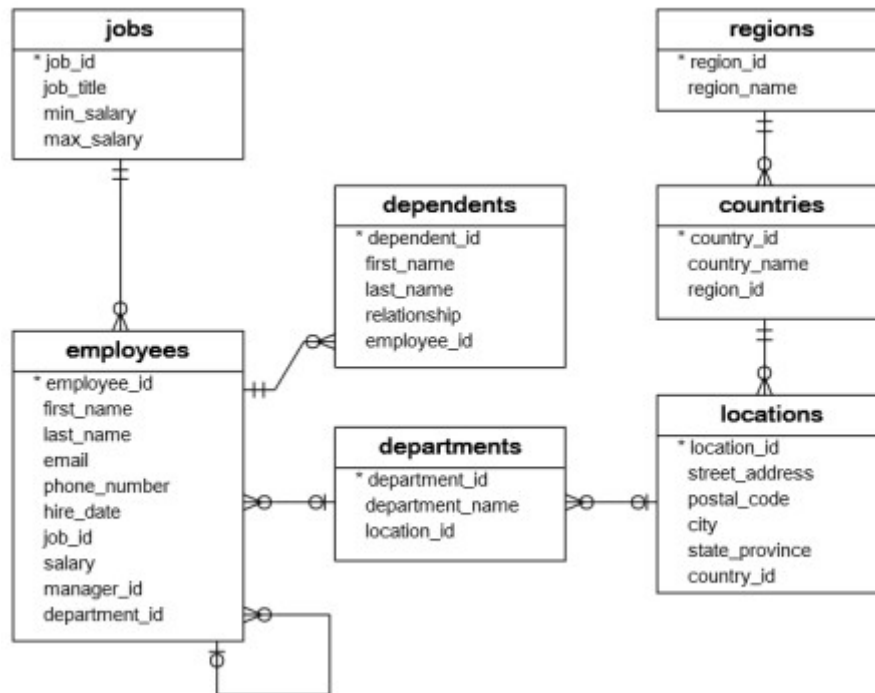


INDEX

| | |
|--|----|
| EXERCISE 1: MySQL Company Database I..... | 1 |
| EXERCISE 2: MySQL Company Database II..... | 13 |
| EXERCISE 3: MySQL University Database..... | 17 |
| EXERCISE 4: PL/SQL Procedures..... | 25 |
| EXERCISE 5: Stored Procedure..... | 31 |
| EXERCISE 6: Triggers..... | 35 |
| EXERCISE 7: MongoDB..... | 39 |

EXERCISE 1

Date: 10/02/2025



The company database has seven tables:

1. The employees table stores the data of employees.
2. The jobs table stores the job data including job title and salary range.
3. The departments table stores department data.
4. The dependents table stores the employee's dependents.
5. The locations table stores the location of the departments of the company.
6. The countries table stores the data of countries where the company is doing business.
7. The regions table stores the data of regions such as Asia, Europe, America, and the Middle East and Africa. The countries are grouped into regions.

I . WRITE QUERIES IN MYSQL TO DO THE FOLLOWING: -

DDL COMMANDS

1. Design and create database which consists of above seven tables.
2. Rename Departments table to Dept.
3. Modify Column Salary from int to smallint
4. Add Commission column to the Employees table.

DML COMMANDS

5. Enter appropriate queries to insert values in table.

R Diagram:-

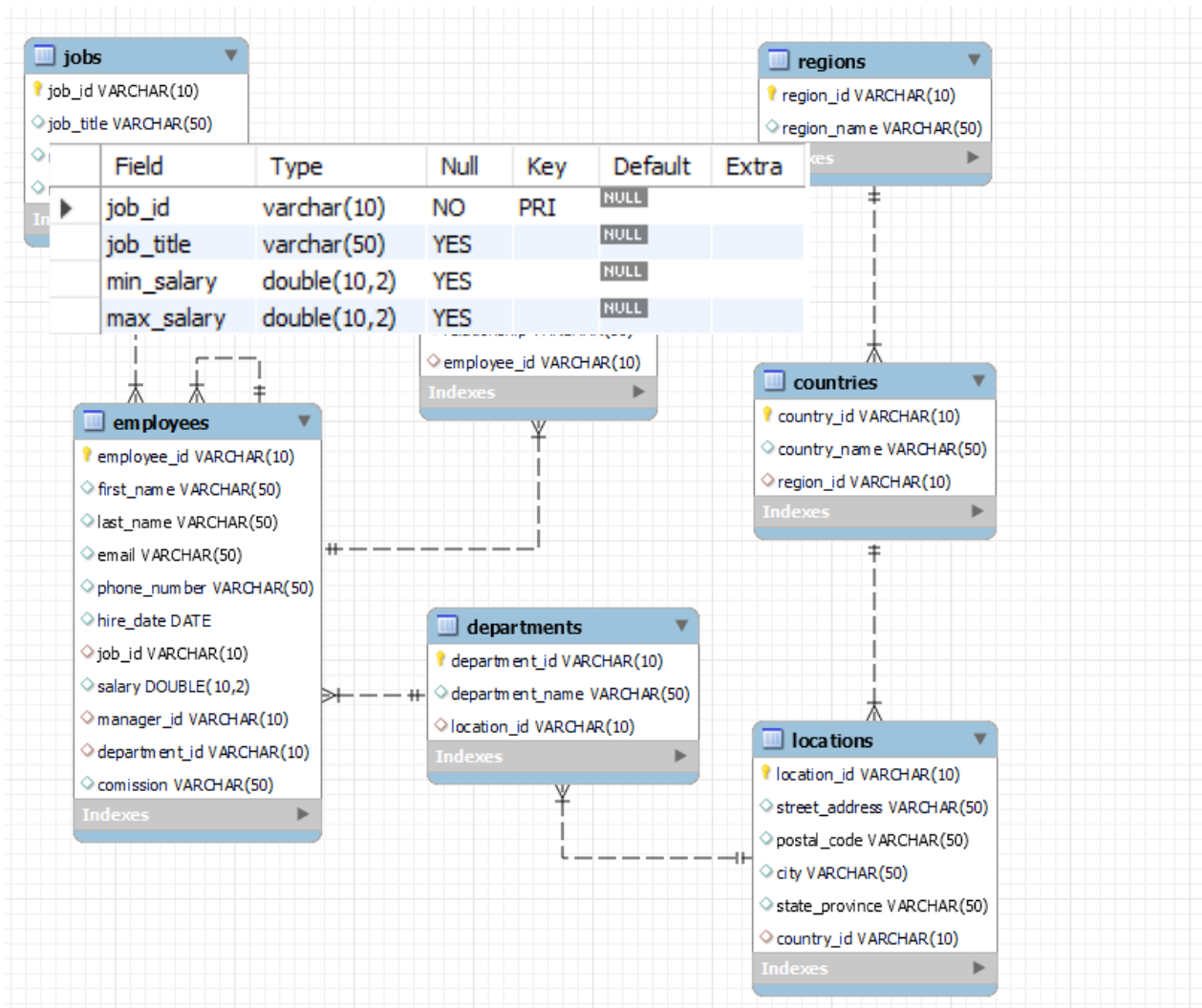


Table structure of jobs:-

Table structure of regions:-

| | Field | Type | Null | Key | Default | Extra |
|---|-------------|-------------|------|-----|---------|-------|
| ▶ | region_id | varchar(10) | NO | PRI | NULL | |
| | region_name | varchar(50) | YES | | NULL | |

SQL QUERIES: -

1. Design and create database which consists of above seven tables.

create database company;

use company;

create table jobs (job_id varchar(10) primary key, job_title varchar(50), min_salary double(10,2), max_salary double(10,2));

create table regions(region_id varchar(10) primary key, region_name varchar(50));

create table countries(country_id varchar(10) primary key, country_name varchar(50), region_id varchar(10), foreign key (region_id) references regions(region_id));

create table locations(location_id varchar(10) primary key, street_address varchar(50), postal_code varchar(50), city varchar(50), state_province varchar(50), country_id varchar(10), foreign key (country_id) references countries(country_id));

create table departments(department_id varchar(10) primary key, department_name varchar(50), location_id varchar(10), foreign key (location_id) references locations(location_id));

create table employees(employee_id varchar(10) primary key, first_name varchar(50), last_name varchar(50), email varchar(50), phone_number varchar(50), hire_date date, job_id varchar(10), salary double(10,2), manager_id varchar(10), department_id varchar(10), foreign key (department_id) references departments(department_id), foreign key (job_id) references jobs (job_id), foreign key (manager_id) references employees(employee_id));

create table dependents(dependent_id varchar(10), first_name varchar(50), last_name varchar(50), relationship varchar(50), employee_id varchar(10), foreign key (employee_id) references employees(employee_id));

2. Rename department to dept.

alter table departments rename dept;

3. Modify column salary from int to smallint.

alter table employees modify column salary SMALLINT;

4. Add commission to employees table.

alter table employees add commission varchar(50);

Table structure of countries:-

| | Field | Type | Null | Key | Default | Extra |
|---|--------------|-------------|------|-----|---------|-------|
| ▶ | country_id | varchar(10) | NO | PRI | NULL | |
| | country_name | varchar(50) | YES | | NULL | |
| | region_id | varchar(10) | YES | MUL | NULL | |

Table structure of

| | Field | Type | Null | Key | Default | Extra |
|---|----------------|-------------|------|-----|---------|-------|
| ▶ | location_id | varchar(10) | NO | PRI | NULL | |
| | street_address | varchar(50) | YES | | NULL | |
| | postal_code | varchar(50) | YES | | NULL | |
| | city | varchar(50) | YES | | NULL | |
| | state_province | varchar(50) | YES | | NULL | |
| | country_id | varchar(10) | YES | MUL | NULL | |

Table structure of departments

| | Field | Type | Null | Key | Default | Extra |
|---|-----------------|-------------|------|-----|---------|-------|
| ▶ | department_id | varchar(10) | NO | PRI | NULL | |
| | department_name | varchar(50) | YES | | NULL | |
| | location_id | varchar(10) | YES | MUL | NULL | |

| | Field | Type | Null | Key | Default | Extra |
|---|---------------|--------------|------|-----|---------|-------|
| ▶ | employee_id | varchar(10) | NO | PRI | NULL | |
| | first_name | varchar(50) | YES | | NULL | |
| | last_name | varchar(50) | YES | | NULL | |
| | email | varchar(50) | YES | | NULL | |
| | phone_number | varchar(50) | YES | | NULL | |
| | hire_date | date | YES | | NULL | |
| | job_id | varchar(10) | YES | MUL | NULL | |
| | salary | double(10,2) | YES | | NULL | |
| | manager_id | varchar(10) | YES | MUL | NULL | |
| | department_id | varchar(10) | YES | MUL | NULL | |
| | comission | varchar(50) | YES | | NULL | |

Table structure of dependents:-

| | Field | Type | Null | Key | Default | Extra |
|---|--------------|-------------|------|-----|---------|-------|
| ▶ | dependent_id | varchar(10) | YES | | NULL | |
| | first_name | varchar(50) | YES | | NULL | |
| | last_name | varchar(50) | YES | | NULL | |
| | relationship | varchar(50) | YES | | NULL | |
| | employee_id | varchar(10) | YES | MUL | NULL | |

5. Enter following queries to insert values to tables.

Data for regions:-

```
INSERT INTO regions(region_id,region_name) VALUES ('1','Europe'), ('2','Americas'), ('3','Asia'), ('4','Middle East and Africa');
```

Data for countries:-

```
INSERT INTO countries (country_id,country_name,region_id) VALUES ('AR','Argentina','2'), ('AU','Australia','3'), ('BE','Belgium','1'), ('BR','Brazil','2'), ('CA','Canada','2'), ('CH','Switzerland','1'), ('CN','China','3'), ('DE','Germany','1'), ('DK','Denmark','1'), ('EG','Egypt','4'), ('FR','France','1'), ('HK','HongKong','3'), ('IL','Israel','4'), ('IN','India','3'), ('IT','Italy','1'), ('JP','Japan','3'), ('KW','Kuwait','4'), ('MX','Mexico','2'), ('NG','Nigeria','4'), ('NL','Netherlands','1'), ('SG','Singapore','3'), ('UK','United Kingdom','1'), ('US','United States of America','2'), ('ZM','Zambia','4'), ('ZW','Zimbabwe','4');
```

Data for locations:-

```
INSERT INTO locations(location_id,street_address,postal_code,city,state_province,country_id) VALUES ('1400','2014 Jabberwocky Rd','26192','Southlake','Texas','US'), ('1500','2011 Interiors Blvd','99236','South San Francisco','California','US'), ('1700','2004 Charade Rd','98199','Seattle','Washington','US'), ('1800','147 Spadina Ave','M5V 2L7','Toronto','Ontario','CA'), ('2400','8204 Arthur St',NULL,'London',NULL,'UK'), ('2500','Magdalen Centre, The Oxford Science Park','OX99ZB','Oxford','Oxford','UK'), ('2700','Schwanthalerstr. 7031','80925','Munich','Bavaria','DE');
```

Data for jobs:-

```
INSERT INTO jobs(job_id,job_title,min_salary,max_salary) VALUES ('1','Public Accountant',4200.00,9000.00), ('2','Accounting Manager',8200.00,16000.00), ('3','Administration Assistant',3000.00,6000.00), ('4','President',20000.00,40000.00), ('5','Administration Vice President',15000.00,30000.00), ('6','Accountant',4200.00,9000.00), ('7','Finance Manager',8200.00,16000.00), ('8','Human Resources Representative',4000.00,9000.00), ('9','Programmer',4000.00,10000.00), ('10','Marketing Manager',9000.00,15000.00), ('11','Marketing Representative',4000.00,9000.00), ('12','Public Relations Representative',4500.00,10500.00), ('13','Purchasing Clerk',2500.00,5500.00), ('14','Purchasing Manager',8000.00,15000.00), ('15','Sales Manager',10000.00,20000.00), ('16','Sales Representative',6000.00,12000.00), ('17','Shipping Clerk',2500.00,5500.00), ('18','Stock Clerk',2000.00,5000.00), ('19','Stock Manager',5500.00,8500.00);
```

Table: regions

| region_id | region_name |
|-----------|------------------------|
| 1 | Europe |
| 2 | Americas |
| 3 | Asia |
| 4 | Middle East and Africa |

Table: countries

| country_id | country_name | region_id |
|------------|------------------|-----------|
| AR | Argentina | 2 |
| AU | Australia | 3 |
| BE | Belgium | 1 |
| BR | Brazil | 2 |
| CA | Canada | 2 |
| CH | Switzerland | 1 |
| CN | China | 3 |
| DE | Germany | 1 |
| DK | Denmark | 1 |
| EG | Egypt | 4 |
| FR | France | 1 |
| HK | HongKong | 3 |
| IL | Israel | 4 |
| IN | India | 3 |
| IT | Italy | 1 |
| JP | Japan | 3 |
| KW | Kuwait | 4 |
| MX | Mexico | 2 |
| NG | Nigeria | 4 |
| NL | Netherlands | 1 |
| SG | Singapore | 3 |
| UK | United Kingdom | 1 |
| US | United States... | 2 |
| ZM | Zambia | 4 |
| ZW | Zimbabwe | 4 |

Table: locations

| location_id | street_address | postal_code | city | state_province | country_id |
|-------------|--|-------------|---------------------|----------------|------------|
| 1400 | 2014 Jabberwocky Rd | 26192 | Southlake | Texas | US |
| 1500 | 2011 Interiors Blvd | 99236 | South San Francisco | California | US |
| 1700 | 2004 Charade Rd | 98199 | Seattle | Washington | US |
| 1800 | 147 Spadina Ave | M5V 2L7 | Toronto | Ontario | CA |
| 2400 | 8204 Arthur St | NULL | London | NULL | UK |
| 2500 | Magdalen Centre, The Oxford Science Park | OX99ZB | Oxford | Oxford | UK |
| 2700 | Schwanthalerstr. 7031 | 80925 | Munich | Bavaria | DE |

```
INSERT INTO departments(department_id,department_name,location_id) VALUES  
(1,'Administration','1700'), (2,'Marketing','1800'), (3,'Purchasing','1700'), (4,'Human Resources',  
'2400'), (5,'Shipping','1500'), (6,'IT','1400'), (7,'Public Relations','2700'), (8,'Sales','2500'),  
(9,'Executive','1700'), (10,'Finance','1700'), (11,'Accounting','1700');
```

Data for employees:-

```
INSERT INTO employees(employee_id,first_name,last_name,email,phone_number,hire_date,  
job_id, salary,manager_id,department_id) VALUES (100,'Steven','King',  
'steven.king@sqltutorial.org','515.123.4567','1987-06-17',4,24000.00,NULL,9),  
(101,'Neena','Kochhar','neena.kochhar@sqltutorial.org','515.123.4568','1989-09-21',5,17000.00,  
100,9), (102,'Lex','DeHaan','lex.de haan@sqltutorial.org','515.123.4569','1993-01-13',5,17000.00,  
100,9), (103,'Alexander','Hunold','alexander.hunold@sqltutorial.org','590.423.4567','1990-01-  
03',9,9000.00,102,6), (104,'Bruce','Ernst','bruce.ernst@sqltutorial.org','590.423.4568','1991-05-  
21',9,6000.00,103,6), (105,'David','Austin','david.austin@sqltutorial.org','590.423.4569','1997-06-  
25',9,4800.00,103,6), (106,'Valli','Pataballa','valli.pataballa@sqltutorial.org','590.423.4560','1998-  
02-05',9,4800.00,103,6), (107,'Diana','Lorentz','diana.lorentz@sqltutorial.org','590.423.5567','1999-  
02-07',9,4200.00,103,6), (108,'Nancy','Greenberg','nancy.greenberg@sqltutorial.org',  
'515.124.4569','1994-08-17',7,12000.00,101,10), (109,'Daniel','Faviet',  
'daniel.faviet@sqltutorial.org','515.124.4169','1994-08-16',6,9000.00,108,10),  
(110,'John','Chen','john.chen@sqltutorial.org','515.124.4269','1997-09-28',6,8200.00,108,10),  
(111,'Ismael','Sciarra','ismael.sciarra@sqltutorial.org','515.124.4369','1997-09-30',6,7700.00,  
108,10), (112,'Jose Manuel','Urman','josemanuel.urman@sqltutorial.org','515.124.4469','1998-03-  
07',6,7800.00,108,10), (113,'Luis','Popp','luis.popp@sqltutorial.org','515.124.4567','1999-12-  
07',6,6900.00,108,10), (114,'Den','Raphaely','den.rafaely@sqltutorial.org','515.127.4561','1994-12-  
07',14,11000.00,100,3), (115,'Alexander','Khoo','alexander.khoo@sqltutorial.org','515.127.4562',  
'1995-05-18',13,3100.00,114,3), (116,'Shelli','Baida','shelli.baida@sqltutorial.org','515.127.4563',  
'1997-12-24',13,2900.00,114,3), (117,'Sigal','Tobias','sigal.tobias@sqltutorial.org','515.127.4564',  
'1997-07-24',13,2800.00,114,3), (118,'Guy','Himuro','guy.himuro@sqltutorial.org','515.127.4565',  
'1998-11-15',13,2600.00,114,3), (119,'Karen','Colmenares','karen.colmenares@sqltutorial.org',  
'515.127.4566','1999-08-10',13,2500.00,114,3), (120,'Matthew','Weiss',  
'matthew.weiss@sqltutorial.org','650.123.1234','1996-07-18',19,8000.00,100,5), (121,'Adam','Fripp',  
'adam.fripp@sqltutorial.org','650.123.2234','1997-04-10',19,8200.00,100,5), (122,'Payam',  
'Kaufling','payam.kaufling@sqltutorial.org','650.123.3234','1995-05-01',19,7900.00,100,5),  
(123,'Shanta','Vollman','shanta.vollman@sqltutorial.org','650.123.4234','1997-10-10',19,6500.00,
```


| job_id | job_title | min_salary | max_salary |
|--------|---------------------------------|------------|------------|
| 1 | Public Accountant | 4200.00 | 9000.00 |
| 10 | Marketing Manager | 9000.00 | 15000.00 |
| 11 | Marketing Representative | 4000.00 | 9000.00 |
| 12 | Public Relations Representative | 4500.00 | 10500.00 |
| 13 | Purchasing Clerk | 2500.00 | 5500.00 |
| 14 | Purchasing Manager | 8000.00 | 15000.00 |
| 15 | Sales Manager | 10000.00 | 20000.00 |
| 16 | Sales Representative | 6000.00 | 12000.00 |
| 17 | Shipping Clerk | 2500.00 | 5500.00 |
| 18 | Stock Clerk | 2000.00 | 5000.00 |
| 19 | Stock Manager | 5500.00 | 8500.00 |
| 2 | Accounting Manager | 8200.00 | 16000.00 |
| 3 | Administration Assistant | 3000.00 | 6000.00 |
| 4 | President | 20000.00 | 40000.00 |
| 5 | Administration Vice President | 15000.00 | 30000.00 |
| 6 | Accountant | 4200.00 | 9000.00 |
| 7 | Finance Manager | 8200.00 | 16000.00 |
| 8 | Human Resources Representa... | 4000.00 | 9000.00 |
| 9 | Programmer | 4000.00 | 10000.00 |

| dependent_id | first_name | last_name | relationship | employee_id |
|--------------|------------|------------|--------------|-------------|
| 1 | Penelope | Gietz | Child | 206 |
| 2 | Nick | Higgins | Child | 205 |
| 3 | Ed | Whalen | Child | 200 |
| 4 | Jennifer | King | Child | 100 |
| 5 | Johnny | Kochhar | Child | 101 |
| 6 | Bette | De Haan | Child | 102 |
| 7 | Grace | Faviet | Child | 109 |
| 8 | Matthew | Chen | Child | 110 |
| 9 | Joe | Sciarra | Child | 111 |
| 10 | Christian | Urman | Child | 112 |
| 11 | Zero | Popp | Child | 113 |
| 12 | Karl | Greenberg | Child | 108 |
| 13 | Uma | Mavris | Child | 203 |
| 14 | Vivien | Hunold | Child | 103 |
| 15 | Cuba | Ernst | Child | 104 |
| 16 | Fred | Austin | Child | 105 |
| 17 | Helen | Pataballa | Child | 106 |
| 18 | Dan | Lorentz | Child | 107 |
| 19 | Bob | Hartstein | Child | 201 |
| 20 | Lucille | Fay | Child | 202 |
| 21 | Kirsten | Baer | Child | 204 |
| 22 | Elvis | Khoo | Child | 115 |
| 23 | Sandra | Baida | Child | 116 |
| 24 | Cameron | Tobias | Child | 117 |
| 25 | Kevin | Himuro | Child | 118 |
| 26 | Rip | Colmenares | Child | 119 |
| 27 | Julia | Raphaely | Child | 114 |
| 28 | Woody | Russell | Child | 145 |
| 29 | Alec | Partners | Child | 146 |
| 30 | Sandra | Taylor | Child | 176 |

Table: dependents

100,5), (126,'Irene','Mikkilineni','irene.mikkilineni@sqltutorial.org','650.124.1224','1998-09-28',18,2700.00,120,5), (145,'John','Russell','john.russell@sqltutorial.org',NULL,'1996-10-01',15,14000.00,100,8), (146,'Karen','Partners','karen.partners@sqltutorial.org',NULL,'1997-01-05',15,13500.00,100,8), (176,'Jonathon','Taylor','jonathon.taylor@sqltutorial.org',NULL,'1998-03-24',16,8600.00,100,8), (177,'Jack','Livingston','jack.livingston@sqltutorial.org',NULL,'1998-04-23',16,8400.00,100,8), (178,'Kimberely','Grant','kimberely.grant@sqltutorial.org',NULL,'1999-05-24',16,7000.00,100,8), (179,'Charles','Johnson','charles.johnson@sqltutorial.org',NULL,'2000-01-04',16,6200.00,100,8), (192,'Sarah','Bell','sarah.bell@sqltutorial.org','650.501.1876','1996-02-04',17,4000.00,123,5), (193,'Britney','Everett','britney.everett@sqltutorial.org','650.501.2876','1997-03-03',17,3900.00,123,5), (200,'Jennifer','Whalen','jennifer.whelen@sqltutorial.org','515.123.4444','1987-09-17',3,4400.00,101,1), (201,'Michael','Hartstein','michael.hartstein@sqltutorial.org','515.123.5555','1996-02-17',10,13000.00,100,2), (202,'Pat','Fay','pat.fay@sqltutorial.org','603.123.6666','1997-08-17',11,6000.00,201,2), (203,'Susan','Mavris','susan.mavris@sqltutorial.org','515.123.7777','1994-06-07',8,6500.00,101,4), (204,'Hermann','Baer','hermann.baer@sqltutorial.org','515.123.8888','1994-06-07',12,10000.00,101,7), (205,'Shelley','Higgins','shelley.higgins@sqltutorial.org','515.123.8080','1994-06-07',2,12000.00,101,11), (206,'William','Gietz','william.gietz@sqltutorial.org','515.123.8181','1994-06-07',1,8300.00,205,11);

Data for dependents:-

```
INSERT INTO dependents(dependent_i,first_name,last_name,relationship,employee_id) VALUES
(1,'Penelope','Gietz','Child',206), (2,'Nick','Higgins','Child',205), (3,'Ed','Whalen','Child',200),
(4,'Jennifer','King','Child',100), (5,'Johnny','Kochhar','Child',101), (6,'Bette','De Haan','Child',102),
(7,'Grace','Faviet','Child',109), (8,'Matthew','Chen','Child',110), (9,'Joe','Sciarra','Child',111),
(10,'Christian','Urman','Child',112), (11,'Zero','Popp','Child',113), (12,'Karl','Greenberg','Child',108),
(13,'Uma','Mavris','Child',203), (14,'Vivien','Hunold','Child',103), (15,'Cuba','Ernst','Child',104),
(16,'Fred','Austin','Child',105), (17,'Helen','Pataballa','Child',106), (18,'Dan','Lorentz','Child',107),
(19,'Bob','Hartstein','Child',201), (20,'Lucille','Fay','Child',202), (21,'Kirsten','Baer','Child',204),
(22,'Elvis','Khoo','Child',115), (23,'Sandra','Baida','Child',116), (24,'Cameron','Tobias','Child',117),
(25,'Kevin','Himuro','Child',118), (26,'Rip','Colmenares','Child',119), (27,'Julia','Raphaely','Child',114),
(28,'Woody','Russell','Child',145), (29,'Alec','Partners','Child',146),
(30,'Sandra','Taylor','Child',176);
```

Table: departments

| department_id | department_name | location_id |
|---------------|------------------|-------------|
| 1 | Administration | 1700 |
| 10 | Finance | 1700 |
| 11 | Accounting | 1700 |
| 2 | Marketing | 1800 |
| 3 | Purchasing | 1700 |
| 4 | Human Resources | 2400 |
| 5 | Shipping | 1500 |
| 6 | IT | 1400 |
| 7 | Public Relations | 2700 |
| 8 | Sales | 2500 |
| 9 | Executive | 1700 |

Table: employees

| employee_id | first_name | last_name | email | phone_number | hire_date | job_id | salary | manager_id | department_id | comission |
|-------------|-------------|-------------|-----------------------------------|--------------|------------|--------|----------|------------|---------------|-----------|
| 100 | Steven | King | steven.king@sqltutorial.org | 515.123.4567 | 1987-06-17 | 4 | 24000.00 | NULL | 9 | NULL |
| 101 | Neena | Kochhar | neena.kochhar@sqltutorial.org | 515.123.4568 | 1989-09-21 | 5 | 17000.00 | 100 | 9 | NULL |
| 102 | Lex | DeHaan | lex.de.haan@sqltutorial.org | 515.123.4569 | 1993-01-13 | 5 | 17000.00 | 100 | 9 | NULL |
| 103 | Alexander | Hunold | alexander.hunold@sqltutorial.org | 590.423.4567 | 1990-01-03 | 9 | 9000.00 | 102 | 6 | NULL |
| 104 | Bruce | Ernst | bruce.ernst@sqltutorial.org | 590.423.4568 | 1991-05-21 | 9 | 6000.00 | 103 | 6 | NULL |
| 105 | David | Austin | david.austin@sqltutorial.org | 590.423.4569 | 1997-06-25 | 9 | 4800.00 | 103 | 6 | NULL |
| 106 | Valli | Pataballa | valli.pataballa@sqltutorial.org | 590.423.4560 | 1998-02-05 | 9 | 4800.00 | 103 | 6 | NULL |
| 107 | Diana | Lorentz | diana.lorentz@sqltutorial.org | 590.423.5567 | 1999-02-07 | 9 | 4200.00 | 103 | 6 | NULL |
| 108 | Nancy | Greenberg | nancy.greenberg@sqltutorial.org | 515.124.4569 | 1994-08-17 | 7 | 12000.00 | 101 | 10 | NULL |
| 109 | Daniel | Faviet | daniel.faviet@sqltutorial.org | 515.124.4169 | 1994-08-16 | 6 | 9000.00 | 108 | 10 | NULL |
| 110 | John | Chen | john.chen@sqltutorial.org | 515.124.4269 | 1997-09-28 | 6 | 8200.00 | 108 | 10 | NULL |
| 111 | Ismael | Sciarra | ismael.sciarra@sqltutorial.org | 515.124.4369 | 1997-09-30 | 6 | 7700.00 | 108 | 10 | NULL |
| 112 | Jose Manuel | Urman | josemanuel.urman@sqltutorial.org | 515.124.4469 | 1998-03-07 | 6 | 7800.00 | 108 | 10 | NULL |
| 113 | Luis | Popp | luis.popp@sqltutorial.org | 515.124.4567 | 1999-12-07 | 6 | 6900.00 | 108 | 10 | NULL |
| 114 | Den | Raphaely | den.raphaely@sqltutorial.org | 515.127.4561 | 1994-12-07 | 14 | 11000.00 | 100 | 3 | NULL |
| 115 | Alexander | Khoo | alexander.khoo@sqltutorial.org | 515.127.4562 | 1995-05-18 | 13 | 3100.00 | 114 | 3 | NULL |
| 116 | Shelli | Baida | shelli.baida@sqltutorial.org | 515.127.4563 | 1997-12-24 | 13 | 2900.00 | 114 | 3 | NULL |
| 117 | Sigal | Tobias | sigal.tobias@sqltutorial.org | 515.127.4564 | 1997-07-24 | 13 | 2800.00 | 114 | 3 | NULL |
| 118 | Guy | Himuro | guy.himuro@sqltutorial.org | 515.127.4565 | 1998-11-15 | 13 | 2600.00 | 114 | 3 | NULL |
| 119 | Karen | Colmenares | karen.colmenares@sqltutorial.org | 515.127.4566 | 1999-08-10 | 13 | 2500.00 | 114 | 3 | NULL |
| 120 | Matthew | Weiss | matthew.weiss@sqltutorial.org | 650.123.1234 | 1996-07-18 | 19 | 8000.00 | 100 | 5 | NULL |
| 121 | Adam | Fripp | adam.fripp@sqltutorial.org | 650.123.2234 | 1997-04-10 | 19 | 8200.00 | 100 | 5 | NULL |
| 122 | Payam | Kaufing | payam.kaufing@sqltutorial.org | 650.123.3234 | 1995-05-01 | 19 | 7900.00 | 100 | 5 | NULL |
| 123 | Shanta | Vollman | shanta.vollman@sqltutorial.org | 650.123.4234 | 1997-10-10 | 19 | 6500.00 | 100 | 5 | NULL |
| 126 | Irene | Mikkilineni | irene.mikkilineni@sqltutorial.org | 650.124.1224 | 1998-09-28 | 18 | 2700.00 | 120 | 5 | NULL |
| 145 | John | Russell | john.russell@sqltutorial.org | NULL | 1996-10-01 | 15 | 14000.00 | 100 | 8 | NULL |
| 146 | Karen | Partners | karen.partners@sqltutorial.org | NULL | 1997-01-05 | 15 | 13500.00 | 100 | 8 | NULL |
| 176 | Jonathon | Taylor | jonathon.taylor@sqltutorial.org | NULL | 1998-03-24 | 16 | 8600.00 | 100 | 8 | NULL |
| 177 | Jack | Livingston | jack.livingston@sqltutorial.org | NULL | 1998-04-23 | 16 | 8400.00 | 100 | 8 | NULL |
| 178 | Kimberely | Grant | kimberely.grant@sqltutorial.org | NULL | 1999-05-24 | 16 | 7000.00 | 100 | 8 | NULL |
| 179 | Charles | Johnson | charles.johnson@sqltutorial.org | NULL | 2000-01-04 | 16 | 6200.00 | 100 | 8 | NULL |
| 192 | Sarah | Bell | sarah.bell@sqltutorial.org | 650.501.1876 | 1996-02-04 | 17 | 4000.00 | 123 | 5 | NULL |
| 193 | Britney | Everett | britney.everett@sqltutorial.org | 650.501.2876 | 1997-03-03 | 17 | 3900.00 | 123 | 5 | NULL |
| 200 | Jennifer | Whalen | jennifer.whelen@sqltutorial.org | 515.123.4444 | 1987-09-17 | 3 | 4400.00 | 101 | 1 | NULL |
| 201 | Michael | Hartstein | michael.hartstein@sqltutorial.org | 515.123.5555 | 1996-02-17 | 10 | 13000.00 | 100 | 2 | NULL |
| 202 | Pat | Fay | pat.fay@sqltutorial.org | 603.123.6666 | 1997-08-17 | 11 | 6000.00 | 201 | 2 | NULL |

Result: -

Queries are executed and output is verified.

1) All Countries.

| Countries |
|--------------------------|
| Argentina |
| Australia |
| Belgium |
| Brazil |
| Canada |
| Switzerland |
| China |
| Germany |
| Denmark |
| Egypt |
| France |
| HongKong |
| Israel |
| India |
| Italy |
| Japan |
| Kuwait |
| Mexico |
| Nigeria |
| Netherlands |
| Singapore |
| United Kingdom |
| United States of America |
| Zambia |
| Zimbabwe |

2) Email and Phone number of all employees.

| email | phone_number |
|-----------------------------------|--------------|
| steven.king@sqltutorial.org | 515.123.4567 |
| neena.kochhar@sqltutorial.org | 515.123.4568 |
| lex.de.haan@sqltutorial.org | 515.123.4569 |
| alexander.hunold@sqltutorial.org | 590.423.4567 |
| bruce.ernst@sqltutorial.org | 590.423.4568 |
| david.austin@sqltutorial.org | 590.423.4569 |
| valli.pataballa@sqltutorial.org | 590.423.4560 |
| diana.lorentz@sqltutorial.org | 590.423.5567 |
| nancy.greenberg@sqltutorial.org | 515.124.4569 |
| daniel.faviet@sqltutorial.org | 515.124.4169 |
| john.chen@sqltutorial.org | 515.124.4269 |
| ismael.sciarra@sqltutorial.org | 515.124.4369 |
| josemanuel.urman@sqltutorial.org | 515.124.4469 |
| luis.popp@sqltutorial.org | 515.124.4567 |
| den.raphaely@sqltutorial.org | 515.127.4561 |
| alexander.khoo@sqltutorial.org | 515.127.4562 |
| shelli.baida@sqltutorial.org | 515.127.4563 |
| sigal.tobias@sqltutorial.org | 515.127.4564 |
| guy.himuro@sqltutorial.org | 515.127.4565 |
| karen.colmenares@sqltutorial.org | 515.127.4566 |
| matthew.weiss@sqltutorial.org | 650.123.1234 |
| adam.fripp@sqltutorial.org | 650.123.2234 |
| payam.kaufing@sqltutorial.org | 650.123.3234 |
| shanta.vollman@sqltutorial.org | 650.123.4234 |
| irene.mikkilineni@sqltutorial.org | 650.124.1224 |
| john.russell@sqltutorial.org | NULL |
| karen.partners@sqltutorial.org | NULL |
| jonathon.taylor@sqltutorial.org | NULL |
| jack.livingston@sqltutorial.org | NULL |
| kimberely.grant@sqltutorial.org | NULL |
| charles.johnson@sqltutorial.org | NULL |
| sarah.bell@sqltutorial.org | 650.501.1876 |
| britney.everett@sqltutorial.org | 650.501.2876 |

3) Employee with last name “Fay”.

| employee_id | first_name | last_name | email | phone_number | hire_date | job_id | salary | manager_id | department_id | comission |
|-------------|------------|-----------|-------------------------|--------------|------------|--------|---------|------------|---------------|-----------|
| 202 | Pat | Fay | pat.fay@sqltutorial.org | 603.123.6666 | 1997-08-17 | 11 | 6000.00 | 201 | 2 | NULL |

4) Hire date of employees with last name “Grant” or “Whalen”

| Employee_Name | hire_date |
|-----------------|------------|
| Kimberely Grant | 1999-05-24 |
| Jennifer Whalen | 1987-09-17 |

5) Employees who are shipping clerk

| Employee_Name | job_id |
|-----------------|--------|
| Sarah Bell | 17 |
| Britney Everett | 17 |

7) Departments in descending order

| department_name |
|------------------|
| Shipping |
| Sales |
| Purchasing |
| Public Relations |
| Marketing |
| IT |
| Human Resources |
| Finance |
| Executive |
| Administration |
| Accounting |

EXERCISE 2

Date: 25/02/2025

SQL QUERIES: -

1) Write a query to display all the countries.

- select country_name as Countries from countries;

2) Write a query to display specific columns like email and phone number for all the employees.

➤ select email, phone_number from employees;

3) Write a query to display the data of employee whose last name is “Fay”.

➤ select * from employees where last_name="Fay";

4) Write a query to find the hire date for employees whose last name is “Grant” or “Whalen”.

➤ select hire_date from employees where last_name in ("Grant","Whalen");

5) Write a query to display name of the employee who is shipping clerk.

➤ select concat(employees.first_name," ",employees.last_name) as Employee_Name, jobs.job_id from employees inner join jobs on employees.job_id=jobs.job_id where jobs.job_title="shipping clerk";

6) Write a query to get all the employees who work for department 8.

➤ select concat(first_name," ",last_name) as Employee_Name, department_id from employees where department_id="8";

7) Write a query to display the departments in the descending order.

➤ select department_name from departments order by department_name desc;

8) Write a query to display all the employees whose last name starts with “K”.

➤ select concat(first_name," ",last_name) as Employee_Name from employees where last_name like "K%";

9) Display name of the employees whose hire dates are between 1995 and 1997.

➤ select concat(first_name," ",last_name) as Employee_Name, hire_date from employees where year(hire_date) between 1995 and 1997;

10) Write a query to display jobs where the maximum salary is less than 5000.

➤ select job_title from jobs where max_salary<=5000;

11) Write a query to display email address in lower case.

➤ select lower(email) as Email from employees;

8) Employees with last name starting with K”

| | Employee_Name |
|---|----------------|
| ► | Steven King |
| | Neena Kochhar |
| | Alexander Khoo |

10) Jobs with maximum salary less than or equal to 5000

| | job_title |
|---|-------------|
| ► | Stock Clerk |

11) Emails of employees in lower case.

| | Email |
|---|-----------------------------------|
| ► | steven.king@sqltutorial.org |
| | neena.kochhar@sqltutorial.org |
| | lex.de haan@sqltutorial.org |
| | alexander.hunold@sqltutorial.org |
| | bruce.ernst@sqltutorial.org |
| | david.austin@sqltutorial.org |
| | valli.pataballa@sqltutorial.org |
| | diana.lorentz@sqltutorial.org |
| | nancy.greenberg@sqltutorial.org |
| | daniel.faviet@sqltutorial.org |
| | john.chen@sqltutorial.org |
| | ismael.sciarra@sqltutorial.org |
| | josemanuel.urman@sqltutorial.org |
| | luis.popp@sqltutorial.org |
| | den.raphaely@sqltutorial.org |
| | alexander.khoo@sqltutorial.org |
| | shelli.baida@sqltutorial.org |
| | sigal.tobias@sqltutorial.org |
| | guy.himuro@sqltutorial.org |
| | karen.colmenares@sqltutorial.org |
| | matthew.weiss@sqltutorial.org |
| | adam.fripp@sqltutorial.org |
| | payam.kaufing@sqltutorial.org |
| | shanta.vollman@sqltutorial.org |
| | irene.mikkilineni@sqltutorial.org |
| | john.russell@sqltutorial.org |
| | karen.partners@sqltutorial.org |
| | jonathon.taylor@sqltutorial.org |
| | jack.livingston@sqltutorial.org |
| | kimberely.grant@sqltutorial.org |
| | charles.johnson@sqltutorial.org |
| | sarah.bell@sqltutorial.org |
| | britney.everett@sqltutorial.org |
| | jennifer.whalen@sqltutorial.org |
| | michael.hartstein@sqltutorial.org |
| | pat.fay@sqltutorial.org |
| | susan.mavris@sqltutorial.org |

9) Employees whose hire dates are between 1995 and 1997.

| | Employee_Name | hire_date |
|---|-------------------|------------|
| ► | Payam Kaufing | 1995-05-01 |
| | Alexander Khoo | 1995-05-18 |
| | Sarah Bell | 1996-02-04 |
| | Michael Hartstein | 1996-02-17 |
| | Matthew Weiss | 1996-07-18 |
| | John Russell | 1996-10-01 |
| | Karen Partners | 1997-01-05 |
| | Britney Everett | 1997-03-03 |
| | Adam Frinn | 1997-04-10 |

12) Employees who were hired in 1995.

| | Employee_Name |
|---|----------------|
| ► | Alexander Khoo |
| | Payam Kaufing |

13) New Employee: Paul Newton in department 11

| | Employee_Name | department_id |
|---|---------------|---------------|
| ► | Paul Newton | 11 |

14) Deleted department Shipping

| | department_id | department_name | location_id |
|---|---------------|-----------------|-------------|
| * | NULL | NULL | NULL |

15) Employees who were hired on a Thursday.

| | Employee_Name | hire_date |
|---|-----------------|------------|
| ► | Neena Kochhar | 1989-09-21 |
| | Valli Pataballa | 1998-02-05 |
| | Alexander Khoo | 1995-05-18 |
| | Sigal Tobias | 1997-07-24 |
| | Matthew Weiss | 1996-07-18 |
| | Adam Frinn | 1997-04-10 |
| | Jack Livingston | 1998-04-23 |
| | Jennifer Whalen | 1987-09-17 |

12) Write a query to display name of the employees who were hired in 1995.

- select concat(first_name," ",last_name) as Employee_Name from employees where year(hire_date)=1995;

13) Write a query to insert an employee “Paul Newton” in department 11.

- insert into employees (employee_id,first_name,last_name,email,phone_number, hire_date, job_id,salary,manager_id,department_id) values(207,'Paul','Newton', 'paul.newton@sqltutorial.org','585.123.4567','1987-07-17',2,12000.00,101,11);
- select concat(first_name," ",last_name) as Employee_Name from employees where employee_id=207;

14) Write a query to delete the shipping department.

- delete from departments where department_name="shipping";
- select * from departments where department_name="shipping";

15) Write a query to display employees who were hired on a Thursday.

- select concat(first_name," ",last_name) as Employee_Name, hire_date from employees where dayname(hire_date)='Thursday';

Result: -

Queries are executed and output is verified.

ER Diagram:

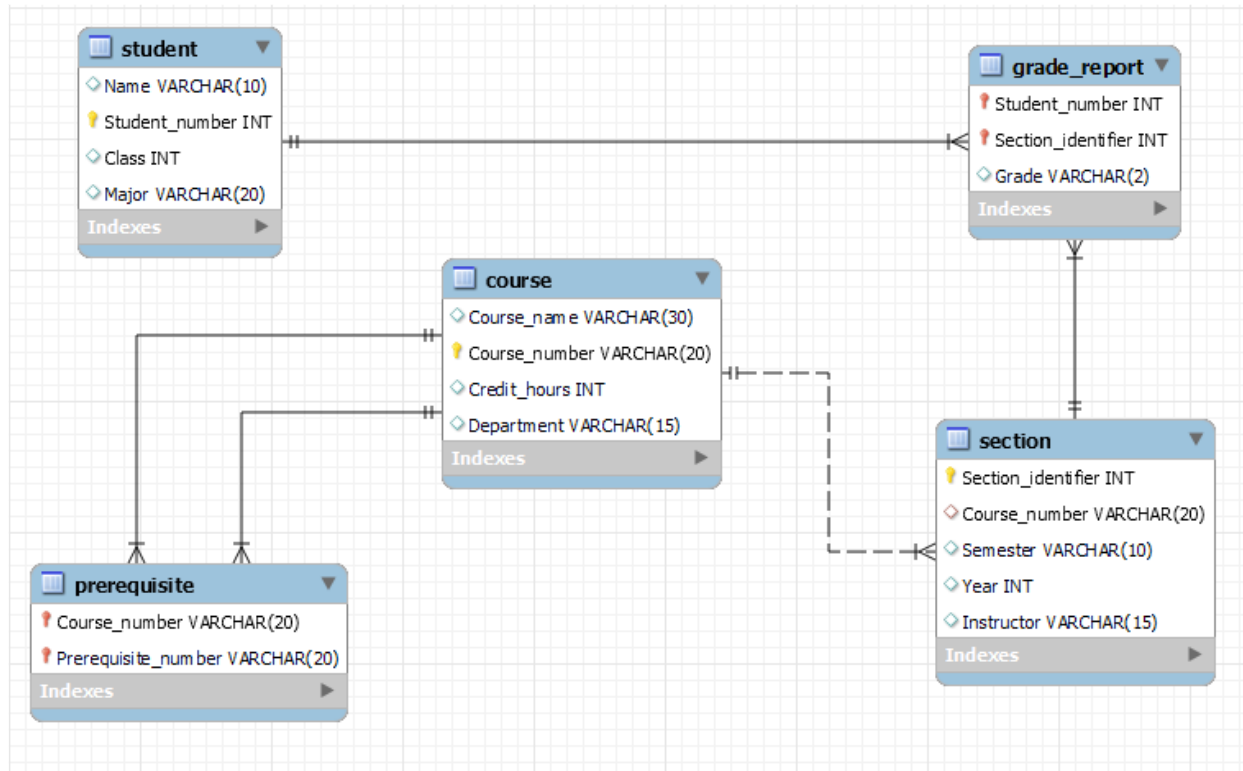


Table structure: STUDENT

| Field | Type | Null | Key | Default | Extra |
|----------------|-------------|------|-----|---------|-------|
| Name | varchar(10) | YES | | NULL | |
| Student_number | int | NO | PRI | NULL | |
| Class | int | YES | | NULL | |
| Major | varchar(20) | YES | | NULL | |

Table structure: COURSE

| Field | Type | Null | Key | Default | Extra |
|---------------|-------------|------|-----|---------|-------|
| Course_name | varchar(30) | YES | | NULL | |
| Course_number | varchar(20) | NO | PRI | NULL | |
| Credit_hours | int | YES | | NULL | |
| Department | varchar(15) | YES | | NULL | |

Table structure: SECTION

| Field | Type | Null | Key | Default | Extra |
|--------------------|-------------|------|-----|---------|-------|
| Section_identifier | int | NO | PRI | NULL | |
| Course_number | varchar(20) | YES | MUL | NULL | |
| Semester | varchar(10) | YES | | NULL | |
| Year | int | YES | | NULL | |
| Instructor | varchar(15) | YES | | NULL | |

Table structure: GRADE_REPORT

| Field | Type | Null | Key | Default | Extra |
|--------------------|------------|------|-----|---------|-------|
| Student_number | int | NO | PRI | NULL | |
| Section_identifier | int | NO | PRI | NULL | |
| Grade | varchar(2) | YES | | NULL | |

Table structure: PREREQUISITE

| Field | Type | Null | Key | Default | Extra |
|---------------------|-------------|------|-----|---------|-------|
| Course_number | varchar(20) | NO | PRI | NULL | |
| Prerequisite_number | varchar(20) | NO | PRI | NULL | |

EXERCISE 3: Join

Date: 25/02/2025

A **UNIVERSITY** database for maintaining information concerning students, courses, and grades in a university environment is given below.

The **STUDENT** file stores data on each student, the **COURSE** file stores data on each course, the **SECTION** file stores data in each section of a course, the **GRADE_REPORT** file stores the grades that students receive in the various section they have completed, and the **PREREQUISITE** files stores the prerequisites of each course.

STUDENT

| Name | Student_number | Class | Major |
|-------|----------------|-------|-------|
| Smith | 17 | 1 | CS |
| Brown | 8 | 2 | CS |

COURSE

| Course_name | Course_number | Credit_hours | Department |
|---------------------------|---------------|--------------|------------|
| Intro to Computer Science | CS1310 | 4 | CS |
| Data Structures | CS3320 | 4 | CS |
| Discrete Mathematics | MATH2410 | 3 | MATH |
| Database | CS3380 | 3 | CS |

SECTION

| Section_identifier | Course_number | Semester | Year | Instructor |
|--------------------|---------------|----------|------|------------|
| 85 | MATH2410 | Fall | 07 | King |
| 92 | CS1310 | Fall | 07 | Anderson |
| 102 | CS3320 | Spring | 08 | Knuth |
| 112 | MATH2410 | Fall | 08 | Chang |
| 119 | CS1310 | Fall | 08 | Anderson |
| 135 | CS3380 | Fall | 08 | Stone |

SQL QUERIES: -

1) Write appropriate MYQL DDL statements to define UNIVERSITY database.

- Create Database

- create database UNIVERSITY;
- use UNIVERSITY;

- Create Table: STUDENT

- create table STUDENT (Name varchar(10), Student_number int(2) primary key, Class int(2), Major varchar(20));

- Create Table: COURSE

- create table COURSE (Course_name varchar(30), Course_number varchar(20) primary key, Credit_hours int(2), Department varchar(15));

Table: GRADE_REPORT

| Student_number | Section_identifier | Grade |
|----------------|--------------------|-------|
| 8 | 85 | A |
| 8 | 92 | A |
| 8 | 102 | B |
| 8 | 135 | A |
| 17 | 112 | B |
| 17 | 119 | C |

Table: SECTION

| Section_identifier | Course_number | Semester | Year | Instructor |
|--------------------|---------------|----------|------|------------|
| 85 | MATH2410 | Fall | 7 | King |
| 92 | CS1310 | Fall | 7 | Anderson |
| 102 | CS3320 | Spring | 8 | Knuth |
| 112 | MATH2410 | Fall | 8 | Chang |
| 119 | CS1310 | Fall | 8 | Anderson |
| 135 | CS3380 | Fall | 8 | Stone |

Table: STUDENT

| Name | Student_number | Class | Major |
|-------|----------------|-------|-------|
| Brown | 8 | 2 | CS |
| Smith | 17 | 1 | CS |

Table: PREREQUISITE

| Course_number | Prerequisite_number |
|---------------|---------------------|
| CS3320 | CS1310 |
| CS3380 | CS3320 |

Table: COURSE

| Course_name | Course_number | Credit_hours | Department |
|---------------------------|---------------|--------------|------------|
| Intro to Computer Science | CS1310 | 4 | CS |
| Data Structures | CS3320 | 4 | CS |
| Database | CS3380 | 3 | CS |
| Discrete Mathamatics | MATH2410 | 3 | MATH |

- Create Table: SECTION

- create table SECTION (Section_identifier int(3) primary key, Course_number varchar(20), Semester varchar(10), Year int(2), Instructor varchar(15), foreign key (Course_number) references COURSE(Course_number));

- Create Table: GRADE_REPORT

- create table GRADE_REPORT (Student_number int(2), Section_identifier int(3), Grade varchar(2), foreign key (Student_number) references STUDENT(Student_number), foreign key (Section_identifier) references SECTION(Section_identifier), primary key (Student_number, Section_identifier));

- Create Table: PREREQUISITE

- create table PREREQUISITE (Course_number varchar(20), Prerequisite_number varchar(20), foreign key (Course_number) references COURSE(Course_number), foreign key (Prerequisite_number) references COURSE(Course_number), primary key (Course_number, Prerequisite_number));

2) Write queries to insert values in all the five tables.

- Inserting Data to Table: STUDENT

- insert into STUDENT values("Smith",17,1,"CS"),("Brown",8,2,"CS");

- Inserting Data to Table: COURSE

- insert into COURSE values("Intro to Computer Science","CS1310",4,"CS"),("Data Structures","CS3320",4,"CS"),("Discrete Mathamatics","MATH2410",3,"MATH"),("Database","CS3380",3,"CS");

- Inserting Data to Table: SECTION

- insert into SECTION values (85,"MATH2410","Fall",07,"King"),(92,"CS1310","Fall",07,"Anderson"),(102,"CS3320","Spring",08,"Knuth"),(112,"MATH2410","Fall",08,"Chang"),(119,"CS1310","Fall",08,"Anderson"),(135,"CS3380","Fall",08,"Stone");

- Inserting Data to Table: GRADE_REPORT

- insert into GRADE_REPORT values (17,112,"B"),(17,119,"C"),(8,85,"A"),(8,92,"A"),(8,102,"B"),(8,135,"A");

- Inserting Data to Table: PREREQUISITE

- insert into PREREQUISITE values("CS3380","CS3320"),("CS3380","MATH2410"),("CS3320","CS1310");

3) All Courses and grades of Smith

| Course_name | Grade |
|---------------------------|-------|
| Discrete Mathamatics | B |
| Intro to Computer Science | C |

4) Names and grades of students who took 'Database' course offered in fall 2008.

| Name | Grade |
|-------|-------|
| Brown | A |

5) Prerequisite for Database Course

| Course_name |
|----------------------|
| Data Structures |
| Discrete Mathamatics |

6) Senior Students

| Name |
|-------|
| Brown |

7) Courses taught by professor King in 2007 and 2008

| Course_name |
|----------------------|
| Discrete Mathamatics |

8) Details on section taught by King

9) Name and transcript of each senior majoring in CS.

| Name | Course_name | Course_number | Semester | Year | Grade |
|-------|---------------------------|---------------|----------|------|-------|
| Brown | Discrete Mathamatics | MATH2410 | Fall | 7 | A |
| Brown | Intro to Computer Science | CS1310 | Fall | 7 | A |
| Brown | Data Structures | CS3320 | Spring | 8 | B |
| Brown | Database | CS3380 | Fall | 8 | A |

| No_of_student |
|---------------|
| 1 |

3) Retrieve the list of all courses and grades of “Smith”.

- select c.Course_name, g.Grade from STUDENT s inner join GRADE_REPORT g on s.Student_number = g.Student_number inner join SECTION se on g.Section_identifier= se.Section_identifier inner join COURSE c on se.Course_number=c.Course_number where s.Name="Smith";

4) List the names of students who took the section of ‘Database’ course offered in fall 2008 and their grades in that section.

- select s.Name,g.Grade from STUDENT s join GRADE_REPORT g on s.Student_number = g.Student_number inner join SECTION se on g.Section_identifier= se.Section_identifier inner join COURSE c on se.Course_number=c.Course_number where c.Course_name="Database" and se.Semester="Fall" and se.Year=08 ;

5) List the prerequisites of the ‘Database’ course.

- select Course_name from COURSE where Course_number in (select p.Prerequisite_number from PREREQUISITE p join COURSE c on p.Course_number=c.Course_number where p.Course_number=(select Course_number from COURSE where Course_name="Database"));

6) Create a view to retrieve the names of all senior students majoring in ‘CS’ (computer science).

- create view seniors as select * from STUDENT where class=2;
- select Name from seniors;

7) Retrieve the names of all courses taught by Professor King in 2007 and 2008.

- select c.Course_name from COURSE c join SECTION s on c.Course_number = s.Course_number where s.Instructor="King";

8) For each section taught by Professor King, retrieve the course number, semester, year, and number of students who took the section.

- select s.Course_number,s.Semester,s.Year,count(g.Student_number) as No_of_students from SECTION s join GRADE_REPORT g on s.Section_identifier=g.Section_identifier where s.Instructor="King" group by g.Section_identifier;

9) Retrieve the name and transcript of each senior student (Class=2) majoring in CS. A transcript includes course name, course number, credit hours, semester, year and grade for each course completed by the student.

- select s.Name,c.Course_name,c.Course_number,se.Semester,se.Year,g.Grade from student s join grade_report g on s.Student_number = g.Student_number join section se on g.Section_identifier = se.Section_identifier join course c on se.Course_number = c.Course_number where s.Class=2 and s.Major="CS";

10 a) Insert new student Johnson

| Name | Student_number | Class | Major |
|---------|----------------|-------|-------|
| Johnson | 25 | 1 | Math |

10 b) Update class of Smith to 2

| Name | Student_number | Class | Major |
|-------|----------------|-------|-------|
| Smith | 17 | 2 | CS |

10 c) Inset new course Knowledge Engineering

| Course_name | Course_number | Credit_hours | Department |
|-----------------------|---------------|--------------|------------|
| Knowledge Engineering | CS4390 | 3 | CS |

10 d) Delete student Smith

| Name | Student_number | Class | Major |
|---------|----------------|-------|-------|
| Brown | 8 | 2 | CS |
| Johnson | 25 | 1 | Math |

10) Write SQL update statements to do the following on the database schema.

a) Insert a new student, < 'Johnson',25,1, 'Math'>, in the database.

- insert into STUDENT values("Johnson",25,1,"Math");
- select * from student where Student_number=25;

b) Change the class of student 'Smith' to 2.

- update STUDENT set Class=2 where Name="Smith";
- select * from student where Name="Smith";

c) Insert a new course, < 'Knowledge Engineering', 'CS4390', 3, 'CS'>.

- insert into COURSE values("Knowledge Engineering","CS4390",3,"CS");
- select * from course where Course_number="CS4390";

d) Delete the record for the student whose name is 'Smith' and whose student number is 17.

- delete from STUDENT where Student_name="Smith";
- select * from student;

Result: -

Queries are executed and output is verified.

1) Even or Odd

| Number | Result |
|--------|--------|
| 7 | odd |

2) Grade

| Marks | Grade |
|-------|------------|
| 87 | Grade is B |

EXERCISE 4: PLSQL

Date: 11/03/2024

PLSQL

Create **PLSQL procedures** and write appropriate **SQL queries** to call the procedures.

1.) Write a procedure to check if a number is even or odd.

Procedure

```
CREATE PROCEDURE `even_odd`(in num int)
BEGIN
    declare r varchar(10);
    if num%2=0
    then
        set r="even";
    else
        set r="odd";
    end if;
    select num as Number, r as Result;
END
```

SQL Query

➤ call even_odd(7);

2.) Write a procedure to display the grade for a mark.

Procedure

```
CREATE PROCEDURE `grade`(in g int)
BEGIN
    declare des varchar(50);
    if g > 90
    then
        set des="Grade is A";
    elseif g > 80
    then
        set des="Grade is B";
    elseif g > 70
```

then

set des="Grade is C";

elseif $g > 60$

then

set des="Grade is D";

3) Positive, Negative or Zero

| Number | Result |
|--------|----------|
| -7 | negative |

4) WeekDay

| Date | WeekDay |
|------------|---------|
| 03/16/2024 | 5 |

```
else
    set des="Grade is E";
end if;
select g as Marks,des as Grade;
END
```

SQL Query

➤ call grade(87);

3.) Write a procedure to check if a number is positive, negative or zero.

Procedure

```
CREATE PROCEDURE `pos_neg_zero`(in num int)
BEGIN
    declare r varchar(10);
    if num >0
    then
        set r= "positive";
    elseif num<0
    then
        set r= "negative";
    else
        set r= "zero";
    end if;
    select num as Number, r as Result;
END
```

SQL Query

➤ call pos_neg_zero (-7);

4.) Write a procedure to display the weekday of a specific day.

Procedure

```
CREATE PROCEDURE `day`(in d date)
BEGIN
    declare dat varchar(10);
```

```
set dat=Weekday(d);  
select d as Date, dat as WeekDay;  
END
```

SQL Query

➤ call day("2024-3-16");

5) Factorial

| Number | Factorial |
|--------|-----------|
| 7 | 5040 |

5.) Write a procedure to find factorial of a number.

Procedure

```
CREATE PROCEDURE `fact`(in num int)
BEGIN
    declare f int default 1;
    declare n1 int;
    set n1=num;
    while num >0
    do
        set f=f*num;
        set num=num-1;
    end while;
    select n1 as Number, f as Factorial;
END
```

SQL Query

➤ call fact(7);

Result: -

Procedure and queries are executed and output is verified.

Table structure: Product

| Field | Type | Null | Key | Default | Extra |
|----------|-------------|------|-----|---------|-------|
| PdtId | varchar(4) | NO | PRI | NULL | |
| PName | varchar(20) | YES | | NULL | |
| Price | double(5,2) | YES | | NULL | |
| Qunatity | int | YES | | NULL | |

Inserted Values

| PdtId | PName | Price | Qunatity |
|-------|--------|-------|----------|
| 1001 | Pen | 10.5 | 10 |
| 1002 | Pencil | 5.75 | 15 |
| 1003 | Book | 25 | 5 |

EXERCISE 5: Stored Procedure

Date: 14/03/2024

Create a database **STORE** with table named **Product**(PdtId, PName, Price, Quantity) and create a stored procedure called **Insertproduct** that inserts a new product into the database, under some conditions. The stored procedure has input parameters (barcode, product name, price, and quantityInStock). The shared procedure should insert a row in the Product table only if the price is greater than 0 and the quantity is greater or equal to 0. If the conditions are not satisfied, the stored procedure just terminates (no errors generated).

SQL Query to create database and table:

- Create Database

- CREATE DATABASE STORE;
- USE STORE;

- Create Table: Product

- CREATE TABLE Product(PdtId varchar(4) primary key,PName varchar(20), Price double(5,2),Qunatity int(3));

Procedure to insert values to the table:

```
CREATE PROCEDURE `Insertproduct` (in barcode int,in name varchar(20),in price  
double(5,2), in quantity int(3))
```

```
BEGIN
```

```
    declare Result varchar(20);
```

```
    if price>0 and quantity >0
```

```
    then
```

```
        insert into Product values (barcode,name,price,quantity);
```

```
    else
```

```
        set Result= "Row Not inserted";
```

```
        select Result;
```

```
    end if;
```

```
END
```

SQL Query to call the procedure:

- call Insertproduct(1001,"Pen",10.5,10);
- call Insertproduct(1002,"Pencil",5.75,15);
- call Insertproduct(1003,"Book",25,5);

➤ `select * from Product;`

Invalid Values

Result

Row Not inserted

- Inserting Invalid Values

- call Insertproduct(1004,"Eraser",-2,20);
- call Insertproduct(1005,"Paper",1.25,-10);
- call Insertproduct(1006,"Sharpner",-2.5,-20);

Result: -

Procedure and queries are executed and output is verified.

Table structure: Product

| Field | Type | Null | Key | Default | Extra |
|------------|-------------|------|-----|---------|-------|
| PdtId | int | NO | PRI | NULL | |
| Pname | varchar(20) | YES | | NULL | |
| Price | double(3,2) | YES | | NULL | |
| Qtyinstock | int | YES | | NULL | |

Table structure: Sale

| Field | Type | Null | Key | Default | Extra |
|-----------------|-------------|------|-----|---------|-------|
| saleId | int | NO | PRI | NULL | |
| Deliveryaddress | varchar(50) | YES | | NULL | |

Table structure: Saleitem

| Field | Type | Null | Key | Default | Extra |
|--------|------|------|-----|---------|-------|
| saleId | int | NO | PRI | NULL | |
| PdtId | int | NO | PRI | NULL | |
| Qty | int | YES | | NULL | |

Table: Product

| PdtId | Pname | Price | Qtyinstock |
|-------|--------|-------|------------|
| 101 | pencil | 5 | 10 |
| 102 | pen | 3 | 10 |

Table: Product

| saleId | Deliveryaddress |
|--------|-----------------|
| 1 | home123 |
| 2 | home345 |

EXERCISE 6: Triggers

Date: 02/04/2024

Triggers

Create three tables named:

- Product (PdtId, Pname, Price, Qtyinstock)
- Sale(saleId, Deliveryaddress)
- Saleitem(saleId, PdtId, Qty)

Create a trigger called updateAvailabeQuantity that updates the quantity in stock in the product table, for every product sold. The trigger should be executed after each insert operation on the Saleitem table: for the product with given PdtId(the one inserted into saleitem), update the available quantity in Product table to old quantity minus sold quantity.

SQL Query to create database and table:

- Create Database

- CREATE DATABASE stores;
- USE stores;

- Create Table: Product

- CREATE TABLE Product(PdtId INT PRIMARY KEY, Pname VARCHAR(20), Price DOUBLE(3,2),Qtyinstock INT);

- Create Table: Sale

- CREATE TABLE Sale(saleId INT PRIMARY KEY, Deliveryaddress VARCHAR(50));

- Create Table: Saleitem

- CREATE TABLE Saleitem(saleId INT, PdtId INT, Qty INT, FOREIGN KEY (saleId) REFERENCES Sale(saleId), FOREIGN KEY (PdtId) REFERENCES Product(PdtId) ,PRIMARY KEY(saleId,PdtId));

- Inserting Data to Table: Product

- INSERT INTO Product (PdtId, Pname, Price, Qtyinstock) VALUES (101,"pencil", 5, 10), (102,"pen", 3, 10);

- Inserting Data to Table: Sale

- INSERT INTO Sale(saleId,Deliveryaddress) VALUES(001,"home123"),(002,"home345");

Product Table after inserting 1st row in Saleitem

| PdtId | Pname | Price | Qtyinstock |
|-------|--------|-------|------------|
| 101 | pencil | 5 | 10 |
| 102 | pen | 3 | 2 |

Product Table after inserting 2nd row in Saleitem

| PdtId | Pname | Price | Qtyinstock |
|-------|--------|-------|------------|
| 101 | pencil | 5 | 7 |
| 102 | pen | 3 | 2 |

Product Table after inserting 3rd row in Saleitem

| PdtId | Pname | Price | Qtyinstock |
|-------|--------|-------|------------|
| 101 | pencil | 5 | 7 |
| 102 | pen | 3 | 1 |

Table: Saleitem

| saleId | PdtId | Qty |
|--------|-------|-----|
| 1 | 102 | 8 |
| 2 | 101 | 3 |
| 2 | 102 | 1 |

- Trigger updateAvailabeQuantity

CREATE TRIGGER updateAvailabeQuantity

AFTER INSERT ON Saleitem

FOR EACH ROW

UPDATE Product SET Qtyinstock=Product.Qtyinstock-new.Qty WHERE PdtId=new.PdtId;

- Inserting Data to Table: Saleitem

1. INSERT INTO Saleitem VALUES (001,102,8);
2. INSERT INTO Saleitem VALUES (002,101,3);
3. INSERT INTO Saleitem VALUES (002,102,1);

Result: -

Triggers and queries are executed and output is verified.

1) Name and marks of all female students in MCA.

Name: Athira Krishnan
Mark: 80

Name: Divya Vijayan
Mark: 70

Name: Renuka Vijayan
Mark: 82

Name: Remya V
Mark: 85

Name: Remya Sugunan
Mark: 72

Name: Vidhya Sugunan
Mark: 79

Name: Soorya S
Mark: 79

Name: Amritha S
Mark: 99

Name: Soorya P
Mark: 74

2) Student with highest mark in course MCA.

```
_id: 20
name: {'fname': 'Amritha', 'lname': 'S'}
address: {'house_name': 'Arya Bhavan', 'city': 'Varkala'}
gender: female
course: MCA
mark: 99
grade: A+
phone: {'type': 'mobile', 'no': 9445365787}
```

3) Male students with A+ grade.

Vimal Vinayan
Vimal Bose
Arun S

EXERCISE 7: MongoDB

Date: 04/03/2025

MongoDB

Create a database named **college** and then create a collection named **studlist**. Insert values to the collection.

MongoDB QUERIES: -

- Create database and collection:

```
import pymongo
conn=pymongo.MongoClient("mongodb://localhost:27017/")
db=conn['college']
col=db['studlist']
```

- Insert values to the collection:

```
import json
with open("data.json") as file:
    data=json.load(file)
db.studlist.insert_many(data)
```

1) Display name (both fname and lname) and mark of all female students in MCA.

```
x=db.studlist.find({"gender":"female","course":"MCA"},{"name":1,"mark":1})
for i in x:
    print("\nName: "+i['name']['fname']+" "+i['name']['lname']+"\nMark: ",i['mark'])
```

2) Display the details of student who secured highest mark in the course MCA.

```
x=db.studlist.find({"course":"MCA"}).sort("mark",-1).limit(1)
for i in x:
    for j in i.keys():
        print(j+": ",i[j])
```

3) Display all male students who scored A+ grade.

```
x=db.studlist.find({"grade":"A+","gender":"male"})
```

for i **in** x:

```
print(i['name']['fname']+" "+i['name']['lname'])
```

4) Top 3 students in Mechanical Department.

Kavya Mohan
Vimal Vinayan
Yadu Kannan

5) Female students with marks more than 90.

Name: Kavya Mohan
Grade: A+
Marks: 95
Contact no: 9448399780

Name: Amritha S
Grade: A+
Marks: 99
Contact no: 9445365787

6) Students with marks in between 80 and 90.

Name: Vidhya S
Grade: A
Marks: 85
Contact no: 8146321420

Name: Yadu Kannan
Grade: A
Marks: 85
Contact no: 9446321780

Name: Renuka Vijayan
Grade: A
Marks: 82
Contact no: 04712547890

Name: Remya V
Grade: A
Marks: 85
Contact no: 04712647890

7) Students with name starting with V.

Name: Varun Nair
Course: MCA
Grade: B+
Marks: 70
Contact no: 04712662690

Name: Vidhya S
Course: Civil
Grade: A
Marks: 85
Contact no: 8146321420

Name: Vivek Bose
Course: MCA
Grade: B
Marks: 60
Contact no: 04842663890

Name: Vimal Vinayan
Course: Mechanical
Grade: A+
Marks: 90
Contact no: 8185399780
...

8) Students from Kollam.

Athira Krishnan
Yadu Kannan
Kavya Mohan
Vimal Vinayan
Jabin S
Arya Satheesh

9) Students neither from Kollam nor Thiruvananthapuram

Name: Arya S
City: Varkala

Name: Vidhya S
City: Kadakkavoor

Name: Vivek Bose
City: Ernakulam

Name: Divya Vijayan
City: Varkala

Name: Vimal Bose
City: Ernakulam

Name: Vinod Paniker
City: Ernakulam

Name: Amritha S
City: Varkala

Name: Arun S
City: Attingal
...

4) Display the names of top three students in Mechanical department.

```
x=db.studlist.find({"course":"Mechanical"}).sort("mark",-1).limit(3)
for i in x:
    print(i['name']['fname']+" "+i['name']['lname'])
```

5) Display the details of female students [fname, lname, grade, mark, contact] who achieved a mark more than 90.

```
x=db.studlist.find({"mark":{"$gt":90},"gender":"female"})
for i in x:
    print("\nName: "+i['name']['fname']+" "+i['name']['lname']+" \n Grade: "+i['grade']+"\nMarks: ",i['mark'],"\nContact no: ", i['phone']['no'])
```

6) Display the details of students who secured mark, more than 80 but less than 90.

```
x=db.studlist.find({"mark":{"$gt":80,"$lt":90}})
for i in x:
    print("\nName: "+i['name']['fname']+" "+i['name']['lname']+" \n Grade: "+i['grade']+"\nMarks: ",i['mark'],"\nContact no: ", i['phone']['no'])
```

7) Display the details of students whose name starts with 'V'.

```
x=db.studlist.find({"name.fname":{"$regex":"^V"}})
for i in x:
    print("\nName: "+i['name']['fname']+" "+i['name']['lname']+" \n Course: "+i['course']+"\nGrade: "+i['grade']+"\nMarks: ", i['mark'],"\nContact no: ",i['phone']['no'])
```

8) Display all students from Kollam.

```
x=db.studlist.find({"address.city":"Kollam"})
for i in x:
    print(i['name']['fname']+" "+i['name']['lname'])
```

9) Display all students who does not belong to neither Kollam nor Thiruvananthapuram.

```
x=db.studlist.find({"address.city":{"$nin":["Kollam","Thiruvananthapuram"]}})
for i in x:
```

```
print("\nName: "+i['name']['fname']+" "+i['name']['lname']+"\nCity: "+ i["address"]["city"])
```

10) Female students from Kollam or Thiruvananthapuram.

Name: Athira Krishnan
City: Kollam

Name: Kavya Mohan
City: Kollam

Name: Renuka Vijayan
City: Thiruvananthapuram

Name: Remya V
City: Thiruvananthapuram

Name: Remya Sugunan
City: Thiruvananthapuram

Name: Vidhya Sugunan
City: Thiruvananthapuram

Name: Arya Satheesh
City: Kollam

Name: Soorya S
City: Thiruvananthapuram

Name: Soorya P
City: Thiruvananthapuram

10) Display all female students who belong to either Kollam or Thiruvananthapuram

```
x=db.studlist.find({"gender":"female","address.city":{"$in":
                        ['Kollam','Thiruvananthapuram']}))
for i in x:
    print("\nName: "+i['name']['fname']+" "+i['name']['lname']+ "\nCity: "+i["address"]
        ["city"])
```

Result: -

Queries are executed and output is verified.

