MySQL Create Table Exercises

1. Write a SQL statement to create a simple table countries including columns country_id,country_name and region_id.

2. Write a SQL statement to create a simple table countries including columns country_id,country_name and region_id which is already exists.

3. Write a SQL statement to create the structure of a table dup_countries similar to countries.

4. Write a SQL statement to create a duplicate copy of countries table including structure and data by name dup_countries.

5. Write a SQL statement to create a table countries set a constraint NULL.

6. Write a SQL statement to create a table named jobs including columns job_id, job_title, min_salary, max_salary and check whether the max_salary amount exceeding the upper limit 25000.

7. Write a SQL statement to create a table named countries including columns country_id, country_name and region_id and make sure that no countries except Italy, India and China will be entered in the table.

8. Write a SQL statement to create a table named job_histry including columns employee_id, start_date, end_date, job_id and department_id and make sure that the value against column end_date will be entered at the time of insertion to the format like '--/--/----'.

9. Write a SQL statement to create a table named countries including columns country_id,country_name and region_id and make sure that no duplicate data against column country_id will be allowed at the time of insertion.

10. Write a SQL statement to create a table named jobs including columns job_id, job_title, min_salary and max_salary, and make sure that, the default value for job_title is blank and min_salary is 8000 and max_salary is NULL will be entered automatically at the time of insertion if no value assigned for the specified columns.

11. Write a SQL statement to create a table named countries including columns country_id, country_name and region_id and make sure that the country_id column will be a key field which will not contain any duplicate data at the time of insertion.

12. Write a SQL statement to create a table countries including columns country_id, country_name and region_id and make sure that the column country_id will be unique and store an auto incremented value.

13. Write a SQL statement to create a table countries including columns country_id, country_name and region_id and make sure that the combination of columns country_id and region_id will be unique.

14. Write a SQL statement to create a table job_history including columns employee_id, start_date, end_date, job_id and department_id and make sure that, the employee_id column does not contain any duplicate value at the time of insertion and the foreign key column job_id contain only those values which are exists in the jobs table.
Here is the structure of the table jobs;

```
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| JOB_ID     | varchar(10)  | NO   | PRI |         |       |
| JOB_TITLE  | varchar(35)  | NO   |     | NULL    |       |
| MIN_SALARY | decimal(6,0) | YES  |     | NULL    |       |
| MAX_SALARY | decimal(6,0) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
```

15. Write a SQL statement to create a table employees including columns employee_id, first_name, last_name, email, phone_number hire_date, job_id, salary, commission, manager_id and department_id and make sure that, the employee_id column does not contain any duplicate value at the time of insertion and the foreign key columns combined by department_id and manager_id columns contain only those unique combination values, which combinations are exists in the departments table.
Assume the structure of departments table below.

```
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| DEPARTMENT_ID   | decimal(4,0) | NO   | PRI | 0       |       |
| DEPARTMENT_NAME | varchar(30)  | NO   |     | NULL    |       |
| MANAGER_ID      | decimal(6,0) | NO   | PRI | 0       |       |
| LOCATION_ID     | decimal(4,0) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

16. Write a SQL statement to create a table employees including columns employee_id, first_name, last_name, email, phone_number hire_date, job_id, salary, commission, manager_id and department_id and make sure that, the employee_id column does not contain any duplicate value at the time of insertion, and the foreign key column department_id, reference by the column department_id of departments table, can contain only those values which are exists in the departments table and another foreign key column job_id, referenced by the column job_id of jobs table, can contain only those values which are exists in the jobs table. The InnoDB Engine have been used to create the tables.
"A foreign key constraint is not required merely to join two tables. For storage engines other than InnoDB, it is possible when defining a column to use a REFERENCES tbl_name(col_name) clause, which has no actual effect, and serves only as a memo or

comment to you that the column which you are currently defining is intended to refer to a column in another table."

Assume that the structure of two tables departments and jobs.

```
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| DEPARTMENT_ID   | decimal(4,0) | NO   | PRI | 0       |       |
| DEPARTMENT_NAME | varchar(30)  | NO   |     | NULL    |       |
| MANAGER_ID      | decimal(6,0) | YES  |     | NULL    |       |
| LOCATION_ID     | decimal(4,0) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
+------------+-------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| JOB_ID     | varchar(10)  | NO   | PRI |         |       |
| JOB_TITLE  | varchar(35)  | NO   |     | NULL    |       |
| MIN_SALARY | decimal(6,0) | YES  |     | NULL    |       |
| MAX_SALARY | decimal(6,0) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

17. Write a SQL statement to create a table employees including columns employee_id, first_name, last_name, job_id, salary and make sure that, the employee_id column does not contain any duplicate value at the time of insertion, and the foreign key column job_id, referenced by the column job_id of jobs table, can contain only those values which are exists in the jobs table. The InnoDB Engine have been used to create the tables. The specialty of the statement is that, The ON UPDATE CASCADE action allows you to perform cross-table update and ON DELETE RESTRICT action reject the deletion. The default action is ON DELETE RESTRICT.

Assume that the structure of the table jobs and InnoDB Engine have been used to create the table jobs.

```
CREATE TABLE IF NOT EXISTS jobs (
JOB_ID integer NOT NULL UNIQUE PRIMARY KEY,
JOB_TITLE varchar(35) NOT NULL DEFAULT ' ',
MIN_SALARY decimal(6,0) DEFAULT 8000,
MAX_SALARY decimal(6,0) DEFAULT NULL
)ENGINE=InnoDB;
```

```
+------------+-------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| JOB_ID     | int(11)      | NO   | PRI | NULL    |       |
| JOB_TITLE  | varchar(35)  | NO   |     |         |       |
| MIN_SALARY | decimal(6,0) | YES  |     | 8000    |       |
| MAX_SALARY | decimal(6,0) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

18. Write a SQL statement to create a table employees including columns employee_id, first_name, last_name, job_id, salary and make sure that, the employee_id column does not contain any duplicate value at the time of insertion, and the foreign key column job_id, referenced by the column job_id of jobs table, can contain only those values which are exists in the jobs table. The InnoDB Engine have been used to create the tables. The specialty of the statement is that, The ON DELETE CASCADE that lets you allow to delete records in the employees(child) table that refer to a record in the jobs(parent) table when the record in the parent table is deleted and the ON UPDATE RESTRICT actions reject any updates.

Assume that the structure of the table jobs and InnoDB Engine have been used to create the table jobs.

```
CREATE TABLE IF NOT EXISTS jobs (
JOB_ID integer NOT NULL UNIQUE PRIMARY KEY,
JOB_TITLE varchar(35) NOT NULL DEFAULT ' ',
MIN_SALARY decimal(6,0) DEFAULT 8000,
MAX_SALARY decimal(6,0) DEFAULT NULL
)ENGINE=InnoDB;
```

```
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| JOB_ID     | int(11)      | NO   | PRI | NULL    |       |
| JOB_TITLE  | varchar(35)  | NO   |     |         |       |
| MIN_SALARY | decimal(6,0) | YES  |     | 8000    |       |
| MAX_SALARY | decimal(6,0) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
```

19. Write a SQL statement to create a table employees including columns employee_id, first_name, last_name, job_id, salary and make sure that, the employee_id column does not contain any duplicate value at the time of insertion, and the foreign key column job_id, referenced by the column job_id of jobs table, can contain only those values which are exists in the jobs table. The InnoDB Engine have been used to create the tables. The specialty of the statement is that, The ON DELETE SET NULL action will set the foreign key column values in the child table(employees) to NULL when the record in the parent table(jobs) is deleted, with a condition that the foreign key column in the child table must accept NULL values and the ON UPDATE SET NULL action resets the values in the rows in the child table(employees) to NULL values when the rows in the parent table(jobs) are updated.

Assume that the structure of two table jobs and InnoDB Engine have been used to create the table jobs.

```
CREATE TABLE IF NOT EXISTS jobs (
JOB_ID integer NOT NULL UNIQUE PRIMARY KEY,
JOB_TITLE varchar(35) NOT NULL DEFAULT ' ',
MIN_SALARY decimal(6,0) DEFAULT 8000,
MAX_SALARY decimal(6,0) DEFAULT NULL
)ENGINE=InnoDB;
```

```
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| JOB_ID     | int(11)      | NO   | PRI | NULL    |       |
| JOB_TITLE  | varchar(35)  | NO   |     |         |       |
| MIN_SALARY | decimal(6,0) | YES  |     | 8000    |       |
| MAX_SALARY | decimal(6,0) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
```

20. Write a SQL statement to create a table employees including columns employee_id, first_name, last_name, job_id, salary and make sure that, the employee_id column does not contain any duplicate value at the time of insertion, and the foreign key column job_id, referenced by the column job_id of jobs table, can contain only those values which are exists in the jobs table. The InnoDB Engine have been used to create the tables. The specialty of the statement is that, The ON DELETE NO ACTION and the ON UPDATE NO ACTION actions will reject the deletion and any updates.
Assume that the structure of two table jobs and InnoDB Engine have been used to create the table jobs.

```
CREATE TABLE IF NOT EXISTS jobs (
JOB_ID integer NOT NULL UNIQUE PRIMARY KEY,
JOB_TITLE varchar(35) NOT NULL DEFAULT ' ',
MIN_SALARY decimal(6,0) DEFAULT 8000,
MAX_SALARY decimal(6,0) DEFAULT NULL
)ENGINE=InnoDB;
```

```
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| JOB_ID     | int(11)      | NO   | PRI | NULL    |       |
| JOB_TITLE  | varchar(35)  | NO   |     |         |       |
| MIN_SALARY | decimal(6,0) | YES  |     | 8000    |       |
| MAX_SALARY | decimal(6,0) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
```

Create-Table-Exercises Solutions

1.

```
CREATE TABLE countries(
COUNTRY_ID varchar(2),
COUNTRY_NAME varchar(40),
REGION_ID decimal(10,0)
);
```

2.

```
CREATE TABLE IF NOT EXISTS countries (
COUNTRY_ID varchar(2),
COUNTRY_NAME varchar(40) ,
REGION_ID decimal(10,0)
);
```

3.

```
CREATE TABLE IF NOT EXISTS dup_countries
LIKE countries;
```

4.

```
CREATE TABLE IF NOT EXISTS dup_countries
AS SELECT * FROM  countries;
```

5.

```
CREATE TABLE IF NOT EXISTS countries (
COUNTRY_ID varchar(2) NOT NULL,
COUNTRY_NAME varchar(40) NOT NULL,
REGION_ID decimal(10,0) NOT NULL
);
```

6.

```
CREATE TABLE IF NOT EXISTS jobs (
JOB_ID varchar(10) NOT NULL ,
JOB_TITLE varchar(35) NOT NULL,
MIN_SALARY decimal(6,0),
MAX_SALARY decimal(6,0)
CHECK(MAX_SALARY<=25000)
);
```

7.

```
CREATE TABLE IF NOT EXISTS countries (
COUNTRY_ID varchar(2),
COUNTRY_NAME varchar(40)
CHECK(COUNTRY_NAME IN('Italy','India','China')) ,
REGION_ID decimal(10,0)
);
```

8.

```
CREATE TABLE IF NOT EXISTS job_history (
EMPLOYEE_ID decimal(6,0) NOT NULL,
START_DATE date NOT NULL,
END_DATE date NOT NULL
CHECK (END_DATE LIKE '--/--/----'),
JOB_ID varchar(10) NOT NULL,
DEPARTMENT_ID decimal(4,0) NOT NULL
);
```

9.

```
CREATE TABLE IF NOT EXISTS countries (
COUNTRY_ID varchar(2) NOT NULL,
COUNTRY_NAME varchar(40) NOT NULL,
REGION_ID decimal(10,0) NOT NULL,
UNIQUE(COUNTRY_ID)
);
```

10.

```
CREATE TABLE IF NOT EXISTS jobs (
JOB_ID varchar(10) NOT NULL UNIQUE,
JOB_TITLE varchar(35) NOT NULL DEFAULT ' ',
MIN_SALARY decimal(6,0) DEFAULT 8000,
MAX_SALARY decimal(6,0) DEFAULT NULL
);
```

11.

```
CREATE TABLE IF NOT EXISTS countries (
COUNTRY_ID varchar(2) NOT NULL UNIQUE PRIMARY KEY,
COUNTRY_NAME varchar(40) NOT NULL,
REGION_ID decimal(10,0) NOT NULL
);
```

12.

```
CREATE TABLE IF NOT EXISTS countries (
COUNTRY_ID integer NOT NULL UNIQUE AUTO_INCREMENT PRIMARY KEY,
COUNTRY_NAME varchar(40) NOT NULL,
REGION_ID decimal(10,0) NOT NULL
);
```

13.

```
CREATE TABLE IF NOT EXISTS countries (
COUNTRY_ID varchar(2) NOT NULL UNIQUE DEFAULT '',
COUNTRY_NAME varchar(40) DEFAULT NULL,
REGION_ID decimal(10,0) NOT NULL,
PRIMARY KEY (COUNTRY_ID,REGION_ID));
```

14.

```
CREATE TABLE job_history (
EMPLOYEE_ID decimal(6,0) NOT NULL PRIMARY KEY,
START_DATE date NOT NULL,
END_DATE date NOT NULL,
JOB_ID varchar(10) NOT NULL,
DEPARTMENT_ID decimal(4,0) DEFAULT NULL,
FOREIGN KEY (job_id) REFERENCES jobs(job_id)
)ENGINE=InnoDB;
```

15.

```
CREATE TABLE IF NOT EXISTS employees (
EMPLOYEE_ID decimal(6,0) NOT NULL PRIMARY KEY,
FIRST_NAME varchar(20) DEFAULT NULL,
LAST_NAME varchar(25) NOT NULL,
EMAIL varchar(25) NOT NULL,
PHONE_NUMBER varchar(20) DEFAULT NULL,
HIRE_DATE date NOT NULL,
JOB_ID varchar(10) NOT NULL,
SALARY decimal(8,2) DEFAULT NULL,
COMMISSION_PCT decimal(2,2) DEFAULT NULL,
MANAGER_ID decimal(6,0) DEFAULT NULL,
DEPARTMENT_ID decimal(4,0) DEFAULT NULL,
FOREIGN KEY(DEPARTMENT_ID,MANAGER_ID)
REFERENCES  departments(DEPARTMENT_ID,MANAGER_ID)
)ENGINE=InnoDB;
```

16.

```
CREATE TABLE IF NOT EXISTS employees (
EMPLOYEE_ID decimal(6,0) NOT NULL PRIMARY KEY,
FIRST_NAME varchar(20) DEFAULT NULL,
LAST_NAME varchar(25) NOT NULL,
EMAIL varchar(25) NOT NULL,
PHONE_NUMBER varchar(20) DEFAULT NULL,
HIRE_DATE date NOT NULL,
JOB_ID varchar(10) NOT NULL,
SALARY decimal(8,2) DEFAULT NULL,
COMMISSION_PCT decimal(2,2) DEFAULT NULL,
MANAGER_ID decimal(6,0) DEFAULT NULL,
DEPARTMENT_ID decimal(4,0) DEFAULT NULL,
FOREIGN KEY(DEPARTMENT_ID)
REFERENCES  departments(DEPARTMENT_ID),
FOREIGN KEY(JOB_ID)
REFERENCES  jobs(JOB_ID)
)ENGINE=InnoDB;
```

17.

```
CREATE TABLE IF NOT EXISTS employees (
EMPLOYEE_ID decimal(6,0) NOT NULL PRIMARY KEY,
FIRST_NAME varchar(20) DEFAULT NULL,
LAST_NAME varchar(25) NOT NULL,
JOB_ID INTEGER NOT NULL,
SALARY decimal(8,2) DEFAULT NULL,
FOREIGN KEY(JOB_ID)
REFERENCES  jobs(JOB_ID)
ON UPDATE CASCADE ON DELETE RESTRICT
)ENGINE=InnoDB;
```

18.

```
CREATE TABLE IF NOT EXISTS employees (
EMPLOYEE_ID decimal(6,0) NOT NULL PRIMARY KEY,
FIRST_NAME varchar(20) DEFAULT NULL,
LAST_NAME varchar(25) NOT NULL,
JOB_ID INTEGER NOT NULL,
SALARY decimal(8,2) DEFAULT NULL,
FOREIGN KEY(JOB_ID)
REFERENCES  jobs(JOB_ID)
ON DELETE CASCADE ON UPDATE RESTRICT
)ENGINE=InnoDB;
```

19.

```
CREATE TABLE IF NOT EXISTS employees (
EMPLOYEE_ID decimal(6,0) NOT NULL PRIMARY KEY,
FIRST_NAME varchar(20) DEFAULT NULL,
LAST_NAME varchar(25) NOT NULL,
JOB_ID INTEGER,
SALARY decimal(8,2) DEFAULT NULL,
FOREIGN KEY(JOB_ID)
REFERENCES  jobs(JOB_ID)
ON DELETE SET NULL
ON UPDATE SET NULL
)ENGINE=InnoDB;
```

20.

```
CREATE TABLE IF NOT EXISTS employees (
EMPLOYEE_ID decimal(6,0) NOT NULL PRIMARY KEY,
FIRST_NAME varchar(20) DEFAULT NULL,
LAST_NAME varchar(25) NOT NULL,
JOB_ID INTEGER NOT NULL,
SALARY decimal(8,2) DEFAULT NULL,
FOREIGN KEY(JOB_ID)
REFERENCES  jobs(JOB_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
)ENGINE=InnoDB;
```

MySQL Insert Rows into the Table

1. Write a SQL statement to insert a record with your own value into the table countries against each columns.nd region_id.

Here in the following is the structure of the table countries.

```
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| COUNTRY_ID   | varchar(2)    | YES  |     | NULL    |       |
| COUNTRY_NAME | varchar(40)   | YES  |     | NULL    |       |
| REGION_ID    | decimal(10,0) | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
```

2. Write a SQL statement to insert one row into the table countries against the column country_id and country_name.

Here in the following is the structure of the table countries.

```
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| COUNTRY_ID   | varchar(2)    | YES  |     | NULL    |       |
| COUNTRY_NAME | varchar(40)   | YES  |     | NULL    |       |
| REGION_ID    | decimal(10,0) | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
```

3. Write a SQL statement to create duplicate of countries table named country_new with all structure and data.

Here in the following is the structure of the table countries.

```
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| COUNTRY_ID   | varchar(2)    | YES  |     | NULL    |       |
| COUNTRY_NAME | varchar(40)   | YES  |     | NULL    |       |
| REGION_ID    | decimal(10,0) | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
```

4. Write a SQL statement to insert NULL values against region_id column for a row of countries table.

5.Write a SQL statement to insert 3 rows by a single insert statement.

6. Write a SQL statement insert rows from country_new table to countries table. Here is the rows for country_new table. Assume that, the countries table is empty.

```
+------------+--------------+-----------+
| COUNTRY_ID | COUNTRY_NAME | REGION_ID |
+------------+--------------+-----------+
| C0001      | India        |      1001 |
| C0002      | USA          |      1007 |
| C0003      | UK           |      1003 |
+------------+--------------+-----------+
```

7.Write a SQL statement to insert one row in jobs table to ensure that no duplicate value will be entered in the job_id column.

8.Write a SQL statement to insert one row in jobs table to ensure that no duplicate value will be entered in the job_id column.

9. Write a SQL statement to insert a record into the table countries to ensure that, a country_id and region_id combination will be entered once in the table.

10. Write a SQL statement to insert rows into the table countries in which the value of country_id column will be unique and auto incremented.

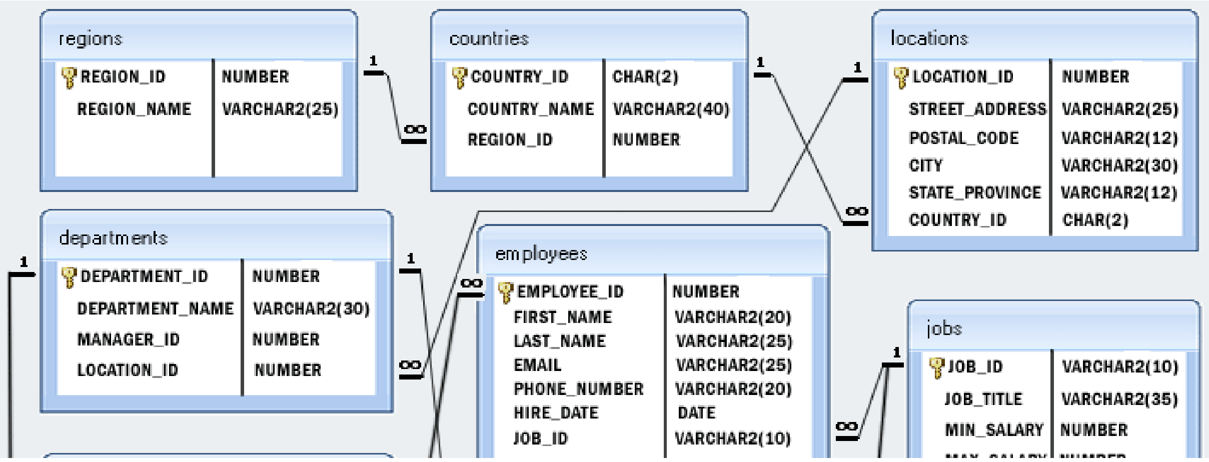
11. Write a SQL statement to insert records into the table countries to ensure that the country_id column will not contain any duplicate data and this will be automatically incremented and the column country_name will be filled up by 'N/A' if no value assigned for that column.

12. Write a SQL statement to insert rows in the job_history table in which one column job_id is containing those values which are exists in job_id column of jobs table.

13. Write a SQL statement to insert rows into the table employees in which a set of columns department_id and manager_id contains a unique value and that combined values must have exists into the table departments.

14. Write a SQL statement to insert rows into the table employees in which a set of columns department_id and job_id contains the values which must have exists into the table departments and jobs.

Structure of 'hr' database :



regions

REGION_ID	NUMBER
REGION_NAME	VARCHAR2(25)

countries

COUNTRY_ID	CHAR(2)
COUNTRY_NAME	VARCHAR2(40)
REGION_ID	NUMBER

locations

LOCATION_ID	NUMBER
STREET_ADDRESS	VARCHAR2(25)
POSTAL_CODE	VARCHAR2(12)
CITY	VARCHAR2(30)
STATE_PROVINCE	VARCHAR2(12)
COUNTRY_ID	CHAR(2)

departments

DEPARTMENT_ID	NUMBER
DEPARTMENT_NAME	VARCHAR2(30)
MANAGER_ID	NUMBER
LOCATION_ID	NUMBER

employees

EMPLOYEE_ID	NUMBER
FIRST_NAME	VARCHAR2(20)
LAST_NAME	VARCHAR2(25)
EMAIL	VARCHAR2(25)
PHONE_NUMBER	VARCHAR2(20)
HIRE_DATE	DATE
JOB_ID	VARCHAR2(10)

jobs

JOB_ID	VARCHAR2(10)
JOB_TITLE	VARCHAR2(35)
MIN_SALARY	NUMBER
MAX_SALARY	NUMBER

Insert-Into-Statement-Exercise Solutions

1.

```
INSERT INTO countries VALUES('C1','India',1001);
```

2.

```
INSERT INTO countries (country_id,country_name) VALUES('C1','India');
```

3.

```
CREATE TABLE IF NOT EXISTS country_new
AS SELECT * FROM countries;
```

4.

```
INSERT INTO countries (country_id,country_name,region_id) VALUES('C1','India',NULL);
```

5.

```
INSERT INTO countries VALUES('C0001','India',1001),
('C0002','USA',1007),('C0003','UK',1003);
```

6.

```
INSERT INTO countries
SELECT * FROM country_new;
```

7.

Create the table jobs.

```
CREATE TABLE IF NOT EXISTS jobs (
JOB_ID integer NOT NULL UNIQUE ,
JOB_TITLE varchar(35) NOT NULL,
MIN_SALARY decimal(6,0)
);

INSERT INTO jobs VALUES(1001,'OFFICER',8000);

mysql> SELECT * FROM JOBS;
+--------+-----------+------------+
| JOB_ID | JOB_TITLE | MIN_SALARY |
+--------+-----------+------------+
|   1001 | OFFICER   |       8000 |
+--------+-----------+------------+
INSERT INTO jobs VALUES(1001,'OFFICER',8000);
```

8.

Create the table jobs.

```
CREATE TABLE IF NOT EXISTS jobs (
JOB_ID integer NOT NULL UNIQUE PRIMARY KEY,
JOB_TITLE varchar(35) NOT NULL,
MIN_SALARY decimal(6,0)
);
```

```
INSERT INTO jobs VALUES(1001,'OFFICER',8000);
```

```
mysql> SELECT * FROM JOBS;
+--------+-----------+------------+
| JOB_ID | JOB_TITLE | MIN_SALARY |
+--------+-----------+------------+
|   1001 | OFFICER   |       8000 |
+--------+-----------+------------+
```

```
INSERT INTO jobs VALUES(1001,'OFFICER',8000);
```

Let execute the above code in MySQL 5.6 command prompt

```
mysql> INSERT INTO jobs VALUES(1001,'OFFICER',8000);
ERROR 1062 (23000): Duplicate entry '1001' for key 'PRIMARY'
```

9.

Create the table countries.

```
CREATE TABLE IF NOT EXISTS countries (
COUNTRY_ID integer NOT NULL AUTO_INCREMENT PRIMARY KEY,
COUNTRY_NAME varchar(40) NOT NULL,
REGION_ID integer NOT NULL
);
INSERT INTO countries(COUNTRY_NAME,REGION_ID) VALUES('India',185);
```

Let execute the above code in MySQL 5.6 command prompt

```
mysql> SELECT * FROM countries;
+------------+--------------+-----------+
| COUNTRY_ID | COUNTRY_NAME | REGION_ID |
+------------+--------------+-----------+
|          1 | India        |       185 |
+------------+--------------+-----------+
1 row in set (0.00 sec)
```

11.

Create the table countries.

```
CREATE TABLE IF NOT EXISTS countries (
COUNTRY_ID integer NOT NULL AUTO_INCREMENT PRIMARY KEY,
COUNTRY_NAME varchar(40) NOT NULL DEFAULT 'N/A',
REGION_ID integer NOT NULL
);
INSERT INTO countries VALUES(501,'India',102);
```

Let execute the above code in MySQL 5.6 command prompt

```
mysql> SELECT * FROM countries;
+------------+--------------+-----------+
| COUNTRY_ID | COUNTRY_NAME | REGION_ID |
+------------+--------------+-----------+
|        501 | India        |       102 |
+------------+--------------+-----------+
1 row in set (0.00 sec)
```

```
INSERT INTO countries(region_id) VALUES(109);
```

Let execute the above code in MySQL 5.6 command prompt

```
mysql> SELECT * FROM countries;
+------------+--------------+-----------+
| COUNTRY_ID | COUNTRY_NAME | REGION_ID |
+------------+--------------+-----------+
|        501 | India        |       102 |
|        502 | N/A          |       109 |
+------------+--------------+-----------+
2 rows in set (0.00 sec)
```

```
INSERT INTO countries(country_name,region_id) VALUES('Australia',121);
```

Let execute the above code in MySQL 5.6 command prompt

```
mysql> SELECT * FROM countries;
+------------+--------------+-----------+
| COUNTRY_ID | COUNTRY_NAME | REGION_ID |
+------------+--------------+-----------+
|        501 | India        |       102 |
|        502 | N/A          |       109 |
|        503 | Australia    |       121 |
+------------+--------------+-----------+
3 rows in set (0.00 sec)
```

12.

Sample table jobs.

```
CREATE TABLE IF NOT EXISTS jobs (
JOB_ID integer NOT NULL UNIQUE PRIMARY KEY,
JOB_TITLE varchar(35) NOT NULL DEFAULT ' ',
MIN_SALARY decimal(6,0) DEFAULT 8000,
MAX_SALARY decimal(6,0) DEFAULT 20000
)ENGINE=InnoDB;

INSERT INTO jobs(JOB_ID,JOB_TITLE) VALUES(1001,'OFFICER');
INSERT INTO jobs(JOB_ID,JOB_TITLE) VALUES(1002,'CLERK');
```

```
+--------+-----------+------------+------------+
| JOB_ID | JOB_TITLE | MIN_SALARY | MAX_SALARY |
+--------+-----------+------------+------------+
|   1001 | OFFICER   |       8000 |      20000 |
|   1002 | CLERK     |       8000 |      20000 |
+--------+-----------+------------+------------+
2 rows in set (0.00 sec)
```

Sample table job_history;

```
CREATE TABLE job_history (
EMPLOYEE_ID integer NOT NULL PRIMARY KEY,
JOB_ID integer NOT NULL,
DEPARTMENT_ID integer DEFAULT NULL,
FOREIGN KEY (job_id) REFERENCES jobs(job_id)
)ENGINE=InnoDB;
INSERT INTO job_history VALUES(501,1001,60);
```

Let execute the above code in MySQL 5.6 command prompt

```
mysql> SELECT * FROM job_history;
+-------------+--------+---------------+
| EMPLOYEE_ID | JOB_ID | DEPARTMENT_ID |
+-------------+--------+---------------+
|         501 |   1001 |            60 |
+-------------+--------+---------------+
1 row in set (0.00 sec)
```

The value against job_id is 1001 which is exists in the job_id column of the jobs table, so no problem arise.

Now insert another row in the job_history table.

```
INSERT INTO job_history VALUES(502,1003,80);
```

Let execute the above code in MySQL 5.6 command prompt

```
mysql> INSERT INTO job_history VALUES(502,1003,80);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
(`hrr`.`job_history`, CONSTRAINT `job_history_ibfk_1`
 (`JOB_ID`) REFERENCES `jobs` (`JOB_ID`))
```

Here in the above, the value against job_id is 1003 which is not exists in the job_id column
 of the jobs(parent table) table and that is why the child table job_history can not contain
 the value of job_id as specified.
Here the primary key - foreign key relationship is violating and shows the above message.

13.

Sample table departments.

```
CREATE TABLE IF NOT EXISTS departments (
DEPARTMENT_ID integer NOT NULL UNIQUE,
DEPARTMENT_NAME varchar(30) NOT NULL,
MANAGER_ID integer DEFAULT NULL,
LOCATION_ID integer DEFAULT NULL,
PRIMARY KEY (DEPARTMENT_ID,MANAGER_ID)
)ENGINE=InnoDB;


INSERT INTO departments VALUES(60,'SALES',201,89);
INSERT INTO departments VALUES(61,'ACCOUNTS',201,89);
INSERT INTO departments VALUES(80,'FINANCE',211,90);

mysql> SELECT * FROM departments;
+---------------+-----------------+------------+-------------+
| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
+---------------+-----------------+------------+-------------+
|            60 | SALES           |        201 |          89 |
|            61 | ACCOUNTS        |        201 |          89 |
|            80 | FINANCE         |        211 |          90 |
+---------------+-----------------+------------+-------------+
3 rows in set (0.00 sec)
```

Sample table employees.

```
CREATE TABLE IF NOT EXISTS employees (
EMPLOYEE_ID integer NOT NULL PRIMARY KEY,
FIRST_NAME varchar(20) DEFAULT NULL,
LAST_NAME varchar(25) NOT NULL,
JOB_ID varchar(10) NOT NULL,
SALARY decimal(8,2) DEFAULT NULL,
MANAGER_ID integer DEFAULT NULL,
DEPARTMENT_ID integer DEFAULT NULL,
FOREIGN KEY(DEPARTMENT_ID,MANAGER_ID)
REFERENCES  departments(DEPARTMENT_ID,MANAGER_ID)
)ENGINE=InnoDB;
```

Now insert the rows in the employees.

```
INSERT INTO employees VALUES(510,'Alex','Hanes','CLERK',18000,201,60);
INSERT INTO employees VALUES(511,'Kim','Leon','CLERK',18000,211,80);
```

Let execute the above code in MySQL 5.6 command prompt

```
mysql> SELECT * FROM employees;
+-------------+------------+-----------+--------+----------+------------+---------------+
| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | JOB_ID | SALARY   | MANAGER_ID |
DEPARTMENT_ID |
+-------------+------------+-----------+--------+----------+------------+---------------+
|         510 | Alex       | Hanes     | CLERK  | 18000.00 |        201 |            60 |
|         511 | Kim        | Leon      | CLERK  | 18000.00 |        211 |            80 |
+-------------+------------+-----------+--------+----------+------------+---------------+
2 rows in set (0.00 sec)
```

The value against department_id and manager_id combination (60,201) and (80,211) are unique in the departmentis(parent) table so, there is no problem arise to insert the rows in the child table employees.

Now insert another row in the employees table.

```
INSERT INTO employees VALUES(512,'Kim','Leon','CLERK',18000,80,211);
```

Let execute the above code in MySQL 5.6 command prompt

```
mysql> INSERT INTO employees VALUES(512,'Kim','Leon','CLERK',18000,80,211);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint
fails (`hrr`.`employees`, CONSTRAINT `employees_ibfk_1` FOREIGN KEY (`D
EPARTMENT_ID`, `MANAGER_ID`) REFERENCES `departments`
(`DEPARTMENT_ID`, `MANAGER_ID`))
```

Here in the above, the value against department_id and manager_id combination (211,80) does not matching with the same combination in departments(parent table) table and that is why the child table employees can not contain the combination of values including department_id and manager_id as specified. Here the primary key - foreign key relationship is being violated and shows the above message.

14.
Sample table departments.

```
CREATE TABLE IF NOT EXISTS departments (
DEPARTMENT_ID integer NOT NULL UNIQUE,
DEPARTMENT_NAME varchar(30) NOT NULL,
MANAGER_ID integer DEFAULT NULL,
LOCATION_ID integer DEFAULT NULL,
PRIMARY KEY (DEPARTMENT_ID)
)ENGINE=InnoDB;
```

```
INSERT INTO departments VALUES(60,'SALES',201,89);
INSERT INTO departments VALUES(61,'ACCOUNTS',201,89);


mysql> SELECT * FROM departments;
+---------------+-----------------+------------+-------------+
| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
+---------------+-----------------+------------+-------------+
|            60 | SALES           |        201 |          89 |
|            61 | ACCOUNTS        |        201 |          89 |
+---------------+-----------------+------------+-------------+
2 rows in set (0.00 sec)
```

Sample table jobs.

```
CREATE TABLE IF NOT EXISTS jobs (
JOB_ID integer NOT NULL UNIQUE PRIMARY KEY,
JOB_TITLE varchar(35) NOT NULL DEFAULT ' ',
MIN_SALARY decimal(6,0) DEFAULT 8000,
MAX_SALARY decimal(6,0) DEFAULT 20000
)ENGINE=InnoDB;


INSERT INTO jobs(JOB_ID,JOB_TITLE) VALUES(1001,'OFFICER');
INSERT INTO jobs(JOB_ID,JOB_TITLE) VALUES(1002,'CLERK');

mysql> SELECT * FROM jobs;
+--------+-----------+------------+------------+
| JOB_ID | JOB_TITLE | MIN_SALARY | MAX_SALARY |
+--------+-----------+------------+------------+
|   1001 | OFFICER   |       8000 |      20000 |
|   1002 | CLERK     |       8000 |      20000 |
+--------+-----------+------------+------------+
2 rows in set (0.00 sec)
```

Sample table employees.

```
CREATE TABLE IF NOT EXISTS employees (
EMPLOYEE_ID integer NOT NULL PRIMARY KEY,
FIRST_NAME varchar(20) DEFAULT NULL,
LAST_NAME varchar(25) NOT NULL,
DEPARTMENT_ID integer DEFAULT NULL,
FOREIGN KEY(DEPARTMENT_ID)
REFERENCES  departments(DEPARTMENT_ID),
JOB_ID integer NOT NULL,
FOREIGN KEY(JOB_ID)
REFERENCES  jobs(JOB_ID),
SALARY decimal(8,2) DEFAULT NULL
)ENGINE=InnoDB;
```

Now insert the rows into the table employees.

```
INSERT INTO employees VALUES(510,'Alex','Hanes',60,1001,18000);
```

Let execute the above code in MySQL 5.6 command prompt

```
mysql> SELECT * FROM employees;
+-------------+------------+-----------+---------------+--------+----------+
| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | DEPARTMENT_ID | JOB_ID |
SALARY   |
+-------------+------------+-----------+---------------+--------+----------+
|         510 | Alex       | Hanes     |            60 |   1001 | 18000.00 |
+-------------+------------+-----------+---------------+--------+----------+
1 row in set (0.00 sec)
```

Here in the above insert statement the child column department_id and job_id of child table
 employees are successfully referencing with the department_id and job_id column of parent
 tables departments and jobs respectively, so no problem have been arisen to the insertion.

Now insert another row in the employees table.

```
INSERT INTO employees VALUES(511,'Tom','Elan',60,1003,22000);
```

Let execute the above code in MySQL 5.6 command prompt

```
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint
 fails (`hrr`.`employees`, CONSTRAINT `employees_ibfk_2` FORE
OB_ID`) REFERENCES `jobs` (`JOB_ID`))
```

Here in the above insert statement show that, within child columns department_id and job_id of child table employees, the department_id are successfully referencing with the department_id of parent table departments but job_id column are not successfully
 referencing with the job_id of parent table jobs,
so the problem have been arisen to the insertion displayed an error message.

Now insert another row in the employees table.

```
INSERT INTO employees VALUES(511,'Tom','Elan',80,1001,22000);
```

Let execute the above code in MySQL 5.6 command prompt

```
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint
 fails (`hrr`.`employees`, CONSTRAINT `employees_ibfk_2` FOREIGN KEY (`J
OB_ID`) REFERENCES `jobs` (`JOB_ID`))
```

Here in the above insert statement show that, within child columns department_id and job_id of child table employees, the job_id are successfully referencing with the job_id of parent table jobs but department_id column are not successfully referencing with the department_id of parent table departments, so the problem have been arisen to the insertion and displayed the error message.

MySQL Update Table exercises

1. Write a SQL statement to change the email column of employees table with 'not available' for all employees.

2. Write a SQL statement to change the email and commission_pct column of employees table with 'not available' and 0.10 for all employees.

3. Write a SQL statement to change the email and commission_pct column of employees table with 'not available' and 0.10 for those employees whose department_id is 110.

4. Write a SQL statement to change the email column of employees table with 'not available' for those employees whose department_id is 80 and gets a commission is less than .20%

5. Write a SQL statement to change the email column of employees table with 'not available' for those employees who belongs to the 'Accouning' department.
.

6. Write a SQL statement to change salary of employee to 8000 whose ID is 105, if the existing salary is less than 5000.
.

7. Write a SQL statement to change job ID of employee which ID is 118, to SH_CLERK if the employee belongs to department, which ID is 30 and the existing job ID does not start with SH.

8. Write a SQL statement to increase the salary of employees under the department 40, 90 and 110 according to the company rules that, salary will be increased by 25% for the department 40, 15% for department 90 and 10% for the department 110 and the rest of the departments will remain same.

9. Write a SQL statement to increase the minimum and maximum salary of PU_CLERK by 2000 as well as the salary for those employees by 20% and commission by .10% .
Here is the sample table employees.

Update-Table-Exercise Solutions

1.

```
UPDATE employees SET email='not available';
SELECT * FROM employees LIMIT 2;
```

2.

```
UPDATE employees SET email='not available',
commission_pct=0.10;
```

3.

```
UPDATE employees
SET email='not available',
commission_pct=0.10
WHERE department_id=110;
```

4.

```
UPDATE employees
SET email='not available'
WHERE department_id=80 AND commission_pct<.20;
```

5.

```
UPDATE employees
SET email='not available'
WHERE department_id=(
SELECT department_id
FROM departments
WHERE department_name='Accounting');
```

6.

```
UPDATE employees
 SET SALARY = 8000
WHERE employee_id = 105 AND salary < 5000;
```

7.

```
UPDATE employees SET JOB_ID= 'SH_CLERK'
WHERE employee_id=118
AND department_id=30
AND NOT JOB_ID LIKE 'SH%';
```

8.

```
UPDATE employees SET salary= CASE department_id
                         WHEN 40 THEN salary+(salary*.25)
                         WHEN 90 THEN salary+(salary*.15)
                         WHEN 110 THEN salary+(salary*.10)
                         ELSE salary
                       END
            WHERE department_id IN (40,50,50,60,70,80,90,110);
```

9.

```
UPDATE jobs,employees
SET jobs.min_salary=jobs.min_salary+2000,
jobs.max_salary=jobs.max_salary+2000,
employees.salary=employees.salary+(employees.salary*.20),
employees.commission_pct=employees.commission_pct+.10
WHERE jobs.job_id='PU_CLERK'
AND employees.job_id='PU_CLERK';
```

MySQL Alter Table exercises

1. Write a SQL statement to rename the table countries to country_new.

2. Write a SQL statement to add a column region_id to the table locations.

3. Write a SQL statement to add a columns ID as the first column of the table locations.

4. Write a SQL statement to add a column region_id after state_province to the table locations.

5. Write a SQL statement change the data type of the column country_id to integer in the table locations.

6. Write a SQL statement to drop the column city from the table locations.

7. Write a SQL statement to change the name of the column state_province to state, keeping the data type and size same.

8. Write a SQL statement to add a primary key for the columns location_id in the locations table.
Here is the sample table employees.
Sample table: employees

9. Write a SQL statement to add a primary key for a combination of columns location_id and country_id.

10. Write a SQL statement to drop the existing primary from the table locations on a combination of columns location_id and country_id.

11. Write a SQL statement to add a foreign key on job_id column of job_history table referencing to the primary key job_id of jobs table.

12. Write a SQL statement to add a foreign key constraint named fk_job_id on job_id column of job_history table referencing to the primary key job_id of jobs table.

13. Write a SQL statement to drop the existing foreign key fk_job_id from job_history table on job_id column which is referencing to the job_id of jobs table.

14. Write a SQL statement to add an index named indx_job_id on job_id column in the table job_history.

15. Write a SQL statement to drop the index indx_job_id from job_history table.

Alter-Table-Exercise Solutions

1.

```
ALTER TABLE countries RENAME country_new;
```

2.

```
ALTER TABLE locations
ADD region_id  INT;
```

3.

```
ALTER TABLE locations
ADD ID  INT FIRST;
```

4.

```
ALTER TABLE locations
ADD region_id INT
AFTER state_province;
```

5.

```
ALTER TABLE locations
MODIFY country_id INT;
```

6.

```
ALTER TABLE locations
DROP city;
```

7.

```
ALTER TABLE locations
DROP state_province,
ADD state varchar(25)
AFTER city;
```

8.

```
ALTER TABLE locations
ADD PRIMARY KEY(location_id);
```

9.

```
ALTER TABLE locations
ADD PRIMARY KEY(location_id,country_id);
```

10.

```
ALTER TABLE locations DROP PRIMARY KEY;
```

11.

```
ALTER TABLE job_history
ADD FOREIGN KEY(job_id)
REFERENCES jobs(job_id);
```

12.

```
ALTER TABLE job_history
ADD CONSTRAINT fk_job_id
FOREIGN KEY (job_id)
REFERENCES jobs(job_id)
```

```
ON UPDATE RESTRICT
ON DELETE CASCADE;
```

13.

```
ALTER TABLE job_history
DROP FOREIGN KEY fk_job_id;
```

14.

```
ALTER TABLE job_history
ADD INDEX indx_job_id(job_id);
```

15.

```
ALTER TABLE job_history
DROP INDEX indx_job_id;
```

Basic SELECT Statement Exercises

Sample table: employees

1. Write a query to display the names (first_name, last_name) using alias name “First Name", "Last Name"

2. Write a query to get unique department ID from employee table.

3. Write a query to get all employee details from the employee table order by first name, descending.

4. Write a query to get the names (first_name, last_name), salary, PF of all the employees (PF is calculated as 12% of salary).

5. Write a query to get the employee ID, names (first_name, last_name), salary in ascending order of salary.

6. Write a query to get the total salaries payable to employees.

7. Write a query to get the maximum and minimum salary from employees table.

8. Write a query to get the average salary and number of employees in the employees table.

9. Write a query to get the number of employees working with the company.

10. Write a query to get the number of jobs available in the employees table.

11. Write a query get all first name from employees table in upper case.
Sample table: employees

12. Write a query to get the first 3 characters of first name from employees table.

13. Write a query to calculate 171*214+625.

14. Write a query to get the names (for example Ellen Abel, Sundar Ande etc.) of all the employees from employees table.

15. Write a query to get first name from employees table after removing white spaces from both side.

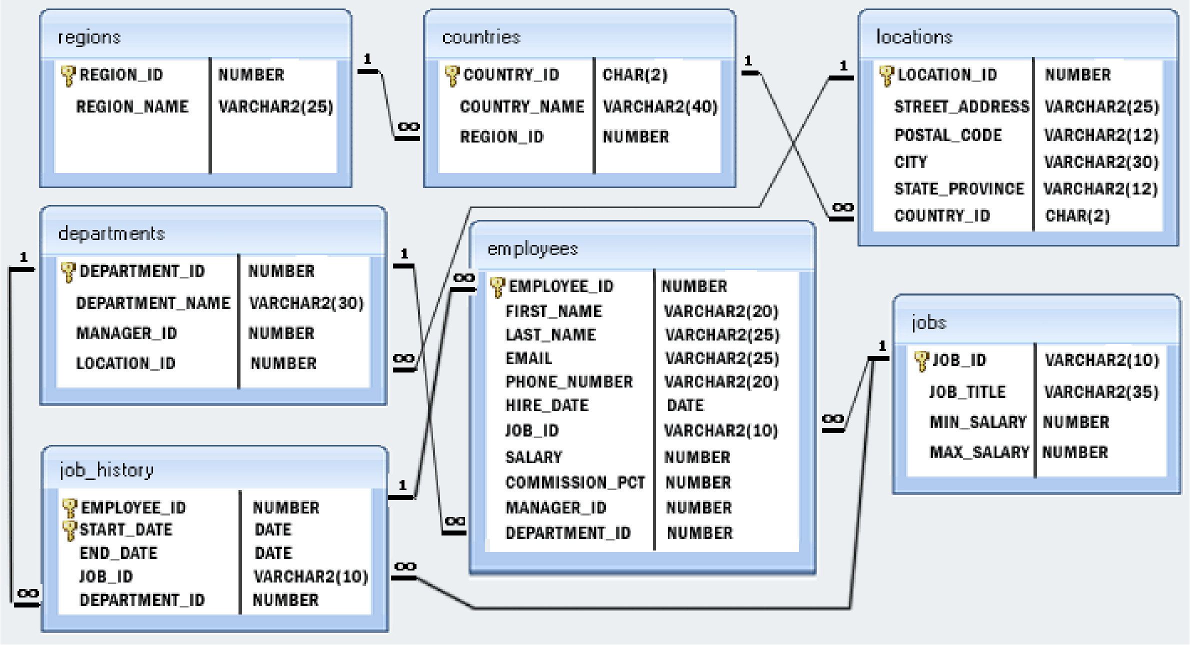
16. Write a query to get the length of the employee names (first_name, last_name) from employees table.

17. Write a query to check if the first_name fields of the employees table contains numbers.

18. Write a query to select first 10 records from a table.

19. Write a query to get monthly salary (round 2 decimal places) of each and every employee
Note : Assume the salary field provides the 'annual salary' information.

... More
Structure of 'hr' database :



regions

REGION_ID	NUMBER
REGION_NAME	VARCHAR2(25)

countries

COUNTRY_ID	CHAR(2)
COUNTRY_NAME	VARCHAR2(40)
REGION_ID	NUMBER

locations

LOCATION_ID	NUMBER
STREET_ADDRESS	VARCHAR2(25)
POSTAL_CODE	VARCHAR2(12)
CITY	VARCHAR2(30)
STATE_PROVINCE	VARCHAR2(12)
COUNTRY_ID	CHAR(2)

departments

DEPARTMENT_ID	NUMBER
DEPARTMENT_NAME	VARCHAR2(30)
MANAGER_ID	NUMBER
LOCATION_ID	NUMBER

employees

EMPLOYEE_ID	NUMBER
FIRST_NAME	VARCHAR2(20)
LAST_NAME	VARCHAR2(25)
EMAIL	VARCHAR2(25)
PHONE_NUMBER	VARCHAR2(20)
HIRE_DATE	DATE
JOB_ID	VARCHAR2(10)
SALARY	NUMBER
COMMISSION_PCT	NUMBER
MANAGER_ID	NUMBER
DEPARTMENT_ID	NUMBER

jobs

JOB_ID	VARCHAR2(10)
JOB_TITLE	VARCHAR2(35)
MIN_SALARY	NUMBER
MAX_SALARY	NUMBER

job_history

EMPLOYEE_ID	NUMBER
START_DATE	DATE
END_DATE	DATE
JOB_ID	VARCHAR2(10)
DEPARTMENT_ID	NUMBER

Basic SELECT statement   Solutions

```
SELECT first_name "First Name",  last_name "Last Name"
FROM employees;
2.
SELECT DISTINCT department_id
FROM employees;
3.
SELECT *
FROM employees
 ORDER BY first_name DESC;
4.
SELECT first_name, last_name, salary, salary*.15 PF
 FROM employees;
5.
SELECT employee_id, first_name, last_name, salary
FROM employees
ORDER BY salary;
6.
SELECT SUM(salary)
FROM employees;
7.
SELECT MAX(salary), MIN(salary)
 FROM employees;
8.
SELECT AVG(salary), COUNT(*)
FROM employees;
9.
SELECT COUNT(DISTINCT job_id)
FROM employees;
10.
SELECT COUNT(DISTINCT job_id)
FROM employees;
11.
SELECT UPPER(first_name)
FROM employees;
12.
SELECT SUBSTRING(first_name,1,3)
 FROM employees;
13.
SELECT 171*214+625 Result;
14.
SELECT  CONCAT(first_name,' ', last_name) 'Employee Name'
```

```
FROM employees;
```

15.

```
SELECT TRIM(first_name)
FROM employees;
```

16.

```
SELECT first_name,last_name, LENGTH(first_name)+LENGTH(last_name)  'Length of
Names'
 FROM employees;
```

17.

```
SELECT * FROM employees
WHERE  first_name REGEXP  '[0-9]';
```

18.

```
SELECT employee_id, first_name
FROM employees  LIMIT 10;
```

19.

```
SELECT first_name, last_name, round(salary/12,2) as 'Monthly Salary'
FROM employees;
```

MySQL Aggregate Functions and Group by- Exercises

Sample table: employees

1. Write a query to list the number of jobs available in the employees table.

2. Write a query to get the total salaries payable to employees.

3. Write a query to get the minimum salary from employees table.

4. Write a query to get the maximum salary of an employee working as a Programmer.

5. Write a query to get the average salary and number of employees working the department 90.

6. Write a query to get the highest, lowest, sum, and average salary of all employees.

7. Write a query to get the number of employees with the same job.

8. Write a query to get the difference between the highest and lowest salaries.

9. Write a query to find the manager ID and the salary of the lowest-paid employee for that manager.

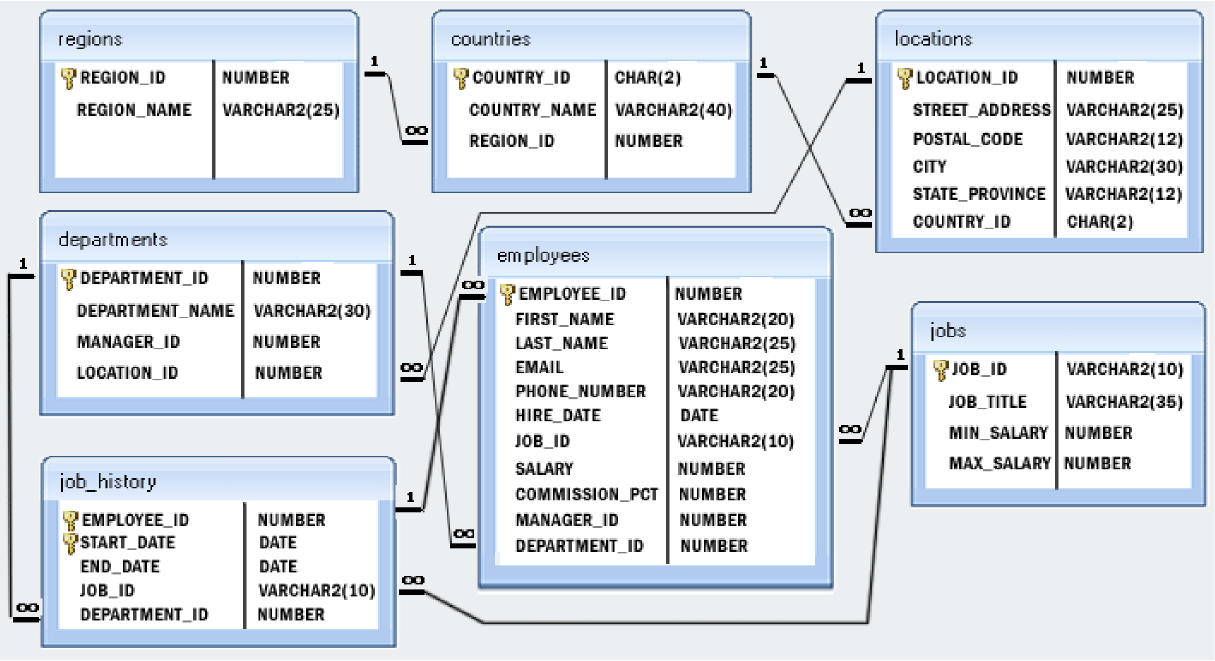
10. Write a query to get the department ID and the total salary payable in each department.

11. Write a query to get the average salary for each job ID excluding programmer.
12. Write a query to get the total salary, maximum, minimum, average salary of employees (job ID wise), for department ID 90 only.

13. Write a query to get the job ID and maximum salary of the employees where maximum salary is greater than or equal to $4000.

14. Write a query to get the average salary for all departments employing more than 10 employees.

... More
Structure of 'hr' database :



regions

REGION_ID	NUMBER
REGION_NAME	VARCHAR2(25)

countries

COUNTRY_ID	CHAR(2)
COUNTRY_NAME	VARCHAR2(40)
REGION_ID	NUMBER

locations

LOCATION_ID	NUMBER
STREET_ADDRESS	VARCHAR2(25)
POSTAL_CODE	VARCHAR2(12)
CITY	VARCHAR2(30)
STATE_PROVINCE	VARCHAR2(12)
COUNTRY_ID	CHAR(2)

departments

DEPARTMENT_ID	NUMBER
DEPARTMENT_NAME	VARCHAR2(30)
MANAGER_ID	NUMBER
LOCATION_ID	NUMBER

employees

EMPLOYEE_ID	NUMBER
FIRST_NAME	VARCHAR2(20)
LAST_NAME	VARCHAR2(25)
EMAIL	VARCHAR2(25)
PHONE_NUMBER	VARCHAR2(20)
HIRE_DATE	DATE
JOB_ID	VARCHAR2(10)
SALARY	NUMBER
COMMISSION_PCT	NUMBER
MANAGER_ID	NUMBER
DEPARTMENT_ID	NUMBER

jobs

JOB_ID	VARCHAR2(10)
JOB_TITLE	VARCHAR2(35)
MIN_SALARY	NUMBER
MAX_SALARY	NUMBER

job_history

EMPLOYEE_ID	NUMBER
START_DATE	DATE
END_DATE	DATE
JOB_ID	VARCHAR2(10)
DEPARTMENT_ID	NUMBER

Aggregate-Function-Group by-Exercises Solutions

1

```
SELECT COUNT(DISTINCT job_id)
FROM employees;
```

2

```
SELECT SUM(salary)
 FROM employees;
```

3

```
SELECT MIN(salary)
FROM employees;
```

4

```
SELECT MAX(salary)
 FROM employees
WHERE job_id = 'IT_PROG';
```

5

```
SELECT ROUND(MAX(salary),0) 'Maximum', ROUND(MIN(salary),0) 'Minimum',
ROUND(SUM(salary),0) 'Sum', ROUND(AVG(salary),0) 'Average'
FROM employees;
```

6

```
SELECT ROUND(MAX(salary),0) 'Maximum', ROUND(MIN(salary),0) 'Minimum',
ROUND(SUM(salary),0) 'Sum', ROUND(AVG(salary),0) 'Average'
FROM employees;
```

7

```
SELECT job_id, COUNT(*)
 FROM employees
 GROUP BY job_id;
```

8

```
SELECT MAX(salary) - MIN(salary) DIFFERENCE
FROM employees;
```

9

```
SELECT manager_id, MIN(salary)
FROM employees
WHERE manager_id IS
NOT NULL GROUP BY manager_id
ORDER BY MIN(salary) DESC;
```

10

```
SELECT department_id, SUM(salary)
FROM employees
 GROUP BY department_id;
```

11

```
SELECT job_id, AVG(salary)
 FROM employees
```

```
WHERE job_id <> 'IT_PROG'
GROUP BY job_id;
```

12

```
SELECT job_id, SUM(salary), AVG(salary), MAX(salary), MIN(salary)
 FROM employees
WHERE department_id = '90'
GROUP BY job_id;
```

13.

```
SELECT job_id, MAX(salary)
FROM employees
GROUP BY job_id
 HAVING MAX(salary) >=4000;
```

14.

```
SELECT job_id, AVG(salary), COUNT(*)
FROM employees
GROUP BY department_id
 HAVING COUNT(*) > 10;
```

MySQL Date and Time – Exercises

1. Write a query to display the first day of the month (in datetime format) three months before the current month.
Sample current date : 2014-09-03
Expected result : 2014-06-01

2. Write a query to display the last day of the month (in datetime format) three months before the current month.

3. Write a query to get the distinct Mondays from hire_date in employees tables.
Sample table: employees

4. Write a query to get the first day of the current year.

5. Write a query to get the last day of the current year.

6. Write a query to calculate the age in year.

7. Write a query to get the current date in the following format.
Sample date : 2014-09-04
Output : September 4, 2014

8. Write a query to get the current date in the following format.
Thursday September 2014

9. Write a query to extract the year from the current date.

10. Write a query to get the DATE value from a given day (number in N).
Sample days : 730677
Output : 2000-07-11

11. Write a query to get the first name and hire date from employees table where hire date between '1987-06-01' and '1987-07-30'
Sample table : employees

12. Write a query to display the current date in the following format.
Sample output : Thursday 4th September 2014 00:00:00

13.Write a query to display the current date in the following format.
Sample output : 05/09/2014

14.Write a query to display the current date in the following format.
Sample output : 12:00 AM Sep 5, 2014

15. Write a query to get the firstname, lastname who joined in the month of June.
Sample table : employees

16. Write a query to get the years in which more than 10 employees joined.
Sample table : job_history

17. Write a query to get the department ID, year, and number of employees joined.
Sample table : employees

18. Write a query to get department name, manager name, and salary of the manager for all managers whose experience is more than 5 years.
Sample table : employees

19. Write a query to get employee ID, last name, and date of first salary of the employees.
Sample table : employees

20. Write a query to get first name, hire date and experience of the employees.
Sample table : employees

21. Write a query to get first name of employees who joined in 1987.
Sample table : employees

Date Time Exercises Solutions

1

```
SELECT date(((PERIOD_ADD
  (EXTRACT(YEAR_MONTH
     FROM CURDATE()),-3)*100)+1));
```

2

```
SELECT (SUBDATE(ADDDATE
     (CURDATE(),INTERVAL 1 MONTH),
         INTERVAL DAYOFMONTH(CURDATE())DAY))
            AS LastDayOfTheMonth;
```

3

```
SELECT DISTINCT(STR_TO_DATE
     (CONCAT(YEARWEEK(hire_date),'1'),'%x%v%w'))
          FROM employees;
```

4

```
SELECT MAKEDATE(EXTRACT(YEAR FROM CURDATE()),1);
```

5

```
SELECT STR_TO_DATE(CONCAT(12,31,
      EXTRACT(YEAR FROM CURDATE())), '%m%d%Y');
```

6

```
SELECT YEAR(CURRENT_TIMESTAMP) -
         YEAR("1967-06-08") -
             (RIGHT(CURRENT_TIMESTAMP, 5) <
                  RIGHT("1967-06-08", 5)) as age;
```

7

```
SELECT DATE_FORMAT(CURDATE(),'%M %e, %Y')
   AS 'Current_date';
```

8

```
SELECT DATE_FORMAT(NOW(), '%W %M %Y');
```

9

```
SELECT EXTRACT(YEAR FROM  NOW());
```

10

```
SELECT FROM_DAYS(730677);
```

11

```
SELECT FIRST_NAME, HIRE_DATE
    FROM employees
       WHERE HIRE_DATE
         BETWEEN '1987-06-01 00:00:00'
            AND '1987-07-30  23:59:59';
```

12

```
SELECT date_format(CURDATE(),'%W %D %M %Y %T');
```

13

```
SELECT date_format(CURDATE(),'%d/%m/%Y');
```

14

```
SELECT date_format(CURDATE(),'%l:%i %p %b %e, %Y');
```

15

```
SELECT first_name, last_name
    FROM employees WHERE MONTH(HIRE_DATE) =  6;
```

16

```
SELECT DATE_FORMAT(HIRE_DATE,'%Y')
     FROM employees
       GROUP BY DATE_FORMAT(HIRE_DATE,'%Y')
          HAVING COUNT(EMPLOYEE_ID) > 10;
```

17

```
SELECT DATE_FORMAT(HIRE_DATE,'%Y')
     FROM employees
       GROUP BY DATE_FORMAT(HIRE_DATE,'%Y')
          HAVING COUNT(EMPLOYEE_ID) > 10;
```

18

```
SELECT DEPARTMENT_NAME, FIRST_NAME, SALARY
   FROM departments D
      JOIN employees E
        ON (D.MANAGER_ID=E.MANAGER_ID)
          WHERE  (SYSDATE()-HIRE_DATE) / 365 > 5;
```

19

```
SELECT employee_id, last_name, hire_date,  LAST_DAY(hire_date)
 FROM employees;
```

20

```
SELECT FIRST_NAME, SYSDATE(), HIRE_DATE, DATEDIFF( SYSDATE(), hire_date )/365
  FROM employees;
```

21

```
SELECT FIRST_NAME, HIRE_DATE FROM employees WHERE
YEAR(HIRE_DATE)=1987;
```

MySQL Restricting and Sorting Data – Exercises

Sample table: employees

1. Write a query to display the names (first_name, last_name) and salary for all employees whose salary is not in the range $10,000 through $15,000.

2. Write a query to display the names (first_name, last_name) and department ID of all employees in departments 30 or 100 in ascending alphabetical order by department ID.

3. Write a query to display the names (first_name, last_name) and salary for all employees whose salary is not in the range $10,000 through $15,000 and are in department 30 or 100.

4. Write a query to display the names (first_name, last_name) and hire date for all employees who were hired in 1987.

5. Write a query to display the first_name of all employees who have both "b" and "c" in their first name.

6. Write a query to display the last name, job, and salary for all employees whose job is that of a Programmer or a Shipping Clerk, and whose salary is not equal to $4,500, $10,000, or $15,000.

7. Write a query to display the last names of employees whose names have exactly 6 characters.

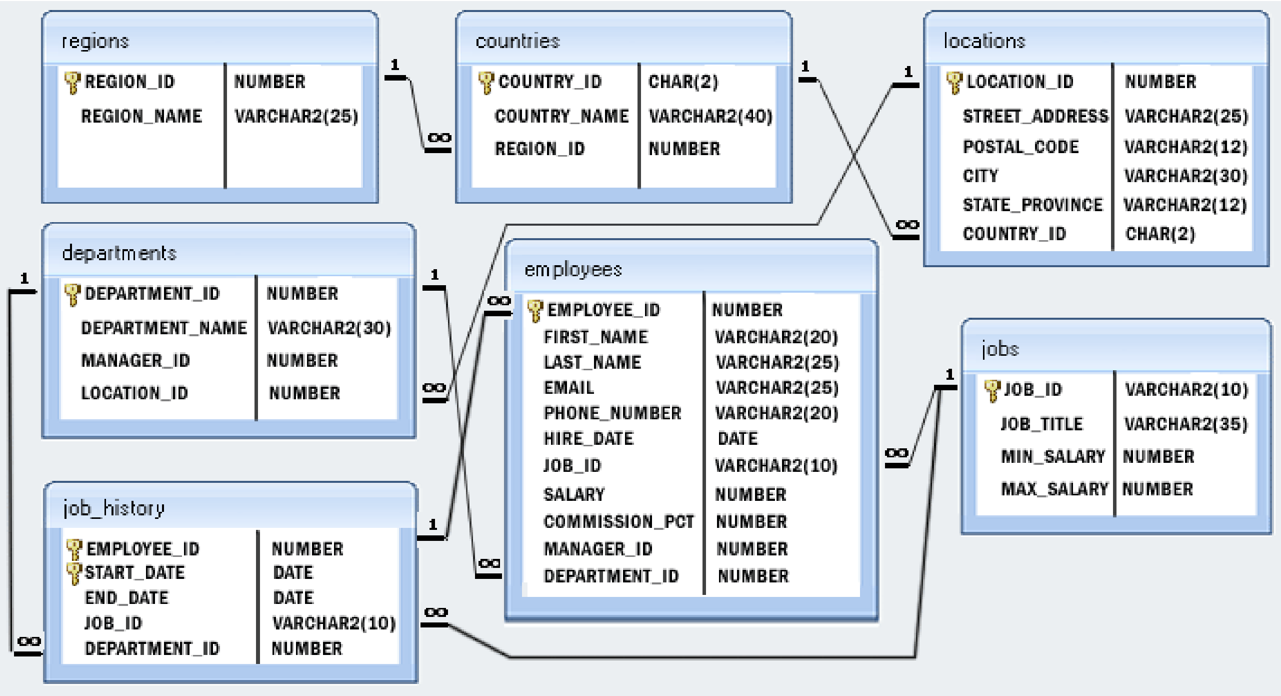
8. Write a query to display the last names of employees having 'e' as the third character.

9. Write a query to display the jobs/designations available in the employees table.

10. Write a query to display the names (first_name, last_name), salary and PF (15% of salary) of all employees.

11. Write a query to select all record from employees where last name in 'BLAKE', 'SCOTT', 'KING' and 'FORD'.

... More
Structure of 'hr' database :



regions

REGION_ID	NUMBER
REGION_NAME	VARCHAR2(25)

countries

COUNTRY_ID	CHAR(2)
COUNTRY_NAME	VARCHAR2(40)
REGION_ID	NUMBER

locations

LOCATION_ID	NUMBER
STREET_ADDRESS	VARCHAR2(25)
POSTAL_CODE	VARCHAR2(12)
CITY	VARCHAR2(30)
STATE_PROVINCE	VARCHAR2(12)
COUNTRY_ID	CHAR(2)

departments

DEPARTMENT_ID	NUMBER
DEPARTMENT_NAME	VARCHAR2(30)
MANAGER_ID	NUMBER
LOCATION_ID	NUMBER

employees

EMPLOYEE_ID	NUMBER
FIRST_NAME	VARCHAR2(20)
LAST_NAME	VARCHAR2(25)
EMAIL	VARCHAR2(25)
PHONE_NUMBER	VARCHAR2(20)
HIRE_DATE	DATE
JOB_ID	VARCHAR2(10)
SALARY	NUMBER
COMMISSION_PCT	NUMBER
MANAGER_ID	NUMBER
DEPARTMENT_ID	NUMBER

jobs

JOB_ID	VARCHAR2(10)
JOB_TITLE	VARCHAR2(35)
MIN_SALARY	NUMBER
MAX_SALARY	NUMBER

job_history

EMPLOYEE_ID	NUMBER
START_DATE	DATE
END_DATE	DATE
JOB_ID	VARCHAR2(10)
DEPARTMENT_ID	NUMBER

MySQL Restricting and Sorting Data solutions

1

```
SELECT first_name, last_name, salary
 FROM employees
WHERE salary NOT BETWEEN 10000 AND 15000;
```

2

```
SELECT first_name, last_name, department_id
 FROM employees
 WHERE department_id IN (30, 100)
ORDER BY  department_id  ASC;
```

3

```
SELECT first_name, last_name, salary, department_id
FROM employees
WHERE salary NOT BETWEEN 10000 AND 15000 AND department_id IN (30, 100);
```

4

```
SELECT first_name, last_name, hire_date
FROM employees
 WHERE YEAR(hire_date)  LIKE '%87';
```

5

```
SELECT first_name
 FROM employees
WHERE first_name LIKE '%b%' AND first_name LIKE '%c%';
```

6

```
SELECT last_name, job_id, salary
 FROM employees
WHERE job_id IN ('IT_PROG', 'SH_CLERK') AND salary NOT IN (4500,10000, 15000);
```

7

```
SELECT last_name
 FROM employees
WHERE last_name LIKE '______';
```

8

```
SELECT last_name
 FROM employees
 WHERE last_name LIKE '__e%';
```

9

```
SELECT DISTINCT job_id
FROM employees;
```

10

```
SELECT first_name, last_name, salary, salary*.15 PF
 from employees;
```

11

```
SELECT *
FROM employees
WHERE last_name IN('JONES', 'BLAKE', 'SCOTT', 'KING', 'FORD');
```

MySQL String – Exercises

1. Write a query to get the job_id and related employee's id.
Partial output of the query :

```
job_id         Employees ID
AC_ACCOUNT 206
AC_MGR         205
AD_ASST        200
AD_PRES        100
AD_VP          101 ,102
FI_ACCOUNT  110 ,113 ,111 ,109 ,112
```

Sample table: employees

2. Write a query to update the portion of the phone_number in the employees table, within the phone number the substring '124' will be replaced by '999'.
Sample table: employees

3. Write a query to get the details of the employees where the length of the first name greater than or equal to 8.
Sample table: employees

4. Write a query to display leading zeros before maximum and minimum salary.
Sample table: jobs

5. Write a query to append '@example.com' to email field.
Sample table: employees
Sample Output :

```
 EMAIL
 --------------------
 SKING@example.com
 NKOCHHAR@example.com
 LDEHAAN@example.com
 AHUNOLD@example.com
 BERNST@example.com
 DAUSTIN@example.com
 VPATABAL@example.com
 DLORENTZ@example.com
 NGREENBE@example.com
 - - - -  - - - - - - -
 - - - -  - - - - - - -
```

6. Write a query to get the employee id, first name and hire month.
Sample table: employees

7. Write a query to get the employee id, email id (discard the last three characters).
Sample table: employees

8. Write a query to find all employees where first names are in upper case.
Sample table: employees

9. Write a query to extract the last 4 character of phone numbers.
Sample table: employees

10. Write a query to get the last word of the street address.
Sample table: locations

11. Write a query to get the locations that have minimum street length.
Sample table: locations

12. Write a query to display the first word in job title.
Sample table: jobs

13. Write a query to display the length of first name for employees where last name contain character 'c' after 2nd position.
Sample table: employees

14. Write a query that displays the first name and the length of the first name for all employees whose name starts with the letters 'A', 'J' or 'M'. Give each column an appropriate label. Sort the results by the employees' first names.
Sample table: employees

15. Write a query to display the first name and salary for all employees. Format the salary to be 10 characters long, left-padded with the $ symbol. Label the column SALARY.
Sample table: employees

16. Write a query to display the first eight characters of the employees' first names and indicates the amounts of their salaries with '$' sign. Each '$' sign signifies a thousand dollars. Sort the data in descending order of salary.
Sample table: employees

17. Write a query to display the employees with their code, first name, last name and hire date who hired either on seventh day of any month or seventh month in any year.
Sample table: employees

MySQL String Solutions

1

```
SELECT job_id, GROUP_CONCAT(employee_id, ' ')  'Employees ID'
FROM employees GROUP BY job_id;
```

2

```
UPDATE employees
SET phone_number = REPLACE(phone_number, '124', '999')
WHERE phone_number LIKE '%124%';
```

3

```
SELECT *
FROM employees
WHERE LENGTH(first_name) >= 8;
```

4

```
SELECT job_id,  LPAD( max_salary, 7, '0') ' Max Salary',
LPAD( min_salary, 7, '0') ' Min Salary'
FROM jobs;
```

5

```
UPDATE employees SET email = CONCAT(email, '@example.com');
```

6

```
SELECT employee_id, first_name, MID(hire_date, 6, 2) as hire_month FROM employees;
```

7

```
SELECT employee_id, REVERSE(SUBSTR(REVERSE(email), 4)) as Email_ID
from employees;
```

8

```
SELECT * FROM employees
WHERE first_name = BINARY UPPER(first_name);
```

9

```
SELECT RIGHT(phone_number, 4) as 'Ph.No.' FROM employees;
```

10

```
SELECT location_id, street_address,
SUBSTRING_INDEX(REPLACE(REPLACE(REPLACE(street_address,',',' '),')',' '),'(',' '),' ',-1)
AS 'Last--word-of-street_address'
FROM locations;
```

11

```
SELECT * FROM locations
WHERE LENGTH(street_address) <= (SELECT  MIN(LENGTH(street_address))
FROM locations);
```

12

```
SELECT job_title, SUBSTR(job_title,1, INSTR(job_title, ' ')-1)
FROM jobs;
```

13

```
SELECT first_name, last_name FROM employees WHERE INSTR(last_name,'C') > 2;
```

14

```
SELECT first_name "Name",
LENGTH(first_name) "Length"
FROM employees
WHERE first_name LIKE 'J%'
OR first_name LIKE 'M%'
OR first_name LIKE 'A%'
ORDER BY first_name ;
```

15.

```
SELECT first_name,
LPAD(salary, 10, '$') SALARY
FROM employees;
```

16.

```
SELECT left(first_name, 8),
REPEAT('$', FLOOR(salary/1000))
'SALARY($)', salary
FROM employees
ORDER BY salary DESC;
```

17.

```
SELECT employee_id,first_name,last_name,hire_date
FROM employees
WHERE POSITION("07" IN DATE_FORMAT(hire_date, '%d %m %Y'))>0;
```

MySQL Sub Queries Exercises

1. Write a query to find the names (first_name, last_name) and the salaries of the employees who have a higher salary than the employee whose last_name='Bull'.

2. Write a query to find the names (first_name, last_name) of all employees who works in the IT department.

3. Write a query to find the names (first_name, last_name) of the employees who have a manager and work for a department based in the United States.
Hint : Write single-row and multiple-row subqueries
Sample table: employees
Sample table: departments
Sample table: locations

4. Write a query to find the names (first_name, last_name) of the employees who are managers.

5. Write a query to find the names (first_name, last_name), the salary of the employees whose salary is greater than the average salary.

6. Write a query to find the names (first_name, last_name), the salary of the employees whose salary is equal to the minimum salary for their job grade.
Sample table: employees
Sample table: jobs

7. Write a query to find the names (first_name, last_name), the salary of the employees who earn more than the average salary and who works in any of the IT departments.
Sample table: employees
Sample table: departments

8. Write a query to find the names (first_name, last_name), the salary of the employees who earn more than Mr. Bell.
Sample table: employees
Sample table: departments

9. Write a query to find the names (first_name, last_name), the salary of the employees who earn the same salary as the minimum salary for all departments.
Sample table: employees
Sample table: departments

10. Write a query to find the names (first_name, last_name), the salary of the employees whose salary greater than the average salary of all departments.

Sample table: employees

11. Write a query to find the names (first_name, last_name) and salary of the employees who earn a salary that is higher than the salary of all the Shipping Clerk (JOB_ID = 'SH_CLERK'). Sort the results of the salary of the lowest to highest.
Sample table: employees

12. Write a query to find the names (first_name, last_name) of the employees who are not supervisors.
Sample table: employees

13. Write a query to display the employee ID, first name, last names, and department names of all employees.
Sample table: employees
Sample table: departments

14. Write a query to display the employee ID, first name, last names, salary of all employees whose salary is above average for their departments.
Sample table: employees
Sample table: departments

15. Write a query to fetch even numbered records from employees table.
Sample table: employees

16. Write a query to find the 5th maximum salary in the employees table.
Sample table: employees

17. Write a query to find the 4th minimum salary in the employees table.
Sample table: employees

18. Write a query to select last 10 records from a table.
Sample table: employees

19. Write a query to list department number, name for all the departments in which there are no employees in the department.
Sample table: employees
Sample table: departments

20. Write a query to get 3 maximum salaries.

21. Write a query to get 3 minimum salaries.

22. Write a query to get nth max salaries of employees.

MySQL Sub Queries Solutions

1

```
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM employees
WHERE SALARY > (SELECT salary FROM employees WHERE last_name = 'Bull');
```

2

```
SELECT first_name, last_name
FROM employees
WHERE department_id
IN (SELECT department_id FROM departments WHERE department_name='IT');
```

3

```
SELECT first_name, last_name
FROM employees
WHERE manager_id IN(select employee_id FROM employees WHERE department_id
IN (SELECT department_id FROM departments WHERE location_id
IN (select location_id from locations where country_id='US')));
```

4

```
SELECT first_name, last_name
FROM employees
WHERE (employee_id IN (SELECT manager_id FROM employees));
```

5

```
SELECT first_name, last_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

6

```
SELECT first_name, last_name, salary
FROM employees
WHERE employees.salary = (SELECT min_salary FROM jobs
  WHERE employees.job_id = jobs.job_id);
```

7

```
SELECT first_name, last_name, salary
FROM employees
WHERE department_id IN (SELECT department_id FROM departments
  WHERE department_name LIKE 'IT%') AND salary > (SELECT avg(salary) FROM
employees);
```

8

```
SELECT first_name, last_name, salary
FROM employees
WHERE salary >
(SELECT salary FROM employees WHERE last_name = 'Bell') ORDER BY first_name;
```

9

```
SELECT *
FROM employees
WHERE salary = (SELECT MIN(salary) FROM employees);
```

10

```
SELECT *
FROM employees
WHERE salary >
ALL(SELECT avg(salary)
FROM employees GROUP BY department_id);
```

11.

```
SELECT first_name,last_name, job_id, salary
FROM employees
WHERE salary >
ALL (SELECT salary FROM employees WHERE job_id = 'SH_CLERK')
ORDER BY salary ;
```

12.

```
SELECT b.first_name,b.last_name
FROM employees b
WHERE NOT EXISTS
(SELECT 'X' FROM employees a WHERE a.manager_id = b.employee_id);
```

13

```
SELECT employee_id, first_name, last_name,
(SELECT department_name FROM departments d
 WHERE e.department_id = d.department_id) department
 FROM employees e ORDER BY department;
```

14

```
SELECT employee_id, first_name
FROM employees
AS A WHERE salary >
( SELECT AVG(salary) FROM employees WHERE department_id = A.department_id);
```

15

```
SET @i = 0;
SELECT i, employee_id
FROM (SELECT @i := @i + 1 AS i, employee_id FROM employees)
a WHERE MOD(a.i, 2) = 0;
```

16

```
SELECT DISTINCT salary
FROM employees e1
WHERE 5 = (SELECT COUNT(DISTINCT salary)
FROM employees  e2
 WHERE e2.salary >= e1.salary);
```

17

```
SELECT DISTINCT salary
FROM employees e1
WHERE 4 = (SELECT COUNT(DISTINCT salary)
FROM employees  e2 WHERE e2.salary <= e1.salary);
```

18

```
SELECT * FROM
(SELECT * FROM employees
ORDER BY employee_id DESC LIMIT 10)
sub ORDER BY employee_id ASC;
```

19

```
SELECT *
FROM departments
WHERE department_id
NOT IN (select department_id FROM employees);
```

20

```
SELECT DISTINCT salary
FROM employees a
WHERE 3 >= (SELECT COUNT(DISTINCT salary)
FROM employees b
WHERE b.salary <= a.salary) ORDER BY a.salary DESC;
```

21

```
SELECT DISTINCT salary
FROM employees a
WHERE  3 >= (SELECT COUNT(DISTINCT salary)
FROM employees b WHERE b.salary >= a.salary)
ORDER BY a.salary DESC;
```

22

```
SELECT * FROM employees emp1 WHERE (1) = (SELECT
COUNT(DISTINCT(emp2.salary)) FROM employees emp2 WHERE emp2.salary >
emp1.salary);
```

MySQL JOINS – Exercises

1. Write a query to find the addresses (location_id, street_address, city, state_province, country_name) of all the departments.
Hint : Use NATURAL JOIN.
Sample table : locations
Sample table : countries

2. Write a query to find the names (first_name, last name), department ID and name of all the employees.
Sample table : employees
Sample table : departments

3. Find the names (first_name, last_name), job, department number, and department name of the employees who work in London.
Sample table : departments
Sample table : locations

4. Write a query to find the employee id, name (last_name) along with their manager_id, manager name (last_name).
Sample table : employees

5. Find the names (first_name, last_name) and hire date of the employees who were hired after 'Jones'.
Sample table : employees

6. Write a query to get the department name and number of employees in the department.
Sample table : employees
Sample table : departments

7. Find the employee ID, job title, number of days between ending date and starting date for all jobs in department 90 from job history.
Sample table : employees

8. Write a query to display the department ID, department name and manager first name.
Sample table : employees
Sample table : departments

9. Write a query to display the department name, manager name, and city.
Sample table : employees
Sample table : departments
Sample table : locations

10. Write a query to display the job title and average salary of employees.
Sample table : employees

11. Display job title, employee name, and the difference between salary of the employee and minimum salary for the job.
Sample table : employees

12. Write a query to display the job history that were done by any employee who is currently drawing more than 10000 of salary.
Sample table : employees
Sample table : Job_history

13. Write a query to display department name, name (first_name, last_name), hire date, salary of the manager for all managers whose experience is more than 15 years.
Sample table : employees
Sample table : departments

Joins Exercises Solutions

1

```
SELECT location_id, street_address, city, state_province, country_name
FROM locations
NATURAL JOIN countries;
```

2

```
SELECT first_name, last_name, department_id, department_name
FROM employees
JOIN
departments
USING (department_id);
```

3

```
SELECT e.first_name, e.last_name, e.job_id, e.department_id, d.department_name
FROM employees e
 JOIN
 departments d ON (e.department_id = d.department_id)
 JOIN
 locations l ON (d.location_id = l.location_id)
 WHERE LOWER(l.city) = 'London';
```

4

```
SELECT e.employee_id 'Emp_Id', e.last_name 'Employee', m.employee_id 'Mgr_Id',
m.last_name 'Manager'
FROM employees e
join employees m
 ON (e.manager_id = m.employee_id);
```

5

```
SELECT e.first_name, e.last_name, e.hire_date
 FROM employees e
 JOIN
 employees davies
 ON (davies.last_name = 'Jones')
 WHERE davies.hire_date < e.hire_date;
```

6.

```
SELECT department_name AS 'Department Name',
COUNT(*) AS 'No of Employees'
FROM departments
INNER JOIN employees
ON employees.department_id = departments.department_id
GROUP BY departments.department_id, department_name
ORDER BY department_name;
```

7

```
SELECT employee_id, job_title, end_date-start_date Days FROM job_history
NATURAL JOIN jobs
WHERE department_id=90;
```

8

```
SELECT d.department_id, d.department_name, e.manager_id, e.first_name
FROM departments d
INNER JOIN employees e
ON (d.manager_id = e.employee_id);
```

9

```
SELECT d.department_name, e.first_name, l.city
FROM departments d
JOIN employees e
ON (d.manager_id = e.employee_id)
JOIN locations l USING (location_id);
```

10

```
SELECT job_title, AVG(salary)
FROM employees
NATURAL JOIN jobs
GROUP BY job_title;
```

11

```
SELECT job_title, first_name, salary-min_salary 'Salary - Min_Salary'
FROM employees
NATURAL JOIN jobs;
```

12

```
SELECT jh.* FROM job_history jh
JOIN employees e
ON (jh.employee_id = e.employee_id)
WHERE salary > 10000;
```

13.

```
SELECT first_name, last_name, hire_date, salary,
(DATEDIFF(now(), hire_date))/365 Experience
FROM departments d JOIN employees e
ON (d.manager_id = e.employee_id)
WHERE (DATEDIFF(now(), hire_date))/365>15;
```

MySQL Northwind database, Products table – Exercises

1. Write a query to get Product name and quantity/unit.

2. Write a query to get current Product list (Product ID and name).

3. Write a query to get discontinued Product list (Product ID and name).

4. Write a query to get most expense and least expensive Product list (name and unit price).

5. Write a query to get Product list (id, name, unit price) where current products cost less than $20.

6. Write a query to get Product list (id, name, unit price) where products cost between $15 and $25.

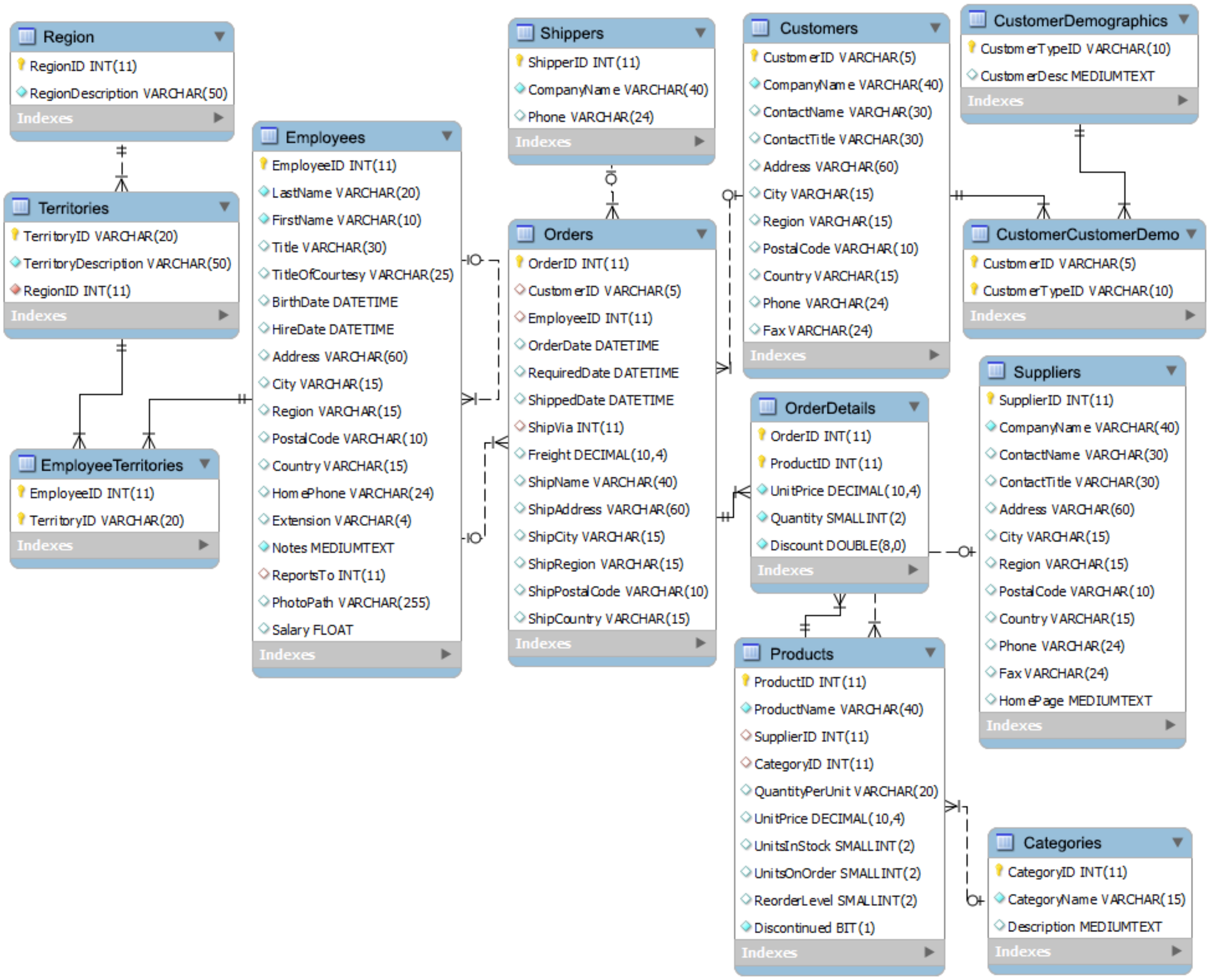
7. Write a query to get Product list (name, unit price) of above average price.

8. Write a query to get Product list (name, unit price) of ten most expensive products.

9. Write a query to count current and discontinued products.

10. Write a query to get Product list (name, units on order , units in stock) of stock is less than the quantity on order.

... More
Structure of 'northwind' database :


Region
RegionID INT(11)
RegionDescription VARCHAR(50)
Indexes
Territories
TerritoryID VARCHAR(20)
TerritoryDescription VARCHAR(50)
RegionID INT(11)
Indexes
EmployeeTerritories
EmployeeID INT(11)
TerritoryID VARCHAR(20)
Indexes
Employees
EmployeeID INT(11)
LastName VARCHAR(20)
FirstName VARCHAR(10)
Title VARCHAR(30)
TitleOfCourtesy VARCHAR(25)
BirthDate DATETIME
HireDate DATETIME
Address VARCHAR(60)
City VARCHAR(15)
Region VARCHAR(15)
PostalCode VARCHAR(10)
Country VARCHAR(15)
HomePhone VARCHAR(24)
Extension VARCHAR(4)
Notes MEDIUMTEXT
ReportsTo INT(11)
PhotoPath VARCHAR(255)
Salary FLOAT
Indexes
Shippers
ShipperID INT(11)
CompanyName VARCHAR(40)
Phone VARCHAR(24)
Indexes
Orders
OrderID INT(11)
CustomerID VARCHAR(5)
EmployeeID INT(11)
OrderDate DATETIME
RequiredDate DATETIME
ShippedDate DATETIME
ShipVia INT(11)
Freight DECIMAL(10,4)
ShipName VARCHAR(40)
ShipAddress VARCHAR(60)
ShipCity VARCHAR(15)
ShipRegion VARCHAR(15)
ShipPostalCode VARCHAR(10)
ShipCountry VARCHAR(15)
Indexes
Customers
CustomerID VARCHAR(5)
CompanyName VARCHAR(40)
ContactName VARCHAR(30)
ContactTitle VARCHAR(30)
Address VARCHAR(60)
City VARCHAR(15)
Region VARCHAR(15)
PostalCode VARCHAR(10)
Country VARCHAR(15)
Phone VARCHAR(24)
Fax VARCHAR(24)
Indexes
CustomerDemographics
CustomerTypeID VARCHAR(10)
CustomerDesc MEDIUMTEXT
Indexes
CustomerCustomerDemo
CustomerID VARCHAR(5)
CustomerTypeID VARCHAR(10)
Indexes
OrderDetails
OrderID INT(11)
ProductID INT(11)
UnitPrice DECIMAL(10,4)
Quantity SMALLINT(2)
Discount DOUBLE(8,0)
Indexes
Suppliers
SupplierID INT(11)
CompanyName VARCHAR(40)
ContactName VARCHAR(30)
ContactTitle VARCHAR(30)
Address VARCHAR(60)
City VARCHAR(15)
Region VARCHAR(15)
PostalCode VARCHAR(10)
Country VARCHAR(15)
Phone VARCHAR(24)
Fax VARCHAR(24)
HomePage MEDIUMTEXT
Indexes
Products
ProductID INT(11)
ProductName VARCHAR(40)
SupplierID INT(11)
CategoryID INT(11)
QuantityPerUnit VARCHAR(20)
UnitPrice DECIMAL(10,4)
UnitsInStock SMALLINT(2)
UnitsOnOrder SMALLINT(2)
ReorderLevel SMALLINT(2)
Discontinued BIT(1)
Indexes
Categories
CategoryID INT(11)
CategoryName VARCHAR(15)
Description MEDIUMTEXT
Indexes

MySQL Northwind database, Products table – Solution

```
1
SELECT ProductName, QuantityPerUnit
FROM Products;
2
SELECT ProductID, ProductName
FROM Products
WHERE Discontinued = "False"
ORDER BY ProductName;
3
SELECT ProductID, ProductName
FROM Products
WHERE Discontinued = "True"
ORDER BY ProductName;
4
SELECT ProductName, UnitPrice
FROM Products
ORDER BY UnitPrice DESC;
5
SELECT ProductID, ProductName, UnitPrice
FROM Products
WHERE (((UnitPrice)<20) AND ((Discontinued)=False))
ORDER BY UnitPrice DESC;
6
SELECT ProductName, UnitPrice
FROM Products
WHERE (((UnitPrice)>=15 And (UnitPrice)<=25)
AND ((Products.Discontinued)=False))
ORDER BY Products.UnitPrice DESC;
7
SELECT DISTINCT ProductName, UnitPrice
FROM Products
WHERE UnitPrice > (SELECT avg(UnitPrice) FROM Products)
ORDER BY UnitPrice;
8
SELECT DISTINCT ProductName as Twenty_Most_Expensive_Products, UnitPrice
FROM Products AS a
WHERE 20 >= (SELECT COUNT(DISTINCT UnitPrice)
             FROM Products AS b
             WHERE b.UnitPrice >= a.UnitPrice)
ORDER BY UnitPrice desc;
```

9

```
SELECT Count(ProductName)
FROM Products
GROUP BY Discontinued;
```

10.

```
SELECT ProductName,  UnitsOnOrder , UnitsInStock
FROM Products
WHERE (((Discontinued)=False) AND ((UnitsInStock)<UnitsOnOrder));
```