Unit 10

Working with SQL

(Practical)

In these exercises, exposure was given to create databases within MySQL, create, drop and modify tables, insert data into tables, update existing data in tables, delete tables databases and existing table data. Also considered, was the role of constraints such as primary keys and foreign keys. Lastly, these exercise tackled aggregate functions available within MySQL.

Introduction to MySQL

Challeng: Select Table Data

```
show databases;
use people;
show tables;
select * from basic_info;
```

Challenge: Limit queries

```
use people;
select name from names;
select * from basic_info limit 3;
```

Challenge: Order query output

```
use people;
select surname from names;
select name, id from names order by id desc;
```

Challenge: Using the 'WHERE' clause

```
use people;

select email from basic_info where email='sed.pede@nisi.com';
```

Challenge: Using query operators

```
select * from basic_info where birthday<=2016 and id between 1 and 5 order by id desc;
```

Challenge: 'SELECT' clauses

```
use directory;

select * from company_profiles limit 5;

select id, company_name from company_profiles order by id desc;
```

Challenge: 'SELECT' operators

```
use directory;

select * from company_profiles where company_name='Lorem Ipsum Institute' or

company_name='Et LLP';

select id, phone_num from company_profiles where id between 40 and 60;
```

Challenge: SQL file statements

```
use directory;

show tables;

select * from company_profiles where id=27;

select id, phone_num from company_profiles order by id asc;

select id, company_name from company_profiles where id>=40 limit 5;
```

Other Statements

#1

```
USE dog_breeds;

CREATE TABLE dog_catalog (id INT(4) NOT NULL auto_increment, breed VARCHAR(255),
region VARCHAR(255), PRIMARY KEY (id)) auto_increment = 1;
```

#2

```
USE dog_breeds;
```

```
INSERT INTO dog_catalog (breed, region) VALUES ("English Sheepdog",     "   England"),
("Tibetan Mastiff", "Asia"), ("Labrador", "Canada");
```

#3

```
USE famous_scientists;
CREATE TABLE scientists (
       id INT(4) NOT NULL auto_increment,
       name VARCHAR(255) NOT NULL,
       discovery TEXT NOT NULL,
       year_of_birt INT(4) NOT NULL,
       year_of_death INT(4) DEFAULT NULL,
       PRIMARY KEY (id)
) auto_increment = 1;
```

#4

```
USE famous_scientists;
SHOW COLUMNS from scientists;
INSERT INTO scientists (name, discovery, year_of_birth, year_of_death)
VALUES
('Archimedes', 'Finding the exact volume of an irregularly shaped object',
 287, 212),
('Galileo Galilei', 'Modern observational astronomy', 1564, 1642),
('Sir Isaac Newton', 'Gravity', 1642, 1726),
('Thomas Alva Edison', 'Electric bulb', 1847, 1931);
```

#5

```
use college;
select * from classrooms join courses on classrooms.course_id=courses.id;
```

#6

```
use college;
select * from classrooms left outer join courses on classrooms.course_id=courses.id;
```

#7

```
use college;
select * from classrooms right outer join courses on courses.id=classrooms.course_id;
```

#8

```
use college;
select   classrooms.course_id   from   courses   left   outer   join   classrooms   on
classrooms.course_id=courses.id;
```

#9

```
use blog;
select users.name, posts.body, users.id
from blog.posts
right outer join users on users.id=posts.user_id;
```

Aggregate Functions

#1

```
use e_store;
select min(stock) from products;
```

3

```
select stock as min_stock from products where id > 3 and stock=(select min(stock)
from products where id > 3);
```

#2

```
use e_store;
select max(created_at) as newest_product from products;
select min(created_at) as newest_product from products;
```

#3

```
select min(stock) as least_stock, max(created_at) as newest_product, max(price) as
product_max_price from products;
```

#4

```
use e_store;
select products.id, products.name, products.price, reviews.stars as stars from
products left join reviews on reviews.product_id = products.id where reviews.stars >
1 order by stock, created_at, reviews.stars limit 3;
```

#5

```
use e_store;
select sum(stock) as total_stock from products;
select product_id as id, sum(stars) as total_stars from reviews where product_id=4;
```

#6

```
use e_store;
select count(*) as total_product_count from products;
select product_id as id, count(*) as stars_fields_count from reviews where
product_id=4;
```

#7

```
use e_store;
select product_id as id, sum(stars) / count(stars) as avg_rating from reviews where
product_id=4;
```

#8

```
use e_store;
select product_id as id, avg(stars) from reviews where product_id=4;
```

#9

```
use e_store;
select products.id, products.name, avg(reviews.stars) as avg_stars
from products
left outer join reviews on reviews.product_id=products.id
where products.id in (2, 3)
group by products.id ;
```

#10

```
select min(stock) as min_stock, max(stock) as max_stock from products;
```

#11

```
select sum(products.stock) as total_stock, count(products.id) as total_products from
products;
```

#12

```
use e_store;
select products.name, products.price, products.stock, avg(reviews.stars) as avg_stars
from products join reviews on reviews.product_id=products.id
group by products.id having avg_stars> 3;
```

Advanced SQL

Exercise 1

```
use Bus_Depot;
```

#1

```
select bdno, bdname from BusDriver where bdsalary<1800;
```

#2

```
select bdno, bdname from BusDriver where bdname like 'J%';
```

#3

```
select * from BusDriver where bdsalary between 2000 and 4000;
```

#4

```
select regno, model from Bus join Depot on Depot.dno=Bus.dno where tno='2' and
Depot.dno<>'101';
```

#5

```
select Bus.* from Bus join BusType on BusType.tno=Bus.tno
where BusType.tno='Volvo' or BusType.tno='Mercedes';
```

#6

```
select distinct dno from Depot;
```

#7

```
select cno, cname, dname, daddress from Cleaner join Depot on Depot.dno=Cleaner.dno;
```

#8

```
select distinct BusDriver.bdno, BusDriver.bdname, BusType.tdescript from BusDriver
join Training on Training.bdno=BusDriver.bdno
join BusType on BusType.tno=Training.tno;
```

#9

```
select Cleaner.cno, Cleaner.cname, dname from Cleaner join Depot on
Depot.dno=Cleaner.dno
join Bus on Bus.cno=Cleaner.cno and Bus.dno=Depot.dno;
```

#10

```
select distinct Cleaner.cno, cname, dname from Cleaner left outer join Depot on
Depot.dno=Cleaner.dno
```

```
left outer join Bus on Bus.dno=Depot.dno
left outer join BusType on BusType.tno=Bus.tno;
```

Exercise 2

```
use Bus_Depot;
```

#1
```
select max(bdsalary), min(bdsalary), avg(bdsalary)
from BusDriver;
```

#2
```
select count(*) as total_drivers
from BusDriver join Depot on Depot.dno=BusDriver.dno
where Depot.dname='Middlesex Transport';
```

#3
```
select rno, rdescript
from Route
where dno=(select dno from Depot where dname='Holloway');
```

#4
```
select rno, rdescript
from Route
join Depot on Depot.dno=Route.dno
where Depot.dname='Holloway';
```

#5
```
select *
from Bus
left outer join Depot on Depot.dno=Bus.dno
where Depot.dno is null;
```

#6
```
select bdname, bdno
from BusDriver
left outer join Route on Route.dno=BusDriver.dno
where Route.dno is null;
```

#7
```
select BusDriver.dno, avg(bdsalary)
from BusDriver
join Depot on Depot.dno=BusDriver.dno
group by BusDriver.dno;
```

#8

```
select dname, count(*)
from Depot
join BusDriver on BusDriver.dno=Depot.dno
group by dname having count(*) > 1;
```

#9

```
select cname, Cleaner.cno, count(regno)
from Cleaner
join Depot on Depot.dno=Cleaner.dno
join Bus on Bus.dno=Depot.dno and Bus.model='Doubledecker' or Bus.model='Minibus'
group by cname, cno;
```

#10.a

```
select bdname, bdno, rno, rdescript
from BusDriver
join Route on Route.dno=BusDriver.dno
order by bdno;
```

#10.b

```
select bdname, bdno, rno, rdescript
from BusDriver
join Route on Route.dno=BusDriver.dno
order by bdno, rdescript;
```