SQL Practice 1

1) To create a table salesman.

```
create table salesman (
    salesman_id int Primary key,
    salesman_name varchar (20),
    city varchar,
    commission float (10));
```

2) To create a table for customer

```
create table customer (
    customer_id int primary key,
    customer_name varchar (20),
    customer_city varchar (20),
    customer_grade int,
    foreign key (salesman_id) references salesman (salesman_id));
```

3) To create a table for order.

```
create table order_num (
    order_no int primary key,
    purc_amt float (15),
    order_date date,
    customer_id int,
    salesman_id int,
    foreign key (salesman_id) references salesman
        (salesman_id),
    foreign key (customer_id) references customer (customer_id));
```

4) Insert values into salesman.

```
insert into salesman values ("5001", "James Hoog", "New York",
    0.5),
```

Output

Salesman_id	salesman_name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13

Customer_id	Customer_name	Customer_city	Customer_grade	Salesman_id
3002	Nick Rimando	New York	100	5001
3004	Fabian Johns	Paris	300	5006
3005	Graham Zusi	California	200	5002
3007	Brad Davis	New York	200	5001
3008	Julian Green	London	300	5002

Order_no	Purc_amt	order_date	Customer_id	salesman_id
70001	150.5	2016-01-06	3002	5002
70002	270.65	2017-01-03	3009	5001
70005	2400.6	2018-01-03	3001	5001
70007	938.5	2013-01-03	3002	5002
70008	1983.43	2019-01-03	3009	5002
70011	75.29	2015-07-03	3001	5002
70012	250.45	2015-09-03	3002	5002

name	commission
James Hoog	0.15
Nail Knite	0.13
Pit Alex	0.11
Mc Lyon	0.14
Paul Adam	0.13

(5002, "Nail Knite", "Paris", 0.13),
(5005, "Pit Alex", "London", 0.11),
(5006, "Mc Lyon", "Paris", 0.14),
(5007, "Paul Adam", "Rome", 0.3));

5) Insert values into customer value table.

Insert into customer values (3002, "Nick Rimando", "New York", 100, 5001)

(3004, "Fabian Johns", "Paris", 300, 5006)
(3005, "Graham Zusi", "California", 200, 5002)
(3007, "Brad Davis", "New York", 200, 5001)
(3008, "Julian Green", "London", 300, 5002);

6) Insert values into order table.

Insert into order_num values (70002, 140.5, "2016-01-03", 3001, 5002),
(70003, 160.5, "2016-01-06", 3002, 5002),
(70004, 170.5, "2017-01-03", 3004, 5001),
(70005, 180.5, "2018-01-03", 3001, 5001),
(70006, 190.5, "2017-01-03", 3002, 5002),
(70007, 190.5, "2019-01-03", 3009, 5002),
(70008, 340.5, "2015-07-03", 3001, 5002),
(70009, 650.5, "2015-09-03", 3002, 5002);

Query 1:

Display name and commission for all the salesman.

=) select salesman_name, commission from salesman.

Query 2:

Retrieve salesman_id of all salesman from orders table without any repeats.

select Distinct salesman_id from orders

Query 3:

Display names and city of salesman, who belongs to the city of Paris.

salesman_id
5002
5003
5006
5001
5005
5007

name	city
Nail Knite	Paris
Mc Lyon	Paris

customer_id	cust name	city	grade	salesman_id
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3003	Jozy Altidor	Moscow	200	5002

ord_no	ord_date	purch_amt
70002	2012-10-05	65.26
70005	2012-07-27	2400.60
70008	2012-09-10	5760.00
70013	2012-06-25	3045.60

winner
Pablo Neruda

=> Select salesman_name, city from salesman where city = "Paris";

Query 4:

Display all the information for those customers with a grade of 200.

=> select * from customer where grade = 200;

Query 5:

Display the order number, order date and the purchase amount for orders which will be delivered by the salesman with ID 5001.

=> select ord_no, ord_date, purch_amt From orders where salesman_id = 5001;

Query 6:

Show the winner of the 1971 prize for literature

=> select winner from nobel_win where year=1971 and subject = 'literature';

output :

~~select *~~

emp_id	first_name	hourly_pay	salary	job
1	xxx	51	106080.00	manager
2	yyy	91	189280.00	cashier
3	zzz	81	168480.00	cook
4	eee	290	603200.00	teacher

expense_id	expense_name	expense_total
1	salaries	898560.00
2	supplies	0.00
3	taxes	0.00

Triggers, Delimiter, Cursor

=> create table employees (
emp_id int primary key,
first_name varchar(20),
hourly_pay int,
job varchar(20));

=> insert into employees values (1, "xxx", 100, "manager");
insert into employees values (2, "yyy", 90, "cashier"),
(3, "zzz", 80 "cook"),
(4, "eee", 150 "teacher");

=> alter table employees
add column salary decimal(10,2) after hourly_pay;
select * from employees;

=> update employees
set salary = hourly_pay * 2080;
select * from employees;

=> create trigger before_hourly_pay_update
before update on employees
for each row
set new.salary = (new.hourly_pay * 2080);
select * from employees;

=> show triggers;

update employees
set hourly_pay = 50
where emp_id = 1;

=> update employees
set hourly_pay = hourly_pay + 1;

-> delete from employees
where emp_id = 4;

=> create trigger before_hourly_pay_interest
before insert on employees
for each row
set new.salary = (new.hourly_pay * 2080);

=> insert into employees
values (4, "aac", 290, null, "teacher")

=> select * from employees;

=> create table expenses (
expense_id int primary key,
expense_name varchar(50),
expense_total decimal(10,2));

=> insert into expenses
values (1, "salaries", 0),
values (2, "supplies", 0);
values (3, "taxes", 0);

-> update expenses
set expense_total = (select sum(salary) from employees)
where expense_name = "salaries";

=> create trigger after_salary_delete
after delete on employees
for each row
update expenses
set expense_total = expense_total - old.salary

where expense_name = "salaries";

=> delete from employees where emp_id = 3;

=> create trigger after_salary_insert
after insert on employees
for each row
update expenses
set expense_total = expense_total + new.salary
where expense_name = "salaries";

=> insert into employees values (3, "111", 180, null, "doctor");

// Delimiter & cursor.

create table customer (
id int not null,
Name varchar (20) Not null,
Age int not null,
Address char (25),
salary decimal (18, 2),
Primary Key (ID));

=> Insert into customer values (1, Ramesh, 32, Ahmedabad, 2000.0
(2, Kilan, 25, Delhi, 1500.00),
(3, "Kaushik", 23, Kota, 2000.00),
(4, "Bhaibali", 25, Mumbai", 6500.00);

=> select * from customers;

=> create table customers_Backup (
ID int not null,
Name varchar (20) Not Null,
Primary key (ID));
Delimiter //

=> create Procedure Fetch_customers ()
Begin

```
Declare done int default false;
Declare customer_id Int;
Declare customer_name varchar(255);
Declare auto_id Int;

-- Declare cursor
Declare my_cursor cursor for
select id, name From customers;
-- Declare exit handler
Declare continue handler for not found set done = true;

--Open cursor
Open my cursor;

-- Fetch and insert rows
read_loop: loop
    Fetch my_cursor into customer_id, customer_name;
    if done = 1 Then
        leave read_loop;
    End if;
-- Get the last auto-generated ID used in the insert
set auto_id = Last_insert_id();
End loop;
-- Close my_cursor;
Delimiter;
call Fetch_customers();
```

Output:

ID	name	dept. name	salary
10101	Srinivasan	Comp. Sci	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
45565	Gold	Physics	87000
58583	Katz	Comp. Sci	75000
76543	Califieri	History	62000
76766	Singh	Finance	80000
83821	Brick	Biology	72000
98345	Kim	Elec. Eng.	80000

ID	course_ID	sec_ID	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017
			Spring	2018

Accessing the database

1) Create the following Relation with primary key integrity constraint instructor

```
create table Instructor (
        ID integer Primary key,
        name text not null,
        dept text not null,
        salary int );
```

```
--Insert
Insert into instructor values (10101, 'Srinivasan', 'Comp.Sci', 65000),
(12121, 'Wu', 'Finance', 90000),
(15151, 'Mozart', 'Music', 40000),
(22222, 'Einstein', 'Physics', 95000),
(32343, 'El Said', 'History', 60000),
(33456, 'Gold', 'Physics', 87000),
(45565, 'Katz', 'Comp.Sci', 75000),
(58583, 'Califieri', 'History', 62000),
(76543, 'Singh', 'Finance', 80000),
(76766, 'Brick', 'Biology', 72000),
(83821, 'Brandt', 'Comp.Sci', 92000),
(98345, 'Kim', 'Elec. Eng', 80000);
```

2) Create the following Relation teacher.

```
create table Teacher (
        ID integer not null
        course_ID varchar (50) not null,
        sec_id varchar not null,
        semester char (50) not null,
        year int not null,
        Foreign key (ID) References instructor (ID)
```

```
-- Insert
Insert into teacher values (10101, 'CS-101', 1, 'Fall', 2017),
(10101, 'CS-315', 1, 'Spring', 2018),
(10101, 'CS-347', 1, 'Fall', 2017),
(12121, 'FIN-201', 1, 'Spring', 2018),
```

ID	Name	dept_name	Salary
32343	El said	History	60000
58583	Califieri	History	62000

name	course_ID
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El said	HIS-351
Katz	CS-101
Crick	CS-319
Crick	BIO-101
Brandt	BIO-301
Kim	CS-319

ID	name	dept_name	Salary
10101	Srinivasan	Comp. Sci	65000
2222	Einstein	Physics	95000
76543	Singh	Finance	80000

ID	name	dept_name	Salary
12121	Wu	Finance	90000
22222	Einstein	Physics	95000
83821	Brandt	Comp. Sci	92000

(15151, 'MU-199', 1, 'Spring', 2017),
(22222, 'PHY-101', 1, 'Fall', 2018),
(32343, 'HIS-351', 1, 'Spring', 2018),
(45565, 'CS-101', 1, 'Spring', 2018),
(45565, 'CS-319', 1, 'Spring', 2017),
(76766, BIO-101, 1, Summer, 2018),
(83821, BIO-301, 1, Summer, 2018),
(83821, CS-190, 1, Spring, 2017),
(83821, CS-190, 2, Spring, 2018),
(98345, CS319, 2, Spring, 2017);

3) Insert following additional tuple in instructor
('10211', 'Smith', 'Biology', 66000)

Insert into instructor values (10211, 'Smith', 'Biology', 66000);

4) Delete this tuple from instructor (10211, Smith, Biology, 66000)
Delete into instructor where ID = '10211';

5) Select tuples from instructor where dept = 'History'.
select * from instructor where dept = 'History';

6) Find the cartesian product instructor x teacher.
select * from instructor cross join Teacher;

7) Find the names of all instructors who have taught some course and the course-id.
select name, course_ID from instructor inner join teacher on Instructor.ID = Teacher.ID;

8) Find the names of all instructors whose name includes the substring "dar".
select * from instructor where name like '%dar%';

course_ID
CS - 101
CS - 347
PHY - 101

ID	name	dept name	salary
10212	Tom	Biology	Null

Avg (salary)
74153.8462

count (*)
7

count (*)
15

dept_name	avg (salary)
comp. sci	77333.3333
Biology	6819.6667
Finance	40000.0000
Physics	48250.0000
History	61000.0000
Elect. Eng.	80000.0000

dept_name	salary
comp. sci	77333.3333
Biology	6819.6667
Finance	40000.0000
Physics	48250.0000
History	61000.0000
Elec. Eng	80000.0000

9) Find the names of all instructors with salary between 90,000 and 100,000 (that is ≥ 90,000, ≤ 100000)

select * from instructor where salary between 90000 and 100000;

Experiment 2: Basic SQL

1) Order the tuples in the instructors relation as per their salary

select * from instructor order by salary ASC;

2) Find courses that ran in Fall 2017 or in Spring 2018

select course_ID from teacher where (semester = "Fall" and year = "2017") or (semester = "Spring" and year = "2018");

3) Find the average salary of instructors in each department

select course_ID from Teacher where (semester = "Fall" and year = "2017") and (semester = "Spring" and year = "2018");

4) Find courses that ran in Fall 2017 but not in Spring 2018.

Select course_ID from Teachers where (semester = "Fall" and year = "2017");

5) Insert following additional tuples in instructor:

Insert into teacher values ("10211", Smith, "Biology", 60000)
("10212", "Tom", "Biology", Null);

6) Find all instructors whose salary is Null

select * from instruction where salary = NULL;

7) Find the average salary of instructors in comp. sci dept

select AVG (salary) from instructor where Dept = "comp.sci";

name
Srinivasan
Smith
Tom
Wu
El Said
Gold
Katz
Singh
Crick
Kim

name
Wu
Einstein
Gold
Katz
Singh
Crick
Brandt
Kim

~~dept_name~~ ~~avg_salary~~

dept_name
Physics

Experiment 3: Intermediate SQL

1) Find the total number of instructors who teach a courses in the spring 2018 semester.

~~select count (distinct ID) as total~~ .

```
select count (*) from instructor where (semester = 'spring'
                                        and year = '2018');
```

2) Find the number of tuples in the teacher relation

```
select count (*) from instructor;
```

3) Find the average salary of instructors in each department.

```
select instructor dept_name, avg (instructor salary) from
instructor group by dept_name;
```

4) Find the names and average salaries of all departments whose average salary is greater than 42000.

```
select dept_name, avg (salary) from instructor group by
dept_name having avg (salary) > 42000;
```

5) Name all instructors whose name is neither "Mozart" nor "Einstein".

```
select name from instructor where name not in ('Mozart',
                                               'Einstein');
```

6) Find names of instructors with salary greater than the salary of ~~all~~ at least one instructor in Biology department.

```
select n. name from instructor where n.salary > any
( select salary from instructor where dept_name = 'Biology');
```

name	course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	Fin-201
Mozart	Mu-199
Einstein	Phy-101
El Said	His-351
Katz	CS-101
Brink	CS-319
Brink	Bio-101
Brandt	Bio-301
Brandt	CS-190
Kim	CS-319
	EE-181

id	name	dept
10101	Srinivasan	CS
10211	Smith	Biology
10212	Kim	Biology
10232	Adam	Biology
10326	Wu	Music
12121	Mozart	Finance
13151	Einstein	Music
22222	El Said	Physics
32343	Gold	History
33456	Katz	Physics
45565	Singh	Comp. Sci.
58583	Brink	History
76543	Brandt	Biology
76766	Kim	Comp. Sci.
83821		

7) Find the names of all instructors whose salary is greater than the salary of all instructors in Biology department.

select name from instructor where salary > all (select salary from instructor where dept_name = 'Biology');

8) Find the average instructors salaries of those departments where the average salary is greater than 42000.

select dept_name, avg (salary) as avg_salary from instructor group by dept_name having avg (salary) > 42000;

Experiment 4: Intermediate and advanced SQL.

1) Find all departments where the total salary is greater than the average of the total salary at all departments.

select dept_name from instructor group by dept_name having sum (salary) > (select avg (total_salary) from (select sum (salary) as total_salary from instructor group by dept_name) as subquery);

2) List the names of instructors along with the course ID of the courses that they taught.

select instructor.name, teaches.course_ID from instructor join teaches on instructor.id = teaches.id;

3) List the names of instructors along with the course ID of the courses that they taught. In case, an instructor teaches no courses keep the course ID as null.

select instructor.name, teacher.course_ID from instructor left join teacher on instructor.id = teacher.ID;

dept_name	total_salary
Comp. Sci	232000
Biology	138000
Finance	170000
Music	40000
Physics	182000
History	122000
Elec. Eng.	80000

4) Create a view of instructors without their salary called faculty.

Create view faculty as select id, name, dept from instructor;

5) Give select privileges on the view faculty to the new user.

Grant select on faculty to new user.

Experiment 5: Advanced SQL

1) Create a view of instructors without their salary called faculty.

Create view faculty as Select ID, name, dept_name from Instructor where salary is Null;

2) Create a view of department salary totals.

Create view department_salary_totals as select dept_name, sum(salary) as total_salary from instructor Group By dept_name;

select * from department_salary_totals;

3) Create a role of student.

Create role student;

4) Give select privileges on the view faculty to the role student.

Grant select on faculty to student;

5) Create a new user and assign her the role of student.

Create user. new_user. identified by 'password';

6) Login as this new user and find all instructors in the Biology department.
select name from instructor where dept_name = 'Biology';

7) Revoke privileges of the new user
Revoke student from new_user;

8) Remove the role of student.
Drop Role student;

9) Give select privileges on the view faculty to the new user.
Grant select on faculty to new_user;

10) Login as this new user and find all instructors in the finance department.
select * from instructor where dept_name = 'Finance';

11) Login again as root user.
select Host, user, plugin from mysql.user;

12) Create table teaches2 with same columns as teaches but with additional constraint that the semester is one of fall, winter, spring or summer.
select * from TEACHES2;

13) create index teaches_id_index on TEACHES(ID);

14) Drop index teaches_id_index on TEACHES2;

9) (32343, 'El said', 'History', 60000)

(58583, 'Califieri', 'History', 62000)

Experiment : 6 Accessing the database through python.

1) Insert following additional tuple in instructor:

(10211, 'Smith', 'Biology', 66000)

```
mycursor = mydb.cursor()
mycursor.execute("Insert into instructor(ID, name, dept_name,
salary) values (%s, %s, %s, %s)", val = (10577, 'Sam', 'Biology',
'46000')
for i in mycursor:
    print(i)
```

2) Delete this tuple from instructor.

```
mycursor.execute("Delete from instructor where ID=10577")
```

3) Select tuples from instructor where dept_name = "History".

```
mycursor.execute("select * from instructor where
                  dept_name = 'History'")
```

4) Find the cartesian product instructor x teaches.

```
mycursor.execute("select * from instructor cross join Teaches")
```

5) Find the name of all instructors who have taught some courses and the course_id.

```
mycursor.execute("select name, course_ID from instructo
inner join Teaches on instructor.ID = Teaches.ID")
```

6) Find the name of all instructors whose name includes the sub string.

```
select * from instructo where name like "Sam";
```

7) Find the name of all instructors with salary between 90,000 and 100,000.

```
mycursor.execute("select * from instructo where salary
```

between 90000 and 1000000")

Experiment :7 Advanced techniques through Python.

1) Order the tuples in the instructor relation as per their salary.

mycursor.execute ("select * from teaches order by salary asc")

2) Find courses that ran in Fall 2017 or in Spring 2018

mycursor.execute ("select course_id from teaches where (semester = 'fall' and year = '2017') or (semester = 'spring' and year = '2018)")

3) Find courses that ran in Fall 2017 and in Spring 2018.

mycursor.execute ("selct course_id from teaches where (semester = 'fall' and year = '2017') and (semester = 'spring' and year = '2018')")

4) Find courses that ran in fall 2017 but not and in Spring 2018

mycursor.execute ("selct course_id from teaches where (semester = 'spring' and ~~and~~ year = '2018')")

5) Insert following additional tuples in instructor.

mycursor.execute ("insert into instructor values ('10211', 'Smith', 'Biology', 66000), ('10212', 'Tom', 'Biology', null)")

6) Find all instructors whose salary is null.

mycursor.execute ("select * from instructor where salary is Null")

7) Find the average of instructors in Computer Science department

mycursor.execute("select avg(salary) from instructor
where dept_name = 'Computer Science'")

8) Find the total number of instructors who teach a course in the Spring 2018 semester.

mycursor.execute("select count(*) from instructor
where (semester = 'Spring' and year = '2018')")

9) Find the number of tuples in the teaches relation.

mycursor.execute("select count(*) from instructor")

10) Find the average salary of instructors in each department.

mycursor.execute("select dept_name, avg(salary) from
instructor group by dept_name")

11) Find the names and average salaries of all department whose average salary is greater than 42000.

mycursor.execute("select dept_name, avg(salary) from
instructor group by dept_name having avg(salary)>42000")

12) Name all instructors whose name is either Mozart nor Einstein.

mycursor.execute("select name from instructor where
name not in ("Mozart", "Einstein")")

13) mycursor.execute("select n.name from instructor n
where n.salary > any (select salary from details where
dept_name = 'biology')")

14) mycursor.execute("select name from instructor where
salary > all (select salary from teaches where dept_name =
'Biology')")

15. mycursor.execute("select avg(salary) from instructor group by dept_name having avg(salary) > 42000")

16. mycursor.execute("select dept_name from instructor group by dept_name having sum(salary) > (select avg(total_salary) from (select sum(salary) as total_salary from instructor group by dept_name) as subquery)")

17. mycursor.execute("select instructor.name, Teaches.course_id from instructor join teaches on instructor.id = Teaches.id")

18. mycursor.execute("select instructor.name, Teaches.course_id from instructor left join Teaches on instructor.id = Teaches.id")

Arijit