Trainhalts(id, seqno, stcode, timein, timeout)
Track(stcode1, stcode2, distance)
Station(stcode, name)
Train(id, name)

Write the following Queries for the above Railway Schema

1. Find pairs of stations (station codes) that have a track (direct connection) with distance less than 20Kms between them.
2. Find the IDs of all the trains which have a stop at THANE
3. Find the names of all trains that start at MUMBAI.
4. List all the stations in order of visit by the train 'CST-AMR_LOCAL'.
5. Find the name of the trains which have stop at Thane, before the 6th station in the route of the train.

Solution:

1. **select** stcode1, stcode2
   from track
   where distance < 20;

2. **select** id
   from trainhalts
   where timein <> timeout **and**
   stcode **in** (**select** station.stcode **from** station **where** station.name='THANE') ;

   Note that instead of **in**, = can be used above, provided that station names are guaranteed to be unique. This query can also be written using a join, for example using

   select distinct id
   from trainhalts, station
   where trainhalts.stcode = station.stcode **and** station.name = 'THANE'
   and timein <> timeout

3. **select t.name from trainhalts th, train t,station st where th.id=t.id and st.stcode=th.stcode and th.timein is null and st.name='MUMBAI';**

4. **select** st.name **from** trainhalts th, train t,station st **where** th.id=t.id and st.stcode=th.stcode **and** t.name='CST-AMR_LOCAL' **order by** th.seqno;

5. **select distinct** name
   from train, trainhalts
   where train.id=trainhalts.id **and** trainhalts.seqno< 6 **and**

```
trainhalts.timeout<>trainhalts.timein
    trainhalts.stcode in (select stcode from station where
station.name='THANE');
```

University Schema

1. Student(id, name, dep_name, tot_credits)
2. Takes(id, course_id, sec_id, sem, year, grade)
3. Section(course_id, sec_id, sem, year, building, room_no, time_slot_id)
4. Time_slot(time_slot_id, day, start_time, end_time)
5. Classroom(building, room_no, capacity)
6. Teaches(i_id, course_id, sec_id, sem, year)
7. Instructor(i_id, name, dep_name, salary)
8. Dept(dep_name, building, budget)
9. Advisor(id, i_id)
10. Course(course_id, title, dep_name, credits)
11. Prereq(course_id, prereq_id)

Write the following simple SQL Queries on the University Schema

1. Find the names of all the students whose total credits are greater than 100
2. Find the course id and grades of all courses taken by any student named 'Tanaka'
3. Find the ID and name of instructors who have taught a course in the Comp. Sci. department, even if they are themselves not from the Comp. Sci. department. To test this query, make sure you add appropriate data, and include the corresponding insert statements along with your query.
4. Find the courses which are offered in both 'Fall' and 'Spring' semester (not necessarily in the same year).
5. Find the names of all the instructors from Comp. Sci. department
6. Find the course id and titles of all courses taught by an instructor named 'Srinivasan'
7. Find names of instructors who have taught at least one course in Spring 2009

Solutions: University Schema

1.
```
select name
from student
where tot_cred>100;
```

2. **select** c.course_id, c.id, c.grade
   from takes as c, student as s
   where c.id=s.id **and** s.name='Tanaka';

3. **select distinct** instructor.ID,name
   from teaches, instructor, course
   where instructor.ID=teaches.ID **and** course.course_id=teaches.course_id
   and course.dept_name='Comp. Sci.' ;

   As in the previous query, the **join using** clause can be used instead. Beware of using natural join, since that would force the instructor's department to match the course's department. Your test data should have included a case where the instructor is from a different department, but has taught a Comp. Sci. course.

4. (**select** course_id **from** section **where** semester='Fall')
   intersect
   (**select** course_id **from** section **where** semester='Spring');

   Alternative solution:

   select distinct a.course_id
   from section a, section b
   where a.course_id=b.course_id **and** a.semester='Fall' **and**
   b.semester='Spring';

   5.select name **from** instructor **where** dept_name = 'Comp. Sci.';

   select course.course_id, title
   from teaches, instructor, course
   where instructor.ID=teaches.ID **and** course.course_id=teaches.course_id
   and instructor.name='Srinivasan';

   Beware of using natural join for this query, since instructor and course both have a common attribute dept_name, and natural join makes these equal; as a result, courses taught by Srinivasan outside his department are excluded.

   You can write the query using the **using** clause though, as

   select course_id, title
   from (teaches **join** instructor using (ID)) **join** course **using** (course_id)
   where name='Srinivasan'

   and can even use the clause
   on (teaches.course_id = instructor.course_id)
   in place of the **using** clause above.

8.select distinct name
from instructor, teaches
where instructor.ID = teaches.ID **and** teaches.semester = 'Spring' **and** teaches.year = '2009'

Write the following SQL Queries on the University Schema (use nested queries)

1. Find the id and title of all courses which do not require any prerequisites.
2. Find the names of students who have not taken any biology dept courses
3. Give a 10% hike to all instructors
4. Increase the tot_creds of all students who have taken the course titled "Genetics" by the number of credits associated with that course.
5. For all instructors who are advisors of at least 2 students, increase their salary by 50000.
6. Set the credits to 2 for all courses which have less than 5 students taking them (across all sections for the course, across all years/semesters).

1. **select** course_id,title **from** course
   where course_id **not in** (**select** course_id **from** prereq);

2. **select** name **from** student
   where not exists (**select** * **from** takes, course
   where takes.course_id=course.course_id
   and course.dept_name = 'Biology'
   and takes.ID = student.ID);

   Note that many people mistakenly write this query as finding students who have taken a course whose dept_name <> Biology; it should be clear that this result may include students who have taken non-Biology courses in addition to Biology courses.

3. **update** instructor **set** salary = (1.1 * salary);

4. **update** student
   set tot_cred=tot_cred + (**select** course.credits **from** course **where** course.title = 'Genetics')
   where id in (**select** id **from** takes, course
   where takes.course_id=course.course_id and course.title = 'Genetics');

5. **update** instructor **set** salary = salary + 50000 **where** instructor.ID in (**select** i_id **from** advisor

group by i_id
having count(s_id) >=2);

Alternatively:
update instructor **set** salary = salary + 50000
where 2 <= (**select count**(*) **from** advisor **where** i_id = instructor.ID)

6. **update** course **set** credits=2 where course_id in (**select** course_id from takes group by course_id **having** count(id)<=5);

1> Each offering of a course (i.e. a section) can have many Teaching assistants; each teaching assistant is a student. Extend the existing schema (Add/Alter tables) to accommodate this requirement.
2> According to the existing schema, one student can have only one advisor.
3> Alter the schema to allow a student to have multiple advisors and make sure that you are able to insert multiple advisors for a student.
4> Write SQL queries on the modified schema. You will need to insert data to ensure the query results are not empty.
5> Find all students who have more than 3 advisors
6> Find all students who are co-advised by Prof. Srinivas and Prof. Ashok.
7> Find students advised by instructors from different departments. etc.
8> Delete all information in the database which is more than 10 years old. Add data as necessary to verify your query.
9> Delete the course CS 101. Any course which has CS 101 as a prereq should remove CS 101 from its prereq set. Create a cascade constraint to enforce the above rule, and verify that it is working.

- Assuming that a student can be a teaching assistant for multiple courses, one solution is to create a table 'teaching_assistants' with the following structure:

create table teaching_assistants
(ID varchar(5),
course_id varchar(8),
sec_id varchar(8),
semester varchar(6),
year numeric(4,0),
primary key (ID, course_id, sec_id, semester, year),
foreign key (course_id,sec_id, semester, year) references section
on delete cascade,
foreign key (ID) references student
on delete cascade

);

- alter table advisor drop constraint advisor_pkey;
  alter table advisor add constraint addvisor_pkey primary key (s_id, i_id);

- select id,name from student where id in (select s_id from advisor group by s_id having count(s_id)>3);

-
- select name from student
  where ID in (select i_id from advisor,instructor where advisor.i_id = instructor.ID and instructor.name='Srinivasan')
  and ID in (select i_id from advisor,instructor where advisor.i_id = instructor.ID and instructor.name='Ashok');

-
  select distinct student.ID, student.name
  from student, advisor A1, instructor I1, advisor A2, instructor I2
  where student.ID = A1.s_id and student.ID = A2.s_id and
  A1.i_id = I1.ID and A2.i_id = I2.ID and I1.dept_name <> I2.dept_name;

  OR

  select distinct student.ID, student.name
  from (select advisor.s_id, advisor.i_id, dept_name
  from instructor,advisor where instructor.ID=advisor.i_id) as s,
  (select advisor.s_id, advisor.i_id, dept_name
  from instructor,advisor where instructor.ID=advisor.i_id) as t,
  student
  where s.s_id=student.ID and s.s_id=t.s_id and s.dept_name<>t.dept_name;
-

  This involves deleting tuples from takes, section and teaches relations and all their dependents.
  delete from takes where year < (select extract(year from current_date) - 10);
  delete from teaches where year < (select extract(year from current_date) - 10);
  delete from section where year < (select extract(year from current_date) - 10);
-

  DELETE FROM course WHERE course_id='CS-101';
  All entries in the prereq table with prereq_id = 'CS-101' should get deleted if an 'ON DELETE CASCADE' specification is present as part of the foreign key from prereq.prereq_id referencing course. Else it can be deleted using the query: delete from prereq where prereq_id='CS-101';

(A) Consider the Movie Database given below. The Primary keys and Foreign keys are in bold,

Movies(**mid**:string,movie_name:string,year:int,reviews:string,rating:int)
Director(**d_id** int,**mid**:string,movie_name:string,schedule:string);
Producer(**pid**:int,**d_id**:int,budget:string);
Playin(actor_name:string,casttype:string)
Music_director(**music_id**:int,**d_id**:int,music_type:string,movie_name:string);

1. Create the above tables by properly specifying the primary keys and foreign keys.
2. Enter at least five tuples for each relation.
3. Update the schedule for a particular movie.
4. sort the name of the movie in descending order
5. Find the names of the movies in which a particular producer and director worked together
6. Write the query to ADD a new column salary in an existing movies table.
7. Display all the fields consisted in a movie name begining with "A"
8. Find a set of movies (title, reviews, rating) M, so that each movie in M is not dominated by any other movie in the database. Return the output on ascending order of title.
9. Delete the row of particular music type.
10. Drop the music_director table.

(B) Given below is the **Hospital Schema**,the Primary keys and Foreign keys are in bold,
.
Doctor(**d_id**,d_name,d_phone)
Patient(**p_id**,p_name,diagnosis,age)
Medicine(**med_id**,med_name)
Prescription(**p_id,d_id,med_id**)
Bed(**B_id,**ward_no)
Bed_Patient(**p_id,b_id**)

1. Create the above tables by properly specifying the primary keys and foreign keys.
2. Enter at least five tuples for each relation.
3 List all medicine name which starts from alphabet 'A'.
4 Add a column floor to Bed table
5.Find number of patients currently admitted
6.List all doctors along with the count of pateints treated by them
7. Display pateint name,diagnosis,medicine name and doctor name of all the patients who are admitted after specified date
8.Demonstrate update medicine name to a newer one
9. Delete a tuple from the Doctor table
10. Drop any one of the schema