Sub-Query 1.1

February-25/ sub-query/001.1 Time: 02:00hrs

MySQL

Diploma in Advance Computing

February 2025

|  |
| --- |
| Sub-Query 1.1 |
| 1. Display all employees whose salary is more than avg sal. |
| select sal from emp where sal > (select avg(sal) from emp); |
|  |
| 1. Display all employees who are getting highest salary in each deptno. |
| select \* from emp where (deptno, sal) in (select deptno, max(sal) from emp group by deptno) order by deptno; |
|  |
| 1. Display all employees who are getting second highest salary in each deptno. |
| select \* from emp where (deptno, sal) in (select deptno, max(sal) from emp where (deptno, sal) not in (select deptno, max(SAl) from emp group by deptno) group by deptno); |
|  |
| 1. Display ename, job, sal of all employees who are having same salary. |
| select ename, job, sal from emp a where exists (select true from emp b where a.sal=b.sal and a.empno <> b.empno) order by sal desc; |
|  |
| 1. Display ename, job, sal of all employees who are having same salary jobwise. |
| 1. select ename, job, sal from emp a where exists (select true from emp b where a.job=b.job and a.sal=b.sal and a.empno<> b.empno) order by job; 2. select ename, job, sal from emp where (job, sal) in (select job, sal from emp group by job, sal having count(\*) > 1) order by job, sal; |
|  |
| 1. Find the employees whose sal > avg sal of the same jobs. |
| select ename, job, sal from emp a where sal > (select avg(Sal) from emp b where a.job=b.job group by job) order by job; |
|  |
| 1. Find all jobs and maximum salary whose maximum salary is less than the average salary for all employees. |
| select job, max(sal) from emp group by job having max(Sal) < (select avg(sal) from emp); |
|  |
|  |
|  |
|  |
| ***Find the following relation*** |
| ***movies*** {movieID, title, movie\_year, movie\_time, language, releaseDate, releaseCountry}  ***actor*** {actorID, firstName, lastName, gender}  ***director*** {directorID, firstName, lastName}  ***movie\_director*** {directorID, movieID}  ***genres*** {genresID, genresTitle}  ***movie\_genres*** {movieID, genresID}  ***movie\_cast*** {actorID, movieID, role}  ***movie\_review*** {reviewID, reviewName}  ***movie\_rating*** {movieID, reviewID, reviewStars, numberOfRatings} |
|  |
| 1. write a SQL query to find the actors who played a role in the movie 'Annie Hall'. |
| select actor.actorID, movies.title, actor.firstName, actor.lastName, actor.gender from movies, actor , movie\_cast where movies.movieID=movie\_cast.movieID and movie\_cast.actorID= actor.actorID and movies.title='Annie Hall'; |
|  |
| 1. write a SQL query to find the director of a film that cast a role in 'Eyes Wide Shut'. Return director first name, last name. |
| select director.firstName, director.lastName from movies, director, movie\_director where movies.movieID= movie\_director.movieID and movie\_director.directorID=director.directorID and movies.title='Eyes Wide Shut'; |
|  |
| 1. write a SQL query to find the director who has directed more than 1 movie. Return director first name, last name. |
| select distinct director.firstname, director.lastName from director, movie\_director where movie\_director.directorid=director.directorid and director.directorid in (select directorID from movie\_director group by directorID having count(\*)>1); |
|  |
| 1. write a SQL query to find the director who has directed more than 1 movie. Return director first name, last name movie title, movie release date and movie language. |
| select movies.title, movies.language, movies.releaseDate, director.firstname, director.lastName from movies, director, movie\_director where movies.movieid=movie\_director.movieid and movie\_director.directorid=director.directorid and director.directorid in (select directorID from movie\_director group by directorID having count(\*)>1); |
|  |
| 1. write a SQL query to find those movies that have been released in countries other than the United Kingdom. Return movie title, movie year, movie time, and date of release, releasing country. |
| select movies.movieID, movies.title, movies.movie\_year Year, movies.movie\_time Duration, movies.language, movies.releaseDate, movies.releaseCountry from movies where movies.releaseCountry<>'UK'; |
|  |
|  |
|  |
| ***Find the following relation*** |
| ***movies*** {movieID, title, movie\_year, movie\_time, language, releaseDate, releaseCountry}  ***actor*** {actorID, firstName, lastName, gender}  ***director*** {directorID, firstName, lastName}  ***movie\_director*** {directorID, movieID}  ***genres*** {genresID, genresTitle}  ***movie\_genres*** {movieID, genresID}  ***movie\_cast*** {actorID, movieID, role}  ***movie\_review*** {reviewID, reviewName}  ***movie\_rating*** {movieID, reviewID, reviewStars, numberOfRatings} |
|  |
| 1. write a SQL query to find for movies whose reviewer is unknown. Return movie title, year, release date, director first name, last name, actor first name, last name. |
| select movies.title, movie\_review.reviewName from movies, movie\_review, movie\_rating where movies.movieID=movie\_rating.movieID and movie\_rating.reviewID=movie\_review.reviewID and movie\_review.reviewName is unknown; |
|  |
| 1. write a SQL query to find those movies directed by the director whose first name is Woody and last name is Allen. Return movie title. |
| select movies.title from movies, director, movie\_director where movies.movieID=movie\_director.movieID and movie\_director.directorID=director.directorID and (director.firstName, director.lastName) =('Woody', 'Allen'); |
|  |
| 1. write a SQL query to search for movies that do not have any ratings. Return movie title. |
| select Title from movies where not exists (select true from movie\_rating where movies.movieid = movie\_rating.movieid); |
|  |
| 1. write a SQL query to search for movies that do not have any ratings. Return movie ID and title. |
| select movieID, Title from movies where not exists (select true from movie\_rating where movies.movieID=movie\_rating.movieID); |
|  |
| 1. write a SQL query to search for movies those were released on same dates. |
| select \* from movies a where exists (select true from movies b where a.releaseDate=b.releaseDate and a.movieID<>b.movieID); |
|  |
| 1. write a SQL query to search all movies whose reviewStars is second highest. |
| select \* from movies where movieID in (select movieID from movie\_rating where reviewStars = (select max(reviewStars) from movie\_rating where reviewStars < (select max(reviewStars) from movie\_rating))); |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| 1. Find the name of the employee who earns the highest salary. |
| select ename from emp where sal = (select max(sal) from emp); |
|  |
| 1. Find the name of the employee who earns the highest salary for every job, sort the rows according to their job and salaries. |
| select ename, job, sal from emp where (job, sal) in (select job, max(sal) from emp group by job) order by job, sal; |
|  |
| 1. List employees who work in the same department as ‘John’. |
| select \* from emp where deptno = (select deptno from emp where ename='jones'); |
|  |
| 1. Write SQL query to display employee where only female employees are working. Return employee name, job, gender and her department number. |
| select ename, job, gender, deptno from emp f where not exists (select true from emp m where m.deptno=f.deptno and m.gender='m'); |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| ***product*** {productid, productname, supplierid, categoryid, quantityperunit, unitprice, unitsinstock, unitsonorder, reorderlevel, discontinued}  ***customer*** {custid, name, address, city, state, zip, area, phone, repid, creditlimit, comments}  ***ord*** {ordid, orderdate, commplan, custid, shipdate, statusdate, status, total} |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |