#### Sample Table – Worker

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **WORKER\_ID** | **FIRST\_NAME** | **LAST\_NAME** | **SALARY** | **JOINING\_DATE** | **DEPARTMENT** |
| 001 | Monika | Arora | 100000 | 2014-02-20 09:00:00 | HR |
| 002 | Niharika | Verma | 80000 | 2014-06-11 09:00:00 | Admin |
| 003 | Vishal | Singhal | 300000 | 2014-02-20 09:00:00 | HR |
| 004 | Amitabh | Singh | 500000 | 2014-02-21 09:00:00 | Admin |
| 005 | Vivek | Bhati | 500000 | 2014-06-11 09:00:00 | Admin |
| 006 | Vipul | Diwan | 200000 | 2014-06-11 09:00:00 | Account |
| 007 | Satish | Kumar | 75000 | 2014-01-20 09:00:00 | Account |
| 008 | Geetika | Chauhan | 90000 | 2014-04-11 09:00:00 | Admin |

CREATE TABLE Worker (

WORKER\_ID INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

FIRST\_NAME CHAR(25),

LAST\_NAME CHAR(25),

SALARY INT(15),

JOINING\_DATE DATETIME,

DEPARTMENT CHAR(25)

);

INSERT INTO Worker

(WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, DEPARTMENT) VALUES

(001, 'Monika', 'Arora', 100000, '14-02-20 09.00.00', 'HR'),

(002, 'Niharika', 'Verma', 80000, '14-06-11 09.00.00', 'Admin'),

(003, 'Vishal', 'Singhal', 300000, '14-02-20 09.00.00', 'HR'),

(004, 'Amitabh', 'Singh', 500000, '14-02-20 09.00.00', 'Admin'),

(005, 'Vivek', 'Bhati', 500000, '14-06-11 09.00.00', 'Admin'),

(006, 'Vipul', 'Diwan', 200000, '14-06-11 09.00.00', 'Account'),

(007, 'Satish', 'Kumar', 75000, '14-01-20 09.00.00', 'Account'),

(008, 'Geetika', 'Chauhan', 90000, '14-04-11 09.00.00', 'Admin');

#### Q-1. Write an SQL query to fetch “FIRST\_NAME” from Worker table using the alias name as <WORKER\_NAME>.

**Ans.**

The required query is:

Select FIRST\_NAME AS WORKER\_NAME from Worker;

#### Q-2. Write an SQL query to fetch “FIRST\_NAME” from Worker table in upper case.

**Ans.**

The required query is:

Select upper(FIRST\_NAME) from Worker;

#### Q-3. Write an SQL query to fetch unique values of DEPARTMENT from Worker table.

**Ans.**

The required query is:

Select distinct DEPARTMENT

from Worker;

#### Q-4. Write an SQL query to print the first three characters of  FIRST\_NAME from Worker table.

**Ans.**

The required query is:

Select substring(FIRST\_NAME,1,3) from Worker;

#### Q-5. Write an SQL query to find the position of the alphabet (‘a’) in the first name column ‘Amitabh’ from Worker table.

**Ans.**

The required query is:

Select INSTR(FIRST\_NAME, BINARY'a') from Worker where FIRST\_NAME = 'Amitabh';

**Notes.**

* The INSTR method is in case-sensitive by default.
* Using Binary operator will make INSTR work as the case-sensitive function.

#### Q-6. Write an SQL query to print the FIRST\_NAME from Worker table after removing white spaces from the right side.

**Ans.**

The required query is:

Select RTRIM(FIRST\_NAME) from Worker;

#### Q-7. Write an SQL query to print the DEPARTMENT from Worker table after removing white spaces from the left side.

**Ans.**

The required query is:

Select LTRIM(DEPARTMENT) from Worker;

#### Q-8. Write an SQL query that fetches the unique values of DEPARTMENT from Worker table and prints its length.

**Ans.**

The required query is:

Select distinct length(DEPARTMENT) from Worker;

#### Q-9. Write an SQL query to print the FIRST\_NAME from Worker table after replacing ‘a’ with ‘A’.

**Ans.**

The required query is:

Select REPLACE(FIRST\_NAME,'a','A') from Worker;

#### Q-10. Write an SQL query to print the FIRST\_NAME and LAST\_NAME from Worker table into a single column COMPLETE\_NAME. A space char should separate them.

**Ans.**

The required query is:

Select CONCAT(FIRST\_NAME, ' ', LAST\_NAME) AS 'COMPLETE\_NAME' from Worker;

#### Q-11. Write an SQL query to print all Worker details from the Worker table order by FIRST\_NAME Ascending.

**Ans.**

The required query is:

Select \* from Worker order by FIRST\_NAME asc;

#### Q-12. Write an SQL query to print all Worker details from the Worker table order by FIRST\_NAME Ascending and DEPARTMENT Descending.

**Ans.**

The required query is:

Select \* from Worker

order by FIRST\_NAME asc,DEPARTMENT desc;

#### Q-13. Write an SQL query to print details for Workers with the first name as “Vipul” and “Satish” from Worker table.

**Ans.**

The required query is:

Select \* from Worker where FIRST\_NAME in ('Vipul','Satish');

#### Q-14. Write an SQL query to print details of workers excluding first names, “Vipul” and “Satish” from Worker table.

**Ans.**

The required query is:

Select \* from Worker where FIRST\_NAME not in ('Vipul','Satish');

#### Q-15. Write an SQL query to print details of Workers with DEPARTMENT name as “Admin”.

**Select \* from worker**

**Where deprtment=’admin’;**

**Select \* from worker**

**Where deprtment in (’admin’);**

**Ans.**

The required query is:

Select \* from Worker where DEPARTMENT like 'Admin';

#### Q-16. Write an SQL query to print details of the Workers whose FIRST\_NAME contains ‘a’.

**Ans.**

The required query is:

Select \* from Worker where FIRST\_NAME like '%a%';

#### Q-17. Write an SQL query to print details of the Workers whose FIRST\_NAME ends with ‘a’.

**Ans.**

The required query is:

Select \* from Worker where FIRST\_NAME like '%a';

#### Q-18. Write an SQL query to print details of the Workers whose FIRST\_NAME ends with ‘h’ and contains six alphabets.

**Ans.**

The required query is:

Select \* from Worker where FIRST\_NAME like '\_\_\_\_\_h';

#### Q-19. Write an SQL query to print details of the Workers whose SALARY lies between 100000 and 500000.

**Ans.**

The required query is:

Select \* from Worker where SALARY between 100000 and 500000;

#### Q-20. Write an SQL query to print details of the Workers who have joined in Feb’2014.

**Select \* from worker**

**Where joining\_date = ‘2014-02-%’;**

**Ans.**

The required query is:

Select \* from Worker where year(JOINING\_DATE) = 2014 and month(JOINING\_DATE) = 2;

#### Q-21. Write an SQL query to fetch the count of employees working in the department ‘Admin’.

#### Select count() from worker where department =’admin’;

**Ans.**

The required query is:

SELECT COUNT() FROM worker WHERE DEPARTMENT = 'Admin';

#### Q-22. Write an SQL query to fetch worker names with salaries >= 50000 and <= 100000.

#### SELECT (FIRSTNAME ||’ ‘|| LASTNME) AS WORKERNAME,SALARIES

#### FROM WORKER

#### WHERE SALARIES BETWEEN 50000 AND 100000

**Ans.**

The required query is:

SELECT CONCAT(FIRST\_NAME, ' ', LAST\_NAME) As Worker\_Name, Salary

FROM worker

WHERE WORKER\_ID IN

(SELECT WORKER\_ID FROM worker

WHERE Salary BETWEEN 50000 AND 100000);

#### Q-23. Write an SQL query to fetch the no. of workers for each department in the descending order.

**Ans.**

The required query is:

SELECT DEPARTMENT, count(WORKER\_ID) No\_Of\_Workers

FROM worker

GROUP BY DEPARTMENT

ORDER BY No\_Of\_Workers DESC;

#### 24.Write an SQL query to show the top n (say 10) records of a table.

**Ans.**

Following MySQL query will return the top n records using the LIMIT method:

SELECT \* FROM Worker ORDER BY Salary DESC LIMIT 10;

#### Write an SQL query to fetch the list of employees with the same salary.

**Ans.**

The required query is:

Select distinct W.WORKER\_ID, W.FIRST\_NAME, W.Salary

from Worker W, Worker W1

where W.Salary = W1.Salary

and W.WORKER\_ID != W1.WORKER\_ID;

#### Write an SQL query to show the second highest salary from a table.

**SELECT MAX(SALARY)AS SALARY FROM WORKER;**

**Ans.**

The required query is:

Select max(Salary) from Worker

where Salary not in (Select max(Salary) from Worker);

#### Write an SQL query to fetch the first 50% records from a table.

**Ans.**

The required query is:

SELECT \*

FROM WORKER

WHERE WORKER\_ID <= (SELECT count(WORKER\_ID)/2 from Worker);

#### Write an SQL query to show all departments along with the number of people in there.

**Ans.**

The following query returns the expected result:

SELECT DEPARTMENT, COUNT(DEPARTMENT) as 'Number of Workers' FROM Worker

GROUP BY DEPARTMENT;

#### Write an SQL query to fetch the departments that have less than five people in it.

**Ans.**

The required query is:

SELECT DEPARTMENT, COUNT(WORKER\_ID) as 'Number of Workers' FROM Worker

GROUP BY DEPARTMENT HAVING COUNT(WORKER\_ID) < 5;

#### Write an SQL query to show the last record from a table.

**Ans.**

The following query will return the last record from the Worker table:

Select \* from Worker where WORKER\_ID = (SELECT max(WORKER\_ID) from Worker);

#### Write an SQL query to print details of the Workers who are also Managers.

**Ans.**

The required query is:

SELECT \*

FROM Worker W

INNER JOIN Title T

ON W.WORKER\_ID = T.WORKER\_REF\_ID

AND T.WORKER\_TITLE in ('Manager');

#### Write an SQL query to show only odd rows from a WORKER table.

#### SELECT \* FROM WORKER WHERE (WORKER\_ID%2)!=0;

**Ans.**

The required query is:

SELECT \* FROM Worker WHERE (WORKER\_ID%2) != 0;

#### Write an SQL query to clone a new table from another table.

INSERT INTO WorkerClone SELECT \* FROM Worker;

SELECT \* FROM WorkerCLONE;

#### Write an SQL query to fetch intersecting records of two tables.

**Ans.**

The required query is:

(SELECT \* FROM Worker)

INTERSECT

(SELECT \* FROM WorkerClone);

#### Q-30. Write an SQL query to show records from one table that another table does not have.

**Ans.**

The required query is:

SELECT \* FROM Worker

MINUS

SELECT \* FROM Title;

#### Write an SQL query to show the current date and time.

**Ans.**

Following MySQL query returns the current date:

SELECT CURDATE();

Following MySQL query returns the current date and time:

SELECT NOW();

Following SQL Server query returns the current date and time:

SELECT getdate();

Following Oracle query returns the current date and time:

SELECT SYSDATE FROM DUAL;

SELECT current\_TIME;

#### Write an SQL query to determine the nth (say n=5) highest salary from a table.

#### SELECT SALARY FROM WORKER

#### OREDER BY SALARY DESC LIMIT5;

SELECT Salary FROM Worker ORDER BY Salary DESC LIMIT 5;

#### Write an SQL query to show one row twice in results from a table.

select FIRST\_NAME, DEPARTMENT from worker W where W.WORKER\_ID='001'

union all

select FIRST\_NAME, DEPARTMENT from Worker W1 where W1.WORKER\_ID='001';

#### Write an SQL query to show the last record from a table.

#### SELECT \* FROM WORKER WHERE WORKER\_ID=(SELECT MAX(WORKER\_ID) FROM WORKER)

**Ans.**

The following query will return the last record from the Worker table:

Select \* from Worker where WORKER\_ID = (SELECT max(WORKER\_ID) from Worker);

#### Q-43. Write an SQL query to fetch the first row of a table.

**Ans.**

The required query is:

Select \* from Worker where WORKER\_ID = (SELECT min(WORKER\_ID) from Worker);

#### Q-49. Write an SQL query to fetch departments along with the total salaries paid for each of them.

**Ans.**

The required query is:

 SELECT DEPARTMENT, sum(Salary) from worker group by DEPARTMENT;

#### Q-50. Write an SQL query to fetch the name of workers who earn the highest salary.

#### Select first\_name ,max(salary) from worker

**Ans.**

The required query is:

SELECT FIRST\_NAME, SALARY from Worker WHERE SALARY=(SELECT max(SALARY) from Worker);

#### Write an SQL query to fetch three min salaries from a table

#### Select salary from worker order by salary limit 3

#### Write an SQL query to print the name of employees having the highest salary in each department.

#### Select first\_name,max(salary) from worker group by department;

**DISPLAYING TABLES:**

**SQL> select \*from salesman;**

|  |  |  |  |
| --- | --- | --- | --- |
| **SALESMAN\_ID** | **NAME** | **CITY** | **COMMISSION** |
| 1000 | adarsh | bangalore | 2 |
| 1001 | bharath | mysore | 8 |
| 1002 | chandru | bangalore | 3 |
| 1003 | dinesh | pune | 1 |
| 1004 | eshwar | chennai | 9 |
| 1005 | kiran | hyd | 4 |
| 1006 | raj | mumbai | 5 |

**SQL> select \*from customer;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUSTOMER\_ID** | **CUST\_NAME** | **CITY** | **GRADE** | **SALESMAN\_ID** |
| 10 | farhan | bangalore | 2 | 1000 |
| 11 | ganesh | bangalore | 4 | 1000 |
| 12 | harsha | mysore | 8 | 1001 |
| 13 | Indra | mysore | 2 | 1001 |
| 14 | jagadish | pune | 10 | 1003 |
| 15 | karthik | chennai | 9 | 1004 |
| 16 | ram | mysore | 4 | 1002 |
| 17 | mahesh | pune | 8 | 1002 |
| 18 | akshay | chennai | 8 | 1002 |

**SQL> select \*from orders;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ORD\_NO** | **PURCHASE\_AMT** | **ORD\_DATE** | **CUSTOMER\_ID** | **SALESMAN\_ID** |
| 1 | 10000 | 12-MAR-10 | 10 | 1000 |
| 2 | 20000 | 12-MAR-10 | 11 | 1000 |
| 3 | 40000 | 12-MAR-10 | 12 | 1001 |
| 4 | 15000 | 10-JAN-10 | 13 | 1001 |
| 5 | 5000 | 10-JAN-10 | 14 | 1003 |

1. **Count the customers with grades above Bangalore’s average.**

**Select grade,count(customer\_id)**

**from customer**

**group by grade**

**having grade>(Select Avg(grade) from customer where city=’bangalore’) ;**

**g c**

**4 2**

**8 3**

**9 1**

**10 1**

**SELECT** grade, **COUNT** (customer\_id)

**FROM** CUSTOMER

**GROUP BY** grade

**HAVING** grade >(SELECT AVG(grade)

**FROM** CUSTOMER

**WHERE** city = 'bangalore');

1. **Find the name and numbers of all salesman who had more than one customer**

**SELECT** salesman\_id,name

**FROM** SALESMAN **WHERE** salesman\_id in ( **SELECT** salesman\_id

**FROM** CUSTOMER

**GROUP BY** salesman\_id

**HAVING COUNT**(customer\_id)>1)

**4.Create a view that finds the salesman who has the customer with the highest order of a day.**

Max(purchse\_a

SQL>**CREATE VIEW** elitsalesman

AS **SELECT s**.salesman\_id,s.name ,PURCHASE\_AMT,ord\_date from ORDERS o, SALESMAN s

**WHERE**  s.salesman\_id=o.salesman\_id

and PURCHASE\_AMT in(**SELECT MAX** (Purchase\_Amt)

**FROM** orders group by ord\_date);

**SQL> select \*from elitsalesman1;**

**3.List all the salesman and indicate those who have customers in their cities**

SQL>**SELECT** salesman.salesman\_id, name, cust\_name, commission

**FROM s**alesman, customer

**WHERE** salesman.city = customer.city

**5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.**

**SQL> DELETE** from **SALESMAN** **WHERE** salesman\_id=1000;

3rd program

create table actor( act\_id varchar2(4) primary key, act\_name varchar2(20) not null,act\_gender varchar(6));

INSERT into actor (act\_id , act\_name,act\_gender) VALUES

('a1' , 'sharuk' , 'male'),

('a2','amir','male'),

('a3','salman','male'),

('a4','madhuri','female'),

('a5','priyanka','female'),

('a6','nasir','male');

create table director(dir\_id varchar2(4) primary key,dir\_name varchar2(20) not null,dir\_phone number(11));

INSERT into director(dir\_id,dir\_name,dir\_phone) VALUES

('d1','hitchcock',45),

('d2','stevan',46),

('d3','karan',47),

('d4','ramhopal',48),

('d5','anurag',49);

create table movies(mov\_id varchar2(4) primary key,mov\_title varchar(20) not null,mov\_year int,mov\_lang varchar2(10), dir\_id varchar2(4) references director(dir\_id));

INSERT INTO movies(mov\_id,mov\_title, mov\_year ,mov\_lang, dir\_id) VALUES

('m1','ddlj',1999,'hindi','d1'),

('m2','bazigar',2002,'hindi','d2'),

('m3','darr',2016,'hindi','d1'),

('m4','dil',1990,'hindi','d2'),

('m5','pk', 2016,'hindi','d3'),

('m6','sulthan', 2016,'hindi','d1'),

('m7','devdas', 2002,'hindi','d4'),

('m8','marykom', 2016,'hindi','d3');

create table movie\_cast(act\_id varchar2(4) references actor(act\_id),mov\_id varchar2(4)references movies(mov\_id),role varchar2(10),primary key(act\_id,mov\_id));

INSERT INTO movie\_cast(act\_id,mov\_id,role) VALUES

('a1','m1','hero'),

('a1','m2','hero'),

('a1','m3','hero'),

('a2','m4','hero'),

('a2','m5','costar'),

('a3','m6','hero'),

('a4','m7','heroin'),

('a5','m1','costar'),

('a6','m1','support'),

('a4','m2','heroin');

create table rating(mov\_id varchar2(4) references movies(mov\_id), rev\_stars number(5,2) ,primary key(mov\_id));

INSERT INTO rating(mov\_id, rev\_stars) VALUES

('m1',1),

('m2',2),

('m3',4),

('m5',6),

('m6',5),

('m7',1),

('m8',4),

('m4',2);

**SQL> select \*from actor; SQL> select \*from director;**

|  |  |  |
| --- | --- | --- |
| **DIR\_ID** | **DIR\_NAME** | **DIR\_PHONE** |
| d1 | Hitchcock | 9845612345 |
| d2 | steven spielberg | 9845612346 |
| d3 | karan | 9845612347 |
| d4 | ramgopal | 9845612348 |
| d5 | anurag | 9845612349 |

|  |  |  |
| --- | --- | --- |
| **ACT\_ ID** | **ACT\_NAME** | **ACT\_GENDER** |
| a1 | Sharuk | male |
| a2 | amir | male |
| a3 | Salman | male |
| a4 | Madhuri | female |
| a5 | Priyanka | female |
| a6 | Nasir | male |

**SQL> select \*from movies;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MOV\_ID** | **MOV\_TITLE** | **MOV\_YEAR** | **MOV\_LANG** | **DIR\_id** |
| m1 | ddlj | 1999 | hindi | d1 |
| m2 | Bazigar | 2002 | hindi | d2 |
| m3 | darr | 2016 | hindi | d1 |
| m4 | Dil | 1990 | hindi | d2 |
| m5 | pk | 2016 | hindi | d3 |
| m6 | sulthan | 2016 | hindi | d1 |
| m7 | devdas | 2002 | hindi | d4 |
| m8 | marykom | 2016 | hindi | d3 |

**SQL> select \*from movie\_cast; SQL> select \*from rating;**

|  |  |  |
| --- | --- | --- |
| **ACT\_ID** | **MOV\_ID** | **ROLE** |
| a1 | m1 | hero |
| a1 | m2 | hero |
| a1 | m3 | hero |
| a2 | m4 | hero |
| a2 | m5 | costar |
| a3 | m6 | hero |
| a4 | m7 | heroine |
| a5 | m1 | costar |
| a6 | m1 | support |
| a4 | m2 | heroine |

|  |  |
| --- | --- |
| **MOV\_ID** | **REV\_STARS** |
| m1 | .1 |
| m2 | 2 |
| m3 | 4 |
| m5 | 6 |
| m6 | .5 |
| m7 | 1 |
| m8 | .4 |
| m4 | .2 |

**Queries:**

1. **List the titles of all movies directed by ‘Hitchcock’**.

**SQL>SELECT** m.mov\_id,m.mov\_title,m.dir\_id,d.dir\_name

**FROM** movies m,director d

**WHERE** m.dir\_id=d.dir\_id and d.dir\_name='hitchcock';

|  |  |  |  |
| --- | --- | --- | --- |
| **MOV\_ID** | **MOV\_TITLE** | **DIR\_ID** | **DIR\_NAME** |
| m6 | ddlj | d1 | hitchcock |
| m3 | darr | d1 | hitchcock |
| m1 | sulthan | d1 | hitchcock |

**2.Find the movie names where one or more actors acted in two or more movies**.

SQL>**SELECT** mov\_title, act\_name, role

**FROM** movies

**JOIN** movie\_cast

**ON** movie\_cast.mov\_id=movies.mov\_id

**JOIN** actor

**ON** movie\_cast.act\_id=actor.act\_id

**WHERE** actor.act\_id IN (

**SELECT** act\_id

**FROM** movie\_cast

**GROUP BY** act\_id HAVING COUNT(\*)>=2);

|  |  |  |
| --- | --- | --- |
| **MOV\_TITLE** | **ACT\_NAME** | **ROLE** |
| ddlj | sharuk | hero |
| bazigar | sharuk | hero |
| darr | sharuk | hero |
| dil | amir | hero |
| pk | amir | costar |
| devdas | madhuri | heroine |
| bazigar | madhuri | heroine |

**3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**

**SQL**>(**SELECT** a.act\_id,a.act\_name

**FROM** actor a

**JOIN** movie\_cast

**ON** a.act\_id=movie\_cast.act\_id

**JOIN** movies

**ON** movie\_cast.mov\_id=movies.mov\_id

**WHERE** mov\_year< 2000 )

**INTERSECT**

(**SELECT** a.act\_id,a.act\_name

**FROM** actor a

**JOIN** movie\_cast

**ON** a.act\_id=movie\_cast.act\_id

**JOIN** movies

**ON** movie\_cast.mov\_id=movies.mov\_id

**WHERE** mov\_year> 2015 );

|  |  |
| --- | --- |
| **ACT\_ID** | **ACT\_NAME** |
| a1 | sharuk |
| a2 | amir |

**4. Find the title of movies and number of stars for each movie that has at least on rating and find the highest number of stars that movie received. Sort the result by movie title.**

**SQL> SELECT** m.mov\_id,m.mov\_title, MAX(rev\_stars) **FROM** Rating

**JOIN** Movies m on m.mov\_id=Rating.mov\_id

and rating.rev\_stars>=1

**GROUP BY** m.mov\_id,m.mov\_title

**ORDER BY** m.mov\_title;

**5.Update rating of all movies directed by ‘Steven Spielberg’ to 5.**

**SQL>UPDATE** rating set rev\_stars=5

**WHERE** mov\_id in( **SELECT** m.mov\_id **FROM** movies m ,director d

**WHERE** d.dir\_id=m.dir\_id and d. dir\_name='steven spielberg');

Exercise example.

Table ***Users***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **First\_Name** | **Last\_Name** | **Birth\_Date** | **Gender** | **Join\_Date** |
| Sophie | Lee | Jan-05-1960 | F | Apr-05-2015 |
| Richard | Brown | Jan-07-1975 | M | Apr-05-2015 |
| Jamal | Santo | Oct-08-1983 | M | Apr-09-2015 |
| Casey | Healy | Sep-20-1969 | M | Apr-09-2015 |
| Jill | Wilkes | Nov-20-1979 | F | Apr-15-2015 |

1. Which of the following SQL statement is valid?  
a) SELECT DISTINCT \* FROM Users;  
b) SELECT DISTINCT First\_Name FROM Users;  
c) SELECT DISTINCT First\_Name Last\_Name FROM Users;

2. What's the result of the following query?  
SELECT DISTINCT Join\_Date From Users;

3. What's the result of the following query?  
SELECT DISTINCT Gender, Join\_Date From Users;

1. b)

2. The result is:

|  |
| --- |
| **Join\_Date** |
| **Apr-05-2015** |
| **Apr-09-2015** |
| **Apr-15-2015** |

3. The result is:

|  |  |
| --- | --- |
| **Gender** | **Join\_Date** |
| **F** | **Apr-05-2015** |
| **M** | **Apr-05-2015** |
| **M** | **Apr-09-2015** |
| **F** | **Apr-15-2015** |

1. Which of the following SQL statement is valid? (There can be more than one answer)  
   a) SELECT \* FROM Users WHERE Gender = 'M';  
   b) SELECT \* WHERE Gender = 'M' FROM Users;  
   c) SELECT Gender= 'M' FROM Users;  
   d) SELECT Gender FROM Users WHERE Last\_Name = 'Wilkes';

Table ***Users***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **First\_Name** | **Last\_Name** | **Birth\_Date** | **Gender** | **Join\_Date** |
| Sophie | Lee | Jan-05-1960 | F | Apr-05-2015 |
| Richard | Brown | Jan-07-1975 | M | Apr-05-2015 |
| Jamal | Santo | Oct-08-1983 | M | Apr-09-2015 |
| Casey | Healy | Sep-20-1969 | M | Apr-09-2015 |
| Jill | Wilkes | Nov-20-1979 | F | Apr-15-2015 |

1. Which of the following SQL statement is valid? (There can be more than one answer)  
a) SELECT First\_Name AND Last\_Name FROM Users;  
b) SELECT First\_Name, Last\_Name FROM Users WHERE Join\_Date > 'Apr-01-2015' AND Birth\_Date < 'Jan-01-1980';  
c) SELECT First\_Name OR User\_Name FROM Users;  
d) SELECT \* FROM Users WHERE Last\_Name = 'Brown' AND Gender = 'F';

2. How many records will be returned by the following query?  
SELECT \* FROM Users WHERE Gender = 'M' AND Join\_Date = 'Apr-09-2015';

3. How many records will be returned by the following query?  
SELECT \* FROM Users WHERE Gender = 'M' OR Join\_Date = 'Apr-05-2015';

1. b), d)

2. 2 records are returned. They are,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **First\_Name** | **Last\_Name** | **Birth\_Date** | **Gender** | **Join\_Date** |
| **Jamal** | **Santo** | **Oct-08-1983** | **M** | **Apr-09-2015** |
| **Casey** | **Healy** | **Sep-20-1969** | **M** | **Apr-09-2015** |

3. 4 records are returned. They are,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **First\_Name** | **Last\_Name** | **Birth\_Date** | **Gender** | **Join\_Date** |
| **Sophie** | **Lee** | **Jan-05-1960** | **F** | **Apr-05-2015** |
| **Richard** | **Brown** | **Jan-07-1975** | **M** | **Apr-05-2015** |
| **Jamal** | **Santo** | **Oct-08-1983** | **M** | **Apr-09-2015** |
| **Casey** | **Healy** | **Sep-20-1969** | **M** | **Apr-09-2015** |

1. Which of the following SQL statement is valid? (There can be more than one answer)  
a) SELECT Gender FROM Users IN ('M');  
b) SELECT \* FROM Users HAVING Gender IN ('M','F');  
c) SELECT First\_Name, Last\_Name FROM Users WHERE Gender IN ('M','F');  
d) SELECT DISTINCT First\_Name, Last\_Name WHERE Gender IN ('M','F');

2. How many records will be returned by the following query?  
SELECT \* FROM Users WHERE Join\_Date IN ('Apr-05-2015','Apr-15-2015');

3. How many records will be returned by the following query?  
SELET \* FROM Users WHERE Gender IN ('M') AND Join\_Date = 'Apr-15-2015');

1. c)

2. 3 records are returned. They are,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **First\_Name** | **Last\_Name** | **Birth\_Date** | **Gender** | **Join\_Date** |
| **Sophie** | **Lee** | **Jan-05-1960** | **F** | **Apr-05-2015** |
| **Richard** | **Brown** | **Jan-07-1975** | **M** | **Apr-05-2015** |
| **Jill** | **Wilkes** | **Nov-20-1979** | **F** | **Apr-15-2015** |

3. 0 record is returned. There is no record that satisfies the **WHERE** condition.

1. Which of the following SQL statement is valid? (There can be more than one answer)  
a) SELECT \* FROM User\_Sales ORDER BY Sales;  
b) SELECT \* FROM User\_Sales ORDER BY Last\_Name DESC;  
c) SELECT \* FROM User\_Sales ORDER BY First\_Name WHERE Sales > 100;  
d) SELECT \* FROM User\_Sales ORDER BY Last\_Name, First\_Name;

2. What is the result of the following query?  
SELECT \* FROM User\_Sales WHERE Join\_Date IN ('Apr-05-2015','Apr-15-2015') ORDER BY Sales;

3. What is the result of the following query?  
SELECT \* FROM User\_Sales ORDER BY Join\_Date DESC, Sales;

1. a), b), d)

2. The result is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **First\_Name** | **Last\_Name** | **Gender** | **Join\_Date** | **Sales** |
| **Richard** | **Brown** | **M** | **Apr-05-2015** | **200** |
| **Jill** | **Wilkes** | **F** | **Apr-15-2015** | **210** |
| **Sophie** | **Lee** | **F** | **Apr-05-2015** | **500** |

3. The result is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **First\_Name** | **Last\_Name** | **Gender** | **Join\_Date** | **Sales** |
| **Jill** | **Wilkes** | **F** | **Apr-15-2015** | **210** |
| **Casey** | **Healy** | **M** | **Apr-09-2015** | **80** |
| **Jamal** | **Santo** | **M** | **Apr-09-2015** | **350** |
| **Richard** | **Brown** | **M** | **Apr-05-2015** | **200** |
| **Sophie** | **Lee** | **F** | **Apr-05-2015** | **500** |

Table ***Region\_Sales***

|  |  |  |  |
| --- | --- | --- | --- |
| **Region** | **Year** | **Orders** | **Total\_Sales** |
| West | 2013 | 1560 | 325000 |
| West | 2014 | 1820 | 380000 |
| North | 2013 | 790 | 148000 |
| North | 2014 | 995 | 185000 |
| East | 2013 | 1760 | 375000 |
| East | 2014 | 2220 | 450000 |
| South | 2013 | 1790 | 388000 |
| South | 2014 | 1695 | 360000 |

1. Which of the following order is correct for a SQL statement?  
a) SELECT...FROM...WHERE...GROUP BY...ORDER BY  
b) SELECT...FROM...ORDER BY...WHERE...GROUP BY  
c) SELECT...FROM...WHERE...ORDER BY...GROUP BY  
d) SELECT...WHERE...FROM...GROUP BY...ORDER BY

2. Write a SQL statement that calculates the total dollar sales amount for each region. What is the result?

3. Write a SQL statement that calculates the average annual dollar sales amount for just the East region and the West region. What is the result?

1. a)

2. SELECT Region, SUM(Total\_Sales) FROM Region\_Sales GROUP BY Region;

|  |  |
| --- | --- |
| **Region** | **SUM(Total\_Sales)** |
| **West** | **705000** |
| **North** | **333000** |
| **East** | **825000** |
| **South** | **748000** |

3. SELECT AVG(Total\_Sales) FROM Region\_Sales WHERE Region IN ('East','West');

|  |
| --- |
| **AVG(Total\_Sales)** |
| **382500** |

Table ***Region\_Sales***

|  |  |  |  |
| --- | --- | --- | --- |
| **Region** | **Year** | **Orders** | **Total\_Sales** |
| West | 2013 | 1560 | 325000 |
| West | 2014 | 1820 | 380000 |
| North | 2013 | 790 | 148000 |
| North | 2014 | 995 | 185000 |
| East | 2013 | 1760 | 375000 |
| East | 2014 | 2220 | 450000 |
| South | 2013 | 1790 | 388000 |
| South | 2014 | 1695 | 360000 |

1. Which of the following order is correct for a SQL statement?  
a) SELECT...FROM...ORDER BY...WHERE...HAVING  
b) SELECT...FROM...WHERE...ORDER BY...HAVING  
c) SELECT...WHERE...FROM...HAVING...ORDER BY  
d) SELECT...FROM...WHERE...HAVING...ORDER BY

2. What is the result of the following SQL statement?  
SELECT Region, SUM(Orders) FROM Region\_Sales GROUP BY Region HAVING SUM(Orders) > 2500;

3. What is the result of the following SQL statement?  
SELECT Region, SUM(Orders) FROM Region\_Sales WHERE Total\_Sales < 385000 GROUP BY Region HAVING SUM(Orders) > 2500;

1. d)

2. The result is,

|  |  |
| --- | --- |
| **Region** | **SUM(Orders)** |
| **East** | **3960** |
| **South** | **3485** |
| **West** | **3380** |

3. The result is,

|  |  |
| --- | --- |
| **Region** | **SUM(Orders)** |
| **West** | **3380** |

In this SQL statement, we first apply the **WHERE** clause "Total\_Sales < 385000". That eliminated the two rows where Total\_Sales is larger than 385000. We then sum up the orders from the remaining rows by region, and this results in the following:

|  |  |
| --- | --- |
| **Region** | **SUM(Orders)** |
| **West** | **3380** |
| **North** | **1785** |
| **East** | **1760** |
| **South** | **1695** |