

# **LAB FILE**



**JAMIA MILLIA ISLAMIA**  
**DEPARTMENT OF COMPUTER ENGINEERING**  
**B. TECH (COMPUTER ENGINEERING)**  
**4<sup>th</sup> SEMESTER**

**DataBase Lab (CEN-491)**

**Submitted To:**

- Dr. Faiyaz Ahmed
- Dr. Musheer Ahmad
- Mr. Saif Ali

**Submitted By:**

Divyansh Thapliyal

21BCS002

B.tech (4<sup>th</sup> Semester)

## Index

S. No	Programs	Date	Remarks	Signature
1	ASSIGNMENT-1	06-02-2023		
2	ASSIGNMENT-2	13-02-2023		
3	ASSIGNMENT-3	27-03-2023		
4	ASSIGNMENT-4	27-03-2023		
5	ASSIGNMENT-5	03-04-2023		
6	ASSIGNMENT-6	03-04-2023		
7	ASSIGNMENT-7	17-04-2023		
8	ASSIGNMENT-8	17-04-2023		

# Assignment 1

## Chapter 3

### Exercise 1: Create the tables described below

#### (a) Table Name: Client\_Master

```
mysql> create table client_master(ClienNo varchar(6),Name varchar(20),Address1
varchar(30),Address2 varchar(30),City varchar(15),PinCode int(8),State
varchar(15),BalDue dec(10,2));
```

```
mysql> desc client_master;
```

Field	Type	Null	Key	Default	Extra
ClientNo	varchar(6)	YES		NULL	
Name	varchar(20)	YES		NULL	
Address1	varchar(30)	YES		NULL	
Address2	varchar(30)	YES		NULL	
City	varchar(15)	YES		NULL	
PinCode	int	YES		NULL	
State	varchar(15)	YES		NULL	
BalDue	decimal(10,2)	YES		NULL	

#### (b) Table Name: Product\_Master

```
mysql> create table product_master(ProductNo varchar(15),Description
varchar(15),ProfitPercent dec(4,2),UnitMeasure varchar(10),QtyOnHand
int(8),ReorderLvl int(8),SellPrice dec(8,2),CostPrice dec(8,2));
```

```
mysql> desc product_master;
```

Field	Type	Null	Key	Default	Extra
ProductNo	varchar(15)	YES		NULL	
Description	varchar(15)	YES		NULL	
ProfitPercent	decimal(4,2)	YES		NULL	
UnitMeasure	varchar(10)	YES		NULL	

QtyOnHand	int	YES		NULL		
ReorderLvl	int	YES		NULL		
SellPrice	decimal(8,2)	YES		NULL		
CostPrice	decimal(8,2)	YES		NULL		
+-----+-----+-----+-----+-----+						

### (c) Table Name: Salesman\_Master

```
mysql> create table salesman_master(SalesmanNo varchar(6),SalesmanName
varchar(20),Address1 varchar(30),Address2 varchar(30),City varchar(20),Pincode
int(8),State dec(20),SalAmt dec(8,2),TgtToGet dec(6,2),YtdSales dec(6,2),Remarks
varchar(60));
```

```
mysql> desc salesman_master;
```

+-----+-----+-----+-----+-----+						
Field	Type	Null	Key	Default	Extra	
+-----+-----+-----+-----+-----+						
SalesmanNo	varchar(6)	YES		NULL		
SalesmanName	varchar(20)	YES		NULL		
Address1	varchar(30)	YES		NULL		
Address2	varchar(30)	YES		NULL		
City	varchar(20)	YES		NULL		
Pincode	int	YES		NULL		
State	decimal(20,0)	YES		NULL		
SalAmt	decimal(8,2)	YES		NULL		
TgtToGet	decimal(6,2)	YES		NULL		
YtdSales	decimal(6,2)	YES		NULL		
Remarks	varchar(60)	YES		NULL		
+-----+-----+-----+-----+-----+						

## Exercise 2: Insert the data into the tables

### (a) Data for Client\_Master

```
mysql> insert into client_master values('C00001','Ivan Bayross','Wall Rose','Paradis
Island','Mumbai',400054,'Maharashtra',15000),('C00002','Mamta Muzumdar','Wall
Maria','Paradis Island','Madras',780001,'Tamil Nadu',0),('C00003','Chhaya Bankar','Wall
Rose','Paradis Island','Mumbai',400057,'Maharashtra',5000),('C00004','Ashwini
Joshi','Wall Sina','Paradis Island','Bangalore',560001,'Karnataka',0),('C00005','Hansel
```

```
Colaco','Wall Rose','Paradis
Island','Mumbai',400060,'Maharashtra',20000),('C00006','Deepak Sharma','Wall
Sina','Paradis Island','Mangalore',560050,'Karnataka',0);
```

```
mysql> select * from client_master;
```

ClientNo	Name	Address1	Address2	City	PinCode	State
BalDue						
C00001	Ivan Bayross	Wall Rose	Paradis Island	Mumbai	400054	Maharashtra
15000.00						
C00002	Mamta Muzumdar	Wall Maria	Paradis Island	Madras	780001	Tamil Nadu
0.00						
C00003	Chhaya Bankar	Wall Rose	Paradis Island	Mumbai	400057	Maharashtra
5000.00						
C00004	Ashwini Joshi	Wall Sina	Paradis Island	Bangalore	560001	Karnataka
0.00						
C00005	Hansel Colaco	Wall Rose	Paradis Island	Mumbai	400060	Maharashtra
20000.00						
C00006	Deepak Sharma	Wall Sina	Paradis Island	Mangalore	560050	Karnataka
0.00						

## (b) Data for Product\_Master

```
mysql> insert into product_master
values('P00001','T-Shirts',5,'Piece',200,50,350,250),('P03450','Shirts',6,'Piece',150,50,50
0,350),('P06734','Cotton
Jeans',5,'Piece',100,20,600,450),('P07865','Jeans',5,'Piece',100,20,750,500),('P07868','Tr
ousers',2,'Piece',150,50,850,550),('P07885','Pull
Overs',2.5,'Piece',80,30,700,450),('P07965','Denim
Shirts',4,'Piece',100,40,350,250),('P00001','Lycra
Tops',5,'Piece',70,30,300,175),('P00001','T-Shirts',5,'Piece',200,50,450,300);
```

```
mysql> select * from product_master;
```

ProductNo	Description	ProfitPercent	UnitMeasure	QtyOnHand	ReorderLvl	SellPrice	CostPrice
P00001	T-Shirts		5.00	Piece	200		50
350.00	250.00						

P03450 500.00	Shirts 350.00		6.00   Piece		150	50
P06734 600.00	Cotton Jeans   450.00		5.00   Piece		100	20
P07865 750.00	Jeans 500.00		5.00   Piece		100	20
P07868 850.00	Trousers 550.00		2.00   Piece		150	50
P07885 700.00	Pull Overs 450.00		2.50   Piece		80	30
P07965 350.00	Denim Shirts   250.00		4.00   Piece		100	40
P00001 300.00	Lycra Tops 175.00		5.00   Piece		70	30
P00001 450.00	T-Shirts 300.00		5.00   Piece		200	50

+-----+-----+-----+-----+-----+-----+-----+

### (c) Data for Salesman\_Master

mysql> insert into salesman\_master

values('S00001','Aman','A/14','Worli','Mumbai',400002,'Maharashtra',3000,100,50,'Good'),('S00002','Omkar','65','Nariman','Mumbai',400001,'Maharashtra',3000,200,100,'Good'),('S00003','Raj','P-7','Bandra','Mumbai',400032,'Maharashtra',3000,200,100,'Good'),('S00004','Ashish','A/5','Juhu','Mumbai',400044,'Maharashtra',3500,200,150,'Good');

mysql> select \* from salesman\_master;

+-----+-----+-----+-----+-----+-----+-----+

SalesmanNo	SalesmanName	Address1	Address2	City	Pincode	state	SalAmt	TgtToGet	YtdSales	Remarks
------------	--------------	----------	----------	------	---------	-------	--------	----------	----------	---------

+-----+-----+-----+-----+-----+-----+-----+

S00001 50.00   Good	Aman 	A/14	Worli	Mumbai	400002	Maharashtra	3000.00	100.00	
S00002 100.00   Good	Omkar 	65	Nariman	Mumbai	400001	Maharashtra	3000.00	200.00	
S00003 100.00   Good	Raj 	P-7	Bandra	Mumbai	400032	Maharashtra	3000.00	200.00	
S00004 150.00   Good	Ashish 	A/5	Juhu	Mumbai	400044	Maharashtra	3500.00	200.00	

+-----+-----+-----+-----+-----+-----+-----+

### Exercise 3: Exercise on retrieving records from a table

(a) Find out the names of all the clients.

```
mysql> select name from client_master;
```

```
+-----+
| name          |
+-----+
| Ivan Bayross  |
| Mamta Muzumdar |
| Chhaya Bankar |
| Ashwini Joshi |
| Hansel Colaco |
| Deepak Sharma |
+-----+
```

**(b) Retrieve the entire contents of the Client\_Master table.**

```
mysql> select * from client_master;
```

```
+-----+-----+-----+-----+-----+-----+-----+
| ClientNo | Name          | Address1 | Address2 | City      | PinCode | State |
| BalDue   |
+-----+-----+-----+-----+-----+-----+-----+
| C00001   | Ivan Bayross  | Wall Rose | Paradis Island | Mumbai    | 400054 | Maharashtra |
15000.00 |
| C00002   | Mamta Muzumdar | Wall Maria | Paradis Island | Madras     | 780001 | Tamil Nadu   |
0.00 |
| C00003   | Chhaya Bankar | Wall Rose | Paradis Island | Mumbai     | 400057 | Maharashtra |
5000.00 |
| C00004   | Ashwini Joshi | Wall Sina | Paradis Island | Bangalore | 560001 | Karnataka    |
0.00 |
| C00005   | Hansel Colaco | Wall Rose | Paradis Island | Mumbai     | 400060 | Maharashtra |
20000.00 |
| C00006   | Deepak Sharma | Wall Sina | Paradis Island | Mangalore | 560050 | Karnataka    |
0.00 |
+-----+-----+-----+-----+-----+-----+-----+
```

**(c) Retrieve the list of names, city and the state of all the clients.**

```
mysql> select name,city,state from client_master;
```

```
+-----+-----+-----+
| name           | city       | state       |
+-----+-----+-----+
| Ivan Bayross   | Mumbai     | Maharashtra |
| Mamta Muzumdar | Madras     | Tamil Nadu  |
| Chhaya Bankar  | Mumbai     | Maharashtra |
| Ashwini Joshi  | Bangalore  | Karnataka   |
| Hansel Colaco  | Mumbai     | Maharashtra |
| Deepak Sharma  | Mangalore  | Karnataka   |
+-----+-----+-----+
```

**(d)List the various products available from the Product\_Master table.**

```
mysql> select ProductNo,Description from product_master;
```

```
+-----+-----+
| ProductNo | Description |
+-----+-----+
| P00001    | T-Shirts    |
| P03450    | Shirts      |
| P06734    | Cotton Jeans |
| P07865    | Jeans       |
| P07868    | Trousers    |
| P07885    | Pull Overs  |
| P07965    | Denim Shirts |
| P00001    | Lycra Tops  |
| P00001    | T-Shirts    |
+-----+-----+
```

**(e)List all the clients who are located in Mumbai.**

```
mysql> select ClieNo,Name from client_master where City='Mumbai';
```

```
+-----+-----+
| ClientNo | Name        |
+-----+-----+
```



C00001	Ivan Bayross
--------	--------------

C00003	Chhaya Bankar
--------	---------------

C00005	Hansel Colaco
--------	---------------

```
+-----+-----+
```

**(f) Find the names of salesmen who have a salary equal to Rs.3000.**

```
mysql> select SalesmanName from salesman_master where SalAmt=3000;
```

```
+-----+
```

SalesmanName
--------------

```
+-----+
```

Aman
------

Omkar
-------

Raj
-----

```
+-----+
```

**Exercise 4: Exercise on updating records in a table**

**(a) Change the city of ClientNo 'C00005' to 'Bangalore'.**

```
mysql> update client_master set City='Bangalore' where ClieNo='C00005';
```

Query OK, 1 row affected (0.01 sec)

Rows matched: 1   Changed: 1   Warnings: 0

**(b) Change the BalDue of ClientNo 'C00001' to Rs. 1000.**

```
mysql> update client_master set BalDue='1000' where ClieNo='C00001';
```

Query OK, 1 row affected (0.00 sec)

Rows matched: 1   Changed: 1   Warnings: 0

**(c) Change the cost price of Trousers' to Rs. 950.00.**

```
mysql> update product_master set CostPrice=950 where Description='Trousers';
```

Query OK, 1 row affected (0.01 sec)

Rows matched: 1   Changed: 1   Warnings: 0

**(d) Change the city of the salesman to Pune.**

```
mysql> update salesman_master set City='Pune';
```

Query OK, 4 rows affected (0.00 sec)

Rows matched: 4    Changed: 4    Warnings: 0

**Exercise 5: Exercise on deleting records in a table****(a) Delete all salesmen from the Salesman\_Master whose salaries are equal to Rs. 3500.**

```
mysql> delete from salesman_master
```

```
    -> where SalAmt=3500;
```

Query OK, 1 row affected (0.00 sec)

**(b) Delete all products from Product\_Master where the quantity on hand is equal to 100.**

```
mysql> delete from product_master
```

```
    -> where QtyOnHand=100;
```

Query OK, 3 rows affected (0.00 sec)

**(c) Delete from Client\_Master where the column state holds the value "Tamil Nadu".**

```
mysql> delete from client_master
```

```
    -> where State='Tamil Nadu';
```

Query OK, 1 row affected (0.00 sec)

**Exercise 6: Exercise on Altering the table structure:****(d) Add a column called 'Telephone' of data type 'number' and size ='10' to the Client\_Master table.**

```
mysql> alter table client_master add column Telephone int(10) not null after state;
```

Query OK, 0 rows affected, 1 warning (0.02 sec)

Records: 0    Duplicates: 0    Warnings: 1

**(e) Change the size of SellPrice column in Product\_Master to 10,2.**

```
mysql> alter table product_master
```

```
-> modify SellPrice
```

```
-> dec(10,2);
```

```
Query OK, 9 rows affected (0.03 sec)
```

```
Records: 9  Duplicates: 0  Warnings: 0
```

### **Exercise 7: Exercise on deleting the table structure**

**Destroy along with the data the table Client\_Master along with its data.**

```
mysql> drop table client_master;
```

```
Query OK, 0 rows affected (0.02 sec)
```

### **Exercise 8: Exercise on renaming the table**

**Change the name of the Salesman\_Master table to sman\_mast.**

```
mysql> rename table salesman_master TO sman_mast;
```

```
Query OK, 0 rows affected (0.01 sec)
```

## Assignment 2

### Chapter 5

#### Exercise 1: Computations on table data

**(a) List the names of all clients having 'a' as the second letter in their names.**

```
mysql> select Name
      -> from client_master
      -> where Name like '_a%';
```

```
+-----+
| Name          |
+-----+
| Mamta Muzumdar |
| Hansel Colaco  |
+-----+
```

**(b) List the clients who stay in a city whose First letter is 'M'.**

```
mysql> select Name
      -> from client_master
      -> where City like 'M%';
```

```
+-----+
| Name          |
+-----+
| Ivan Bayross  |
| Mamta Muzumdar |
| Chhaya Bankar |
| Deepak Sharma |
+-----+
```

**(c) List all clients who stay in 'Bangalore' or 'Mangalore'**

```
mysql> select Name
```

-> from client\_master

-> where City='Bangalore' or 'Mangalore';

+-----+

| Name |

+-----+

| Ashwini Joshi |

| Hansel Colaco |

+-----+

**(d) List all clients whose BalDue is greater than value 10000.**

mysql> select Name

-> from client\_master

-> where BalDue>10000;

+-----+

| Name |

+-----+

| Hansel Colaco |

+-----+

**(e) List all information from the Sales\_Order table for orders placed in the month of June.**

mysql> select \*

-> from sales\_order

-> where month(Order\_date)=6;

+-----+-----+-----+-----+-----+-----+-----+-----+

Order_no	Order_date	Dely_addr	Salesman_no	Dely_type	Billed_yn	Dely_date	Order_status
567890	2023-06-24	654 Pine St	234567	B	N	2023-08-29	
234567	2023-06-21	456 Elm St	901234	B	N	2023-09-26	

+-----+-----+-----+-----+-----+-----+-----+-----+

| 567890 | 2023-06-24 | 654 Pine St | 234567 | B | N | 2023-08-29 |

Shipped |

| 234567 | 2023-06-21 | 456 Elm St | 901234 | B | N | 2023-09-26 |

Shipped |

+-----+-----+-----+-----+-----+-----+-----+-----+

**(f) List products whose selling price is greater than 500 and less than or equal to 750.**

```
mysql> select *
```

```
-> from product_master
```

```
-> where SellPrice>500 and SellPrice<=750;
```

```
+-----+-----+-----+-----+-----+-----+-----+
| ProductNo | Description | ProfitPercent | UnitMeasure | QtyOnHand | ReorderLvl | SellPrice | CostPrice |
|
+-----+-----+-----+-----+-----+-----+-----+
| P06734    | Cotton Jeans |              | 5.00 | Piece |      100 |      20 |
600.00 | 450.00 |
| P07865    | Jeans        |              | 5.00 | Piece |      100 |      20 |
750.00 | 500.00 |
| P07885    | Pull Overs   |              | 2.50 | Piece |      80 |      30 |
700.00 | 450.00 |
+-----+-----+-----+-----+-----+-----+-----+
```

**(g) List products whose selling price is more than 500. Calculate a new selling price as, original selling price\* .15. Rename the new column in the output of the above query as new\_price.**

```
mysql> select *, SellPrice * 1.15 AS new_pricerice
```

```
-> from product_master
```

```
-> where SellPrice>500;
```

```
+-----+-----+-----+-----+-----+-----+-----+
| ProductNo | Description | ProfitPercent | UnitMeasure | QtyOnHand | ReorderLvl | SellPrice | CostPrice |
| new_pricerice |
+-----+-----+-----+-----+-----+-----+-----+
| P06734    | Cotton Jeans |              | 5.00 | Piece |      100 |      20 |
600.00 | 450.00 | 690.0000 |
| P07865    | Jeans        |              | 5.00 | Piece |      100 |      20 |
750.00 | 500.00 | 862.5000 |
| P07868    | Trousers     |              | 2.00 | Piece |      150 |      50 |
850.00 | 950.00 | 977.5000 |
| P07885    | Pull Overs   |              | 2.50 | Piece |      80 |      30 |
700.00 | 450.00 | 805.0000 |
+-----+-----+-----+-----+-----+-----+-----+
```

**(h) List the names, city and state of clients who are not in the state of 'Maharashtra'.**

```
mysql> select Name, City, State
```

```
-> from client_master
```

```
-> where State <> 'Maharashtra';
```

```
+-----+-----+-----+
| Name           | City       | State       |
+-----+-----+-----+
| Mamta Muzumdar | Madras     | Tamil Nadu |
| Ashwini Joshi   | Bangalore | Karnataka   |
| Deepak Sharma   | Mangalore | Karnataka   |
+-----+-----+-----+
```

**(i) Count the total number of orders.**

```
mysql> select count(*) as TotalOrders
```

```
-> from Sales_Order;
```

```
+-----+
| TotalOrders |
+-----+
|             | 5 |
+-----+
```

**(j) Calculate the average price of all the products.**

```
mysql> select AVG(SellPrice) as AveragePrice
```

```
-> from product_master;
```

```
+-----+
| AveragePrice |
+-----+
| 538.888889   |
+-----+
```

**(k) Determine the maximum and minimum product prices. Rename the output as max\_price and min\_price respectively.**

```
mysql> select MAX(SellPrice) as max_price, MIN(SellPrice) as min_price
        -> from product_master;
```

```
+-----+-----+
| max_price | min_price |
+-----+-----+
|      850.00 |      300.00 |
+-----+-----+
```

**(I) Count the number of products having price less than or equal to 500.**

```
mysql> select COUNT(*) as ProductCount
        -> from product_master
        -> where SellPrice<=500;
```

```
+-----+
| ProductCount |
+-----+
|              5 |
+-----+
```

**Exercise 2: Data Manipulation**

**(a) List the order number and day on which clients placed their order**

```
mysql> select Order_no, DAY(Order_date) AS Order_Day
        -> from sales_order;
```

```
+-----+-----+
| Order_no | Order_Day |
+-----+-----+
| 456789   |      23   |
| 567890   |      24   |
| 678901   |      25   |
| 234567   |      21   |
| 345678   |      22   |
+-----+-----+
```



**(b) List the month (in alphabets) and date when the orders must be delivered.**

```
mysql> select DATE_FORMAT(Dely_date, '%M') AS Delivery_Month, DAY(Dely_date) AS
Delivery_Day
```

```
-> from sales_order;
```

```
+-----+-----+
| Delivery_Month | Delivery_Day |
+-----+-----+
| May           |           28 |
| August        |           29 |
| May           |           30 |
| September     |           26 |
| December      |           27 |
+-----+-----+
```

**(c) List the OrderDate in the format 'DD-Month-YY. e.g. 12-February-02.**

```
mysql> select DATE_FORMAT(Order_date, '%d-%M-%y') AS Formatted_OrderDate
```

```
-> from sales_order;
```

```
+-----+
| Formatted_OrderDate |
+-----+
| 23-May-23          |
| 24-June-23         |
| 25-April-23        |
| 21-June-23         |
| 22-May-23          |
+-----+
```

**(d) List the date, 15 days after today's date.**

```
mysql> select DATE_ADD(CURDATE(), INTERVAL 15 DAY) AS DateAfter15Days;
```

```
+-----+
| DateAfter15Days |
+-----+
| 2023-06-05      |
```

+-----+

## Assignment 3

### Exercise 1:

a) List the description and total qty sold for each product.

```
mysql> SELECT Product_name AS Description, SUM(Quantity) AS TotalQuantitySold
      -> FROM product_master
      -> GROUP BY Product_name;
```

```
+-----+-----+
| Description | TotalQuantitySold |
+-----+-----+
| T-Shirts   | 5 |
| Shirts     | 6 |
| Cotton Jeans | 5 |
| Jeans      | 5 |
| Trousers    | 2 |
| Pull Overs  | 3 |
| Denim Shirts | 4 |
| Lycra Tops  | 5 |
| Skirts     | 5 |
+-----+-----+
```

b) List the value of each product sold.

```
mysql> SELECT Product_name AS Description, Quantity * Sale_price AS Value
      -> FROM product_master;
```

```
+-----+-----+
| Description | Value |
+-----+-----+
| T-Shirts   | 1250 |
| Shirts     | 2100 |
| Cotton Jeans | 2250 |
| Jeans      | 2500 |
```

Trousers	1100	
Pull Overs	1350	
Denim Shirts	1000	
Lycra Tops	875	
Skirts	1500	

+-----+

**c) List the value of each product sold. c)List the average qty sold for each client that has a maximum order value of 15000.00.**

```
mysql> SELECT client_no, AVG(qty_ordered) AS avg_qty_sold
-> FROM Sales_Order_Details
-> GROUP BY client_no
-> HAVING MAX(product_rate * qty_ordered) <= 15000.00;
```

+-----+
client_no   avg_qty_sold
+-----+
C00006   7.0000
C00004   4.0000
C00003   9.0000
C00001   6.0000
C00005   5.0000
+-----+

**d) List the sum total of all the billed orders for the month of June.**

```
mysql> SELECT COUNT(*) AS Total_Billed_Orders
```

```
-> FROM sales_order
```

```
-> WHERE MONTH(Order_date) = 7 AND Billed_yn = 'Y';
```

```
+-----+
```

```
| Total_Billed_Orders |
```

```
+-----+
```

```
|                1 |
```

```
+-----+
```

**Exercise 2:****a) List the products, which have been sold to 'Ivan Bayross'**

```
mysql> SELECT DISTINCT p.Product_name
-> FROM sales_order so
-> JOIN Sales_Order_Details sod ON so.Client_no = sod.client_no
-> JOIN Client_Master cm ON so.client_no = cm.ClientNo
-> JOIN product_master p ON sod.product_no = p.Product_no
-> WHERE cm.Name = 'Ivan Bayross';
```

```
+-----+
| Product_name |
+-----+
| Jeans        |
+-----+
```

**b) List the products and their quantities that will have to be delivered in the current month**

```
mysql> SELECT p.Product_name, sod.qty_disp
-> FROM Sales_Order_Details sod
-> JOIN sales_order so ON sod.Client_no = so.client_no
-> JOIN product_master p ON sod.product_no = p.Product_no
-> WHERE MONTH(so.Dely_date) = MONTH(CURRENT_DATE())
-> AND YEAR(so.Dely_date) = YEAR(CURRENT_DATE());
```

```
+-----+-----+
| Product_name | qty_disp |
+-----+-----+
| Shirts       | 9        |
| Cotton Jeans | 0        |
| Trousers     | 1        |
+-----+-----+
```

**c) List the ProductNo and description of constantly sold i.e. rapidly moving products.**

```
mysql> SELECT p.Product_no, p.Product_name
-> FROM product_master p
-> JOIN Sales_Order_Details sod ON p.Product_no = sod.product_no
-> GROUP BY p.Product_no, p.Product_name
-> HAVING COUNT(sod.product_no) >= 1;
```

Product_no	Product_name
P0345	Shirts
P07885	Pull Overs
P06734	Cotton Jeans
P00001	T-Shirts
P07865	Jeans
P08865	Skirts
P07965	Denim Shirts
P07868	Trousers
P07975	Lycra Tops

**d) List the names of clients who have purchased Trousers'**

```
mysql> SELECT cm.Name
-> FROM Client_Master cm
-> JOIN Sales_Order so ON cm.Clientno = so.Client_no
-> JOIN Sales_Order_Details sod ON so.Client_no = sod.client_no
-> JOIN product_master pm ON sod.product_no = pm.Product_no
-> WHERE pm.Product_name = 'Trousers';
```

Name
Deepak Sharma

- e) **List the ProductNo and OrderNo of customers having QtyOrdered less than 5 from the Sales\_Order\_Details table for 'Pull Overs'.**

```
mysql> SELECT sod.Product_no, so.Order_no
-> FROM Sales_Order_Details sod
-> JOIN Sales_Order so ON sod.client_no = so.Client_no
-> JOIN product_master pm ON sod.Product_no = pm.Product_no
-> WHERE pm.Product_name = 'Pull Overs' AND sod.qty_ordered < 5;
```

Product_no	Order_no
P07885	ORD00006

- f) **list the products and their quantities for the orders placed by 'Ivan Bayross' and 'Mamta Muzumdar'.**

```
mysql> SELECT sod.Product_no, sod.qty_ordered
-> FROM Sales_Order_Details sod
-> JOIN Sales_Order so ON sod.client_no = so.Client_no
-> JOIN Client_Master cm ON so.Client_no = cm.ClientNo
-> WHERE cm.Name IN ('Ivan Bayross', 'Mamta Muzumdar');
```

Product_no	qty_ordered
------------	-------------



P07865	6
--------	---

- g) List the products and their quantities for the orders placed by ClientNo C00001' and CO0002.

```
mysql> SELECT sod.Product_no, sod.qty_ordered
-> FROM Sales_Order_Details sod
-> JOIN Sales_Order so ON sod.client_no = so.client_no
-> WHERE sod.client_no IN ('C00001', 'C00002');
```

Product_no	qty_ordered
P07865	6

### Exercise 3:

- a) Find the ProductNo and description of non-moving products i.e. products not being sold.

```
mysql> SELECT pm.Product_no, pm.Product_name
-> FROM product_master pm
-> LEFT JOIN Sales_Order_Details sod ON pm.Product_no = sod.Product_no
-> WHERE sod.Product_no IS NULL;
Empty set (0.00 sec)
```

- b) List the customer Name, Address1, Address2, City and PinCode for the client who has placed order no ORD00001.

```
mysql> SELECT cm.Name, cm.Address1, cm.Address2, cm.City, cm.PinCode
-> FROM Sales_Order so
-> JOIN Client_Master cm ON so.Client_no = cm.ClientNo
-> WHERE so.Order_no = 'ORD00001';
```

Name	Address1	Address2	City	PinCode
Hasel Colaco	Mumbai	40060	Maharashtra	234234

**c) List the client names that have placed orders before the month of May'02**

```
mysql> SELECT cm.Name
-> FROM Sales_Order so
-> JOIN Client_Master cm ON so.Client_no = cm.ClientNo
-> WHERE so.Order_date < '2002-05-01';
Empty set (0.00 sec)
```

**d) List if the product 'Lycra Top' has been ordered by any client and print the Client\_no, Name to whom it was sold.**

```
mysql> SELECT cm.ClientNo, cm.Name
-> FROM Sales_Order_Details sod
-> JOIN Sales_Order so ON sod.client_no= so.Client_no
-> JOIN Client_Master cm ON so.Client_no = cm.ClientNo
-> JOIN product_master pm ON sod.Product_no = pm.Product_no
-> WHERE pm.Product_name = 'Lycra Top';
Empty set (0.00 sec)
```

**e) List the names of clients who have placed orders worth Rs. 5000 or more.**

```
mysql> SELECT cm.Name
-> FROM Client_Master cm
-> JOIN Sales_Order so ON cm.ClientNo = so.Client_no
-> JOIN Sales_Order_Details sod ON so.Client_no = sod.Client_no
-> GROUP BY cm.Name
-> HAVING SUM(sod.qty_ordered * sod.product_rate) >= 5000;
```

Name
Deepak Sharma
Ashwini Joshi

+-----+

## **Assignment 4**

### **Exercise 1:**

- a) **create a simple index idx\_prod on product cost price from the Product\_master table**

```
mysql> CREATE INDEX idx_prod ON Product_master (Cost_price);  
Query OK, 0 rows affected (0.03 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
mysql> SHOW INDEX FROM Product_master WHERE Key_name = 'idx_prod';
```

- b) **Create a Unique index ClientNo and ProductNo columns of the Sales\_order\_details table.**

```
CREATE UNIQUE INDEX idx_client_product ON Sales_order_details (client_no,  
Product_No);  
Query OK, 0 rows affected (0.01 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

- c) **create a sequence inv\_seq with the following parameters, increment by 3, cycle, cache 4 and which will generate the numbers from 1 to 9999 in ascending order.**

The CREATE SEQUENCE statement is not supported in MySQL. MySQL does not have a built-in sequence feature like some other database systems.

**d) Create a view on OrderNo, OrderDate, OrderStatus of the sales\_order table and Product\_no, Productrate and qtyOrdered of the Sales\_Order\_Details.**

```
mysql> CREATE VIEW order_details_view AS
-> SELECT so.Order_no, so.Order_date, so.Order_status,
->        sod.Product_no, sod.product_rate, sod.qty_ordered
-> FROM sales_order so
-> JOIN sales_order_details sod ON so.Client_no = sod.client_no;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT *
-> FROM order_details_view;
```

Order_no	Order_date	Order_status	Product_no	product_rate	qty_ordered
ORD00001	2023-02-08	Closed	P07975	288.639	5
ORD00002	2022-08-01	Closed	P0345	456.117	8
ORD00002	2022-08-01	Closed	P07868	555.361	6
ORD00003	2022-08-16	Closed	P07865	541.785	6
ORD00004	2022-07-26	Open	P06734	509.762	9
ORD00006	2023-05-01	Open	P00001	345.316	4
ORD00006	2023-05-01	Open	P07885	640.512	2
ORD00006	2023-05-01	Open	P07965	292.957	5
ORD00006	2023-05-01	Open	P08865	324.589	5

## Assignment 5

WORKER TABLE:

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

BONUS TABLE:

WORKER_REF_ID	BONUS_DATE	BONUS_AMOUNT
1	2016-02-20 00:00:00	5000.00
2	2016-06-11 00:00:00	3000.00
3	2016-02-20 00:00:00	4000.00
1	2016-02-20 00:00:00	4500.00
2	2016-06-11 00:00:00	3500.00

TITLE TABLE:

WORKER_REF_ID	WORKER_TITLE	AFFECTED_FROM
1	Manager	2016-02-20 00:00:00
2	Executive	2016-06-11 00:00:00
8	Executive	2016-06-11 00:00:00
5	Manager	2016-06-11 00:00:00
4	Asst. Manager	2016-06-11 00:00:00
7	Executive	2016-06-11 00:00:00
6	Lead	2016-06-11 00:00:00
3	Lead	2016-06-11 00:00:00

**Q1: Write an SQL query to fetch "FIRST\_NAME" from Worker table using the alias name as <WORKER\_NAME>.**

```
mysql> SELECT FIRST_NAME AS WORKER_NAME FROM Worker;
```

```
+-----+
|WORKER_NAME |
+-----+
| Monika      |
| Niharika    |
| Vishal      |
| Amitabh     |
| Vivek       |
| Vipul       |
| Satish      |
| Geetika     |
+-----+
```

**Q2: Write an SQL query to fetch "FIRST\_NAME" from Worker table in upper case.**

```
mysql> SELECT UPPER(FIRST_NAME) AS FIRST_NAME_UPPERCASE FROM Worker;
```

```
+-----+
| FIRST_NAME |
+-----+
| MONIKA     |
| NIHARIKA   |
| VISHAL     |
| AMITABH    |
| VIVEK      |
| VIPUL      |
| SATISH     |
+-----+
```

```
mysql> SELECT DISTINCT DEPARTMENT AS DIST_DEPT FROM Worker;
```

```
+-----+
| DIST_DEPT |
+-----+
| HR        |
| Admin     |
| Account   |
+-----+
```

```
mysql> SELECT SUBSTRING(FIRST_NAME, 1, 3) AS FIRST_NAME_FIRST_THREE_CHARS FROM Worker;
```

```
+-----+
| FIRST_NAME_FIRST_THREE_CHARS |
+-----+

| Mon           |
| Nih           |
| Vis           |
| Ami           |
| Viv           |
```



```

| Vip |
| Sat |
| Gee |
+-----+

```

**Q5: Write an SQL query to find the position of the alphabet ('a') in the first name column 'Amitabh' from Worker table.**

```
mysql> SELECT POSITION('a' IN FIRST_NAME) AS A_POSITION_IN_FIRST_NAME FROM
Worker WHERE FIRST_NAME = 'Amitabh';
```

```

+-----+
| A_POSITION_IN_FIRST_NAME |
+-----+
| 1 |
+-----+

```

**Q6: Write an SQL query to print the FIRST\_NAME from Worker table after removing white spaces from the right side.**

```
mysql> SELECT RTRIM(FIRST_NAME) AS FIRST_NAME_WITHOUT_RIGHT_WHITESPACES
FROM Worker;
```

```

+-----+
| FIRST_NAME_WITHOUT_RIGHT_WHITESPACES |
+-----+
| Monika |
| Niharika |
| Vishal |
| Amitabh |
| Vivek |

```

Vipul
Satish
Geetika

+-----+

**Q7: Write an SQL query to print the DEPARTMENT from Worker table after removing white spaces from the left side.**

```
mysql> SELECT LTRIM(DEPARTMENT) AS DEPARTMENT_WITHOUT_LEFT_WHITESPACES FROM Worker;
```

DEPARTMENT_WITHOUT_LEFT_WHITESPACES
HR
Admin
HR
Admin
Admin
Account
Account
Admin

+-----+

**Q8: Write an SQL query that fetches the unique values of DEPARTMENT from Worker table and prints its length.**

```
mysql> SELECT DISTINCT DEPARTMENT, LENGTH(DEPARTMENT) AS DEPARTMENT_LENGTH FROM Worker;
```

DEPARTMENT	DEPARTMENT_LENGTH
HR	2

Admin		5	
Account		7	
+-----+			

**Q9: Write an SQL query to print the FIRST\_NAME from Worker table after replacing 'a' with 'A'.**

```
mysql> SELECT REPLACE(FIRST_NAME, 'a', 'A') AS FIRST_NAME_REPLACED_A_WITH_A
FROM Worker;
```

+-----+	
FIRST_NAME_REPLACED_A_WITH_A	
+-----+	
Monika	
NihArika	
VishAl	
AmitAbh	
Vivek	
Vipul	
SATish	
Geetika	
+-----+	

**Q10) 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.**

```
mysql> SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) AS COMPLETE_NAME FROM Worker;
```

+-----+	
COMPLETE_NAME	
+-----+	

```
| Monika Arora      |
| Niharika Verma    |
| Vishal Singhal    |
| Amitabh Singh     |
| Vivek Bhati       |
| Vipul Diwan       |
| Satish Kumar      |
| Geetika Chauhan   |
+-----+
```

**Q11) Write an SQL query to print all Worker details from the Worker table order by FIRST\_NAME Ascending.**

```
mysql> SELECT * FROM Worker ORDER BY FIRST_NAME ASC;
```

```
+-----+-----+-----+-----+-----+-----+
| WORKER_ID | FIRST_NAME | LAST_NAME | SALARY | JOINING_DATE | DEPARTMENT |
+-----+-----+-----+-----+-----+-----+
|      4 | Amitabh   | Singh    | 500000 | 2014-02-20 09:00:00 | Admin      |
|      8 | Geetika   | Chauhan  | 90000  | 2014-04-11 09:00:00 | Admin      |
|      1 | Monika    | Arora     | 100000 | 2014-02-20 09:00:00 | HR         |
|      2 | Niharika  | Verma     | 80000  | 2014-06-11 09:00:00 | Admin      |
|      7 | Satish    | Kumar     | 75000  | 2014-01-20 09:00:00 | Account    |
|      6 | Vipul     | Diwan     | 200000 | 2014-06-11 09:00:00 | Account    |
|      3 | Vishal    | Singhal   | 300000 | 2014-02-20 09:00:00 | HR         |
|      5 | Vivek     | Bhati     | 500000 | 2014-06-11 09:00:00 | Admin      |
+-----+-----+-----+-----+-----+-----+
```

**Q12) Write an SQL query to print all Worker details from the Worker table order by FIRST\_NAME Ascending and DEPARTMENT Descending.**

```
mysql> SELECT * FROM Worker ORDER BY FIRST_NAME ASC, DEPARTMENT DESC;
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin

**Q13) Write an SQL query to print details for Workers with the first name as “Vipul” and “Satish” from Worker table.**

```
mysql> SELECT * FROM Worker WHERE FIRST_NAME IN ('Vipul', 'Satish');
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account

```
mysql> SELECT * FROM Worker WHERE FIRST_NAME NOT IN ('Vipul', 'Satish');
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

```
mysql> SELECT * FROM Worker WHERE DEPARTMENT = 'Admin';
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

**Q16) Write an SQL query to print details of the Workers whose FIRST\_NAME contains 'a'.**

```
mysql> SELECT * FROM Worker WHERE FIRST_NAME LIKE '%a%';
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

**Q17) Write an SQL query to print details of the Workers whose FIRST\_NAME ends with 'a'.**

```
mysql> SELECT * FROM Worker WHERE FIRST_NAME LIKE '%a';
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
-----------	------------	-----------	--------	--------------	------------

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

**Q18) Write an SQL query to print details of the Workers whose FIRST\_NAME ends with 'h' and contains six alphabets.**

```
mysql> SELECT * FROM Worker WHERE FIRST_NAME LIKE '____h';
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account

**Q19) Write an SQL query to print details of the Workers whose SALARY lies between 100000 and 500000.**

```
mysql> SELECT * FROM Worker WHERE SALARY BETWEEN 100000 AND 500000;
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR



	4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin	
	5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin	
	6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account	
+-----+-----+-----+-----+-----+-----+-----+							

**Q20) Write an SQL query to print details of the Workers who have joined in Feb'2014.**

```
mysql> SELECT * FROM Worker WHERE JOINING_DATE BETWEEN '2014-02-01' AND '2014-02-28';
```

+-----+-----+-----+-----+-----+-----+-----+						
WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT	
+-----+-----+-----+-----+-----+-----+-----+						
1	Monika	Arora	100000	2014-02-20 09:00:00	HR	
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR	
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin	
+-----+-----+-----+-----+-----+-----+-----+						

**Q21) Write an SQL query to fetch the count of employees working in the department 'Admin'.**

```
mysql> SELECT COUNT(*) AS EmployeeCount FROM Worker WHERE DEPARTMENT = 'Admin';
```

```
+-----+
| EmployeeCount |
+-----+
|           4   |
+-----+
```

**Q22) Write an SQL query to fetch worker names with salaries >= 50000 and <= 100000.**

```
mysql> SELECT FIRST_NAME FROM Worker WHERE SALARY BETWEEN 50000 AND 100000;
```

```
+-----+
| FIRST_NAME |
+-----+
| Monika     |
| Niharika   |
| Satish     |
| Geetika    |
+-----+
```

**Q23) Write an SQL query to fetch the no. of workers for each department in the descending order.**

```
mysql> SELECT DEPARTMENT, COUNT(*) AS EmployeeCount FROM Worker GROUP BY
DEPARTMENT ORDER BY EmployeeCount DESC;
```

```
+-----+-----+
```

DEPARTMENT	EmployeeCount
Admin	4
HR	2
Account	2

**Q24) Write an SQL query to print details of the Workers who are also Managers.**

```
mysql> SELECT w.*
      -> FROM Worker w
      -> JOIN title wt ON w.WORKER_ID = wt.WORKER_REF_ID
      -> WHERE wt.WORKER_TITLE = 'Manager';
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin

**Q25) Write an SQL query to fetch duplicate records having matching data in some fields of a table.**

```
mysql> SELECT FIRST_NAME, LAST_NAME, COUNT(*) AS DuplicateCount
-> FROM Worker
-> GROUP BY FIRST_NAME, LAST_NAME
-> HAVING COUNT(*) > 1;
Empty set (0.01 sec)
```

**Q.26) Write an SQL query to show only odd rows from a table.**

```
mysql> SELECT * FROM WORKER WHERE WORKER_ID % 2 = 1;
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account

**Q.27) Write an SQL query to show only even rows from a table**

```
mysql> SELECT * FROM WORKER WHERE WORKER_ID % 2 = 0;
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account

8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin
---	---------	---------	-------	---------------------	-------

### Q.28) Write an SQL query to clone a new table from another table

```
mysql> CREATE TABLE NEW_WORKER AS SELECT * FROM WORKER;
```

Query OK, 8 rows affected (0.02 sec)

Records: 8 Duplicates: 0 Warnings: 0

### Q-29: Write an SQL query to fetch intersecting records of two tables.

```
mysql> SELECT T.* FROM TITLE T INNER JOIN BONUS B ON T.WORKER_REF_ID = B.WORKER_REF_ID;
```

WORKER_REF_ID	WORKER_TITLE	AFFECTED_FROM
1	Manager	2016-02-20 00:00:00
1	Manager	2016-02-20 00:00:00
2	Executive	2016-06-11 00:00:00
2	Executive	2016-06-11 00:00:00
3	Lead	2016-06-11 00:00:00

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

```
mysql> SELECT T.* FROM TITLE T LEFT JOIN BONUS B ON T.WORKER_REF_ID = B.WORKER_REF_ID WHERE B.WORKER_REF_ID IS NULL;
```

WORKER_REF_ID	WORKER_TITLE	AFFECTED_FROM
8	Executive	2016-06-11 00:00:00
5	Manager	2016-06-11 00:00:00

4	Asst. Manager	2016-06-11 00:00:00
7	Executive	2016-06-11 00:00:00
6	Lead	2016-06-11 00:00:00

**Q-31: Write an SQL query to show the current date and time.**

```
mysql> SELECT CURRENT_TIMESTAMP AS Current_Date_Time;
```

Current_Date_Time
2023-05-19 03:23:19

**Q-32: Write an SQL query to show the top n (say 10) records of a table.**

```
mysql> SELECT * FROM WORKER LIMIT 10;
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

**Q-33: Write an SQL query to determine the nth (say n=5) highest salary from a table.**

```
mysql> SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY DESC LIMIT 1 OFFSET 4;
```

SALARY
--------

```
| 90000 |
+-----+
```

**Q-34: Write an SQL query to determine the 5th highest salary without using TOP or limit method.**

```
mysql> SELECT DISTINCT SALARY FROM WORKER w1
-> WHERE 5 = (
->     SELECT COUNT(DISTINCT SALARY)
->     FROM WORKER w2
->     WHERE w2.SALARY >= w1.SALARY
-> );
```

```
+-----+
| SALARY |
+-----+
| 90000  |
+-----+
```

**Q-35: Write an SQL query to fetch the list of employees with the same salary.**

```
mysql> SELECT SALARY, GROUP_CONCAT(FIRST_NAME) AS EMPLOYEES FROM WORKER GROUP
BY SALARY HAVING COUNT(*) > 1;
```

```
+-----+-----+
| SALARY | EMPLOYEES |
+-----+-----+
| 500000 | Amitabh,Vivek |
+-----+-----+
```

**Q-36: Write an SQL query to show the second highest salary from a table.**

```
mysql> SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY DESC LIMIT 1 OFFSET
1;
```

```
+-----+
| SALARY |
+-----+
| 300000 |
+-----+
```

**Q-37: Write an SQL query to show one row twice in results from a table.**

```
mysql> SELECT * FROM WORKER UNION ALL SELECT * FROM WORKER;
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

**Q-39: Write an SQL query to fetch the first 50% records from a table.**

```
mysql> SELECT * FROM WORKER WHERE WORKER_ID <= (SELECT count(WORKER_ID)/2 from Worker);
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR



4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
---	---------	-------	--------	---------------------	-------

**Q-40: Write an SQL query to fetch the departments that have less than five people in it.**

```
mysql> SELECT DEPARTMENT FROM WORKER GROUP BY DEPARTMENT HAVING COUNT(*) < 5;
```

DEPARTMENT
HR
Admin
Account

**Q-41: Write an SQL query to show all departments along with the number of people in there.**

```
mysql> SELECT DEPARTMENT, COUNT(*) AS No_Of_People FROM Worker GROUP BY DEPARTMENT;
```

DEPARTMENT	No_Of_People
HR	2
Admin	4
Account	2

**Q-42: Write an SQL query to show the last record from a table.**

```
mysql> SELECT * FROM WORKER ORDER BY WORKER_ID DESC LIMIT 1;
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

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

```
mysql> SELECT * FROM WORKER LIMIT 1;
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR

**Q-44: Write an SQL query to fetch the last five records from a table.**

```
mysql> (SELECT * FROM WORKER ORDER BY WORKER_ID DESC LIMIT 5) ORDER BY WORKER_ID ASC;
```

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

**Q-45: Write an SQL query to print the name of employees having the highest salary in each department.**

```
mysql> SELECT FIRST_NAME, LAST_NAME, SALARY, DEPARTMENT FROM WORKER
-> WHERE (DEPARTMENT, SALARY) IN
-> (SELECT DEPARTMENT, MAX(SALARY) FROM WORKER GROUP BY DEPARTMENT);
```

FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT
Vishal	Singhal	300000	HR
Amitabh	Singh	500000	Admin
Vivek	Bhati	500000	Admin
Vipul	Diwan	200000	Account

```
+-----+-----+-----+-----+
```

**Q-46: Write an SQL query to fetch three max salaries from a table.**

```
mysql> SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY DESC LIMIT 3;
```

```
+-----+
| SALARY |
+-----+
| 500000 |
| 300000 |
| 200000 |
+-----+
```

**Q-47: Write an SQL query to fetch three min salaries from a table.**

```
mysql> SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY LIMIT 3;
```

```
+-----+
| SALARY |
+-----+
| 75000  |
| 80000  |
| 90000  |
+-----+
```

**Q-48: Write an SQL query to fetch nth max salaries from a table.**

(assuming n = 3)

```
mysql> SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY DESC LIMIT 1 OFFSET
2;
```

```
+-----+
| SALARY |
+-----+
| 200000 |
+-----+
```

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

```
mysql> SELECT DEPARTMENT, SUM(SALARY) AS TOTAL_SALARY FROM WORKER GROUP BY DEPARTMENT;
```

DEPARTMENT	TOTAL_SALARY
HR	400000
Admin	1170000
Account	275000

**Q-50: Write an SQL query to fetch the names of workers who earn the highest salary**

```
mysql> SELECT FIRST_NAME, LAST_NAME FROM WORKER WHERE SALARY = (SELECT MAX(SALARY) FROM WORKER);
```

FIRST_NAME	LAST_NAME
Amitabh	Singh
Vivek	Bhati

## Assignment 6

### 1. Select all records from the EMP table

```
mysql>SELECT * FROM EMP;
```

EMP_NO	EMP_NAME	JOB	MGR	HIRE_DATE	SAL	COMM	DEPT_NO
7369	SMITH	CLERK	7902	1980-12-17	800	NULL	20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
7566	JONES	MANAGER	7839	1981-04-02	2975	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850	NULL	30
7782	CLARK	MANAGER	7839	1981-06-09	2450	NULL	10
7788	SCOTT	ANAYLST	7566	1987-04-19	3000	NULL	20
7839	KING	PRESIDENT	NULL	1981-11-17	5000	NULL	10
7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30
7876	ADAMS	CLERK	7788	1987-05-23	1100	NULL	20
7900	JAMES	CLERK	7698	1981-12-03	950	NULL	30
7902	FORD	ANALYST	7566	1987-12-03	3000	NULL	20
7934	MILLER	CLERK	7782	1982-01-23	1300	NULL	10

14 rows in set (0.00 sec)

### 2. Select all records from the DEPT table

```
mysql>SELECT * FROM DEPT;
```

DEPT_NO	DEPT_NAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO

```
|      40 | OPERATIONS | BOSTON |  
+-----+-----+-----+  
4 rows in set (0.00 sec)
```

### 3. Find the employee name and salary of employees working in DEPT NO 2

```
mysql>SELECT EMP_NAME, SAL FROM EMP WHERE DEPT_NO=20;
```

```
+-----+-----+  
| EMP_NAME | SAL |  
+-----+-----+  
| SMITH    | 800 |  
| JONES    | 2975 |  
| SCOTT    | 3000 |  
| ADAMS    | 1100 |  
| FORD     | 3000 |  
+-----+-----+  
5 rows in set (0.00 sec)
```

### 4. Find the name, job, salary of the employee who is a manager

```
mysql>SELECT EMP_NAME, JOB, SAL FROM EMP WHERE JOB='MANAGER';
```

```
+-----+-----+-----+  
| EMP_NAME | JOB      | SAL |  
+-----+-----+-----+  
| JONES    | MANAGER | 2975 |  
| BLAKE    | MANAGER | 2850 |  
| CLARK    | MANAGER | 2450 |  
+-----+-----+-----+  
3 rows in set (0.00 sec)
```

## 5. Find the name, job, salary of the employee who is not a manager

```
mysql>SELECT EMP_NAME, JOB, SAL FROM EMP WHERE JOB!='MANAGER';
```

```
+-----+-----+-----+
| EMP_NAME | JOB       | SAL  |
+-----+-----+-----+
| SMITH    | CLERK     | 800  |
| ALLEN    | SALESMAN  | 1600 |
| WARD     | SALESMAN  | 1250 |
| MARTIN   | SALESMAN  | 1250 |
| SCOTT    | ANALYST   | 3000 |
| KING     | PRESIDENT | 5000 |
| TURNER   | SALESMAN  | 1500 |
| ADAMS    | CLERK     | 1100 |
| JAMES    | CLERK     | 950  |
| FORD     | ANALYST   | 3000 |
| MILLER   | CLERK     | 1300 |
+-----+-----+-----+
```

```
11 rows in set (0.00 sec)
```

## 6. Find those employees who were hired between 1 MAR 1981 and 1 JUNE 1983

```
mysql>SELECT EMP_NAME FROM EMP WHERE HIRE_DATE BETWEEN '1981-03-01' AND '1983-06-01';
```

```
+-----+
| EMP_NAME |
+-----+
| JONES    |
| MARTIN   |
| BLAKE    |
| CLARK    |
| KING     |
| TURNER   |
| JAMES    |
| MILLER   |
+-----+
```

8 rows in set (0.00 sec)

### 7. Find employee names who were hired in 1981

```
mysql>SELECT EMP_NAME FROM EMP WHERE HIRE_DATE BETWEEN '1981-01-01' AND '1981-12-31';
```

```
+-----+
```

```
| EMP_NAME |
```

```
+-----+
```

```
| ALLEN    |
```

```
| WARD     |
```

```
| JONES    |
```

```
| MARTIN   |
```

```
| BLAKE    |
```

```
| CLARK    |
```

```
| KING     |
```

```
| TURNER   |
```

```
| JAMES    |
```

```
+-----+
```

9 rows in set (0.00 sec)

### 8. Find employee names whose names start with 'S'

```
mysql>SELECT * FROM EMP WHERE EMP_NAME LIKE 'S%';
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
| EMP_NO | EMP_NAME | JOB      | MGR  | HIRE_DATE | SAL  | COMM | DEPT_NO |
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
| 7369 | SMITH    | CLERK    | 7902 | 1980-12-17 | 800  | NULL | 20      |
```

```
| 7788 | SCOTT    | ANALYST  | 7566 | 1987-04-19 | 3000 | NULL | 20      |
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
```

2 rows in set (0.00 sec)



### 9. Find employee names whose names end with 'S'

```
mysql>SELECT EMP_NAME FROM EMP WHERE EMP_NAME LIKE '%S';
```

```
+-----+
```

```
| EMP_NAME |
```

```
+-----+
```

```
| JONES    |
```

```
| ADAMS    |
```

```
| JAMES    |
```

```
+-----+
```

```
3 rows in set (0.00 sec)
```

### 10. Find employee names working in DEPT\_NO 20 and 40

```
mysql>SELECT EMP_NAME FROM EMP WHERE DEPT_NO=20 OR DEPT_NO=40;
```

```
+-----+
```

```
| EMP_NAME |
```

```
+-----+
```

```
| SMITH    |
```

```
| JONES    |
```

```
| SCOTT    |
```

```
| ADAMS    |
```

```
| FORD     |
```

```
+-----+
```

```
5 rows in set (0.00 sec)
```

### 11. Find EMP\_NAME, JOB and DEPT\_NO who are CLERK and SALESMAN

```
mysql>SELECT EMP_NAME, JOB, DEPT_NO FROM EMP WHERE JOB='CLERK' OR JOB='SALESMAN';
```

```
+-----+-----+-----+
| EMP_NAME | JOB      | DEPT_NO |
+-----+-----+-----+
| SMITH    | CLERK    | 20      |
| ALLEN    | SALESMAN | 30      |
| WARD     | SALESMAN | 30      |
| MARTIN   | SALESMAN | 30      |
| TURNER   | SALESMAN | 30      |
| ADAMS    | CLERK    | 20      |
| JAMES    | CLERK    | 30      |
| MILLER   | CLERK    | 10      |
+-----+-----+-----+
```

```
8 rows in set (0.00 sec)
```

## 12. Find EMP\_NAME who are manage and getting salary above 2000

```
mysql>SELECT EMP_NAME FROM EMP WHERE SAL>2000;
```

```
+-----+
| EMP_NAME |
+-----+
| JONES    |
| BLAKE    |
| CLARK    |
| SCOTT    |
| KING     |
| FORD     |
+-----+
```

```
6 rows in set (0.00 sec)
```

## 13. Find EMP\_NAME who are working in DEPT\_NO 30 order by salary in DESC order

```
mysql>SELECT EMP_NAME FROM EMP WHERE DEPT_NO=30 ORDER BY SAL DESC;
```

```
+-----+
```

```
| EMP_NAME |  
+-----+  
| BLAKE   |  
| ALLEN   |  
| TURNER  |  
| WARD    |  
| MARTIN  |  
| JAMES   |  
+-----+
```

6 rows in set (0.00 sec)

#### 14. Find out the total salary of all the employees

```
mysql>SELECT SUM(SAL) FROM EMP;
```

```
+-----+  
| SUM(SAL) |  
+-----+  
|    29025 |  
+-----+
```

1 row in set (0.00 sec)

#### 15. Find out the average salary of all the employees who are working in DEPT\_NO 30

```
mysql>SELECT AVG(SAL) FROM EMP WHERE DEPT_NO=30;
```

```
+-----+  
| AVG(SAL) |  
+-----+  
| 1566.6667 |  
+-----+
```

1 row in set (0.00 sec)

#### 16. Find out the minimal salary of DEPT\_NO 30

```
mysql>SELECT MIN(SAL) FROM EMP WHERE DEPT_NO=30;
```

```
+-----+  
| MIN(SAL) |
```

```
+-----+
|      950 |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

### 17. Find out the maximum hired date

```
mysql>SELECT MAX(HIRE_DATE) FROM EMP;
```

```
+-----+
| MAX(HIRE_DATE) |
```

```
+-----+
```

```
| 1987-12-03      |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

### 18. Find out the total number of employees who are working in DEPT\_NO 12

```
mysql>SELECT COUNT(EMP_NAME) FROM EMP WHERE DEPT_NO=12;
```

```
+-----+
| COUNT(EMP_NAME) |
```

```
+-----+
```

```
|                0 |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

### 19. Find out the DET\_NO, Total salary of those departments where there is no salesman and total salary of the department is more than 8500

```
mysql> SELECT DEPT_NO, SUM(SAL) AS TotalSalary
```

```
-> FROM EMP
```

```
-> WHERE DEPT_NO NOT IN (
```

```

-> SELECT DEPT_NO
-> FROM EMP
-> WHERE JOB = 'SALESMAN'
-> )
-> GROUP BY DEPT_NO
-> HAVING TotalSalary > 8500;

```

```

+-----+-----+
| DEPT_NO | TotalSalary |
+-----+-----+
|      20 |      10875 |
|      10 |      8750 |
+-----+-----+
2 rows in set (0.01 sec)

```

## 20. Find EMP\_NAME who were hired first

```
mysql> SELECT *
```

```

-> FROM EMP
-> ORDER BY HIRE_DATE ASC
-> LIMIT 1;

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| EMP_NO | EMP_NAME | JOB   | MGR   | HIRE_DATE | SAL  | COMM | DEPT_NO |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 7369   | SMITH    | CLERK | 7902  | 1980-12-17 | 800  | NULL | 20      |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

## 21. Find total salary of those who are not manager

```
mysql> SELECT SUM(SAL) FROM EMP WHERE JOB!='MANAGER';
```

```

+-----+
| SUM(SAL) |
+-----+
| 20750    |

```

```
+-----+
```

```
1 row in set (0.00 sec)
```

22. Find out jobs and average salary for all the job types with more than 2 employees

```
mysql> SELECT JOB, AVG(SAL) AS AverageSalary
```

```
-> FROM EMP
```

```
-> GROUP BY JOB
```

```
-> HAVING COUNT(*) > 2;
```

```
+-----+-----+
```

```
| JOB      | AverageSalary |
```

```
+-----+-----+
```

```
| CLERK    | 1037.5000 |
```

```
| SALESMAN | 1400.0000 |
```

```
| MANAGER  | 2758.3333 |
```

```
+-----+-----+
```

```
3 rows in set (0.01 sec)
```

23. Find out the EMP\_NAME having maximum salary in each department

```
mysql> SELECT EMP_NAME, SAL, DEPT_NO
```

```
-> FROM EMP t1
```

```
-> WHERE SAL = (
```

```
->     SELECT MAX(SAL)
```

```
->     FROM EMP t2
```

```
->     WHERE t1.DEPT_NO = t2.DEPT_NO
```

```
-> )
```

```
-> ;
```

```
+-----+-----+-----+
```

```
| EMP_NAME | SAL  | DEPT_NO |
```

```
+-----+-----+-----+
```

```
| BLAKE    | 2850 | 30 |
```

```
| SCOTT    | 3000 | 20 |
```

```
| KING     | 5000 | 10 |
```

```
| FORD     | 3000 | 20 |
```

```
+-----+-----+-----+
```

4 rows in set (0.00 sec)

## 24. Find out the square root of the salary in EMP table

```
mysql>SELECT SQRT(SAL) FROM EMP;
```

```
+-----+
| SQRT(SAL) |
+-----+
| 28.284271247461902 |
|                40 |
| 35.35533905932738 |
| 54.543560573178574 |
| 35.35533905932738 |
| 53.38539126015655 |
| 49.49747468305833 |
| 54.772255750516614 |
| 70.71067811865476 |
| 38.72983346207417 |
| 33.166247903554 |
| 30.822070014844883 |
| 54.772255750516614 |
| 36.05551275463989 |
+-----+
```

14 rows in set (0.00 sec)

## 25. Find average salary of employees whose JOB = 'CLERK'

```
mysql>SELECT AVG(SAL) FROM EMP WHERE JOB='CLERK';
```

```
+-----+
| AVG(SAL) |
+-----+
| 1037.5000 |
+-----+
```

1 row in set (0.00 sec)

## 26. Find total salary of those employees who were hired in 1981

```
mysql>SELECT SUM(SAL) FROM EMP WHERE HIRE_DATE BETWEEN '1981-01-01' AND '1981-12-31';
```

```
+-----+
| SUM(SAL) |
+-----+
|    19825 |
+-----+
```

1 row in set (0.00 sec)

## 27. Change the JOB, DEPT\_NO, SALARY of employee with EMP\_NO = 7788

```
mysql> UPDATE EMP SET JOB=NULL, DEPT_NO=NULL, SAL=NULL WHERE EMP_NO=7788;
```

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

## 28. Create a NEW table using all records of EMP

```
mysql> create table NEW
```

```
-> (EMP_NO int PRIMARY KEY, EMP_NAME varchar(20), JOB varchar(10), MGR int,
HIRE_DATE date, SAL int, COMM int, DEPT_NO int);
```

Query OK, 0 rows affected (0.04 sec)

## 29. Change the JOB of the table NEW to 'SALES'

```
mysql> ALTER TABLE NEW
```

```
-> RENAME COLUMN JOB TO SALES;
```

Query OK, 0 rows affected (0.02 sec)

Records: 0 Duplicates: 0 Warnings: 0

## 30. Select all records from NEW

```
mysql> SELECT * FROM NEW;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| EMP_NO | EMP_NAME | SALES | MGR | HIRE_DATE | SAL | COMM | DEPT_NO |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 7369 | SMITH | CLERK | 7902 | 1980-12-17 | 800 | NULL | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1600 | 300 | 30 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

2 rows in set (0.00 sec)

## 31. Add a new column ADDRESS VARCHAR (10) to table NEW

```
mysql> ALTER TABLE NEW
```

```
-> ADD COLUMN ADDRESS VARCHAR (10);
```



Query OK, 0 rows affected (0.01 sec)

Records: 0 Duplicates: 0 Warnings: 0

### 32. Insert the values to the column address of table NEW

mysql> UPDATE NEW

-> SET ADDRESS='NEW YORK' WHERE EMP\_NO=7369;

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

### 33. Select all records from table NEW

mysql> SELECT \* FROM NEW;

```
+-----+-----+-----+-----+-----+-----+-----+-----+
----+
| EMP_NO | EMP_NAME | SALES   | MGR   | HIRE_DATE | SAL   | COMM | DEPT_NO | ADDRESS
|
+-----+-----+-----+-----+-----+-----+-----+-----+
----+
| 7369 | SMITH   | CLERK   | 7902 | 1980-12-17 | 800   | NULL | 20 | NEW YORK
|
| 7499 | ALLEN   | SALESMAN | 7698 | 1981-02-20 | 1600  | 300  | 30 | NULL
|
+-----+-----+-----+-----+-----+-----+-----+-----+
----+
```

2 rows in set (0.00 sec)

### 34. Update the size of the ADDRESS column from 10 to 4

mysql> ALTER TABLE NEW

-> MODIFY ADDRESS VARCHAR(40);

Query OK, 0 rows affected (0.01 sec)

Records: 0 Duplicates: 0 Warnings: 0

### 35. Delete table NEW

mysql> DROP TABLE NEW;

Query OK, 0 rows affected (0.01 sec)

## Assignment 7

TABLES CREATED:

(Q1) TABLE Passenger

pid	pname	age
0	Sachin	65
1	Rahul	66
2	Sourav	67
3	Anil	69

(Q1) Table reservation

pid	class	tid
0	AC	8200
1	AC	8201
2	SC	8201
5	AC	8203
1	SC	8204
3	AC	8202

(Q3) Table Employee

name	sex	salary	deptName
Utahime Iori	F	40000	Maths
Gojo Saturo	M	40000	Psychology
Geto Suguru	M	40000	English

Shoko Ieri	F	50000	Biology	
Kento Nanami	M	60000	Physics	

+-----+-----+-----+-----+

(Q4) Table Book

+-----+-----+

title	price
-------	-------

+-----+-----+

Shadow and Bone	250
-----------------	-----

Six Of Crows	350
--------------	-----

King Of Scars	450
---------------	-----

Cruel Prince	100
--------------	-----

Wicked King	200
-------------	-----

Queen Of Nothing	300
------------------	-----

+-----+-----+

(Q5) Table enrolled

+-----+-----+

student	course
---------	--------

+-----+-----+

abc	c1
-----	----

xyz	c1
-----	----

abc	c2
-----	----

pqr	c1
-----	----

+-----+-----+

(Q5) Table paid

+-----+-----+

student	amount
---------	--------

+-----+-----+

abc	20000
-----	-------

xyz	10000
-----	-------

rst	10000
-----	-------

+-----+-----+

(Q6) Table account

+-----+-----+

customer	balance
----------	---------

+-----+-----+		
Kento Nanami	70000	
Shoko Ieri	70000	
Gojo Saturo	60000	
Utahime Iori	60000	
Geto Suguru	30000	
+-----+-----+		

(Q7) TABLE Loan\_Records

+-----+-----+-----+			
Borrower	Bank_Manager	Loan_Amount	
+-----+-----+-----+			
Ramesh	Sunderajan	10000	
Suresh	Ramgopa	15000	
Mahesh	Sunderajan	7000	
+-----+-----+-----+			

Table employee (Q8)

+-----+-----+-----+				
empId	name	department	salary	
+-----+-----+-----+				
e1	A	1	10000	
e2	B	5	20000	
e3	C	3	20000	
e4	D	5	30000	
e5	E	2	9000	
+-----+-----+-----+				

1. What pids are returned by the following SQL query for the above instance of the tables?

**SELECT pid FROM Reservation WHERE class 'AC' AND EXISTS (SELECT \* FROM Passenger WHERE age > 65 AND Passenger. pid = Reservation.pid)**

(a) 1, 0 (b) 1, 2 (c) 1, 5 (d) 1, 3

**ANS.** (d) 1, 3

```
mysql> SELECT pid FROM Reservation WHERE class='AC' AND EXISTS (SELECT* FROM Passenger
WHERE age > 65 AND Passenger.pid = Reservation.pid);
```

```
+-----+
```

```
| pid |
```

```
+-----+
```

```
| 1 |
```

```
| 3 |
```

```
+-----+
```

```
2 rows in set (0.00 sec)
```

2. Consider the following relational schema:

```
Suppliers(sid:integer, sname:string, city:string,
street:string) Parts(pid:integer, pname:string, color:string)
Catalog(sid:integer, pid:integer, cost:real)
```

Consider the following relational query on the above database:

```
SELECT S.sname
FROM Suppliers S
WHERE S.sid NOT IN (SELECT C.sid FROM Catalog C
WHERE C.pid NOT IN (SELECT P.pid FROM Parts P
WHERE P.color<> 'blue'))
```

Assume that relations corresponding to the above schema are not empty. Which one of the following is

the correct interpretation of the above query?

- (a) Find the names of all suppliers who have not supplied a non-blue part.
- (b) Find the names of all suppliers who have supplied only blue parts.
- (c) Find the names of all suppliers who have not supplied only blue parts.
- (d) Find the names of all suppliers who have supplied a non-blue part.

**ANS.** (D) option matched because given query returns suppliers who have not supplied any blue parts. That means it can include other than blue parts.

3. The employee information in a company is stored in the relation

```
Employee (name, sex, salary, deptName)
```

Consider the following SQL query

```
Select deptName From Employee Where sex = 'M' Group by deptName Having avg(salary) >
(select avg (salary) from Employee)
```

It returns the names of the department in which

- (a) the average salary of male employees is more than the average salary in the company
- (b) the average salary is more than the average salary in the company
- (c) the average salary of male employees is more than the average salary of all male employees in the company
- (d) the average salary of male employees is more than the average salary of employees in the same department

**ANS.** (D) the average salary of male employees is more than the average salary of employees in the same department

```
+-----+
```

```
| deptName |
```

```
+-----+
```

```
| Physics  |
```

```
+-----+
```

This query would return the name of all departments in which the average salary of male employees is greater than the average salary of all employees in the company.

**4.** The relation book (title, price) contains the titles and prices of different books. Assuming that no two

books have the same price, what does the following SQL query list?

```
Select title From book as B Where (Select count(*) from book as T Where T.price > B.price)
< 5
```

- (a) Titles of the four most expensive books (b) Title of the fifth most inexpensive book
- (c) Title of the fifth most expensive book
- (d) Titles of the five most expensive books

**ANS.** (d) Titles of the five most expensive books

```
+-----+
```

```
| title          |
```

```
+-----+
```

```
| Shadow and Bone |
```

```
| Six Of Crows    |
```

```
| King Of Scars   |
```

```
| Wicked King      |
| Queen Of Nothing |
+-----+
5 rows in set (0.01 sec)
```

5. Consider the relation enrolled (student, course) in which (student, course) is the primary key, and the relation paid (student, amount) where student is the primary key. Assume no null values and no foreign keys or integrity constraints. Given the following four queries:

Query1: select student from enrolled where student in (select student from paid)

Query2: select student from paid where student in (select student from enrolled)

Query3: select E.student from enrolled E, paid P where E.student = P.student

Query4: select student from paid where exists

(select \* from enrolled where enrolled.student = paid.student)

Which one of the following statements is correct?

(a) All queries return identical row sets for any database

(b) Query2 and Query4 return identical row sets for all databases but there exist databases for which

Query1 and Query2 return different row sets.

(c) There exist databases for which Query3 returns strictly fewer rows than Query2.

(d) There exist databases for which Query4 will encounter an integrity violation at runtime

Output of Query 1

abc

abc

xyz

Output of Query 2

abc

xyz

Output of Query 3

abc

xyz

Output of Query 4

abc

xyz

Query 1 and Query 3 may return repetitive student values as “student” is not a key in relation enrolled, however query 2 and query 4 always return same row sets

6. Consider the relation account (customer, balance) where customer is a primary key and there are no null values. We would like to rank customers according to decreasing balance. The customer with the largest balance gets rank 1. Ties are not broken but ranks are skipped: if exactly two customers have the largest balance they each get rank 1 and rank 2 is not assigned.

Query1:

```
select A.customer, count(B.customer) from account A, account B where A.balance
<=B.balance group by A.customer
```

Query2:

```
select A.customer, 1+count(B.customer) from account A, account B where A.balance <
B.balance group by A.customer
```

Consider these statements about Query1 and Query2.

1. Query1 will produce the same row set as Query2 for some but not all databases.
2. Both Query1 and Query2 are correct implementation of the specification
3. Query1 is a correct implementation of the specification but Query2 is not
4. Neither Query1 nor Query2 is a correct implementation of the specification
5. Assigning rank with a pure relational query takes less time than scanning in decreasing balance

order assigning ranks using ODBC.

Which two of the above statements are correct?

- (a) 2 and 5
- (b) 1 and 3
- (c) 1 and 4
- (d) 3 and 5

**ANS.** (c) Query 1 and Query 2 will give the same result if all the customers have distinct balance. So, for some databases, the result of query 1 and query 2 will be same.

Query 1:

```
+-----+-----+
| customer      | count(B.customer) |
```



Geto Suguru	5	
Utahime Iori	4	
Gojo Saturo	4	
Shoko Ieri	2	
Kento Nanami	2	

Query 2:

customer	count(B.customer)	
Geto Suguru	5	
Utahime Iori	4	
Gojo Saturo	4	
Shoko Ieri	2	
Kento Nanami	2	

7. Database table by name Loan\_Records is given below.

Borrower	Bank_Manager Loan_Amount
Ramesh Sunderajan	10000.00
Suresh Ramgopa	15000.00
Mahesh Sunderajan	7000.00

What is the output of the following SQL query?

```
SELECT Count(*)FROM ( (SELECT Borrower, Bank_Manager FROM Loan_Records) AS S NATURAL
JOIN (SELECT Bank_Manager, Loan_Amount FROM Loan_Records) AS T );
```

(a) 3 (b) 5 (c) 9 (d) 6

ANS. (b) 5

Count(*)	

```
|      5 |
+-----+
```

8. Consider the table employee(empId, name, department, salary) and the two queries Q1, Q2 below. Assuming

that department 5 has more than one employee, and we want to find the employees who get higher salary than

anyone in the department 5, which one of the statements is TRUE for any arbitrary employee table?

Q1 : Select e.empId From employee e Where not exists (Select \* From employee s where s.department = "5" and s.salary >=e.salary)

Q2 : Select e.empId From employee e Where e.salary > Any (Select distinct salary From employee s Where s.department = "5")

- (a) Q1 is the correct query
- (b) Q2 is the correct query
- (c) Both Q1 and Q2 produce the same answer.
- (d) Neither Q1 nor Q2 is the correct query

ANS. (a) Q1 is the correct query

Q1 results in Empty set

Q2 results in

```
+-----+
| empId |
+-----+
| e4    |
+-----+
```

9. Given relations r(w, x) and s(y, z), the result of select distinct w, x from r, s is guaranteed to be same as r, provided

- (a) r has no duplicates and s is non-empty
- (b) r and s have no duplicates
- (c) s has no duplicates and r is non-empty
- (d) r and s have the same number of tuples

ANS. (b) r and s have no duplicates

10. Consider the following relations:

Student	
Roll_No	Student_Name
1	Raj
2	Rohit
3	Raj

Performance		
Roll_No	Course	Marks
1	Math	80
1	English	70
2	Math	75
3	English	80
2	Physics	65
3	Math	80

Consider the following SQL query.

**SELECT S. Student\_Name, sum (P.Marks) FROM Student S, Performance P WHERE S. Roll\_No =P.Roll\_No GROUP BY S.Student\_Name**

The number of rows that will be returned by the SQL query is \_\_\_\_\_.

The number of rows that will be returned by the SQL query is \_\_\_\_\_

(A) 0

(B) 1

(C) 2

(D) 3

**ANS. (c) 2**

```
+-----+-----+
| Student_Name | Marks |
|      Raj      |   310 |
|      Rohit     |   140 |
+-----+-----+
```

11. Consider the following relation

**Cinema (theater, address, capacity)**

Which of the following options will be needed at the end of the SQL query

**SELECT P1.address FROM Cinema P1**

such that it always finds the addresses of theaters of theaters with maximum capacity?

(a) WHERE P1.capacity > = Any (select P2. capacity from Cinema P2)

(b) WHERE P1.capacity > All (select max (P2. capacity) from Cinema P2)

(c) WHERE P1.capacity >Any (select max (P2. capacity) from Cinema P2)

(d) WHERE P1.capacity > = All (select P2. capacity from Cinema P2)

**ANS. (a) WHERE P1.capacity > = Any (select P2. capacity from Cinema P2)**

When the ALL condition is followed by a list, the optimizer expands the initial condition to all elements of the list and strings them together with AND operators.

When the ANY condition is followed by a list, the optimizer expands the initial condition to all elements of the list and strings them together with OR operators

## Assignment 8

1. The following relations keep track of airline flight information:

Flights (*flno: integer, from : string, to: string, distance: integer, departs: time, arrives: time, price: integer*)

Aircraft(*aid: integer, aname: string, cruisingrange: integer*)

Certified(*eid:integer, aid: integer*)

Employees(*eid: integer, ename: string, salary: integer*)

**ANS.**

```
mysql> CREATE TABLE Flights(FLNO VARCHAR(10),Origin
VARCHAR(30),Destination VARCHAR(30),DISTANCE INT,DEPARTS
TIME,ARRIVES TIME,PRICE INT);
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> INSERT INTO Flights values("AI-101","Delhi","New
York",7302,"1:45:00","6:50:00",30000);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> INSERT INTO Flights values("UA-121","Los
Angeles","Honolulu",2558,"2:45:00","8:00:00",15000);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Flights values("DL-541","Los
Angeles","Chicago",1745,"3:45:00","7:55:00",25000);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Flights values("UA-925","Madison","New
York",809,"1:45:00","3:55:00",14000);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Flights
values("AI-121","Delhi","Frankfurt",3800,"1:25:00","5:50:00",35000);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Flights
values("AI-20","Delhi","Kolkata",869,"2:00:00","4:10:00",4650);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Flights
values("UK-21","Delhi","Paris",4084,"2:30:00","8:55:00",14650);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> SELECT * FROM Flights;
```

FLNO	Origin	Destination	DISTANCE	DEPARTS	ARRIVES	PRICE
AI-101	Delhi	New York	7302	01:45:00	06:50:00	30000
UA-121	Los Angeles	Honolulu	2558	02:45:00	08:00:00	15000
DL-541	Los Angeles	Chicago	1745	03:45:00	07:55:00	25000
UA-925	Madison	New York	809	01:45:00	03:55:00	14000
AI-121	Delhi	Frankfurt	3800	01:25:00	05:50:00	35000
AI-20	Delhi	Kolkata	869	02:00:00	04:10:00	4650
UK-21	Delhi	Paris	4084	02:30:00	08:55:00	14650

7 rows in set (0.00 sec)

```
mysql> CREATE TABLE Aircraft(AID INT,AName VARCHAR(20),CruisingRange
INT);
```

Query OK, 0 rows affected (0.02 sec)

```
mysql> INSERT INTO Aircraft VALUES(112,"Boeing787-8",9000);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> INSERT INTO Aircraft VALUES(151,"Boeing777-200LR",9500);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> INSERT INTO Aircraft VALUES(135,"Airbus A330-300",8300);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Aircraft VALUES(135,"Airbus A320NEO",4500);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Aircraft VALUES(189,"Boeing787-8",9000);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Aircraft VALUES(191,"Boeing777-300ER",9300);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> INSERT INTO Aircraft VALUES(131,"Boeing787-9",9360);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> SELECT * FROM Aircraft;
```

```
+-----+-----+-----+
| AID | AName | CruisingRange |
```

```

+-----+-----+-----+
| 112 | Boeing787-8 | 9000 |
| 151 | Boeing777-200LR | 9500 |
| 135 | Airbus A330-300 | 8300 |
| 144 | Airbus A320NEO | 4500 |
| 189 | Boeing787-8 | 9000 |
| 191 | Boeing777-300ER | 9300 |
| 131 | Boeing787-9 | 9360 |

```

```

+-----+-----+-----+

```

7 rows in set (0.00 sec)

```
mysql> CREATE TABLE Certified(EID INT,AID INT);
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> INSERT INTO Certified VALUES(591,112);
```

Query OK, 1 row affected (0.09 sec)

```
mysql> INSERT INTO Certified VALUES(601,112);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Certified VALUES(621,151);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Certified VALUES(641,135);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> INSERT INTO Certified VALUES(661,144);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Certified VALUES(681,144);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Certified VALUES(701,144);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> INSERT INTO Certified VALUES(721,144);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> SELECT * FROM Certified;
```

```

+-----+-----+

```

```
| EID | AID |  
+-----+-----+  
| 591 | 112 |  
| 601 | 112 |  
| 621 | 151 |  
| 641 | 135 |  
| 661 | 144 |  
| 681 | 144 |  
| 701 | 144 |  
| 721 | 144 |  
+-----+-----+
```

8 rows in set (0.01 sec)

```
mysql> CREATE TABLE Employees(EID INT,Ename VARCHAR(20),Salary INT);
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> INSERT INTO Employees VALUES(591,"Devi Sharan",12000);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> INSERT INTO Employees VALUES(601,"Aditya",12900);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Employees VALUES(621,"Deepak",13900);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Employees VALUES(641,"Vasant",15000);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Employees VALUES(661,"Abhishek",15000);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Employees VALUES(681,"Devendra",16000);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Employees VALUES(701,"Sudarshan",16000);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> INSERT INTO Employees VALUES(721,"Sharad",16000);
```

Query OK, 1 row affected (0.00 sec)



```
mysql> SELECT * FROM Employees;
```

```
+-----+-----+-----+
| EID | Ename | Salary |
+-----+-----+-----+
| 591 | Devi Sharan | 12000 |
| 601 | Aditya | 12900 |
| 621 | Deepak | 13900 |
| 641 | Vasant | 15000 |
| 661 | Abhishek | 15000 |
| 681 | Devendra | 16000 |
| 701 | Sudarshan | 16000 |
| 721 | Sharad | 16000 |
+-----+-----+-----+
```

```
8 rows in set (0.00 sec)
```

**a. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs 80,000.**

```
mysql> SELECT DISTINCT a.AName FROM Aircraft a WHERE a.Aid IN(SELECT
C.aid FROM Certified C,Employees E WHERE C.EID = E.EID AND NOT EXISTS
(SELECT * FROM Employees E1 WHERE E1.EID = E.EID AND Salary<80000));
```

```
+-----+
| AName |
+-----+
| Boeing787-8 |
| Boeing777-200LR |
| Airbus A330-300 |
| Airbus A320NEO |
+-----+
```

```
4 rows in set (0.01 sec)
```

**b. .For each pilot who is certified for more than three aircrafts,find the eid and the maximum cruisingrange of the aircraft for which he/she is certified.**

```
mysql> SELECT DISTINCT a.AName FROM Aircraft a WHERE a.Aid IN(SELECT
C.aid FROM Certified C,Employees E WHERE C.EID = E.EID AND NOT EXISTS
(SELECT * FROM Employees E1 WHERE E1.EID = E.EID AND Salary<80000));
```

```
+-----+
```

```
| AName |
```

```
+-----+
```

```
| Boeing787-8 |
```

```
| Boeing777-200LR |
```

```
| Airbus A330-300 |
```

```
| Airbus A320NEO |
```

```
+-----+
```

```
4 rows in set (0.01 sec)
```

**c. Find the names of all pilots whose salary is less than the price of the cheapest route from Los Angeles to Honolulu.**

```
mysql> SELECT DISTINCT E.Ename FROM Employees E WHERE E.Salary<(SELECT
MIN(F.Price) FROM Flights F WHERE Origin="Los Angeles" AND
Destination="Honolulu");
```

```
Empty set (0.00 sec)
```

**d. For all aircrafts with cruisingrange over 1000 kms,find the name of the aircraft and the average salary of all pilots certified for this aircraft.**

```
mysql> SELECT A.AName,AVG(E.Salary) FROM Aircraft A,Employees E Group
by A.AName;
```

```
+-----+
```

```
| AName | AVG(E.Salary) |
```

```
+-----+
```

```
| Boeing787-9 | 55737.5000 |
```

```
| Boeing777-300ER | 55737.5000 |
| Boeing787-8 | 55737.5000 |
| Airbus A320NEO | 55737.5000 |
| Airbus A330-300 | 55737.5000 |
| Boeing777-200LR | 55737.5000 |
+-----+
6 rows in set (0.00 sec)
```

**e. Find the names of pilots certified for some Boeing aircraft.**

```
mysql> SELECT DISTINCT E.ENAME FROM Employees E,Certified C,Aircraft A
WHERE E.EID = C.EID AND C.AID = A.AID AND
```

```
A.AName = "Boeing777-200LR";
```

```
+-----+
| ENAME |
+-----+
| Deepak |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT DISTINCT E.ENAME FROM Employees E,Certified C,Aircraft A
WHERE E.EID = C.EID AND C.AID = A.AID AND
```

```
A.AName = "Boeing787-8";
```

```
+-----+
| ENAME |
+-----+
| Devi Sharan |
| Aditya |
+-----+
2 rows in set (0.00 sec)
```

**f. Find the aid's of all aircraft that can be used on routes from Los Angeles to Chicago.**

```
mysql> SELECT AName FROM Aircraft;
```

```
+-----+
```

```
| AName |
```

```
+-----+
```

```
| Boeing787-8 |
```

```
| Boeing777-200LR |
```

```
| Airbus A330-300 |
```

```
| Airbus A320NEO |
```

```
| Boeing787-8 |
```

```
| Boeing777-300ER |
```

```
| Boeing787-9 |
```

```
+-----+
```

```
7 rows in set (0.00 sec)
```

**g. Identify the routes that can be piloted by every pilot who makes more than \$100,000**

```
SELECT DISTINCT F.Origin, F.Destination FROM Flights F WHERE NOT EXISTS
(SELECT * FROM Employees E WHERE E.Salary>100000 AND NOT EXISTS (SELECT
* FROM Aircraft A, Certified C WHERE A.CruisingRange > F.Distance AND
E.EID = C.EID AND A.AID = C.AID))
```

**h. Find the ENames of pilots who can operate planes with CruisingRange greater than 3000 miles but are not certified on any Boeing Aircraft.**

```
mysql> SELECT DISTINCT E.ENAME FROM Employees E,Certified C,Aircraft A
WHERE E.EID = C.EID AND C.AID = A.AID AND CruisingRange>3000;
```

```
+-----+
```

```
| EName |
```

```
+-----+
```

```
| Devi Sharan |
```

```
| Aditya |
| Deepak |
| Vasant |
| Abhishek |
| Devendra |
| Sudarshan |
| Sharad |
+-----+
```

**i. A customer wants to travel from Madison to New York with no more than two changes of flight. List the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.**

```
SELECT F.DEPARTS FROM Flights F WHERE F.FLNO IN ((SELECT F0.FLNO FROM
Flights F0 WHERE F0.Origin ="Madison" AND F0.Destination ="New York" AND
F0.Arrives < '18:00') UNION (SELECT F0.FLNO FROM Flights F0,Flights F1
WHERE F0.Origin ="Madison" AND F0.Destination <> 'New York' AND
F1.Departs>F0.Arrives AND F1.Arrives < '18:00') UNION (SELECT F0.FLNO
FROM Flights F0, Flights F1, Flights F2 WHERE F0.Origin ="Madison"
AND F0.Destination=F1.Origin
AND F1.Destination=F2.Origin
AND F2.Destination='New York'
AND F0.Destination<>'New York'
AND F1.Destination<>'New York'
AND F1.Departs>F0.Arrives
AND F2.Departs>F1.Arrives
AND F1.Arrives<'18:00'))
```

**j. Compute the difference between the average salary of a pilot and the average salary of all employees (including pilots).**

```
SELECT Temp1.Avg - Temp2.Avg FROM (SELECT AVG(E.Salary) AS Avg FROM
Employees E WHERE E.EID IN (SELECT DISTINCT C.EID FROM Certified C)) AS
Temp1, (SELECT AVG(E1.Salary) AS Avg FROM Employees E1)AS Temp2
```

**k. Print the name and salary of every nonpilot whose salary is more than the average salary for pilots.**

```
SELECT E.ENAME, E.Salary FROM Employees E WHERE E.EID NOT IN (SELECT
DISTINCT C.EID FROM Certified C) AND E.Salary > (SELECT AVG (E1.Salary)
FROM Employees E1 WHERE E1 WHERE E1.EID IN (SELECT DISTINCT C1.EID FROM
Certified C1))
```

**l. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles.**

```
Select E.ENAME from Employees E,Certified C,Aircraft A where C.AID=A.AID
AND E.EID=C.EID Group By E.EID,E.ENAME HAVING Every
(A.CruisingRange>1000);
```

**m. Print the names of the employees who are certified on aircrafts with cruising range longer than 1000 miles but on two such aircrafts.**

```
: Select E.ENAME from Employees E,Certified C,Aircraft A where
C.AID=A.AID AND E.EID=C.EID Group By E.EID,E.ENAME HAVING Every
(A.CruisingRange>1000) AND Count(*)>1;
```

**n. Print the names of employees who are certified only on aircrafts with cruising longer than 1000 miles and who are certified on some Boeing aircraft.**

```
Select E.ENAME from Employees E,Certified C,Aircraft A where C.AID=A.AID
AND E.EID=C.EID Group By E.EID,E.ENAME HAVING Every
(A.CruisingRange>1000) AND ANY (A.AName='Boeing');
```