1. **From the Table DDL of employees table given below, Write a query to create a table employees**.

Note:

- Make sure to apply **NOT NULL** and Primary Key (PK) condition for the columns where ever applicable.
- While creating a table, table name and column name should be exactly same as given in DDL (including upper/lower case)

Table name: employees

Column Name	Data Type	Can be Nullable?
employee_id (PK)	int	No
first_name	varchar(25)	No
last_name	varchar(25)	No
email	varchar(25)	No
phone_number	varchar(20)	No
hire_date	date	No
job_id	varchar(10)	Yes
salary	decimal(8,2)	No
commission_pct	int	Yes
manager_id	int	Yes
department_id	int	Yes

Ans:

```
create table employees(
   employee_id int not null primary key,
   first_name varchar(25) not null,
   last_name varchar(25) not null,
   email varchar(25)not null,
   phone_number varchar(20) not null,
   hire_date date not null,
   job_id varchar(10),
   salary decimal(8,2) not null,
   commission_pct int ,
   manager_id int,
   department_id int
);
```

2. **Write a query to drop the table facilities.**
   Hint: Refer the table for the field names and its data type of the table.

Table name: facilities

Column Name	Data Type	Can be Nullable?
facid(PK)	int	No
name	varchar(100)	No
membercost	int	No
guestcost	int	No
initialoutlay	int	Yes
monthlymaintenance	int	Yes

Input format

The required tables are populated in the backend.

Ans:

```
drop table facilities;
```

3. **From the DDL given below,**

Write a query to change the column name zipcode to pincode in members table.

Table name: members

Column Name	Data Type	Can be Nullable?
memid(PK)	int	No
surname	varchar(200)	No
firstname	varchar(200)	No
address	varchar(300)	No
zipcode	int	No
telephone	varchar(20)	No
recommendedby	int	Yes
joindate	varchar(10)	No

Input format

The members table is already created.

Ans:

```
alter table members
change column zipcode pincode int;
```

4. **Write a query to insert any 4 records to the table 'employees'**
   The column names and data types are given below.

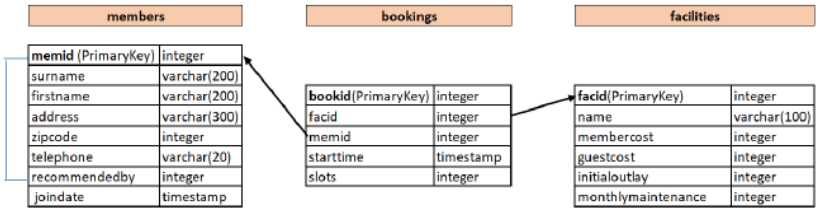
Column Name	Data type
EMPLOYEE_ID(PK)	INT
FIRST_NAME	VARCHAR (20)
LAST_NAME	VARCHAR (25)
EMAIL	VARCHAR (25)
PHONE_NUMBER	VARCHAR (20)
HIRE_DATE	DATE
JOB_ID	VARCHAR (10)
SALARY	DECIMAL (8, 2)
COMMISSION_PCT	INT
MANAGER_ID	INT
DEPARTMENT_ID	INT

Input format
The required input table is created in the back end.

Ans:

```
INSERT into
employees(EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMAIL,PHONE_NUMBER,HI
RE_DATE,JOB_ID,SALARY,COMMISSION_PCT,MANAGER_ID,DEPARTMENT_ID)

VALUES(1,'Adhi','B','adhi@gmail.com','129988','2004/04/22','100',700000.00,11,141,
111);

INSERT into
employees(EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMAIL,PHONE_NUMBER,HI
RE_DATE,JOB_ID,SALARY,COMMISSION_PCT,MANAGER_ID,DEPARTMENT_ID)

VALUES(2,'nikita','M','niki@gmail.com','16166','2022/04/04','200',600000.00,12,121,1
221);

INSERT into
employees(EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMAIL,PHONE_NUMBER,HI
RE_DATE,JOB_ID,SALARY,COMMISSION_PCT,MANAGER_ID,DEPARTMENT_ID)

VALUES(3,'monish','K','monish@gmail.com','18881','2004/08/18','300',650000.00,13
,131,1331);

INSERT into
employees(EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMAIL,PHONE_NUMBER,HI
RE_DATE,JOB_ID,SALARY,COMMISSION_PCT,MANAGER_ID,DEPARTMENT_ID)

VALUES(4,'aryan','M','aryan@gmail.com','19991','2022/05/13','400',200000.00,14,14
1,1441);
```

5. **Write a query to update facility name as 'Snooker Table-2' instead of facility name 'Pool Table' infacilities table.**
name column in facilities table refers to facilities name.
Refer the table details below:



members	
memid (PrimaryKey)	integer
surname	varchar(200)
firstname	varchar(200)
address	varchar(300)
zipcode	integer
telephone	varchar(20)
recommendedby	integer
joindate	timestamp

bookings	
bookid(PrimaryKey)	integer
facid	integer
memid	integer
starttime	timestamp
slots	integer

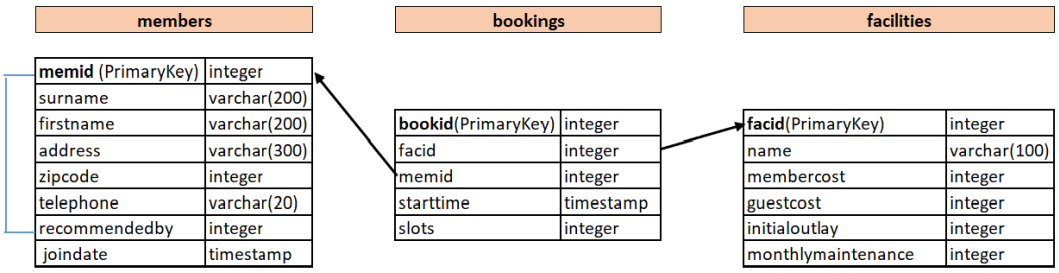
facilities	
facid(PrimaryKey)	integer
name	varchar(100)
membercost	integer
guestcost	integer
initialoutlay	integer
monthlymaintenance	integer

Input format
The required input table is created in the back end.

Ans:

update facilities set name='Snooker Table-2' where name='Pool Table';

6. **Write a query to delete all the records from 'members' table**
Refer the table details below:



members	
memid (PrimaryKey)	integer
surname	varchar(200)
firstname	varchar(200)
address	varchar(300)
zipcode	integer
telephone	varchar(20)
recommendedby	integer
joindate	timestamp

bookings	
bookid(PrimaryKey)	integer
facid	integer
memid	integer
starttime	timestamp
slots	integer

facilities	
facid(PrimaryKey)	integer
name	varchar(100)
membercost	integer
guestcost	integer
initialoutlay	integer
monthlymaintenance	integer

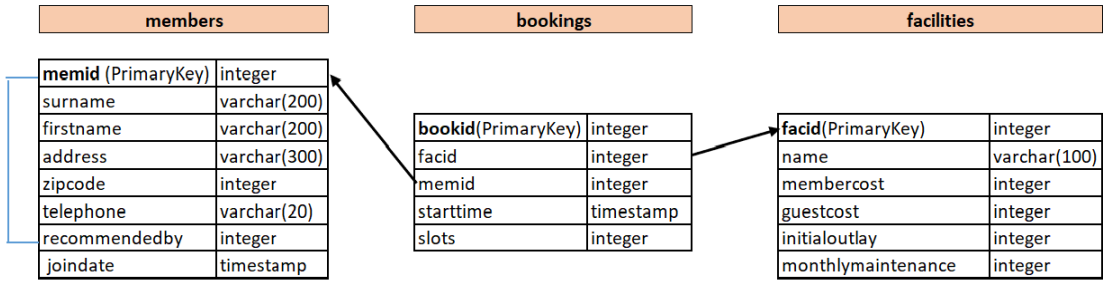
Note:
Table names are case sensitive.
members table is already created in the backend.

Input format
The required input table is created in the back end.

Ans:

delete from members;

7. **Write a query update the column membercost to 1000 from the table facilities where the monthlymaintenance is greater than 500.**



members

memid (PrimaryKey)	integer
surname	varchar(200)
firstname	varchar(200)
address	varchar(300)
zipcode	integer
telephone	varchar(20)
recommendedby	integer
joindate	timestamp

bookings

bookid(PrimaryKey)	integer
facid	integer
memid	integer
starttime	timestamp
slots	integer

facilities

facid(PrimaryKey)	integer
name	varchar(100)
membercost	integer
guestcost	integer
initialoutlay	integer
monthlymaintenance	integer

Note:
Table names are case sensitive.
facilities table is already created in the backend.

Input format
The required input table is created in the back end.

Ans:

```
update facilities

set membercost=1000

where monthlymaintenance>500;
```

8. **From the table schema given below, write a query to create a table orders.**

(Follow the same table names/column names as given along with the case)
The customer table is already created with primary key as **customerNumber**

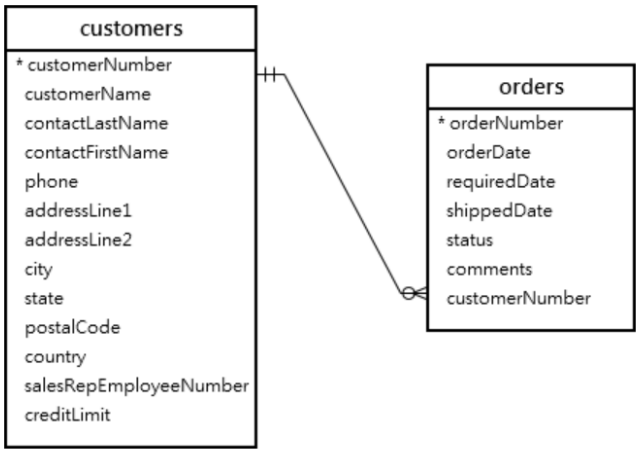
Note:

In **orders** table, **orderNumber**is the Primary key and **customerNumber** is the Foreign Key referencing customerNumber of customers table.
While creating orders table,

- orderNumber is the Primary Key
- define the foreign key relation between 2 tables.
- define all the columns with NOT NULL condition
- Follow the below data type:

orderNumber (int), orderDate(date), requiredDate(date), shippedDate(date), status(varchar(15)), comments(text),customerNumber(int)

Input format
The customers table is created in the backend.



Ans:

```
create table orders(
   orderNumber int not null primary key,
   orderDate date not null,
   requiredDate date not null,
   shippedDate date not null,
   status varchar(15) not null,
   comments text not null,
   customerNumber int not null,
   foreign key (customerNumber) references customers(customerNumber)
);
```

9. Adding Foreign key Constraint in the existing table

We have created an employee and department tables with the below definition without adding foreign key relation

Table name: employee

Column Name	Data Type
id (PK)	int
name	varchar(100)
address	varchar(100)
age	int
dob	date
deptId	int

Table name: department

Column Name	Data Type
departmentId (PK)	int
departmentName	varchar(100)

We want to link departmentId (department table) and deptId column (employee table) to maintain data integrity by establishing foreign key relation.

Write a alter query to employee table so that deptId will be foreign key referencing departmentId of department table.with reference name as depIdFk.

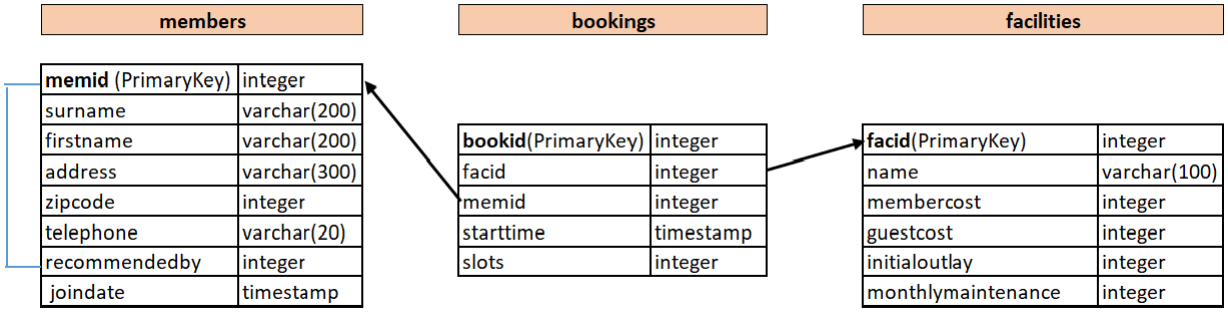
Input format
The employee and department table are created without referencing foreign key relation.

Ans:

```
alter table employee
add constraint deptIdFk
foreign key(deptId) references department(departmentId);
```

10. From the Schema given below, write a query to produce a count of the number of recommendations each member has made. Sort the output by ascending order of memid (recommendedby). If a particular member did not recommend anyone, ignore that memid. Output recommendedby and the corresponding count.

Note:
1. Refer table **members** from the given Schema.
2. 'recommendedby' field denotes 'memid' who recommended that particular member in the members table.
3. Table names are case sensitive.
4. Refer Output Format section for the output header names.



members

memid (PrimaryKey)	integer
surname	varchar(200)
firstname	varchar(200)
address	varchar(300)
zipcode	integer
telephone	varchar(20)
recommendedby	integer
joindate	timestamp

bookings

bookid(PrimaryKey)	integer
facid	integer
memid	integer
starttime	timestamp
slots	integer

facilities

facid(PrimaryKey)	integer
name	varchar(100)
membercost	integer
guestcost	integer
initialoutlay	integer
monthlymaintenance	integer

Input format
The required input table is populated in the back end.
Output format
The output should have the below header for the query to be considered.
recommendedby, count

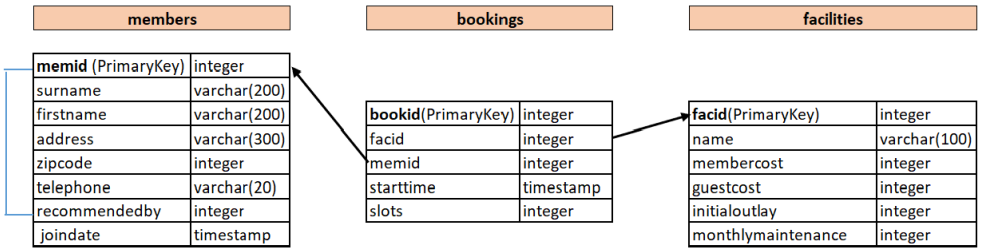
Ans:

```
select recommendedby,count(recommendedby) as count from members
where recommendedby is not null
group by recommendedby
order by recommendedby;
```

11. **From the Schema given below, write a query to produce a list of the total number of slots booked per facility per month for July and August 2012. (month should be fetched as 7 and 8 corresponding to July and August)**

    The output should have the details of facility id (facid), month, total slots booked sorted by the facid and month in ascending order.

    Note:
    1. Refer table **bookings** from the given Schema.
    2. Refer the field 'starttime' to filter month
    3. Table names are case sensitive.
    4. Refer Output Format section for the output header names.



members

memid (PrimaryKey)	integer
surname	varchar(200)
firstname	varchar(200)
address	varchar(300)
zipcode	integer
telephone	varchar(20)
recommendedby	integer
joindate	timestamp

bookings

bookid(PrimaryKey)	integer
facid	integer
memid	integer
starttime	timestamp
slots	integer

facilities

facid(PrimaryKey)	integer
name	varchar(100)
membercost	integer
guestcost	integer
initialoutlay	integer
monthlymaintenance	integer

Input format
The required input table is populated in the back end.
Output format
The output should have the below header for the query to be considered.
facid, month, Total Slots

Sample Output Rows:
facid | *month* | *Total Slots*
0 | 7 | 78
0 | 8 | 8
1 | 7 | 4
1 | 8 | 7

Ans:

```
select facid,extract(month from starttime) as month,sum(slots) as "Total Slots"
from bookings
where starttime>='2012-07-01' and starttime <='2012-08-31'
group by facid,month
order by facid,month;
```

12. **Given a table with order details. Please find the table details and sample data below.**

TABLE NAME: AGG_ORDERS
FIELD NAMES: ord_no, purchase_amt, ord_date, cust_id, salesman_id

ord_no	purchase_amt	ord_date	cust_id	salesman_id
20001	788.5	15-08-2019	123	5000
20005	975.5	15-01-2019	278	5012

Note: This is only a sample data.
Write a SQL statement to find the highest purchase amount with their Customer ID and order date, for only those customers who have the highest purchase amount in a day equal to any of these values 1000, 2200, 3700, 4000. Sort the records in ascending order of Customer ID.
Note: If a customer has done 2 purchases on a same day with purchase amount falls in our list (1000, 2200, 3700, 4000), then output should display both the records.Group by ord_no,cust_id,ord_date in same order.
Input format
The required input details will be populated in the backend.
Table Name: AGG_ORDERS
Output format
Output displays Customer ID, Order date along with their highest purchase amount from the given list.
Follow the output header as
Customer_ID, Date, Amount

Ans:

```
select cust_id as Customer_ID ,ord_date as Date,max(purchase_amt) as Amount

from AGG_ORDERS

where purchase_amt in (1000,2200,3700,4000)

group by ord_no,cust_id,ord_date

order by Customer_ID;
```

13. **You are given two tables of XYZ company, EMPLOYEE_XYZ and INCENTIVES_XYZ which has details about employees of a company and their incentives. Write a SQL Query that fetches the employee Id as ID, first name of the employees as First_name and sum of the incentives as Incentive for each IDs from the Incentives table.**

Display in ascending order of employee ID.
Table names are case sensitive.
TABLE 1: EMPLOYEE_XYZ
Column Names: Employee_id, First_name,Last_name,Salary,Joining_date,Department
TABLE 2: INCENTIVES_XYZ
Column Names: Id, Incentive_date,Incentive_amount

Below are only sample tables with sample data. Original data will be populated in the back end.

TABLE 1: EMPLOYEE_XYZ

Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000	11-10-2008	Banking
2	Michael	Clarke	800000	11-09-2007	Insurance
3	Roy	Thomas	700000	11-08-2006	Banking
4	Tom	Jose	600000	22-11-2008	Insurance

TABLE 2: INCENTIVES_XYZ

Id	Incentive_date	Incentive_amount
1	01-02-2013	5000
2	01-02-2013	3000

Input format
The required table will be populated in the back end.
Output format
The output consists of ID, First_name and Incentive.
Follow output header as below. (case sensitive)
ID, First_name, Incentive

Ans:

```
select Employee_id as ID,First_name as First_name,sum(Incentive_amount) as
Incentive from EMPLOYEE_XYZ e
join  INCENTIVES_XYZ i
on e.Employee_id=i.id
group by i.id
order by Employee_Id;
```

14. **Dave owns a book shop. He is planning to stock all the best selling books in his shop. He collected all the information on author, book names and the number of copies sold in the below format.**

Table1 Details: Table names are case sensitive
Table Name: authorBooks
Column Name: authorName, bookName

authorName	bookName
Daniel Pink	Drive
Leo Tolstoy	War and Peace
William Shakespeare	Hamlet
William Shakespeare	Othello
J.K.Rowling	Harry Potter
Agatha Christie	The Mouse Trap
Agatha Christie	And Then There Were None

Table2 Details:
Table Name: soldCopies
Column Name: bookName, soldCopies

bookName	soldCopies
Drive	30000
War and Peace	200000
Hamlet	500000
Othello	400000
Harry Potter	700000
The Mouse Trap	80000
And Then There Were None	100000

Create an SQL query that shows the top 3 authors who sold the most books in total along with total number of copies sold.
Sort the output in descending order of sold copies.

The headers in the output must be named as follows
Author_Namesold_sum

Input format
The input tables are populated in the back end.

Output format
The output consists of Author Name and sum of the books sold arranged in descending order based on sold copies.
The headers in the output must be named as follows (case sensitive)
Author_Name, sold_sum

Ans:

```
select authorName as Author_Name,sum(soldCopies) as sold_sum from authorBooks a
join soldCopies s
on a.bookName=s.bookName
group by a.authorName
order by sold_sum desc
limit 3;
```

15. **Here is the sample employee and salary information of ABC Infotech Limited company. The company has total of 30 employees.**

Below are only the sample tables. The original table will be populated in the backend

Table names are case sensitive
Table Name: EMPLOYEES_ABC
Column Names: employeeID, employeeName,departmentName

employeeID	employeeName	departmentName
1	John	Management
2	Smith	Sales
3	Dave	Management
4	Sebolt	Business Analyst
5	Francis	Systems Engineer

Table Name: SALARIES_ABC
Column Names: employeeID, employeeName,Salary

employeeID	employeeName	Salary
1	John	10000
2	Smith	8000
3	Dave	15000
4	Sebolt	12000
5	Francis	7000

Write a query to display every department name where the average salary per employee is lower than 7000.
The headers in the output must be named as follows
Department_Name, Avg_salaries

Input format
The required input table will be populated in the backend.
Output format
Output displays Department Names and average salaries.
The headers in the output must be named as follows (case sensitive)
Department_Name, Avg_salaries

Ans:

```
select departmentName as Department_Name,avg(Salary) as Avg_salaries
from EMPLOYEES_ABC e
join SALARIES_ABC s
on e.employeeID=s.employeeID
group by departmentName
having Avg_salaries<7000;
```

16. **Suppose that a website contains two tables, the Customers table, and the Orders table. Write a SQL query to find all customers who never order anything.**

Sort the result by ascending order of Customer Names

Table Details
Table1:
Tablename: **CUSTOMERS**
Columnname: id, cust_name

```
+----+-------+
| id | cust_name|
+----+-------+
| 1 | Prasanth |
| 2 | Harish |
| 3 | Sam  |
| 4 | Nadhini |
+----+-------+
```

Table2:
Table name: **ORDERS**
Column name: id, customer_id

```
+----+------------+
| id| customer_id|
+----+------------+
| 1 | 3     |
| 2 | 1     |
+----+------------+
```

Sample output

```
+-----------+
| Customers |
+-----------+
| Harish |
| Nandhini |
+-----------+
```

Note:
Table names are case-sensitive.
Use the same output header as given in the 'Output Format' section.
The above data are only sample data
Input format
The required tables will be populated in the back end.
Output format
The output will have the below header

Ans:

```
select cust_name as Customers from CUSTOMERS
where id not in (select customer_id from ORDERS)
order by cust_name asc;
```

17. **Write a query to display the name (first name and last name) for those employees who get more salary than the employee whose ID is 105.**

Table details are as follows:
Table Name: EMPLOYEES_SUBQ (Table names are case sensitive)

Column Names (along with data types)
employee_id - INT *Primary key*,
first_name - VARCHAR(30), last_name - VARCHAR(30),
email - VARCHAR(30), phone_number - VARCHAR(20),
hire_date - DATE,
job_id - VARCHAR(10),
salary - DECIMAL(18,2),
manager_id - INT,
department_id - INT

Input format
The required input table is populated in the back end.
Output format
The output header should be as follows:
first_name, last_name

Ans:

```
select first_name,last_name from EMPLOYEES_SUBQ where salary >
(select salary from EMPLOYEES_SUBQ WHERE employee_id=105);
```

18. **Write a query to display the name (first name and last name), salary, department id, job id for those employees who work in the same designation(job id) as the employee works whose id is 110.**

Table details are as follows:

Table Name: EMPLOYEES_SUBQ (Table names are case sensitive)

Column Names (along with data types)
employee_id - INT *Primary key*,
first_name - VARCHAR(30), last_name - VARCHAR(30),
email - VARCHAR(30), phone_number - VARCHAR(20),
hire_date - DATE,
job_id - VARCHAR(10),
salary - DECIMAL(18,2),
manager_id - INT,
department_id– INT

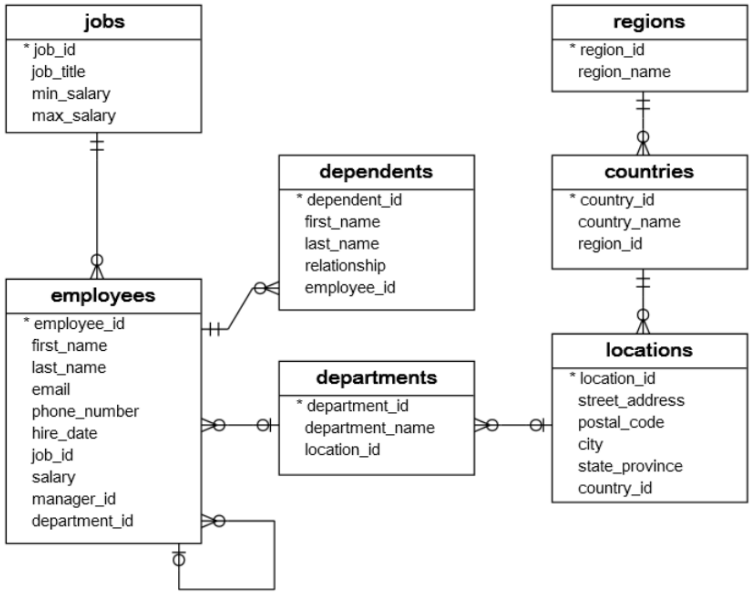
Input format
The required input table is populated in the back end.
Output format
The output header should be as follows:
first_name, last_name, salary, department_id, job_id

Ans:

```
select first_name,last_name,salary,department_id,job_id from EMPLOYEES_SUBQ
where job_id=(select job_id from EMPLOYEES_SUBQ where employee_id=110);
```

19. **From the below ER diagram of HR database, Write a query to find all the employees (employee_id, first_name, last_name, salary, department_id) whose salary is higher than the average salary of the employees in their departments. Use Correlated Subquery concept.**

Refer the tables employees from the below diagram for the column names (Table names are case sensitive)
Sort the output by ascending order of department_id, first_name, last_name.



Input format

The required input tables are populated in the back end.

Output format

The output header should be as follows:

employee_id, first_name, last_name, salary, department_id

Sort the output by ascending order of department_id, first_name, last_name.

Ans:

```
select employee_id,first_name,last_name,salary,department_id from employees e
where salary>(select avg(salary)from employees where
department_id=e.department_id)
order by department_id,first_name,last_name;
```

20. **From the following table, create a view 'newyorkstaff' for those salespersons belonging to the city 'New York'.**

Sample table: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13

Note: Table name and column names are case sensitive.

Output format

The output will display the View of salespersons belonging to the city 'New York'.

Ans:

```
create view newyorkstaff as

select * from salesman

where city='New York';
```

21. **From the following table, create a view 'totalforday' to count the number of unique customers, compute the average and total purchase amount of customer orders by each date. Round off the average and sum values to 2 decimal places using round() and Give an Alice name to Average as AVG and for the sum as SUM.**

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005

Note: Table name and column names are case sensitive.
Output column name

ord_date	dist_customer	AVG	SUM

Output format
The output will display order date, Count of distinct customer id followed by Average, and Sum value.

Ans:

```
create view totalforday as

select ord_date,count(distinct customer_id)as dist_customer,round(avg(purch_amt),2) as AVG,

round(sum(purch_amt),2) as SUM from orders

group by ord_date;
```

22. **From the following table, create a view 'gradecount' to count the number of customers in each grade.**

Table Name: **customer**

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002

Note: Table names and column names are case-sensitive.
Output should have following fields

```
grade   number
```

Output format
The output will display the number of customers in each grade using views.

Ans:

create view gradecount as

select grade,count(customer_id) as number

from customer

group by grade;

23. **Given the Employees table, Create a Cursor to find the list of all employees in a department with department Id passes as an argument.**

As Cursor can be implemented inside Stored Procedure, Create a procedure

Procedure Name: EmployeeList(IN dept_id INT, INOUT employee_list VARCHAR(2000))

Department ID should be passed as an argument. employee_list will have all the employee names corresponding to the particular department.

Table Name: Employees

EMPLOYEE_ID INT PRIMARY KEY,
EMP_NAME VARCHAR(100),
EMAIL VARCHAR(30),
PHONE_NUMBER VARCHAR(20),
HIRE_DATE DATE,
JOB_ID VARCHAR(10),
EMP_SALARY INTEGER,
MANAGER_ID INT,
DEPARTMENT_ID INT

Each Employee name should end with ;

Sample Output:
@employeelist
Dave Sebolt;BrianCaulfield;Robert Brown;

Ans:

```
DELIMITER $$
CREATE PROCEDURE EmployeeList(
In dept_id INT,INOUT employee_list VARCHAR(2000))

BEGIN
   DECLARE v_finished INTEGER DEFAULT 0;
   DECLARE v_ename VARCHAR(100) DEFAULT "";

   DECLARE C1 CURSOR FOR
      SELECT EMP_NAME FROM Employees WHERE DEPARTMENT_ID=dept_id;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;

   OPEN C1;

   get_emp: LOOP
      FETCH C1 INTO v_ename;
      IF v_finished = 1 THEN
         LEAVE get_emp;
      END IF;

         SET employee_list = CONCAT(employee_list,v_ename,";");
   END LOOP get_emp;
      CLOSE C1;
END$$
DELIMITER ;
```

Footer:

```
SET @employeelist = '';
CALL EmployeeList(97, @employeelist);
SELECT @employeelist;
```

24. **Given Product table, Implement a Cursor inside Stored Procedure to count how many products(product_name) of each product type exists with product type passed as a input argument to the procedure.**

As Cursor can be implemented inside Stored Procedure, Create a procedure
Procedure Name:
ProductCount(IN in_prod_type VARCHAR(30))

Product type should be passed as an argument.

Table Name: Product
product_id int
product_typevarchar(30)
product_namevarchar(30)

Count will be displayed with header name ProductCount

Sample Output:
ProductCount
10

Note:
Table names are case sensitive
Use proper delimiter to define procedure. At the end of procedure, reset the delimiter to ;
Implement the cursor concept inside the stored procedure. Calling the stored procedure will be taken care in the back end.
Input format
The required input table is created and populated in the backend.

Ans:

```
DELIMITER $$
CREATE PROCEDURE ProductCount(
IN in_prod_type varchar(30))
BEGIN
DECLARE v_finished integer default 0;
DECLARE count_ int default 0;
DECLARE pro_name varchar(100) default '';
declare C1 CURSOR for select product_type
from Product
where product_type=in_prod_type;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished =1;
OPEN C1;
Me :LOOP
FETCH C1 into pro_name;
if v_finished= 1 then leave Me;
end if;
SET count_=count_+1;
END LOOP;
```

select count_ as ProductCount;
CLOSE C1;
END $$
DELIMITER ;

Footer:

CALL ProductCount('Hygiene');

25. **Create a trigger named product_availability that inserts mapping records into the products_to_stores table.**
    This trigger is used to enforce business logic; in particular, it helps define the product availability for the different stores.
    When an item is being inserted into the products table, the trigger will check the availability field.
    If availability column is marked with the LOCAL value, the product will be made available in one store only. (We need to insert a new row in products_to_stores table with the new product id and store_id as 1.)

Example:
INSERT INTO products_to_stores (product_id, store_id) VALUES (*new product id*, 1);

Any other value will instruct the trigger to make the product available to the two stores that we created earlier.
(We need to insert 2 new rows in products_to_storestable with the new product id and store_id as 1 as one row and new product id and store_id 2 as second row)
Example:
INSERT INTO products_to_stores (product_id, store_id) VALUES (new product id, 1';
INSERT INTO products_to_stores (product_id, store_id) VALUES (new product id, 2);

Main Table Details:

stores	
store_id (PK)	BIGINT Auto increment
store_name	VARCHAR(50)

products	
product_id (PK)	BIGINT Auto increment
product_name	VARCHAR(40)
cost_price	DOUBLE
retail_price	DOUBLE
availability	VARCHAR(5)

products_to_stores Table:

products_to_stores	
ref_id (PK)	BIGINT Auto increment
product_id	BIGINT
store_id	BIGINT

Note:

1. **Use proper delimiters before defining triggers.**
2. **In Solution, write a trigger alone with the given specifications. The insert query and selecting products_to_stores table will be taken care in the backend.**

Input format

The required input table is populated in the backend.

Output format

The output will have the below header:

ref_id, product_id, store_id

Ans:

```
DELIMITER $$
create trigger product_availability
after insert on products
for each row
begin
if new.availability='LOCAL' THEN
insert into products_to_stores(product_id,store_id)
values(new.product_id,1);
else
insert into products_to_stores(product_id,store_id)
values(new.product_id,1);
insert into products_to_stores(product_id,store_id)
values(new.product_id,2);
end if;
end $$
DELIMITER ;
```

26. **Create a trigger named product_archiver that inserts a new row in archived_products table after we delete a particular product in products table.**

Main Table Details:

stores	
store_id (PK)	BIGINT Auto increment
store_name	VARCHAR(50)

products	
product_id (PK)	BIGINT Auto increment
product_name	VARCHAR(40)
cost_price	DOUBLE
retail_price	DOUBLE
availability	VARCHAR(5)

Note:

- **Use proper delimiters before defining triggers.**
- **In Solution, write a trigger alone with the given specifications. The delete query which will trigger the insert and selecting rows from archived_products table will be taken care in the backend.**

Input format
The required input tables are created and populated in the backend.
Output format
The output will have the below header:
product_id, product_name, cost_price, retail_price, availability

27. **Create a trigger called 'account_balance_validator' which should be triggered before update operation taking place on customer_accounts table.If customer account balance is updated the amount which is less than 0, then the account balance should be updated as 0.**

Table Name: customer_accounts

customer_accounts	
cust_id (PK)	INT
cust_name	VARCHAR(60)
account_no	VARCHAR(60)
account_balance	INT

Note:
- Use proper delimiters before defining triggers.
- In Solution, write a trigger alone with the given specifications. Updating sample row and selecting the rows from customer_accounts table will be taken care in the backend.
- Table names are case sensitive

Input format
The required input table is populated in the backend.
Output format
The output will have the below header:
cust_id, cust_name, account_no, account_balance

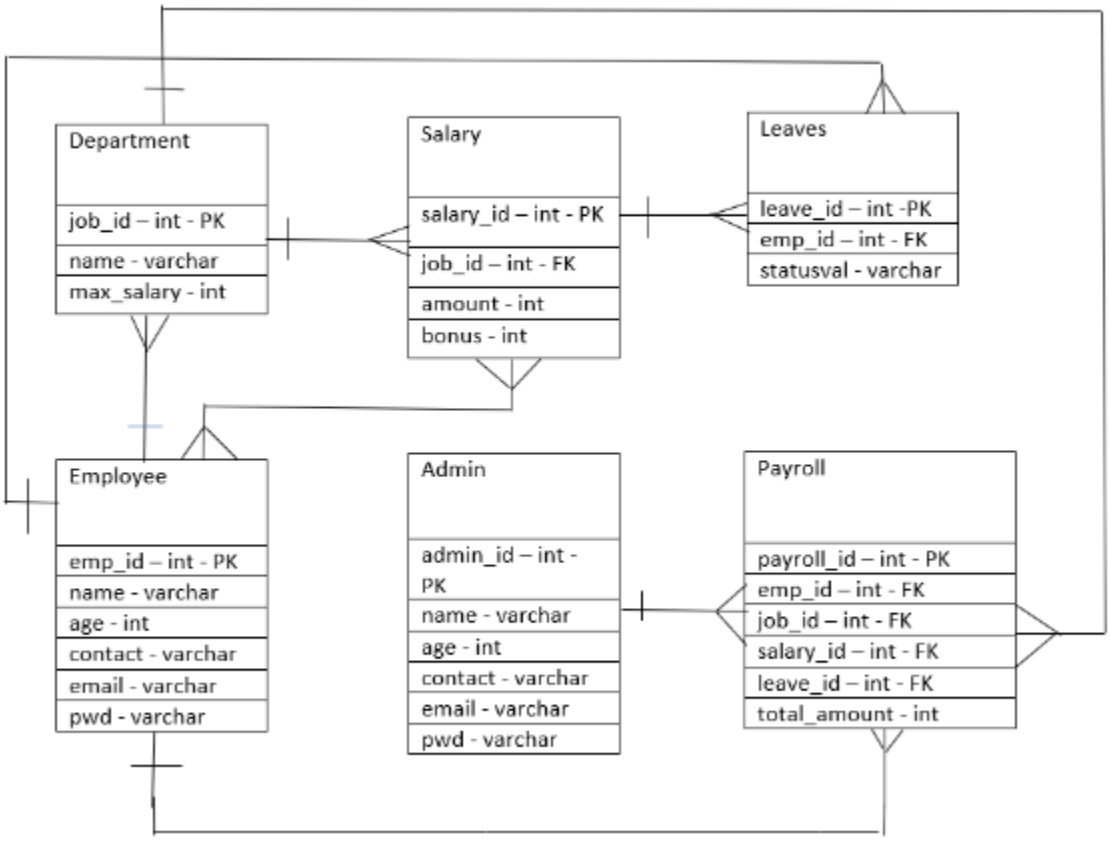
Ans:

```
delimiter $$
 create trigger account_balance_validator before
 update on customer_accounts
 for each row begin
 if new.account_balance<0
 then
 set new.account_balance=0;
 end if;
 end $$
 delimiter ;
```

Footer:

UPDATE customer_accounts
SET
account_balance = -56
WHERE cust_id = 3;

28. Given the schema of a Payroll Processing System,



Write a query to count the number of Employee with age<35.

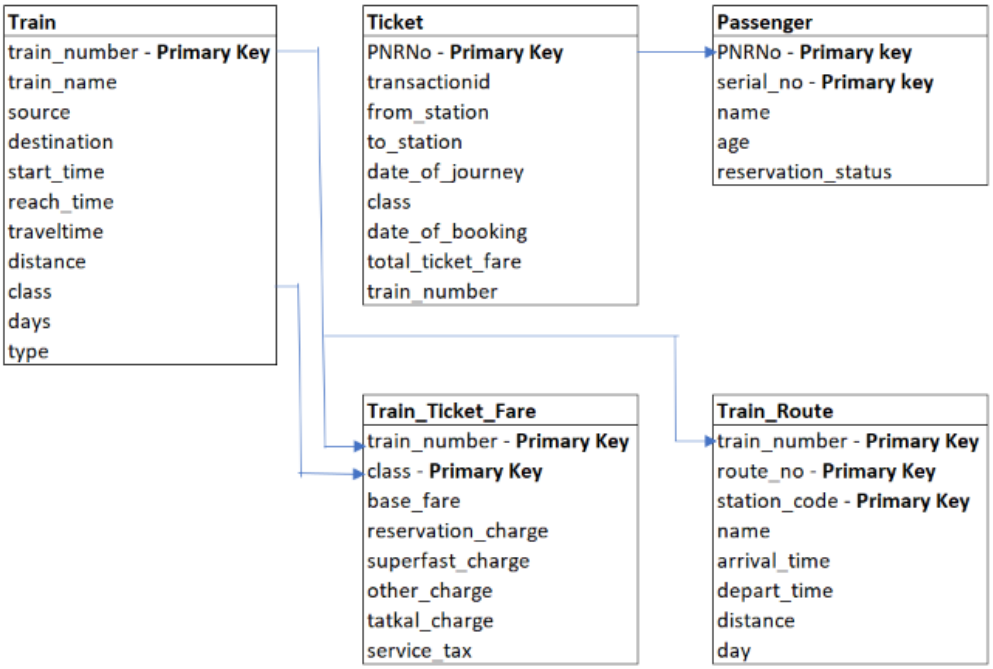
Note:
The required input tables are populated in the back end.
The table names are case-sensitive.
Refer to the schema and the output section for further clarifications.
Input format
No console input.
Output format
The output prints the count of the Employee with age<35.
Output Header: AgeCount

Ans:
select count(emp_id) as AgeCount from Employee where age<35;

29. Given the schema of Railway Reservation System, write a query to print the station codes replacing 'M' with 'K' along with the station name.Sort the output by ascending order of station code.

Note:

- Refer the table **Train_Route** from the given schema.
- Table names are case sensitive.
- Refer Output Format section for the output header names.



Input format

The required input tables are populated in the back end.

Output format

The output should have the below header for the query to be considered.

station_code, name

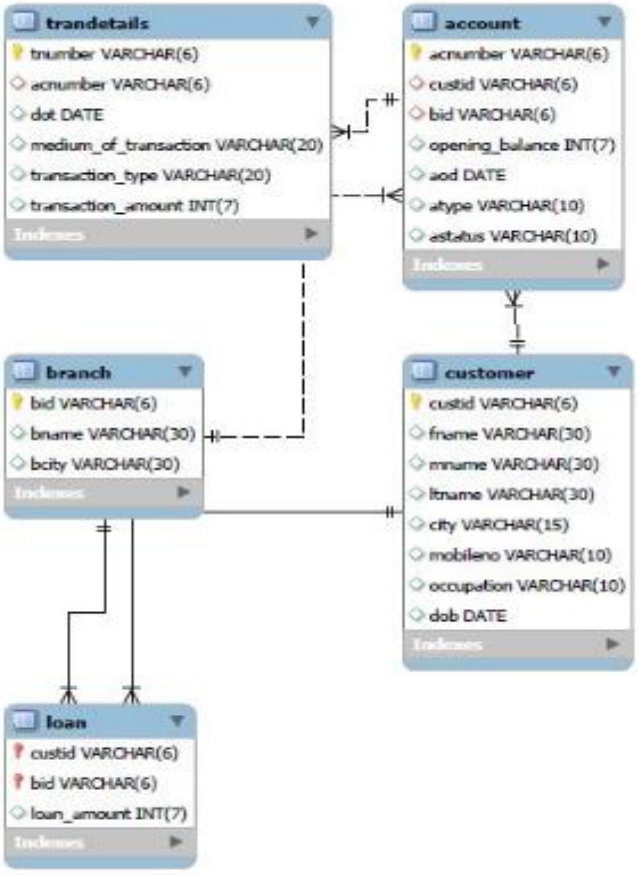
Ans:

select station_code,name from Train_Route

where station_code like 'M%'

order by station_code asc;

30. Given the schema of a Bank Management system



Write a query to display the custid, fname and customer's dob. Display in sorted order of date of birth year and within that sort by firstname.

Note:

The required input tables are populated in the back end.

The table names are case-sensitive.

Refer to the schema and the output section for further clarifications.

Input format

No console input.

Output format

The output prints the custid, fname and customer's dob in sorted order of date of birth year and within that in the sorted order of firstname.

Ans:

```
select custid,fname,dob from customer

order by dob,fname;
```