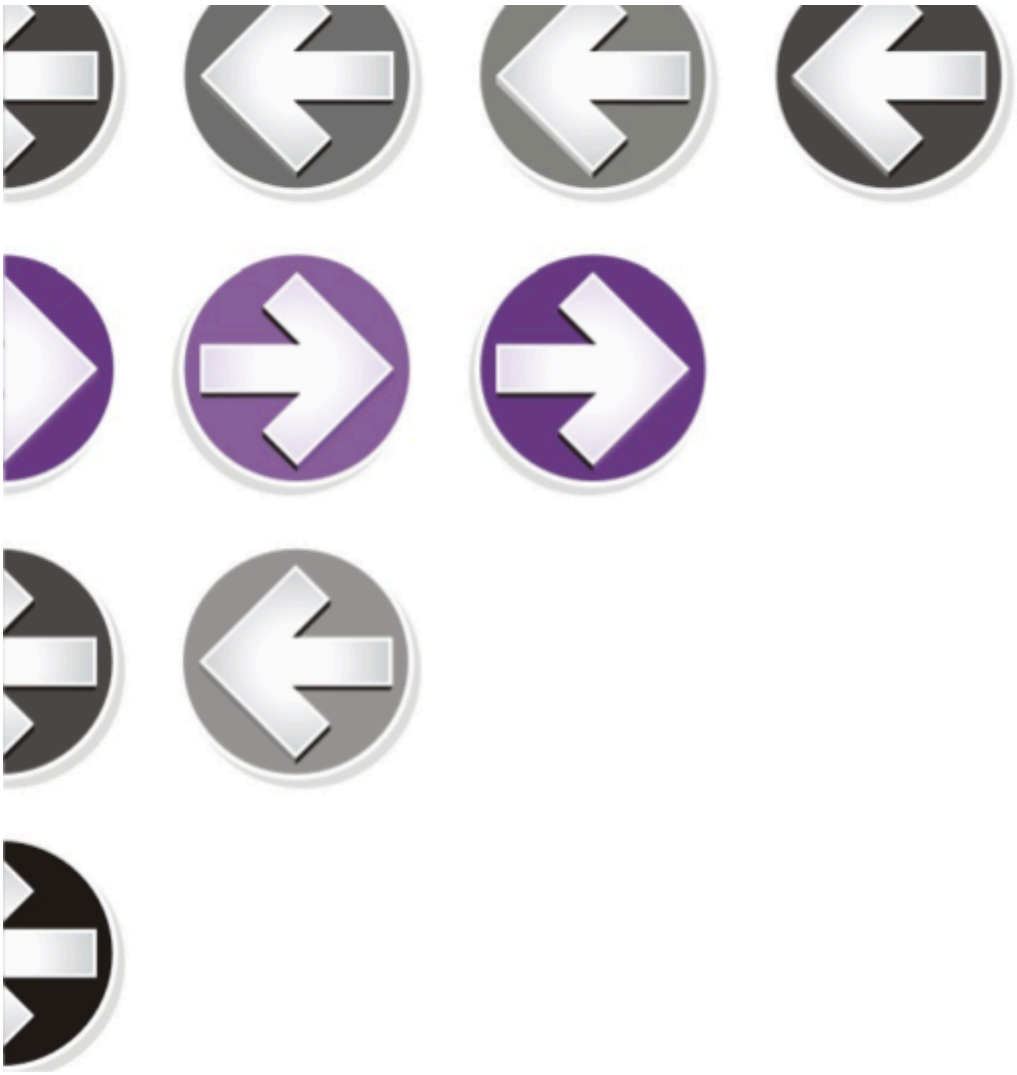


aspire SYSTEMS
Transforming Product Development

Assignments on SQL

This document contains few assignments on topics that covered under the SQL Course.

Talent Nurturing – Aspire Systems India Private Limited, Chennai

Summary

In this assignment, you are going to write SQL queries for each question that are given below. If you are having your own system you can install any database software like MySQL, MSSQL, Oracle, etc and create tables that are given below with the same structure. Sample data's that are provided below for those who are not having their own PC in home, to understand what type of data's will be there in the tables and you can insert the values as what you like. The purpose of this assignment is to assess your understanding of RDBMS concepts and how you are going to apply SQL Queries for each scenario that is given below.

Table Structure

1) Table Name	Programmer		
name	not null	varchar2(8)	name
dob	not null	date	date of birth
doj	not null	date	date of joining
sex	not null	varchar2(1)	male/ female
prof1		varchar2(8)	known language 1
prof2		varchar2(8)	known language 2
salary	not null	number(4)	salary

Sample Data						
somdutt	21-Apr-66	21-Apr-92	m	pascal	basic	3200

2) Table Name	Software		
name	not null	varchar2(8)	name
title	not null	varchar2(20)	developed project name
dev_in	not null	varchar2(8)	language developed
scost		number(7,2)	software cost
dcost		number(5)	development cost
sold		number(3)	number of software sold

Sample Data					
somdutt	parachutes	basic	399.95	6000	43

3) Table Name	Studies		
name	not null	varchar2(8)	name
splace	not null	varchar2(9)	studies place
course	not null	varchar2(5)	course studies
ccost	not null	varchar2(5)	course cost

Sample Data			
somdutt	sabhari	pgdca	4500
devdutt	bdps	dcs	5000

QUERIES - I

1) Find out the SELLING COST AVERAGE for the packages developed in PASCAL?

```
mysql> SELECT AVG(scost) AS average_selling_cost
    -> FROM Software
    -> WHERE dev_in = 'PASCAL';
+----------------------+
| average_selling_cost |
+----------------------+
|         10000.500000 |
+----------------------+
1 row in set (0.01 sec)
```

2) Display the names and ages of all programmers.

```
mysql> SELECT name, TIMESTAMPDIFF(YEAR, dob, CURDATE()) AS age
    -> FROM Programmer;
+-----------+------+
| name      | age  |
+-----------+------+
| John      |   33 |
| Alice     |   35 |
| Bob       |   29 |
| Emily     |   31 |
| Michael   |   38 |
| Sophia    |   25 |
| David     |   32 |
| Emma      |   36 |
| James     |   30 |
| Olivia    |   27 |
| William   |   34 |
| Ava       |   29 |
| Alexander |   27 |
| Mia       |   34 |
| Ethan     |   37 |
| Charlotte |   24 |
| Daniel    |   31 |
| Amelia    |   36 |
| Alexander |   32 |
| Abigail   |   28 |
| Liam      |   36 |
| Harper    |   26 |
| Henry     |   30 |
| Evelyn    |   28 |
+-----------+------+
24 rows in set (0.00 sec)
```

3) Display the names and ages of all the programmers who have undergone training in DCS course.

```
mysql> SELECT P.name, TIMESTAMPDIFF(YEAR, P.dob, CURDATE()) AS age
    -> FROM Programmer P
    -> JOIN Studies S ON P.name = S.name
    -> WHERE S.course = 'DCS';
Empty set (0.01 sec)
```

4) What is the highest numbers of copies sold by a package?

```
mysql> SELECT P.name, TIMESTAMPDIFF(YEAR, P.dob, CURDATE()) AS age
    -> FROM Programmer P
    -> JOIN Studies S ON P.name = S.name
    -> WHERE S.course = 'DCS';
Empty set (0.01 sec)
```

5) Display the names and date of birth of all the programmer born in JANUARY.

```
mysql> SELECT P.name, TIMESTAMPDIFF(YEAR, P.dob, CURDATE()) AS age
    -> FROM Programmer P
    -> JOIN Studies S ON P.name = S.name
    -> WHERE S.course = 'DCS';
Empty set (0.01 sec)
```

6) Display lowest course fee.

```
mysql> SELECT P.name, TIMESTAMPDIFF(YEAR, P.dob, CURDATE()) AS age
    -> FROM Programmer P
    -> JOIN Studies S ON P.name = S.name
    -> WHERE S.course = 'DCS';
Empty set (0.01 sec)
```

7) How many programmer has done PGDCA course.

```
mysql> SELECT P.name, TIMESTAMPDIFF(YEAR, P.dob, CURDATE()) AS age
    -> FROM Programmer P
    -> JOIN Studies S ON P.name = S.name
    -> WHERE S.course = 'DCS';
Empty set (0.01 sec)
```

8) How much revenue has been earned through sales of packages in C.

```
mysql> SELECT SUM(scost * sold) AS revenue
    -> FROM Software
    -> WHERE dev_in = 'C';
+---------+
| revenue |
+---------+
|    NULL |
+---------+
1 row in set (0.01 sec)
```

9) Display the details of software developed by Ramesh?

```
mysql> SELECT *
    -> FROM Software
    -> WHERE name = 'Ramesh';
Empty set (0.01 sec)
```

10) How many programmers studied at SABHARI.

```
mysql> SELECT COUNT(*) AS sabhari_programmers
    -> FROM Studies
    -> WHERE splace = 'SABHARI';
+---------------------+
| sabhari_programmers |
+---------------------+
|                   0 |
+---------------------+
1 row in set (0.00 sec)
```

11) Display the details of PACKAGES whose sales crossed the 20000 mark.

```
mysql> SELECT *
    -> FROM Software
    -> WHERE sold > 20000;
Empty set (0.01 sec)
```

12) Find out the number of copies which should be sold in order to recover the development cost of each package.

```
mysql> SELECT name, CEIL(dcost / scost) AS copies_to_recover_cost
    -> FROM Software;
+-----------+-------------------------+
| name      | copies_to_recover_cost |
+-----------+-------------------------+
| Abigail   |                       1 |
| Alexander |                       1 |
| Alice     |                       1 |
| Amelia    |                       1 |
| Ava       |                       1 |
| Bob       |                       1 |
| Charlotte |                       1 |
| Daniel    |                       1 |
| David     |                       1 |
| Emily     |                       1 |
| Emma      |                       1 |
| Ethan     |                       1 |
| Evelyn    |                       1 |
| Harper    |                       1 |
| Henry     |                       1 |
| James     |                       1 |
| John      |                       1 |
| Liam      |                       1 |
| Mia       |                       1 |
| Michael   |                       1 |
| Olivia    |                       1 |
| Ram       |                       1 |
| Sophia    |                       1 |
| William   |                       1 |
+-----------+-------------------------+
24 rows in set (0.00 sec)
```

13) What is the price of the costliest software developed in BASIC?

```
mysql> SELECT MAX(scost) AS costliest_software_price
    -> FROM Software
    -> WHERE dev_in = 'BASIC';
+---------------------------+
| costliest_software_price |
+---------------------------+
|                      NULL |
+---------------------------+
1 row in set (0.00 sec)
```

14) Display the details of packages for which development cost has been recovered.

```
mysql> SELECT *
    -> FROM Software
    -> WHERE sold * scost >= dcost;
+-----------+----------------------+------------+----------+-------+------+
| name      | title                | dev_in     | scost    | dcost | sold |
+-----------+----------------------+------------+----------+-------+------+
| Abigail   | calendar management  | JavaScript | 14000.45 |  6000 |   77 |
| Alexander | project management   | Java       | 11800.70 |  6200 |   78 |
| Alice     | library management   | Python     | 15000.75 |  6000 |   75 |
| Amelia    | time tracking        | Python     | 14900.40 |  5500 |   67 |
| Ava       | inventory tracking   | JavaScript | 13700.65 |  5600 |   62 |
| Bob       | accounting software  | C++        | 12000.25 |  5500 |   60 |
| Charlotte | document management  | JavaScript | 13900.75 |  5400 |   69 |
| Daniel    | ticketing system     | Java       | 11900.60 |  5800 |   76 |
| David     | e-learning platform  | C++        | 12500.70 |  5300 |   58 |
| Emily     | social network       | JavaScript | 13000.80 |  6500 |   70 |
| Emma      | customer  management | JavaScript | 13500.90 |  7000 |   80 |
| Ethan     | video conferencing   | C++        | 12900.95 |  5700 |   66 |
| Evelyn    | e-commerce           | JavaScript | 14100.55 |  6100 |   75 |
| Harper    | inventory management | Python     | 15000.90 |  6300 |   79 |
| Henry     | project management   | C++        | 13100.60 |  5700 |   64 |
| James     | point of sale        | Java       | 11500.55 |  5700 |   68 |
| John      | e-commerce           | Java       | 10000.50 |  5000 |   50 |
| Liam      | content management   | Java       | 12100.70 |  5800 |   70 |
| Mia       | expense tracking     | Python     | 14800.85 |  6100 |   74 |
| Michael   | inventory management | Java       | 11000.60 |  5200 |   55 |
| Olivia    | task management      | Python     | 14500.35 |  5400 |   63 |
| Ram       | Ecommece             | PASCAL     | 10000.50 |  5000 |   50 |
| Sophia    | hospital management  | Python     | 14000.40 |  5800 |   65 |
| William   | ticketing system     | C++        | 12700.45 |  5900 |   72 |
+-----------+----------------------+------------+----------+-------+------+
24 rows in set (0.00 sec)
```

15) How many packages were developed in dbase?

```
mysql> SELECT COUNT(*) AS dbase_packages
    -> FROM Software
    -> WHERE dev_in = 'dbase';
+-----------------+
| dbase_packages |
+-----------------+
|               0 |
+-----------------+
1 row in set (0.00 sec)
```

16) How many programmers studies at paragathi?

```
mysql> SELECT COUNT(DISTINCT name) AS paragathi_students
    -> FROM Studies
    -> WHERE splace = 'paragathi';
+--------------------+
| paragathi_students |
+--------------------+
|                  0 |
+--------------------+
1 row in set (0.00 sec)
```

17) How many programmers paid 5000 to 10000 for their course?

```
mysql> SELECT COUNT(*) AS programmers_paid_5k_to_10k
    -> FROM Studies
    -> WHERE ccost BETWEEN 5000 AND 10000;
+----------------------------+
| programmers_paid_5k_to_10k |
+----------------------------+
|                         22 |
+----------------------------+
1 row in set (0.01 sec)
```

18) What is the average course fee?

```
mysql> SELECT AVG(ccost) AS average_course_fee
    -> FROM Studies;
+---------------------+
| average_course_fee |
+---------------------+
|   6129.62962962963 |
+---------------------+
1 row in set (0.00 sec)
```

19) Display the details of programmers knowing c?

```
mysql> SELECT *
    -> FROM Programmer
    -> WHERE prof1 = 'C' OR prof2 = 'C';
Empty set (0.00 sec)
```

20) How many programmers know either Cobol or Pascal?

```
mysql> SELECT COUNT(*) AS cobol_or_pascal_programmers
    -> FROM Programmer
    -> WHERE prof1 IN ('Cobol', 'Pascal') OR prof2 IN ('Cobol', 'Pascal');
+----------------------------+
| cobol_or_pascal_programmers |
+----------------------------+
|                          0 |
+----------------------------+
1 row in set (0.01 sec)
```

21) How many programmers don't know Pascal & C?

```
mysql> SELECT COUNT(*) AS programmers_not_know_pascal_c
    -> FROM Programmer
    -> WHERE prof1 NOT IN ('Pascal', 'C') AND prof2 NOT IN ('Pascal', 'C');
+-------------------------------+
| programmers_not_know_pascal_c |
+-------------------------------+
|                            22 |
+-------------------------------+
1 row in set (0.00 sec)
```

22) How old is the oldest male programmers?

```
mysql> SELECT TIMESTAMPDIFF(YEAR, dob, CURDATE()) AS oldest_male_age
    -> FROM Programmer
    -> WHERE sex = 'male'
    -> ORDER BY dob DESC
    -> LIMIT 1;
Empty set (0.01 sec)
```

23) What is the average age of female programmers?

```
mysql> SELECT AVG(TIMESTAMPDIFF(YEAR, dob, CURDATE())) AS average_female_age
    -> FROM Programmer
    -> WHERE sex = 'female';
+--------------------+
| average_female_age |
+--------------------+
|               NULL |
+--------------------+
1 row in set (0.00 sec)
```

24) Calculate the experience in years for each programmers and display along with the names in descending order?

```
mysql> SELECT name, TIMESTAMPDIFF(YEAR, doj, CURDATE()) AS experience_years
    -> FROM Programmer
    -> ORDER BY experience_years DESC;
+-----------+------------------+
| name      | experience_years |
+-----------+------------------+
| Michael   |               14 |
| Ethan     |               13 |
| Liam      |               13 |
| Emma      |               12 |
| Amelia    |               12 |
| Alice     |               11 |
| William   |               11 |
| David     |               10 |
| Mia       |               10 |
| Emily     |                9 |
| Alexander |                9 |
| John      |                8 |
| James     |                8 |
| Daniel    |                8 |
| Henry     |                7 |
| Ava       |                6 |
| Abigail   |                6 |
| Evelyn    |                6 |
| Bob       |                5 |
| Alexander |                5 |
| Sophia    |                4 |
| Olivia    |                4 |
| Charlotte |                3 |
| Harper    |                3 |
+-----------+------------------+
24 rows in set (0.00 sec)
```

25) Who are the programmers who celebrate their birthday during the current month?

```
mysql> SELECT name
    -> FROM Programmer
    -> WHERE MONTH(dob) = MONTH(CURDATE());
+--------+
| name   |
+--------+
| David  |
| Amelia |
+--------+
2 rows in set (0.00 sec)
```

26) How many female programmers are there?

```
mysql> SELECT COUNT(*) AS female_programmers_count
    -> FROM Programmer
    -> WHERE sex = 'female';
+---------------------------+
| female_programmers_count |
+---------------------------+
|                         0 |
+---------------------------+
1 row in set (0.00 sec)
```

27) What are the languages known by the male programmers?

```
mysql> SELECT DISTINCT prof1, prof2
    -> FROM Programmer
    -> WHERE sex = 'male';
Empty set (0.01 sec)
```

28) What is the Average salary?

```
mysql> SELECT AVG(salary) AS average_salary
    -> FROM Programmer;
+----------------+
| average_salary |
+----------------+
|     57958.3333 |
+----------------+
1 row in set (0.00 sec)
```

29) How many people draw 2000 to 4000?

```
mysql> SELECT COUNT(*) AS people_salary_2k_to_4k
    -> FROM Programmer
    -> WHERE salary BETWEEN 2000 AND 4000;
+------------------------+
| people_salary_2k_to_4k |
+------------------------+
|                      0 |
+------------------------+
1 row in set (0.00 sec)
```

30) Display the details of those who don't know Clipper, Cobol or Pascal?

```
mysql> SELECT *
    -> FROM Programmer
    -> WHERE prof1 NOT IN ('Clipper', 'Cobol', 'Pascal')
    ->   AND prof2 NOT IN ('Clipper', 'Cobol', 'Pascal');
+-----------+------------+------------+-----+------------+------------+--------+
| name      | dob        | doj        | sex | prof1      | prof2      | salary |
+-----------+------------+------------+-----+------------+------------+--------+
| John      | 1990-05-15 | 2015-08-20 | M   | Java       | Python     |  50000 |
| Alice     | 1988-10-25 | 2012-04-10 | F   | C++        | JavaScript |  55000 |
| Emily     | 1992-07-12 | 2014-09-15 | F   | Java       | C#         |  60000 |
| Michael   | 1985-12-30 | 2010-02-18 | M   | JavaScript | SQL        |  65000 |
| Sophia    | 1998-08-07 | 2019-06-22 | F   | Python     | Java       |  52000 |
| David     | 1991-04-18 | 2013-07-30 | M   | C++        | Python     |  59000 |
| Emma      | 1987-09-05 | 2011-11-11 | F   | Java       | JavaScript |  63000 |
| James     | 1993-06-20 | 2016-03-25 | M   | Python     | Ruby       |  57000 |
| Olivia    | 1997-01-10 | 2020-01-05 | F   | JavaScript | C++        |  54000 |
| William   | 1989-11-28 | 2012-09-08 | M   | Java       | Python     |  61000 |
| Ava       | 1994-10-08 | 2017-07-14 | F   | C#         | Java       |  56000 |
| Alexander | 1996-09-03 | 2018-12-20 | M   | Python     | JavaScript |  59000 |
| Mia       | 1990-03-28 | 2013-08-30 | F   | Java       | C++        |  58000 |
| Charlotte | 1999-07-22 | 2021-02-15 | F   | C++        | Python     |  51000 |
| Daniel    | 1992-12-15 | 2015-04-20 | M   | JavaScript | Java       |  62000 |
| Amelia    | 1988-04-05 | 2011-10-10 | F   | Python     | JavaScript |  67000 |
| Alexander | 1991-06-18 | 2014-12-02 | M   | Java       | C++        |  66000 |
| Abigail   | 1995-09-20 | 2017-05-25 | F   | JavaScript | Python     |  55000 |
| Liam      | 1987-11-23 | 2010-09-28 | M   | C++        | Java       |  60000 |
| Harper    | 1998-02-14 | 2020-08-10 | F   | Python     | Java       |  53000 |
| Henry     | 1993-08-30 | 2016-11-15 | M   | JavaScript | C++        |  57000 |
| Evelyn    | 1996-01-02 | 2018-04-05 | F   | Java       | Python     |  59000 |
+-----------+------------+------------+-----+------------+------------+--------+
22 rows in set (0.00 sec)
```

31) How many Female programmers knowing C are above 24 years of age?

```
mysql> SELECT COUNT(*) AS female_programmers_above_24_with_c
    -> FROM Programmer
    -> WHERE sex = 'female' AND TIMESTAMPDIFF(YEAR, dob, CURDATE()) > 24
    -> AND (prof1 = 'C' OR prof2 = 'C');
+-----------------------------------+
| female_programmers_above_24_with_c |
+-----------------------------------+
|                                 0 |
+-----------------------------------+
1 row in set (0.01 sec)
```

32) Who are the programmers who will be celebrating their Birthday within a week?

```
mysql> SELECT name
    -> FROM Programmer
    -> WHERE DAYOFYEAR(dob) BETWEEN DAYOFYEAR(CURDATE()) AND DAYOFYEAR(DATE_ADD(CURDATE(), INTERVAL 7 DAY));
Empty set (0.01 sec)
```

33 Display the details of those with less than a year's experience?

```
mysql> SELECT *
    -> FROM Programmer
    -> WHERE TIMESTAMPDIFF(YEAR, doj, CURDATE()) < 1;
Empty set (0.00 sec)
```

34 Display the details of those who will be completing 2 years of service this year?

```
mysql> SELECT *
    -> FROM Programmer
    -> WHERE TIMESTAMPDIFF(YEAR, doj, CURDATE()) = 2;
Empty set (0.00 sec)
```

35 Calculate the amount to be recovered for those packages whose development cost has not been recovered?

```
mysql> SELECT name, (dcost - (sold * scost)) AS amount_to_recover
    -> FROM Software
    -> WHERE sold * scost < dcost;
Empty set (0.00 sec)
```

36) List the packages which have not been sold so far?

```
mysql> SELECT *
    -> FROM Software
    -> WHERE sold = 0;
Empty set (0.00 sec)
```

37) Find out the cost of the software developed by Mary?

```
mysql> SELECT scost
    -> FROM Software
    -> WHERE name = 'Mary';
Empty set (0.00 sec)
```

38) Display the institute's names from the studies table without duplicates?

```
mysql> SELECT DISTINCT splace
    -> FROM Studies;
+----------------+
| splace         |
+----------------+
| ABC University |
| XYZ College    |
| University A   |
| College B      |
| University C   |
| College D      |
| University E   |
| College F      |
| University G   |
| College H      |
| University I   |
| College J      |
| University K   |
| College L      |
| University M   |
| College N      |
| University O   |
| College P      |
| University Q   |
| College R      |
| University S   |
| College T      |
| University U   |
| College V      |
| University W   |
| BC University  |
+----------------+
```

39) How many different courses are mentioned in the studies table?

```
mysql> SELECT COUNT(DISTINCT course) AS distinct_courses_count
    -> FROM Studies;
+------------------------+
| distinct_courses_count |
+------------------------+
|                     27 |
+------------------------+
1 row in set (0.00 sec)
```

40) Display the names of the programmers whose names contain 2 occurrences of the letter A?

```
mysql> SELECT name
    -> FROM Programmer
    -> WHERE LENGTH(name) - LENGTH(REPLACE(name, 'A', '')) = 2;
Empty set (0.01 sec)
```

41) Display the names of programmers whose names contain unto 5 characters?

```
mysql> SELECT name
    -> FROM Programmer
    -> WHERE LENGTH(name) <= 5;
+-------+
| name  |
+-------+
| John  |
| Alice |
| Bob   |
| Emily |
| David |
| Emma  |
| James |
| Ava   |
| Mia   |
| Ethan |
| Liam  |
| Henry |
+-------+
12 rows in set (0.00 sec)
```

42) How many female programmers knowing COBOL have more than 2 years experience?

```
mysql> SELECT COUNT(*) AS female_programmers_with_cobol_more_than_2_years_exp
    -> FROM Programmer
    -> WHERE sex = 'female' AND prof1 = 'COBOL' AND TIMESTAMPDIFF(YEAR, doj, CURDATE()) > 2;
+-----------------------------------------------------+
| female_programmers_with_cobol_more_than_2_years_exp |
+-----------------------------------------------------+
|                                                   0 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

43) What is the length of the shortest name in the programmer table?

```
mysql> SELECT MIN(LENGTH(name)) AS shortest_name_length
    -> FROM Programmer;
+----------------------+
| shortest_name_length |
+----------------------+
|                    3 |
+----------------------+
1 row in set (0.00 sec)
```

44) What is the average development cost of a package developed in COBOL?

```
mysql> SELECT AVG(dcost) AS average_development_cost
    -> FROM Software
    -> WHERE dev_in = 'COBOL';
+--------------------------+
| average_development_cost |
+--------------------------+
|                     NULL |
+--------------------------+
1 row in set (0.00 sec)
```

45) Display the name, sex, dob (DD/MM/YY format), doj for all the programmers without using conversion function?

```
mysql> SELECT name, sex, DATE_FORMAT(dob, '%d/%m/%y') AS dob, DATE_FORMAT(doj, '%d/%m/%y') AS doj
    -> FROM Programmer;
+-----------+-----+----------+----------+
| name      | sex | dob      | doj      |
+-----------+-----+----------+----------+
| John      | M   | 15/05/90 | 20/08/15 |
| Alice     | F   | 25/10/88 | 10/04/12 |
| Bob       | M   | 02/03/95 | 05/11/18 |
| Emily     | F   | 12/07/92 | 15/09/14 |
| Michael   | M   | 30/12/85 | 18/02/10 |
| Sophia    | F   | 07/08/98 | 22/06/19 |
| David     | M   | 18/04/91 | 30/07/13 |
| Emma      | F   | 05/09/87 | 11/11/11 |
| James     | M   | 20/06/93 | 25/03/16 |
| Olivia    | F   | 10/01/97 | 05/01/20 |
| William   | M   | 28/11/89 | 08/09/12 |
| Ava       | F   | 08/10/94 | 14/07/17 |
| Alexander | M   | 03/09/96 | 20/12/18 |
| Mia       | F   | 28/03/90 | 30/08/13 |
| Ethan     | M   | 17/05/86 | 05/06/10 |
| Charlotte | F   | 22/07/99 | 15/02/21 |
| Daniel    | M   | 15/12/92 | 20/04/15 |
| Amelia    | F   | 05/04/88 | 10/10/11 |
| Alexander | M   | 18/06/91 | 02/12/14 |
| Abigail   | F   | 20/09/95 | 25/05/17 |
| Liam      | M   | 23/11/87 | 28/09/10 |
| Harper    | F   | 14/02/98 | 10/08/20 |
| Henry     | M   | 30/08/93 | 15/11/16 |
| Evelyn    | F   | 02/01/96 | 05/04/18 |
+-----------+-----+----------+----------+
24 rows in set (0.00 sec)
```

46) Who are the programmers who were born on the last day of the month?

```
mysql> SELECT name
    -> FROM Programmer
    -> WHERE DAY(LAST_DAY(dob)) = DAY(dob);
Empty set (0.01 sec)
```

47) What is the amount paid in salaries of the male programmers who do not know Cobol?

```
mysql> SELECT SUM(salary) AS total_salary_paid
    -> FROM Programmer
    -> WHERE sex = 'male' AND (prof1 != 'Cobol' OR prof2 != 'Cobol');
+-------------------+
| total_salary_paid |
+-------------------+
|              NULL |
+-------------------+
1 row in set (0.00 sec)
```

48) Display the title, scost, dcost and difference between scost and dcost in descending order of difference?

```
mysql> SELECT title, scost, dcost, (scost - dcost) AS cost_difference
    -> FROM Software
    -> ORDER BY cost_difference DESC;
+----------------------+----------+-------+-----------------+
| title                | scost    | dcost | cost_difference |
+----------------------+----------+-------+-----------------+
| time tracking        | 14900.40 |  5500 |         9400.40 |
| task management      | 14500.35 |  5400 |         9100.35 |
| library management   | 15000.75 |  6000 |         9000.75 |
| inventory management | 15000.90 |  6300 |         8700.90 |
| expense tracking     | 14800.85 |  6100 |         8700.85 |
| document management  | 13900.75 |  5400 |         8500.75 |
| hospital management  | 14000.40 |  5800 |         8200.40 |
| inventory tracking   | 13700.65 |  5600 |         8100.65 |
| e-commerce           | 14100.55 |  6100 |         8000.55 |
| calendar management  | 14000.45 |  6000 |         8000.45 |
| project management   | 13100.60 |  5700 |         7400.60 |
| video conferencing   | 12900.95 |  5700 |         7200.95 |
| e-learning platform  | 12500.70 |  5300 |         7200.70 |
| ticketing system     | 12700.45 |  5900 |         6800.45 |
| customer  management | 13500.90 |  7000 |         6500.90 |
| social network       | 13000.80 |  6500 |         6500.80 |
| accounting software  | 12000.25 |  5500 |         6500.25 |
| content management   | 12100.70 |  5800 |         6300.70 |
| ticketing system     | 11900.60 |  5800 |         6100.60 |
| inventory management | 11000.60 |  5200 |         5800.60 |
| point of sale        | 11500.55 |  5700 |         5800.55 |
| project management   | 11800.70 |  6200 |         5600.70 |
| e-commerce           | 10000.50 |  5000 |         5000.50 |
| Ecommece             | 10000.50 |  5000 |         5000.50 |
+----------------------+----------+-------+-----------------+
24 rows in set (0.00 sec)
```

49) Display the name, dob, doj of those month of birth and month of joining are same?

```
mysql> SELECT name, dob, doj
    -> FROM Programmer
    -> WHERE MONTH(dob) = MONTH(doj);
+--------+------------+------------+
| name   | dob        | doj        |
+--------+------------+------------+
| Olivia | 1997-01-10 | 2020-01-05 |
+--------+------------+------------+
1 row in set (0.00 sec)
```

50) Display the names of the packages whose names contain more than 1 word?

```
mysql> SELECT name
    -> FROM Software
    -> WHERE name LIKE '% %';
Empty set (0.01 sec)
```

QUERIES - II

1) Display THE NUMBER OF packages developed in EACH language

```
mysql> SELECT dev_in AS language, COUNT(*) AS num_packages
    -> FROM Software
    -> GROUP BY dev_in;
Empty set (0.10 sec)
```

.

2) Display THE NUMBER OF packages developed by EACH person.

```
mysql> SELECT name AS developer, COUNT(*) AS num_packages
    -> FROM Software
    -> GROUP BY name;
Empty set (0.07 sec)
```

3) Display THE NUMBER OF male and female programmer.

```
mysql> SELECT sex, COUNT(*) AS num_programmers
    -> FROM programmer
    -> GROUP BY sex;
+-----+-----------------+
| sex | num_programmers |
+-----+-----------------+
| M   |              12 |
| F   |              12 |
+-----+-----------------+
2 rows in set (0.00 sec)
```

4) Display THE COSTLIEST packages and HIGEST selling developed in EACH language.

```
mysql> SELECT dev_in AS language, MAX(dcost) AS costliest_package, MAX(scost) AS highest_selling
    -> FROM Software
    -> GROUP BY dev_in;
Empty set (0.07 sec)
```

5) Display THE NUMBER OF people BORN in EACH YEAR.

```
mysql> SELECT YEAR(dob) AS birth_year, COUNT(*) AS num_people
    -> FROM programmer
    -> GROUP BY YEAR(dob);
+------------+------------+
| birth_year | num_people |
+------------+------------+
|       1990 |          2 |
|       1988 |          2 |
|       1995 |          2 |
|       1992 |          2 |
|       1985 |          1 |
|       1998 |          2 |
|       1991 |          2 |
|       1987 |          2 |
|       1993 |          2 |
|       1997 |          1 |
|       1989 |          1 |
|       1994 |          1 |
|       1996 |          2 |
|       1986 |          1 |
|       1999 |          1 |
+------------+------------+
15 rows in set (0.07 sec)
```

6) Display THE NUMBER OF people JOINED in EACH YEAR.

```
mysql> SELECT YEAR(doj) AS join_year, COUNT(*) AS num_people
    -> FROM programmer
    -> GROUP BY YEAR(doj);
+-----------+------------+
| join_year | num_people |
+-----------+------------+
|      2015 |          2 |
|      2012 |          2 |
|      2018 |          3 |
|      2014 |          2 |
|      2010 |          3 |
|      2019 |          1 |
|      2013 |          2 |
|      2011 |          2 |
|      2016 |          2 |
|      2020 |          2 |
|      2017 |          2 |
|      2021 |          1 |
+-----------+------------+
12 rows in set (0.00 sec)
```

7) Display THE NUMBER OF people BORN in EACH MONTH.

```
mysql> SELECT MONTH(dob) AS birth_month, COUNT(*) AS num_people
    -> FROM programmer
    -> GROUP BY MONTH(dob);
+-------------+------------+
| birth_month | num_people |
+-------------+------------+
|           5 |          2 |
|          10 |          2 |
|           3 |          2 |
|           7 |          2 |
|          12 |          2 |
|           8 |          2 |
|           4 |          2 |
|           9 |          3 |
|           6 |          2 |
|           1 |          2 |
|          11 |          2 |
|           2 |          1 |
+-------------+------------+
12 rows in set (0.07 sec)
```

8) Display THE NUMBER OF people JOINED in EACH MONTH.

```
mysql> SELECT MONTH(doj) AS join_month, COUNT(*) AS num_people
    -> FROM programmer
    -> GROUP BY MONTH(doj);
+------------+------------+
| join_month | num_people |
+------------+------------+
|          8 |          3 |
|          4 |          3 |
|         11 |          3 |
|          9 |          3 |
|          2 |          2 |
|          6 |          2 |
|          7 |          2 |
|          3 |          1 |
|          1 |          1 |
|         12 |          2 |
|         10 |          1 |
|          5 |          1 |
+------------+------------+
12 rows in set (0.00 sec)
```

9) Display the language wise COUNTS of prof1.

```
mysql> SELECT prof1 AS language, COUNT(*) AS count_prof1
    -> FROM programmer
    -> GROUP BY prof1;
+------------+-------------+
| language   | count_prof1 |
+------------+-------------+
| Java       |           7 |
| C++        |           4 |
| Python     |           6 |
| JavaScript |           5 |
| C#         |           1 |
| SQL        |           1 |
+------------+-------------+
6 rows in set (0.00 sec)
```

10) Display the language wise COUNTS of prof2.

```
mysql> SELECT prof2 AS language, COUNT(*) AS count_prof2
    -> FROM programmer
    -> GROUP BY prof2;
+------------+-------------+
| language   | count_prof2 |
+------------+-------------+
| Python     |           6 |
| JavaScript |           4 |
| NULL       |           2 |
| C#         |           1 |
| SQL        |           1 |
| Java       |           5 |
| Ruby       |           1 |
| C++        |           4 |
+------------+-------------+
8 rows in set (0.00 sec)
```

11) Display THE NUMBER OF people in EACH salary group.

```
mysql> SELECT FLOOR(salary / 1000) * 1000 AS salary_group, COUNT(*) AS num_people
    -> FROM programmer
    -> GROUP BY salary_group;
+--------------+------------+
| salary_group | num_people |
+--------------+------------+
|        50000 |          1 |
|        55000 |          2 |
|        48000 |          1 |
|        60000 |          2 |
|        65000 |          1 |
|        52000 |          1 |
|        59000 |          3 |
|        63000 |          1 |
|        57000 |          2 |
|        54000 |          1 |
|        61000 |          1 |
|        56000 |          1 |
|        58000 |          1 |
|        64000 |          1 |
|        51000 |          1 |
|        62000 |          1 |
|        67000 |          1 |
|        66000 |          1 |
|        53000 |          1 |
+--------------+------------+
19 rows in set (0.07 sec)
```

12) Display THE NUMBER OF people who studied in EACH institute

```
mysql> SELECT splace, COUNT(*) AS num_people
    -> FROM Studies
    -> GROUP BY splace;
+----------------+------------+
| splace         | num_people |
+----------------+------------+
| ABC University |          1 |
| XYZ College    |          1 |
| University A   |          1 |
| College B      |          1 |
| University C   |          1 |
| College D      |          1 |
| University E   |          1 |
| College F      |          1 |
| University G   |          1 |
| College H      |          1 |
| University I   |          1 |
| College J      |          1 |
| University K   |          1 |
| College L      |          1 |
| University M   |          1 |
| College N      |          1 |
| University O   |          1 |
| College P      |          1 |
| University Q   |          1 |
| College R      |          1 |
| University S   |          1 |
| College T      |          1 |
| University U   |          1 |
| College V      |          1 |
| University W   |          1 |
+----------------+------------+
25 rows in set (0.00 sec)
```

13) Display THE NUMBER OF people who studied in EACH course.

```
mysql> SELECT course, COUNT(*) AS num_people
    -> FROM Studies
    -> GROUP BY course;
+------------------------+------------+
| course                 | num_people |
+------------------------+------------+
| PGDCA                  |          1 |
| BDPS DCS               |          1 |
| Computer Science       |          1 |
| Engineering            |          1 |
| Business Administration |          1 |
| Accounting             |          1 |
| Law                    |          1 |
| Medicine               |          1 |
| Psychology             |          1 |
| Biology                |          1 |
| Chemistry              |          1 |
| Mathematics            |          1 |
| Physics                |          1 |
| Literature             |          1 |
| History                |          1 |
| Sociology              |          1 |
| Anthropology           |          1 |
| Philosophy             |          1 |
| Political Science      |          1 |
| Economics              |          1 |
| Geography              |          1 |
| Environmental Science  |          1 |
| Foreign Languages      |          1 |
| Fine Arts              |          1 |
| Music                  |          1 |
+------------------------+------------+
25 rows in set (0.00 sec)
```

14) Display the TOTAL development COST of the packages developed in EACH language.

```
mysql> SELECT dev_in AS language, SUM(dcost) AS total_development_co
    -> FROM Software
    -> GROUP BY dev_in;
+------------+-------------------------+
| language   | total_development_cost |
+------------+-------------------------+
| JavaScript |                   36600 |
| Java       |                   33700 |
| Python     |                   35100 |
| C++        |                   28100 |
+------------+-------------------------+
4 rows in set (0.00 sec)
```

15) Display the selling cost of the package developed in EACH language

.

```
mysql> SELECT dev_in AS language, SUM(scost) AS total_selling_cost
    -> FROM Software
    -> GROUP BY dev_in;
+------------+--------------------+
| language   | total_selling_cost |
+------------+--------------------+
| JavaScript |           82204.10 |
| Java       |           68303.65 |
| Python     |           88203.65 |
| C++        |           63202.95 |
+------------+--------------------+
4 rows in set (0.00 sec)
```

16) Display the cost of the package developed by EACH programmer.

```
mysql> SELECT name AS programmer_name, SUM(dcost) AS total_development_cost
    -> FROM Software
    -> GROUP BY name;
+-----------------+------------------------+
| programmer_name | total_development_cost |
+-----------------+------------------------+
| Abigail         |                   6000 |
| Alexander       |                   6200 |
| Alice           |                   6000 |
| Amelia          |                   5500 |
| Ava             |                   5600 |
| Bob             |                   5500 |
| Charlotte       |                   5400 |
| Daniel          |                   5800 |
| David           |                   5300 |
| Emily           |                   6500 |
| Emma            |                   7000 |
| Ethan           |                   5700 |
| Evelyn          |                   6100 |
| Harper          |                   6300 |
| Henry           |                   5700 |
| James           |                   5700 |
| John            |                   5000 |
| Liam            |                   5800 |
| Mia             |                   6100 |
| Michael         |                   5200 |
| Olivia          |                   5400 |
| Sophia          |                   5800 |
| William         |                   5900 |
+-----------------+------------------------+
23 rows in set (0.07 sec)
```

17) Display the sales values of the package developed in EACH programmer.

```
mysql> SELECT name AS programmer_name, SUM(scost) AS total_selling_cost
    -> FROM Software
    -> GROUP BY name;
+-----------------+--------------------+
| programmer_name | total_selling_cost |
+-----------------+--------------------+
| Abigail         |           14000.45 |
| Alexander       |           11800.70 |
| Alice           |           15000.75 |
| Amelia          |           14900.40 |
| Ava             |           13700.65 |
| Bob             |           12000.25 |
| Charlotte       |           13900.75 |
| Daniel          |           11900.60 |
| David           |           12500.70 |
| Emily           |           13000.80 |
| Emma            |           13500.90 |
| Ethan           |           12900.95 |
| Evelyn          |           14100.55 |
| Harper          |           15000.90 |
| Henry           |           13100.60 |
| James           |           11500.55 |
| John            |           10000.50 |
| Liam            |           12100.70 |
| Mia             |           14800.85 |
| Michael         |           11000.60 |
| Olivia          |           14500.35 |
| Sophia          |           14000.40 |
| William         |           12700.45 |
+-----------------+--------------------+
23 rows in set (0.00 sec)
```

18) Display the NUMBER of packages developed by EACH programmer.

```
mysql> SELECT name AS programmer_name, COUNT(*) AS num_packages
    -> FROM Software
    -> GROUP BY name;
+-----------------+--------------+
| programmer_name | num_packages |
+-----------------+--------------+
| Abigail         |            1 |
| Alexander       |            1 |
| Alice           |            1 |
| Amelia          |            1 |
| Ava             |            1 |
| Bob             |            1 |
| Charlotte       |            1 |
| Daniel          |            1 |
| David           |            1 |
| Emily           |            1 |
| Emma            |            1 |
| Ethan           |            1 |
| Evelyn          |            1 |
| Harper          |            1 |
| Henry           |            1 |
| James           |            1 |
| John            |            1 |
| Liam            |            1 |
| Mia             |            1 |
| Michael         |            1 |
| Olivia          |            1 |
| Sophia          |            1 |
| William         |            1 |
+-----------------+--------------+
23 rows in set (0.00 sec)
```

19) Display the sales COST of packages developed by EACH programmer language wise.

```
mysql> SELECT name AS programmer_name, dev_in AS language, SUM(scost) AS total_selling_cost
    -> FROM Software
    -> GROUP BY name, dev_in;
+-----------------+------------+--------------------+
| programmer_name | language   | total_selling_cost |
+-----------------+------------+--------------------+
| Abigail         | JavaScript |           14000.45 |
| Alexander       | Java       |           11800.70 |
| Alice           | Python     |           15000.75 |
| Amelia          | Python     |           14900.40 |
| Ava             | JavaScript |           13700.65 |
| Bob             | C++        |           12000.25 |
| Charlotte       | JavaScript |           13900.75 |
| Daniel          | Java       |           11900.60 |
| David           | C++        |           12500.70 |
| Emily           | JavaScript |           13000.80 |
| Emma            | JavaScript |           13500.90 |
| Ethan           | C++        |           12900.95 |
| Evelyn          | JavaScript |           14100.55 |
| Harper          | Python     |           15000.90 |
| Henry           | C++        |           13100.60 |
| James           | Java       |           11500.55 |
| John            | Java       |           10000.50 |
| Liam            | Java       |           12100.70 |
| Mia             | Python     |           14800.85 |
| Michael         | Java       |           11000.60 |
| Olivia          | Python     |           14500.35 |
| Sophia          | Python     |           14000.40 |
| William         | C++        |           12700.45 |
+-----------------+------------+--------------------+
23 rows in set (0.00 sec)
```

20) Display EACH programmers name, costliest package and cheapest packages developed by Him/Her.

```
mysql> SELECT name AS programmer_name, MAX(dcost) AS costliest_package, MIN(dcost) AS cheapest_package
    -> FROM Software
    -> GROUP BY name;
+-----------------+------------------+------------------+
| programmer_name | costliest_package | cheapest_package |
+-----------------+------------------+------------------+
| Abigail         |             6000 |             6000 |
| Alexander       |             6200 |             6200 |
| Alice           |             6000 |             6000 |
| Amelia          |             5500 |             5500 |
| Ava             |             5600 |             5600 |
| Bob             |             5500 |             5500 |
| Charlotte       |             5400 |             5400 |
| Daniel          |             5800 |             5800 |
| David           |             5300 |             5300 |
| Emily           |             6500 |             6500 |
| Emma            |             7000 |             7000 |
| Ethan           |             5700 |             5700 |
| Evelyn          |             6100 |             6100 |
| Harper          |             6300 |             6300 |
| Henry           |             5700 |             5700 |
| James           |             5700 |             5700 |
| John            |             5000 |             5000 |
| Liam            |             5800 |             5800 |
| Mia             |             6100 |             6100 |
| Michael         |             5200 |             5200 |
| Olivia          |             5400 |             5400 |
| Sophia          |             5800 |             5800 |
| William         |             5900 |             5900 |
+-----------------+------------------+------------------+
23 rows in set (0.06 sec)
```

21) Display EACH language name with AVERAGE development cost, AVERAGE cost, selling

cost and AVERAGE price per copy.

```
mysql> SELECT dev_in AS language,
    ->        AVG(dcost) AS avg_development_cost,
    ->        AVG(scost) AS avg_selling_cost,
    ->        AVG(scost - dcost) AS avg_profit,
    ->        AVG(scost / sold) AS avg_price_per_copy
    -> FROM Software
    -> GROUP BY dev_in;
+------------+----------------------+------------------+-------------+--------------------+
| language   | avg_development_cost | avg_selling_cost | avg_profit  | avg_price_per_copy |
+------------+----------------------+------------------+-------------+--------------------+
| JavaScript |            6100.0000 |     13700.683333 | 7600.683333 |     191.1261156667 |
| Java       |            5616.6667 |     11383.941667 | 5767.275000 |     174.9819425000 |
| Python     |            5850.0000 |     14700.608333 | 8850.608333 |     209.6425635000 |
| C++        |            5620.0000 |     12640.590000 | 7020.590000 |     198.4188860000 |
+------------+----------------------+------------------+-------------+--------------------+
4 rows in set (0.00 sec)
```

22) Display EACH institute name with NUMBER of courses, AVERAGE cost per course

```
mysql> SELECT splace AS institute_name,
    ->        COUNT(*) AS num_courses,
    ->        AVG(ccost) AS avg_cost_per_course
    -> FROM Studies
    -> GROUP BY splace;
+----------------+-------------+----------------------+
| institute_name | num_courses | avg_cost_per_course |
+----------------+-------------+----------------------+
| ABC University |           1 |                 4500 |
| XYZ College    |           1 |                 5000 |
| University A   |           1 |                 6000 |
| College B      |           1 |                 7000 |
| University C   |           1 |                 5500 |
| College D      |           1 |                 4000 |
| University E   |           1 |                 8000 |
| College F      |           1 |                 9000 |
| University G   |           1 |                 6500 |
| College H      |           1 |                 5500 |
| University I   |           1 |                 7000 |
| College J      |           1 |                 6000 |
| University K   |           1 |                 7500 |
| College L      |           1 |                 4500 |
| University M   |           1 |                 5000 |
| College N      |           1 |                 5500 |
| University O   |           1 |                 6500 |
| College P      |           1 |                 6000 |
| University Q   |           1 |                 7000 |
| College R      |           1 |                 8000 |
| University S   |           1 |                 7500 |
| College T      |           1 |                 7000 |
| University U   |           1 |                 6000 |
| College V      |           1 |                 5500 |
| University W   |           1 |                 6500 |
+----------------+-------------+----------------------+
25 rows in set (0.00 sec)
```

23) Display EACH institute name with NUMBER of students.

```
mysql> SELECT splace AS institute_name,
    ->        COUNT(*) AS num_students
    -> FROM Studies
    -> GROUP BY splace;
+----------------+--------------+
| institute_name | num_students |
+----------------+--------------+
| ABC University |            1 |
| XYZ College    |            1 |
| University A   |            1 |
| College B      |            1 |
| University C   |            1 |
| College D      |            1 |
| University E   |            1 |
| College F      |            1 |
| University G   |            1 |
| College H      |            1 |
| University I   |            1 |
| College J      |            1 |
| University K   |            1 |
| College L      |            1 |
| University M   |            1 |
| College N      |            1 |
| University O   |            1 |
| College P      |            1 |
| University Q   |            1 |
| College R      |            1 |
| University S   |            1 |
| College T      |            1 |
| University U   |            1 |
| College V      |            1 |
| University W   |            1 |
+----------------+--------------+
25 rows in set (0.00 sec)
```

24) Display names of male and female programmers

```
mysql> SELECT name, sex
    -> FROM programmer;
+-----------+-----+
| name      | sex |
+-----------+-----+
| John      | M   |
| Alice     | F   |
| Bob       | M   |
| Emily     | F   |
| Michael   | M   |
| Sophia    | F   |
| David     | M   |
| Emma      | F   |
| James     | M   |
| Olivia    | F   |
| William   | M   |
| Ava       | F   |
| Alexander | M   |
| Mia       | F   |
| Ethan     | M   |
| Charlotte | F   |
| Daniel    | M   |
| Amelia    | F   |
| Alexander | M   |
| Abigail   | F   |
| Liam      | M   |
| Harper    | F   |
| Henry     | M   |
| Evelyn    | F   |
+-----------+-----+
24 rows in set (0.00 sec)
```

25) Display the programmer's name and their packages.

```
mysql> SELECT p.name AS programmer_name, s.title AS package_name
    -> FROM programmer p
    -> INNER JOIN Software s ON p.name = s.dev_in;
Empty set (0.07 sec)
```

26) Display the NUMBER of packages in EACH language.

```
mysql> SELECT dev_in AS language, COUNT(*) AS num_packages
    -> FROM Software
    -> GROUP BY dev_in;
+------------+--------------+
| language   | num_packages |
+------------+--------------+
| JavaScript |            6 |
| Java       |            6 |
| Python     |            6 |
| C++        |            5 |
+------------+--------------+
4 rows in set (0.00 sec)
```

27) Display the NUMBER of packages in EACH language for which development cost is less than 1000.

```
mysql> SELECT dev_in AS language, COUNT(*) AS num_packages
    -> FROM Software
    -> WHERE dcost < 1000
    -> GROUP BY dev_in;
Empty set (0.07 sec)
```

28) Display the AVERAGE difference BETWEEN scost and dcost for EACH language.

```
mysql> SELECT dev_in AS language, AVG(scost - dcost) AS avg_difference
    -> FROM Software
    -> GROUP BY dev_in;
+------------+----------------+
| language   | avg_difference |
+------------+----------------+
| JavaScript |    7600.683333 |
| Java       |    5767.275000 |
| Python     |    8850.608333 |
| C++        |    7020.590000 |
+------------+----------------+
4 rows in set (0.00 sec)
```

29) Display the TOTAL scost, dcsot and amount TOBE recovered for EACH programmer for whose dcost HAS NOT YET BEEN recovered.

```
mysql> SELECT s.dev_in AS programmer_name,
    ->        SUM(s.scost) AS total_selling_cost,
    ->        SUM(s.dcost) AS total_development_cost,
    ->        SUM(s.scost) - SUM(s.dcost) AS amount_to_recover
    -> FROM Software s
    -> GROUP BY s.dev_in
    -> HAVING SUM(s.dcost) < SUM(s.scost);
+-----------------+-------------------+-----------------------+-------------------+
| programmer_name | total_selling_cost | total_development_cost | amount_to_recover |
+-----------------+-------------------+-----------------------+-------------------+
| JavaScript      |          82204.10 |                 36600 |          45604.10 |
| Java            |          68303.65 |                 33700 |          34603.65 |
| Python          |          88203.65 |                 35100 |          53103.65 |
| C++             |          63202.95 |                 28100 |          35102.95 |
+-----------------+-------------------+-----------------------+-------------------+
4 rows in set (0.00 sec)
```

30) Display highest, lowest and average salaries for THOSE earning MORE than 2000.

```
mysql> SELECT MAX(salary) AS highest_salary,
    ->        MIN(salary) AS lowest_salary,
    ->        AVG(salary) AS average_salary
    -> FROM programmer
    -> WHERE salary > 2000;
+----------------+---------------+----------------+
| highest_salary | lowest_salary | average_salary |
+----------------+---------------+----------------+
|          67000 |         48000 |     57958.3333 |
+----------------+---------------+----------------+
1 row in set (0.00 sec)
```

QUERIES - III

1) Who is the highest paid C programmer?

```
mysql> SELECT name
    -> FROM programmer
    -> WHERE prof1 = 'C'
    -> ORDER BY salary DESC
    -> LIMIT 1;
Empty set (0.00 sec)
```

2) Who is the highest paid female cobol programmer?

```
mysql> SELECT name
    -> FROM programmer
    -> WHERE sex = 'F' AND prof1 = 'COBOL'
    -> ORDER BY salary DESC
    -> LIMIT 1;
Empty set (0.00 sec)
```

3) Display the name of the HIGEST paid programmer for EACH language (prof1)

```
mysql> SELECT p.prof1 AS language, p.name AS highest_paid_programmer
    -> FROM (
    ->     SELECT prof1, MAX(salary) AS max_salary
    ->     FROM programmer
    ->     WHERE prof1 IS NOT NULL
    ->     GROUP BY prof1
    -> ) AS max_salaries
    -> JOIN programmer AS p ON p.prof1 = max_salaries.prof1 AND p.salary = max_salaries.max_salary;
+------------+-------------------------+
| language   | highest_paid_programmer |
+------------+-------------------------+
| JavaScript | Michael                 |
| C#         | Ava                     |
| SQL        | Ethan                   |
| Python     | Amelia                  |
| Java       | Alexander               |
| C++        | Liam                    |
+------------+-------------------------+
6 rows in set (0.01 sec)
```

4) Who is the LEAST experienced programmer?

```
mysql> SELECT name
    -> FROM programmer
    -> ORDER BY doj
    -> LIMIT 1;
+---------+
| name    |
+---------+
| Michael |
+---------+
1 row in set (0.00 sec)
```

5) Who is the MOST experienced programmer?

```
mysql> SELECT name
    -> FROM programmer
    -> ORDER BY doj DESC
    -> LIMIT 1;
+-----------+
| name      |
+-----------+
| Charlotte |
+-----------+
1 row in set (0.00 sec)
```

6) Which language is known by ONLY ONE programmer?

```
mysql> SELECT prof1 AS language, COUNT(*) AS num_programmers
    -> FROM programmer
    -> GROUP BY prof1
    -> HAVING num_programmers = 1;
+----------+-----------------+
| language | num_programmers |
+----------+-----------------+
| C#       |               1 |
| SQL      |               1 |
+----------+-----------------+
2 rows in set (0.00 sec)
```

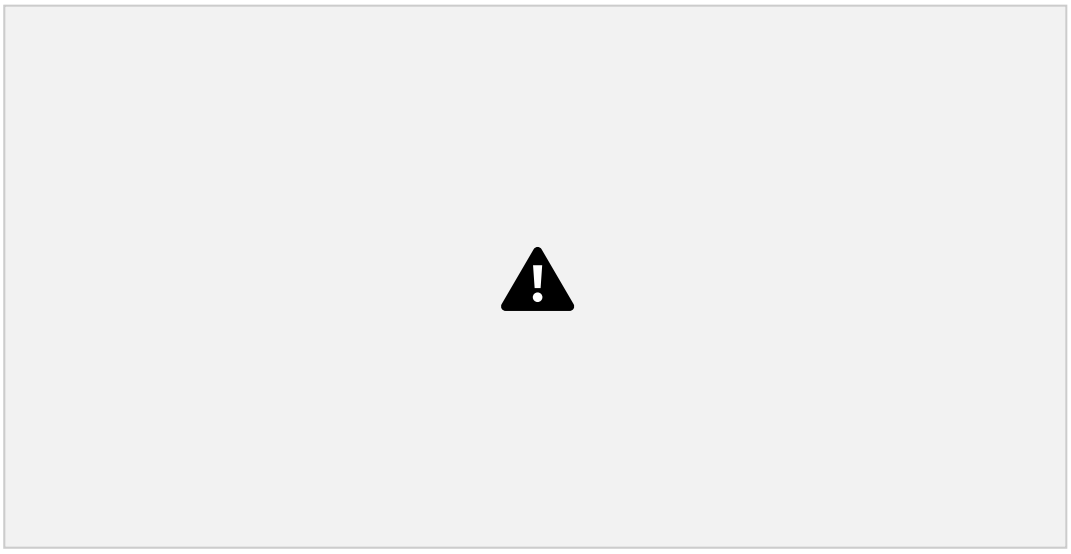
7) Who is the YONGEST programmer knowing DBASE?

```
mysql> SELECT name
    -> FROM programmer
    -> WHERE prof1 = 'SQL' OR prof2 = 'SQL'
    -> ORDER BY dob DESC
    -> LIMIT 1;
+-------+
| name  |
+-------+
| Ethan |
+-------+
1 row in set (0.00 sec)
```

8) Which institute has MOST NUMBER of students?

```
mysql> SELECT splace AS institute_name, COUNT(*) AS num_students
    -> FROM Studies
    -> GROUP BY splace
    -> ORDER BY num_students DESC
    -> LIMIT 1;
+---------------+--------------+
| institute_name | num_students |
+---------------+--------------+
| ABC University |            1 |
+---------------+--------------+
1 row in set (0.00 sec)
```

9) Who is the above programmer?



10) Which female programmer earns MORE than 3000/- but DOES NOT know C, C++, Oracle or Dbase?

```
mysql> SELECT name
    -> FROM programmer
    -> WHERE sex = 'F' AND salary > 3000
    -> AND prof1 NOT IN ('C', 'C++', 'Oracle', 'DBASE')
    -> AND prof2 NOT IN ('C', 'C++', 'Oracle', 'DBASE');
+---------+
| name    |
+---------+
| Emily   |
| Sophia  |
| Emma    |
| Ava     |
| Amelia  |
| Abigail |
| Harper  |
| Evelyn  |
+---------+
8 rows in set (0.00 sec)
```

11) Which is the COSTLIEST course?

```
mysql> SELECT course
    -> FROM Studies
    -> ORDER BY ccost DESC
    -> LIMIT 1;
+----------+
| course   |
+----------+
| Medicine |
+----------+
1 row in set (0.00 sec)
```

12) Which course has been done by MOST of the students?

```
mysql> SELECT course
    -> FROM Studies
    -> GROUP BY course
    -> ORDER BY COUNT(*) DESC
    -> LIMIT 1;
+--------+
| course |
+--------+
| PGDCA  |
+--------+
1 row in set (0.00 sec)
```

13) Display name of the institute and course Which has below AVERAGE course fee?

```
mysql> SELECT splace AS institute_name, course
    -> FROM Studies
    -> WHERE ccost < (SELECT AVG(ccost) FROM Studies);
+----------------+------------------------+
| institute_name | course                 |
+----------------+------------------------+
| ABC University | PGDCA                  |
| XYZ College    | BDPS DCS               |
| University A   | Computer Science       |
| University C   | Business Administration |
| College D      | Accounting             |
| College H      | Biology                |
| College J      | Mathematics            |
| College L      | Literature             |
| University M   | History                |
| College N      | Sociology              |
| College P      | Philosophy             |
| University U   | Foreign Languages      |
| College V      | Fine Arts              |
+----------------+------------------------+
13 rows in set (0.07 sec)
```

14) Which institute conducts COSTLIEST course?

```
mysql> SELECT splace AS institute_name
    -> FROM Studies
    -> WHERE ccost = (SELECT MAX(ccost) FROM Studies);
+----------------+
| institute_name |
+----------------+
| College F      |
+----------------+
1 row in set (0.00 sec)
```

15) Which course has below AVERAGE number of students?

```
mysql> SELECT course
    -> FROM Studies
    -> GROUP BY course
    -> HAVING COUNT(*) < (SELECT AVG(num_students) FROM (SELECT COUNT(*) AS num_students FROM Studies GROUP BY course) AS subquery);
Empty set (0.00 sec)
```

16) Which institute conducts the above course?

```
mysql> SELECT splace AS institute_name
    -> FROM Studies
    -> WHERE course = (SELECT course
    ->                 FROM Studies
    ->                 GROUP BY course
    ->                 HAVING COUNT(*) < (SELECT AVG(num_students) FROM (SELECT COUNT(*) AS num_students FROM Studies GROUP BY course) AS subquery)
    ->                 LIMIT 1);
Empty set (0.00 sec)
```

17) Display names of the course WHOSE fees are within 1000(+ or -) of the AVERAGE fee.

```
mysql> SELECT course
    -> FROM Studies
    -> WHERE ccost BETWEEN ((SELECT AVG(ccost) FROM Studies) - 1000) AND ((SELECT AVG(ccost) FROM Studies) + 1000);
+-------------------------+
| course                  |
+-------------------------+
| Computer Science        |
| Engineering             |
| Business Administration |
| Psychology              |
| Biology                 |
| Chemistry               |
| Mathematics             |
| Sociology               |
| Anthropology            |
| Philosophy              |
| Political Science       |
| Environmental Science   |
| Foreign Languages       |
| Fine Arts               |
| Music                   |
+-------------------------+
15 rows in set (0.07 sec)
```

18) Which package has the HIGEST development cost?

```
mysql> SELECT name
    -> FROM Software
    -> ORDER BY dcost DESC
    -> LIMIT 1;
+------+
| name |
+------+
| Emma |
+------+
1 row in set (0.00 sec)
```

19) Which package has the LOWEST selling cost?

```
mysql> SELECT name
    -> FROM Software
    -> ORDER BY scost ASC
    -> LIMIT 1;
+------+
| name |
+------+
| John |
+------+
1 row in set (0.00 sec)
```

20) Who developed the package, which has sold the LEAST number of copies?

```
mysql> SELECT dev_in
    -> FROM Software
    -> ORDER BY sold ASC
    -> LIMIT 1;
+--------+
| dev_in |
+--------+
| Java   |
+--------+
1 row in set (0.00 sec)
```

21) Which language was used to develop the package WHICH has the HIGEST sales amount?

```
mysql> SELECT dev_in
    -> FROM Software
    -> ORDER BY scost DESC
    -> LIMIT 1;
+--------+
| dev_in |
+--------+
| Python |
+--------+
1 row in set (0.00 sec)
```

22) How many copies of the package that has the LEAST DIFFRENCE between development and selling cost were sold?

```
mysql> SELECT sold
    -> FROM Software
    -> ORDER BY (scost - dcost) ASC
    -> LIMIT 1;
+------+
| sold |
+------+
|   50 |
+------+
1 row in set (0.00 sec)
```

23) Which is the COSTLIEAST package developed in PASCAL?

```
mysql> SELECT name
    -> FROM Software
    -> WHERE dev_in = 'PASCAL'
    -> ORDER BY dcost DESC
    -> LIMIT 1;
Empty set (0.00 sec)
```

24) Which language was used to develop the MOST NUMBER of package?

```
mysql> SELECT dev_in AS language, COUNT(*) AS num_packag
    -> FROM Software
    -> GROUP BY dev_in
    -> ORDER BY num_packages DESC
    -> ;
+------------+--------------+
| language   | num_packages |
+------------+--------------+
| JavaScript |            6 |
| Java       |            6 |
| Python     |            6 |
| C++        |            5 |
+------------+--------------+
4 rows in set (0.00 sec)
```

25) Which programmer has developed the HIGEST NUMBER of package?

```
mysql> SELECT name
    -> FROM (
    ->     SELECT name, COUNT(*) AS num_packages
    ->     FROM Software
    ->     GROUP BY name
    ->     ORDER BY num_packages DESC
    ->     LIMIT 1
    -> ) AS max_packages;
+---------+
| name    |
+---------+
| Abigail |
+---------+
1 row in set (0.00 sec)
```

26) Who is the author of the COSTLIEST package?

```
mysql> SELECT name
    -> FROM Software
    -> WHERE dcost = (
    ->     SELECT MAX(dcost)
    ->     FROM Software
    -> );
+------+
| name |
+------+
| Emma |
+------+
1 row in set (0.00 sec)
```

27) Display names of packages WHICH have been sold LESS THAN the AVERAGE number of copies?

```
mysql> SELECT title
    -> FROM Software
    -> WHERE sold < (SELECT AVG(sold) FROM Software);
+----------------------+
| title                |
+----------------------+
| time tracking        |
| inventory tracking   |
| accounting software  |
| e-learning platform  |
| video conferencing   |
| project management   |
| point of sale        |
| e-commerce           |
| inventory management |
| task management      |
| hospital management  |
+----------------------+
11 rows in set (0.00 sec)
```

28) Who are the female programmers earning MORE than the HIGEST paid male programmers?

```
mysql> SELECT name
    -> FROM programmer
    -> WHERE sex = 'F' AND salary > (
    ->     SELECT MAX(salary)
    ->     FROM programmer
    ->     WHERE sex = 'M'
    -> );
+--------+
| name   |
+--------+
| Amelia |
+--------+
1 row in set (0.00 sec)
```

29) Which language has been stated as prof1 by MOST of the programmers?

```
mysql> SELECT prof1
    -> FROM programmer
    -> GROUP BY prof1
    -> ORDER BY COUNT(*) DESC
    -> LIMIT 1;
+-------+
| prof1 |
+-------+
| Java  |
+-------+
1 row in set (0.00 sec)
```

30) Who are the authors of packages, WHICH have recovered MORE THAN double the development cost?

```
mysql> SELECT name
    -> FROM Software
    -> WHERE scost > (2 * dcost);
+-----------+
| name      |
+-----------+
| Abigail   |
| Alice     |
| Amelia    |
| Ava       |
| Bob       |
| Charlotte |
| Daniel    |
| David     |
| Emily     |
| Ethan     |
| Evelyn    |
| Harper    |
| Henry     |
| James     |
| John      |
| Liam      |
| Mia       |
| Michael   |
| Olivia    |
| Sophia    |
| William   |
+-----------+
21 rows in set (0.00 sec)
```

31) Display programmer names and CHEAPEST package developed by them in EACH language?

```
mysql> SELECT p.name AS programmer_name, s.dev_in AS language, s.title AS cheapest_package
    -> FROM programmer p
    -> INNER JOIN Software s ON p.name = s.name
    -> INNER JOIN (
    ->     SELECT dev_in, MIN(dcost) AS min_cost
    ->     FROM Software
    ->     GROUP BY dev_in
    -> ) AS min_costs ON s.dev_in = min_costs.dev_in AND s.dcost = min_costs.min_cost;
+-----------------+------------+---------------------+
| programmer_name | language   | cheapest_package    |
+-----------------+------------+---------------------+
| John            | Java       | e-commerce          |
| David           | C++        | e-learning platform |
| Olivia          | Python     | task management     |
| Charlotte       | JavaScript | document management |
+-----------------+------------+---------------------+
4 rows in set (0.07 sec)
```

32) Who is the YOUNGEST male programmer born in 1965?

```
mysql> SELECT name
    -> FROM programmer
    -> WHERE sex = 'M' AND dob = (
    ->     SELECT MIN(dob)
    ->     FROM programmer
    ->     WHERE sex = 'M' AND YEAR(dob) = 1965
    -> );
Empty set (0.07 sec)
```

33) Display language used by EACH programmer to develop the HIGEST selling and LOWEST selling package.

```
    -> ) AS min_package ON p.name = min_package.name;
+-----------------+--------------------------+-------------------------+
| programmer_name | highest_selling_language | lowest_selling_language |
+-----------------+--------------------------+-------------------------+
| John            | NULL                     | Java                    |
| Alice           | NULL                     | NULL                    |
| Bob             | NULL                     | NULL                    |
| Emily           | NULL                     | NULL                    |
| Michael         | NULL                     | NULL                    |
| Sophia          | NULL                     | NULL                    |
| David           | NULL                     | NULL                    |
| Emma            | JavaScript               | NULL                    |
| James           | NULL                     | NULL                    |
| Olivia          | NULL                     | NULL                    |
| William         | NULL                     | NULL                    |
| Ava             | NULL                     | NULL                    |
| Alexander       | NULL                     | NULL                    |
| Mia             | NULL                     | NULL                    |
| Ethan           | NULL                     | NULL                    |
| Charlotte       | NULL                     | NULL                    |
| Daniel          | NULL                     | NULL                    |
| Amelia          | NULL                     | NULL                    |
| Alexander       | NULL                     | NULL                    |
| Abigail         | NULL                     | NULL                    |
| Liam            | NULL                     | NULL                    |
| Harper          | NULL                     | NULL                    |
| Henry           | NULL                     | NULL                    |
| Evelyn          | NULL                     | NULL                    |
+-----------------+--------------------------+-------------------------+
24 rows in set (0.07 sec)
```

34) Who is the OLDEST female programmer WHO joined in 1992

```
mysql> SELECT name
    -> FROM programmer
    -> WHERE sex = 'F' AND doj = (
    ->     SELECT MIN(doj)
    ->     FROM programmer
    ->     WHERE sex = 'F' AND YEAR(doj) = 1992
    -> );
Empty set (0.00 sec)
```

35) In WHICH year where the MOST NUMBER of programmer born?

```
mysql> SELECT YEAR(dob) AS birth_year, COUNT(*) AS num_programmers_born
    -> FROM programmer
    -> GROUP BY YEAR(dob)
    -> ORDER BY num_programmers_born DESC
    -> LIMIT 1;
+------------+----------------------+
| birth_year | num_programmers_born |
+------------+----------------------+
|       1990 |                    2 |
+------------+----------------------+
1 row in set (0.00 sec)
```

36) In WHICH month did MOST NUMBRER of programmer join?

```
mysql> SELECT MONTH(doj) AS join_month, COUNT(*) AS num_programmers_joined
    -> FROM programmer
    -> GROUP BY MONTH(doj)
    -> ORDER BY num_programmers_joined DESC
    -> LIMIT 1;
+------------+-----------------------+
| join_month | num_programmers_joined |
+------------+-----------------------+
|          8 |                     3 |
+------------+-----------------------+
1 row in set (0.00 sec)
```

37) In WHICH language are MOST of the programmer's proficient?

```
mysql> SELECT prof1 AS proficient_language, COUNT(*) AS num_programmers
    -> FROM programmer
    -> WHERE prof1 IS NOT NULL
    -> GROUP BY prof1
    -> ORDER BY num_programmers DESC
    -> LIMIT 1;
+--------------------+-----------------+
| proficient_language | num_programmers |
+--------------------+-----------------+
| Java               |               7 |
+--------------------+-----------------+
1 row in set (0.00 sec)
```

38) Who are the male programmers earning BELOW the AVERAGE salary of female programmers?

```
mysql> SELECT name
    -> FROM programmer
    -> WHERE sex = 'M' AND salary < (
    ->     SELECT AVG(salary)
    ->     FROM programmer
    ->     WHERE sex = 'F'
    -> );
+------+
| name |
+------+
| John |
| Bob  |
+------+
2 rows in set (0.00 sec)
```

QUERY - IV

1) Display the details of THOSE WHO are drawing the same salary.

```
+-----------+------------+------------+-----+------------+------------+--------+
| name      | dob        | doj        | sex | prof1      | prof2      | salary |
+-----------+------------+------------+-----+------------+------------+--------+
| Alice     | 1988-10-25 | 2012-04-10 | F   | C++        | JavaScript |  55000 |
| Emily     | 1992-07-12 | 2014-09-15 | F   | Java       | C#         |  60000 |
| David     | 1991-04-18 | 2013-07-30 | M   | C++        | Python     |  59000 |
| James     | 1993-06-20 | 2016-03-25 | M   | Python     | Ruby       |  57000 |
| Alexander | 1996-09-03 | 2018-12-20 | M   | Python     | JavaScript |  59000 |
| Abigail   | 1995-09-20 | 2017-05-25 | F   | JavaScript | Python     |  55000 |
| Liam      | 1987-11-23 | 2010-09-28 | M   | C++        | Java       |  60000 |
| Henry     | 1993-08-30 | 2016-11-15 | M   | JavaScript | C++        |  57000 |
| Evelyn    | 1996-01-02 | 2018-04-05 | F   | Java       | Python     |  59000 |
+-----------+------------+------------+-----+------------+------------+--------+
9 rows in set (0.02 sec)
```

2) Display the details of software developed by male programmers earning MORE than 3000.

```
mysql> SELECT *
    -> FROM Software
    -> WHERE dev_in IN (
    ->     SELECT name
    ->     FROM Programmer
    ->     WHERE sex = 'male' AND salary > 3000
    -> );
Empty set (0.01 sec)
```

3) Display details of packages developed in PASCAL by female programmers.

```
mysql> SELECT *
    -> FROM Software
    -> WHERE dev_in = 'PASCAL' AND name IN (
    ->     SELECT name
    ->     FROM Programmer
    ->     WHERE sex = 'female'
    -> );
Empty set (0.01 sec)
```

4) Display the details of the programmer WHO joined BEFORE 1990.

```
mysql> SELECT *
    -> FROM Programmer
    -> WHERE doj < '1990-01-01';
Empty set (0.00 sec)
```

5) Display details of software developed in C by female programmers of PRAGATHI.

```
mysql> SELECT s.*
    -> FROM Software s
    -> JOIN Programmer p ON s.name = p.name
    -> JOIN Studies st ON p.name = st.name
    -> WHERE s.dev_in = 'C' AND p.sex = 'female' AND st.splace = 'PRAGATHI';
Empty set (0.00 sec)
```

6) Display NUMBER of packages NUMBER of copies sold and sales value of EACH programmer Institute-wise.

```
mysql> SELECT st.splace, p.name, COUNT(s.name) AS num_packages, SUM(s.sold) AS num_copies_sold, SUM(s.sold * s.scost) AS sales_value
    -> FROM Programmer p
    -> LEFT JOIN Software s ON p.name = s.name
    -> JOIN Studies st ON p.name = st.name
    -> GROUP BY st.splace, p.name;
Empty set (0.01 sec)
```

7) Display details of software developed in DBASE by male programmers WHO belong to the institute on which MOST NUMBER OF programmer's studies.

```
mysql> SELECT s.*
    -> FROM Software s
    -> JOIN Programmer p ON s.name = p.name
    -> JOIN Studies st ON p.name = st.name
    -> WHERE s.dev_in = 'DBASE' AND p.sex = 'male'
    -> AND st.splace = (
    ->     SELECT splace
    ->     FROM (
    ->         SELECT st2.splace, COUNT(*) AS num_programmers
    ->         FROM Programmer p2
    ->         JOIN Studies st2 ON p2.name = st2.name
    ->         GROUP BY st2.splace
    ->         ORDER BY COUNT(*) DESC
    ->         LIMIT 1
    ->     ) AS subquery
    -> );
Empty set (0.01 sec)
```

8) Display the details of the software that was developed by male programmers born BEFORE 1965 and female programmers born AFTER 1975.

```
mysql> SELECT s.*
    -> FROM Software s
    -> JOIN Programmer p ON s.name = p.name
    -> WHERE (p.sex = 'male' AND p.dob < '1965-01-01') OR (p.sex = 'female' AND p.dob > '1975-12-31');
Empty set (0.00 sec)
```

9) Display the details of the software that was developed in the language that is NOT the programmer's first proficiency.

```
mysql> SELECT s.*
    -> FROM Software s
    -> JOIN Programmer p ON s.name = p.name
    -> WHERE s.dev_in NOT IN (p.prof1, p.prof2);
+----------+----------------------+------------+----------+-------+------+
| name     | title                | dev_in     | scost    | dcost | sold |
+----------+----------------------+------------+----------+-------+------+
| Alice    | library management   | Python     | 15000.75 |  6000 |   75 |
| Emily    | social network       | JavaScript | 13000.80 |  6500 |   70 |
| Michael  | inventory management | Java       | 11000.60 |  5200 |   55 |
| James    | point of sale        | Java       | 11500.55 |  5700 |   68 |
| Olivia   | task management      | Python     | 14500.35 |  5400 |   63 |
| William  | ticketing system     | C++        | 12700.45 |  5900 |   72 |
| Ava      | inventory tracking   | JavaScript | 13700.65 |  5600 |   62 |
| Alexander| project management   | Java       | 11800.70 |  6200 |   78 |
| Mia      | expense tracking     | Python     | 14800.85 |  6100 |   74 |
| Charlotte| document management  | JavaScript | 13900.75 |  5400 |   69 |
| Evelyn   | e-commerce           | JavaScript | 14100.55 |  6100 |   75 |
+----------+----------------------+------------+----------+-------+------+
11 rows in set (0.00 sec)
```

10) Display details of software that was developed in the language which is NEITHER first

NOR second proficiency of the programmer.

```
mysql> SELECT s.*
    -> FROM Software s
    -> JOIN Programmer p ON s.name = p.name
    -> WHERE s.dev_in NOT IN (p.prof1, p.prof2)
    -> AND s.dev_in NOT IN (p.prof1, p.prof2);
+-----------+----------------------+------------+----------+-------+------+
| name      | title                | dev_in     | scost    | dcost | sold |
+-----------+----------------------+------------+----------+-------+------+
| Alice     | library management   | Python     | 15000.75 |  6000 |   75 |
| Emily     | social network       | JavaScript | 13000.80 |  6500 |   70 |
| Michael   | inventory management | Java       | 11000.60 |  5200 |   55 |
| James     | point of sale        | Java       | 11500.55 |  5700 |   68 |
| Olivia    | task management      | Python     | 14500.35 |  5400 |   63 |
| William   | ticketing system     | C++        | 12700.45 |  5900 |   72 |
| Ava       | inventory tracking   | JavaScript | 13700.65 |  5600 |   62 |
| Alexander | project management   | Java       | 11800.70 |  6200 |   78 |
| Mia       | expense tracking     | Python     | 14800.85 |  6100 |   74 |
| Charlotte | document management  | JavaScript | 13900.75 |  5400 |   69 |
| Evelyn    | e-commerce           | JavaScript | 14100.55 |  6100 |   75 |
+-----------+----------------------+------------+----------+-------+------+
11 rows in set (0.00 sec)
```

11) Display details of software developed by male students of SABHARI.

```
mysql> SELECT s.*
    -> FROM Software s
    -> JOIN Programmer p ON s.name = p.name
    -> JOIN Studies st ON p.name = st.name
    -> WHERE p.sex = 'male' AND st.splace = 'SABHARI';
Empty set (0.01 sec)
```

12) Display the names of programmers WHO HAVE NOT developed any package.

```
mysql> SELECT name
    -> FROM Programmer
    -> WHERE name NOT IN (SELECT name FROM Software);
Empty set (0.00 sec)
```

13) What is the total cost of the software developed by the programmers byAPPLE?

```
mysql> SELECT SUM(scost) AS total_cost
    -> FROM Software
    -> WHERE name IN (SELECT name FROM Programmer WHERE name LIKE '%APPLE%');
+------------+
| total_cost |
+------------+
|       NULL |
+------------+
1 row in set (0.00 sec)
```

14) Who are the programmers WHO JOINED in the same day?

```
mysql> SELECT name, doj
    -> FROM Programmer
    -> GROUP BY name, doj
    -> HAVING COUNT(*) > 1;
Empty set (0.00 sec)
```

15) Who are the programmers WHO HAVE THE SAME PROF2?

```
mysql> SELECT prof2
    -> FROM Programmer
    -> GROUP BY prof2
    -> HAVING COUNT(*) > 1;
+------------+
| prof2      |
+------------+
| Python     |
| JavaScript |
| NULL       |
| Java       |
| C++        |
+------------+
5 rows in set (0.00 sec)
```

16) Display the total sales values of software, institutes-wise.

```
mysql> SELECT st.splace, SUM(s.sold * s.scost) AS total_sales_value
    -> FROM Software s
    -> JOIN Programmer p ON s.name = p.name
    -> JOIN Studies st ON p.name = st.name
    -> GROUP BY st.splace;
Empty set (0.00 sec)
```

17) In which institutes did the person who developed the COSTLIEST package study?

```
mysql> SELECT st.splace
    -> FROM Software s
    -> JOIN Programmer p ON s.name = p.name
    -> JOIN Studies st ON p.name = st.name
    -> WHERE scost = (SELECT MAX(scost) FROM Software);
Empty set (0.01 sec)
```

18) Which language listed in prof1 and prof2 HAS NOT BEEN used to develop any package?

```
mysql> SELECT prof
    -> FROM (
    ->     SELECT prof1 AS prof FROM Programmer
    ->     UNION
    ->     SELECT prof2 FROM Programmer
    -> ) AS languages
    -> WHERE prof NOT IN (SELECT dev_in FROM Software);
+------+
| prof |
+------+
| C#   |
| SQL  |
| Ruby |
+------+
3 rows in set (0.01 sec)
```

19) How much does the person WHO developed the HIGHEST selling package earn and WHAT course did he/she undergo?

```
mysql> SELECT p.salary, st.course
    -> FROM Programmer p
    -> JOIN Software s ON p.name = s.name
    -> JOIN Studies st ON p.name = st.name
    -> WHERE s.sold = (SELECT MAX(sold) FROM Software);
Empty set (0.00 sec)
```

20) How many months will it take for each programmer to recover the cost of the course underwent?

```
mysql> SELECT p.name, (st.ccost / p.salary) * 12 AS months_to_recover FROM Programmer p JOIN Studies st ON p.name = st.name;
Empty set (0.00 sec)
```

21) Which is the COSTLIEST package developed by a person with under 3 year's expenences?

```
mysql> SELECT * FROM Software WHERE dcost = (SELECT MAX(dcost) FROM Software WHERE name IN (SELECT name FROM Programmer WHERE TIMESTAMPDIFF(YEAR, doj, CURDATE()) < 3));
Empty set (0.00 sec)
```

22) What is the AVERAGE salary for those WHOSE software's sales value is more than 50,000?

```
mysql> SELECT AVG(p.salary) AS average_salary FROM Programmer p JOIN Software s ON p.name = s.name WHERE s.sold * s.scost > 50000;
+----------------+
| average_salary |
+----------------+
|     57958.3333 |
+----------------+
1 row in set (0.00 sec)
```

23) How many packages were developed by the students WHO studied in the institute that Charge the LOWEST course fee?

```
mysql> SELECT COUNT(*) AS num_packages FROM Software WHERE name IN (SELECT s.name FROM Studies s WHERE ccost = (SELECT MIN(ccost) FROM Studies));
+--------------+
| num_packages |
+--------------+
|            0 |
+--------------+
1 row in set (0.00 sec)
```

24) How many packages were developed by the person WHO developed the CHEAPEST package? Where did he\she study?

```
mysql> SELECT COUNT(*) AS num_packages, st.splace FROM Software s JOIN Programmer p ON s.name = p.name JOIN Studies st ON p.name = st.name WHERE dcost = (SELECT MIN(dcost) FROM Software) GROUP BY st.splace;
Empty set (0.00 sec)
```

25) How many packages were developed by female programmers earning MORE than the HIGHEST paid male programmer?

```
mysql> SELECT COUNT(*) AS num_packages FROM Software WHERE name IN (SELECT s.name FROM Software s JOIN Programmer p ON s.name = p.name WHERE p.sex = 'female' AND p.salary > (SELECT MAX(salary) FROM Programmer WH
ERE sex = 'male'));
+--------------+
| num_packages |
+--------------+
|            0 |
+--------------+
1 row in set (0.01 sec)
```

26) How many packages were developed by the MOST experienced programmers from BDPS?

27) List the programmers (from software table) and institutes they studied, including those WHO DIDN'T develop any package.

```
mysql> SELECT p.name, s.splace
    -> FROM Programmer p
    -> LEFT JOIN Studies s ON p.name = s.name;
+-----------+--------+
| name      | splace |
+-----------+--------+
| John      | NULL   |
| Alice     | NULL   |
| Bob       | NULL   |
| Emily     | NULL   |
| Michael   | NULL   |
| Sophia    | NULL   |
| David     | NULL   |
| Emma      | NULL   |
| James     | NULL   |
| Olivia    | NULL   |
| William   | NULL   |
| Ava       | NULL   |
| Alexander | NULL   |
| Mia       | NULL   |
| Ethan     | NULL   |
| Charlotte | NULL   |
| Daniel    | NULL   |
| Amelia    | NULL   |
| Alexander | NULL   |
| Abigail   | NULL   |
| Liam      | NULL   |
| Harper    | NULL   |
| Henry     | NULL   |
| Evelyn    | NULL   |
+-----------+--------+
24 rows in set (0.01 sec)
```

28) List each profit with the number of programmers having that prof1 and the number of packages developed in that prof1.

```
mysql> SELECT
    ->     prof1 AS proficiency,
    ->     COUNT(*) AS num_programmers,
    ->     (SELECT COUNT(*) FROM Software WHERE dev_in = prof1) AS num_packages
    -> FROM
    ->     Programmer
    -> GROUP BY
    ->     prof1;
+-------------+-----------------+--------------+
| proficiency | num_programmers | num_packages |
+-------------+-----------------+--------------+
| Java        |               7 |            6 |
| C++         |               4 |            5 |
| Python      |               6 |            6 |
| JavaScript  |               5 |            6 |
| C#          |               1 |            0 |
| SQL         |               1 |            0 |
+-------------+-----------------+--------------+
6 rows in set (0.00 sec)
```

29) List programmer names (from programmer table) and number of packages EACH developed.

```
mysql> SELECT
    ->     p.name,
    ->     COUNT(s.name) AS num_packages
    -> FROM
    ->     Programmer p
    -> LEFT JOIN
    ->     Software s ON p.name = s.name
    -> GROUP BY
    ->     p.name;
+-----------+--------------+
| name      | num_packages |
+-----------+--------------+
| John      |            1 |
| Alice     |            1 |
| Bob       |            1 |
| Emily     |            1 |
| Michael   |            1 |
| Sophia    |            1 |
| David     |            1 |
| Emma      |            1 |
| James     |            1 |
| Olivia    |            1 |
| William   |            1 |
| Ava       |            1 |
| Alexander |            2 |
| Mia       |            1 |
| Ethan     |            1 |
| Charlotte |            1 |
| Daniel    |            1 |
| Amelia    |            1 |
| Abigail   |            1 |
| Liam      |            1 |
| Harper    |            1 |
| Henry     |            1 |
| Evelyn    |            1 |
+-----------+--------------+
23 rows in set (0.00 sec)
```

30) List all the details of programmers who have done a course at S.S.I.L.

```
mysql> SELECT
    ->     p.*
    -> FROM
    ->     Programmer p
    -> JOIN
    ->     Studies s ON p.name = s.name
    -> WHERE
    ->     s.splace = 'S.S.I.L.';
Empty set (0.00 sec)
```