Answer to the question no: 1

* Here is the differences between primary key and composite primary keys in below:

|  |  |
| --- | --- |
| **Primary key** | **Composite primary key** |
| 1) A primary key basically is an identifier which is a column and it contains unique values of each record. | 1) A composite primary key is a key which is a combination of two different column which can identify the each values or record from the table. |
| 2) Using a primary key is pretty much easier and faster to identify any value from a table because of one single column can easily ideantify each values. | 2) On the contrary because of having multiple column there is a slower processing of query and identify the target value. Using this key is not easy also. |
| 3) A primary column must have to be unique and can not be null. | 3) Here we can use different kind of datatypes to combine a composite key. |
| 4) It can used only when a unique column can identify like have an ID column. In this case we use this key. | 4) Another hand we use this key when a single column unable to find the record because the column can contains the same thing multiple times. For ignoring that problem we use a combination where we use both key or column to rid of that issue. |
| 5) 4) Here is an example of Primary key:  CREATE TABLE Class (  Col\_id INT NOT NULL,  Col2 teacher VARCHAR(40),,  Col3 section VARCHAR(40),,  PRIMARY KEY(Col\_id)  ) | 5)Here is an example of Composite primary key: Create TABLE Class (  Col\_1 student VARCHAR(40),  Col\_2 teacher VARCHAR(40),  Col\_3 section VARCHAR(40),  PRIMARY KEY(Col\_1, Col\_2)  ) |

Answer to the question no: 2

Here is the differences using join query and not using join query in below:

|  |  |
| --- | --- |
| **Using Join Query** | **Not Using Join Query** |
| 1) Join Query basically combine related data from different table. | 1) Not using Join query keeps data same without combining them. |
| 2) For using join query there is a good structure of data organization and a very good reduce of less important data. | 2) In this case there is a possibilities of data duplication problem. |
| 3) If data is bigger, it is very easy to work with because of combining multiple tables. | 3) Here data is messy and complex to work with if there is more data. |
| 4) We can use this operation for complex data. | 4) Without using join query data always independent and simple also. |
| 5)There is different kinds of join such as right join, left join, inner join etc which control data combination for matching or non-matching data. | 5) Without this there is a lack of efficiency to working with data from different table. |

Answer to the question no: 3

CREATE TABLE Employees (

Firstname VARCHAR(100) NOT NULL,

Lastname VARCHAR(50) NOT NULL,

Dateofbirth Date,

Departmentid INT,

Salary INT,

FOREIGN KEY (Departmentid) REFERENCES Departments(Departmentid)

)

CREATE TABLE Departments (

Departmentid INT PRIMARY KEY AUTO\_INCREMENT,

Departmentname VARCHAR(100) NOT NULL

)

Answer to the question no: 4

SELECT DISTINCT salary

FROM employees

ORDER BY salary DESC

LIMIT 1 OFFSET 1;

Answer to the question no: 5

SELECT department\_id, AVG(salary) AS average\_salary

FROM employees

GROUP BY department\_id;

Answer to the question no: 6

HERE I am providing inner join, left join, right join and self join in below accordingly:

**Inner join:**

SELECT departments.department\_name, employees.last\_name, employees.first\_name

FROM employees

INNER JOIN departments ON employees.department\_id = departments.department\_id;

**Left Join:**

SELECT departments.department\_name, employees.first\_name, employees.last\_name

FROM departments

LEFT JOIN employees ON departments.department\_id = employees.department\_id;

**Right Join:**

SELECT employees.first\_name, employees.last\_name, job\_history.start\_date

FROM employees

RIGHT JOIN job\_history ON employees.employee\_id = job\_history.employee\_id;

**Self Join:**

SELECT e1.first\_name, e1.last\_name, e2.first\_name AS manager\_firstname, e2.last\_name AS manager\_lastname

FROM employees e1

LEFT JOIN employees e2 ON e1.manager\_id = e2.employee\_id;

Answer to the question no: 8

SELECT e1.employee\_name

FROM employees e1

JOIN employees e2 ON e1.salary < e2.salary

WHERE e2.employee\_name = ‘Steven’;

Answer to the question no: 9

SELECT job\_id, COUNT(\*) as emplo\_num

FROM employees

GROUP BY job\_id;

Answer to the question no: 10

SELECT department\_name

FROM departments

WHERE department\_id NOT IN (SELECT DISTINCT department\_id FROM employees);