

# Aug 26, 2022 Day 1

## SOLUTIONS TO FIRST EXAMPLE

```
create table ElectrifyElectronics(Pid varchar(20) PRIMARY KEY, Pname varchar(30), Price number(6));
```

```
desc ElectrifyElectronics;
```

```
insert into ElectrifyElectronics values('P1', 'Earphone', 500);
insert into ElectrifyElectronics values('P2', 'Mobile', 12000);
insert into ElectrifyElectronics values('P3', 'USB Disk Storage', 400);
insert into ElectrifyElectronics values('P4', 'Airpods', 2500);
insert into ElectrifyElectronics values('P5', 'Smart watch', 4500);
```

```
select * from ElectrifyElectronics;
```

```
select Pname from ElectrifyElectronics;
```

```
select Pname, Price from ElectrifyElectronics;
```

```
select Pid, Pname from ElectrifyElectronics;
```

```
select Pname from ElectrifyElectronics where price > 1000;
```

```
select pName from electrifyelectronics where price < 1000;
```

```
select pName from electrifyelectronics where price between 2000 and 4000;
```

```
select pName from electrifyelectronics where price>=2000 and price<=4000;
```

```
select pName from electrifyelectronics where price>=3000;
```

```
select pName from electrifyelectronics where price<=5000;
```

## Important Points

- A collection of one or more attributes used to identify every record in a table uniquely is known as a key. Example: The AADHAAR number can be used to identify every citizen of India uniquely when their details are stored in a table.
- Whenever there exists two or more keys, the collection of these keys is referred to as candidate keys. Example: Let us assume that there exists a table with the schema:

Employee (Eid, Ename, Salary, Address, AADHAAR\_ID, PAN\_ID, DeptNo), where Eid is the unique employee ID, Ename refers to the employee name, AADHAAR\_ID refers to the unique AADHAAR number allotted to every citizen of India, PAN\_ID refers to the unique Permanent Account Number allotted to every bank account holder of India and DeptNo is the department ID for which the employee is working. Using only the attribute Eid, every record of the Employee table can be uniquely identified. Similarly, using only AADHAAR\_ID and only PAN\_ID, every record of the Employee table can be uniquely identified as well. Thus, there are 3 keys for the Employee table. These are called candidate keys.

- Out of all the candidate keys, only one key is chosen as the primary key. This decision is taken by the database designer, depending on prior experience and intuition. However, one thumb rule for selecting a primary key from a list of candidate keys is to choose the key with fewer number of attributes. Example: Assuming there is a table TAB1 with the schema TAB1(A1, A2, A3, A4, A5, A6). TAB1 has the following candidate keys: {A1, A4A5}, i.e., using only attribute A1, every record of TAB1 can be uniquely identified. Similarly, using attributes A4 and A5 together, every record of TAB1 can also be uniquely identified. The database designer will choose A1 as the primary key because it has only one attribute, as compared to A4A5, which has two attributes.
- Foreign key is a collection of one or more attributes in a table that refer to the primary key of another table or the same table. The foreign key is used to link tables. Example: There exists two tables Employee (Eid, Ename, DNo) and Department (DNo, DName). The DNo attribute of the Employee table depends on the DNo attribute of the Department table for its existence. Thus, the DNo attribute of the Employee table is a foreign key.
- Schema is the structure of a table which represents the overall design of the table. Examples: TAB1(A1, A2, A3, A4, A5, A6), Employee ( Eid, Ename, DNo), Department (DNo, DName) and Employee (Eid, Ename, Salary, Address, AADHAAR\_ID, PAN\_ID, DeptNo) are examples of schema. The underlined attribute is the primary key.
- Instance refers to the collection of values in the table at a particular instant of time. Example: The contents of the Product table is as shown below. It represents the instance of the table.

<u>Pid</u>	Pname	Price
P1	Mixer	2500
P2	Grinder	1800
P3	Oven	5000

- Schema changes infrequently whereas instance changes rapidly. Example: The design of the Product table (name of table, number of attributes, names of attributes, etc.) rarely changes, but the contents (new products can be added, existing products may be checked out) of the table will change frequently.
- SQL is a fourth generation programming language (4GL) which is used to handle databases and its components. It stands for Structured Query Language.

- SQL is non-procedural, i.e. There is a need to specify what is required without specifying how to get it. This is different from procedural programming languages like Python, where there is a need to specify what is required and how to get it.
- Names of tables, names of attributes of tables, clauses (also known as commands; such as select, create table, insert into, etc.), data types (varchar, number, etc.) are case-insensitive, i.e., SELECT is same as select or Select or any other variation one could think of
- Values entered into a table are case sensitive
- Text values entered must be enclosed within single quotes only
- Names of attributes, names of tables should not contain any intermediate spaces. If any name is made up of multiple words, the underscore symbol can be used to separate them. For example, there is an attribute Date of birth. In SQL, this can be represented as DateOfBirth or Date\_Of\_Birth or any other suitable representation which does not contain intermediate spaces. However, Date Of Birth is not allowed because of the presence of spaces in between
- Names of tables and attributes must not be the same as any of the SQL clauses. For example, creating a table called SELECT is disallowed because SELECT is a keyword
- In order to specify an attribute at the time of table creation, 3 things need to be mentioned: name of the attribute, data type of the attribute (the type of data that the attribute can hold) and size of the attribute (size must be in parentheses)
- In order to represent whole numbers, the data type used is number
- In order to represent text, the data type used is varchar or varchar2
- [Difference between VARCHAR and VARCHAR2](#)
- Components of SQL are:
  - Data Definition Language (DDL) to handle the schema of a table
  - Data Manipulation Language (DML) to handle instance of a table
  - Data Control Language (DCL) to control access to various parts of a database
  - Transaction Control Language (TCL) to handle transactions (a bit advanced for now!)

## Sep 2, 2022 Day 2

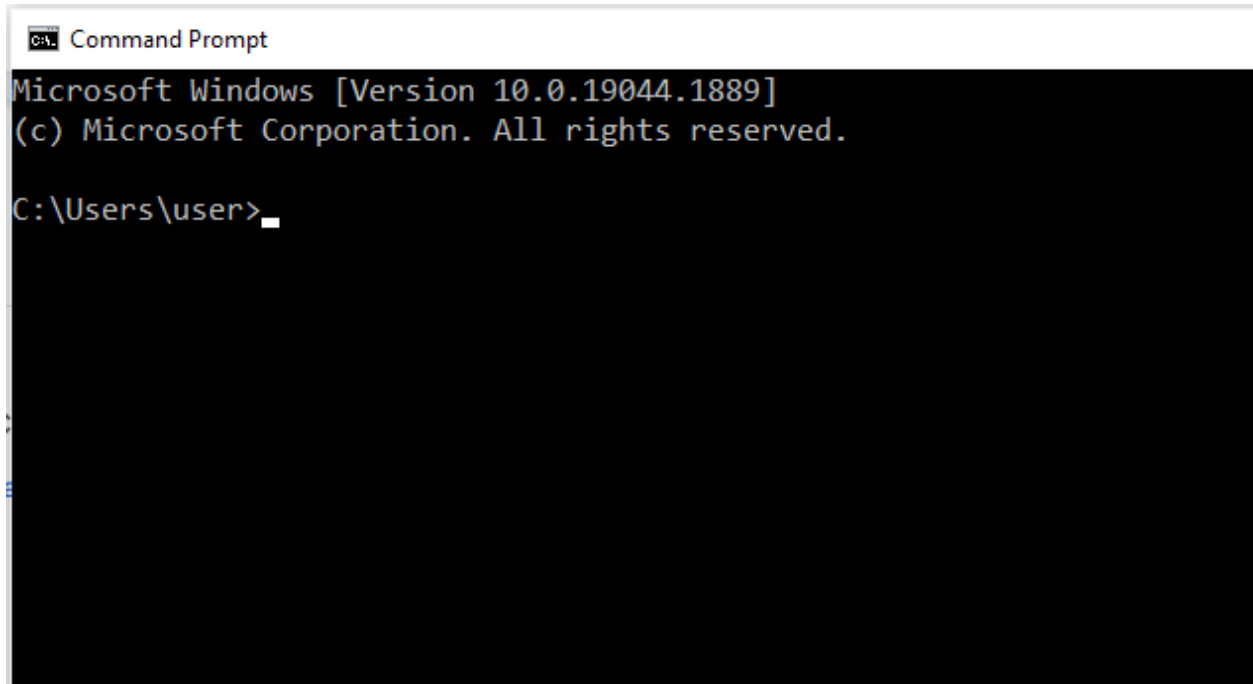
Please go through the following link to download the required software. Even if you are able to follow the instructions and proceed till 6:50 of the video, you can run Oracle using the following steps:

[Instructions to download and install Oracle 11g](#)

Assuming you have gone through the video till 6 minutes 50 seconds, the following steps will help you to use Oracle:

**Step 1:** Click on the start button

**Step 2:** Search for command prompt. A black screen like the one shown below shows up:

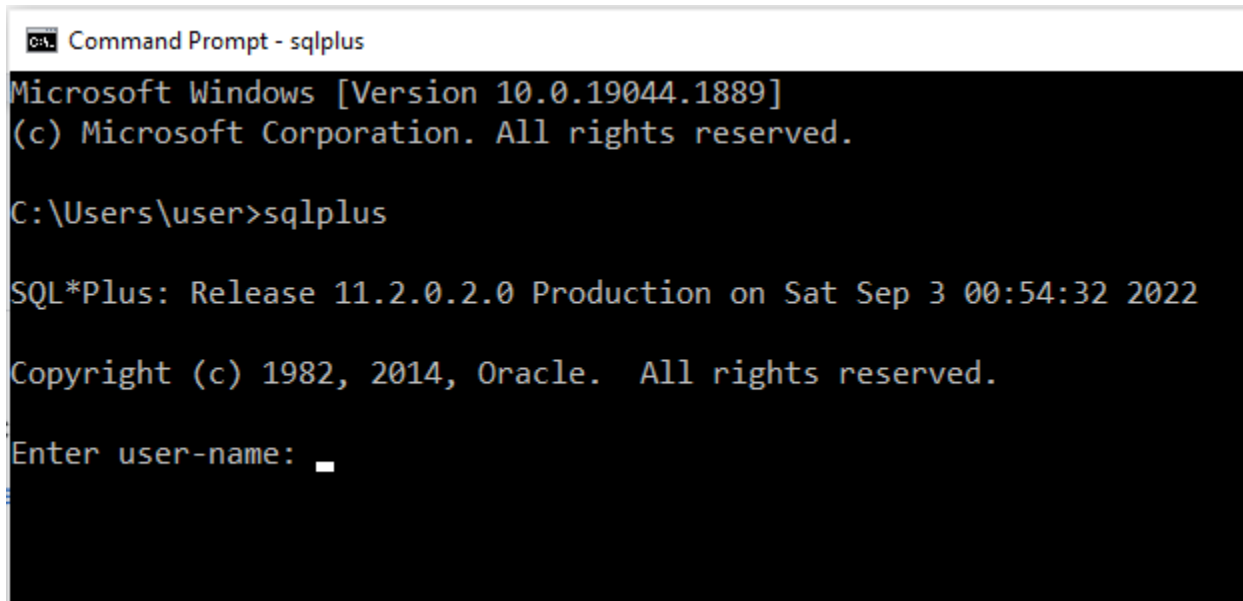


```
C:\> Command Prompt

Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>
```

**Step 3:** Type sqlplus. You should see an image like the one shown below:



```
C:\> Command Prompt - sqlplus

Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>sqlplus

SQL*Plus: Release 11.2.0.2.0 Production on Sat Sep 3 00:54:32 2022

Copyright (c) 1982, 2014, Oracle. All rights reserved.

Enter user-name: 
```

**Step 4:** Type the username as shown in the screenshot below and press the enter key:

```
C:\ Command Prompt - sqlplus

Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>sqlplus

SQL*Plus: Release 11.2.0.2.0 Production on Sat Sep 3 00:54:32 2022

Copyright (c) 1982, 2014, Oracle. All rights reserved.

Enter user-name: / as sysdba

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> _
```

**Step 5:** Now you can start executing all your SQL codes. The one shown below is from today's class:

```
SQL> create table Department(Did varchar(10) PRIMARY KEY, Dname varchar(30) NOT NULL CHECK(Dname IN ('Development', 'Analytics', 'R&D', 'Tech Support')), Dloc varchar(40) NOT NULL, DateOfSetup DATE NOT NULL);
Enter value for d:
```

**Step 6:** The same problem that we experienced from today's class arises. This snag can be solved using the following command. But before you type this new command, you need to exit this existing create table command using Ctrl + C. You will have to reconnect Steps 3 and 4 as shown above.

On pressing **Ctrl + C** (exit the current command), we get the following message:

```
SQL> create table Department(Did varchar(10) PRIMARY KEY, Dname varchar(30) NOT NULL CHECK(Dname IN ('Development', 'Analytics', 'R&D', 'Tech Support')), Dloc varchar(40) NOT NULL, DateOfSetup DATE NOT NULL);
Enter value for d:
SP2-0546: User requested Interrupt or EOF detected.

SQL> Disconnected from Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

C:\Users\user>
```

Use the SQL statement **set define off;** (as shown below) to avoid the technical snag shown in Step 5

```
Enter user-name: / as sysdba

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> set define off;
SQL>
```

**Step 7:** Now, we can start using SQL codes the way we want to:

```
SQL> set define off;
SQL> create table Department(Did varchar(10) PRIMARY KEY, Dname varchar(30) NOT NULL CHECK(Dname IN ('Development', 'Analytics', 'R&D', 'Tech Support')), Dloc varchar(40) NOT NULL, DateOfSetup DATE NOT NULL);

Table created.

SQL>
```

All SQL statements will work seamlessly now onward. If you want to exit from SQLPLUS, do the following:

```
Command Prompt

SQL> exit
Disconnected from Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

C:\Users\user>
```

## SOLUTIONS TO SECOND EXAMPLE

Q: Display the locations of Analytics departments

S: select Dloc from Department where Dname='Analytics';

Q: List the location and date of establishment of technical support department

S: select Dname, Dloc, Dateofsetup from Department where Dname='Tech Support';

Q: List the departments established in 2020

S: select \* from Department where Dateofsetup>='01-JAN-2020' AND Dateofsetup<='31-DEC-2020';

Explanation: A particular year ranges from 1st Jan to 31st Dec

Q: List the departments established in first quarter of 2020

S: select \* from Department where Dateofsetup>='01-JAN-2020' AND Dateofsetup<='30-APR-2020';

Explanation: First quarter ranges from 1st Jan to 30th Apr

Q: List the departments established in Dec. 2021

S: select \* from Department where Dateofsetup between '01-DEC-2021' AND '31-DEC-2021';

Explanation: Similar to relational operators >= and <=, we can also use the BETWEEN clause for checking range of dates

A domain for an attribute is chosen. Every value must follow the domain. Ex: for student names, the domain may be defined as the set of all characters of variable length.

*Domain constraints* are enforced with the help of data types. Ex: The DOB attribute in the employee table is defined to have a date data type. This implies, it must accept only date values in the prescribed format.

*Entity integrity constraint* refers to the restriction that primary key attribute(s) cannot accept NULL values. This has to be enforced because the primary key is used to uniquely identify every record in a table. If NULLs are allowed, these will repeat, and the ability of the primary key to uniquely identify every row will be lost.

*Referential integrity constraint* refers to the restriction that a foreign key attribute depends on a primary key attribute for its existence. Ex: If the department number attribute of the Employee table is a foreign key which depends on the department number attribute of the Department table, the department number attribute of the Employee table will only accept those values which are permissible in the department number attribute of the Department table. If any other value is inserted, it will lead to an error. Also, the data type of the attributes must exactly be the same. The referential integrity constraint may be present across two tables or in the same table.

*Semantic constraints* refer to restrictions imposed on the range of values that an attribute can accept. Ex: Restricting the salary attribute of the employee table to accept values only in the range [25000, 50000] / Restricting the department locations to a given set of values.

*NOT NULL constraint* refers to the restriction that an attribute cannot accept NULL values. Utility of NULLs can be understood in the following two situations:

1. Value is unknown. Ex: Employee table has a department number attribute. A newly joined employee may not have been allocated to any department. In such a case, the department number attribute for this particular employee may be represented using NULL

2. Value is not applicable. Ex: Employee table has a middle name attribute. Those people who do not have any middle name will have the NULL value for the middle name attribute

*UNIQUE constraint* enforces the restriction that an attribute will take different values for every record. However, an attribute having the UNIQUE constraint will be able to accept NULLs as well.

*DEFAULT constraint* enforces the restriction that an attribute will have a starting value. This will be used in case no value is provided for that attribute for a particular record.

All the aforementioned constraints can be applied either at the column level (right next to the definition of an attribute) or at the table level (after all the attributes of a table have been defined). However, the NOT NULL constraint has to be defined at the column level at all times, and the composite key constraint has to be defined at the table level at all times.

#### *Primary key v/s foreign key*

- A table will always have a single primary key but may have multiple foreign keys
- Primary key cannot accept NULLs, but foreign key(s) can
- Values for a primary key attribute will always be unique, whereas foreign key may accept repetitive values

#### *Primary key v/s UNIQUE*

- Primary key attribute cannot accept NULL values but UNIQUE attribute can

### **Practice Problems: SET 1**

Create an **Employee** table with the following properties:

Eid (data type = text, primary key),

Ename (data type = text, should not be unknown),

Salary (data type = integer, should be at least 25000 and at most 90000),

DOB (data type = date, should not be unknown),

Pid (data type = text, foreign key refers to Pid of Project table),

SuperNum (data type = text, foreign key refers to Eid of Employee table)

Create a **Project** table with the following properties:

Pid (data type = text; primary key),

Pname (data type = text; default value is "STARTUP"),

Plocation (data type = text; should not be unknown; list of possible values include Kolkata,

Pune, Delhi and Chennai),

Budget (data type = integer, should be at most 500000)

Meaning of each of the attributes:

Eid: Employee ID which is unique for each employee

Ename: Name of employee

Salary: Salary of the employee



DOB: Date of Birth

Pid: Project ID to which the employee has been allocated

SuperNum: Employee ID of the supervisor under which the employee is functioning

Pid: Unique project ID which is ongoing at the organization

Pname: Name of the project

Plocation: Location at which the project is being carried out

Budget: Amount of funds allocated for the project

Multiple employees can work under the same supervisor

A supervisor will not be working under anyone

Multiple employees can work for the same project

Insert the following 6 records into the Project table

('P1', 'Kolkata', 300000)

('P2', 'IT', 'Chennai', 400000)

('P3', 'R&D', 'Delhi', 350000)

('P4', 'R&D', 'Delhi', 250000)

('P5', 'Fintech', 'Pune', 480000)

('P6', 'Big Data', 'Pune', 320000)

Insert the following 10 records into the Employee table

('E1', 'Rajesh', 25000, 23<sup>rd</sup> June 1991, 'P1')

('E2', 'Rakesh', 52000, 4<sup>th</sup> January 1993, 'P2', 'E1')

('E3', 'Sumit', 45000, 7<sup>th</sup> September 1991, 'P6', 'E1')

('E4', 'Sharon', 65000, 1<sup>st</sup> April 1992, 'P3')

('E5', 'Kavya', 62000, 23<sup>rd</sup> August 1995, 'P4', 'E4')

('E6', 'Kriti', 35000, 3<sup>rd</sup> June 1990, 'P2')

('E7', 'Ankush', 52000, 13<sup>th</sup> February 1996, 'P5', 'E6')

('E8', 'Sameer', 45000, 12<sup>th</sup> Nov 1993, 'P6', 'E6')

('E9', 'Nadeem', 56000, 10<sup>th</sup> June 1993, 'P3', 'E1')

('E10', 'Shruti', 82000, 30<sup>th</sup> July 1994, 'P5', 'E4')

Write SQL statements to solve the following queries:

Q1 Display the instance of the Project table

Q2 Display the schema of the Project table

Q3 Display the instance of the Employee table

Q4 Display the schema of the Employee table

Q5 Find the names and DOBs of employees receiving salary between 50000 and 60000

- Q6 Find the names and salaries of employees working on project P5  
Q7 Find the names of employees born in 1993  
Q8 Display details of projects for which the budget allocation is at least 400000  
Q9 List out the names of all the projects  
Q10 List out details of projects being carried out at Kolkata  
Q11 List out project details being carried out at Pune and having a budget at least 300000

## Sep 9, 2022 Day 3

### SOLUTIONS TO THIRD EXAMPLE

```
create table Department(Did varchar(10) PRIMARY KEY, Dname varchar(30) NOT NULL
CHECK(Dname IN ('Development', 'Analytics', 'R&D', 'Tech Support')), Dloc varchar(40) NOT
NULL, DateOfSetup DATE NOT NULL);
```

```
create table Employee(Eid varchar(10), Ename varchar(50) NOT NULL, Salary number(8)
DEFAULT 30000, DOB date NOT NULL, Did varchar(5) references Department(Did), SuperNum
varchar(10) references Employee(Eid), PRIMARY KEY(Eid));
```

#### Important points:

- The DEFAULT constraint is added after specifying the attribute details such as attribute name, attribute data type, size of attribute. It involves using the DEFAULT keyword followed by the default value. In the above CREATE TABLE statement, the default salary is 30000. As 30000 is a numeric value, it is specified as it is after the DEFAULT keyword. Had the default value been a string, it would have to be included in single quotes
- The PRIMARY KEY constraint is added at the table level as shown above. Table level constraints are always added at the end of the CREATE TABLE statement. Had the PRIMARY KEY been made up of two attributes, say A1 and A2, the constraint would have been specified as **PRIMARY KEY (A1, A2)**

```
insert into Department values('D1', 'Analytics', 'Kolkata', '23-JUN-2020');
```

```
insert into Department values('D2', 'Analytics', 'Chennai', '22-JUN-2020');
```

```
insert into Department values('D3', 'Development', 'Bengaluru', '19-DEC-2020');
```

```
insert into Department values('D4', 'Development', 'Mumbai', '15-NOV-2020');
```

```
insert into Department values('D5', 'Tech Support', 'Silchar', '1-DEC-2021');
```

```
insert into Department values('D6', 'Tech Support', 'Noida', '03-APR-2021');
```

```
insert into employee(Eid, Ename, Salary, DOB, Did) values('E1', 'ABC', 75000, '23-JUN-1991',
'D1');
```

### Important points:

- The salary attribute of the Employee table has the default constraint. If the INSERT INTO contains a value for the salary column, it will override the default value
- The SuperNum attribute specifies the employee ID of the supervisor. If an employee is a supervisor herself, she will not have any value for the SuperNum attribute. In this case, the SuperNum attribute will have a NULL value.
- These two points are highlighted by the above INSERT statement

```
insert into employee(Eid, Ename, DOB, Did, SuperNum) values ('E2', 'DEF', '04-JAN-1993', 'D1', 'E1');
```

```
insert into employee(Eid, Ename, DOB, Did, SuperNum) values ('E3', 'GHI', '07-SEP-1991', 'D6', 'E1');
```

### Important points:

- The above two INSERT INTO statements do not specify any value for the salary attribute. Thus, the default value of 30000 is applied for both the records

```
insert into employee(Eid, Ename, Salary, DOB, Did, SuperNum) values('E4', 'JKL', 65000, '01-APR-1991', 'D4', 'E1');
```

```
insert into employee(Eid, Ename, Salary, DOB, Did, SuperNum) values('E5', 'JKL', 62000, '23-AUG-1995', 'D4', 'E1');
```

```
insert into employee(Eid, Ename, Salary, DOB, Did) values('E6', 'ABC', 35000, '3-JUN-1990', 'D2');
```

```
insert into employee(Eid, Ename, Salary, DOB, Did, SuperNum) values('E7', 'PQR', 52000, '13-FEB-1996', 'D3', 'E6');
```

```
insert into employee(Eid, Ename, Salary, DOB, Did, SuperNum) values('E8', 'PQR', 45000, '12-NOV-1993', 'D4', 'E6');
```

```
insert into employee(Eid, Ename, Salary, DOB, Did, SuperNum) values('E9', 'XYZ', 56000, '10-JUN-1993', 'D1', 'E6');
```

```
insert into employee(Eid, Ename, Salary, DOB, Did, SuperNum) values('E10', 'DEF', 62000, '30-JUL-1994', 'D5', 'E6');
```

*Q Display the schema and instance of both tables*

```
SELECT * FROM Employee;
```

```
SELECT * FROM Department;
```

```
DESC Department;
```

```
DESCRIBE Department;
```

*Q Find the names of departments being run*

```
select distinct dname from department;
```

### Important points:

- The SELECT DISTINCT clause will only list out the unique department names which is required

*Q Find the details of supervisors*

```
select * from employee where SuperNum IS NULL;
```

### Important points:

- A supervisor is an employee who does not work under any other employee. As a result, the SuperNum attribute for such employees will be NULL. In order to check whether an employee is a supervisor or not, checking the SuperNum attribute is NULL or not will work. However, checking for NULL value is carried out using **IS NULL**, and not the = operator

*Q Find the details of employees who are not supervising anyone*

```
select * from employee where SuperNum IS NOT NULL;
```

### Important points:

- Employees who are not supervising anyone do not have NULL values for the SuperNum attribute
- Checking the SuperNum attribute for non null value will work
- This is carried out using IS NOT NULL

*Q Display the name and annual salary of the employees*

```
select Ename, Salary*12 AS "Annual Salary"  
FROM Employee;
```

### Important points:

- Annual Salary is not present as a separate attribute
- To find the annual salary, a temporary column may be generated using the AS clause
- If the AS clause is not provided, Salary\*12 becomes the name of the column which does not look proper
- In order to provide a neat representation of the output, the AS clause is used to create a temporary column with a meaningful name
- This new temporary name must be enclosed within double quotation and not single quotes
- Single quotes are used only for values
- The table contains the monthly salary values which are multiplied by 12 in order to generate the annual salary values
- The \* operator is used to perform multiplication

*Q Increase the salary of supervisors by 20%*

update employee set salary = 1.2 \* salary  
where SuperNum IS NULL;

**Important points:**

- Increasing salary by 20% is same as multiplying by 1.2

*Q Add a column called commission to the Employee table having float data type*

ALTER TABLE Employee ADD Commission NUMBER(8,2);

*Q Add a column called Email in the Department table*

ALTER TABLE Department ADD email char(255);

*Q Modify data type of email to varchar*

ALTER TABLE Department

MODIFY email varchar(255);

**Important points:**

- When to use ALTER TABLE?
  - To add a new column to an existing table
  - To remove an existing column from an existing table
  - To change the data type of an attribute of an existing table
  - To change the size of the data type of an attribute of an existing table
  - To add a new constraint to an attribute of a table
  - To remove an existing constraint from an attribute from a table
  - To modify a constraint of an attribute of a table
  - To change the names of tables and/or attributes of a table

*Q Assign a starting commission of 1000 to all employees who are not supervisors*

UPDATE Employee set commission = 1000  
where SuperNum IS NOT NULL;

## Sep 16, 2022 Day 4

*Q Add a column called Email in the Department table*

alter table department add email char(255);

**Important point(s):**

- The keyword for adding a new column is **add**
- Need to mention column name, data type and size while adding new columns
- The char data type is also used to store text values
- However, the space allotted using char is fixed but space allotted using varchar is

variable. For example, if the size of a column is 255 with char data type, 255 units of space will always be used regardless of the size of the value inserted. With varchar, the size will vary depending on the value inserted. If the length of the value is 25, only 25 units of space will be allocated with varchar, but 255 units will be allocated with char.

*Q Modify data type of email to varchar*

alter table department modify email varchar(255);

**Important point(s):**

- The keyword for changing the data type is **modify**
- Need to mention column name, data type and size while modifying data type of an existing column

*Q Remove the email column from the department table*

alter table department drop column email;

**Important point(s):**

- The keyword for removing a column is **drop column**
- Need to mention column name while removing a column

*Q Change name of DLOC to DEPARTMENT\_LOCATION*

alter table Department RENAME COLUMN DLOC to DEPARTMENT\_LOCATION;

**Important point(s):**

- The keyword for renaming a column is **rename column**
- Need to mention: **current column name TO new column name**

*Q Delete the details of non-supervisors*

DELETE FROM Employee where SuperNum IS NOT NULL;

**Important point(s):**

- Deletion takes place row-by-row
- A non-supervisor is supervised by an employee and has the employee ID of the supervisor under the SuperNum attribute. So, the check must be done using IS NOT NULL

*Q Delete the details of supervisors*

DELETE FROM Employee where SuperNum IS NULL;

**Important point(s):**

- Deletion takes place row-by-row
- A supervisor is not supervised by an employee and has NULL value under the SuperNum attribute. So, the check must be done using IS NULL

*Q Delete the contents of the department table*

DELETE FROM Department;

**Important point(s):**

- All rows are deleted one by one and this takes up a considerable amount of time
- An alternative approach is to use TRUNCATE TABLE
- The TRUNCATE TABLE command works in two steps internally:
  - Step 1 = uses DROP TABLE to remove the table entirely
  - Step 2 = uses CREATE TABLE to create the table again
- This effectively empties the table and no rows are left, which is exactly the same as the DELETE command, but proceeds in a much faster way

*Q Remove both the tables, including the schema*

DROP TABLE Employee;

DROP TABLE Department;

**Important point(s):**

- The DROP TABLE command removes the instance as well as the schema of a table
- Need to remove the Employee table before removing the Department table because the Employee table has an attribute Did which depends on the Did attribute of the Department table. If the Department table is dropped before dropping the Employee table, the Did attribute in the Employee table would have nothing to depend on

**SOLUTIONS TO FOURTH EXAMPLE**

```
create table dept(deptno number(2,0), dname varchar2(14), loc varchar2(13), constraint  
pk_dept primary key (deptno));
```

```
create table emp(empno number(4,0), ename varchar2(10), job varchar2(9), mgr number(4,0),  
hiredate date, sal number(7,2), comm number(7,2), deptno number(2,0), constraint pk_emp  
primary key (empno), constraint fk_deptno foreign key (deptno) references dept (deptno));
```

```
insert into DEPT (DEPTNO, DNAME, LOC) values(10, 'ACCOUNTING', 'NEW YORK');  
insert into dept values(20, 'RESEARCH', 'DALLAS');  
insert into dept values(30, 'SALES', 'CHICAGO');  
insert into dept values(40, 'OPERATIONS', 'BOSTON');
```

```
insert into emp values(7839, 'KING', 'PRESIDENT', null, '17-NOV-1981', 5000, null, 10);  
insert into emp values(7698, 'BLAKE', 'MANAGER', 7839, '1-MAY-1981', 2850, null, 30);  
insert into emp values(7782, 'CLARK', 'MANAGER', 7839, '09-JUN-1981', 2450, null, 10);  
insert into emp values(7566, 'JONES', 'MANAGER', 7839, '2-APR-1981', 2975, null, 20);  
insert into emp values(7788, 'SCOTT', 'ANALYST', 7566, '13-JUL-1987', 3000, null, 20);  
insert into emp values(7902, 'FORD', 'ANALYST', 7566, '3-DEC-1981', 3000, null, 20);
```

```
insert into emp values(7369, 'SMITH', 'CLERK', 7902, '17-DEC-1980', 800, null, 20);
insert into emp values(7499, 'ALLEN', 'SALESMAN', 7698, '20-FEB-1981', 1600, 300, 30);
insert into emp values(7521, 'WARD', 'SALESMAN', 7698, '22-FEB-1981', 1250, 500, 30);
insert into emp values(7654, 'MARTIN', 'SALESMAN', 7698, '28-SEP-1981', 1250, 1400, 30);
insert into emp values(7844, 'TURNER', 'SALESMAN', 7698, '8-SEP-1981', 1500, 0, 30);
insert into emp values(7876, 'ADAMS', 'CLERK', 7788, '13-JUL-1987', 1100, null, 20);
insert into emp values(7900, 'JAMES', 'CLERK', 7698, '3-DEC-1981', 950, null, 30);
insert into emp values(7934, 'MILLER', 'CLERK', 7782, '23-JAN-1982', 1300, null, 10);
```

*Q List the employees whose salaries are 800, 1600 or 2450*

```
select ename from emp where sal in(800,1600,2450);
```

*Q List the employee names start with 'F'*

```
select ename from emp where ename like 'F%';
```

*Q List all employees whose names end with 'N'*

```
select ename from emp where ename like '%N';
```

*Q List all employees whose job does not start with "CL"*

```
select ename from emp where job not like 'CL%';
```

*Q Calculate the average salary of all employees*

```
select avg(sal) "AVERAGE SALARY" from emp;
```

*Q Calculate the total salary of all managers*

```
select sum(sal) "TOTAL SALARY OF MANAGERS" from emp where job='MANAGER';
```

*Q Find the minimum salary earned by the employees*

```
select min(sal) "MINIMUM SALARY" from emp;
```

*Q Find the maximum salaries earned by the employees*

```
select max(sal) "MAXIMUM SALARY" from emp;
```

*Q Find the minimum, maximum and average salary earned by the employees*

```
select min(sal) "MINIMUM SALARY", max(sal) "MAXIMUM SALARY", avg(sal) "AVERAGE SALARY" from emp;
```

## Sep 23, 2022 Day 5

*Q List the total number of employees and the average salaries of the different departments*



```
select deptno, count(ename) "NO OF EMPLOYEES", avg(sal) "AVERAGE SALARY" from emp
group by deptno;
```

*Q Calculate total number of managers*

```
select count(ename) "TOTAL NO OF MANAGERS" from emp where job='MANAGER';
```

*Q List the details of all managers in ascending order of joining dates*

```
select * from emp where job='MANAGER' order by hiredate;
```

*Q List the average salaries for each different job*

```
select job, avg(sal) "AVERAGE SALARY" from emp group by job;
```

*Q Find all departments which have less than 3 employees.*

```
select dname, count(ename)
from emp, dept
where emp.deptno=dept.deptno
group by dname
having count(ename) < 3;
```

*Q List the details of the employees in ascending order of department number and within each department in descending order of salary.*

```
select * from emp order by deptno asc, sal desc;
```

*Q Display the name of employee who earns maximum salary*

```
select ename, sal from emp where sal=(select max(sal) from emp);
```

*Q Display the name of employee who earns maximum salary and whose job is a salesman*

```
select ename, sal
from emp
where sal = ( select max(sal) from emp where job='SALESMAN' );
```

*Q List all employee names, department names and the cities, in order of department name*

```
select ename,dname,loc
from emp,dept
where emp.deptno=dept.deptno
order by dname;
```

*Q List all employee names, jobs, salaries and department names except clerks. Sort the report with respect to job and salary*

```
select ename,dname,job,sal
from emp,dept
where emp.deptno=dept.deptno and job<>'CLERK'
order by job,sal;
```

## Oct 28, 2022 Day 6

```
create table dept(deptno number(2,0), dname varchar2(14), loc varchar2(13), constraint  
pk_dept primary key (deptno));
```

```
create table emp(empno number(4,0), ename varchar2(10), job varchar2(9), mgr number(4,0),  
hiredate date, sal number(7,2), comm number(7,2), deptno number(2,0), constraint pk_emp  
primary key (empno), constraint fk_deptno foreign key (deptno) references dept (deptno));
```

```
insert into DEPT (DEPTNO, DNAME, LOC) values(10, 'ACCOUNTING', 'NEW YORK');  
insert into dept values(20, 'RESEARCH', 'DALLAS');  
insert into dept values(30, 'SALES', 'CHICAGO');  
insert into dept values(40, 'OPERATIONS', 'BOSTON');
```

```
insert into emp values(7839, 'KING', 'PRESIDENT', null, '17-NOV-1981', 5000, null, 10);  
insert into emp values(7698, 'BLAKE', 'MANAGER', 7839, '1-MAY-1981', 2850, null, 30);  
insert into emp values(7782, 'CLARK', 'MANAGER', 7839, '09-JUN-1981', 2450, null, 10);  
insert into emp values(7566, 'JONES', 'MANAGER', 7839, '2-APR-1981', 2975, null, 20);  
insert into emp values(7788, 'SCOTT', 'ANALYST', 7566, '13-JUL-1987', 3000, null, 20);  
insert into emp values(7902, 'FORD', 'ANALYST', 7566, '3-DEC-1981', 3000, null, 20);  
insert into emp values(7369, 'SMITH', 'CLERK', 7902, '17-DEC-1980', 800, null, 20);  
insert into emp values(7499, 'ALLEN', 'SALESMAN', 7698, '20-FEB-1981', 1600, 300, 30);  
insert into emp values(7521, 'WARD', 'SALESMAN', 7698, '22-FEB-1981', 1250, 500, 30);  
insert into emp values(7654, 'MARTIN', 'SALESMAN', 7698, '28-SEP-1981', 1250, 1400, 30);  
insert into emp values(7844, 'TURNER', 'SALESMAN', 7698, '8-SEP-1981', 1500, 0, 30);  
insert into emp values(7876, 'ADAMS', 'CLERK', 7788, '13-JUL-1987', 1100, null, 20);  
insert into emp values(7900, 'JAMES', 'CLERK', 7698, '3-DEC-1981', 950, null, 30);  
insert into emp values(7934, 'MILLER', 'CLERK', 7782, '23-JAN-1982', 1300, null, 10);
```

a) List name of the employee who earns the minimum salary

```
select ename  
from emp  
where sal=(select min(sal) from emp);
```

b)List all employees who work in the same post as Smith

```
select ename  
from emp  
where job=(select job  
            from emp  
            where ename='SMITH')  
and ename <>'SMITH';
```

c) List all employees who earn more than every employee in the 'Sales' department

```
select ename
from emp
where sal > (select max(sal)
            from emp, dept
            where emp.deptno = dept.deptno and dname = 'SALES');
```

d) Find the job with the highest average salary

```
select A.job
from (select job, avg(sal) as avgsal
      from emp
      group by job) A,
(select max(avg(sal)) as maxsal
 from emp group by job) B
where A.avgsal = B.maxsal;
```

e) Find the highest salary of each job

```
select job, Max(sal) as "HIGHEST SALARY"
from emp
group by job;
```

f) List the names, jobs and salaries of employees whose salary is greater than the highest salary in Research department

```
select ename, job, sal
from emp
where sal > (select max(sal)
            from emp, dept
            where emp.deptno = dept.deptno and dname = 'RESEARCH');
```

g) Find the department that is not having any employee

```
select dname
from dept
where not exists (select job from emp where emp.deptno = dept.deptno);
```

h) List the top 3 earners in the organization

```
select E.ename, E.sal
from (select *
```

```
from emp
order by sal desc)E
where rownum<4;
```

i) List the years and the number of people joining in that year

```
select to_char(hiredate,'YYYY') as "Year", count(ename) as "NUMBER OF EMPLOYEES"
from emp
group by to_char(hiredate,'YYYY');
```

### Important point(s):

#### Using the TO\_CHAR Function with Dates :

##### SYNTAX :

```
TO_CHAR(date, 'format_model')
```

The format model:

- Must be enclosed in single quotation marks and is case sensitive
- Can include any valid date format element
- Has an fm element to remove padded blanks or suppress leading zeros
- Is separated from the date value by a comma

#### EXAMPLE :

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired
FROM employees
WHERE last_name = 'Higgins';
```

#### OUTPUT :

EMPLOYEE_ID	MONTH_HIRED
205	06/94

#### Elements of the Date Format Model :

YYYY Full year in Numbers

YEAR Year spelled out

MM Two digit value for month

MONTH Full name of the month

MON Three Letter abbreviation of the month

DY Three letter abbreviation of the day of the week

DAY Full Name of the week

DD Numeric day of the month

j) Give an increment of 20% to all employees who have joined before 1/1/82 or earner less than Rs. 3000

```
select ename,sal,hiredate,sal*20/100 "INCREMENT"
```

```
from emp
where to_char(hiredate,'YY')<82 or sal<3000;
```

## Nov 4, 2022 Day 7

k) Display the name and salary of the employees who receive maximum and minimum salary in one row with proper heading

```
select A.ename NAME, A.sal "MAXIMUM SALARY", B.ename NAME, B.sal "MINIMUM
SALARY"
from emp A, emp B
where A.sal=(select max(sal) from emp) and B.sal=(select min(sal) from emp);
```

l) Find the job which exist in department no 30 but not in 10

```
select job from emp where deptno=30
minus
select job from emp where deptno=10;
```

m) Find the highest salary in each job type

```
select job,max(sal)"highest salary"
from emp
group by job;
```

n) Find the most recently hired employees in each department

```
select dname,ename,hiredate
from emp,dept
where dept.deptno=emp.deptno
and (dname,hiredate) in (select dname,max(hiredate)
                        from emp,dept
                        where dept.deptno=emp.deptno
                        group by dname);
```

o) Display the employee names getting salaries more than their managers

```
select employee from
                        (select a.ename as employee, a.sal as salary, b.ename as  manager,
b.sal as msalary from emp a, emp b where a.mgr=b.empno)e
                        where e.salary>e.msalary;
```

**Practice Exercise (continuation from the previous class): Please try to solve the following queries. I shall provide the solution in a couple of days.**

Create the following tables:

Borrower(Id: data type = varchar, size = 3, primary key  
Name: data type = varchar, size = 20)

Book(Id: data type = varchar, size = 3, primary key  
Title: data type = varchar, size = 20  
Author: data type = varchar, size = 20  
Subject: data type = varchar, size = 10)

Borrows(Book\_Id: data type = varchar, size = 3, key attribute, foreign key depends on Id attribute of Book table

B\_Id: data type = varchar, size = 3, key attribute, foreign key depends on Id attribute of Borrower table

Date\_Of\_Issue: data type = date

Date\_Of\_Return: data type = date)

The following records are to be inserted in the Borrower table:

ID NAME

-----

BR1 SOHINI RAI

BR2 SUMANA CHANDRA

BR3 KARAN DOSHI

BR4 SIDD MEHRA

BR5 TAMAL CHAUHAN

BR6 MOUMITA DAS

BR7 NEIL MORRIS

The following records are to be inserted in the Book table:

ID	TITLE	AUTHOR	SUBJECT
-----	-----	-----	-----
BK1	MODERN HISTORY 1	P.Mukherjee	HISTORY
BK2	SOCIAL ETHICS	G.Morrison	SOCIOLOGY
BK3	HUMAN NATURE	A.Das	PSYCHOLOGY
BK4	PHILOSOPHY VOLUME1	A.Das	PHILOSOPHY
BK5	SOCIAL SYSTEM	G.Morrison	SOCIOLOGY
BK6	HISTORY 2	A.Roy	HISTORY
BK7	GEOGRAPHY VOLUME 1	S.Bhattacharya	GEOGRAPHY
BK8	JULIUS CEASAR	W.Shakespeare	ENGLISH
BK9	GODAN	M.Premchand	HINDI

The following records are to be inserted in the Borrows table:

BOOK_Id	B_I	DATE_OF_I	DATE_OF_R
-----	---	-----	-----
BK1	BR3	01-SEP-2018	07-SEP-2018
BK2	BR3	01-SEP-2018	14-SEP-2018
BK1	BR1	07-SEP-2018	14-SEP-2018
BK6	BR1	19-NOV-2018	26-NOV-2018
BK5	BR2	20-NOV-2018	27-NOV-2018
BK4	BR5	26-NOV-2018	28-NOV-2018

a) Display the details of all the books on sociology and psychology

```
SELECT *  
FROM Book  
WHERE SUBJECT='SOCIOLOGY' OR SUBJECT='PSYCHOLOGY';
```

b) Find all the books written by A.Das on philosophy.

```
SELECT * FROM Book WHERE AUTHOR='A.Das' AND SUBJECT='PHILOSOPHY';
```



c) Find out the total number of books in the library.

```
SELECT COUNT(ID) FROM Book;
```

d) Find out the names of the borrowers who have borrowed at least one book on History.

```
SELECT NAME FROM Borrower
WHERE Id IN
(SELECT B_Id FROM Borrows WHERE Book_Id
IN(SELECT Id FROM Book WHERE SUBJECT='HISTORY'));
```

e) Display the names of the borrowers who have issued books written by G.Morrison.

```
SELECT NAME
FROM Borrower
WHERE Id IN(SELECT B_Id FROM Borrows WHERE Book_Id IN(SELECT Id FROM Book
WHERE Author='G.Morrison'));
```

f) Find out the titles and authors of all the books issued by the borrower whose id is 'BR3'.

```
SELECT Title,Author FROM Book
WHERE Id IN(SELECT Book_Id FROM Borrows WHERE B_Id='BR3');
```

[PL/SQL Code Walkthrough 1](#)

[PL/SQL Code Walkthrough 2](#)

[PL/SQL Code Walkthrough 3](#)

[PL/SQL Code Walkthrough 4](#)

[PL/SQL Code Walkthrough 5](#)

## Nov 11, 2022 Day 8

### Practice Exercise

Consider the following schema of a relational database:

Product(maker:char, model:number, type:char)

PC(model:number, speed:number, RAM:number, hdd:number, cd:number, price:number)

Laptop(model:number, speed:number, RAM:number, screen:number, price:number)

Printer(model:number, colour:char, type:char, price:number)

### **Constraints on Product table**

- Model is the primary key
- Type can only take the values PC, LAPTOP or PRINTER
- Maker cannot take empty values

### **Constraints on PC table**

- Model is the primary key
- Model is the foreign key which depends on the model attribute of Product
- Speed, RAM, HDD, Price cannot take blank values
- Speed attribute can take decimal values upto 5 places

### **Constraints on Laptop table**

- Model is the primary key
- Model is the foreign key which depends on the model attribute of Product
- Speed, RAM, Screen, Price cannot be left blank
- Speed attribute can take decimal values upto 2 places

### **Constraints on Printer table**

- Model is the primary key
- Model is the foreign key which depends on the model attribute of Product
- Colour, Type, Price cannot be left blank

SQL> SELECT \* FROM Product;

MAKER    MODEL TYPE

```
-----
samsung      1      PC
dell         2      PC
hp           11     printer
canon        12     printer
epson        13     printer
toshiba      21     laptop
hp           22     laptop
```

dell	23 laptop
samsung	3 PC
intel	4 PC
samsung	5 PC
apple	24 laptop
hp	6 PC
canon	7 PC

SQL> SELECT \* FROM PC;

MODEL	SPEED	RAM	HDD	CD	PRICE
1	2	500	1000	6	8000
2	3	1000	512	7	60000
3	1	256	512	8	9000
4	2	512	80	10	20000
6	4	256	512	8	60000
5	10	512	256	10	20000
7	3	1000	1024	4	80000

SQL> SELECT \* FROM LAPTOP;

MODEL	SPEED	RAM	SCREEN	PRICE
21	12	1024	15	20000
22	15	2048	17	20000
23	0	512	20	20000
24	2	256	14	45000

SQL> SELECT \* FROM PRINTER;

MODEL	COLOUR	TYPE	PRICE
11	white	inkjet	5000
12	black	laser	20000
13	grey	dot matrix	3000

**The INSERT INTO statements for Product are-**

```
insert into product values('samsung', 1, 'PC');
insert into product values('dell', 2, 'PC');
insert into product values('hp', 11, 'printer');
insert into product values('canon', 12, 'printer');
insert into product values('epson', 13, 'printer');
insert into product values('toshiba', 21, 'laptop');
insert into product values('hp', 22, 'laptop');
insert into product values('dell', 23, 'laptop');
insert into product values('samsung', 3, 'PC');
insert into product values('intel', 4, 'PC');
insert into product values('samsung', 5, 'PC');
insert into product values('apple', 24, 'laptop');
insert into product values('hp', 6, 'PC');
insert into product values('canon', 7, 'PC');
```

**The INSERT INTO statements for PC are-**

```
INSERT INTO PC VALUES (1,2,500,1000,6,8000);
INSERT INTO PC VALUES (2,3,1000,512,7,60000);
INSERT INTO PC VALUES (3,1,256,512,8,9000);
INSERT INTO PC VALUES (4,2,512,80,10,20000);
INSERT INTO PC VALUES (5,10,512,256,10,20000);
INSERT INTO PC VALUES (6,4,256,512,8,60000);
```

INSERT INTO PC VALUES (7,3,1000,1024,4,80000);

**The INSERT INTO statements for Laptop are-**

INSERT INTO LAPTOP VALUES (21,12,1024,15,20000);

INSERT INTO LAPTOP VALUES (22,15,2048,17,20000);

INSERT INTO LAPTOP VALUES (23,0,512,20,20000);

INSERT INTO LAPTOP VALUES (24,2,256,14,45000);

**The INSERT INTO statements for Printer are-**

INSERT INTO PRINTER VALUES (11,'white','inkjet',5000);

INSERT INTO PRINTER VALUES (12,'black','laser',20000);

INSERT INTO PRINTER VALUES (13,'grey','dot matrix',3000);

**Prepare the output of below queries:**

Q) Find laptops whose speed is slower than any PC

SELECT MODEL

FROM LAPTOP

WHERE SPEED < (SELECT min(SPEED) FROM PC);

Q) Find average hard disk size of PCs for all those manufacturers that make printers

SELECT AVG(HDD)

FROM PC,PRODUCT

WHERE PC.MODEL=PRODUCT.MODEL

AND MAKER IN

(SELECT P.MAKER

FROM PRODUCT P, PRINTER T

WHERE P.MODEL=T.MODEL);

Q) Find the makers of PC with the fastest processor among all those whose RAM is the smallest

SELECT MAKER FROM PRODUCT

WHERE MODEL IN

(SELECT MODEL FROM PC

WHERE SPEED=(SELECT MAX(SPEED) FROM PC WHERE MODEL IN  
(SELECT MODEL FROM PC WHERE RAM=(SELECT MIN(RAM) FROM PC))));

Q) Find hard disk sizes that occur in 3 or more PCs

```
SELECT HDD
FROM PC
GROUP BY HDD
HAVING COUNT(MODEL)>2;
```

Q) Give an alternate to laptop schema by adding a new attribute cd. Let the default value of the attribute be "NULL" if the laptop does not have a cd drive

```
ALTER TABLE Laptop ADD (Cd NUMBER(5));
```

Q) Find the manufacturer that makes at least 3 different models of PC

```
SELECT MAKER
FROM PRODUCT
WHERE TYPE='PC'
GROUP BY MAKER
HAVING COUNT(MODEL)>=3;
```

Q) Find the model number, speed and hard disk size for a PC that has either a 6x or 8x cd and price less than 10000. The output must be sorted by model number

```
SELECT MODEL,SPEED,HDD
FROM PC
WHERE CD IN (6,8)
AND PRICE<10000
ORDER BY MODEL;
```

Q) Find those pairs of PC models that have the same speed and RAM. A pair should be listed only once

```
SELECT DISTINCT P1.Model,P2.Model
FROM PC P1, PC P2
WHERE P1.Speed=P2.Speed AND P1.RAM=P2.RAM
AND NOT(P1.MODEL=P2.MODEL);
```

Q) Find all those laptops whose speed is more than any PC with the same price tag of Rs. 20000

```
Select model
From laptop
```

Where speed > (select max(speed) from pc where price=20000)  
and price=20000;

Q) Change the price of the PC to include fractional numbers as well.

ALTER TABLE PC MODIFY (Price NUMBER (10,2));

Assessment conducted

## Nov 18, 2022 Day 9

Consider the following schema of a Boat Reservation System relational database

Sailors (sid: integer, sname: string, rating: integer, age: integer, phno: integer)

Boats (bid: integer, bname: string, color: string, type: string)

Reserves (sid: integer, bid: integer, day: date)

Constraints on the Sailors table:

- sid is the primary key
- sname, rating, age cannot be blank
- rating lies between 1 and 10
- age lies between 20 and 50
- appropriate size should be chosen for the attribute phno

Constraints on the Boats table:

- bid is the primary key
- bname, color, type cannot be empty

Constraints on the Reserves table:

- sid, bid, day together make up the primary key
- sid is the foreign key which depends on sid of Sailors
- bid is the foreign key which depends on bid of Boats

Q) Find the name of sailors who have reserved a red boat.

Q) Find the color of the boats reserved by 'Ram' more than once.

- Q) Find the ages of the sailors whose name starts and ends with B and has at least three characters.
- Q) Find the names of the sailors who have reserved at least a boat.
- Q) Find the names of sailors who have reserved both a red and a green boat.
- Q) Find the names of sailors who have reserved a red boat but not a green boat.
- Q) For each red boat find the number of reservations for this boat.
- Q) Find sailors whose rating is better than some sailor named Tinku.
- Q) Find the sailors with the highest rating.
- Q) Find the age of the youngest sailor who is eligible to vote, i.e., older than 18) for each level of rating with at least two such sailors.
- Q) Find the names of sailors who are older than the oldest sailor with a rating of 10.
- Q) Find the average age of sailors for each rating level that has at least 2 sailors.
- Q) Find the no. of reservations for each Red Boat.
- Q) Find the name of the sailors who have reserved all the boats.
- Q) Find the names of sailors who have not reserved any boat.
- Q) Find the sailors who have reserved a yellow and a red boat on 25<sup>th</sup> June 2013.
- Q) Find the names of sailors who have reserved two different boats on the same day.

## Nov 25, 2022 Day 10

As a DBA, you have been tasked with segregating the employee details department-wise. This will enable department heads to manage their personnel better. The master table, containing details of all employees from all departments, is provided to you. Write PL/SQL code to carry out this operation.

```
create table Employee(Eid integer primary key, Ename varchar(100) NOT NULL, Salary integer NOT NULL, Dname varchar(20) NOT NULL);
```

```
insert into Employee VALUES(1, 'ABC', 4500, 'Marketing');
insert into Employee VALUES(2, 'ABC', 3500, 'Marketing');
```



```
insert into Employee VALUES(3, 'XYZ', 7500, 'Analytics');
insert into Employee VALUES(4, 'DEF', 2500, 'Finance');
insert into Employee VALUES(5, 'PQR', 5000, 'Marketing');
insert into Employee VALUES(6, 'GHI', 8000, 'Analytics');
insert into Employee VALUES(7, 'FFF', 6000, 'Finance');
insert into Employee VALUES(8, 'DWT', 5200, 'Finance');
insert into Employee VALUES(9, 'FFT', 2000, 'Operations');
insert into Employee VALUES(10, 'DCT', 3700, 'Analytics');
```

```
create table Marketing(Eid integer primary key, Ename varchar(100) NOT NULL, Salary integer NOT NULL);
```

```
create table Finance(Eid integer primary key, Ename varchar(100) NOT NULL, Salary integer NOT NULL);
```

```
create table Analytics(Eid integer primary key, Ename varchar(100) NOT NULL, Salary integer NOT NULL);
```

```
create table Operations(Eid integer primary key, Ename varchar(100) NOT NULL, Salary integer NOT NULL);
```

```
declare
```

```
    noOfRows integer;
    Deptname varchar(20);
    loopVar integer := 1;
    Empname varchar(100);
    Sal integer;
```

```
begin
```

```
    select count(*) into noOfRows from Employee;
    while loopVar <= noOfRows
    loop
        select Dname into Deptname from Employee where Eid = loopVar;
        select Ename into Empname from Employee where Eid = loopVar;
        select Salary into Sal from Employee where Eid = loopVar;
        if Deptname = 'Marketing' then
            insert into Marketing values(loopVar, Empname, Sal);
        elsif Deptname = 'Finance' then
            insert into Finance values(loopVar, Empname, Sal);
        elsif Deptname = 'Analytics' then
            insert into Analytics values(loopVar, Empname, Sal);
        elsif Deptname = 'Operations' then
            insert into Operations values(loopVar, Empname, Sal);
```

```
        end if;  
        loopVar := loopVar + 1;  
    end loop;  
end;  
/
```