**SQL PROJECT ON**

**UNIVERSITY DATABASE MANAGEMENT.**

**1.INTRODUCTION**

**1.1 PROJECT OVERVIEW:**

The university management is designed to manage the students, courses, faculties in an university. Here student, course how they are linked to each other. Here I am using Oracle SQL to run queries.

**1.2 OBJECTIVE**

* The main objective is to create a university database and manage tables in it.
* Here we will see different tables like student, faculty, course, department tables which are interlinked to each other.
* To implement SQL queries that create tables , insert a sample data into it and perform some necessary operations on that database.
* We will see the SQL queries from basic .

**SQL INTRO:**

* Basically SQL works on data, a data is a raw facts.
* A **database** is an organized collection of data that is stored and accessed electronically. It is designed to manage large amounts of information in a structured way, allowing users to efficiently store, retrieve, update, and delete data.
* SQL, or Structured Query Language, is a standardized programming language used to manage and manipulate relational databases. It is the most commonly used language for interacting with databases, allowing users to perform tasks such as querying data, updating records, inserting new data, and deleting existing data.

**RELATIONAL DATABASE:**

* Here I am working on relational database. The relational database is also a type of **database in which the data is in the form of tables.**

**Table:**

* The table contains the data in the form of rows and columns.
* **Rows** are horizontal rows used to store records.
* **Columns** are vertical lines which represent the attributes of the table.

**Constraints:**

* Constraints are nothing but rules given to the column names.
* There are different type of constraints like
* Primary key : In simple it follows unique property and doesn’t accept null values.
* common table name.
* Unique: It is nothing but the values are unique no duplicates are allowed.
* Not Null : Here the field shouldn’t allow null values from user.
* Check: It is used to check condition on column or condition given to column.

It has some statements to perform queries on it .

The statements are divided into categories:

* **DDL(Data Definition Languages): These are used to define the database** or table.

1. It contains operations like:
2. **Create**: It is used to create a table or database:
3. **Alter:** It is used to make changes in the names of the columns in table.
4. **Drop:** The drop is used to delete the object in the table like table or column name:
5. **Note:** After performing the drop and without performing any query we can get back It by using flashback in oracle SQL.
6. **Truncate:** It is used to delete records from the table.
7. **Rename :** It is used to change the name of the Table.

* **DML(Data Manipulation Language):** These are used to make changes in the records.

1. **Insert:** it is used to insert records into the table.
2. **Update:** It is used to update records in table.
3. **Delete:** It is used to delete records from table.

* **DCL(Data Control Language):** These are used to control the access permission to data base.
* **Grant:** Used to give permission to user.
* **Revoke:** Used to decline permissions given to use.
* **TCL(Transaction Control Language):** These are used when working on transactional data.
* **Commit:** It is used to give permission to store table in database.
* **Roll back:** It is used to get last transaction back.
* **Save point:** **SAVEPOINT** is a way to set a point within a transaction to which you can later roll back without affecting the entire transaction.
* **DQL(Data Query Language):** Italso called retrieval language .These are used to retrieve data from table.
* **Select:** Mainly we use select statement to retrieve data from table.
* **Projection** is nothing but retrieving data by mentioning column names.

**Software:** *Oracle SQL*

**2.1 Tables and Relationships**

Describe each table in the database, including its columns, data types, and constraints.

**2.1.1 Departments Table**

* **dept\_id**: Integer, Primary Key
* **dept\_name**: Varchar2(20), Not Null
* **hod**: Varchar2(20), Head of Department

**2.1.2 Courses Table**

* **course\_id**: Integer, Primary Key, Auto-Increment
* **course\_name**: Varchar2(100), Not Null
* **credits**: Integer, Not Null, must be > 0
* **department\_id**: Integer, Foreign Key (References Departments(dept\_id))
* **semester**: Varchar2(20), Not Null

**2.1.3 Faculties Table**

* **faculty\_id**: Integer, Primary Key, Auto-Increment
* **faculty\_name**: Varchar2(50), Not Null
* **faculty\_salary**:Number(10)
* **faculty\_email**: Varchar2(100), Unique, Not Null
* **dept\_id** :Number(3),Foreign key (References department(dept\_id)

**2.1.4 Students Table**

* **student\_id**: Integer, Primary Key, Auto-Increment
* **first\_name**: Varchar2(50), Not Null
* **last\_name**: Varchar2(50), Not Null
* **email**: Varchar2(100), Unique, Not Null
* **date\_of\_birth**: Date, Not Null
* **department\_id**: Integer, Foreign Key (References Departments(dept\_id))
* **course\_id**:Integer,Foreign key(References course(course\_id)
* **faculty\_id**:Integer,Foreign key(References faculty(faculty\_id)

**Queries:**

* **Create database:**

Create database university;

* Use university;

The above line will use database university to store data.

* **Create department table:**

*create table department(*

*dept\_id number(3),*

*dept\_name varchar2(20),*

*HOd varchar2(20),*

*constraint primaryk primary key(dept\_id));*

***output:*** *Table DEPARTMENT created****.***

Here we have created dept table with dept\_id as primary key.

* **Create course table:**

*create table course(*

*course\_id number(5),*

*course\_name varchar(20),*

*credits number(2),*

*semester number(2),*

*dept\_id number(3),*

*constraint prik primary key(course\_id),*

*constraint fk foreign key(dept\_id) references department(dept\_id));*

* + ***output****: Table COURSE created.*

Here I have created a course table with course\_id as primary key and dept\_id foreign key from department table.

* **Create faculties table:**

*CREATE TABLE faculty (*

*faculty\_id NUMBER(3),*

*faculty\_name VARCHAR2(20),*

*faculty\_email VARCHAR2(30) NOT NULL,*

*faculty\_salary NUMBER(10),*

*dept\_id NUMBER(3),*

*CONSTRAINT pkey\_faculty PRIMARY KEY (faculty\_id),*

*CONSTRAINT unique\_faculty\_email UNIQUE (faculty\_email),*

*CONSTRAINT fkey\_dept FOREIGN KEY (dept\_id) REFERENCES department(dept\_id)*

*);*

***Output*:** Table FACULTY created.

I have created a faculty table with faculty\_id as primary key and email is denoted as not null which doesn’t accept null values for email.

Dept\_id is a foreign key.

* + **Create student table:**

*create table std(*

*std\_id number(3),*

*std\_name varchar2(20),*

*std\_dob date not null,*

*std\_email varchar2(20)not null,*

*std\_course varchar2(10) ,*

*std\_faculty varchar2(20),*

*dept\_id number(3),*

*std\_marks number(4),*

*constraint pk\_stud primary key(std\_id),*

*constraint fk\_stud foreign key(dept\_id) references department(dept\_id),*

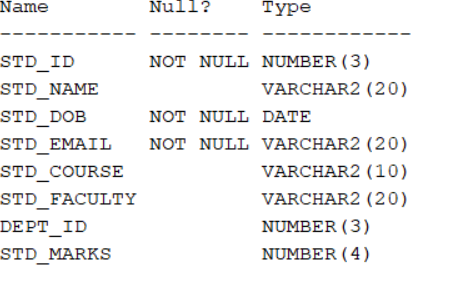
*constraint check\_stud check(std\_marks>0));*

***output:*** Table STD created.

**NOTE:** To see the structure of table use this:

**Desc table\_name;**

desc std



**To insert data into tables:**

**Syntax :**INSERT INTO table\_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

Now I am inserting values into dept table:

**Query**:

**Dept table:**

insert into department(dept\_id,dept\_name,hod)values (10,'CSE','SSN');

insert into department(dept\_id,dept\_name,hod)values (20,'CSE-AI','SAI');

insert into department(dept\_id,dept\_name,hod)values (30,'CSE-DS','VAM');

insert into department(dept\_id,dept\_name,hod)values (50,'CSE-ML','RAV');

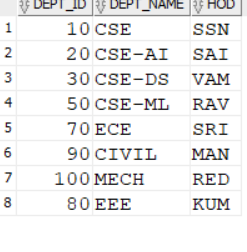
insert into department(dept\_id,dept\_name,hod)values (70,'ECE','SRI');

insert into department(dept\_id,dept\_name,hod)values (90,'CIVIL','MAN');

insert into department(dept\_id,dept\_name,hod)values (100,'MECH','RED');

insert into department(dept\_id,dept\_name,hod)values (80,'EEE','KUM');

**To see the details or records in table :** Select \* from department;



**Course table:**

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1003,'Course\_3',1,7,40);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1004,'Course\_4',3,5,90);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1005,'Course\_5',4,7,60);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1006,'Course\_6',3,6,60);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1007,'Course\_7',1,6,110);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1008,'Course\_8',3,4,70);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1009,'Course\_9',3,2,30);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1010,'Course\_10',3,7,50);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1011,'Course\_11',3,4,110);

--Row 12

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1012,'Course\_12',NULL,2,50);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1013,'Course\_13',3,3,70);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1014,'Course\_14',4,2,50);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1015,'Course\_15',2,4,50);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1016,'Course\_16',4,2,30);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1017,'Course\_17',1,NULL,90);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1018,'Course\_18',2,3,60);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1019,'Course\_19',NULL,1,NULL);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1020,'Course\_20',4,8,80);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1021,'Course\_21',4,5,40);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1022,'Course\_22',2,8,100);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1023,'Course\_23',NULL,5,70);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1024,'Course\_24',3,8,NULL);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1025,'Course\_25',4,3,100);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1026,'Course\_26',1,5,110);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1027,'Course\_27',3,7,40);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1028,'Course\_28',3,2,110);

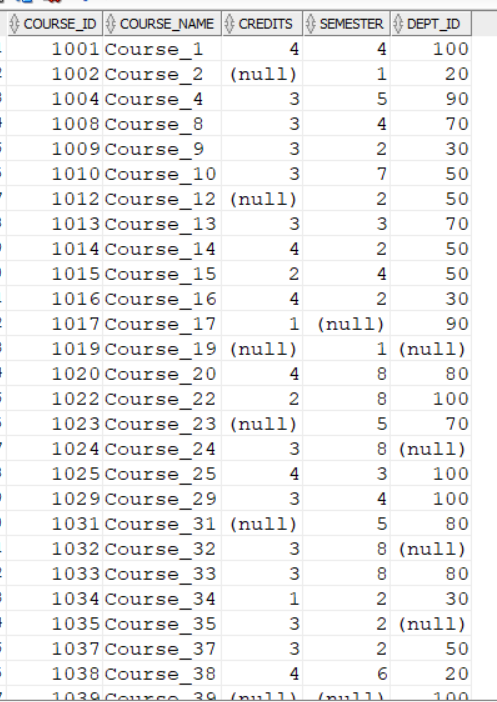
INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1029,'Course\_29',3,4,100);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1030,'Course\_30',1,2,110);

INSERT INTO COURSE (COURSE\_ID, COURSE\_NAME, CREDITS, SEMESTER, DEPT\_ID) VALUES (1031,'Course\_31',NULL,5,80);

**To see the details or records in table :**

Use : select \* from course;

****

**Faculty Table:**

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (1, 'John Doe', 'john.doe@example.com', 70000, 10);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (2, 'Jane Smith', 'jane.smith@example.com', 75000, 20);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (3, 'Alice Johnson', 'alice.johnson@example.com', NULL, 30);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (4, 'Bob Brown', 'bob.brown@example.com', 65000, 50);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (5, 'Charlie Davis', 'charlie.davis@example.com', 70000, 70);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (6, 'Deborah Clark', 'deborah.clark@example.com', 72000, 80);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (7, 'Evan Wright', 'evan.wright@example.com', 73000, 90);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (8, 'Fiona Harris', 'fiona.harris@example.com', 74000, 100);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (9, 'George Martinez', 'george.martinez@example.com', 75000, 10);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (10, 'Hannah Lewis', 'hannah.lewis@example.com', NULL, 20);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (11, 'Ian Walker', 'ian.walker@example.com', 68000, 30);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (12, 'Jasmine Hall', 'jasmine.hall@example.com', NULL, 50);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (13, 'Kevin Young', 'kevin.young@example.com', 71000, 70);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (14, 'Laura King', 'laura.king@example.com', NULL, 80);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (15, 'Michael Adams', 'michael.adams@example.com', 69000, 90);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (16, 'Nina Scott', 'nina.scott@example.com', 74000, 100);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (17, 'Oscar Baker', 'oscar.baker@example.com', NULL, 10);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (18, 'Paula Nelson', 'paula.nelson@example.com', 70000, 20);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (19, 'Quincy Carter', 'quincy.carter@example.com', 72000, 30);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (20, 'Rachel Mitchell', 'rachel.mitchell@example.com', NULL, 50);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (21, 'Steve Roberts', 'steve.roberts@example.com', 75000, 70);

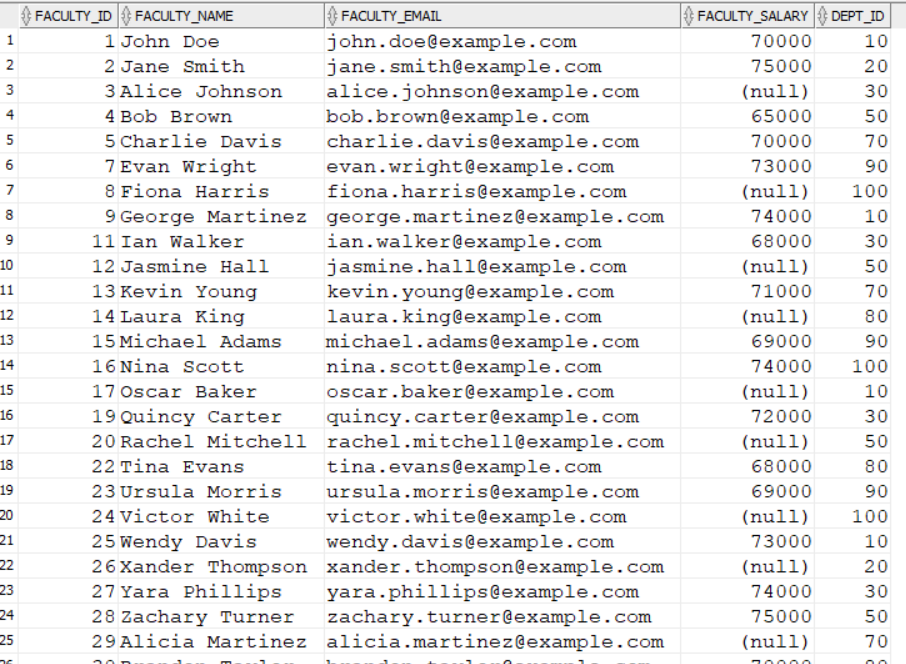
INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (22, 'Tina Evans', 'tina.evans@example.com', 68000, 80);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (45, 'Quinn Lewis', 'quinn.lewis@example.com', NULL, 10);

INSERT INTO faculty (faculty\_id, faculty\_name, faculty\_email, faculty\_salary, dept\_id) VALUES (46, 'Rebecca Wilson', 'rebecca.wilson@example.com', NULL, 20);

**To see the details or records in table :**

Use : select \* from faculty;



**Here** I **have changed the column names in student table using alter command i.e std\_course to course\_id**

And std\_faculty to faculty\_id

ALTER TABLE std RENAME COLUMN std\_course TO course\_id;

**Add foreign key constraint to link two tables:**

ALTER TABLE std ADD constraint fk\_std foreign key(course\_id) references course(course\_id);

**Change data type**: ALTER TABLE std MODIFY course\_id number(3);

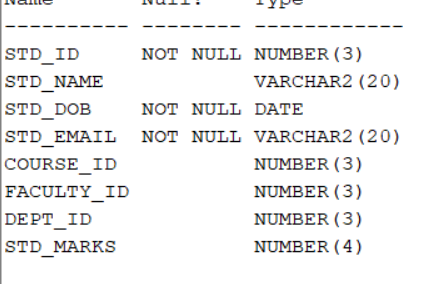
MODIFY course\_id number(3);ALTER TABLE std RENAME COLUMN std\_faculty TO faculty\_id;

**Change data type**: ALTER TABLE std MODIFY faculty\_id number(3);

**Add foreign key constraint to link two tables:**

ALTER TABLE std ADD constraint fkp\_std foreign key(faculty\_id) references faculty(faculty\_id);

**Desc std; --**change table



**Data into student table**:

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (1, 'John Doe', TO\_DATE('2000-01-15', 'YYYY-MM-DD'), 'john.doe@example.com', 1001, 8, 100, 85);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (2, 'Jane Smith', TO\_DATE('2000-02-20', 'YYYY-MM-DD'), 'jane.smith@example.com', 1002, 26, 20, 90);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (3, 'Alice Johnson', TO\_DATE('2000-03-25', 'YYYY-MM-DD'), 'alice.johnson@example.com', 1004, 15, 90, 78);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (4, 'Bob Brown', TO\_DATE('2000-04-30', 'YYYY-MM-DD'), 'bob.brown@example.com', 1008, 29, 70, 82);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (5, 'Charlie Davis', TO\_DATE('2000-05-05', 'YYYY-MM-DD'), 'charlie.davis@example.com', 1009, 27, 30, 88);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (6, 'Daisy Evans', TO\_DATE('2000-06-10', 'YYYY-MM-DD'), 'daisy.evans@example.com', 1010, 28, 50, 75);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (7, 'Edward Green', TO\_DATE('2000-07-15', 'YYYY-MM-DD'), 'edward.green@example.com', 1012, 28, 50, 92);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (44, 'Paul Scott', TO\_DATE('2003-09-25', 'YYYY-MM-DD'), 'paul.scott@example.com', 1014, 20, 50, 93);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (45, 'Quinn Turner', TO\_DATE('2003-10-30', 'YYYY-MM-DD'), 'quinn.turner@example.com', 1015, 28, 50, 84);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (46, 'Rachel Underwood', TO\_DATE('2003-11-05', 'YYYY-MM-DD'), 'rachel.underwood@example.com', 1016, 35, 30, 80);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (47, 'Sam Vance', TO\_DATE('2003-12-10', 'YYYY-MM-DD'), 'sam.vance@example.com', 1017, 23, 90, 91);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (48, 'Tina Walker', TO\_DATE('2004-01-15', 'YYYY-MM-DD'), 'tina.walker@example.com', 1020, 30, 80, 88);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

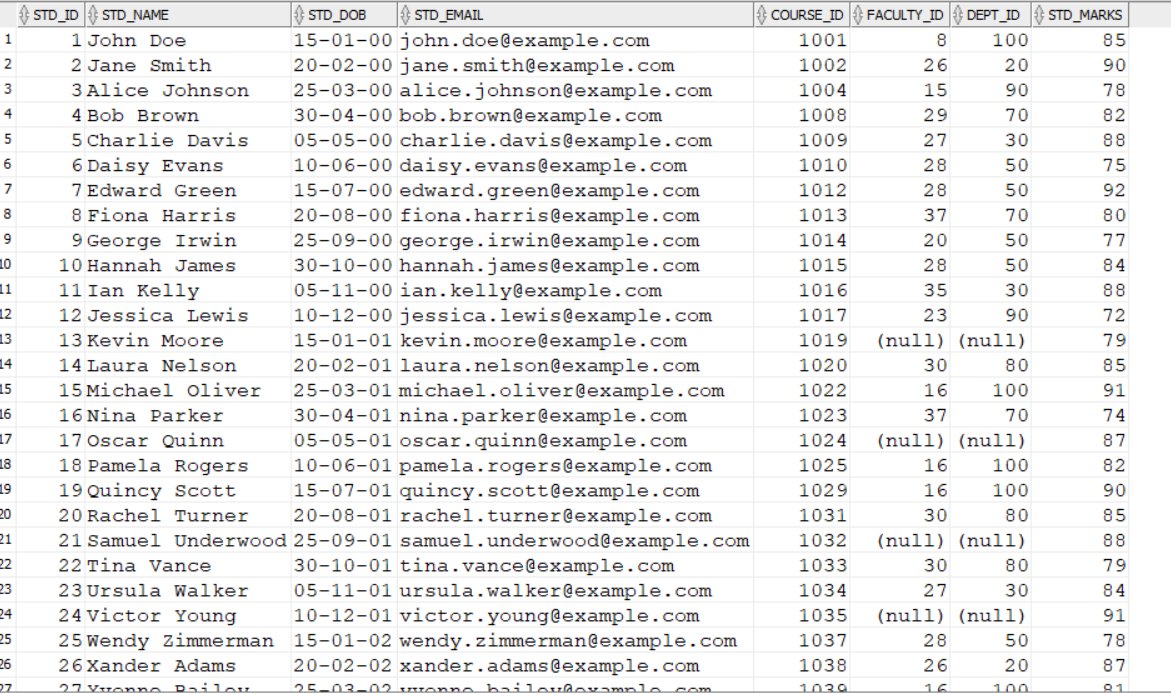
VALUES (49, 'Ulysses Young', TO\_DATE('2004-02-20', 'YYYY-MM-DD'), 'ulysses.young@example.com', 1022, 16, 100, 76);

INSERT INTO std (std\_id, std\_name, std\_dob, std\_email, course\_id, faculty\_id, dept\_id, std\_marks)

VALUES (50, 'Vera Zimmerman', TO\_DATE('2004-03-25', 'YYYY-MM-DD'), 'vera.zimmerman@example.com', 1023, 37, 70, 83);

**To see the details or records in table :**

Use : *select \* from std;*



**Update records:**

To update records we use update statement :

**Syntax:**

*UPDATE table\_name*

*SET column1 = value1,*

*column2 = value2,*

*...*

*WHERE condition;*

**Query**

*update course set*

*credits=credits+1 ;*

In the above I have updated credits in the course table by increasing every column with +1.

To update particular columns we use where clause.

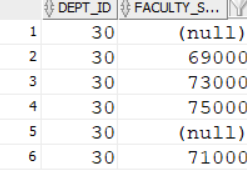
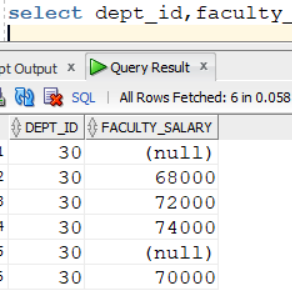
**Where :** Where is used to filter the records . The WHERE clause in SQL is used to filter records that meet specific conditions. It is commonly used in SELECT, UPDATE, DELETE, and other SQL statements to specify which records should be affected.

Now I am updating the salary of the faculty who are working in CSE-DS dept i.e. dept\_id=30 with increment of 1000.

**Query:**

*update faculty*

*set faculty\_salary=faculty\_salary+1000 where dept\_id=30;*

**Afterupdate: Before update: **

**Queries on where clause:**

**1.To display student details who have marks greater than 90.**

*SELECT \* FROM std WHERE std\_marks > 90;*

**2.To display student details who have born in year “03”.**

*SELECT \* FROM std WHERE std\_dob >= to\_date('01-01-03','dd-mm-yy')*

*and std\_dob <=to\_date('31-12-03','dd-mm-yy');*

**3.To display student details whose name is “Iris Lewis”.**

*SELECT \* FROM std WHERE std\_name='Iris Lewis';*

**4. To display faculty details where salary is Null.**

*select \* from faculty where faculty\_salary is null;*

**5. To display course details wher semester is above 5.**

*select \* from course where semester>5;*

**DISTINCT:**

The distinct is used to find unique values in that column.

**Distinct (coulumn\_name)**

**Queries:**

*select distinct(std\_name) from std;*

**output:**

Alice Johnson Vera Zimmerman

Jane Smith Hannah James

Kevin Moore

Leo Oliver

Ulysses Young

***Order by:*** The ORDER BY keyword is used to sort the result-set in ascending or descending order.

**Syntax**: *SELECT*column1*,* column2, ... *FROM*table\_name *ORDER BY*column1, column2, ... *ASC|DESC;*

*By default the order is ascending if you wont mention any order.*

**Queries:**

**To sort the student names based on their marks in ascending order.**

*select std\_name,std\_marks from std order by std\_marks asc;*

***Output:***

Jessica Lewis 72

Nina Parker 74

Leo Oliver 75

Daisy Evans 75

Ulysses Young 76

George Irwin 77

Frank Ives 77

**To sort student names in alphabetical order:**

*select std\_name,std\_marks from std order by std\_name asc;*

**Output:**

Alice Johnson 78

Amanda Davis 92 Catherine Fisher 85

Amanda Davis 92 Charlie Davis 88

Bob Brown 82 Daisy Evans 75

Brian Edwards 79

**To display the dept name and dept\_id and dept\_name in asc and dept\_id in desc:**

*select dept\_id,dept\_name from department order by dept\_id desc,dept\_name asc;*

*output:*

100 MECH

90 CIVIL

80 EEE

70 ECE

50 CSE-ML

30 CSE-DS

20 CSE-AI

10 CSE

**AND and OR :**

AND is used when we want to filter the data based on two or conditions. The and returns the record if all the conditons are true **­­­­­**

OR is used to display record when any one condition is true among the given records.

**Queries:**

**To display student details who are having dept\_id=20 and faculty\_id=26:**

*select \* from std where dept\_id=20 and faculty\_id=26;*

**To display faculty details who are working in dept\_id =70 and earning more than 50000 salary:**

*select \* from faculty where faculty\_salary>50000 and dept\_id=70;*

**To display the details of dept who HOD as SSN and who are working at dept\_id greater than 50:**

*select \* from department where HOD='SSN' or dept\_id>50;*

Here it displays whose name is SSN or who are working having dept\_id >50.

**Aggregate functions:**

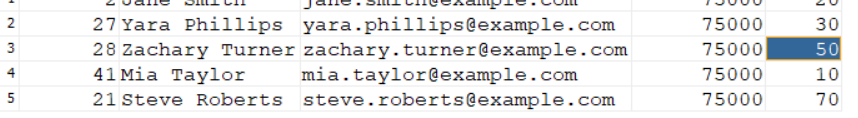
**1.Max 4.Average**

**2.Min 5.Count**

**3.Sum**

**To display the faculty details who are getting maximum salary:**

*select \* from faculty where faculty\_salary= (select max(faculty\_salary) from faculty);*

**

**To display the faculty details who are getting minimum salary:**

*select \* from faculty where faculty\_salary= (select min(faculty\_salary) from faculty);*

***4 Bob Brown bob.brown@example.com 65000 50***

***To display the no of students in department name CSE-DS as dept\_id is 30:***

*select count(std\_id) from std where dept\_id=30;*

***Result: 7***

**To find the students who are getting marks greater than average score:**

*select \* from std where std\_marks>(select avg(std\_ml.arks) from std );*

**Count the total number of departments.**

*select count(\*) from department;*

**Find the average marks of all students.**

select avg(std\_marks) from std;

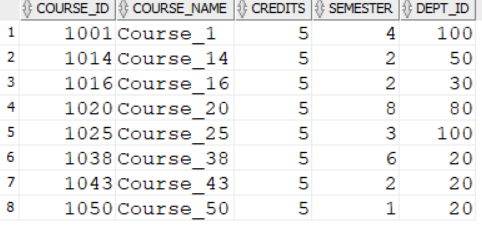
**Calculate the total salary of all faculties in the 'CSE' department**.

*select sum(faculty\_salary) from faculty where dept\_id=(select dept\_id from department where dept\_name='CSE');*

*Result:* *364000*

**Find the maximum credits offered by any course.**

*select \* from course where credits=(select max(credits) from course);*

**

**Find the number of students in each department.**

*select std.dept\_id,count(std\_id)as count from std group by dept\_id;*

**Calculate the total number of courses offered across all departments.**

*select course.dept\_id,count(course\_name) from course group by dept\_id;*

**Calculate the average credits for all courses in semester 2.**

*select course.course\_id,avg(credits) from course where semester=2 group by course\_id;*

**Retrieve the highest marks in the 'Mech' department.**

*SELECT STD.DEPT\_ID,MAX(STD\_MARKS) FROM STD WHERE DEPT\_ID*

*in (SELECT DEPT\_ID FROM DEPARTMENT WHERE DEPT\_NAME='MECH')*

*group by std.dept\_id;*

**Retrieve the second highest salary from faculty table.**

*select max(faculty\_salary)as second\_max\_salary*

*from faculty*

*where faculty\_salary<(select max(faculty\_salary)*

*from faculty);*

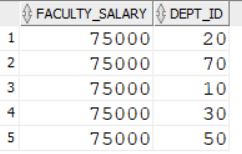
**Retrieve first 5 maximum salaries and their dept\_id from faculty:**

*select distinct(faculty\_salary),dept\_id from faculty*

*where faculty\_salary is not null*

*order by faculty\_salary desc*

*fetch first 5 rows only;*

**

**Retrieve 9 max salary and their dept\_id and dept\_name from faculty:**

*select faculty.dept\_id ,faculty.faculty\_salary,department.dept\_name from*

*faculty inner join department on faculty.dept\_id=department.dept\_id*

*where faculty\_salary is not null*

*order by faculty.faculty\_salary desc*

*offset 8 rows*

*fetch first 1 rows only;*

**Display the student details along with their hod name for all the students :**

*select std.\*,HOD from std inner join department on std.dept\_id=department.dept\_id ;*

**Display the faculty who are getting salary greater than “Bob Brown” and less than “Paul Harris” and working in “CIVIL” department.**

*select \* from faculty where faculty\_salary between*

*(select faculty\_salary from faculty where faculty\_name='Bob Brown') and*

*(select faculty\_salary from faculty where faculty\_name='Paul Harris') and*

*dept\_id=(select dept\_id from department where dept\_name ='CIVIL');*

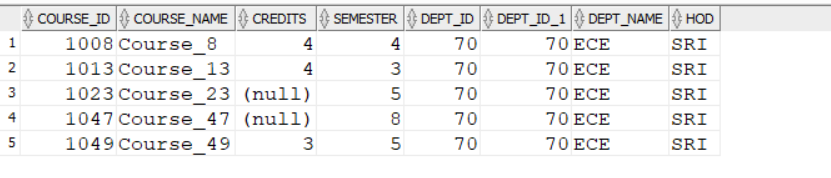
**List all courses offered by the departments where the HOD is ' SRI '.**

*select \* from course where dept\_id=(select dept\_id from department where HOD='SRI');*

***or***

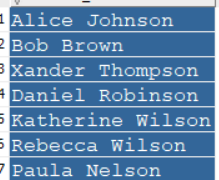
*select \* from course inner join department on course.dept\_id=department.dept\_id*

*where department.HOD='SRI';*

**

**Display names of faculty whose name ends with “N”:**

select faculty\_name from faculty where faculty\_name like '%n';



**Display names of faculty whose name contain “or” string in their name:**

*select faculty\_name from faculty where faculty\_name like '%or%';*

**Display names of student whose name starts with “V” and belongs to department “MECH”:**

*select faculty\_name,dept\_id from faculty where faculty\_name like 'V%' and*

*dept\_id=(select dept\_id from department where dept\_name='MECH');*

**Display student details whose name contains “e” as the second letter and belongs to “CSE-ML” and marks greater than 70:**

*select \* from std where std\_name like '\_e%' and std\_marks>70 and*

*dept\_id=(select dept\_id from department where dept\_name='CSE-ML') ;*

**Display faculty details whose name contain exactly one ‘V’ either it is capital or small:**

For these first convert name to either capital or small letters then apply conditions.

*select \* from faculty where upper(faculty\_name) like '%V%' and*

*upper(faculty\_name) not like '%V%V%';*

**Display student name and Date of Birth(DOB) who are born in “JULY”**:

select std\_name,std\_dob from std where std\_dob like '%07%';

**Display student name and Date of Birth(DOB) who are born in the year “2004” and has faculty as “Isabella scott”**:

*select std\_name,std\_dob from std where std\_dob like '%04' and*

*faculty\_id=(select faculty\_id from faculty where faculty\_name='Isabella Scott');*

**Display the no of students working in each department remove who have null values:**

*select dept\_id,count(\*) from std*

*where dept\_id is not Null*

*group by dept\_id;*

**Calculate the total number of courses offered across all departments***:*

*select dept\_id, count(course\_id) from course*

*where dept\_id is not Null*

*group by dept\_id ;*

**Calculate the average credits for all courses in semester 2.**

*select course\_id,course\_name,avg(credits) as AVG\_CREDITS from course where semester=2 and credits is not null*

*group by course\_id,course\_name;*

**Display maximum salary in each department and sort them in descending order in faculty table:**

*select dept\_id,max(faculty\_salary)as salary from faculty*

*Group by dept\_id*

*order by salary desc;*

**Find the name of the department with the most students.**

*select dept\_name from department where dept\_id=(*

*select dept\_id from std*

*group by dept\_id*

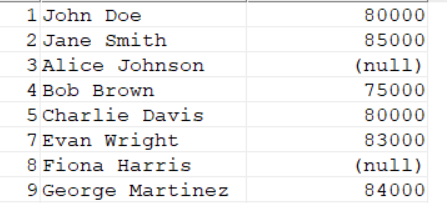
*order by count(std\_id)*

*fetch first 1 row only);*

**Display faculty\_id,faculty\_name and salary with bonus of 10% on their salary:**

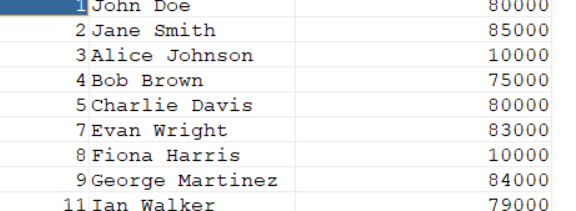
*select faculty\_id,faculty\_name,faculty\_salary + faculty\_salary\*(10/100) from faculty;*

**Display faculty details with increment of salary for everyone with 10000 and for null values place these 10000:**

****

*select faculty\_id,faculty\_name,faculty\_salary+10000 from faculty;*

If I use above query the null values are not replaces for these we have to use “NVL” called as NULL VALUE LOGIC which places the values in null.**NVL(column\_name,value)**

*Select faculty\_id,faculty\_name,NVL(faculaty\_salary+10000,10000 )from faculty; *

**List the courses offered by the department with the highest number of students.**

*select course\_name from course where dept\_id=*

*(select dept\_id from std*

*group by dept\_id*

*order by count(\*) desc*

*fetch first 1 row only);*

**Retrieve the details of the faculty with the highest salary.**

*select \* from faculty where faculty\_salary=(select max(faculty\_salary) from faculty );*

**Get the names of departments where no faculty has a salary below 30,000.**

*select dept\_name from department where dept\_id in*

*(select dept\_id from faculty where faculty\_salary >=30000);*

**List the students who are enrolled in more than one course.**

*select \* from std where course\_id in*

*(select course\_id from std*

*group by course\_id*

*having count(std\_id)>1);*

**Retrieve the details of courses offered by departments with greater than 5 faculties.**

*select \* from course where dept\_id in*

*(select dept\_id from faculty group by dept\_id*

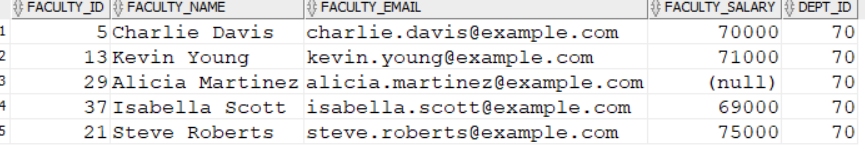
*having count(faculty\_id)>5);*

**Find the students who do not belong to any department.**

*select \* from std where dept\_id is Null;*

**Display the faculty details of student “Olivia rogers”:**

*select \* from faculty where dept\_id in (select dept\_id from std where std\_name='Olivia Rogers');*

**

**Retrieve all students along with their department names.**

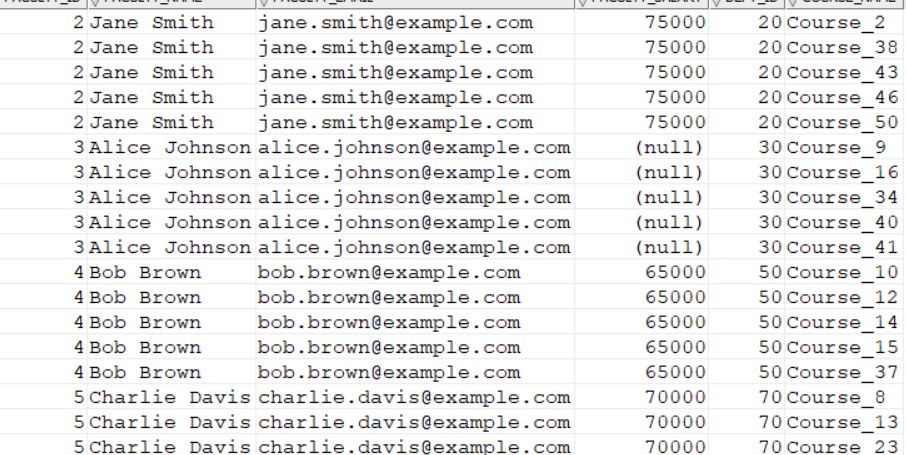
*select std.\*,department.dept\_name from std inner join department on std.dept\_id=department.dept\_id;*

**List all courses offered by the departments where the HOD is “VAM”.**

*select c.\* from course c inner join department d on c.dept\_id =d.dept\_id where d.HOD='VAM';*

**Find the faculties along with the courses they are teaching.**

*select f.\*,c.course\_name from faculty f inner join course c on f.dept\_id=c.dept\_id;*

**

**List students along with their respective faculty members. List students along with their respective faculty members.**

*select s.\*,f.faculty\_name from std s inner join faculty f on s.faculty\_id=f.faculty\_id;*

**

**Retrieve the names of students and the departments they belong to.**

*select s.std\_name,d.dept\_name from std s inner join department d on s.dept\_id=d.dept\_id;*

**Find the courses and the names of students enrolled in each course.**

*select c.course\_name,s.std\_name from course c inner join std s on s.course\_id=c.course\_id*

*order by c.course\_name;*

**List the faculties who belong to the 'CIVIL' department.**

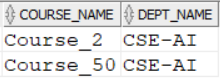
*select f.\* from faculty f inner join department d on f.dept\_id=d.dept\_id where d.dept\_name='CIVIL';*

**Show all students who are being taught by faculties with a salary greater than 70,000.**

*select s.\* from std s inner join faculty f on s.faculty\_id=f.faculty\_id where f.faculty\_salary>70000;*

**Retrieve the course names and department names for courses offered in semester 1.**

*select c.course\_name,d.dept\_name from course c inner join department d on c.dept\_id=d.dept\_id where semester=1;*

**

**Find the details of students who have the same dept\_id as their faculty members.**

*select s.\* from std s inner join faculty f on s.dept\_id=f.dept\_id where s.dept\_id=f.dept\_id;*

**Retrieve the names of departments that do not offer any courses.**

*SELECT d.dept\_name*

*FROM department d*

*LEFT JOIN course c ON d.dept\_id = c.dept\_id*

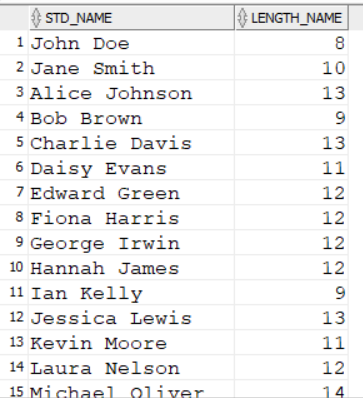
*WHERE c.dept\_id IS NULL;*

***Single Row Functions:***

**Length:** Length function is used to find the length of the record:

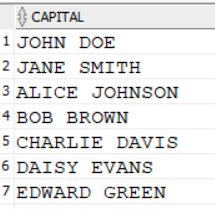
**Syntax: length(string)**

*select std\_name,length(std\_name)as length\_name from std;*

****

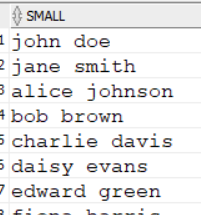
**Upper:** This function is used to convert all the characters to the capital letters.

*select Upper(std\_name) as capital from std;*

**

**Lower:** This is used to display all the characters in the small letters.

*select lower(std\_name) as small from std;*

**

**INITCAP:** This function makes first letter as capital

*select INITCAP(std\_name) as first\_capital from std;*

**Concat(str1,str2):** This joins two strings

*select concat(std\_name, std\_email) from std;*

**

*If we want to concate more than three words we can use nested concatenation:*

***select concat('Royal ',concat('Challenges','Bangalore')) from dual;***

**Royal ChallengesBangalore**

We can use **“||”** for concatenation:

select std\_name ||' ' ||std\_email from std**;**

**Substr:**  Used to get sub strings

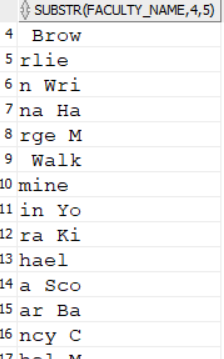
**Syntax: Substr(string,position,[length])**

String is the value or word

Position is the from which position you have to get string

Length is the number of positions from position.

*select substr(faculty\_name,4,5) from faculty;*

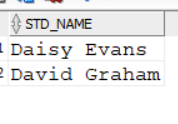
**

*If position is “1” the entire word is the output without number of positions.*

*select substr(faculty\_name,1) from faculty;*

**To display student name whose name starts with “D”:**

*select std\_name from std where substr(std\_name,1,1)='D';*

****

**Replace:**

Replace(original\_string,string,new\_string) :

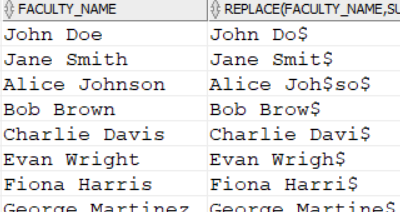
**Replace “V” with “S” in the word “RVKR”:**

*select replace('RVKR','V','S') from dual;*

***RSKR***

**Place “$” at the last letter of the faculty name:**

*select faculty\_name,replace(faculty\_name,substr(faculty\_name,-1),'$') from faculty;*

**

**INSTR:** It is used to find the index position of the letter or string:

**Syntax: INSTR(original\_string, string,position,[occurrence])**

Position says that from which position we have to check and occurance is the frequency by default occurance is 1:

*select INSTR('ANDHRA PRADESH','A',1) from dual; “1”*

*select INSTR('ANDHRA PRADESH','A',1) from dual; “6”*

*select INSTR('ANDHRA PRADESH','A',1,3) from dual; “10”*

**TRIM:** Remove spaces from the words .It is of two types

LTRIM Removes left space

RTRIM Removes right space

*select trim(' abc ') from dual; “abc”*

*select ltrim(' abc ') from dual; “ abc “*

**Case:**

*SELECT*

*column1,*

*column2,*

*CASE*

*WHEN condition1 THEN result1*

*WHEN condition2 THEN result2*

*...*

*ELSE result\_default*

*END AS alias\_name*

*FROM*

*table\_name;*

**Extract student id , name and marks from student with grades as**

**When std\_marks > 70 AND std\_marks < 75 THEN 'E' WHEN std\_marks >= 75 AND std\_marks < 80 THEN 'D' WHEN std\_marks >= 80 AND std\_marks < 85 THEN 'C' WHEN std\_marks >= 85 AND std\_marks < 90 THEN 'B' WHEN std\_marks >= 90 AND std\_marks < 95 THEN 'A' WHEN std\_marks >= 95 AND std\_marks <= 100 THEN 'A+' ELSE 'F'**

*select std\_id,std\_name,std\_marks,*

*case*

*when std\_marks > 70 and std\_marks<75 then 'E'*

*when std\_marks >=75 and std\_marks<80 then 'D'*

*when std\_marks >= 80 and std\_marks<85 then 'C'*

*when std\_marks >= 85 and std\_marks<90 then 'B'*

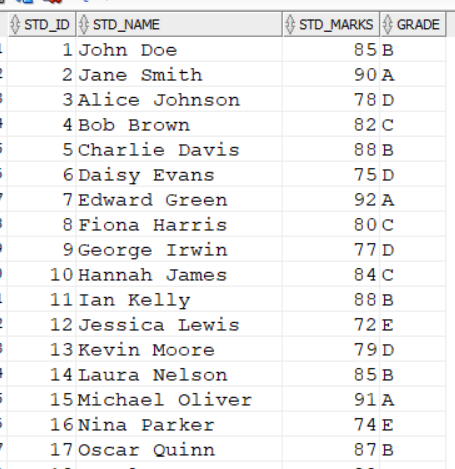
*when std\_marks >= 90 and std\_marks<95 then 'A'*

*when std\_marks >= 95 and std\_marks<100 then 'A+'*

*else 'F'*

*end as grade*

*from std;*

**

**Extract faculty id,name ,faculty salary and give a hike of 20% if the salary is between 60k and 70k and a 24% if the salary between 70k and 80k and 30% If the salary between more than 80k and a base salary of 30k if salary is “NULL”**

**And sort them in ascending order by their hike.**

*select faculty\_id,faculty\_name,faculty\_salary,*

*case*

*when faculty\_salary between 60000 and 70000 then faculty\_salary\*20/100*

*when faculty\_salary between 70000 and 80000 then faculty\_salary\*24/100*

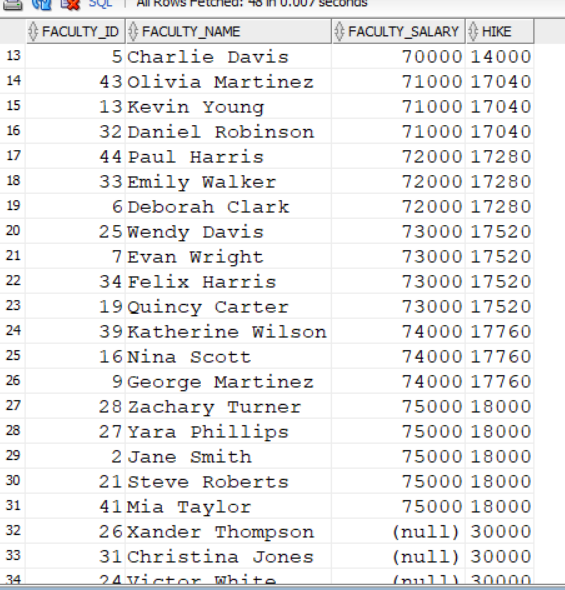
*when faculty\_salary>80000 then faculty\_salary\*30/100*

*when faculty\_salary is Null then 30000*

*end as hike*

*from faculty*

*order by hike asc;*

****