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:
     I. **Create**: It is used to create a table or database:
     II. **Alter:** It is used to make changes in the names of the columns in table.
     III. **Drop:** The drop is used to delete the object in the table like table or column name:
        1. **Note:** After performing the drop and without performing any query we can get back It by using flashback in oracle SQL.
     IV. **Truncate:** It is used to delete records from the table.
     V. **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):** It also 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

```
Name          Null?    Type
------------ -------- ------------
STD_ID       NOT NULL NUMBER(3)
STD_NAME              VARCHAR2(20)
STD_DOB      NOT NULL DATE
STD_EMAIL    NOT NULL VARCHAR2(20)
STD_COURSE            VARCHAR2(10)
STD_FACULTY           VARCHAR2(20)
DEPT_ID               NUMBER(3)
STD_MARKS             NUMBER(4)
```

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;

	DEPT_ID	DEPT_NAME	HOD
1	10	CSE	SSN
2	20	CSE-AI	SAI
3	30	CSE-DS	VAM
4	50	CSE-ML	RAV
5	70	ECE	SRI
6	90	CIVIL	MAN
7	100	MECH	RED
8	80	EEE	KUM

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;

COURSE_ID	COURSE_NAME	CREDITS	SEMESTER	DEPT_ID
1001	Course_1	4	4	100
1002	Course_2	(null)	1	20
1004	Course_4	3	5	90
1008	Course_8	3	4	70
1009	Course_9	3	2	30
1010	Course_10	3	7	50
1012	Course_12	(null)	2	50
1013	Course_13	3	3	70
1014	Course_14	4	2	50
1015	Course_15	2	4	50
1016	Course_16	4	2	30
1017	Course_17	1	(null)	90
1019	Course_19	(null)	1	(null)
1020	Course_20	4	8	80
1022	Course_22	2	8	100
1023	Course_23	(null)	5	70
1024	Course_24	3	8	(null)
1025	Course_25	4	3	100
1029	Course_29	3	4	100
1031	Course_31	(null)	5	80
1032	Course_32	3	8	(null)
1033	Course_33	3	8	80
1034	Course_34	1	2	30
1035	Course_35	3	2	(null)
1037	Course_37	3	2	50
1038	Course_38	4	6	20
1039	Course_39	(null)	(null)	100

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;

	FACULTY_ID	FACULTY_NAME	FACULTY_EMAIL	FACULTY_SALARY	DEPT_ID
1	1	John Doe	john.doe@example.com	70000	10
2	2	Jane Smith	jane.smith@example.com	75000	20
3	3	Alice Johnson	alice.johnson@example.com	(null)	30
4	4	Bob Brown	bob.brown@example.com	65000	50
5	5	Charlie Davis	charlie.davis@example.com	70000	70
6	7	Evan Wright	evan.wright@example.com	73000	90
7	8	Fiona Harris	fiona.harris@example.com	(null)	100
8	9	George Martinez	george.martinez@example.com	74000	10
9	11	Ian Walker	ian.walker@example.com	68000	30
10	12	Jasmine Hall	jasmine.hall@example.com	(null)	50
11	13	Kevin Young	kevin.young@example.com	71000	70
12	14	Laura King	laura.king@example.com	(null)	80
13	15	Michael Adams	michael.adams@example.com	69000	90
14	16	Nina Scott	nina.scott@example.com	74000	100
15	17	Oscar Baker	oscar.baker@example.com	(null)	10
16	19	Quincy Carter	quincy.carter@example.com	72000	30
17	20	Rachel Mitchell	rachel.mitchell@example.com	(null)	50
18	22	Tina Evans	tina.evans@example.com	68000	80
19	23	Ursula Morris	ursula.morris@example.com	69000	90
20	24	Victor White	victor.white@example.com	(null)	100
21	25	Wendy Davis	wendy.davis@example.com	73000	10
22	26	Xander Thompson	xander.thompson@example.com	(null)	20
23	27	Yara Phillips	yara.phillips@example.com	74000	30
24	28	Zachary Turner	zachary.turner@example.com	75000	50
25	29	Alicia Martinez	alicia.martinez@example.com	(null)	70

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

```
Name          Null?    Type
----------- -------- ------------
STD_ID      NOT NULL NUMBER(3)
STD_NAME             VARCHAR2(20)
STD_DOB     NOT NULL DATE
STD_EMAIL   NOT NULL VARCHAR2(20)
COURSE_ID            NUMBER(3)
FACULTY_ID           NUMBER(3)
DEPT_ID              NUMBER(3)
STD_MARKS            NUMBER(4)
```

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;*

	STD_ID	STD_NAME	STD_DOB	STD_EMAIL	COURSE_ID	FACULTY_ID	DEPT_ID	STD_MARKS
1	1	John Doe	15-01-00	john.doe@example.com	1001	8	100	85
2	2	Jane Smith	20-02-00	jane.smith@example.com	1002	26	20	90
3	3	Alice Johnson	25-03-00	alice.johnson@example.com	1004	15	90	78
4	4	Bob Brown	30-04-00	bob.brown@example.com	1008	29	70	82
5	5	Charlie Davis	05-05-00	charlie.davis@example.com	1009	27	30	88
6	6	Daisy Evans	10-06-00	daisy.evans@example.com	1010	28	50	75
7	7	Edward Green	15-07-00	edward.green@example.com	1012	28	50	92
8	8	Fiona Harris	20-08-00	fiona.harris@example.com	1013	37	70	80
9	9	George Irwin	25-09-00	george.irwin@example.com	1014	20	50	77
10	10	Hannah James	30-10-00	hannah.james@example.com	1015	28	50	84
11	11	Ian Kelly	05-11-00	ian.kelly@example.com	1016	35	30	88
12	12	Jessica Lewis	10-12-00	jessica.lewis@example.com	1017	23	90	72
13	13	Kevin Moore	15-01-01	kevin.moore@example.com	1019	(null)	(null)	79
14	14	Laura Nelson	20-02-01	laura.nelson@example.com	1020	30	80	85
15	15	Michael Oliver	25-03-01	michael.oliver@example.com	1022	16	100	91
16	16	Nina Parker	30-04-01	nina.parker@example.com	1023	37	70	74
17	17	Oscar Quinn	05-05-01	oscar.quinn@example.com	1024	(null)	(null)	87
18	18	Pamela Rogers	10-06-01	pamela.rogers@example.com	1025	16	100	82
19	19	Quincy Scott	15-07-01	quincy.scott@example.com	1029	16	100	90
20	20	Rachel Turner	20-08-01	rachel.turner@example.com	1031	30	80	85
21	21	Samuel Underwood	25-09-01	samuel.underwood@example.com	1032	(null)	(null)	88
22	22	Tina Vance	30-10-01	tina.vance@example.com	1033	30	80	79
23	23	Ursula Walker	05-11-01	ursula.walker@example.com	1034	27	30	84
24	24	Victor Young	10-12-01	victor.young@example.com	1035	(null)	(null)	91
25	25	Wendy Zimmerman	15-01-02	wendy.zimmerman@example.com	1037	28	50	78
26	26	Xander Adams	20-02-02	xander.adams@example.com	1038	26	20	87
27	27	Yvonne Bailey	25-03-02	yvonne.bailey@example.com	1039	16	100	81

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:

	DEPT_ID	FACULTY_S...
1	30	(null)
2	30	69000
3	30	73000
4	30	75000
5	30	(null)
6	30	71000

Before update:

```
select dept_id,faculty_
```

ot Output × Query Result ×

SQL | All Rows Fetched: 6 in 0.058

	DEPT_ID	FACULTY_SALARY
1	30	(null)
2	30	68000
3	30	72000
4	30	74000
5	30	(null)
6	30	70000

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);*

1	2	Jane Smith	jane.smith@example.com	75000	20
2	27	Yara Phillips	yara.phillips@example.com	75000	30
3	28	Zachary Turner	zachary.turner@example.com	75000	50
4	41	Mia Taylor	mia.taylor@example.com	75000	10
5	21	Steve Roberts	steve.roberts@example.com	75000	70

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);*

	COURSE_ID	COURSE_NAME	CREDITS	SEMESTER	DEPT_ID
1	1001	Course_1	5	4	100
2	1014	Course_14	5	2	50
3	1016	Course_16	5	2	30
4	1020	Course_20	5	8	80
5	1025	Course_25	5	3	100
6	1038	Course_38	5	6	20
7	1043	Course_43	5	2	20
8	1050	Course_50	5	1	20

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;

	FACULTY_SALARY	DEPT_ID
1	75000	20
2	75000	70
3	75000	10
4	75000	30
5	75000	50

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';

	COURSE_ID	COURSE_NAME	CREDITS	SEMESTER	DEPT_ID	DEPT_ID_1	DEPT_NAME	HOD
1	1008	Course_8	4	4	70	70	ECE	SRI
2	1013	Course_13	4	3	70	70	ECE	SRI
3	1023	Course_23	(null)	5	70	70	ECE	SRI
4	1047	Course_47	(null)	8	70	70	ECE	SRI
5	1049	Course_49	3	5	70	70	ECE	SRI

Display names of faculty whose name ends with "N":

select faculty_name from faculty where faculty_name like '%n';

1	Alice Johnson
2	Bob Brown
3	Xander Thompson
4	Daniel Robinson
5	Katherine Wilson
6	Rebecca Wilson
7	Paula Nelson

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:

1	John Doe	80000
2	Jane Smith	85000
3	Alice Johnson	(null)
4	Bob Brown	75000
5	Charlie Davis	80000
7	Evan Wright	83000
8	Fiona Harris	(null)
9	George Martinez	84000

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;

1	John Doe	80000
2	Jane Smith	85000
3	Alice Johnson	10000
4	Bob Brown	75000
5	Charlie Davis	80000
7	Evan Wright	83000
8	Fiona Harris	10000
9	George Martinez	84000
11	Ian Walker	79000

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');*

	FACULTY_ID	FACULTY_NAME	FACULTY_EMAIL	FACULTY_SALARY	DEPT_ID
1	5	Charlie Davis	charlie.davis@example.com	70000	70
2	13	Kevin Young	kevin.young@example.com	71000	70
3	29	Alicia Martinez	alicia.martinez@example.com	(null)	70
4	37	Isabella Scott	isabella.scott@example.com	69000	70
5	21	Steve Roberts	steve.roberts@example.com	75000	70

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;*

2	Jane Smith	jane.smith@example.com	75000	20	Course_2
2	Jane Smith	jane.smith@example.com	75000	20	Course_38
2	Jane Smith	jane.smith@example.com	75000	20	Course_43
2	Jane Smith	jane.smith@example.com	75000	20	Course_46
2	Jane Smith	jane.smith@example.com	75000	20	Course_50
3	Alice Johnson	alice.johnson@example.com	(null)	30	Course_9
3	Alice Johnson	alice.johnson@example.com	(null)	30	Course_16
3	Alice Johnson	alice.johnson@example.com	(null)	30	Course_34
3	Alice Johnson	alice.johnson@example.com	(null)	30	Course_40
3	Alice Johnson	alice.johnson@example.com	(null)	30	Course_41
4	Bob Brown	bob.brown@example.com	65000	50	Course_10
4	Bob Brown	bob.brown@example.com	65000	50	Course_12
4	Bob Brown	bob.brown@example.com	65000	50	Course_14
4	Bob Brown	bob.brown@example.com	65000	50	Course_15
4	Bob Brown	bob.brown@example.com	65000	50	Course_37
5	Charlie Davis	charlie.davis@example.com	70000	70	Course_8
5	Charlie Davis	charlie.davis@example.com	70000	70	Course_13
5	Charlie Davis	charlie.davis@example.com	70000	70	Course_23

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;*

1	John Doe	15-01-00	john.doe@example.com	1001	8	100	85	Fiona Harris
36	Henry King	15-01-03	henry.king@example.com	1001	8	100	88	Fiona Harris
3	Alice Johnson	25-03-00	alice.johnson@example.com	1004	15	90	78	Michael Adams
38	Jack Moore	25-03-03	jack.moore@example.com	1004	15	90	82	Michael Adams
15	Michael Oliver	25-03-01	michael.oliver@example.com	1022	16	100	91	Nina Scott
18	Pamela Rogers	10-06-01	pamela.rogers@example.com	1025	16	100	82	Nina Scott
19	Quincy Scott	15-07-01	quincy.scott@example.com	1029	16	100	90	Nina Scott
27	Yvonne Bailey	25-03-02	yvonne.bailey@example.com	1039	16	100	81	Nina Scott
32	David Graham	20-08-02	david.graham@example.com	1044	16	100	78	Nina Scott
49	Ulysses Young	20-02-04	ulysses.young@example.com	1022	16	100	76	Nina Scott

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;

COURSE_NAME	DEPT_NAME
Course_2	CSE-AI
Course_50	CSE-AI

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;

	STD_NAME	LENGTH_NAME
1	John Doe	8
2	Jane Smith	10
3	Alice Johnson	13
4	Bob Brown	9
5	Charlie Davis	13
6	Daisy Evans	11
7	Edward Green	12
8	Fiona Harris	12
9	George Irwin	12
10	Hannah James	12
11	Ian Kelly	9
12	Jessica Lewis	13
13	Kevin Moore	11
14	Laura Nelson	12
15	Michael Oliver	14

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

select Upper(std_name) as capital from std;

	CAPITAL
1	JOHN DOE
2	JANE SMITH
3	ALICE JOHNSON
4	BOB BROWN
5	CHARLIE DAVIS
6	DAISY EVANS
7	EDWARD GREEN

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

select lower(std_name) as small from std;

	SMALL
1	john doe
2	jane smith
3	alice johnson
4	bob brown
5	charlie davis
6	daisy evans
7	edward green

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;

	CONCAT(STD_NAME,STD_EMAIL)
1	John Doejohn.doe@example.com
2	Jane Smithjane.smith@example.com
3	Alice Johnsonalice.johnson@example.com
4	Bob Brownbob.brown@example.com
5	Charlie Davischarlie.davis@example.com
6	Daisy Evansdaisy.evans@example.com
7	Edward Greenedward.green@example.com
8	Fiona Harrisfiona.harris@example.com
9	George Irwingeorge.irwin@example.com
10	Hannah Jameshannah.james@example.com
11	Ian Kellyian.kelly@example.com

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

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

<u>Royal ChallengesBangalore</u>

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;

	SUBSTR(FACULTY_NAME,4,5)
4	Brow
5	rlie
6	n Wri
7	na Ha
8	rge M
9	Walk
10	mine
11	in Yo
12	ra Ki
13	hael
14	a Sco
15	ar Ba
16	ncy C

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';

STD_NAME
Daisy Evans
David Graham

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;

FACULTY_NAME	REPLACE(FACULTY_NAME,SU
John Doe	John Do$
Jane Smith	Jane Smit$
Alice Johnson	Alice Johso
Bob Brown	Bob Brow$
Charlie Davis	Charlie Davi$
Evan Wright	Evan Wrigh$
Fiona Harris	Fiona Harri$
George Martinez	George Martine$

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;

STD_ID	STD_NAME	STD_MARKS	GRADE
1	John Doe	85	B
2	Jane Smith	90	A
3	Alice Johnson	78	D
4	Bob Brown	82	C
5	Charlie Davis	88	B
6	Daisy Evans	75	D
7	Edward Green	92	A
8	Fiona Harris	80	C
9	George Irwin	77	D
10	Hannah James	84	C
11	Ian Kelly	88	B
12	Jessica Lewis	72	E
13	Kevin Moore	79	D
14	Laura Nelson	85	B
15	Michael Oliver	91	A
16	Nina Parker	74	E
17	Oscar Quinn	87	B

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;
```

All Rows Fetched: 48 in 0.007 seconds

	FACULTY_ID	FACULTY_NAME	FACULTY_SALARY	HIKE
13	5	Charlie Davis	70000	14000
14	43	Olivia Martinez	71000	17040
15	13	Kevin Young	71000	17040
16	32	Daniel Robinson	71000	17040
17	44	Paul Harris	72000	17280
18	33	Emily Walker	72000	17280
19	6	Deborah Clark	72000	17280
20	25	Wendy Davis	73000	17520
21	7	Evan Wright	73000	17520
22	34	Felix Harris	73000	17520
23	19	Quincy Carter	73000	17520
24	39	Katherine Wilson	74000	17760
25	16	Nina Scott	74000	17760
26	9	George Martinez	74000	17760
27	28	Zachary Turner	75000	18000
28	27	Yara Phillips	75000	18000
29	2	Jane Smith	75000	18000
30	21	Steve Roberts	75000	18000
31	41	Mia Taylor	75000	18000
32	26	Xander Thompson	(null)	30000
33	31	Christina Jones	(null)	30000
34	24	Victor White	(null)	30000