CREATE VIEW hire\_student\_job\_company AS SELECT \* FROM hires NATURAL JOIN student NATURAL JOIN company NATURAL JOIN job;

**By Sounak**

1) SELECT DISTINCT companyname FROM COMPANY NATURAL JOIN JOB WHERE ToDate<'01-06-20115' AND FromDate>'01-06-2014';

2) SELECT \* FROM job WHERE jobofferid IN (SELECT jobofferid FROM COMPANY NATURAL JOIN JOB WHERE ToDate<'06-01-2014' AND FromDate>'06-01-2013' AND companyname='FLIPKART');

3) SELECT studentid,fname FROM student NATURAL JOIN program WHERE batch='2014' AND ProgramName='MSC-IT' AND CPI>7;

4) (SELECT StudentID,Fname FROM student NATURAL JOIN Program NATURAL JOIN registers WHERE Batch='2014' AND ProgramName='MSC-IT' AND (Grade='AA' OR Grade='AB'))

EXCEPT

(SELECT StudentID,Fname FROM student NATURAL JOIN Program NATURAL JOIN registers WHERE Batch='2014' AND ProgramName='MSC-IT' AND (Grade<>'AA' AND Grade<>'AB'))

7) SELECT count(\*) FROM participates JOIN Round NATURAL JOIN Company NATURAL JOIN Job ON participates.jobofferid=job.jobofferid WHERE round.Roundname='APTITUDE' AND companyname='FLIPKART';

8) SELECT studentID,FName FROM Student

WHERE studentID

IN

(SELECT applicantid FROM participates WHERE JobOfferID='FLI01\_201402' AND

applicantid

NOT IN

(SELECT applicantid From

((SELECT ApplicantID,roundname FROM participates NATURAL JOIN Round WHERE JobOfferID='FLI01\_201402')

EXCEPT

(SELECT ApplicantID,roundname FROM Participates WHERE Status='t' AND JobOfferID='FLI01\_201402')) AS r

)

)

;

;

9) SELECT studentID,FName FROM Student

WHERE studentID

IN

(SELECT applicantid From

((SELECT DISTINCT ApplicantID FROM participates WHERE JobOfferID='FLI01\_201402')

EXCEPT

(SELECT DISTINCT ApplicantID FROM Participates WHERE Status='t' AND JobOfferID='FLI01\_201402')) AS r

)

;

10) SELECT \* FROM JOB WHERE CompanyID='FLI01' ORDER BY Packageamount DESC LIMIT 10;

11) SELECT StudentID,FName FROM Student WHERE StudentID IN

(SELECT ApplicantID FROM hires WHERE HireDate>'01-06-2014' AND HireDate<'01-06-2016' AND

NegotiatedPackage=(SELECT max(NegotiatedPackage) FROM hires WHERE HireDate>'01-06-2014' AND HireDate<'01-06-2016'))

;

12) SELECT companyname FROM COMPANY WHERE companyID IN

(SELECT CompanyID FROM JOB WHERE ToDate<'2015-10-25' AND FromDate>'2014-01-05'

AND

Packageamount>(SELECT avg(Packageamount) FROM JOB WHERE ToDate<'2015-10-25' AND FromDate>'2014-01-05'))

;

13)select \* from hires join student on hires.applicantid=student.studentid where batch='2014' and programid=12; (Edited By Lokesh)

**BY MOIZ**

-- 1 --

SELECT \* FROM student WHERE studentid IN(SELECT applicantid FROM applicant);

-- 2 --

SELECT applicantid FROM qualification WHERE applicantid IN (SELECT applicantid FROM applicationyear WHERE applicationyear=2015) AND qualifyingexam='B.C.A.';

-- 3 --

SELECT \* FROM job WHERE fromdate > '6-1-2014' AND todate <= '6-1-2015' AND companyid=(SELECT companyid FROM company WHERE companyname='FLIPKART');

-- 4 --

SELECT \* FROM job WHERE companyid=(SELECT companyid FROM company WHERE companyname='FLIPKART') AND todate < '12-31-2013';

-- 5 --

SELECT \* FROM round WHERE rounddate='02-05-2015' AND jobofferid='FLI01\_201402' ORDER BY roundorder;

-- 6 --

SELECT \* FROM company NATURAL JOIN

(

SELECT DISTINCT companyid, packageamount, offeryear FROM job JOIN

(

SELECT MAX(packageamount)AS max\_packageamount, MIN(packageamount) AS min\_packageamount, calc\_acad\_year(fromdate) AS offeryear FROM job GROUP BY calc\_acad\_year(fromdate)

) AS x

ON

calc\_acad\_year(fromdate)=x.offeryear

AND (job.packageamount=x.max\_packageamount OR job.packageamount=x.min\_packageamount)

) AS y

ORDER BY offeryear, packageamount

-- 7 --

SELECT companyname, max(packageamount), offeryear FROM company NATURAL JOIN

(SELECT companyid, packageamount, calc\_acad\_year(fromdate) AS offeryear

FROM job

WHERE calc\_acad\_year(fromdate)='2013-2014' AND packageamount > 1000000) X

GROUP BY companyname, offeryear

-- 8 --

SELECT applicationid FROM application

WHERE jobofferid IN

(SELECT jobofferid FROM hire\_student\_job\_company WHERE calc\_acad\_year(fromdate)='2013-2014')

EXCEPT

SELECT applicantid FROM hire\_student\_job\_company WHERE calc\_acad\_year(fromdate)='2013-2014';

-- 9 --

SELECT \* FROM placementcell NATURAL JOIN (SELECT \* FROM coordinates WHERE roundname='TECHNICAL' AND jobofferid='FLI01\_201401') AS x;

-- 10 --

SELECT \* FROM participates WHERE jobofferid='FLI01\_201402' AND (applicantid=201412039 AND status=false) OR (applicantid=2 AND status=true)

-- 11 --

SELECT \* FROM increasingcompany();

-- 12 --

SELECT StudentID,CPI,year FROM student JOIN

(

SELECT DISTINCT ApplicantID, NegotiatedPackage, year FROM hires

JOIN (

SELECT MAX(NegotiatedPackage)AS max\_package, MIN(NegotiatedPackage) AS min\_package, calc\_acad\_year(hiredate) AS year FROM hires GROUP BY calc\_acad\_year(hiredate)

) AS x

ON

calc\_acad\_year(hiredate)=x.year

AND (hires.NegotiatedPackage=x.max\_package OR hires.NegotiatedPackage=x.min\_package)

) AS y

ON StudentID=ApplicantID

ORDER BY year, CPI

-- 13 --

SELECT year FROM hires WHERE ApplicantID IN

(

)

-- 14 --

(Select \* FROM Hires JOIN Student JOIN Program ON Hires.ApplicantID=Student.StudentID AND Program.ProgramID=Student.ProgramID WHERE ProgramName='MSc.(IT)')

-- 15 --

SELECT CompanyID,CompanyName From Company WHERE CompanyID IN

(

(Select \* FROM Hires JOIN Student JOIN Program ON Hires.ApplicantID=Student.StudentID AND Program.ProgramID=Student.ProgramID WHERE ProgramName='MTech.(VLSI)')

EXCEPT

(Select \* FROM Hires JOIN Student JOIN Program ON Hires.ApplicantID=Student.StudentID AND Program.ProgramID=Student.ProgramID WHERE ProgramName<>'MTech.(VLSI)')

)

-- Trigger to add row automatically upon successful registration in a course by a student

-- The row added in result will have spi as NULL since updating SPI will be a manual process.

CREATE TRIGGER register\_result AFTER INSERT OR UPDATE OR DELETE ON registers FOR EACH ROW

EXECUTE PROCEDURE trigger\_result();

CREATE OR REPLACE FUNCTION trigger\_result() RETURNS TRIGGER AS $registers$

DECLARE count\_var\_registers INTEGER;

DECLARE count\_var\_result INTEGER;

BEGIN

IF (TG\_OP = 'DELETE') THEN

SELECT INTO count\_var\_registers count(\*) FROM registers WHERE studentid=old.studentid AND acadyear=old.acadyear AND season=old.season;

IF (count\_var\_registers = 0) THEN

DELETE FROM result WHERE studentid=old.studentid AND academicyear=old.acadyear AND season=old.season;

END IF;

RETURN OLD;

ELSIF (TG\_OP = 'UPDATE') THEN

SELECT INTO count\_var\_result count(\*) FROM result WHERE studentid=new.studentid AND academicyear=new.acadyear AND season=new.season;

SELECT INTO count\_var\_registers count(\*) FROM registers WHERE studentid=old.studentid AND acadyear=old.acadyear AND season=old.season;

IF (count\_var\_result = 0) THEN

INSERT INTO result (studentid, academicyear,season) VALUES(new.studentid, new.acadyear,new.season);

IF (count\_var\_registers = 0) THEN

DELETE FROM result WHERE studentid=OLD.studentid AND academicyear=OLD.acadyear AND season=OLD.season;

END IF;

END IF;

RETURN NEW;

ELSIF (TG\_OP = 'INSERT') THEN

SELECT INTO count\_var\_result count(\*) FROM result WHERE studentid=new.studentid AND academicyear=new.acadyear AND season=new.season;

IF(count\_var\_result=0) THEN

INSERT INTO result (studentid, academicyear, season) VALUES(new.studentid, new.acadyear, new.season);

RETURN NEW;

END IF;

END IF;

RETURN NULL;

END;

$registers$ LANGUAGE plpgsql;

-- Calculate CPI and update it whenever there is a change in SPI of result

DROP FUNCTION trigger\_cpi\_count() CASCADE;

CREATE OR REPLACE FUNCTION trigger\_cpi\_count() RETURNS TRIGGER AS $cpi$

DECLARE sum REAL;

DECLARE count REAL;

BEGIN

SELECT INTO sum cast(SUM(SPI)/COUNT(SPI) as REAL) FROM result WHERE studentid=NEW.studentid;

UPDATE student SET cpi = sum WHERE studentid=NEW.studentid;

RETURN NEW;

END;

$cpi$ LANGUAGE plpgsql;

CREATE TRIGGER result\_student AFTER INSERT OR UPDATE ON result FOR EACH ROW

EXECUTE PROCEDURE trigger\_cpi\_count();

-- Function to get the acad\_year from date

-- $1=FROM

CREATE OR REPLACE FUNCTION calc\_acad\_year(DATE) RETURNS TEXT AS $acad\_year$

BEGIN

IF(EXTRACT (MONTH FROM $1)>='6') THEN

RETURN EXTRACT (YEAR FROM $1) || '-' || EXTRACT (YEAR FROM $1)+1;

ELSE

RETURN EXTRACT (YEAR FROM $1)-1 || '-' || EXTRACT (YEAR FROM $1);

END IF;

END;

$acad\_year$ LANGUAGE plpgsql;