**Q1.1：**

SELECT

tc.constraint\_name,

tc.table\_name,

tc.constraint\_type,

pg\_get\_constraintdef(pc.oid) AS constraint\_definition

FROM

information\_schema.table\_constraints AS tc

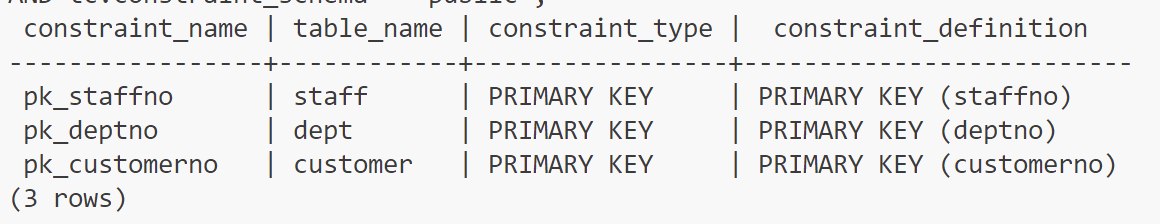
JOIN

pg\_constraint AS pc ON tc.constraint\_name = pc.conname

WHERE

tc.table\_name IN ('staff', 'dept', 'sales', 'customer')

AND tc.constraint\_schema = 'public';



**Q1.2:**

ALTER TABLE SALES ADD CONSTRAINT PK\_SALENO PRIMARY KEY (SaleNo);

ALTER TABLE DEPT ADD CONSTRAINT UN\_DNAME UNIQUE(DName);

ALTER TABLE STAFF ADD CONSTRAINT CK\_STAFFNAME CHECK (StaffName IS NOT NULL);

ALTER TABLE DEPT ADD CONSTRAINT CK\_DNAME CHECK (DName IS NOT NULL);

ALTER TABLE CUSTOMER ADD CONSTRAINT CK\_CNAME CHECK (CName IS NOT NULL);

ALTER TABLE SALES ADD CONSTRAINT CK\_RECEIPTNO CHECK (ReceiptNo IS NOT NULL);

ALTER TABLE SALES ADD CONSTRAINT CK\_AMOUNT CHECK (Amount>0);

ALTER TABLE STAFF ADD CONSTRAINT CK\_POSITION

CHECK (Position IN ('Group Manager','Group Assistant', 'Group Member','Team Leader', 'Branch Manager'));

ALTER TABLE SALES ADD CONSTRAINT CK\_SERCIVETYPE

CHECK (ServiceType IN ('Software Installation','Software Repair', 'Training','Consultation','Data Recovery'));

ALTER TABLE SALES ADD CONSTRAINT CK\_PAYMENTTYPE

CHECK (PaymentType IN ('Debit', 'Cash', 'Credit'));

ALTER TABLE SALES ADD CONSTRAINT CK\_GST

CHECK (GST IN ('Yes','No'));

ALTER TABLE STAFF ADD CONSTRAINT FK\_DEPTNO

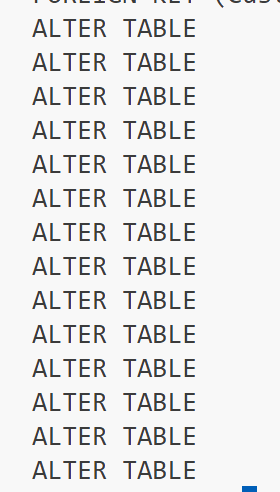
FOREIGN KEY (DeptNo) REFERENCES DEPT(DeptNo);

ALTER TABLE SALES ADD CONSTRAINT FK\_STAFFNO

FOREIGN KEY (ServedBy) REFERENCES STAFF(StaffNo);

ALTER TABLE SALES ADD CONSTRAINT FK\_CUSTOMERNO

FOREIGN KEY (CustomerNo) REFERENCES CUSTOMER(CustomerNo);



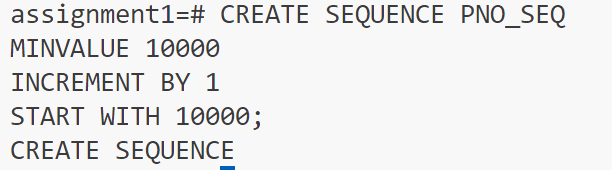
**Q2.1**

CREATE SEQUENCE PNO\_SEQ

MINVALUE 10000

INCREMENT BY 1

START WITH 10000;



**Q2.2**

CREATE OR REPLACE FUNCTION

UDF\_BI\_PNO()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.SaleNo IS NULL THEN

NEW.SaleNo := nextval('PNO\_SEQ');

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

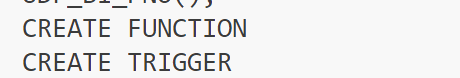
CREATE TRIGGER BI\_PNO

BEFORE INSERT ON SALES

FOR EACH ROW

EXECUTE FUNCTION

UDF\_BI\_PNO();



**Q2.3**

CREATE OR REPLACE FUNCTION UDF\_TOP\_DISCOUNT()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.customerNo IN (

SELECT customerNo FROM SALES

GROUP BY CustomerNo

HAVING SUM(amount)=(

SELECT MAX(Total) FROM

(

SELECT SUM(amount) AS Total FROM SALES

GROUP BY CustomerNo) total

)

)

THEN NEW.amount := NEW.amount \* 0.85;

END IF;

RETURN NEW;

END;

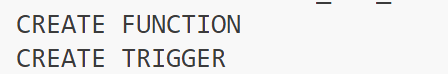
$$ LANGUAGE plpgsql;

CREATE TRIGGER TOP\_DISCOUNT

BEFORE INSERT ON SALES

FOR EACH ROW

EXECUTE FUNCTION UDF\_TOP\_DISCOUNT();



**Q2.4**

CREATE OR REPLACE FUNCTION UDF\_SUNSHINE\_DEPT()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.servedby IN (

SELECT s.staffno FROM STAFF s

JOIN DEPT d ON d.deptno = s.deptno

WHERE d.dname = 'SALES - Sunshine'

)

THEN NEW.paymenttype :='Cash';

IF NEW.servicetype='Data Recovery' THEN

NEW.amount:=NEW.amount\*0.7;

END IF;

END IF;

RETURN NEW;

END;

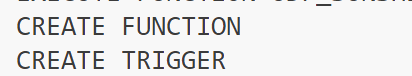
$$ LANGUAGE plpgsql;

CREATE TRIGGER SUNSHINE\_DEPT

BEFORE INSERT ON SALES

FOR EACH ROW

EXECUTE FUNCTION UDF\_SUNSHINE\_DEPT();



**Q2.5**

CREATE OR REPLACE FUNCTION UDF\_TIME\_CHECK()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.customerno IN(

SELECT s.customerno FROM SALES s

JOIN STAFF st ON st.staffno = s.servedby

JOIN DEPT d ON d.deptno=st.deptno

WHERE d.dlocation NOT IN(

SELECT d2.dlocation FROM STAFF st2

JOIN DEPT d2 ON d2.deptno = st2.deptno

WHERE st2.staffno = NEW.servedby)

AND s.saletime >= NEW.saletime - INTERVAL '5 minutes'

AND s.saletime<NEW.saletime)

THEN RAISE EXCEPTION 'Invalid';

END IF;

IF (CAST(NEW.saletime AS time) < TIME '06:00' OR CAST(NEW.saletime AS time) > TIME '21:00')

AND NEW.servedby NOT IN (

SELECT st.staffno FROM STAFF st

WHERE st.position ILIKE '%manager%')

THEN RAISE EXCEPTION 'Invalid';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER TIME\_CHECK

BEFORE INSERT ON sales

FOR EACH ROW

EXECUTE FUNCTION UDF\_TIME\_CHECK();

