EXPERIMENT - V
DATA CONSTRAINTS AND VIEWS

August 8, 2019

ADITHYA D RAJAGOPAL
ROLL NO : 9
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM

AIM

To study about various data constraints and views in SQL.

Using Referential Integrity Constraints

Whenever two tables are related by a common column (or set of columns), define a PRIMARY or UNIQUE key constraint on the column in the parent table, and define a FOREIGNKEY constraint on the column in the child table, to maintain the relationship between the two tables.

Foreign keys can be comprised of multiple columns. However, a composite foreign key must reference a composite primary or unique key of the exact same structure (the same number of columns and datatypes). Because composite primary and unique keys are limited to 16 columns, a composite foreign key is also limited to 16 columns.

NULLS AND FOREIGN KEYS

By default (without any NOT NULL or CHECK clauses), and in accordance with the ANSI/ISO standard, the FOREIGNKEY constraint enforces the "match none" rule for composite foreign keys. The "full" and "partial" rules can also be enforced by using CHECK and NOTNULL constraints, as follows:

1. To enforce the "match full" rule for nulls in composite foreign keys, which requires that all components of the key be null or all be non-null, define a CHECK constraint that allows only all nulls or all non-nulls in the composite foreign key as follows, assuming a composite key comprised of columns A, B, and C:
   CHECK ((A IS NULL AND B IS NULL AND C IS NULL) OR
   (A IS NOT NULL AND B IS NOT NULL AND C IS NOT NULL))

2. In general, it is not possible to use declarative referential integrity to enforce the "match partial" rule for nulls in composite foreign keys, which requires the non-null portions of the key to appear in the corresponding portions in the primary or unique key of a single row in the referenced table. You can often use triggers to handle this case.

Relationships Between Parent and Child Tables

Several relationships between parent and child tables can be determined by the other types of integrity constraints defined on the foreign key in the child table.

No Constraints on the Foreign Key

When no other constraints are defined on the foreign key, any number of rows in the child table can reference the same parent key value. This model allows nulls in the foreign key.
This model establishes a "one-to-many" relationship between the parent and foreign keys that allows undetermined values (nulls) in the foreign key.

NOT NULL Constraint on the Foreign Key

When nulls are not allowed in a foreign key, each row in the child table must explicitly reference a value in the parent key because nulls are not allowed in the foreign key. However, any number of rows in the child table can reference the same parent key value.
This model establishes a "one-to-many" relationship between the parent and foreign keys. However, each row in the child table must have a reference to a parent key value; the absence of a value (a null) in the foreign key is not allowed. The same example in the previous section can be used to illustrate such a relationship. However, in this case, employees must have a reference to a specific department.

UNIQUE Constraint on the Foreign Key

When a UNIQUE constraint is defined on the foreign key, one row in the child table can reference a parent key value. This model allows nulls in the foreign key.

This model establishes a "one-to-one" relationship between the parent and foreign keys that allows undetermined values (nulls) in the foreign key.

UNIQUE and NOT NULL Constraints on the Foreign Key

When both UNIQUE and NOTNULL constraints are defined on the foreign key, only one row in the child table can reference a parent key value. Because nulls are not allowed in the foreign key, each row in the child table must explicitly reference a value in the parent key.
This model establishes a "one-to-one" relationship between the parent and foreign keys that does not allow undetermined values (nulls) in the foreign key.

VIEWS

Views in SQL are considered as a virtual table. A view also contains rows and columns. To create the view, we can select the fields from one or more tables present in the database. A view can either have specific rows based on certain condition or all the rows of a table.

Creating a View

```
CREATE VIEW <View_Name> AS
SELECT column1,column2,...
FROM <Table_Name>
WHERE [condition];
```

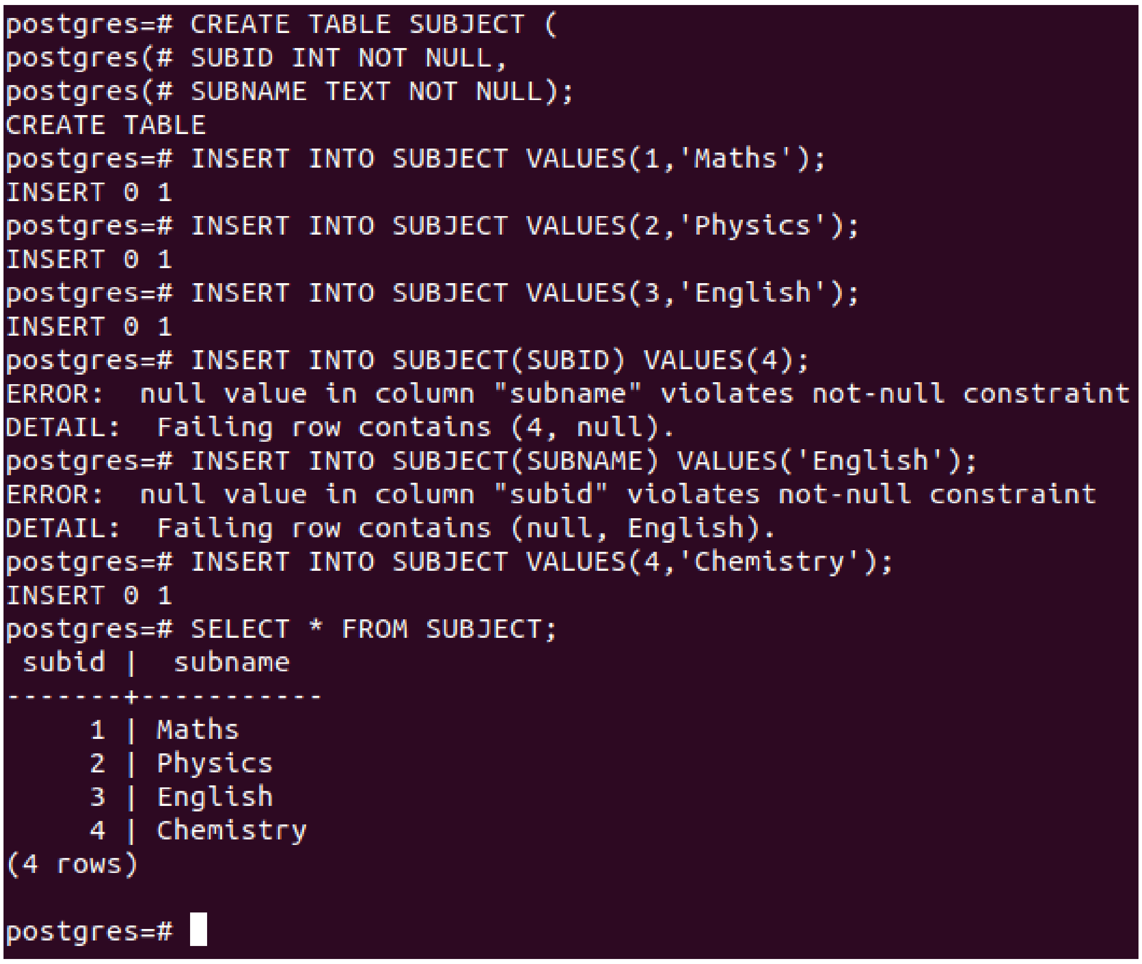
Deleting a View

```
DROP VIEW <View_Name>
```

QUESTIONS

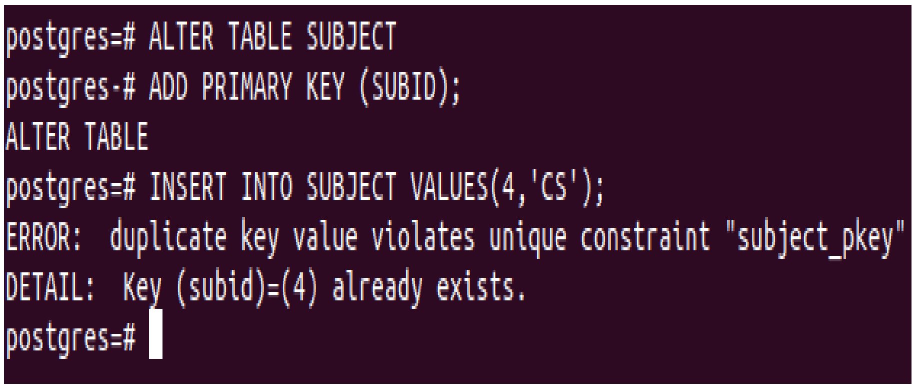
1. Create the following tables with given constraints:

   (a) Create a table named Subjects with the given attributes
   Subid(Should not be NULL)
   Subname (Should not be NULL)
   Populate the database. Make sure that all constraints are working properly.



```
postgres=# CREATE TABLE SUBJECT (
postgres(# SUBID INT NOT NULL,
postgres(# SUBNAME TEXT NOT NULL);
CREATE TABLE
postgres=# INSERT INTO SUBJECT VALUES(1,'Maths');
INSERT 0 1
postgres=# INSERT INTO SUBJECT VALUES(2,'Physics');
INSERT 0 1
postgres=# INSERT INTO SUBJECT VALUES(3,'English');
INSERT 0 1
postgres=# INSERT INTO SUBJECT(SUBID) VALUES(4);
ERROR:  null value in column "subname" violates not-null constraint
DETAIL:  Failing row contains (4, null).
postgres=# INSERT INTO SUBJECT(SUBNAME) VALUES('English');
ERROR:  null value in column "subid" violates not-null constraint
DETAIL:  Failing row contains (null, English).
postgres=# INSERT INTO SUBJECT VALUES(4,'Chemistry');
INSERT 0 1
postgres=# SELECT * FROM SUBJECT;
 subid |  subname
-------+-----------
     1 | Maths
     2 | Physics
     3 | English
     4 | Chemistry
(4 rows)

postgres=#
```

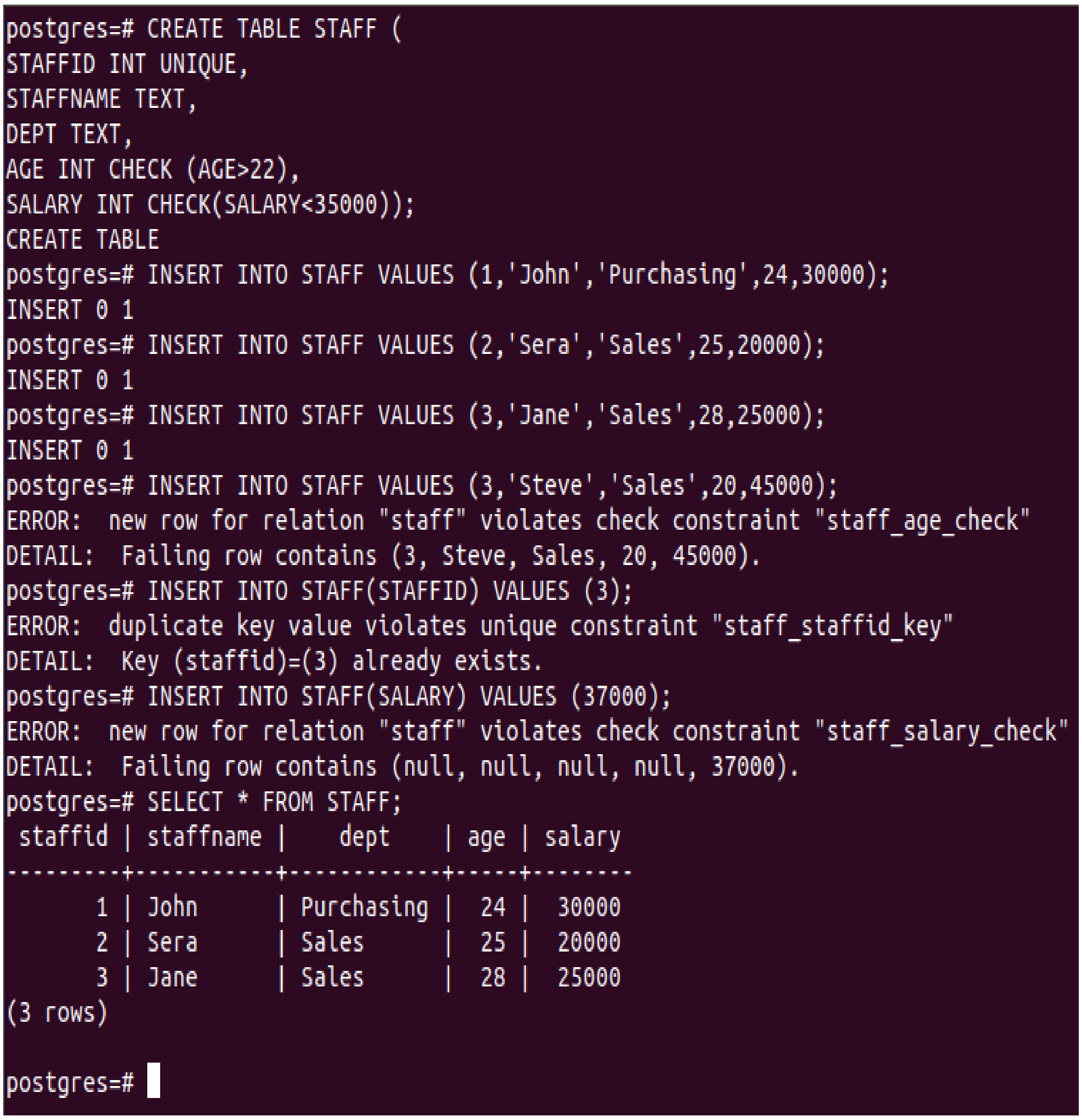
i. Alter the table to set subid as the primary key.



```
postgres=# ALTER TABLE SUBJECT
postgres-# ADD PRIMARY KEY (SUBID);
ALTER TABLE
postgres=# INSERT INTO SUBJECT VALUES(4,'CS');
ERROR:  duplicate key value violates unique constraint "subject_pkey"
DETAIL:  Key (subid)=(4) already exists.
postgres=# 
```

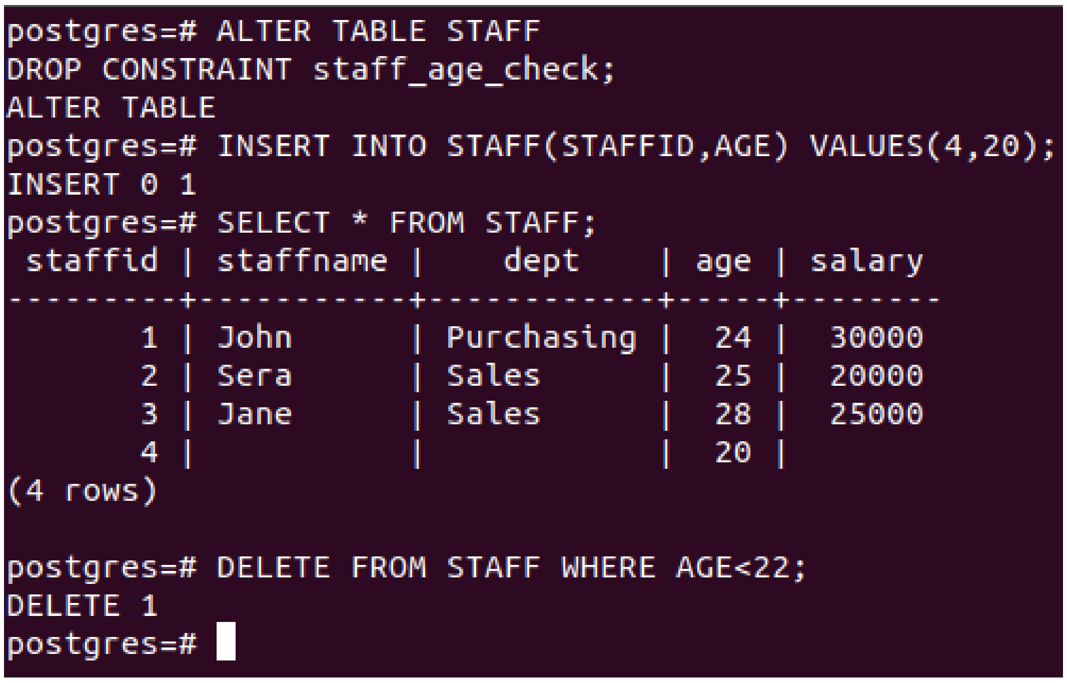
(b) Create a table named Staff with the given attributes
staffid (Should be UNIQUE)
staffname
dept
Age (Greater than 22)
Salary (Less than 35000)

Populate the database. Make sure that all constraints are working properly.



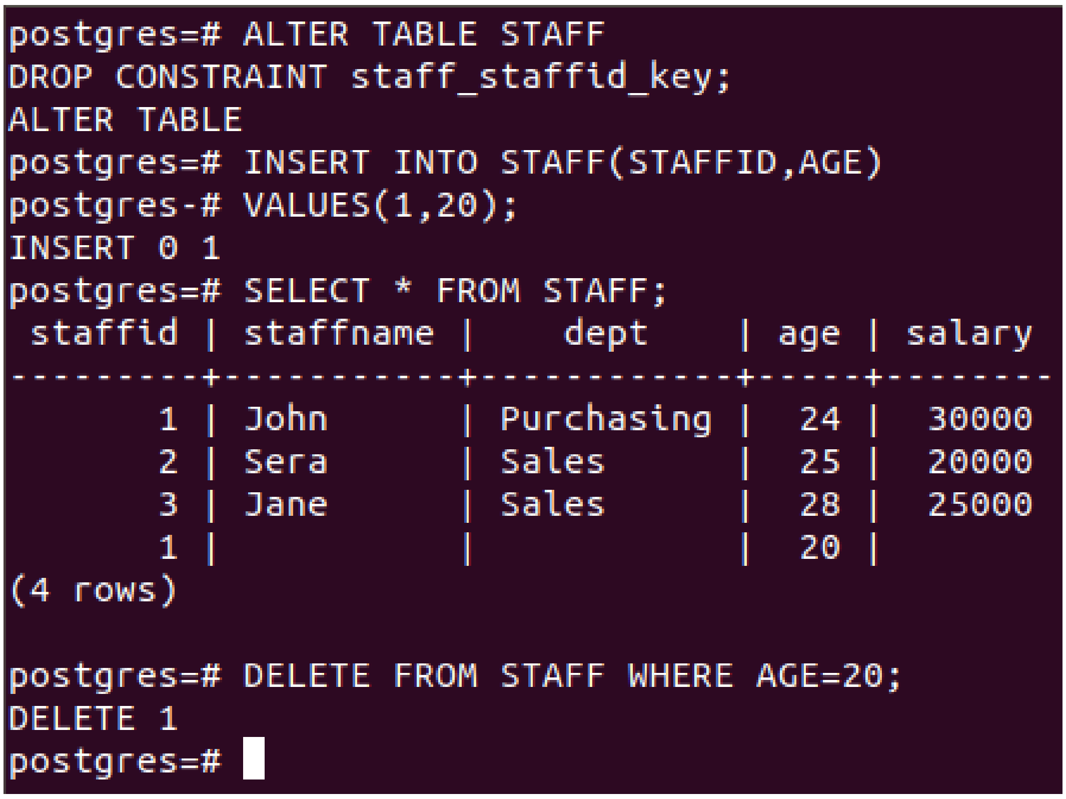
```
postgres=# CREATE TABLE STAFF (
STAFFID INT UNIQUE,
STAFFNAME TEXT,
DEPT TEXT,
AGE INT CHECK (AGE>22),
SALARY INT CHECK(SALARY<35000));
CREATE TABLE
postgres=# INSERT INTO STAFF VALUES (1,'John','Purchasing',24,30000);
INSERT 0 1
postgres=# INSERT INTO STAFF VALUES (2,'Sera','Sales',25,20000);
INSERT 0 1
postgres=# INSERT INTO STAFF VALUES (3,'Jane','Sales',28,25000);
INSERT 0 1
postgres=# INSERT INTO STAFF VALUES (3,'Steve','Sales',20,45000);
ERROR:  new row for relation "staff" violates check constraint "staff_age_check"
DETAIL:  Failing row contains (3, Steve, Sales, 20, 45000).
postgres=# INSERT INTO STAFF(STAFFID) VALUES (3);
ERROR:  duplicate key value violates unique constraint "staff_staffid_key"
DETAIL:  Key (staffid)=(3) already exists.
postgres=# INSERT INTO STAFF(SALARY) VALUES (37000);
ERROR:  new row for relation "staff" violates check constraint "staff_salary_check"
DETAIL:  Failing row contains (null, null, null, null, 37000).
postgres=# SELECT * FROM STAFF;
 staffid | staffname |    dept    | age | salary
---------+-----------+------------+-----+--------
       1 | John      | Purchasing |  24 |  30000
       2 | Sera      | Sales      |  25 |  20000
       3 | Jane      | Sales      |  28 |  25000
(3 rows)

postgres=# 
```

i. Delete the check constraint imposed on the attribute salary



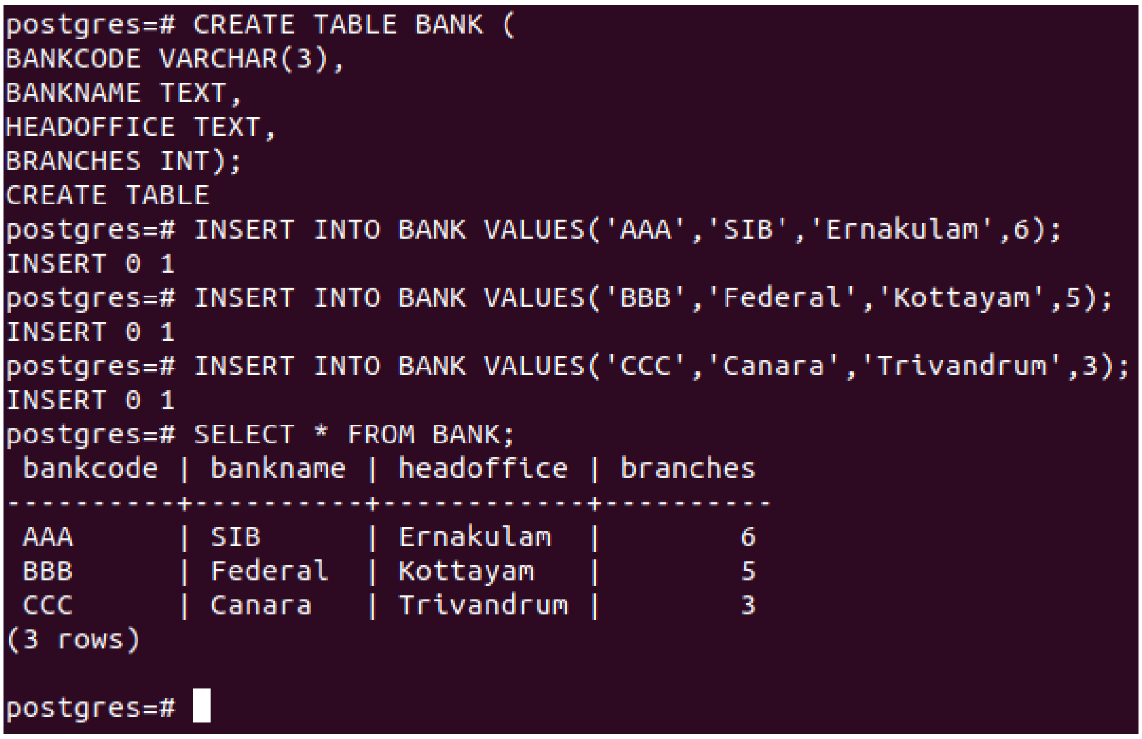
```
postgres=# ALTER TABLE STAFF
DROP CONSTRAINT staff_age_check;
ALTER TABLE
postgres=# INSERT INTO STAFF(STAFFID,AGE) VALUES(4,20);
INSERT 0 1
postgres=# SELECT * FROM STAFF;
 staffid | staffname |    dept    | age | salary
---------+-----------+------------+-----+--------
       1 | John      | Purchasing |  24 |  30000
       2 | Sera      | Sales      |  25 |  20000
       3 | Jane      | Sales      |  28 |  25000
       4 |           |            |  20 |
(4 rows)

postgres=# DELETE FROM STAFF WHERE AGE<22;
DELETE 1
postgres=# 
```

ii. Delete the unique constraint on the attribute staffid

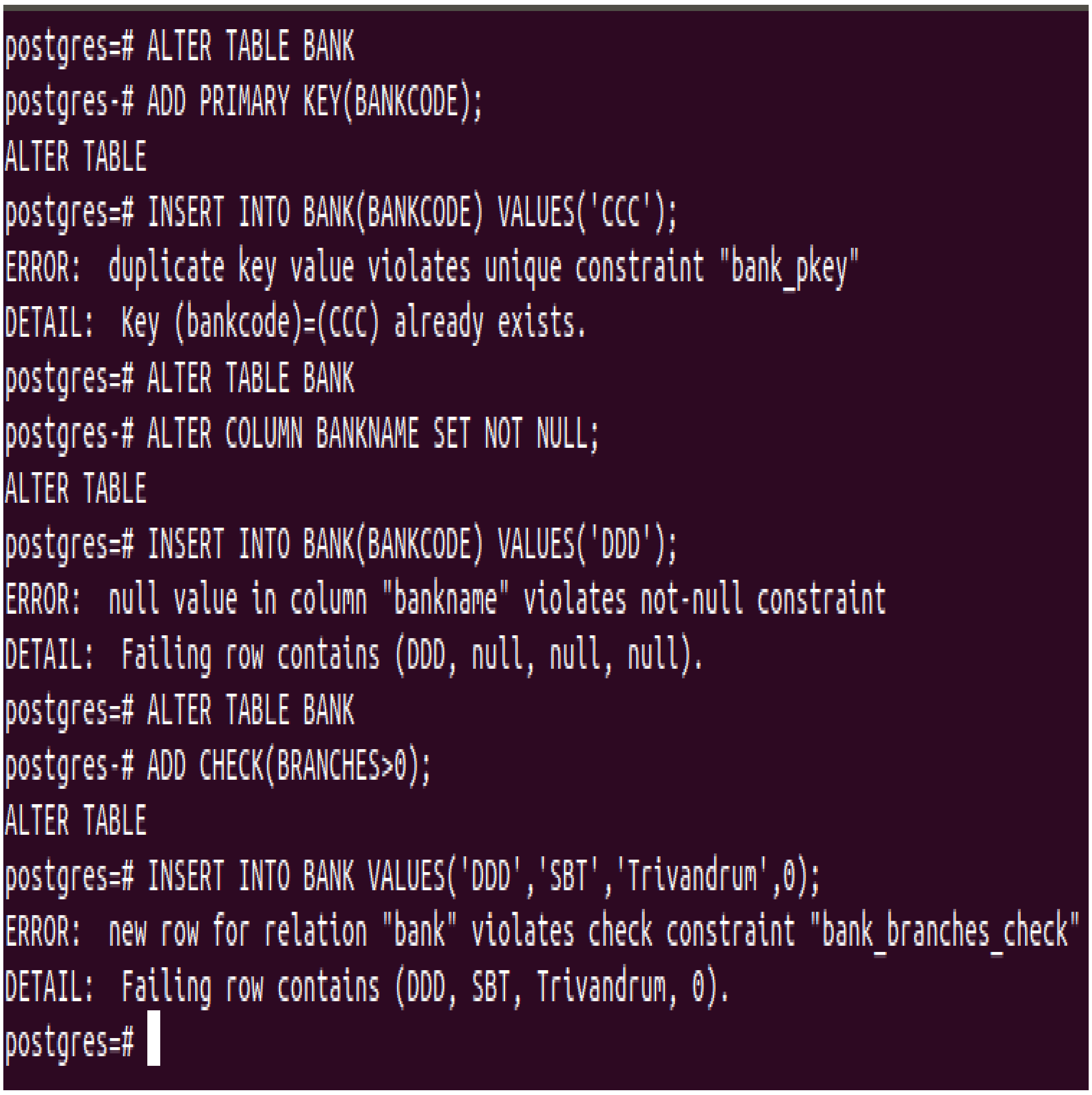


```
postgres=# ALTER TABLE STAFF
DROP CONSTRAINT staff_staffid_key;
ALTER TABLE
postgres=# INSERT INTO STAFF(STAFFID,AGE)
postgres-# VALUES(1,20);
INSERT 0 1
postgres=# SELECT * FROM STAFF;
 staffid | staffname |    dept    | age | salary
---------+-----------+------------+-----+--------
       1 | John      | Purchasing |  24 |  30000
       2 | Sera      | Sales      |  25 |  20000
       3 | Jane      | Sales      |  28 |  25000
       1 |           |            |  20 |
(4 rows)

postgres=# DELETE FROM STAFF WHERE AGE=20;
DELETE 1
postgres=# 
```

(c) Create a table named Bank with the following attributes
bankcode (To be set as Primary Key, type= varchar(3))
bankname (Should not be NULL)
headoffice
branches (Integer value greater than Zero)
Populate the database. Make sure that all constraints are working properly.All constraints have to be set after creating the table.



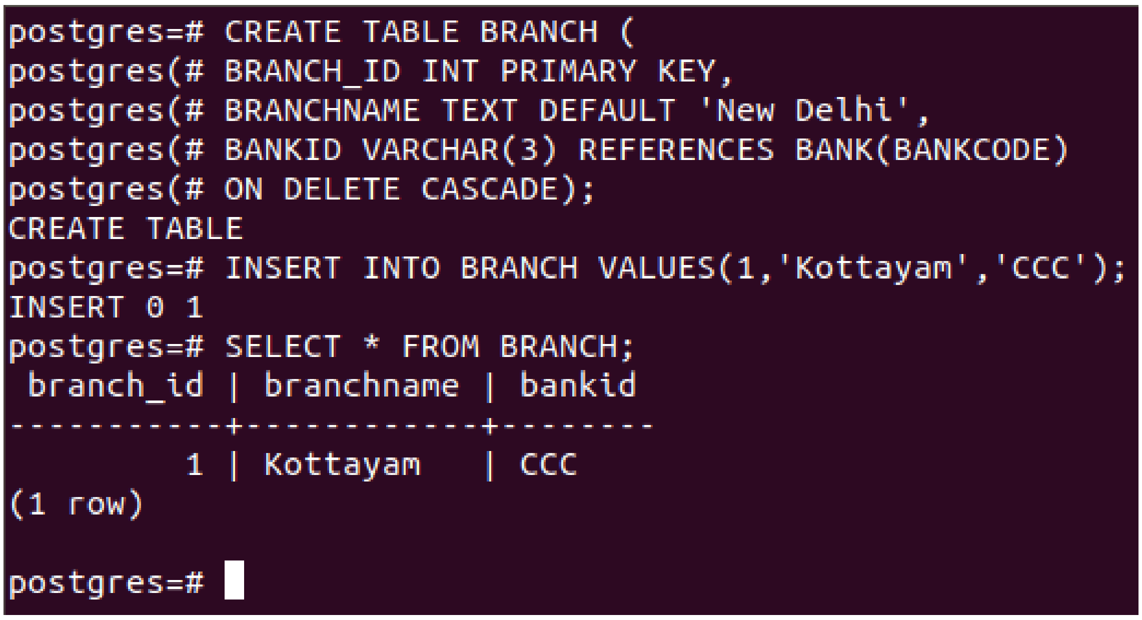
```
postgres=# CREATE TABLE BANK (
BANKCODE VARCHAR(3),
BANKNAME TEXT,
HEADOFFICE TEXT,
BRANCHES INT);
CREATE TABLE
postgres=# INSERT INTO BANK VALUES('AAA','SIB','Ernakulam',6);
INSERT 0 1
postgres=# INSERT INTO BANK VALUES('BBB','Federal','Kottayam',5);
INSERT 0 1
postgres=# INSERT INTO BANK VALUES('CCC','Canara','Trivandrum',3);
INSERT 0 1
postgres=# SELECT * FROM BANK;
 bankcode | bankname | headoffice | branches
----------+----------+------------+----------
 AAA      | SIB      | Ernakulam  |        6
 BBB      | Federal  | Kottayam   |        5
 CCC      | Canara   | Trivandrum |        3
(3 rows)

postgres=# 
```



```
postgres=# ALTER TABLE BANK
postgres-# ADD PRIMARY KEY(BANKCODE);
ALTER TABLE
postgres=# INSERT INTO BANK(BANKCODE) VALUES('CCC');
ERROR:  duplicate key value violates unique constraint "bank_pkey"
DETAIL:  Key (bankcode)=(CCC) already exists.
postgres=# ALTER TABLE BANK
postgres-# ALTER COLUMN BANKNAME SET NOT NULL;
ALTER TABLE
postgres=# INSERT INTO BANK(BANKCODE) VALUES('DDD');
ERROR:  null value in column "bankname" violates not-null constraint
DETAIL:  Failing row contains (DDD, null, null, null).
postgres=# ALTER TABLE BANK
postgres-# ADD CHECK(BRANCHES>0);
ALTER TABLE
postgres=# INSERT INTO BANK VALUES('DDD','SBT','Trivandrum',0);
ERROR:  new row for relation "bank" violates check constraint "bank_branches_check"
DETAIL:  Failing row contains (DDD, SBT, Trivandrum, 0).
postgres=# 
```

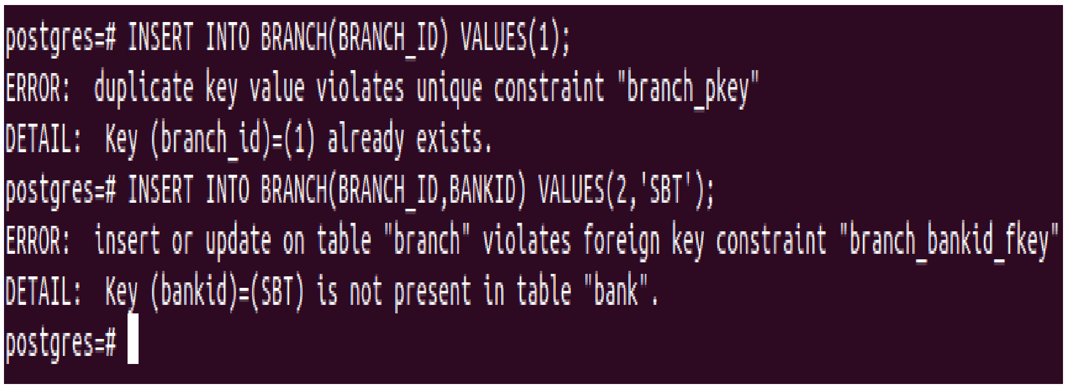
(d) Create a table named Branch with the following attributes

branchid (To be set as Primary Key)
branchname (Set Default value as ’New Delhi’)
bankid (Foreign Key:- Refers to bank code of Bank table)



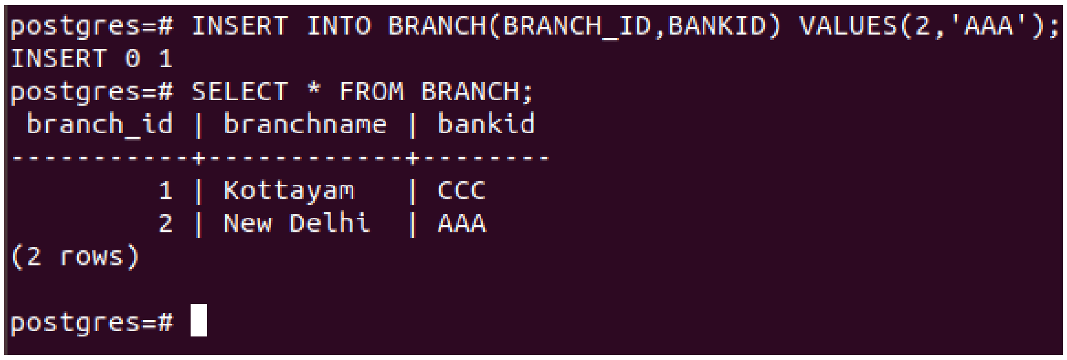
```
postgres=# CREATE TABLE BRANCH (
postgres(# BRANCH_ID INT PRIMARY KEY,
postgres(# BRANCHNAME TEXT DEFAULT 'New Delhi',
postgres(# BANKID VARCHAR(3) REFERENCES BANK(BANKCODE)
postgres(# ON DELETE CASCADE);
CREATE TABLE
postgres=# INSERT INTO BRANCH VALUES(1,'Kottayam','CCC');
INSERT 0 1
postgres=# SELECT * FROM BRANCH;
 branch_id | branchname | bankid
-----------+------------+--------
         1 | Kottayam   | CCC
(1 row)

postgres=#
```

i. Populate the database. Make sure that all constraints are working properly.



```
postgres=# INSERT INTO BRANCH(BRANCH_ID) VALUES(1);
ERROR:  duplicate key value violates unique constraint "branch_pkey"
DETAIL:  Key (branch_id)=(1) already exists.
postgres=# INSERT INTO BRANCH(BRANCH_ID,BANKID) VALUES(2,'SBT');
ERROR:  insert or update on table "branch" violates foreign key constraint "branch_bankid_fkey"
DETAIL:  Key (bankid)=(SBT) is not present in table "bank".
postgres=# 
```

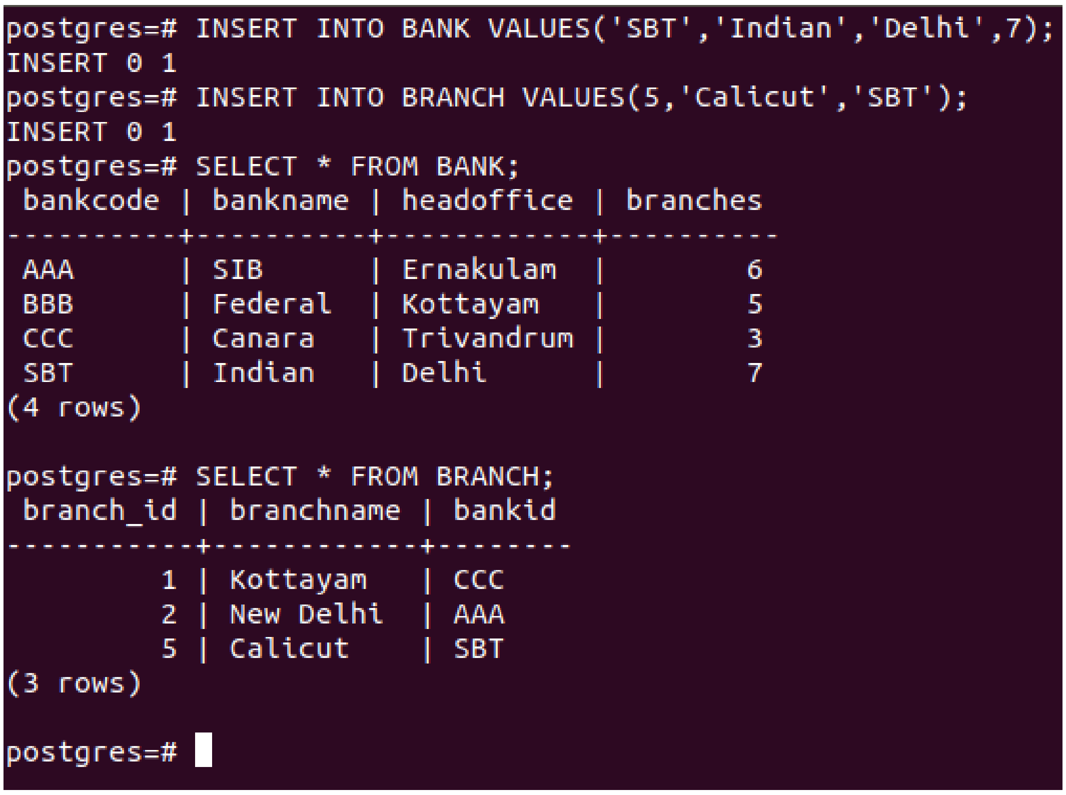
ii. During database population, demonstrate how the DEFAULT Constraint is satisfied.



```
postgres=# INSERT INTO BRANCH(BRANCH_ID,BANKID) VALUES(2,'AAA');
INSERT 0 1
postgres=# SELECT * FROM BRANCH;
 branch_id | branchname | bankid
-----------+------------+--------
         1 | Kottayam   | CCC
         2 | New Delhi  | AAA
(2 rows)

postgres=#
```

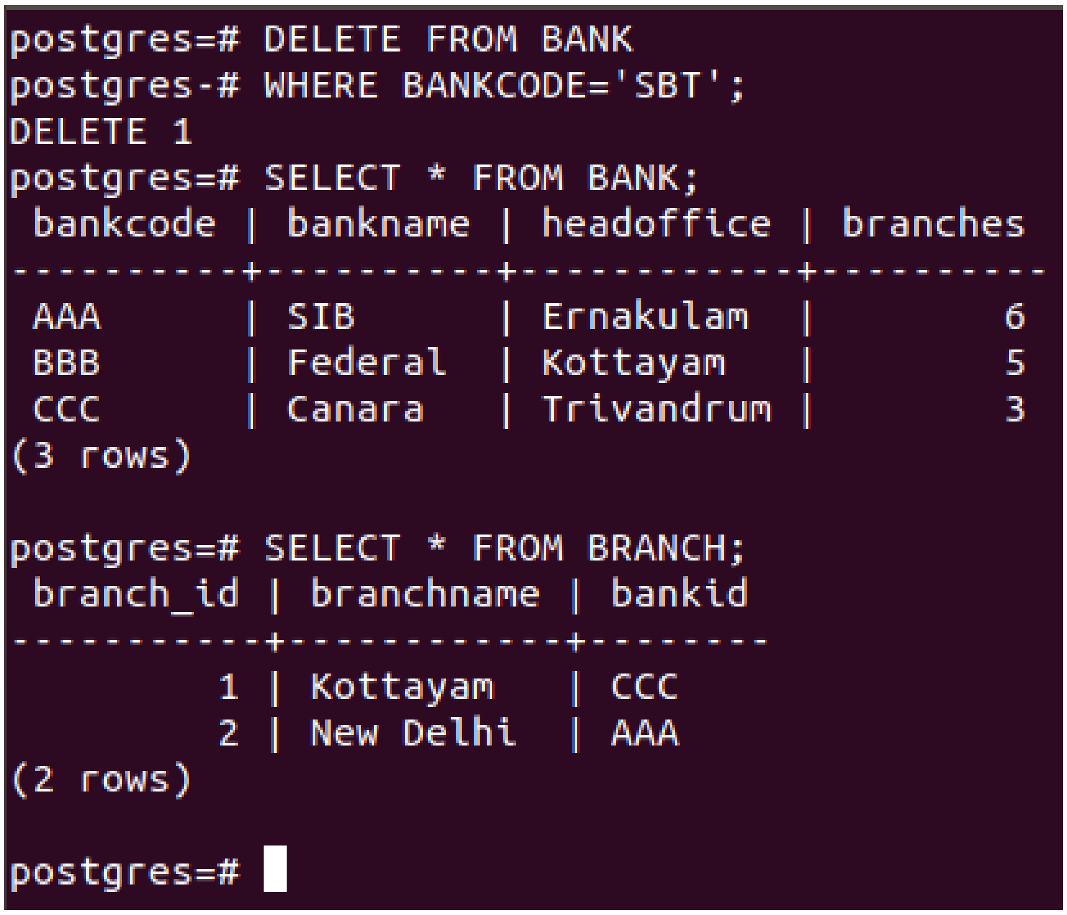
iii. Delete the bank with bank code 'SBT' and make sure that the corresponding entries are getting deleted from the related tables.

Before deletion



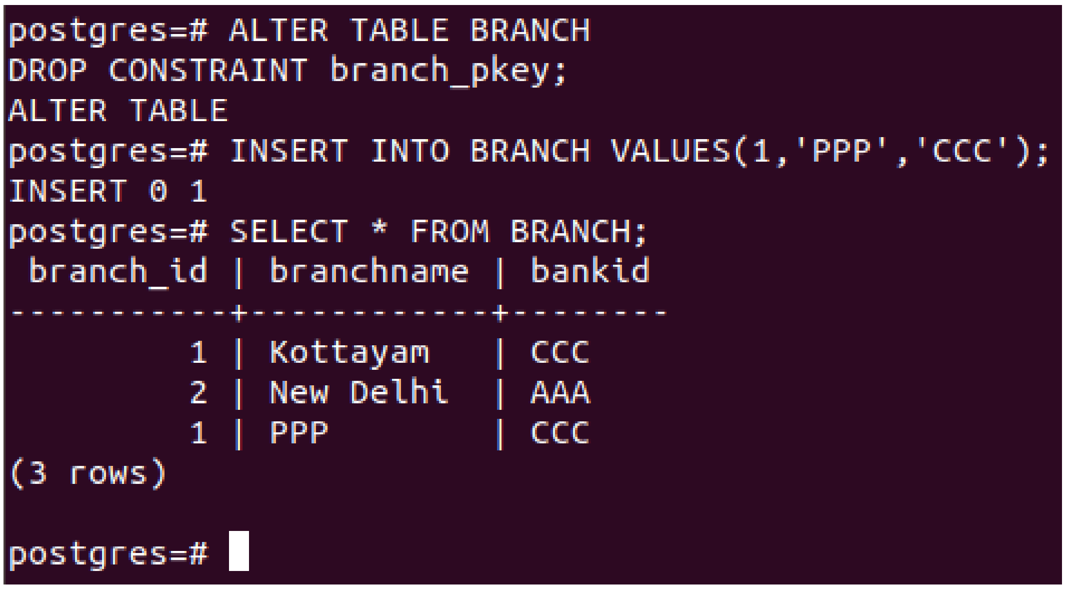
```
postgres=# INSERT INTO BANK VALUES('SBT','Indian','Delhi',7);
INSERT 0 1
postgres=# INSERT INTO BRANCH VALUES(5,'Calicut','SBT');
INSERT 0 1
postgres=# SELECT * FROM BANK;
 bankcode | bankname | headoffice | branches
----------+----------+------------+----------
 AAA      | SIB      | Ernakulam  |        6
 BBB      | Federal  | Kottayam   |        5
 CCC      | Canara   | Trivandrum |        3
 SBT      | Indian   | Delhi      |        7
(4 rows)

postgres=# SELECT * FROM BRANCH;
 branch_id | branchname | bankid
-----------+------------+--------
         1 | Kottayam   | CCC
         2 | New Delhi  | AAA
         5 | Calicut    | SBT
(3 rows)

postgres=# 
```

After deleteion



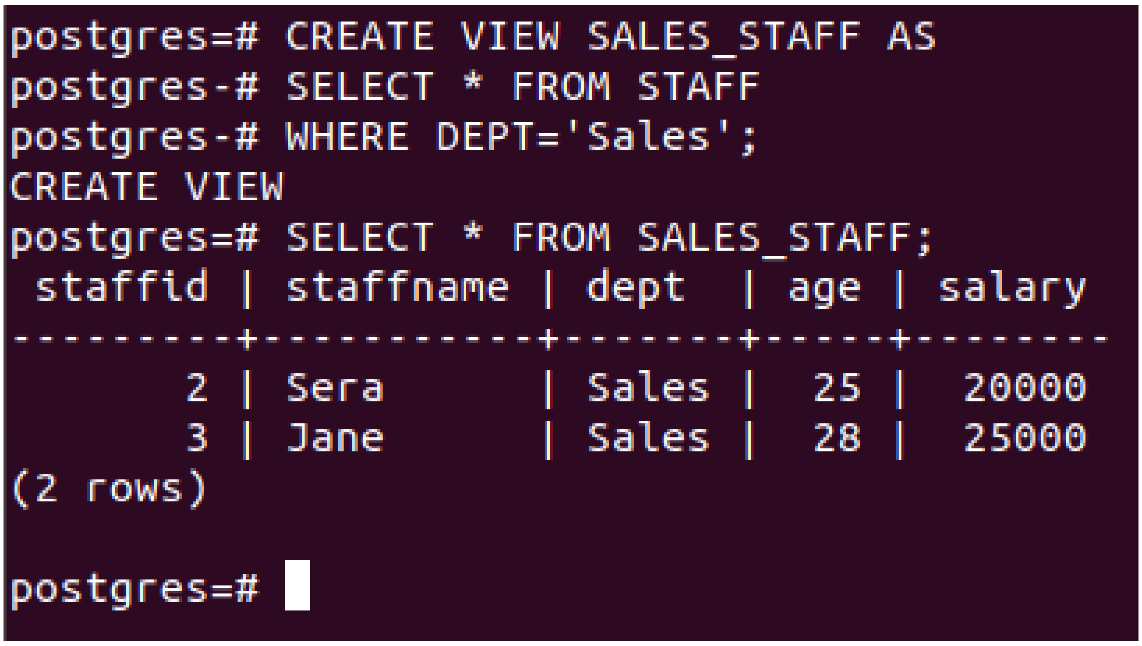
```
postgres=# DELETE FROM BANK
postgres-# WHERE BANKCODE='SBT';
DELETE 1
postgres=# SELECT * FROM BANK;
 bankcode | bankname | headoffice | branches
----------+----------+------------+----------
 AAA      | SIB      | Ernakulam  |        6
 BBB      | Federal  | Kottayam   |        5
 CCC      | Canara   | Trivandrum |        3
(3 rows)

postgres=# SELECT * FROM BRANCH;
 branch_id | branchname | bankid
-----------+------------+--------
         1 | Kottayam   | CCC
         2 | New Delhi  | AAA
(2 rows)

postgres=# 
```

iv. Drop the Primary Key using ALTER command



```
postgres=# ALTER TABLE BRANCH
DROP CONSTRAINT branch_pkey;
ALTER TABLE
postgres=# INSERT INTO BRANCH VALUES(1,'PPP','CCC');
INSERT 0 1
postgres=# SELECT * FROM BRANCH;
 branch_id | branchname | bankid
-----------+------------+--------
         1 | Kottayam   | CCC
         2 | New Delhi  | AAA
         1 | PPP        | CCC
(3 rows)

postgres=# 
```

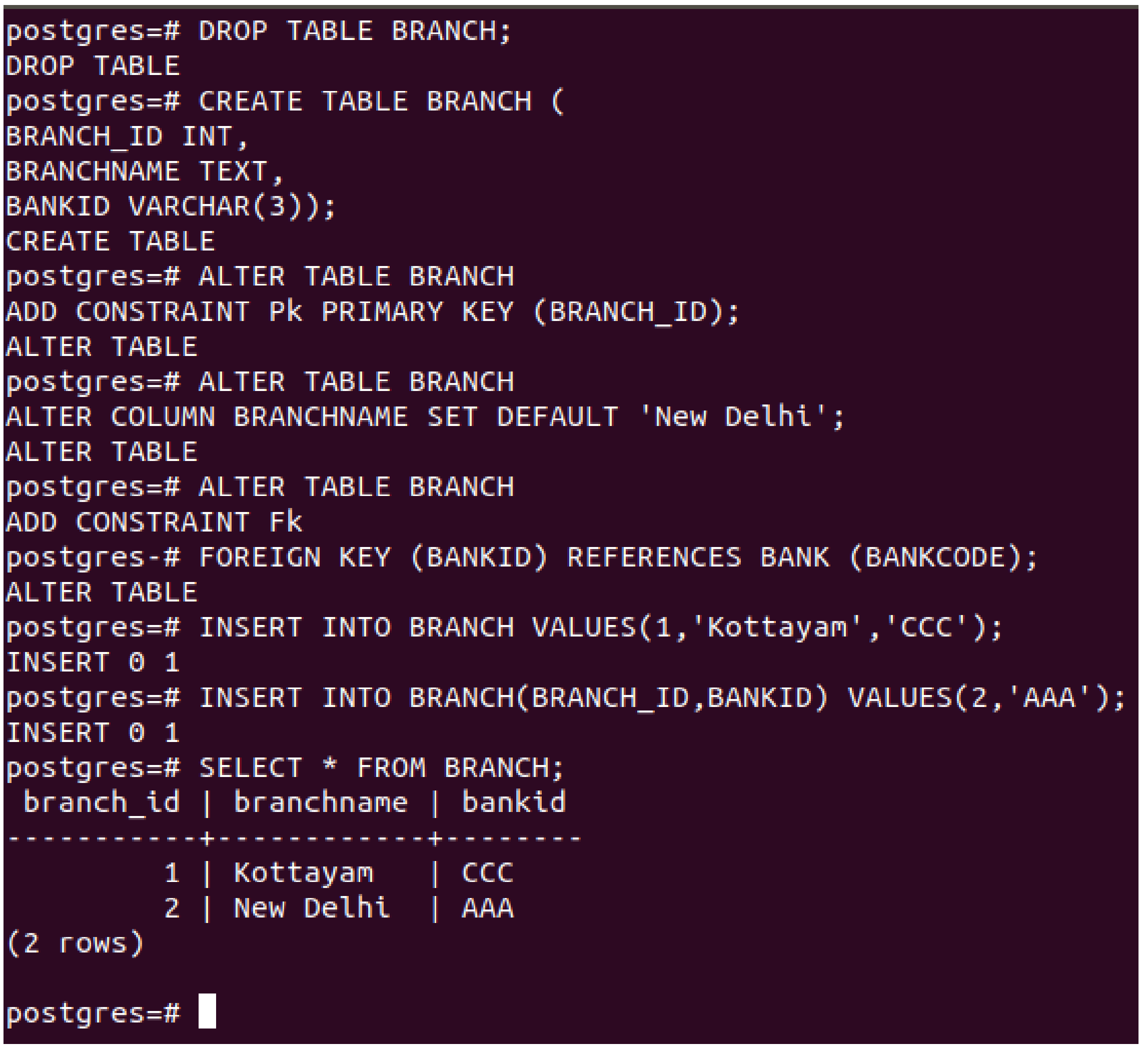
2. Create a View named sales_staff to hold the details of all staff working in sales Department



```
postgres=# CREATE VIEW SALES_STAFF AS
postgres-# SELECT * FROM STAFF
postgres-# WHERE DEPT='Sales';
CREATE VIEW
postgres=# SELECT * FROM SALES_STAFF;
 staffid | staffname | dept  | age | salary
---------+-----------+-------+-----+--------
       2 | Sera      | Sales |  25 |  20000
       3 | Jane      | Sales |  28 |  25000
(2 rows)

postgres=#
```

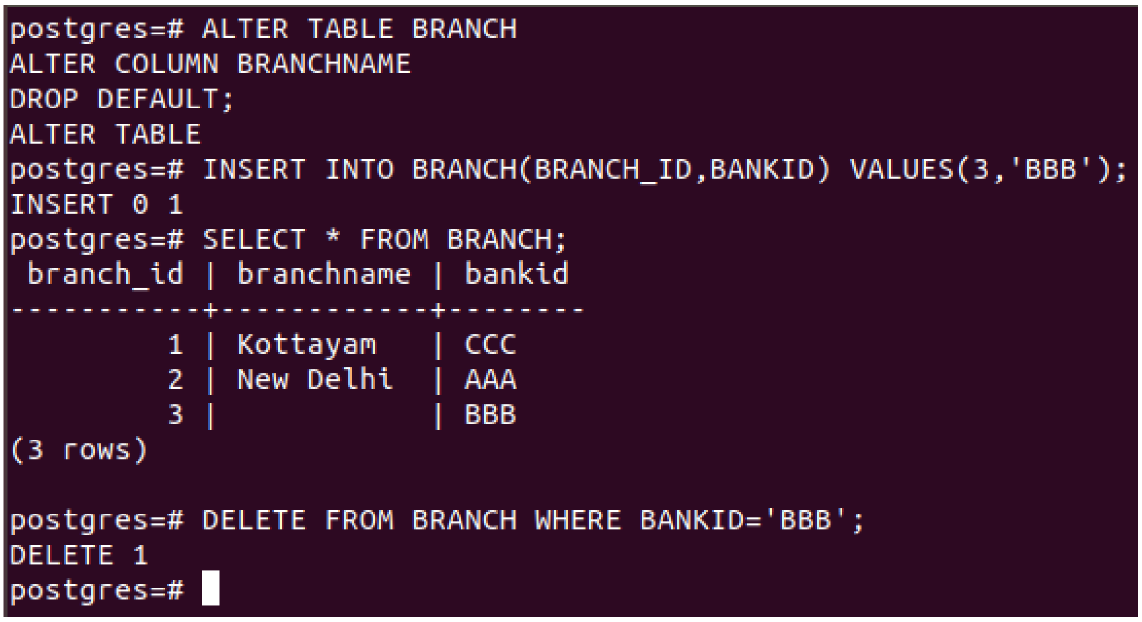
3. Drop table branch. Create another table named branch and name all the constraints as given below:

Constraint Name	Column	Constraint
Pk	branch_id	Primary Key
Df	branch_name	Default :'New Delhi'
Fk	bankid	Foreign key/References



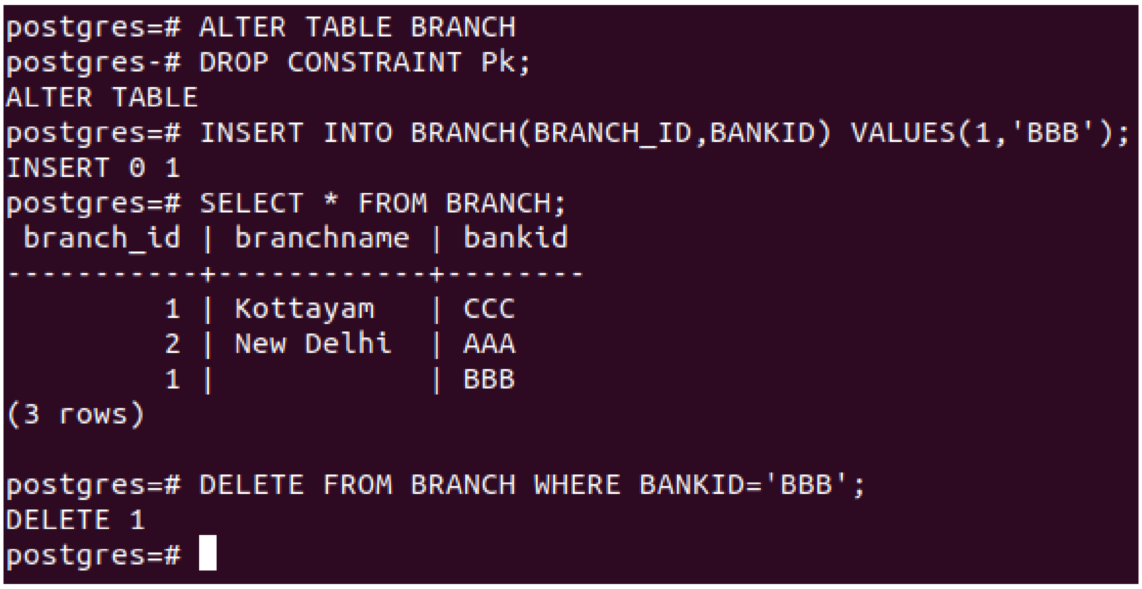
```
postgres=# DROP TABLE BRANCH;
DROP TABLE
postgres=# CREATE TABLE BRANCH (
BRANCH_ID INT,
BRANCHNAME TEXT,
BANKID VARCHAR(3));
CREATE TABLE
postgres=# ALTER TABLE BRANCH
ADD CONSTRAINT Pk PRIMARY KEY (BRANCH_ID);
ALTER TABLE
postgres=# ALTER TABLE BRANCH
ALTER COLUMN BRANCHNAME SET DEFAULT 'New Delhi';
ALTER TABLE
postgres=# ALTER TABLE BRANCH
ADD CONSTRAINT Fk
postgres-# FOREIGN KEY (BANKID) REFERENCES BANK (BANKCODE);
ALTER TABLE
postgres=# INSERT INTO BRANCH VALUES(1,'Kottayam','CCC');
INSERT 0 1
postgres=# INSERT INTO BRANCH(BRANCH_ID,BANKID) VALUES(2,'AAA');
INSERT 0 1
postgres=# SELECT * FROM BRANCH;
 branch_id | branchname | bankid
-----------+------------+--------
         1 | Kottayam   | CCC
         2 | New Delhi  | AAA
(2 rows)

postgres=# 
```

(a) Delete the default constraint in the table



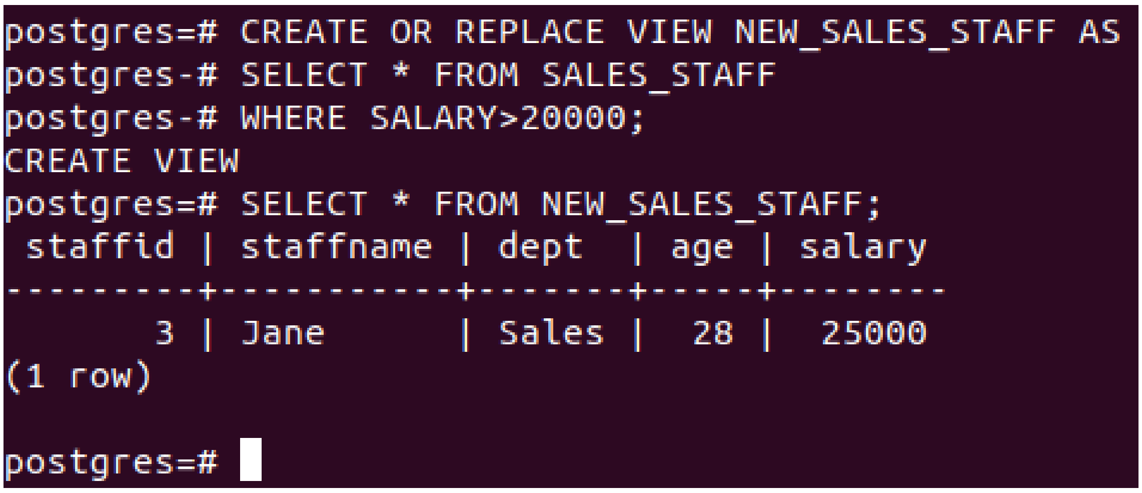
```
postgres=# ALTER TABLE BRANCH
ALTER COLUMN BRANCHNAME
DROP DEFAULT;
ALTER TABLE
postgres=# INSERT INTO BRANCH(BRANCH_ID,BANKID) VALUES(3,'BBB');
INSERT 0 1
postgres=# SELECT * FROM BRANCH;
 branch_id | branchname | bankid
-----------+------------+--------
         1 | Kottayam   | CCC
         2 | New Delhi  | AAA
         3 |            | BBB
(3 rows)

postgres=# DELETE FROM BRANCH WHERE BANKID='BBB';
DELETE 1
postgres=# 
```

(b) Delete the primary key constraint



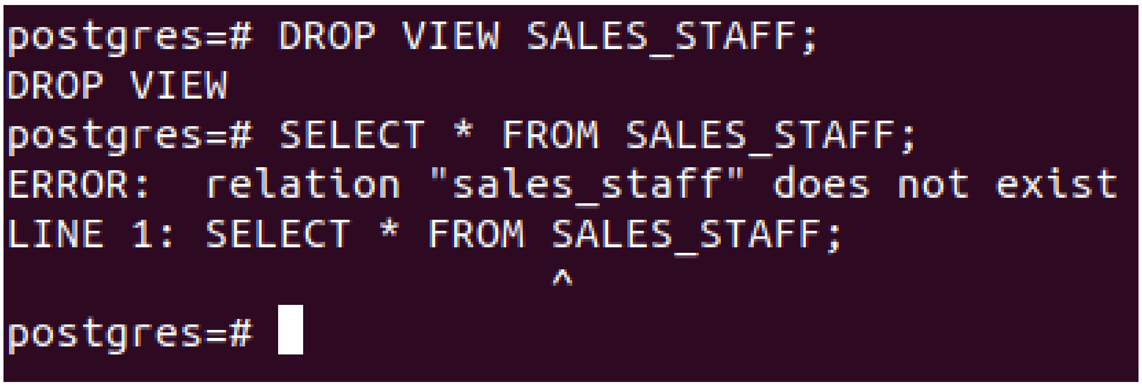
```
postgres=# ALTER TABLE BRANCH
postgres-# DROP CONSTRAINT Pk;
ALTER TABLE
postgres=# INSERT INTO BRANCH(BRANCH_ID,BANKID) VALUES(1,'BBB');
INSERT 0 1
postgres=# SELECT * FROM BRANCH;
 branch_id | branchname | bankid
-----------+------------+--------
         1 | Kottayam   | CCC
         2 | New Delhi  | AAA
         1 |            | BBB
(3 rows)

postgres=# DELETE FROM BRANCH WHERE BANKID='BBB';
DELETE 1
postgres=# 
```

4. Update the view sales_staff to include the details of staff belonging to sales department whose salary is greater than 20000.



```
postgres=# CREATE OR REPLACE VIEW NEW_SALES_STAFF AS
postgres-# SELECT * FROM SALES_STAFF
postgres-# WHERE SALARY>20000;
CREATE VIEW
postgres=# SELECT * FROM NEW_SALES_STAFF;
 staffid | staffname | dept  | age | salary
---------+-----------+-------+-----+--------
       3 | Jane      | Sales |  28 |  25000
(1 row)

postgres=# 
```

5. Delete the view sales_staff.



```
postgres=# DROP VIEW SALES_STAFF;
DROP VIEW
postgres=# SELECT * FROM SALES_STAFF;
ERROR:  relation "sales_staff" does not exist
LINE 1: SELECT * FROM SALES_STAFF;
                      ^
postgres=# 
```

RESULT

The query was executed and output was obtained.