COLLEGE OF ENGINEERING TRIVANDRUM
DEPARTMENT OF COMPUTER SCIENCE

APPLICATION SOFTWARE DEVELOPMENT
LAB EXAM REPORT(CS333)

HISHAM ALI P

REG. NO : LTVE17CS066

ROLL NO : 67

S5 CSE

05-11-2019

Staff In-charge:

VIPIN VASU A. V

Associate Professor

Department of Computer Science

Theory

1. SQL

   SQL is a database computer language designed for the retrieval and management of data in a relational database. SQL stands for Structured Query Language.

   The uses of SQL include modifying database table and index structures; adding, updating and deleting rows of data; and retrieving subsets of information from within a database for transaction processing and analytics applications. Queries and other SQL operations take the form of commands written as statements. Commonly used SQL statements include select, add, insert, update, delete, create, alter and truncate.

2. PL/SQL

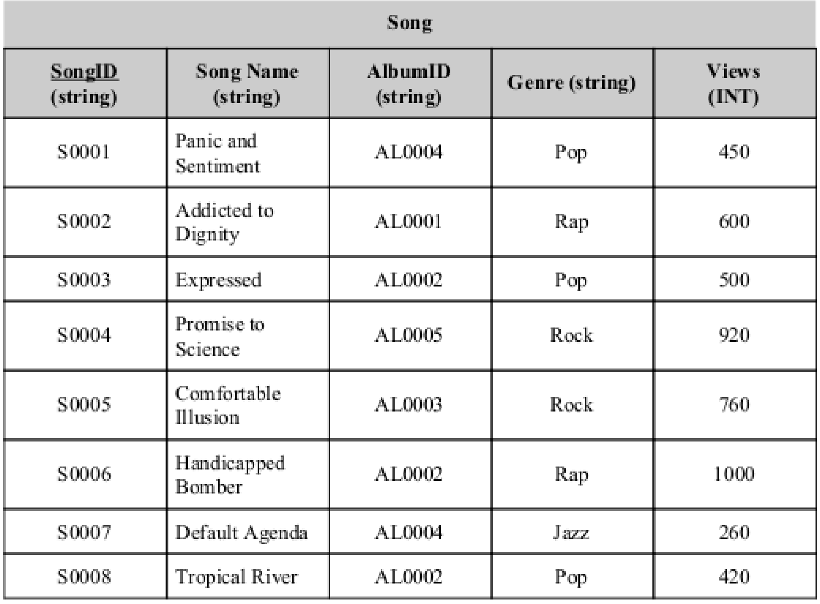
   PL/SQL, Oracle's procedural extensions to SQL, is an advanced fourth-generation programming language (4GL). It offers modern features such as data encapsulation, overloading, collection types, exception handling, and information hiding. PL/SQL also offers seamless SQL access, tight integration with the Oracle server and tools, portability, and security.

   PL/SQL is a block-structured language. That is, the basic units (procedures, functions, and anonymous blocks) that make up a PL/SQL program are logical blocks, which can contain any number of nested sub-blocks. Typically, each logical block corresponds to a problem or subproblem to be solved. Thus, PL/SQL supports the divide-and-conquer approach to problem solving called stepwise refinement. A block (or sub-block) lets you group logically related declarations and statements. That way, you can place declarations close to where they are used. The declarations are local to the block and cease to exist when the block completes. A PL/SQL block has three parts: a declarative part, an executable part, and an exception handling part. (In PL/SQL, a warning or error condition is called an exception.) Only the executable part is required. The order of the parts is logical. First comes the declarative part, in which items can be declared. Once declared, items can be manipulated in the executable part. Exceptions raised during execution can be dealt with in the exception-handling part. You can nest sub-blocks in the executable and exception-handling parts of a PL/SQL block or subprogram but not in the declarative part. Also, you can define local subprograms in the declarative part of any block. However, you can call local subprograms only from the block in which they are defined.

Question 1

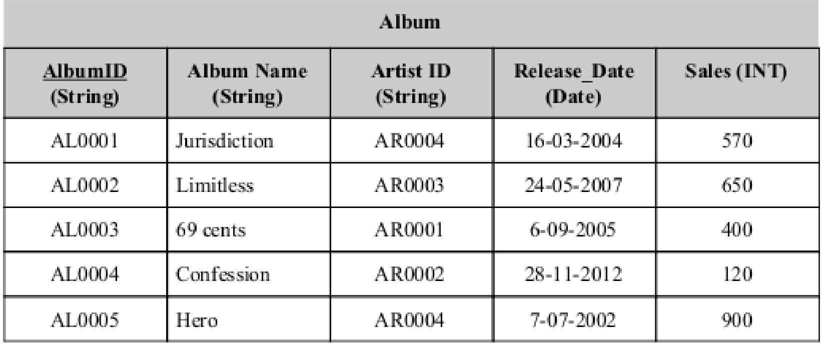
1. Create and populate shown tables. Check the following constraints:

a. Song_ID should be of the format “S00_ _”
b. Album_ID should be of the format “AL00_ _”
c. Album_ID should be of the format “AL00_ _”
d. Artist_ID should be of the format “AR00_ _”



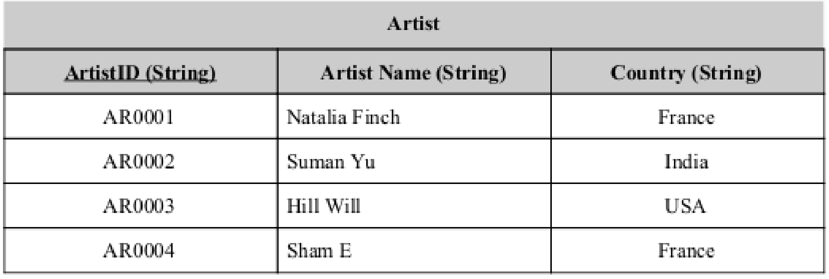
Song

SongID (string)	Song Name (string)	AlbumID (string)	Genre (string)	Views (INT)
S0001	Panic and Sentiment	AL0004	Pop	450
S0002	Addicted to Dignity	AL0001	Rap	600
S0003	Expressed	AL0002	Pop	500
S0004	Promise to Science	AL0005	Rock	920
S0005	Comfortable Illusion	AL0003	Rock	760
S0006	Handicapped Bomber	AL0002	Rap	1000
S0007	Default Agenda	AL0004	Jazz	260
S0008	Tropical River	AL0002	Pop	420

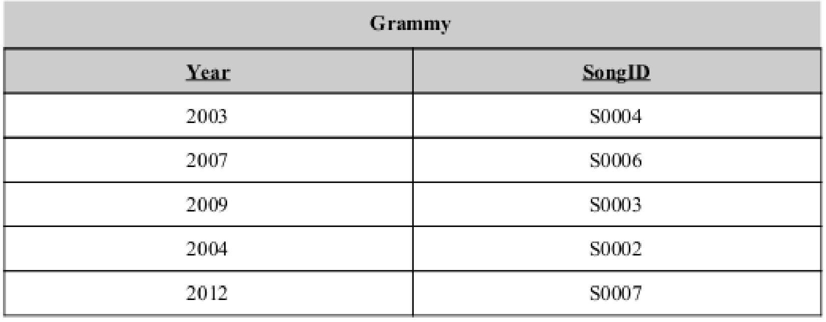


Album

AlbumID (String)	Album Name (String)	Artist ID (String)	Release_Date (Date)	Sales (INT)
AL0001	Jurisdiction	AR0004	16-03-2004	570
AL0002	Limitless	AR0003	24-05-2007	650
AL0003	69 cents	AR0001	6-09-2005	400
AL0004	Confession	AR0002	28-11-2012	120
AL0005	Hero	AR0004	7-07-2002	900



Artist		
ArtistID (String)	Artist Name (String)	Country (String)
AR0001	Natalia Finch	France
AR0002	Suman Yu	India
AR0003	Hill Will	USA
AR0004	Sham E	France

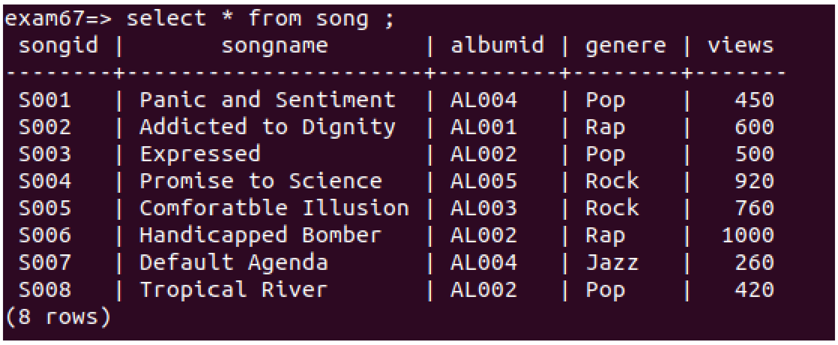


Grammy	
Year	SongID
2003	S0004
2007	S0006
2009	S0003
2004	S0002
2012	S0007

song

create table song(songid text primary key check (songid LIKE ’S00%’), songname text not null, albumid text not null check (albumid LIKE ’AL00%’) references album(albumid), genre text not null, views int not null);

INSERT INTO song values(’S0001’, ’Panic and Sentiment’, ’AL0004’, ’Pop’, 450),
(’S0002’, ’Addicted to Dignity’, ’AL0001’, ’Rap’, 600),
(’S0003’, ’Expressed’, ’AL0002’, ’Pop’, 500),
(’S0004’, ’Promise to Science’, ’AL0005’, ’Rock’, 920),
(’S0005’, ’Comforatble Illusion’, ’AL0003’, ’Rock’, 760),
(’S0006’, ’Handicapped Bomber’, ’AL0002’, ’Rap’, 1000),
(’S0007’, ’Default Agenda’, ’AL0004’, ’Jazz’, 260),
(’S0008’, ’Tropical River’, ’AL0002’, ’Pop’, 420);



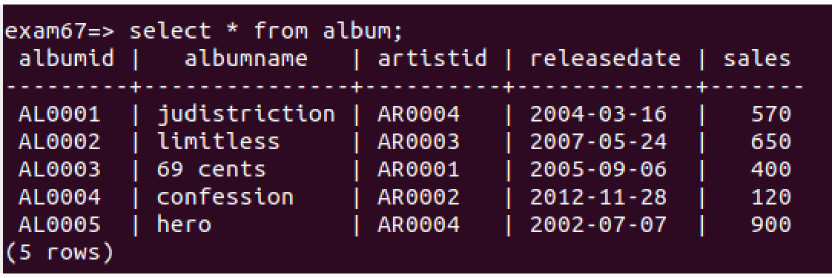
```
exam67=> select * from song ;
 songid |       songname        | albumid | genere | views
--------+-----------------------+---------+--------+-------
 S001   | Panic and Sentiment   | AL004   | Pop    |   450
 S002   | Addicted to Dignity   | AL001   | Rap    |   600
 S003   | Expressed             | AL002   | Pop    |   500
 S004   | Promise to Science    | AL005   | Rock   |   920
 S005   | Comforatble Illusion  | AL003   | Rock   |   760
 S006   | Handicapped Bomber    | AL002   | Rap    |  1000
 S007   | Default Agenda        | AL004   | Jazz   |   260
 S008   | Tropical River        | AL002   | Pop    |   420
(8 rows)
```

song.png

album

CREATE TABLE album(albumid text primary key, albumname text not null, artistid text check (artistid like ’AR00%’) references artist(artistid), releasedate date not null, sales int not null);

INSERT INTO album VALUES(’AL0001’,’judistriction’,’AR0004’,’2004-03-16’, 570),
(’AL0002’,’limitless’,’AR0003’,’2007-05-24’, 650),
(’AL0003’,’69 cents’,’AR0001’,’2005-09-06’, 400),
(’AL0004’,’confession’,’AR0002’,’2012-11-28’, 120),
(’AL0005’,’hero’,’AR0004’,’2002-07-07’, 900);



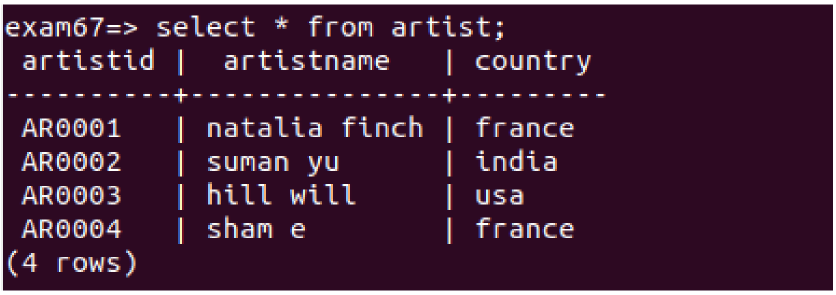
```
exam67=> select * from album;
 albumid |   albumname   | artistid | releasedate | sales
---------+---------------+----------+-------------+-------
 AL0001  | judistriction | AR0004   | 2004-03-16  |   570
 AL0002  | limitless     | AR0003   | 2007-05-24  |   650
 AL0003  | 69 cents      | AR0001   | 2005-09-06  |   400
 AL0004  | confession    | AR0002   | 2012-11-28  |   120
 AL0005  | hero          | AR0004   | 2002-07-07  |   900
(5 rows)
```

album.png

artist

CREATE TABLE artist(artistid text primary key check (artistid LIKE ’AR00%’), artistname text not null, country text);

INSERT INTO artist VALUES(’AR0001’, ’natalia finch’, ’france’),
(’AR0002’, ’suman yu’, ’india’),
(’AR0003’, ’hill will’, ’usa’),
(’AR0004’,’sham e’, ’france’);



```
exam67=> select * from artist;
 artistid |  artistname   | country
----------+---------------+---------
 AR0001   | natalia finch | france
 AR0002   | suman yu      | india
 AR0003   | hill will     | usa
 AR0004   | sham e        | france
(4 rows)
```

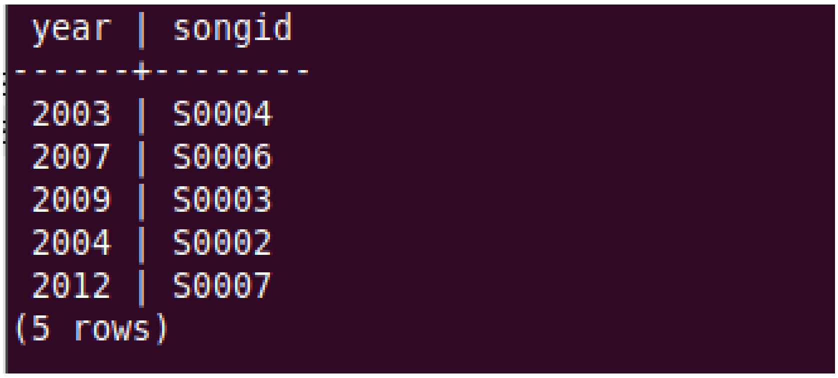
artist.png

grammy

create table grammy(year int, songid text check (songid LIKE 'S00%') references song(songid));

INSERT INTO grammy VALUES(2003,'S0004'), (2007,'S0006'), (2009,'S0003'), (2004,'S0002'), (2012,'S0007');

grammy.png

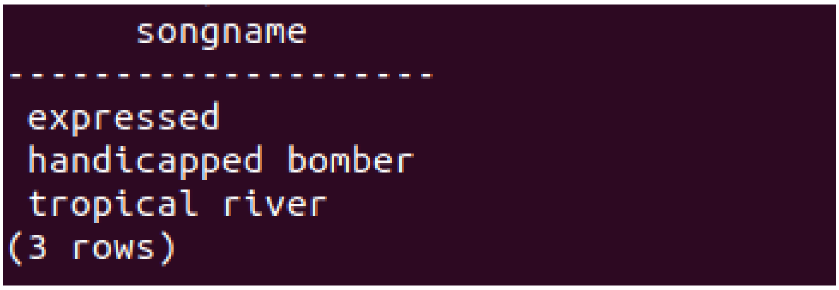


```
 year | songid
------+--------
 2003 | S0004
 2007 | S0006
 2009 | S0003
 2004 | S0002
 2012 | S0007
(5 rows)
```

Question 3

3. List all songs in the same album as Expressed.

SELECT s.songname FROM song as s, album as a WHERE s.albumid=a.albumid AND a.albumname='limitless';

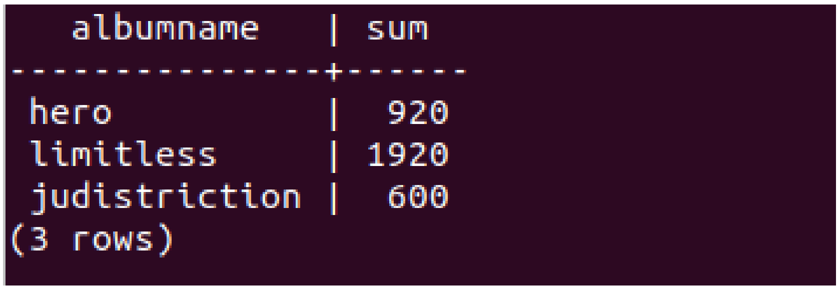


```
      songname
--------------------
 expressed
 handicapped bomber
 tropical river
(3 rows)
```

Question 4

4. Display the name and total number of views of all albums with sales more than 500

SELECT a.albumname,sum(s.views) FROM album as a, song as s WHERE s.albumid=a.albumid AND sales>500 GROUP BY by a.albumname;

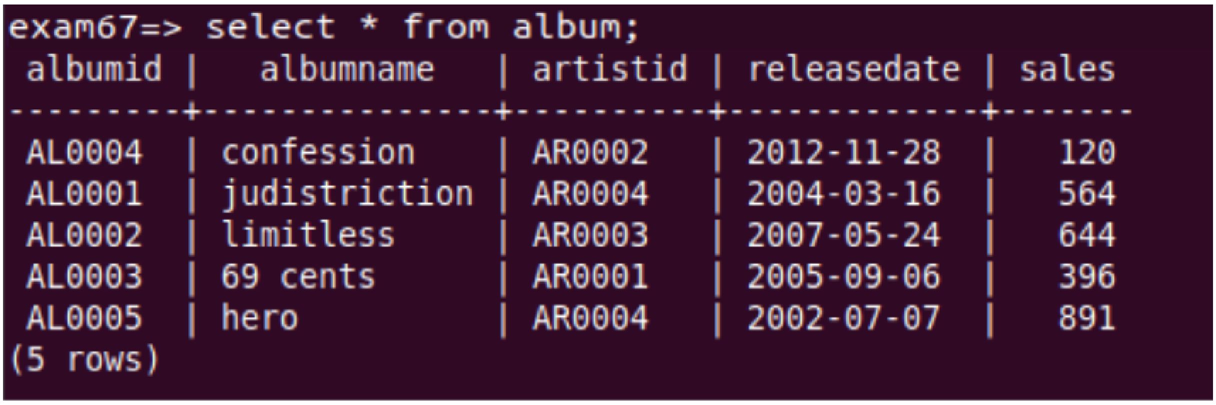


```
   albumname    | sum
----------------+------
 hero           |  920
 limitless      | 1920
 judistriction  |  600
(3 rows)
```

Question 8

8. Decrease the sales by 10% for all albums released before 2009. Perform the operation using cursor.

```
create or replace function cur1() returns void as $$
declare
c cursor for select * from album;
r record;
begin
open c;
loop
fetch from c into r;
exit when not found;
if cast(substr(cast(r.releasedate as text),1,4)as int) < 2009 then
update album set sales = sales - (sales * 0.01) where current of c;
end if;
end loop;
close c;
end;
$$ language plpgsql;
```



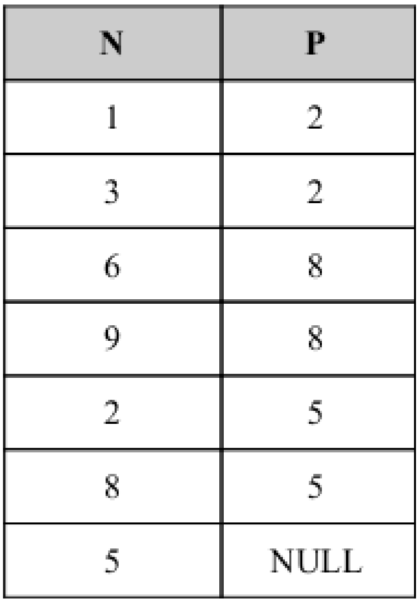
```
exam67=> select * from album;
 albumid |   albumname   | artistid | releasedate | sales
---------+---------------+----------+-------------+-------
 AL0004  | confession    | AR0002   | 2012-11-28  |   120
 AL0001  | judistriction | AR0004   | 2004-03-16  |   564
 AL0002  | limitless     | AR0003   | 2007-05-24  |   644
 AL0003  | 69 cents      | AR0001   | 2005-09-06  |   396
 AL0005  | hero          | AR0004   | 2002-07-07  |   891
(5 rows)
```

Question 9

9. You are given a table, BST, containing two columns: N and P, where N represents the value of a node in a Binary Tree, and P is the parent of N.
Write a query to find the node type of Binary Tree ordered by the value of the node.
Output one of the following for each node:

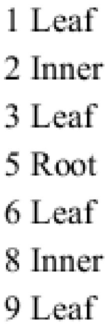
- Root: If node is the root node.
- Leaf: If node is leaf node.
- Inner: If node is neither root nor leaf node

Sample Input



N	P
1	2
3	2
6	8
9	8
2	5
8	5
5	NULL

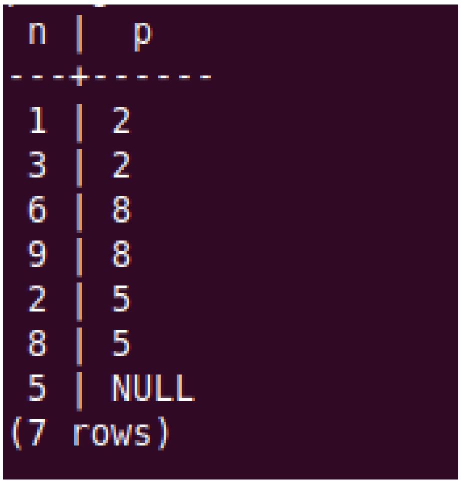
Sample Output



```
1 Leaf
2 Inner
3 Leaf
5 Root
6 Leaf
8 Inner
9 Leaf
```

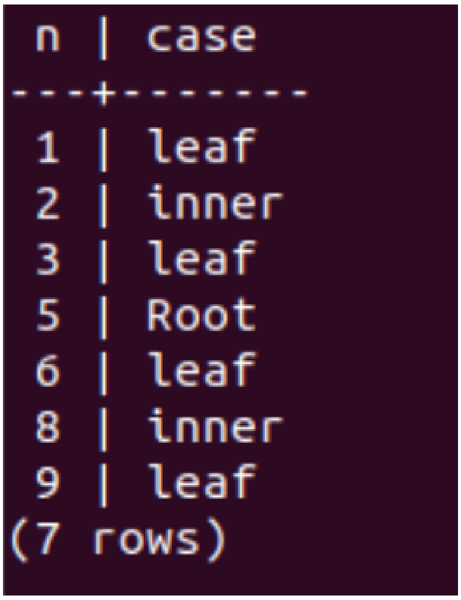
```
    CREATE TABLE bst(n text, p text);
```

```
INSERT INTO bst VALUES(’1’,’2’), (’3’,’2’), (’6’,’8’), (’9’,’8’), (’2’,’5’), (’8’,’5’), (’5’,’NULL’);
```



```
 n |  p
---+------
 1 | 2
 3 | 2
 6 | 8
 9 | 8
 2 | 5
 8 | 5
 5 | NULL
(7 rows)
```

```
SELECT n, CASE
WHEN p=’NULL’ THEN ’Root’
WHEN n IN (SELECT p FROM bst) THEN ’inner’
ELSE ’leaf’
END
FROM bst ORDER BY n ;
```



```
 n | case
---+-------
 1 | leaf
 2 | inner
 3 | leaf
 5 | Root
 6 | leaf
 8 | inner
 9 | leaf
(7 rows)
```

PL/SQL

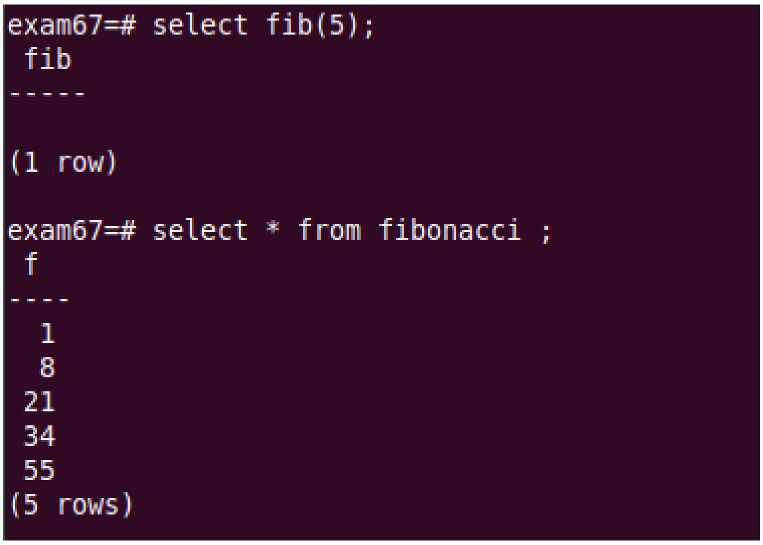
2. Write a function to find the first n Fibonacci numbers which are not prime and store them into a table.

```
create table fibonacci(f int);



create or replace function fib(n int) returns void as $$
declare
f int := 0;
s int := 1;
temp int;
i int := 1;
c int := 1;
flag int;
begin
insert into fibonacci values(1);
while(c<n) loop
flag := 0;
temp=f+s;
if temp>2 then
for i in 2..n loop
if temp !=i and temp%i=0 then
flag := 1;
end if;
end loop;
end if;
if flag = 1 then
insert into fibonacci values(temp);
c := c+1;
end if;
f := s;
s := temp;
end loop;
```

end
$$ language plpgsql;



```
exam67=# select fib(5);
 fib
-----

(1 row)

exam67=# select * from fibonacci ;
 f
----
  1
  8
 21
 34
 55
(5 rows)
```