Name : Sumon Singh

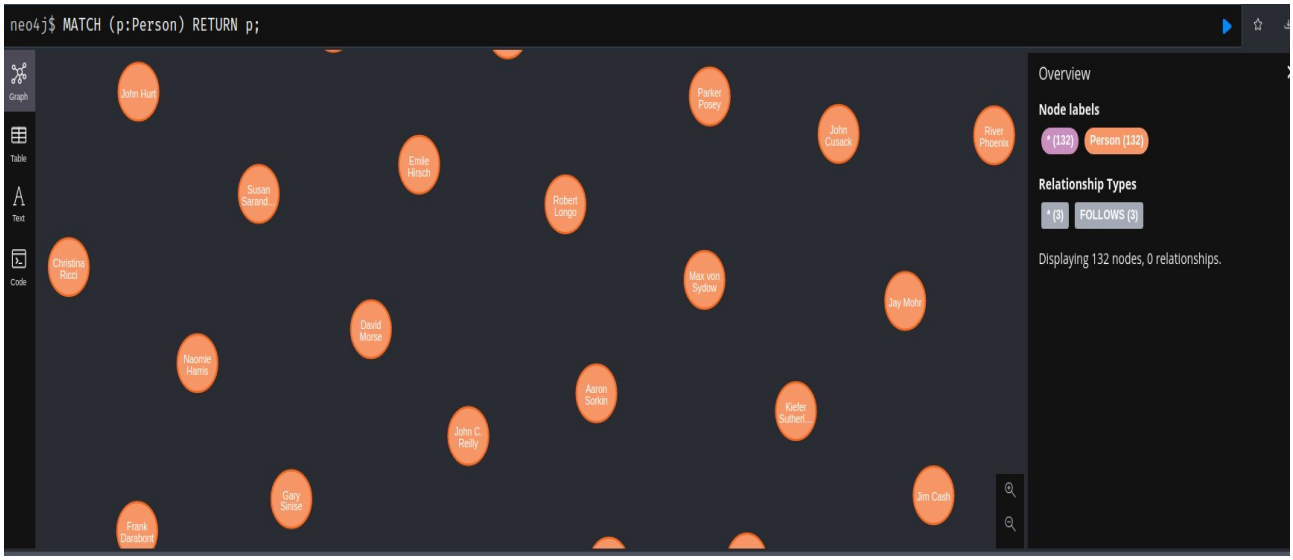
Roll no : 16

Paper : Database Technologies

Assignment : 1

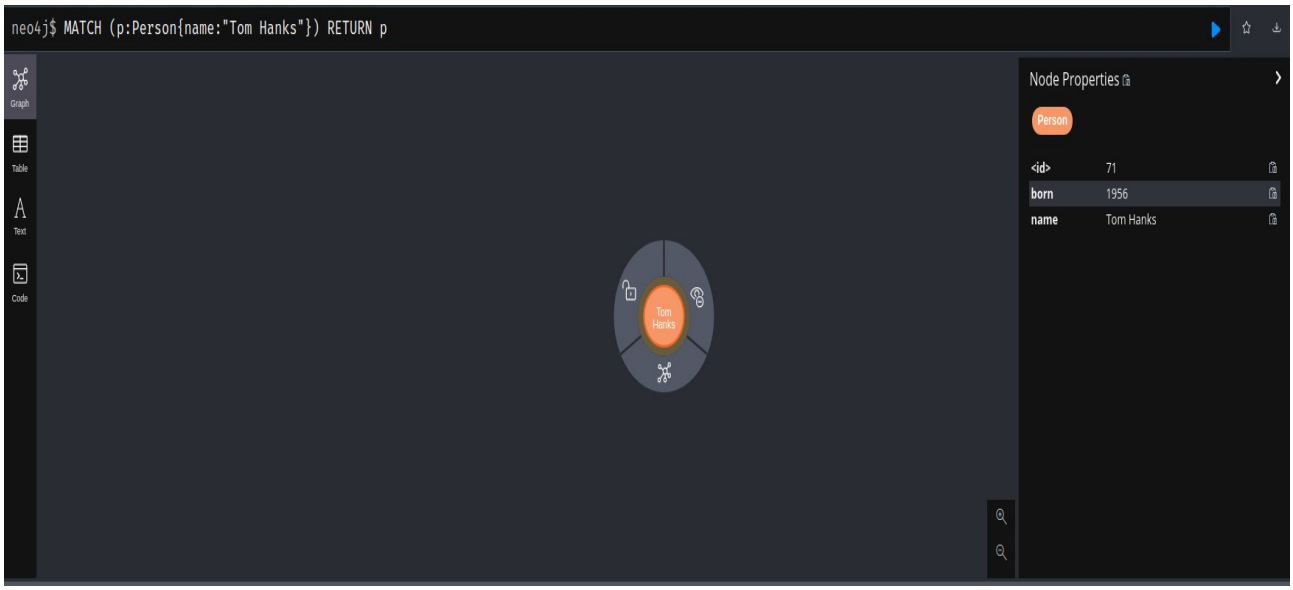
1: Find the labelled Person nodes in the graph. Note that we must use a variable like p for the Person node if we want to retrieve the node in the RETURN clause.

= > MATCH (p:Person) RETURN p ;



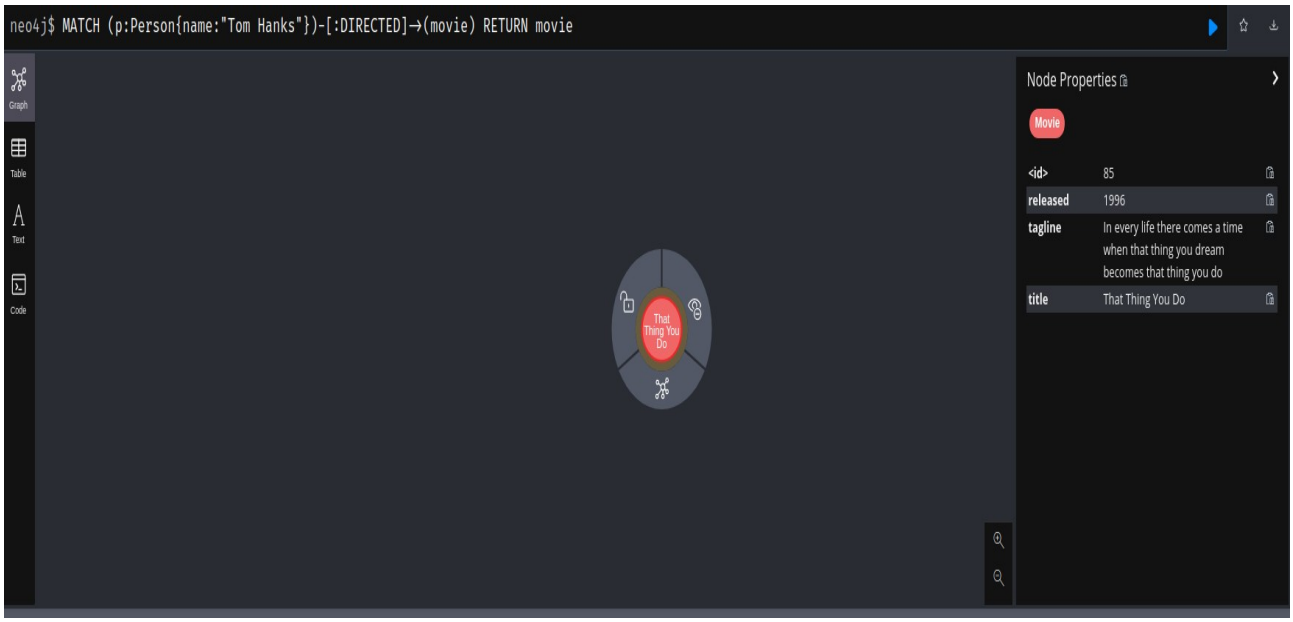
2: Find Person nodes in the graph that have a name of 'Tom Hanks'.

= > MATCH (p:Person{name:"Tom Hanks"}) RETURN p ;



3: Find which `Movie`s Tom Hanks has directed.

= > MATCH (p:Person{name:"Tom Hanks"})-[:DIRECTED]->(movie) RETURN movie

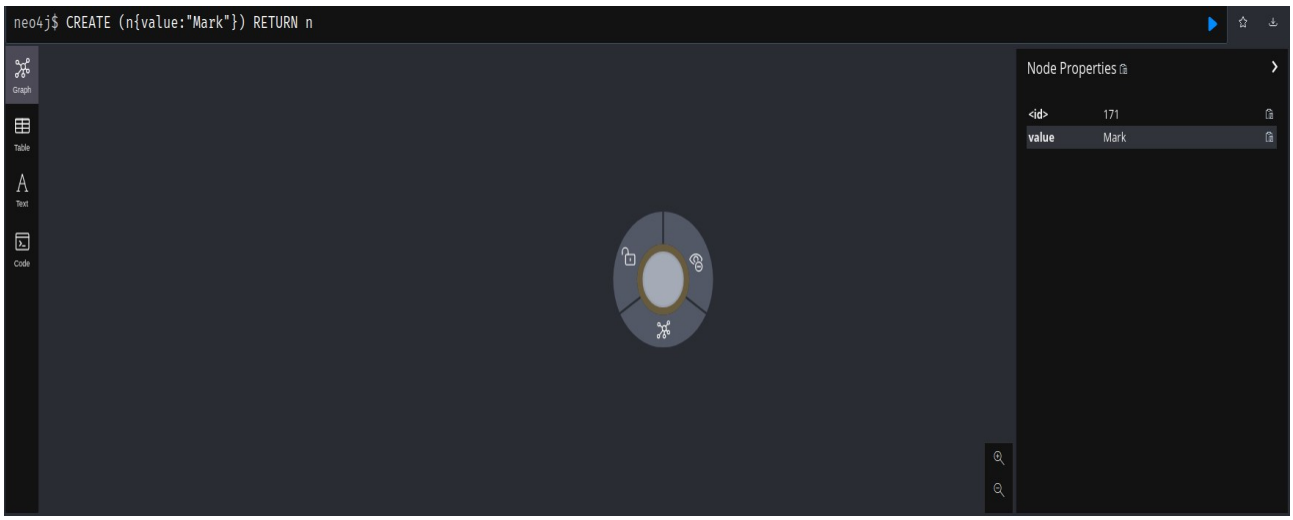


4: Find which Movie Tom Hanks has directed, but this time, return only the title of the movie.

= > MATCH (p:Person{name:"Tom Hanks"})-[:DIRECTED]->(m:Movie) RETURN m.title

5: Create a node with value Mark

= > CREATE (n{value:"Mark"}) RETURN n



6: Example application of knowledge graphs in fraud analytics.

= >

With the advancement of science and technology industries and companies are more likely to switch all their daily and major activities via online .One of the major example of this online system is money transaction,but due to fraud people and hackers the security is one of the major concern of online system. Though there are softwares and tools available to detect fraudery but none of them can assure 100% prevention of fraudery. To maximize the accuarcy for detecting a fraud one of the most renowned technology is **Knowledge graphs**.

A knowledge graph is a combination of machine learning and graph technologies to demonstrate a database with all kinds of information in a complex network manner. The main advantages of a knowledge graph over a normal table based database are visualization and generating patterns. With knowledge graph it’s easy to visualize patterns in a pictural manner.

Below we will deep dive of a case study of Credit Card fraud transaction and analysis using knowledge graphs:

Problem :

Banks, merchent and payment gateway companies loses a huge amount of money due to creadit card fraud. Hackers steal the details of credit card using some software or tools and use that informations for fraudery.An

incident of such credit card fraudery was caught in February 2017,where a 60 year old Doctor in India was duped of 1.40 lakh rupees, where hackers took the details of his credit card and made some huge amount of transaction . Later the doctor got a call from the bank to ensure if he himself has done the transaction but it was noticed that the doctor never had such kind of transaction. This kind of incidents of fraud can be seen in lot of places in the world nowadays.

Method :

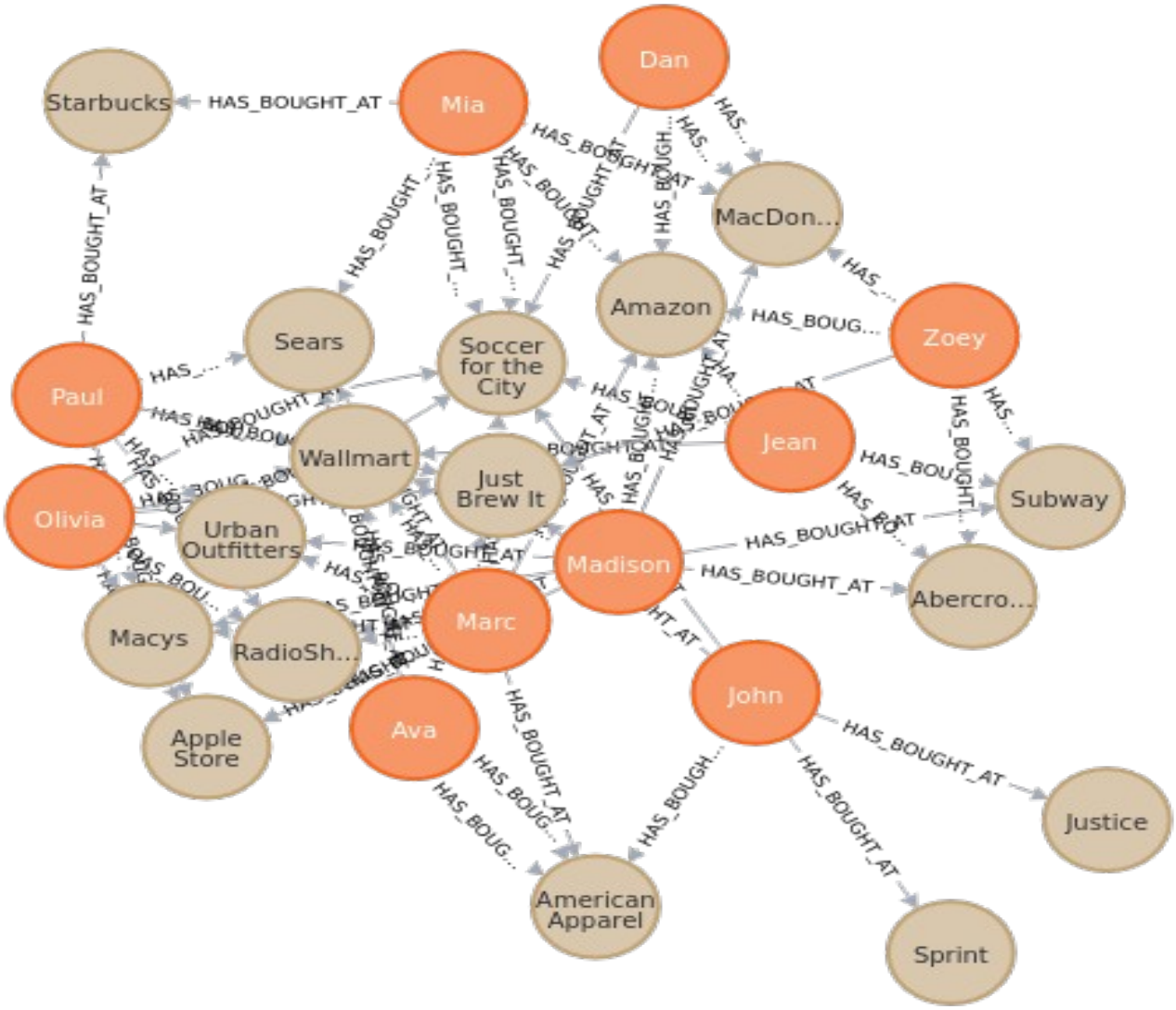
In this case study we will use knowledge graph to represent each of the transaction as graph and will try to detect the pattern of fraud transaction and find the origin of scam.

Data :

Neo4j sample dataset has been used for this case study.

Design and Development :

Step 1: First we will view our sample dataset .



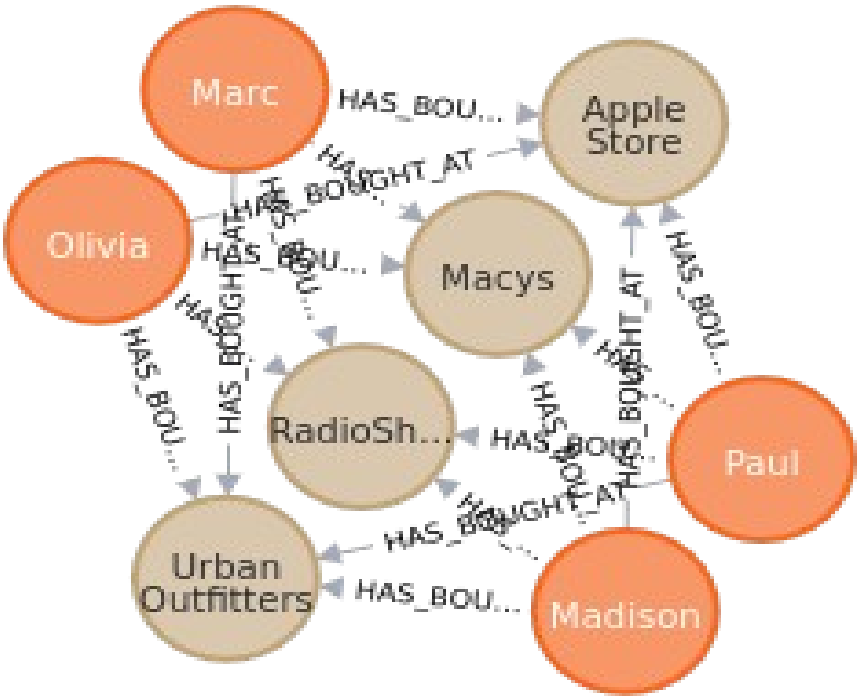
The above graph represent our dataset where the nodes in orange color are the details of customers such as id,name,gender and age .The nodes in gray color are the details of store/merchant such as id,name,street and address. The edges represent which customer has purchased at what time in what amount from which merchant and the status of the purchased item which is whether the purchase is disputed or undisputed.

Step 2: Identify the fraud transaction

Code :

```
MATCH (victim:Person)-[r:HAS_BOUGHT_AT]->(merchant)
WHERE r.status = "Disputed"
RETURN *
```

Output :



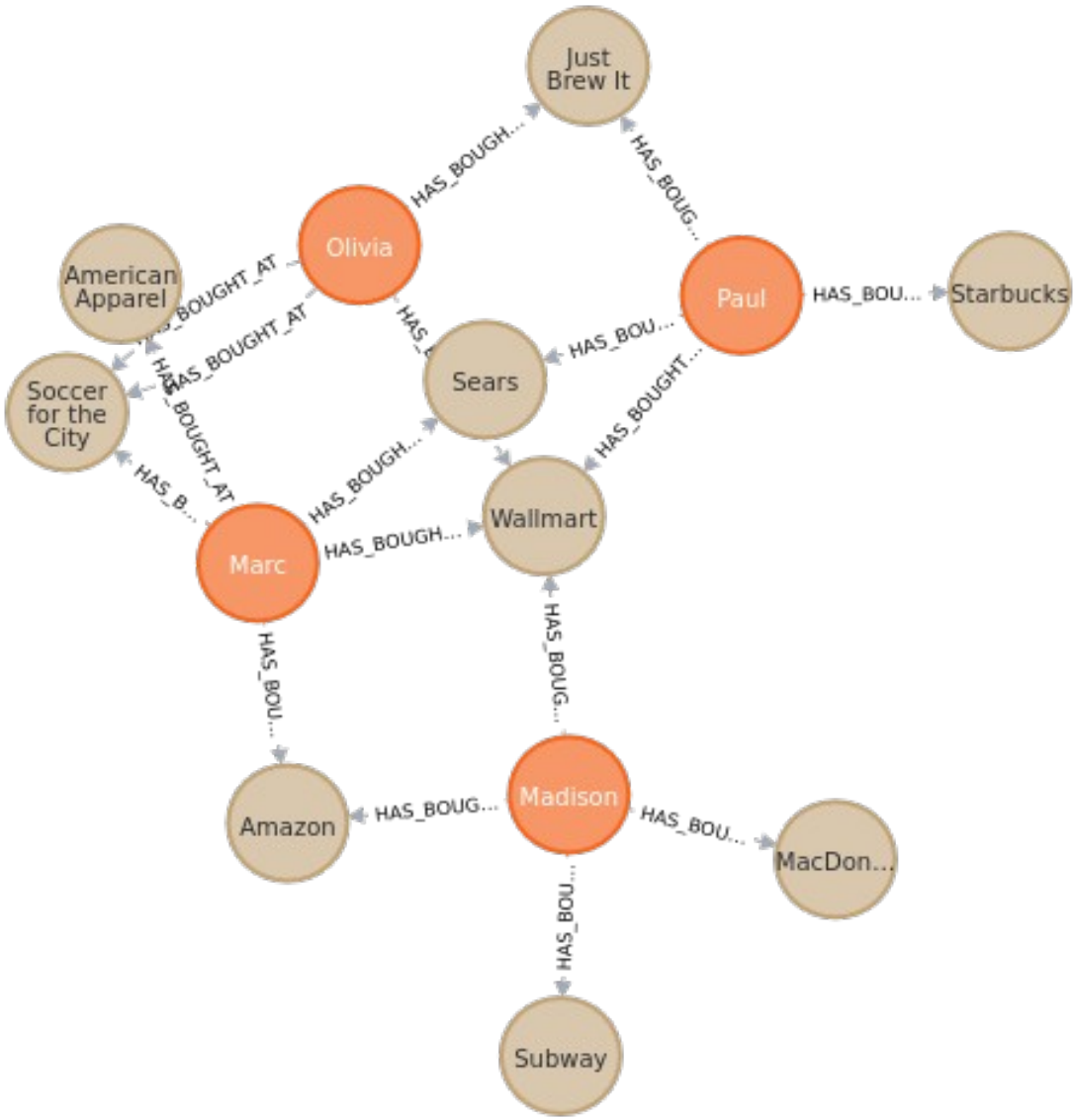
In the above graph we are representing the customers whose transaction status is disputed ,which may be a symbol of fraud. The graph is showing the details of the customers' transaction.

Step 3 : Identify the legitimate and illegitimate transactions date and time

Code :

```
MATCH (victim:Person)-[r:HAS_BOUGHT_AT]->(merchant)
WHERE r.status = "Disputed"
MATCH (victim)-[t:HAS_BOUGHT_AT]->(othermerchants)
WHERE t.status = "Undisputed" AND t.time < r.time
WITH victim, othermerchants, t ORDER BY t.time DESC
RETURN *
```

Output :



Above graph represents all the transactions where disputed transaction and undisputed transaction where time of transaction of undisputed is less than the disputed one.

Step 4: Identify all the merchants for which the transaction may same doubtful.

Code :

```
MATCH (victim:Person)-[r:HAS_BOUGHT_AT]->(merchant)
WHERE r.status = "Disputed"
MATCH (victim)-[t:HAS_BOUGHT_AT]->(othermerchants)
WHERE t.status = "Undisputed" AND t.time < r.time
WITH victim, othermerchants, t ORDER BY t.time DESC
```

RETURN DISTINCT othermerchants.name AS `Suspicious Store`, count(DISTINCT t) AS Count, collect(DISTINCT victim.name) AS Victims ORDER BY Count DESC

Output :

Suspicious Store	Count	Victims
Wallmart	4	[Olivia,Madison,Marc,Paul]
Starbucks	1	[Paul]
American Apparel	1	[Marc]
Just Brew It	1	[Paul]
Amazon	1	[Marc]
Sears	1	[Paul]
Subway	1	[Madison]
MacDonalds	1	[Madison]
Soccer for the City	1	[Olivia]

The above table shows the list of Suspicious Store and the name of victims.

Thus we can detect a fraud of credit card using knowledge graphs.Though it doesn't claim to detect 100% credit card fraud but knowledge graph increases the accuracy of fraud detection.

References :

[1] Jean Villedieu,Credit Card Fraud Detection,Neo4j

[2] Dr. Shruti Mantri and Vishal Siram,Knowledge Graph: Facilitates Fraud Analytics,isb-institute-of-data-science, https://isb-institute-of-data-science.medium.com/knowledge-graph-for-financial-services-c9cb7c3fe2b9

[3] Data Source : https://neo4j.com/graphgist/credit-card-fraud-detection

[4] TheIndianExpress,Credit card fraud: 60-year-old doctor 'duped' of Rs 1.40 lakh, https://indianexpress.com/article/cities/mumbai/credit-card-fraud-60-year-old-doctor-duped-of-rs-1-40-lakh-5387258/

Name : Sumon Singh

Roll No : 16

Assignment : 2

Subject : Database Technologies

Create Database :

```
CREATE DATABASE Store_Db;
use Store_Db;
```

Create The Tables :

```
CREATE TABLE employee
(id INTEGER,
CONSTRAINT pk_employee PRIMARY KEY (id),
name VARCHAR(20),
salary INTEGER,
manager INTEGER,
birthyear INTEGER,
startyear INTEGER);

CREATE TABLE dept
(id INTEGER ,CONSTRAINT
pk_dept PRIMARY KEY (id),
name
VARCHAR(20),
store INTEGER NOT NULL,
floor
INTEGER,
manager
INTEGER);
```

```
CREATE TABLE item
(id INTEGER ,CONSTRAINT
pk_item PRIMARY KEY (id),
name
VARCHAR(20),
dept INTEGER NOT NULL,
price INTEGER,
qoh INTEGER CONSTRAINT ck_item_qoh CHECK (qoh >= 0),
supplier INTEGER NOT NULL);

CREATE TABLE parts
(id INTEGER ,CONSTRAINT
pk_parts PRIMARY KEY(id),
name
VARCHAR(20),
color
VARCHAR(8),
weight
INTEGER,
qoh
INTEGER);

CREATE TABLE supply
(supplier INTEGER
NOT NULL, part
INTEGER NOT NULL,
shipdate DATE NOT
NULL, quan
INTEGER,
CONSTRAINT pk_supply PRIMARY KEY (supplier, part,
shipdate));

CREATE TABLE sale
(debit INTEGER
```

```
NOT NULL, item
INTEGER NOT
NULL,
quantity INTEGER,
CONSTRAINT pk_sale PRIMARY KEY (debit, item));

CREATE TABLE debit
(id INTEGER ,CONSTRAINT
pk_debit PRIMARY KEY(id),
sdate DATE DEFAULT(current_date()) NOT NULL,
employee INTEGER NOT NULL,
account INTEGER NOT NULL);

CREATE TABLE city
(id INTEGER,name VARCHAR(15) ,CONSTRAINT
pk_city PRIMARY KEY(id), state
VARCHAR(6));

CREATE TABLE store
(id INTEGER ,CONSTRAINT
pk_store PRIMARY KEY(id), city
INTEGER NOT NULL);

CREATE TABLE supplier
(id INTEGER ,CONSTRAINT
pk_supplier PRIMARY KEY(id), name
VARCHAR(20),
city INTEGER NOT NULL);
```

Add Constraints :

```
ALTER
TABLE dept
ADD CONSTRAINT fk_dept_store FOREIGN KEY (store)
```

```
REFERENCES store (id);

ALTER TABLE dept
ADD CONSTRAINT fk_dept_employee FOREIGN KEY
(manager)
REFERENCES employee (id) ON DELETE SET NULL;

ALTER TABLE item
ADD CONSTRAINT fk_item_dept FOREIGN KEY (dept)
REFERENCES dept
(id);


ALTER TABLE item
ADD CONSTRAINT fk_item_supplier FOREIGN KEY
(supplier)
REFERENCES supplier (id);

ALTER TABLE supply
ADD CONSTRAINT fk_supply_supplier FOREIGN KEY
(supplier)
REFERENCES supplier (id);

ALTER TABLE supply
ADD CONSTRAINT fk_supply_parts FOREIGN KEY (part)
REFERENCES
parts (id);

ALTER TABLE sale
ADD CONSTRAINT fk_sale_item FOREIGN KEY (item)
REFERENCES item (id);

ALTER TABLE sale
ADD CONSTRAINT fk_sale_debit FOREIGN KEY (debit)
REFERENCES debit(id);
```

```
ALTER TABLE debit
ADD CONSTRAINT fk_debit_employee FOREIGN KEY
(employee)
REFERENCES employee (id);

ALTER TABLE store
ADD CONSTRAINT fk_store_city FOREIGN KEY (city)
REFERENCES city
(id);

ALTER TABLE supplier
ADD CONSTRAINT fk_supplier_city FOREIGN KEY (city)
REFERENCES
city (id);
```

Insert Datas :

```
INSERT INTO employee
values(157,"Jones,Tim",12000,199,1940,1960);
INSERT INTO employee
values(1110,"Smith,Paul",6000,33,1952,1973);
INSERT INTO employee
values(35,"Evans,Michael",5000,32,1952,1974);
INSERT INTO employee
values(129,"Thomas,Tom",10000,199,1941,1962);
INSERT INTO employee
values(13,"Edwards,Peter",9000,199,1928,1958);
INSERT INTO employee
values(215,"Collins,Joanne",7000,10,1950,1971);
INSERT INTO employee
values(55,"James,Mary",12000,199,1920,1969);
INSERT INTO employee
values(26,"Thompson,Bob",13000,199,1930,1970);
```

```
INSERT INTO employee
values(98,"Williams,Judy",9000,199,1935,1969);
INSERT INTO employee
values(32,"Smyth,Carlo",9050,199,1929,1967);
INSERT INTO employee
values(33,"Hayes,Evelyn",10100,199,1931,1963);
INSERT INTO employee
values(199,"Bullock,J.D",27000,NULL,1920,1920);
INSERT INTO employee values(4901,"Bailey,Chas
M.",8377,32,1956,1975);
INSERT INTO employee
values(843,"Schmidt,Herman",11204,26,1936,1956);
INSERT INTO employee values(2398,"Wallace,Maggie
J.",7880,26,1940,1959);
INSERT INTO employee
values(1639,"Choy,Wanda",11160,55,1947,1970);
INSERT INTO employee
values(5119,"Bono,Sonny",13621,55,1939,1963);
INSERT INTO employee
values(37,"Raveen,Lemont",11985,26,1950,1974);
INSERT INTO employee values(5219,"Schwarz,Jason
B.",13374,33,1944,1959);
INSERT INTO employee values(1529,"Zugnoni,Arthur
A.",19868,129,1928,1949);
INSERT INTO employee values(430,"Brunet,Paul
C.",17674,129,1938,1959);
INSERT INTO employee
values(994,"Iwano,Masahiro",15641,129,1944,1970);
INSERT INTO employee
values(1330,"Onstad,Richard",8779,13,1952,1971);
INSERT INTO employee
values(10,"Ross,Stanley",15908,199,1927,1945);
INSERT INTO employee
values(11,"Ross,Stuart",12067,NULL,1931,1932);
```

```
INSERT INTO city VALUES(900,"Los Angeles","Calif");
INSERT INTO city VALUES(946,"Oakland","Calif");
INSERT INTO city VALUES(945,"El Cerrito","Calif");
INSERT INTO city VALUES(303,"Atlanta","Ga");
INSERT INTO city VALUES(941,"San Francisco","Calif");
INSERT INTO city VALUES(021,"Boston","Mass");
INSERT INTO city VALUES(752,"Dallas","Tex");
INSERT INTO city VALUES(802,"Denver","Colo");
INSERT INTO city VALUES(106,"White Plains","Neb");
INSERT INTO city VALUES(010,"Amherst","Mass");
INSERT INTO city VALUES(981,"Seattle","Wash");
INSERT INTO city VALUES(609,"Paxton","Ill");
INSERT INTO city VALUES(100,"New York","NY");
INSERT INTO city VALUES(921,"San Diego","Calif");
INSERT INTO city VALUES(118,"Hickville","Okla");
INSERT INTO city VALUES(841,"Salt Lake City","Utah");
INSERT INTO city VALUES(537,"Madison","Wisc");

INSERT INTO store VALUES(5,941);
INSERT INTO store VALUES(7,946);
INSERT INTO store VALUES(8,945);
INSERT INTO store VALUES(9,941);

INSERT INTO dept VALUES(35,"Book",5,1,55);
INSERT INTO dept VALUES(10,"Candy",5,1,13);
INSERT INTO dept VALUES(19,"Furniture",7,4,26);
INSERT INTO dept VALUES(20,"Major Appliances",7,4,26);
INSERT INTO dept VALUES(14,"Jewelry",8,1,33);
INSERT INTO dept VALUES(43,"Children's",8,2,32);
INSERT INTO dept VALUES(65,"Junior's",7,3,37);
INSERT INTO dept VALUES(58,"Men's",7,2,129);
INSERT INTO dept VALUES(60,"Sportswear",5,1,10);
INSERT INTO dept VALUES(99,"Giftwrap",5,1,98);
INSERT INTO dept VALUES(1,"Bargain",5,0,37);
INSERT INTO dept VALUES(26,"Linens",7,3,157);
```

```
INSERT INTO dept VALUES(63,"Women's",7,3,32);
INSERT INTO dept VALUES(49,"Toys",8,2,35);
INSERT INTO dept VALUES(70,"Women's",5,1,10);
INSERT INTO dept VALUES(73,"Children's",5,1,10);
INSERT INTO dept VALUES(34,"Stationary",5,1,33);
INSERT INTO dept VALUES(47,"JuniorMiss",7,2,129);
INSERT INTO dept VALUES(28,"Women's",8,2,32);


INSERT INTO supplier VALUES(199,"Koret",900);
INSERT INTO supplier VALUES(213,"Cannon",303);
INSERT INTO supplier VALUES(33,"Levi-Strauss",941);
INSERT INTO supplier VALUES(89,"Fisher-Price",021);
INSERT INTO supplier VALUES(125,"Playskool",752);
INSERT INTO supplier VALUES(42,"Whiteman's",802);
INSERT INTO supplier VALUES(15,"White Stag",106);
INSERT INTO supplier VALUES(475,"DEC",010);
INSERT INTO supplier VALUES(122,"White Paper",981);
INSERT INTO supplier VALUES(440,"Spooley",609);
INSERT INTO supplier VALUES(241,"IBM",100);
INSERT INTO supplier VALUES(62,"Data General",303);
INSERT INTO supplier VALUES(5,"Amdahl",921);
INSERT INTO supplier VALUES(20,"Wormley",118);
INSERT INTO supplier VALUES(67,"Edger",841);
INSERT INTO supplier VALUES(999,"A E Neumann",537);

INSERT INTO item VALUES(26,"Earrings",14,1000,20,199);
INSERT INTO item
VALUES(118,"Towels,Bath",26,250,1000,213);
INSERT INTO item VALUES(43,"Maze",49,325,200,89);
INSERT INTO item VALUES(106,"Clock
Book",49,198,150,125);
INSERT INTO item VALUES(23,"1 lb Box",10,215,100,42);
INSERT INTO item VALUES(52,"Jacket",60,3295,300,15);
INSERT INTO item VALUES(165,"Jean",65,825,500,33);
```

```
INSERT INTO item VALUES(258,"Shirt",58,650,1200,33);
INSERT INTO item VALUES(120,"Twin Sheet",26,800,750,213);
INSERT INTO item VALUES(301,"Boy's Jean
suit",43,1250,500,33);
INSERT INTO item VALUES(121,"Queen
sheet",26,1375,600,213);
INSERT INTO item VALUES(101,"Slacks",63,1600,325,15);
INSERT INTO item VALUES(115,"Gold Ring",14,4995,10,199);
INSERT INTO item VALUES(25,"2 lb Box,Mix",10,450,75,42);
INSERT INTO item VALUES(119,"Squeeze
Ball",49,250,400,89);
INSERT INTO item VALUES(11,"Wash cloth",1,75,575,213);
INSERT INTO item VALUES(19,"Bellbottoms",43,450,600,33);
INSERT INTO item VALUES(21,"ABC Blocks",1,198,405,125);
INSERT INTO item VALUES(107,"The `Feel`
Book",35,225,225,89);
INSERT INTO item VALUES(127,"Ski
Jumpsuit",65,4350,125,15);

INSERT INTO debit VALUES(100581,"95-1-15",157,10000000);
INSERT INTO debit VALUES(100582,"95-1-
15",1110,14356540);
INSERT INTO debit VALUES(100586,"95-1-16",35,14096831);
INSERT INTO debit VALUES(100592,"95-1-17",129,10000000);
INSERT INTO debit VALUES(100593,"95-1-18",35,11652133);
INSERT INTO debit VALUES(100594,"95-1-19",215,12591815);

INSERT INTO sale VALUES(100581,118,5);
INSERT INTO sale VALUES(100581,120,1);
INSERT INTO sale VALUES(100582,26,1);
INSERT INTO sale VALUES(100586,127,3);
INSERT INTO sale VALUES(100586,106,2);
INSERT INTO sale VALUES(100592,258,1);
INSERT INTO sale VALUES(100593,23,2);
INSERT INTO sale VALUES(100594,52,1);
```

```
INSERT INTO parts VALUES(1,"central processor","pink",10,1);
INSERT INTO parts VALUES(2,"memory","gray",20,32);
INSERT INTO parts VALUES(3,"disk drive","black",685,2);
INSERT INTO parts VALUES(4,"tape drive","black",450,4);
INSERT INTO parts VALUES(5,"tapes","gray",1,250);
INSERT INTO parts VALUES(6,"line printer","yellow",578,3);
INSERT INTO parts VALUES(7,"l-p paper","white",15,95);
INSERT INTO parts VALUES(8,"terminals","blue",19,5);
INSERT INTO parts VALUES(13,"paper tape
reader","black",107,0);
INSERT INTO parts VALUES(14,"paper tape
punch","black",147,0);
INSERT INTO parts VALUES(9,"terminal paper","white",2,350);
INSERT INTO parts VALUES(10,"byte-soap","clear",0,143);
INSERT INTO parts VALUES(11,"card reader","gray",327,0);
INSERT INTO parts VALUES(12,"card punch","gray",427,0);

INSERT INTO supply VALUES(475,1,"1993-12-31",1);
INSERT INTO supply VALUES(475,2,"1994-05-31",32);
INSERT INTO supply VALUES(475,3,"1993-12-31",2);
INSERT INTO supply VALUES(475,4,"1994-05-31",1);
INSERT INTO supply VALUES(122,7,"1995-02-01",144);
INSERT INTO supply VALUES(122,7,"1995-02-02",48);
INSERT INTO supply VALUES(122,9,"1995-02-01",144);
INSERT INTO supply VALUES(440,6,"1994-10-10",2);
INSERT INTO supply VALUES(241,4,"1993-12-31",1);
INSERT INTO supply VALUES(62,3,"1994-06-18",3);
INSERT INTO supply VALUES(475,2,"1993-12-31",32);
INSERT INTO supply VALUES(475,1,"1994-07-01",1);
INSERT INTO supply VALUES(5,4,"1994-11-15",3);
INSERT INTO supply VALUES(5,4,"1995-01-22",6);
INSERT INTO supply VALUES(20,5,"1995-01-10",20);
INSERT INTO supply VALUES(20,5,"1995-01-11",75);
INSERT INTO supply VALUES(241,1,"1995-06-01",1);
```

INSERT INTO supply VALUES(241,2,"1995-06-01",32);
INSERT INTO supply VALUES(241,3,"1995-06-01",1);
INSERT INTO supply VALUES(67,4,"1995-07-01",1);
INSERT INTO supply VALUES(999,10,"1996-01-01",144);
INSERT INTO supply VALUES(241,8,"1995-07-01",1);
INSERT INTO supply VALUES(241,9,"1995-07-01",144);
INSERT INTO supply VALUES(89,3,"1995-07-04",1000);
INSERT INTO supply VALUES(89,4,"1995-07-04",1000);

Queries :

1) List all employees, i.e. all tuples in the employee relation.

Code : select * from employee;

```
mysql> select * from employee;
+------+--------------------+--------+---------+-----------+-----------+
| id   | name               | salary | manager | birthyear | startyear |
+------+--------------------+--------+---------+-----------+-----------+
|   10 | Ross,Stanley       |  15908 |     199 |      1927 |      1945 |
|   11 | Ross,Stuart        |  12067 |    NULL |      1931 |      1932 |
|   13 | Edwards,Peter      |   9000 |     199 |      1928 |      1958 |
|   26 | Thompson,Bob       |  13000 |     199 |      1930 |      1970 |
|   32 | Smyth,Carlo        |   9050 |     199 |      1929 |      1967 |
|   33 | Hayes,Evelyn       |  10100 |     199 |      1931 |      1963 |
|   35 | Evans,Michael      |   5000 |      32 |      1952 |      1974 |
|   37 | Raveen,Lemont      |  11985 |      26 |      1950 |      1974 |
|   55 | James,Mary         |  12000 |     199 |      1920 |      1969 |
|   98 | Williams,Judy      |   9000 |     199 |      1935 |      1969 |
|  129 | Thomas,Tom         |  10000 |     199 |      1941 |      1962 |
|  157 | Jones,Tim          |  12000 |     199 |      1940 |      1960 |
|  199 | Bullock,J.D        |  27000 |    NULL |      1920 |      1920 |
|  215 | Collins,Joanne     |   7000 |      10 |      1950 |      1971 |
|  430 | Brunet,Paul C.     |  17674 |     129 |      1938 |      1959 |
|  843 | Schmidt,Herman     |  11204 |      26 |      1936 |      1956 |
|  994 | Iwano,Masahiro     |  15641 |     129 |      1944 |      1970 |
| 1110 | Smith,Paul         |   6000 |      33 |      1952 |      1973 |
| 1330 | Onstad,Richard     |   8779 |      13 |      1952 |      1971 |
| 1529 | Zugnoni,Arthur A.  |  19868 |     129 |      1928 |      1949 |
| 1639 | Choy,Wanda         |  11160 |      55 |      1947 |      1970 |
| 2398 | Wallace,Maggie J.  |   7880 |      26 |      1940 |      1959 |
| 4901 | Bailey,Chas M.     |   8377 |      32 |      1956 |      1975 |
| 5119 | Bono,Sonny         |  13621 |      55 |      1939 |      1963 |
| 5219 | Schwarz,Jason B.   |  13374 |      33 |      1944 |      1959 |
+------+--------------------+--------+---------+-----------+-----------+
25 rows in set (0.00 sec)
```

2) List the name of all departments in alphabetical order. Note: by “name” we mean the name attribute for all tuples in the dept relation.

Code : select distinct(name) from dept order by(name);

name
Bargain
Book
Candy
Children's
Furniture
Giftwrap
Jewelry
Junior's
JuniorMiss
Linens
Major Appliances
Men's
Sportswear
Stationary
Toys
Women's

16 rows in set (0.00 sec)

3) Which employees have a salary between 9000 (included) and 10000 (included)?

Code : select * from employee where salary>=9000 and salary<=10000;

id	name	salary	manager	birthyear	startyear
13	Edwards,Peter	9000	199	1928	1958
32	Smyth,Carlo	9050	199	1929	1967
98	Williams,Judy	9000	199	1935	1969
129	Thomas,Tom	10000	199	1941	1962

4 rows in set (0.00 sec)

4) What was the age of each employee when they started working (startyear)?

Code : select name , startyear-birthyear as age from employee;

name	age
Ross,Stanley	18
Ross,Stuart	1
Edwards,Peter	30
Thompson,Bob	40
Smyth,Carlo	38
Hayes,Evelyn	32
Evans,Michael	22
Raveen,Lemont	24
James,Mary	49
Williams,Judy	34
Thomas,Tom	21
Jones,Tim	20
Bullock,J.D	0
Collins,Joanne	21
Brunet,Paul C.	21
Schmidt,Herman	20
Iwano,Masahiro	26
Smith,Paul	21
Onstad,Richard	19
Zugnoni,Arthur A.	21
Choy,Wanda	23
Wallace,Maggie J.	19
Bailey,Chas M.	19
Bono,Sonny	24
Schwarz,Jason B.	15

25 rows in set (0.00 sec)

5) Which employees have a last name ending with “son”?

Code : select name from employee where name like "%son,%";

name
Thompson,Bob

1 row in set (0.00 sec)

6) Which items (note items, not parts) have been delivered by a supplier called
Fisher-Price? Formulate this query using a subquery in the where-clause.

Code : select name from item where supplier in (select id from supplier where name = "Fisher-Price");

```
+-----------------+
| name            |
+-----------------+
| Maze            |
| The `Feel` Book |
| Squeeze Ball    |
+-----------------+
3 rows in set (0.00 sec)
```

7) Formulate the same query as above, but without a subquery.
Code : select item.name from item join supplier on item.supplier=supplier.id and supplier.name="Fisher-Price";

```
+-----------------+
| name            |
+-----------------+
| Maze            |
| The `Feel` Book |
| Squeeze Ball    |
+-----------------+
3 rows in set (0.00 sec)
```

8) Show all cities that have suppliers located in them. Formulate this query using a
subquery in the where-clause.

Code : SELECT * FROM city WHERE id IN (SELECT city FROM supplier);

id	name	state
10	Amherst	Mass
21	Boston	Mass
100	New York	NY
106	White Plains	Neb
118	Hickville	Okla
303	Atlanta	Ga
537	Madison	Wisc
609	Paxton	Ill
752	Dallas	Tex
802	Denver	Colo
841	Salt Lake City	Utah
900	Los Angeles	Calif
921	San Diego	Calif
941	San Francisco	Calif
981	Seattle	Wash

15 rows in set (0.01 sec)

9) What is the name and color of the parts that are heavier than a card reader?
Formulate this query using a subquery in the where-clause. (The SQL query must
not contain the weight as a constant.)

Code : select name,color from parts where weight > (select weight from parts where name="card reader");

name	color
disk drive	black
tape drive	black
line printer	yellow
card punch	gray

4 rows in set (0.00 sec)

10) Formulate the same query as above, but without a subquery. (The query must not
contain the weight as a constant.)

Code : select part1.name,part1.color from parts as part1 join parts as part2 on part2.name="card reader" and part1.weight>part2.weight;

```
+--------------+--------+
| name         | color  |
+--------------+--------+
| disk drive   | black  |
| tape drive   | black  |
| line printer | yellow |
| card punch   | gray   |
+--------------+--------+
4 rows in set (0.00 sec)
```

11) What is the average weight of black parts?

Code : select avg(weight) as Avg_Weight from parts where color="black";

```
+------------+
| Avg_Weight |
+------------+
|   347.2500 |
+------------+
1 row in set (0.00 sec)
```

12) What is the total weight of all parts that each supplier in Massachusetts (“Mass”)
has delivered? Retrieve the name and the total weight for each of these suppliers.
Do not forget to take the quantity of delivered parts into account. Note that one
row should be returned for each supplier.

Code : select supplier.name,sum(parts.weight*supply.quan) as Weight_total
from supplier,parts,supply,city

where city.state = "Mass" and supply.part = parts.id and
supplier.city = city.id and supply.supplier=supplier.id
group by supplier.id;

name	Weight_total
DEC	3120
Fisher-Price	1135000

2 rows in set (0.02 sec)

13) Create a new relation (a table), with the same attributes as the table items using the
CREATE TABLE syntax where you define every attribute explicitly (i.e. not as a
copy of another table). Then fill the table with all items that cost less than the average
price for items. Remember to define primary and foreign keys in your table!

Code : CREATE TABLE item_duplicate
(
id INTEGER ,CONSTRAINT
pk_item_duplicate PRIMARY KEY (id),
name VARCHAR(20),
dept INTEGER NOT NULL,
price INTEGER,
qoh INTEGER CONSTRAINT ck_itemduplicate_qoh
CHECK (qoh >= 0),
supplier INTEGER NOT NULL
);
ALTER TABLE item_duplicate
ADD CONSTRAINT fk_itemduplicate_dept FOREIGN
KEY (dept) REFERENCES dept(id);

ALTER TABLE item_duplicate

ADD CONSTRAINT fk_itemduplicate_supplier FOREIGN KEY (supplier)
REFERENCES supplier (id);

INSERT INTO item_duplicate(select * from item where Price < (select avg(Price) from item));

```
mysql> select * from item_duplicate;
+-----+-----------------+------+-------+------+----------+
| id  | name            | dept | price | qoh  | supplier |
+-----+-----------------+------+-------+------+----------+
|  11 | Wash cloth      |    1 |    75 |  575 |      213 |
|  19 | Bellbottoms     |   43 |   450 |  600 |       33 |
|  21 | ABC Blocks      |    1 |   198 |  405 |      125 |
|  23 | 1 lb Box        |   10 |   215 |  100 |       42 |
|  25 | 2 lb Box,Mix    |   10 |   450 |   75 |       42 |
|  26 | Earrings        |   14 |  1000 |   20 |      199 |
|  43 | Maze            |   49 |   325 |  200 |       89 |
| 106 | Clock Book      |   49 |   198 |  150 |      125 |
| 107 | The `Feel` Book |   35 |   225 |  225 |       89 |
| 118 | Towels,Bath     |   26 |   250 | 1000 |      213 |
| 119 | Squeeze Ball    |   49 |   250 |  400 |       89 |
| 120 | Twin Sheet      |   26 |   800 |  750 |      213 |
| 165 | Jean            |   65 |   825 |  500 |       33 |
| 258 | Shirt           |   58 |   650 | 1200 |       33 |
+-----+-----------------+------+-------+------+----------+
14 rows in set (0.01 sec)
```

14) Create a view that contains the items that cost less than the average price for items.

Code : create view item_view as select * from item where Price < (select avg(Price) from item);

```
mysql> select * from item_view;
+-----+-----------------+------+-------+------+----------+
| id  | name            | dept | price | qoh  | supplier |
+-----+-----------------+------+-------+------+----------+
|  11 | Wash cloth      |    1 |    75 |  575 |      213 |
|  19 | Bellbottoms     |   43 |   450 |  600 |       33 |
|  21 | ABC Blocks      |    1 |   198 |  405 |      125 |
|  23 | 1 lb Box        |   10 |   215 |  100 |       42 |
|  25 | 2 lb Box,Mix    |   10 |   450 |   75 |       42 |
|  26 | Earrings        |   14 |  1000 |   20 |      199 |
|  43 | Maze            |   49 |   325 |  200 |       89 |
| 106 | Clock Book      |   49 |   198 |  150 |      125 |
| 107 | The `Feel` Book |   35 |   225 |  225 |       89 |
| 118 | Towels,Bath     |   26 |   250 | 1000 |      213 |
| 119 | Squeeze Ball    |   49 |   250 |  400 |       89 |
| 120 | Twin Sheet      |   26 |   800 |  750 |      213 |
| 165 | Jean            |   65 |   825 |  500 |       33 |
| 258 | Shirt           |   58 |   650 | 1200 |       33 |
+-----+-----------------+------+-------+------+----------+
14 rows in set (0.01 sec)
```

15) What is the difference between a table and a view?

= >

1. A view is a database object that allows generating a logical subset of data from one or more tables, while a table is a database object or an entity that stores the data of a database. Thus, this explains the main difference between view and table.

2. Table is a physical entity that means data is actually stored in the table whereas view is a virtual entity, which means data is not actually stored in the table.

3. Furthermore, the view depends on the table, while the table is an independent data object.

4. Table occupies space on the system whereas view doesn’t require any space in the system.

One is static and the other is dynamic. Which is which and what do we mean by static respectively dynamic?

View is dynamic and table is static because view is a virtual table which are compiled during runtime and it combines the attributes of one or more tables whereas table need to be created before executing any query and it stores informations of a paticular entity.

Example : In a flight ticket booking system a customer only needs to view required informations such as flight time,ticket price ,flight name,offers,benifits,reviews,ratings,terminal number ,so the admin of the ticket booking system can only grant the access to that particular useful informations to the customers as a result it will create a view for the customers. Whereas for any bussiness partner the admin grant access to some more infomations such as monthly booking,advertisments,revenue,ratings,other bussiness partners etc, so the admin will create another view for the bussiness partner. So it can be noticed the admin is not granting access to whole datebase for a particular user but creating a view so that the privacy of other informations can be maintained. The view is getting created by combining the columns of one or more tables as per requirements so it is dynamic whereas a table stores details of a particular entity and as it is static so it can’t be directly exposed to the user.

16) Create a view, using only the implicit join notation, i.e. only use where statements but no inner join, right join or left join statements, that calculates the total cost of each debit, by considering price and quantity of each bought item. (To be used for charging customer accounts). The view should contain the sale identifier (debit) and total cost.

Code : create view view1 as select sale.debit,sum(sale.quantity*item.price) as total_cost from sale,item where sale.item=item.id group by(sale.debit);

```
mysql> select * from view1;
+--------+----------------+
| debit  | total_costcost |
+--------+----------------+
| 100581 |           2050 |
| 100582 |           1000 |
| 100586 |          13446 |
| 100592 |            650 |
| 100593 |            430 |
| 100594 |           3295 |
+--------+----------------+
6 rows in set (0.00 sec)
```

17) Do the same as in (16), using only the explicit join notation, i.e. using only left, right or inner joins but no where statement. Motivate why you use the join you do (left, right or inner), and why this is the correct one (unlike the others).

Code : create view view2 as select sale.debit,sum(sale.quantity*item.price) as total_cost from sale inner join item on sale.item=item.id group by(sale.debit);

```
mysql> select * from view2;
+--------+------------+
| debit  | total_cost |
+--------+------------+
| 100581 |       2050 |
| 100582 |       1000 |
| 100586 |      13446 |
| 100592 |        650 |
| 100593 |        430 |
| 100594 |       3295 |
+--------+------------+
6 rows in set (0.00 sec)
```

Join is used to combine the rows of two or more tables based on some condition. It is usually helpful when the tables of a database are stored in normalized form and user requires some certain quries which requires the combination of tables.

A join is better than others as it creates a instance of a new table combining the attributes of more than one table which makes the quries look simple,easy and understandable.