**DATABASE TEAM BASED ASSIGNMENT**

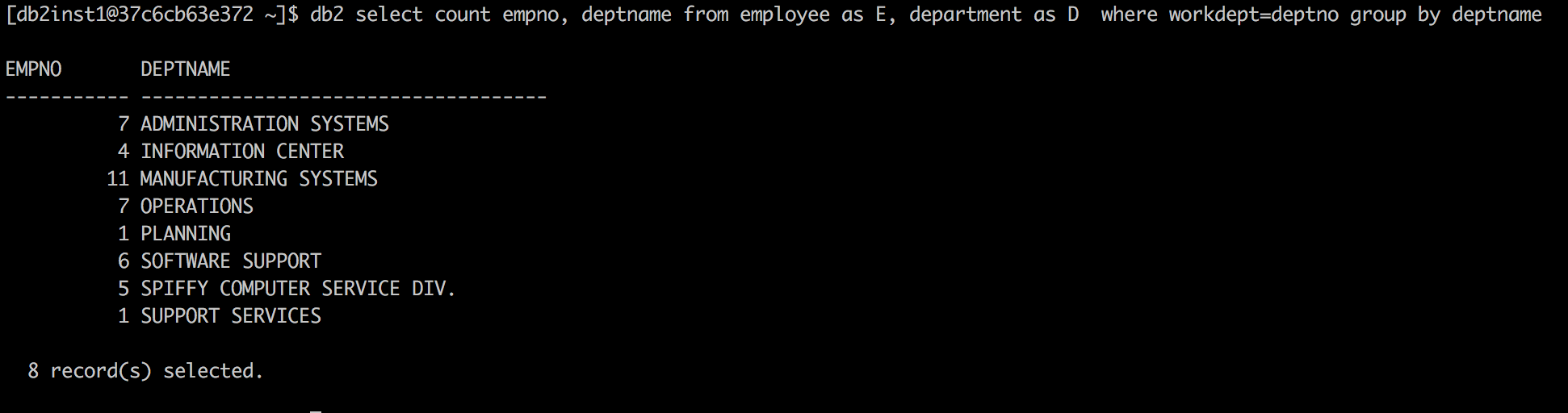
**PROJECT TEAM NUMBER: 16**

**DB2 EXPRESS C**

**QUERY:**

db2 select count empno, deptname from employee as E, department as D where workdept=deptno group by deptname

**QUERY OUTPUT:**

****

**EXPLAIN PLAN QUERY:** db2expln -d sample -z \; -f query.sql -t

**QUERY EXPLAIN PLAN:**

==================== STATEMENT ==========================================

Isolation Level = Cursor Stability

Blocking = Block Unambiguous Cursors

Query Optimization Class = 5

Partition Parallel = No

Intra-Partition Parallel = No

SQL Path = "SYSIBM", "SYSFUN", "SYSPROC", "SYSIBMADM",

"DB2INST1"

Statement:

select count (empno), deptname

from employee as E, department as D

where workdept=deptno

group by deptname

Section Code Page = 1208

Estimated Cost = 6.852824

Estimated Cardinality = 8.000000

Access Table Name = DB2INST1.EMPLOYEE ID = 2,6

| Index Scan: Name = DB2INST1.XEMP2 ID = 2

| | Regular Index (Not Clustered)

| | Index Columns:

| | | 1: WORKDEPT (Ascending)

| #Columns = 1

| Skip Inserted Rows

| Avoid Locking Committed Data

| Currently Committed for Cursor Stability

| #Key Columns = 0

| | Start Key: Beginning of Index

| | Stop Key: End of Index

| Index-Only Access

| Index Prefetch: None

| Lock Intents

| | Table: Intent Share

| | Row : Next Key Share

| Sargable Index Predicate(s)

| | Process Build Table for Hash Join

Hash Join

| Early Out: Single Match Per Inner Row

| Estimated Build Size: 4000

| Estimated Probe Size: 4000

| Access Table Name = DB2INST1.DEPARTMENT ID = 2,5

| | #Columns = 2

| | Skip Inserted Rows

| | Avoid Locking Committed Data

| | Currently Committed for Cursor Stability

| | May participate in Scan Sharing structures

| | Scan may start anywhere and wrap, for completion

| | Fast scan, for purposes of scan sharing management

| | Scan can be throttled in scan sharing management

| | Relation Scan

| | | Prefetch: Eligible

| | Lock Intents

| | | Table: Intent Share

| | | Row : Next Key Share

| | Sargable Predicate(s)

| | | Process Probe Table for Hash Join

Insert Into Sorted Temp Table ID = t1

| #Columns = 2

| #Sort Key Columns = 1

| | Key 1: DEPTNAME (Ascending)

| Sortheap Allocation Parameters:

| | #Rows = 8.000000

| | Row Width = 32

| Piped

| Buffered Partial Aggregation

Access Temp Table ID = t1

| #Columns = 2

| Relation Scan

| | Prefetch: Eligible

Final Aggregation

| Group By

| Column Function(s)

Return Data to Application

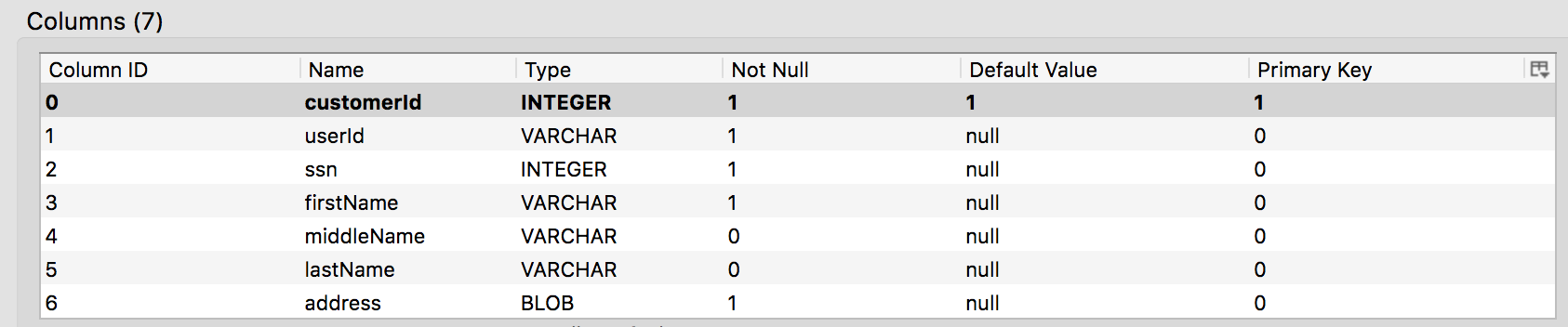
| #Columns = 2

End of section

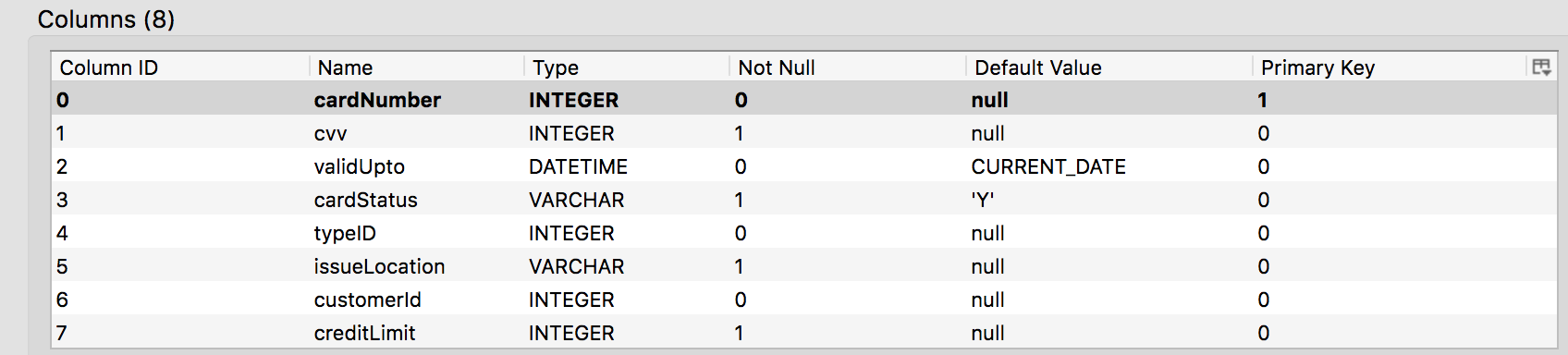
**SQLite: Credit Card Fraud Detection Database and Queries**

**DATABASE**

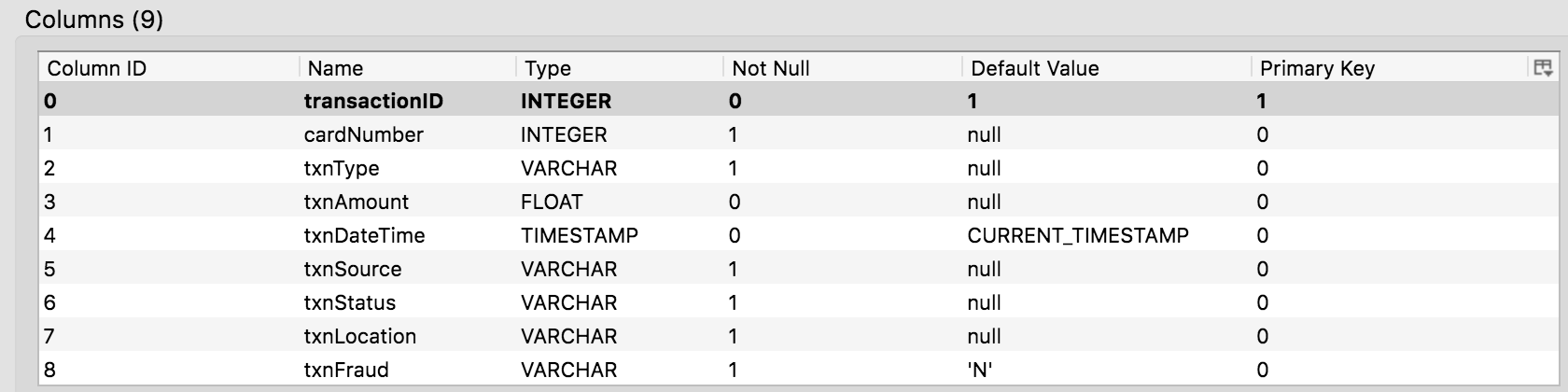
**cust\_account\_details**:



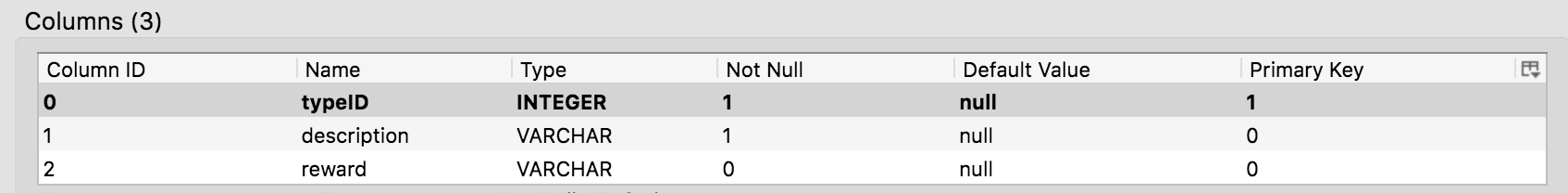
**credit\_card\_details:**



**credit\_card\_transaction:**



**credit\_card\_type:**



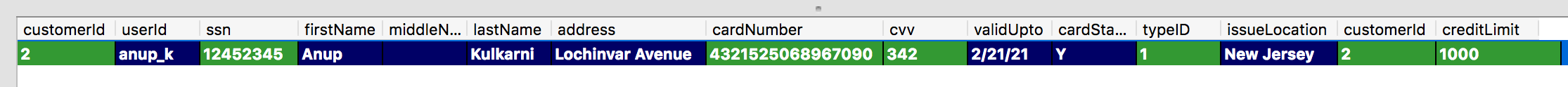
**SAMPLE FRAUD SCENARIO**

Customer having card number 4321525068967090 and amount 500 and date 2016-12-24 reported a Fraud transaction

Queries needed to get the details of Fraudulent transaction:

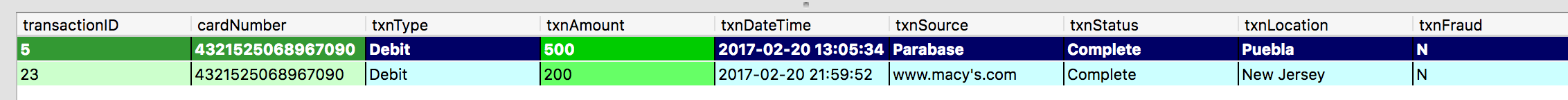
**Query 1:** Fetching customer and card details from cust\_account\_details and credit\_card\_details

***Select \* from cust\_account\_details cad, credit\_card\_details ccd where ccd.cardNumber= 4321525068967090 AND ccd.customerId= cad.customerId;***



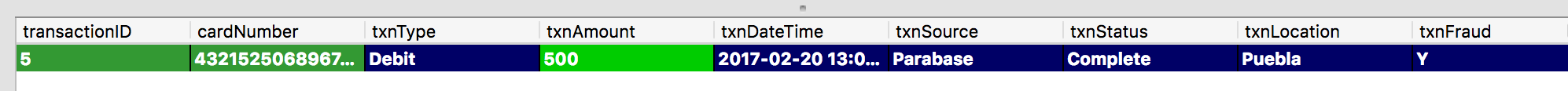
**Query 2:** Fetching latest transaction details from credit\_card\_transaction table for the customer depending on customer input

***Select \* from credit\_card\_transaction where cardNumber=4321525068967090 and txnDateTime like “2017-02-20%“;***



**Query 3:** After verifying the backend data based on the customer’s input, decision about the fraud is taken and the database is updated accordingly.

***Update credit\_card\_transaction set txnFraud="Y" where cardNumber= 4321525068967090 and txnAmount=500 and txnDateTime like "2017-02-20%"***

**

**IBM Graph**

**Schema creation request:**

****

**Schema creation response:**

****

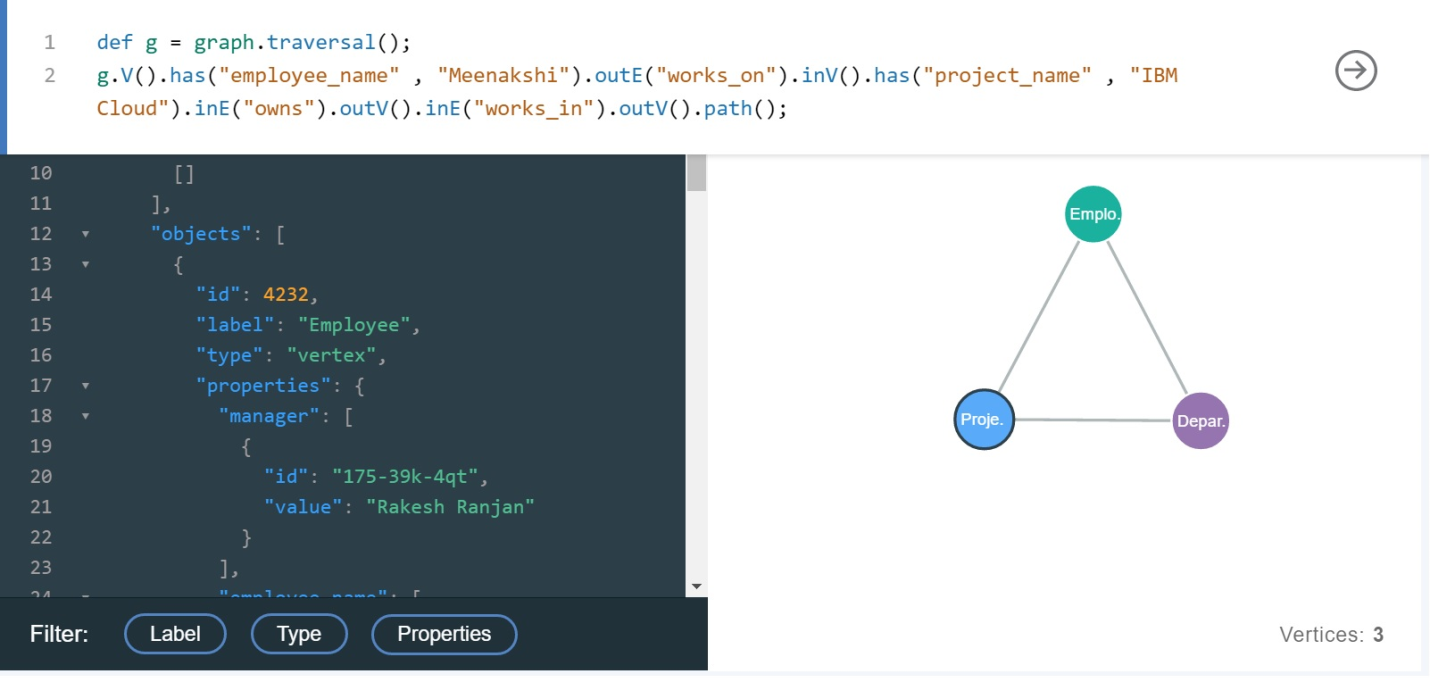
**Graph Traversal query:**

***def g = graph.traversal();***

***g.V().has("employee\_name" , "Meenakshi").outE("works\_on").inV().has("project\_name" , "IBM Cloud").inE("owns").outV().inE("works\_in").outV().path();***

**Traversal Query Response Json and Graph:**

****

****