DATABASE TEAM BASED ASSIGNMENT

Submitted by: Team 20

- Arshdeep Singh
- Dishant Kimtani
- Suhel Mehta
- Shikhar Gaur

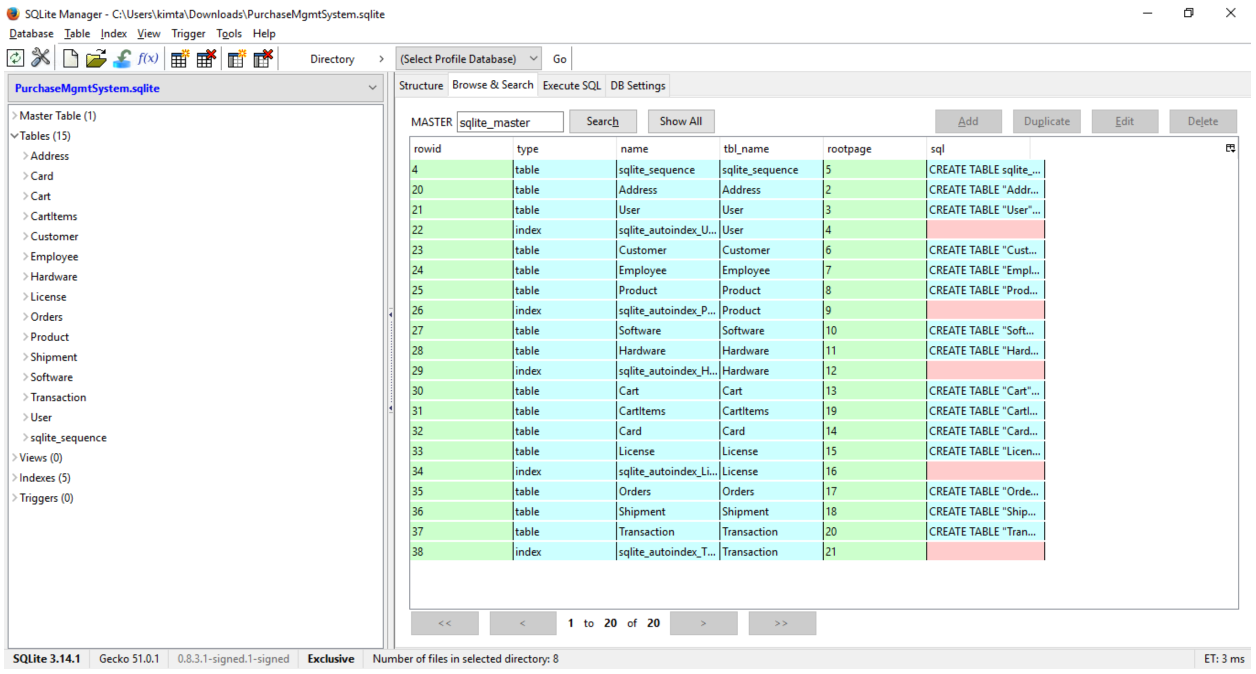
SQLite

Description:

Database **'ITShop'** was designed to support operations in a purchase order management for a company in software distribution. The company deals in with categories software and hardware and a customer can order software by placing an order.

Infrastructure:

SQLite Add-ons for Firefox was used to create the schema and design queries.


SQLite Manager - C:\Users\kimta\Downloads\PurchaseMgmtSystem.sqlite
Database Table Index View Trigger Tools Help
Directory
(Select Profile Database)
Go
PurchaseMgmtSystem.sqlite
Structure Browse & Search Execute SQL DB Settings
MASTER
Search
Show All
Add Duplicate Edit Delete

Master Table (1)
Tables (15)
Address
Card
Cart
CartItems
Customer
Employee
Hardware
License
Orders
Product
Shipment
Software
Transaction
User
sqlite_sequence
Views (0)
Indexes (5)
Triggers (0)

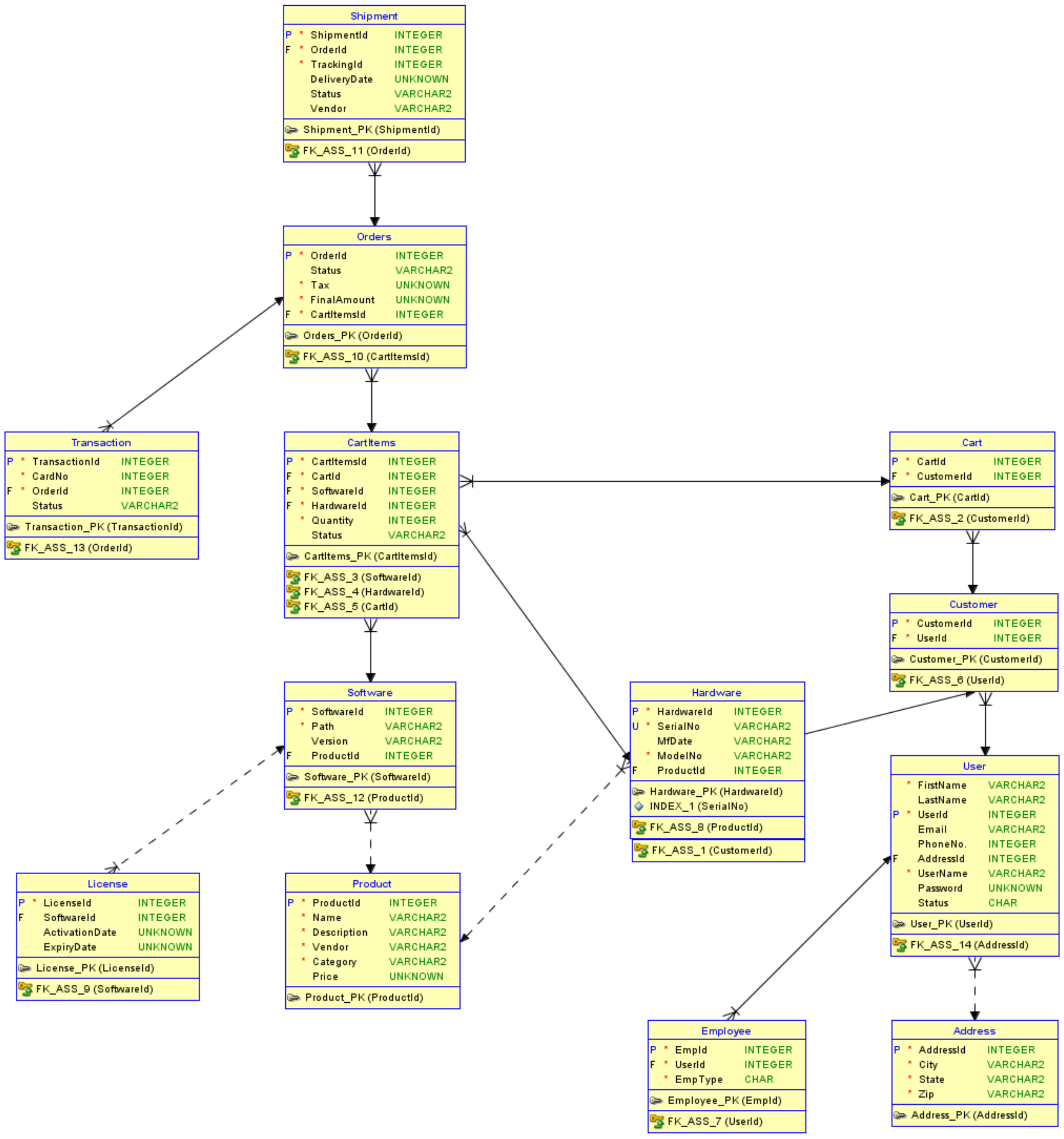
MASTER sqlite_master

rowid	type	name	tbl_name	rootpage	sql
4	table	sqlite_sequence	sqlite_sequence	5	CREATE TABLE sqlite_...
20	table	Address	Address	2	CREATE TABLE "Addr...
21	table	User	User	3	CREATE TABLE "User"...
22	index	sqlite_autoindex_U...	User	4	
23	table	Customer	Customer	6	CREATE TABLE "Cust...
24	table	Employee	Employee	7	CREATE TABLE "Empl...
25	table	Product	Product	8	CREATE TABLE "Prod...
26	index	sqlite_autoindex_P...	Product	9	
27	table	Software	Software	10	CREATE TABLE "Soft...
28	table	Hardware	Hardware	11	CREATE TABLE "Hard...
29	index	sqlite_autoindex_H...	Hardware	12	
30	table	Cart	Cart	13	CREATE TABLE "Cart"...
31	table	CartItems	CartItems	19	CREATE TABLE "CartI...
32	table	Card	Card	14	CREATE TABLE "Card...
33	table	License	License	15	CREATE TABLE "Licen...
34	index	sqlite_autoindex_Li...	License	16	
35	table	Orders	Orders	17	CREATE TABLE "Orde...
36	table	Shipment	Shipment	18	CREATE TABLE "Ship...
37	table	Transaction	Transaction	20	CREATE TABLE "Tran...
38	index	sqlite_autoindex_T...	Transaction	21	

<< < 1 to 20 of 20 > >>

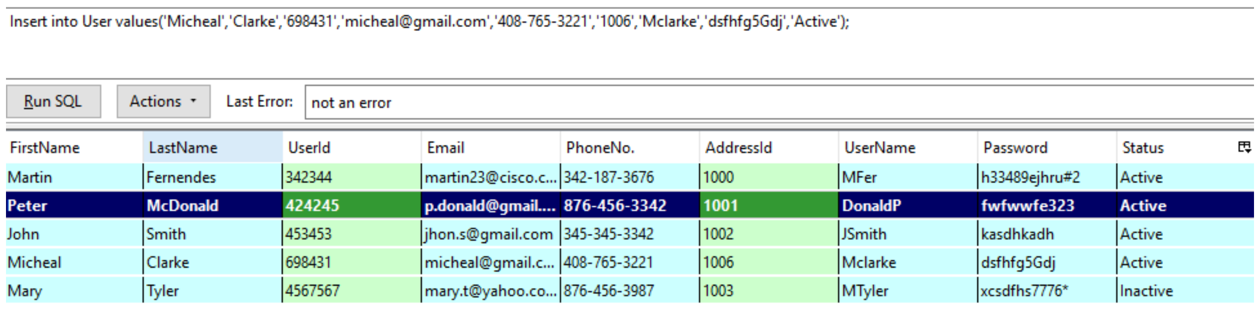
SQLite 3.14.1 | Gecko 51.0.1 | 0.8.3.1-signed.1-signed | Exclusive | Number of files in selected directory: 8 | ET: 3 ms

Database Model:



Operations:

- Creation of User:

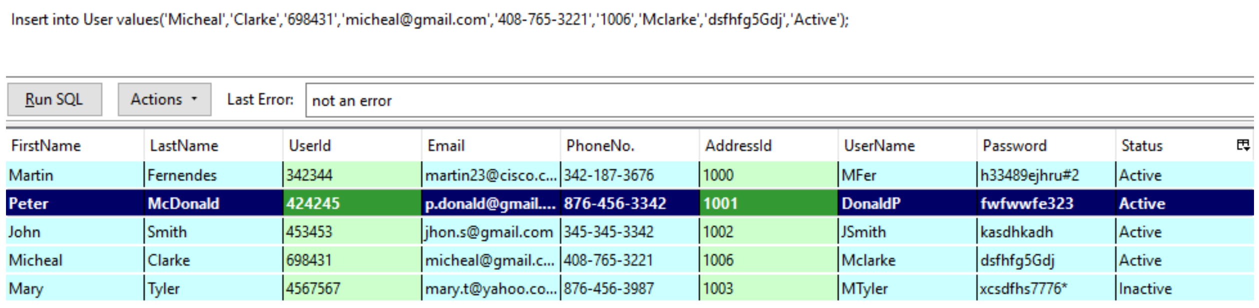


```
Insert into User values('Micheal','Clarke','698431','micheal@gmail.com','408-765-3221','1006','Mclarke','dsfhfg5Gdj','Active');
```

Last Error: not an error

FirstName	LastName	UserId	Email	PhoneNo.	AddressId	UserName	Password	Status
Martin	Fernendes	342344	martin23@cisco.c...	342-187-3676	1000	MFer	h33489ejhru#2	Active
Peter	McDonald	424245	p.donald@gmail....	876-456-3342	1001	DonaldP	fwfwwfe323	Active
John	Smith	453453	jhon.s@gmail.com	345-345-3342	1002	JSmith	kasdhkadh	Active
Micheal	Clarke	698431	micheal@gmail.c...	408-765-3221	1006	Mclarke	dsfhfg5Gdj	Active
Mary	Tyler	4567567	mary.t@yahoo.co...	876-456-3987	1003	MTyler	xcsdfhs7776*	Inactive

- User selecting one of the Product from software subcategory:

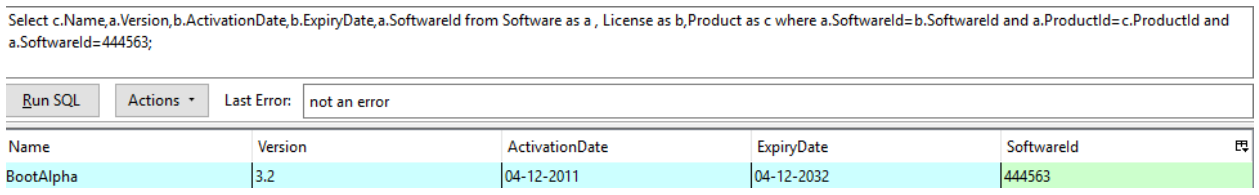


```
Insert into User values('Micheal','Clarke','698431','micheal@gmail.com','408-765-3221','1006','Mclarke','dsfhfg5Gdj','Active');
```

Last Error: not an error

FirstName	LastName	UserId	Email	PhoneNo.	AddressId	UserName	Password	Status
Martin	Fernendes	342344	martin23@cisco.c...	342-187-3676	1000	MFer	h33489ejhru#2	Active
Peter	McDonald	424245	p.donald@gmail....	876-456-3342	1001	DonaldP	fwfwwfe323	Active
John	Smith	453453	jhon.s@gmail.com	345-345-3342	1002	JSmith	kasdhkadh	Active
Micheal	Clarke	698431	micheal@gmail.c...	408-765-3221	1006	Mclarke	dsfhfg5Gdj	Active
Mary	Tyler	4567567	mary.t@yahoo.co...	876-456-3987	1003	MTyler	xcsdfhs7776*	Inactive

- User Checking the details of the Software Product he chooses(In this case , it is BootAlpha):

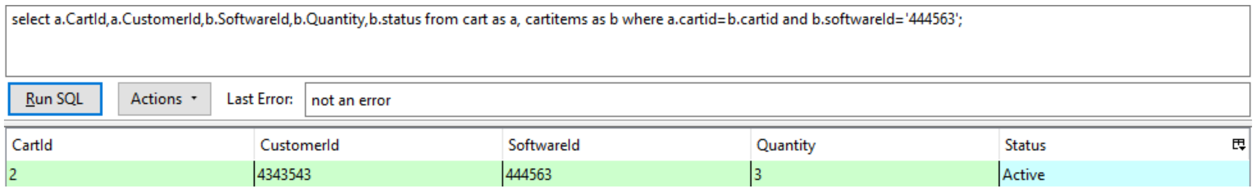


```
Select c.Name,a.Version,b.ActivationDate,b.ExpiryDate,a.SoftwareId from Software as a , License as b,Product as c where a.SoftwareId=b.SoftwareId and a.ProductId=c.ProductId and a.SoftwareId=444563;
```

Last Error: not an error

Name	Version	ActivationDate	ExpiryDate	SoftwareId
BootAlpha	3.2	04-12-2011	04-12-2032	444563

- The items are placed into the cart which customer has selected:

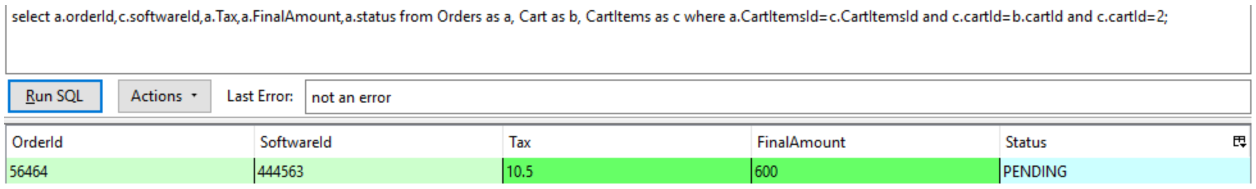


```
select a.CartId,a.CustomerId,b.SoftwareId,b.Quantity,b.status from cart as a, cartitems as b where a.cartid=b.cartid and b.softwareId='444563';
```

Last Error: not an error

CartId	CustomerId	SoftwareId	Quantity	Status
2	4343543	444563	3	Active

- The generated order is as follows:

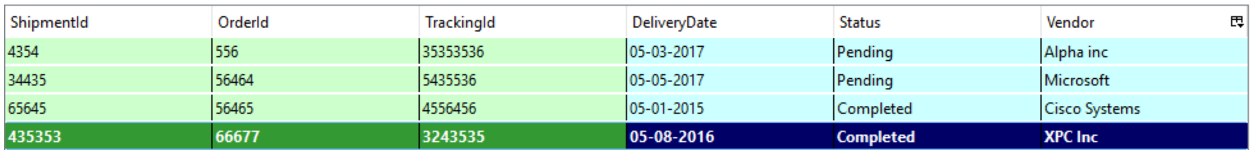


```
select a.orderId,c.softwareId,a.Tax,a.FinalAmount,a.status from Orders as a, Cart as b, CartItems as c where a.CartItemsId=c.CartItemsId and c.cartId=b.cartId and c.cartId=2;
```

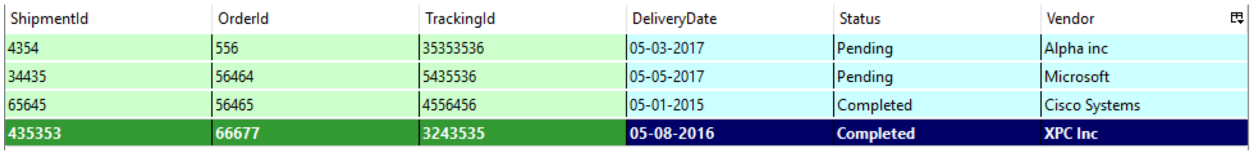
Run SQL | Actions | Last Error: not an error

OrderId	SoftwareId	Tax	FinalAmount	Status
56464	444563	10.5	600	PENDING

- Then a Transaction and shipment detail is generated for the respective ordered item:



ShipmentId	OrderId	TrackingId	DeliveryDate	Status	Vendor
4354	556	35353536	05-03-2017	Pending	Alpha inc
34435	56464	5435536	05-05-2017	Pending	Microsoft
65645	56465	4556456	05-01-2015	Completed	Cisco Systems
435353	66677	3243535	05-08-2016	Completed	XPC Inc



ShipmentId	OrderId	TrackingId	DeliveryDate	Status	Vendor
4354	556	35353536	05-03-2017	Pending	Alpha inc
34435	56464	5435536	05-05-2017	Pending	Microsoft
65645	56465	4556456	05-01-2015	Completed	Cisco Systems
435353	66677	3243535	05-08-2016	Completed	XPC Inc

DB2 Express C

Description:

A sample database was created in DB2 using the db2sample command and a query plan was generated for a query using where and group by clause.

Commands:

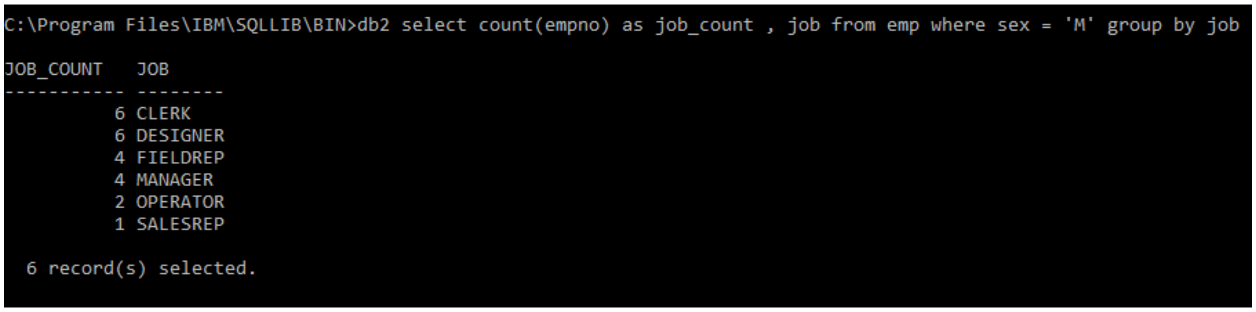
- Sample database creation:

  db2sampl -dbpath E -name sample -sql -force -verbose
  db2 connect to sample

- Explain plan:

  db2 -tf EXPLAIN.DDL in sqllib
  db2 set current explain mode yes
  db2 set current explain snapshot yes

  db2 select count(empno) as job_count , job from emp where sex = 'M' group by job
  db2exfmt



```
C:\Program Files\IBM\SQLLIB\BIN>db2 select count(empno) as job_count , job from emp where sex = 'M' group by job

JOB_COUNT   JOB
----------- --------
          6 CLERK
          6 DESIGNER
          4 FIELDREP
          4 MANAGER
          2 OPERATOR
          1 SALESREP

  6 record(s) selected.
```

Explain Plan:

The generated explain plan has been attached as a separate file in the submission.

IBM Graph Datastore

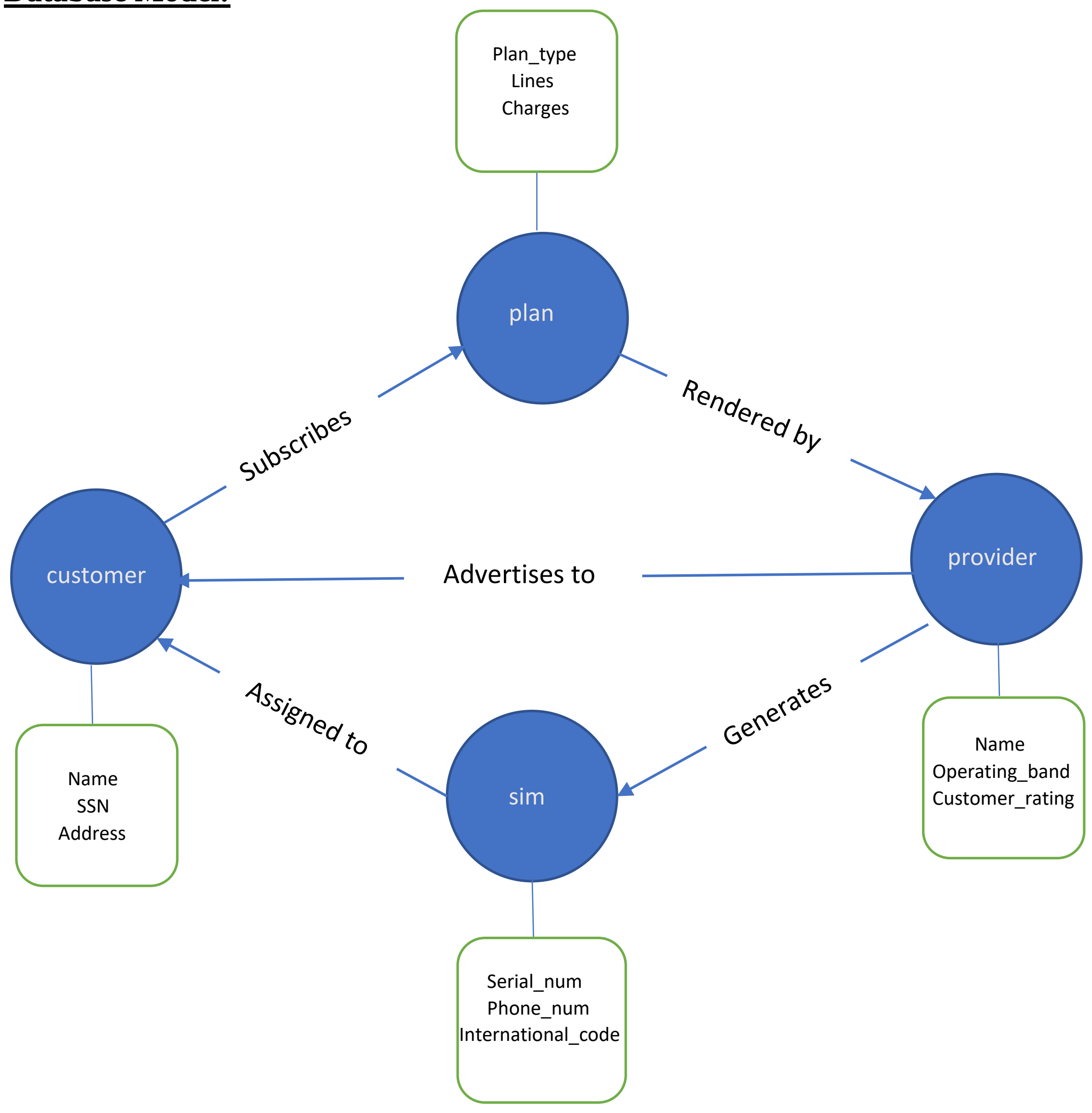
Description:

A graph database was created to store data of customers subscribing to different cellular plans rendered by different vendors. The IBM graph schema was created using the schema API and sample data was loaded into the created schema.

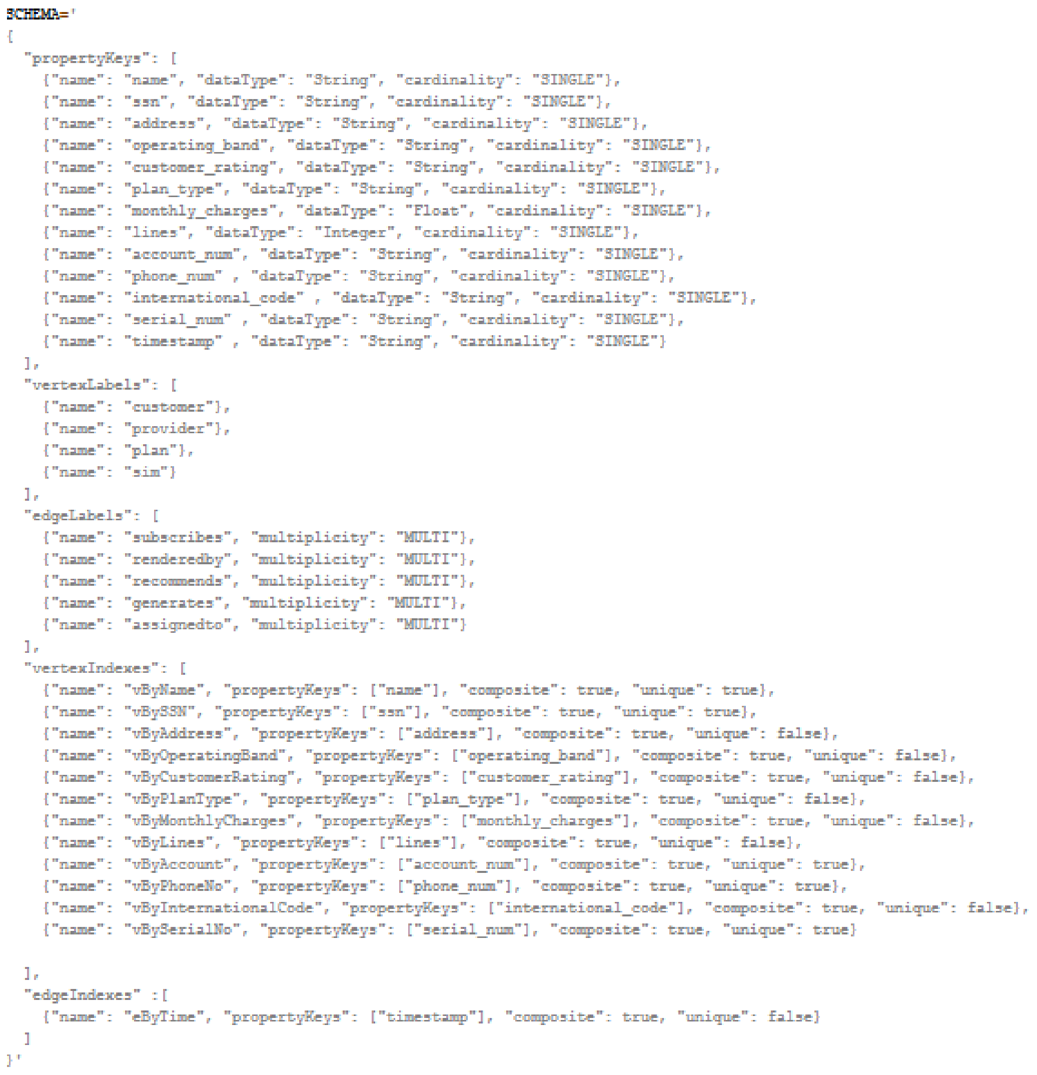
Infrastructure:

1. IBM Graph service
2. Curl
3. Jq

Database Model:



Schema Structure:

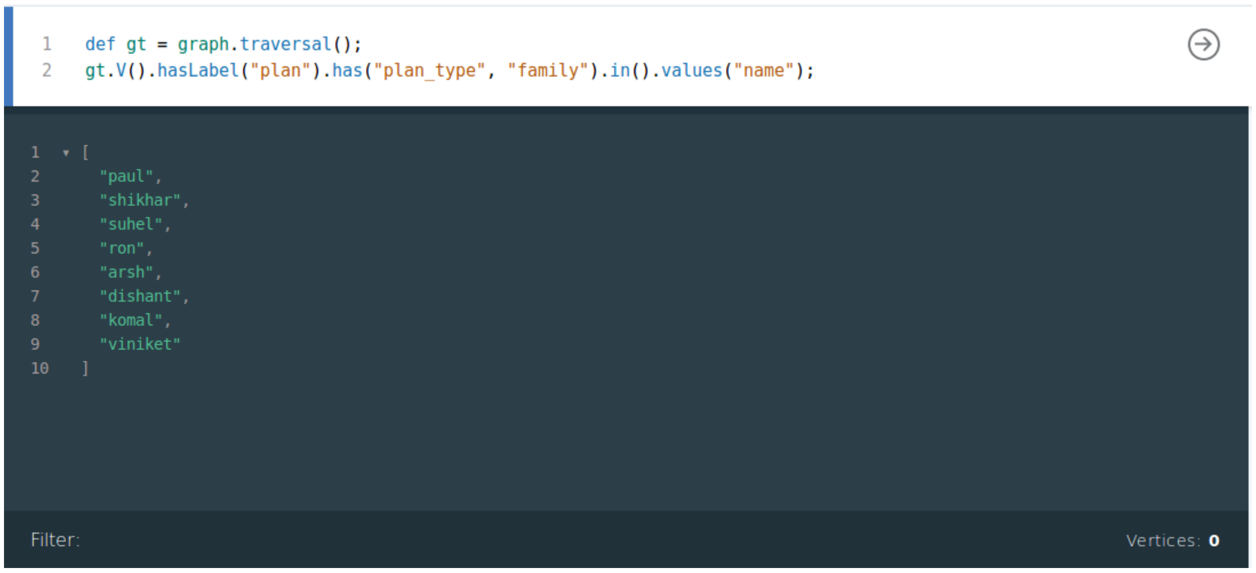


```
SCHEMA='
{
  "propertyKeys": [
    {"name": "name", "dataType": "String", "cardinality": "SINGLE"},
    {"name": "ssn", "dataType": "String", "cardinality": "SINGLE"},
    {"name": "address", "dataType": "String", "cardinality": "SINGLE"},
    {"name": "operating_band", "dataType": "String", "cardinality": "SINGLE"},
    {"name": "customer_rating", "dataType": "String", "cardinality": "SINGLE"},
    {"name": "plan_type", "dataType": "String", "cardinality": "SINGLE"},
    {"name": "monthly_charges", "dataType": "Float", "cardinality": "SINGLE"},
    {"name": "lines", "dataType": "Integer", "cardinality": "SINGLE"},
    {"name": "account_num", "dataType": "String", "cardinality": "SINGLE"},
    {"name": "phone_num" , "dataType": "String", "cardinality": "SINGLE"},
    {"name": "international_code" , "dataType": "String", "cardinality": "SINGLE"},
    {"name": "serial_num" , "dataType": "String", "cardinality": "SINGLE"},
    {"name": "timestamp" , "dataType": "String", "cardinality": "SINGLE"}
  ],
  "vertexLabels": [
    {"name": "customer"},
    {"name": "provider"},
    {"name": "plan"},
    {"name": "sim"}
  ],
  "edgeLabels": [
    {"name": "subscribes", "multiplicity": "MULTI"},
    {"name": "renderedby", "multiplicity": "MULTI"},
    {"name": "recommends", "multiplicity": "MULTI"},
    {"name": "generates", "multiplicity": "MULTI"},
    {"name": "assignedto", "multiplicity": "MULTI"}
  ],
  "vertexIndexes": [
    {"name": "vByName", "propertyKeys": ["name"], "composite": true, "unique": true},
    {"name": "vBySSN", "propertyKeys": ["ssn"], "composite": true, "unique": true},
    {"name": "vByAddress", "propertyKeys": ["address"], "composite": true, "unique": false},
    {"name": "vByOperatingBand", "propertyKeys": ["operating_band"], "composite": true, "unique": false},
    {"name": "vByCustomerRating", "propertyKeys": ["customer_rating"], "composite": true, "unique": false},
    {"name": "vByPlanType", "propertyKeys": ["plan_type"], "composite": true, "unique": false},
    {"name": "vByMonthlyCharges", "propertyKeys": ["monthly_charges"], "composite": true, "unique": false},
    {"name": "vByLines", "propertyKeys": ["lines"], "composite": true, "unique": false},
    {"name": "vByAccount", "propertyKeys": ["account_num"], "composite": true, "unique": true},
    {"name": "vByPhoneNo", "propertyKeys": ["phone_num"], "composite": true, "unique": true},
    {"name": "vByInternationalCode", "propertyKeys": ["international_code"], "composite": true, "unique": false},
    {"name": "vBySerialNo", "propertyKeys": ["serial_num"], "composite": true, "unique": true}

  ],
  "edgeIndexes" :[
    {"name": "eByTime", "propertyKeys": ["timestamp"], "composite": true, "unique": false}
  ]
}'
```

Sample Gremlin Traversals:

- List of all customers who have cellular plan as “family”.

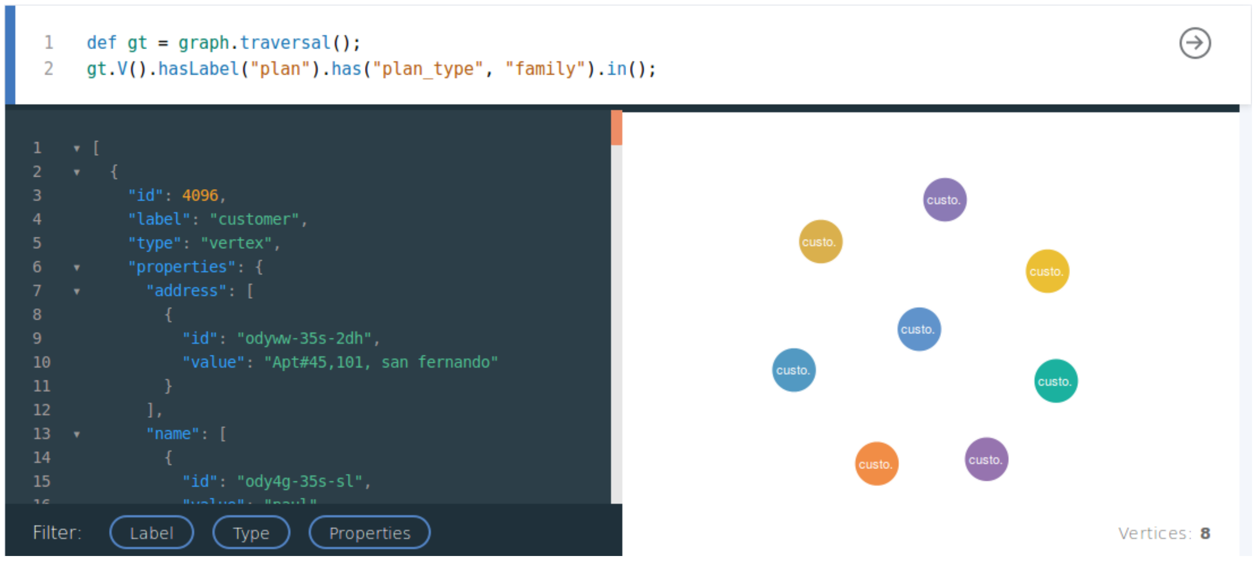


```
def gt = graph.traversal();
gt.V().hasLabel("plan").has("plan_type", "family").in().values("name");
```

```
[
  "paul",
  "shikhar",
  "suhel",
  "ron",
  "arsh",
  "dishant",
  "komal",
  "viniket"
]
```

Filter: Vertices: **0**

- List of all customers who have cellular plan as “family”

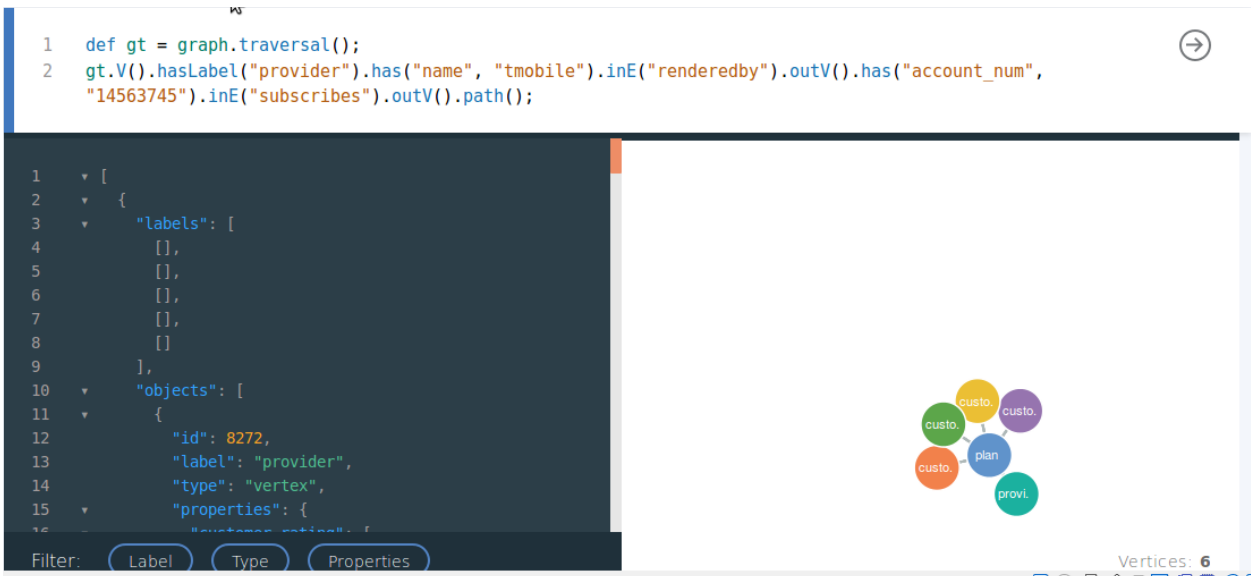


```
def gt = graph.traversal();
gt.V().hasLabel("plan").has("plan_type", "family").in();
```

```
[
  {
    "id": 4096,
    "label": "customer",
    "type": "vertex",
    "properties": {
      "address": [
        {
          "id": "odyww-35s-2dh",
          "value": "Apt#45,101, san fernando"
        }
      ],
      "name": [
        {
          "id": "ody4g-35s-sl",
```

Filter: Label Type Properties   Vertices: **8**

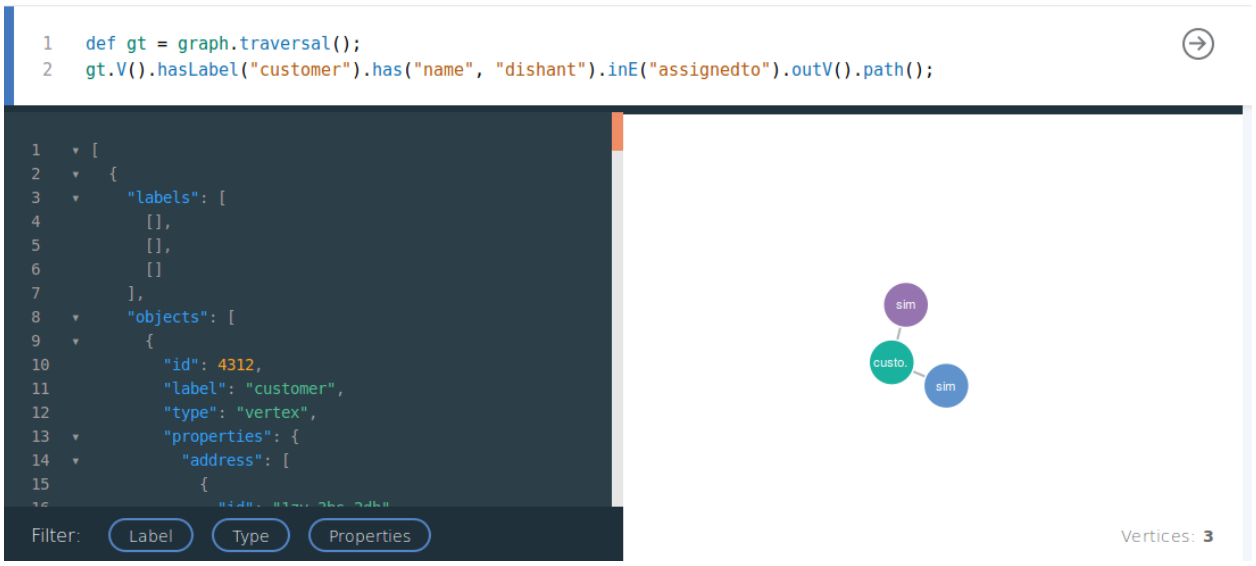
- List of customers who have subscribed for same cellular plan (account_num = “14563745”) and have provider as “tmobile”



```
def gt = graph.traversal();
gt.V().hasLabel("provider").has("name", "tmobile").inE("renderedby").outV().has("account_num",
"14563745").inE("subscribes").outV().path();
```

```
[
  {
    "labels": [
      [],
      [],
      [],
      [],
      []
    ],
    "objects": [
      {
        "id": 8272,
        "label": "provider",
        "type": "vertex",
        "properties": {
```

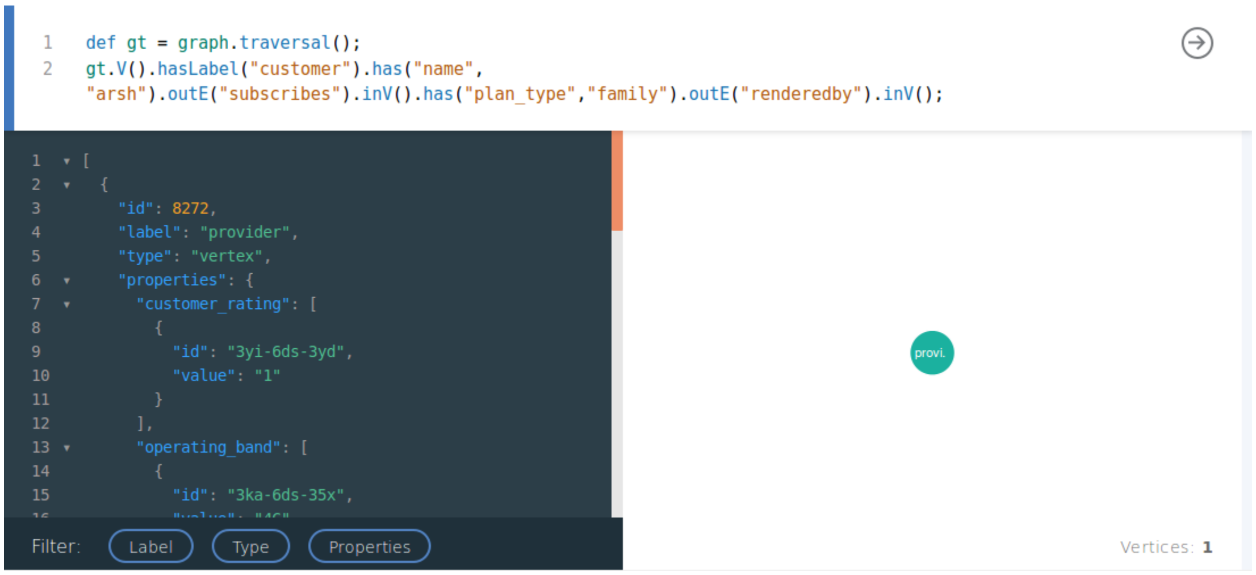
- Get all the sim assigned to customer “Dishant”.



```
def gt = graph.traversal();
gt.V().hasLabel("customer").has("name", "dishant").inE("assignedto").outV().path();
```

```
[
  {
    "labels": [
      [],
      [],
      []
    ],
    "objects": [
      {
        "id": 4312,
        "label": "customer",
        "type": "vertex",
        "properties": {
          "address": [
            {
```

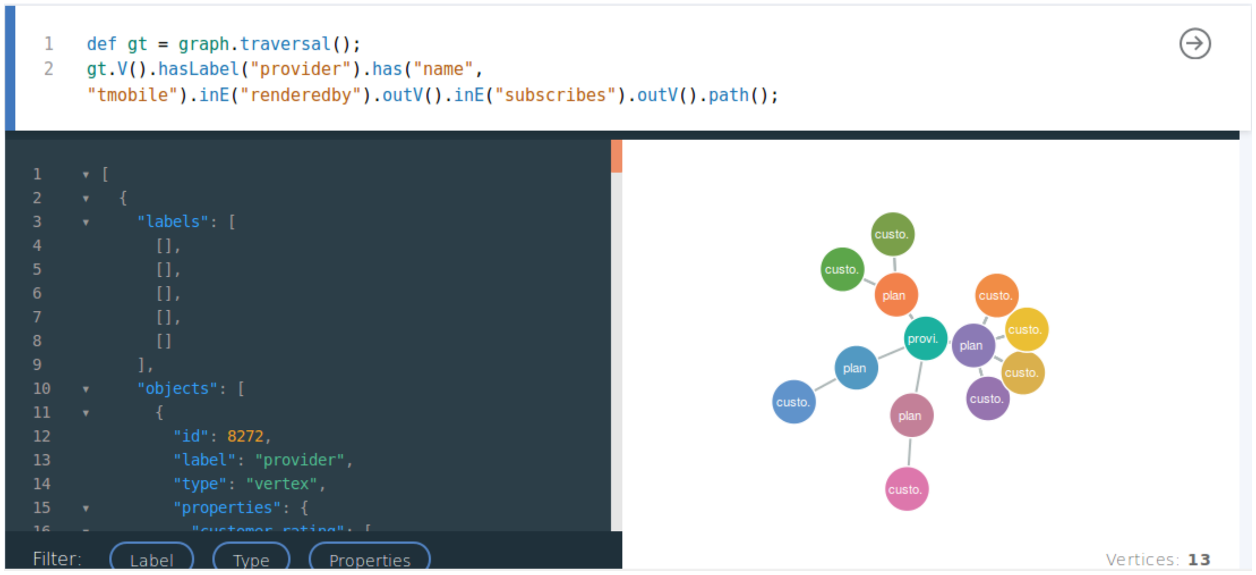
- Get “family” plan cellular connection provider for customer “arsh”



```
def gt = graph.traversal();
gt.V().hasLabel("customer").has("name",
"arsh").outE("subscribes").inV().has("plan_type","family").outE("renderedby").inV();
```

```
[
  {
    "id": 8272,
    "label": "provider",
    "type": "vertex",
    "properties": {
      "customer_rating": [
        {
          "id": "3yi-6ds-3yd",
          "value": "1"
        }
      ],
      "operating_band": [
        {
          "id": "3ka-6ds-35x",
```

- Get all customers for service provider “tmobile”.



```
def gt = graph.traversal();
gt.V().hasLabel("provider").has("name",
"tmobile").inE("renderedby").outV().inE("subscribes").outV().path();
```

```
[
  {
    "labels": [
      [],
      [],
      [],
      [],
      []
    ],
    "objects": [
      {
        "id": 8272,
        "label": "provider",
        "type": "vertex",
        "properties": {
```