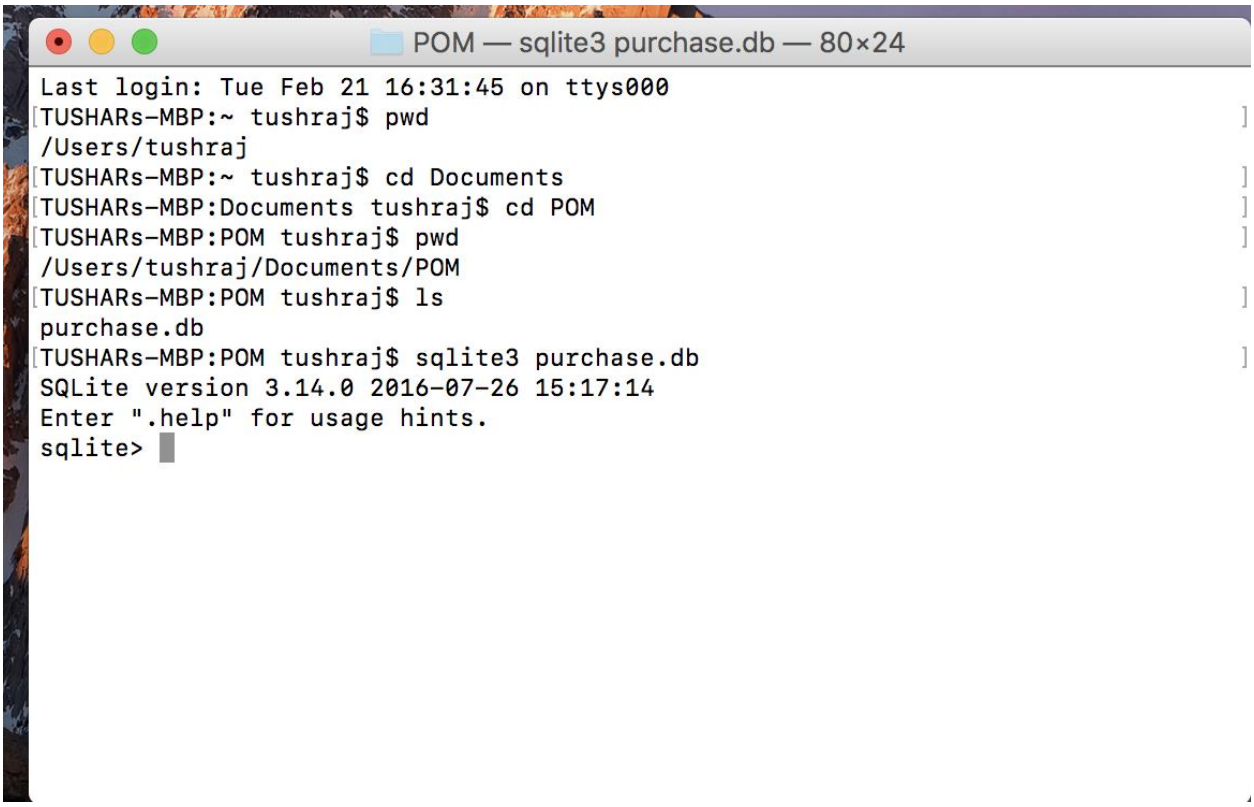
TEAM 17

DATABASE ASSIGNMENT

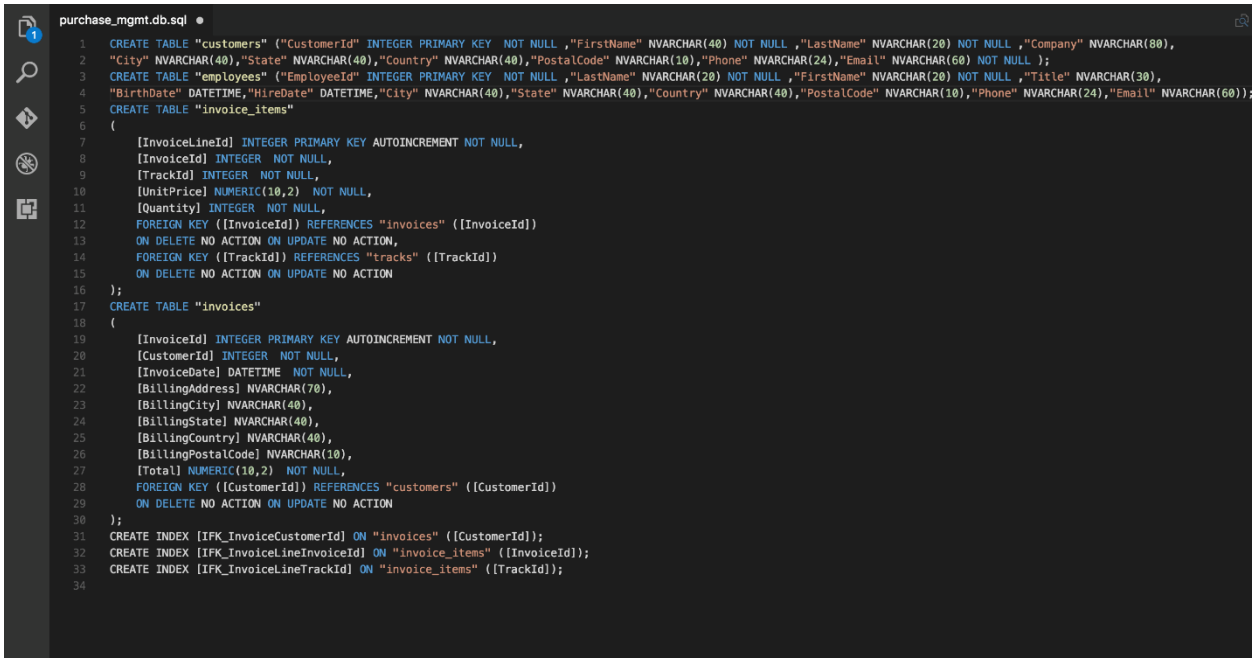
SQLite Manager

Steps followed:

1. Installation on SQlite as an add-on for Firefox.
2. Designed a database for Purchase order management system.



3. Creating a sample schema with the following tables:



4. Following script was run while creating the sample schema:

CREATE TABLE "customers" ("CustomerId" INTEGER PRIMARY KEY NOT NULL ,"FirstName" NVARCHAR(40) NOT NULL ,"LastName" NVARCHAR(20) NOT NULL ,"Company" NVARCHAR(80),"City" NVARCHAR(40),"State" NVARCHAR(40),"Country" NVARCHAR(40),"PostalCode" NVARCHAR(10),"Phone" NVARCHAR(24),"Email" NVARCHAR(60) NOT NULL);

CREATE TABLE "employees" ("EmployeeId" INTEGER PRIMARY KEY NOT NULL ,"LastName" NVARCHAR(20) NOT NULL ,"FirstName" NVARCHAR(20) NOT NULL ,"Title" NVARCHAR(30),"BirthDate" DATETIME,"HireDate" DATETIME,"City" NVARCHAR(40),"State" NVARCHAR(40),"Country" NVARCHAR(40),"PostalCode" NVARCHAR(10),"Phone" NVARCHAR(24),"Email" NVARCHAR(60));

CREATE TABLE "invoice_items"

(

[InvoiceLineId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

[InvoiceId] INTEGER NOT NULL,

[TrackId] INTEGER NOT NULL,

[UnitPrice] NUMERIC(10,2) NOT NULL,

[Quantity] INTEGER NOT NULL,

FOREIGN KEY ([InvoiceId]) REFERENCES "invoices" ([InvoiceId])

ON DELETE NO ACTION ON UPDATE NO ACTION,

```
    FOREIGN KEY ([TrackId]) REFERENCES "tracks" ([TrackId])
                ON DELETE NO ACTION ON UPDATE NO ACTION
);
CREATE TABLE "invoices"
(
    [InvoiceId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    [CustomerId] INTEGER  NOT NULL,
    [InvoiceDate] DATETIME  NOT NULL,
    [BillingAddress] NVARCHAR(70),
    [BillingCity] NVARCHAR(40),
    [BillingState] NVARCHAR(40),
    [BillingCountry] NVARCHAR(40),
    [BillingPostalCode] NVARCHAR(10),
    [Total] NUMERIC(10,2)  NOT NULL,
    FOREIGN KEY ([CustomerId]) REFERENCES "customers" ([CustomerId])
                ON DELETE NO ACTION ON UPDATE NO ACTION
);
CREATE INDEX [IFK_InvoiceCustomerId] ON "invoices" ([CustomerId]);
CREATE INDEX [IFK_InvoiceLineInvoiceId] ON "invoice_items" ([InvoiceId]);
CREATE INDEX [IFK_InvoiceLineTrackId] ON "invoice_items" ([TrackId]);
```

5. Insertion of sample data.

The sample data was sourced from here :

http://www.sqlitetutorial.net/sqlite-sample-database/

```
Last login: Tue Feb 21 23:34:33 on ttys000
TUSHARs-MBP:~ tushraj$ cd Downloads
TUSHARs-MBP:Downloads tushraj$ sqlite3 chinook.db_2
SQLite version 3.14.0 2016-07-26 15:17:14
Enter ".help" for usage hints.
sqlite> select * from customers;
1|Kara|Nielsen||Copenhagen||Denmark|1720|+453 3331 9991|kara.nielsen@jubii.dk
10|Eduardo|Martins|Woodstock Discos|São Paulo|SP|Brazil|01007-010|+55 (11) 3033-5446|eduardo@woodstock.com.br
11|Alexandre|Rocha|Banco do Brasil S.A.|São Paulo|SP|Brazil|01310-200|+55 (11) 3055-3278|alero@uol.com.br
12|Roberto|Almeida|Riotur|Rio de Janeiro|RJ|Brazil|20040-020|+55 (21) 2271-7000|roberto.almeida@riotur.gov.br
13|Fernanda|Ramos||Brasília|DF|Brazil|71020-677|+55 (61) 3363-5547|fernadaramos4@uol.com.br
14|Mark|Philips|Telus|Edmonton|AB|Canada|T6G 2C7|+1 (780) 434-4554|mphilips12@shaw.ca
15|Jennifer|Peterson|Rogers Canada|Vancouver|BC|Canada|V6C 1G8|+1 (604) 688-2255|jenniferp@rogers.ca
16|Frank|Harris|Google Inc.|Mountain View|CA|USA|94043-1351|+1 (650) 253-0000|fharris@google.com
17|Jack|Smith|Microsoft Corporation|Redmond|WA|USA|98052-8300|+1 (425) 882-8080|jacksmith@microsoft.com
18|Michelle|Brooks||New York|NY|USA|10012-2612|+1 (212) 221-3546|michelleb@aol.com
19|Tim|Goyer|Apple Inc.|Cupertino|CA|USA|95014|+1 (408) 996-1010|tgoyer@apple.com
20|Dan|Miller||Mountain View|CA|USA|94040-111|+1 (650) 644-3358|dmiller@comcast.com
21|Kathy|Chase||Reno|NV|USA|89503|+1 (775) 223-7665|kachase@hotmail.com
22|Heather|Leacock||Orlando|FL|USA|32801|+1 (407) 999-7788|hleacock@gmail.com
23|John|Gordon||Boston|MA|USA|2113|+1 (617) 522-1333|johngordon22@yahoo.com
24|Frank|Ralston||Chicago|IL|USA|60611|+1 (312) 332-3232|fralston@gmail.com
25|Victor|Stevens||Madison|WI|USA|53703|+1 (608) 257-0597|vstevens@yahoo.com
26|Richard|Cunningham||Fort Worth|TX|USA|76110|+1 (817) 924-7272|ricunningham@hotmail.com
27|Patrick|Gray||Tucson|AZ|USA|85719|+1 (520) 622-4200|patrick.gray@aol.com
28|Julia|Barnett||Salt Lake City|UT|USA|84102|+1 (801) 531-7272|jubarnett@gmail.com
29|Robert|Brown||Toronto|ON|Canada|M6J 1V1|+1 (416) 363-8888|robbrown@shaw.ca
30|Edward|Francis||Ottawa|ON|Canada|K2P 1L7|+1 (613) 234-3322|edfrancis@yachoo.ca
31|Martha|Silk||Halifax|NS|Canada|B3S 1C5|+1 (902) 450-0450|marthasilk@gmail.com
32|Aaron|Mitchell||Winnipeg|MB|Canada|R3L 2B9|+1 (204) 452-6452|aaronmitchell@yahoo.ca
42|Wyatt|Girard||Bordeaux||France|33000|+33 05 56 96 96 96|wyatt.girard@yahoo.fr
52|Emma|Jones||London||United Kingdom|N1 5LH|+44 020 7707 0707|emma_jones@hotmail.com
55|Mark|Taylor||Sidney|NSW|Australia|2010|+61 (02) 9332 3633|mark.taylor@yahoo.au
58|Manoj|Pareek||Delhi||India|110017|+91 0124 39883988|manoj.pareek@rediff.com
59|Puja|Srivastava||Bangalore||India|560001|+91 080 22289999|puja_srivastava@yahoo.in
sqlite>
```

6. Following queries were executed :

Query1:

```
sqlite> SELECT DISTINCT title from employees;
General Manager
Sales Manager
Sales Support Agent
IT Manager
IT Staff
sqlite>
sqlite> SELECT CustomerId from invoices WHERE BillingState = 'CA';
16
19
19
19
19
20
16
20
16
20
16
20
sqlite> SELECT Total from invoices WHERE BillingState = 'CA';
0.99
1.98
3.96
5.94
1.99
3.98
1.98
3.96
3.96
5.94
5.94
0.99
sqlite> SELECT Total from invoices WHERE BillingCountry = 'France';
0.99
1.99
3.96
5.94
8.91
0.99
1.98
1.98
16.86
1.98
3.96
13.86
5.94
8.91
8.91
0.99
1.98
sqlite>
```

Query2:

```
13.86
sqlite> SELECT Total from invoices WHERE BillingCountry LIKE 'C%';
0.99
1.98
8.91
3.96
0.99
1.98
3.96
5.94
1.98
3.96
0.99
5.94
13.86
5.94
1.98
1.98
16.86
0.99
3.96
0.99
8.91
5.94
0.99
1.98
1.98
8.91
13.86
1.98
1.98
3.96
13.86
3.96
5.94
0.99
1.98
25.86
5.94
sqlite>
```

Query3:

```
sqlite> INSERT INTO customers
   ...> (preference) VALUES('A1');
Error: table customers has no column named preference
sqlite> UPDATE employees
   ...> SET lastname = 'Smith'
   ...> WHERE
   ...>  employeeid = 3;
sqlite> SELECT
   ...>  employeeid,
   ...>  firstname,
   ...>  lastname,
   ...>  title,
   ...>  email
   ...> FROM
   ...>  employees
   ...> WHERE
   ...>  employeeid = 3;
3|Jane|Smith|Sales Support Agent|jane@chinookcorp.com
sqlite>
```

Query4:

```
sqlite> UPDATE employees
   ...> SET email = lower(
   ...>  firstname || "." || lastname || "@chinookcorp.com"
   ...> );
sqlite> SELECT
   ...>  employeeid,
   ...>  firstname,
   ...>  lastname,
   ...>  email
   ...> FROM
   ...>  employees;
1|Andrew|Adams|andrew.adams@chinookcorp.com
2|Nancy|Edwards|nancy.edwards@chinookcorp.com
3|Jane|Smith|jane.smith@chinookcorp.com
4|Margaret|Park|margaret.park@chinookcorp.com
5|Steve|Johnson|steve.johnson@chinookcorp.com
6|Michael|Mitchell|michael.mitchell@chinookcorp.com
7|Robert|King|robert.king@chinookcorp.com
8|Laura|Callahan|laura.callahan@chinookcorp.com
sqlite>
```

Query5:

```
sqlite>
sqlite> SELECT customerid FROM customers;
1
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
42
52
55
58
59
sqlite> SELECT firstname FROM customers;
Kara
Eduardo
Alexandre
Roberto
Fernanda
Mark
Jennifer
Frank
Jack
Michelle
Tim
Dan
Kathy
Heather
John
Frank
Victor
Richard
Patrick
Julia
Robert
Edward
Martha
Aaron
Wyatt
Emma
Mark
Manoj
Puja
sqlite>
```

Query6:

```
Puja|Srivastava|59
sqlite> sqlite> SELECT FirstName, Lastname, CustomerId from customers orderby
   ...> ;
Kara|Nielsen|1
Eduardo|Martins|10
Alexandre|Rocha|11
Roberto|Almeida|12
Fernanda|Ramos|13
Mark|Philips|14
Jennifer|Peterson|15
Frank|Harris|16
Jack|Smith|17
Michelle|Brooks|18
Tim|Goyer|19
Dan|Miller|20
Kathy|Chase|21
Heather|Leacock|22
John|Gordon|23
Frank|Ralston|24
Victor|Stevens|25
Richard|Cunningham|26
Patrick|Gray|27
Julia|Barnett|28
Robert|Brown|29
Edward|Francis|30
Martha|Silk|31
Aaron|Mitchell|32
Wyatt|Girard|42
Emma|Jones|52
Mark|Taylor|55
Manoj|Pareek|58
Puja|Srivastava|59
sqlite> SELECT FirstName, Lastname, CustomerId from customers ORDER BY CustomerId DESC;
Puja|Srivastava|59
Manoj|Pareek|58
Mark|Taylor|55
Emma|Jones|52
Wyatt|Girard|42
Aaron|Mitchell|32
Martha|Silk|31
Edward|Francis|30
Robert|Brown|29
Julia|Barnett|28
Patrick|Gray|27
Richard|Cunningham|26
Victor|Stevens|25
Frank|Ralston|24
John|Gordon|23
Heather|Leacock|22
Kathy|Chase|21
Dan|Miller|20
Tim|Goyer|19
Michelle|Brooks|18
Jack|Smith|17
Frank|Harris|16
Jennifer|Peterson|15
Mark|Philips|14
Fernanda|Ramos|13
Roberto|Almeida|12
Alexandre|Rocha|11
Eduardo|Martins|10
Kara|Nielsen|1
```

Query7:

```
Last login: Tue Feb 21 23:05:36 on ttys000
TUSHARs-MBP:~ tushraj$ cd Documents
TUSHARs-MBP:Documents tushraj$ pws
-bash: pws: command not found
TUSHARs-MBP:Documents tushraj$ pwd
/Users/tushraj/Documents
TUSHARs-MBP:Documents tushraj$ ls
BOOKING SFO-DEL, DEL-SFO.pdf            labs
POM                                     psu.py
Purchase Order Management System.sqlite socket-mon.py
index.html                              tp.py
index.rtf
TUSHARs-MBP:Documents tushraj$ cd POM
TUSHARs-MBP:POM tushraj$ ls
POM.db          purchase.db
TUSHARs-MBP:POM tushraj$ sqlite3 POM.db
SQLite version 3.14.0 2016-07-26 15:17:14
Enter ".help" for usage hints.
sqlite>
sqlite>
sqlite>
sqlite>
sqlite> CREATE TABLE "employees" ("EmployeeId" INTEGER PRIMARY KEY  NOT NULL ,"LastName" NVARCHAR(20) NOT NULL ,"FirstName" NVARCHAR(20) NOT NULL ,"Title" NVARCHAR(30),"BirthDate" DATETIME,"HireDate" DAT
TIME,"City" NVARCHAR(40),"State" NVARCHAR(40),"Country" NVARCHAR(40),"PostalCode" NVARCHAR(10),"Phone" NVARCHAR(24),"Email" NVARCHAR(60));
sqlite> CREATE TABLE "customers" ("CustomerId" INTEGER PRIMARY KEY  NOT NULL ,"FirstName" NVARCHAR(40) NOT NULL ,"LastName" NVARCHAR(20) NOT NULL ,"Company" NVARCHAR(80),"City" NVARCHAR(40),"State" NVARC
AR(40),"Country" NVARCHAR(40),"PostalCode" NVARCHAR(10),"Phone" NVARCHAR(24),"Email" NVARCHAR(60) NOT NULL )
   ...> ;
sqlite> CREATE TABLE "invoice_items"
   ...> (
   ...>     [InvoiceLineId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
   ...>     [InvoiceId] INTEGER  NOT NULL,
   ...>     [TrackId] INTEGER  NOT NULL,
   ...>     [UnitPrice] NUMERIC(10,2)  NOT NULL,
   ...>     [Quantity] INTEGER  NOT NULL,
   ...>     FOREIGN KEY ([InvoiceId]) REFERENCES "invoices" ([InvoiceId])
   ...>
.DS_Store   POM.db      purchase.db
   ...> ON DELETE NO ACTION ON UPDATE NO ACTION,
   ...>     FOREIGN KEY ([TrackId]) REFERENCES "tracks" ([TrackId])
   ...>
.DS_Store   POM.db      purchase.db
   ...> ON DELETE NO ACTION ON UPDATE NO ACTION
   ...> );
sqlite> CREATE TABLE "invoices"
   ...> (
   ...>     [InvoiceId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
   ...>     [CustomerId] INTEGER  NOT NULL,
   ...>     [InvoiceDate] DATETIME  NOT NULL,
   ...>     [BillingAddress] NVARCHAR(70),
   ...>     [BillingCity] NVARCHAR(40),
   ...>     [BillingState] NVARCHAR(40),
   ...>     [BillingCountry] NVARCHAR(40),
   ...>     [BillingPostalCode] NVARCHAR(10),
   ...>     [Total] NUMERIC(10,2)  NOT NULL,
   ...>     FOREIGN KEY ([CustomerId]) REFERENCES "customers" ([CustomerId])
   ...>
.DS_Store   POM.db      purchase.db
   ...> ON DELETE NO ACTION ON UPDATE NO ACTION
   ...> )
   ...> ;
sqlite>
```

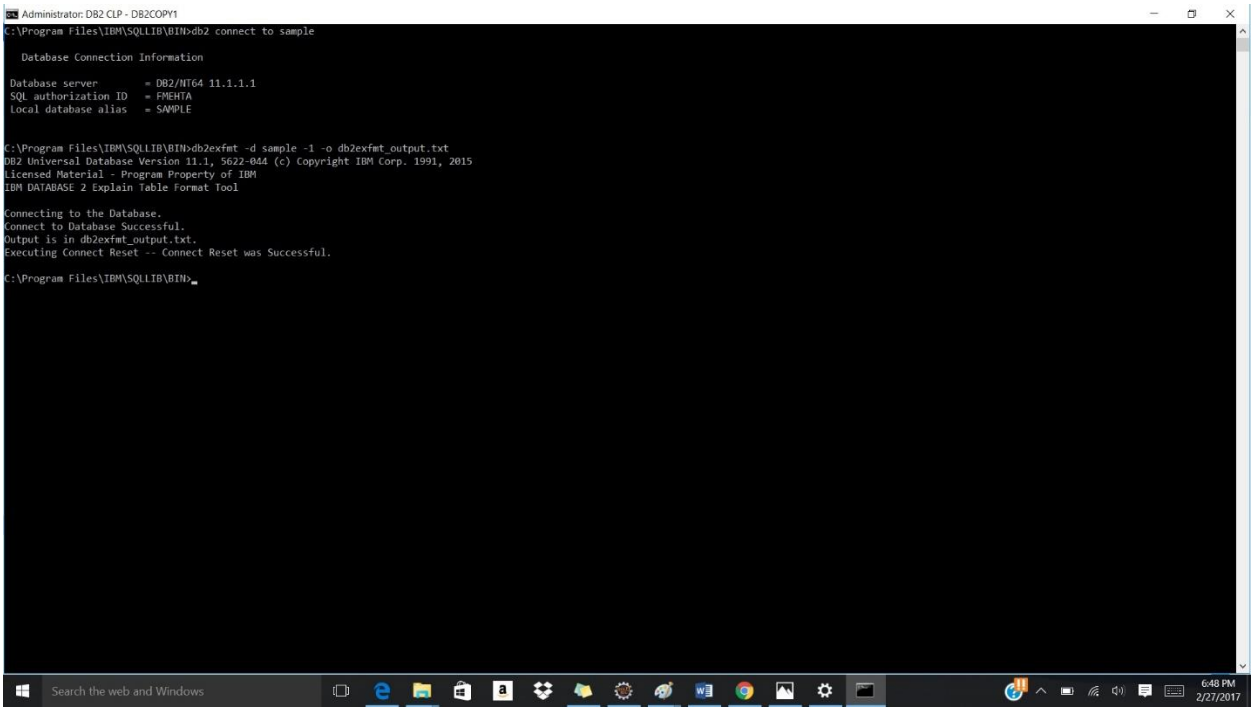
DB2 Express C

Steps Followed:

1. Installation of db2 express C
2. Creation of sample database using db2sampl command.
3. The following commands were executed to establish a connection with database successfully: set schema SAMPLE
   set db2instance=server1
   db2 connect to SAMPLE
4. To view the list of tables in the schema we used this command - db2 list tables.
5. The sample query which we executed is as follows:

   select WORKDEPT,JOB,count(*) as "total_count" ,sum(SALARY+BONUS+COMM) as "total_sum" from EMPLOYEE e where SEX = 'F' group by WORKDEPT,JOB having sum(SALARY+BONUS+COMM) > 100000.00

6. Generate query explain plan. In the query db2exfmt '-1' is equivalent to "-e % -n % -s % -v % -w -1 -# 0" '-o' represents that output is redirected to the file mentioned.



We created explain plan for the query:

db2 explain plan for "select WORKDEPT,JOB,count(*) as "total_count" ,sum(SALARY+BONUS+COMM) as "total_sum" from EMPLOYEE e where SEX = 'F' group by WORKDEPT,JOB having sum(SALARY+BONUS+COMM) > 100000.00"

DB2 Universal Database Version 11.1, 5622-044 (c) Copyright IBM Corp. 1991, 2015
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

********** EXPLAIN INSTANCE **********

DB2_VERSION: 11.01.1
FORMATTED ON DB: SAMPLE
SOURCE_NAME: SQLC2O26
SOURCE_SCHEMA: NULLID
SOURCE_VERSION:
EXPLAIN_TIME: 2017-02-19-00.38.15.335000
EXPLAIN_REQUESTER: FMEHTA

Database Context:

Parallelism: None
CPU Speed: 3.503220e-007
Comm Speed: 0
Buffer Pool size: 250
Sort Heap size: 256
Database Heap size: 600
Lock List size: 4096
Maximum Lock List: 22
Average Applications: 1
Locks Available: 28835

Package Context:

SQL Type: Dynamic
Optimization Level: 5
Blocking: Block All Cursors
Isolation Level: Cursor Stability

---------- STATEMENT 1 SECTION 201 ----------
QUERYNO: 2
QUERYTAG: CLP
Statement Type: Select
Updatable: No
Deletable: No
Query Degree: 1

Original Statement:

```
select
 WORKDEPT,
 JOB,
 count(*) as "total_count",
 sum(SALARY+BONUS+COMM) as "total_sum"
from
 EMPLOYEE
where
 SEX = 'F'
```

```
group by
  WORKDEPT,
  JOB
having
  sum(SALARY+BONUS+COMM) > 100000.00


Optimized Statement:
--------------------
SELECT
  Q3.WORKDEPT AS "WORKDEPT",
  Q3.JOB AS "JOB",
  Q3.$C3 AS "total_count",
  Q3.$C0 AS "total_sum"
FROM
  (SELECT
     SUM(((Q2.SALARY + Q2.BONUS) + Q2.COMM)),
     Q2.WORKDEPT,
     Q2.JOB,
     COUNT(*)
   FROM
     (SELECT
        Q1.WORKDEPT,
        Q1.JOB,
        Q1.SALARY,
        Q1.BONUS,
        Q1.COMM
      FROM
        FMEHTA.EMPLOYEE AS Q1
      WHERE
        (Q1.SEX = 'F')
     ) AS Q2
   GROUP BY
     Q2.JOB,
     Q2.WORKDEPT
  ) AS Q3
WHERE
  (+100000.00 < Q3.$C0)

Access Plan:
-----------
          Total Cost:             6.87588
          Query Degree:           1


      Rows
     RETURN
     (   1)
      Cost
       I/O
       |
```

6.33333
FILTER
(2)
6.87588
1
|
19
GRPBY
(3)
6.8698
1
|
19
TBSCAN
(4)
6.86733
1
|
19
SORT
(5)
6.86513
1
|
19
TBSCAN
(6)
6.85521
1
|
42
TABLE: FMEHTA
EMPLOYEE
Q1

Extended Diagnostic Information:

Diagnostic Identifier: 1
Diagnostic Details: EXP0073W The following MQT or statistical view was not eligible because one or more data filtering predicates from the query could not be matched with the MQT: "FMEHTA "."ADEFUSR".
Diagnostic Identifier: 2
Diagnostic Details: EXP0148W The following MQT or statistical view was considered in query matching: "FMEHTA "."ADEFUSR".

Plan Details:
---

1) RETURN: (Return Result)

Cumulative Total Cost: 6.87588
Cumulative CPU Cost: 202327
Cumulative I/O Cost: 1
Cumulative Re-Total Cost: 0.0432402
Cumulative Re-CPU Cost: 123430
Cumulative Re-I/O Cost: 0
Cumulative First Row Cost: 6.86662
Estimated Bufferpool Buffers: 0

Arguments:

BLDLEVEL: (Build level)

DB2 v11.1.1010.160 : s1612051900

HEAPUSE : (Maximum Statement Heap Usage)

112 Pages

PLANID : (Access plan identifier)

5a9821bc152b1f05

PREPTIME: (Statement prepare time)

184 milliseconds

SEMEVID : (Semantic environment identifier)

e0bf29ee9704d9c8

STMTHEAP: (Statement heap size)

8192

STMTID : (Normalized statement identifier)

c7ce0578972f2d70

Input Streams:

---

6) From Operator #2

Estimated number of rows: 6.33333
Number of columns: 4
Subquery predicate ID: Not Applicable

Column Names:

+Q4."total_sum"+Q4."total_count"+Q4.JOB
+Q4.WORKDEPT

2) FILTER: (Filter)

Cumulative Total Cost: 6.87588
Cumulative CPU Cost: 202327
Cumulative I/O Cost: 1
Cumulative Re-Total Cost: 0.0432402
Cumulative Re-CPU Cost: 123430
Cumulative Re-I/O Cost: 0
Cumulative First Row Cost: 6.86662

Estimated Bufferpool Buffers: 0

Predicates:
---
2) Residual Predicate,
Comparison Operator: Less Than (<)
Subquery Input Required: No
Filter Factor: 0.333333

Predicate Text:

(+100000.00 < Q3.$C0)

Input Streams:
---
5) From Operator #3

Estimated number of rows: 19
Number of columns: 4
Subquery predicate ID: Not Applicable

Column Names:

+Q3.$C3+Q3.JOB+Q3.WORKDEPT+Q3.$C0

Output Streams:
---
6) To Operator #1

Estimated number of rows: 6.33333
Number of columns: 4
Subquery predicate ID: Not Applicable

Column Names:

+Q4."total_sum"+Q4."total_count"+Q4.JOB
+Q4.WORKDEPT

3) GRPBY : (Group By)
Cumulative Total Cost: 6.8698
Cumulative CPU Cost: 184972
Cumulative I/O Cost: 1
Cumulative Re-Total Cost: 0.0371604
Cumulative Re-CPU Cost: 106075
Cumulative Re-I/O Cost: 0
Cumulative First Row Cost: 6.86579
Estimated Bufferpool Buffers: 0

Arguments:

AGGMODE : (Aggregation Mode)
COMPLETE
GROUPBYC: (Group By columns)
TRUE
GROUPBYN: (Number of Group By columns)
2
GROUPBYR: (Group By requirement)
1: Q2.WORKDEPT
2: Q2.JOB
ONEFETCH: (One Fetch flag)
FALSE

Input Streams:
---
4) From Operator #4

Estimated number of rows: 19
Number of columns: 5
Subquery predicate ID: Not Applicable

Column Names:

+Q2.WORKDEPT(A)+Q2.JOB(A)+Q2.COMM+Q2.BONUS
+Q2.SALARY

Output Streams:
---
5) To Operator #2

Estimated number of rows: 19
Number of columns: 4
Subquery predicate ID: Not Applicable

Column Names:

+Q3.$C3+Q3.JOB+Q3.WORKDEPT+Q3.$C0

4) TBSCAN: (Table Scan)
Cumulative Total Cost: 6.86733
Cumulative CPU Cost: 177920
Cumulative I/O Cost: 1
Cumulative Re-Total Cost: 0.0346899
Cumulative Re-CPU Cost: 99023
Cumulative Re-I/O Cost: 0
Cumulative First Row Cost: 6.86567
Estimated Bufferpool Buffers: 0

Arguments:

MAXPAGES: (Maximum pages for prefetch)
ALL
PREFETCH: (Type of Prefetch)
NONE
SCANDIR : (Scan Direction)
FORWARD
SPEED : (Assumed speed of scan, in sharing structures)
SLOW
THROTTLE: (Scan may be throttled, for scan sharing)
FALSE
VISIBLE : (May be included in scan sharing structures)
FALSE
WRAPPING: (Scan may start anywhere and wrap)
FALSE

Input Streams:
---
3) From Operator #5

Estimated number of rows: 19
Number of columns: 5
Subquery predicate ID: Not Applicable

Column Names:

+Q2.WORKDEPT(A)+Q2.JOB(A)+Q2.COMM+Q2.BONUS
+Q2.SALARY

Output Streams:
---
4) To Operator #3

Estimated number of rows: 19
Number of columns: 5
Subquery predicate ID: Not Applicable

Column Names:

+Q2.WORKDEPT(A)+Q2.JOB(A)+Q2.COMM+Q2.BONUS
+Q2.SALARY

5) SORT : (Sort)
Cumulative Total Cost: 6.86513
Cumulative CPU Cost: 171633
Cumulative I/O Cost: 1
Cumulative Re-Total Cost: 0.0324875

Cumulative Re-CPU Cost: 92736
Cumulative Re-I/O Cost: 0
Cumulative First Row Cost: 6.86513
Estimated Bufferpool Buffers: 1

Arguments:

DUPLWARN: (Duplicates Warning flag)
FALSE
KEYS : (Key cardinality)
19
NUMROWS : (Estimated number of rows)
19
ROWWIDTH: (Estimated width of rows)
36.000000
SORTKEY : (Sort Key column)
1: Q2.WORKDEPT(A)
2: Q2.JOB(A)
TEMPSIZE: (Temporary Table Page Size)
8192
UNIQUE : (Uniqueness required flag)
FALSE

Input Streams:
---
2) From Operator #6

Estimated number of rows: 19
Number of columns: 5
Subquery predicate ID: Not Applicable

Column Names:

+Q2.COMM+Q2.BONUS+Q2.SALARY+Q2.JOB+Q2.WORKDEPT

Output Streams:
---
3) To Operator #4

Estimated number of rows: 19
Number of columns: 5
Subquery predicate ID: Not Applicable

Column Names:

+Q2.WORKDEPT(A)+Q2.JOB(A)+Q2.COMM+Q2.BONUS
+Q2.SALARY

6) TBSCAN: (Table Scan)

Cumulative Total Cost: 6.85521
Cumulative CPU Cost: 143329
Cumulative I/O Cost: 1
Cumulative Re-Total Cost: 0.0324875
Cumulative Re-CPU Cost: 92736
Cumulative Re-I/O Cost: 0
Cumulative First Row Cost: 6.8244
Estimated Bufferpool Buffers: 1

Arguments:

CUR_COMM: (Currently Committed)
TRUE
LCKAVOID: (Lock Avoidance)
TRUE
MAXPAGES: (Maximum pages for prefetch)
ALL
PREFETCH: (Type of Prefetch)
NONE
ROWLOCK : (Row Lock intent)
SHARE (CS/RS)
SCANDIR : (Scan Direction)
FORWARD
SKIP_INS: (Skip Inserted Rows)
TRUE
SPEED : (Assumed speed of scan, in sharing structures)
FAST
TABLOCK : (Table Lock intent)
INTENT SHARE
TBISOLVL: (Table access Isolation Level)
CURSOR STABILITY
THROTTLE: (Scan may be throttled, for scan sharing)
TRUE
VISIBLE : (May be included in scan sharing structures)
TRUE
WRAPPING: (Scan may start anywhere and wrap)
TRUE

Predicates:
---
5) Sargable Predicate,
Comparison Operator: Equal (=)
Subquery Input Required: No
Filter Factor: 0.452381

Predicate Text:

(Q1.SEX = 'F')

Input Streams:
---

1) From Object FMEHTA.EMPLOYEE

Estimated number of rows: 42
Number of columns: 7
Subquery predicate ID: Not Applicable

Column Names:

+Q1.RID+Q1.COMM+Q1.BONUS+Q1.SALARY+Q1.JOB
+Q1.WORKDEPT+Q1.SEX

Output Streams:
---

2) To Operator #5

Estimated number of rows: 19
Number of columns: 5
Subquery predicate ID: Not Applicable

Column Names:

+Q2.COMM+Q2.BONUS+Q2.SALARY+Q2.JOB+Q2.WORKDEPT

Objects Used in Access Plan:

Schema: FMEHTA
Name: ADEFUSR
Type: Materialized View (reference only)

Schema: FMEHTA
Name: EMPLOYEE
Type: Table

Time of creation: 2017-02-18-21.37.59.024001
Last statistics update: 2017-02-18-21.58.57.934000
Number of columns: 14
Number of rows: 42
Width of rows: 63
Number of buffer pool pages: 1
Number of data partitions: 1
Distinct row values: No
Tablespace name: USERSPACE1
Tablespace overhead: 6.725000
Tablespace transfer rate: 0.080000
Source for statistics: Single Node
Prefetch page count: 32
Container extent page count: 32

Table overflow record count:	0
Table Active Blocks:	-1
Average Row Compression Ratio:	0
Percentage Rows Compressed:	0
Average Compressed Row Size:	0

Graph Datastore

Steps followed:

1. We created an account on IBM Bluemix.
2. We created a graph service instance. In order to query the graph database we consumed the graph service through REST API calls using bash.
3. The following commands were used to create and query our own schema:

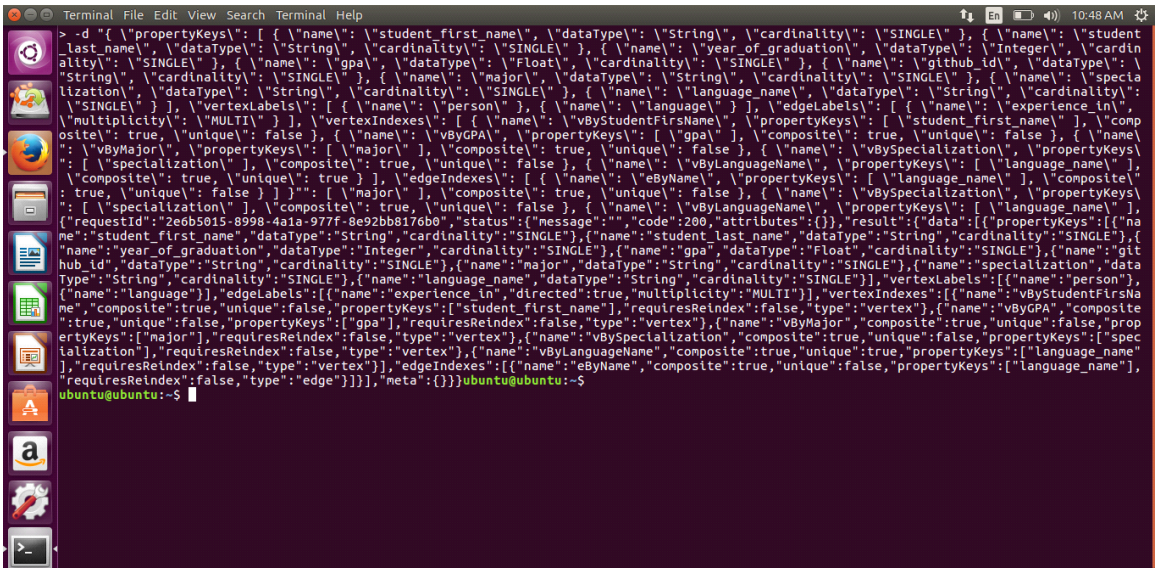
Creating schema:

curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/schema" \

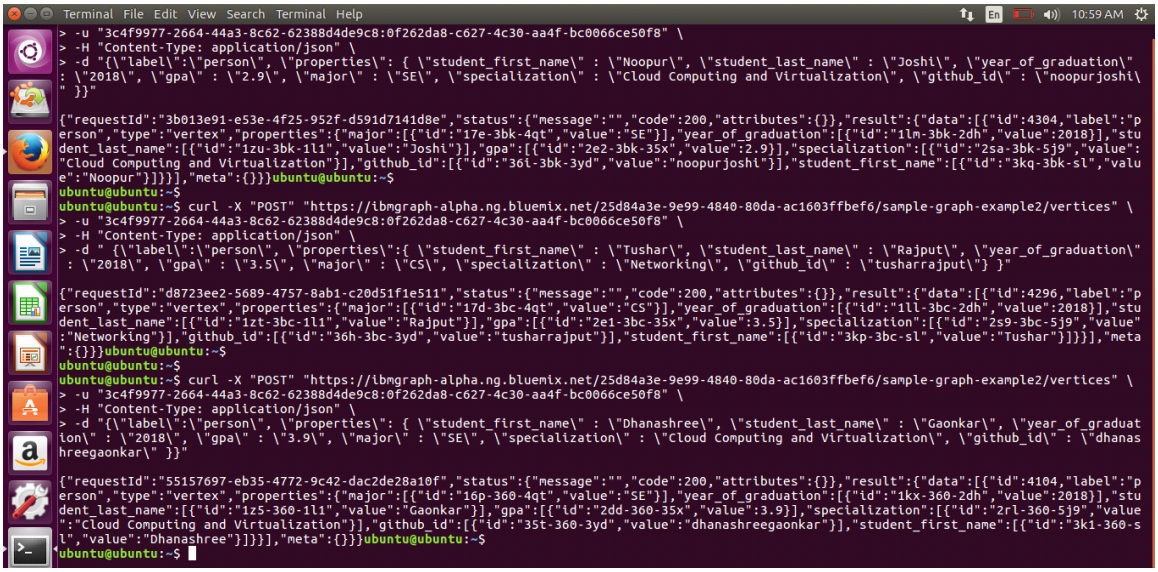
-u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

-H "Content-Type: application/json" \

-d "{ \"propertyKeys\": [{ \"name\": \"application_name\", \"dataType\": \"String\", \"cardinality\": \"SINGLE\" },{ \"name\": \"student_first_name\", \"dataType\": \"String\", \"cardinality\": \"SINGLE\" }, { \"name\": \"student_last_name\", \"dataType\": \"String\", \"cardinality\": \"SINGLE\" }, { \"name\": \"year_of_graduation\", \"dataType\": \"Integer\", \"cardinality\": \"SINGLE\" }, { \"name\": \"gpa\", \"dataType\": \"Float\", \"cardinality\": \"SINGLE\" }, { \"name\": \"github_id\", \"dataType\": \"String\", \"cardinality\": \"SINGLE\" }, { \"name\": \"major\", \"dataType\": \"String\", \"cardinality\": \"SINGLE\" }, { \"name\": \"specialization\", \"dataType\": \"String\", \"cardinality\": \"SINGLE\" }, { \"name\": \"language_name\", \"dataType\": \"String\", \"cardinality\": \"SINGLE\" }], \"vertexLabels\": [{ \"name\": \"person\" }, { \"name\": \"language\" }], \"edgeLabels\": [{ \"name\": \"experience_in\", \"multiplicity\": \"MULTI\" },{ \"name\": \"uses\", \"multiplicity\": \"MULTI\" }], \"vertexIndexes\": [{ \"name\": \"vByStudentFirstName\", \"propertyKeys\": [\"student_first_name\"], \"composite\": true, \"unique\": false }, { \"name\": \"vByGPA\", \"propertyKeys\": [\"gpa\"], \"composite\": true, \"unique\": false }, { \"name\": \"vByMajoar\", \"propertyKeys\": [\"major\"], \"composite\": true, \"unique\": false }, { \"name\": \"vBySpecialization\", \"propertyKeys\": [\"specialization\"], \"composite\": true, \"unique\": false }, { \"name\": \"vByLanguageName\", \"propertyKeys\": [\"language_name\"], \"composite\": true, \"unique\": true }], \"edgeIndexes\": [{ \"name\": \"eByName\", \"propertyKeys\": [\"language_name\"], \"composite\": true, \"unique\": false }, { \"name\": \"vByLanguageName\", \"propertyKeys\": [\"language_name\"], \"composite\": true, \"unique\": false }, { \"name\": \"vBySpecialization\", \"propertyKeys\": [\"specialization\"], \"composite\": true, \"unique\": false }, { \"name\": \"vByAppName\", \"propertyKeys\": [\"application_name\"], \"composite\": true, \"unique\": true }], \"edgeIndexes\": [{ \"name\": \"eByName\", \"propertyKeys\": [\"application_name\"], \"composite\": true, \"unique\": false },{ \"name\": \"eByAppName\", \"propertyKeys\": [\"application_name\"], \"composite\": true, \"unique\": false }] }"



Creating vertices:



curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/vertices" \

-u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

-H "Content-Type: application/json" \

-d "{\"label\":\"person\", \"properties\": { \"student_first_name\" : \"Noopur\", \"student_last_name\" : \"Joshi\", \"year_of_graduation\" : \"2018\", \"gpa\" : \"2.9\", \"major\" : \"SE\", \"specialization\" : \"Cloud Computing and Virtualization\", \"github_id\" : \"noopurjoshi\" }}"

curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/vertices" \

-u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

-H "Content-Type: application/json" \

-d " {\"label\":\"person\", \"properties\":{ \"student_first_name\" : \"Tushar\", \"student_last_name\" : \"Rajput\", \"year_of_graduation\" : \"2018\", \"gpa\" : \"3.5\", \"major\" : \"CS\", \"specialization\" : \"Networking\", \"github_id\" : \"tusharrajput\"} }"

curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/vertices" \

-u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

-H "Content-Type: application/json" \

-d "{\"label\":\"person\", \"properties\": { \"student_first_name\" : \"Dhanashree\", \"student_last_name\" : \"Gaonkar\", \"year_of_graduation\" : \"2018\", \"gpa\" : \"3.9\", \"major\" : \"SE\", \"specialization\" : \"Cloud Computing and Virtualization\", \"github_id\" : \"dhanashreegaonkar\" }}"

```
curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/vertices" \

        -u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

        -H "Content-Type: application/json" \

        -d "{\"label\":\"person\", \"properties\": { \"student_first_name\" : \"Foram\", \"student_last_name\" : \"Mehta\", \"year_of_graduation\" : \"2018\", \"gpa\" : \"3.2\", \"major\" : \"SE\", \"specialization\" : \"Software Systems Engineering\", \"github_id\" : \"forammehta\" }}"


curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/vertices" \

        -u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

        -H "Content-Type: application/json" \

        -d      "{\"label\":\"language\", \"properties\": { \"language_name\" : \"Java\"}}"



curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/vertices" \

        -u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

        -H "Content-Type: application/json" \

        -d      "{\"label\":\"language\", \"properties\": { \"language_name\" : \"Python\"}}"


curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/vertices" \

        -u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

        -H "Content-Type: application/json" \

        -d      "{\"label\":\"language\", \"properties\": { \"language_name\" : \"Javascript\"}}"


curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/vertices" \

        -u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

        -H "Content-Type: application/json" \

        -d      "{\"label\":\"language\", \"properties\": { \"language_name\" : \"Perl\"}}"
```

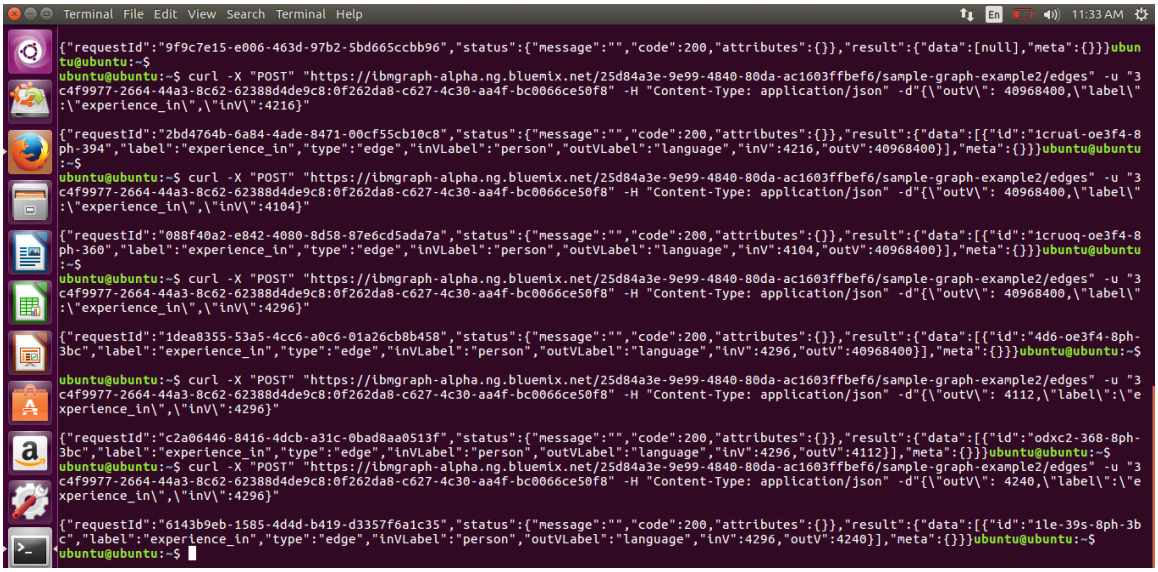
curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/vertices" \

-u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

-H "Content-Type: application/json" \

-d "{\"label\":\"application\", \"properties\": { \"application_name\" : \"App1\"}}"

Creating Edges:



curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/ sample-graph-example2/edges" \

-u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

-H "Content-Type: application/json" \

-d "{\"outV\":\" 8288\",\"label\":\"experience_in\",\"inV\":\"4302\"}"

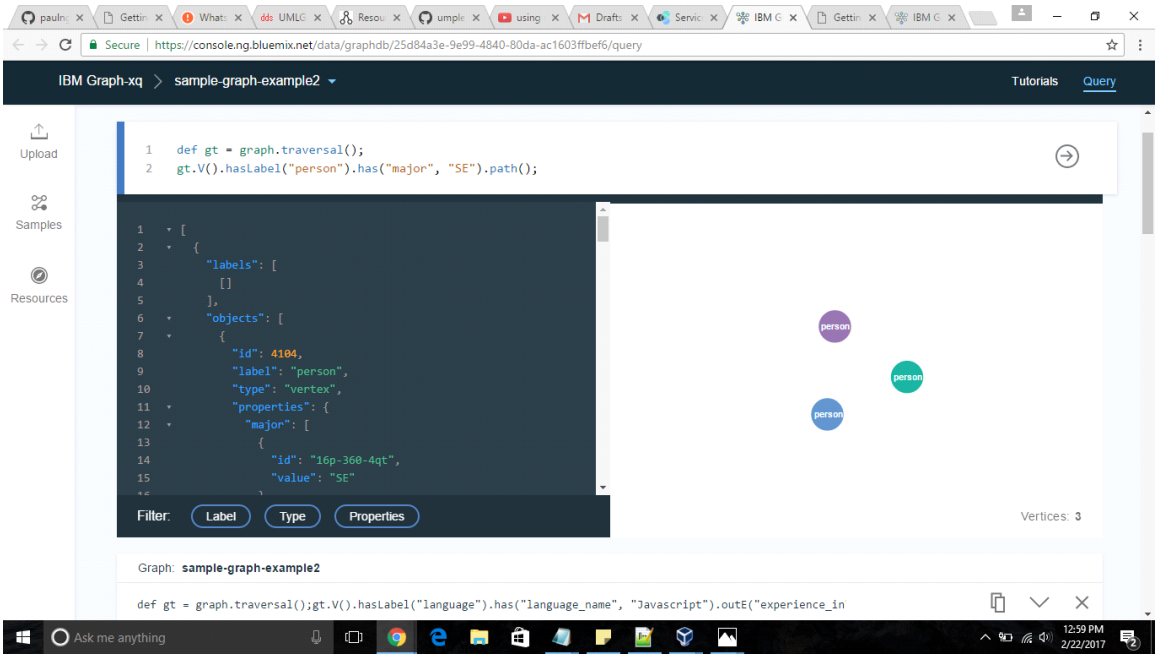
curl -X "POST" "https://ibmgraph-alpha.ng.bluemix.net/25d84a3e-9e99-4840-80da-ac1603ffbef6/sample-graph-example2/edges" \

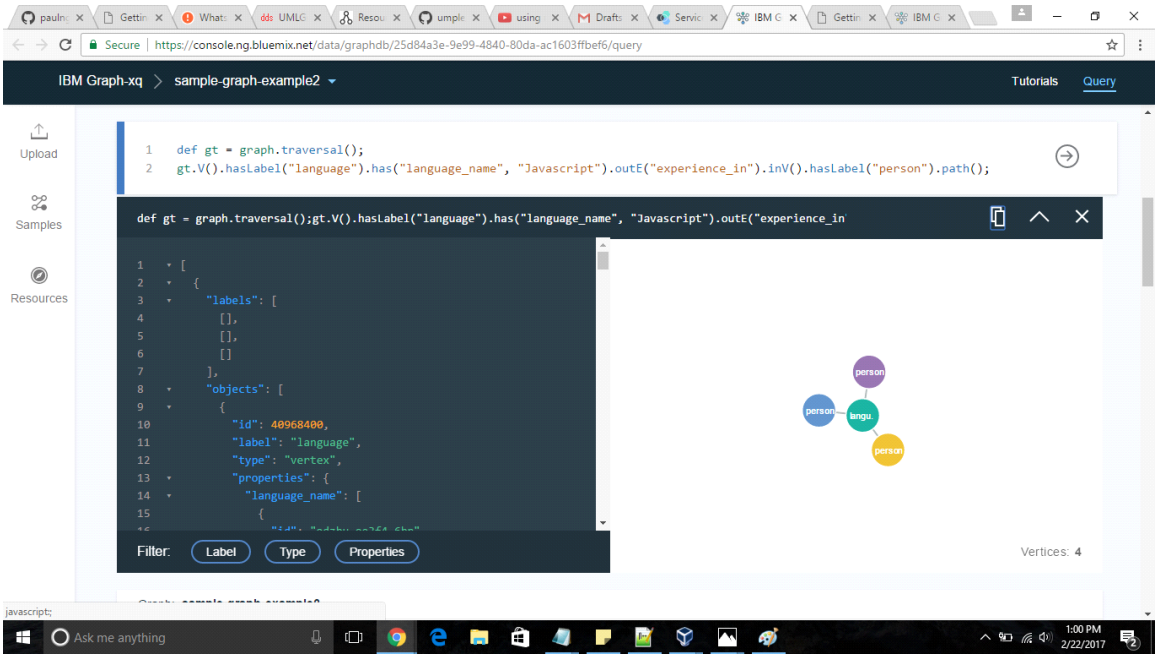
-u "3c4f9977-2664-44a3-8c62-62388d4de9c8:0f262da8-c627-4c30-aa4f-bc0066ce50f8" \

-H "Content-Type: application/json" \

-d "{\"outV\":\" 4208\",\"label\":\"uses\",\"inV\":\" 4192\"}"

4. Output for Gremlin command execution.





Summary:

	SQLite	DB2 Express	Graph Datastore
Unique Features	Easy to use. Minimal setup and configuration time.	Gives optimized query. Visual representation of query statistics.	NoSQL database.
Issues Faced	-	Not easily installable in OS X. Had to pull the image of DB2 from docker and implement it.	Difficulty in debugging requests sent through bash. Could not find the implementation of aggregate function in gremlin.
Drawbacks	Browser Dependency	-	-