

SQLITE : Purchase Order Management System

DB SCHEMA:-

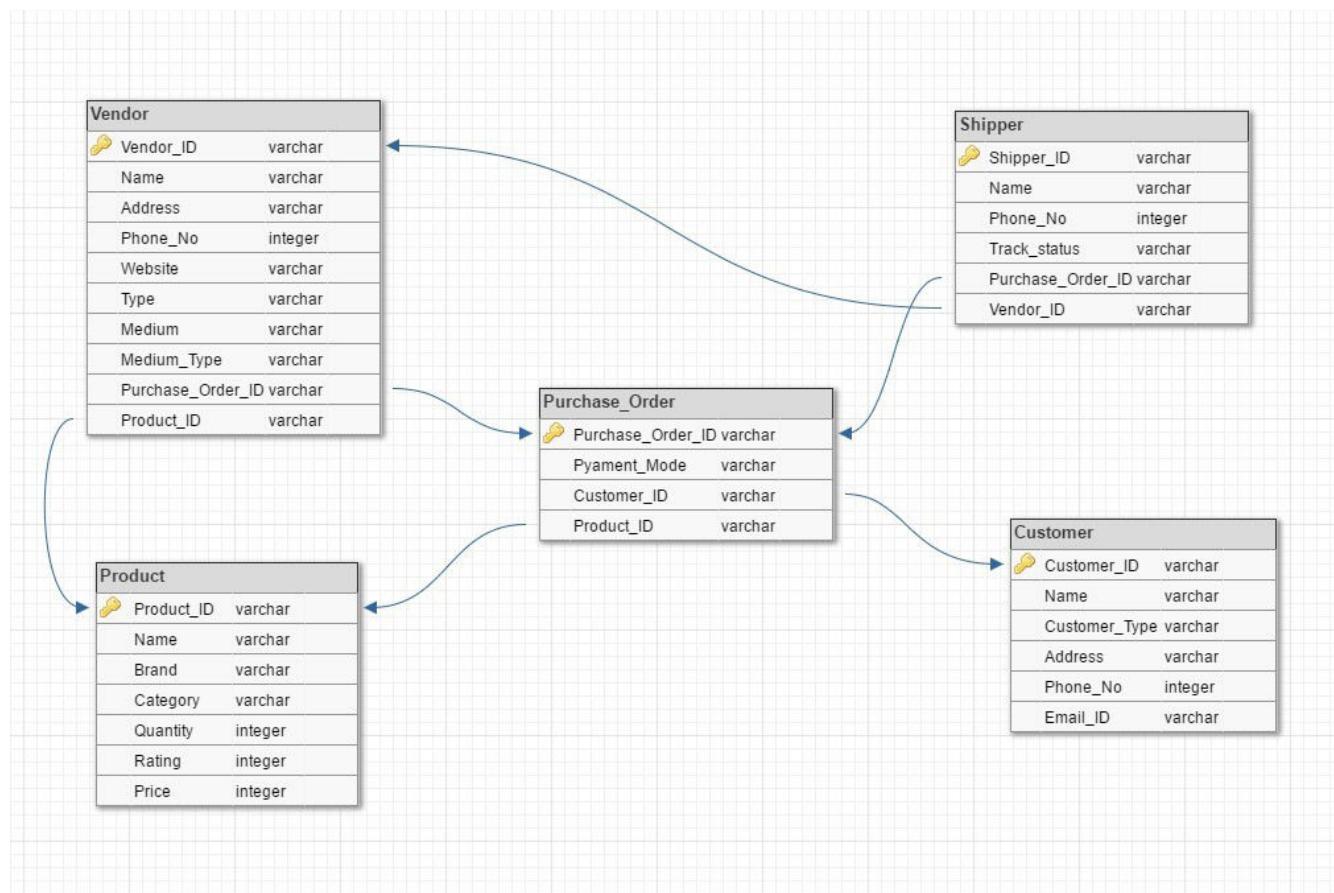
```
CREATE TABLE "customer"(
    Customer_ID varchar PRIMARY KEY NOT NULL,
    Name varchar NOT NULL,
    Customer_Type varchar NOT NULL,
    Address varchar,
    Phone_No integer,
    Email_ID varchar
);
;
CREATE TABLE __sm_ext_mgmt (`id` integer primary key, `type` text not null , `value` text);
CREATE TABLE Product(
    Product_ID varchar PRIMARY KEY NOT NULL,
    Name varchar NOT NULL,
    Brand varchar NOT NULL,
    Category varchar NOT NULL,
    Quantity varchar NOT NULL,
    Rating varchar NOT NULL,
    Price varchar NOT NULL);
;
CREATE TABLE "purchase_order"(
    Purchase_Order_ID varchar PRIMARY KEY NOT NULL,
    Payment_Mode varchar NOT NULL,
    Customer_ID varchar NOT NULL,
    Product_ID varchar NOT NULL,
    FOREIGN KEY(Customer_ID) REFERENCES Customer(Customer_ID)
    FOREIGN KEY(Product_ID) REFERENCES Product(Product_ID));
;
CREATE TABLE Vendor(
    Vendor_ID varchar PRIMARY KEY NOT NULL,
    Name varchar NOT NULL,
    Address varchar NOT NULL,
    Phone_No Integer NOT NULL,
    Website varchar NOT NULL,
    Type varchar NOT NULL,
    Medium varchar NOT NULL,
    Medium_Type varchar NOT NULL,
    Purchase_Order_ID varchar NOT NULL,
    Product_ID varchar NOT NULL,
    FOREIGN KEY(Purchase_Order_ID) REFERENCES "Purchase_Order"(Purchase_Order_ID)
    FOREIGN KEY(Product_ID) REFERENCES Product(Product_ID));
;
```

```

CREATE TABLE Shipper(
    Shipper_ID varchar PRIMARY KEY NOT NULL,
    Name varchar NOT NULL,
    Phone_No Integer NOT NULL,
    Track_Status varchar NOT NULL,
    Purchase_Order_ID varchar NOT NULL,
    Vendor_ID varchar NOT NULL,
    FOREIGN KEY(Purchase_Order_ID) REFERENCES "Purchase_Order"(Purchase_Order_ID)
    FOREIGN KEY(Vendor_ID) REFERENCES Vendor(Vendor_ID));
;

```

ER DIAGRAM:



1.) Screenshot for Table Customer :

Purchase Order Management System.sqlite - Sqliteman

Schema Pragmas

Database

main

Tables (6)

- __sm_ext_mgmt
- customer
- product
- purchase_order
- shipper
- vendor

Views (0)

System Catalogue (1)

1 select * from Customer;
2 select * from product;
3 select * from purchase_order;
4 select * from shipper;
5 select * from vendor;

Duration: 0.013 seconds

Full View Item View Script Output

Customer_ID	Name	Customer_Type	Address	Phone_No	Email_ID
1 aa	Akansha	Person	San Jose	3657219	aaa@abc.com
2 bb	Raksha	Person	San Jose	8324293	afg@pqr.com
3 kl	ABC	Company	Santa Clara	2991827	qqk@abc.com
4 sj	PQR	Company	Fremont	2197214	fkg@pqr.com

Query OK
Row(s) returned: 4
select * from Customer;

Sqlite: 3.14.1

2.) Screenshot for Table product :

Purchase Order Management System.sqlite - Sqliteman

Schema Pragmas

Database

main

Tables (6)

- __sm_ext_mgmt
- customer
- product
- purchase_order
- shipper
- vendor

Views (0)

System Catalogue (1)

1 select * from Customer;
2 Select * from product;
3 select * from purchase_order;
4 select * from shipper;
5 select * from vendor;

Duration: 0.012 seconds

Full View Item View Script Output

Product_ID	Name	Brand	Category	Quantity	Rating	Price
1 pro_123	Laptop	Apple	Electronics	1	4	500
2 pro_999	PlayStation4	Sony	Electronics	1	5	200
3 pro_666	BOOK	Wren&Martin	Books	2	3	30

Query OK
Row(s) returned: 3
select * from product;

Sqlite: 3.14.1

3.) Table purchase_order :

The screenshot shows the Sqliteman interface with the database 'Purchase Order Management System.sqlite' open. The left sidebar has icons for various tools. The main area has tabs for 'Schema' and 'Pragmas'. In the 'Database' tree, under 'main', there are 'Tables (6)' which include '_sm_ext_mgmt', 'customer', 'product', 'purchase_order', 'shipper', and 'vendor'. The 'Views (0)' and 'System Catalogue (1)' are also listed. The central pane contains a SQL editor with the following code:

```
1 select * from Customer;
2 select * from product;
3 select * from purchase_order;
4 select * from shipper;
5 select * from vendor;
```

The status bar at the bottom right shows 'Sqlite: 3.14.1'.

The results pane shows a table titled 'Full View' with the following data:

Purchase_Order_ID	Payment_Mode	Customer_ID	Product_ID
pur_111	CreditCard	1aa	pro_123
pur_222	PayPal	2bb	pro_999
pur_333	DebitCard	3kl	pro_666

The status bar at the bottom right shows 'Duration: 0.008 seconds' and '* Col: 30 Row: 3/5'.

The bottom pane shows the query results:

```
Query OK
Row(s) returned: 3
select * from purchase_order;
```

4.) Table vendor :

The screenshot shows the Sqliteman interface with the same database 'Purchase Order Management System.sqlite' open. The left sidebar has icons for various tools. The main area has tabs for 'Schema' and 'Pragmas'. In the 'Database' tree, under 'main', there are 'Tables (6)' which include '_sm_ext_mgmt', 'customer', 'product', 'purchase_order', 'shipper', and 'vendor'. The 'Views (0)' and 'System Catalogue (1)' are also listed. The central pane contains a SQL editor with the following code:

```
1 select * from Customer;
2 select * from product;
3 select * from purchase_order;
4 select * from shipper;
5 select * from vendor;
```

The status bar at the bottom right shows 'Sqlite: 3.14.1'.

The results pane shows a table titled 'Full View' with the following data:

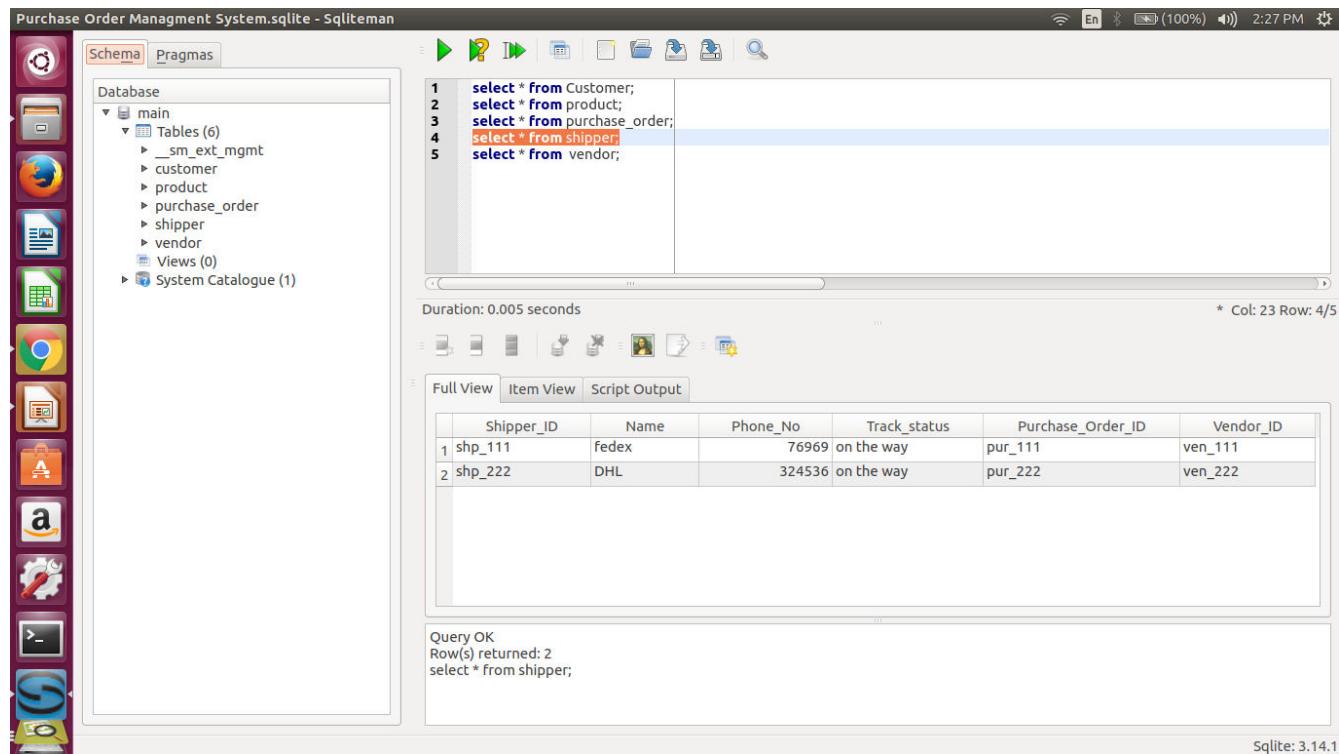
Vendor_ID	Name	Address	Phone_No	Website	Type	Medium	Medium_Type	Purchase_Order_ID	Product_ID
ven_111	Zinus	San Jose	6879879	zinus.com	shop	Amazon	e-commerce	pur_111	pro_123
ven_222	mehta	San Jose	7596	mehta.com	industry	ebay	e-commerce	pur_222	pro_999

The status bar at the bottom right shows 'Duration: 0.014 seconds' and '* Col: 23 Row: 5/5'.

The bottom pane shows the query results:

```
Query OK
Row(s) returned: 2
select * from vendor;
```

5.) Table shipper :



Purchase Order Management System.sqlite - Sqliteman

Schema Pragmas

Database

- main
 - Tables (6)
 - __sm_ext_mgmt
 - customer
 - product
 - purchase_order
 - shipper
 - vendor
 - Views (0)
 - System Catalogue (1)

```

1 select * from Customer;
2 select * from product;
3 select * from purchase_order;
4 select * from shipper; -- This line is highlighted in red
5 select * from vendor;

```

Duration: 0.005 seconds * Col: 23 Row: 4/5

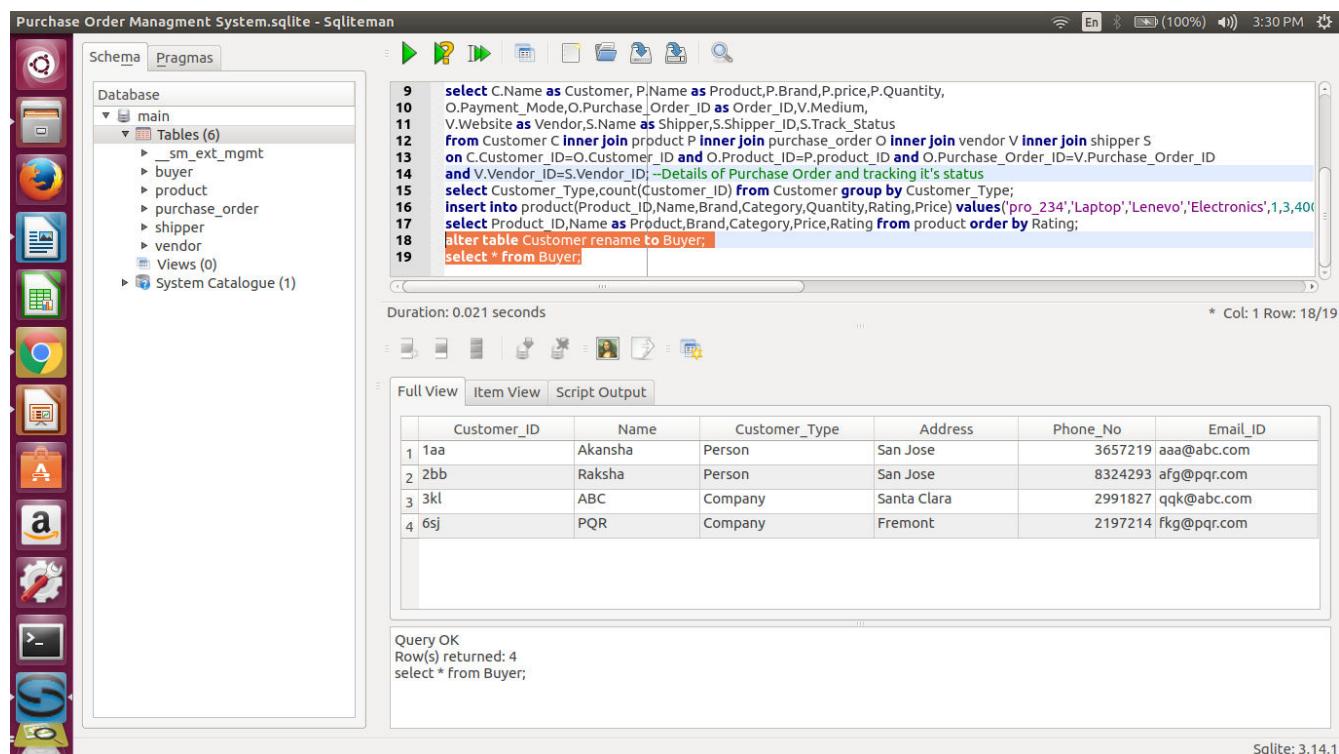
Full View Item View Script Output

Shipper_ID	Name	Phone_No	Track_Status	Purchase_Order_ID	Vendor_ID
1 shp_111	fedex	76969	on the way	pur_111	ven_111
2 shp_222	DHL	324536	on the way	pur_222	ven_222

Query OK
Row(s) returned: 2
select * from shipper;

Sqlite: 3.14.1

6.) Alter Query :



Purchase Order Management System.sqlite - Sqliteman

Schema Pragmas

Database

- main
 - Tables (6)
 - __sm_ext_mgmt
 - buyer
 - product
 - purchase_order
 - shipper
 - vendor
 - Views (0)
 - System Catalogue (1)

```

9 select C.Name as Customer, P.Name as Product,P.Brand,P.price,P.Quantity,
10 O.Payment_Mode,O.Purchase_Order_ID as Order_ID,V.Medium,
11 V.Website as Vendor,S.Name as Shipper,S.Shipper_ID,S.Track_Status
12 from Customer C inner join product P inner join purchase_order O inner join vendor V inner join shipper S
13 on C.Customer_ID=O.Customer_ID and O.Product_ID=P.product_ID and O.Purchase_Order_ID=V.Purchase_Order_ID
14 and V.Vendor_ID=S.Vendor_ID; --Details of Purchase Order and tracking it's status
15 select Customer_Type,count(Customer_ID) from Customer group by Customer_Type;
16 insert into product(Product_ID,Name,Brand,Category,Quantity,Rating,Price) values('pro_234','Laptop','Lenevo','Electronics',1,3,400);
17 select Product_ID,Name as Product,Brand,Category,Price,Rating from product order by Rating;
18 alter table Customer rename to Buyer;
19 select * from Buyer;

```

Duration: 0.021 seconds * Col: 1 Row: 18/19

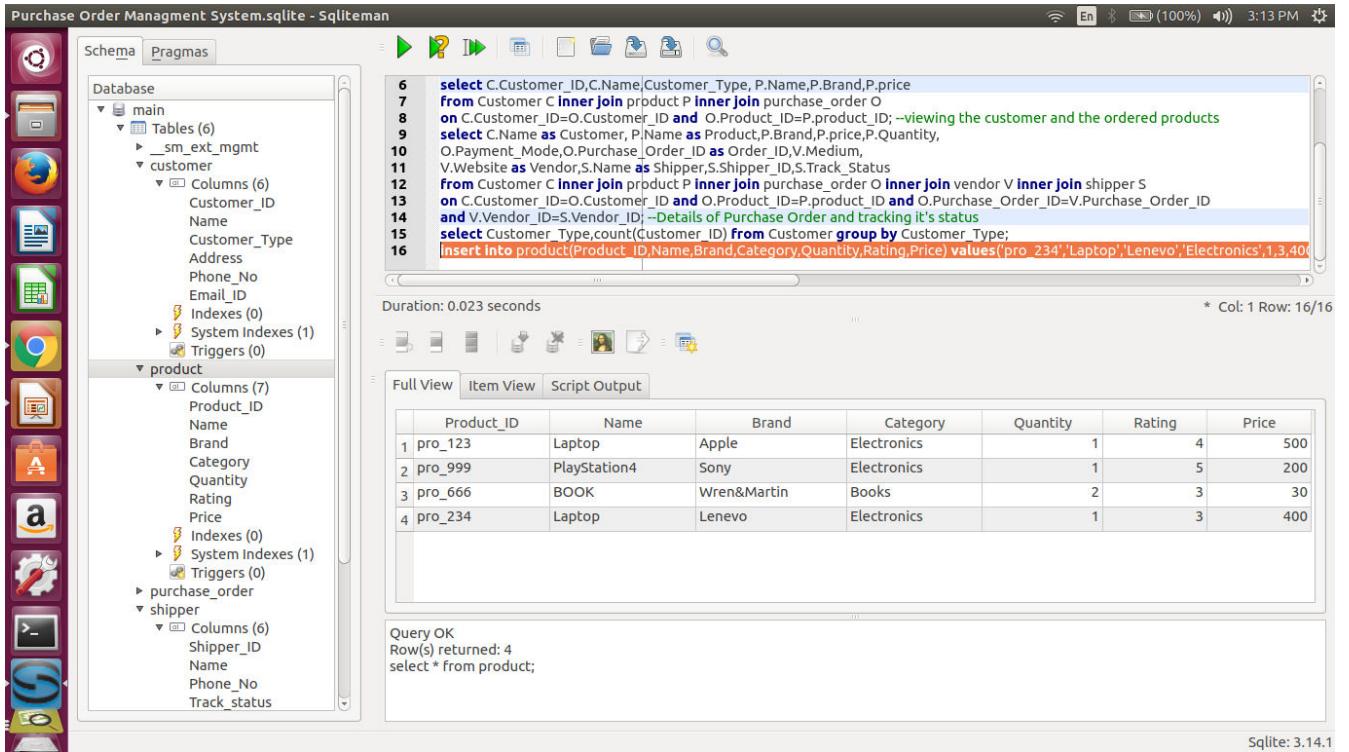
Full View Item View Script Output

Customer_ID	Name	Customer_Type	Address	Phone_No	Email_ID
1 1aa	Akansha	Person	San Jose	3657219	aaa@abc.com
2 2bb	Raksha	Person	San Jose	8324293	afg@pqr.com
3 3kl	ABC	Company	Santa Clara	2991827	qqk@abc.com
4 6sj	PQR	Company	Fremont	2197214	fkg@pqr.com

Query OK
Row(s) returned: 4
select * from Buyer;

Sqlite: 3.14.1

7.) Insert Query :



The screenshot shows the Sqliteman interface with the database 'Purchase Order Management System.sqlite' open. The left pane displays the schema with tables: main, customer, product, purchase_order, shipper, and vendor. The right pane shows the SQL command window with the following code:

```

6  select C.Customer_ID,C.Name,Customer_Type, P.Name,P.Brand,P.Price
7  from Customer C inner join product P inner join purchase_order O
8  on C.Customer_ID=O.Customer_ID and O.Product_ID=P.product_ID; --viewing the customer and the ordered products
9  select C.Name as Customer, P.Name as Product,P.Brand,P.Price,P.Quantity,
10  O.Payment_Mode,O.Purchase_Order_ID as Order_ID,V.Medium,
11  V.Website as Vendor,S.Name as Shipper,S.Shipper_ID,S.Track_Status
12  from Customer C inner join product P inner join purchase_order O inner join vendor V inner join shipper S
13  on C.Customer_ID=O.Customer_ID and O.Product_ID=P.product_ID and O.Purchase_Order_ID=V.Purchase_Order_ID
14  and V.Vendor_ID=S.Vendor_ID; --Details of Purchase Order and tracking it's status
15  select Customer_Type,count(Customer_ID) from Customer group by Customer_Type;
16  insert into product(Product_ID,Name,Brand,Category,Quantity,Rating,Price) values('pro_234','Laptop','Lenovo','Electronics',1,3,400)

```

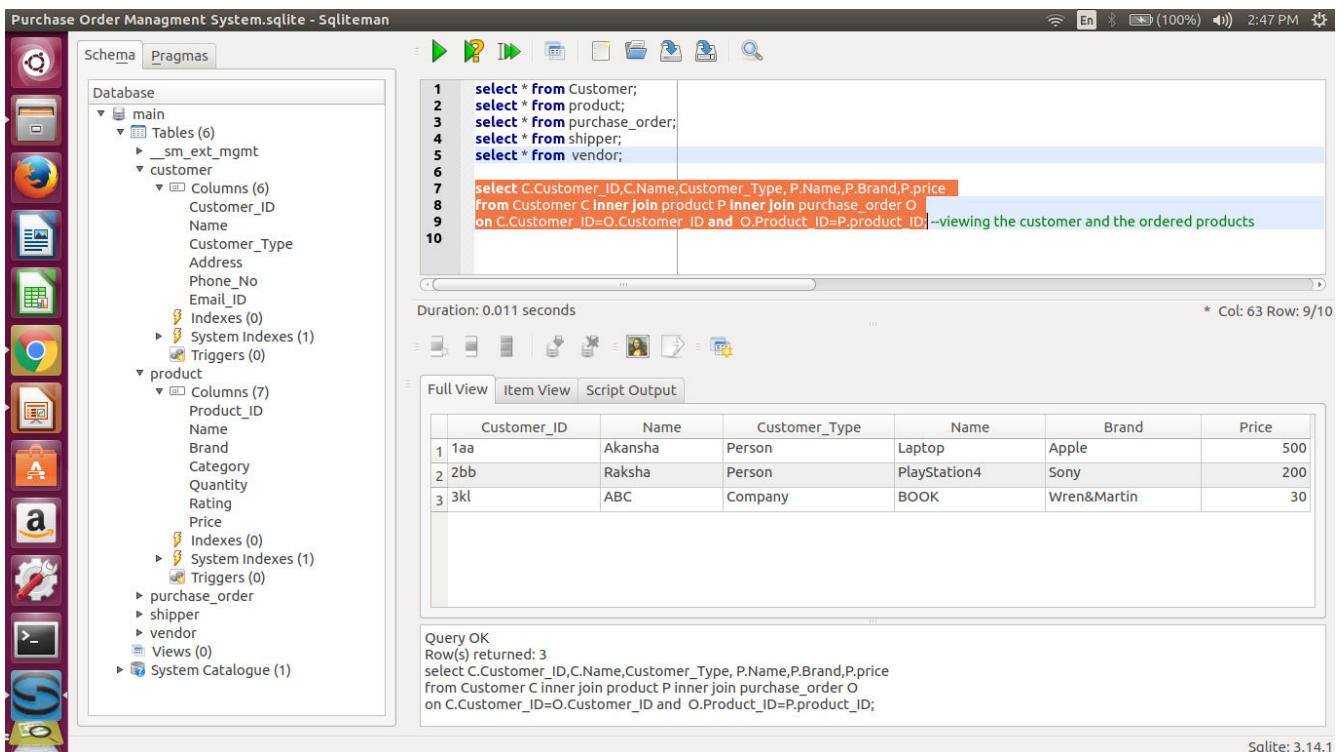
The duration of the execution is 0.023 seconds. Below the code, the results of the query are displayed in a table:

Product_ID	Name	Brand	Category	Quantity	Rating	Price
1 pro_123	Laptop	Apple	Electronics	1	4	500
2 pro_999	PlayStation4	Sony	Electronics	1	5	200
3 pro_666	BOOK	Wren&Martin	Books	2	3	30
4 pro_234	Laptop	Lenovo	Electronics	1	3	400

The message 'Query OK Row(s) returned: 4' is shown at the bottom.

8.) JOIN Query :

Viewing customer, it's type and ordered products:



The screenshot shows the Sqliteman interface with the database 'Purchase Order Management System.sqlite' open. The left pane displays the schema with tables: main, customer, product, purchase_order, shipper, and vendor. The right pane shows the SQL command window with the following code:

```

1  select * from Customer;
2  select * from product;
3  select * from purchase_order;
4  select * from shipper;
5  select * from vendor;
6
7  select C.Customer_ID,C.Name,Customer_Type, P.Name,P.Brand,P.Price
8  from Customer C inner join product P inner join purchase_order O
9  on C.Customer_ID=O.Customer_ID and O.Product_ID=P.product_ID; --viewing the customer and the ordered products
10

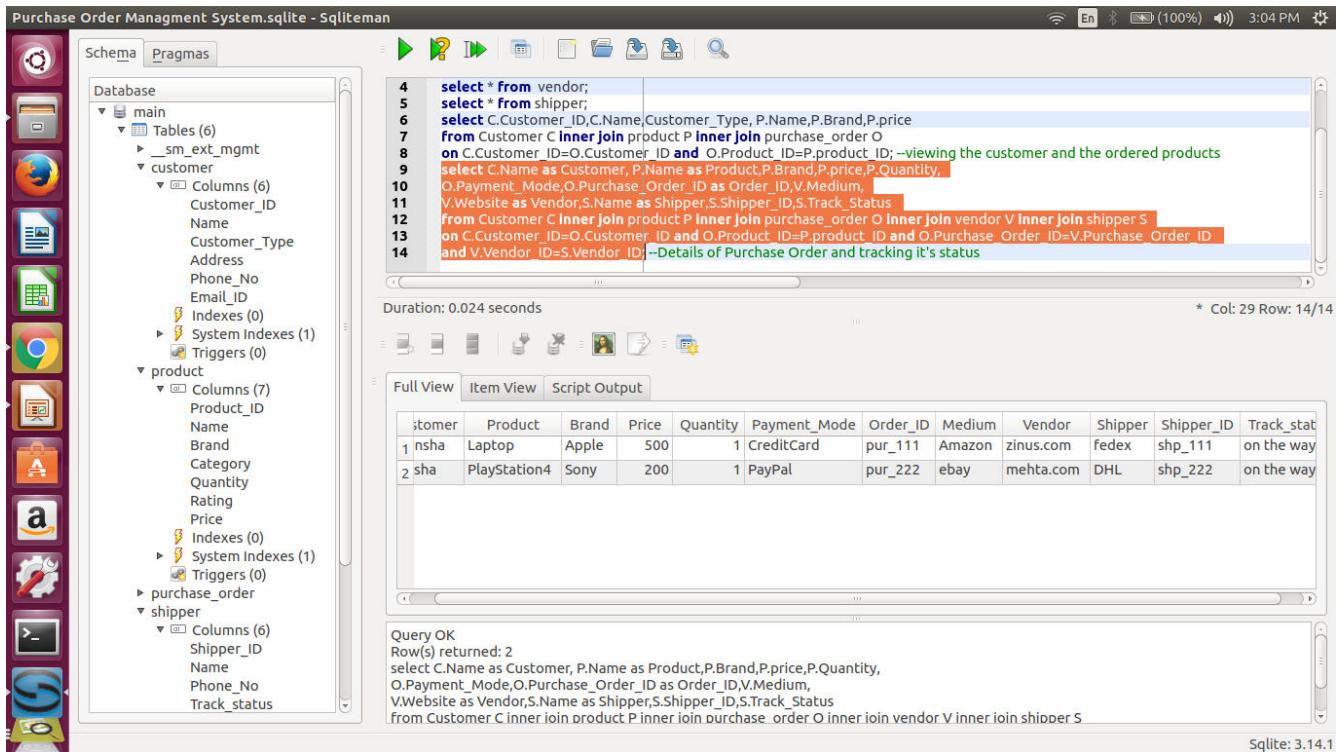
```

The duration of the execution is 0.011 seconds. Below the code, the results of the query are displayed in a table:

Customer_ID	Name	Customer_Type	Name	Brand	Price
1 1aa	Akansha	Person	Laptop	Apple	500
2 2bb	Raksha	Person	PlayStation4	Sony	200
3 3kl	ABC	Company	BOOK	Wren&Martin	30

The message 'Query OK Row(s) returned: 3' is shown at the bottom.

Viewing details of purchased order and tracking status of ordered product:



The screenshot shows the Sqliteman interface with the database 'Purchase Order Management System.sqlite' open. The left sidebar displays the schema with tables: main, customer, product, purchase_order, and shipper. The main window contains a SQL query window with the following code:

```

4 select * from vendor;
5 select * from shipper;
6 select C.Customer_ID,C.Name,Customer_Type,P.Name,P.Brand,P.Price
7 from Customer C inner join product P inner join purchase_order O
8 on C.Customer_ID=O.Customer_ID and O.Product_ID=P.product_ID; --viewing the customer and the ordered products
9 select C.Name as Customer,P.Name as Product,P.Brand,P.Price,P.Quantity,
10 O.Payment_Mode,O.Purchase_Order_ID as Order_ID,V.Medium,
11 V.Website as Vendor,S.Name as Shipper,S.Shipper_ID,S.Track_Status
12 from Customer C inner join product P inner join purchase_order O inner join vendor V inner join shipper S
13 on C.Customer_ID=O.Customer_ID and O.Product_ID=P.product_ID and O.Purchase_Order_ID=V.Purchase_Order_ID
14 and V.Vendor_ID=S.Vendor_ID; --Details of Purchase Order and tracking its status

```

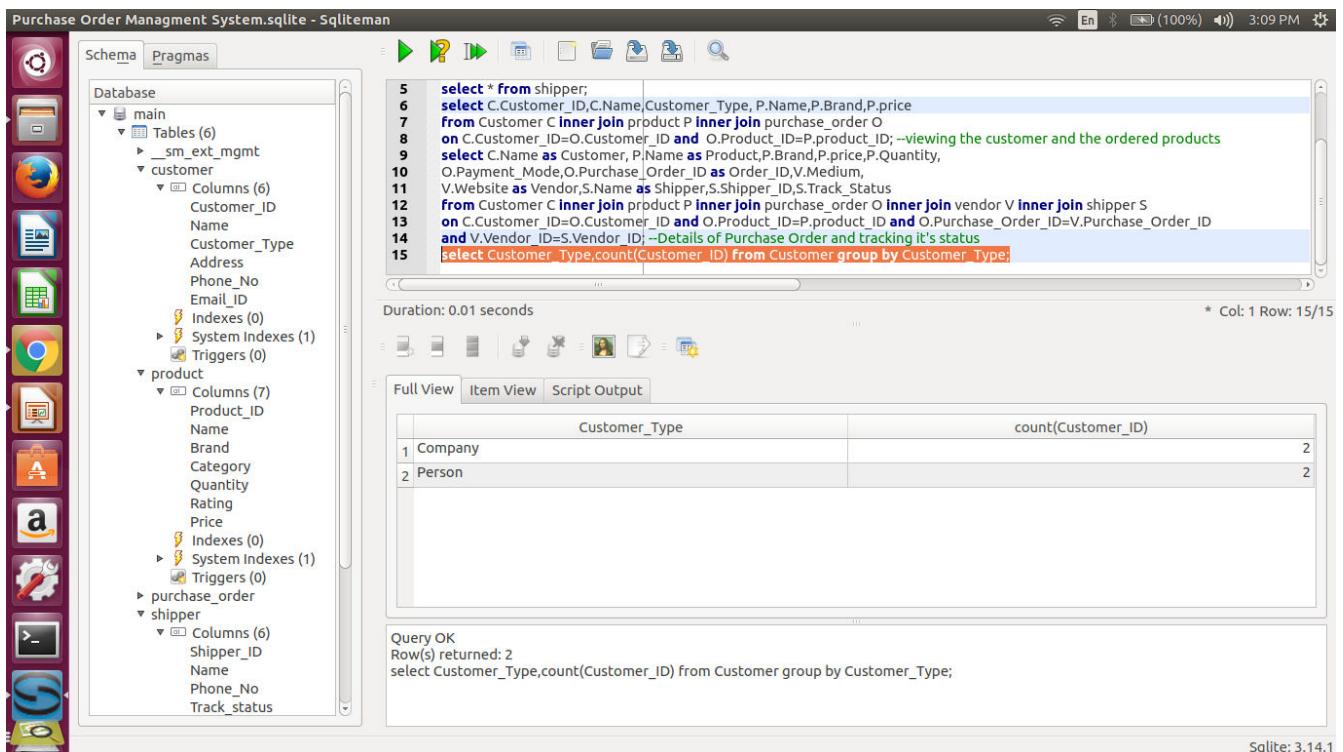
The results show two rows of data:

Customer	Product	Brand	Price	Quantity	Payment_Mode	Order_ID	Medium	Vendor	Shipper	Shipper_ID	Track_Status
1 hsha	Laptop	Apple	500	1	CreditCard	pur_111	Amazon	zinus.com	fedex	shp_111	on the way
2 Sha	PlayStation4	Sony	200	1	PayPal	pur_222	ebay	mehta.com	DHL	shp_222	on the way

The status bar indicates Duration: 0.024 seconds and Col: 29 Row: 14/14. The bottom status bar shows Sqlite: 3.14.1.

9.) Group By Query :

Counting customers grouped on their type(Company/Person):



The screenshot shows the Sqliteman interface with the same database. The left sidebar shows the schema. The main window contains a SQL query window with the following code:

```

5 select * from shipper;
6 select C.Customer_ID,C.Name,Customer_Type,P.Name,P.Brand,P.Price
7 from Customer C inner join product P inner join purchase_order O
8 on C.Customer_ID=O.Customer_ID and O.Product_ID=P.product_ID; --viewing the customer and the ordered products
9 select C.Name as Customer,P.Name as Product,P.Brand,P.Price,P.Quantity,
10 O.Payment_Mode,O.Purchase_Order_ID as Order_ID,V.Medium,
11 V.Website as Vendor,S.Name as Shipper,S.Shipper_ID,S.Track_Status
12 from Customer C inner join product P inner join purchase_order O inner join vendor V inner join shipper S
13 on C.Customer_ID=O.Customer_ID and O.Product_ID=P.product_ID and O.Purchase_Order_ID=V.Purchase_Order_ID
14 and V.Vendor_ID=S.Vendor_ID; --Details of Purchase Order and tracking its status
15 select Customer_Type,count(Customer_ID) from Customer group by Customer_Type;

```

The results show two rows of data:

Customer_Type	count(Customer_ID)
1 Company	2
2 Person	2

The status bar indicates Duration: 0.01 seconds and Col: 1 Row: 15/15. The bottom status bar shows Sqlite: 3.14.1.

10.) Order By Query :

The screenshot shows the Sqliteman interface for a Purchase Order Management System database. The left pane displays the database schema with tables: main, customer, product, purchase_order, and shipper. The right pane shows a SQL query window with the following code:

```
7 from Customer C inner join product P inner join purchase_order O
8 on C.Customer_ID=O.Customer_ID and O.Product_ID=P.product_ID; --viewing the customer and the ordered products
9 select C.Name as Customer, P.Name as Product,P.Brand,P.Price,P.Quantity,
10 O.Payment_Mode,O.Purchase_Order_ID as Order_ID,V.Medium,
11 V.Website as Vendor,S.Name as Shipper,S.Shipper_ID,S.Track_Status
12 From Customer C inner join product P inner join purchase_order O inner join vendor V inner join shipper S
13 on C.Customer_ID=O.Customer_ID and O.Product_ID=P.product_ID and O.Purchase_Order_ID=V.Purchase_Order_ID
14 and V.Vendor_ID=S.Vendor_ID; --Details of Purchase Order and tracking it's status
15 select Customer_Type,count(Customer_ID) from Customer group by Customer_Type;
16 Insert into product(Product_ID,Name,Brand,Category,Quantity,Rating,Price) values('pro_234','Lenevo','Electronics',1,3,400);
17 select Product_ID,Name as Product,Brand,Category,Price,Rating from product order by Rating;
```

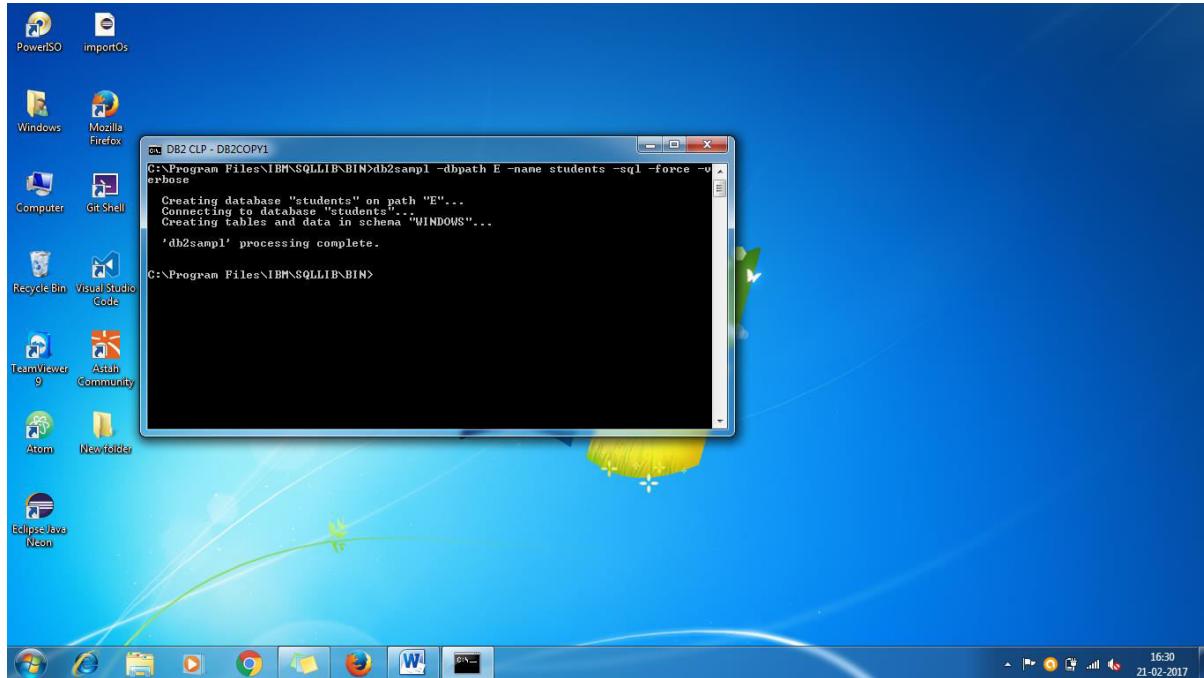
The results of the query are displayed in a table:

Product_ID	Product	Brand	Category	Price	Rating
1 pro_666	BOOK	Wren&Martin	Books	30	3
2 pro_234	Laptop	Lenevo	Electronics	400	3
3 pro_123	Laptop	Apple	Electronics	500	4
4 pro_999	PlayStation4	Sony	Electronics	200	5

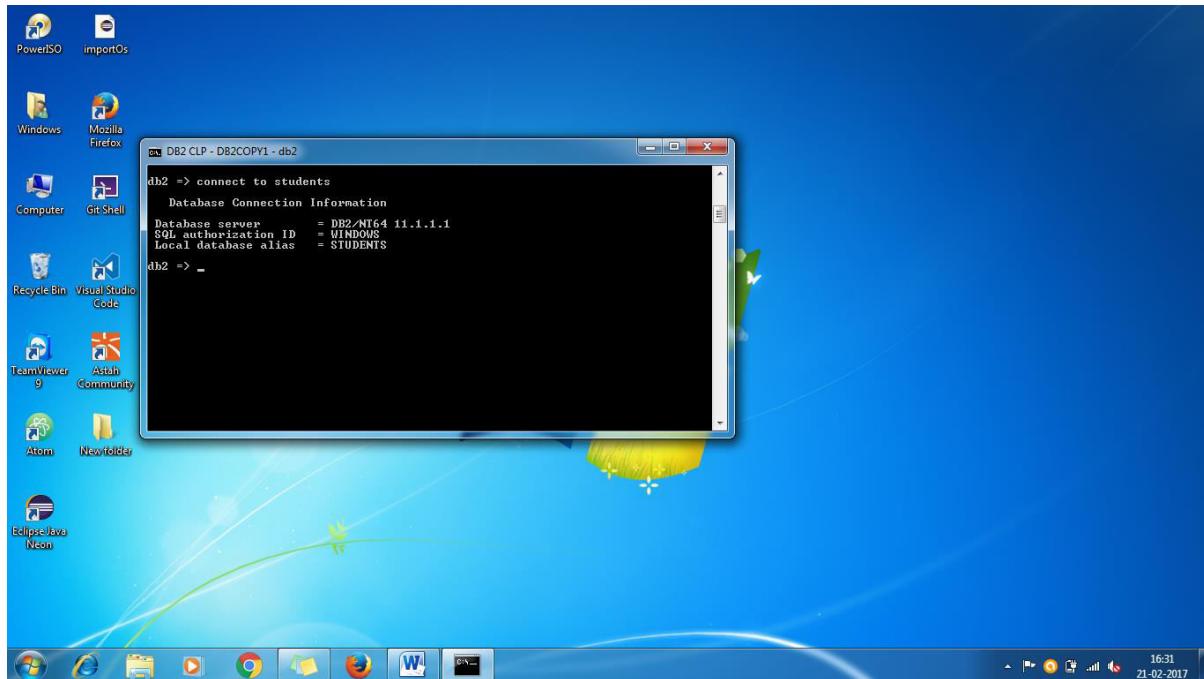
Below the table, the message "Query OK Row(s) returned: 4" is shown, along with the executed query: "select Product_ID,Name as Product,Brand,Category,Price,Rating from product order by Rating;"

Steps to create a database using DB2 Express C:

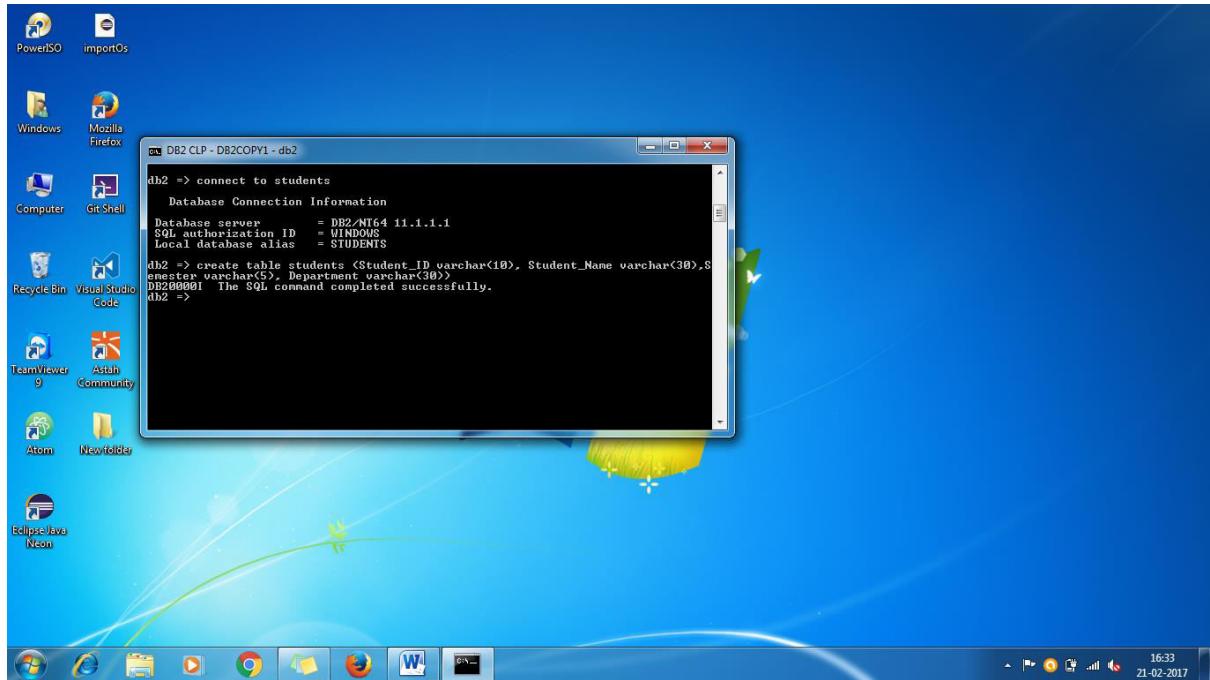
1. Using db2sampl command to create a sample database called “students”.



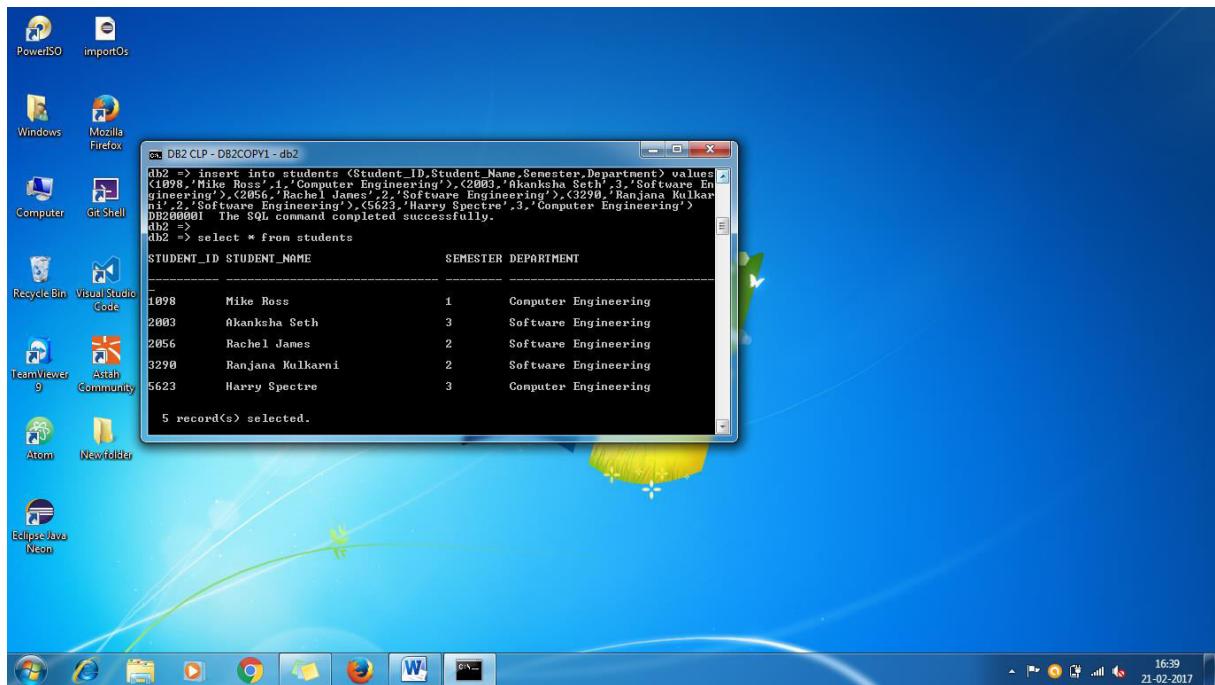
2. Connecting to the database “students”.



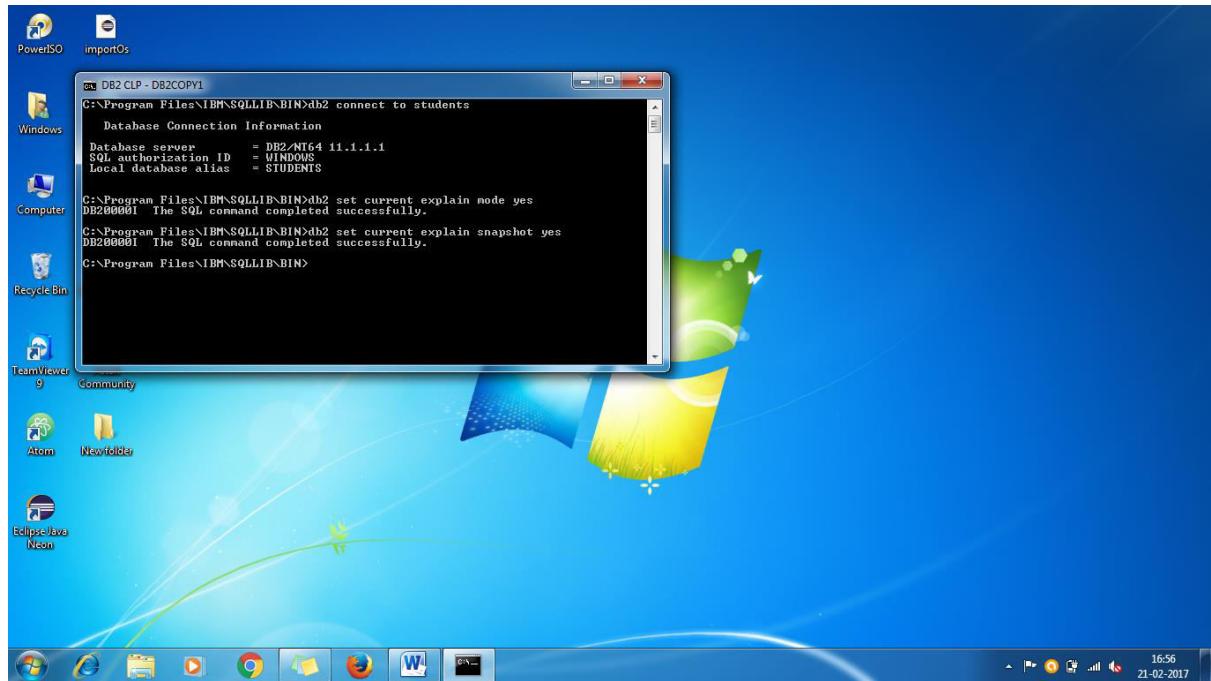
3. Creating a table “students” in the database using the ‘create table’ query.



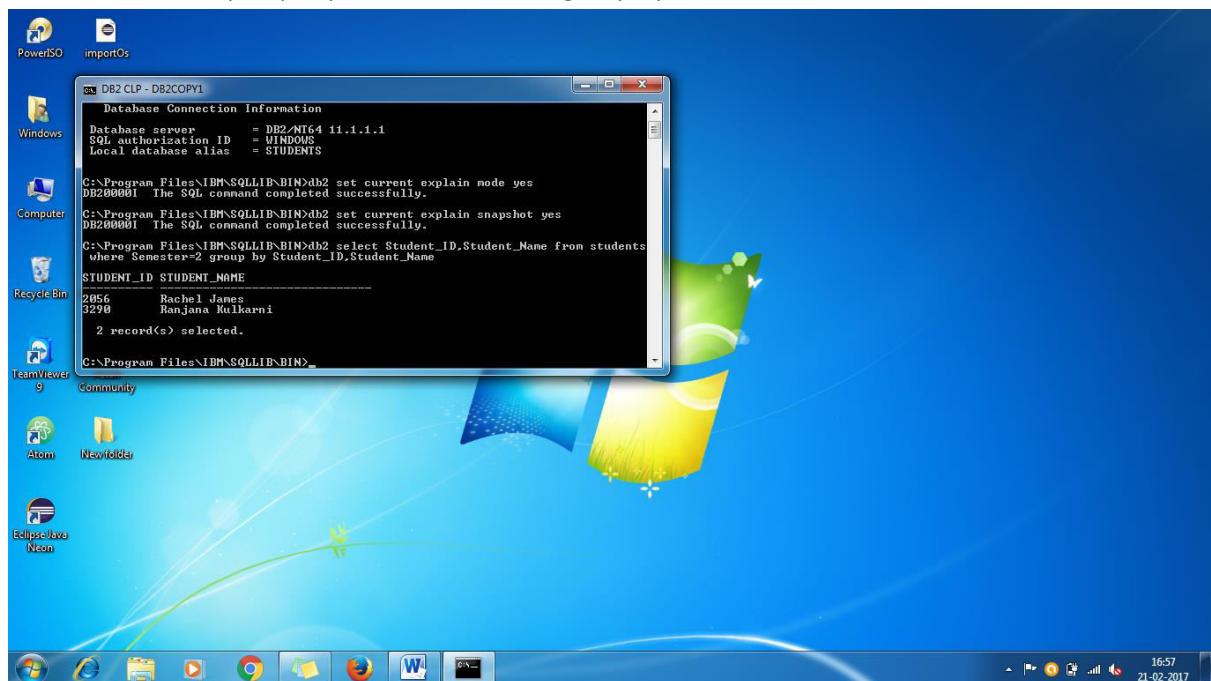
4. Insert sample data into the table “students” using ‘insert into table-name’ query.



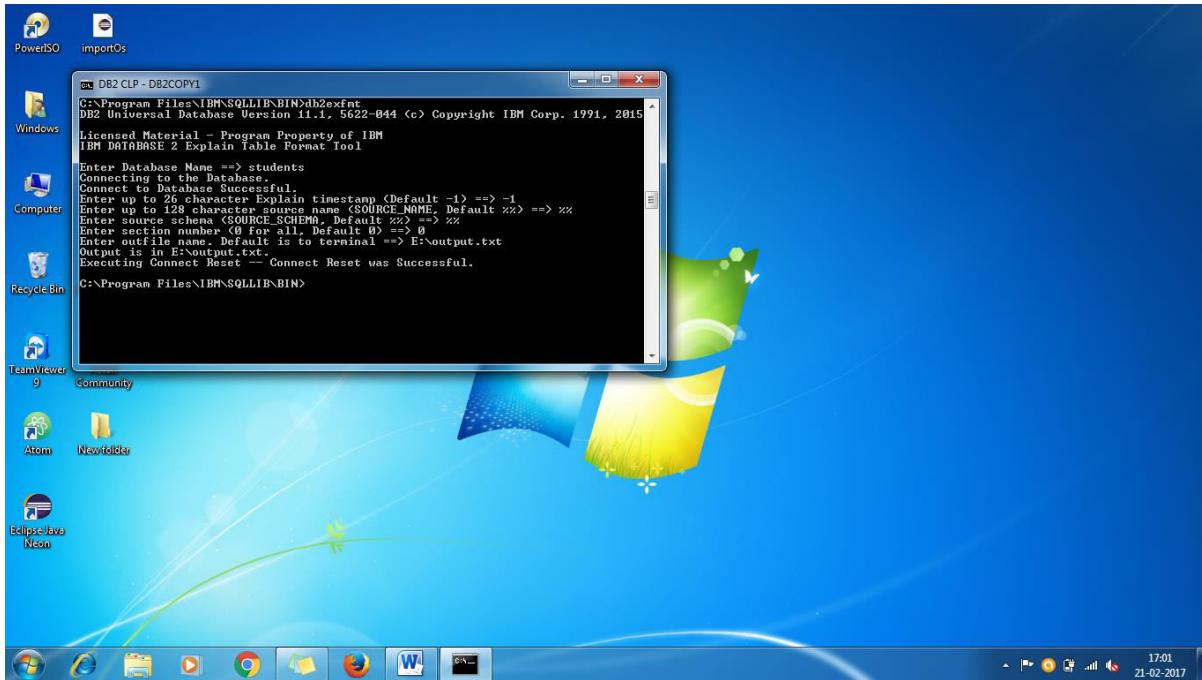
5. Setting the explain mode and snapshot to Yes using the 'db2 set current explain mode' query.



6. Run a sample query with 'where' and 'group by' clauses.



7. Generating query explain plan using 'db2exfmt' tool.



8. Snapshots of the 'db2exfmt' output.

```
output - Notepad
File Edit Format View Help
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: STUDENTS
SOURCE_NAME: SQLC2026
SOURCE_SCHEMA: NULLID
SOURCE_VERSION:
EXPLAIN_TIME: 2017-02-21-16.57.44.717000
EXPLAIN_REQUESTER: WINDOWS

Database Context:
-----
Parallelism: None
CPU Speed: 5.274511e-007
Comm Speed: 0
Buffer Pool size: 250
Sort Heap size: 256
Datafile 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
  Student_ID,
  Student_Name
```

```

output - Notepad
File Edit Format View Help
Original statement:
-----
select
    Student_ID,
    Student_Name
from
    students
where
    Semester=2
group by
    Student_ID,
    Student_Name

Optimized Statement:
-----
SELECT
    Q3.STUDENT_ID AS "STUDENT_ID",
    Q3.STUDENT_NAME AS "STUDENT_NAME"
FROM
    (SELECT
        Q2.STUDENT_ID,
        Q2.STUDENT_NAME
    FROM
        (SELECT
            Q1.STUDENT_ID,
            Q1.STUDENT_NAME
        FROM
            WINDOWS.STUDENTS AS Q1
        WHERE
            (DECFLOAT(Q1.SEMESTER, 34, '.') = 2)
        ) AS Q2
    GROUP BY
        Q2.STUDENT_NAME,
        Q2.STUDENT_ID
    ) AS Q3

Access Plan:
-----
Total Cost: 6.84174
Query Degree: 1

ROWS
RETURN
( 1)
Cost
I/O
|
0.2
GRPBY
( 2)
6.84174
1
|
0.2
TBSAN
( 3)
6.84159
1
|
0.2
SORT
( 4)
6.84091
1
|
0.2
TBSAN
( 5)
6.83948
1
|
5
TABLE: WINDOWS
STUDENTS
Q1

Extended Diagnostic Information:
-----
No extended Diagnostic Information for this statement.

Plan Details:
-----

```

```

output - Notepad
File Edit Format View Help
Plan Details:
-----
1) RETURN: (Return Result)
    Cumulative Total Cost:      6.84174
    Cumulative CPU Cost:       69662
    Cumulative I/O Cost:        1
    Cumulative Re-Total Cost:   0.00879208
    Cumulative Re-CPU Cost:    16669
    Cumulative Re-I/O Cost:    0
    Cumulative First Row Cost: 6.84172
    Estimated Bufferpool Buffers: 0

    Arguments:
    -----
    BLDLEVEL: (Build level)
        DB2 v11.1.1010.160 : s1612051900
    HEAPUSE : (Maximum statement Heap Usage)
        96 Pages
    PLANID : (Access plan identifier)
        c4daff39c3b7a
    PREPTIME: (Statement prepare time)
        363 milliseconds
    SEMEVID : (Semantic environment identifier)
        a45949157ebc089a
    STMTHAEP: (Statement heap size)
        8192
    STMTID : (Normalized statement identifier)
        d21aff391602871c

    Input Streams:
    -----
    5) From Operator #2
        Estimated number of rows:      0.2
        Number of columns:            2
        Subquery predicate ID:        Not Applicable
        Column Names:
        -----
        +Q4.STUDENT_NAME+Q4.STUDENT_ID

2) GRPBY : (Group By)
    Cumulative Total Cost:      6.84174
    Cumulative CPU Cost:       69662
    Cumulative I/O Cost:        1
    Cumulative Re-Total Cost:   0.00879208
    Cumulative Re-CPU Cost:    16669
    Cumulative Re-I/O Cost:    0

```

```

output - Notepad
File Edit Format View Help
2) GRPBY : (Group By)
    Cumulative Total Cost:      6.84174
    Cumulative CPU Cost:       69662
    Cumulative I/O Cost:        1
    Cumulative Re-Total Cost:   0.00879208
    Cumulative Re-CPU Cost:    16669
    Cumulative Re-I/O Cost:    0
    Cumulative First Row Cost: 6.84172
    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.STUDENT_ID
        2: Q2.STUDENT_NAME
    ONEFETCH: (One Fetch flag)
        FALSE

    Input Streams:
    -----
    4) From Operator #3
        Estimated number of rows:      0.2
        Number of columns:            2
        Subquery predicate ID:        Not Applicable
        Column Names:
        -----
        +Q2.STUDENT_NAME(A)+Q2.STUDENT_ID(A)

    Output Streams:
    -----
    5) To Operator #1
        Estimated number of rows:      0.2
        Number of columns:            2
        Subquery predicate ID:        Not Applicable
        Column Names:
        -----
        +Q4.STUDENT_NAME+Q4.STUDENT_ID

```

```

output - Notepad
File Edit Format View Help
3) TBSCAN: (Table Scan)
    Cumulative Total Cost:      6.84159
    Cumulative CPU Cost:       69362
    Cumulative I/O Cost:        1
    Cumulative Re-Total Cost:   0.00863385
    Cumulative Re-CPU Cost:    16369
    Cumulative Re-I/O Cost:     0
    Cumulative First Row Cost: 6.84159
    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 #4
        Estimated number of rows:      0.2
        Number of columns:            2
        Subquery predicate ID:        Not Applicable
        Column Names:
        +Q1.STUDENT_NAME(A)+Q1.STUDENT_ID(A)

    Output Streams:
    -----
    4) To Operator #2
        Estimated number of rows:      0.2
        Number of columns:            2
        Subquery predicate ID:        Not Applicable
        Column Names:
        +Q2.STUDENT_NAME(A)+Q2.STUDENT_ID(A)

-----
```

17:04
21-02-2017

```

File Edit Format View Help
4) SORT : (Sort)
    Cumulative Total Cost:      6.84091
    Cumulative CPU Cost:       68075
    Cumulative I/O Cost:        1
    Cumulative Re-Total Cost:   0.00779678
    Cumulative Re-CPU Cost:    14782
    Cumulative Re-I/O Cost:     0
    Cumulative First Row Cost: 6.84091
    Estimated Bufferpool Buffers: 1

    Arguments:
    -----
    DUPLWARN: (Duplicates warning flag)
    TRUE
    KEYS : (Key cardinality)
    1
    NUMROWS : (Estimated number of rows)
    1
    ROWWIDTH: (Estimated width of rows)
    32.000000
    SORTKEY : (Sort Key column)
    1: Q1.STUDENT_NAME(A)
    2: Q1.STUDENT_ID(A)
    TEMPSPACE: (Temporary Table Page Size)
    8192
    UNIQUE : (Uniqueness required flag)
    TRUE

    Input Streams:
    -----
    2) From Operator #5
        Estimated number of rows:      0.2
        Number of columns:            2
        Subquery predicate ID:        Not Applicable
        Column Names:
        +Q1.STUDENT_NAME+Q1.STUDENT_ID

    Output Streams:
    -----
    3) To Operator #3
        Estimated number of rows:      0.2
        Number of columns:            2
        Subquery predicate ID:        Not Applicable
        Column Names:
        -----
```

17:05
21-02-2017

```

output - Notepad
File Edit Format View Help
Column Names:
-----+Q1.STUDENT_NAME(A)+Q1.STUDENT_ID(A)

5) TBSCAN: (Table Scan)
    Cumulative Total Cost:      6.83948
    Cumulative CPU Cost:        65375
    Cumulative I/O Cost:        1
    Cumulative Re-IO Cost:      0.00779678
    Cumulative Re-CPU Cost:     14782
    Cumulative Re-I/O Cost:     0
    Cumulative First Row Cost:  6.8382
    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:
-----
4) Sargable Predicate,
    Comparison Operator: Equal (=)
    Subquery Input Required: No
    Filter Factor: 0.04

```

17:05
21-02-2017

```

output - Notepad
File Edit Format View Help
Predicates:
-----
4) Sargable Predicate,
    Comparison Operator: Equal (=)
    Subquery Input Required: No
    Filter Factor: 0.04

Predicate Text:
(DECFLOAT(Q1_SEMESTER, 34, '.') = 2)

Input Streams:
-----
1) From Object WINDOWS.STUDENTS
    Estimated number of rows: 5
    Number of columns: 4
    Subquery predicate ID: Not Applicable
    Column Names:
-----+Q1.$RID$+Q1.STUDENT_NAME+Q1.STUDENT_ID
+Q1_SEMESTER

Output Streams:
-----
2) To Operator #4
    Estimated number of rows: 0.2
    Number of columns: 2
    Subquery predicate ID: Not Applicable
    Column Names:
-----+Q1.STUDENT_NAME+Q1.STUDENT_ID

Objects Used in Access Plan:
-----
Schema: WINDOWS
Name: STUDENTS
Type: Table

Extended Statistics Information:
-----
```

17:05
21-02-2017

```

output - Notepad
File Edit Format View Help
Extended statistics Information:
-----
Tablesce Context:
-----
Name: IBMDB2SAMPLEREL
Overhead: 6.725000
Transfer Rate: 0.080000
Prefetch Size: 32
Extent Size: 32
Type: Database managed
Partition Group Name: NULLP
Buffer Pool Identifier: 0

Base Table Statistics:
-----
Name: STUDENTS
Schema: WINDOWS
Number of Columns: 4
Number of Pages with Rows: 1
Number of Delta Pages: 0
Number of Paged: 1
Number of Rows: 5
Table Overflow Record Count: 0
Width of Rows: 47
Time of Creation: 2017-02-21-16.33.18.072000
Last Statistics Update: 2017-02-21-16.42.39.116000
Primary Tablespace: IBMDB2SAMPLEREL
Tablespace for Indexes: NULLP
Tablespace for Long Data: NULLP
Number of Referenced Columns: 3
Number of Indexes: 0
Volatile Table: No
Number of Active Blocks: -1
Number of Column Groups: 0
Number of Data Partitions: 1
Average Row Compression Ratio: -1.000000
Percent Rows Compressed: -1.000000
Average Compressed Row Size: 5
Statistics Type: S
Number of Worker Nodes: 0

Column Information:
-----
Number: 2
Name: STUDENT_NAME
DFS Partition Column: No
Statistics Available: Yes

Column Statistics:
-----
column statistics:
-----
Schema name of the column type: SYSIBM
Name of column type: VARCHAR
Maximum column length: 30
Scale for decimal or timestamp column: 0
Number of distinct column values: 5
Average column length: 18
Number of most frequent values: -1
Number of quantiles: 5
Second highest data value: Rachel James
Second lowest data value: Harry Spectre
Column sequence in partition key: 0
Average number of sub-elements: -1
Average length of delimiters: -1
Percentage encoded column values: -1
Column clustering: -1
Average encoded column length: -1.000000

column Distribution statistics:
-----
Quantile statistics:
----- Valcount Value Distcount -----
1 Akanksha Seth 0
2 Harry Spectre 0
3 Mike Ross 0
4 Ranjana Kulkarni 0
5 Ranjana Kulkarni 0

Column Information:
-----
Number: 1
Name: STUDENT_ID
DFS Partition Column: No
Statistics Available: Yes

Column Statistics:
-----
Schema name of the column type: SYSIBM
Name of column type: VARCHAR
Maximum column length: 10
Scale for decimal or timestamp column: 0
Number of distinct column values: 5
Average column length: 9
Number of most frequent values: -1
Number of quantiles: 5
Second highest data value: 3290
Second lowest data value: 2003
Column sequence in partition key: 0
Average number of sub-elements: -1

```

```

output - Notepad
File Edit Format View Help

column statistics:
-----
Schema name of the column type: SYSIBM
Name of column type: VARCHAR
Maximum column length: 10
Scale for decimal or timestamp column: 0
Number of distinct column values: 5
Average column length: 9
Number of most frequent values: -1
Number of quantiles: 5
Second highest data value: 3290
Second lowest data value: 2003
Column sequence in partition key: 0
Average number of sub-elements: -1
Average length of delimiters: -1
Percentage encoded column values: -1
Column clustering: -1
Average encoded column length: -1.000000

column distribution statistics:
-----
Quantile statistics:
  Valcount   Value   Distcount
  -----  -----
  1       1098      0
  2       2003      0
  3       2056      0
  4       5623      0
  5       5623      0

Column Information:
-----
Number: 3
Name: SEMESTER
DFS Partition Column: No
Statistics Available: Yes

column statistics:
-----
Schema name of the column type: SYSIBM
Name of column type: VARCHAR
Maximum column length: 5
Scale for decimal or timestamp column: 0
Number of distinct column values: 3
Average column length: 6
Number of most frequent values: 2
Number of quantiles: 4
Second highest data value: 3
Second lowest data value: 1
Column sequence in partition key: 0

output - Notepad
File Edit Format View Help

----- quantile statistics:
  Valcount   Value   Distcount
  -----  -----
  1       1098      0
  2       2003      0
  3       2056      0
  4       5623      0
  5       5623      0

Column Information:
-----
Number: 3
Name: SEMESTER
DFS Partition Column: No
Statistics Available: Yes

column statistics:
-----
Schema name of the column type: SYSIBM
Name of column type: VARCHAR
Maximum column length: 5
Scale for decimal or timestamp column: 0
Number of distinct column values: 3
Average column length: 6
Number of most frequent values: 2
Number of quantiles: 4
Second highest data value: 3
Second lowest data value: 1
Column sequence in partition key: 0
Average number of sub-elements: -1
Average length of delimiters: -1
Percentage encoded column values: -1
Column clustering: -1
Average encoded column length: -1.000000

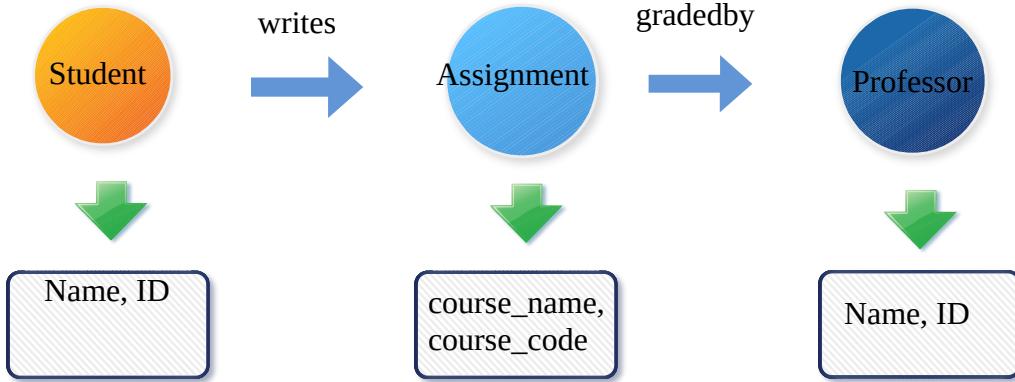
column distribution statistics:
-----
Frequency statistics:
  Valcount   Value
  -----  -----
  2       2
  2       3

----- quantile statistics:
  Valcount   Value   Distcount
  -----  -----
  1       1       0
  2       2       0
  3       3       0
  5       3       0

```

IBM Bluemix Graph DB:

Dataset Model:



1.) Created Schemas for Student, Assignment and Professor.

Creating Student Schema :

```
akansha@akansha-Lenovo-G50-80:~$ curl -X POST -d '{"propertyKeys": [{"name": "name", "datatype": "String", "cardinality": "SINGLE"}, {"name": "ID", "datatype": "Integer", "cardinality": "SINGLE"}], "vertexIndexes": [{"name": "vByName", "propertyKeys": ["name"]}], "composite": true, "unique": true}, {"name": "vByID", "propertyKeys": ["ID"], "composite": true, "unique": true}}' -H 'Authorization: gds-token ZDIxNjMxGItYWFiy00YzVlWEzN2QtYzdhMjh1MWFjNmNh0jE00DgxNzcXjk4MDM65jJBQUDyOFV1QjB2Uk85W084MDRaaTUrRDdaMEEdDYkJ4cjBNN292a2xrYz0=' -H 'content-type: application/json' https://ibmgraph-alpha.ng.bluemix.net/7b319cd6-c014-4389-b67d-031e3733a0de/g/schema{"requestId": "942832a8-5ff8-443b-8b63-3e18412a0428", "status": {"message": "", "code": 200, "attributes": {}}, "result": {"data": [{"propertyKeys": [{"name": "name", "datatype": "String", "cardinality": "SINGLE"}, {"name": "ID", "datatype": "Integer", "cardinality": "SINGLE"}], "vertexIndexes": [{"name": "vByName", "composite": true, "unique": true, "propertyKeys": ["name"]}], "composite": true, "unique": true, "requiresReIndex": false, "type": "vertex"}, {"name": "vByID", "composite": true, "unique": true, "propertyKeys": ["ID"], "requiresReIndex": false, "type": "vertex"}, "edgeIndexes": []}, "meta": {}}akansha@akansha-Lenovo-G50-80:~$
```

2.) Inserting values in Student Schema :

```
akansha@akansha-Lenovo-G50-80:~$ curl -X POST -d '{"name": "Jane", "ID": 102, "label": "Student"}' -H 'content-type: application/json' -H 'Authorization: gds-token ZDIxNjMxGItYWFiy00YzVlWEzN2QtYzdhMjh1MWFjNmNh0jE00DgxNzcXjk4MDM65jJBQUDyOFV1QjB2Uk85W084MDRaaTUrRDdaMEEdDYkJ4cjBNN292a2xrYz0=' https://ibmgraph-alpha.ng.bluemix.net/7b319cd6-c014-4389-b67d-031e3733a0de/g/vertices{"requestId": "7a56fb8a-f0a2-4437-b5a2-98ba996b6b3f", "status": {"message": "", "code": 200, "attributes": {}}, "result": {"data": [{"id": 4264, "label": "Student", "type": "vertex", "properties": {"name": [{"id": "179-3ag-sl", "value": "Jane"}]}, "ID": [{"id": "1lh-3ag-1l1", "value": 102}]}], "meta": {}}akansha@akansha-Lenovo-G50-80:~$ akansha@akansha-Lenovo-G50-80:~$ akansha@akansha-Lenovo-G50-80:~$ curl -X POST -d '{"name": "John", "ID": 103, "label": "Student"}' -H 'content-type: application/json' -H 'Authorization: gds-token ZDIxNjMxGItYWFiy00YzVlWEzN2QtYzdhMjh1MWFjNmNh0jE00DgxNzcXjk4MDM65jJBQUDyOFV1QjB2Uk85W084MDRaaTUrRDdaMEEdDYkJ4cjBNN292a2xrYz0=' https://ibmgraph-alpha.ng.bluemix.net/7b319cd6-c014-4389-b67d-031e3733a0de/g/vertices{"requestId": "7d1f76c0-bbe1-4ce8-8ca0-ba52abfcadd0", "status": {"message": "", "code": 200, "attributes": {}}, "result": {"data": [{"id": 4128, "label": "Student", "type": "vertex", "properties": {"name": [{"id": "16s-36o-sl", "value": "John"}]}, "ID": [{"id": "1lh-36o-1l1", "value": 103}]}], "meta": {}}akansha@akansha-Lenovo-G50-80:~$ akansha@akansha-Lenovo-G50-80:~$ akansha@akansha-Lenovo-G50-80:~$
```

Using the same command, created schemas for Assignment and Professor.

3.) Adding vertex to above schemas using gremlin query in IBM Graph Bluemix :

The screenshot shows the IBM Graph Bluemix interface. On the left, there's a sidebar with icons for Upload, Samples, and Resources. The main area is a code editor with a syntax-highlighted Gremlin query. The code defines three vertices: v1 (Student), v2 (Virtualization), and v3 (professor). It also creates an edge from v1 to v2 labeled 'writes' and another edge from v2 to v3 labeled 'gradedby'. Below the code editor, there's a preview pane showing the resulting graph structure.

```
def v1 = graph.addVertex('name','Sam',label,'Student','ID',108); def v2 = graph.addVertex('course_name','Virtualization',label,'assignment','course_code',275); def v3 = graph.addVertex('name','Prof. P.Parker',label,'professor','ID',207); v1.addEdge('writes',v2);v2.addEdge('gradedby',v3)
```

```
def v1 = graph.addVertex('name','Sam',label,'Student','ID',108); def v2 = graph.addVertex('cou
```

```
1  [ 2  { 3      "id": "1zl-39k-6o7p-39c", 4      "label": "gradedby", 5      "type": "edge", 6      "inVLabel": "professor", 7      "outVLabel": "assignment", 8      "inV": 4224, 9      "outV": 4232 10     } 11   ]
```

4.) Viewing Student Vertex :

The screenshot shows the IBM Graph Bluemix interface. The sidebar includes Upload, Samples, and Resources. The main area contains a Gremlin query to find a vertex named 'Sam'. The results pane shows a single vertex node with properties: id (4288), label (Student), type (vertex), name (Sam), and ID (108). A green circular badge labeled 'Student' is placed next to the vertex node in the preview area. At the bottom, there are filter options for Label, Type, and Properties, and a status bar indicating 1 vertex found.

```
def g = graph.traversal(); g.V().has("name","Sam")
```

```
def g = graph.traversal(); g.V().has("name","Sam")
```

```
1  [ 2  { 3      "id": 4288, 4      "label": "Student", 5      "type": "vertex", 6      "properties": { 7          "name": [ 8              { 9                  "id": "17c-3b4-s1", 10                  "value": "Sam" 11              } 12          ], 13          "ID": [ 14              { 15                  "id": "1lk-3b4-1l1", 16                  "value": 108 17              } 18          ] 19      } 20  } 21  ]
```

Filter: Label Type Properties Vertices: 1

Graph: g

4.) Viewing Assignment vertex :

The screenshot shows the IBM Graph interface with a query editor and a results panel. The query editor contains the following code:

```
1 def g = graph.traversal(); g.V().has("course_code", "275")
```

The results panel displays the properties of the found vertex:

```
def g = graph.traversal(); g.V().has("course_code", "275")
3   "id": 4232,
4   "label": "assignment",
5   "type": "vertex",
6   "properties": [
7     "course_code": [
8       {
9         "id": "1ld-39k-4flx",
10        "value": 275
11      }
12    ],
13   "course_name": [
14     {
15       "id": "175-39k-4eth",
16       "value": "Virtualization"
17     }
18   ]
```

A green circle with the text "assig." is overlaid on the results panel.

5.) Viewing Professor vertex :

The screenshot shows the IBM Graph interface with a query editor and a results panel. The query editor contains the following code:

```
1 def g = graph.traversal(); g.V().has("name", "Prof. P.Parker")
```

The results panel displays the properties of the found vertex:

```
def g = graph.traversal(); g.V().has("name", "Prof. P.Parker")
3   "id": 4224,
4   "label": "professor",
5   "type": "vertex",
6   "properties": [
7     "name": [
8       {
9         "id": "174-39c-sl",
10        "value": "Prof. P.Parker"
11      }
12    ],
13   "ID": [
14     {
15       "id": "1lc-39c-ll1",
16       "value": 207
17     }
18   ]
```

A green circle with the text "prof." is overlaid on the results panel.

6.) Viewing Student and Assignment vertices connected by an edge ‘writes’:

IBM Graph-rc > g

Tutorials [Query](#)

Upload Samples Resources

```
1 def g = graph.traversal(); g.V().has("name","Sam").outE("writes").inV().hasLabel("assignment").path()
```

def g = graph.traversal(); g.V().has("name","Sam").outE("writes").inV().hasLabel("assignment")
15 {
16 "id": "17c-3b4-sl",
17 "value": "Sam"
18 }]
19],
20 "ID": [
21 {
22 "id": "1lk-3b4-1l1",
23 "value": 108
24 }
25]}
26 },
27 {
28 "id": "1zs-3b4-4h79-39k",
29 "label": "writes",
30 "type": "edge"
31 }

Filter: [Label](#) [Type](#) [Properties](#)

Vertices: 2

Graph: g



7.) Viewing all vertices and edges, visualizing the whole graph application :

IBM Graph

Documentation Bluemix Dashboard

Tutorials [Query](#)

Upload Samples Resources

```
1 def g = graph.traversal();  
g.V().has("name","Sam").outE("writes").inV().hasLabel("assignment").outE("gradedby").inV().hasLabel("professor").path()
```

Graph: g

```
def g = graph.traversal(); g.V().has("name","Sam").outE("writes").inV().hasLabel("assignment")  
1 [ [ ]  
2 { [ ]  
3 "labels": [ [ ]  
4 [ ]  
5 [ ]  
6 [ ]  
7 [ ]  
8 [ ]  
9 ]]  
10 "objects": [ [ ]  
11 { [ ]  
12     "id": 4288,  
13     "label": "Student",  
14     "type": "vertex",  
15     "properties": { [ ]  
16 }
```

Filter: [Label](#) [Type](#) [Properties](#)

