



# **DEPARTMENT OF COMPUTER APPLICATION**

**MCA 'A'2024  
SEMESTER - II**

**Database Design and  
Implementation- E1CAB612**

**Class Project**  
**NAME : Shivangi Pathak**  
**REG. NO:2411022250027**

**Signature of the Faculty**

## TITLE: WORKING WITH A NOSQL DATABASE

### ORIENTDB



#### **What is NoSQL database?**

A **NoSQL database** is a type of database that provides a mechanism for storing and retrieving data that is different from traditional **relational databases (SQL databases)**.

#### **What is ORIENTDB?? Why choose Orientdb??**

OrientDB is a **multi-model NoSQL database** that combines the power of graphs with the flexibility of documents. It allows you to store data as **documents, graphs, objects, or key-values**, all in one database engine. This makes it quite unique compared to other databases that usually stick to one model.

## Here are some key reasons you might choose OrientDB:

### 1. Multi-Model Support

Store and query **graph**, **document**, **object**, and **key-value** data using one engine.

Avoid the complexity of using multiple databases for different types of data.

### 2. SQL-Like Query Language

Familiar SQL syntax with extensions for graph traversal.

Easier for teams coming from relational DBs.

### 3. Graph + Document Hybrid

Powerful for use cases like **social networks**, **recommendation engines**, or **network topologies**.

You get the performance and modeling benefits of graphs, plus the flexibility of documents

.

### 4. High Performance and Scalability

Supports horizontal scaling (sharding + replication)

Optimized for fast reads/writes and graph traversals.

### 5. Schema-less, Schema-full, or Mixed

Flexibility to define schema if needed—or go schema-less.

Great for rapidly evolving applications.

### 6. ACID Transactions

Unlike many NoSQL databases, OrientDB supports full ACID transactions.

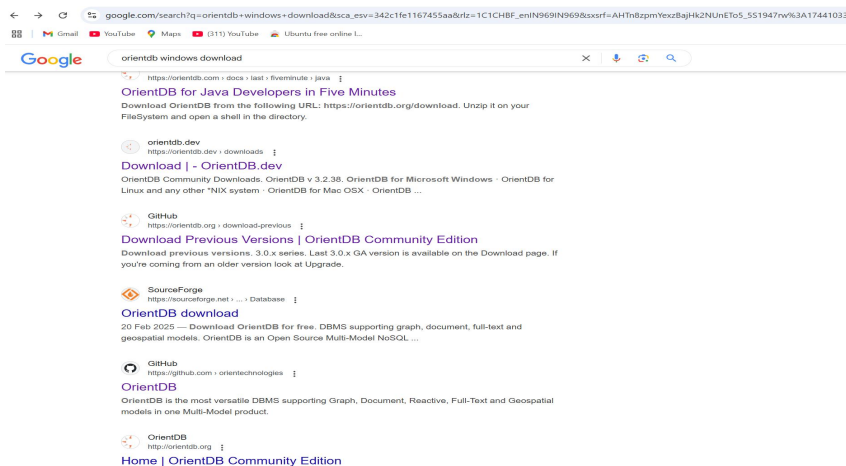
### 7. Embedded and Standalone

Can run embedded in Java apps or as a standalone DB server.



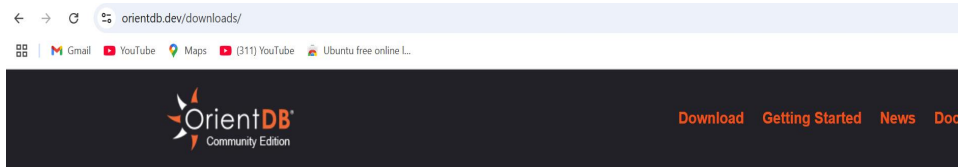
### INSTALLATION OF ORIENTDB:

**Step1: Open Chrome and search orient database download then click on mentioned link.**



**Step2: after clicking that we will get one more screen there click download option as we shown here.**

## DATABASE DESIGN AND IMPLEMENTATION



### OrientDB Community Downloads

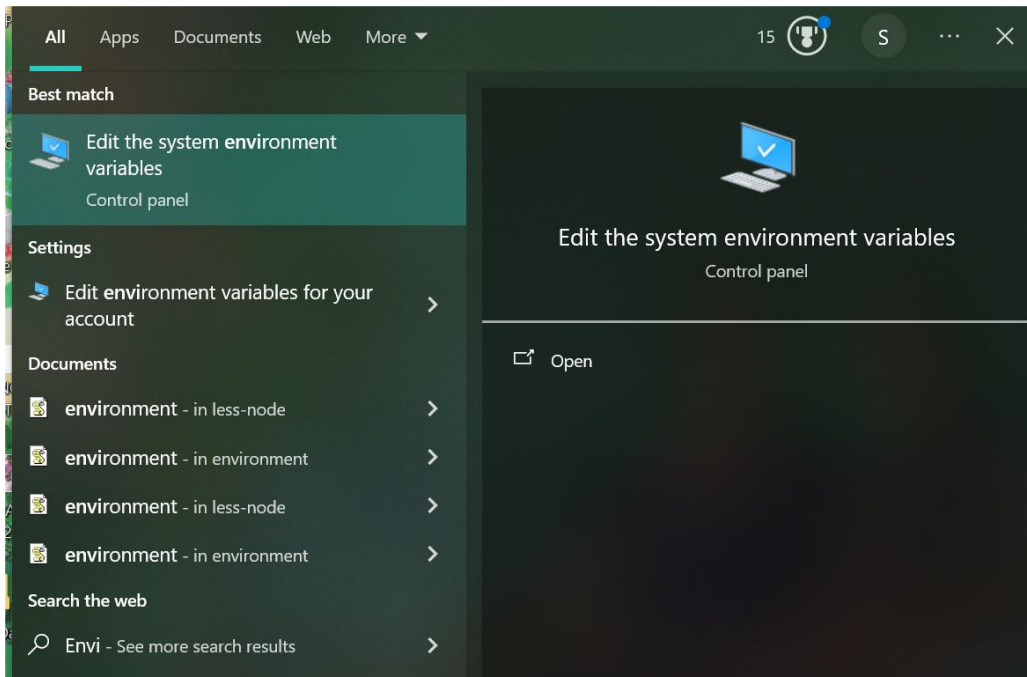
#### OrientDB v 3.2.38

- [OrientDB for Microsoft Windows](#)
- [OrientDB for Linux and any other \\*NIX system](#)
- [OrientDB for Mac OSX](#)
- [OrientDB Zip for any OS](#)
- [OrientDB with Gremlin Server - zip](#)
- [OrientDB with Gremlin Server - tar.gz](#)
- [OrientDB Enterprise Agent](#)

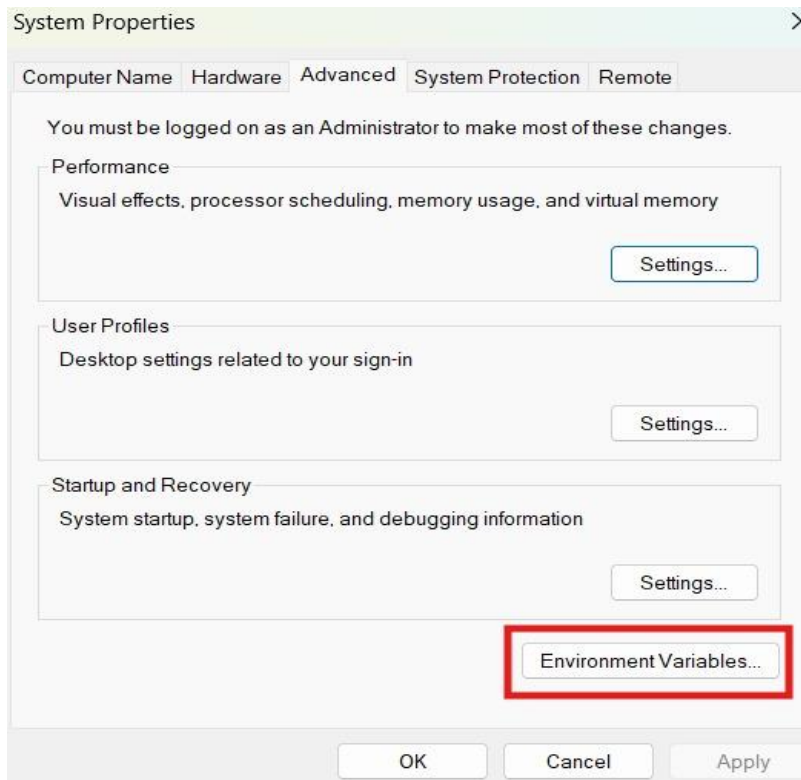
#### OrientDB v 3.1.19

- [OrientDB for Microsoft Windows](#)
- [OrientDB for Linux and any other \\*NIX system](#)
- [OrientDB for Mac OSX](#)
- [OrientDB Zip for any OS](#)
- [OrientDB with Gremlin Server - zip](#)

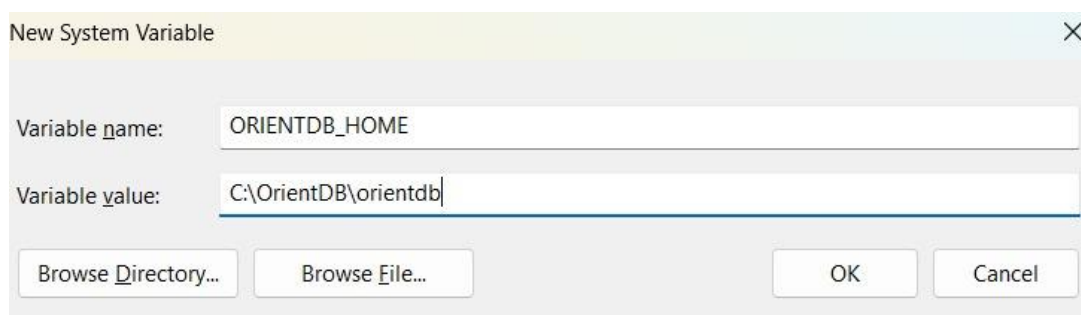
**Step3: go to windows and search for environmental variables then click on that,**



**Step4: in that windows search for environmental variables then click on that.**

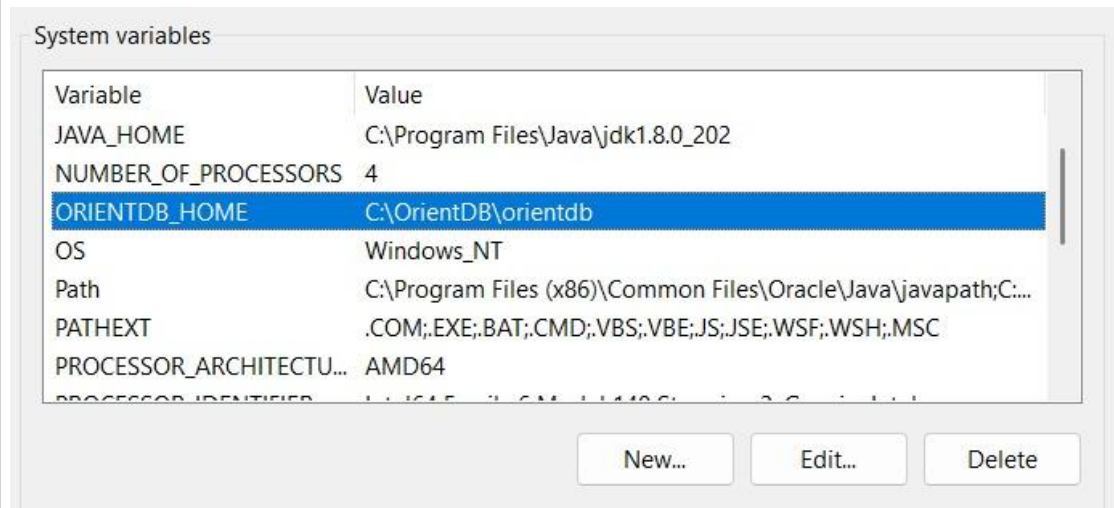


**Step5: in that windows create new variables name and variables value and then click on okk.**

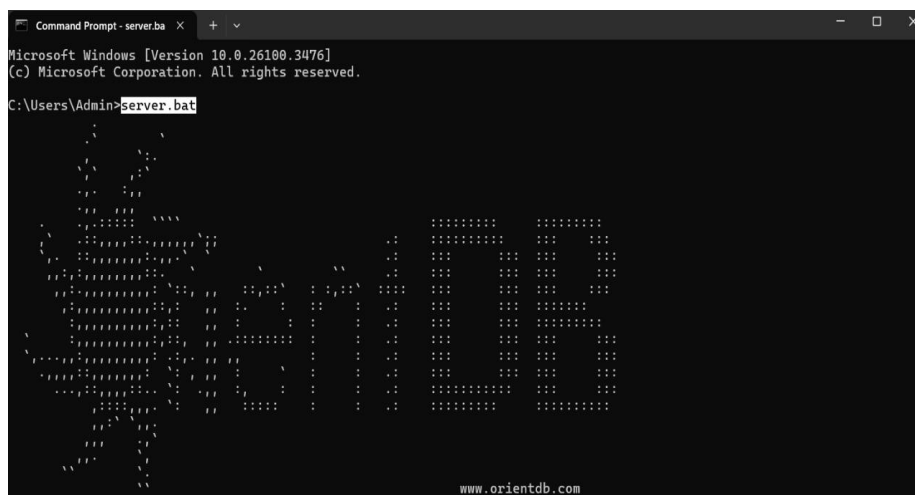


**Step6: in the system variable create the path.**

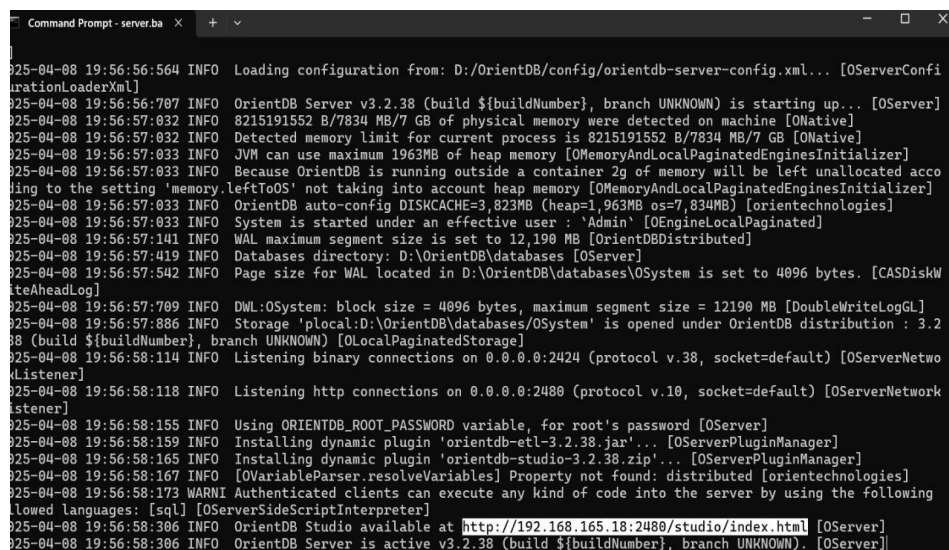
## DATABASE DESIGN AND IMPLEMENTATION



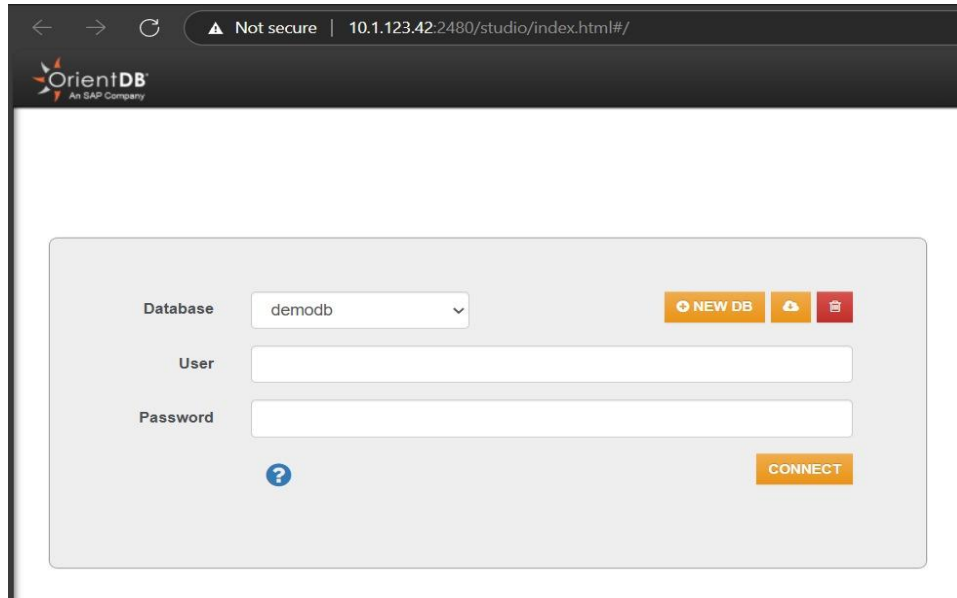
**Step7: open command prompt type server.bat.**



**Step8: Then click on below highlited link.**



### Step9: Creating a database.



### Step10: Clicking newDb option will come to create new database.

New Database

Name

CourseInfo

Server User

root

Server Password

.....

☐ Create Admin user

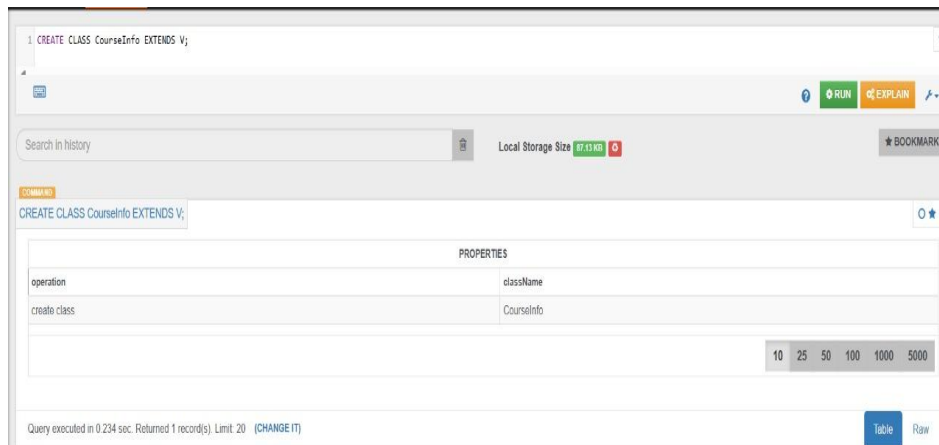
☐ Advanced Options

CLOSE

CREATE DATABASE

### Step11: to create class.





### Step12: Creating Property Command.

```
CREATE PROPERTY CourseInfo.courseId STRING;
CREATE PROPERTY CourseInfo.courseName STRING;
CREATE PROPERTY CourseInfo.instructor STRING;
CREATE PROPERTY CourseInfo.credits INTEGER;
CREATE PROPERTY CourseInfo.semester STRING;
```



### Step13: Inserted data into class .

```
INSERT INTO CourseInfo SET courseId = "C101", courseName = "DBMS", instructor = "Dr. Ravi", credits = 4, semester = "Sem 4";
INSERT INTO CourseInfo SET courseId = "C102", courseName = "OOP", instructor = "Dr. Meena", credits = 3, semester = "Sem 3";
INSERT INTO CourseInfo SET courseId = "C103", courseName = "Computer Networks", instructor = "Dr. Arjun", credits = 4, semester = "Sem 5";
INSERT INTO CourseInfo SET courseId = "C104", courseName = "Software Engineering", instructor = "Dr. Komal", credits = 3, semester = "Sem 6";
INSERT INTO CourseInfo SET courseId = "C105", courseName = "Web Development", instructor = "Ms. Shivangi", credits = 3, semester = "Sem 4";
INSERT INTO CourseInfo SET courseId = "C106", courseName = "AI Fundamentals", instructor = "Dr. Ramesh", credits = 4, semester = "Sem 6";
INSERT INTO CourseInfo SET courseId = "C107", courseName = "Data Structures", instructor = "Mr. Prakash", credits = 4, semester = "Sem 2";
INSERT INTO CourseInfo SET courseId = "C108", courseName = "Python Programming", instructor = "Mrs. Shivangi Patel", credits = 3, semester = "Sem 1";
INSERT INTO CourseInfo SET courseId = "C109", courseName = "Cloud Computing", instructor = "Dr. Nisha", credits = 4, semester = "Sem 5";
```

## DATABASE DESIGN AND IMPLEMENTATION

INSERT INTO CourseInfo SET courseId = "C110", courseName = "Cybersecurity", instructor = "Mr. Arav", credits = 3, semester = "Sem 6";

The screenshot shows a 'Graph Editor' window. At the top, it displays 'Nodes 10' and 'Edges 0'. Below this is a text area containing 11 SQL INSERT statements for a 'CourseInfo' table. The statements insert data for courses C101 through C110, including details like courseName, instructor, credits, and semester. Below the text area is a graph visualization showing a single node labeled 'CourseInfo' with a red circle icon. To the right of the node, there is a vertical stack of red circles, each labeled with a unique identifier (e.g., #23:1, #22:0, #24:0, #25:0, #26:0, #27:1).

Select \*FROM CourseInfo;

**Output:**

The screenshot shows a database query result for the query 'select \* FROM CourseInfo;'. The result is displayed in a table with two main sections: 'METADATA' and 'PROPERTIES'. The 'METADATA' section lists columns: @id, @version, @class, courseId, courseName, instructor, credits, and semester. The 'PROPERTIES' section shows the data for each row, with columns corresponding to the metadata. The table contains 10 rows of data, representing courses C101 through C110.

@id	@version	@class	courseId	courseName	instructor	credits	semester
#22:0	1	CourseInfo	C101	DBMS	Dr. Ravi	4	Sem 4
#22:1	1	CourseInfo	C105	Web Development	Ms. Shivangi	3	Sem 4
#22:2	1	CourseInfo	C109	Cloud Computing	Dr. Nisha	4	Sem 5
#23:0	1	CourseInfo	C102	OOP	Dr. Meena	3	Sem 3
#23:1	1	CourseInfo	C106	AI Fundamentals	Dr. Ramesh	4	Sem 6
#23:2	1	CourseInfo	C110	Cybersecurity	Mr. Arav	3	Sem 6
#24:0	1	CourseInfo	C103	Computer Networks	Dr. Arjun	4	Sem 5
#24:1	1	CourseInfo	C107	Data Structures	Mr. Prakash	4	Sem 2
#25:0	1	CourseInfo	C104	Software Engineering	Dr. Komal	3	Sem 6
#25:1	1	CourseInfo	C108	Python Programming	Mrs. Shivangi Patel	3	Sem 1

**Step14:Inserted Insertion Query:**

INSERT INTO CourseInfo SET courseId = "C101", courseName = "DBMS", instructor = "Dr. Ravi", credits = 4, semester = "Sem 4";

## DATABASE DESIGN AND IMPLEMENTATION

The screenshot shows the OrientDB web interface. At the top, there's a navigation bar with links: BROWSE, SCHEMA, SECURITY, GRAPH, FUNCTIONS, and DB. Below this, a SQL editor contains the following query:

```
1 INSERT INTO CourseInfo SET courseId = "C101", courseName = "DBMS", instructor = "Dr. Ravi", credits = 4, semester = "Sem 4";
```

Below the editor, a search bar and a 'Local Storage Size' indicator (100.31 KB) are visible. The 'COMMAND' section shows the executed query:

```
INSERT INTO CourseInfo SET courseId = "C101", courseName = "DBMS", instructor = "Dr. Ravi", credits = 4, semester = "Sem 4";
```

The result is displayed in a table with two main sections: METADATA and PROPERTIES.

METADATA			PROPERTIES		
@rid	@version	@class	courseId	courseName	instructor
#24.2	1	CourseInfo	C101	DBMS	Dr. Ravi

### Step15: Inserted Selection Query.

SELECT FROM CourseInfo WHERE courseId ="C104";

The screenshot shows the OrientDB web interface. The SQL editor contains the following query:

```
1 SELECT FROM CourseInfo WHERE courseId = "C104";
```

Below the editor, a search bar and a 'Local Storage Size' indicator (102.77 KB) are visible. The 'COMMAND' section shows the executed query:

```
SELECT FROM CourseInfo WHERE courseId = "C104";
```

The result is displayed in a table with two main sections: METADATA and PROPERTIES.

METADATA			PROPERTIES				
@rid	@version	@class	courseId	courseName	instructor	credits	semester
#25.0	1	CourseInfo	C104	Software Engineering	Dr. Komal	3	Sem 6

At the bottom, there's a status bar indicating 'Query executed in 0.026 sec. Returned 1 record(s). Limit: 20 (CHANGE IT)' and buttons for 'Table', 'Raw', and 'Explain'.

### Step16: Inserted Update Query1.

UPDATE CourseInfo SET instructor = "Dr.Komala" WHERE courseId = "C104";

The screenshot shows the OrientDB web interface. The SQL editor contains the following query:

```
1 UPDATE CourseInfo SET instructor = "Dr. Komala" WHERE courseId = "C104";
```

Below the editor, a search bar and a 'Local Storage Size' indicator (100.32 KB) are visible. The 'COMMAND' section shows the executed query:

```
UPDATE CourseInfo SET instructor = "Dr. Komala" WHERE courseId = "C104";
```

The result is displayed in a table with two main sections: METADATA and PROPERTIES.

METADATA			PROPERTIES				
@rid	@version	@class	courseId	courseName	instructor	credits	semester
#25.0	1	CourseInfo	C104	Software Engineering	Dr. Komal	3	Sem 6

At the bottom, there's a status bar indicating 'Query executed in 0.035 sec. Returned 1 record(s). Limit: 20 (CHANGE IT)' and buttons for 'Table', 'Raw', and 'Explain'.

### Step17: Inserted Update Query2.

UPDATE CourseInfo SET courseName = "Advanced Python Programming" WHERE courseID = "C108";

The screenshot shows a database query interface. At the top, a text area contains the SQL query: `1 UPDATE CourseInfo SET courseName = "Advanced Python Programming" WHERE courseId = "C108";`. Below the text area are buttons for `RUN` and `EXPLAIN`. A search bar labeled "Search in history" and a "Local Storage Size" indicator (909.8 KB) are also visible. Below the query text, a "COMMAND" section shows the same query. The "PROPERTIES" section displays a table with one row: 

count
1

. At the bottom, a status bar indicates "Query executed in 0.012 sec. Returned 1 record(s). Limit 20 (CHANGE IT)".

### Step18: Inserted Delete Query1.

DELETE VERTEX FROM CourseInfo WHERE courseID = "C110";

The screenshot shows a database query interface. At the top, a text area contains the SQL query: `1 DELETE VERTEX FROM CourseInfo WHERE courseId = "C110";`. Below the text area are buttons for `RUN` and `EXPLAIN`. A search bar labeled "Search in history" and a "Local Storage Size" indicator (111.33 KB) are also visible. Below the query text, a "COMMAND" section shows the same query. The "PROPERTIES" section displays a table with one row: 

count
1

. At the bottom, a status bar indicates "Query executed in 0.022 sec. Returned 1 record(s). Limit 20 (CHANGE IT)".

## Step19: Inserted Delete Query2.

DELETE VERTEX FROM CourseInfo WHERE Semester = "Sem 3";

The screenshot shows a database query interface. At the top, a text area contains the query: `1 DELETE VERTEX FROM CourseInfo WHERE semester = "Sem 3";`. Below the text area are buttons for `RUN` and `EXPLAIN`. A search bar and a local storage size indicator (112.15 KB) are also visible. The query is repeated in a command box. Below this, a table titled 'PROPERTIES' shows the result of the query. The table has two columns: 'count' and '1'. The 'count' column is highlighted. At the bottom, a status bar indicates 'Query executed in 0.014 sec. Returned 1 record(s). Limit: 20'. There are also buttons for 'Table' and 'Raw'.

PROPERTIES	
count	1

## Step20: Inserted Delete Query3.

DELETE VERTEX FROM CourseInfo;

The screenshot shows a database query interface. At the top, a text area contains the query: `1 DELETE VERTEX FROM CourseInfo;`. Below the text area are buttons for `RUN` and `EXPLAIN`. A search bar and a local storage size indicator (114.18 KB) are also visible. The query is repeated in a command box. Below this, a table titled 'PROPERTIES' shows the result of the query. The table has two columns: 'count' and '9'. The 'count' column is highlighted. At the bottom, a status bar indicates 'Query executed in 0.029 sec. Returned 1 record(s). Limit: 20'. There are also buttons for 'Table' and 'Raw'.

PROPERTIES	
count	9