| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| USN | | | | | | | | |

## Internal Assessment Test 2 – DEC 2023
## Scheme of Evaluation

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Sub: | **BIG DATA and ANALYTICS** | | | Sub Code: | **18CS72** | Branch: | **ISE** | |
| Date: | **04/12/2023** | Duration: | **90 min** | Max Marks: **50** | Sem/Sec: | **VII/ A, B & C** | | **OBE** |

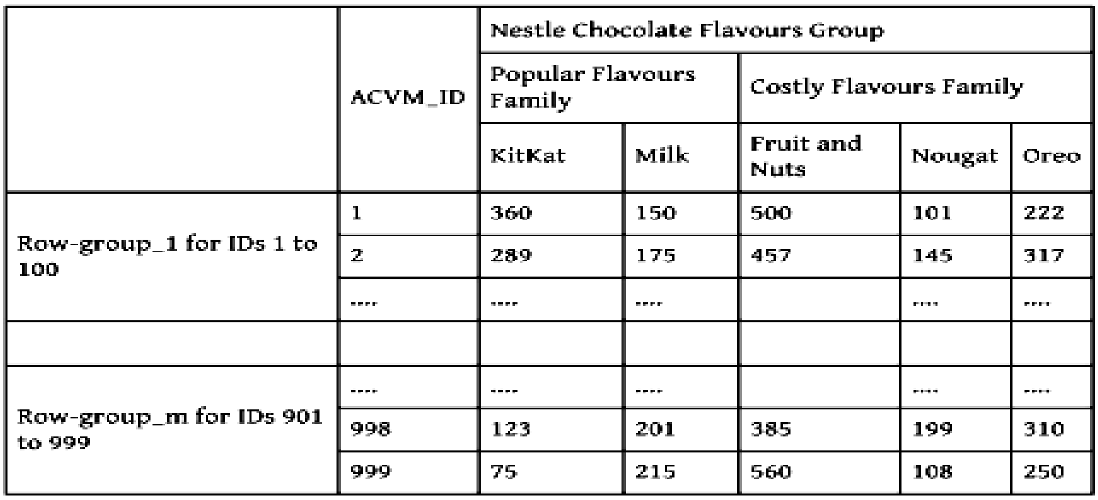
**Answer any FIVE FULL Questions** — MARKS | CO | RBT

1. Apply in-memory columnar storage for ACVM and Data for a large number of ACVMs with an ACVM_ID each, store in column 1. Data for each day sales at each ACVM for KitKat, Milk, Fruits & Nuts, Nougat and Oreo store in columns 2 to 6. Each row has six cells (ID +five sales data). [10] 4 L3

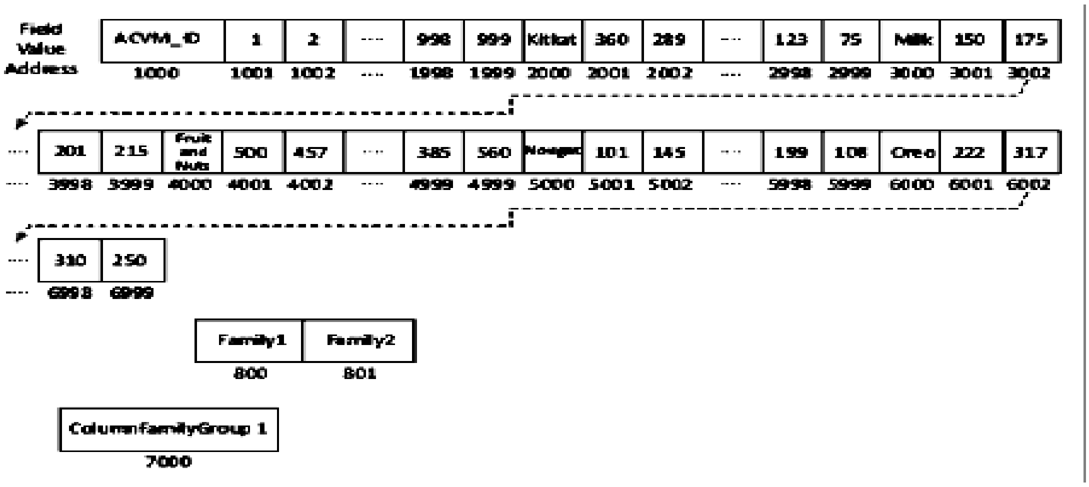
    a. How do the column key values store in memory?
    b. How do the values store in the memory in columnar storage format?
    c. How does analytics of each day's sales help?
    d. How are a column family and column-family head (key) specified?
    e. How do a column families group specify?
    f. f. How do row groups form?

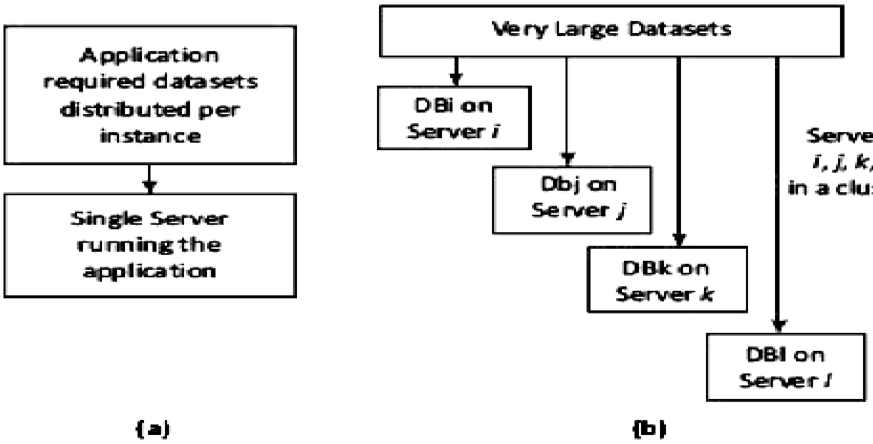
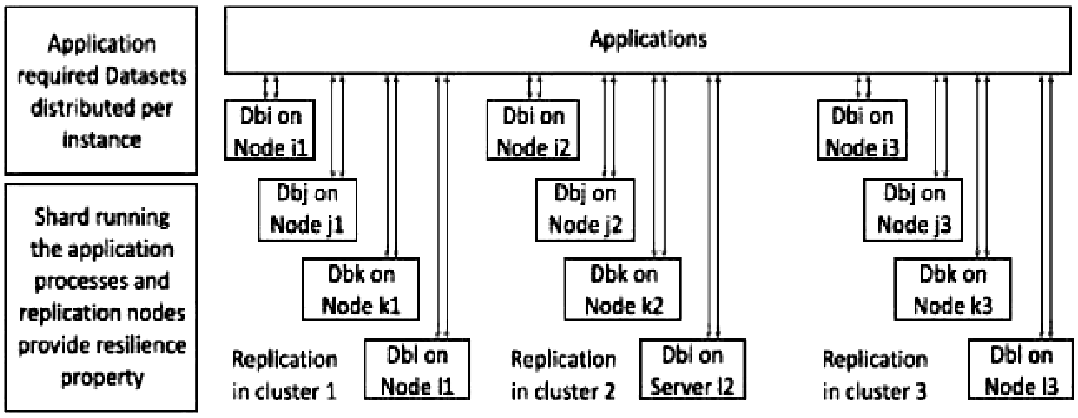
**Scheme:** Diagram+ Computation of each – 3+7 Marks.

**Solution:**



| | ACVM_ID | Nestle Chocolate Flavours Group | | | | |
|---|---|---|---|---|---|---|
| | | Popular Flavours Family | | Costly Flavours Family | | |
| | | KitKat | Milk | Fruit and Nuts | Nougat | Oreo |
| Row-group_1 for IDs 1 to 100 | 1 | 360 | 150 | 500 | 101 | 222 |
| | 2 | 289 | 175 | 457 | 145 | 317 |
| | .... | .... | .... | | .... | .... |
| | | | | | | |
| Row-group_m for IDs 901 to 999 | .... | .... | .... | | .... | .... |
| | 998 | 123 | 201 | 385 | 199 | 310 |
| | 999 | 75 | 215 | 560 | 108 | 250 |



| | | | | |
|---|---|---|---|---|
| 2. | **a. With a neat diagram, explain the following shared nothing architecture for big data tasks. i. Single server model ii. Sharding very large databases**<br>**Scheme :** Diagram+ Explanation – 3+3 Marks each<br>**Solution:**<br><br> Application required datasets distributed per instance<br>Single Server running the application<br>(a)<br>Very Large Datasets<br>DBi on Server i<br>Dbj on Server j<br>DBk on Server k<br>DBl on Server l<br>Servers i, j, k, l in a cluster<br>(b) | [6] | 4 | L2 |
| | **b. Explain Peer-to-peer distribution Model in handling big data.**<br>**Scheme :** Diagram+ Explanation – 2+2 Marks<br>**Solution:**<br><br> Application required Datasets distributed per instance<br>Shard running the application processes and replication nodes provide resilience property<br>Applications<br>Dbi on Node i1<br>Dbj on Node j1<br>Dbk on Node k1<br>Dbl on Node l1<br>Replication in cluster 1<br>Dbi on Node i2<br>Dbj on Node j2<br>Dbk on Node k2<br>Dbl on Server l2<br>Replication in cluster 2<br>Dbi on Node i3<br>Dbj on Node j3<br>Dbk on Node k3<br>Dbl on Node l3<br>Replication in cluster 3 | [4] | | |
| 3. | **Discuss Hadoop ecosystem tools with functionalities.**<br>**Scheme :** Explanation of any 5 tools – 2 Marks each<br>**Solution:**<br>• Zookeeper in Hadoop behaves as a centralized repository where distributed applications can write data at a node called JournalNode and read the data out of it.<br>• Apache Oozie is an open-source project of Apache that schedules Hadoop jobs.<br>• An efficient process for job handling is required.<br>• Analysis of Big Data requires creation of multiple jobs and sub-tasks in a process.<br>• The two basic Oozie functions are:<br>• Oozie workflow jobs are represented as Directed Acrylic Graphs (DAGs), specifying a sequence of actions to execute.<br>• Oozie coordinator jobs are recurrent Oozie workflow jobs that are triggered by | [10] | 2 | L2 |

time and data availability.

- Apache Sqoop is a tool that is built for loading efficiently the voluminous amount of data between Hadoop and external data repositories that resides on enterprise application servers or relational databases.
- Apache Flume provides a distributed, reliable and available service.
- Flume efficiently collects, aggregates and transfers a large amount of streaming data into HDFS.
- Flume enables upload of large files into Hadoop clusters.
- Ambari enables an enterprise to plan, securely install, manage clusters in the Hadoop.
- Apache Hive is an open-source data warehouse software.

4. **Explain the key value pairs and document store in NoSQL data architectural patterns with an example.** [10] 3 L2

**Scheme:** Key value pairs and Document Store explanation with examples – 5+5 Marks each.

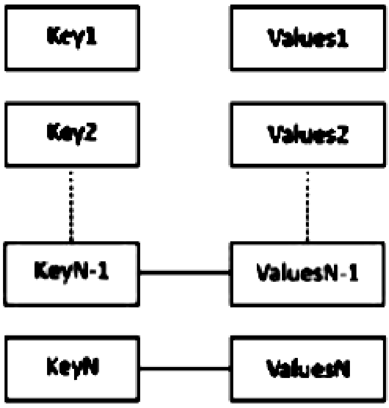
**Solution:**

The simplest way to implement a schema-less data store is to use key-value pairs.

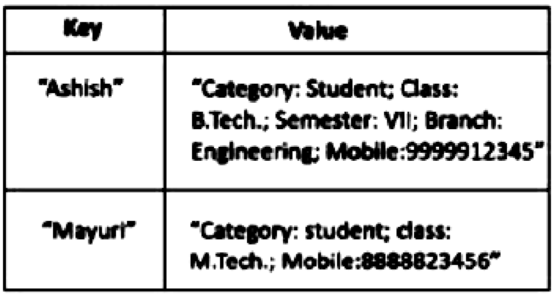
• The data store characteristics are high performance, scalability and flexibility.

• Data retrieval is fast in key-value pair’s data store.

• The concept is similar to a hash table where a unique key points to a particular item(s) of data.



Number of key-values pair, N can be a very large number

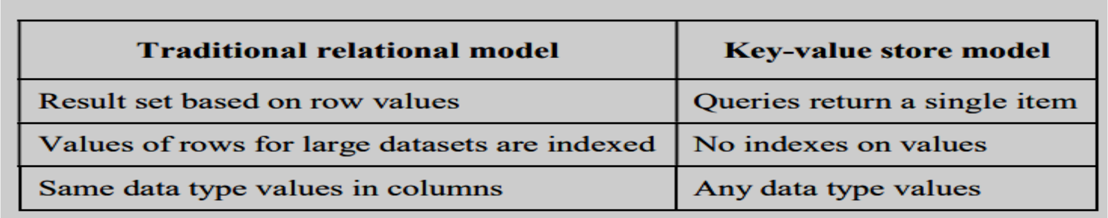


| Key | Value |
|---|---|
| "Ashish" | "Category: Student; Class: B.Tech.; Semester: VII; Branch: Engineering; Mobile:9999912345" |
| "Mayuri" | "Category: student; class: M.Tech.; Mobile:8888823456" |

**Advantages of a key-value store** 1. Data Store can store any data type in a value field. 2. A query just requests the values and returns the values as a single item. Values can be of any data type. 3. Key-value store is eventually consistent. 4. Key-value data store may be hierarchical or may be ordered key-value store. 5. Returned values on queries can be used to convert into lists, table-columns, data-frame fields and columns. 6. Have (i) scalability, (ii) reliability, (iii) portability and (iv) low operational cost. 7. The key can be synthetic or auto-generated. The key is flexible and can be represented in many formats: (i) Artificially generated strings created from a hash of a value, (ii) Logical path names to images or files

**The key-value store provides client to read and write values using a key as**

**follows:** (i) Get (key) , returns the value associated with the key. (ii) Put (key, value), associates the value with the key and updates a value if this key is already present. (iii) Multi-get (key1, key2, .., keyN), returns the list of values associated with the list of keys. (iv) Delete (key) , removes a key and its value from the data store.



| Traditional relational model | Key-value store model |
|---|---|
| Result set based on row values | Queries return a single item |
| Values of rows for large datasets are indexed | No indexes on values |
| Same data type values in columns | Any data type values |

**Features in Document Store:** 1. Document stores unstructured data. 2. Storage has similarity with object store. 3. Data stores in nested hierarchies. For example, in JSON formats data model XML document object model (DOM), or machine-readable data as one BLOB [Binary Large Object]. Hierarchical information stores in a single unit called document tree. Logical data stores together in a unit. 4. Querying is easy. For example, using section number, sub-section number and figure caption and table headings to retrieve document partitions. 5. Transactions on the document store exhibit ACID properties.

**Typical uses of a document store are:** (i) office documents, (ii) inventory store, (iii) forms data, (iv) document exchange

5. **Discuss the NoSQL data store characteristics and features in transactions with examples along with consistency, availability, and partition.** [10] 3 L2

**Scheme:** Characteristics+Features+explanation with examples+CAP Theorem – 2+3+3+2 Marks.

**Solution:**

NoSQL data store characteristics

1. NoSQL is a class of non-relational data storage system with flexible data model.
2. NoSQL not necessarily has a fixed schema.

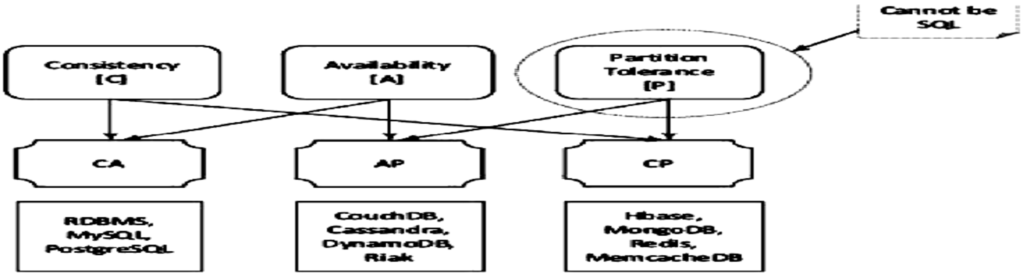
Features in NoSQL Transactions

(i) Relax one or more of the ACID properties.

(ii) Characterize by two out of three properties (consistency, availability and partitions) of CAP theorem, two are at least present for the application/service/process.

(iii) Can be characterized by BASE properties.

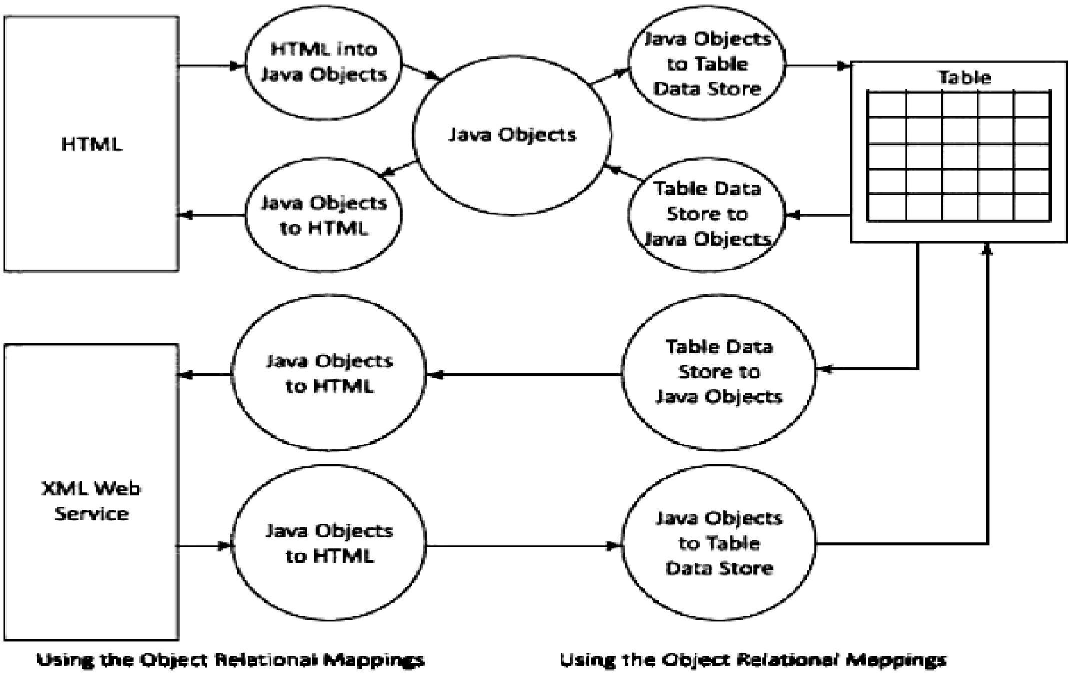
- Consistency means all copies have the same value like in traditional DBs.
- Availability means at least one copy is available in case a partition becomes inactive or fails.
- Partition means parts which are active but may not cooperate (share) as in distributed DBs.



6. **a. Apply Tabular data stores in handling HTML object and XML based web services.**

**Scheme :** Applying Tabular Data Stores for the given scenario with diagram–5 Marks

**Solution:**


HTML
HTML into Java Objects
Java Objects
Java Objects to Table Data Store
Table
Java Objects to HTML
Table Data Store to Java Objects
XML Web Service
Java Objects to HTML
Table Data Store to Java Objects
Java Objects to HTML
Java Objects to Table Data Store
Using the Object Relational Mappings
Using the Object Relational Mappings

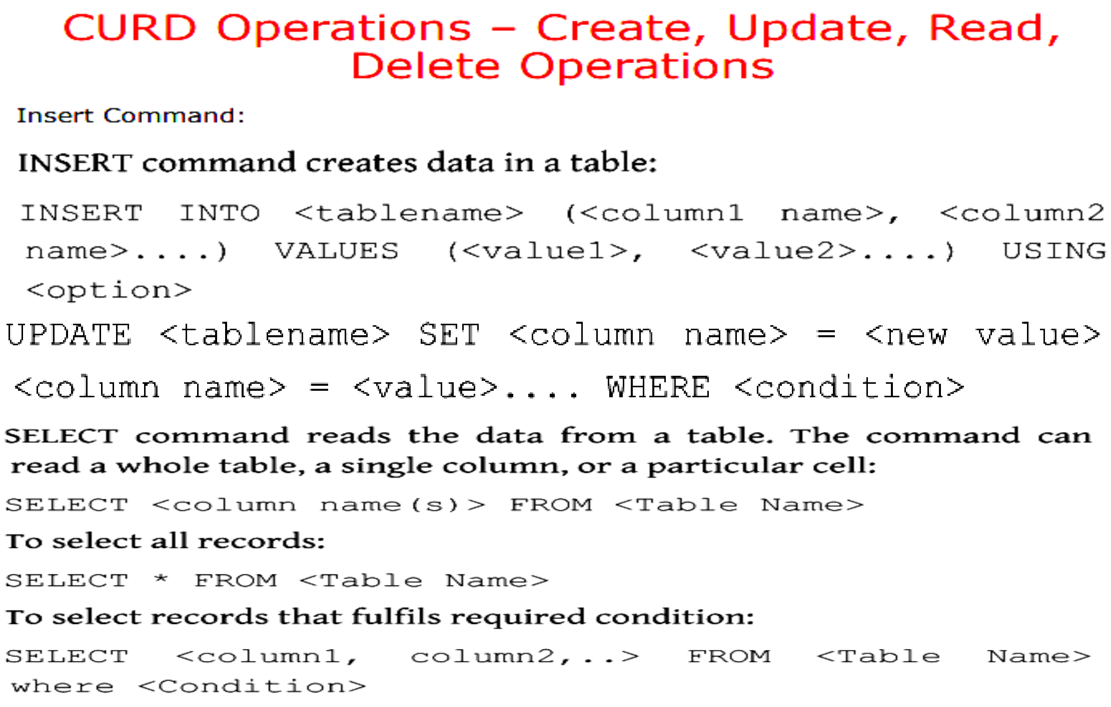
[5] 3 L3

**b. Solve queries in Cassandra for CURD operations by taking Product Info table by considering suitable columns and decide the partition key.**

[5]

**Scheme :** CURD Operations with queries –5 Marks

**Solution:**



## CURD Operations – Create, Update, Read, Delete Operations

Insert Command:

**INSERT command creates data in a table:**

```
INSERT INTO <tablename> (<column1 name>, <column2
name>....) VALUES (<value1>, <value2>....) USING
<option>
```

```
UPDATE <tablename> SET <column name> = <new value>
<column name> = <value>.... WHERE <condition>
```

**SELECT command reads the data from a table. The command can read a whole table, a single column, or a particular cell:**

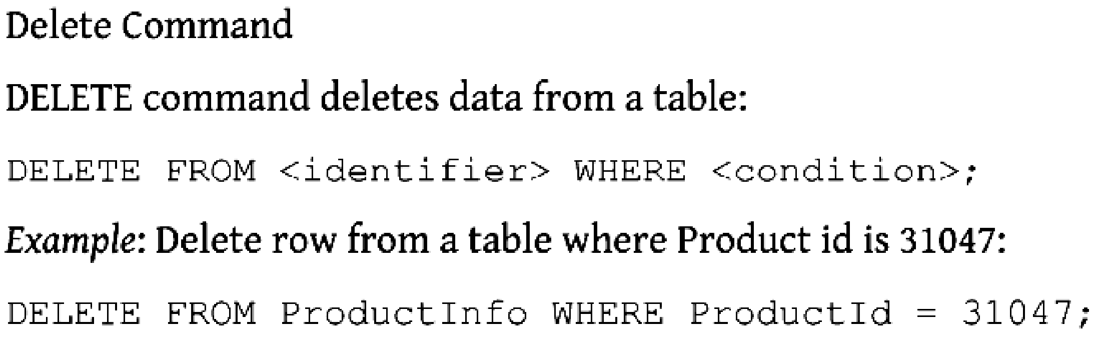
```
SELECT <column name(s)> FROM <Table Name>
```

**To select all records:**

```
SELECT * FROM <Table Name>
```

**To select records that fulfils required condition:**

```
SELECT <column1, column2,..> FROM <Table Name>
where <Condition>
```



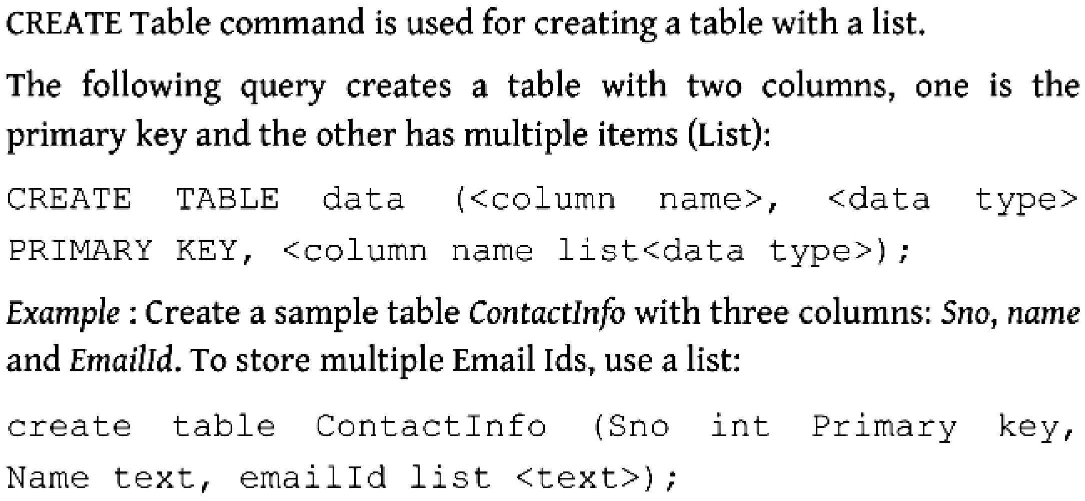
Delete Command

DELETE command deletes data from a table:

```
DELETE FROM <identifier> WHERE <condition>;
```

*Example*: Delete row from a table where Product id is 31047:

```
DELETE FROM ProductInfo WHERE ProductId = 31047;
```



CREATE Table command is used for creating a table with a list.

The following query creates a table with two columns, one is the primary key and the other has multiple items (List):

```
CREATE TABLE data (<column name>, <data type>
PRIMARY KEY, <column name list<data type>);
```

*Example* : Create a sample table *ContactInfo* with three columns: *Sno*, *name* and *EmailId*. To store multiple Email Ids, use a list:

```
create table ContactInfo (Sno int Primary key,
Name text, emailId list <text>);
```

Faculty Signature CCI Signature HOD Signature