Views, Indexes & Data Partitioning

🎯 Session Goals

- ✅ Understand Views and their practical uses
- ✅ Learn what Indexes are and how they speed up queries
- ✅ Grasp Data Partitioning and types: List, Range, and Hash

What is a View?

🔍 A View is a virtual table created by a query.
It doesn't store actual data – it simply shows data from one or more tables.

🎉 Syntax to Create a View

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2
FROM table_name
WHERE condition;
```

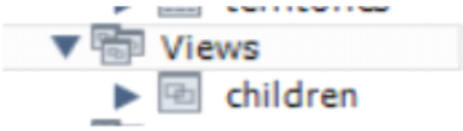
```
USE bike_analysis;
CREATE OR REPLACE VIEW `Children` AS
SELECT
    CustomerKey,
    FirstName,
    LastName,
    TotalChildren
FROM Customers
WHERE TotalChildren > 3;

SELECT * FROM Children;
```

CustomerKey	FirstName	LastName	TotalChildren
11004	ELIZABETH	JOHNSON	5
11008	ROBIN	VERHOFF	4
11011	CURTIS	LU	4
11017	SHANNON	WANG	4
11031	THERESA	RAMOS	4
11032	DENISE	STONE	4
11033	JAIME	NATH	4
11034	EBONY	GONZALEZ	4
11102	JULIA	NELSON	5
11113	MICHEAL	BLANCO	5
11114	LESLIE	MORENO	5
11115	ALVIN	CAI	5

```
615 •  SELECT COUNT(*) FROM Children;
616
617
```

Result Grid | Filter Rows: | Export: | Wrap C

COUNT(*)
418




```
CREATE OR REPLACE VIEW `SalesTrend` AS
SELECT
    DATE_FORMAT(s.OrderDate, "%Y-%m") AS year_and_month,
    ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
FROM `sales-2015` s
JOIN Products p
ON p.ProductKey = s.ProductKey
GROUP BY year_and_month
ORDER BY year_and_month;

SELECT * FROM SalesTrend;
```

year_and_month	TotalRevenue
2015-01	585313
2015-02	532226
2015-03	643436
2015-04	653364
2015-05	659326
2015-06	669989
2015-07	486115
2015-08	536453
2015-09	344063
2015-10	404277
2015-11	326611
2015-12	563762


```
SHOW FULL TABLES WHERE Table_type = 'VIEW'; -- command line
```

What is Indexing in SQL?

An index is a data structure used to speed up data retrieval.
Think of it as a book's index – you don't read every page to find a topic!

Syntax to Create an Index

```
CREATE INDEX index_name ON table_name (column_name);
```

```
CREATE TABLE student_info (
  studentid INT NOT NULL AUTO_INCREMENT,
  name VARCHAR(45),
  age VARCHAR(3),
  mobile VARCHAR(20),
  email VARCHAR(25),
  PRIMARY KEY (studentid),
  UNIQUE KEY email_UNIQUE (email)
);
```

```
CREATE TABLE Employee_Detail (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(45),
    Email VARCHAR(45),
    Phone VARCHAR(15),
    City VARCHAR(25),
    UNIQUE KEY unique_email (Email)
);
```

```
631 •  DESC Customers;
632 •  SHOW INDEXES FROM Customers;
633
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

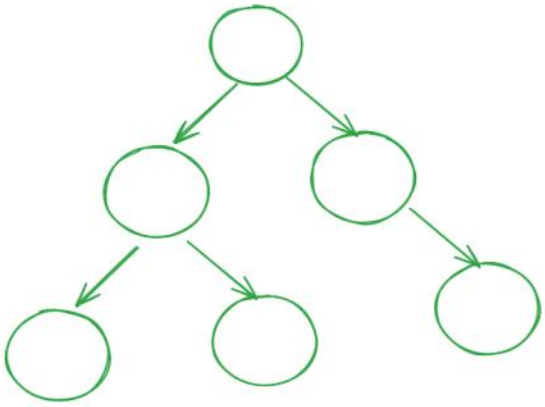
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression

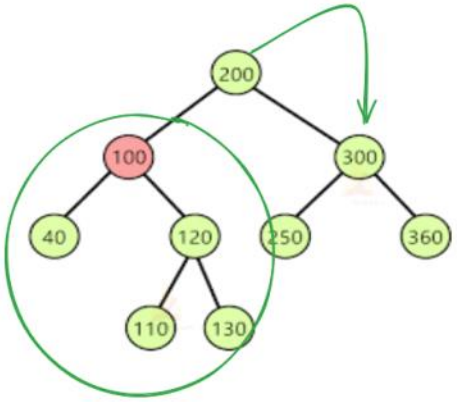
```
631 •  DESC Customers;
632 •  SHOW INDEXES FROM Customers;
633 •  CREATE INDEX idx_customerKey ON Customers(CustomerKey);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
customers	1	idx_customerKey	1	CustomerKey	A	2062	NULL	NULL	YES	BTREE			YES	NULL

Ascending





Field	Type	Null	Key	Default	Extra
CustomerKey	int	YES	MUL	NULL	
Prefix	text	YES		NULL	
FirstName	varchar...	YES		NULL	
LastName	varchar...	YES		NULL	
FullName	varchar...	YES		NULL	
DateOfBirth	date	YES		NULL	
Country	varchar...	YES		NULL	
MaritalStatus	text	YES		NULL	
Gender	text	YES		NULL	
EmailAddress	varchar...	YES		NULL	
AnnualIncome	int	YES		NULL	
IncomeCate...	varchar...	YES		NULL	
TotalChildren	int	YES		NULL	
EducationLevel	text	YES		NULL	
Occupation	text	YES		NULL	
HomeOwner	text	YES		NULL	
Phone_number	text	YES		NULL	

There is no limit to store information on 'Text Datatype'

```
639 •  SELECT DISTINCT Occupation FROM Customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Occupation
Professional
Management
Skilled Manual
Clerical
Manual

```
CREATE INDEX idx_occupation ON Customers(Occupation);
-- Error Code: 1170. BLOB/TEXT column 'Occupation' used in key
specification without a key length
```

```
CREATE INDEX idx_occupation ON Customers(Occupation(25));
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
customers	1	idx_customerKey	1	CustomerKey	A	2062	NULL	NULL	YES	BTREE
customers	1	idx_occupation	1	Occupation	A	5	25	NULL	YES	BTREE

Clustered Index - 1 Key - Primary Key
Non-Clustered Index - More than 1 -
Anything apart from Primary Key will fall in Non-Clustered
- Composite Index - (combination of non-clustered index)

```
CREATE TABLE student_info (
  studentid INT NOT NULL AUTO_INCREMENT,
  name VARCHAR(45),
  age VARCHAR(3),
  mobile VARCHAR(20),
  email VARCHAR(25),
  PRIMARY KEY (studentid),
  UNIQUE KEY email_UNIQUE (email)
);
```

Primary Key - Clustered Index

Non Clustered Index

COMPOSITE(email,phone_number) Non Clustered Index.

studentid	int	NO	PRI	NULL	auto_increment
name	varchar...	YES		NULL	
age	varchar(3)	YES		NULL	
mobile	varchar...	YES		NULL	
email	varchar...	YES		NULL	

```
SHOW INDEX FROM student_info;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment
student_info	0	PRIMARY	1	studentid	A	0	NULL	NULL		BTREE	

Field	Type	Null	Key	Default	Extra
studentid	int	NO	PRI	NULL	auto_increment
name	varchar...	YES		NULL	
age	varchar(3)	YES		NULL	
mobile	varchar...	YES		NULL	
email	varchar...	YES	UNI	NULL	

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
student_info	0	PRIMARY	1	studentid	A	0	NULL	NULL		BTREE
student_info	0	email_UNIQUE	1	email	A	0	NULL	NULL	YES	BTREE

```
DESC Employee_Detail;
```

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	auto_increment
Name	varchar...	YES		NULL	
Email	varchar...	YES	UNI	NULL	
Phone	varchar...	YES		NULL	
City	varchar...	YES		NULL	

COMPOSITE INDEX → unique_index(Email,Phone)

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality
employee_detail	0	PRIMARY	1	ID	A	0
employee_detail	0	unique_email	1	Email	A	0

```
672 •  CREATE UNIQUE INDEX index_email_phone
673    ON Employee_Detail(email,phone);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality
employee_detail	0	PRIMARY	1	ID	A	0
employee_detail	0	unique_email	1	Email	A	0
employee_detail	0	index_email_phone	1	Email	A	0
employee_detail	0	index_email_phone	2	Phone	A	0

```sql
CREATE TABLE Employee_Detail (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(45),
    Email VARCHAR(45),
    Phone VARCHAR(15),
    City VARCHAR(25),
    UNIQUE KEY unique_email (Email)
);
DESC Employee_Detail;
SHOW INDEX FROM Employee_Detail;

CREATE UNIQUE INDEX index_email_phone
ON Employee_Detail(email,phone);
```

composite index

```sql
SELECT * FROM Employee_Detail;

INSERT INTO Employee_Detail
VALUES(1,'Aditya','aditya@gmail.com','9999911111','Delhi');
INSERT INTO Employee_Detail
VALUES(9,'Utkarsh','utkarsh@gmail.com','9999811118','Navi Mumbai');


INSERT INTO Employee_Detail VALUES
(2, 'Jane Smith', 'jane.smith@example.com', '9123456789', 'Delhi'),
(3, 'Raj Kumar', 'raj.kumar@example.com', '9988776655', 'Bangalore'),
(4, 'Priya Sharma', 'priya.sharma@example.com', '8877665544', 'Chennai'),
(5, 'Amit Patel', 'amit.patel@example.com', '9090909090', 'Ahmedabad'),
(6, 'Sara Khan', 'sara.khan@example.com', '8787878787', 'Hyderabad'),
(7, 'Vikram Rao', 'vikram.rao@example.com', '7676767676', 'Pune'),
(8, 'Neha Jain', 'neha.jain@example.com', '9898989898', 'Kolkata');

SELECT * FROM Employee_Detail;

INSERT INTO Employee_Detail
VALUES(10,'Utkarsh','utkarsh1@gmail.com','9999811118','Navi Mumbai');

INSERT INTO Employee_Detail
VALUES(11,'Aditya','aditya@gmail.com','9999911112','Delhi');

DROP INDEX unique_email ON Employee_Detail;
```

📌 What is Data Partitioning?

Data Partitioning = Splitting a large table into smaller, logical chunks (called partitions) to improve performance and manageability.

🔍 Types of Partitioning

1. Range Partitioning
2. List Partitioning
3. Hash Partitioning

1 Range Partitioning

```sql
CREATE TABLE Sales (
  cust_id INT NOT NULL,
  name VARCHAR(40),
  store_id VARCHAR(20) NOT NULL,
  bill_no INT NOT NULL,
  bill_date DATE NOT NULL PRIMARY KEY,
  amount DECIMAL(8,2) NOT NULL
)
PARTITION BY RANGE (YEAR(bill_date)) (
  PARTITION p0 VALUES LESS THAN (2016),
  PARTITION p1 VALUES LESS THAN (2017),
  PARTITION p2 VALUES LESS THAN (2018),
  PARTITION p3 VALUES LESS THAN (2020)
);

INSERT INTO Sales VALUES
(1, 'Mike', 'S001', 101, '2015-01-02', 125.56),
(2, 'Robert', 'S003', 103, '2015-01-25', 476.50),
(3, 'Peter', 'S012', 122, '2016-02-15', 335.00),
(4, 'Joseph', 'S345', 121, '2016-03-26', 787.00),
(5, 'Harry', 'S234', 132, '2017-04-19', 678.00),
(6, 'Stephen', 'S743', 111, '2017-05-31', 864.00),
(7, 'Jacson', 'S234', 115, '2018-06-11', 762.00),
(8, 'Smith', 'S012', 125, '2019-07-24', 300.00),
(9, 'Adam', 'S456', 119, '2019-08-02', 492.20);


SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS
FROM INFORMATION_SCHEMA.PARTITIONS
WHERE TABLE_NAME = 'Sales';
```

TABLE_NAME	PARTITION_NAME	TABLE_ROWS
sales	p0	2
sales	p1	2
sales	p2	2
sales	p3	3

cust_id	name	store_id	bill_no	bill_date	amount	
1	Mike	S001	101	2015-01-02	125.56	p0
2	Robert	S003	103	2015-01-25	476.50	
3	Peter	S012	122	2016-02-15	335.00	p1
4	Joseph	S345	121	2016-03-26	787.00	
5	Harry	S234	132	2017-04-19	678.00	p2
6	Stephen	S743	111	2017-05-31	864.00	
7	Jacson	S234	115	2018-06-11	762.00	p3
8	Smith	S012	125	2019-07-24	300.00	
9	Adam	S456	119	2019-08-02	492.20	

```
INSERT INTO Sales VALUES
(10, 'Mikey', 'S001', 129, '2019-01-02', 129.59);
```

TABLE_NAME	PARTITION_NAME	TABLE_ROWS
sales	p0	2
sales	p1	2
sales	p2	2
sales	p3	4

2 List Partitioning

```
CREATE TABLE sales1 (
  sale_id INT,
  product_id INT,
  sale_date DATE,
  category VARCHAR(20),
  amount DECIMAL(10,2)
)
PARTITION BY LIST COLUMNS (category) (
  PARTITION p_electronics VALUES IN ('Electronics'),
  PARTITION p_clothing VALUES IN ('Clothing'),
  PARTITION p_furniture VALUES IN ('Furniture'),
  PARTITION p_books VALUES IN ('Books')
);


INSERT INTO sales1 (sale_id, product_id, sale_date, category, amount)
VALUES
(1, 101, '2024-01-01', 'Electronics', 199.99),
(2, 102, '2024-01-02', 'Clothing', 49.99),
(3, 103, '2024-01-03', 'Furniture', 299.99),
(4, 104, '2024-01-04', 'Books', 19.99),
(5, 105, '2024-01-05', 'Electronics', 499.99),
(6, 106, '2024-01-06', 'Clothing', 89.99),
(7, 107, '2024-01-07', 'Furniture', 1299.99),
(8, 108, '2024-01-08', 'Books', 9.99),
(9, 109, '2024-01-09', 'Electronics', 299.99),
(10, 110, '2024-01-10', 'Clothing', 59.99),
(11, 111, '2024-01-11', 'Furniture', 799.99),
(12, 112, '2024-01-12', 'Books', 14.99),
(13, 113, '2024-01-13', 'Electronics', 399.99),
(14, 114, '2024-01-14', 'Clothing', 109.99),
(15, 115, '2024-01-15', 'Furniture', 499.99),
(16, 116, '2024-01-16', 'Books', 24.99),
(17, 117, '2024-01-17', 'Electronics', 599.99),
(18, 118, '2024-01-18', 'Clothing', 79.99),
(19, 119, '2024-01-19', 'Furniture', 699.99),
(20, 120, '2024-01-20', 'Books', 29.99);
```

sale_id	product_id	sale_date	category	amount
1	101	2024-01-01	Electronics	199.99
5	105	2024-01-05	Electronics	499.99
9	109	2024-01-09	Electronics	299.99
13	113	2024-01-13	Electronics	399.99
17	117	2024-01-17	Electronics	599.99
2	102	2024-01-02	Clothing	49.99
6	106	2024-01-06	Clothing	89.99
10	110	2024-01-10	Clothing	59.99
14	114	2024-01-14	Clothing	109.99
18	118	2024-01-18	Clothing	79.99
3	103	2024-01-03	Furniture	299.99
7	107	2024-01-07	Furniture	1299.99
11	111	2024-01-11	Furniture	799.99
15	115	2024-01-15	Furniture	499.99
19	119	2024-01-19	Furniture	699.99
4	104	2024-01-04	Books	19.99
8	108	2024-01-08	Books	9.99
12	112	2024-01-12	Books	14.99

TABLE_NAME	PARTITION_NAME	TABLE_ROWS
sales1	p_books	5
sales1	p_clothing	5
sales1	p_electronics	5
sales1	p_furniture	5

```
INSERT INTO sales1 (sale_id, product_id, sale_date, category, amount)
VALUES (21, 121, '2025-01-01', 'Electronics', 197.99);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TABLE_NAME	PARTITION_NAME	TABLE_ROWS
sales1	p_books	5
sales1	p_clothing	5
sales1	p_electronics	6
sales1	p_furniture	5

3 Hash Partitioning

```
CREATE TABLE Stores (
  cust_name VARCHAR(40),
  bill_no VARCHAR(20) NOT NULL,
  store_id INT PRIMARY KEY NOT NULL,
  bill_date DATE NOT NULL,
  amount DECIMAL(8,2) NOT NULL
)
PARTITION BY HASH(store_id)
PARTITIONS 4;

INSERT INTO Stores (cust_name, bill_no, store_id, bill_date, amount) VALUES
('Alice', 'B001', 1, '2024-01-01', 150.75),
('Bob', 'B002', 2, '2024-01-02', 200.00),
('Charlie', 'B003', 3, '2024-01-03', 99.99),
('David', 'B004', 4, '2024-01-04', 175.50),
('Eva', 'B005', 5, '2024-01-05', 250.00),
('Frank', 'B006', 6, '2024-01-06', 300.75),
('Grace', 'B007', 7, '2024-01-07', 80.25),
('Hannah', 'B008', 8, '2024-01-08', 120.50),
('Ivan', 'B009', 9, '2024-01-09', 450.00),
('Jack', 'B010', 10, '2024-01-10', 60.00),
('Karen', 'B011', 11, '2024-01-11', 110.75),
('Leo', 'B012', 12, '2024-01-12', 220.00),
('Mia', 'B013', 13, '2024-01-13', 330.50),
('Nathan', 'B014', 14, '2024-01-14', 55.00),
('Olivia', 'B015', 15, '2024-01-15', 95.25),
('Paul', 'B016', 16, '2024-01-16', 500.00);
```

```sql
SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS
FROM INFORMATION_SCHEMA.PARTITIONS
WHERE TABLE_NAME = 'Stores';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TABLE_NAME	PARTITION_NAME	TABLE_ROWS
stores	p0	4
stores	p1	5
stores	p2	4
stores	p3	4