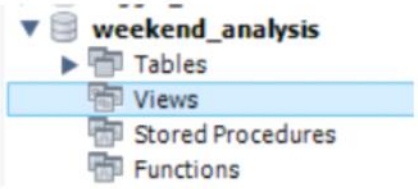
Views, Indexes & Data Partitioning

🎯 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;
```

```
CREATE OR REPLACE VIEW Children AS
SELECT
    CustomerKey,
    FirstName,
    LastName,
    MaritalStatus,
    EmailAddress,
    Gender,
    AnnualIncome,
    TotalChildren
FROM Customers
WHERE TotalChildren > 3; -- 418 row(s) returned
```

```
543 •  SELECT * FROM Children;
544
```

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

CustomerKey	FirstName	LastName	MaritalStatus	EmailAddress	Gender	AnnualIncome	TotalChildren
11004	ELIZABETH	JOHNSON	S	elizabeth5@learnsector.com	F	80000	5
11008	ROBIN	VERHOFF	S	rob4@learnsector.com	F	60000	4
11011	CURTIS	LU	M	curtis9@learnsector.com	M	60000	4
11017	SHANNON	WANG	S	shannon1@learnsector.com	F	20000	4
11031	THERESA	RAMOS	M	theresa13@learnsector.com	F	20000	4
11032	DENISE	STONE	M	denise10@learnsector.com	F	20000	4
11033	JAIME	NATH	M	jaime41@learnsector.com	M	20000	4
11034	EBONY	GONZALEZ	M	ebony19@learnsector.com	F	20000	4
11102	JULIA	NELSON	S	julia7@learnsector.com	F	80000	5
11113	MICHEAL	BLANCO	M	micheal11@learnsector.com	M	70000	5
11114	LESLIE	MORENO	S	leslie7@learnsector.com	F	70000	5
11115	ALVIN	CAI	M	alvin20@learnsector.com	M	70000	5
11116	CLINTON	CARLSON	M	clinton14@learnsector.com	M	70000	5
11117	APRIL	DENG	M	april1@learnsector.com	F	70000	5

```sql
CREATE OR REPLACE VIEW AllSales AS
SELECT * FROM Sales2015
UNION ALL
SELECT * FROM Sales2016
UNION ALL
SELECT * FROM Sales2017;

SELECT * FROM AllSales; -- 56046 row(s) returned
```

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

OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity
2015-01-01	2001-09-21	SO45080	332	14657	1	1	1
2015-01-01	2001-12-05	SO45079	312	29255	4	1	1
2015-01-01	2001-10-29	SO45082	350	11455	9	1	1
2015-01-01	2001-11-16	SO45081	338	26782	6	1	1
2015-01-02	2001-12-15	SO45083	312	14947	10	1	1
2015-01-02	2001-10-12	SO45084	310	29143	4	1	1
2015-01-02	2001-12-18	SO45086	314	18747	9	1	1
2015-01-02	2001-10-09	SO45085	312	18746	9	1	1
2015-01-03	2001-10-03	SO45093	312	18906	9	1	1
2015-01-03	2001-09-29	SO45090	310	29170	4	1	1
2015-01-03	2001-12-11	SO45088	345	11398	10	1	1
2015-01-03	2001-10-24	SO45092	313	18899	9	1	1
2015-01-03	2001-12-16	SO45089	351	25977	4	1	1
2015-01-03	2001-10-26	SO45091	314	18909	9	1	1


```sql
CREATE OR REPLACE VIEW Trend_Analysis AS
WITH Understanding_Revenue AS (
    SELECT
        DATE_FORMAT(s.OrderDate , '%Y-%m') AS YearMonth,
        ROUND(SUM(p.ProductPrice * s.OrderQuantity),0) AS TotalRevenue
    FROM AllSales s
    JOIN Products p
    ON p.ProductKey = s.ProductKey
    GROUP BY 1
    ORDER BY 1
)
SELECT
    *,
    LAG(TotalRevenue) OVER(ORDER BY YearMonth) AS PreviousMonthRevenue,
    LEAD(TotalRevenue) OVER(ORDER BY YearMonth) AS NextMonthRevenue,
    TotalRevenue - LAG(TotalRevenue) OVER(ORDER BY YearMonth) AS SaleAmountChange,
    CASE
        WHEN TotalRevenue > LAG(TotalRevenue) OVER(ORDER BY YearMonth) THEN 'Increase'
        WHEN TotalRevenue < LAG(TotalRevenue) OVER(ORDER BY YearMonth) THEN 'Decrease'
        ELSE 'No Change'
    END AS SalesTrend
FROM Understanding_Revenue;

SELECT * FROM Trend_Analysis;
```

YearMonth	TotalRevenue	PreviousMonthRevenue	NextMonthRevenue	SaleAmountChange	SalesTrend
2015-01	585313	NULL	532226	NULL	No Change
2015-02	532226	585313	643436	-53087	Decrease
2015-03	643436	532226	653364	111210	Increase
2015-04	653364	643436	659326	9928	Increase
2015-05	659326	653364	669989	5962	Increase
2015-06	669989	659326	486115	10663	Increase
2015-07	486115	669989	536453	-183874	Decrease
2015-08	536453	486115	344063	50338	Increase
2015-09	344063	536453	404277	-192390	Decrease
2015-10	404277	344063	326611	60214	Increase
2015-11	326611	404277	563762	-77666	Decrease
2015-12	563762	326611	432426	237151	Increase
2016-01	432426	563762	474163	-131336	Decrease
2016-02	474163	432426	471962	41737	Increase
2016-03	471962	474163	494957	-2201	Decrease
2016-04	494957	471962	545535	22995	Increase
2016-05	545535	494957	533825	50578	Increase
2016-06	533825	545535	815356	-11710	Decrease
2016-07	815356	533825	804193	281531	Increase
2016-08	804193	815356	952743	-11163	Decrease
2016-09	952743	804193	1029821	148550	Increase

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);
```

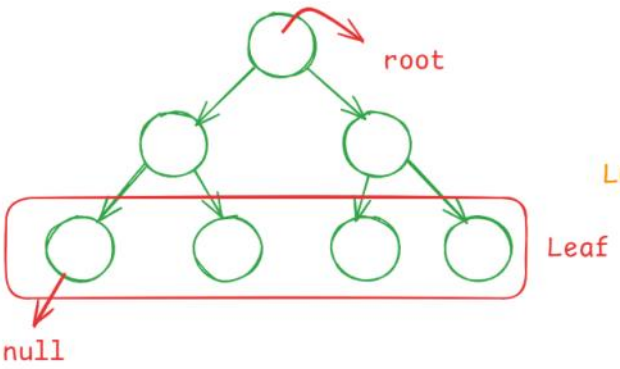
```
CREATE TABLE student_info (
  studentid INT NOT NULL AUTO_INCREMENT,
  name VARCHAR(45),
  age VARCHAR(3),
  mobile VARCHAR(20),
  email VARCHAR(25),
  PRIMARY KEY (studentid),
  UNIQUE KEY email_UNIQUE (email)
);
```

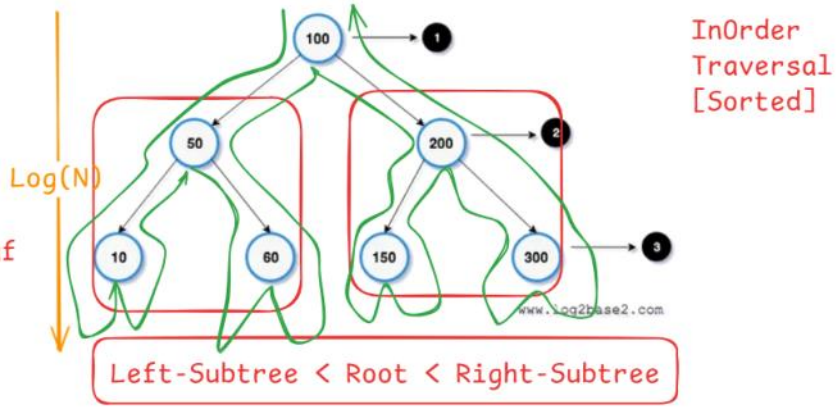
```
CREATE TABLE Employee_Detail (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(45),
    Email VARCHAR(45),
    Phone VARCHAR(15),
    City VARCHAR(25),
    UNIQUE KEY unique_email (Email)
);
```

```
580    -- ================================ INDEX ============================
581 •  DESC Customers;
582 •  SHOW INDEXES FROM Customers;
583
```

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	0	PRIMARY	1	CustomerKey	A	2124	NULL	NULL		BTREE			YES	NULL





Field	Type	Null	Key	Default	Extra
CustomerKey	int	NO	PRI	NULL	
Prefix	text	YES		NULL	
FirstName	varchar(50)	YES		NULL	
LastName	varchar(50)	YES		NULL	
FullName	varchar(100)	YES		NULL	
DateOfBirth	date	YES		NULL	
MaritalStatus	text	YES		NULL	
EmailAddress	varchar(100)	YES		NULL	
Gender	text	YES		NULL	

```
590 •  CREATE UNIQUE INDEX idx_email
591    ON Customers(EmailAddress);
592 •  SHOW INDEXES FROM Customers;
593
```

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	0	PRIMARY	1	CustomerKey	A	2124	NULL	NULL		BTREE			YES	NULL
customers	0	idx_email	1	EmailAddress	A	2062	NULL	NULL	YES	BTREE			YES	NULL

```
-- ================================ INDEX ============================
DESC Customers;
SHOW INDEXES FROM Customers;

SELECT * FROM Customers; --  13057 & destiny57@learnsector.com

                                                          Before
SELECT * FROM Customers WHERE EmailAddress = 'destiny57@learnsector.com'; -- 0.016 sec / 0.000 sec

SELECT * FROM Customers WHERE CustomerKey = 13057; -- 0.000 sec / 0.000 sec

CREATE UNIQUE INDEX idx_email
ON Customers(EmailAddress);
SHOW INDEXES FROM Customers;

                                                          After
SELECT * FROM Customers WHERE EmailAddress = 'destiny57@learnsector.com'; -- 0.000 sec / 0.000 sec
```

```sql
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), -- Clustered Index
    UNIQUE KEY email_UNIQUE (email) -- Non-Clustured Index
);

DESC student_info;
```

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
student_info	0	PRIMARY	1	studentid	A	0	NULL	NULL		BTREE
student_info	0	email_UNIQUE	1	email	A	0	NULL	NULL	YES	BTREE

```sql
DESC student_info;

SHOW INDEXES FROM Student_info;
```

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

Field	Type	Null	Key	Default	Extra
studentid	int	NO	PRI	NULL	auto_increment
name	varchar(45)	YES		NULL	
age	varchar(3)	YES		NULL	
mobile	varchar(20)	YES		NULL	
email	varchar(25)	YES	UNI	NULL	

```sql
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)
);
```

Clustered Index

Non-Clustured Index

COMPOSITE(Email , Mobile) Non-Clustured Index

```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;
```

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

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	auto_increment
Name	varchar(45)	YES		NULL	
Email	varchar(45)	YES	UNI	NULL	
Phone	varchar(15)	YES		NULL	
City	varchar(25)	YES		NULL	

```
622 •  SHOW INDEXES FROM Employee_Detail;
```

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
employee_detail	0	PRIMARY	1	ID	A	0	NULL	NULL		BTREE
employee_detail	0	unique_email	1	Email	A	0	NULL	NULL	YES	BTREE

```
624 •  CREATE UNIQUE INDEX idx_email_phone
625    ON Employee_Detail(Email,Phone);
626
627
```

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
employee_detail	0	PRIMARY	1	ID	A	0	NULL	NULL		BTREE
employee_detail	0	unique_email	1	Email	A	0	NULL	NULL	YES	BTREE
employee_detail	0	idx_email_phone	1	Email	A	0	NULL	NULL	YES	BTREE
employee_detail	0	idx_email_phone	2	Phone	A	0	NULL	NULL	YES	BTREE

```
DESC Employee_Detail;

SHOW INDEXES FROM Employee_Detail;

CREATE UNIQUE INDEX idx_email_phone
ON Employee_Detail(Email,Phone);


INSERT INTO Employee_Detail (Name, Email, Phone, City)
VALUES
('Amit Sharma', 'amit.sharma@example.com', '9876543210', 'Delhi'),
('Priya Mehta', 'priya.mehta@example.com', '9812345678', 'Mumbai'),
('Ravi Kumar', 'ravi.kumar@example.com', '9823456789', 'Bangalore'),
('Sneha Patel', 'sneha.patel@example.com', '9898989898', 'Ahmedabad'),
('Rahul Verma', 'rahul.verma@example.com', '9900112233', 'Chennai'),
('Neha Singh', 'neha.singh@example.com', '9876123456', 'Kolkata'),
('Vikas Gupta', 'vikas.gupta@example.com', '9988776655', 'Pune'),
('Anjali Nair', 'anjali.nair@example.com', '9765432189', 'Hyderabad'),
('Suresh Reddy', 'suresh.reddy@example.com', '9123456780', 'Hyderabad'),
('Meena Joshi', 'meena.joshi@example.com', '9345678901', 'Jaipur');


SELECT * FROM Employee_Detail;
INSERT INTO Employee_Detail (Name, Email, Phone, City)
VALUES('Amit Sharma', 'amit123.sharma@example.com', '9876543210', 'Delhi');

INSERT INTO Employee_Detail (Name, Email, Phone, City)
VALUES('Amit Sharma', 'amit.sharma@example.com', '9876543222', 'Delhi');
-- Error Code: 1062. Duplicate entry 'amit.sharma@example.com' for key
'employee_detail.unique_email
```

```
650    -- DROP THE unique_email INDEX
651 •  DROP INDEX unique_email ON Employee_Detail;
652 •  SHOW INDEXES FROM Employee_Detail;
```

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
employee_detail	0	PRIMARY	1	ID	A	0	NULL	NULL		BTREE
employee_detail	0	idx_email_phone	1	Email	A	0	NULL	NULL	YES	BTREE
employee_detail	0	idx_email_phone	2	Phone	A	0	NULL	NULL	YES	BTREE

```
654 •  SELECT * FROM Employee_Detail;
```

Result Grid | Filter Rows: | Edit: | Export/Import

ID	Name	Email	Phone	City
1	Amit Sharma	amit.sharma@example.com	9876543210	Delhi
2	Priya Mehta	priya.mehta@example.com	9812345678	Mumbai
3	Ravi Kumar	ravi.kumar@example.com	9823456789	Bangalore
4	Sneha Patel	sneha.patel@example.com	9898989898	Ahmedabad
5	Rahul Verma	rahul.verma@example.com	9900112233	Chennai
6	Neha Singh	neha.singh@example.com	9876123456	Kolkata
7	Vikas Gupta	vikas.gupta@example.com	9988776655	Pune
8	Anjali Nair	anjali.nair@example.com	9765432189	Hyderabad
9	Suresh Reddy	suresh.reddy@example.com	9123456780	Hyderabad
10	Meena Joshi	meena.joshi@example.com	9345678901	Jaipur
11	Amit Sharma	amit123.sharma@example....	9876543210	Delhi
NULL	NULL	NULL	NULL	NULL

```
CREATE UNIQUE INDEX idx_email_phone
ON Employee_Detail(Email,Phone); -- Composite Non-Clustered Index
```

```
INSERT INTO Employee_Detail (Name, Email, Phone, City)
VALUES ('Suresh Reddy', 'suresh.reddy123@example.com', '9123456780', 'Hyderabad');

INSERT INTO Employee_Detail (Name, Email, Phone, City)
VALUES ('Suresh Reddy', 'suresh.reddy123@example.com', '9123456781', 'Hyderabad');
-- Error Code: 1062. Duplicate entry 'suresh.reddy123@example.com-9123456781' for key
'employee_detail.idx_email_phone'
```

Special Note: In Composite Non-Clustered Index either columns should be unique to insert the data into the table. If both columns are non-unique then insertion will throw an error.

```
664 •  INSERT INTO Employee_Detail (Name, Email, Phone, City)
665    VALUES ('Suresh Reddy', 'suresh.reddy007@example.com', '9123499999', 'Hyderabad');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ID	Name	Email	Phone	City
5	Rahul Verma	rahul.verma@example.com	9900112233	Chennai
6	Neha Singh	neha.singh@example.com	9876123456	Kolkata
7	Vikas Gupta	vikas.gupta@example.com	9988776655	Pune
8	Anjali Nair	anjali.nair@example.com	9765432189	Hyderabad
9	Suresh Reddy	suresh.reddy@example.com	9123456780	Hyderabad
10	Meena Joshi	meena.joshi@example.com	9345678901	Jaipur
11	Amit Sharma	amit123.sharma@example....	9876543210	Delhi
13	Suresh Reddy	suresh.reddy123@example...	9123456780	Hyderabad
14	Suresh Reddy	suresh.reddy123@example...	9123456781	Hyderabad
16	Suresh Reddy	suresh.reddy007@example...	9123499999	Hyderabad
NULL	NULL	NULL	NULL	NULL

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

```
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	NULL	5035
sales	p0	2
sales	p1	2
sales	p2	2
sales	p3	3

```
PARTITION p0 VALUES LESS THAN (2016),
PARTITION p1 VALUES LESS THAN (2017),
PARTITION p2 VALUES LESS THAN (2018),
PARTITION p3 VALUES LESS THAN (2020)
```

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

```
INSERT INTO Sales VALUES
(10, 'Omkar', 'S007', 199, '2016-01-22', 125.56),
(11, 'Deepak', 'S013', 122, '2015-11-29', 476.50),
(12, 'Paramjeet', 'S099', 121, '2016-02-25', 335.00);
```

sales	p0	3
sales	p1	4
sales	p2	2
sales	p3	3

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);
```

```sql
SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS
FROM INFORMATION_SCHEMA.PARTITIONS
WHERE TABLE_NAME = 'Sales1';
```

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	5
sales1	p_furniture	5

```sql
INSERT INTO sales1 (sale_id, product_id, sale_date, category, amount)
VALUES
(21, 101, '2024-01-01', 'Electronics', 199.99),
(22, 102, '2024-01-02', 'Clothing', 49.99),
(23, 103, '2024-01-03', 'Furniture', 299.99),
(24, 104, '2024-01-04', 'Books', 19.99),
(25, 105, '2024-01-05', 'Electronics', 499.99);
```

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

TABLE_NAME	PARTITION_NAME	TABLE_ROWS
sales1	p_books	6
sales1	p_clothing	6
sales1	p_electronics	7
sales1	p_furniture	6

3 Hash Partitioning

```sql
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);
```

```
795 • INSERT INTO Stores (cust_name, bill_no, store_id, bill_date, amount) VALUES
796   ('Rajat', 'B017', 17, '2024-01-17', 150.75),
797   ('Shyam', 'B018', 18, '2024-01-18', 200.00);
798
```

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

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

```
799 • INSERT INTO Stores (cust_name, bill_no, store_id, bill_date, amount) VALUES
800   ('Omkar', 'B019', 19, '2024-01-19', 150.75);
801
```

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

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

```
802 • INSERT INTO Stores (cust_name, bill_no, store_id, bill_date, amount) VALUES
803   ('Naveen', 'B020', 20, '2024-01-20', 999.99);
804
```

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

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