KC Report Task Document:-

**Project**: Sales Reporting System (Spiral Framework)  
**Date**: June 20, 2025  
**Author**: Saroj Kumar Singh

# Monthly sales by region :-

Initial query:-

SELECT

YEAR(order\_date) AS year,

MONTH(order\_date) AS month,

s.region\_id,

SUM(o.unit\_price \* o.quantity) AS total\_sales,

COUNT(\*) AS number\_of\_orders

FROM orders o

JOIN stores s ON o.store\_storeId = s.store\_id

WHERE o.order\_date BETWEEN ? AND ?

GROUP BY YEAR(o.order\_date), MONTH(o.order\_date), s.region\_id

ORDER BY year, month, s.region\_id;

## **Performance issue / Slow execution time in above query due to:-**

* No computed/generated columns
* No compound index on order\_date and store\_storeId
* Runtime YEAR()/MONTH() function calls blocking index usage

**Schema Optimization done:-**

Added computed columns and indexes as below-

ALTER TABLE orders

ADD COLUMN order\_year INT GENERATED ALWAYS AS (YEAR(order\_date)) STORED,

ADD COLUMN order\_month INT GENERATED ALWAYS AS (MONTH(order\_date)) STORED;

ALTER TABLE `orders`

ADD INDEX `orders\_idx\_order\_year\_month\_storeId` (`order\_year`, `order\_month`, `store\_storeId`);

ALTER TABLE `orders`

ADD INDEX `orders\_index\_order\_date` (`order\_date`),

ADD INDEX `orders\_index\_storeID\_orderDate` (`store\_storeId`, `order\_date`),

ADD INDEX `orders\_index\_productId\_orderDate` (`product\_productId`, `order\_date`);

**Optimised Query:-**

SELECT

o.order\_year,

o.order\_month,

MONTHNAME(o.order\_date) AS `orderMonth`,

s.region\_id,

SUM(o.unit\_price \* o.quantity) AS total\_sales,

COUNT(\*) AS number\_of\_orders

FROM orders o

LEFT JOIN stores s ON o.store\_storeId = s.store\_id

WHERE o.order\_date >= ? AND o.order\_date <= ?

GROUP BY o.order\_year, o.order\_month, s.region\_id

ORDER BY o.order\_year, o.order\_month, s.region\_id

**Performance Comparison:-**

|  |  |  |
| --- | --- | --- |
| **Metric** | **Before Optimization** | **After Optimization** |
| Avg Execution Time | ~1450 ms | **~460 ms** |
| Index Used | None | orders\_idx\_order\_year\_month\_storeId  orders\_index\_order\_date  PRIMARY,stores\_region\_id |
| Query Plan | Full Table Scan | Index Scan |
| Meets > 1 sec | No | Yes |

# Top categories by store (3 months Range) :-

Initial Query -

SELECT

o.store\_id,

p.category\_id,

SUM(o.unit\_price \* o.quantity) AS total\_sales

FROM orders o

JOIN products p ON o.product\_productId = p.product\_id

WHERE order\_date BETWEEB ? AND ?

GROUP BY o.store\_id, p.category\_id

ORDER BY o.store\_id, total\_sales DESC;

## **Performance issue / Slow execution time in above query due to:-**

* Lacked compound index on store\_storeId, order\_date
* Full scan of orders table even when limited by date
* Ranking logic was missing and added later.

## **Schema Optimization done:-**

Added computed columns and indexes as below-

ALTER TABLE `stores`

ADD INDEX `stores\_region\_id` (`store\_id`, `region\_id`);

ALTER TABLE `orders`

ADD INDEX `orders\_index\_order\_date` (`order\_date`),

ADD INDEX `orders\_index\_storeID\_orderDate` (`store\_storeId`, `order\_date`),

ADD INDEX `orders\_index\_productId\_orderDate` (`product\_productId`, `order\_date`);

## Optimised Query with RANK()

SELECT

t.store\_storeId,

t.store\_name,

t.category\_id,

t.total\_sales,

RANK() OVER (PARTITION BY t.store\_storeId ORDER BY t.total\_sales DESC) AS rank\_within\_store

FROM (

SELECT

o.store\_storeId,

s.store\_name,

p.category\_id,

SUM(o.unit\_price \* o.quantity) AS total\_sales

FROM orders o

LEFT JOIN products p ON o.product\_productId = p.product\_id

LEFT JOIN stores s ON s.store\_id = o.store\_storeId

WHERE o.order\_date >= ? AND o.order\_date <= ?

GROUP BY o.store\_id, p.category\_id

) AS `t`

## Performance Compairson –

|  |  |  |
| --- | --- | --- |
| **Metric** | **Before Optimization** | **After Optimization** |
| Avg Execution Time | ~1220 ms | **~390 ms** |
| Index Used | None | orders\_index\_order\_date  PRIMARY,products\_idx\_productId\_categoryId  PRIMARY,stores\_region\_id |
| Filtered by Date Range | Slow | Fast |
| Meets < 1sec Execution time | No | Yes |

## Poupulate table data:-

“**SeedCommand.php**” created in “**app\src\Command**” folder to seed test data using **Faker\Factory** class