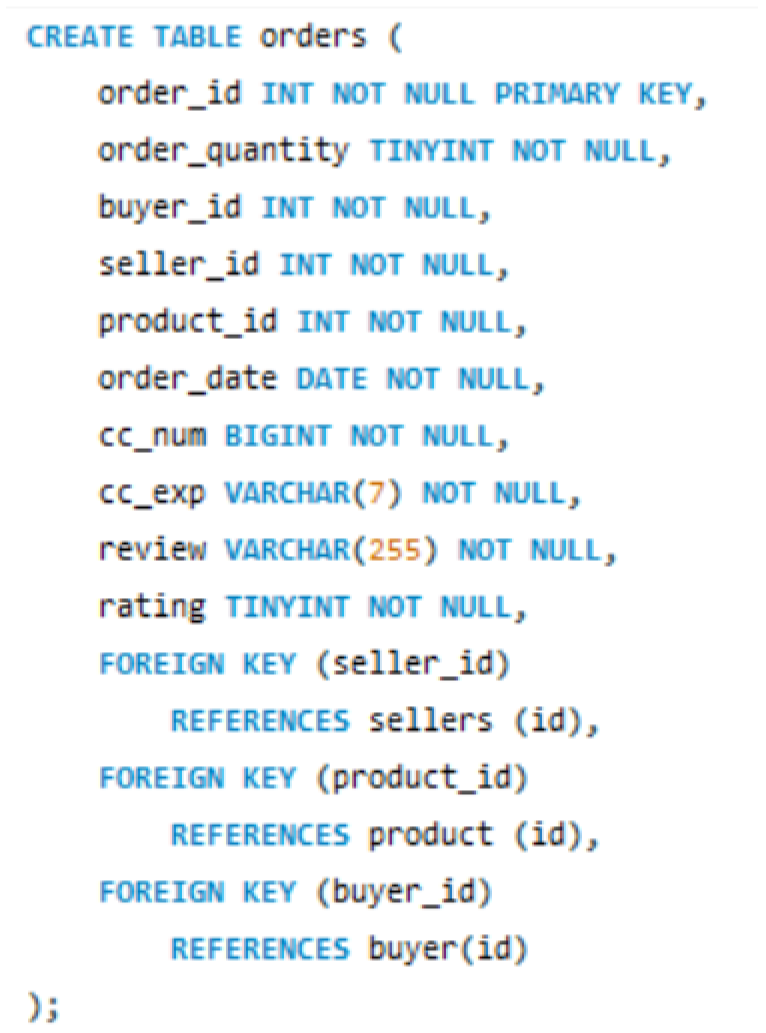
Matthew Jednacz 5/2/24

Tables created:

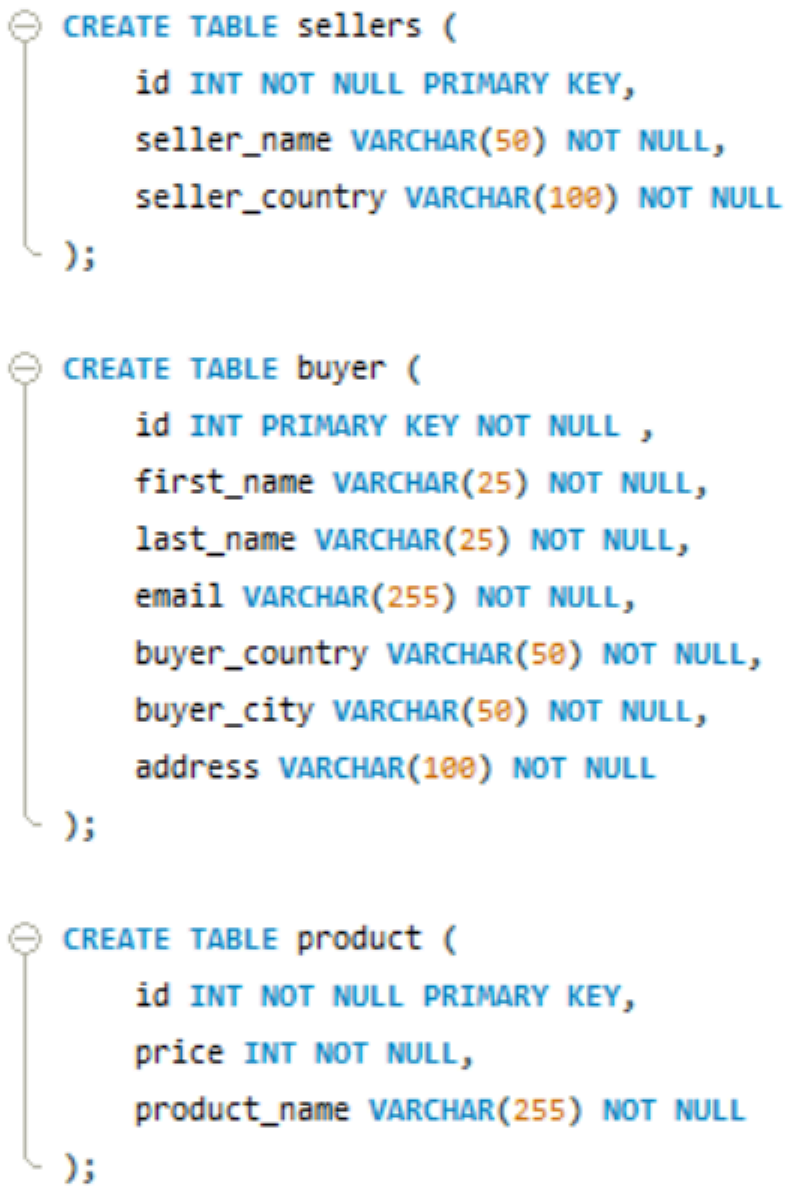
- orders
    - Used to track the orders that are made, largest table with the most foreign keys, those keys being to the buyer, seller, and product table. Each order has a unique review, rating, cc_num and, cc_exp. Therefore not requiring their own tables and may reside in the orders table.
- sellers
    - Used to track the seller id, seller_name, and seller_country. The seller is unlikely to move countries so the data is stable and may stay in the table.
- buyer
    - Contains id, first_name, last_name, email, country, city, and address. While the buyer is more likely to move cities and country it is still considered fairly stable and will only require one table change anyways.
- product
    - Contains id, price, and product_name. Pretty straight forward, table is linked to the orders table and only needs to contain the product specific data.

Order for stored procedure and views,

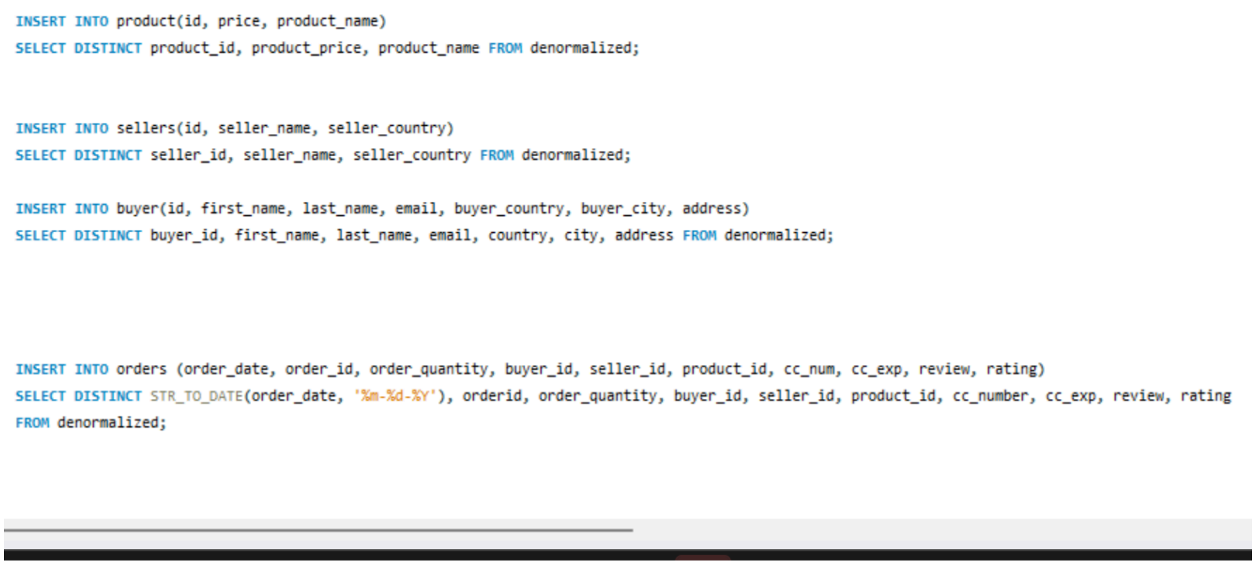
1. sql and result
2. indexed explain
3. non-indexed explain



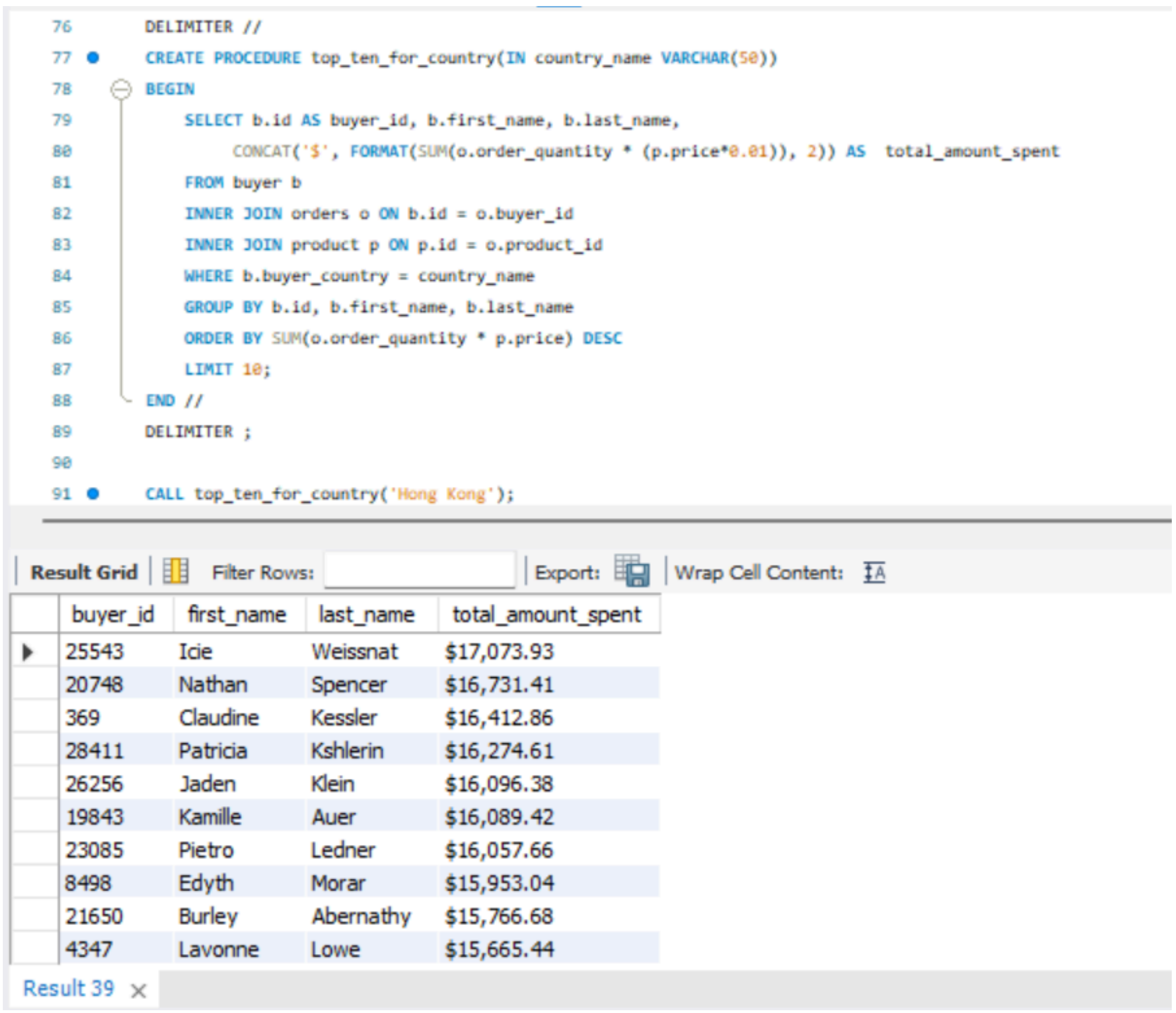
```sql
CREATE TABLE orders (
    order_id INT NOT NULL PRIMARY KEY,
    order_quantity TINYINT NOT NULL,
    buyer_id INT NOT NULL,
    seller_id INT NOT NULL,
    product_id INT NOT NULL,
    order_date DATE NOT NULL,
    cc_num BIGINT NOT NULL,
    cc_exp VARCHAR(7) NOT NULL,
    review VARCHAR(255) NOT NULL,
    rating TINYINT NOT NULL,
    FOREIGN KEY (seller_id)
        REFERENCES sellers (id),
    FOREIGN KEY (product_id)
        REFERENCES product (id),
    FOREIGN KEY (buyer_id)
        REFERENCES buyer(id)
);
```



```
CREATE TABLE sellers (
    id INT NOT NULL PRIMARY KEY,
    seller_name VARCHAR(50) NOT NULL,
    seller_country VARCHAR(100) NOT NULL
);

CREATE TABLE buyer (
    id INT PRIMARY KEY NOT NULL ,
    first_name VARCHAR(25) NOT NULL,
    last_name VARCHAR(25) NOT NULL,
    email VARCHAR(255) NOT NULL,
    buyer_country VARCHAR(50) NOT NULL,
    buyer_city VARCHAR(50) NOT NULL,
    address VARCHAR(100) NOT NULL
);

CREATE TABLE product (
    id INT NOT NULL PRIMARY KEY,
    price INT NOT NULL,
    product_name VARCHAR(255) NOT NULL
);
```

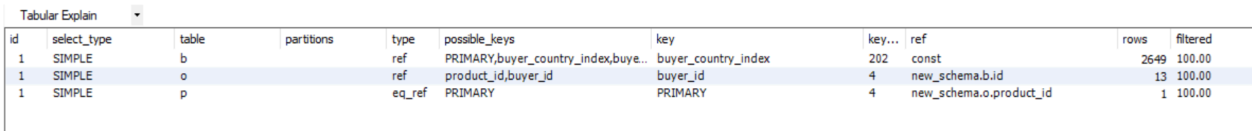


```sql
INSERT INTO product(id, price, product_name)
SELECT DISTINCT product_id, product_price, product_name FROM denormalized;


INSERT INTO sellers(id, seller_name, seller_country)
SELECT DISTINCT seller_id, seller_name, seller_country FROM denormalized;

INSERT INTO buyer(id, first_name, last_name, email, buyer_country, buyer_city, address)
SELECT DISTINCT buyer_id, first_name, last_name, email, country, city, address FROM denormalized;



INSERT INTO orders (order_date, order_id, order_quantity, buyer_id, seller_id, product_id, cc_num, cc_exp, review, rating)
SELECT DISTINCT STR_TO_DATE(order_date, '%m-%d-%Y'), orderid, order_quantity, buyer_id, seller_id, product_id, cc_number, cc_exp, review, rating
FROM denormalized;
```

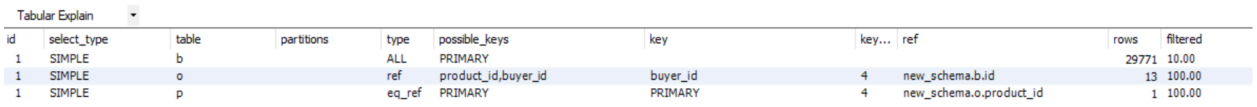


```sql
DELIMITER //
CREATE PROCEDURE top_ten_for_country(IN country_name VARCHAR(50))
BEGIN
    SELECT b.id AS buyer_id, b.first_name, b.last_name,
        CONCAT('$', FORMAT(SUM(o.order_quantity * (p.price*0.01)), 2)) AS  total_amount_spent
    FROM buyer b
    INNER JOIN orders o ON b.id = o.buyer_id
    INNER JOIN product p ON p.id = o.product_id
    WHERE b.buyer_country = country_name
    GROUP BY b.id, b.first_name, b.last_name
    ORDER BY SUM(o.order_quantity * p.price) DESC
    LIMIT 10;
END //
DELIMITER ;

CALL top_ten_for_country('Hong Kong');
```

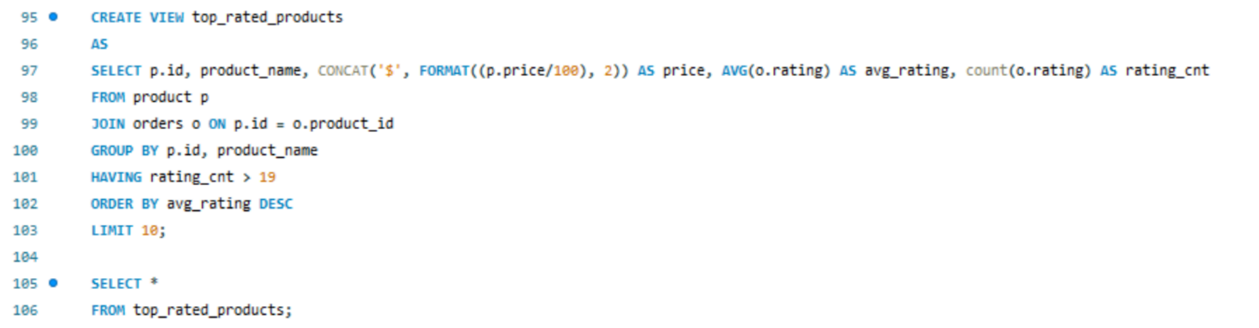
buyer_id	first_name	last_name	total_amount_spent
25543	Icie	Weissnat	$17,073.93
20748	Nathan	Spencer	$16,731.41
369	Claudine	Kessler	$16,412.86
28411	Patricia	Kshlerin	$16,274.61
26256	Jaden	Klein	$16,096.38
19843	Kamille	Auer	$16,089.42
23085	Pietro	Ledner	$16,057.66
8498	Edyth	Morar	$15,953.04
21650	Burley	Abernathy	$15,766.68
4347	Lavonne	Lowe	$15,665.44



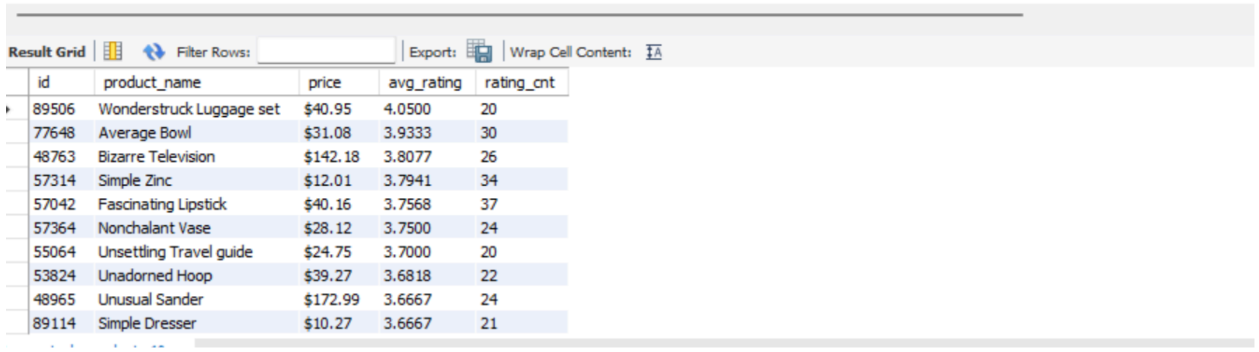
id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	SIMPLE	b		ref	PRIMARY,buyer_country_index,buye...	buyer_country_index	202	const	2649	100.00
1	SIMPLE	o		ref	product_id,buyer_id	buyer_id	4	new_schema.b.id	13	100.00
1	SIMPLE	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00



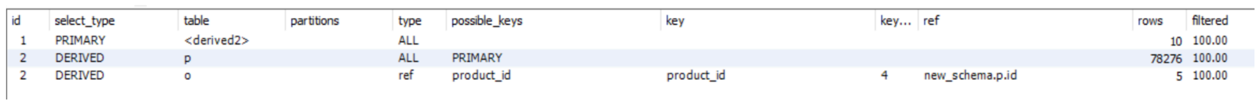
id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	SIMPLE	b		ALL	PRIMARY				29771	10.00
1	SIMPLE	o		ref	product_id,buyer_id	buyer_id	4	new_schema.b.id	13	100.00
1	SIMPLE	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00



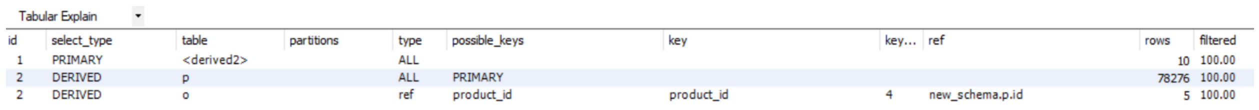
```sql
CREATE VIEW top_rated_products
AS
SELECT p.id, product_name, CONCAT('$', FORMAT((p.price/100), 2)) AS price, AVG(o.rating) AS avg_rating, count(o.rating) AS rating_cnt
FROM product p
JOIN orders o ON p.id = o.product_id
GROUP BY p.id, product_name
HAVING rating_cnt > 19
ORDER BY avg_rating DESC
LIMIT 10;

SELECT *
FROM top_rated_products;
```



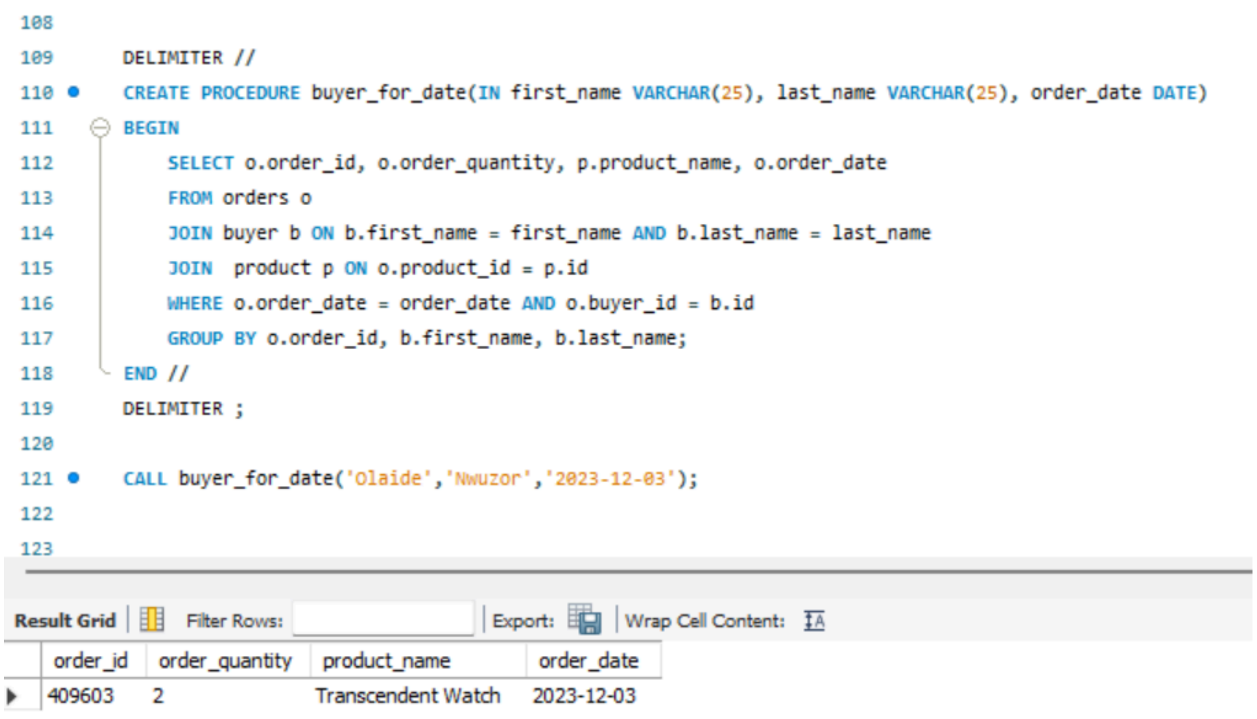
id	product_name	price	avg_rating	rating_cnt
89506	Wonderstruck Luggage set	$40.95	4.0500	20
77648	Average Bowl	$31.08	3.9333	30
48763	Bizarre Television	$142.18	3.8077	26
57314	Simple Zinc	$12.01	3.7941	34
57042	Fascinating Lipstick	$40.16	3.7568	37
57364	Nonchalant Vase	$28.12	3.7500	24
55064	Unsettling Travel guide	$24.75	3.7000	20
53824	Unadorned Hoop	$39.27	3.6818	22
48965	Unusual Sander	$172.99	3.6667	24
89114	Simple Dresser	$10.27	3.6667	21



id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	PRIMARY	<derived2>		ALL					10	100.00
2	DERIVED	p		ALL	PRIMARY				78276	100.00
2	DERIVED	o		ref	product_id	product_id	4	new_schema.p.id	5	100.00

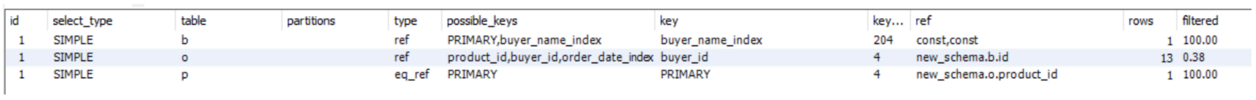


id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	PRIMARY	<derived2>		ALL					10	100.00
2	DERIVED	p		ALL	PRIMARY				78276	100.00
2	DERIVED	o		ref	product_id	product_id	4	new_schema.p.id	5	100.00

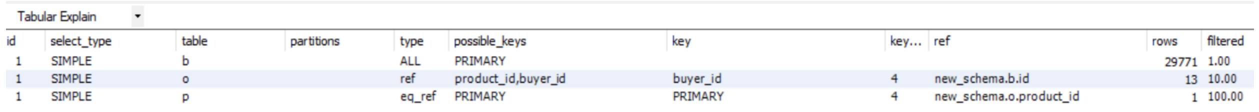


```sql
DELIMITER //
CREATE PROCEDURE buyer_for_date(IN first_name VARCHAR(25), last_name VARCHAR(25), order_date DATE)
BEGIN
    SELECT o.order_id, o.order_quantity, p.product_name, o.order_date
    FROM orders o
    JOIN buyer b ON b.first_name = first_name AND b.last_name = last_name
    JOIN  product p ON o.product_id = p.id
    WHERE o.order_date = order_date AND o.buyer_id = b.id
    GROUP BY o.order_id, b.first_name, b.last_name;
END //
DELIMITER ;

CALL buyer_for_date('Olaide','Nwuzor','2023-12-03');
```

order_id	order_quantity	product_name	order_date
409603	2	Transcendent Watch	2023-12-03

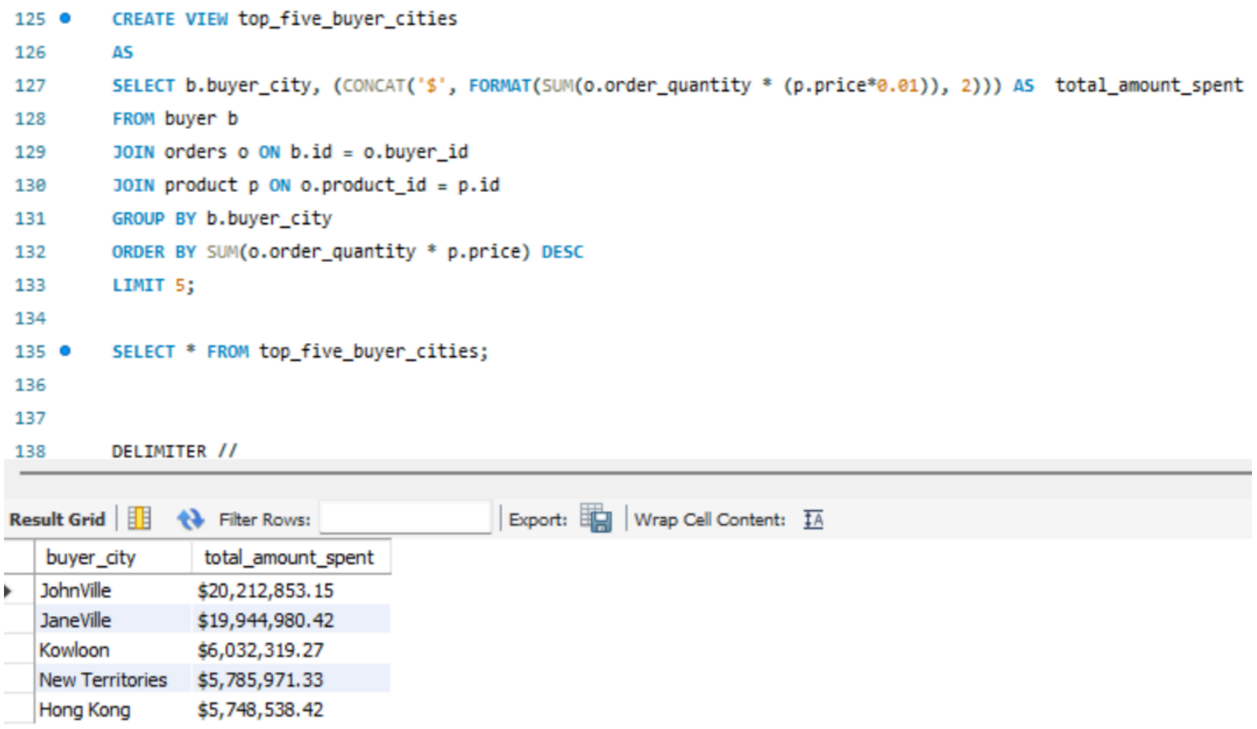


id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	SIMPLE	b		ref	PRIMARY,buyer_name_index	buyer_name_index	204	const,const	1	100.00
1	SIMPLE	o		ref	product_id,buyer_id,order_date_index	buyer_id	4	new_schema.b.id	13	0.38
1	SIMPLE	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00



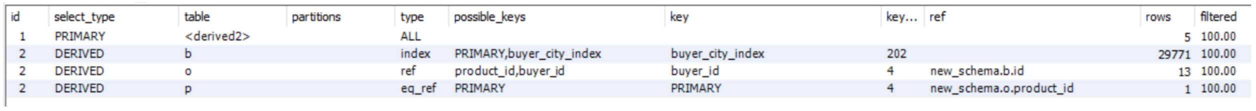
Tabular Explain

id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	SIMPLE	b		ALL	PRIMARY				29771	1.00
1	SIMPLE	o		ref	product_id,buyer_id	buyer_id	4	new_schema.b.id	13	10.00
1	SIMPLE	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00

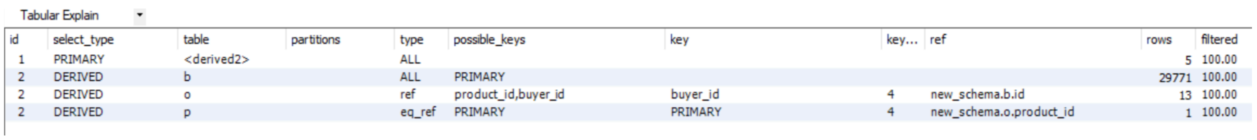


```sql
CREATE VIEW top_five_buyer_cities
AS
SELECT b.buyer_city, (CONCAT('$', FORMAT(SUM(o.order_quantity * (p.price*0.01)), 2))) AS  total_amount_spent
FROM buyer b
JOIN orders o ON b.id = o.buyer_id
JOIN product p ON o.product_id = p.id
GROUP BY b.buyer_city
ORDER BY SUM(o.order_quantity * p.price) DESC
LIMIT 5;

SELECT * FROM top_five_buyer_cities;


DELIMITER //
```

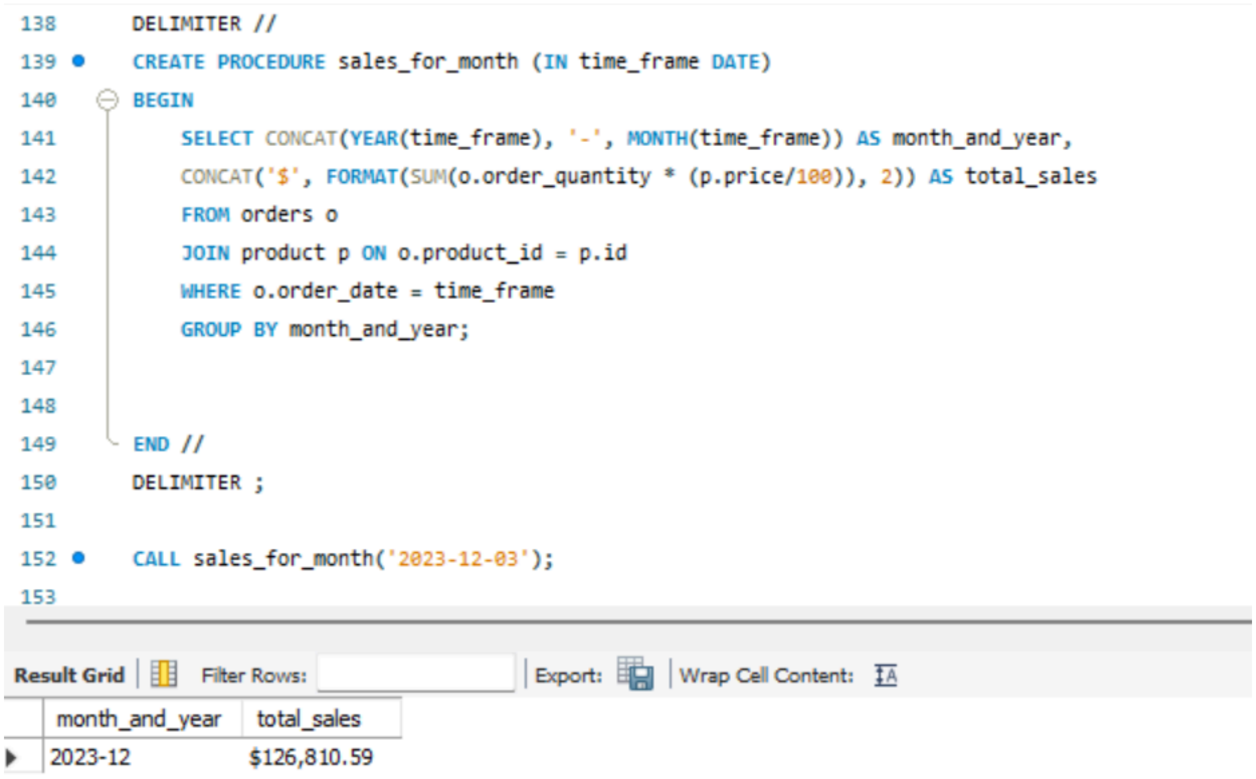
buyer_city	total_amount_spent
JohnVille	$20,212,853.15
JaneVille	$19,944,980.42
Kowloon	$6,032,319.27
New Territories	$5,785,971.33
Hong Kong	$5,748,538.42



id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	PRIMARY	<derived2>		ALL					5	100.00
2	DERIVED	b		index	PRIMARY,buyer_city_index	buyer_city_index	202		29771	100.00
2	DERIVED	o		ref	product_id,buyer_id	buyer_id	4	new_schema.b.id	13	100.00
2	DERIVED	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00



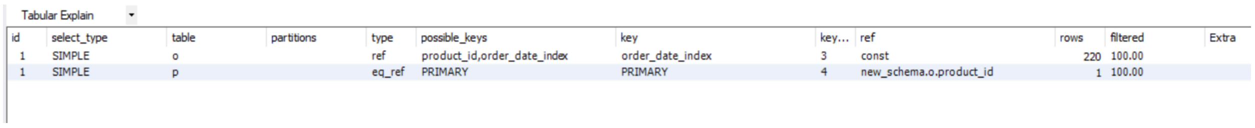
id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	PRIMARY	<derived2>		ALL					5	100.00
2	DERIVED	b		ALL	PRIMARY				29771	100.00
2	DERIVED	o		ref	product_id,buyer_id	buyer_id	4	new_schema.b.id	13	100.00
2	DERIVED	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00



```sql
DELIMITER //
CREATE PROCEDURE sales_for_month (IN time_frame DATE)
BEGIN
    SELECT CONCAT(YEAR(time_frame), '-', MONTH(time_frame)) AS month_and_year,
    CONCAT('$', FORMAT(SUM(o.order_quantity * (p.price/100)), 2)) AS total_sales
    FROM orders o
    JOIN product p ON o.product_id = p.id
    WHERE o.order_date = time_frame
    GROUP BY month_and_year;


END //
DELIMITER ;

CALL sales_for_month('2023-12-03');
```

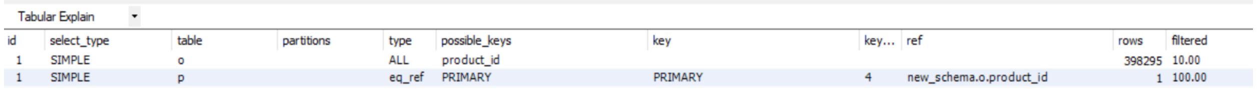
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

month_and_year	total_sales
2023-12	$126,810.59



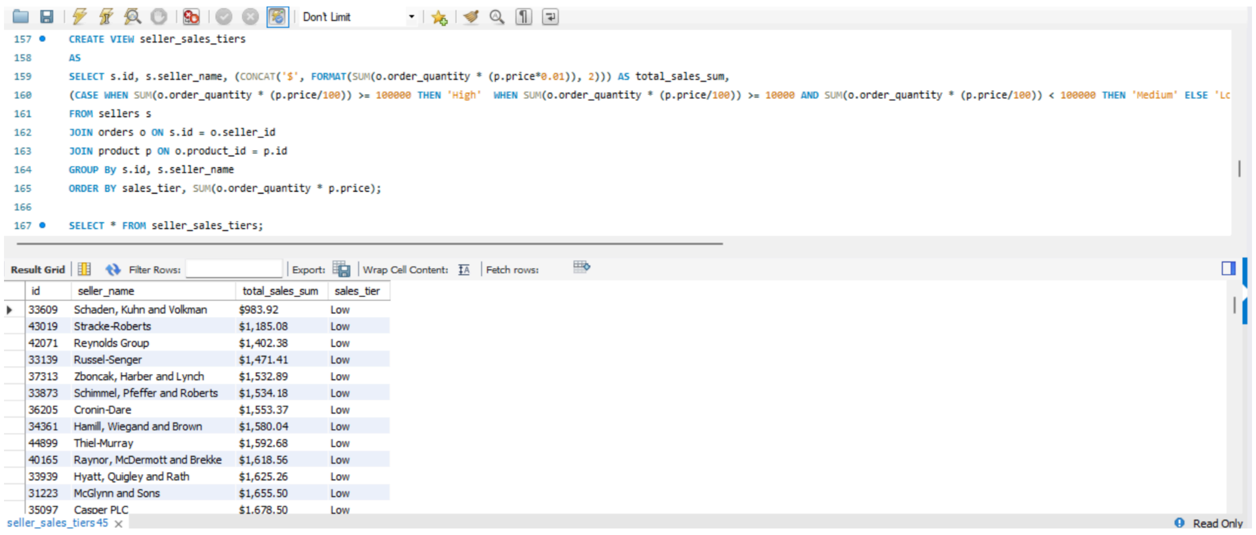
Tabular Explain

id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered	Extra
1	SIMPLE	o		ref	product_id,order_date_index	order_date_index	3	const	220	100.00	
1	SIMPLE	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00	



Tabular Explain

id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	SIMPLE	o		ALL	product_id				398295	10.00
1	SIMPLE	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00

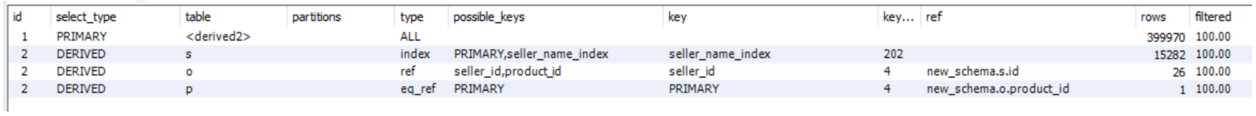


```sql
CREATE VIEW seller_sales_tiers
AS
SELECT s.id, s.seller_name, (CONCAT('$', FORMAT(SUM(o.order_quantity * (p.price*0.01)), 2))) AS total_sales_sum,
(CASE WHEN SUM(o.order_quantity * (p.price/100)) >= 100000 THEN 'High'  WHEN SUM(o.order_quantity * (p.price/100)) >= 10000 AND SUM(o.order_quantity * (p.price/100)) < 100000 THEN 'Medium' ELSE 'Lc
FROM sellers s
JOIN orders o ON s.id = o.seller_id
JOIN product p ON o.product_id = p.id
GROUP By s.id, s.seller_name
ORDER BY sales_tier, SUM(o.order_quantity * p.price);

SELECT * FROM seller_sales_tiers;
```

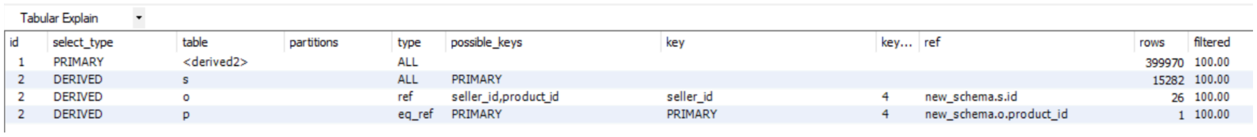
id	seller_name	total_sales_sum	sales_tier
33609	Schaden, Kuhn and Volkman	$983.92	Low
43019	Stracke-Roberts	$1,185.08	Low
42071	Reynolds Group	$1,402.38	Low
33139	Russel-Senger	$1,471.41	Low
37313	Zboncak, Harber and Lynch	$1,532.89	Low
33873	Schimmel, Pfeffer and Roberts	$1,534.18	Low
36205	Cronin-Dare	$1,553.37	Low
34361	Hamill, Wiegand and Brown	$1,580.04	Low
44899	Thiel-Murray	$1,592.68	Low
40165	Raynor, McDermott and Brekke	$1,618.56	Low
33939	Hyatt, Quigley and Rath	$1,625.26	Low
31223	McGlynn and Sons	$1,655.50	Low
35097	Casper PLC	$1,678.50	Low

seller_sales_tiers 45

Read Only

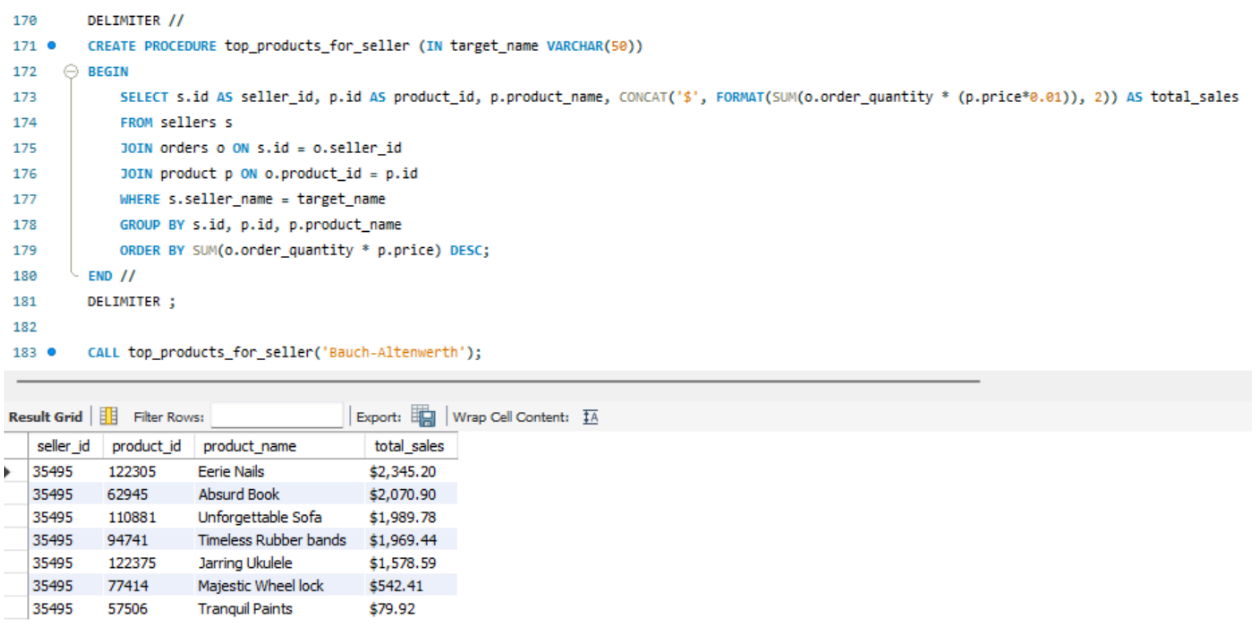


id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	PRIMARY	<derived2>		ALL					399970	100.00
2	DERIVED	s		index	PRIMARY,seller_name_index	seller_name_index	202		15282	100.00
2	DERIVED	o		ref	seller_id,product_id	seller_id	4	new_schema.s.id	26	100.00
2	DERIVED	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00



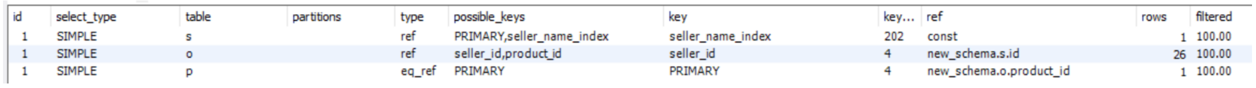
Tabular Explain

id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	PRIMARY	<derived2>		ALL					399970	100.00
2	DERIVED	s		ALL	PRIMARY				15282	100.00
2	DERIVED	o		ref	seller_id,product_id	seller_id	4	new_schema.s.id	26	100.00
2	DERIVED	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00

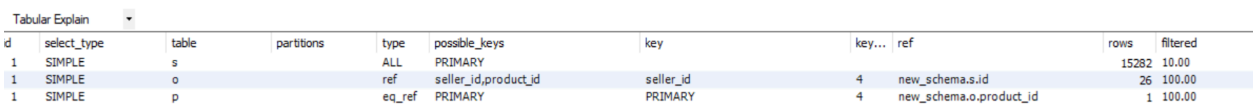


```sql
DELIMITER //
CREATE PROCEDURE top_products_for_seller (IN target_name VARCHAR(50))
BEGIN
    SELECT s.id AS seller_id, p.id AS product_id, p.product_name, CONCAT('$', FORMAT(SUM(o.order_quantity * (p.price*0.01)), 2)) AS total_sales
    FROM sellers s
    JOIN orders o ON s.id = o.seller_id
    JOIN product p ON o.product_id = p.id
    WHERE s.seller_name = target_name
    GROUP BY s.id, p.id, p.product_name
    ORDER BY SUM(o.order_quantity * p.price) DESC;
END //
DELIMITER ;

CALL top_products_for_seller('Bauch-Altenwerth');
```

seller_id	product_id	product_name	total_sales
35495	122305	Eerie Nails	$2,345.20
35495	62945	Absurd Book	$2,070.90
35495	110881	Unforgettable Sofa	$1,989.78
35495	94741	Timeless Rubber bands	$1,969.44
35495	122375	Jarring Ukulele	$1,578.59
35495	77414	Majestic Wheel lock	$542.41
35495	57506	Tranquil Paints	$79.92

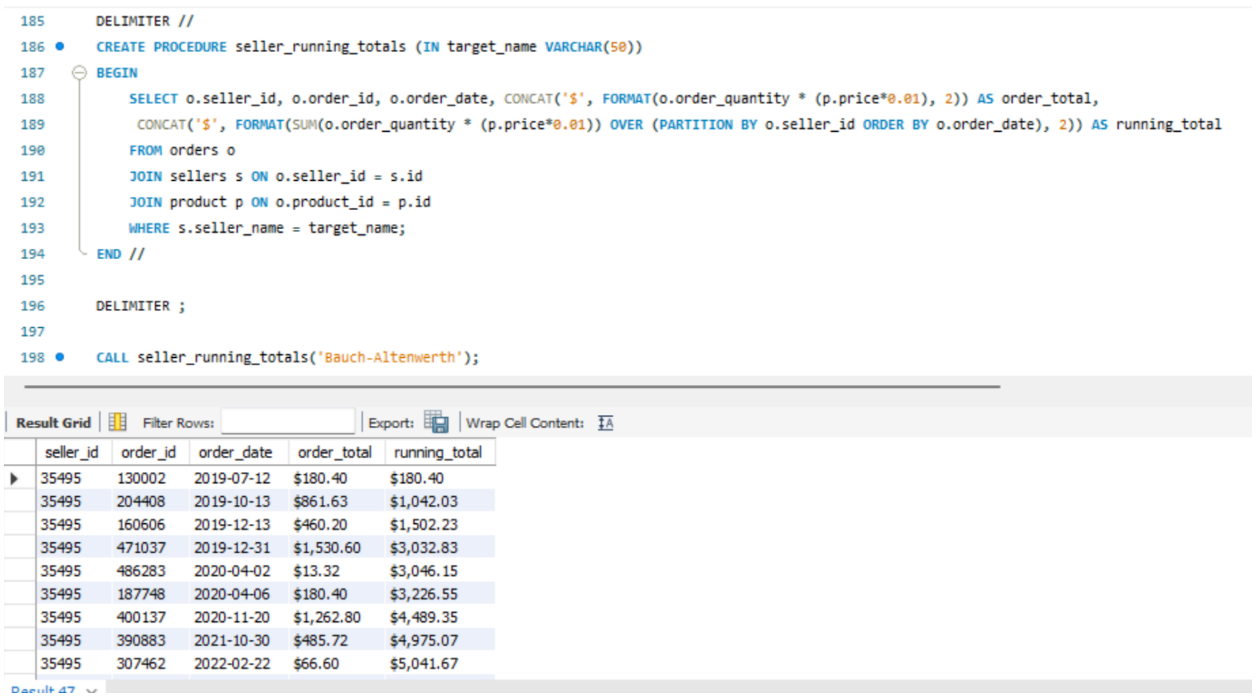


id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	SIMPLE	s		ref	PRIMARY,seller_name_index	seller_name_index	202	const	1	100.00
1	SIMPLE	o		ref	seller_id,product_id	seller_id	4	new_schema.s.id	26	100.00
1	SIMPLE	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00



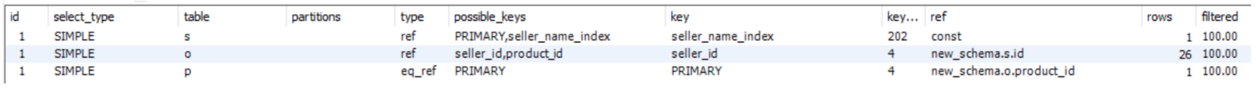
Tabular Explain

d	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	SIMPLE	s		ALL	PRIMARY				15282	10.00
1	SIMPLE	o		ref	seller_id,product_id	seller_id	4	new_schema.s.id	26	100.00
1	SIMPLE	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00

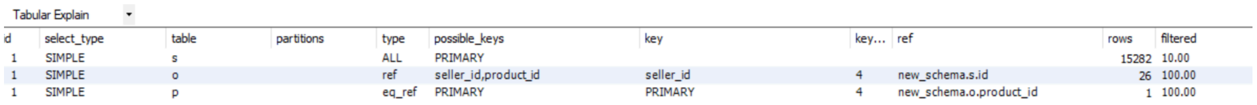


```sql
DELIMITER //
CREATE PROCEDURE seller_running_totals (IN target_name VARCHAR(50))
BEGIN
    SELECT o.seller_id, o.order_id, o.order_date, CONCAT('$', FORMAT(o.order_quantity * (p.price*0.01), 2)) AS order_total,
     CONCAT('$', FORMAT(SUM(o.order_quantity * (p.price*0.01)) OVER (PARTITION BY o.seller_id ORDER BY o.order_date), 2)) AS running_total
    FROM orders o
    JOIN sellers s ON o.seller_id = s.id
    JOIN product p ON o.product_id = p.id
    WHERE s.seller_name = target_name;
END //

DELIMITER ;

CALL seller_running_totals('Bauch-Altenwerth');
```

seller_id	order_id	order_date	order_total	running_total
35495	130002	2019-07-12	$180.40	$180.40
35495	204408	2019-10-13	$861.63	$1,042.03
35495	160606	2019-12-13	$460.20	$1,502.23
35495	471037	2019-12-31	$1,530.60	$3,032.83
35495	486283	2020-04-02	$13.32	$3,046.15
35495	187748	2020-04-06	$180.40	$3,226.55
35495	400137	2020-11-20	$1,262.80	$4,489.35
35495	390883	2021-10-30	$485.72	$4,975.07
35495	307462	2022-02-22	$66.60	$5,041.67



id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	SIMPLE	s		ref	PRIMARY,seller_name_index	seller_name_index	202	const	1	100.00
1	SIMPLE	o		ref	seller_id,product_id	seller_id	4	new_schema.s.id	26	100.00
1	SIMPLE	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00



Tabular Explain

id	select_type	table	partitions	type	possible_keys	key	key...	ref	rows	filtered
1	SIMPLE	s		ALL	PRIMARY				15282	10.00
1	SIMPLE	o		ref	seller_id,product_id	seller_id	4	new_schema.s.id	26	100.00
1	SIMPLE	p		eq_ref	PRIMARY	PRIMARY	4	new_schema.o.product_id	1	100.00

```sql
ALTER TABLE buyer ADD INDEX buyer_country_index (buyer_country);

ALTER TABLE buyer ADD INDEX buyer_name_index (first_name, last_name);

ALTER TABLE buyer ADD INDEX buyer_city_index (buyer_city);

ALTER TABLE orders ADD INDEX order_date_index (order_date);

ALTER TABLE orders ADD INDEX order_quantity_index (order_quantity);

ALTER TABLE sellers ADD INDEX seller_name_index (seller_name);

ALTER TABLE product ADD INDEX price_index (price);
```