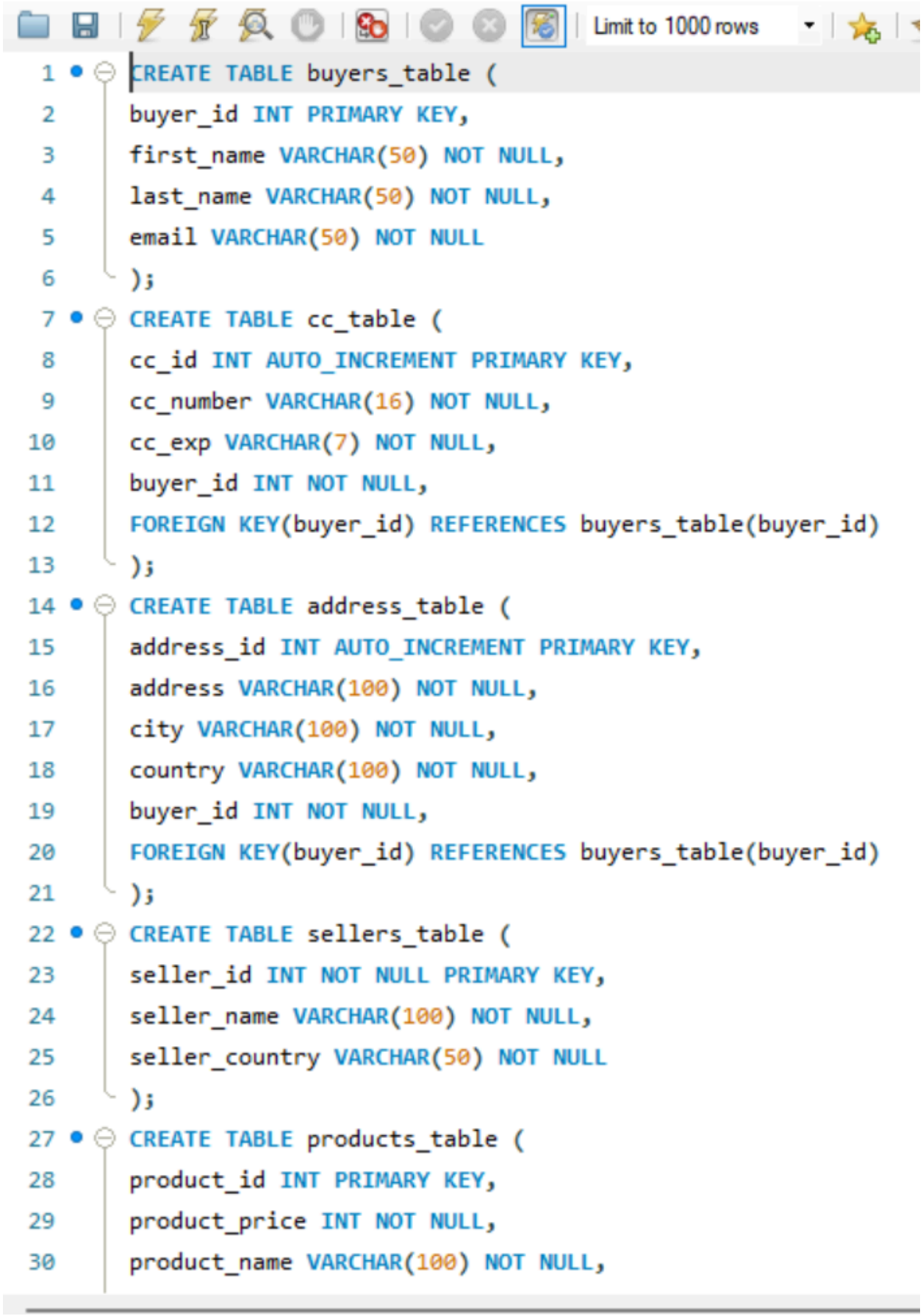
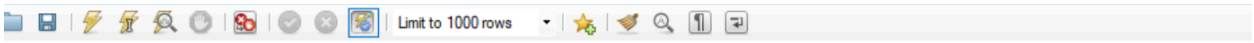

Limit to 1000 rows
CREATE TABLE buyers_table (
buyer_id INT PRIMARY KEY,
first_name VARCHAR(50) NOT NULL,
last_name VARCHAR(50) NOT NULL,
email VARCHAR(50) NOT NULL
);
CREATE TABLE cc_table (
cc_id INT AUTO_INCREMENT PRIMARY KEY,
cc_number VARCHAR(16) NOT NULL,
cc_exp VARCHAR(7) NOT NULL,
buyer_id INT NOT NULL,
FOREIGN KEY(buyer_id) REFERENCES buyers_table(buyer_id)
);
CREATE TABLE address_table (
address_id INT AUTO_INCREMENT PRIMARY KEY,
address VARCHAR(100) NOT NULL,
city VARCHAR(100) NOT NULL,
country VARCHAR(100) NOT NULL,
buyer_id INT NOT NULL,
FOREIGN KEY(buyer_id) REFERENCES buyers_table(buyer_id)
);
CREATE TABLE sellers_table (
seller_id INT NOT NULL PRIMARY KEY,
seller_name VARCHAR(100) NOT NULL,
seller_country VARCHAR(50) NOT NULL
);
CREATE TABLE products_table (
product_id INT PRIMARY KEY,
product_price INT NOT NULL,
product_name VARCHAR(100) NOT NULL,

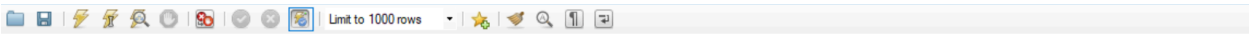
```sql
CREATE TABLE products_table (
product_id INT PRIMARY KEY,
product_price INT NOT NULL,
product_name VARCHAR(100) NOT NULL,
seller_id INT NOT NULL,
FOREIGN KEY(seller_id) REFERENCES sellers_table(seller_id)
);
CREATE TABLE orders_table (
order_id INT PRIMARY KEY,
order_quantity INT NOT NULL,
order_date DATE NOT NULL,
cc_id INT NOT NULL,
buyer_id INT NOT NULL,
seller_id INT NOT NULL,
product_id INT NOT NULL,
review VARCHAR(255) NOT NULL,
rating DECIMAL(2,1) NOT NULL,
FOREIGN KEY (cc_id) REFERENCES cc_table(cc_id),
FOREIGN KEY (buyer_id) REFERENCES buyers_table(buyer_id),
FOREIGN KEY (seller_id) REFERENCES sellers_table(seller_id),
FOREIGN KEY (product_id) REFERENCES products_table(product_id)
);
```

```
DELIMITER $$

CREATE PROCEDURE top_ten_for_country(IN country_input VARCHAR(100))
BEGIN
SELECT b.buyer_id, b.first_name, b.last_name, CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_amount_spent
 FROM buyers_table b
 JOIN address_table a ON b.buyer_id = a.buyer_id
 JOIN orders_table o ON b.buyer_id = o.buyer_id
 JOIN products_table p ON p.product_id = o.product_id
 WHERE a.country = country_input
 GROUP BY b.buyer_id, b.first_name, b.last_name
 ORDER BY SUM(o.order_quantity * p.product_price) DESC
 LIMIT 10;
 END $$
 DELIMITER ;


 CREATE VIEW top_rated_products AS
 SELECT p.product_id, p.product_name, CONCAT('$',FORMAT(p.product_price / 100, 2))  AS product_price,
AVG(o.rating) AS avg_rating, COUNT(o.rating) AS rating_count
FROM products_table p
JOIN orders_table o ON p.product_id = o.product_id
GROUP BY p.product_id, p.product_name
HAVING COUNT(o.rating) > 20
ORDER BY AVG(o.rating) DESC
LIMIT 10;
```

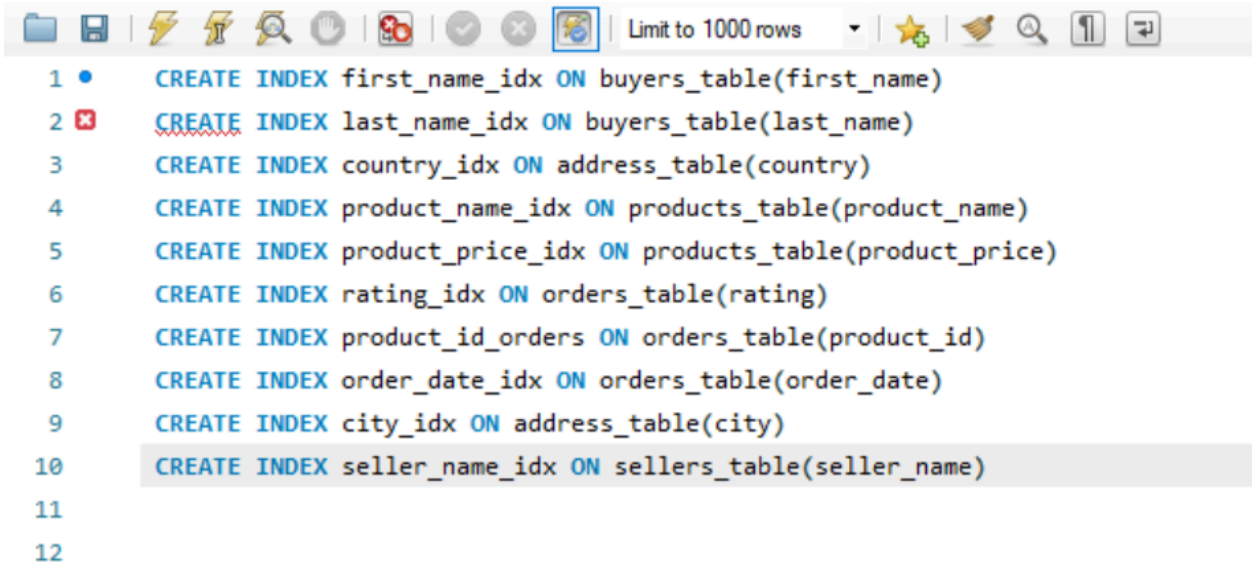


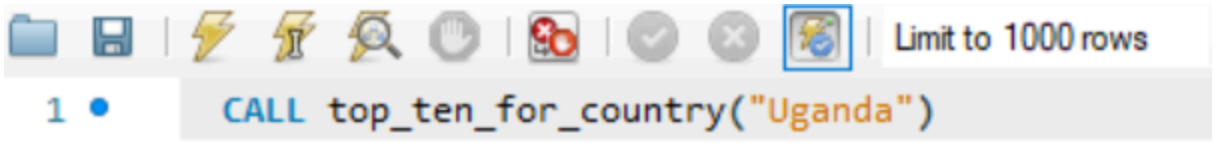

```
30    DELIMITER $$
31
32    CREATE PROCEDURE buyer_for_date(IN first_name_input VARCHAR(50), IN last_name_input VARCHAR(50), IN order_date_input DATE)
33    BEGIN
34    SELECT o.order_id, o.order_quantity, p.product_name, o.order_date
35    FROM buyers_table b
36    JOIN orders_table o ON b.buyer_id = o.buyer_id
37    JOIN products_table p ON p.product_id = o.product_id
38    WHERE b.first_name = first_name_input AND b.last_name = last_name_input AND o.order_date = order_date_input;
39    END $$
40    DELIMITER ;
41
42
43    CREATE VIEW top_five_buyer_cities AS
44    SELECT a.city,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_amount_spent
45    FROM buyers_table b
46    JOIN address_table a ON b.buyer_id = a.buyer_id
47    JOIN orders_table o ON b.buyer_id = o.buyer_id
48    JOIN products_table p ON p.product_id = o.product_id
49    GROUP BY a.city
50    ORDER BY SUM(o.order_quantity * p.product_price) DESC
51    LIMIT 5;
52
53
```

```
DELIMITER $$

 CREATE PROCEDURE sales_for_month(IN month_input INT, IN year_input INT)
 BEGIN
  SELECT CONCAT(YEAR(o.order_date), '-', DATE_FORMAT(o.order_date, '%m')) AS month_and_year,
      CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2)) AS total_sales
  FROM products_table p
 JOIN orders_table o ON p.product_id = o.product_id
   WHERE MONTH(o.order_date) = month_input
   AND YEAR(o.order_date) = year_input
  GROUP BY month_and_year;
  END $$
DELIMITER ;


CREATE VIEW seller_sales_tiers AS
  SELECT s.seller_id,s.seller_name,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_sales,
 CASE
     WHEN SUM(o.order_quantity * p.product_price) >= 10000000 THEN 'High'
     WHEN SUM(o.order_quantity * p.product_price) >= 1000000 AND SUM(o.order_quantity * p.product_price) < 10000000 THEN 'Medium'
     WHEN SUM(o.order_quantity * p.product_price) < 1000000 THEN 'Low'
    END AS sales_tier
 FROM sellers_table s
 JOIN products_table p ON s.seller_id = p.seller_id
 JOIN orders_table o ON p.product_id = o.product_id
 GROUP BY s.seller_id, s.seller_name
 ORDER BY SUM(o.order_quantity * p.product_price) DESC
```




```

DELIMITER $$

 CREATE PROCEDURE top_products_for_seller(IN seller_name_input VARCHAR(100))
 BEGIN
  SELECT s.seller_id, p.product_id, p.product_name,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_sales
  FROM sellers_table s
  JOIN products_table p ON s.seller_id = p.seller_id
  JOIN orders_table o ON p.product_id = o.product_id
  WHERE s.seller_name = seller_name_input
  GROUP BY s.seller_id,p.product_id,p.product_name
  ORDER BY SUM(o.order_quantity * p.product_price) DESC;
  END $$
  DELIMITER ;



DELIMITER $$

 CREATE PROCEDURE seller_running_totals(IN seller_name_input VARCHAR(100))
 BEGIN
  SELECT  s.seller_id, o.order_id, o.order_date,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS order_total,
   CONCAT('$', FORMAT(SUM(o.order_quantity * p.product_price) OVER ( ORDER BY o.order_date) /100, 2 )) AS running_total
   FROM sellers_table s
   JOIN products_table p ON s.seller_id = p.seller_id
   JOIN orders_table o ON p.product_id = o.product_id
   WHERE s.seller_name = seller_name_input
   GROUP BY s.seller_id,o.order_id,o.order_date;
  END $$
  DELIMITER ;
```


Limit to 1000 rows
1 CREATE INDEX first_name_idx ON buyers_table(first_name)
2 CREATE INDEX last_name_idx ON buyers_table(last_name)
3 CREATE INDEX country_idx ON address_table(country)
4 CREATE INDEX product_name_idx ON products_table(product_name)
5 CREATE INDEX product_price_idx ON products_table(product_price)
6 CREATE INDEX rating_idx ON orders_table(rating)
7 CREATE INDEX product_id_orders ON orders_table(product_id)
8 CREATE INDEX order_date_idx ON orders_table(order_date)
9 CREATE INDEX city_idx ON address_table(city)
10 CREATE INDEX seller_name_idx ON sellers_table(seller_name)
11
12



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

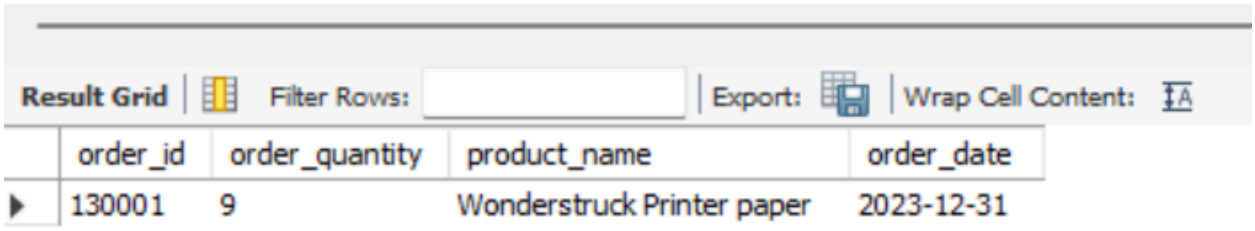
buyer_id	first_name	last_name	total_amount_spent
16518	Dolly	Were	$16,871.33
14518	Gaston	Mushabe	$16,855.98
10873	Ezra	Ahebwe	$16,764.56
14410	Juliana	Abayisenga	$16,399.46
13309	Frankie	Gamwera	$16,243.88
4657	Nadia	Kabanda	$16,120.58
25624	Charlie	Kabuubi	$16,116.06
11145	Troy	Kazibwe	$15,878.06
17450	Margaret	Atuhe	$15,682.56
10639	Viola	Nyeko	$15,653.46

```
2 •    SELECT *
3      FROM top_rated_products
```

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

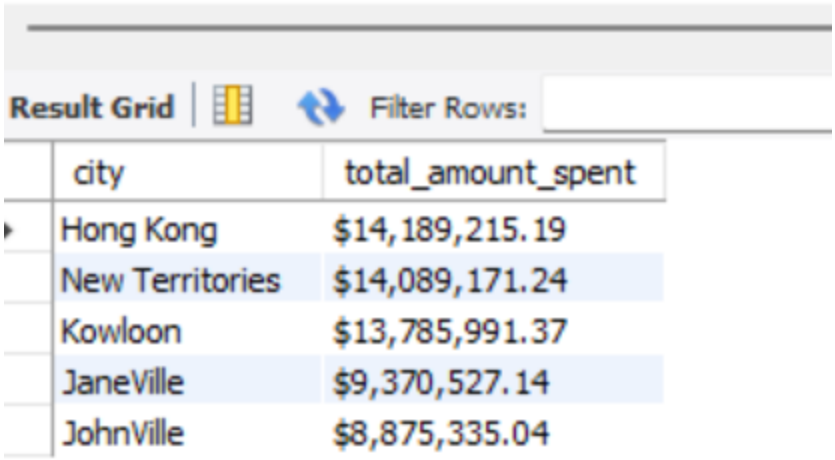
product_id	product_name	product_price	avg_rating	rating_count
55262	Unearthly Fridge	$49.93	3.76000	25
49946	Enchanting Triangle	$20.30	3.75000	22
52110	Impressive Cologne	$18.81	3.66667	24
93788	Bewitched Trousers	$47.22	3.66667	21
99073	Witching Scissors	$111.49	3.66667	24
47977	Ominous Bird feeder	$118.67	3.64815	27
119112	Mesmeric Modem	$14.78	3.59524	21
59314	Unclear Curling iron	$43.90	3.54839	31
58796	Magnetic Umbrella	$47.14	3.53448	29
47416	Bewitched First aid kit	$27.66	3.51667	30

```
4 •  CALL  buyer_for_date("Jimmy","Tumusiime","2023-12-31")
```



order_id	order_quantity	product_name	order_date
130001	9	Wonderstruck Printer paper	2023-12-31

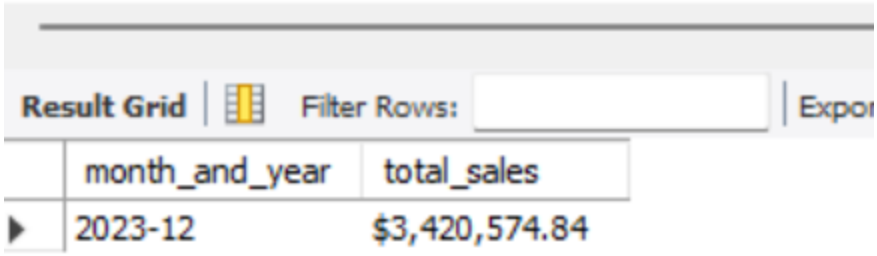
```
5 •   SELECT *
6     FROM top_five_buyer_cities
```



Result Grid | Filter Rows:

city	total_amount_spent
Hong Kong	$14,189,215.19
New Territories	$14,089,171.24
Kowloon	$13,785,991.37
JaneVille	$9,370,527.14
JohnVille	$8,875,335.04

```
7 •  CALL sales_for_month(12,2023)
```



Result Grid | Filter Rows: | Expor

month_and_year	total_sales
2023-12	$3,420,574.84

```
8 •  SELECT *
9    FROM seller_sales_tiers
10
```

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

seller_id	seller_name	total_sales	sales_tier
31584	Erdman LLC	$46,014.54	Medium
32354	Ferry, Cartwright and Crona	$39,884.51	Medium
33120	Eichmann, Williamson and Mann	$39,730.35	Medium
33212	Hill-Prohaska	$39,199.54	Medium
34784	Feeney-Daniel	$39,137.40	Medium
37170	Friesen, Goodwin and Hills	$36,913.40	Medium
43646	Balistreri, Schmitt and Ernser	$36,518.88	Medium
36812	Mayer-Von	$36,340.64	Medium
32164	Herzog, Grant and Rice	$35,732.14	Medium
37164	Kling Group	$35,016.33	Medium
43320	Abernathy-Koss	$34,600.45	Medium
41362	Rolfson, Ankunding and Bergst...	$34,525.58	Medium
45686	Walsh-Kassulke	$34,506.83	Medium
38264	Homenick-Hand	$34,190.28	Medium
44426	Wiza Group	$34,112.25	Medium
41694	Hyatt-Jacobi	$33,964.48	Medium
41250	Wolff, Gleichner and Gulgowski	$33,889.56	Medium
35442	Johnston, Effertz and Effertz	$33,838.32	Medium

seller_sales_tiers 40 ×

```
10 •  CALL top_products_for_seller("Armstrong, Fahey and Bergnaum")
11
```

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

seller_id	product_id	product_name	total_sales
45138	47863	Eerie Travel blanket	$6,012.00
45138	122265	Breathtaking Wedge	$5,310.63
45138	76933	Unappetizing Screwdriver	$1,566.18
45138	88448	Typical Highlighter	$895.68
45138	67060	Wonderstruck Printer paper	$495.45

```
11 •  CALL seller_running_totals("Armstrong, Fahey and Bergnaum")
```

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

seller_id	order_id	order_date	order_total	running_total
45138	134049	2019-12-25	$18.35	$18.35
45138	494223	2020-02-29	$1,180.14	$1,198.49
45138	374836	2020-07-02	$139.95	$1,338.44
45138	267867	2020-12-14	$1,336.00	$2,674.44
45138	244779	2021-01-23	$146.80	$2,821.24
45138	315116	2021-03-19	$251.91	$3,073.15
45138	505494	2021-04-27	$1,139.04	$4,212.19
45138	420325	2021-07-04	$427.14	$4,639.33
45138	167416	2021-09-23	$27.99	$4,667.32
45138	407032	2022-01-04	$393.38	$5,060.70
45138	211201	2022-01-28	$1,966.90	$7,027.60
45138	240113	2022-06-25	$801.60	$7,829.20
45138	221611	2022-08-07	$36.70	$7,865.90
45138	281146	2023-02-15	$267.20	$8,133.10
45138	219409	2023-03-02	$223.92	$8,357.02
45138	433873	2023-06-15	$1,336.00	$9,693.02
45138	146442	2023-07-08	$1,770.21	$11,463.23
45138	264059	2023-09-17	$668.00	$12,131.23

Result 44

```
1  EXPLAIN SELECT b.buyer_id, b.first_name, b.last_name, CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_amount_spent
2  FROM buyers_table b
3  JOIN address_table a ON b.buyer_id = a.buyer_id
4  JOIN orders_table o ON b.buyer_id = o.buyer_id
5  JOIN products_table p ON p.product_id = o.product_id
6  WHERE a.country = "Uganda"
7  GROUP BY b.buyer_id, b.first_name, b.last_name
8  ORDER BY SUM(o.order_quantity * p.product_price) DESC
9  LIMIT 10;
10
11 -- CREATE VIEW top_rated_products AS
12 -- SELECT p.product_id, p.product_name, CONCAT('$',FORMAT(p.product_price / 100, 2))  AS product_price,
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	a	NULL	ALL	buyer_id	NULL	NULL	NULL	30151	10.00	Using where; Using temporary; Using filesort
1	SIMPLE	b	NULL	eq_ref	PRIMARY	PRIMARY	4	cisc3810_project.a.buyer_id	1	100.00	NULL
1	SIMPLE	o	NULL	ref	buyer_id,product_id	buyer_id	4	cisc3810_project.a.buyer_id	13	100.00	NULL
1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	cisc3810_project.o.product_id	1	100.00	NULL

abase Server Tools Scripting Help

Query 1 | SQL File 4* | SQL File 5* | address_table | SQL File 7* | SQL File 8* | SQL File 9* | SQL File 10* | SQL File 11* | SQL File 12*

```
1  EXPLAIN SELECT b.buyer_id, b.first_name, b.last_name, CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_amount_spent
2  FROM buyers_table b
3  JOIN address_table a ON b.buyer_id = a.buyer_id
4  JOIN orders_table o ON b.buyer_id = o.buyer_id
5  JOIN products_table p ON p.product_id = o.product_id
6  WHERE a.country = "Uganda"
7  GROUP BY b.buyer_id, b.first_name, b.last_name
8  ORDER BY SUM(o.order_quantity * p.product_price) DESC
9  LIMIT 10;
10
11 -- CREATE VIEW top_rated_products AS
12 -- SELECT p.product_id, p.product_name, CONCAT('$',FORMAT(p.product_price / 100, 2))  AS product_price,
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	a	NULL	ref	buyer_id,country_idx	country_idx	402	const	6226	100.00	Using temporary; Using filesort
1	SIMPLE	b	NULL	eq_ref	PRIMARY	PRIMARY	4	cisc3810_project.a.buyer_id	1	100.00	NULL
1	SIMPLE	o	NULL	ref	buyer_id,product_id	buyer_id	4	cisc3810_project.a.buyer_id	13	100.00	NULL
1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	cisc3810_project.o.product_id	1	100.00	NULL

```
*/

-- CREATE VIEW top_rated_products AS
EXPLAIN SELECT p.product_id, p.product_name, CONCAT('$',FORMAT(p.product_price / 100, 2))  AS product_price,
AVG(o.rating) AS avg_rating, COUNT(o.rating) AS rating_count
FROM products_table p
JOIN orders_table o ON p.product_id = o.product_id
GROUP BY p.product_id, p.product_name
HAVING COUNT(o.rating) > 20
ORDER BY AVG(o.rating) DESC
LIMIT 10;
```

Result Grid

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	p	NULL	ALL	PRIMARY	NULL	NULL	NULL	78789	100.00	Using temporary; Using filesort
1	SIMPLE	o	NULL	ref	product_id	product_id	4	cisc3810_project.p.product_id	5	100.00	NULL

Query 1 | SQL File 4* | SQL File 5* | address_table | SQL File 7* | SQL File 8* | SQL File 9* | SQL File 10* | SQL File 11* | SQL File 12*

```
*/

-- CREATE VIEW top_rated_products AS
EXPLAIN SELECT p.product_id, p.product_name, CONCAT('$',FORMAT(p.product_price / 100, 2))  AS product_price,
AVG(o.rating) AS avg_rating, COUNT(o.rating) AS rating_count
FROM products_table p
JOIN orders_table o ON p.product_id = o.product_id
GROUP BY p.product_id, p.product_name
HAVING COUNT(o.rating) > 20
ORDER BY AVG(o.rating) DESC
LIMIT 10;
```

Result Grid

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	p	NULL	ALL	PRIMARY,product_name_idx	NULL	NULL	NULL	78789	100.00	Using temporary; Using filesort
1	SIMPLE	o	NULL	ref	product_id_orders	product_id_orders	4	cisc3810_project.p.product_id	4	100.00	NULL

```
JOIN orders_table o ON p.product_id = o.product_id
GROUP BY p.product_id, p.product_name
HAVING COUNT(o.rating) > 20
ORDER BY AVG(o.rating) DESC
LIMIT 10;
*/
EXPLAIN SELECT o.order_id, o.order_quantity, p.product_name, o.order_date
FROM buyers_table b
JOIN orders_table o ON b.buyer_id = o.buyer_id
JOIN products_table p ON p.product_id = o.product_id
WHERE b.first_name = "Jimmy" AND b.last_name = "Tumusiime" AND o.order_date = '2023-12-31';
```

Result Grid

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	b	NULL	index_merge	PRIMARY,first_name_idx,last_name_idx	first_name_idx,last_name_idx	202,202	NULL	1	100.00	Using intersect(first_name_idx,last_nam
1	SIMPLE	o	NULL	ref	buyer_id,product_id_orders	buyer_id	4	cisc3810_project.b.buyer_id	13	10.00	Using where
1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	cisc3810_project.o.product_id	1	100.00	NULL

```sql
EXPLAIN SELECT o.order_id, o.order_quantity, p.product_name, o.order_date
FROM buyers_table b
JOIN orders_table o ON b.buyer_id = o.buyer_id
JOIN products_table p ON p.product_id = o.product_id
WHERE b.first_name = "Jimmy" AND b.last_name = "Tumusiime" AND o.order_date = '2023-12-31';
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	b	NULL	index_merge	PRIMARY,first_name_idx,last_name_idx	first_name_idx,last_name_idx	202,202	NULL	1	100.00	Using intersect(first_name_idx,last_
1	SIMPLE	o	NULL	ref	buyer_id,product_id_orders,order_date_idx	buyer_id	4	cisc3810_project.b.buyer_id	13	0.38	Using where
1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	cisc3810_project.o.product_id	1	100.00	NULL

Limit to 1000 rows

```sql
*/

-- CREATE VIEW top_five_buyer_cities AS
EXPLAIN SELECT a.city,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_amount_spent
 FROM buyers_table b
 JOIN address_table a ON b.buyer_id = a.buyer_id
 JOIN orders_table o ON b.buyer_id = o.buyer_id
 JOIN products_table p ON p.product_id = o.product_id
 GROUP BY a.city
 ORDER BY SUM(o.order_quantity * p.product_price) DESC
 LIMIT 5;
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	b	NULL	index	PRIMARY	first_name_idx	202	NULL	28711	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	a	NULL	ref	buyer_id	buyer_id	4	cisc3810_project.b.buyer_id	1	100.00	NULL
1	SIMPLE	o	NULL	ref	buyer_id,product_id_orders	buyer_id	4	cisc3810_project.b.buyer_id	13	100.00	NULL
1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	cisc3810_project.o.product_id	1	100.00	NULL

```sql

-- CREATE VIEW top_five_buyer_cities AS
EXPLAIN SELECT a.city,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_amount_spent
 FROM buyers_table b
 JOIN address_table a ON b.buyer_id = a.buyer_id
 JOIN orders_table o ON b.buyer_id = o.buyer_id
 JOIN products_table p ON p.product_id = o.product_id
 GROUP BY a.city
 ORDER BY SUM(o.order_quantity * p.product_price) DESC
 LIMIT 5;
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	b	NULL	index	PRIMARY	first_name_idx	202	NULL	28711	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	a	NULL	ref	buyer_id,city_idx	buyer_id	4	cisc3810_project.b.buyer_id	1	100.00	NULL
1	SIMPLE	o	NULL	ref	buyer_id,product_id_orders	buyer_id	4	cisc3810_project.b.buyer_id	13	100.00	NULL
1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	cisc3810_project.o.product_id	1	100.00	NULL

```
37   ORDER BY SUM(o.order_quantity * p.product_price) DESC
38   LIMIT 5;
39  */
40  EXPLAIN SELECT CONCAT(YEAR(o.order_date), '-', DATE_FORMAT(o.order_date, '%m')) AS month_and_year,
41       CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2)) AS total_sales
42    FROM products_table p
43   JOIN orders_table o ON p.product_id = o.product_id
44     WHERE MONTH(o.order_date) = 09
45     AND YEAR(o.order_date) = 2022
46    GROUP BY month_and_year
47
48  -- SELECT s.seller_id,s.seller_name,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_sales,
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	p	NULL	index	PRIMARY	product_price_idx	4	NULL	78789	100.00	Using index; Using temporary
1	SIMPLE	o	NULL	ref	product_id_orders,order_date_idx	product_id_orders	4	cisc3810_project.p.product_id	4	100.00	Using where

```
46     AND YEAR(o.order_date) = 2022
47    GROUP BY month_and_year
48  */
49  EXPLAIN SELECT s.seller_id,s.seller_name,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_sales,
50   CASE
51       WHEN SUM(o.order_quantity * p.product_price) >= 10000000 THEN 'High'
52       WHEN SUM(o.order_quantity * p.product_price) >= 1000000 AND SUM(o.order_quantity * p.product_price) < 10000000 THEN 'Medium'
53       WHEN SUM(o.order_quantity * p.product_price) < 1000000 THEN 'Low'
54      END AS sales_tier
55    FROM sellers_table s
56    JOIN products_table p ON s.seller_id = p.seller_id
57    JOIN orders_table o ON p.product_id = o.product_id
```

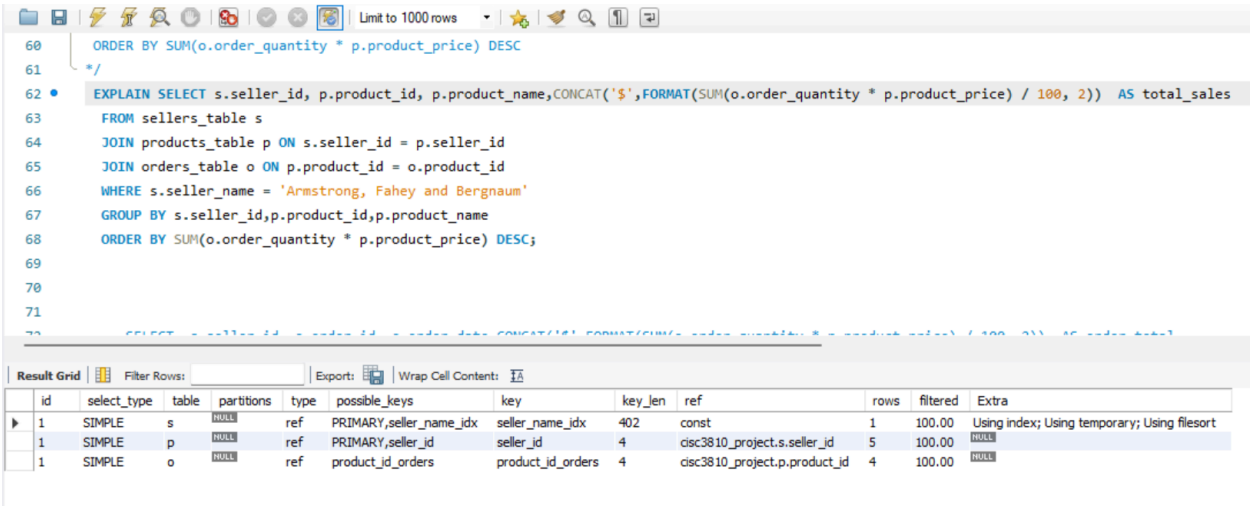
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	s	NULL	ALL	PRIMARY	NULL	NULL	NULL	14581	100.00	Using temporary; Using filesort
1	SIMPLE	p	NULL	ref	PRIMARY,seller_id	seller_id	4	cisc3810_project.s.seller_id	5	100.00	NULL
1	SIMPLE	o	NULL	ref	product_id_orders	product_id_orders	4	cisc3810_project.p.product_id	4	100.00	NULL

```
49  EXPLAIN SELECT s.seller_id,s.seller_name,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_sales,
50  CASE
51      WHEN SUM(o.order_quantity * p.product_price) >= 10000000 THEN 'High'
52      WHEN SUM(o.order_quantity * p.product_price) >= 1000000 AND SUM(o.order_quantity * p.product_price) < 10000000 THEN 'Medium'
53      WHEN SUM(o.order_quantity * p.product_price) < 1000000 THEN 'Low'
54     END AS sales_tier
55  FROM sellers_table s
56  JOIN products_table p ON s.seller_id = p.seller_id
57  JOIN orders_table o ON p.product_id = o.product_id
58  GROUP BY s.seller_id, s.seller_name
59  ORDER BY SUM(o.order_quantity * p.product_price) DESC
60
```

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

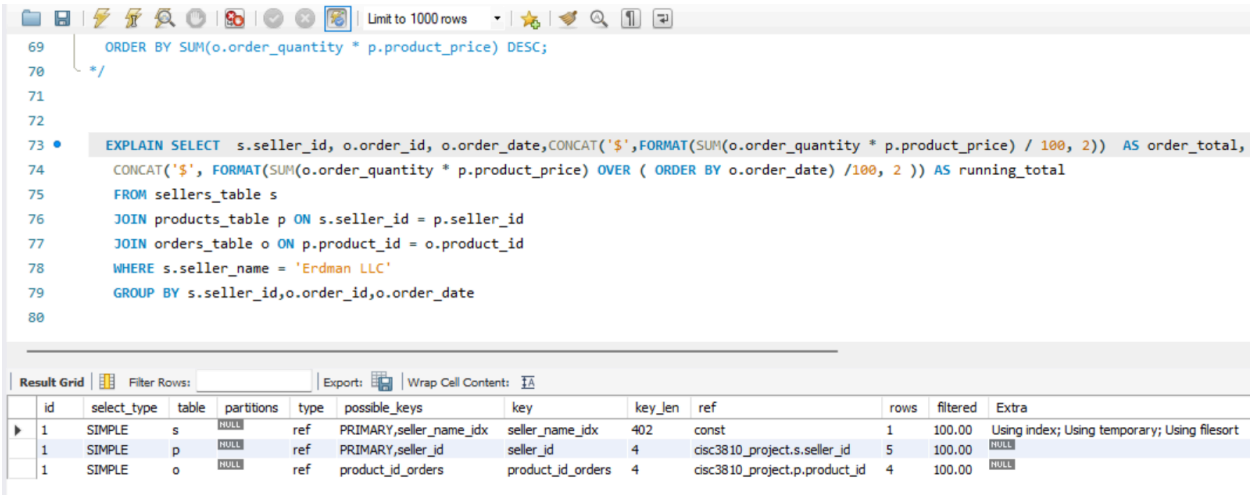
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	s	NULL	index	PRIMARY,seller_name_idx	seller_name_idx	402	NULL	14581	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	p	NULL	ref	PRIMARY,seller_id	seller_id	4	cisc3810_project.s.seller_id	5	100.00	NULL
1	SIMPLE	o	NULL	ref	product_id_orders	product_id_orders	4	cisc3810_project.p.product_id	4	100.00	NULL



```
Limit to 1000 rows
60    ORDER BY SUM(o.order_quantity * p.product_price) DESC
61    */
62    EXPLAIN SELECT s.seller_id, p.product_id, p.product_name,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS total_sales
63     FROM sellers_table s
64     JOIN products_table p ON s.seller_id = p.seller_id
65     JOIN orders_table o ON p.product_id = o.product_id
66     WHERE s.seller_name = 'Armstrong, Fahey and Bergnaum'
67     GROUP BY s.seller_id,p.product_id,p.product_name
68     ORDER BY SUM(o.order_quantity * p.product_price) DESC;
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	s	NULL	ref	PRIMARY,seller_name_idx	seller_name_idx	402	const	1	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	p	NULL	ref	PRIMARY,seller_id	seller_id	4	cisc3810_project.s.seller_id	5	100.00	NULL
1	SIMPLE	o	NULL	ref	product_id_orders	product_id_orders	4	cisc3810_project.p.product_id	4	100.00	NULL



```
Limit to 1000 rows
69     ORDER BY SUM(o.order_quantity * p.product_price) DESC;
70    */
71
72
73    EXPLAIN SELECT  s.seller_id, o.order_id, o.order_date,CONCAT('$',FORMAT(SUM(o.order_quantity * p.product_price) / 100, 2))  AS order_total,
74      CONCAT('$', FORMAT(SUM(o.order_quantity * p.product_price) OVER ( ORDER BY o.order_date) /100, 2 )) AS running_total
75      FROM sellers_table s
76      JOIN products_table p ON s.seller_id = p.seller_id
77      JOIN orders_table o ON p.product_id = o.product_id
78      WHERE s.seller_name = 'Erdman LLC'
79      GROUP BY s.seller_id,o.order_id,o.order_date
80
```

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	s	NULL	ref	PRIMARY,seller_name_idx	seller_name_idx	402	const	1	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	p	NULL	ref	PRIMARY,seller_id	seller_id	4	cisc3810_project.s.seller_id	5	100.00	NULL
1	SIMPLE	o	NULL	ref	product_id_orders	product_id_orders	4	cisc3810_project.p.product_id	4	100.00	NULL

WRITTEN JUSTIFICATION FOR EACH SCHEMA:
Buyers_table- the first and last name needed to be associated with the buyer, same with the email because how else would you identify the buyer. Also there was a column in the denormalized table dedicated to seller names so it made sense to put these into a buyer table, same with the buyer_id, it needed to be placed in a table dedicated to buyer info.

CC_table- the cc_number and cc_exp columns were directly associated with each other and as such needed to be placed in a table that handled just credit card info, I added a cc_id to link each distinct cc_number and exp to each other.

Address_table- at first i created separate tables for address, city, and country, but realized that it would make things alot easier if they were put into the same table also, follows 3nf as none of the none primary key columns rely on each other since different countries can have cities with the same name and vice versa. I created an address_id to link each address to its corresponding city and country. Also I added buyer_id as a foreign key due to the fact that an address is linked to one buyer.

Sellers_table- seller_id, seller_name and seller_country needed to be moved to a table that handled seller info specifically.

Products_table- same here with any column product related, I also added seller_id as a foreign key because a product is linked to a specific seller, but a seller can sell multiple products of different ids, showing a one to many relationship.

Orders_table- So pretty much any column that started with the word order got stuffed in this table for handling order info in just the orders table, I also put the rating and review columns in this table because of the fact that every order id has 1 rating and 1 review associated with it, according to my data. I also included a host of foreign key ids that would be relevant towards an order such as having the buyer, seller, cc, and product info related to the order.