Final project

Advanced SQL: MySQL for Ecommerce Data Analysis

Maribel Diaz Bajo

Data set: mavenfuzzyfactory

Table 1: order_item_refunds

order_item_refund_id	created_at	order_item_id	order_id	refund_amount_usd

Table 2: order_items

order_item_id	created_at	order_id	product_id	is_primary_item	price_usd	cogs_usd

Table 3: order_items

order_id	created_at	website_session_id	user_id	primary_product_id	items_purchased	price_usd	cogs_usd

Table 4: products

product_id	created_at	product_name

Table 5: website_pageviews

website_pageview_id	created_at	website_session_id	pageview_url

Table 6: website_sessions

website_session_id	created_at	user_id	is_repeat_session	utm_source

utm_campaign	utm_content	device_type	http_referer

#1

Overall session and order volume, trended by quarter for the life of the business

```sql
-- 1. Get first and last sessions' date
SELECT
    MIN(created_at),
    MAX(created_at)
FROM website_sessions;
```

MIN(created_at)	MAX(created_at)
2012-03-19 09:04:16	2015-03-19 08:59:08

```sql
/* 2. Aiming for a fair QoQ comparison, I considered only full quarters,
starting on '2012-04-01' and finishing on '2014-12-31'. */
SELECT
    YEAR(website_sessions.created_at) AS year,
    QUARTER(website_sessions.created_at) AS quarter,
    COUNT(DISTINCT website_sessions.website_session_id) AS sessions,
    COUNT(DISTINCT order_id) AS orders
FROM website_sessions
    LEFT JOIN orders
        ON orders.website_session_id = website_sessions.website_session_id
WHERE website_sessions.created_at BETWEEN '2012-04-01' AND '2015-01-01'
GROUP BY year, quarter;
```

year	quarter	sessions	orders
2012	2	11433	347
2012	3	16886	684
2012	4	32274	1495
2013	1	19828	1273
2013	2	24737	1717
2013	3	27665	1841
2013	4	40551	2616
2014	1	46759	3068
2014	2	53125	3848
2014	3	57146	4036
2014	4	76392	5908

#2

Quarterly figures since launching, for session to order conversion rate, revenue per order, and revenue per session.

```sql
SELECT
    YEAR(website_sessions.created_at) AS year,
    QUARTER(website_sessions.created_at) AS quarter,
    COUNT(DISTINCT order_id)/
        COUNT(DISTINCT website_sessions.website_session_id) AS conv_rate,
    SUM(price_usd)/COUNT(DISTINCT order_id) AS revenue_per_order,
    SUM(price_usd)/
        COUNT(DISTINCT website_sessions.website_session_id) revenue_per_session
FROM website_sessions
    LEFT JOIN orders
        ON orders.website_session_id = website_sessions.website_session_id
WHERE website_sessions.created_at BETWEEN '2012-04-01' AND '2015-01-01'
GROUP BY year, quarter;
```

year	quarter	conv_rate	revenue_per_order	revenue_per_session
2012	2	0.0304	49.990000	1.517233
2012	3	0.0405	49.990000	2.024941
2012	4	0.0463	49.990000	2.315643
2013	1	0.0642	52.142396	3.347653
2013	2	0.0694	51.539214	3.577347
2013	3	0.0665	51.733585	3.442672
2013	4	0.0645	54.715688	3.529783
2014	1	0.0656	62.145098	4.077529
2014	2	0.0724	64.389797	4.663942
2014	3	0.0706	64.491355	4.554774
2014	4	0.0773	63.793497	4.933658

#3.1

Quarterly view of orders from Gsearch nonbrand, Bsearch nonbrand, brand search overall, organic search, and direct type in

```
-- Get distinct utm source, campaign and referrers
SELECT DISTINCT
    utm_source,
    utm_campaign,
    http_referer
FROM website_sessions;
```

utm_source	utm_campaign	http_referer
gsearch	nonbrand	https://www.gsearch.com
NULL	NULL	NULL
gsearch	brand	https://www.gsearch.com
NULL	NULL	https://www.gsearch.com
bsearch	brand	https://www.bsearch.com
NULL	NULL	https://www.bsearch.com
bsearch	nonbrand	https://www.bsearch.com
socialbook	pilot	https://www.socialbook.com
socialbook	desktop_targeted	https://www.socialbook.com

#3.2

Based on the previous, to get the orders for each channel I need to consider the following:

- If e verything is NULL, the channel is direct type in
- If only the referer is NULL, the channel is organic search
- If utm_campaing is brand, the channel is brand
- If utm_campaing is nonbrand, and utm_source is gsearch, the channel is gsearh nonbrand
- If utm_campaing is nonbrand, and utm_source is bsearch, the channel is bserach nonbrand

```sql
SELECT
    YEAR(website_sessions.created_at) AS year,
    QUARTER(website_sessions.created_at) AS quarter,
    COUNT(DISTINCT CASE WHEN utm_source = 'gsearch' AND utm_campaign = 'nonbrand' THEN order_id ELSE NULL END) AS orders_gsearch_nonb,
    COUNT(DISTINCT CASE WHEN utm_source = 'bsearch' AND utm_campaign = 'nonbrand' THEN order_id ELSE NULL END) AS orders_bsearch_nonb,
    COUNT(DISTINCT CASE WHEN utm_campaign = 'brand' THEN order_id ELSE NULL END) AS orders_brand_search,
    COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IS NOT NULL THEN order_id ELSE NULL END) AS orders_organic,
    COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IS NULL THEN order_id ELSE NULL END) AS orders_direct_type_in
FROM website_sessions
    LEFT JOIN orders
        ON orders.website_session_id = website_sessions.website_session_id
WHERE website_sessions.created_at BETWEEN '2012-04-01' AND '2015-01-01'
GROUP BY year, quarter;
```

year	quarter	orders_gsearch_nonb	orders_bsearch_nonb	orders_brand_search	orders_organic	orders_direct_type_in
2012	2	291	0	20	15	21
2012	3	482	82	48	40	32
2012	4	913	311	88	94	89
2013	1	766	183	108	125	91
2013	2	1114	237	113	134	119
2013	3	1132	245	154	167	143
2013	4	1657	291	248	223	197
2014	1	1666	344	354	338	311
2014	2	2208	427	410	436	367
2014	3	2260	434	432	445	402
2014	4	3248	683	615	605	532

#4

Overall session to order conversion rate trends for those same channels, by quarter

```sql
SELECT
    YEAR(website_sessions.created_at) AS year,
    QUARTER(website_sessions.created_at) AS quarter,
    COUNT(DISTINCT CASE WHEN utm_source = 'gsearch' AND utm_campaign = 'nonbrand' THEN order_id ELSE NULL END)/
        COUNT(DISTINCT CASE WHEN utm_source = 'gsearch' AND utm_campaign = 'nonbrand' THEN website_sessions.website_session_id ELSE NULL END) AS gsearch_nonb_conv_rt,
    COUNT(DISTINCT CASE WHEN utm_source = 'bsearch' AND utm_campaign = 'nonbrand' THEN order_id ELSE NULL END)/
        COUNT(DISTINCT CASE WHEN utm_source = 'bsearch' AND utm_campaign = 'nonbrand' THEN website_sessions.website_session_id ELSE NULL END) AS bsearch_nonb_conv_rt,
    COUNT(DISTINCT CASE WHEN utm_campaign = 'brand' THEN order_id ELSE NULL END)/
        COUNT(DISTINCT CASE WHEN utm_campaign = 'brand' THEN website_sessions.website_session_id ELSE NULL END) AS brand_search_conv_rt,
    COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IS NOT NULL THEN order_id ELSE NULL END)/
        COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IS NOT NULL THEN website_sessions.website_session_id ELSE NULL END) AS organic_conv_rt,
    COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IS NULL THEN order_id ELSE NULL END)/
        COUNT(DISTINCT CASE WHEN utm_source IS NULL AND http_referer IS NULL THEN website_sessions.website_session_id ELSE NULL END) AS direct_type_in_conv_rt
FROM website_sessions
    LEFT JOIN orders
        ON orders.website_session_id = website_sessions.website_session_id
WHERE website_sessions.created_at BETWEEN '2012-04-01' AND '2015-01-01'
GROUP BY year, quarter;
```

year	quarter	gsearch_nonb_conv_rt	bsearch_nonb_conv_rt	brand_search_conv_rt	organic_conv_rt	direct_type_in_conv_rt
2012	2	0.0284	NULL	0.0526	0.0359	0.0536
2012	3	0.0384	0.0409	0.0602	0.0498	0.0443
2012	4	0.0436	0.0497	0.0531	0.0539	0.0537
2013	1	0.0612	0.0693	0.0704	0.0753	0.0614
2013	2	0.0686	0.0691	0.0673	0.0760	0.0736
2013	3	0.0639	0.0696	0.0708	0.0734	0.0719
2013	4	0.0629	0.0601	0.0801	0.0693	0.0647
2014	1	0.0693	0.0704	0.0840	0.0756	0.0765
2014	2	0.0702	0.0695	0.0803	0.0798	0.0738
2014	3	0.0703	0.0698	0.0756	0.0732	0.0702
2014	4	0.0782	0.0841	0.0812	0.0784	0.0748

The biggest jump in conversion rate for all the channels was between the 4th quarter of 2012 and the first of 2013 - around 2% for each channel. At the beginning of January 2013 the second product was introduced.

#5.1

Monthly trending for revenue and margin by product, along with total revenue and margin

```
/* Aiming for a fair comparison considering only full months,
hence data between April 2012 and February 2015. */

-- Get the amount of different products and respective IDs.
SELECT DISTINCT
    product_id
FROM order_items
WHERE created_at BETWEEN '2012-04-01' AND '2015-03-01';
```

product_id
1
2
3
4

```
SELECT
    YEAR(created_at) AS year,
    MONTHNAME(created_at) AS month,
    SUM(price_usd) AS total_revenue,
    SUM(price_usd - cogs_usd) AS total_margin,
    SUM(CASE WHEN product_id = 1 THEN price_usd ELSE NULL END) AS revenue_prod_1,
    SUM(CASE WHEN product_id = 1 THEN price_usd ELSE NULL END) - SUM(CASE WHEN product_id = 1 THEN cogs_usd ELSE NULL END) AS margin_prod_1,
    SUM(CASE WHEN product_id = 2 THEN price_usd ELSE NULL END) AS revenue_prod_2,
    SUM(CASE WHEN product_id = 2 THEN price_usd ELSE NULL END) - SUM(CASE WHEN product_id = 2 THEN cogs_usd ELSE NULL END) AS margin_prod_2,
    SUM(CASE WHEN product_id = 3 THEN price_usd ELSE NULL END) AS revenue_prod_3,
    SUM(CASE WHEN product_id = 3 THEN price_usd ELSE NULL END) - SUM(CASE WHEN product_id = 3 THEN cogs_usd ELSE NULL END) AS margin_prod_3,
    SUM(CASE WHEN product_id = 4 THEN price_usd ELSE NULL END) AS revenue_prod_4,
    SUM(CASE WHEN product_id = 4 THEN price_usd ELSE NULL END) - SUM(CASE WHEN product_id = 4 THEN cogs_usd ELSE NULL END) AS margin_prod_4
FROM order_items
WHERE created_at BETWEEN '2012-04-01' AND '2015-03-01'
GROUP BY 1, 2, MONTH(created_at)
ORDER BY 1, MONTH(created_at);
```

#5.2

year	month	total_revenue	total_margin	revenue_prod_1	margin_prod_1	revenue_prod_2	margin_prod_2	revenue_prod_3	margin_prod_3	revenue_prod_4	margin_prod_4
2012	April	4949.01	3019.50	4949.01	3019.50	NULL	NULL	NULL	NULL	NULL	NULL
2012	May	5348.93	3263.50	5348.93	3263.50	NULL	NULL	NULL	NULL	NULL	NULL
2012	June	7048.59	4300.50	7048.59	4300.50	NULL	NULL	NULL	NULL	NULL	NULL
2012	July	8448.31	5154.50	8448.31	5154.50	NULL	NULL	NULL	NULL	NULL	NULL
2012	August	11397.72	6954.00	11397.72	6954.00	NULL	NULL	NULL	NULL	NULL	NULL
2012	September	14347.13	8753.50	14347.13	8753.50	NULL	NULL	NULL	NULL	NULL	NULL
2012	October	18546.29	11315.50	18546.29	11315.50	NULL	NULL	NULL	NULL	NULL	NULL
2012	November	30893.82	18849.00	30893.82	18849.00	NULL	NULL	NULL	NULL	NULL	NULL
2012	December	25294.94	15433.00	25294.94	15433.00	NULL	NULL	NULL	NULL	NULL	NULL
2013	January	19966.10	12224.00	17146.57	10461.50	2819.53	1762.50	NULL	NULL	NULL	NULL
2013	February	26515.02	16323.00	16796.64	10248.00	9718.38	6075.00	NULL	NULL	NULL	NULL
2013	March	19896.15	12197.50	15996.80	9760.00	3899.35	2437.50	NULL	NULL	NULL	NULL
2013	April	28524.48	17487.00	22945.41	13999.50	5579.07	3487.50	NULL	NULL	NULL	NULL
2013	May	29364.29	17989.50	24445.11	14914.50	4919.18	3075.00	NULL	NULL	NULL	NULL
2013	June	30604.06	18754.00	25144.97	15341.50	5459.09	3412.50	NULL	NULL	NULL	NULL
2013	July	31093.97	19056.50	25394.92	15494.00	5699.05	3562.50	NULL	NULL	NULL	NULL
2013	August	31373.92	19230.00	25494.90	15555.00	5879.02	3675.00	NULL	NULL	NULL	NULL
2013	September	32773.64	20084.00	26894.62	16409.00	5879.02	3675.00	NULL	NULL	NULL	NULL
2013	October	38242.62	23454.00	30143.97	18391.50	8098.65	5062.50	NULL	NULL	NULL	NULL
2013	November	46631.02	28607.00	36192.76	22082.00	10438.26	6525.00	NULL	NULL	NULL	NULL
2013	December	58262.60	36190.00	40891.82	24949.00	10978.17	6862.50	6392.61	4378.50	NULL	NULL
2014	January	56568.89	35366.50	36392.72	22204.00	10978.17	6862.50	9198.00	6300.00	NULL	NULL
2014	February	66012.52	41762.00	29194.16	17812.00	21056.49	13162.50	9703.89	6646.50	6057.98	4141.00
2014	March	68079.75	43000.50	39192.16	23912.00	11518.08	7200.00	11221.56	7686.00	6147.95	4202.50
2014	April	78785.42	49751.00	45840.83	27968.50	12897.85	8062.50	12279.33	8410.50	7767.41	5309.50
2014	May	88909.28	56146.00	51539.69	31445.50	14757.54	9225.00	13705.02	9387.00	8907.03	6088.50
2014	June	80077.24	50622.00	44591.08	27206.00	14697.55	9187.50	13291.11	9103.50	7497.50	5125.00
2014	July	82902.64	52317.00	47940.41	29249.50	14517.58	9075.00	12647.25	8662.50	7797.40	5330.00
2014	August	85151.98	53859.00	48040.39	29310.50	14337.61	8962.50	13567.05	9292.50	9206.93	6293.50
2014	September	92172.50	58272.00	52789.44	32208.00	14997.50	9375.00	14578.83	9985.50	9806.73	6703.50
2014	October	103965.97	65784.50	58638.27	35776.50	17097.15	10687.50	16924.32	11592.00	11306.23	7728.50
2014	November	128162.98	80985.00	72535.49	44255.50	22616.23	14137.50	19545.75	13387.50	13465.51	9204.50
2014	December	144823.02	91857.00	79184.16	48312.00	23216.13	14512.50	24788.61	16978.50	17634.12	12054.00
2015	January	132211.54	83911.00	69586.08	42456.00	23636.06	14775.00	20695.50	14175.00	18293.90	12505.00
2015	February	129212.94	82006.00	55638.87	33946.50	38633.56	24150.00	18625.95	12757.50	16314.56	11152.00

For the years in analysis, revenue is always higher in November and December, which are the months of Black Friday, Cyber Monday, and Christmas.
In the first year, 2012, sales were higher in November than in December; that year Black Friday and Cyber Monday were both in November, which could explain the higher sales.
For the other 2 years, December had higher revenue, which could be explained by the fact that Cyber Monday was in December.
For product 2, there was a spike in sales during February for the three years, which makes sense as it is Valentine's month and product 2 is the-forever-love-bear.

#6.1

Monthly sessions to the /products page, and how the % of those sessions clicking through another page has changed over time, along with a view of how the conversion from /products to placing an order has improved

```sql
-- 1. Create temporary table with the sessions that got to product, corresponding pageview, and created time
CREATE TEMPORARY TABLE products_session_pageview
SELECT
    website_session_id,
    website_pageview_id,
    created_at
FROM website_pageviews
WHERE created_at BETWEEN '2012-04-01' AND '2015-03-01'
    AND pageview_url = '/products';
```

website_session_id	website_pageview_id	created_at
1878	3716	2012-04-01 00:36:51
1879	3718	2012-04-01 00:58:43
1884	3728	2012-04-01 05:48:32
1885	3730	2012-04-01 05:59:54
1886	3734	2012-04-01 07:05:14
1887	3738	2012-04-01 07:10:54
1889	3743	2012-04-01 07:34:20
1890	3746	2012-04-01 07:35:58
1893	3754	2012-04-01 09:18:09

```sql
-- 2. Create a new temp table that adds to the previous table the pageview after /products for each session
CREATE TEMPORARY TABLE products_session_pgv_next_pgv
SELECT
    products_session_pageview.website_session_id,
    products_session_pageview.website_pageview_id,
    products_session_pageview.created_at,
    MIN(website_pageviews.website_pageview_id) AS pageview_after_products
FROM products_session_pageview
    LEFT JOIN website_pageviews
        ON website_pageviews.website_session_id = products_session_pageview.website_session_id
            AND website_pageviews.website_pageview_id > products_session_pageview.website_pageview_id
GROUP BY 1,2,3;
```

website_session_id	website_pageview_id	created_at	pageview_after_products
1878	3716	2012-04-01 00:36:51	NULL
1879	3718	2012-04-01 00:58:43	3719
1884	3728	2012-04-01 05:48:32	NULL
1885	3730	2012-04-01 05:59:54	3731
1886	3734	2012-04-01 07:05:14	3736
1887	3738	2012-04-01 07:10:54	3739

#6.2

```sql
SELECT
    YEAR(products_session_pgv_next_pgv.created_at) AS year,
    MONTHNAME(products_session_pgv_next_pgv.created_at) AS month,
    COUNT(DISTINCT products_session_pgv_next_pgv.website_session_id) AS product_sessions,
    COUNT(DISTINCT pageview_after_products)/
        COUNT(DISTINCT products_session_pgv_next_pgv.website_session_id) AS pct_clicking_through,
    COUNT(DISTINCT order_id)/COUNT(DISTINCT products_session_pgv_next_pgv.website_session_id) AS conv_prod_to_order
FROM products_session_pgv_next_pgv
    LEFT JOIN orders
        ON orders.website_session_id = products_session_pgv_next_pgv.website_session_id
GROUP BY 1, 2, MONTH(products_session_pgv_next_pgv.created_at)
ORDER BY 1, MONTH(products_session_pgv_next_pgv.created_at);
```

year	month	product_sessions	pct_clicking_through	conv_prod_to_order
2012	April	1447	0.7104	0.0684
2012	May	1584	0.7165	0.0682
2012	June	1752	0.7118	0.0799
2012	July	2019	0.7132	0.0837
2012	August	3011	0.7297	0.0757
2012	September	3125	0.7219	0.0918
2012	October	4031	0.7318	0.0920
2012	November	6745	0.7191	0.0916
2012	December	5013	0.7221	0.1009
2013	January	3380	0.7678	0.1157
2013	February	3685	0.7607	0.1349
2013	March	3369	0.7640	0.1143
2013	April	4362	0.7694	0.1265
2013	May	4680	0.7705	0.1220
2013	June	4598	0.7686	0.1292
2013	July	5022	0.7750	0.1201
2013	August	5229	0.7560	0.1163
2013	September	5396	0.7543	0.1168
2013	October	6042	0.7560	0.1172
2013	November	7888	0.7481	0.1092
2013	December	8840	0.7948	0.1184
2014	January	7790	0.8199	0.1262
2014	February	7960	0.8147	0.1283
2014	March	8100	0.8223	0.1314
2014	April	9743	0.8166	0.1274
2014	May	10264	0.8250	0.1333
2014	June	10005	0.8252	0.1238
2014	July	10834	0.8264	0.1183
2014	August	10775	0.8342	0.1234
2014	September	11127	0.8227	0.1280
2014	October	12341	0.8299	0.1304
2014	November	14481	0.8303	0.1371
2014	December	17240	0.8474	0.1342
2015	January	15217	0.8538	0.1379
2015	February	14373	0.8479	0.1438

#7

Orders and cross-sell data since the 4th product was made available as a primary product (December 05, 2014)

```sql
SELECT
    primary_product_id,
    COUNT(DISTINCT orders.order_id) AS orders,
    COUNT(DISTINCT CASE WHEN order_items.product_id = 1 AND is_primary_item = 0 THEN order_item_id ELSE NULL END) AS p1_xsell,
    COUNT(DISTINCT CASE WHEN order_items.product_id = 2 AND is_primary_item = 0 THEN order_item_id ELSE NULL END) AS p2_xsell,
    COUNT(DISTINCT CASE WHEN order_items.product_id = 3 AND is_primary_item = 0 THEN order_item_id ELSE NULL END) AS p3_xsell,
    COUNT(DISTINCT CASE WHEN order_items.product_id = 4 AND is_primary_item = 0 THEN order_item_id ELSE NULL END) AS p4_xsell,
    COUNT(DISTINCT CASE WHEN order_items.product_id = 1 AND is_primary_item = 0 THEN order_item_id ELSE NULL END)/
        COUNT(DISTINCT orders.order_id) p1_xsell_rt,
    COUNT(DISTINCT CASE WHEN order_items.product_id = 2 AND is_primary_item = 0 THEN order_item_id ELSE NULL END)/
        COUNT(DISTINCT orders.order_id) p2_xsell_rt,
    COUNT(DISTINCT CASE WHEN order_items.product_id = 3 AND is_primary_item = 0 THEN order_item_id ELSE NULL END)/
        COUNT(DISTINCT orders.order_id) p3_xsell_rt,
    COUNT(DISTINCT CASE WHEN order_items.product_id = 4 AND is_primary_item = 0 THEN order_item_id ELSE NULL END)/
        COUNT(DISTINCT orders.order_id) p4_xsell_rt
    FROM orders
        LEFT JOIN order_items
            ON order_items.order_id = orders.order_id
    WHERE orders.created_at > '2014-12-05'
    GROUP BY primary_product_id;
```

primary_product_id	orders	p1_xsell	p2_xsell	p3_xsell	p4_xsell	p1_xsell_rt	p2_xsell_rt	p3_xsell_rt	p4_xsell_rt
1	4467	0	238	553	933	0.0000	0.0533	0.1238	0.2089
2	1277	25	0	40	260	0.0196	0.0000	0.0313	0.2036
3	929	84	40	0	208	0.0904	0.0431	0.0000	0.2239
4	581	16	9	22	0	0.0275	0.0155	0.0379	0.0000

```sql
-- Getting first appearance of each product to add context

SELECT
    pageview_url AS product,
    MIN(created_at) AS first_appearance
FROM website_pageviews
WHERE pageview_url IN ('/the-birthday-sugar-panda', '/the-forever-love-bear',
    '/the-hudson-river-mini-bear', '/the-original-mr-fuzzy')
GROUP BY pageview_url
ORDER BY 2;
```

product	first_appearance
/the-original-mr-fuzzy	2012-03-19 10:10:52
/the-forever-love-bear	2013-01-06 13:27:48
/the-birthday-sugar-panda	2013-12-12 13:06:48
/the-hudson-river-mini-bear	2014-12-05 14:28:49

Product 4 shows the lowest number of orders for the period, which makes sense as it was made available as a primary product for the first time, while the rest had been there for at least one year more. However, we can see it cross-sells well, with the highest cross-selling rates of all the products. The opposite happens to product 1, showing the highest number of orders for the period but lower cross-selling rates when compared with products 4 and even 3. Product 2 behavior is difficult to compare with the others as we have Valentine's Day in the period in analysis, and this product was launched for that date a couple of years back and targets that festivity. It is likely that the orders for product 2 as a primary product and the cross-selling rates would fall in the following months. It would be interesting to analyze cross-selling data for the full year.