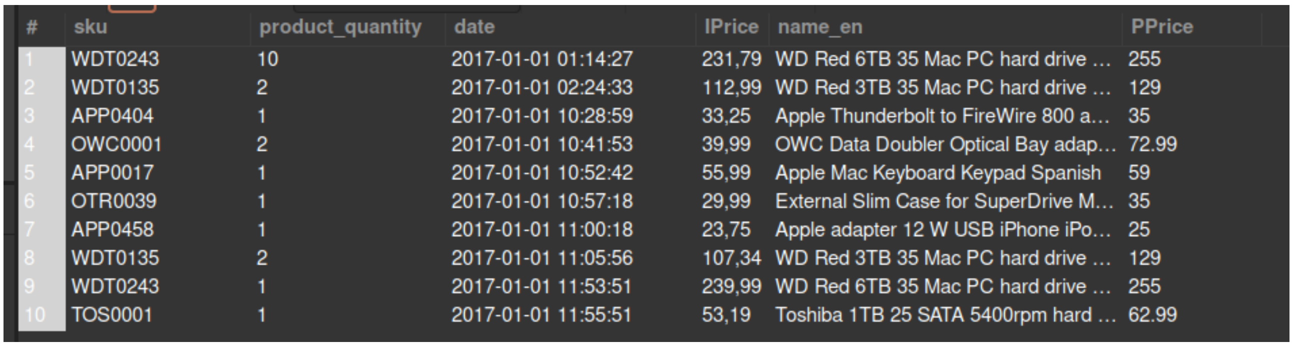
SQL CRASH COURSE PART 2
Query the database joining tables

-- ---
-- Query 1 Our first query should return the "sku", "product_quantity", "date" and "unit_price" from the line_item table together with the "name" and the "price" of each product from the "products" table. We want only products present in both tables.

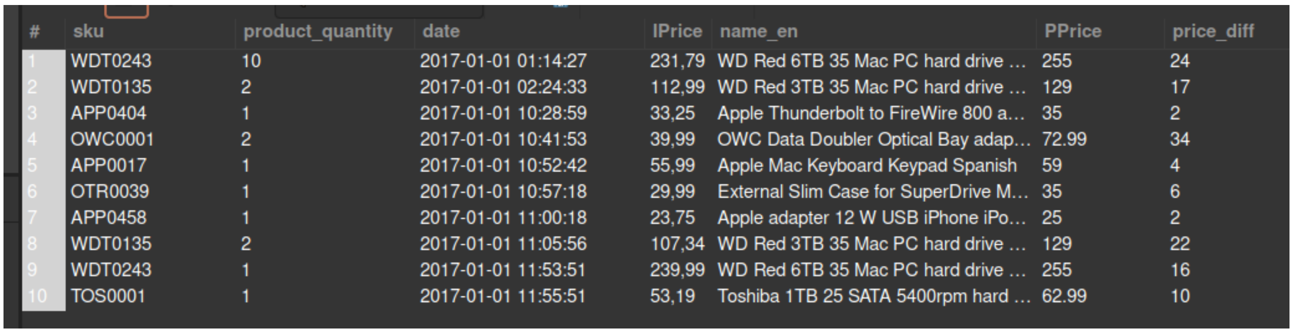
```
SELECT
   itm.sku,
   itm.product_quantity,
   itm.date,
   itm.unit_price AS IPrice,
   prd.name_en,
   prd.price AS PPrice
FROM
   line_item itm
     INNER JOIN
   products prd ON itm.sku = prd.sku;
```



#	sku	product_quantity	date	IPrice	name_en	PPrice
1	WDT0243	10	2017-01-01 01:14:27	231,79	WD Red 6TB 35 Mac PC hard drive ...	255
2	WDT0135	2	2017-01-01 02:24:33	112,99	WD Red 3TB 35 Mac PC hard drive ...	129
3	APP0404	1	2017-01-01 10:28:59	33,25	Apple Thunderbolt to FireWire 800 a...	35
4	OWC0001	2	2017-01-01 10:41:53	39,99	OWC Data Doubler Optical Bay adap...	72.99
5	APP0017	1	2017-01-01 10:52:42	55,99	Apple Mac Keyboard Keypad Spanish	59
6	OTR0039	1	2017-01-01 10:57:18	29,99	External Slim Case for SuperDrive M...	35
7	APP0458	1	2017-01-01 11:00:18	23,75	Apple adapter 12 W USB iPhone iPo...	25
8	WDT0135	2	2017-01-01 11:05:56	107,34	WD Red 3TB 35 Mac PC hard drive ...	129
9	WDT0243	1	2017-01-01 11:53:51	239,99	WD Red 6TB 35 Mac PC hard drive ...	255
10	TOS0001	1	2017-01-01 11:55:51	53,19	Toshiba 1TB 25 SATA 5400rpm hard ...	62.99

-- ---
-- Query 2 You might notice that the unit_price from the line_item table and the price from the product table is not the same. Let's investigate that! Extend your previous query by adding a column with the difference in price. Name that column price_difference .

```
SELECT
   itm.sku,
   itm.product_quantity,
   itm.date,
   itm.unit_price AS IPrice,
   prd.name_en,
   prd.price AS PPrice,
   ROUND(prd.price - itm.unit_price, 1) AS price_diff
FROM
   line_item itm
      INNER JOIN
   products prd ON itm.sku = prd.sku;
```

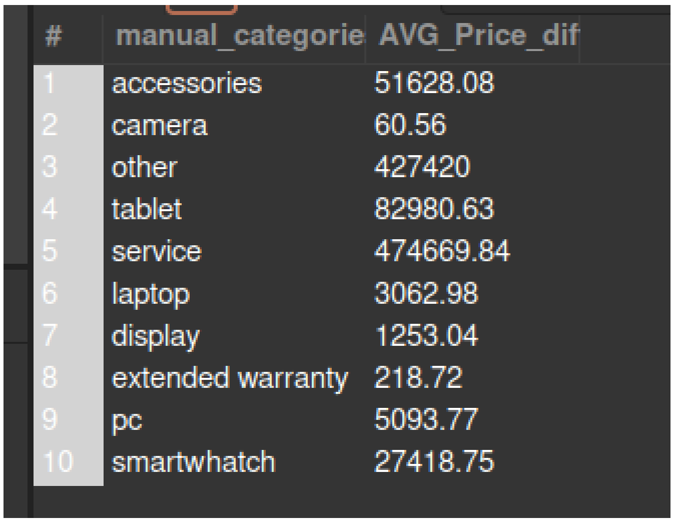


#	sku	product_quantity	date	lPrice	name_en	PPrice	price_diff
1	WDT0243	10	2017-01-01 01:14:27	231,79	WD Red 6TB 35 Mac PC hard drive ...	255	24
2	WDT0135	2	2017-01-01 02:24:33	112,99	WD Red 3TB 35 Mac PC hard drive ...	129	17
3	APP0404	1	2017-01-01 10:28:59	33,25	Apple Thunderbolt to FireWire 800 a...	35	2
4	OWC0001	2	2017-01-01 10:41:53	39,99	OWC Data Doubler Optical Bay adap...	72.99	34
5	APP0017	1	2017-01-01 10:52:42	55,99	Apple Mac Keyboard Keypad Spanish	59	4
6	OTR0039	1	2017-01-01 10:57:18	29,99	External Slim Case for SuperDrive M...	35	6
7	APP0458	1	2017-01-01 11:00:18	23,75	Apple adapter 12 W USB iPhone iPo...	25	2
8	WDT0135	2	2017-01-01 11:05:56	107,34	WD Red 3TB 35 Mac PC hard drive ...	129	22
9	WDT0243	1	2017-01-01 11:53:51	239,99	WD Red 6TB 35 Mac PC hard drive ...	255	16
10	TOS0001	1	2017-01-01 11:55:51	53,19	Toshiba 1TB 25 SATA 5400rpm hard ...	62.99	10

-- ---

-- Query 3 Build a query that outputs the price difference that you just calculated, grouping products by category. Round the result.

```
SELECT
          inn.manual_categories,
   ROUND(AVG(inn.price_dif),2) as AVG_Price_diff
FROM
          (SELECT
                    line_item.sku,
                    line_item.product_quantity,
                    line_item.date,
                    line_item.unit_price,
                    products.name_en,
      products.price,
                    ROUND(ABS(line_item.unit_price - products.price), 1) AS price_dif,
                    products.manual_categories
          FROM line_item
                    INNER JOIN products
                    ON line_item.sku = products.sku)  as inn
GROUP BY inn.manual_categories;
```

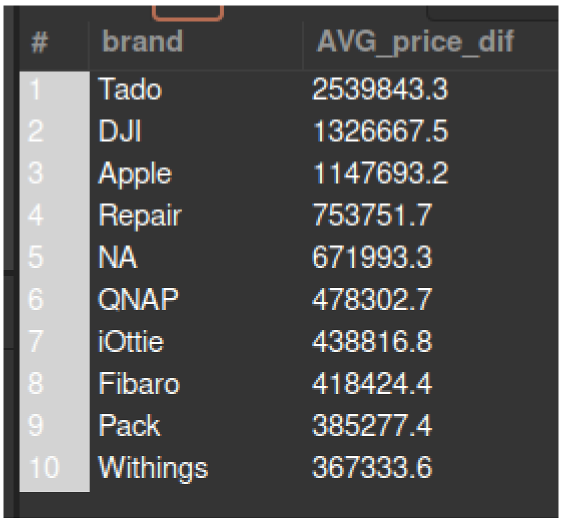


#	manual_categorie	AVG_Price_dif
1	accessories	51628.08
2	camera	60.56
3	other	427420
4	tablet	82980.63
5	service	474669.84
6	laptop	3062.98
7	display	1253.04
8	extended warranty	218.72
9	pc	5093.77
10	smartwhatch	27418.75

-- ---

-- Query 4. Create the same query as before (calculating the price difference between the line_item and the products tables, but now grouping by brands instead of categories.

```
SELECT
          inn.brand,
   ROUND(AVG(inn.price_dif),1) as AVG_price_dif
FROM
        (SELECT
                itm.sku,
                itm.product_quantity,
                itm.date,
                prd.name_en,
                itm.unit_price AS IPrice,
                prd.brand,
                prd.price AS PPrice,
                ROUND(prd.price - itm.unit_price, 1) AS price_dif
        FROM
                line_item itm
                        INNER JOIN
                products prd ON itm.sku = prd.sku ) AS inn
GROUP BY inn.brand
ORDER BY AVG_price_dif DESC;
```

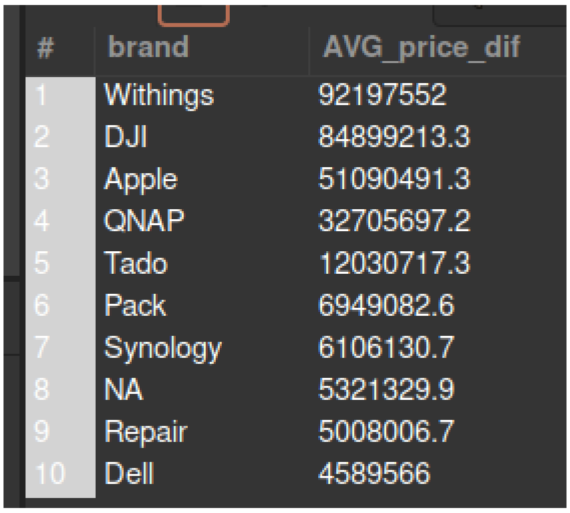


#	brand	AVG_price_dif
1	Tado	2539843.3
2	DJI	1326667.5
3	Apple	1147693.2
4	Repair	753751.7
5	NA	671993.3
6	QNAP	478302.7
7	iOttie	438816.8
8	Fibaro	418424.4
9	Pack	385277.4
10	Withings	367333.6

-- ---
-- Query 5. Let's focus on the brands with a big price difference: run the same query as before, but now limiting the results to only brands with an avg_price_dif of more than 50000. Order the results by avg_price_dif (bigger to smaller).

```
SELECT
        inn.brand,
   ROUND(AVG(inn.price_dif),1) as AVG_price_dif
FROM
        (SELECT
                itm.sku,
                itm.product_quantity,
                itm.date,
                prd.name_en,
                itm.unit_price AS IPrice,
                prd.brand,
                prd.price AS PPrice,
                ROUND(prd.price - itm.unit_price, 1) AS price_dif
        FROM
```

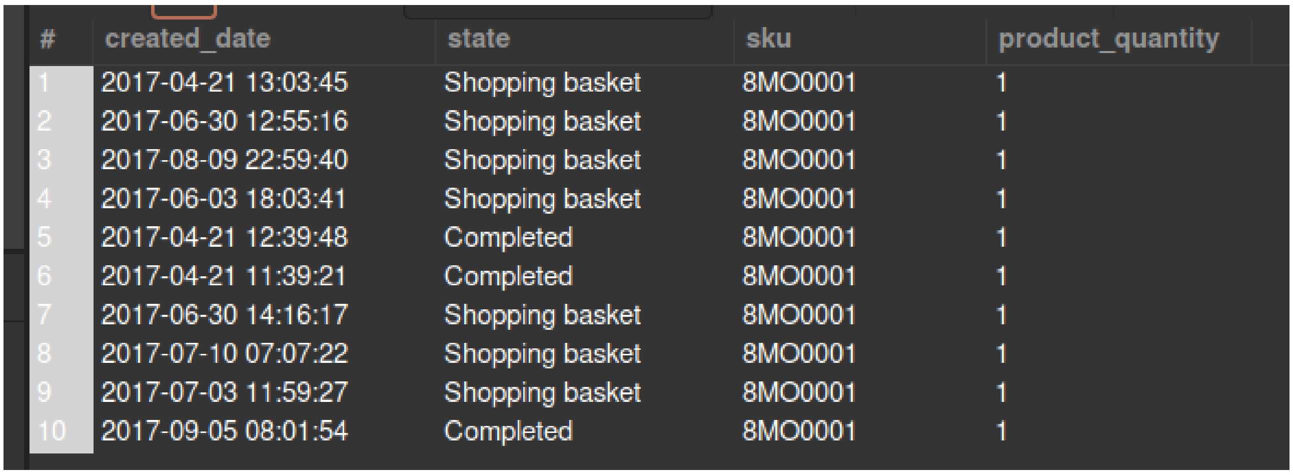
```
                    line_item itm
                              INNER JOIN
                    products prd ON itm.sku = prd.sku ) AS inn
WHERE
    inn.price_dif > 50000
GROUP BY inn.brand
ORDER BY AVG_price_dif DESC;
```



#	brand	AVG_price_dif
1	Withings	92197552
2	DJI	84899213.3
3	Apple	51090491.3
4	QNAP	32705697.2
5	Tado	12030717.3
6	Pack	6949082.6
7	Synology	6106130.7
8	NA	5321329.9
9	Repair	5008006.7
10	Dell	4589566

-- --
-- Query 6. Query 6. We want to know the sku, product_quantity and date of all the orders, ordered by SKU. If it takes too long and/or the connection gets lost try first selecting only the first 50 results and then ordering by sku.

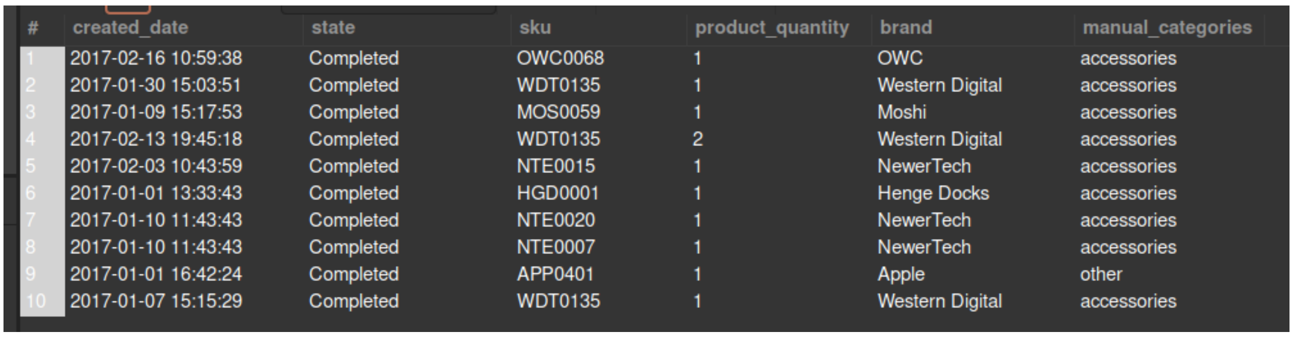
```
SELECT
                    ord.created_date,
        ord.state,
        itm.sku,
        itm.product_quantity
FROM line_item AS itm
INNER JOIN orders AS ord ON itm.id_order = ord.id_order
ORDER BY itm.sku;
```



#	created_date	state	sku	product_quantity
1	2017-04-21 13:03:45	Shopping basket	8MO0001	1
2	2017-06-30 12:55:16	Shopping basket	8MO0001	1
3	2017-08-09 22:59:40	Shopping basket	8MO0001	1
4	2017-06-03 18:03:41	Shopping basket	8MO0001	1
5	2017-04-21 12:39:48	Completed	8MO0001	1
6	2017-04-21 11:39:21	Completed	8MO0001	1
7	2017-06-30 14:16:17	Shopping basket	8MO0001	1
8	2017-07-10 07:07:22	Shopping basket	8MO0001	1
9	2017-07-03 11:59:27	Shopping basket	8MO0001	1
10	2017-09-05 08:01:54	Completed	8MO0001	1

-- ---
-- Query 7. Add to the previous information about "brand" and "manual_categories" fields from the products table.

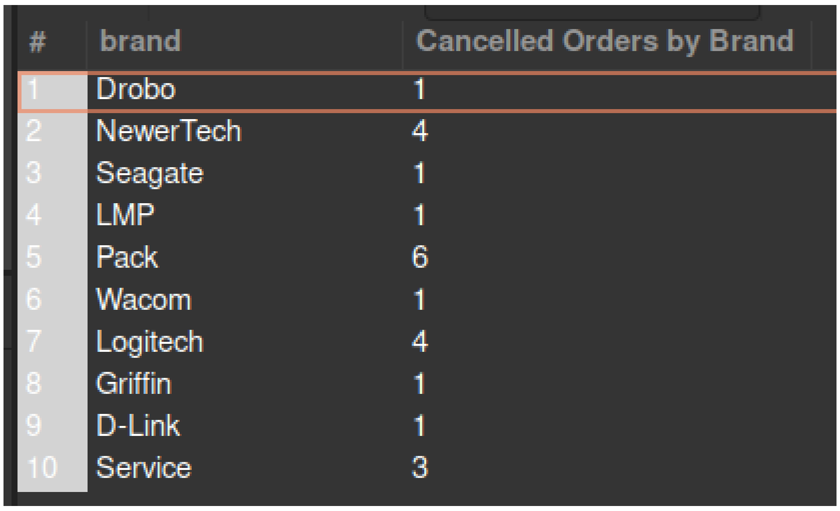
```
SELECT
   ord.created_date, ord.state,
   itm.sku, itm.product_quantity,
   prd.brand, prd.manual_categories
FROM
   line_item AS itm
      INNER JOIN
   orders AS ord ON itm.id_order = ord.id_order
      INNER JOIN
   products AS prd ON itm.sku = prd.sku;
```



#	created_date	state	sku	product_quantity	brand	manual_categories
1	2017-02-16 10:59:38	Completed	OWC0068	1	OWC	accessories
2	2017-01-30 15:03:51	Completed	WDT0135	1	Western Digital	accessories
3	2017-01-09 15:17:53	Completed	MOS0059	1	Moshi	accessories
4	2017-02-13 19:45:18	Completed	WDT0135	2	Western Digital	accessories
5	2017-02-03 10:43:59	Completed	NTE0015	1	NewerTech	accessories
6	2017-01-01 13:33:43	Completed	HGD0001	1	Henge Docks	accessories
7	2017-01-10 11:43:43	Completed	NTE0020	1	NewerTech	accessories
8	2017-01-10 11:43:43	Completed	NTE0007	1	NewerTech	accessories
9	2017-01-01 16:42:24	Completed	APP0401	1	Apple	other
10	2017-01-07 15:15:29	Completed	WDT0135	1	Western Digital	accessories

-- ---
-- Query 8. We want to know which brand and which categories are most frequent in Cancelled orders.
-- Let's keep working on the same query: now we want to keep only Cancelled orders. Modify this query to group the results from the previous query, first by category and then by brand, adding in both cases a count and ordering by descending count. If it takes too long and/or the connection gets lost try putting a LIMIT command.

```
SELECT
         inn.brand, COUNT(*) AS 'Cancelled Orders by Brand'
FROM (
         SELECT
                  -- ord.created_date,
      ord.state,
                  -- itm.sku,
      -- itm.product_quantity,
                  prd.brand -- , prd.manual_categories
         FROM
                  line_item AS itm
                           INNER JOIN
                  orders AS ord ON itm.id_order = ord.id_order
                           INNER JOIN
                  products AS prd ON itm.sku = prd.sku
     LIMIT 1000
      ) as inn
WHERE inn.state = 'Cancelled'
GROUP BY inn.brand ;
```



#	brand	Cancelled Orders by Brand
1	Drobo	1
2	NewerTech	4
3	Seagate	1
4	LMP	1
5	Pack	6
6	Wacom	1
7	Logitech	4
8	Griffin	1
9	D-Link	1
10	Service	3