

Dunnhumby – CASE STUDY

TOPIC - SQL

1. Exploratory queries

- a. Find out which age group is the most active shopper(join hh_demographic and transaction_data)

Ans -

```
select
    age_desc,
    count(t.household_key) as most_active_shopper
from `dunnhumby_dataset.transaction_data` t
join `dunnhumby_dataset.hh_demographic` d
on t.household_key = d.household_key
group by age_desc
order by count(t.household_key) desc;
```

Query results			
JOB INFORMATION		RESULTS	JSON
Row	age_desc	most_active_shopper	
1	45-54	520586	
2	35-44	386327	
3	25-34	249829	
4	65+	103857	
5	55-64	91498	
6	19-24	75206	

b. Which week had the best sales

Ans –

```
select
    week_no,
    round(sum(sales_value),2) as sales
from `dunnhumby_dataset.transaction_data`
group by WEEK_NO
order by 2 desc;
```

Query results

JOB INFORMATION		RESULTS	JSON
Row	week_no	sales	
1	92	113192.87	
2	99	101363.92	
3	98	98949.62	
4	68	97967.05	
5	85	97663.46	
6	94	96964.24	
7	72	96930.47	
8	59	96701.23	
9	46	95878.08	
10	42	93370.29	

- c. What is the average basket size for shoppers (Divide it in small, medium, large)

Ans –

```
select
  case
    when sales_value between 0 and 5 then "small"
    when sales_value between 5 and 10 then "medium"
    when sales_value > 10 then "large"
  end as Type_of_order,
  count(*) as number_of_orders
from `dunnhumby_dataset.transaction_data`
group by 1
order by 2 desc;
```

Query results

JOB INFORMATION		RESULTS	JSON
Row	Type_of_order	number_of_orders	
1	small	2291394	
2	medium	225562	
3	large	78776	

- d. Find foot traffic for each store per week. (Foot traffic: number of customers transacting)

Or

Find top 3 stores with highest foot traffic for each week

Ans –

```
select
    week_no, store_id, x.Highest_foot_traffic
from
    (select WEEK_NO,
        store_id, count(household_key) as Highest_foot_traffic,
        dense_rank() over(partition by WEEK_NO order by count(*)desc) as drnk
    from `dunnhumby_dataset.transaction_data`
    group by STORE_ID, WEEK_NO
    order by week_no, count(household_key) desc) x
where drnk <= 3;
```

Query results

JOB INFORMATION		RESULTS		JSON
Row	week_no	store_id	Highest_foot_traffic	
1	1	324	154	
2	1	321	124	
3	1	32004	117	
4	2	375	205	
5	2	292	169	
6	2	315	135	
7	3	367	346	
8	3	375	310	

e. Top5 spending customers (households) with sales value in integer

Ans-

```
select
    household_key,
    cast(sum(sales_value) as int) as sales
from `dunnhumby_dataset.transaction_data`
group by household_key
order by 2 desc
limit 5;
```

Query results		
JOB INFORMATION		RESULTS
Row	household_key	sales
1	1023	38320
2	1609	27860
3	2322	23647
4	1453	21661
5	2459	20672

F. Find out which income group shops the most

Ans –

```
select  
  
    income_desc, count(*) as most_number_transaction  
  
from `dunnhumby_dataset.hh_demographic` d  
join `dunnhumby_dataset.transaction_data` t  
on d.household_key = t.household_key  
group by income_desc  
order by 2 desc;
```


Query results		
JOB INFORMATION		RESULTS
Row	income_desc	most_number_transaction
1	50-74K	348536
2	35-49K	278341
3	75-99K	168837
4	25-34K	128678
5	Under 15K	114408
6	15-24K	104112
7	125-149K	88004
8	150-174K	71330
9	100-124K	59480

2. Customer profiling

- a. Create a basic customer profiling with first, last visit, number of visits, average money spent per visit and total money spent order by highest avg money

Ans –

```
select
    household_key, min(week_no) as first_visit,
    max(WEEK_NO) as last_visit,
    count(distinct day) as number_of_visits,
    round(sum(sales_value)/ count(distinct BASKET_ID),2)
    as avg_money_spent,
    round(sum(sales_value),2) as total_money_spent
from `dunnhumby_dataset.transaction_data`
group by household_key
order by avg_money_spent;
```

Query results								 SAVE RESULTS ▾
JOB INFORMATION		RESULTS		JSON	EXECUTION DETAILS		EXECUTION GRAPH	PREVIEW
Row	household_key	first_visit	last_visit		number_of_visits	avg_money_spent	total_money_spent	
1	70	2	101		32	2.39	76.4	
2	2304	12	83		69	2.73	240.09	
3	1381	17	101		38	2.74	134.4	
4	2166	16	73		21	2.81	84.21	
5	534	17	102		339	2.83	1315.44	
6	1289	1	102		385	2.88	1512.08	

- b. Do a single customer analysis selecting most spending customer for whom we have demographic information(because not all customers in transaction data are present in demographic table)

Ans -

```
with t as
(
select
    t.household_key,
    round(sum(sales_value),2) as most_spending_customer
from `dunnhumby_dataset.transaction_data` t
join `dunnhumby_dataset.hh_demographic` h
on t.household_key = h.household_key
group by t.household_key)
select t1.*, d.*
from t t1
join `dunnhumby_dataset.hh_demographic` d
on t1.household_key = d.household_key
order by most_spending_customer desc
limit 1;
```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATIONRESULTSJSONEXECUTION DETAILSEXECUTION GRAPHPREVIEW

Row	household_key	most_spending_customer	AGE_DESC	MARITAL_STATUS_CODE	INCOME_DESC	HOMEOWNER_DESC	HH_COMP_DESC	HOUSEHOLD_SIZE
1	1609	27859.68	45-54	A	125-149K	Homeowner	2 Adults Kids	5+

c. What products does the customer buy most

Ans –

```
select
    household_key, product_id, count(quantity) as Qty
from `dunnhumby_dataset.transaction_data`
where household_key in
    (select
        t.household_key
    from `dunnhumby_dataset.transaction_data` t
    join `dunnhumby_dataset.hh_demographic` d
    on t.household_key = d.household_key
    group by household_key
    order by sum(sales_value) desc
    limit 1)
group by household_key, PRODUCT_ID
order by 3 desc;
```

Query results				
JOB INFORMATION		RESULTS	JSON	
Row	household_key	product_id	Qty	
1	1609	1082185	160	
2	1609	6632283	141	
3	1609	951590	125	
4	1609	1029743	109	
5	1609	1127831	98	
6	1609	892728	96	
7	1609	6533889	84	
8	1609	9527158	72	
9	1609	1055737	72	

d. Which promotional campaigns were they a part of(campaign_table)

Ans –

```
select * from `dunnhumby_dataset.campaign_table`  
  
where household_key in  
  (select  
    t.household_key  
  from `dunnhumby_dataset.transaction_data` t  
  join `dunnhumby_dataset.hh_demographic` d  
  on t.household_key = d.household_key  
  group by household_key  
  order by sum(sales_value) desc  
  limit 1);
```

Query results				
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	DESCRIPTION	household_key	CAMPAIGN	
1	TypeA	1609	13	
2	TypeA	1609	18	
3	TypeB	1609	11	

3. Product analysis

- a. Find the most selling product

Ans –

```
with t as
(
select
    t.PRODUCT_ID, count(quantity) as most_selling_product
from `dunnhumby_dataset.transaction_data` t
join `dunnhumby_dataset.product` p
on t.PRODUCT_ID = p.PRODUCT_ID
group by t.product_id
order by 2 desc
limit 1)
select t1.*, pt.*
from t t1
join `dunnhumby_dataset.product` pt
on t1.product_id = pt.PRODUCT_ID;
```

Query results										
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS		EXECUTION GRAPH		PREVIEW		
Row	PRODUCT_ID	most_selling_product	PRODUCT_ID_1	MANUFACTURER	DEPARTMENT	BRAND	COMMODITY_DESC	SUB_COMMOD	CURR_SIZE_OF_PR	
1	1082185	29778	1082185	2	PRODUCE	Natio...	TROPICAL FRUIT	BANANAS	40 LB	

b. When did the product sell the most and where

Ans –

```
select
    product_id, count(quantity) as qty, week_no, store_id
from `dunnhumby_dataset.transaction_data`
where product_id in
    (select
        product_id
    from `dunnhumby_dataset.transaction_data`
    group by PRODUCT_ID
    order by count(*) desc
    limit 1)
group by 1, 3, 4
order by 2 desc
limit 1;
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	
Row	product_id	qty	week_no	store_id	
1	1082185	17	34	367	

c. Where was the product placed in store and featured in ad for that particular store and week

Ans –

```
with t as
(
select
    product_id, count(quantity) as qty, week_no, store_id
from `dunnhumby_dataset.transaction_data`
where product_id in
(
select
    product_id
from `dunnhumby_dataset.transaction_data`
group by PRODUCT_ID
order by count(*) desc
limit 1)
group by 1, 3, 4
order by 2 desc
limit 1)
select c.*
from `dunnhumby_dataset.causal_data` c
join t t1
on c.product_id = t1.product_id
where c.STORE_ID in (t1.store_id)
and c.WEEK_NO in (t1.week_no);
```

Query results [SAVE RESULTS](#)

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS		EXECUTION GRAPH	PREVIEW
Row	PRODUCT_ID	STORE_ID	WEEK_NO	display	mailer		
1	1082185	367	34	0	D		

d. Was it a part of some campaigns

Ans –

```
select
    product_id, CAMPAIGN
from `dunnhumby_dataset.coupon`
where product_id in
    (select
        product_id
    from `dunnhumby_dataset.transaction_data`
    group by PRODUCT_ID
    order by count(*) desc
    limit 1);
```

Query results		
JOB INFORMATION		RESULTS
Row	product_id	CAMPAIGN
1	1082185	8
2	1082185	13
3	1082185	18

- e. How many household did actually redeem coupons for this product in each campaign

Ans –

```
with t as
(select COUPON_UPC, PRODUCT_ID
from `dunnhumby_dataset.coupon`
where product_id in
(select
    product_id
from `dunnhumby_dataset.transaction_data`
group by PRODUCT_ID
order by count(*) desc
limit 1))
select product_id, CAMPAIGN, count(household_key) as count
from t t1
join `dunnhumby_dataset.coupon_redempt` cr
on t1.coupon_upc = cr.COUPON_UPC
group by product_id, CAMPAIGN
order by 2 desc;
```

Query results			
JOB INFORMATION		RESULTS	JSON
Row	product_id	CAMPAIGN	count
1	1082185	18	63
2	1082185	13	35

- f. Which products were the best seller(top 3) for each week and what quantity did they sell

Ans –

```
select *
from
(select
    week_no, product_id, count(product_id) as Qty,
    dense_rank() over(partition by WEEK_NO order by count(PRODUCT_ID) desc)
as top_3
from `dunnhumby_dataset.transaction_data`
group by WEEK_NO, PRODUCT_ID
order by week_no) x
where top_3 <= 3
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	
Row	week_no	product_id	Qty	top_3	
1	1	981760	21	1	
2	1	1082185	20	2	
3	1	1029743	15	3	
4	2	1082185	38	1	
5	2	1029743	28	2	
6	2	1106523	20	3	
7	3	1082185	54	1	
8	3	995242	34	2	
9	3	1133018	31	3	

4. Advance analysis and queries

- a. Find out on which weeks does each household shop and find their cumulative spending over time(sum of all previous) (uses sum over partition)

Ans –

```
with t as
(select
    household_key, WEEK_NO, round(sum(SALES_VALUE),2) as spending
from `dunnhumby_dataset.transaction_data`
group by 1, 2)
select
    t.household_key, t.week_no, t.spending,
    round(sum(spending) over(partition by household_key order by WEEK_NO),2) as spending
from t;
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	
Row	household_key	week_no	spending	spending_1	
1	1	8	78.66	78.66	
2	1	10	41.1	119.76	
3	1	13	26.9	146.66	
4	1	14	63.43	210.09	
5	1	15	53.45	263.54	
6	1	16	26.76	290.3	
7	1	17	23.55	313.85	
8	1	19	110.34	424.19	
9	1	20	87.44	511.63	

- b. Find the trend in spending for each customer
(spending compared to last purchase)(use lag function)

Ans –

```
with t as
(select
    household_key, week_no, round(sum(sales_value),2) as revenue
from `dunnhumby_dataset.transaction_data`
group by 1, 2
order by 1, 2
)
select
    t.household_key, t.week_no, t.revenue,
    lag(revenue,1) over(partition by household_key order by week_no) as prev_revenue,
    round((lag(revenue,1) over(partition by household_key order by week_no) -
        t.revenue),2) as change_in_revenue
from t
order by t.household_key asc;
```

Query results

JOB INFORMATION		RESULTS		JSON	EXECUTION DETAILS	
Row	household_key	week_no	revenue	prev_revenue	change_in_revenue	
1	1	8	78.66	<i>null</i>	<i>null</i>	
2	1	10	41.1	78.66	37.56	
3	1	13	26.9	41.1	14.2	
4	1	14	63.43	26.9	-36.53	
5	1	15	53.45	63.43	9.98	
6	1	16	26.76	53.45	26.69	
7	1	17	23.55	26.76	3.21	
8	1	19	110.34	23.55	-86.79	
9	1	20	87.44	110.34	22.9	

C. Find number of returning customers and percent of returning customers for all week

Ans –

```
with t as
(select
    t2.week_no, t1.household_key,
    case when min(t1.week_no) < t2.week_no then 1 else 0
    end as decider
from `dunnhumby_dataset.transaction_data` t1
left join `dunnhumby_dataset.transaction_data` t2
on t1.household_key = t2.household_key
group by 1, 2)
select
    week_no, sum(decider) as returning_cust, count(decider) as total_cust,
    sum(decider) / count(decider) * 100 as percent_return
from t
group by 1
order by 1;
```

Query results					
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	
Row	week_no	returning_cust	total_cust	percent_return	
1	1	0	88	0.0	
2	2	46	175	26.2857142...	
3	3	115	228	50.4385964...	
4	4	152	270	56.2962962...	
5	5	232	370	62.7027027...	
6	6	314	433	72.5173210...	
7	7	360	491	73.3197556...	
8	8	398	530	75.0943396...	
9	9	493	622	79.2604501...	
10	10	568	708	80.2259887...	

- d. Quarterly analysis: sales comparison(create a new quarter column using case when,12 weeks(3 months)=1 quarter)
(Use cte tables)

Ans –

```
select
  case
    when week_no between 0 and 12 then "quarter1"
    when week_no between 13 and 25 then "quarter2"
    when week_no between 26 and 38 then "quarter3"
    when week_no between 39 and 51 then "quarter4"
    when week_no between 52 and 64 then "quarter5"
    when week_no between 65 and 77 then "quarter6"
    when week_no between 78 and 90 then "quarter7"
    when week_no between 91 and 102 then "quarter8"
  end as Quarter,
  round(sum(sales_value),2) as Sales
from `dunnhumby_dataset.transaction_data`
group by 1;
```

Query results		
JOB INFORMATION		RESULTS
Row	Quarter	Sales
1	quarter1	328865.31
2	quarter2	1001743.24
3	quarter3	1073977.47
4	quarter4	1123719.64
5	quarter5	1148910.61
6	quarter6	1143552.58
7	quarter7	1144484.66
8	quarter8	1092209.57

- e. Are the customers spending more or less over time
(group in 25 week segments)

Ans –

```
select household_key,
       case
         when week_no between 0 and 25 then "quarter1"
         when week_no between 26 and 51 then "quarter2"
         when week_no between 52 and 77 then "quarter3"
         when week_no between 78 and 102 then "quarter4"
       end as Quarter,
       round(sum(sales_value),2) as spending
from `dunnhumby_dataset.transaction_data`
group by 1,2
order by 1 asc;
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	
Row	household_key	Quarter		spending	
1	1	quarter1		765.38	
2	1	quarter2		1148.71	
3	1	quarter3		1131.98	
4	1	quarter4		1284.09	
5	2	quarter1		409.65	
6	2	quarter2		580.54	
7	2	quarter3		395.05	
8	2	quarter4		569.1	
9	3	quarter1		464.58	
10	3	quarter2		1424.74	

f. Customer churn analysis for each quarter

Ans –

```
with t as
(
select *,
    case
        when week_no between 0 and 12 then "quarter1"
        when week_no between 13 and 25 then "quarter2"
        when week_no between 26 and 38 then "quarter3"
        when week_no between 39 and 51 then "quarter4"
        when week_no between 52 and 64 then "quarter5"
        when week_no between 65 and 77 then "quarter6"
        when week_no between 78 and 90 then "quarter7"
        when week_no between 91 and 102 then "quarter8"
    end as Quarter
from `dunnhumby_dataset.transaction_data`)
select
    t1.Quarter, count(distinct t1.household_key) as churned_cust
from t t1
full outer join t t2
on t1.household_key = t2.household_key
and t1.Quarter < t2.Quarter
where t2.household_key is null
group by 1
order by 1;
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	Quarter	churned_cust		
1	quarter1	5		
2	quarter2	3		
3	quarter3	5		
4	quarter4	8		
5	quarter5	16		
6	quarter6	34		
7	quarter7	119		
8	quarter8	2310		

- g. Find the retained customers for each quarter
(retained :Households who were there in previous quarters and are there in the current quarter)

Ans –

```
with t as
(
    select *,
        case
            when week_no between 0 and 12 then "quarter1"
            when week_no between 13 and 25 then "quarter2"
            when week_no between 26 and 38 then "quarter3"
            when week_no between 39 and 51 then "quarter4"
            when week_no between 52 and 64 then "quarter5"
            when week_no between 65 and 77 then "quarter6"
            when week_no between 78 and 90 then "quarter7"
            when week_no between 91 and 102 then "quarter8"
        end as Quarter
    from `dunnhumby_dataset.transaction_data`)
select
    t1.Quarter, count(distinct t1.household_key) as retained_cust
from t t1
left join t t2
on t1.household_key = t2.household_key
where t1.Quarter > t2.Quarter
group by 1
order by 1;
```

Query results

JOB INFORMATION		RESULTS	JSON
Row	Quarter	retained_cust	
1	quarter2	1476	
2	quarter3	2285	
3	quarter4	2283	
4	quarter5	2302	
5	quarter6	2316	
6	quarter7	2323	
7	quarter8	2309	


- h. Find products(product table :SUB_COMMODITY_DESC) which are most frequently bought together

Ans –

with a as

```
(select *
from `dunnhumby_dataset.product` p
join `dunnhumby_dataset.transaction_data` t
on p.PRODUCT_ID = t.PRODUCT_ID)
SELECT
a1.sub_commodity_desc as product1,
a2.sub_commodity_desc as product2,
count(distinct a1.basket_id) as no_of_times_together
from a a1
join a as a2
on a1.basket_id = a2.basket_id
where a1.sub_commodity_desc < a2.sub_commodity_desc
group by 1 , 2
order by 3 desc
limit 100;
```

Query results

 SAVE RESULTS ▾

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	product1	product2	no_of_times_together
1	BANANAS	FLUID MILK WHITE ONLY	15662
2	FLUID MILK WHITE ONLY	MAINSTREAM WHITE BREAD	14075
3	FLUID MILK WHITE ONLY	SOFT DRINKS 12/18&15PK CA...	10576
4	FLUID MILK WHITE ONLY	SHREDDED CHEESE	10349
5	DAIRY CASE 100% PURE JUICE...	FLUID MILK WHITE ONLY	9549
6	FLUID MILK WHITE ONLY	KIDS CEREAL	8428
7	FLUID MILK WHITE ONLY	SFT DRNK 2 LITER BTL CARB I...	8021
8	FLUID MILK WHITE ONLY	POTATO CHIPS	7660
9	EGGS - LARGE	FLUID MILK WHITE ONLY	7569

i. Calculate Customer lifetime value(CLV) for different age group

Average purchase value — the value of all customer purchases over a particular time frame , divided by the number of purchases in that period

Average purchase frequency — divide the number of purchases in that same time period by the number of individual customers who made a transaction over the same period

Customer value — the average purchase frequency multiplied by the average purchase value

Average customer lifespan — the average length of time a customer continues buying from you

CLV = customer value X average customer lifespan

Ans –

```
select
    AGE_DESC,
    round((avg_purch_val*avg_purch_freq*avg_cust_lifespan),2) as clv
from
    (with cte as
    (select
        household_key,
        (max(WEEK_NO)- min (WEEK_NO))
        as cust_duration
    from `dunnhumby_dataset.transaction_data`
    group by household_key)
    select
        AGE_DESC,
        sum(SALES_VALUE)/count(distinct(BASKET_ID)) as avg_purch_val,
        count(distinct(BASKET_ID))/count(distinct(d.household_key))
        as avg_purch_freq,
        (sum(cte.cust_duration)/count(1)) as avg_cust_lifespan,
    from `dunnhumby_dataset.transaction_data` t
    inner join `dunnhumby_dataset.hh_demographic` d
    on t.household_key=d.household_key
    join cte
    on cte.household_key=d.household_key
    group by AGE_DESC)
```

Query results

JOB INFORMATION

RESULTS

JSON

Row	AGE_DESC	clv
1	65+	382170.97
2	55-64	461916.72
3	35-44	588905.9
4	25-34	503434.69
5	45-54	525416.21
6	19-24	427939.21