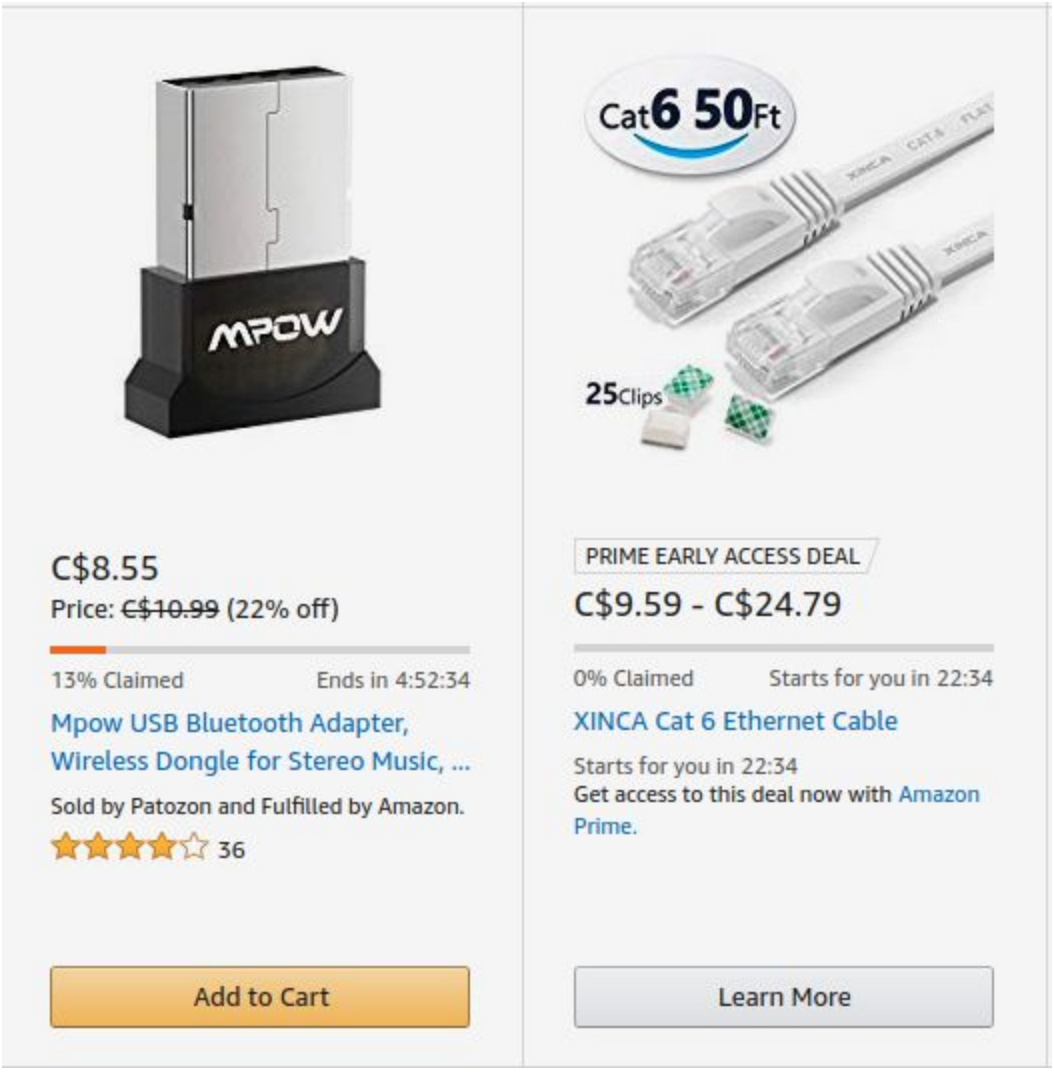
Amazon is running Blackfriday campaign and coming up with instant promotional offers. In that case they apply offers for a specific period of time on limited products. Such as live example of this blackfriday:



The retail team sends promotion information in following format:

country_cd|date|start_time|end_time|promo_cd:p_cd1,p_cd2,p_cd3,...
CA|22-Nov-2018|08:00:00|24:00:00|FLAT_30:DENIM_001,PE_230,ZARA_MEN_3,GAP_98,PINK_29,HP_29
CA|23-Nov-2018|00:00:00|12:00:00|LOYALTY_CASHBACK:DELL_32,REEBOK_393,NIKE_20,PUMA_192,PUMA_102
USA|23-Nov-2018|08:00:00|12:00:00|FLAT_35_ABOVE_70:DELL_32,REEBOK_393,NIKE_20,PUMA_192,PUMA_102

However, while storing in data warehouse we always prefer it to be in flat structure as

country_cd|date|start_time|end_time|promo_cd|product_cd
CA|22-Nov-2018|08:00:00|24:00:00|FLAT_30|DENIM_001
CA|22-Nov-2018|08:00:00|24:00:00|FLAT_30|PE_230
CA|22-Nov-2018|08:00:00|24:00:00|FLAT_30|ZARA_MEN_3
CA|22-Nov-2018|08:00:00|24:00:00|FLAT_30|GAP_98
CA|22-Nov-2018|08:00:00|24:00:00|FLAT_30|PINK_29
CA|22-Nov-2018|08:00:00|24:00:00|FLAT_30|HP_29
CA|23-Nov-2018|00:00:00|12:00:00|LOYALTY_CASHBACK|DELL_32
CA|23-Nov-2018|00:00:00|12:00:00|LOYALTY_CASHBACK|REEBOK_393
CA|23-Nov-2018|00:00:00|12:00:00|LOYALTY_CASHBACK|NIKE_20
CA|23-Nov-2018|00:00:00|12:00:00|LOYALTY_CASHBACK|PUMA_192
CA|23-Nov-2018|00:00:00|12:00:00|LOYALTY_CASHBACK|PUMA_102
USA|23-Nov-2018|08:00:00|12:00:00|FLAT_35_ABOVE_70|DELL_32
USA|23-Nov-2018|08:00:00|12:00:00|FLAT_35_ABOVE_70|REEBOK_393
USA|23-Nov-2018|08:00:00|12:00:00|FLAT_35_ABOVE_70|NIKE_20
USA|23-Nov-2018|08:00:00|12:00:00|FLAT_35_ABOVE_70|PUMA_192
USA|23-Nov-2018|08:00:00|12:00:00|FLAT_35_ABOVE_70|PUMA_102

User Defined Tabular Function (UDTF)

Purpose:

UDTF takes single record as input and generates multiple records in output.

Example:

Generate combination of transaction id and product id for a given transaction with all products flattened in single record.

Coding Approach:

Create a class which extends org.apache.hadoop.hive.ql.udf.generic.GenericUDTF
Define methods

initialize: will return the structure information of output record
process: will be called on each new record
close: any cleanup tasks to be carried out

Example:

```
hive (mydb)> select flattrans("1|2,3,4");
OK
trans_id        product_id
1       2
1       3
1       4
```

Optimize HQL

Partition Table:

Create table:

CREATE TABLE transaction (tx_id int, product_id int, amt double, qty double) partitioned by (trans_dt string) row format delimited fields terminated by ',';

Insert data:

insert into transaction partition(trans_dt='23-Nov-2018') values (1,1,2,3);
insert into transaction partition(trans_dt='24-Nov-2018') values (2,1,2,3);

Select data:

select * from transaction;
explain select * from transaction;

select * from transaction where trans_dt='23-Nov-2018';
explain select * from transaction where trans_dt='23-Nov-2018';

See all existing partitions:

Show partitions transaction;

Take a look at HDFS directory structure

Exercise:

1. Create a new partition directory
2. Copy some data files into that directory
3. Go back to the terminal and see the result of select command.

Statistics:

Collect statistics:

Compute stats transaction;
Compute incremental stats transaction;
Compute incremental stats transaction partition (trans_dt='24-Nov-2018');
Show table stats transaction;

Show statistics:

Show table stats transaction;
Show column stats transaction;

Drop statistics:

Drop stats transaction;
Drop incremental stats transaction partition (trans_dt='24-Nov-2018');

Reference:

- https://www.phdata.io/hands-on-example-with-hive-partitioning/
- https://www.cloudera.com/documentation/enterprise/5-9-x/topics/impala_compute_stats.html