*require* **"keycdn"***require* **"sendgrid-ruby"***include* ***ActionView***::***Helpers***::***NumberHelper*class *Merchant*** < ***ActiveRecord***::***Base*** *attr\_accessor* **:create\_defaults** *attr\_accessible* **:description**, **:logo\_url**, **:tag\_line**, **:company\_name**, **:is\_approved**, **:third\_party\_id**,  
 **:website**, **:owner\_id**, **:guid**, **:currency**, **:is\_participating\_in\_cc\_loyalty**, **:api\_key**, **:api\_access\_enabled**,  
 **:widget\_enabled**, **:widget\_text**, **:widget\_bg\_color**, **:widget\_text\_color**, **:widget\_expanded\_by\_default**, **:widget\_location**,  
 **:from\_name**, **:from\_email**, **:has\_store**, **:logo**, **:logo\_content\_type**, **:slug**, **:completed\_profile**,  
 **:uninstalled\_rewards\_at**, **:platform**, **:uninstalled\_analytics\_at**,  
 **:has\_rewards\_installed**, **:installed\_rewards\_at**, **:program\_live**,  
 **:has\_analytics\_installed**, **:installed\_analytics\_at**,  
 **:widget\_theme**, **:widget\_mobile**, **:widget\_rounded**, **:widget\_height**,  
 **:widget\_distance\_from\_end**, **:widget\_brand\_color**, **:widget\_location\_mobile**,  
 **:swell\_redemptions\_enabled**, **:points\_per\_dollar**, **:install\_order\_volume**,  
 **:email\_header\_image**, **:email\_header\_image\_content\_type**, **:email\_layout**, **:email\_header\_width**, **:email\_layout\_html**, **:force\_store\_accounts**,  
 **:store\_account\_registration\_url**, **:store\_account\_login\_url**, **:store\_account\_host**,  
 **:widget\_font\_url**, **:widget\_font\_family**, **:widget\_header\_text**, **:include\_shipping**, **:plan**,  
 **:cross\_origin\_store\_accounts**, **:process\_order\_webhook**, **:points\_expire\_after\_days**,  
 **:product\_exclusions**, **:collection\_exclusions**, **:contact\_email**, **:exclude\_customer\_tags**, **:customer\_tags**,  
 **:sync\_newsletter\_with\_platform**, **:oauth\_app\_id**,  
 **:limit\_redemptions**, **:max\_redemptions\_limit**, **:max\_redemptions\_limit\_days**, **:weekly\_digest\_enabled**,  
 **:recurring\_charge\_id**, **:usage\_charge\_job\_id**, **:free\_trial\_expires\_at**, **:free\_trial\_expiration\_job\_id**, **:on\_free\_trial**,  
 **:program\_locked**, **:stripe\_customer\_id**, **:free\_products\_enabled**, **:webhook\_url**, **:points\_reminder\_delay\_days**,  
 **:invoice\_job\_id**, **:source**, **:setup\_rewards\_page**, **:sync\_fields\_with\_processor**, **:metafield\_namespace**,  
 **:vip\_tiers\_window\_months**, **:vip\_tiers\_start\_date**, **:vip\_tiers\_job\_running**, **:vip\_tiers\_enabled**, **:can\_generate\_coupons**, **:referral\_reminder\_days**,  
 **:order\_paid\_status**, **:order\_paid\_state**, **:process\_order\_status**, **:process\_order\_state**, **:process\_refund\_status**, **:process\_refund\_state**,  
 **:security\_level**, **:include\_purchases\_with\_details**, **:facebook\_app\_id**, **:referral\_security\_disabled**, **:process\_discounts\_webhook**,  
 **:order\_price\_setting**, **:order\_price\_eval**, **:include\_storefront\_css**, **:custom\_script\_url**, **:nav\_exclusions**, **:cash\_payout\_setting**, **:default\_payout\_percentage**,  
 **:affiliate\_include\_shipping**, **:affiliate\_order\_price\_setting**,  
 **:discount\_code\_based\_referral\_program\_enabled**, **:discount\_code\_generation\_strategy**, **:first\_time\_referral\_discounts\_enable\_flag**,  
 **:default\_paypal\_email\_subject**, **:default\_paypal\_email\_message**, **:manual\_adjustment\_admin\_only**, **:sync\_tiers\_with\_platform**, **:verified\_email\_domain**, **:app\_key**,  
 **:show\_billing\_information**, **:delay\_purchase\_vip\_tier\_eligibility**, **:delay\_purchase\_vip\_tier\_eligibility\_days**, **:code\_based\_referral\_program\_on\_demand**, **:referral\_blocked\_domains**,  
 **:limit\_referral\_by\_ip\_max\_amount**, **:limit\_referral\_by\_ip\_hours\_interval** *belongs\_to* **:owner**, **:class\_name** *=>* ***User*** *has\_one* **:onboard** *has\_many* **:customers** *has\_many* **:purchases** *has\_many* **:campaigns** *has\_many* **:perks**, **:through** *=>* **:campaigns** *has\_many* **:email\_campaigns** *has\_many* **:referral\_codes** *has\_many* **:processors** *has\_many* **:processor\_snapshots**, **:through** *=>* **:processors** *has\_many* **:merchant\_categories** *has\_many* **:oauth\_tokens** *has\_one* **:platform\_token**, -> (*merchant*) { *where*(**type**: *merchant*.platform\_token\_type) }, **class\_name**: ***OauthToken*** *has\_many* **:redemption\_options** *has\_many* **:campaign\_popup\_configurations** *has\_many* **:localizations** *has\_many* **:user\_actions** *has\_many* **:referral\_snapshots** *has\_many* **:affiliate\_snapshots** *has\_many* **:dashboard\_snapshots** *has\_many* **:dashboard\_snapshot\_records** *has\_many* **:point\_redemptions** *has\_many* **:redemption\_codes** *has\_many* **:referral\_discount\_codes** *has\_many* **:onboarding\_tasks** *has\_many* **:subscriptions** *has\_many* **:event\_consumers** *has\_many* **:event\_maps**, **:through** *=>* **:event\_consumers** *has\_many* **:images** *has\_many* **:referral\_receipts** *has\_many* **:kit\_conversations** *has\_many* **:vip\_tiers** *has\_many* **:background\_jobs** *has\_many* **:referrals** *has\_many* **:rewards\_page\_configurations** *has\_one* **:discount\_code\_based\_referral\_program** *has\_many* **:affiliates** *has\_many* **:cash\_payouts** *has\_many* **:batch\_payouts** *has\_one* **:vip\_tiers\_setting** monetize **:thirty\_day\_revenue\_cents**, **with\_model\_currency**: **:currency** *accepts\_nested\_attributes\_for* **:merchant\_categories** *validates* **:merchant\_categories**, **length**: { **maximum**: 3 }  
  
 *validate* **:tiers\_require\_subscription** *validate* **:referral\_reminder\_days\_format** *validate* **:referral\_campaign\_requirements**, **if**: ***Proc***.new { |*merchant*| *merchant*.discount\_code\_based\_referral\_program\_enabled\_changed? }  
  
 *before\_create* **:set\_guid** *before\_save* **:handle\_changed\_attrs** *before\_save* **:monitor\_profile** *before\_save* **:strip\_blocked\_domains** *after\_update* **:copy\_vip\_tiers\_settings**, **if**: ***Proc***.new { |*merchant*| *merchant*.has\_tier\_settings\_changed? }  
 *after\_update* **:monitor\_widget**, **:monitor\_swell\_redemption\_options**, **:monitor\_points\_expiration**, **:monitor\_newsletter\_sync** *after\_update* **:manage\_discount\_code\_based\_referral\_program**, **if**: ***Proc***.new { |*merchant*| *merchant*.discount\_code\_based\_referral\_program\_enabled\_changed? }  
 *after\_create* **:setup\_rewards\_program** *after\_create* **:setup\_initial\_subscription** *after\_create* **:monitor\_merchant\_created** *after\_create* **:create\_vip\_tiers\_settings** has\_attached\_file **:logo**, **:default\_url** *=>* **"/assets/merchant.png"** validates\_attachment\_content\_type **:logo**, **:content\_type** *=>* **/**\Aimage**/** has\_attached\_file **:email\_header\_image**, **:default\_url** *=>* **"/assets/email-logo.png"** validates\_attachment\_content\_type **:email\_header\_image**, **:content\_type** *=>* **/**\Aimage**/** after\_post\_process **:update\_redemption\_option\_images  
  
 def** *to\_loader\_obj  
 obj* = ***Hash***.new  
 *attrs\_to\_include* = [**:store\_account\_login\_url**, **:store\_account\_registration\_url**, **:force\_store\_accounts**, **:company\_name**, **:description**, **:id**, **:platform**, **:currency**]  
 *attrs\_to\_include*.each **do** |*attr*|  
 **if** *attr*.to\_s.eql?(**"currency"**)  
 *obj*[*attr*] = ***Money***.new(1,**self**.*send*(*attr*))  
 **else** *obj*[*attr*] = **self**.*send*(*attr*)  
 **end  
 end** *obj* **end  
  
 def** *invalidate\_loader\_cache\_async* **if $redis**.*get*(**"invalidate-cache-job-**#{**self**.id}**"**).nil?  
 **$redis**.setex(**"invalidate-cache-job-**#{**self**.id}**"**, 5, **true**)  
 ***AccountProcessing***::***InvalidateLoaderJob***.perform\_in(5.seconds, **self**.id)  
 **end  
 end  
  
 def** *invalidate\_loader\_cache  
 ratelimit* = ***Sidekiq***::***Limiter***.window(**"keycdn-invalidation-request"**, 20, 60, **wait\_timeout**: 240)  
 *ratelimit*.within\_limit **do** *keycdn* = ***KeyCDN***::***Client***.new(***ENV***[**"KEYCDN\_APIKEY"**])  
 *keycdn*.del(**"zones/purgeurl/**#{***ENV***[**'KEYCDN\_ZONE\_ID'**]}**.json"**, { **"urls[0]"** *=>* **"**#{***ENV***[**'KEYCDN\_ZONE\_URL'**]}**/loader/**#{**self**.guid}**.js"** })  
 **end  
 end  
  
 def** *has\_default\_tier\_settings?* (**self**.vip\_tiers\_window\_months == 12 && **self**.vip\_tiers\_start\_date == **nil** &&  
 **self**.delay\_purchase\_vip\_tier\_eligibility == **false** && **self**.delay\_purchase\_vip\_tier\_eligibility\_days == 30)  
 **end  
  
 def** *tiers\_require\_subscription  
 is\_vip\_tiers\_enabled* = **self**.is\_vip\_tiers\_enabled?  
 **if** *is\_vip\_tiers\_enabled  
 sub* = **self**.current\_subscription  
  
 **if** *sub* && *sub*.plan\_id <= 2  
 errors.add(**:vip\_tiers\_enabled**, **'You must be on the GOLD plan or higher to enable VIP Tiers'**)  
 **return false  
 end  
 end  
 end  
  
 def** *referral\_reminder\_days\_format* **if self**.referral\_reminder\_days.to\_i < 0  
 errors.add(**:referral\_reminder\_days**, **"Please enter valid input value should be a positive number."**)  
 **return false  
 end  
 end  
  
 def** *referral\_campaign\_requirements* **if** discount\_code\_based\_referral\_program\_enabled\_changed? && discount\_code\_based\_referral\_program\_enabled  
 *ref\_campaign* = campaigns.*where*(**:type** *=>* **"ReferralCampaign"**, **:active** *=>* **true**).take  
 **if** *ref\_campaign* && !*ref\_campaign*.referral\_redemption\_option\_id  
 errors.add(**:discount\_code\_based\_referral\_program\_enabled**, **"cannot enable discount code program please select coupon as reward for referred customer to enable this program."**)  
 **return false  
 end  
 unless** *ref\_campaign* errors.add(**:discount\_code\_based\_referral\_program\_enabled**, **"cannot enable discount code program please enable referral program first."**)  
 **return false  
 end  
 end  
 end  
  
 def** *copy\_account\_setup*(*dest\_merchant\_id*)  
 *dest\_merchant* = ***Merchant***.*find*(*dest\_merchant\_id*)  
  
 *dest\_merchant*.campaigns.delete\_all  
 *dest\_merchant*.redemption\_options.delete\_all  
 *dest\_merchant*.localizations.delete\_all  
 *dest\_merchant*.email\_campaigns.delete\_all  
 *dest\_merchant*.vip\_tiers.delete\_all  
  
 *dest\_merchant*.campaign\_popup\_configurations.each **do** |*cpc*|  
 *cpc*.campaign\_popup\_items.delete\_all  
 *cpc*.delete  
 **end** *# vip tier attributes  
 #* ***:redemption\_option\_id*** *roIdMap* = ***Hash***.new  
 *campaignIdMap* = ***Hash***.new  
 *vipIdMap* = ***Hash***.new  
  
 **self**.redemption\_options.each **do** |*ro*|  
 *copy* = *ro*.dup  
 *copy*.merchant\_id = *dest\_merchant\_id  
 copy*.save  
  
 *roIdMap*[*ro*.id] = *copy*.id  
 **end  
  
 self**.vip\_tiers.each **do** |*vip\_tier*|  
 *copy* = *vip\_tier*.dup  
 *copy*.merchant\_id = *dest\_merchant\_id* **if** *copy*.redemption\_option\_id.present?  
 *copy*.redemption\_option\_id = *roIdMap*[*vip\_tier*.redemption\_option\_id]  
 **end** *copy*.save  
  
 *vipIdMap*[*vip\_tier*.id] = *copy*.id  
 **end  
  
 self**.campaigns.each **do** |*campaign*|  
 *copy* = *campaign*.dup  
 *copy*.merchant\_id = *dest\_merchant\_id  
 copy*.save  
 *campaignIdMap*[*campaign*.id] = *copy*.id  
 **end** *dest\_merchant*.campaigns.each **do** |*campaign*|  
 **if** *campaign*.redemption\_option\_id.present?  
 *campaign*.redemption\_option\_id = *roIdMap*[*campaign*.redemption\_option\_id]  
 **end  
  
 if** *campaign*.referral\_redemption\_option\_id.present?  
 *campaign*.referral\_redemption\_option\_id = *roIdMap*[*campaign*.referral\_redemption\_option\_id]  
 **end  
  
 if** *campaign*.referrer\_target\_campaign\_id.present?  
 *campaign*.referrer\_target\_campaign\_id = *campaignIdMap*[*campaign*.referrer\_target\_campaign\_id]  
 **end  
  
 if** *campaign*.referred\_user\_target\_campaign\_id.present?  
 *campaign*.referred\_user\_target\_campaign\_id = *campaignIdMap*[*campaign*.referred\_user\_target\_campaign\_id]  
 **end  
  
 if** *campaign*.vip\_tier\_id.present?  
 *campaign*.vip\_tier\_id = *vipIdMap*[*campaign*.vip\_tier\_id]  
 **end** *campaign*.save  
  
 **end** *cpcIdMap* = ***Hash***.new  
  
 **self**.campaign\_popup\_configurations.each **do** |*cpc*|  
 *copy* = *cpc*.dup  
 *copy*.merchant\_id = *dest\_merchant\_id* **if** *copy*.campaign\_id.present?  
 *copy*.campaign\_id = *campaignIdMap*[*copy*.campaign\_id]  
 **end** *copy*.save  
  
 *cpcIdMap*[*cpc*.id] = *copy*.id  
  
 *cpc*.campaign\_popup\_items.each **do** |*cpi*|  
 *c* = *cpi*.dup  
 *c*.campaign\_popup\_configuration\_id = *copy*.id  
 *c*.save  
 **end  
  
 end** *# campaign\_popup\_configuration* **self**.localizations.each **do** |*loc*|  
 *copy* = *loc*.dup  
 *copy*.merchant\_id = *dest\_merchant\_id* **if** *copy*.campaign\_popup\_configuration\_id.present?  
 *copy*.campaign\_popup\_configuration\_id = *cpcIdMap*[*copy*.campaign\_popup\_configuration\_id]  
 **end** *copy*.save  
 **end  
  
 self**.email\_campaigns.each **do** |*ec*|  
 *copy* = *ec*.dup  
 *copy*.merchant\_id = *dest\_merchant\_id* **if** *copy*.campaign\_id.present?  
 *copy*.campaign\_id = *campaignIdMap*[*copy*.campaign\_id]  
 **end** *copy*.save  
 **end** *arc* = *dest\_merchant*.campaigns.*where*(**:type** *=>* **"ReferralCampaign"**, **:active** *=>* **true**).take  
  
 **if** *arc  
 dest\_merchant*.referral\_codes.find\_each **do** |*rc*|  
 *rc*.campaign\_id = *arc*.id  
 *rc*.save  
 **end  
 end  
  
 end  
  
 def** *loader\_script\_url  
 scriptUrl* = ***ENV***[**"API\_HOST"**] + **"/loader/**#{**self**.guid}**.js"  
 end  
  
 def** *manage\_discount\_code\_based\_referral\_program  
 ref\_campaign* = campaigns.*where*(**:type** *=>* **"ReferralCampaign"**, **:active** *=>* **true**).take  
  
 **if** *ref\_campaign*.present?  
 **if** discount\_code\_based\_referral\_program\_enabled  
  
 *#* ***if*** *program is enabled, this checks to see* ***if*** *they have any exiting codes  
 # we will need to re-enable old codes as well as update them* ***if*** *referral program changed  
 # current implementation looks like both these will run at the same time one after another  
 # this might be okay on shopify since it's just one api call each for the price rule  
 # otherwise I think it's a better solution to just check* ***if*** *we need to update both  
 # and update both in the same job otherwise it feels like there could be some consistency issues  
 # where the status update and the rule update overwrite each other  
  
 referral\_discount* = **self**.referral\_discount\_codes.take  
   
 *#* ***if*** *this is re-enabling it* **if** *referral\_discount*.present?  
 *# status being whether the discount (or price rule) is enabled or not* update\_referral\_discounts\_status(*ref\_campaign*.id, **true**)  
 **end** *# maybe we just have the normal setup handle updating existing codes as well  
 # so that only one long running job happens regardless of first time or not?  
  
 # in addition to all these possible updates, let's do our normal setup  
 # which will create codes for all people who don't have them* setup\_discount\_code\_based\_referral\_program(*ref\_campaign*)  
 **else** *# disable all codes* ***when*** *program is disabled* update\_referral\_discounts\_status(*ref\_campaign*.id, **false**)  
 **end  
 end** *#* ***if*** *discount\_code\_generation\_strategy\_changed?  
 # self.update\_discounts\_code\_strategy  
 #* ***end* end  
  
 def** *update\_discounts\_code\_strategy* **if self**.referral\_discount\_codes.any?  
 *ref\_campaign* = **self**.campaigns.*where*(**:type** *=>* **"ReferralCampaign"**, **:active** *=>* **true**).take  
 *ref\_campaign* = **self**.campaigns.*where*(**:type** *=>* **"ReferralCampaign"**).take **unless** *ref\_campaign  
  
 job\_id* = ***CustomerProcessing***::***UpdateReferralDiscountsStrategyJob***.perform\_async(**self**.id, *ref\_campaign*.id, **self**.discount\_code\_generation\_strategy)  
 ***DiscountCodeReferralProgramBackgroundJob***.*create*(**merchant\_id**: **self**.id, **job\_id**: *job\_id*, **title**: **"Updating discount codes"**)  
 **end  
 end  
  
 def** *update\_referral\_discounts\_status*(*campaign\_id*, *option*)  
 **self**.processors.each **do** |*processor*|  
 **if** *processor*.supports\_code\_based\_referral\_program  
 **if** *processor*.type == **'ShopifyProcessor'  
 *CustomerProcessing***::***UpdateReferralDiscountsStatusJob***.perform\_async(**self**.id, *campaign\_id*, *processor*.id, *option*)  
 **else** *bg\_job* = ***DiscountCodeReferralProgramBackgroundJob***.*create*(**merchant\_id**: **self**.id, **title**: **"Updating discount codes"**)  
 *bg\_job*.job\_id = ***CustomerProcessing***::***UpdateReferralDiscountsStatusJob***.perform\_async(**self**.id, *campaign\_id*, *processor*.id, *option*, *bg\_job*.id)  
 *bg\_job*.save  
 **end  
 end  
 end  
 end  
  
 def** *update\_referral\_discounts\_price\_rule*(*campaign\_id*, *referral\_redemption\_option\_id*=**nil**)  
 **self**.processors.each **do** |*processor*|  
 **if** *processor*.supports\_code\_based\_referral\_program  
 **if** *processor*.type == **'ShopifyProcessor'  
 *CustomerProcessing***::***UpdateReferralDiscountsPriceRuleJob***.perform\_async(**self**.id, *campaign\_id*, *processor*.id, *referral\_redemption\_option\_id*)  
 **else** *bg\_job* = ***DiscountCodeReferralProgramBackgroundJob***.*create*(**merchant\_id**: **self**.id, **title**: **"Updating discount codes"**)  
 *bg\_job*.job\_id = ***CustomerProcessing***::***UpdateReferralDiscountsPriceRuleJob***.perform\_async(**self**.id, *campaign\_id*, *processor*.id, *referral\_redemption\_option\_id*, *bg\_job*.id)  
 *bg\_job*.save  
 **end  
 end  
 end  
 end  
  
 def** *setup\_discount\_code\_based\_referral\_program*(*ref\_campaign*=**nil**)  
 *# should probably update total\_codes count* ***if*** *it does exist and  
 # this is being run after being turned off for a* ***while* self**.create\_discount\_code\_based\_referral\_program(**campaign\_id**: *ref\_campaign*.id) **unless self**.discount\_code\_based\_referral\_program.present?  
  
 **unless** code\_based\_referral\_program\_on\_demand  
 *bg\_job* = ***DiscountCodeReferralProgramBackgroundJob***.*create*(**merchant\_id**: **self**.id, **title**: **"Generating discount codes"**)  
 *bg\_job*.job\_id = ***CustomerProcessing***::***CreateReferralDiscountCodesJob***.perform\_async(**self**.id, *ref\_campaign*.id, *bg\_job*.id)  
 *bg\_job*.save  
 **end  
 end  
  
 def** *setup\_free\_subscription\_for\_dev\_store  
 sub* = **self**.current\_subscription  
 *sub*.current = **false** *sub*.was\_current = **true  
  
 if** *sub*.billing\_provider.eql?(**"invoiced"**) && **self**.invoice\_job\_id.present?  
 ***Workers***::***Control***.cancel\_by\_id(**self**.invoice\_job\_id)  
 **end** *sub2* = ***Subscription***.new  
 *sub2*.merchant\_id = **self**.id  
 *sub2*.current = **true** *sub2*.billing\_provider = **"invoiced"** *sub2*.plan\_id = 5  
 *sub2*.base\_price\_cents = 0  
 *sub2*.per\_order\_price\_cents = 0  
 *sub2*.included\_orders = 100000000  
 *sub*.save  
 *sub2*.save  
 **end  
  
 def** *setup\_and\_schedule\_invoiced\_subscription*(*date\_str*, *plan\_id*, *base\_price\_cents*=0, *per\_order\_price\_cents*=0, *included\_orders*=9999999)  
 create\_and\_switch\_to\_new\_subscription(**"invoiced"**, *plan\_id*, *base\_price\_cents*, *per\_order\_price\_cents*, *included\_orders*)  
  
 **self**.schedule\_invoice\_on(*date\_str*)  
 **end  
  
 def** *create\_and\_switch\_to\_new\_subscription*(*billing\_provider*, *plan\_id*, *base\_price\_cents*, *per\_order\_price\_cents*, *included\_orders*)  
 *old\_subscription* = **self**.current\_subscription  
 *old\_subscription*.current = **false** *old\_subscription*.was\_current = **true** *new\_subscription* = ***Subscription***.new  
 *new\_subscription*.merchant\_id = **self**.id  
 *new\_subscription*.current = **true** *new\_subscription*.billing\_provider = *billing\_provider  
 new\_subscription*.plan\_id = *plan\_id  
 new\_subscription*.base\_price\_cents = *base\_price\_cents  
 new\_subscription*.per\_order\_price\_cents = *per\_order\_price\_cents  
 new\_subscription*.included\_orders = *included\_orders  
 old\_subscription*.save  
 *new\_subscription*.save  
 **end  
  
 def** *setup\_stripe\_billing\_plan*(*stripe\_billing\_interval*, *stripe\_coupon*=**""**)  
  
 **self**.stripe\_customer  
  
 *stripe\_plan\_id* = find\_or\_create\_stripe\_plan(*stripe\_billing\_interval*)  
  
 **if** valid\_stripe\_coupon\_code?(*stripe\_coupon*)  
 *resp* = **self**.stripe\_customer.create\_subscription(**:plan** *=> stripe\_plan\_id*, **:coupon** *=> stripe\_coupon*)  
 **else** *resp* = **self**.stripe\_customer.create\_subscription(**:plan** *=> stripe\_plan\_id*)  
 **end  
  
 *CacheProcessing***::***FillUserJob***.perform\_async(**self**.id)   
 **end  
  
 def** *valid\_stripe\_coupon\_code?*(*coupon\_code*)  
 **return false if** *coupon\_code*.blank?  
  
 **begin  
 *Stripe***::***Coupon***.retrieve(*coupon\_code*)  
 **rescue *Stripe***::***InvalidRequestError*** *=> e* **return false  
 end  
   
 true  
 end  
  
 def** *find\_or\_create\_stripe\_plan*(*stripe\_billing\_interval*)  
 *stripe\_plan\_id* = **nil** *plan\_name* = **self**.current\_subscription.plan.name  
 *base\_price\_cents* = **self**.current\_subscription.base\_price\_cents  
 *amount* = *stripe\_billing\_interval*.eql?(**"month"**) ? *base\_price\_cents* : *base\_price\_cents*\*12  
 *dollar\_amount* = *base\_price\_cents* / 100  
 *starting\_after\_id* = **nil  
  
 begin** *all\_plans* = ***Stripe***::***Plan***.*all*(**:limit***=>*100, **:starting\_after***=>starting\_after\_id*)  
 *all\_plans*[**"data"**].each **do** |*plan*|  
 *starting\_after\_id* = *plan*[**"id"**]  
 **if** *plan*[**"interval"**].eql?(*stripe\_billing\_interval*) && *plan*[**"amount"**].eql?(*amount*)  
 *stripe\_plan\_id* = *plan*[**"id"**]  
 **break  
 end  
 end  
 break if** *stripe\_plan\_id* **end while** *all\_plans*[**"has\_more"**]  
  
 **unless** *stripe\_plan\_id  
 stripe\_plan\_id* = ***Stripe***::***Plan***.*create*(  
 **:id** *=>* **"**#{*plan\_name*.upcase}**\_**#{*dollar\_amount*}**\_**#{*stripe\_billing\_interval*.upcase}**LY"**,  
 **:interval** *=> stripe\_billing\_interval*,  
 **:currency** *=>* **"USD"**,  
 **:product** *=>* {**:name** *=>* **"**#{*plan\_name*.upcase} #{*dollar\_amount*} #{*stripe\_billing\_interval*.upcase}**LY"**},  
 **:amount** *=> amount*,  
 )  
 **end** *stripe\_plan\_id* **end  
  
 def** *get\_stripe\_billing\_history* **unless self**.stripe\_customer\_id  
 **self**.stripe\_customer  
 **end** *invoices* = ***Stripe***::***Invoice***.*all*(**:customer** *=>* **self**.stripe\_customer\_id, **:date** *=>* {**:gt** *=>* 1.year.ago.to\_i})  
  
 **unless** *invoices* **return** []  
 **end** *flattened\_items* = []  
  
 *invoices*[**"data"**].each **do** |*invoice*|  
 *invoice*[**"lines"**][**"data"**].each **do** |*line*|  
   
 *description* = *line*[**"description"**]  
  
 **if** *line*[**"type"**].eql?(**"subscription"**)  
 *description* = *line*[**"plan"**][**"name"**]  
 **end** *flattened\_items* << ***OpenStruct***.new({  
 **"date"** *=>* ***Time***.at(*invoice*[**"date"**]).strftime(**"%m-%d-%Y"**),  
 **"description"** *=> description*,  
 **"amount"** *=> line*[**"amount"**]  
 })  
 **end  
  
 end** *flattened\_items* **end** *# date\_str =>* ***"2016-10-15"*** *for October 15 2016* **def** *schedule\_invoice\_on*(*date\_str*)  
 **if self**.invoice\_job\_id.present?  
 ***Workers***::***Control***.cancel\_by\_id(**self**.invoice\_job\_id)  
 **end** *invoice\_on* = ***Date***.parse(*date\_str*).to\_datetime + 9.hours  
  
 **self**.invoice\_job\_id = ***AccountProcessing***::***CreateInvoiceJob***.perform\_at(*invoice\_on*, **self**.id)  
 **self**.save  
 **end  
  
 def** *invoice!  
 # do invoice  
  
 subscription* = **self**.current\_subscription  
  
 **self**.swell\_plan\_paid(**self**.current\_mrr\_cents)  
  
 *invoice\_data* = ***Hash***.new  
  
 *invoice\_data*[**:total\_invoice\_cents**] = **self**.current\_mrr\_cents  
  
 *invoice\_data*[**:base\_price\_cents**] = 0  
 *invoice\_data*[**:order\_overage\_cents**] = 0  
 *invoice\_data*[**:redemption\_cents**] = 0  
 *invoice\_data*[**:referrals\_cents**] = 0  
 *invoice\_data*[**:total\_orders**] = 0  
 *invoice\_data*[**:extra\_orders**] = 0  
  
 *invoice\_data*[**:total\_redemption\_cents**] = 0  
 *invoice\_data*[**:total\_referral\_cents**] = 0  
  
  
 *invoice\_data*[**:redemption\_options**] = []  
  
 *invoice\_data*[**:included\_orders**] = *subscription*.included\_orders  
 *invoice\_data*[**:redemption\_percentage**] = *subscription*.percentage\_of\_redemptions  
 *invoice\_data*[**:referral\_percentage**] = *subscription*.percentage\_of\_referrals  
 *invoice\_data*[**:cents\_per\_order**] = *subscription*.per\_order\_price\_cents  
  
  
 **if** *subscription*.manual\_price\_cents  
 *invoice\_data*[**:base\_price\_cents**] = *subscription*.manual\_price\_cents  
 **else** *invoice\_data*[**:base\_price\_cents**] = *subscription*.base\_price\_cents  
 **end** *invoice\_data*[**:total\_orders**] = **self**.thirty\_day\_order\_volume  
 *invoice\_data*[**:extra\_orders**] = *invoice\_data*[**:total\_orders**] - *subscription*.included\_orders  
  
 **if** *invoice\_data*[**:extra\_orders**] > 0  
 *invoice\_data*[**:order\_overage\_cents**] = (*subscription*.per\_order\_price\_cents \* *invoice\_data*[**:extra\_orders**])  
 **else** *invoice\_data*[**:extra\_orders**] = 0  
 **end  
  
 if** *subscription*.percentage\_of\_redemptions > 0  
 **self**.redemption\_options.*order*(**"discount\_amount\_cents ASC"**).each **do** |*redemption\_option*|  
 *data* = ***Hash***.new  
 *data*[**:name**] = *redemption\_option*.name  
 *data*[**:discount\_amount\_cents**] = *redemption\_option*.discount\_amount\_cents  
 *data*[**:quantity**] = *redemption\_option*.point\_redemptions.*where*(**"created\_at >= ?"**, ***Date***.today - 1.month).count  
 *data*[**:total\_discount\_amount\_cents**] = *data*[**:discount\_amount\_cents**] \* *data*[**:quantity**]  
 *data*[**:charge\_amount\_cents**] = ((*subscription*.percentage\_of\_redemptions / 100.0) \* *data*[**:total\_discount\_amount\_cents**]).to\_i  
 *invoice\_data*[**:redemption\_options**] << *data* **end** *invoice\_data*[**:total\_redemption\_cents**] = **self**.point\_redemptions.*joins*(**:redemption\_option**).*where*(**"point\_redemptions.created\_at >= ?"**, ***Date***.today - 1.month).*sum*(**"redemption\_options.discount\_amount\_cents"**)  
 *invoice\_data*[**:redemption\_cents**] = ((*subscription*.percentage\_of\_redemptions / 100.0) \* *invoice\_data*[**:total\_redemption\_cents**]).to\_i  
 **end  
  
 if** *subscription*.percentage\_of\_referrals > 0  
 *invoice\_data*[**:total\_referral\_cents**] = **self**.purchases.*where*(**"created\_at >= ? and referral\_id is not null"**, ***Date***.today - 1.month).*sum*(**:amount\_cents**)  
 *invoice\_data*[**:referrals\_cents**] += ((*subscription*.percentage\_of\_referrals / 100.0) \* *invoice\_data*[**:total\_referral\_cents**]).to\_i  
 **end** *# send email using data  
 invoice\_data*[**:total\_invoice\_cents**] = **"$"** + number\_with\_precision(*invoice\_data*[**:total\_invoice\_cents**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *invoice\_data*[**:base\_price\_cents**] = **"$"** + number\_with\_precision(*invoice\_data*[**:base\_price\_cents**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *invoice\_data*[**:order\_overage\_cents**] = **"$"** + number\_with\_precision(*invoice\_data*[**:order\_overage\_cents**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *invoice\_data*[**:redemption\_cents**] = **"$"** + number\_with\_precision(*invoice\_data*[**:redemption\_cents**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *invoice\_data*[**:referrals\_cents**] = **"$"** + number\_with\_precision(*invoice\_data*[**:referrals\_cents**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *invoice\_data*[**:total\_redemption\_cents**] = **"$"** + number\_with\_precision(*invoice\_data*[**:total\_redemption\_cents**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *invoice\_data*[**:total\_referral\_cents**] = **"$"** + number\_with\_precision(*invoice\_data*[**:total\_referral\_cents**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *invoice\_data*[**:cents\_per\_order**] = **"$"** + number\_with\_precision(*invoice\_data*[**:cents\_per\_order**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
  
 *invoice\_data*[**:redemption\_options**].each **do** |*ro*|  
 *ro*[**:discount\_amount\_cents**] = **"$"** + number\_with\_precision(*ro*[**:discount\_amount\_cents**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *ro*[**:total\_discount\_amount\_cents**] = **"$"** + number\_with\_precision(*ro*[**:total\_discount\_amount\_cents**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *ro*[**:charge\_amount\_cents**] = **"$"** + number\_with\_precision(*ro*[**:charge\_amount\_cents**] / 100.0, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 **end** *content\_body* = ***ActionController***::***Base***.new.*render\_to\_string*(**"emails/invoices/admin.html.erb"**, **:layout** *=>* **false**, **:locals** *=> invoice\_data*)  
 *content* = ***ActionController***::***Base***.new.*render\_to\_string*(**:text** *=> content\_body*, **:layout** *=>* **false**, **:locals** *=> invoice\_data*)  
  
 *mail* = ***SendGrid***::***Mail***.new  
 *mail*.from = ***SendGrid***::***Email***.new(**email**: **"james@swellrewards.com"**, **name**: **"Swell Invoice"**)  
 *mail*.subject = **"[Swell Invoice] -** #{**self**.company\_name} **-** #{***Date***.today} **-** #{*invoice\_data*[**:total\_invoice\_cents**]}**"** *mail*.contents = ***SendGrid***::***Content***.new(**type**: **'text/html'**, **value**: *content*)  
  
 *personalization* = ***SendGrid***::***Personalization***.new  
   
 [**"james"**,**"josh"**,**"dj"**,**"dzenita"**,**"nikhil"**,**"alison"**,**"melissa"**,**"mike"**,**"stewart"**,**"samuel"**].each **do** |*member*|   
 *personalization*.to = ***SendGrid***::***Email***.new(**email**: **"**#{*member*}**@swellrewards.com"**)  
 **end** *mail*.personalizations = *personalization  
 sg* = ***SendGrid***::***API***.new(**api\_key**: ***ENV***[**'SENDGRID\_API\_KEY'**])  
 *response* = *sg*.client.mail.\_(**'send'**).post(**request\_body**: *mail*.to\_json)  
  
 **self**.schedule\_invoice\_on((***Date***.today + 1.month).to\_s)  
 **end  
  
 def self**.*take\_swell\_snapshot  
 today* = ***Date***.today  
 *yesterday* = ***Date***.today - 1.day  
  
 *today\_snapshot* = ***SwellSnapshot***.find\_or\_create\_by(**:day** *=> today*)  
 *compare\_snapshot* = ***SwellSnapshot***.*where*(**:day** *=> yesterday*).take  
  
  
 *subscriptions* = ***Subscription***.*where*(**"plan\_id >= 2 and current=true"**)  
 *today\_snapshot*.mrr\_cents = ***Merchant***.total\_mrr\_cents  
  
 *expansion\_cents*, *contraction\_cents* = ***Merchant***.expansion\_contraction\_mrr\_cents  
  
 *today\_snapshot*.expansion\_cents = *expansion\_cents  
 today\_snapshot*.contraction\_cents = *contraction\_cents  
 today\_snapshot*.base\_mrr\_cents = *subscriptions*.*sum*(**:base\_price\_cents**)  
 *today\_snapshot*.subscriptions = *subscriptions*.count  
 *today\_snapshot*.shopify\_subscriptions = *subscriptions*.*where*(**"billing\_provider='shopify'"**).count  
 *today\_snapshot*.silver = *subscriptions*.*where*(**"plan\_id = 2"**).count  
 *today\_snapshot*.gold = *subscriptions*.*where*(**"plan\_id = 3"**).count  
 *today\_snapshot*.platinum = *subscriptions*.*where*(**"plan\_id = 4"**).count  
 *today\_snapshot*.enterprise = *subscriptions*.*where*(**"plan\_id = 5"**).count  
  
 *today\_snapshot*.installed\_users = ***Merchant***.*where*(**"has\_rewards\_installed = true"**).count  
 *today\_snapshot*.installed\_users\_over\_100 = ***Merchant***.*where*(**"has\_rewards\_installed = true and install\_order\_volume >= 100"**).count  
  
 *today\_snapshot*.live = ***Merchant***.*where*(**"has\_rewards\_installed=true and program\_live=true"**).count  
 *today\_snapshot*.locked = ***Merchant***.*where*(**"has\_rewards\_installed=true and program\_locked=true"**).count  
 *today\_snapshot*.live\_unlocked = ***Merchant***.*where*(**"has\_rewards\_installed=true and program\_locked=false and program\_live=true"**).count  
  
 **if** *today\_snapshot*.subscriptions > 0  
 *today\_snapshot*.arpu\_cents = *today\_snapshot*.mrr\_cents / *today\_snapshot*.subscriptions  
 **else** *today\_snapshot*.arpu\_cents = 0  
 **end** *today\_snapshot*.arr\_cents = *today\_snapshot*.mrr\_cents \* 12  
  
 *today\_snapshot*.trial\_over\_100 = 0  
 *today\_snapshot*.trial\_under\_100 = 0  
  
 ***Merchant***.*where*(**"has\_rewards\_installed=true and on\_free\_trial=true"**).each **do** |*trial\_merchant*|  
 **if** *trial\_merchant*.thirty\_day\_order\_volume >= 100  
 *today\_snapshot*.trial\_over\_100 += 1  
 **else** *today\_snapshot*.trial\_under\_100 += 1  
 **end  
 end** *today\_snapshot*.save  
  
 **unless** *compare\_snapshot  
 compare\_snapshot* = *today\_snapshot*.dup  
 **end  
  
 *Merchant***.send\_snapshot\_comparison\_email(*today\_snapshot*, *compare\_snapshot*)  
 **end  
  
 def self**.*send\_snapshot\_comparison\_email*(*current\_snapshot*, *compare\_snapshot*)  
 *# mrr\_cents  
 mrr\_cents\_change* = *current\_snapshot*.mrr\_cents - *compare\_snapshot*.mrr\_cents  
  
 **if** *compare\_snapshot*.mrr\_cents > 0  
 *mrr\_cents\_change\_percent* = ((*mrr\_cents\_change* / *compare\_snapshot*.mrr\_cents.to\_f) \* 100).round(2)  
 **else** *mrr\_cents\_change\_percent* = 0.00  
 **end** *expansion\_cents\_change* = *current\_snapshot*.expansion\_cents - *compare\_snapshot*.expansion\_cents  
  
 **if** *compare\_snapshot*.expansion\_cents > 0  
 *expansion\_cents\_change\_percent* = ((*expansion\_cents\_change* / *compare\_snapshot*.expansion\_cents.to\_f) \* 100).round(2)  
 **else** *expansion\_cents\_change\_percent* = 0.00  
 **end** *contraction\_cents\_change* = *current\_snapshot*.contraction\_cents - *compare\_snapshot*.contraction\_cents  
  
 **if** *compare\_snapshot*.contraction\_cents > 0  
 *contraction\_cents\_change\_percent* = ((*contraction\_cents\_change* / *compare\_snapshot*.contraction\_cents.to\_f) \* 100).round(2)  
 **else** *contraction\_cents\_change\_percent* = 0.00  
 **end** *base\_mrr\_cents\_change* = *current\_snapshot*.base\_mrr\_cents - *compare\_snapshot*.base\_mrr\_cents  
  
 **if** *compare\_snapshot*.base\_mrr\_cents > 0  
 *base\_mrr\_cents\_change\_percent* = ((*base\_mrr\_cents\_change* / *compare\_snapshot*.base\_mrr\_cents.to\_f) \* 100).round(2)  
 **else** *base\_mrr\_cents\_change\_percent* = 0.00  
 **end** *# subscriptions  
 subscriptions\_change* = *current\_snapshot*.subscriptions - *compare\_snapshot*.subscriptions  
  
 **if** *compare\_snapshot*.subscriptions > 0  
 *subscriptions\_change\_percent* = ((*subscriptions\_change* / *compare\_snapshot*.subscriptions.to\_f) \* 100).round(2)  
 **else** *subscriptions\_change\_percent* = 0.00  
 **end** *# silver  
 silver\_change* = *current\_snapshot*.silver - *compare\_snapshot*.silver  
  
 **if** *compare\_snapshot*.silver > 0  
 *silver\_change\_percent* = ((*silver\_change* / *compare\_snapshot*.silver.to\_f)\*100).round(2)  
 **else** *silver\_change\_percent* = 0.00  
 **end** *# gold  
 gold\_change* = *current\_snapshot*.gold - *compare\_snapshot*.gold  
  
 **if** *compare\_snapshot*.gold > 0  
 *gold\_change\_percent* = ((*gold\_change* / *compare\_snapshot*.gold.to\_f) \* 100).round(2)  
 **else** *gold\_change\_percent* = 0.00  
 **end** *# platinum  
 platinum\_change* = *current\_snapshot*.platinum - *compare\_snapshot*.platinum  
  
 **if** *compare\_snapshot*.platinum > 0  
 *platinum\_change\_percent* = ((*platinum\_change* / *compare\_snapshot*.platinum.to\_f) \* 100).round(2)  
 **else** *platinum\_change\_percent* = 0.00  
 **end** *# enterprise  
 enterprise\_change* = *current\_snapshot*.enterprise - *compare\_snapshot*.enterprise  
  
 **if** *compare\_snapshot*.enterprise > 0  
 *enterprise\_change\_percent* = ((*enterprise\_change* / *compare\_snapshot*.enterprise.to\_f) \* 100).round(2)  
 **else** *enterprise\_change\_percent* = 0.00  
 **end** *# installed\_users  
 installed\_users\_change* = *current\_snapshot*.installed\_users - *compare\_snapshot*.installed\_users  
  
 **if** *compare\_snapshot*.installed\_users > 0  
 *installed\_users\_change\_percent* = ((*installed\_users\_change* / *compare\_snapshot*.installed\_users.to\_f) \* 100).round(2)  
 **else** *installed\_users\_change\_percent* = 0.00  
 **end** *# installed\_users\_over\_100  
 installed\_users\_over\_100\_change* = *current\_snapshot*.installed\_users\_over\_100 - *compare\_snapshot*.installed\_users\_over\_100  
  
 **if** *compare\_snapshot*.installed\_users\_over\_100 > 0  
 *installed\_users\_over\_100\_change\_percent* = ((*installed\_users\_over\_100\_change* / *compare\_snapshot*.installed\_users\_over\_100.to\_f) \* 100).round(2)  
 **else** *installed\_users\_over\_100\_change\_percent* = 0.00  
 **end** *# live  
 live\_change* = *current\_snapshot*.live - *compare\_snapshot*.live  
  
 **if** *compare\_snapshot*.live > 0  
 *live\_change\_percent* = ((*live\_change* / *compare\_snapshot*.live.to\_f) \* 100).round(2)  
 **else** *live\_change\_percent* = 0.00  
 **end** *# locked  
 locked\_change* = *current\_snapshot*.locked - *compare\_snapshot*.locked  
  
 **if** *compare\_snapshot*.locked > 0  
 *locked\_change\_percent* = ((*locked\_change* / *compare\_snapshot*.locked.to\_f) \* 100).round(2)  
 **else** *locked\_change\_percent* = 0.00  
 **end** *# live\_unlocked  
 live\_unlocked\_change* = *current\_snapshot*.live\_unlocked - *compare\_snapshot*.live\_unlocked  
  
 **if** *compare\_snapshot*.live\_unlocked > 0  
 *live\_unlocked\_change\_percent* = ((*live\_unlocked\_change* / *compare\_snapshot*.live\_unlocked.to\_f) \* 100).round(2)  
 **else** *live\_unlocked\_change\_percent* = 0.00  
 **end** *# trial\_under\_100  
 trial\_under\_100\_change* = *current\_snapshot*.trial\_under\_100 - *compare\_snapshot*.trial\_under\_100  
  
 **if** *compare\_snapshot*.trial\_under\_100 > 0  
 *trial\_under\_100\_change\_percent* = ((*trial\_under\_100\_change* / *compare\_snapshot*.trial\_under\_100.to\_f) \* 100).round(2)  
 **else** *trial\_under\_100\_change\_percent* = 0.00  
 **end** *# trial\_over\_100  
 trial\_over\_100\_change* = *current\_snapshot*.trial\_over\_100 - *compare\_snapshot*.trial\_over\_100  
  
 **if** *compare\_snapshot*.trial\_over\_100 > 0  
 *trial\_over\_100\_change\_percent* = ((*trial\_over\_100\_change* / *compare\_snapshot*.trial\_over\_100.to\_f) \* 100).round(2)  
 **else** *trial\_over\_100\_change\_percent* = 0.00  
 **end** *# arpu\_cents  
 arpu\_cents\_change* = *current\_snapshot*.arpu\_cents - *compare\_snapshot*.arpu\_cents  
  
 **if** *compare\_snapshot*.arpu\_cents > 0  
 *arpu\_cents\_change\_percent* = ((*arpu\_cents\_change* / *compare\_snapshot*.arpu\_cents.to\_f) \* 100).round(2)  
 **else** *arpu\_cents\_change\_percent* = 0.00  
 **end** *# arr\_cents  
 arr\_cents\_change* = *current\_snapshot*.arr\_cents - *compare\_snapshot*.arr\_cents  
  
 **if** *compare\_snapshot*.arr\_cents > 0  
 *arr\_cents\_change\_percent* = ((*arr\_cents\_change* / *compare\_snapshot*.arr\_cents.to\_f) \* 100).round(2)  
 **else** *arr\_cents\_change\_percent* = 0.00  
 **end** *# shopify\_subscriptions  
 shopify\_subscriptions\_change* = *current\_snapshot*.shopify\_subscriptions - *compare\_snapshot*.shopify\_subscriptions  
  
 **if** *compare\_snapshot*.shopify\_subscriptions > 0  
 *shopify\_subscriptions\_change\_percent* = ((*shopify\_subscriptions\_change* / *compare\_snapshot*.shopify\_subscriptions.to\_f) \* 100).round(2)  
 **else** *shopify\_subscriptions\_change\_percent* = 0.00  
 **end** *local\_data* = ***Hash***.new  
  
 *local\_data*[**:new\_premiums**] = []  
 *local\_data*[**:lost\_premiums**] = []  
  
 *local\_data*[**:new\_installs**] = ***Merchant***.*where*(**"created\_at >= ?"**, ***Time***.now - 24.hours).count  
  
 *local\_data*[**:new\_installs\_sub\_10**] = ***Merchant***.*where*(**"created\_at >= ? and install\_order\_volume is not null and install\_order\_volume < 10"**, ***Time***.now - 24.hours).count  
 *local\_data*[**:new\_installs\_above\_10\_below\_100**] = ***Merchant***.*where*(**"created\_at >= ? and install\_order\_volume is not null and install\_order\_volume >= 10 and install\_order\_volume < 100"**, ***Time***.now - 24.hours).count  
 *local\_data*[**:new\_installs\_above\_100**] = ***Merchant***.*where*(**"created\_at >= ? and install\_order\_volume is not null and install\_order\_volume >= 100"**, ***Time***.now - 24.hours).count  
 *local\_data*[**:new\_installs\_max**] = ***Merchant***.*where*(**"install\_order\_volume is not null and created\_at >= ?"**, ***Time***.now - 24.hours).*pluck*(**:install\_order\_volume**).max  
  
 *local\_data*[**:orders\_processed**] = ***Purchase***.*where*(**"created\_at >= ?"**, ***Time***.now - 24.hours).count  
  
  
 **if** *local\_data*[**:new\_installs**] > 0  
 *local\_data*[**:new\_installs\_average\_order\_volume**] = ***Merchant***.*where*(**"install\_order\_volume is not null and created\_at >= ?"**, ***Time***.now - 24.hours).*sum*(**:install\_order\_volume**) / (*local\_data*[**:new\_installs**] \* 1.0)  
 **else** *local\_data*[**:new\_installs\_average\_order\_volume**] = 0  
 **end  
  
 *Subscription***.*where*(**"plan\_id >= 2 and current=true and created\_at >= ?"**, ***Time***.now - 24.hours).all.each **do** |*subscription*|  
 *sub* = ***Hash***.new  
 *sub*[**:plan\_name**] = *subscription*.plan.name  
 *sub*[**:base\_price\_cents**] = *subscription*.base\_price\_cents  
 *sub*[**:total\_price\_cents**] = *subscription*.base\_price\_cents  
 *sub*[**:per\_order\_price\_cents**] = *subscription*.per\_order\_price\_cents  
 *sub*[**:included\_orders**] = *subscription*.included\_orders  
 *sub*[**:billing\_provider**] = *subscription*.billing\_provider  
 *sub*[**:merchant\_name**] = *subscription*.merchant.company\_name  
 *sub*[**:thirty\_day\_order\_volume**] = *subscription*.merchant.thirty\_day\_order\_volume  
 *sub*[**:platform**] = *subscription*.merchant.platform  
 *sub*[**:days\_installed**] = (***Date***.today - *subscription*.merchant.created\_at.to\_date).to\_i  
  
 *extra\_orders* = *sub*[**:thirty\_day\_order\_volume**] - *sub*[**:included\_orders**]  
  
 **if** *extra\_orders* > 0  
 *sub*[**:total\_price\_cents**] += *extra\_orders* \* *sub*[**:per\_order\_price\_cents**]  
 **end** *local\_data*[**:new\_premiums**] << *sub* **end  
  
 *Subscription***.*where*(**"plan\_id >= 2 and was\_current=true and current=false and updated\_at >= ?"**, ***Time***.now - 24.hours).all.each **do** |*subscription*|  
 *sub* = ***Hash***.new  
 *sub*[**:plan\_name**] = *subscription*.plan.name  
 *sub*[**:base\_price\_cents**] = *subscription*.base\_price\_cents  
 *sub*[**:total\_price\_cents**] = *subscription*.base\_price\_cents  
 *sub*[**:per\_order\_price\_cents**] = *subscription*.per\_order\_price\_cents  
 *sub*[**:included\_orders**] = *subscription*.included\_orders  
 *sub*[**:billing\_provider**] = *subscription*.billing\_provider  
 *sub*[**:merchant\_name**] = *subscription*.merchant.company\_name  
 *sub*[**:thirty\_day\_order\_volume**] = *subscription*.merchant.thirty\_day\_order\_volume  
 *sub*[**:platform**] = *subscription*.merchant.platform  
 *sub*[**:days\_installed**] = (***Date***.today - *subscription*.merchant.created\_at.to\_date).to\_i  
  
 *extra\_orders* = *sub*[**:thirty\_day\_order\_volume**] - *sub*[**:included\_orders**]  
  
 **if** *extra\_orders* > 0  
 *sub*[**:total\_price\_cents**] += *extra\_orders* \* *sub*[**:per\_order\_price\_cents**]  
 **end** *local\_data*[**:lost\_premiums**] << *sub* **end** *local\_data*[**:adjusted\_mrr\_cents**] = ***Merchant***.adjusted\_total\_mrr\_cents(**"2016-08-01"**, **"2016-08-31"**)  
 *local\_data*[**:adjusted\_mrr\_cents**] = **"$"** + number\_with\_precision(*local\_data*[**:adjusted\_mrr\_cents**] / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
  
 *local\_data*[**:mrr\_cents**] = **"$"** + number\_with\_precision(*current\_snapshot*.mrr\_cents / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:expansion\_cents**] = **"$"** + number\_with\_precision(*current\_snapshot*.expansion\_cents / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:contraction\_cents**] = **"$"** + number\_with\_precision(*current\_snapshot*.contraction\_cents / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:base\_mrr\_cents**] = **"$"** + number\_with\_precision(*current\_snapshot*.base\_mrr\_cents / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:arr\_cents**] = **"$"** + number\_with\_precision(*current\_snapshot*.arr\_cents / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:arpu\_cents**] = **"$"** + number\_with\_precision(*current\_snapshot*.arpu\_cents / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:subscriptions**] = number\_with\_delimiter(*current\_snapshot*.subscriptions)  
 *local\_data*[**:installed\_users**] = number\_with\_delimiter(*current\_snapshot*.installed\_users)  
 *local\_data*[**:installed\_users\_over\_100**] = number\_with\_delimiter(*current\_snapshot*.installed\_users\_over\_100)  
 *local\_data*[**:silver**] = number\_with\_delimiter(*current\_snapshot*.silver)  
 *local\_data*[**:gold**] = number\_with\_delimiter(*current\_snapshot*.gold)  
 *local\_data*[**:platinum**] = number\_with\_delimiter(*current\_snapshot*.platinum)  
 *local\_data*[**:enterprise**] = number\_with\_delimiter(*current\_snapshot*.enterprise)  
 *local\_data*[**:shopify\_subscriptions**] = number\_with\_delimiter(*current\_snapshot*.shopify\_subscriptions)  
 *local\_data*[**:live**] = number\_with\_delimiter(*current\_snapshot*.live)  
 *local\_data*[**:locked**] = number\_with\_delimiter(*current\_snapshot*.locked)  
 *local\_data*[**:live\_unlocked**] = number\_with\_delimiter(*current\_snapshot*.live\_unlocked)  
 *local\_data*[**:trial\_over\_100**] = number\_with\_delimiter(*current\_snapshot*.trial\_over\_100)  
 *local\_data*[**:trial\_under\_100**] = number\_with\_delimiter(*current\_snapshot*.trial\_under\_100)  
  
 *local\_data*[**:mrr\_cents\_change**] = **"$"** + number\_with\_precision(*mrr\_cents\_change* / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:expansion\_cents\_change**] = **"$"** + number\_with\_precision(*expansion\_cents\_change* / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:contraction\_cents\_change**] = **"$"** + number\_with\_precision(*contraction\_cents\_change* / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:base\_mrr\_cents\_change**] = **"$"** + number\_with\_precision(*base\_mrr\_cents\_change* / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:arr\_cents\_change**] = **"$"** + number\_with\_precision(*arr\_cents\_change* / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:arpu\_cents\_change**] = **"$"** + number\_with\_precision(*arpu\_cents\_change* / 100, **:precision** *=>* 2, **:delimiter** *=>* **','**)  
 *local\_data*[**:subscriptions\_change**] = number\_with\_delimiter(*subscriptions\_change*)  
 *local\_data*[**:installed\_users\_change**] = number\_with\_delimiter(*installed\_users\_change*)  
 *local\_data*[**:installed\_users\_over\_100\_change**] = number\_with\_delimiter(*installed\_users\_over\_100\_change*)  
 *local\_data*[**:silver\_change**] = number\_with\_delimiter(*silver\_change*)  
 *local\_data*[**:gold\_change**] = number\_with\_delimiter(*gold\_change*)  
 *local\_data*[**:platinum\_change**] = number\_with\_delimiter(*platinum\_change*)  
 *local\_data*[**:enterprise\_change**] = number\_with\_delimiter(*enterprise\_change*)  
 *local\_data*[**:shopify\_subscriptions\_change**] = number\_with\_delimiter(*shopify\_subscriptions\_change*)  
 *local\_data*[**:live\_change**] = number\_with\_delimiter(*live\_change*)  
 *local\_data*[**:locked\_change**] = number\_with\_delimiter(*locked\_change*)  
 *local\_data*[**:live\_unlocked\_change**] = number\_with\_delimiter(*live\_unlocked\_change*)  
 *local\_data*[**:trial\_over\_100\_change**] = number\_with\_delimiter(*trial\_over\_100\_change*)  
 *local\_data*[**:trial\_under\_100\_change**] = number\_with\_delimiter(*trial\_under\_100\_change*)  
  
 *local\_data*[**:mrr\_cents\_change\_percent**] = *mrr\_cents\_change\_percent  
 local\_data*[**:expansion\_cents\_change\_percent**] = *expansion\_cents\_change\_percent  
 local\_data*[**:contraction\_cents\_change\_percent**] = *contraction\_cents\_change\_percent  
 local\_data*[**:base\_mrr\_cents\_change\_percent**] = *base\_mrr\_cents\_change\_percent  
 local\_data*[**:arr\_cents\_change\_percent**] = *arr\_cents\_change\_percent  
 local\_data*[**:arpu\_cents\_change\_percent**] = *arpu\_cents\_change\_percent  
 local\_data*[**:subscriptions\_change\_percent**] = *subscriptions\_change\_percent  
 local\_data*[**:installed\_users\_change\_percent**] = *installed\_users\_change\_percent  
 local\_data*[**:installed\_users\_over\_100\_change\_percent**] = *installed\_users\_over\_100\_change\_percent  
 local\_data*[**:silver\_change\_percent**] = *silver\_change\_percent  
 local\_data*[**:gold\_change\_percent**] = *gold\_change\_percent  
 local\_data*[**:platinum\_change\_percent**] = *platinum\_change\_percent  
 local\_data*[**:enterprise\_change\_percent**] = *enterprise\_change\_percent  
 local\_data*[**:shopify\_subscriptions\_change\_percent**] = *shopify\_subscriptions\_change\_percent  
 local\_data*[**:live\_change\_percent**] = *live\_change\_percent  
 local\_data*[**:locked\_change\_percent**] = *locked\_change\_percent  
 local\_data*[**:live\_unlocked\_change\_percent**] = *live\_unlocked\_change\_percent  
 local\_data*[**:trial\_over\_100\_change\_percent**] = *trial\_over\_100\_change\_percent  
 local\_data*[**:trial\_under\_100\_change\_percent**] = *trial\_under\_100\_change\_percent  
  
 local\_data*[**:report**] = ***SubscriptionReportService***.new.generate\_report(***Time***.now.beginning\_of\_month.to\_date)  
  
 *content\_body* = ***ActionController***::***Base***.new.*render\_to\_string*(**"emails/digests/swell.html.erb"**, **:layout** *=>* **false**, **:locals** *=> local\_data*)  
 *content* = ***ActionController***::***Base***.new.*render\_to\_string*(**:text** *=> content\_body*, **:layout** *=>* **false**, **:locals** *=> local\_data*)  
  
 *mail* = ***SendGrid***::***Mail***.new  
 *mail*.from = ***SendGrid***::***Email***.new(**email**: **"james@swellrewards.com"**, **name**: **"Swell Stats"**)  
 *mail*.subject = **"[Swell Stats] -** #{***Date***.today}**"** *mail*.contents = ***SendGrid***::***Content***.new(**type**: **'text/html'**, **value**: *content*)  
 *personalization* = ***SendGrid***::***Personalization***.new  
   
 [**"james"**,**"josh"**,**"dj"**,**"dzenita"**,**"nikhil"**,**"alison"**,**"melissa"**,**"mike"**,**"stewart"**,**"samuel"**].each **do** |*member*|   
 *personalization*.to = ***SendGrid***::***Email***.new(**email**: **"**#{*member*}**@swellrewards.com"**)  
 **end** *mail*.personalizations = *personalization  
 sg* = ***SendGrid***::***API***.new(**api\_key**: ***ENV***[**'SENDGRID\_API\_KEY'**])  
 *response* = *sg*.client.mail.\_(**'send'**).post(**request\_body**: *mail*.to\_json)  
  
 **end  
  
 def self**.*expansion\_contraction\_mrr\_cents  
 expansion\_cents* = 0  
 *contraction\_cents* = 0  
  
 *bronze* = ***Plan***.*where*(**:name** *=>* **"BRONZE"**).take  
  
 *# who are paying us or who have ever paid us  
 merchant\_ids* = ***Subscription***.*where*(**"plan\_id != "** + *bronze*.id.to\_s).*pluck*(**:merchant\_id**).uniq  
  
 *merchant\_ids*.each **do** |*merchant\_id*|  
 *merchant* = ***Merchant***.*where*(**id**: *merchant\_id*).take  
  
 **if** *merchant*.present?  
 *old\_mrr* = *merchant*.yesterdays\_mrr\_cents  
 *mrr* = *merchant*.current\_mrr\_cents  
  
 *change\_in\_mrr* = *mrr* - *old\_mrr* **if** *change\_in\_mrr* < 0  
 *contraction\_cents* += *change\_in\_mrr* **else** *expansion\_cents* += *change\_in\_mrr* **end  
 end  
 end** [*expansion\_cents*, *contraction\_cents*]  
 **end  
  
 def self**.*total\_mrr\_cents  
 mrr\_cents* = 0  
  
 *bronze* = ***Plan***.*where*(**:name** *=>* **"BRONZE"**).take  
  
 ***Subscription***.*includes*(**:merchant**).*where*(**"current = true and plan\_id != "** + *bronze*.id.to\_s).each **do** |*subscription*|  
 *mrr\_cents* += *subscription*.merchant.current\_mrr\_cents  
 **end** *mrr\_cents* **end  
  
 def** *yesterdays\_mrr\_cents* **return** 0 **if self**.plan.eql?(**"staff"**)  
  
 *subscription* = **self**.yesterdays\_subscription  
 *mrr\_cents* = 0  
  
 **if** *subscription*.manual\_price\_cents  
 *mrr\_cents* = *subscription*.manual\_price\_cents  
 **else** *mrr\_cents* = *subscription*.base\_price\_cents  
 **end** *orders* = **self**.yesterdays\_thirty\_day\_order\_volume  
 *extra\_orders* = *orders* - *subscription*.included\_orders  
  
 **if** *extra\_orders* > 0  
 *mrr\_cents* += (*subscription*.per\_order\_price\_cents \* *extra\_orders*)  
 **end** *mrr\_cents* **end  
  
 def self**.*adjusted\_total\_mrr\_cents*(*start\_date*, *end\_date*)  
 *mrr\_cents* = 0  
  
 *bronze* = ***Plan***.*where*(**:name** *=>* **"BRONZE"**).take  
  
 ***Subscription***.*includes*(**:merchant**).*where*(**"current = true and plan\_id != "** + *bronze*.id.to\_s).each **do** |*subscription*|  
 *mrr\_cents* += *subscription*.merchant.adjusted\_mrr\_cents(*start\_date*, *end\_date*)  
 **end** *mrr\_cents* **end  
  
 def** *adjusted\_mrr\_cents*(*start\_date*, *end\_date*)  
 **return** 0 **if self**.plan.eql?(**"staff"**)  
  
 *subscription* = **self**.current\_subscription  
 *mrr\_cents* = 0  
  
 **if** *subscription*.manual\_price\_cents  
 *mrr\_cents* = *subscription*.manual\_price\_cents  
 **else** *mrr\_cents* = *subscription*.base\_price\_cents  
 **end** *orders* = **self**.purchases.*where*(**"created\_at >= ? and created\_at <= ? and amount\_cents > 0"**, *start\_date*, *end\_date*).count  
 *extra\_orders* = *orders* - *subscription*.included\_orders  
  
 **if** *extra\_orders* > 0  
 *mrr\_cents* += (*subscription*.per\_order\_price\_cents \* *extra\_orders*)  
 **end  
  
 if** *subscription*.percentage\_of\_redemptions > 0  
 *redemption\_cents* = **self**.point\_redemptions.*joins*(**:redemption\_option**).*where*(**"point\_redemptions.created\_at >= ?"**, ***Date***.today - 1.month).*sum*(**"redemption\_options.discount\_amount\_cents"**)  
 *mrr\_cents* += ((*subscription*.percentage\_of\_redemptions / 100.0) \* *redemption\_cents*).to\_i  
 **end  
  
 if** *subscription*.percentage\_of\_referrals > 0  
 *referral\_cents* = **self**.purchases.*where*(**"created\_at >= ? and referral\_id is not null"**, ***Date***.today - 1.month).*sum*(**:amount\_cents**)  
 *mrr\_cents* += ((*subscription*.percentage\_of\_referrals / 100.0) \* *referral\_cents*).to\_i  
 **end** *mrr\_cents* **end  
  
 def** *adjusted\_mrr\_cents\_with\_subscription*(*start\_date*, *end\_date*, *subscription*)  
 **return** 0 **if self**.plan.eql?(**"staff"**)  
  
 *mrr\_cents* = 0  
  
 **if** *subscription*.manual\_price\_cents  
 *mrr\_cents* = *subscription*.manual\_price\_cents  
 **else** *mrr\_cents* = *subscription*.base\_price\_cents  
 **end** *orders* = **self**.purchases.*where*(**"created\_at >= ? and created\_at <= ? and amount\_cents > 0"**, *start\_date*, *end\_date*).count  
 *extra\_orders* = *orders* - *subscription*.included\_orders  
  
 **if** *extra\_orders* > 0  
 *mrr\_cents* += (*subscription*.per\_order\_price\_cents \* *extra\_orders*)  
 **end  
  
 if** *subscription*.percentage\_of\_redemptions > 0  
 *redemption\_cents* = **self**.point\_redemptions.*joins*(**:redemption\_option**).*where*(**"point\_redemptions.created\_at >= ?"**, ***Date***.today - 1.month).*sum*(**"redemption\_options.discount\_amount\_cents"**)  
 *mrr\_cents* += ((*subscription*.percentage\_of\_redemptions / 100.0) \* *redemption\_cents*).to\_i  
 **end  
  
 if** *subscription*.percentage\_of\_referrals > 0  
 *referral\_cents* = **self**.purchases.*where*(**"created\_at >= ? and referral\_id is not null"**, ***Date***.today - 1.month).*sum*(**:amount\_cents**)  
 *mrr\_cents* += ((*subscription*.percentage\_of\_referrals / 100.0) \* *referral\_cents*).to\_i  
 **end** *mrr\_cents* **end  
  
 def** *current\_mrr\_cents* **return** 0 **if self**.plan.eql?(**"staff"**)  
  
 *subscription* = **self**.current\_subscription  
 *mrr\_cents* = 0  
  
 **if** *subscription*.manual\_price\_cents  
 *mrr\_cents* = *subscription*.manual\_price\_cents  
 **else** *mrr\_cents* = *subscription*.base\_price\_cents  
 **end** *orders* = **self**.thirty\_day\_order\_volume  
 *extra\_orders* = *orders* - *subscription*.included\_orders  
  
 **if** *extra\_orders* > 0  
 *mrr\_cents* += (*subscription*.per\_order\_price\_cents \* *extra\_orders*)  
 **end  
  
 if** *subscription*.percentage\_of\_redemptions > 0  
 *redemption\_cents* = **self**.point\_redemptions.*joins*(**:redemption\_option**).*where*(**"point\_redemptions.created\_at >= ?"**, ***Date***.today - 1.month).*sum*(**"redemption\_options.discount\_amount\_cents"**)  
 *mrr\_cents* += ((*subscription*.percentage\_of\_redemptions / 100.0) \* *redemption\_cents*).to\_i  
 **end  
  
 if** *subscription*.percentage\_of\_referrals > 0  
 *referral\_cents* = **self**.purchases.*where*(**"created\_at >= ? and referral\_id is not null"**, ***Date***.today - 1.month).*sum*(**:amount\_cents**)  
 *mrr\_cents* += ((*subscription*.percentage\_of\_referrals / 100.0) \* *referral\_cents*).to\_i  
 **end  
  
 if** *subscription*.billing\_provider.eql?(**"shopify"**)  
 *mrr\_cents* = [*mrr\_cents*, *subscription*.plan.capped\_amount\_cents].min  
 **end** *mrr\_cents* **end  
  
 def** *fill\_cache  
 cards* = **self**.credit\_cards  
 *subs* = **self**.stripe\_subscriptions  
 *dci* = **self**.default\_card\_id  
 **end  
  
 def** *credit\_cards* ***Rails***.cache.fetch(**"merchants/**#{id}**/stripe\_cards"**)  
 **if self**.stripe\_customer\_id.present?  
 **self**.stripe\_customer.sources.*all*(**:object** *=>* **"card"**).data  
 **else** []  
 **end  
 end  
  
 def** *default\_card\_id* ***Rails***.cache.fetch(**"merchants/**#{id}**/stripe\_default\_card"**) **do  
 if self**.stripe\_customer\_id.present?  
 **self**.stripe\_customer.default\_source  
 **else  
 nil  
 end  
 end  
 end  
  
 def** *set\_default\_card*(*card\_id*)  
 *sc* = **self**.stripe\_customer  
 *sc*.default\_source = *card\_id  
 sc*.save  
 ***Rails***.cache.delete(**"merchants/**#{id}**/stripe\_default\_card"**)  
 **self**.default\_card\_id  
 **end  
  
 def** *add\_stripe\_card\_from\_token*(*token*)  
 *card* = **self**.stripe\_customer.sources.*create*(**:source** *=> token*)  
 **self**.set\_default\_card(*card*.id)  
 **self**.save  
 ***Rails***.cache.delete(**"merchants/**#{id}**/stripe\_cards"**)  
 *card* **end  
  
 def** *remove\_stripe\_card\_by\_id*(*id*)  
 **self**.stripe\_customer.sources.retrieve(*id*).delete  
 ***Rails***.cache.delete(**"merchants/**#{*id*}**/stripe\_cards"**)  
 **self**.credit\_cards  
 **end  
  
 def** *default\_card* **self**.credit\_cards.select { |*card*| *card* **if** *card*.id.eql?(**self**.default\_card\_id) }.first  
 **end  
  
 def** *stripe\_subscriptions* ***Rails***.cache.fetch(**"merchants/**#{id}**/stripe\_subscriptions"**) **do  
 if self**.stripe\_customer\_id.present?  
 **self**.stripe\_customer.subscriptions.data  
 **else** []  
 **end  
 end  
 end  
  
 def** *yesterdays\_subscription  
 current* = **self**.current\_subscription  
 *yesterday* = **nil  
  
 if** *current*.created\_at < (***Time***.now - 24.hours)  
 *yesterday* = *current* **else** *yesterday* = **self**.subscriptions.*where*(**"current = false and was\_current = true and updated\_at >= ?"**, ***Time***.now - 24.hours).*order*(**"updated\_at ASC"**).first  
  
 **if** *yesterday*.nil?  
 *yesterday* = *current* **end  
 end** *yesterday* **end  
  
 def** *current\_subscription  
 current\_subscription* = **self**.subscriptions.*where*(**:current** *=>* **true**).take  
 **if** *current\_subscription*.present?  
 *current\_subscription* **else** *subscription* = **self**.subscriptions.*where*(**:was\_current** *=>* **true**).*order*(**updated\_at**: **:desc**).first  
   
 **if** *subscription*.present?  
 *subscription*.current = **true** *subscription*.save  
 *subscription* **end  
 end  
 end  
  
 def** *active\_subscription* **self**.stripe\_subscriptions.first  
 **end  
  
 def** *switch\_to\_stripe\_billing  
 customer* = ***Stripe***::***Customer***.*create*(**:email** *=>* **self**.contact\_email, **:metadata** *=>* {**:customer\_key** *=>* **self**.id})  
 **self**.stripe\_customer\_id = *customer*.id  
 ***CacheProcessing***::***FillUserJob***.perform\_async(**self**.id)  
 *sub* = **self**.current\_subscription  
 *sub*.billing\_provider = **"stripe"** *sub*.save  
 **end  
  
 def** *setup\_initial\_subscription  
 plan* = ***Plan***.*where*(**:name** *=>* **"BRONZE"**).take  
  
 *billing\_provider* = **"stripe"  
  
 if self**.platform.eql?(**"shopify"**)  
 *billing\_provider* = **"shopify"  
 else** *customer* = ***Stripe***::***Customer***.*create*(**:email** *=>* **self**.contact\_email, **:metadata** *=>* {**:customer\_key** *=>* **self**.id})  
 **self**.stripe\_customer\_id = *customer*.id  
 **self**.save  
 ***CacheProcessing***::***FillUserJob***.perform\_in(5.seconds, **self**.id)  
 **end** *subscription* = ***Subscription***.*create* **:billing\_provider** *=> billing\_provider*, **:merchant\_id** *=>* **self**.id, **:plan\_id** *=> plan*.id, **:base\_price\_cents** *=> plan*.base\_price\_cents, **:per\_order\_price\_cents** *=> plan*.per\_order\_price\_cents, **:included\_orders** *=> plan*.included\_orders  
  
 **if self**.reload.install\_order\_volume >= 100  
 **self**.start\_free\_trial  
 **end** *subscription*.update\_third\_party  
  
 **end  
  
 def** *create\_custom\_shopify\_subscription\_by\_id*(*plan\_name*, *base\_price\_cents*=**nil**, *per\_order\_price\_cents*=**nil**, *included\_orders*=**nil**)  
 *processor* = **self**.processors.*where*(**:type** *=>* **"ShopifyProcessor"**).take  
 **return nil unless** *processor  
  
 plan* = ***Plan***.*where*(**:name** *=> plan\_name*).take  
 **return nil unless** *plan  
  
 plan*.base\_price\_cents = *base\_price\_cents*.to\_i **if** *base\_price\_cents*.present?  
 *plan*.per\_order\_price\_cents = *per\_order\_price\_cents*.to\_i **if** *per\_order\_price\_cents*.present?  
 *plan*.included\_orders = *included\_orders*.to\_i **if** *included\_orders*.present?  
  
 *recurring\_charge* = *processor*.create\_recurring\_charge(*plan*)  
 **return nil unless** *recurring\_charge  
  
 temp\_subscription* = ***Subscription***.new  
 *temp\_subscription*.merchant\_id = **self**.id  
 *temp\_subscription*.plan\_id = *plan*.id  
 *temp\_subscription*.base\_price\_cents = *plan*.base\_price\_cents  
 *temp\_subscription*.per\_order\_price\_cents = *plan*.per\_order\_price\_cents  
 *temp\_subscription*.included\_orders = *plan*.included\_orders  
 *temp\_subscription*.current = **false** *temp\_subscription*.*pending* = **true** *temp\_subscription*.billing\_provider = **"shopify"** *temp\_subscription*.recurring\_charge\_id = *recurring\_charge*.id  
 *temp\_subscription*.save  
  
 *recurring\_charge* **end  
  
 def** *create\_shopify\_subscription\_by\_id*(*plan\_name*)  
 *processor* = **self**.processors.*where*(**:type** *=>* **"ShopifyProcessor"**).take  
 **return nil unless** *processor  
  
 plan* = ***Plan***.*where*(**:name** *=> plan\_name*).take  
 **return nil unless** *plan  
  
 recurring\_charge* = *processor*.create\_recurring\_charge(*plan*)  
 **return nil unless** *recurring\_charge  
  
 temp\_subscription* = ***Subscription***.new  
 *temp\_subscription*.merchant\_id = **self**.id  
 *temp\_subscription*.plan\_id = *plan*.id  
 *temp\_subscription*.base\_price\_cents = *plan*.base\_price\_cents  
 *temp\_subscription*.per\_order\_price\_cents = *plan*.per\_order\_price\_cents  
 *temp\_subscription*.included\_orders = *plan*.included\_orders  
 *temp\_subscription*.current = **false** *temp\_subscription*.*pending* = **true** *temp\_subscription*.billing\_provider = **"shopify"** *temp\_subscription*.recurring\_charge\_id = *recurring\_charge*.id  
 *temp\_subscription*.save  
  
 *recurring\_charge* **end  
  
 def** *calculate\_and\_create\_monthly\_usage\_charge*(*start\_at* = **nil**)  
 **if** *start\_at*.nil?  
 *start\_at* = ***Time***.now - (30\*24).hours  
 **end** *orders* = **self**.purchases.*where*(**"created\_at >= ? and amount\_cents > 0"**, *start\_at*).count  
  
 *current\_subscription* = **self**.subscriptions.*where*(**:current** *=>* **true**).take  
  
 *current\_subscription*.last\_usage\_charge\_at = ***Time***.now  
 *current\_subscription*.save  
  
 *orders\_over* = *orders* - *current\_subscription*.included\_orders  
  
 **if** *current\_subscription*.billing\_provider.eql?(**"shopify"**)  
 **self**.swell\_plan\_paid(*current\_subscription*.base\_price\_cents)  
 **end  
  
 if** *orders\_over* > 0  
 monitor\_orders\_limit\_reach(*orders*)  
 *charge\_amount\_cents* = *orders\_over* \* *current\_subscription*.per\_order\_price\_cents  
 *description* = **"Swell Usage of** #{*orders\_over*} **orders at** #{*current\_subscription*.per\_order\_price\_cents} **cents each"  
 self**.create\_usage\_charge(*charge\_amount\_cents*, *description*, *current\_subscription*.recurring\_charge\_id)  
  
 ***UsageCharge***.*create* **:subscription\_id** *=> current\_subscription*.id, **:merchant\_id** *=>* **self**.id, **:per\_order\_price\_cents** *=> current\_subscription*.per\_order\_price\_cents, **:orders** *=> orders\_over*, **:total\_amount\_cents** *=> charge\_amount\_cents*, **:description** *=> description* **end  
 end  
  
 def** *calculate\_and\_create\_new\_subscription\_usage\_charge*(*old\_subscription*, *new\_subscription*)  
 *last\_usage\_charge\_at* = *old\_subscription*.last\_usage\_charge\_at  
  
 **unless** *last\_usage\_charge\_at*.present?  
 *last\_usage\_charge\_at* = *old\_subscription*.created\_at  
 **end** *old\_subscription*.last\_usage\_charge\_at = ***Time***.now  
 *old\_subscription*.save  
  
 *orders* = **self**.purchases.*where*(**"created\_at > ?"**, *last\_usage\_charge\_at*).count  
 *orders\_over* = *orders* - *old\_subscription*.included\_orders  
  
 **if** *orders\_over* > 0  
 monitor\_orders\_limit\_reach(*orders*)  
 *charge\_amount\_cents* = *orders\_over* \* *old\_subscription*.per\_order\_price\_cents  
 *description* = **"Swell Usage of** #{*orders\_over*} **orders at** #{*old\_subscription*.per\_order\_price\_cents} **cents each"  
 self**.create\_usage\_charge(*charge\_amount\_cents*, *description*, *new\_subscription*.recurring\_charge\_id)  
 ***UsageCharge***.*create* **:subscription\_id** *=> old\_subscription*.id, **:merchant\_id** *=>* **self**.id, **:per\_order\_price\_cents** *=> old\_subscription*.per\_order\_price\_cents, **:orders** *=> orders\_over*, **:total\_amount\_cents** *=> charge\_amount\_cents*, **:description** *=> description* **end  
 end  
  
 def** *create\_usage\_charge*(*charge\_amount\_cents*, *description*, *recurring\_charge\_id*=**nil**)  
 **if self**.current\_subscription.billing\_provider.eql?(**"shopify"**)  
 *processor* = **self**.processors.*where*(**:type** *=>* **"ShopifyProcessor"**).take  
  
 **if** *processor  
 capped\_amount\_cents* = **self**.current\_subscription.plan.capped\_amount\_cents  
  
 **if** *charge\_amount\_cents* > *capped\_amount\_cents  
 charge\_amount\_cents* = *capped\_amount\_cents* **end** *processor*.create\_usage\_charge(*recurring\_charge\_id*, *charge\_amount\_cents*, *description*)  
 **self**.swell\_plan\_paid(*charge\_amount\_cents*)  
 **end  
 elsif self**.current\_subscription.billing\_provider.eql?(**"stripe"**)  
 *# charge on stripe  
 invoice\_item* = ***Stripe***::***InvoiceItem***.*create*(  
 **:customer** *=>* **self**.stripe\_customer\_id,  
 **:amount** *=> charge\_amount\_cents*,  
 **:currency** *=>* **"usd"**,  
 **:description** *=> description* )  
 **end  
 end  
  
 def** *switch\_subscription*(*new\_subscription*)  
 *old\_mrr\_cents* = **self**.current\_mrr\_cents  
 *plan* = *new\_subscription*.plan  
 *current\_subscription* = **self**.current\_subscription  
 *old\_plan\_name* = *current\_subscription*.plan.name  
 *current\_subscription*.current = **false** *current\_subscription*.was\_current = **true** *current\_subscription*.save  
  
 *new\_subscription*.current = **true** *new\_subscription*.*pending* = **false** *new\_subscription*.save  
  
 *new\_subscription*.update\_third\_party  
  
 *plan\_names* = [**"BRONZE"**, **"SILVER"**, **"GOLD"**, **"PLATINUM"**, **"ENTERPRISE"**]  
  
 *old\_index* = *plan\_names*.index(*old\_plan\_name*)  
 *new\_index* = *plan\_names*.index(*plan*.name)  
  
 **if** *old\_plan\_name* == **"BRONZE"  
 if self**.program\_locked  
 **self**.program\_locked = **false  
 self**.save  
 **end  
 else  
  
 if** *new\_subscription*.billing\_provider.eql?(**"shopify"**)  
 *# add usage charge from old plan and cancel current usage charge job* **self**.calculate\_and\_create\_new\_subscription\_usage\_charge(*current\_subscription*, *new\_subscription*)  
 ***Workers***::***Control***.cancel\_by\_id(**self**.usage\_charge\_job\_id)  
 **end  
 end  
  
 if** *plan*.name.eql?(**"BRONZE"**) && !*old\_plan\_name*.eql?(**"BRONZE"**)  
 **if self**.thirty\_day\_order\_volume >= 100  
 **self**.program\_locked = **true  
 self**.save  
 **end  
 else  
 self**.on\_free\_trial = **false  
 self**.save  
 **end** *# automatically enabled syncing* ***when*** *upgrading from free plan* **if** *old\_plan\_name*.eql?(**"BRONZE"**) && !*plan*.name.eql?(**"BRONZE"**)  
 **unless self**.sync\_newsletter\_with\_platform  
 **self**.sync\_newsletter\_with\_platform = **true  
 self**.save  
 **end  
 end  
  
 if** *plan*.name.eql?(**"BRONZE"**) && *current\_subscription*.billing\_provider.eql?(**"shopify"**)  
 *current\_subscription*.cancel  
 **else  
 if** *new\_subscription*.billing\_provider.eql?(**"shopify"**)  
 **self**.usage\_charge\_job\_id = ***AccountProcessing***::***CreateUsageChargeJob***.perform\_at(***Time***.now + 30.days, **self**.id)  
 **self**.save  
 **end  
 end  
  
 if** *new\_subscription*.billing\_provider.eql?(**"shopify"**)  
 *net\_plan\_amount\_cents* = *new\_subscription*.base\_price\_cents - *current\_subscription*.base\_price\_cents  
 **self**.swell\_plan\_paid(*net\_plan\_amount\_cents*)  
 **end  
  
 self**.owner.remove\_intercom\_tag(*old\_plan\_name*)  
 **self**.owner.add\_intercom\_tag(*plan*.name)  
  
 **if** *old\_index* < *new\_index* ***AdminEmails***::***MerchantUpgradedJob***.perform\_async({ **:old\_mrr\_cents** *=> old\_mrr\_cents*, **:current\_mrr\_cents** *=>* **self**.current\_mrr\_cents, **:old\_plan\_name** *=> old\_plan\_name*, **:new\_plan\_name** *=> plan*.name, **:company\_name** *=>* **self**.company\_name, **:email** *=>* **self**.owner.email, **:alternate\_email** *=>* **self**.owner.alternate\_email, **:app** *=>* **"Rewards"**, **:platform** *=>* **self**.platform, **:thirty\_day\_order\_volume** *=>* **self**.thirty\_day\_order\_volume})  
 **self**.owner.log\_intercom\_event(**"upgraded"**, {**:plan\_id** *=> plan*.name, **:from\_plan\_id** *=> old\_plan\_name*})  
 **else  
 *AdminEmails***::***MerchantDowngradedJob***.perform\_async({ **:old\_mrr\_cents** *=> old\_mrr\_cents*, **:current\_mrr\_cents** *=>* **self**.current\_mrr\_cents, **:old\_plan\_name** *=> old\_plan\_name*, **:new\_plan\_name** *=> plan*.name, **:company\_name** *=>* **self**.company\_name, **:email** *=>* **self**.owner.email, **:alternate\_email** *=>* **self**.owner.alternate\_email, **:app** *=>* **"Rewards"**, **:platform** *=>* **self**.platform, **:thirty\_day\_order\_volume** *=>* **self**.thirty\_day\_order\_volume})  
 **self**.owner.log\_intercom\_event(**"downgraded"**, {**:plan\_id** *=> plan*.name, **:from\_plan\_id** *=> old\_plan\_name*})  
 **end** *new\_subscription* **end  
  
 def** *switch\_plan\_by\_id*(*plan\_id*)  
 *plan* = ***Plan***.*find*(*plan\_id*)  
 *current\_subscription* = **self**.current\_subscription  
 *new\_subscription* = ***Subscription***.new  
 *new\_subscription*.merchant\_id = **self**.id  
 *new\_subscription*.plan\_id = *plan*.id  
 *new\_subscription*.base\_price\_cents = *plan*.base\_price\_cents  
 *new\_subscription*.per\_order\_price\_cents = *plan*.per\_order\_price\_cents  
 *new\_subscription*.included\_orders = *plan*.included\_orders  
 *new\_subscription*.current = **false** *new\_subscription*.billing\_provider = *current\_subscription*.billing\_provider  
 *new\_subscription*.save  
  
 **self**.switch\_subscription(*new\_subscription*)  
 **end  
  
 def** *switch\_plan\_by\_name*(*plan\_name*)  
 *plan* = ***Plan***.*where*(**:name** *=> plan\_name*).take  
  
 **if** *plan* **self**.switch\_plan\_by\_id(*plan*.id)  
 **end  
 end  
  
 def** *start\_free\_trial*(*start\_time* = ***Time***.now, *trial\_length\_days* = 30)  
 **self**.on\_free\_trial = **true  
 self**.free\_trial\_expires\_at = *start\_time* + *trial\_length\_days*.days  
 **self**.free\_trial\_expiration\_job\_id = ***AccountProcessing***::***ExpireFreeTrialJob***.perform\_at(**self**.free\_trial\_expires\_at, **self**.id)  
 **self**.save  
 **end  
  
 def** *expire\_trial* **if self**.on\_free\_trial  
 **self**.on\_free\_trial = **false  
  
 if self**.thirty\_day\_order\_volume >= 100  
 **self**.program\_locked = **true  
  
 *AdminEmails***::***MerchantExpiredJob***.perform\_async({ **:created\_at** *=>* **self**.created\_at.to\_date, **:company\_name** *=>* **self**.company\_name, **:email** *=>* **self**.owner.email, **:alternate\_email** *=>* **self**.owner.alternate\_email, **:app** *=>* **"Rewards"**, **:platform** *=>* **self**.platform, **:thirty\_day\_order\_volume** *=>* **self**.thirty\_day\_order\_volume})  
 **end  
  
 self**.save  
 **end  
 end  
  
 def** *days\_left\_on\_free\_trial* **if self**.on\_free\_trial  
 *free\_trial\_days\_left* = [((**self**.free\_trial\_expires\_at - ***Time***.now) / 86400).to\_i, 0].max  
 **else** *free\_trial\_days\_left* = 0  
 **end** *free\_trial\_days\_left* **end  
  
 def self**.*process\_weekly\_reports* **return** *# disable* ***until*** *we allow merchants to disable this email  
 feed\_url* = **"http://blog.swellrewards.com/feed"** *feed* = ***Feedjira***::***Feed***.fetch\_and\_parse(*feed\_url*)  
 *most\_recent\_blog\_post* = *feed*.entries.first  
  
 ***Merchant***.*where*(**:has\_rewards\_installed** *=>* **true**, **:weekly\_digest\_enabled** *=>* **true**).find\_each **do** |*merchant*|  
 *merchant*.send\_digest\_email(most\_recent\_blog\_post)  
 **end  
 end  
  
 def** *lifetime\_roi  
 lcc* = **self**.lifetime\_cost\_cents  
 *roi* = 0  
  
 **if** *lcc* == 0  
 *roi* = -1  
 **else** *roi* = (**self**.lifetime\_value\_cents.to\_f / *lcc*.to\_f).round(2)  
 **end** *roi* **end  
  
 def** *lifetime\_revenue\_generated\_cents* **self**.dashboard\_snapshots.*sum*(**:total\_revenue\_cents**) + **self**.dashboard\_snapshot\_records.*sum*(**:total\_revenue\_cents**)  
 **end  
  
 def** *lifetime\_value\_cents* **self**.dashboard\_snapshots.*sum*(**:total\_value\_cents**) + **self**.dashboard\_snapshot\_records.*sum*(**:total\_value\_cents**)  
 **end  
  
 def** *lifetime\_cost\_cents* **self**.lifetime\_swell\_discount\_cost\_cents + **self**.lifetime\_swell\_plan\_cost\_cents  
 **end  
  
 def** *lifetime\_swell\_discount\_cost\_cents* **self**.dashboard\_snapshots.*sum*(**:swell\_discount\_cost\_cents**) + **self**.dashboard\_snapshot\_records.*sum*(**:swell\_discount\_cost\_cents**)  
 **end  
  
 def** *lifetime\_swell\_plan\_cost\_cents* **self**.dashboard\_snapshots.*sum*(**:swell\_plan\_cost\_cents**) + **self**.dashboard\_snapshot\_records.*sum*(**:swell\_plan\_cost\_cents**)  
 **end  
  
 def** *points\_earned*(*points* = 0, *date* = ***Date***.today)  
 **self**.dashboard\_snapshot\_records.*create*(**:day** *=> date*, **:points\_earned** *=> points*)  
 *# snapshot = self.dashboard\_snapshots.find\_or\_create\_by(****:day*** *=> date)  
 # snapshot.points\_earned += points  
 # snapshot.save* **end  
  
 def** *points\_redeemed*(*points* = 0, *date* = ***Date***.today)  
 **self**.dashboard\_snapshot\_records.*create*(**:day** *=> date*, **:points\_redeemed** *=> points*, **:redemptions** *=>* 1)  
 *# snapshot = self.dashboard\_snapshots.find\_or\_create\_by(****:day*** *=> date)  
 # snapshot.points\_redeemed += points  
 # snapshot.redemptions += 1  
 # snapshot.save* **end  
  
 def** *point\_redemption\_reversed*(*points*, *date* = ***Date***.today)  
 **self**.dashboard\_snapshot\_records.*create*(**:day** *=> date*, **:points\_redeemed** *=>* -1 \* *points*, **:redemptions** *=>* -1)  
 **end  
  
 def** *action\_taken*(*date* = ***Date***.today)  
 **self**.dashboard\_snapshot\_records.*create*(**:day** *=> date*, **:actions** *=>* 1)  
 *# snapshot = self.dashboard\_snapshots.find\_or\_create\_by(****:day*** *=> date)  
 # snapshot.actions += 1  
 # snapshot.save* **end  
  
 def** *newsletter\_signup*(*date* = ***Date***.today)  
 **self**.dashboard\_snapshot\_records.*create*(**:day** *=> date*, **:email\_subscribers** *=>* 1)  
 *# snapshot = self.dashboard\_snapshots.find\_or\_create\_by(****:day*** *=> date)  
 # snapshot.email\_subscribers += 1  
 # snapshot.save* **end  
  
 def** *email\_notification\_purchase\_made*(*purchase*)  
 **self**.dashboard\_snapshot\_records.*create*(**:day** *=> purchase*.created\_at.to\_date, **:email\_notification\_purchases\_cents** *=> purchase*.total\_amount\_cents, **:total\_revenue\_cents** *=> purchase*.total\_amount\_cents, **:swell\_discount\_cost\_cents** *=> purchase*.swell\_discount\_cost\_cents)  
 *# snapshot = self.dashboard\_snapshots.find\_or\_create\_by(****:day*** *=> purchase.created\_at.to\_date)  
 # snapshot.email\_notification\_purchases\_cents += purchase.total\_amount\_cents  
 # snapshot.purchase\_made(purchase)  
 # snapshot.save* **end  
  
 def** *loyalty\_purchase\_made*(*purchase*)  
 **self**.dashboard\_snapshot\_records.*create*(**:day** *=> purchase*.created\_at.to\_date, **:loyalty\_purchases** *=>* 1, **:loyalty\_purchases\_cents** *=> purchase*.total\_amount\_cents, **:total\_revenue\_cents** *=> purchase*.total\_amount\_cents, **:swell\_discount\_cost\_cents** *=> purchase*.swell\_discount\_cost\_cents)  
 *# snapshot = self.dashboard\_snapshots.find\_or\_create\_by(****:day*** *=> purchase.created\_at.to\_date)  
 # snapshot.loyalty\_purchases += 1  
 # snapshot.loyalty\_purchases\_cents += purchase.total\_amount\_cents  
 # snapshot.purchase\_made(purchase)  
 # snapshot.save* **end  
  
 def** *referral\_purchase\_made*(*purchase*)  
 **self**.dashboard\_snapshot\_records.*create*(**:day** *=> purchase*.created\_at.to\_date, **:referral\_revenue\_cents** *=> purchase*.total\_amount\_cents, **:total\_revenue\_cents** *=> purchase*.total\_amount\_cents, **:swell\_discount\_cost\_cents** *=> purchase*.swell\_discount\_cost\_cents)  
 *# snapshot = self.dashboard\_snapshots.find\_or\_create\_by(****:day*** *=> purchase.created\_at.to\_date)  
 # snapshot.referral\_revenue\_cents += purchase.total\_amount\_cents  
 # snapshot.purchase\_made(purchase)  
 # snapshot.save* **end  
  
 def** *swell\_plan\_paid*(*amount\_cents*, *date* = ***Date***.today)  
 **self**.dashboard\_snapshot\_records.*create*(**:day** *=> date*, **:swell\_plan\_cost\_cents** *=> amount\_cents*)  
 *# snapshot = self.dashboard\_snapshots.find\_or\_create\_by(****:day*** *=> date)  
 # snapshot.swell\_plan\_cost\_cents += amount\_cents  
 # snapshot.save* **end  
  
  
 def** *delete\_dashboard\_snapshots*(*start\_date*, *end\_date*)  
 **self**.dashboard\_snapshots.*where*(**:day** *=> start\_date*..*end\_date*).delete\_all  
 **end  
  
 def** *backfill\_dashboard\_snapshots*(*start\_date*, *end\_date*)  
 ***Perk***.select(**'id'**).*where*(**merchant\_id**: **self**.id).find\_in\_batches **do** |*perk\_ids\_batch*|  
 ***Perk***.select(**'id'**, **'reward\_points'**, **'completed\_at'**).*where*(**id**: *perk\_ids\_batch*, **completed**: **true**, **completed\_at**: *start\_date*..*end\_date*).find\_each **do** |*perk*|  
 **if** *perk*.reward\_points.present? && *perk*.reward\_points > 0  
 **self**.points\_earned(*perk*.reward\_points, *perk*.completed\_at.to\_date)  
 **end  
 end  
 end  
  
 self**.point\_redemptions.*where*(**:created\_at** *=> start\_date*..*end\_date*).*includes*(**:redemption\_option**).*includes*(**:redemption\_code**).find\_each **do** |*point\_redemption*|  
 **if** *point\_redemption*.redemption\_option.present?  
 **if** *point\_redemption*.redemption\_option.variable\_amount?  
 **if** *point\_redemption*.redemption\_code.nil? || *point\_redemption*.redemption\_code.uses == 0  
  
 **else  
 if** *point\_redemption*.amount.present?  
 **self**.points\_redeemed(*point\_redemption*.amount, *point\_redemption*.created\_at.to\_date)  
 **else  
 self**.points\_redeemed(0, *point\_redemption*.created\_at.to\_date)  
 **end  
 end  
 else  
 if** *point\_redemption*.redemption\_option.amount.present?  
 **self**.points\_redeemed(*point\_redemption*.redemption\_option.amount, *point\_redemption*.created\_at.to\_date)  
 **else  
 self**.points\_redeemed(0, *point\_redemption*.created\_at.to\_date)  
 **end  
 end  
 end  
 end  
  
 self**.customers.select(**'customers.id'**).find\_in\_batches(**batch\_size**: 10000) **do** |*customer\_ids*|  
 ***UserAction***.select(**'id'**, **'created\_at'**).*where*(**customer\_id**: *customer\_ids*).*where*(**:created\_at** *=> start\_date*..*end\_date*).find\_each **do** |*action*|  
 **self**.action\_taken(*action*.created\_at.to\_date)  
 **end  
 end  
  
 self**.purchases.select(**'purchases.id'**).*where*(**:created\_at** *=> start\_date*..*end\_date*).find\_in\_batches(**batch\_size**: 10000) **do** |*purchase\_ids*|  
 ***Purchase***.*where*(**id**: *purchase\_ids*).find\_each **do** |*purchase*|  
 **if** *purchase*.redemption\_code\_id.present? && *purchase*.referral\_id.nil?  
 **self**.loyalty\_purchase\_made(*purchase*)  
 **end  
  
 if** *purchase*.referral\_id.present?  
 **self**.referral\_purchase\_made(*purchase*)  
 **end  
 end  
 end  
  
 self**.email\_campaigns.*where*(**:type** *=>* **"CustomerPointsReminder"**).find\_each **do** |*ec*|  
 *ec*.email\_campaign\_emails.select(**'id'**, **'customer\_id'**, **'sent\_at'**).*where*(**:sent** *=>* **true**, **:sent\_at** *=> start\_date*..*end\_date*).find\_in\_batches **do** |*customer\_ids\_and\_sent\_at\_batch*|  
 *customer\_ids\_to\_sent\_at* = *customer\_ids\_and\_sent\_at\_batch*.index\_by(&**:customer\_id**)  
 ***Purchase***.*where*(**customer\_id**: *customer\_ids\_to\_sent\_at*.keys).find\_each **do** |*purchase*|  
 *sent\_at\_sql* = *customer\_ids\_to\_sent\_at*[*purchase*.customer\_id].sent\_at  
 *sent\_at\_date* = ***Date***.parse(*sent\_at\_sql*.to\_s)  
 **if** *purchase*.created\_at >= *sent\_at\_date* && *purchase*.created\_at <= *sent\_at\_date* + 7.days  
 **self**.email\_notification\_purchase\_made(*purchase*)  
 **end  
 end  
 end  
 end  
  
 self**.user\_actions.*where*(**:type** *=>* **"NewsletterSignup"**, **:created\_at** *=> start\_date*..*end\_date*).find\_each **do** |*signup*|  
 **self**.newsletter\_signup(*signup*.created\_at.to\_date)  
 **end  
  
 if self**.stripe\_customer\_id.present?  
 **begin  
 self**.backfill\_stripe\_invoices(*start\_date*, *end\_date*)  
 **rescue  
 begin** sleep 2  
 **self**.backfill\_stripe\_invoices(*start\_date*, *end\_date*)  
 **rescue  
 begin** sleep 2  
 **self**.backfill\_stripe\_invoices(*start\_date*, *end\_date*)  
 **rescue  
 end  
 end  
 end  
 end  
  
 end  
  
 def** *backfill\_stripe\_invoices*(*start\_date*, *end\_date*)  
 *invoices* = ***Stripe***::***Invoice***.*all*(**:customer** *=>* **self**.stripe\_customer\_id, **:limit** *=>* 100)  
 *invoices*.data.each **do** |*invoice*|  
 *invoice\_date* = ***DateTime***.strptime(**"**#{*invoice*.date}**"**,**'%s'**).to\_date  
 **if** *invoice\_date* >= *start\_date* && *invoice\_date* <= *end\_date* **self**.swell\_plan\_paid(*invoice*.total, *invoice\_date*)  
 **end  
 end  
 end  
   
 def** *ecommerce\_platform\_processor* **if self**.platform.eql?(**"shopify"**)  
 **self**.processors.*where*(**:type** *=>* **"ShopifyProcessor"**).take  
 **elsif self**.platform.eql?(**"magento"**)  
 **self**.processors.*where*(**:type** *=>* **"MagentoProcessor"**).take  
 **elsif self**.platform.eql?(**"magento2"**)  
 **self**.processors.*where*(**:type** *=>* **"Magento2Processor"**).take  
 **elsif self**.platform.eql?(**"weebly"**)  
 **self**.processors.*where*(**:type** *=>* **"WeeblyProcessor"**).take  
 **elsif self**.platform.eql?(**"ecwid"**)  
 **self**.processors.*where*(**:type** *=>* **"EcwidProcessor"**).take  
 **elsif self**.platform.eql?(**"bigcommerce"**)  
 **self**.processors.*where*(**:type** *=>* **"BigcommerceProcessor"**).take  
 **elsif self**.platform.eql?(**"woocommerce"**)  
 **self**.processors.*where*(**:type** *=>* **"WoocommerceProcessor"**).take  
 **elsif self**.platform.eql?(**"commerce\_cloud"**)  
 **self**.processors.*where*(**:type** *=>* **"CommerceCloudProcessor"**).take  
 **else  
 nil  
 end  
 end  
  
 def** *recharge\_processor* **self**.processors.*where*(**:type** *=>* **"RechargeProcessor"**).take  
 **end  
  
 def** *get\_srs* ***ShopifyRequestService***.new(**self**.ecommerce\_platform\_processor.oauth\_token)  
 **end  
  
 def** *get\_request\_service* **if** platform.eql?(**"shopify"**)  
 ***ShopifyRequestService***.new(**self**.ecommerce\_platform\_processor.oauth\_token)  
 **elsif** platform.eql?(**"bigcommerce"**)  
 ***BigcommerceRequestService***.new(**self**.ecommerce\_platform\_processor.oauth\_token)  
 **elsif** platform.eql?(**"woocommerce"**)  
 ***WoocommerceRequestService***.new(**self**.ecommerce\_platform\_processor.oauth\_token)  
 **end  
 end  
  
 def** *account\_status* **if** on\_free\_trial  
 **"trial"  
 elsif** program\_locked  
 **"locked"  
 else  
 "live"  
 end  
 end  
  
 def** *generate\_demo\_referral\_snapshots* **self**.referral\_snapshots.delete\_all  
 *start\_day* = ***Date***.today - 12.months  
 *current\_day* = *start\_day  
  
 bonus\_percent* = 1.02  
  
 **while** *current\_day* < ***Date***.today  
 *snapshot* = **self**.referral\_snapshots.find\_or\_create\_by(**:day** *=> current\_day*)  
  
 *snapshot*.shares = ((10 + rand(30)) \* *bonus\_percent*).to\_i  
 *snapshot*.facebook\_shares = (*snapshot*.shares \* 0.50).to\_i  
 *snapshot*.twitter\_shares = (*snapshot*.shares \* 0.30).to\_i  
 *snapshot*.email\_shares = (*snapshot*.shares \* 0.20).to\_i  
  
 *snapshot*.links\_clicked = ((12 + rand(20)) \* *bonus\_percent*).to\_i  
 *snapshot*.links\_clicked\_from\_facebook = (*snapshot*.links\_clicked \* 0.50).to\_i  
 *snapshot*.links\_clicked\_from\_twitter = (*snapshot*.links\_clicked \* 0.30).to\_i  
 *snapshot*.links\_clicked\_from\_email = (*snapshot*.links\_clicked \* 0.20).to\_i  
  
 *snapshot*.orders = (rand(2) \* *bonus\_percent*).to\_i  
 *snapshot*.amount\_cents = ((5000 + rand(7500)) \* *bonus\_percent*).to\_i \* *snapshot*.orders  
  
 **if** *snapshot*.orders > 0  
 *snapshot*.average\_amount\_cents = (*snapshot*.amount\_cents / *snapshot*.orders).to\_i  
 **else** *snapshot*.average\_amount\_cents = 0  
 **end** *snapshot*.save  
 *current\_day* += 1.day  
 *bonus\_percent* += 0.02  
 **end  
  
 self**.referral\_codes.each **do** |*referral\_code*|  
 *referral\_code*.shares = ((10 + rand(30)) \* *bonus\_percent*).to\_i  
 *referral\_code*.facebook\_shares = (*referral\_code*.shares \* 0.50).to\_i  
 *referral\_code*.twitter\_shares = (*referral\_code*.shares \* 0.30).to\_i  
 *referral\_code*.email\_shares = (*referral\_code*.shares \* 0.20).to\_i  
  
 *referral\_code*.links\_clicked = ((12 + rand(20)) \* *bonus\_percent*).to\_i  
 *referral\_code*.links\_clicked\_from\_facebook = (*referral\_code*.links\_clicked \* 0.50).to\_i  
 *referral\_code*.links\_clicked\_from\_twitter = (*referral\_code*.links\_clicked \* 0.30).to\_i  
 *referral\_code*.links\_clicked\_from\_email = (*referral\_code*.links\_clicked \* 0.20).to\_i  
  
 *referral\_code*.orders = (rand(2) \* *bonus\_percent*).to\_i  
 *referral\_code*.amount\_cents = ((5000 + rand(7500)) \* *bonus\_percent*).to\_i \* *referral\_code*.orders  
  
 **if** *referral\_code*.orders > 0  
 *referral\_code*.average\_amount\_cents = (*referral\_code*.amount\_cents / *referral\_code*.orders).to\_i  
 **else** *referral\_code*.average\_amount\_cents = 0  
 **end** *referral\_code*.save  
 **end  
 end  
  
 def** *get\_referral\_snapshots*(*start\_date*, *end\_date*)  
 **self**.referral\_snapshots.*where*(**'day between ? and ?'**, *start\_date*, *end\_date*).*order*(**'day ASC'**)  
 **end  
  
 def** *get\_affiliate\_snapshots*(*start\_date*, *end\_date*)  
 **self**.affiliate\_snapshots.*where*(**'day between ? and ?'**, *start\_date*, *end\_date*).*order*(**'day ASC'**)  
 **end  
  
 def** *get\_dashboard\_snapshots*(*start\_date*, *end\_date*)  
 **self**.dashboard\_snapshots.*where*(**'day between ? and ?'**, *start\_date*, *end\_date*).*order*(**'day ASC'**)  
 **end  
  
 def** *customers\_to\_csv\_soloshot  
 attributes* = **%w{email first\_name last\_name points\_earned points\_balance total\_purchases total\_amount\_spent\_cents referral\_link referral\_code referred\_by last\_seen referral\_code\_active referral\_code\_expired}  
 *CSV***.generate(**headers**: **true**) **do** |*csv*|  
 *csv* << *attributes* customers.*where*(**"store\_account\_id like '%-%'"**).find\_each **do** |*customer*|  
 *referred\_by* = *customer*.referred\_by\_customer  
 **if** *referred\_by*.nil?  
 *referred\_by* = **"Nobody"  
 else** *referred\_by* = *referred\_by*.email  
 **end** *referral\_code* = *customer*.active\_referral\_code\_object  
  
 *active* = **false** *expired* = **false  
  
 if** *referral\_code*.present?  
 *expired* = *referral\_code*.expired  
  
 **if** *referral\_code*.expires\_at.present? && !*expired  
 active* = **true  
 end  
 end** *csv* << [*customer*.email, *customer*.first\_name, *customer*.last\_name, *customer*.points\_earned, *customer*.points\_balance, *customer*.purchases.count, *customer*.purchases.*sum*(**:amount\_cents**), *customer*.active\_referral\_link, *customer*.active\_referral\_code, *referred\_by*, *customer*.calculated\_last\_seen\_at, *active*, *expired*]  
 **end  
 end  
 end  
  
 def** *referral\_host  
 rc* = **self**.campaigns.*where*(**:active** *=>* **true**, **:type** *=>***"ReferralCampaign"**).*order*(**"created\_at DESC"**).first  
 *rc* = **self**.campaigns.*where*(**:type** *=>***"ReferralCampaign"**).*order*(**"created\_at DESC"**).first **if** *rc*.nil?  
 *host* = **""** *host* = *rc*.referral\_host **if** *rc* **return** *host* **end  
  
 def** *customers\_to\_csv*(*start\_date\_str*, *end\_date\_str*, *filter*)  
 *ref\_host* = **self**.referral\_host  
  
 **if self**.discount\_code\_based\_referral\_program\_enabled  
 *attributes* = **%w{email first\_name last\_name points\_earned points\_balance referral\_link referral\_discount\_code referred\_by last\_seen vip\_tier birth\_month birth\_day birth\_year created\_at}  
 else** *attributes* = **%w{email first\_name last\_name points\_earned points\_balance referral\_link referred\_by last\_seen vip\_tier birth\_month birth\_day birth\_year created\_at}  
 end  
  
 *CSV***.generate(**headers**: **true**) **do** |*csv*|  
 *csv* << *attributes* **if** *filter*.eql?(**"all"**)  
 customers.select(**:id**).find\_in\_batches(**batch\_size**: 1000) **do** |*customers\_ids*|  
 customers.*where*(**id**: *customers\_ids*).  
 *where*(**'customers.created\_at >= ? and customers.created\_at <= ?'**, *start\_date\_str*.to\_datetime, *end\_date\_str*.to\_datetime.tomorrow).  
 select(**:id**, **:email**, **:first\_name**, **:name**, **:last\_name**, **:points\_earned**, **:points\_balance**, **:last\_seen\_at**, **:merchant\_id**, **:total\_spend\_cents**, **:referral\_code\_id**, **:referred\_by\_customer\_id**, **:vip\_tier\_id**, **:has\_provided\_tax\_details**, **:created\_at**, **:last\_purchase\_at**).  
 *includes*(**:vip\_tier**, **:referral\_code**, **:customer\_birthday**, **:referred\_by\_customer**).  
 find\_each **do** |*customer*|  
 *referred\_by* = *customer*.referred\_by\_customer  
 **if** *referred\_by*.nil?  
 *referred\_by* = **"Nobody"  
 else** *referred\_by* = *referred\_by*.email  
 **end** *tier\_name* = *customer*.vip\_tier  
  
 **if** *tier\_name*.present?  
 *tier\_name* = *tier\_name*.name  
 **else** *tier\_name* = **"None"  
 end** *birthday* = *customer*.customer\_birthday  
  
 **if** *birthday*.present?  
 *birth\_month* = *birthday*.month  
 *birth\_day* = *birthday*.day  
 *birth\_year* = *birthday*.year.present? ? *birthday*.year : **"Unknown"  
 else** *birth\_month* = **"Unknown"** *birth\_day* = **"Unknown"** *birth\_year* = **"Unknown"  
 end** *ref\_code* = *customer*.referral\_code  
  
 **if** *ref\_code*.present?  
 *ref\_code* = *ref\_code*.code  
 **else** *ref\_code* = **"None"  
 end** *ref\_link* = *ref\_host* + **"/"** + *ref\_code  
  
 ref\_discount\_code* = *customer*.referral\_discount\_code.present? ? *customer*.referral\_discount\_code.code : **""** *customer\_email* = *customer*.email  
 *customer\_email* = *customer\_email*.tr(**'"'**,**''**) **if** *customer\_email*.present?  
 **if self**.discount\_code\_based\_referral\_program\_enabled  
 *csv* << [*customer\_email*, *customer*.first\_name, *customer*.last\_name, *customer*.points\_earned, *customer*.points\_balance, *ref\_link*, *ref\_discount\_code*, *referred\_by*, *customer*.calculated\_last\_seen\_at, *tier\_name*, *birth\_month*, *birth\_day*, *birth\_year*, *customer*.created\_at]  
 **else** *csv* << [*customer\_email*, *customer*.first\_name, *customer*.last\_name, *customer*.points\_earned, *customer*.points\_balance, *ref\_link*, *referred\_by*, *customer*.calculated\_last\_seen\_at, *tier\_name*, *birth\_month*, *birth\_day*, *birth\_year*, *customer*.created\_at]  
 **end  
 end  
 end  
 elsif** *filter*.eql?(**"newsletter"**)  
 user\_actions.*includes*(**:customer**).*where*(**:type** *=>* **"NewsletterSignup"**).find\_each **do** |*user\_action*|  
 *customer* = *user\_action*.customer  
 **if** *customer*.present?  
 *referred\_by* = *customer*.referred\_by\_customer  
 **if** *referred\_by*.nil?  
 *referred\_by* = **"Nobody"  
 else** *referred\_by* = *referred\_by*.email  
 **end** *tier\_name* = *customer*.vip\_tier  
  
 **if** *tier\_name*.present?  
 *tier\_name* = *tier\_name*.name  
 **else** *tier\_name* = **"None"  
 end** *birthday* = *customer*.customer\_birthday  
  
 **if** *birthday*.present?  
 *birth\_month* = *birthday*.month  
 *birth\_day* = *birthday*.day  
 *birth\_year* = *birthday*.year.present? ? *birthday*.year : **"Unknown"  
 else** *birth\_month* = **"Unknown"** *birth\_day* = **"Unknown"** *birth\_year* = **"Unknown"  
 end** *ref\_code* = *customer*.referral\_code  
  
 **if** *ref\_code*.present?  
 *ref\_code* = *ref\_code*.code  
 **else** *ref\_code* = **"None"  
 end** *ref\_link* = *ref\_host* + **"/"** + *ref\_code  
  
 ref\_discount\_code* = *customer*.referral\_discount\_code.present? ? *customer*.referral\_discount\_code.code : **""  
  
 if self**.discount\_code\_based\_referral\_program\_enabled  
 *csv* << [*customer*.email, *customer*.first\_name, *customer*.last\_name, *customer*.points\_earned, *customer*.points\_balance, *ref\_link*, *ref\_discount\_code*, *referred\_by*, *customer*.calculated\_last\_seen\_at, *tier\_name*, *birth\_month*, *birth\_day*, *birth\_year*, *customer*.created\_at]  
 **else** *csv* << [*customer*.email, *customer*.first\_name, *customer*.last\_name, *customer*.points\_earned, *customer*.points\_balance, *ref\_link*, *referred\_by*, *customer*.calculated\_last\_seen\_at, *tier\_name*, *birth\_month*, *birth\_day*, *birth\_year*, *customer*.created\_at]  
 **end  
 end  
 end  
 end  
 end  
 end** add\_method\_tracer **:customers\_to\_csv**, **'Merchant::CustomersToCsv'  
  
 def** *payouts\_to\_csv*(*filter* = **"all"**)  
 *attributes* = **%w{payout\_email owner\_email total\_amount currency amount status approved\_at paypal\_transaction\_id paypal\_transaction\_status batch\_id batch\_status purchase\_id point\_redemption\_id customer\_transaction\_date sent\_at received\_at}  
 *CSV***.generate(**headers**: **true**) **do** |*csv*|  
 *csv* << *attributes  
 currency\_symbol* = ***Monetize***.parse(**self**.currency).symbol  
  
 *#lazy evaluation  
 cash\_payouts* = **self**.cash\_payouts  
 *cash\_payouts* = *cash\_payouts*.*where*(**:status** *=>* **"pending"**) **if** *filter*.eql?(**"pending"**)  
  
 **self**.cash\_payouts.find\_each **do** |*cash\_payout*|  
 *email* = *cash\_payout*.get\_payout\_email  
 *purchase* = *cash\_payout*.purchase  
 **if** *purchase*.present?  
 *\_email* = *purchase*.sender\_email  
 *\_amount* = *purchase*.affiliate\_amount  
 **else** *\_email* = *cash\_payout*.point\_redemption.customer.email  
 *\_amount* = *cash\_payout*.point\_redemption.redemption\_option.cash\_amount  
 **end** *csv* << [*email*, *\_email*, **"**#{*currency\_symbol*}#{*\_amount*}**"**, *cash\_payout*.currency, **"**#{*currency\_symbol*}#{*cash\_payout*.amount}**"**, *cash\_payout*.status, *cash\_payout*.sent\_at, *cash\_payout*.transaction\_id, *cash\_payout*.get\_transaction\_status, *cash\_payout*.third\_party\_batch\_id, *cash\_payout*.get\_batch\_status, *cash\_payout*.purchase\_id, *cash\_payout*.point\_redemption\_id, *cash\_payout*.created\_at, *cash\_payout*.sent\_at, *cash\_payout*.received\_at]  
 **end  
 end  
 end  
  
 def** *affiliates\_to\_csv*(*filter* = **"all"**)  
 *attributes* = **%w{email payout\_percentage has\_provided\_tax\_details is\_locked total\_amount\_earned created\_at updated\_at}  
 *CSV***.generate(**headers**: **true**) **do** |*csv*|  
 *csv* << *attributes* **if** *filter*.eql?(**"all"**)  
 **self**.affiliates.find\_each **do** |*affiliate*|  
 *csv* << [*affiliate*.email, *affiliate*.payout\_percentage, *affiliate*.customer.has\_provided\_tax\_details, *affiliate*.customer.is\_locked, *affiliate*.customer.get\_affiliate\_total\_earning, *affiliate*.created\_at, *affiliate*.updated\_at]  
 **end  
 elsif** *filter*.eql?(**"locked"**)  
 *locked\_customers* = **self**.customers.*where*(**:is\_locked** *=>* **true**).*pluck*(**:id**)  
 **if** *locked\_customers*.present?  
 **self**.affiliates.*where*(**customer\_id**: *locked\_customers*).find\_each **do** |*affiliate*|  
 *csv* << [*affiliate*.email, *affiliate*.payout\_percentage, *affiliate*.customer.has\_provided\_tax\_details, *affiliate*.customer.is\_locked, *affiliate*.customer.get\_affiliate\_total\_earning, *affiliate*.created\_at, *affiliate*.updated\_at]  
 **end  
 end  
 end  
 end  
 end  
  
 def** *redemptions\_to\_csv*(*start\_date\_str*, *end\_date\_str*)  
 *start\_time* = ***Time***.now  
 ***Rails***.logger.debug { { **message**: **'redemptions to csv has started'**, **start\_time**: *start\_time*, **merchant\_id**: **self**.id, **start\_date**: *start\_date\_str*, **end\_date**: *end\_date\_str*, **method\_name**: **'redemptions\_to\_csv'**} }  
 *headers* = **%w{date first\_name last\_name email description points discount\_id discount\_code times\_used amount\_used\_cents order\_id}** *headers* << **"recharge\_order" if self**.recharge\_api\_enabled?  
  
 *merchant\_redemption\_options\_by\_ids* = ***RedemptionOption***.select(**:id**, **:name**, **:amount**, **:discount\_type**).*where*(**merchant\_id**: **self**.id).index\_by(&**:id**)  
  
 *merchant\_reacharge\_api\_enabled* = **self**.recharge\_api\_enabled?  
 *csv\_num\_rows* = 0  
 *redemptions\_csv* = ***CSV***.generate(**headers**: **true**) **do** |*csv*|  
 *csv* << *headers* **self**.point\_redemptions  
 .select(**:id**, **:customer\_id**, **:redemption\_option\_id**, **:amount**, **:created\_at**)  
 .*where*(**visible**: **true**)  
 .*where*(**"created\_at >= ? and created\_at <= ?"**, *start\_date\_str*.to\_datetime, *end\_date\_str*.to\_datetime.tomorrow).find\_in\_batches(**batch\_size**: 10000) **do** |*point\_redemptions\_batch*|  
  
 *customers\_ids* = *point\_redemptions\_batch*.map(&**:customer\_id**)  
 *point\_redemptions\_customers\_by\_ids* = ***Customer***.select(**:id**, **:first\_name**, **:last\_name**, **:email**).*where*(**id**: *customers\_ids*).index\_by(&**:id**)  
  
 *redemption\_codes\_by\_point\_redemption\_id* = ***RedemptionCode***.select(**:id**, **:third\_party\_id**, **:uses**, **:amount\_used\_cents**, **:code**, **:point\_redemption\_id**)  
 .*where*(**point\_redemption\_id**: *point\_redemptions\_batch*)  
 .index\_by(&**:point\_redemption\_id**)  
  
 *purchases\_redemptions\_batch* = ***Purchase***.select(**'purchases\_redemptions.redemption\_id'**, **'order\_id'**, **'transaction\_hash'**)  
 .*joins*(**'INNER JOIN `purchases\_redemptions` ON `purchases\_redemptions`.`purchase\_id` = `purchases`.`id`'**)  
 .*where*(**'purchases\_redemptions.redemption\_id IN (?)'**, *point\_redemptions\_batch*)  
  
 *redemptions\_to\_purchases* = ***Hash***.new { |*h*, *k*| *h*[*k*] = { **order\_ids**: [], **transaction\_hashes**: [] } }  
 *purchases\_redemptions\_batch*.each **do** |*purchase\_redemption*|  
 *redemptions\_to\_purchases*[*purchase\_redemption*.redemption\_id][**:order\_ids**].push(*purchase\_redemption*.order\_id)  
 *redemptions\_to\_purchases*[*purchase\_redemption*.redemption\_id][**:transaction\_hashes**].push(*purchase\_redemption*.transaction\_hash)  
 **end** *point\_redemptions\_batch*.each **do** |*point\_redemption*|  
 *customer* = *point\_redemptions\_customers\_by\_ids*[*point\_redemption*.customer\_id]  
 **next unless** *customer*.present?  
  
 *redemption\_code\_id* = **"N/A"** *redemption\_code\_code* = **"N/A"** *times\_used* = **"N/A"** *amount\_used\_cents* = **"N/A"** *order\_id* = **"N/A"** *points* = 0  
 *description* = **"Point Redemption"** *redemption\_option* = *merchant\_redemption\_options\_by\_ids*[*point\_redemption*.redemption\_option\_id]  
 **if** *redemption\_option*.present?  
 *description* = *redemption\_option*.name  
  
 **if** *redemption\_option*.variable\_amount?  
 **if** *point\_redemption*.amount.present?  
 *points* = -1 \* *point\_redemption*.amount  
 **end  
 elsif** *redemption\_option*.amount.present?  
 *points* = -1 \* *redemption\_option*.amount  
 **end  
 end** *redemption\_code* = *redemption\_codes\_by\_point\_redemption\_id*[*point\_redemption*.id]  
 **if** *redemption\_code*.present?  
 *redemption\_code\_id* = *redemption\_code*.third\_party\_id  
 *redemption\_code\_code* = *redemption\_code*.code  
 *times\_used* = *redemption\_code*.uses  
 *amount\_used\_cents* = *redemption\_code*.amount\_used\_cents  
  
 *order\_id* = *redemptions\_to\_purchases*[*point\_redemption*.id][**:order\_ids**].first  
  
 **if** *merchant\_reacharge\_api\_enabled  
 order\_tags* = *redemptions\_to\_purchases*[*point\_redemption*.id][**:transaction\_hashes**].first  
 **if** *order\_tags* && *order\_tags*.split(**","**).map(&**:downcase**).index(**"subscription"**).present?  
 *recharge\_order* = **"yes"  
 else** *recharge\_order* = **"no"  
 end  
 end  
 end** *row* = [ *point\_redemption*.created\_at.to\_date.to\_s, *customer*.first\_name, *customer*.last\_name, *customer*.email, *description*, *points*, *redemption\_code\_id*, *redemption\_code\_code*, *times\_used*, *amount\_used\_cents*, *order\_id* ]  
 *row* << *recharge\_order* **if** *merchant\_reacharge\_api\_enabled  
  
 csv* << *row  
 csv\_num\_rows* += 1  
 **end  
 end  
 end  
 *Rails***.logger.debug ( { **message**: **'redemptions to csv has finished'**, **elapsed\_time\_in\_seconds**: ***Time***.now - *start\_time*, **csv\_number\_of\_rows**: *csv\_num\_rows*, **method\_name**: **'redemptions\_to\_csv'**})  
 *redemptions\_csv* **end  
  
 def** *perks\_to\_csv*(*start\_date\_str*, *end\_date\_str*)  
 *start\_time* = ***Time***.now  
 ***Rails***.logger.debug { { **message**: **'perks to csv has started'**, **start\_time**: *start\_time*, **merchant\_id**: **self**.id, **start\_date**: *start\_date\_str*, **end\_date**: *end\_date\_str*, **method\_name**: **'perks\_to\_csv'**} }  
  
 *merchant\_campaigns* = ***Campaign***.select(**:id**, **:title**, **:hidden**, **:type**).*where*(**merchant\_id**: **self**.id)  
 *merchant\_campaigns\_by\_ids* = *merchant\_campaigns*.index\_by(&**:id**)  
  
 *merchant\_redemption\_options* = ***RedemptionOption***.select(**:id**, **:name**).*where*(**merchant\_id**: **self**.id)  
 *merchant\_redemption\_options\_by\_ids* = *merchant\_redemption\_options*.index\_by(&**:id**)  
  
 *headers* = **%w{date first\_name last\_name email description points orders}** *csv\_num\_rows* = 0  
 *perks\_csv* = ***CSV***.generate(**headers**: **true**) **do** |*csv*|  
 *csv* << *headers* **self**.perks  
 .select(**:id**, **:history\_title**, **:created\_at**, **:reward\_points**, **:customer\_id**, **:campaign\_id**, **:redemption\_option\_id**)  
 .*where*(**completed**: **true**).*where*(**'perks.created\_at >= ? and perks.created\_at <= ?'**, *start\_date\_str*.to\_datetime, *end\_date\_str*.to\_datetime.tomorrow)  
 .find\_in\_batches(**batch\_size**: 100000) **do** |*perk\_batch*|  
 *customers\_ids* = *perk\_batch*.map(&**:customer\_id**)  
 *perks\_customers\_by\_ids* = ***Customer***.select(**:id**, **:first\_name**, **:last\_name**, **:email**).*where*(**id**: *customers\_ids*).index\_by(&**:id**)  
  
 *perks\_purchases\_start\_time* = ***Time***.now  
 *perks\_purchases\_batch* = ***Purchase***.select(**'perks\_purchases.perk\_id'**, **'order\_id'**)  
 .*joins*(**'INNER JOIN `perks\_purchases` ON `perks\_purchases`.`purchase\_id` = `purchases`.`id`'**)  
 .*where*(**'perks\_purchases.perk\_id IN (?)'**, *perk\_batch*)  
  
 *perks\_to\_order\_ids* = ***Hash***.new { |*h*, *k*| *h*[*k*] = [] }  
 *perks\_purchases\_batch*.each **do** |*perk\_purchase*|  
 *perks\_to\_order\_ids*[*perk\_purchase*.perk\_id].push(*perk\_purchase*.order\_id)  
 **end  
 *Rails***.logger.debug { { **message**: **'fetched perks purchases batch'**, **elapsed\_time\_in\_seconds**: ***Time***.now - *perks\_purchases\_start\_time*, **method\_name**: **'perks\_to\_csv'**} }  
  
 *perks\_user\_actions\_start\_time* = ***Time***.now  
 *perks\_user\_actions\_batch* = ***UserAction***.select(**'perks\_user\_actions.perk\_id'**, **'type'**)  
 .*joins*(**'INNER JOIN `perks\_user\_actions` ON `perks\_user\_actions`.`user\_action\_id` = `user\_actions`.`id`'**)  
 .*where*(**'perks\_user\_actions.perk\_id IN (?)'**, *perk\_batch*)  
  
 *perks\_to\_user\_action\_type* = ***Hash***.new  
 *perks\_user\_actions\_batch*.each **do** |*perk\_user\_action*|  
 *perks\_to\_user\_action\_type*[*perk\_user\_action*.perk\_id] ||= *perk\_user\_action*.type  
 **end  
 *Rails***.logger.debug { { **message**: **'fetched perks user actions batch'**, **elapsed\_time\_in\_seconds**: ***Time***.now - *perks\_user\_actions\_start\_time*, **method\_name**: **'perks\_to\_csv'**} }  
  
 *perk\_batch*.each **do** |*perk*|  
 *campaign* = *merchant\_campaigns\_by\_ids*[*perk*.campaign\_id]  
 *redemption\_option* = *merchant\_redemption\_options\_by\_ids*[*perk*.redemption\_option\_id]  
 *customer* = *perks\_customers\_by\_ids*[*perk*.customer\_id]  
 *orders* = *perks\_to\_order\_ids*[*perk*.id]  
 *user\_action\_type* = *perks\_to\_user\_action\_type*[*perk*.id]  
  
 **next if** *customer*.nil? || *campaign*.nil? || *campaign*.hidden  
  
 *description* = *perk*.history\_title || *campaign*.title  
  
 **if** *campaign*.type.eql?(**"ReferralCampaign"**) && *user\_action\_type*.present?  
 **if** *user\_action\_type*.eql?(**"ReferredUserCompletion"**)  
 *description* = **"Referred by a Friend"  
 elsif** *user\_action\_type*.eql?(**"ReferrerCompletion"**)  
 *description* = **"Referred a Friend"  
 end  
 end** *points* = 0  
 **if** *perk*.reward\_points.present?  
 *points* = *perk*.reward\_points  
 **elsif** *redemption\_option*.present?  
 *description* += **" - "** + *redemption\_option*.name  
 **end** *order\_ids* = *orders*.present? ? *orders*.join(**","**) : **"N/A"** *csv* << [*perk*.created\_at.to\_date.to\_s, *customer*.first\_name, *customer*.last\_name, *customer*.email, *description*, *points*, *order\_ids*]  
 *csv\_num\_rows* += 1  
 **end  
 end  
 end  
 *Rails***.logger.debug ( { **message**: **'perks to csv has finished'**, **elapsed\_time\_in\_seconds**: ***Time***.now - *start\_time*, **csv\_number\_of\_rows**: *csv\_num\_rows*, **method\_name**: **'perks\_to\_csv'**})  
 *perks\_csv* **end** add\_method\_tracer **:perks\_to\_csv**, **'Merchant::PerksToCsv'  
  
 def** *referral\_codes\_to\_csv*(*start\_date\_str*, *end\_date\_str*)  
 *start\_time* = ***Time***.now  
 ***Rails***.logger.debug { { **message**: **'referral codes to csv has started'**, **merchant\_id**: **self**.id, **method\_name**: **'referral\_codes\_to\_csv'**} }  
  
 *headers* = **%w{first\_name last\_name email code clicks links\_clicked shares facebook\_shares twitter\_shares email\_shares facebook\_messenger\_shares sms\_shares orders amount\_cents}** *csv\_num\_rows* = 0  
 *referral\_codes\_csv* = ***CSV***.generate(**headers**: **true**) **do** |*csv*|  
 *csv* << *headers* **self**.referral\_codes.select(**:id**, **:customer\_id**, **:code**, **:clicks**, **:links\_clicked**, **:shares**, **:facebook\_shares**, **:twitter\_shares**, **:facebook\_messenger\_shares**, **:sms\_shares**, **:orders**, **:amount\_cents**)  
 .*where*(**"referral\_codes.clicks > 0 or referral\_codes.links\_clicked > 0 or referral\_codes.shares > 0 or referral\_codes.orders > 0"**)  
 .find\_in\_batches(**batch\_size**: 100000) **do** |*referral\_code\_batch*|  
  
 *customers\_ids* = *referral\_code\_batch*.map(&**:customer\_id**)  
 *referral\_codes\_customers\_by\_ids* = ***Customer***.select(**:id**, **:first\_name**, **:last\_name**, **:email**).*where*(**id**: *customers\_ids*).index\_by(&**:id**)  
  
 *referral\_code\_to\_referral\_receipt\_sum* = ***Hash***.new { |*h*, *k*| *h*[*k*] = 0 }  
 *referral\_receipt\_batch* = ***ReferralReceipt***.select(**:id**, **:referral\_code\_id**).*where*(**referral\_code\_id**: *referral\_code\_batch*, **merchant\_id**: **self**.id)  
 *referral\_receipt\_batch*.each **do** |*referral\_receipt*|  
 *referral\_code\_to\_referral\_receipt\_sum*[*referral\_receipt*.referral\_code\_id] += 1  
 **end** *referral\_code\_batch*.each **do** |*referral\_code*|  
 *customer* = *referral\_codes\_customers\_by\_ids*[*referral\_code*.customer\_id]  
 *referral\_receipts\_count* = *referral\_code\_to\_referral\_receipt\_sum*[*referral\_code*.id]  
 *csv* << [  
 *customer*.first\_name,  
 *customer*.last\_name,  
 *customer*.email,  
 *referral\_code*.code,  
 *referral\_code*.clicks,  
 *referral\_code*.links\_clicked,  
 *referral\_code*.shares,  
 *referral\_code*.facebook\_shares,  
 *referral\_code*.twitter\_shares,  
 *referral\_receipts\_count*,  
 *referral\_code*.facebook\_messenger\_shares,  
 *referral\_code*.sms\_shares,  
 *referral\_code*.orders,  
 *referral\_code*.amount\_cents  
 ]  
 *csv\_num\_rows* += 1  
 **end  
 end  
 end  
 *Rails***.logger.debug ( { **message**: **'referral codes to csv has finished'**, **elapsed\_time\_in\_seconds**: ***Time***.now - *start\_time*, **csv\_number\_of\_rows**: *csv\_num\_rows*, **method\_name**: **'referral\_codes\_to\_csv'**})  
 *referral\_codes\_csv* **end  
  
 def** *referral\_receipts\_to\_csv*(*start\_date\_str*, *end\_date\_str*)  
 *start\_time* = ***Time***.now  
 ***Rails***.logger.debug { { **message**: **'referral receipts to csv has started'**, **merchant\_id**: **self**.id, **method\_name**: **'referral\_receipts\_to\_csv'**} }  
  
 *headers* = **%w{date first\_name last\_name referrer\_email referred\_user\_email signed\_up\_at purchased\_at completed\_at link\_share\_type}** *csv\_num\_rows* = 0  
 *referral\_receipts\_csv* = ***CSV***.generate(**headers**: **true**) **do** |*csv*|  
 *csv* << *headers* **self**.referral\_receipts.select(**:id**, **:created\_at**, **:signed\_up\_at**, **:purchased\_at**, **:completed\_at**, **:email**, **:customer\_id**, **:referral\_id**).find\_in\_batches(**batch\_size**: 100000) **do** |*referral\_receipt\_batch*|  
  
 *customers\_ids* = *referral\_receipt\_batch*.map(&**:customer\_id**)  
 *referral\_receipt\_customers\_by\_ids* = ***Customer***.select(**:id**, **:first\_name**, **:last\_name**, **:email**).*where*(**id**: *customers\_ids*).index\_by(&**:id**)  
  
 *referral\_receipt\_batch*.each **do** |*referral\_receipt*|  
 *customer* = *referral\_receipt\_customers\_by\_ids*[*referral\_receipt*.customer\_id]  
 *customer\_referral\_link* = *referral\_receipt*.referral ? *referral\_receipt*.referral.link\_share\_type : **nil  
 next unless** *customer*.present?  
 *created\_at\_date* = *referral\_receipt*.created\_at ? *referral\_receipt*.created\_at.to\_date.to\_s : **"N/A"** *signed\_up\_at\_date* = *referral\_receipt*.signed\_up\_at ? *referral\_receipt*.signed\_up\_at.to\_date.to\_s : **"N/A"** *purchased\_at\_date* = *referral\_receipt*.purchased\_at ? *referral\_receipt*.purchased\_at.to\_date.to\_s : **"N/A"** *completed\_at\_date* = *referral\_receipt*.completed\_at ? *referral\_receipt*.completed\_at.to\_date.to\_s : **"N/A"** *link\_share\_type* = *customer\_referral\_link* ? *customer\_referral\_link* : **"N/A"** *csv* << [*created\_at\_date*, *customer*.first\_name, *customer*.last\_name, *customer*.email, *referral\_receipt*.email, *signed\_up\_at\_date*, *purchased\_at\_date*, *completed\_at\_date*, *link\_share\_type*]  
 *csv\_num\_rows* += 1  
 **end  
 end  
 end  
 *Rails***.logger.debug ( { **message**: **'referral receipts to csv has finished'**, **elapsed\_time\_in\_seconds**: ***Time***.now - *start\_time*, **csv\_number\_of\_rows**: *csv\_num\_rows*, **method\_name**: **'referral\_receipts\_to\_csv'**})  
 *referral\_receipts\_csv* **end  
  
 def** *unsubscribers\_to\_csv*(*start\_date\_str*, *end\_date\_str*)  
 *start\_time* = ***Time***.now  
 ***Rails***.logger.debug { { **message**: **'unsubscribers to csv has started'**, **merchant\_id**: **self**.id, **method\_name**: **'unsubscribers\_to\_csv'**} }  
 *csv\_num\_rows* = 0  
 *unsubscribers\_csv* = ***CSV***.generate(**headers**: **false**) **do** |*csv*|  
 **self**.customers.select(**:id**).find\_in\_batches(**batch\_size**: 100000) **do** |*customer\_batch*|  
 *customers* = ***Customer***.select(**:email**).*where*(**id**: *customer\_batch*).*where*(**subscribed**: **false**)  
 *customers*.each **do** |*customer*|  
 *csv* << [*customer*.email]  
 *csv\_num\_rows* += 1  
 **end  
 end  
 end  
 *Rails***.logger.debug ( { **message**: **'unsubscribers to csv has finished'**, **elapsed\_time\_in\_seconds**: ***Time***.now - *start\_time*, **csv\_number\_of\_rows**: *csv\_num\_rows*, **method\_name**: **'unsubscribers\_to\_csv'**})  
 *unsubscribers\_csv* **end  
  
 def** *customer\_history\_to\_csv  
 headers* = **%w{date first\_name last\_name email description points discount\_id discount\_code times\_used amount\_used\_cents}  
 *CSV***.generate(**headers**: **true**) **do** |*csv*|  
 *csv* << *headers* **self**.perks.*includes*(**:customer**, **:campaign**, **:redemption\_option**).*where*(**:completed** *=>* **true**).find\_each **do** |*perk*|  
 *customer* = *perk*.customer  
 **if** *customer*.present? && !*perk*.campaign.hidden  
  
 *description* = *perk*.history\_title  
  
 **if** *description*.nil? && *perk*.campaign.present?  
 *description* = *perk*.campaign.title  
 **end  
  
 if** *perk*.campaign.present? && *perk*.campaign.type.eql?(**"ReferralCampaign"**)  
 *ua* = *perk*.user\_actions.first  
  
 **if** *ua*.present?  
 **if** *ua*.type.eql?(**"ReferredUserCompletion"**)  
 *description* = **"Referred by a Friend"  
 elsif** *ua*.type.eql?(**"ReferrerCompletion"**)  
 *description* = **"Referred a Friend"  
 end  
 end  
 end** *points* = 0  
  
 **if** *perk*.reward\_points.present?  
 *points* = *perk*.reward\_points  
 **elsif** *perk*.redemption\_option.present?  
 *description* += **" - "** + *perk*.redemption\_option.name  
 **end** *csv* << [*perk*.created\_at.to\_date.to\_s, *customer*.first\_name, *customer*.last\_name, *customer*.email, *description*, *points*, **"N/A"**, **"N/A"**, **"N/A"**, **"N/A"**]  
 **end  
 end  
  
 self**.point\_redemptions.*includes*(**:customer**, **:redemption\_option**, **:redemption\_code**).find\_each **do** |*point\_redemption*|  
 *customer* = *point\_redemption*.customer  
 **if** *customer*.present?  
 *redemption\_code\_id* = **"N/A"** *redemption\_code\_code* = **"N/A"** *times\_used* = **"N/A"** *amount\_used\_cents* = **"N/A"  
  
 if** *point\_redemption*.redemption\_option.present?  
 *description* = *point\_redemption*.redemption\_option.name  
 *points* = -1 \* *point\_redemption*.redemption\_option.amount  
  
 **if** *point\_redemption*.redemption\_code.present?  
 *redemption\_code\_id* = *point\_redemption*.redemption\_code.third\_party\_id  
 *redemption\_code\_code* = *point\_redemption*.redemption\_code.code  
 *times\_used* = *point\_redemption*.redemption\_code.uses  
 *amount\_used\_cents* = *point\_redemption*.redemption\_code.amount\_used\_cents  
 **end  
 else** *description* = **"Point Redemption"** *points* = 0  
 **end  
  
 if** *points*.nil?  
 *points* = 0  
 **end** *csv* << [*point\_redemption*.created\_at.to\_date.to\_s, *customer*.first\_name, *customer*.last\_name, *customer*.email, *description*, *points*, *redemption\_code\_id*, *redemption\_code\_code*, *times\_used*, *amount\_used\_cents*]  
 **end  
 end  
 end  
 end  
  
 def** *localization\_as\_json* ***Rails***.cache.fetch(**"**#{**self**.id}**/loader/localizations"**) **do** *rewards\_popup\_configuration* = **self**.campaign\_popup\_configurations.*where*(**type**: **"RewardsPopup"**, **active**: **true**).take  
  
 **if** *rewards\_popup\_configuration*.present?  
 ***RablRails***.*render*(*rewards\_popup\_configuration*.localization, **"api/v1/localizations/show"**)  
 **else  
 "{}"  
 end  
 end  
 end  
  
 def** *active\_vip\_tiers\_as\_json* ***Rails***.cache.fetch(**"**#{**self**.id}**/loader/vip\_tiers"**) **do  
 *RablRails***::***Renderer***::***Context***.*send*(**:include**, ***MoneyRails***::***ActionViewExtension***)  
 ***RablRails***.*render*(**self**.vip\_tiers, **"api/v1/vip\_tiers/index"**)  
 **end  
 end  
  
 def** *active\_popups\_as\_json* ***Rails***.cache.fetch(**"**#{**self**.id}**/loader/active\_popups"**) **do** *popups* = campaign\_popup\_configurations.*where*(**:active** *=>* **true**).*includes*(**:campaign\_popup\_items**).all  
 ***RablRails***.*render*(*popups*, **"api/v1/campaign\_popup\_configurations/index"**)  
 **end  
 end  
  
 def** *newsletter\_signup\_campaign* ***Rails***.cache.fetch(**"**#{**self**.id}**/loader/newsletter\_signup\_campaign"**) **do** *nsc* = **self**.campaigns.*where*(**:type** *=>* **"NewsletterSignupCampaign"**).*order*(**"created\_at DESC"**).first  
  
 **if** *nsc*.present?  
 ***RablRails***::***Renderer***::***Context***.*send*(**:include**, ***MoneyRails***::***ActionViewExtension***)  
 ***RablRails***.*render*(*nsc*, **"api/v1/campaigns/nochildren"**)  
 **else  
 "null"  
 end  
 end  
 end  
  
 def** *active\_campaigns\_as\_json* ***Rails***.cache.fetch(**"**#{**self**.id}**/loader/active\_campaigns"**) **do  
 *RablRails***::***Renderer***::***Context***.*send*(**:include**, ***MoneyRails***::***ActionViewExtension***)  
 ***RablRails***.*render*(**self**.consumer\_campaigns, **"api/v1/campaigns/nopaging"**)  
 **end  
 end  
  
 def** *active\_redemption\_options\_as\_json* ***Rails***.cache.fetch(**"**#{**self**.id}**/loader/redemption\_options"**) **do  
 *RablRails***::***Renderer***::***Context***.*send*(**:include**, ***MoneyRails***::***ActionViewExtension***)  
 *ros* = **self**.active\_redemption\_options.all  
 ***RablRails***.*render*(*ros*, **"api/v1/redemption\_options/nopaging"**)  
 **end  
 end  
  
 def** *active\_pos\_redemption\_options\_as\_json* ***Rails***.cache.fetch(**"**#{**self**.id}**/loader/pos\_redemption\_options"**) **do  
 *RablRails***::***Renderer***::***Context***.*send*(**:include**, ***MoneyRails***::***ActionViewExtension***)  
 *ros* = **self**.redemption\_options.*where*(**:deleted** *=>* **false**, **:display\_in\_popup** *=>* **true**, **:is\_pos** *=>* **true**).*order*(**"display\_order ASC"**).all  
 ***RablRails***.*render*(*ros*, **"api/v1/redemption\_options/nopaging"**)  
 **end  
 end  
  
 def** *active\_variable\_redemption\_options\_as\_json* ***Rails***.cache.fetch(**"**#{**self**.id}**/loader/variable\_redemption\_options"**) **do  
 *RablRails***::***Renderer***::***Context***.*send*(**:include**, ***MoneyRails***::***ActionViewExtension***)  
 *ros* = **self**.redemption\_options.*where*(**"deleted = false and (discount\_type = 'variable' or discount\_type = 'store\_credit\_variable' or discount\_type = 'gift\_certificate\_variable')"**).*order*(**"amount ASC"**).all  
 ***RablRails***.*render*(*ros*, **"api/v1/redemption\_options/nopaging"**)  
 **end  
 end  
  
 def** *active\_non\_variable\_redemption\_options\_as\_json* ***Rails***.cache.fetch(**"**#{**self**.id}**/loader/non\_variable\_redemption\_options"**) **do  
 *RablRails***::***Renderer***::***Context***.*send*(**:include**, ***MoneyRails***::***ActionViewExtension***)  
 *ros* = **self**.active\_redemption\_options.where.not(**discount\_type**: [**"variable"**, **"store\_credit\_variable"**, **"gift\_certificate\_variable"**]).all  
 ***RablRails***.*render*(*ros*, **"api/v1/redemption\_options/nopaging"**)  
 **end  
 end  
  
 def** *should\_process\_customer*(*tags* = **""**)  
 **if** customer\_tags && customer\_tags.length > 0  
 *customerTagsArray* = customer\_tags.split(**","**).map(&**:strip**).map(&**:downcase**)  
  
 **if** *tags*.present?  
 *tagsArray* = *tags*.split(**","**).map(&**:strip**).map(&**:downcase**)  
 **else** *tagsArray* = []  
 **end** *hasTag* = **false** *tagsArray*.each **do** |*customerTag*|  
 **if** *customerTagsArray*.index(*customerTag*).present?  
 *hasTag* = **true  
 end  
 end  
  
 if** exclude\_customer\_tags  
 **if** *hasTag* **return false  
 else  
 return true  
 end  
 else  
 if** *hasTag* **return true  
 else  
 return false  
 end  
 end  
 else  
 return true  
 end  
 end  
  
 def** *reset\_data* customers.destroy\_all  
 purchases.destroy\_all  
 user\_actions.destroy\_all  
 perks.destroy\_all  
 processors.each **do** |*processor*|  
 *processor*.processor\_snapshots.destroy\_all  
 **end  
 end  
  
 def** *reset\_data\_sql* ***ActiveRecord***::***Base***.connection.execute(**"delete from perks\_purchases where purchase\_id in (select id from purchases where merchant\_id=**#{**self**.id}**);"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from perks\_purchases where perk\_id in (select id from perks where merchant\_id=**#{**self**.id}**);"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from perks\_user\_actions where perk\_id in (select id from perks where merchant\_id=**#{**self**.id}**);"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from perks\_user\_actions where user\_action\_id in (select id from user\_actions where merchant\_id=**#{**self**.id}**);"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from processor\_snapshots where processor\_id in (select id from processors where merchant\_id=**#{**self**.id}**);"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from purchase\_items where purchase\_id in (select id from purchases where merchant\_id=**#{**self**.id}**);"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from refunds where purchase\_id in (select id from purchases where merchant\_id=**#{**self**.id}**);"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from customers where merchant\_id=**#{**self**.id}**;"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from purchases where merchant\_id=**#{**self**.id}**;"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from perks where merchant\_id=**#{**self**.id}**;"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from user\_actions where merchant\_id=**#{**self**.id}**;"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from referral\_codes where campaign\_id in (select id from campaigns where merchant\_id=**#{**self**.id}**);"**)  
 ***ActiveRecord***::***Base***.connection.execute(**"delete from referrals where merchant\_id=**#{**self**.id}**;"**)  
 **end  
  
 def** *reset\_customer\_data\_sql* ***AccountProcessing***::***ResetData***::***ResetMerchantJob***.perform\_async(**self**.id)  
 **end  
  
 def** *re\_sync\_event\_consumers* **self**.event\_consumers.*where*(**:active** *=>* **true**).each **do** |*ec*|  
 *ec*.resync\_customers  
 **end  
 end  
  
 def** *active\_event\_consumers* **self**.event\_consumers.*where*(**:active** *=>* **true**)  
 **end  
  
 def** *active\_redemption\_options* **self**.redemption\_options.*where*(**:deleted** *=>* **false**, **:display\_in\_popup** *=>* **true**, **:is\_pos** *=>* **false**).*order*(**"display\_order ASC"**)  
 **end  
  
 def** *shopify\_pos\_redemption\_options* **self**.redemption\_options.*where*(**:deleted** *=>* **false**, **:display\_in\_popup** *=>* **true**, **:discount\_type** *=>* [**"fixed\_amount"**, **"percentage"**]).*order*(**"display\_order ASC"**)  
 **end  
  
 def** *setup\_default\_rewards\_program* **self**.ensure\_default\_coupons  
  
 *create\_account\_campaign* = ***CreateAccountCampaign***.create\_default(**self**.id)  
 *points\_for\_purchases\_campaign* = ***PointsForPurchasesCampaign***.create\_default(**self**.id)  
 *birthday\_campaign* = ***BirthdayCampaign***.create\_default(**self**.id)  
  
 **self**.setup\_default\_newsletter\_signup\_campaign  
 **self**.setup\_default\_referral\_program(**true**)  
  
 **if self**.platform.eql?(**"magento"**) || **self**.platform.eql?(**"magento2"**) || **self**.plan.eql?(**"shopify\_plus"**) || **self**.platform.eql?(**"woocommerce"**) || **self**.platform.eql?(**"commerce\_cloud"**)  
 **self**.program\_live = **false  
 self**.save  
 **end  
  
 self**.setup\_default\_rewards\_popup  
 **self**.setup\_default\_onboarding\_tasks  
 **end  
  
 def** *setup\_default\_onboarding\_tasks* **if self**.platform.eql?(**"bigcommerce"**)  
 **if self**.platform\_engine.eql?(**"stencil"**)  
 ***OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 1, **:task** *=>* **"Setup Swell on your Bigcommerce Storefront"**, **:icon** *=>* **"fa-wrench"**, **:description** *=>* **"Learn how to setup Swell on your Bigcommerce storefront."**, **:short\_name** *=>* **"setup\_swell"**, **:destination** *=>* **"https://www.youtube.com/watch?v=q3fYHFWa1D8"**, **:external\_link** *=>* **true  
 else  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 1, **:task** *=>* **"Setup Swell on your Bigcommerce Storefront"**, **:icon** *=>* **"fa-wrench"**, **:description** *=>* **"Learn how to setup Swell on your Bigcommerce storefront."**, **:short\_name** *=>* **"setup\_swell"**, **:destination** *=>* **"https://www.youtube.com/watch?v=q3fYHFWa1D8"**, **:external\_link** *=>* **true  
 end  
 end  
  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 2, **:task** *=>* **"Customize Rewards Popup"**, **:icon** *=>* **"fa-paint-brush"**, **:description** *=>* **"Take a quick tour to learn how to customize every single aspect of your program."**, **:short\_name** *=>* **"customize\_popup"**, **:destination** *=>* **"/rewards-popup/tour"**, **:external\_link** *=>* **false  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 3, **:task** *=>* **"Review How Your Customers Earn Points"**, **:icon** *=>* **"fa-trophy"**, **:description** *=>* **'Choose the "actions" customers earn rewards for.'**, **:short\_name** *=>* **"customize\_campaigns"**, **:destination** *=>* **"/campaigns"**, **:external\_link** *=>* **false  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 4, **:task** *=>* **"Review How Your Customers Spend Points"**, **:icon** *=>* **"fa-gift"**, **:description** *=>* **"Change, add, or remove redemption options for your customers."**, **:short\_name** *=>* **"customize\_coupons"**, **:destination** *=>* **"/coupons"**, **:external\_link** *=>* **false  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 5, **:task** *=>* **"Learn How to Set Up Your Referral Program"**, **:icon** *=>* **"fa-users"**, **:description** *=>* **"Watch a brief video to learn how to utilize Swell's powerful referral program."**, **:short\_name** *=>* **"learn\_referral"**, **:destination** *=>* **"https://www.youtube.com/watch?v=nDiEkvA394k"**, **:external\_link** *=>* **true  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 6, **:task** *=>* **"Learn How to Use Swell's Newsletter Signup Tool"**, **:icon** *=>* **"fa-envelope"**, **:description** *=>* **"In just a few minutes, learn how to use Swell to boost your email subscriber base."**, **:short\_name** *=>* **"learn\_newsletter"**, **:destination** *=>* **"https://www.youtube.com/watch?v=q3fYHFWa1D8"**, **:external\_link** *=>* **true  
 end  
  
   
 def** *setup\_rewards\_program  
 manual\_adjustment\_campaign* = ***CustomActionCampaign***.create\_default(**self**.id)  
 *manual\_adjustment\_csv\_campaign* = ***CustomActionCampaign***.create\_default\_csv(**self**.id)  
  
 *vec* = ***VipTierEarned***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **false** *rc* = ***RedeemedCoupon***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **true** *mrc* = ***MerchantPointsRedeemed***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **false** *cpr* = ***CustomerPointsReminder***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **false**, **:delay\_days** *=>* 30  
 *crr* = ***CustomerRedemptionReminder***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **false**, **:delay\_days** *=>* 30  
 *rty* = ***ReferrerThankYou***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **false** *res* = ***ReferralEmailShare***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **true** *res* = ***ReferralEmailShareReminder***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **true** *wd* = ***WeeklyDigest***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **true** *hb* = ***HappyBirthday***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **true  
 *ReferralBlockedDomain***.*create* **:merchant\_id** *=>* **self**.id, **:active** *=>* **true  
 if self**.plan.eql?(**"shopify\_plus"**)  
 ***RedeemedStoreGiftCard***.*create* **:merchant\_id** *=>* **self**.id  
 **end  
  
 self**.program\_live = **false  
  
 if *Rails***.env.test?  
 **self**.setup\_default\_rewards\_program  
 **self**.program\_live = **true  
 end  
   
 self**.save  
  
 **end  
  
 def** *setup\_onboarding\_tasks*(*plan*, *display\_type*, *email\_provider*, *email\_notification*, *campaign\_details*)  
 **if self**.platform.eql?(**"bigcommerce"**)  
 **if self**.platform\_engine.eql?(**"stencil"**)  
 ***OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 1, **:task** *=>* **"Setup Swell on your Bigcommerce Storefront"**, **:icon** *=>* **"fa-wrench"**, **:description** *=>* **"Learn how to setup Swell on your Bigcommerce storefront."**, **:short\_name** *=>* **"setup\_swell"**, **:destination** *=>* **"https://www.youtube.com/watch?v=q3fYHFWa1D8"**, **:external\_link** *=>* **true  
 else  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 1, **:task** *=>* **"Setup Swell on your Bigcommerce Storefront"**, **:icon** *=>* **"fa-wrench"**, **:description** *=>* **"Learn how to setup Swell on your Bigcommerce storefront."**, **:short\_name** *=>* **"setup\_swell"**, **:destination** *=>* **"https://www.youtube.com/watch?v=q3fYHFWa1D8"**, **:external\_link** *=>* **true  
 end  
 end  
  
 if** *campaign\_details*.present?  
 *campaign\_count* = *campaign\_details*.count  
 *using\_referral\_program* = *campaign\_details*.select { | *campaign* | *campaign*[**"name"**] == **"ReferralCampaign"** }.present?  
 *using\_newsletter\_signup\_campaign* = *campaign\_details*.select {|*campaign*| *campaign*[**"name"**] == **"NewsletterSignupCampaign"** }.present?  
 *campaign\_count* -=1 **if** *using\_referral\_program  
 using\_rewards\_program* = *campaign\_count* > 0  
 *using\_only\_referral\_program* = *using\_referral\_program* && !*using\_rewards\_program* **else** *#* ***when*** *is campaign\_details not present?  
 using\_referral\_program* = **false** *using\_rewards\_program* = **false** *using\_only\_referral\_program* = **false** *using\_newsletter\_signup\_campaign* = **false  
 end  
  
 if** *using\_rewards\_program* **if** *display\_type* == **"standard"  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 2, **:task** *=>* **"Personalize your Rewards Popup and Tab"**, **:icon** *=>* **"fa-paint-brush"**, **:description** *=>* **"Customize the text, colors, fonts, and icons of your rewards popup to match your brand."**, **:short\_name** *=>* **"customize\_popup"**, **:destination** *=>* **"/rewards-popup"**, **:external\_link** *=>* **false  
 elsif** *display\_type* == **"premium"  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 2, **:task** *=>* **"Personalize your Premium Rewards Popup and Tab"**, **:icon** *=>* **"fa-paint-brush"**, **:description** *=>* **"Customize the images, text, colors, fonts, and icons of your premium rewards popup to match your brand."**, **:short\_name** *=>* **"customize\_popup"**, **:destination** *=>* **"/rewards-popup"**, **:external\_link** *=>* **false  
 elsif** *display\_type* == **"page"  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 2, **:task** *=>* **"Create and design your Rewards Page"**, **:icon** *=>* **"fa-paint-brush"**, **:description** *=>* **"Customize the text, colors, fonts, and icons of your rewards page to match your brand."**, **:short\_name** *=>* **"customize\_popup"**, **:destination** *=>* **"/rewards-page"**, **:external\_link** *=>* **false  
 end  
 end  
  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 3, **:task** *=>* **"Review how your customers earn points"**, **:icon** *=>* **"fa-trophy"**, **:description** *=>* **'Choose the "actions" customers earn rewards for.'**, **:short\_name** *=>* **"customize\_campaigns"**, **:destination** *=>* **"/campaigns"**, **:external\_link** *=>* **false  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 4, **:task** *=>* **"Review how your customers spend points"**, **:icon** *=>* **"fa-gift"**, **:description** *=>* **"Change, add, or remove redemption options for your customers."**, **:short\_name** *=>* **"customize\_coupons"**, **:destination** *=>* **"/coupons"**, **:external\_link** *=>* **false  
  
 if** *using\_referral\_program* ***OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 5, **:task** *=>* **"Customize your Referral Program and Popups"**, **:icon** *=>* **"fa-users"**, **:description** *=>* **"Customize referred and referral customer rewards and popup's text, colors, and fonts"**, **:short\_name** *=>* **"learn\_referral"**, **:destination** *=>* **"https://www.youtube.com/watch?v=nDiEkvA394k"**, **:external\_link** *=>* **true  
 end  
  
 if** *using\_newsletter\_signup\_campaign* ***OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 6, **:task** *=>* **"Design your Newsletter Signup Popup"**, **:icon** *=>* **"fa-envelope"**, **:description** *=>* **"Customize the background image, text, colors, and fonts for your email capture form"**, **:short\_name** *=>* **"learn\_newsletter"**, **:destination** *=>* **"https://www.youtube.com/watch?v=q3fYHFWa1D8"**, **:external\_link** *=>* **true  
 end  
  
 if** *email\_notification*.present?  
 ***OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 7, **:task** *=>* **"Customize your Email Notifications messaging"**, **:icon** *=>* **"fa-envelope"**, **:description** *=>* **"Easily change the subject, body, and layout used for these email notifications."**, **:short\_name** *=>* **"customize email notifications"**, **:destination** *=>* **"/email-notifications"**, **:external\_link** *=>* **false  
 end  
  
 if** *plan*.eql?(**"GOLD"**)  
 **if** *email\_provider*.present? && *email\_provider*[**:emailProvider**].present?  
 ***OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 8, **:task** *=>* **"Setup your Email Service Provider"**, **:icon** *=>* **"fa-envelope-o"**, **:description** *=>* **"Connect your ESP to take full control of every email sent."**, **:short\_name** *=>* **"email\_provider"**, **:destination** *=>* **"/settings"**, **:external\_link** *=>* **false  
 end  
  
 *OnboardingTask***.*create* **:merchant\_id** *=>* **self**.id, **:display\_order** *=>* 9, **:task** *=>* **"Setup and Customize your VIP Tiers"**, **:icon** *=>* **"fa-trophy"**, **:description** *=>* **"Setup the requirements and reawrds for each tier in your VIP program."**, **:short\_name** *=>* **"email\_provider"**, **:destination** *=>* **"/vip-tiers"**, **:external\_link** *=>* **false  
 end  
 end  
  
 def** *platform\_percentage\_discount\_type  
 discount\_type* = **"fixed\_amount"  
 if self**.platform.eql?(**"custom"**) || **self**.platform.eql?(**"commerce\_cloud"**)  
 *discount\_type* = **"custom"  
 end** *discount\_type* **end  
  
 def** *platform\_fixed\_amount\_discount\_type  
 discount\_type* = **"fixed\_amount"  
 if self**.platform.eql?(**"custom"**)  
 *discount\_type* = **"custom"  
 elsif self**.platform.eql?(**"commerce\_cloud"**)  
 *discount\_type* = **"gift\_certificate\_fixed"  
 end** *discount\_type* **end  
  
 def** *default\_newsletter\_coupon  
 discount\_type* = **self**.platform\_percentage\_discount\_type  
 ***Coupon***.find\_or\_create\_by(**:discount\_percentage** *=>* 10, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **end  
  
 def** *default\_referral\_coupon  
 discount\_type* = **self**.platform\_percentage\_discount\_type  
 ***Coupon***.find\_or\_create\_by(**:discount\_percentage** *=>* 15, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **end  
  
 def** *ensure\_default\_coupons* **self**.ensure\_default\_fixed\_coupons  
 **self**.ensure\_default\_percentage\_coupons  
 **end  
  
 def** *ensure\_default\_fixed\_coupons  
 discount\_type* = **self**.platform\_fixed\_amount\_discount\_type  
  
 *five\_dollars\_off* = ***Coupon***.find\_or\_create\_by(**:discount\_amount\_cents** *=>* 500, **:single\_use** *=>* **true**, **:amount** *=>* 500, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 *ten\_dollars\_off* = ***Coupon***.find\_or\_create\_by(**:discount\_amount\_cents** *=>* 1000, **:single\_use** *=>* **true**, **:amount** *=>* 1000, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 *twentyfive\_dollars\_off* = ***Coupon***.find\_or\_create\_by(**:discount\_amount\_cents** *=>* 2500, **:single\_use** *=>* **true**, **:amount** *=>* 2500, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **end  
  
 def** *ensure\_default\_percentage\_coupons  
 discount\_type* = **self**.platform\_percentage\_discount\_type  
  
 *ten\_percent\_off* = ***Coupon***.find\_or\_create\_by(**:discount\_percentage** *=>* 10, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 *fifteen\_percent\_off* = ***Coupon***.find\_or\_create\_by(**:discount\_percentage** *=>* 15, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 *twentyfive\_percent\_off* = ***Coupon***.find\_or\_create\_by(**:discount\_percentage** *=>* 25, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **end  
  
 def** *setup\_default\_newsletter\_signup\_campaign  
 newsletter\_signup\_campaign* = **self**.campaigns.*where*(**:type** *=>* **"NewsletterSignupCampaign"**, **:active** *=>* **true**).take  
  
 **unless** *newsletter\_signup\_campaign  
 newsletter\_signup\_campaign* = ***NewsletterSignupCampaign***.create\_default(**self**.id, **self**.default\_newsletter\_coupon.id)  
 *newsletter\_signup\_campaign*.setup\_default\_campaign\_popup\_configurations  
 **end  
 end  
  
 def** *setup\_locked\_customer\_campaing  
 newsletter\_signup\_campaign* = **self**.campaigns.*where*(**:type** *=>* **"NewsletterSignupCampaign"**, **:active** *=>* **true**).take  
  
 **unless** *newsletter\_signup\_campaign  
 newsletter\_signup\_campaign* = ***NewsletterSignupCampaign***.create\_default(**self**.id, **self**.default\_newsletter\_coupon.id)  
 *newsletter\_signup\_campaign*.setup\_default\_campaign\_popup\_configurations  
 **end  
 end  
  
 def** *setup\_default\_rewards\_popup  
 rewards\_popup* = **self**.campaign\_popup\_configurations.*where*(**:type** *=>* **"RewardsPopup"**).take  
 *rewards\_popup* = ***RewardsPopup***.create\_default(**self**.id) **if** *rewards\_popup*.nil?  
   
 *localization* = **self**.localizations.*where*(**:campaign\_popup\_configuration\_id** *=> rewards\_popup*.id).take   
 **return** *rewards\_popup* **if** *localization*.present?  
  
 *localization* = **self**.localizations.*create* **:campaign\_popup\_configuration\_id** *=> rewards\_popup*.id, **:faqs\_html** *=>***%q(<h2 class='swell-question'>How do I participate?</h2>  
<p class='swell-answer'>Joining is easy! Just click the Create An Account button to get started. Once you're registered with our store, you'll have the opportunity to take part in all of the exciting ways we currently offer to earn points!</p>  
  
<h2 class='swell-question'>How can I earn points?</h2>  
<p class='swell-answer'>You can earn points by participating in any of our innovative promotions! Simply click on the 'Earn Points' tab to view and take part in our current opportunities. In addition, make sure to check back often, as we're adding great new ways for you to earn points all the time!</p>  
  
<h2 class='swell-question'>What can I redeem my points for?</h2>  
<p class='swell-answer'>Glad you asked! We want to make it easy and fun to redeem your hard-earned points. Just visit the 'Get Rewards' tab to view all of our exciting reward options.</p>  
  
<h2 class='swell-question'>How do I redeem my points?</h2>  
<p class='swell-answer'>Exchanging your points for great rewards couldn't be easier! Simply visit the 'Get Rewards' tab to view all of our great reward options and click the 'Redeem' button to redeem your reward.</p>  
  
<h2 class='swell-question'>How do I check my points balance?</h2>  
<p class='swell-answer'>Your up-to-date points balance is always displayed in the top of this popup.</p>  
  
<h2 class='swell-question'>Does it cost anything to begin earning points?</h2>  
<p class='swell-answer'>Absolutely not! Sign up is 100% free, and it will never cost you anything to earn points. Make sure to visit the 'Earn Points' tab to get started.</p>  
  
<h2 class='swell-question'>Do I have to enroll or register in individual promotions?</h2>  
<p class='swell-answer'>Once you register for an account, you're all set – we don't require you to register for individual promotions in order to be eligible. Just fulfill the requirements of a promotion, and we'll post the points to your account immediately!</p>  
  
<h2 class='swell-question'>How long will it take for points to post to my account?</h2>  
<p class='swell-answer'>You should receive points in your account instantly once you complete a promotion!</p>  
  
<h2 class='swell-question'>Do my points expire?</h2>  
<p class='swell-answer'>Nope! Your points will never expire.</p>  
  
<h2 class='swell-question'>What happens to my points if I make a return?</h2>  
<p class='swell-answer'>When you return an item, you lose the associated credit you originally earned by buying the item in the first place. <br/> Sound kind of confusing? Let's take an example: let's say you had previously spent $50 towards a 'spend $100, earn 500 points' promotion, and you decide to buy a $20 item, which bumps you up to $70. If you decide to return that item, your progress would also go back down to $50 – it's just like you hadn't bought the item in the first place.</p>  
  
<h2 class='swell-question'>How do I contact support if I have questions about my points?</h2>  
<p class='swell-answer'>Our team is ready and waiting to answer your questions about our rewards program! Just send us an email and we'll be in touch.</p>  
  
<h2 class='swell-question'> I'm very close to earning a reward. Can I buy extra points to get there?</h2>  
<p class='swell-answer'>We currently require you to have enough points to redeem any of the awards you see listed on the 'Get Rewards' tab.</p>  
  
<h2 class='swell-question'>What if I don't want to receive promotional emails?</h2>  
<p class='swell-answer'>From time to time, you'll receive program-related emails from us. If you'd prefer to not receive those types of emails anymore, just click the 'Unsubscribe' button when you receive your next email.</p>  
)** *rewards\_popup* **end  
  
 def** *setup\_default\_referral\_program*(*active* = **false**)  
 *referral\_campaign* = ***ReferralCampaign***.create\_default(**self**.id, **self**.default\_referral\_coupon.id)  
 *referral\_campaign*.active = *active  
 referral\_campaign*.setup\_default\_campaign\_popup\_configurations  
 *referral\_campaign*.save  
 *referral\_campaign* **end  
  
 def** *store\_login\_path  
 platform\_login\_path\_map* = {  
 **shopify**: **"/account/login"**,  
 **magento**: **"/index.php/customer/account/login/"**,  
 **magento2**: **"/customer/account/login/"**,  
 **bigcommerce**: **"/login.php"**,  
 **woocommerce**: **"/my-account"**,  
 **commerce\_cloud**: **"/account"** }  
  
 *platform\_login\_path\_map*[**self**.platform.to\_sym]  
 **end  
  
 def** *store\_register\_path  
 platform\_register\_path\_map* = {  
 **shopify**: **"/account/register"**,  
 **magento**: **"/index.php/customer/account/create/"**,  
 **magento2**: **"/customer/account/create/"**,  
 **bigcommerce**: **"/login.php?action=create\_account"**,  
 **woocommerce**: **"/my-account"**,  
 **commerce\_cloud**: **"/register"** }  
  
 *platform\_register\_path\_map*[**self**.platform.to\_sym]  
 **end  
  
 def** *protocol\_independent\_website* **self**.website.gsub(**/**https?:**\/\//**,**"//"**)  
 **end  
  
 def** *has\_known\_store\_login\_register\_path* **self**.store\_login\_path.present? && **self**.store\_register\_path.present?  
 **end  
   
 def** *handle\_website\_changed* **if self**.website.present?  
 **if self**.website.index(**"http"**).nil?  
 **self**.website = **"http://"** + **self**.website  
 **end  
  
 if self**.has\_known\_store\_login\_register\_path  
 **self**.store\_account\_registration\_url = **self**.protocol\_independent\_website + **self**.store\_register\_path  
 **self**.store\_account\_login\_url = **self**.protocol\_independent\_website + **self**.store\_login\_path  
 **end  
 end  
 end  
  
 def** *thirty\_day\_order\_volume*(*end\_time* = ***Time***.now)  
 **self**.purchases.*where*(**"created\_at > ? and created\_at <= ? and amount\_cents > 0"**, *end\_time* - (30\*24).hours, *end\_time*).count  
 **end  
  
 def** *yesterdays\_thirty\_day\_order\_volume* **self**.purchases.*where*(**"created\_at > ? and created\_at < ? and amount\_cents > 0"**, ***Time***.now - (31 \* 24).hours, ***Time***.now - 24.hours).count  
 **end  
  
 def** *monitor\_swell\_redemption\_options* **if self**.swell\_redemptions\_enabled\_changed? || **self**.points\_per\_dollar\_changed?  
 ***AccountProcessing***::***SyncRewardOptionsJob***.perform\_async(**self**.id)  
 **end  
 end  
  
 def** *monitor\_newsletter\_sync* **if self**.sync\_newsletter\_with\_platform\_changed? && **self**.sync\_newsletter\_with\_platform  
 ***AccountProcessing***::***SyncNewsletterSignupsJob***.perform\_async(**self**.id)  
 **end  
  
 if self**.manual\_adjustment\_admin\_only\_changed?  
 **self**.campaigns.*where*(**:type** *=>* **"CustomActionCampaign"**, **:action\_name** *=>* **"manual\_adjustment"**).find\_each **do** |*campaign*|  
 *campaign*.admin\_only = **self**.manual\_adjustment\_admin\_only  
 *campaign*.save  
 **end  
 end  
 end  
  
 def** *customer\_for\_user*(*user*)  
 customers.*where*(**:identity\_id** *=> user*.identity.id).take  
 **end  
  
 def** *create\_store\_account*(*params*)  
 **if** platform.eql?(**"shopify"**)  
 *shopify\_processor* = processors.*where*(**:type** *=>* **"ShopifyProcessor"**).take  
 **if** *shopify\_processor  
 shopify\_processor*.create\_store\_account(*params*)  
 **end  
 end  
 end  
  
 def** *sync\_reward\_options* **if self**.swell\_redemptions\_enabled  
 ***RedemptionOption***.*where*(**'merchant\_id is null'**).find\_each **do** |*ro*|  
 *option* = ***RedemptionOption***.*where*(**:merchant\_id** *=>* **self**.id, **:parent\_option\_id** *=> ro*.id).take  
  
 **unless** *option  
 option* = *ro*.dup  
 *option*.merchant\_id = **self**.id  
 *option*.parent\_option\_id = *ro*.id  
 **end** *option*.amount = (*ro*.value\_cents / 100) \* **self**.points\_per\_dollar  
 *option*.save  
 **end  
 else** *# delete all non-merchant created options they have setup* ***RedemptionOption***.*where*(**:type** *=>* **"GiftCard"**, **:merchant\_id** *=>* **self**.id).delete\_all  
 **end  
 end  
  
 def** *find\_customer\_by\_id*(*id*)  
 **if** platform.eql?(**"shopify"**)  
 *shopify\_processor* = processors.*where*(**:type** *=>* **"ShopifyProcessor"**).take  
 *shopify\_processor*.find\_customer\_by\_id(*id*)  
 **else  
 nil  
 end  
 end  
  
 def** *find\_customer\_by\_email*(*email*)  
 **if** platform.eql?(**"shopify"**)  
 *shopify\_processor* = processors.*where*(**:type** *=>* **"ShopifyProcessor"**).take  
 *shopify\_processor*.find\_customer\_by\_email(*email*)  
 **else  
 nil  
 end  
 end  
  
 def** *send\_preview\_email  
 content\_body* = ***ActionController***::***Base***.new.*render\_to\_string*(**"emails/campaigns/preview.html.erb"**, **:layout** *=>* **false**)  
  
 **if** email\_layout.eql?(**"custom"**)  
 *content* = ***ActionController***::***Base***.new.*render\_to\_string*(**:text** *=>* email\_layout\_html.gsub(**"\*|BODY|\*"**, *content\_body*), **:layout** *=>* **false**)  
 **else** *content* = ***ActionController***::***Base***.new.*render\_to\_string*(**:text** *=> content\_body*, **:layout** *=>* **"email\_campaign"**, **:locals** *=>* {**:header\_image\_url** *=>* email\_header\_image.url, **:header\_image\_width** *=>* email\_header\_width})  
 **end** *mail* = ***SendGrid***::***Mail***.new  
 *mail*.from = ***SendGrid***::***Email***.new(**email**: **self**.from\_email, **name**: **self**.from\_name)  
 *mail*.subject = **"[Swell Preview] - Customer Email Template"** *mail*.contents = ***SendGrid***::***Content***.new(**type**: **'text/html'**, **value**: *content*)  
 *personalization* = ***SendGrid***::***Personalization***.new  
 *personalization*.to = ***SendGrid***::***Email***.new(**email**: **self**.contact\_email)  
 *mail*.personalizations = *personalization  
 sg* = ***SendGrid***::***API***.new(**api\_key**: ***ENV***[**'SENDGRID\_API\_KEY'**])  
 *response* = *sg*.client.mail.\_(**'send'**).post(**request\_body**: *mail*.to\_json)  
 **end  
  
 def** *send\_digest\_email*(*most\_recent\_blog\_post* = **nil**)  
 **if self**.weekly\_digest\_enabled  
  
 **unless** *most\_recent\_blog\_post  
 feed\_url* = **"http://blog.swellrewards.com/feed"** *feed* = ***Feedjira***::***Feed***.fetch\_and\_parse(*feed\_url*)  
 *most\_recent\_blog\_post* = *feed*.entries.first  
 **end** *incomplete\_onboarding\_tasks* = **self**.onboarding\_tasks.*where*(**:completed** *=>* **false**).count  
  
 *free\_trial\_days\_left* = **self**.days\_left\_on\_free\_trial  
  
 *is\_free\_user* = **self**.current\_subscription.plan.name.eql?(**"BRONZE"**)  
 *program\_paused* = !**self**.program\_live  
 *not\_using\_newsletter\_tool* = **self**.campaign\_popup\_configurations.*where*(**:type** *=>* **"EmailCapturePopup"**, **:active** *=>* **true**).count == 0  
 *not\_using\_referral\_program* = **self**.campaigns.*where*(**:type** *=>* **"ReferralCampaign"**, **:active** *=>* **true**).count == 0  
 *not\_using\_vip\_tiers* = !**self**.is\_vip\_tiers\_enabled?  
  
 *campaign\_types* = **self**.campaigns.*where*(**"active=true and hidden=false and type not in (?)"**,[**"ReferralCampaign"**, **"NewsletterSignupCampaign"**]).*pluck*(**:type**).uniq  
 *not\_using\_rewards* = *campaign\_types*.count == 0  
  
 *recommended\_campaign\_types* = [**"CustomActionCampaign"**, **"PointsForPurchasesCampaign"**, **"BirthdayCampaign"**, **"CreateAccountCampaign"**, **"FacebookPageVisitCampaign"**, **"FacebookShareCampaign"**, **"InstagramFollowCampaign"**, **"ReadContentCampaign"**, **"SpendBasedCampaign"**, **"TransactionBasedCampaign"**, **"TwitterFollowCampaign"**, **"TwitterRetweetCampaign"**, **"TwitterTweetCampaign"**, **"YotpoReviewCampaign"**]  
 *campaign\_recommendations* = ***Hash***.new  
  
 *campaign\_recommendations*[**"CustomActionCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"CustomActionCampaign"**][**:message**] = **"Swell's Custom Action campaign is a great way to reward customers for anything you want! Use this to incentivize any action you'd like your customers to perform. Technical expertise required to send notifications to Swell."** *campaign\_recommendations*[**"CustomActionCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"CustomActionCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/custom-action"** *campaign\_recommendations*[**"PointsForPurchasesCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"PointsForPurchasesCampaign"**][**:message**] = **"Swell's Points for Purchases campaign lets you reward points for every dollar spent. It helps e-commerce merchants increase average cart size and purchase frequency."** *campaign\_recommendations*[**"PointsForPurchasesCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"PointsForPurchasesCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/points-for-purchases"** *campaign\_recommendations*[**"BirthdayCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"BirthdayCampaign"**][**:message**] = **"Swell's Happy Birthday campaign is a great way to reward your customers with a gift on their birthday. This drives massive customer engagement, happiness and new marginal revenue."** *campaign\_recommendations*[**"BirthdayCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"BirthdayCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/happy-birthday"** *campaign\_recommendations*[**"CreateAccountCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"CreateAccountCampaign"**][**:message**] = **"Swell's Create Account campaign is a great way to reward customers for creating an account! Having an account is an incredible way to engage and educate customers between purchases."** *campaign\_recommendations*[**"CreateAccountCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"CreateAccountCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/create-account"** *campaign\_recommendations*[**"FacebookPageVisitCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"FacebookPageVisitCampaign"**][**:message**] = **"Swell's Facebook Page Visit campaign is a great way to incentivize visits to your Facebook Page! It's an excellent way to drive engagement and generate Page Likes."** *campaign\_recommendations*[**"FacebookPageVisitCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"FacebookPageVisitCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/facebook-page-visit"** *campaign\_recommendations*[**"FacebookShareCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"FacebookShareCampaign"**][**:message**] = **"Swell's Facebook Share campaign is a great way to incentivize sharing your website on Facebook! It's an excellent way to drive engagement and generate traffic."** *campaign\_recommendations*[**"FacebookShareCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"FacebookShareCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/facebook-share"** *campaign\_recommendations*[**"InstagramFollowCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"InstagramFollowCampaign"**][**:message**] = **"Swell's Instagram Follow campaign is a great way to reward consumers for following your business on Instagram! There aren't many better ways to boost engagement than this."** *campaign\_recommendations*[**"InstagramFollowCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"InstagramFollowCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/instagram-follow"** *campaign\_recommendations*[**"ReadContentCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"ReadContentCampaign"**][**:message**] = **"Swell's Read Content campaign is a great way to reward consumers for reading a specific piece of content. It is an incredible way to deeply educate consumers about your business."** *campaign\_recommendations*[**"ReadContentCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"ReadContentCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/read-content"** *campaign\_recommendations*[**"SpendBasedCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"SpendBasedCampaign"**][**:message**] = **"Swell's Goal Spend campaign is a great way to reward customers after spending a certain amount with your store. This campaign helps e-commerce merchants increase average cart size and purchase frequency."** *campaign\_recommendations*[**"SpendBasedCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"SpendBasedCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/goal-spend"** *campaign\_recommendations*[**"TransactionBasedCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"TransactionBasedCampaign"**][**:message**] = **"Swell's Punch Card campaign is a great way to reward customers after making a certain number of purchases with your store. This campaign is proven to increase purchase frequency for e-commerce businesses."** *campaign\_recommendations*[**"TransactionBasedCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"TransactionBasedCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/punch-card"** *campaign\_recommendations*[**"TwitterFollowCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"TwitterFollowCampaign"**][**:message**] = **"Swell's Twitter Follow campaign is a great way to reward consumers for following your business on Twitter! There aren't many better ways to boost engagement than this one."** *campaign\_recommendations*[**"TwitterFollowCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"TwitterFollowCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/twitter-follow"** *campaign\_recommendations*[**"TwitterRetweetCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"TwitterRetweetCampaign"**][**:message**] = **"Swell's Twitter Retweet campaign is a great way to communicate an important message. Use this campaign to drastically increase its reach on Twitter!"** *campaign\_recommendations*[**"TwitterRetweetCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"TwitterRetweetCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/retweet"** *campaign\_recommendations*[**"TwitterTweetCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"TwitterTweetCampaign"**][**:message**] = **"Swell's Tweet About Us campaign is a great way to encourage consumers to Tweet about you to their followers! This is a great way to generate loads of new impressions."** *campaign\_recommendations*[**"TwitterTweetCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"TwitterTweetCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/tweet-about-us"** *campaign\_recommendations*[**"YotpoReviewCampaign"**] = ***Hash***.new  
 *campaign\_recommendations*[**"YotpoReviewCampaign"**][**:message**] = **"Swell's Yotpo Review campaign is a great way to reward your customers for writing product reviews! This campaign is an excellent way for Yotpo users to dramatically increase review counts."** *campaign\_recommendations*[**"YotpoReviewCampaign"**][**:button\_text**] = **"START ONE TODAY"** *campaign\_recommendations*[**"YotpoReviewCampaign"**][**:link**] = **"https://app.swellrewards.com/campaigns/new/yotpo-review"** *email\_data* = ***Hash***.new  
  
 *recommendations\_arr* = []  
 *email\_data*[**:recommendation**] = []  
  
 *recommended\_campaign\_types*.each **do** |*recommended\_campaign\_type*|  
 **if** *campaign\_types*.index(*recommended\_campaign\_type*).nil? && *campaign\_recommendations*[*recommended\_campaign\_type*].present?  
 *recommendations\_arr*.push *campaign\_recommendations*[*recommended\_campaign\_type*]  
 **end  
 end  
  
 if** *program\_paused* **if self**.thirty\_day\_order\_volume > 800  
 *recommendation* = { **message**: **"We noticed that your program is currently paused. Is there anything we can do to help get you set up? If so, simply chat with us while in your Swell admin or send us an email. Our team is standing by to assist!"**, **button\_text**: **"Send Email"**, **link**: **"mailto:support@swellrewards.com?subject=Help with Swell Setup!"**}  
 *recommendations\_arr*.push *recommendation* **else** *recommendation* = {**message**: **"We noticed that your program is currently paused -- need help getting set up? If so, check out our detailed help section, which has over 70 articles to guide you through the setup process!"**, **button\_text**: **"Get Help"**, **link**: **"https://support.yotpo.com/en/swell-loyalty-referrals/"**}  
 *recommendations\_arr*.push *recommendation* **end  
 end  
  
 if** *incomplete\_onboarding\_tasks* > 0  
 *recommendation* = { **message**: **"It looks like you haven't finished exploring all of the exciting features that Swell has to offer! Simply visit the explore section of your account to learn more."**, **button\_text**: **"Explore Swell"**, **link**: **"https://app.swellrewards.com/explore"**}  
 *recommendations\_arr*.push *recommendation* **end** *#* ***if*** *is\_free\_user  
 # recommendation = {message:* ***"Did you know that our average merchant on a Swell Premium plan earns a 9:1 ROI on their program? <a target='\_blank' href='https://support.yotpo.com/en/swell-loyalty-referrals/'>Upgrade today</a> to ensure you have access to all of our powerful premium features."****, button\_text:* ***"Check premium features"****, link:* ***"http://help.swellrewards.com/collection/61-premium-features"****}  
 # recommendations\_arr.push recommendation  
 #* ***end* if** *not\_using\_vip\_tiers  
 recommendation* = {**message**: **"VIP Tiers are a great way to recognize your most valuable customers. Tiers create incentives that drastically increase order frequency and cart size."**, **button\_text**: **"Setup VIP Tiers"**, **link**: **"https://app.swellrewards.com/vip-tiers"**}  
 *recommendations\_arr*.push *recommendation* **end  
  
 if** *not\_using\_referral\_program  
 recommendation* = {**message**: **"Give your most dedicated customers a reason to tell their friends about your business. Join thousands of other merchants who are using Swell's referral program to earn loads of new customers!"**, **button\_text**: **"Setup Referral Program"**, **link**: **"https://app.swellrewards.com/referral-program"**}  
 *recommendations\_arr*.push *recommendation* **end  
  
 if** *not\_using\_rewards  
 recommendation* = {**message**: **"Have you considered whether a points program is right for your business? Reward your customers for over 10 unique 'actions', including purchases, social media activity, and much more! Give Swell's rewards program a try today."**, **button\_text**: **"Setup Rewards Program"**, **link**:**"https://app.swellrewards.com/campaigns"**}  
 *recommendations\_arr*.push *recommendation* **end  
  
 if** *not\_using\_newsletter\_tool  
 recommendation* = {**message**: **"Swell's Newsletter Signup tool allows you to capture more emails with a beautiful popup that appears when and where you'd like it to. Consider giving it a try today!"**, **button\_text**: **"Setup Newsletter Signup"**, **link**: **"https://app.swellrewards.com/newsletter-signup"**}  
 *recommendations\_arr*.push *recommendation* **end** *email\_data*[**:latest\_from\_blog**] = *most\_recent\_blog\_post  
  
 current\_plan\_name* = **self**.current\_subscription.plan.name.downcase  
  
 *upsell\_map* = {**"bronze"** *=>* **"silver"**, **"silver"** *=>* **"gold"**, **"gold"** *=>* **"platinum"**, **"platinum"** *=>* **nil**, **"enterprise"** *=>* **nil**}  
 *view\_upsell* = *upsell\_map*[*current\_plan\_name*]  
  
 *# override upsells to higher plan based on order volume  
 tdov* = **self**.thirty\_day\_order\_volume  
 **if** *tdov* >= 400 && (*current\_plan\_name*.eql?(**"bronze"**) || *current\_plan\_name*.eql?(**"silver"**))  
 *view\_upsell* = **"gold"  
 elsif** *tdov* >= 2000 && (*current\_plan\_name*.eql?(**"bronze"**) || *current\_plan\_name*.eql?(**"silver"**) || *current\_plan\_name*.eql?(**"gold"**))  
 *view\_upsell* = **"platinum"  
 end** *email\_data*[**:view\_upsell**] = *view\_upsell  
 email\_data*[**:company\_name**] = **self**.company\_name  
  
 *email\_data*[**:current\_date**] = ***Date***.today.try(**:strftime**, **"%B %d, %Y"**) **rescue 'N/A'** *email\_data*[**:beginning\_of\_week**] = ***Date***.today.beginning\_of\_week.try(**:strftime**, **"%B %d"**) **rescue 'N/A'** *email\_data*[**:end\_of\_week**] = ***Date***.today.end\_of\_week.try(**:strftime**, **"%B %d"**) **rescue 'N/A'** *email\_data*[**:owner\_name**] = **self**.owner.name  
 *email\_data*[**:recommendation**] = *recommendations\_arr*[rand(*recommendations\_arr*.length)]  
 *email\_data*[**:roi**] = **self**.lifetime\_roi  
  
 *email\_data*[**:week\_end\_date**] = ***Date***.today  
 *email\_data*[**:week\_start\_date**] = *email\_data*[**:week\_end\_date**] - 7.days  
 *last\_week\_snapshots* = **self**.dashboard\_snapshots.*where*(**"day >= ? and day < ?"**, *email\_data*[**:week\_start\_date**], *email\_data*[**:week\_end\_date**])  
  
 *email\_data*[**:referrals\_completed**] = ***ActiveSupport***::***NumberHelper***.number\_to\_delimited(**self**.user\_actions.*where*(**:type** *=>* **"ReferrerCompletion"**).count)  
 *email\_data*[**:last\_week\_referrals\_completed**] = ***ActiveSupport***::***NumberHelper***.number\_to\_delimited(**self**.user\_actions.*where*(**"type = 'ReferrerCompletion' and date(created\_at) >= ?"**, *email\_data*[**:week\_start\_date**]).count)  
  
 *email\_data*[**:value\_cents**] = **self**.lifetime\_value\_cents  
 *email\_data*[**:last\_week\_value\_added\_cents**] = *last\_week\_snapshots*.*sum*(**:total\_value\_cents**)  
  
 *email\_data*[**:pretty\_value\_cents**] = ***Money***.new(*email\_data*[**:value\_cents**], **self**.currency).format  
 *email\_data*[**:pretty\_last\_week\_value\_added\_cents**] = ***Money***.new(*email\_data*[**:last\_week\_value\_added\_cents**], **self**.currency).format  
  
 *email\_data*[**:redemptions**] = **self**.dashboard\_snapshots.*sum*(**:redemptions**)  
 *email\_data*[**:last\_week\_redemptions\_added**] = *last\_week\_snapshots*.*sum*(**:redemptions**)  
  
 *email\_data*[**:pretty\_redemptions**] = ***ActiveSupport***::***NumberHelper***.number\_to\_delimited(*email\_data*[**:redemptions**])  
 *email\_data*[**:pretty\_last\_week\_redemptions\_added**] = ***ActiveSupport***::***NumberHelper***.number\_to\_delimited(*email\_data*[**:last\_week\_redemptions\_added**])  
  
 *email\_data*[**:points\_earned**] = **self**.dashboard\_snapshots.*sum*(**:points\_earned**)  
 *email\_data*[**:last\_week\_points\_earned\_added**]= *last\_week\_snapshots*.*sum*(**:points\_earned**)  
  
 *email\_data*[**:pretty\_points\_earned**] = ***ActiveSupport***::***NumberHelper***.number\_to\_delimited(*email\_data*[**:points\_earned**])  
 *email\_data*[**:pretty\_last\_week\_points\_earned\_added**] = ***ActiveSupport***::***NumberHelper***.number\_to\_delimited(*email\_data*[**:last\_week\_points\_earned\_added**])  
  
 *email\_data*[**:actions**] = **self**.dashboard\_snapshots.*sum*(**:actions**)  
 *email\_data*[**:last\_week\_actions\_added**] = *last\_week\_snapshots*.*sum*(**:actions**)  
  
 *email\_data*[**:pretty\_actions**] = ***ActiveSupport***::***NumberHelper***.number\_to\_delimited(*email\_data*[**:actions**])  
 *email\_data*[**:pretty\_last\_week\_actions\_added**] = ***ActiveSupport***::***NumberHelper***.number\_to\_delimited(*email\_data*[**:last\_week\_actions\_added**])  
  
 *email\_data*[**:email\_subscribers**] = **self**.dashboard\_snapshots.*sum*(**:email\_subscribers**)  
 *email\_data*[**:last\_week\_email\_subscribers\_added**] = *last\_week\_snapshots*.*sum*(**:email\_subscribers**)  
  
 *email\_data*[**:pretty\_email\_subscribers**] = ***ActiveSupport***::***NumberHelper***.number\_to\_delimited(*email\_data*[**:email\_subscribers**])  
 *email\_data*[**:pretty\_last\_week\_email\_subscribers\_added**] = ***ActiveSupport***::***NumberHelper***.number\_to\_delimited(*email\_data*[**:last\_week\_email\_subscribers\_added**])  
  
 *content\_body* = ***ActionController***::***Base***.new.*render\_to\_string*(**"emails/digests/weekly\_digest.html.erb"**, **:layout** *=>* **false**, **:locals** *=> email\_data*)  
 *content* = ***ActionController***::***Base***.new.*render\_to\_string*(**:text** *=> content\_body*, **:layout** *=>* **"weekly\_digest"**, **:locals** *=> email\_data*)  
  
 *wd* = **self**.email\_campaigns.*where*(**:type** *=>* **"WeeklyDigest"**, **:active** *=>* **true**).take  
 *ece* = ***EmailCampaignEmail***.create! **:merchant\_id** *=>* **self**.id, **:email\_campaign\_id** *=> wd*.id  
  
 *mail* = ***SendGrid***::***Mail***.new  
  
 *tracking\_settings* = ***SendGrid***::***TrackingSettings***.new  
 *tracking\_settings*.subscription\_tracking = ***SendGrid***::***SubscriptionTracking***.new(**enable**: **false**)  
 *mail*.tracking\_settings = *tracking\_settings  
  
 mail*.from = ***SendGrid***::***Email***.new(**email**: **"team@swellrewards.com"**, **name**: **"Swell Team"**)  
 *mail*.subject = **"[Swell Weekly Digest] Prepared For** #{**self**.company\_name} **On** #{*email\_data*[**:current\_date**]}**"** *mail*.contents = ***SendGrid***::***Content***.new(**type**: **'text/html'**, **value**: *content*)  
 *personalization* = ***SendGrid***::***Personalization***.new  
 *personalization*.to = ***SendGrid***::***Email***.new(**email**: **self**.contact\_email)  
 *mail*.personalizations = *personalization  
 personalization*.custom\_args = ***SendGrid***::***CustomArg***.new(**key**: **'ece\_id'**, **value**: *ece*.id.to\_s)  
 *sg* = ***SendGrid***::***API***.new(**api\_key**: ***ENV***[**'SENDGRID\_API\_KEY'**])  
 *response* = *sg*.client.mail.\_(**'send'**).post(**request\_body**: *mail*.to\_json)  
 **end  
 end  
  
 def** *coupons* redemption\_options.*where*(**:type** *=>* **"Coupon"**)  
 **end  
  
 def** *store\_gift\_cards* redemption\_options.*where*(**:type** *=>* **"StoreGiftCard"**)  
 **end  
  
 def** *has\_store?* has\_store  
 **end  
  
 def** *uninstall\_analytics* **self**.has\_analytics\_installed = **false  
 self**.uninstalled\_analytics\_at = ***Time***.now  
 **self**.save  
 owner.uninstall\_analytics(**self**)  
 **end  
  
 def** *admin\_login\_url* ***ENV***[**"API\_HOST"**] + **"/oauth/login/**#{owner.authentication\_token}**/**#{owner.id}**/**#{owner.email}**/**#{id}**"  
 end  
  
 def** *profile\_page\_url* ***ENV***[**"API\_HOST"**] + **"/merchants/**#{id}**"  
 end  
  
 def** *pending\_perks\_count* perks.*where*(**:pending** *=>* **true**).count  
 **end  
  
 def** *active\_perks\_count* perks.*where*(**:pending** *=>* **false**, **:completed** *=>* **false**, **:expired** *=>* **false**, **:awarded** *=>* **false**).count  
 **end  
  
 def** *redeemed\_perks\_count* perks.*where*(**:completed** *=>* **true**).count  
 **end  
  
 def** *first\_order\_date* **if** purchases.count > 0  
 purchases.*order*(**:created\_at**).first.created\_at.to\_date  
 **else  
 nil  
 end  
 end  
  
 def** *update\_redemption\_option\_images* coupons.each **do** |*coupon*|  
 *coupon*.image = **self**.logo.url  
 *coupon*.save  
 **end** store\_gift\_cards.each **do** |*store\_gift\_card*|  
 *store\_gift\_card*.image = **self**.logo.url  
 *store\_gift\_card*.save  
 **end  
 end  
  
 def** *total\_revenue\_cents* purchases.*sum*(**:amount\_cents**)  
 **end  
  
 def** *thirty\_day\_revenue\_cents* purchases.*where*(**'created\_at >= ?'**, ***Date***.today - 30.days).*sum*(**:amount\_cents**)  
 **end  
  
 def** *uninstall\_rewards* **self**.has\_rewards\_installed = **false  
 self**.uninstalled\_rewards\_at = ***Time***.now  
  
 **if** platform.eql?(**"shopify"**) || platform.eql?(**"woocommerce"**)  
 **self**.has\_analytics\_installed = **false  
 self**.uninstalled\_analytics\_at = ***Time***.now  
 **end  
  
 self**.customers.*where*(**"points\_reminder\_job\_id is not null"**).find\_each **do** |*customer*|  
 *customer*.cancel\_points\_reminder\_email  
 **end** *current\_subscription* = **self**.subscriptions.*where*(**:current** *=>* **true**).take  
  
 **unless** *current\_subscription*.plan.name.eql?(**"BRONZE"**)  
 **self**.switch\_plan\_by\_name(**"BRONZE"**)  
 **end  
  
 self**.owner.add\_intercom\_tag(**"uninstalled"**)  
 **self**.owner.log\_intercom\_event(**"uninstalled"**)  
  
 **self**.is\_approved = **false  
 self**.program\_live = **false  
 self**.vip\_tiers\_enabled = **false  
 self**.save  
  
 **if *Utils***::***Features***.merchant\_has\_feature?(**'NEW\_VIP\_TIERS\_MERCHANTS'**, **self**.id)  
 **self**.vip\_tiers\_setting.update\_attribute(**:is\_enabled**, **false**)  
 **end** owner.uninstall\_rewards(**self**)  
 **end  
  
 def** *can\_show\_widget?* **self**.program\_live && **self**.widget\_enabled && **self**.consumer\_campaigns.count > 0  
 **end  
   
 def** *points\_expiration\_in\_use?* **self**.points\_expire\_after\_days.present? && **self**.points\_expire\_after\_days > 0  
 **end  
  
 def** *monitor\_profile* **if** !**self**.completed\_profile && **self**.profile\_completed?  
 **self**.completed\_profile = **true  
 self**.completed\_profile\_at = ***Time***.now  
 **self**.is\_approved = **true  
 end  
 end  
  
 def** *monitor\_points\_expiration* **if self**.points\_expire\_after\_days\_changed?  
 ***CustomerProcessing***::***ReschedulePointsExpirationJob***.perform\_async(**self**.id, ***Time***.now)  
 **end  
 end  
  
 def** *create\_vip\_tiers\_settings* **if *Utils***::***Features***.merchant\_has\_feature?(**'NEW\_VIP\_TIERS\_MERCHANTS'**, **self**.id)  
 **self**.vip\_tiers\_setting = ***CalendarVipTiersSetting***.create!(**merchant\_id**: **self**.id, **sync\_to\_platform**: !is\_bigcommerce?, **last\_enabled\_type**: ***CalendarVipTiersSetting***.to\_s)  
 **else** copy\_vip\_tiers\_settings  
 **end  
 end  
  
 def** *has\_lifetime\_vip\_tiers\_plan?* **self**.vip\_tiers\_window\_months.nil?  
 **end  
  
 def** *copy\_vip\_tiers\_settings* ***VipTiersSetting***.map\_old\_settings\_to\_new(**self**)  
 **end  
  
 def** *has\_tier\_settings\_changed?  
 tier\_settings* = [**:vip\_tiers\_window\_months**, **:vip\_tiers\_start\_date**,  
 **:delay\_purchase\_vip\_tier\_eligibility**, **:delay\_purchase\_vip\_tier\_eligibility\_days**,  
 **:vip\_tiers\_enabled**]  
 *tier\_settings*.any? { |*setting*| **self**.*send*(**"**#{*setting*}**\_changed?"**)}  
 **end  
  
 def** *monitor\_widget* **if self**.widget\_enabled\_changed?  
 **if self**.widget\_enabled  
 **self**.processors.each { |*p*| *p*.enable\_widget }  
 **else  
 self**.processors.each { |*p*| *p*.disable\_widget }  
 **end  
 end  
 end  
  
 def** *monitor\_merchant\_created* ***RedCross***.track(  
 **user\_id**: **self**.owner.id,  
 **event**: **'Swell Signup Completed'**,  
 **properties**: **self**.red\_cross\_custom\_attributes  
 )  
 **end  
  
 def** *monitor\_orders\_limit\_reach*(*orders\_count*)  
 *event\_properties* = {  
 **count\_orders**: *orders\_count* }  
 ***RedCross***.track(  
 **user\_id**: **self**.owner.id,  
 **event**: **'Order Limit Reached'**,  
 **properties**: **self**.red\_cross\_custom\_attributes.merge(*event\_properties*)  
 )  
 **end  
  
 def** *profile\_completed?* **self**.description.present? && **self**.website.present? && **self**.company\_name.present? && **self**.logo.url.present? && !**self**.logo.url.eql?(**"/assets/merchant.png"**) && merchant\_categories.count > 0  
 **end  
  
 def** *handle\_changed\_attrs* **if** company\_name\_changed?  
 **self**.slug = company\_name.parameterize  
 **end  
  
 self**.handle\_website\_changed **if** website\_changed?  
  
 **if** source\_changed? && source.eql?(**"affiliate"**) && **self**.stripe\_customer\_id.nil?  
 **self**.switch\_to\_stripe\_billing  
 **end  
  
 if self**.platform\_supports\_syncing\_fields\_with\_processor? && sync\_fields\_with\_processor\_changed? && sync\_fields\_with\_processor  
 **if self**.platform.eql?(**"shopify"**)  
 ***AccountProcessing***::***Shopify***::***SetupMetafieldsJob***.perform\_async(**self**.id)  
 **elsif self**.platform.eql?(**"woocommerce"**)  
 ***AccountProcessing***::***Woocommerce***::***SetupMetafieldsJob***.perform\_async(**self**.id)  
 **end  
 end  
  
 true  
 end  
   
 def** *platform\_supports\_syncing\_fields\_with\_processor?* **%w(shopify woocommerce)**.index(**self**.platform).present?  
 **end  
  
 def** *cleanup\_merchant\_record  
 o* = **self**.owner  
 *o*.email = *o*.email + **"\_deleted"** *o*.save  
  
 *sub* = **self**.current\_subscription  
 *sub*.current = **false** *sub*.save  
  
 *token* = **self**.oauth\_tokens.first  
 *processor* = **self**.processors.first  
  
 *token*.delete  
 *processor*.delete  
 **self**.delete  
 **end  
  
 def** *setup\_metafields  
 ecommerce\_processor* = **self**.ecommerce\_platform\_processor  
  
 **if** *ecommerce\_processor* **self**.customers.from(**"customers FORCE INDEX(index\_customers\_on\_merchant\_id\_and\_points\_earned)"**).find\_each **do** |*customer*|  
  
 **begin  
 self**.sync\_customer\_metafields(*customer*, *ecommerce\_processor*)  
 **rescue** sleep 5  
 **begin  
 self**.sync\_customer\_metafields(*customer*, *ecommerce\_processor*)  
 **rescue** sleep 5  
 **self**.sync\_customer\_metafields(*customer*, *ecommerce\_processor*)  
 **end  
 end  
 end  
 end  
 end  
  
 def** *sync\_customer\_metafields*(*customer*, *processor* = **nil**)  
 **unless** *processor  
 processor* = **self**.ecommerce\_platform\_processor  
 **end  
  
 if** *processor  
 processor*.update\_customer\_metafields(*customer*)  
 **end  
 end  
  
 def** *bitpay\_client* ***BitPay***::***SDK***::***Client***.new(**api\_uri**: **'https://bitpay.com'**, **pem**: **self**.bitpay\_pem, **debug**: **true**, **insecure**: **true**)  
 **end  
  
 def** *get\_pairing\_code* **self**.bitpay\_client.pair\_client()[0][**"pairingCode"**]  
 **end  
  
 def** *generate\_api\_key* loop **do** *token* = ***SecureRandom***.base64.tr(**'+/='**, **'Qrt'**)  
 **break** *token* **unless *Merchant***.exists?(**api\_key**: *token*)  
 **end  
 end  
  
 def** *active\_campaigns* **self**.campaigns.*where*(**:active** *=>* **true**).all  
 **end  
  
 def** *consumer\_campaigns* **self**.campaigns.*where*(**:active** *=>* **true**, **:hidden** *=>* **false**, **:show\_in\_rewards\_popup** *=>* **true**).*order*(**'display\_order desc'**).all  
 **end  
  
 def** *stripe\_customer* **unless self**.stripe\_customer\_id  
 *customer* = ***Stripe***::***Customer***.*create*(**:email** *=>* **self**.contact\_email, **:metadata** *=>* {**:customer\_key** *=>* **self**.id})  
 **self**.stripe\_customer\_id = *customer*.id  
 save  
 **end  
 *Stripe***::***Customer***.retrieve(**self**.stripe\_customer\_id)  
 **end  
  
 def** *order\_callback\_url* ***ENV***[**"API\_HOST"**] + **"/coinbase/order/**#{id}**/**#{guid}**"  
 end  
  
 def** *set\_guid* **self**.guid = ***SecureRandom***.urlsafe\_base64(16)  
 **self**.api\_key = **self**.generate\_api\_key  
 **self**.contact\_email = **self**.owner.email  
 **end  
  
 def** *clean\_company\_name* company\_name.gsub(**/**[^0-9a-zA-Z\-' ]**/i**, **''**)  
 **end  
  
 def** *convert\_company\_name\_to\_filename* **self**.clean\_company\_name.gsub(**" "**, **"-"**).gsub(**"'"**, **""**).downcase  
 **end  
  
 def** *sweet\_tooth\_migration\_bookmarklet* **"javascript:void function(){var e=\"**#{**self**.owner.email}**\",t=\"**#{**self**.owner.authentication\_token}**\",o=\"**#{**self**.owner.id}**\",n=\"**#{**self**.id}**\",s=null,a=\"\";a+=\"background-color: rgba(0,0,0,0.6);\",a+=\"position:fixed;top:0;right:0;bottom:0;left:0;z-index:980;\",modalOverlayIsVisibleCSS=\"display:block;z-index:1009;\";var i=document.createElement(\"div\");i.id=\"swellrewards-overlay\",i.style.cssText=a,document.body.appendChild(i);var c=\"\";c+=\"z-index:2147483647;\",c+=\"position:fixed; top:50%; left:50%; right:auto; bottom: auto;\",c+=\"max-height: calc(100% - 100px); transform: translate(-50%, -50%);\",c+=\"width:90%;max-width:450px;\",c+=\"background: white;text-align:center;padding:15px;\";var r=document.createElement(\"div\");r.id=\"swellrewards-modal\",r.style.cssText=c;var u=\"\";u+=\"<h3>Sweet Tooth to Swell Migration Tool</h3>\",u+=\"<div id='swell-import-status' style='font-size:16px;'>\",u+=\"<img id='swell-status-image' src='https://upload.wikimedia.org/wikipedia/commons/d/de/Ajax-loader.gif'/><br/>\",u+=\"<span id='swell-imported-customers-count'>0</span> customers imported<br/>\",u+=\"<span id='swell-imported-coupons-count'>0</span> coupons imported<br/>\",u+=\"<span id='swell-imported-codes-count'>0</span> coupon codes imported<br/>\",u+=\"</div>\",r.innerHTML=u,document.body.appendChild(r);var d=0,p=0,l=0,m=1,f=!0,h=function(e){$(\"#swell-imported-customers-count\").html(e)},w=function(e){$(\"#swell-imported-coupons-count\").html(e)},g=function(e){$(\"#swell-imported-codes-count\").html(e)},x=function(a){var i=[];a.forEach(function(e){null==s&&(s=e.account\_id),i.push({email:e.email,first\_name:e.first\_name,last\_name:e.last\_name,points\_balance:e.points\_balance})}),d+=i.length,h(d),0==a.length?\_():$.ajax({type:\"POST\",url:\"https://app.swellrewards.com/api/v1/import-customers\",dataType:\"json\",contentType:\"application/json; charset=utf-8\",data:JSON.stringify({customers:i}),beforeSend:function(s){s.setRequestHeader(\"x-user-token\",t),s.setRequestHeader(\"x-user-email\",e),s.setRequestHeader(\"x-user-id\",o),s.setRequestHeader(\"x-merchant-id\",n)},success:function(){m+=1,f=!0},error:function(){m+=1,f=!0}})},v=function(){f&&(f=!1,$.getJSON(\"https://manage.sweettoothrewards.com/api/v1/customers?page=\"+m,function(e){x(e.items)}))},\_=function(){clearInterval(R),m=1,f=!1,s&&H()},y=function(s,a,i,c){0==i.length?(f=!1,m=1,clearInterval(O),c(!0)):(codes=i.map(function(e){return e.coupon\_code}).join(\",\"),l+=codes.split(\",\").length,g(l),$.ajax({type:\"POST\",url:\"https://app.swellrewards.com/api/v1/redemption\_codes\",dataType:\"json\",contentType:\"application/json; charset=utf-8\",data:JSON.stringify({codes:codes,redemption\_option\_id:a,merchant\_id:n}),beforeSend:function(s){s.setRequestHeader(\"x-user-token\",t),s.setRequestHeader(\"x-user-email\",e),s.setRequestHeader(\"x-user-id\",o),s.setRequestHeader(\"x-merchant-id\",n)},success:function(){m+=1,f=!0},error:function(){c(!1)}}))},b=function(s,a){var i=s.id;delete s.id,$.ajax({type:\"POST\",url:\"https://app.swellrewards.com/api/v1/coupons\",dataType:\"json\",contentType:\"application/json; charset=utf-8\",data:JSON.stringify(s),beforeSend:function(s){s.setRequestHeader(\"x-user-token\",t),s.setRequestHeader(\"x-user-email\",e),s.setRequestHeader(\"x-user-id\",o),s.setRequestHeader(\"x-merchant-id\",n)},success:function(e,t,o){m=1,f=!0,O=setInterval(function(){j(i,e.id,a)},500)},error:function(){a(!1)}})},S=function(e,t){e.length>0?(p+=1,w(p),nextCoupon=e.pop(),b(nextCoupon,function(o){o?S(e,t):t(!1)})):t(!0)},T=function(e){couponsToProcess=[],e.forEach(function(e){var t={type:\"Coupon\",single\_use:!0,id:e.coupon.id},o=!0;\"fixed\"==e.coupon.discount\_type?(t.discount\_type=\"fixed\_amount\",t.discount\_amount=e.coupon.value):\"percantage\"==e.coupon.discount\_type?(t.discount\_type=\"percentage\",t.discount\_percentage=e.coupon.value,o=!1):o=!1,\"fixed\_points\"==e.points\_exchange.type?t.amount=e.points\_exchange.points\_cost:o=!1,o&&couponsToProcess.push(t)}),S(couponsToProcess,function(e){q(e)})},H=function(){$.getJSON(\"https://manage.sweettoothrewards.com/api/v1/points\_products?account\_id=\"+s,function(e){T(e.items)})},j=function(e,t,o){f&&(f=!1,$.getJSON(\"https://manage.sweettoothrewards.com/api/v1/coupon\_codes?available\_only=true&coupon\_id=\"+e+\"&page=\"+m,function(n){y(e,t,n.items,o)}).error(function(){f=!1,m=1,clearInterval(O),o(!0)}))},q=function(e){$(\"#swell-status-image\").attr(\"src\",\"https://app.swellrewards.com/assets/logos/thumbs-up-no-border.png\"),$(\"#swell-status-image\").css(\"height\",\"100px\")},R=setInterval(v,500),O=null}();"  
 end  
  
 def self**.*newsletter\_list* ***User***.*where*(**"account\_type = 'merchant' and email not like '%shopify%' and email not like '%ecwid%' and email not like '%weebly%' and email not like '%mailinator%' and ignore\_in\_reports=false"**).find\_each **do** |*user*|  
 puts *user*.email + **","** + *user*.merchants.take.platform + **","** + *user*.merchants.take.clean\_company\_name + **", $"** + (*user*.merchants.take.thirty\_day\_revenue\_cents / 100.0).to\_s  
 **end  
 end  
  
 def self**.*cohort\_snapshot*(*start\_date* = **"2016-01-01"**, *end\_date* = ***Date***.today.to\_s, *cohort\_size* = 1.month)  
 *start\_date* = *start\_date*.to\_date  
 *end\_date* = *end\_date*.to\_date  
  
 *weekly\_data* = []  
  
 **while** *start\_date* < *end\_date* **do** *data* = ***Hash***.new  
 *data*[**:week\_of**] = *start\_date*.to\_s  
  
 *cohort\_merchants* = ***Merchant***.*where*(**"date(created\_at) >= ? and date(created\_at) < ?"**, *start\_date*, *start\_date* + *cohort\_size*)  
  
 *all\_ids* = *cohort\_merchants*.*pluck*(**:id**)  
 *above\_10\_below\_100\_ids* = *cohort\_merchants*.*where*(**"install\_order\_volume >= 10 and install\_order\_volume < 100"**).*pluck*(**:id**)  
 *above\_100\_ids* = *cohort\_merchants*.*where*(**"install\_order\_volume >= 100"**).*pluck*(**:id**)  
  
 *# installed absolute  
 data*[**:all\_installs**] = *all\_ids*.count  
 *data*[**:above\_10\_below\_100\_installs**] = *above\_10\_below\_100\_ids*.count  
 *data*[**:above\_100\_installs**] = *above\_100\_ids*.count  
  
 *# uninstalled absolute  
 data*[**:all\_uninstalls**] = ***Merchant***.*where*(**"has\_rewards\_installed = false and id in (?)"**, *all\_ids*).count  
 *data*[**:above\_10\_below\_100\_uninstalls**] = ***Merchant***.*where*(**"has\_rewards\_installed = false and id in (?)"**, *above\_10\_below\_100\_ids*).count  
 *data*[**:above\_100\_uninstalls**] = ***Merchant***.*where*(**"has\_rewards\_installed = false and id in (?)"**, *above\_100\_ids*).count  
  
 *# # campaign creation absolute  
 # data[****:all\_campaign\_created****] = Campaign.joins(****:merchant****).where(****"campaigns.created\_at > addtime(merchants.created\_at, '0 00:01:00.00') and merchant\_id in (?)"****, all\_ids).pluck(****:merchant\_id****).uniq.count  
 # data[****:above\_10\_below\_100\_campaign\_created****] = Campaign.joins(****:merchant****).where(****"campaigns.created\_at > addtime(merchants.created\_at, '0 00:01:00.00') and merchant\_id in (?)"****, above\_10\_below\_100\_ids).pluck(****:merchant\_id****).uniq.count  
 # data[****:above\_100\_campaign\_created****] = Campaign.joins(****:merchant****).where(****"campaigns.created\_at > addtime(merchants.created\_at, '0 00:01:00.00') and merchant\_id in (?)"****, above\_100\_ids).pluck(****:merchant\_id****).uniq.count  
  
  
 # ever converted absolute  
 data*[**:all\_premiums**] = ***Subscription***.*where*(**"plan\_id >= 2 and (current = true or was\_current = true) and merchant\_id in (?)"**, *all\_ids*).*pluck*(**:merchant\_id**).uniq.count  
 *data*[**:above\_10\_below\_100\_premiums**] = ***Subscription***.*where*(**"plan\_id >= 2 and (current = true or was\_current = true) and merchant\_id in (?)"**, *above\_10\_below\_100\_ids*).*pluck*(**:merchant\_id**).uniq.count  
 *data*[**:above\_100\_premiums**] = ***Subscription***.*where*(**"plan\_id >= 2 and (current = true or was\_current = true) and merchant\_id in (?)"**, *above\_100\_ids*).*pluck*(**:merchant\_id**).uniq.count  
  
 *# still premium absolute  
 data*[**:all\_still\_premiums**] = ***Subscription***.*where*(**"plan\_id >= 2 and (current = true) and merchant\_id in (?)"**, *all\_ids*).*pluck*(**:merchant\_id**).uniq.count  
 *data*[**:above\_10\_below\_100\_still\_premiums**] = ***Subscription***.*where*(**"plan\_id >= 2 and (current = true) and merchant\_id in (?)"**, *above\_10\_below\_100\_ids*).*pluck*(**:merchant\_id**).uniq.count  
 *data*[**:above\_100\_still\_premiums**] = ***Subscription***.*where*(**"plan\_id >= 2 and (current = true) and merchant\_id in (?)"**, *above\_100\_ids*).*pluck*(**:merchant\_id**).uniq.count  
  
 *# churn absolute  
 data*[**:all\_churn**] = *data*[**:all\_premiums**] - *data*[**:all\_still\_premiums**]  
 *data*[**:above\_10\_below\_100\_churn**] = *data*[**:above\_10\_below\_100\_premiums**] - *data*[**:above\_10\_below\_100\_still\_premiums**]  
 *data*[**:above\_100\_churn**] = *data*[**:above\_100\_premiums**] - *data*[**:above\_100\_still\_premiums**]  
  
  
 *# uninstallation rate  
 data*[**:all\_uninstall\_rate**] = ((*data*[**:all\_uninstalls**] / (*data*[**:all\_installs**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *data*[**:above\_10\_below\_100\_uninstall\_rate**] = ((*data*[**:above\_10\_below\_100\_uninstalls**] / (*data*[**:above\_10\_below\_100\_installs**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *data*[**:above\_100\_uninstall\_rate**] = ((*data*[**:above\_100\_uninstalls**] / (*data*[**:above\_100\_installs**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *# ever convert premium rate  
 data*[**:all\_premium\_rate**] = ((*data*[**:all\_premiums**] / (*data*[**:all\_installs**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *data*[**:above\_10\_below\_100\_premium\_rate**] = ((*data*[**:above\_10\_below\_100\_premiums**] / (*data*[**:above\_10\_below\_100\_installs**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *data*[**:above\_100\_premium\_rate**] = ((*data*[**:above\_100\_premiums**] / (*data*[**:above\_100\_installs**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *# still premium rate  
 data*[**:all\_still\_premium\_rate**] = ((*data*[**:all\_still\_premiums**] / (*data*[**:all\_premiums**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *data*[**:above\_10\_below\_100\_still\_premium\_rate**] = ((*data*[**:above\_10\_below\_100\_still\_premiums**] / (*data*[**:above\_10\_below\_100\_premiums**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *data*[**:above\_100\_still\_premium\_rate**] = ((*data*[**:above\_100\_still\_premiums**] / (*data*[**:above\_100\_premiums**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *# churn rate  
 data*[**:all\_churn\_rate**] = ((*data*[**:all\_churn**] / (*data*[**:all\_premiums**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *data*[**:above\_10\_below\_100\_churn\_rate**] = ((*data*[**:above\_10\_below\_100\_churn**] / (*data*[**:above\_10\_below\_100\_premiums**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *data*[**:above\_100\_churn\_rate**] = ((*data*[**:above\_100\_churn**] / (*data*[**:above\_100\_premiums**] \* 1.0)) \* 100.0).round(2).to\_s + **"%"** *rate\_vars* = [**:all\_uninstall\_rate**, **:above\_10\_below\_100\_uninstall\_rate**, **:above\_100\_uninstall\_rate**, **:all\_premium\_rate**, **:above\_10\_below\_100\_premium\_rate**, **:above\_100\_premium\_rate**, **:all\_still\_premium\_rate**, **:above\_10\_below\_100\_still\_premium\_rate**, **:above\_100\_still\_premium\_rate**, **:all\_churn\_rate**, **:above\_10\_below\_100\_churn\_rate**, **:above\_100\_churn\_rate**]  
 *rate\_vars*.each **do** |*rv*|  
 **if** *data*[*rv*].eql?(**"NaN%"**)  
 *data*[*rv*] = **"0%"  
 end  
 end** *start\_date* += *cohort\_size  
 weekly\_data* << *data* **end** *local\_data* = ***Hash***.new  
 *local\_data*[**:weekly\_data**] = *weekly\_data  
  
 content\_body* = ***ActionController***::***Base***.new.*render\_to\_string*(**"emails/digests/cohorts.html.erb"**, **:layout** *=>* **false**, **:locals** *=> local\_data*)  
 *content* = ***ActionController***::***Base***.new.*render\_to\_string*(**:text** *=> content\_body*, **:layout** *=>* **false**, **:locals** *=> local\_data*)  
  
 *mail* = ***SendGrid***::***Mail***.new  
 *mail*.from = ***SendGrid***::***Email***.new(**email**: **"james@swellrewards.com"**, **name**: **"Swell Stats"**)  
 *mail*.subject = **"[Swell Cohort Data] -** #{***Date***.today}**"** *mail*.contents = ***SendGrid***::***Content***.new(**type**: **'text/html'**, **value**: *content*)  
 *personalization* = ***SendGrid***::***Personalization***.new  
  
 [**"james"**,**"josh"**,**"dj"**,**"dzenita"**,**"nikhil"**,**"alison"**,**"melissa"**,**"mike"**,**"stewart"**,**"samuel"**].each **do** |*member*|   
 *personalization*.to = ***SendGrid***::***Email***.new(**email**: **"**#{*member*}**@swellrewards.com"**)  
 **end** *mail*.personalizations = *personalization  
 sg* = ***SendGrid***::***API***.new(**api\_key**: ***ENV***[**'SENDGRID\_API\_KEY'**])  
 *response* = *sg*.client.mail.\_(**'send'**).post(**request\_body**: *mail*.to\_json)  
  
 **end  
  
 def** *enable\_shopify\_webhook*(*topic*)  
 *processor* = **self**.processors.*where*(**:type** *=>* **"ShopifyProcessor"**).take  
  
 **if** *processor  
 processor*.enable\_webhook(*topic*)  
 **end  
 end  
  
 def** *apply\_shopify\_credit*(*amount\_cents*, *description* = **"Swell Application Credit"**)  
 **if** *amount\_cents* && *amount\_cents* > 0  
 *amount* = *amount\_cents* / 100.0  
 *processor* = **self**.processors.*where*(**:type** *=>* **"ShopifyProcessor"**).take  
 **if** *processor  
 processor*.create\_application\_credit(*description*, *amount*)  
 **end  
 end  
 end  
  
 def** *has\_vip\_tiers?* **self**.vip\_tiers.count > 0  
 **end  
  
 def** *trigger\_vip\_tiers\_calculation* ***Rails***.logger.info { { **message**: **'recalculate tiers btn was pressed'**, **merchant\_id**: **self**.id, **vip\_tiers\_job\_running**: **self**.vip\_tiers\_job\_running } }  
 **unless self**.vip\_tiers\_job\_running  
 ***Rails***.logger.info { { **message**: **'recalculate tiers starting to run'**, **merchant\_id**: **self**.id } }  
 **self**.vip\_tiers\_job\_running = **true  
 self**.save!  
 ***AccountProcessing***::***RecalculateTiersJob***.perform\_async(**self**.id)  
 **end  
 end  
  
 def** *recalculate\_vip\_tiers*(*customer\_ids*)  
 *customers\_relation* = ***Customer***.*where*(**id**: *customer\_ids*)  
 *total\_customers\_size* = *customers\_relation*.size  
 ***Rails***.logger.info { { **message**: **'starting to recalculate\_vip\_tiers'**, **total\_customers**: **"**#{*total\_customers\_size*}**"** } }  
 *already\_processed* = 0  
  
 *customer\_ids\_updated* = []  
 *customers\_relation*.*includes*(**:merchant**, **:customer\_vip\_tier** *=>* [**:vip\_tier** *=>* **:vip\_tiers\_threshold**]).find\_in\_batches(**batch\_size**: 10000) **do** |*customers\_batch*|  
 *customers\_batch*.each **do** |*customer*|  
 *tier\_changed* = *customer*.recalculate\_tier  
 *customer\_ids\_updated* << *customer*.id **if** *tier\_changed* **end** *already\_processed* += *customers\_batch*.size  
 ***Rails***.logger.info { { **message**: **'processing recalculate\_vip\_tiers'**, **processed**: **"**#{*already\_processed*}**"**, **total\_customers**: **"**#{*total\_customers\_size*}**"** } }  
 **end  
  
 *Rails***.logger.info{ { **message**: **'finished recalculate\_vip\_tiers'**, **processed**: **"**#{*already\_processed*}**"**, **total\_customers**: **"**#{*total\_customers\_size*}**"** } }  
 *customer\_ids\_updated* **end  
  
 def** *vip\_tiers\_window\_start\_date*(*window\_end\_at* = ***Time***.now)  
 **if self**.vip\_tiers\_window\_months.present?  
 **if self**.vip\_tiers\_start\_date.present?  
 *# beginning of window or start date, whichever is more recent* [*window\_end\_at* - **self**.vip\_tiers\_window\_months.months, **self**.vip\_tiers\_start\_date].max  
 **else** *# no start date, so it's the beginning of the window* (*window\_end\_at* - **self**.vip\_tiers\_window\_months.months)  
 **end  
 else** *# infinite window so it's either the start date or include ALL history* ***if*** *nil* **self**.vip\_tiers\_start\_date  
 **end  
 end  
  
 def** *vip\_tiers\_to\_hash  
 vip\_tiers\_hash* = ***Hash***.new  
 *vip\_tiers* = **self**.vip\_tiers.*order*(**"rank ASC"**)  
 *prev\_vip\_tier\_hash* = **nil** *vip\_tiers*.each **do** |*vip\_tier*|  
 *vip\_tiers\_hash*[*vip\_tier*.id] = ***OpenStruct***.new(**:vip\_tier** *=> vip\_tier*, **:redemption\_option\_name** *=> vip\_tier*.redemption\_option.present? ? *vip\_tier*.redemption\_option.name : **""**, **:next\_vip\_tier\_id** *=>* **nil**)  
 **if** *prev\_vip\_tier\_hash  
 prev\_vip\_tier\_hash*.next\_vip\_tier\_id = *vip\_tier*.id  
 **end** *prev\_vip\_tier\_hash* = *vip\_tiers\_hash*[*vip\_tier*.id]  
 **end** *vip\_tiers\_hash* **end  
  
 def** *third\_party\_plugin\_url  
 token* = **self**.oauth\_tokens.*where*(**:type** *=>* **"WoocommerceToken"**).*order*(**'updated\_at DESC'**).first  
 **if** *token  
 token*.plugin\_url  
 **else  
 ""  
 end  
 end  
  
 def** *lightspeed\_enabled?  
 oauth\_token* = **self**.oauth\_tokens.*where*(**type**: **"LightspeedToken"**)  
 *oauth\_token*.any? ? **true** : **false  
 end  
  
 def** *recharge\_api\_enabled?  
 oauth\_token* = **self**.oauth\_tokens.*where*(**type**: **"RechargeToken"**)  
 *oauth\_token*.any? ? **true** : **false  
 end  
  
 def** *setup\_campaigns*(*campaign\_details*)  
 **self**.ensure\_default\_fixed\_coupons  
 **if** *campaign\_details*.present?  
 *campaign\_details*.each **do** |*campaign*|  
 **begin  
 if** *campaign*[**"name"**] == **"ReferralCampaign"** *redemption\_option* = **nil** *referral\_redemption\_option* = **nil** *campaign*[**"referred\_customer\_min\_spend"**] = **"5"** *redemption\_amount* = *campaign*[**"discount\_selector"**] == **"coupon"** ? *campaign*[**"redemption\_option\_id"**].to\_i : **nil** *referral\_redemption\_amount* = *campaign*[**"referral\_discount\_selector"**] == **"coupon"** ? *campaign*[**"referral\_redemption\_option\_id"**].to\_i : **nil** *min\_spend\_cents* = ***Monetize***.parse(*campaign*[**"referred\_customer\_min\_spend"**], **self**.currency).cents  
  
 *reward\_type* = *campaign*[**"reward\_type\_discount\_selector"**]  
 **if** *redemption\_amount* **if** *reward\_type* == **"fixed\_amount"** *discount\_type* = **self**.platform\_fixed\_amount\_discount\_type  
 *redemption\_option* = ***Coupon***.find\_or\_create\_by(**:discount\_amount\_cents** *=> redemption\_amount*\*100, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **else** *discount\_type* = **self**.platform\_percentage\_discount\_type  
 *redemption\_option* = ***Coupon***.find\_or\_create\_by(**:discount\_percentage** *=> redemption\_amount*, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **end  
 end** *referral\_reward\_type* = *campaign*[**"referral\_reward\_type\_discount\_selector"**]  
 **if** *referral\_redemption\_amount* **if** *referral\_reward\_type* == **"fixed\_amount"** *discount\_type* = **self**.platform\_fixed\_amount\_discount\_type  
 *referral\_redemption\_option* = ***Coupon***.find\_or\_create\_by(**:discount\_amount\_cents** *=> referral\_redemption\_amount*\*100, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **else** *discount\_type* = **self**.platform\_percentage\_discount\_type  
 *referral\_redemption\_option* = ***Coupon***.find\_or\_create\_by(**:discount\_percentage** *=> referral\_redemption\_amount*, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **end  
 end** *redemption\_option\_id* = *redemption\_option*.present? ? *redemption\_option*.id : **nil** *referral\_redemption\_option\_id* = *referral\_redemption\_option*.present? ? *referral\_redemption\_option*.id : **nil** *referral\_campaign* = ***ReferralCampaign***.create\_default(**self**.id, *referral\_redemption\_option\_id*, *campaign*[**"reward\_points"**], *min\_spend\_cents*, *campaign*[**"referral\_reward\_points"**], *redemption\_option\_id*)  
 *referral\_campaign*.setup\_default\_campaign\_popup\_configurations  
  
 **elsif** *campaign*[**"name"**] == **"CreateAccountCampaign"** || *campaign*[**"name"**] == **"BirthdayCampaign"** *redemption\_option* = **nil** *campaign\_name* = *campaign*[**"name"**]  
 *redemption\_amount* = *campaign*[**"discount\_selector"**] == **"coupon"** ? *campaign*[**"redemption\_option\_id"**].to\_i : **nil** *reward\_type* = *campaign*[**"reward\_type\_discount\_selector"**]  
  
 **if** *redemption\_amount* **if** *reward\_type* == **"fixed\_amount"** *discount\_type* = **self**.platform\_fixed\_amount\_discount\_type  
 *redemption\_option* = ***Coupon***.find\_or\_create\_by(**:discount\_amount\_cents** *=> redemption\_amount*\*100, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **else** *discount\_type* = **self**.platform\_percentage\_discount\_type  
 *redemption\_option* = ***Coupon***.find\_or\_create\_by(**:discount\_percentage** *=> redemption\_amount*, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **end  
 end** *redemption\_option\_id* = *redemption\_option*.present? ? *redemption\_option*.id : **nil** *campaign\_name*.constantize.create\_default(**self**.id, *redemption\_option\_id*, *campaign*[**"reward\_points"**])  
  
 **elsif** *campaign*[**"name"**] == **"PointsForPurchasesCampaign"** *campaign\_name* = *campaign*[**"name"**]  
 *campaign\_name*.constantize.create\_default(**self**.id, *campaign*[**"reward\_points"**])  
  
 **elsif** *campaign*[**"name"**] == **"NewsletterSignupCampaign"** *redemption\_option* = **nil** *amount* = *campaign*[**"discount\_selector"**] == **"coupon"** ? *campaign*[**"redemption\_option\_id"**].to\_i : **nil** *reward\_type* = *campaign*[**"reward\_type\_discount\_selector"**]  
  
 **if** *amount* **if** *reward\_type* == **"fixed\_amount"** *discount\_type* = **self**.platform\_fixed\_amount\_discount\_type  
 *redemption\_option* = ***Coupon***.find\_or\_create\_by(**:discount\_amount\_cents** *=> amount*\*100, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **else** *discount\_type* = **self**.platform\_percentage\_discount\_type  
 *redemption\_option* = ***Coupon***.find\_or\_create\_by(**:discount\_percentage** *=> amount*, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **end  
 else** *reward\_points* = *campaign*[**"reward\_points"**]  
 **end** *redemption\_option\_id* = *redemption\_option*.present? ? *redemption\_option*.id : **nil** *newsletter\_signup\_campaign* = ***NewsletterSignupCampaign***.create\_default(**self**.id, *redemption\_option\_id*, *reward\_points*)  
 *newsletter\_signup\_campaign*.setup\_default\_campaign\_popup\_configurations  
 **self**.sync\_newsletter\_with\_platform = !**self**.current\_subscription.plan.name.eql?(**"BRONZE"**) ? *campaign*[**"sync\_newsletter\_with\_platform"**] : **false  
 self**.save  
  
 **else** *redemption\_option* = **nil** *campaign\_name* = *campaign*[**"name"**]  
 *campaign*[**"merchant\_id"**] = **self**.id  
 *campaign*[**"min\_spend\_cents"**] = *campaign*[**"min\_spend\_cents"**].present? ? ***Monetize***.parse(*campaign*[**"min\_spend\_cents"**], **self**.currency).cents : 0  
 *redemption\_amount* = *campaign*[**"discount\_selector"**] == **"coupon"** ? *campaign*[**"redemption\_option\_id"**].to\_i : **nil** *reward\_type* = *campaign*[**"reward\_type\_discount\_selector"**]  
  
 **if** *redemption\_amount* **if** *reward\_type* == **"fixed\_amount"** *discount\_type* = **self**.platform\_fixed\_amount\_discount\_type  
 *redemption\_option* = ***Coupon***.find\_or\_create\_by(**:discount\_amount\_cents** *=> redemption\_amount*\*100, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **else** *discount\_type* = **self**.platform\_percentage\_discount\_type  
 *redemption\_option* = ***Coupon***.find\_or\_create\_by(**:discount\_percentage** *=> redemption\_amount*, **:display\_in\_popup** *=>* **false**, **:single\_use** *=>* **true**, **:amount** *=>* 0, **:merchant\_id** *=>* **self**.id, **:discount\_type** *=> discount\_type*, **:deleted** *=>* 0)  
 **end  
 end** *campaign*.delete(**"name"**)  
 *campaign*.delete(**"supported\_plan"**)  
 *campaign*.delete(**"campaign\_class"**)  
 *campaign*.delete(**"discount\_selector"**)  
 *campaign*.delete(**"reward\_type\_discount\_selector"**)  
 *campaign*.delete(**"campaign\_index"**)  
 *campaign*.delete(**"sync\_newsletter\_with\_platform"**)  
  
 **if** *campaign\_name* == **"CustomActionCampaign"** *campaign*[**"action\_name"**] = *campaign*[**"title"**].downcase.gsub(**" "**,**"\_"**)  
 **end** *campaign*[**"username"**][0] = **"" if** *campaign*[**"username"**].present? && *campaign*[**"username"**][0] == **"@"** *campaign*[**"redemption\_option\_id"**] = *redemption\_option*.present? ? *redemption\_option*.id : **nil** *# campaign\_obj = eval(****"#{campaign\_name}.create(#{campaign})"****)  
 # campaign\_obj.setup\_email\_campaigns  
 campaign\_name*.constantize.create\_default\_onboarding(*campaign*)  
 **end  
 rescue *Exception*** *=> e* **end  
 end  
 end  
  
 unless** has\_referral\_campaign  
 **self**.setup\_default\_referral\_program  
 **end  
  
 unless** has\_news\_letter\_signup\_campaign  
 **self**.setup\_default\_newsletter\_signup\_campaign  
 **end  
  
 end  
  
 def** *setup\_popups*(*popup*)  
 **self**.setup\_default\_rewards\_popup  
 **end  
  
 def** *has\_referral\_campaign* **self**.campaigns.*where*(**:active** *=>* **true**, **:type** *=>***"ReferralCampaign"**).any?  
 **end  
  
 def** *has\_news\_letter\_signup\_campaign* **self**.campaigns.*where*(**:type** *=>* **"NewsletterSignupCampaign"**, **:active** *=>* **true**).any?  
 **end  
  
 def** *setup\_email\_campaigns*(*email\_notifications*)  
 *email\_campaigns\_info* = []  
 **if** *email\_notifications*.present?  
 *email\_notifications*.each **do** |*notification*|  
 *email\_campaigns\_info* << *notification*[**"name"**]  
 **end  
 self**.email\_campaigns.*where*(**type**: *email\_campaigns\_info*).update\_all(**active**: **true**)  
 **end  
 end  
  
 def** *paypal\_enabled?  
 oauth\_token* = **self**.oauth\_tokens.*where*(**type**: **"PaypalToken"**)  
 *oauth\_token*.any? ? **true** : **false  
 end  
  
 def** *active\_payout\_integration  
 token* = **self**.oauth\_tokens.*where*(**:type** *=>* **"PaypalToken"**).*order*(**'updated\_at DESC'**).first  
 *token*.present?  
 **end** *# for launch we are keeping this option hidden* **def** *show\_referral\_discount\_option  
 # supported\_platforms = [****"shopify"****,* ***"woocommerce"****,* ***"bigcommerce"****]  
 # supported\_platforms.include?(self.platform)* **false  
 end  
  
 def** *affiliate\_program\_enabled?* **self**.active\_payout\_integration  
 **end  
  
 def** *app\_install\_url* **case** platform  
 **when 'shopify'  
 return "**#{***ShopifyToken***.new.authorize\_url}**?shop=**#{**self**.third\_party\_id}**"  
 when 'bigcommerce'  
 return "**#{***BigcommerceToken***.new.authorize\_url}**&context=stores/**#{**self**.third\_party\_id}**"  
 when 'magento'  
 return "https://support.yotpo.com/en/article/setting-up-swell-on-magento-1"  
 when 'magento2'  
 return "https://support.yotpo.com/en/article/setting-up-swell-on-magento-2"  
 when 'woocommerce'  
 return "https://support.yotpo.com/en/article/setting-up-swell-on-woocommerce"  
 when 'commerce\_cloud'  
 return "https://support.yotpo.com/en/article/integrating-loyalty-referrals-with-salesforce-commerce-cloud"  
 when 'custom'  
 return "https://support.yotpo.com/en/article/implementing-swell-on-custom-or-generic-ecommerce-platforms"  
 end  
 end  
  
 def** *platform\_token\_type* **"**#{**self**.platform.camelize}**Token"  
 end  
  
 def** *platform\_token* **self**.oauth\_tokens.*where*(**:type** *=>* **self**.platform\_token\_type).take  
 **end  
   
 def** *platform\_processor\_type* **"**#{**self**.platform.camelize}**Processor"  
 end** *############################  
 #### CONVEINECE METHODS ####  
 ############################* **def** *is\_bigcommerce?* platform.eql?(**"bigcommerce"**)  
 **end  
  
 def** *is\_shopify?* platform.eql?(**"shopify"**)  
 **end  
  
 def** *red\_cross\_custom\_attributes  
 pos\_processors* = **%w(ShopifyPosProcessor LightspeedProcessor)** *owner* = **self**.owner  
 *package* = **self**.current\_subscription.plan.name **rescue ""** {  
 **merchant\_id**: **self**.id,  
 **email**: **self**.from\_email,  
 **name**: *owner*.name,  
 **first\_name**: *owner*.first\_name,  
 **last\_name**: *owner*.last\_name,  
 **has\_pos**: (**self**.processors.*pluck*(**:type**) & *pos\_processors*).any?,  
 **install\_order\_volume**: **self**.install\_order\_volume,  
 **thirty\_day\_order\_volume**: **self**.thirty\_day\_order\_volume,  
 **thirty\_day\_revenue\_cents**: **self**.thirty\_day\_revenue\_cents,  
 **platform**: **self**.platform,  
 **platform\_plan**: **self**.plan,  
 **package**: *package*,  
 **company\_name**: **self**.company\_name,  
 **website**: **self**.website  
 }  
 **end  
  
 def** *anonymize!*(*email\_to\_anonymize*)  
 ***Rails***.logger.info(**"starting to anonymize merchant customer. merchant\_id:** #{**self**.id}**, email:** #{*email\_to\_anonymize*}**"**)  
 *anonymous\_email* = ***Gdpr***.get\_anonymous\_email  
 *anonymous\_name* = ***Gdpr***::***ANONYMOUS\_USER\_NAME*** *anonymized\_ids* = {  
 **customer\_id**: **nil**,  
 **affiliate\_ids**: [],  
 **referral\_receipt\_ids**: []  
 }  
 *customer* = **self**.customers.find\_by\_email(*email\_to\_anonymize*)  
 **if** *customer*.present?  
 *customer*.anonymize!(*anonymous\_email*, *anonymous\_name*)  
 *anonymized\_ids*[**:customer\_id**] = *customer*.id  
 **end  
 self**.affiliates.*where*(**email**: *email\_to\_anonymize*).each **do** |*affiliate*|  
 *anonymized\_ids*[**:affiliate\_ids**] << *affiliate*.id  
 *affiliate*.anonymize!(*anonymous\_email*)  
 **end  
 self**.referral\_receipts.*where*(**email**: *email\_to\_anonymize*).each **do** |*referral\_receipt*|  
 *anonymized\_ids*[**:referral\_receipt\_ids**] << *referral\_receipt*.id  
 *referral\_receipt*.anonymize!(*anonymous\_email*)  
 **end  
 self**.user\_actions.*where*(**email**: *email\_to\_anonymize*).each **do** |*user\_action*|  
 *user\_action*.anonymize!(*anonymous\_email*)  
 **end  
 *Rails***.logger.info(**"finished anonymize merchant customer. merchant\_id:** #{**self**.id}**, email:** #{*email\_to\_anonymize*}**, anonymous\_email:** #{*anonymous\_email*}**, anonymous\_name:** #{*anonymous\_name*}**, anonymized\_ids:** #{*anonymized\_ids*}**"**)  
 **end  
  
 def** *platform\_supports\_add\_product\_to\_cart?  
 supported\_platforms* = **%w(magento magento2 commerce\_cloud bigcommerce)** *supported\_platforms*.include?(**self**.platform)  
 **end  
  
 def** *subunits\_per\_unit* ***Money***::***Currency***.new(**self**.currency).subunit\_to\_unit  
 **end  
  
 def** *using\_points\_based\_tiers?* **if self**.is\_vip\_tiers\_enabled?  
 **if *Utils***::***Features***.merchant\_has\_feature?(**'NEW\_VIP\_TIERS\_MERCHANTS'**, **self**.id.to\_s)  
 *tier\_requirement* = ***VipTiers***::***Requirements***::***RequirementsFactory***.*get*(**'PointsEarned'**)  
 **return self**.vip\_tiers.*includes*(**:vip\_tiers\_threshold**).any? { |*vip\_tier*| *vip\_tier*.vip\_tiers\_threshold.any? { |*threshold*| *tier\_requirement*.should\_consider\_requirement?(*threshold*) } }  
 **else  
 return self**.vip\_tiers.*sum*(**:points\_earned**) > 0  
 **end  
 end  
  
 false  
 end  
  
 def** *on\_shopify\_plus?* **self**.plan.eql?(**"plus\_partner\_sandbox"**) || **self**.plan.eql?(**"shopify\_plus"**)  
 **end  
  
 def** *can\_use\_public\_giftcards\_api?  
 allowed\_merchant\_ids* = ***ENV***.fetch(**'USE\_PUBLIC\_GIFTCARD\_API\_MERCHANT\_IDS'**,**''**).split(**","**)  
 *allowed\_merchant\_ids*.include?(**self**.id.to\_s) && **self**.on\_shopify\_plus?  
 **end  
  
 def** *is\_vip\_tiers\_enabled?* **if *Utils***::***Features***.merchant\_has\_feature?(**'NEW\_VIP\_TIERS\_MERCHANTS'**, **self**.id)  
 **return self**.vip\_tiers\_setting && **self**.vip\_tiers\_setting.is\_enabled  
 **end  
 self**.vip\_tiers\_enabled  
 **end  
  
 def** *should\_delay\_vip\_tiers\_purchases?* **return self**.vip\_tiers\_setting.delay\_purchase\_eligibility\_enabled **if *Utils***::***Features***.merchant\_has\_feature?(**'NEW\_VIP\_TIERS\_MERCHANTS'**, **self**.id)  
 **self**.delay\_purchase\_vip\_tier\_eligibility  
 **end  
  
 def** *get\_vip\_tiers\_purchases\_delay\_days* **return self**.vip\_tiers\_setting.delay\_purchase\_eligibility\_days **if *Utils***::***Features***.merchant\_has\_feature?(**'NEW\_VIP\_TIERS\_MERCHANTS'**, **self**.id)  
 **self**.delay\_purchase\_vip\_tier\_eligibility\_days  
 **end  
  
 def** *should\_sync\_vip\_tiers\_with\_platform?* **return self**.vip\_tiers\_setting.sync\_to\_platform **if *Utils***::***Features***.merchant\_has\_feature?(**'NEW\_VIP\_TIERS\_MERCHANTS'**, **self**.id)  
 **self**.sync\_tiers\_with\_platform  
 **end  
  
 def** *get\_vip\_tiers\_start\_date* **return self**.vip\_tiers\_setting.custom\_start\_date **if *Utils***::***Features***.merchant\_has\_feature?(**'NEW\_VIP\_TIERS\_MERCHANTS'**, **self**.id)  
 **self**.vip\_tiers\_start\_date  
 **end  
   
 def** *get\_vip\_tiers\_requirement\_attrs\_used* **self**.vip\_tiers.map { |*tier*| *tier*.requirement\_attrs\_used }.flatten.uniq  
 **end  
  
 def** *strip\_blocked\_domains* **if self**.referral\_blocked\_domains.present?  
 *urls* = **self**.referral\_blocked\_domains.delete(**' '**).split(**','**)  
 *hosts* = *urls*.map { |*url*| ***Utils***::***Url***.get\_url\_host(*url*) }  
 **self**.referral\_blocked\_domains = *hosts*.join(**','**)  
 **end  
 end  
  
 def** *calculate\_verified\_hmac*(*swell\_hmac\_expires\_at*)  
 *message* = {  
 **merchant\_id**: **self**.id,  
 **guid**: **self**.guid,  
 **api\_key**: **self**.api\_key,  
 **expires\_at**: *swell\_hmac\_expires\_at* }  
  
 *digest* = ***OpenSSL***::***Digest***.new(**'sha256'**)  
 *verified\_hmac* = ***OpenSSL***::***HMAC***.hexdigest(*digest*, ***ENV***.fetch(**'API\_V2\_SECRET'**, **''**), *message*.to\_s)  
 **end  
end**