IncentMe Web Design Notes:

\*On AD Click

1. Identify the ad clicked
2. Show details of the ad
   1. Show price expected to pay for item(s)
   2. Show details of item(s)
3. Save button for each item
4. OnClick()
   1. if logged-in & has card on file >> save offer
   2. if logged-in & doesn't have card >> ad credit card form >> save offer

\*create an object with values for each ad

*- need a function that will load store details on pageLoad();*

\*On Normal Display/View

1. Create an Ad Timer
2. Link Ad Timer to Point Value
   1. Create an Array of values for Algorithm
      1. Need Time remaining
      2. Need Ad-Point Spectrum Of Values (e.g. 0 to 100)
      3. Set Ad Remaining Spectrum Factor/Value (0 to 50)
      4. ~~Set Store/Merchant Rank (Hard Code)~~
      5. ~~Set discount value (Hard Code)~~
      6. ~~Set Number of Ad Offers (Hard Code)~~
      7. ~~Set a Base Ad Point Figure (i.e. 100 pts)~~
      8. ~~Set Mid-Point Value (i.e. 50% of Base Ad Point Figure)~~
      9. ~~Set Base Percentage (i.e. 20%)~~
      10. ~~Calculate Base Discount Rate~~
   2. Create Activity Occurrence
      1. Ad Ledger (Total of all combined ad offers)
      2. Time of calculated Updates
      3. Base Coefficient Values (y1, y2, x1, x2, etc.)
3. Link Point Value to Ad Availability

Function Details:

* function purpose
* state of function (working, complete, refactor, review)
* for which stakeholder (consumer, advertiser, both)
* category of function (ad-display, ad-algorithm )

ID-0: mParam:

* Object: used for defining “rank time” related measurement parameters
* (working) – update property names to define purpose more clearly
* Advertiser
* Ad-algorithm, rank-algorithm

ID-1: setRank:

* Object: used defining measurement configuration tables per rank level
* (working) – requires changes in configuration values
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-2: formatTimeIntoDate(endTime, createTime, nowTime) –

* Function: used for converting time into days, hours, minutes, seconds
* (review) -

ID-3: calculateTime –

* Calculate time left before an ad offer/campaign expires
* (working) – need to update the adEndTime variable to get from store object
  + (Update) – added for loop to get values for each object
  + (Update) – added adCreateTime property to “stores” Object
  + (Update) – added adEndTime property to “stores” Object
* (review) – possibly break up internal methods
* Advertiser
* Ad-display

ID-4:

ID-4: getAdPointValue –

* Calculates the “**BASE AD POINT”** value used for converting discount percentages
* (working) – need to associate a unique adBasePercentageDiscountAverage for store type [a]
* (working) – need to associate a unique adPointMidPercentage for store type [b]
* (working) – need to update store object to have storeType property & values for [a] & [b]
* Advertiser
* Ad-algorithm

ID-4: getXAdPointSpectrum –

* Generate discount percentage value for horizontal (x-axis) row; (used for discount percent value set by advertiser)
* (working) – need x-axis variable to be set to at “.0001” or less. << lower decimal values cause browser to freeze
* Advertiser
* Ad-algorithm

ID-5: getYAdPointSpectrum –

* Generate vertical (y-axis) column for ad-ledger; (used for the current/live discount percent value of all ad offers)
* (working) – need y-axis variable to be set to at “.0001” or less. << lower decimal values cause browser to freeze
* Advertiser
* Ad-algorithm

ID-6: getYPivotPointValues –

* Generates vertical (y-axis) values from a mid/pivot point; values are used as a coefficient for the calculated yAdPointSpectrum & advertisers set discount percentage
* (working) – need different y-Axis pivot points; change by store type
* Advertiser
* Ad-algorithm

ID-7: getBaseDr –

* Calculates the adjusted base discount rate from the pivot point for y-plus and y-minus.
* (working) – variable baseMidStartValue needs to be different for each store type; its associated by store type
* Advertiser
* Ad-algorithm

ID-8: combineDr –

* Combines baseDrPlus & baseDrMinus array into a single array
* (review) – possibly review to add into one function with function ID-7
* Advertiser
* Ad-algorithm

ID-9: getFinalAdLedgerTable –

* Generates & calculates a final x & y table for all possible values??
* (review for removal) – y-axis and x-axis are not multiplied by each other; generating a table outputs incorrect values. (leaving in case its use was overlooked)
* Advertiser
* Ad-algorithm

ID-10: getTotalAdOffers –

* Calculates the discount percentage total & ad offer total in circulation; returns the discount average percent. The percentage value returned is the active y-axis column value. The index value of the y-axis position is then used to get the active discount-rate coefficient.
* (working) – need to calculation to include/be specific to store Type
  + (update) – remove arguments; store object has been passed in as object. Arguments was resulting in NaN due to each property of stores object not being an integer.
* Advertiser
* Ad-algorithm

ID-11: getOriginatingDiscountPercent –

* Calculates the ad-point conversion rate for a single percentage point; stores discount percentage is multiplied by the discount-rate coefficient
* (working) – if changes are made to yAdPointSpectrum, yIncrement NEEDS to be same. Otherwise, problems will occur with using index values of other tables.
* (working) – can this function be used within each stores Object? Once a value is received, it processes and determines the discount convert rate
  + (refactor) – removed arguments; updated return statement; upconvert point values; upconvert of BaseDr is optional and will depend on how large y-axis and x-axis values are.
* Advertiser
* Ad-algorithm

ID-12: getStoreAdPointValue –

* Uses the stores discount percentage, multiplies it by the calculated/current discount convert rate. The returned value is the ad offers active point value and displayed within the ad offer(s).
* (working) – Insertion is hard-coded. Needs to change to be more flexible and work for all stores.
  + (Review) – Remove function; refactored into function ID-1
* Advertiser
* Ad-algorithm

ID-13: getXYAdActivity –

* Sets the Y axis values for multiple ad-activity tables; Used for further manipulating ad-point system.
* (review) – review for improvements
* Advertiser
* Ad-algorithm

ID-14: getAdActivityTable –

* Generates the X values and calculates the multiplier values for the ad-activity table; uses the x1 values as multiplier and ad-activity Y-values to result in modifier values.
* (working) – need tables that are created associated to rank levels
* Advertiser
* Ad-Algorithm

ID-15: adOccuranceFactor –

* Gets the active multiplier from the generated ad-activity table(s); Returns the modifier value that’s used to further adjust point value of offer.
* (review) – review for improvements
* Advertiser
* Ad-algorithm

ID-16: setMTableAdver –

* An object for holding the measurement conditions for each rank level. This is currently manually defined.
* (working) – work on formula that’s capable of modify values using the users previous activity as the base.
* Advertiser
* Rank-algorithm

ID-17: calculateObjectValues –

* Used as a function to gather data from multiple (4) objects; calls function ID-18 passing values as parameters
* (working) – requires significant rework, but works in its ability to get values and input
* (working) – correct to pass object values into arrays; objects are causing multiple passes resulting in output errors (if statements not directing data as intended)
* Advertiser
* Rank-algorithm

ID-18: calculateStoreValues –

* Collects rank measurement actions, determines whether measurement action satisfies minimum condition, adds rank points to advertiser object
* (working) – works in its ability to get values and input; requires complete rework
* (working) – too many nested for loops; passing objects is causing multiple output bugs (if statements not directing data as intended; one click causes all rank levels of a matched measurement event to execute)
* (working) – measurement param values currently assigned manually; Need to get correct values through loop or some other method
* Advertiser
* Rank-algorithm

ID-19: getObjectValuesFromSetTableAdver –

* A general function that’s used for getting values from **setMTableAdver** Object. Returns each level of the object as an indexed array.
* (working) – add variables as they are required from the requesting function(s)
* (working) – modify present variables to be general by level deep
  + (update) – completed; modified to output object into an array by level of depth
* General use get function

ID-20: getObjectValuesFromMParam –

* A general function used for getting values from **mParam** Object. Returns each level of the object as an indexed array.
* (working) – add variables as required from requesting function(s)
* General use get function

ID-21: getObjectValuesFromStores –

* A general function used for getting values from **Stores** Object. Returns each level of the object as an indexed array.
* (review) – in review for completion
* (review) – the array of the last object is returning over 4,700 records for an object containing only 12 stores?? Review for logic error within loop structure.
* General use get function

ID-22: getObjectValuesFromSetRank –

* A general function used for getting values from **SetRank** Object. Returns each level of the object as an indexed array.
* (working) – add variables as required from requesting function(s).
* General use get function

ID-23: storeloginStatus { TEMP OBJECT}

* An object containing Login state of **true** or **false**. Used for testing function ID-24.
* (working) – possibly integrate login state into stores object of store/advertiser.
* Object

ID-24: whichStoreLoggedIn(cv, index, arr) –

* Function: gets the store **PROPERY:KEY** name value pair from ID-23 Object. Returns the result to a calling function. Checks for which store is logged-in.
* Relies on ID-23 (object).
* (review) – review for improvements?
* (working) – *MUST BE ASSOCIATED WITH ID-23* Object; if changes occur in ID-23, make changes to this function.
* Advertiser
* Ad-algorithm, rank-algorithm, ad-display

ID-25: checkStoreRankLv –

* Function; gets the stores identifier (store\_1, store\_2, et.) property the identified store.
* Relies on function ID-24;
* (review) – possibly integrate into function ID-24. Instead function ID-24 would return the stores property; this would eliminate this function.
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-26: getLoggedInStoresRankLv –

* Function: gets the stores current numerical rank level value (e.g. 1, 2, 3, 4, 5, etc.) of the identified store.
* Relies on function ID-25;
* (review) – possibly integrate into function ID-24. Instead have function ID-24 return the store property; this would eliminate this function.
* Advertiser
* Rank-algorithm

ID-27: calcTotalPossibleRankPoints –

* Function: getting the **Logged-in** stores “current” rank level from function ID-26; this function calculates the total number of rank points possible for their current rank level.
* Relies on Object ID-22 [internal function associated with the identified rank level].
* Relies on Object ID-19 [internal function associated with the identified rank level].
* Relies on Function ID-26;
* (working) – need to rework for a modular function; too many hard-code requirements to get values. Possibly change name of calling function; improving the IF statement Logic.
* Advertiser
* Rank-algorithm

ID-28: genAdvertiserRankLevelTable –

* Function: this function calculates the ***point spread*** for EACH level of the Current rank level of the identified store/advertiser.
* Add: stores Object { rankLv\_Inner } – 05/20/16
* Relies on Function ID-27;
* (working) – make the function modular; able to process different rank levels.
* (working) – Need to change inner incremental conditions within an Object relevant to the rank level. (Rank Level 1, 2, 3 has their own set of numStart, numPos, NumNeg etc. values).
* Advertiser
* Rank-algorithm

ID-29: genRankLevelPointThresholdValues –

* Function: this function calculates the ***point threshold values*** for each internal level of the current rank. The values returned are indicative of the level (i.e. rLevelPointSpread[0] == current rank level 1 of 40 etc.)
* Relies on Function ID-28;
* (working) – Need complete rework; redo logic – get the loop to work correctly
* (working) – Change the name of arr variable after rework
* Advertiser
* Rank-algorithm

ID-30: updateRankAdvertiser –

* Function: updates value of the stores RankLv\_Inner property to the current inner rank level.
* Relies on Function ID-29
* Relies on Function action(s) of ID-18 & ID-17
* (review) – working. Review for logic improvements. Possibly create a CLASS with methods for all rank-level related functions.
* Advertiser
* Rank-algorithm

ID-31: objectForAdPointAddition –

* Object: contains the configuration settings for Rank ADD-On Points. Used by function ID-32 and its internal methods/functions.
* Relies on values set within ID-3
* Relies on values settings from ID-0 & ID-00
* (working) – requires review for improvements as Rank Levels are added.
* ***!important >>*** Review configuration values >> possibly increase }
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-32: getAdPointAdditionValues –

* Function: Compile add-on point values/table for the identified rank level. Adds the value to base discount convert rate specific to the individual advertiser/store.
* (working) –

ID-33: compileY1Top –

* Function: Generates and compiles the top divider portion of Y-Axis values.
* Relies on Object ID-31 configuration values
* Method of function ID-32

ID-34: compileY1Bottom –

* Function: Generates and compiles the bottom divider portion of X-Axis values
* Relies on Object ID-31 configuration values
* Method of function ID-32

ID-35: compileYTopLeftAddOnTable –

* Function: Calculates and compiles the xy top-left segment of the add-on point table. The resulting array values is later reversed.
* Relies on Object ID-31 configuration values
* Method of function ID-32
* (working) – improve logic | move into its own function
* (working) – improve rank level if statement | code is heavy and needs a more modular technique.
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-36: compileYTopRightAddOnTable –

* Function: Calculates and compiles the xy top-right segment of the add-on point table. The resulting array values is later reversed.
* Relies on Object ID-31 configuration values
* Method of function ID-32
* (working) – improve logic | move into its own function
* (working) – improve rank level if statement | code is heavy and needs a more modular technique.
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-37: compileYBottomLeftAddOnTable –

* Function: Calculates and compiles the xy bottom-left segment of the add-on point table. The resulting array values is later reversed.
* Relies on Object ID-31 configuration values
* Method of function ID-32
* (working) – improve logic | move into its own function
* (working) – improve rank level if statement | code is heavy and needs a more modular technique.
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-38: compileYBottomRightAddOnTable –

* Function: Calculates and compiles the xy bottom-right segment of the add-on point table. The resulting array values is NOT reversed
* Relies on Object ID-31 configuration values
* Method of function ID-32
* (working) – improve logic | move into its own function
* (working) – improve rank level if statement | code is heavy and needs a more modular technique.
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-39: compileXMeasureSegments – { # This has been added above ID 33 }

* Function: Calculates the x-Axis Time measurement header table. Used for locating the current time position of the rank period so that Y-axis value can be located within completed xy-addOn table.
* Relies on Object ID-31 configuration values
* Method of Function ID-32
* Advertiser
* Ad-algorithm

ID-40: getYAxisY1 –

* Function: generates a new Y-Axis Vertical table to be used for sections Y2 as starting values to continue generation of adjustment DRC table
* Method of Function ID-32
* Advertiser
* Ad-algorithm

ID-41: getYAxisY3 –

* Function: generates a new Y-Axis Vertical table to be used for sections Y4 as starting values to continue generation of adjustment DRC table
* Method of Function ID-32
* Advertiser
* Ad-algorithm

ID-42: getStoresObj –

* Function: Get the **Stores Object** into an indexed array format
* General use

ID-43: getStoresProp –

* Function: Get the **Stores Property** into an indexed array format
* General use

ID-44: getSetRankProp –

* Function: Get the **SetRank Object** into an indexed array format
* General use

ID-45: getSetRankProp –

* Function: Get the **SetRank Property** into an indexed array format
* General use

ID-46: getMParamObj –

* Function: Get the **mParam Object** into an indexed array format
* General use

ID-47: getMParamProp –

* Function: Get the **mParam Property** into an indexed array format
* General use

ID-48: loadDRCTable(n) – { (n) accepts an integer; identifies the Rank Level }

* Function: Invokes function ID-32; Returns the correct table to Global Variable DRC.
* Relies on Function ID-32
* (working) – global variable outside of function to prevent generation of table with each invocation of ID-49
* Advertiser
* Rank-algorithm, Ad-algorithm

ID-49: updateAdjDCR(storeNumber) – { (storeNumber) accepts integer; identifies Store }

* Function: Identifies the store; gets the stores rank level; determines the time remaining in stores rank period, returns the adjusted Discount Convert Rate to add to the Base Discount convert rate of the store
* Relies on AT LEAST ONE discount convert table generated for the rank
* Relies
* (working) – individual the function for a single advertiser; currently updates every store for demonstrative purposes
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-50: storesObj –

* Variable: Invokes function ID-42; Returns Stores Object as array
* Advertiser
* General use

ID-51: setRankObj –

* Variable: Invokes function ID-44; Returns Stores Object as array
* Advertiser
* General use

ID-52: mParamObj –

* Variable: Invokes function ID-46; Returns Stores Object as array
* Advertiser
* General use

ID-53: rankLvl –

* Variable: hold the identified stores Rank Level
* Advertiser
* General use

ID-54: activeAdjDRC –

* Variable: this is for the calculated DCR; Value is returned to Stores object
* Advertiser
* General use

ID-55: xyAdjDRCTable –

* Variable: gets the global variable drcTable; returns the DCR table for the stores identified Rank Level
* Advertiser
* General use

ID-56: xTimeMSegmentActive –

* Variable: stores present time segment out of the rank levels maximum duration
* Advertiser
* General use

ID-57: index –

* Variable: used for using to index array objects; storeNumber results in errors of undefined
* Advertiser
* General use

ID-58: getStoreData(storeNumber) – { accepts integer to identify store }

* Function: Determines the store and its current Rank Level; Returns the rank level
* Method of ID-49
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-59: getRankPeriodDuration (mp, rankLvl) – { accepts ID-52, ID-53 }

* Function: Calculates the identified rank periods full duration in minutes
* Method of ID-49
* (working) – relocate function outside ID-49
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-60: getXTimeMSegment(so, sr, mp, stNum) – { Accepts ID-50, 51, 53, & storeNumber }

* Function: Calculates the time segment position along X-Axis; Returns the segment number
* Method of ID-42
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-61: getAdjDRC(so, storeNum, actTimePt, adjTableVals, mp) – { accepts ID-50, storeNumber, 56, 55, 52}

* Function: Transverses XY DCR table; Returns the adjusted Discount Convert Rate
* Method of ID-42
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-62: changeStoreDRC(stNum) – { accepts storeNumber }

* Function: Updates the each property adjDcr with variable ID-54
* Method of ID-42
* Advertiser
* Ad-algorithm, Rank-algorithm

ID-Last: CompileAdList –

* Insert descriptive text about the store into ad offers.
* (working) – needs non-manual process implemented
  + (Update) – add for loop to get values for Each Stores Object
  + (Update) – MOVED function to under ID-12 Position. Other functions weren’t executing causing points not to display within ads. (i.e. y-axis and x-axis tables where undefined).
  + (Update) – Add Point updating feature to ad offer from object values
* Advertiser
* Ad-display

Function Details:

* function purpose
* additions to current objects/functions & date of add.
* list of other function/objects it relies on by ID tag.
* state of function (working, complete, refactor, review)
* for which stakeholder (consumer, advertiser, both)
* category of function (ad-display, ad-algorithm, rank-algorithm)

\*possibly create a storeType object? Inherit the values by stores associated type.

FUNCTION & OBJECT DESCRIPTIONS

Objectives:

* Demonstrate offer displaying points in an ad setting
* Demonstrate offer displaying point and price in an e-commerce setting
* Demonstrate conversion of discount percent of an offer into a point value
  + Change the discount percent in floating point format
    - apply to individual offers
    - apply to all offers
    - show the applied discount percent within the offer
    - show the discount conversion rate within the offer
    - show the offer quantity that remains
* Demonstrate base convert rate difference amongst similar advertisers
* Demonstrate effects that an ad/offer ledger has on discount convert rates and point conversions
* Demonstrate effects of rank measurement having on point conversions
* Demonstrate measurement effects for offer performance
* Demonstrate reward distribution grids
* Demonstrate individualized stakeholder pods
* Demonstrate individualized reward distribution grids to stakeholder pods
* Demonstrate monetary distribution of collected discount dollars
* Demonstrate group linking of pods to reward grids upon redemption

Measurement types:

* Rank measurement
  + rank measurements grade the actions and performance of users. When adequate points are acquired, a rank level increase in awarded. Each “Rank-Level” has a subset of inner-levels. For example, Rank-level 1 may contain 40 inner-subset levels.
  + It’s when after a rank period duration ends, the inner-subset level achieved by the user will determine the rank attained for the following rank duration period.
* Offer qty/performance measurement.
  + qty/performance measures the market demand of an offer. Whereas, the longer the offer is available, the less valuable it is and thus worth less points. This is measured by the qty that remains over time. Live point values are affected by this measurement.
* Ad-ledger modifier
  + measures the total qty and combined value of all offers of specific categories. The combined discount percent is then used to modify the “base discount convert rate” for that period. This affects any offer/advertiser creating an offer at that time.
  + For example, the “base discount convert rate” may be 2.5 points per 1% of discount. But with a modifier of this measurement applied, that rate may increase to 7.2 points per 1% of discount. Whereby this modifier can be either benefit or disadvantage, less discount or greater discount respectfully.

ID-00: rankTimeMeasurementSettingsForAdvertiserObject

* Object contains “TIME” measurement parameters for “every Rank-Level”. The time settings are used for rank time metrics.
* In example, every offer that’s created by an advertiser/store is given a “Time Limit”. The time limit in which the offer is displayed/available is determined by the advertisers/stores current rank level at the time the offer was created.
* In addition to, the rank level of the advertiser/store is configured to each have different “time measurement occurrences” and “measurement time”. In a case when an advertiser/store increases their rank-level, that duration is extended and/or spread over a longer duration of time.
* The advantage being the offer is awarded “more time” to satisfy or exceed the rank measurement minimums for each period. This further granting each advertiser/store the benefit of achieving higher levels of rank and the benefits associated with that level.
  + rankMTime: This is the duration of a rank levels measurement period.
  + rankMOccurences: This is how often rank measurements are measured in minutes.
  + actionValue: This is the amount of points awarded for each measurement action that’s satisfied.
  + minutes: time duration of an hour
  + hours: hours in one full day
  + setRLv: the rank level in which the settings apply to
  + adEndTimeForOffers: This is the maximum duration of time - in “days” - an offer created by an advertiser is given from the time it became available.

ID-01: rankXyTableConfigSettingsForAdvertiserObject

* Object contains “AD PERFORMANCE TABLE” configuration parameters for “every Rank-Level”. It provides x-axis and y-axis step values for each measurement time segment. The time segment differs for each rank level. This object also contains offer-issuance limitations – restricting the amount of offers available in a given rank duration.
* For example, at rank-level 1, the adLimit is 50. This means an advertiser/store is only allowed to issue 50 offers for that rank-levels measurement duration, 5 days for example.
  + maxScale: maximum Y-axis scale range (is able to exceed 1 to provide Y-axis elasticity).
  + y1: is negative step value from y-axis pivot point
  + y2: is positive step value from y-axis pivot point
  + x1: is x-axis step value from y-axis sidebar/scale
  + xTimeScale: is the time measurement scale for ad acquisition. Number of times – in hours – when an ad acquisition / performance / QTY remain snapshot is taken.
  + adLimit: maximum number of offers an advertiser/store can create per measurement period.
  + estAdRedemptionAvg: used as a y-axis pivot point and acts as a baseline performance expectation/minimum threshold for offers created by advertiser/store of the level.
  + mTimeSegments: number of measurement segments for “Rank-Level”.
    - This needs to be moved into Object ID-00.

ID-02: rankMeasurementSettingsForAdvertiserObject

* Object contains “MEASURED RANK PARAMETERS” for each rank level. This means when an advertiser/store exceeds the parameter point value, a rank point is earned.
* For example, for clicks, a click occurs for a click count of 1. The satisfying condition for clicks action for segment one 0.21; exceeding the minimum and awarding one rank point for that measurement segment and any other until action parameter exceeds click count of 1. If no other clicks occurred, measurement segment #4 would be the last segment to award a rank point because each measurement segment increases by the starting i.e. (0.21 x 4 = 0.84 segment 4, 0.21 x 5 = 1.05 segment 5).
* Also, rank points are only awarded for the current segment and not those that have passed. For example, getting 2 clicks in segment #6 does not grant the 1 rank point missed in segment #5. Measurement base parameters requirement are multiplied by its segment number.
  + lv1: indicates the rank level
    - shopper activity: identifies the measurement category
      * clicks: clicks
      * views: views
      * reviews: reviews
      * payablePod: number of payable pods earned by users
      * notPayablePods: number of pods resulting in not-payable by users
      * podMax: number of users with maximized pods
      * rewardGridVariationLow: number of low level reward grid links
        + low1: will be represented by a first point threshold
        + low2: will be represented by a second point threshold
        + low3: will be represented by a third point threshold
      * rewardGridVariationMid: number of mid-level reward grid links
        + mid1: will be represented by a first point threshold
        + mid2: will be represented by a second point threshold
        + mid3: will be represented by a third point threshold
      * rewardGridVariationHigh: number of high-level reward grids
        + high1: will be represented by a first point threshold
        + high2: will be represented by a second point threshold
        + high3: will be represented by a third point threshold
    - adRewardCurrency: identifies the measurement category
      * eligibleAmount: amount of awardable reward currency
      * paidAmount: amount paid out of eligible reward currency
      * adRcAvailable: amount of reward currency held
      * secondTransaction: number of repeat purchases from same customer
      * rcSecondTransaction: repeat purchase from same customer with RC
    - advertiserPerformance: identifies measurement category
      * savedAdOccurences: number of offer saved by users
      * purchaseOccurences: number of offers resulting in purchase
      * compoundDiscountOfferPercent: accumulative discount percent total of all offers issued by advertiser
      * accumulatedAdPoint: number of accumulated ad points
      * rankPointEarned: number of rank points earned
      * totalCashValueOfTransactions; total dollar amount of all transactions
      * adOfferProductVariation:
        + cat1: offers issued from category one
        + cat2: offers issued from category two
      * adAcquisitionPercent
        + oneToTenPercent: offers between one and ten percent
        + tenToTwentyPercent: offers between ten and twenty percent
        + twentyToThirtyPercent: offer between twenty and thirty percent
        + thirtyToFortyPercent: offers between thirty and forty percent
        + fortyToFiftyPercent; offers between forty and fifty percent
    - advertiserSocialActions: identifies the measurement category
      * promotions: number of offers with add-on promotion
      * sharing: number of shares from users
      * numberOfLinks:
      * totalNumberOfOffers: total number of offers

ID-03: getOfferEndTimeInNumberFormat()

* Function gets the selected store ad offer end time into number format.