Eligible users

```
//reading the email table
var email = spark.read.parquet("s3://ha-prod-omnidata-us-east-1/marketing/email/ocelot/temp/training/input/email")
```

```
email: org.apache.spark.sql.DataFrame = [source_id: string, first_name: string ... 14 more fields]
```

```
//checking the schema and count
email.printSchema()
email.count
```

```
root
 |-- source_id: string (nullable = true)
 |-- first_name: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- email_address: string (nullable = true)
 |-- email_parallax: string (nullable = true)
 |-- email_hashid: string (nullable = true)
 |-- user_name: string (nullable = true)
 |-- user_uuid: string (nullable = true)
 |-- company_name: string (nullable = true)
 |-- salesforce_account_name: string (nullable = true)
 |-- user_guid: string (nullable = true)
 |-- public_uuid: string (nullable = true)
 |-- create_date: timestamp (nullable = true)
 |-- update_date: timestamp (nullable = true)
 |-- last_modified_datetime: timestamp (nullable = true)
 |-- source_type: string (nullable = true)
res107: Long = 5000000
```

Select first_name, last_name, email_address, user_uuid, public_uuid, email_hashid column from Email table

```
email.createOrReplaceTempView("Email")
var email_table = spark.sql("SELECT first_name, last_name, email_address, user_uuid, public_uuid, email_hashid from Email")
email_table.printSchema()
email_table.show()
```

```
email_table: org.apache.spark.sql.DataFrame = [first_name: string, last_name: string ... 4 more fields]
root
 |-- first_name: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- email_address: string (nullable = true)
 |-- user_uuid: string (nullable = true)
 |-- public_uuid: string (nullable = true)
 |-- email_hashid: string (nullable = true)
+------------------+----------+--------------------+--------------------+--------------------+--------------------+
|        first_name| last_name|       email_address|           user_uuid|         public_uuid|        email_hashid|
+------------------+----------+--------------------+--------------------+--------------------+--------------------+
|             Leann|   Ritchie|leann.ritchie@gma...|000b584d-dfb1-419...|a87bae9e-6712-4cb...|1954f20136d7758ac...|
|              null|      null|gaemulliez@gmail.com|001bc1bb-df46-434...|13637571-4d59-46f...|6d932e3f3e444e798...|
|            Marisa|Castricone|marisacastricone@...|00689c5c-dee4-42e...|3151a7b9-3d91-482...|5ad7e8e6bc3d5bc1b...|
|         Jacqueline|Timmermans|jacqueline.timmer...|0090d398-bf68-496...|fd021252-79cf-474...|f1e0bd5539401d928...|
|             tiana|     myers|tianamarie.christ...|00953994-d1e0-45c...|1e379bb9-8b6a-4fa...|819ade92a6c7363ae...|
|A Verified Traveler|      null|663598792@expedia...|009c0765-5e64-499...|fb411334-38b9-450...|ad881264e5c669cd8...|
|             Nancy|   Culbert|   ncbert@rogers.com|009ce037-f95e-4a5...|f5af844c-7e35-432...|7496e563b21c39021...|
```

Clean the first_name and last_name column i.e., trim, capitalize.

```
import org.apache.spark.sql.functions.{trim,ltrim,rtrim,col}
email_table = email_table.withColumn("first_name", initcap(trim(col("first_name"))))
email_table = email_table.withColumn("last_name", initcap(trim(col("last_name"))))

email_table.select("first_name","last_name").show()
```

```
import org.apache.spark.sql.functions.{trim, ltrim, rtrim, col}
email_table: org.apache.spark.sql.DataFrame = [first_name: string, last_name: string ... 4 more fields]
email_table: org.apache.spark.sql.DataFrame = [first_name: string, last_name: string ... 4 more fields]
+------------------+----------+
|        first_name| last_name|
+------------------+----------+
|             Leann|   Ritchie|
|              null|      null|
|            Marisa|Castricone|
|         Jacqueline|Timmermans|
|             Tiana|     Myers|
|A Verified Traveler|      null|
|             Nancy|   Culbert|
|             Nikki|    Barron|
|              null|      null|
|            Andrea|   Dickson|
|            Ashley|    Loftis|
|          Margaret| Perreault|
```

Remove invalid email_address where @ is not present

```
email_table = email_table.filter(col("email_address").contains("@"))
email_table.show()
email_table.count()
```

```
email_table: org.apache.spark.sql.DataFrame = [first_name: string, last_name: string ... 4 more fields]
+------------------+----------+--------------------+--------------------+--------------------+--------------------+
|        first_name| last_name|       email_address|           user_uuid|         public_uuid|        email_hashid|
+------------------+----------+--------------------+--------------------+--------------------+--------------------+
|             Leann|   Ritchie|leann.ritchie@gma...|000b584d-dfb1-419...|a87bae9e-6712-4cb...|1954f20136d7758ac...|
|              null|      null|gaemulliez@gmail.com|001bc1bb-df46-434...|13637571-4d59-46f...|6d932e3f3e444e798...|
|            Marisa|Castricone|marisacastricone@...|00689c5c-dee4-42e...|3151a7b9-3d91-482...|5ad7e8e6bc3d5bc1b...|
|         Jacqueline|Timmermans|jacqueline.timmer...|0090d398-bf68-496...|fd021252-79cf-474...|f1e0bd5539401d928...|
|             Tiana|     Myers|tianamarie.christ...|00953994-d1e0-45c...|1e379bb9-8b6a-4fa...|819ade92a6c7363ae...|
|A Verified Traveler|      null|663598792@expedia...|009c0765-5e64-499...|fb411334-38b9-450...|ad881264e5c669cd8...|
|             Nancy|   Culbert|   ncbert@rogers.com|009ce037-f95e-4a5...|f5af844c-7e35-432...|7496e563b21c39021...|
|             Nikki|    Barron|nikkimariebarron@...|00a0adb2-42ed-45a...|d1db09dc-c9cf-409...|703432ec5bc725c81...|
|              null|      null| dpadair@hotmail.com|00a1b712-29dc-499...|0cb1bb97-862f-464...|57f169aafdf5e030c...|
|            Andrea|   Dickson|    acdu@comcast.net|00b153fe-5960-4c0...|edbff5c2-adfd-40d...|1562a583166bc30d1...|
|            Ashley|    Loftis|ashleykloftis@gma...|00c98ed6-1761-4e3...|d0fbd455-8a8d-43b...|e81b217300b31bf86...|
|          Margaret| Perreault|perreaultmargaret...|00cf2378-bd0e-4e2...|fc0d071f-bc82-411...|60448bb0894a253a4...|
|          Danielle|      null|interhome_6cd8446...|00e76b9a-ac7e-415...|b4c2131b-e1b8-444...|90a05a0d5432f413f...|
|          Patricia|  Lockyer|patricialockyer@h...|00e92ada-0919-430...|574ffea1-fadb-4ea...|9083392c20f2e5a83...|
```

```
//reading visitor preference table
var vis_pref=spark.read.parquet("s3://ha-prod-omnidata-us-east-1/marketing/email/ocelot/temp/training/input/visitor_preference")
```

```
vis_pref: org.apache.spark.sql.DataFrame = [emailhash: string, brandid: int ... 3 more fields]
```

```
//checking the schema and count
vis_pref.printSchema()
vis_pref.count

vis_pref.show()
```

```
root
```

```
 |-- emailhash: string (nullable = true)
 |-- brandid: integer (nullable = true)
 |-- subscribed: boolean (nullable = true)
 |-- subscribed_date: timestamp (nullable = true)
 |-- email_hashid: string (nullable = true)
res123: Long = 3876065
+--------------------+-------+----------+--------------------+--------------------+
|           emailhash|brandid|subscribed|     subscribed_date|        email_hashid|
+--------------------+-------+----------+--------------------+--------------------+
|0000c97970344e422...|    311|      true|2014-04-16 19:38:...|0000c97970344e422...|
|000170f4912a4a230...|    617|      true| 2022-03-27 15:17:31|000170f4912a4a230...|
|000272709f3a8ed02...|    731|      true|2015-01-19 05:06:...|000272709f3a8ed02...|
|0008224ff19b722ca...|    321|      true|2018-09-21 09:29:...|0008224ff19b722ca...|
|000cafc0a61279f1c...|    321|      true|2015-12-10 06:06:...|000cafc0a61279f1c...|
|0014846d4a84a3006...|    321|      true|2018-05-20 06:06:...|0014846d4a84a3006...|
|00152093b78fb4e36...|    138|      true|2016-05-09 06:12:...|00152093b78fb4e36...|
```

```
//checking the distinct count of email hashid and brandid in the respective tables
email_table.select(countDistinct("email_hashid")).show()
vis_pref.select(countDistinct("email_hashid","brandid")).show()
```

```
+---------------------------+
|count(DISTINCT email_hashid)|
+---------------------------+
|                    4930902|
+---------------------------+
+-----------------------------------+
|count(DISTINCT email_hashid, brandid)|
+-----------------------------------+
|                            3876065|
+-----------------------------------+
```

```
//checking if there are any duplicates
val c = vis_pref.groupBy("brandid","email_hashid").count().show(100,false)
```

```
+-------+----------------------------------------------------------------+-----+
|brandid|email_hashid                                                    |count|
+-------+----------------------------------------------------------------+-----+
|321    |017a9e71b916165086027c084575c143474100f5276bf68a3c72e1d9698cd592|1    |
|321    |048971eb423940a0c17695f79bb1ac844cbf7b5a23ab8a20a888f24ae1ab9be3|1    |
|138    |04b547f1cb25195290ec54ca595e37c5b2d4410ea8607bd711b8c69765ac67b4|1    |
|321    |09458926d1ae989d94fb8feada0b73c6b83acc5af676a8b39e59deac11f1cb12|1    |
|321    |0a187a65376d3671518a66625cda5057b41e91e9b8e2ace703e0a0e26b9887a3|1    |
|321    |0cd4415ddb96ce6f4545d73670613defbbf64c271ca717f164e3703e606439a9|1    |
|611    |0ec0e17d2637ae6e2b1a61dcc6719ef0ac559b05b459c97fd6d9fd2d3664d305|1    |
|321    |109bc84f1eb116ea583425a954a70c7cc5c6fe0ec376bfff8ebd028ff86b6c89|1    |
|321    |11781344417784bc69fae4bb92fa464ee138254fb65d68ef983f9a0b5acbfd8bc|1    |
|614    |132cc6a54c1ada310a82a29bb7de72778d5a83209cfd98a022fac0716699e4d0|1    |
|321    |134e493275421f8443c70ce64df544de919eba02d7306f4f20d8a8fc4d5b29f7|1    |
|311    |1358791999acfffeb6e185b2714e5592d675506dfe399cd481e8d7114936780d|1    |
|321    |188db7348720f1a02e1bc3847676668e82001ef83c2170da3a4d6b5f7f33dc3b|1    |
|321    |1cde5f48b8096175614d86a6e99ee8eea8dff73522923caf3efef286a84a92e9|1    |
|321    |1ce0cdd01e8774d79d100601eb01eb9be0ac8c046dd3063b3636b13e2520f9ee|1    |
```

```
//Again verifying for no duplicacy
vis_pref.filter(vis_pref("email_hashid")==="1e28594aa1429bfc36d10e93331273685ec25285ac569fb06d09406a2dddc966").show(false)
```

```
+----------------------------------------------------------------+-------+----------+----------------------+----------------------------------------------------------------+
|emailhash                                                       |brandid|subscribed|subscribed_date       |email_hashid                                                    |
+----------------------------------------------------------------+-------+----------+----------------------+----------------------------------------------------------------+
|1e28594aa1429bfc36d10e93331273685ec25285ac569fb06d09406a2dddc966|321    |true      |2020-01-22 13:32:21.403|1e28594aa1429bfc36d10e93331273685ec25285ac569fb06d09406a2dddc966|
|1e28594aa1429bfc36d10e93331273685ec25285ac569fb06d09406a2dddc966|632    |true      |2020-01-23 04:28:47   |1e28594aa1429bfc36d10e93331273685ec25285ac569fb06d09406a2dddc966|
+----------------------------------------------------------------+-------+----------+----------------------+----------------------------------------------------------------+
```

```
//counting the distinct email_hashid
vis_pref.select(countDistinct("email_hashid")).show()
```

```
+---------------------------+
|count(DISTINCT email_hashid)|
+---------------------------+
|                    3131179|
+---------------------------+
```

```
//import org.apache.spark.sql.functions._
//import org.apache.spark.sql.expressions.Window
//val w = Window.partitionBy("email_hashid").orderBy(col("subscribed_date"))
//var min_subs = vis_pref.withColumn("row",row_number.over(w)).filter(col("row")=== 1).drop("row")
```

```
// do the inner join and check
var email_join = email_table.join(vis_pref, email_table("email_hashid") === vis_pref("email_hashid"),"inner").drop(vis_pref("email_hashid"))
email_join.printSchema
email_join.count
email_join.select(countDistinct("brandid")).show()
```

```
email_join: org.apache.spark.sql.DataFrame = [first_name: string, last_name: string ... 8 more fields]
root
 |-- first_name: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- email_address: string (nullable = true)
 |-- user_uuid: string (nullable = true)
 |-- public_uuid: string (nullable = true)
 |-- email_hashid: string (nullable = true)
 |-- emailhash: string (nullable = true)
 |-- brandid: integer (nullable = true)
 |-- subscribed: boolean (nullable = true)
 |-- subscribed_date: timestamp (nullable = true)
res142: Long = 3876064
+----------------------+
|count(DISTINCT brandid)|
+----------------------+
|                    56|
+----------------------+
```

Rename subscribed_date column as min_subscribed_date and change data type as date.

```
email_join = email_join.withColumnRenamed("subscribed_date","min_subscribe_date")
email_join = email_join.withColumn("min_subscribe_date", to_date(col("min_subscribe_date")))
email_join.printSchema
```

```
email_join: org.apache.spark.sql.DataFrame = [first_name: string, last_name: string ... 8 more fields]
email_join: org.apache.spark.sql.DataFrame = [first_name: string, last_name: string ... 8 more fields]
root
 |-- first_name: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- email_address: string (nullable = true)
 |-- user_uuid: string (nullable = true)
 |-- public_uuid: string (nullable = true)
 |-- email_hashid: string (nullable = true)
 |-- emailhash: string (nullable = true)
```

```
 |-- brandid: integer (nullable = true)
 |-- subscribed: boolean (nullable = true)
 |-- min_subscribe_date: date (nullable = true)
```

```
//reading the ocelot table
var ocelot = spark.read.parquet("s3://ha-prod-omnidata-us-east-1/marketing/email/ocelot/temp/training/input/ocelot")
```

```
ocelot: org.apache.spark.sql.DataFrame = [eapid: bigint, brandid: int ... 5 more fields]
```

```
//checking the schema and count
ocelot.printSchema
ocelot.count
ocelot.select(countDistinct("brandid")).show()
```

```
root
 |-- eapid: long (nullable = true)
 |-- brandid: integer (nullable = true)
 |-- tpid: integer (nullable = true)
 |-- locale: string (nullable = true)
 |-- moniker_site: string (nullable = true)
 |-- country_name: string (nullable = true)
 |-- langid: long (nullable = true)
res151: Long = 28
+----------------------+
|count(DISTINCT brandid)|
+----------------------+
|                    28|
+----------------------+
```

```
ocelot.show()
```

```
+-----+-------+----+--------------------+--------------+-------------+------+
|eapid|brandid|tpid|              locale|   moniker_site|  country_name|langid|
+-----+-------+----+--------------------+--------------+-------------+------+
|    2|    311|9001|en_us_vacationren...|vacationrentals| united states|  1033|
|   19|    635|9003|      el_gr_homeaway|    homeaway_gr|        greece|  1032|
|   24|    138|9005|         en_au_stayz|         stayz|     australia|  3081|
|    3|    127|9002|          en_ca_vrbo|    homeaway_ca|        canada|  4105|
|    8|    656|9003|          de_at_vrbo|    homeaway_at|       austria|  3079|
|   29|    133|9006|      en_nz_homeaway|    homeaway_nz|   new zealand|  5129|
|    7|    661|9004|  en_gb_ownersdirect|          odhr|united kingdom|  2057|
|    6|    611|9004|          en_gb_vrbo|    homeaway_uk|united kingdom|  2057|
|   29|    137|9006|     en_nz_bookabach|      bookabach|   new zealand|  5129|
|   30|    145|9001|      en_lk_homeaway|    homeaway_lk|     sri lanka|  2057|
|   24|    130|9005|      en_au_homeaway|    homeaway_au|     australia|  3081|
|   16|    617|9003|          nl_nl_vrbo|    homeaway_nl|   netherlands|  1043|
|   20|    651|9003|      de_de_homeaway|    homeaway_de|       germany|  1031|
|   18|    629|9001|      pl_pl_homeaway|    homeaway_pl|        poland|  1045|
|   17|    622|9001|          fi_fi_vrbo|    homeaway_fi|       finland|  1035|
```

```
//reading the blacklisted table
var blacklisted = spark.read.parquet("s3://ha-prod-omnidata-us-east-1/marketing/email/ocelot/temp/training/output/premnivas/blacklisted2")
```

```
blacklisted: org.apache.spark.sql.DataFrame = [email_hash: string, activity_type: string ... 4 more fields]
```

```
//checking the schema and count
blacklisted.printSchema
blacklisted.count
```

```
root
 |-- email_hash: string (nullable = true)
 |-- activity_type: string (nullable = true)
 |-- activity_timestamp: timestamp (nullable = true)
 |-- category_code: integer (nullable = true)
 |-- row_number: integer (nullable = true)
 |-- is_blacklisted: boolean (nullable = true)
res160: Long = 1139724
```

Bring out tpid, eapid and locale using Ocelot Brand Table for a brand id

```
//left joining the email_join table and ocelot table on brandid
var brand_join = email_join.join(ocelot, email_join("brandid")===ocelot("brandid"),"left").drop(ocelot("brandid"))
```

```
brand_join: org.apache.spark.sql.DataFrame = [first_name: string, last_name: string ... 14 more fields]
```

```
//checking the schema and count
brand_join.printSchema
brand_join.count
brand_join.show(false)
```

```
root
 |-- first_name: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- email_address: string (nullable = true)
 |-- user_uuid: string (nullable = true)
 |-- public_uuid: string (nullable = true)
 |-- email_hashid: string (nullable = true)
 |-- emailhash: string (nullable = true)
 |-- brandid: integer (nullable = true)
 |-- subscribed: boolean (nullable = true)
 |-- min_subscribe_date: date (nullable = true)
 |-- eapid: long (nullable = true)
 |-- tpid: integer (nullable = true)
 |-- locale: string (nullable = true)
 |-- moniker_site: string (nullable = true)
 |-- country_name: string (nullable = true)
 |-- langid: long (nullable = true)
res166: Long = 3876064
```

```
//left joining brand_join table and blacklisted table on email_hashid
var final_set= brand_join.join(blacklisted,brand_join("email_hashid")===blacklisted("email_hash"),"left")
```

```
final_set: org.apache.spark.sql.DataFrame = [first_name: string, last_name: string ... 20 more fields]
```

```
//checking the schema and count
final_set.printSchema()
final_set.count
final_set.show(false)
```

```
root
 |-- first_name: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- email_address: string (nullable = true)
 |-- user_uuid: string (nullable = true)
 |-- public_uuid: string (nullable = true)
 |-- email_hashid: string (nullable = true)
 |-- emailhash: string (nullable = true)
 |-- brandid: integer (nullable = true)
```

```
 |-- subscribed: boolean (nullable = true)
 |-- min_subscribe_date: date (nullable = true)
 |-- eapid: long (nullable = true)
 |-- tpid: integer (nullable = true)
 |-- locale: string (nullable = true)
 |-- moniker_site: string (nullable = true)
 |-- country_name: string (nullable = true)
 |-- langid: long (nullable = true)
```

Remove the Blacklisted user (is_blacklisted = TRUE) from above data frame

```
// filtering false and null value
var eligible_users = final_set.filter(final_set("is_blacklisted")===false || final_set("is_blacklisted").isNull)
```

```
eligible_users: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [first_name: string, last_name: string ... 20 more fields]
```

```
//Checking schema and count
eligible_users.printSchema
eligible_users.count
eligible_users.select("is_blacklisted").show()
```

```
root
 |-- first_name: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- email_address: string (nullable = true)
 |-- user_uuid: string (nullable = true)
 |-- public_uuid: string (nullable = true)
 |-- email_hashid: string (nullable = true)
 |-- emailhash: string (nullable = true)
 |-- brandid: integer (nullable = true)
 |-- subscribed: boolean (nullable = true)
 |-- min_subscribe_date: date (nullable = true)
 |-- eapid: long (nullable = true)
 |-- tpid: integer (nullable = true)
 |-- locale: string (nullable = true)
 |-- moniker_site: string (nullable = true)
 |-- country_name: string (nullable = true)
 |-- langid: long (nullable = true)
 |-- email_hash: string (nullable = true)
```

Write the data into the Eligible Users Path

```
//overwrite the data
import org.apache.spark.sql.SaveMode
eligible_users.write.mode("overwrite").parquet("s3://ha-prod-omnidata-us-east-1/marketing/email/ocelot/temp/training/output/premnivas/eligible_user_task")
```

```
import org.apache.spark.sql.SaveMode
```

```
//reading the data
var eligible_users = spark.read.parquet("s3://ha-prod-omnidata-us-east-1/marketing/email/ocelot/temp/training/output/premnivas/eligible_user_task")
```

```
eligible_users: org.apache.spark.sql.DataFrame = [first_name: string, last_name: string ... 20 more fields]
```

```
// checking the schema and count
eligible_users.printSchema
eligible_users.count()
eligible_users.show(10,false)
```

```
root
 |-- first_name: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- email_address: string (nullable = true)
 |-- user_uuid: string (nullable = true)
 |-- public_uuid: string (nullable = true)
 |-- email_hashid: string (nullable = true)
 |-- emailhash: string (nullable = true)
 |-- brandid: integer (nullable = true)
 |-- subscribed: boolean (nullable = true)
 |-- min_subscribe_date: date (nullable = true)
 |-- eapid: long (nullable = true)
 |-- tpid: integer (nullable = true)
 |-- locale: string (nullable = true)
 |-- moniker_site: string (nullable = true)
 |-- country_name: string (nullable = true)
 |-- langid: long (nullable = true)
 |-- email_hash: string (nullable = true)
```