# AirBnB Dataset

The following steps allows the download and import of the AirBnB dataset

- Open the following Link and download the *listings* and *reviews* datasets.
- Place these files within the *import* folder in your Neo4j installation folder.
- Run the following scripts to import the data. It may take quite a while to import the *reviews* file (about 10-15 minutes).

**SCRIPT #1**

```
CREATE CONSTRAINT ON (h:Host) ASSERT h.host_id IS UNIQUE;
CREATE CONSTRAINT ON (l:Listing) ASSERT l.lising_id IS UNIQUE;
CREATE CONSTRAINT ON (u:User) ASSERT u.user_id IS UNIQUE;
CREATE CONSTRAINT ON (a:Amenity) ASSERT a.name IS UNIQUE;
CREATE CONSTRAINT ON (c:City) ASSERT c.citystate IS UNIQUE;
CREATE CONSTRAINT ON (s:State) ASSERT s.code IS UNIQUE;
CREATE CONSTRAINT ON (c:Country) ASSERT c.code IS UNIQUE;
CREATE CONSTRAINT ON (r:Review) ASSERT r.review_id IS
UNIQUE;
```

**SCRIPT #2**

```
LOAD CSV WITH HEADERS FROM "file:///listings.csv" AS row
WITH row WHERE row.id IS  NOT NULL
MERGE (l:Listing {listing_id: row.id})
ON CREATE SET l.name  = row.name,
l.latitude  = toFloat(row.latitude),
l.longitude  = toFloat(row.longitude),
l.reviews_per_month  = toFloat(row.reviews_per_month),
l.cancellation_policy  = row.cancellation_policy,
l.instant_bookable  = CASE WHEN row.instant_bookable = "t" THEN
true ELSE false END,
l.review_scores_value  = toInteger(row.review_scores_value),
l.review_scores_location  = toInteger(row.review_scores_location),
l.review_scores_communication =
toInteger(row.review_scores_communication),
l.review_scores_checkin  = toInteger(row.review_scores_checking),
```

The Datasets and Codes were taken and updated from
https://github.com/johnymontana/neo4j-datasets/tree/master

```
l.review_scores_cleanliness  =
toInteger(row.review_scores_cleanliness),
l.reivew_scores_accuracy  = toInteger(row.review_scores_accuracy),
l.review_scores_rating  = toInteger(row.review_scores_rating),
l.availability_365  = toInteger(row.availability_365),
l.availability_90  = toInteger(row.availability_90),
l.availability_60  = toInteger(row.availability_60),
l.availability_30  = toInteger(row.availability_30),
l.price  = toFloat(substring(row.price, 1)),
l.cleaning_fee  = toFloat(substring(row.cleaning_free, 1)),
l.security_deposit  = toFloat(substring(row.security_deposit, 1)),
l.monthly_price  = toFloat(substring(row.monthly_price, 1)),
l.weekly_price  = toFloat(substring(row.weekly_price, 1)),
l.square_feet  = toInteger(row.square_feet),
l.bed_type  = row.bed_type,
l.beds  = toInteger(row.beds),
l.bedrooms  = toInteger(row.bedrooms),
l.bathrooms  = toFloat(row.bathrooms),
l.accommodates  = toInteger(row.accommodates),
l.room_type  = row.room_type,
l.property_type  = row.property_type
ON MATCH SET l.count = coalesce(l.count, 0) + 1


MERGE (n:Neighborhood {neighborhood_id:
coalesce(row.neighbourhood_cleansed, "NA")})
SET n.name = row.neighbourhood
MERGE (c:City {citystate: coalesce(row.city + "-" + row.state, "NA")})
ON CREATE SET c.name = row.city
MERGE (l)-[:IN_NEIGHBORHOOD]->(n)
MERGE (n)-[:LOCATED_IN]->(c)
MERGE (s:State {code: coalesce(row.state, "NA")})
MERGE (c)-[:IN_STATE]->(s)
MERGE (country:Country {code: coalesce(row.country_code, "NA")})
SET country.name = row.country
MERGE (s)-[:IN_COUNTRY]->(country)
```

```
WITH l, split(replace(replace(replace(row.amenities, "{", ""), "}", ""), "\"",
""), ",") AS amenities
UNWIND amenities AS amenity
MERGE (a:Amenity {name: amenity})
MERGE (l)-[:HAS]->(a);

LOAD CSV WITH HEADERS FROM "file:///listings.csv" AS row
WITH row WHERE row.host_id IS NOT NULL
MERGE (h:Host {host_id: row.host_id})
ON CREATE SET h.name  = row.host_name,
h.about  = row.host_about,
h.verifications  = row.host_verifications,
h.listings_count  = toInteger(row.host_listings_count),
h.acceptance_rate = toFloat(row.host_acceptance_rate),
h.host_since  = row.host_since,
h.url  = row.host_url,
h.response_rate  = row.host_response_rate,
h.superhost  = CASE WHEN row.host_is_super_host = "t" THEN True
ELSE False END,
h.location  = row.host_location,
h.verified  = CASE WHEN row.host_identity_verified = "t" THEN True
ELSE False END,
h.image  = row.host_picture_url
ON MATCH SET h.count = coalesce(h.count, 0) + 1
MERGE (l:Listing {listing_id: row.id})
MERGE (h)-[:HOSTS]->(l);
```

**SCRIPT #3**

```
:auto LOAD CSV WITH HEADERS FROM "file:///reviews.csv" AS row

MERGE (u:User {user_id: row.reviewer_id})
SET u.name = row.reviewer_name



MERGE (r:Review {review_id: row.id})
ON CREATE SET r.date  = row.date,
```

```
r.comments = row.comments
WITH row, u, r
MATCH (l:Listing {listing_id: row.listing_id})
MERGE (u)-[:WROTE]->(r)
MERGE (r)-[:REVIEWS]->(l);
```

After these operations are done running, you may check that your data exists by running this simple query.

```
MATCH (n:Listing)-[r]-(m) RETURN n,r,m LIMIT 50
```