**Milestone 3 - Populate DB in MySQL & Server Side**

**Group Members & Github Username**

Jingle Wang, [gillianw@seas.upenn.edu](mailto:gillianw@seas.upenn.edu) (gillianwang0325)

Ziqi He, [ziqihe@seas.upenn.edu](mailto:ziqihe@seas.upenn.edu) (HeidiHe2001)

Lingpei Luo, [luol@seas.upenn.edu](mailto:luol@seas.upenn.edu) (JLmm123)

Guangqiuse Hu, [violahu@seas.upenn.edu](mailto:violahu@seas.upenn.edu) (violahu930)

[Crime Data from 2020 to Present - Catalog](https://catalog.data.gov/dataset/crime-data-from-2020-to-present)

[Airbnb listings](https://data.insideairbnb.com/united-states/ca/los-angeles/2023-12-03/data/listings.csv)

1. Clean airbnb data, separate into R
2. Revise crime data, delete unnecessary info
3. Aws RDS setup, document credential
4. Populate MySQL database
5. Skeleton - folder directory setup
6. Write SQL query - test in data grip
7. Embed SQL into server.js
8. Check website and make sure the data are pulled successfully

Datasets:

| R1 | DR\_NO, Date Occ, Time Occ, AREA, Crm Cd, Premis Cd, Weapon Used Cd, Status |
| --- | --- |
| R2 | AREA, AREA NAME, SUBAREA NAME |
| R3 | Crm Cd, Crm Cd Desc |
| R4 | Premis Cd, Premis Desc |
| R5 | Weapon Used Cd, Weapon Desc |
| R6 | Status, Status Desc |
| R7 | Crm Cd, Part1-2 |
| Airbnb | Id, host\_id, host\_name, neighborhood, room\_type, price, minimum\_nights, number\_of\_reviews, availability\_365, airbnb\_name, star, bedrooms, beds, baths |

**Queries with Description**

1. Identifying Neighborhoods with High Average Price and Low Crime Rates

| WITH avg\_price\_per\_neighbourhood AS (  SELECT R2.AREA AS area, AVG(ai.price) AS avg\_price  FROM Airbnb  LEFT JOIN R2 ON airbnb.neighbourhood = R2.SUBAREA NAME  GROUP BY R2.AREA),  crime\_summary AS (  SELECT AREA, COUNT(\*) AS total\_crimes  FROM R1  GROUP BY AREA)  SELECT apn.area, apn.avg\_price, cs.total\_crimes  FROM avg\_price\_per\_neighbourhood apn  LEFT JOIN crime\_summary cs ON apn.area = cs.AREA  WHERE apn.avg\_price > (SELECT AVG(price) FROM Airbnb) -- *Filter by above average price*  AND cs.total\_crimes < (SELECT AVG(total\_crimes) FROM crime\_summary) *-- Filter by below average crimes*  ORDER BY apn.avg\_price DESC; |
| --- |

1. Given area, retrieve total\_incidents in this area, unresolved\_incident\_rate in this area, avg\_price of Airbnbs, total\_listings of Airbnbs, total\_reviews of Airbnbs in this area

| WITH crime\_summary AS (  SELECT AREA, COUNT(\*) AS total\_incidents,  AVG(CASE WHEN `Status` = 'IC' THEN 1 ELSE 0 END) AS unresolved\_incident\_rate  FROM R1  WHERE YEAR(`Date Occ`) = 2023 *-- Consider incidents from the year 2023*  GROUP BY AREA),  airbnb\_summary AS (  SELECT R2.AREA AS AREA, AVG(airbnb.price) AS avg\_price, COUNT(\*) AS total\_listings  FROM airbnb JOIN R2 ON airbnb.neighborhood = R2. SUBAREA NAME  GROUP BY R2.AREA),  popular\_area AS (  SELECT R2.AREA AS AREA, COUNT(\*) AS total\_reviews  FROM Airbnb JOIN R2 ON airbnb.neighborhood = R2. SUBAREA NAME  WHERE number\_of\_reviews > 50 *-- Filter by listings with more than 50 reviews*  GROUP BY R2.AREA)  SELECT cs.AREA, cs.total\_incidents, cs.unresolved\_incident\_rate, asum.avg\_price, asum.total\_listings, pn.total\_reviews  FROM crime\_summary cs  LEFT JOIN airbnb\_summary asum ON cs.AREA = asum.AREA  LEFT JOIN popular\_neighbourhoods pn ON cs.AREA = pn.AREA  ORDER BY cs.total\_incidents DESC; |
| --- |

1. Identifying High-Demand Airbnb Listings in Areas with Low Crime Rates

| WITH crime\_summary AS (  SELECT AREA, COUNT(\*) AS total\_crimes  FROM R1  GROUP BY AREA),  low\_crime\_areas AS (  SELECT AREA  FROM R1  GROUP BY AREA  HAVING COUNT(\*) < (SELECT AVG(total\_crimes) FROM crime\_summary))  SELECT airbnb\_name, price  FROM Airbnb  WHERE neighborhood IN (  SELECT R2.SUBAREA NAME FROM low\_crime\_areas la  LEFT JOIN R2 ON la.AREA = R2.Area)  ORDER BY number\_of\_reviews DESC; |
| --- |

1. Identifying Airbnb Listings Near High Crime Density Areas

| WITH crime\_density\_per\_area AS (  SELECT AREA, COUNT(\*) AS crime\_density  FROM R1  GROUP BY AREA),  avg\_density\_per\_neighborhood AS(  SELECT AVG(crime\_density) FROM crime\_density\_per\_neighbourhood  )  SELECT airbnb\_data.airbnb\_name, airbnb\_data.price, airbnb\_data.area, crime\_density  FROM  (SELECT  a.airbnb\_name AS airbnb\_name, a.price AS price, R2.AREA AS area  FROM airbnb a  LEFT JOIN R2 ON a.neighborhood = R2.SUBAREA\_NAME) airbnb\_data  LEFT JOIN crime\_density\_per\_area cd ON airbnb\_data.area = cd.AREA  ORDER BY cd.crime\_density DESC; |
| --- |

5. Most popular neighborhood: sort neighborhood with the most Airbnb listings

| SELECT neighborhood, COUNT(\*) AS listing\_count  FROM airbnb a  GROUP BY a.neighborhood  ORDER BY COUNT(\*) DESC; |
| --- |

6. Safest area: sort area with the lowest crime number

| WITH crime\_counts AS (  SELECT AREA AS area\_name, COUNT(\*) AS crime\_count  FROM R1  GROUP BY AREA)  SELECT area\_name, crime\_count  FROM crime\_counts  ORDER BY crime\_count ASC; |
| --- |

7. Given Airbnb id, retrieve Airbnb Info

Parameters: given\_id

| SELECT \* FROM Airbnb  WHERE Airbnb.id = given\_id |
| --- |

8. Given area select all airbnb id in area

Parameters: given\_area

| SELECT id  FROM Airbnb a  JOIN R2 ON R2.SUBAREA NAME= Airbnb.neighbourhood  WHERE R2.AREA = given\_area |
| --- |

9. For slider: filter airbnb listings by high/low star; high/low crime rate; high/low bedroom

Parameters: high\_star, low\_star, high\_crime\_rate, low\_crime\_rate, high\_bedroom, low\_bedroom

| WITH crime\_neighborhood\_summary AS (  SELECT R2.AREA AS AREA, R2. SUBAREA NAME AS neighborhood, COUNT(\*) AS total\_crimes  FROM R1  JOIN R2 ON R2. AREA = R1.AREA  GROUP BY R2.AREA),  SELECT id  FROM Airbnb a  JOIN crime\_neighborhood\_summary cns ON a.neighborhood = cns.neighborhood  WHERE cns.total\_crimes > low\_crime\_rate AND cns.total\_crimes < high\_crime\_rate  AND a.star > low\_star AND a.star < high\_star  AND a.bedroom > low\_bedroom AND a.bedroom < high\_bedroom |
| --- |

10. Rank each neighborhood by crime count and severance level, and also calculate the average rating of Airbnb listings in each neighborhood

| WITH CrimeCounts AS (  SELECT R2.SUBAREA\_NAME AS neighborhood, COUNT(\*) AS crime\_count,  SUM(CASE WHEN R7.Part1\_2 = 1 THEN 1 ELSE 0 END) AS severe\_crime\_count  FROM R1  JOIN R2 ON R1.AREA = R2.AREA  JOIN R7 ON R1.Crm\_Cd = R7.Crm\_Cd  GROUP BY R2.SUBAREA\_NAME),  AverageRatings AS (  SELECT neighborhood, AVG(star) AS avg\_rating  FROM Airbnb  GROUP BY neighborhood)  SELECT c.subarea, c.crime\_count, c.severe\_crime\_count, a.avg\_rating,  RANK() OVER (ORDER BY c.crime\_count, c.severe\_crime\_count) AS crime\_rank,  RANK() OVER (ORDER BY a.avg\_rating DESC) AS rating\_rank  FROM CrimeCounts c JOIN AverageRatings a ON c.neighborhood = a.neighborhood; |
| --- |

**Credentials**

Host: admin

User: database-1.c7uq84c4kqir.us-east-2.rds.amazonaws.com

Password: aScC\*t{531yA1m8Y

Port: 3306

**Schema DDL:**

CREATE DATABASE CRIME\_AIRBNB

USE CRIME\_AIRBNB

CREATE TABLE Premises (

Premis\_Cd INT PRIMARY KEY,

Premis\_Desc VARCHAR(255)

);

CREATE TABLE Areas (

AREA INT,

AREA\_NAME VARCHAR(255),

SUBAREA\_NAME VARCHAR(255),

PRIMARY KEY(AREA, SUBAREA\_NAME)

);

CREATE TABLE Weapons (

Weapon\_Used\_Cd INT PRIMARY KEY,

Weapon\_Desc VARCHAR(255)

);

CREATE TABLE Status (

Status CHAR(2) PRIMARY KEY,

Status\_Desc VARCHAR(255)

);

CREATE TABLE CrimeData (

DR\_NO INT PRIMARY KEY,

Date\_Occ DATE,

Time\_Occ TIME,

AREA INT,

Crm\_Cd INT,

Premis\_Cd INT,

Weapon\_Used\_Cd INT,

Status CHAR(2),

FOREIGN KEY (Premis\_Cd) REFERENCES Premises(Premis\_Cd),

FOREIGN KEY (Weapon\_Used\_Cd) REFERENCES Weapons(Weapon\_Used\_Cd),

FOREIGN KEY (Status) REFERENCES Status(Status)

);

CREATE TABLE CrimeCodes (

Status INT PRIMARY KEY,

Status\_Desc VARCHAR(255)

);

CREATE TABLE Severance (

Crm\_Cd INT,

Part1-2 INT,

PRIMARY KEY(Crm\_Cd)

);

CREATE TABLE Airbnb (

ID INT,

HOST\_ID INT,

HOST\_NAME VARCHAR(255),

NEIGHBORHOOD VARCHAR(255),

ROOM\_TYPE VARCHAR(50),

PRICE INT,

MINIMUM\_NIGHTS INT,

NUMBER\_OF\_REVIEWS INT,

AVAILABILITY\_365 INT,

AIRBNB\_NAME VARCHAR(255),

STAR FLOAT,

BEDROOMS INT,

BEDS INT,

BATHS FLOAT,

PRIMARY KEY (ID)

);