UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA IN DATA SCIENCE A.A. 2023/2024

RELAZIONE DI PROGETTO

CASO DI STUDIO

CORSO DI GESTIONE DI DATI STRUTTURATI E NON STRUTTURATI

DOCENTI

Prof. Mario Alessandro Bochicchio

Prof. Corrado Loglisci

STUDENTI

Detomaso Giacomo

(e-mail: g.detomaso7@studenti.uniba.it)

Detomaso Gabriele

(e-mail: g.detomaso6@studenti.uniba.it)

Sommario

Gestione dati non strutturati

Obbiettivi

Un imprenditore che opera nel campo dei **B&B**, gestendone uno, con ottimi risultati a Peschici, necessita di **espandere** la propria attività in una grande città estera in modo da avviare l'apertura di una **catena di B&B**. In particolare, è rimasto molto colpito dalle opportunità che la città di **New York** *sembra* offrire e ha messo in evidenza due questioni essenziali:

1. Il luogo più vantaggioso dove aprire un nuovo B&B;
2. Le caratteristiche consigliate (idealmente), intese come servizi e costi, da assegnare al suddetto B&B.

In definitiva nell'ambito di questa analisi di dati non strutturati si vuole:

- Analizzare i prezzi delle case nelle vicinanze dell'area di Central Park, essendo quest'ultimo una dei principali punti di interesse dell'intera città;
- Analizzare quali sono i punti di interesse principali da avere nelle immediate vicinanze di una unità di affitto;
- Analizzare i neighborhood valutando le unità di affitto in essi presenti sulla base di: criminalità, vicinanza a parchi, a fermate di trasporti e a POI individuati sia nella precedente analisi sia in base a specifiche richieste dello stakeholder. L'obiettivo è quello di estrapolare i neighborhood ottimali per l'apertura della rental unit richiesta dallo stakeholder;
- Analizzare i neighborhood estratti individuando le aree migliori all'interno di essi, per poter avviare l'attività preposta dallo stakeholder.

L'obiettivo finale di questa analisi è quello di individuare una precisa area geografica dove lo stakeholder dovrebbe aprire la propria attività.

Fonti dati individuate

Le fonti dati scelte per il progetto sono elencate nelle seguenti tabelle, dove è presente un link per il download e un codice identificativo.

Nome file	Link	Codice
listings.csv	http://data.insideairbnb.com/united-states/ny/new-york-city/2023-11-01/visualisations/listings.csv	D01
stops.csv	https://www.kaggle.com/datasets/monsieurwagner/nyctransit?select=stops.csv	D02
[district_name].csv	https://www.nyc.gov/site/finance/taxes/property-rolling-sales-data.page	D03
NYPD_Arrest_2023.csv	https://www.kaggle.com/datasets/justinpakzad/nypd-arrests-2023-dataset	D04
nyc_borough_boundaries.zip	https://data.cityofnewyork.us/City-Government/2020-Neighborhood-Tabulation-Areas-NTAs-Tabular/9nt8-h7nd	D05
nyc_roads.zip	https://data.cityofnewyork.us/City-Government/NYC-Street-Centerline-CSCL-/exjm-f27b	D06
nyc_parks.zip	https://data.cityofnewyork.us/Recreation/Parks-Properties/enfh-gkve	D07
nyc_bus_stops_shelters.zip	https://data.cityofnewyork.us/Transportation/Bus-Stop-Shelters/qafz-7myz	D08
nyc_points_of_Interest.zip	https://data.cityofnewyork.us/City-Government/Points-Of-Interest/rxuy-2muj	D09

nyc_borough.zip	https://data.cityofnewyork.us/City-Government/Borough-Boundaries/tqmj-j8zm	D10

Tabella 1

Le fonti dati, elencate precedentemente sono descritte sinteticamente nella seguente tabella.

Codice	Descrizione	Formato dati	Numerosità
D01	Dati sulle prenotazioni di AirBnB a NYC, aggiornate al 2023.	Numerico, stringa, spaziale	10^4
D02	Dati sulle ubicazioni di fermate della metro di New York City	Numerico, stringa, spaziale	10^3
D03	Dati sulle case vendute nell'area di NYC nel corso del 2023, correlate con le tasse relative in base al distretto.	Numerico, stringa, spaziale, date	10^4
D04	Dati relativi agli arresti e relativi crimini avvenuti in NYC nel 2023 e relativa ubicazione spaziale	Numerico, stringa, spaziale, date	10^6
D05	Dati relativi ai perimetri dei 5 distretti di NYC	Numerico, stringa, spaziale	5
D06	Aree di tabulazione dei quartieri (NTA) del 2020 per i 5 distretti di NYC.	Numerico, stringa, spaziale	10^2
D07	Dati relativi alla rete stradale di NYC.	Numerico, stringa, spaziale	10^6
D08	Dati relativi ai perimetri dei parchi dei 5 distretti di NYC	Numerico, stringa, spaziale	10^3
D09	Dati relativi ai punti di fermata dei BUS pubblici di NYC	Numerico, stringa, spaziale	10^3
D10	Dati relativi ai punti di interesse per la città dei NYC nei 5 distretti	Numerico, stringa, spaziale	10^5

Tabella 2

Diagramma ER e modello relazionale

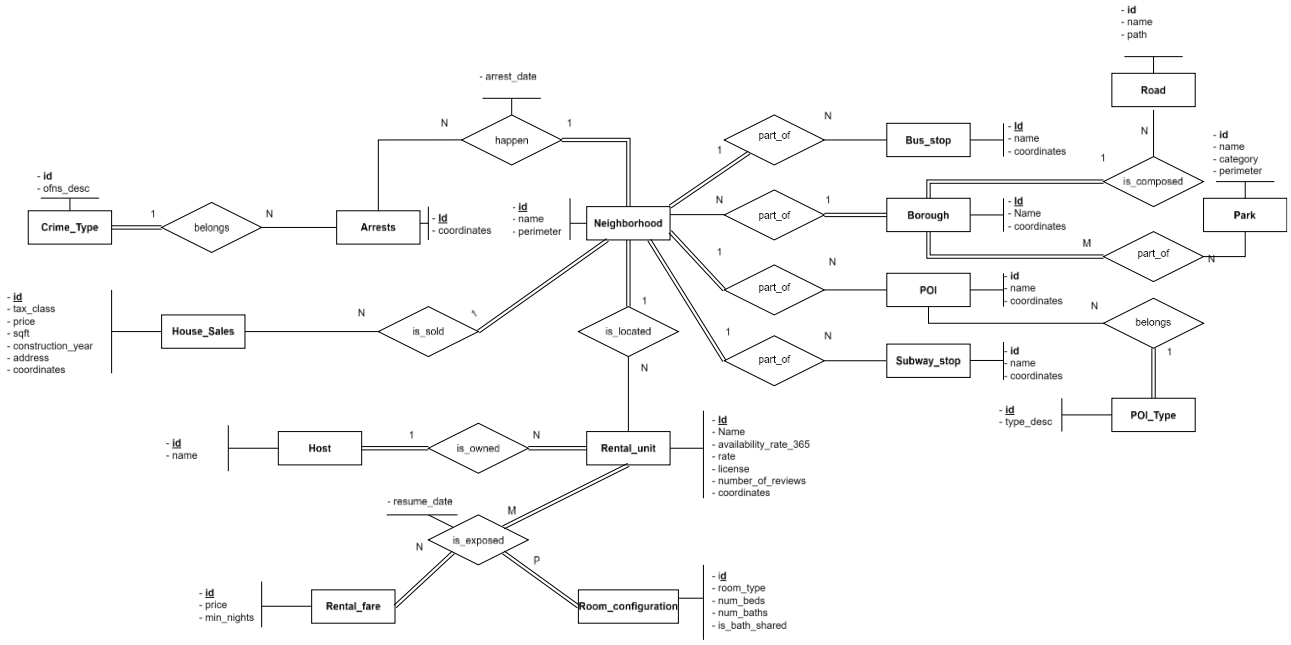


Figura 1

Di seguito è presente il **modello relazionale**. In grassetto sono indicati le **chiavi primarie**, in corsivo con sottolineatura tratteggiata le *chiavi esterne*, inoltre, il nome della chiave esterna in corsivo è lo stesso della tabella a cui il vincolo fa riferimento (ma al singolare e non al plurale).

- **Rental_fares**: (**id**, price, min_nights)*;*
- **Room_configurations**: (**id**, room_type, n_rooms, num_beds, num_baths, is_bath_shared)*;*
- **Rental_units**: (**Id**, name, availability_rate_365, rate, rewiew_number, coordinates, *neighborhood, host*)*;*
- **Rental_resumes**: (***rental_fare***, ***rental_unit***, ***room_configuration***, resume_date);
- **Hosts**: (**id**, min_nights);
- **Neighborhoods**: (**id**, name, perimeter, *borough*);
- **House_sales**: (**id**, tax_class, price, sqft, construction_year, address, coordinates, *neighborhood*)*;*
- **Crimes**: (**id**, ofns_desc);
- **Arrests**: (**id**, coordinates, *crime*, *neighborhood*);
- **Bus_stops**: (**id**, name, coordinates, *neighborhood*);
- **Boroughs**: (**id**, name, perimeter, *positioning*)*;*
- **Positionings**: (***borough***, ***park***);
- **Parks**: (**id**, name, perimeter)*;*
- **Roads**: (**id**, name, path, *borough*);
- **Pois**: (**id**, name, coordinates, *neighborhood, poi_type*);
- **Poi_types**: (**id**, *type_desc*);
- **Subway_stops**: (**id**, name, coordinates, *neighborhood*).

ETL

Di seguito alcune note circa questa sezione:

- Questa sezione include le operazione di DDL e DML effettuate nel corso di questa analisi. In particolare, si rende noto che per le operazioni di DDL le tabelle contenenti dati provenienti da:
  - file CSV sono state create nel file DDL/ddl_csv.sql;
  - file shapefiles sono state create nel file DDL/ddl_shapefiles.sql.
- I codici relativi sono presenti nella sezione appendice.

Fonte dati D01

Nota: la tabella *listings* è una tabella di appoggio per caricare i dati da CSV su database.

Operazione	Descrizione	File di progetto
O01	Estrapolazione di attributi significativi dalla colonna bnb_names usando espressioni regolari. (39k righe elaborate)	*ETL/listings_etl.ipynb*
O02	Inserimento dei dati nella tabella **rental_units** a partire dalla tabella *listings.* (39k righe inserite)	*DML/csv/dml_listings.sql*
O03	Conversione dell'attributo *availability_rate_365* in valore percentuale (39k righe elaborate) e assegnazione del valor medio ove la percentuale è uguale a 0. (13.5k righe elaborate)	*DML/csv/dml_listings.sql*
O04	Inserimento dei dati nella tabella **room_configuration** raggruppando la tabella	*DML/csv/dml_listings.sql*

	listings per *room_type*, *n_beds*, *n_baths*, *is_bath_shared* (40 righe inserite).	
O05	Inserimento dei dati nella tabella **rental_fares** raggruppando la tabella *listings* per price e minimum_nights (3.5k righe inserite).	*DML/csv/dml_listings.sql*
O06	Inserimento dei dati nella tabella **rental_resumes** mappando, a partire dalla tabella *listings* le room_configurations e rental_fares con i rispettivi id (39k righe inserite).	*DML/csv/dml_listings.sql*
O07	1. Separazione ed estrazione degli attributi *room_type ed n_rooms* (39k righe elaborate). 2. Assegnazione del valore 1 alle camere di tipo 'Studio' (12 righe elaborate).	*DML/csv/dml_listings.sql*
O08	Inserimento dei dati nella tabella **hosts** raggruppando i vari host per *id* e *nome dalla tabella listings*. (29k righe inserite).	*DML/csv/dml_listings.sql*

Tabella 3

Fonte dati D02

Nota: la tabella *subway_stops_temp* è una tabella di appoggio per caricare i dati da CSV su database.

Operazione	Descrizione	File di progetto
O01	Eliminazione dell'indice dal file csv (1.5k righe eliminate).	*ETL/subway_stops_etl.ipynb*
O02	Inserimento dei dati nella tabelle **subway_stops**, (1.5k righe inserite)	*DML/csv/dml_subway_stops.sql*
O03	Eliminazione delle tuple che fanno riferimento a fermate Nord e (1k righe eliminate).	*DML/csv/dml_subway_stops.sql*

Tabella 4

Fonte dati D03

Nota: la tabella *house_sales_temp* è una tabella di appoggio per caricare i dati da CSV su database.

Operazione	Descrizione	File di progetto
O01	Standardizzazione degli indirizzi delle case (73.2k righe elaborate).	*ETL/rolling_sales.ipynb*
O02	Geocoding degli indirizzi (73.2k righe elaborate).	*ETL/rolling_sales.ipynb*
O03	Inserimento dei dati nella tabella **house** a partire dalla tabella *house_sales_temp*, escludendo le tuple dove il valore relativo ai m^2 è mancante (39.3k righe inserite).	DML/csv/dml_house_sales.sql
O04	Standardizzazione a zero (passaggio di proprietà) per i prezzi delle case con valore <= 10 (17.3k righe elaborate).	DML/csv/dml_house_sales.sql

Tabella 5

Fonte dati D04

Nota: la tabella *nypd_arrests* è una tabella di appoggio per caricare i dati da CSV su database.

Operazione	Descrizione	File di progetto
O01	Formattazione della data con un tipo compatibile con SQL (170k righe elaborate).	*ETL/ NYPD_Arrest_2023_etl.ipynb*

O02	Inserimento dei dati nella tabella **crimes** a partire dalla tabella *nypd_arrests*	*DML\csv \dml_nypd_arrests.sql*
O03	Inserimento dei dati nella tabella **arrests** a partire tabella *nypd_arrests*, mappando la descrizione (univoca) dell'arresto con l'id del tipo del crimine (valorizzando quindi la chiave esterna)	*DML\csv \dml_nypd_arrests.sql*

Tabella 6

Fonte dati D05

Nota: *boroughs_temp* è una tabella di appoggio per caricare i dati da shapefile su database.

Operazione	Descrizione	File di progetto
O01	Inserimento dati nella tabella **boroughs**, a partire dalla tabella *boroughs_temp* (5 righe inserite).	*DML\shapefiles\dml_borough.sql*
O02	Validazione delle geometrie tramite la funzione *ST_MakeValid*	*DML\shapefiles\dml_borough.sql*
O03	Mapping del numero del borough con il suo codice letterale (5 righe elaborate).	*DML\shapefiles\dml_borough.sql*

Tabella 7

Fonte dati D06

Nota: *neighborhoods_temp* è una tabella di appoggio per caricare i dati da shapefile su database.

Operazione	Descrizione	File di progetto
O01	Inserimento dei dati nella tabelle **neighborhoods** a partire dalla tabella *neighborhoods_temp*. (262 righe inserite).	*DML\shapefiles\dml_neighborhood.sql*
O02	Validazione delle geometrie tramite la funzione *ST_MakeValid*	*DML\shapefiles\dml_neighborhood.sql*
O03	Mapping del nome del borough con il codice letterale (262 righe elaborate).	*DML\shapefiles\dml_neighborhood.sql*

Tabella 8

Fonte dati D07

Nota: *roads_temp* è una tabella di appoggio per caricare i dati da shapefile su database.

Operazione	Descrizione	File di progetto
O01	Inserimento dei dati nella tabella **roads** a partire dalla tabella *roads_temp*. (121.5k righe inserite).	*DML/shapefiles/dml_roads.sql*
O02	Validazione delle geometrie tramite la funzione *ST_MakeValid*	*DML/shapefiles/dml_roads.sql*
O03	Mapping del numero del borough con il suo codice letterale (121.5k righe elaborate).	*DML/shapefiles/dml_roads.sql*
O04	Mapping dello status number delle strade con la descrizione letterale (121.5k righe elaborate).	*DML/shapefiles/dml_roads.sql*

Tabella 9

Fonte dati D08

Nota: *parks_temp* è una tabella di appoggio per caricare i dati da shapefile su database.

Operazione	Descrizione	File di progetto
O01	Inserimento dei dati nella tabella **parks**, a partire dalla tabella *parks_temp* (2k righe inserite)	*DML/shapefiles/dml_parks.sql*
O02	Validazione delle geometrie tramite la funzione *ST_MakeValid*	*DML/shapefiles/dml_parks.sql*
O03	Eliminazione delle tuple con codici che non rappresentano parchi (200 righe eliminate).	*DML/shapefiles/dml_parks.sql*
O04	Inserimento dei dati nella tabella **positionings** intersecando i perimetri del parco con quelli dei borough, un parco infatti può estendersi anche in più borough.	*DML/shapefiles/dml_parks.sql*

Tabella 10

Fonte dati D09

Nota: *bus_stops_temp* è una tabella di appoggio per caricare i dati da shapefile su database.

Operazione	Descrizione	File di progetto
O01	Inserimento dei dai nella tabella **bus_stops**, a partire dalla tabella *bus_stops_temp*. (3.3k righe inserite).	*DML\shapefiles\dml_bus_stops.sql*
O02	Validazione delle geometrie tramite la funzione *ST_MakeValid*	*DML\shapefiles\dml_bus_stops.sql*
O03	Individuazione del borough di appartenenza utilizzando la funzione ST_Within	*DML\shapefiles\dml_bus_stops.sql*

Tabella 11

Fonte dati D10

Nota: *poi_temp* è una tabella di appoggio per caricare i dati da shapefile su database.

Operazione	Descrizione	File di progetto
O01	Inserimento dei dai nella tabella **poi_types**, selezionando in maniera distinta tutti i tipi di facilities (13 righe inserite).	*DML\shapefiles\dml_POI.sql*
O02	Inserimento dati nella tabella **poi**, a partire dalla tabella *poi_temp* (15.6k righe inserite).	*DML\shapefiles\dml_POI.sql*
O03	Eliminazione di particolari tipi di poi dalla tabella poi	*DML\shapefiles\dml_POI.sql*
O02	Validazione delle geometrie tramite la funzione *ST_MakeValid*	*DML\shapefiles\dml_POI.sql*
O04	Mapping della descrizione del poi type con il relativo codice nella tabella poi	*DML\shapefiles\dml_POI.sql*
O05	Sostituzione delle colonna domain della tabella poi con la corrispondente descrizione	*DML\shapefiles\dml_POI.sql*

Tabella 12

Funzioni applicate sulle tabelle

Operazione	Descrizione	File di progetto
O01	Conversione delle coordinate in una geometria POINT sfruttando *ST_MakePoint* per le tabelle rental_units, subway_stops, house_sales e arrests.	DML\csv_tables\dml_function_make_point.sql
O02	Individuazione del neighborhood tramite la funzione	DML\csv_tables\dml_function_find_neighborhood.sql

	find_neighborhood per le tabelle rental_units, subway_stops, house_sales, arrests e poi	

Tabella 13

Nota: le constraints sono state definite, al termine del processo di DML nel file DDL/ddl_constraints.sql.

Queries di analisi dei dati

Di seguito è riportata una descrizione e il relativo codice delle query di analisi dati effettuate.

Q1

Lo stakeholder, come prima esplorazione delle possibilità offerte dalla città di New York, è interessato a valutare come il prezzo delle case in città vari a seconda della vicinanza a un grosso punto di interesse. La scelta è ricaduta su Central Park.

In particolare, ha richiesto di visualizzare il prezzo medio delle case vendute in un intorno di 700 metri dal parco, considerando però solo quelle ubicate su strade che conducono direttamente al punto di interesse. Tale prezzo deve essere confrontato con il prezzo medio delle case vendute a Manhattan*, escludendo però quelle ubicate entro 700 metri da Central Park*.

Nota: Gli appartamenti presi in considerazione sono quelli che hanno una class tax uguale ad 1 o 2 [1] in quanto 3 e 4 indicano strutture commerciali o fabbriche (non destinate all'uso ricercato dallo stakeholder). Inoltre, non sono state considerate case con prezzo uguale a zero, in quanto indicanti un semplice passaggio di proprietà senza esborso di denaro.

Q2

Lo stakeholder ha richiesto un'analisi più approfondita, atta a individuare quali sono i POI più comuni che in un raggio di 500 metri dal POI stesso presentano più unità di affitto significative (aventi un rate compreso tra 3.5 e 5 con un numero minimo di recensioni di 50).

Q3

Lo stakeholder è interessato a conoscere quali sono i neighborhood che presentano una alta densità di unità di affitto intorno alle quali, in un raggio di 100 metri, sono avvenuti il minor numero di arresti. La lista di neighborhood con le relative percentuali di densità di BnB deve essere ordinata in ordine decrescente.

Q4

Lo stakeholder è interessato a conoscere quali sono i neighborhood che presentano una alto numero di unità di affitto intorno alle quali, in un raggio di 700 metri, sono presenti almeno due parchi . La lista di neighborhood con i relativi conteggi di BnB deve essere ordinata in ordine decrescente.

Q5

Lo stakeholder è interessato a conoscere quali sono i neighborhood che presentano un alto numero di unità di affitto intorno alle quali, in un raggio di 1 chilometro, sono presenti almeno 15 fermate di trasporto pubblico, suddivise in: autobus e metropolitana . La lista di neighborhood con i relativi conteggi di BnB deve essere ordinata in ordine decrescente.

Q6

Seguendo i risultati della Q2 circa i POI significativi intorno ai quali sono presenti più attività, lo stakeholder ha richiesto di elencare i neighborhood che presentano un alto numero di unità di affitto intorno alle quale, in un raggio di 500 metri, sono presenti almeno 20 POI per ognuna delle seguenti categorie indicate dallo stakeholder stesso:

- **Daily routine POI**: piscine, supermercati, posti in cui praticare sport e ponti (utili per spostarsi da un borough ad un altro;
- **Free time POI:** zoo, librerie, musei, cinema/teatri, ristoranti, chiese;
- **Emergency POI**: ospedali e centri di assistenza.

Nota: Come possibile notare ai POI individuati in Q2 sono stati aggiunti anche supermercati e ristoranti, ritenuti dallo stakeholder fondamentali per la propria attività.

Query intermedia

Al fine di individuare **i migliori neighborhood**, i dati ottenuti dalle query Q3 a Q6 sono stati incrociati e filtrati secondo determinati criteri forniti dallo stakeholder stesso:

- Densità di bnb sicuri per neighborhood compresa tra il 10% e l'80%;
- Numerosità compresa tra 5 e 50 di BnB con almeno 2 parchi vicino;
- Numerosità compresa tra 10 e 75 di BnB con almeno 15 fermate di trasporto vicino;
- Numerosità inferiore o uguale a 100 di BnB con almeno 20 POI appartenenti all'insieme daily routine che free time. Per la categoria emergency la numerosità non deve superare le 150 unità

Il tutto è ordinato secondo il neighborhood con densità di BnB sicuri maggiore.

Nota: questa operazione non presenta uso di dati spaziali, ma usufruisce di viste ottenute dal query precedenti. I criteri indicati, sono stati forniti dallo stakeholder una volta visualizzati i risultati delle query indicate. Tale operazione è stata inserita in quanto risulta **fondamentale** per le prossime due query che rappresentano una conclusione di questa analisi.

Q7

Lo stakeholder ha richiesto di suddividere in un certo numero di aree più piccole, i neighborhood individuati nella "query intermedia" al fine di individuare quale neighborhood prediligere per l'apertura, individuando al contempo una precisa zona geografica all'interno del neighborhood che presenti un prezzo medio di venduta delle case, minore del prezzo medio totale delle case vendute nell'neighborhood.

L'obiettivo è quindi produrre una lista di sotto-neighborhood (aree individuate all'interno dei neighborhood stessi), con le seguenti voci:

- Id assegnato al sotto-neighborhood concatenato al neighborhood di appartenenza ed al codice letterale del borough;
- Prezzo medio delle case vendute all'interno del sotto-neighborhood;
- Il perimetro del sotto-neighborhood, al fine di mostrare allo stakeholder su una mappa l'area selezionata.

Ovviamente sono stati considerati solo i sotto-neighborhoods il cui prezzo medio delle case è inferiore al prezzo medio dell'intero vicinato.

I risultati inoltre sono ordinati in maniera ascendete, per ogni neighborhood, considerando i prezzi medi delle sotto aree selezionate.

Q8

Una volta stabilito attraverso la visualizzazione di Q7 quale sotto area considerare per l'apertura della propria unità di affitto a New York, lo stakeholder ha richiesto di visualizzare all'interno di essa il numero di fermate di autobus e metropolitane presenti in un raggio di un chilometro dal centro, analizzando al contempo prezzi medi e disponibilità annua media delle unità di affitto.

Nota: nella presentazione dei risultati le scelte effettuate dallo stakeholder saranno rese esplicite. In questa fase si è ritenuto necessario descrivere la sola operazione di query considerata.

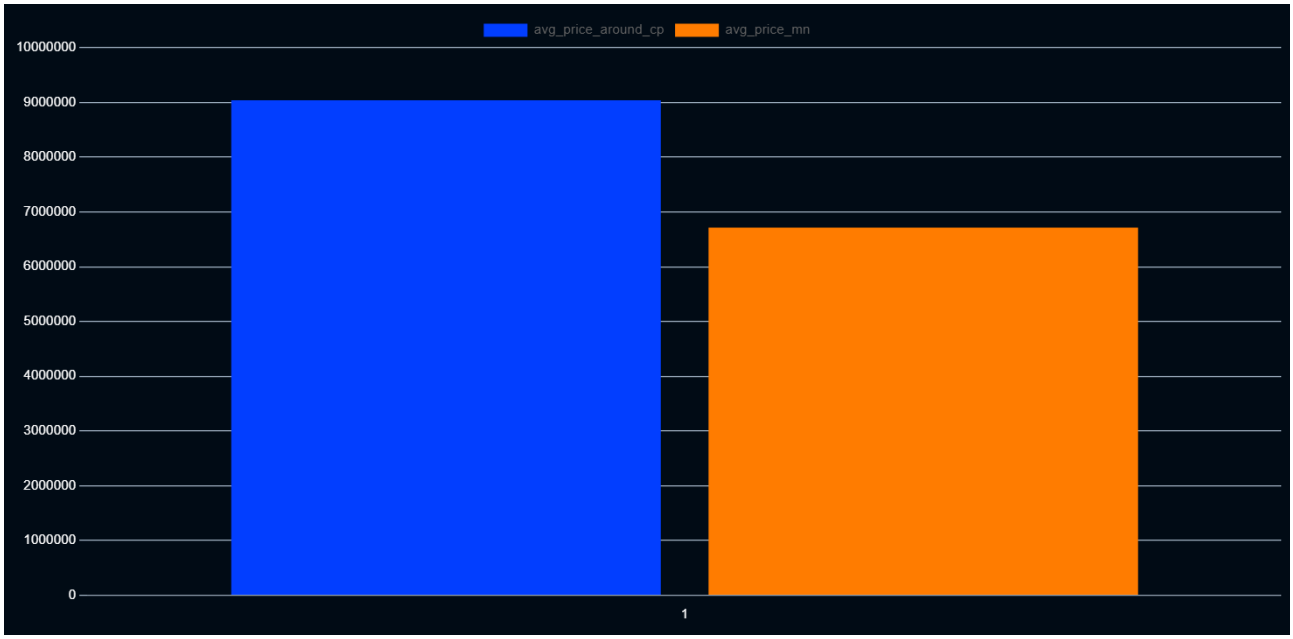
Q Final

Lo stakeholder è interessato a valutare, senza considerare spese successive, quanti mesi sarebbero necessari, lavorando il 70% dell'anno, per coprire l'esborso iniziale per l'acquisto di una casa da utilizzare come unità di affitto. Si vuole analizzare questa situazione analizzando i prezzi medi, sia in un area di un chilometro dal centro della zona prescelta in Q7, sia considerando i prezzi medi globali, per la fascia excellent (Q2 strutturata) e le relative disponibilità medie annue.

Nota: questa query combina risultati ottenuti in entrambe le analisi.

Presentazione dei risultati

Presentazione Q1



Presentazione 1

In questo grafico è possibile notare come il prezzo medio delle case in un intorno di 700 metri da Central park, e collegate **direttamente ad esso**, è circa 2 milioni di dollari più rispetto alla media delle case vendute nell'intero Manhattan (escludendo quelle utilizzate nel calcolo precedente).

Lo stakeholder ha concluso che POI importanti come Central Park sono un fattore discriminanti sull'aumento dei prezzi delle case.

Presentazione Q2

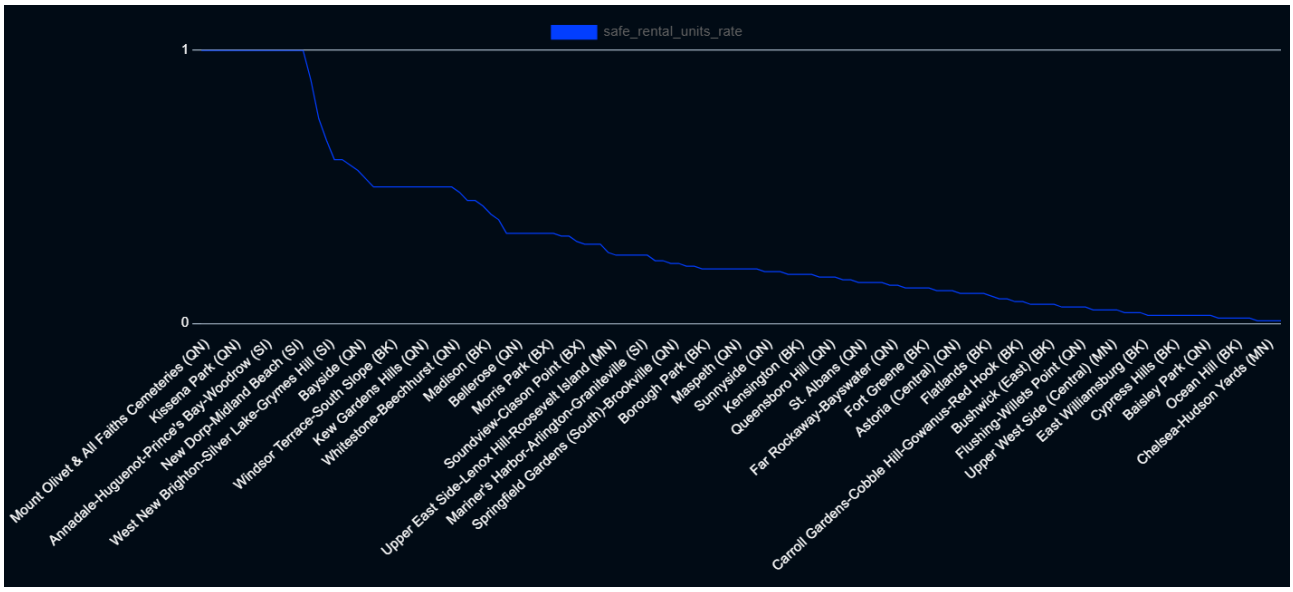
	domain character varying (50)
1	Hotel/Motel
2	Bridge
3	Theater/Concert Hall
4	Day Care Center
5	Hospital
6	Church

Presentazione 2

Dall'analisi effettuata i POI presentati in figura risultano avere un alta numerosità di unità d'affito in un raggio di 500 metri. Da notare l'importanza dei ponti in quanto strutture che permettono l'attraversamento dei borough di New York.

Contrariamente a quanto si potrebbe pensare in questa tabella, ristoranti e supermercati, non occupano le prime posizioni. Questo è principalmente dovuto ad un bassa densità di BnB in un raggio di 500 metri da queste categorie di poi e non è riconducibile ad una scarsità di importanza per le attività di affitto.

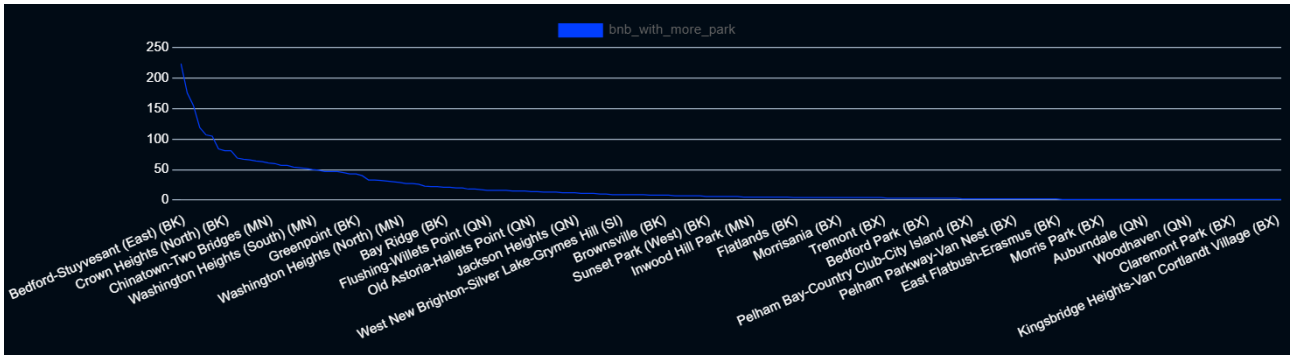
Presentazione Q3



Presentazione 3

In questo grafico si può notare, per ogni neighborhood, quali sono le percentuali di BnB con il minor numero di crimi in un area di raggio 100 metri. Seguendo le analisi fatte per la parte strutturata ,dalla seguente curva si evince che Queens e Staten Island sono i borough con un numero maggiore di BnB sicuri.

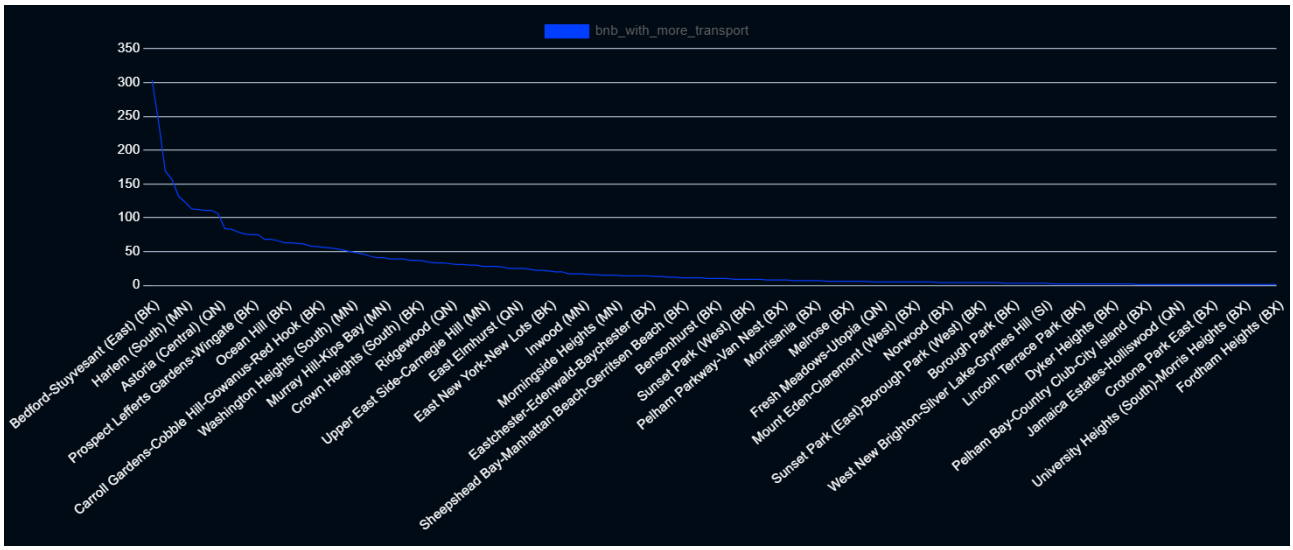
Presentazione Q4



Presentazione 4

In questo grafico è possibile notare il numero di bnb con almeno 2 parchi per i vari neighborhood. È possibile concludere quanto la presenza di bnb che rispettano questa condizione è concentrata maggiormente A Brooklyn e Manhattan, due dei borough più turistici della "grande mela".

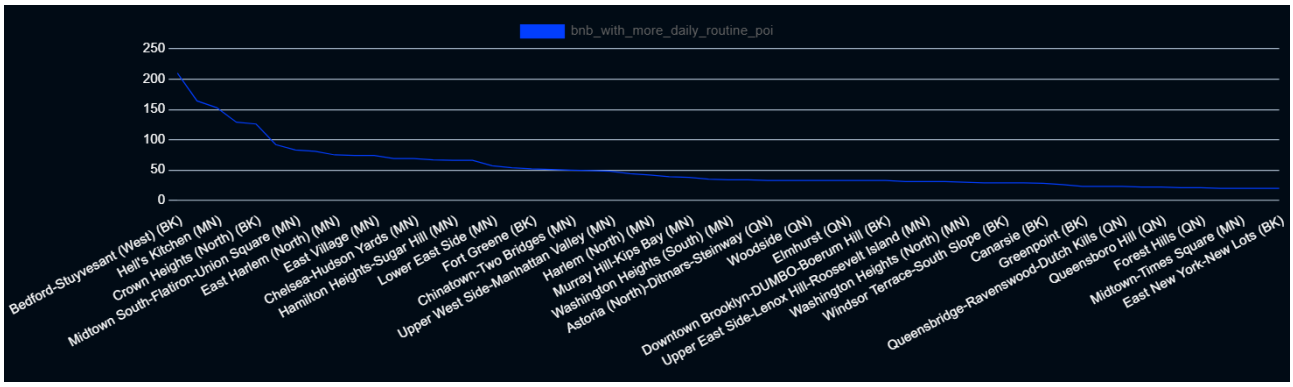
Presentazione Q5



Presentazione 5

Da questo grafico si può notare come l'elevato numero di BnB con una densità di fermate per il trasporto pubblico, in un area di 1km, abbia un distacco abbastanza elevato per borough come Manhattan e Brooklyn, di.

Presentazione Q6 – 1



Presentazione 6

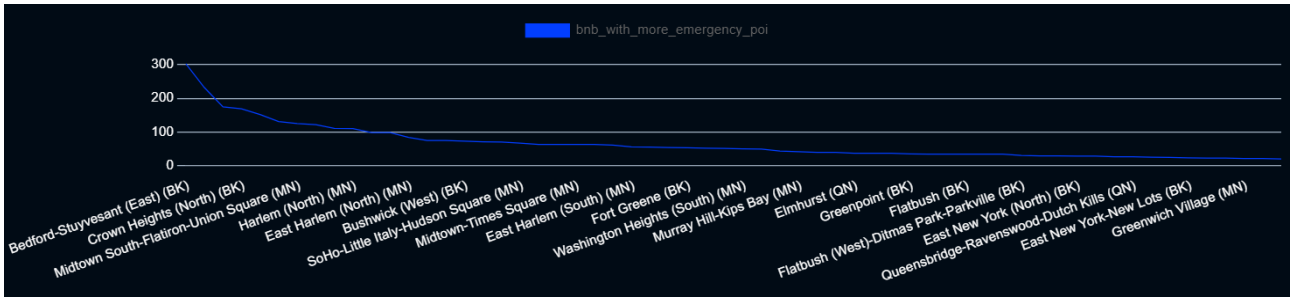
Il seguente grafico mostra che i neighborhood con il maggior numero di BnB con una densità alta, in un intorno di 500 metri, di *daily routine poi* sono collocati in Brooklyn e Manhattan.

Presentazione Q6 – 2

Presentazione 7

Il seguente grafico mostra che i neighborhood con il maggior numero di BnB con una densità alta, in un intorno di 500 metri, di *free time poi* sono collocati in Brooklyn Queens e Manhattan.

Presentazione Q6 – 3



Presentazione 8

Il seguente grafico mostra che i neighborhood con il maggior numero di BnB con una densità alta, in un intorno di 500 metri, di emergency *poi* sono collocati in Brooklyn Queens e Manhattan.

Presentazione aree individuate in query intermedia

neighborhood text	safe_rental_units_rate numeric	bnb_with_more_park bigint	bnb_with_more_transport bigint	bnb_with_more_daily_routine_poi bigint	bnb_with_more_free_time_poi bigint	bnb_with_more_emergency_poi bigint
Flatbush (West)-Ditmas Park-Parkville (BK)	0.32	6	28	33	31	30
Upper East Side-Lenox Hill-Roosevelt Island (MN)	0.26	13	43	31	43	43
Prospect Heights (BK)	0.21	31	34	31	34	34
Jackson Heights (QN)	0.15	12	48	34	32	34

Presentazione 9

Le query precedenti hanno permesso allo stakeholder di valutare **qualitativamente** la presenza di B&B aventi determinate caratteristiche tra i vari neighborhood di NYC. Al fine di scegliere quello ideale è stato necessario **incrociare e filtrare** i dati ottenuto dalle queries precedenti, in modo tale da ottenere un sottoinsieme di essi, ristretto e significativo.

Per filtrarli, lo stakeholder ci ha comunicato, sulla base dei risultati consultati precedentemente diversi criteri (elencati in fase di descrizione della query). I medesimi, sono stati decisi cercando di mediare la numerosità di unità di affitto, rispondenti a determinate caratteristiche, in un neighborhood con l'eccessiva competitività derivate da quest'ultima.

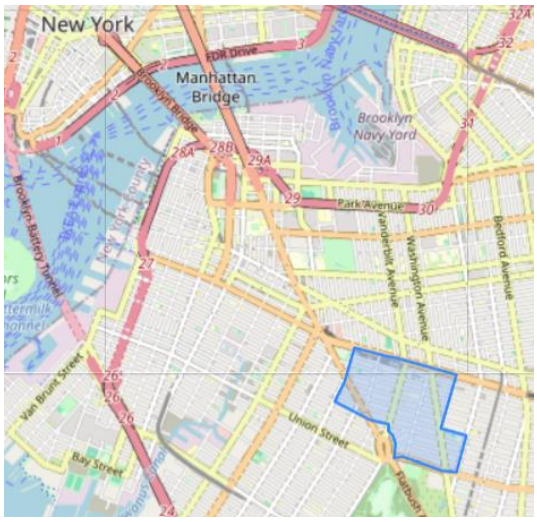
Ad esempio, considerato il tasso di sicurezza calcolato in Q3, si è deciso di non considerare i neighborhood con un tasso di sicurezza troppo elevato, in quanto alternativamente:

- sono già stanziate sul territorio, numerose realtà che rispondono alla caratteristica di sicurezza (alta competitività)
- sono neighborhood all'interno borough che presentano una scarsità di attrattiva in termini di POI (vedasi Staten Island).

Tale ragionamento è applicabile anche per i POI, parchi e mezzi di trasporto. Incrociare questi dati, ha permesso quindi di individuare 4 possibili aree (su oltre 150 selezionate), vantaggiose rispettivamente alle richieste dello stakeholder:

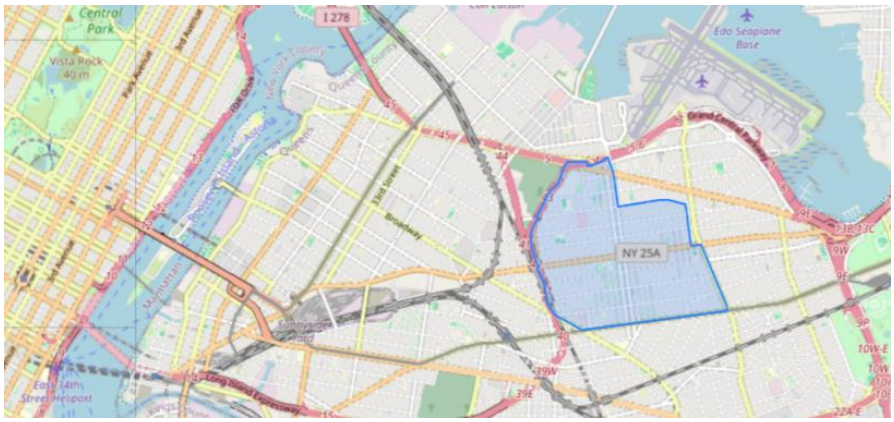
- Upper East Side-Lenox Hill-Roosevelt Island (MN);
- Prospect Heights (BK)
- Flatbush (West)-Ditmas Park-Parkville (BK);
- Jackson Heights (QN).

[Upper East Side-Lenox Hill-Roosevelt Island (MN)] Presentazione 10



[Prospect Heights (BK)] Presentazione 11

[Flatbush (West)-Ditmas Park-Parkville (BK)] Presentazione 12

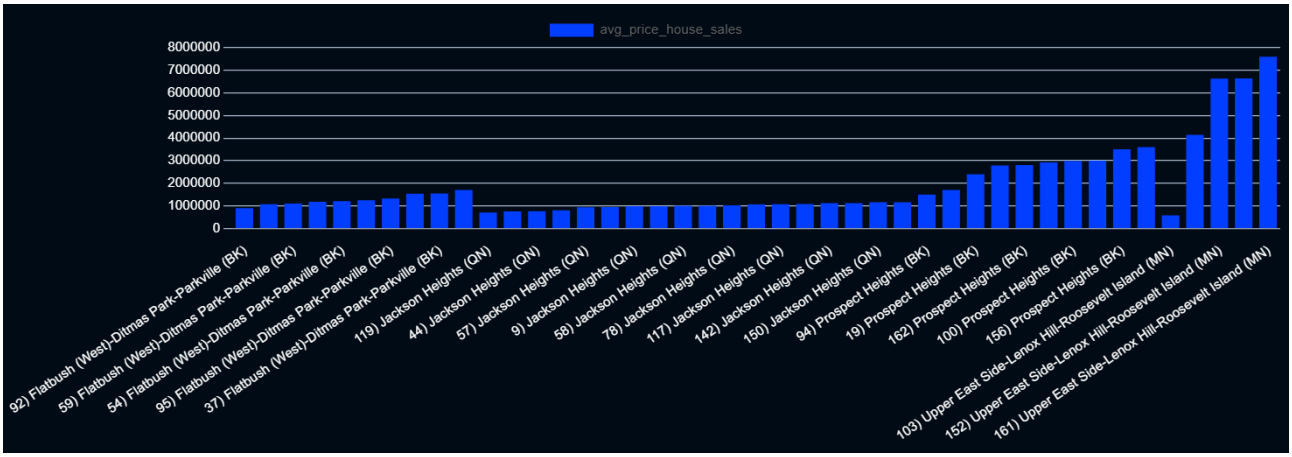


[Jackson Heights (QN)] Presentazione 13

Il fatto che i borough, nei quali sono stati individuati i neighborhood, siano proprio Manhattan, Brooklyn e Queens, non deve stupire, in quanto nel corso dell'analisi strutturata era chiaramente emerso che in termini di POI e tasso di arresti tali borough rappresentavano la scelta più indicata per lo stakeholder (escludendo a priori Bronx e Staten Island).

A questo punto, avendo chiari i punti di forza di questi neighborhood la scelta passa dal prezzo delle case in ciascuno di essi

Presentazione Q7



Presentazione 14

Una volta individuati i 4 neighborhood candidati è stata fatta un analisi dettagliata sui prezzi medi delle case da usare come discriminante per la decisione del neighborhood in cui aprire il nuovo BnB. Nello specifico:

1. Ogni neighborhood è stato suddiviso in un numero variabile di sotto aree dipendenti dalla dimensione.
2. Ad ogni area è stato assegnato un identificativo ed è stato calcolato il prezzo medio di vendita delle case per ogni sotto area.

Le sotto aree presenti nel grafico sono tutte e sole quelle che riportano un prezzo medio inferiore alla media del neighborhood a cui sono collocate. Da tale analisi è risultato che la sotto area migliore secondo il criterio espresso dallo stakeholder è in **Upper East Side-Lenox Hill-Roosvelt Island** identificato dal numero 103 e mostrato nella seguente immagine.

Upper East Side-Lenox Hill-Roosevel Island

Tale risultato è ottimale per lo stakeholder in quanto svariate volte ha espresso la preferenza di aprire un BnB in zona Manhattan nelle vicinanze di Central Park, come dimostrato dalla prima query richiesta nella quale era richiesto di analizzare il prezzo delle case nell'intorno di Central Park.

Presentazione Q8

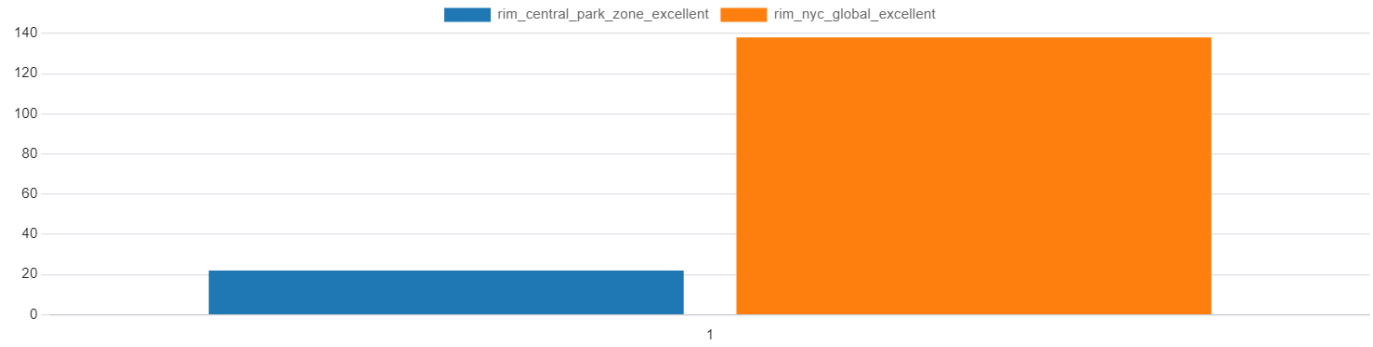
vis_id text	n_subway_stops bigint	n_bus_stops bigint	avg_price_ex_band numeric	avg_availability_rate_ex_band numeric	avg_price_house_sales numeric
103) Upper East Side-Lenox Hill-Roosevelt Island (M...	6	38	1251	0.65	585000.00

Presentazione 15

Lo scopo della Q8 era quello di analizzare la zona scelta (103) in termini di concentrazione di trasporto pubblico, prezzo medio B&B presenti e tasso medio di disponibilità annua per prenotazioni relativi ad unità di affitto di fascia excellent (definita nell'analisi strutturata come rate compreso tra 4.8 e 5 con almeno 50 recensioni). Tale analisi è stata fatta considerato un area di raggio 1000 metri dal centro. Per lo stakeholder la numerosità di fermata in un area ristretta ha rappresentato una ulteriore conferma sulla bontà della scelta.

La decisione di analizzare questi elementi a 1km dal centro è stata effettuata dopo aver visualizzato l'estensione della sotto area con id 103 che risulta essere **di 274m** di diametro. Inoltre, è stato fatto notare allo stakeholder che la distanza da Central Park è di soli **1.03km**. Tale area, protagonista della Q1 (nella quale sono stati analizzati i prezzi delle case immediatamente confinanti), era infatti una delle più desiderate dallo stakeholder (prima di approfondire i pro e i contro con questa analisi) per l'apertura della proprio unità di affitto.

Presentazione Q final



Presentazione 16

Dopo aver individuato l'area di NYC più indicata per aprire un BnB, lo stakeholder ha espresso un'ultima richiesta, ovvero analizzare qual è il tempo stimato per recuperare l'investimento iniziale (l'acquisto dell'appartamento). Per rispondere a tale richiesta è stato necessario stabilire:

- Il prezzo del BnB per notte;
- L'occupazione annua del BnB (presa di default al 70%).

Lo stakeholder a sua volta (non avendo un criterio chiaro per la scelta di tale prezzo) ha richiesto di prevedere il tempo di recupero dell'investimento valutando:

- Il prezzo medio dei bnb a partire dal centro della sotto area '103 Upper East Side-Lenox Hill-Roosevel Island (MN)', con un raggio di un chilometro (Q8 non strutturata);
- Il prezzo medio dei bnb di fascia 'Excellent' dislocati in tutta NYC (Q2 strutturata).

Fatta tale premessa è ora possibile dedurre dal grafico che il prezzo nella sotto area, essendo consistentemente maggiore, comporterà un tempo (espresso nel grafico in mesi) di recupero dell'investimento inziale sensibilmente più rapido (circa 2 anni con 1251.00 $ a notte) rispetto a quello ottenuto utilizzando il prezzo medio della fascia 'Excellent' (circa 12 anni con 199.14 $ a notte).

Nota: Gli anni risultanti sono una stima che tiene conto solo della spesa iniziale e ignora tutte le spese successive all'acquisto (ristrutturazione appartamento, stipendio dipendenti, altri servizi). Inoltre questa query si propone di essere un primo passo, verso una vera e una propria stima dei prezzi ai quali offrire i propri servizi (questione non pertinente in queste analisi in base agli obiettivi descritti).

Matrice delle responsabilità

OPERAZIONI	GABRIELE DETOMASO	GIACOMO DETOMASO
Definizione degli obiettivi dell'analisi	X	X
Definizione fonti dati	X	X
Descrizione esaustiva delle fonti dati	X	
Schema E/R		X
Modello logico	X	
ETL fonti dati: D01, D02, D09, D10,	X	
ETL fonti dati: da D03 a D08		X
ETL Funzione Applicate alle tabelle	X	X
Definizione di vincoli di integrità referenziale		X
Query di analisi: Q2, Q3, Q6	X	
Query di analisi: Q4, Q5, Q8		X
Query di analisi: Q1, Q7, finale	X	X
Presentazione dei risultati	X	X

Appendici

In questa appendice sono riportati **solo** pezzi di codice **significativo**.

Codici per ETL

DDL per file CSV

```sql
-- This table will load CSV data about bnb listings
CREATE TABLE listings (
    id BIGINT,
    name VARCHAR(255),
    host_id INT,
    host_name VARCHAR(255),
    neighbourhood_group VARCHAR(255),
    neighbourhood VARCHAR(255),
    latitude FLOAT,
    longitude FLOAT,
    bnb_type VARCHAR(255),
    price INT,
    minimum_nights INT,
    number_of_reviews INT,
    last_review DATE,
    reviews_per_month FLOAT,
    calculated_host_listings_count INT,
    availability_365 DECIMAL,
    number_of_reviews_ltm INT,
    license VARCHAR(255),
    rate DECIMAL,
    room_type VARCHAR(255),
    n_beds INTEGER,
    n_baths INTEGER,
    is_bath_shared BOOLEAN
);
```

Blocco di codice 1

Il codice riportato qui sopra indica la creazione della tabella listings, la quale è stata importata da un file .CSV.

Per tutti i file .CSV è stato utilizzato lo stesso modus operandi inquanto è stata creata una tabella con lo stesso nome del file da cui sono state successivamente estratte le colonne utili riportate all'interno dello schema ER.

DDL file csv

Questi blocchi di codice mostrano il DDL per le tabelle create dalle fonti dati CSV.

```sql
-- This table will contain every Airbnb rental unit in NYC
CREATE TABLE IF NOT EXISTS rental_units (
    id BIGINT,
    name VARCHAR(50) NOT NULL,
    availability_rate_365 DECIMAL,
    rate DECIMAL,
    number_of_reviews INTEGER,
    latitude DECIMAL,
    longitude DECIMAL,
    host INTEGER NOT NULL,
    license BOOLEAN DEFAULT FALSE,
    neighborhood INTEGER,
    coordinates geometry(Point)
);

-- This table will contain every host for the renal units
CREATE TABLE IF NOT EXISTS hosts (
    id INTEGER,
    name VARCHAR(50)
);
```

Blocco di codice 2

```sql
-- This table will contain every possible room configuration for the rental units
CREATE TABLE IF NOT EXISTS room_configurations (
    id SERIAL,
    room_type VARCHAR(20),
    n_beds INTEGER,
    n_baths INTEGER,
    n_rooms INTEGER,
    is_bath_shared BOOLEAN
);

-- This table will contain every possible rental fare for the rental units
CREATE TABLE IF NOT EXISTS rental_fares (
    id SERIAL,
    minimum_nights INTEGER NOT NULL,
    price DECIMAL NOT NULL
);

-- This table resume the listing of a rental unit
CREATE TABLE IF NOT EXISTS rental_resumes (
    id SERIAL,
    rental_unit BIGINT NOT NULL,
    room_configuration INTEGER NOT NULL,
    rental_fare INTEGER NOT NULL,
    resume_date DATE
);
```

Blocco di codice 3

```sql
-- This table will contain the arrests made in NYC
CREATE TABLE IF NOT EXISTS arrests (
    id INTEGER,
    arrest_date DATE NOT NULL,
    latitude DECIMAL,
    longitude DECIMAL,
    neighborhood INTEGER,
    coordinates geometry(Point),
    crime INTEGER NOT NULL
);

-- This table will contain NYC subway stops
CREATE TABLE IF NOT EXISTS subway_stops (
    id VARCHAR(4),
    name VARCHAR(50) NOT NULL,
    latitude DECIMAL,
    longitude DECIMAL,
    neighborhood INTEGER,
    coordinates geometry(Point)
 );
```

Blocco di codice 4

```sql
-- This table will contain the house sales in NYC during 2023
CREATE TABLE IF NOT EXISTS house_sales (
    id SERIAL,
    tax_class CHAR(2),
    sqft DECIMAL NOT NULL,
    price DECIMAL,
    construction_year INTEGER,
    address VARCHAR(100),
    latitude DECIMAL,
    longitude DECIMAL,
    neighborhood INTEGER,
    coordinates geometry(Point)
);
```

Blocco di codice 5

DDL Shapefiles

Questi blocchi di codice mostrano il DDL per le tabelle create dalle fonti dati CSV.

```
-- This table will contain every borough in NYC
CREATE TABLE IF NOT EXISTS boroughs (
    id CHAR(2),
    name VARCHAR(50),
    perimeter geometry(MultiPolygon, 4326)
);

-- This table will contain every neighborhoods
CREATE TABLE IF NOT EXISTS neighborhoods (
    id SERIAL,
    name VARCHAR(100),
    borough CHAR(2),
    perimeter geometry(MultiPolygon, 4326)
);

-- This tables will contain every POIs' types
CREATE TABLE IF NOT EXISTS poi_types (
    id INTEGER,
    type_desc VARCHAR(50)
);

-- This tables will contain every POI
CREATE TABLE IF NOT EXISTS poi (
    id SERIAL,
    name VARCHAR(100),
    domain VARCHAR(50),
    poi_type INTEGER,
    neighborhood INTEGER,
    coordinates geometry(Point, 4326)
);
```

Blocco di codice 6

```
-- This table will contain every bus stop
CREATE TABLE IF NOT EXISTS bus_stops (
    id SERIAL,
    name VARCHAR(50),
    corner VARCHAR(100),
    neighborhood INTEGER,
    coordinates geometry(Point, 4326)
);

-- This table will contain information about NYC roads
CREATE TABLE IF NOT EXISTS roads (
    id SERIAL,
    traffic_direction CHAR(2),
    name VARCHAR(50),
    status VARCHAR(20),
    borough CHAR(2),
    path geometry(LineString, 4326)
);

CREATE INDEX roads_idx
    ON roads
    USING GIST(path);
```

Blocco di codice 7

```sql
-- This table will contain every NYC park
CREATE TABLE IF NOT EXISTS parks (
    id SERIAL,
    name VARCHAR(100),
    category VARCHAR(50),
    boroughs CHAR(2)[],
    perimeter geometry(MultiPolygon, 4326)
);

CREATE INDEX parks_idx
    ON parks
    USING GIST(perimeter);

-- This table will contain the boroughs where each park belongs
CREATE TABLE IF NOT EXISTS positionings (
    borough CHAR(2),
    park INTEGER
);
```

Blocco di codice 8

Fonte dati D01

```sql
-- 1)
INSERT INTO rental_units (id, name, availability_rate_365, rate,
                          number_of_reviews, latitude, longitude,
                          host, license)
    SELECT id, name, availability_365, rate,
           number_of_reviews, latitude, longitude,
           host_id, set_licence(license)
    FROM listings;

-- 2)
INSERT INTO room_configurations(room_type, n_beds, n_baths, is_bath_shared)
    SELECT room_type, n_beds, n_baths, is_bath_shared
    FROM listings
    GROUP BY room_type, n_beds, n_baths, is_bath_shared;

-- 3)
INSERT INTO hosts (id, name)
    SELECT host_id AS id, host_name AS name
    FROM listings
    WHERE name IS NOT NULL
    GROUP BY host_id, host_name;

-- 4)
INSERT INTO rental_fares(price, minimum_nights)
    SELECT price, minimum_nights
    FROM listings
    GROUP BY price, minimum_nights;
```

Blocco di codice 9

Popolamento delle tabelle rental_units, room_configurations, rental_fares e house_sales(O02, O04, O05 e O08).

```
-- 5)
INSERT INTO rental_resumes(rental_unit, room_configuration, rental_fare, resume_date)
    SELECT l.id AS rental_unit, rg.id AS room_configuration,
           rf.id AS rental_fare, CURRENT_DATE AS resume_date
    FROM listings l, room_configurations rg, rental_fares rf
    WHERE l.room_type = rg.room_type AND l.n_beds = rg.n_beds AND
          l.n_baths = rg.n_baths AND l.is_bath_shared = rg.is_bath_shared AND
          l.price = rf.price AND l.minimum_nights = rf.minimum_nights;
```

Blocco di codice 10

Popolamento della della tabella rental_resumes (O06).

```
/*
    This function is used to dind the number of rooms
    for each room configuration
*/
CREATE OR REPLACE FUNCTION find_n_room()
RETURNS VOID AS
$$ BEGIN
    FOR i IN 1..26 LOOP
        UPDATE room_configurations
        SET n_rooms = I
        WHERE room_type LIKE Concat(CAST(i AS VARCHAR),' %');
   END LOOP;

END $$
LANGUAGE plpgsql;
```

Blocco di codice 11

```
-- Find the number of rooms for each room_configuration
SELECT find_n_room();

-- Set the room type to bedroom where the number of rooms is known
-- since, by default, only Studio room have a null n_rooms column
UPDATE room_configurations
SET room_type = 'Bedroom'
WHERE n_rooms IS NOT NULL;

-- Set n_rooms to 1 to Studio rooms
UPDATE room_configurations
SET n_rooms = 1
WHERE room_type LIKE 'Studio';
```

Blocco di codice 12

Definizione della funzione find_n_room() la quale estrae dalla colonna room_type il numero di stanze della configurazione utilizzata dall'unità di affitto. Assegnazione del valore standard 1 a tutte le camere di tipo 'Studio' (O07).

```
/*
    This function is used to set a license boolean value
    inside the rental units.
*/
CREATE OR REPLACE FUNCTION set_licence(VARCHAR(255))
RETURNS BOOLEAN AS
$$
    DECLARE is_licensed BOOLEAN;

    BEGIN
        IF $1 IS NOT NULL THEN
            SELECT TRUE INTO is_licensed;
        ELSE
            SELECT FALSE INTO is_licensed;
        END IF;

        RETURN is_licensed;
    END
$$
LANGUAGE plpgsql;
```

Blocco di codice 13

Funzione usata in fase di popolamento per settare il possesso o meno di una licenza.

```
-- Cast the column to decimal with 2 decimal digits
UPDATE rental_units
SET availability_rate_365 = ROUND(
    CAST(availability_rate_365 AS DECIMAL)/365, 2
);

-- Set availability_rate_365 as the avg of this column when
-- the value is zero
UPDATE rental_units
SET availability_rate_365 = (SELECT ROUND(AVG(availability_rate_365), 2)
                             FROM rental_units
                             WHERE availability_rate_365 <> 0)
WHERE availability_rate_365 = 0; -- unknown
```

Blocco di codice 14

Conversione in valore percentuale dell'attributo avalability_rate_365 e assegnazione del valor medio ove la percentuale è uguale a 0 (O02).

```
n_beds = df['name'].str.extract(r'([\d\.]+) bed[s]?\s*')
```

Blocco di codice 15

In questo blocco è riportata l'espressione regolare utilizzata per estrarre il numero di letti appartenenti a ciascun bnb. Tutte le altre operazioni di estrazione (numero bagni, bagni condivisi, nome bnb) sono simili, differiscono solo per l'espressione regolare utilizzata (O01).

Fonte dati D02

```
df.to_csv('../out/subway_stops.csv', index=False)
```

Blocco di codice 16

Eliminazione dell'indice dal file CSV (operazione eseguita per ogni file CSV, ma essendo ripetitiva è stata riportata solo in questo caso) (O01).

```
-- 0) Insert data into table
INSERT INTO subway_stops (id, name, latitude, longitude)
    SELECT  stop_id, stop_name, stop_lat, stop_lon
    FROM subway_stops_temp;
```

Blocco di codice 17

Popolamento della tabella subway_stops (O02).

```
/*
   Every subway stop in the imported dataset is differentiated in terms of its id
   in nord and sud, but the relative coordinates are always equals.
   These duplicates are eliminated
*/
DELETE
FROM subway_stops
WHERE id LIKE '%N' OR id LIKE '%S';
```

Blocco di codice 18

Eliminazione delle tuple che riportano la specializzazione della fermata della metropolitana (nord/sud), lasciando nel database solo l afermata con descrizione generale (O03).

Fonte dati D03

```python
def adjust_street_format(x: str):
    fap = re.split(pattern=r'\s(STREET|AVENUE)', string=x)[0] # first part of the address

    suffix_dict = {'1': 'ST', '2': 'ND', '3': 'RD'}

    street_number = re.split(pattern='\s', string=fap)[-1]

    # Obtains the correct suffix to concatenate
    th_condition = (
        # Conditions on last number of the street
        int(street_number[-1]) >= 4 or
        int(street_number[-1]) == 0 or
        # Conditions if the number end with a number between 11 and 19
        (len(street_number) >= 2 and street_number[-2] == '1')
    )

    # Selects the suffix to apply
    suffix = ('TH' if th_condition else suffix_dict[street_number[-1]])

    fap += suffix

    return fap + (' STREET' if 'STREET' in x else ' AVENUE')
```

Blocco di codice 19

Definizione della funzione adjust_street_format() per la standardizzazione degli indirizzi (O01).

```python
def geocode_address(address):
    global i

    location = geocoder.geocode(address)

    if location is not None:
        response = f'{location.latitude},{location.longitude}' # type: ignore
    else:
        response = None

    return response
```

Blocco di codice 20

Definizione della funzione di geocoding geocode_address() per individuare le coordinate geografiche dell'indirizzo dato in input (O02).

```sql
-- 0) Insert data into table
INSERT INTO house_sales(tax_class, sqft, price, construction_year, address, latitude, longitude)
    SELECT TAX_CLASS_AT_PRESENT, LAND_SQUARE_FEET, SALE_PRICE,
           YEAR_BUILT, address, latitude, longitude
    FROM house_sales_temp
    WHERE LAND_SQUARE_FEET IS NOT NULL;
```

Blocco di codice 21

Creazione della tabella house_sales popolandola solo con le tuple aventi anche i m^2 (O03).

```
-- Where the price is so low, it is considered as a
-- property swap (which is indicated by 0 for price)
UPDATE house_sales
SET price = 0
WHERE price = 10
```

Blocco di codice 22

Assegnazione di un valore standard per le case con prezzo di vendita <= 10 (O04).

Fonte dati D04

```
df['ARREST_DATE'] = pd.to_datetime(df['ARREST_DATE'])
df['ARREST_DATE'] = df['ARREST_DATE'].dt.strftime('%Y/%m/%d')
```

Blocco di codice 23

Conversione dell'attributo contenente la data di arresto in modo da renderla compatibile con i tipi standard di SQL (O01).

```
-- Insert data into table
INSERT INTO crimes(description)
    SELECT DISTINCT OFNS_DESC
    FROM nypd_arrests;

INSERT INTO arrests (id, arrest_date, crime, latitude, longitude)
    SELECT ARREST_KEY, arrest_date, ct.id, latitude, longitude
    FROM nypd_arrests na, crimes ct
    WHERE ct.description = na.OFNS_DESC -- map the description with its id
```

Blocco di codice 24

Creazione della tabella crimes (O02) e arrests (O03)

Fonti dati: DDL shapefiles da D05 a D10

Per queste fonti dati le modalità di popolamento sono le medesime. Pertanto, è riportato lo screen dell'istruzione INSERT INTO solo per i boroughs, in quanto il modus operandi è il medesimo per le altre tabelle.

```
-- 0) Insert data into table
INSERT INTO boroughs (id, name, perimeter)
    SELECT boro_code, boro_name, geom
    FROM boroughs_temp;
```

Blocco di codice 25

DML shapefile:

Esempio operazioni di mapping

```
-- 4) Map borough numeric code to its literal
WITH borough_mapping AS (
    SELECT digit, literal
    FROM (VALUES ('1', 'MN'), ('2', 'BX'), ('3', 'BK'), ('4', 'QN'), ('5', 'SI'))
    AS boro_codes_literal (digit, literal)
) UPDATE boroughs n
    SET id = (
        SELECT literal
        FROM borough_mapping bcm
        WHERE n.id = bcm.digit
    );
```

Blocco di codice 26

I mapping effettuati e descritti per le varie fonti dati, hanno **tutti** il formato riportato nel blocco di codice inserito precedentemente.

Esempio di inserimento

```
-- Insert value into positionings (n n table)
INSERT INTO positionings
    SELECT b.id AS borough, p.id AS park
    FROM parks p, boroughs b
    WHERE ST_Intersects(b.perimeter, p.perimeter);
```

Blocco di codice 27

Nel blocco di codice è presentato l'inserimento per la tabella positionings, è stata effettuata un'intersezione con il perimetro del borough relativo.

Esempio di inserimento di geometria

```
-- Turn the invalid geometries to valid ones
UPDATE neighborhoods
SET perimeter = ST_MakeValid(perimeter)
WHERE St_IsValid(perimeter) = 'False';
```

Blocco di codice 28

L'operazione si è resa necessaria per la presenza di alcune geometrie non valide.

Esempio di mapping dei POI tramite array

```
CREATE OR REPLACE FUNCTION Fill_poi_type(varchar[])
RETURNS VOID
AS $$
BEGIN
   FOR i IN 1..array_length($1, 1) LOOP
    UPDATE poi_types
    SET type_desc = $1[i]
    WHERE id = i;
   END LOOP;
END $$
LANGUAGE plpgsql;


SELECT Fill_poi_type(ARRAY ['Residential',
                            'Education Facility',
                            'Cultural Facility',
                            'Recreational Facility',
                            'Social Services',
                            'Transportation Facility',
                            'Commercial',
                            'Government Facility (non public safety)',
                            'Religious Institution',
                            'Health Services',
                            'Public Safety',
                            'Water',
                            'Miscellaneous' ]);
```

Nota: Per ulteriori chiarimenti si demanda alla consultazione dei commenti del codice presente nei vari scripts collocati all'interno delle cartelle di progetto.

Funzione di individuazione del neighborhood di riferimento

```
-- THIS FILE MUST BE EXECUTED AFTER dml_function_make_point.sql


-- This function finds the neighborhood, that contains the point, for every geometry point in the table
CREATE OR REPLACE FUNCTION find_neighborhood(table_name TEXT)
RETURNS VOID AS
$$
    BEGIN
        EXECUTE FORMAT('UPDATE %I '
                       'SET neighborhood = n.id '
                       'FROM neighborhoods AS n '
                       'WHERE ST_Within(coordinates, n.perimeter);', table_name);
    END
$$
LANGUAGE plpgsql;

SELECT find_neighborhood('rental_units');
SELECT find_neighborhood('arrests');
SELECT find_neighborhood('house_sales');
SELECT find_neighborhood('poi');
SELECT find_neighborhood('subway_stops');
```

Blocco di codice 29

Constraints

```sql
-- Constraints for "boroughs" table
ALTER TABLE boroughs
    ADD CONSTRAINT pk_borough PRIMARY KEY(id);

-- Constraints for "neighborhoods" table
ALTER TABLE neighborhoods
    ADD CONSTRAINT pk_neighborhoods PRIMARY KEY(id),
    ADD CONSTRAINT fk_neighborhoods_borough FOREIGN KEY(borough) REFERENCES boroughs(id)
        ON DELETE CASCADE ON UPDATE CASCADE;

-- Constraints for "crimes" table
ALTER TABLE crimes
    ADD CONSTRAINT pk_crimes PRIMARY KEY(id);

-- Constraints for "arrests" table
ALTER TABLE arrests
    ADD CONSTRAINT pk_arrests PRIMARY KEY(id),
    ADD CONSTRAINT fk_arrest_crime FOREIGN KEY(crime) REFERENCES crimes(id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    ADD CONSTRAINT fk_crimes_neighborhood FOREIGN KEY(neighborhood) REFERENCES neighborhoods(id)
        ON DELETE CASCADE ON UPDATE CASCADE;

-- Constraints for "hosts" table
ALTER TABLE hosts
    ADD CONSTRAINT pk_hosts PRIMARY KEY(id);
```

Blocco di codice 30

Conversione di latitudine e longitudine in punto geometrico

```sql
CREATE OR REPLACE FUNCTION make_point(table_name TEXT)
RETURNS VOID AS
$$
    BEGIN
        EXECUTE FORMAT('UPDATE %I SET coordinates = ST_SetSRID(ST_MakePoint(longitude, latitude), 4326);', table_name);
    END
$$
LANGUAGE plpgsql;

-- 1) convert coordinates into POINT geometry
SELECT make_point('arrests');
SELECT make_point('subway_stops');
SELECT make_point('rental_units');
SELECT make_point('house_sales');
```

Blocco di codice 31

La funzione attraverso una relazione Within individua i neighborhood di riferimento per i punti geometrici delle tabelle selezionate.

Codici per queries di analisi dati

Essendo le queries particolarmente lunghe, si rimanda alla consultazione dei file non-structuredX.sql (con x che varia tra 1 e 3), essendo essi efficacemente commentati.In questa sezione si danno solo alcune indicazioni:

In questa sezione si danno solo alcune indicazioni:

```sql
-- This function wraps the coordinates of the selected rental units (3.5 <= rate <= 5)
-- with a circle that uses the coordinates' point as center. The radius length, in meters,
-- is provided as input.
CREATE OR REPLACE FUNCTION create_sru_circle_x_meters(radius integer)
RETURNS TABLE (id bigint, name character varying, rate numeric, number_of_reviews integer,
               availability_rate_365 numeric, license boolean, neighborhood integer, circle geometry)
LANGUAGE 'plpgsql' AS
$$
BEGIN
    RETURN QUERY
        WITH sru_circle AS (
            SELECT  ru.id, ru.name, ru.rate, ru.number_of_reviews, ru.availability_rate_365,
                    ru.license, ru.neighborhood,
                    ST_Buffer(ru.coordinates::geography, $1, 'quad_segs=16')::geometry AS circle
            FROM    rental_units ru
            WHERE   ru.number_of_reviews > 50 AND ru.rate BETWEEN 3.5 AND 5
        ) SELECT * FROM sru_circle;
END $$;
```

Blocco di codice 32

La funzione riportata è usata rispetto alla ST_DWithin, viste le migliori prestazioni fornite, per conteggiare il numero di elementi di interesse presenti in una area circiolare sviluppata intorno ad un punto con un raggio specificato come input. Ciò avviene nelle query da Q2 a Q6.

```sql
WITH
    neighborhood AS (
        SELECT *
        FROM neighborhoods
        WHERE name LIKE n_name
    ),
    -- GeneratePoints takes 24 as seed to make the process repeatable
    n_pts AS (
        SELECT (ST_Dump(ST_GeneratePoints(perimeter, n_sub_perimeters * 100, 24))).geom AS geom
        FROM neighborhood
    ),
    n_pts_cluster AS (
        SELECT geom, ST_ClusterKMeans(geom, n_sub_perimeters) OVER() AS cluster
        FROM n_pts
    ),
    n_cluster_center AS (
        SELECT cluster, ST_Centroid(ST_Collect(geom)) AS geom
        FROM n_pts_cluster
        GROUP BY cluster
    ),
    n_voronoi AS (
        SELECT (ST_Dump(ST_VoronoiPolygons(ST_Collect(geom)))).geom AS geom
        FROM n_cluster_center
    )
    SELECT n.name, ST_Intersection(n.perimeter, v.geom) AS sub_perimeter
    FROM neighborhood n CROSS JOIN n_voronoi v;
```

Blocco di codice 33

In questo blocco di codice, incluso in una funzione più ampia sono usate funzioni avanzate di Postgres. L'utilizzo delle stesse è stato ripreso dalla documentazione ufficiale. Si seguano i seguenti link:

- https://blog.cleverelephant.ca/2018/06/polygon-splitting.html
- https://postgis.net/docs/ST_ClusterKMeans.html
- https://postgis.net/docs/ST_VoronoiPolygons.html
- https://postgis.net/docs/ST_Dump.html

```
CREATE TEMPORARY TABLE nearest_street_from_park_border_point AS (
    SELECT roads.id, roads.name, roads.dist, roads.path
    FROM Central_Park_decomposition cpd
        CROSS JOIN LATERAL (
        SELECT r.*, r.path::geography <-> cpd.geom::geography AS dist
        FROM roads r
        WHERE ST_Intersects(r.path, (SELECT perimeter FROM parks WHERE name LIKE 'Central Park')) = FALSE
        ORDER BY dist
        LIMIT 5

    ) AS roads
);
```

Blocco di codice 34

L'uso dell'operatore in questione per l'effettuazione di una nearest neighbor search spaziale è stato approfondito dal seguente link: https://postgis.net/workshops/postgis-intro/knn.html

```
CREATE INDEX sru_circle_700_meter_idx
    ON sru_circle_700_meter
    USING GIST(circle);
```

L'uso di un indice geometrico è stato un fattore chiave per migliorare le performances di determinate query. Si è consultato il seguente link: https://postgis.net/workshops/postgis-intro/indexing.html

```
-- Everything in this analysis comes in full circle. Central Park
-- (the place around which the stakeholder wanted to check house price to open its B&B in Q1)
-- is just 1km distant from the selected area.
SELECT ROUND(ST_Distance(a.sub_perimeter::geography,
                         p.perimeter::geography)::DECIMAL / 1000, 2) AS dist_from_central_park_km
FROM sub_perimeters_with_lower_avg_price_per_houses a, parks p
WHERE a.sd_id = 103 AND p.name = 'Central Park';
```

Blocco di codice 35

Per operazioni riguardanti distanze, al posto di dover cambiare SRID di riferimento della geometria con ST_Transform si è preferito usare in maniera più rapida il casting nel tipo ADT geography, la cui unità di misura è il metro e usa come piano di riferimento sferico. Il casting non porta perdita di informazioni come mostrato in documentazione: https://postgis.net/workshops/postgis-intro/geography.html#:~:text=Fortunately%2C%20you%20can%20convert%20objects,value%20you%20wish%20to%20cast.