Summative Assessment of Big Data Analytics

Word count: 2861

Task 1: Rental "demand" investigation

a. Discrete Variables: These include bedrooms, cats_allowed, dogs_allowed, smoking_allowed, wheelchair_access, electric_vehicle_charge, and comes_furnished. These are primarily binary (0 or 1) except for bedrooms, which is a count.

Assumptions:
The dataset assumes variables indicating demand, either directly or indirectly, and categorizes bedrooms and smoking_allowed as discrete factors affecting demand. Rent, type, and property size (sqfeet) are significant in determining demand.

Pre-processing undertaken to make the data fit for purpose
The data pre-processing steps for the datasets housing_train.csv and housing_test.csv involved several cleaning and encoding operations to prepare them for analysis. Initially, columns incompatible with Weka [1] , such as url, region_url, and description, were removed. In the demand column of housing_train.csv, URL links were deleted, and variations of the value 'no' were standardized. Additionally, the demand column was encoded to represent 'yes' as 1 and 'no' as 0. After cleaning, the files were converted into ARFF format [2] using ArffViewer for further processing.

Subsequently, Correlation Based Feature Selection (CBFS) [3] was applied to both the training and testing datasets, as illustrated in Figures 1 and 2. This process involved computing the Pearson correlation [4] for each attribute against the output variable, selecting only those attributes with a moderate-to-high positive or negative correlation. This selection technique, which utilizes the CorrelationAttributeEval [5] method and a Ranker search method in Weka, aims to improve the predictive accuracy by focusing on the most relevant features.

Feature Importance Analysis:
- Bedrooms: This feature has the most significant impact on predicting property demand, with a correlation of approximately both -0.42 in both training the testing dataset. This suggests that the number of bedrooms is a critical factor in determining low demand for properties.
- Other Features: Other discrete variables such as cats_allowed, dogs_allowed, smoking_allowed, wheelchair_access, electric_vehicle_charge, and comes_furnished have much lower importance scores, indicating a relatively minor role in predicting property demand within this dataset.

Justification for Techniques and Approach

Correlation Based Feature Selection (CBFS) was used for its efficacy in spotting linear links between independent variables and a target. This approach is ideal for datasets

with discrete variables and a binary outcome, aiding in pinpointing direct influencers without assuming complex or non-linear interactions. It's in line with literature that advises CBFS for initial feature selection in such analytical contexts.

Critical Evaluation

Effectiveness:

The approach was effective in pinpointing the most influential discrete variable (bedrooms) in predicting property demand. This aligns with intuitive understanding and previous research [6], which often highlights the significance of property size and type in rental decisions.

Limitations:

The analysis focused on linear relationships, possibly missing complex variable interactions affecting demand dynamics [7], like the impact of pets and furnishing on certain market segments. It suggests that the minor significance of variables beyond bedrooms doesn't imply unimportance but highlights the need for advanced analytical methods [8], such as machine learning, to reveal hidden patterns.

Learnings:

The exploration emphasized correlational analysis for initial feature selection, yet necessitated advanced techniques for comprehensive demand predictor understanding. It showcased data preprocessing and encoding's vital role in dataset preparation, crucial for categorical data, to accurately represent underlying phenomena.

b. Simple Linear Regression analysis [9] on preprocessed datasets revealed very low positive correlations between rent and demand, indicating minimal linear relationships. Specifically, training and testing datasets showed Pearson correlation coefficients of 0.0001 (Figure 3) and 0.1791 (Figure 4), respectively. Further analysis was conducted for the variable 'type' after converting nominal values to binary (Figure 5), and for other variables against demand across training and testing datasets (Figure 6 – Figure 24). The comprehensive Pearson correlation coefficients for each variable with demand were consolidated in Table 1, elucidating the extent of linear relationships between various factors and demand within the dataset.

Variable	Pearson correlation for training dataset	Pearson correlation for testing dataset
rent	0.0001	0.1791
type_apartment	0.2676	0.2222
type_house	-0.3311	-0.2487
type_manufactured	0.0047	0.0124

type_townhouse	-0.0203	0.015
type_condo	-0.0237	0.001
type_ duplex	0.0095	-0.0172
type_ flat	0.0043	0.0038
type_ cottage/cabin	0.0097	N.A.
type_ in-law	0.0047	N.A.
type_ loft	0.006	0.0072
type_land	0.0009	N.A.

Table 1: Pearson correlation for each variable with demand

In analyzing a dataset with mixed categorical and numerical variables, such as property type, demand, and rent, the study applied one-hot encoding for property type and binary encoding for demand to enable correlation analysis. This preprocessing aligns with established methods in the field [11]. The findings indicated a minor positive correlation between apartments and demand, a slight negative correlation for houses, and negligible correlation between rent prices and demand. These results suggest that property type is a stronger indicator of demand than rent levels.

Justification for Selection of Techniques

The approach to analyze the correlation between property demand and characteristics such as rent and type was driven by the nature of the data and the objectives of the investigation. Correlation analysis is a widely accepted statistical method for identifying the strength and direction of linear relationships between variables [12]. It is particularly useful in exploratory data analysis to understand potential predictors for a model or to identify trends within the data.

Evaluation of Approach and Techniques

Effectiveness:

One-hot Encoding [13]: Effective for transforming categorical data into a numerical format suitable for correlation analysis, allowing us to include property type in our analysis.

Correlation Analysis [14]: Useful for identifying potential linear relationships between variables at a high level.

Ineffectiveness:

Overlooking Non-linear Relationships: The chosen method focuses on linear relationships, potentially missing non-linear dynamics between variables and demand [15].

Simplicity of Demand Encoding: Treating demand as binary oversimplifies the complexity of demand, possibly overlooking variations in demand levels [16] .

Reflections:

The simplicity of the approach allowed for a straightforward analysis but may have limited the depth of insights. For example, machine learning models like decision trees or regression models could provide more nuanced understandings of how different property characteristics interact to influence demand. Additionally, considering demand as a continuous variable, if data permits, could offer a more granular view of how rent and property types affect demand levels.

c. In the study, Excel [17] and Weka were used to preprocess and analyze the datasets, focusing on "demand" and "sqfeet" attributes. Clustering analysis [18] on the training dataset identified optimal sqfeet ranges, with a mean of 970.9358 and standard deviation (SD) of 786.6424 for sqfeet, and a mean of 0.9998 and SD of 0.0021 for demand (Figure 25). The calculated optimal range for sqfeet was between 577.6146 and 1364.257, derived from subtracting and adding half of the SD to the mean, respectively. This range was applied to clean the data in Excel, filtering sqfeet values outside this range and selecting records with demand=1. A simple linear regression indicated a significant linear relationship between cleaned sqfeet and demand, evidenced by a correlation coefficient of 0.2606 (Figure 26). Further analysis revealed two clusters with distinct means and SDs for sqfeet and corresponding demand values, suggesting different optimal sqfeet ranges for generating high demand.

   The testing dataset underwent a similar process, where the upper bound for the optimal sqfeet range was extended to 1651.958, based on cluster analysis (Figure 28). Cleaning the testing dataset and further cluster analysis (Figure 29) confirmed the linear relationship between sqfeet and demand within this new range, with the same correlation coefficient of 0.2606. The findings suggest that property size, represented by sqfeet, has an optimal range that significantly influences demand.

   Assumptions Made:

   Demand for properties is quantifiable and directly correlates with specific attributes, primarily "sqfeet".

   The data is normally distributed [19], allowing for the application of mean and standard deviation in identifying optimal ranges .

   A linear relationship exists between "sqfeet" and "demand", making linear regression a suitable analysis technique [19].

Justification for Techniques

Clustering Analysis was chosen for its effectiveness in identifying patterns and groupings in datasets without predefined categories [20], making it ideal for exploring the relationship between "sqfeet" and "demand".

Simple Linear Regression was selected due to its suitability in modeling the linear relationship between two continuous variables, supported by previous studies suggesting a linear relationship between property attributes and demand [21].

Critical Evaluation:

The approach effectively identified a significant relationship between "sqfeet" and "demand", with the clustering and linear regression analyses providing a robust framework for analysis [22]. However, the reliance on linear regression and normal distribution assumptions may not capture non-linear relationships or the influence of outlier data points [23].

The process highlighted the importance of data preprocessing and the selection of appropriate analysis techniques tailored to the nature of the data and the investigation's goals. Future investigations could benefit from incorporating more sophisticated models, such as multiple regression or machine learning algorithms, to account for multiple variables affecting demand [24].

Task 2: Storing data and possible solutions

Part 1: Design a relational database

a. Initial Data Structure and Normalization Process

The dataset initially comprises various fields including id, url, region, rent, and more. The normalization process begins by ensuring the dataset adheres to the First Normal Form (1NF), where id is selected as the primary key for its unique identification capability, essential for data integrity and efficient retrieval. The dataset inherently meets 1NF requirements as it presents atomic values, lacks repeating groups, and each row is uniquely identifiable [25].

Transitioning to Second Normal Form (2NF) requires the database to be in 1NF and mandates that all non-key attributes are fully functional and dependent on the primary key. This stage often involves splitting the database into multiple tables (Listings, Regions, Amenities, Options, Demand) to eliminate partial dependencies and ensure all non-key attributes directly depend on the primary key [26].

- Listings: (id, url, rent, type, sqfeet, bedrooms, bathrooms, description)

- Regions: (region, state, region_url, latitude, long)

- Amenities: (id, cats_allowed, dogs_allowed, smoking_allowed, wheelchair_access, electric_vehicle_charge, comes_furnished)

- Options: (id, laundry_options, parking_options)

- Demand: (id, demand)

Achieving Third Normal Form (3NF) necessitates the database being in 2NF and having no transitive dependencies, where no non-key attribute depends on another non-key attribute. Adjustments ensure each attribute in every table directly depends on the primary key, refining the database structure to avoid anomalies during data operations [27].

- Listings: Primary Key (id), Foreign Keys (region), and attributes (url, rent, type, sqfeet, bedrooms, bathrooms, description).

- Regions: Primary Key (region), attributes (state, region_url, latitude, long).

- Amenities: Primary Key (id), attributes (cats_allowed, dogs_allowed, smoking_allowed, wheelchair_access, electric_vehicle_charge, comes_furnished)

- Options: Primary Key (id), attributes for laundry_options and parking_options.

- Demand: Primary Key (id), attribute (demand).

Final Design and ER Diagram Components

The final database design reflects a careful consideration of attribute grouping based on logical relationships and dependencies, with the primary aim of enhancing data integrity, reducing redundancy, and facilitating efficient data management. Key components and relationships are illustrated in the ER diagram, including:

- Listings with a primary key (id) and attributes related to the listing itself.
- Regions categorized geographically with attributes including state and coordinates, reflecting a one-to-many relationship with Listings.
- Amenities and Options tables, each linked to Listings by a primary key, ensuring attributes like laundry_options and parking_options are normalized to eliminate redundancy.
- Demand table, directly related to Listings, capturing the demand attribute.

This structured approach, guided by the principles of normalization, ensures a robust database design capable of supporting accurate and efficient data operations. By focusing on the selection of primary keys and the logical grouping of attributes, the design achieves a balance between data integrity and operational efficiency, providing a solid foundation for data management and retrieval within the relational database structure [28].

Lucidchart [29] was used to create a UML standard ER diagram (Figure 31)

b. Sample SQL Statements

Please see the appendix for the sample SQL [30] for the database. DuckDB [31] was chosen to be the database. DuckDB and pandas [32] python [33] package were used on Jupyter notebook [34] to build the database. Output of the sample SQL can be seen in the screenshot of the Jupyter notebook (Figures 41 – 44). Jupyter notebook was converted to html to include in the submission of the summative assessment.

Part 2: Consider scaling

For addressing the needs of a housing manager dealing with significant data volumes across international offices, a distributed system architecture that utilizes Hadoop [35] ecosystem technologies is proposed as a comprehensive solution. The Hadoop ecosystem is especially well-suited for processing and analyzing large datasets within a distributed computing environment, making it an ideal option for a global rental landscape that demands quick data processing and high responsiveness.

Scalable Solution Using Hadoop

Hadoop Distributed File System (HDFS): Forms the basis of the Hadoop ecosystem, offering a reliable and scalable storage solution capable of managing data from tens of megabytes to petabytes over multiple computers. It divides large files into blocks and distributes them throughout the cluster to ensure high availability and fault tolerance [36].

Apache Hadoop YARN: Acts as a resource manager, dynamically allocating system resources to various applications within the Hadoop ecosystem to optimize cluster utilization [37].

Apache HBase: A NoSQL distributed database running atop HDFS, suitable for real-time read/write access to large datasets, such as those generated by international rental listings [38].

Apache Hive: Enables querying and managing large datasets in distributed storage via SQL-like commands, ideal for analytical queries and reports on collected data [39].

Apache Spark: A cluster-computing framework that surpasses traditional MapReduce jobs in speed, apt for real-time analytics and rapid data processing tasks [40].

Apache Kafka: A distributed streaming platform that complements Hadoop for real-time data processing and messaging, capable of handling large data volumes from various sources [41].

Justification for Using Hadoop Ecosystem

Scalability and Fault Tolerance: With components like HDFS and YARN, the Hadoop ecosystem offers excellent scalability and fault tolerance, distributing data across many

servers to handle growing data volumes and replicating data blocks across nodes to enhance resilience against failures [42].

Cost-Effectiveness: Hadoop operates on commodity hardware, providing an economical solution for storing and processing large datasets, crucial for international expansion [43].

Flexibility and Real-Time Processing: The ecosystem's ability to process various data types and integrate tools for real-time analytics (Apache Spark) and data ingestion (Apache Kafka) makes it highly suitable for the dynamic global rental market [44].

Example Use Case

An application of the Hadoop ecosystem could involve monitoring and analyzing rental trends across regions to identify high-demand areas. Data from international listings would be ingested into HDFS, analyzed in real-time using Spark, and stored in HBase for quick access. Kafka would stream data changes [45], triggering alerts or actions based on Spark's analysis, enabling rapid market response and effective resource allocation across international offices.

Comparison Against Other Technologies

Cloud-Based Solutions: While cloud platforms like AWS or Google Cloud Platform offer managed services that can simplify global data infrastructure management, they may become costly with increasing data and compute needs. These platforms provide scalability and global reach but may lack the control and cost-effectiveness of a self-managed Hadoop cluster for large-scale data processing [46].

Traditional RDBMS: These systems, designed for structured data, often fail to scale efficiently for the volume and variety of data encountered in a global rental market. They lack the scalability and fault tolerance of Hadoop and can become prohibitively expensive to scale for petabyte-level data handling [47].

NoSQL Databases: Although databases like Cassandra or MongoDB offer scalability and flexibility for unstructured data, they don't provide an integrated ecosystem for comprehensive data processing, analytics, and real-time processing like Hadoop, when combined with Spark and Hive, does [48].

Task 3: Considering web-based application

Given the context of developing a public-facing application to promote a housing manager's business and the intention to capture personal details of potential clients, here are the three most salient privacy issues to consider, informed by the details provided in Task 1 and Task 2:

1. Data Security and Privacy Compliance

Issue: The collection of personal details online raises significant concerns regarding data security and compliance with privacy regulations [49] (e.g., GDPR in the EU [50], CCPA in

California [51]). Without proper safeguards, sensitive information could be vulnerable to breaches, leading to legal repercussions and loss of trust.

Strategies:

Encryption: Implement robust encryption protocols for data at rest and in transit to protect personal information [52].

Compliance Audits: Regularly audit practices to ensure adherence to all relevant privacy laws and regulations [53].

Data Minimization: Collect only the information necessary for the provided services, reducing the potential impact of a data breach [54].

2. Consent and Transparency

Issue: Potential clients must be fully informed about how their data will be used, including data analysis and storage intentions. Lack of transparency and consent could lead to privacy violations and erode client trust [55].

Strategies:

Clear Privacy Policy: Develop a clear and accessible privacy policy detailing data use, storage, and sharing practices [56].

Consent Mechanism: Implement an explicit consent mechanism for users to agree to the collection and use of their data [57].

User Control: Provide users with tools to view, modify, and delete their personal information, enhancing transparency and control [58].

3. Long-term Data Storage and Accessibility

Issue: The move towards permanent data storage, as indicated in Task 2's use of distributed systems like Hadoop, increases the risk of data becoming outdated, inaccurately reflecting client preferences, or being accessed without authorization over time [59].

Strategies:

Data Retention Policy: Establish and communicate a clear data retention policy that defines how long data will be stored and the criteria for its deletion [60].

Regular Data Review: Periodically review stored data for accuracy and relevance, deleting or updating information as necessary [61].

Access Controls: Implement strict access controls and authentication mechanisms to ensure only authorized personnel can access sensitive data [62].

Reference

[1] “Home - Weka Wiki,” waikato.github.io. https://waikato.github.io/weka-wiki/ (accessed: Mar. 3, 2024).

[2] Attribute-relation file format (ARFF), https://www.cs.waikato.ac.nz/~ml/weka/arff.html (accessed Mar. 3, 2024).

[3] M. A. Hall, Correlation-based Feature Selection for Machine Learning. 1999.

[4] “Vii. note on regression and inheritance in the case of two parents,” Proceedings of the Royal Society of London, vol. 58, no. 347–352, pp. 240–242, Dec. 1895. doi:10.1098/rspl.1895.0041

[5] Correlationattributeeval, https://weka.sourceforge.io/doc.dev/weka/attributeSelection/CorrelationAttributeEval.html (accessed Mar. 3, 2024).

[6] S. Clark and N. Lomax, “Rent/price ratio for English housing sub-markets using matched sales and rental data,” Area, vol. 52, no. 1, pp. 136–147, May 2019. doi:10.1111/area.12555

[7] A. C. Goodman, “Demographics of individual housing demand,” Regional Science and Urban Economics, vol. 20, no. 1, pp. 83–102, Jun. 1990. doi:10.1016/0166-0462(90)90026-y

[8] M. Cajias and J.-A. Zeitler, “Quantifying the drivers of residential housing demand – an interpretable machine learning approach,” Journal of European Real Estate Research, vol. 16, no. 2, pp. 172–199, Jul. 2023. doi:10.1108/jerer-02-2023-0008

[9] Simplelinearregression. (2022, January 28). https://weka.sourceforge.io/doc.dev/weka/classifiers/functions/SimpleLinearRegression.html

[10] J. Brownlee, “Why one-hot encode data in machine learning?,” MachineLearningMastery.com, https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/ (accessed Mar. 3, 2024).

[11] T. Hastie, R. Tibshirani, and J. Friedman, TheElements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed. New York, NY, USA: Springer, 2009

[12] T. J. Cleophas and A. H. Zwinderman, “Bayesian Pearson Correlation Analysis,” Modern Bayesian Statistics in Clinical Research, pp. 111–118, 2018. doi:10.1007/978-3-319-92747-3_11

[13] J. Brownlee, “Ordinal and one-hot encodings for Categorical Data,” MachineLearningMastery.com, https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/ (accessed Mar. 3, 2024).

[14] K. I. Park, Fundamentals of Probability and Stochastic Processes with Applications to Communications. Cham: Springer International Publishing, 2018.

[15] P. Ranganathan and R. Aggarwal, “Common pitfalls in statistical analysis: The use of Correlation Techniques,” Perspectives in Clinical Research, vol. 7, no. 4, p. 187, 2016. doi:10.4103/2229-3

[16] B. Ranković, R.-R. Griffiths, H. B. Moss, and P. Schwaller, “Bayesian optimisation for additive screening and yield improvements – beyond one-hot encoding,” Digital Discovery, 2024. doi:10.1039/d3dd00096f 485.192046

[17] “Free online spreadsheet software: Excel: Microsoft 365,” Free Online Spreadsheet Software: Excel | Microsoft 365, https://www.microsoft.com/en-ca/microsoft-365/excel (accessed Mar. 3, 2024).

[18] C. C. Aggarwal and C. K. Reddy, Data Clustering: Algorithms and Applications. Boca Raton, FL: Chapman and Hall/CRC, 2018.

[19] D. A. Freedman, Statistical Models: Theory and Practice. New York: Cambridge University Press, 2012.

[20] E. Uprichard and D. S. Byrne, Cluster Analysis. London: SAGE, 2012.

[21] A. C. Rencher and W. F. Christensen, Methods of Multivariate Analysis. Hoboken: Wiley, 2012.

[22] G. Paliouras, V. Karkaletsis, and C. D. Spyropoulos, Machine Learning and Its Applications: Advanced Lectures. Berlin: Springer, 2001.

[23] J. P. Stevens, “Outliers and influential data points in regression analysis.,” Psychological Bulletin, vol. 95, no. 2, pp. 334–344, Mar. 1984. doi:10.1037/0033-2909.95.2.334

[24] T. Doan and J. Kalita, "Selecting Machine Learning Algorithms Using Regression Models," 2015 IEEE International Conference on Data Mining Workshop (ICDMW), Atlantic City, NJ, USA, 2015, pp. 1498-1505, doi: 10.1109/ICDMW.2015.43.

[25] E. F. Codd, The Relational Model for Database Management: Version 2. Reading, Mass. : Addison-Wesley, 1991.

[26] C. J. Date, An Introduction to Database Systems. Reading, MA: Pearson, 2004.

[27] W. Kent, “A simple guide to five normal forms in relational database theory,” Communications of the ACM, vol. 26, no. 2, pp. 120–125, Feb. 1983. doi:10.1145/358024.358054

[28] D. Yeh, Y. Li, and W. Chu, “Extracting entity-relationship diagram from a table-based Legacy Database,” Journal of Systems and Software, vol. 81, no. 5, pp. 764–771, May 2008. doi:10.1016/j.jss.2007.07.005

[29] “Intelligent diagramming,” Lucidchart, https://www.lucidchart.com/ (accessed Mar. 3, 2024).

[30] E. F. Codd, “A relational model of data for large shared data banks,” Communications of the ACM, vol. 13, no. 6, pp. 377–387, Jun. 1970. doi:10.1145/362384.362685

[31] “An in-process SQL OLAP Database Management System,” DuckDB, https://duckdb.org/ (accessed Mar. 3, 2024).

[32] “Pandas,” pandas, https://pandas.pydata.org/ (accessed Mar. 3, 2024).

[33] “Welcome to Python.org,” Python.org, https://www.python.org/ (accessed Mar. 3, 2024).

[34] “Project jupyter,” Project Jupyter, https://jupyter.org/ (accessed Mar. 3, 2024).

[35] H. Almansouri and Y. Masmoudi, Hadoop distributed file system for Big Data Analysis, Oct. 2022. doi:10.21203/rs.3.rs-2125174/v1

[36] T. White, Hadoop: The Definitive Guide. Sebastopol: O’Reilly & Associates, 2015. Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop Distributed File System.

[37] “Apache Hadoop Yarn,” Apache Hadoop 3.3.6 – Apache Hadoop YARN, https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html (accessed Mar. 3, 2024).

[38] “Apache HBase – Apache HBaseTM home,” Apache HBase – Apache HBaseTM Home, https://hbase.apache.org/ (accessed Mar. 3, 2024).

[39] “Apache Hive,” Apache Software Foundation, https://hive.apache.org/ (accessed Mar. 3, 2024).

[40] “Apache Hive,” Apache Software Foundation, https://hive.apache.org/ (accessed Mar. 3, 2024).

[41] “Apache kafka,” Apache Kafka, https://kafka.apache.org/ (accessed Mar. 3, 2024).

[42] T. Hussain, A. Sanga, and S. Mongia, “Big Data Hadoop Tools and technologies: A Review,” SSRN Electronic Journal, 2019. doi:10.2139/ssrn.3462554

[43] M. Sogodekar, S. Pandey, I. Tupkari and A. Manekar, "Big data analytics: hadoop and tools," 2016 IEEE Bombay Section Symposium (IBSS), Baramati, India, 2016, pp. 1-6, doi: 10.1109/IBSS.2016.7940204.

[44] N. Marz and J. Warren, Big Data: Principles and Best Practices of Scalable Real-Time Data Systems. Shelter Island: Manning, 2015.

[45] R. Gancarz, “Expedia uses WebSockets and Kafka to query near real-time streaming data,” InfoQ, https://www.infoq.com/news/2023/12/expedia-websockets-kafka-query/ (accessed Mar. 3, 2024).

[46] C. Foot, “On-premises vs. Cloud Data Warehouses: Pros and cons,” Data Management, https://www.techtarget.com/searchdatamanagement/tip/On-premises-vs-cloud-data-warehouses-Pros-and-cons (accessed Mar. 3, 2024).

[47] “Big Data, RDBMS and Hadoop - A Comparative Study,” International Journal of Science and Research (IJSR), vol. 5, no. 3, pp. 1455–1458, Mar. 2016. doi:10.21275/v5i3.nov162167

[48] “Hadoop vs. mongodb: A comprehensive comparison for Big Data and NoSQL,” sprinkle, https://www.sprinkledata.com/blogs/hadoop-vs-mongodb-a-comprehensive-comparison-for-big-data-and-nosql (accessed Mar. 3, 2024).

[49] O. Olukoya, “Assessing frameworks for eliciting Privacy & Security Requirements from laws and regulations,” Computers & Security, vol. 117, p. 102697, Jun. 2022. doi:10.1016/j.cose.2022.102697

[50] “Official Legal Text,” General Data Protection Regulation (GDPR), https://gdpr-info.eu/ (accessed Mar. 3, 2024).

[51] “California Consumer Privacy Act (CCPA),” State of California - Department of Justice - Office of the Attorney General, https://oag.ca.gov/privacy/ccpa (accessed Mar. 3, 2024).

[52] P. Farshim, B. Libert, K. G. Paterson, and E. A. Quaglia, “Robust encryption, revisited,” Public-Key Cryptography – PKC 2013, pp. 352–368, 2013. doi:10.1007/978-3-642-36362-7_22

[53] J. Reuben, L. A. Martucci, and S. Fischer-Hübner, “Automated log audits for privacy compliance validation: A literature survey,” IFIP Advances in Information and Communication Technology, pp. 312–326, 2016. doi:10.1007/978-3-319-41763-9_21

[54] A. J. Biega, P. Potash, H. Daumé, F. Diaz, and M. Finck, “Operationalizing the legal principle of data minimization for personalization,” Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Jul. 2020. doi:10.1145/3397271.3401034

[55] R. J. Smith, D. Grande, and R. M. Merchant, “Transforming scientific inquiry,” Academic Medicine, vol. 91, no. 4, pp. 469–472, Apr. 2016. doi:10.1097/acm.0000000000001022

[56] Y. Chang, S. F. Wong, C. F. Libaque-Saenz, and H. Lee, “The role of privacy policy on consumers’ perceived privacy,” Government Information Quarterly, vol. 35, no. 3, pp. 445–459, Sep. 2018. doi:10.1016/j.giq.2018.04.002

[57] B. W. Schermer, B. Custers, and S. van der Hof, “The crisis of consent: How stronger legal protection may lead to weaker consent in data protection,” Ethics and Information Technology, Mar. 2014. doi:10.1007/s10676-014-9343-8

[58] O. Tene and J. Polonetsky, “Big data for all: Privacy and user control in the age of analytics,” Northwestern J. Technol. Intell. Property, vol. 11, no. 5, 2012, Art. no. 1.

[59] S. K. Sarin and N. A. Lynch, "Discarding Obsolete Information in a Replicated Database System," in IEEE Transactions on Software Engineering, vol. SE-13, no. 1, pp. 39-47, Jan. 1987, doi: 10.1109/TSE.1987.232564.

[60] J. Li, S. Singhal, R. Swaminathan and A. H. Karp, "Managing Data Retention Policies at Scale," in IEEE Transactions on Network and Service Management, vol. 9, no. 4, pp. 393-406, December 2012, doi: 10.1109/TNSM.2012.101612.110203.

[61] “Developing routines for Regular Data Review,” National Student Support Accelerator, https://studentsupportaccelerator.org/tutoring/data-use/evaluation-improvement/developing-routines-regular-data-review (accessed Mar. 3, 2024).

Appendices

(Due to the limit of pages, marker can enlarge the document to see the details)

Figures and code in jupyter notebook/html are attached in the submission file

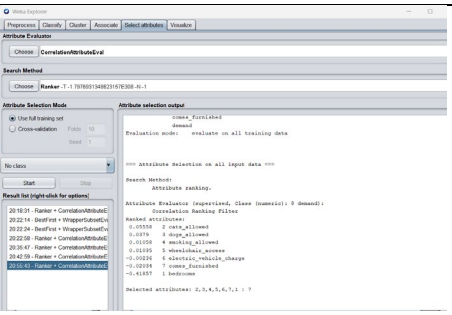

Figure 1: Correlation Based Feature Selection on training dataset

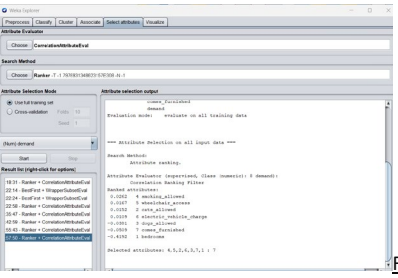

Figure 2: Correlation Based Feature Selection on testing dataset

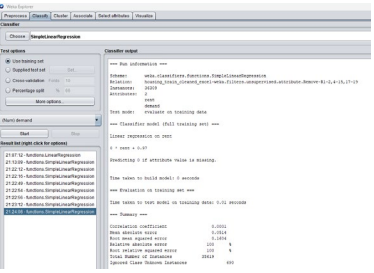

Figure 3: Simple Linear Regression with rent and demand on the training dataset

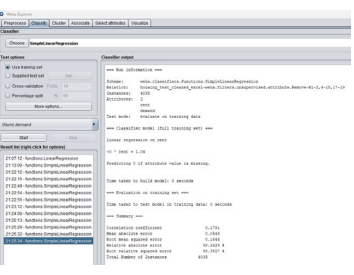

Figure 4: Simple Linear Regression with rent and demand on the testing dataset

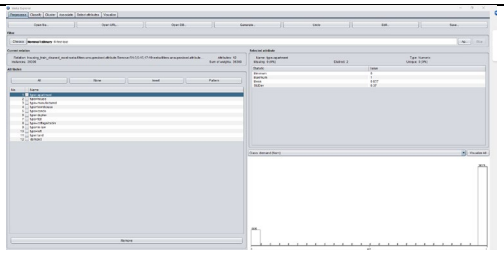

Figure 5: preprocessing of “nominal to binary” was performed on the type variable.

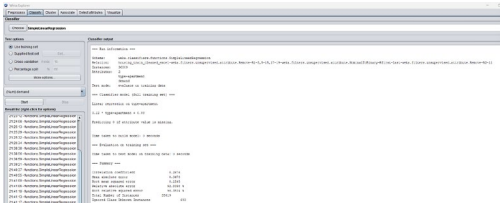

Figure 6: Simple Linear Regression with type_apartment and demand on the training dataset

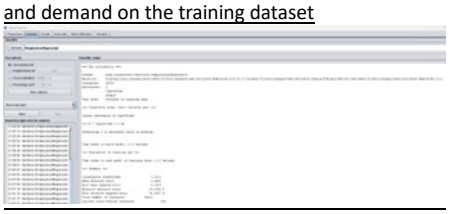

Figure 7: Simple Linear Regression with type_house and demand on the training dataset

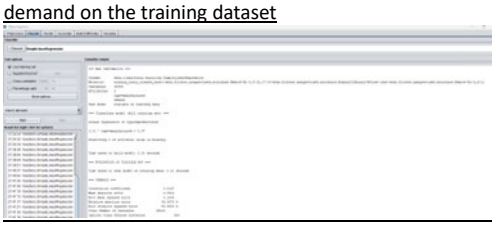

Figure 8: Simple Linear Regression with type_manufactured and demand on the training dataset

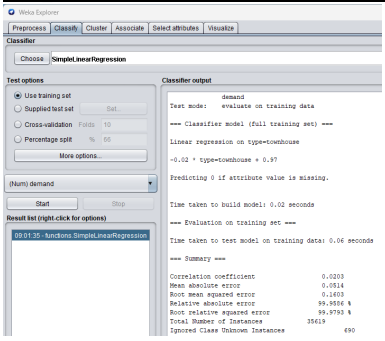

Figure 9: Simple Linear Regression with type_townhouse and demand on the training dataset

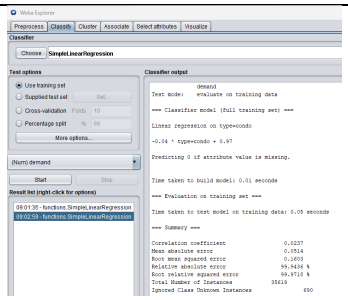

Figure 10: Simple Linear Regression with type_condo and dema

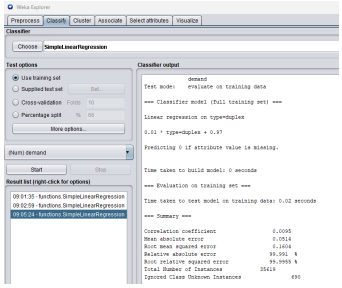

Figure 11: Simple Linear Regression with type_duplex and dema

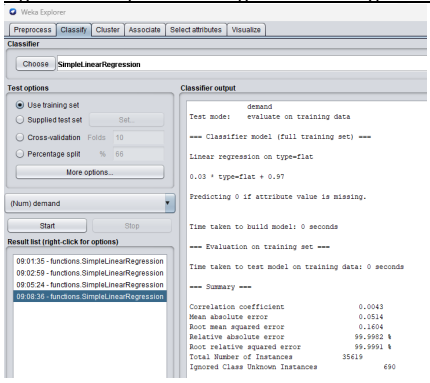

Figure 12: Simple Linear Regression with type_flat and demand

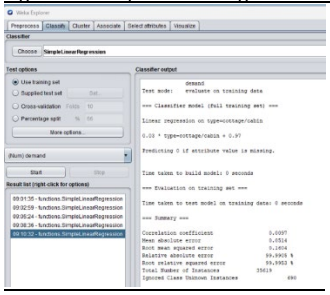

Figure 13: Simple Linear Regression with type_cottage/cabin an

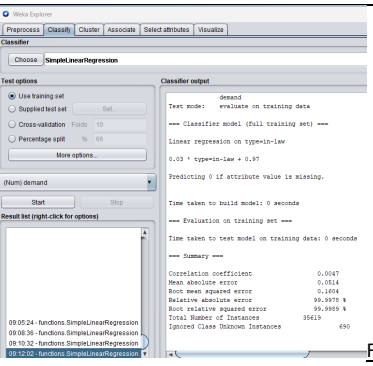

Figure 14: Simple Linear Regression with type_ in-law and demand on the training dataset

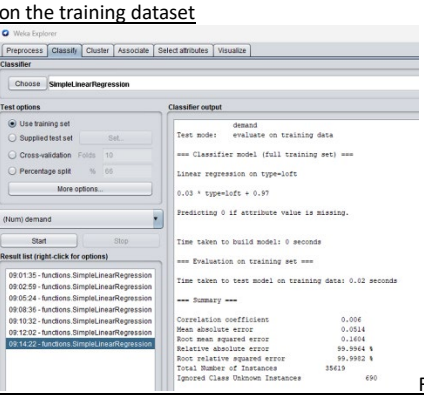

Figure 15: Simple Linear Regression with type_ loft and demand on the training dataset

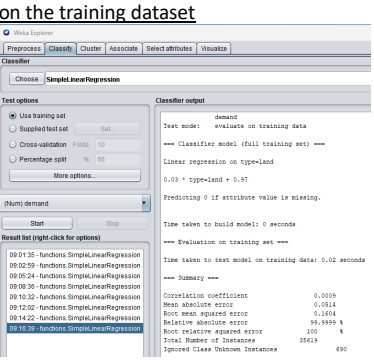

Figure 16: Simple Linear Regression with type_ land and demand on the training dataset

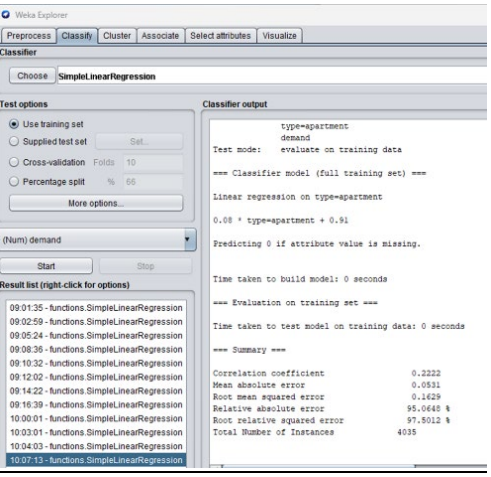

Figure 17: Simple Linear Regression with type_ apartment and demand on the testing dataset

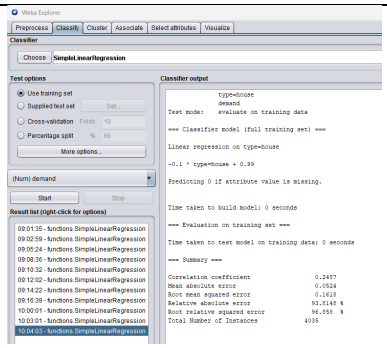

Figure 18: Simple Linear Regression with type_ house and demand on the testing dataset

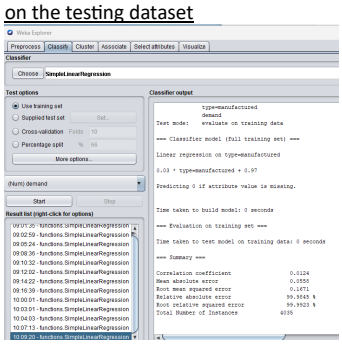

Figure 19: Simple Linear Regression with type_ manufactured and demand on the testing dataset

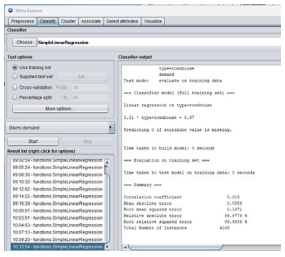

Figure 20: Simple Linear Regression with type_ townhouse and demand on the testing dataset

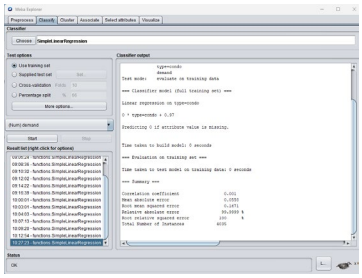

Figure 21: Simple Linear Regression with type_ condo and demand on the testing dataset

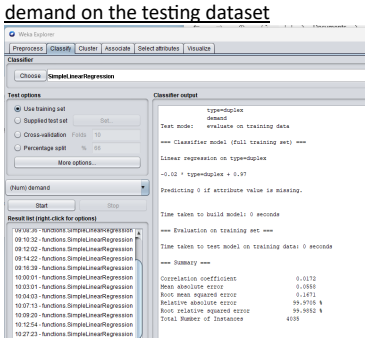

Figure 22: Simple Linear Regression with type_ duplex and demand on the testing dataset

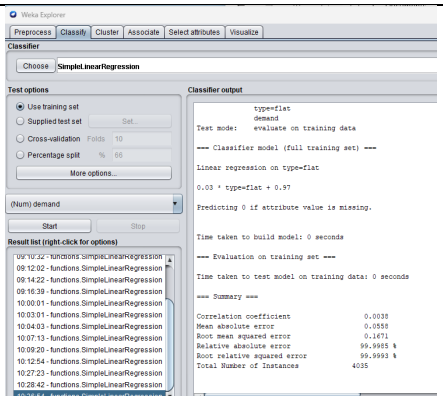

Figure 23: Simple Linear Regression with type_ flat and demand

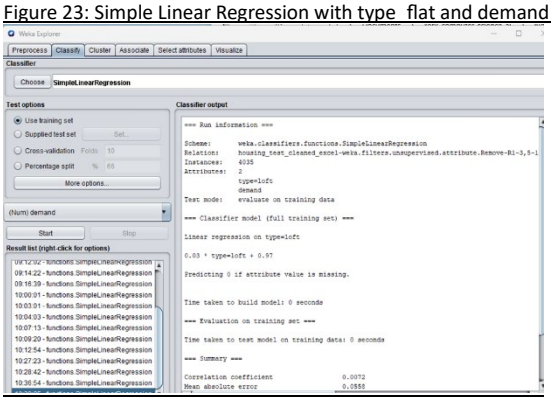

Figure 24: Simple Linear Regression with type_ loft and demand

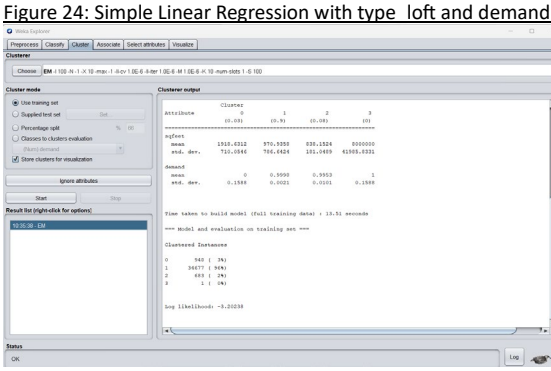

Figure 25: Clustering with sqlfeet and demand on the training da

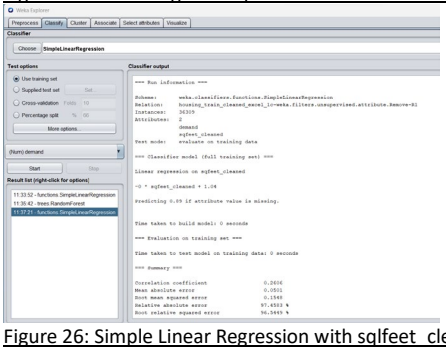

Figure 26: Simple Linear Regression with sqlfeet_ cleaned and d

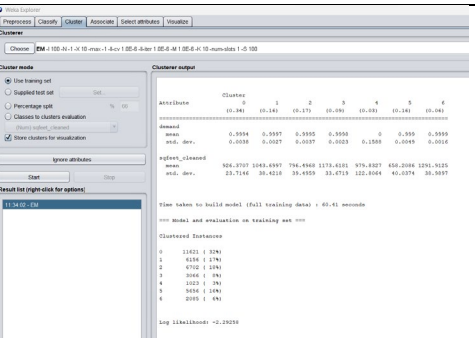

Figure 27: Further clustering with sqlfeet and demand on the selected cleaned training dataset

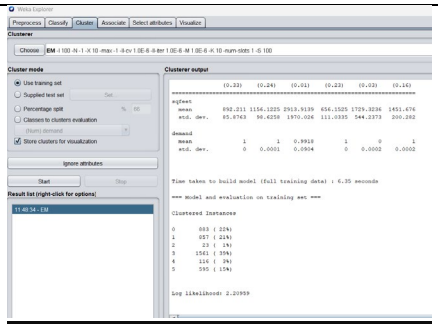

Figure 28: Clustering with sqlfeet and demand on the testing dataset

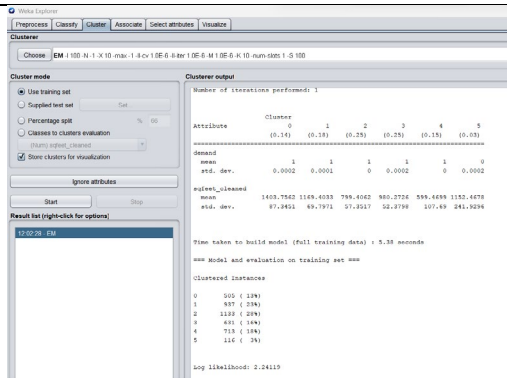

Figure 29: Further clustering with sqlfeet and demand on the selected cleaned testing dataset

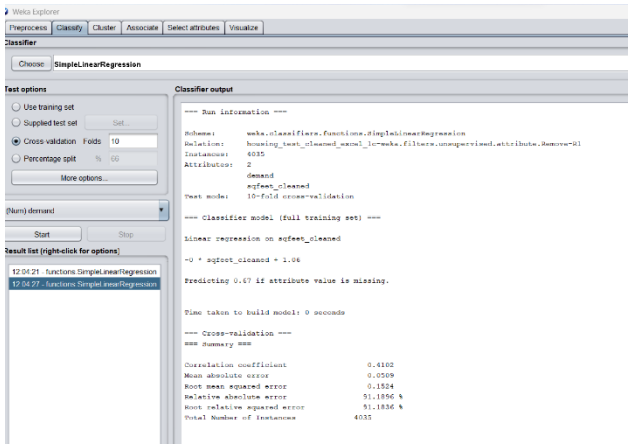

Figure 30: Simple Linear Regression with sqlfeet_cleaned and demand on the testing dataset

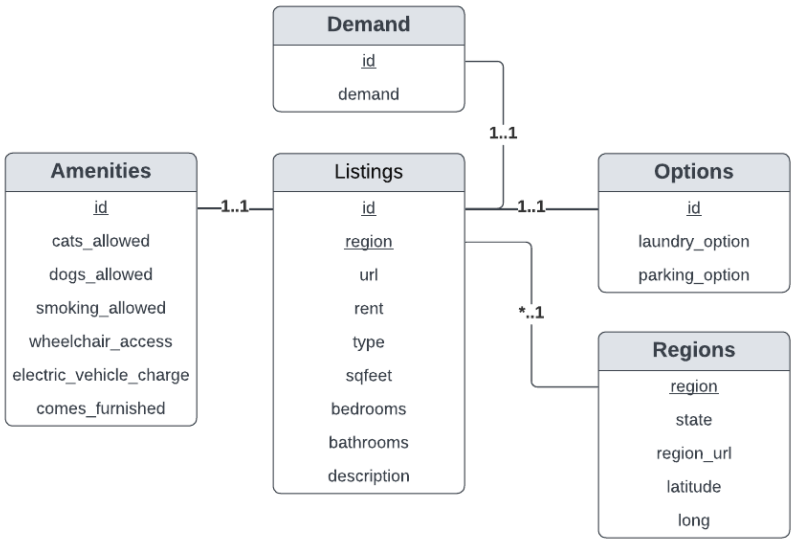

Figure 31: UML standard ER diagram of the database

Sample SQL statement

i. Inserting a New Line of Data

Assuming the tables have been created according to the ER diagram, here's how you would insert a new listing, along with its amenities, options, and demand information. Note that this is a simplified example and assumes all necessary foreign keys and relationships have been properly set up.

-- Insert into Regions INSERT INTO Regions (region, state, region_url, latitude, long) VALUES ('San Francisco', 'CA', 'https://sfbay.craigslist.org', 37.7749, -122.4194);	-- Insert into Listings INSERT INTO Listings (id, url, region, rent, sqfeet, bedrooms, bathrooms, description) VALUES (123456789, 'https://sfbay.craigslist.org/apa/d/san-francisco-spacious-one-bedroom/123456789.html', 'San Francisco', 950, 700, 1, 1, 'Spacious one-bedroom apartment in the heart of the city. Close to public transportation and parks.');
-- Insert into Amenities INSERT INTO Amenities (id, cats_allowed, dogs_allowed, smoking_allowed, wheelchair_access, electric_vehicle_charge, comes_furnished) VALUES (123456789, 1, 1, 0, 1, 0, 0); -- Insert into Options INSERT INTO Options (id, laundry_options, parking_options) VALUES (123456789, 'on-site', 'street parking');	-- Insert into Demand INSERT INTO Demand (id, demand) VALUES (123456789, 'yes');

ii. Extracting Description for Specific Properties

To extract the distinct descriptions of properties meeting certain criteria, you would join the relevant tables and apply the conditions:

SELECT distinct(L.description) FROM listings L JOIN amenities A ON L.id = A.id JOIN regions R ON L.region = R.region

WHERE L.rent <= 1000 AND A.cats_allowed = 1 AND A.dogs_allowed = 1 AND R.state = 'ca';

This query joins the Listings, Amenities, and Regions tables to filter listings with rent equal to or less than $1000, that allow both cats and dogs, and are located in California (ca).

iii. Extracting Average Rental Value for Each State

To compare the average rental values by state, use the GROUP BY clause:

SELECT R.state, AVG(L.rent) AS average_rent FROM listings L JOIN regions R ON L.region = R.region

GROUP BY R.state ORDER BY average_rent DESC;

This query calculates the average rent for listings in each state, grouping the results by state and ordering them by the average rent in descending order to easily compare.

These SQL examples demonstrate basic interactions with a normalized database structure. The specific SQL syntax might require adjustments based on your actual database schema, including table names, column types, and constraints.

Figure 32: reading and combining datasets

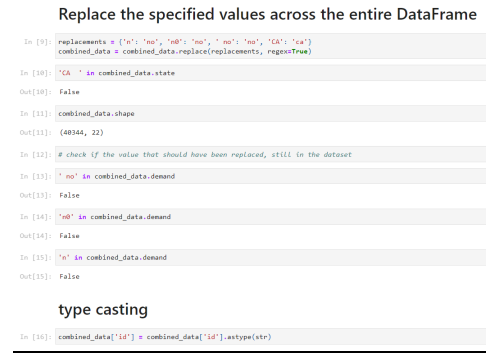

Figure 33: data cleaning and type casting of id column

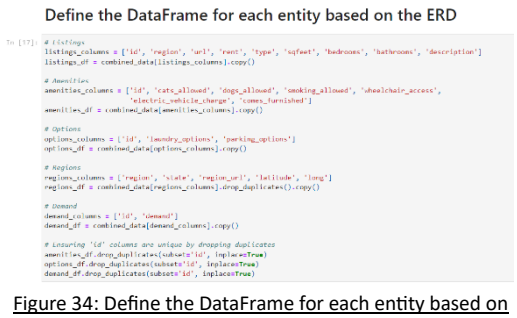

Figure 34: Define the DataFrame for each entity based on the ERD

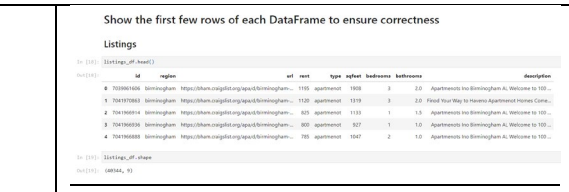

Figure 35: listings table

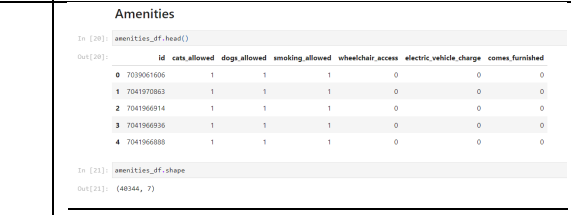

Figure 36: amenities table

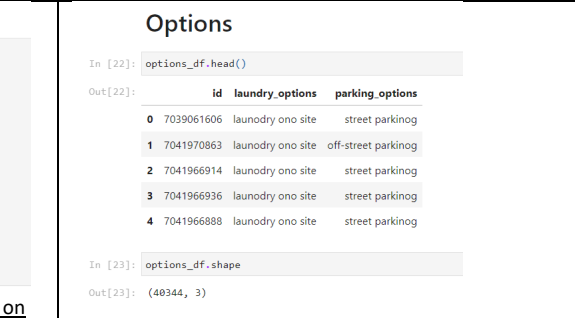

Figure 37: options table

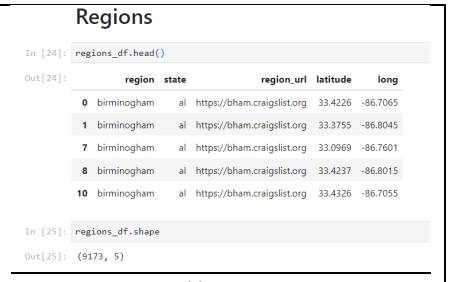

Figure 38: regions table

Figure 39: demand table

Create tables in duckDB database

```
In [28]: # Establish a connection to DuckDB
conn = duckdb.connect(database=':memory:', read_only=False)

# Create the SQL tables in DuckDB
create_table_queries = {
    'listings': '''
        CREATE TABLE listings (
            id VARCHAR,
            region VARCHAR,
            url VARCHAR,
            rent BIGINT,
            type VARCHAR,
            sqfeet BIGINT,
            bedrooms BIGINT,
            bathrooms FLOAT,
            description TEXT
        );
    ''',
    'amenities': '''
        CREATE TABLE amenities (
            id VARCHAR,
            cats_allowed BOOLEAN,
            dogs_allowed BOOLEAN,
            smoking_allowed BOOLEAN,
            wheelchair_access BOOLEAN,
            electric_vehicle_charge BOOLEAN,
            comes_furnished BOOLEAN
        );
    ''',
    'options': '''
        CREATE TABLE options (
            id VARCHAR,
            laundry_options VARCHAR,
            parking_options VARCHAR
        );
    ''',
    'regions': '''
        CREATE TABLE regions (
            region VARCHAR,
            state VARCHAR,
            region_url VARCHAR,
            latitude FLOAT,
            longitude FLOAT
        );
    ''',
    'demand': '''
        CREATE TABLE demand (
            id VARCHAR,
            demand VARCHAR
        );
    '''
}

# Execute the create table queries
for table, query in create_table_queries.items():
    conn.execute(query)

# Confirm that tables were created
conn.execute("SHOW TABLES").fetchall()
```

Out[28]: [('amenities',), ('demand',), ('listings',), ('options',), ('regions',)]

Figure 40: Create tables in duckDB database

Present sample SQL for the database

Inserting a New Line of Data

```
In [40]: # Insert into Regions
conn.execute("""
    INSERT INTO Regions (region, state, region_url, latitude, long)
    VALUES ('San Francisco', 'ca', 'https://sfbay.craigslist.org', 37.7749, -122.4194);
""")
```

Out[40]: <duckdb.duckdb.DuckDBPyConnection at 0x167eb219530>

Figure 41: inserting a New Line of Data 1 (For data insertion operations like (INSERT INTO statements), DuckDB doesn't return a result set that can be directly displayed as a pandas DataFrame because these operations do not produce output rows but instead modify the database state.)

```
In [41]: # Insert into Listings
conn.execute("""
    INSERT INTO Listings (id, url, region, rent, sqfeet, bedrooms, bathrooms, description)
    VALUES (
        123456789,
        'https://sfbay.craigslist.org/apa/d/san-francisco-spacious-one-bedroom/123456789.html',
        'San Francisco', 950, 700, 1, 1,
        'Spacious one-bedroom apartment in the heart of the city. Close to public transportation and parks.'
    );
""")
```

Out[41]: <duckdb.duckdb.DuckDBPyConnection at 0x167eb219530>

```
In [42]: # Insert into Amenities
conn.execute("""
    INSERT INTO Amenities (id, cats_allowed, dogs_allowed, smoking_allowed, wheelchair_access, electric_vehicle_charge, comes_furnished)
    VALUES (123456789, TRUE, TRUE, FALSE, TRUE, FALSE, FALSE);
""")
```

Out[42]: <duckdb.duckdb.DuckDBPyConnection at 0x167eb219530>

```
In [43]: # Insert into Options
conn.execute("""
    INSERT INTO Options (id, laundry_options, parking_options)
    VALUES (123456789, 'on-site', 'street parking');
""")
```

Out[43]: <duckdb.duckdb.DuckDBPyConnection at 0x167eb219530>

```
In [44]: conn.execute("""
    INSERT INTO Demand (id, demand) VALUES (123456789, 'High');
""")
```

Out[44]: <duckdb.duckdb.DuckDBPyConnection at 0x167eb219530>

Figure 42: inserting a New Line of Data 2 (For data insertion operations like (INSERT INTO statements), DuckDB doesn't return a result set that can be directly displayed as a pandas DataFrame because these operations do not produce output rows but instead modify the database state.)

Extracting Description for Specific Properties

```
In [46]: query = """
    SELECT distinct(L.description)
    FROM listings L
    JOIN amenities A ON L.id = A.id
    JOIN regions R ON L.region = R.region
    WHERE L.rent <= 1000 AND A.cats_allowed = 1 AND A.dogs_allowed = 1 AND R.state = 'ca';
    """
description = pd.read_sql_query(query, conn)
```

```
C:\Users\kench\AppData\Local\Temp\ipykernel_21860\2911187581.py:8: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection.
Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  description = pd.read_sql_query(query, conn)
```

```
In [47]: description
```

Out[47]:

	description
0	Cleano 1 b/r 1 ba.apartmenot. Northridge Apa...
1	Features: - Openo, spacious floor planos - P...
2	Hello.. Lookinog for a friendly house for a ...
3	Mirage Apartmenots inovites you to joino us fo...
4	This spacious 1 bedroom, 2 bathroom 1650+ sq f...
...	...
453	Photos showno from a differenot apartmenot. Pl...
454	At Castle Apartmenots we take pride ino our wo...
455	This is a beautiful onoe bedroom onoe bath apa...
456	Save your moneoy by movinog inoto this spaciou...
457	Close to HWY 1, enod unoit withnoo uneit above...

458 rows × 1 columns

Figure 43: output of sample sql statement to extract Description for Specific Properties

Extracting Average Rental Value for Each State

```
In [48]: query = """
SELECT R.state, AVG(L.rent) AS average_rent
FROM listings L
JOIN regions R ON L.region = R.region
GROUP BY R.state
ORDER BY average_rent DESC;

"""
avg_rental_value = pd.read_sql_query(query, conn)
```

```
C:\Users\kench\AppData\Local\Temp\ipykernel_21860\2969542546.py:9: UserWarning: pandas o
Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  avg_rental_value = pd.read_sql_query(query, conn)
```

```
In [49]: avg_rental_value
```

Out[49]:

	state	average_rent
0	ca	4032.352424
1	noh	1718.620253
2	co	1630.459459
3	or	1526.247839
4	ak	1134.828151
5	az	1042.868304
6	nod	965.391548
7	oh	946.483843
8	al	880.620456
9	ar	860.294982
10	ok	775.772234

Figure 44: output of sample sql statement to extract Average Rental Value for Each State