University of Michigan – Dearborn

CIS 556 Database Systems Fall 2024

Project Name: Analyzing Real Estate Trends Using Zillow Data

Project Members	Email
Meghana Basavanna	bmeghanaumich.edu
Nikhil Shankar	
Prkriti Puri	ppurirti@umich.edu
Shilpee Awasthi	shilpee@umich.edu

Submission to: Prof. Niccolo Meneghetti

Team Composition and Responsibilities:

- Meghana Basavanna: Clean datasets, develop the relational schema, load data for Zillow Home Value Index(ZHVI), handle complex SQL queries, optimize database performance, and contribute to the final report.
- Nikhil Shankar: Clean datasets, assist in relational schema design, load data for the Zillow Home Value Forecast(ZHVF), write advanced SQL queries, optimize indexing, and contribute to the final report.
- Prkriti Puri: Clean datasets, design the ER diagram, load data for Sales Data, Write SQL queries, and contribute to the final report.
- Shilpee Awasthi: Clean datasets, load data for Market Heat Index, create relational algebra, write sql queries, and contribute to the final report.

Project Specification

Context: This project aims to analyze and optimize real estate market data using Zillow's housing datasets. By utilizing data from the Zillow home value Index(ZHVI), Zillow home value forecast (ZHVF), Sales data and Market heat index, the project seeks to monitor real estate trends, predict future home value, and analyze sales prices across geographic regions. The unlimited goal is to provide actionable insights for buyers, investors, and real estate professionals. Using PostgreSQL, the project will focus on database management, including schema design, data import, and query optimization. The work involves designing a comprehensive relational schema, creating ER Diagrams, and employing indexing and query optimization techniques to improve performance. Expected outcomes include a well structured database, detailed market insights, and a final report documenting the findings and methodologies. The project aligns with academic objectives by incorporating topics such as ER diagrams, SQL, performance profiling, and relational algebra, offering practical experience in database and data analysis process.

Problem Statement: The problem this project aims to solve is the difficulty in understanding and predicting real estate market trends, which are influenced by many changing factors like supply, demand, and location. Without accurate and comprehensive data, it's hard for homebuyers, investors, and real estate professionals to make good decisions. This project tackles the issue by organizing and analyzing Zillow's data to provide clear and helpful insights into home values, sales trends, and market conditions.

Objectives: The main objective of this project is to understand the real estate trends by analyzing past and current data on home values and sales across different areas and to predict future home values over short and long periods. It also aims to study the balance between housing supply and demand using the Market Heat Index to provide insights into market conditions. The project will give Helpful information to buyers and sellers for better decision making. Additionally it focuses on a well organised database that works efficiently with large datasets, ensuring the data is clean and accurate, and summarizing all finds and methods in this final report.

Datasets Used: Source: Housing Data - Zillow Research

1. **Zillow Home Value Index (ZHVI)**: Contains data on typical home values by region and housing type.
2. **Home Value Forecast (ZHVF)**: Includes predictions for month-ahead, quarter-ahead, and year-ahead home values.
3. **Sales Data**: Tracks home sale prices and total dollar value by geography.
4. **Market Heat Index**: Captures the balance of supply and demand in housing markets.

Decisions made in the design stage

Data Modeling and Schema: The database was designed with a relational model, creating separate tables for each entity (`sales_data`, `zillow_home_index`, `market_index_data`, `growth_data`). This structure enables clear separation of data and efficient querying.

Relationships and Foreign Keys: Foreign keys were used to establish relationships between tables, particularly using `regionid` as a common reference. This ensures referential integrity and connects related data (e.g., sales data linked to regions).

Indexing Strategy: Indexes were applied to frequently queried columns, such as `regionid` and `date`, to improve performance. Composite indexes were considered for columns frequently used together in queries (e.g., `RegionId` and `Date`).

Data Types: Appropriate data types were chosen for each column, ensuring efficient storage and accurate representation of data. For example, `INTEGER` for numerical data, `FLOAT` for values like `zhvi`, and `DATE` for time-related data.

Time-Series Data Handling: Date columns were indexed to optimize queries involving time-based analysis. This is crucial for operations that filter or group data by date.

Query Optimization: The schema was designed to optimize common reporting queries, with indexing and relationships in place to ensure efficient joins and aggregations.

Data Integrity: Constraints (e.g., `NOT NULL`, `PRIMARY KEY`) were used to ensure data accuracy and consistency across tables, preventing invalid or incomplete data.

Dataset Transformation

To load the Zillow Home dataset into PostgreSQL, We follow these steps:

Preparing your environment, inspecting and cleaning the dataset, creating the database and table, and finally loading the data into PostgreSQL.

Importing the required packages.

Import package pandas in IDLE

import pandas as pd.

1. **Reading the dataset**

```
df = pd.read_csv('/Path-to-dataset.csv')
```

Example

```
df = pd.read_csv('/Users/meghanab/Downloads/Improved_Cleaned_Megana_Dataset (1).csv')
```

2. **Cleaning the data for null and missing values**

Identify columns and rows that are majority empty (e.g., more than 50% missing values) threshold = 0.5

```
columns_to_drop = df.columns[df.isnull().mean() > threshold]
rows_to_drop = df.index[df.isnull().mean(axis=1) > threshold]
```

Drop identified columns and rows

```
df_cleaned = df.drop(columns=columns_to_drop).drop(index=rows_to_drop)
```

Replace remaining null or missing values with calculated values (mean for numeric columns)

```
for col in df_cleaned.select_dtypes(include=['float64', 'int64']).columns:
        if df_cleaned[col].isnull().sum() > 0:
        df_cleaned[col].fillna(df_cleaned[col].mean(), inplace=True)
```

Replace missing values in categorical columns with the most frequent value

```
for col in df_cleaned.select_dtypes(include=['object']).columns:
        if df_cleaned[col].isnull().sum() > 0:
        most_frequent_value = df_cleaned[col].mode()[0]
        df_cleaned[col].fillna(most_frequent_value, inplace=True)
```

Display the cleaned dataset for review

```
import ace_tools as tools;
tools.display_dataframe_to_user(name="Cleaned Metro Market Dataset Without Altering
Values", dataframe=df_cleaned)
```

3. **Unpivot the dataset**

To Simplifying Data Analysis and Visualization we need to melt dataset Zillow home value index, Zillow forecast and sales datasets

Unpivoting (or melting) is the process of transforming data from a wide format to a long format. This is useful for making the dataset easier to analyze and work with, especially in SQL databases and analysis tools.

Python Script to unpivot Zillow home value index

pd.melt(df, id_vars=['RegionID', 'SizeRank', 'RegionName', 'RegionType', 'StateName'], var_name='Date', value_name='ZHVI')

# Python Script to unpivot market heat index
pd.melt(df, id_vars=['RegionID', 'SizeRank', 'RegionName', 'RegionType', 'StateName'], var_name='Date', value_name='Market_heat_Index')

# Python Script to unpivot Sales data
pd.melt(df, id_vars=['RegionID', 'SizeRank', 'RegionName', 'RegionType', 'StateName'], var_name='Date', value_name='Sale_count')

4. **Extract the data where year is >= 2018**
   For the Zillow home index, growth data and sales data

   Step 1- Format the date field to DateTime if already not done
   Command - pd.to_datetime(df['Date'], format='%d-%m-%Y')

   Step 2 – Extract the rows having year >= 2018
   Command - df[df['Date'].dt.year >= 2018]

   We can view the dataset which are modified by
   print(df.head())

5. **Saving the dataset**
   After the dataset is formatted as required save it
   Save Command - df.to_csv('/path-to-save.csv', index=False)

   For eg to save it in Documents use below command.
   df.to_csv('/Users/meghanab/Documents/cleaned_zillow_data_meghana.csv', index=False)

6. **Loading the dataset to Postgress SQL.**
   Loading a dataset into PostgreSQL involves two main stages: creating the appropriate table structure and transferring the cleaned data into the database.

   Once the table has been created, the next step is to populate it with data. This involves transferring the cleaned dataset from your local machine (or wherever it is stored) into the PostgreSQL database using copy command.

Command - COPY Table_name FROM 'path.csv'
DELIMITER ',' CSV HEADER;

Copy code for sales_data

```
COPY sales_data FROM
'/Users/meghanab/Documents/cleaned_zillow_sales_data_prkrithi.csv'
DELIMITER ',' CSV HEADER;
```

Copy code for market_index_data

```
COPY market_index_data FROM
'/Users/meghanab/Documents/cleaned_zillow_market-index_data_shilpee.csv'
DELIMITER ',' CSV HEADER;
```

Copy code for growth_data

```
COPY growth_data FROM
'/Users/meghanab/Documents/cleaned_forecastData_nikhil.csv'
DELIMITER ',' CSV HEADER;
```

Copy code zillow_home_index

```
COPY zillow_home_index FROM
'/Users/meghanab/Documents/cleaned_zillow_data_meghana.csv'
DELIMITER ',' CSV HEADER;
```

Index performance demonstration

Indexing is a critical technique used to optimize database performance, especially for large datasets. The primary purpose of indexing is to speed up query performance, allowing the database to retrieve data more efficiently. Here's a breakdown of why indexing is important:

Some of the indexing queries used are in our project are as below

```
CREATE INDEX idx_zillow_date ON zillow_home_index (date);
CREATE INDEX idx_growth_basedate ON growth_data (basedate);
CREATE INDEX idx_market_regionid_date ON market_index_data (regionid, date);
CREATE INDEX idx_sales_regionname ON sales_data (RegionName);
CREATE INDEX idx_sales_salecount ON sales_data (Sale_count);
```

Examples demonstrating the Indexing

Before Indexing

```
postgres=# EXPLAIN ANALYZE
WITH LatestZHVI AS (
    SELECT regionid, zhvi AS current_zhvi, date
    FROM zillow_home_index
    WHERE date = '2024-10-31'
),
GrowthForecast AS (
    SELECT regionid, regionname, yearforecast, statename
    FROM growth_data
    WHERE basedate = '2024-10-31'
    AND yearforecast IS NOT NULL
)
SELECT g.regionid, g.regionname, g.statename, z.current_zhvi, g.yearforecast,
       z.current_zhvi * (1 + (g.yearforecast / 100)) AS forecasted_zhvi
FROM GrowthForecast g
JOIN LatestZHVI z ON g.regionid = z.regionid
ORDER BY g.yearforecast DESC;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=2382.51..2385.48 rows=1187 width=44) (actual time=44.676..44.863 rows=1284 loops=1)
   Sort Key: growth_data.yearforecast DESC
   Sort Method: quicksort  Memory: 155kB
   ->  Hash Join  (cost=32.38..2321.90 rows=1187 width=44) (actual time=23.384..43.340 rows=1284 loops=1)
         Hash Cond: (zillow_home_index.regionid = growth_data.regionid)
         ->  Seq Scan on zillow_home_index  (cost=0.00..2264.30 rows=1187 width=12) (actual time=21.584..41.095 rows=1292 loops=1)
               Filter: (date = '2024-10-31'::date)
               Rows Removed by Filter: 104652
         ->  Hash  (cost=21.19..21.19 rows=895 width=28) (actual time=1.485..1.486 rows=895 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 65kB
               ->  Seq Scan on growth_data  (cost=0.00..21.19 rows=895 width=28) (actual time=0.257..1.080 rows=895 loops=1)
                     Filter: ((yearforecast IS NOT NULL) AND (basedate = '2024-10-31'::date))
 Planning Time: 4.021 ms
 Execution Time: 45.697 ms
(14 rows)
```

In the above Screenshot we can see that the execution time is given as 45.697 ms without adding any indexing to the tables.

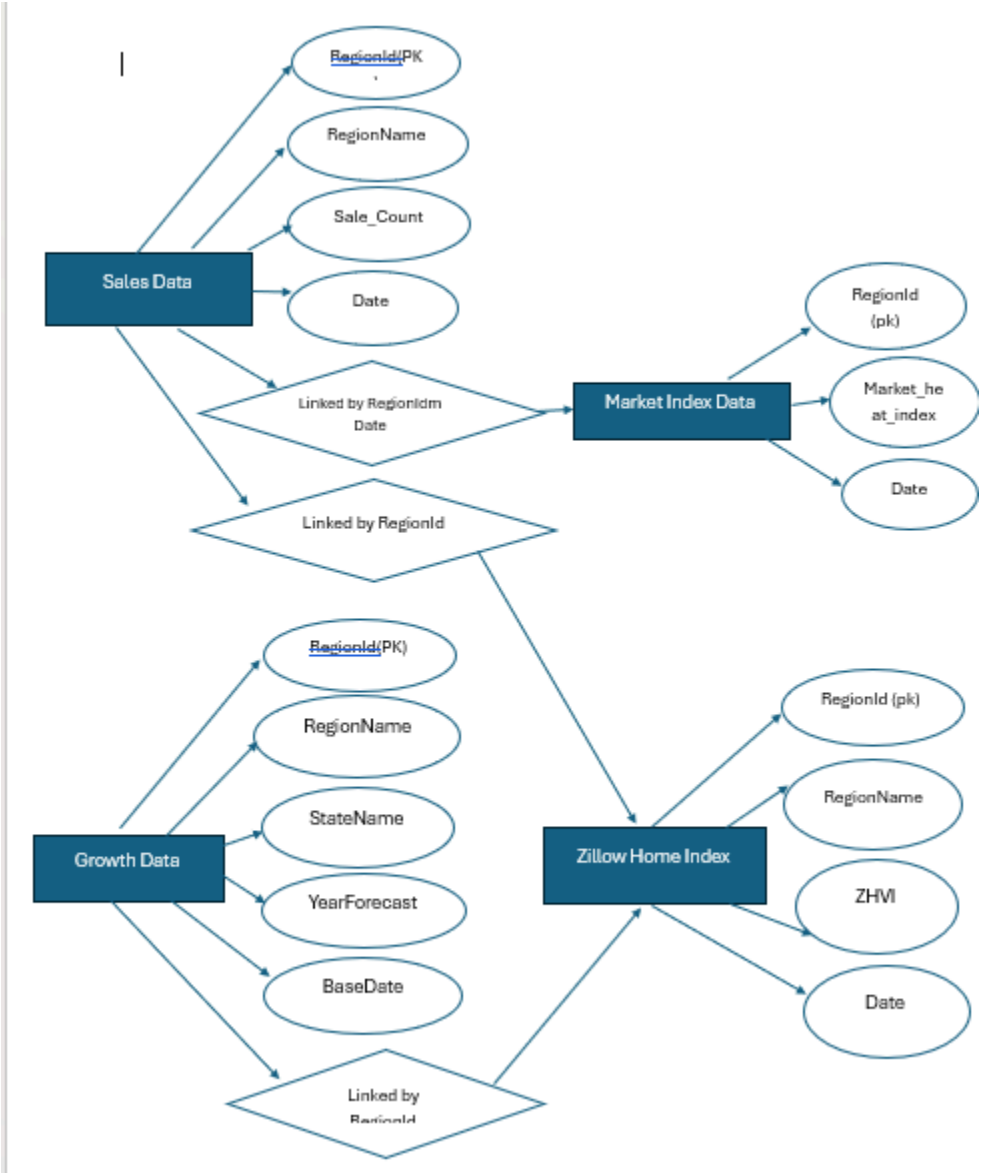
After Indexing

```
postgres=# EXPLAIN ANALYZE
WITH LatestZHVI AS (
    SELECT regionid, zhvi AS current_zhvi, date
    FROM zillow_home_index
    WHERE date = '2024-10-31'
),
GrowthForecast AS (
    SELECT regionid, regionname, yearforecast, statename
    FROM growth_data
    WHERE basedate = '2024-10-31'
    AND yearforecast IS NOT NULL
)
SELECT g.regionid, g.regionname, g.statename, z.current_zhvi, g.yearforecast,
       z.current_zhvi * (1 + (g.yearforecast / 100)) AS forecasted_zhvi
FROM GrowthForecast g
JOIN LatestZHVI z ON g.regionid = z.regionid
ORDER BY g.yearforecast DESC;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=1140.54..1143.51 rows=1187 width=44) (actual time=5.039..5.165 rows=1284 loops=1)
   Sort Key: growth_data.yearforecast DESC
   Sort Method: quicksort  Memory: 155kB
   ->  Hash Join  (cost=49.87..1079.92 rows=1187 width=44) (actual time=2.211..3.159 rows=1284 loops=1)
         Hash Cond: (zillow_home_index.regionid = growth_data.regionid)
         ->  Bitmap Heap Scan on zillow_home_index  (cost=17.49..1022.32 rows=1187 width=12) (actual time=0.821..1.090 rows=1292 loops=1)
               Recheck Cond: (date = '2024-10-31'::date)
               Heap Blocks: exact=14
               ->  Bitmap Index Scan on idx_zillow_date  (cost=0.00..17.20 rows=1187 width=0) (actual time=0.491..0.492 rows=1292 loops=1)
                     Index Cond: (date = '2024-10-31'::date)
         ->  Hash  (cost=21.19..21.19 rows=895 width=28) (actual time=1.368..1.368 rows=895 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 65kB
               ->  Seq Scan on growth_data  (cost=0.00..21.19 rows=895 width=28) (actual time=0.031..0.466 rows=895 loops=1)
                     Filter: ((yearforecast IS NOT NULL) AND (basedate = '2024-10-31'::date))
 Planning Time: 6.730 ms
 Execution Time: 5.655 ms
(16 rows)

postgres=#
```

From the above Screenshot we can see that the execution time reduced to 5.655 after adding the above mentioned indexing queries.

ER Diagram


RegionId(PK
RegionName
Sale_Count
Date
Sales Data
Linked by RegionIdm
Date
Market Index Data
RegionId
(pk)
Market_he
at_index
Date
Linked by RegionId
RegionId(PK)
RegionName
StateName
YearForecast
BaseDate
Growth Data
Linked by
RegionId
Zillow Home Index
RegionId (pk)
RegionName
ZHVI
Date

Methodology:

Data Cleaning and Preparation:

Data cleaning is a crucial step of preparing raw datasets for analysis. For the Zillow datasets, this process ensures data quality, consistency and usability. The cleaning steps involve handling missing values, transforming data formats, and filtering relevant records. We followed these steps in data cleaning:

Step 1: Inspect the dataset: In this step understand the dataset structure and identify issues such as null values, duplicates, and

Tool used: Python's pandas library, Import package pandas in IDLE (import pandas as pd).

Step 2: Handle missing values: In this step, Rows with critical null values are cleaned based on context.

- Identify columns with missing data.
- Impute null values.

Step 3: Remove Duplicates: In this step, Ensure no redundant rows exist in the dataset.

Step 4: Unpivot wide-format data:

In this step, transformed datasets from a wide format, facilitating easier SQL queries and visualization.To Simplifying Data Analysis and Visualization we need to melt the **Zillow home value index, Market Heat Index and sales datasets**.

Unpivoting (or melting) is the process of transforming data from a wide format to a long format. This is useful for making the dataset easier to analyze and work with, especially in SQL databases and analysis tools.

Step 5: Extract relevant records: Filtered data to include only records from 2018 onwards. For the Zillow home index, growth data and sales data:

- **Format the date field to DateTime if already not done**
  Command - pd.to_datetime(df['Date'], format= '%d-%m-%Y')
- **Extract the rows having year >= 2018**
  Command - df[df['Date'].dt.year >= 2018]

Step 6: Save cleaned data: Stored the cleaned dataset for further use in PostgreSQL.

Schema and Table creation for datasets:

After preparing and cleaning the Zillow datasets, the next critical step is to create structured tables in PostgreSQL. These tables provide a well-defined schema to store the data, ensuring efficient storage, retrieval and analysis.

To **designed and created four database tables** in PostgreSQL based on cleaned datasets:

- **growth_data:** Stores forecasted growth data for regions
- **market_index_data:** Holds market heat index data
- **sales_data:** Contains sales date by region.
- **zillow_home_index:** Represents Zillow Home Value Index(ZHVI).

Database Schemas:

- **growth_data table:** This table includes 9 schemas for managing the project.
    - rigionid (INT): unique identifier for the region (Primary Key).
    - sizerank (INT): size rank of the region.
    - monthforecast (NUMERIC(5,2)): monthly growth forecast.
    - quarterforecast (NUMERIC(5,2)): quarterly growth forecast.
    - yearforecast (NUMERIC(5,2)): yearly growth forecast.
    - basedate (DATE): date for which the forecast is based (Primary Key).
    - regionname(VARCHAR(255)): name of region.
    - regiontype (VARCHAR(255)): type of region (e.g. city, state).
    - Statename (VARCHAR(255)): name of the state.

We converted the above conceptual design into following SQL schema:

```
CREATE TABLE growth_data (
regionid INT NOT NULL,
sizerank INT NOT NULL,
monthforecast NUMERIC(5, 2),
quarterforecast NUMERIC(5, 2),
yearforecast NUMERIC(5, 2),
basedate DATE NOT NULL,
regionname VARCHAR(255),
regiontype VARCHAR(255),
statename VARCHAR(255),
PRIMARY KEY (regionid, basedate)
```

);

- **market_index_data table:** This table includes 7 schemas.
    - regionid (INT): Unique identifier for the region.
    - sizerank (INT): Size rank of the region.
    - date (DATE): Date of the market index data.
    - market_heat_index (NUMERIC(5,2)): Market heat index value.
    - regiontype (VARCHAR(255)): Type of the region (e.g., city, state, country).
    - regionname (VARCHAR(255)) Name of the region.
    - statename (VARCHAR(255)) Name of the state.

We converted the above conceptual design into following SQL schema:

```
CREATE TABLE market_index_data (
regionid INT NOT NULL,
sizerank INT NOT NULL,
date DATE NOT NULL,
market_heat_index NUMERIC(5, 2),
regiontype VARCHAR(255),
regionname VARCHAR(255),
statename VARCHAR(255),
PRIMARY KEY (regionid, date)
);
```

- **Sales_data table:** This table also includes 7 schemas.
    - regionid (INT): Unique identifier for the region.
    - sizerank (INT): Size rank of the region.
    - date (DATE): Date of the sales data.
    - sale_count (INT): Number of sales.
    - regiontype (VARCHAR(255)): Type of the region (e.g., city, state, country).
    - regionname (VARCHAR(255)): Name of the region.
    - statename VARCHAR(255) Name of the state.

We converted the above conceptual design into following SQL schema:

```
CREATE TABLE market_index_data (
regionid INT NOT NULL,
sizerank INT NOT NULL,
date DATE NOT NULL,
market_heat_index NUMERIC(5, 2),
regiontype VARCHAR(255),
regionname VARCHAR(255),

statename VARCHAR(255),
PRIMARY KEY (regionid, date)
);
```

- **Zillow_home_index table:**
  - regionid (INT): Unique identifier for the region.
  - sizerank (INT): Size rank of the region.
  - date (DATE): Date of the ZHVI data.
  - zhvi (NUMERIC(15,2)): Zillow Home Value Index.
  - regiontype (VARCHAR(255)): Type of the region (e.g., city, state, country).
  - regionname (VARCHAR(255)): Name of the region.
  - statename (VARCHAR(255)): Name of the state.

We converted the above conceptual design into following SQL schema:

```
CREATE TABLE zillow_home_index (
regionid INT NOT NULL,
sizerank INT NOT NULL,
date DATE NOT NULL,
zhvi NUMERIC(15, 2),
regiontype VARCHAR(255),
regionname VARCHAR(255),
statename VARCHAR(255),
PRIMARY KEY (regionid, date)
 );
```

Key Insights and Facts Extracted from the Dataset

1.Monthly sales growth rate.

Purpose:
The query calculates the month-over-month growth rate in home sales for each region. This growth rate indicates how the number of homes sold changes from one month to the next.

```
WITH SalesWithPrevious AS (
	SELECT
	sd1.RegionId,
	sd1.Date,
	sd1.Sale_count,
	LAG(sd1.Sale_count) OVER (PARTITION BY sd1.RegionId ORDER BY sd1.Date) AS
PrevMonthSales
	FROM
	sales_data sd1
)
SELECT
	RegionId,
	Date,
	CASE
	WHEN PrevMonthSales = 0 THEN 0
	WHEN PrevMonthSales IS NULL THEN NULL
	ELSE ROUND(((Sale_count - PrevMonthSales) / PrevMonthSales::numeric) * 100, 2)
	END AS MonthlyGrowthRate
FROM
	SalesWithPrevious
ORDER BY
	RegionId, Date;
```

Significance

The monthly sales growth rate metric reveals changes in housing sales volumes across regions, helping identify patterns and anomalies. This data aids real estate professionals in advising

clients, policymakers in guiding housing policies, and stakeholders in forecasting trends for future planning. Comparing growth rates across regions provides insights into market dynamics, facilitating strategic decisions and resource allocation.

2.**Regions with Market Heat Index Favorable to Buyers and Higher than Average Sales**

Description:
This query identifies regions where the market heat index is favorable to buyers (assumed to be less than 100) and the average monthly sales are higher than the overall average.

```
SELECT
    sd.RegionName,
    AVG(sd.Sale_count) AS AvgMonthlySales,
    AVG(mi.Market_heat_Index) AS AvgMarketHeatIndex
FROM
    sales_data sd
JOIN
    market_index_data mi ON sd.RegionId = mi.RegionId AND sd.Date = mi.Date
WHERE
    mi.Market_heat_Index < 100 -- Assuming less than 100 is favorable to buyers
GROUP BY
    sd.RegionName
HAVING
    AVG(sd.Sale_count) > (SELECT AVG(Sale_count) FROM sales_data)
ORDER BY
    AvgMonthlySales DESC;
```

Significance of the Data: This query is significant for identifying regions where the real estate market is more favorable to buyers, yet still experiencing high sales activity. This information is valuable for potential buyers looking for advantageous conditions, real estate professionals advising clients, and policymakers monitoring market dynamics. By focusing on regions with a lower market heat index but higher-than-average sales, stakeholders can identify attractive markets for investment or policy intervention.

3 Regions with Consistent Sales Growth

This query identifies regions that have experienced exactly three consecutive years of sales growth. It outputs the RegionId, Year, TotalSales, and Growth.

```
WITH YearlySales AS (
	SELECT
	RegionId,
	EXTRACT(YEAR FROM Date) AS Year,  -- Extracting the year from Date column
	SUM(Sale_count) AS TotalSales
	FROM
	sales_data
	GROUP BY
	RegionId, Year
),
SalesGrowth AS (
	SELECT
	RegionId,
	Year,
	TotalSales,
	LAG(TotalSales) OVER (PARTITION BY RegionId ORDER BY Year) AS
PrevYearSales
	FROM
	YearlySales
),
ConsistentGrowth AS (
	SELECT
	RegionId,
	Year,
	TotalSales,
	PrevYearSales,
	(TotalSales - PrevYearSales) AS Growth,
	-- Calculate the consecutive growth streak using the window function
	COUNT(*) OVER (PARTITION BY RegionId ORDER BY Year ROWS BETWEEN 2
PRECEDING AND CURRENT ROW) AS GrowthStreak
	FROM
	SalesGrowth
	WHERE
	PrevYearSales IS NOT NULL
)
-- Now, filter the result in the outer query
SELECT
```

```
    RegionId,
    Year,
    TotalSales,
    Growth
FROM
    ConsistentGrowth
WHERE
    Growth > 0
    AND GrowthStreak = 3  -- Filter by GrowthStreak
ORDER BY
    RegionId, Year;
```

Significance

Identifying regions with consistent sales growth is crucial for understanding stable market trends and making informed decisions. This information can guide investors towards regions with reliable growth, helping in strategic planning and resource allocation. Consistent growth can indicate a healthy market with sustainable demand, making these regions attractive for long-term investments and development projects.

4 Forecasted ZHVI for November 2025 with Current ZHVI for October 2024

Description:

This query calculates the forecasted Zillow Home Value Index (ZHVI) for November 2025 and includes the current ZHVI for October 2024. The results are sorted in descending order of the forecasted value, showing regions with the highest forecast at the top.

```
WITH LatestZHVI AS (
    SELECT
    regionid,
    zhvi AS current_zhvi,
    date
    FROM
    zillow_home_index
    WHERE
    date = '2024-10-31'  -- Assuming we are working with October 2024 data for current
ZHVI
),
GrowthForecast AS (
    SELECT
    regionid,
```

```
        regionname,
        yearforecast,
        statename
        FROM
        growth_data
        WHERE
        basedate = '2024-10-31'  -- Using data based on October 2024
        AND yearforecast IS NOT NULL  -- Only include regions with forecasted growth
)
SELECT
        g.regionid,
        g.regionname,
        g.statename,
        z.current_zhvi,
        g.yearforecast,
        z.current_zhvi * (1 + (g.yearforecast / 100)) AS forecasted_zhvi  -- Calculating
forecasted ZHVI for November 2025
FROM
        GrowthForecast g
JOIN
        LatestZHVI z ON g.regionid = z.regionid
ORDER BY
        g.yearforecast DESC;
```

Significance of the Data:
This query is significant as it helps identify regions with the highest forecasted growth in home values over the next year. This information is crucial for investors, real estate professionals, and policy makers as it provides insights into potential market trends and areas of high demand. By understanding these forecasts, stakeholders can make informed decisions regarding investments, development projects, and policy adjustments to support housing market stability and growth.

5 Average Home Value Index by Market Heat Index Category

Description: This query categorizes regions into three market heat index categories ("High", "Medium", "Low") based on their market heat index values and calculates the average home value index (ZHVI) for each category. The results are sorted by the market heat index category.

```
SELECT
        CASE
        WHEN mi.Market_heat_Index >= 150 THEN 'High'
```

```
        WHEN mi.Market_heat_Index >= 100 AND mi.Market_heat_Index < 150 THEN
'Medium'
        ELSE 'Low'
        END AS MarketHeatCategory,
        AVG(zh.ZHVI) AS AvgHomeValueIndex
FROM
        zillow_home_index zh
JOIN
        market_index_data mi ON zh.RegionID = mi.RegionID AND zh.Date = mi.Date
GROUP BY
        MarketHeatCategory
ORDER BY
        MarketHeatCategory;
```

Significance of the Data:
This query is significant because it helps in understanding the relationship between the market heat index and home values across different regions. By categorizing regions based on their market heat index, stakeholders can gain insights into how market favorability affects home values. This information is valuable for real estate professionals, investors, and policy makers as it provides a clear picture of market dynamics, enabling them to make informed decisions regarding investments, market strategy, and policy formulation. For example, regions with a high market heat index and high average home values may indicate a strong seller's market, while regions with a low market heat index and lower home values may indicate a buyer's market.

Regions with Declining Growth but High Market Activity

Description: This query identifies regions that exhibit declining growth in home values but still maintain high market activity. Declining growth is indicated by a negative year-over-year forecast, while high market activity is represented by a high current ZHVI compared to the average ZHVI.

```
WITH GrowthData AS (
        SELECT
        gd.RegionID,
        gd.RegionName,
        gd.StateName,
        gd.YearForecast,
        zhvi.ZHVI AS CurrentZHVI
        FROM
        growth_data gd
```

```
        JOIN
        zillow_home_index zhvi ON gd.RegionID = zhvi.RegionID
        WHERE
        zhvi.Date = '2024-10-31'  -- Current ZHVI as of October 2024
),
DecliningGrowth AS (
        SELECT
        RegionID,
        RegionName,
        StateName,
        YearForecast,
        CurrentZHVI,
        CASE
        WHEN YearForecast < 0 THEN 'Declining'
        ELSE 'Growing'
        END AS GrowthStatus
        FROM
        GrowthData
)
SELECT
        RegionID,
        RegionName,
        StateName,
        CurrentZHVI,
        YearForecast
FROM
        DecliningGrowth
WHERE
        GrowthStatus = 'Declining'  -- Focus on regions with negative forecasted growth
        AND CurrentZHVI > (SELECT AVG(ZHVI) FROM zillow_home_index WHERE Date
= '2024-10-31')  -- High market activity
ORDER BY
        CurrentZHVI DESC;
```

Significance of the Data:

This query helps identify regions where market activity remains high despite a forecasted decline in home values. Such regions may present unique investment opportunities or challenges. For example, high market activity could indicate strong demand or significant buyer interest, while the declining growth forecast might suggest underlying issues or an impending market

correction. Real estate professionals, investors, and policy makers can use this information to make informed decisions about resource allocation, market strategies, and policy interventions. Identifying these regions allows stakeholders to better understand market dynamics and potential future trends.

Top 5 Regions with the Highest Cumulative ZHVI and Sales

This query calculates the cumulative ZHVI (Zillow Home Value Index) and Sales for each region and selects the top 5 regions with the highest total values of both metrics combined.

```
WITH CumulativeSales AS (
	SELECT
	sd.RegionId,
	sd.RegionName,
	SUM(sd.Sale_count) AS TotalSales
	FROM
	sales_data sd
	GROUP BY
	sd.RegionId, sd.RegionName
),
CumulativeZHVI AS (
	SELECT
	zhvi.RegionID,
	zhvi.RegionName,
	SUM(zhvi.ZHVI) AS TotalZHVI
	FROM
	zillow_home_index zhvi
	WHERE
	zhvi.Date BETWEEN '2018-01-01' AND '2024-12-31'  -- Calculate over a 6-year period
	GROUP BY
	zhvi.RegionID, zhvi.RegionName
)
SELECT
	cs.RegionId,
	cs.RegionName,
	cs.TotalSales,
	cz.TotalZHVI,
	(cs.TotalSales + cz.TotalZHVI) AS TotalCombinedValue
FROM
	CumulativeSales cs
JOIN
```

```
    CumulativeZHVI cz ON cs.RegionId = cz.RegionID
ORDER BY
    TotalCombinedValue DESC
LIMIT 5;
```

Significance of the Data:
This query identifies regions with the highest overall market activity, based on both the total sales of homes and the cumulative value of homes (ZHVI). It provides insights into which regions have experienced both high sales volumes and significant increases in home values over time. For real estate investors, developers, and policymakers, this data is crucial for identifying the most active and valuable markets. High combined values indicate regions with strong demand and market stability, which can guide decisions regarding investment, resource allocation, and market forecasting. The top 5 regions identified in the result set are likely to be prime targets for future real estate activities, investments, and strategic initiatives.

The queries mentioned above make use of several advanced SQL features that help in performing complex data manipulations and analysis.

1.**Common Table Expressions (CTEs)**: CTEs modularize complex queries, making them easier to read and maintain, and allow for reusable intermediate result sets, improving query clarity.

2. **Window Functions**: Functions like LAG() and COUNT() enable row-level calculations across related rows, such as comparing current values with previous ones, improving analysis of trends and growth.

3. **Aggregations**: Functions like SUM() and AVG() summarize data, helping to calculate metrics like total sales or average home values, essential for business decision-making.

4. **Subqueries**: Subqueries allow dynamic calculations, like comparing a region's sales to the overall average, enabling more flexible and refined filtering.

5.**Conditional Logic (CASE):** Allows categorization of data (e.g., high/medium/low market heat) based on conditions, transforming raw data into actionable insights.

6.**Joins**: These combine data from multiple tables, essential for comprehensive analysis across different data domains like sales and market heat index.

7. **Date Functions**: Date manipulations like EXTRACT() allow for time-based analysis, critical for trend analysis over specific periods.

8.**HAVING Clause**: Used to filter data after aggregation, refining results based on calculated metrics.

9. **LIMIT and ORDER BY:** These allow ranking and limiting results, showing the top regions based on specific criteria like sales or growth.